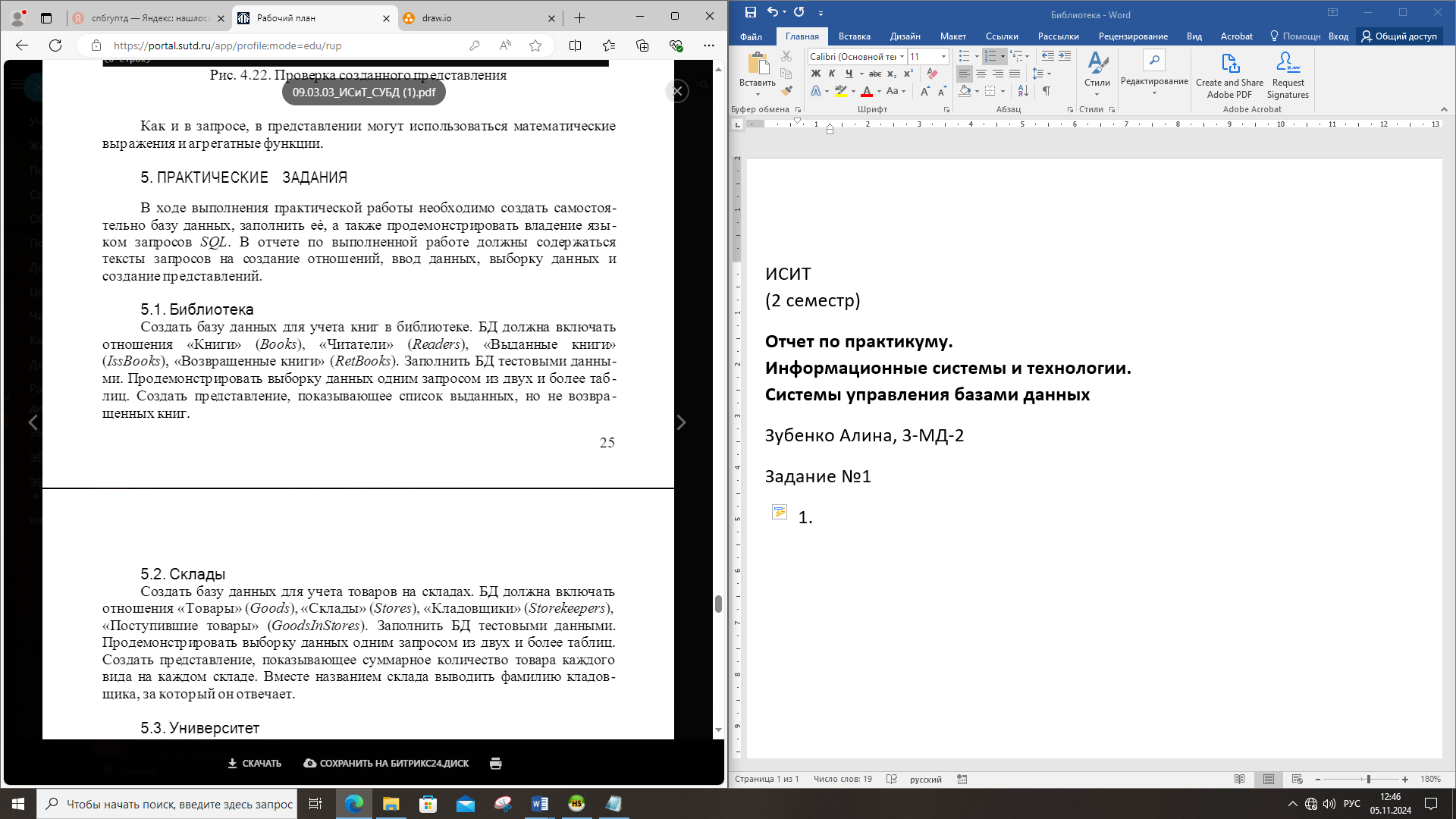
ИСИТ  
(2 семестр)

