Курсоры

Курсоры

Курсор - это результирующий набор данных, сформированный сервером базы данных, в котором можно выполнять операции над отдельными строками.

Курсоры могут быть реализованы на различных уровнях

на уровне T-SQL - используются внутри ХП, триггеров и сценариев:

на уровне API - в интерфейсах доступа к БД (ODBC, OLE DB и др.) и используются в приложениях как специальные объекты.

Курсоры создаются на базе оператора SELECT

Типы курсоров в T-SQL

Курсоры различаются по предоставляемым возможностям (моделям поведения)

В T-SQL имеются следующие типы курсоров

Статические курсоры (snapshot cursor)

Динамические курсоры (dynamic cursor)

Последовательные курсоры (forward-only cursor)

Ключевые курсоры (keyset cursor)

Статические курсоры T-SQL

Статические курсоры – это копия строк, выбранных из таблиц по запросу и размещенных в системной базе tempdb.

При открытии курсора устанавливаются блокировки на все строки, включаемые в набор

Изменения, вносимые в выбранные строки курсора другими пользователями, не отражаются в курсоре

Вносить изменения в курсор нельзя, он считывается в режиме «только чтение»

Динамические курсоры T-SQL

Динамические курсоры – это строки из таблиц запроса, данные из которых выбираются только при обращении к ним.

При обращении к строке курсора производится блокировка соответствующих строки в соответствующих таблицах

Изменения, вносимые другими пользователями после открытия курсора, но до обращения к строке, в ней будут отражаться, но после обращения – не будут

Можно вносить изменения в курсор и они будут автоматически вноситься в таблицы БД

Последовательные курсоры T-SQL

Последовательные курсоры – это строки из таблиц запроса, данные из которых выбираются только при обращении к ним и обращение к которым выполняется только от начала к концу.

При обращении к строке курсора производится блокировка соответствующих строки в соответствующих таблицах

Изменения, вносимые другими пользователями после открытия курсора, но до обращения к строке, в ней будут отражаться, но после обращения – не будут

Вносить изменения в курсор нельзя, он считывается в режиме «только чтение»

Ключевые курсоры T-SQL

Ключевые курсоры – это набор уникальных ключей, размещенных в системной базе tempdb и определяющих строки запроса, по которым производится доступ к данным в таблицах

При обращении к строке курсора производится блокировка соответствующих строки в соответствующих таблицах

Изменения, вносимые другими пользователями в строки курсора после его открытия, в курсоре отражаются, добавленные строки - не отображаются, а удаленные строки будут показываются как поврежденные

Вносить изменения в курсор нельзя, он считывается в режиме «только чтение»

Управление курсорами T-SQL

При работе с курсором выполняются следующие операции:

1. **Создание (объявление) курсора**
2. **Открытие (заполнение данными) курсора**
3. **Выборка данных из курсора**
4. **Изменение данных в курсоре (если позволяет тип)**
5. **Закрытие курсора (отключение данных)**
6. **Удаление курсора**

Создание курсора в T-SQL

```
DECLARE cursor_name CURSOR
[ LOCAL | GLOBAL ]
[ FORWARD_ONLY | SCROLL ]
[ STATIC | KEYSET | DYNAMIC | FAST_FORWARD ]
[ READ_ONLY | SCROLL_LOCKS | OPTIMISTIC ]
[ TYPE_WARNING ]
FOR select_statement
[ FOR UPDATE [ OF column_name [ ,...n ] ] ]
```

Не все параметры совместимы между собой, например, SCROLL_LOCKS и FAST_FORWARD, FAST_FORWARD и SCROLL или FOR_UPDATE

Параметры

LOCAL – локальный курсор, видимый только внутри триггера, ХП

GLOBAL– глобальный курсор, существующий до закрытия соединения

FORWARD_ONLY – последовательный курсор

SCROLL– просматриваемый в любых направлениях курсор

STATIC, KEYSET, DYNAMIC, FAST_FORWARD – тип курсора

READ_ONLY– курсор только для чтения

SCROLL_LOCKS– курсор для изменения

OPTIMISTIC – блокирует изменение и удаление строк в БД после открытия курсора

FOR UPDATE– курсор для изменения

Открытие курсора в T-SQL

OPEN { [GLOBAL] *cursor_name* | *cursor_variable_name* }

Количество строк в открытом курсоре сохраняется в глобальной переменной @@CURSOR_ROWS (n – количество строк в наборе, -n – курсор загружается и на текущий момент загружено n строк, 0 – нет строк, -1 – курсор динамический и количество строк неизвестно**)**

