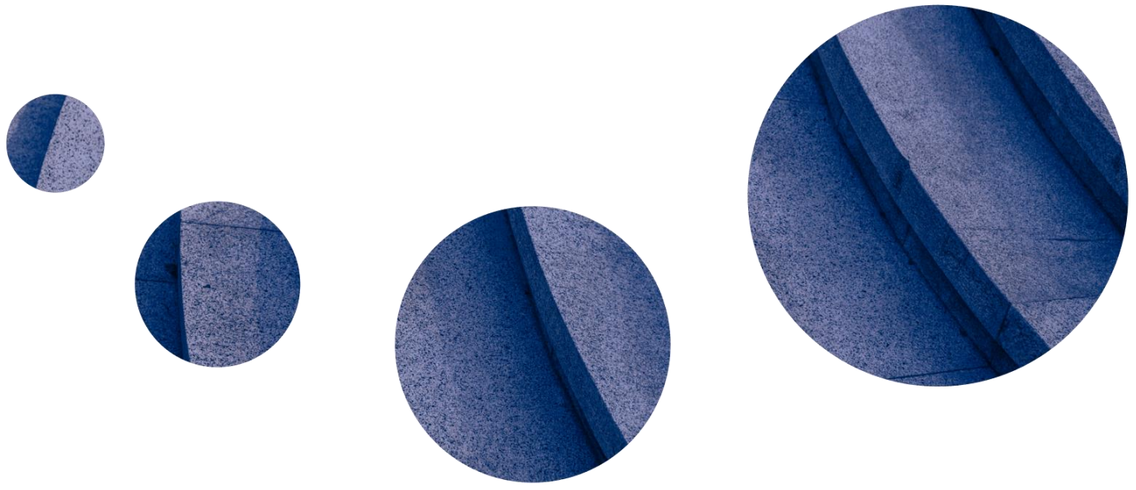


Our project report

For this project, we were tasked with reverse engineering five slides of the popular webpage IMDb. We conducted a careful analysis of the slides and visited the website itself in order to create an Entity Relationship Diagram (ERD), a Relational Schema (RS) and SQL code. Our goal was to include every piece of information - entity, attribute or otherwise - that is shown on the slides, with emphasis on optimizing our project for good performance and avoiding redundancy.

After careful analysis of the different parts of the database, we discerned the following predominant entities;

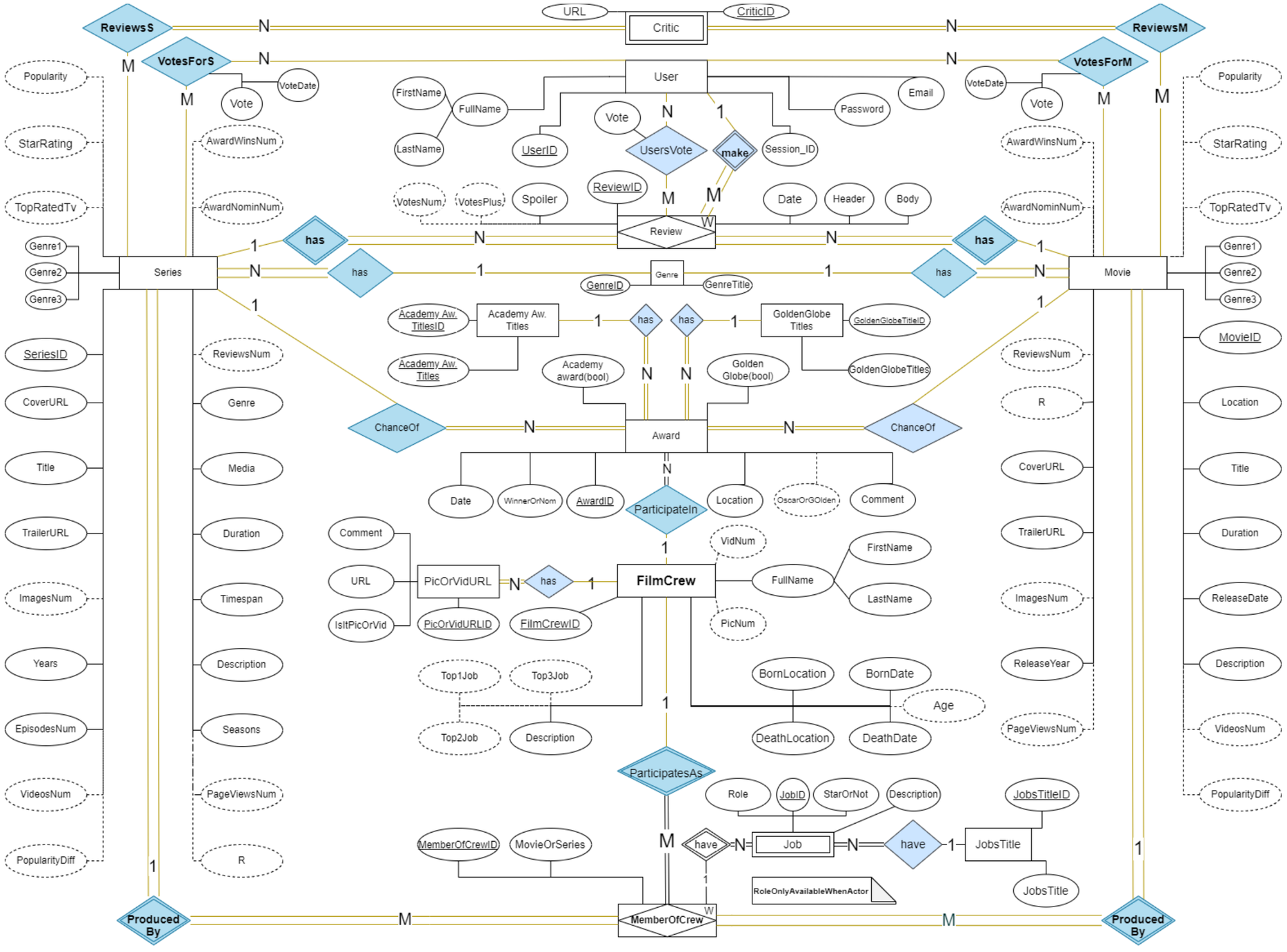
-FilmCrew

-User

-Award

-Movie and Series (Film Entity separated by type)

Bearing in mind the aforementioned entities, we devised the relationships between them, we created the weak entities that are formed by some of these relationships and defined the ownership of each attribute that we found in the analysis to its corresponding entity. Hence, we ended up with the following Entity Relationship Diagram;


ReviewsS
VotesForS
ReviewsM
VotesForM
Critic
URL
CriticID
User
FirstName
FullName
LastName
UserID
Vote
UsersVote
make
Session_ID
Password
Email
VoteDate
Popularity
StarRating
TopRatedTv
AwardWinsNum
AwardNominNum
VotesNum
VotesPlus
Spoiler
ReviewID
Review
Date
Header
Body
has
Series
Movie
Genre1
Genre2
Genre3
Genre
GenreID
GenreTitle
Academy Aw. TitlesID
Academy Aw. Titles
GoldenGlobe Titles
GoldenGlobeTitleID
GoldenGlobeTitles
Academy award(bool)
Golden Globe(bool)
ChanceOf
Award
SeriesID
CoverURL
Title
TrailerURL
ImagesNum
Years
EpisodesNum
VideosNum
PopularityDiff
ReviewsNum
Genre
Media
Duration
Timespan
Description
Seasons
PageViewsNum
R
MovieID
Location
ReleaseDate
VideosNum
Date
WinnerOrNom
AwardID
ParticipateIn
OscarOrGOlden
Comment
VidNum
PicOrVidURL
IsItPicOrVid
PicOrVidURLID
FilmCrewID
FilmCrew
PicNum
Top1Job
Top3Job
Top2Job
BornLocation
BornDate
Age
DeathLocation
DeathDate
ParticipatesAs
Role
JobID
StarOrNot
JobsTitleID
JobsTitle
have
Job
RoleOnlyAvailableWhenActor
MemberOfCrewID
MovieOrSeries
MemberOfCrew
Produced By

Some clarifications regarding our ERD:

- On certain occasions in our diagram, it may seem like certain weak entities form a relationship with another weak entity. In reality, in these cases, one of these weak entities is in fact a strong entity in that particular relationship. For example, the 'MemberOfCrew' entity is a weak entity in its relationship ('ParticipateAs') to 'FilmCrew' and ‘Movies’/’Series’. However, 'MemberOfCrew' is a strong entity in its relationship ('have') to the weak entity 'Jobs', since the combination of FilmCrew/Series/Movie acts as the identifying factor that defines an instance of 'MemberOfCrew'.

- To avoid any confusion; the 'FilmCrew' entity represents a person related to the production of a film (e.g., an actor), and the 'MemberOfCrew' entity holds an instance for each FilmCrew (person) that matches them with the crew of a movie or series. In this way, the combination of the 'MemberOfCrew' instances forms a crew.

- To understand the 1-M (one-to-many) relationship between 'MemberOfCrew' and 'Movie'/'Series', we follow the reasoning that one specific crew of a certain movie or series never consists of the exact same members. Hence, a specific crew schema can produce only one 'Movie'/'Series'.

- We deem that the 'Critic' entity's attributes are filled with data from an external source (URL) and are not part of the website. Thus, the critic votes and reviews are not included in our database and not associated with the 'Reviews' and 'Votes' entities in our diagram.

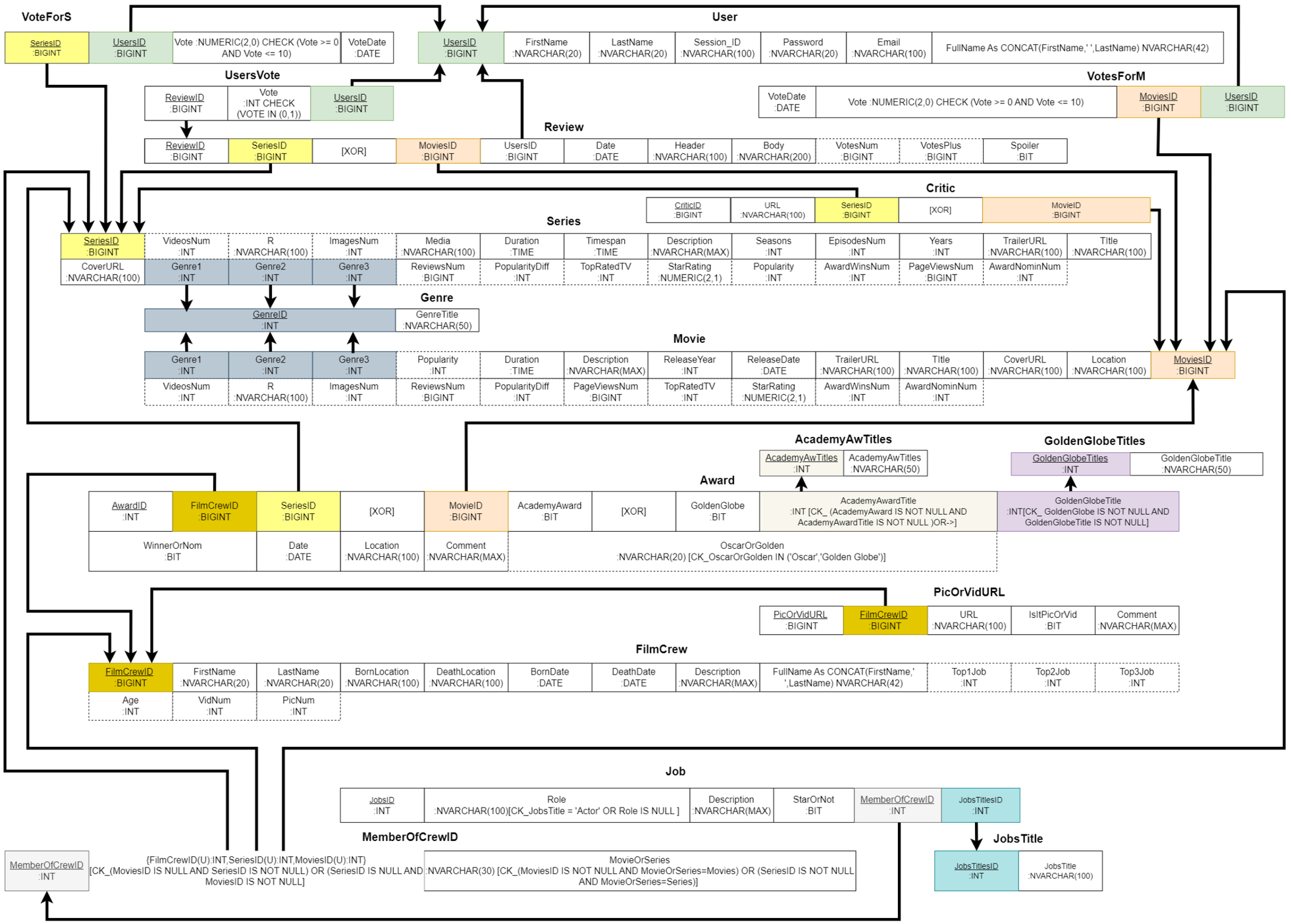
- The relationship between 'Series'/'Movie' and 'Genre' has been made with the reasoning that each 'Series'/'Movie' can have up to 3 separate genre attributes, each of which corresponds to one genre of the 'Genre' entity.

- The jobs in the 'FilmCrew' entity are calculated attributes (not input), derived from the 'Jobs' entity.

- To avoid redundancy by having an entity for each type of award, we have integrated the types of awards as attributes of the 'Award' entity. We enable each field only if chosen (boolean values distinguish between Academy Awards and Golden Globe Awards).

- In the weak entity 'Job' the 'Role' and 'StarOrNot' attributes are only enabled if the 'JobsTitle' has the 'Actor' value.

Moving on, we used the same data to define the relationships between entities more clearly and created the following Relational Schema. We acknowledge that each entity's attributes are typically represented in a single line, but due to our project's size, we used double rows to depict the attributes of some of the entities.


VoteForS
SeriesID :BIGINT
UsersID :BIGINT
Vote :NUMERIC(2,0) CHECK (Vote >= 0 AND Vote <= 10)
VoteDate :DATE
User
UsersID :BIGINT
FirstName :NVARCHAR(20)
LastName :NVARCHAR(20)
Session_ID :NVARCHAR(100)
Password :NVARCHAR(20)
Email :NVARCHAR(100)
FullName As CONCAT(FirstName,' ',LastName) NVARCHAR(42)
UsersVote
ReviewID :BIGINT
Vote :INT CHECK (VOTE IN (0,1))
UsersID :BIGINT
VotesForM
VoteDate :DATE
Vote :NUMERIC(2,0) CHECK (Vote >= 0 AND Vote <= 10)
MoviesID :BIGINT
UsersID :BIGINT
Review
ReviewID :BIGINT
SeriesID :BIGINT
[XOR]
MoviesID :BIGINT
UsersID :BIGINT
Date :DATE
Header :NVARCHAR(100)
Body :NVARCHAR(200)
VotesNum :BIGINT
VotesPlus :BIGINT
Spoiler :BIT
Critic
CriticID :BIGINT
URL :NVARCHAR(100)
SeriesID :BIGINT
[XOR]
MovieID :BIGINT
Series
SeriesID :BIGINT
VideosNum :INT
R :NVARCHAR(100)
ImagesNum :INT
Media :NVARCHAR(100)
Duration :TIME
Timespan :TIME
Description :NVARCHAR(MAX)
Seasons :INT
EpisodesNum :INT
Years :INT
TrailerURL :NVARCHAR(100)
TItle :NVARCHAR(100)
CoverURL :NVARCHAR(100)
Genre1 :INT
Genre2 :INT
Genre3 :INT
ReviewsNum :BIGINT
PopularityDiff :INT
TopRatedTV :INT
StarRating :NUMERIC(2,1)
Popularity :INT
AwardWinsNum :INT
PageViewsNum :BIGINT
AwardNominNum :INT
Genre
GenreID :INT
GenreTitle :NVARCHAR(50)
Movie
Genre1 :INT
Genre2 :INT
Genre3 :INT
Popularity :INT
Duration :TIME
Description :NVARCHAR(MAX)
ReleaseYear :INT
ReleaseDate :DATE
TrailerURL :NVARCHAR(100)
TItle :NVARCHAR(100)
CoverURL :NVARCHAR(100)
Location :NVARCHAR(100)
MoviesID :BIGINT
VideosNum :INT
R :NVARCHAR(100)
ImagesNum :INT
ReviewsNum :BIGINT
PopularityDiff :INT
PageViewsNum :BIGINT
TopRatedTV :INT
StarRating :NUMERIC(2,1)
AwardWinsNum :INT
AwardNominNum :INT
AcademyAwTitles
AcademyAwTitles :INT
AcademyAwTitles :NVARCHAR(50)
GoldenGlobeTitles
GoldenGlobeTitles :INT
GoldenGlobeTitle :NVARCHAR(50)
Award
AwardID :INT
FilmCrewID :BIGINT
SeriesID :BIGINT
[XOR]
MovieID :BIGINT
AcademyAward :BIT
[XOR]
GoldenGlobe :BIT
AcademyAwardTitle :INT [CK_ (AcademyAward IS NOT NULL AND AcademyAwardTitle IS NOT NULL)OR->]
GoldenGlobeTitle :INT[CK_ GoldenGlobe IS NOT NULL AND GoldenGlobeTitle IS NOT NULL]
WinnerOrNom :BIT
Date :DATE
Location :NVARCHAR(100)
Comment :NVARCHAR(MAX)
OscarOrGolden :NVARCHAR(20) [CK_OscarOrGolden IN ('Oscar','Golden Globe')]
PicOrVidURL
PicOrVidURL :BIGINT
FilmCrewID :BIGINT
URL :NVARCHAR(100)
IsItPicOrVid :BIT
Comment :NVARCHAR(MAX)
FilmCrew
FilmCrewID :BIGINT
FirstName :NVARCHAR(20)
LastName :NVARCHAR(20)
BornLocation :NVARCHAR(100)
DeathLocation :NVARCHAR(100)
BornDate :DATE
DeathDate :DATE
Description :NVARCHAR(MAX)
FullName As CONCAT(FirstName,' ',LastName) NVARCHAR(42)
Top1Job :INT
Top2Job :INT
Top3Job :INT
Age :INT
VidNum :INT
PicNum :INT
Job
JobsID :INT
Role :NVARCHAR(100)[CK_JobsTitle = 'Actor' OR Role IS NULL]
Description :NVARCHAR(MAX)
StarOrNot :BIT
MemberOfCrewID :INT
JobsTitlesID :INT
MemberOfCrewID
MemberOfCrewID :INT
{FilmCrewID(U):INT,SeriesID(U):INT,MoviesID(U):INT} [CK_(MoviesID IS NULL AND SeriesID IS NOT NULL) OR (SeriesID IS NULL AND MoviesID IS NOT NULL]
MovieOrSeries :NVARCHAR(30) [CK_(MoviesID IS NOT NULL AND MovieOrSeries=Movies) OR (SeriesID IS NOT NULL AND MovieOrSeries=Series)]
JobsTitle
JobsTitlesID :INT
JobsTitle :NVARCHAR(100)

Finally, we made a SQL script (attached to this report) to bring everything together. A few characteristics of our SQL script:

1. Constraints exist almost in every attribute to ensure input of the correct data values (email structure, URLs, dates diffs etc.).
2. Optimization by indexing (non-clustered) columns of specific tables such as UsersVote (ReviewID) to increase performance.
3. Carefully attached triggers (no cursor/lock) to keep the tables up to date (Ex VideosNum etc.) without affecting the performance.
4. Scheduled Daily Procedures/Jobs (Update Mov/Ser votes and ReviewNum) at 5-6 AM (low traffic hours).
5. All counts and any other statements are carefully made, in order to avoid needless repetitiveness and therefore boost performance.

To conclude, in the process of reverse engineering IMDb, we came across numerous stumbling blocks which forced us to think outside the box. We therefore approached this project not only from a functionality standpoint; we wished to provide solutions to our problems that would not simply serve a purpose, but also enhance optimization.