



Промышленные СУБД
Лабораторная работа №3

Лабораторная работа №3

Выборка данных

Цель: научить использовать оператор SELECT и его основные разделы для выборки данных в соответствии с заданными условиями.

Теоретический материал: перед выполнением лабораторной работы рекомендуется изучить лекцию №4 «Выборка данных», в которой изучается один из основных операторов языка SQL – SELECT и объясняется назначение каждого из его разделов.

Требования к отчету: по результатам работы представить набор SQL-скриптов, решающих задачи из раздела «Самостоятельная работа».

Замечание. Для выполнения данной лабораторной работы требуется БД demo-small-20161013. Скачать БД можно по ссылке ниже.

<https://postgrespro.ru/docs/postgrespro/10/demodb-bookings-installation.html>

После скачивания, распаковать архив. Далее, открываем терминал. Переходим в каталог, куда мы скачивали PostgreSQL командой cd и заходим в bin. Например: `cd C:\Program Files\PostgreSQL\15\bin`. Далее пишем следующую команду:

`psql -f *путь до demo-small-20170815.sql* -h *сервер* -p *порт* -U *имя пользователя* -d *название БД куда импортируем*`

Затем вводим пароль пользователя (тот который мы придумали в самом начале работы).

Пример:

```
C:\Program Files\PostgreSQL\15\bin>psql -f C:\Users\UZ1\Downloads\demo-small\demo-small-20170815.sql -h localhost -p 5432 -U postgres -d demo
Пароль пользователя postgres:
```

Готово, в БД demo, появилась схема bookings, а в ней нужные нам таблицы.

Задание 1. Получите список аэропортов с указанием их кода и города из таблицы `airports_data` БД *demo*.

Указания к выполнению:

1. Для выполнения задания нам потребуются атрибуты `airport_code`, `airport_name`, `city` из таблицы `airports_data`.

2. Убедитесь, что БД demo является текущей, и выполните следующий запрос:

```
SELECT airport_code, airport_name, city FROM airports_data;
```

Результат выполнения запроса показан на рис. 1.

	airport_code [PK] character (3)	airport_name jsonb	city jsonb
1	YKS	{"en": "Yakutsk Airport", "ru": "Якутск"}	{"en": "Yakutsk", "ru": "Якутск"}
2	MJZ	{"en": "Mirny Airport", "ru": "Мирный"}	{"en": "Mirnyj", "ru": "Мирный"}
3	KHV	{"en": "Khabarovsk-Novy Airport", "ru": "Хабаровск-Новый"}	{"en": "Khabarovsk", "ru": "Хабаровск"}
4	PKC	{"en": "Yelizovo Airport", "ru": "Елизово"}	{"en": "Petropavlovsk", "ru": "Петропавловск-Камчатский"}
5	UUS	{"en": "Yuzhno-Sakhalinsk Airport", "ru": "Хомутово"}	{"en": "Yuzhno-Sakhalinsk", "ru": "Южно-Сахалинск"}
6	VVO	{"en": "Vladivostok International Airport", "ru": "Владивосток"}	{"en": "Vladivostok", "ru": "Владивосток"}
7	LED	{"en": "Pulkovo Airport", "ru": "Пулково"}	{"en": "St. Petersburg", "ru": "Санкт-Петербург"}
8	KGD	{"en": "Khrabrovo Airport", "ru": "Храброво"}	{"en": "Kaliningrad", "ru": "Калининград"}
9	KEJ	{"en": "Kemerovo Airport", "ru": "Кемерово"}	{"en": "Kemerovo", "ru": "Кемерово"}
10	CEK	{"en": "Chelyabinsk Balandino Airport", "ru": "Челябинск"}	{"en": "Chelyabinsk", "ru": "Челябинск"}
11	MQF	{"en": "Magnitogorsk International Airport", "ru": "Магнитогорск"}	{"en": "Magnitogorsk", "ru": "Магнитогорск"}
12	PEE	{"en": "Bolshoye Savino Airport", "ru": "Пермь"}	{"en": "Perm", "ru": "Пермь"}
13	SGC	{"en": "Surgut Airport", "ru": "Сургут"}	{"en": "Surgut", "ru": "Сургут"}
14	BZK	{"en": "Bryansk Airport", "ru": "Брянск"}	{"en": "Bryansk", "ru": "Брянск"}
15	MRV	{"en": "Mineralnyye Vody Airport", "ru": "Минеральные Воды"}	{"en": "Mineralnye Vody", "ru": "Минеральные Воды"}

Рис. 1. Список аэропортов.

