Санкт-Петербургский Национальный Исследовательский Университет ИТМО

Факультет Программной Инженерии и Компьютерной Техники



Вариант № 9320

**Лабораторная работа №3**

По дисциплине

Базы Данных

Выполнил студент группы P3121:

Фам Данг Чунг Нгиа

Преподаватель:

Карапетян Эрик Акопович

Санкт-Петербург 2024 г.

**Заголовок**

[**1.** **Текст задания** 2](#_Toc166955929)

[**2.** **Исходная, нормализованная и денормализованная модели.** 2](#_Toc166955930)

[**3.** **Полезная денормализация** 4](#_Toc166955931)

[**4.** **Функция и триггер на языке PL/pgSQL** 5](#_Toc166955932)

[**5.** **Вывод по работе** 6](#_Toc166955933)

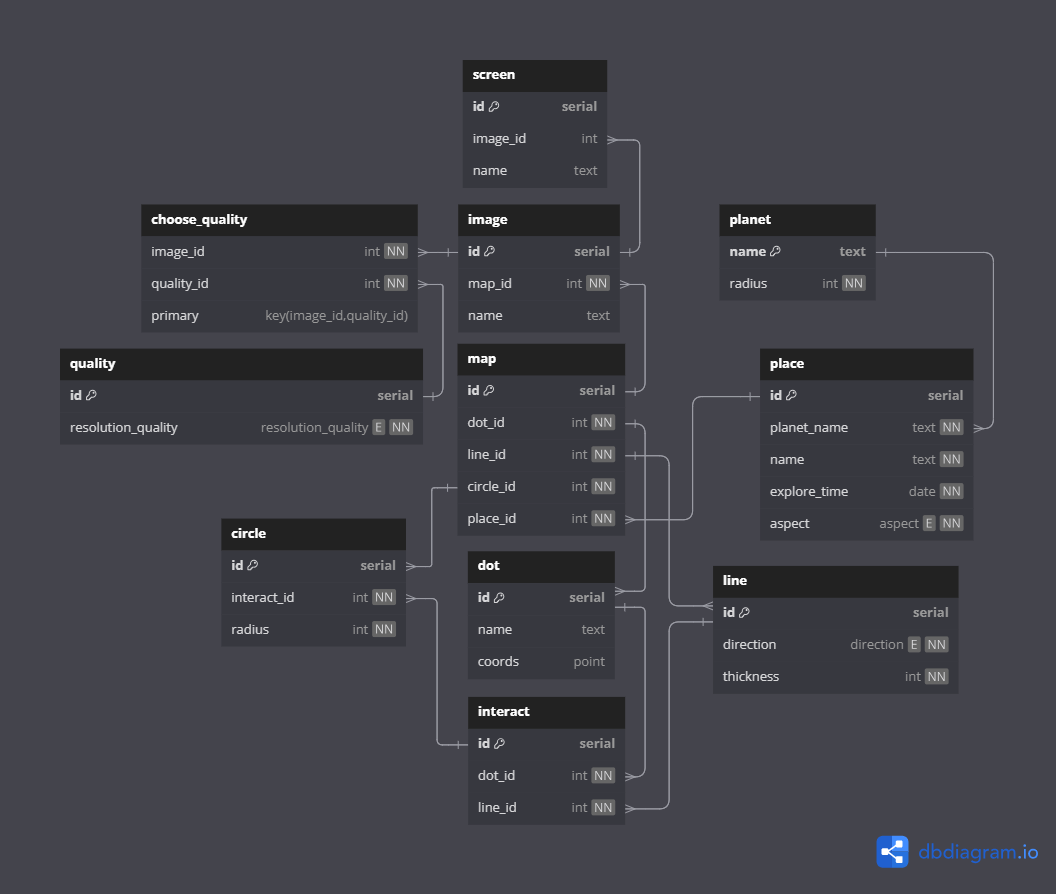
1. **Текст задания**

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.

1. **Исходная, нормализованная и денормализованная модели.**
   1. **Даталогическая модель (исходная) и Функциональные зависимости (изначальные)**

****

|  |
| --- |
| screen: id → name; id → image\_id  image: id → name; id → map\_id  choose\_quality: (image\_id, quality\_id) → ( )  quality: id → resolution\_quality  map: id → dot\_id; id → line\_id; id → circle\_id; id → place\_id  dot: id → name; id → coords  line: id → direction; id → thickness  circle: id → radius; id → interact\_id  interact: id → dot\_id; id → line\_id  place: id → name; id → explore\_time; id → aspect; id → planet\_id  planet: id → name; id → radius |

* 1. **Преобразование к НФ**

**Преобразование к 1НФ**

Не потребовалось, условие “на пересечении каждой строки и столбца – 1 значение” и так выполнялось.

