Модификация БД в SQL

Лазар В. И., Козлова Е. Р.

22 января 2025 г.

План занятия

1. Связи в базах данных
2. Что такое ER-модель?
3. Создание, модификация и удаление таблиц
4. Задания

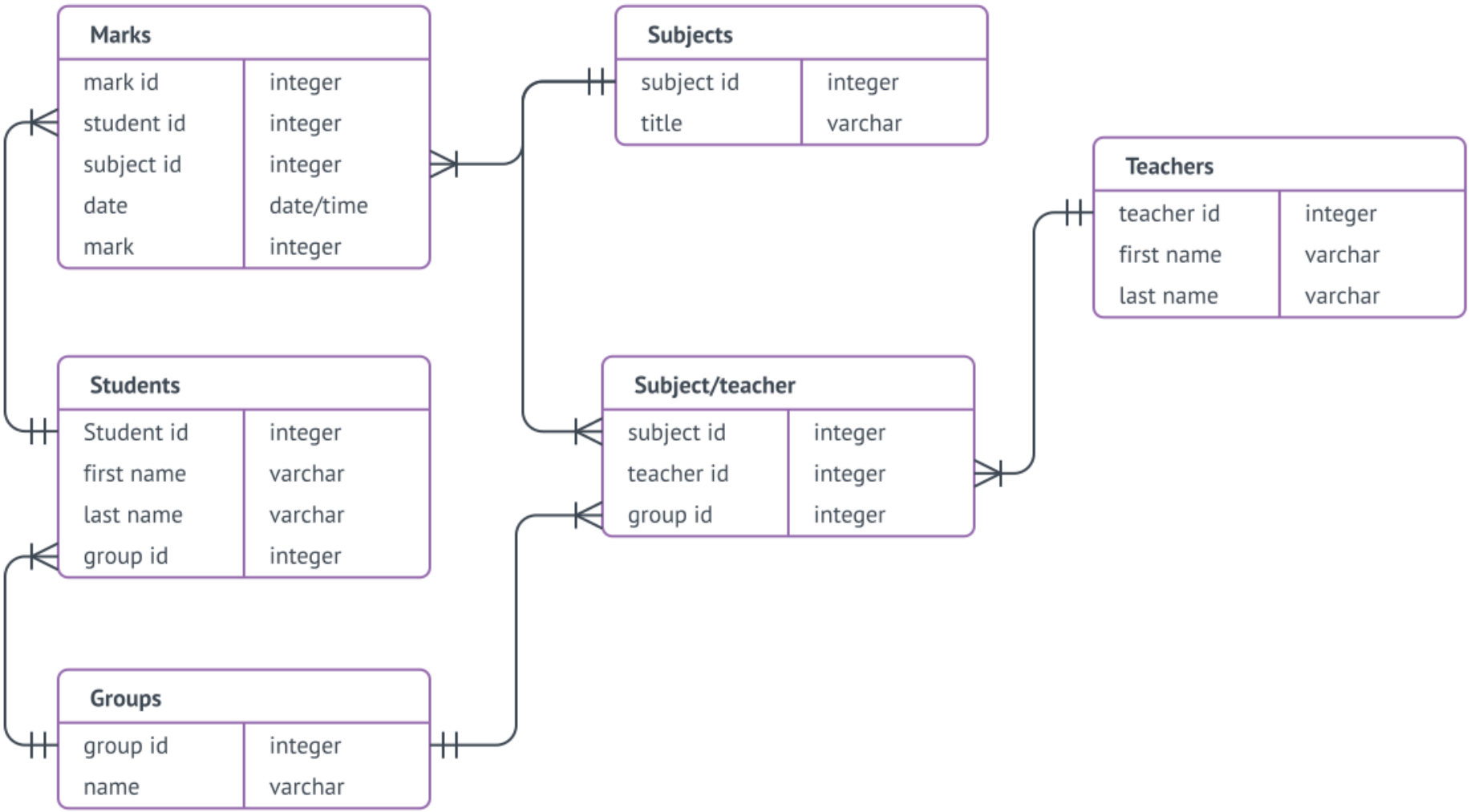
Зачем нужны связи?

- **Реляционная** модель данных предполагает хранение информации в таблицах, связанных между собой.
- Связи (*relationships*) позволяют избегать дублирования данных и повышают целостность:
  - **One-to-One** (1:1) — например, один сотрудник — одна личная учётная запись.
  - **One-to-Many** (1:N) — один отдел содержит много сотрудников.
  - **Many-to-Many** (M:N) — одна книга может иметь нескольких авторов и один автор — несколько книг.
- **FOREIGN KEY** — внешний ключ, указывающий на `PRIMARY KEY` другой таблицы. Реализует связь и гарантирует целостность (нельзя ссылаться на несуществующую запись).

