Санкт-Петербургский Национальный Исследовательский Университет ИТМО

Факультет Программной Инженерии и Компьютерной Техники



Вариант №434300

Лабораторная работа №2

По дисциплине

Базы Данных

Выполнил студент группы P3119:

Ануфриев Андрей Сергеевич

Преподаватель:

Бойко Владислав Алексеевич

Санкт-Петербург 2025 г.

Оглавление

[Текст задания 3](#_Toc196083282)

[Реализация запросов на SQL 5](#_Toc196083283)

[Вывод 8](#_Toc196083284)

[Ответы на вопросы 8](#_Toc196083285)

# Текст задания

По варианту, выданному преподавателем, составить и выполнить запросы к базе данных "Учебный процесс".

Команда для подключения к базе данных ucheb:

psql -h pg -d ucheb

Отчёт по лабораторной работе должен содержать:

1. Текст задания.
2. Реализацию запросов на SQL.
3. Выводы по работ.

Темы для подготовки к защите лабораторной работы:

1. SQL
2. Соединение таблиц
3. Подзапросы
4. Представления
5. Последовательности

Составить запросы на языке SQL (пункты 1-7).

1. Сделать запрос для получения атрибутов из указанных таблиц, применив фильтры по указанным условиям:  
   Таблицы: Н\_ЛЮДИ, Н\_ВЕДОМОСТИ.  
   Вывести атрибуты: Н\_ЛЮДИ.ОТЧЕСТВО, Н\_ВЕДОМОСТИ.ИД.  
   Фильтры (AND):  
   a) Н\_ЛЮДИ.ИМЯ > Роман.  
   b) Н\_ВЕДОМОСТИ.ДАТА < 2010-06-18.  
   c) Н\_ВЕДОМОСТИ.ДАТА = 2022-06-08.  
   Вид соединения: INNER JOIN.
2. Сделать запрос для получения атрибутов из указанных таблиц, применив фильтры по указанным условиям:  
   Таблицы: Н\_ЛЮДИ, Н\_ОБУЧЕНИЯ, Н\_УЧЕНИКИ.  
   Вывести атрибуты: Н\_ЛЮДИ.ФАМИЛИЯ, Н\_ОБУЧЕНИЯ.ЧЛВК\_ИД, Н\_УЧЕНИКИ.НАЧАЛО.  
   Фильтры: (AND)  
   a) Н\_ЛЮДИ.ИМЯ = Владимир.  
   b) Н\_ОБУЧЕНИЯ.НЗК > ‘999080.  
   Вид соединения: LEFT JOIN.
3. Вывести число имен без учета повторений.  
   При составлении запроса нельзя использовать DISTINCT.
4. В таблице Н\_ГРУППЫ\_ПЛАНОВ найти номера планов, по которым обучается (обучалось) более 2 групп ФКТИУ.  
   Для реализации использовать соединение таблиц.
5. Выведите таблицу со средними оценками студентов группы 4100 (Номер, ФИО, Ср\_оценка), у которых средняя оценка равна максимальной оценк(е|и) в группе 1101.
6. Получить список студентов, зачисленных до первого сентября 2012 года на первый курс очной или заочной формы обучения. В результат включить:  
   номер группы;  
   номер, фамилию, имя и отчество студента;  
   номер и состояние пункта приказа;  
   Для реализации использовать соединение таблиц.
7. Сформировать запрос для получения числа на ФКТИУ хорошистов.

Реализация запросов на SQL

1. Сделать запрос для получения атрибутов из указанных таблиц, применив фильтры по указанным условиям:  
   Таблицы: Н\_ЛЮДИ, Н\_ВЕДОМОСТИ.  
   Вывести атрибуты: Н\_ЛЮДИ.ОТЧЕСТВО, Н\_ВЕДОМОСТИ.ИД.  
   Фильтры (AND):  
   a) Н\_ЛЮДИ.ИМЯ > Роман.  
   b) Н\_ВЕДОМОСТИ.ДАТА < 2010-06-18.  
   c) Н\_ВЕДОМОСТИ.ДАТА = 2022-06-08.  
   Вид соединения: INNER JOIN.

