CSCI 327 - Introduction to Databases Spring 2025

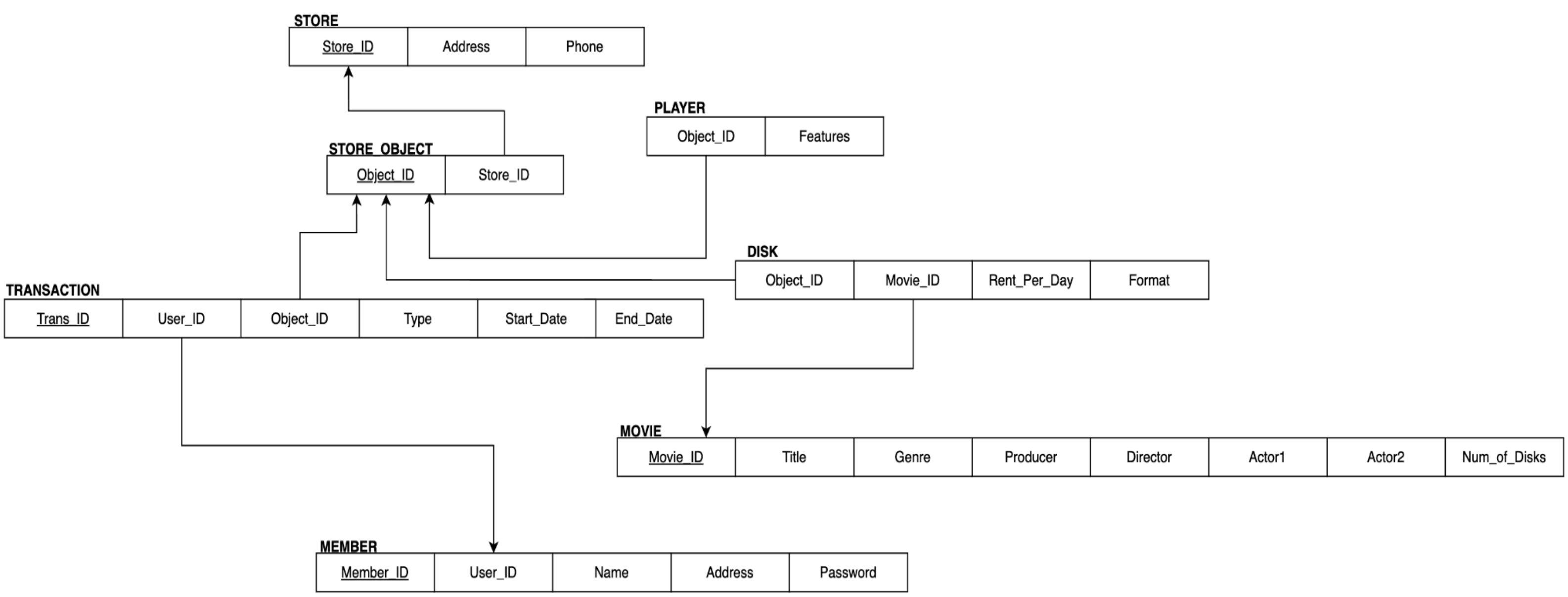
Project Deliverable 2

Team:
- **Adam Kaderbhai and**
- **Lata Bhata**

1. Goals for Phase 2

- Design a logical database schema for the VideoStore system by mapping the EER diagram from Deliverable 1 into a relational schema.
- Translate entities, relationships, and generalization/specialization hierarchies into tables.
- Ensure proper identification of primary and foreign keys.
- Define constraints to maintain data integrity.
- Create SQL scripts for creating and populating tables for Phase 3 implementation in XAMPP.
- Identify challenges encountered during mapping and justify solutions.

2. EER to Relational Mapping



We mapped the EER diagram to a relational schema by following the EER-to-Relational algorithm:

Entities:

- Each entity (**STORE, MEMBER, MOVIE, STORE_OBJECT, TRANSACTION**) was mapped to a table with its respective attributes. For example, the **STORE** entity became a table with attributes *Store_ID* (primary key), *Address*, and *Phone*.

Relationships:

- The **1:N** relationship between **STORE** and **STORE_OBJECT** was mapped by including *Store_ID* as a foreign key in the **STORE_OBJECT** table.

- The **M:N** relationship between **MEMBER** and **STORE_OBJECT** (via the **TRANSACTION** entity) was mapped by creating a **TRANSACTION** table with foreign keys *User_ID* (referencing **MEMBER**) and *Object_ID* (referencing **STORE_OBJECT**), along with attributes *Trans_ID*, *Start_Date*, *End_Date*, and *Type*.

- The **N:1** relationship between **MOVIE** and **DISK** (is_on) was mapped by including *Movie_ID* as a foreign key in the **DISK** table.

Generalization/Specialization:

- The **STORE_OBJECT** superclass with subclasses **DISK** and **PLAYER** was mapped using the multiple-table strategy. We created a **STORE_OBJECT** table for shared attributes (*Object_ID*, *Store_ID*), and separate tables **DISK** (*Object_ID, Movie_ID, Rent_per_day*) and **PLAYER** (*Object_ID, Featured*) for subclass-specific attributes. The *Object_ID* in **DISK** and **PLAYER** serves as both a primary key and a foreign key referencing **STORE_OBJECT**, ensuring a **1:1** relationship between each subclass row and its corresponding **STORE_OBJECT** row.

Primary Keys and Foreign Keys

- **Primary Keys:** Each table has a primary key to uniquely identify its rows:
  - *Store_ID* for **STORE**
  - *Object_ID* for **STORE_OBJECT**, **DISK**, and **PLAYER**
  - *Trans_ID* for **TRANSACTION**
  - *Movie_ID* for **MOVIE**
  - *User_ID* for **MEMBER**

Foreign Keys: Foreign keys were added to represent relationships:

- **STORE_OBJECT**: *Store_ID* references **STORE**(*Store_ID*).
- **DISK**: *Object_ID* references **STORE_OBJECT**(*Object_ID*), *Movie_ID* references **MOVIE**(*Movie_ID*).
- **PLAYER**: *Object_ID* references **STORE_OBJECT**(*Object_ID*).
- **TRANSACTION**: *User_ID* references **MEMBER**(*User_ID*), *Object_ID* references **STORE_OBJECT**(*Object_ID*).

Constraints Beyond Referential Integrity

- **STORE:** *Phone* must follow a valid format (e.g., 10 digits).
- **DISK:** Rent_per_day must be a positive value (e.g., *Rent_per_day* > 0).
- **PLAYER:** Featured must not be an empty string.
- **TRANSACTION:**
  - End_Date must be greater than or equal to Start_Date.
  - Type must be either **'DISK'** or **'PLAYER'**.
  - A store object cannot be rented out if it's already rented (no overlapping rental periods for the same *Object_ID*).

- **MOVIE:** *Num_of_Disks* must be a positive integer (e.g., _of_Disks > 0).
  - A member cannot rent more than 10 movies and one player at a time.
  - Passwords must not be NULL.

3. SQL queries for Creating and Populating Tables

```
-- CREATE TABLE Statements

CREATE TABLE STORE (
    Store_ID INT PRIMARY KEY,
    Address VARCHAR(100),
    Phone VARCHAR(10)
);

CREATE TABLE MOVIE (
    Movie_ID INT PRIMARY KEY,
    Title VARCHAR(100),
```

```sql
    Genre VARCHAR(50),
    Producer VARCHAR(50),
    Director VARCHAR(50),
    Actor1 VARCHAR(50),
    Actor2 VARCHAR(50),
    Num_of_Disks INT
  );

CREATE TABLE STORE_OBJECT (
    Object_ID INT PRIMARY KEY,
    Store_ID INT,
    FOREIGN KEY (Store_ID) REFERENCES STORE(Store_ID)
);

CREATE TABLE DISK (
    Object_ID INT PRIMARY KEY,
    Movie_ID INT,
    Rent_per_day DECIMAL(5,2),
    Type VARCHAR(10),
    FOREIGN KEY (Object_ID) REFERENCES STORE_OBJECT(Object_ID),
    FOREIGN KEY (Movie_ID) REFERENCES MOVIE(Movie_ID)
 );

CREATE TABLE PLAYER (
    Object_ID INT PRIMARY KEY,
    Features VARCHAR(50),
    FOREIGN KEY (Object_ID) REFERENCES STORE_OBJECT(Object_ID)
 );

CREATE TABLE MEMBER (
    User_ID INT PRIMARY KEY,
    Member_ID INT UNIQUE,
    Name VARCHAR(50),
    Address VARCHAR(100),
    Password VARCHAR(50) NOT NULL
);

CREATE TABLE TRANSACTION (
    Trans_ID INT PRIMARY KEY,
    User_ID INT,
    Object_ID INT,
    Start_Date DATE,
    End_Date DATE,
```

```sql
    Type VARCHAR(10),
    FOREIGN KEY (User_ID) REFERENCES MEMBER(User_ID),
    FOREIGN KEY (Object_ID) REFERENCES STORE_OBJECT(Object_ID)
  );

-- INSERT INTO Statements
INSERT INTO STORE (Store_ID, Address, Phone) VALUES
    (1, '123 Main St', '5551234567'),
    (2, '456 Oak Ave', '5559876543'),
    (3, '789 Elm St, Springfield, IL', '2175550123'),
    (4, '101 Maple Dr, Chicago, IL', '3125550199'),
    (5, '202 Pine Rd, Naperville, IL', '6305550145');

INSERT INTO MOVIE (Movie_ID, Title, Genre, Producer, Director, Actor1,
Actor2, Num_of_Disks) VALUES
    (1, 'The Matrix', 'Sci-Fi', 'Wachowskis', 'Wachowskis', 'Keanu Reeves',
'Laurence Fishburne', 2),
    (2, 'Inception', 'Thriller', 'Christopher Nolan', 'Christopher Nolan',
'Leonardo DiCaprio', 'Joseph Gordon-Levitt', 1),
    (3, 'The Dark Knight', 'Action', 'Christopher Nolan', 'Christopher
Nolan', 'Christian Bale', 'Heath Ledger', 3),
    (4, 'Titanic', 'Romance', 'James Cameron', 'James Cameron', 'Leonardo
DiCaprio', 'Kate Winslet', 2),
    (5, 'Avatar', 'Sci-Fi', 'James Cameron', 'James Cameron', 'Sam
Worthington', 'Zoe Saldana', 4),
    (6, 'Jurassic Park', 'Adventure', 'Steven Spielberg', 'Steven
Spielberg', 'Sam Neill', 'Laura Dern', 2);

INSERT INTO STORE_OBJECT (Object_ID, Store_ID) VALUES
    (1, 1),
    (2, 1),
    (3, 1),
    (4, 1),
    (5, 1),
    (6, 1),
    (7, 1),
    (8, 1),
    (9, 1),
    (10, 5);

INSERT INTO DISK (Object_ID, Movie_ID, Rent_per_day, Type) VALUES
    (1, 1, 2.50, 'DVD'),
    (3, 2, 3.00, 'BLU-RAY'),
```

```sql
    (4, 1, 2.00, 'DVD'),
    (5, 2, 3.50, 'BLU-RAY'),
    (7, 1, 4.00, 'BLU-RAY'),
    (8, 4, 2.75, 'DVD');

INSERT INTO PLAYER (Object_ID, Features) VALUES
    (2, 'DVD/Blu-Ray'),
    (6, 'Blu-Ray'),
    (8, 'DVD/Blu-Ray'),
    (10, 'DVD');

INSERT INTO MEMBER (User_ID, Member_ID, Name, Address, Password) VALUES
    (1, 1001, 'John Doe', '789 Pine St', 'password123'),
    (2, 1002, 'Jane Smith', '321 Elm St', 'securepass456'),
    (3, 1003, 'Michael Brown', '303 Cedar Ln, Springfield, IL',
'mikepass789'),
    (4, 1004, 'Emily Davis', '404 Birch Ave, Chicago, IL', 'emily2025'),
    (5, 1005, 'David Wilson', '505 Willow Ct, Naperville, IL',
'davidpass321'),
    (6, 1006, 'Sarah Johnson', '606 Spruce St, Evanston, IL',
'sarahpass654');

INSERT INTO TRANSACTION (Trans_ID, User_ID, Object_ID, Start_Date,
End_Date, Type) VALUES
    (1, 1, 1, '2025-04-01', '2025-04-05', 'DISK'),
    (2, 2, 2, '2025-04-02', '2025-04-07', 'PLAYER'),
    (3, 3, 4, '2025-04-03', '2025-04-06', 'DISK'),
    (4, 3, 6, '2025-04-03', '2025-04-06', 'PLAYER'),
    (5, 4, 5, '2025-04-05', '2025-04-10', 'DISK'),
    (6, 5, 7, '2025-04-06', '2025-04-09', 'DISK'),
    (7, 6, 8, '2025-04-07', '2025-04-12', 'DISK');
```

4. <u>Difficulties Encountered and Solutions</u>

- **Generalization/Specialization Mapping:** Initially, we mapped the **STORE_OBJECT** hierarchy using a single-table strategy, combining **DISK** and **PLAYER** into one table with a *Type* attribute. However, this led to an issue with the is_on relationship, as **Movie_ID** was incorrectly applicable to **PLAYER** rows. To resolve this, we switched to the multiple-table strategy, creating separate **DISK** and **PLAYER** tables. This ensured that

Movie_ID only exists in **DISK**, aligning with the EER diagram and improving data integrity by eliminating unnecessary NULL values.

- **Primary Key for MEMBER:** The **MEMBER** entity had both *Member_ID* and *User_ID* attributes, and it was unclear which should be the primary key. Since **TRANSACTION** uses *User_ID* as the foreign key, we chose *User_ID* as the primary key for **MEMBER** and made *Member_ID* a unique attribute to ensure consistency across the schema.