ЛАБОРАТОРНАЯ РАБОТА №2

ОСНОВЫ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ SQL. SELECT для СУБД PostgreSQL.

Цель: сформировать знания и умения по программированию на языке SQL, приобрести практические навыки работы со средствами языка SQL для выборки и редактирования данных в БД.

Содержание лабораторной работы:

1. Изучить теоретические сведения лабораторной работы.
2. Открыть базу данных, созданную в предыдущей лабораторной работе.
3. Создать к базе данных SELECT-запросы следующих видов:

    a. запрос, выбирающий все данные из таблицы;
    b. запрос, выбирающий данные из некоторых столбцов таблицы;
    c. запрос с использованием сортировки данных;
    d. запрос с использованием ограничения на выборку данных;
    e. запрос с использованием операторов сравнения;
    f. запрос с использованием оператора BETWEEN;
    g. запрос с использованием оператора IN, содержащий подзапрос;
    h. запрос с использованием оператора LIKE и строковых функций;
    i. запрос с использованием предиката IS NULL;
    j. запрос с использованием агрегатных функций;
    k. запрос с использованием агрегатных функций и предложения HAVING;
    l. запрос, выбирающий данные из нескольких таблиц с использованием соединения по предикату;
    m. запрос с использованием ключевого слова DISTINCT;
    n. запрос с использованием оператора EXISTS;
    o. запрос с использованием функции CASE;

4. Выполнить задания по варианту.
5. Подготовиться к защите лабораторной работы.

Краткий вспомогательный материал

SQL — аббревиатура выражения Structured Query Language (язык структурированных запросов). SQL основывается на реляционной алгебре и специально разработан для взаимодействия с реляционными базами данных.

SQL является, информационно-логическим языком, предназначенным для описания хранимых данных, их извлечения и модификации. SQL не является языком программирования. Конкретные реализации языка, как правило, включают различные процедурные расширения.

Язык SQL представляет собой совокупность операторов, которые можно разделить на четыре группы:

- ***DDL*** (Data Definition Language) - операторы определения данных
- ***DML*** (Data Manipulation Language) – операторы манипуляции данными
- ***DCL*** (Data Control Language) - операторы определения доступа к данным
- ***TCL*** (Transaction Control Language) - операторы управления транзакциями

SQL является стандартизированным языком. Стандартный SQL поддерживается комитетом стандартов ANSI (Американский национальный институт стандартов), и соответственно называется ANSI SQL.

Многие разработчики СУБД расширили возможности SQL, добавив в язык дополнительные операторы или инструкции. В PostgreSQL Server используется язык PL/pgSQL.

Оператор SELECT.

Для выборки данных используется команда:

```
SELECT [ ALL | DISTINCT ] < список полей > FROM < таблица> [ , < таблица2
>...n ] ]

[ WHERE < условие > ]

[ GROUP BY < поле > | <Integer> [,...]] [ HAVING < условие >]
[ ORDER BY < поле > | <Integer> [ ASC|DESC ]  [,...]]

[LIMIT < число >] [OFFSET  < число >]
```

Пример. Выбрать все сведения о проектах.

```
SELECT * FROM projects;   /* будут выведены все поля */

SELECT id, project_name FROM projects; /* будут выведены два поля */
```

Рассмотрим отдельные элементы синтаксиса инструкции SELECT.

ALL - указывает на то, что в результирующем наборе могут появляться повторяющиеся элементы, является значением по умолчанию.

DISTINCT - в результирующем наборе возвращаются только уникальные результаты.

ORDER BY - сортировка строк результирующей таблицы данных

LIMIT - возвращается не больше заданного числа строк

OFFSET - пропустить указанное число строк, прежде чем начать выдавать строки.

```
ORDER BY  < поле > | < Integer > [ ASC|DESC ]  [,...]
```

ASC сортирует данные в восходящем порядке, **DESC** – в обратном. Вместо имен полей могут быть использованы их порядковые номера в списке полей результирующей таблицы.

Пример. Выбрать сведения о проектах, отсортировав их по названию проекта.

```
SELECT id, project_name FROM projects
ORDER BY project;
```

Пример. Извлечь из выборки последние пять записей о проектах.

```
SELECT id, project_name FROM projects ORDER BY DESC LIMIT 5; -- первые 5
                записей из запроса, отсортированного в обратном порядке
```

WHERE - ограничение выборки данных из указанных таблиц

```
WHERE < условие >
```

Пример. Выбрать сведения о сотрудниках, работающих в первом отделе:

```
SELECT   *
FROM employees WHERE id_depart=1;
```

GROUP BY - объединяет результат запроса в группы

```
GROUP BY < поле > | < Integer > [,...]
```

Если внутри GROUP BY используется ORDER BY, то строки сортируются внутри каждой группы результирующих строк.

