**Лабораторные работы по курсу**

**Базы данных**

**Лабораторная работа 2**

**«Оператор SELECT языка SQL»**

**Москва, 2024**

Оглавление

[1. Теоретическая часть 3](#_Toc159597613)

[1.1. Фильтрация строк 3](#_Toc159597614)

[1.2. Условия отбора строк 4](#_Toc159597615)

[1.3. Проверка на членство во множестве 5](#_Toc159597616)

[1.4. Поиск с использованием шаблона 5](#_Toc159597617)

[1.5. Вычисляемые столбцы 7](#_Toc159597618)

[1.6. Агрегирование и группировка 7](#_Toc159597619)

[1.7. Сортировка 9](#_Toc159597620)

[1.8. Оконные функции 9](#_Toc159597621)

[1.9. Несколько важных замечаний 10](#_Toc159597622)

[2. Практическая часть 12](#_Toc159597623)

[2.1. Задание 1 12](#_Toc159597624)

[2.2. Задание 2 13](#_Toc159597625)

[2.3. Задание 3 13](#_Toc159597626)

[Контрольные вопросы 17](#_Toc159597627)

[Список использованной литературы 17](#_Toc159597628)

1. Теоретическая часть

Основой работы с базами данных является умение правильно составить запрос на фильтрацию данных. Для данной задачи используется оператор SELECT. Ниже представлен сокращенный синтаксис данного оператора. Более подробно можно прочитать в документации [1].

SELECT [ ALL | DISTINCT [ ON ( выражение [, ...] ) ] ]

[ \* | выражение [ [ AS ] имя\_результата ] [, ...] ]

[ FROM элемент\_FROM [, ...] ]

[ WHERE условие ]

[ GROUP BY элемент\_группирования [, ...] ]

[ HAVING условие ]

[ { UNION | INTERSECT | EXCEPT } [ ALL | DISTINCT ] выборка ]

[ ORDER BY выражение [ ASC | DESC | USING оператор ] ]

[ LIMIT { число | ALL } ]

[ OFFSET начало [ ROW | ROWS ] ]

Оператор SELECTполучает строки из базы данных и возвращает их в виде таблицы результатов запроса. Результат выполнения запроса может оказаться пустым множеством. Ключевые слова указываются строго в порядке, приведенном выше.

Рассмотрим выполнение SQL запроса на фильтрацию данных по шагам.

* 1. Фильтрация строк

Первыми и обязательными элементами запроса являются ключевые слова SELECT и FROM. После ключевого слова SELECT указывается список столбцов, значения которых необходимо вывести. Возвращаемые столбцы могут быть получены непосредственно из базы данных или вычислены во время выполнения запроса. Для краткого обозначения того, что необходимо вывести все столбцы таблицы используют символ \*.

После ключевого слова FROM указывается имя таблицы, из которой будет производиться выборка данных. Данная таблица может быть реальной таблицей из базы данных, или виртуальной, полученной в результате подзапроса.

Рассмотрим пример. [[1]](#footnote-1)

SELECT \*

FROM students;

student\_id | last\_name | first\_name | patronymic | student…number | birthday | email

------------+--------------+------------+------------+----------------+------------+----------------------

847516 | Блюм | Андрей | Антонович | ИВТ-11 | 2005-09-01 | BljumAndrej@miet.ru

813156 | Верховенский | Петр | Степанович | ИВТ-11 | 2005-12-15 | VerhovenskijPetr@miet.ru

844688 | Верховенский | Степан | Трофимович | ИВТ-11 | 2004-11-18 | VerhovenskijStepan@miet.ru

862231 | Виргинская | Арина | Прохоровна | ИВТ-11 | 2004-07-02 | VirginskajaArina@miet.ru

816390 | Гаганов | Артемий | Павлович | ИВТ-11 | 2004-09-23 | GaganovArtemij@miet.ru

830114 | Гаганов | Павел | Павлович | ИВТ-11 | 2004-06-20 | GaganovPavel@miet.ru

827840 | Дроздов | Маврикий | Николаевич | ИВТ-11 | 2005-07-15 | DrozdovMavrikij@miet.ru

834759 | Дроздова | Прасковья | Ивановна | ИВТ-11 | 2005-03-17 | DrozdovaPraskovja@miet.ru

Данный запрос возвращает список всех студентов университета. Для того, чтобы вывести только ФИО студентов, необходимо вместо знака \* указать названия требуемых полей.

SELECT last\_name, first\_name, patronymic

FROM students;

last\_name | first\_name | patronymic

--------------+------------+------------

Блюм | Андрей | Антонович

Верховенский | Петр | Степанович

Верховенский | Степан | Трофимович

Виргинская | Арина | Прохоровна

Гаганов | Артемий | Павлович

Гаганов | Павел | Павлович

Дроздов | Маврикий | Николаевич

Дроздова | Прасковья | Ивановна

Следующий запрос

SELECT first\_name

FROM students;

Выведет все имена студентов, с учетом повторений одинаковых имен. Чтобы их исключить возможно использовать ключевое слово DISTINCT

SELECT DISTINCT first\_name

FROM students;

Для вывода ограниченного числа строк из базы используется ключевое слово LIMIT. Например, запрос

SELECT last\_name, first\_name, patronymic

FROM students

LINIT 1;

Выведет только одну строку из общего числа.

* 1. Условия отбора строк

Для наложения условий поиска используется оператор WHERE. В качестве операторов сравнения используются операторы: =, <>,>,> =, <, <=.

