

Документ подписан простой электронной подписью  
Информация о владельце:  
ФИО: Куижева Саида Казбековна  
Должность: Ректор  
Дата подписания: 24.10.2024 00:15:13  
Уникальный программный ключ:  
71183e1134ef9cfa69b206d480271b3c1a975e6f

**Филиал ФГБОУ ВО «Майкопский государственный технологический университет» в поселке Яблоновском**  
**Политехнический колледж**

## **МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ**

**Профессиональный модуль:** ПМ.01 Разработка, администрирование и защита баз данных

**МДК.01.01:** Проектирование и разработка баз данных

**МДК.01.02:** Управление базами данных

**Специальность:** 09.02.11 Разработка и управление программным обеспечением

## **1. Общие положения**

Практические работы направлены на формирование профессиональных компетенций (ПК 1.1 – ПК 1.5) и общих компетенций (ОК 01 – ОК 09).

**Цель рекомендаций:** Помочь студентам систематизировать теоретические знания и освоить практические навыки проектирования, программирования, администрирования и защиты баз данных.

**Требования к отчету:**

1. **Титульный лист** (согласно шаблону колледжа).
2. **Цель работы.**
3. **Задание** (условие задачи/вариант).
4. **Ход выполнения** (SQL-код, скриншоты результатов выполнения запросов, описание действий).
5. **Вывод** (какие навыки получены, какие инструменты использованы).

## 2. МДК.01.01 Проектирование и разработка баз данных

### Тема 1.1. Язык структурированных запросов (SQL)

Формируемые компетенции: ПК 1.1, ПК 1.2, ПК 1.3

#### Практическая работа № 1-3. Создание и использование индексов.

##### Анализ плана запросов (EXPLAIN).

- **Задание:** Создайте таблицу Orders (Заказы) с полями id, customer\_name, order\_date, amount. Заполните таблицу 10 000 случайных записей (можно через генератор или скрипт). Напишите запрос на поиск заказов конкретного клиента. Проанализируйте план выполнения запроса. Создайте индекс на поле customer\_name и сравните время выполнения.
- **Технология решения:**
  1. Создать таблицу: `CREATE TABLE Orders (id SERIAL PRIMARY KEY, customer_name VARCHAR(100), order_date DATE, amount DECIMAL);`
  2. Запустить скрипт генерации данных (или использовать `INSERT ... SELECT generate_series(...)`).
  3. Выполнить запрос с анализом: `EXPLAIN ANALYZE SELECT * FROM Orders WHERE customer_name = 'Иванов И.И.';`
  4. *Технология:* В выводе обратить внимание на Seq Scan (последовательное сканирование) и высокое время выполнения (Execution Time).
  5. Создать индекс: `CREATE INDEX idx_orders_customer ON Orders(customer_name);`
  6. Повторить `EXPLAIN ANALYZE`. Убедиться, что СУБД теперь использует Index Scan и время выполнения сократилось.

#### Практическая работа № 13-20. Создание хранимых процедур и функций.

- **Задание:** Создайте хранимую процедуру, которая принимает ID клиента и сумму, и обновляет баланс клиента. Если баланса не хватает, процедура должна выдать ошибку.
- **Технология решения (на примере PostgreSQL):**
  1. Написать код процедуры (используя PL/pgSQL):

```
sql
CREATE OR REPLACE PROCEDURE UpdateBalance(p_client_id
INT, p_amount DECIMAL)
LANGUAGE plpgsql AS $$
DECLARE
    current_balance DECIMAL;
BEGIN
    SELECT balance INTO current_balance FROM Clients WHERE id
= p_client_id;
    IF current_balance >= p_amount THEN
        UPDATE Clients SET balance = balance - p_amount WHERE id
= p_client_id;
        RAISE NOTICE 'Баланс успешно обновлен';
```

- ```

ELSE
    RAISE EXCEPTION 'Недостаточно средств';
END IF;
END;
$$;

```
2. Вызвать процедуру: `CALL UpdateBalance(1, 500);`
  3. Проверить обработку ошибки: `CALL UpdateBalance(1, 999999);`  
(Должно вывестись исключение).

