DELETE

Оператор *DELETE* удаляет строки из одной или нескольких таблиц. Операторы *DELETE*, применяемые к таблицам, называются иногда *поисковыми удалениями* (search deletes). Оператор *DELETE* также можно использовать совместно с курсорами, тогда он называется *позиционным удалением*.

СУБД	Уровень поддержки
MySQL	Поддерживается с вариациями
Oracle	Поддерживается с вариациями
PostgreSQL	Поддерживается
SQL Server	Поддерживается с ограничениями

Синтаксис SQL2003

```
DELETE FROM { имя_таблицы | ONLY (имя_таблицы) }
[{ WHERE условие_поиска | WHERE CURRENT OF имя_курсора }]
```

Ключевые слова

FROM {имя_таблицы | ONLY (имя_таблицы)}

Указывает таблицу, из которой удаляются строки. Если явно не указано имя схемы, то подразумевается, что таблица находится в текущей схеме. Можно указать также имя представления. Ключевое слово *FROM* обязательно, если только не используется *DELETE...WHERE CURRENT OF*. Если вы не используете ключевое слово *ONLY*, то имя таблицы не заключается в скобки. *ONLY* используется только для объектных таблиц и представлений и ограничивает каскадное удаление записей в таблицах-потомках целевой таблицы. Если использовать слово *ONLY* для обычных таблиц, то оно будет проигнорировано и не вызовет ошибки.

WHERE условие_поиска

Определяет условие поиска удаляемых записей. В *WHERE* допускаются любые корректные выражения. Обычно условие поиска оценивается для каждой записи таблицы перед началом удаления.

WHERE CURRENT OF имя_курсора

Удаляет текущую запись указанного открытого курсора.

Общие правила

Оператор *DELETE* удаляет строки из таблицы или представления. Освобожденное при удалении строк пространство возвращается в базу данных, но не всегда это происходит сразу.

В простейшем виде оператор *DELETE* не содержит фразы *WHERE* и удаляет все записи из таблицы:

```
DELETE FROM sales;
```

Вы можете использовать фразу *WHERE* для определения тех записей, которые следует удалить. В следующем примере все три оператора *DELETE* являются до-

пустимыми, и во всех используется фраза *FROM*, так как они являются поисковыми удалениями:

```
DELETE FROM sales
WHERE qty IS NULL;
DELETE FROM suppliers
WHERE supplierid = 17
  OR companyname = 'Tokyo Traders';
DELETE FROM distributors
WHERE postalcode IN
  (SELECT territorydescription FROM territories);
```

Помните, что в позиционных удалениях фраза *FROM* не используется.

В некоторых случаях вам может требоваться удалять записи из открытого курсора:

```
DELETE titles WHERE CURRENT OF title_cursor;
```

В этом операторе подразумевается, что вы объявили и открыли курсор с названием **title_cursor**. При выполнении оператора *DELETE* удаляется та запись курсора, на которой он находится в этот момент.

Советы и хитрости

Оператор *DELETE* достаточно редко выполняется без фразы *WHERE*, так как в этом случае удаляются все строки таблицы. Перед удалением выполните оператор *SELECT* с тем же фильтром в *WHERE*, это позволит вам убедиться, что вы удаляете те записи, которые нужно.

Если вам требуется удалить все записи из таблицы, то используйте не описанный в ANSI, но весьма распространенный оператор *TRUNCATE TABLE*. В тех базах данных, в которых он поддерживается, оператор *TRUNCATE TABLE* обычно работает быстрее, чем *DELETE*, так как при его выполнении не журналируется удаление каждой записи. При удалении большого числа записей отказ от журналирования значительно экономит время, но при этом делает невозможным откат удаления. Более того, в некоторых базах данных перед выполнением *TRUNCATE TABLE* требуется удалить все внешние ключи.

MySQL

В MySQL есть несколько расширений стандарта ANSI, но не поддерживается фраза *WHERE CURRENT OF*. Синтаксис оператора следующий:

```
DELETE [LOW_PRIORITY] [QUICK] [имя_таблицы[.*][,...]]
{FROM имя_таблицы [.*][,...]|[USING имя_таблицы[.*][,...]]}
[WHERE условие_поиска]
[ORDER BY фраза]
[LIMIT число_строк]
```

Параметры имеют следующие значения:

LOW PRIORITY

Откладывает удаление до того момента, когда другие клиенты закончат чтение таблицы.

QUICK

Предотвращает слияние листьев индекса в процессе удаления.

DELETE имя_таблицы[,...]

Позволяет за один раз удалить строки из нескольких таблиц. Таблицы, перечисленные перед фразой *FROM*, при наличии списка таблиц после *FROM* являются целевыми таблицами: из всех этих таблиц будут удалены все требуемые записи.

FROM имя_таблицы [.]*

Указывает одну или несколько таблиц, из которых удаляются записи. (.* используется для улучшенной совместимости с MS Access.) Если перед *FROM* указан список таблиц, то таблицы после *FROM* используются в операциях соединения и поиска по справочникам.

USING имя_таблицы [.][,...]*

Заменяет одну или несколько таблиц, указанных перед фразой *FROM*, таблицами после фразы *FROM*.

ORDER BY фраза

Определяет порядок, в котором будут удаляться строки. Имеет смысл только при использовании вместе с фразой *LIMIT*.

LIMIT число_строк

LIMIT позволяет ограничить число строк, после удаления которых управление возвращается пользователю.

MySQL позволяет одной командой удалить данные из нескольких таблиц. В следующем примере два оператора *DELETE* функционально эквивалентны:

```
DELETE orders FROM customers, orders
WHERE customers.customerid = orders.customerid
  AND orders.orderdate BETWEEN '19950101' AND '19951231'
DELETE FROM orders USING customers, orders
WHERE customers.customerid = orders.customerid
  AND orders.orderdate BETWEEN '19950101' AND '19951231'
```

В этих примерах мы удаляем все заказы из таблицы **orders**, сделанные клиентами из таблицы **customers** в течение 1995 года. Обратите внимание, что в мультитабличных удалениях нельзя использовать фразы *LIMIT* и *ORDER BY*.

Используя *LIMIT* и *ORDER BY* вы можете удалять записи упорядоченными наборами:

```
DELETE FROM sales
WHERE customerid = 'TORTU'
ORDER BY customerid
LIMIT 5
```

MySQL в некоторых случаях позволяет ускорить операции удаления. Например, обычно после завершения удаления возвращается число удаленных записей. Но если удаляются все записи таблицы, то MySQL не будет считать удаленные записи и вернет 0, так как это быстрее. В режиме *AUTOCOMMIT* оператор *DELETE* без фразы *WHERE* будет заменен на более быструю операцию *TRUNCATE*.

