

Министерство образования и науки Российской Федерации

Калужский филиал  
федерального государственного бюджетного образовательного  
учреждения высшего образования  
**«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»**  
(КФ МГТУ им. Н.Э. Баумана)

**С.А. Глебов**

**СОЗДАНИЕ ЗАПРОСОВ НА ВЫБОРКУ**  
Методические указания по выполнению лабораторной работы  
по курсу «Базы данных»

Калуга – 2018

УДК 004.65  
ББК 32.972.134  
Г53

Методические указания составлены в соответствии с учебным планом КФ МГТУ им. Н.Э. Баумана по направлению подготовки 09.03.04 «Программная инженерия» кафедры «Программного обеспечения ЭВМ, информационных технологий и прикладной математики».

Методические указания рассмотрены и одобрены:

- Кафедрой «Программного обеспечения ЭВМ, информационных технологий и прикладной математики» (ФН1-КФ) протокол № 7 от «21» февраля 2018 г.


И.о. зав. кафедрой ФН1-КФ  к.т.н., доцент Ю.Е. Гагарин

- Методической комиссией факультета ФНК протокол № 2 от «18» 02 2018 г.

Председатель методической комиссии факультета ФНК  к.х.н., доцент К.Л. Анфилов

- Методической комиссией КФ МГТУ им.Н.Э. Баумана протокол № 2 от «06» 03 2018 г.

Председатель методической комиссии КФ МГТУ им.Н.Э. Баумана

 д.э.н., профессор О.Л. Перерва

Рецензент:  
к.т.н., доцент кафедры ЭИУ6-КФ

 А.Б. Лачихина

Авторы  
к.ф.-м.н., доцент кафедры ФН1-КФ

 С.А. Глебов

#### Аннотация

Методические указания по выполнению лабораторной работы по курсу «Базы данных» содержит руководство по получению практических навыков использования оператора select с учетом выбранной СУБД и задание на выполнение лабораторной работы.

Предназначены для студентов 3-го курса бакалавриата КФ МГТУ им. Н.Э. Баумана, обучающихся по направлению подготовки 09.03.04 «Программная инженерия».

## ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ .....	4
ЦЕЛЬ И ЗАДАЧИ РАБОТЫ, ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ЕЕ ВЫПОЛНЕНИЯ.....	5
КРАТКАЯ ХАРАКТЕРИСТИКА ОБЪЕКТА ИЗУЧЕНИЯ, ИССЛЕДОВАНИЯ .....	6
УНАРНЫЕ ОПЕРАЦИИ НА ЯЗЫКЕ СТРУКТУРИРОВАННЫХ ЗАПРОСОВ .....	8
БИНАРНЫЕ ОПЕРАЦИИ НА ЯЗЫКЕ СТРУКТУРИРОВАННЫХ ЗАПРОСОВ .....	12
ИСПОЛЬЗОВАНИЕ ПОДЗАПРОСОВ.....	21
ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ .....	25
ВАРИАНТЫ ЗАДАНИЙ.....	<b>Ошибка! Закладка не определена.</b>
КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ .....	25
ФОРМА ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ .....	25
ОСНОВНАЯ ЛИТЕРАТУРА .....	26
ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА .....	26

## **ВВЕДЕНИЕ**

Настоящие методические указания составлены в соответствии с программой проведения лабораторных работ по курсу «Базы данных» на кафедре «Программное обеспечение ЭВМ, информационные технологии и прикладная математика» факультета фундаментальных наук Калужского филиала МГТУ им. Н.Э. Баумана.

Методические указания, ориентированные на студентов 3-го курса направления подготовки 09.03.04 «Программная инженерия», содержат руководство по получению практических навыков использования оператора select с учетом выбранной СУБД и задание на выполнение лабораторной работы.

## **ЦЕЛЬ И ЗАДАЧИ РАБОТЫ, ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ЕЕ ВЫПОЛНЕНИЯ**

Целью выполнения лабораторной работы является: получение практических навыков использования оператора `select`.

Основной задачей выполнения лабораторной работы является: на основе имеющейся базы данных (н-р, по полученной в предыдущей работе) разработать `select`-запросы с использованием унарных, бинарных операций, а также содержащие подзапросы.

Результатами работы являются:

- Разработанные `select`-запросы к базе данных.
- Подготовленный отчет.

## КРАТКАЯ ХАРАКТЕРИСТИКА ОБЪЕКТА ИЗУЧЕНИЯ, ИССЛЕДОВАНИЯ

Центральное место в языке структурированных запросов SQL занимает оператор Select, с помощью которого реализуется самая востребованная операция при работе с базами данных – запросы.

Оператор Select осуществляет вычисление выражений как реляционной, так и псевдореляционной алгебры. В данном курсе мы рассмотрим реализацию только уже пройденных нами унарных и бинарных операций реляционной алгебры, а также осуществление запросов, с помощью так называемых подзапросов.

Кстати, необходимо заметить, что в случае работы с операциями реляционной алгебры в результирующих отношениях могут появляться дублирующие кортежи. В правилах языка структурированных запросов нет строгого запрещения на присутствие повторяющихся строк в отношениях (в отличие от обычной реляционной алгебры), поэтому исключать дубликаты из результата не обязательно.

Итак, рассмотрим базовую структуру оператора Select. Она достаточно проста и включает в себя следующие стандартные обязательные фразы:

**Select ...**

**From ...**

**Where ... ;**

На месте многоточия в каждой строчке должны стоять отношения, атрибуты и условия конкретной базы данных и задания к ней. В самом общем случае базовая структура Select должна выглядеть следующим образом:

**Select** *выбрать такие-то атрибуты*

**From** *из таких-то отношений*

**Where** *с такими-то условиями выборки кортежей*

Таким образом, выбираем мы атрибуты из схемы отношений (заголовки некоторых столбцов), при этом указывая, из каких отношений (а их, как видно, может быть несколько) мы производим

нашу выборку и, наконец, на основании каких условий мы останавливаем свой выбор на тех или иных кортежах.

Важно заметить, что ссылки на атрибуты происходят с помощью их имен.

Таким образом, получается следующий **алгоритм работы** этого базового оператора Select:

- 1) запоминаются условия выборки кортежей из отношения;
- 2) проверяется, какие кортежи удовлетворяют указанным свойствам. Такие кортежи запоминаются;
- 3) на выход выводятся перечисленные в первой строчке базовой структуры оператора Select атрибуты со своими значениями. (Если говорить о табличной форме записи отношения, то выведутся те столбцы таблицы, заголовки которых были перечислены как необходимые атрибуты; разумеется, столбцы выведутся не полностью, в каждом из них останутся только те кортежи, которые удовлетворили названным условиям.)

Пусть нам дано следующее отношение  $r_1$ , как фрагмент некой базы данных книжного магазина:

Код книги	Название книги	Автор книги	Цена книги
1258963	Холодные берега	С. Лукьяненко	186,9
1236954	Мобильник	С. Кинг	201,4

Пусть также нам дано следующее выражение с оператором Select:

**Select** *Название книги, Автор книги*

**From**  $r_1$

**Where** *Цена книги* > 200;

Результатом этого оператора будет следующий фрагмент кортежа:  
(Мобильник, С. Кинг).

(В дальнейшем мы подвергнем рассмотрению множество примеров реализации запросов с использованием этой базовой структуры и ее применение изучим очень подробно.)

## УНАРНЫЕ ОПЕРАЦИИ НА ЯЗЫКЕ СТРУКТУРИРОВАННЫХ ЗАПРОСОВ

В этом параграфе мы рассмотрим, как реализуются на языке структурированных запросов с помощью оператора Select уже знакомые нам унарные операции выборки, проекции и переименования.

Важно заметить, что если раньше мы могли работать только с отдельными операциями, то даже один оператор Select в общем случае позволяет определить целое выражение реляционной алгебры, а не какую-то одну операцию.

Итак, перейдем непосредственно к анализу представления унарных операций на языке структурированных запросов.

### **Операция выборки.**

Операция выборки на языке SQL реализуется оператором Select следующего вида:

**Select** *все атрибуты*

**From** *имя отношения*

**Where** *условие выборки;*

Здесь вместо того, чтобы писать «все атрибуты», можно использовать значок «\*». В теории языка структурированных запросов этот значок означает выбор всех атрибутов из схемы отношения.

Условие выборки здесь (и во всех остальных реализациях операций) записывается в виде логического выражения со стандартными связками not (не), and (и), or (или). На атрибуты отношения ссылаемся посредством их имен.

Рассмотрим пример. Определим следующую схему отношения:

Успеваемость (*№ зачетной книжки*, *Семестр*, *Код предмета*, *Оценка*, *Дата*);

Здесь, как уже упоминалось ранее, подчеркнутые атрибуты образуют ключ отношения.