### Практическая работа № 21-25. Создание триггеров.

- **Задание:** Создайте таблицу `Audit_Log` (id, operation, old\_data, new\_data, changed\_at). Создайте триггер, который при обновлении цены в таблице `Products` записывает старое и новое значение в лог.
- **Технология решения:**
  1. Создать функцию триггера:

```

sql
CREATE OR REPLACE FUNCTION log_price_change()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO Audit_Log(operation, old_data, new_data,
changed_at)
    VALUES ('UPDATE_PRICE', OLD.price::text, NEW.price::text,
NOW());
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

```
  2. Привязать триггер к таблице:

```

sql
CREATE TRIGGER trg_price_update
AFTER UPDATE OF price ON Products
FOR EACH ROW EXECUTE FUNCTION log_price_change();

```
  3. Выполнить `UPDATE Products SET price = price * 1.1 WHERE id = 1;` и проверить таблицу `Audit_Log`.

### Практическая работа № 26-29. Транзакции и уровни изоляции.

- **Задание:** Продемонстрируйте проблему «грязного чтения» (Dirty Read). Откройте две сессии, выполните изменения в одной, не завершая транзакцию, и прочитайте данные во второй. Затем исправьте ситуацию, установив уровень изоляции `READ COMMITTED`.
- **Технология решения:**
  1. **Сессия 1:** `BEGIN; UPDATE Accounts SET balance = balance - 100 WHERE id = 1; (Не делаем COMMIT).`
  2. **Сессия 2:** Попытаться прочитать данные. При стандартном уровне изоляции данные не изменятся (заблокированы).
  3. **Технология демонстрации:** Для реального «грязного чтения» (в старых СУБД) или для демонстрации «неповторяемого чтения» используйте `SET TRANSACTION ISOLATION LEVEL READ`

COMMITTED; (или REPEATABLE READ). Показать, как транзакции изолируют друг друга.

---

## Тема 1.2. NoSQL базы данных

Формируемые компетенции: ПК 1.1 – ПК 1.3

### Практическая работа № 30-34. Работа с документо-ориентированной СУБД (MongoDB).

- **Задание:** Создайте коллекцию `users`. Добавьте документ с полями `name`, `email`, `roles` (массив), `created_at`. Найдите всех пользователей с ролью "admin". Обновите `email` пользователя.
- **Технология решения:**
  1. Подключиться к MongoDB (через `mongosh` или GUI-клиент).
  2. Выбрать базу: `use my_app_db`;
  3. Вставить документ:

```
javascript
db.users.insertOne({
  name: "Алексей",
  email: "alex@example.com",
  roles: ["user", "admin"],
  created_at: new Date()
});
```
  4. Запрос на поиск: `db.users.find({ roles: "admin" });`
  5. Обновление: `db.users.updateOne({ name: "Алексей" }, { $set: { email: "new_alex@example.com" } });`
  6. Создать индекс для оптимизации: `db.users.createIndex({ email: 1 });`

---

## 3. МДК.01.02 Управление базами данных

### Тема 2.1. Установка и настройка сервера СУБД

Формируемые компетенции: ПК 1.4

### Практическая работа № 2-5. Настройка СУБД и скрипты инициализации.

- **Задание:** Напишите SQL-скрипт, который создает базу данных `CompanyDB`, создает в ней таблицы `Departments` и `Employees`, и заполняет их тестовыми данными. Выполните скрипт через консоль.
- **Технология решения:**
  1. Создать файл `init.sql`.
  2. Записать команды:

```
sql
CREATE DATABASE CompanyDB;
\c CompanyDB; -- Для PostgreSQL
CREATE TABLE Departments (id SERIAL PRIMARY KEY, name VARCHAR(50));
INSERT INTO Departments (name) VALUES ('IT'), ('HR'), ('Sales');
```
  3. *Технология:* Выполнить скрипт из командной строки (например,

psql -U postgres -f init.sql для PostgreSQL или mysql -u root -p < init.sql для MySQL).