Помните, что скорость удаления строк из таблицы напрямую зависит от количества индексов, построенных по этой таблице и от размера индексного кэша. Удаление будет выполняться быстрее, если данные удаляются из таблицы с небольшим числом индексов или вообще без индексов, либо если доступен большой индексный кэш.

Oracle

Oracle позволяет удалять строки из таблиц, представлений, материализованных представлений, вложенных подзапросов и секционированных таблиц и представлений:

```
DELETE [FROM]
  {имя_таблицы | ONLY (имя_таблицы)} [псевдоним]
    [{PARTITION (имя_секции) |
    SUBPARTITION (имя_подсекции)}] |
  (подзапрос [WITH {READ ONLY |
    CHECK OPTION [CONSTRAINT имя_ограничения]}]) |
  TABLE (выражение_коллекция) [ (+) ]}
[подсказка]
[WHERE условие_поиска]
[RETURNING выражение[, ...] INTO переменная[, ...]]
[LOG ERRORS [INTO [схема.]имя_таблицы] [(простое_выражение)]
  [REJECT LIMIT {UNLIMITED |целое_число }]]
```

Параметры имеют следующие значения:

имя_таблицы [псевдоним]

Указывает таблицу, материализованное представление, секционированную таблицу или представление, из которого удаляются строки. Вы можете при необходимости указать имя схемы или связь с другой базой данных. По умолчанию Oracle будет использовать таблицу в текущей схеме локального сервера. Также можно указать псевдоним для таблицы. Псевдоним необходим, если целевая таблица использует атрибут или метод объектного типа.

PARTITION имя_секции

Применяет операцию удаления только к указанной секции. Использование *PARTITION* не является обязательным при удалении из секционированной таблицы, но позволяет уменьшить сложность фразы *WHERE*.

SUBPARTITION имя_секции

Применяет операцию удаления только к указанной подсекции вместо всей таблицы.

(подзапрос [WITH {READ ONLY|CHECK OPTION [CONSTRAINT имя_ограничения]}])

Указывает, что строки удаляются из подзапроса, а не из таблицы или представления. Для этой фразы используются следующие параметры:

подзапрос

Определяет *SELECT*-оператор подзапроса. В качестве подзапроса можно использовать любой корректный подзапрос, но без фразы *ORDER BY*.

WITH READ ONLY

Указывает, что подзапрос не может обновляться.

WITH CHECK OPTION [CONSTRAINT имя_ограничения]

Гарантирует, что Oracle не выполнит удаление тех записей, которые не попадают в результирующее множество подзапроса. Фразой [*CONSTRAINT имя_ограничения*] можно ограничить допустимые изменения с помощью указанного ограничения.

TABLE (выражение_коллекция) [(+)]

Позволяет работать с указанной коллекцией как с таблицей, хотя коллекция может задаваться подзапросом, функцией или другим конструктором коллекций. В любом случае, указанное выражение должно быть либо вложенной таблицей, либо иметь тип *VARRAY*.

подсказка

Указывает оптимизатору необходимый порядок выполнения запроса, отличный от того, который мог бы быть выбран по умолчанию (например, можно игнорировать при выполнении определенный индекс). Информацию о подсказках оптимизатору ищите в документации.

RETURNING выражение

Возвращает строки, обработанные командой. (По умолчанию *DELETE* возвращает только число удаленных записей.) Фразу *RETURNING* можно использовать при удалении из таблицы, материализованного представления или представления с одной базовой таблицей. При удалении единичной записи *RETURNING* возвращает значения из удаленной записи (определяемые выражением) в переменные PL/SQL или связанные переменные. При удалении нескольких записей их значения сохраняются в связанных массивах.

INTO переменная

Определяет переменные, в которые сохраняются значения удаленных записей, определенные в *RETURNING*. Для каждого выражения из *RETURNING* в *INTO* должна быть определена соответствующая переменная.

LOG ERRORS [INTO [схема.]имя_таблицы] [(простое_выражение)] [REJECT LIMIT {UNLIMITED|целое_число}]

Сохраняет в журнальной таблице информацию об ошибках DML-операции и значения соответствующих строк. Нарушения ограничений целостности всегда вызывают остановку и откат операции, вне зависимости от того, используется или нет фраза *LOG ERRORS*. При помощи *LOG ERRORS* нельзя журналировать ошибки в столбцах типов *LONG* и *LOB*, а также в столбцах объектных типов, хотя сами таблицы могут содержать такие столбцы. Параметры этой фразы следующие:

INTO [схема.]имя_таблицы

Указывает имя журнальной таблицы. По умолчанию используется значение **ERR$_xxx**, где xxx соответствует первым 25 символам из названия таблицы, из которой удаляются записи.

(простое_выражение)

Указывает выражение, которым помечаются записи в таблице ошибок. Это позволяет различать ошибки, вызванные разными DML-операциями.

REJECT LIMIT {UNLIMITED |целое_число}

Указывает предельное значение числа ошибок, при достижении которого операция DELETE прерывается и транзакция откатывается. По умолчанию используется значение 0. Ключевое слово *UNLIMITED* используется при неограниченном числе ошибок.

При выполнении *DELETE* Oracle освобождает место и возвращает его таблице или индексу, в котором хранились данные.

Для удаления из представления требуется, чтобы представление не содержало операторов работы с множествами, ключевого слова *DISTINCT*, соединений, агрегатных функций, аналитических функций, подзапросов или коллекций в списке *SELECT*, фраз *GROUP BY*, *ORDER BY*, *CONNECT BY* и *START WITH*.

Вот пример удаления записей из таблицы удаленного сервера:

```
DELETE FROM scott.sales@chicago;
```

Следующий оператор удаляет данные из производной таблицы, то есть из коллекции:

```
DELETE TABLE(SELECT contactname FROM customers c
  WHERE c.customerid = 'BOTTM') s
WHERE s.region IS NULL OR s.country = 'MEXICO';
```

Вот пример удаления данных из определенной секции:

```
DELETE FROM sales PARTITION (sales_q3_1997)
WHERE qty > 10000;
```

В последнем примере мы возвращаем удаленные значения в переменные с помощью фразы *RETURNING*:

```
DELETE FROM employee
WHERE job_id = 13
AND hire_date + TO_YMINTERVAL('01-06') =< SYSDATE;
RETURNING job_lvl
INTO :int01;
```

В этом примере удаляются записи из таблицы **employee**, и значения **job_lvl** сохраняются в заранее объявленной переменной **int01**.

