Язык SQL. Изменение и удаление данных.

Оператор UPDATE

Оператор **UPDATE** изменяет имеющиеся данные в таблице. Команда имеет следующий синтаксис

```
UPDATE <имя таблицы>
  SET {имя столбца = {выражение для вычисления значения столбца
   | NULL
   | DEFAULT},...}
   [ {WHERE <предикат>}];
```

С помощью одного оператора могут быть заданы значения для любого количества столбцов. Однако в одном и том же операторе **UPDATE** можно вносить изменения в каждый столбец указанной таблицы только один раз. При отсутствии предложения **WHERE** будут обновлены все строки таблицы.

Если столбец допускает NULL-значение, то его можно указать в явном виде. Кроме того, можно заменить имеющееся значение на значение по умолчанию (**DEFAULT**) для данного столбца.

Ссылка на "выражение" может относиться к текущим значениям в изменяемой таблице. Например, мы можем уменьшить все цены ПК-блокнотов на 10 процентов с помощью следующего оператора:

```
UPDATE Laptop SET price=price*0.9
```

Разрешается также значения одних столбцов присваивать другим столбцам. Пусть, например, требуется заменить жесткие диски менее 10 Гб в ПК-блокнотах. При этом емкость новых дисков должна составлять половину объема RAM, имеющейся в данных устройствах. Эту задачу можно решить следующим образом:

```
UPDATE Laptop SET hd=ram/2 WHERE hd < 10;
```

Естественно, типы данных столбцов *hd* и *ram* должны быть совместимы. Для приведения типов может использоваться выражение **CAST**.

Если требуется изменять данные в зависимости от содержимого некоторого столбца, можно воспользоваться выражением **CASE**. Если, скажем, нужно поставить жесткие диски объемом 20 Гб на ПК-блокноты с памятью менее

128 Мб и 40 гигабайтные - на остальные ПК-блокноты, то можно написать такой запрос:

```
UPDATE Laptop
  SET hd = CASE WHEN ram<128 THEN 20 ELSE 40 END
```

Для вычисления значений столбцов допускается также использование подзапросов. Например, требуется укомплектовать все ПК-блокноты самыми быстрыми процессорами из имеющихся. Тогда можно написать:

```
UPDATE Laptop
  SET speed = (SELECT MAX(speed) FROM Laptop)
```

Необходимо сказать несколько слов об автоинкрементируемых столбцах. Если столбец *code* в таблице *Laptop* определен как **IDENTITY(1,1)**, то следующий оператор

```
UPDATE Laptop SET code=5 WHERE code=4
```

не будет выполнен, т.к. автоикрементируемое поле не допускает обновления, и мы получим соответствующее сообщение об ошибке. Чтобы выполнить все же эту задачу, можно поступить следующим образом. Сначала вставить нужную строку, используя **SET IDENTITY_INSERT**, после чего удалить старую строку:

```
SET IDENTITY_INSERT Laptop ON
INSERT INTO Laptop_ID(code, model, speed, ram, hd, price, screen)
  SELECT 5, model, speed, ram, hd, price, screen
  FROM Laptop_ID WHERE code=4
DELETE FROM Laptop_ID WHERE code=4
```

Разумеется, другой строки со значением *code*=5 в таблице быть не должно.

В Transact-SQL оператор **UPDATE** расширяет стандарт за счет использования необязательного предложения **FROM**. В этом предложении специфицируется таблица, обеспечивающая критерий для операции обновления. Дополнительную гибкость здесь дает использование операций соединения таблиц.

Пример. Пусть требуется указать "No PC" (нет ПК) в столбце *type* для тех моделей ПК из таблицы *Product*, для которых нет соответствующих строк в таблице *PC*. Решение посредством соединения таблиц можно записать так:

```
UPDATE Product
  SET type='No PC'
  FROM Product pr LEFT JOIN PC ON pr.model=pc.model
  WHERE type='pc' AND pc.model IS NULL
```

Здесь используется внешнее соединение, в результате чего столбец *pc.model* для моделей ПК, отсутствующих в таблице *PC*, будет содержать NULL-значение, что и используется для идентификации подлежащих обновлению строк. Естественно, эта задача имеет решение и в "стандартном" исполнении:

```
UPDATE Product
  SET type='No PC'
  WHERE type='pc' and model NOT IN (SELECT model FROM PC)
```

Оператор DELETE

Оператор **DELETE** удаляет строки из временных или постоянных базовых таблиц, представлений или курсоров, причем в двух последних случаях действие оператора распространяется на те базовые таблицы, из которых извлекались данные в эти представления или курсоры. Оператор удаления имеет простой синтаксис:

DELETE FROM <имя таблицы > [WHERE <предикат>];

Если предложение **WHERE** отсутствует, удаляются все строки из таблицы или представления (представление должно быть обновляемым). Более быстро эту операцию (удаление всех строк из таблицы) в Transact-SQL можно также выполнить с помощью команды

```
TRUNCATE TABLE <имя таблицы>
```

Однако есть ряд отличий в реализации команды **TRUNCATE TABLE** по сравнению с использованием оператора **DELETE**, которые следует иметь в виду:

1. Не журнализируется удаление отдельных строк таблицы. В журнал записывается только освобождение страниц, которые были заняты данными

таблицы.
2. Не отрабатывают триггеры, в частности, триггер на удаление.
3. Команда неприменима, если на данную таблицу имеется ссылка по внешнему ключу, и даже если внешний ключ имеет опцию каскадного удаления.
4. Значение счетчика (**IDENTITY**) сбрасывается в начальное значение.

Пример. Требуется удалить из таблицы *Laptop* все ПК-блокноты с размером экрана менее 12 дюймов.

```
DELETE FROM Laptop
  WHERE screen < 12;
```

Все блокноты можно удалить с помощью оператора

```
DELETE FROM Laptop
```

или

```
TRUNCATE TABLE Laptop
```

Transact-SQL расширяет синтаксис оператора **DELETE**, вводя дополнительное предложение **FROM**

FROM <источник табличного типа>

При помощи *источника табличного типа* можно конкретизировать данные, удаляемые из таблицы в первом предложении **FROM**.

При помощи этого предложения можно выполнять соединения таблиц, что логически заменяет использование подзапросов в предложении **WHERE** для идентификации удаляемых строк.

Поясним сказанное на примере. Пусть требуется удалить те модели ПК из таблицы *Product*, для которых нет соответствующих строк в таблице *PC*.

Используя стандартный синтаксис, эту задачу можно решить следующим запросом:

```
DELETE FROM Product
  WHERE type='pc' AND model NOT IN (SELECT model FROM PC)
```

Заметим, что предикат *type*='pc' необходим здесь, чтобы не были удалены также модели принтеров и ПК-блокнотов.

Эту же задачу можно решить с помощью дополнительного предложения **FROM** следующим образом:

```
DELETE FROM Product
  FROM Product pr LEFT JOIN PC ON pr.model=pc.model
  WHERE type='pc' AND pc.model IS NULL
```

Здесь используется внешнее соединение, в результате чего столбец *pc.model* для моделей ПК, отсутствующих в таблице *PC*, будет содержать NULL-значение, что и используется для идентификации подлежащих удалению строк.