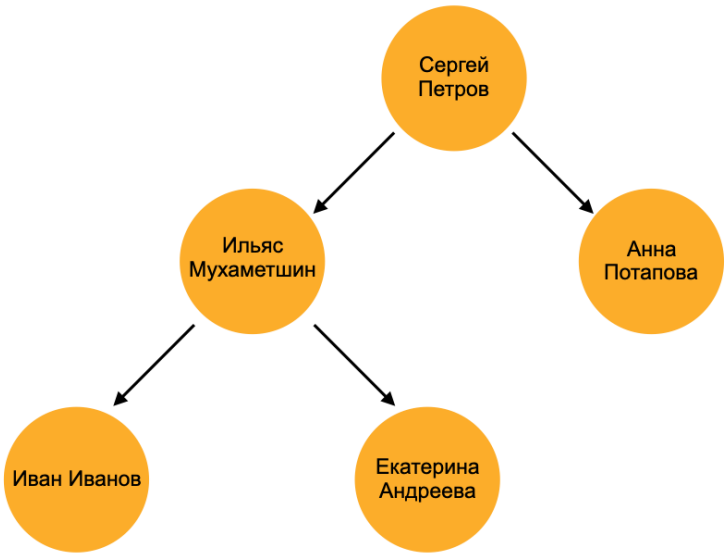
a. Все зарегистрированные студенты познакомились друг с другом и решили организовать свой собственный технологический стартап. Ребята выстроили между собой следующие взаимоотношения в рамках проектной деятельности.

- Сергей Петров - лидер проекта
- Ильяс Мухаметшин - тимлид команды разработчиков
- Иван Иванов - frontend разработчик
- Екатерина Андреева - backend разработчик
- Анна Потапова - тимлид команды специалистов по тестированию




В связи с этой активностью необходимо расширить модель данных

1) Добавьте отдельную таблицу с зависимостями между участниками стартапа. Укажите внешние ключи необходимые для поддержания консистентности данных. Заполните таблицу соответствующими записями по зависимостям.

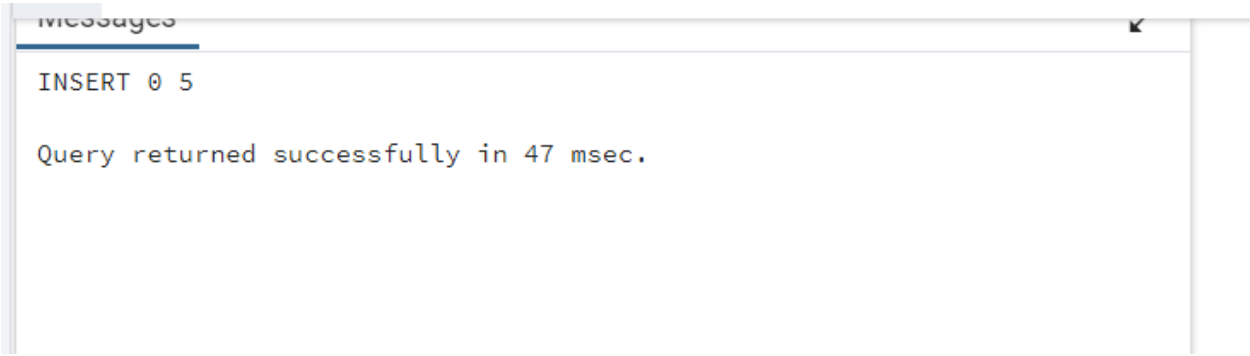
2) Добавьте таблицу-справочник, хранящую роли участников проекта и использующуюся как источник для понимания, какой студент имеет ту или иную роль. Заполните таблицу соответствующими записями.

**1) Код запроса:**

```
CREATE TABLE Participants (
 ID INT PRIMARY KEY,
 Name VARCHAR(255),
```

```
  ParentID INT,
  RoleID INT,
  FOREIGN KEY (ParentID) REFERENCES Participants(ID),
  FOREIGN KEY (RoleID) REFERENCES Role(ID)
);

INSERT INTO Participants (ID, Name, ParentID, RoleID) VALUES
  (1, 'Сергей Петров', NULL, 1),
  (2, 'Анна Потапова', 1, 2),
  (3, 'Ильяс Мухаметшин', 1, 3),
  (4, 'Иван Иванов', 3, 4),
  (5, 'Екатерина Андреева', 3, 5);
```

