Лабораторная работа 5:

Обработка данных

Добавление данных

Создание сценария в Query Editor

1. Откройте окно **Query Editor** (Редактор запросов), соединитесь с БД **ApressFinancial** (удостоверьтесь, что вошли с учетной записью, имеющей разрешение на вставку данных)**.**
2. 1ПКМ по таблице **ShareDetails.Shares** и выберите
   Script Table As → INSERT To → New Query Editor Window
   (Создать сценарий для таблицы | Используя INSERT | Новое окно редактора запросов).
3. Появится следующий код:

```
INSERT INTO [ApressFinancial].[ShareDetails].[Shares]
           ([ShareDesc]
           ,[ShareTickerID]
           ,[CurrentPrice])
     VALUES
           (<ShareDesc, nvarchar(50),>
           ,<ShareTickerID, nvarchar(50),>
           ,<CurrentPrice, numeric(18,5),>)
```

4. Отредактируйте код следующим образом:

```
SET QUOTED_IDENTIFIER OFF
GO
INSERT INTO [ApressFinancial].[ShareDetails].[Shares]
           ([ShareDesc]
           ,[ShareTickerID]
           ,[CurrentPrice])
     VALUES
           ("ACME'S HOMEBAKE COOKIES INC"
           ,'AHCI'
           ,2.34125)
```

5. Выполните код. Появится сообщение, что в таблицу была вставлена одна строка данных:

```
(1 row(s) affected)
```

Значения *NULL*

1. Запустите SSMS (удостоверьтесь, что вошли с учетной записью, которой разрешена вставка записей)**.**
2. В **Object Explorer** раскройте узел БД **ApressFinancial** до таблицы **CustomerDetails.Customers.** Перед добавлением значений, измените свойства ее столбцов:
   - для поля **CustomerId** свойство **(Is Identity)** установите **Yes** (Да);
   - для поля **DateAdded** укажите значение по умолчанию **(getdate()) –** дату и время будет вводить SQL Server при добавлении строки**;**

   Примечание. *Изменить свойства можно в* ***Table Designer*** *(Конструктор таблиц) или с помощью инструкций T-SQL.*
3. Для добавления значений в таблицу **CustomerDetails.Customers** щелкните по ней 1ПКМ и выберите команду **Open Table** (Открыть таблицу).
4. Справа от окна **Object Explorer** должны появиться все строки данных таблицы. Но так как таблица не содержит никаких данных, то вы увидите пустую сетку для ввода первой записи (обратите внимание на звездочку в левой части).
5. Введите одну запись: проанализируйте, какие столбцы допускают пропуск в заполнении, а какие должны быть обязательно заполнены правильными (!) значениями.
   Примечание. *Обратите внимание на то, что идентификатор в поле со свойством* **IDENTITY** *генерируется независимо от того, успешна вставка записи или нет.*
6. Теперь откройте окно **Query Editor** (Редактор запросов) и напишите следующий код:

```
USE ApressFinancial
GO
```

```
INSERT INTO CustomerDetails.Customers
  (CustomerTitleId, CustomerLastName, CustomerFirstName,
   CustomerOtherInitials, AddressId, AccountNumber,
   AccountTypeId, ClearеBalance, UnclearеBalance)
VALUES (3, 'Lobel', 'Leonard', NULL, 145, 53431993, 1, 437.97, -10.56)
```

7. Этот код должен успешно добавить строку в таблицу. Сравните этот код с выполненным кодом в предыдущем задании. Есть ли отличия?

Инструкции проверки целостности *DBCC (DataBase console commands)*

Инструкции проверки целостности БД могут использоваться для многих операций, например, для работы со столбцами IDENTITY. Если вы обнаружите, что при тестировании столбцов IDENTITY возникает целый ряд ошибок, а в результате удаления записей или ошибок при добавлении записей номера идентификатора «слишком скачут», можно переустановить начальное значение. Синтаксис такой инструкции:

```
DBCC CHECKIDENT ('table_name' [,{NORESEED | {RESEED[, new_reseed_value}}])
```

где *table_name* – имя таблицы, в которой нужно переустановить значение идентификатора; NORESEED можно использовать для возвращения к тому, что является текущим максимальным значением идентификатора в столбце IDENTITY; RESEED без значений задает автоматическую переустановку, или, дополнительно указав определенное значение *new_reseed_value*, можно установить это значение.

Переустановка начального значения для столбца IDENTITY несет в себе возможную опасность: если установить начальное значение, предшествующее самому большому значению, то после добавления записей может сгенерироваться значение, которое уже существует в этом столбце, что приведет к ошибке.

В приведенном ниже коде сначала удаляются все записи в таблице, затем значение для столбца IDENTITY снова переустанавливаете в 0, и информацию о клиентах заново добавляется:

```
USE ApressFinancial
GO
DELETE FROM  CustomerDetails.Customers
DBCC CHECKIDENT ('CustomerDetails.Customers',RESEED,0)
INSERT INTO CustomerDetails.Customers
  (CustomerTitleId, CustomerLastName, CustomerFirstName,
   CustomerOtherInitials, AddressId, AccountNumber,
   AccountTypeId, ClearеBalance, UnclearеBalance)
VALUES (1, 'Brust', 'Andrew', 'J.', 133, 18176111, 1, 200.00, 2.00),
       (3, 'Lobel', 'Leonard', NULL, 145, 53431993, 1, 437.97, -10.56)
```

Должно появиться сообщение:

```
(2 row(s) affected)
Checking identity information: current identity value '2', current column value '0'.
DBCC execution completed. If DBCC printed error messages, contact your system administrator.

(1 row(s) affected)

(1 row(s) affected)
```

Вставка значений в столбцы *Identity*

Самостоятельно откройте и выполните сценарий из файла **SQLQuery_Insert_to_Identity.sql** Проанализируйте результаты выполнения.

Ограничения для столбцов

1. Откройте окно **Query Editor** (Редактор запросов)**.**
2. Добавьте ограничения к таблице **CustomerDetails.CustomersProducts**. Для этого в окне запросов введите и выполните следующий код:

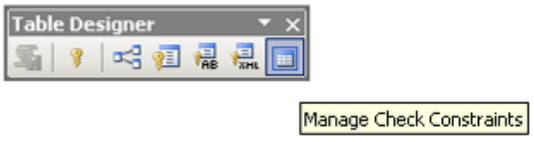
```
USE ApressFinancial
GO
ALTER TABLE CustomerDetails.CustomersProducts
ADD  CONSTRAINT PK_CustomersProducts
     PRIMARY KEY CLUSTERED
    (CustomerFinancialProductID)
ON [PRIMARY]
GO
```

```
ALTER TABLE CustomerDetails.CustomersProducts
WITH NOCHECK
ADD  CONSTRAINT CK_CustProds_AmtCheck
CHECK  (AmountToCollect > 0 )
GO

ALTER TABLE CustomerDetails.CustomersProducts
WITH NOCHECK
ADD  CONSTRAINT DF_CustProds_Renewable
  DEFAULT (0)
  FOR Renewable
GO
```

3. Таким образом, вы добавили:
   - первичный кластеризованный ключ;
   - ограничение для столбца **AmountToCollect**, которое гарантирует, что с этого момента для всех вставляемых записей долг должен быть больше нуля;
   - ограничение DEFAULT для столбца **Renewable**, т.е. будет вставлено значение 0, если в этот столбец не будет введено значение (это означает, что полученная премия является разовой);

   Примечание. *Параметр **NOCHECK** уточняет, что для уже вставленных записей это ограничение проверяться не будет.*
4. Обновите панель **Object Explorer**. Вы должны увидеть эти ограничения: одно под узлом **Keys** (Ключи) и два под узлом **Constraints** (Ограничения).
5. *Добавьте еще одно ограничение для этой таблицы*. Для этого откройте таблицу **CustomerDetails.CustomersProducts** в режиме **Design** (Конструктор таблиц). С помощью кнопки **Manage Check Constraints** (Управление проверочными ограничениями) откройте ОД **Check Constraints** (Проверочные ограничения). С помощью кнопки **Add** (Добавить) добавьте еще одно ограничение, которое будет гарантировать, что дата в **LastCollection** (когда последний раз получили платеж) была больше, чем дата в **LastCollected** (когда последний платеж должен быть получен).Для этого в поле **Expression** (Выражение) введите следующее ограничение: LastCollection >= LastCollected. Закройте ОД, чтобы увидеть и это ограничение в узле **Constraints** (Ограничения), закройте таблицу, сохранив изменения.



6. *Проверьте работоспособность ограничений*. Введите и попробуйте выполнить следующие инструкции (проанализируйте результаты):

```
INSERT INTO CustomerDetails.CustomersProducts
  (CustomerId,FinancialProductID,AmountToCollect,Frequency
  ,LastCollected,LastCollection,Renewable)
VALUES (1, 1, -100, 0, '24 Aug 2010', '24 Aug 2010', 0)

INSERT INTO CustomerDetails.CustomersProducts
  (CustomerId,FinancialProductID,AmountToCollect,Frequency
  ,LastCollected,LastCollection,Renewable)
VALUES (1, 1, 100, 0, '24 Aug 2010', '20 Aug 2010', 0)
```

Одновременная вставка нескольких записей

Откройте окно **Query Editor** (Редактор запросов)**.** Введите и выполните следующий код:

```
INSERT INTO CustomerDetails.Customers
  (CustomerTitleId, CustomerFirstName, CustomerOtherInitials,
   CustomerLastName, AddressId, AccountNumber,
   AccountTypeId, CleareBalance, UnclearеBalance)
VALUES (3, 'Bernie', 'I', 'McGee', 314, 65368765, 1, 6653.11, 0.00),
       (2, 'Julie', 'A','Dewson', 2134, 81625422, 1, 53.32, -12.21),
       (1, 'Kirsty', NULL,'Hull', 4312, 96565334, 1, 1266.00, 10.32);
```

```
INSERT INTO ShareDetails.Shares
    (ShareDesc, ShareTickerId,CurrentPrice)
VALUES ('FAT-BELLY.COM','FBC',45.20),
       ('NetRadio Inc','NRI',29.79),
       ('Texas Oil Industries','TOI',0.455),
       ('London Bridge Club','LBC',1.46);
```

Извлечение данных: отображение результатов

1. Откройте окно **Query Editor** (Редактор запросов)**.** Введите и выполните следующий код:

```
USE ApressFinancial
GO

SELECT * FROM CustomerDetails.Customers

SELECT CustomerFirstName, CustomerLastName, CleareBalance
FROM CustomerDetails.Customers

SELECT CustomerFirstName AS 'First Name',
       CustomerLastName AS 'Last Name',
       CleareBalance 'Balance'
FROM CustomerDetails.Customers
```

2. Теперь в меню **Query** (Запрос) выберите
   Results To → Results To Text (Отправить результаты | В виде текста)
   и выполните снова запросы. Вы увидите следующий результат:

```
First Name                                         Last Name                                          Balance
-------------------------------------------------- -------------------------------------------------- --------------
Andrew                                             Brust                                              200,00
Leonard                                            Lobel                                              437,97
Bernie                                             McGee                                              6653,11
Julie                                              Dewson                                             53,32
Kirsty                                             Hull                                               1266,00

(5 row(s) affected)
```

Обновление данных

1. Откройте окно **Query Editor** (Редактор запросов)**.** Введите и выполните следующий код:

```
USE ApressFinancial
GO

UPDATE CustomerDetails.Customers
SET CustomerLastName = 'Brodie'
WHERE CustomerId = 4
```

   Проверьте, обновилось ли значение.

2. Можно обновить данные с помощью информации из другого столбца таблицы или из переменной. Введите и выполните следующий код:

```
USE ApressFinancial
GO
DECLARE @ValueToUpdate VARCHAR(30)
SET @ValueToUpdate = 'McGlynn'
UPDATE CustomerDetails.Customers
SET CustomerLastName = @ValueToUpdate,
    CleareBalance = CleareBalance + UnclearеBalance,
    UnclearеBalance = 0
WHERE CustomerLastName = 'Brodie'
```

3. Можно попытаться обновить столбцы значениями из столбцов, типы данных которых не совпадают. Введите и выполните следующий код:

```
USE ApressFinancial
GO
DECLARE @WrongDataType VARCHAR(20)
SET @WrongDataType = '4311.22'
UPDATE CustomerDetails.Customers
SET CleareBalance = @WrongDataType
WHERE CustomerId = 4
```

Код выполнится без ошибки, т.к. SQL Server выполнил внутреннее (неявное) преобразование данных из значения типа `varchar` к типу `money`.
Никогда не поручайте SQL Server выполнять преобразование типов данных! Используйте функции CAST и CONVERT.

Удаление данных

1. Создайте в БД **tempdb** таблицу, используя предложение INTO. Для этого в окне **Query Editor** (Редактор запросов) введите и выполните следующий код:
```
USE tempdb
GO
SELECT CustomerId, CustomerFirstName,
       CustomerOtherInitials, CustomerLastName
INTO dbo.Customers
FROM ApressFinancial.CustomerDetails.Customers
```
   Вы должны получить сообщение: `(5 row(s) affected)`
2. Теперь удалите из созданной таблицы запись для `CustomerId = 4`
```
DELETE FROM dbo.Customers
WHERE CustomerId = 4
```
3. Откройте таблицу и добавьте запись (введите любые данные). Для новой записи должно сгенерироваться значение `CustomerId = 6`
4. Удалите из созданной таблицы все записи:
```
DELETE FROM dbo.Customers
```
   Вы должны получить сообщение: `(5 row(s) affected)`
5. Снова откройте таблицу `dbo.Customers` и добавьте запись. Для новой записи должно сгенерироваться значение `CustomerId = 7`
6. Удалите из созданной таблицы все записи, используя инструкцию TRUNCATE TABLE:
```
TRUNCATE TABLE dbo.Customers
```
   Вы должны получить сообщение: `(Command(s) completed successfully.`
7. Введите в таблицу `dbo.Customers` новую запись. Для новой записи сгенерируется значение `CustomerId = 1` (инструкция TRUNCATE TABLE заново создает начальное значение для столбцов IDENTITY).