Course Project : First Part

1. *Basic Tables*

a. **User**(user_id, email_id, name, password, dob, gender, city, country, about_me, tagline, interests(topic_id), ProfilePictureURL, University, ComapnyName)

b. **Follows**(user_id, follow_id, flag, timestamp)

c. **Chat**(sender(user_id), receiver(user_id), message_id, message, isSeen, timestamp)

d. **Question**(question_id, questionBy(user_id), question, timestamp, topic_id, isAnonymous, viewers(user_id), askedBy(user_id), askedTo(user_id), response)

e. **Topic**(topic_id, topic_name)

f. **Answer**(answer_id, question_id, answerBy(user_id), answer, timestamp, viewers(user_id), bookmarkedBy(user_id))

g. **Comment**(question_id, answer_id, comment_id, comment, parent_id(comment_id), commentBy(user_id), timestamp)

h. **Vote**(question_id, answer_id, comments_id, voteBy(user_id), vote)

i. **Notification**(notification_id, notificationTo(user_id), url, string, isRead, timestamp)

2. *Assumptions*

a. { interests(topic_id), topic_id, viewers(user_id), askedBy(user_id), askedTo(user_id), bookmarkedBy(user_id) } are multi-valued attributes in the basic table.

b. “Flag” attribute in Follows table depicts whether the follow_id represents user_id, topic_id or question_id.

c. Question_id remains unique over the globe while answer_id is unique under one question_id and similarly, comment_id is unique under one answer_id. This is done to make the database scalable with large amount of data.

d. Parent_id attribute in Comment table has default value -1 to depict the 1st level comments while any positive value depicts lower level comments with the parent comment as comment with given comment_id.

e. Comment_id attribute in Vote table has default value -1 to depict the vote on an answer. Any value > 0 depicts the vote is for the comment with given comment_id.

3. *Functional Dependencies and Prime/Non-Prime Attributes*

a. For (a) in Basic Tables
user_id -> email_id
email_id -> user_id
user_id -> name, password, dob, gender, city, country, about_me, tagline, topic_id, profilePicURL, university, companyName
Prime Attributes : user_id, email_id Primary Key : user_id
Non-Prime Attributes : {rest all}

b. For (b) in Basic Tables
user_id, follow_id, flag -> timestamp
Prime Attributes : user_id, follow_id, flag Primary Key : user_id, follow_id, flag
Non-Prime Attributes : timestamp

c. For (c) in Basic Tables
sender, receiver, message_id -> message, isSeen, timestamp
Prime Attributes : sender, receiver, message_id
Primary Key : sender, receiver, message_id
Non-Prime Attributes : message, isSeen, timestamp

d. For (d) in Basic Tables
question_id -> questionBy, question, timestamp, isAnonymous, viewers, topic_id
question_id, askedBy -> askedTo, response,
Prime Attributes : question_id, askedBy Primary Key : question_id, askedBy
Non-Prime Attributes : {rest all}

e. For (e) in Basic Tables
topic_id -> topic_name
Prime Attributes : topic_id Primary Key : topic_id
Non-Prime Attributes : topic_name

f. For (f) in Basic Tables
answer_id, question_id -> answerBy, answer, timestamp, viewer, bookmarkedBy
answerBy, question_id -> answer-id
Prime Attributes : answer_id, question_id, answerBy
Primary Key : answer_id, question_id
Non-Prime Attributes : answer, timestamp, viewer, bookmarkedBy

g. For (g) in Basic Tables
answer_id, question_id, comment_id -> comment, parent_id, commentBy,

timestam
Prime Attributes : answer_id, question_id, comment_id
Primary Key : answer_id, question_id, comment_id
Non-Prime Attributes : comment, parent_id, commentBy, timestamp

h. For (h) in Basic Tables
answer_id, question_id, comment_id, voteBy -> vote
Prime Attributes : answer_id, question_id, comment_id, voteBy
Primary Key : answer_id, question_id, comment_id, voteBy
Non-Prime Attributes : vote

i. For (i) in Basic Tables
notification_id, notificationTo -> url, string, isRead, timestamp
Prime Attributes : notification_id, notificationTo
Primary Key : notification_id, notificationTo
Non-Prime Attributes : url, string, isRead, timestamp