**Результат запроса:**



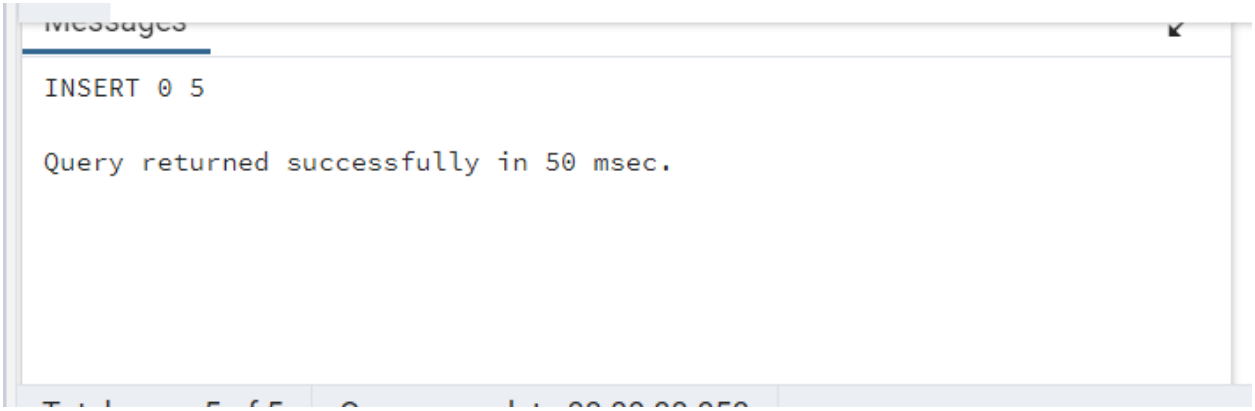

| | id [PK] integer | name character varying (255) | parentid integer | roleid integer |
|---|---|---|---|---|
| 1 | 1 | Сергей Петров | [null] | 1 |
| 2 | 2 | Анна Потапова | 1 | 2 |
| 3 | 3 | Ильяс Мухаметшин | 1 | 3 |
| 4 | 4 | Иван Иванов | 3 | 4 |
| 5 | 5 | Екатерина Андреева | 3 | 5 |

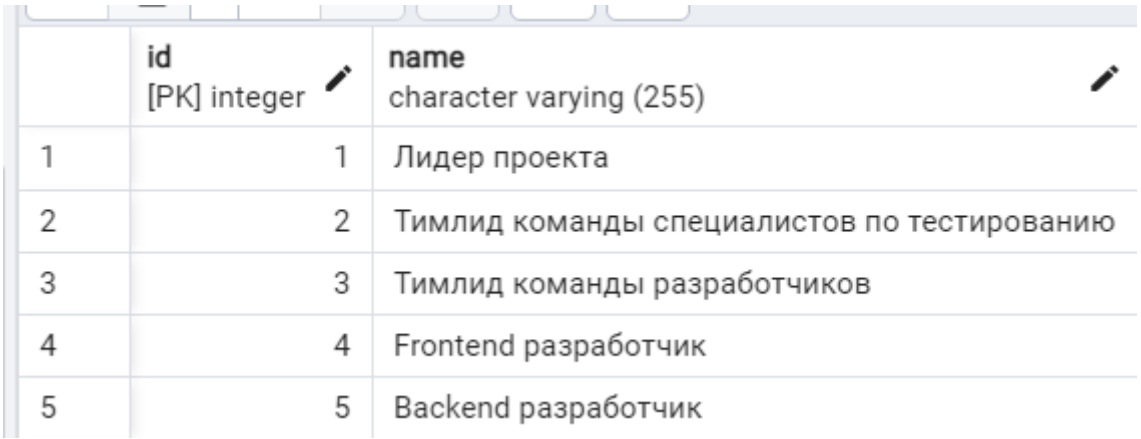
**2) Код запроса:**

```
CREATE TABLE Role (
  ID INT PRIMARY KEY,
  Name VARCHAR(255)
);
```

```
INSERT INTO Role (ID, Name) VALUES
 (1, 'Лидер проекта'),
 (2, 'Тимлид команды специалистов по тестированию'),
 (3, 'Тимлид команды разработчиков'),
 (4, 'Frontend разработчик'),
 (5, 'Backend разработчик');
```

**Результат запроса:**



| | id [PK] integer | name character varying (255) |
|---|---|---|
| 1 | 1 | Лидер проекта |
| 2 | 2 | Тимлид команды специалистов по тестированию |
| 3 | 3 | Тимлид команды разработчиков |
| 4 | 4 | Frontend разработчик |
| 5 | 5 | Backend разработчик |

b. Напишите SQL запрос который возвращает имена студентов, их роль на проекте и уровень вложенности от “лидера всего проекта” в следующем виде.

