**Лабораторная работа № 5.** [**Использование подзапросов**](#Лаб7)

**КУРНОСЕНКО СОФЬЯ**

**Подзапрос** – это SELECT-запрос, который выполняется в рамках другого запроса. Подзапросы могут применяться в секции WHERE. Подзапросы бывают двух видов: коррелируемые и независимые.

*Коррелируемый* подзапрос зависит от внешнего запроса и выполняется для каждой строки результирующего набора.

*Независимый* подзапрос не зависит от внешнего запроса и выполняется только один раз, но результат его выполнения подставляется в каждую строку результирующего набора.

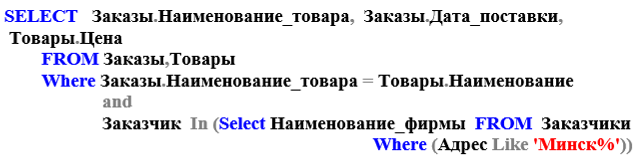
**1.** На основе таблиц **FACULTY**, **PULPIT** и **PROFESSION** сформировать список наименований кафедр (столбец **PULPIT\_NAME**), которые находятся на факультете (таблица **FACULTY**), обеспечивающем подготовку по специальности, в наименовании (столбец **PROFESSION\_ NAME**) которого содержится слово ***технология*** или ***технологии***. Примечание: использовать в секции WHERE предикат IN c некоррелированным подзапросом к таблице **PROFESSION**.

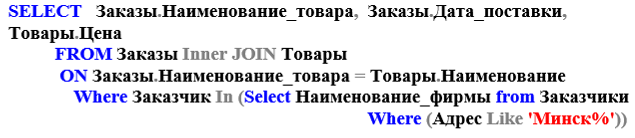
***→ Теория (методичка)***

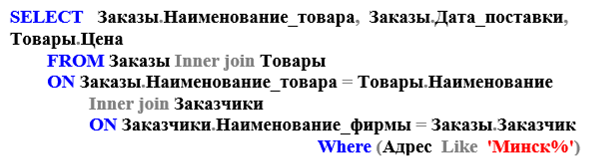
Операция IN формирует логическое значение «истина» в том случае, если значение, указанное слева от ключевого слова IN, равно хотя бы одному из значений списка, указанного справа.

Пусть требуется определить список, даты поставки и исходные цены тех товаров, в адресе доставки которых есть слово Минск.

Запросы ниже дают один и тот же результат:







***→ Выполнение заданий***

-- на основе таблиц PULPIT и PROFESSION

select PULPIT\_NAME

from PULPIT

where PULPIT.FACULTY in (select FACULTY from PROFESSION

where (PROFESSION\_NAME like '%технология%'

or

PROFESSION\_NAME like'%технологии%'));

-- на основе таблиц PULPIT, FACULTY и PROFESSION

select PULPIT\_NAME

from PULPIT, FACULTY

where FACULTY.FACULTY = PULPIT.FACULTY

and

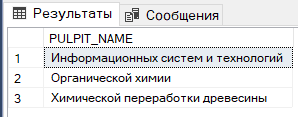
PULPIT.FACULTY in (select FACULTY from PROFESSION

where (PROFESSION\_NAME like '%технология%'

or

PROFESSION\_NAME like'%технологии%'));

Для обоих запросов результат одинаковый:

******

**2.** Переписать запрос пункта 1 таким образом, чтобы тот же подзапрос был записан в конструкции INNER JOIN секции FROM внешнего запроса. При этом результат выполнения запроса должен быть аналогичным результату исходного запроса.