PostgreSQL

В PostgreSQL команда *DELETE* используется для удаления строк и любых подклассов из таблицы. В остальном реализация соответствует стандарту ANSI. Синтаксис следующий:

```
DELETE [FROM] [ONLY] [схема.]имя_таблицы
[ USING список_использования ]
[ WHERE условие_поиска | WHERE CURRENT OF имя_курсора ]
[ RETURNING { * | выражение [AS псевдоним][, ...] } ]
```

Для удаления записей только из указанной таблицы используйте фразу *ONLY*. В противном случае PostgreSQL удалит записи также из любых подтаблиц. В PostgreSQL также поддерживаются две другие важные фразы:

USING список_использования

Определяет список табличных выражений, что позволяет использовать столбцы этих таблиц в *WHERE*. Эта функциональность аналогична использованию нескольких таблиц во фразе *FROM* оператора *SELECT*.

*RETURNING { * | выражение [AS псевдоним][, ...] }*

Определяет выражение, возвращаемое оператором для каждой удаленной строки. Можно использовать либо символ * для возврата всех столбцов, либо указать произвольное выражение, использующее столбцы таблиц из *FROM* и *USING*.

Следующий оператор удаляет все записи из таблицы **titiles**:

```
DELETE titles
```

Для удаления из таблицы **authors** всех строк, начинающихся на 'Mc', можно использовать следующий оператор:

```
DELETE FROM authors
WHERE au_lname LIKE 'Mc%'
```

Для удаления из **titles** всех строк со старыми значениями **title_id** можно выполнить следующее:

```
DELETE titles WHERE title_id >= 40
```

Для удаления записей из **titles** с отсутствующими продажами:

```
DELETE titles WHERE ytd_sales IS NULL
```

Можно удалить строки из одной таблицы на основании результатов запроса к другой таблице (в примере мы удаляем записи из таблицы **titleauthors**, для которых в таблице **titles** есть записи со словом «computers»):

```
DELETE FROM titleauthor
WHERE title_id IN
  (SELECT title_id
  FROM titles
  WHERE title LIKE '%computers%')
DISCONNECT
```

Можно также при удалении вернуть полную информацию обо всех удаленных строках:

```
DELETE FROM titles WHERE ytd_sales IS NULL RETURNING *;
```

SQL Server

В SQL Server можно удалять строки как из таблиц, так и из представлений, построенных по одной таблице. В SQL Server можно использовать вторую фразу *FROM* для соединений. Синтаксис оператора следующий:

```
[WITH табличное_выражение[, ...]]
DELETE [TOP ( число ) [PERCENT]]
[FROM] имя_таблицы [[AS] псевдоним]
```

```
[WITH ( подсказка [...] )]
[OUTPUT выражение INTO {@табличная_переменная | таблица} [ (список_столбцов[, ...]) ]]
[FROM таблица_источник[, ...]]
[ [{INNER | CROSS | [LEFT | RIGHT | FULL] OUTER}]
  JOIN соединяемая_таблицы ON выражение][, ...] ]
[WHERE условие_поиска | WHERE CURRENT OF [GLOBAL]
  имя_курсора]
[OPTION ( подсказка[, ...N] )]
```

Элементы синтаксиса имеют следующие значения:

WITH табличное_выражение

Определяет именованный временный набор строк, порождаемый оператором *SELECT*.

DELETE имя_таблицы

Указывает таблицу или представление, из которого удаляются строки. Вы можете удалить строки из представления, если оно построено на базе одной таблицы, в нем не используются агрегатные функции и производные столбцы. Если для таблицы или представления вы не указываете имя сервера, базы данных или схемы, то SQL Server будет использовать текущий контекст. Вместо таблицы или представления вы можете использовать функции *OPENDATASOURCE* и *OPENQUERY*, описанные в разделе, посвященном оператору *SELECT*.

TOP (число) [PERCENT]

Определяет либо число строк, которое должен удалить оператор, либо процент (*PERCENT*) от общего числа строк таблицы. Если вместо константного значения используется выражение или переменная, то необходимо заключать выражение в круглые скобки. Если указывается процент строк, то выражение должно иметь тип *FLOAT* и значение в диапазоне от 0 до 100. Если указывается точное число строк, то выражение должно иметь тип *BIGINT*.

WITH (подсказка)

Указывает оптимизатору необходимый порядок выполнения запроса, отличный от того, который мог бы быть выбран по умолчанию (например, можно игнорировать при выполнении определенный индекс). *WITH* позволяет указать одну или несколько подсказок, применимых к целевой таблице. Информацию о подсказках оптимизатору ищите в документации.

OUTPUT выражение INTO {@табличная_переменная | таблица}[(список_столбцов[,...])]

Возвращает значения (задаваемые выражением) удаленных строк в указанную табличную переменную или таблицу (в обычном режиме *DELETE* вернет только число удаленных строк). Если не указан список_столбцов, то столбцы, указанные в выражении *OUTPUT*, должны соответствовать столбцам табличной переменной или таблицы, в которую сохраняются удаленные строки. (Также эта таблица не может иметь триггеры, ограничения *CHECK* и внешние ключи.)

FROM таблица_источник

Вторая фраза *FROM* используется для соединений таблицы из первой фразы *FROM* с другими таблицами, что позволяет обходиться без коррелированных подзапросов. В этой фразе *FROM* можно указать несколько таблиц.

[[{INNER | CROSS | [LEFT | RIGHT | FULL] OUTER}] JOIN соединяемая_таблицы ON выражение][, ...]]

Используется для соединений таблиц во второй фразе *FROM*. Вы можете использовать любой тип соединений, поддерживаемый в SQL Server. Соединения объясняются в разделе, посвященном оператору *SELECT*, позднее в этой главе.

GLOBAL имя_курсора

Указывает, что нужно удалить текущую запись открытого глобального курсора. Эта фраза является аналогом определенной стандартом фразы *WHERE CURRENT OF*.

OPTION (подсказка[, ...])

Позволяет изменить некоторые шаги плана выполнения запроса. Так как оптимизатор обычно выбирает наилучший план выполнения запроса, мы не рекомендуем использовать подсказки оптимизатору в запросах.