*----1--  
--Надо вывести Н\_ЛЮДИ.ОТЧЕСТВО и Н\_ВЕДОМОСТИ.ИД*SELECT  
 Н\_ЛЮДИ.ОТЧЕСТВО,  
 Н\_ВЕДОМОСТИ.ИД  
FROM *--берём сначала таблицу Н\_ЛЮДИ* Н\_ЛЮДИ  
INNER JOIN *--потом объединяем её с Н\_ВЕДОМОСТИ, так чтобы совпадали ИД* Н\_ВЕДОМОСТИ ON Н\_ЛЮДИ.ИД = Н\_ВЕДОМОСТИ.ЧЛВК\_ИД  
WHERE  
 Н\_ЛЮДИ.ИМЯ > 'Роман' *--а)Н\_ЛЮДИ.ИМЯ > 'Роман' Роман в кавычках, т.к. субд должен искать по совпадению строки и вывести всё, что больше по алфавитному порядку* AND Н\_ВЕДОМОСТИ.ДАТА > '2010-06-18' *--б)Н\_ВЕДОМОСТИ.ДАТА < '2010-06-18' также ищем как значение в столбце, которое меньше заданной* AND Н\_ВЕДОМОСТИ.ДАТА < '2022-06-08';*--в) ищет точное совпадение со строкой  
 --б) и в) никогда вместе не выполняются*

\*2 и 3 условие никогда не выполняются вместе, надо тогда их по отдельности запрашивать

1. Сделать запрос для получения атрибутов из указанных таблиц, применив фильтры по указанным условиям:  
   Таблицы: Н\_ЛЮДИ, Н\_ОБУЧЕНИЯ, Н\_УЧЕНИКИ.  
   Вывести атрибуты: Н\_ЛЮДИ.ФАМИЛИЯ, Н\_ОБУЧЕНИЯ.ЧЛВК\_ИД, Н\_УЧЕНИКИ.НАЧАЛО.  
   Фильтры: (AND)  
   a) Н\_ЛЮДИ.ИМЯ = Владимир.  
   b) Н\_ОБУЧЕНИЯ.НЗК > 999080.  
   Вид соединения: LEFT JOIN.

*----2----  
--Надо вывести Н\_ЛЮДИ.ФАМИЛИЯ Н\_ОБУЧЕНИЯ.ЧЛВК\_ИД Н\_УЧЕНИКИ.НАЧАЛО*SELECT  
 Н\_ЛЮДИ.ФАМИЛИЯ,  
 Н\_ОБУЧЕНИЯ.ЧЛВК\_ИД,  
 Н\_УЧЕНИКИ.НАЧАЛО  
FROM *--берём сначала таблицу Н\_ЛЮДИ* Н\_ЛЮДИ  
LEFT JOIN *--потом объединяем её с Н\_ОБУЧЕНИЯ по ИД* Н\_ОБУЧЕНИЯ ON Н\_ЛЮДИ.ИД = Н\_ОБУЧЕНИЯ.ЧЛВК\_ИД  
LEFT JOIN *--потом объединяем результат предыдущей операции с Н\_УЧЕНИКИ также по ИД* Н\_УЧЕНИКИ ON Н\_ЛЮДИ.ИД = Н\_УЧЕНИКИ.ЧЛВК\_ИД  
WHERE  
 Н\_ЛЮДИ.ИМЯ = 'Владимир'*--а)фильтр на атрибут ИМЯ = 'Владимир'* AND Н\_ОБУЧЕНИЯ.НЗК > '990080';*--б)фильтр на атрибут НЗК > '999080'(сравниваем как со строкой) иначе надо приводить НЗК к INT*

1. Вывести число имен без учета повторений.  
   При составлении запроса нельзя использовать DISTINCT.

*----3----  
--надо вывести кол-во уникальных имён*SELECT *COUNT*(*\**) AS количество\_уникальных\_имен *--COUNT(\*) считает кол-во строк в подзапросе*FROM (  
 SELECT Н\_ЛЮДИ.ИМЯ  
 FROM Н\_ЛЮДИ  
 GROUP BY Н\_ЛЮДИ.ИМЯ *--группирует все записи по полю ИМЯ и убирает дубликаты*) AS уникальные\_имена;

1. В таблице Н\_ГРУППЫ\_ПЛАНОВ найти номера планов, по которым обучается (обучалось) более 2 групп ФКТИУ.  
   Для реализации использовать соединение таблиц.

