

Understanding Database

Group 21



yifei.huang-3@postgrad.manchester.ac.uk [ID11597876]
huijin.he@postgrad.manchester.ac.uk [ID11531666]
yinhang.zhang@postgrad.manchester.ac.uk [ID11613159]
jingzhou.shi@postgrad.manchester.ac.uk [ID14104388]
zihan.zhang-21@postgrad.manchester.ac.uk [ID14142055]
rahmah.rifat@postgrad.manchester.ac.uk [ID10534099]
tania.diamanta@postgrad.manchester.ac.uk [ID11585796]

Deliverables



Task allocation between group members;



NoSQL database design showcase

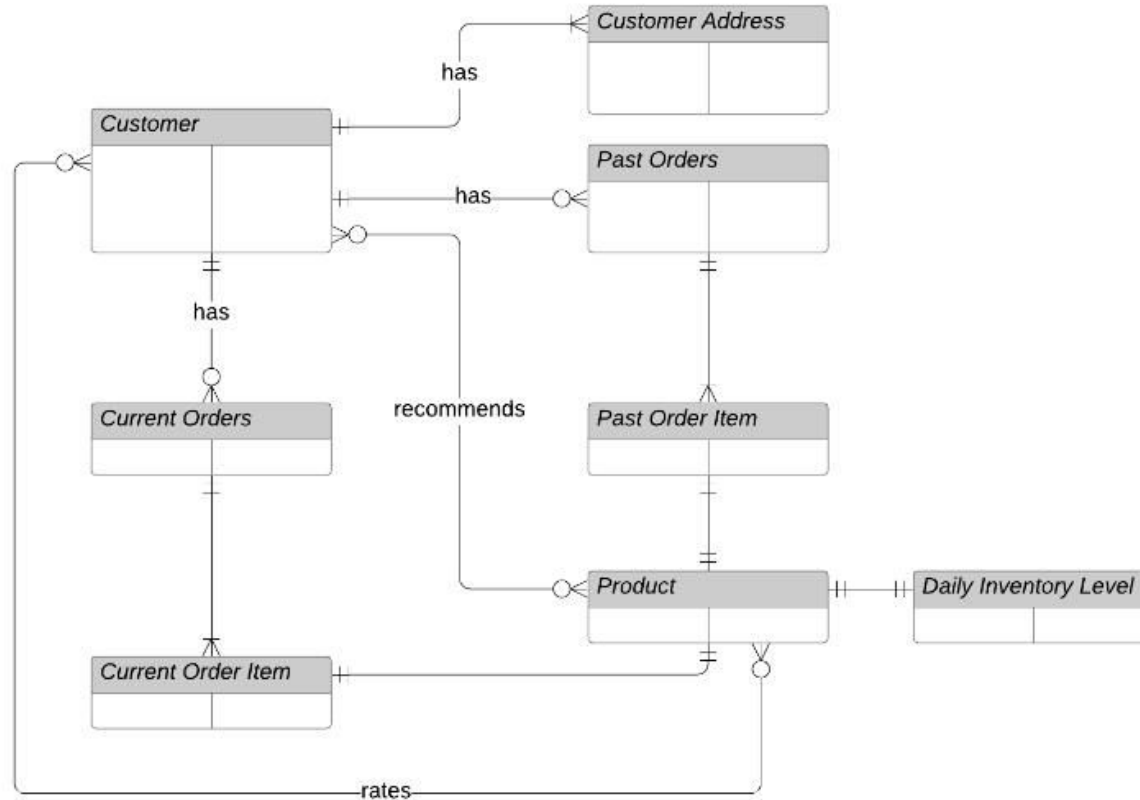


