МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Национальный исследовательский университет ИТМО»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И  
КОМПЬЮТЕРНОЙ ТЕХНИКИ

**ЛАБОРАТОРНАЯ РАБОТА №3**

по дисциплине

«БАЗЫ ДАННЫХ»

***Выполнил:***Рязанов Никита Сергеевич

***Проверила:***Байрамова Хумай Бахруз Кызы

**Содержание**

[Задание 3](#_Toc195640932)

[Предметная область 4](#_Toc195640933)

[Заключение 8](#_Toc195640934)

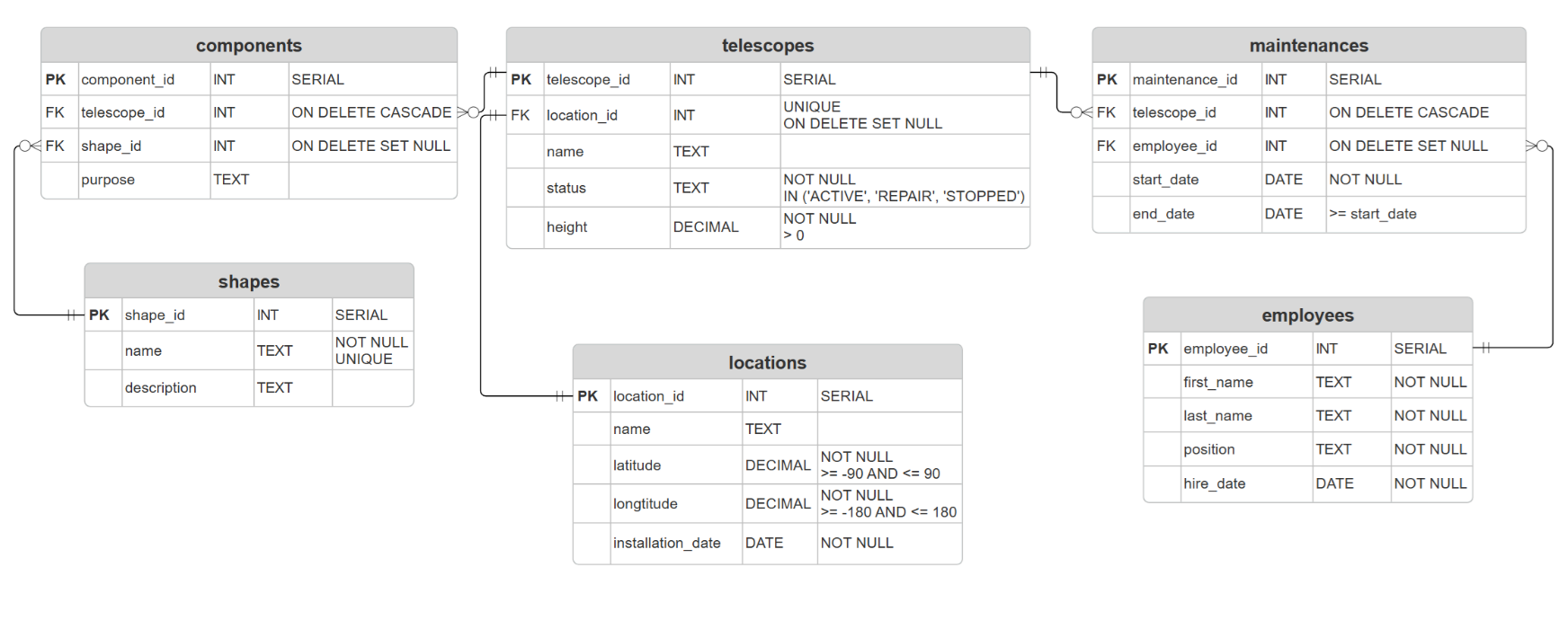
Задание

Для отношений, полученных при построении предметной области из лабораторной работы №1, выполните следующие действия:

* Опишите функциональные зависимости для отношений полученной схемы (минимальное множество);
* Приведите отношения в 3NF (как минимум). Постройте схему на основе NF (как минимум);
* Опишите изменения в функциональных зависимостях, произошедшие после преобразования в 3NF (как минимум). Постройте схему на основе NF;
* Преобразуйте отношения в BCNF. Докажите, что полученные отношения представлены в BCNF. Если ваша схема находится уже в BCNF, докажите это;
* Какие денормализации будут полезны для вашей схемы? Приведите подробное описание.

