**Introduction**

My database is loosely based on a blogging and digital publishing website *Medium.com*. The website describes itself as an extensive publishing platform where over 100 million readers come to find insightful and dynamic thinking. The idea behind Medium.com is to be a place or a so-called *medium* where ideas take shape, take off, and spark powerful conversations. Both an expert and a blogging novice can [share their ideas](https://help.medium.com/hc/en-us/articles/360006362473-Medium-s-Curation-Guidelines-everything-writers-need-to-know) and deepen their understanding of the world.

The platform is open to anyone, and no topic is off limits as long as it is within their [curation guidelines](https://help.medium.com/hc/en-us/articles/360006362473-Medium-s-Curation-Guidelines-everything-writers-need-to-know). However, if one wants to start publishing their own posts on the website, one needs to sign up for a free account to do so. Medium.com also has an option of paid membership either with monthly or yearly contributions. Paid membership allows a member to have an unlimited article viewing privileges with no advertisements and directly support writers who are part of [the Partner Program](https://help.medium.com/hc/en-us/articles/115011694187-Join-the-Partner-Program).

The database of the platform would be a lot more complex than the one I am employing. The principle behind my database architecture is the fact that any member of the platform can become an author of a post. Before publishing, an author selects publication name (if it has one) where the story is going to be featured in. The platform also has employed editors who moderate content and make sure members follow the guidelines.

As a disclaimer, I have never published anything on Medium.com, so they way publishing works is only my understanding from what I gathered reading the website.

**Mission Statement**

The purpose of the Medium.com website database is to maintain the data we generate, and to supply information that supports the publishing services we provide to our writing members and browsing services we provide to our reading audience.

**Mission Objectives**

* Maintain complete membership information.
* Keep track of all posts published.
* Maintain information on our editors.
* Maintain information on publications.

More detailed:

* A post can be published only in one publication if any and by one member.
* A publication can have many editors.
* An editor can have no more than 10 publications.

**Conceptual Design**

Diagram

Description automatically generated

**Logical Design**

Table

Description automatically generated

No default or Range Values specified for this database, so I am not including these columns.

**Business Rule**

|  |
| --- |
| **Rule Information**  **Rule Statement**: Each **Editor** can have no more than **10** publications.  **Constraint**: A single record in Editors table can be associated with up to 10 records in EditorPublications table.  **Type**: Database Oriented **Category**: Field Specific  Application Oriented Relationship Specific |
| **Structure Affect:**  Field Names:  Table Names: Editors, EditorPublications |
| **Field Elements affected:**  Data Type Character Support  Data length Null Support  Decimal Points Range of Values  Key Type Default Value |
| **Relationship Characteristics Affected**  Degree of Participation Type of Relationship  Deletion Rule |
| **Actions Taken**  **The degree of participation was changed to (1, 10).** |

Trigger:

BEGIN

DECLARE dummy, baddata INT;

SET baddata = 0;

IF COUNT(NEW.EditorPublicationsID) > 10 THEN

SET baddata = 1;

END IF;

IF COUNT(NEW.EditorPublicationsID) < 1 THEN

SET baddata = 1;

END IF;

IF baddata = 1 THEN

SELECT concat('Cannot Insert This Because Value', COUNT(NEW.EditorPublicationsID), 'exceeds the allowed number of publications per editor')

INTO dummy FROM information\_schema.tables;

END IF;

**Views**

Member Posts: Member Name, Number of posts published.

Diagram

Description automatically generated

CREATE VIEW MemberPosts AS

SELECT m.MemName, COUNT(p.PostID) AS Posts

FROM Members AS m

INNER JOIN Posts AS p ON p.MemberID = m.MemberID

GROUP BY MemName

ORDER BY Posts desc;

Member Details: Member Name, Member Description, Member Email.

Text, chat or text message

Description automatically generated

CREATE VIEW MemberDetails AS

SELECT MemName AS Name,

MemDescription AS Description,

MemEmail AS Email

FROM Members;

Editors per Publication: Publication Name, Editor First Name and Editor Last Name as Editor Name, Editor Email address.

Diagram

Description automatically generated

CREATE VIEW EditorsPerPublication AS

SELECT p.PublicationName, CONCAT(e.EditorLastName,', ', e.EditorFirstName) AS EditorName, e.EditorEmail

FROM Publications AS p

INNER JOIN EditorPublications AS ep ON p.PublicationID = ep.PublicationID

INNER JOIN Editors AS e ON e.EditorID = ep.EditorID;

CREATE TABLE Memberships

(

MembershipID INT NOT NULL,

MembershipName VARCHAR(20) NOT NULL,

MembershipDescription VARCHAR(150) NOT NULL,

PRIMARY KEY (MembershipID)

)

ENGINE=InnoDB;

CREATE INDEX MembershipName ON Memberships(MembershipName);

CREATE INDEX MembershipDescription ON Memberships(MembershipDescription);