Например, чтобы найти всех студентов с именем «Сергей», возможно выполнить следующий запрос:

SELECT last\_name, first\_name, patronymic

FROM students

WHERE first\_name = 'Сергей';

last\_name | first\_name | patronymic

------------+------------+---------------

Липутин | Сергей | Васильевич

Сокольский | Сергей | Петрович

Голубков | Сергей | Павлович

Тальберг | Сергей | Иванович

Дорофеев | Сергей | Евгеньевич

Каренин | Сергей | Александрович

Кознышев | Сергей | Иванович

Паратов | Сергей | Сергеич

Обратите внимание, что сравниваемая строка заключается в одинарные кавычки. Об их использовании подробнее будет рассказано ниже.

Аналогично можно найти всех студентов, у которых номер студенческого билета больше числа 850000.

SELECT student\_id, last\_name, first\_name, patronymic

FROM students

WHERE student\_id > 850000;

last\_name | first\_name | patronymic

------------------+------------+----------------

Виргинская | Арина | Прохоровна

Лебядкина | Марья | Тимофеевна

Липутин | Сергей | Васильевич

Ставрогин | Николай | Всеволодович

Ставрогина | Варвара | Петровна

Тушина | Лизавета | Николаевна

Для наложения нескольких условий возможно воспользоваться логическими операторами – AND, OR, NOT.

Например, следующий запрос вернет ФИО студентов, чей день рождения попадает в диапазон с 25 июня 2002 до 25 июня 2003 года включительно.

SELECT last\_name, first\_name, patronymic, birthday

FROM students

WHERE birthday > '25/06/2002' AND birthday < '25/06/2003';

last\_name | first\_name | patronymic | birthday

----------------+------------+---------------+------------

Прозоров | Андрей | Сергеевич | 2003-03-23

Прозорова | Мария | Сергеевна | 2003-01-14

Соленый | Василий | Васильевич | 2003-01-11

Пралинский | Иван | Ильич | 2002-07-17

Зубиков | Аким | Петрович | 2003-02-18

Валковский | Петр | Александрович | 2003-02-14

Сизобрюхов | Степан | Терентьевич | 2002-12-11

Заметов | Александр | Григорьевич | 2003-04-04

Аналогичный запрос можно составить, используя ключевое слово BETWEEN.

SELECT last\_name, first\_name, patronymic, birthday

FROM students

WHERE birthday BETWEEN '25/06/2002' AND '25/06/2003'

* 1. Проверка на членство во множестве

С помощью ключевого слова IN возможно отобрать только те кортежи, заданный атрибут которых находится в указанном списке.

Например, следующий запрос выводит список студентов, кто родился 20 июня 2005 года или 28 июля 2002 года.

**SELECT last\_name, first\_name, birthday**

**FROM students**

**WHERE birthday IN ('20/06/2005', '28/07/2002');**

last\_name | first\_name | birthday

------------+------------+------------

Карамазова | Аделаида | 2005-06-20

Суриков | Иван | 2002-07-28

Степанов | Федор | 2002-07-28

Пежёнова | Анфиса | 2002-07-28

(4 rows)

* 1. Поиск с использованием шаблона

Для наложения более сложных условий поиска возможно воспользоваться оператором поиска шаблонов LIKE и регулярными выражениями.

*строка* LIKE *шаблон* [ESCAPE *спецсимвол*]

*строка* NOT LIKE *шаблон* [ESCAPE *спецсимвол*]

Оператор LIKE сравнивает анализируемую строку с заданным шаблоном и в случае совпадения отбирает эту строку. Для построения шаблона используются следующие спецсимволы:

% - любая последовательность символов

\_ - строго один символ

Также возможно использовать регулярные выражения POSIX.

^ - начало строки

$ - окончание строки

\* - повторение предыдущего символа любое число раз

\ - проверка наличия указанного после \ символа

| - выбор одного из двух вариантов

~ - поиск с учетом регистра

[...] – указание класса символов

При проверке по шаблону LIKE всегда рассматривается вся строка. Поэтому, если нужно найти последовательность символов где-то в середине строки, шаблон должен начинаться и заканчиваться знаками процента.

Например, данный запрос позволяет найти всех студентов, чье имя начинается на букву ‘А’.

**SELECT last\_name, first\_name, patronymic**

**FROM students**

**WHERE first\_name LIKE 'А%';**

last\_name | first\_name | patronymic

---------------+------------+---------------

Блюм | Андрей | Антонович

Виргинская | Арина | Прохоровна

Гаганов | Артемий | Павлович

Кириллов | Алексей | Нилыч

Лембке | Андрей | Антонович

Телятников | Алексей | null

Вельчанинов | Алексей | Иванович

Лобов | Александр | null

Погорельцев | Александр | Павлович

Данный запрос вернет всех студентов, чье имя состоит из 5 символов и заканчивается на букву “я”.

SELECT last\_name, first\_name, patronymic

FROM students

WHERE first\_name LIKE '\_\_\_\_я';

last\_name | first\_name | patronymic

-------------+------------+---------------

Лебядкина | Марья | Тимофеевна

Улитина | Софья | Матвеевна

Шатова | Дарья | Павловна

Шатова | Мария | null

Карамазова | Софья | Ивановна

Прозорова | Мария | Сергеевна

Мармеладова | Софья | Семеновна

Епанчина | Аглая | Ивановна

Рассмотрим пример с регулярными выражениями

SELECT last\_name, first\_name, patronymic

FROM students

WHERE last\_name ~ '^[ЛМН]';

last\_name | first\_name | patronymic

--------------+------------+---------------

Лебядкин | Игнат | Тимофеевич

Лембке | Андрей | Антонович

Липутин | Сергей | Васильевич

Макаров | Михаил | Макарович

Миусов | Петр | Александрович

Нелюдов | Николай | Парфенович

Мейер | Карл | Федорович

Незванова | Анна | null

Данный пример вернет всех студентов, чьи фамилии начинаются на буквы Л, М, Н. Здесь символом ^ обозначается, что следующий символ будет первым в строке, а в квадратных скобках указывается список допустимых символов. Аналогично можно записать через диапазон – [Л-Н]. Символ ~ указывает, что сравнение идет с учетом регистра.