Придумайте триггер и связанную с ним функцию, относящиеся к вашей предметной области, согласуйте их с преподавателем и реализуйте на языке PL/pgSQL.

Предметная область

Даталогическая модель

Функциональные зависимости

**telescopes**:

* telescope\_id → name
* telescope\_id → status
* telescope\_id → height
* telescope\_id → location\_id

**shapes**:

* shape\_id → name
* shape\_id → description
* name → description

**locations**:

* location\_id → name
* location\_id → latitude
* location\_id → longitude
* location\_id → installation\_date

**employees**:

* employee\_id → first\_name
* employee\_id → last\_name
* employee\_id → position
* employee\_id → hire\_date

**components**:

* component\_id → telescope\_id
* component\_id → shape\_id
* component\_id → purpose

**maintenances**:

* maintenance\_id → telescope\_id
* maintenance\_id → employee\_id
* maintenance\_id → start\_date
* maintenance\_id → end\_date

Нормализация

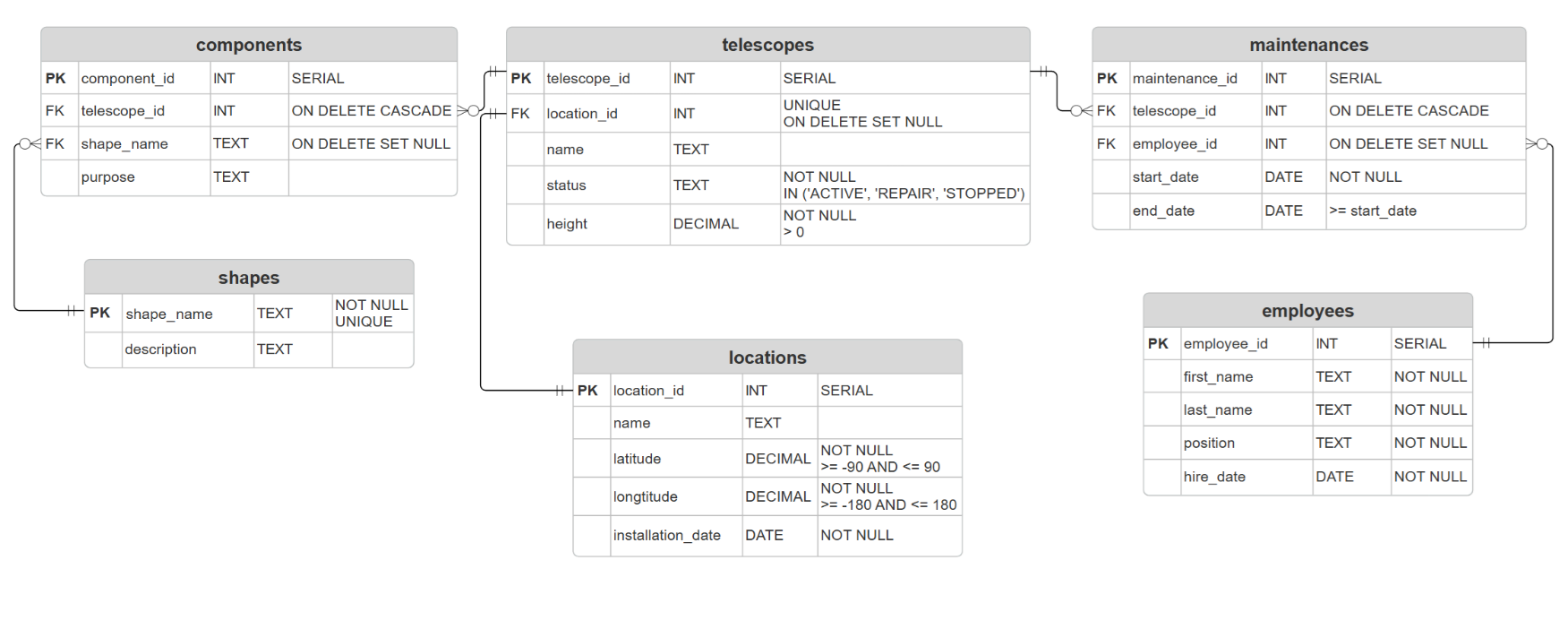
**1NF:**

Отношение в 1НФ, если на пересечении каждой строки и столбца – одно значение. Все таблицы соответствуют данному условию, изменения не требуются.

