04/02/25

Board Game Library and Lending System Database

Description: This is a database management system for lending and record-keeping of a library of board games. Users can loan board games in a similar fashion to a book library.

Team Members:

1. Rohith Gowda Ranganatha
2. Kavit Mehta *(Database path: LIS 751 Orson Database > SCHEMAS > `user027_DB3`)*

Mission Statement:

The Board Game Library and Lending System is designed to help board game enthusiasts, clubs, and community centers manage their game collection, track game loans, and provide insights into game popularity and user engagement.

Mission Objectives:

1. Maintain a catalog of board games, including details such as name, genre, playtime, age group, and number of players.
2. Track game checkouts, returns, and games loaned for events to ensure proper lending management.
3. Record user information, including borrowing history and preferences.
4. Generate reports on most borrowed games, user activity, and overdue items.
5. Provide game recommendations based on user preferences.

Business Rules:

1. **Rule 1 (Attribute-Oriented Rule)**
   A board game must have a minimum and maximum number of players defined.

   To ensure that all games have valid player range information to guide users in game selection, **game_num_players_min** and **game_num_players_max** in the '**game**' table must be greater than zero and cannot be NULL.

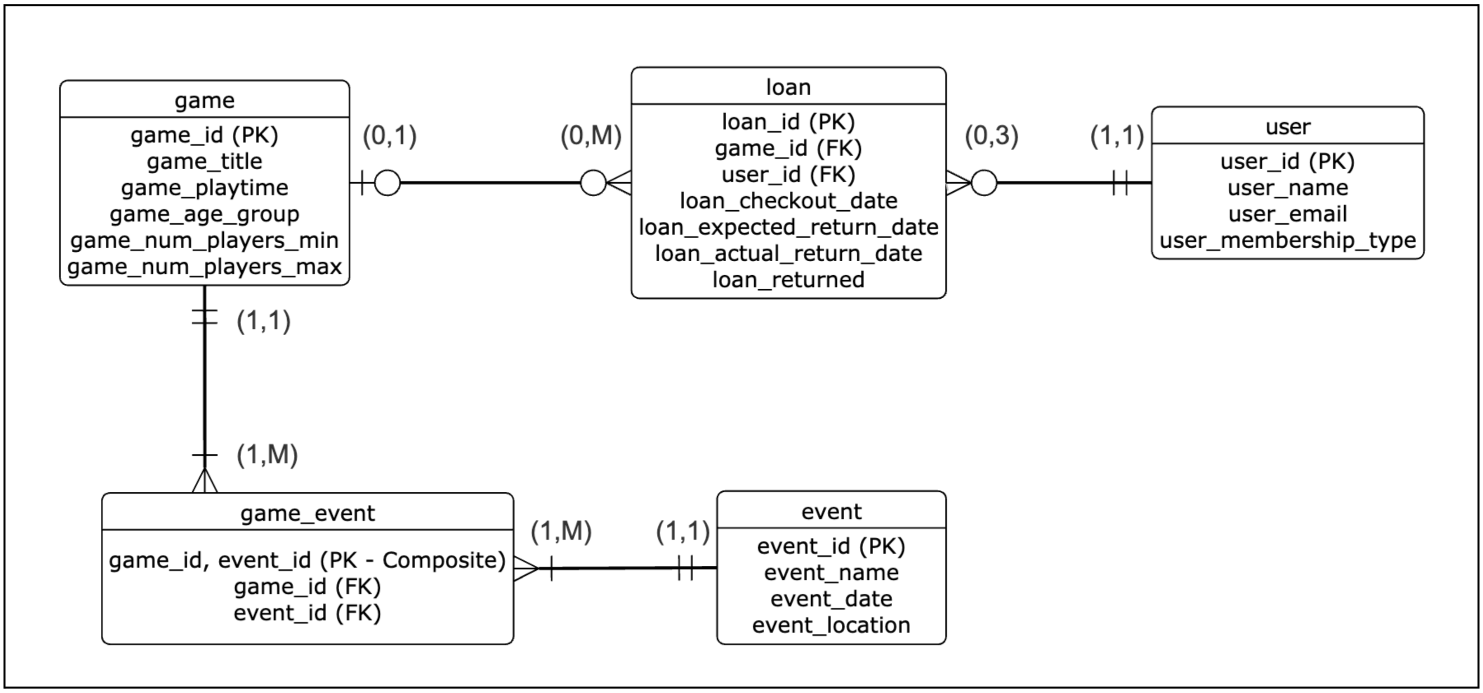
2. **Rule 2 (Relationship-Oriented Rule)**
   A user can't borrow more than 3 games at a time i.e., a maximum of 3 active loans allowed.

   To prevent users from monopolizing the game library, before inserting a new record into the '**loan**' table, check if the user already has 3 active loans (where **loan_returned = 0**).

3. **Rule 3 (General Constraint Rule)**
   A game loan must have a valid return date that is after the checkout date.

   To ensure logical consistency in loan transactions, enforce that **loan_expected_return_date > loan_checkout_date**. If **loan_actual_return_date** is provided, it must be on or after **loan_checkout_date**.

Entity Relationship Diagram:



Data Dictionary Table:

Table Name	Field Name	Description	Key Type	Data Type	Data Length	Null Support	Index	Default Value	Value Range
game	game_id	Unique identifier for each game	PK	INT	4	No	Yes	AUTO_ INCREMENT	N/A
	game_ title	Name of the board game	None	VARCHAR	100	No	Yes	N/A	N/A
	game_ genre	Game category	None	SET	N/A	No	No	N/A	'Strategy', 'Party', 'Co-op', 'Card', 'Dice', 'Family', 'Abstract'
	game_ playtime	Estimated playtime in minutes	None	INT	4	No	No	N/A	Positive integers only

Table Name	Field Name	Description	Key Type	Data Type	Data Length	Null Support	Index	Default Value	Value Range
	game_age_group	Suggested age group	None	ENUM	N/A	Yes	No	'All Ages'	'Kids', 'Teens', 'Adults', 'All Ages'
	game_num_players_min	Minimum number of players	None	INT	4	No	No	N/A	Positive integers only
	game_Num_players_max	Maximum number of players	None	INT	4	No	No	N/A	Positive integers only
user	user_id	Unique identifier for each user	PK	INT	4	No	Yes	AUTO_INCREMENT	N/A
	user_name	Full name of the user	None	VARCHAR	100	No	No	N/A	N/A
	user_email	User's email address	None	VARCHAR	255	No	Yes	N/A	Valid email format
	user_membership_type	User's membership level	None	ENUM	N/A	Yes	No	'Standard'	'Standard', 'Premium'
	user_date_joined	Date user joined	None	DATETIME	N/A	Yes	No	CURRENT_TIMESTAMP	N/A
loan	loan_id	Unique identifier for each loan	PK	INT	4	No	Yes	AUTO_INCREMENT	N/A
	game_id	References game(game_id)	FK	INT	4	No	Yes	N/A	N/A
	user_id	References user(user_id)	FK	INT	4	No	Yes	N/A	N/A
	loan_	Date and	None	DATETIME	N/A	Yes	No	CURRENT	N/A

Table Name	Field Name	Description	Key Type	Data Type	Data Length	Null Support	Index	Default Value	Value Range
loan	checkout_date	time the game was borrowed						_TIMESTAMP	
	loan_expected_return_date	Expected return date	None	DATE	N/A	No	No	N/A	Future date
	loan_actual_return_date	Actual return date if returned	None	DATE	N/A	Yes	No	NULL	Future date
	loan_returned	Indicates if the game has been returned	None	TINYINT(1)	1	Yes	No	0	0 = No, 1 = Yes
event	event_id	Unique identifier for each event	PK	INT	4	No	Yes	AUTO_INCREMENT	N/A
	event_name	Name of the event	None	VARCHAR	100	No	No	N/A	N/A
	event_date	Date and time of the event	None	DATETIME	N/A	No	No	N/A	N/A
	event_location	Event venue location	None	VARCHAR	255	No	No	N/A	N/A
game_event	game_id	References game(game_id)	PK, FK	INT	4	No	Yes	N/A	N/A
	event_id	References event(event_id)	PK, FK	INT	4	No	Yes	N/A	N/A

Queries:

1. List all available board games with their details.
   ```
   SELECT game_id, game_title, game_genre, game_playtime, game_age_group,
   game_num_players_min, game_num_players_max
   FROM game;
   ```

2. Retrieve all overdue loans along with user details.
   ```
   SELECT loan.loan_id, user.user_name, game.game_title,
   loan.loan_expected_return_date
   FROM loan
   JOIN user ON loan.user_id = user.user_id
   JOIN game ON loan.game_id = game.game_id
   WHERE loan.loan_actual_return_date IS NULL AND
   loan.loan_expected_return_date < CURDATE();
   ```

3. Get the Top 5 most frequently borrowed board games.
   ```
   SELECT game.game_title, COUNT(loan.loan_id) AS times_borrowed
   FROM loan
   JOIN game ON loan.game_id = game.game_id
   GROUP BY game.game_id
   ORDER BY times_borrowed DESC
   LIMIT 5;
   ```

4. Show all loans that include details from three tables (user, game, and loan)
   ```
   SELECT loan.loan_id, user.user_name, game.game_title,
   loan.loan_checkout_date, loan.loan_expected_return_date, loan.loan_returned
   FROM loan
   JOIN user ON loan.user_id = user.user_id
   JOIN game ON loan.game_id = game.game_id;
   ```

5. Find all games played at a specific event.
   ```
   SELECT event.event_name, event.event_location, game.game_title
   FROM game_event
   JOIN event ON game_event.event_id = event.event_id
   JOIN game ON game_event.game_id = game.game_id
   WHERE event.event_name = 'Game Night 2025';
   ```

6. Count the number of games borrowed by each user (GROUP BY query).
   ```
   SELECT user.user_name, COUNT(loan.loan_id) AS games_borrowed
   FROM loan
   JOIN user ON loan.user_id = user.user_id
   GROUP BY user.user_id;
   ```

7. **List all users with their latest borrowed game.**

```
SELECT user.user_name, game.game_title, loan.loan_checkout_date
FROM loan
JOIN user ON loan.user_id = user.user_id
JOIN game ON loan.game_id = game.game_id
WHERE loan.loan_checkout_date = (SELECT MAX(l.loan_checkout_date) FROM
loan l WHERE l.user_id = loan.user_id);
```

Note: All the above queries are saved as views in the user027_DB3 schema.