Дополним выражение таким образом, чтобы отбираемые строки оканчивались на буквосочетание `ин`.

SELECT last\_name, first\_name, patronymic

FROM students

WHERE last\_name ~ '^[ЛМН].\*ин$';

last\_name | first\_name | patronymic

-----------+------------+---------------

Лебядк**ин** | Игнат | Тимофеевич

Липутин | Сергей | Васильевич

Луж**ин** | Петр | Петрович

Мышкин | Лев | Николаевич

Нащок**ин** | Ипполит | Александрович

Марк**ин** | Даниэль | Ильич

Лев**ин** | Константин | Дмитриевич

Символом $ указывается, что предыдущие символы будут стоять в конце строки. Т.к. между ними и начальными символами могут находиться еще любое число символов, то обозначим их с помощью ‘.\*’. В данном случае точка обозначает «любой символ», а звездочка продублирует его от 0 до любого числа раз.

Если убрать символ окончания строки $, то будет производиться поиск строк, начинающихся на буквы Л, М, Н и содержащие в себе «ин».

SELECT last\_name, first\_name, patronymic

FROM students

WHERE last\_name ~ '^[ЛМН].\*ин';

last\_name | first\_name | patronymic

------------+------------+---------------

Лебядк**ин** | Игнат | Тимофеевич

Лебядк**ин**а | Марья | Тимофеевна

Липут**ин** | Сергей | Васильевич

Миз**ин**чиков | Иван | Иванович

Луж**ин** | Петр | Петрович

Мышк**ин** | Лев | Николаевич

Нащок**ин** | Ипполит | Александрович

Марк**ин** | Даниэль | Ильич

Лев**ин** | Константин | Дмитриевич

Более подробно про регулярные выражения можно прочитать в источнике [2].

* 1. Вычисляемые столбцы

На языке SQL возможно добавлять к итоговой выборке отдельные столбцы, значения которых будут вычисляться в процессе фильтрации. Для этого, к отбираемым столбцам после ключевого слова SELECT добавляется выражение, которое будет вычисляться для каждой строки.

Например, рассчитаем возраст всех студентов вуза и выведем его вместе с ФИО. Для этого добавим еще один столбец и укажем в нем функцию *age,* позволяющуювычислить разницу между двумя датами. В качестве точки отсчета выберем текущую дату, значение которой можно получить с помощью функции CURRENT\_DATE.

SELECT last\_name, first\_name, patronymic, age(CURRENT\_DATE, birthday)

FROM students;

last\_name | first\_name | patronymic | age

------------------+------------+----------------+--------------------------

Блюм | Андрей | Антонович | 18 years 5 mons 22 days

Верховенский | Петр | Степанович | 18 years 2 mons 8 days

Верховенский | Степан | Трофимович | 19 years 3 mons 5 days

Виргинская | Арина | Прохоровна | 19 years 7 mons 21 days

Гаганов | Артемий | Павлович | 19 years 5 mons

Гаганов | Павел | Павлович | 19 years 8 mons 3 days

Дроздов | Маврикий | Николаевич | 18 years 7 mons 8 days

Дроздова | Прасковья | Ивановна | 18 years 11 mons 6 days

Для того, чтобы переименовать столбец age, воспользуемся ключевым словом AS. После него укажем новое название столбца – «Возраст».

SELECT last\_name, first\_name, patronymic, age(CURRENT\_DATE, birthday) AS "Возраст"

FROM students;

last\_name | first\_name | patronymic | Возраст

------------------+------------+----------------+--------------------------

Блюм | Андрей | Антонович | 18 years 5 mons 22 days

Верховенский | Петр | Степанович | 18 years 2 mons 8 days

Верховенский | Степан | Трофимович | 19 years 3 mons 5 days

Виргинская | Арина | Прохоровна | 19 years 7 mons 21 days

Гаганов | Артемий | Павлович | 19 years 5 mons

Гаганов | Павел | Павлович | 19 years 8 mons 3 days

Дроздов | Маврикий | Николаевич | 18 years 7 mons 8 days

Дроздова | Прасковья | Ивановна | 18 years 11 mons 6 days

* 1. Агрегирование и группировка

Для проведения статистических вычислений в SQL существуют агрегатные функции. Данные функции принимают на вход множество значений и возвращают одно.

Основные агрегатные функции:

1. AVG: находит среднее значение
2. COUNT: находит количество строк в запросе
3. SUM: находит сумму значений
4. MIN: находит наименьшее значение
5. MAX: находит наибольшее значение

Например, в данном примере рассчитывается средний оклад всех преподавателей вуза. Т.к. атрибут «Оклад» (*salary*) имеет тип *money*, то для применения агрегатной функции необходимо привести его к числовому значению. Для этого используется конструкция ::numeric. Более подробно о приведении типов указано в п. 1.9.2.

**SELECT AVG(salary::numeric)::numeric(10,2) AS "Average salary"**

**FROM professors;**

Average salary

----------------

78214.29

(1 row)

В данном запросе рассчитывается общее число ставок, выделенное на весь вуз

SELECT SUM(wage\_rate) AS "Общее число ставок"

FROM employments;

Общее число ставок

--------------------

33.60

(1 row)

Однако, если потребуется рассчитать общее число ставок для каждого из направлений, то такой подход не сработает. Перед тем, как применять агрегатную функцию необходимо сгруппировать кортежи по определенному признаку – в данном примере, по номеру направления. Затем необходимо применить функцию SUM к каждой из получившихся групп.

Для группировки строк в SQL служит оператор GROUP BY. Данный оператор распределяет строки, образованные в результате запроса по группам, в которых значения некоторого столбца, по которому происходит группировка, являются одинаковыми. Группировку можно производить как по одному столбцу, так и по нескольким.

