Large-Scale and Multi-Structured Databases

Document Databases

Design Tips

Prof. Pietro Ducange

Collections

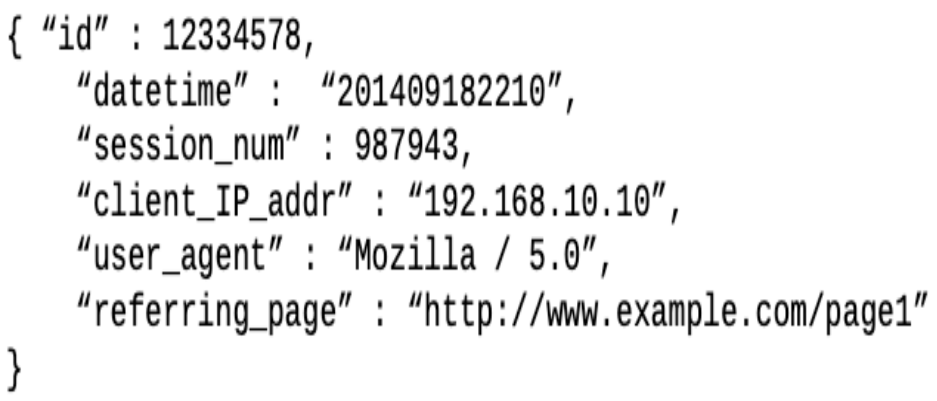
Collections are ***sets*** of documents .

A collection can store documents of ***different types*** (no need of a specific structure/scheme for a document).

In general, collections ***should store*** documents about the ***same type*** of entity.

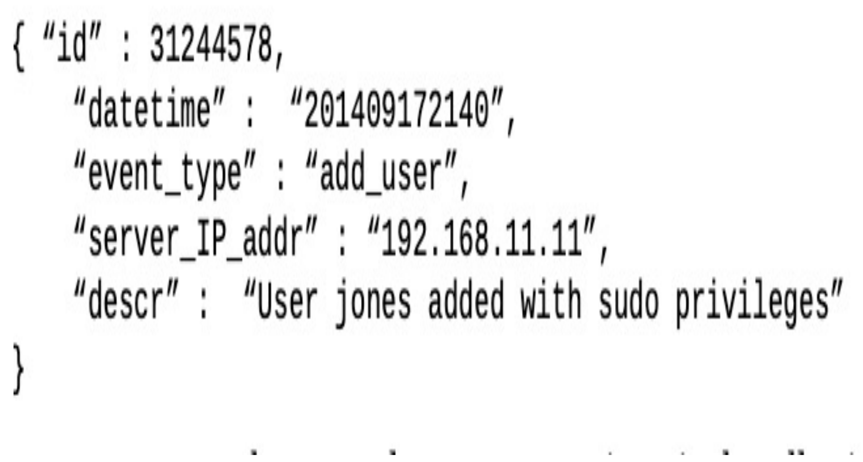
What is the «type» of entity?

Example of Two Entities (?)..



```
{ "id" : 12334578,
    "datetime" :  "201409182210",
    "session_num" : 987943,
    "client_IP_addr" : "192.168.10.10",
    "user_agent" : "Mozilla / 5.0",
    "referring_page" : "http://www.example.com/page1"
}
```

web clickstream data



```
{ "id" : 31244578,
    "datetime" :  "201409172140",
    "event_type" : "add_user",
    "server_IP_addr" : "192.168.11.11",
    "descr" :  "User jones added with sudo privileges"
}
```

