# Раздел 4. Основы языка SQL

## Тема 4.3

## Язык манипулирования данными (Data Manipulation Language)

**Вопросы лекции:**

1. **Запросы вставки строк данных INSERT.**
2. Запросы на удаление строк таблиц DELETE.
3. Запросы модификации данных UPDATE.

# Назначение операторов языка DML

Выполнение оператора CREATE TABLE никак не влияет на наполнение таблиц данными.

Для этих целей используется специальный язык манипулирования данными (Data Manipulation Language), который позволяет полностью контролировать процессы наполнения таблиц и изменения данных.

Основные команды манипулирования данными:

- INSERT – добавить новые данные
- DELETE – удалить данные
- UPDATE – изменить (обновить) данные.

# Назначение операторов языка DML

В версии **Microsoft SQL Server 2005** было введено предложение **OUTPUT**, позволяющее возвращать информацию из каждой строки, обработанной инструкциями INSERT, UPDATE или DELETE.

В версии **SQL Server 2008** была введена инструкция **MERGE** для того, чтобы усовершенствовать выполнение инструкций INSERT, UPDATE и DELETE, обращенных к таблице, основанной на результатах запроса к соединенной таблице.

**MERGE** — оператор языка SQL, который позволяет слить данные одной таблицы с данными другой таблицы.

При слиянии таблиц проверяется условие, и если оно истинно, то выполняется Update, а если нет — Insert.

Эти два средства **повышают гибкость** основных инструкций DML.

# Назначение операторов языка DML

Все операторы DML **изменяют состояние** базы данных, поэтому их рекомендуется использовать в сочетании с операторами управления транзакциями:

- COMMIT – зафиксировать результаты изменений или
- ROLLBACK – откат (отмена) изменений, произведенных транзакцией.

**Важно** – при выполнении операторов DML СУБД автоматически проверяет все заданные ограничения для изменяемых столбцов и таблиц – а это занимает определенное время.

# Вставка данных в таблицы – команда **INSERT**

Инструкция **INSERT** позволяет вставлять в таблицу **новые** строки.

В зависимости от схемы таблицы вам может понадобиться предоставить данные для всех или части столбцов таблицы.

На требования к данным в ваших инструкциях INSERT могут оказывать влияние:

- ограничения DEFAULT,
- свойства IDENTITY и
- значения NULL.

Инструкция **INSERT** позволяет вставлять в таблицы одну строку или сразу несколько строк, как **на основе параметров команды**, так и **на основе вложенного запроса на выборку** из другой таблицы.

# Вставка данных в таблицы – команда **INSERT**

У команды **INSERT** следующая синтаксическая запись:

```
INSERT  [ TOP ( expression ) [ PERCENT ] ]
[ INTO ]
 { <object> | rowset_function_limited [ WITH ( <Table_Hint_Limited> [
...n ] ) ] }
{ [ ( column_list ) ]  [ <OUTPUT Clause> ]
{ VALUES ( { DEFAULT | NULL | expression } [ ,...n ] ) [ ,...n ]
| derived_table
| execute_statement
| <dml_table_source>
| DEFAULT VALUES } }
[; ]
```

# Вставка данных в таблицы – команда **INSERT**

Наиболее часто команда INSERT применяется в двух вариантах:

- Вставка в заданную таблицу или представление **новой строки (строк) непосредственно из команды**;
- Вставка **новых строк на основе запроса** по другим таблицам (в сочетании с дополнительным оператором SELECT).

**Вариант 1: Синтаксис команды**:

INSERT [INTO] имя таблицы [(список имен столбцов)]

VALUES (список значений) **[, (список значений) ]**

Если список имен столбцов не указан, то будут **заполнены все столбцы** в порядке, указанном в команде **CREATE TABLE**,

иначе требуется указать значения для заполняемых столбцов, в том числе обязательно для имеющих ограничение NOT NULL, остальные поля строки получат значение NULL или DEFAULT.

# Вставка данных в таблицы – команда **INSERT**

Наиболее часто команда INSERT применяется в двух вариантах: (**1**)

- Вставка в заданную таблицу **новой строки (строк) из команды**;

Если в аргументе **column_list** указаны не все столбцы таблицы или представления, то в пропущенные столбцы вставляется либо значение по умолчанию (если для столбца оно задано), или значение NULL.

