Лабораторная работа 3

Цель работы: Выборка данных из нескольких таблиц.

Структурированный язык запросов позволяет получать данные из нескольких таблиц одновременно. Существует простой и понятный синтаксис подобных запросов, но для полного понимания необходимо рассмотреть несколько общих вопросов.

Допустим пользователь имеет две таблицы. Первая таблица будет называться Корма, и в ней один столбец - название корма. Для примера в таблице будет две строки: сено, мясо. Вторая таблица будет называться Животные, и так же будет иметь один столбец – название животного, в данном случае будет две строки: собака, лошадь.

Вполне допустим следующий запрос:

```
SELECT [название корма], [название животного] FROM Корма, Животные
```

Однако полученный результат вряд ли устроит пользователя:

сено собака

сено лошадь

мясо собака

мясо лошадь

Для того что бы результат соответствовал реальному миру, СУБД должно иметь возможность исключить не существующие в реальном мире варианты сочетания корма и животного. Модифицируем наши таблицы. Добавим в таблицу Корма столбе – номер корма, это будет первичный ключ для этой таблицы. В таблицу Животные необходимо будет добавить два столбца. Столбец – номер животного, станет первичным ключом этой таблицы, а еще один столбец – номер корма, станет внешним ключом, связанным с первичным ключом таблицы Корма. Естественно, значения в столбце – номер корма таблицы Животные должен содержать значения, которые удовлетворяют двум требованиям. Во-первых, есть точно такие же значения в столбце – номер корма таблицы Корма. Во-вторых, эти значения соответствуют реальному миру, т.е. в строке где содержится информация о лошади, должен стоять число равное номеру сена, а в строке где упомянута собака номер ее корма должен соответствовать номеру мяса.

Используемые определения: суперключ, потенциальный ключ, первичный и внешний ключ.

Суперключ – атрибут или множество атрибутов, единственным образом идентифицирующие кортеж

Потенциальный ключ –суперключ, который не содержит подмножества, также являющегося суперключем данного отношения. Отношение может иметь несколько потенциальных ключей. Если потенциальный ключ состоит из нескольких атрибутов – он называется составной ключ.

Первичный ключ – потенциальный ключ, который выбран для уникальной идентификации кортежей внутри отношения.

Внешний ключ – атрибут или множество атрибутов внутри отношения, которое соответствует потенциальному ключу некоторого (может быть того же самого) отношения. Внешние ключи позволяют описать связь между отношениями.

После добавления столбцов и внесения необходимых данных можно выполнить запрос следующего вида:

```
SELECT [название корма], [название животного] FROM Корма, Животные WHERE Корма.[Номер корма]=Животные.[Номер корма]
```

Он даст результат, который соответствует реальному миру:

сено лошадь

мясо собака

В данном случае СУБД получила однозначную инструкцию не включать в результирующий набор данных, полученный из двух таблиц те строки, для которых не выполняется равенство внешнего и первичного ключей.

Такой способ соединения таблиц допустим. Пользователь может производить соединения двух, и более, таблиц используя сравнения на равенство, или не равенство, значений столбца одной таблицы со значениями столбца другой таблицы. Надо отметить, что подобные соединения не подразумевают что столбцы обязательно являются первичным и внешним ключом соответственно. У подобного соединения есть один существенный недостаток. Операция сравнения на равенство не применима к значениям NULL. Допустим у нас появилось информация о животном, например шимпанзе, для которой корм не определен. В итоговую выборку это животное не попадет. Для разрешения подобной трудности используется операция соединения JOIN. Общий синтаксис, стандартный для всех реляционных СУБД выглядит так:

```
FROM первая_таблица ТИП_СОЕДИНЕНИЯ вторая_таблица [ON (условия_соединения)]
```

Порядок следования таблиц при соединении играет важную роль, от этого зависит результат при использовании некоторых типов соединений. Иногда таблицы называют левая и правая таблицы.