***→ Выполнение заданий***

select PULPIT\_NAME

from PULPIT join FACULTY

on FACULTY.FACULTY = PULPIT.FACULTY

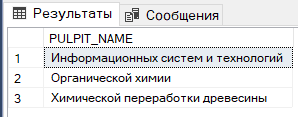
and

PULPIT.FACULTY in (select FACULTY from PROFESSION

where (PROFESSION\_NAME like '%технология%'

or

PROFESSION\_NAME like'%технологии%'));

******

**3.** Переписать запрос, реализующий 1 пункт без использования подзапроса. Примечание: использовать соединение INNER JOIN трех таблиц.

select distinct PULPIT\_NAME

from PULPIT join FACULTY

on FACULTY.FACULTY = PULPIT.FACULTY

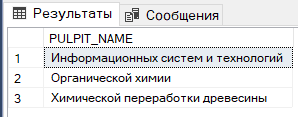
join PROFESSION

on PROFESSION.FACULTY = PULPIT.FACULTY

and (PROFESSION\_NAME like '%технология%'

or

PROFESSION\_NAME like'%технологии%');

******

*distinct* нужен потому что на одном факультете могут быть разные кафедры с одинаковыми специальностями.

select PULPIT\_NAME, FACULTY\_NAME, PROFESSION\_NAME

from PULPIT join FACULTY

on FACULTY.FACULTY = PULPIT.FACULTY

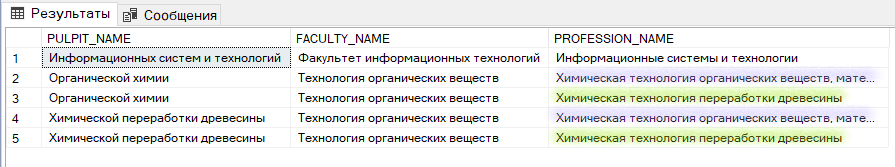
join PROFESSION

on PROFESSION.FACULTY = PULPIT.FACULTY

and (PROFESSION\_NAME like '%технология%'

or

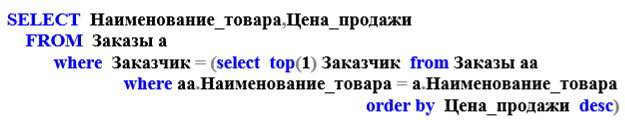
PROFESSION\_NAME like'%технологии%');



**4.** На основе таблицы **AUDITORIUM** сформировать список аудиторий самых больших вместимостей (столбец **AUDITORIUM\_CAPACITY**) для каждого типа аудитории (**AUDITORIUM\_TYPE**). При этом результат следует отсортировать в порядке убывания вместимости. Примечание: использовать коррелируемый подзапрос c секциями TOP и ORDER BY.

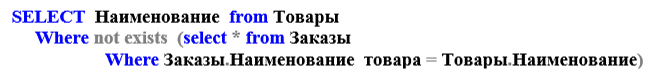
***→ Теория (методичка)***

На основании таблицы **Заказы** сформировать перечень товаров, для которых их цены продажи являются максимальными. Здесь одной и той же таблице присваиваются разные псевдонимы:



Операция EXISTS формирует значение «истина», если результирующий набор подзапроса содержит хотя бы одну строку, в противоположном случае − значение «ложь».

На основании таблиц **Товары** и **Заказы** сформировать перечень товаров из таблицы **Товары**, которые не заказаны покупателями:



***→ Выполнение заданий***

select \*

from AUDITORIUM as a

where a.AUDITORIUM = (

select top(1) b.AUDITORIUM

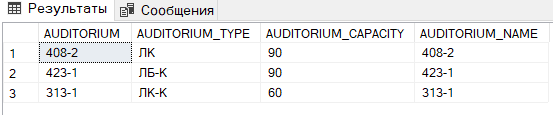
from AUDITORIUM as b

where b.AUDITORIUM\_TYPE = a.AUDITORIUM\_TYPE

order by b.AUDITORIUM\_CAPACITY desc

)

order by a.AUDITORIUM\_CAPACITY desc;

******

В основном запросе мы проходимся по каждой аудитории, в подзапросе мы формируем список аудиторий, таких, чтобы их тип был таким же, как у аудитории, на которой мы остановились в основном запросе. Сформированный список сортируем по убыванию по значению вместимости и выбираем первую аудиторию сформированного списка, то есть аудиторию с наиболбшей вместимостью. Далее в основном запросе мы смотрим, чтобы найденная аудитория из основного запроса соответствовала аудитории из подзапроса. Если это так, то добавляем аудиторию в итоговую выборку, иначе пропускаем.

**5.** На основе таблиц **FACULTY** и **PULPIT** сформировать список наименований факультетов (столбец **FACULTY\_NAME**) на котором нет ни одной кафедры (таблица **PULPIT**). Примечание: использовать предикат EXISTS и коррелированный подзапрос.

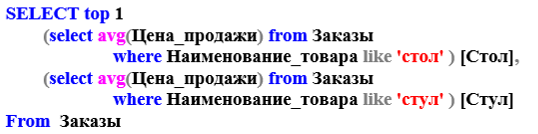
***→ Выполнение заданий***

***→ Теория (моя)***

**6.** На основе таблицы **PROGRESS** сформировать строку, содержащую средние значения оценок (столбец **NOTE**) по дисциплинам, имеющим следующие коды: **ОАиП**, **БД** и **СУБД**. Примечание: использовать три некоррелированных подзапроса в списке SELECT; в подзапросах применить агрегатные функции AVG.

***→ Теория (методичка)***

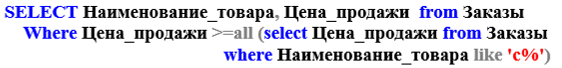
Определить среднее значение цены продажи столов и стульев из таблицы **Заказы**:



Операция ***>=ALL*** формирует истинное значение в том случае, если значение стоящее слева больше или равно каждому значению в списке, указанном справа.

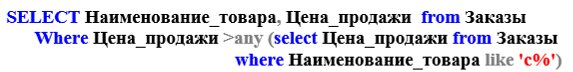
Пусть надо определить наименования товаров, цены продажи которых превышают или равны значениям цен продажи товаров, наименования которых начинаются на букву ‘**с**’.

Запрос может иметь вид:



Операция ***>=ANY*** формирует истинное значение в том случае, если значение стоящее слева, больше или равно хотя бы одному значению в списке, указанном справа.

Определить наименования товаров, цены продажи которых превышают хотя бы одно значение цены продажи товаров, наименования которых начинаются на букву ‘**с**’:



***→ Выполнение заданий***

select top(1)

isnull((select avg(NOTE) from PROGRESS

where SUBJECT like 'ОАиП'), '0') as [Средний балл по ОАиП],

isnull((select avg(NOTE) from PROGRESS

where SUBJECT like 'БД'), '0') as [Средний балл по БД],

isnull((select avg(NOTE) from PROGRESS

where SUBJECT like 'СУБД'), '0') as [Средний балл по СУБД]

from PROGRESS;

******

**7.** Разработать SELECT-запрос, демонстрирующий принцип применения ALL совместно с подзапросом.

***→ Выполнение заданий***

-- 7 найдем студентов со средним баллом выше среднего по всем предметам

select STUDENT.NAME,

STUDENT.IDSTUDENT,

avg(PROGRESS.NOTE) AS [Средний балл по всем предметам]

from STUDENT inner join PROGRESS

on STUDENT.IDSTUDENT = PROGRESS.IDSTUDENT

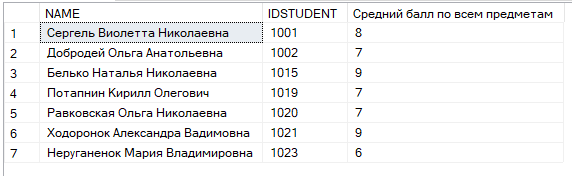
where PROGRESS.NOTE >=ALL (

select avg(PROGRESS.NOTE)

from PROGRESS

)

group by STUDENT.NAME, STUDENT.IDSTUDENT;

