Y.Market

Ключевая проблема: доставка и хранение

Этапы, чтобы товар дошел до пользователя:

1. **Поставка на склад**
2. Прием товара на складе
3. **Хранение товара на складе**
4. Сбор и обработка заказа на складе
5. Упаковка заказа на складе
6. **Доставка заказа** до сортировочных центров
7. Доставка заказа до конечного клиента

Три основные схемы, по которым работает marketplace:

1. FBO (Fulfilment by Operator) - все делает маркет
2. FBS (Fulfilment by Seller) - до заказа товара он хранится у магазина
3. DBS (Delivery by Seller) - все делает магазин, маркет только “точка входа и покупка товара”

Рассмотрим первый вариант.

Склад: для каждого склада добавить n-нное количество полок.

Доставка: в пункт выдачи/на дом. Курьеры, водители, водители партнеров.

Если магазин неподалеку, то отправляется водителями напрямую.

Таблица недобросовестных продавцов.

Яндекс-Маркет - крупный маркетплейс, так что поставим более узкий вопрос: как товар добирается от точки А, когда пользователь нажимает кнопку заказать в приложении или на сайте, до точки Б, когда все тот же пользователь забирает его в ближайшем пункте выдачи или получает посылку от курьера.

1. пользователь делает за раз заказ в приложении/на сайте на n продуктов, выбирая пункт выдачи или вводя адрес места жительства;
2. происходит разбиение на n заказов и каждый из них попадает в таблицу “Orders”;
3. Рассмотрим один из этих n заказов. В зависимости от продавца идут три случая:
    a. FBY (Fulfilment by Yandex) - Яндекс делает все от принятия заказа и его хранения на собственных складах до его доставки пользователю;
    b. FBS (Fulfilment by Seller) - Яндекс производит принятия заказа от пользователя и его доставку, сами товары хранятся на складах продавца;
    c. DBS (Delivery by Seller) - Яндекс осуществляет только принятие заказа, остальное делает сам продавец.
4. попав на один из случаев, информация о заказе передается на соответствующий склад:
    a. в первом случае (FBY) мы запрашиваем для нашего заказа информацию о нем: что за товар и куда везти (в пункт выдачи или на определенный адрес) - после мы обращаемся к таблице “Fulfilment center”, в которой *условный алгоритм* подбирает ближайший склад к месту доставки, на котором есть в наличии требуемый нам товар;

    b. во втором и третьем - мы запрашиваем только информацию о самом заказе.

5. в конце концов, получив место нахождения товара и нужное место его доставки, мы находим ближайшего к складу свободного водителя (если речь о пункте выдачи) или курьера (если речь о доставке на дом в пределах города) в табличке “Delivery”.