Составим оператор Select следующего вида, реализующий унарную операцию выборки:

Select \*

From *Успеваемость*

Where № зачетной книжки = 100 and Семестр = 6;

Понятно, что в результат этого оператора машина выведет успеваемость студента с номером зачетки сто за шестой семестр.

### Операция проекции.

Операция проекции на языке структурированных запросов реализуется даже проще, чем операция выборки. Напомним, что при применении операции проекции выбираются не строки (как при применении операции выборки), а столбцы. Поэтому достаточно перечислить заголовки нужных столбцов (т. е. имена атрибутов), без указания каких-либо посторонних условий. Итого, получаем оператор следующего вида:

Select *список имен атрибутов*

From *имя отношения*;

После применения этого оператора машина выдаст те столбцы таблицы-отношения, имена которых были указаны в первой строчке этого оператора Select.

Как мы уже упоминали ранее, повторяющиеся строки и столбцы исключать из результирующего отношения не обязательно. Но если в заказе или в задании требуется обязательно элиминировать дубликаты, следует использовать специальную опцию языка структурированных запросов – **distinct**. Эта опция задает автоматическое исключение дубликатов кортежей из отношения. С применением этой опции оператор Select будет выглядеть следующим образом:

Select *distinct список имен атрибутов*

From *имя отношения*;

В языке SQL существует специальное обозначение для необязательных элементов выражений – квадратные скобки [...]. Поэтому в самом общем виде операция проекции будет выглядеть следующим образом:

Select [*distinct*] *список имен атрибутов*

**From** *имя отношения*;

Однако если результат применения операции гарантированно не содержит дубликатов или же дубликаты все-таки допустимы, то опцию **distinct** лучше не указывать, чтобы не загромождать запись, т. е. из соображений производительности работы оператора.

Рассмотрим пример, иллюстрирующий возможность стопроцентной уверенности в отсутствии дубликатов. Пусть дана уже известная нам схема отношений:

Успеваемость (*№ зачетной книжки, Семестр, Код предмета, Оценка, Дата*).

Пусть дан следующий оператор Select:

**Select** *№ зачетной книжки, Семестр, Код предмета*

**From** *Успеваемость*;

Здесь, как легко видеть, три возвращающихся оператором атрибута образуют ключ отношения. Именно поэтому опция **distinct** становится излишней, ведь дубликатов гарантированно не будет. Это следует из требования, накладываемого на ключи, называемого ограничением уникальности. Подробнее это свойство мы рассмотрим дальше, но если атрибут ключевой, то дубликатов в нем нет.

### **Операция переименования.**

Операция переименования атрибутов на языке структурированных запросов осуществляется довольно просто. А именно воплощается в действительность следующим алгоритмом:

- 1) в списке имен атрибутов фразы Select перечисляются те атрибуты, которые необходимо переименовать;
- 2) к каждому указанному атрибуту добавляется специальное ключевое слово as;
- 3) после каждого вхождения слова as указывается то имя соответствующего атрибута, на которое необходимо поменять имя исходное.

Таким образом, с учетом всего вышесказанного, оператор, соответствующий операции переименования атрибутов, будет выглядеть следующим образом:

**Select** *имя атрибута I as новое имя атрибута I, ...*

**From** *имя отношения;*

Покажем работу этого оператора на примере. Пусть дана уже знакомая нам схема отношения:

Успеваемость (№ зачетной книжки, Семестр, Код предмета, Оценка, Дата);

Пусть у нас имеется заказ поменять имена некоторых атрибутов, а именно вместо «№ зачетной книжки» должно стоять «№ зачетки» и вместо «Оценка» – «Балл».

Запишем, как будет выглядеть оператор Select, реализующий эту операцию переименования:

**Select** *зачетной книжки as № зачетки, Семестр, Код предмета, Оценка as Балл, Дата*

**From** *Успеваемость;*

Таким образом, результатом применения этого оператора будет новая схема отношения, отличающаяся от исходной схемы отношения «Успеваемость» именами двух атрибутов.

## БИНАРНЫЕ ОПЕРАЦИИ НА ЯЗЫКЕ СТРУКТУРИРОВАННЫХ ЗАПРОСОВ

Как и унарные операции, операции бинарные также имеют свою реализацию на языке структурированных запросов или SQL. Итак, рассмотрим осуществление на этом языке уже пройденных нами бинарных операций, а именно – [операций объединения](#), [пересечения](#), [разности](#), [декартового произведения](#), [естественного соединения](#), [внутреннего](#) и [левого](#), [правого](#), [полного внешнего соединения](#).