**Преобразование к 2НФ**

Не потребовалось, поскольку у всех первичных ключей нет подмножеств, а значит атрибуты всех отношений – в полной функциональной зависимости от соответствующих первичных ключей.

**Преобразование к 3НФ**

В некоторых отношения:

* interact: (id → dot\_id), (id → line\_id)
* circle: id → interact\_id
* circle: (id → dot\_id), (id → line\_id)
* map: (id → dot\_id), (id → line\_id), (id → circle\_id)

Посмотрите в отношении map, где наблюдалась транзитивная зависимость. По сути, транзитивность здесь даёт то, что в этих отношениях как бы два первичных ключа, только один указан явно (id), а другой получается (circle\_id). Поэтому:

* в map убираем dot\_id, line\_id.

**Преобразование к НФБК**

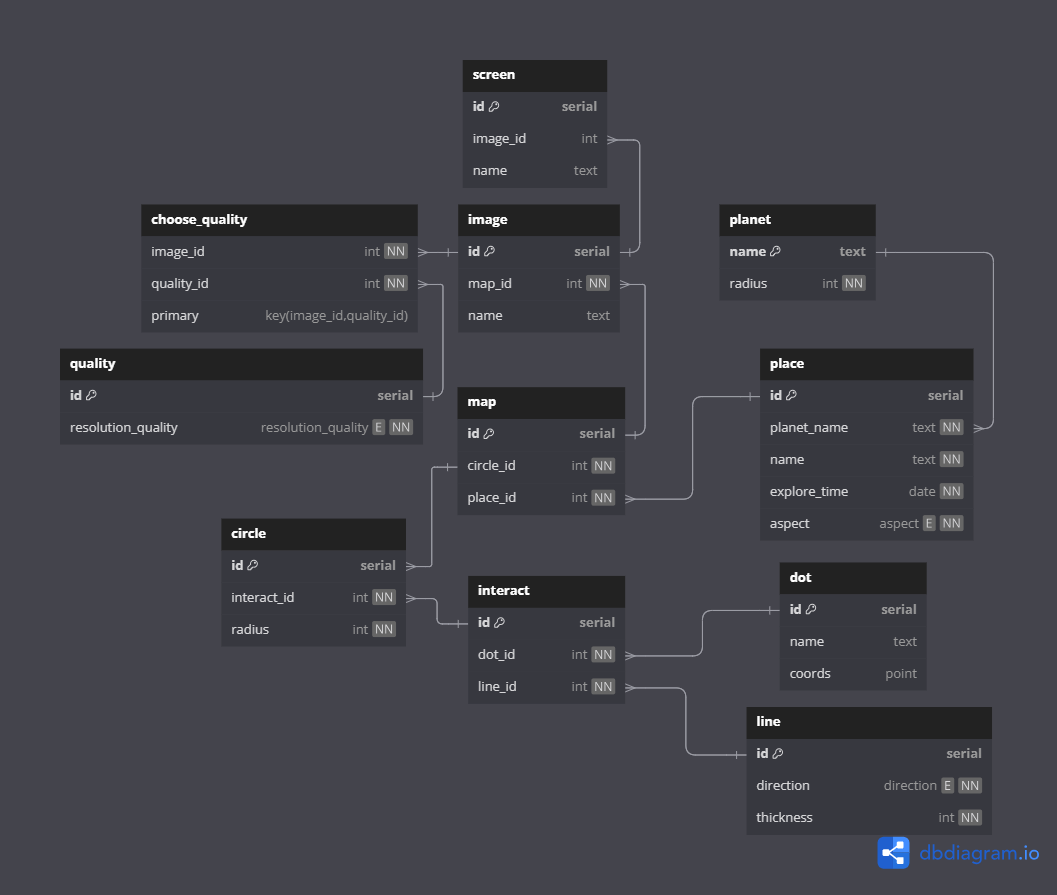
planet: id → name, id → radius

Названия планет (*name*) уникальны (значит из *name* можно узнать *id* и наоборот, другими словами, *id → name, name → id*). Итак, задаем *name* в качестве первичного ключа.

