

**МУ**

# МЕТОДИЧЕСКИЕ УКАЗАНИЯ

## РАСПРЕДЕЛЕННЫЕ БАЗЫ ДАННЫХ

Книги ФБ СПб ГЭТУ "ЛЭТИ"



**U13469**

Санкт-Петербург  
2008

Санкт-Петербургский государственный  
электротехнический университет «ЛЭТИ»

## ПОДГОТОВКА К ВЫПОЛНЕНИЮ ВЫБОРКИ ИЗ ТАБЛИЦЫ

Цель работы: знакомство с концепцией SQL и ее базами данных в среде открытия базы данных Microsoft SQL Server.

Задание: создать базу данных, содержащую таблицу, в которой хранятся данные о студентах, и выполнить запрос к ней для получения информации о студентах, у которых оценка по предмету "Информатика" не ниже 4,5 баллов.

1. Откройте SQL Server Management Studio и вспомните базу данных Books.
2. Напишите запрос, извлекающий значения полей name, title, rate из таблицы #t1.
3. Добавьте строку WHERE rate >= 4.5 в конец запроса. Результат должен состоять из 10 строк.

## РАСПРЕДЕЛЕННЫЕ БАЗЫ ДАННЫХ

### Методические указания к лабораторным работам

5. В текущем занятии необходимо выполнить строки с помощью оператора SELECT этого языка, методом написания которого является использование оператора SELECT. Результат выполнения будет выведен в виде таблицы.
6. Для каждого строки, должна существовать одна строка с обратной скобкой. Необходимо подбросить значение полей name и title из таблички #t1 в таблицу #t2, чтобы в дальнейшем можно было использовать эти значения в дальнейших действиях.
- Выполните логику, написавшие строки, содержащие операторы NOT, OR, AND, LIKE, DATE, а также таблицы #t1 и #t2 должны поменять значения своих частей.

Использование: в лаборатории для выполнения выборок. В этом упражнении необходимо выполнить запросы, которые включают в себя все предыдущие.

Санкт-Петербург

Издательство СПбГЭТУ «ЛЭТИ»

2008

Распределенные базы данных: Методические указания к лабораторным работам / Сост.: А. В. Горячев, Н. Е. Новакова. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2008. 32 с.

Содержат описания лабораторных работ по созданию запросов и программированию объектов базы данных.

Предназначены для студентов специальности 220300, а также могут быть полезны инженерно-техническим работникам этой области знаний.

Утверждено

редакционно-издательским советом университета  
в качестве методических указаний

Эффективность работы любой информационной системы во многом определяется структурами и способами хранения и обработки данных. В лабораторных работах рассматриваются базовые концепции обработки реляционных данных с помощью языка Transact-SQL.

## Лабораторная работа 1

### ВЫПОЛНЕНИЕ ВЫБОРКИ ИЗ ТАБЛИЦЫ

*Цель работы:* знакомство с командой SELECT и ее опциями. В лабораторной работе используется база данных (БД) Library.

#### Порядок выполнения

*Упражнение 1 – извлечение данных из таблиц БД:*

1. Откройте SQL Server Management Studio и в списке БД укажите Library.
2. Напишите запрос, извлекающий значения полей title и title\_no из таблицы title.
3. Добавьте ограничение для извлекаемых в запросе строк. Результат должен содержать название книги, для которой значение поля title\_no = 10.
4. Напишите запрос к таблице loanhist, представляющей номера читательских билетов и размер штрафа (поле fine\_assessed) тех читателей, которые имеют штраф в диапазоне от 8.00 до 9.00 \$.
5. В следующем запросе необходимо выбрать строки с использованием списка значений. Из таблицы title выберите автора и номер книги. Причем автор может быть либо Charles Dickens, либо Jane Austen.
6. Далее выберите строки, используя сравнение со строкой символов. Необходимо выбрать значения полей title и title\_no из таблицы title так, чтобы название включало слово «Adventures».
7. Выполните запрос, возвращающий строки, содержащие значения NULL для поля fine\_paid. Запрос из таблицы loanhist должен возвращать значения номера читателя библиотеки, величины штрафа и оплаты.

*Упражнение 2 – управление результирующими наборами.* В этом упражнении необходимо написать и выполнить запросы, которые изменяют способ отображения данных.

## 1. Использование опции DISTINCT:

- 1) напишите запрос, который извлекает уникальные пары названий городов и штатов из таблицы Adult;
- 2) отсортируйте данные по названию книг, представленных в таблице title.

## 2. Вычисление значений, применение псевдонимов для вычисляемых полей:

- 1) напишите и выполните запрос, который извлекает из таблицы loanhist, следующие поля: member\_no, isbn и fine\_assessed для всех значений поля fine\_assessed, не равных NULL;
- 2) добавьте к списку выборки удвоенное значение поля fine\_assessed. Введите для этого поля псевдоним 'double fine'.

## 3. Форматирование результирующего набора с помощью функций обработки строк:

- 1) напишите запрос, представляющий значения полей firstname, middleinitial и lastname из таблицы member, как единое поле. Значения должны представляться для всех читателей библиотеки с фамилией Anderson;
- 2) используйте псевдоним email\_name для результата объединения значений столбцов;
- 3) модифицируйте возвращаемое значение следующим образом: используйте функцию SUBSTRING для выделения первых двух символов фамилии, примените функцию LOWER ко всему возвращаемому значению для представления результата строчными буквами.

## 4. Обработка символьных значений:

- 1) напишите запрос, представляющий значения полей title и title\_no из таблицы title;
- 2) измените запрос так, чтобы результат выглядел следующим образом: 'The title is: Poems, title number 7'. Для формирования результата необходимо выполнить конкатенацию следующих компонентов:

- The title is: – строковая константа;
- title.title – значение поля;
- title number – строковая константа;
- title.title\_no – значение поля.

- 3) используйте функцию CONVERT для преобразования значения поля title.title\_no в символьную форму.

### Упражнение 3 – использование системных функций:

- Для определения идентификаторов серверных процессов запустите хранимую процедуру sp\_who без параметров.
- Выполните запрос:

```
SELECT @@spid
```

- Определите, кто запускает процесс с номером, полученным в предыдущем пункте:

```
EXEC sp_who n
```

- Выполните запрос:

```
SELECT @@version
```

- Выполните запрос:

```
SELECT USER_NAME(), DB_NAME(), @@servername
```

- Для извлечения метаданных о пользовательских таблицах напишите и выполните следующий запрос:

```
USE library
```

```
SELECT *
```

```
FROM information_schema.tables
```

```
WHERE table_type = 'base table'
```

## Лабораторная работа 2

### ГРУППИРОВКА И АГРЕГИРОВАНИЕ ДАННЫХ

**Цель работы:** знакомство с опциями GROUP BY и HAVING, а также агрегированием данных. В лабораторной работе используется база данных AdventureWorks.

#### Порядок выполнения

##### Упражнение 1 – использование ключевого слова TOP в команде SELECT.

В этом упражнении в команде SELECT используются ключевое слово TOP и предложение WITH TIES для возвращения части отсортированных значений из результирующего набора данных.

- Из таблицы Sales.SalesPerson выведите значения полей SalesPersonID и Bonus. Отсортируйте запрос по полю Bonus по убыванию.
- Модифицируйте код запроса таким образом, чтобы возвращались только 4 записи о значениях самых больших премий (бонусов) для продавцов.

### Упражнение 3 – использование системных функций:

- Для определения идентификаторов серверных процессов запустите хранимую процедуру sp\_who без параметров.
- Выполните запрос:

```
SELECT @@spid
```

- Определите, кто запускает процесс с номером, полученным в предыдущем пункте:

```
EXEC sp_who n
```

- Выполните запрос:

```
SELECT @@version
```

- Выполните запрос:

```
SELECT USER_NAME(), DB_NAME(), @@servername
```

- Для извлечения метаданных о пользовательских таблицах напишите и выполните следующий запрос:

```
USE library
```

```
SELECT *
```

```
FROM information_schema.tables
```

```
WHERE table_type = 'base table'
```

## Лабораторная работа 2

### ГРУППИРОВКА И АГРЕГИРОВАНИЕ ДАННЫХ

**Цель работы:** знакомство с опциями GROUP BY и HAVING, а также агрегированием данных. В лабораторной работе используется база данных AdventureWorks.

#### Порядок выполнения

##### Упражнение 1 – использование ключевого слова TOP в команде SELECT.

В этом упражнении в команде SELECT используются ключевое слово TOP и предложение WITH TIES для возвращения части отсортированных значений из результирующего набора данных.

- Из таблицы Sales.SalesPerson выведите значения полей SalesPersonID и Bonus. Отсортируйте запрос по полю Bonus по убыванию.
- Модифицируйте код запроса таким образом, чтобы возвращались только 4 записи о значениях самых больших премий (бонусов) для продавцов.

3. Модифицируйте запрос из предыдущего задания так, чтобы он возвращал строки не только со значениями первых четырех самых больших премий для продавцов, но и данные по тем продавцам, чьи премии имеют то же значение, что и последнее значение, полученное в предыдущем задании. В результате выполнения запроса должно получиться 5 строк.

*Упражнение 2 – использование агрегатных функций и конструкций GROUP BY и HAVING.*

1. Использование агрегатных функций:

- 1) подсчитайте общее количество строк в таблице Employee схемы HumanResources;
- 2) Подсчитайте общее количество сотрудников, имеющих менеджеров (поле ManagerID). Запрос выполняется по той же таблице.

2. Использование опции GROUP BY:

- 1) напишите запрос к таблице Sales.SalesOrderDetail, подсчитывающий суммарное количество заказанного товара (поле OrderQty) для каждого продукта (поле ProductID);
- 2) отсортируйте результат запроса по суммарному количеству заказанного товара. Выполните запрос и проанализируйте результат;
- 3) модифицируйте запрос таким образом, чтобы в результатирующем наборе попадали только те товары, суммарное значение заказов по которым не менее 2000. Выполните запрос и проанализируйте результат.

3. Использование предложения GROUP BY для формирования нескольких групп:

- 1) напишите запрос к таблице Sales.SalesOrderDetail, в списке SELECT которого должны быть представлены поля ProductID, SpecialOfferID, среднее значение цены за единицу товара (поле UnitPrice) и суммарное значение по полю LineTotal. Выполните группировку;
- 2) отсортируйте полученный результат по полю ProductID по возрастанию. Дайте псевдонимы тем элементам списка SELECT, которые соответствуют агрегированным значениям. Выполните группировку;
- 3) выполните запрос и проверьте результат.

*Упражнение 3 – использование операторов ROLLUP и CUBE:*

1. Использование оператора ROLLUP для создания сводных результатов:

- напишите запрос к таблице Sales.SalesPerson. В списке SELECT укажите поле SalesQuota и суммарное значение по полю SalesYTD. Выполните группировку. Дайте псевдоним TotalSalesYTD для суммы;
- измените запрос так, чтобы получать сводный результат по полученной выборке. Дополнительно примените функцию GROUPING;
- выполните запрос и проверьте результат. Обратите внимание на наличие строк со значениями NULL. Какой смысл этих значений NULL?

SalesQuota      TotalSalesYTD

NULL	1533087,5999
250000,00	33461260,59
300000,00	9299677,9445
NULL	44294026,1344

(4 row(s) affected)

## 2. Использование оператора CUBE для создания сводных результатов:

- напишите запрос к таблице Sales.SalesOrderDetail. В списке SELECT укажите поле ProductID и сумму по полю LineTotal. Выводить необходимо только те значения, для которых UnitPrice < \$5.00. Выполните сортировку и группировку по полю ProductID;
- модифицируйте запрос, добавьте оператор CUBE, а в группировку добавьте поле OrderQty;
- выполните запрос и проверьте результат:

ProductID    Total

NULL	86579.210714
923	7425.120000
923	7425.120000

(119 row(s) affected)

*Упражнение 4 – использование предложений COMPUTE и COMPUTE BY в команде SELECT для создания отчетов:*

- Напишите запрос к таблице Sales.SalesOrderHeader. В списке SELECT укажите поля SalesPersonID, CustomerID, OrderDate, SubTotal, TotalDue.
- Отсортируйте по полям SalesPersonID, OrderDate.

3. Измените запрос, используя COMPUTE так, чтобы вычислялись суммарные значения по полям SubTotal и TotalDue.

### Лабораторная работа 3

## ВЫПОЛНЕНИЕ ЗАПРОСОВ ПО НЕСКОЛЬКИМ ТАБЛИЦАМ

*Цель работы:* научиться соединять данные из нескольких таблиц.

### Порядок выполнения

*Упражнение 1* – создание списка почтовой рассылки с использованием оператора “JOIN”. Требуется создать список рассылки читателей библиотеки. Список должен включать полное имя и информацию о месте жительства читателя.

1. Откройте SQL Server Management Studio и в списке БД укажите Library.
2. Напишите такой запрос для таблиц member и adult, чтобы он возвращал значения полей firstname, middleinitial, lastname, street, city, state и zip. Значения полей firstname, middleinitial и lastname должны быть конкатенированы в один столбец с псевдонимом name.
3. Выполните запрос, убедитесь в правильности полученного результата, который должен выглядеть примерно так:

name	street	city	state	zip
Amy A Anderson	Bowery Estates	Montgomery	AL	36100
Brian A Anderson	Dogwood Drive	Sacramento	CA	94203
Daniel A Anderson	Fir Street	Washington	DC	20510
Eva A Anderson	The Highlands	Atlanta	GA	30026
Gary A Anderson	James Road	Springfield	IL	62700

5000 row(s) affected

*Упражнение 2* – объединение нескольких таблиц и сортировка результатов. Нужно выполнить запрос по таблицам title, item и copy, который возвращал бы поля isbn, copy\_no, on\_loan, title, translation и cover, а также строки из таблицы copy, где ISBN равен 1, 500 или 1000. Полученный набор должен быть отсортирован по полю isbn.

1. Напишите список SELECT. Указывайте имена столбцов при помощи псевдонимов таблиц. Псевдоним должен состоять как минимум из двух символов.

3. Измените запрос, используя COMPUTE так, чтобы вычислялись суммарные значения по полям SubTotal и TotalDue.

### Лабораторная работа 3

## ВЫПОЛНЕНИЕ ЗАПРОСОВ ПО НЕСКОЛЬКИМ ТАБЛИЦАМ

*Цель работы:* научиться соединять данные из нескольких таблиц.

### Порядок выполнения

*Упражнение 1* – создание списка почтовой рассылки с использованием оператора “JOIN”. Требуется создать список рассылки читателей библиотеки. Список должен включать полное имя и информацию о месте жительства читателя.

1. Откройте SQL Server Management Studio и в списке БД укажите Library.
2. Напишите такой запрос для таблиц member и adult, чтобы он возвращал значения полей firstname, middleinitial, lastname, street, city, state и zip. Значения полей firstname, middleinitial и lastname должны быть конкатенированы в один столбец с псевдонимом name.
3. Выполните запрос, убедитесь в правильности полученного результата, который должен выглядеть примерно так:

name	street	city	state	zip
Amy A Anderson	Bowery Estates	Montgomery	AL	36100
Brian A Anderson	Dogwood Drive	Sacramento	CA	94203
Daniel A Anderson	Fir Street	Washington	DC	20510
Eva A Anderson	The Highlands	Atlanta	GA	30026
Gary A Anderson	James Road	Springfield	IL	62700

5000 row(s) affected

*Упражнение 2* – объединение нескольких таблиц и сортировка результатов. Нужно выполнить запрос по таблицам title, item и copy, который возвращал бы поля isbn, copy\_no, on\_loan, title, translation и cover, а также строки из таблицы copy, где ISBN равен 1, 500 или 1000. Полученный набор должен быть отсортирован по полю isbn.

1. Напишите список SELECT. Указывайте имена столбцов при помощи псевдонимов таблиц. Псевдоним должен состоять как минимум из двух символов.

2. Запишите предложение FROM, создающее связывание INNER JOIN между таблицами title и copy по столбцам isbn.
3. Добавьте второе предложение INNER JOIN, обеспечивающее связывание между таблицами item и copy по столбцам isbn.
4. Напишите предложение WHERE так, чтобы усечь результирующий набор данных: нужны только те строки, в которых ISBN (таблица copy) равен 1, 500 или 1000.
5. Установите при помощи ORDER BY сортировку по полю ISBN по возрастанию. Выполните запрос и проверьте результат.

*Упражнение 3 – объединение таблиц с использованием OUTER JOIN.*

Необходимо выполнить запрос, возвращающий полное имя читателя member\_no из таблицы member, isbn и log\_date из таблицы reservation для читателей с номерами 250, 341 и 1675. Результат отсортировать по member\_no. Показать информацию об этих читателях вне зависимости от того, взяты ими книги или нет.

1. Для формирования списка SELECT:
  - создайте столбец name путем конкатенации полей lastname, firstname и middleinitial;
  - создайте столбец date, преобразовав log\_date к char(8).
2. Запишите предложение FROM, создающее связь OUTER JOIN между таблицами member и reservation.
3. Отсеките лишние записи предложением WHERE: из таблицы member выбираете только читателей с номерами 250, 341 и 1675.
4. Для представления даты используйте функцию CONVERT.
5. Отсортируйте записи по полю member.member\_no.

*Упражнение 4 – использование оператора UNION для соединения результирующих наборов:*

1. Требуется определить читателей, живущих в Аризоне, у которых более двух детей посещают библиотеку:
  - 1) напишите предложение SELECT, возвращающее member\_no и число записей на детей этого читателя. Полученное в вычисляемом поле значение представить псевдонимом (например, numkids). Сгруппировать результат по полю member\_no таблицы adult;
  - 2) выполните запрос. Не стирайте запрос.

- Нужно определить читателей, живущих в Калифорнии, у которых более трех детей ходят в библиотеку:
  - нажмите CTRL+N для создания нового окна запроса.
  - скопируйте предыдущий запрос и вставьте его в новое окно.
  - измените запрос в соответствии с новым заданием.
  - выполните запрос. Оцените результат.

Теперь следует объединить результаты разных запросов:

- нажмите CTRL+N;
- скопируйте первый запрос во второе окно;
- добавьте в конце запроса оператор UNION и вставьте второй запрос;
- выполните получившийся запрос. Обратите внимание на результат.

## Лабораторная работа 4

### МОДИФИКАЦИЯ ДАННЫХ

*Цель работы:* научиться выполнять команды вставки, удаления и обновления данных. В лабораторной работе используется БД Library.

#### Порядок выполнения

*Упражнение 1* – применение команды INSERT. В этом задании необходимо использовать команду INSERT для добавления строк в таблицы БД Librargy. После выполнения вставки данных надо написать запрос для проверки сделанных изменений.

- Вставка строк в таблицу item, которая представляет книги из собрания библиотеки:

- откройте SQL Server Management Studio;
- добавьте в таблицу item две строки для книги с заголовком номер 8, The Cherry Orchard. Укажите имена полей, значения которых вводите. Для остальных полей используйте следующие значения:

Номер строки	Имя поля				
	isbn	Title_no	Cover	Loanable	Translation
1	10001	8	HARDBACK	Y	ENGLISH
2	10101	8	SOFTBACK	Y	ENGLISH

2. Нужно определить читателей, живущих в Калифорнии, у которых более трех детей ходят в библиотеку:
- 1) нажмите CTRL+N для создания нового окна запроса.
  - 2) скопируйте предыдущий запрос и вставьте его в новое окно.
  - 3) измените запрос в соответствии с новым заданием.
  - 4) выполните запрос. Оцените результат.

Теперь следует объединить результаты разных запросов:

- 1) нажмите CTRL+N;
- 2) скопируйте первый запрос во второе окно;
- 3) добавьте в конце запроса оператор UNION и вставьте второй запрос;
- 4) выполните получившийся запрос. Обратите внимание на результат.

## Лабораторная работа 4

### МОДИФИКАЦИЯ ДАННЫХ

*Цель работы:* научиться выполнять команды вставки, удаления и обновления данных. В лабораторной работе используется БД Library.

#### *Порядок выполнения*

*Упражнение 1 – применение команды INSERT.* В этом задании необходимо использовать команду INSERT для добавления строк в таблицы БД Librargy. После выполнения вставки данных надо написать запрос для проверки сделанных изменений.

1. Вставка строк в таблицу item, которая представляет книги из собрания библиотеки:

- 1) откройте SQL Server Management Studio;
- 2) добавьте в таблицу item две строки для книги с заголовком номер 8, The Cherry Orchard. Укажите имена полей, значения которых вводите. Для остальных полей используйте следующие значения:

Номер строки	Имя поля				
	isbn	Title_no	Cover	Loanable	Translation
1	10001	8	HARDBACK	Y	ENGLISH
2	10101	8	SOFTBACK	Y	ENGLISH

- 3) напишите и выполните запрос к таблице item, чтобы убедиться в добавлении строк.
2. Вставка строк в таблицу copy, которая содержит данные о копиях книг в собрании библиотеки:
  - 1) добавьте строку в таблицу copy для элемента с твердым переплетом, который вы добавили в предыдущем задании. Используйте следующие значения для столбцов: ISBN 10001, copy\_no 1, title\_no 8, On\_loan N;
  - 2) выполните запрос для проверки операции вставки.
3. Определение языка, на который была переведен экземпляр книги из собрания библиотеки:
  - 1) напишите запрос, который возвратит язык (поле translation) одного из элементов, который был добавлен в предыдущих заданиях;
  - 2) выполните запрос и убедитесь в правильности выполненной ранее операции вставки.

*Упражнение 2 – использование команды INSERT с ключевым словом DEFAULT:*

1. Определение столбцов, для которых разрешено значение null. Выполните системную хранимую процедуру sp\_help для того, чтобы определить какие столбцы в таблице title допускают значение null. Для этих столбцов, а также для столбцов со значением default не обязательно указывать значение при добавлении строк. Значения полей со свойством IDENTITY формируются автоматически.
2. Вставка значений в таблицу title. Вставьте строку в таблицу title для книги *The Art of Lawn Tennis* автора William T. Tilden. Используйте ключевое слово DEFAULT для тех полей, которые разрешают значение null или имеют значение default. Не указывайте значение для поля title\_no, поскольку это поле имеет свойство IDENTITY. Напишите и выполните запрос, подтверждающий факт вставки.
3. Определение последнего использованного значения IDENTITY. Напишите запрос для определения значения поля title\_no для заголовка, добавленного в предыдущем задании.
4. Получение последней вставленной записи в таблице title. Напишите запрос, возвращающий последнюю запись, вставленную в таблицу title. Используйте результат предыдущего запроса.

- Добавление новых записей в таблицу title. Вставьте в таблицу title записи для названия книги *Riders of the Purple Sage* автора Zane Grey. Укажите список полей и соответствующие им значения для полей, не разрешающих null и не имеющих значений default.
- Проверка вставки значений в таблицу title. Напишите и выполните запрос для проверки того, что новое название книги и автор были успешно добавлены.

*Упражнение 3 – Использование команды INSERT с ключевыми словами DEFAULT VALUES.*

- Создание новой таблицы sample1. Выполните приведенный ниже запрос для создания новой таблицы sample1 в базе данных library:

```
USE LIBRARY  
CREATE TABLE sample1 (  
    Cust_id int NOT NULL IDENTITY(100,5)  
    ,Name char(10) NULL )
```

- Вставка записи со значениями по умолчанию в таблицу sample1:

- создайте запрос, добавляющий запись в таблицу sample1. Не указывайте имен полей. Используйте ключевые слова DEFAULT VALUES;
- напишите запрос, чтобы убедиться в добавлении новой записи.

*Упражнение 4 – использование команды DELETE:*

- Напишите запрос, возвращающий запись из таблицы item. Свойства записи: мягкий переплет, isbn 10101, название – «*The Cherry Orchard*», title\_no 8.
- Удалите указанную запись.

*Упражнение 5 – использование команды UPDATE:*

- Получение записи, которую следует обновить. Напишите запрос, возвращающий пользователя библиотеки с номером 507 из таблицы member.
- Обновление записи. Перепишите запрос так, чтобы он изменял фамилию указанного читателя.

*Упражнение 6 – изменение данных на основе информации других таблиц:*

- Добавление нового юного читателя в базу данных. Разберитесь с приведенным далее запросом и выполните его. Оператор SET IDENTITY INSERT

используется для разрешения задания пользовательского значения для поля со свойством IDENTITY.

```
USE library
```

```
BEGIN TRANSACTION
```

```
SET IDENTITY_INSERT member ON
```

```
INSERT member (member_no, lastname, firstname, middleinitial)
```

```
VALUES ( 16101, 'Walters', 'B.', 'L' )
```

```
SET IDENTITY_INSERT member OFF
```

```
INSERT juvenile
```

```
VALUES ( 16101, 1, DATEADD(YY, -18, DATEADD(DD, -1,
```

```
GETDATE()) )
```

```
COMMIT TRANSACTION
```

2. Получение записей, которые должны быть перенесены из таблицы juvenile в таблицу adult:

1) напишите запрос, возвращающий значения полей member\_no из таблицы juvenile и street, city, state, zip и phone\_no из таблицы adult. Также включите в запрос текущую дату плюс 1 год. Соответствующее выражение: DATEADD (YY, 1, GETDATE());

2) запрос должен связывать таблицы juvenile и adult по полю member: juvenile.adult\_member\_no = adult.member\_no;

3) добавьте выражение WHERE, чтобы показывать только тех читателей из таблицы juvenile, которым более 18 лет. Используйте функцию DATEADD.

3. Вставка записей в таблицу juvenile из таблицы adult:

1) измените запрос так, чтобы полученные записи добавлялись в таблицу adult;

2) напишите запрос, подтверждающий добавление читателя с номером 16101 в таблицу adult.

4. Определение подлежащих удалению записей из таблицы juvenile. Напишите запрос, который связывает таблицы juvenile и adult следующим образом: juvenile.member\_no = adult.member\_no.

5. Удаление записей из таблицы juvenile:

1) измените запрос так, чтобы полученные записи удалялись;

2) убедитесь, что читатель 16101 был удален из таблицы juvenile.

## Лабораторная работа 5

### РАБОТА С ВЛОЖЕННЫМИ ЗАПРОСАМИ

*Цель работы:* научиться писать и применять вложенные запросы.

#### *Порядок выполнения*

*Упражнение 1 – использование вложенных запросов как производных таблиц. В лабораторной работе используется БД Library.*

1. Выполнение запроса, использующего производные таблицы. Запрос, формирующий производную таблицу, должен возвращать столбец juvenile.adult\_member\_no и количество подростков для каждого взрослого читателя библиотеки, имеющего более трех детей (подростков) записанных в библиотеку (являющихся ее читателями). Список раздела SELECT основного запроса должен включать поля adult\_member\_no и No\_Of\_Children из производного запроса и поле expr\_date из таблицы adult. Выполните запрос, использующий производную таблицу. Проверьте результат.
2. Создание запроса с производной таблицей в виде двух разных запросов. В этом случае необходимо переписать предыдущий запрос так, чтобы он выглядел как два разных запроса, показывающие работу с производными таблицами. Напишите запрос, который возвращает поле adult\_member\_no, вычисляет количество детей для каждого взрослого читателя и возвращает те записи, в которых число детей в таблице juvenile больше 3. Выполните запрос и проверьте результат. Сравните результат с результатом предыдущего запроса. Напишите ещё один запрос, который возвращает значения поле expr\_date таблицы adult. Перепишите запрос так, чтобы он использовал JOIN.

*Упражнение 2 – использование подзапросов как выражений:*

1. Использование подзапросов с одним значением (single-value). В этой процедуре нужно написать запрос, возвращающий значения полей member.firstname, member.lastname, loanhist.isbn и loanhist.fine\_paid для читателей библиотеки, заплативших максимальных штраф за все книги:
  - 1) напишите запрос, который возвращает максимальное значение поля loanhist.fine\_paid;
  - 2) выполните запрос и проверьте результат.

2. Использование подзапроса как части условия поиска:
- 1) напишите запрос, который соединяет таблицы member и loanhist и возвращает значения firstname, lastname, isbn и fine\_paid для всех строк;
  - 2) используйте запрос из шага 1 предыдущего задания как критерий выбора в предложении WHERE, так чтобы возвращались только те записи, в которых штраф имеет максимальное значение;
  - 3) включите ключевое слово DISTINCT;
  - 4) выполните запрос и проверьте результат:

```
Firstname    lastname    isbn    fine_paid
Michael      Nash        883     8.0000
Robert       Rothenberg  330     8.0000
(2 row(s) affected)
```

3. Использование запросов для создания списка значений. В этом задании нужно написать запрос к таблицам title, loan и reservation, возвращающий значения 4 полей: title\_no, title, isbn и Total Reserved. Поле Total Reserved представляет собой количество резервных экземпляров для каждой книги. Нужно отобразить те записи, которых в резерве или более 50, или менее 5.
- 1) Напишите запрос, возвращающий номера isbn из таблицы reservations для книг, у которых более 50 копий;
  - 2) выполните запрос и проверьте результат.
4. Использование подзапроса с несколькими значениями. Напишите внешний запрос, который возвращает поля title\_no, title, isbn и Total Reserved, где Total Reserved – это число записей для каждой группы остальных полей. Для этого:
- 1) ограничьте число записей, задав условие на количество копий книг, которое должно быть менее 5;
  - 2) используйте ключевое слово IN, как часть предложения WHERE для списка, созданного на шаге 1 предыдущего задания. Выполните запрос.

*Упражнение 3* – использование коррелированных подзапросов. В этом задании нужно создать запрос, использующий коррелированный подзапрос для вычисления значений, основанных на данных из внешнего запроса, и использующий эти значения как часть условия сравнения. Необходимо отобразить список читателей, имеющих сумму штрафов, превышающую 5 у. е.

1. Напишите внешний запрос, возвращающий значения полей member\_no и lastname. Используйте псевдоним для таблицы member.
2. Напишите внутренний запрос, вычисляющий общий штраф для каждого читателя. Для этого используйте псевдоним для таблицы loanhist.
3. Свяжите поля member.member\_no из внешнего запроса с полем loanhist.member\_no внутреннего подзапроса.
4. Используйте оператор сравнения в предложении WHERE.
5. Выполните запрос и проверьте результат.

## Лабораторная работа 6

### ОБЕСПЕЧЕНИЕ ЦЕЛОСТНОСТИ ДАННЫХ

*Цель работы:* научиться создавать таблицы, применять и отключать ограничения. В лабораторной работе используется БД AdventureWorks.

#### *Порядок выполнения*

*Упражнение 1 – создание новой таблицы и применение ограничений целостности:*

1. Создать новую таблицу с именем HumanResources.JobCandidateHistory.

Таблица должна содержать следующие столбцы и ограничения:

- JobCandidateID. Столбец с типом данных int. Этот столбец не может содержать пустые значения. Значения в этом столбце должны быть уникальны;
- Resume. Столбец с типом данных xml, может содержать пустые значения;
- Rating. Столбец с типом данных int, не может содержать пустые значения. Значения этого столбца должны находиться в диапазоне 1...10, значение по умолчанию – 5;
- RejectedDate. Столбец с типом данных datetime, не может содержать пустые значения;
- ContactID. Столбец с типом данных int, может содержать пустые значения. Этот столбец является внешним ключом для столбца ContactID в таблице Person.Contact.

1. Напишите внешний запрос, возвращающий значения полей member\_no и lastname. Используйте псевдоним для таблицы member.
2. Напишите внутренний запрос, вычисляющий общий штраф для каждого читателя. Для этого используйте псевдоним для таблицы loanhist.
3. Свяжите поля member.member\_no из внешнего запроса с полем loanhist.member\_no внутреннего подзапроса.
4. Используйте оператор сравнения в предложении WHERE.
5. Выполните запрос и проверьте результат.

## Лабораторная работа 6

### ОБЕСПЕЧЕНИЕ ЦЕЛОСТНОСТИ ДАННЫХ

*Цель работы:* научиться создавать таблицы, применять и отключать ограничения. В лабораторной работе используется БД AdventureWorks.

#### *Порядок выполнения*

*Упражнение 1 – создание новой таблицы и применение ограничений целостности:*

1. Создать новую таблицу с именем HumanResources.JobCandidateHistory.

Таблица должна содержать следующие столбцы и ограничения:

- JobCandidateID. Столбец с типом данных int. Этот столбец не может содержать пустые значения. Значения в этом столбце должны быть уникальны;
- Resume. Столбец с типом данных xml, может содержать пустые значения;
- Rating. Столбец с типом данных int, не может содержать пустые значения. Значения этого столбца должны находиться в диапазоне 1...10, значение по умолчанию – 5;
- RejectedDate. Столбец с типом данных datetime, не может содержать пустые значения;
- ContactID. Столбец с типом данных int, может содержать пустые значения. Этот столбец является внешним ключом для столбца ContactID в таблице Person.Contact.

- Проверить результат. В обозревателе объектов разверните папку DataBase, разверните AdventureWorks, разверните Tables и убедитесь, что в списке имеется таблица HumanResources.JobCandidateHistory. В обозревателе объектов разверните эту таблицу. Проверьте наличие столбцов и необходимых ограничений.
- Проверка таблицы JobCandidateHistory и ограничений:
  - с помощью меню File утилиты SQL Server Management Studio откройте файл TestConstraints.sql, который находится в папке C:\Labfiles;
  - выделите код, показанный под примечанием This should fail, и нажмите на панели инструментов кнопку Execute. Операция INSERT завершится ошибкой, поскольку значение Rating противоречит ограничению CHECK;
  - выделите код под примечанием This should succeed и нажмите на панели инструментов кнопку Execute. Операция INSERT выполнится успешно. Оставьте приложение Microsoft® SQL Server™ Management Studio открытym. Оно будет использоваться в следующем упражнении.

*Упражнение 2 – отключение ограничений:*

- Для отключения ограничений выполните следующие команды:

```
USE [AdventureWorks]
GO
ALTER TABLE HumanResources.JobCandidateHistory
    NOCHECK CONSTRAINT [CK_JobCandidateHistory_Rating]
GO
ALTER TABLE HumanResources.JobCandidateHistory
    NOCHECK CONSTRAINT [DF_JobCandidateHistory_Rating]
GO
```

- Откройте файл InsertTestData.sql из папки C:\Labfiles.
- Для таблицы HumanResources.JobCandidateHistory включите ограничения, выполнив следующие команды:

```
ALTER TABLE HumanResources.JobCandidateHistory
    CHECK CONSTRAINT [CK_JobCandidateHistory_Rating]
GO
ALTER TABLE HumanResources.JobCandidateHistory
    CHECK CONSTRAINT [DF_JobCandidateHistory_Rating]
GO
```

- Убедитесь, что команды выполнены успешно.

## Лабораторная работа 7

### ИСПОЛЬЗОВАНИЕ ПРЕДСТАВЛЕНИЙ

*Цель работы:* научиться писать и применять представления. В лабораторной работе используется БД AdventureWorks.

#### Порядок выполнения

*Упражнение 1 – создание представления.* В этом упражнении необходимо создать новое представление HumanResources.vEmployeeDetails, использующее следующую логику команды SELECT:

SELECT

```
e.[EmployeeID]
, c.[Title]
, c.[FirstName]
, c.[MiddleName]
, c.[LastName]
, c.[Suffix]
, e.[Title] AS [JobTitle]
, c.[Phone]
, c.[EmailAddress]
, c.[EmailPromotion]
, a.[AddressLine1]
, a.[AddressLine2]
, a.[City]
, sp.[Name] AS [StateProvinceName]
, a.[PostalCode]
, cr.[Name] AS [CountryRegionName]
, c.[AdditionalContactInfo]
```

FROM [HumanResources].[Employee] e  
INNER JOIN [Person].[Contact] c  
ON c.[ContactID] = e.[ContactID]  
INNER JOIN [HumanResources].[EmployeeAddress] ea

```
ON e.[EmployeeID] = ea.[EmployeeID]
INNER JOIN [Person].[Address] a
ON ea.[AddressID] = a.[AddressID]
INNER JOIN [Person].[StateProvince] sp
ON sp.[StateProvinceID] = a.[StateProvinceID]
INNER JOIN [Person].[CountryRegion] cr
ON cr.[CountryRegionCode] = sp.[CountryRegionCode]
```

Добавьте при определении представления параметр SCHEMABINDING. Этот параметр необходимо указывать при создании индексированного представления. Напишите запрос и проверьте работу представления.

*Упражнение 2* – создание индексированного представления. Выполните следующую команду:

```
USE AdventureWorks
```

```
GO
```

```
CREATE UNIQUE CLUSTERED INDEX IX_vEmployeeDetails
ON HumanResources.vEmployeeDetails (EmployeeID)
```

*Упражнение 3* – просмотр системной информации о представлениях. Просмотреть зависимости представлений от других объектов базы данных AdventureWorks с помощью системной хранимой процедуры `sp_depends`. Просмотреть программный код созданного в предыдущем упражнении представления с помощью системной хранимой процедуры `sp_helptext`.

## Лабораторная работа 8

### СОЗДАНИЕ И ИСПОЛЬЗОВАНИЕ ХРАНИМЫХ ПРОЦЕДУР

*Цель работы:* научиться писать и применять хранимые процедуры. В лабораторной работе используется БД AdventureWorks.

**Общие сведения.** Для хранения и оперативного управления данными порталов, как правило, используются системы управления базами данных. В этой работе рассмотрены технологии применения хранимых процедур, которые обеспечивают процедурную обработку данных на стороне сервера баз данных, упрощают доступ к обрабатываемым данным, обеспечивают безопасность хранения данных, повышают производительность их обработки, сокращают сетевой трафик.

```
ON e.[EmployeeID] = ea.[EmployeeID]
INNER JOIN [Person].[Address] a
ON ea.[AddressID] = a.[AddressID]
INNER JOIN [Person].[StateProvince] sp
ON sp.[StateProvinceID] = a.[StateProvinceID]
INNER JOIN [Person].[CountryRegion] cr
ON cr.[CountryRegionCode] = sp.[CountryRegionCode]
```

Добавьте при определении представления параметр SCHEMABINDING. Этот параметр необходимо указывать при создании индексированного представления. Напишите запрос и проверьте работу представления.

*Упражнение 2* – создание индексированного представления. Выполните следующую команду:

```
USE AdventureWorks
```

```
GO
```

```
CREATE UNIQUE CLUSTERED INDEX IX_vEmployeeDetails
ON HumanResources.vEmployeeDetails (EmployeeID)
```

*Упражнение 3* – просмотр системной информации о представлениях. Просмотреть зависимости представлений от других объектов базы данных AdventureWorks с помощью системной хранимой процедуры `sp_depends`. Просмотреть программный код созданного в предыдущем упражнении представления с помощью системной хранимой процедуры `sp_helptext`.

## Лабораторная работа 8

### СОЗДАНИЕ И ИСПОЛЬЗОВАНИЕ ХРАНИМЫХ ПРОЦЕДУР

*Цель работы:* научиться писать и применять хранимые процедуры. В лабораторной работе используется БД AdventureWorks.

**Общие сведения.** Для хранения и оперативного управления данными порталов, как правило, используются системы управления базами данных. В этой работе рассмотрены технологии применения хранимых процедур, которые обеспечивают процедурную обработку данных на стороне сервера баз данных, упрощают доступ к обрабатываемым данным, обеспечивают безопасность хранения данных, повышают производительность их обработки, сокращают сетевой трафик.

## Порядок выполнения

*Упражнение 1 – создание хранимой процедуры без параметров.* Создайте хранимую процедуру GetDiscounts в схеме Sales, которая извлекает следующие столбцы из таблицы Sales.SpecialOffer: Description, DiscountPct, Type, Category, StartDate, EndDate, MinQty и MaxQty. Процедура должна возвращать все строки, отсортированные по параметрам StartDate и EndDate. Для проверки хранимой процедуры введите команду:

```
EXEC Sales.GetDiscounts
```

*Упражнение 2 – создание хранимой процедуры с параметром.* Создайте хранимую процедуру GetDiscountsForCategory в схеме Sales. Эта процедура должна иметь входной параметр @Category, имеющий тип данных nvarchar и принимающий до 50 символов. Она должна извлекать те же столбцы, что и запрос в процедуре GetDiscounts, но фильтровать строки на основе параметра @Category. Для проверки хранимой процедуры введите команду:

```
EXEC Sales.GetDiscountsForCategory 'Reseller'
```

*Упражнение 3 – создание хранимой процедуры с параметрами и значениями по умолчанию.* В схеме Sales создайте хранимую процедуру GetDiscountsForCategoryAndDate. Эта процедура должна иметь входной параметр @Category, как и процедура GetDiscountsForCategory, но включает дополнительный входной параметр @DateToCheck datetime. Параметр @DateToCheck должен иметь возможность принимать стандартное значение NULL. Если значение NULL указано для параметра @DateToCheck, задайте для этого параметра значение текущих даты и времени, используя функцию GETDATE(). Для проверки хранимой процедуры введите команду

```
EXEC Sales.GetDiscountsForCategoryAndDate 'Reseller'
```

Необходимо также проверить вариант, когда указываются оба параметра:

```
DECLARE @DateToCheck datetime
```

```
SET @DateToCheck = DateAdd(month, 1, GetDate())
```

```
EXEC Sales.GetDiscountsForCategoryAndDate 'Reseller', @DateToCheck
```

*Упражнение 4 – создание хранимой процедуры с входными и выходными параметрами.* Создайте процедуру AddDiscount в схеме Sales, которая вставляет новые записи в таблицу Sales.SpecialOffer. В следующей таблице указаны требуемые параметры для вставки:

Название параметра	Тип данных
@Description	nvarchar(255)
@DiscountPct	smallmoney
@Type	nvarchar(50)
@Category	nvarchar(50)
@StartDate	Datetime
@EndDate	Datetime
@MinQty	int
@MaxQty	int
@NewProductID	int (output)

Команда INSERT должна быть защищена соответствующей обработкой ошибок, и любые ошибки должны быть записаны в таблицу dbo.ErrorLog. Если новая вставка завершится успешно, параметр @NewProductID должен быть обновлен на значение функции SCOPE\_IDENTITY. Возвращаемое значение должно также указывать успех или неудачу вставки. Для обработки ошибок используйте конструкцию Try...Catch.

Для проверки хранимой процедуры введите следующие команды:

```
DECLARE @StartDate datetime, @EndDate datetime
SET @StartDate = GetDate()
SET @EndDate = DateAdd(month, 1, @StartDate)
DECLARE @NewId int
EXEC Sales.AddDiscount
'Half price off everything',
0.5,
'Seasonal Discount',
'Customer',
@StartDate,
@EndDate,
0,
20,
@NewID OUTPUT
SELECT @NewID
```

Убедитесь, что в результатах запроса возвращается новое значение параметра SpecialOfferID.

Для повторной проверки хранимой процедуры введите инструкции, как показано в следующем примере кода Transact-SQL:

```
DECLARE @StartDate datetime, @EndDate datetime
SET @StartDate = GetDate()
SET @EndDate = DateAdd(month, 1, @StartDate)
DECLARE @NewId int, @ReturnValue int
EXEC @ReturnValue = Sales.AddDiscount
'Half price off everything',
-0.5,          -- UNACCEPTABLE VALUE
'Seasonal Discount',
'Customer',
@StartDate,
@EndDate,
0,
20,
@NewID OUTPUT
IF (@ReturnValue = 0)
SELECT @NewID
ELSE
SELECT TOP 1 * FROM dbo.ErrorLog ORDER BY ErrorTime DESC
```

Убедитесь, что в результатах запроса отображается запись об ошибке, включающая значение параметра ErrorProcedure, равное AddDiscount.

## Лабораторная работа 9

### СОЗДАНИЕ UDF

*Цель работы:* научиться писать и применять функции, определяемые пользователем (UDF). В лабораторной работе используются две базы данных: AdventureWorks и AdventureWorksDW. Перед выполнением работы откройте файл InitializeData.sql и выполните его.

Убедитесь, что в результатах запроса возвращается новое значение параметра SpecialOfferID.

Для повторной проверки хранимой процедуры введите инструкции, как показано в следующем примере кода Transact-SQL:

```
DECLARE @StartDate datetime, @EndDate datetime
SET @StartDate = GetDate()
SET @EndDate = DateAdd(month, 1, @StartDate)
DECLARE @NewId int, @ReturnValue int
EXEC @ReturnValue = Sales.AddDiscount
'Half price off everything',
-0.5,          -- UNACCEPTABLE VALUE
'Seasonal Discount',
'Customer',
@StartDate,
@EndDate,
0,
20,
@NewID OUTPUT
IF (@ReturnValue = 0)
SELECT @NewID
ELSE
SELECT TOP 1 * FROM dbo.ErrorLog ORDER BY ErrorTime DESC
```

Убедитесь, что в результатах запроса отображается запись об ошибке, включающая значение параметра ErrorProcedure, равное AddDiscount.

## Лабораторная работа 9

### СОЗДАНИЕ UDF

*Цель работы:* научиться писать и применять функции, определяемые пользователем (UDF). В лабораторной работе используются две базы данных: AdventureWorks и AdventureWorksDW. Перед выполнением работы откройте файл InitializeData.sql и выполните его.

## **Порядок выполнения**

*Упражнение 1* – создание скалярной функции. Создать определяемую пользователем скалярную функцию Sales.GetMaximumDiscountForCategory, которая находит максимальный процент скидки (поле DiscountPct), доступный на данный момент для конкретной категории. Создать параметр @Category nvarchar(50) для ограничения результатов на основе категории и использовать функцию GETDATE() для ограничения строк на основе дос-тупности скидки на данный момент в диапазоне StartDate и EndDate. Для проверки функции введите следующую команду:

```
SELECT Sales.GetMaximumDiscountForCategory('Reseller')
```

*Упражнение 2* – создание функции, возвращающей табличное значение (In-Line Table-valued UDF). Создать функцию Sales.GetDiscountsForDate , ко-торая находит те же столбцы, что и хранимая процедура GetDiscounts (лаб. раб. 8, упр. 1). У функции должен быть входной параметр @DateToCheck datetime. Он должен использоваться для фильтрации скидок на основе вве-денной даты. Это позволяет компании Adventure Works проверить, какие скидки будут доступны на указанную дату. Для проверки функции введите следующую команду:

```
SELECT *  
FROM Sales.GetDiscountsForDate(GetDate())  
ORDER BY DiscountPct DESC
```

*Упражнение 3* – создание функции, возвращающей табличное значение (Multi-Statement Table-valued UDF). Создать функцию GetDiscountedProducts в схеме Sales. В этой функции должен быть написан запрос для поиска про-дуктов, имеющих скидку. Для формирования этого запроса надо соединить таблицы: Sales.SpecialOfferProduct, Sales.SpecialOffer и Production.Product. Запрос должен выводить следующие данные: столбцы ProductID, Name, ListPrice из таблицы Production.Product, столбцы Description и DiscountPct из таблицы Sales.SpecialOffer, а также два вычисляемых столбца. Первый вы-числяемый столбец получается в результате произведения значений из поля ListPrice на DiscountPct; второй – в результате вычитания из ListPrice произ-ведения значений поля ListPrice на DiscountPct. У функции должен быть па-раметр @IncludeHistory bit, который применяется для фильтрации возвра-щенной таблицы на основе того, требуются ли сведения об истории скидок

и.ги необходимы только текущие сведения. Возвращаемая функцией таблица должна содержать следующее определение:

Название столбца	Тип данных
ProductID	int
Name	nvarchar(50)
ListPrice	money
DiscountDescription	nvarchar(255)
DiscountPercentage	smallmoney
DiscountAmount	money
DiscountedPrice	money

Для проверки функции введите следующие команды:

```
SELECT * FROM Sales.GetDiscountedProducts(0)
```

```
SELECT * FROM Sales.GetDiscountedProducts(1)
```

**Упражнение 4 – контроль контекста выполнения.** Откройте и выполните файл ExecutionContext.sql.

1. *Создание скалярной функции.* Создайте скалярную функцию GetCurrencyRate в схеме Sales, которая находит последнюю ставку конвертации валюты для указанной валюты. Эта функция принимает параметр @CurrencyCode nchar(3), чтобы указать валюту, для которой должна возвращаться ставка конвертации (тип данных float). Для обозначения ставки конвертации используйте переменную @CurrencyRate float; для выполнения запроса – соединение двух таблиц DimCurrency и FactCurrencyRate. Сведения о валюте находятся в БД AdventureWorksDW, и доступ к ним должен осуществляться с использованием учетной записи МАМ\Adam. Сведения о валюте хранятся в таблице DimCurrency БД Adventure-WorksDW, которая имеет следующее определение:

Название столбца	Тип данных
CurrencyKey (PK)	int, not null
CurrencyAlternateKey	nchar(3), not null
CurrencyName	nvarchar(50), not null

Данные ставки конвертации хранятся в таблице FactCurrencyRate, которая содержит следующие столбцы:

Название столбца	Тип данных
TimeKey (FK)	int, not null
CurrencyKey (FK)	int, not null
EndOfDayRate	float, not null

Столбец TimeKey в таблице FactCurrencyRate – это внешний ключ к таблице DimTime, которая содержит следующие столбцы.

Название столбца	Тип данных
TimeKey (PK)	int, not null
DayNumberOfYear	smallint, null

Для проверки функции введите следующую команду:

```
SELECT Sales.GetCurrencyRate('GBP')
```

2. Установление отношений доверия между базами данных. Чтобы функция GetCurrencyRate могла искать сведения о валюте, необходимо установить отношения доверия между базами данных AdventureWorks и AdventureWorksDW. Выполните следующие команды:

- 1) создание учетной записи пользователя в БД AdventureWorksDW

```
USE AdventureWorksDW
```

```
IF NOT EXISTS (SELECT * FROM sys.database_principals WHERE name = 'Adam' AND type = 'U')
```

```
CREATE USER Adam FOR LOGIN [MIAMI\Adam]
```

- 2) предоставление разрешения AUTHENTICATE для пользователя в БД AdventureWorksDW:

```
GRANT AUTHENTICATE TO Adam
```

- 3) предоставление разрешения SELECT пользователю Adam:

```
GRANT SELECT TO Adam
```

- 4) установка параметра базы данных TRUSTWORTHY для базы данных AdventureWorks:

```
ALTER DATABASE AdventureWorks SET TRUSTWORTHY ON
```

3. Создание и тестирование функции GetCurrencyDiscountedProducts. Создать функцию GetCurrencyDiscountedProducts в схеме Sales, возвращающую табличное значение. Эта функция аналогична по функциональности функции GetDiscountedProducts, но включает цену и цену со скидкой в альтернативной валюте. Для указания валюты применяется входной параметр @CurrencyCode nchar(3). Возвращаемая таблица должна содержать следующее определение:

Название столбца	Тип данных
ProductID	int
Name	nvarchar(50)
ListPrice	money
CurrencyPrice	money
DiscountDescription	nvarchar(255)
DiscountPercentage	smallmoney
DiscountAmount	money
DiscountedPrice	money
DiscountedCurrencyPrice	money

В теле функции объявит переменную @CurrencyRate float. Присвойте этой переменной значение, которое возвращается скалярной функцией Sales.GetCurrencyRate. Укажите @CurrencyCode в качестве входного параметра функции GetCurrencyRate. Выражение для первого вычисляемого столбца, формирующего цену (поле CurrencyPrice в возвращаемой таблице), получается как произведение ListPrice на @CurrencyRate. Выражение для последнего вычисляемого столбца, формирующего цену со скидкой, получается как произведение @CurrencyRate на результат вычитания из ListPrice произведения значений поля ListPrice на DiscountPct. Для проверки функции введите следующую команду:

```
SELECT * FROM Sales.GetCurrencyDiscountedProducts('GBP')
```

## Лабораторная работа 10

### УПРАВЛЕНИЕ ТРАНЗАКЦИЯМИ И БЛОКИРОВКАМИ

*Цель работы:* получить представление о транзакциях и блокировках. В этой работе применяется база данных AdventureWorks.

#### Порядок выполнения

*Упражнение 1 – применение транзакций:*

1. Откройте файл Tran1.sql. В сценарии производится обновление записи в таблице Person.Contact. Инструкции SELECT и глобальная переменная @@trancount используются, чтобы показать ход выполнения транзакции. Найдите комментарий START TRANSACTION HERE и добавьте команду BEGIN TRANSACTION.
2. Нажмите кнопку Execute, чтобы выполнить сценарий и затем просмотреть полученные результаты. На этом этапе транзакция по-прежнему активна,

Название столбца	Тип данных
ProductID	int
Name	nvarchar(50)
ListPrice	money
CurrencyPrice	money
DiscountDescription	nvarchar(255)
DiscountPercentage	smallmoney
DiscountAmount	money
DiscountedPrice	money
DiscountedCurrencyPrice	money

В теле функции объявит переменную @CurrencyRate float. Присвойте этой переменной значение, которое возвращается скалярной функцией Sales.GetCurrencyRate. Укажите @CurrencyCode в качестве входного параметра функции GetCurrencyRate. Выражение для первого вычисляемого столбца, формирующего цену (поле CurrencyPrice в возвращаемой таблице), получается как произведение ListPrice на @CurrencyRate. Выражение для последнего вычисляемого столбца, формирующего цену со скидкой, получается как произведение @CurrencyRate на результат вычитания из ListPrice произведения значений поля ListPrice на DiscountPct. Для проверки функции введите следующую команду:

```
SELECT * FROM Sales.GetCurrencyDiscountedProducts('GBP')
```

## Лабораторная работа 10

### УПРАВЛЕНИЕ ТРАНЗАКЦИЯМИ И БЛОКИРОВКАМИ

*Цель работы:* получить представление о транзакциях и блокировках. В этой работе применяется база данных AdventureWorks.

#### Порядок выполнения

*Упражнение 1 – применение транзакций:*

1. Откройте файл Tran1.sql. В сценарии производится обновление записи в таблице Person.Contact. Инструкции SELECT и глобальная переменная @@trancount используются, чтобы показать ход выполнения транзакции. Найдите комментарий START TRANSACTION HERE и добавьте команду BEGIN TRANSACTION.
2. Нажмите кнопку Execute, чтобы выполнить сценарий и затем просмотреть полученные результаты. На этом этапе транзакция по-прежнему активна,

и блокировки все еще удерживаются. Внесенные изменения не зафиксированы в базе данных.

3. Найдите комментарий END TRANSACTION HERE, добавьте и выполните команду COMMIT TRANSACTION для фиксации изменений в базе данных. Теперь изменения в базе данных зафиксированы.

*Упражнение 2 – выполнение отката транзакций:*

1. Откройте файл Tran2.sql. В сценарии с помощью команды UPDATE обновляется другая запись в таблице Person.Contact. Команда SELECT и глобальная переменная @@trancount используются, чтобы показать ход выполнения транзакции.
2. Найдите комментарий END TRANSACTION HERE, добавьте и выполните команду ROLLBACK TRANSACTION. Эта команда должна выполнить откат изменений в базе данных. Обратите внимание, что транзакция завершена, поскольку количество транзакций равно нулю, а изменения отменены.

*Упражнение 3 – просмотр сведений о блокировках:*

1. В среде SQL Server Management Studio в дереве консоли последовательно разверните узел Management уровня сервера, щелкните на Activity Monitor правой кнопкой мыши и выберите в контекстном меню опцию View Process. Просмотрите сведения, которые отображаются в разделах «Сведения о процессе» (Process Info), «Блокировки по процессам» (Locks by Process) и «Блокировки по объектам» (Locks by Objects). Закройте монитор активности.
2. Напишите запрос к динамическому представлению в текущем соединении:

```
SELECT resource_type, request_mode, request_type, request_status,  
request_session_id  
FROM sys.dm_tran_locks
```

3. Откройте файл Lock1.sql. В сценарии присутствует команда BEGIN TRANSACTION, но нет команд ROLLBACK TRANSACTION и COMMIT TRANSACTION.
4. Нажмите кнопку Execute, чтобы выполнить сценарий. Обратите внимание на принадлежащий соединению идентификатор SPID. Переключитесь обратно в окно с запросом к представлению sys.dm\_tran\_locks. Обратите внимание, что теперь идентификатором SPID второго соединения удер-

живаются блокировки на различных объектах. Посмотрите, как изменились сведения о процессе в мониторе активности. Перейдите в окно со сценарием Lock1.sql. Выполните команду ROLLBACK TRANSACTION. Снова переключитесь в окно с запросом к представлению sys.dm\_tran\_locks. Убедитесь, что блокировки сняты. Аналогичную информацию можно получить в мониторе активности.

