Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Отчет

По дисциплине: “Управление данными”

Лабораторная работа №2

“Язык SQL. Генераторы. Триггеры”

Выполнил:

ст.гр. ИС/б-32

Долженко И.А.

Проверил:

Гончаренко Д. Г.

Севастополь

2019

1 ЦЕЛЬ РАБОТЫ

Ознакомится с принципом работы генераторов и триггеров, продемонстрировать работу на примерах.

2 ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Записать запросы, соединяющие две таблицы с помощью JOIN и без него.

2. Записать запросы, соединяющие более двух таблиц с помощью JOIN и без него.

3. Продемонстрировать следующие возможности SQL:

* использование псевдонимов на примере рекурсивного запроса;
* привести пример запроса с подзапросом;
* использование агрегатных функций в подзапросе;
* подзапросы, возвращающие единственное и множественные значения;
* подзапросы, использующие вычисление;
* использование подзапросов в HAVING.

4. Записать запрос, соединяющий таблицу со своей копией.

5. Привести пример коррелированного запроса, использующего две разные таблицы.

6. Продемонстрировать следующие возможности SQL:

* работу оператора EXISTS;
* работу оператора ALL;
* работу оператора ANY.

7. В соответствии с вариантом задания создать генератор и триггер.

8. Изменить значение генератора, в соответствии с хранимыми данными.

9. Ввести данные в таблицу, используя генератор (не менее 5 строк). Просмотреть полученный результат.

10. Внести изменения в указанные таблицы, используя триггеры (не менее 5 строк). Просмотреть полученный результат.

Вариант 12

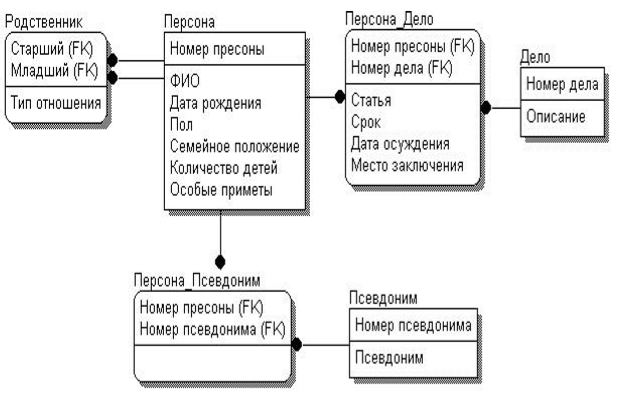


Рисунок 1 – База данных по варианту

3 ХОД РАБОТЫ

1. Запишем запрос, соединяющий две таблицы с помощью JOIN и без него.

С JOIN:

SELECT `персона`.`Номер персоны`, `персона`.`ФИО`, `персона`.`Дата рождения`, `персона`.`Пол`, `персона\_дело`.`Статья`, `персона\_дело`.`Срок`

FROM `персона`

JOIN `персона\_дело` ON `персона`.`Номер персоны` = `персона\_дело`.`Номер персоны`

Без JOIN:

SELECT `персона`.`Номер персоны`, `персона`.`ФИО`, `персона`.`Дата рождения`, `персона`.`Пол`, `персона\_дело`.`Статья`, `персона\_дело`.`Срок`

FROM `персона`, `персона\_дело`

WHERE `персона`.`Номер персоны` = `персона\_дело`.`Номер персоны`



2. Запишем запрос, соединяющий более двух таблиц с помощью JOIN и без него.

С JOIN:

SELECT `персона`.`Номер персоны`, `персона`.`ФИО`, `персона\_дело`.`Статья`, `персона\_дело`.`Срок`, `дело`.`Описание`

FROM `персона`

JOIN `персона\_дело` ON `персона`.`Номер персоны` = `персона\_дело`.`Номер персоны`

JOIN `дело` ON `персона\_дело`.`Номер дела` = `дело`.`Номер дела`

Без JOIN:

SELECT `персона`.`Номер персоны`, `персона`.`ФИО`, `персона\_дело`.`Статья`, `персона\_дело`.`Срок`, `дело`.`Описание`

FROM `персона`, `персона\_дело`, `дело`

WHERE `персона`.`Номер персоны` = `персона\_дело`.`Номер персоны` AND `персона\_дело`.`Номер дела` = `дело`.`Номер дела`



3. Продемонстрируем следующие возможности SQL:

1) Использование псевдонимов на примере рекурсивного запроса.

SELECT one.`Номер персоны`, two.`Номер персоны`, one.`Количество детей`

FROM `Персона` AS one, `Персона` AS two

WHERE one.`Количество детей` = two.`Количество детей` AND one.`Номер персоны` < two.`Номер персоны`



2) Пример запроса с подзапросом.

SELECT \* FROM `персона`

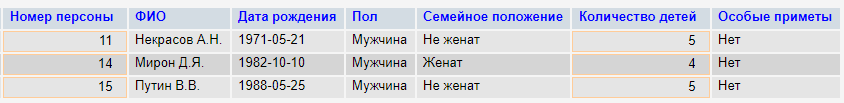
WHERE `Номер персоны` IN (

SELECT `Номер персоны`

FROM `персона`

WHERE `Количество детей` > 3

)



3) Использование агрегатных функций в подзапросе.

SELECT \* FROM `персона`

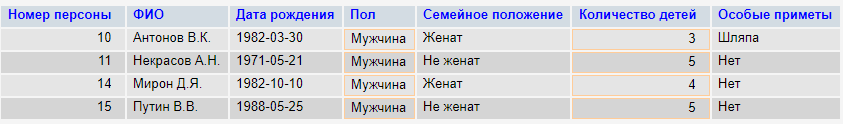
WHERE `Количество детей` > (

SELECT AVG(`Количество детей`)

FROM `персона`

WHERE `Пол` = 'Мужчина'

)



4) Подзапросы, возвращающие единственное и множественные значения.

Единственное значение:

SELECT `ФИО` FROM `персона`

WHERE `Количество детей` > (

SELECT AVG(`Количество детей`)

FROM `персона`

)



Множественные значения:

SELECT `Номер персоны`, `ФИО`, `Дата рождения`

FROM `персона`

WHERE `Дата рождения` < ALL (

SELECT `Дата осуждения`

FROM `персона` JOIN `персона\_дело` ON `персона`.`Номер персоны` = `персона\_дело`.`Номер персоны`

)



5) Подзапросы, использующие вычисление.

SELECT \* FROM `персона`

WHERE `Номер персоны` IN (

SELECT `Номер персоны`

FROM `персона`

WHERE `Количество детей` % 2 = 1

)



6) Использование подзапросов в HAVING.

SELECT `Номер дела`, AVG(`Срок`) as 'Средний срок'

FROM `персона\_дело`

GROUP BY `Номер дела`

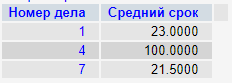
HAVING AVG(`Срок`) > (

SELECT AVG(`Срок`)

FROM `персона\_дело`

WHERE `Номер дела` = 3

)

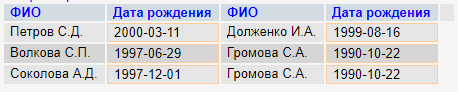


4. Запишем запрос, соединяющий таблицу со своей копией.

SELECT п1.`ФИО`, п1.`Дата рождения`, п2.`ФИО`, п2.`Дата рождения`

FROM `персона` п1, `персона` п2

WHERE п1.`Пол` = п2.`Пол` AND п2.`Количество детей` = 0 AND п1.`Дата рождения` > п2.`Дата рождения`



5. Пример коррелированного запроса, использующего две разные таблицы.

SELECT `ФИО`, `Дата рождения`, `Пол`

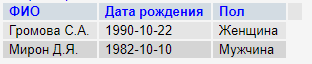
FROM `персона` П WHERE 'Киев' IN (

SELECT `Место заключения`

FROM `персона\_дело` Д

WHERE Д.`Номер персоны` = П.`Номер персоны`

)



6. Продемонстрируем следующие возможности SQL:

1) Работа оператора EXISTS.

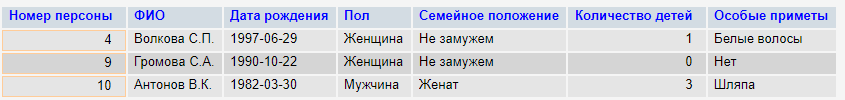
SELECT \* FROM `Персона`

WHERE EXISTS(

SELECT \* FROM `Персона\_дело`

WHERE `Персона\_дело`.`Номер персоны` = `Персона`.`Номер персоны` AND `Персона\_дело`.`Срок`/5 < 3

)



2) Работа оператора ALL.

SELECT \* FROM `Персона`

WHERE `Количество детей` > ALL (

SELECT `Количество детей`

FROM `Персона`

WHERE `Номер персоны` < 4

)



3) Работа оператора ANY.

SELECT \* FROM `Персона`

WHERE `Персона`.`Номер персоны` = ANY (

SELECT `Персона\_псевдоним`.`Номер персоны`

FROM `Персона\_псевдоним`

)



7. Создать генераторы для полей «Номер персоны», «Номер дела», «Номер псевдонима». Автоматическая генерация полей «Номер персоны», «Номер дела», «Номер псевдонима». Создать обновляемое представление «Псевдонимы», состоящее из полей «Номер персоны», «ФИО», «Псевдоним».

Создадим представление:

CREATE VIEW Псевдонимы\_Представление AS

SELECT `персона`.`Номер персоны`, `персона`.`ФИО`, `псевдоним`.`Псевдоним`

FROM `псевдоним`, `персона`, `персона\_псевдоним`

WHERE `персона`.`Номер персоны` = `персона\_псевдоним`.`Номер персоны` AND `персона\_псевдоним`.`Номер псевдонима` = `псевдоним`.`Номер псевдонима`