**Отчет по практикуму.**  
**Информационные системы и технологии.**   
**Системы управления базами данных**

Зубенко Алина, 3-МД-2

Задание №1

Текст задания



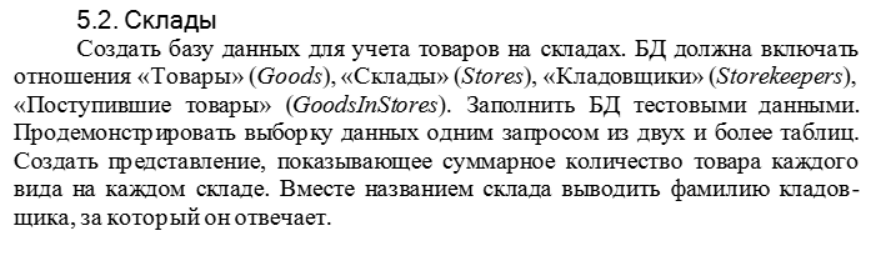
|  |
| --- |
|  |
| Рис 1. Схема ER |

|  |
| --- |
| **CREATE** **TABLE** Books (  book\_id **INT** **PRIMARY** **KEY**,  title **VARCHAR**(100),  author **VARCHAR**(100),  genre **VARCHAR**(50)  );  **INSERT** **INTO** Books (book\_id, title, author, genre) **VALUES**  (1, 'Война и мир', 'Лев Толстой', 'Роман'),  (2, 'Отцы и дети', 'Иван Тургенев', 'Роман'),  (3, 'Бойцовский клуб', 'Чак Паланик', 'Роман');  **CREATE** **TABLE** Readers (  reader\_id **INT** **PRIMARY** **KEY**,  **name** **VARCHAR**(50),  email **VARCHAR**(50)  );  **INSERT** **INTO** Readers (reader\_id, **name**, email) **VALUES**  (1, 'Амина', 'amina@gmail.com'),  (2, 'Алина', 'alina@gmail.com');  **CREATE** **TABLE** IssBooks (  issue\_id **INT** **PRIMARY** **KEY**,  book\_id **INT**,  reader\_id **INT**,  issue\_date **DATE**,  due\_date **DATE**,  **FOREIGN** **KEY** (book\_id) **REFERENCES** Books(book\_id),  **FOREIGN** **KEY** (reader\_id) **REFERENCES** Readers(reader\_id)  );  **INSERT** **INTO** IssBooks (issue\_id, book\_id, reader\_id, issue\_date, due\_date) **VALUES**  (1, 1, 1, '2022-01-01', '2022-02-01'),  (2, 2, 2, '2022-01-15', '2022-02-15');  **CREATE** **TABLE** RetBooks (  return\_id **INT** **PRIMARY** **KEY**,  issue\_id **INT**,  return\_date **DATE**,  **FOREIGN** **KEY** (issue\_id) **REFERENCES** IssBooks(issue\_id)  );  **INSERT** **INTO** RetBooks (return\_id, issue\_id, return\_date) **VALUES**  (1, 1, '2022-01-25');  **SELECT** b.title, r.**name**  **FROM** IssBooks ib  **JOIN** Books b **ON** ib.book\_id = b.book\_id  **JOIN** Readers r **ON** ib.reader\_id = r.reader\_id;  **CREATE** **VIEW** IssuedBooks **AS**  **SELECT** b.title, r.**name**  **FROM** IssBooks ib  **JOIN** Books b **ON** ib.book\_id = b.book\_id  **JOIN** Readers r **ON** ib.reader\_id = r.reader\_id  **LEFT** **JOIN** RetBooks rb **ON** ib.issue\_id = rb.issue\_id  **WHERE** rb.return\_id **IS** **NULL**; |
| Листинг 1. SQL-скрипт реализации БД Библиотека |

|  |
| --- |
|  |
| Рис.2 Экранный снимок исполнения сценария SQL БД Библиотека |

