Лабораторные работы по курсу
Базы данных

Лабораторная работа 2
«Оператор SELECT языка SQL»

Москва, 2023

Оглавление

Теоретические сведения

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 SELECT получает строки из базы данных и возвращает их в виде таблицы результатов запроса. Результат выполнения запроса может оказаться пустым множеством.

Рассмотрим выполнение SQL запроса на фильтрацию данных по шагам.

1.1.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.

```
SELECT * FROM "Студент";
```

	Номер студенческого билета [PK] bigint	Фамилия character varying (30)	Имя character varying (30)	Отчество character varying (30)	Номер группы character varying (7)	Дата рождения date	Почта character varying (30)
1	841576	Дроздов	Марк	Артёмович	ИТД-33	2002-11-23	DrozdovMark@miet.ru
2	842676	Самсонов	Максим	Андреевич	ИТД-33	2002-11-02	SamsonovMaksim@...
3	835406	Смирнов	Ярослав	Владиславович	ИТД-33	2002-04-03	SmirnovJaroslav@mi...
4	882599	Соловьев	Сергей	Владимирович	ИТД-33	2002-08-02	SolovevSergej@miet.ru
5	820080	Селиванов	Дмитрий	Маркович	ИТД-33	2002-06-07	SelivanovDmitrij@mie...
6	815406	Егоров	Артём	Юрьевич	ИТД-33	2002-11-09	EgorovArtem@miet.ru
7	846147	Грачев	Андрей	Всеволодович	ИТД-33	2002-08-21	GrachevAndrej@miet....
8	883720	Маслов	Михаил	Дмитриевич	ИТД-33	2002-06-21	MaslovMihail@miet.ru

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

```
SELECT "Фамилия", "Имя", "Отчество"
FROM "Студент";
```

	Фамилия character varying (30)	Имя character varying (30)	Отчество character varying (30)
1	Дроздов	Марк	Артёмович
2	Самсонов	Максим	Андреевич
3	Смирнов	Ярослав	Владиславович
4	Соловьев	Сергей	Владимирович
5	Селиванов	Дмитрий	Маркович
6	Егоров	Артём	Юрьевич
7	Грачев	Андрей	Всеволодович
8	Маслов	Михаил	Дмитриевич

1.2.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 "Фамилия", "Имя", "Отчество"
FROM "Студент"
WHERE "Имя" = 'Сергей';
```

	Фамилия character varying (30)	Имя character varying (30)	Отчество character varying (30)
1	Соловьев	Сергей	Владимирович
2	Дорофеев	Сергей	Евгеньевич

Обратите внимание, что сравниваемая строка заключается в одинарные кавычки.
Аналогично можно найти всех студентов, у которых номер студенческого билета больше числа 850000.

```
SELECT "Фамилия", "Имя", "Отчество"
FROM "Студент"
WHERE "Номер студенческого билета" > 850000;
```

	Фамилия character varying (30)	Имя character varying (30)	Отчество character varying (30)
1	Соловьев	Сергей	Владимирович
2	Маслов	Михаил	Дмитриевич
3	Кондратьева	Варвара	Никитична
4	Коровина	Мария	Георгиевна
5	Князева	Есения	Александровна
6	Родионова	Арина	Борисовна
7	Андреева	Ева	Кирилловна
8	Королева	Ксения	Артёмовна

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 "Фамилия", "Имя", "Отчество"
FROM "Студент"
WHERE "Дата рождения">'25/06/2002' AND "Дата рождения"<'25/06/2003'
```

	Фамилия character varying (30)	Имя character varying (30)	Отчество character varying (30)
1	Дроздов	Марк	Артёмович
2	Самсонов	Максим	Андреевич
3	Соловьев	Сергей	Владимирович
4	Егоров	Артём	Юрьевич
5	Грачев	Андрей	Всеволодович
6	Кондратьева	Варвара	Никитична
7	Князева	Маргарита	Арсентьевна
8	Емельянова	Екатерина	Платоновна

Аналогичный запрос можно составить, используя ключевое слово BETWEEN.

```
SELECT "Фамилия", "Имя", "Отчество"
FROM "Студент"
WHERE "Дата рождения" BETWEEN '25/06/2002' AND '25/06/2003'
```