Дополним запрос из предыдущего примера. В нем происходит подсчет числа сотрудников в каждом структурном подразделении. Например в структурном подразделении 9 работают два сотрудника.

**SELECT structural\_unit\_id AS "Structural unit number", count(\*) AS "Number of employees"**

**FROM employments**

**GROUP BY structural\_unit\_id;**

Structural unit number | Number of employees

------------------------+---------------------

11 | 1

8 | 3

19 | 1

4 | 10

14 | 4

3 | 8

17 | 1

20 | 1

Рассмотрим еще один пример. В данном случае сумма будет рассчитана для каждого структурного подразделения. В подразделении 9 работают три преподавателя – поэтому их ставки суммировались и результатом стало значение 1.00.

SELECT structural\_unit\_id AS "Structural unit number", SUM(wage\_rate) AS "Wage rate sum"

FROM Employments

GROUP BY structural\_unit\_id;

Structural unit number | Wage rate sum

------------------------+---------------

11 | 0.35

8 | 1.75

19 | 0.35

4 | 4.25

14 | 1.40

3 | 3.65

17 | 0.35

20 | 0.35

Для фильтрации строк перед группировкой использовалось ключевое слово WHERE. В случае, если нужно отфильтровать строки после неё – используется ключевое слово HAVING.

Добавим в приведенный выше пример условие, чтобы число сотрудников выводилось только для подразделений, в которых более 2х преподавателей.

SELECT structural\_unit\_id AS "Structural unit number", count(\*) AS "Number of employees"

FROM employments

GROUP BY structural\_unit\_id

HAVING count(\*) > 2;

Structural unit number | Number of employees

------------------------+---------------------

8 | 3

4 | 10

14 | 4

3 | 8

1 | 19

5 | 5

2 | 6

6 | 3

* 1. Сортировка

Для сортировки результата запроса необходимо использовать ключевое слово ORDER BY. После него указывается атрибуты, по которым производится сортировка. Далее указывается порядок с помощью слов DESC (в порядке убывания) и ASC (в порядке возрастания). По умолчанию строки сортируются по возрастанию, поэтому ASC можно опускать.

SELECT structural\_unit\_id AS "Номер структурного подразделения", count(\*) AS "Число сотрудников"

FROM employments

GROUP BY structural\_unit\_id

ORDER BY structural\_unit\_id;

Номер структурного подразделения | Число сотрудников

----------------------------------+-------------------

1 | 19

2 | 6

3 | 8

4 | 10

5 | 5

6 | 3

7 | 1

8 | 3

SELECT structural\_unit\_id AS "Номер структурного подразделения", count(\*) AS "Число сотрудников"

FROM employments

GROUP BY structural\_unit\_id

ORDER BY structural\_unit\_id DESC;

Номер структурного подразделения | Число сотрудников

----------------------------------+-------------------

20 | 1

19 | 1

18 | 1

17 | 1

16 | 1

15 | 1

14 | 4

13 | 1

* 1. Оконные функции

При составлении запросов с использованием агрегатных функций применялась группировка. Все строки с одинаковыми значениями, указанными после GROUP BY объединялись в одну и над каждой из данных групп совершалось определенное действие. Таким образом, число строк в результирующей таблице уменьшалось. Однако в некоторых случаях бывает полезным провести вычисления над группой и добавить вычисленное значение в качестве дополнительного столбца для каждой строки таблицы. Например, необходимо вывести студентов всех групп и пронумеровать в алфавитном порядке внутри каждой группы. Для этого можно воспользоваться оконными функциями.

В общем виде оконные функции выглядят следующим образом:

SELECT

Название функции (столбец для вычислений)

OVER (

PARTITION BY столбец для группировки

ORDER BY столбец для сортировки

)

В качестве функции может выступать одна из агрегатных функций (SUM, COUNT, AVG, MIN, MAX) или специальные функции, предназначенные для оконных вычислений.

Приведем некоторые из них:

**ROW\_NUMBER** – данная функция возвращает номер строки и используется для нумерации;

**FIRST\_VALUE** или **LAST\_VALUE** — с помощью функции можно получить первое и последнее значение в окне. В качестве параметра принимает столбец, значение которого необходимо вернуть;

**CUME\_DIST** — вычисляет интегральное распределение (относительное положение) значений в окне;

После ключевых слов PARTITION BY необходимо указать поле, по которому будет производиться объединение в группы. Далее возможно отсортировать значения внутри каждой из групп. [3]

В итоге запрос для вычисления порядкового номера студента будет выглядеть следующим образом:

**SELECT row\_number() OVER (**

**partition by students\_group\_number ORDER BY last\_name),**

**last\_name, first\_name, patronymic, students\_group\_number**

**FROM students;**

row\_number | last\_name | first\_name | patronymic | students\_group\_number

------------+------------------+------------+----------------+-----------------------

1 | Беляков | Иван | Константинович | ИБ-11

2 | Быков | Лев | Михайлович | ИБ-11

3 | Виноградов | Иван | Макарович | ИБ-11

4 | Волков | Кирилл | Денисович | ИБ-11

5 | Григорьев | Егор | Матвеевич | ИБ-11

6 | Губанов | Артём | Андреевич | ИБ-11

7 | Денисов | Леонид | Маркович | ИБ-11

8 | Еремин | Лука | Львович | ИБ-11

* 1. Несколько важных замечаний
     1. Экранирование кавычек

При использовании текстовых строк в запросах их необходимо обрамлять одинарными кавычками. Как, например, в следующем запросе:

SELECT last\_name, first\_name

FROM students

WHERE first\_name = 'Анна'

Однако, как поступить, в случае наличия в самой строке символа кавычки. Например, в институт поступил иностранный студент с фамилией O’Brien.

