**Лабораторная работа № 5\_1.**

**Coderlesson PostgreSQL**

**https://coderlessons.com/tutorials/bazy-dannykh/vyuchit-postgresql/postgresql-kratkoe-rukovodstvo**

**Загрузка учебной базы HR, запросы.**

**Задание №1. Создать схему**

**Схема** – это именованная коллекция таблиц. Схема также может содержать представления, индексы, последовательности, типы данных, операторы и функции. Схемы аналогичны каталогам на уровне операционной системы, за исключением того, что схемы не могут быть вложенными:

create schema myschema;

создание таблицы в схеме

create table myschema.company(

ID INT NOT NULL,

NAME VARCHAR (20) NOT NULL,

AGE INT NOT NULL,

ADDRESS CHAR (25),

SALARY DECIMAL (18, 2),

PRIMARY KEY (ID)

);

Посмотреть таблицу

select \* from myschema.company;

удалить схему

DROP SCHEMA myschema;

удалить схему, если в ней есть объекты

DROP SCHEMA myschema CASCADE;

**Преимущества использования схемы**

* Это позволяет многим пользователям использовать одну базу данных, не мешая друг другу.
* Он организует объекты базы данных в логические группы, чтобы сделать их более управляемыми.
* Сторонние приложения могут быть помещены в отдельные схемы, чтобы они не конфликтовали с именами других объектов.

**Задание №2.**

Создать схему **testdb**, создать таблицу COMPANY

create schema testdb;

CREATE TABLE COMPANY(

ID INT PRIMARY KEY NOT NULL,

NAME TEXT NOT NULL,

AGE INT NOT NULL,

ADDRESS CHAR(50),

SALARY REAL,

JOIN\_DATE DATE

);

INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY,JOIN\_DATE) VALUES (1, 'Paul', 32, 'California', 20000.00,'2001-07-13');

INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,JOIN\_DATE) VALUES (2, 'Allen', 25, 'Texas', '2007-12-13');

В следующем примере используется предложение DEFAULT для столбца JOIN\_DATE вместо указания значения –

INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY,JOIN\_DATE) VALUES (3, 'Teddy', 23, 'Norway', 20000.00, DEFAULT );

В следующем примере вставляется несколько строк с использованием многострочного синтаксиса VALUES –

INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY,JOIN\_DATE) VALUES (4, 'Mark', 25, 'Rich-Mond ', 65000.00, '2007-12-13' ), (5, 'David', 27, 'Texas', 85000.00, '2007-12-13');

INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)

VALUES (6, 'Kim', 22, 'South-Hall', 45000.00 );

INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)

VALUES (7, 'James', 24, 'Houston', 10000.00 );

ПРОСМОТРЕТЬ СТРУКТУРУ ТАБЛИЦЫ МОЖНО Ч.З. ЗАПРОС (аналог DESC в Oracle):

SELECT column\_name, column\_default, data\_type FROM INFORMATION\_SCHEMA. COLUMNS WHERE table\_name = 'COMPANY';

**Задание №3.**

**ПРИМЕРЫ SELECT:**

SELECT \* FROM COMPANY WHERE AGE >= 25 OR SALARY >= 65000;

SELECT \* FROM COMPANY WHERE AGE >= 25 AND SALARY >= 65000;

SELECT \* FROM COMPANY WHERE NAME LIKE 'Pa%';

SELECT \* FROM COMPANY WHERE AGE IS NOT NULL;

SELECT \* FROM COMPANY WHERE AGE IN ( 25, 27 );

SELECT \* FROM COMPANY WHERE AGE NOT IN ( 25, 27 );

SELECT \* FROM COMPANY WHERE AGE BETWEEN 25 AND 27;

SELECT AGE FROM COMPANY

WHERE EXISTS (SELECT AGE FROM COMPANY WHERE SALARY > 65000);

SELECT FROM table\_name

WHERE column LIKE 'XXXX%'

SELECT \* FROM COMPANY WHERE AGE::text LIKE '2%'