*----4--*SELECT Н\_ГРУППЫ\_ПЛАНОВ.ПЛАН\_ИД FROM Н\_ГРУППЫ\_ПЛАНОВ  
JOIN  
 Н\_ПЛАНЫ ON Н\_ГРУППЫ\_ПЛАНОВ.ПЛАН\_ИД = Н\_ПЛАНЫ.ИД  
JOIN *--объединяем с таблицей Н\_ПЛАНЫ по ИД* Н\_ОТДЕЛЫ ON Н\_ПЛАНЫ.ОТД\_ИД = Н\_ОТДЕЛЫ.ИД  
WHERE *--фильтр только на КТиУ* Н\_ОТДЕЛЫ.КОРОТКОЕ\_ИМЯ = 'КТиУ'  
GROUP BY *--груперуем по ПЛАН\_ИД* Н\_ГРУППЫ\_ПЛАНОВ.ПЛАН\_ИД  
HAVING *-- и выводим только те, которых в группе больше 2  
 COUNT*(Н\_ГРУППЫ\_ПЛАНОВ.ПЛАН\_ИД) > 2;

1. Выведите таблицу со средними оценками студентов группы 4100 (Номер, ФИО, Ср\_оценка), у которых средняя оценка равна максимальной оценк(е|и) в группе 1101.

SELECT avg\_grade *--выведем максимальное значение из средних оценок каждого человека 1101группы* FROM (  
 SELECT *AVG*(*CAST*(В.ОЦЕНКА AS int)) as avg\_grade*--выведем среднее значение оценок ученика* FROM Н\_ВЕДОМОСТИ В  
 JOIN Н\_УЧЕНИКИ У ON В.ЧЛВК\_ИД = У.ЧЛВК\_ИД *--подключаем таблицу Н\_УЧЕНИКИ, чтобы получить Группу* WHERE У.ГРУППА = '1101'*--фильтр на группу* AND В.ОЦЕНКА ~ '^[0-5]+$' ;

*----5--  
--MAX()-максимальное значение по атрибуту в скобках  
--AVG()- среднее значение, игнорирует NULL  
--CAST(\_ AS \_)-преобразует типы  
-- Находим максимальную среднюю оценку в группе 1101*WITH max\_grade AS (  
 SELECT *MAX*(avg\_grade) as max\_value*--выведем максимальное значение из средних оценок каждого человека 1101группы* FROM (  
 SELECT *AVG*(*CAST*(В.ОЦЕНКА AS int)) as avg\_grade*--выведем среднее значение оценок ученика* FROM Н\_ВЕДОМОСТИ В  
 JOIN Н\_УЧЕНИКИ У ON В.ЧЛВК\_ИД = У.ЧЛВК\_ИД *--подключаем таблицу Н\_УЧЕНИКИ, чтобы получить Группу* WHERE У.ГРУППА = '1101'*--фильтр на группу* AND В.ОЦЕНКА ~ '^[0-5]+$' *--фильтр только на числовые оценки. (нам надо зачёт и незачёт скипнуть)* GROUP BY В.ЧЛВК\_ИД*--делаем группы по каждому студенту* ) t *--алиас - типо имя таблицы для подзапроса FROM (оно обязательно из синтаксиса SQL, но дальше использоваться не будет*),  
*--Выбираем студентов 4100 с такой же средней оценкой*students AS (  
 SELECT  
 Л.ИД AS Номер,  
 Л.ФАМИЛИЯ ||' '|| Л.ИМЯ ||' '|| Л.ОТЧЕСТВО AS ФИО,*--эти поля не пустые, худшее, что там может быть это .  
 AVG*(*CAST*(В.ОЦЕНКА AS int)) AS Ср\_оценка*--берём среднее значение из оценок по каждому ученику* FROM Н\_ЛЮДИ Л  
 JOIN Н\_УЧЕНИКИ У ON Л.ИД = У.ЧЛВК\_ИД  
 JOIN Н\_ВЕДОМОСТИ В ON Л.ИД = В.ЧЛВК\_ИД *--подключаем Н\_УЧЕНИКИ и Н\_ВЕДОМОСТИ по ЧЛВК\_ИД* WHERE У.ГРУППА = '4100'  
 AND В.ОЦЕНКА ~ '^[0-5]+$'  
 GROUP BY Л.ИД, Л.ФАМИЛИЯ, Л.ИМЯ, Л.ОТЧЕСТВО  
 HAVING *AVG*(*CAST*(В.ОЦЕНКА AS int)) = (SELECT max\_value FROM max\_grade)  
)  
*-- Выводим результат*SELECT *\** FROM students;

1. Получить список студентов, зачисленных до первого сентября 2012 года на первый курс очной или заочной формы обучения. В результат включить:  
   номер группы;  
   номер, фамилию, имя и отчество студента;  
   номер и состояние пункта приказа;  
   Для реализации использовать соединение таблиц.