## Тема 2.2. Управление доступом к базам данных

Формируемые компетенции: ПК 1.4, ПК 1.5

### Практическая работа № 6-14. Создание пользователей и назначение ролей.

- **Задание:** Создайте пользователя `analyst_ivanov`. Назначьте ему роль, которая позволяет выполнять `SELECT` на таблицу `Sales`, но запрещает любые изменения (`INSERT`, `UPDATE`, `DELETE`). Настройте шифрование пароля.
- **Технология решения:**
  1. Создать пользователя: `CREATE ROLE analyst_ivanov LOGIN PASSWORD 'SecurePass123!';`
  2. Выдать права: `GRANT SELECT ON Sales TO analyst_ivanov;`
  3. Отозвать права на изменение (по умолчанию их нет, но для проверки): `REVOKE INSERT, UPDATE, DELETE ON Sales FROM analyst_ivanov;`
  4. Проверить: подключиться под новым пользователем и попробовать сделать `DELETE FROM Sales;` (должен быть отказ `Permission denied`).

## Тема 2.3. Резервное копирование и восстановление данных

Формируемые компетенции: ПК 1.4, ПК 1.5

### Практическая работа № 15-18. Выполнение резервного копирования и восстановления.

- **Задание:** Создайте логическую резервную копию базы данных `CompanyDB`. Удалите одну из таблиц. Восстановите базу из резервной копии.
- **Технология решения (PostgreSQL):**
  1. Создать дамп: `pg_dump -U postgres -F c -b -v -f "backup_company.backup" CompanyDB`
  2. Удалить таблицу: `DROP TABLE Employees;`
  3. Восстановить: `pg_restore -U postgres -d CompanyDB -v "backup_company.backup"`
  4. *Технология:* Убедиться, что таблица `Employees` и данные в ней вернулись в исходное состояние.

## Тема 2.4 - 2.5. Мониторинг и безопасность

Формируемые компетенции: ПК 1.5

### Практическая работа № 19-24. Аудит безопасности и защищенные соединения.

- **Задание:** Проанализируйте журналы (логи) сервера БД на предмет неудачных попыток входа. Составьте инструкцию по настройке SSL-соединения для клиента.
- **Технология решения:**
  1. Найти файл конфигурации (например, `postgresql.conf` или `my.cnf`) и проверить параметры логирования (`log_connections = on`,

`log_disconnections = on`).

2. Открыть лог-файл и найти записи `FATAL: password authentication failed for user....`
3. Для SSL: указать в конфиге `ssl = on`, сгенерировать сертификаты (`openssl req -new -x509 -days 365 -nodes -text -out server.crt -keyout server.key`), перезапустить сервер.
4. Настроить клиент для использования `sslmode=require` при подключении.

#### **4. Критерии оценивания практических работ**

Согласно программе модуля (Раздел 5):

- **Оценка «отлично»:** Студент демонстрирует глубокое понимание архитектуры СУБД, самостоятельно пишет сложные SQL-запросы, процедуры и триггеры, безошибочно выполняет администрирование, настройку прав доступа и резервное копирование.
- **Оценка «хорошо»:** Работа выполнена в полном объеме, но допущены незначительные неточности в синтаксисе SQL или при настройке параметров, которые исправлены с помощью преподавателя.
- **Оценка «удовлетворительно»:** Студент знает базовый синтаксис, но испытывает затруднения при написании хранимых процедур/триггеров, оптимизации запросов или настройке прав доступа.
- **Оценка «неудовлетворительно»:** Задания не выполнены, студент не может объяснить принципы работы транзакций, индексов, репликации и безопасности.