Тип соединения определяет принцип соединения таблиц. *Условия соединения* определяют условия сравнения значений в столбцах соединяемых таблиц, равенство или не равенство.

Рассмотрим все типы соединения.

Соединение INNER JOIN.

Рассмотрим запрос следующего вида:

```
SELECT p.Name, s.Name
FROM [Production].[Product] p INNER JOIN [Production].[ProductSubcategory] s
ON p.ProductSubcategoryID=s.ProductSubcategoryID
```

Пользователь получит список названий товаров и названий категорий, к которым относиться товар, при этом не будут рассмотрены товары, для которых подкатегория не определена. INNER JOIN не учитывает NULL значения как в левой, так и в правой таблице при объединении. Синтаксически допустимо, и традиционно используется, сокращение названия соединения INNER JOIN до просто JOIN.

В рассмотренном выше запросе объединяются таблицы Product и ProductSubcategory, каждая из которых имеет столбец, атрибут, с именем Name. Что бы различать эти столбцы при объединении, были введены псевдонимы таблиц, и к столбцу Name обращение идет через оператор доступ – точка.

Соединение LEFT JOIN.

LEFT JOIN учитывает значения NULL из левой таблицы. Таким образом нижеприведенный запрос вернет названия продуктов и название подкатегорий, в том числе и те случаи, когда у продукта не определена подкатегория. Если подкатегория продукта не определена, то ее название будет NULL.

```
SELECT p.Name, s.Name
```

```
FROM [Production].[Product] p LEFT JOIN [Production].[ProductSubcategory] s
ON p.ProductSubcategoryID=s.ProductSubcategoryID
```

Соединение RIGHT JOIN учитывает значения NULL из правой таблицы.

Рассмотрим запрос:

```
SELECT p.Name, s.Name
FROM [Production].[Product] p RIGHT JOIN [Production].[ProductSubcategory] s
ON p.ProductSubcategoryID=s.ProductSubcategoryID
```

В данном случае соединение RIGHT JOIN учитывает ситуацию, что значения ключевого столбца правой таблицы, ProductSubcategory, могут принимать значение NULL. Это правило формально соблюдено СУБД, несмотря на то что в правой таблице, ProductSubcategory, столбец для соединения, ProductSubcategoryID, является первичным ключом.

Соединение FULL OUTER JOIN учитывает значение NULL и в левой, и в правой таблице. На практике синтаксическая единица FULL OUTER JOIN сокращается до FULL JOIN.

Рассмотрим пример. Даны две таблицы, tA и tB, со следующими значениями:

	id	tAvalue
1	1	1
2	2	2
3	3	3
4	4	NULL

	id	tBvalue
1	1	2
2	2	3
3	3	4
4	4	NULL

Запрос `SELECT [tAvalue], [tBvalue] FROM [dbo].[tA] FULL JOIN [dbo].[tB] ON tA.[tAvalue]=tB.[tBvalue]` вернет следующий результат:

	tAvalue	tBvalue
1	1	NULL
2	2	2
3	3	3
4	NULL	NULL
5	NULL	4
6	NULL	NULL

В данном случае был рассмотрен пример соединения, в котором условие соединения не связано с первичным или внешним ключом. Так же надо учитывать, что запрос вернул две строки со значением NULL в каждом из столбцов.

Соединение CROSS JOIN производит полное декартово произведение двух таблиц, и не имеет условия соединения. Рассмотрим запрос к таблицам из предыдущего примера

```
SELECT [tAvalue], [tBvalue] FROM

[dbo].[tA] CROSS JOIN [dbo].[tB]
```

Этот запрос вернет все возможные сочетания значений [tAvalue] и [tBvalue], т.е. шестнадцать строк. Соединение CROSS JOIN может потребовать значительных ресурсов СУБД, и серьезно затруднить работу других пользователей.

