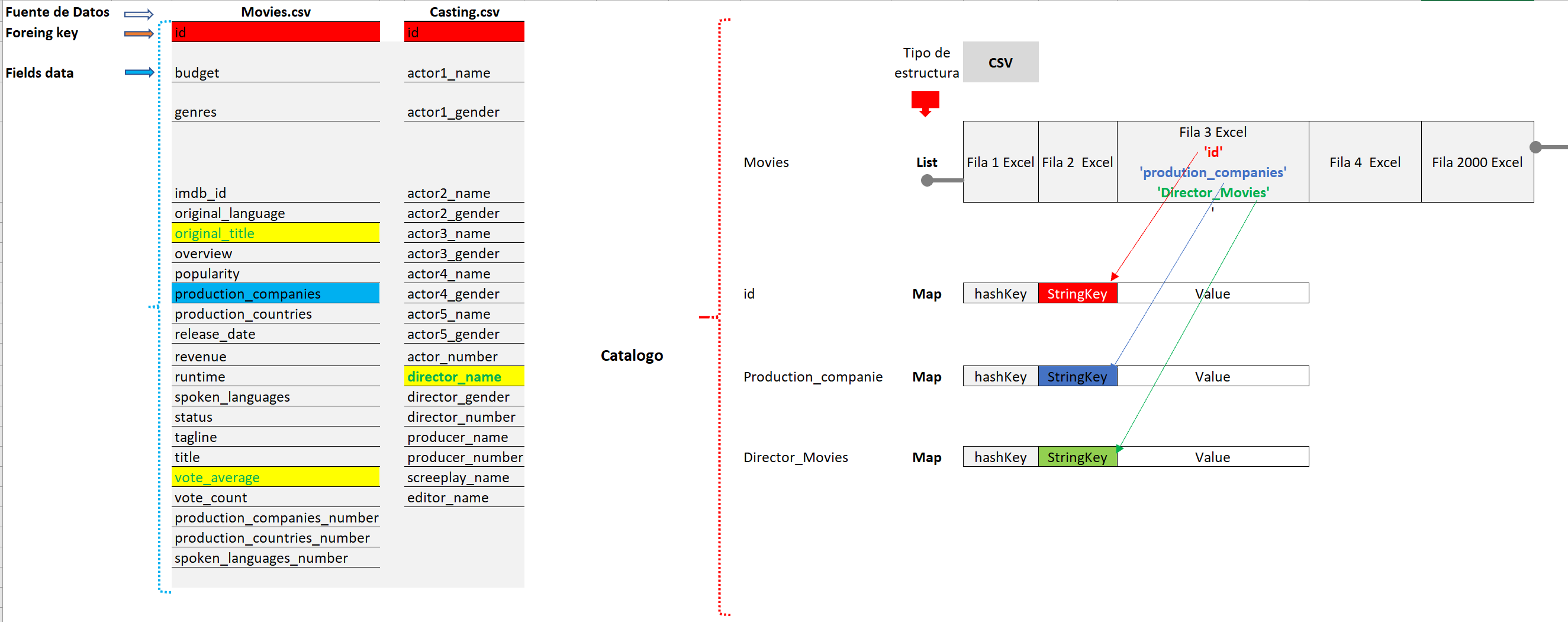
**PREGUNTAS DE ANALISIS DEL RETO 2**

# **¿Qué estructura debería tener el catálogo?**

De manera ilustrativa, se configura gráficamente como se ilustra a continuación, se parte del análisis de los datos de entrada los .CSVs, entendiendo las llaves foráneas de enlace, los campos de entrada y necesarios para todos y cada uno de los requerimientos del Reto 2, ya en la parte derecha, se muestra esquemáticamente como se configuraría el catálogo, donde la Movies funciona como un modelo de relación de uno a muchos tanto con los Directores, como actores, genero, en este sentido, la estructura de Movies es una lista array list, igualmente de ilustran los diferentes hash-maps fundamentales para la configuración del catálogo, donde cada Movie hace referencia a estos mapas para relacionarlo con un atributo especifico de la Movie, como lo es las productoras, los actores, los directores, países, etc., estos mapas, si bien se definen como la pareja key, value, se ilustra un llave previa que la que configura el hash table para convertir la key del map, que tiene una características de string en la mayoría de los casos y convertirla en valor numérico para poder realizar las diferentes operaciones sobre el map.



En línea con la anterior ilustración, se relaciona a continuación como quedo ya en le Model el init\_catalog

**def newCatalog():**

    catalog = {'movies': None,

                'id': None,

                'production\_companies': None,

                'original\_title': None,

                'vote\_average': None,

                'vote\_count': None,

                'genres': None,

                'production\_countries':None,

                'actor1\_name': None,

                'director\_name':None}

    catalog['id'] = mp.newMap(2000,

                                    maptype='PROBING',

                                    #PROBING, CHAINING

                                    loadfactor=0.4,

                                    comparefunction=compareproductionCompanies)

    catalog['production\_companies'] = mp.newMap(2000,

                                    maptype='PROBING',

                                    #PROBING, CHAINING

                                    loadfactor=0.4,

                                    comparefunction=compareproductionCompanies)

    catalog['genres'] = mp.newMap(2000,

                                    maptype='PROBING',

                                    #PROBING, CHAINING

                                    loadfactor=0.4,

                                    comparefunction=compareproductionCompanies)

    catalog['production\_countries'] = mp.newMap(2000,

                                    maptype='PROBING',

                                    #PROBING, CHAINING

                                    loadfactor=0.4,

                                    comparefunction=compareproductionCompanies)

    catalog['directors'] = mp.newMap(2000,

                                    maptype='PROBING',

                                    #PROBING, CHAINING

                                    loadfactor=0.4,

                                    comparefunction=compareActors)

    catalog['movieIds'] = mp.newMap(2000,

                                    maptype='PROBING',

                                    #PROBING, CHAINING

                                    loadfactor=0.4,

                                    comparefunction=compareActors)

    catalog['actors1'] = mp.newMap(2000,

                                    maptype='PROBING',

                                    #PROBING, CHAINING

                                    loadfactor=0.4,

                                    comparefunction=compareActors)

    catalog['movies'] = lt.newList('ARRAY\_LIST', compareMovieIds)

    catalog['casting'] = lt.newList('ARRAY\_LIST', compareDirectorIds)

**return catalog**

# **¿Qué hacer al procesar cada línea de cada uno de los archivos?**

Lo interesante de esta pregunta es que logramos gestionar este Reto bajo un modelo **VMC, view, model y controller**, esto nos permite segmentar las líneas de código, de tal forma que **View** tiene como función desarrollar la interfaz de entrada y salida de datos hacia el usuario final, el **Model** se concentra la definición y creación de los modelos de datos, como se enuncio en punto 1, con respecto al **Controller**, este tiene como principal fin desarrollar las funciones o procedimientos para la tratamiento de los datos e implementación de las operaciones.

# **¿Cómo se debería ordenar la información? ¿Por qué concepto (fecha, actor, director) ?**

En línea con los diferentes requerimientos, el ordenar la información depende de la necesidad de acceso a los datos de cada requerimiento, por ejemplo, para el requerimiento 1, la información se acceso con el campo **production\_companies**, realizando agrupación por productora y asociándole los campos **original\_title** y **vote\_average**, no se nos requería hacer un ordenamiento, pero de requerirse nuestra sugerencia es ordenar por el campo “**production\_companies**”. En general para todo el Reto 2, considero prudente que para cada requerimiento se ordene por el **map.key,** de cada componente del catálogo.

# **¿Cómo usar la menor cantidad de memoria posible?**

Esta pregunta nos genero mucha expectativa, toda vez que reducir la memoria implicaba entender con mucha claridad: el modelo de datos, las estructuras datos, los datos por requerimientos y los algoritmos usados para la creación de todos y cada uno de los requerimientos, en este sentido en desconocer alguno de ellos repercute tanto en el nivel de complejidad como el desempeño de las soluciones tanto en memoria como en ciclos de proceso (mayor o menor tiempo).

# **¿Cómo e pueden utilizar las estructuras de datos vistas en clase?**

En este Reto usamos muchas las estructuras de datos vistas en clase, como se indicó en la pregunta No.1, se usaron tanto listas como hash maps, en este sentido todas las estructuras con sus respectivos métodos disponibles fueron utilizados para el desarrollo con éxito de este Reto.

# **¿Cuáles son los campos ideales para definir una llave de un Map?**

De primera mano, pensamos que definitivamente son los campos numéricos, no obstante las tablas hash son fundamentales para ayudar a manejar llaves de otros tipos como string, colores, tamaños, categorías, países, entre otros, es así como las hash tables, nos ayudan a tomar cualquier llave este tipo y convertirla en numérico. En la siguiente ilustración como de manera didáctica un mapa tiene un hashkey numérico, stringkey u otro tipo, y su respectivo valor.

