# Базы данных

**Практическая работа № 12 «Табличные языки запросов.»**

## **Цель:**

## Освоение различных аспектов разработки функций хранимых процедур с использованием языка SQL.

## **Задание.**

## Использование команды SELECT INTO в хранимых процедурах (в отчете приведите пример хранимой процедуры, обрабатывающей данные таблицы БД вашего варианта и содержащей команду SELECT INTO).

## Использование атрибута %TYPE в хранимых процедурах.

## Использование атрибута %ROWTYPE в хранимых.

## Использование условных команд в хранимых.

## Использование циклов в хранимых.

## **Ход выполнения.**

В SQLite нет встроенной поддержки хранимых процедур (stored procedures), как в других СУБД вроде PostgreSQL, MySQL, Oracle или MS SQL Server. Однако, можно реализовать аналог функциональности хранимых процедур в SQLite через триггеры и использование встроенных функций на других языках (например, Python).

**Создание базы данных и таблицы**

CREATE TABLE Прокат\_ПР12 (

id INTEGER PRIMARY KEY AUTOINCREMENT,

клиент TEXT NOT NULL,

транспорт TEXT NOT NULL,

дата\_начала TEXT NOT NULL,

дата\_окончания TEXT NOT NULL,

стоимость REAL NOT NULL

);

**Наполняем таблицу данными**

INSERT INTO Прокат\_ПР12 (клиент, транспорт, дата\_начала, дата\_окончания, стоимость)

VALUES

('Иванов Иван', 'Автомобиль', '2024-11-01', '2024-11-10', 15000.0),

('Петров Петр', 'Мотоцикл', '2024-11-05', '2024-11-15', 8000.0),

('Сидорова Анна', 'Скутер', '2024-11-07', '2024-11-09', 2000.0),

('Козлов Михаил', 'Велосипед', '2024-11-01', '2024-11-07', 1500.0),

('Смирнов Олег', 'Автомобиль', '2024-11-10', '2024-11-20', 18000.0);

**Использование SELECT INTO**

В SQLite команда SELECT INTO заменяется на CREATE TABLE AS.

**Создаем новую таблицу с отфильтрованными данными**

CREATE TABLE Прокат\_Автомобили AS

SELECT \*

FROM Прокат\_ПР12

WHERE транспорт = 'Автомобиль';

**Проверка содержимого новой таблицы**

SELECT \* FROM Прокат\_Автомобили;

**Эмуляция %TYPE и %ROWTYPE**

SQLite не поддерживает атрибуты %TYPE и %ROWTYPE, так как у него отсутствует строгая типизация. Но можно использовать переменные и обращаться к значениям полей через SQL-запросы.

Пример: Получение типа данных столбца в контексте триггера.

**Триггер для копирования данных в другую таблицу при добавлении**

CREATE TRIGGER trg\_after\_insert

AFTER INSERT ON Прокат\_ПР12

BEGIN

INSERT INTO Прокат\_Автомобили (id, клиент, транспорт, дата\_начала, дата\_окончания, стоимость)

SELECT NEW.id, NEW.клиент, NEW.транспорт, NEW.дата\_начала, NEW.дата\_окончания, NEW.стоимость

WHERE NEW.транспорт = 'Автомобиль';

END;

**Условные команды**

В SQLite условные операции могут быть использованы в триггерах или в выражениях. Например:

**Триггер для проверки стоимости**

CREATE TRIGGER trg\_check\_cost

BEFORE INSERT ON Прокат\_ПР12

BEGIN

SELECT CASE

WHEN NEW.стоимость < 0 THEN

RAISE(ABORT, 'Стоимость не может быть отрицательной')

ELSE

NULL

END;

END;

**Использование циклов**

SQLite не поддерживает циклы напрямую в SQL-запросах. Однако можно использовать рекурсивные CTE для их имитации.

**Пример рекурсивного CTE для генерации дат между диапазоном**

WITH RECURSIVE date\_range AS (

SELECT date('2024-11-01') AS текущая\_дата

UNION ALL

SELECT date(текущая\_дата, '+1 day')

FROM date\_range

WHERE текущая\_дата < date('2024-11-10')

)

SELECT \* FROM date\_range;

**Код**: