Part 1: Выбранный сценарий
Для данной работы выбран сценарий: Управление строительными проектами: Управление проектами, сотрудниками, клиентами и задачами проекта.
Part 2: Проектирование Базы Данных и Документация
Идентификация сущностей и ключевых атрибутов

1. **Clients (Клиенты)** - идентификация и контактные данные заказчика.
2. **Roles (Роли сотрудников)** - справочник ролей (Project Manager, Site Engineer и т.д.).
3. **Employees (Сотрудники)** - данные сотрудников, роль, менеджер.
4. **Projects (Проекты)** - проект, связанный с клиентом, бюджет, даты, статус.
5. **Tasks (Задачи проекта)** - задачи внутри проекта, оценка времени, приоритет, статус.
6. **Project_assignments (Назначения на проекты)** - M:N между проектами и сотрудниками.
7. **Task_assignments (Назначения на задачи)** - M:N между задачами и сотрудниками.

Проектирование таблиц (имя, описание, атрибуты: тип, PK/FK, ограничения)
1) Table Name: clients

- **Description:** Информация о клиентах.
- **Attributes:**
  - client_id - SERIAL - **PK**, NOT NULL (уникальный идентификатор).
  - company_name - VARCHAR(255) - NOT NULL.
  - contact_name - VARCHAR(255) - NULLABLE.
  - email - VARCHAR(255) - NULLABLE, **UNIQUE** (clients_email_unique).
  - phone - VARCHAR(50) - NULLABLE.
  - address - TEXT - NULLABLE.
- **Constraints:** PK on client_id; UNIQUE on email.

2) Table Name: roles

- **Description:** Роли сотрудников.
- **Attributes:**
  - role_id - SERIAL - **PK**, NOT NULL.
  - role_name - VARCHAR(100) - NOT NULL, **UNIQUE**.
  - description - TEXT - NULLABLE.
- **Constraints:** PK on role_id; UNIQUE on role_name.

3) Table Name: employees

- **Description:** Данные сотрудников.
- **Attributes:**
  - employee_id - SERIAL - **PK**, NOT NULL.
  - role_id - INT - **FK** → roles(role_id), NOT NULL.
  - first_name - VARCHAR(100) - NOT NULL.
  - last_name - VARCHAR(100) - NOT NULL.
  - email - VARCHAR(255) - NOT NULL, **UNIQUE**.
  - phone - VARCHAR(50) - NULLABLE.
  - hire_date - DATE - NOT NULL.
  - manager_id - INT - **FK (self)** → employees(employee_id), NULLABLE.
- **Constraints:**
  - CONSTRAINT fk_employee_role FOREIGN KEY (role_id) REFERENCES roles(role_id) ON DELETE RESTRICT - роль не удалится, если есть сотрудник.
  - CONSTRAINT fk_employee_manager FOREIGN KEY (manager_id) REFERENCES employees(employee_id) ON DELETE SET NULL - при удалении менеджера поле у подчинённых станет NULL.
  - UNIQUE на email.

4) Table Name: projects

- **Description:** Сведения о проектах и привязка к клиенту.
- **Attributes:**
  - project_id - SERIAL - **PK**, NOT NULL.
  - client_id - INT - **FK** → clients(client_id), NOT NULL.
  - name - VARCHAR(255) - NOT NULL.
  - description - TEXT - NULLABLE.
  - start_date - DATE - NOT NULL.
  - end_date - DATE - NULLABLE.
  - budget - NUMERIC(12,2) - DEFAULT 0, CHECK (budget >= 0).
  - status - VARCHAR(20) - NOT NULL, DEFAULT 'planned'.
- **Constraints:**
  - CONSTRAINT fk_project_client FOREIGN KEY (client_id) REFERENCES clients(client_id) ON DELETE CASCADE - при удалении клиента удалять его проекты.
  - CONSTRAINT chk_project_dates CHECK (end_date IS NULL OR end_date >= start_date) - конец не раньше старта.

