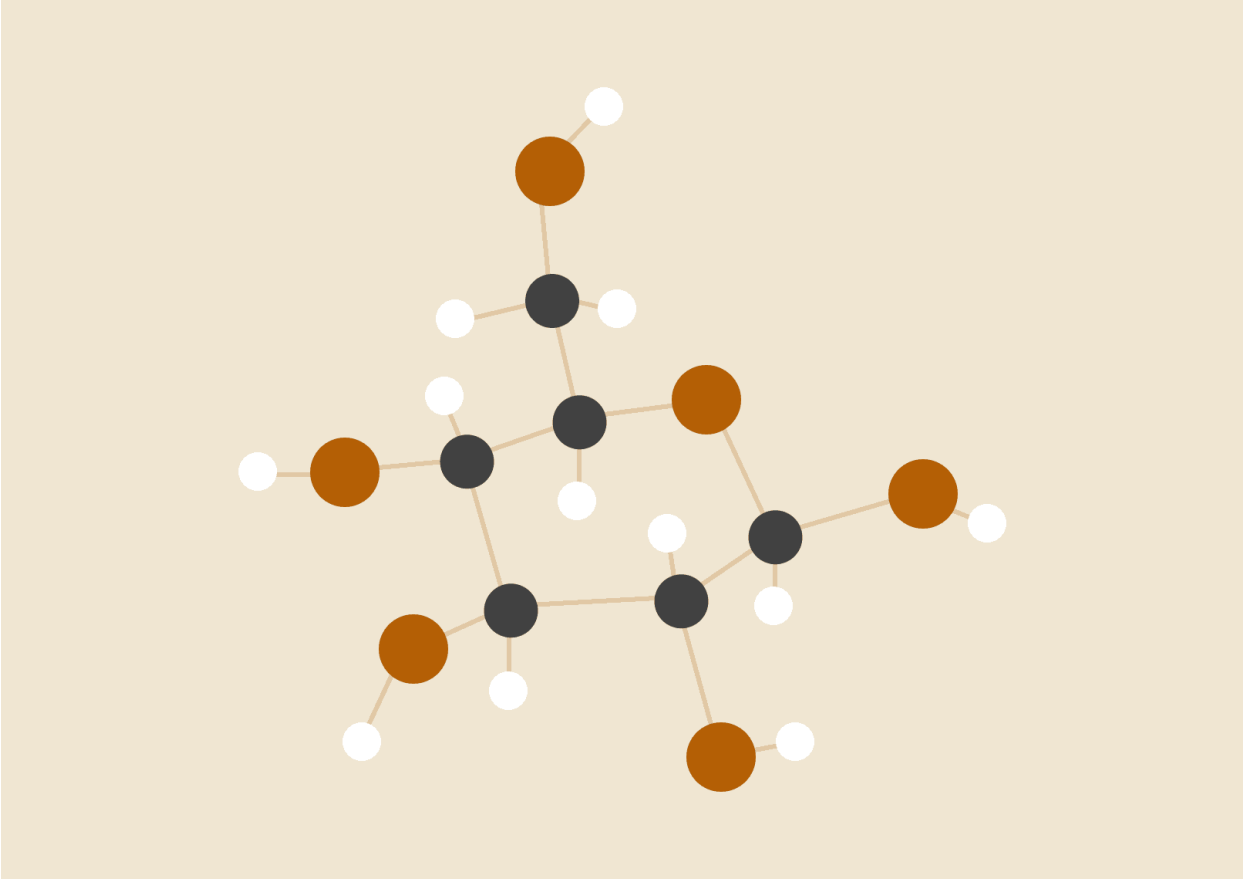
Cluster

Public Collaborative Forum & Social News Aggregation



Mohd Zaid, Nikhil Agrawal & Jai Ahuja

2024353, 2024380 & 2024275

Tutorial Group **1** | Project Group **8**

Assigned TAs: **Sharen Ganesh** and **Shivangi Gaur**

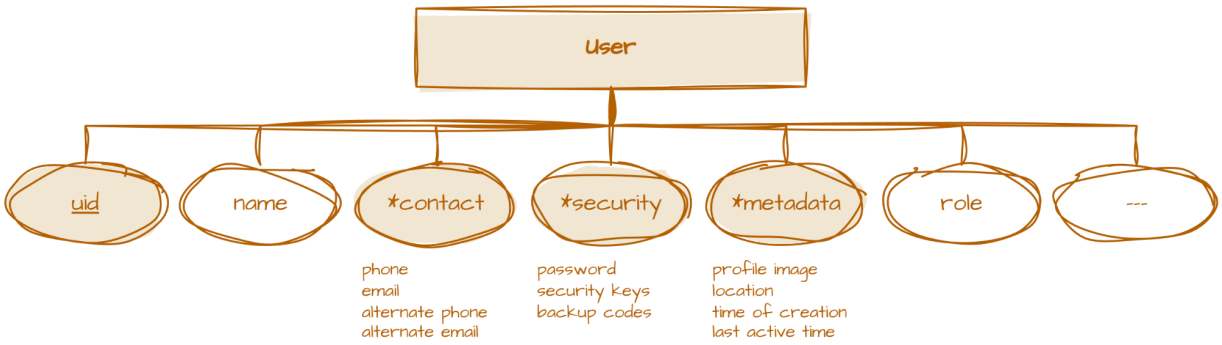
ENTITY RELATIONSHIP MODELS

The design of the **Entity Relationship** (**ER**) model begins with the identification of core entities derived from the requirements specification outlined in **Task 1**. Subsequently, the relationships and cardinality constraints governing these entities are established to accurately reflect the system's business logic.

STRONG & WEAK ENTITIES

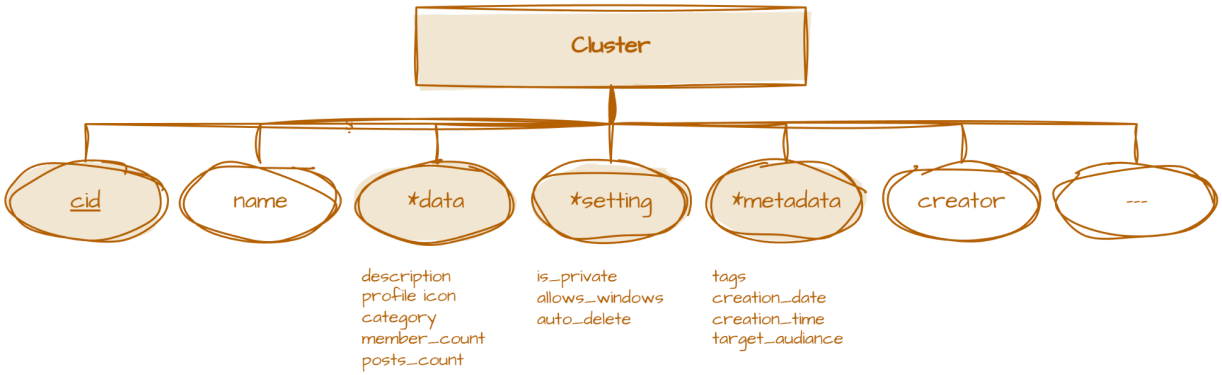
User Entity (**Strong**):

- Represents the registered individuals interacting with the system.
- This is the central entity and **target audience** for this project.
- The user's existence does not depend on any other entity in the database.



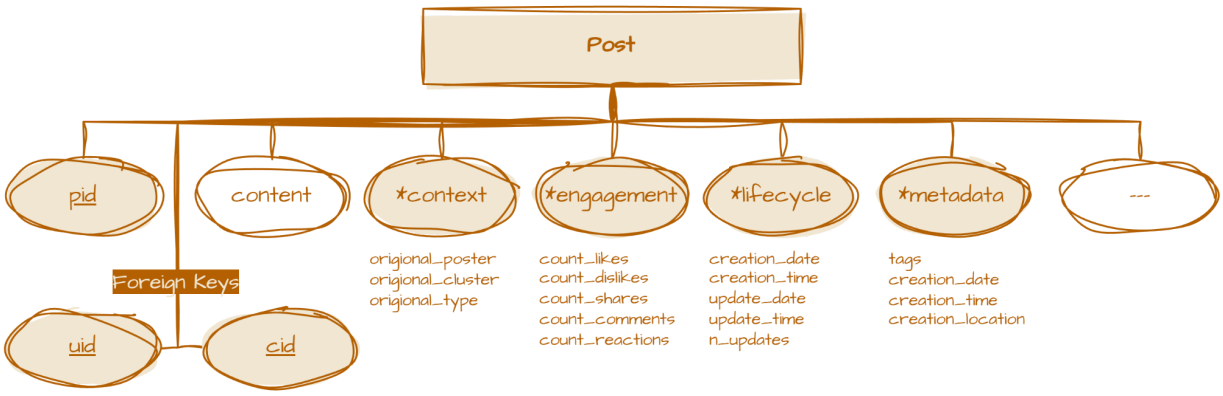
Cluster Entity (**Strong**):

- Represents the distinct **interest groups** or **communities** within the platform.
- It serves as the primary container for all content.
- It can exist without being linked to a specific entity, though it must have a creator.



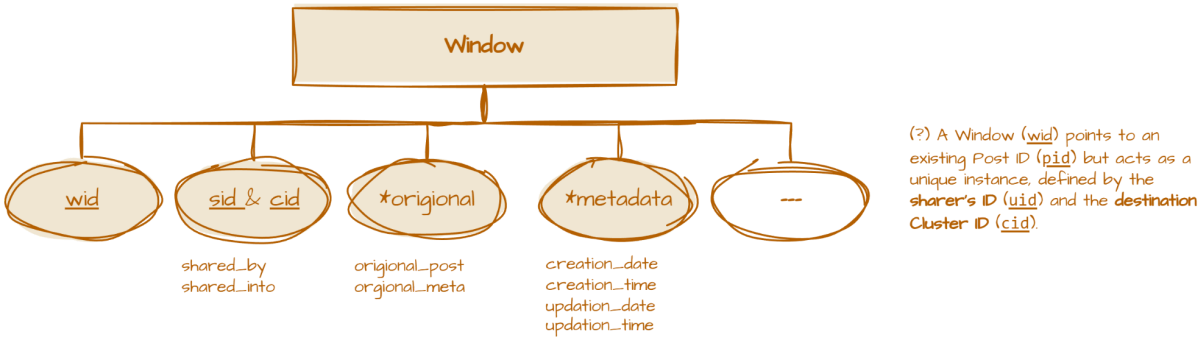
Post Entity (**Weak**):

- The fundamental unit of discussion within the platform.
- It acts as the parent entity for Comments and Reactions.
- It functions as a **Supertype** that encompasses itself and the **Window** (reference).
- It depends upon the existence of **User** and **Cluster,** therefore Weak.



Window Entity (**Weak / Subtype**):

- A specialized form of content that acts as a "**reference wrapper**".
- It points to an existing post in a **different cluster** without duplicating discussion.
- It allows for content curation while maintaining a single source of truth.
- It depends upon the existence of **User** and **Cluster,** therefore Weak.

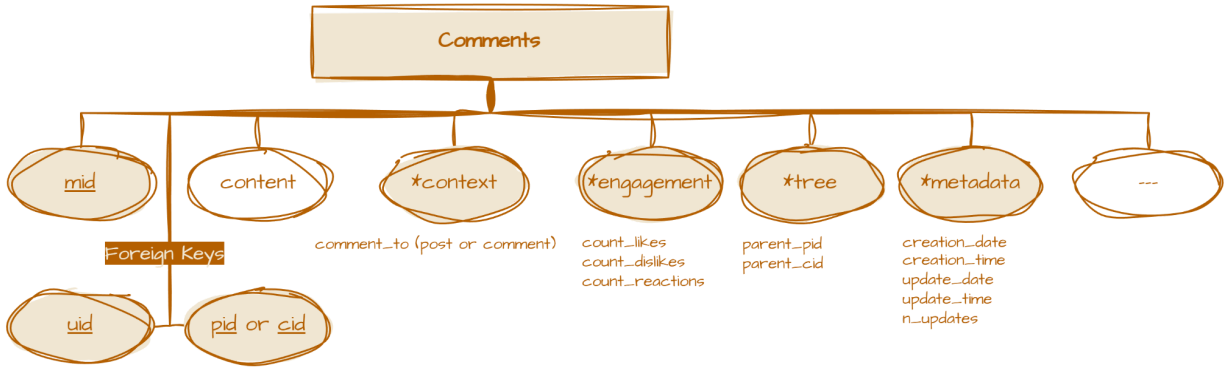


A Window shares the same identifier structure as a Post (**wid ≈ pid**). Although it references the content of an original pid, the Window is distinct because it is created by a **different user (uid)** in a **different cluster (cid)**. This allows content to be cross-posted while maintaining a unique identity in the new location.

** it requires future implementation concerns to deal with this. (considering different)

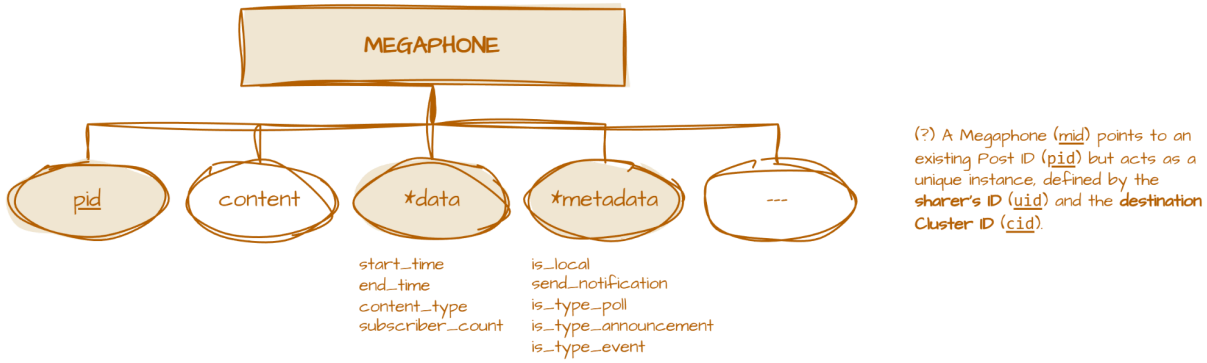
Comment Entity (**Weak**):

- The critical feature here is the "**recursive**" capability.
- It depends upon the existence of **User** and **Post,** therefore Weak.



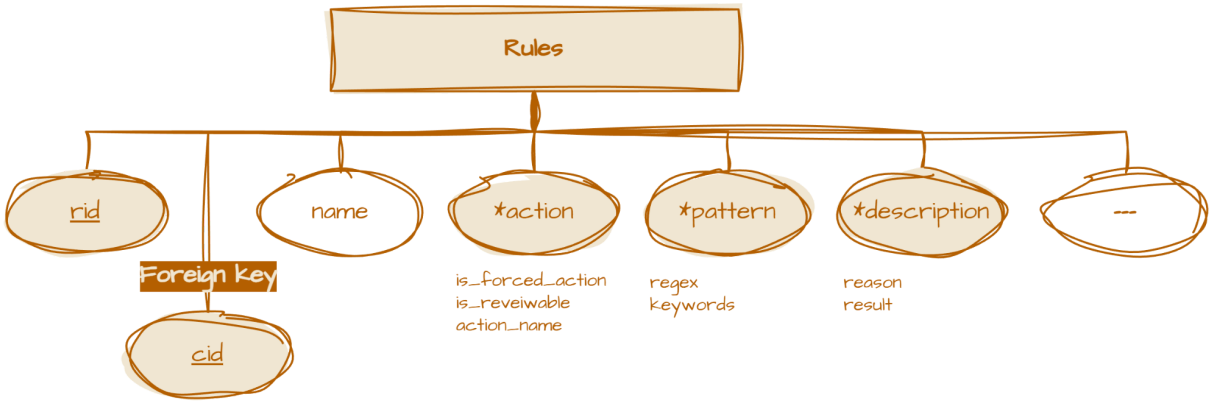
Megaphone Entity (**Weak**):

- A time-sensitive overlay applied to specific posts to turn into announcements.
- It is a **Weak Entity** because it cannot exist without a corresponding Post.



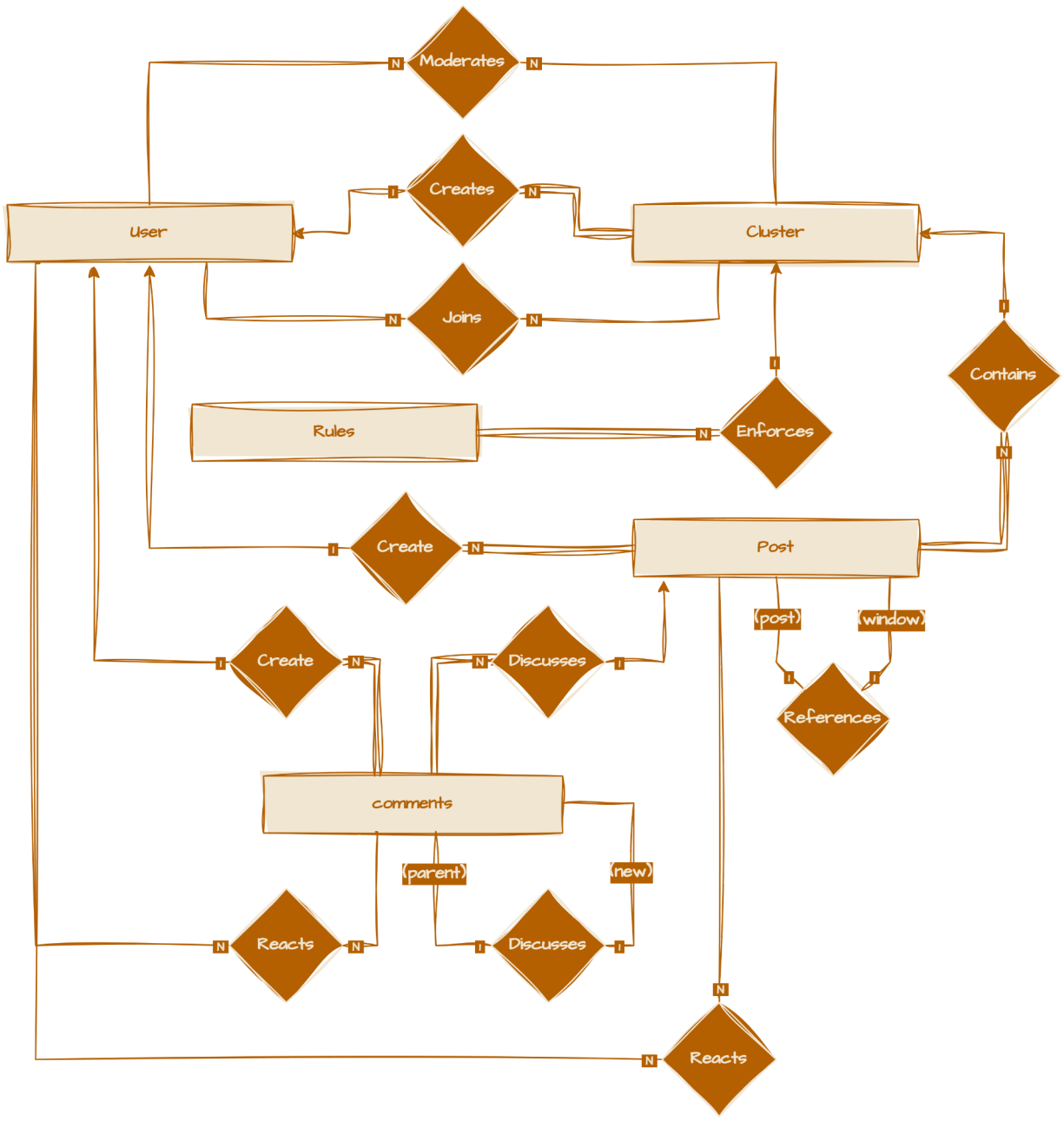
Rules Entity (**Weak**):

- Represents the automated moderation logic configured for a specific Cluster.
- It is a **Weak Entity** dependent on the Cluster custom for every Cluster.



ENTITY RELATIONSHIP MODEL

Iteration of all possible relations between different entities, some entities however have been collapsed into relations such as **Windows** can be considered as reference of post to a post for simplification of model, same was opted for **Reactions**.



** iterations may be made to this diagram in future in order to better adapt to changing implementation constraints that we may face in future.

RELATIONAL MODELS

In this segment we go through all relations and argue whether it should exist as a table of relation or it is easier for it to be incorporated into an existing entity that was created above.

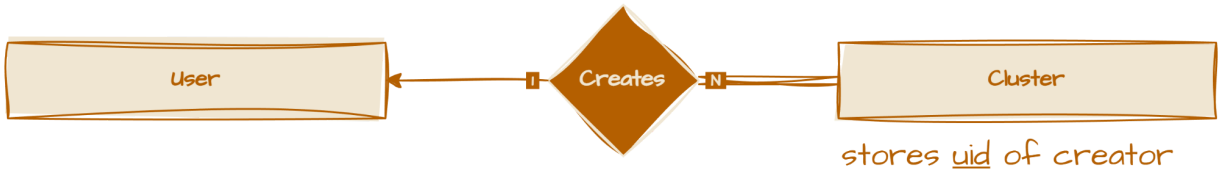
(*) some information that is enforced like **MIN, MAX, PRIMARY KEY** and **NOT NULL** are presented in the relational **schema** section after this section.

Moderators Relation

- **Multiple** Moderators (User) can Moderate **One** Cluster.
- **One** User (Moderator) can Moderate **Multiple** Clusters.
- **N** : **N** - (requires new table) with **UID** and **CID**.

Creates (Cluster) Relation

- **Multiple** Users (Verified) can create **Multiple** Clusters.
- **One** Cluster can be created by only **One** Verified (User).
- **1** : **N** - (no need for a new table). Updated Cluster to store the UID of the creator.



Joins Relation

- **Multiple** Users can join **One** Cluster.
- **One** User can join **Multiple** Clusters.
- **N** : **N** - (requires new table) with **UID** and **CID**.

Enforces Relation

- **One** Cluster can enforce **Multiple** Rules.
- **One** Rule can be enforced on **One** Clusters.
- **1** : **N** - (no need for a new table). Updated Rules to store the CID of the cluster.

Contains Relation

- **One** Cluster can contain **Multiple** Posts.
- **One** Post can be present in **One** Clusters.
- **1** : **N** - (no need for a new table). Updated Post to store the CID of the cluster.

Create (Comment) Relation

- **One** User can create **Multiple** Comments.
- **One** Comment can be created by **One** User.
- **1** : **N** - (no need for a new table). Updated Comment to store the UID of the User.

Create (Post) Relation

- **One** User can create **Multiple** Posts.
- **One** Post can be created by **One** User.
- **1** : **N** - (no need for a new table). Updated Post to store the UID of the User.

Reacts (Comment) Relation

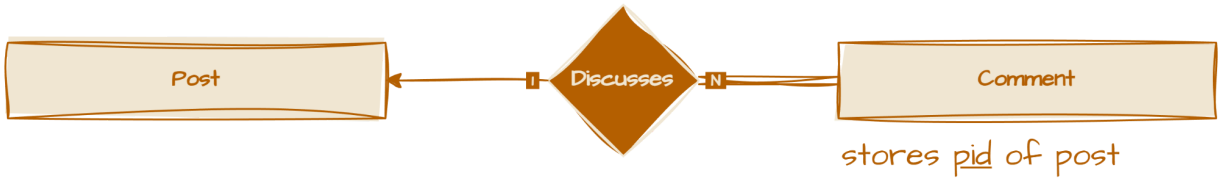
- **One** User can react to **Multiple** Comments.
- **One** Comment can be reacted to by **Multiple** Users.
- **N** : **N** - (requires new table) with **UID** and **CID**.

Reacts (Post) Relation

- **One** User can react to **Multiple** Posts.
- **One** Post can be reacted to by **Multiple** Users.
- **N** : **N** - (requires new table) with **UID** and **PID**.

Discusses (Post) Relation

- **One** Post can be discussed by **Multiple** Comments.
- **One** Comment can discuss **One** Post.
- **1** : **N** - (no need for a new table). Updated Comment to store the PID of the Post.



Discusses (Comment) Relation

- **One** Comment can be discussed by **Multiple** Comments.
- **One** Comment can discuss **One** Comment.
- **1** : **N** - (no need for a new table). Updated comment to store the parent CID of the Parent Comment that children discuss.

References Relation

- **One** Post can be referenced by **Multiple** Post.
- **One** Post can discuss **One** Post.
- **1** : **N** - (no need for a new table). Updated Window (Post's subtype) to store the original PID of the Post that is being referenced.

(*) some information that is enforced like **MIN, MAX, PRIMARY KEY** and **NOT NULL** are presented in the relational **schema** section after this section.

RELATIONAL SCHEMAS

This section exists to explain schemas and constraints of each entity/relation in case they were not clear in previously shown diagrams.

```
User (
   uid                PRIMARY KEY,
   name,
   email        UNIQUE,   -- Can be NULL if phone is used, but usually one is required
   phone        UNIQUE,   -- Can be NULL if email is used
   password_hash,
   role,              -- Enum: 'GUEST', 'MEMBER', 'VERIFIED', 'ADMIN'
   is_verified,       -- Boolean
   created_at,
   last_active        NULL,
   profile_image      NULL,
   location           NULL
)
```

```
Cluster (
   cid                PRIMARY KEY,
   name,
   description        NULL,
   is_private,        -- Boolean
   category           NULL,
   member_count       DEFAULT 0,
   creator_uid        FOREIGN KEY REFERENCES User(uid),
   created_at,
   profile_icon       NULL,
   tags               NULL
)
```

```
Cluster_Moderator (
    cid                 FOREIGN KEY REFERENCES Cluster(cid),
    uid                 FOREIGN KEY REFERENCES User(uid),
    assigned_at,
    PRIMARY KEY (cid, uid)
)
```

```
Cluster_Member (
    cid                 FOREIGN KEY REFERENCES Cluster(cid),
    uid                 FOREIGN KEY REFERENCES User(uid),
    joined_at,
    role,               -- Enum: 'MEMBER', 'MODERATOR'
    PRIMARY KEY (cid, uid)
)
```

```
Post_Reaction (
    pid                 FOREIGN KEY REFERENCES Post(pid),
    uid                 FOREIGN KEY REFERENCES User(uid),
    reaction_type,      -- Enum: 'LIKE', 'DISLIKE', etc.
    timestamp,
    PRIMARY KEY (pid, uid)
)
```

```
Comment_Reaction (
    mid                 FOREIGN KEY REFERENCES Comment(mid),
    uid                 FOREIGN KEY REFERENCES User(uid),
    reaction_type,
    timestamp,
    PRIMARY KEY (mid, uid)
)
```

```
Post (
    pid                 PRIMARY KEY,
    content,
    type,               -- Enum: 'TEXT', 'LINK', 'WINDOW'
    created_at,
    update_at           NULL,
    uid                 FOREIGN KEY REFERENCES User(uid),
    cid                 FOREIGN KEY REFERENCES Cluster(cid),
    tags                NULL
)
```

```
Window (
  wid               PRIMARY KEY REFERENCES Post(pid),
  origin_pid        FOREIGN KEY REFERENCES Post(pid),
  shared_by_uid     FOREIGN KEY REFERENCES User(uid),
  shared_into_cid   FOREIGN KEY REFERENCES Cluster(cid),
  created_at
)
```

```
Comment (
  mid               PRIMARY KEY,
  content,
  created_at,
  update_at         NULL,
  uid               FOREIGN KEY REFERENCES User(uid),
  parent_pid        NULL, -- FOREIGN KEY REFERENCES Post(pid) (If root comment)
  parent_mid        NULL  -- FOREIGN KEY REFERENCES Comment(mid) (If reply)
)
```

```
Megaphone (
  pid               PRIMARY KEY REFERENCES Post(pid),
  start_time,
  end_time,
  type,             -- Enum: 'ANNOUNCEMENT', 'POLL', 'EVENT'
  is_active,        -- Boolean
  subscriber_count  DEFAULT 0
)
```

```
Rules (
  rid               PRIMARY KEY,
  name,
  pattern,          -- Regex string
  action,           -- Enum: 'BLOCK', 'FLAG'
  description       NULL,
  cid               FOREIGN KEY REFERENCES Cluster(cid)
)
```

(*) In provided **Relation Schemas** All parameters that are not marked as **NULL** are **NOT NULL** as only those attributes that can have **NULL** values are mentioned.

CONTRIBUTORS

Mohd Zaid, Nikhil Agrawal & **Jai Ahuja**.