Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа №3

По дисциплине

“Базы данных”

Вариант 10102

Выполнил:

Ахроров Кароматуллохон Фирдавсович

Группа P3110

Проверил:

Гаврилов Антон Валерьевич

Преподаватель практики

Санкт-Петербург 2025

**1. Текст задания**

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

1. Описать функциональные зависимости для отношений полученной схемы (минимальное множество).
2. Привести отношения в 3NF. Построить схему на основе 3NF.
3. Описать изменения в функциональных зависимостях после преобразования в 3NF.
4. Преобразовать отношения в BCNF. Доказать, что полученные отношения находятся в BCNF.
5. Определить полезные денормализации для схемы и описать их.
6. Придумать триггер и связанную функцию в предметной области, реализовать их на PL/pgSQL.

Отчет должен содержать текст задания, исходную, нормализованные и денормализованные модели, ответы на вопросы, функцию и триггер, выводы.

**2. Исходная модель (схема из ЛР №1)**

**Основные отношения:**

* tasks(task\_id PK, name, description, assigned\_at, completed\_at)
* trees(tree\_id PK, species, planted\_at, location)
* metal\_fastenings(fastening\_id PK, location, status)
* markers(marker\_id PK, name, placed\_at, task\_id FK)
* supports(support\_id PK, type, installed\_at, marker\_id FK)
* greenkeepers(greenkeeper\_id PK, full\_name, hired\_at, phone)
* workers(worker\_id PK, full\_name, role, hired\_at)
* fences(fence\_id PK, location, erected\_at)
* rivers(river\_id PK, name, length\_km)
* dilofosavrs(dilofosavr\_id PK, name, birth\_date, river\_id FK)
* spit\_events(event\_id PK, dilofosavr\_id FK, occurred\_at, description)
* observers(observer\_id PK, full\_name, assigned\_area)
* spit\_event\_logs(log\_id, event\_id, dilofosavr\_id ,old\_count, new\_count, logged\_at)
* spit\_alerts(alert\_id, dilofosavr\_id, alert\_type,alert\_message, alert\_at)

**Ассоциативные отношения:**

* support\_builders(support\_id FK, worker\_id FK) PK(support\_id,worker\_id)
* observer\_tasks(observer\_id FK, task\_id FK) PK(observer\_id,task\_id)

**3. Функциональные зависимости (минимальное множество)**

Ниже приведено минимальное множество функциональных зависимостей для каждой таблицы схемы:

1. **tasks**(task\_id):  
   task\_id → description, duration
2. **markers**(marker\_id):  
   marker\_id → status, task\_id
3. **trees**(tree\_id):  
   tree\_id → status, task\_id
4. **metal\_fastenings**(fastening\_id):  
   fastening\_id → type, status, tree\_id
5. **supports**(support\_id):  
   support\_id → scheduled\_time, marker\_id
6. **greenkeepers**(greenkeeper\_id):  
   greenkeeper\_id → name
7. **support\_builders**(support\_id, greenkeeper\_id):  
   — нет дополнительных атрибутов для зависимостей
8. **observers**(observer\_id):  
   observer\_id → name
9. **observer\_tasks**(observer\_id, task\_id):  
   — нет дополнительных атрибутов для зависимостей
10. **fences**(fence\_id):  
    fence\_id → description
11. **workers**(worker\_id):  
    worker\_id → name, fence\_id, side
12. **rivers**(river\_id):  
    river\_id → name
13. **dilofosavrs**(dilofosavr\_id):  
    dilofosavr\_id → name,species, location, river\_id, spits\_saliva,spit\_count
14. **spit\_events**(event\_id):  
    event\_id → dilofosavr\_id, worker\_id, event\_time, description
15. spit\_event\_logs(log\_id): event\_id, dilofosavr\_id ,old\_count, new\_count, logged\_at
16. spit\_alerts(alert\_id): dilofosavr\_id, alert\_type,alert\_message, alert\_at

**4. Приведение к 3NF**

Все отношения из ЛР №1 уже находятся в 3NF, так как:

1. Каждый нетранзитивный FD основывается на первичном ключе.
2. Нет зависимостей типа A → B, где A — не ключ.
3. Нет частичных зависимостей в отношениях с простым ключом.