Существенным расширением в SQL Server стандартного оператора *DELETE* является вторая фраза *FROM*. Она позволяет использовать соединения таблиц для удаления строк из одной таблицы на основании значений связанных строк из других таблиц. Например, вы могли бы использовать достаточно сложный подзапрос для удаления записей о продажах из таблицы **sales** для книг о компьютерах:

```
DELETE FROM sales
WHERE title_id IN
  (SELECT title_id
  FROM titles
  WHERE type = 'computer')
```

В SQL Server ту же операцию можно выполнить намного элегантнее[1], используя вторую фразу *FROM*:

```
DELETE FROM sales
FROM sales AS s
INNER JOIN titles AS t ON s.title_id = t.title_id
  AND type = 'computer'
```

В следующем примере мы удаляем пачками по 2500 строк все строки с датой в поле **order_date** раньше 2003 года:

```
WHILE 1 = 1
BEGIN
  DELETE TOP (2500)
  FROM sales_history WHERE order_date <= '20030101'
  IF @@rowcount < 2500 BREAK
END
```

TOP следует применять вместо использовавшейся ранее команды *SET ROWCOUNT*, так как теперь *TOP* может быть использована большим числом алгоритмов оптимизации запроса. (До версии SQL Server 2005 оператор *SET ROW-*

[1] Элегантность этого решения сомнительна, т. к. работа оператора DELETE с JOIN будет значительно менее эффективна. – *Прим. науч. ред.*

COUNT часто использовался для пакетной обработки большого числа строк небольшими пачками, что позволяло избежать переполнения журнала транзакций и эскалации блокировок строк в табличную блокировку.)

В операторах *SELECT*, *INSERT*, *UPDATE*, *DELETE* и *CREATE VIEW* можно использовать табличные выражения. Эти выражения используются для объявления временных результирующих наборов строк, задаваемых оператором *SELECT*. При объявлении табличных выражений нельзя использовать фразы *COMPUTE*, *COMPUTE BY*, *FOR XML*, *FOR BROWSE*, *INTO*, *OPTION* и *ORDER BY*. В табличном выражении можно использовать несколько операторов *SELECT*, если они объединены при помощи операторов *UNION*, *UNION ALL*, *EXCEPT* или *INTERSECT*. Вот пример простого оператора *DELETE*, использующего табличное выражение:

```
WITH direct_reports (Manager_ID, DirectReports) AS
( SELECT manager_ID, COUNT(*)
  FROM hr.employee AS e
  WHERE manager_id IS NOT NULL
  GROUP BY manager_id )
DELETE FROM direct_reports
WHERE DirectReports <= 1;
```

Фраза *OUTPUT* позволяет увидеть все удаленные строки:

```
DELETE TOP 10 error_log WITH (READPAST)
OUTPUT deleted.*
WHERE error_log_id = '28-OCT-2008';
```

См. также

INSERT
SELECT
TRUNCATE TABLE
UPDATE

DISCONNECT

Оператор *DISCONNECT* используется для разрыва одного или нескольких соединений между текущим SQL-процессом и сервером базы данных.

СУБД	Уровень поддержки
MySQL	Не поддерживается
Oracle	Поддерживается с ограничениями
PostgreSQL	Не поддерживается
SQL Server	Поддерживается с ограничениями

Синтаксис SQL2003

```
DISCONNECT {CURRENT | ALL | имя_подключения | DEFAULT}
```

Ключевые слова

CURRENT

Используется для закрытия активного соединения.

ALL

Закрывает все активные соединения текущего пользователя.

Общие правила

DISCONNECT может быть использована для закрытия именованной SQL-сессии (имя_подключения), текущего соединения (*CURRENT*), соединения, используемого по умолчанию (*DEFAULT*), или всех соединений пользователя (*ALL*). Например, для закрытия соединения с названием *new_york* можно выполнить следующую команду:

```
DISCONNECT new_york
```

А для закрытия всех сессий, открытых для текущего пользовательского процесса, используйте:

```
DISCONNECT ALL
```

Советы и хитрости

DISCONNECT не является универсальным оператором, поддерживаемым всеми платформами. Не используйте *DISCONNECT* в кросс-платформенных приложениях. Вместо этого используйте способы закрытия соединений, являющиеся предпочтительными для каждой конкретной платформы.

MySQL

Не поддерживается.

Oracle

В Oracle оператор *DISCONNECT* поддерживается только в SQL*Plus, инструменте выполнения запросов. Используется следующий синтаксис:

```
DISC[ONNECT]
```

Этот оператор закрывает текущую сессию с сервером базы данных, но при этом можно продолжать работу в SQL*Plus. Например, программист может продолжить редактировать буфер, сохранять файлы и т. д., но для выполнения любых команд SQL потребуется установить новое подключение. Для выхода из SQL*Plus используются команды *EXIT* или *QUIT*. Для закрытия текущего подключения к Oracle выполните:

```
DISCONNECT;
```

Аналогичный результат вы можете получить используя оператор *ALTER SYSTEM DISCONNECT SESSION*.

PostgreSQL

Не поддерживается. Вместо этого в каждом программном интерфейсе имеются отдельные команды для закрытия соединений. В Server Programming Interface это *SPI_FINISH*, а в PL/TCL это *PG_DISCONNECT*.

SQL Server

SQL Server поддерживает стандартный оператор *DISCONNECT* только во встраиваемом SQL (Embedded SQL, ESQL), но не в инструменте выполнения запросов SQL Server Management Studio. Для «аккуратного» отключения от SQL Server из ESQL-приложения используйте оператор *DISCONNECT ALL*.

См. также

CONNECT

DROP

Любые объекты базы данных, создаваемые операторами *CREATE*, могут быть удалены соответствующими операторами *DROP*. В некоторых платформах *ROLLBACK* позволяет восстановить удаленный объект. В других платформах оператор *DROP* не подлежит отмене, так что его стоит использовать с осторожностью.

СУБД	Уровень поддержки
MySQL	Поддерживается с ограничениями
Oracle	Поддерживается с вариациями
PostgreSQL	Поддерживается с ограничениями
SQL Server	Поддерживается с ограничениями

Синтаксис SQL2003

На текущий момент стандарт SQL2003 поддерживает возможность удаления большого числа разных типов объектов, многие из которых не поддерживаются производителями. Синтаксис оператора согласно ANSI следующий:

```
DROP [тип_объекта] имя_объекта {RESTRICT | CASCADE}
```

Ключевые слова

DROP [тип_объекта] имя_объекта

Безвозвратно удаляет объект указанного типа с указанным именем. Если не указана схема, в которой находится объект, то подразумевается текущая схема. ANSI SQL2003 поддерживает длинный список типов объектов, каждый из которых создается соответствующим оператором *CREATE*. Операторы *CREATE*, рассматриваемые в этой книге и имеющие парный оператор *DROP*, используются с фразами:

- *DOMAIN*
- *FUNCTION*
- *METHOD*
- *PROCEDURE*
- *ROLE*
- *SCHEMA*
- *TABLE*

- *TRIGGER*
- *TYPE*
- *VIEW*

RESTRICT|CASCADE

Позволяет предотвратить выполнение оператора *DROP* при наличии зависимых объектов (*RESTRICT*), либо позволяет одновременно удалить как сам объект, так и все зависимые объекты (*CASCADE*). Эта опция недопустима для некоторых типов объектов, например для *DROP TRIGGER*, а в некоторых случаях она обязательна, например для *DROP SCHEMA*. В частности, *DROP SCHEMA RESTRICT* удалит только пустую схему, а если в схеме есть объекты, то оператор будет прерван. А *DROP SCHEMA CASCADE* удалит и схему, и все содержащиеся в ней объекты.

Общие правила

Правила создания и модификации объектов различных типов приводятся в разделах, посвященных соответствующим операторам *CREATE/ALTER*.

Оператор *DROP* удаляет существующий объект. Объект удаляется безвозвратно, и все пользователи, имевшие ранее доступ к этому объекту, мгновенно теряют возможность использовать объект.

Вы можете указывать полное имя объекта вместе со схемой, в которой находится объект, например:

```
DROP TABLE scott.sales_2008 CASCADE;
```

Этот оператор удалит не только таблицу **scott.sales_2008**, но и все представления, триггеры и ограничения, созданные на базе этой таблицы. С другой стороны, можно не указывать имя схемы, если подразумевается текущая схема. Например:

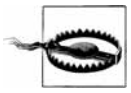
```
DROP TRIGGER before_ins_emp;
DROP ROLE sales_mgr;
```

Хотя это и не оговорено стандартом, в большинстве реализаций оператор *DROP* завершается с ошибкой, если удаляемый объект используется другим пользователем.

Советы и хитрости

Оператор *DROP* отработает корректно, только если выполнен для существующего объекта правильного типа пользователем, имеющим соответствующие привилегии (обычно требуется привилегия *DROP TABLE*, подробности приводятся в описании оператора *GRANT*). Стандарт SQL требует, чтобы объект мог удалить только создатель этого объекта, но в большинстве платформ есть некоторые отступления от этого правила. Например, суперпользователь или администратор базы данных обычно может удалить любой объект сервера баз данных.

В некоторых платформах оператор *DROP* не удалит объекты, имеющие какие-либо расширенные свойства. Например, в SQL Server нельзя удалить реплицируемую таблицу, пока она не будет исключена из репликации.



Имейте в виду, что в большинстве платформ вы не увидите предупреждения или сообщения об ошибке, если оператор *DROP* порождает проблемы с зависимыми объектами. Например, если вы попытаетесь удалить таблицу, используемую в нескольких представлениях и хранимых процедурах, то таблица удалится без проблем, но зато вы получите ошибку при попытке использования зависящих от таблицы представлений или процедур. Для предотвращения таких проблем вы можете либо воспользоваться опцией *RESTRICT*, либо проверить зависимости до выполнения оператора *DROP*.

Вы, возможно, обратили внимание, что стандарт ANSI не поддерживает некоторые операторы *DROP*, например *DROP DATABASE* и *DROP INDEX*, хотя они поддерживаются в каждой платформе, рассматриваемой в этой книге (и почти всеми платформами, представленными на рынке). Точный синтаксис таких команд рассматривается в разделах, посвященных каждой платформе.

MySQL

MySQL поддерживает сравнительно небольшое число типов объектов, соответственно синтаксис оператора *DROP* тоже ограничен:

```
DROP { {DATABASE | SCHEMA} | FUNCTION |
   INDEX [ONLINE | OFFLINE] | PROCEDURE |
   [TEMPORARY] TABLE | TRIGGER | VIEW }
[IF EXISTS] имя_объекта[, ...]
[RESTRICT |CASCADE]
```

Элементы синтаксиса имеют следующие значения:

{DATABASE | SCHEMA} *имя_базы_данных*

Удаляет указанную базу данных, а также все объекты внутри базы данных. В MySQL операторы *DROP SCHEMA* и *DROP DATABASE* являются синонимами. *DROP DATABASE* удаляет все файлы и каталоги, используемые базой данных. MySQL возвращает сообщение о числе удаленных файлов из каталога базы данных. (Удаляются файлы со следующими расширениями: *.BAK*, *.DAT*, *.FRM*, *.HSH*, *.ISD*, *.ISM*, *.MRG*, *.MYD*, *.MYI*, *.DM* и *.FM*) Вы можете удалить одним оператором только одну базу данных. Опции *RESTRICT* и *CASCADE* в операторе *DROP DATABASE* не используются.

FUNCTION *имя_функции*

В версиях MySQL 5.1 и старше используется для удаления пользовательских функций. Вы также можете использовать *IF EXISTS*.

INDEX [ONLINE | OFFLINE] *имя_индекса* *ON* *имя_таблицы*

В версиях MySQL 3.22 и старше используется для удаления индексов таблиц. В версиях младше 3.22 этот оператор ничего не делает, а в 3.22 он фактически заменяется на *ALTER TABLE ... DROP INDEX*. Начиная с версии 5.1.22-ndb-6.2.5 вы можете удалять индексы в оперативном режиме, используя ключевое слово *ONLINE*. При оперативном удалении индекса не создается промежуточная временная копия таблицы. В *DROP INDEX* нельзя использовать опции *RESTRICT*, *CASCADE* и *IF EXISTS*.

PROCEDURE имя_процедуры

В версиях MySQL 5.1 и старше используется для удаления хранимых процедур. Вы также можете использовать *IF EXISTS*.

[TEMPORARY] TABLE имя_таблицы[, ...]

Удаляет одну или несколько таблиц (имена таблиц разделяются запятыми). MySQL удаляет описание таблицы, а также соответствующие файлы с расширениями *.FRM*, *.MYD* и *.MYI*. При выполнении этого оператора фиксируются все активные транзакции. Слово *TEMPORARY* позволяет удалить временную таблицу, при этом не фиксируются транзакции и не проверяются права доступа.

TRIGGER [имя_схемы.]имя_триггера

Используется в MySQL версий 5.0.2 и старше для удаления триггеров. Вы можете использовать *IF EXISTS* для удаления триггера только в том случае, если он реально существует.

VIEW имя_представления

Используется для удаления представления с указанным именем. Вы также можете использовать *IF EXISTS*.

IF EXISTS

Подавляет сообщение об ошибке в случае попытки удаления реально не существующего объекта. Поддерживается начиная с версии MySQL 3.22.

RESTRICT|CASCADE

Слова используются как синтаксические заглушки, не вызывают сообщений об ошибках и не имеют никакого другого эффекта.

