CineTravel

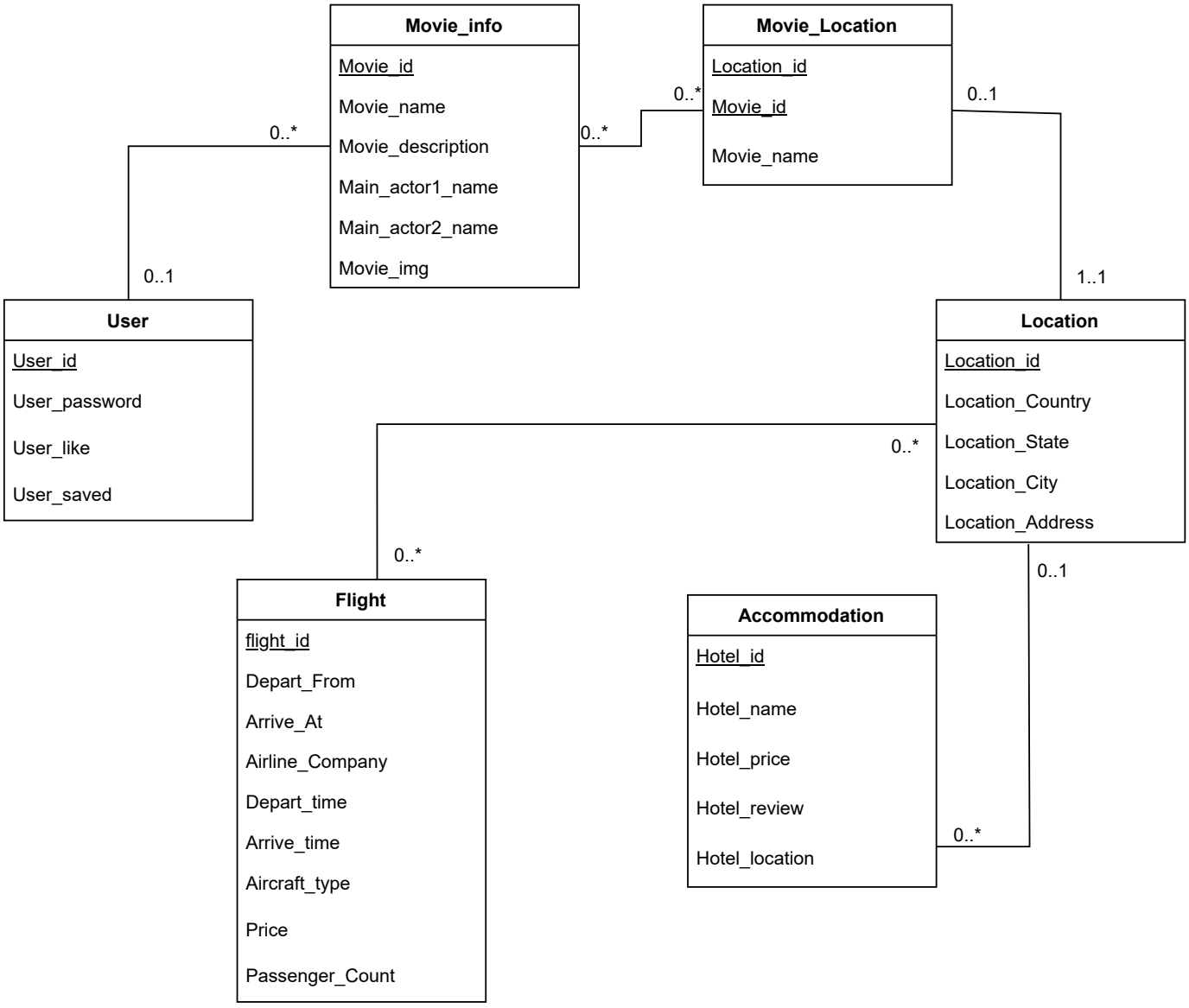
Movie_info
Movie_id
Movie_name
Movie_description
Main_actor1_name
Main_actor2_name
Movie_img
Movie_Location
Location_id
Movie_id
Movie_name
User
User_id
User_password
User_like
User_saved
Location
Location_id
Location_Country
Location_State
Location_City
Location_Address
Flight
flight_id
Depart_From
Arrive_At
Airline_Company
Depart_time
Arrive_time
Aircraft_type
Price
Passenger_Count
Accommodation
Hotel_id
Hotel_name
Hotel_price
Hotel_review
Hotel_location
0..*
0..*
0..*
0..1
0..1
1..1
0..*
0..*
0..1
0..*

Table1: User
Table2: Movie_info
Table3: Flight
Table4: Accommodation
Table5: Location
Table6: Movie_Location

Movie.Movieid -> Movie_Location.Movie_id
Flight.Depart_From -> Location.Location_City
Flight.Depart_From -> Location.Location_City

-

Movie (Movie_id, Movie_name, Movie_description, Main_actor1, Main_actor2, ...)

- *Primary Key*: Movie_id
- *Assumption*: Each movie has a unique identifier and basic information.
- *Normalization*: All non-key attributes are directly and fully functionally dependent on Movie_id. There is no dependency between the non-key attributes themselves (for example, Main_actor1_name does not determine Movie_img), which guarantees that no transitive dependencies exist. Hence, Movie_info satisfies 3NF.

-

Location (Location_id, Location_city, Location_state, Location_country, Location_Address)

-

- *Primary Key*: Location_id
- *Assumption*: A filming location is uniquely identified and might be reused across different movies.
- *Normalization*: City, state, and country are attributes of **Location**, not stored in **Movie** to avoid redundancy.

-

Movie_Location (Movie_id, Location_id, Movie_name)

- *Primary Key*: (Movie_id, Location_id)
- *Assumption*: A movie can have multiple filming locations, and a location can be used for multiple movies.
- *Relationship*: **Many-to-Many (Movie ↔ Location)**
- *Normalization*: This avoids repeating location data in the **Movie** table.

-

Flight (Flight_id, Depart_time, Arrival_time, Depart_From, Arrival_At, price, Airline_company, Aircraft_type, Passenger_count)

- *Primary Key*: Flight_id
- *Assumption*: Each flight has a unique identifier and connects two locations.
- *Relationship*: **1-to-Many (Location ↔ Flight)** (A flight departs from one location but can be linked to multiple destinations.)
- Normalization: Each attribute in the Flight table is fully functionally dependent on Flight_id. None of the non-key attributes determine other non-key attributes (e.g., Depart_time does not determine Price), eliminating any transitive dependencies. Thus, the Flight table is in 3NF.

-

Accommodation (Hotel_id, Hotel_location, Hotel_name, Hotel_price, Hotel_rating, Hotel_review)

- *Primary Key*: Hotel_id
- *Assumption*: Hotels are linked to a location.
- *Normalization*: In the Accommodation table, all non-key attributes are entirely dependent on Hotel_id, and no non-key attribute is transitively dependent on another. Hotel_location is a foreign key linking to the Location table but still remains fully functionally dependent on Hotel_id. Therefore, Accommodation meets the requirements for 3NF.

User (User_id, User_password, User_like, User_saved)

- Primary Key: User_id
- Assumtion: The User_like and User_saved fields are expected to store multiple or varied entries
- Normalization: Every attribute in the User table is fully dependent on the unique User_id. There are no transitive dependencies among the non-key attributes (User_password, User_like, User_saved), ensuring that each piece of user data is stored without redundancy. Therefore, the User table is in 3NF.

User(User_id: INT [PK],
User_password: VARCHAR(255),
User_like: TEXT,
User_saved: TEXT)

Movie(Movie_id: INT [PK],
Movie_name: VARCHAR(255),
Movie_description: TEXT,
Main_actor1_name: VARCHAR(255),
Main_actor2_name: VARCHAR(255),
Movie_img: VARCHAR(255))

Movie_Location(Location_id: INT [FK to Location.Location_id],
Movie_id: INT [FK to Movie_info.Movie_id],
Movie_name: VARCHAR(255),
PRIMARY KEY (Location_id, Movie_id))

Location(Location_id: INT [PK],
Location_Country: VARCHAR(255),
Location_State: VARCHAR(255),
Location_City: VARCHAR(255),
Location_Address: TEXT)

Flight(Flight_id: INT [PK],
Depart_From: VARCHAR(255),
Arrive_At: VARCHAR(255),
Airline_Company: VARCHAR(255),
Depart_time: DATETIME,
Arrive_time: DATETIME,
Aircraft_type: VARCHAR(255),
Price: DECIMAL(10,2),
Passenger_Count: INT)

Accommodation(Hotel_id: INT [PK],
Hotel_name: VARCHAR(255),
Hotel_price: DECIMAL(10,2),
Hotel_review: TEXT,
Hotel_location: INT [FK to Location.Location_id])