Design decisions, design patterns, operations assumptions

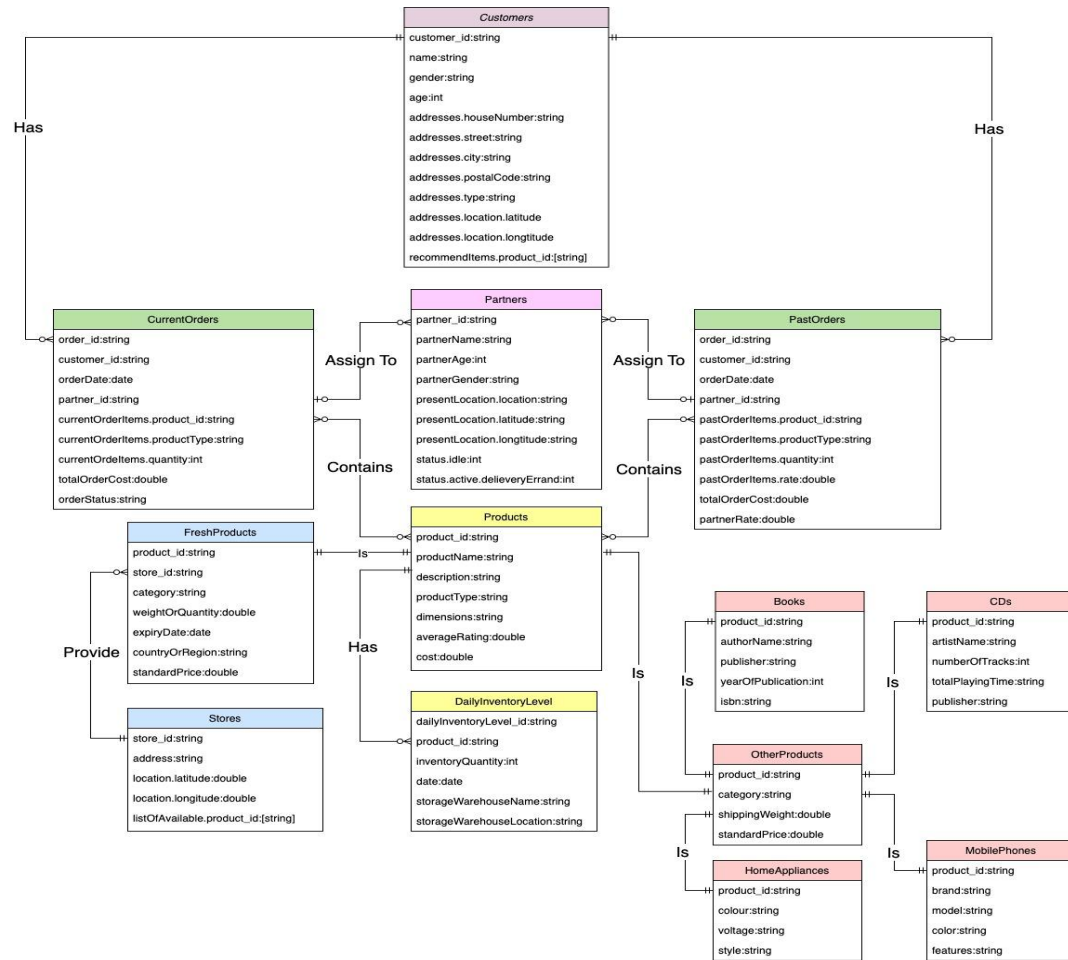


Queries design decisions.

[Original] NoSQL database design



[Updated] NoSQL database design



Why Document Format?

01

Flexibility

- Variety of products
- Unique and tailored structure per collection
- Easy to update

02

Simplicity

- Nested data in a single document
- I.e. Orders (multiple products per order)

03

Speed

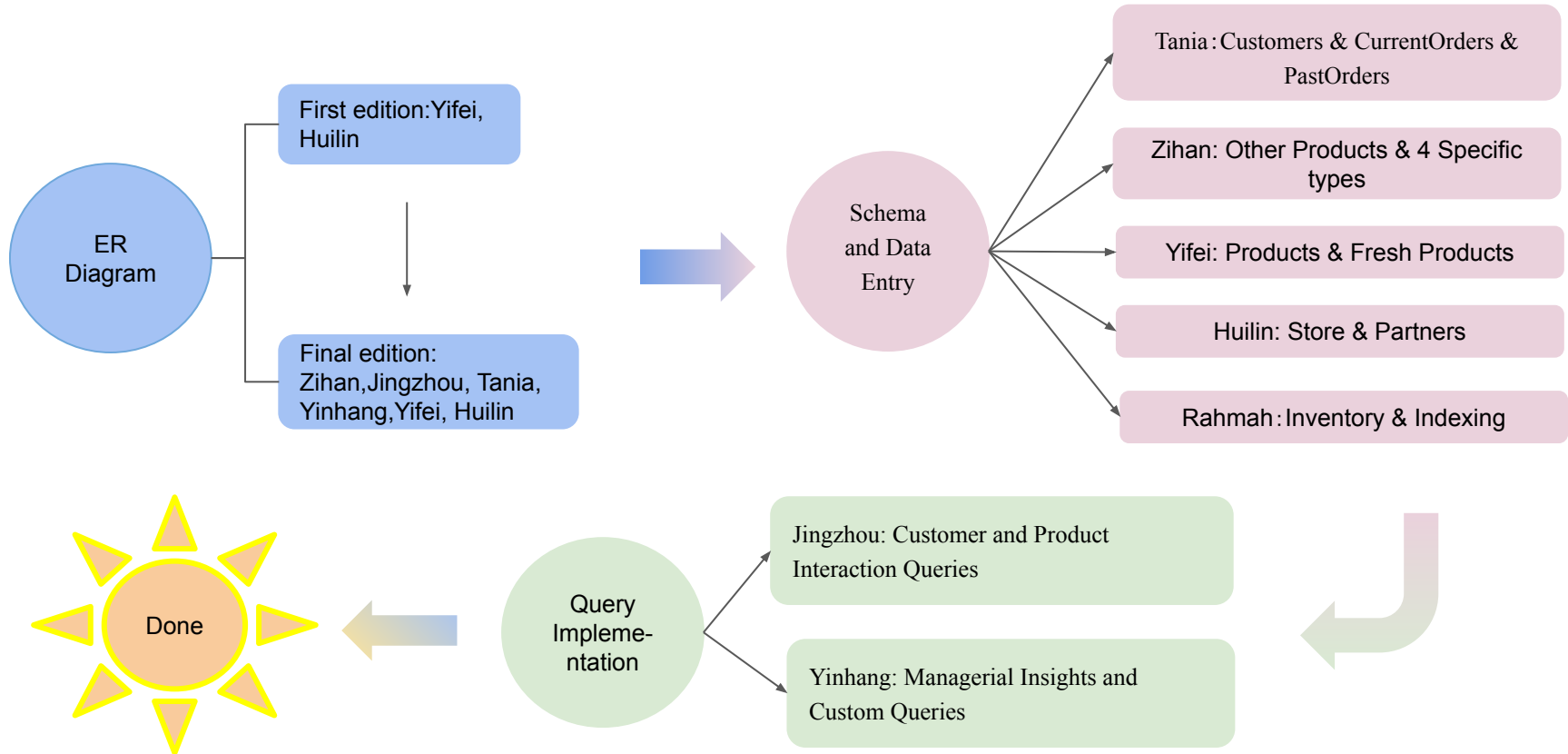
- No joins
- Faster, cheaper, and more simple queries

