CS340 - Intro to Databases

Video Game Speedruning Database - Proposal

Cameron Brooks & Brayden Plumb

a) Fixes Based on Feedback from Step 1

We received four peer reviews on our Draft submission. Below is a summary of the feedback and actions taken.

Reviewers: Kain Sherman, Olivia Choi, Bryant Aragon, Lucas Feldsien

Actions Based on Feedback

Based on the peer reviews above, we made the following corrections:

1. Naming Consistency – Singular to Plural (Kai Sherman): Changed all entity names from singular to plural (Game → Games, Player → Players, RunCategory → RunCategories, Platform → Platforms, RunSubmission → RunSubmissions) to follow the rubric's suggested convention.

2. Intersection Table Naming Inconsistency (Kai Sherman): Fixed the inconsistency between the ERD and Relationship Summary table. The table previously mentioned "GamePlatforms" while the ERD showed "GameOnPlatform." We standardized to "GamesOnPlatforms" (plural, matching our other entity naming convention) across both the ERD and all documentation.

3. Game-RunCategory Relationship Wording Confusion (Kai Sherman, Lucas Feldsien): Both reviewers identified that the relationship description was contradictory – the Game section implied one RunCategory could be associated with many Games, while the RunCategory section stated it could only belong to one Game. We corrected this by clarifying that the relationship is 1:M from Games to RunCategories: one Game can have multiple RunCategories, but each RunCategory belongs to exactly one Game. The gameID FK in RunCategories enforces this constraint.

4. Numerical Scope – Annual Run Estimates (Kai Sherman, Olivia Choi): Kai suggested more concrete numbers to scope the system, and Olivia specifically recommended estimating annual run submissions. We added the calculation: with 25–100 players per competition and 1–30 runs per player per year, the system is expected to handle approximately 750–3,000 run submissions annually.

5. Entity Highlighting in Overview (Kai Sherman): Added colored highlighting to entity names in the Overview section so they can be quickly mapped to the Database Outline, improving readability as suggested.

6. Verification Date Attribute (Olivia Choi): Added verifiedDate: date attribute to RunSubmissions to track when a run was verified by an admin, supporting admin workflows and record-keeping as suggested.

7. User Roles Clarification (Olivia Choi): Clarified in the Overview that the website will be "editable by admins, and viewable by players," explicitly stating who will use the site and their primary actions as suggested.

8. Run Category Edge Cases (Olivia Choi): Olivia asked whether every run submission must always belong to exactly one category. We confirm this is by design: each RunSubmission has a NOT NULL FK to RunCategories, enforcing that every run belongs to exactly one category. This matches standard speedrunning practice where runs are submitted to specific categories with distinct rulesets (e.g., a "100%" run cannot simultaneously be an "Any%" run).

Additional Design Changes for Step 2

- **Normalization Verified:** Applied the normalization process to all tables *(details in the Normalization section below)*. All tables satisfy 3NF.
- **Cascade Constraints:** Added ON DELETE CASCADE and ON UPDATE CASCADE to all foreign key constraints, as recommended by the course materials.
- **Schema Diagram:** Created based on DLL implementation, and generated by MySQLWorkbench.
- **Example Data:** Added 3-5 rows of sample data per table that demonstrate all relationship types in action.

b) Project Overview

The following overview is a fictional scenario we are using as a framework for our project.

Northern Oregon Speedrunners Association is a nonprofit organization that runs various video game speed running events in local communities in Northern Oregon. They would like to have a website where they can store the various speedrun submissions from their competitions, which typically consist of 25 - 100 players, depending on the competition. The amount of runs submitted per year will vary drastically, depending on activity in the local speedrunning community, but on average most players will submit 1 - 30 runs per year. Based on these figures, the system is expected to handle approximately 750-3,000 run submissions annually.

Due to the limited needs of this system, they don't need a way to categorize the runs by the event they are a part of, and storing them by date is enough information for them to infer the event they are from. This then allows users to submit runs to be verified that they may complete from home by sending the info about their run to an administrator of the association.

The association only wants ***run submissions*** to be able to be submitted by admin board members of the association. They would like to be able to see the name of the ***players***, the runs they have successfully completed, what ***game*** they speedran, what ***run categories*** these runs are in, and what ***platform*** they were on when they completed the run. This information will serve as a record for all local speedrunners in the area to show off their previous times, and gives *Northern Oregon Speedrunners Association* a record to determine local speedrunning records. This website will be editable by admins, and viewable by players.

Extra Note: To accommodate for players who may have moved to a different part of the world, the player entity will contain a country.

Database Outline

Players

Description: A speedrunner participant (via a displayName) that can speedrun various games.

- playerID: int, auto_increment, unique, not NULL, PK
- displayName: varchar(100), not NULL, unique - The player's username/unique handle
- country: varchar(100) - The player's country of residence
- **Relationship(s)**
    - 1:M with RunSubmission with playerID as a FK in RunSubmission (one Player can have many RunSubmissions)

Games

Description: A video game that is able to be speedrun.

- gameID: int, auto_increment, unique, not NULL, PK
- title: varchar(255), not NULL - - The name of the video game
- releaseYear: year(4) - year the game was released.
- developer: varchar(255) - The company that developed the game
- **Relationship(s)**
    - 1:M with RunSubmission with gameID as a FK in RunSubmission (one Game can have many RunSubmissions)
    - 1:M with RunCategory with gameID as a FK in RunCategory (one Game can have multiple RunCategories)
    - M:N with Platform with an intersection table with both gameID and submissionID as FKs in that table (one Game can be on many Platforms; one Platform can have many Games)

RunCategories

Description: A speedrun submission type which different games provide.

- runCategoryID: int, auto_increment, unique, not NULL, PK
- name: varchar(100), not NULL - The category name (*e.g., "Any%", "Glitchless", "100%")*
- ruleset: text - description of rules
- **Relationship(s)**
  - 1:M with RunSubmission with runCategoryID as a FK in RunSubmission. (one platform that can have many runsubmissions)
  - M:1 with Game with gameID as a FK in this RunCategory. (RunCategory can have one and only one Game it's associated with, due to each Category having specific rules related to its game.)

Platforms

Description: The hardware or environment a game can be played on.

- platformID: int, auto_increment, unique, not NULL, PK
- name: varchar(100), not NULL, unique - The platform name *(e.g., "PC", "Nintendo Switch", "PlayStation 5")*
- **Relationship(s)**
  - 1:M with RunSubmission with platformID as a FK in RunSubmission (one platform that can have many run submissions)
  - M:N with Game with an intersection table that has platformID and gameID as FKs in this table (one Platform can have many Games; one Game can be on many platforms)

RunSubmissions

Description: A specific speedrun record that has all the information about that speedrun attempt.

- runSubmissionID: int, auto_increment, unique, not NULL, PK
- runTime: time(3), not NULL - The duration of the speedrun (HH:MM:SS)
- submissionDate: date, not NULL - The date the run was submitted
- verified: boolean, default FALSE - Whether the run has been verified by an admin
- verifiedDate: date, not NULL - The date the run was verified
- playerID: int, not NULL, FK - References Player.playerID
- gameID: int, not NULL, FK - References Game.gameID
- platformID: int, not NULL, FK - References Platform.platformID
- runCategoryID: int, not NULL, FK - References RunCategory.runCategoryID
- videoLink: varchar(2048) - URL to video proof of the run *(max url character count is 2048)*
- **Relationship(s)**
  - M:1 with Player with playerID as a FK in this RunSubmission - (many RunSubmissions belong to one Player)

- M:1 with Game with gameID as a FK in this RunSubmission - (many RunSubmissions belong to one Game)
- M:1 with Platform with platformID as a FK in this RunSubmission - (many RunSubmissions were played on one Platform)

GamesOnPlatforms

(Intersection Table)
Description: A specific speedrun record that has all the information about that speedrun attempt.

- gameOnPlatformID: int, auto_increment, unique, not NULL, PK
- gameID: int, not NULL, FK
- platformID: int, not NULL, FK
- **Constraint:** Unique combination of *(gameID, platformID)*

