

# Базы данных

## Лекция 16

Программирование в СУБД SQL Server  
(язык Transact-SQL)

Индексы. DCL.

# Ограничители

[ ]

Пример:

```
CREATE TABLE [Temp SELECT Table]  
([Column 1] int PRIMARY KEY,  
[Column 2] int);
```

```
SELECT *  
FROM [Temp SELECT TABLE];
```

```
DROP TABLE [Temp SELECT Table];
```

# Триггеры

- Триггеры представляют собой хранимые процедуры, которые вызываются автоматически при наступлении определенных событий (как правило, выполнении DML-операторов).
- Основные задачи
  - проверка корректности введенных данных и выполнение сложных ограничений целостности данных;
  - выдача предупреждений, напоминающих о необходимости выполнения некоторых действий при обновлении таблицы;
  - накопление аудиторской информации посредством фиксации сведений о внесенных изменениях и тех лицах, которые их выполнили;
  - поддержка репликации.

# Триггеры

Создание триггера:

```
CREATE TRIGGER <имя_триггера>  
ON { <таблица> }  
{ AFTER | INSTEAD OF }  
{ [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }  
AS  
<тело_триггера>
```

Удаление триггера:

```
DROP TRIGGER <имя_триггера>;
```

# Триггеры

Inserted – таблица со строками, которые будут вставлены по завершению транзакции триггера;

Deleted - таблица со строками, которые будут удалены по завершению транзакции триггера;

@@rowcount – количество строк, обработанных последней командой;

ROLLBACK TRANSACTION – отмена изменений, которые пытается выполнить пользователь.

# Триггеры

## Пример

```
CREATE TRIGGER CheckInsertTrigger
ON Deal AFTER INSERT AS
IF @@ROWCOUNT=1
BEGIN
    IF NOT EXISTS(SELECT * FROM inserted WHERE
inserted.Quantity<=ALL(SELECT Wharehouse.Rest FROM Wharehouse,
Deal WHERE Wharehouse.ItemCode= Deal.ItemCode))
        BEGIN
            ROLLBACK TRAN; PRINT 'Transaction was cancelled!'
        END
    END
END
```

# Курсоры

**Курсор** в SQL – это область в памяти базы данных, которая предназначена для хранения последнего оператора SQL.

**Статический курсор.** Информация читается из базы данных один раз и хранится в виде моментального снимка, поэтому изменения, внесенные в базу данных другим пользователем, не видны. В *статический курсор* внести *изменения* невозможно, поэтому он всегда открывается в режиме "только для чтения".

**Динамический курсор.** При использовании *динамических курсоров* не создается полная копия исходных данных, а выполняется динамическая выборка из исходных таблиц только при обращении пользователя к тем или иным данным.

# Курсоры

## **Объявление курсора**

```
DECLARE имя_курсора CURSOR [LOCAL | GLOBAL]
[FORWARD_ONLY | SCROLL] [STATIC | KEYSET | DYNAMIC]
[READ_ONLY | OPTIMISTIC]
FOR SELECT_оператор [FOR UPDATE [OF имя_столбца[,...n]]]
```

## **Пример**

```
DECLARE ClientCurs CURSOR SCROLL FOR SELECT * FROM Client;
DECLARE @ClientCursVar CURSOR;
SET @ClientCursVar=ClientCurs;
Либо можно при присваивании указывать описание:
SET @ClientCursVar=CURSOR LOCAL SCROLL FOR SELECT *
FROM Client;
```

# Курсоры

## *Открытие курсора*

```
OPEN {[GLOBAL] имя_курсора }  
| @имя_переменной_курсора}
```

## **Пример**

```
OPEN ClientCurs;
```

```
OPEN @ClientCursVar;
```

# Курсоры

## ***Выборка данных***

```
FETCH [[NEXT | PRIOR | FIRST | LAST | ABSOLUTE  
{номер_строки | @переменная_номера_строки} |  
RELATIVE {номер_строки |  
@переменная_номера_строки}]  
FROM ]{{[GLOBAL ]имя_курсора } |  
@имя_переменной_курсора }  
[INTO @имя_переменной [,...n]]
```

## **Пример**

```
FETCH NEXT FROM ClientCurs INTO @Id, @FirstName;
```

# Курсоры

## ***Выборка данных***

@@FETCH\_STATUS – статус курсора

- 0, если *выборка* завершилась успешно;
- 1, если *выборка* завершилась неудачно вследствие попытки *выборки* строки, находящейся за пределами *курсора* ;
- 2, если *выборка* завершилась неудачно вследствие попытки обращения к удаленной или измененной строке.

## **Пример**

```
WHILE @@FETCH_STATUS=0
```

```
BEGIN
```

```
SELECT @message='Client: '+@Id + ' ' + @FirstName
```

```
FETCH NEXT FROM ClientCurs INTO @Id, @FirstName
```

```
END
```

# Курсоры

## *Изменение и удаление данных*

- UPDATE имя\_таблицы SET {имя\_столбца={ DEFAULT | NULL | выражение}}[,...n] WHERE CURRENT OF {[GLOBAL] имя\_курсора} | @имя\_переменной\_курсора}
- DELETE имя\_таблицы WHERE CURRENT OF {[GLOBAL] имя\_курсора} | @имя\_переменной\_курсора}

# Курсоры

## ***Закрытие курсора***

- CLOSE {имя\_курсора | @имя\_переменной\_курсора}

```
CLOSE ClientCurs;
```

## **Освобождение курсора**

- DEALLOCATE { имя\_курсора | @имя\_переменной\_курсора }

```
DEALLOCATE ClientCurs;
```

# Индексы

Команды специфичны для различных СУБД.

Создание индекса

- CREATE [UNIQUE] INDEX <имя>  
ON <таблица> (<столбец [DESC] [,...]>)

Назначение – ускорение поиска данных

Удаление индекса

- DROP INDEX <имя>

Изменение индекса

- ALTER INDEX { <имя> | ALL } ON < таблица >  
{ REBUILD | DISABLE }

В ряде случаев индексы создаются автоматически, например, при наличии фраз GROUP BY, ORDER BY, при слиянии таблиц и др.

# Разграничение полномочий. DCL

Основные понятия: пользователи, объекты, права (привилегии).

Предоставление привилегий

- GRANT {<привилегия>[,...n] | ALL PRIVILEGES} ON имя\_объекта TO {<идентификатор\_пользователя> [,...n] | PUBLIC} [ WITH GRANT OPTION]
- <привилегия> ::= {SELECT | DELETE | INSERT [(имя\_столбца [,...n])]} | UPDATE [(имя\_столбца[,...n])]} | REFERENCES [(имя\_столбца[,...n])]} | EXECUTE }

Пример

- GRANT SELECT, INSERT, DELETE, UPDATE ON Student TO User1

# Разграничение полномочий. DCL

Изъятие привилегий

- REVOKE [GRANT OPTION FOR] {<привилегия>[,...n] | ALL PRIVILEGES} ON имя\_объекта FROM {<идентификатор\_пользователя> [,...n] | PUBLIC} [RESTRICT | CASCADE]

Пример

- REVOKE ALL PRIVILEGES ON Session FROM PUBLIC