4. Minimal Cover

a. For (a) in Basic Tables
user_id *-> email_id*
email_id *-> user_id*
user_id *-> name, password, dob, gender, city, country, about_me, tagline, topic_id, profilePicURL, university, companyName*

Closures:
{***user_id***}$^+$ = {user_id, email_id, name, password, dob, gender, city, country, about_me, tagline, topic_id, profilePicURL, university, companyName}
{***email_id***}$^+$ = {email_id, user_id, name, password, dob, gender, city, country, about_me, tagline, topic_id, profilePicURL, university, companyName}

No redundant Functional Dependencies, hence the above mentioned set of Functional Dependencies is the Minimal Cover.

b. For (b) in Basic Tables
user_id, follow_id, flag *-> timestamp*

Closures:
{***user_id, follow_id, flag***}$^+$ = {user_id, follow_id, flag, timestamp}

No redundant Functional Dependencies, hence the above mentioned set of Functional Dependencies is the Minimal Cover.

c. For (c) in Basic Tables
sender, receiver, message_id *-> message, isSeen, timestamp*

Closures:
{***sender, receiver, message_id***}$^+$ = {sender, receiver, message_id, message, isSeen, timestamp}

No redundant Functional Dependencies, hence the above mentioned set of Functional Dependencies is the Minimal Cover.

d. For (d) in Basic Tables
question_id *-> questionBy, question, timestamp, isAnonymous, viewers, topic_id*
question_id, askedBy *-> askedTo, response*

Closures:
{***question_id***}$^+$ = {question_id, questionBy, question, timestamp, isAnonymous, viewers, topic_id}
{***question_id, askedBy***}$^+$ = {question_id, askedBy, askedTo, response}

No redundant Functional Dependencies, hence the above mentioned set of Functional Dependencies is the Minimal Cover.

e. For (e) in Basic Tables
topic_id *-> topic_name*

Closures:
{***topic_id***}$^+$ = {topic_id, topic_name}

No redundant Functional Dependencies, hence the above mentioned set of Functional Dependencies is the Minimal Cover.

f. For (f) in Basic Tables
answer_id, question_id *-> answerBy, answer, timestamp, viewer, bookmarkedBy*
answerBy, question_id *-> answer_id*

<u>Closures:</u>
{***answer_id, question_id***}$^{+}$ = {answer_id, question_id, answerBy, answer, timestamp, viewer, bookmarkedBy}
{***answerBy, question_id***}$^{+}$ = {answerBy, question_id, answer_id}

No redundant Functional Dependencies, hence the above mentioned set of Functional Dependencies is the Minimal Cover.

g. For (g) in Basic Tables
answer_id, question_id, comment_id *-> comment, parent_id, commentBy, timestamp*

<u>Closures:</u>
{***answer_id, question_id, comment_id***}$^{+}$ = {answer_id, question_id, comment_id, comment, parent_id, commentBy, timestamp}

No redundant Functional Dependencies, hence the above mentioned set of Functional Dependencies is the Minimal Cover.

h. For (h) in Basic Tables
answer_id, question_id, comment_id, voteBy *-> vote*

<u>Closures:</u>
{***answer_id, question_id, comment_id, voteBy***}$^{+}$ = {answer_id, question_id, comment_id, voteBy, vote}

