1. Некоторые события нужно продавать с дополнительными типами билетов - льготный и групповой, у которых будут свои цены и название. Имеется информация, что вероятно, будут другие типы билетов, которые нужно будет добавить. Нужно уметь сохранять при заказе 2 дополнительных типа билета, льготный и групповой в бд. **Задача** - Нормализовать таблицу учитывая добавленные типы билетов, показать конечный вид таблицы. Объяснить свое решение.

Для решения этой задачи я предлагаю создать таблицу заказов. Она хранит информацию о каждом заказе. Содержит общую сумму заказа и внешний ключ к событию.

CREATE TABLE Orders (

id INT PRIMARY KEY IDENTITY,

event\_id INT NOT NULL,

event\_date DATETIME NOT NULL,

equal\_price int(11) NOT NULL,

created DATETIME DEFAULT GETDATE(),

FOREIGN KEY (event\_id) REFERENCES Events(id)

);

Создать таблицу билетов. Эта таблица позволяет хранить различные типы билетов для каждого заказа. Это обеспечивает гибкость для добавления новых типов билетов в будущем.

CREATE TABLE Tickets (

id INT PRIMARY KEY IDENTITY,

order\_id INT NOT NULL,

ticket\_type VARCHAR(50) NOT NULL,

price DECIMAL(11, 2) NOT NULL,

quantity INT NOT NULL,

barcode VARCHAR(120) UNIQUE NOT NULL,

FOREIGN KEY (order\_id) REFERENCES Orders(id)

);

Эти изменения позволяют:

* Сохранять разные типы билетов (взрослые, детские, льготные, групповые) в одной таблице.
* Легко добавлять новые типы билетов или специальных предложений без изменения структуры базы данных.
* Улучшать производительность запросов, так как не все информация о заказе хранится в одной таблице.
* Уменьшить дублирование данных и улучшить целостность базы данных.

Такое нормализованное представление позволяет легко расширять структуру базы данных в будущем, добавляя новые типы билетов или специальных предложений без необходимости изменять существующие таблицы.

2. Часто посетители из одного заказа приходят не одновременно на события. Возникает необходимость проверять их билеты по отдельности. Для этого у каждого билета должен быть свой баркод. Если в одном заказе было куплено несколько билетов, 2 взрослых, 3 детских, 4 льготных - то должно быть 9 баркодов для каждого билета соответственно. Задача - Нормализовать таблицу, учитывая что у каждого билета свой баркод, показать конечный вид таблицы. Объяснить свое решение.

Для решения этой задачи я так же предлагаю сделать две таблицы.

CREATE TABLE Orders (

id INT PRIMARY KEY IDENTITY,

event\_id INT NOT NULL,

event\_date DATETIME NOT NULL,

equal\_price int(11) NOT NULL,

created DATETIME DEFAULT GETDATE(),

FOREIGN KEY (event\_id) REFERENCES Events(id)

);

CREATE TABLE Tickets (

id INT PRIMARY KEY IDENTITY,

order\_id INT NOT NULL,

ticket\_type VARCHAR(50) NOT NULL,

price DECIMAL(11, 2) NOT NULL,

quantity INT NOT NULL,

barcode VARCHAR(120) UNIQUE NOT NULL,

FOREIGN KEY (order\_id) REFERENCES Orders(id)

);

Таблица Orders: Эта основная таблица хранит общую информацию о заказе и связь с событием.

Таблица Tickets: Эта таблица хранит информацию о каждом отдельном билете в заказе. Каждый билет имеет свой уникальный barcode.

Каждый заказ может содержать несколько билетов разного типа (взрослые, детские, льготные, групповые).

Внешний ключ order\_id в таблице Tickets для связи с основной таблицей Orders.

Поле barcode в таблице Tickets является уникальным для каждого билета в заказе, что обеспечивает уникальность каждого билета.

Это решение позволяет легко добавлять новые типы билетов в будущем без изменения структуры базы данных.

Каждый билет имеет свой собственный barcode, что позволяет проверять их по отдельности.

Это решение обеспечивает нормализацию данных и улучшает производительность запросов.

Такое нормализованное представление позволяет легко расширять структуру базы данных в будущем, добавляя новые типы билетов или специальных предложений без необходимости изменять существующие таблицы. Оно также обеспечивает уникальность каждого билета в заказе и позволяет проверять их по отдельности.