Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Институт информационных технологий

Кафедра ИС

# ОТЧЕТ

по лабораторной работе №1

МАНИПУЛИРОВАНИЕ БАЗОЙ ДАННЫХ. РЕЛЯЦИОННАЯ АЛГЕБРА И ЯЗЫК SQL

Выполнил:

ст. гр. ИС/б-21-2-о

Мовенко К. М.

Проверила:

Абрамович А.Ю.

Севастополь

2023

# ЦЕЛЬ РАБОТЫ

Изучить основы реляционной алгебры, как базового средства манипулирования. Выработать у обучающихся практические навыки по работе с реляционными базами данных и представлению запросов как на языке реляционной алгебре, так и SQL.

# ВАРИАНТ ЗАДАНИЯ

Проанализировать схему БД своего варианта задания (Рисунок 2.1), выделить и классифицировать все существующие связи, определить необходимые ограничения целостности.

Создать базу данных и все её таблицы. Особое внимание уделить описанию первичного ключа, значений по умолчанию, описателям NOT NULL и конструкции CHECK. Установить связи между таблицами.



Рисунок 2.1 – Вариант 12, схема базы данных

# ХОД РАБОТЫ

В СУБД Firebird была создана база данных labs.fdb, в которой затем проводилась вся представленная в отчёте работа.

Далее в ней была создана таблица Person, содержащая информацию о личностях заключённых. Далее таблица была заполнена рядом кортежей с помощью команды insert (Рисунок 3.1).

create table person(

p\_id integer not null (check p\_id > 0),

primary key(p\_id),

full\_name varchar(30) not null,

birth\_date date not null,

sex char check (sex in (‘M’, ‘F’)) not null,

mar\_stat varchar(12) check (mar\_stat in (‘married’, ‘not married’)) not null,

n\_child integer not null,

features varchar(30)

);

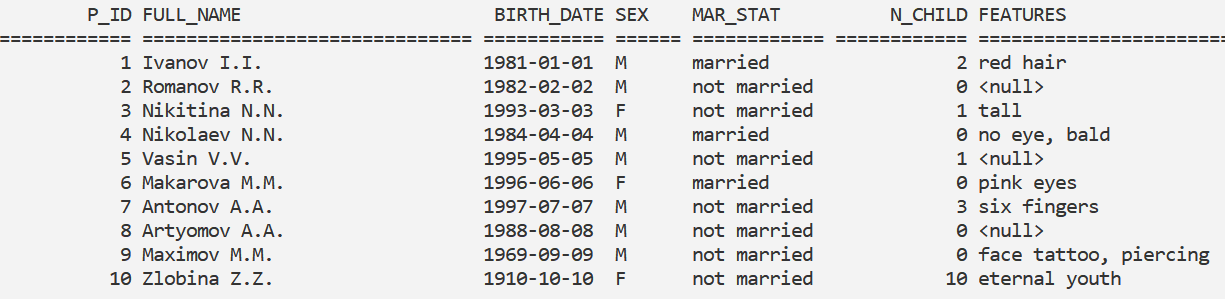


Рисунок 3.1 – Таблица Person

Далее была создана таблица Cases, содержащая информацию об уголовных делах заключённых. Затем она была заполнена набором кортежей (Рисунок 3.2).

create table cases(

c\_id integer not null check (c\_id > 0),

primary key (c\_id),

descr varchar(30) not null

);

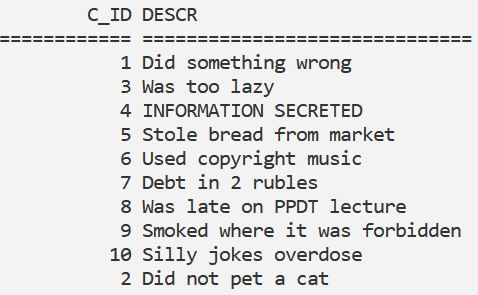


Рисунок 3.2 – Таблица Cases

Была создана и заполнена таблица Alias, содержащая информацию о псевдонимах заключённых (Рисунок 3.3).