| **студент** | **роль** | **уровень** | **путь от лидера** |
|---|---|---|---|
| Сергей Петров | лидер проекта | 1 | Сергей Петров |

| | | | |
|---|---|---|---|
| Ильяс Мухаметшин | тимлид команды разработчиков | 2 | Сергей Петров -> Ильяс Мухаметшин |
| Анна Потапова | тимлид команды специалистов по тестированию | 2 | Сергей Петров ->Анна Потапова |
| Екатерина Андреева | backend разработчик | 3 | Сергей Петров -> Ильяс Мухаметшин ->Екатерина Андреева |
| Иван Иванов | frontend разработчик | 3 | Сергей Петров -> Ильяс Мухаметшин -> Иван Иванов |

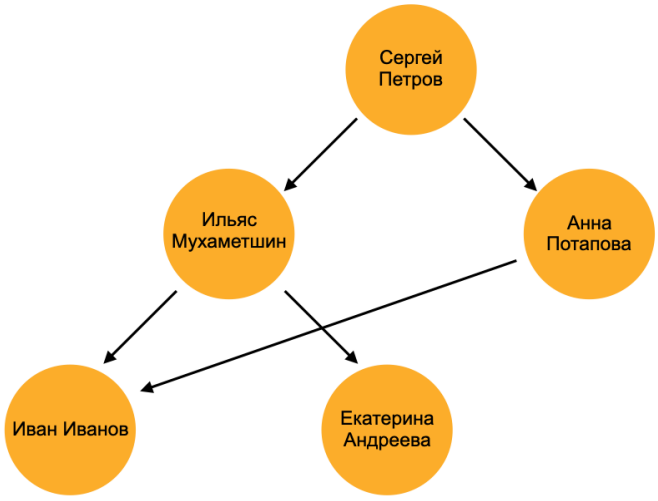
**Код запроса:**

```
WITH RECURSIVE EmployeeHierarchy AS (
  SELECT
    p.ID,
    p.Name AS Студент,
    r.Name AS Роль,
    1 AS Уровень,
    CAST(p.Name AS VARCHAR(255)) AS "Путь от лидера"
  FROM Participants p
  JOIN Role r ON p.RoleID = r.ID
  WHERE p.ParentID IS NULL

  UNION ALL

  SELECT
    p.ID,
    p.Name AS Студент,
    r.Name AS Роль,
    eh.Уровень + 1 AS Уровень,
    CAST(COALESCE(eh."Путь от лидера", '') || ' -> ' || p.Name AS VARCHAR(255))
  FROM Participants p
  JOIN EmployeeHierarchy eh ON p.ParentID = eh.ID
  JOIN Role r ON p.RoleID = r.ID
)
SELECT
  Студент,
  Роль,
  Уровень,
  "Путь от лидера"
FROM EmployeeHierarchy
ORDER BY Уровень, Студент
```

**Результат запроса:**

| | Студент character varying (255) | Роль character varying (255) | Уровень integer | Путь от лидера character varying (255) |
|---|---|---|---|---|
| 1 | Сергей Петров | Лидер проекта | 1 | Сергей Петров |
| 2 | Анна Потапова | Тимлид команды специалистов по тестированию | 2 | Сергей Петров -> Анна Потапова |
| 3 | Ильяс Мухаметшин | Тимлид команды разработчиков | 2 | Сергей Петров -> Ильяс Мухаметшин |
| 4 | Екатерина Андреева | Backend разработчик | 3 | Сергей Петров -> Ильяс Мухаметшин -> Екатерина Андрее... |
| 5 | Иван Иванов | Frontend разработчик | 3 | Сергей Петров -> Ильяс Мухаметшин -> Иван Иванов |

c. Студент Иван Иванов решил параллельно выполнять роль “QA тестировщик” на проекте и стал подчиняться Анне Потаповой. Сделайте соответствующие изменения в данных вашей модели, чтобы выполнив запрос из пункта b) вы получили следующий результат.



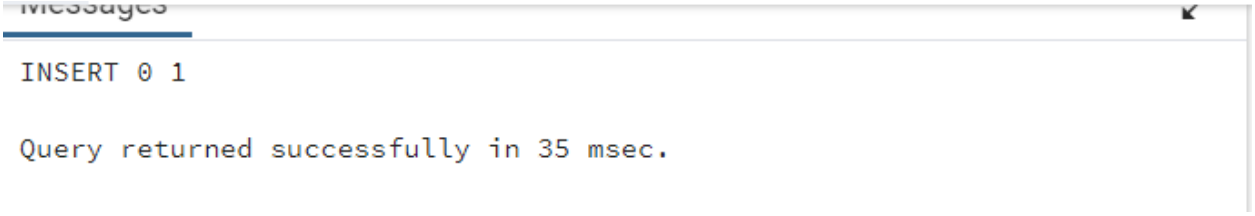

| **студент** | **роль** | **уровень** | **путь от лидера** |
|---|---|---|---|
| Сергей Петров | лидер проекта | 1 | Сергей Петров |
| Ильяс Мухаметшин | тимлид команды разработчиков | 2 | Сергей Петров -> Ильяс Мухаметшин |
| Анна Потапова | тимлид команды специалистов по тестированию | 2 | Сергей Петров ->Анна Потапова |
| Екатерина Андреева | backend разработчик | 3 | Сергей Петров -> Ильяс Мухаметшин ->Екатерина Андреева |

| | | | |
|---|---|---|---|
| Иван Иванов | frontend разработчик | 3 | Сергей Петров -> Ильяс Мухаметшин -> Иван Иванов |
| Иван Иванов | QA тестировщик | 3 | Сергей Петров->Анна Потапова->Иван Иванов |

**Код запроса вставки новой роли:**

```
INSERT INTO Role (ID, Name)
VALUES (6, 'QA тестировщик');
```

**Результат запроса:**




| | id [PK] integer | name character varying (255) |
|---|---|---|
| 1 | 1 | Лидер проекта |
| 2 | 2 | Тимлид команды специалистов по тестированию |
| 3 | 3 | Тимлид команды разработчиков |
| 4 | 4 | Frontend разработчик |
| 5 | 5 | Backend разработчик |
| 6 | 6 | QA тестировщик |

**Код запроса вставки новой связи:**

```
INSERT INTO Participants (ID, Name, ParentID, RoleID)
VALUES (
    6,
    'Иван Иванов',
    (SELECT ID FROM Participants WHERE Name = 'Анна Потапова'),
    (SELECT ID FROM Role WHERE Name = 'QA тестировщик')
```

)

**Результат запроса:**

```
INSERT 0 1

Query returned successfully in 54 msec.
```

| | id [PK] integer | name character varying (255) | parentid integer | roleid integer |
|---|---|---|---|---|
| 1 | 1 | Сергей Петров | [null] | 1 |
| 2 | 2 | Анна Потапова | 1 | 2 |
| 3 | 3 | Ильяс Мухаметшин | 1 | 3 |
| 4 | 4 | Иван Иванов | 3 | 4 |
| 5 | 5 | Екатерина Андреева | 3 | 5 |
| 6 | 6 | Иван Иванов | 2 | 6 |