**2NF:**

Отношение в 2НФ, если отношение в 1НФ и все атрибуты не из первичного ключа находятся в полной функциональной зависимости от него. Во всех таблицах первичный ключ не имеет подмножеств, поэтому модель удовлетворяют условиям и никакие изменения не требуются.

**3NF:**

Отношение в 3НФ, если отношение в 2НФ и все атрибуты не из первичного ключа не находятся в транзитивной функциональной зависимости от него. В отношении **shapes** есть транзитивная зависимость shape\_id → name → description (т. к. атрибут name выступает как дополнительный ключ из-за уникальности и не-null значения), поэтому модель не удовлетворяет условиям 3НФ, если не внести следующие изменения.

Теперь в качестве первичного ключа в **shapes** выступает shape\_name, что исключает возникшую транзитивность.

Функциональные зависимости поменялось только в **shapes**:

* shape\_name → description

**BCNF:**

Отношение в НФБК, если отношение в 3НФ и для всех функциональных зависимостей отношения детерминантом является суперключ. Во всех таблицах функциональные зависимости исходят от суперключа, поэтому модель находится в НФБК и не требует дополнительных изменений.

Денормализация

**Объединение сущностей:**

Для ускорения выполнения запросов можно сократить количество сущностей путем их объединения, что позволит уменьшить число соединений таблиц за счет увеличение избыточности данных. Так в данной модели можно объединить **telescopes** и **locations**, тем самым информация о расположении телескопа будет храниться с ним же. При этом нарушится 2НФ, так как у нас появится частичная зависимость:

* (~~telescope\_id~~, location\_id) → (name, latitude, longitude, installation\_date)

**Добавление атрибутов:**

Для ускорения различных подсчетов и запросов можно добавить дополнительные атрибуты. Например, в **telescopes** можно создать атрибут components\_count, который будет хранить количество компонентов в данном телескопе, таким образом не придется каждый раз обращаться к **components** для их подсчета. Или в **employees** добавить атрибут **is\_working**, который будет показывать, обслуживает ли работник какой-либо телескоп в данный момент или нет.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Триггер

Идея триггера заключается в том, что если телескоп переводится в статус **'REPAIR'**, то автоматически создается запись о техобслуживании начиная с текущей даты свободным в данный момент сотрудником. Если свободных сотрудников нет, то просто выводим сообщение об этом.

|  |
| --- |
| CREATE OR REPLACE FUNCTION create\_maintenance\_on\_repair()  RETURNS TRIGGER AS $$  DECLARE  free\_employee\_id INT;  BEGIN  IF NEW.status = 'REPAIR' AND OLD.status IS DISTINCT FROM 'REPAIR' THEN  SELECT employee\_id  INTO free\_employee\_id  FROM employees  WHERE NOT EXISTS (  SELECT 1  FROM maintenances  WHERE (employees.employee\_id = maintenances.employee\_id)  AND (maintenances.end\_date IS NULL OR maintenances.end\_date > CURRENT\_DATE)  )  LIMIT 1;  IF free\_employee\_id IS NOT NULL THEN  INSERT INTO maintenances (telescope\_id, employee\_id, start\_date, end\_date)  VALUES (NEW.telescope\_id,free\_employee\_id,CURRENT\_DATE,NULL);  ELSE  RAISE NOTICE 'Нет доступных сотрудников для техобслуживания.';  END IF;  END IF;  RETURN NEW;  END;  $$ LANGUAGE plpgsql;    CREATE TRIGGER trigger\_create\_maintenance  AFTER UPDATE ON telescopes  FOR EACH ROW  EXECUTE FUNCTION create\_maintenance\_on\_repair(); |

Заключение

В ходе проделанной лабораторной работы было изучено что такое нормализация, а также нормальные формы 1NF, 2NF, 3NF, BCNF. На основе предметной области был написан собственный триггер. Кроме того, были изучены полезные способы денормализации.