CREATE TABLE Members

(

MemberID INT NOT NULL,

MemName VARCHAR(50) NOT NULL,

MemLogin VARCHAR(20) NOT NULL,

MemPassword VARCHAR(20) NOT NULL,

MemDescription tinytext,

MemEmail VARCHAR(30) NOT NULL,

MembershipID INT NOT NULL,

SignUpDate DATE NOT NULL,

PRIMARY KEY(MemberID),

FOREIGN KEY (MembershipID) REFERENCES Memberships(MembershipID)

ON DELETE CASCADE

ON UPDATE CASCADE

)

ENGINE=InnoDB;

CREATE INDEX MemName ON Members(MemName);

CREATE INDEX MemEmail ON Members(MemEmail);

CREATE INDEX MembershipID ON Members(MembershipID);

CREATE INDEX SignUpDate ON Members(SignUpDate);

CREATE TABLE Editors

(

EditorID INT NOT NULL,

EditorLastName VARCHAR(25) NOT NULL,

EditorFirstName VARCHAR(25) NOT NULL,

EditorMiddleName VARCHAR(25),

EditorEmail VARCHAR(50) NOT NULL,

EditorPhoneNum CHAR(12),

EditorSSN CHAR(11) NOT NULL,

PRIMARY KEY (EditorID)

) ENGINE=InnoDB;

CREATE INDEX EditorLastName ON Editors (EditorLastName);

CREATE INDEX EditorFirstName ON Editors (EditorFirstName);

CREATE INDEX EditorEmail ON Editors (EditorEmail);

CREATE INDEX EditorPhoneNum ON Editors (EditorPhoneNum);

CREATE TABLE Publications

(

PublicationID INT NOT NULL,

PublicationName VARCHAR(25) NOT NULL,

PublicationDescription tinytext,

PRIMARY KEY (PublicationID)

) ENGINE=InnoDB;

CREATE INDEX PublicationName ON Publications (PublicationName);

CREATE TABLE EditorPublications

(

EditorPublicationsID INT NOT NULL,

EditorID INT NOT NULL,

PublicationID INT NOT NULL,

PRIMARY KEY (EditorPublicationsID),

FOREIGN KEY (EditorID) REFERENCES Editors(EditorID),

FOREIGN KEY (PublicationID) REFERENCES Publications(PublicationID)

ON DELETE CASCADE

ON UPDATE CASCADE

) ENGINE=InnoDB;

CREATE INDEX EditorID on EditorPublications (EditorID);

CREATE INDEX PublicationID on EditorPublications (PublicationID);

CREATE TABLE Posts

(

PostID INT NOT NULL,

PostTitle VARCHAR(100) NOT NULL,

PostDescription tinytext,

PostBody text NOT NULL,

MemberID INT NOT NULL,

PublicationID INT,

FOREIGN KEY (MemberID) REFERENCES Members(MemberID),

FOREIGN KEY (PublicationID) REFERENCES Publications(PublicationID)

ON DELETE CASCADE

ON UPDATE CASCADE

) ENGINE=InnoDB;

CREATE INDEX PostTitle ON Posts (PostTitle);

CREATE INDEX MemberID ON Posts (MemberID);

**SQL Queries:**

1. **I want to know the information on Editors (their names and email addresses) and how many publications they review. I would like to limit my search to the editors who lead 5 and more publications at once.**

SELECT COUNT(p.PublicationID) AS Count, CONCAT(e.EditorLastName,', ', e.EditorFirstName) AS EditorName, e.EditorEmail

FROM Publications AS p

INNER JOIN EditorPublications AS ep ON p.PublicationID = ep.PublicationID

INNER JOIN Editors AS e ON e.EditorID = ep.EditorID

Group By EditorName

Having Count >=5

Order By Count desc;

Graphical user interface, text, application, email

Description automatically generated

1. **I need information on posts, publications these posts appear in and names and description of members who wrote them.**

SELECT m.MemName, m.MemDescription, p.PostTitle, pb.PublicationName

FROM Members AS m

INNER JOIN Posts AS p ON m.MemberID = p.MemberID

INNER JOIN Publications AS pb ON pb.PublicationID = p.PublicationID

ORDER BY m.MemName;

Graphical user interface, text, application, email

Description automatically generated

1. **I want to know what publication has the most posts.**

SELECT Count(p.PostID) AS 'Number of Posts', pb.PublicationName, pb.PublicationDescription

From Posts AS p

INNER JOIN Publications AS pb ON p.PublicationID = pb.PublicationID;

Graphical user interface, text, application

Description automatically generated

1. **Who wrote the greatest number of posts?**

SELECT Count(p.PostID) AS 'Number of Posts', m.MemName, m.MemDescription

From Posts AS p

INNER JOIN Members AS m ON p.MemberID = m.MemberID;

A screenshot of a computer

Description automatically generated

1. **Names and Emails of members who pay yearly subscription fee.**

SELECT m.MemName, m.MemEmail, m.SignUpDate, mb.MembershipName NOT IN (select MembershipID from

Memberships where MembershipName = 'free') AS PaidMembership

FROM Members AS m

INNER JOIN Memberships AS mb ON m.MembershipID = mb.MembershipID

ORDER BY m.SignUpDate desc;

Graphical user interface, text, application, email

Description automatically generated