Disynergy
Страница материала ↗
EXQT
Готовые ответы — на EXQT
Доступ к ответам приобретается отдельно.
Найти ответы ↗

Этап 1. Проектирование базы данных

1. Реляционная модель данных

Реляционная база данных — это совокупность данных, организованных в виде таблиц (отношений), связанных между собой по ключевым полям. Модель предложена Эдгаром Коддом в 1970 году и по сей день является основой большинства промышленных СУБД: MySQL, PostgreSQL, MS SQL Server, Oracle.

Каждая таблица в реляционной БД:

- имеет уникальное имя в пределах базы данных;
- состоит из строк (записей, кортежей) и столбцов (атрибутов, полей);
- каждый столбец хранит данные только одного типа;
- порядок строк и столбцов не имеет семантического значения;
- каждая строка уникально идентифицируется первичным ключом.

---

2. Нормализация базы данных

Нормализация — это процесс организации данных в таблицах с целью устранения избыточности и предотвращения аномалий при вставке, обновлении и удалении данных. Нормализация выполняется поэтапно по нормальным формам.

2.1 Первая нормальная форма (1НФ)

Таблица находится в 1НФ, если:

- все значения атомарны (неделимы);
- каждая ячейка содержит ровно одно значение;
- каждый столбец имеет уникальное имя;
- в каждой строке хранятся данные только об одном объекте.

Нарушение 1НФ — хранение нескольких телефонов в одном поле:

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

order_id	client	phones
1	Иванов	+7-900-111, +7-900-222

Исправление: выделить телефоны в отдельную таблицу `client_phones`.

2.2 Вторая нормальная форма (2НФ)

Таблица находится в 2НФ, если она находится в 1НФ и каждый неключевой атрибут **полностью функционально зависит** от первичного ключа (актуально для составных ключей).

Нарушение 2НФ — таблица с составным ключом `(order_id, product_id)`, где `product_name` зависит только от `product_id`:

order_id	product_id	product_name	quantity
1	10	Клавиатура	2
2	10	Клавиатура	1

`product_name` зависит только от `product_id`, а не от пары `(order_id, product_id)` — это частичная зависимость.

Исправление: вынести `product_name` в отдельную таблицу `products`.

2.3 Третья нормальная форма (3НФ)

Таблица находится в 3НФ, если она находится в 2НФ и отсутствуют **транзитивные зависимости** неключевых атрибутов от первичного ключа.

Нарушение 3НФ — атрибут `city` зависит от `zip_code`, а `zip_code` зависит от `user_id`:

user_id	name	zip_code	city
1	Иванов	141700	Долгопрудный
2	Петров	141700	Долгопрудный

`city` транзитивно зависит от `user_id` через `zip_code`.

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Исправление: вынести `zip_code → city` в отдельную таблицу `zip_codes`.

> **Практический совет**
>
> В большинстве учебных и производственных проектов достаточно привести схему к 3НФ. Более высокие нормальные формы (BCNF, 4НФ, 5НФ) применяются в специализированных случаях.

---

3. Типы связей между таблицами

3.1 Один к одному (1:1)

Каждой записи в таблице A соответствует не более одной записи в таблице B, и наоборот. Используется редко — обычно для разделения редко используемых полей (например, основная информация о сотруднике и расширенный профиль).

employees	employee_profiles
employee_id (PK)	profile_id (PK, FK → employees)
name	bio
position	photo_url

3.2 Один ко многим (1:N)

Наиболее распространённый тип связи. Одной записи в таблице A соответствует множество записей в таблице B. Внешний ключ хранится на стороне "многих".

Пример: один отдел → много сотрудников.

departments	employees
dept_id (PK)	emp_id (PK)
dept_name	emp_name
	dept_id (FK → departments)

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер disynergy.ru

3.3 Многие ко многим (N:M)

Одной записи таблицы A соответствует множество записей таблицы B и наоборот. В реляционной модели такая связь **не может быть реализована напрямую** — она разрешается введением промежуточной (связующей, junction) таблицы.

Пример: студенты и курсы — один студент посещает несколько курсов, один курс посещается несколькими студентами.

Без промежуточной таблицы:

students	**courses**
student_id (PK)	course_id (PK)
name	title

С промежуточной таблицей `enrollments`:

enrollments
student_id (FK → students)
course_id (FK → courses)
enrolled_at
grade

Первичный ключ промежуточной таблицы — составной: `(student_id, course_id)`.

---

4. ER-диаграммы

ER-диаграмма (Entity-Relationship Diagram, диаграмма «сущность–связь») — графический инструмент моделирования данных, введённый Питером Ченом в 1976 году.

4.1 Основные элементы нотации

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Элемент	Описание
Сущность (Entity)	Прямоугольник — объект реального мира, о котором хранятся данные
Атрибут (Attribute)	Овал (классическая нотация) или список внутри прямоугольника (IE/Crow's Foot)
Связь (Relationship)	Ромб (классическая нотация) или линия между таблицами (IE-нотация)
Первичный ключ	Подчёркивание атрибута (классическая) или маркер PK
Внешний ключ	Маркер FK, стрелка к PK родительской таблицы

4.2 Нотация "Вороньей лапки" (Crow's Foot / IE-нотация)

Современные инструменты (MySQL Workbench, dbdiagram.io, Mermaid) используют IE-нотацию:

Символ на конце линии	Смысл
`||`	Ровно один
`|o`	Ноль или один
`}|` или `>|`	Один или много
`}o` или `>o`	Ноль или много

4.3 Пример ER-диаграммы (Mermaid)

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

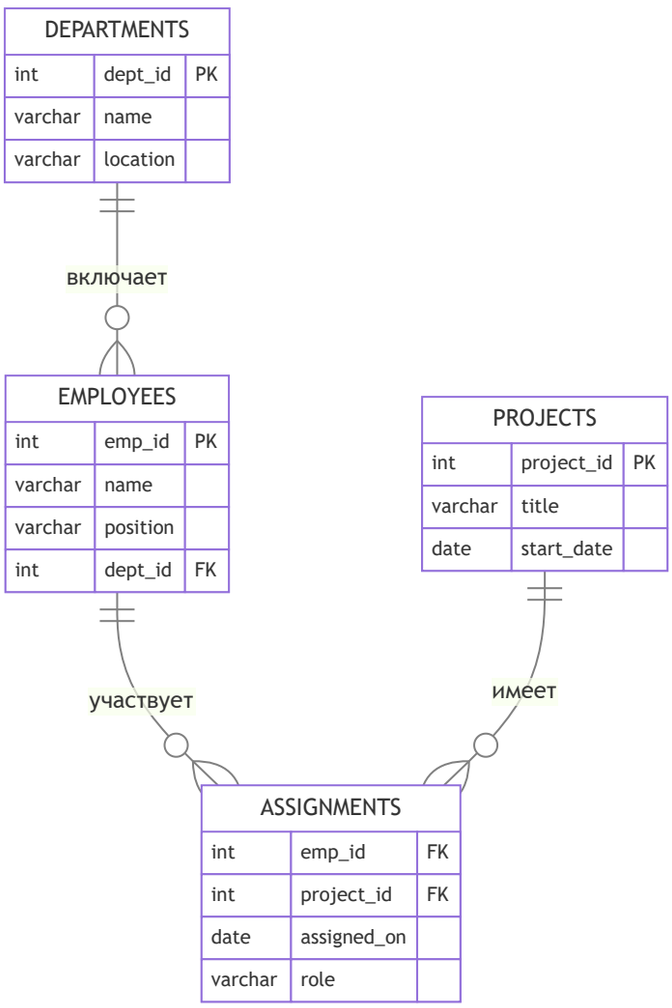

Рисунок 1 — ER-диаграмма: отделы, сотрудники, проекты (IE-нотация, Mermaid)

---

5. Первичные и внешние ключи

5.1 Первичный ключ (PRIMARY KEY)

Первичный ключ — атрибут (или группа атрибутов), однозначно идентифицирующий каждую строку таблицы.

Требования к первичному ключу:

- уникальность: не может быть двух строк с одинаковым значением PK;
- ненулевость: значение PK не может быть NULL;

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

- неизменность: желательно не изменять значение PK после создания строки;
- минимальность: составной PK не должен содержать лишних атрибутов.

На практике чаще всего используют **суррогатный ключ** — целочисленный автоинкрементный идентификатор, не имеющий смысла в предметной области.

5.2 Внешний ключ (FOREIGN KEY)

Внешний ключ — атрибут в дочерней таблице, ссылающийся на первичный ключ родительской таблицы. FK обеспечивает **ссылочную целостность**: СУБД не позволит вставить значение FK, которого нет в PK родительской таблицы.

Действия при удалении/обновлении родительской записи (`ON DELETE` / `ON UPDATE`):

Действие	Поведение
`RESTRICT`	Запрещает удаление/обновление, если есть дочерние записи
`CASCADE`	Каскадно удаляет/обновляет дочерние записи
`SET NULL`	Устанавливает FK в NULL у дочерних записей
`NO ACTION`	Аналогично RESTRICT (проверяется в конце транзакции)
`SET DEFAULT`	Устанавливает значение по умолчанию (поддерживается не всеми СУБД)

---

6. Индексы

Индекс — вспомогательная структура данных, ускоряющая поиск строк по значению столбца. Работает по принципу предметного указателя книги: вместо перебора всех строк СУБД обращается к индексу и получает адрес нужной строки.

6.1 Виды индексов

Вид	Описание
PRIMARY (кластерный)	Создаётся автоматически для PRIMARY KEY; строки физически упорядочены по нему (InnoDB)

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Вид	Описание
UNIQUE	Гарантирует уникальность значений столбца; ускоряет поиск
INDEX (не уникальный)	Ускоряет поиск, не гарантирует уникальность
FULLTEXT	Полнотекстовый поиск по TEXT/VARCHAR столбцам
COMPOSITE	Индекс по нескольким столбцам одновременно

6.2 Когда создавать индексы

Индексы рекомендуется создавать:

- на столбцах, используемых в `WHERE` фильтрах часто выполняемых запросов;
- на столбцах внешних ключей (JOIN-операции);
- на столбцах, используемых в `ORDER BY` и `GROUP BY`;
- на столбцах с высокой кардинальностью (много уникальных значений).

Индексы **замедляют** операции `INSERT`, `UPDATE`, `DELETE`, так как СУБД должна обновлять индексную структуру. Поэтому не стоит создавать индексы на всех столбцах подряд.

---

7. Проектирование схемы данных: от требований к таблицам

Процесс проектирования БД включает следующие этапы:

Шаг 1. Анализ предметной области. Изучить бизнес-требования, выявить объекты (сущности), их свойства (атрибуты) и взаимодействия (связи).

Шаг 2. Концептуальная модель. Построить ER-диаграмму — независимо от конкретной СУБД. Определить сущности, атрибуты, типы связей.

Шаг 3. Логическая модель. Преобразовать ER-диаграмму в набор таблиц с учётом нормализации. Определить первичные и внешние ключи. Разрешить связи N:M через промежуточные таблицы.

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Шаг 4. Физическая модель. Выбрать конкретные типы данных для каждого столбца с учётом используемой СУБД. Определить ограничения (`NOT NULL`, `UNIQUE`, `DEFAULT`, `CHECK`). Создать индексы.

Шаг 5. DDL-скрипт. Написать и выполнить скрипт `CREATE TABLE` для выбранной СУБД.

---

8. Сравнение типов данных в MySQL, PostgreSQL и MS SQL Server

Правильный выбор типа данных влияет на объём хранилища, производительность запросов и корректность обработки данных.

8.1 Целочисленные типы

Назначение	MySQL	PostgreSQL	MS SQL Server
Небольшое целое	`TINYINT` (1 байт)	`SMALLINT` (2 байта)	`TINYINT` (1 байт)
Стандартное целое	`INT` (4 байта)	`INTEGER` / `INT`	`INT` (4 байта)
Большое целое	`BIGINT` (8 байт)	`BIGINT`	`BIGINT`
Автоинкремент	`INT AUTO_INCREMENT`	`SERIAL` / `GENERATED ALWAYS AS IDENTITY`	`INT IDENTITY(1,1)`

8.2 Строковые типы

Назначение	MySQL	PostgreSQL	MS SQL Server
Короткая строка фиксированной длины	`CHAR(n)`	`CHAR(n)`	`CHAR(n)` / `NCHAR(n)`

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Назначение	MySQL	PostgreSQL	MS SQL Server
Строка переменной длины	`VARCHAR(n)`	`VARCHAR(n)`	`VARCHAR(n)` / `NVARCHAR(n)`
Длинный текст	`TEXT` / `MEDIUMTEXT` / `LONGTEXT`	`TEXT` (без ограничения)	`VARCHAR(MAX)` / `NVARCHAR(MAX)`

Unicode в MS SQL

В MS SQL Server для хранения Unicode (кириллицы, иероглифов и т.д.) используйте типы `NCHAR`, `NVARCHAR`, `NVARCHAR(MAX)` — они хранят символы в кодировке UTF-16. Обычные `VARCHAR` хранят данные в кодировке сервера.

8.3 Числа с дробной частью

Назначение	MySQL	PostgreSQL	MS SQL Server
Точная дробная (деньги)	`DECIMAL(p,s)` / `NUMERIC(p,s)`	`NUMERIC(p,s)`	`DECIMAL(p,s)` / `MONEY`
Приближённая дробная	`FLOAT` / `DOUBLE`	`REAL` / `DOUBLE PRECISION`	`FLOAT` / `REAL`

8.4 Дата и время

Назначение	MySQL	PostgreSQL	MS SQL Server
Только дата	`DATE`	`DATE`	`DATE`
Только время	`TIME`	`TIME`	`TIME`
Дата и время	`DATETIME`	`TIMESTAMP`	`DATETIME` / `DATETIME2`

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Назначение	MySQL	PostgreSQL	MS SQL Server
С часовым поясом	`DATETIME` (без TZ)	`TIMESTAMPTZ`	`DATETIMEOFFSET`
Текущее время	`NOW()` / `CURRENT_TIMESTAMP`	`NOW()` / `CURRENT_TIMESTAMP`	`GETDATE()` / `SYSDATETIME()`

DATETIME vs TIMESTAMP в MySQL

В MySQL `DATETIME` хранит значение без учёта часового пояса (диапазон 1000–9999 г.), а `TIMESTAMP` автоматически конвертирует в UTC при хранении и обратно при чтении (диапазон 1970–2038 г.). Для новых проектов рекомендуется использовать `DATETIME`.

8.5 Логический тип

СУБД	Тип	Значения
MySQL	`TINYINT(1)` или `BOOLEAN` (псевдоним)	0 / 1
PostgreSQL	`BOOLEAN`	`TRUE` / `FALSE` / `NULL`
MS SQL Server	`BIT`	0 / 1 / NULL

8.6 Прочие типы

Назначение	MySQL	PostgreSQL	MS SQL Server
UUID / GUID	`CHAR(36)` или `BINARY(16)`	`UUID`	`UNIQUEIDENTIFIER`
JSON	`JSON` (с MySQL 5.7)	`JSON` / `JSONB`	`NVARCHAR(MAX)` + `ISJSON()` / `JSON` (с 2016)

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Назначение	MySQL	PostgreSQL	MS SQL Server
Бинарные данные	`BLOB` / `LONGBLOB`	`BYTEA`	`VARBINARY(MAX)`
Перечисление	`ENUM('val1','val2')`	Тип-перечисление (`CREATE TYPE`)	`VARCHAR` + `CHECK`

---

9. DDL-примеры: создание таблиц в трёх СУБД

Рассмотрим типичные таблицы для системы управления заказами и покажем синтаксис для каждой СУБД.

9.1 Таблица `orders`

MySQL

```
CREATE TABLE orders (
    order_id    INT           NOT NULL AUTO_INCREMENT,
    client_id   INT           NOT NULL,
    order_date  DATETIME      NOT NULL DEFAULT NOW(),
    total       DECIMAL(10,2) NOT NULL DEFAULT 0.00,
    status      VARCHAR(30)   NOT NULL DEFAULT 'new',
    note        TEXT,
    PRIMARY KEY (order_id),
    CONSTRAINT fk_order_client FOREIGN KEY (client_id)
        REFERENCES clients (client_id)
        ON DELETE RESTRICT
        ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

PostgreSQL

```
CREATE TABLE orders (
    order_id    SERIAL         NOT NULL,
    client_id   INTEGER        NOT NULL,
    order_date  TIMESTAMPTZ    NOT NULL DEFAULT NOW(),
    total       NUMERIC(10,2)  NOT NULL DEFAULT 0.00,
    status      VARCHAR(30)    NOT NULL DEFAULT 'new',
    note        TEXT,
    PRIMARY KEY (order_id),
    CONSTRAINT fk_order_client FOREIGN KEY (client_id)
        REFERENCES clients (client_id)
        ON DELETE RESTRICT
        ON UPDATE CASCADE
);
```

MS SQL Server

```
CREATE TABLE orders (
    order_id    INT             NOT NULL IDENTITY(1,1),
    client_id   INT             NOT NULL,
    order_date  DATETIME2       NOT NULL DEFAULT SYSDATETIME(),
    total       DECIMAL(10,2)   NOT NULL DEFAULT 0.00,
    status      NVARCHAR(30)    NOT NULL DEFAULT N'new',
    note        NVARCHAR(MAX),
    PRIMARY KEY (order_id),
    CONSTRAINT fk_order_client FOREIGN KEY (client_id)
        REFERENCES clients (client_id)
        ON DELETE NO ACTION
        ON UPDATE CASCADE
);
```

9.2 Таблица `clients`

MySQL

```
CREATE TABLE clients (
    client_id   INT          NOT NULL AUTO_INCREMENT,
    full_name   VARCHAR(150) NOT NULL,
    email       VARCHAR(150) NOT NULL,
    phone       VARCHAR(20),
    created_at  DATETIME     NOT NULL DEFAULT NOW(),
    is_active   TINYINT(1)   NOT NULL DEFAULT 1,
    PRIMARY KEY (client_id),
    UNIQUE KEY uq_client_email (email)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

PostgreSQL

```
CREATE TABLE clients (
    client_id   SERIAL       NOT NULL,
    full_name   VARCHAR(150) NOT NULL,
    email       VARCHAR(150) NOT NULL,
    phone       VARCHAR(20),
    created_at  TIMESTAMPTZ  NOT NULL DEFAULT NOW(),
    is_active   BOOLEAN      NOT NULL DEFAULT TRUE,
    PRIMARY KEY (client_id),
    UNIQUE (email)
);
```

MS SQL Server

```
CREATE TABLE clients (
    client_id   INT             NOT NULL IDENTITY(1,1),
    full_name   NVARCHAR(150)   NOT NULL,
    email       NVARCHAR(150)   NOT NULL,
    phone       NVARCHAR(20),
    created_at  DATETIME2       NOT NULL DEFAULT SYSDATETIME(),
    is_active   BIT             NOT NULL DEFAULT 1,
    PRIMARY KEY (client_id),
    UNIQUE (email)
);
```

9.3 Создание индексов

MySQL

```
-- Индекс на внешний ключ (ускоряет JOIN)
CREATE INDEX idx_orders_client ON orders (client_id);

-- Составной индекс (фильтрация по статусу и дате)
CREATE INDEX idx_orders_status_date ON orders (status, order_date);
```

PostgreSQL

```
CREATE INDEX idx_orders_client ON orders (client_id);
CREATE INDEX idx_orders_status_date ON orders (status, order_date);

-- Частичный индекс: только активные заказы
CREATE INDEX idx_orders_active ON orders (order_date)
WHERE status NOT IN ('completed', 'cancelled');
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

MS SQL Server

```
CREATE INDEX idx_orders_client ON orders (client_id);
CREATE INDEX idx_orders_status_date ON orders (status, order_date);

-- Фильтрованный индекс
CREATE INDEX idx_orders_active ON orders (order_date)
WHERE status NOT IN ('completed', 'cancelled');
```

10. Полный пример схемы: три таблицы с нормализацией

Приведём пример небольшой схемы из трёх таблиц для системы учёта обращений. Схема полностью нормализована до 3НФ.

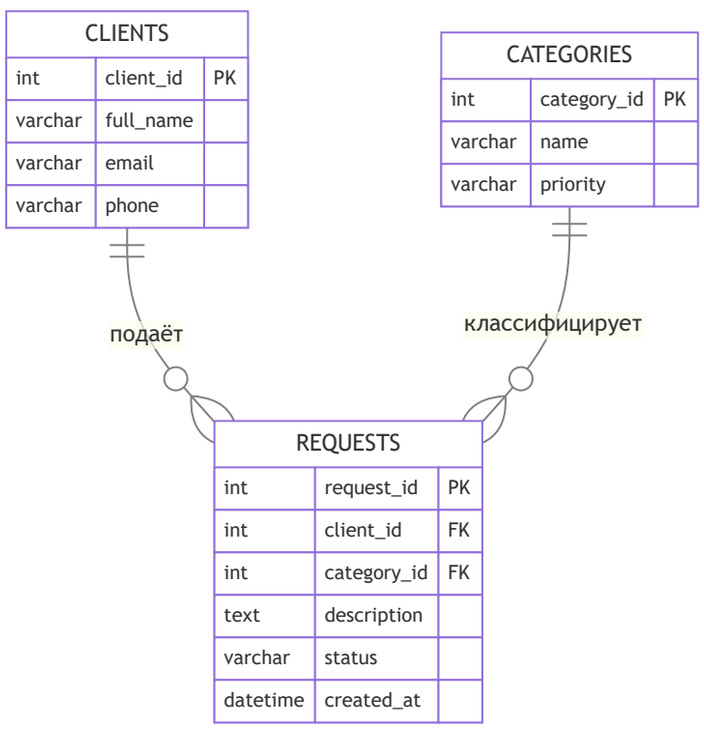

Рисунок 2 — Упрощённая ER-диаграмма системы учёта обращений (3 таблицы)

Схема удовлетворяет требованиям 3НФ:

- каждый атрибут атомарен (1НФ);
- нет частичных зависимостей от составного ключа (2НФ, ключи простые);

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

- нет транзитивных зависимостей: `priority` хранится в `categories`, а не в `requests` (иначе возникла бы транзитивность через `category_id`).

---

11. Итоги

Проектирование базы данных — фундаментальный этап разработки информационной системы. Грамотная схема данных:

- обеспечивает целостность и непротиворечивость данных;
- снижает избыточность хранения;
- упрощает написание запросов и снижает вероятность ошибок;
- облегчает расширение системы в будущем.

При проектировании следует руководствоваться принципами нормализации (1НФ–3НФ), правильно выбирать типы связей и типы данных, а также документировать схему с помощью ER-диаграмм.

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Этап 2. Объекты БД — VIEW, хранимые процедуры, триггеры

1. Представления (VIEW)

1.1 Что такое VIEW

Представление (VIEW) — это именованный хранимый SQL-запрос, к которому можно обращаться как к обычной таблице. VIEW не хранит данные физически (в большинстве СУБД); при каждом обращении к представлению СУБД выполняет исходный запрос и возвращает результат.

Представления используются для:

- **упрощения сложных запросов** — вместо многострочного JOIN можно написать `SELECT * FROM v_my_view;`
- **разграничения доступа** — пользователю предоставляется доступ к VIEW, а не к базовым таблицам, что скрывает чувствительные столбцы;
- **унификации логики** — бизнес-правила формирования выборки определяются один раз в VIEW, а не дублируются в каждом запросе приложения;
- **обратной совместимости** — при изменении структуры таблицы можно сохранить интерфейс VIEW.

1.2 Синтаксис CREATE VIEW

MySQL

```
-- Создание представления
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;

-- Пересоздание (замена) без ошибки
CREATE OR REPLACE VIEW view_name AS
SELECT ...;

-- Удаление
DROP VIEW IF EXISTS view_name;
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

PostgreSQL

```sql
CREATE OR REPLACE VIEW view_name AS
SELECT column1, column2
FROM table_name
WHERE condition;

-- Удаление
DROP VIEW IF EXISTS view_name;
```

MS SQL Server

```sql
-- Создание
CREATE VIEW view_name AS
SELECT column1, column2
FROM table_name
WHERE condition;

-- Изменение (ALTER VIEW заменяет определение)
ALTER VIEW view_name AS
SELECT ...;

-- Удаление
DROP VIEW IF EXISTS view_name;
```

1.3 Примеры VIEW

Простое представление — активные пользователи:

MySQL

```sql
CREATE OR REPLACE VIEW v_active_users AS
SELECT
    user_id,
    full_name,
    email,
    created_at
FROM users
WHERE is_active = 1;
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

PostgreSQL

```postgresql
CREATE OR REPLACE VIEW v_active_users AS
SELECT
    user_id,
    full_name,
    email,
    created_at
FROM users
WHERE is_active = TRUE;
```

MS SQL Server

```mssql
CREATE VIEW v_active_users AS
SELECT
    user_id,
    full_name,
    email,
    created_at
FROM users
WHERE is_active = 1;
```

VIEW с JOIN — заказы с именем клиента и категории:

MySQL

```mysql
CREATE OR REPLACE VIEW v_orders_detail AS
SELECT
    o.order_id,
    o.order_date,
    o.status,
    o.total,
    c.full_name  AS client_name,
    c.email      AS client_email,
    cat.name     AS category_name
FROM orders o
JOIN clients c      ON o.client_id   = c.client_id
JOIN categories cat ON o.category_id = cat.category_id;
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

PostgreSQL

```
CREATE OR REPLACE VIEW v_orders_detail AS
SELECT
    o.order_id,
    o.order_date,
    o.status,
    o.total,
    c.full_name  AS client_name,
    c.email      AS client_email,
    cat.name     AS category_name
FROM orders o
JOIN clients c      ON o.client_id   = c.client_id
JOIN categories cat ON o.category_id = cat.category_id;
```

MS SQL Server

```
CREATE VIEW v_orders_detail AS
SELECT
    o.order_id,
    o.order_date,
    o.status,
    o.total,
    c.full_name  AS client_name,
    c.email      AS client_email,
    cat.name     AS category_name
FROM orders o
JOIN clients c      ON o.client_id   = c.client_id
JOIN categories cat ON o.category_id = cat.category_id;
```

VIEW с агрегатом — количество заказов и средний чек по клиентам:

MySQL

```
CREATE OR REPLACE VIEW v_client_stats AS
SELECT
    c.client_id,
    c.full_name,
    COUNT(o.order_id)   AS order_count,
    AVG(o.total)        AS avg_order_total,
    MAX(o.order_date)   AS last_order_date
FROM clients c
LEFT JOIN orders o ON c.client_id = o.client_id
GROUP BY c.client_id, c.full_name;
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения

Нажмите на мессенджер, чтобы написать

Telegram ↗ WhatsApp ↗ MAX ↗

+7 (924) 305-23-08 · Пишите в мессенджер

disynergy.ru

PostgreSQL

```
CREATE OR REPLACE VIEW v_client_stats AS
SELECT
    c.client_id,
    c.full_name,
    COUNT(o.order_id)        AS order_count,
    AVG(o.total)             AS avg_order_total,
    MAX(o.order_date)        AS last_order_date
FROM clients c
LEFT JOIN orders o ON c.client_id = o.client_id
GROUP BY c.client_id, c.full_name;
```

MS SQL Server

```
CREATE VIEW v_client_stats AS
SELECT
    c.client_id,
    c.full_name,
    COUNT(o.order_id)        AS order_count,
    AVG(o.total)             AS avg_order_total,
    MAX(o.order_date)        AS last_order_date
FROM clients c
LEFT JOIN orders o ON c.client_id = o.client_id
GROUP BY c.client_id, c.full_name;
```

1.4 Обновляемые и необновляемые VIEW

VIEW считается **обновляемым** (через него можно выполнять `INSERT`, `UPDATE`, `DELETE`), если оно:

- основано ровно на одной таблице;
- не содержит `GROUP BY`, `HAVING`, `DISTINCT`, агрегатных функций;
- не содержит подзапросов в `SELECT`;
- не использует `UNION`.

VIEW с `JOIN`, `GROUP BY` или агрегатами — **необновляемые**. Через них можно только читать данные.

Осторожно с обновляемыми VIEW

Даже обновляемые VIEW могут создавать неожиданное поведение. Например, после `INSERT` новая строка может не отображаться в VIEW, если она не соответствует фильтру `WHERE`. Чтобы предотвратить это, используйте опцию `WITH CHECK OPTION`.

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

1.5 Когда использовать VIEW vs запрос

Ситуация	Рекомендация
Запрос используется в одном месте	Обычный SQL-запрос в коде приложения
Запрос используется в нескольких местах	VIEW
Нужно скрыть структуру таблиц от пользователей	VIEW
Нужна максимальная производительность с кешированием	Материализованное VIEW (только PostgreSQL и MS SQL)
Сложный запрос в отчёте	VIEW или хранимая процедура

---

2. Хранимые процедуры

2.1 Что такое хранимая процедура

Хранимая процедура (Stored Procedure) — это именованный блок SQL-кода, хранящийся на сервере БД и выполняемый по вызову. В отличие от VIEW, процедура может содержать:

- условные операторы (`IF` , `CASE`);
- циклы (`WHILE` , `LOOP` , `FOR`);
- объявление переменных;
- обработку ошибок (исключений);
- транзакции (`COMMIT` , `ROLLBACK`);
- несколько SQL-операторов.

Преимущества хранимых процедур:

Безопасность — приложение вызывает процедуру, не имея прямого доступа к таблицам; SQL-инъекция через параметры процедуры значительно сложнее.

Производительность — план выполнения процедуры компилируется и кешируется СУБД; при повторных вызовах разбор SQL-кода не требуется.

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Повторное использование — бизнес-логика определяется один раз на сервере и вызывается из любого приложения (веб, десктоп, мобильное).

Уменьшение трафика — вместо передачи многострочного SQL-запроса клиент передаёт только имя процедуры и параметры.

2.2 Хранимые процедуры в MySQL

В MySQL перед созданием процедуры необходимо изменить разделитель команд с `;` на другой символ (обычно `$$` или `//`), иначе клиент завершит ввод на первой `;` внутри тела процедуры.

Виды параметров:

Вид	Описание
`IN`	Входной параметр — значение передаётся в процедуру (по умолчанию)
`OUT`	Выходной параметр — процедура записывает значение, которое возвращается вызывающей стороне
`INOUT`	Входной и выходной одновременно

MySQL — процедура с IN параметром

```
DELIMITER $$

CREATE PROCEDURE sp_update_order_status(
    IN p_order_id INT,
    IN p_new_status VARCHAR(30)
)
BEGIN
    UPDATE orders
    SET    status     = p_new_status,
           updated_at = NOW()
    WHERE  order_id = p_order_id;
END$$

DELIMITER ;

-- Вызов:
CALL sp_update_order_status(42, 'completed');
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

MySQL — процедура с OUT параметром

```
DELIMITER $$

CREATE PROCEDURE sp_get_client_order_count(
    IN  p_client_id  INT,
    OUT p_order_count INT
)
BEGIN
    SELECT COUNT(*) INTO p_order_count
    FROM   orders
    WHERE  client_id = p_client_id;
END$$

DELIMITER ;

-- Вызов:
CALL sp_get_client_order_count(5, @cnt);
SELECT @cnt;
```

MySQL — процедура с условием и SIGNAL

```
DELIMITER $$

CREATE PROCEDURE sp_place_order(
    IN p_client_id  INT,
    IN p_category_id INT,
    IN p_description TEXT
)
BEGIN
    DECLARE v_active TINYINT DEFAULT 0;

    SELECT is_active INTO v_active
    FROM   clients
    WHERE  client_id = p_client_id;

    IF v_active = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Клиент заблокирован';
    END IF;

    INSERT INTO orders (client_id, category_id, description, status,
order_date)
    VALUES (p_client_id, p_category_id, p_description, 'new', NOW());
END$$

DELIMITER ;
```

2.3 Хранимые процедуры в PostgreSQL

В PostgreSQL хранимые процедуры создаются с помощью `CREATE OR REPLACE PROCEDURE` (с PostgreSQL 11) или через функции (`CREATE FUNCTION ... RETURNS void`).

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Язык тела — `PL/pgSQL` (процедурный язык).

PostgreSQL — процедура (PL/pgSQL)

```
CREATE OR REPLACE PROCEDURE sp_update_order_status(
    p_order_id    INT,
    p_new_status  VARCHAR
)
LANGUAGE plpgsql
AS $$
BEGIN
    UPDATE orders
    SET    status     = p_new_status,
           updated_at = NOW()
    WHERE  order_id = p_order_id;
END;
$$;

-- Вызов:
CALL sp_update_order_status(42, 'completed');
```

PostgreSQL — функция с RETURNS TABLE

```
CREATE OR REPLACE FUNCTION fn_get_client_orders(p_client_id INT)
RETURNS TABLE (
    order_id   INT,
    order_date TIMESTAMPTZ,
    status     VARCHAR,
    total      NUMERIC
)
LANGUAGE plpgsql
AS $$
BEGIN
    RETURN QUERY
    SELECT o.order_id, o.order_date, o.status, o.total
    FROM   orders o
    WHERE  o.client_id = p_client_id
    ORDER  BY o.order_date DESC;
END;
$$;

-- Вызов:
SELECT * FROM fn_get_client_orders(5);
```

2.4 Хранимые процедуры в MS SQL Server

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

MS SQL Server — процедура

```
CREATE OR ALTER PROCEDURE sp_update_order_status
    @order_id   INT,
    @new_status NVARCHAR(30)
AS
BEGIN
    SET NOCOUNT ON;

    UPDATE orders
    SET    status     = @new_status,
           updated_at = SYSDATETIME()
    WHERE  order_id = @order_id;
END;
GO

-- Вызов:
EXEC sp_update_order_status @order_id = 42, @new_status = N'completed';
```

MS SQL Server — процедура с OUTPUT параметром

```
CREATE OR ALTER PROCEDURE sp_insert_client
    @full_name   NVARCHAR(150),
    @email       NVARCHAR(150),
    @new_id      INT OUTPUT
AS
BEGIN
    SET NOCOUNT ON;
    INSERT INTO clients (full_name, email, created_at, is_active)
    VALUES (@full_name, @email, SYSDATETIME(), 1);

    SET @new_id = SCOPE_IDENTITY();
END;
GO

-- Вызов:
DECLARE @id INT;
EXEC sp_insert_client N'Иванов Иван', N'ivanov@mail.ru', @new_id = @id OUTPUT;
SELECT @id AS new_client_id;
```

---

3. Триггеры

3.1 Что такое триггер

Триггер (Trigger) — это именованный объект БД, автоматически выполняемый СУБД в ответ на событие модификации данных в таблице: `INSERT`, `UPDATE` или `DELETE`.

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер disynergy.ru

Триггеры не вызываются явно — они срабатывают автоматически при наступлении связанного события. Это делает их мощным инструментом для:

- **аудита** — записи истории изменений (кто, что, когда изменил);
- **поддержания целостности** — проверки дополнительных бизнес-правил, которые нельзя выразить через `CHECK` ограничения;
- **автоматизации** — автоматического заполнения полей (например, `updated_at`);
- **синхронизации** — каскадного обновления данных в связанных таблицах.

3.2 Виды триггеров

Тип	Описание
`BEFORE INSERT`	Срабатывает до вставки строки; можно изменить значения `NEW`
`AFTER INSERT`	Срабатывает после вставки строки
`BEFORE UPDATE`	Срабатывает до обновления; доступны `OLD` (старые значения) и `NEW` (новые)
`AFTER UPDATE`	Срабатывает после обновления
`BEFORE DELETE`	Срабатывает до удаления; доступны значения `OLD`
`AFTER DELETE`	Срабатывает после удаления

В MySQL и PostgreSQL триггер срабатывает **для каждой строки** (`FOR EACH ROW`). В MS SQL Server триггеры срабатывают **один раз для всей операции**, но через виртуальные таблицы `INSERTED` и `DELETED` можно обработать все затронутые строки.

3.3 Триггеры в MySQL

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

MySQL — AFTER UPDATE: логирование изменений (audit log)

```
DELIMITER $$

CREATE TRIGGER trg_orders_audit
AFTER UPDATE ON orders
FOR EACH ROW
BEGIN
    IF OLD.status <> NEW.status THEN
        INSERT INTO orders_audit_log
            (order_id, old_status, new_status, changed_at)
        VALUES
            (OLD.order_id, OLD.status, NEW.status, NOW());
    END IF;
END$$

DELIMITER ;
```

MySQL — BEFORE UPDATE: автоматический timestamp

```
DELIMITER $$

CREATE TRIGGER trg_orders_updated_at
BEFORE UPDATE ON orders
FOR EACH ROW
BEGIN
    SET NEW.updated_at = NOW();
END$$

DELIMITER ;
```

MySQL — BEFORE INSERT: проверка значений

```
DELIMITER $$

CREATE TRIGGER trg_orders_check_total
BEFORE INSERT ON orders
FOR EACH ROW
BEGIN
    IF NEW.total < 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Сумма заказа не может быть отрицательной';
    END IF;
END$$

DELIMITER ;
```

В MySQL таблицы `OLD` и `NEW` содержат значения строки до и после изменения:

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Событие	`OLD`	`NEW`
`INSERT`	недоступно	вставляемая строка
`UPDATE`	строка до изменения	строка после изменения
`DELETE`	удаляемая строка	недоступно

3.4 Триггеры в PostgreSQL

В PostgreSQL триггер состоит из двух объектов: **триггерной функции** (возвращает тип `TRIGGER`) и самого **триггера**, привязывающего функцию к таблице и событию.

PostgreSQL — AFTER UPDATE: логирование изменений

```
-- Шаг 1: создаём функцию-триггер
CREATE OR REPLACE FUNCTION fn_orders_audit()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
    IF OLD.status IS DISTINCT FROM NEW.status THEN
        INSERT INTO orders_audit_log (order_id, old_status, new_status,
changed_at)
        VALUES (OLD.order_id, OLD.status, NEW.status, NOW());
    END IF;
    RETURN NEW;
END;
$$;

-- Шаг 2: создаём триггер, привязанный к функции
CREATE OR REPLACE TRIGGER trg_orders_audit
AFTER UPDATE ON orders
FOR EACH ROW
EXECUTE FUNCTION fn_orders_audit();
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

PostgreSQL — BEFORE UPDATE: автоматический timestamp

```
CREATE OR REPLACE FUNCTION fn_set_updated_at()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
    NEW.updated_at := NOW();
    RETURN NEW;
END;
$$;

CREATE OR REPLACE TRIGGER trg_orders_updated_at
BEFORE UPDATE ON orders
FOR EACH ROW
EXECUTE FUNCTION fn_set_updated_at();
```

IS DISTINCT FROM в PostgreSQL

В PostgreSQL для сравнения значений, которые могут быть `NULL`, используйте `IS DISTINCT FROM` вместо `<>`. Оператор `<>` вернёт `NULL` при сравнении с `NULL`, тогда как `IS DISTINCT FROM` корректно обрабатывает `NULL`.

3.5 Триггеры в MS SQL Server

В MS SQL Server триггеры создаются непосредственно как объекты таблицы. Вместо `OLD` / `NEW` используются виртуальные таблицы `INSERTED` (новые/изменённые строки) и `DELETED` (удалённые/старые строки).

Событие	`INSERTED`	`DELETED`
`INSERT`	вставленные строки	пуста
`UPDATE`	новые значения строк	старые значения строк
`DELETE`	пуста	удалённые строки

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

MS SQL Server — AFTER UPDATE: логирование изменений

```
CREATE OR ALTER TRIGGER trg_orders_audit
ON orders
AFTER UPDATE
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO orders_audit_log (order_id, old_status, new_status, changed_at)
    SELECT
        d.order_id,
        d.status    AS old_status,
        i.status    AS new_status,
        SYSDATETIME()
    FROM DELETED d
    JOIN INSERTED i ON d.order_id = i.order_id
    WHERE d.status <> i.status;
END;
GO
```

MS SQL Server — INSTEAD OF INSERT: перехват вставки

```
-- Триггер INSTEAD OF полностью заменяет стандартную операцию
CREATE OR ALTER TRIGGER trg_clients_insert_log
ON clients
INSTEAD OF INSERT
AS
BEGIN
    SET NOCOUNT ON;
    -- Нормально вставляем строки
    INSERT INTO clients (full_name, email, created_at, is_active)
    SELECT full_name, email, SYSDATETIME(), 1
    FROM INSERTED;

    -- Дополнительно: записываем в журнал
    INSERT INTO insert_log (table_name, record_count, logged_at)
    VALUES (N'clients', @@ROWCOUNT, SYSDATETIME());
END;
GO
```

3.6 Рекомендации по использованию триггеров

Триггеры — мощный инструмент, но злоупотребление ими приводит к проблемам:

- **Сложность отладки** — триггер срабатывает неявно; поведение системы может быть неочевидным.
- **Производительность** — каждая строка, затронутая `INSERT` / `UPDATE` / `DELETE`, активирует триггер; при массовых операциях это существенная нагрузка.

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

- **Цепочки триггеров** — триггер может вызвать изменение другой таблицы, которая имеет свой триггер, и т.д.

Рекомендуется использовать триггеры для: аудита изменений, автоматического заполнения служебных полей (`updated_at`, `created_at`) и простых проверок целостности. Сложную бизнес-логику лучше выносить в хранимые процедуры или код приложения.

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Этап 3. Безопасность — роли СУБД и хэширование паролей

1. Роли и привилегии СУБД

1.1 Системные роли СУБД и бизнес-роли приложения

В информационных системах существуют два принципиально разных уровня ролей, которые нельзя путать.

Параметр	Роли СУБД	Бизнес-роли в таблице приложения
Где хранятся	На сервере СУБД (системные объекты)	В таблице БД (например, `roles`)
Кто проверяет	СУБД при подключении	Код приложения при выполнении операций
Управляет	DBA (администратор БД)	Администратор приложения
Примеры	`app_readonly`, `app_support`, `app_admin`	`client`, `support`, `manager`
Гранулярность	На уровне объектов БД (таблица, схема)	На уровне функциональности приложения

Роли СУБД ограничивают то, что приложение (как подключение к БД) может делать на уровне сервера. Например, подключение с ролью `app_readonly` физически не сможет выполнить `INSERT`, даже если в коде приложения ошибка.

Бизнес-роли в таблице приложения определяют, что видит и может делать конкретный пользователь системы. Например, пользователь с ролью `client` не видит кнопку «Назначить исполнителя» в интерфейсе.

1.2 Принцип минимальных привилегий

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Принцип минимальных привилегий (Principle of Least Privilege, PoLP) гласит: каждый субъект (пользователь, процесс, приложение) должен иметь только те права, которые необходимы для выполнения своих функций — не больше.

Практическое применение:

- Приложение для работы с клиентами не должно иметь право `DROP TABLE`.
- Публичный API, доступный из интернета, должен работать через учётную запись только с `SELECT` на нужные таблицы.
- Администратор БД должен использовать привилегированную учётную запись только при необходимости, а для повседневной работы — обычную.

1.3 Роли и привилегии в MySQL

MySQL 8.0 ввёл поддержку ролей. В более ранних версиях роли эмулировались через несколько пользователей с одинаковыми правами.

MySQL — создание ролей и пользователей

```
-- Создание ролей (MySQL 8+)
CREATE ROLE IF NOT EXISTS 'role_readonly';
CREATE ROLE IF NOT EXISTS 'role_editor';
CREATE ROLE IF NOT EXISTS 'role_admin';

-- Назначение привилегий ролям
GRANT SELECT ON mydb.* TO 'role_readonly';

GRANT SELECT, INSERT, UPDATE ON mydb.* TO 'role_editor';

GRANT ALL PRIVILEGES ON mydb.* TO 'role_admin';

-- Создание пользователей
CREATE USER 'user_viewer'@'localhost' IDENTIFIED BY 'ViewPass!99';
CREATE USER 'user_editor'@'localhost' IDENTIFIED BY 'EditPass!99';
CREATE USER 'user_dba'@'localhost'    IDENTIFIED BY 'DbaPass!99';

-- Назначение ролей пользователям
GRANT 'role_readonly' TO 'user_viewer'@'localhost';
GRANT 'role_editor'   TO 'user_editor'@'localhost';
GRANT 'role_admin'    TO 'user_dba'@'localhost';

-- Установить роль активной по умолчанию
SET DEFAULT ROLE ALL TO 'user_viewer'@'localhost';
SET DEFAULT ROLE ALL TO 'user_editor'@'localhost';
SET DEFAULT ROLE ALL TO 'user_dba'@'localhost';

FLUSH PRIVILEGES;
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

MySQL — просмотр привилегий

```
-- Привилегии конкретного пользователя
SHOW GRANTS FOR 'user_viewer'@'localhost';

-- Роли конкретного пользователя
SHOW GRANTS FOR 'user_editor'@'localhost' USING 'role_editor';

-- Текущий пользователь
SHOW GRANTS;
```

MySQL — отзыв привилегий и удаление

```
-- Отозвать конкретную привилегию
REVOKE INSERT ON mydb.orders FROM 'user_editor'@'localhost';

-- Отозвать роль
REVOKE 'role_readonly' FROM 'user_viewer'@'localhost';

-- Удалить пользователя
DROP USER IF EXISTS 'user_viewer'@'localhost';
```

1.4 Роли и привилегии в PostgreSQL

В PostgreSQL понятия «пользователь» и «роль» объединены: `CREATE USER` — это синоним `CREATE ROLE ... WITH LOGIN`.

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

PostgreSQL — роли и привилегии

```
-- Создание ролей без возможности входа (группы)
CREATE ROLE role_readonly;
CREATE ROLE role_editor;
CREATE ROLE role_admin;

-- Привилегии на схему (право использовать объекты схемы)
GRANT USAGE ON SCHEMA public TO role_readonly;
GRANT USAGE ON SCHEMA public TO role_editor;

-- Привилегии на таблицы
GRANT SELECT ON ALL TABLES IN SCHEMA public TO role_readonly;
GRANT SELECT, INSERT, UPDATE ON ALL TABLES IN SCHEMA public TO role_editor;
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO role_admin;
GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA public TO role_editor;

-- Создание пользователей
CREATE USER user_viewer WITH PASSWORD 'ViewPass!99';
CREATE USER user_editor WITH PASSWORD 'EditPass!99';
CREATE USER user_dba    WITH PASSWORD 'DbaPass!99';

-- Назначение ролей
GRANT role_readonly TO user_viewer;
GRANT role_editor   TO user_editor;
GRANT role_admin    TO user_dba;
```

1.5 Роли и привилегии в MS SQL Server

В MS SQL Server модель безопасности двухуровневая: `LOGIN` (уровень сервера) → `USER` (уровень базы данных).

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

MS SQL Server — создание логина, пользователя, роли

```
-- Уровень сервера: создание логина
CREATE LOGIN login_viewer WITH PASSWORD = 'ViewPass!99';
CREATE LOGIN login_editor WITH PASSWORD = 'EditPass!99';
CREATE LOGIN login_dba    WITH PASSWORD = 'DbaPass!99';

-- Уровень БД: создание пользователей, привязанных к логинам
USE mydb;
GO

CREATE USER user_viewer FOR LOGIN login_viewer;
CREATE USER user_editor FOR LOGIN login_editor;
CREATE USER user_dba    FOR LOGIN login_dba;

-- Создание ролей БД
CREATE ROLE role_readonly;
CREATE ROLE role_editor;
CREATE ROLE role_admin;

-- Назначение привилегий ролям
GRANT SELECT ON SCHEMA::dbo TO role_readonly;
GRANT SELECT, INSERT, UPDATE ON SCHEMA::dbo TO role_editor;
GRANT CONTROL ON SCHEMA::dbo TO role_admin;

-- Включение пользователей в роли
ALTER ROLE role_readonly ADD MEMBER user_viewer;
ALTER ROLE role_editor   ADD MEMBER user_editor;
ALTER ROLE role_admin    ADD MEMBER user_dba;
```

sp_addrolemember (устаревший способ)

В старом коде можно встретить `EXEC sp_addrolemember 'role_readonly', 'user_viewer'`. Начиная с SQL Server 2012 рекомендуется использовать `ALTER ROLE ... ADD MEMBER`.

---

2. Хэширование паролей

2.1 Почему нельзя хранить пароль в открытом виде

Хранение паролей в открытом виде (plaintext) — критическая уязвимость. При компрометации базы данных злоумышленник получает мгновенный доступ ко всем аккаунтам, а пользователи, использующие один пароль на нескольких сайтах, подвергают риску все свои учётные записи.

Согласно нормам информационной безопасности (OWASP, NIST SP 800-63B), пароли **никогда** не должны храниться в открытом виде. Минимально допустимый способ — одностороннее хэширование с солью.

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

2.2 Шифрование vs хэширование vs соль

Операция	Описание	Обратимость	Применение для паролей
Шифрование	Преобразование данных с ключом	Обратимо (есть ключ)	Нет — при компрометации ключа все пароли раскрыты
Хэширование	Одностороннее преобразование в строку фиксированной длины	Необратимо	Да, но без соли уязвимо к rainbow-таблицам
Хэширование + соль	Хэш вычисляется от `salt + password`	Необратимо	Да — рекомендуемый минимум
Адаптивное хэширование (bcrypt, Argon2)	Специализированные алгоритмы с настраиваемой стоимостью	Необратимо	Да — наилучший вариант

Соль (salt) — случайная строка, уникальная для каждого пользователя, добавляемая к паролю перед хэшированием. Соль хранится в базе данных рядом с хэшем (она не секретна). Смысл соли:

- два пользователя с одинаковым паролем будут иметь разные хэши;
- rainbow-таблицы (предвычисленные хэши) становятся бесполезны.

2.3 MD5 и SHA-1 — устаревшие алгоритмы

MD5 и **SHA-1** — алгоритмы, созданные не для хранения паролей, а для проверки целостности файлов. Они:

- очень быстрые (злоумышленник может перебирать миллиарды вариантов в секунду на GPU);
- имеют известные коллизии (MD5);
- давно скомпрометированы для использования в качестве хранения паролей.

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Не используйте MD5/SHA-1 для паролей

Использование `MD5(password)` или `SHA1(password)` для хранения паролей является грубой ошибкой безопасности. Базы данных с MD5-хэшами паролей взламываются за минуты с помощью онлайн-сервисов.

2.4 SHA-256 + соль — базовый уровень

SHA-256 — значительно более стойкий алгоритм, чем MD5/SHA-1. В паре с уникальной солью обеспечивает приемлемый уровень защиты для учебных и небольших производственных систем.

Схема хранения:

- столбец `password_salt VARCHAR(32)` — случайная соль (hex-строка)
- столбец `password_hash VARCHAR(64)` — SHA-256 хэш от `(salt + password)`, 64 hex-символа

MySQL — SHA-256 хранение

```
-- Вычисление хэша SHA-256 средствами MySQL
SELECT SHA2('mysaltpassword123', 256);
-- Результат: 64-символьная hex-строка

-- При регистрации:
-- соль генерируется в коде приложения (C#), передаётся в процедуру
INSERT INTO users (login, password_salt, password_hash, full_name, email)
VALUES ('ivanov', 'a3f1...', SHA2(CONCAT('a3f1...', 'mypassword'), 256),
'Иванов', 'iv@mail.ru');

-- Проверка пароля при входе:
SELECT user_id, full_name
FROM users
WHERE login = 'ivanov'
  AND password_hash = SHA2(CONCAT(password_salt, 'mypassword'), 256)
  AND is_active = 1;
```

2.5 bcrypt — рекомендуемый алгоритм

bcrypt — адаптивная хэш-функция, специально разработанная для хранения паролей (Niels Provos, David Mazières, 1999). Её ключевые особенности:

- **Встроенная соль** — bcrypt автоматически генерирует и включает соль в хэш; хранить соль отдельно не нужно.

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер disynergy.ru

- **Настраиваемый cost factor** — параметр «стоимости» (work factor, rounds) определяет, сколько вычислений требует хэширование. При росте производительности железа cost factor можно увеличить.
- **Фиксированная длина вывода** — результат всегда 60 символов, включает алгоритм, cost factor и соль.

Хэш bcrypt выглядит так:
`$2b$12$LQv3c1yqBWVHxkd0LQ1Cr.eDmqb0KOBfFMNfMz6gFJjpQGhJM9Hs`

- `$2b$` — версия алгоритма
- `12` — cost factor (2^12 = 4096 итераций)
- Следующие 22 символа — соль
- Оставшиеся 31 символ — хэш

PostgreSQL — bcrypt через расширение pgcrypto

```
-- Установка расширения
CREATE EXTENSION IF NOT EXISTS pgcrypto;

-- Хэширование пароля (cost = 12)
SELECT crypt('mypassword', gen_salt('bf', 12));
-- Результат: строка ~60 символов

-- Хранение при регистрации:
INSERT INTO users (login, password_hash, full_name, email)
VALUES ('ivanov', crypt('mypassword', gen_salt('bf', 12)), 'Иванов',
'iv@mail.ru');

-- Проверка при входе (хэш сравнивается сам с собой):
SELECT user_id, full_name
FROM users
WHERE login = 'ivanov'
  AND password_hash = crypt('mypassword', password_hash);
```

В MySQL встроенной поддержки bcrypt нет. Рекомендуется выполнять хэширование bcrypt в коде приложения (C#) и хранить готовый хэш в базе данных.

2.6 Хэширование паролей в C

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер disynergy.ru

C# — SHA-256 + соль

```
using System;
using System.Security.Cryptography;
using System.Text;

public static class PasswordHelper
{
    /// <summary>
    /// Генерирует случайную соль: 16 байт → hex-строка (32 символа).
    /// </summary>
    public static string GenerateSalt()
    {
        byte[] saltBytes = new byte[16];
        using (var rng = RandomNumberGenerator.Create())
            rng.GetBytes(saltBytes);
        return BitConverter.ToString(saltBytes).Replace("-", "").ToLower();
    }

    /// <summary>
    /// Вычисляет SHA-256 хэш от строки (соль + пароль).
    /// Возвращает hex-строку (64 символа).
    /// </summary>
    public static string HashPassword(string password, string salt)
    {
        using var sha = SHA256.Create();
        byte[] bytes = sha.ComputeHash(Encoding.UTF8.GetBytes(salt +
password));
        return BitConverter.ToString(bytes).Replace("-", "").ToLower();
    }

    /// <summary>
    /// Проверяет пароль: вычисляет хэш и сравнивает со сохранённым.
    /// </summary>
    public static bool VerifyPassword(string password, string salt, string
storedHash)
    {
        string computed = HashPassword(password, salt);
        return computed.Equals(storedHash, StringComparison.OrdinalIgnoreCase);
    }
}
```

Таблица `users` должна содержать поля:

```
password_salt VARCHAR(32)  NOT NULL,
password_hash VARCHAR(64)  NOT NULL,
```

2.7 Регистрация и вход в C# WinForms

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

C# — регистрация пользователя

```
private void RegisterUser(string login, string password, string fullName,
string email)
{
    string salt = PasswordHelper.GenerateSalt();
    string hash = PasswordHelper.HashPassword(password, salt);

    using var conn = DBConnection.GetConnection();
    conn.Open();
    using var cmd = new MySqlCommand("sp_register_user", conn)
    {
        CommandType = System.Data.CommandType.StoredProcedure
    };
    cmd.Parameters.AddWithValue("p_login",    login);
    cmd.Parameters.AddWithValue("p_salt",     salt);
    cmd.Parameters.AddWithValue("p_hash",     hash);
    cmd.Parameters.AddWithValue("p_fullname", fullName);
    cmd.Parameters.AddWithValue("p_email",    email);
    cmd.ExecuteNonQuery();
}
```

C# — проверка пароля при входе

```
private bool TryLogin(string login, string inputPassword, out int userId)
{
    userId = 0;
    using var conn = DBConnection.GetConnection();
    conn.Open();

    // 1. Получить соль и хэш пользователя по логину
    const string query = @"
        SELECT user_id, password_salt, password_hash
        FROM users
        WHERE login = @login AND is_active = 1";

    using var cmd = new MySqlCommand(query, conn);
    cmd.Parameters.AddWithValue("@login", login);
    using var reader = cmd.ExecuteReader();

    if (!reader.Read()) return false;

    userId             = reader.GetInt32("user_id");
    string storedSalt  = reader.GetString("password_salt");
    string storedHash  = reader.GetString("password_hash");
    reader.Close();

    // 2. Проверить пароль
    return PasswordHelper.VerifyPassword(inputPassword, storedSalt,
storedHash);
}
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения

Нажмите на мессенджер, чтобы написать

Telegram ↗ WhatsApp ↗ MAX ↗

+7 (924) 305-23-08 · Пишите в мессенджер

disynergy.ru

2.8 Сравнительная таблица алгоритмов

Алгоритм	Скорость	Стойкость к перебору	Рекомендуется для паролей
MD5	Очень быстрый	Крайне низкая	Нет
SHA-1	Очень быстрый	Низкая	Нет
SHA-256	Быстрый	Средняя (с солью)	С осторожностью
SHA-256 + соль	Быстрый	Удовлетворительная	Для учебных проектов
bcrypt (cost 12)	Медленный (намеренно)	Высокая	Да
Argon2id	Медленный, настраиваемый	Очень высокая	Да (лучший выбор)

Для учебного проекта

В рамках производственной практики реализация SHA-256 + соль на C# является достаточной и демонстрирует понимание принципов безопасного хранения паролей. Для реального производственного приложения рекомендуется использовать библиотеку BCrypt.Net-Next.

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Этап 4. CRUD и запросы с JOIN

1. CRUD-операции

CRUD — аббревиатура четырёх базовых операций с данными: **C**reate, **R**ead, **U**pdate, **D**elete. В SQL эти операции реализуются командами `INSERT`, `SELECT`, `UPDATE`, `DELETE` соответственно.

1.1 INSERT — вставка данных

Базовый синтаксис:

```
INSERT INTO table_name (column1, column2, column3)
VALUES (value1, value2, value3);
```

Вставка нескольких строк одним запросом (MySQL, PostgreSQL):

```
INSERT INTO categories (name, description)
VALUES
    ('Сеть',        'Проблемы с сетью'),
    ('Оборудование','Неисправности техники'),
    ('Программы',   'Ошибки ПО');
```

В MS SQL Server синтаксис аналогичен с MySQL 10 начиная с версии 2008:

```
INSERT INTO categories (name, description)
VALUES
    (N'Сеть',         N'Проблемы с сетью'),
    (N'Оборудование', N'Неисправности техники');
```

Получение ID вставленной строки:

СУБД	Способ	Пример
MySQL	`LAST_INSERT_ID()`	`SELECT LAST_INSERT_ID();`
PostgreSQL	`RETURNING` в INSERT	`INSERT INTO ... RETURNING id;`
MS SQL Server	`SCOPE_IDENTITY()` или `OUTPUT`	`SELECT SCOPE_IDENTITY();`

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

MySQL — вставка с получением ID

```
INSERT INTO tickets (title, description, status, category_id, created_by,
created_at)
VALUES ('Нет интернета', 'Не работает wi-fi', 'new', 1, 3, NOW());

SELECT LAST_INSERT_ID() AS new_ticket_id;
```

PostgreSQL — RETURNING

```
INSERT INTO tickets (title, description, status, category_id, created_by)
VALUES ('Нет интернета', 'Не работает wi-fi', 'new', 1, 3)
RETURNING ticket_id;
```

MS SQL Server — SCOPE_IDENTITY

```
INSERT INTO tickets (title, description, status, category_id, created_by)
VALUES (N'Нет интернета', N'Не работает wi-fi', N'new', 1, 3);

SELECT SCOPE_IDENTITY() AS new_ticket_id;

-- Или через OUTPUT:
INSERT INTO tickets (title, description, status)
OUTPUT INSERTED.ticket_id
VALUES (N'Нет интернета', N'Не работает wi-fi', N'new');
```

1.2 SELECT — выборка данных

Фильтрация с WHERE:

```
-- Все открытые заявки с высоким приоритетом
SELECT ticket_id, title, status, priority, created_at
FROM tickets
WHERE status = 'new'
  AND priority = 'high';
```

Сортировка ORDER BY:

```
SELECT ticket_id, title, status, created_at
FROM tickets
ORDER BY created_at DESC;  -- сначала новые

-- Сортировка по нескольким полям
ORDER BY priority ASC, created_at DESC;
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Ограничение количества строк:

СУБД	Синтаксис
MySQL / PostgreSQL	`LIMIT n` / `LIMIT n OFFSET m`
MS SQL Server	`TOP n` / `OFFSET m ROWS FETCH NEXT n ROWS ONLY`

```sql
-- MySQL / PostgreSQL: первые 10 строк, начиная с 21-й
SELECT * FROM tickets
ORDER BY created_at DESC
LIMIT 10 OFFSET 20;
```

```sql
-- MS SQL Server: аналог
SELECT * FROM tickets
ORDER BY created_at DESC
OFFSET 20 ROWS FETCH NEXT 10 ROWS ONLY;
```

Подзапрос в WHERE:

```sql
-- Заявки, назначенные специалистам из отдела ИТ
SELECT t.ticket_id, t.title, t.status
FROM tickets t
WHERE t.assigned_to IN (
    SELECT u.user_id
    FROM users u
    JOIN departments d ON u.department_id = d.department_id
    WHERE d.name = 'ИТ-отдел'
);
```

1.3 UPDATE — обновление данных

Базовый синтаксис:

```sql
UPDATE table_name
SET column1 = value1,
    column2 = value2
WHERE condition;
```

> ⚠ **Всегда используйте WHERE в UPDATE**
>
> Команда `UPDATE` без `WHERE` обновит **все** строки таблицы. Это одна из наиболее распространённых ошибок при работе с БД.

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Обновление через JOIN (MySQL):

```
-- Переназначить все заявки уволенного сотрудника (user_id=5)
-- другому сотруднику (user_id=8)
UPDATE tickets t
JOIN users u ON t.assigned_to = u.user_id
SET t.assigned_to = 8,
    t.updated_at  = NOW()
WHERE u.user_id = 5
  AND t.status NOT IN ('closed', 'resolved');
```

Обновление через JOIN (PostgreSQL / MS SQL):

```
-- PostgreSQL: UPDATE c FROM
UPDATE tickets t
SET    assigned_to = 8,
       updated_at  = NOW()
FROM   users u
WHERE  t.assigned_to = u.user_id
  AND  u.user_id = 5
  AND  t.status NOT IN ('closed', 'resolved');
```

1.4 DELETE — удаление данных

Базовый синтаксис:

```
DELETE FROM table_name
WHERE condition;
```

Каскадное удаление: если таблица `comments` имеет внешний ключ на `tickets` с `ON DELETE CASCADE`, то при удалении заявки все её комментарии удаляются автоматически.

```
-- При наличии ON DELETE CASCADE на comments и ticket_history:
DELETE FROM tickets WHERE ticket_id = 42;
-- Автоматически удалятся: comments и ticket_history для этой заявки
```

Поведение ON DELETE:

Действие	Поведение	Когда использовать
`CASCADE`	Дочерние записи удаляются вместе	Когда дочерние записи не имеют смысла без родителя (комментарии к заявке)

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Действие	Поведение	Когда использовать
`RESTRICT`	Удаление запрещено, если есть дочерние	Когда удаление опасно (заявки к пользователю)
`SET NULL`	FK в дочерних ставится в NULL	Когда связь необязательна (заявка без исполнителя)

---

2. JOIN — объединение таблиц

Оператор `JOIN` позволяет комбинировать строки из двух и более таблиц на основе условия связи. Без `JOIN` реляционная БД теряет основное преимущество — способность хранить данные без дублирования и связывать их при запросах.

2.1 INNER JOIN

`INNER JOIN` возвращает только те строки, для которых есть совпадение в **обеих** таблицах.

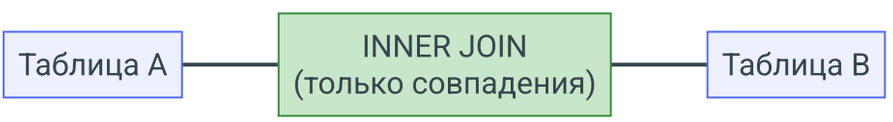

Рисунок 1 — INNER JOIN: строки только при наличии совпадения в обеих таблицах

```sql
-- Заявки с именем автора (только заявки, у которых есть пользователь-автор)
SELECT
    t.ticket_id,
    t.title,
    t.status,
    u.full_name AS author
FROM tickets t
INNER JOIN users u ON t.created_by = u.user_id;
```

Ключевое слово `INNER` необязательно: `JOIN` без модификатора — это `INNER JOIN`.

2.2 LEFT JOIN

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

`LEFT JOIN` (LEFT OUTER JOIN) возвращает **все** строки из левой таблицы, и соответствующие строки из правой. Если соответствия нет — правая часть заполняется `NULL`.

```
-- Все пользователи, даже если у них нет заявок
SELECT
    u.user_id,
    u.full_name,
    COUNT(t.ticket_id) AS ticket_count
FROM users u
LEFT JOIN tickets t ON u.user_id = t.created_by
GROUP BY u.user_id, u.full_name;
```

LEFT JOIN для поиска «записей без пары»:

```
-- Найти заявки без назначенного исполнителя
SELECT t.ticket_id, t.title, t.status
FROM tickets t
LEFT JOIN users u ON t.assigned_to = u.user_id
WHERE u.user_id IS NULL;
```

Паттерн `LEFT JOIN ... WHERE right_table.id IS NULL` — стандартный способ найти записи в левой таблице, для которых нет соответствия в правой.

2.3 RIGHT JOIN

`RIGHT JOIN` (RIGHT OUTER JOIN) — зеркальный к `LEFT JOIN`: возвращает **все** строки из правой таблицы и соответствующие строки из левой. На практике `RIGHT JOIN` можно заменить `LEFT JOIN` с переставленными таблицами; используется реже.

```
-- Все категории, включая те, у которых нет заявок
SELECT
    c.name         AS category,
    COUNT(t.ticket_id) AS ticket_count
FROM tickets t
RIGHT JOIN categories c ON t.category_id = c.category_id
GROUP BY c.category_id, c.name;
```

2.4 FULL OUTER JOIN

`FULL OUTER JOIN` возвращает все строки из обеих таблиц. При отсутствии совпадения с одной стороны — `NULL`.

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер disynergy.ru

FULL OUTER JOIN в MySQL

MySQL не поддерживает `FULL OUTER JOIN` напрямую. Его эмулируют через `UNION` двух запросов: `LEFT JOIN UNION RIGHT JOIN`.

```
-- MySQL: эмуляция FULL OUTER JOIN через UNION
SELECT u.user_id, u.full_name, t.ticket_id, t.title
FROM users u
LEFT JOIN tickets t ON u.user_id = t.created_by

UNION

SELECT u.user_id, u.full_name, t.ticket_id, t.title
FROM users u
RIGHT JOIN tickets t ON u.user_id = t.created_by;
```

```
-- PostgreSQL / MS SQL Server: прямой синтаксис
SELECT u.user_id, u.full_name, t.ticket_id, t.title
FROM users u
FULL OUTER JOIN tickets t ON u.user_id = t.created_by;
```

2.5 Self JOIN

Self JOIN — таблица объединяется сама с собой. Используется, когда таблица хранит иерархические или рекурсивные связи (например, сотрудник → руководитель, категория → родительская категория).

```
-- Сотрудники с именем их руководителя
-- (employees.manager_id → employees.employee_id)
SELECT
    e.employee_id,
    e.full_name      AS employee,
    m.full_name      AS manager
FROM employees e
LEFT JOIN employees m ON e.manager_id = m.employee_id;
```

2.6 JOIN трёх и более таблиц

Практические запросы часто объединяют три и более таблиц. Каждый следующий `JOIN` добавляет связь.

```
-- Полный вид заявки: заявка + автор + отдел автора + категория + исполнитель
SELECT
    t.ticket_id,
    t.title,
    t.status,
    t.priority,
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

```
    c.name          AS category,
    u1.full_name    AS author,
    d.name          AS author_dept,
    u2.full_name    AS assignee
FROM tickets t
JOIN  categories c   ON t.category_id   = c.category_id
JOIN  users u1       ON t.created_by    = u1.user_id
LEFT JOIN departments d  ON u1.department_id = d.department_id
LEFT JOIN users u2   ON t.assigned_to   = u2.user_id
ORDER BY t.created_at DESC;
```

Здесь используется `INNER JOIN` для обязательных связей (заявка обязательно имеет автора и категорию) и `LEFT JOIN` для необязательных (заявка может не иметь исполнителя или автор может не принадлежать отделу).

2.7 Агрегат с GROUP BY и HAVING

`GROUP BY` группирует строки по значению столбца, `HAVING` фильтрует уже сгруппированные данные (аналог `WHERE`, но для агрегатов).

```
-- Количество заявок по статусам, показать только статусы с > 5 заявками
SELECT
    status,
    COUNT(*) AS cnt
FROM tickets
GROUP BY status
HAVING COUNT(*) > 5
ORDER BY cnt DESC;
```

```
-- Нагрузка по специалистам: только те, у кого > 3 активных заявок
SELECT
    u.full_name,
    COUNT(t.ticket_id) AS active_count
FROM users u
JOIN tickets t ON u.user_id = t.assigned_to
WHERE t.status NOT IN ('closed', 'resolved')
GROUP BY u.user_id, u.full_name
HAVING COUNT(t.ticket_id) > 3
ORDER BY active_count DESC;
```

2.8 Подзапрос в WHERE и EXISTS

Подзапрос (subquery) — запрос, вложенный в другой запрос. Может использоваться в `WHERE`, `FROM`, `SELECT`.

```
-- Пользователи, у которых нет ни одной заявки (NOT IN)
SELECT user_id, full_name
FROM users
WHERE user_id NOT IN (
    SELECT DISTINCT created_by FROM tickets
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

```
);

-- Аналог через NOT EXISTS (часто эффективнее)
SELECT u.user_id, u.full_name
FROM users u
WHERE NOT EXISTS (
    SELECT 1 FROM tickets t WHERE t.created_by = u.user_id
);
```

2.9 Различия между СУБД

Особенность	MySQL	PostgreSQL	MS SQL Server
`FULL OUTER JOIN`	Через UNION	Поддерживается	Поддерживается
`LIMIT n`	`LIMIT n`	`LIMIT n`	`TOP n` или `FETCH NEXT n ROWS ONLY`
Псевдоним таблицы в UPDATE/DELETE	Поддерживается	Поддерживается (через FROM)	Поддерживается
`RETURNING` после DML	Нет	Поддерживается	Через `OUTPUT`
Последний вставленный ID	`LAST_INSERT_ID()`	`RETURNING id` или `lastval()`	`SCOPE_IDENTITY()`

---

3. Итоги

Операции CRUD и JOIN являются основой работы с реляционными базами данных. Понимание различных видов JOIN позволяет правильно формулировать запросы:

- `INNER JOIN` — когда нужны только записи с совпадением в обеих таблицах;
- `LEFT JOIN` — когда нужны все записи из основной таблицы, включая те, у которых нет связанных записей;
- `RIGHT JOIN` — когда нужны все записи из присоединяемой таблицы;
- `FULL OUTER JOIN` — когда нужны все записи из обеих таблиц.

Правильное использование `WHERE`, `GROUP BY`, `HAVING`, `ORDER BY` и подзапросов позволяет получать точные выборки, необходимые для работы информационной

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер disynergy.ru

системы.

Помогу с учебным заданием без предоплаты. Оплата после выполнения

Нажмите на мессенджер, чтобы написать

Telegram ↗ WhatsApp ↗ MAX ↗

+7 (924) 305-23-08 · Пишите в мессенджер

disynergy.ru

Этап 1. Проектирование базы данных

Задание

На данном этапе необходимо:

1. Выбрать предметную область из утверждённого списка (см. раздел «Предметные области»).
2. Обосновать выбор и описать предметную область (3–5 предложений).
3. Построить ER-диаграмму (не менее 5 таблиц).
4. Описать каждую таблицу: поля, типы данных, ключи.
5. Описать связи между таблицами.
6. Написать и выполнить DDL-скрипт `CREATE TABLE` для выбранной СУБД.
7. Внести тестовые данные (`INSERT`) — не менее 2–3 записей в каждую таблицу.
8. Сделать скриншоты: ER-диаграмма, список таблиц в СУБД, результаты SELECT из созданных таблиц.

> ⚠ **Важно**
>
> Ниже приведён **пример** выполнения этапа для предметной области «Система управления заявками технической поддержки (Helpdesk)» на стеке **MySQL + C# Windows Forms**. Используйте этот пример как образец оформления. В своём отчёте опишите **собственную** предметную область.

Пример: Helpdesk — система управления заявками

Описание предметной области

Компания «ТехПоддержка Плюс» предоставляет услуги технической поддержки сотрудникам и клиентам. Основной инструмент работы — система Helpdesk: централизованный журнал заявок, позволяющий регистрировать обращения, назначать исполнителей и отслеживать статус каждой заявки.

Пользователи системы делятся на три категории: **клиенты** (создают заявки), **специалисты поддержки** (принимают и решают заявки) и **администраторы** (управляют

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

справочниками, пользователями и настройками).

Обоснование выбора

Предметная область выбрана по следующим причинам:

- Содержит богатую логику (смена статусов, назначение исполнителей, история изменений), при которой триггеры и хранимые процедуры органично встраиваются в архитектуру.
- Включает 7 таблиц с разнообразными типами связей (1:N, N:1, ссылки на себя).
- Предметная область не пересекается с проектом ПМ.03 (библиотека / склад / другая тема).
- Может служить основой для дипломного проекта (ВКР).

ER-диаграмма

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

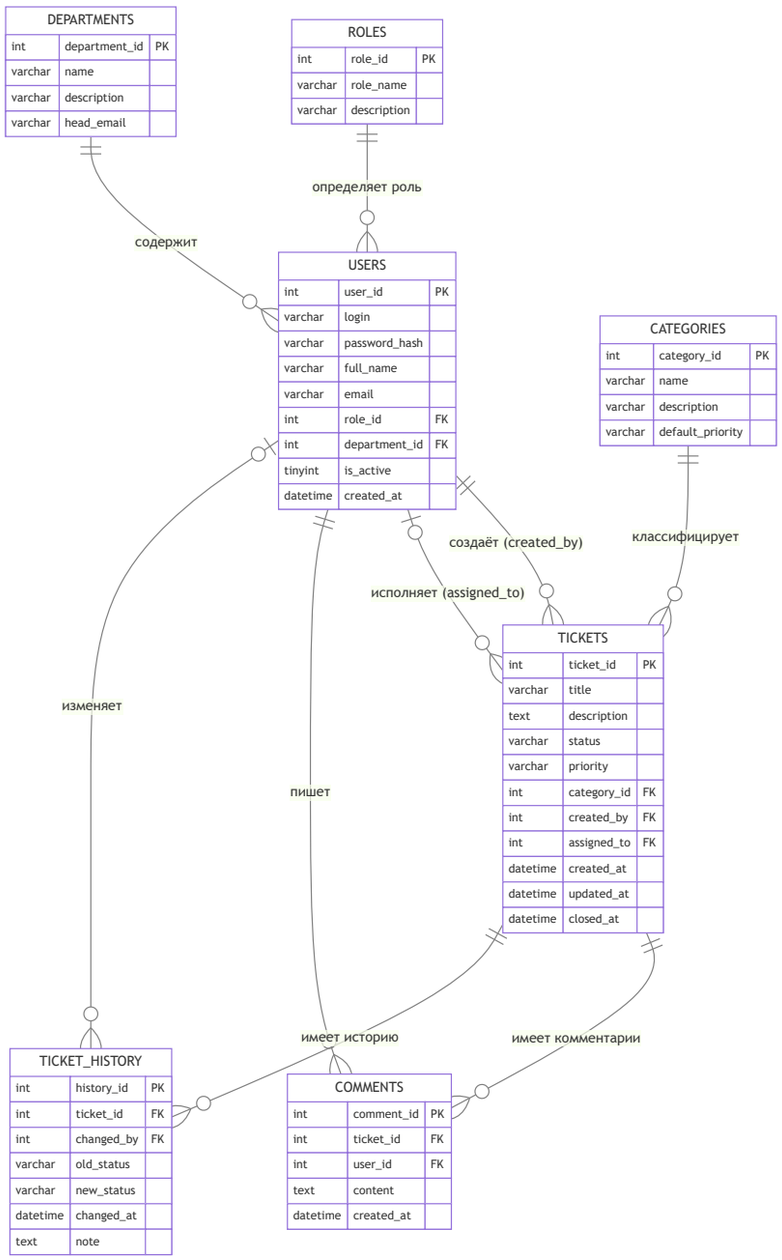

Рисунок 1 — ER-диаграмма системы Helpdesk (IE-нотация, 7 таблиц)

Описание таблиц

Таблица `roles` — Роли пользователей

Помогу с учебным заданием без предоплаты. Оплата после выполнения

Нажмите на мессенджер, чтобы написать

Telegram ↗ WhatsApp ↗ MAX ↗

+7 (924) 305-23-08 · Пишите в мессенджер disynergy.ru

Поле	Тип MySQL	Ключ	Описание
`role_id`	`INT AUTO_INCREMENT`	PK	Идентификатор роли
`role_name`	`VARCHAR(50)`	UNIQUE	Название роли: `admin`, `support`, `client`
`description`	`VARCHAR(255)`	—	Описание роли

Таблица `departments` — Отделы

Поле	Тип MySQL	Ключ	Описание
`department_id`	`INT AUTO_INCREMENT`	PK	Идентификатор отдела
`name`	`VARCHAR(100)`	—	Название отдела
`description`	`VARCHAR(255)`	—	Описание деятельности отдела
`head_email`	`VARCHAR(150)`	—	Email руководителя отдела

Таблица `users` — Пользователи системы

Поле	Тип MySQL	Ключ	Описание
`user_id`	`INT AUTO_INCREMENT`	PK	Идентификатор пользователя
`login`	`VARCHAR(80)`	UNIQUE	Логин для входа в систему
`password_hash`	`VARCHAR(64)`	—	SHA-256 хэш пароля
`full_name`	`VARCHAR(150)`	—	Полное имя

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Поле	Тип MySQL	Ключ	Описание
`email`	`VARCHAR(150)`	UNIQUE	Email
`role_id`	`INT`	FK → roles	Роль пользователя
`department_id`	`INT`	FK → departments	Принадлежность к отделу
`is_active`	`TINYINT(1)`	—	1 — активен, 0 — заблокирован
`created_at`	`DATETIME`	—	Дата и время регистрации

Таблица `categories` — Категории заявок

Поле	Тип MySQL	Ключ	Описание
`category_id`	`INT AUTO_INCREMENT`	PK	Идентификатор категории
`name`	`VARCHAR(100)`	UNIQUE	Название категории
`description`	`VARCHAR(255)`	—	Описание
`default_priority`	`VARCHAR(20)`	—	Приоритет по умолчанию: `low`, `medium`, `high`

Таблица `tickets` — Заявки

Поле	Тип MySQL	Ключ	Описание
`ticket_id`	`INT AUTO_INCREMENT`	PK	Идентификатор заявки
`title`	`VARCHAR(200)`	—	Краткое описание проблемы

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Поле	Тип MySQL	Ключ	Описание
`description`	`TEXT`	—	Подробное описание
`status`	`VARCHAR(30)`	—	Статус: `new`, `in_progress`, `resolved`, `closed`
`priority`	`VARCHAR(20)`	—	Приоритет: `low`, `medium`, `high`, `critical`
`category_id`	`INT`	FK → categories	Категория заявки
`created_by`	`INT`	FK → users	Автор заявки
`assigned_to`	`INT` NULL	FK → users	Назначенный исполнитель
`created_at`	`DATETIME`	—	Время создания
`updated_at`	`DATETIME` NULL	—	Время последнего обновления
`closed_at`	`DATETIME` NULL	—	Время закрытия

Таблица `comments` — Комментарии к заявкам

Поле	Тип MySQL	Ключ	Описание
`comment_id`	`INT AUTO_INCREMENT`	PK	Идентификатор комментария
`ticket_id`	`INT`	FK → tickets (CASCADE)	Заявка
`user_id`	`INT`	FK → users	Автор комментария
`content`	`TEXT`	—	Текст комментария

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Поле	Тип MySQL	Ключ	Описание
`created_at`	`DATETIME`	—	Время публикации

Таблица `ticket_history` — История изменений заявок

Поле	Тип MySQL	Ключ	Описание
`history_id`	`INT AUTO_INCREMENT`	PK	Идентификатор записи
`ticket_id`	`INT`	FK → tickets (CASCADE)	Заявка
`changed_by`	`INT` NULL	FK → users	Кто изменил (NULL — триггер/система)
`old_status`	`VARCHAR(30)`	—	Предыдущий статус
`new_status`	`VARCHAR(30)`	—	Новый статус
`changed_at`	`DATETIME`	—	Время изменения
`note`	`TEXT`	—	Примечание

Связи между таблицами

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Связь	FK (дочерняя таблица)	→ PK (родительская)	Тип	Смысл
roles → users	`users.role_id`	`roles.role_id`	1:N	Пользователь имеет одну роль
departments → users	`users.department_id`	`departments.department_id`	1:N	Пользователь принадлежит отделу
users → tickets (автор)	`tickets.created_by`	`users.user_id`	1:N	Пользователь создаёт заявки
users → tickets (исполнитель)	`tickets.assigned_to`	`users.user_id`	1:N	Специалист назначается на заявки
categories → tickets	`tickets.category_id`	`categories.category_id`	1:N	Заявка принадлежит категории
tickets → comments	`comments.ticket_id`	`tickets.ticket_id`	1:N CASCADE	Заявка имеет комментарии
users → comments	`comments.user_id`	`users.user_id`	1:N	Пользователь пишет комментарии
tickets → ticket_history	`ticket_history.ticket_id`	`tickets.ticket_id`	1:N CASCADE	История принадлежит заявке

DDL-скрипт CREATE TABLE (MySQL)

```
-- ============================================================
-- База данных: helpdesk
-- СУБД: MySQL 8.x, InnoDB, UTF-8
-- ============================================================
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

```sql
CREATE DATABASE IF NOT EXISTS helpdesk
    CHARACTER SET utf8mb4
    COLLATE utf8mb4_unicode_ci;

USE helpdesk;

-- ------------------------------------------------------------
-- Таблица 1: roles
-- ------------------------------------------------------------
CREATE TABLE IF NOT EXISTS roles (
    role_id     INT          NOT NULL AUTO_INCREMENT,
    role_name   VARCHAR(50)  NOT NULL,
    description VARCHAR(255),
    PRIMARY KEY (role_id),
    UNIQUE KEY uq_role_name (role_name)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- ------------------------------------------------------------
-- Таблица 2: departments
-- ------------------------------------------------------------
CREATE TABLE IF NOT EXISTS departments (
    department_id INT          NOT NULL AUTO_INCREMENT,
    name          VARCHAR(100) NOT NULL,
    description   VARCHAR(255),
    head_email    VARCHAR(150),
    PRIMARY KEY (department_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- ------------------------------------------------------------
-- Таблица 3: users
-- ------------------------------------------------------------
CREATE TABLE IF NOT EXISTS users (
    user_id       INT          NOT NULL AUTO_INCREMENT,
    login         VARCHAR(80)  NOT NULL,
    password_hash VARCHAR(64)  NOT NULL,
    full_name     VARCHAR(150) NOT NULL,
    email         VARCHAR(150) NOT NULL,
    role_id       INT          NOT NULL,
    department_id INT,
    is_active     TINYINT(1)   NOT NULL DEFAULT 1,
    created_at    DATETIME     NOT NULL DEFAULT NOW(),
    PRIMARY KEY (user_id),
    UNIQUE KEY uq_users_login (login),
    UNIQUE KEY uq_users_email (email),
    CONSTRAINT fk_users_role
        FOREIGN KEY (role_id) REFERENCES roles (role_id)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    CONSTRAINT fk_users_dept
        FOREIGN KEY (department_id) REFERENCES departments (department_id)
        ON DELETE SET NULL ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- ------------------------------------------------------------
-- Таблица 4: categories
-- ------------------------------------------------------------
CREATE TABLE IF NOT EXISTS categories (
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения

Нажмите на мессенджер, чтобы написать

Telegram ↗ WhatsApp ↗ MAX ↗

+7 (924) 305-23-08 · Пишите в мессенджер disynergy.ru

```
    category_id      INT          NOT NULL AUTO_INCREMENT,
    name             VARCHAR(100) NOT NULL,
    description      VARCHAR(255),
    default_priority VARCHAR(20)  NOT NULL DEFAULT 'medium',
    PRIMARY KEY (category_id),
    UNIQUE KEY uq_category_name (name)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- ------------------------------------------------------------
-- Таблица 5: tickets
-- ------------------------------------------------------------
CREATE TABLE IF NOT EXISTS tickets (
    ticket_id   INT          NOT NULL AUTO_INCREMENT,
    title       VARCHAR(200) NOT NULL,
    description TEXT,
    status      VARCHAR(30)  NOT NULL DEFAULT 'new',
    priority    VARCHAR(20)  NOT NULL DEFAULT 'medium',
    category_id INT          NOT NULL,
    created_by  INT          NOT NULL,
    assigned_to INT,
    created_at  DATETIME     NOT NULL DEFAULT NOW(),
    updated_at  DATETIME,
    closed_at   DATETIME,
    PRIMARY KEY (ticket_id),
    CONSTRAINT fk_tickets_category
        FOREIGN KEY (category_id) REFERENCES categories (category_id)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    CONSTRAINT fk_tickets_creator
        FOREIGN KEY (created_by) REFERENCES users (user_id)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    CONSTRAINT fk_tickets_assignee
        FOREIGN KEY (assigned_to) REFERENCES users (user_id)
        ON DELETE SET NULL ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- ------------------------------------------------------------
-- Таблица 6: comments
-- ------------------------------------------------------------
CREATE TABLE IF NOT EXISTS comments (
    comment_id INT      NOT NULL AUTO_INCREMENT,
    ticket_id  INT      NOT NULL,
    user_id    INT      NOT NULL,
    content    TEXT     NOT NULL,
    created_at DATETIME NOT NULL DEFAULT NOW(),
    PRIMARY KEY (comment_id),
    CONSTRAINT fk_comments_ticket
        FOREIGN KEY (ticket_id) REFERENCES tickets (ticket_id)
        ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT fk_comments_user
        FOREIGN KEY (user_id) REFERENCES users (user_id)
        ON DELETE RESTRICT ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- ------------------------------------------------------------
-- Таблица 7: ticket_history
-- ------------------------------------------------------------
CREATE TABLE IF NOT EXISTS ticket_history (
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения

Нажмите на мессенджер, чтобы написать

Telegram ↗ WhatsApp ↗ MAX ↗

+7 (924) 305-23-08 · Пишите в мессенджер disynergy.ru

```
    history_id INT         NOT NULL AUTO_INCREMENT,
    ticket_id  INT         NOT NULL,
    changed_by INT,
    old_status VARCHAR(30),
    new_status VARCHAR(30) NOT NULL,
    changed_at DATETIME    NOT NULL DEFAULT NOW(),
    note       TEXT,
    PRIMARY KEY (history_id),
    CONSTRAINT fk_history_ticket
        FOREIGN KEY (ticket_id) REFERENCES tickets (ticket_id)
        ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT fk_history_user
        FOREIGN KEY (changed_by) REFERENCES users (user_id)
        ON DELETE SET NULL ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Тестовые данные (INSERT)

```
-- ------------------------------------------------------------
-- Роли
-- ------------------------------------------------------------
INSERT INTO roles (role_name, description) VALUES
    ('admin',   'Администратор системы'),
    ('support', 'Специалист технической поддержки'),
    ('client',  'Пользователь — создаёт заявки');

-- ------------------------------------------------------------
-- Отделы
-- ------------------------------------------------------------
INSERT INTO departments (name, description, head_email) VALUES
    ('ИТ-отдел',          'Отдел информационных технологий',
'it@techplus.ru'),
    ('Отдел продаж',       'Коммерческий отдел компании',
'sales@techplus.ru'),
    ('Бухгалтерия',        'Финансово-бухгалтерский отдел',
'buh@techplus.ru');

-- ------------------------------------------------------------
-- Пользователи (password_hash — SHA2('pass123', 256) для примера)
-- ------------------------------------------------------------
INSERT INTO users (login, password_hash, full_name, email, role_id,
department_id, is_active) VALUES
    ('admin',    SHA2('AdminPass123', 256),   'Соколов Андрей Игоревич',
'admin@techplus.ru',   1, 1, 1),
    ('support1', SHA2('SuppPass123', 256),    'Петрова Марина Сергеевна',
'petrov@techplus.ru',  2, 1, 1),
    ('ivanov',   SHA2('IvPass123', 256),      'Иванов Пётр Николаевич',
'ivanov@techplus.ru',  3, 2, 1),
    ('klimova',  SHA2('KlimPass123', 256),    'Климова Анна Дмитриевна',
'klimova@techplus.ru', 3, 3, 1);

-- ------------------------------------------------------------
-- Категории заявок
-- ------------------------------------------------------------
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

```sql
INSERT INTO categories (name, description, default_priority) VALUES
    ('Сетевые проблемы',      'Нет доступа к сети, интернету, VPN',
'high'),
    ('Программное обеспечение','Ошибки, сбои, установка ПО',
'medium'),
    ('Оборудование',          'Неисправность техники, замена
комплектующих','high');

-- ------------------------------------------------------------
-- Заявки
-- ------------------------------------------------------------
INSERT INTO tickets (title, description, status, priority, category_id,
created_by, assigned_to, created_at) VALUES
    ('Не работает интернет в офисе 205',
     'После перезагрузки роутера интернет не поднялся. Все компьютеры без
сети.',
     'new', 'high', 1, 3, NULL, NOW()),

    ('Ошибка при запуске 1С',
     'При открытии 1С:Предприятие выдаёт ошибку "Платформа не найдена".',
     'in_progress', 'medium', 2, 4, 2, NOW()),

    ('Сломался монитор',
     'Монитор не включается, мигает индикатор питания.',
     'new', 'medium', 3, 3, NULL, NOW()),

    ('Нет доступа к корпоративной почте',
     'Outlook не открывается, выдаёт ошибку авторизации.',
     'resolved', 'high', 1, 4, 2, DATE_SUB(NOW(), INTERVAL 2 DAY));

-- ------------------------------------------------------------
-- Комментарии
-- ------------------------------------------------------------
INSERT INTO comments (ticket_id, user_id, content) VALUES
    (2, 2, 'Начал диагностику. Перепроверяю путь к платформе 1С.'),
    (2, 4, 'Хорошо, жду результата.'),
    (4, 2, 'Проблема решена: сбросил пароль учётной записи Exchange.');

-- ------------------------------------------------------------
-- История изменений
-- ------------------------------------------------------------
INSERT INTO ticket_history (ticket_id, changed_by, old_status, new_status,
note) VALUES
    (2, 2, 'new',         'in_progress', 'Заявка принята в работу'),
    (4, 2, 'new',         'in_progress', 'Начата диагностика'),
    (4, 2, 'in_progress', 'resolved',    'Сброшен пароль Exchange');
```

Вывод по этапу

На данном этапе спроектирована реляционная база данных системы Helpdesk «ТехПоддержка Плюс», включающая 7 таблиц. Схема нормализована до 3НФ: отсутствует дублирование данных, каждый атрибут функционально зависит от первичного ключа своей таблицы.

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Определены связи: `roles → users` (1:N), `departments → users` (1:N), `users → tickets` (1:N, дважды), `categories → tickets` (1:N), `tickets → comments` (1:N CASCADE), `tickets → ticket_history` (1:N CASCADE).

DDL-скрипт успешно выполнен, созданы все 7 таблиц. Внесены тестовые данные: 3 роли, 3 отдела, 4 пользователя, 3 категории, 4 заявки, 3 комментария, 3 записи истории.

Помогу с учебным заданием без предоплаты. Оплата после выполнения

Нажмите на мессенджер, чтобы написать

Telegram ↗ WhatsApp ↗ MAX ↗

+7 (924) 305-23-08 · Пишите в мессенджер

disynergy.ru

Этап 2. Объекты БД — VIEW, хранимые процедуры, триггеры

Задание

На данном этапе необходимо:

1. Создать не менее 2 представлений (VIEW): одно с JOIN, одно с агрегатом.
2. Написать не менее 3 хранимых процедур, автоматизирующих типовые операции.
3. Создать не менее 2 триггеров (например, аудит изменений и авто-timestamp).
4. Продемонстрировать вызов процедур и работу триггеров.
5. Показать, как VIEW и процедуры вызываются из кода приложения (C#).
6. Сделать скриншоты: объекты в MySQL Workbench, результаты SELECT из VIEW, результаты CALL процедур.

> ⚠ **Важно**
>
> Ниже приведён **пример** выполнения этапа для предметной области «Helpdesk» (MySQL + C# Windows Forms). Используйте этот пример как образец. В своём варианте объекты должны соответствовать **вашей** предметной области.

Пример: объекты БД для системы Helpdesk

1. Представления (VIEW)

1.1 VIEW `v_tickets_full` — Полная информация о заявках

Это представление объединяет таблицы `tickets`, `users` (дважды: автор и исполнитель), `categories` и `departments`. Оно используется как основной источник данных для `DataGridView` в форме списка заявок.

```
CREATE OR REPLACE VIEW v_tickets_full AS
SELECT
    t.ticket_id,
    t.title,
    t.status,
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

```
    t.priority,
    c.name          AS category,
    u1.full_name    AS author,
    d.name          AS author_dept,
    u2.full_name    AS assignee,
    t.created_at,
    t.updated_at
FROM tickets t
JOIN  categories c   ON t.category_id   = c.category_id
JOIN  users u1       ON t.created_by    = u1.user_id
JOIN  departments d  ON u1.department_id = d.department_id
LEFT JOIN users u2   ON t.assigned_to   = u2.user_id;
```

Проверка:

```
SELECT * FROM v_tickets_full ORDER BY created_at DESC;
```

1.2 VIEW `v_support_stats` — Нагрузка на специалистов поддержки

Это представление показывает, сколько активных и решённых заявок у каждого специалиста. Используется на дашборде администратора.

```
CREATE OR REPLACE VIEW v_support_stats AS
SELECT
    u.user_id,
    u.full_name,
    COUNT(CASE WHEN t.status NOT IN ('closed', 'resolved') THEN 1 END)
        AS active_tickets,
    COUNT(CASE WHEN t.status = 'resolved' THEN 1 END)
        AS resolved_tickets
FROM users u
LEFT JOIN tickets t ON u.user_id = t.assigned_to
WHERE u.role_id = (SELECT role_id FROM roles WHERE role_name = 'support')
GROUP BY u.user_id, u.full_name;
```

Проверка:

```
SELECT * FROM v_support_stats ORDER BY active_tickets DESC;
```

> **Почему LEFT JOIN в v_support_stats?**
>
> Используется `LEFT JOIN`, а не `INNER JOIN`, чтобы в результат попали специалисты с нулём заявок. Если применить `INNER JOIN`, специалисты без заявок не отобразятся в статистике.

---

2. Хранимые процедуры

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

2.1 `sp_change_ticket_status` — Смена статуса заявки

Процедура обновляет статус заявки и фиксирует изменение в `ticket_history`. Параметры: ID заявки, новый статус, ID пользователя, примечание.

```
DELIMITER $$

CREATE PROCEDURE sp_change_ticket_status(
    IN p_ticket_id  INT,
    IN p_new_status VARCHAR(30),
    IN p_user_id    INT,
    IN p_note       TEXT
)
BEGIN
    DECLARE v_old_status VARCHAR(30);

    -- Получить текущий статус
    SELECT status INTO v_old_status
    FROM tickets
    WHERE ticket_id = p_ticket_id;

    -- Обновить статус заявки
    UPDATE tickets
    SET    status     = p_new_status,
           updated_at = NOW(),
           closed_at  = CASE
                            WHEN p_new_status IN ('closed', 'resolved') THEN
NOW()
                            ELSE closed_at
                        END
    WHERE  ticket_id = p_ticket_id;

    -- Зафиксировать изменение в истории
    INSERT INTO ticket_history (ticket_id, changed_by, old_status, new_status,
changed_at, note)
    VALUES (p_ticket_id, p_user_id, v_old_status, p_new_status, NOW(),
p_note);
END$$

DELIMITER ;
```

Вызов:

```
CALL sp_change_ticket_status(1, 'in_progress', 2, 'Принято в работу');
```

2.2 `sp_assign_ticket` — Назначение исполнителя

Процедура назначает исполнителя на заявку. Перед назначением проверяет, что указанный пользователь имеет роль `support`. При ошибке — генерирует исключение.

```
DELIMITER $$

CREATE PROCEDURE sp_assign_ticket(
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

```
    IN p_ticket_id       INT,
    IN p_support_user_id INT
)
BEGIN
    DECLARE v_role VARCHAR(50);

    -- Получить название роли пользователя
    SELECT r.role_name INTO v_role
    FROM users u
    JOIN roles r ON u.role_id = r.role_id
    WHERE u.user_id = p_support_user_id;

    -- Проверить роль
    IF v_role = 'support' THEN
        UPDATE tickets
        SET assigned_to = p_support_user_id,
            status      = 'in_progress',
            updated_at  = NOW()
        WHERE ticket_id = p_ticket_id;

        INSERT INTO ticket_history (ticket_id, changed_by, old_status,
new_status, note)
        SELECT p_ticket_id, p_support_user_id, status, 'in_progress',
               CONCAT('Назначен исполнитель: user_id=', p_support_user_id)
        FROM tickets WHERE ticket_id = p_ticket_id;
    ELSE
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Пользователь не является специалистом поддержки';
    END IF;
END$$

DELIMITER ;
```

Вызов:

```
CALL sp_assign_ticket(1, 2);
```

2.3 `sp_register_user` — Регистрация нового пользователя

Процедура принимает готовый хэш пароля (хэширование выполняется в C# перед вызовом) и вставляет запись в таблицу `users`.

```
DELIMITER $$

CREATE PROCEDURE sp_register_user(
    IN p_login    VARCHAR(80),
    IN p_hash     VARCHAR(64),
    IN p_fullname VARCHAR(150),
    IN p_email    VARCHAR(150),
    IN p_role_id  INT,
    IN p_dept_id  INT
)
BEGIN
    -- Проверить, не занят ли логин
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер disynergy.ru

```
    IF EXISTS (SELECT 1 FROM users WHERE login = p_login) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Логин уже занят';
    END IF;

    -- Проверить, не занят ли email
    IF EXISTS (SELECT 1 FROM users WHERE email = p_email) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Email уже зарегистрирован';
    END IF;

    INSERT INTO users (login, password_hash, full_name, email, role_id,
department_id, is_active, created_at)
    VALUES (p_login, p_hash, p_fullname, p_email, p_role_id, p_dept_id, 1,
NOW());
END$$

DELIMITER ;
```

Вызов:

```
CALL sp_register_user('novak', SHA2('TempPass!1', 256), 'Новак Дмитрий',
'novak@co.ru', 3, 2);
```

---

3. Триггеры

3.1 `trg_log_status_change` — Аудит смены статуса

Триггер срабатывает **после** обновления строки в `tickets`. Если статус изменился, записывает событие в `ticket_history`.

```
DELIMITER $$

CREATE TRIGGER trg_log_status_change
AFTER UPDATE ON tickets
FOR EACH ROW
BEGIN
    IF OLD.status <> NEW.status THEN
        INSERT INTO ticket_history
            (ticket_id, changed_by, old_status, new_status, changed_at, note)
        VALUES
            (OLD.ticket_id, NULL, OLD.status, NEW.status, NOW(),
             'Автоматическая запись триггера');
    END IF;
END$$

DELIMITER ;
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Триггер vs процедура sp_change_ticket_status

Процедура `sp_change_ticket_status` тоже записывает историю вручную. Если использовать и триггер, и процедуру одновременно, в истории появятся дублирующие записи. В учебном примере они показаны оба для иллюстрации двух механизмов. В реальном проекте выберите один подход.

3.2 `trg_update_timestamp` — Авто-обновление `updated_at`

Триггер срабатывает **перед** обновлением строки в `tickets` и автоматически проставляет текущее время в поле `updated_at`.

```
DELIMITER $$

CREATE TRIGGER trg_update_timestamp
BEFORE UPDATE ON tickets
FOR EACH ROW
BEGIN
    SET NEW.updated_at = NOW();
END$$

DELIMITER ;
```

Благодаря этому триггеру разработчику не нужно вручную передавать `updated_at = NOW()` в каждом `UPDATE`-запросе — СУБД делает это автоматически.

---

4. Работа с VIEW и процедурами из C

4.1 SELECT из VIEW в C# — как обычная таблица

С точки зрения приложения VIEW ничем не отличается от таблицы. Достаточно указать имя VIEW в `SELECT`.

```
using MySql.Data.MySqlClient;
using System.Data;
using System.Windows.Forms;

public partial class TicketsForm : Form
{
    private void LoadTickets()
    {
        using var conn = DBConnection.GetConnection();
        conn.Open();

        const string sql = @"
            SELECT ticket_id, title, status, priority,
                   category, author, author_dept, assignee, created_at
            FROM v_tickets_full
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

```
            ORDER BY created_at DESC";

        using var adapter = new MySqlDataAdapter(sql, conn);
        var table = new DataTable();
        adapter.Fill(table);

        dataGridViewTickets.DataSource = table;

        // Настройка заголовков столбцов
        dataGridViewTickets.Columns["ticket_id"].HeaderText  = "№";
        dataGridViewTickets.Columns["title"].HeaderText      = "Тема";
        dataGridViewTickets.Columns["status"].HeaderText     = "Статус";
        dataGridViewTickets.Columns["priority"].HeaderText   = "Приоритет";
        dataGridViewTickets.Columns["category"].HeaderText   = "Категория";
        dataGridViewTickets.Columns["author"].HeaderText     = "Автор";
        dataGridViewTickets.Columns["author_dept"].HeaderText = "Отдел";
        dataGridViewTickets.Columns["assignee"].HeaderText   = "Исполнитель";
        dataGridViewTickets.Columns["created_at"].HeaderText = "Создана";
    }
}
```

4.2 Вызов хранимой процедуры из C

Для вызова хранимой процедуры используется `MySqlCommand` с `CommandType.StoredProcedure`. Параметры передаются через коллекцию `Parameters`.

```
using MySql.Data.MySqlClient;
using System.Data;

public class TicketService
{
    /// <summary>
    /// Сменить статус заявки через хранимую процедуру
sp_change_ticket_status.
    /// </summary>
    public void ChangeTicketStatus(int ticketId, string newStatus, int userId,
string note)
    {
        using var conn = DBConnection.GetConnection();
        conn.Open();

        using var cmd = new MySqlCommand("sp_change_ticket_status", conn)
        {
            CommandType = CommandType.StoredProcedure
        };

        cmd.Parameters.AddWithValue("p_ticket_id",  ticketId);
        cmd.Parameters.AddWithValue("p_new_status", newStatus);
        cmd.Parameters.AddWithValue("p_user_id",    userId);
        cmd.Parameters.AddWithValue("p_note",       note ?? string.Empty);

        cmd.ExecuteNonQuery();
    }

    /// <summary>
    /// Назначить исполнителя на заявку через sp_assign_ticket.
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер disynergy.ru

```
    /// </summary>
    public void AssignTicket(int ticketId, int supportUserId)
    {
        using var conn = DBConnection.GetConnection();
        conn.Open();

        using var cmd = new MySqlCommand("sp_assign_ticket", conn)
        {
            CommandType = CommandType.StoredProcedure
        };

        cmd.Parameters.AddWithValue("p_ticket_id",       ticketId);
        cmd.Parameters.AddWithValue("p_support_user_id", supportUserId);

        cmd.ExecuteNonQuery();
    }

    /// <summary>
    /// Зарегистрировать нового пользователя через sp_register_user.
    /// Хэширование выполняется в C# до вызова процедуры.
    /// </summary>
    public void RegisterUser(string login, string password, string fullName,
                             string email, int roleId, int deptId)
    {
        string hash = PasswordHelper.HashPassword(password,
PasswordHelper.GenerateSalt());

        using var conn = DBConnection.GetConnection();
        conn.Open();

        using var cmd = new MySqlCommand("sp_register_user", conn)
        {
            CommandType = CommandType.StoredProcedure
        };

        cmd.Parameters.AddWithValue("p_login",    login);
        cmd.Parameters.AddWithValue("p_hash",     hash);
        cmd.Parameters.AddWithValue("p_fullname", fullName);
        cmd.Parameters.AddWithValue("p_email",    email);
        cmd.Parameters.AddWithValue("p_role_id",  roleId);
        cmd.Parameters.AddWithValue("p_dept_id",  deptId);

        cmd.ExecuteNonQuery();
    }
}
```

4.3 Обработка ошибок процедуры (SIGNAL) в C

Если процедура вызывает `SIGNAL SQLSTATE '45000'`, MySQL генерирует исключение, которое в C# ловится как `MySqlException`.

```
private void btnAssign_Click(object sender, EventArgs e)
{
    if (dataGridViewTickets.SelectedRows.Count == 0) return;

    int ticketId      =
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер disynergy.ru

```
Convert.ToInt32(dataGridViewTickets.SelectedRows[0].Cells["ticket_id"].Value);
    int supportUserId = Convert.ToInt32(comboBoxSupport.SelectedValue);

    try
    {
        var service = new TicketService();
        service.AssignTicket(ticketId, supportUserId);
        MessageBox.Show("Исполнитель назначен.", "Успех",
                        MessageBoxButtons.OK, MessageBoxIcon.Information);
        LoadTickets(); // обновить DataGridView
    }
    catch (MySql.Data.MySqlClient.MySqlException ex)
    {
        MessageBox.Show($"Ошибка БД: {ex.Message}", "Ошибка",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Вывод по этапу

На данном этапе созданы объекты базы данных системы Helpdesk:

- 2 представления: `v_tickets_full` (JOIN из 5 таблиц) и `v_support_stats` (агрегат с `COUNT` и `CASE`).
- 3 хранимые процедуры: `sp_change_ticket_status`, `sp_assign_ticket`, `sp_register_user` — автоматизируют основные бизнес-операции.
- 2 триггера: `trg_log_status_change` (аудит в `AFTER UPDATE`) и `trg_update_timestamp` (авто-поле в `BEFORE UPDATE`).

Показано обращение к VIEW и вызов процедур из C# через `MySqlCommand` с `CommandType.StoredProcedure`.

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер disynergy.ru

Этап 3. Безопасность — роли СУБД и хэширование паролей

Задание

На данном этапе необходимо:

1. Создать не менее 3 ролей СУБД с разными уровнями доступа (только чтение, запись, администратор).
2. Создать пользователей MySQL и назначить им роли.
3. Реализовать хэширование паролей пользователей приложения (SHA-256 + соль или bcrypt).
4. Показать процесс регистрации пользователя с хэшированием в C#.
5. Показать проверку пароля при входе в C#.
6. Обновить схему таблицы `users` — добавить поле для хранения соли.
7. Сделать скриншоты: список ролей в MySQL, SHOW GRANTS, форма входа в приложении.

> ⚠ **Важно**
>
> Ниже приведён **пример** выполнения этапа для предметной области «Helpdesk» (MySQL + C# Windows Forms). В своём варианте создайте роли и реализуйте хэширование для **вашей** предметной области.

Пример: безопасность системы Helpdesk

1. Роли СУБД (MySQL 8+)

1.1 Концепция ролей для Helpdesk

Для системы Helpdesk определены три роли СУБД, соответствующие принципу минимальных привилегий:

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Роль СУБД	Соответствие в приложении	Разрешённые операции
`helpdesk_readonly`	Просмотр отчётов (клиентский интерфейс)	`SELECT` на все таблицы БД
`helpdesk_support`	Специалист поддержки	`SELECT, INSERT, UPDATE` на рабочие таблицы
`helpdesk_admin`	Администратор системы	`ALL PRIVILEGES` на БД

1.2 SQL-скрипт создания ролей и пользователей

```
-- ============================================================
-- Создание ролей (MySQL 8+)
-- ============================================================

CREATE ROLE IF NOT EXISTS 'helpdesk_readonly';
CREATE ROLE IF NOT EXISTS 'helpdesk_support';
CREATE ROLE IF NOT EXISTS 'helpdesk_admin';

-- ------------------------------------------------------------
-- Привилегии роли: только чтение
-- ------------------------------------------------------------
GRANT SELECT ON helpdesk.* TO 'helpdesk_readonly';

-- ------------------------------------------------------------
-- Привилегии роли: специалист поддержки
-- ------------------------------------------------------------
GRANT SELECT, INSERT, UPDATE ON helpdesk.tickets         TO
'helpdesk_support';
GRANT SELECT, INSERT, UPDATE ON helpdesk.comments        TO
'helpdesk_support';
GRANT SELECT, INSERT         ON helpdesk.ticket_history  TO
'helpdesk_support';
GRANT SELECT                 ON helpdesk.users           TO
'helpdesk_support';
GRANT SELECT                 ON helpdesk.categories      TO
'helpdesk_support';
GRANT SELECT                 ON helpdesk.departments     TO
'helpdesk_support';
GRANT SELECT                 ON helpdesk.roles           TO
'helpdesk_support';

-- Разрешить вызов хранимых процедур
GRANT EXECUTE ON PROCEDURE helpdesk.sp_change_ticket_status TO
'helpdesk_support';
GRANT EXECUTE ON PROCEDURE helpdesk.sp_assign_ticket        TO
'helpdesk_support';

-- ------------------------------------------------------------
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

```
-- Привилегии роли: администратор БД
-- ------------------------------------------------------------
GRANT ALL PRIVILEGES ON helpdesk.* TO 'helpdesk_admin';

-- ============================================================
-- Создание учётных записей MySQL и назначение ролей
-- ============================================================

-- Пользователь для клиентского интерфейса (только чтение)
CREATE USER IF NOT EXISTS 'app_readonly'@'localhost'
    IDENTIFIED BY 'ReadPass!23';
GRANT 'helpdesk_readonly' TO 'app_readonly'@'localhost';
SET DEFAULT ROLE ALL TO 'app_readonly'@'localhost';

-- Пользователь для специалиста поддержки
CREATE USER IF NOT EXISTS 'app_support'@'localhost'
    IDENTIFIED BY 'SuppPass!23';
GRANT 'helpdesk_support' TO 'app_support'@'localhost';
SET DEFAULT ROLE ALL TO 'app_support'@'localhost';

-- Пользователь-администратор
CREATE USER IF NOT EXISTS 'app_admin'@'localhost'
    IDENTIFIED BY 'AdminPass!23';
GRANT 'helpdesk_admin' TO 'app_admin'@'localhost';
SET DEFAULT ROLE ALL TO 'app_admin'@'localhost';

FLUSH PRIVILEGES;
```

1.3 Проверка привилегий

```
-- Привилегии пользователя app_support
SHOW GRANTS FOR 'app_support'@'localhost';

-- Привилегии с учётом ролей
SHOW GRANTS FOR 'app_support'@'localhost' USING 'helpdesk_support';
```

---

2. Обновление таблицы `users` — добавление соли

В схеме из Этапа 1 таблица `users` содержала только `password_hash VARCHAR(64)`. Для корректной реализации SHA-256 + соль необходимо добавить поле `password_salt`.

```
-- Добавить столбец password_salt (если таблица уже создана)
ALTER TABLE users
    ADD COLUMN password_salt VARCHAR(32) NOT NULL DEFAULT ''
    AFTER password_hash;

-- Обновить ограничение: password_salt не может быть пустым
ALTER TABLE users
    MODIFY COLUMN password_salt VARCHAR(32) NOT NULL;
```

Обновлённая структура полей пароля в таблице `users`:

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Поле	Тип	Описание
`password_salt`	`VARCHAR(32)`	Случайная соль — hex-строка (32 символа = 16 байт)
`password_hash`	`VARCHAR(64)`	SHA-256 хэш от `(salt + password)`, hex-строка (64 символа)

Обновлённая хранимая процедура `sp_register_user`

Процедура теперь принимает соль и хэш отдельными параметрами. Хэширование выполняется в C#.

```
DELIMITER $$

DROP PROCEDURE IF EXISTS sp_register_user$$

CREATE PROCEDURE sp_register_user(
    IN p_login    VARCHAR(80),
    IN p_salt     VARCHAR(32),
    IN p_hash     VARCHAR(64),
    IN p_fullname VARCHAR(150),
    IN p_email    VARCHAR(150),
    IN p_role_id  INT,
    IN p_dept_id  INT
)
BEGIN
    IF EXISTS (SELECT 1 FROM users WHERE login = p_login) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Логин уже занят';
    END IF;

    IF EXISTS (SELECT 1 FROM users WHERE email = p_email) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Email уже зарегистрирован';
    END IF;

    INSERT INTO users
        (login, password_salt, password_hash, full_name, email,
         role_id, department_id, is_active, created_at)
    VALUES
        (p_login, p_salt, p_hash, p_fullname, p_email,
         p_role_id, p_dept_id, 1, NOW());
END$$

DELIMITER ;
```

---

3. Хэширование паролей в C

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

3.1 Класс `PasswordHelper`

```
using System;
using System.Security.Cryptography;
using System.Text;

/// <summary>
/// Вспомогательный класс для безопасного хэширования паролей.
/// Алгоритм: SHA-256 с уникальной солью для каждого пользователя.
/// </summary>
public static class PasswordHelper
{
    /// <summary>
    /// Генерирует случайную соль: 16 случайных байт → hex-строка (32
символа).
    /// Соль уникальна для каждого пользователя и хранится в БД открыто.
    /// </summary>
    public static string GenerateSalt()
    {
        byte[] saltBytes = new byte[16];
        using (var rng = RandomNumberGenerator.Create())
            rng.GetBytes(saltBytes);
        return BitConverter.ToString(saltBytes).Replace("-", "").ToLower();
    }

    /// <summary>
    /// Вычисляет SHA-256 хэш от строки (соль + пароль).
    /// Возвращает hex-строку (64 символа).
    /// </summary>
    public static string HashPassword(string password, string salt)
    {
        using var sha = SHA256.Create();
        byte[] inputBytes = Encoding.UTF8.GetBytes(salt + password);
        byte[] hashBytes  = sha.ComputeHash(inputBytes);
        return BitConverter.ToString(hashBytes).Replace("-", "").ToLower();
    }

    /// <summary>
    /// Проверяет введённый пароль: вычисляет хэш от (соль + пароль)
    /// и сравнивает с хранимым хэшем без учёта регистра.
    /// </summary>
    public static bool VerifyPassword(string inputPassword, string salt,
string storedHash)
    {
        string computedHash = HashPassword(inputPassword, salt);
        return computedHash.Equals(storedHash,
StringComparison.OrdinalIgnoreCase);
    }
}
```

3.2 Регистрация пользователя в C# WinForms

```
using MySql.Data.MySqlClient;
using System;
using System.Data;
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер disynergy.ru

```
using System.Windows.Forms;

public partial class RegisterForm : Form
{
    private void btnRegister_Click(object sender, EventArgs e)
    {
        string login    = txtLogin.Text.Trim();
        string password = txtPassword.Text;
        string fullName = txtFullName.Text.Trim();
        string email    = txtEmail.Text.Trim();
        int    roleId   = Convert.ToInt32(comboBoxRole.SelectedValue);
        int    deptId   = Convert.ToInt32(comboBoxDept.SelectedValue);

        // Базовая валидация
        if (string.IsNullOrWhiteSpace(login) || password.Length < 6)
        {
            MessageBox.Show("Логин не может быть пустым, пароль — не менее 6
символов.",
                            "Ошибка ввода", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
            return;
        }

        // Генерация соли и хэша в C# (до обращения к БД)
        string salt = PasswordHelper.GenerateSalt();
        string hash = PasswordHelper.HashPassword(password, salt);

        try
        {
            using var conn = DBConnection.GetConnection();
            conn.Open();

            using var cmd = new MySqlCommand("sp_register_user", conn)
            {
                CommandType = CommandType.StoredProcedure
            };

            cmd.Parameters.AddWithValue("p_login",    login);
            cmd.Parameters.AddWithValue("p_salt",     salt);
            cmd.Parameters.AddWithValue("p_hash",     hash);
            cmd.Parameters.AddWithValue("p_fullname", fullName);
            cmd.Parameters.AddWithValue("p_email",    email);
            cmd.Parameters.AddWithValue("p_role_id",  roleId);
            cmd.Parameters.AddWithValue("p_dept_id",  deptId);

            cmd.ExecuteNonQuery();

            MessageBox.Show($"Пользователь «{login}» успешно
зарегистрирован.",
                            "Регистрация", MessageBoxButtons.OK,
MessageBoxIcon.Information);
            this.Close();
        }
        catch (MySqlException ex)
        {
            MessageBox.Show($"Ошибка: {ex.Message}", "Ошибка БД",
                            MessageBoxButtons.OK, MessageBoxIcon.Error);
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения

Нажмите на мессенджер, чтобы написать

Telegram ↗ WhatsApp ↗ MAX ↗

+7 (924) 305-23-08 · Пишите в мессенджер disynergy.ru

```
        }
    }
}
```

3.3 Проверка пароля при входе в C# WinForms

```
public partial class LoginForm : Form
{
    private void btnLogin_Click(object sender, EventArgs e)
    {
        string login    = txtLogin.Text.Trim();
        string password = txtPassword.Text;

        if (string.IsNullOrWhiteSpace(login) ||
string.IsNullOrWhiteSpace(password))
        {
            MessageBox.Show("Введите логин и пароль.", "Ошибка",
                            MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return;
        }

        try
        {
            using var conn = DBConnection.GetConnection();
            conn.Open();

            // Шаг 1: получить соль, хэш и данные пользователя по логину
            const string sql = @"
                SELECT u.user_id, u.full_name, u.password_salt,
u.password_hash,
                       r.role_name
                FROM users u
                JOIN roles r ON u.role_id = r.role_id
                WHERE u.login = @login AND u.is_active = 1";

            using var cmd = new MySqlCommand(sql, conn);
            cmd.Parameters.AddWithValue("@login", login);
            using var reader = cmd.ExecuteReader();

            if (!reader.Read())
            {
                MessageBox.Show("Пользователь не найден или заблокирован.",
"Ошибка входа",
                                MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return;
            }

            int    userId    = reader.GetInt32("user_id");
            string fullName  = reader.GetString("full_name");
            string salt      = reader.GetString("password_salt");
            string storedHash= reader.GetString("password_hash");
            string roleName  = reader.GetString("role_name");
            reader.Close();

            // Шаг 2: проверить пароль
            if (!PasswordHelper.VerifyPassword(password, salt, storedHash))
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

```
            {
                MessageBox.Show("Неверный пароль.", "Ошибка входа",
                                MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return;
            }

            // Шаг 3: сохранить данные сессии и открыть главную форму
            AppSession.UserId   = userId;
            AppSession.UserName = fullName;
            AppSession.RoleName = roleName;

            var mainForm = new MainForm();
            mainForm.Show();
            this.Hide();
        }
        catch (MySqlException ex)
        {
            MessageBox.Show($"Ошибка подключения к БД: {ex.Message}",
"Ошибка",
                            MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}

/// <summary>
/// Хранит данные текущего сеанса (статический класс).
/// </summary>
public static class AppSession
{
    public static int    UserId   { get; set; }
    public static string UserName { get; set; }
    public static string RoleName { get; set; }

    public static void Clear()
    {
        UserId   = 0;
        UserName = null;
        RoleName = null;
    }
}
```

Вывод по этапу

На данном этапе реализованы следующие меры безопасности для системы Helpdesk:

Уровень СУБД: созданы 3 роли MySQL (`helpdesk_readonly` , `helpdesk_support` , `helpdesk_admin`) с разграниченными привилегиями по принципу минимальных прав. Каждой роли назначен отдельный пользователь MySQL.

Уровень приложения: реализовано хэширование паролей по схеме SHA-256 + уникальная соль. Схема таблицы `users` расширена полем `password_salt` .

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Хэширование выполняется в C# средствами `System.Security.Cryptography.SHA256`. Процедура входа: получить соль → вычислить хэш → сравнить с хранимым.

Пароли в базе данных хранятся **только в виде хэша** — восстановить исходный пароль по хэшу невозможно.

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Этап 4. CRUD и запросы с JOIN

Задание

На данном этапе необходимо:

1. Написать и выполнить запросы с INNER JOIN, LEFT JOIN, RIGHT JOIN (минимум по одному каждого вида).
2. Продемонстрировать все четыре CRUD-операции через приложение C# WinForms.
3. Реализовать загрузку данных в `DataGridView` через View или SELECT.
4. Реализовать создание, редактирование и удаление записи с параметризованными запросами.
5. Описать архитектуру приложения: список форм и схему навигации.
6. Привести класс `DBConnection` для управления подключением.
7. Сделать скриншоты: результаты каждого типа JOIN, формы приложения с данными.

> ⚠ **Важно**
>
> Ниже приведён **пример** выполнения этапа для предметной области «Helpdesk» (MySQL + C# Windows Forms). В своём варианте реализуйте CRUD и JOIN-запросы для **вашей** предметной области.

Пример: CRUD и JOIN для системы Helpdesk

1. JOIN-запросы

1.1 INNER JOIN — заявки с именем автора

Возвращает только те заявки, у которых гарантированно указан автор (поле `created_by` ссылается на существующего пользователя).

```
SELECT
    t.ticket_id,
    t.title,
    t.status,
    u.full_name AS author
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

```
FROM tickets t
JOIN users u ON t.created_by = u.user_id
ORDER BY t.created_at DESC;
```

1.2 INNER JOIN (3 таблицы) — полная информация о заявке

Объединяет заявку с категорией, автором, отделом автора и исполнителем. `LEFT JOIN` применяется для необязательных связей.

```
SELECT
    t.ticket_id,
    t.title,
    t.status,
    t.priority,
    c.name          AS category,
    u1.full_name    AS author,
    d.name          AS department,
    u2.full_name    AS assignee
FROM tickets t
JOIN  categories c   ON t.category_id   = c.category_id
JOIN  users u1       ON t.created_by    = u1.user_id
LEFT JOIN departments d  ON u1.department_id = d.department_id
LEFT JOIN users u2   ON t.assigned_to   = u2.user_id
ORDER BY t.created_at DESC;
```

1.3 LEFT JOIN — все отделы, включая пустые (без сотрудников)

```
SELECT
    d.name        AS department,
    COUNT(u.user_id) AS staff_count
FROM departments d
LEFT JOIN users u ON d.department_id = u.department_id
GROUP BY d.department_id, d.name
ORDER BY staff_count DESC;
```

1.4 LEFT JOIN — заявки без назначенного исполнителя

Классический паттерн: `LEFT JOIN ... WHERE right_id IS NULL` находит записи без соответствия в правой таблице.

```
SELECT
    t.ticket_id,
    t.title,
    t.status,
    t.priority,
    t.created_at
FROM tickets t
LEFT JOIN users u ON t.assigned_to = u.user_id
WHERE u.user_id IS NULL
ORDER BY t.priority DESC, t.created_at ASC;
```

1.5 RIGHT JOIN — все категории, включая те без заявок

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

```sql
SELECT
    c.name           AS category,
    COUNT(t.ticket_id) AS ticket_count
FROM tickets t
RIGHT JOIN categories c ON t.category_id = c.category_id
GROUP BY c.category_id, c.name
ORDER BY ticket_count DESC;
```

1.6 Агрегат с GROUP BY + HAVING — категории с более чем 2 заявками

```sql
SELECT
    c.name           AS category,
    COUNT(t.ticket_id) AS ticket_count,
    AVG(CASE t.priority
            WHEN 'critical' THEN 4
            WHEN 'high'     THEN 3
            WHEN 'medium'   THEN 2
            WHEN 'low'      THEN 1
            ELSE 2
        END) AS avg_priority_score
FROM categories c
JOIN tickets t ON c.category_id = t.category_id
GROUP BY c.category_id, c.name
HAVING COUNT(t.ticket_id) > 2
ORDER BY ticket_count DESC;
```

---

2. CRUD в C# WinForms

2.1 Класс `DBConnection` — управление подключением

```csharp
using MySql.Data.MySqlClient;
using System.Configuration;

/// <summary>
/// Статический класс для получения соединения с MySQL.
/// Строка подключения берётся из App.config.
/// </summary>
public static class DBConnection
{
    private static readonly string _connectionString =
        ConfigurationManager.ConnectionStrings["HelpdeskDB"].ConnectionString;

    /// <summary>
    /// Возвращает новый объект MySqlConnection.
    /// Вызывающий код обязан вызвать conn.Open() и затем Dispose().
    /// </summary>
    public static MySqlConnection GetConnection()
    {
        return new MySqlConnection(_connectionString);
    }
}
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Строка подключения в `App.config`:

```
<connectionStrings>
  <add name="HelpdeskDB"
       connectionString="server=localhost;port=3306;database=helpdesk;
                         user=app_support;password=SuppPass!23;
                         charset=utf8mb4;"
       providerName="MySql.Data.MySqlClient" />
</connectionStrings>
```

2.2 READ — загрузка заявок из VIEW в DataGridView

```
public partial class TicketsForm : Form
{
    private void LoadTickets(string statusFilter = null)
    {
        try
        {
            using var conn = DBConnection.GetConnection();
            conn.Open();

            string sql = @"
                SELECT ticket_id   AS `№`,
                       title       AS `Тема`,
                       status      AS `Статус`,
                       priority    AS `Приоритет`,
                       category    AS `Категория`,
                       author      AS `Автор`,
                       author_dept AS `Отдел`,
                       assignee    AS `Исполнитель`,
                       created_at  AS `Создана`
                FROM v_tickets_full";

            if (!string.IsNullOrEmpty(statusFilter))
                sql += " WHERE status = @status";

            sql += " ORDER BY created_at DESC";

            using var cmd     = new MySqlCommand(sql, conn);
            if (!string.IsNullOrEmpty(statusFilter))
                cmd.Parameters.AddWithValue("@status", statusFilter);

            using var adapter = new MySqlDataAdapter(cmd);
            var table = new DataTable();
            adapter.Fill(table);

            dataGridViewTickets.DataSource = table;

            // Скрыть столбец с ID (он нужен для операций, но не для
отображения)
            if (dataGridViewTickets.Columns.Contains("№"))
                dataGridViewTickets.Columns["№"].Visible = false;

            dataGridViewTickets.AutoResizeColumns();
        }
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

```
        catch (MySqlException ex)
        {
            MessageBox.Show($"Ошибка загрузки данных: {ex.Message}", "Ошибка",
                            MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}
```

2.3 CREATE — создание новой заявки

```
public partial class NewTicketForm : Form
{
    private void btnCreate_Click(object sender, EventArgs e)
    {
        string title       = txtTitle.Text.Trim();
        string description = txtDescription.Text.Trim();
        int    categoryId  = Convert.ToInt32(comboBoxCategory.SelectedValue);
        string priority    = comboBoxPriority.SelectedItem?.ToString() ??
"medium";

        if (string.IsNullOrWhiteSpace(title))
        {
            MessageBox.Show("Введите тему заявки.", "Ошибка",
                            MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return;
        }

        try
        {
            using var conn = DBConnection.GetConnection();
            conn.Open();

            const string sql = @"
                INSERT INTO tickets
                    (title, description, status, priority, category_id,
created_by, created_at)
                VALUES
                    (@title, @desc, 'new', @priority, @catId, @createdBy,
NOW())";

            using var cmd = new MySqlCommand(sql, conn);
            cmd.Parameters.AddWithValue("@title",     title);
            cmd.Parameters.AddWithValue("@desc",      description);
            cmd.Parameters.AddWithValue("@priority",  priority);
            cmd.Parameters.AddWithValue("@catId",     categoryId);
            cmd.Parameters.AddWithValue("@createdBy", AppSession.UserId);

            cmd.ExecuteNonQuery();

            // Получить ID созданной заявки
            long newId = cmd.LastInsertedId;

            MessageBox.Show($"Заявка №{newId} успешно создана.", "Успех",
                            MessageBoxButtons.OK, MessageBoxIcon.Information);
            this.DialogResult = DialogResult.OK;
            this.Close();
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер disynergy.ru

```
        }
        catch (MySqlException ex)
        {
            MessageBox.Show($"Ошибка: {ex.Message}", "Ошибка БД",
                            MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}
```

2.4 UPDATE — изменение статуса заявки через хранимую процедуру

```
public partial class TicketsForm : Form
{
    private void btnChangeStatus_Click(object sender, EventArgs e)
    {
        if (dataGridViewTickets.SelectedRows.Count == 0)
        {
            MessageBox.Show("Выберите заявку.", "Внимание",
                            MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return;
        }

        // Получить ID из скрытого столбца
        int    ticketId  =
Convert.ToInt32(dataGridViewTickets.SelectedRows[0].Cells["№"].Value);
        string newStatus = comboBoxNewStatus.SelectedItem?.ToString();

        if (string.IsNullOrEmpty(newStatus)) return;

        try
        {
            using var conn = DBConnection.GetConnection();
            conn.Open();

            using var cmd = new MySqlCommand("sp_change_ticket_status", conn)
            {
                CommandType = System.Data.CommandType.StoredProcedure
            };
            cmd.Parameters.AddWithValue("p_ticket_id",  ticketId);
            cmd.Parameters.AddWithValue("p_new_status", newStatus);
            cmd.Parameters.AddWithValue("p_user_id",    AppSession.UserId);
            cmd.Parameters.AddWithValue("p_note",       $"Изменено через форму
пользователем {AppSession.UserName}");

            cmd.ExecuteNonQuery();

            MessageBox.Show($"Статус заявки №{ticketId} изменён на
«{newStatus}».", "Успех",
                            MessageBoxButtons.OK, MessageBoxIcon.Information);
            LoadTickets();
        }
        catch (MySqlException ex)
        {
            MessageBox.Show($"Ошибка: {ex.Message}", "Ошибка БД",
                            MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

```
    }
}
```

2.5 DELETE — удаление заявки с подтверждением

```
private void btnDelete_Click(object sender, EventArgs e)
{
    if (dataGridViewTickets.SelectedRows.Count == 0) return;

    int    ticketId =
Convert.ToInt32(dataGridViewTickets.SelectedRows[0].Cells["№"].Value);
    string title    =
dataGridViewTickets.SelectedRows[0].Cells["Тема"].Value?.ToString();

    var confirm = MessageBox.Show(
        $"Удалить заявку №{ticketId} «{title}»?\n\nВместе с ней будут удалены
все комментарии и история.",
        "Подтверждение удаления",
        MessageBoxButtons.YesNo,
        MessageBoxIcon.Warning);

    if (confirm != DialogResult.Yes) return;

    try
    {
        using var conn = DBConnection.GetConnection();
        conn.Open();

        // Благодаря ON DELETE CASCADE удалятся comments и ticket_history
автоматически
        const string sql = "DELETE FROM tickets WHERE ticket_id = @id";
        using var cmd = new MySqlCommand(sql, conn);
        cmd.Parameters.AddWithValue("@id", ticketId);
        cmd.ExecuteNonQuery();

        MessageBox.Show($"Заявка №{ticketId} удалена.", "Удалено",
                        MessageBoxButtons.OK, MessageBoxIcon.Information);
        LoadTickets();
    }
    catch (MySqlException ex)
    {
        MessageBox.Show($"Ошибка при удалении: {ex.Message}", "Ошибка БД",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

---

3. Архитектура приложения C# WinForms

3.1 Структура проекта

Приложение реализовано как один проект C# WinForms (.NET). Формы организованы по принципу «одна форма — одна функциональная область».

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Форма	Назначение
`LoginForm`	Авторизация пользователя
`MainForm`	Главное окно (MDI-контейнер или форма с вкладками); меню навигации
`TicketsForm`	Список заявок, фильтры, управление статусами
`NewTicketForm`	Создание новой заявки
`TicketDetailForm`	Просмотр заявки, комментарии, история изменений
`UsersForm`	Управление пользователями (только для Admin)
`CategoriesForm`	Управление категориями
`DepartmentsForm`	Управление отделами
`ReportsForm`	Статистика: `v_support_stats`, агрегирующие запросы

3.2 Схема навигации

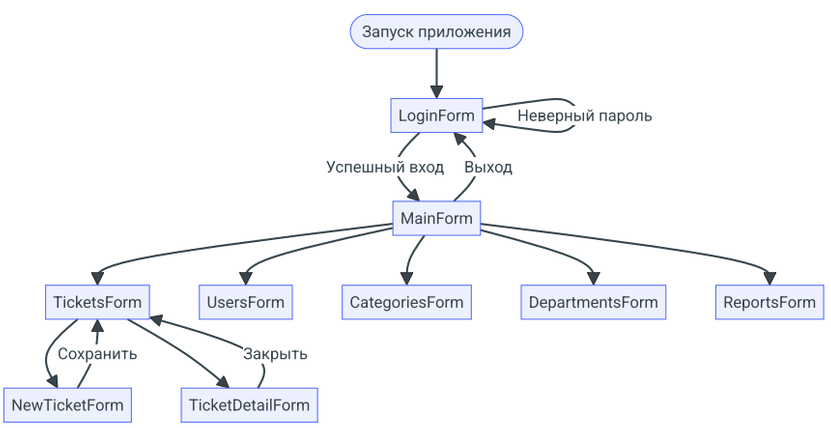

Рисунок 1 — Схема навигации между формами приложения Helpdesk

3.3 Управление доступом на уровне интерфейса

После входа в систему `AppSession.RoleName` определяет видимость элементов интерфейса.

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

```
public partial class MainForm : Form
{
    private void ApplyRolePermissions()
    {
        bool isAdmin   = AppSession.RoleName == "admin";
        bool isSupport = AppSession.RoleName == "support" || isAdmin;

        // Управление пользователями — только администратор
        menuItemUsers.Visible       = isAdmin;
        menuItemDepartments.Visible = isAdmin;

        // Назначение исполнителей — поддержка и администратор
        btnAssignTicket.Visible = isSupport;
        btnChangeStatus.Visible = isSupport;

        // Отображение имени пользователя в заголовке
        this.Text = $"Helpdesk — {AppSession.UserName}
[{AppSession.RoleName}]";
    }
}
```

3.4 Загрузка справочников (ComboBox)

```
/// <summary>
/// Универсальный метод загрузки справочника в ComboBox.
/// </summary>
private void LoadComboBox(ComboBox comboBox, string query,
                          string valueMember, string displayMember)
{
    using var conn = DBConnection.GetConnection();
    conn.Open();
    using var adapter = new MySqlDataAdapter(query, conn);
    var table = new DataTable();
    adapter.Fill(table);

    comboBox.DataSource    = table;
    comboBox.ValueMember   = valueMember;
    comboBox.DisplayMember = displayMember;
}

// Использование в форме:
private void NewTicketForm_Load(object sender, EventArgs e)
{
    LoadComboBox(comboBoxCategory,
        "SELECT category_id, name FROM categories ORDER BY name",
        "category_id", "name");

    LoadComboBox(comboBoxPriority_dummy, null, null, null); // заполняется
вручную
    comboBoxPriority.Items.AddRange(new[] { "low", "medium", "high",
"critical" });
    comboBoxPriority.SelectedIndex = 1; // medium по умолчанию
}
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер disynergy.ru

Вывод по этапу

На данном этапе реализованы все четыре CRUD-операции для таблицы `tickets` в C# WinForms:

- **READ** — загрузка из VIEW `v_tickets_full` в `DataGridView` с фильтрацией по статусу;
- **CREATE** — параметризованный `INSERT` с получением `LastInsertedId`;
- **UPDATE** — вызов хранимой процедуры `sp_change_ticket_status` через `CommandType.StoredProcedure`;
- **DELETE** — параметризованный `DELETE` с каскадным удалением через `ON DELETE CASCADE`.

Выполнены пять видов JOIN-запросов: `INNER JOIN` (с 2 и 5 таблицами), `LEFT JOIN` (агрегат и поиск «записей без пары»), `RIGHT JOIN` (все категории включая пустые).

Описана архитектура приложения: 9 форм, схема навигации, управление видимостью UI-элементов на основе роли пользователя.

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru