Санкт-Петербургский Национальный Исследовательский Университет ИТМО

Факультет Программной Инженерии и Компьютерной Техники



Вариант №1902

Лабораторная работа №1

По дисциплине

Базы Данных

Выполнил студент группы P3119:

Ануфриев Андрей Сергеевич

Преподаватель:

Бойко Владислав Алексеевич

Санкт-Петербург 2025 г.

Оглавление

[Текст задания 3](#_Toc193646402)

[Описание предметной области 4](#_Toc193646403)

[Список сущностей 4](#_Toc193646404)

[Инфологическая модель 5](#_Toc193646405)

[Датологическая модель 6](#_Toc193646406)

[Реализация 7](#_Toc193646407)

[Заполнение значений 8](#_Toc193646408)

[Вывод 8](#_Toc193646409)

[Ответы на вопросы 9](#_Toc193646410)

[1. Архитектура ANSI-SPARC 9](#_Toc193646411)

[Уровни архитектуры ANSI-SPARC: 9](#_Toc193646412)

[Преимущества архитектуры ANSI-SPARC: 9](#_Toc193646413)

[2. Модель "Сущность-Связь" (ER-модель) 9](#_Toc193646414)

[Основные компоненты ER-модели: 9](#_Toc193646415)

[Классификация сущностей: 10](#_Toc193646416)

[Виды связей: 10](#_Toc193646417)

[3. Ограничения целостности 10](#_Toc193646418)

[4. DDL (Data Definition Language) 11](#_Toc193646419)

[5. DML (Data Manipulation Language) 12](#_Toc193646420)

# Текст задания

Для выполнения лабораторной работы №1 необходимо:

1. На основе предложенной предметной области (текста) составить ее описание. Из полученного описания выделить сущности, их атрибуты и связи.
2. Составить инфологическую модель.
3. Составить даталогическую модель. При описании типов данных для атрибутов должны использоваться типы из СУБД PostgreSQL.
4. Реализовать даталогическую модель в PostgreSQL. При описании и реализации даталогической модели должны учитываться ограничения целостности, которые характерны для полученной предметной области.
5. Заполнить созданные таблицы тестовыми данными.

Для создания объектов базы данных у каждого студента есть своя схема. Название схемы соответствует имени пользователя в базе studs (sXXXXXX). Команда для подключения к базе studs:

*psql -h pg -d studs*

Каждый студент должен использовать свою схему при работе над лабораторной работой №1 (а также в рамках выполнения 2, 3 и 4 этапа курсовой работы).

Отчёт по лабораторной работе должен содержать:

1. ~~Текст задания.~~
2. ~~Описание предметной области.~~
3. ~~Список сущностей и их классификацию (стержневая, ассоциация, характеристика).~~
4. ~~Инфологическая модель (ER-диаграмма в расширенном виде - с атрибутами, ключами...).~~
5. ~~Даталогическая модель (должна содержать типы атрибутов, вспомогательные таблицы для отображения связей "многие-ко-многим").~~
6. Реализация даталогической модели на SQL.
7. Выводы по работе.

Темы для подготовки к защите лабораторной работы:

1. Архитектура ANSI-SPARC
2. Модель "Сущность-Связь". Классификация сущностей. Виды связей. Ограничения целостности.
3. DDL
4. DML

Описание предметной области**, по которой должна быть построена доменная модель:**

Диаспар почти не видел Олвина в последующие несколько недель, хотя всего лишь какая-то горстка людей заметила его отсутствие. Джизирак, обнаружив, что его ученик, вместо того чтобы бродить в районе границ города, все свое время проводит в Зале Совета, испытал некоторое облегчение, ибо полагал, что уж там-то с Олвином никакой беды не приключится. Эристон и Итания раз-другой навестили его комнату, убедились, что сын отсутствует, и не придали этому значения. Что же касается Алистры, то она оказалась более настойчивой.