server log data

...or Two Instances of the Same Entity

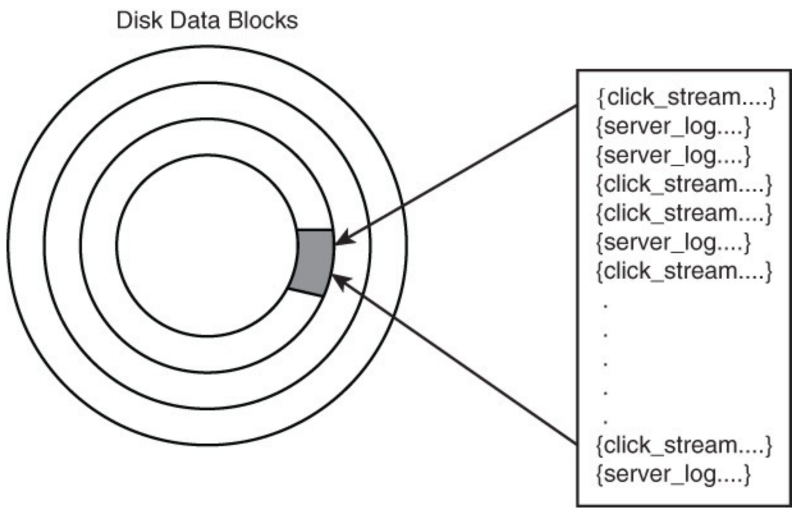
Entity Name: ***System Event***

```
{ "id" : 12334578,
    "datetime" :  "201409182210",
    "doc_type": "click_stream",
    "session_num" : 987943,
    "client_IP_addr" : "192.168.10.10",
    "user_agent" : "Mozilla / 5.0",
    "referring_page" : "http://www.example.com/page1"
}

{ "id" : 31244578,
    "datetime" :  "201409172140"
    "doc_type" : "server_log"
    "event_type" : "add_user"
    "server_IP_addr" : "192.168.11.11"
    "descr" :  "User jones added with sudo privileges"
}
```

Can we store the two documents in the same collection?

Let’s Store the Two Documents Together (I)

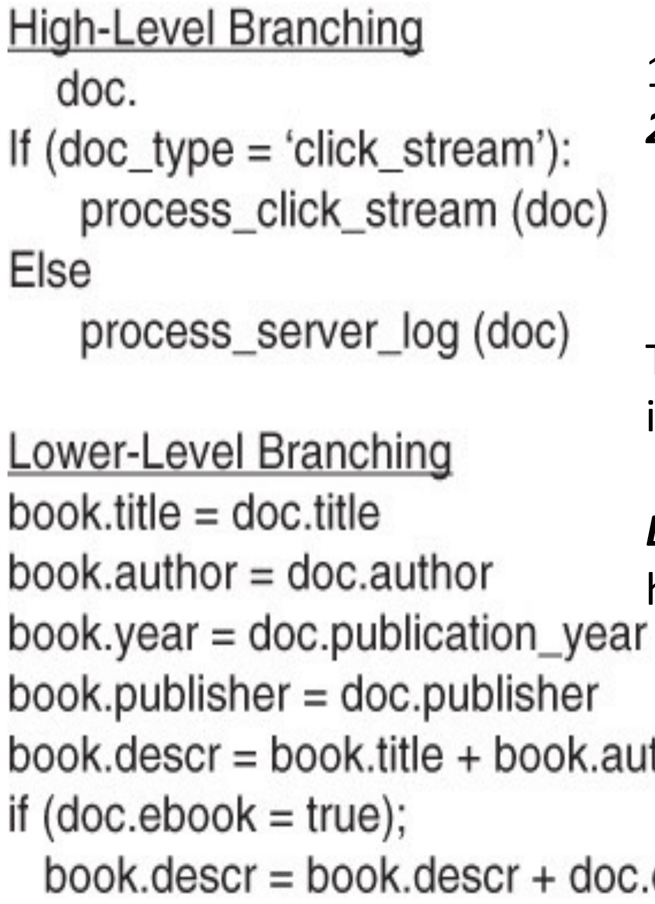


Mixing document types in the same collection can lead to ***multiple document types*** in a ***disk*** data ***block***.

This can lead to ***inefficiencies*** whenever data is read from disk but not used by the application that filters documents based on type.

Filtering collections is often ***slower*** than working directly ***with multiple collections***, each of which contains a single document type.

What About the Code?



High-Level Branching

```
    doc.
If (doc_type = 'click_stream'):
      process_click_stream (doc)
Else
      process_server_log (doc)
```

Lower-Level Branching

```
book.title = doc.title
book.author = doc.author
book.year = doc.publication_year
book.publisher = doc.publisher
book.descr = book.title + book.author + book.year + book.publisher
if (doc.ebook = true);
   book.descr = book.descr + doc.ebook_size
```

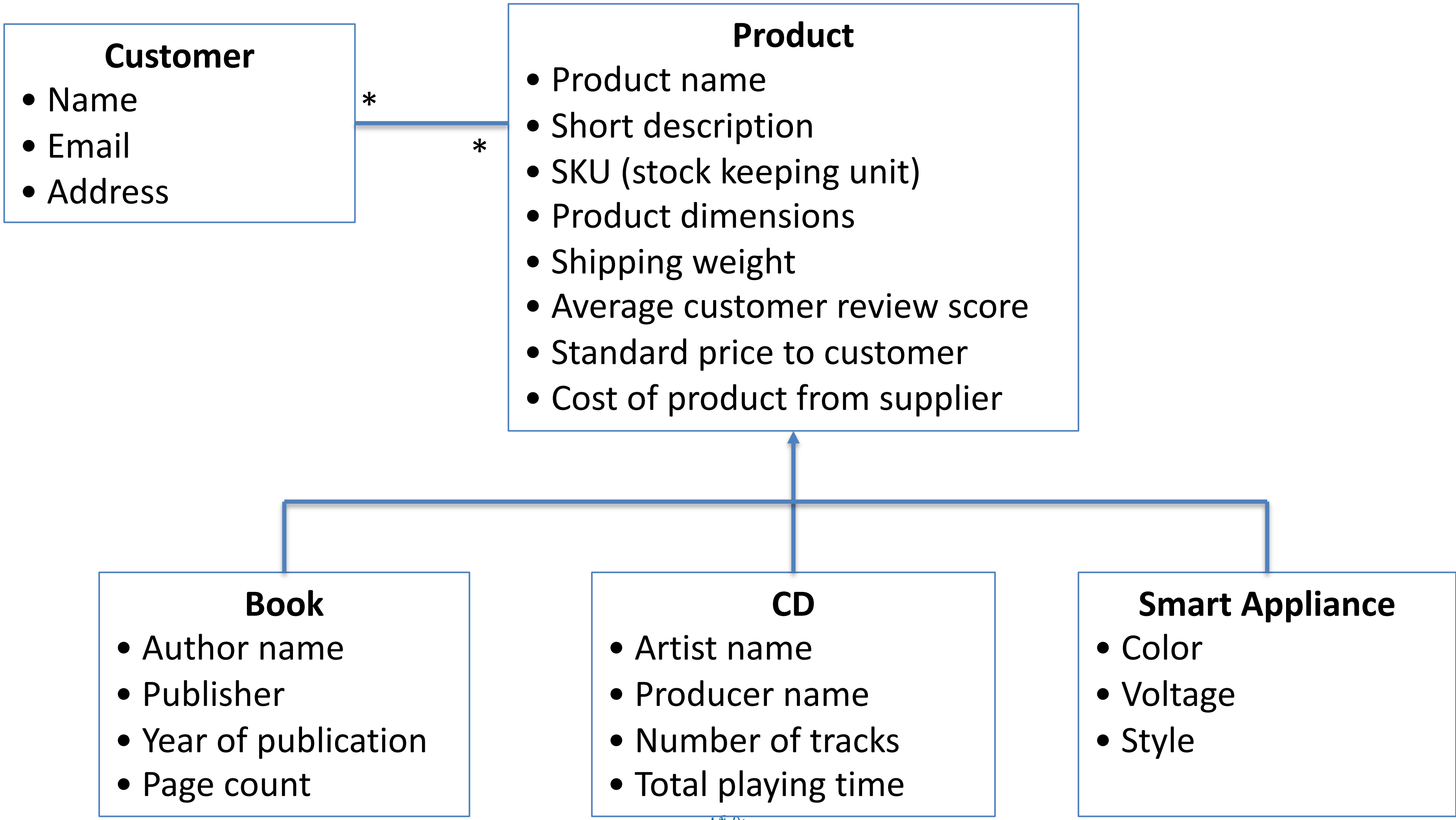
In general, the application code written for ***manipulating*** a collection should have:

1) A ***substantial*** amounts of code that apply to ***all documents***
2) ***Some amount*** of code that accommodates ***specialized fields*** in some documents.

The case of ***High-Level Branching*** like in the picture, can indicate a need to create ***separate*** collections.

Branching at lower levels is common when some documents have ***optional attributes***.

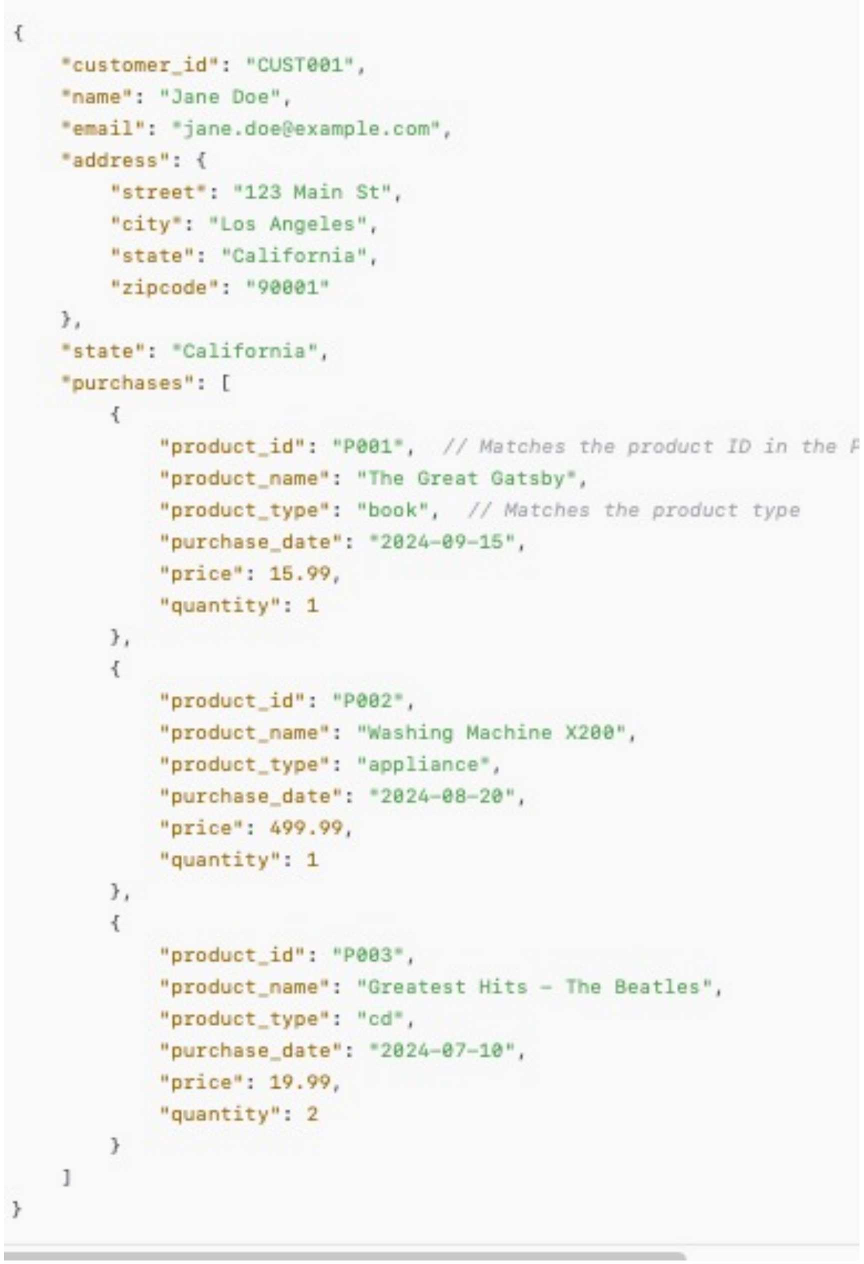
Follow the Definition of Queries



Follow the Definition of Queries

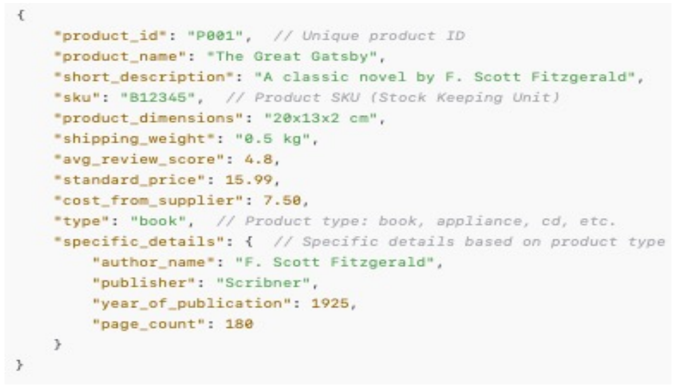
Our application might to be able to answer the following queries:

- What is the average number of products bought by each customer?
- What is the range of number of products purchased by customers
- What are the top 20 most popular products by customer state?
- What is the average value of sales by customer state
- How many of each type of product were sold in the last 30 days?

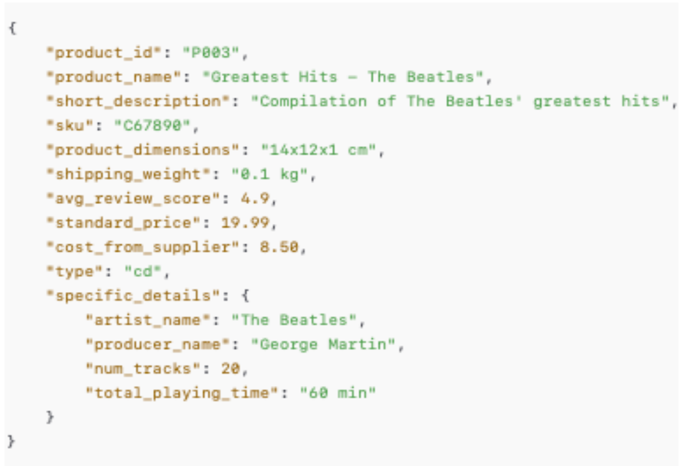


```
{
    "customer_id": "CUST001",
    "name": "Jane Doe",
    "email": "jane.doe@example.com",
    "address": {
        "street": "123 Main St",
        "city": "Los Angeles",
        "state": "California",
        "zipcode": "90001"
    },
    "state": "California",
    "purchases": [
        {
            "product_id": "P001",  // Matches the product ID in the P
            "product_name": "The Great Gatsby",
            "product_type": "book",  // Matches the product type
            "purchase_date": "2024-09-15",
            "price": 15.99,
            "quantity": 1
        },
        {
            "product_id": "P002",
            "product_name": "Washing Machine X200",
            "product_type": "appliance",
            "purchase_date": "2024-08-20",
            "price": 499.99,
            "quantity": 1
        },
        {
            "product_id": "P003",
            "product_name": "Greatest Hits - The Beatles",
            "product_type": "cd",
            "purchase_date": "2024-07-10",
            "price": 19.99,
            "quantity": 2
        }
    ]
}
```

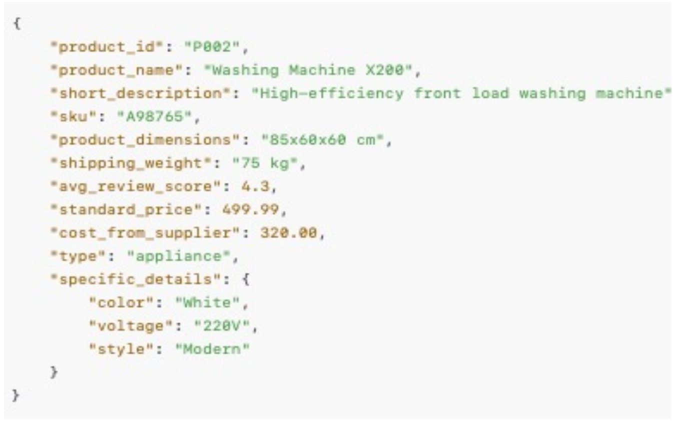
Customer Document (stored in a specific Customers Collection)



```
{
    "product_id": "P001",  // Unique product ID
    "product_name": "The Great Gatsby",
    "short_description": "A classic novel by F. Scott Fitzgerald",
    "sku": "B12345",  // Product SKU (Stock Keeping Unit)
    "product_dimensions": "20x13x2 cm",
    "shipping_weight": "0.5 kg",
    "avg_review_score": 4.8,
    "standard_price": 15.99,
    "cost_from_supplier": 7.50,
    "type": "book",  // Product type: book, appliance, cd, etc.
    "specific_details": {  // Specific details based on product type
        "author_name": "F. Scott Fitzgerald",
        "publisher": "Scribner",
        "year_of_publication": 1925,
        "page_count": 180
    }
}
```



