

ОТЧЁТ

Лабораторная работа №2. Схема данных. EER-диаграмма

Выполнил: Тоц Леонид Александрович

Группа: ИВТ-2

1. Цель работы

Продолжить знакомство с MySQL Workbench, языком запросов SQL и освоить инструмент проектирования баз данных в визуальном редакторе EER-диаграмм.

Задачи:

- Изучить интерфейс инструмента для построения EER-диаграмм;
- Освоить связывание таблиц базы данных (Relationship);
- Изучить понятие ключевого поля и типы ключей;
- Спроектировать собственную базу данных на основе технического задания.

2. Теоретическая часть

2.1. EER-диаграмма

EER (Extended Entity-Relationship Model) — расширенная модель «сущность-связь», используемая для визуального проектирования реляционных баз данных.

2.2. Типы связей между таблицами

Тип связи	Обозначение	Описание	Пример
Один к одному	1:1	Одной записи в таблице А соответствует ровно одна запись в таблице Б	Автомобильный номер ↔ Автомобиль
Один ко многим	1:N	Одной записи в таблице А соответствует одна или несколько записей в таблице Б	Студент ↔ Записи на курсы

Тип связи	Обозначение	Описание	Пример
Многие ко многим	N:M	Множеству записей в таблице А соответствует множество записей в таблице Б	Студенты ↔ Дисциплины ↔ Преподаватели

2.3. Идентифицирующие и неидентифицирующие связи

- **Identifying (идентифицирующая):** зависимая сущность не может существовать без независимой; внешний ключ входит в состав первичного ключа зависимой таблицы.
- **Non-identifying (неидентифицирующая):** зависимая сущность может существовать самостоятельно; внешний ключ не является частью первичного ключа.

2.4. Ключевые поля

- **Первичный ключ (PK)** — уникальный идентификатор записи в таблице. Может быть простым (один атрибут) или составным (несколько атрибутов).
- **Внешний ключ (FK)** — атрибут, устанавливающий связь с первичным ключом другой таблицы, обеспечивая целостность данных.

3. Практическая часть

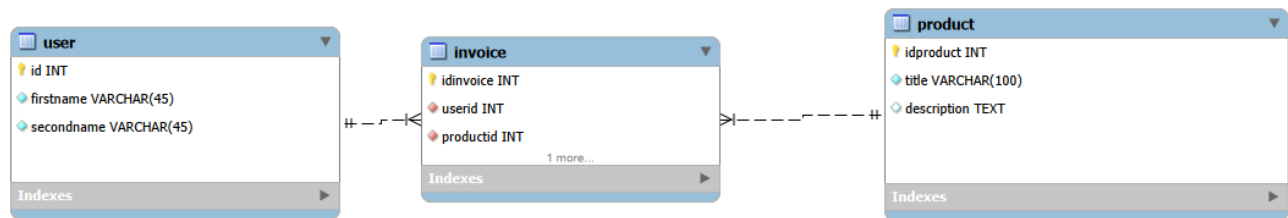
Задание 1. Воспроизведение диаграммы из обучающего ролика

Выполненные действия:

1. Открыт MySQL Workbench, загружена модель `sakila_full` для ознакомления с интерфейсом.
2. Просмотрен ролик «Обзор интерфейса инструмента проектирования EER-диаграмм».
3. Воссоздана диаграмма, продемонстрированная во второй половине ролика.
4. Выполнен экспорт модели:
 - В виде изображения (PNG);
 - В виде SQL-скрипта.

Результаты:

1. Схема в виде изображения:



2. Ссылка на SQL-скрипт создания БД.(Gist)

3. Фрагмент запроса для таблицы invoice :

```
CREATE TABLE IF NOT EXISTS `firstModel`.`invoice` (
  `idinvoice` INT NOT NULL AUTO_INCREMENT,
  `userid` INT NOT NULL,
  `productid` INT NOT NULL,
  `price` DECIMAL(10,2) NOT NULL,
  PRIMARY KEY (`idinvoice`),
  INDEX `user_idx` (`userid` ASC) VISIBLE,
  INDEX `prod_idx` (`productid` ASC) VISIBLE,
  CONSTRAINT `user`
    FOREIGN KEY (`userid`)
      REFERENCES `firstModel`.`user` (`id`)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
  CONSTRAINT `prod`
    FOREIGN KEY (`productid`)
      REFERENCES `firstModel`.`product` (`idproduct`)
      ON DELETE CASCADE
      ON UPDATE CASCADE)
ENGINE = InnoDB;
```

Задание 2. Проектирование собственной EER-диаграммы

Источник: Статья на Habr — <https://habr.com/ru/post/175985/>

Внесённые корректировки согласно инструкции:

Таблица	Поле	Исправление
products	name	Удалено как избыточное (название хранится в shop)
orders	tel	Тип изменён на VARCHAR(100) вместо VARCHAR(255)
deliveries	time	Тип изменён на VARCHAR(45) вместо VARCHAR(255)

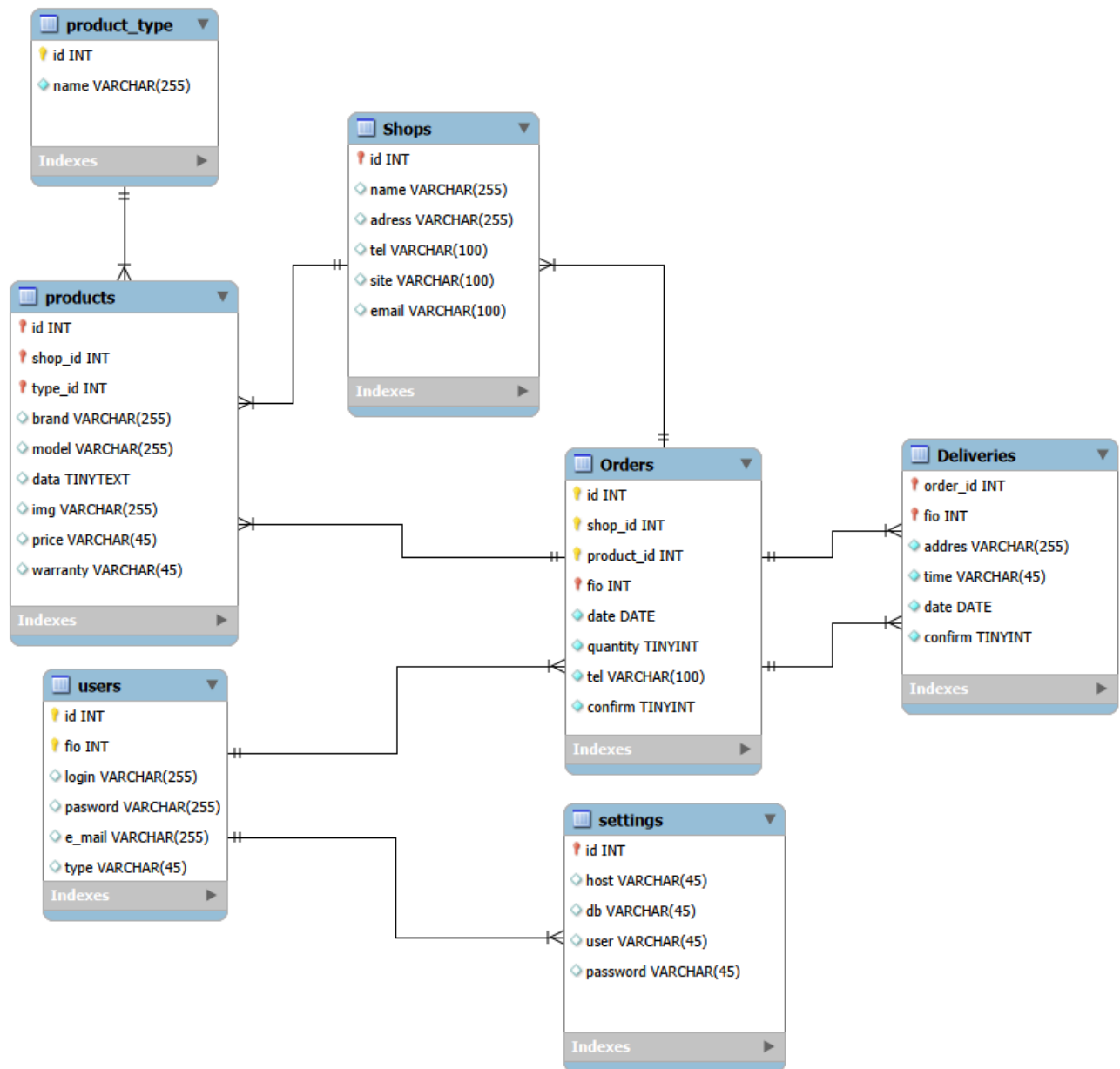
Таблица	Поле	Исправление
orders , deliveries	confirm	Тип TINYINT(1) вместо BOOLEAN (эквивалент в MySQL)

