Министерство науки и высшего образования Российской Федерации

федеральное государственное бюджетное образовательное учреждение высшего образования

“Российский экономический университет имени Г.В. Плеханова”

**МОСКОВСКИЙ ПРИБОРОСТРОИТЕЛЬНЫЙ ТЕХНИКУМ**

Специальность: 09.02.07 Информационные системы и программирование  
Квалификация: программист

ПРАКТИЧЕСКИЕ РАБОТЫ ПО ДИСЦИПЛИНЕ

«ОСНОВЫ ПРОЕКТИРОВАНИЯ БАЗ ДАННЫХ»

Листов:9

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Выполнил студент  Группы: П50-4-21  Игошев Ростислав Вадимович |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | Проверил преподаватель  \_\_\_\_\_\_\_\_\_\_\_А.В. Архангельский  «\_\_\_\_\_\_» \_\_\_\_\_\_2023 года |

Москва 2023

ПРАКТИЧЕСКАЯ РАБОТА №2

Тема: «Физические и логические модели данных»

Цель: Научиться работать с физическими и логическими моделями данных. Выбрать себе предметную область и записать ее в таблицу (повторяющихся быть не должно). Создать data и info-логические модели к выбранной предметной области.

Для начала, разберем и создадим логическую модель данных, ибо по ней проще ориентироваться обычному человеку.

Логическая модель:

В данной базе данных присутствует основная таблица, от которой идут все связи к остальным, образуя связь один ко многим.

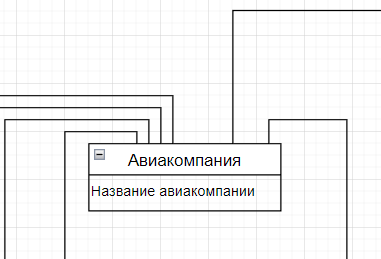


Рисунок – Основная таблица

Первая ветка, которая исходит от основной таблицы – аэропорты. Тут присутствует различная информация о аэропортах, что ясно из названия.

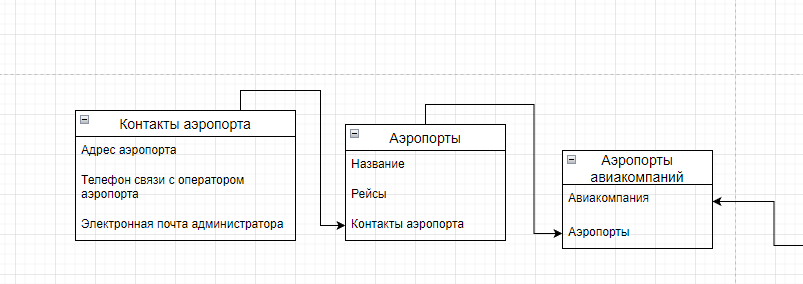


Рисунок - Ветка аэропортов

Следующая ветка – самолеты. Здесь описаны характеристики каждого отдельного самолета.

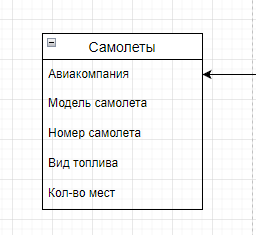


Рисунок - Ветка самолетов

Третья ветка – вип-персоны. Здесь описана информация о вип-персонах в авиакомпаниях, их статус, приоритет, личная информация.

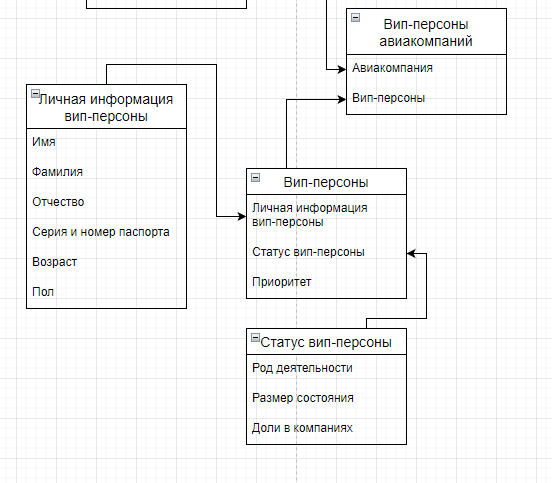


Рисунок - Ветка вип-персон

Далее – ветка сотрудников. Здесь описана их личная информация, служебное положение, контактные данные.

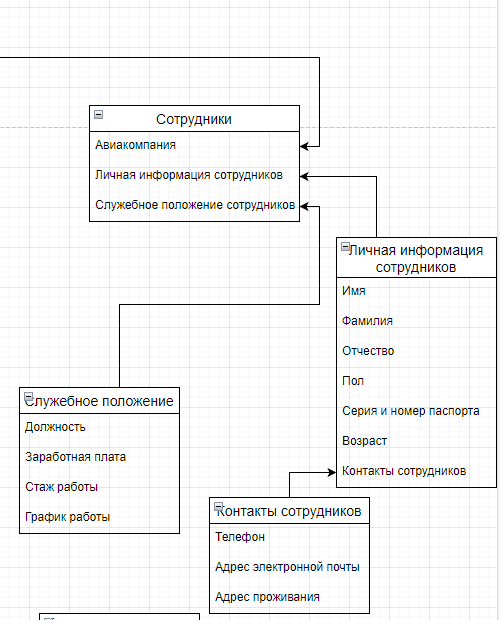


Рисунок - Ветка сотрудников

Последние две ветки – ветка дочерних компаний и ветка турагентств – партнеров. Здесь описаны названия, рейсы и прочая информация о данных отделах.

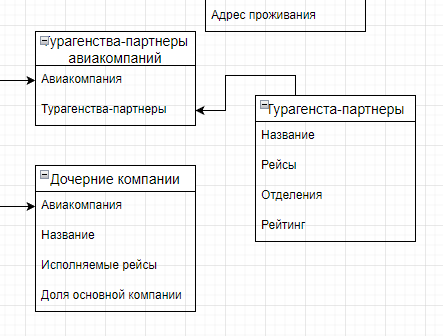


Рисунок - Ветка дочерних компаний и ветка турагентств-партнеров

В целом, наша конечная модель выглядит так:

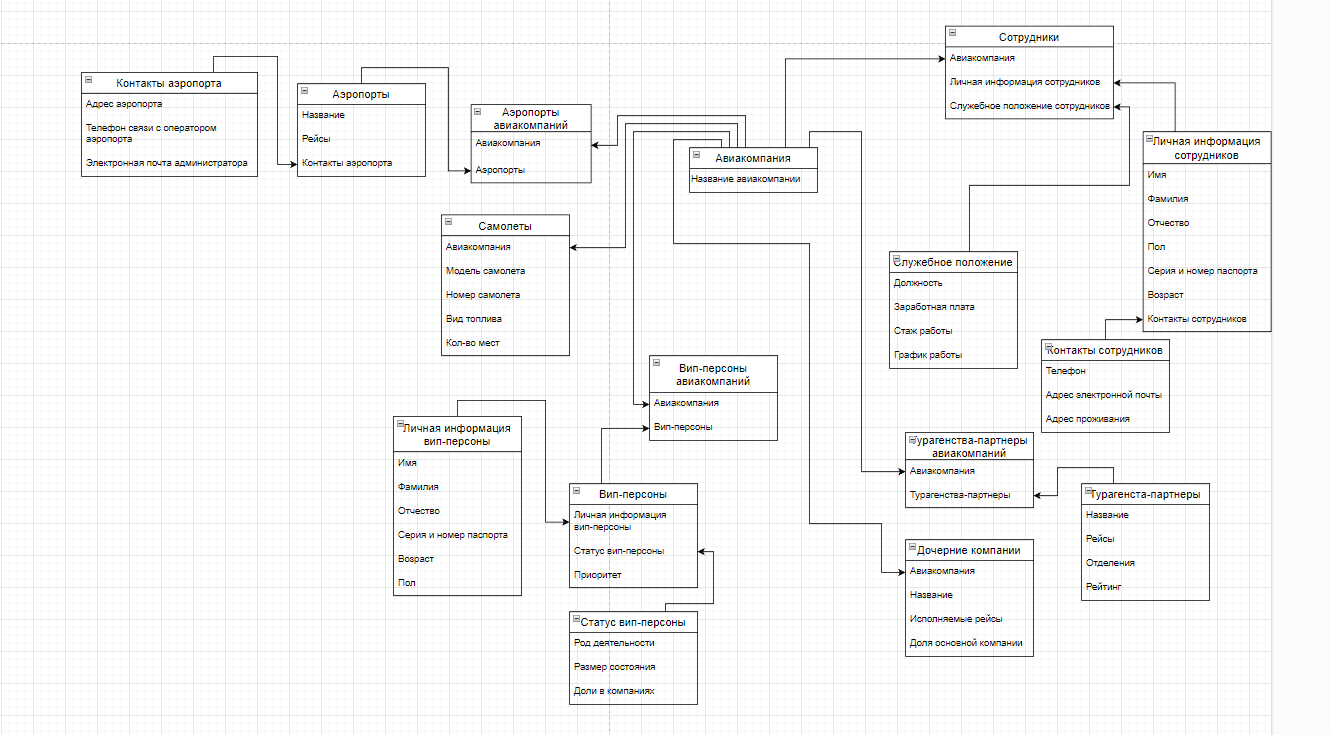


Рисунок - Целая итоговая модель данных

Физическая модель:

Теперь, создадим физическую модель на основе логической. Отличается физическая тем, что названия должны быть понятны СУБД, что расшифровывается как Система Управления Базами Данных. Грубо говоря, названия должны быть на английском, применяться в стиле snake\_case и указывать в себе тип данных с ограничением и возможностью пустых ячеек, то-есть значения NULL. Кроме того, связь должна быть организована между id таблиц.

Первая таблица выглядит так же, за исключением того, что в ней появилась ячейка id и поменялся формат записи.