Задание №2

Текст задания



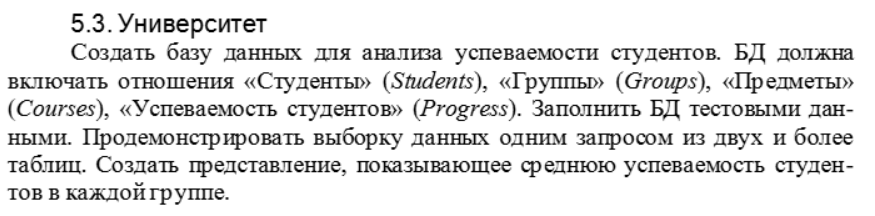
|  |
| --- |
|  |
| Рис 3. Схема ER |

|  |
| --- |
| CREATE DATABASE Warehouse;  USE Warehouse;      CREATE TABLE Stores (  id INT AUTO\_INCREMENT PRIMARY KEY,  name VARCHAR(100) NOT NULL  );      CREATE TABLE Storekeepers (  id INT AUTO\_INCREMENT PRIMARY KEY,  last\_name VARCHAR(100) NOT NULL,  store\_id INT,  FOREIGN KEY (store\_id) REFERENCES Stores(id)  );      CREATE TABLE Goods (  id INT AUTO\_INCREMENT PRIMARY KEY,  name VARCHAR(100) NOT NULL  );      CREATE TABLE GoodsInStores (  id INT AUTO\_INCREMENT PRIMARY KEY,  goods\_id INT,  store\_id INT,  quantity INT NOT NULL,  FOREIGN KEY (goods\_id) REFERENCES Goods(id),  FOREIGN KEY (store\_id) REFERENCES Stores(id)  );      INSERT INTO Stores (name) VALUES ('Склад 1'), ('Склад 2');      INSERT INTO Storekeepers (last\_name, store\_id) VALUES ('Альборуева', 1), ('Зубенко', 2);    --  INSERT INTO Goods (name) VALUES ('Товар A'), ('Товар B'), ('Товар C');      INSERT INTO GoodsInStores (goods\_id, store\_id, quantity) VALUES  (1, 1, 100),  (2, 1, 200),  (1, 2, 150),  (3, 2, 300);      SELECT  s.name AS store\_name,  g.name AS goods\_name,  gis.quantity  FROM  GoodsInStores gis  JOIN  Stores s ON gis.store\_id = s.id  JOIN  Goods g ON gis.goods\_id = g.id;      CREATE VIEW SummaryGoods AS  SELECT  s.name AS store\_name,  sk.last\_name AS storekeeper\_last\_name,  g.name AS goods\_name,  SUM(gis.quantity) AS total\_quantity  FROM  GoodsInStores gis  JOIN  Stores s ON gis.store\_id = s.id  JOIN  Storekeepers sk ON s.id = sk.store\_id  JOIN  Goods g ON gis.goods\_id = g.id  GROUP BY  s.name, sk.last\_name, g.name;      SELECT \* FROM SummaryGoods; |
| Листинг 2. SQL-скрипт реализации БД Склад |

|  |
| --- |
|  |
| Рис.4 Экранный снимок исполнения сценария SQL БД Склад |

Задание №3

Текс задания



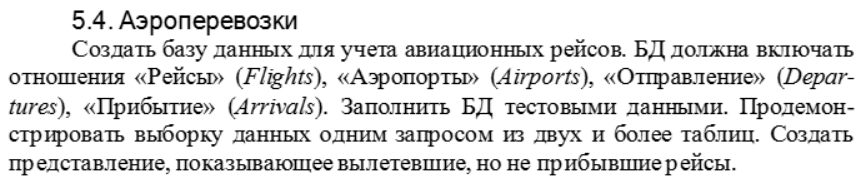
|  |
| --- |
|  |
| Рис 5. Схема ER |

|  |
| --- |
| CREATE DATABASE university;  USE university;      CREATE TABLE Groups (  id INT AUTO\_INCREMENT PRIMARY KEY,  group\_name VARCHAR(50) NOT NULL  );      CREATE TABLE Students (  id INT AUTO\_INCREMENT PRIMARY KEY,  name VARCHAR(100) NOT NULL,  group\_id INT,  FOREIGN KEY (group\_id) REFERENCES Groups(id)  );      CREATE TABLE Courses (  id INT AUTO\_INCREMENT PRIMARY KEY,  course\_name VARCHAR(100) NOT NULL  );      CREATE TABLE Progress (  id INT AUTO\_INCREMENT PRIMARY KEY,  student\_id INT,  course\_id INT,  grade FLOAT,  FOREIGN KEY (student\_id) REFERENCES Students(id),  FOREIGN KEY (course\_id) REFERENCES Courses(id)  );    INSERT INTO Groups (group\_name) VALUES ('1-МД-2'), ('2-МД-2'), ('3-МД-2');    -- Заполнение таблицы Students  INSERT INTO Students (name, group\_id) VALUES  ('Вафобеков Вафобек', 1),  ('Кодиров Азам', 1),  ('Альборуева Амина', 2),  ('Зубенко Алина', 2),  ('Жаворонков Кирилл', 3);      INSERT INTO Courses (course\_name) VALUES  ('Математика'),  ('Физика'),  ('Химия'),  ('История');      INSERT INTO Progress (student\_id, course\_id, grade) VALUES  (1, 1, 4.5),  (1, 2, 3.0),  (2, 1, 5.0),  (2, 3, 4.0),  (3, 2, 3.5),  (3, 4, 5.0),  (4, 1, 4.0),  (4, 2, 4.5),  (5, 3, 2.0);        SELECT  s.name AS student\_name,  g.group\_name,  AVG(p.grade) AS average\_grade  FROM  Students s  JOIN  Groups g ON s.group\_id = g.id  JOIN  Progress p ON s.id = p.student\_id  GROUP BY  s.id, g.group\_name;      CREATE VIEW average\_progress AS  SELECT  g.group\_nae,  AVG(p.grade) AS average\_grade  FROM  Groups g  JOIN  Students s ON g.id = s.group\_id  JOIN  Progress p ON s.id = p.student\_id  GROUP BY  g.group\_name;    SELECT \* FROM average\_progress; |
| Листинг 3. SQL-скрипт реализации БД Университет |

|  |
| --- |
|  |
| Рис.6 Экранный снимок исполнения сценария SQL БД Университет |

Задание №4

Текст задания



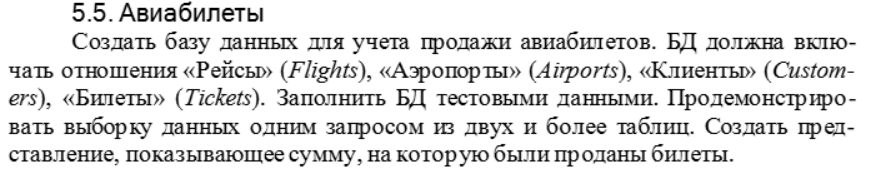
|  |
| --- |
|  |
| Рис 7. Схема ER |

|  |
| --- |
| CREATE DATABASE flight\_management;  USE flight\_management;      CREATE TABLE Airports (  id INT AUTO\_INCREMENT PRIMARY KEY,  airport\_name VARCHAR(100) NOT NULL,  city VARCHAR(100) NOT NULL,  country VARCHAR(100) NOT NULL  );      CREATE TABLE Flights (  id INT AUTO\_INCREMENT PRIMARY KEY,  flight\_number VARCHAR(20) NOT NULL,  departure\_airport\_id INT,  arrival\_airport\_id INT,  departure\_time DATETIME,  arrival\_time DATETIME,  FOREIGN KEY (departure\_airport\_id) REFERENCES Airports(id),  FOREIGN KEY (arrival\_airport\_id) REFERENCES Airports(id)  );    INSERT INTO Airports (airport\_name, city, country) VALUES  ('Шереметьево', 'Москва', 'Россия'),  ('Домодедово', 'Москва', 'Россия'),  ('Пулково', 'Санкт-Петербург', 'Россия'),  ('Хитроу', 'Лондон', 'Великобритания'),  ('Джон Кеннеди', 'Нью-Йорк', 'США');      INSERT INTO Flights (flight\_number, departure\_airport\_id, arrival\_airport\_id, departure\_time, arrival\_time) VALUES  ('S7 120', 1, 4, '2023-10-01 10:00:00', '2023-10-01 12:00:00'),  ('BA 234', 4, 2, '2023-10-01 14:30:00', '2023-10-01 18:00:00'),  ('AA 567', 2, 5, '2023-10-01 16:00:00', NULL),  ('DL 890', 3, 1, '2023-10-01 09:00:00', '2023-10-01 10:30:00');    SELECT  f.flight\_number,  a1.airport\_name AS departure\_airport,  a2.airport\_name AS arrival\_airport,  f.departure\_time,  f.arrival\_time  FROM  Flights f  JOIN  Airports a1 ON f.departure\_airport\_id = a1.id  JOIN  Airports a2 ON f.arrival\_airport\_id = a2.id;    CREATE VIEW Departed\_Not\_Arrived AS  SELECT  f.flight\_number,  a1.airport\_name AS departure\_airport,  a2.airport\_name AS arrival\_airport,  f.departure\_time  FROM  Flights f  JOIN  Airports a1 ON f.departure\_airport\_id = a1.id  JOIN  Airports a2 ON f.arrival\_airport\_id = a2.id  WHERE  f.arrival\_time IS NULL;    SELECT \* FROM Departed\_Not\_Arrived; |
| Листинг 4. SQL-скрипт реализации БД Аэроперевозки |

|  |
| --- |
|  |
| Рис.8 Экранный снимок исполнения сценария SQL БД Аэроперевозки |