### Операция объединения.

Для того чтобы реализовать операцию объединения двух отношений приходится использовать одновременно два оператора Select, каждый из которых соответствует какому-то одному из исходных отношений-операндов. И к этим двум базовым операторам Select необходимо применить специальную операцию **Union**. Учитывая все вышесказанное, запишем, как же операция объединения будет выглядеть с использованием семантики языка структурированных запросов:

**Select** список имен атрибутов отношения 1

**From** имя отношения 1

**Union**

**Select** список имен атрибутов отношения 2

**From** имя отношения 2;

Важно заметить, что списки имен атрибутов двух объединяемых отношений должны ссылаться на атрибуты совместимых типов и быть перечислены в согласованном порядке. Если это требование не соблюдать, ваш запрос не сможет быть выполнен, и компьютер выдаст сообщение об ошибке.

Но, что интересно отметить, сами имена атрибутов в этих отношениях могут быть различными. В таком случае результирующему отношению приписываются имена атрибутов, указанные в первом операторе Select.

Также необходимо знать, что использование операции Union предполагает автоматическое исключение из результирующего

отношения всех дубликатов кортежей. Поэтому, если вам нужно, чтобы все повторяющиеся строки в конечном результате сохранились, вместо операции **Union** следует применять модификацию этой операции – операцию **Union All**. В таком случае операция объединения двух отношений будет выглядеть следующим образом:

**Select** список имен атрибутов отношения 1

**From** имя отношения 1

**Union All**

**Select** список имен атрибутов отношения 2

**From** имя отношения 2;

В этом случае из результирующего отношения дубликаты кортежей удаляться не будут.

Используя уже упоминавшееся ранее обозначение для необязательных элементов и опций в операторах **Select**, запишем самый общий вид операции объединения двух отношений на языке структурированных запросов:

**Select** список имен атрибутов отношения 1

**From** имя отношения 1

**Union [All]**

**Select** список имен атрибутов отношения 2

**From** имя отношения 2;

### **Операция пересечения.**

Операция пересечения и операция разности двух отношений на языке структурированных запросов реализуются похожим образом (мы рассматриваем наиболее простой способ представления, так как, чем проще метод, тем он экономичнее, актуальнее и, следовательно, наиболее востребован). Итак, мы подвергнем разбору способ реализации операции пересечения с использованием **ключей**.

Этот способ предполагает участие двух конструкций **Select**, но они не равноправны (как в представлении операции объединения), одна из них является как бы «подконструкцией», «подциклом». Такой оператор обычно называют **подзапросом**.

Итак, пусть у нас имеются две схемы отношений ( $R_1$  и  $R_2$ ), приблизительно определенные следующим образом:

$R_1$  (ключ, ...) и

$R_2$  (ключ, ...);

Воспользуемся также при записи этой операции специальной опцией **in**, что буквально означает «в» или (как в данном конкретном случае) «содержится в».

Итак, с учетом всего вышесказанного, операция пересечения двух отношений с помощью языка структурированных запросов запишется следующим образом:

**Select** \*

**From**  $R_1$

**Where** *ключ* **in**

(**Select** *ключ* **From**  $R_2$ );

Таким образом, мы видим, что подзапросом в данном случае будет являться оператор в круглых скобках. Этот подзапрос в нашем случае возвращает список значений ключа отношения  $R_2$ . И, как следует из нашей записи операторов, из анализа условия выборки, в результирующее отношение попадут только те кортежи отношения  $R_1$ , ключ которых содержится в списке ключей отношения  $R_2$ . То есть, в итоговом отношении, если вспомнить определение пересечения двух отношений, останутся лишь те кортежи, которые принадлежат обоим отношениям.

### Операция разности.

Как уже было сказано ранее, унарная операция разности двух отношений реализуется аналогично операции пересечения. Здесь также, кроме главного запроса с оператором **Select**, используется второй, вспомогательный запрос – так называемый подзапрос.

Но в отличие от воплощения в жизнь предыдущей операции, при реализации операции разности необходимо использовать другое ключевое слово, а именно **not in**, что в дословном переводе означает «не в» или (как уместно перевести в нашем рассматриваемом случае) – «не содержится в».

Итак, пусть, как и в предыдущем примере, у нас имеются две схемы отношений ( $R_1$  и  $R_2$ ), приблизительно заданные:

$R_1$  (ключ, ...) и

$R_2$  (ключ, ...);

Как видим, среди атрибутов этих отношений снова заданы ключевые атрибуты.