**Для всех пропущенных столбцов** должно быть либо определено значение по умолчанию, либо допускаться значение **NULL**.

Возможные ошибки, приводящие к откату команды:

- В изменяемой таблице уже есть строки с такими данными в одном или нескольких столбцах и для них имеются ограничения «уникальные значения» (первичные ключи);
- В изменяемом столбце имеется ограничения на значения (например CHECK (mark BETWEEN 2 AND 5)).

# Вставка данных в таблицы – команда **INSERT**

Наиболее часто команда INSERT применяется в двух вариантах: (**1**)

- Вставка в заданную таблицу **новой строки (строк) из команды**;

Этот **пример вставляет одну строку** в таблицу Production.UnitMeasure.

Так как значения для всех столбцов предоставлены и перечислены в том же порядке, что и столбцы таблицы, имена столбцов не указаны в параметре column_list.

```
INSERT  [INTO]  Production.UnitMeasure
VALUES (N'F2', N'Square Feet', GETDATE());
```

Этот пример вставляет несколько строк в таблицу Production.UnitMeasure.

```
INSERT INTO Production.UnitMeasure
VALUES (N'F2', N'Square Feet', DEFAULT), (N'Y2', N'Square Yards', DEFAULT );
```

При желании порядок перечисления значений столбцов можно изменить, но при этом нужно указать их имена во фразе **INTO** в желаемом порядке.

# Вставка данных в таблицы – команда **INSERT**

Команда INSERT может применяться в двух вариантах: **(вариант 2)**

- Вставка **новых строк на основе запроса** по другим таблицам (в сочетании с дополнительным оператором SELECT).

**Синтаксис команды**:

INSERT INTO имя таблицы [(список столбцов)]
SELECT ….(выборка строк из другой таблицы или таблиц)

При выполнении команды ограничения проверяются для каждой вставляемой строки и в случае их нарушения хотя бы для одной из строк **все данные не будут вставлены** – автоматически СУБД выполнится команда **ROLLBACK – откат**.

```
INSERT INTO marks (cod_st, cod_sub)
SELECT students.cod_st, subject.cod_sub
FROM students, subject WHERE cod_sub=5
```

В данном примере в таблицу оценок будут добавлены все студенты, а в столбец «предметы» будет помещен предмет с кодом 5.

# Вставка данных в таблицы – команда **INSERT**

Команда INSERT может применяться в двух вариантах: (вариант 2)

- Вставка **новых строк на основе запроса** по другим таблицам (в сочетании с дополнительным оператором SELECT).

Вставка с использованием выражения **TOP** используется для добавления верхних N строк одной таблицы в другую таблицу.

```
INSERT TOP (#) INTO SomeTableA
SELECT SomeColumnX, SomeColumnY
FROM SomeTableB
```

Чтобы вернуть вставленные строки как часть операции вставки можно использовать **OUTPUT** с выражением **INSERT**.

Пример:
```
INSERT SomeTable
OUTPUT dml_select_list INTO
(@table_variable | output_table) (column_list)
```

# Раздел 4. Основы языка SQL

## Тема 4.3

## Язык манипулирования данными (Data Manipulation Language)

**Вопросы лекции:**

1. Запросы вставки строк данных INSERT.
2. **Запросы на удаление строк таблиц DELETE.**
3. Запросы модификации данных UPDATE.

# Команда **DELETE**

Удаляет одну или несколько строк **целиком** из таблицы или представления.

**Синтаксис команды:**

DELETE FROM имя_таблицы | имя_представления
[ WHERE условие отбора строк для удаления ] или очистка таблиц

```
Удалить всех студентов с фамилией Иванов
DELETE FROM students
WHERE name_st='Иванов'
```

Нельзя удалить строку родительской таблицы, если есть связанные строки в подчиненной таблице и действует правило (ограничение) ссылочной целостности **ON DELETE RESTRICT**.

Если действует правило (ограничение) ссылочной целостности **ON DELETE CASCADE** - будут автоматически удаляться и все связанные строки в подчиненных таблицах.

# Команда **DELETE**

DELETE может быть использован без WHERE.

При этом **удаляются все строки** таблицы **без ограничений**.

Таблица, из которой удалены все строки, **остается в базе данных** и при желании может быть удалена из базы данных с помощью инструкции DROP TABLE.