Задание№ 5

Текст задания



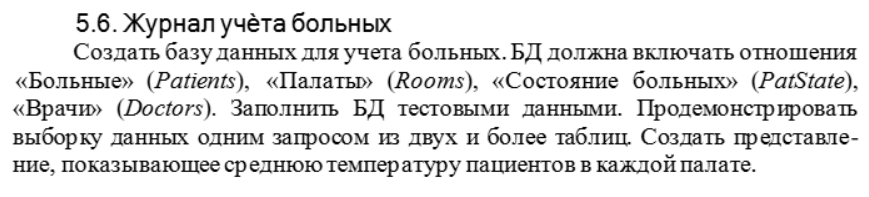
|  |
| --- |
|  |
| Рис 9. Схема ER |

|  |
| --- |
| CREATE DATABASE FlightBooking;    USE FlightBooking;      CREATE TABLE Airports (  id INT AUTO\_INCREMENT PRIMARY KEY,  airport\_name VARCHAR(100) NOT NULL,  city VARCHAR(100) NOT NULL,  country VARCHAR(100) NOT NULL  );      CREATE TABLE Flights (  id INT AUTO\_INCREMENT PRIMARY KEY,  flight\_number VARCHAR(20) NOT NULL,  departure\_airport\_id INT,  arrival\_airport\_id INT,  departure\_time DATETIME NOT NULL,  arrival\_time DATETIME NOT NULL,  FOREIGN KEY (departure\_airport\_id) REFERENCES Airports(id),  FOREIGN KEY (arrival\_airport\_id) REFERENCES Airports(id)  );      CREATE TABLE Customers (  id INT AUTO\_INCREMENT PRIMARY KEY,  first\_name VARCHAR(50) NOT NULL,  last\_name VARCHAR(50) NOT NULL,  email VARCHAR(100) NOT NULL UNIQUE  );      CREATE TABLE Tickets (  id INT AUTO\_INCREMENT PRIMARY KEY,  flight\_id INT,  customer\_id INT,  ticket\_price DECIMAL(10, 2) NOT NULL,  purchase\_date DATETIME NOT NULL,  FOREIGN KEY (flight\_id) REFERENCES Flights(id),  FOREIGN KEY (customer\_id) REFERENCES Customers(id)  );    INSERT INTO Airports (airport\_name, city, country) VALUES  ('Джон Кеннеди', 'Нью-Йорк', 'США'),  ('Лос-Анджелес Аэропорт', 'Лос-Анджелес', 'США'),  ('Хитроу', 'Лондон', 'Великобритания'),  ('Аэропорт Токио', 'Токио', 'Япония');      INSERT INTO Flights (flight\_number, departure\_airport\_id, arrival\_airport\_id, departure\_time, arrival\_time) VALUES  ('AA101', 1, 2, '2024-05-01 10:00:00', '2024-05-01 13:00:00'),  ('BA202', 3, 1, '2024-05-02 14:00:00', '2024-05-02 17:00:00'),  ('JL303', 4, 1, '2024-05-03 09:30:00', '2024-05-03 15:00:00');      INSERT INTO Customers (first\_name, last\_name, email) VALUES  ('Амина', 'Альборуева', 'amina.al@gmail.com'),  ('Алина', 'Зубенко', 'alina.zu@gmail.com'),  ('Михаил', 'Вышинский', 'mishel.vysh@gmail.com');      INSERT INTO Tickets (flight\_id, customer\_id, ticket\_price, purchase\_date) VALUES  (1, 1, 150.00, NOW()),  (2, 2, 200.00, NOW()),  (1, 3, 150.00, NOW());    SELECT  c.first\_name as Customer\_First\_Name,  c.last\_name as Customer\_Last\_Name,  f.flight\_number as Flight\_Number,  f.departure\_time as Departure\_Time,  t.ticket\_price as Ticket\_Price  FROM  Tickets t  JOIN  Customers c ON t.customer\_id = c.id  JOIN  Flights f ON t.flight\_id = f.id;    CREATE VIEW Total\_Ticket\_Sales AS  SELECT SUM(ticket\_price) AS Total\_Sales  FROM Tickets;  SELECT \* FROM Total\_Ticket\_Sales; |
| Листинг 5. SQL-скрипт реализации БД Авиабилеты |

|  |
| --- |
|  |
| Рис.10 Экранный снимок исполнения сценария SQL БД Авиабилеты |