Задание 2. Получите список мест с указанием числа места (первые 2 символа), сектора (3 символ) места и класса (бизнес, эконом, комфорт), а также его идентификационного номера. Список должен быть упорядочен по номеру места.

Указания к выполнению:

1. Первый столбец мы должны будем переименовать в seats_class при помощи AS, так как в нем необходимо объединить данные из двух столбцов: seat_no и fare_conditions.
2. Для получения только первых буквы места и получения только последнего символа места воспользуемся функцией SUBSTRING (название столбца, первый символ, количество символов), выводим значения через пробел при помощи || ' '.
3. Для сортировки результирующей таблицы необходимо добавить раздел ORDER BY.
4. Выполните следующий код:

```
SELECT (SUBSTRING(seat_no,1,2) || ' ' || SUBSTRING(seat_no,3,1) || ' ' ||
fare_conditions )AS seats_class, aircraft_code FROM seats
ORDER BY seat_no;
```

Результат выполнения запроса показан на рис. 2.

	seats_class text	aircraft_code character (3)
1	10 A Economy	321
2	10 A Economy	319
3	10 A Economy	320
4	10 A Economy	SU9
5	10 B Economy	321
6	10 B Economy	320
7	10 B Economy	319
8	10 B Economy	733

Рис. 2. Список мест.

Замечание. По умолчанию сортировка с помощью **ORDER BY** осуществляется по возрастанию, для сортировки в убывающем порядке указывается – **DESC** и команда будет иметь следующий вид: **ORDER BY seat_no DESC**.

Задание 3. Получите список самолётов, дальность полёта которых находится в диапазоне от 3000 км до 6000 км, отсортировав его по дальности.

Указания к выполнению:

1. Для выбора записей по заданному критерию необходимо воспользоваться разделом **WHERE**.
2. Можно использовать составное условие, тогда код будет выглядеть так:

```
SELECT * FROM aircrafts
WHERE (range>=3000) and (range<=6000)
ORDER BY 3;
```

Замечание. При сортировке командой **ORDER BY** можно указывать как название столбца, по которому происходит сортировка, так и его порядковый номер из строки **SELECT**.

3. Другой вариант: заменить два оператора сравнения одним логическим оператором **BETWEEN**, с помощью которого можно получить ответ на вопрос, лежит ли величина в указанном диапазоне:

```
SELECT * FROM aircrafts
WHERE range BETWEEN 3000 and 6000
ORDER BY 3;
```

Результат выполнения запроса показан на рис. 3.

aircraft_code character (3)	model text	range integer
SU9	Сухой Суперджет-100	3000
733	Боинг 737-300	4200
321	Аэробус A321-200	5600
320	Аэробус A320-200	5700

Рис. 3. Список самолётов с диапазоном полёта 3000-6000 км.

Задание 4. Выведите все кодировки самолётов, их модели и дальность полёта, кроме самолётов модели Аэробус и Боинг.

Указания к выполнению:

1. Для поиска по шаблону символьных строк используется логический оператор **LIKE**, который чаще всего применяется в ситуациях, когда не известно точное совпадение.
2. В шаблоне нам потребуется указать служебный символ **%**, который подразумевает любую строку, состоящую из 0 и более символов.
3. Используя таблицу *aircrafts* составим запрос:

```
SELECT * FROM aircrafts
WHERE model NOT LIKE 'Аэробус%'
AND model NOT LIKE 'Боинг%';
```

Результат выполнения запроса показан на рис. 4.

	aircraft_code character (3)	model text	range integer
1	SU9	Сухой Суперджет-100	3000
2	CN1	Сессна 208 Караван	1200
3	CR2	Бомбардье CRJ-200	2700

Рис. 4. Все самолёты авиакомпании, кроме Аэробуса и Боинга.

