CS4.301 — Data And Applications

Project Phase 2: ER Model

Entity-Relationship Model

for

The Olympian Codex

Team RNA (Team 42)

Kavish Vora	2024101100
Dhyey Thummar	2024101024
Shivansh Santoki	2024101119

October 30, 2025

Summary of Diagram Notational Assumptions

This diagram was created using Mermaid's flowchart tool (mermaidchart.com), which requires visual workarounds to represent formal Entity-Relationship (ER) notation. The following conventions have been adopted:

- **Weak Entity (e.g., Quest_Log, Sighting_Log)**
  - *Standard:* Double-lined rectangle.
  - *Workaround:* Represented by an orange-filled rectangle with an extra-thick (6px) border.
- **Identifying Relationship (e.g., Participates_In, Tracks)**
  - *Standard:* Double-lined rhombus.
  - *Workaround:* Represented by a purple-filled rhombus with an extra-thick (6px) border to distinguish it from regular relationships (which are red).
- **Total Participation (e.g., a Demigod's participation in Parents)**
  - *Standard:* A double line.
  - *Workaround:* Represented by a thick, yellow-colored line to distinguish it from the standard thin lines used for partial participation.
- **Partial Key / Discriminator (e.g., Sighting_Timestamp)**
  - *Standard:* Dotted underline.
  - *Workaround:* Represented by a solid underline combined with a blue text color for identification.

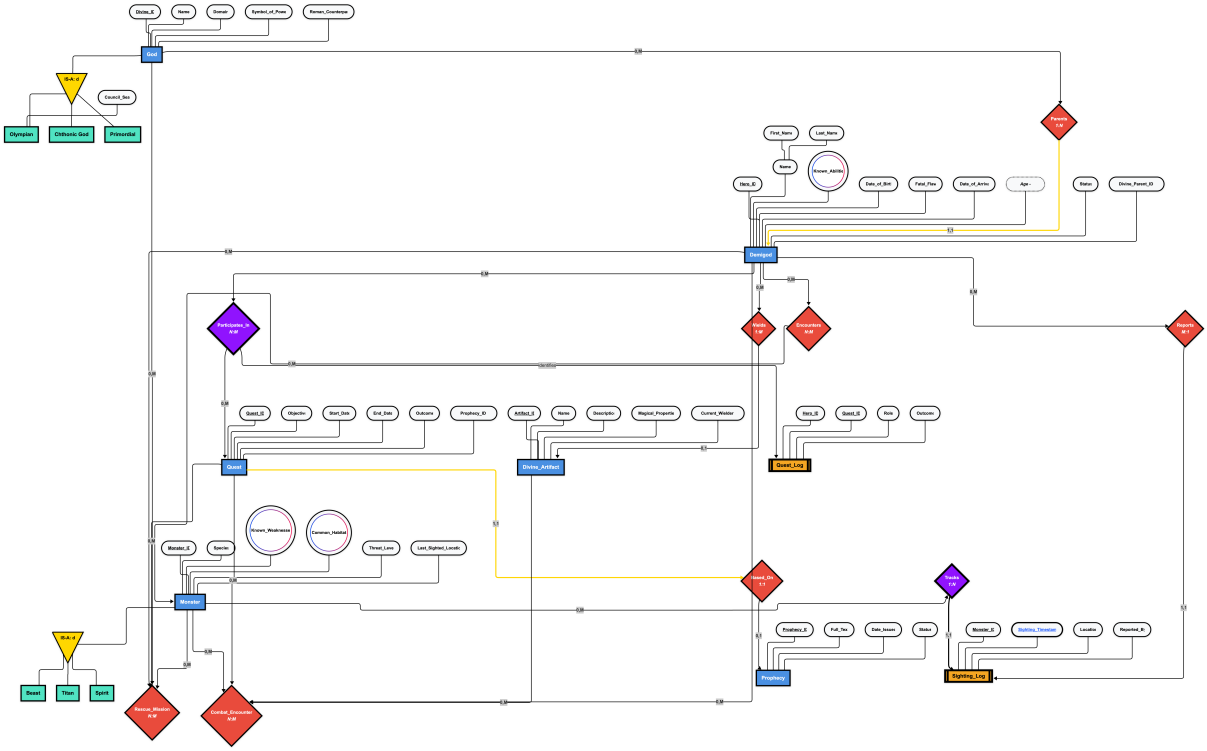

Figure 1: Entity-Relationship Diagram for The Olympian Codex.

View-Only version available at:
https://www.mermaidchart.com/d/0809770e-887b-4d43-a764-05456134320c

Design Quality Analysis

This analysis identifies potential data quality issues in the ER model.

5.1 Redundancy Analysis

Redundancy occurs when the same piece of information can be stored or derived from multiple places, leading to potential inconsistencies.

- **Identified Redundancy:** The ‘Monster‘ entity contains an attribute ‘Last_Sighted_Location‘. However, location data is also stored in the ‘Sighting_Log‘ weak entity, which has a ‘Location‘ attribute and a ‘Sighting_Timestamp‘. The last sighted location for any monster can always be derived by finding the most recent record in the ‘Sighting_Log‘ for that specific ‘Monster_ID‘.
- **Potential Problem:** Storing this location in two places could lead to data inconsistency. If a new sighting is added to the ‘Sighting_Log‘ but the ‘Last_Sighted_Location‘ attribute in the ‘Monster‘ table is not updated simultaneously, the database will contain conflicting information about the monster’s whereabouts.

5.2 Update Anomaly

An update anomaly exists when a simple change requires updating multiple records, creating a risk of partial updates.

- **Example Scenario:** Following from the redundancy issue above, an update anomaly is present. When a demigod reports a new monster sighting, two separate write operations are required: first, an ‘INSERT‘ into the ‘Sighting_Log‘ table, and second, an ‘UPDATE‘ on the ‘Monster‘ table to change the ‘Last_Sighted_Location‘. If the second operation fails for any reason, the data becomes inconsistent, which could be critical for strategic decisions at Camp Half-Blood.

5.3 Insertion Anomaly

An insertion anomaly occurs when you cannot add certain data because other required data does not yet exist.

- **Example Scenario:** A new camper arrives at Camp Half-Blood. They are clearly a demigod, but their divine parent has not yet claimed them. You need to add them to the Demigod entity to track their training and assign them to a cabin.
  Due to the "total participation" rule, every demigod record must have a Divine_Parent_ID. This creates an insertion anomaly when a new, unclaimed camper arrives. It’s impossible to add them to the database because their divine parent is unknown, forcing you to either wait or create a placeholder "Unknown" god, which compromises data integrity.

5.4 Deletion Anomaly

A deletion anomaly is the unintentional loss of other important information when a record is deleted.

- **Example Scenario:** The relationship between ‘God‘ and ‘Demigod‘ is 1:M, and referential integrity is enforced. Consider a minor god who fades from existence and must be deleted from the ‘God‘ table. If we delete this god’s record, the database must decide what to do with the records of their demigod children who reference the god’s ‘Divine_Parent_ID‘. A cascading delete would erase the records of potentially famous heroes, causing an unacceptable loss of historical data. Alternatively, setting their ‘Divine_Parent_ID‘ to NULL would orphan the records and violate the constraint that a demigod has "exactly one divine parent".