Оглавление

[Проектирование модели БД 2](#_Toc177055615)

[Проектирование NOSQL модели 2](#_Toc177055616)

[Проектирование SQL модели 4](#_Toc177055617)

# Проектирование модели БД

## Проектирование NOSQL модели

Используем в качестве нереляционной БД MongoDB. MongoDB работает не с таблицами, а с коллекциями и документами.

Какие данные необходимо хранить:

Данные о пользователях: имя, логин, пароль, email, телефон, дату регистрации, а также id\_товаров, которые добавил в «Желаемое», личные данные, такие как ДР и пол

Данные об оформленных заказах: хранится id\_пользователя, дата заказ, оплачен ли, вручен ли, и возможно неограниченное множество элементов: id\_товара их количество и цена, по которой купили

Данные о товарах: имя, страна производитель, цена, количество на складе, а также id\_пользователей, которые добавили в желаемое. У каждого товара, в зависимости от типа, будут дополнительные данные. А также подтипы, вроде отдельных характеристик для спортивного снаряжения.

Первый вариант проектирования для данных о товарах:

Все данные хранятся в одной коллекции, где тип и подтип прописан в отдельном атрибуте.

Плюсы:

* Легко создавать новые типы товаров
* Всего одна коллекция (а значит уменьшается количество подключений на программном уровне)
* Раскрываются преимущества нереляционной БД, когда у всех объектов разные атрибуты

Минусы:

* Очень большой объем данных, разных типов в одной коллекции, плюс еще существуют подтипы для типов, может привести к путанице данных (а может не привести)
* Из большого объема данных следует, что индексирование и поиск по данным будет занимать больше времени

Нейтрально: можно отказаться от подтипов и просто создать множество типов

Второй вариант проектирования для данных о товарах:

Данные о товарах, где у каждого типа товаров своя коллекция. А подтип прописан как отдельный атрибут.

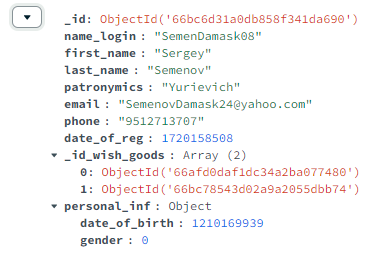
Плюсы:

* Данные о товарах расположены более структурированы, что уменьшает количество переменных для запросов и уменьшает необходимость в агрегированных запросах
* Из-за малого объема данных в каждой коллекции, уменьшается время на запросы + (индексирование и поиск)
* Более автономный при утрате части данных

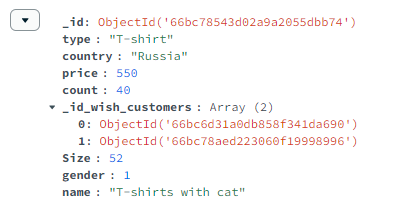
Минусы:

* Тяжело вводить новые типы товаров, для этого придется создавать новую коллекцию
* Необходимо будет делать множество запросов для вывода всего ассортимента или вывода элементов в корзине

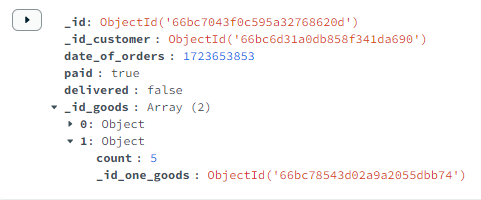
После некоторого раздумья был принят первый вариант, в конфигурации, где отсутствуют подтипы. Стоит также сказать, что администратору будет необходимо более трепетно подходить к заполнению данных, так как ошибка в названии типа, может создать дублирующие группы. Пример: «Sneaker», «Sneakers». Это можно обезопасить созданием отдельной БД, которая будет выступать в качестве словаря, где будут хранится данные формата: {1: protein}, {2: Bag}, {3: Sneakers}. Однако это заставит производить дополнительный запрос, для расшифровки айдишников. (С другой стороны можно хранить в кеше на сервере).