Соединение таблиц допускает соединение таблицы со своей копией. Стоит задача – найти такие продукты, для которых существуют продукты с таким же названием. Задачу можно решить с использованием выражения GROUP BY, а можно и с использованием самосоединения.

```
SELECT p1.Name

FROM [Production].[Product] p1 JOIN

[Production].[Product] p2

ON p1.Name=p2.Name and

p1.ProductID!=p2.ProductID
```

Стоит обратить внимание что в данном запросе обязательно использование псевдонимов. Для соединения использованы два условия, одно из которых на неравенство.

Соединение таблиц допускает использование ключевых слов WHERE, GROUP BY, HAVING, ORDER BY и других. Допускается соединение более чем двух таблиц.

Рассмотрим несколько примеров. Используется схема данных из первой лабораторной работы, а так же схема приведенная в приложении к данной работе.

1 Найти название продуктов и название подкатегорий этих продуктов, у которых отпускная цена больше 100, не включая случаи, когда продукт не относится ни к какой подкатегории

```
SELECT P.Name, PSC.Name

FROM [Production].[Product] AS P INNER JOIN

[Production].[ProductSubcategory] AS PSC

ON P.ProductSubcategoryID=PSC.ProductSubcategoryID

WHERE [ListPrice]>100
```

2 Найти название продуктов и название подкатегорий этих продуктов, у которых отпускная цена больше 100, включая случаи, когда продукт не относится ни к какой категории

```
SELECT P.Name, PSC.Name

FROM [Production].[Product] AS P LEFT JOIN

[Production].[ProductSubcategory] AS PSC

ON P.ProductSubcategoryID=PSC.ProductSubcategoryID

WHERE [ListPrice]>100
```

3 Найти название продуктов и название категорий из таблицы ProductCategory с которой связан этот продукт, не включая случаи, когда у продукта нет подкатегории

```
SELECT P.Name, PC.Name
```

```
FROM [Production].[Product] AS P INNER JOIN

[Production].[ProductSubcategory] AS PSC

ON P.ProductSubcategoryID=PSC.ProductSubcategoryID

INNER JOIN [Production].[ProductCategory] AS PC

ON PSC.ProductCategoryID=PC.ProductCategoryID
```

4 Найти название продукта, отпускную цену продукта, а также последнюю отпускную цену этого продукта (LastReceiptCost) которую можно узнать из таблицы ProductVendor

```
SELECT P.Name, P.ListPrice, PV.LastReceiptCost

FROM [Production].[Product] AS P INNER JOIN

[Purchasing].[ProductVendor] AS PV

ON P.ProductID=PV.ProductID
```

5 Найти название продукта, отпускную цену продукта, а также последнюю отпускную цену этого продукта (LastReceiptCost) которую можно узнать из таблицы ProductVendor, для таких продуктов, у которых отпускная цена оказалась ниже последней отпускной цены у поставщика, исключив те товары, для которых отпускная цена равна нулю

```
SELECT P.Name, P.ListPrice, PV.LastReceiptCost

FROM [Production].[Product] AS P INNER JOIN

[Purchasing].[ProductVendor] AS PV

ON P.ProductID=PV.ProductID

where P.ListPrice!=0 and P.ListPrice<PV.LastReceiptCost
```

6 Найти количество товаров, которые поставляют поставщики с самым низким кредитным рейтингом (CreditRating принимает целые значение от минимального, равного 1, до максимального, равного 5)

```
SELECT COUNT(DISTINCT PV.ProductID)

FROM [Purchasing].[ProductVendor] AS PV INNER JOIN

[Purchasing].[Vendor] AS V

ON PV.BusinessEntityID=V.BusinessEntityID

WHERE [CreditRating]=1
```