**Итоговая схема 3NF совпадает с исходной.**

**5. Изменения в функциональных зависимостях после 3NF**

Поскольку все исходные отношения были в 3NF, преобразование не изменило множества FD.

**6. Преобразование в BCNF и доказательство**

Для каждого отношения проверяем, что каждая FD X → Y имеет X суперклассом ключа:

* Во всех таблицах единственный ключ — это \*\_id (или составной PK), и все FDs исходили от этого ключa → BCNF.
* В observer\_tasks: FD (observer\_id, task\_id) → assigned\_at, где (observer\_id,task\_id) — PK → BCNF.

**Итог:** Все отношения уже находятся в BCNF.

**7. Предложения по денормализации**

1. **Добавить счетчик плевков** в таблице dilofosavrs для быстрого доступа к числу плевков:
2. ALTER TABLE dilofosavrs ADD COLUMN spit\_count INT NOT NULL DEFAULT 0;
3. **Объединить tasks и markers** в одну таблицу при активном мониторинге, если для каждой задачи всегда есть маркер.
4. **Материализованные представления** для отчетов: например, представление vw\_dino\_events объединяет dilofosavrs и spit\_events с агрегатами.

**8. Триггер и функция на PL/pgSQL**

Автоматически вести учёт количества «плевков» каждого дилофозавра, фиксировать историю изменений и выдавать тревогу, если число плевков превысило заданный порог (10).

CREATE TABLE spit\_event\_logs (

log\_id SERIAL PRIMARY KEY,

event\_id INT REFERENCES spit\_events(event\_id),

dilofosaur\_id INT REFERENCES dilofosavrs(dilofosaur\_id),

old\_count INT NOT NULL,

new\_count INT NOT NULL,

logged\_at TIMESTAMP NOT NULL DEFAULT now()

);

CREATE TABLE spit\_alerts (

alert\_id SERIAL PRIMARY KEY,

dilofosaur\_id INT REFERENCES dilofosavrs(dilofosaur\_id),

alert\_type TEXT NOT NULL,

alert\_message TEXT NOT NULL,

alert\_at TIMESTAMP NOT NULL DEFAULT now()

);

CREATE OR REPLACE FUNCTION fn\_handle\_spit\_event()

RETURNS TRIGGER AS $$

DECLARE

old\_count INT;

new\_count INT;

BEGIN

SELECT spit\_count

INTO old\_count

FROM dilofosavrs

WHERE dilofosaur\_id = NEW.dilofosaur\_id;

new\_count := old\_count + 1;

UPDATE dilofosavrs

SET spit\_count = new\_count

WHERE dilofosaur\_id = NEW. dilofosaur\_id;

INSERT INTO spit\_event\_logs(

event\_id, dilofosaur\_id, old\_count, new\_count

) VALUES (

NEW.event\_id, NEW.dilofosaur\_id, old\_count, new\_count

);

IF new\_count > 10 THEN

INSERT INTO spit\_alerts(

dilofosaur\_id, alert\_type, alert\_message

) VALUES (

NEW.dilofosaur\_id,

'HighSpitCount',

'Spit count exceeded threshold (10). Current count: ' || new\_count

);

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg\_spit\_event\_logging

AFTER INSERT ON spit\_events

FOR EACH ROW

EXECUTE PROCEDURE fn\_handle\_spit\_event();

**9. Выводы**

1. Исходная схема была спроектирована корректно и уже соответствовала требованиям 3NF и BCNF.
2. Денормализация через добавление spit\_count повышает производительность аналитики по плевкам.
3. Триггер и функция упрощают поддержание целостности данных при подсчете событий.
4. В отчетах полезно использовать материализованные представления для ускорения сложных запросов.