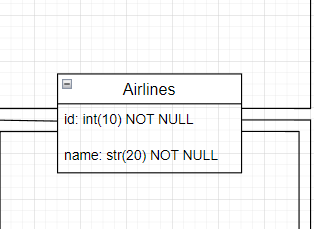


Рисунок - Основная таблица физической модели

Первая ветка:

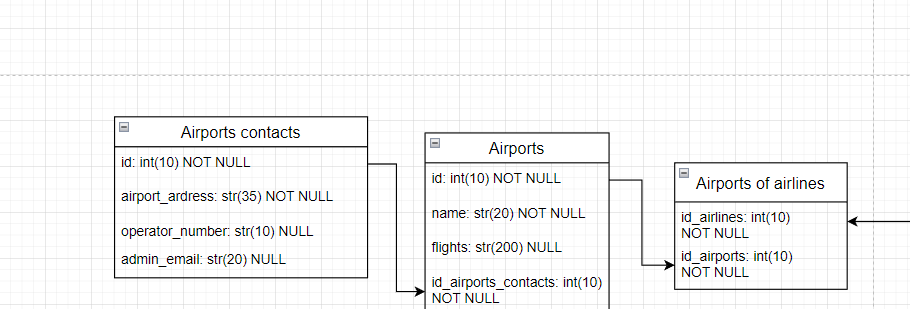


Рисунок - Ветка Airports

Ветка самолетов:

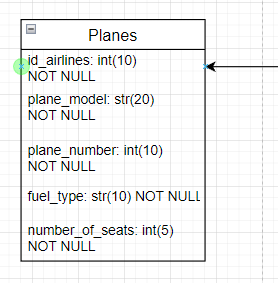


Рисунок - Ветка Planes

Ветка вип-персон:

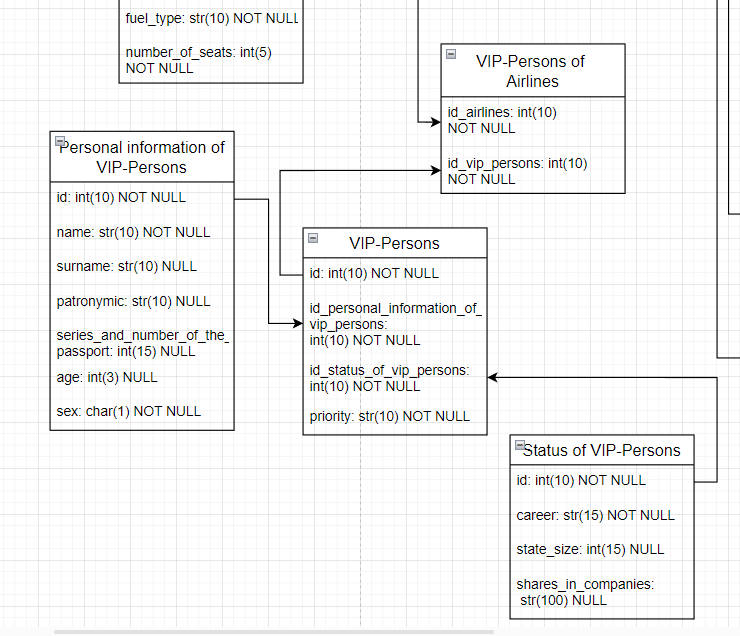


Рисунок - Ветка VIP – Persons

Ветка рабочих:

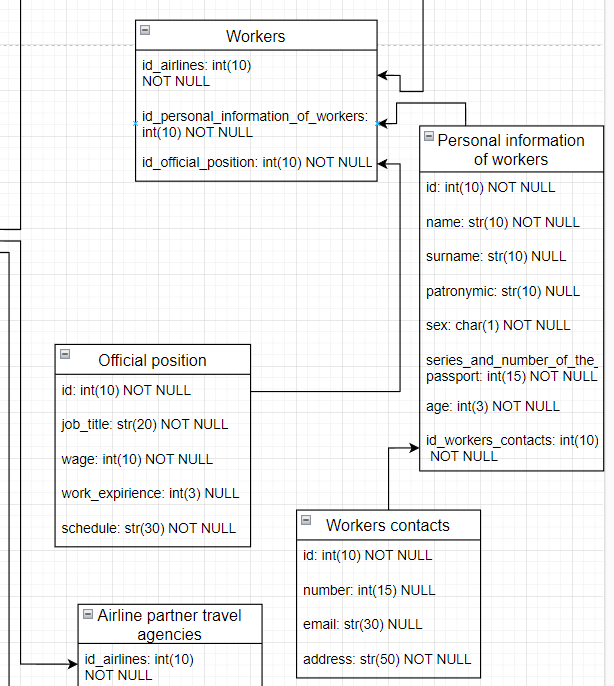


Рисунок - Ветка Workers

Ветки турагентств и дочерних компаний:

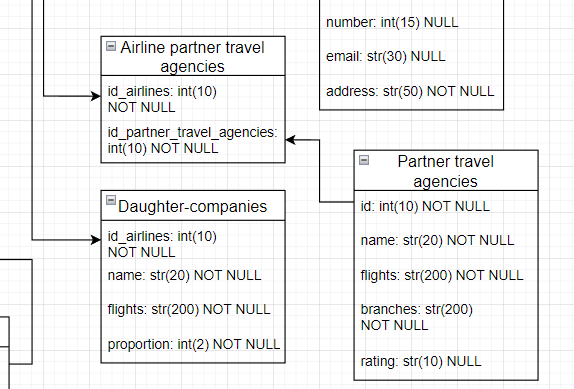


Рисунок 13 - Ветка Daughter-companies и ветка Airline partner travel agencies

После таких изменений, видим следующую картину:

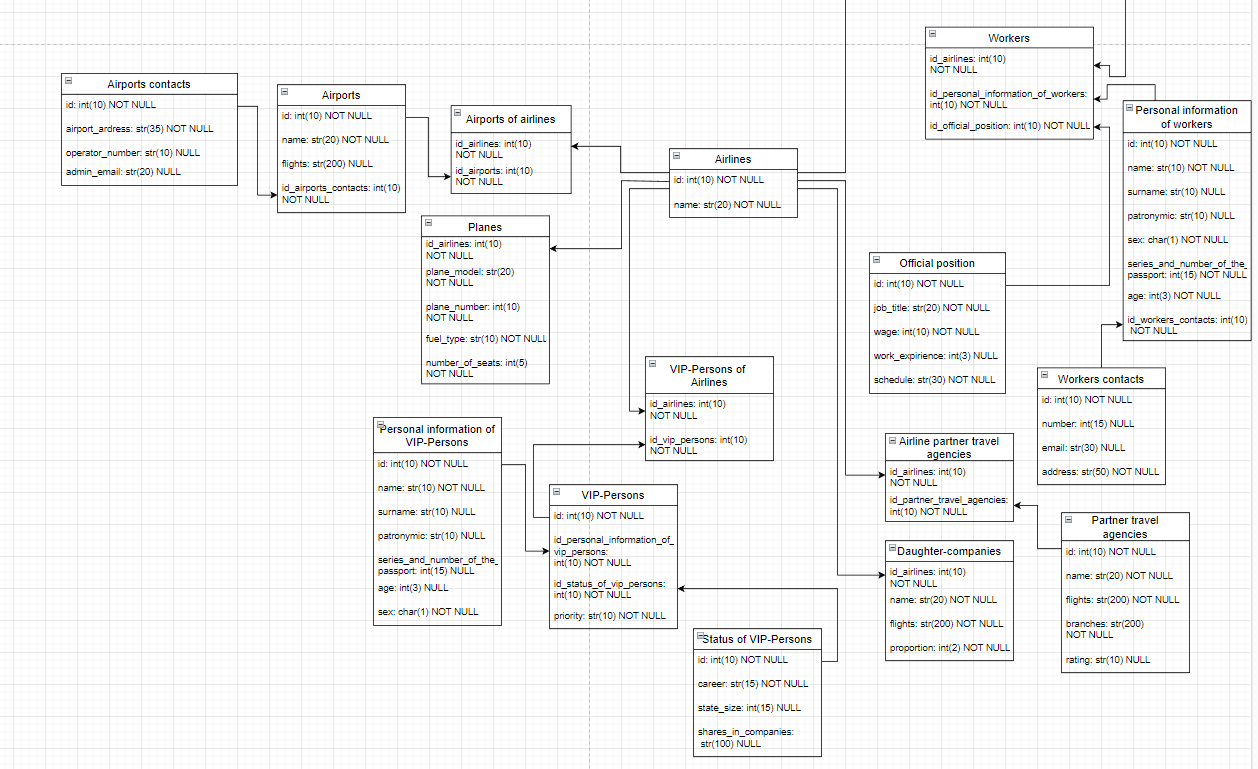


Рисунок - Целая физическая модель

Вывод: Научились работать с физическими и логическими моделями данных. Выбрали себе предметную область и записали ее в таблицу (повторяющихся быть не должно). Создали data и info-логические модели к выбранной предметной области.

ПРАКТИЧЕСКАЯ РАБОТА №3

Тема: «Создание БД»

Цель работы: создать базу данных при помощи SQL запросов на основе физической и логической моделей из предыдущей практической.

Ссылка на код работы:

<https://drive.google.com/file/d/1CMfxVjPVC4FOrupjGv2R7JQHTYoTuoa/view>.

Начнем создавать нашу БД. Для взаимодействия с ней следует зарегистрировать суперпользователя. Попробуем создать БД следующей командой запросов:



Рисунок 15 – Создание БД

У БД есть несколько параметров, которые определяют ее характеристики и особенности взаимодействия с ней. Вот они:

* Logical Name: логическое имя, которое присваивается файлу базы данных.
* File Type: есть несколько типов файлов, но, как правило, основная работа ведется с файлами данных (ROWS Data) и файлом лога (LOG)
* Filegroup: обозначает группу файлов. Группа файлов может хранить множество файлов и может использоваться для разбиения базы данных на части для размещения в разных местах.
* Initial Size (MB): устанавливает начальный размер файлов при создании (фактический размер может отличаться от этого значения).
* Autogrowth/Maxsize: при достижении базой данных начального размера SQL Server использует это значение для увеличения файла.
* Path: каталог, где будут храниться базы данных.
* File Name: непосредственное имя физического файла. Если оно не указано, то применяется логическое имя.

Теперь, попробуем выделить нашу созданную БД для работы с ней.



Рисунок 16 – Использование БД

Как видим, в поле активной БД отобразилась ранее нами созданная БД.



Рисунок 17 – Демонстрация работы с БД

Для создания таблицы нам следует использовать язык определения данных (DDL - Data Definition Language). Создание, редактирование и удаление информации. В эту группу входят следующие операторы:

CREATE – используется для создания объектов базы данных;

ALTER – используется для изменения объектов базы данных;

DROP – используется для удаления объектов базы данных.

Вот типы данных которые будут использоваться в этой работе:

TINYINT: хранит числа от 0 до 255. Хорошо подходит для хранения небольших чисел.

SMALLINT: хранит числа от –32 768 до 32 767.

INT: хранит числа от –2 147 483 648 до 2 147 483 647. Наиболее используемый тип для хранения чисел.

MONEY: хранит дробные значения от -922 337 203 685 477.5808 до 922 337 203 685 477.5807. Представляет денежные величины.

VARCHAR: хранит строку. Можно указать конкретную длину для столбца - от 1 до 8 000 символов.

Теперь, попробуем создать центральную таблицу нашей БД с полем и первичным ключем. Для этого нам нужно написать следующую конструкцию в случае с ID: (name) INT IDENTITY (seed, increment) PRIMARY KEY. В случае с переменной: (name) (TYPE)(limit) (NULL/NOT NULL).

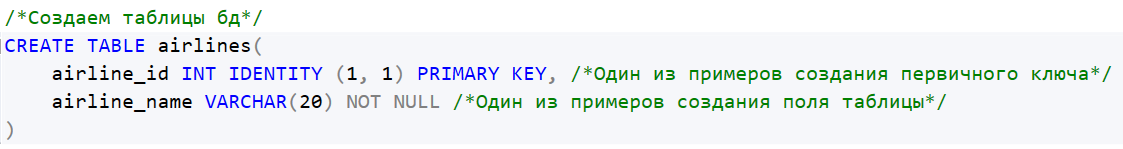


Рисунок 18 – Создание таблицы и ее полей и первичного ключа

Выделив кусок кода и запустив его, наша таблица будет создана. Чтобы регулировать связь между таблицами недостаточно лишь первичного ключа, нужен еще и вторичный. Вот пример создания одного из внешних ключей:



Рисунок 19 – Создание внешнего ключа

В следующем блоке кода используем конструкцию ALTER. Она выполняет функции изменения данных какого-либо объекта.

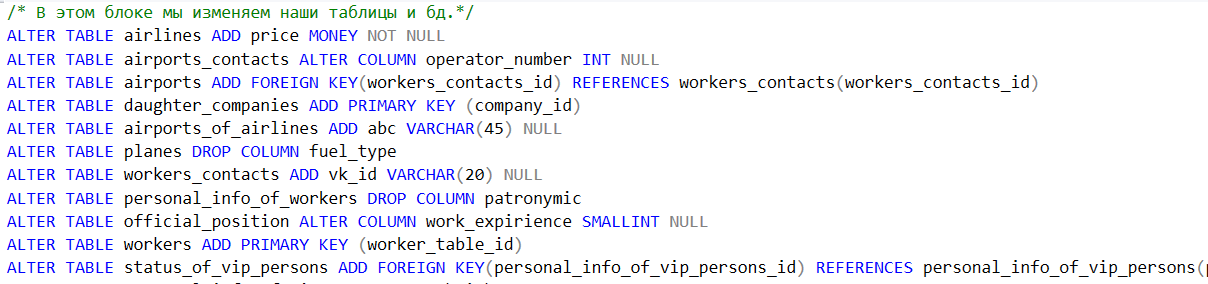


Рисунок 20 – Изменение таблиц и БД

Этот блок отвечает за удаление таблиц и баз данных.

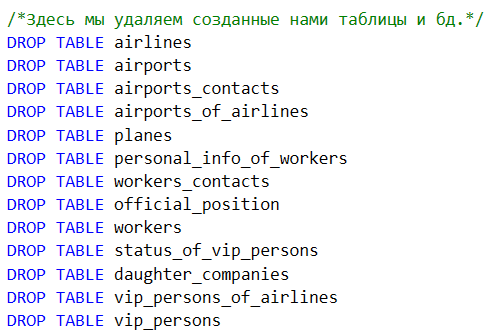


Рисунок 21 – Удаление таблиц и БД

Включив диаграмму, мы увидим следующую картину:

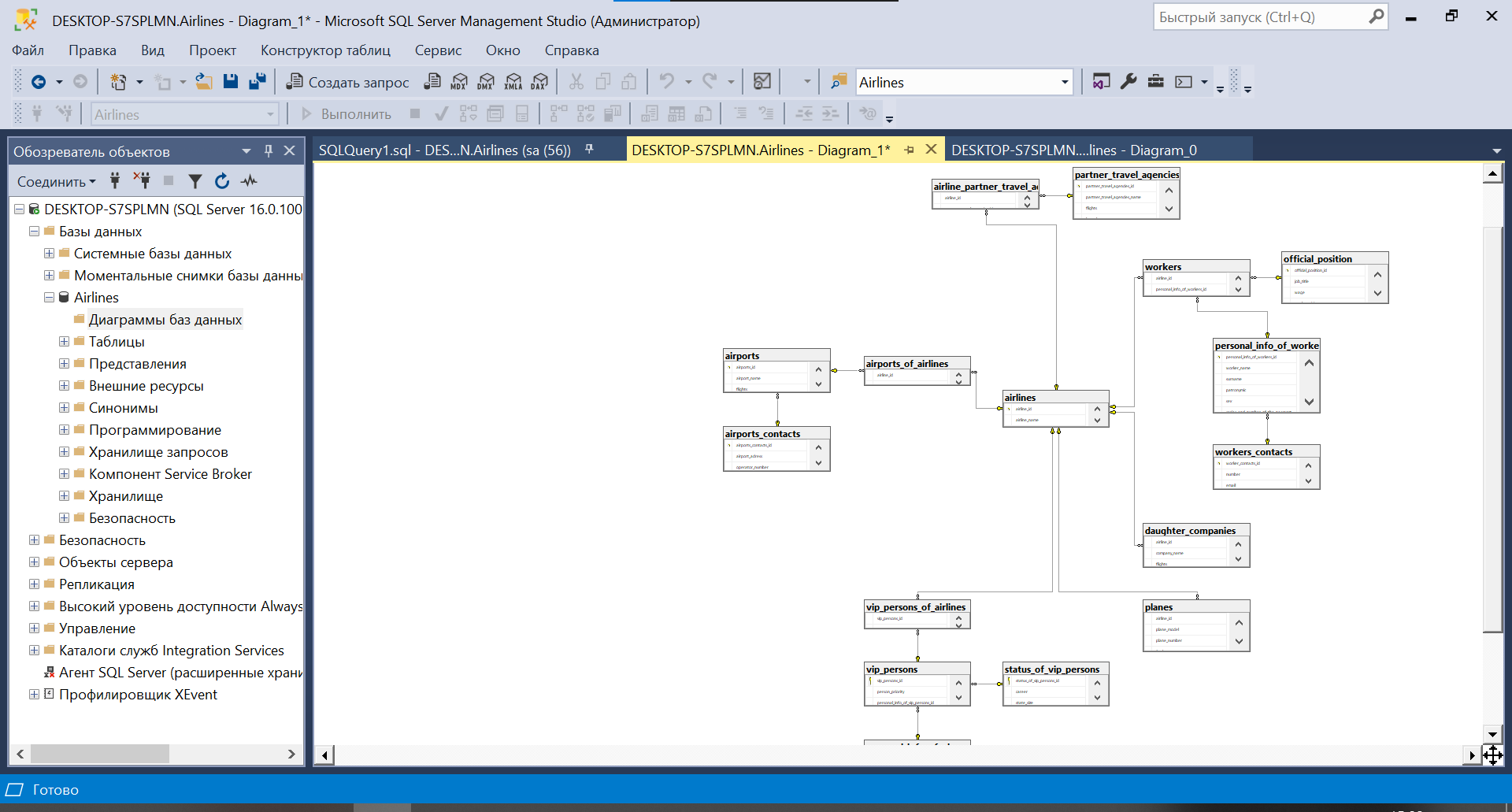


Рисунок – Диаграмма БД

Как видим, наша БД была успешно создана и воспроизведена графическим образом.

Ниже представлен скрипт БД:

/\*Создаем бд\*/

CREATE DATABASE Airlines

/\*Начинаем работать с созданной бд\*/

USE Airlines

/\*Создаем таблицы бд\*/

CREATE TABLE airlines(

airline\_id INT IDENTITY (1, 1) PRIMARY KEY, /\*Один из примеров создания первичного ключа\*/

airline\_name VARCHAR(20) NOT NULL /\*Один из примеров создания поля таблицы\*/

)

CREATE TABLE airports\_contacts(

airports\_contacts\_id INT IDENTITY (1, 1) PRIMARY KEY,

airport\_adress VARCHAR(35) NOT NULL,

operator\_number VARCHAR(10) NULL,

admin\_email VARCHAR(20) NULL

)

CREATE TABLE airports(

airports\_id INT IDENTITY (1,1) PRIMARY KEY,

airport\_name VARCHAR(20) NOT NULL,

flights VARCHAR(200) NULL,

airports\_contacts\_id INT,

FOREIGN KEY (airports\_contacts\_id) /\*Указание поля, использующегося в качестве внешнего ключа\*/

REFERENCES airports\_contacts(airports\_contacts\_id) /\*Указание расположения первичного ключа для образования связи\*/

)

CREATE TABLE airports\_of\_airlines(

airline\_id INT,

airports\_id INT,

FOREIGN KEY (airline\_id)

REFERENCES airlines(airline\_id),

FOREIGN KEY (airports\_id)

REFERENCES airports(airports\_id)

)

CREATE TABLE planes(

airline\_id INT,

plane\_model VARCHAR(20) NOT NULL,

plane\_number INT NOT NULL,

fuel\_type VARCHAR(10) NOT NULL,

number\_of\_seats SMALLINT NOT NULL

FOREIGN KEY (airline\_id)

REFERENCES airlines(airline\_id),

)

CREATE TABLE workers\_contacts(

worker\_contacts\_id INT IDENTITY (1, 1) PRIMARY KEY,

number INT NULL,

email VARCHAR(30) NULL,

adress VARCHAR(50) NOT NULL

)

CREATE TABLE personal\_info\_of\_workers(

personal\_info\_of\_workers\_id INT IDENTITY (1, 1) PRIMARY KEY,

worker\_name VARCHAR(10) NOT NULL,

surname VARCHAR (10) NULL,

patronymic VARCHAR(10) NULL,

sex VARCHAR(1) NOT NULL,

series\_and\_number\_of\_the\_passport INT NOT NULL,

age INT NOT NULL,

worker\_contacts\_id INT,

FOREIGN KEY (worker\_contacts\_id)

REFERENCES workers\_contacts(worker\_contacts\_id),

)

CREATE TABLE official\_position(

official\_position\_id INT IDENTITY (1, 1) PRIMARY KEY,

job\_title VARCHAR(20) NOT NULL,

wage INT NOT NULL,

work\_expirience TINYINT NULL,

schedule VARCHAR(30) NOT NULL

)

CREATE TABLE workers(

airline\_id INT,

personal\_info\_of\_workers\_id INT,

official\_position\_id INT

FOREIGN KEY (airline\_id)

REFERENCES airlines(airline\_id),

FOREIGN KEY (personal\_info\_of\_workers\_id)

REFERENCES personal\_info\_of\_workers(personal\_info\_of\_workers\_id),

FOREIGN KEY (official\_position\_id)

REFERENCES official\_position(official\_position\_id),

)

CREATE TABLE status\_of\_vip\_persons(

status\_of\_vip\_persons\_id INT IDENTITY (1, 1) PRIMARY KEY,

career VARCHAR(15) NOT NULL,

state\_size INT NULL,

shares\_in\_companies VARCHAR(100) NULL

)