******

***→ Выполнение заданий***

-- 8 выбрать преподавателей, которые преподают кафедрах, где ведутся предметы со словом "программирование" в названии

select \*

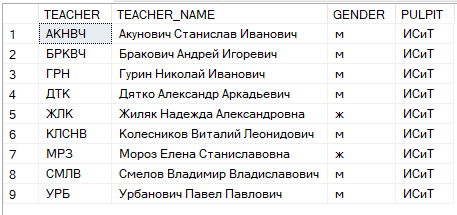
from TEACHER

where TEACHER.PULPIT =any (

select SUBJECT.PULPIT

from SUBJECT

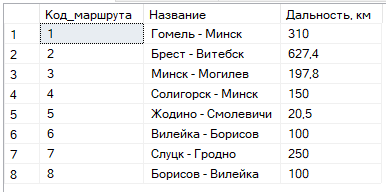
where SUBJECT\_NAME like '%программирование%');



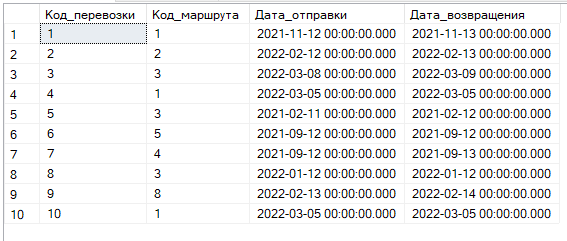
Находим в подзапросе кафедры, где преподаются предметы со словом "программирование" в названии. В основном запросе смотрим, чтобы кафедра текущего преподавателя совпадала хотя бы с одной кафедрой из списка подзапроса.

9. Разработать и выполнить аналогичные запросы для базы данных **X\_MyBASE**.

Маршруты



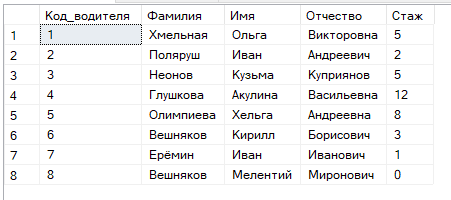
Перевозки



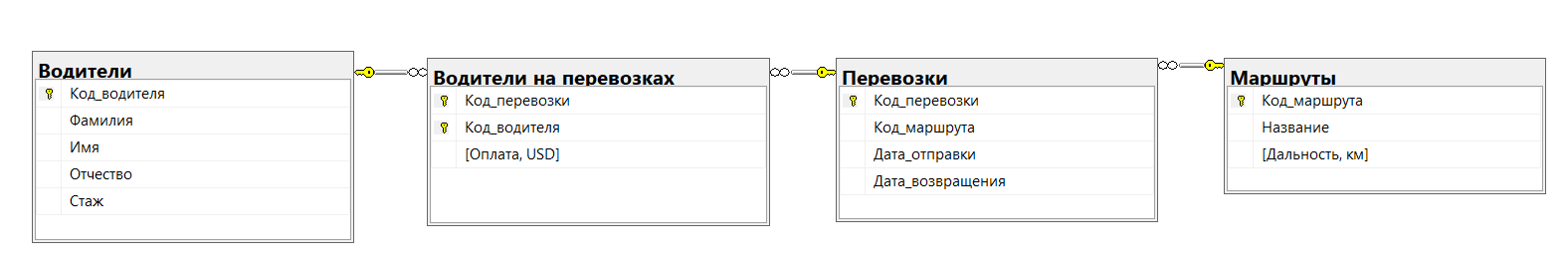
Водители на перевозках



Водители



Диаграмма



-- 1 найти перевозки, которые идут по маршруту, в названии которых содержится слово "Минск" или "Гомель" и вывести коды водителей на этих перевозках

-- На основе таблиц "Маршруты", "Перевозки", "Водители на перевозках"

-- Примечание: использовать в секции WHERE предикат IN c некоррелированным подзапросом к таблице Маршруты

select distinct Код\_водителя

from [Водители на перевозках], Перевозки

where [Водители на перевозках].Код\_перевозки = Перевозки.Код\_перевозки

and

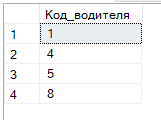
Перевозки.Код\_маршрута in

(select Код\_маршрута

from Маршруты

where Название like '%минск%'

or Название like '%гомель%');



