# Техническое задание

на курсовую работу по дисциплине «Базы данных»

## 1. Цель работы

Создать полноценную **информационную систему с реляционной базой данных**, демонстрирующую навыки:

- проектирования структуры базы данных;
- реализации связей и ограничений целостности;
- написания SQL-запросов и функций;
- работы с триггерами и аудитом изменений;
- интеграции базы данных с backend-приложением.

## 2. Требования к базе данных

### 2.1. Структура

Предметная область согласовывается с преподавателем, ведущим практические занятия. В общем случае предметная область может быть любой.

- Не менее **9–10 таблиц**, связанных по типам $1:1$, $1:N$, $N:M$;
- Каждая таблица содержит не менее **5 атрибутов разных типов**;
- Обязательно наличие:
  - ключевых сущностей (например, `Users`, `Orders`, `Products`);
  - **журнала аудита** для отслеживания изменений;

### 2.2. Ограничения целостности

- Использовать `PRIMARY KEY`, `FOREIGN KEY`, `UNIQUE`, `CHECK`, `NOT NULL`;
- Обеспечить каскадное обновление и удаление при связях.

### 2.3. Объём и наполнение

- Крупные таблицы — **500–1000 записей**, транзакционные — не менее **5000**;
- Данные должны быть реалистичными;
- Предусмотреть **батчевую загрузку данных** с параметрами и логированием ошибок.

## 3. SQL-функциональность

### 3.1. CRUD-операции

- Реализовать через backend-API;
- Базовые CRUD — через ORM;
- Сложные выборки и отчёты — через чистый SQL с использованием `JOIN`, агрегатов и подзапросов.

### 3.2. Триггеры и функции

**Триггеры:**

- аудит изменений при `INSERT`, `UPDATE`, `DELETE`;
- автоматическое обновление агрегированных данных.

**Функции:**

- скалярные — расчёт итогов, рейтингов, показателей;
- табличные — отчёты и сводные таблицы.

### 3.3. Представления (VIEW)

Необходимо реализовать не менее трёх представлений:

- агрегированные данные (например, продажи по пользователям);
- аналитические отчёты.

## 4. Оптимизация и анализ запросов

- Создать индексы для полей, участвующих в `WHERE`, `JOIN`, `ORDER BY` (где это уместно);
- Показать улучшение производительности с помощью `EXPLAIN ANALYZE` (до/после индексов).

## 5. Интеграция с backend

- Язык реализации — любой (например, Python, Java, C#);
- API должно обеспечивать:
  - CRUD-операции;
  - агрегации и отчёты;
  - массовую загрузку данных (`/api/batch-import`);
  - транзакции, изменяющие несколько таблиц.
- Обязательно наличие документации API (Swagger / OpenAPI).

## 6. Интерфейс пользователя

Полноценный фронтенд разрабатывать не требуется, однако его наличие приветствуется. Для демонстрации достаточно качественно оформленного интерфейса Swagger.

## 7. Контейнеризация и запуск

- Использовать **Docker и docker-compose** (для базы данных, backend-сервиса(-ов) и frontend (при наличии));
- Предоставить инструкции по развёртыванию и наполнению базы данных.

## 8. Демонстрация работы системы

На защите курсовой работы необходимо продемонстрировать:

1. структуру и связи таблиц;

2. выполнение CRUD-операций и сложных SQL-запросов;
3. работу триггеров и аудита изменений;
4. оптимизацию запросов с использованием индексов;
5. процесс батчевой загрузки данных и обработку ошибок;
6. выполнение функций и представлений.

## 9. Пояснительная записка к курсовой работе

К работе должна быть подготовлена пояснительная записка, оформленная в соответствии с требованиями **ГОСТ 7.32–2017 «Отчёт о научно-исследовательской работе. Структура и правила оформления»**.

### Общие требования

- Объём пояснительной записки — **не более 30 страниц основного текста** (без приложений).
- Текст должен быть набран шрифтом Times New Roman, размер 14 pt, межстрочный интервал 1.5.
- Поля: левое — 30 мм, правое — 15 мм, верхнее и нижнее — 20 мм.

