

Chapter 4

Database Design

4.1 Introduction

- **Database:** A Database is a collection of related data, which can be of any size and complexity. By using the concept of a database, we can easily store and retrieve the data. The primary purpose of a database is to provide the information, which utilizes it with the system's information according to its requirements.
- **Database Design:** Database design is done before building it to meet end-users' needs within a given information-system that the database is intended to support. The database design defines the needed data and data structures that such a database comprises


The database is physically implemented using MySQL.

The database for **LNMIIT Connect** is organized into 9 tables:


- users
- messages
- about
- comments
- notifications
- friendrequests
- likes
- posts
- trends

Each entity can be described as follows along with its attributes:


- users

Column	Type	Description
id 	int(11)	User id
first_name	varchar(25)	Full name of user
last_name	varchar(25)	batch of user
username	varchar(100)	Username of the user
email	varchar(100)	E-mail of the user
password	varchar(255)	Password of the user in md5
signup_date	date	Sign up date of the user
profile_pic	varchar(255)	URL for the profile pic
num_post	int(11)	Number of post made by the user
num_like	int(11)	Number of the likes received by the user
user_closed	varchar(3)	if the account is active or not
friend_array	text	list of friends of the user


- messages

Column	Type	Description
id 	int(11)	Message id
user_to	varchar(50)	To which user the message is send
user_from	varchar(50)	from which user the message is send
body	text	body of the message
deleted	varchar(3)	if the message is deleted
date	datetime	date and time of message
opened	varchar(3)	if the message is opened
viewed	varchar(3)	if the message is viewed


- about

Column	Type	Description
username 	varchar(100)	Username of the user
about	varchar(200)	Information about the user


- comments

Column	Type	Description
id 	int(11)	Comment id for the comment
post_body	text	body of the comment
posted_by	varchar(60)	Username of the user who posted comments
posted_to	varchar(60)	Username of the post on which the comment is posted
date_added	datetime	date of the comment
removed	varchar(3)	if the comment is removed
post_id	int(11)	id of the post on which the comment is posted


- notifications

Column	Type	Description
id 	int(11)	Notifications id
user_to	varchar(50)	To which user the notifications is send
user_from	varchar(50)	from which user the notifications is send
message	text	body of the notification
link	varchar(100)	link of the notification(e.g post link)
datetime	datetime	date and time of notification
opened	varchar(3)	if the notification is opened
viewed	varchar(3)	if the notification is viewed


- friend-requests

Column	Type	Description
id 	int(11)	table key
user_to	varchar(50)	user to whom friend request is send
user_from	varchar(50)	user who has send the friend request

- likes

Column	Type	Description
id 	int(11)	table key
username	varchar(60)	Username of the user who liked the post
post_id	int(11)	post id of the post which is liked

- posts

Column	Type	Description
id 	int(11)	Post id
body	text	Body of the post
added_by	varchar(60)	Username of the user who added post
user_to	varchar(60)	Username of the user to whom the is added post
date_added	datetime	date and time of the post
user_closed	varchar(3)	if the user if active or not
deleted	varchar(3)	if the post is deleted or not
likes	int(11)	Number of likes on the post
image	varchar(60)	image link of the post

- trends

Column	Type	Description
title	varchar(50)	word for which we store the count
hits	int(11)	Number of hits of a word

4.2 Database Schema Design

MySQL Workbench simplifies database design and maintenance, automates time-consuming and error-prone tasks, and improves communication among DBA and developer teams. It enables data architects to visualize requirements, communicate with stakeholders, and resolve design issues before a significant investment of time and resources is made. It helps model-driven database design, which is the most efficient methodology for creating valid and well-performing databases while providing the flexibility to respond to evolving business requirements. Model and Schema Validation utilities enforce best practice standards for data modeling, also enforce MySQL-specific physical design standards, so no mistakes are made when building new ER diagrams or generating physical MySQL databases.

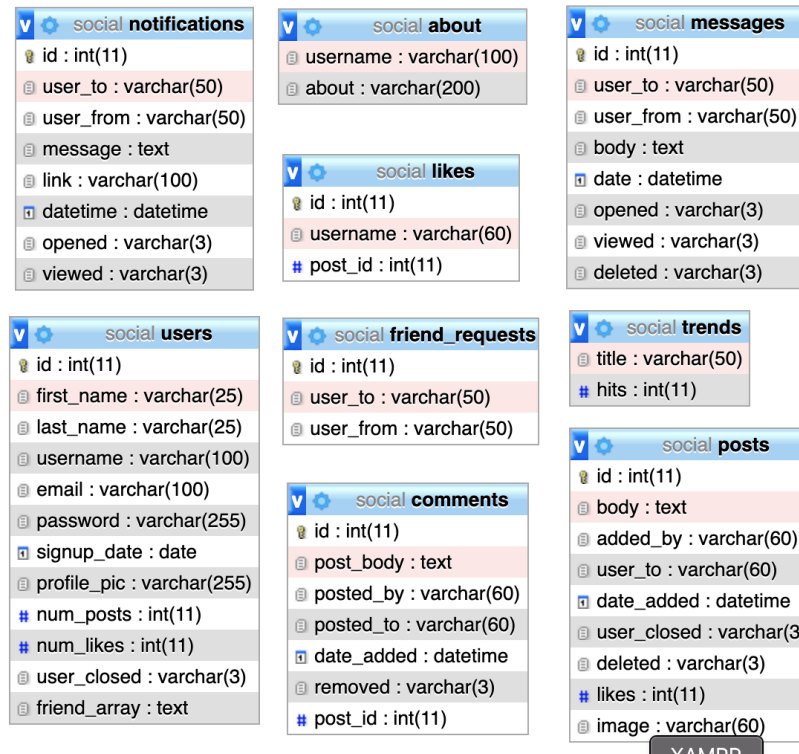


Figure 4.1 Database Schema Design of LNMIIT Connect