Constructor University

F25_CO-560-B Databases Project

Project Name: Cultura

Cultura connects students, encourages cultural exchange, and facilitates participation in campus life. Its user-centered design helps both students and administrators engage with and manage the community effectively, creating an inclusive and vibrant environment.

Mohammad Abdullah Khurram - mkhurram@constructor.university

Miras Algatay - malgatay@constructor.university

Ayah Rmaili - armaili@constructor.university

Akshat Singh - aksingh@constructor.university

September 18, 2025

Mapping approach for users and roles:

- The user entity from the ER diagram is mapped to a table users.
- Attributes: user_id (primary key), name, email (unique), password, status (ENUM('active','suspended','deactivated'), default 'active'), created_at (timestamp).
- A separate table roles handles the list of possible roles.
- Attributes: role_id (primary key), role_name (unique), description.
- As a user can have multiple roles and a role can belong to many, a join table user_roles is introduced to represent the many-to-many relationship.
- Attributes: user_id (foreign key to users), role_id (foreign key to roles), assigned_at (timestamp), and a composite primary key (user_id, role_id) to prevent duplicates.
- Foreign keys: user_roles.user_id references users.user_id ON DELETE CASCADE so that if a user deletes their account, all their role assignments are removed automatically.
- user_roles.role_id references roles.role_id ON DELETE CASCADE so that if a role is deleted from the catalog, all assignments of that role are also cleaned up.
- This design keeps role definitions centralized and lets you assign any combination of roles to any user without changing the users schema. It also provides a clean review point through assigned_at and supports future expansion (e.g., role descriptions, permissions, or role hierarchies) without modifying existing tables.

Alternative Considered:

One other way was to put a single role_id directly on the users table (one-to-many) or to use an ENUM/VARCHAR for the user's role. However, this would make it harder to support multiple roles per user, would be less flexible to maintain, and would require schema changes whenever new roles are added. The separate roles catalogue and the user_roles mapping table keeps the schema stable, avoids duplication, and supports easy extension of roles and permissions in the future.

Mapping approach for post and post types:

- A table posts is mapped to the post entity from the ER diagram.
- Features: title, content, country, theme, attachments_url, created_at, updated_at, post_id (primary key), creator_id (foreign key to users), and type_id (foreign key to post_types).
- To ensure that every post is from an existing user, a foreign key is utilised for creator_id. In the event that a user deletes their account, all of their postings will be deleted because ON DELETE CASCADE is selected.
- A distinct table post_types was constructed with the attributes type_id (primary key) and name in order to handle the post type rather than storing it directly in the posts table.
- This prevents replication, increases schema stability, and makes it simple to extend types without altering the post's schema.
- A post type that is still in use can't be deleted by using ON DELETE RESTRICT.

Alternative Considered:

Another option was to use a basic VARCHAR or ENUM for type inside the posts table, but this would be more difficult to maintain and less adaptable. To maintain the schema tidy and make it simpler to add types for future growth if necessary, a separate table was created.

Mapping approach for events and categories:

- The table Event is mapped to the events entity from the ER diagram. Name, description, event_poster_url, location, start_time, end_time, capacity (CHECK >= 0), is_published (boolean), created_at (timestamp), datetime, and event_id (primary key, auto-increment) are among the attributes.
- A distinct table called Category is used to manage categories.
- Features: name (unique), category_id (primary key).
- The many-to-many relationship is represented by a join table called EventCategory since an event can fall under more than one category and a category can apply to numerous events.
- Features: composite primary key (event_id, category_id) to avoid duplication, event_id (foreign key to Event), and category_id (foreign key to Category).
- Foreign keys: References to EventCategory.event_id Category.category_id ON DELETE CASCADE is used by EventCategory.category_id and Event.event_id ON DELETE CASCADE.
- This approach facilitates future growth without altering current tables, maintains category definitions centrally, and permits numerous categories per event without altering the Event table.

Alternative Considered:

Using ENUM/VARCHAR or storing a single category_id in Event were two further methods. This would hinder flexibility, make renaming more difficult, and restrict each event to a single category. The structure remains adaptable and future-proof by combining distinct tables with a join table.

Mapping approach for RSVP and RSVP Status

- RSVP statuses are associated with the rsvp_status table.
- Features: is_confirmed (boolean), code (unique: going, waitlisted, not_going, interested), and status_id (primary key).
- RSVPs are kept in an independent table called rsvp.
- Features: email_confirmed (boolean), created_at, event_id (foreign key to Event), user_id (foreign key to users), status_id (foreign key to rsvp_status), and rsvp_id (primary key).
- One RSVP per person per event is guaranteed via a uniqueness constraint on (event_id, user_id).
- When a person or event is removed, related RSVPs are cleaned up thanks to foreign keys that guarantee consistency and cascade deletes.

Alternative Considered:

Using ENUM/VARCHAR for status directly in the rsvp database is a more straightforward approach, but it limits flexibility and the ability to add attributes to statuses (like is_confirmed). An alternative to rsvp_id was to use a composite primary key (event_id, user_id). The surrogate rsvp_id makes it simpler to reference individual RSVPs (for instance, for logs or external references), even though it also ensures uniqueness. Although status, user, and queue indexes were taken into consideration for efficiency, they are not necessary in the basic structure and can be added later depending on query patterns.