Пример. Посчитать, сколько заданий в каждом проекте.

```
SELECT id_project, COUNT(*) AS num_tasks /* AS   используется для
                                          назначения  псевдонима  столбцу */
FROM tasks GROUP BY id_project ;
```

HAVING - используется вместе с GROUP, для того чтобы выбирать только определенные группы строк данных, которые удовлетворяют указанному условию.

```
HAVING < условие >
```

Пример. Вывести сведения о проектах, в которых больше трех заданий.

```
SELECT id_project FROM tasks GROUP BY id_project
HAVING COUNT(*)>3;
```

BETWEEN - проверяет, попадают ли значения проверяемого выражения в диапазон, задаваемый пограничными выражениями, соединяемыми служебным словом AND.

```
< Проверяемое_выражение > [NOT] BETWEEN < Начальное_выражение > AND
                                                    < Конечное_выражение >
```

Пример. Выбрать сведения о сотрудниках из первого, второго и третьего отделов.

```
SELECT   * FROM employees
WHERE id_depart BETWEEN 1 AND 3;
```

IN - проверяет, попадают ли значения проверяемого выражения во множество:

```
< Проверяемое_выражение > [NOT] IN (< подзапрос >)|
                                   (< выражение_для_вычисления_значения >,...)
```

Пример. Выбрать сведения о сотрудниках из первого и третьего отделов.

```
SELECT   * FROM employees WHERE id_depar IN(1,3);
```

Пример. Выбрать сведения о сотрудниках из отделов, относящихся к подразделению 3.

```
SELECT   * FROM employees WHERE id_depar IN
                    (SELECT id_dep from department WHERE dep_branch = 3);
```

LIKE - сравнение строк с шаблоном

```
< проверяемое_значение > [NOT] LIKE < шаблон > [ESCAPE < символ >]
```

Пример. Найти все проекты, названия которых начинаются с буквы А.

```
SELECT * FROM projects WHERE project LIKE 'A%'
```

Стандартом предусмотрены следующие строковые функции.

```
CONCAT()       Объединение строк

LOWER(A)       Приведение A к нижнему регистру

LEFT()         Возвращает строку символов указанной длины, отсчитывая слева
LEN()       Длина строки

SUBSTRING(A,B,C) Возвращает подстроку из A, с позиции B до позиции C

LTRIM(str)     Удаляет все начальные пробелы из строки str
RTRIM(str)     Удаляет хвостовые пробелы из строки str
REPLACE(A,B,C) Заменяет все подстроки B в строке A на подстроку C
STRCMP()       Возвращает 0, если строки одинаковые

UPPER(A)       Переводит A в верхний регист
```

Пример. Вывести названия проектов в верхнем регистре.

```
SELECT UPPER(project_name) FROM projects;
```

IS [NOT] NULL - позволяет проверить отсутствие (наличие) значения в полях

Пример. Выбрать сведения о несданных заданиях:

```
SELECT * FROM tasks WHERE date_turn IS NULL;
```

Стандартом предусмотрены следующие агрегатные функции:

```
COUNT(*)        Возвращает количество строк источника записей.
COUNT(<имя поля>) Возвращает количество значений в указанном
столбце. SUM(<имя_поля>) Возвращает сумму значений в указанном
столбце. AVG(<имя_поля>) Возвращает среднее значение в указанном
столбце. MIN(<имя_поля>) Возвращает минимальное значение в
указанном столбце. MAX(<имя_поля>) Возвращает максимальное
значение в указанном столбце.
```

JOIN - часто для решения задач необходимо выбирать данные, находящиеся в разных, связанных логически между собой таблицах. Синтаксис соединения таблиц имеет вид:

```
FROM < таблица1 > [AS < псевдоним_табл1 >] [INNER | LEFT | RIGHT | FULL] JOIN
     < таблица2 > [AS < псевдоним_табл2 >] [ON < предикат >]
```

Пример. Вывести номер телефона сотрудника Василькова.

```
SELECT    d.phone
FROM employees JOIN departments as d ON(employees.id_depart = d.id_depart)
WHERE employees.surname='Васильков';
```

