Systems and Methods for Big and Unstructured Data

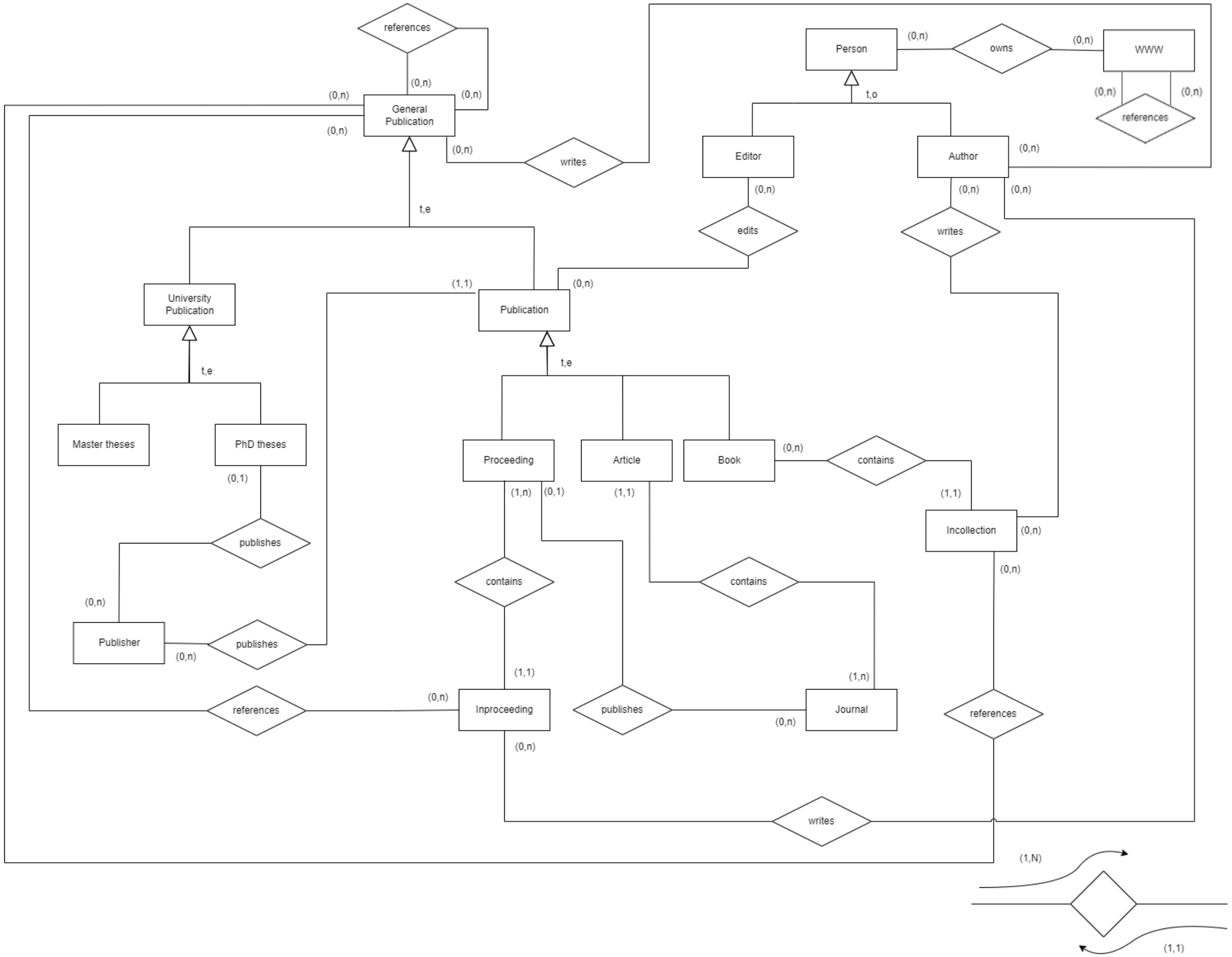
Group 24 – Project presentation

Macaccaro Roberto, Montemurro Elena, Radaelli Marta, Rondini Luca, Scandale Francesco

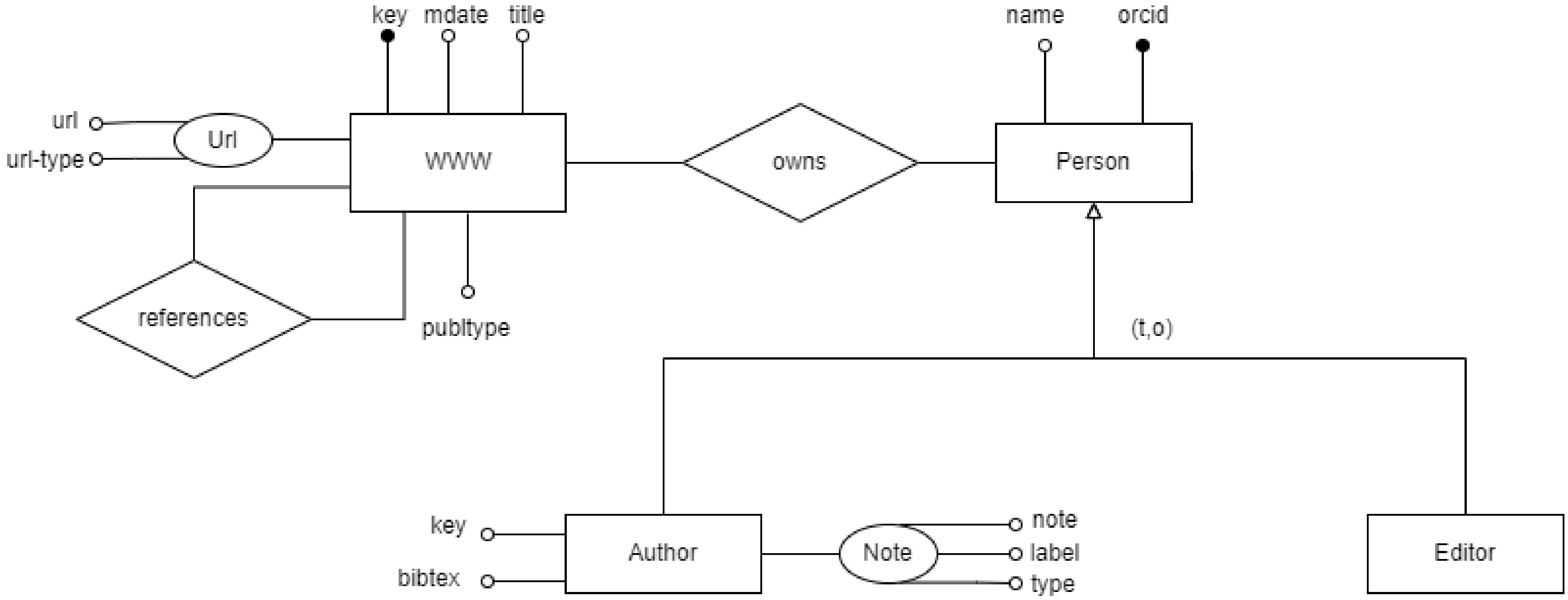
Project Introduction & Assumptions

- **Purpose**: effectively manage, with the use of different technologies, the data of a bibliographic database while keeping track of all published scientific publications, their characteristic, citations, authors, editors, and publishers.
- **DBLP database**: analysis of the document to infer the model structure.
- **Mockaroo:** generation of a sample dataset.

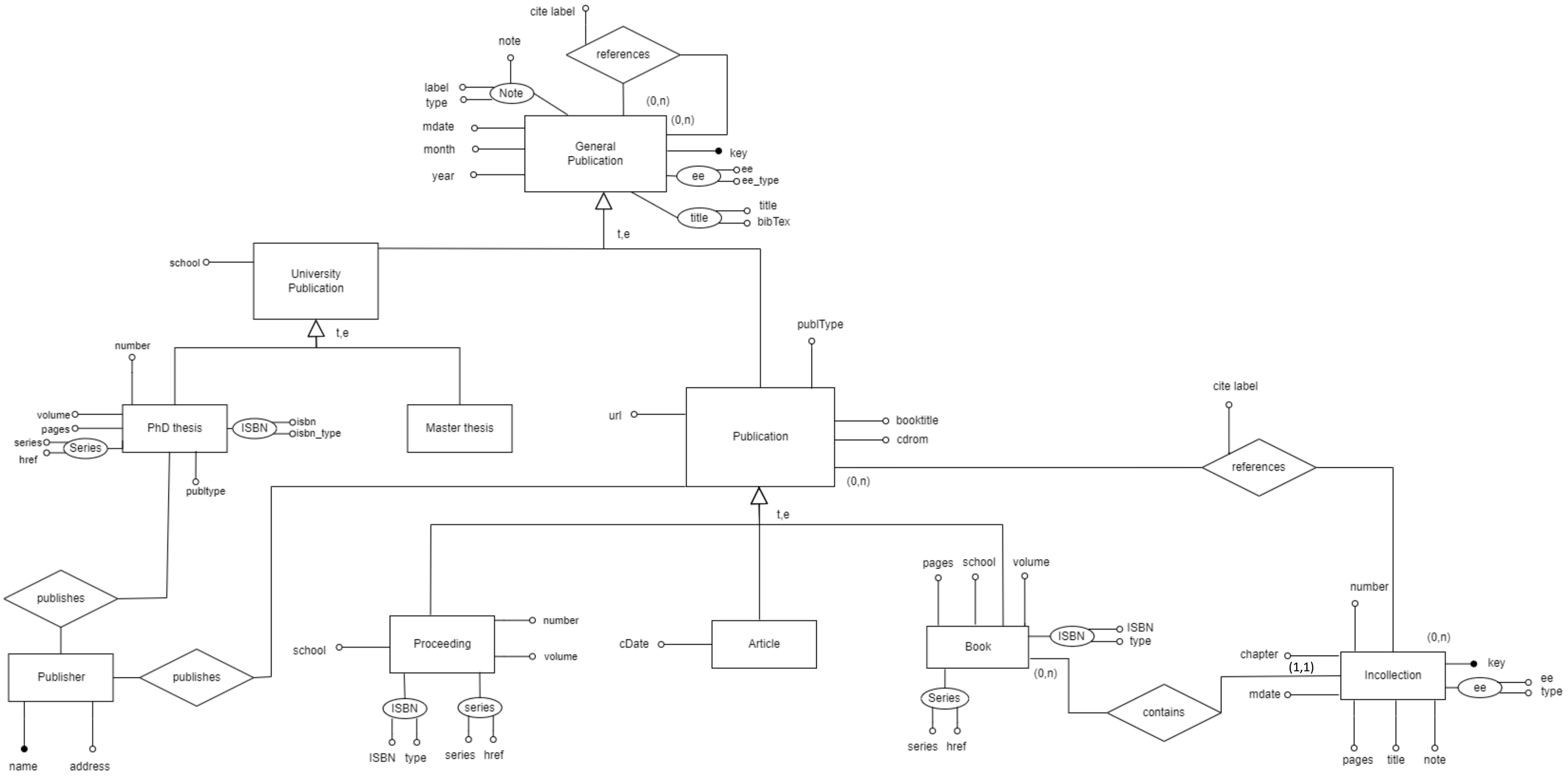
ER model


references
(0,n)
(0,n)
(0,n)
General Publication
(0,n)
(0,n)
writes
Person
(0,n)
owns
(0,n)
WWW
t,o
(0,n)
(0,n)
references
Editor
Author
(0,n)
(0,n)
(0,n)
(0,n)
t,e
edits
writes
University Publication
(1,1)
Publication
(0,n)
t,e
t,e
Master theses
PhD theses
Proceeding
Article
Book
(0,n)
contains
(0,1)
(1,n)
(0,1)
(1,1)
(1,1)
Incollection
(0,n)
publishes
(0,n)
contains
contains
(0,n)
(0,n)
Publisher
publishes
(0,n)
(1,1)
(1,n)
(0,n)
references
Inproceeding
publishes
(0,n)
Journal
references
(0,n)
writes
(1,N)
(1,1)

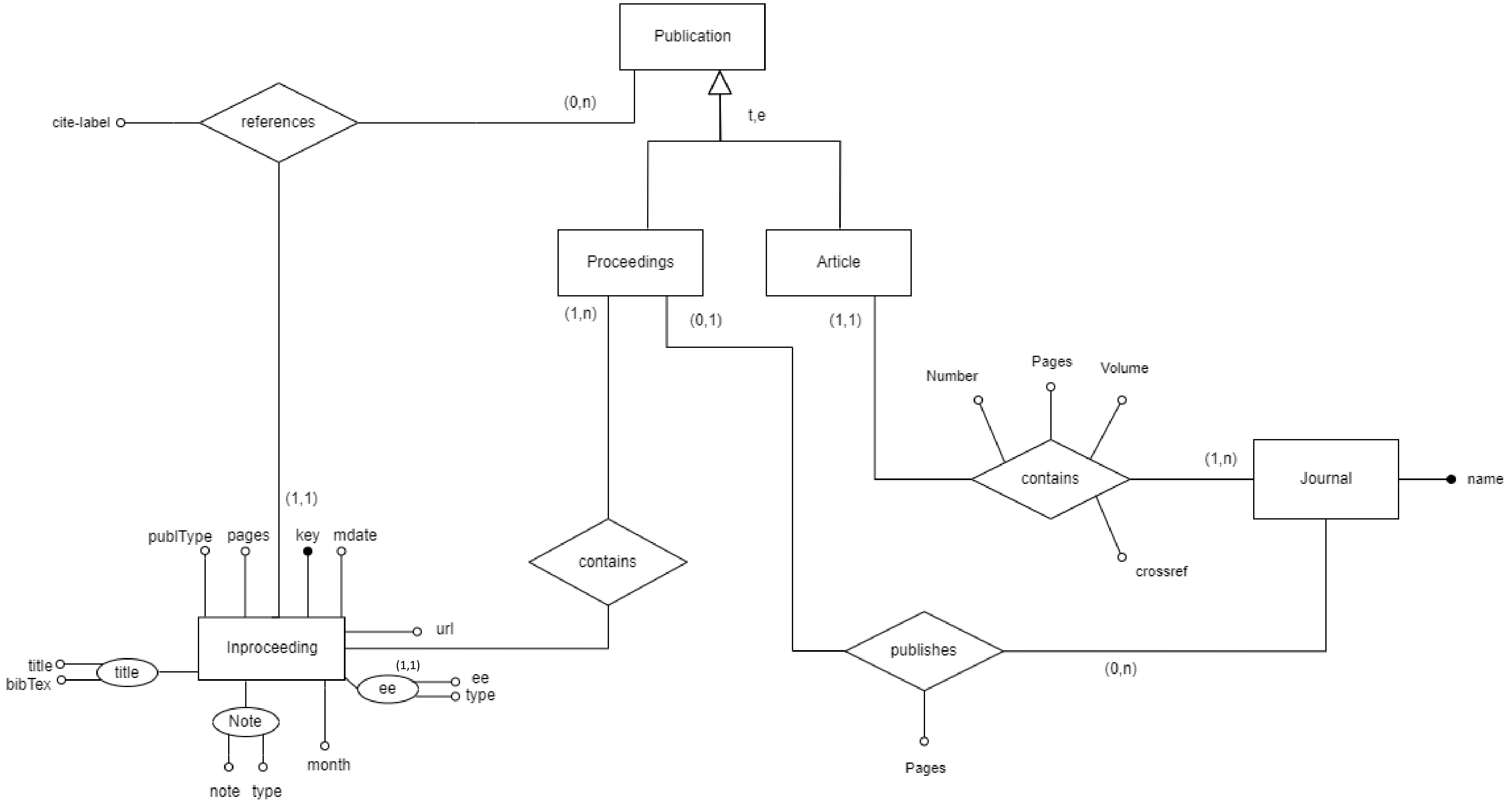
ER model - Person



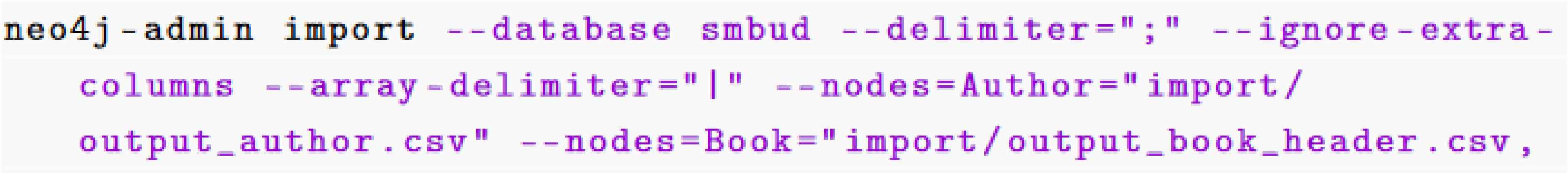
ER model – General Publication



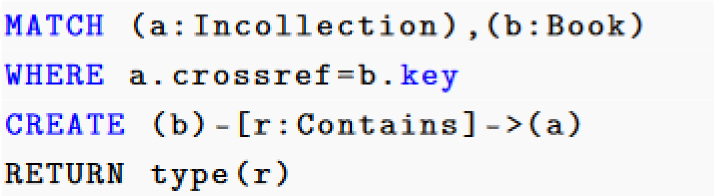
ER model - Publication



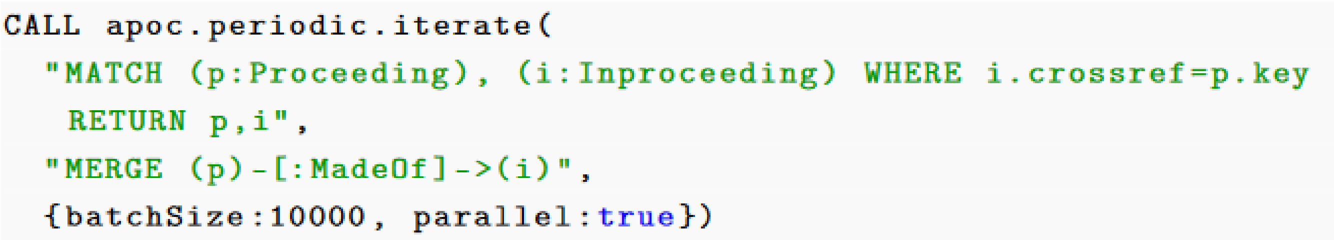
Neo4j – Loading data and creating relationships



```
neo4j-admin import --database smbud --delimiter=";" --ignore-extra-
   columns --array-delimiter="|" --nodes=Author="import/
   output_author.csv" --nodes=Book="import/output_book_header.csv,
```

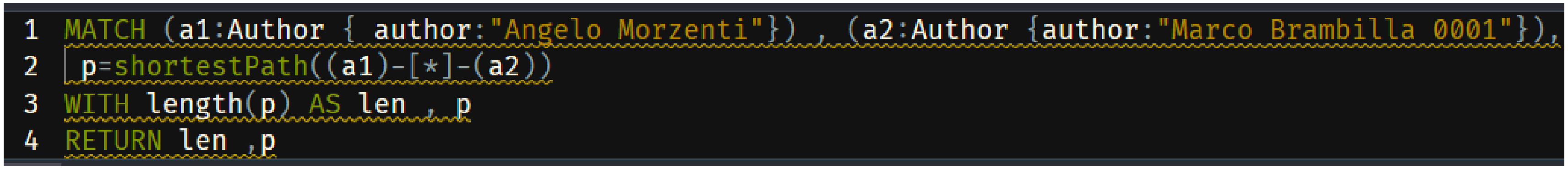


```
MATCH (a:Incollection),(b:Book)
WHERE a.crossref=b.key
CREATE (b)-[r:Contains]->(a)
RETURN type(r)
```

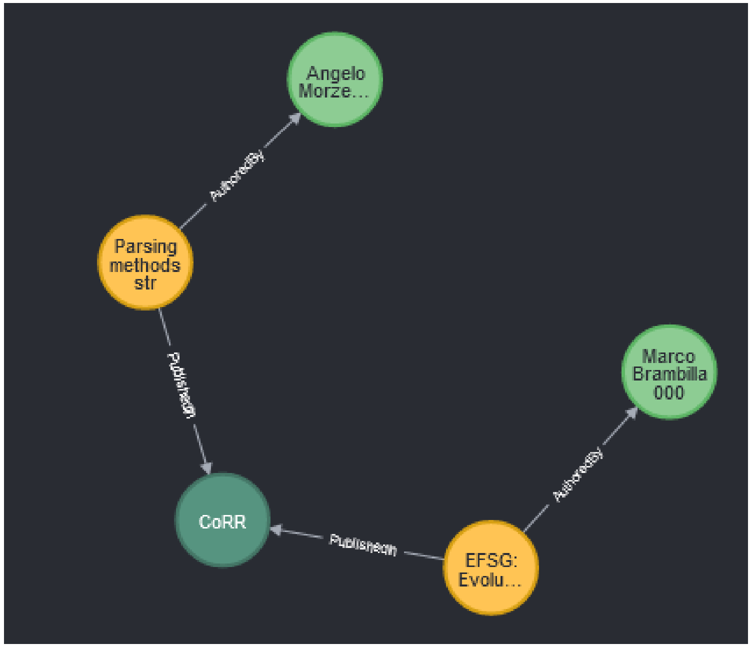


```
CALL apoc.periodic.iterate(
  "MATCH (p:Proceeding), (i:Inproceeding) WHERE i.crossref=p.key
   RETURN p,i",
  "MERGE (p)-[:MadeOf]->(i)",
  {batchSize:10000, parallel:true})
```

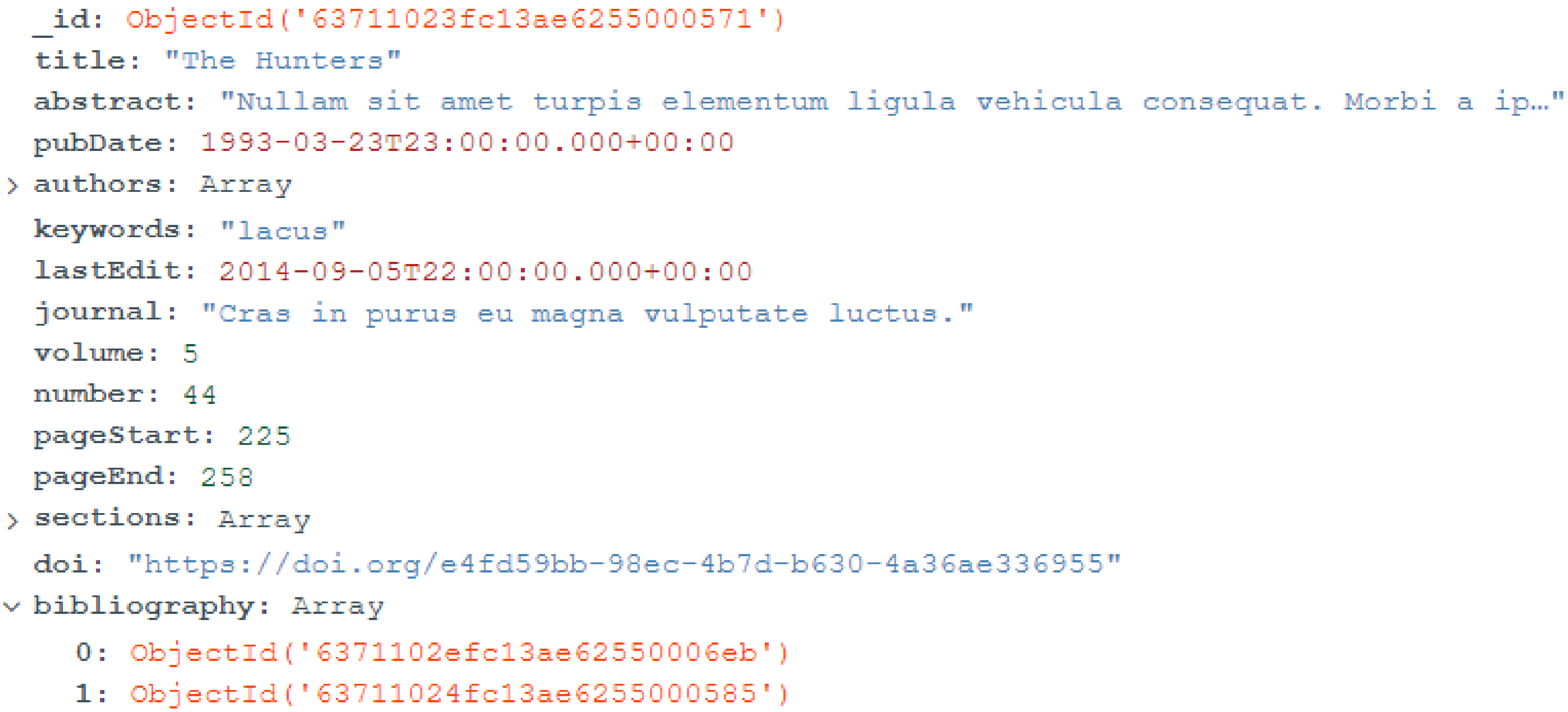
Neo4j – Shortest path between two authors - Fast Bidirectional Breadth-first Search Algorithm



```
MATCH (a1:Author { author:"Angelo Morzenti"}) , (a2:Author {author:"Marco Brambilla 0001"}),
 p=shortestPath((a1)-[*]-(a2))
WITH length(p) AS len , p
RETURN len ,p
```

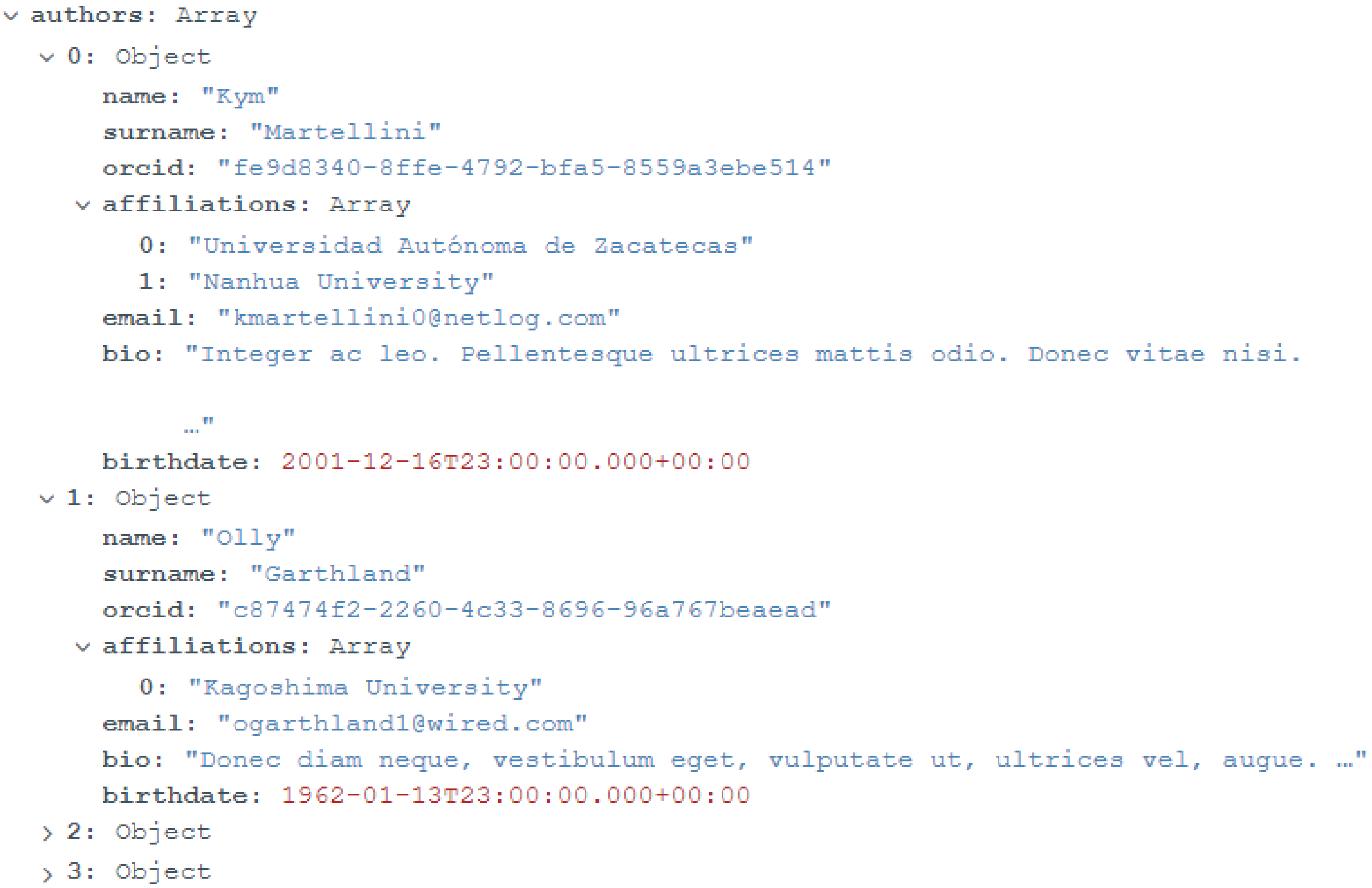


MongoDB - Article



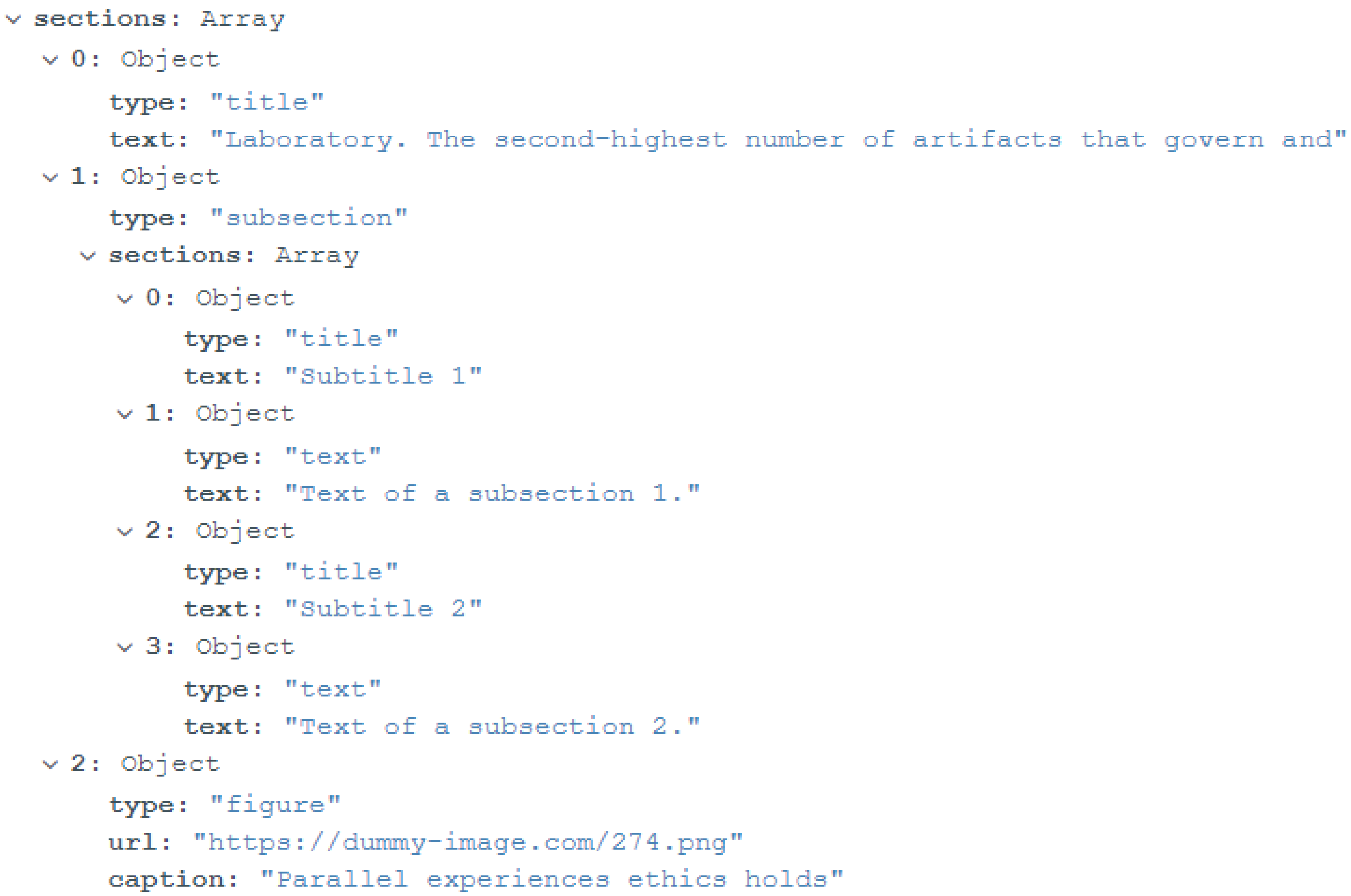
```
  _id: ObjectId('63711023fc13ae6255000571')
  title: "The Hunters"
  abstract: "Nullam sit amet turpis elementum ligula vehicula consequat. Morbi a ip…"
  pubDate: 1993-03-23T23:00:00.000+00:00
› authors: Array
  keywords: "lacus"
  lastEdit: 2014-09-05T22:00:00.000+00:00
  journal: "Cras in purus eu magna vulputate luctus."
  volume: 5
  number: 44
  pageStart: 225
  pageEnd: 258
› sections: Array
  doi: "https://doi.org/e4fd59bb-98ec-4b7d-b630-4a36ae336955"
⌄ bibliography: Array
    0: ObjectId('6371102efc13ae62550006eb')
    1: ObjectId('63711024fc13ae6255000585')
```

MongoDB - Author



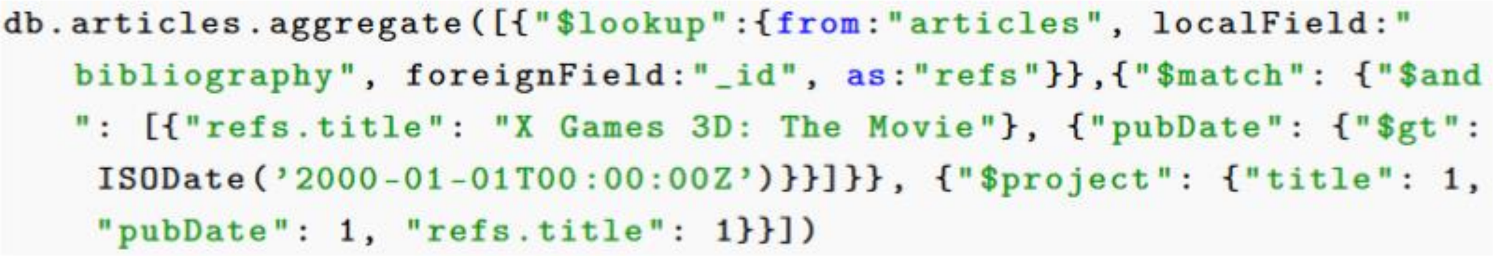
```
⌄ authors: Array
  ⌄ 0: Object
      name: "Kym"
      surname: "Martellini"
      orcid: "fe9d8340-8ffe-4792-bfa5-8559a3ebe514"
    ⌄ affiliations: Array
        0: "Universidad Autónoma de Zacatecas"
        1: "Nanhua University"
      email: "kmartellini0@netlog.com"
      bio: "Integer ac leo. Pellentesque ultrices mattis odio. Donec vitae nisi.

           …"
      birthdate: 2001-12-16T23:00:00.000+00:00
  ⌄ 1: Object
      name: "Olly"
      surname: "Garthland"
      orcid: "c87474f2-2260-4c33-8696-96a767beaead"
    ⌄ affiliations: Array
        0: "Kagoshima University"
      email: "ogarthland1@wired.com"
      bio: "Donec diam neque, vestibulum eget, vulputate ut, ultrices vel, augue. …"
      birthdate: 1962-01-13T23:00:00.000+00:00
  › 2: Object
  › 3: Object
```

MongoDB - Section

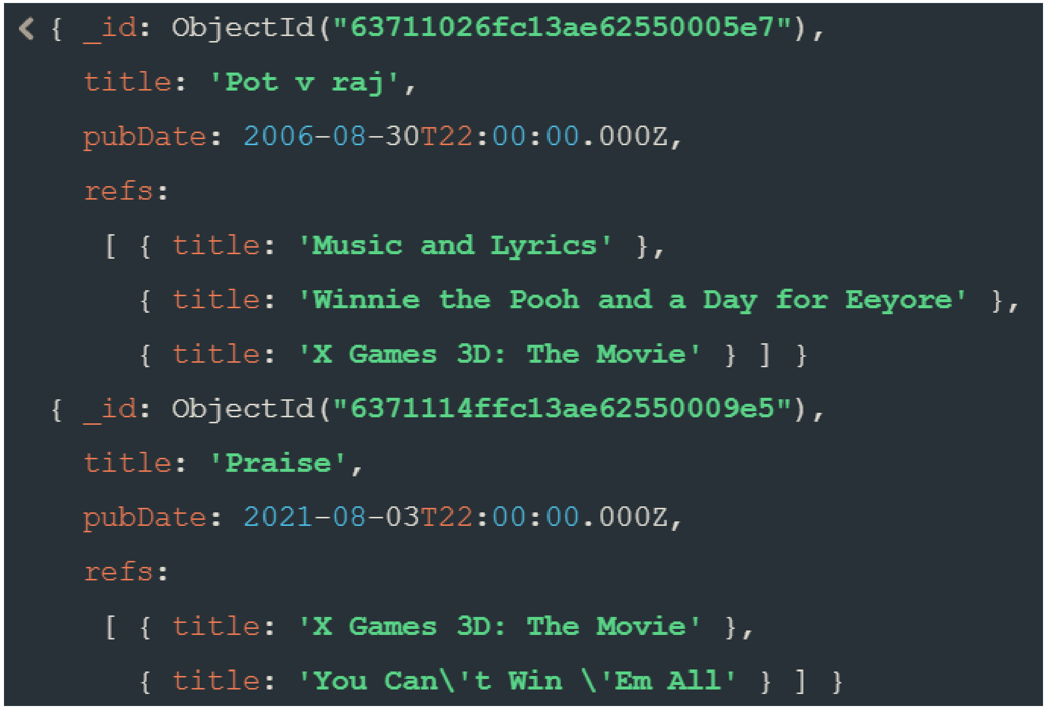


```
⌄ sections: Array
  ⌄ 0: Object
      type: "title"
      text: "Laboratory. The second-highest number of artifacts that govern and"
  ⌄ 1: Object
      type: "subsection"
    ⌄ sections: Array
      ⌄ 0: Object
          type: "title"
          text: "Subtitle 1"
      ⌄ 1: Object
          type: "text"
          text: "Text of a subsection 1."
      ⌄ 2: Object
          type: "title"
          text: "Subtitle 2"
      ⌄ 3: Object
          type: "text"
          text: "Text of a subsection 2."
  ⌄ 2: Object
      type: "figure"
      url: "https://dummy-image.com/274.png"
      caption: "Parallel experiences ethics holds"
```

MongoDB - Find the name, the publication date and the title of referenced documents of all documents written after 2000 that reference an article called “X Games 3D: The Movie”



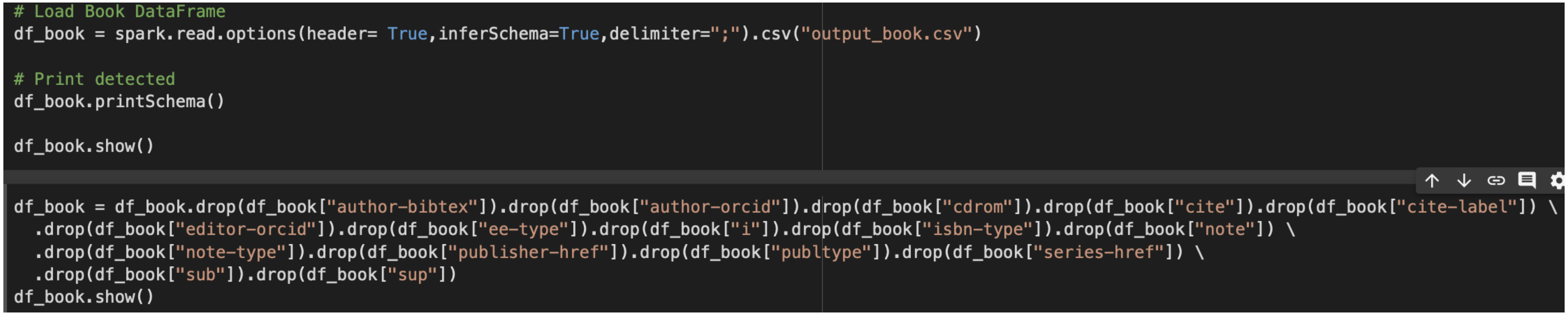
```
db.articles.aggregate([{"$lookup":{from:"articles", localField:"
   bibliography", foreignField:"_id", as:"refs"}},{"$match": {"$and
   ": [{"refs.title": "X Games 3D: The Movie"}, {"pubDate": {"$gt":
    ISODate(’2000-01-01T00:00:00Z’)}}]}}, {"$project": {"title": 1,
    "pubDate": 1, "refs.title": 1}}])
```



```
< { _id: ObjectId("63711026fc13ae62550005e7"),
    title: 'Pot v raj',
    pubDate: 2006-08-30T22:00:00.000Z,
    refs:
     [ { title: 'Music and Lyrics' },
       { title: 'Winnie the Pooh and a Day for Eeyore' },
       { title: 'X Games 3D: The Movie' } ] }
  { _id: ObjectId("6371114ffc13ae62550009e5"),
    title: 'Praise',
    pubDate: 2021-08-03T22:00:00.000Z,
    refs:
     [ { title: 'X Games 3D: The Movie' },
       { title: 'You Can\'t Win \'Em All' } ] }
```

Apache Spark – Data Loading

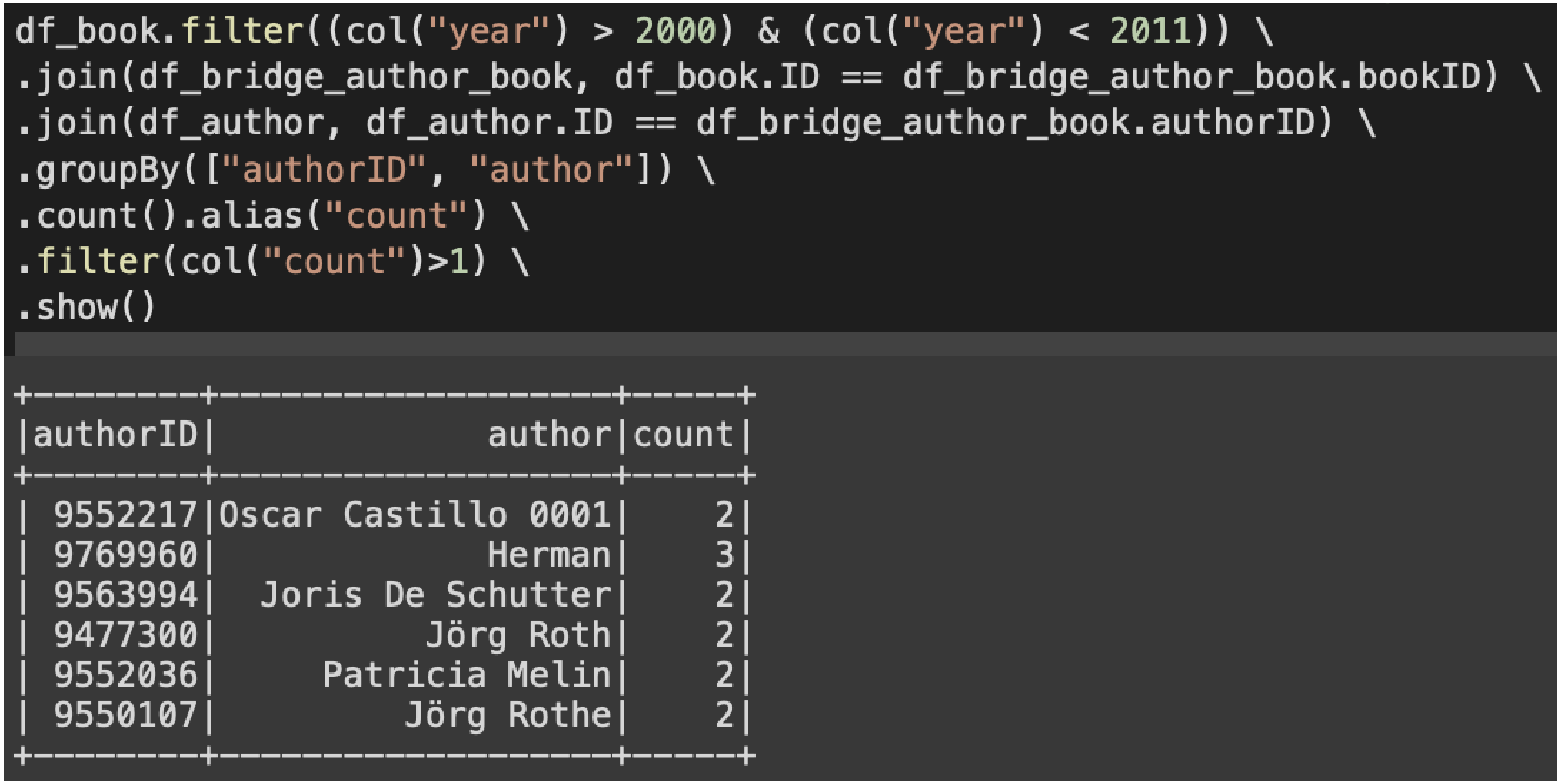
Example: loading the "Book" DataFrame



```
# Load Book DataFrame
df_book = spark.read.options(header= True,inferSchema=True,delimiter=";").csv("output_book.csv")

# Print detected
df_book.printSchema()

df_book.show()
```

```
df_book = df_book.drop(df_book["author-bibtex"]).drop(df_book["author-orcid"]).drop(df_book["cdrom"]).drop(df_book["cite"]).drop(df_book["cite-label"]) \
  .drop(df_book["editor-orcid"]).drop(df_book["ee-type"]).drop(df_book["i"]).drop(df_book["isbn-type"]).drop(df_book["note"]) \
  .drop(df_book["note-type"]).drop(df_book["publisher-href"]).drop(df_book["publtype"]).drop(df_book["series-href"]) \
  .drop(df_book["sub"]).drop(df_book["sup"])
df_book.show()
```

Apache Spark – Show how many books have been written by each author, considering only books written between 2001 and 2010 and authors that have written at least 2 books



```
df_book.filter((col("year") > 2000) & (col("year") < 2011)) \
.join(df_bridge_author_book, df_book.ID == df_bridge_author_book.bookID) \
.join(df_author, df_author.ID == df_bridge_author_book.authorID) \
.groupBy(["authorID", "author"]) \
.count().alias("count") \
.filter(col("count")>1) \
.show()
```

```
+--------+-------------------+-----+
|authorID|             author|count|
+--------+-------------------+-----+
| 9552217|Oscar Castillo 0001|    2|
| 9769960|             Herman|    3|
| 9563994|  Joris De Schutter|    2|
| 9477300|          Jörg Roth|    2|
| 9552036|     Patricia Melin|    2|
| 9550107|         Jörg Rothe|    2|
+--------+-------------------+-----+
```

Preprocessing

Neo4j:

- DBLP to CSV parser - https://github.com/ThomHurks/dblp-to-csv

MongoDB:

- Dataset generator - https://www.mockaroo.com/
- Python script to inject subsections
- Python script for converting date strings in MongoDB Date objects

Apache Spark:

- Python script to reduce the size of the CSV files