# Список сущностей

People – стержневая

State – характеристика

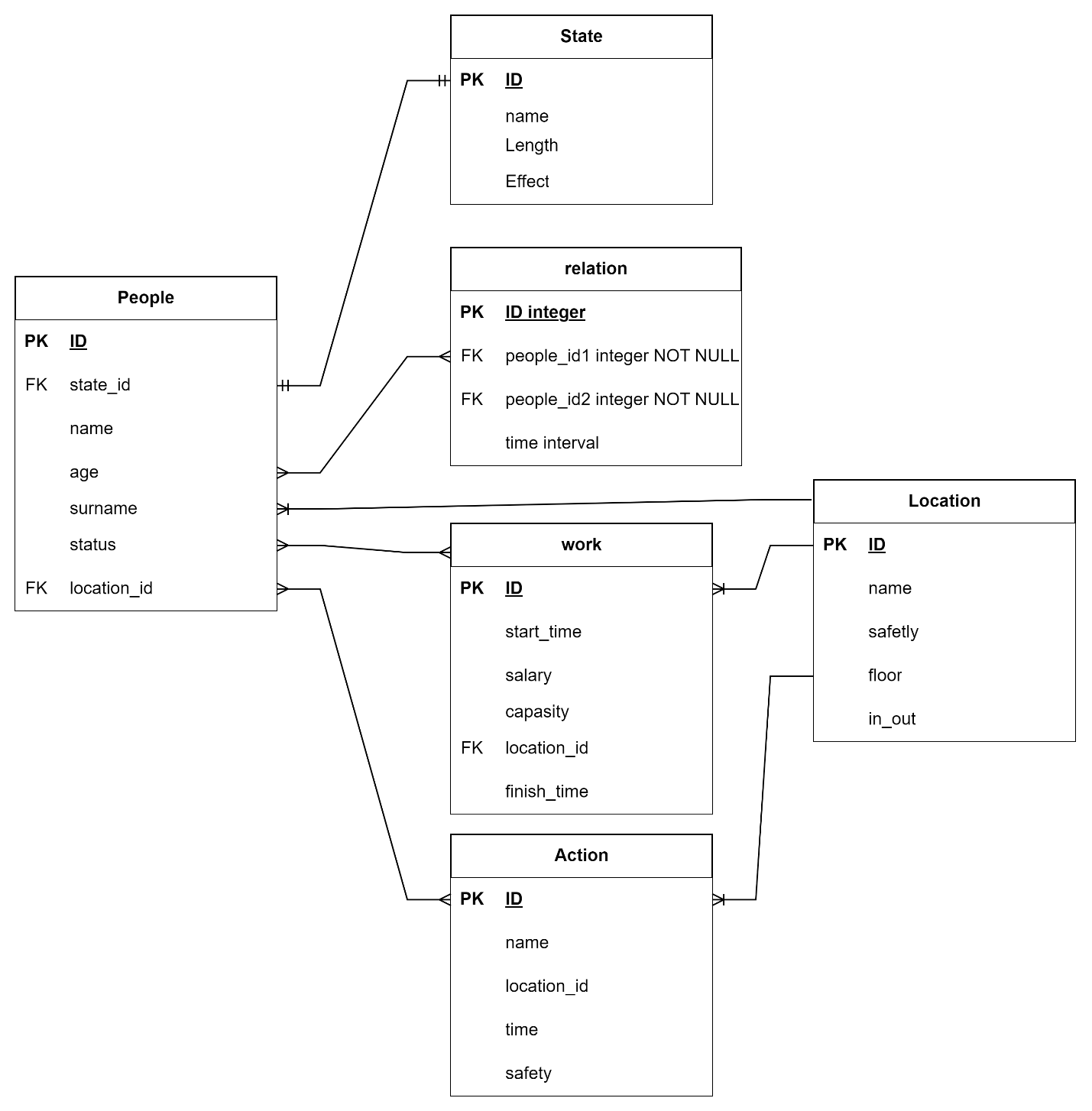
Relation – ассоциативная

Work – характеристика

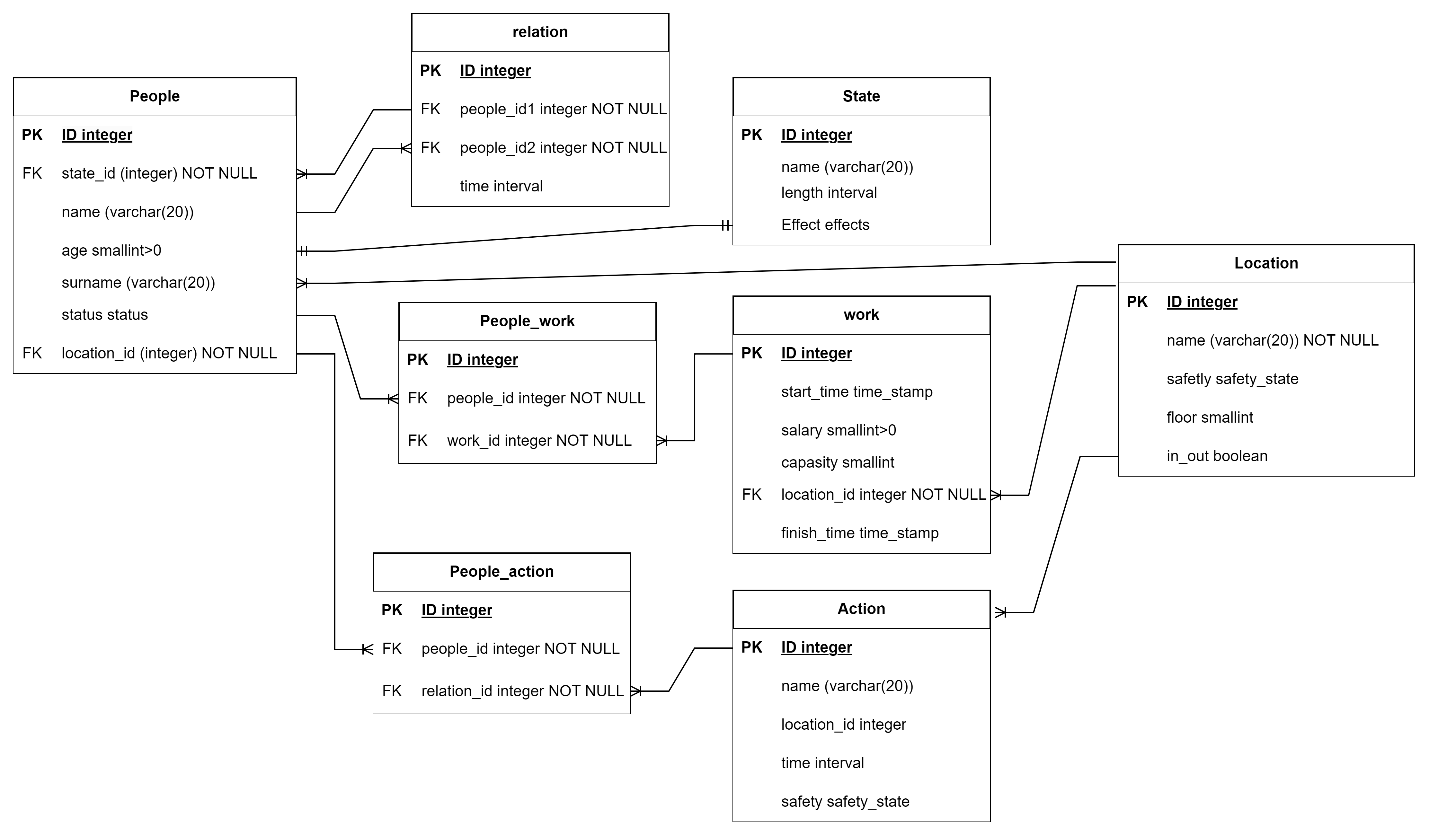
Location - характеристика

Action - характеристика

# Инфологическая модель



# Датологическая модель



# Реализация

CREATE TYPE safety\_state AS ENUM('безопасный', 'очень опасный','опасный');  
CREATE TYPE effects AS ENUM('бояться', 'радоваться','болеть', 'сидеть', 'стоять', 'лежать', 'бежать');  
  
CREATE TABLE location(  
 id int PRIMARY KEY,  
 name varchar(20) NOT NULL DEFAULT 'безымянный',  
 safety safety\_state,  
 floor smallint,  
 in\_out BOOLEAN DEFAULT 'true'  
);  
CREATE TABLE work(  
 id int PRIMARY KEY,  
 start\_time timestamp,  
 finish\_time timestamp,  
 salary smallint check ( salary>0 ),  
 capacity smallint check ( capacity>0 ),  
 location\_id int NOT NULL references location(id)  
  
);  
CREATE TABLE action(  
 id int PRIMARY KEY,  
 name varchar(20),  
 location\_id int NOT NULL references location(id),  
 time interval,  
 safety safety\_state  
);  
CREATE TABLE state(  
 id int PRIMARY KEY,  
 name varchar(20) NOT NULL DEFAULT 'безымянный',  
 length interval,  
 effect effects  
);  
CREATE TABLE people(  
 id int PRIMARY KEY,  
 name varchar(20) NOT NULL DEFAULT 'безымянный',  
 surname varchar(20),  
 age smallint check ( age>0 ),  
 state\_id int NOT NULL references state(id),  
 location\_id int NOT NULL references location(id)  
);  
CREATE TABLE relation(  
 id int PRIMARY KEY,  
 people\_id1 int NOT NULL references people(id),  
 people\_id2 int NOT NULL references people(id),  
 time interval  
);  
  