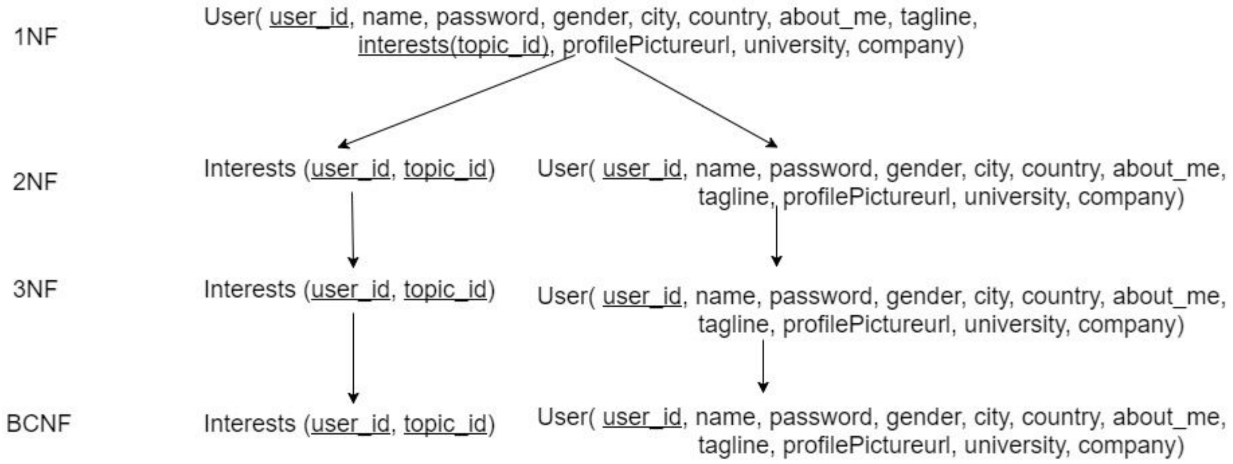
No redundant Functional Dependencies, hence the above mentioned set of Functional Dependencies is the Minimal Cover.

i. For (i) in Basic Tables
notification_id, notificationTo *-> url, string, isRead, timestamp*

<u>Closures:</u>
{***notification_id, notificationTo***}$^{+}$ = {notification_id, notificationTo, url, string, isRead, timestamp}
No redundant Functional Dependencies, hence the above mentioned set of Functional Dependencies is the Minimal Cover.

5. Normalization

a. User table



1NF

User(user_id, name, password, gender, city, country, about_me, tagline, interests(topic_id), profilePictureurl, university, company)

2NF

Interests (user_id, topic_id)

User(user_id, name, password, gender, city, country, about_me, tagline, profilePictureurl, university, company)

3NF

Interests (user_id, topic_id)

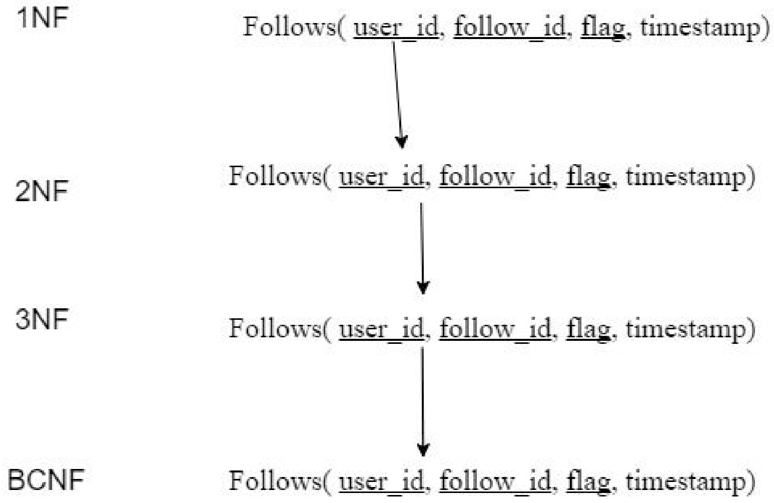
User(user_id, name, password, gender, city, country, about_me, tagline, profilePictureurl, university, company)

BCNF

Interests (user_id, topic_id)

User(user_id, name, password, gender, city, country, about_me, tagline, profilePictureurl, university, company)

b. Follows table



1NF

Follows(user_id, follow_id, flag, timestamp)

2NF

Follows(user_id, follow_id, flag, timestamp)