CREATE TABLE personal\_info\_of\_vip\_persons(

personal\_info\_of\_vip\_persons\_id INT IDENTITY (1, 1) PRIMARY KEY,

person\_name VARCHAR(10) NOT NULL,

surname VARCHAR(10) NULL,

patronymic VARCHAR(10) NULL,

series\_and\_number\_of\_the\_passport INT NULL,

age INT NULL,

sex VARCHAR(1) NOT NULL

)

CREATE TABLE vip\_persons(

vip\_persons\_id INT IDENTITY (1, 1) PRIMARY KEY,

person\_priority VARCHAR(10) NOT NULL,

personal\_info\_of\_vip\_persons\_id INT,

status\_of\_vip\_persons\_id INT,

FOREIGN KEY (status\_of\_vip\_persons\_id)

REFERENCES status\_of\_vip\_persons(status\_of\_vip\_persons\_id),

FOREIGN KEY (personal\_info\_of\_vip\_persons\_id)

REFERENCES personal\_info\_of\_vip\_persons(personal\_info\_of\_vip\_persons\_id),

)

CREATE TABLE vip\_persons\_of\_airlines(

vip\_persons\_id INT,

airline\_id INT,

FOREIGN KEY (vip\_persons\_id)

REFERENCES vip\_persons(vip\_persons\_id),

FOREIGN KEY (airline\_id)

REFERENCES airlines(airline\_id)

)

CREATE TABLE daughter\_companies(

airline\_id INT,

company\_name VARCHAR(20) NOT NULL,

flights VARCHAR(200) NOT NULL,

proportion INT NOT NULL,

FOREIGN KEY (airline\_id)

REFERENCES airlines(airline\_id)

)

CREATE TABLE partner\_travel\_agencies(

partner\_travel\_agencies\_id INT IDENTITY (1, 1) PRIMARY KEY,

partner\_travel\_agencies\_name VARCHAR(20) NOT NULL,

flights VARCHAR(200) NOT NULL,

branches VARCHAR(200) NOT NULL,

rating VARCHAR(10) NULL

)

CREATE TABLE airline\_partner\_travel\_agencies(

airline\_id INT,

partner\_travel\_agencies\_id INT,

FOREIGN KEY (airline\_id)

REFERENCES airlines(airline\_id),

FOREIGN KEY (partner\_travel\_agencies\_id)

REFERENCES partner\_travel\_agencies(partner\_travel\_agencies\_id)

)

/\* В этом блоке мы изменяем наши таблицы и бд.\*/

ALTER TABLE airlines ADD price MONEY NOT NULL

ALTER TABLE airports\_contacts ALTER COLUMN operator\_number INT NULL

ALTER TABLE airports ADD FOREIGN KEY(workers\_contacts\_id) REFERENCES workers\_contacts(workers\_contacts\_id)

ALTER TABLE daughter\_companies ADD PRIMARY KEY (company\_id)

ALTER TABLE airports\_of\_airlines ADD abc VARCHAR(45) NULL

ALTER TABLE planes DROP COLUMN fuel\_type

ALTER TABLE workers\_contacts ADD vk\_id VARCHAR(20) NULL

ALTER TABLE personal\_info\_of\_workers DROP COLUMN patronymic

ALTER TABLE official\_position ALTER COLUMN work\_expirience SMALLINT NULL

ALTER TABLE workers ADD PRIMARY KEY (worker\_table\_id)

ALTER TABLE status\_of\_vip\_persons ADD FOREIGN KEY(personal\_info\_of\_vip\_persons\_id) REFERENCES personal\_info\_of\_vip\_persons(personal\_info\_of\_vip\_persons\_id)

ALTER TABLE personal\_info\_of\_vip\_persons ADD height INT NULL

ALTER TABLE vip\_persons DROP COLUMN person\_priority

ALTER TABLE vip\_persons\_of\_airlines ADD PRIMARY KEY (vip\_id)

ALTER TABLE partner\_travel\_agencies ALTER COLUMN rating REAL NULL

ALTER TABLE airline\_partner\_travel\_agencies ADD idk VARCHAR(1000) NULL

ALTER DATABASE Airlines MODIFY FILE (NAME = "Airline", SIZE = 100MB, FILEGROWTH = 5%)

/\*Здесь мы удаляем созданные нами таблицы и бд.\*/

DROP TABLE airlines

DROP TABLE airports

DROP TABLE airports\_contacts

DROP TABLE airports\_of\_airlines

DROP TABLE planes

DROP TABLE personal\_info\_of\_workers

DROP TABLE workers\_contacts

DROP TABLE official\_position

DROP TABLE workers

DROP TABLE status\_of\_vip\_persons

DROP TABLE daughter\_companies

DROP TABLE vip\_persons\_of\_airlines

DROP TABLE vip\_persons

DROP TABLE airline\_partner\_travel\_agencies

DROP TABLE partner\_travel\_agencies

DROP TABLE personal\_info\_of\_vip\_persons

DROP DATABASE Airlines

Вывод: создали базу данных при помощи SQL запросов на основе физической и логической моделей из предыдущей практической.

ПРАКТИЧЕСКАЯ РАБОТА №4  
Тема: «Работа с данными в БД»

Цель работы: заполнить нашу созданную ранее БД данными, следуя данным критериям: 1. Заполнить каждую таблицу данными (по 6 записей в каждой таблице минимум) (INSERT); 2. Выполнить 10 способами вывод информации (SELECT); 3. Изменить данные в 3 таблицах (UPDATE); 4. Сделать удаление c условием и полную очистку таблиц (DELETE/TRUNCATE).

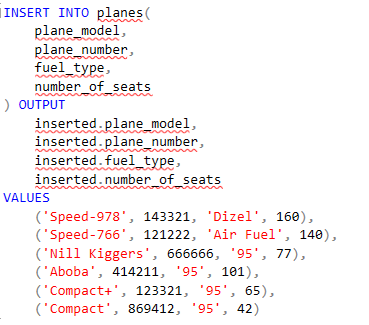


Рисунок – Заполнение таблицы

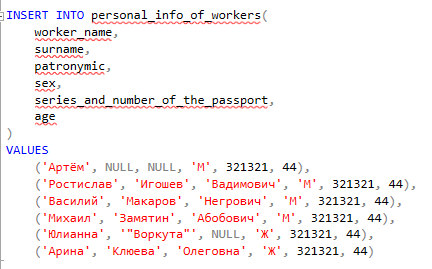


Рисунок – Еще одно заполнение таблицы

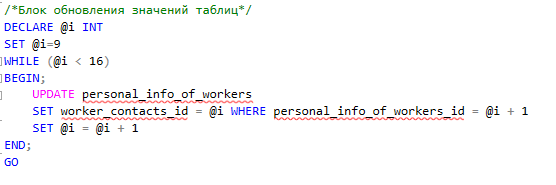


Рисунок – Изменение значений таблицы

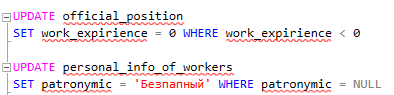


Рисунок – Еще одно изменение значений таблицы



Рисунок – Удаление с условием

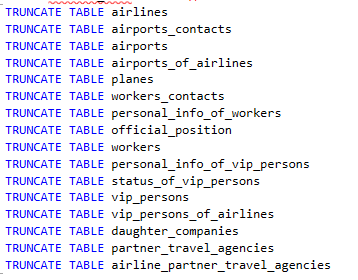


Рисунок – Очистка значений таблиц

Код работы:

/\*Блок Insert\*/

INSERT INTO planes(

plane\_model,

plane\_number,

fuel\_type,

number\_of\_seats

) OUTPUT

inserted.plane\_model,

inserted.plane\_number,

inserted.fuel\_type,

inserted.number\_of\_seats

VALUES

('Speed-978', 143321, 'Dizel', 160),

('Speed-766', 121222, 'Air Fuel', 140),

('Nill Kiggers', 666666, '95', 77),

('Aboba', 414211, '95', 101),

('Compact+', 123321, '95', 65),

('Compact', 869412, '95', 42)

INSERT INTO airlines(

airline\_name,

price

)

VALUES

('Аэрофлот', 1400000),

('Dzyuba Airlines', 9999999),

('Победа', 10),

('Bukanov.Fly', 69),

('Air Emirates', 1111),

('Poland Air', 787878)

INSERT INTO airports\_contacts(

airport\_adress,

operator\_number,

admin\_email

)

VALUES

('ул. Селезнева', 3112122, 'gay@mpt.ru'),

('бул. Блэк лайвс мэттер', 9999999, 'bbc@gmail.us'),

('ул. Победы', 10, 'blm@rambler.ru'),

('ул. Буканова', 69, 'pobeda@pobeda.nig'),

('проспект Абобы', 787878, 'bukanov@yandex.ua'),

('площадь ЛГБТ', 1111, 'aboba@gmail.com')

INSERT INTO airports(

airport\_name,

flights

)

VALUES

('Домодедово', 'Дубаи, Монако, Астана, Бишкек, Кабул, Пхеньянь, Сеул'),

('Внуково', 'Бодрум, Анталия, Париж, Мадрид'),

('Борисполь','Сан-франциско, Рио-де-жанейро, Бразилиа, Мехико'),

('Жуляны', 'Прага, Венеция, Винница, Мачу-пикчу'),

('Шереметьево', 'Варшава, Вроцлав, Минск, Мюнхен'),

('Ататюрк', 'Берлин, Рим, Лиссабон, Воркута')

INSERT INTO personal\_info\_of\_workers(

worker\_name,

surname,

patronymic,

sex,

series\_and\_number\_of\_the\_passport,

age

)

VALUES

('Артём', NULL, NULL, 'M', 321321, 44),

('Ростислав', 'Игошев', 'Вадимович', 'M', 321321, 44),

('Василий', 'Макаров', 'Негрович', 'M', 321321, 44),

('Михаил', 'Замятин', 'Абобович', 'M', 321321, 44),

('Юлианна', '"Воркута"', NULL, 'Ж', 321321, 44),

('Арина', 'Клюева', 'Олеговна', 'Ж', 321321, 44)

INSERT INTO workers\_contacts(

number,

email,

adress

)

VALUES

(32112, 'adasda@f.c', 'Ул. Гитлера'),

(5232, 'baza@baza.baza', 'Ул. Сталина'),

(745454, 'gson@gmail.com', 'Ул. Путина'),

(3222322, 'amygdala@yandex.ru', 'Ул. Хрущева'),

(9870634, 'zarossiy@rambler.ru', 'Ул. Медведева'),

(686895, 'nig@bingo.com', 'Ул. Горбачева')

INSERT INTO official\_position(

job\_title,

wage,

work\_expirience,

schedule

)

VALUES

('Менеджер', 70000, 5, '2-2, 10:00 - 22:00'),

('Программист', 120000, 1, '5-2, 08:00 - 16:00'),

('Тимлид', 300000, 9, '5-2, 08:00 - 18:00'),

('Уборщик', 45000, 8, '7-0, 06:00, 17:00'),

('Администратор', 100000, 6, '7-0, 10:00 - 18:00'),

('Директор', 700000, 15, '5-2, 10:00 - 20:00')

INSERT INTO personal\_info\_of\_vip\_persons(

person\_name,

surname,

patronymic,

sex,

series\_and\_number\_of\_the\_passport,

age

)

VALUES

('Нигган', NULL, NULL, 'M', 1111, 50),

('Рик', NULL, NULL, 'M', 322, 322),

('Василий', 'Ус', 'Силачевич', 'M', 22222, 22),

('Евгений', 'Замятин', NULL, 'M', 13221178, 44),

('Саша', 'Чепуха', NULL, 'Ж', 867556, 34),

('Кристина', 'Христос', 'Воскресова', 'Ж', 978780, 12)

INSERT INTO status\_of\_vip\_persons(

career,

state\_size,

shares\_in\_companies

)

VALUES

('Мачо', 20, '20% в Магните'),

('Полицейский', 1000, NULL),

('Спортик', 0, '2% в Спортмастере'),

('Писатель', 2099, '100% в ЛитКлуб'),

('Глава редан', 13120, '20% в Авиапарке'),

('Иисус', 99999999, '100% в Раю')

INSERT INTO vip\_persons(

person\_priority

)

VALUES

('Высокая'),

('Низкая'),

('Низкая'),

('Низкая'),

('Высокая'),

('Наивысшая')

INSERT INTO daughter\_companies(

company\_name,

flights,

proportion

)

VALUES

('Проигрыш', 'Чебоксары, Ульяновск', 15),

('Взлетыш', 'Кабул, Нью-Дели, Шанхай, Пекин', 7),

('Не важно где!', 'Чита, Магнитогорск', 6),

('Доля риска', 'Магнитошахтинск', 30),

('Братья Глэковы', 'Киев, Львов, Одесса, Херсон', 40),

('МММ', 'Донецк, Луганск, Бали, Мальдивы', 65)

INSERT INTO partner\_travel\_agencies(

partner\_travel\_agencies\_name,

flights,

branches,

rating

)

VALUES

('База туризма', 'Кипр, Бали', 'Москва, Санкт-Петербург, Екатеринбург', '8.9/10'),

('Кринж-путевки', 'Париж, Мадрид, Толедо', 'Москва, Санкт-Петербург, Екатеринбург', '9/10'),

('ТурЦия', 'Анкара, Стамбул, Кемер', 'Москва', '5/10'),

('Coral Travel', 'Вильнюс, Рига, Лима', 'Брянск, Белгород', '8/10'),

('Уголочек терпилы', 'Севастополь, Евпатория, Ялта, Алупка', 'Москва, Подольск, Домодедово, Челябинск', '4.9/10'),

('UATUR', 'Прага, Брест, Варшава, Шампань', 'Львов, Киев, Полтава, Тернополь, Ивано-Франковск', '8.1/10')

INSERT INTO airports\_of\_airlines(

airline\_id,

airports\_id

)

VALUES

(1,1),

(2,2),

(3,3),

(4,4),

(5,5),

(6,6)

INSERT INTO workers(

airline\_id,

personal\_info\_of\_workers\_id,

official\_position\_id

)

VALUES

(1, 10, 1),

(2, 11, 2),

(3, 12, 3),

(4, 13, 4),

(5, 14, 5),

(6, 15, 6)

INSERT INTO vip\_persons\_of\_airlines(

vip\_persons\_id,

airline\_id

)

VALUES

(6, 1),

(7, 2),

(8, 3),

(9, 4),

(10, 5),

(11, 6)

INSERT INTO airline\_partner\_travel\_agencies(

airline\_id,

partner\_travel\_agencies\_id

)

VALUES

(1, 1),

(2, 2),

(3, 3),

(4, 4),

(5, 5),

(6, 6)

/\*Блок обновления значений таблиц\*/

DECLARE @i INT

SET @i=9

WHILE (@i < 16)

BEGIN;

UPDATE personal\_info\_of\_workers

SET worker\_contacts\_id = @i WHERE personal\_info\_of\_workers\_id = @i + 1

SET @i = @i + 1

END;

GO

DECLARE @i INT

SET @i=6

WHILE (@i < 12)

BEGIN;

UPDATE daughter\_companies

SET airline\_id = 6 WHERE proportion = 65

SET @i = @i + 1

END;

GO

UPDATE official\_position

SET work\_expirience = 0 WHERE work\_expirience < 0

UPDATE personal\_info\_of\_workers

SET patronymic = 'Безпапный' WHERE patronymic = NULL

/\*Блок очистки таблиц\*/

DELETE TOP (1) FROM airlines

WHERE airline\_name = 'Победа'

TRUNCATE TABLE airlines

TRUNCATE TABLE airports\_contacts

TRUNCATE TABLE airports

TRUNCATE TABLE airports\_of\_airlines

TRUNCATE TABLE planes

TRUNCATE TABLE workers\_contacts

TRUNCATE TABLE personal\_info\_of\_workers

TRUNCATE TABLE official\_position

TRUNCATE TABLE workers

TRUNCATE TABLE personal\_info\_of\_vip\_persons

TRUNCATE TABLE status\_of\_vip\_persons

TRUNCATE TABLE vip\_persons

TRUNCATE TABLE vip\_persons\_of\_airlines

TRUNCATE TABLE daughter\_companies

TRUNCATE TABLE partner\_travel\_agencies

TRUNCATE TABLE airline\_partner\_travel\_agencies

Вывод: заполнили нашу ранее созданную БД данными, следуя критериям, описанным в цели работы.

ПРАКТИЧЕСКАЯ РАБОТА №5

Тема: «Объединение данных»

Цель работы: Изучить конструкции JOIN в SQL, их структуру, разновидности, функциональность, применив некоторые из них на практике в качестве задания.

Для начала работы выполним конструкции JOIN. Список выполненных конструкций:

* Сделать INNER JOIN c 3 условиями;
* Сделать RIGHT JOIN c 3 условиями;
* Сделать LEFT JOIN с 3 условиями;
* Сделать FULL JOIN с 3 условиями ;
* Сделать 2 CROSS GOIN;
* Сделать 2 UNION и 1 UNION ALL;
* Сделать 1 EXCEPT.

Результаты работы:

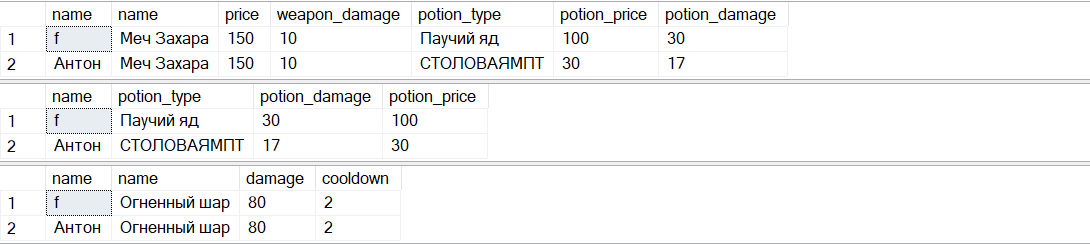


Рисунок – INNER JOIN

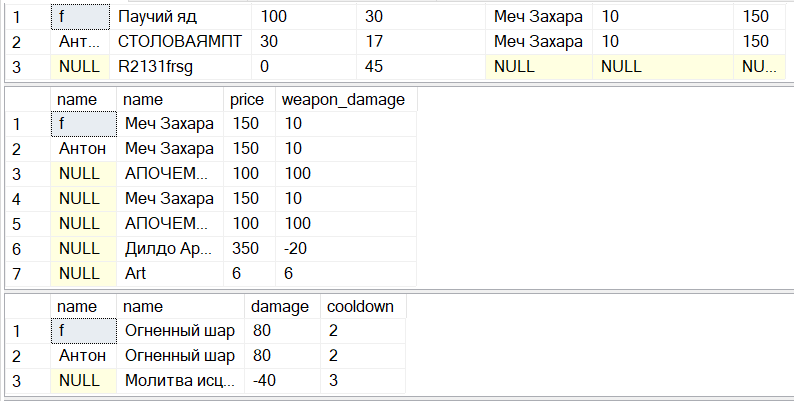


Рисунок – LEFT JOIN

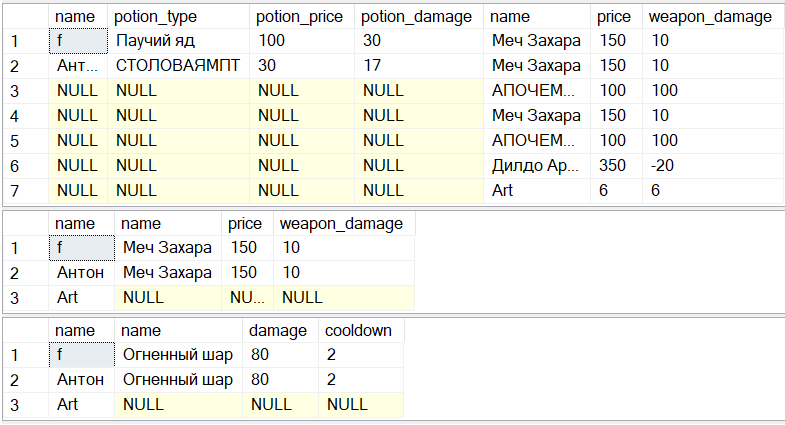


Рисунок – RIGHT JOIN

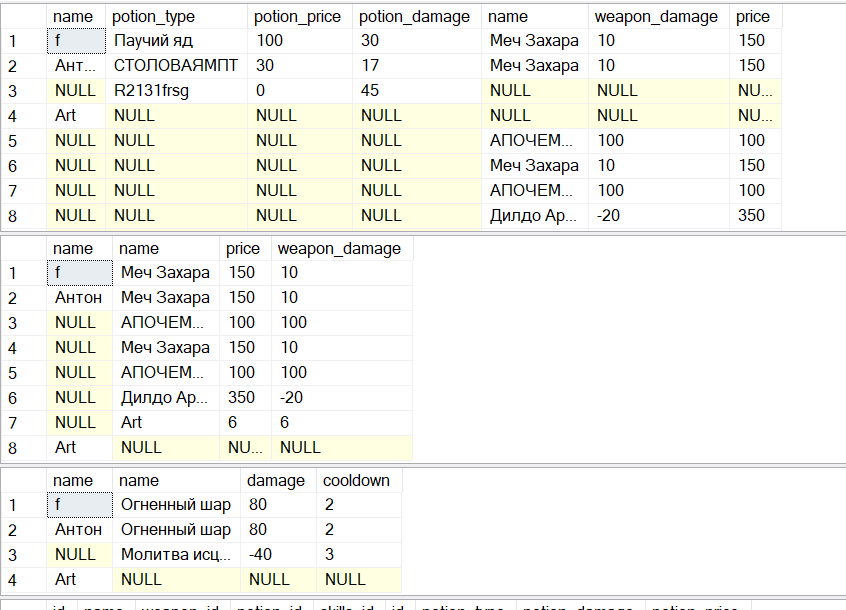


Рисунок – FULL JOIN

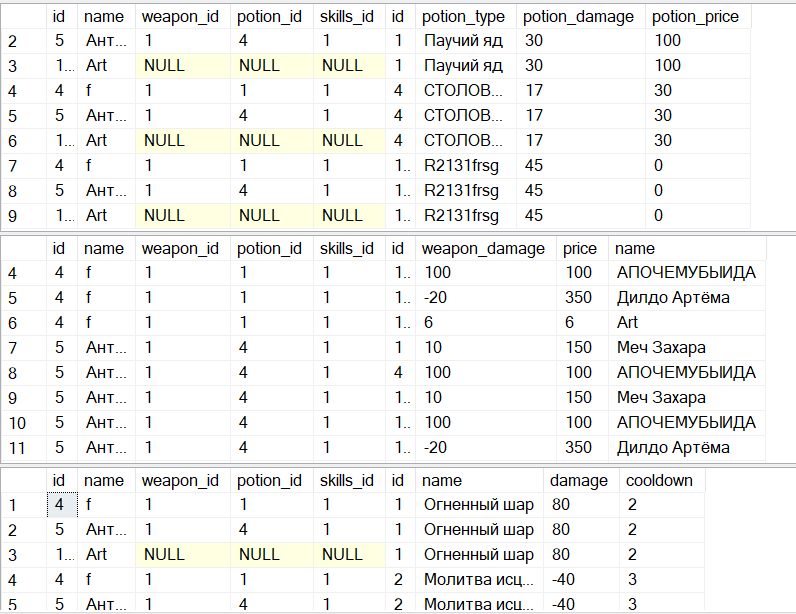


Рисунок – CROSS JOIN

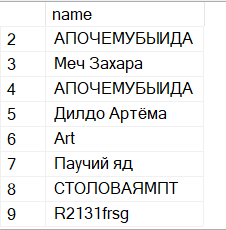


Рисунок – UNION ALL

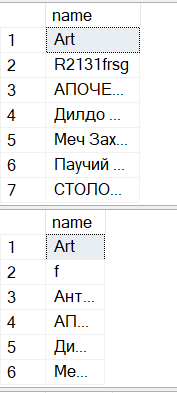


Рисунок - UNION

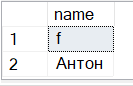


Рисунок 36 – EXCEPT

Код работы:

/\*Блок INNER JOIN\*/

SELECT

player.name,

weapon.name,

weapon.price,

weapon.weapon\_damage,

potion.potion\_type,

potion.potion\_price,

potion.potion\_damage

FROM player

INNER JOIN weapon

ON player.weapon\_id = weapon.id

INNER JOIN potion

ON potion.id = player.potion\_id

SELECT

player.name,

potion.potion\_type,

potion.potion\_damage,

potion.potion\_price

FROM player

INNER JOIN potion

ON player.potion\_id = potion.id

SELECT

player.name,

skills.name,

skills.damage,

skills.cooldown

FROM player

INNER JOIN skills

ON player.skills\_id = skills.id

/\*Блок LEFT OUTER JOIN\*/

SELECT

player.name,

potion.potion\_type,

potion.potion\_price,

potion.potion\_damage,

weapon.name,

weapon.weapon\_damage,

weapon.price

FROM potion

left outer JOIN player

ON player.potion\_id = potion.id

left outer JOIN weapon

ON player.weapon\_id = weapon.id

SELECT

player.name,

weapon.name,

weapon.price,

weapon.weapon\_damage

FROM weapon

left outer JOIN player

ON player.weapon\_id = weapon.id

SELECT

player.name,

skills.name,

skills.damage,

skills.cooldown

FROM skills

left outer JOIN player

ON player.skills\_id = skills.id

/\*Блок RIGHT JOIN\*/

SELECT

player.name,

potion.potion\_type,

potion.potion\_price,

potion.potion\_damage,

weapon.name,

weapon.price,

weapon\_damage

FROM potion

right outer JOIN player

ON player.potion\_id = potion.id

right outer JOIN weapon

ON player.weapon\_id = weapon.id

SELECT

player.name,

weapon.name,

weapon.price,

weapon.weapon\_damage

FROM weapon

right outer JOIN player

ON player.weapon\_id = weapon.id

SELECT

player.name,

skills.name,

skills.damage,

skills.cooldown

FROM skills

right outer JOIN player

ON player.skills\_id = skills.id

/\*Блок FULL JOIN\*/

SELECT

player.name,

potion.potion\_type,

potion.potion\_price,

potion.potion\_damage,

weapon.name,

weapon\_damage,

weapon.price

FROM potion

full JOIN player

ON player.potion\_id = potion.id

full JOIN weapon

ON weapon.id = player.weapon\_id

SELECT

player.name,

weapon.name,

weapon.price,

weapon.weapon\_damage

FROM weapon

FULL JOIN player

ON player.weapon\_id = weapon.id

SELECT

player.name,

skills.name,

skills.damage,

skills.cooldown

FROM skills

FULL JOIN player

ON player.skills\_id = skills.id

/\*Блок CROSS JOIN\*/

SELECT \* FROM player

CROSS JOIN potion

SELECT \* FROM player

CROSS JOIN weapon

SELECT \* FROM player

CROSS JOIN skills

/\*Блок UNION ALL\*/

SELECT name

FROM weapon

UNION ALL SELECT potion\_type

FROM potion

/\*Блок UNION\*/

SELECT name

FROM weapon

UNION SELECT potion\_type

FROM potion

SELECT name

FROM player

UNION SELECT name

FROM weapon

/\*Блок EXCEPT\*/

SELECT name

FROM player

EXCEPT SELECT name

FROM weapon

Вывод: Изучили конструкции JOIN в SQL, их структуру, разновидности, функциональность, применив некоторые из них на практике в качестве задания.

ПРАКТИЧЕСКАЯ РАБОТА №6

Тема: «Встроенные и агрегатные функции»

Цель работы: научиться работать с встроенными и агрегатными функциями в запросах SQL, разобраться в их структуре, функциональности, особенностях. Выполнить задания на отработку темы.

Первое задание будет заключаться в работе со строками, числами, датами, и условными операторами. Код первого задания выглядит следующим образом:

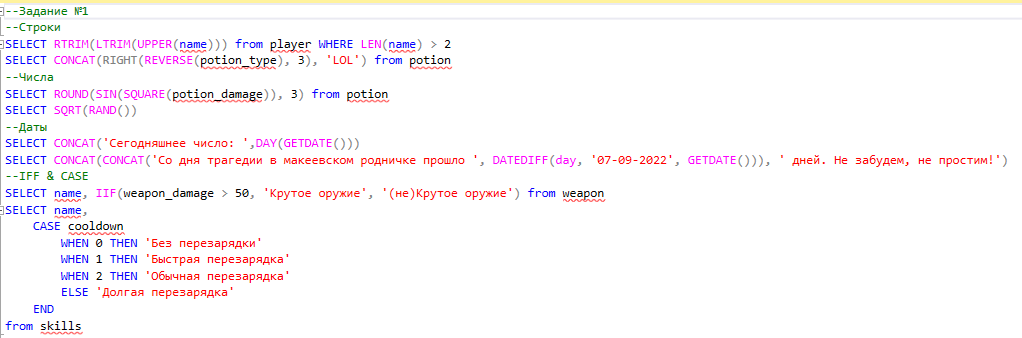


Рисунок – Первое задание

Приступим к третьему заданию. Его код выглядит так:

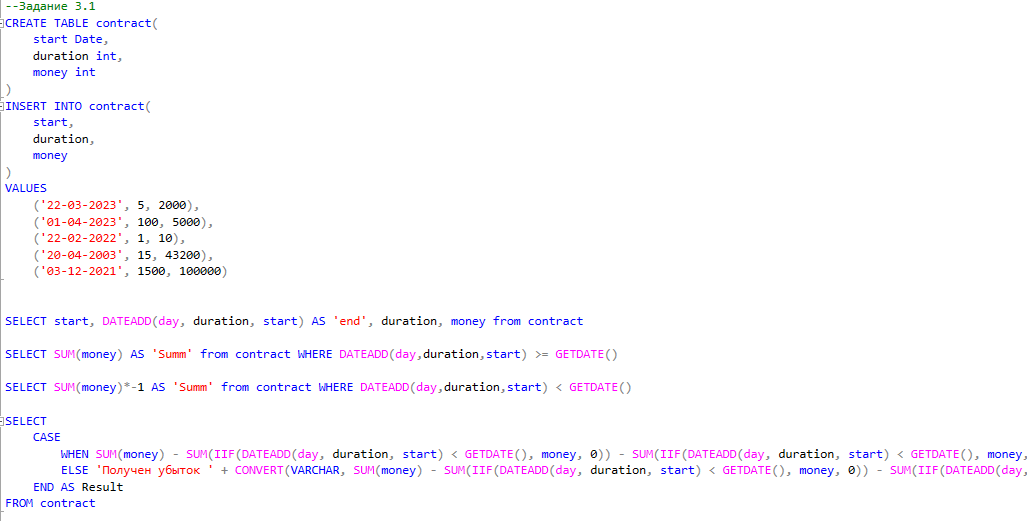


Рисунок – Задание 3 номер 1

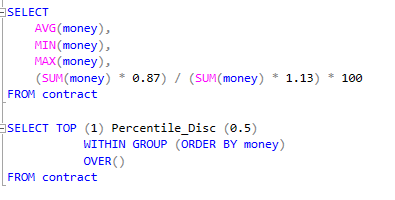


Рисунок – Задание 3 номер 1

Результаты работы:

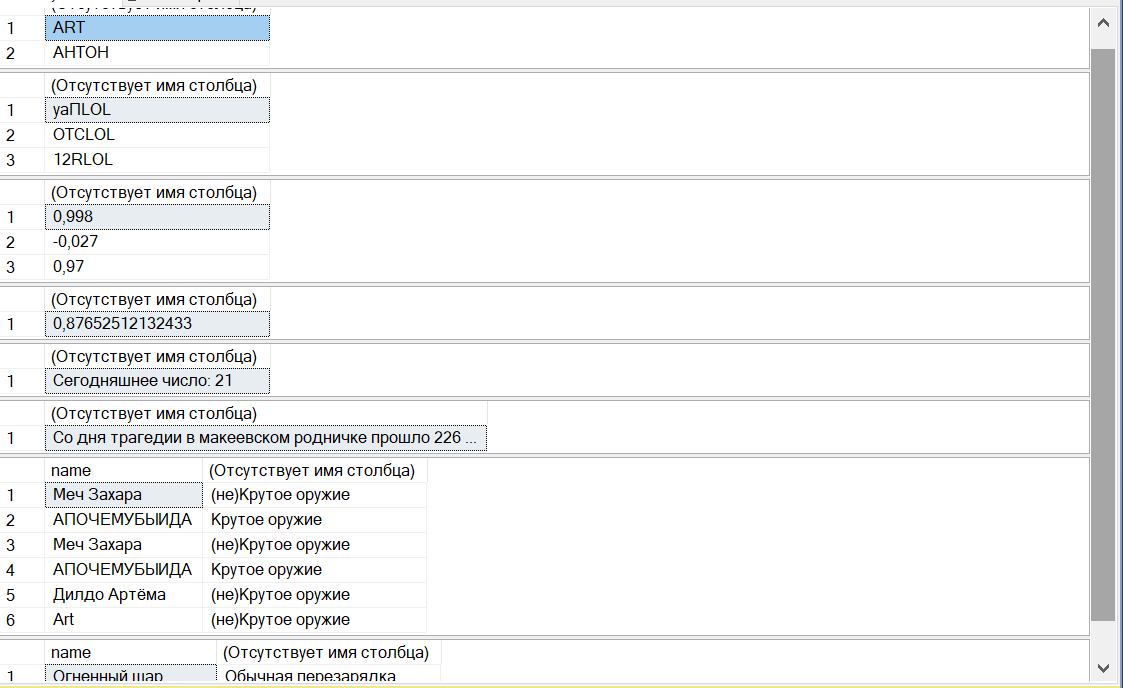


Рисунок – Результат работы первого задания

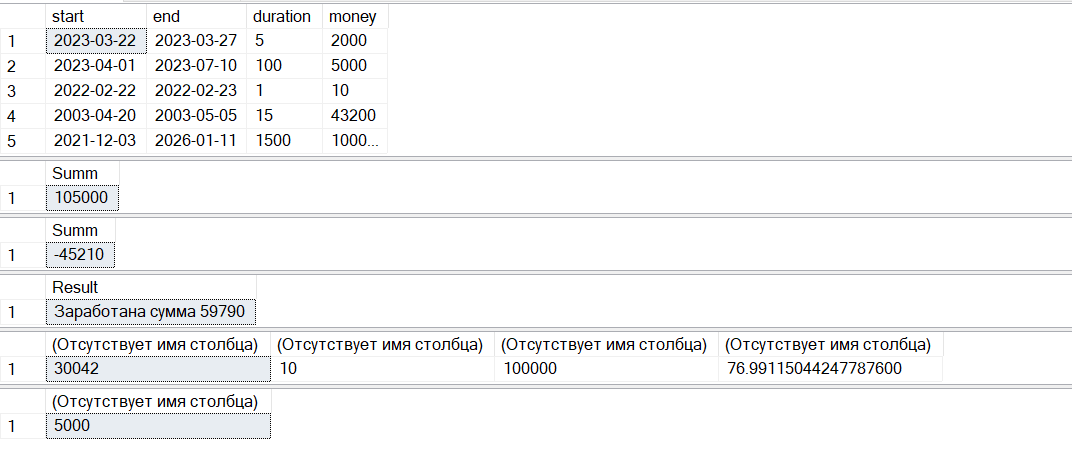


Рисунок – Результат работы второго задания

Вывод: научились работать с встроенными и агрегатными функциями в запросах SQL, разобрались в их структуре, функциональности, особенностях. Выполнили задания на отработку темы.

ПРАКТИЧЕСКАЯ РАБОТА №7

Тема: «Хранимые процедуры и триггеры»

Цель: научиться работать с хранимыми процедурами и триггерами в языке запросов SQL, выполнив практическое задание, нацеленное на отработку перечисленных тем.

Для начала, создадим процедуры для создания объектов. Процедуры будут принимать значения и записывать их в соответствующие таблицы.

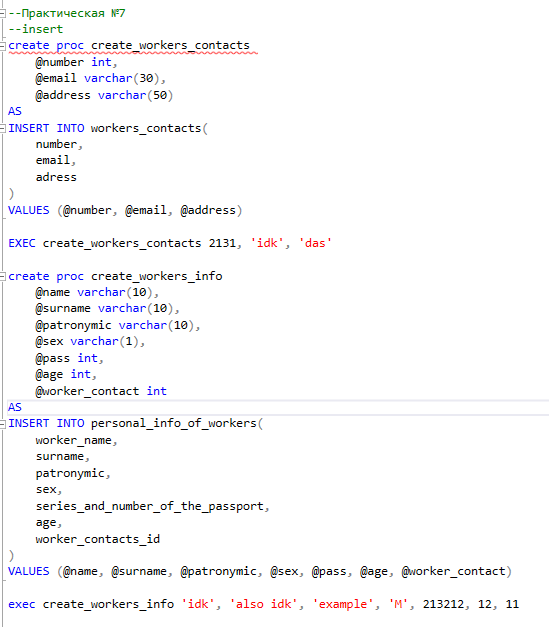


Рисунок – Процедуры для добавления

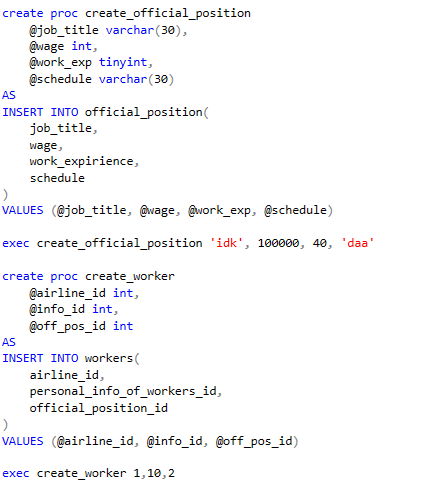


Рисунок – Процедуры для добавления

Теперь, сделаем то же самое с DELETE.

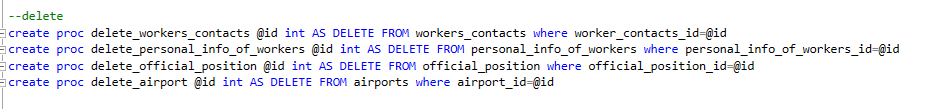


Рисунок – Процедуры для удаления

Остались процедуры для изменения.

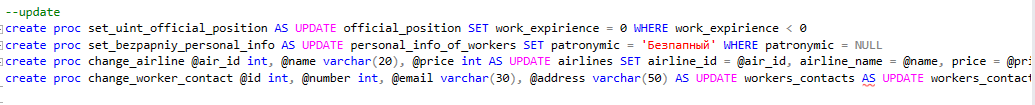


Рисунок – Процедуры для изменения

Теперь же создадим процедуры, которые будут выводить некоторую информацию из таблиц, используя в некоторых из них конструкцию JOIN.

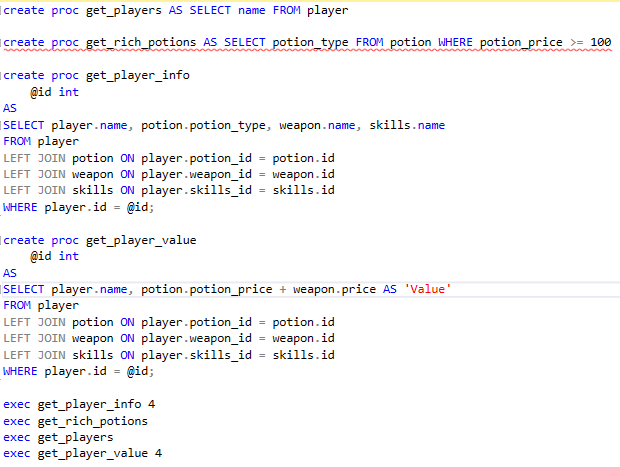


Рисунок – Процедуры вывода некоторой информации

Теперь попробуем реализовать историю добавления оружий. Для этого создадим таблицу истории в которой будет хранится название и цена оружия, и создадим триггер, который автоматически будет добавлять оружие в историю.

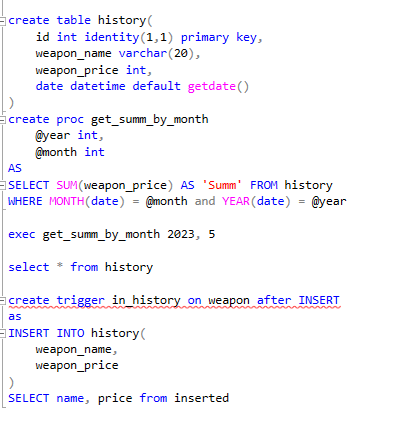


Рисунок – Таблица истории, триггер на добавление нового оружия, процедура суммы за месяц

Проверим процедуры вывода информации:

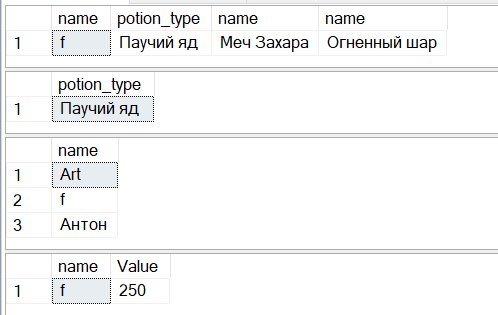


Рисунок – Результат работы процедур вывода информации

Теперь, проверим работу триггера. Для начала я создам объект в таблице оружий для примера. После этого – проверим таблицу истории.

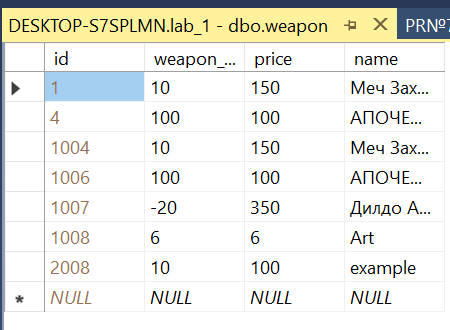


Рисунок – Объекты в таблице weapon

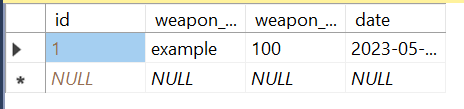


Рисунок – Объекты в таблице history

Как видим, объект в таблицу истории перенесся успешно. Остальные объекты не были перенесены, ибо были созданы до добавления триггера.