create table alias(

a\_id integer not null check (a\_id > 0),

primary key (a\_id);

alias varchar(15) not null

);

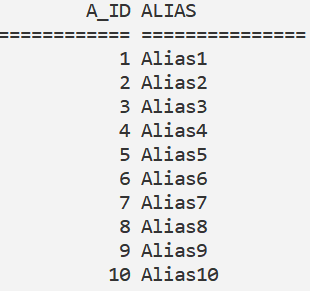


Рисунок 3.3 – Таблица Alias

Была создана таблица Relatives, связывающая людей, указанных в таблице Person, различными родственными связями. Данная таблица была заполнена рядом значений (Рисунок 3.4).

create table relatives(

eld\_id int not null,

jun\_id int not null,

rel\_type varchar(20) not null,

primary key (eld\_id, jun\_id),

foreign key (eld\_id) references person(p\_id) on delete cascade,

foreign key (jun\_id) references person(p\_id) on delete cascade

);

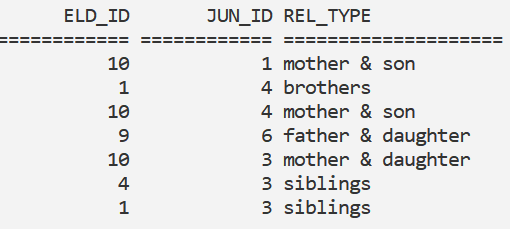


Рисунок 3.4 – Таблица Relatives

Была создана таблица Person\_Case, связывающая людей из таблицы Person с уголовными делами из таблицы Cases. Затем она была заполнена рядом кортежей (Рисунок 3.5).

create table person\_case(

p\_id int not null,

c\_id int not null,

clause varchar(30) not null,

term int check (term > 0),

conv\_date date not null,

conv\_place varchar(30),

primary key (p\_id, c\_id),

foreign key (p\_id) references person(p\_id) on delete cascade,

foreign key (c\_id) references cases(c\_id) on delete cascade

);

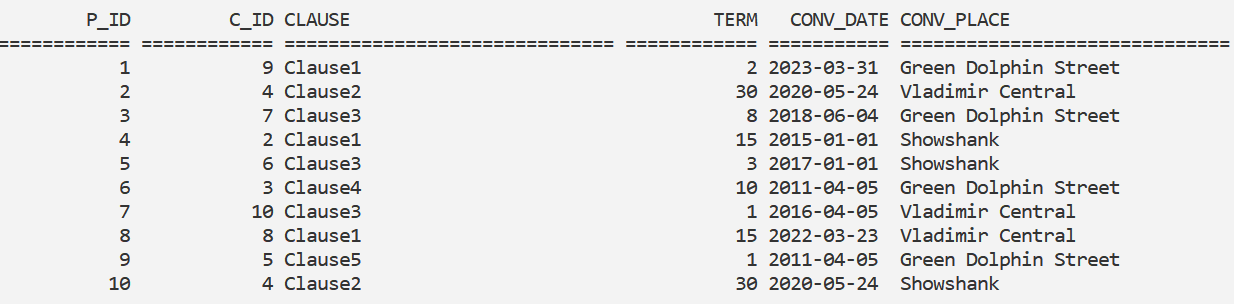


Рисунок 3.5 – Таблица Person\_Case

Была создана таблица Person\_Alias, связывающая людей из таблицы Person с их псевдонимами из таблицы Alias. Затем она была заполнена рядом кортежей (Рисунок 3.6).

create table person\_alias(

p\_id int not null,

a\_id int not null,

foreign key (p\_id) references person(p\_id) on delete cascade,

foreign key (a\_id) references alias(a\_id) on delete cascade,

primary key (p\_id, a\_id)

);

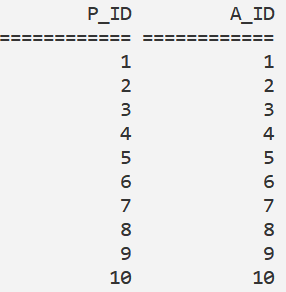


Рисунок 3.6 – Таблица Person\_Alias

Был протестирован принцип каскадного удаления, который поддерживает целостность информации в таблицах, связанных внешними ключами. Из таблицы Alias был удален кортеж с ID=10, вследствие чего был удалён соответствующий ему кортеж из таблицы Person\_Alias (Рисунок 3.7).

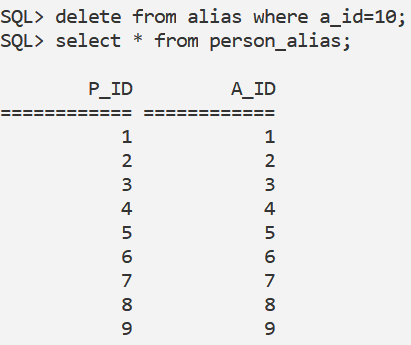


Рисунок 3.7 – Каскадное удаление

Был сделан запрос с простым вычислением и агрегатной функцией AVG. Так было выведено среднее количество дней, составляющих срок заключения (Рисунок 3.8).

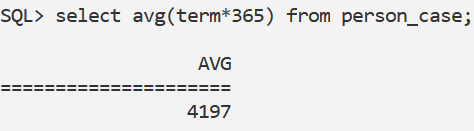


Рисунок 3.8 – Запрос с простым вычислением и агрегатной функцией

Для демонстрации работы предложения GROUP BY был сделан запрос на вывод максимального числа детей у одного человека среди женатых и неженатых лиц (Рисунок 3.9).

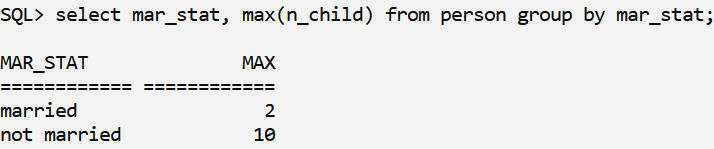


Рисунок 3.9 – Запрос с предложением GROUP BY

Для демонстрации работы предложения HAVING был сделан запрос на вывод среднего срока заключения для тюрем, указанных в таблице менее четырёх раз (Рисунок 3.10).

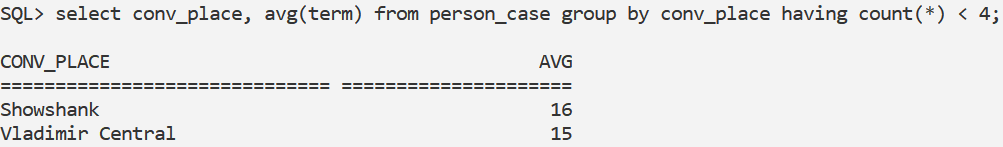


Рисунок 3.10 – Запрос с HAVING

Был совершён запрос, включающий в себя операции селекции и соединения. Требовалось выбрать из таблицы Person заключённых мужчин и через таблицу Person\_Alias соединить их имена с соответствующими псевдонимами (Рисунок 3.11).

select P.p\_id, P.full\_name, A.alias

from (person P join person\_alias PA on P.p\_id = PA.p\_id)

join alias A on PA.a\_id = A.a\_id

where P.sex = 'M';

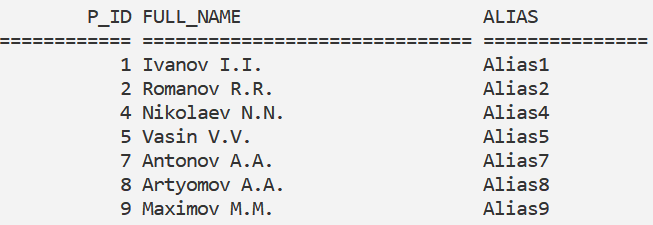


Рисунок 3.11 – Результат выполнения запроса с селекцией и соединением

Был сделан запрос, включающий в себя операции проекции и деления: вывести из таблицы Person\_Case все уголовные статьи, для которых есть заключённые в каждой из представленных в столбце conv\_place тюрем (Рисунок 3.12). Таблица Person\_Case делится на проекцию своего атрибута.