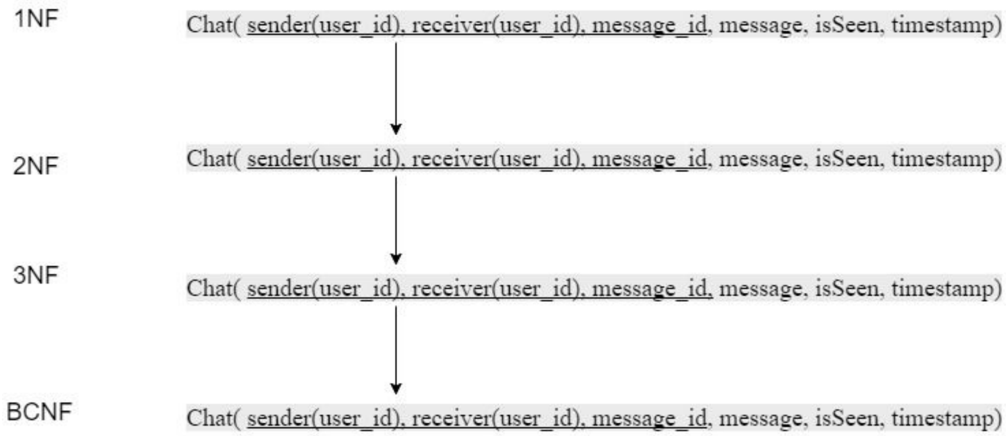
3NF

Follows(user_id, follow_id, flag, timestamp)

BCNF

Follows(user_id, follow_id, flag, timestamp)

c. Chat table



1NF

Chat(sender(user_id), receiver(user_id), message_id, message, isSeen, timestamp)

2NF

Chat(sender(user_id), receiver(user_id), message_id, message, isSeen, timestamp)

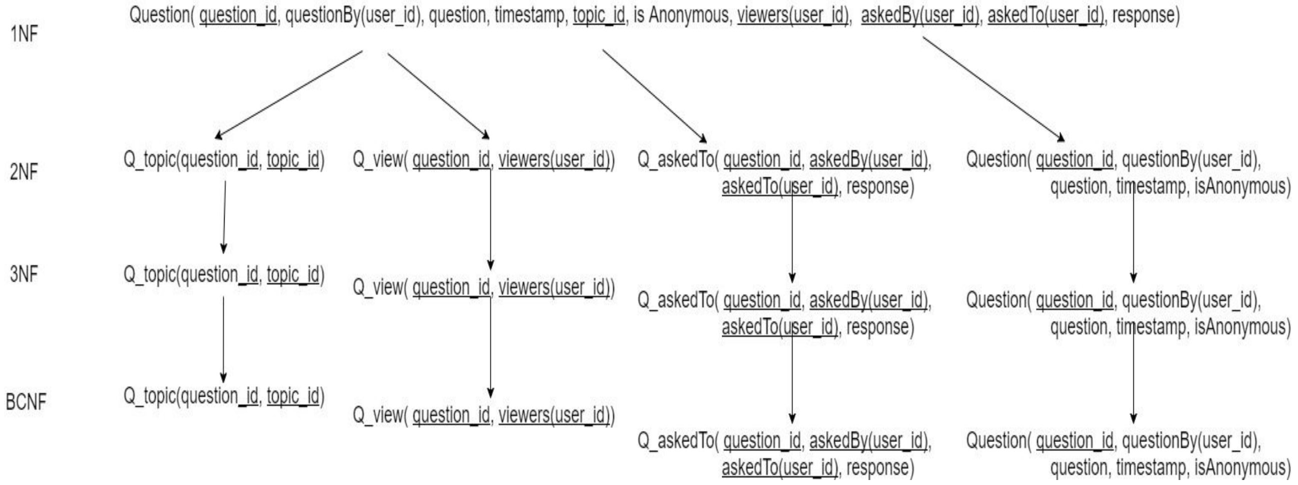
3NF

Chat(sender(user_id), receiver(user_id), message_id, message, isSeen, timestamp)