Спроектированные сущности и связи:

- shop (PK: shop_id)
- products (PK: product_id; FK: shop_id → shop)
- customers (PK: customer_id)
- orders (PK: order_id; FK: customer_id, shop_id)
- order_items (PK: order_item_id; FK: order_id, product_id)
- deliveries (PK: delivery_id; FK: order_id)

Результаты:

4. Схема в виде изображения:



5. [Ссылка на SQL-скрипт создания БД \(Gist\)](#)

6. Фрагмент запроса для таблицы Orders :

```
CREATE TABLE IF NOT EXISTS `mydb`.`Orders` (  
  `id` INT NOT NULL AUTO_INCREMENT,  
  `shop_id` INT NOT NULL,  
  `product_id` INT NOT NULL,  
  `fio` INT NOT NULL,  
  `date` DATE NOT NULL,  
  `quantity` TINYINT NOT NULL,  
  `tel` VARCHAR(100) NOT NULL,  
  `confirm` TINYINT NOT NULL,  
  PRIMARY KEY (`id`, `shop_id`, `product_id`, `fio`),  
  UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE,
```

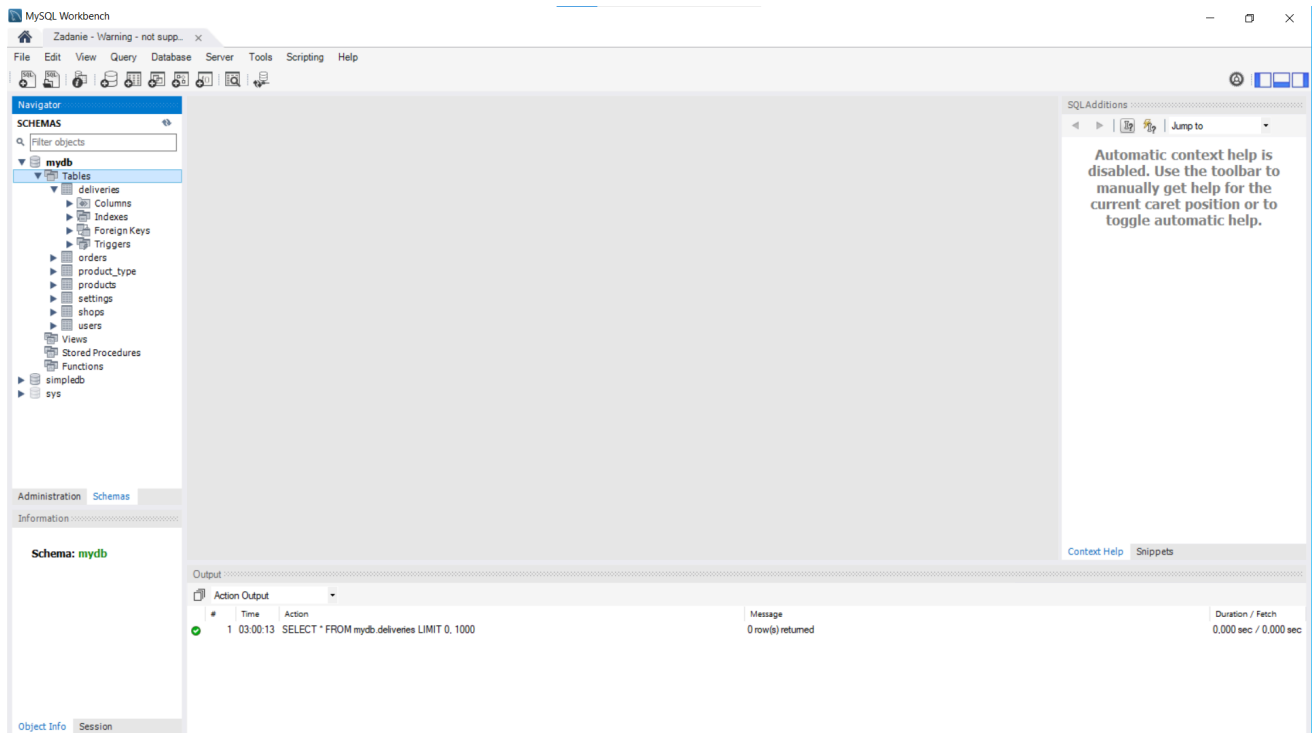
```
INDEX `fio_idx` (`fio` ASC) VISIBLE,  
CONSTRAINT `fio`  
  FOREIGN KEY (`fio`)  
  REFERENCES `mydb`.`users` (`fio`)  
  ON DELETE CASCADE  
  ON UPDATE CASCADE)  
ENGINE = InnoDB;
```