Интересен пример использования оператора ESCAPE вместе с оператором LIKE, который позволяет найти в строках данных такие значения, которые содержат специальные символы.

Выполним для начала запрос, который отобразит все данные таблицы:

SELECT * FROM public."Employee"

	EmployeId [PK] integer	SecondName character varying	FirstName character varying	LastName character varying	Position character varying	BirthDate date	hasLogin boolean
1	2	Петров	Петр	Петрович	Директов	1965-08-28	false
2	3	Калинова	Мария	Петровна	Директор	1993-03-17	false
3	5	Торбова	Виктория	Петровна	Менеджер	1983-03-01	false
4	4	Грибоедова	Екатерина	Александровна	Бухгалтер	1993-12-12	false
5	1	Иванов	Иван	Иванович	Секретарь	1987-03-30	true
6	6	Петров%	Петр	Петрович	Садовник	1953-03-13	false
7	7	Петров_	Петр	Петрович	Садовник	1953-03-13	false

Как можно увидеть из рисунка в третьем столбце «SecondName» есть очень похожие значения у трех персон: «Петров», «Петров_», «Петров%». Напишем запрос с помощью оператора LIKE, который выведет данный список:

SELECT * FROM public."Employee" WHERE "SecondName" LIKE 'П%'

	EmployeId [PK] integer	SecondName character varying	FirstName character varying	LastName character varying	Position character varying	BirthDate date	hasLogin boolean
1	2	Петров	Петр	Петрович	Директов	1965-08-28	false
2	6	Петров%	Петр	Петрович	Садовник	1953-03-13	false
3	7	Петров_	Петр	Петрович	Садовник	1953-03-13	false

А теперь попробуем создать запрос, который поможет нам выбрать из списка персону «Петров_». С помощью оператора LIKE такой запрос создать невозможно, так как отразятся те же самые три строки данных, так как символ '_' в операторе LIKE будет означать только любой одинарный символ в значении:

SELECT * FROM public."Employee" WHERE "SecondName" LIKE 'П%_'

	EmployeId [PK] integer	SecondName character varying	FirstName character varying	LastName character varying	Position character varying	BirthDate date	hasLogin boolean
1	2	Петров	Петр	Петрович	Директов	1965-08-28	false
2	6	Петров%	Петр	Петрович	Садовник	1953-03-13	false
3	7	Петров_	Петр	Петрович	Садовник	1953-03-13	false

И только когда применим дополнительно оператор ESCAPE, указав в одинарных кавычках восклицательный знак, который будет означать, что если знак ! есть в LIKE, то после такого знака берется следующий за ним и воспринимается, как специальный символ, который надо найти в строке данных. Результат вы видите ниже:

```
SELECT * FROM public."Employee" WHERE "SecondName" LIKE 'П%!_'
ESCAPE '!'
```

	EmployeId [PK] integer	SecondName character varying	FirstName character varying	LastName character varying	Position character varying	BirthDate date	hasLogin boolean
1	7	Петров_	Петр	Петрович	Садовник	1953-03-13	false

Задание 5. Получите список самолётов компаний модели Аэробус или Боинг

Указания к выполнению:

1. Оператор ~ ищет совпадение с шаблоном с учетом регистра символов.
2. Символ «^» означает, что поиск совпадения будет привязан к началу строки.
3. Если же требуется проверить наличие такого символа в составе строки, то перед ним нужно поставить символ обратной косой черты: «\^».
4. Выражение в круглых скобках означает альтернативный выбор между значениями, разделяемыми символом «|». Поэтому в выборку попадут значения, начинающиеся либо на «А», либо на «Боин».
5. Запрос будет выглядеть следующим образом:

```
SELECT * FROM aircrafts WHERE model ~ '^(A|Боин)';
```

Результат выполнения запроса показан на рис. 5.

aircraft_code character (3)	model text	range integer
773	Боинг 777-300	11100
763	Боинг 767-300	7900
320	Аэробус A320-200	5700
321	Аэробус A321-200	5600
319	Аэробус A319-100	6700
733	Боинг 737-300	4200

Рис. 5. Самолёты модели Боинг и Аэробус.