Вывод: научились работать с хранимыми процедурами и триггерами в языке запросов SQL, выполнив практическое задание, нацеленное на отработку перечисленных тем.

ПРАКТИЧЕСКАЯ РАБОТА №8

Тема: «Представления и функции»

Цель работы: научиться работать с представлениями и функциями в языке запросов SQL, используя среду разработки SSMS. Выполнить практическое задание, нацеленное на отработку вышеуказанных знаний.

Начнём с того, что такое представления. Виртуальные таблицы или же представления в SQL - это временные таблицы, которые создаются на основе запросов к другим таблицам. Они не существуют физически в базе данных, но могут использоваться для выполнения сложных запросов и агрегирования данных. В них можно использовать большинство операторов SQL, но нельзя изменять данные в исходных таблицах. Чаще всего подобные конструкции используют для упрощенного вывода информации из таблиц и для ограничения взаимодействия с ней.

Теперь, создадим несколько представлений.

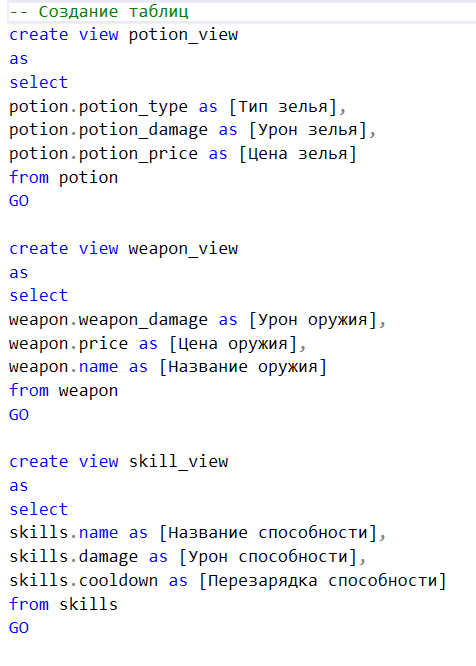


Рисунок – Создание представлений

Представления мы разработали. Может возникнуть вопрос, в чем различие представлений и хранимых процедур? Разница в том, что представления – это совокупность запросов, обращающихся к изначальной таблицы, нацеленные на упрощение работы вывода информации из них. А хранимые процедуры тоже нужны для работы с таблицами, но выполняют они другие функции и нужны чаще всего для изменения информации в таблицах. С этим разобрались. Теперь создадим DML – запросы (Insert, delete, update) к нашим представлениям.

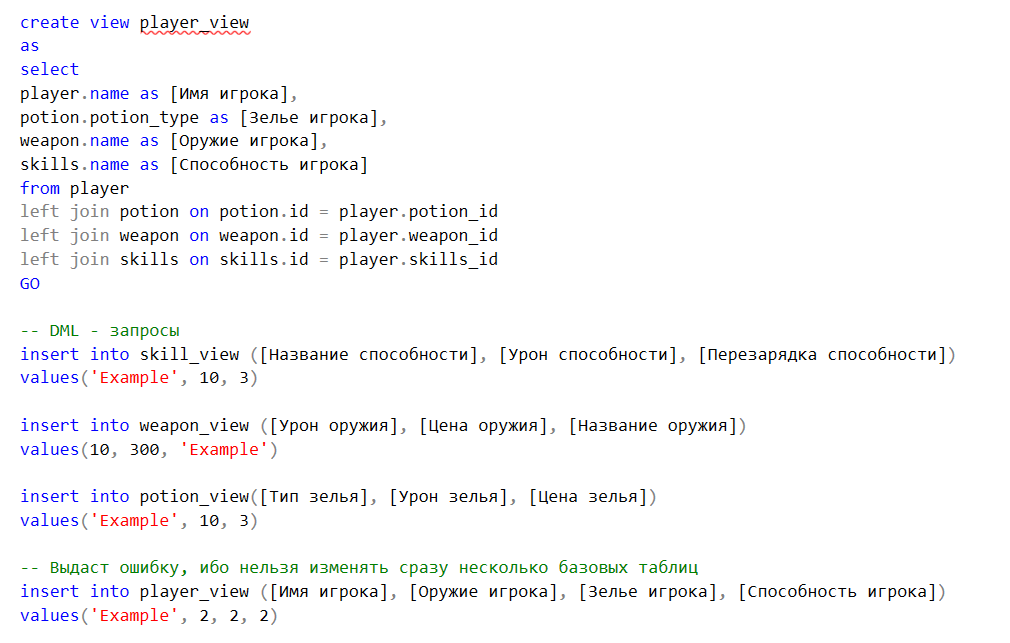


Рисунок – Создание представления и DML – запросов

DML – запросы мы создали. Теперь – попробуем создать функции. Существует два вида функций – табличные и скалярные. Скалярные возвращают какое-либо значение, а табличные – возвращают в качестве значения несколько строк (таблицу).

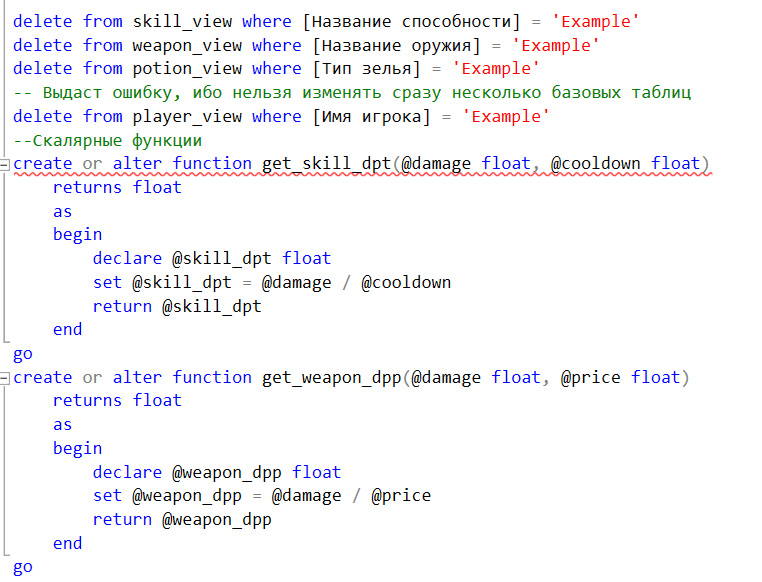


Рисунок – Скалярные функции

Скалярные функции мы создали. На очереди – табличные.

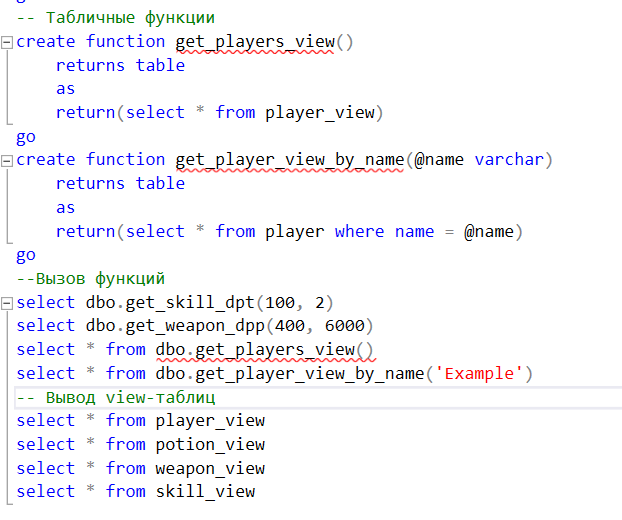


Рисунок – Табличные функции и их вывод, вывод представлений

Отличие функций от процедур в том, что процедуры могут изменять данные в таблице.

Табличные функции и вызов всех функций мы сделали, теперь посмотрим на результат работы:

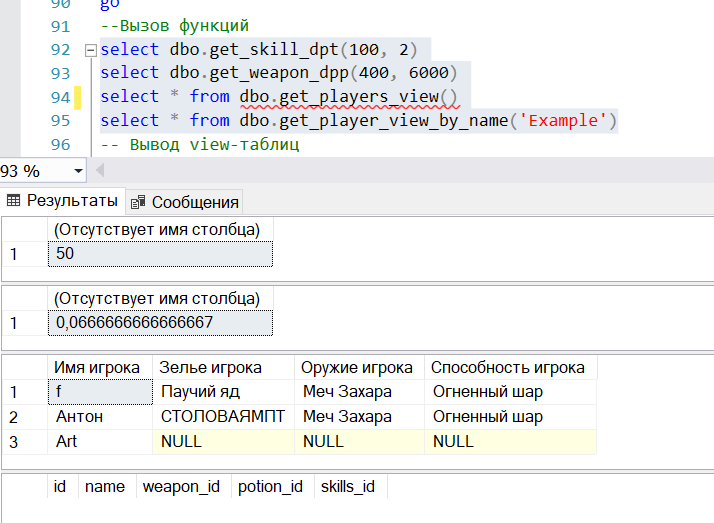


Рисунок – Результат работы функций

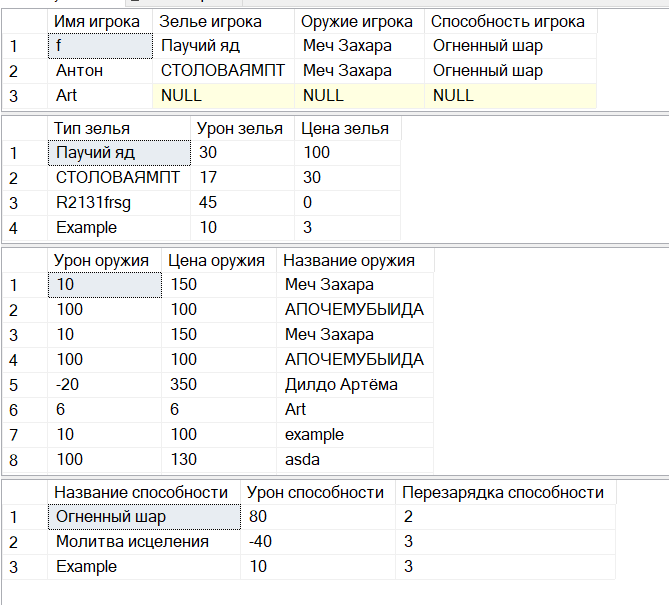


Рисунок – Результат работы вывода всех представлений

Вывод: научились работать с представлениями и функциями в языке запросов SQL, используя среду разработки SSMS, выполнили практическое задание, нацеленное на отработку вышеуказанных знаний.