

# Практика 2

## Физическое проектирование БД

# Ход занятия

Рассмотрим особенности физического проектирования и какие нюансы не были учтены в прошлой практической.

# Цель физ. проектирования

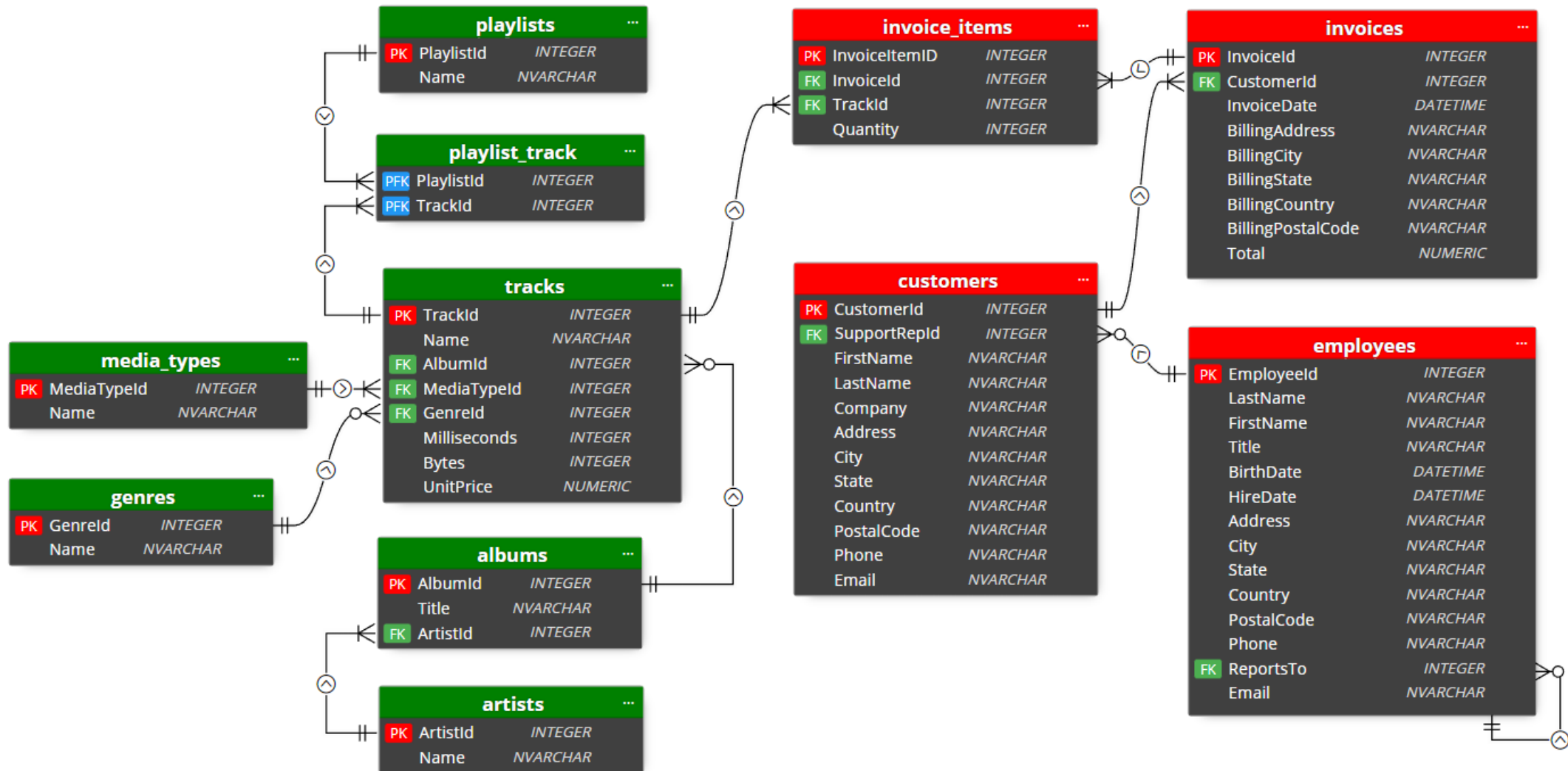
- Создание базовой функциональной схемы реляционной базы данных на основе глобальной логической модели данных, которая может быть реализована в целевой СУБД.
- Если на стадии логического проектирования отвечают на вопрос **что** должно быть сделано, то на стадии физического проектирования отвечают на вопрос **как** это должно быть сделано.

# Физ. проектирование

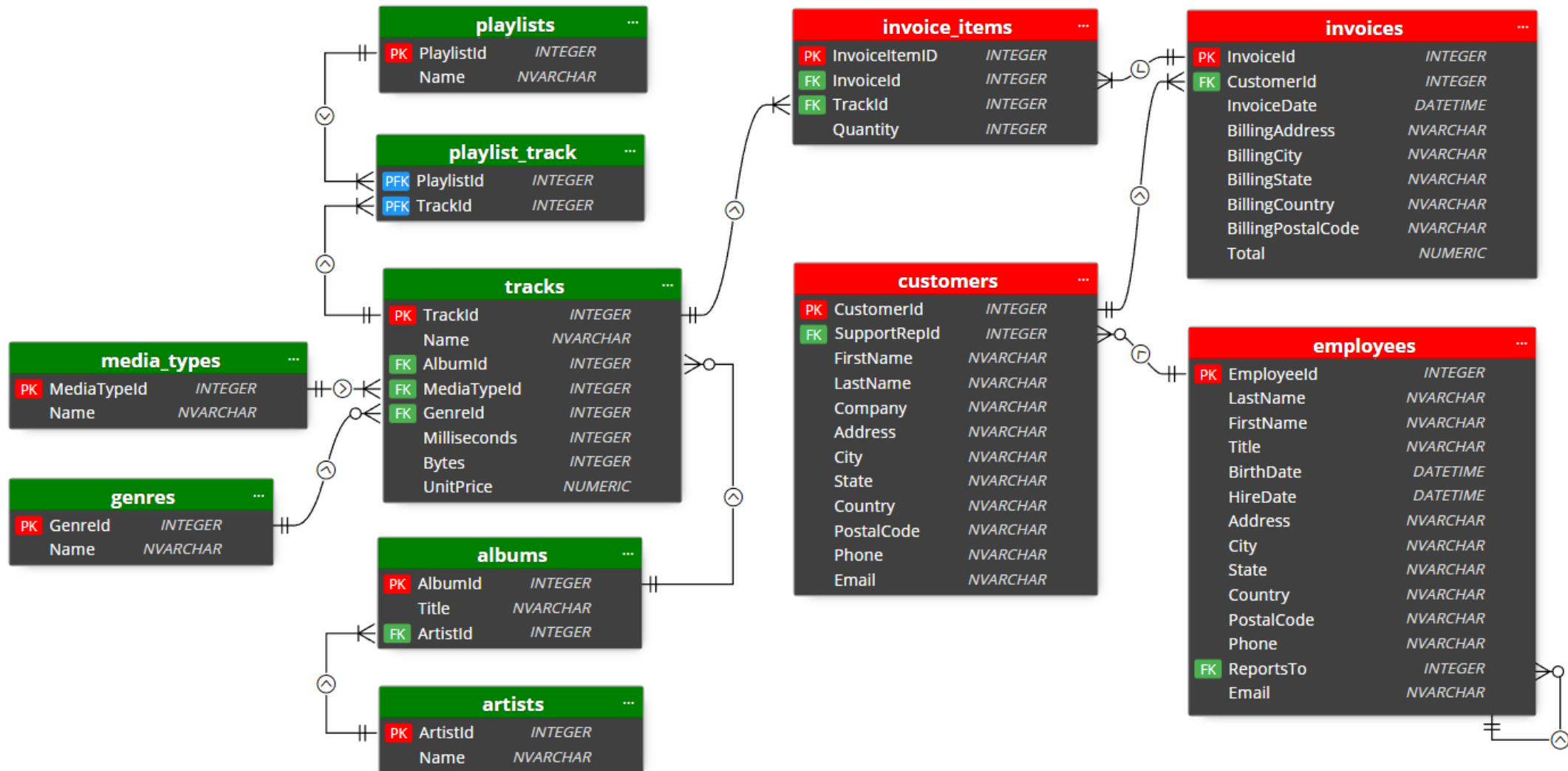
Для этого необходимо последовательно спроектировать:

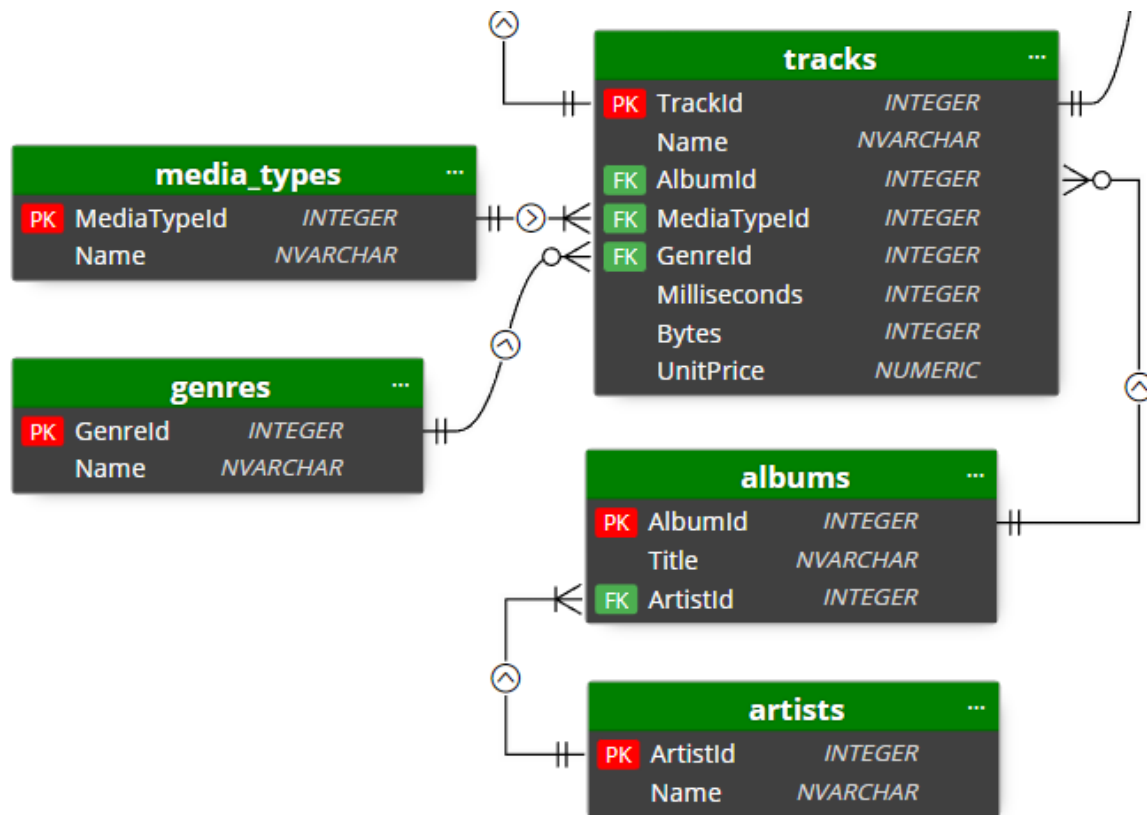
- базовые отношения (CREATE TABLE)
- представление производных данных
- общие ограничения

