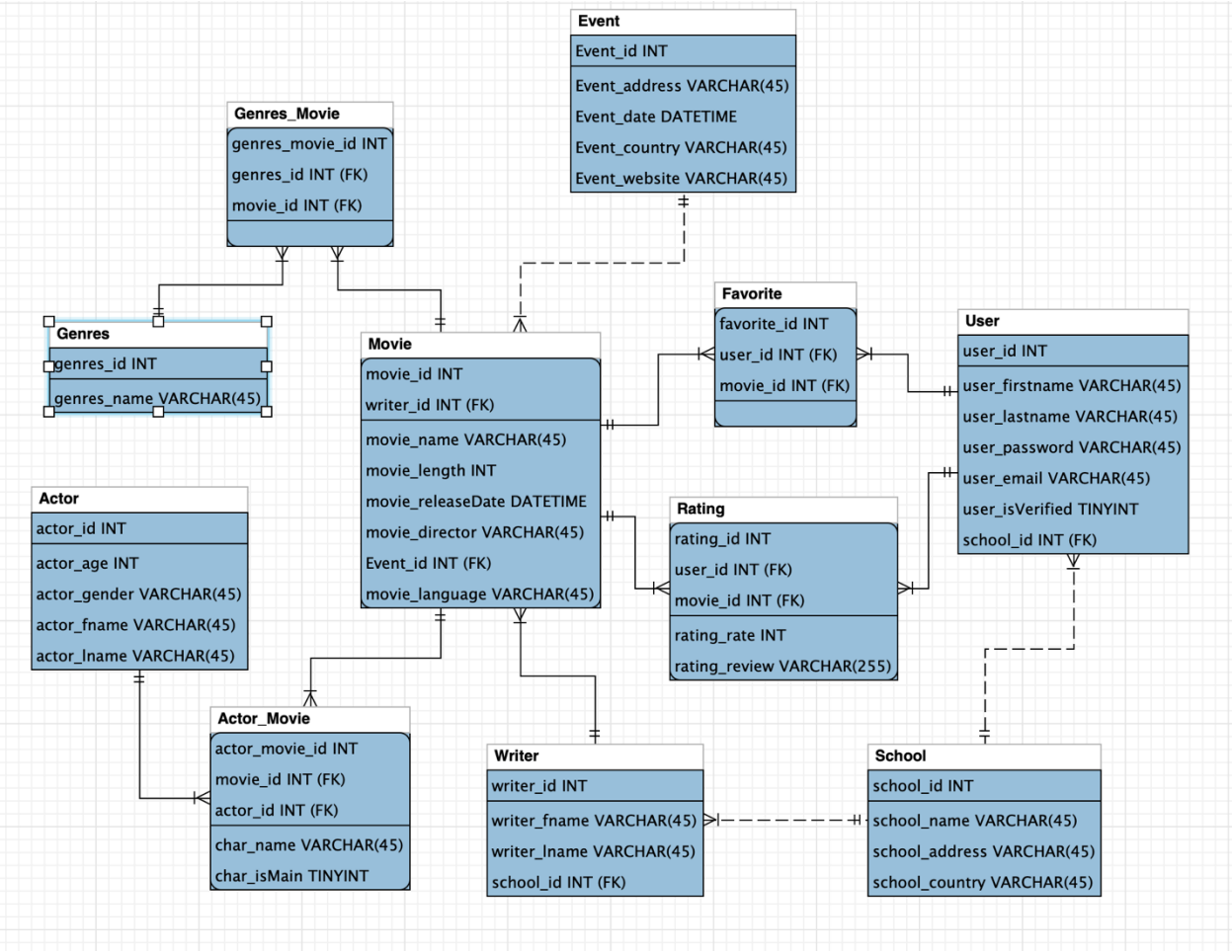
Milestone 2

<Physical Model ER Diagram >



<Query 1>

-- As a movie database it is essential to provide users with the highest rated movies from our database.
-- Query 5 overall highest rating movie
use mk_team03_movie_db;
SELECT * FROM Movie m INNER JOIN Rating r ON m.movie_id = r.movie_id ORDER BY r.rating_rate desc LIMIT 5;

movie...	movie_name	movie_length	movie_releaseDate	movie_director	Event_id	movie_language	writer_id	rating_id	user_id	movie_id	rating_rate
1	Trouble with the Curve	46	2020-06-20 18:53:56	Justina Itskovitz	2	Punjabi	32	18	28	1	5
2	Standing in the Shadows of Motown	119	2021-12-12 23:50:40	Annette Kitchingham	2	Lao	20	46	8	2	5
4	Love Me Tender	65	2022-01-06 06:42:16	Amalee Bricksey	1	Burmese	61	45	25	4	5
5	Education, An	99	2021-02-06 13:34:21	Maria Ullrich	7	English	39	24	32	5	5
5	Education, An	99	2021-02-06 13:34:21	Maria Ullrich	7	English	39	43	21	5	5

<Query 2>

-- This query is essential because it allows users to view specific movies as opposed to scrolling to the movie they would like to see and/or are interested in.
-- Query the review for a certain movie

```
use mk_team03_movie_db;
SELECT m.movie_name, r.rating_review FROM Movie m INNER JOIN Rating r ON m.movie_id =
r.movie_id WHERE m.movie_id = 3 AND r.rating_review IS NOT NULL;
```

movie_name	rating_review
That Cold Day in the Park	good
That Cold Day in the Park	good
That Cold Day in the Park	normal

<Query 3>

-- Querying a user's favorite movie will help additonal users discover new movies.
-- Query one user’s favorite movies

```
use mk_team03_movie_db;
SELECT user_firstname, movie_name FROM User u INNER JOIN Favorite f ON u.user_id =
f.user_id INNER JOIN Movie m ON m.movie_id = f.movie_id WHERE u.user_id = 40;
```

user_firstna...	movie_name
Marita	Trouble with the Curve
Marita	Dante's Inferno: An Animated Epic

<Query 4>

-- Often times users are in specific moods and only want to wview movies in a specific genre. This query allows users to sort by genre.
-- Query movies according genres

```
use mk_team03_movie_db;
SELECT g.genres_name, m.movie_name FROM Genres g INNER JOIN Genres_Movie gm ON g.genres_id = gm.genres_id INNER JOIN Movie m ON gm.movie_id = m.movie_id WHERE g.genres_id = 7;
```

genres_name	movie_name
Romance	Dante's Inferno: An Animated Epic
Romance	Lupin the 3rd
Romance	Business as Usual
Romance	Perfume: The Story of a Murderer
Romance	Business as Usual
Romance	Education, An

<Query 5>

-- Users my have actors/actresses that they favor and are fans of, this query is essential in that it will display movies that the actor has played in.
-- Query all movies a certain actor has performed in

```
use mk_team03_movie_db;
SELECT a.actor_fname,a.actor_lname, am.char_name, m.movie_name FROM Actor a INNER JOIN Actor_Movie am ON a.actor_id = am.actor_id INNER JOIN Movie m ON am.movie_id = m.movie_id WHERE a.actor_id = 5;
```

actor_fname	actor_lname	char_name	movie_name
Palm	Penson	Salasar	Standing in the Shadows of Motown
Palm	Penson	Ringer	Education, An

<Query 6>

-- Our database is specific to films made by university students. This query helps find the writer and school associated with each writer.
-- Query writer associated for specific schools.

```
use mk_team03_movie_db;
SELECT w.writer_fname, s.school_name FROM Writer w INNER JOIN School s ON w.school_id = s.school_id WHERE s.school_id = 5;
```

writer_fname	school_name
Federica	Shimer College
Berne	Shimer College

<Query 7>

-- It is important to know where to watch a movie when it premiers, this query allows users to do so by displaying the event information where a specific movie will be played.
-- Query movies featured in each event.

```
use mk_team03_movie_db;
SELECT m.movie_name, e.Event_address FROM Event e INNER JOIN Movie m ON m.Event_id = e.Event_id WHERE e.Event_id = 2;
```

movie_name	Event_address
Trouble with the Curve	7850 Northview Street
Standing in the Shadows of Motown	7850 Northview Street

<Query 8>

-- This query helps to reward users who use and contribute to the database often, displaying the user with the most contributions, motivating other users to contribute as well.
-- Query users that have rated the most movies

```
use mk_team03_movie_db;
SELECT u.user_firstname, a.RATING_COUNT FROM (
        SELECT user_id, count(rating_rate) RATING_COUNT
   FROM Rating
   GROUP BY user_id
   ORDER BY RATING_COUNT desc
   ) a
INNER JOIN User u ON a.user_id = u.user_id
LIMIT 1;
```

movie_name	Event_address
Trouble with the Curve	7850 Northview Street
Standing in the Shadows of Motown	7850 Northview Street

<Query 9>

-- -- Our database is specific to films made and reviewed by university students. This query helps find users assocaited with each school.
-- Query users associated with specific school.

```
use mk_team03_movie_db;
SELECT u.user_firstname, s.school_name FROM User u INNER JOIN School s ON u.school_id =
s.school_id Where s.school_id = 5;
```

user_firstna...	school_name
Ruby	Shimer College
Jennica	Shimer College
Sandor	Shimer College
Greggory	Shimer College

<Query 10>

-- This query allows users to know which movies are playing and on what day.
-- Query movie shown on specific date (e.g. weekend) so that client can come to watch

use mk_team03_movie_db;
SELECT m.movie_name, e.event_address FROM Movie m INNER JOIN Event e ON m.event_id = e.event_id WHERE e.event_date = "2022-07-20";

movie_name	event_address
That Cold Day in the Park	48 Glacier Hill Park
Lupin the 3rd	48 Glacier Hill Park
Business as Usual	4 Warner Alley
Dante's Inferno: An Animated Epic	4 Warner Alley

<Stored Procedure 1>

This stored procedure is to add a movie. It asks for all of the column information and checks to make sure that the movie being added has a valid event id and writer id associated with the appropriate event/writer in their respective table. After it checks that, if those are valid, it adds the movie to the movie table. If the event id and writer id are not valid, it does not add the movie to the movie table.

CREATE DEFINER=`mk_team03`@`%` PROCEDURE `add_movie`(IN movieName VARCHAR(45),
IN movieLength INT,
IN movieReleaseDate DATETIME,

```
IN movieDirector VARCHAR(45),
IN eventID INT,
IN movieLanguage VARCHAR(45),
IN writerID INT
)
BEGIN
	IF EXISTS(SELECT * FROM Writer WHERE writer_id = writerID) THEN
		IF EXISTS(SELECT * FROM Event WHERE Event_id = eventID)
		THEN
			INSERT INTO Movie (movie_name, movie_length, movie_releaseDate,
movie_director, Event_id, movie_language, writer_id)
			Values(movieName, movieLength, movieReleaseDate, movieDirector,
eventID, movieLanguage, writerID);
	SELECT * FROM Movie WHERE movie_name = movieName;
		END IF;
  END IF;
END
```

movie_id	movie_name	movie_length	movie_releaseDate	movie_direct...	Event_id	movie_language	writer_id
16	Another movie	200	2022-07-11 00:00:00	Mathew	7	English	2

<Stored Procedure 2>

This stored procedure is to find a movie. It asks for the movie id and pulls up the movie associated with the given id.

```
CREATE DEFINER=`mk_team03`@`%` PROCEDURE `find_movie`(IN movieID smallint)
BEGIN
	select *
  from Movie
  where movie_id = movieID;
END
```

movie_id	movie_name	movie_length	movie_releaseDate	movie_director	Event_id	movie_language	writer_id
3	That Cold Day in the Park	83	2020-12-23 13:46:34	Thacher Clavey	5	Dari	68