SELECT last\_name, first\_name

FROM students

WHERE last\_name = 'O'Brien'

Данный запрос выполнен не будет, т.к. СУБД распознает как строку только символ 'O'. Чтобы одинарная кавычка воспринималась как часть строки, а не указание на её окончание, в PostgreSQL предусмотрена возможность дублирования данной кавычки.

SELECT last\_name, first\_name

FROM students

WHERE last\_name = 'O''Brien'

Способ, указанный выше, сработает.

* + 1. Типы данных в PostgreSQL

PostgreSQL предоставляет пользователям большой выбор встроенных типов данных. В таблице 1 приведены основные из них. Более подробно можно прочитать в Главе 8 документации PostgreSQL. [1]

Таблица 1 Типы данных PostgreSQL

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| bigint | знаковое целое из 8 байт |
| boolean | логическое значение (true/false) |
| date | календарная дата (год, месяц, день) |
| integer | знаковое четырёхбайтное целое |
| json | текстовые данные JSON |
| money | денежная сумма |
| numeric [ (*p*, *s*) ] | вещественное число заданной точности |
| real | число одинарной точности с плавающей точкой (4 байта) |
| smallint | знаковое двухбайтное целое |
| serial | четырёхбайтное целое с автоувеличением |
| text | символьная строка переменной длины |
| time [ (*p*) ] [ without time zone ] | время суток (без часового пояса) |
| uuid | универсальный уникальный идентификатор |
| varchar(n) | строка ограниченной переменной длины |

К основным целым числовым типам можно отнести типы smallint(2 байта), integer (4 байта), bigint (8 байт). В большинстве случаем рекомендуется использовать тип integer.

Для хранения дробных чисел произвольной точности возможно использовать тип numeric.

NUMERIC(точность, масштаб)

где точность – число значащих цифр (по обе стороны запятой), масштаб – число цифр после запятой. По умолчанию, тип *numeric* может хранить числа, содержащие до 1000 значащих цифр.

Для хранения чисел с плавающей точкой используются типы real и double. Обратим внимание, что при арифметике чисел с плавающей точкой возможны ошибки округления и неточности при хранении данных, поэтому при необходимости точности вычислений рекомендуется использовать тип *numeric.* Данная проблема может стать критичной при хранении денежных сумм. Поэтому в случае хранения денежных единиц рекомендуется использовать тип numeric или money.

Для хранения строковых типов ограниченного размера рекомендуется использовать тип *varchar*(N), где N – максимальный размер строки. При необходимости хранения длинного текста возможно применять тип *text*.

Отдельно остановимся на типе, хранящим значения даты и времени. В PostgreSQL существует множество подобных типов. Наиболее популярными являются типы *date* и *time*. Все даты в PostgreSQL считаются по Григорианскому календарю, даже для времени до его введения.

Данные о дате могут храниться разными способами. В соответствии со стандартом ISO 8601, рекомендуется хранить дату в формате «год-месяц-день». Например, 1998-08-26. Также возможно использовать следующую запись: месяц/день/год в режиме MDY или день/месяц/год в режиме DMY. Для проверки, в каком режиме работает PostgreSQL возможно выполнить команду:

SELECT current\_setting('datestyle');

current\_setting

-----------------

ISO, DMY

(1 row)

Для переключения между режимами возможно использовать команду:

SET datestyle TO "ISO, DMY";

Для работы со значениями типа дата и время существует набор функций, встроенных в PostgreSQL. Приведем некоторые из них.

|  |  |  |
| --- | --- | --- |
| Функция | Описание | Пример |
| extract(*field* from timestamp) | Извлечение полей из даты | SELECT extract(month from '2024-02-23'::date) |
| age(timestamp, [timestamp]) | Вычисление временного интервала между датами. При указании одного аргумента, вычисляется интервал между ним и текущей датой. | SELECT age('2024-02-22'::date, '2024-01-22'::date) |

В качестве первичного ключа очень часто используют последовательность из чисел от 1 и выше. Для хранения подобной последовательности используется тип *serial*. При создании переменной данного типа создается автоинкрементирующийся счетчик, увеличивающийся с добавлением новой записи. При вызове команды INSERT поле с данным типом возможно отметить значением DEFAULT или просто оставить пустым.

В некоторых случаях необходимо, чтобы первичным ключом служил уникальный идентификатор, не повторяющийся ни в одной другой базе данных. Это может быть полезно, при объединении двух таблиц из различных баз данных. Таким идентификатором может служить GUID - глобальный уникальный идентификатор. Для его хранения в PostgreSQL используется тип *uuid*. UUID записывается в виде последовательности шестнадцатеричных цифр, разделённых знаками минуса на несколько групп: группа из 8 цифр, три группы из 4 цифр, группа из 12 цифр, что в сумме составляет 32 цифры и представляет 128 бит.

Пример *uuid*: 47a8229b-9e8a-0473-fd20-21c889da75bf

PostrgeSQL позволяет хранить данные в одной ячейке в виде массива. Для его объявления необходимо после ключевого слова, обозначающего тип элемента добавить квадратные скобки. Например, *int[]*. Размерность массива указывается в виде целого числа внутри скобок. Для объявления одномерных массивов возможно использовать ключевое слово ARRAY.

SELECT ARRAY[1,2,3];

array

---------

{1,2,3}

(1 row)

Обращение к элементам массива происходит с помощью оператора [].

* + 1. Преобразование типов

Преобразование типов в PostgreSQL возможно выполнить несколькими способами. Самый простой из них – напрямую указать тип данных через символ “::”

SELECT 1234::int;

int4

------

1234

(1 row)

SELECT 1234::text;

text

------

1234

(1 row)

Аналогичного результата можно добиться, используя оператор CAST

SELECT CAST(1234 AS int);

int4

------

1234

(1 row)

SELECT CAST(1234 AS text);

text

------

1234

(1 row)

* + 1. Оператор CASE

В SQL возможно создавать конструкции, подобные SWITCH – CASE из языка программирования С. Общий синтаксис выглядит следующим образом:

CASE WHEN *условие* THEN *результат*

[WHEN ...]

[ELSE *результат*]

END

Если условие оказывается верным, то значением выражения CASE становится результат, идущий после ключевого слова THEN. Если не выполняется ни одно из указанных условий, то результатом выражения будет значение после ветви ELSE или NULL при её отсутствии.

Рассмотрим пример, в котором каждой оценке сопоставляется словесная запись.

SELECT mark, CASE

WHEN mark=5 THEN 'Отлично'

WHEN mark=4 THEN 'Хорошо'

WHEN mark=3 THEN 'Удовлетворительно'

WHEN mark=2 THEN 'Неудовлетворительно'

END

FROM field\_comprehensions

mark | case

------+---------------------

3 | Удовлетворительно

2 | Неудовлетворительно

5 | Отлично

4 | Хорошо

* + 1. Некоторые полезные функции

Для выполнения практических заданий лабораторных работ вам могут потребоваться некоторые функции. Приведем их в виде таблицы.

|  |  |  |
| --- | --- | --- |
| **Функция** | **Описание** | **Пример** |
| overlay(*string* placing *string* from int [for int]) | Заменяет подстроку | SELECT overlay('Беляков' placing 'улан' from 2 for 4);  overlay  ---------  Буланов  (1 row) |
| char\_length(string) | Число символов в строке | SELECT char\_length('Базы данных');  char\_length  -------------  11  (1 row) |
| substring(string [from int] [for int]) | Извлекает подстроку | SELECT substring('Базы данных' from 1 for 4);  substring  -----------  Базы  (1 row) |
| position(substring in string) | Положение указанной подстроки | SELECT position('данных' in 'Базы данных');  position  ----------  6  (1 row) |
| left(str text, n int) | Возвращает первые n символов в строке. | SELECT left('Базы данных', 4);  left  ------  Базы  (1 row) |
| right(str text, n int) | Возвращает последние n символов в строке. | SELECT right('Базы данных', 6);  right  --------  данных  (1 row) |
| mod(y int, x int) | Остаток от деления y/x | SELECT mod(9,4);  mod  -----  1  (1 row) |
| round(x numeric) | Округление до ближайшего числа | SELECT round(9.5);  round  -------  10  (1 row) |
| random() | Генерация случайного числа в диапазоне 0.0 <= x < 1.0 | SELECT random();  random  --------------------  0.9942907746544045  (1 row) |

1. Практическая часть

Вариант к практической части выбирается по формуле: V = (N % 10) +1, где N – номер в списке группы, % - остаток от деления.

* 1. Задание 1

*Исследование типов данных*

Предположим, что в магазине новый конструктор стоит 999 рублей и 99 копеек. Студент С. решил приобрести для дальнейшей перепродажи 100000 таких товаров. Для расчета общей суммы, которую необходимо заплатить был создан следующий скрипт на языке PL/pgSQL. Более подробно о нём будет рассказано в одной из следующих лабораторных работ. Обратите внимание, что значение суммы имеет тип *real*.

DO

$$

DECLARE

summ real :=0.0;

BEGIN

FOR i IN 1..100000 LOOP

summ := summ + 999.99;

END LOOP;

RAISE NOTICE 'Summ = %;', summ;

--RAISE NOTICE 'Diff = %;', 99999000.00 - summ;

END

$$ language plpgsql;

Запустите скрипт. Раскомментируйте строку с вычислением разницы и определите, сколько денег переплатил студент С? Объясните полученный результат. Измените тип суммы на *numeric* и *money*. Какой результат был получен в обоих случаях?

* 1. Задание 2

*Написание запросов на языке SQL*

Напишите SQL запросы к учебной базе данных в соответствии с вариантом. Запросы брать из сборник запросов к учебной базе данных, расположенного ниже

|  |  |
| --- | --- |
| № варианта | № запросов |
| 1 | 1, 11, 21, 31, 41, 51, 61, 71 |
| 2 | 2, 12, 22, 32, 42, 52, 62, 72 |
| 3 | 3, 13, 23, 33, 43, 53, 63, 73 |
| 4 | 4, 14, 24, 34, 44, 54, 64, 74 |
| 5 | 5, 15, 25, 35, 45, 55, 65, 75 |
| 6 | 6, 16, 26, 36, 46, 56, 66, 76 |
| 7 | 7, 17, 27, 37, 47, 57, 67, 77 |
| 8 | 8, 18, 28, 38, 48, 58, 68, 78 |
| 9 | 9, 19, 29, 39, 49, 59, 69, 79 |
| 10 | 10, 20, 30, 40, 50, 60, 70, 80 |

* 1. Задание 3

Самостоятельно разработайте **7 осмысленных** запросов к базе данных, используя приведенные в данной лабораторной работе материалы.

Сборник запросов к учебной базе данных

Условия *WHERE, ORDER BY*

1. Вывести всех студентов группы отсортировав по возрасту
2. Вывести возраст студентов группы, отсортировав по номеру студенческого билета
3. Вывести самого младшего студента группы ИВТ-41
4. Вывести самого старшего студента группы ИВТ-42
5. Выведите студентов, у которых дата рождения совпадает с вашей (месяц и день)
6. Вывести почту всех Кириллов, отсортировав их по дате рождения
7. Выведите студентов, которые моложе 20 лет
8. Вывести номер студенческого билета всех Евгениев, отсортировав их по возрасту
9. Вывести почту всех студентов группы ИТД-21, отсортировав их по фамилии обучающихся
10. Вывести фамилии всех студентов, чей студенческий лежит в интервале от 820000 до 830000
11. Вывести список всех предметов, отсортировав по уникальному Id
12. Вывести должности всех преподавателей, чей оклад выше 100000, отсортировать по размеру оклада
13. Вывести всех студентов, родившихся зимой и весной
14. Вывести всех студентов, родившихся летом и осенью
15. Вывести всех студентов, родившихся под летними знаками зодиака
16. Вывести всех студентов, родившихся под осенними знаками зодиака
17. Выведите все группы, обучающиеся очно
18. Выведите все группы, обучающиеся заочно
19. Вывести номера студенческих билетов студентов, имеющих двойки.
20. Вывести фамилию и академические звание преподавателя с максимальным стажем

*Группировка GROUP BY*

1. Подсчитать количество двоечников, троечников, хорошистов и отличников среди студентов.
2. Выведите количество студентов обучающихся в каждой группе и отсортируйте их по количеству
3. Выведите количество студентов, которые обучаются на третьем курсе.
4. Выведите коды дисциплин, по которым зачёт получило больше 100 студентов
5. Вывести средний оклад преподавателей по должностям в порядке возрастания, включить в расчет только тех у кого оклад больше 40000
6. Посчитать число преподавателей, имеющих разные ученые степени, отсортировать их по количеству, оставив только те строчки, где число больше 2
7. Найти среднюю оценку каждого студента по всем дисциплинам, отсортировать записи по оценке и дисциплине, оставив только тех, у кого средняя оценка больше 4
8. Вывести неуникальные фамилии и их количество для групп ИБ, отсортировать по алфавиту. Рядом вывести количество таких фамилий, оставив только тех, которых больше 2
9. Вывести список студентов (номера студенческих билетов) и число их оценок больших, чем 4, отсортировать по числу оценок в порядке убывания. Оставить только тех, у кого число таких оценок больше 5.
10. Подсчитать количество долгов у студентов, вывести номер студенческого билета и количество долгов по всем дисциплинам, отсортировав их по номеру студенческого. В вывод включить только тех, у кого долгов больше 10.
11. Подсчитать количество должников по каждой дисциплине и вывести на экран код дисциплины и количество должников, превышающих 50 человек.
12. Вывести число учащихся по каждой дисциплине, отсортировать по количеству. Оставить только те числа, где число учащихся больше 100
13. Вывести среднюю оценку учащихся по каждой дисциплине, отсортировать по ней же. Оставить только средние оценки больше 3.5
14. Вывести количество отличных оценок у каждого студента, отсортировать по количеству. Оставить только тех, у кого пятерок больше 10
15. Вывести стаж преподавателя, среднюю зарплату для каждого стажа и количество преподавателей, имеющих этот стаж. При подсчете учитывать только преподавателей, имеющих стаж больше 2 лет и вывести записи, у которых число преподавателей, имеющих данный стаж больше 1
16. Вывести дату рождения (только число) студентов и подсчитать количество студентов, родившихся в один день, отсортировать по возрастанию числа. Оставить только те дни, в которые родилось меньше 15 студентов.
17. Подсчитать количество различных отчеств у девушек студенток, вывести только те записи, где количество отчеств > 1 и отсортировать их в порядке возрастания количества
18. Вывести коды структурных подразделений и количество работающих в них преподавателей, оставив только те в которых больше 1 сотрудника. Отсортировать по количеству.
19. Подсчитать количество каждой оценки у каждого студента, отсортировать по номеру студенческого и оценке. Вывести номер студенческого билета, оценку и ее количество, оставив билеты из диапазона 820000–840000.
20. Вывести номера студенческих билетов, среднюю оценку студента по всем дисциплинам и количество оценок. В подсчет включить только студентов с четными номерами студенческих., заполненными полями оценок и вывести записи, у которых средняя оценка лежит в диапазоне 3,2 – 3,6. Среднюю оценку округлить до сотых.

*Регулярные выражения*

1. Вывести весь 3-й курс ИБ, отсортировать по возрасту
2. Вывести весь 2-й курс ИВТ, отсортировать по фамилии
3. Вывести весь 4-й курс ИТД, отсортировать по фамилии, имени и отчеству
4. Вывести студентов с почтой, начинающейся на латинскую букву A, отсортировать по имени студентов
5. Вывести студентов, у которых ВНУТРИ имени почты содержится латинская букву A, отсортировать по фамилиям в обратном порядке
6. Вывести всех студентов, название почты которых начинается с букв из диапазона A-G, отсортировать по студенческому билету
7. Вывести всех студентов ИБ с именем Андрей
8. Вывести всех студентов ИВТ чьи фамилии начинаются на буквы из диапазона А-Г, отсортировать по фамилии
9. Вывести всех студентов ИБ с отчеством, заканчивающимся на «нович» или «ловна»
10. Вывести всех студентов, у которых номер студенческого начинается на 8 и заканчивается на 1, отсортировать по нему же
11. Вывести всех студентов, родившихся зимой, используя регулярные выражения
12. Вывести всех студентов ИТД с фамилией, заканчивающейся на -ин/ов/ев, отсортировать по фамилии
13. Вывести всех студентов вечерних отделений
14. Выведите всех студентов, обучающихся на 2 курсе, исключая номер группы
15. Вывести всех студентов третьего курса, исключая группы ИБ в порядке убывания номеров студенческих билетов
16. Выведите количество по группам студентов из 3 курса, чьи фамилии не заканчиваются на ‘а’. Вывести группу и количество.
17. Была потеряна контрольная работа студента с инициалами В.Ф. Необходимо определить фамилию, группу, в которой обучается студент и вернуть работу.
18. Выведите количество групп в каждом подразделении с очной формой обучения, исключая 4 подразделение.
19. Посчитать количество групп 4-го курса
20. Посчитать всех Михайловичей, отсортировать по фамилиям в алфавитном порядке. Оставить только тех, у кого в имени есть буква "А" или "а"

*Общее*

1. Выведите фамилии и количество их повторений, которые начинаются на ту же

букву, что и ваша фамилия, а две последние буквы фамилии с вашей не совпадают.