04

Scalability

- Horizontal
- Ideal for high traffic

Task Allocation



Operations assumptions

- Coordinates (In GeoJSON format) is added for easy location management
- The distance we used are straight line distance, but in reality this is not the case
- Fresh product are stored in the store and other products are placed in the warehouse

Design decisions

- Combine collections
- Separating 'FreshProducts' & 'OtherProducts', all following 'Products'
- Adding longitude and latitude to 'Customers' and 'Stores' to calculate distance
- Add orderStatus to 'CurrentOrders'
- Add averageRating to 'Products'; Calculate the average from the rate of 'PastordelItems'

Design Patterns - 1

What are Design Patterns?

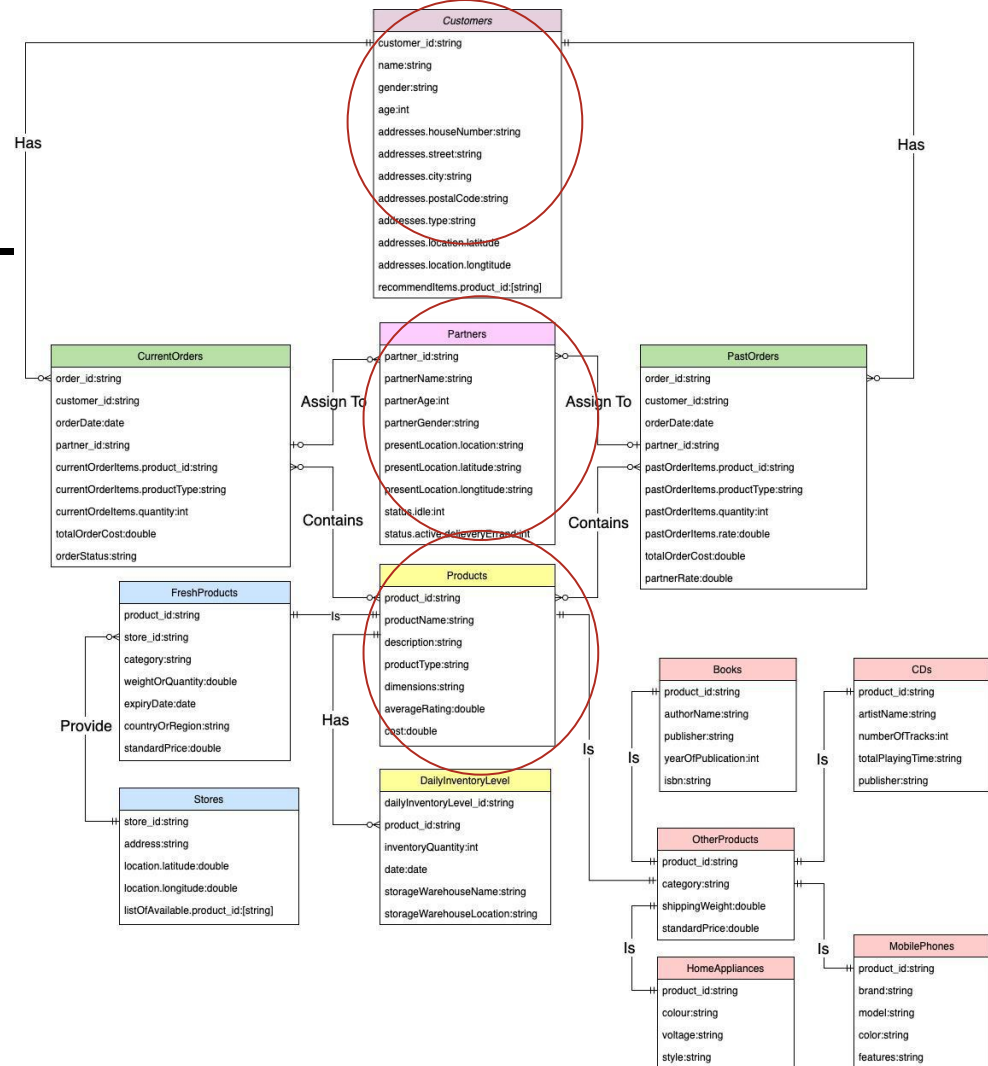
- Solutions to database design problems that are reusable
- Ensure the database is efficient and prevents redundancy, ensuring it can be scaled to other requirements. For example, including all products in Products would make it harder to find specific products from the database.

Reasons to use design patterns in this Database:

- Handling diverse product attributes
- Integrate spatial data for entities like Store, where location-based attributes exist
- Enhance relationships between entities, such as: customers, orders; products, stores.

Design Patterns - ER Diagram

Key entities
highlighted in ER
diagram:



Key Design Patterns - 3

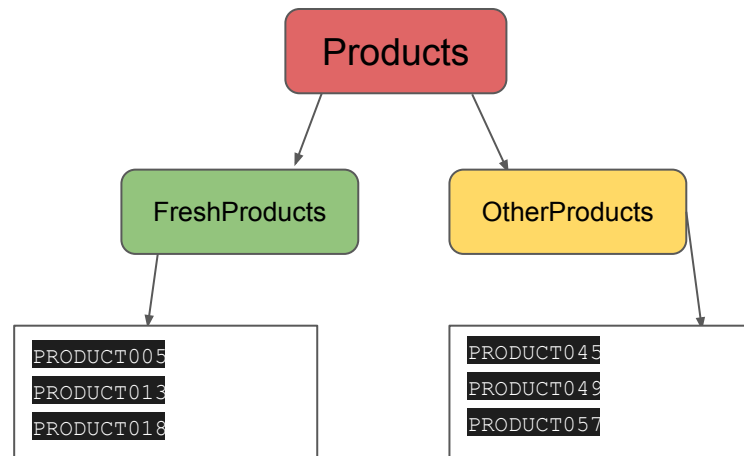
- Entity-Attribute-Value model:

The database separates Product attributes into child tables like:

- CDs
- Books
- FreshProducts etc.

Where Books, CDs, MobilePhones and HomeAppliances belong to OtherProducts, which is then connected to Products.

This helps to manage product variations efficiently; as some entities have less/more attributes. This key-value pattern stores each data item as a pair of unique keys and values. General product attributes exist in Products and more specific attributes in each child entity. Here, these entities inherit attributes through 'product_id.'



Key Design Patterns - 4

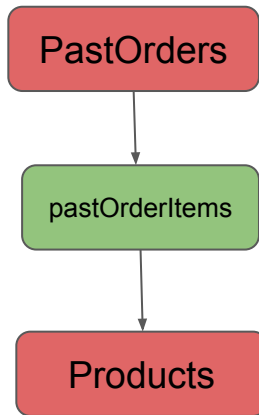
- **Parent-Child design pattern:**

This design allows orders to contain multiple items in a single entity, which in turn, reduces redundancy as there are less repeated attributes.

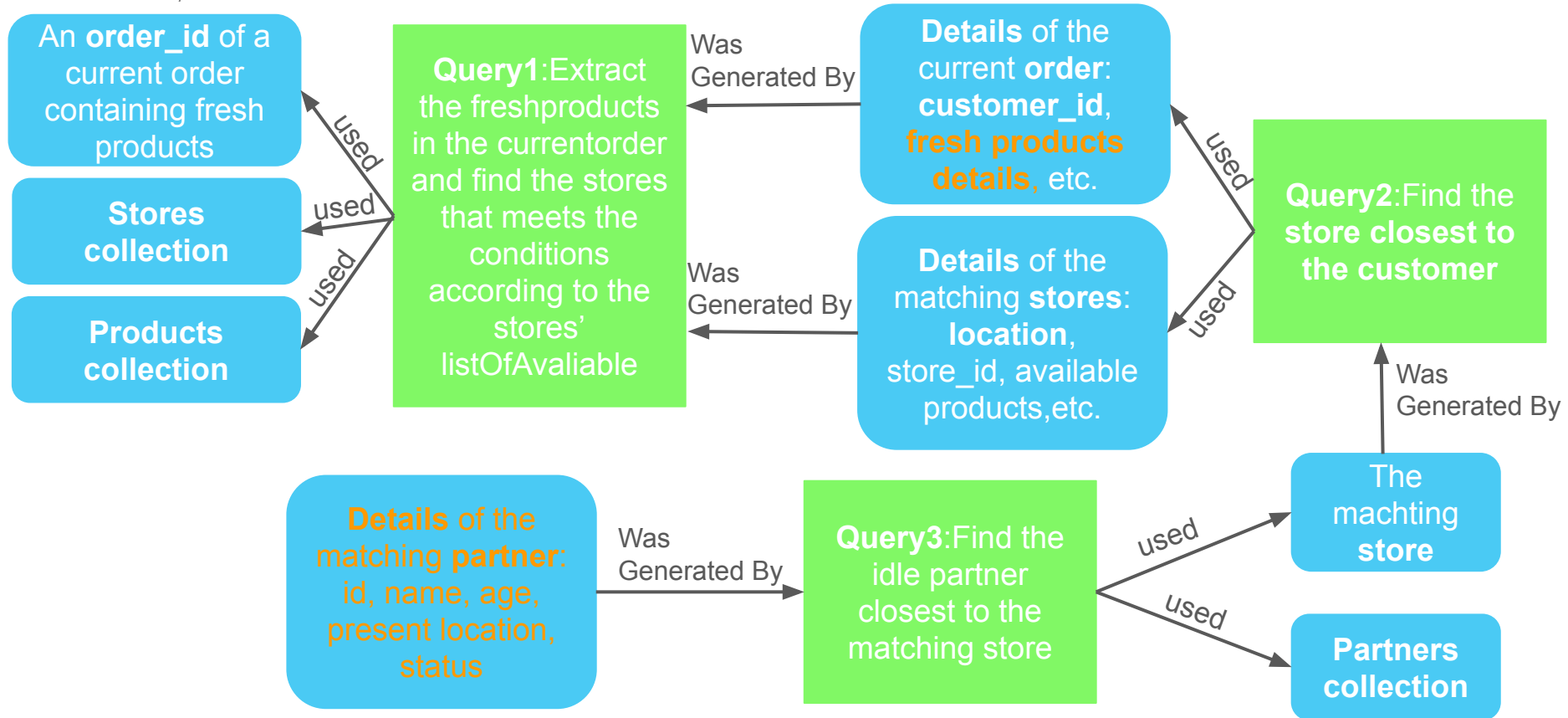
Example:

PastOrders and CurrentOrders contain an array of items which include product_id, productType and quantity etc.

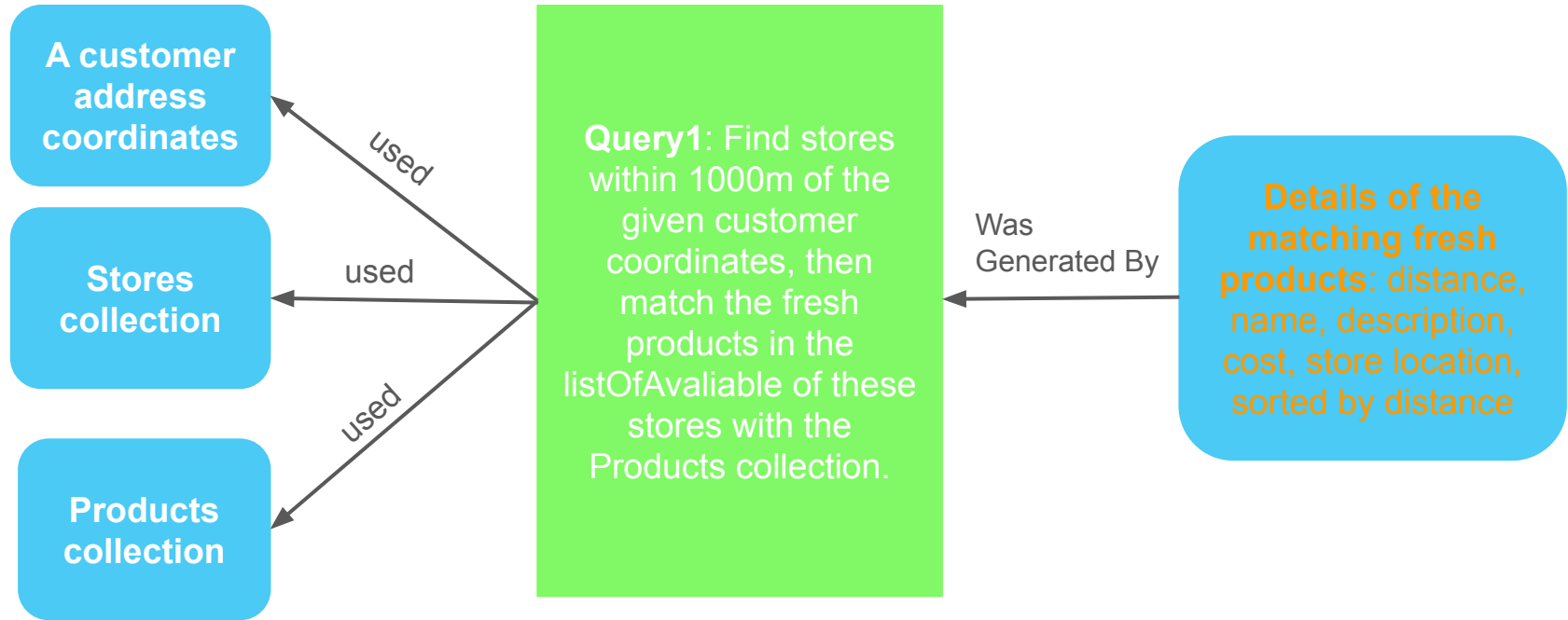
This allows for efficient order processing.



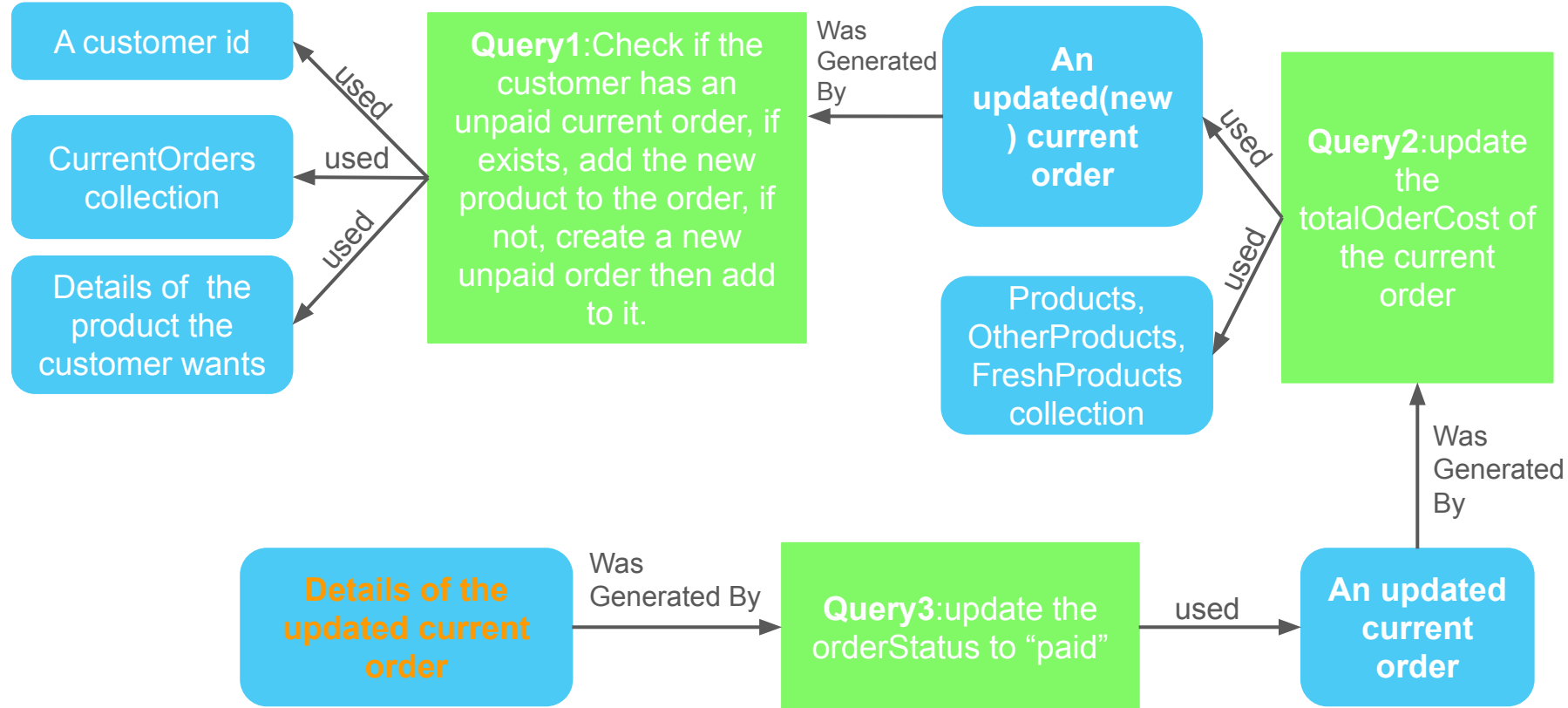
Queries design decisions — Task 1



Queries design decisions — Task 2



Queries design decisions — Task 3



Queries design decisions — Task 4

Query1: Check the total sale quantity, sale value and current inventory (in Current Order)

Was unwind by

Current
order items

Was grouped by

Product Id
(to sum the
total sale
quantity)

Look up

Product Id from
DailyInventoryLevel
as inventory details

Was unwind by

Inventory
details to
save the
document

Look up

Product details
from Fresh and
Others

Was unwind by

Fresh/other
Product
details

project

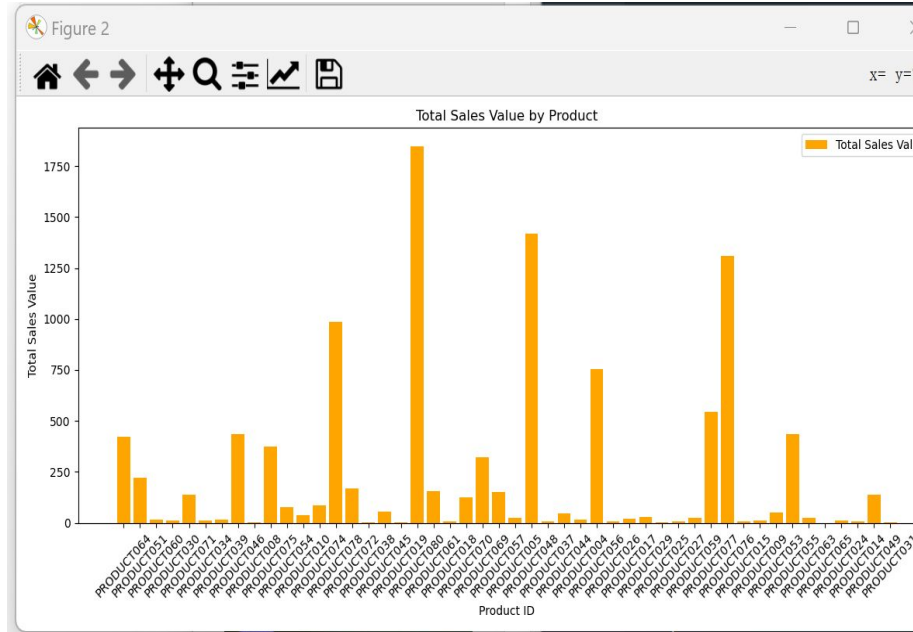
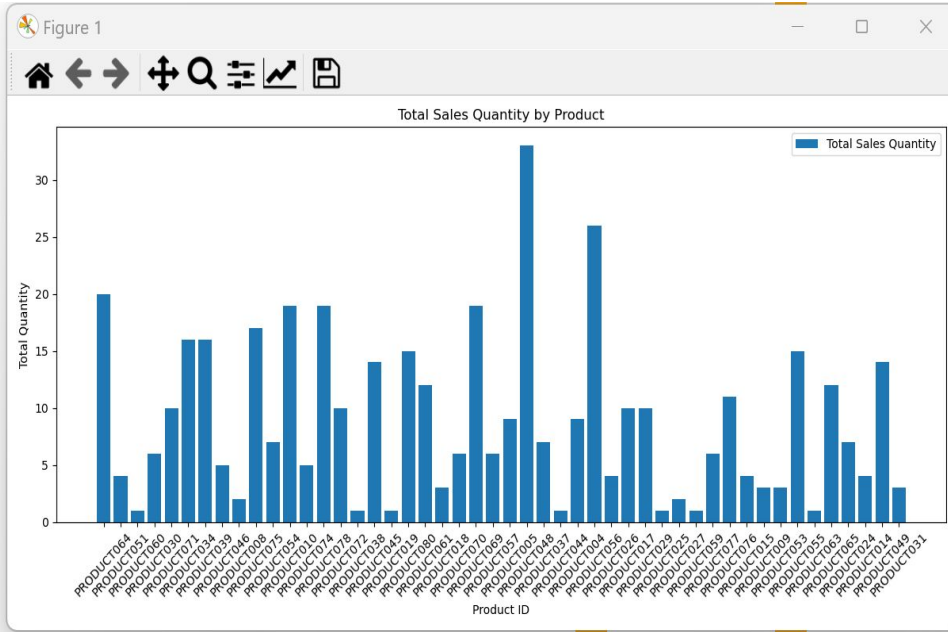
productId, totalSaleQuantity, totalSaleValue, currentInventory

Output example:

Output after [\\$project](#) stage (Sample of 10 documents)

```
{
  "_id": "PRODUCT080"
  totalSalesQuantity : 15
  productId : "PRODUCT080"
  currentInventory : 233
  totalSalesValue : 1845
}
```


Visualization



Queries design decisions — Task 4

Query2: Check the total sale quantity and average Rating (in Past Order)

Was unwinded by

Past order items

Was grouped by

Product Id (to sum the total sale quantity and average rating)

Look up

Product Id from Products as product details

Was unwind by

Procut details

project

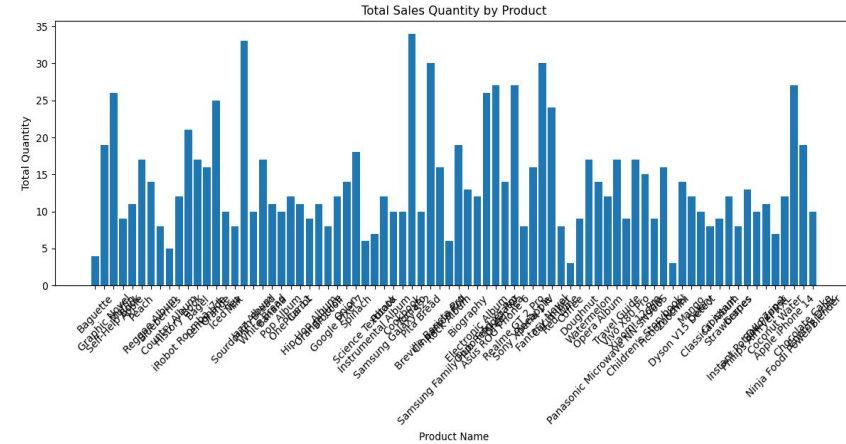
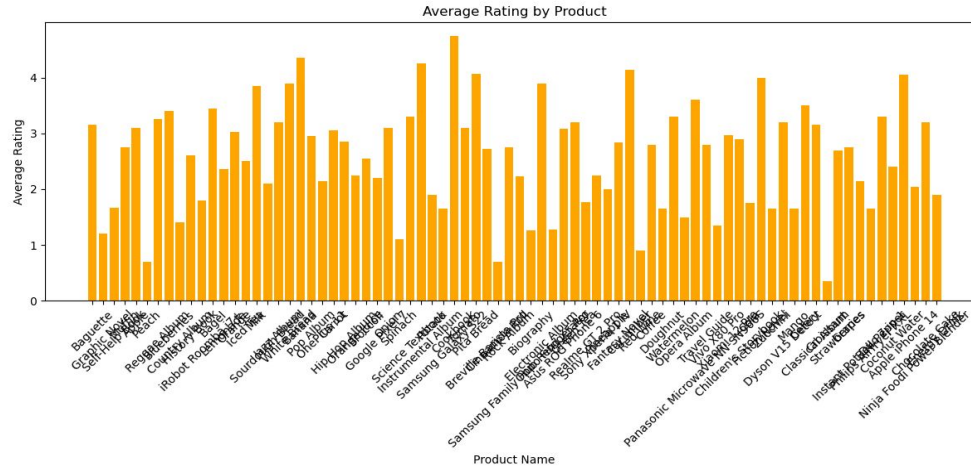
productId, totalSaleQuantity, averageRating

Output example:

Output after [\\$project](#) stage (Sample of 10 documents)

```
{
  "_id": "PRODUCT068"
  totalQuantity : 9
  averageRating : 1.35
  productId : "PRODUCT068"
  productName : "Vivo X80 Pro"
}
```

Visualization



Queries design decisions — Task 5

Query1: Recommend products to customers

Was unwind by

Past order items

Was grouped by

Customer Id (Create a collection of different product IDs purchased by each customer using the \$addToSet method.)

Look up

Top 100 ordered by average rating from Products as all recommend product

Add fields

Add new fields for recommended products

project

customerId, recommend products

Queries design decisions — Task 5

Query2: Show the current location and number of orders for each partners

Look up

Partner Id from
Current Orders
as orders

Was unwinded by

orders

Was
grouped by

Partner Id ,Total
delivery
orders,present
location

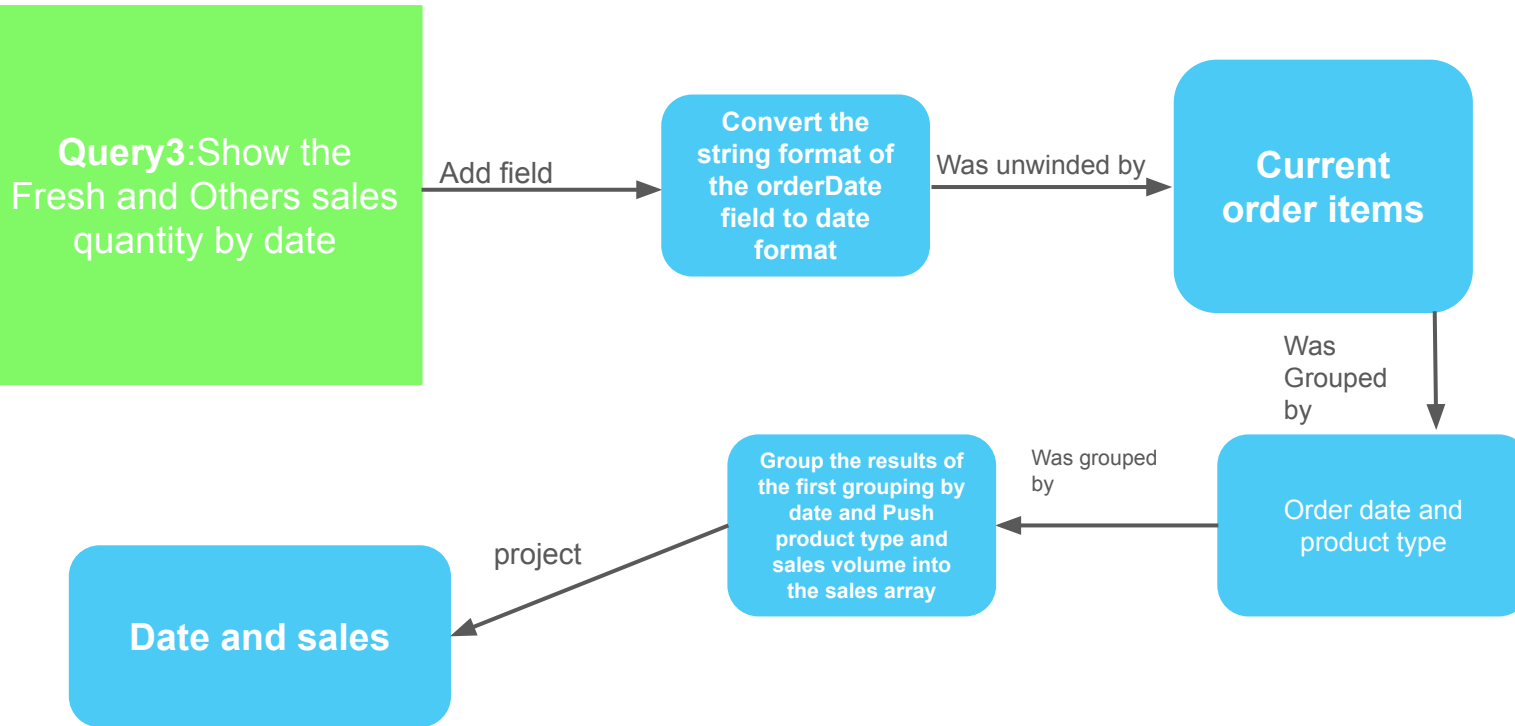
project

Partner Id(and only
sum the paid orders)

Output example:

```
_id: "PARTNER002"
Total_delivery_Orders : 1
partnerName : "Sophia Taylor"
▼ presentLocation : Object
  type : "Point"
  ▼ coordinates : Array (2)
    0: -2.2483
    1: 53.4771
partner_id : "PARTNER002"
```

Queries design decisions — Task 5



Visualization

