## 1 Schema Diagram

Primary keys are shown in bold.

- actors(**id**, first_name, last_name, gender)
- movies(**id**, name, year, rank)
- roles(**actor_id**, **movie_id**, role) *If you assume that an actor plays only one role in a movie, this is ok, if you allow an actor to play multiple roles such as Robin Williams in Mrs. Doubtfire then it should be roles(**actor_id**, **movie_id**, **role**).
- movies_genres(**movie_id**, **genre**)
- directors(**id**, first_name, last_name)
- movies_directors(**director_id**s, **movie_id**)

We draw a line from an attribute to the primary key to great the schema diagram (which are the foreign key constraints).

Foreign Keys:

- **actor_id** in roles is a foreign key for actors.
- **movie_id** in roles is foreign key for movies.
- **movie_id** in movies_genres is a foreign key for movies.
- **movie_id** in movies_directors is a foreign key for movies.
- **director_id** in movies_directors is a foreign key for directors.

## 2 Relational Algebra

1. $\Pi_{name}(\sigma_{genres='comedy'}(movies \bowtie_{(movies.id=movies\_genres.movie\_id)} movies\_genres))$
2. $\Pi_{name,first\_name,last\_name}(((\sigma_{rank\geq 8.5}(movies)) \bowtie_{movies.id=roles.movie\_id} roles) \bowtie_{(actors.id=roles.actor\_id)} actors)$
3. $\Pi_{name}(\sigma_{rank<5\wedge date\geq 1970\wedge date<1980}(movies))$
4. $\Pi_{last\_name}((\sigma_{gender='F'}actors) \bowtie_{(actor.id=roles.actor\_id)} (\sigma_{genres='Sci-Fi'}roles \bowtie_{roles.movie\_id=movies.id} movies))$
5. Idea: first get all movies with rank > 9. Then join with movie directors table. Call this joined table "m". Repeat and call new table "n". To find directors with more than one movie, self join "m" and "n" with director id equal and movie id not equal. Return the projection on the last and first name of the directors.
   $\Pi_{last\_name,first\_name}(directors \bowtie_{directors.id=movies\_directors.directors.id}$
   $(\rho_m(movies\_directors \bowtie_{movies\_directors.movie\_id=movies.id} (\sigma_{rank>9}movies)))$
   $\bowtie_{m.movie\_id\neq n.movie\_id\wedge m.director\_id=n.director\_id}$
   $(\rho_n(movies\_directors \bowtie_{movies\_directors.movie\_id=movies.id} (\sigma_{rank>9}movies)))$
6. $\Pi_{directors.first\_name,directors.last\_name}((directors \bowtie_{(directors.first\_name=actors.first\_name)\wedge(directors.last\_name=actors.last\_name)} actors))$
7. $\Pi_{year}((\sigma_{(first\_name=`Robin')\wedge(last\_name='Williams')}actors) \bowtie_{(actors.id=roles.actor\_id)} roles \bowtie_{(roles.movie\_id=movies.id)} movies)$

8. $\Pi_{first\_name,last\_name,name,role}((movies \bowtie_{movies.id=roles.movie\_id} roles) \bowtie_{roles.actor\_id=actors.id} actors)$

9. $\Pi_{movies.name}(movies \bowtie_{movies.id=movies\_genres.movies\_id} (\sigma_{genres='Family' \wedge genres \neq 'Comedy'} moves\_genres))$

10. The **intersect** of two relations $A$ and $B$ is simply the set of tuples that are in both relations (the assumption if you are doing intersect is that the attributes are the same). This is simply a natural join on all the attributes.

    One can think of the difference $A - B$, the tuples in $A$ that are not in $B$. To find these tuples simply doe `select * from A where A not in B`.