-- 2 Переписать запрос пункта 1 таким образом, чтобы тот же подзапрос был записан в конструкции INNER JOIN секции FROM внешнего запроса. При этом результат выполнения запроса должен быть аналогичным результату исходного запроса.

select distinct Код\_водителя

from [Водители на перевозках] inner join Перевозки

on [Водители на перевозках].Код\_перевозки = Перевозки.Код\_перевозки

and

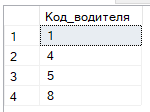
Перевозки.Код\_маршрута in

(select Код\_маршрута

from Маршруты

where Название like '%минск%'

or Название like '%гомель%');



-- 3 Переписать запрос, реализующий 1 пункт без использования подзапроса. Примечание: использовать соединение INNER JOIN трех таблиц.

select distinct Код\_водителя

from [Водители на перевозках] inner join Перевозки

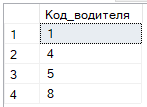
on [Водители на перевозках].Код\_перевозки = Перевозки.Код\_перевозки

inner join Маршруты

on Перевозки.Код\_маршрута = Маршруты.Код\_маршрута

and (Название like '%минск%'

or Название like '%гомель%');



-- 4 Для каждого водителя найти код перевозки, за которую он(а) получил(а) наибольшую оплату

-- При этом результат следует отсортировать в порядке убывания вместимости. Примечание: использовать коррелируемый подзапрос c секциями TOP и ORDER BY.

select \*

from [Водители на перевозках] as a

where a.Код\_перевозки = (

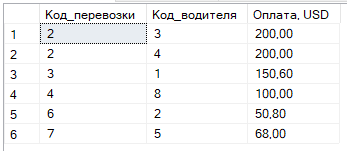
select top(1) Код\_перевозки

from [Водители на перевозках] as b

where a.Код\_водителя = b.Код\_водителя

order by b.[Оплата, USD] desc

);



-- 5 сформировать список водителей, ни разу не бывавших на перевозках

-- Примечание: использовать предикат EXISTS и коррелированный подзапрос.

select \*

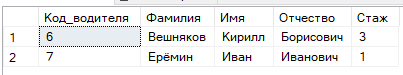
from Водители

where not exists(

select \*

from [Водители на перевозках]

where [Водители на перевозках].Код\_водителя = Водители.Код\_водителя)



-- 6 на основе таблиц [Водители на перевозках] и Перевозки сформировать строку, содержащую средние значения оплаты за маршруты с кодом 1, 2 и 3

-- Примечание: использовать три некоррелированных подзапроса в списке SELECT; в подзапросах применить агрегатные функции AVG.

select top(1)

(select avg([Оплата, USD])

from [Водители на перевозках], Перевозки

where [Водители на перевозках].Код\_перевозки = Перевозки.Код\_перевозки

and

Перевозки.Код\_маршрута = 1) as [Маршрут 1],

(select avg([Оплата, USD])

from [Водители на перевозках], Перевозки

where [Водители на перевозках].Код\_перевозки = Перевозки.Код\_перевозки

and

Перевозки.Код\_маршрута = 2) as [Маршрут 2],

(select avg([Оплата, USD])

from [Водители на перевозках], Перевозки

where [Водители на перевозках].Код\_перевозки = Перевозки.Код\_перевозки

and

Перевозки.Код\_маршрута = 3) as [Маршрут 3];



-- 7 выбрать маршрут(ы) с наибольшей дальностью

-- Разработать SELECT-запрос, демонстрирующий принцип применения ALL совместно с подзапросом.

select \*

from Маршруты

where [Дальность, км] >=all(

select [Дальность, км]

from Маршруты

);



-- 8 выбрать маршруты, дальность которых превышает хотя бы один маршрут, содержащем "Минск" в своем названии

-- Разработать SELECT-запрос, демонстрирующий принцип применения ANY совместно с подзапросом.

select \*

from Маршруты

where [Дальность, км] >any(

select [Дальность, км]

from Маршруты

where Название like '%Минск%'

);