MySQL поддерживает только удаление базы данных, таблицы или индекса из таблицы. Хотя в операторе *DROP* можно использовать ключевые слова *RESTRICT* и *CASCADE*, они не имеют реального эффекта. Вы можете использовать *IF EXISTS* для того, чтобы при попытке удаления несуществующего объекта не возникали сообщения об ошибках.

MySQL позволяет удалять и другие типы объектов с использованием аналогичного синтаксиса:

```
DROP { EVENT | FOREIGN KEY | LOGFILE GROUP | PREPARE |
  PRIMARY KEY | SERVER | TABLESPACE | USER }
имя_объекта
```

Эти варианты оператора *DROP* выходят за рамки этой книги. Более подробную информацию о них ищите в документации.

Oracle

В Oracle поддерживается большая часть вариантов оператора *DROP*, описанных в ANSI, а также некоторые другие варианты, соответствующие типам объектов, поддерживаемым только в Oracle. Oracle поддерживает удаление следующих типов объектов из SQL3:

```
DROP { DATABASE | FUNCTION | INDEX | PROCEDURE | ROLE |
  TABLE | TRIGGER | TYPE [BODY] | VIEW }
имя_объекта
```

Операторы *DROP* для разных объектов могут иметь различный вид, поэтому для каждого варианта оператора приводится его полный синтаксис:

DATABASE имя_базы_данных

Удаляет указанную базу данных.

FUNCTION имя_функции

Удаляет функции, не входящие в состав пакетов. (Если вы хотите удалить функцию из пакета, то используйте оператор *CREATE PACKAGE...OR REPLACE* для объявления пересоздания пакета без этой функции.) Любые локальные объекты, использующие удаленную функцию, становятся невалидными, а любые статистические типы, связанные с функцией, отвязываются от нее.

INDEX имя_индекса [FORCE]

Удаляет обычный или предметный индекс. Удаление индекса делает невалидными все объекты (представления, пакеты, функции, хранимые процедуры), зависящие от таблицы, по которой был создан индекс. Также удаление индекса делает невалидными курсоры и планы выполнения запросов, использовавшие индекс, что приводит к полным разборам при следующих выполнениях соответствующих операторов SQL.

Обычные индексы являются вторичными объектами и могут удаляться и пересоздаваться без потери пользовательских данных. Индексно-организованные таблицы комбинируют в себе и данные, и индекс, поэтому их нельзя удалять и пересоздавать таким же образом, как и обычные индексы. Индексно-организованные таблицы удаляются оператором *DROP TABLE*.

При удалении секционированного индекса удаляются все его секции, а при удалении индекса с составным секционированием удаляются все секции и подсекции. При удалении предметного индекса удаляются все связанные с ним статистики, и удаляется связь со всеми статистическими типами. Ключевое слово *FORCE* используется только при удалении предметного индекса. *FORCE* позволяет удалить предметный индекс, находящийся в состоянии *IN PROGRESS*, или для которого процедура *indextype* возвращает ошибку. Например:

```
DROP INDEX ndx_sales_salesperson_quota;
```

PROCEDURE имя_процедуры

Удаляет указанную хранимую процедуру. Любые зависимые объекты становятся невалидными, а попытки их использования завершаются с ошибками, пока вы не пересоздадите процедуру. Если вы пересоздали хранимую процедуру, а затем используете зависимый объект, то зависимый объект перекомпилируется.

ROLE имя_роли

Удаляет роль и отзывает ее у всех пользователей и других ролей. Вновь создаваемые сессии не могут использовать удаленную роль, но те сессии, которые использовали роль в момент ее удаления, не затрагиваются. Вот пример удаления роли:

```
DROP ROLE sales_mgr;
```

TABLE имя_таблицы [CASCADE CONSTRAINTS] [PURGE]

Удаляет указанную таблицу и все ее данные, индексы и триггеры (даже в другой схеме), делает невалидными все зависимые объекты, а также отменяются все разрешения на таблицу. Для секционированных таблиц удаляются все секции и подсекции. При удалении индексно-организованной таблицы удаляются и таблицы соответствия идентификаторов. От удаленной таблицы отвязываются все статистические типы. Если для таблицы были созданы журналы материализованных представлений, то они тоже удаляются.

Оператор *DROP TABLE* применяется к обычным, индексно-организованным и объектным таблицам. Удаленная таблица не удаляется полностью, а помещается в корзину, если только явно не указана опция *PURGE*, позволяющая сразу полностью освободить место, занимаемое таблицей. (В Oracle отдельно поддерживается оператор *PURGE*, с помощью которого можно удалить таблицу из корзины.) Для внешних таблиц *DROP TABLE* удаляет только метаданные, а файл с данными нужно удалять командами операционной системы.

Используйте фразу *CASCADE CONSTRAINTS* для удаления всех внешних ключей, ссылающихся на первичные и уникальные ключи удаляемой таблицы. Вы не сможете удалить таблицу, используемую в ограничениях ссылочной целостности, без использования *CASCADE CONSTRAINTS*.

В следующем примере удаляется таблица **job_desc** в схеме **emp**, а затем таблица **job** и все внешние ключи, ссылающиеся на первичный ключ таблицы **job**:

```
DROP TABLE emp.job_desc;
DROP TABLE job CASCADE CONSTRAINTS;
```

TRIGGER имя_триггера

Удаляет из базы данных указанный триггер.

TYPE [BODY] имя_типа [{FORCE|VALIDATE}]

Удаляет спецификацию и тело пользовательского объектного типа данных, вложенной таблицы или *VARRAY*. В случае если тип является супертипом, имеет связанный статистический тип или зависимости любого другого вида, то для удаления такого типа нужно использовать опцию *FORCE*. В этом случае все подтипы и статистические типы становятся невалидными. Oracle удаляет все публичные синонимы, связанные с удаляемым типом. Можно использовать опцию *BODY* для того, чтобы удалить только тела типа, но оставить его спецификацию. *BODY* нельзя использовать совместно с *FORCE* и *VALIDATE*. Используйте опцию *VALIDATE*, чтобы при удалении подтипа проверить существование экземпляров этого подтипа в любом из его супертипов. Удаление произойдет только при отсутствии сохраненных экземпляров. Пример удаления типа:

```
DROP TYPE salesperson_type;
```

VIEW имя_представления [CASCADE CONSTRAINTS]

Удаляет указанное представление, а также делает невалидными все зависимые представления, синонимы и материализованные представления. Используйте необязательную опцию *CASCADE CONSTRAINTS* для удаления всех ссылочных ограничений целостности, зависящих от представления. Без