7 Найти сколько товаров приходится на каждый кредитный рейтинг, т.е. сформировать таблицу, первая колонка которой будет содержать номер кредитного рейтинга, вторая количество товаров, поставляемых всеми поставщиками, имеющими соответствующий кредитный рейтинг. Необходимо сформировать универсальный запрос, который будет и в случае появления новых значений кредитного рейтинга

```
SELECT [CreditRating], COUNT(DISTINCT PV.ProductID)

FROM [Purchasing].[ProductVendor] AS PV INNER JOIN

[Purchasing].[Vendor] AS V

ON PV.BusinessEntityID=V.BusinessEntityID
```

```
GROUP BY [CreditRating]
```

8 Найти номера первых трех подкатегорий (ProductSubcategoryID) с наибольшим количеством наименований товаров

```
SELECT TOP 3 [ProductSubcategoryID]
FROM [Production].[Product]
WHERE [ProductSubcategoryID] IS NOT NULL
GROUP BY [ProductSubcategoryID]
ORDER BY COUNT(*) DESC
```

9 Получить названия первых трех подкатегорий с наибольшим количеством наименований товаров

```
SELECT TOP 3 PC.ProductCategoryID
FROM [Production].[Product] AS P INNER JOIN
[Production].[ProductSubcategory] AS PSC
ON P.ProductSubcategoryID=PSC.ProductSubcategoryID
INNER JOIN [Production].[ProductCategory] AS PC
ON PSC.ProductCategoryID=PC.ProductCategoryID
GROUP BY PC.ProductCategoryID
ORDER BY COUNT(*) DESC
select top 3 psc.name, count(*)
from [Production].[Product] as p inner join
[Production].[ProductSubcategory] as psc
on p.ProductSubcategoryID=psc.ProductSubcategoryID
where p.ProductSubcategoryID is not null
group by p.ProductSubcategoryID, psc.Name
order by count(*) desc
```

10 Высчитать среднее количество товаров на приходящихся на одну подкатегорию, с точностью минимум до одной десятой.

```
SELECT 1.0*COUNT(*)/COUNT(DISTINCT [ProductSubcategoryID])
FROM [Production].[Product]
WHERE [ProductSubcategoryID] IS NOT NULL

SELECT CAST(COUNT(*) AS FLOAT)/COUNT(DISTINCT [ProductSubcategoryID])
FROM [Production].[Product]
WHERE [ProductSubcategoryID] IS NOT NULL
```

```
select cast((cast(count(p.ProductID) as float)/count(distinct p.ProductSubcategoryID))
as decimal(6,3))
from [Production].[Product] as p
where p.ProductSubcategoryID is not null
```

11 Вычислить среднее количество товаров, приходящихся на одну категорию в целых числах.

```
SELECT COUNT(*)/COUNT(DISTINCT PC.ProductCategoryID)
FROM [Production].[Product] AS P INNER JOIN
[Production].[ProductSubcategory] AS PSC
ON P.ProductSubcategoryID=PSC.ProductSubcategoryID
RIGHT JOIN [Production].[ProductCategory] AS PC
ON PSC.ProductCategoryID=PC.ProductCategoryID
```

12 Найти количество цветов товаров, приходящихся на каждую категорию, без учета товаров для которых цвет не определен

```
SELECT COUNT(DISTINCT [Color])
FROM [Production].[Product] AS P INNER JOIN
[Production].[ProductSubcategory] AS PSC
ON P.ProductSubcategoryID=PSC.ProductSubcategoryID
RIGHT JOIN [Production].[ProductCategory] AS PC
ON PSC.ProductCategoryID=PC.ProductCategoryID
GROUP BY PC.ProductCategoryID
```

13 Найти средний вес продуктов. Просмотреть таблицу продуктов, и убедиться, что есть продукты, для которых вес не определен. Модифицировать запрос так, чтобы при нахождении среднего веса продуктов, продукты для которых вес не определен считались как продукты с весом 10.

```
SELECT AVG(ISNULL([Weight],10))
FROM [Production].[Product]
```

