

Information Integration project

Integrating heterogeneous data sources into a unified global schema

Goal of the project

Accomplishing a **domain-based integration** of multiple and heterogeneous **data sources**, merging them into a single global view that can be queried by the users.

The global schema should be able to satisfy queries not trivially satisfiable by accessing directly the data sources.



GAV – Mappings (1)

In GAV approach, the (sound) mappings are assertions as such:

$$\forall x. \phi_S(x) \rightarrow g(x)$$

Where:

- S is the source schema
- ϕ_S is a CQ over the source DB
- g is a predicate of arity n that indicates an element of G



GAV (Global As View): the global schema is a view of the set of sources



GAV – Retrieved global database (2)

Given $I = \langle G, S, M \rangle$ and source database C for S , we denote the retrieved global database as $M(C)$, that is the database obtained by:

- Applying the CQs specified in the mappings
- Transferring the tuples from the sources to the global DB



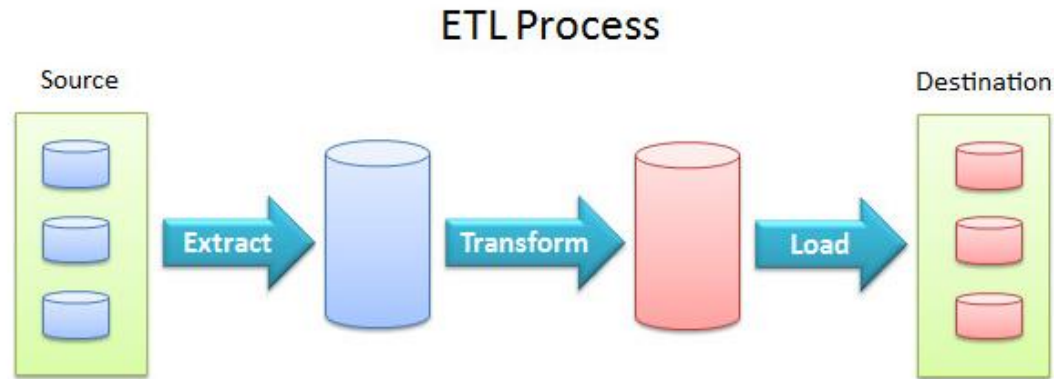
Data integration using Pentaho (1)

The end result of the project is a **one-shot materialization** of the global schema, achieved by mappings in the form of Pentaho transformations.



Data integration using Pentaho (2)

Pentaho Kettle is a tool that allows loading input data in **heterogeneous formats and from different sources**, performing the transformations needed in order to map the information as requested.



Finally, the data is materialized in output as a SQL database in the example shown for this project.



Domain and data sources (1)



Critic
reviews



Directors



Actors



Movies



Users
Ratings



Production
studios



Oscar
awards



Domain and data sources (2)

- 5 Data sources

kaggle



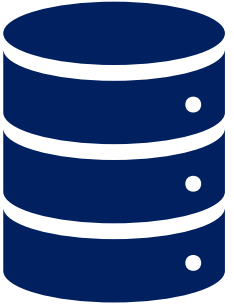
- 11 Files (TSV and CSV)

- +45.000 Correctly reconciled movies across datasets



Domain and data sources (3)

IMDb



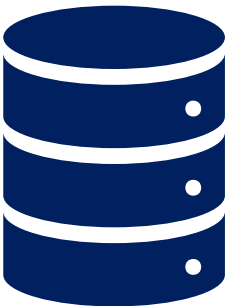
- title.principals.tsv
- title.basics.tsv
- name.basics.tsv

kaggle 



- the_oscar_award.csv

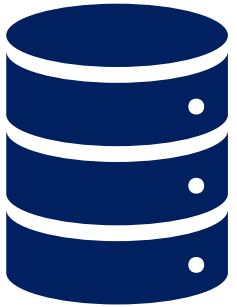
kaggle 



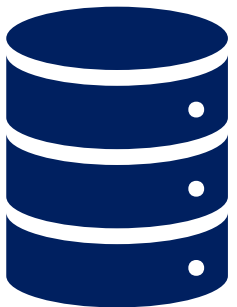
- ratings.csv
- movies_metadata.csv
- links.csv



Domain and data sources (4)



- studios.csv
- movies.csv



- rotten_tomatoes_critic_reviews.csv
- rotten_tomatoes_movies.csv



Domain and data sources – Reconciliation issues (5)

➤ Physical heterogeneity

Data stored in different formats: CSV and TSV

Easily mitigated by Pentaho transformations that can handle multiple input formats

➤ Conceptual heterogeneity

- Different datasets uniquely identify movies in diverse formats: IDs both in string and number data types
- Some datasets refer to movies with their original title, some with their English title, some have both fields



Domain and data sources – Reconciliation issues (6)

➤ Contextual heterogeneity

Datasets can refer to the same field using different representation: both IMDB and Movielens sources contain the IMDB ID, but the latter lacks a prefix in the string.

```
import pandas as pd
import os

input_file = os.path.abspath("../datasets/links.csv")
output_file = os.path.abspath("../datasets/links_upd.csv")
column_name = "imdbId"

try:
    # Load the CSV file into a DataFrame
    df = pd.read_csv(input_file)

    # Ensure the specified column exists
    if column_name not in df.columns:
        raise ValueError(f"Column '{column_name}' not found in the CSV file.")

    # Update the column by prepending 'tt' to each value
    df[column_name] = 'tt' + df[column_name].astype(str)

    # Save the updated DataFrame to a new CSV file
    df.to_csv(output_file, index=False)
    print(f"Updated CSV file saved to {output_file}")

except Exception as e:
    print(f"An error occurred: {e}")
```

Python preprocessing required to solve the heterogeneity between sources



Source schema (1)

IMDBTitleBasics_{/10} (*tconst, titleType, primaryTitle, originalTitle, isAdult, startYear, endYear, runtimeMinutes, genres*)

IMDBTitlePrincipals_{/6} (*tconst, ordering, nconst, category, job, characters*)

IMDBNameBasics_{/6} (*nconst, primaryName, birthYear, deathYear, primaryProfession, knownForTitles*)

MovielensRatings_{/4} (*userId, movieId, rating, timestamp*)

MovielensMoviesMetadata_{/24} (*adult, belongsToCollection, budget, genres, homepage, id, imdb_id, original_language, originalTitle, overview, popularity, poster_path, production_companies, production_countries, release_date, revenue, runtime, spoken_languages, status, tagline, title, video, vote_average, vote_count*)

MovielensLinks_{/3} (*movieId, imdbId, tmdbId*)



Source schema (2)

TheOscarAward_{/7} (*year_film, year_ceremony, ceremony, category, name, film, winner*)

LetterboxStudios_{/2} (*id, studio*)

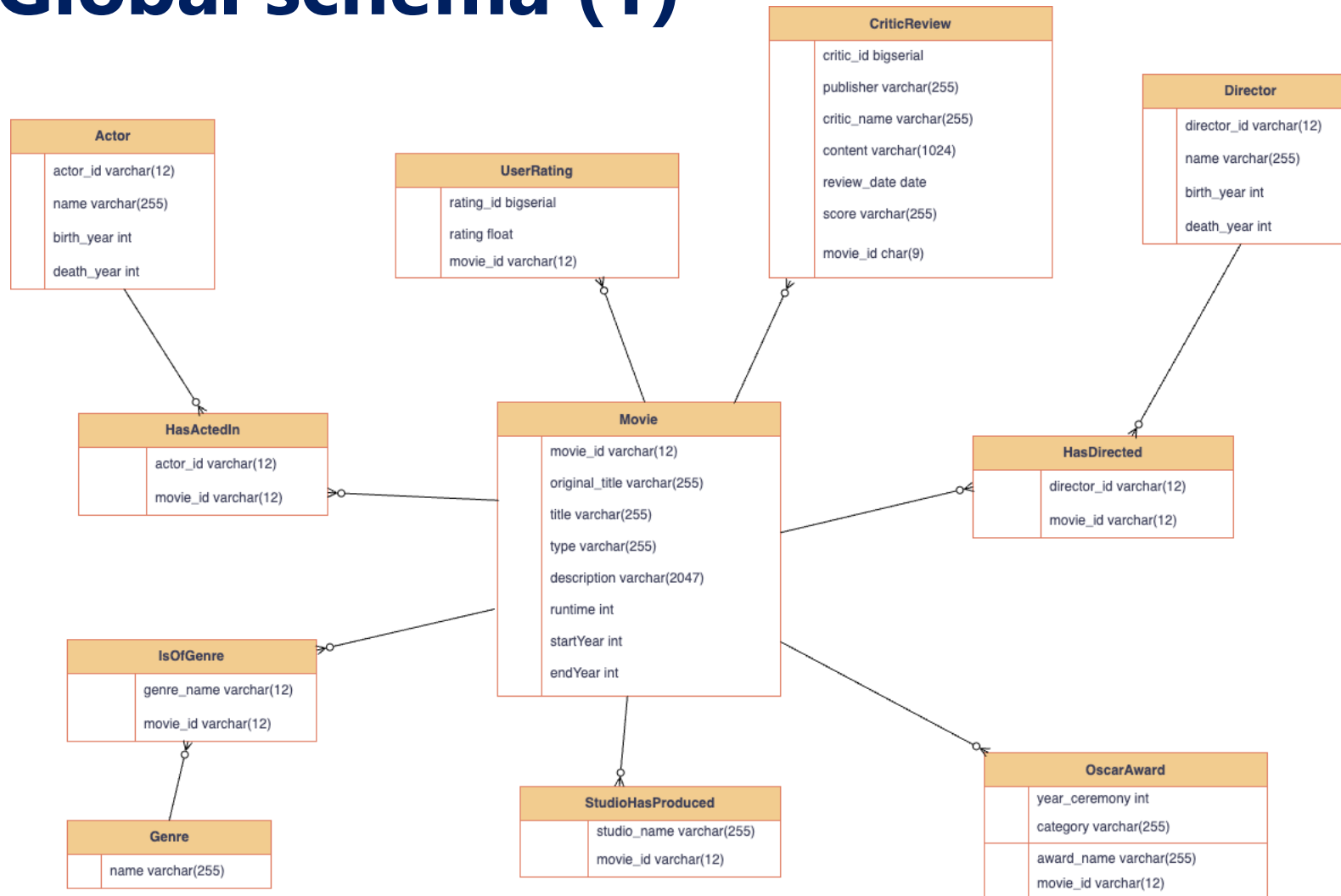
LetterboxMovies_{/7} (*id, name, date, tagline, description, minute, rating*)

RottenTomatoesCriticReviews_{/8} (*rotten_tomatoes_link, critic_name, top_critic, publisher_name, review_type, review_score, review_date, review_content*)

RottenTomatoesMovies_{/22} (*rotten_tomatoes_link, movie_title, movie_info, critics_consensus, content_rating, genres, directors, authors, actors, original_release_date, streaming_release_date, runtime, production_company, tomatometer_status, tomatometer_rating, tomatometer_count, audience_status, audience_rating, audience_count, tomatometer_top_critics_count, tomatometer_fresh_critics_count, tomatometer_rotten_critics_count*)



Global schema (1)



Global schema (2)

Movie_{/8} (*movie_id, original_title, title, type, description, runtime, startYear, endYear*)

UserRating_{/3} (*rating_id, rating, movie_id*)

CriticReview_{/7} (*critic_id, publisher, critic_name, content, review_date, score, movie_id*)

Actor_{/4} (*actor_id, name, birth_year, death_year*)

HasActedIn_{/2} (*actor_id, movie_id*)

StudioHasProduced_{/2} (*studio_name, movie_id*)



Global schema (3)

Director_{/4} (*director_id, name, birth_year, death_year*)

HasDirected_{/2} (*director_id, movie_id*)

Genre_{/1} (*name*)

IsOfGenre_{/2} (*genre_name, movie_id*)

OscarAward_{/4} (*year_ceremony, category, award_name, movie_id*)



Conjunctive GAV FOL Mappings (1)

Movie:

$\forall id, ot, ti, ty, dc, rt, sy, ey. (\exists ia, g, ad, btc, bg, hp, mid, ol, ott, pop, pp, pcp, pcn, rd, rev, rut, sl, st, tl, tit, va, vc. (IMDBTitleBasics(id, ty, ti, ot, ia, sy, ey, rt, g) \wedge$
 $MovielensMovieMetadata(ad, btc, bg, hp, mid, id, ol, ott, dc, pop, pp, pcp, pcn, rd, rev, rut, sl, st, tl, tit, va, vc))) \rightarrow Movie(id, ot, ti, ty, dc, rt, sy, ey)$

UserRating:

$\forall rid, r, mid. (\exists uid, mid2, ts, tid. (MovielensRatings(uid, mid2, r, ts) \wedge$
 $MovielensLinks(mid2, mid, tid))) \rightarrow UserRating(rid, r, mid)$

Genre:

$\forall g. (\exists id, ty, ti, ot, ia, sy, ey, rt. (IMDBTitleBasics(id, ty, ti, ot, ia, sy, ey, rt, g))) \rightarrow$
 $Genre(g)$

IsOfGenre:

$\forall g, id. (\exists id, ty, ti, ot, ia, sy, ey, rt. (IMDBTitleBasics(id, ty, ti, ot, ia, sy, ey, rt, g)))$
 $\rightarrow IsOfGenre(g, id)$



Conjunctive GAV FOL Mappings (2)

Actor:

$\forall \text{aid, n, by, dy.} (\exists \text{pp, kft.} (\text{IMDBNameBasics}(\text{aid, n, by, dy, pp, kft})))$
 $\rightarrow \text{Actor}(\text{aid, n, by, dy})$

HasActedIn:

$\forall \text{aid, mid.} (\exists \text{o, ca, j, ch.} (\text{IMDBTitlePrincipals}(\text{mid, o, aid, ca, j, ch}) \wedge \text{ca} = \text{'actor'}))$
 $\rightarrow \text{HasActedIn}(\text{aid, mid})$

Director:

$\forall \text{did, n, by, dy.} (\exists \text{pp, kft.} (\text{IMDBNameBasics}(\text{did, n, by, dy, pp, kft})))$
 $\rightarrow \text{Actor}(\text{did, n, by, dy})$

HasDirected:

$\forall \text{did, mid.} (\exists \text{o, ca, j, ch.} (\text{IMDBTitlePrincipals}(\text{did, o, aid, ca, j, ch}) \wedge \text{ca} = \text{'director'}))$
 $\rightarrow \text{HasDirected}(\text{did, mid})$

StudioHasProduced:

$\forall \text{sn, mid.} (\exists \text{mid2, n, d, tl, dc, m, ty, ot, ia, ey, rt, g, r.} (\text{LetterboxStudios}(\text{mid2, sn}) \wedge$
 $\text{LetterboxMovies}(\text{mid2, n, d, tl, dc, m, r}) \wedge$
 $\text{IMDBTitleBasics}(\text{mid, ty, n, ot, ia, d, ey, rt, g}))) \rightarrow \text{StudioHasProduced}(\text{sn, mid})$



Conjunctive GAV FOL Mappings (3)

CriticReview:

$\forall \text{cid, p, cn, ct, rd, sc, mid.}$

$(\exists \text{rtl, tc, rt, mt, mi, cc, cr, g, d, a, ord, srd, r, pc, ts, tr, tc, as, ar, ac, ttcc, tfcc, trcc, ty, ot, ia, ey, rt2, g2. (RottenTomatoesCriticReviews (rtl, cn, tc, p, rt, sc, rd, ct) \wedge$

$\text{RottenTomatoesMovies}(\text{rtl, mt, mi, cc, cr, g, d, a, ord, srd, r, pc, ts, tr, tc, as, ar, ac, ttcc, tfcc, trcc}) \wedge \text{IMDBTitleBasics}(\text{mid, ty, mt, ot, ia, d, ey, rt2, g2})) \rightarrow \text{CriticReview}(\text{cid, p, cn, ct, rd, sc, mid})$

OscarAward:

$\forall \text{yc, ca, an, mid.}$

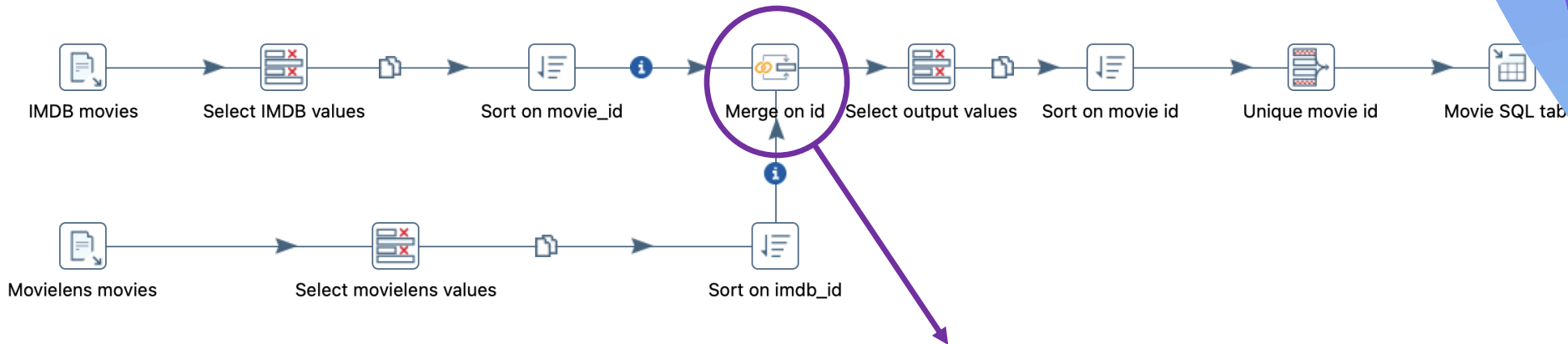
$(\exists \text{yf, ce, n, f, w, ty, ot, ia, ey, rt, g. TheOscarAward}(\text{yf, yc, ce, ca, n, f, w}) \wedge$

$\text{IMDBTitleBasics}(\text{mid, ty, f, ot, ia, yf, ey, rt, g}) \wedge \text{w='true'}) \rightarrow \text{OscarAward}(\text{yc, ca, an, mid})$



Pentaho implementation (1)

Movie transformation



The screenshot shows the 'Merge join' configuration window in Pentaho. The 'Step name' is 'Merge on id'. The 'First Step' is 'Sort on movie_id' and the 'Second Step' is 'Sort on imdb_id'. The 'Join Type' is set to 'INNER'. Below these settings are two tables for defining key fields for each step.

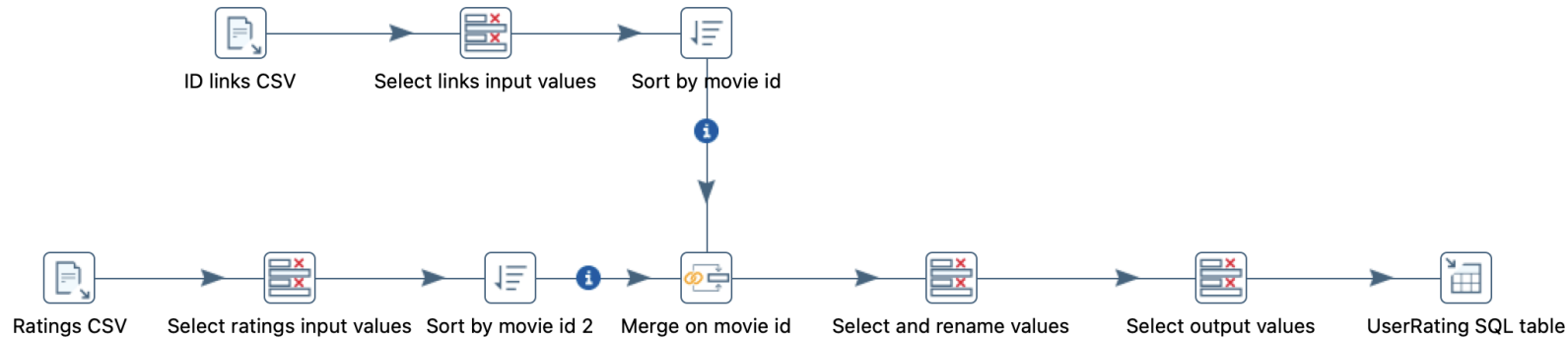
Keys for 1st step:		Keys for 2nd step:	
#	Key field	#	Key field
1	movie_id	1	imdb_id

At the bottom of the window, there are buttons for 'Get key fields', 'Help', 'OK', and 'Cancel'.

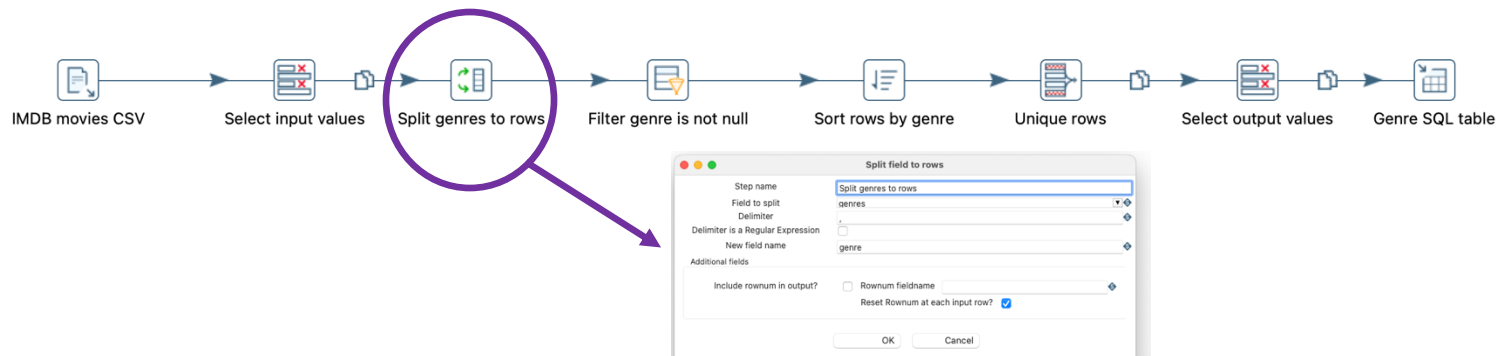


Pentaho implementation (2)

UserRating transformation



Genre transformation

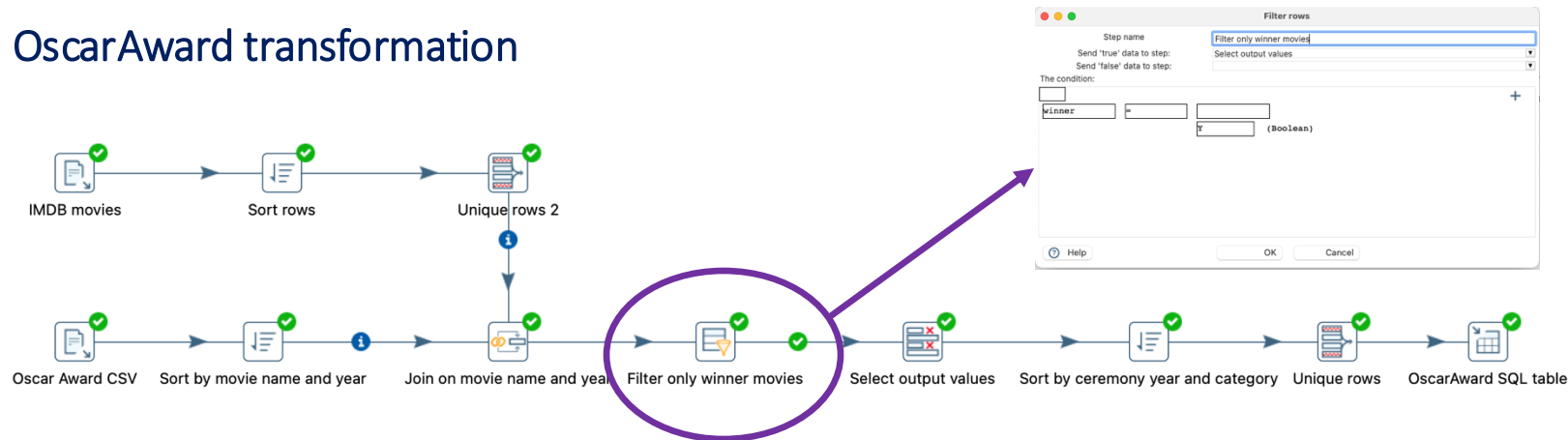


Pentaho implementation (3)

IsOfGenre transformation



OscarAward transformation

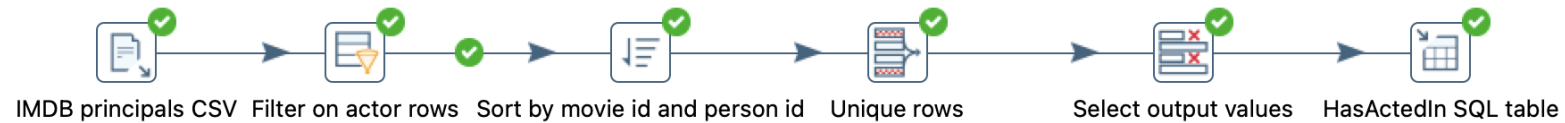


Pentaho implementation (4)

Actor transformation



HasActedIn transformation



Pentaho implementation (5)

Director transformation

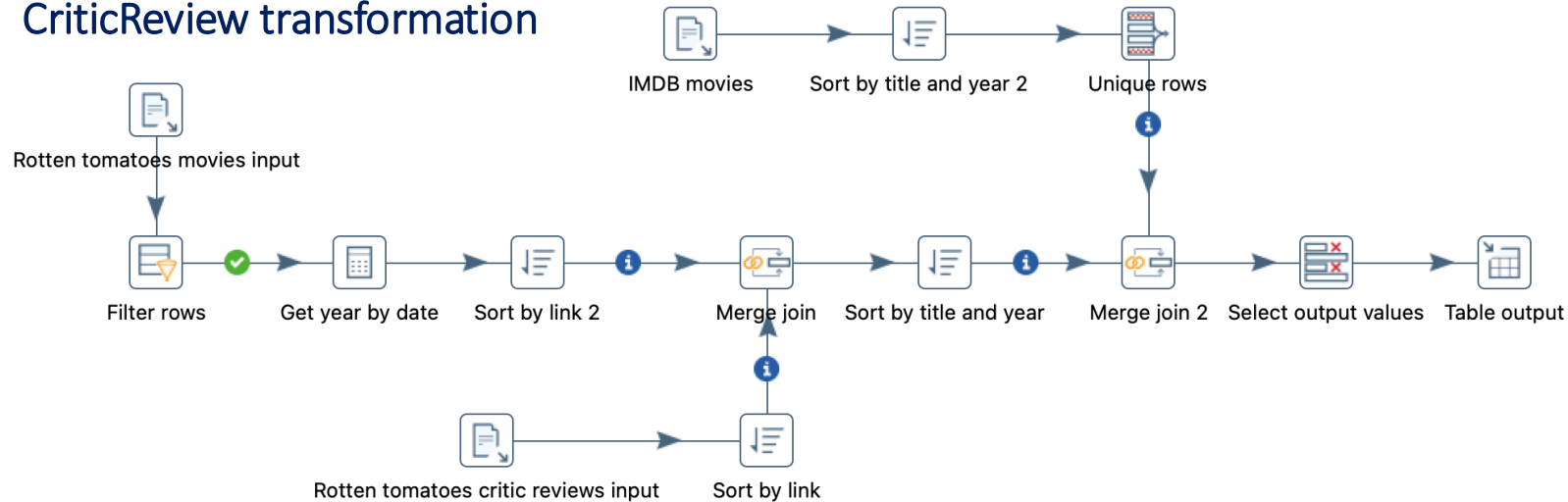


HasDirected transformation

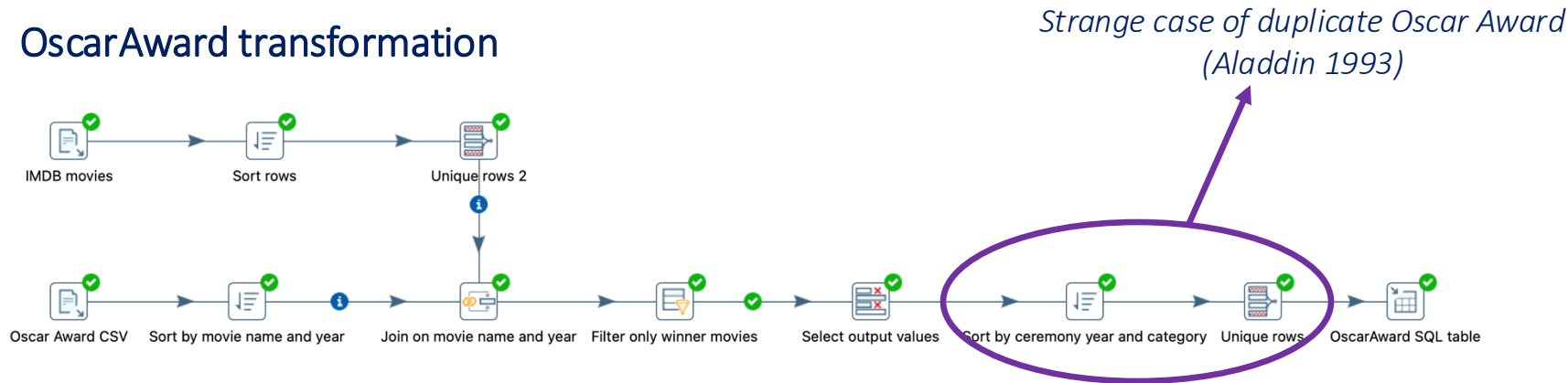


Pentaho implementation (6)

CriticReview transformation



OscarAward transformation



Some statistics...

- In the RGBD (Retrieved Global Database)
 - >11.000.000 movies
 - >26.000.000 reconciled user ratings
 - >20.000.000 actor appearances
- 17 minutes each for CriticRating, OscarAward, StudioHasProduced transformations (ran on M1 Pro)
- 7,09 GBs of source data



Query answering

Given the source database C , now that we have obtained the **RGBD (Retrieved Global Database)** $M(C)$ by performing a one-shot materialization, we can query it

We have proved that, given GAV mappings with sound sources and conjunctive queries, we can compute **certain answers** in polynomial time complexity



Query 1

Name of actors born after 1960 participating in a "Best Picture" Oscar-winning movie, criticized by the Independent (UK) publisher

```
select distinct a."name"
from actor a, hasactedin h, movie m, oscaraward o, criticreview c
where a.actor_id = h.actor_id
    and h.movie_id = m.movie_id
    and m.movie_id = o.movie_id
    and m.movie_id = c.movie_id
    and birth_year > 1960
    and o.category = 'BEST PICTURE'
    and c.publisher = 'Independent (UK)'
```

	A-Z name
1	Azharuddin Mohammed Ismail
2	Danny Dorland
3	Dev Patel
4	Dominic West
5	Irrfan Khan
6	Javier Bardem
7	John C. Reilly
8	Joseph Fiennes
9	Josh Brolin
10	Mark Ivanir
11	Martin Clunes

$q = \{(name) \mid$
 $\exists aid, mid, name, by, dy, ot, ti, ty, dc, r, sy, ey, yc, cat, aw, cid, pub, cn, co, rd, sc.$
 $Actor(aid, name, by, dy) \wedge HasActedIn(aid, mid)$
 $\wedge Movie(mid, ot, ti, ty, dc, r, sy, ey) \wedge OscarAward(yc, 'BEST PICTURE', aw, mid)$
 $\wedge CriticReview(cid, 'Independent (UK)', cn, co, rd, sc, mid) \wedge by > 1960$



Query 2

Name of movies of genre Drama produced by "Paramount Pictures" where Cillian Murphy acted

```
select m.title
from genre g, isofgenre i, movie m, hasactedin h, studiohasproduced s, actor a
where g."name" = i.genre_name
      and i.movie_id = m.movie_id
      and m.movie_id = s.movie_id
      and i.movie_id = h.movie_id
      and h.actor_id = a.actor_id
      and s.studio_name = 'Paramount'
      and a."name" = 'Cillian Murphy'
      and g."name" = 'Drama'
```

	A-Z title
1	A Quiet Place Part II
2	On the Edge

$q = \{(\text{title}) \mid$
 $\exists \text{aid, mid, by, dy, ot, ty, dc, r, sy, ey, .}$
 $\text{Actor}(\text{aid}, \text{'Cillian Murphy'}, \text{by, dy}) \wedge \text{HasActedIn}(\text{aid, mid})$
 $\wedge \text{Movie}(\text{mid, ot, title, ty, dc, r, sy, ey}) \wedge \text{IsOfGenre}(\text{'Drama', mid}) \wedge$
 $\text{StudioHasProduced}(\text{'Paramount', 'CillianMurphy'})$



Query 3

Name of critics that have criticized a movie directed by Cristopher Nolan
where Al Pacino acted

query elapsed time to join actors, movies and directors: ~5 minutes

```
select c.critic_name
from actor a, hasactedin h, movie m, hasdirected h2, criticreview c, director d
where a.actor_id = h.actor_id
    and h.movie_id = m.movie_id
    and m.movie_id = h2.movie_id
    and h2.director_id = d.director_id
    and m.movie_id = c.movie_id
    and d.name = 'Christopher Nolan'
    and a.name = 'Al Pacino'
```

	A-Z critic_name
1	Rob Blackwelder
2	Harvey S. Karten
3	Richard Roeper
4	Robin Clifford
5	Jeanne Aufmuth
6	Ed Gonzalez
7	Kevin N. Laforest
8	Frank Swietek
9	Andy Weil
10	Steve Rhodes
11	Kirk Honeycutt

$q = \{(\text{name}) \mid$
 $\exists \text{aid, mid, by, dy, ot, ti, ty, dc, r, sy, ey, did, by2, dy2, cid, pub, co, rd, sc.}$
 $\text{Actor}(\text{aid}, \text{'Al Pacino'}, \text{by}, \text{dy}) \wedge \text{HasActedIn}(\text{aid}, \text{mid})$
 $\wedge \text{Movie}(\text{mid}, \text{ot}, \text{ti}, \text{ty}, \text{dc}, \text{r}, \text{sy}, \text{ey}) \wedge \text{HasDirected}(\text{did}, \text{mid})$
 $\wedge \text{Director}(\text{did}, \text{'Christopher Nolan'}, \text{by2}, \text{dy2})$
 $\wedge \text{CriticReview}(\text{cid}, \text{pub}, \text{name}, \text{co}, \text{rd}, \text{sc}, \text{mid})$



Query 4

Name of directors and movie titles of oscar-winning movies in the DIRECTING category, produced by Warner Bros, with at least one user rating less than 0.5

```
select distinct d."name", m.title
from movie m, oscaraward o, studiohasproduced s, userrating u, hasdirected h, director d
where m.movie_id = o.movie_id
    and m.movie_id = u.movie_id
    and s.movie_id = m.movie_id
    and h.movie_id = m.movie_id
    and h.director_id = d.director_id
    and u.rating <= 0.5
    and s.studio_name = 'Warner Bros. Pictures'
    and o.category = 'DIRECTING'
```

[illegible]

$q = \{(\text{name}, \text{title}) \mid$
 $\exists \text{ did, by, dy, mid, ot, ty, dc, r, sy, ey, yc, aw, rid, rating.}$
 $\text{Director}(\text{did}, \text{name}, \text{by}, \text{dy}) \wedge \text{HasDirected}(\text{did}, \text{mid})$
 $\wedge \text{Movie}(\text{mid}, \text{ot}, \text{title}, \text{ty}, \text{dc}, \text{r}, \text{sy}, \text{ey})$
 $\wedge \text{StudioHasProduced}(\text{'Warner Bros. Pictures'}, \text{mid})$
 $\wedge \text{OscarAward}(\text{yc}, \text{'DIRECTING'}, \text{aw}, \text{mid})$
 $\wedge \text{UserRating}(\text{rid}, \text{rating}, \text{mid}) \wedge \text{rating} \leq 0,5$



Query 5

Title of horror movies that last less than 100 minutes, and name of actors participating, born after the year 2000, with a critic review on rotten tomato

```
select distinct m.title, a."name"
from movie m, genre g, isofgenre i, criticreview c, hasactedin h, actor a
where m.movie_id = i.movie_id
and g."name" = i.genre_name
and h.movie_id = m.movie_id
and h.actor_id = a.actor_id
and c.movie_id = m.movie_id
and a.birth_year >= 2000
and g."name" = 'Horror'
and m.runtime < 100
```

$q = \{(title, name) \mid$
 $\exists aid, by, dy, mid, ot, ty, dc, rt, sy, ey, cid, pub, cname, co, rd, sc.$
 $Actor(aid, name, by, dy) \wedge HasActedIn(aid, mid)$
 $\wedge Movie(mid, ot, title, ty, dc, rt, sy, ey)$
 $\wedge IsOfGenre('Horror', mid)$
 $\wedge CriticReview(cid, pub, cname, co, rd, sc, mid)$
 $\wedge by \geq 2000 \wedge rt < 100$

	A-Z title	A-Z name
1	14 Cameras	Gavin White
2	211	Michael Rainey Jr.
3	Annabelle	Gabriel Bateman
4	Artik	Gavin White
5	Brightburn	Jackson A. Dunn
6	Children of the Corn	Preston Bailey
7	Eli	Austin Foxx
8	Eli	Charlie Shotwell
9	Haunter	Peter DaCunha
10	Hellions	Peter DaCunha
11	Incarnate	David Mazouz
12	Little Evil	Owen Atlas



...Thanks!

References

- Theoretical notions: LSDM course taught by prof. Antonella Poggi
- Github: [Github repository](#)
- Datasets: see README file in repository
- [Guide link](#) to install Pentaho Kettle on Apple Silicon