использования этой опции и при наличии ссылочных ограничений целостности представление не будет удалено. Например, следующий оператор удаляет представление **active_employees** в схеме **hr**:

```
DROP VIEW hr.active_employees;
```

В операторе *DROP* перед именем объекта можно также указать схему. Если схема не указана, то используется схема по умолчанию для текущей сессии. Следующий оператор удаляет представление из схемы **sales_archive**:

```
DROP VIEW sales_archive.sales_1994;
```

А если ваша схема **scott**, то в следующем операторе подразумевается удаление представления **scott.sales_1994**:

```
DROP VIEW sales_1994;
```

Oracle поддерживает удаление большого числа объектов, не описанных в стандарте SQL3:

```
DROP { CLUSTER | CONTEXT | DATABASE LINK | DIMENSION |
  DIRECTORY | DISKGROUP | FLASHBACK ARCHIVE | INDEXTYPE |
  JAVA | LIBRARY | MATERIALIZED VIEW |
  MATERIALIZED VIEW LOG | OPERATOR | OUTLINE | PACKAGE |
  PROFILE | RESTORE POINT | ROLLBACK SEGMENT | SEQUENCE |
  SYNONYM | TABLESPACE | TYPE BODY | USER }
имя_объекта
```

Эти операторы находятся вне рамок книги. За информацией об этих операторах обращайтесь к документации (хотя базовый синтаксис всех этих операторов практически одинаковый).

PostgreSQL

В PostgreSQL не поддерживаются ключевые слова *RESTRICT* и *CASCADE*, описанные в стандарте. Поддерживает широкий набор вариантов *DROP* в соответствии со следующим синтаксисом:

```
DROP { DATABASE | DOMAIN | FUNCTION | INDEX | ROLE |
       SCHEMA | TABLE | TRIGGER | TYPE | VIEW }
[IF EXISTS]
имя_объекта
[ CASCADE | RESTRICT ]
```

Синтаксис для каждого оператора DROP из стандарта следующий:

DATABASE имя_базы_данных

Удаляет указанную базу данных и очищает все каталоги с данными удаленной базы. Этот оператор может быть выполнен только владельцем базы данных, причем пользователь должен быть подключен не к удаляемой базе. Например, мы можем удалить базу данных **sales_archive**:

```
DROP DATABASE sales_archive;
```

DOMAIN имя_домена [, ...] [CASCADE | RESTRICT]

Удаляет один или несколько указанных доменов, принадлежащих текущему пользователю. *CASCADE* позволяет автоматически удалить все объекты, за-

висящие от домена, а *RESTRICT* предотвращает удаление, если найдены зависящие объекты. Если не указана ни одна опция, то по умолчанию подразумевается *RESTRICT*.

FUNCTION имя_функции ([тип_данных1[, ...]) [CASCADE | RESTRICT]

Удаляет указанную пользовательскую функцию. Так как в PostgreSQL допускается создание функций с одинаковыми именами, отличающихся только входными параметрами, то вам может потребоваться указать типы входных параметров для точного указания функции, которую вы хотите удалить. PostgreSQL не выполняет проверки зависимостей при удалении пользовательских функций. (Если вы попытаетесь использовать объект, зависящий от удаленной функции, то получите сообщение об ошибке.) Например:

```
DROP FUNCTION median_distribution (int, int, int, int);
```

INDEX имя_индекса [, ...] [CASCADE | RESTRICT]

Удаляет один или несколько указанных индексов. Например:

```
DROP INDEX ndx_titles, ndx_authors;
```

ROLE имя_роли[, ...]

Удаляет одну или несколько указанных ролей. В PostgreSQL нельзя удалить роль, на которую ссылается хотя бы один объект в базе данных. Поэтому вам необходимо удалить либо поменять владельца для всех объектов, принадлежащих удаляемой роли, используя *REASSIGN OWNED* или *DROP OWNED*, а также отозвать у роли все привилегии.

SCHEMA имя_схемы[, ...] [CASCADE | RESTRICT]

Удаляет из текущей базы данных одну или несколько схем. Схема может быть удалена только суперпользователем базы данных или владельцем схемы (при том, что владелец схемы может и не владеть всеми объектами в схеме).

TABLE имя_таблицы[, ...] [CASCADE | RESTRICT]

Удаляет одну или несколько таблиц вместе с индексами и триггерами. Например:

```
DROP TABLE authors, titles;
```

TRIGGER имя_триггера ON имя_таблицы [CASCADE | RESTRICT]

Удаляет из базы данных указанный триггер. При удалении триггера требуется указывать также имя таблицы, так как в PostgreSQL имя триггера должно быть уникально только в пределах таблицы, к которой прикреплен триггер. То есть возможно наличие нескольких триггеров с именем **insert_trigger**, каждый из которых создан для своей таблицы. Например:

```
DROP TRIGGER insert_trigger ON authors;
```

TYPE имя_типа [, ...] [CASCADE | RESTRICT]

Удаляет из базы данных один или несколько пользовательских типов данных. PostgreSQL не проверяет влияние удаления типа на зависимые объекты, такие как функции, агрегаты и таблицы, вы должны проверять зависимости объектов самостоятельно. (Не удаляйте стандартные типы, поставляемые вместе с PostgreSQL!) Имейте в виду, что реализация типов в PostgreSQL

отличается от стандарта ANSI. Обратитесь к разделу *CREATE/ALTER TYPE* за дополнительной информацией.

VIEW имя_представления [, ...] [CASCADE | RESTRICT]

Удаляет одно или несколько указанных представлений.

CASCADE | RESTRICT

CASCADE автоматически удаляет все объекты, зависящие от основного удаляемого объекта. *RESTRICT* запрещает удаление при обнаружении зависимых объектов. По умолчанию используется поведение опции *RESTRICT*. Так как эти опции допустимы не в каждом варианте оператора *DROP*, то мы указали их в синтаксисе тех операторов, где возможно их использование.

IF EXISTS

Подавляет сообщение об ошибках при попытках удаления несуществующих объектов. Может использоваться с любыми вариантами оператора *DROP*.

Обратите внимание, что в PostgreSQL нельзя указать базу данных, для которой выполняется удаление (за исключением оператора *DROP DATABASE*). Поэтому для удаления объекта вам нужно подключиться к той базе данных, в которой этот объект находится.

PostgreSQL поддерживает удаление некоторых объектов, не описанных в стандарте SQL3:

```
DROP { AGGREGATE | CAST | CONVERSION | GROUP | LANGUAGE |
OPERATOR [CLASS] | RULE | SEQUENCE | TABLESPACE | USER } имя_объекта
```

Эти операторы находятся вне рамок книги. За информацией об этих операторах обращайтесь к документации (хотя базовый синтаксис всех этих операторов практически одинаковый).