Задание №6

Текст задания



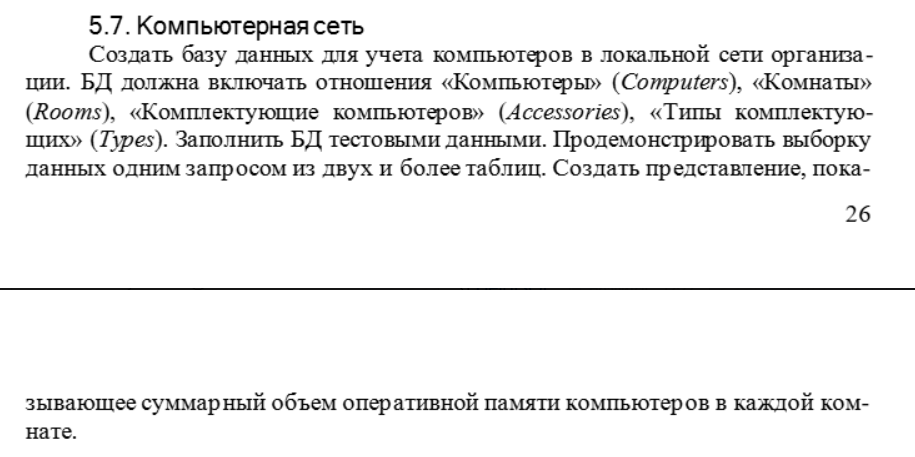
|  |
| --- |
|  |
| Рис 11. Схема ER |

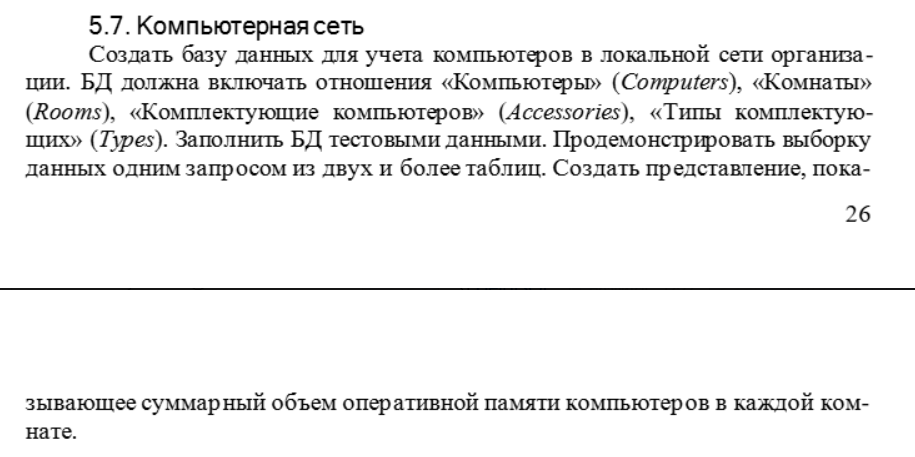
|  |
| --- |
| CREATE TABLE Patients (  patient\_id SERIAL PRIMARY KEY,  name VARCHAR(100) NOT NULL,  age INT NOT NULL,  doctor\_id INT,  room\_id INT,  state\_id INT  );  CREATE TABLE Rooms (  room\_id SERIAL PRIMARY KEY,  room\_number INT NOT NULL,  capacity INT NOT NULL  );  CREATE TABLE PatState (  state\_id SERIAL PRIMARY KEY,  description VARCHAR(100) NOT NULL,  temperature DECIMAL(3, 1) NOT NULL  FOREIGN KEY (patient\_id) REFERENCES Patients(patient\_id)  FOREIGN KEY (room\_id) REFERENCES Rooms(room\_id)  FOREIGN KEY (room\_id) REFERENCES Rooms(room\_id)  );  CREATE TABLE Doctors (  doctor\_id SERIAL PRIMARY KEY,  name VARCHAR(100) NOT NULL,  specialty VARCHAR(100)  );  INSERT INTO Rooms (room\_number, capacity) VALUES  (101, 2),  (102, 3);  INSERT INTO Doctors (name, specialty) VALUES  ('Фадеев И.', 'Кардиолог'),  ('Петров Е.', 'Невролог');  INSERT INTO PatState (description, temperature) VALUES  ('Номальное состояние', 36.6),  ('Хуже среднего', 38.0),  ('Критическое', 39.5);  INSERT INTO Patients (name, age, doctor\_id, room\_id, state\_id) VALUES  ('Алиса', 30, 1, 1, 1),  ('Боб', 45, 2, 1, 2),  ('Чарли', 60, 1, 2, 3),  ('Диана', 50, 2, 2, 1);  CREATE VIEW AvgTemperaturePerRoom AS  SELECT Rooms.room\_number, AVG(PatState.temperature) AS avg\_temperature  FROM Patients  JOIN Rooms ON Patients.room\_id = Rooms.room\_id  JOIN PatState ON Patients.state\_id = PatState.state\_id  GROUP BY Rooms.room\_number;  select\*from AvgTemperaturePerRoom; |
| Листинг 6. SQL-скрипт реализации БД Журнала учета больных |

|  |
| --- |
|  |
| Рис.12 Экранный снимок исполнения сценария SQL БД Журнала учета больных |

