Cloud Data Infrastructure
Chapter 1 : NoSQL Data Model

Juliette Danel
Polytechnic Institute of Paris
juliette.danel@ip-paris.fr

Godefroy Lambert
Polytechnic Institute of Paris
godefroy.lambert@ip-paris.fr

Marius Ortega
Polytechnic Institute of Paris
marius.ortega@ip-paris.fr

9 February 2024

1 Dataset choice

We chose to work on the Internet Movie Database (IMDb). IMDb is a famous online database archiving a lot of information on movies, series and even video games. As the recommended website, is currently down, we directly downloaded the datasets that are freely available on the IMDb website. Therefore, we are working on a heavy dataset (approximately 6.84 Go) composed of seven tables, namely:

- **title.akas**: Contains alternative titles for titles identified by *titleId*. Includes details such as title localization, region, language, title types, and attributes.
- **title.basics**: Provides basic information about titles, including title type, primary and original titles, whether it's an adult title, release years, runtime, and associated genres.
- **title.crew**: Contains information about the crew associated with titles, specifically directors and writers identified by unique alphanumeric identifiers *nconsts*.
- **title.episode**: Pertains to TV episodes, linking episodes *tconst* to their parent TV series *parentTconst* and providing season and episode numbers.
- **title.principal**s: Contains information about individuals associated with titles *tconst*, including their roles (category), job titles, and characters played.

- **title.ratings**: Stores ratings-related data for titles, including the average rating and the number of votes received.
- **name.basics**: Provides information about individuals (names), including birth and death years, primary professions, and titles they are known for.

Our dataset contains two types of joins: one based on titles, facilitated through the *tconst* attribute, and another involving individuals through the *nconst* attribute.

Let's note that our data stored in tsv (tab separated values) files, making them ideal to model them in a relational database.

2 Use cases

2.1 End-user view

We asked ourselves what could be four frequent requests of a user of an application using the IMDB database, and we came up with the following four queries:

- Select the top 10 movie of the year X shorter than 2 hours
- Select a movie containing the string "XXX" in at least one of its titles
- Actors that played in at least 3 adult films
- Select the director that participated in the most film for each genre

Those request can be frequently asked and are pretty basic queries.

2.2 Data analyst view

This time, we considered the queries that a data analyst might have about the IMDB database. We finally decided to use the following four requests:

- Count the number of actors dead before the end of a series
- Select the top Y favorite genre of the year X
- TOP X of average number of appearance for an actor per series
- Select the average number of film released per decade per region

These queries are heavier than the previous four, but they would be less frequently asked.

3 Database schema

To provide visual support for our rationales, we express the previously defined queries in technical terms using an Entity-Association Schema created based on IMDb data (figure 1).

Also, to ease the reading of the tables, we renamed them as follows:

- title.akas → filmAkas
- title.basics → film
- title.crew → crewMember
- title.episode → episode
- title.principals → castingMember
- title.ratings → rating
- name.basics → person

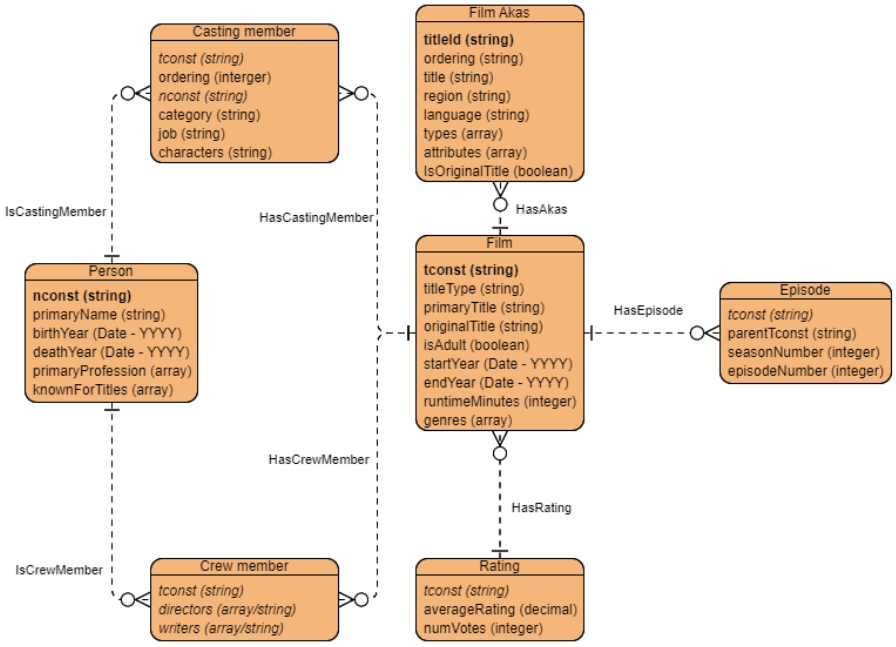


Figure 1: Entity Relationship diagram

Note: For each of the previously presented queries, we provide a technical summary in the form of a list of requirements.

3.1 End-user queries

3.1.1 Select the top 10 movies of the year X shorter than 2 hours

Tables and attributes

- **film** (tconst, startYear, primaryTitle, titleType, runTimeMinutes)
- **rating** (tconst, averageRating, numVotes)

Joins

Inner join between **film** and **rating** on tconst.

Projections

tconst, primaryTitle, startYear, ratingScore, titleType, runTimeminutes

Filters, and aggregations and operations

- titleType equals to "movie"
- startYear equals to the specific year we are looking for
- ratingScore equals to $averageRating * log(numVotes + 1)$
- runTimeMinutes is less than 120
- order by descending ratingScore
- keep the 10 first results

3.1.2 Select a title containing the string "XXX" in at least one of its titles

Tables and attributes

- **film** (tconst, primaryTitle, originalTitle)
- **filmAkas** (titleId, title)

Joins

Left join between **film** and **filmAkas**, left on tconst, right on titleId.

Projections

tconst, primaryTitle, originalTitle, title

Filters, aggregations and operations

- keep rows where primaryTitle or originalTitle or title contains the specific string we are looking for

3.1.3 Actors that played in at least 3 adult films

Tables and attributes

- **film** (tconst, isAdult)
- **CastingMember** (tconst, nconst, job)
- **Person** (nconst, primaryName)

Joins

- Left join Film and CastingMember on **tconst**
- Right join Person and CastingMember on **nconst**

Projections

Person.nconst, Person.primaryName, count(Film.tconst)

Filters, aggregations and operations

- Filter Film on isAdult = $True$
- Left join Film and CrewMember
- Filter CrewMember on job = *Actor*
- Right join Person and CastingMember
- Group by **nconst**
- Count the number of **tconst**

3.1.4 Select the director that participated in the most title for each genre

Tables and attributes

- **film** (tconst, genres)
- **crewMember** (tconst, directors)

Joins

Inner join between **film** and **crewMember** on tconst.

Projections

tconst, genres, directors

Filters, aggregations and operations

- for each director, count the number of titles per genre
- for each genre, keep the director that has the most titles

3.2 Data Analyst queries

3.2.1 Count the number of actors dead before the end of a series

Tables and attributes

- **CastingMember** (nconst, tconst, job)
- **Film** (tconst, startYear, endYear, typeTitle)

Joins

Inner join **CastingMember** and **Film** on tconst

Projections

Count(nconst)

Filters, aggregations and operations

- Filter on titleType = *tvSeries*
- Join Film and CastingMember
- Group by Film
- Having at least an actor for which deathYear >= startYear and deathYear <= endYear
- Count the output

3.2.2 Select the top Y favorite genre of the year X

Tables and attributes

- **Film** (tconst, genre, startYear)
- **Rating** (tconst, averageRating, numVotes)

Joins

Left join Film and Rating on tconst

Projections

tconst, startYear, averageRating $\times \log(numVotes)$

Filters, aggregations and operations

- Filter film on year X
- Left join Film and Rating
- Calculate the score : averageRating $\times \log(numVotes)$
- Order by score in descending order
- Keep the top Y results

3.2.3 Average number of appearance for an actor per series

Tables and attributes

- **film** (tconst, titleType, primaryTitle)
- **castingMember** (tconst, nconst, category)
- **episode** (tconst, parentTconst)

Joins

Inner join between **film** and **episode**, left on tconst, right on parentTconst

Inner join between **episode** and **castingMember** on tcont

Projections

tconst, primaryTitle

Filters, aggregations and operations

- titleType equals to "tvSeries"
- for each parentTconst for each series, count the number of episodes linked to each actor
- compute the average appearance for each parentTconst per series

3.2.4 Select the average number of movies released per decade per region

Tables and attributes

- **film** (tconst, titleType, startYear)
- **filmAkas** (titleId, isOriginal, region)

Joins

Inner join between **film** and **filmAkas**, left on tconst, right on titleId.

Projections

tconst, titleType, startYear, isOriginal, region

Filters, aggregations and operations

- titleType equals to "movie"
- isOriginal equals to True
- modify the startYear to only keep the decade (ex: 1956 → 1950, 2020 → 2020, etc.)
- group by region and decade
- count the number of movies made by year for each decade

4 Statistics

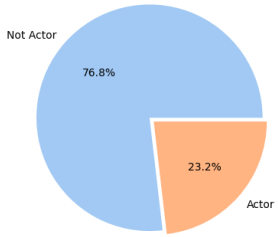
4.1 Person

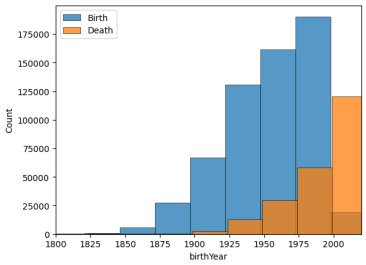
From the associated statistics, we see that the table **person** contains **13221794** documents, each corresponding to a person (it could an actor, director, etc.).

	nconst	primaryName	birthYear	deathYear	primaryProfession
count	13221794	13221787	13221794	13221794	10567492
unique	13221794	10202708	529	471	21690
top	nm0000001	Alex	\N	\N	actor
freq	1	466	12617124	12996973	2384332

Table 1: Summary statistics - Person

Among these documents, we chose to show cardinality for a filter on actor profession showing that the table Person has 23.2% of actors (see figure 2a). Another interesting aspect about this table is the distribution of Person.birthYear and Person.deathYear that are showed in figure 2b.





(a) Pie Chart : Actor proportion in the Table Person

(b) Histogram : Birth and death year distributions

Finally, the table Person is linked to tables CastingMember and CrewMember by a $1 \rightarrow n$ cardinality relationship as expressed in the E/A diagram (figure 1).

4.2 Film

	count	unique	top	freq
tconst	10521657	10521657	tt0000001	1
titleType	10521657	11	tvEpisode	8043546
primaryTitle	10521640	4726114	Episode #1.1	50323
originalTitle	10521640	4749379	Episode #1.1	50323
isAdult	10521657	14	0	10123910
startYear	10521657	153	\N	1394112
endYear	10521657	97	\N	10403566
runtimeMinutes	10521657	950	\N	7318857
genres	10521639	2360	Drama	1205066

Table 2: Film - summary statistics

As observed in the summary statistics (table 2), ImDb counts 10 521 657 different films. In other words, the table Film contains 10 521 657 documents.

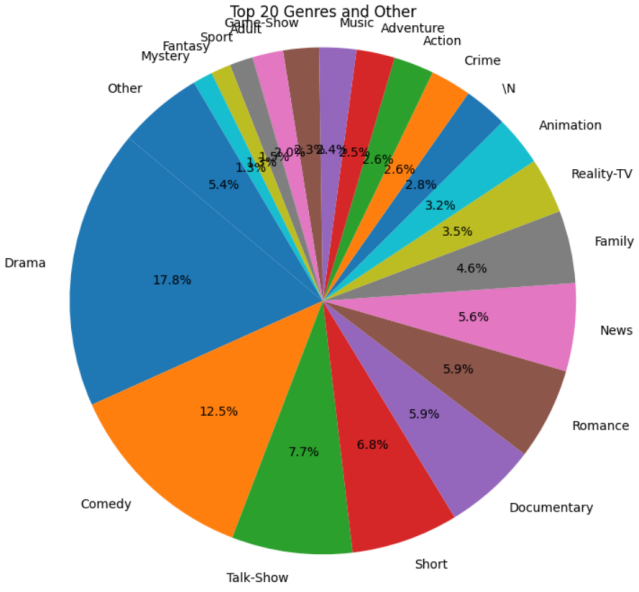


Figure 3: Top 20 genres of titles

Finally, we can notice Film is the table tied to the most table and consequently has multiple cardinalities. All tables Episode, Akas, Rating, CastingMember, CrewMember have a $1 \rightarrow n$ cardinality with Film.

4.3 CrewMember

CrewMember table counts 10,523,978 unique documents (table 3).

	tconst	directors	writers
count	10,523,978	10,523,978	10,523,978
unique	10,523,978	956,976	1,320,269
top	tt0000001	\N	\N
freq	1	4,481,658	5,080,160

Table 3: Summary statistics - CrewMember

We can observe that about half the titles do not have a director or a writer.

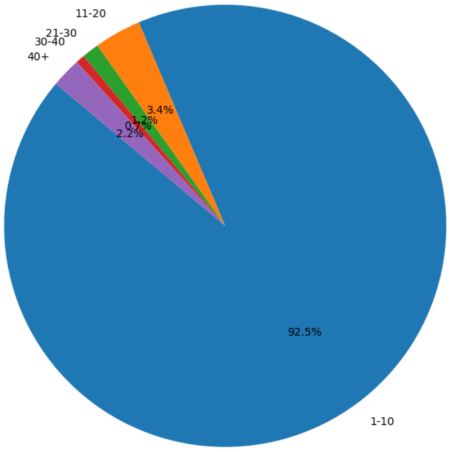


Figure 4: Distribution of Directors by Number of Films Directed

CrewMember have a n $\rightarrow$ 1 cardinality with Film, and also a n $\rightarrow$ 1 cardinality with Person.

4.4 CastingMember

CastingMember table counts 60,307,595 documents (table 4).

	tconst	ordering	nconst	category	job	characters
count	60307595	6.030760e+07	60307595	60307595	60307595	60307595
unique	9531725	NaN	5256726	12	40541	2885044
top	tt2028230	NaN	nm0914844	actor	N/	N/
freq	10	NaN	24588	13325054	50420872	31321115

Table 4: Summary statistics - CrewMember

We decided to look at the cardinality and distribution for the column "category". We can observe 12 different categories, the top 3 categories being containing more than 50% of the categories.

CastingMember have a n $\rightarrow$ 1 cardinality with Film.

CastingMember have a n $\rightarrow$ 1 cardinality with Person.

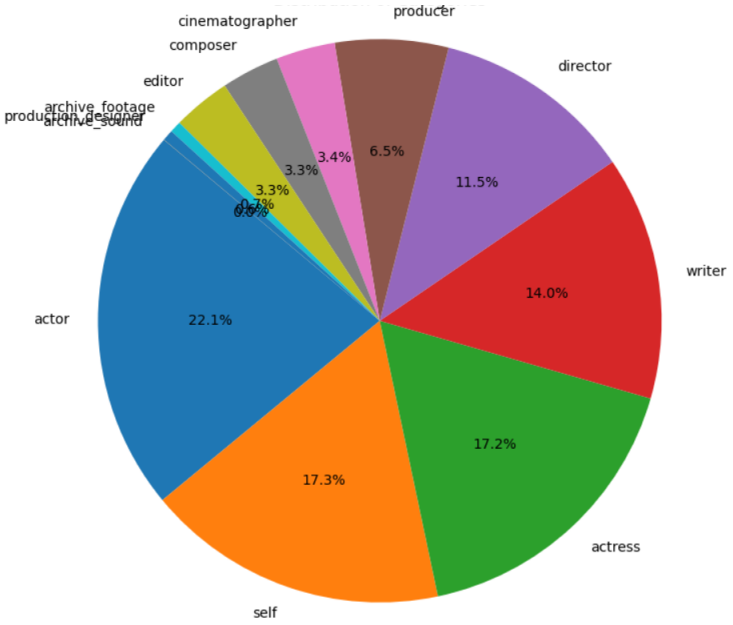


Figure 5: Description of categories

4.5 FilmAkas

Akas table contains 38,576,732 documents, as showed in the summary statistics (figure 5).

	attributes	isOriginalTitle	language	region	title	titleId
count	38576732	38576732	38576732	38576615	38576714	38576732
unique	187	5	108	248	4509809	7538269
top	\N	0	\N	JP	Episodio #1.1	tt0088814
freq	38303491	23640442	6913801	4637497	100294	251

Table 5: Summary statistics - Film

We chose to visualize the reparation of film production among the top 10 most translated languages over the world. We notice that the top 7 languages are

relatively even, and English comes in 8th position. This comes from the fact that United States (Hollywood) produce most films. Consequently, there don't need to be translated in English as it is the primary language.

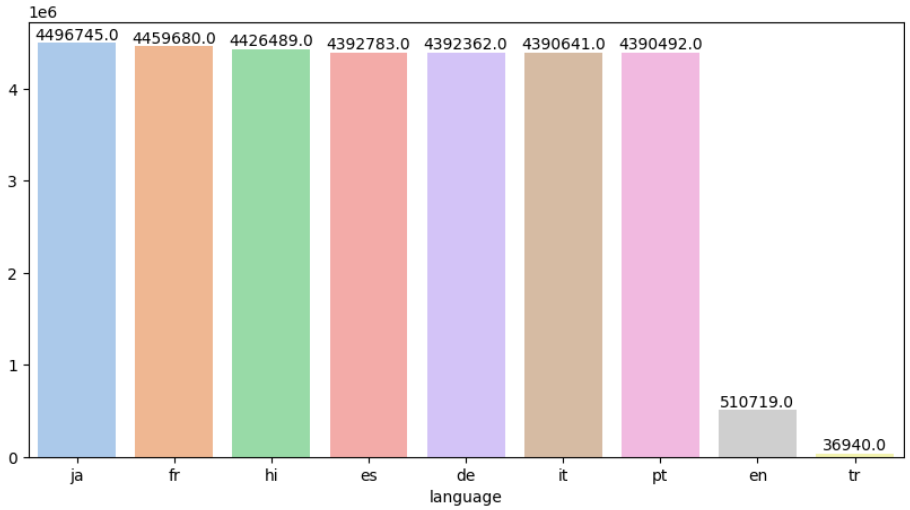


Figure 6: Bar plot : Partitioning of film translation among top 10 languages

Akas is only linked to Film with a $n \to 1$ relationship as referenced by the E/A diagram (figure 1).

4.6 Table Episode

	tconst	parentTconst	seasonNumber	episodeNumber
count	8043359	8043359	8043359	8043359
unique	8043359	199279	303	15782
top	tt0041951	tt12164062	1	N/
freq	1	18593	4100559	1633270

Table 6: Summary statistics - Episode

Episode table counts 8,043,359 unique documents (table 6). Episode's table is linked n $\to$ 1 with the film table.

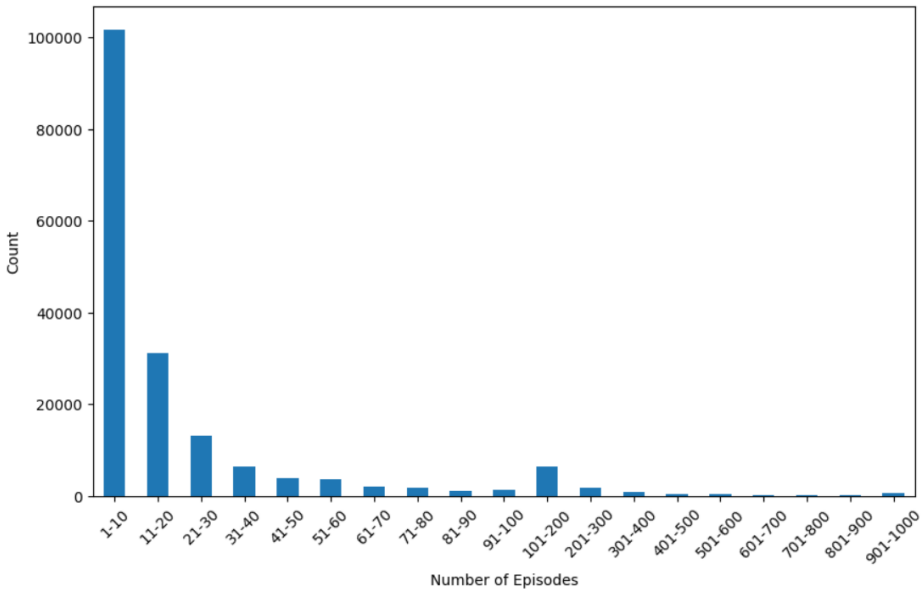


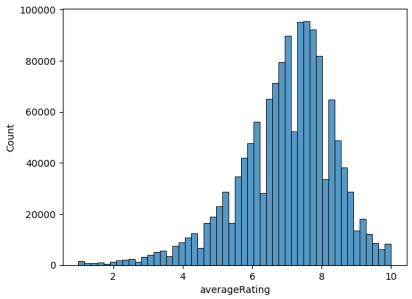
Figure 7: Number of Episodes per series

4.7 Rating

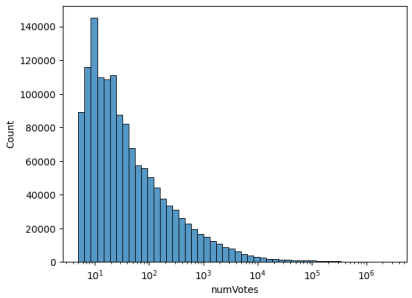
	averageRating	numVotes
count	1.397149e+06	1.397149e+06
mean	6.955867e+00	1.037138e+03
std	1.385599e+00	1.764119e+04
min	1.000000e+00	5.000000e+00
25%	6.200000e+00	1.100000e+01
50%	7.100000e+00	2.600000e+01
75%	7.900000e+00	1.010000e+02
max	1.000000e+01	2.850766e+06

Table 7: Summary statistics - Ratings

In order to better grasp the attributes of Rating, we plot the distribution of each table. From there, we denote that the rating produces a slightly skewed Gaussian distribution around the rating 8/10. Votes distribution is consistent with our expectations ; a few films have a lot of votes. Note : we had to plot Votes distribution in log scale for the x axis to make it readable.



(a) Histogram : Ratings distribution



(b) Histogram : Votes distribution

As a conclusion for this table's section, we analyze it's cardinality using the E/A diagram (figure 1). Ratings as $n \rightarrow 1$ cardinality with Film which is the only table it is tied to.

5 Conclusion & discussion

We have a rather heavy dataset, nearly 7 Go, divided into seven tables that enable us to execute various queries. Although the data is well-organized, it's worth noting that certain queries can be computationally intensive, despite the optimization efforts such as retaining only the year in date fields to minimize data size.