CREATE TABLE people\_action(  
 id int PRIMARY KEY,  
 people\_id int NOT NULL references people(id),  
 action\_id int NOT NULL references action(id)  
);  
CREATE TABLE people\_work(  
 id int PRIMARY KEY,  
 people\_id int NOT NULL references people(id),  
 work\_id int NOT NULL references work(id)  
);

# Заполнение значений

INSERT INTO location (id, name, safety, floor, in\_out)  
VALUES (1, 'школа', 'безопасный', 2, false),  
 (2, 'университет', 'опасный', 3, false);  
INSERT INTO location (id, name)  
VALUES (3,'магазин' ),  
 (4, 'метро');  
  
INSERT INTO state(id, name, length, effect)  
VALUES (1, 'отчаяние','3 days 4 hours','радоваться' ),  
 (2, 'боязнь', '1 days', 'сидеть' );  
INSERT INTO state(id, name)  
VALUES (3, 'учится'),  
 (4, 'идёт домой');  
  
INSERT INTO work(id, start\_time, finish\_time, salary, capacity, location\_id)  
VALUES (1, '2023-10-01 8:30:00','2023-10-01 18:30:00', 20000, 100, 3 ),  
 (2,'2025-11-03 9:50:00', '2025-11-03 20:20:00', 2100, 90, 2 );  
INSERT INTO work(id,location\_id)  
VALUES (3, 4),  
 (4, 1);  
  
insert into action(id, name, location\_id, time, safety)  
VALUES (1, 'работает',1,'1 hour','безопасный' ),  
 (2, 'спать', 2,'8 hour','опасный');  
insert into action(id, name, location\_id)  
VALUES (3, 'читает книжку', 4),  
 (4, 'принимаю ванну', 3);  
  
INSERT INTO people(id, state\_id, name, surname, age, location\_id)  
VALUES (1, 1, 'АНдрей', 'ыупи', 84,2),  
 (2, 2, 'Pavel', 'wfvk', 1, 4);  
INSERT INTO people(id, surname, age, state\_id, location\_id)  
VALUES (3, 'иванов', 2, 1, 2),  
 (4, 'ыеиыуф', 312, 2, 3);  
  
INSERT INTO relation(id, people\_id1, people\_id2, time)  
VALUES (1, 4, 1, '4 year'),  
 (2, 2, 4, '7 day'),  
 (3, 2, 4, '10 month'),  
 (4, 3, 2, '1 min');  
  
INSERT INTO people\_action(id, people\_id, action\_id)  
VALUES (1,1, 1),  
 (2, 1, 2),  
 (3, 4, 2),  
 (4, 2, 1);  
  
INSERT INTO people\_work(id, people\_id, work\_id)  
VALUES (1, 1, 3),  
 (2, 4, 1),  
 (3, 2, 4),  
 (4, 3, 2);

# Вывод

Я узнал, как работать на Postgesql и как создавать и заполнять таблицы.

# Ответы на вопросы

## 1. **Архитектура ANSI-SPARC**

Архитектура ANSI-SPARC (American National Standards Institute - Standards Planning and Requirements Committee) — это концептуальная модель, предложенная для стандартизации архитектуры систем управления базами данных (СУБД). Она состоит из трех уровней:

### Уровни архитектуры ANSI-SPARC:

1. **Внешний уровень (External Level)**:
   * Описывает представление данных для конкретных пользователей или приложений.
   * Каждый пользователь или приложение видит только те данные, которые ему нужны, в удобном формате.
   * Пример: Отчет для бухгалтерии, где выводятся только финансовые данные.
2. **Концептуальный уровень (Conceptual Level)**:
   * Описывает логическую структуру всей базы данных, независимо от физического хранения.
   * Включает все сущности, атрибуты, связи и ограничения.
   * Пример: ER-диаграмма, которая описывает все таблицы и связи между ними.
3. **Внутренний уровень (Internal Level)**:
   * Описывает физическое хранение данных на диске.
   * Включает структуры хранения, индексы, методы доступа и т.д.
   * Пример: Файлы базы данных, индексы B-дерева.