# Пример даталогической модели



# Чего не хватает для реализации?





```

CREATE TABLE tracks (
  TrackId INTEGER PRIMARY KEY,
  Name VARCHAR(?) NOT NULL,
  AlbumId INTEGER NOT NULL,
  MediaTypeId INTEGER NOT NULL,
  GenreId INTEGER NOT NULL,
  Milliseconds INTEGER DEFAULT (?),
  Bytes INTEGER,
  UnitPrice NUMERIC(10,2) NOT NULL,
  FOREIGN KEY (AlbumId) REFERENCES
  albums(AlbumId) ???,
  FOREIGN KEY (GenreId) REFERENCES
  genres(GenreId) ???,
  FOREIGN KEY (MediaTypeId) REFERENCES
  media_types(MediaTypeId) ???
)
  
```

# Про значения данных

Необходимо понимать, какие ограничения на значения следует (или **не следует!**) реализовывать

- <https://habr.com/ru/post/431866/>
- <https://habr.com/ru/post/279751/>
- <https://habr.com/ru/companies/hflabs/articles/260601/>

| employees |            |          |
|-----------|------------|----------|
| PK        | EmployeeId | INTEGER  |
|           | LastName   | NVARCHAR |
|           | FirstName  | NVARCHAR |
|           | Title      | NVARCHAR |
|           | BirthDate  | DATETIME |
|           | HireDate   | DATETIME |
|           | Address    | NVARCHAR |
|           | City       | NVARCHAR |
|           | State      | NVARCHAR |
|           | Country    | NVARCHAR |
|           | PostalCode | NVARCHAR |
|           | Phone      | NVARCHAR |
| FK        | ReportsTo  | INTEGER  |
|           | Email      | NVARCHAR |



# Виды операций

- **Каскадное удаление/обновление**

- При удалении записи из родительской таблицы автоматически удаляются/обновляются все относящиеся к ней записи из дочерней таблицы.

- **Установка пустых ключей или default-значений**

- При удалении записи из родительской таблицы автоматически старое значение во внешнем ключе всех соответствующих записей дочерней таблицы меняется на NULL/DEFAULT.

- **Запрет операции**

- СУБД не позволит удалить из родительской таблицы запись (или изменить значение её первичного ключа), значение ПК которой присутствует во внешнем ключе хотя бы одной записи дочерней таблицы.

# Виды операций

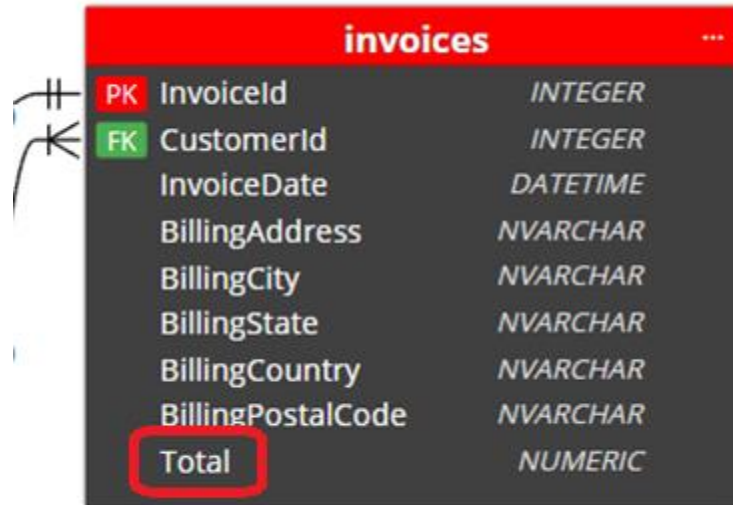
- Определение каждого выделенного в глобальной логической модели данных отношения включает следующие элементы:
- имя отношения
- список простых атрибутов, заключенный в круглые скобки
- определение первичного ключа и внешних ключей
- список производных атрибутов и описание способов их вычисления;
- определение требований ссылочной целостности для любых внешних ключей.

# Производные атрибуты

– атрибуты, значения которых можно определить с использованием значений других атрибутов.