EXISTS - принимает значение TRUE, если подзапрос возвращает любое количество строк, иначе его значение равно FALSE.

```
EXISTS ( < подзапрос1 > )
```

Пример. Выбрать проекты, которые не выполняются в данный момент.

```
SELECT id_project FROM projects
WHERE NOT EXISTS (SELECT id FROM tasks
                  WHERE id_project= projects.id_project)
```

UNION - выводит результаты двух или более запросов в один результирующий набор.

```
< запрос1 > UNION [ALL] < запрос2 >;
```

количество и порядок столбцов должны быть одинаковыми во всех запросах.

Пример. Вывести время выдачи зарплаты сотрудников первого отдела и сотрудников, работающих над проектом «P1»

```
SELECT id_employee, salary_hour FROM employees WHERE
id_depart=1 UNION

SELECT distinct id_employee, salary_hour

FROM employees as e join tasks as t on (e.id_employee = t.id_empl)
    JOIN projects ON (t.id_project=projects.id_project)

WHERE project= 'P1'
ORDER BY
salary_hour;
```

Выражение CASE - общее условное выражение, напоминающее операторы if/else в других языках программирования:

```
CASE WHEN условие THEN результат
     [WHEN ...]
     [ELSE результат]
END
```

Пример. Указать занятость сотрудников

```
SELECT id_employee,
      CASE WHEN count(tasks.id)<1 THEN 'free'
           ELSE 'busy'
      END
FROM employees left join tasks on(employees.id_employee = tasks.id_empl)
WHERE date_turn is NULL /*дата сдачи задания пустая */
GROUP BY id_employee;
```

Варианты:

Вариант	Задание
1-5	1. Из таблицы ORDERS выбрать заказы со сроком даты заказа более ранним, чем <любая дата>. Список отсортировать по номеру заказа в обратном порядке.
	2. Получить информацию о покупателях, которые не сделали ни одного заказа. Список отсортировать по фамилии.
6-10	1. Получить информацию о заказе: id заказа, фамилию, имя, адрес, дата заказа, дата отправки. Список отсортировать так, чтоб заказы, отправленные раньше, выводились в конце.
	2. Получить информацию о покупателях (фамилия, имя, адрес, телефон), которые оплатили заказ. Список отсортировать по фамилиям.
11-15	1. Получить список заказов от компании «НАЗВАНИЕ КОМПАНИИ». Список отсортировать по дате заказа.

	2. Получить информацию о покупателях (фамилия, имя, адрес, телефон, город), чьи заказы были отменены. Список отсортировать по городу и фамилиям.
16-20	1. Получить информацию о покупателях (компания, фамилия, имя, адрес, телефон, город), которые совершили заказ со статусом «P». Список отсортировать по городу и фамилиям.
	2. Получить информацию о количестве покупателей в каждом из городов, считать только оплаченные заказы. Список отсортировать по количеству покупателей.
21-25	1. Получить список заказов, фамилии, телефоны и адреса покупателей, которые совершили заказ с <любая дата> по <любая дата>. Список отсортировать по дате заказа.
	2. Получить информацию о покупателе (фамилия, адрес, телефон, город, дата заказа) с максимальной суммой заказа.
26-30	1. Получить информацию о товарах, которые находятся на складе и цена которых от 90 до 1000. Список отсортировать по цене.
	2. Получить информацию о покупателе (фамилия, адрес, телефон, город, дата заказа) с минимальной суммой заказа.

Контрольные вопросы

1. Что такое SQL? Предназначение?
2. Какие существуют группы операторов в языке SQL?
3. Каково назначение команды SELECT?
4. Опишите структуру команды SELECT.
5. Как осуществляется сортировка? Использование сортировки вместе с группировкой. Прямой и обратный порядок сортировки.
6. Как осуществляется ограничение количества выбираемых записей?
7. Как реализуется ограничение выбора данных с помощью конструкции WHERE и операторов BETWEEN, IN, LIKE, IS NULL?
8. Как осуществляется группировка данных?
9. Какие агрегатные функции предусмотрены стандартом? Их назначение.
10. Как применяется конструкция HAVING BY?
11. Опишите синтаксис соединения таблиц.
12. Опишите структуру и принцип работы оператора UNION.
13. Каково назначение предиката EXIST?
14. Каково назначении функции CASE?