Получение данных курсора в T-SQL

FETCH [[NEXT | PRIOR | FIRST | LAST | ABSOLUTE { *n* | @*nvar* }
| RELATIVE { *n* | @*nvar* }]
FROM] { [GLOBAL] *cursor_name* | @*cursor_variable_name* }
[INTO @*variable_name* [,...*n*]]

Имя курсора

Переменные, к которые заносятся считываемые значения

Параметры

NEXT | PRIOR – чтение следующей (предыдущей) строки за текущей (после открытия курсора указатель находиться над 1-й строкой)

FIRST | LAST – выбирается 1-я (последняя) строка и она же становиться текущей

ABSOLUTE – выбирается n-я строка от начала (если n – положительное) или от конца (если n – отрицательное) набора

RELATIVE – выбирается строка, находящаяся через n строк от текущей

Состояние выполнения последней команды FETCH сохраняется в глобальной переменной @@FETCH_STATUS (0 – успешная выборка, -1 – выход за пределы результирующего набора, и т.д. **)**

Изменение данных курсора в T-SQL

```
UPDATE  table_name
      SET column_name = { expression | DEFAULT | NULL }
WHERE CURRENT OF   [ GLOBAL ] cursor_name
```

```
DELETE   [ FROM ] { table_name
WHERE  CURRENT OF  [ GLOBAL ] cursor_name
```

Имя курсора

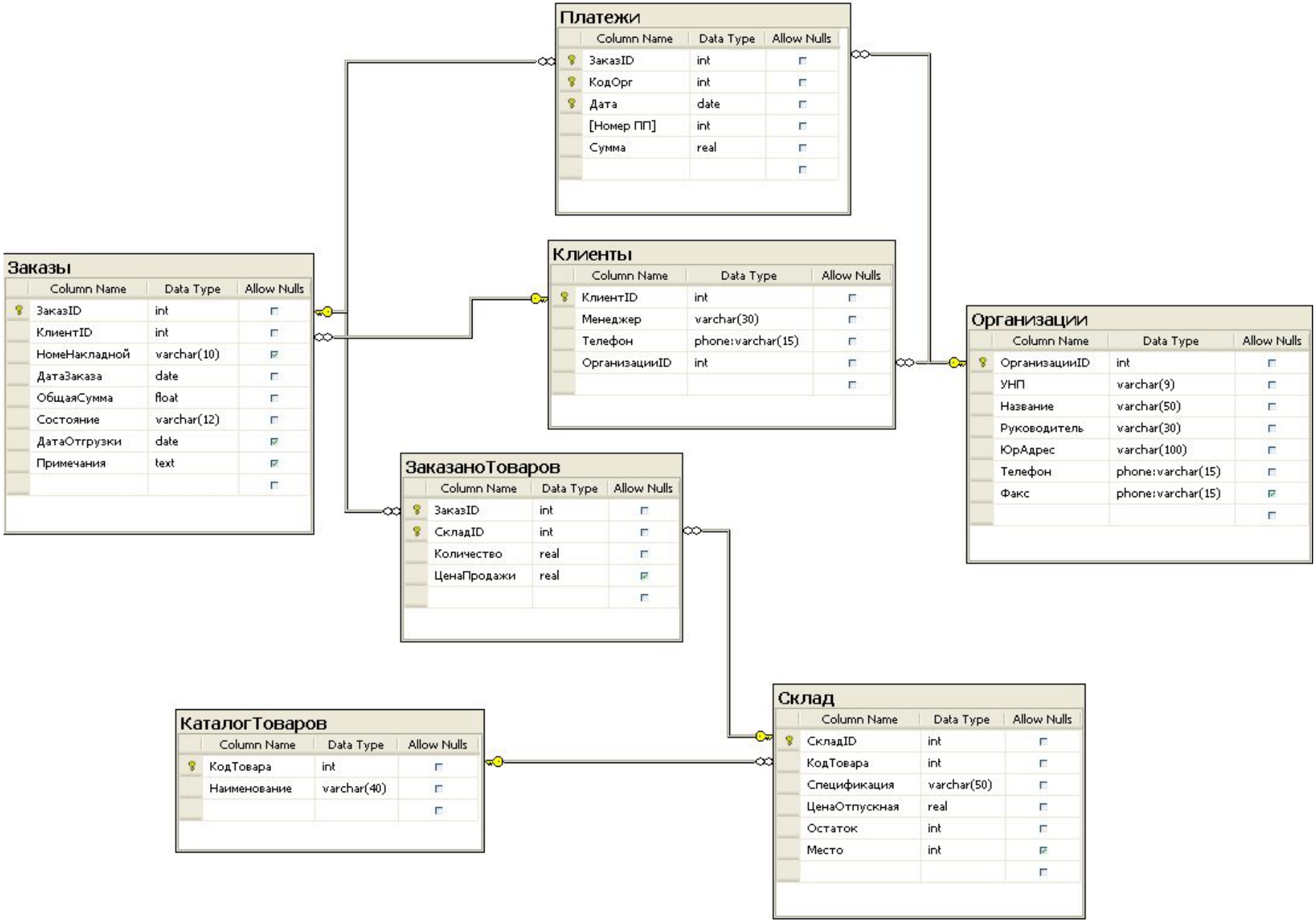
Закрытие курсора в T-SQL

CLOSE { [GLOBAL] *cursor_name* | *cursor_variable_name* }

Удаление курсора в T-SQL

DELOCATE { [GLOBAL] ***cursor_name*** **|** ***cursor_variable_name*** **}**

Схема БД «Заказы»



Пример курсора в T-SQL

```
CREATE  FUNCTION РасчетСтоимЗаказа (@КодЗаказа int)
RETURNS float
AS
BEGIN
   DECLARE @СуммаЗаказа float, @СтоимостьТовара float
   DECLARE cur CURSOR LOCAL FORWARD_ONLY STATIC
   FOR
      SELECT Количество*ЦенаПродажи FROM ЗаказаноТоваров
      WHERE ЗаказID= @КодЗаказа
  OPEN cur
  SET @СуммаЗаказа = 0
  SET @СтоимостьТовара = 0
  FETCH NEXT FROM cur INTO @СтоимостьТо
  WHILE @@FETCH_STATUS = 0
  BEGIN
     SET @СуммаЗаказа = @СуммаЗаказа + @СтоимостьТовара
     FETCH NEXT FROM cur INTO @СтоимостьТовара
  END
  CLOSE cur
  DEALLOCATE cur
  RETURN @СуммаЗаказа
END
```

Состояние текущей команды FETCH:
0 – успешная выборка
-1 – выход за пределы результирующего набора
-2 - строка помечена как удаленная
…
-9 – выборка еще не производилась

Пример курсора в T-SQL

```
CREATE  PROC СтоимостьЗаказа
@КодЗаказа int
AS
if exists (SELECT * FROM Заказы
            WHERE ЗаказID= @КодЗаказа and Состояние = 'оформление' )
UPDATE Заказы SET @ОбщаяСумма = РасчетСтоимЗаказа (@КодЗаказа)
WHERE ЗаказID= @КодЗаказа
```

Пример курсора в T-SQL

```
CREATE  PROC АннулированиеНеоплаченныхЗаказов
As
DECLARE @КодЗаказа int
DECLARE curКодыЗаказов CURSOR LOCAL STATIC
FOR
      SELECT ЗаказID FROM Заказы
      WHERE Состояние = 'оформление' AND
      DATEDIFF(day, ДатаЗаказа, Getdate()) >10
OPEN curКодыЗаказов
FETCH NEXT FROM curКодыЗаказов INTO @КодЗаказа
WHILE @@FETCH_STATUS = 0
BEGIN
   EXEC АннулированиеЗаказа @КодЗаказа
   FETCH NEXT FROM curКодыЗаказов INTO @КодЗаказа
END
CLOSE curКодыЗаказов
DEALLOCATE curКодыЗаказов
```

Функция возвращает разницу между 2-й и 1-й датами в значениях, указанных в 1-м параметре

Пример курсора в T-SQL

```
CREATE  PROC УдалениеЗаказовБезТоваров
As
DECLARE @КодЗаказа int
DECLARE curКодыЗаказов CURSOR LOCAL DYNAMIC
FOR
      SELECT ЗаказID FROM Заказы
      WHERE not Exists
          (SELECT * FROM ЗаказаноТоваров                WHERE
Заказы.ЗаказID = ЗаказаноТоваров.ЗаказID)
OPEN curКодыЗаказов
FETCH NEXT FROM curКодыЗаказов INTO @КодЗаказа
WHILE @@FETCH_STATUS = 0
BEGIN
   DELETE Заказы WHELE current of curКодыЗаказов
   FETCH NEXT FROM curКодыЗаказов INTO @КодЗаказа
END
CLOSE curКодыЗаказов
DEALLOCATE curКодыЗаказов
```

Имя курсора