Large-Scale and Multi-Structured Databases

Document Databases

Design Tips

Prof. Pietro Ducange

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

UNIVERSITÀ DI PISA

CROSSLAB Innovation for industry 4.0

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

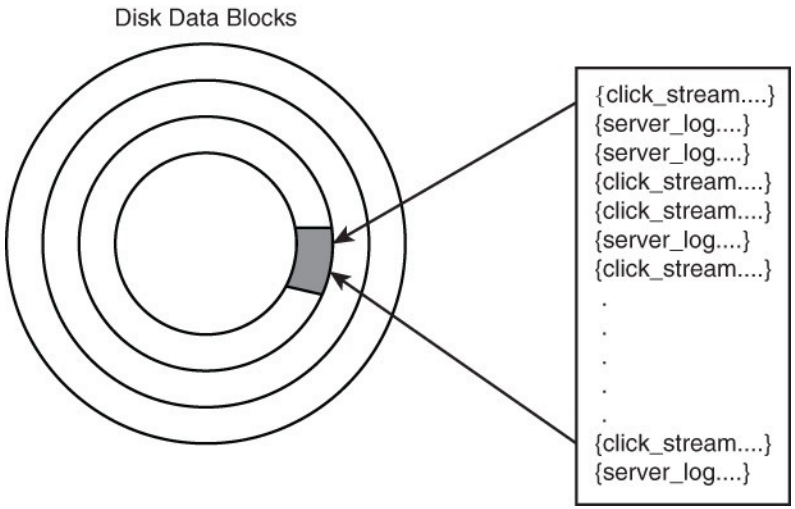
Entity Name: ***System Event***

```
{ "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.

Let's Store the Two Documents Together (II)

If we decide to use ***indexes***, recall that they reference a data block that contains both clickstream and server log data, the disk will read ***both types*** of records even though one will be filtered out in your application.

In the case of large collection and large number of distinct "types" of documents, it may be faster ***to scan the full document*** collection rather than use an index .

Store different types of documents in the same collection only if they will be ***frequently used together*** in the application.

What About the Code?

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.

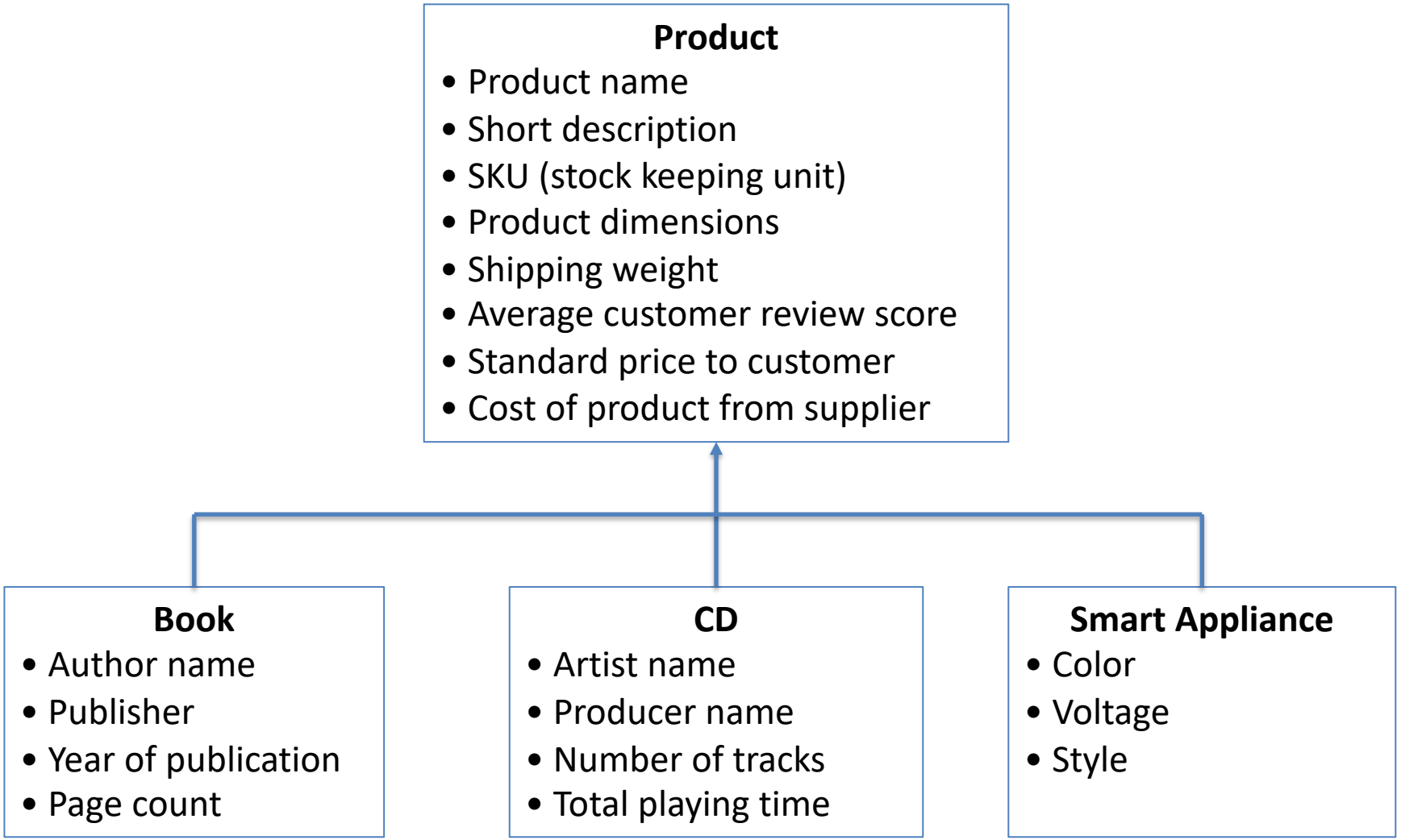
```
High-Level Branching
   doc.
If (doc_type = 'click_stream'):
    process_click_stream (doc)
Else
    process_server_log (doc)

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
```

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

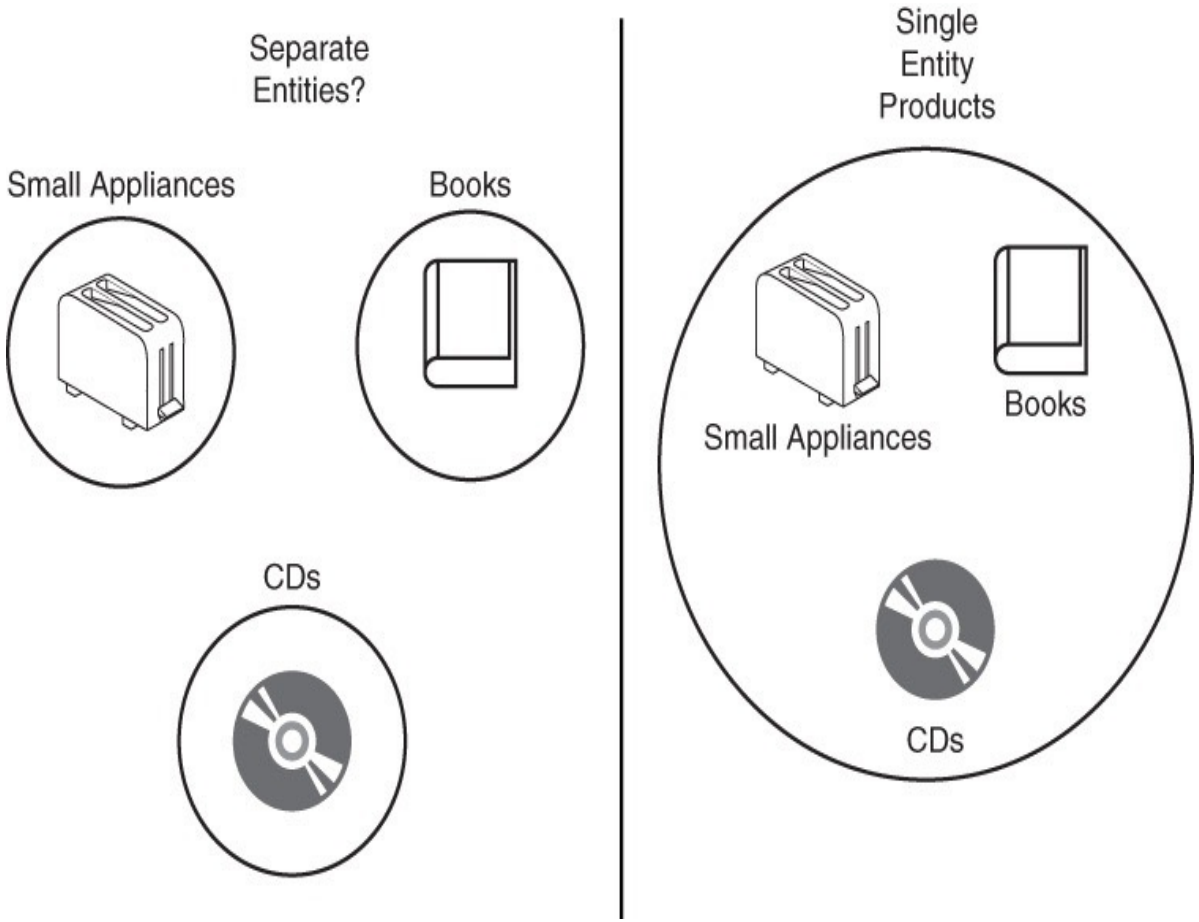


Follow the Definition of Queries

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?

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*"

Work Opportunity For Students

https://alboufficiale.unipi.it/blog/2022/10/17/bando-per-laffidamento-di-n-4-incarichi-per-attivita-tutoriali-a-favore-di-studenti-con-disturbi-specifici-dellapprendimento-riservato-a-studenti-iscritti-ai-cdlm-afferenti-al-dip-2/