To create a data model for the movie scenario described, we can identify the key entities, their attributes, and the relationships between them. Here's how you can structure it:

Entities and Attributes

1. **Movie**
   - **Attributes:**
     - Movie ID (Primary Key)
     - Title
     - Release Date
2. **Actor**
   - **Attributes:**
     - Actor ID (Primary Key)
     - Name
     - Date of Birth
3. **Director**
   - **Attributes:**
     - Director ID (Primary Key)
     - Name
     - Date of Birth

Relationships

1. **Movie to Actor**
   - **Type:** Many-to-Many
   - **Explanation:** A movie can have many actors, and an actor can act in many movies.
   - **Join Table:** Movie_Actor
     - **Attributes:**
       - Movie ID (Foreign Key)
       - Actor ID (Foreign Key)
2. **Movie to Director**
   - **Type:** Many-to-Many
   - **Explanation:** A movie can have multiple directors, and a director can direct many movies.
   - **Join Table:** Movie_Director
     - **Attributes:**
       - Movie ID (Foreign Key)
       - Director ID (Foreign Key)

Visual Representation

In Visual Paradigm, you would represent these entities as tables, with lines connecting them to depict relationships. Use crow's feet notation to indicate the many-to-many relationships, with the join tables clearly defined.

Summary

This data model effectively captures the complex relationships among movies, actors, and directors, allowing for flexible queries and data management in a movie database system. The join tables facilitate the many-to-many relationships, ensuring that all relevant associations are accurately represented.