Таким образом, получаем следующий вид для представления в языке структурированных запросов операции разности:

**Select \***

**From**  $R_1$

**Where** ключ **not in**

(**Select** ключ **From**  $R_2$ );

Таким образом, в результирующее отношение выбираются только те кортежи отношения  $R_1$ , ключ которых не содержится в списке ключей отношения  $R_2$ . Если рассматривать запись буквально, то действительно получается, что из отношения  $R_1$  «вычли» отношение  $R_2$ . Отсюда делаем вывод, что условие выборки в этом операторе записано верно (ведь определение разности двух отношений выполняется) и использование ключей, как и в случае реализации операции пересечения, полностью оправдано.

Два случая применения «метода ключей», которые мы рассмотрели, являются самыми распространенными. На этом изучение использования ключей в составлении операторов, представляющих отношения, завершим. Все оставшиеся бинарные операции реляционной алгебры записываются иными способами.

### **Операция декартова произведения.**

Как мы помним из предыдущих лекций, декартово произведение двух отношений-операндов составляется как набор всех возможных пар именованных значений кортежей на атрибутах. Поэтому на языке структурированных запросов операция декартова произведения реализуется при помощи перекрестного соединения, обозначаемого ключевым словом **cross join**, что буквально и переводится «перекрестное объединение» или «перекрестное соединение».

Оператор **Select** в конструкции, представляющей операцию декартова произведения на языке структурированных запросов, присутствует только один и имеет следующий вид:

**Select \***

**From R<sub>1</sub> cross join R<sub>2</sub>**

Здесь R<sub>1</sub> и R<sub>2</sub> – имена исходных отношений-операндов. Опция **cross join** обеспечивает, что в результирующее отношение запишутся все атрибуты (все, потому что в первой строчке оператора поставлен значок «\*»), соответствующие всем парам кортежей отношений R<sub>1</sub> и R<sub>2</sub>.

Очень важно помнить одну особенность воплощения в жизнь операции декартова произведения. Эта особенность является следствием определения бинарной операции декартова произведения. Напомним его:

$$r_4(S_4) = r_1(S_1) \text{ ? } r_2(S_2) = \{t(S_1 \text{ ? } S_2) \mid t[S_1] \text{ ? } r_1 \ \& \ t(S_2) \text{ ? } r_2\}, S_1 \text{ ? } S_2 = ?;$$

Как видно из приведенного определения, пары кортежей образуются при обязательно непересекающихся схемах отношений. Поэтому и при работе на языке структурированных запросов SQL непременно оговаривается, что исходные отношения-операнды не должны иметь совпадающих имен атрибутов. Но если эти отношения все же имеют одинаковые имена, сложившуюся ситуацию можно легко разрешить с помощью операции переименования атрибутов, т. е. в подобных случаях необходимо просто использовать опцию **as**, о которой упоминалось ранее.

Рассмотрим пример, в котором нужно найти декартово произведение двух отношений, имеющих некоторые имена своих атрибутов совпадающими. Итак, пусть даны следующие отношения:

R<sub>1</sub> ( A, B),

R<sub>2</sub> (B, C);

Мы видим, что атрибуты R<sub>1</sub>.B и R<sub>2</sub>.B имеют одинаковые имена. С учетом этого оператор **Select**, реализующий на языке структурированных запросов эту операцию декартова произведения, будет выглядеть следующим образом:

**Select A, R<sub>1</sub>.B as B1, R<sub>2</sub>.B as B2, C**

**From R<sub>1</sub> cross join R<sub>2</sub>;**

Таким образом, с использованием опции переименования **as**, у машины не возникнет «вопросов», по поводу совпадающих имен двух исходных отношений-операндов.



### Операции внутреннего соединения.

На первый взгляд может показаться странным, что мы рассматриваем операцию внутреннего соединения раньше операции естественного соединения, ведь, когда мы проходили бинарные операции, все было наоборот. Но анализируя выражение операций на языке структурированных запросов, можно прийти к выводу, что операция естественного соединения является частным случаем операции внутреннего соединения. Именно поэтому рационально рассмотреть эти операции как раз в таком порядке.

Итак, для начала вспомним определение операции внутреннего соединения, которое мы проходили раньше:

$$r_1(S_1) \text{ ? }_P r_2(S_2) = ? <P> (r_1 \text{ ? } r_2), S_1 \text{ ? } S_2 = ?.$$

Для нас в этом определении особенно важно то, что рассматриваемые схемы отношений-операндов  $S_1$  и  $S_2$  не должны пересекаться.