К сожалению, это сильно упрощенная схема, в реальности все в разы сложнее.

```
CREATE TABLE users (
  id BIGSERIAL NOT NULL PRIMARY KEY,
  first_name VARCHAR(20) NOT NULL,
  last_name VARCHAR(20) NOT NULL,
  gender CHARACTER(1) NOT NULL CHECK (gender = 'M' OR gender
= 'F'),
  date_of_birth DATE NOT NULL,
  email VARCHAR(100) NOT NULL UNIQUE,
  phone_number VARCHAR(16) NOT NULL,
  bank_card_number CHARACTER(16) NOT NULL
);

CREATE TABLE orders (
  id BIGSERIAL NOT NULL PRIMARY KEY,
  user_id BIGSERIAL NOT NULL,
  product_id BIGSERIAL NOT NULL,
  delivery_id BIGSERIAL NOT NULL,
  FOREIGN KEY (user_id) REFERENCES users (id),
  FOREIGN KEY (product_id) REFERENCES products (id),
  FOREIGN KEY (delivery_id) REFERENCES delivery (id)
);
```

```
CREATE TABLE products (
  id BIGSERIAL NOT NULL PRIMARY KEY,
  seller_id BIGSERIAL NOT NULL,
  product_name VARCHAR(200) NOT NULL,
  description VARCHAR(10000) NOT NULL,
  price NUMERIC(12,2) NOT NULL CHECK (price > 0.0),
  rating NUMERIC(5,1) NOT NULL
            CHECK (rating >= 0.0 AND rating <= 5.0),
    FOREIGN KEY (seller_id) REFERENCES sellers (id)
);

CREATE TABLE delivery (
  id BIGSERIAL NOT NULL PRIMARY KEY,
  address_of_pick_up_point VARCHAR(100) NOT NULL,
  delivery_man_first_name VARCHAR(20) NOT NULL,
  delivery_man_last_name VARCHAR(20) NOT NULL,
  delivery_man_licence VARCHAR(16) NOT NULL,
    EXCLUDE (delivery_man_licence WITH =)
);

CREATE TABLE sellers (
  id BIGSERIAL NOT NULL PRIMARY KEY,
  first_name VARCHAR(20) NOT NULL,
  last_name VARCHAR(20) NOT NULL,
  gender CHARACTER(1) NOT NULL CHECK (gender = 'M' OR gender
= 'F'),
  date_of_birth DATE NOT NULL,
  email VARCHAR(100) NOT NULL UNIQUE,
  phone_number VARCHAR(16) NOT NULL,
  fulfilment_center_address VARCHAR(100) NOT NULL
);


INSERT INTO users (
    first_name,
    last_name,
```

```
    gender,
    date_of_birth,
    email,
    phone_number,
    bank_card_number)
VALUES ('Vadim', 'Dmitriyev', 'M', '1995-04-13', 'Dmitriyev@s
omeone.com', '+79043373443', '1234567812345678');

INSERT INTO users (
  first_name,
  last_name,
  gender,
  date_of_birth,
  email,
  phone_number,
  bank_card_number)
VALUES ('Pasha', 'Likhosherstov', 'M', '2003-04-13', 'Likhosh
erstov@nepapa.com', '+79043365432', '1256334412563344');

INSERT INTO users (
  first_name,
  last_name,
  gender,
  date_of_birth,
  email,
  phone_number,
  bank_card_number)
VALUES ('Afanasy', 'Vakin', 'M', '2004-04-13', 'Vakin@nepapa.
com', '+79043382454', '1346895413468954');

INSERT INTO users (
  first_name,
  last_name,
  gender,
  date_of_birth,
  email,
```

```
  phone_number,
  bank_card_number)
VALUES ('Ksenia', 'Kosterova', 'F', '2005-04-13', 'Kosterova@
sister.com', '+79043391465', '4539875445398754');

INSERT INTO users (
  first_name,
  last_name,
  gender,
  date_of_birth,
  email,
  phone_number,
  bank_card_number)
VALUES ('Irina', 'Sorokina', 'F', '2004-04-13', 'Sorokina@nem
ama.com', '+79043400476', '5943875459438754');

INSERT INTO users (
  first_name,
  last_name,
  gender,
  date_of_birth,
  email,
  phone_number,
  bank_card_number)
VALUES ('Lada', 'Batalova', 'F', '2003-04-13', 'Batalova@nema
ma.com', '+79043419487', '4353455343534553');

INSERT INTO users (
  first_name,
  last_name,
  gender,
  date_of_birth,
  email,
  phone_number,
  bank_card_number)
VALUES ('Anya', 'Kavunova', 'F', '2004-04-13', 'Kavunova@mam
```

```
a.com', '+79043428498', '5930475035453934');

INSERT INTO users (
  first_name,
  last_name,
  gender,
  date_of_birth,
  email,
  phone_number,
  bank_card_number)
VALUES ('Nastya', 'Meshkova', 'F', '2004-04-13', 'Meshkova@ne
mama.com', '+79043437409', '0359103048572434');

INSERT INTO users (
  first_name,
  last_name,
  gender,
  date_of_birth,
  email,
  phone_number,
  bank_card_number)
VALUES ('Artem', 'Brandy', 'M', '2004-04-13', 'Brandy@steppap
a.com', '+79043446410', '0423857843901849');

INSERT INTO users (
  first_name,
  last_name,
  gender,
  date_of_birth,
  email,
  phone_number,
  bank_card_number)
VALUES ('Rasul', 'Abdurashidov', 'M', '2004-04-13', 'Abdurash
idov@nepapa.com', '+79043455421', '0423857843943849');

INSERT INTO users (
```

```
  first_name,
  last_name,
  gender,
  date_of_birth,
  email,
  phone_number,
  bank_card_number)
VALUES ('Mikhail', 'Eremin', 'M', '2004-04-13', 'Eremin@nepap
a.com', '+79043464432', '0420007843943849');

INSERT INTO users (
  first_name,
  last_name,
  gender,
  date_of_birth,
  email,
  phone_number,
  bank_card_number)
VALUES ('Tatyana', 'Mosina', 'F', '2004-04-13', 'Mosina@net.c
om', '+79043473443', '0420107843943848');

INSERT INTO users (
  first_name,
  last_name,
  gender,
  date_of_birth,
  email,
  phone_number,
  bank_card_number)
VALUES ('Sasha', 'Vinogradov', 'M', '1980-10-04', 'Vinogradov
@chel.com', '+79653473443', '0420107843945658');

INSERT INTO users (
  first_name,
  last_name,
  gender,
```

```
  date_of_birth,
  email,
  phone_number,
  bank_card_number)
VALUES ('Nastya', 'Morozova', 'F', '2002-07-28', 'Nastya@disc
rete.math', '+79653573444', '0420107840010658');

INSERT INTO users (
  first_name,
  last_name,
  gender,
  date_of_birth,
  email,
  phone_number,
  bank_card_number)
VALUES ('Akim', 'Yun', 'M', '2004-03-01', 'Akim@geometry.roo
f', '+79653673445', '0420107840020658');

users (15) : products (17 / 2) : delivery (9)
1 - 1 - 1
2 - 1 - 3
2 - 5 - 3
3 - 6 - 4
4 - 3 - 2
4 - 4 - 2
5 - 7 - 5
6 - 7 - 5
7 - 8 - 6
8 - 9 - 6
9 - 5 - 6
10 - 10 - 7
10 - 1 - 7
11 - 11 - 8
11 - 14 - 8
12 - 13 - 8
13 - 12 - 9
```

```
14 - 15 - 9
15 - 16 - 9

1.
вывести пользователей мужского пола с годом рождения раньше 2
004:
SELECT first_name, last_name, gender,
EXTRACT(YEAR FROM date_of_birth) AS year_of_birth FROM users
WHERE gender='M' AND EXTRACT(YEAR FROM date_of_birth) < 2004;
2.
вывести товар, который стоит дешевле 1000, о отсортировать ег
о
по рейтингу в порядке возрастания:
SELECT product_name, rating FROM products
WHERE price < 1000.0
GROUP BY product_name, rating
ORDER BY rating;
3.
вывести информацию о заказах, содержащую id пользователей
и id товара, для курьеров под id = 8 или 9:
SELECT user_id, product_id FROM orders
WHERE delivery_id=8 OR delivery_id=9
ORDER BY delivery_id DESC;
Оптимизация:
CREATE INDEX
ON orders (delivery_id);
4.
Сначала введем вспомогательные VIEW:
CREATE VIEW orderBySellerID AS
SELECT seller_id, COUNT(product_name) AS num_of_products
FROM products
GROUP BY seller_id;
После можем перейти к самому запросу:
SELECT seller_id, num_of_products FROM orderBySellerID
ORDER BY seller_id;
ИЛИ в более короткой форме:
```

```
SELECT seller_id, COUNT(product_name) AS num_of_products
FROM products
GROUP BY 1
ORDER BY 1;



1.
возьмем склад "St. Petersburg, Pargolovo settlement, 61 Podgo
rnaya str.,
building 1, building 5" и выведем всю информацию о хранящихся
там товарах
и их продавце:
SELECT products.seller_id, sellers.first_name, sellers.last_n
ame,
products.product_name, products.price, products.rating
FROM products JOIN sellers ON products.seller_id = sellers.id
WHERE sellers.fulfilment_center_address=
'St. Petersburg, Pargolovo settlement, 61 Podgornaya str., bu
ilding 1, building 5'
ORDER BY products.rating;
2.
вывести названия товаров и количество раз, которое он сейчас
заказан
SELECT products.product_name, COUNT(*) AS num_of_orders
FROM orders JOIN products ON orders.product_id = products.id
GROUP BY 1
HAVING COUNT(*) > 1
ORDER BY 2;
3.
вывести всю информацию о товаре из таблички products и
адрес склада, на котором хранится этот товар, из таблички sel
lers
SELECT products.product_name, products.description,
products.price, products.rating, sellers.fulfilment_center_ad
dress
FROM products JOIN sellers ON
```

```
products.seller_id = sellers.id
ORDER BY 5;


1.
вывести всю основную информацию о заказах, у которых склад, н
а котором они
храняться “Saint Petersburg, Pushkinsky district, Shushary se
ttlement,
Moskovskoe shosse, 70, building 4”
SELECT users.first_name AS user_first_name, users.last_name A
S user_last_name,
users.phone_number AS user_phone, products.product_name,
sellers.first_name AS seller_first_name, sellers.last_name AS
seller_last_name,
sellers.fulfilment_center_address
FROM orders JOIN users ON orders.user_id = users.id
JOIN products ON orders.product_id = products.id
JOIN sellers ON products.seller_id = sellers.id
WHERE fulfilment_center_address =
'Saint Petersburg, Pushkinsky district, Shushary settlement,
Moskovskoe shosse, 70, building 4'
ORDER BY 1;
2.
подсчитать количество заказов в каждом ценовом промежутке с ш
агом в 1000
SELECT cost.min_sum, cost.max_sum, count(orders.*) AS number_
of_orders
FROM orders
JOIN products ON orders.product_id = products.id
RIGHT OUTER JOIN
(VALUES (0, 1000), (1000, 2000), (2000, 3000), (3000, 4000),
        (4000, 5000), (5000, 6000), (6000, 7000),
                (7000, 8000), (8000, 9000)
) AS cost (min_sum, max_sum)
ON products.price >= cost.min_sum AND products.price < cost.m
```

```
ax_sum
GROUP BY cost.min_sum, cost.max_sum
ORDER BY cost.min_sum;
3.
выведим таблицу сопоставляющую адреса складов с адресами пунк
тов выдачи,
на которые приходят заказы с первых
SELECT sellers.fulfilment_center_address, delivery.address_of
_pick_up_point
FROM orders JOIN delivery ON orders.delivery_id = delivery.id
JOIN products ON orders.product_id = products.id
JOIN sellers ON products.seller_id = sellers.id
GROUP BY 1, 2
ORDER BY 1, 2;

CREATE VIEW center_pick_up_point AS
SELECT sellers.fulfilment_center_address, delivery.address_of
_pick_up_point
FROM orders JOIN delivery ON orders.delivery_id = delivery.id
JOIN products ON orders.product_id = products.id
JOIN sellers ON products.seller_id = sellers.id;

SELECT * FROM center_pick_up_point
GROUP BY 1, 2
ORDER BY 1, 2;

Оптимизация:
CREATE INDEX
ON sellers (fulfilment_center_address);
подсчитаем, сколько приходится пунктов выдачи на каждый склад
SELECT sellers.fulfilment_center_address,
COUNT(delivery.address_of_pick_up_point)
FROM orders JOIN delivery ON orders.delivery_id = delivery.id
JOIN products ON orders.product_id = products.id
JOIN sellers ON products.seller_id = sellers.id
GROUP BY 1;
```

```
(вывести таблицу пользователей и продавцов, у которых совпада
ют две последние
цифры номера телефона)
вывести на каждый товар количество курьеров, которым надо дос
тавить этот товар
SELECT delivery.first_name, delivery.last_name,
COUNT(*) AS num_of_deliveries, products.product_name
FROM delivery LEFT JOIN products
ON delivery. = products.
GROUP BY 4;
```