*----6--*SELECT  
 У.ГРУППА AS Номер\_группы,  
 Л.ИД AS Номер\_студента,  
 Л.ФАМИЛИЯ AS Фамилия,  
 Л.ИМЯ AS Имя,  
 Л.ОТЧЕСТВО AS Отчество,  
 У.П\_ПРКОК\_ИД AS Номер\_приказа,  
 У.СОСТОЯНИЕ AS Состояние  
FROM  
 Н\_ЛЮДИ Л  
 JOIN  
 Н\_УЧЕНИКИ У ON Л.ИД = У.ЧЛВК\_ИД  
 JOIN  
 Н\_ПЛАНЫ П ON У.ПЛАН\_ИД = П.ИД  
 JOIN  
 Н\_ФОРМЫ\_ОБУЧЕНИЯ ФО ON П.ФО\_ИД = ФО.ИД  
WHERE  
 У.НАЧАЛО < '2012-09-01' *-- Зачислен до 1 сентября 2012* AND П.КУРС = 1 *-- Первый курс* AND ФО.НАИМЕНОВАНИЕ IN ('Очная', 'Заочная') *-- Форма обучения(нет заочной, есть только Очно-заочная(вечерняя))*ORDER BY  
 У.ГРУППА, Л.ФАМИЛИЯ, Л.ИМЯ, Л.ОТЧЕСТВО;*--сортируем сначала по группе потом по фамилии и т.д*

1. Сформировать запрос для получения числа на ФКТИУ хорошистов.

*----7--*SELECT *COUNT*(*\**) AS Количество\_хорошистов  
FROM (  
 SELECT Н\_ВЕДОМОСТИ.ЧЛВК\_ИД  
 FROM Н\_ВЕДОМОСТИ  
 JOIN Н\_ОЦЕНКИ ON Н\_ВЕДОМОСТИ.ОЦЕНКА = Н\_ОЦЕНКИ.КОД  
 JOIN Н\_УЧЕНИКИ ON Н\_ВЕДОМОСТИ.ЧЛВК\_ИД = Н\_УЧЕНИКИ.ЧЛВК\_ИД  
 JOIN Н\_ПЛАНЫ ON Н\_УЧЕНИКИ.ПЛАН\_ИД = Н\_ПЛАНЫ.ИД  
 JOIN Н\_ОТДЕЛЫ ON Н\_ПЛАНЫ.ОТД\_ИД = Н\_ОТДЕЛЫ.ИД  
 WHERE Н\_ОТДЕЛЫ.КОРОТКОЕ\_ИМЯ = 'КТиУ'  
 GROUP BY Н\_ВЕДОМОСТИ.ЧЛВК\_ИД  
 HAVING *COUNT*(CASE WHEN Н\_ОЦЕНКИ.КОД IN ('неявка', 'незач', '3', '2') THEN 1 END) = 0) AS хорошисты;

# Вывод

В ходе работы я узнал структуру базы данных "Учебный процесс", узнал как работают подзапросы, соединение таблиц. Потренировался составлять более сложные запросы к субд.

# Ответы на вопросы

ANY\SOME

* Подзапрос должен возвращать ровно один столбец
* Значение выражения сравнивается со значением в каждой

строке результата подзапроса с помощью заданного

оператора, который должен возвращать логическое

значение.

* Результатом ANY будет «true», если хотя бы для одной

строки условие истинно, и «false» в противном случае (в

том числе, если подзапрос не возвращает строк).

