Отчет

**по лабораторной работе №3**

по дисциплине «Базы данных»

Выполнил: Караганов П.Э., группа P3110

Преподаватель: Гаврилов А. В.

Николаев В. В.

Оглавление

[Текст задания 3](#_Toc195133018)

[Исходная модель и Нормализированная модель 3](#_Toc195133019)

[Функциональные зависимости 3](#_Toc195133020)

[Нормальные формы 3](#_Toc195133021)

[Денормализованная модели 3](#_Toc195133022)

[Триггер 3](#_Toc195133023)

[Вывод 4](#_Toc195133024)

# Текст задания

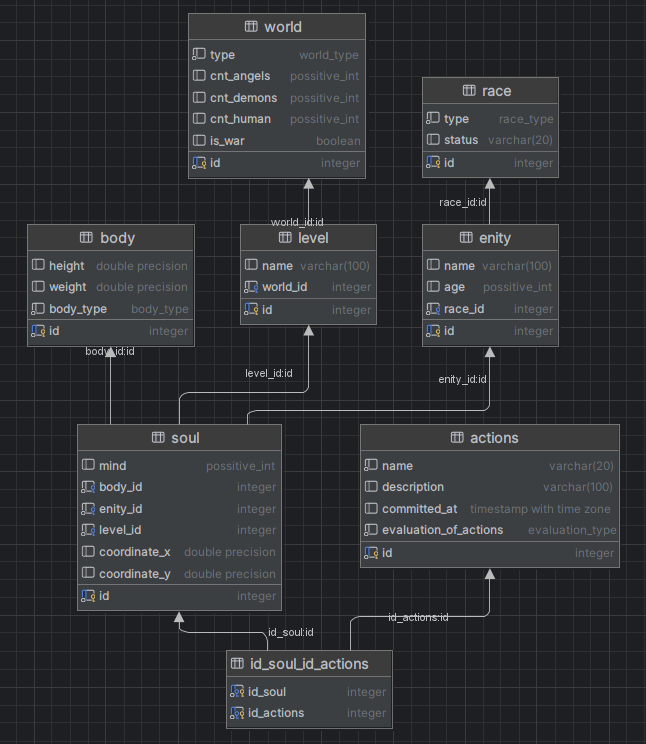
Задание.

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 основеNF (как минимум).
* Опишите изменения в функциональных зависимостях, произошедшие после преобразования в 3NF (как минимум). Постройте схему на основеNF;
* 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.

# Исходная модель и Нормализированная модель



# Функциональные зависимости

* **Body**: (id) → {height, weight, body\_type}
* **Soul**: (id) → {mind, body\_id, enity\_id, level\_id, coordinate\_x, coordinate\_y}
* **Level**: (id) → {name, world\_id}
* **Race**: (id) → {status, type}
* **Enity:** (id) → {name, age, race\_id}
* **World:** (id) → {type, cnt\_angels, cnt\_demons, cnt\_human, is\_war}
* **Actions**: (id) → {name, description, committed\_at, evaluation\_of\_actions}
* **Id\_soul\_id\_actions**: (id\_soul, id\_actions) → (id\_soul, id\_actions) (X) → (X)

# Нормальные формы

* *Первая нормальная форма (1NF):*

Требования:

1. Все атрибуты атомарны (неделимы).
2. Нет повторяющихся групп или массивов.
3. Есть первичный ключ.

Соответствие в моей БД:

1. Все поля содержат скалярные значения (числа, строки, ENUM, даты — нет составных или множественных значений).
2. В каждой таблице есть явный id как первичный ключ.
3. Нет столбцов с несколькими значениями.

* *Вторая нормальная форма (2NF):*

Требования:

1. Соответствует 1NF.
2. Все неключевые атрибуты зависят от всего первичного ключа (нет частичных зависимостей).

Соответствие в моей БД:

1. Во всех таблицах первичный ключ состоит из одного столбца (id), поэтому частичные зависимости невозможны.
2. В таблице id\_soul\_id\_actions (связь many-to-many) первичный ключ — составной (id\_soul, id\_actions), но у неё нет неключевых атрибутов, что автоматически удовлетворяет 2NF.

* *Третья нормальная форма (3NF):*

Требования:

1. Соответствует 2NF.
2. Нет транзитивных зависимостей (неключевые атрибуты зависят только от первичного ключа, а не от других неключевых атрибутов).

Соответствие в моей БД:

1. Нет атрибутов, которые зависят от других неключевых атрибутов.

* *Нормальная форма Бойса-Кодда (BCNF):*

Требования:

1. Для всех нетривиальных завчисимостей X → Y, X должен быть суперключом.

Соответствие в моей БД:

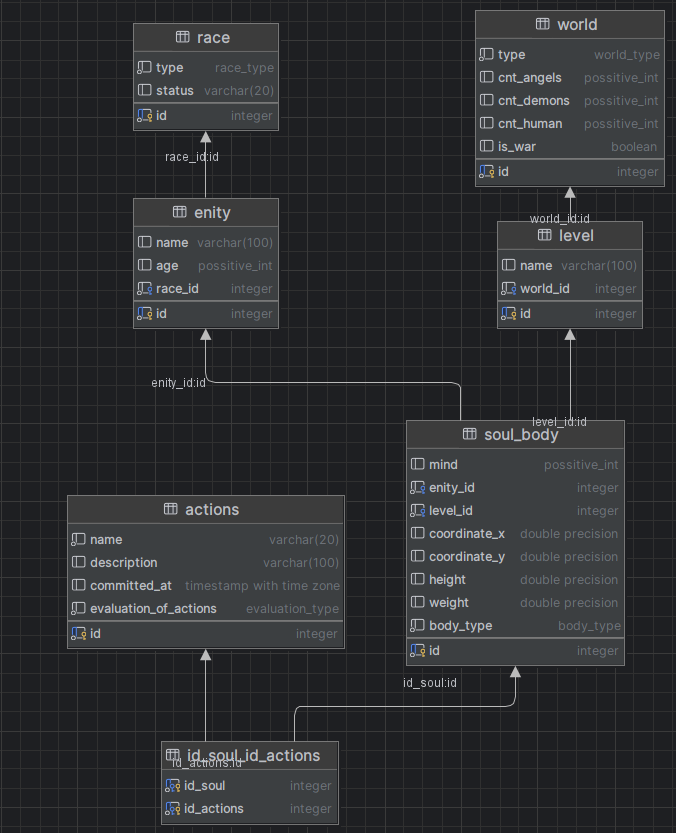
1. Во всех таблицах первичный ключ — это id, и нет других нетривиальных зависимостей, где X не является суперключом.
2. В таблице id\_soul\_id\_actions нет неключевых атрибутов, что автоматически удовлетворяет BCNF.

Пример нарушения:

Если бы в race было type → status (т.е. type уникален), но type не являлся бы первичным ключом или суперключом. Но в моей БД type не объявлен как UNIQUE, поэтому зависимостей type → status нет.

# Денормализованная модели

*Объединение связанных таблиц*: например, можно объединить отношения Soul и Body. Это позволяет устранить JOIN между Soul и Body для **частых запросов** типа «*Получить все данные о душе и её теле*». Также такое объединение упрощает вставку/обновление данных (1 запрос вместо 2). Но нарушается целостность данных – теперь тела могут существовать без душ, а это противоречит нашим правилам мира, который мы выдумали, создавая эту БД.



# Триггер

*Первый триггер:* проверка на то, что у одной души не может быть больше одного тела. Благодаря этой проверке обеспечивается целостность данных.

-- Проверка, что у одной души не может быть много тел  
CREATE OR REPLACE FUNCTION *check\_single\_body\_per\_soul*()  
 RETURNS TRIGGER AS  
$$  
BEGIN  
 IF *EXISTS* (SELECT 1  
 FROM soul  
 WHERE body\_id = NEW.body\_id  
 AND id != NEW.id)  
 THEN  
 RAISE EXCEPTION 'Тело % уже принадлежит другой душе', NEW.body\_id;  
 END IF;  
 RETURN NEW;  
END;  
$$ LANGUAGE plpgsql;  
  
CREATE TRIGGER single\_body\_per\_soul\_trigger  
 BEFORE INSERT OR UPDATE  
 ON SOUL  
 FOR EACH ROW  
EXECUTE FUNCTION *check\_single\_body\_per\_soul*();

Второй триггер: автоматически обновляет кол-во сущностей на уровне. Благодаря этой проверке мы можем избавиться от Count запросов при запросе данных из БД

--Автоматически обновляем кол-во сущностей на каждом уровне

ALTER TABLE level ADD COLUMN entity\_count INTEGER DEFAULT 0;  
  
CREATE OR REPLACE FUNCTION *update\_level\_entity\_count*()  
 RETURNS TRIGGER AS $$  
BEGIN  
 -- При добавлении новой души  
 IF TG\_OP = 'INSERT' THEN  
 UPDATE level  
 SET entity\_count = entity\_count + 1  
 WHERE id = NEW.level\_id;  
 -- При удалении души  
 ELSIF TG\_OP = 'DELETE' THEN  
 UPDATE level  
 SET entity\_count = entity\_count - 1  
 WHERE id = OLD.level\_id;  
 -- При изменении уровня души  
 ELSIF TG\_OP = 'UPDATE' AND NEW.level\_id != OLD.level\_id THEN  
 UPDATE level  
 SET entity\_count = entity\_count - 1  
 WHERE id = OLD.level\_id;  
  
 UPDATE level  
 SET entity\_count = entity\_count + 1  
 WHERE id = NEW.level\_id;  
 END IF;  
 RETURN NULL;  
END;  
$$ LANGUAGE plpgsql;  
  
-- Триггер для таблицы soul  
CREATE TRIGGER level\_entity\_count\_trigger  
 AFTER INSERT OR UPDATE OR DELETE  
 ON SOUL  
 FOR EACH ROW  
 EXECUTE FUNCTION *update\_level\_entity\_count*();

# Вывод

В ходе выполнения лабораторной работы были успешно проведены процессы нормализации и денормализации базы данных, начиная с анализа функциональных зависимостей и приведения отношений к третьей нормальной форме (3NF), а затем к нормальной форме Бойса-Кодда (BCNF). Были выявлены и устранены аномалии, связанные с избыточностью данных и нарушением целостности, что позволило оптимизировать структуру базы данных для эффективного хранения и обработки информации.