14 Вывести названия продуктов, и период их активных продаж, период между SellStartDate и SellEndDate, в днях, отсортировав уменьшению времени продаж. Если продажи идут до сих пор, SellEndDate – не определен, то считать периодом продаж число дней с начала продаж, и по текущие сутки.

```
SELECT [Name], [SellStartDate], [SellEndDate],
DATEDIFF(D,[SellStartDate],ISNULL([SellEndDate],GETDATE()))
FROM [Production].[Product]
WHERE [SellStartDate] IS NOT NULL
```

15 Разбить продукты по количеству символов в названии, и для каждой группы определить количество продуктов

```
SELECT LEN([Name]), COUNT(*)
```

```
FROM [Production].[Product]

GROUP BY LEN([Name])
```

16 Найти для каждого поставщика количество подкатегорий продуктов, к которым относится продукты поставляемые им, без учета ситуации когда продукт не относится ни к какой подкатегории.

```
SELECT PV.BusinessEntityID, COUNT(DISTINCT P.ProductSubcategoryID)

FROM [Production].[Product] AS P INNER JOIN

[Purchasing].[ProductVendor] AS PV

ON P.ProductID=PV.ProductID

WHERE P.ProductSubcategoryID IS NOT NULL

GROUP BY PV.BusinessEntityID
```

17 Проверить есть ли продукты с одинаковым названием, если есть, то вывести эти названия.

```
SELECT P1.Name

FROM [Production].[Product] AS P1,

[Production].[Product] AS P2

WHERE P1.ProductID!=P2.ProductID AND

P1.Name=P2.Name


SELECT [Name]

FROM [Production].[Product]

GROUP BY [ProductID], [Name]

HAVING COUNT(*)>1
```

18 Найти первые 10 самых дорогих товаров, с учетом ситуации, когда цена цены у некоторых товаров могут совпадать.

```
SELECT TOP 10 WITH TIES [Name]

FROM [Production].[Product]

ORDER BY [ListPrice] DESC
```

19 Найти первые 10 процентов самых дорогих товаров, с учетом ситуации, когда цена цены у некоторых товаров могут совпадать.

```
SELECT TOP 10 PERCENT WITH TIES [Name]

FROM [Production].[Product]

ORDER BY [ListPrice] DESC
```

20 Найти первых трех поставщиков, отсортированных по количеству поставляемых товаров, с учетом ситуации что количество поставляемых товаров может совпадать для разных поставщиков.

```
SELECT TOP 3 WITH TIES PV.BusinessEntityID

FROM [Production].[Product] AS P INNER JOIN
```

```
[Purchasing].[ProductVendor] AS PV

ON P.ProductID=PV.ProductID

GROUP BY PV.BusinessEntityID

ORDER BY COUNT(P.ProductID) DESC
```

Задания для самостоятельной работы.

1 Найти и вывести на экран название продуктов и название категорий товаров, к которым относиться этот продукт, с учетом того, что в выборку попадут только товары с цветом Red, и ценой не менее 100

2 Вывести на экран название подкатегорий, таких что есть подкатегории с таким же названием

3 Вывести на экран название категорий, и количество товаров в данной категории.

4 Вывести на экран название подкатегории, а также количество товаров в данной подкатегории, с учетом ситуации, что могут существовать подкатегории с одинаковыми именами

5 Вывести на экран название первых трех подкатегорий с небольшим количеством товаров.