SELECT \* FROM COMPANY WHERE ADDRESS LIKE '%-%';

SELECT \* FROM COMPANY LIMIT 4;

SELECT \* FROM COMPANY LIMIT 3 OFFSET 2;

SELECT \* FROM COMPANY ORDER BY AGE ASC;

SELECT \* FROM COMPANY ORDER BY NAME, SALARY ASC;

SELECT \* FROM COMPANY ORDER BY NAME DESC;

SELECT NAME, SUM(SALARY) FROM COMPANY GROUP BY NAME;

SELECT NAME FROM COMPANY GROUP BY name HAVING count(name) < 2;

SELECT NAME FROM COMPANY GROUP BY name HAVING count(name) > 1;

INSERT INTO COMPANY VALUES (8, 'Paul', 24, 'Houston', 20000.00);

INSERT INTO COMPANY VALUES (9, 'James', 44, 'Norway', 5000.00);

INSERT INTO COMPANY VALUES (10, 'James', 45, 'Texas', 5000.00);

SELECT NAME, SUM(SALARY) FROM COMPANY GROUP BY NAME ORDER BY NAME;

INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)

VALUES (8, 'Paul', 32, 'California', 20000.00 );

INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)

VALUES (9, 'Allen', 25, 'Texas', 15000.00 );

SELECT name FROM COMPANY;

SELECT DISTINCT name FROM COMPANY;

**ОБНОВЛЕНИЕ запроса**

UPDATE COMPANY SET SALARY = 15000 WHERE ID = 3;

UPDATE COMPANY SET ADDRESS = 'Texas', SALARY=20000;

**Задание №4.**

Предложение WITH

В PostgreSQL запрос WITH предоставляет возможность написать вспомогательные операторы для использования в запросе большего размера. Это помогает разбить сложные и большие запросы на более простые формы, которые легко читаются. Эти операторы, часто называемые выражениями общих таблиц или CTE, могут рассматриваться как определяющие временные таблицы, которые существуют только для одного запроса.

Запрос WITH, являющийся запросом CTE, особенно полезен, когда подзапрос выполняется несколько раз. Это одинаково полезно вместо временных таблиц. Он вычисляет агрегацию один раз и позволяет нам ссылаться на нее по ее имени (может быть несколько раз) в запросах.

Рекурсивные запросы WITH или Иерархические запросы – это форма CTE, в которой CTE может ссылаться на себя, т. Е. Запрос WITH может ссылаться на свой собственный вывод, следовательно, имя рекурсивно.

With CTE AS

(Select

ID

, NAME

, AGE

, ADDRESS

, SALARY

FROM COMPANY )

Select \* From CTE;

select \* from COMPANY;

Теперь напишем запрос, используя ключевое слово RECURSIVE вместе с предложением WITH, чтобы найти сумму зарплат менее 20000, как показано ниже:

WITH RECURSIVE t(n) AS (

VALUES (0)

UNION ALL

SELECT SALARY FROM COMPANY WHERE SALARY < 20000

)

SELECT sum(n) FROM t;

Напишем запрос, используя операторы изменения данных вместе с предложением WITH, как показано ниже.

Сначала создайте таблицу COMPANY1, аналогичную таблице COMPANY. Запрос в примере эффективно перемещает строки из КОМПАНИИ в КОМПАНИ1. DELETE в WITH удаляет указанные строки из COMPANY, возвращая их содержимое с помощью предложения RETURNING; а затем первичный запрос читает этот вывод и вставляет его в КОМПАНИЮ TABLE –

CREATE TABLE COMPANY1(

ID INT PRIMARY KEY NOT NULL,

NAME TEXT NOT NULL,

AGE INT NOT NULL,

ADDRESS CHAR(50),

SALARY REAL

);

WITH moved\_rows AS (

DELETE FROM COMPANY

WHERE

SALARY >= 30000

RETURNING \*

)

INSERT INTO COMPANY1 (SELECT \* FROM moved\_rows);

SELECT \* FROM COMPANY;