### Преимущества архитектуры ANSI-SPARC:

* **Независимость данных**: Изменения на одном уровне не влияют на другие уровни.
* **Гибкость**: Позволяет адаптировать базу данных под разные требования пользователей.
* **Безопасность**: Пользователи видят только те данные, которые им разрешены.

## 2. **Модель "Сущность-Связь" (ER-модель)**

Модель "Сущность-Связь" (Entity-Relationship Model) — это концептуальный инструмент для проектирования баз данных. Она используется для визуализации структуры данных и связей между ними.

### Основные компоненты ER-модели:

1. **Сущность (Entity)**:
   * Объект, который существует в реальном мире и может быть идентифицирован.
   * Пример: Студент, Курс, Преподаватель.
   * Сущности представляются в виде прямоугольников.
2. **Атрибут (Attribute)**:
   * Свойство сущности.
   * Пример: Для сущности "Студент" атрибутами могут быть ID, Имя, Фамилия.
   * Атрибуты представляются в виде овалов.
3. **Связь (Relationship)**:
   * Отношение между сущностями.
   * Пример: Студент "записан" на Курс.
   * Связи представляются в виде ромбов.

### Классификация сущностей:

* **Сильные сущности**: Сущности, которые могут существовать независимо (например, "Студент").
* **Слабые сущности**: Сущности, которые зависят от других сущностей (например, "Оценка" зависит от "Студента" и "Курса").

### Виды связей:

1. **Один к одному (1:1)**:
   * Одна сущность связана с одной сущностью.
   * Пример: Человек и Паспорт.
2. **Один ко многим (1:N)**:
   * Одна сущность связана с несколькими сущностями.
   * Пример: Преподаватель и Курсы.
3. **Многие ко многим (N:M)**:
   * Несколько сущностей связаны с несколькими сущностями.
   * Пример: Студенты и Курсы (студент может записаться на несколько курсов, и на курс может записаться несколько студентов).

## 3. **Ограничения целостности**

Ограничения целостности — это правила, которые обеспечивают корректность данных в базе данных. Они делятся на несколько типов:

1. **Ограничения домена (Domain Constraints)**:
   * Определяют допустимые значения для атрибутов.
   * Пример: Возраст не может быть отрицательным.
2. **Ограничения сущности (Entity Constraints)**:
   * Каждая сущность должна иметь уникальный идентификатор (первичный ключ).
   * Пример: У каждого студента должен быть уникальный ID.
3. **Ограничения ссылочной целостности (Referential Integrity Constraints)**:
   * Обеспечивают корректность связей между таблицами.
   * Пример: Если студент записан на курс, то курс должен существовать.
4. **Пользовательские ограничения (User-Defined Constraints)**:
   * Определяются разработчиком для конкретных бизнес-правил.
   * Пример: Зарплата сотрудника не может превышать зарплату его руководителя.

## 4. **DDL (Data Definition Language)**

DDL — это язык определения данных, используемый для создания и изменения структуры базы данных. Основные команды DDL:

1. **CREATE**:
   * Создает новые объекты (таблицы, индексы, базы данных).
   * Пример:

sql

Copy

CREATE TABLE Students (

ID INT PRIMARY KEY,

Name VARCHAR(50),

Age INT

);

1. **ALTER**:
   * Изменяет структуру существующих объектов.
   * Пример:

sql

Copy

ALTER TABLE Students ADD COLUMN Email VARCHAR(100);

1. **DROP**:
   * Удаляет объекты из базы данных.
   * Пример:

sql

Copy

DROP TABLE Students;

1. **TRUNCATE**:
   * Удаляет все данные из таблицы, но сохраняет её структуру.
   * Пример:

sql

Copy

TRUNCATE TABLE Students;

## 5. **DML (Data Manipulation Language)**

DML — это язык манипуляции данными, используемый для работы с данными в таблицах. Основные команды DML:

1. **SELECT**:
   * Извлекает данные из таблицы.
   * Пример:

sql

Copy

SELECT \* FROM Students;

1. **INSERT**:
   * Добавляет новые строки в таблицу.
   * Пример:

sql

Copy

INSERT INTO Students (ID, Name, Age) VALUES (1, 'Alice', 20);

1. **UPDATE**:
   * Обновляет существующие данные в таблице.
   * Пример:

sql

Copy

UPDATE Students SET Age = 21 WHERE ID = 1;

1. **DELETE**:
   * Удаляет строки из таблицы.
   * Пример:

sql

Copy

DELETE FROM Students WHERE ID = 1;