Для реализации операции внутреннего соединения в языке структурированных запросов существует специальная опция **inner join**, которая и переводится с английского буквально «внутреннее объединения» или «внутреннее соединение».

Оператор Select в случае осуществления операции внутреннего соединения будет выглядеть следующим образом:

**Select \***

**From R<sub>1</sub> inner join R<sub>2</sub>;**

Здесь, как и раньше,  $R_1$  и  $R_2$  — имена исходных отношений-операндов.

При реализации этой операции нельзя допускать пересечения схем отношений-операндов.

### Операция естественного соединения.

Как мы уже говорили, операция естественного соединения является частным случаем операции внутреннего соединения. Почему? Да потому что при действии естественного соединения кортежи исходных отношений-операндов соединяются по особому условию. А именно по условию равенства кортежей на пересечении отношений-операндов,

тогда как при действии операции внутреннего соединения такой ситуации допускать было бы нельзя.

Так как рассматриваемая нами операция естественного соединения является частным случаем операции внутреннего соединения, для ее реализации используется та же опция, что и для предыдущей рассмотренной операции, т. е. опция **inner join**. Но поскольку при составлении оператора Select для операции естественного соединения необходимо еще учесть условие равенства кортежей исходных отношений-операндов на пересечении их схем, то дополнительно к означенной опции применяется ключевое слово **on**. В переводе с английского, это буквально означает «на», а применительно к нашему смыслу, можно перевести как «при условии».

Общий вид оператора Select для выполнения операции естественного соединения следующий:

**Select \***

**From** *имя отношения 1* **inner join** *имя отношения 2*

**on** *условие равенства кортежей;*

Рассмотрим пример.

Пусть даны два отношения:

$R_1 (A, B, C),$

$R_2 (B, C, D);$

Операцию естественного соединения этих отношений можно реализовать с помощью следующего оператора:

**Select**  $A, R_1.B, R_1.C, D$

**From**  $R_1$  **inner join**  $R_2$

**on**  $R_1.B = R_2.B$  and  $R_1.C = R_2.C$

В итоге этой операции в результат выведутся атрибуты, указанные в первой строке оператора Select, соответствующие кортежам, равным на указанном пересечении.

Следует заметить, что здесь мы обращаемся к общим атрибутам B и C не просто по именам. Это необходимо делать не по той причине, что и в случае реализации операции декартова произведения, а потому, что в противном случае будет не ясно, к какому отношению они относятся.

Интересно, что использованная формулировка условия соединения ( $R_1.B = R_2.B$  and  $R_1.C = R_2.C$ ) предполагает, что общие атрибуты

соединяемых отношений Null-значений не допускают. Это изначально встроено в систему языка структурированных запросов.

### **Операция левого внешнего соединения.**

Выражение на языке структурированных запросов SQL операции левого внешнего соединения получается из реализации операции естественного соединения заменой ключевого слова **inner** на ключевое слово **left outer**.

Таким образом, на языке структурированных запросов эта операция запишется следующим образом:

**Select \***

**From** *имя отношения 1* **left outer join** *имя отношения 2*

**on** *условие равенства кортежей;*

### **Операция правого внешнего соединения.**

Выражение для операции правого внешнего соединения на языке структурированных запросов получается из осуществления операции естественного соединения заменой ключевого слова **inner** на ключевое слово **right outer**.

Итак, получаем, что на языке структурированных запросов SQL операция правого внешнего соединения запишется следующим образом:

**Select \***

**From** *имя отношения 1* **right outer join** *имя отношения 2*

**on** *условие равенства кортежей;*

### **Операция полного внешнего соединения.**

Выражение на языке структурированных запросов операции полного внешнего соединения получается, как и в двух предыдущих случаях, из выражения для операции естественного соединения путем замены ключевого слова **inner** на ключевое слово **full outer**.

Таким образом, на языке структурированных запросов эта операция запишется так:

**Select \***

**From** *имя отношения 1* **full outer join** *имя отношения 2*

**on** *условие равенства кортежей;*

Очень удобно, что в семантику языка структурированных запросов SQL изначально встроены эти опции, ведь иначе каждому программисту приходилось бы выводить их самостоятельно и вводить в каждую новую базу данных.