1.3.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 23 ноября 2002 года или 09 февраля 2003 года.

```
SELECT "Фамилия", "Имя", "Отчество"
FROM "Студент"
WHERE "Дата рождения" IN ('23/11/2002' , '09/02/2003')
```

	Фамилия character varying (30)	Имя character varying (30)	Отчество character varying (30)
1	Дроздов	Марк	Артёмович
2	Худяков	Андрей	Львович

1.4.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 "Фамилия", "Имя", "Отчество"
FROM "Студент"
WHERE "Имя" LIKE 'А%';
```

	Фамилия character varying (30)	Имя character varying (30)	Отчество character varying (30)
1	Егоров	Артём	Юрьевич
2	Грачев	Андрей	Всеволодович
3	Родионова	Арина	Борисовна
4	Александрова	Агата	Львовна
5	Кузнецова	Алиса	Александровна
6	Богданов	Артур	Тимофеевич
7	Гаврилов	Александр	Всеволодович
8	Самойлов	Артём	Матвеевич

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

```
SELECT "Фамилия", "Имя", "Отчество"
FROM "Студент"
WHERE "Имя" LIKE '____я';
```

Рассмотрим пример с регулярными выражениями

```
SELECT "Фамилия", "Имя", "Отчество"
FROM "Студент"
WHERE "Фамилия" ~ '^[ЛМН]';
```

	Фамилия character varying (30)	Имя character varying (30)	Отчество character varying (30)
1	Маслов	Михаил	Дмитриевич
2	Новиков	Кирилл	Никитич
3	Маркин	Даниэль	Ильич
4	Молчанов	Александр	Артемьевич
5	Лобанова	Виктория	Артёмовна
6	Мухин	Ростислав	Янович
7	Любимов	Кирилл	Андреевич
8	Лебедев	Дмитрий	Марсельевич

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

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

```
SELECT "Фамилия", "Имя", "Отчество"
FROM "Студент"
WHERE "Фамилия" ~ '^[ЛМН].*ин$';
```

	Фамилия character varying (30)	Имя character varying (30)	Отчество character varying (30)
1	Маркин	Даниэль	Ильич
2	Мухин	Ростислав	Янович

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 "Фамилия", "Имя", "Отчество"
FROM "Студент"
WHERE "Фамилия" ~ '^[ЛМН].*ин';
```

	Фамилия character varying (30)	Имя character varying (30)	Отчество character varying (30)
1	Маркин	Даниэль	Ильич
2	Мухин	Ростислав	Янович
3	Литвинов	Георгий	Артёмович
4	Литвинов	Иван	Александрович

1.5.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,* позволяющую вычислить разницу между двумя датами. В качестве точки отсчета выберем текущую дату, значение которой можно получить с помощью функции CURRENT_DATE.

```
SELECT "Фамилия", "Имя", "Отчество", age(CURRENT_DATE, "Дата рождения")
FROM "Студент"
```

	Фамилия character varying (30)	Имя character varying (30)	Отчество character varying (30)	age interval
1	Дроздов	Марк	Артёмович	20 years ...
2	Самсонов	Максим	Андреевич	20 years ...
3	Смирнов	Ярослав	Владиславович	20 years ...
4	Соловьев	Сергей	Владимирович	20 years ...
5	Селиванов	Дмитрий	Маркович	20 years ...
6	Егоров	Артём	Юрьевич	20 years ...
7	Грачев	Андрей	Всеволодович	20 years ...
8	Маслов	Михаил	Дмитриевич	20 years ...

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

```
SELECT "Фамилия", "Имя", "Отчество", age(CURRENT_DATE, "Дата рождения") AS
"Возраст"
FROM "Студент"
```

	Фамилия character varying (30)	Имя character varying (30)	Отчество character varying (30)	Возраст interval
1	Дроздов	Марк	Артёмович	20 years ...
2	Самсонов	Максим	Андреевич	20 years ...
3	Смирнов	Ярослав	Владиславович	20 years ...
4	Соловьев	Сергей	Владимирович	20 years ...
5	Селиванов	Дмитрий	Маркович	20 years ...
6	Егоров	Артём	Юрьевич	20 years ...
7	Грачев	Андрей	Всеволодович	20 years ...
8	Маслов	Михаил	Дмитриевич	20 years ...