Необходимо определить, должен ли производный атрибут храниться в базе данных или вычисляться каждый раз, когда в нем возникает необходимость.

# На примере



| invoices |                   |          |
|----------|-------------------|----------|
| PK       | InvoiceId         | INTEGER  |
| FK       | CustomerId        | INTEGER  |
|          | InvoiceDate       | DATETIME |
|          | BillingAddress    | NVARCHAR |
|          | BillingCity       | NVARCHAR |
|          | BillingState      | NVARCHAR |
|          | BillingCountry    | NVARCHAR |
|          | BillingPostalCode | NVARCHAR |
|          | Total             | NUMERIC  |

Поле Total в таблице Invoices

- необходимо обновлять при изменении (как?)

+ если бы Total не хранился в Invoices, то его значение приходилось бы вычислять каждый раз, когда оно потребуется

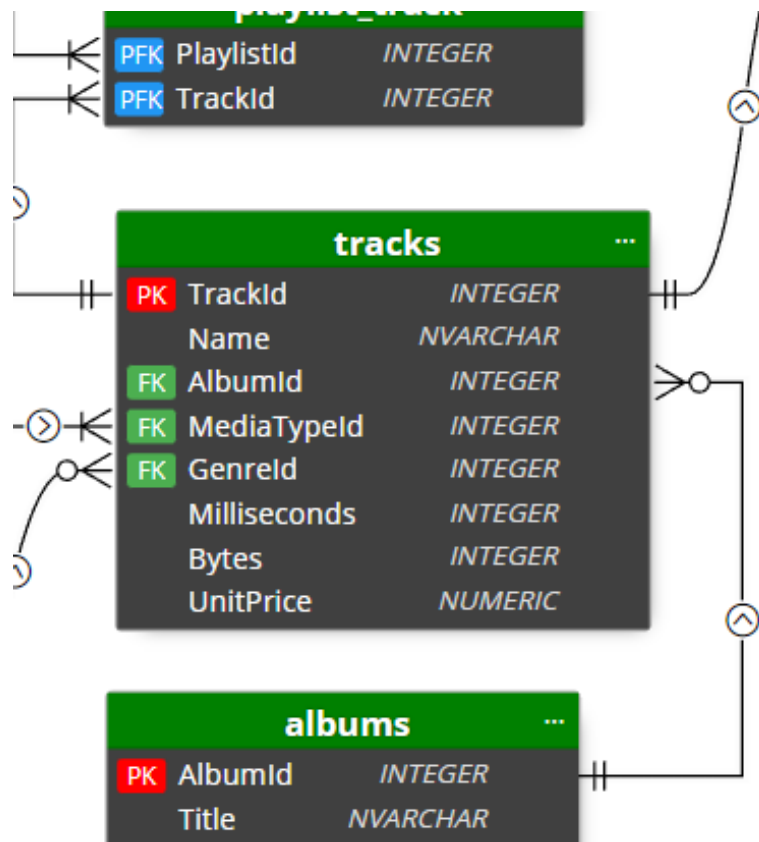
# Общие ограничения

Обновление информации в таблицах может регламентироваться ограничениями предметной области, регулирующими выполнение операций обновлений.

Примеры:

- Total не может быть меньше 10 (CHECK)
- В плейлисте до 100 треков (?)

# Про индексы



Основной вопрос: будет ли добавление индексов способствовать повышению производительности системы?

```
CREATE INDEX idx_t_name ON tracks (name);
```

```
CREATE INDEX idx_t_name_album ON tracks  
(name, albumId);
```

```
DROP INDEX idx_trackname;
```

- Ничего не мешает сделать потом/исправить, посмотрев планы запросов и тд

# Денормализация

Иногда оказывается, что нормализованная модель не обеспечивает максимальной производительности при обработке данных.

Необходимо учесть следующие особенности:

- необходимо продумать технологию поддержания целостности данных
- денормализация может ускорить чтение данных, но при этом замедлить обновление записей