6 Вывести на экран название подкатегории, и максимальную цену продукта, с цветом Red, в этой подкатегории

7 Вывести на экран название поставщика, и количество товаров, которые он поставляет

8 Вывести на экран название товаров, таких что они поставляются более чем одним поставщиком

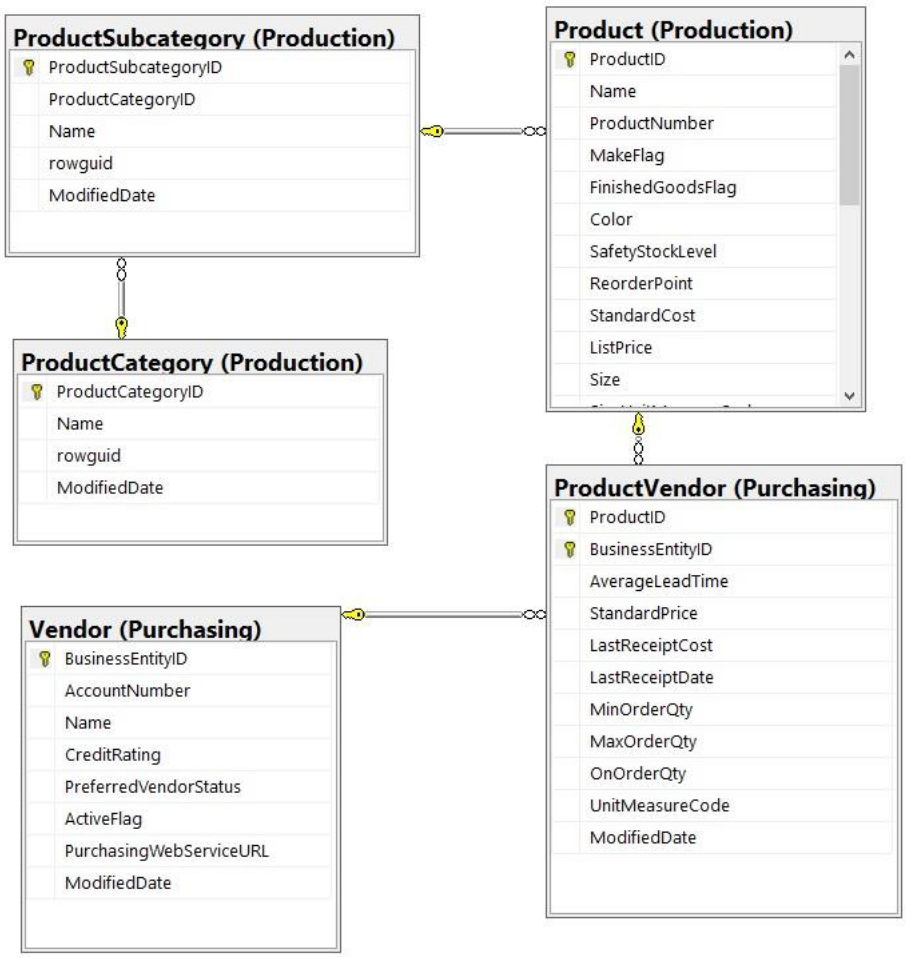
9 Вывести на экран название самого продаваемого товара

10 Вывести на экран название категории, товары из которых продаются наиболее активно

11 Вывести на экран название категорий, количество подкатегорий и количество товаров в них

12 Вывести на экран номер кредитного рейтинга, и количество товаров, поставляемых компаниями имеющих этот кредитный рейтинг

Приложение


ProductSubcategory (Production)
ProductSubcategoryID
ProductCategoryID
Name
rowguid
ModifiedDate
Product (Production)
ProductID
Name
ProductNumber
MakeFlag
FinishedGoodsFlag
Color
SafetyStockLevel
ReorderPoint
StandardCost
ListPrice
Size
ProductCategory (Production)
ProductCategoryID
Name
rowguid
ModifiedDate
ProductVendor (Purchasing)
ProductID
BusinessEntityID
AverageLeadTime
StandardPrice
LastReceiptCost
LastReceiptDate
MinOrderQty
MaxOrderQty
OnOrderQty
UnitMeasureCode
ModifiedDate
Vendor (Purchasing)
BusinessEntityID
AccountNumber
Name
CreditRating
PreferredVendorStatus
ActiveFlag
PurchasingWebServiceURL
ModifiedDate