**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

Практическая работа №6

по курсу

«Сетевое программное обеспечение»

Выполнили:

студенты группы ИКПИ-14

Хохлов Т.В.

Принял:

Тарабанов И.Ф.

Санкт-Петербург

2025 г.

# Задача

На основе любой реляционной клиент-серверной СУБД требуется показать пример «неправильно» спроектированной таблицы и 1-2 SQL-операции обращения к ней, которые при массовом выполнении приводили бы к существенному увеличению блокировок и очередей. Затем предложить способ устранения этой проблемы (добавлением другой таблицы, перестройкой структуры полей, добавлением вспомогательных элементов - индексов, джобов и др.) В отчёте приложить тексты DDL и SQL конструкций, а также краткое словесное описание иллюстрируемой проблемы. В качестве отладочного стенда можно использовать доступные online-средства («песочницы») для работы с БД, например sqlfiddle.com Решение

# Решение

В данном примере рассматривается проблема проектирования реляционной базы данных, приводящая к возникновению блокировок при конкурентном доступе. На примере MySQL демонстрируется:

Проблемная архитектура — хранение агрегированных данных (счётчиков) в основной таблице.

Оптимальное решение — нормализованная структура с вынесением агрегатов в отдельную таблицу.

**Проблемный пример:**

CREATE TABLE bad\_orders (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT,

item VARCHAR(100),

-- Проблемное поле - счётчик заказов пользователя

order\_count INT

);

INSERT INTO bad\_orders (user\_id, item, order\_count) VALUES

(1, 'Телефон', 1),

(1, 'Чехол', 2),

(2, 'Ноутбук', 1);

DELIMITER //

CREATE PROCEDURE add\_bad\_order(IN user INT, IN product VARCHAR(100))

BEGIN

DECLARE cnt INT;

START TRANSACTION;

SELECT COUNT(\*) INTO cnt FROM bad\_orders WHERE user\_id = user;

INSERT INTO bad\_orders (user\_id, item, order\_count)

VALUES (user, product, cnt + 1);

COMMIT;

END //

DELIMITER ;

CALL add\_bad\_order(1, 'Наушники');

CALL add\_bad\_order(1, 'Кабель');

SELECT \* FROM bad\_orders WHERE user\_id = 1;

# Основные проблемы

* **Блокировки**:

Запрос SELECT COUNT(\*) требует сканирования таблицы, что приводит к очередям при параллельных запросах для одного user\_id.

* **Низкая производительность**:

Сложность O(n) для подсчёта заказов вместо O(1).

**Решение проблемы:**

CREATE TABLE good\_orders (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT,

item VARCHAR(100)

);

-- Отдельная таблица-счётчик

CREATE TABLE user\_stats (

user\_id INT PRIMARY KEY,

order\_count INT DEFAULT 0

);

INSERT INTO good\_orders (user\_id, item) VALUES

(1, 'Телефон'),

(1, 'Чехол'),

(2, 'Ноутбук');

INSERT INTO user\_stats VALUES

(1, 2),

(2, 1);

DELIMITER //

CREATE PROCEDURE add\_good\_order(IN user INT, IN product VARCHAR(100))

BEGIN

START TRANSACTION;

INSERT INTO good\_orders (user\_id, item) VALUES (user, product);

INSERT INTO user\_stats VALUES (user, 1)

ON DUPLICATE KEY UPDATE order\_count = order\_count + 1;

COMMIT;

END //

DELIMITER ;

CALL add\_good\_order(1, 'Наушники');

CALL add\_good\_order(1, 'Кабель');

SELECT \* FROM good\_orders WHERE user\_id = 1;

SELECT \* FROM user\_stats WHERE user\_id = 1;

# Преимущества решения

|  |  |  |
| --- | --- | --- |
| Критерий | Проблемный вариант | Оптимальное решение |
| Блокировки | Табличные | Только по user\_id |
| Сложность | O(n) | O(1) |
| Масштабируемость | Низкая | Высокая |
| Согласованность | Риск аномалий | ACID-гарантии |