select distinct(clause) from person\_case A

where not exists

(select distinct(conv\_place) from person\_case B

where not exists

(select \* from person\_case C

where B.conv\_place = C.conv\_place and A.clause = C.clause));

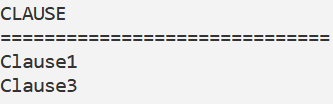


Рисунок 3.12 – Результат выполнения запроса с проекцией и делением

Был написан запрос, включающий в себя операции проекции, объединения и пересечения: вывести ID всех заключённых, для которых указаны родственники (старшие или младшие) и сопоставить из со списком ID вех заключённых (Рисунок 3.13). Вторая часть вопроса не имеет смысла, но реализует принцип операции пересечения.

select p\_id from person

where p\_id in

(select eld\_id from relatives

union select jun\_id from relatives);

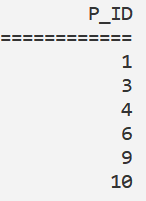


Рисунок 3.13 – Результат запроса с проекцией, объединением и пересечением

Был написан запрос, включающий операции соединения и деления: вывести из таблицы Person\_Case все уголовные статьи и посаженных по ним заключённых (с именами), где уголовная статья представлена для каждой из указанных тюрем (Рисунок 3.14).

select P.p\_id, P.full\_name, PC.clause

from person\_case PC join person P on (PC.p\_id = P.p\_id)

where not exists

(select distinct(conv\_place) from person\_case PC1

where not exists

(select \* from person\_case PC2

where PC1.conv\_place = PC2.conv\_place and PC.clause = PC2.clause));

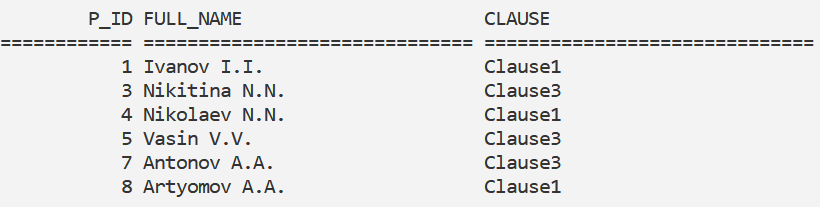


Рисунок 3.14 – Результат запроса с операциями деления и соединения

Был сделан запрос, включающий операции вычитания и объединения: вывести людей из таблицы Person, для которых не указаны родственники в таблице Relatives (Рисунок 3.15).

select p\_id, full\_name from person

where p\_id not in

(select eld\_id from relatives

union

select jun\_id from relatives);

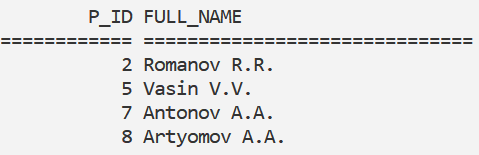


Рисунок 3.15 – Запрос с операциями вычитания и объединения

Был написан запрос SQL, не реализующийся языком реляционной алгебры. В качестве примера было выведено среднее количество детей у заключённых, сгруппированное по семейному положению и полу заключённых (Рисунок 3.16).

select sex, mar\_stat, avg(n\_child) from person

group by mar\_stat, sex;

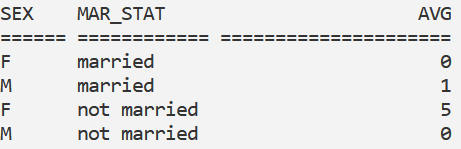


Рисунок 3.16 – Выполнение запроса с агрегатной функцией и группировкой

# ВЫВОД

В ходе работы были изучены основы реляционной алгебры, ряд её основополагающих операций и формирование через них запросов к базе данных. Данные операции были реализованы на языке запросов SQL.

Также была проведена отработка навыков проектирования баз данных и формирования запросов к ним средствами СУБД и языка SQL.