Общее представление об ER-модели

- **ER-модель (Entity-Relationship model)** — это способ моделирования структуры базы данных с помощью сущностей (Entities) и связей (Relationships).
- **Сущность (Entity)** — это объект реального мира, информация о котором хранится (например, «Сотрудник»).
- **Связь (Relationship)** — показывает, как сущности соотносятся друг с другом (1:1, 1:N, M:N).
- **Атрибут (Attribute)** — свойство или характеристика сущности (например, «Имя», «Зарплата»).
- ER-модель обычно представляют в виде диаграммы: прямоугольники — сущности, ромбы (или линии с пометками) — связи, овалы — атрибуты.
- На основе ER-модели потом создаются таблицы и их связи (`FOREIGN KEY`) в реляционных СУБД.

Пример ER-модели



CREATE TABLE

Синтаксис

```
CREATE TABLE table_name (
        column1 datatype [CONSTRAINT ...] [AUTOINCREMENT],
        column2 datatype [CONSTRAINT ...] [AUTOINCREMENT],
    ...
);
```

Пример:

```
CREATE TABLE departments (
    dept_id INTEGER PRIMARY KEY,
    dep_name VARCHAR(100) NOT NULL
);
```

FOREIGN KEY в CREATE TABLE

Пример:

```
CREATE TABLE employees (
    emp_id     INTEGER PRIMARY KEY,
    emp_name   VARCHAR(50),
    position   VARCHAR(50),
    dept_id    INTEGER,
    FOREIGN KEY (dept_id)
        REFERENCES departments(dept_id)
);
```

Разбор:

- Столбец `dept_id` в `employees` ссылается на `departments.dept_id`.
- При попытке вставить `dept_id`, отсутствующий в `departments`, СУБД выдаст ошибку.

ALTER TABLE

Что можно сделать?

- **Добавить столбец**:

```
ALTER TABLE table_name
ADD COLUMN new_column datatype;
```

- **Удалить столбец**:

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

- **Добавить FOREIGN KEY**:

```
ALTER TABLE employees
ADD CONSTRAINT fk_emp_dept
FOREIGN KEY (dept_id)
REFERENCES departments(dept_id);
```

DROP TABLE

Синтаксис

```
DROP TABLE table_name;
```

Особенности:

- Удаляет таблицу целиком вместе со всеми данными.
- Если есть внешние ключи, может потребоваться сначала удалить связанные таблицы или отключить ограничения.

INSERT

Синтаксис

```
INSERT INTO table_name (col1, col2, ...)
VALUES (val1, val2, ...);
```

Пример:

```
INSERT INTO departments (dept_id, dept_name)
VALUES (1, 'Отдел продаж'),
       (2, 'Отдел разработки');
```

INSERT с учётом FOREIGN KEY

```
-- employees.dept_id -> departments.dept_id
INSERT INTO employees (emp_id, emp_name, position, dept_id)
VALUES
  (100, 'Иван Петров', 'Менеджер', 1),
  (101, 'Мария Сидорова', 'Разработчик', 2);

-- Если указать dept_id = 99, которого нет
-- в departments, будет ошибка
```

Задания

1. **Создание простой таблицы:** Создайте таблицу `products` с колонками `id` (PRIMARY KEY) и `name` (VARCHAR(50), NOT NULL).
2. **Добавление столбца:** В таблицу `products` добавьте новый столбец `price` (DECIMAL(10,2)), по умолчанию равный 0.
3. **Создание связанной таблицы:** Создайте таблицу `orders` со столбцами `order_id` (PRIMARY KEY), `product_id` (FOREIGN KEY, ссылается на `products.id`), `quantity` (целое число).
4. **Вставка данных:** Вставьте несколько записей в таблицу `products` (например, 3-4 продукта) и в таблицу `orders` (попробуйте сослаться на существующие `product_id`).

Задания

5. **Проверка внешнего ключа:** Убедитесь, что при попытке вставить запись в `orders` с `product_id`, которого нет в `products`, возникает ошибка.
6. **Изменение типа столбца:** Измените тип данных `price` в `products` на `FLOAT`.
7. **Добавление ограничения CHECK:** В таблицу `products` добавьте ограничение, запрещающее ставить `price` менее 0.
8. **Удаление столбца:** Удалите столбец `quantity` из таблицы `orders`. Примечание: заранее можете подумать, не повлияет ли это на логику.

Задания

9. **Создание таблицы со связью 1:N:** Создайте таблицу `categories` (`cat_id`, `cat_name`) и добавьте в `products` столбец `cat_id` (FOREIGN KEY), чтобы несколько продуктов могли принадлежать одной категории.
10. **Удаление и воссоздание таблиц:** Попробуйте удалить (`DROP TABLE`) таблицу `orders`, затем `products`, а затем заново воссоздать их, учитывая все необходимые FOREIGN KEY.
11. **Использование каскадного удаления/обновления:** Перед удалением таблиц измените внешние ключи `orders.product_id` так, чтобы при удалении строки из `products` автоматически удалялись связанные строки в `orders` (`ON DELETE CASCADE`).
12. **Добавление связанной таблицы пользователей:** Создайте таблицу `users` (`user_id`, `user_name`, `created_at`) и расширьте таблицу `orders`, добавив столбец `user_id` (FOREIGN KEY). Заполните таблицу `users` несколькими записями, вставьте связанные строки в `orders`.