## ИСПОЛЬЗОВАНИЕ ПОДЗАПРОСОВ

Как можно было понять из пройденного материала, понятие «подзапрос» в языке структурированных запросов является понятием базовым и довольно широко применимым (иногда, кстати, их еще называют SQL-запросами. Действительно, практика программирования и работы с базами данных показывает, что составление системы подзапросов для решения различных сопутствующих задач – деятельность гораздо более благодарная по сравнению с какими-то другими приемами работы со структурированной информацией. Поэтому, рассмотрим пример для лучшего понимания действий с подзапросами, их составлением и использованием.

Пусть имеется следующий фрагмент некой базы данных, которая вполне может использоваться в каком-либо учебном заведении:

Предметы (*Код предмета*, Имя предмета);

Студенты (*№ зачетной книжки*, Фамилия, Имя, Отчество);

Сессия (*Код предмета*, *№ зачетной книжки*, Оценка);

Сформулируем SQL-запрос, возвращающий ведомость с указанием номера зачетной книжки, фамилии и инициалов студента и оценки для предмета с наименованием «Базы данных». Такую информацию в университетах необходимо получать всегда и своевременно, поэтому приведенный далее запрос является едва ли не самой востребованной единицей программирования с использованием таких баз данных.

Для удобства работы, дополнительно предположим, что атрибуты «Фамилия», «Имя» и «Отчество» не допускают Null-значений и не являются пустыми. Это требование вполне объяснимо и закономерно, ведь в базу данных любого учебного заведения первыми из данных на нового ученика вводятся именно данные о его фамилии, имени и отчестве. И само собой разумеется, что не может быть записи в подобной базе данных, в которой присутствуют данные на ученика, но при этом неизвестно его имя.

Заметим, что атрибут «Имя предмета» схемы отношения «Предметы» является ключом, поэтому, как следует из определения (подробнее об этом будет сказано дальше), все наименования

предметов являются уникальными. Это тоже понятно и без пояснения представления ключа, ведь все преподающиеся в учебном заведении предметы должны иметь и имеют различные имена.

Теперь, прежде чем мы приступим к составлению текста самого оператора, введем в рассмотрение две функции, которые нам пригодятся по мере нашей деятельности.

Во-первых, нам будет полезна функция **Trim**, записывается Trim («строка»), т. е. аргументом этой функции является строка. Что делает эта функция? Она возвращает сам аргумент без пробелов, стоящих в начале и в конце этой строки, т. е., эту функцию применяют, например, в случаях: Trim («Богучарников») или Trim («Максиме-енко»), когда после или до аргумента стоят по несколько лишних пробелов.

А во-вторых, необходимо также рассмотреть функцию Left, которая записывается Left (строка, число), т. е. функцию от уже двух аргументов, одним из которых является, как и раньше, строка. Второй ее аргумент – число, оно показывает, сколько символов из левой части строки следует вывести в результат.

Например, результатом операции:

Left («Михаил, 1») + «.» + Left («Зиновьевич, 1»)

будут инициалы «М. З.». Именно для выведения инициалов студентов мы и будем использовать эту функцию в нашем запросе.

Итак, приступим к составлению искомого запроса.

Для начала составим небольшой вспомогательный запрос, который потом используем в основном, главном запросе:

**Select** № зачетной книжки, Оценка

**From** Сессия

**Where** Код предмета = (Select Код предмета

**From** Предметы

**Where** Имя предмета = «Базы данных»)

**as** «Оценки „Базы данных“»;

Применение здесь опции as означает, что мы присвоили этому запросу псевдоним «Оценки „Базы данных“». Сделали мы это для удобства дальнейшей работы с этим запросом.

Далее, в этом запросе подзапрос:

**Select** Код предмета

**From** *Предметы*

**Where** *Имя предмета* = «*Базы данных*»;

позволяет выделить из отношения «Сессия» те кортежи, которые относятся к рассматриваемому предмету, т. е. к базам данных.

Интересно, что этот внутренний подзапрос может возвращать не более одного значения, так как атрибут «Имя предмета» является ключом отношения «Предметы», т. е. все его значения уникальны.

А весь запрос «*Оценки „Базы данных“*» позволяет выделить из отношения «Сессия» данные о тех студентах (их номера зачетных книжек и оценки), которые удовлетворяют условию, оговоренному в подзапросе, т. е. информацию о предмете под названием «База данных».

Теперь составим основной запрос, используя уже полученные результаты.

