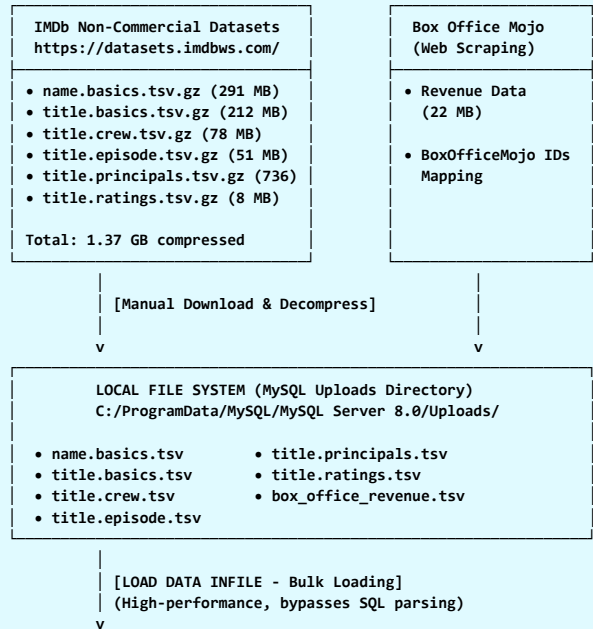


Extract

EXTERNAL DATA SOURCES
=====

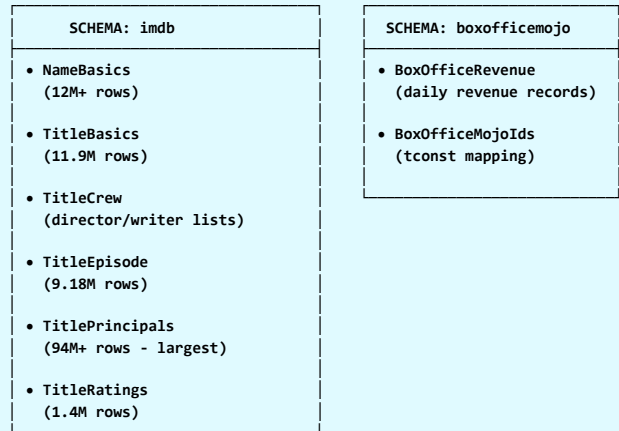


=====

LAYER 2: STAGING AREA

=====

{ STAGING SCHEMAS }



Transform

[1] STRING PARSING FOR MULTI-VALUED ATTRIBUTES

Fixed-Width Attributes (SUBSTRING_INDEX):
genres = "Action,Drama,Thriller"
--> genre1 = "Action"
--> genre2 = "Drama"
--> genre3 = "Thriller"

knownForTitles = "tt001,tt002,tt003,tt004"
--> knownForTitle1 = "tt001"
--> knownForTitle2 = "tt002"
--> knownForTitle3 = "tt003"
--> knownForTitle4 = "tt004"

Variable-Width Attributes (RECURSIVE CTEs):
directors = "nm0001,nm0002,nm0003,...,nm0150"

WITH RECURSIVE DirectorSplit AS (
-- Base case: Extract first director
-- Recursive case: Extract next director
-- Repeat until exhausted
)

--> Row 1: titleID=1, directorID=nm0001
--> Row 2: titleID=1, directorID=nm0002
--> Row 3: titleID=1, directorID=nm0003
... (up to 150+ rows for single title)

⚠ Bottleneck: 30% of transformation time
⚠ Recursion depth: Can exceed 1,000 iterations
⚠ Config: cte_max_recursion_depth = 5000

[2] DATA TYPE CONVERSIONS

Boolean Flags:
isAdult = '1' --> TRUE (BOOLEAN)
isAdult = '0' --> FALSE (BOOLEAN)

NULL Markers:
'\N' --> NULL (handled by LOAD DATA INFILE)

Numerical Precision:
averageRating = "7.5" --> 7.5 (DECIMAL(3,1))
numVotes = "12345" --> 12345 (INT)
revenue = "12345678" --> 12345678.00 (DECIMAL(12,2))

[3] SURROGATE KEY GENERATION

Natural Keys (IMDb IDs):
tconst = "tt0123456" --> titleID = 1 (AUTO_INCREMENT)
nconst = "nm0987654" --> personID = 1 (AUTO_INCREMENT)

Benefits:
• 4-byte integer vs 20-character string (join performance)
• Immutable (natural keys may change)
• Sequential allocation (cache-friendly)

Preservation:
• Natural keys stored as UNIQUE columns
• Enables source data traceability
• Supports incremental ETL

[4] DATE DIMENSION POPULATION (ON-DEMAND STRATEGY)

INSERT IGNORE INTO DimDate (...)
SELECT DISTINCT revenue_date FROM BoxOfficeRevenue
UNION
SELECT CURDATE()
UNION
SELECT CURDATE() + INTERVAL 1 DAY;

Approach: Only materialize dates referenced in fact tables
Trade-off: Smaller storage vs. ETL complexity

[5] DATA VALIDATION & FILTERING

Defensive WHERE Clauses:

- Filter NULL references before insertion
- Validate foreign key relationships
- Remove orphaned records (e.g., episodes without parents)

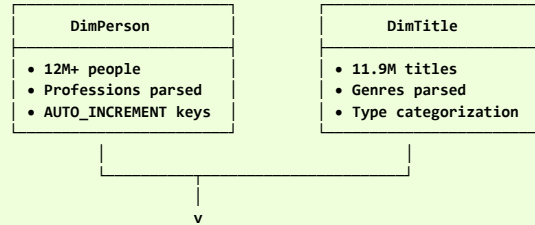
Temporary Constraint Disabling:

SET FOREIGN_KEY_CHECKS = 0;
SET UNIQUE_CHECKS = 0;
-- Bulk insert operations --
SET FOREIGN_KEY_CHECKS = 1;
SET UNIQUE_CHECKS = 1;

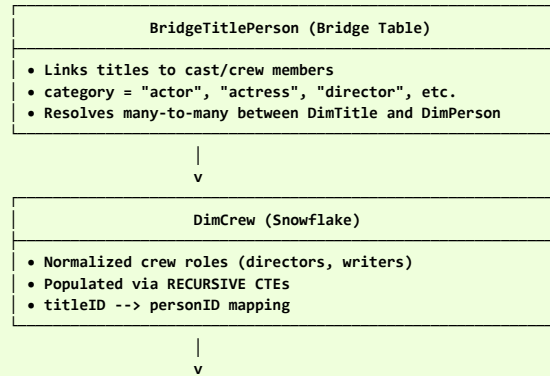
Load

LOADING ORDER (Dependency-Aware):

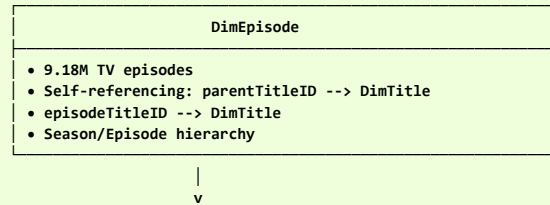
STEP 1: Independent Dimensions



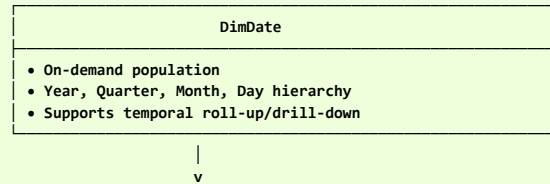
STEP 2: Bridge Tables (Many-to-Many Relationships)



STEP 3: Hierarchical Dimensions



STEP 4: Time Dimension



STEP 5: Fact Tables (Transaction Grain)