Задание 3. Forward Engineer — создание БД на сервере

Выполненные действия:

1. В MySQL Workbench выбрано меню: Database → Forward Engineer.
2. Указаны параметры подключения к локальному серверу MySQL.
3. Выбрана созданная модель, сгенерирован и выполнен SQL-скрипт.
4. Подтверждено успешное создание базы данных.

Результат:



Задание 4. Тестирование целостности данных

Выполненные действия:

1. Добавление тестовых данных:

```
-- Добавление магазинов
INSERT INTO shop (shop_id, name, address) VALUES
(1, 'Магазин №1', 'г. Москва, ул. Примерная, 10'),
(2, 'Магазин №2', 'г. СПб, Невский пр., 25');

-- Добавление товаров
INSERT INTO products (product_id, shop_id, title, price) VALUES
(101, 1, 'Товар А', 1500.00),
(102, 2, 'Товар Б', 2300.50);

-- Добавление клиентов и заказов
INSERT INTO customers (customer_id, name, email) VALUES
(1001, 'Иванов И.И.', 'ivanov@example.com');

INSERT INTO Orders (order_id, customer_id, shop_id, tel, confirm, total_sum)
VALUES
(5001, 1001, 1, '+79991234567', 1, 1500.00);
```

2. Добавление новых атрибутов:

```
ALTER TABLE Orders ADD COLUMN discount_percent DECIMAL(5,2) DEFAULT 0.00;
ALTER TABLE products ADD COLUMN stock_quantity INT DEFAULT 0;
ALTER TABLE deliveries ADD COLUMN courier_name VARCHAR(100);
```

3. Попытка удаления связанных данных:

Действие	Результат	Объяснение
DELETE FROM shop WHERE shop_id = 1;	Ошибка Foreign Key Constraint	На магазин ссылаются записи в products и orders
DELETE FROM customers WHERE customer_id = 1001;	Ошибка Foreign Key Constraint	Клиент связан с заказом в Orders
DELETE FROM Orders WHERE order_id = 5001;	Успешно	Нет зависимых записей в order_items / deliveries
После удаления заказа: DELETE FROM customers WHERE customer_id = 1001;	Успешно	Связи устранены, ограничение не нарушено

Выводы по заданию 4:

- СУБД MySQL корректно отслеживает целостность данных через внешние ключи.
 - При настройке `ON DELETE CASCADE` удаление родительской записи автоматически удаляет зависимые.
 - При `ON DELETE RESTRICT` (по умолчанию) операция блокируется до устранения зависимостей.
 - Добавление атрибутов через `ALTER TABLE` не нарушает существующие связи.
-

4. Выводы

В ходе выполнения лабораторной работы были освоены:

- Интерфейс MySQL Workbench для визуального проектирования БД;
- Принципы построения EER-диаграмм и типы связей между сущностями;
- Различия между первичными и внешними ключами, составными ключами;
- Практика экспорта моделей в изображение и SQL-скрипт;
- Применение Forward Engineer для развёртывания БД на сервере;
- Тестирование механизмов обеспечения целостности данных при операциях CRUD.

Полученные навыки позволяют самостоятельно проектировать реляционные базы данных средней сложности с учётом бизнес-правил и требований к целостности информации.