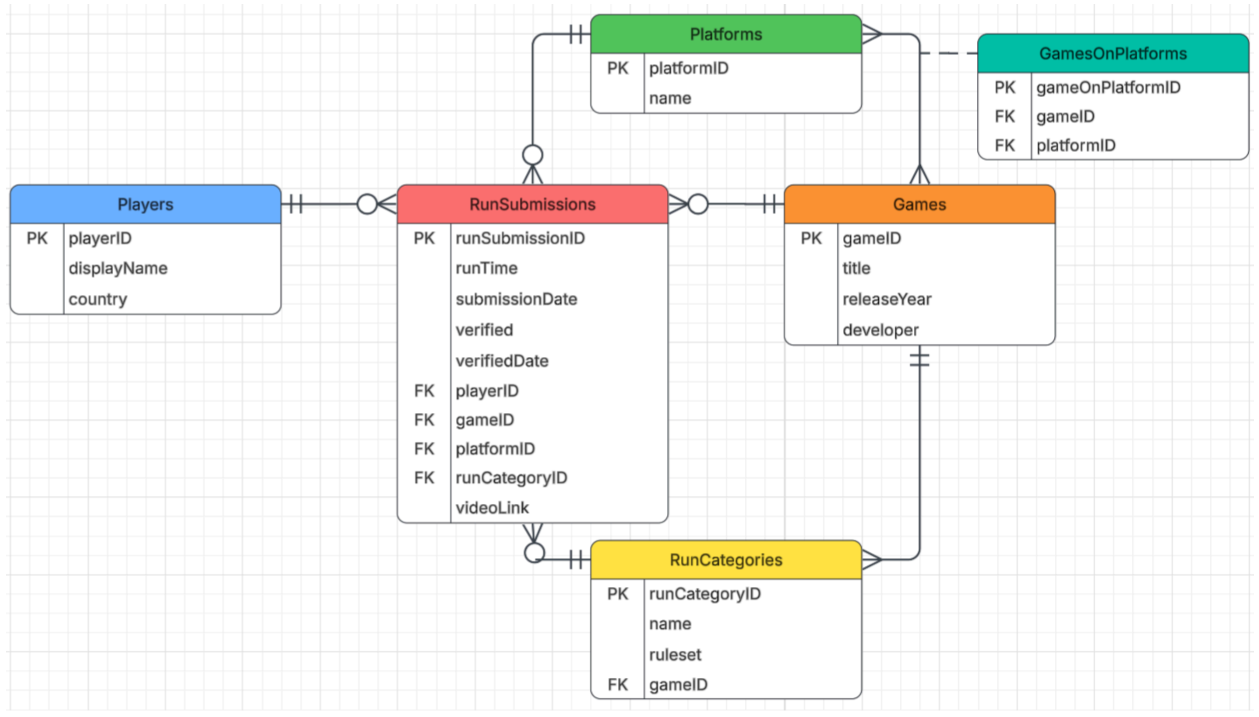
Relationship Summary

Players→RunSubmissions	**1:M**	playerID FK in RunSubmission
Games→RunSubmissions	**1:M**	gameID FK in RunSubmission
Platforms→RunSubmissions	**1:M**	platformID FK in RunSubmission
RunCategories→RunSubmissions	**1:M**	runCategoryID FK in RunSubmission
Games ↔ Platforms	**M:N**	GameOnPlatform intersection table
Games→RunCategories	**1:M**	gameID FK in RunCategory

Total: 6 relationships (Including 1 M:N) **across 6 tables**

c) Entity-Relationship Diagram

Demonstrates a visual representation of the relationships between the entities listed above.



Note: The ERD matches the Database Outline and Schema Diagram. All entity names are plural, the intersection table is named GamesOnPlatforms, and the verifiedDate attribute has been added to RunSubmissions.

---

d) Schema Diagram

The schema diagram below represents the final normalized database design. All tables, their attributes, primary keys (PK), and foreign keys (FK) are shown, along with the relationships between tables.

Normalization Analysis

We applied the normalization process that we learned about in the Exploration - Normalization Steps. We started by examining a hypothetical sample report (similar to the construction company example from the exploration) and checked for redundancies and anomalies.

Hypothetical Denormalized Report

Consider a report of various speedruns that an administrator would want to add to the database, listing all run submissions based on the following criteria:

Player	Country	Game	Developer	Category	Platform	RunTime	Verified

SpeedDemon42	US	Mario	Nintendo	Any %	N64	15:35	Yes
SpeedDemon42	US	Mario	Nintendo	16 Star	N64	49:12	Yes
PixelRunner	CA	Zelda: OoT	Nintendo	Any %	Switch	18:22	No
GlitchHunter	US	Celeste	MaddyMakes	Any %	PC	32:45	Yes
NightOwlPlays	JP	Portal	Valve	Inbounds	PC	14:08	No

Redundancy and Anomaly Analysis

If all data were stored in a single flat table, the following issues would occur:

- **Redundant Data:** “SpeedDemon42” and “US” are repeated for each run by that player. “Nintendo” is repeated for each game developed by Nintendo. “Super Mario 64” is repeated for each run of that game.
- **Insert Anomaly:** To add a new game, we would need to create a dummy run submission. To add a new player, we would need a dummy run as well.
- **Update Anomaly:** To change a player’s country, it would require updating every row with that player. Changing a game’s developer name would require the same process.
- **Delete Anomaly:** Deleting the only run for a game would lose all information about that game. Deleting the only run for a player would result in the same issue.