5) Table Name: tasks

- **Description:** Задачи внутри проекта.
- **Attributes:**
  - task_id - SERIAL - **PK**, NOT NULL.
  - project_id - INT - **FK** → projects(project_id), NOT NULL.
  - title - VARCHAR(255) - NOT NULL.
  - description - TEXT - NULLABLE.
  - status - VARCHAR(20) - NOT NULL, DEFAULT 'todo'.
  - priority - VARCHAR(10) - NOT NULL, DEFAULT 'medium'.
  - estimated_hours - NUMERIC(7,2) - DEFAULT 0, CHECK (estimated_hours >= 0).
  - actual_hours - NUMERIC(7,2) - DEFAULT 0, CHECK (actual_hours >= 0).
  - due_date - DATE - NULLABLE.
  - parent_task_id - INT - **FK (self)** → tasks(task_id), NULLABLE.
- **Constraints:**
  - CONSTRAINT fk_task_project FOREIGN KEY (project_id) REFERENCES projects(project_id) ON DELETE CASCADE.
  - CONSTRAINT fk_task_parent FOREIGN KEY (parent_task_id) REFERENCES tasks(task_id) ON DELETE SET NULL - при удалении родителя дочерняя задача станет корневой (parent=NULL).
  - НЕОБЯЗАТЕЛЬНО: можно убрать parent_task_id, если иерархия не нужна.

6) Table Name: project_assignments

- **Description:** Назначения сотрудников на проекты (реализует M:N).
- **Attributes:**
  - project_id - INT - **FK** → projects(project_id), NOT NULL.
  - employee_id - INT - **FK** → employees(employee_id), NOT NULL.
  - assigned_at - TIMESTAMP WITHOUT TIME ZONE - DEFAULT now().
  - role_on_project - VARCHAR(100) - NULLABLE (роль в конкретном проекте).
- **Constraints:**
  - PRIMARY KEY (project_id, employee_id) - композитный PK, предотвращает дубликаты назначений.
  - FK с ON DELETE CASCADE (в SQL: fk_pa_project, fk_pa_employee).

7) Table Name: task_assignments

- **Description:** Назначения сотрудников на задачи (реализует M:N).
- **Attributes:**
  - task_id - INT - **FK** → tasks(task_id), NOT NULL.

    - employee_id - INT - **FK** → employees(employee_id), NOT NULL.
    - assigned_at - TIMESTAMP WITHOUT TIME ZONE - DEFAULT now().
    - allocation_percent - SMALLINT - CHECK (allocation_percent >= 0 AND allocation_percent <= 100), NULLABLE.
- **Constraints:**
    - PRIMARY KEY (task_id, employee_id) - композитный PK.
    - FK с ON DELETE CASCADE (fk_ta_task, fk_ta_employee).

3. Объяснение связей и их мощность

- **clients - projects**: **один-ко-многим** - один клиент (clients.client_id) может иметь много проектов (projects.client_id). Реализовано FK projects.client_id → clients.client_id.
- **projects - tasks**: **один-ко-многим** - один проект может содержать много задач. Реализовано FK tasks.project_id → projects.project_id.
- **employees - roles**: **многие-к-одному** - многие сотрудники имеют одну роль. Реализовано FK employees.role_id → roles.role_id.
- **employees - employees (manager)**: **один-ко-многим (self-reference)** - у сотрудника может быть менеджер (который тоже сотрудник). Реализовано FK employees.manager_id → employees.employee_id. Поле nullable.
- **projects - employees**: **многие-ко-многим** - сотрудник может работать в нескольких проектах; проект может иметь многих сотрудников. Реализовано через project_assignments (композитный PK (project_id, employee_id) и два FK).
- **tasks - employees**: **многие-ко-многим** - одна задача может быть распределена на нескольких сотрудников; сотрудник работает над многими задачами. Реализовано через task_assignments (композитный PK (task_id, employee_id)).
- **tasks - tasks (parent_task_id)**: **один-ко-многим (self-reference)** - родительская задача → подзадачи. parent_task_id nullable; FK tasks.parent_task_id → tasks.task_id.