**Результат выполнения запроса из пункта b:**

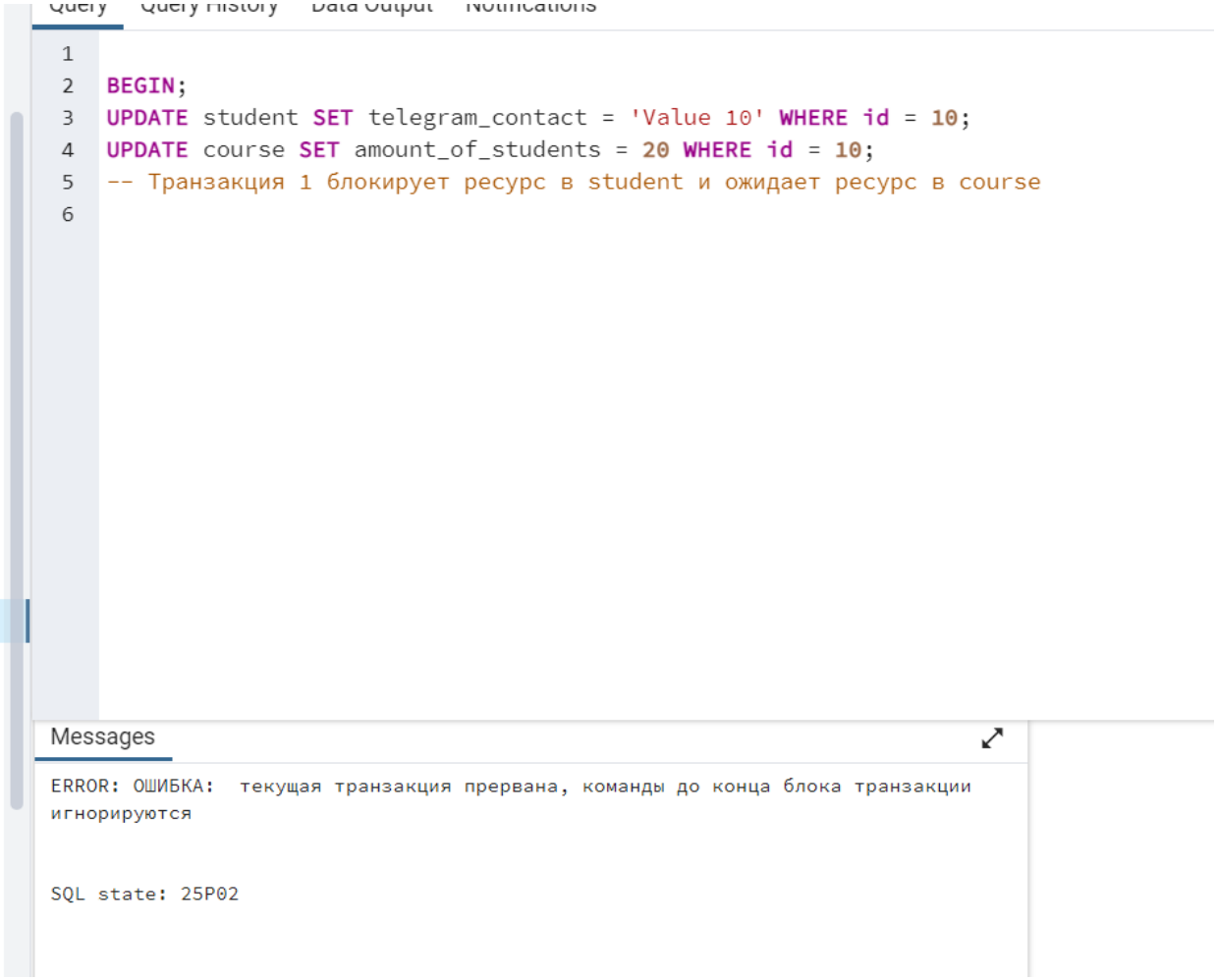
| | Студент character varying (255) | Роль character varying (255) | Уровень integer | Путь от лидера character varying (255) |
|---|---|---|---|---|
| 1 | Сергей Петров | Лидер проекта | 1 | Сергей Петров |
| 2 | Анна Потапова | Тимлид команды специалистов по тестированию | 2 | Сергей Петров -> Анна Потапова |
| 3 | Ильяс Мухаметшин | Тимлид команды разработчиков | 2 | Сергей Петров -> Ильяс Мухаметшин |
| 4 | Екатерина Андреева | Backend разработчик | 3 | Сергей Петров -> Ильяс Мухаметшин -> Екатерина Андрее... |
| 5 | Иван Иванов | Frontend разработчик | 3 | Сергей Петров -> Ильяс Мухаметшин -> Иван Иванов |
| 6 | Иван Иванов | QA тестировщик | 3 | Сергей Петров -> Анна Потапова -> Иван Иванов |

d. Смоделируйте ситуацию deadlock на любой таблице в вашей базе данных. Предоставьте ошибку (сделав скриншот), которая доказывает, что ситуация “мертвой блокировки” случилась в рамках параллельных транзакций.

**Код запроса первой транзакции:**

```
BEGIN;
UPDATE student SET telegram_contact = 'Value 10' WHERE id = 10;
UPDATE course SET amount_of_students = 20 WHERE id = 10;

BEGIN;
UPDATE course SET amount_of_students = 30 WHERE id = 10;
UPDATE student SET telegram_contact = 'Value 40' WHERE id = 10;
```

**Результат запроса:**



e. Смоделируйте ситуацию блокировок на уровне изоляции данных SERIALIZABLE на любой таблице в вашей базе данных. Предоставьте ошибку (сделав скриншот), которая доказывает что уровень изоляции SERIALIZABLE предотвращает обновление данных, которое случилось в параллельной транзакции с вашей.

**Код запроса:**

```
-- создаем таблицу для примера
    CREATE TABLE products (
      id SERIAL PRIMARY KEY,
      name VARCHAR(100)
    );
    -- вставляем данные
    INSERT INTO products (name) VALUES ('Product 1'), ('Product 2');
    -- Открытие транзакций
    BEGIN;
    BEGIN;
    -- Обновление в первой транзакции
    UPDATE products SET name = 'Updated Product 1' WHERE id = 1;
    -- Попытка обновления во второй транзакции
    UPDATE products SET name = 'Updated Product 1' WHERE id = 1;
```

**Результат запроса:**