УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники

Информационные системы и базы данных

Лабораторная работа№ 3

Вариант 1147

Преподаватель: Харитонова А. Е.

Выполнила: Блинова М.А.

Группа: P33301

Задание:

Составить запросы на языке SQL (пункты 1-7).

1. Сделать запрос для получения атрибутов из указанных таблиц, применив фильтры по указанным условиям:
   Н_ТИПЫ_ВЕДОМОСТЕЙ, Н_ВЕДОМОСТИ.
   Вывести атрибуты: Н_ТИПЫ_ВЕДОМОСТЕЙ.ИД, Н_ВЕДОМОСТИ.ЧЛВК_ИД.
   Фильтры (AND):
   a) Н_ТИПЫ_ВЕДОМОСТЕЙ.НАИМЕНОВАНИЕ > Ведомость.
   b) Н_ВЕДОМОСТИ.ИД = 1250981.
   c) Н_ВЕДОМОСТИ.ИД < 1457443.
   Вид соединения: LEFT JOIN.
2. Сделать запрос для получения атрибутов из указанных таблиц, применив фильтры по указанным условиям:
   Таблицы: Н_ЛЮДИ, Н_ВЕДОМОСТИ, Н_СЕССИЯ.
   Вывести атрибуты: Н_ЛЮДИ.ИМЯ, Н_ВЕДОМОСТИ.ИД, Н_СЕССИЯ.ЧЛВК_ИД.
   Фильтры (AND):
   a) Н_ЛЮДИ.ФАМИЛИЯ = Петров.
   b) Н_ВЕДОМОСТИ.ДАТА > 2010-06-18.
   Вид соединения: INNER JOIN.
3. Вывести число названий дисциплин без учета повторений.
   При составлении запроса нельзя использовать DISTINCT.
4. Выдать различные фамилии студентов и число людей с каждой из этих фамилий, ограничив список фамилиями, встречающимися более 50 раз на ФКТИУ.
   Для реализации использовать подзапрос.
5. Выведите таблицу со средним возрастом студентов во всех группах (Группа, Средний возраст), где средний возраст равен минимальному возрасту в группе 3100.
6. Получить список студентов, зачисленных после первого сентября 2012 года на первый курс заочной формы обучения (специальность: Программная инженерия). В результат включить:
   номер группы;
   номер, фамилию, имя и отчество студента;
   номер и состояние пункта приказа;
   Для реализации использовать подзапрос с EXISTS.
7. Сформировать запрос для получения числа на ФКТИУ отличников.

Реализация запросов на SQL:

1.
```
SELECT type.ИД, ved.ЧЛВК_ИД FROM Н_ТИПЫ_ВЕДОМОСТЕЙ AS type LEFT JOIN Н_ВЕДОМОСТИ AS ved ON type.ИД = ved.ТВ_ИД WHERE (
        type.НАИМЕНОВАНИЕ > 'Ведомость' AND
        ved.ИД = 1250981 AND
        ved.ИД < 1457443
);
```

2.
```
SELECT people.ИМЯ, ved.ИД, session.ЧЛВК_ИД FROM Н_ЛЮДИ AS people JOIN Н_ВЕДОМОСТИ AS ved ON people.ИД = ved.ЧЛВК_ИД  JOIN Н_СЕССИЯ AS session ON ved.СЭС_ИД = session.СЭС_ИД WHERE (
        people.ФАМИЛИЯ = 'Петров' AND
        ved.ДАТА > '2010-06-18'
);
```

3. Готовый запрос:

```
SELECT (
        SELECT COUNT(НАИМЕНОВАНИЕ) FROM Н_ДИСЦИПЛИНЫ
                WHERE НАИМЕНОВАНИЕ IN (
                        SELECT НАИМЕНОВАНИЕ FROM Н_ДИСЦИПЛИНЫ
                        GROUP BY НАИМЕНОВАНИЕ HAVING COUNT(НАИМЕНОВАНИЕ) = 1
                )
```

```
)
+ (
        SELECT COUNT(*) FROM (
                SELECT COUNT(НАИМЕНОВАНИЕ) FROM Н_ДИСЦИПЛИНЫ
                GROUP BY НАИМЕНОВАНИЕ HAVING COUNT(НАИМЕНОВАНИЕ) > 1
        ) AS value
);
```

Пояснение:

Следующим запросом мы находим количество уникальных названий дисциплин, не учитывая повторяющиеся:

```
SELECT COUNT(НАИМЕНОВАНИЕ) FROM Н_ДИСЦИПЛИНЫ WHERE НАИМЕНОВАНИЕ IN (SELECT НАИМЕНОВАНИЕ FROM Н_ДИСЦИПЛИНЫ GROUP BY НАИМЕНОВАНИЕ HAVING COUNT(НАИМЕНОВАНИЕ) = 1);        //456
```

Этот запрос находит количество повторяющихся названий дисциплин, это нужно, чтобы учесть их по одному разу:

```
SELECT COUNT(*) FROM (SELECT COUNT(НАИМЕНОВАНИЕ) FROM Н_ДИСЦИПЛИНЫ GROUP BY НАИМЕНОВАНИЕ HAVING COUNT(НАИМЕНОВАНИЕ) > 1) AS value;          //4
```