SQL Server

В SQL Server поддерживается несколько описанных в SQL3 вариантов оператора *DROP*:

```
DROP { DATABASE | FUNCTION | INDEX | PROCEDURE | ROLE |
SCHEMA | TABLE | TRIGGER | TYPE | VIEW } имя_объекта
```

Вот полный синтаксис каждого варианта:

DATABASE имя_базы_данных[, ...]

Удаляет указанную базу данных и стирает с диска все ее файлы. Оператор может быть выполнен только из базы **master**. Реплицируемые базы данных перед удалением должны быть удалены из своих цепочек репликации. Вы не можете удалить используемую базу данных и любую из системных базы данных (**master**, **model**, **msdb**, **temp**). Например, мы можем удалить базы **northwind** и **pubs** одной командой:

```
DROP DATABASE northwind, pubs
GO
```

FUNCTION [схема.]имя_функции[, ...]

Удаляет из текущей базы данных одну или несколько пользовательских функций.

INDEX имя_индекса ON имя_таблицы_или_представления[, ...] [WITH { MAXDOP = целое_число | ONLINE = {ON | OFF} | MOVE TO местоположение [FILESTREAM_ON местоположение] }]

Удаляет один или несколько индексов таблицы или индексированного представления. Этот оператор не следует использовать для удаления первичного ключа или уникального ключа: их следует удалять оператором *ALTER TABLE ... DROP CONSTRAINT*. При удалении кластерного индекса таблицы удаляются все его некластерные индексы. При удалении кластерного индекса можно использовать фразу *WITH*. *MAXDOP* определяет максимальную степень параллелизма, используемую при удалении. Значение *MAXDOP* может быть равно 1 (для последовательного выполнения операции), 0 (для автоматического определения степени параллелизма), либо быть целым число больше 1 (для явного указания числа параллельных процессов). Если для параметра *ONLINE* установлено значение *ON*, то при удалении индексов допускаются запросы и обновления таблиц, а при значении *OFF* на таблицы накладываются блокировки на время удаления индексов. *MOVE TO* указывает файловую группу или секцию, в которую перемещается кластерный индекс.[1] В новое место кластерный индекс[2] перемещается уже в форме кучи.

PROC[EDURE] имя_процедуры[, ...]

Удаляет из текущей базы данных одну или несколько хранимых процедур. В SQL Server допускается наличие нескольких версий одной хранимой процедуры, но их нельзя удалять по отдельности: все они удаляются одновременно. Например:

```
DROP PROCEDURE calc_sales_quota
GO
```

ROLE имя_роли[, ...]

Удаляет из текущей базы данных одну или несколько ролей. Роли не должны принадлежать объекты, иначе удаление завершится с ошибкой. Эти объекты предварительно следует удалить или изменить им владельца.

SCHEMA имя_схемы

Удаляет пустую схему. Если в схеме находятся объекты, то их следует удалить или переместить в другую схему.

TABLE [имя_базы_данных.][имя_схемы.]имя_таблицы[, ...]

Удаляет указанную таблицу, все ее данные, индексы, триггеры и ограничения. (Вы можете указать только имя таблицы, если она принадлежит текущему пользователю и находится в текущей базе данных, или явно указать базу данных и схемы.) Представления, функции и хранимые процедуры, использующие удаленную таблицу, не удаляются и не помечаются как невалидные, но вернут ошибку при попытке их вызова. Вы не сможете удалить

---

[1] Вероятно, авторы имели в виду: «MOVE TO определяет место, куда будут перемещаться строки данных, находящиеся на конечном уровне кластеризованного индекса». – *Прим. науч. ред.*

[2] Вероятно, авторы имели в виду: «данные перемещаются уже в форме кучи». – *Прим. науч. ред.*

таблицу, на которую ссылается внешний ключ, если не удалите предварительно сам ключ. Аналогично, вы не сможете удалить реплицируемую таблицу, пока не исключите ее из репликационной схемы. Любые пользовательские правила и значения по умолчанию отвязываются от удаленной таблицы. Их потребуется привязывать к таблице заново, если она будет пересоздана.

TRIGGER имя_триггера[, ...] [ON {DATABASE | ALL SERVER}]

Удаляет из текущей базы данных один или несколько триггеров. Фраза [*ON {DATABASE | ALL SERVER}]* может быть использована при удалении триггеров на DDL, а фраза *[ON ALL SERVER]* используется также при удалении триггеров на *LOGON*. *ON DATABASE* означает, что областью действия удаляемого триггера является база данных, и эту фразу следует использовать, если эта же фраза использовалась при создании триггера. *ON ALL SERVER* означает, что областью действия триггера на DDL или *LOGON* является весь текущий сервер.

TYPE [имя_схемы.]имя_типа[, ...]

Удаляет один или несколько пользовательских типов данных из текущей базы данных.

VIEW [имя_схемы.]имя_представления[, ...]

Удаляет из базы данных обычное или индексированное представление и освобождает занимаемое им пространство.

В SQL Server поддерживается большое число объектов, расширяющих стандарт ANSI и для удаления которых используются более или менее стандартные формы оператора *DROP*:

```
DROP { AGGREGATE | APPLICATION ROLE | ASSEMBLY |
  ASYMMETRIC KEY | BROKER PRIORITY | CERTIFICATE |
  CONTRACT | CREDENTIAL | CRYPTOGRAPHIC PROVIDER |
  DATABASE AUDIT SPECIFICATION | DATABASE ENCRYPTIIN KEY |
  DEFAULT | ENDPOINT | EVENT NOTIFICATION | EVENT SESSION |
  FULLTEXT CATALOG | FULLTEXT INDEX | FULLTEXT STOPLIST |
  LOGIN | MASTER KEY | MESSAGE TYPE | PARTITION FUNCTION |
  PARTITION SCHEME | QUEUE | REMOTE SERVICE BINDING |
  ESOURCE POOL | ROUTE | SERVER AUDIT |
  SERVER AUDIT SPECIFICATION | SERVICE | SIGNATURE |
  STATISTICS | SYMMETRIC KEY | SYNONYM | USER |
  WORKLOAD GROUP | XML SCHEMA COLLECTION }
имя_объекта
```

Эти операторы находятся вне рамок книги. За информацией об этих операторах обращайтесь к документации (хотя базовый синтаксис всех этих операторов практически одинаковый).

См. также

CALL
CONSTRAINTS
CREATE/ALTER FUNCTION/PROCEDURE/METHOD
CREATE SCHEMA
CREATE/ALTER TABLE