1. вывести только те строчки orders, у которых адрес “A”, то есть берем информацию через products в sellers
2.

```
SELECT ('столбцы или * для выбора всех столбцов; обязательно’)
FROM ('таблица; обязательно’)
WHERE ('условие/фильтрация, например, city = 'Moscow'; необязательно’)
GROUP BY ('столбец, по которому хотим сгруппировать данные; необязательно’)
HAVING ('условие/фильтрация на уровне сгруппированных данных; необязательно’)
ORDER BY ('столбец, по которому хотим отсортировать вывод; необязательно')
```

WHERE - используется IN / NOT IN для фильтрации столбца по нескольким значениям, AND / OR для фильтрации таблицы по нескольким стобцам.

Например, WHERE Country=’Germany’ AND City NOT IN (‘Berlin’, ‘Aachen’)

(более подробная информация во 2-й презентации на 26 слайде)

3. pg_dump

В PostgreSQL есть встроенный инструмент для создания резервных копий — утилита pg_dump. Утилита имеет простой синтаксис:

```
# pg_dump <параметры> <имя базы> > <файл для сохранения копии>
```

В простейшем случае достаточно указать имя базы данных, которую в дальнейшем нужно будет восстановить. Резервная копия создается следующей командой:

```
# pg_dump zabbix > /tmp/zabbix.dump
```