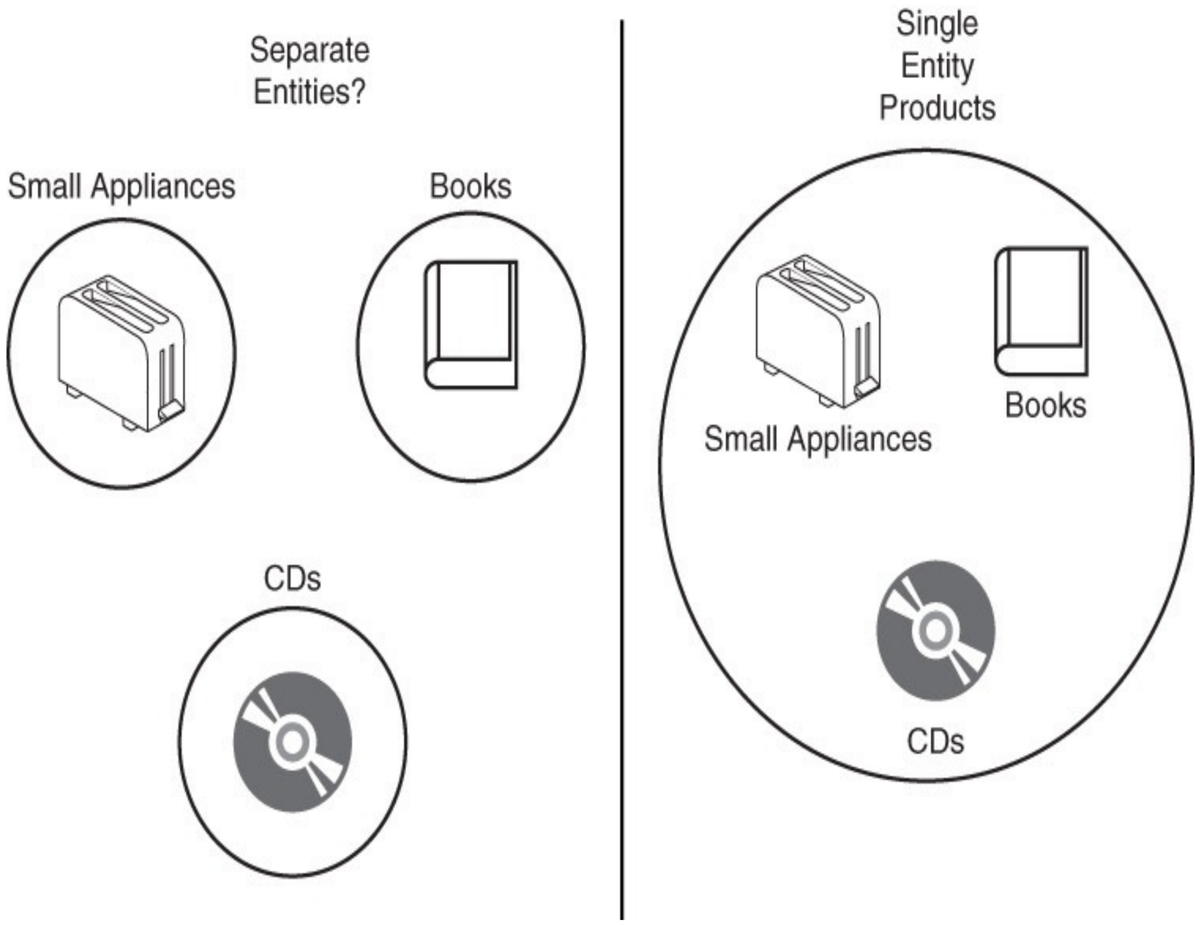
```
{
    "product_id": "P003",
    "product_name": "Greatest Hits - The Beatles",
    "short_description": "Compilation of The Beatles' greatest hits",
    "sku": "C67890",
    "product_dimensions": "14x12x1 cm",
    "shipping_weight": "0.1 kg",
    "avg_review_score": 4.9,
    "standard_price": 19.99,
    "cost_from_supplier": 8.50,
    "type": "cd",
    "specific_details": {
        "artist_name": "The Beatles",
        "producer_name": "George Martin",
        "num_tracks": 20,
        "total_playing_time": "60 min"
    }
}
```



```
{
    "product_id": "P002",
    "product_name": "Washing Machine X200",
    "short_description": "High-efficiency front load washing machine"
    "sku": "A98765",
    "product_dimensions": "85x60x60 cm",
    "shipping_weight": "75 kg",
    "avg_review_score": 4.3,
    "standard_price": 499.99,
    "cost_from_supplier": 320.00,
    "type": "appliance",
    "specific_details": {
        "color": "White",
        "voltage": "220V",
        "style": "Modern"
    }
}
```

Collection of Products

Follow the Definition of Queries



Notice that: If we separate the product into different collections, and the number of product types grows the number of collections would become unwieldy.

Normalization or Denormalization?

Normalization helps ***avoid*** data ***anomalies***, but it can cause ***performance problems.***

With ***normalized*** data, we need ***join operations***, which must be optimized for improving performances.

If we use ***denormalized*** data, we may introduce ***redundancies*** and cause anomalies.

On the other hand, we may ***improve the performances*** of the queries because we ***reduce*** the number of collections and ***avoid join operations***.

Denormalization supports improving read operations when ***indexes*** are adopted.

Suggested Readings

Chapters 6 of the book "*Dan Sullivan, NoSQL For Mere Mortals, Addison-Wesley, 2015*"