Задание 6. Получите список рейсов, у которых не указан ближайший вылет.

Указания к выполнению:

1. Для проверки наличия/отсутствия значения используется функция выборки **IS/NOTNULL / ISNULL**.

2. Составим следующий запрос по таблице flights:

```
SELECT * FROM flights WHERE actual_departure ISNULL;
```

Результат выполнения запроса показан на рис. 6.

	flight_id [PK] integer	flight_no character (6)	scheduled_departure timestamp with time zone	scheduled_arrival timestamp with time zone	departure_airport character (3)	arrival_airport character (3)	status character var	aircraft_code character (3)	actual_departure timestamp with time zone
1	1185	PG0134	2017-09-10 11:50:00+05	2017-09-10 16:55:00+05	DME	BTK	Scheduled	319	[null]
2	3979	PG0052	2017-08-25 16:50:00+05	2017-08-25 19:35:00+05	VKO	HMA	Scheduled	CR2	[null]
3	4739	PG0561	2017-09-05 14:30:00+05	2017-09-05 16:15:00+05	VKO	AER	Scheduled	763	[null]
4	5502	PG0529	2017-09-12 11:50:00+05	2017-09-12 13:20:00+05	SVO	UFA	Scheduled	763	[null]
5	6938	PG0461	2017-09-04 14:25:00+05	2017-09-04 15:20:00+05	SVO	ULV	Scheduled	SU9	[null]
6	7784	PG0667	2017-09-10 17:00:00+05	2017-09-10 19:30:00+05	SVO	KRO	Scheduled	CR2	[null]
7	9478	PG0360	2017-08-28 11:00:00+05	2017-08-28 13:35:00+05	LED	REN	Scheduled	CR2	[null]
8	11095	PG0560	2017-09-24 17:05:00+05	2017-09-24 19:10:00+05	SVO	SCW	Scheduled	733	[null]

Рис. 6. Рейсы, для которых не указан ближайший вылет.

Самостоятельная работа

Необходимо построить SQL-запросы к базе данных, спроектированной по индивидуальному заданию в рамках лабораторной работы №1.

Требования к запросам: запросы должны быть логичны и целесообразны, и соответствовать специфике предметной области.

Необходимо построить по одному SQL-запросу следующих типов:

1. Запрос на полную выборку данных.
2. Запрос на выборку данных без повторений.
3. Запрос на выборку первых 10 записей.
4. Запрос на выборку последних 15 записей.
5. Запросы на выполнение функций Average, Max, Min.
6. Сконструируйте запросы с использованием оператора Where:
 - запрос на возвращение определенного кортежа по первичному ключу;
 - запросы на возвращение значения по условиям больше, меньше и между;
 - запросы на возвращении всех кортежей по условию с использованием оператора LIKE и ESCAPE;
 - запрос на возвращение кортежей со сложным условием на основе логических операторов И, ИЛИ, НЕ, EXISTS;
 - запрос с использованием оператора NOT NULL в условии отбора.
7. Запрос с простыми условиями, условиями, содержащими IN или BETWEEN.
8. Запросы с сортировкой по нескольким полям, направлениям.
9. Запросы с использованием групповых операций (группировка статистические функции, отбор по групповым функциям).
10. Запросы с операцией над множествами (обязательно используя сортировку).
11. Запросы на обновление.
12. Запросы на удаление.
13. Запросы на вставку.
14. Используя таблицу с персональными данными из своей БД или demo БД в PostgreSQL отобразите список сотрудников/персон (указав их **Фамилию И.** в одной колонке), которые в следующем месяце будут отмечать юбилей, **с указанием возраста, даты рождения, даты юбилея.** Заголовки должны соответствовать шаблону вывода данных.