**Select** *Студенты. № зачетной книжки,*

**Trim** (*Фамилия*) + « » + **Left** (*Имя*, 1) + «.» + **Left** (*Отчество*, 1) + «.» **as** *ФИО, Оценки «Базы данных». Оценка*

**From** *Студенты* **inner join**

(

**Select** *№ зачетной книжки, Оценка*

**From** *Сессия*

**Where** *Код предмета* = (**Select** *Код предмета*

**From** *Предметы*

**Where** *Имя предмета* = «*Базы данных*»)

) **as** «*Оценки „Базы данных“*».

**on** *Студенты. № зачетной книжки = Оценки «Базы данных». № зачетной книжки.*

Итак, сначала мы перечисляем атрибуты, которые будет необходимо вывести, после окончания работы запроса. Необходимо упомянуть, что атрибут «№ зачетной книжки» из отношения *Студенты*, оттуда же – атрибуты «Фамилия», «Имя» и «Отчество». Правда, два последних атрибута выводим не полностью, а только первые буквы. Также мы упоминаем атрибут «Оценка» из запроса *Оценки «Базы данных*, которое ввели раньше.

Выбираем мы все эти атрибуты из внутреннего соединения отношения «Студенты» и запроса «Оценки „Базы данных“». Это внутреннее соединение, как мы можем видеть, берется нами по условию равенства номеров зачетной книжки. В результате этой операции внутреннего соединения, к отношению «Студенты» добавляются оценки.

Надо заметить, что так как атрибуты «Фамилия», «Имя» и «Отчество» по условию не допускают Null-значений и не являются пустыми, то формула вычисления, возвращающая атрибут «ФИО» ( $\text{Trim (Фамилия)} + \text{« »} + \text{Left (Имя, 1)} + \text{«.»} + \text{Left (Отчество, 1)} + \text{«.»asФИО}$ ), соответственно не требует дополнительных проверок, упрощается.



## **ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ**

Разработать select-запросы к базе данных, разработанной в предыдущей лабораторной. Запросы должны включать в себя перечисленные в теоретической части унарные операции, бинарные операции, а также подзапросы.

## **КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ**

1. Опишите различие механизмов реляционной алгебры и реляционного исчисления.
2. Перечислите операции реляционной алгебры.
3. Раскройте понятие «теоретико-множественные» операций реляционной алгебры.
4. Перечислите зависимые операции реляционной алгебры.
5. Перечислите предложения оператора SELECT, соответствующие операциям реляционной алгебры.
6. Опишите роль псевдонимов.
7. Опишите назначение предложения WHERE?
8. Опишите назначение предложения GROUP BY?
9. Опишите назначение предложения HAVING.

## **ФОРМА ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ**

На выполнение лабораторной работы отводится 3 занятия (6 академических часов: 5 часов на выполнение и сдачу лабораторной работы и 1 час на подготовку отчета).

Номер варианта студенту выдается преподавателем.

Отчет на защиту предоставляется в печатном виде.

Структура отчета (на отдельном листе(-ах)): титульный лист, формулировка задания (вариант), ход выполнения, результаты выполнения работы, выводы.

## ОСНОВНАЯ ЛИТЕРАТУРА

1. Карпова, Т.С. Базы данных: модели, разработка, реализация : учебное пособие / Т.С. Карпова. - 2-е изд., исправ. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 241 с. : ил. ; То же [Электронный ресурс]. - URL: <http://biblioclub.ru/index.php?page=book&id=429003>
2. Давыдова, Е.М. Базы данных [Электронный ресурс] : учеб. пособие / Е.М. Давыдова, Н.А. Новгородова. — Электрон. дан. — Москва : ТУСУР, 2007. — 166 с. — Режим доступа: <https://e.lanbook.com/book/11636>. — Загл. с экрана.
3. Харрингтон, Д. Проектирование объектно ориентированных баз данных [Электронный ресурс] — Электрон. дан. — Москва : ДМК Пресс, 2007. — 272 с. — Режим доступа: <https://e.lanbook.com/book/1231>. — Загл. с экрана.

## ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

4. . Голицина О.Л., Максимов Н.В., Попов И.И. Базы данных: учеб. пособие. – М.: Форум:Инфра-М, 2007.
5. 5. Гагарин Ю.Е. Применение языка SQL в MS Access: учебно-методическое пособие. – М.: МГТУ им. Н.Э. Баумана, 2012.

### Электронные ресурсы:

1. Научная электронная библиотека <http://eLIBRARY.RU>
2. Электронно-библиотечная система <http://e.lanbook.com>
3. Электронно-библиотечная система «Университетская библиотека онлайн» <http://biblioclub.ru>
4. Электронно-библиотечная система IPRBook <http://www.iprbookshop.ru>