1. Вывести всех студентов-тёзок, родившихся в 2004-2005 годах
2. Подсчитать количество групп в каждом структурном подразделении, столбцы назвать "код подразделения” и "количество” соответственно. Исключить из подсчёта вторые группы (оканчивающиеся на 2), отсортировать по коду подразделения.
3. Вывести общую сумму оклада по должностям, преподавателей чей стаж выше трех лет, округлив ее до сотых. Сумма превышающая 500т не выводится.
4. Подсчитать и вывести на экран количество студенческих билетов для каждой даты выдачи. Дайте столбцам русские наименования. Исключите из подсчета билеты заканчивающиеся на ‘3’.
5. Вывести фамилии и группы всех студентов второго и четвертого курса, исключая вечерников. Дать столбцам русские названия. Отсортировать по фамилиям в обратном порядке.
6. Выведите количество студентов девушек, которые обучаются на третьем курсе.
7. Подсчитать количество студентов обучающихся в каждой группе. В подсчет включить только студентов третьего и четвертого курсов. Вывести только те группы, у которых количество студентов> 20, дать столбцам русские названия. Отсортировать по количеству.
8. SQL запрос показывает сколько студентов получили определенную оценку за все технические дисциплины, не считая те случаи, когда оценку всего получили меньше 10 человек.
9. Найти все дисциплины, которые ведут преподаватели с кодами: 810006, 840010, 805004, 840004. Вывести код подразделения, код преподавателя, название дисциплины. Отсортировать по коду подразделения и коду преподавателя. Всем столбцам дать русские имена.
10. Подсчитать количество студентов, родившихся раньше '01/01/2004' в каждой группе, вывести названия и количество студентов тех групп, в которых таких студентов больше 20. Отсортировать по количеству. Всем столбцам дать русские имена.
11. Подсчитать количество студентов в каждой группе, при подсчете учитывать только первые группы, включая вечерников. Вывести группы, в которых количество студентов >20, а также число студентов по группам. Отсортировать по количеству. Всем столбцам дать русские имена.
12. Подсчитать количество студентов в каждой группе имя которых состоит из пяти символов, вывести только те группы, у которых количество таких студентов больше одного. Отсортировать по количеству. Всем столбцам дать русские имена.
13. Вывести всех студентов всех курсов второй группы ИТД, фамилии которых начинаются на букву из диапазона А-Н, убрать повторяющиеся фамилии, отсортировать по фамилии. Всем столбцам дать русские имена.
14. Вывести средний оклад преподавателей по всем должностям, кроме доцента. B подсчёте учитывать только тех у кого оклад > 15000. Отсортировать по окладу. Всем столбцам дать русские имена. Средний оклад округлить до сотых.
15. Подсчитать количество дисциплин, которые ведет каждый преподаватель, в подсчете учитывать только тех, чей код лежит в диапазоне 810000-850000. Вывести коды преподавателей, которые читают больше одной дисциплины. Отсортировать по количеству. Всем столбцам дать русские имена.
16. Вывести пять самых популярных отчеств у студентов и их количество, Значение null в подсчёт не включать.
17. Подсчитать количеств однофамильцев в каждой группе. Вывести фамилию, номер группы и количество таких фамилий. Уникальные фамилии не выводить. Отсортировать по фамилии и группе. Всем столбцам дать русские имена
18. Найти самого младшего студента. у которого нет отчества, и фамилия состоит из шести символов. Вывести всю возможную информацию о нем, в конце добавить столбец с возрастом.
19. Найти студентов, родившихся в високосном году и вывести только тех, у кого на текущий момент прошло меньше месяца со дня рождения. В вывод включить всю информацию о таких студентах и добавить столбец с возрастом. Отсортировать записи по возрасту студента.

Контрольные вопросы

1. Как задать условие фильтрации на языке SQL?
2. Сколькими способами возможно произвести поиск с помощью шаблона?
3. Для чего предназначены оконные функции?
4. Чем тип *real* отличается от типа *numeric*?
5. Для чего предназначены *uuid*?

Список использованной литературы

|  |  |
| --- | --- |
| [1] | Документация к PostgreSQL 15.1, 2022. |
| [2] | «Regular-expressions,» [В Интернете]. Available: https://www.regular-expressions.info/tutorial.html. [Дата обращения: 23 06 2023]. |
| [3] | Р. Романчук, «Учимся применять оконные функции,» [В Интернете]. Available: http://thisisdata.ru/blog/uchimsya-primenyat-okonnyye-funktsii/. [Дата обращения: 29 06 2023]. |
| [4] | «Исходный код СУБД postgres,» [В Интернете]. Available: https://github.com/postgres/postgres. [Дата обращения: 30 01 2023]. |
| [5] | Е. Рогов, PostgreSQL изнутри, 1-е ред., Москва: ДМК Пресс, 2023, p. 662 . |
| [6] | Б. А. Новиков, Е. А. Горшкова и Н. Г. Графеева, Основы технологии баз данных, 2-е ред., Москва: ДМК пресс, 2020, p. 582. |
| [7] | Е. П. Моргунов, PostgreSQL. Основы языка SQL, 1-е ред., Санкт-Петербург: БХВ-Петербург, 2018, p. 336. |

1. В результате данного запроса должно быть выведено 163 строчки. Для экономии места, в данном лабораторном практикуме будет оставлено не более 8 строк от результата любого запроса [↑](#footnote-ref-1)