Пример: DELETE FROM Sales.SalesPerson;

**Для полной очистки таблицы** без ее удаления есть более эффективная команда, работающая быстрее (**не во всех СУБД**)

TRANCATE TABLE имя таблицы,

например : TRANCATE TABLE **students**

# Команда **DELETE**

## DELETE с использованием подзапроса

DELETE может быть определен как подзапрос для того, чтобы удалить строки из базовой таблицы в зависимости от данных, хранящихся **в другой таблице**.

В примере удаляются строки из таблицы SalesPersonQuotaHistory на основе года до даты продажи, указанных в таблице SalesPerson.

```
DELETE FROM Sales.SalesPersonQuotaHistory
WHERE SalesPersonID IN
(SELECT SalesPersonID
FROM Sales.SalesPerson
WHERE SalesYTD > 2500000.00);
```

# Команда **DELETE**

**DELETE** с использованием **TOP**

DELETE может быть изменен с помощью **TOP** так же, как **INSERT** , для того, чтобы удалить некоторое количество или процент строк из таблицы.

В примере удаляется 2,5 процента строк из таблицы ProductionInventory.

```
DELETE TOP (2.5) PERCENT
FROM Production.ProductInventory;
```

# Команда **DELETE**

## **DELETE** с **OUTPUT**

Фраза **OUTPUT** возвращает данные из строк, изменившихся в результате выполнения инструкций INSERT, UPDATE, DELETE или MERGE, или выражения на основе этих данных.

Результаты могут быть возвращены приложению, например, для вывода подтверждающих сообщений, архивирования и т. п.

Результаты также могут быть вставлены в таблицу или табличную переменную. Кроме того, можно записать результаты предложения OUTPUT во вложенных инструкциях INSERT, UPDATE, DELETE или MERGE и вставить эти результаты в целевую таблицу или представление.

Пример: DELETE Production.Culture **OUTPUT** DELETED.*;

Результат:

| **CultureID** | **Name** | **ModifiedDate** |
|---|---|---|
| Ar | Arabic | 1998-06-01 |
| En | English | 1998-06-01 |

# Раздел 4. Основы языка SQL

## Тема 4.3

## Язык манипулирования данными (Data Manipulation Language)

**Вопросы лекции:**

1. Запросы вставки строк данных INSERT.
2. Запросы на удаление строк таблиц DELETE.
3. **Запросы модификации данных UPDATE.**

# Команда **UPDATE**

Данная команда не изменяет количества строк в таблице, а только изменяет содержимое строк в указанных столбцах (или всей строки).

**Синтаксис команды:**

**UPDATE** tab_name

{**SET** столбец_1 = {значение | DEFAULT | NULL} [,...n]

[ FROM имена_таблиц условия_выборки] - **таблицы – источники**

[WHERE условие_отбора_строк] - строки, подлежащие обновлению

Например:

Прибавим всем студентам по одному баллу к каждой оценке

```
UPDATE marks SET mark= mark+1
```

С помощью инструкции UPDATE данные можно модифицировать **только в одной таблице**.

# Команда **UPDATE**

При выполнении данная команда автоматически проверяет все ограничения заданных столбцов.

Если хотя бы для одной изменяемой строки оно выполняется, **все действия**, уже выполненные командой будут отменены и дальнейшее ее **выполнение прекращается**.

Чтобы гарантированно выполнить задание предыдущего примера можно задать эту команду таким образом:

```
UPDATE marks SET mark= mark+1
WHERE mark<5
```

С помощью фразы SET можно присваивать не только конкретные значения, но также значения NULL и DEFAULT, если в данных столбцах обновляемой таблицы они допустимы или были определены

# Команда **UPDATE**

## **UPDATE** с использованием **OUTPUT**

Чтобы вернуть обновленные строки как часть операции обновления можно использовать OUTPUT в инструкции UPDATE.

Результаты могут быть возвращены в панель результатов SQL Server Management Studio для просмотра или в табличную переменную для использования в клиентских приложениях БД.

Пример: **DECLARE** @NewTableVar table ( Dollars money );

```
UPDATE Sales.SalesPerson
SET Bonus = 10000
OUTPUT INSERTED.Bonus INTO @NewTableVar;
```

Результат:

```
Dollars money
-----------------
10000.00
```