SELECT \* FROM COMPANY1;

**Задание №4.**

**СОЕДИНЕНИЯ**

CROSS JOIN

ВНУТРЕННЕЕ СОЕДИНЕНИЕ

ЛЕВОЕ НАРУЖНОЕ СОЕДИНЕНИЕ

ПРАВИЛЬНОЕ НАРУЖНОЕ СОЕДИНЕНИЕ

ПОЛНОЕ НАРУЖНОЕ СОЕДИНЕНИЕ

drop table IF EXISTS department, company;

CREATE TABLE DEPARTMENT(

ID INT PRIMARY KEY NOT NULL,

DEPT CHAR(50) NOT NULL,

EMP\_ID INT NOT NULL

);

INSERT INTO DEPARTMENT (ID, DEPT, EMP\_ID)

VALUES (1, 'IT Billing', 1 );

INSERT INTO DEPARTMENT (ID, DEPT, EMP\_ID)

VALUES (2, 'Engineering', 2 );

INSERT INTO DEPARTMENT (ID, DEPT, EMP\_ID)

VALUES (3, 'Finance', 7 );

CREATE TABLE COMPANY(

ID INT PRIMARY KEY NOT NULL,

NAME TEXT NOT NULL,

AGE INT NOT NULL,

ADDRESS CHAR(50),

SALARY REAL,

JOIN\_DATE DATE

);

INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY,JOIN\_DATE) VALUES (1, 'Paul', 32, 'California', 20000.00,'2001-07-13');

INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,JOIN\_DATE) VALUES (2, 'Allen', 25, 'Texas', '2007-12-13');

INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY,JOIN\_DATE) VALUES (3, 'Teddy', 23, 'Norway', 20000.00, DEFAULT );

INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY,JOIN\_DATE) VALUES (4, 'Mark', 25, 'Rich-Mond ', 65000.00, '2007-12-13' ), (5, 'David', 27, 'Texas', 85000.00, '2007-12-13');

INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)

VALUES (6, 'Kim', 22, 'South-Hall', 45000.00 );

INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)

VALUES (7, 'James', 24, 'Houston', 10000.00 );

SELECT EMP\_ID, NAME, DEPT FROM COMPANY CROSS JOIN DEPARTMENT;

SELECT EMP\_ID, NAME, DEPT FROM COMPANY INNER JOIN DEPARTMENT

ON COMPANY.ID = DEPARTMENT.EMP\_ID;

SELECT EMP\_ID, NAME, DEPT FROM COMPANY LEFT OUTER JOIN DEPARTMENT

ON COMPANY.ID = DEPARTMENT.EMP\_ID;

SELECT EMP\_ID, NAME, DEPT FROM COMPANY RIGHT OUTER JOIN DEPARTMENT

ON COMPANY.ID = DEPARTMENT.EMP\_ID;

SELECT EMP\_ID, NAME, DEPT FROM COMPANY FULL OUTER JOIN DEPARTMENT

ON COMPANY.ID = DEPARTMENT.EMP\_ID;

SELECT EMP\_ID, NAME, DEPT FROM COMPANY INNER JOIN DEPARTMENT

ON COMPANY.ID = DEPARTMENT.EMP\_ID

UNION

SELECT EMP\_ID, NAME, DEPT FROM COMPANY LEFT OUTER JOIN DEPARTMENT

ON COMPANY.ID = DEPARTMENT.EMP\_ID;

SELECT EMP\_ID, NAME, DEPT FROM COMPANY INNER JOIN DEPARTMENT

ON COMPANY.ID = DEPARTMENT.EMP\_ID

UNION ALL

SELECT EMP\_ID, NAME, DEPT FROM COMPANY LEFT OUTER JOIN DEPARTMENT

ON COMPANY.ID = DEPARTMENT.EMP\_ID;

SELECT C.ID, C.NAME, C.AGE, D.DEPT

FROM COMPANY AS C, DEPARTMENT AS D

WHERE C.ID = D.EMP\_ID;