1NF Verification

All tables in the final Schema are in 1NF:
- Table has a defined primary key (no repeat groups)
- All attributes contain atomic values (single value per cell, and no lists or sets)
- No table has repeating groups, and each row is uniquely identifiable by its auto incrementing PK.

2NF Verification

All tables in the final Schema are in 2NF:
- No table has a composite primary key with partial dependencies. They all use single column auto-increment primary keys, so partial dependency is impossible.
- The GamesOnPlatforms table contains no non-key attributes beyond the two foreign keys, so there are no partial dependencies.

3NF Verification

Finally, all tables in the final Schema are in 3NF:
- For all the tables, none of them have attributes that rely indirectly on another through a third value. All non key values are data facts about the entities, and all references to other entities are through FKs.

Conclusion: All six tables satisfy 3NF. No denormalization decisions were necessary, as the schema was already properly decomposed from the initial design in Step 1

---

e) Example Data

The following tables show the sample data inserted into each table. This data is included in the accompanying DLL.sql file via INSERT statements. The data demonstrates how foreign keys enforce the 1:M relationships between the entities, and how the intersection table shows the M:N relationship in action.

Players

playerID	displayName	country
1	SpeedDemon42	United States
2	PixelRunner	Canada
3	GlitchHunter	United States
4	NightOwlPlays	Japan
5	TurboTina	United Kingdom

Games

gameID	title	releaseYear	developer
1	Super Mario 64	1996	Nintendo
2	The Legend of Zelda: Ocarina of Time	1998	Nintendo
3	Celeste	2018	Maddy Makes Games
4	Portal	2007	Valve

RunCategories

runCategoryID	name	ruleset	gameID (FK)
1	Any %	Complete the game as fast as possible by any means.	1
2	16 Star	Complete the game collecting exactly 16 stars.	1

3	Any %	Complete the game as fast as possible by any means.	2
4	Any %	Complete the game as fast as possible by any means.	3
5	Inbounds	Complete all chambers without leaving intended boundaries.	4

Platforms

platformID	name
1	PC
2	Nintendo 64
3	Nintendo Switch
4	PlayStation 5

RunSubmissions

ID	runTime	subDate	verified	verifiedDate	playerID	gameID	platID	catID
1	00:15:35.230	2026-01-10	True	2026-01-11	1	1	2	1
2	00:49:12.800	2026-01-15	True	2026-01-16	1	1	2	2
3	00:18:22.450	2026-01-20	False	NULL	2	2	3	3
4	00:32:45.110	2026-02-01	True	2026-02-02	3	3	1	4
5	00:14:08.670	2026-02-03	False	NULL	4	4	1	5

Note: Column headers abbreviated for readability. Full column names: runSubmissionID, runTime, submissionDate, verified, verifiedDate, playerID, gameID, platformID, runCategoryID. The videoLink column is omitted from the table for space but is included in the DLL.sql file.

GamesOnPlatforms

gameOnPlatformID	gameID (FK)	platformID (FK)
1	1	2
2	1	3
3	1	1

4	2	2
5	2	3
6	3	1
7	4	1

Relationship demonstrations:

- **1:M Players → RunSubmissions:** Player 1 (SpeedDemon42) has 2 runs (IDs 1 and 2), while Player 5 (TurboTina) has 0 runs.
- **1:M Games → RunSubmissions:** Game 1 (Super Mario 64) has 2 runs (IDs 1 and 2).
- **1:M RunCategories → RunSubmissions:** Each run belongs to exactly one category.
- **NULL verifiedDate:** Runs 3 and 5 are unverified, so verifiedDate is NULL.
- **NULL videoLink:** Run 4 has no video link, showing that videoLink is optional.

Citations

None. All content of this proposal is our original work.