Пример документа в коллекции с пользователями (онлайн-покупателями)



Пример документа в коллекции с товаром

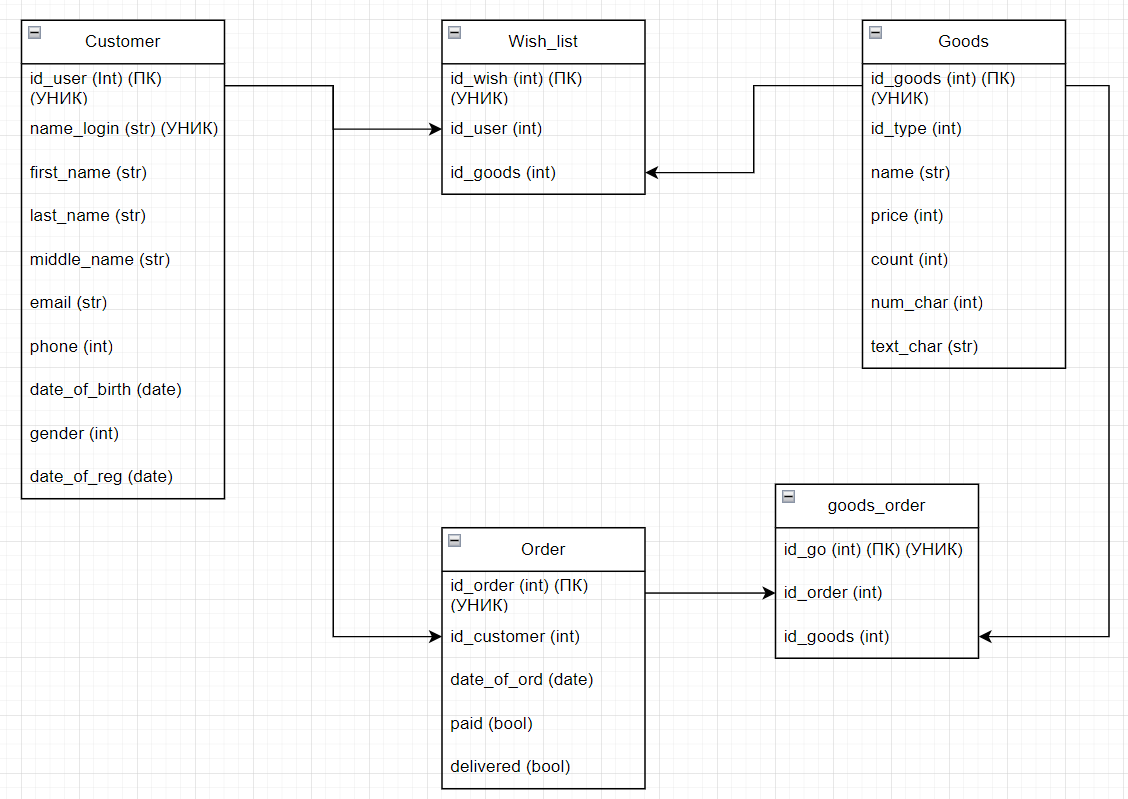


Пример документа в коллекции с заказом

## Проектирование SQL модели

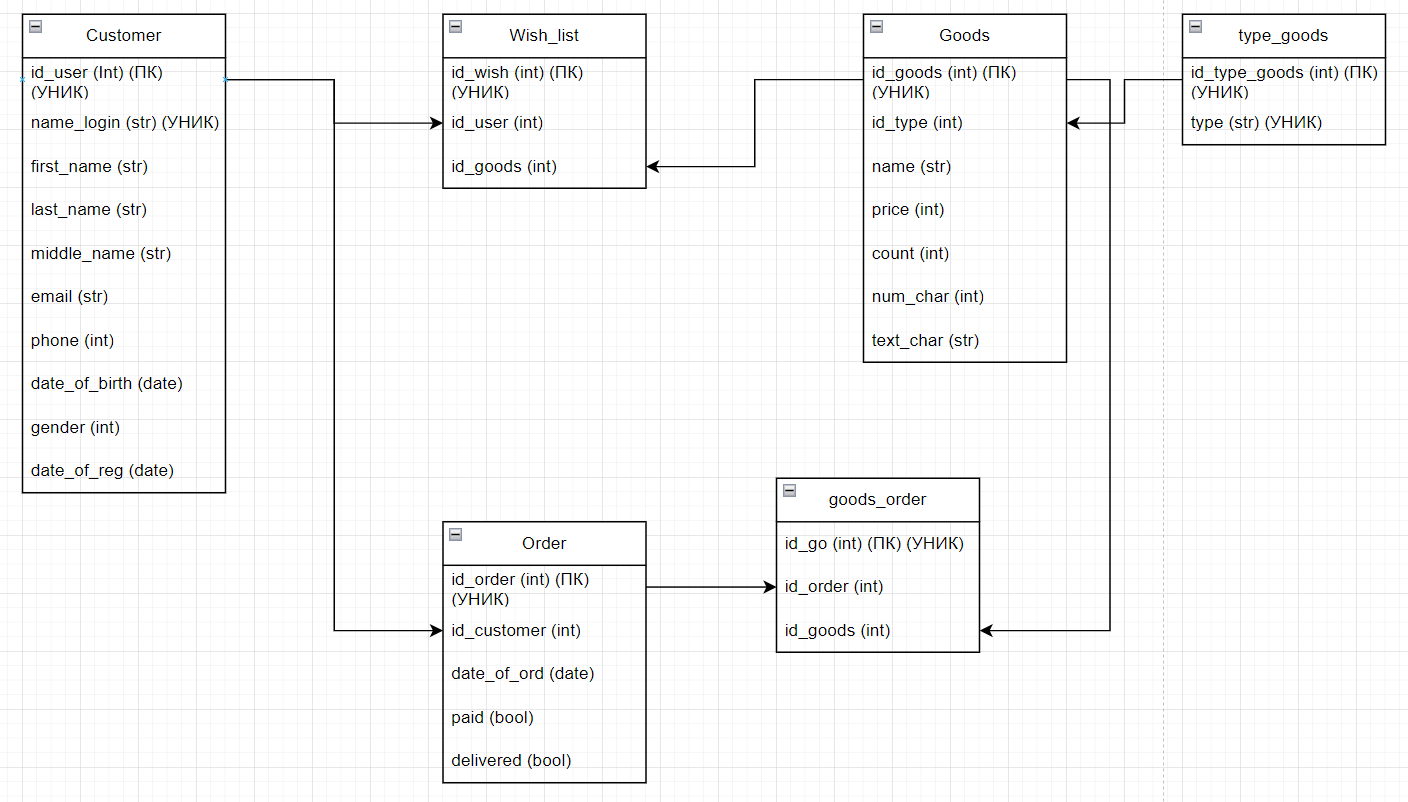
Также был спроектирован вариант с использованием реляционных баз данных

Были созданы 3 сущности (Пользователь, товары, заказы), и связи между ними, один ко многим между Покупателем и Заказом, и связь многое ко многим между покупателем и товаром (для отслеживания желаемых товаров) а также между заказом и товаром. Связь многое ко многим была реализована через промежуточные (связующие) таблицы.



При проектирование реляционной базы, сталкиваемся с той же проблемой, что и в нереляционной БД, а именно что в одной таблице хранятся данные, с абсолютно разными атрибутами. В таблице с товарами, у нас вместе с гантелями хранятся мячи, тренажеры, и т.д. И каждый тип имеет разные характеристики. Это можно исправить, создавая для каждого типа свои, используемые лишь этим типом, колонки, вроде массы для гантели. Но тогда получается ситуация, где в таблице становится бесконечно много колонок. Было принято решение ограничиться созданием для каждого элемента 2-ух дополнительных колонок, в каждой из которых будут хранится числовые или текстовые характеристики. По типу объема для бутылки воды, и её цвета, или размер для обуви/одежды и её цвет.

Отдельно можно отметить, что создание словаря для типов может уменьшить риск случайных ошибок.



Создание одной таблицы/коллекции для хранения усложняет администрирование. А именно изменение, добавление, удаление данных. А случайные ошибки, вероятность которых возрастает, при условии использования такого проекта БД, могут критически нарушить целостность данных.

В качестве основной БД использовался MongoDB. Главным фактором было более удобное хранение данных о товарах чем в реляционном аналоге.