*Упражнение 4 – настройка параметров блокировки:*

1. Откройте файл Lock2.sql. Просмотрите содержимое файла и обратите внимание, что уровень изоляции транзакций установлен в значение SERIALIZABLE, при котором пользователи не могут получить доступ к строкам, соответствующим условиям в операторе WHERE. Выполните запрос.
2. Откройте файл LockList.sql и выполните запрос. Обратите внимание на присутствие в столбце request\_status записи WAIT, указывающей на ожидание вторым запросом предоставления блокировки, прежде чем будет продолжено его выполнение.
3. Переключитесь на окно запроса Lock2.sql и нажмите кнопку Cancel Executing Query. Переключитесь на окно запроса LockList.sql и убедитесь, что ожидание транзакций отменено.
4. Установите таймаут блокировки. Для этого перейдите в окно запроса Lock2.sql и отредактируйте его, добавив следующую команду непосредственно перед командой BEGIN TRANSACTION:  
SET lock\_timeout 5000

5. Выполните запрос, подождите в течение пяти секунд. Обратите внимание, что запрос больше не ожидает неопределенно долго предоставления блокировки.
6. Закройте среду SQL Server Management.

## Лабораторная работа 11

### СОЗДАНИЕ ТРИГГЕРОВ

*Цель работы:* научится создавать триггеры. В этой работе применяется база данных AdventureWorks.

живаются блокировки на различных объектах. Посмотрите, как изменились сведения о процессе в мониторе активности. Перейдите в окно со сценарием Lock1.sql. Выполните команду ROLLBACK TRANSACTION. Снова переключитесь в окно с запросом к представлению sys.dm\_tran\_locks. Убедитесь, что блокировки сняты. Аналогичную информацию можно получить в мониторе активности.

*Упражнение 4 – настройка параметров блокировки:*

1. Откройте файл Lock2.sql. Просмотрите содержимое файла и обратите внимание, что уровень изоляции транзакций установлен в значение SERIALIZABLE, при котором пользователи не могут получить доступ к строкам, соответствующим условиям в операторе WHERE. Выполните запрос.
2. Откройте файл LockList.sql и выполните запрос. Обратите внимание на присутствие в столбце request\_status записи WAIT, указывающей на ожидание вторым запросом предоставления блокировки, прежде чем будет продолжено его выполнение.
3. Переключитесь на окно запроса Lock2.sql и нажмите кнопку Cancel Executing Query. Переключитесь на окно запроса LockList.sql и убедитесь, что ожидание транзакций отменено.
4. Установите таймаут блокировки. Для этого перейдите в окно запроса Lock2.sql и отредактируйте его, добавив следующую команду непосредственно перед командой BEGIN TRANSACTION:  
SET lock\_timeout 5000

5. Выполните запрос, подождите в течение пяти секунд. Обратите внимание, что запрос больше не ожидает неопределенно долго предоставления блокировки.
6. Закройте среду SQL Server Management.

## Лабораторная работа 11

### СОЗДАНИЕ ТРИГГЕРОВ

*Цель работы:* научится создавать триггеры. В этой работе применяется база данных AdventureWorks.

## *Порядок выполнения*

*Упражнение 1 – создание новой таблицы.* Напишите и выполните следующую команду:

```
USE [AdventureWorks]
GO
CREATE TABLE [HumanResources].[JobCandidateHistory](
    [JobCandidateID] [int] NOT NULL UNIQUE,
    [Resume] [xml] NULL,
    [Rating] [int] NOT NULL CONSTRAINT
    [DF_JobCandidateHistory_Rating] Default (5),
    [RejectedDate] [datetime] NOT NULL,
    [ContactID] [int] NULL,
    CONSTRAINT [FK_JobCandidateHistory_Contact_ContactID]
        FOREIGN KEY(ContactID) REFERENCES [Person].[Contact]
        (ContactID),
    CONSTRAINT [CK_JobCandidateHistory_Rating]
        CHECK ([Rating]>=0 AND [Rating]<=10)
) ON [PRIMARY]
```

*Упражнение 2 – создание триггера для таблицы JobCandidate схемы HumanResources.* Создайте триггер dJobCandidate. Триггер должен вставлять данные в таблицу JobCandidateHistory после выполнения удаления данных из таблицы JobCandidate. Триггер копирует сведения о кандидате, если их кто-нибудь удалит. Необходимо копировать столбцы JobCandidateID и Resume. В поле RejectedDate надо записывать текущую дату с помощью функции GETDATE(). В столбце Rating следует оставить значение по умолчанию, а столбцу ContactID присвоить значение NULL.

*Упражнение 3 – проверка работы триггера.* Выполните следующую команду:

```
USE AdventureWorks
GO
DELETE FROM HumanResources.JobCandidate
```

```
WHERE JobCandidateID =  
      (SELECT MIN(JobCandidateID) FROM  
HumanResources.JobCandidate)
```

В результате срабатывания триггера на удаление данные о кандидате должны быть скопированы в таблицу JobCandidateHistory. Выполните запрос:

```
SELECT * FROM [HumanResources].[JobCandidateHistory]
```

Удалите данные из таблицы JobCandidateHistory, выполнив команду:

```
TRUNCATE TABLE [HumanResources].[JobCandidateHistory]
```

*Упражнение 4 – создание триггера на обновление и вставку:*

- 1 Создайте триггер OrderDetailNotDiscontinued на таблицу Sales.SalesOrderDetail. Этот триггер должен отвергать попытки ввода заказов на товары, прием которых на склад прекращен. Информация о прекращении поставок товара находится в таблице Production.Product. Если поставки товара прекращены, то значение поля DiscontinuedDate будет иметь значение, отличное от NULL. При попытке заказать такой товар триггер должен выдать сообщение с помощью команды RAISERROR и откатить транзакцию.
- 2 Выполните проверку триггера. В базу данных необходимо ввести хотя бы одну строку, при обнаружении которой во время выполнения кода триггера должна активизироваться ошибка. Для этого в таблицу Production.Product необходимо ввести данные хотя бы об одном товаре, поставка которого прекращена. Проверим, есть ли подходящие данные в таблице Product. Выполните запрос:

```
USE AdventureWorks
```

```
GO
```

```
SELECT ProductID, Name FROM Production.Product
```

```
WHERE DiscontinuedDate IS NOT NULL
```

3. Если данных нет, то введите в строку изменения, выполнив запрос:

```
UPDATE Production.Product
```

```
SET DiscontinuedDate = GETDATE()
```

```
WHERE ProductID = 680
```

4. После завершения подготовительного этапа перейдите к проверке работы триггера, выполнив запрос:

## INSERT Sales.SalesOrderDetail

(SalesOrderID, OrderQty, ProductID, SpecialOfferID, UnitPrice, UnitPriceDiscount)

VALUES (43660, 5, 680, 1, 1431, 0)

В результате попытка ввода недопустимых данных должна быть отвергнута.

## Содержание

Лабораторная работа 1. Выполнение выборки из таблицы .....	3
Лабораторная работа 2. Группировка и агрегирование данных .....	5
Лабораторная работа 3. Выполнение запросов по нескольким таблицам .....	8
Лабораторная работа 4. Модификация данных .....	10
Лабораторная работа 5. Работа с вложенными запросами .....	14
Лабораторная работа 6. Обеспечение целостности данных .....	16
Лабораторная работа 7. Использование представлений .....	18
Лабораторная работа 8. Создание и использование хранимых процедур .....	19
Лабораторная работа 9. Создание UDF .....	22
Лабораторная работа 10. Управление транзакциями и блокировками .....	26
Лабораторная работа 11. Выполнение выборки из таблицы .....	28

1. Время работы над проектом. Время работы в среде Microsoft SQL Server Management Studio. Если побывало время на проектирование, то пишите какое. Рассмотрим Дел. Будет многое меняться, отличное от СИ. Тут можно было бы сказать, какой товар рентабельнее, например сообщения о продажах ИМПЕРИАЛ и открыть транзакции.
2. Выполните проверку транзакций. В базе данных введём данные о продаже яблок, цена которых, за которую мы покупаем яблоки у фермера, составляет 100 рублей. Для этого в таблицу Редиссия Рено в информационной системе должны ходить обновления товаров, несмотря на которое произошло. Проверка, есть ли обновления в таблице Банкноты, не приводит к ошибкам.

УСП Администрации

ОГ

Редактор Н. В. Лукина

---

Подписано в печать 09.10.08. Формат 60×84 1/16. Бумага офсетная.

Печать офсетная. Гарнитура «Times». Печ. л. 2,0.

Тираж 80 экз. Заказ 91

---

После звонка: Издательство СПбГЭТУ «ЛЭТИ»  
197376, С.-Петербург, ул. Проф. Попова, 5

# МЕТОДИЧЕСКИЕ УКАЗАНИЯ

М