Далее, чтобы получить количество различных названий дисциплин нам необходимо просто сложить эти значения, для этого я использую следующую конструкцию:

```
SELECT (результат запроса1) + (результат запроса2);                    //460
```

А могло бы быть все проще, если бы не был запрещен DISTINCT 😋:

```
SELECT COUNT(DISTINCT НАИМЕНОВАНИЕ) FROM Н_ДИСЦИПЛИНЫ;      //460
```

4.
```
SELECT ФАМИЛИЯ, COUNT(ФАМИЛИЯ) FROM Н_ЛЮДИ WHERE ИД IN(
        SELECT ЧЛВК_ИД FROM Н_УЧЕНИКИ AS students
        JOIN Н_ПЛАНЫ AS plans ON students.ПЛАН_ИД = plans.ИД
        JOIN Н_ОТДЕЛЫ AS otd ON plans.ОТД_ИД = otd.ИД
        WHERE(
                otd.КОРОТКОЕ_ИМЯ = 'КТиУ'
        )
) GROUP BY ФАМИЛИЯ HAVING COUNT(ФАМИЛИЯ) > 50;
```

5. SELECT students.ГРУППА, AVG(AGE(people.ДАТА_РОЖДЕНИЯ)) FROM Н_УЧЕНИКИ AS students

```
JOIN Н_ЛЮДИ AS people ON students.ЧЛВК_ИД = people.ИД
WHERE(
    AGE(people.ДАТА_РОЖДЕНИЯ) = (
        SELECT MIN(AGE(ДАТА_РОЖДЕНИЯ)) FROM Н_ЛЮДИ
        JOIN Н_УЧЕНИКИ ON Н_УЧЕНИКИ.ЧЛВК_ИД = Н_ЛЮДИ.ИД
        WHERE Н_УЧЕНИКИ.ГРУППА = '3100'
        )
)
GROUP BY students.ГРУППА;
```

6. SELECT uch.ГРУППА, people.ИД, people.ИМЯ, people.ФАМИЛИЯ, people.ОТЧЕСТВО, uch.СОСТОЯНИЕ, uch.П_ПРКОК_ИД

```
FROM Н_УЧЕНИКИ AS uch
JOIN Н_ЛЮДИ AS people ON people.ИД = uch.ЧЛВК_ИД
WHERE EXISTS(
    SELECT * FROM Н_УЧЕНИКИ AS uch
    JOIN Н_ПЛАНЫ AS plans ON uch.ПЛАН_ИД = plans.ИД
    JOIN Н_НАПРАВЛЕНИЯ_СПЕЦИАЛ AS napr ON plans.НАПС_ИД = napr.ИД
    JOIN Н_НАПР_СПЕЦ AS spets ON napr.НС_ИД = spets.ИД
    WHERE(
        plans.ДАТА_УТВЕРЖДЕНИЯ > '01.09.2012' AND
        plans.КУРС = '1' AND
        plans.ФО_ИД = '3' AND
        spets.НАИМЕНОВАНИЕ = 'Программная Инженерия'
    )
);
```

7.
```
SELECT COUNT(*) FROM (
    SELECT uch.ЧЛВК_ИД, AVG(ved.ОЦЕНКА::int) AS mark FROM Н_УЧЕНИКИ AS uch
    JOIN Н_ЛЮДИ AS people ON uch.ЧЛВК_ИД = people.ИД
    JOIN Н_ВЕДОМОСТИ AS ved ON ved.ЧЛВК_ИД = people.ИД
    JOIN Н_ПЛАНЫ AS plans ON uch.ПЛАН_ИД = plans.ИД
    JOIN Н_ОТДЕЛЫ AS otd ON plans.ОТД_ИД = otd.ИД
    WHERE(
        ved.ОЦЕНКА ~'[2-5]'AND
        otd.КОРОТКОЕ_ИМЯ = 'КТиУ'
    ) GROUP BY uch.ЧЛВК_ИД
) AS res WHERE(
    mark = 5
);
```

Вывод:

SQL («язык структурированных запросов») — декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных.

Вообще, стандарт SQL выделяет следующие модификаторов JOIN, использующиеся для соединения таблиц:

INNER JOIN – внутреннее объединение таблиц.

LEFT JOIN или LEFT OUTER JOIN – левое внешнее объединение таблиц.

RIGHT JOIN или RIGHT OUTER JOIN – правое внешнее объединение таблиц.

FULL JOIN – полное объединение таблиц.

Подзапрос, Внутренний запрос или Вложенный запрос — это запрос в другом запросе PostgreSQL, встроенный в предложение WHERE.

Подзапрос используется для возврата данных, которые будут использоваться в основном запросе в качестве условия для дальнейшего ограничения данных, подлежащих извлечению.

Подзапросы могут использоваться с операторами SELECT, INSERT, UPDATE и DELETE вместе с такими операторами, как =, <,>,> =, <=, IN и т. д.