1.6. 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» имеет тип *money*, то для применения агрегатной функции необходимо привести его к числовому значению. Для этого используется конструкция ::numeric.

```
SELECT AVG("Оклад"::numeric) AS "Средний оклад"
FROM "Преподаватель"
```

	Средний оклад numeric
1	56833.3333333333

```
SELECT SUM("Ставка") AS "Общее число ставок"
FROM "Трудоустройство";
```

	Общее число ставок numeric
1	2.25

Однако, если потребуется рассчитать общее число ставок для каждого из направлений, то такой подход не сработает. Перед тем, как применять агрегирующую 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 данном случае сумма будет рассчитана для каждого структурного подразделения. В подразделении 1 работают три преподавателя – поэтому их ставки просуммировались и результатом стало значение 1.00.

```
SELECT "Номер структурного подразделения", SUM("Ставка") AS "Общее число
ставок"
FROM "Трудоустройство"
GROUP BY "Номер структурного подразделения";
```

	Номер структурного подразделения bigint	Общее число ставок numeric
1	3	0.60
2	4	0.40
3	2	0.25
4	1	1.00

Рассмотрим еще один пример. В нем происходит подсчет числа сотрудников в каждом структурном подразделении.

```
SELECT "Номер структурного подразделения", count(*) AS "Число сотрудников"
FROM "Трудоустройство"
GROUP BY "Номер структурного подразделения";
```

	Номер структурного подразделения bigint	Число сотрудников bigint
1	3	1
2	4	1
3	2	1
4	1	3

При выполнении запросов, с помощью ключевого слова WHERE возможно отобрать только те строки, которые удовлетворяют указанному условию. В случае группировки возможно аналогично включать в каждую из групп не все строки, а только определенные. Для этого используется ключевое слово HAVING.

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

```
SELECT "Номер структурного подразделения", count(*) AS "Число сотрудников"
FROM "Трудоустройство"
GROUP BY "Номер структурного подразделения"
HAVING count(*) > 2;
```

	Номер структурного подразделения bigint	Число сотрудников bigint
1	1	3

1.7.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 "Номер структурного подразделения", count(*) AS "Число сотрудников"
FROM "Трудоустройство"
GROUP BY "Номер структурного подразделения"
ORDER BY "Номер структурного подразделения"
```

	Номер структурного подразделения bigint	Число сотрудников bigint
1	1	3
2	2	1
3	3	1
4	4	1

```
SELECT "Номер структурного подразделения", count(*) AS "Число сотрудников"
FROM "Трудоустройство"
GROUP BY "Номер структурного подразделения"
ORDER BY "Номер структурного подразделения" DESC
```

	Номер структурного подразделения bigint	Число сотрудников bigint
1	4	1
2	3	1
3	2	1
4	1	3

Лабораторное задание

Разработайте 5 **осмысленных** запросов к базе данных, используя приведенные в данной лабораторной работе материалы. Вариант выбирается в соответствии с номером по списку.

	Вариант 1					Вариант 2					Вариант 3					Вариант 4				
Ключевое слово	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
WHERE	+	+			+	+	+	+	+	+	+	+	+		+	+	+		+	+
GROUP BY			+	+	+			+	+	+			+	+	+			+	+	+
HAVING				+	+					+				+	+					+
ORDER BY			+		+	+			+	+		+			+	+			+	+
AS	+				+			+		+	+				+			+		+
LIKE					+		+								+		+			
Агрегатные функции			+							+				+				+		
Регулярные выражения		+						+				+					+			

Список литературы

[1] Документация к PostgreSQL 15.1, 2022.

[2] «Исходный код СУБД postgres,» [В Интернете]. Available: https://github.com/postgres/postgres. [Дата обращения: 30 01 2023].

[3] Е. Рогов, PostgreSQL изнутри, 1-е ред., Москва: ДМК Пресс, 2023, p. 662 .

[4] Б. А. Новиков, Е. А. Горшкова и Н. Г. Графеева, Основы технологии баз данных, 2-е ред., Москва: ДМК пресс, 2020, p. 582.

[5] Е. П. Моргунов, PostgreSQL. Основы языка SQL, 1-е ред., Санкт-Петербург: БХВ-Петербург, 2018, p. 336.