### Структура пояснительной записки

1. Титульный лист;
2. Содержание;
3. Введение (объект и предмет исследования) не более страницы;
4. Аналитическая часть (обзор предметной области, постановка задачи) не более 2 страниц;
5. Проектная часть:
    - описание архитектуры системы;
    - проектирование структуры базы данных (ER-диаграммы, связи, ключи, ограничения);
    - описание таблиц, индексов, представлений, функций, триггеров;
    - описание API и взаимодействия с БД;
    - примеры SQL-запросов, результаты выполнения, анализ производительности.
6. Технологическая часть (контейнеризация, запуск, тестирование);
7. Заключение (результаты, оценка эффективности);
8. Приложения (при необходимости: схемы, скрипты, фрагменты кода, Swagger-документация).

### Требования к содержанию

Пояснительная записка должна:

- содержать полное описание созданного приложения с акцентом на архитектуру и базу данных;
- раскрывать логику проектирования, структуру таблиц, связи, ограничения и методы оптимизации;

- включать результаты тестирования и анализа производительности;
- подтверждать корректность и устойчивость работы системы;
- иметь аккуратное и единообразное оформление всех разделов, таблиц, рисунков и ссылок.

## 10. Система оценивания

Оценивание производится по 10 критериям. Каждый критерий оценивается в диапазоне 0–10 баллов. Максимальное количество — 100 баллов. Итоговая оценка переводится в пятибалльную шкалу.

| **№** | **Критерий оценки** | **Макс. балл** | **Примечание** |
|---|---|---|---|
| 1 | Разработка и обоснование структуры базы данных | 15 | Полнота и корректность модели |
| 2 | Использование ограничений целостности (`PK`, `FK`, `CHECK`, `NOT NULL`) | 10 | Грамотная реализация и обоснование |
| 3 | Реализация CRUD-операций и корректность API | 20 | Наличие всех операций, стабильность |
| 4 | Реализация триггеров и функций (скалярных и табличных) | 10 | Работоспособность, логическая корректность |
| 5 | Реализация представлений (`VIEW`) и аналитических запросов | 10 | Полнота, корректность агрегаций |
| 6 | Наличие и качество журнала аудита | 5 | Фиксация изменений, информативность |
| 7 | Оптимизация и анализ запросов (`EXPLAIN ANALYZE`, индексы) | 10 | Демонстрация улучшения производительности |
| 8 | Реализация батчевой загрузки данных и логирования ошибок | 5 | Корректная работа, устойчивость |
| 9 | Интеграция с backend и документирование API (Swagger/OpenAPI) | 10 | Полнота и функциональность |
| 10 | Качество пояснительной записки и демонстрации проекта | 5 | Соответствие ГОСТ, логичность и полнота описания |
| **Итого** | | **100** | |

**Перевод баллов в оценку**

- **90–100 баллов** — оценка «5» (отлично): работа выполнена полностью, демонстрирует высокий уровень самостоятельности, полноту и глубину проработки.
- **75–89 баллов** — оценка «4» (хорошо): работа выполнена в основном правильно, но имеет отдельные недочёты или незначительные упрощения.
- **60–74 балла** — оценка «3» (удовлетворительно): выполнена частично, содержит методические ошибки, отсутствуют отдельные функциональные элементы.
- $<$ **60 баллов** — оценка «2» (неудовлетворительно): не соответствует требованиям задания или не демонстрируется в работе.

## 11. Штрафы и ограничения оценки

Для исключения небезопасных и непрофессиональных решений вводятся ограничения. Нарушения снижают итоговую оценку независимо от общего балла.

**1. Хранение учётных данных в коде**

- Запрещено хранить пароли, ключи, URI баз данных и иные секреты в исходном коде или репозитории.
- При обнаружении — **оценка не выше «3»**.

**2. Использование `f-string` или конкатенации в SQL**

- Прямое формирование SQL-запросов через `f-string` или конкатенацию без параметризации считается уязвимым к SQL-инъекциям.
- В таком случае — **оценка не выше «4»**.

**3. Вайбкодинг (код без понимания)**

- Если код сгенерирован или студент не может объяснить его логику — **оценка не выше «2–4»** в зависимости от уровня понимания и выполнения общих формальных требований.
- Возможен дополнительный опрос или задание для подтверждения авторства.