Задание №7

Текст задания



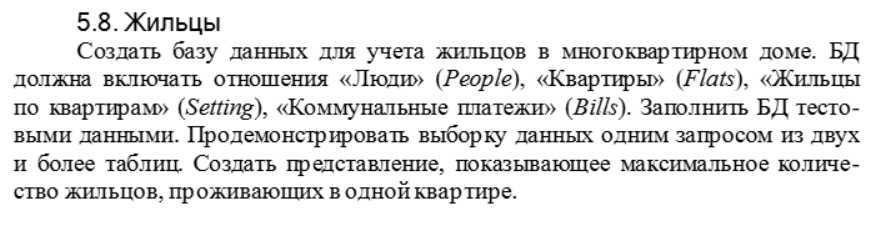


|  |
| --- |
|  |
| Рис 13. Схема ER |

|  |
| --- |
| CREATE TABLE Rooms (  room\_id SERIAL PRIMARY KEY,  room\_number INT NOT NULL  );  CREATE TABLE Computers (  computer\_id SERIAL PRIMARY KEY,  room\_id INT NOT NULL,  ram\_size INT NOT NULL,  FOREIGN KEY (room\_id) REFERENCES Rooms(room\_id)  );  CREATE TABLE Types (  type\_id SERIAL PRIMARY KEY,  type\_name VARCHAR(100) NOT NULL  );  CREATE TABLE Accessories (  accessory\_id SERIAL PRIMARY KEY,  computer\_id INT NOT NULL,  type\_id INT NOT NULL,  description VARCHAR(100),  FOREIGN KEY (computer\_id) REFERENCES Computers(computer\_id),  FOREIGN KEY (type\_id) REFERENCES Types(type\_id)  );  INSERT INTO Rooms (room\_number) VALUES  (101),  (102),  (103);  INSERT INTO Computers (room\_id, ram\_size) VALUES  (1, 8),  (1, 16),  (2, 4),  (3, 8);  INSERT INTO Types (type\_name) VALUES  ('Клавиатура'),  ('Мышь'),  ('Монитор');  INSERT INTO Accessories (computer\_id, type\_id, description) VALUES  (1, 1, 'Беспроводная клавиатура'),  (1, 2, 'Оптическая мышь'),  (2, 3, 'Монитор 24 дюйма'),  (3, 1, 'Механическая клавиатура'),  (4, 2, 'Беспроводная мышь');  CREATE VIEW TotalRAMPerRoom AS  SELECT Rooms.room\_number, SUM(Computers.ram\_size) AS total\_ram  FROM Computers  JOIN Rooms ON Computers.room\_id = Rooms.room\_id  GROUP BY Rooms.room\_number;  select\*from TotalRAMPerRoom; |
| Листинг 7. SQL-скрипт реализации БД Компьютерной сети |

|  |
| --- |
|  |
| Рис.14 Экранный снимок исполнения сценария SQL БД Компьютерной сети |

Задание №8

Текст задания

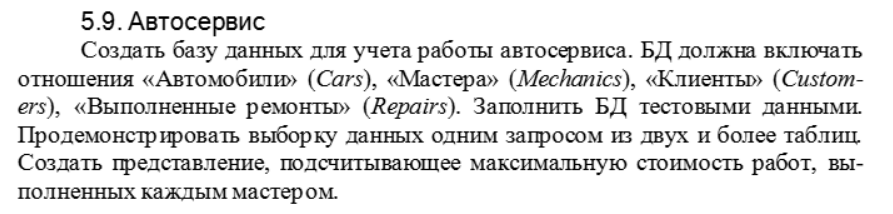
|  |
| --- |
|  |
| Рис 15. Схема ER |

|  |
| --- |
| CREATE TABLE People (  person\_id SERIAL PRIMARY KEY,  name VARCHAR(100) NOT NULL,  age INT NOT NULL  );  CREATE TABLE Flats (  flat\_id SERIAL PRIMARY KEY,  flat\_number INT NOT NULL  );  CREATE TABLE Setting (  setting\_id SERIAL PRIMARY KEY,  flat\_id INT NOT NULL,  person\_id INT NOT NULL,  FOREIGN KEY (flat\_id) REFERENCES Flats(flat\_id),  FOREIGN KEY (person\_id) REFERENCES People(person\_id)  );  CREATE TABLE Bills (  bill\_id SERIAL PRIMARY KEY,  flat\_id INT NOT NULL,  amount DECIMAL(10, 2) NOT NULL,  date DATE NOT NULL,  FOREIGN KEY (flat\_id) REFERENCES Flats(flat\_id)  );  INSERT INTO People (name, age) VALUES  ('Иван Иванов', 30),  ('Мария Петрова', 28),  ('Алексей Смирнов', 45),  ('Ольга Сидорова', 35),  ('Сергей Кузнецов', 40);  INSERT INTO Flats (flat\_number) VALUES  (101),  (102),  (103);  INSERT INTO Setting (flat\_id, person\_id) VALUES  (1, 1),  (1, 2),  (1, 3),  (2, 4),  (3, 5);  INSERT INTO Bills (flat\_id, amount, date) VALUES  (1, 1500.00, '2024-10-01'),  (1, 1600.00, '2024-11-01'),  (2, 2000.00, '2024-10-01'),  (2, 2100.00, '2024-11-01'),  (3, 1800.00, '2024-10-01');  CREATE VIEW MaxResidentsPerFlat AS  SELECT Flats.flat\_number, COUNT(Setting.person\_id) AS resident\_count  FROM Setting  JOIN Flats ON Setting.flat\_id = Flats.flat\_id  GROUP BY Flats.flat\_number  ORDER BY resident\_count DESC  LIMIT 1;  select\*from MaxResidentsPerFlat; |
| Листинг 8. SQL-скрипт реализации БД Жильцы |