* Оператор IN аналогичен = ANY (= SOME)

1. **SQL**

**WHERE** — для определения, какие строки должны быть выбраны или включены в GROUP BY.

**GROUP BY** — для объединения строк с общими значениями в элементы меньшего набора строк.

**HAVING** — для определения, какие строки после GROUP BY должны быть выбраны.

**ORDER BY** — сортировка результирующего набора данных

**DISTINCT -** Указывает, что для вычислений применяются только уникальные значения столбца.

**AVG**()- среднее значение, игнорирует NULL

**COUNT**() — подсчитывает количество строк в наборе результатов или количество ненулевых значений в столбце.

**CAST** **(expression AS target\_type)** для преобразования значения одного типа данных в другой. Можно и через :: SELECT '123'::INTEGER;

**Агрегатные функции это MIN, MAX, SUM, AVG, COUNT, COUNT\_BIG**

1. **Соединение таблиц**

**OUTER** - это внешнее соединение. Оно возвращает не только строгое пересечение между двумя таблицами, но и отдельные элементы, которые принадлежат только одному из множеств.

**USING** — сокращённая запись условия, когда с обеих сторон соединения столбцы имеют одинаковые имена.

**NATURAL** — упрощённая форма USING: образует список USING из всех имён столбцов, существующих в обеих входных таблицах. Если столбцов с одинаковыми именами нет, NATURAL работает как CROSS JOIN

**1. INNER JOIN (Внутреннее соединение)**

Возвращает только строки, где есть совпадения в обеих таблицах.

SELECT a.\*, b.\*

FROM table\_a a

INNER JOIN table\_b b ON a.id = b.a\_id;

**2. LEFT JOIN (LEFT OUTER JOIN)**

Возвращает все строки из левой таблицы и соответствующие строки из правой. Если соответствия нет, возвращает NULL для правой таблицы.

SELECT a.\*, b.\*

FROM table\_a a

LEFT JOIN table\_b b ON a.id = b.a\_id;

**3. NATURAL JOIN**

Соединяет таблицы по столбцам с одинаковыми именами.

**4. FULL JOIN (FULL OUTER JOIN)**

Возвращает все строки из обеих таблиц, с NULL значениями там, где нет соответствий.

SELECT a.\*, b.\*

FROM table\_a a

FULL JOIN table\_b b ON a.id = b.a\_id;

**5. CROSS JOIN**

Декартово произведение - каждая строка из первой таблицы соединяется с каждой строкой из второй.

1. **Подзапросы**

Вложенный подзапрос (subquery) — предложение SELECT, которое заключено в круглые скобки и вложено в WHERE/HAVING часть другого SQL предложения

Коррелированные вложенные подзапросы

Вложенный подзапрос не может быть выполнен до обработки внешнего запроса: SELECT Surname FROM STUDENT WHERE EXISTS ( SELECT 1 FROM STUDENT\_OLYMPIAD WHERE StID = STUDENT.StudentID );

1. Представления

Представление — именованный запрос.

CREATE VIEW VIEW\_NAME

[ ( ColumnName [, ...] ) ]

AS подзапрос

Материализованные представления

CREATE MATERIALIZED VIEW PICTStudents3 AS

(PICTId, pSurname) AS

SELECT StudentID, Surname FROM STUDENT

WHERE GroupID IN (

SELECT GroupID FROM GROUP

WHERE GroupName LIKE 'P3%' );

Результат запроса сохраняется в базе данных.

Для обновления данных:

REFRESH MATERIALIZED VIEW PICTStudents3;

1. Последовательности

Последовательности это – это определяемые пользователем объекты базы данных, предназначенные для генерации ряда числовых значений. В отличие от столбцов идентификаторов, которые тесно связаны с конкретными таблицами, последовательности являются независимыми объектами и могут использоваться в нескольких таблицах.

Автоматическая генерация первичного ключа: последовательности автоматически генерируют уникальные значения, которые можно использовать в качестве первичных или уникальных ключей в таблицах баз данных.

По возрастанию или по убыванию: последовательности можно настроить на генерацию чисел в порядке возрастания или убывания.

Использование нескольких таблиц: одна последовательность может использоваться для генерации значений для нескольких таблиц, что делает ее гибкой и пригодной для повторного использования.

Независимые от таблиц: в отличие от столбцов идентификаторов, последовательности являются независимыми и могут использоваться в разных таблицах.

CREATE SEQUENCE sequence\_1  
start with 1  
increment by 1  
minvalue 0  
maxvalue 100  
cycle;

SELECT nextval('sequence\_1);