planet: name → radius

place: id → name, id → explore\_time, id → aspect, id → planet\_name

* 1. **Даталогическая модель и Функциональные зависимости (после преобразований)**



screen: id → name; id → image\_id

image: id → name; id → map\_id

choose\_quality: (image\_id, quality\_id) → ( )

quality: id → resolution\_quality

map: id → circle\_id; id → place\_id

dot: id → name; id → coords

line: id → direction; id → thickness

circle: id → radius; id → interact\_id

interact: id → dot\_id; id → line\_id

place: id → name; id → explore\_time; id → aspect; id → planet\_name

planet: name → radius

1. **Полезная денормализация**

|  |
| --- |
| **Объединение связанных таблиц:** В некоторых случаях, объединение таблиц может уменьшить количество операций **JOIN** и ускорить обработку запросов. Например, можно рассмотреть объединение таблиц circle и interact, если часто одновременно запрашиваются данные о том, из каких точки и линии круг создан.  **Добавление избыточных атрибутов:** В некоторых случаях добавление избыточных атрибутов может улучшить производительность запросов. Например, если часто запрашивается количество линий длиной 10 см на графике, можно добавить атрибут *line\_with\_thickness\_10cm\_count* в таблицу map. Это позволит избежать операций подсчета при каждом запросе, однако необходимо будет обновлять этот атрибут при добавлении или удалении линий . |

1. **Функция и триггер на языке PL/pgSQL**

Сделать логирование (circle\_log) для таблицы circle, чтобы отслеживать появление, изменение или исчезновение кругов на графике.

CREATE TABLE circle\_log (

    id SERIAL PRIMARY KEY,

    circle\_id INT NOT NULL REFERENCES circle(id),

    log\_time TIMESTAMP NOT NULL,

    log\_type TEXT NOT NULL,

    old\_radius INT,

    new\_radius INT

);

-- Trigger function for insert

CREATE OR REPLACE FUNCTION insert\_circle()

RETURNS TRIGGER AS $$

BEGIN

    INSERT INTO circle\_log (circle\_id, log\_time, log\_type, new\_radius)

    VALUES (NEW.id, now(), 'APPEAR', NEW.radius);

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER insert\_circle\_trigger

AFTER INSERT ON circle

FOR EACH ROW

EXECUTE FUNCTION insert\_circle();

-- Trigger function for update

CREATE OR REPLACE FUNCTION update\_circle()

RETURNS TRIGGER AS $$

BEGIN

    INSERT INTO circle\_log (circle\_id, log\_time, log\_type, old\_radius, new\_radius)

    VALUES (NEW.id, now(), 'CHANGED', OLD.radius, NEW.radius);

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER update\_circle\_trigger

AFTER UPDATE ON circle

FOR EACH ROW

EXECUTE FUNCTION update\_circle();

-- Trigger function for delete

CREATE OR REPLACE FUNCTION delete\_circle()

RETURNS TRIGGER AS $$

BEGIN

    INSERT INTO circle\_log (circle\_id, log\_time, log\_type, old\_radius)

    VALUES (OLD.id, now(), 'DISAPPEAR', OLD.radius);

    RETURN OLD;

END;

$$ LANGUAGE plpgsql;

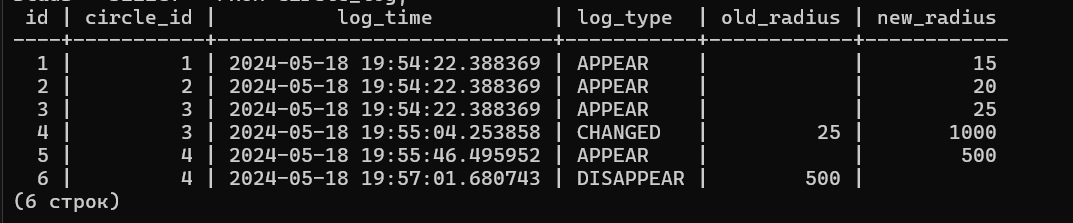
CREATE TRIGGER delete\_circle\_trigger

AFTER DELETE ON circle

FOR EACH ROW

EXECUTE FUNCTION delete\_circle();

**Модуль логирования**



1. **Вывод по работе**

При выполнении лабораторной работы я познакомился с понятием нормализации и денормализации. Научился определять функциональные зависимости модели, а также анализировать последнюю на соответствие различным нормальным формам. Познакомился с процедурным языком PL/pgSQL. Изучил эффективные способы денормализации схемы базы данных и ситуации, в которых возможно их применение.