|  |
| --- |
|  |
| Рис.16 Экранный снимок исполнения сценария SQL БД Жильцы |

Задание №9

Текст задания



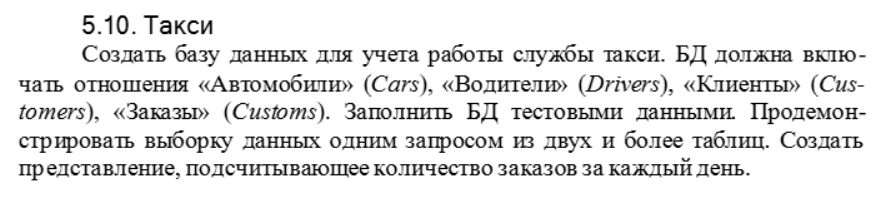
|  |
| --- |
|  |
| Рис 17. Схема ER |

|  |
| --- |
| CREATE TABLE Customers (  customer\_id SERIAL PRIMARY KEY,  name VARCHAR(100) NOT NULL,  phone VARCHAR(20)  );  CREATE TABLE Cars (  car\_id SERIAL PRIMARY KEY,  model VARCHAR(100) NOT NULL,  customer\_id INT NOT NULL,  FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id)  );  CREATE TABLE Mechanics (  mechanic\_id SERIAL PRIMARY KEY,  name VARCHAR(100) NOT NULL,  specialty VARCHAR(100)  );  CREATE TABLE Repairs (  repair\_id SERIAL PRIMARY KEY,  car\_id INT NOT NULL,  mechanic\_id INT NOT NULL,  cost DECIMAL(10, 2) NOT NULL,  repair\_date DATE NOT NULL,  description VARCHAR(255),  FOREIGN KEY (car\_id) REFERENCES Cars(car\_id),  FOREIGN KEY (mechanic\_id) REFERENCES Mechanics(mechanic\_id)  );  INSERT INTO Customers (name, phone) VALUES  ('Иван Иванов', '123-456-7890'),  ('Мария Петрова', '098-765-4321');  INSERT INTO Cars (model, customer\_id) VALUES  ('Toyota Corolla', 1),  ('Honda Civic', 2);  INSERT INTO Mechanics (name, specialty) VALUES  ('Сергей Смирнов', 'Двигатель'),  ('Алексей Кузнецов', 'Подвеска');  INSERT INTO Repairs (car\_id, mechanic\_id, cost, repair\_date, description) VALUES  (1, 1, 5000.00, '2024-10-01', 'Замена масла и фильтров'),  (1, 2, 7000.00, '2024-10-10', 'Ремонт подвески'),  (2, 1, 3000.00, '2024-11-05', 'Диагностика двигателя'),  (2, 2, 8000.00, '2024-11-10', 'Замена амортизаторов');  CREATE VIEW MaxRepairCostPerMechanic AS  SELECT Mechanics.name AS mechanic\_name, MAX(Repairs.cost) AS max\_cost  FROM Repairs  JOIN Mechanics ON Repairs.mechanic\_id = Mechanics.mechanic\_id  GROUP BY Mechanics.name;  select\*from MaxRepairCostPerMechanic; |
| Листинг 9. SQL-скрипт реализации БД Автосервис |

|  |
| --- |
|  |
| Рис.18 Экранный снимок исполнения сценария SQL БД Автосервиса |

Задание №10

Текст задания



|  |
| --- |
|  |
| Рис 19. Схема ER |

|  |
| --- |
| CREATE TABLE Customers (  customer\_id SERIAL PRIMARY KEY,  name VARCHAR(100) NOT NULL,  phone VARCHAR(20)  );  CREATE TABLE Cars (  car\_id SERIAL PRIMARY KEY,  model VARCHAR(100) NOT NULL,  license\_plate VARCHAR(20)  );  CREATE TABLE Drivers (  driver\_id SERIAL PRIMARY KEY,  name VARCHAR(100) NOT NULL,  license\_number VARCHAR(20)  );  CREATE TABLE Customs (  custom\_id SERIAL PRIMARY KEY,  car\_id INT NOT NULL,  driver\_id INT NOT NULL,  customer\_id INT NOT NULL,  order\_date DATE NOT NULL,  cost DECIMAL(10, 2) NOT NULL,  FOREIGN KEY (car\_id) REFERENCES Cars(car\_id),  FOREIGN KEY (driver\_id) REFERENCES Drivers(driver\_id),  FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id)  );  INSERT INTO Customers (name, phone) VALUES  ('Алексей Иванов', '123-456-7890'),  ('Ольга Петрова', '098-765-4321');  INSERT INTO Cars (model, license\_plate) VALUES  ('Toyota Camry', 'A123BC'),  ('Hyundai Solaris', 'B456CD');  INSERT INTO Drivers (name, license\_number) VALUES  ('Иван Сидоров', 'AB1234567'),  ('Сергей Смирнов', 'BC2345678');  INSERT INTO Customs (car\_id, driver\_id, customer\_id, order\_date, cost) VALUES  (1, 1, 1, '2024-10-01', 500.00),  (2, 2, 2, '2024-10-01', 300.00),  (1, 1, 2, '2024-10-02', 400.00),  (2, 2, 1, '2024-10-03', 700.00);  CREATE VIEW OrdersPerDay AS  SELECT order\_date, COUNT(custom\_id) AS order\_count  FROM Customs  GROUP BY order\_date;  select\*from OrdersPerDay; |
| Листинг 10. SQL-скрипт реализации БД Такси |

|  |
| --- |
|  |
| Рис.20 Экранный снимок исполнения сценария SQL БД Такси |