BCNF

Chat(sender(user_id), receiver(user_id), message_id, message, isSeen, timestamp)

d. Question table



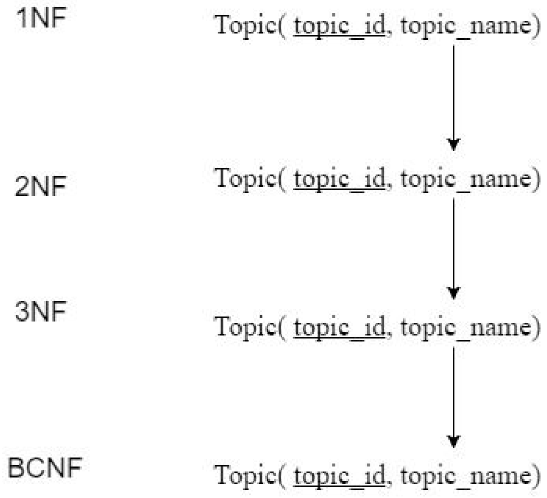
1NF Question(question_id, questionBy(user_id), question, timestamp, topic_id, is Anonymous, viewers(user_id), askedBy(user_id), askedTo(user_id), response)

2NF Q_topic(question_id, topic_id) Q_view(question_id, viewers(user_id)) Q_askedTo(question_id, askedBy(user_id), askedTo(user_id), response) Question(question_id, questionBy(user_id), question, timestamp, isAnonymous)

3NF Q_topic(question_id, topic_id) Q_view(question_id, viewers(user_id)) Q_askedTo(question_id, askedBy(user_id), askedTo(user_id), response) Question(question_id, questionBy(user_id), question, timestamp, isAnonymous)

BCNF Q_topic(question_id, topic_id) Q_view(question_id, viewers(user_id)) Q_askedTo(question_id, askedBy(user_id), askedTo(user_id), response) Question(question_id, questionBy(user_id), question, timestamp, isAnonymous)

e. Topic table



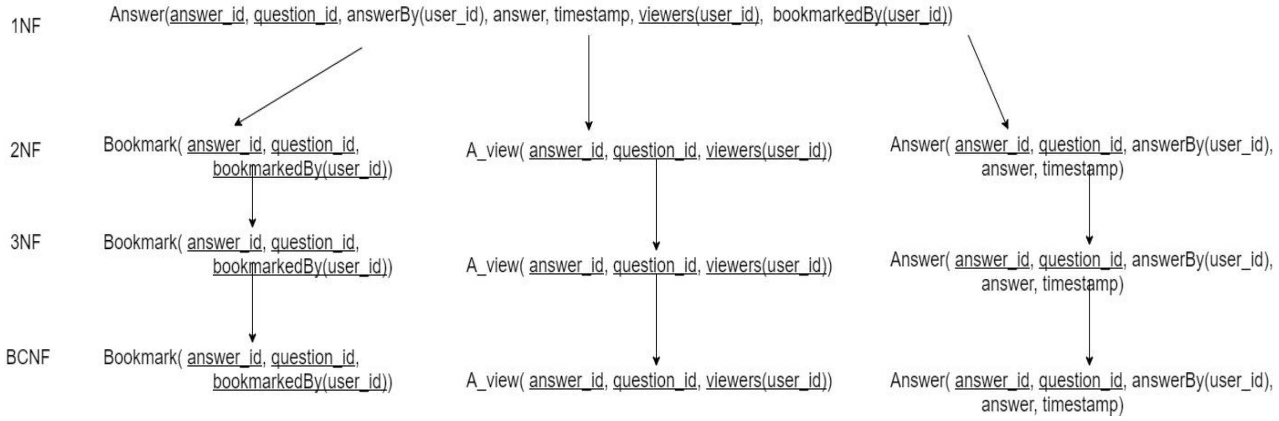
1NF Topic(topic_id, topic_name)

2NF Topic(topic_id, topic_name)

3NF Topic(topic_id, topic_name)

BCNF Topic(topic_id, topic_name)

f. Answer table



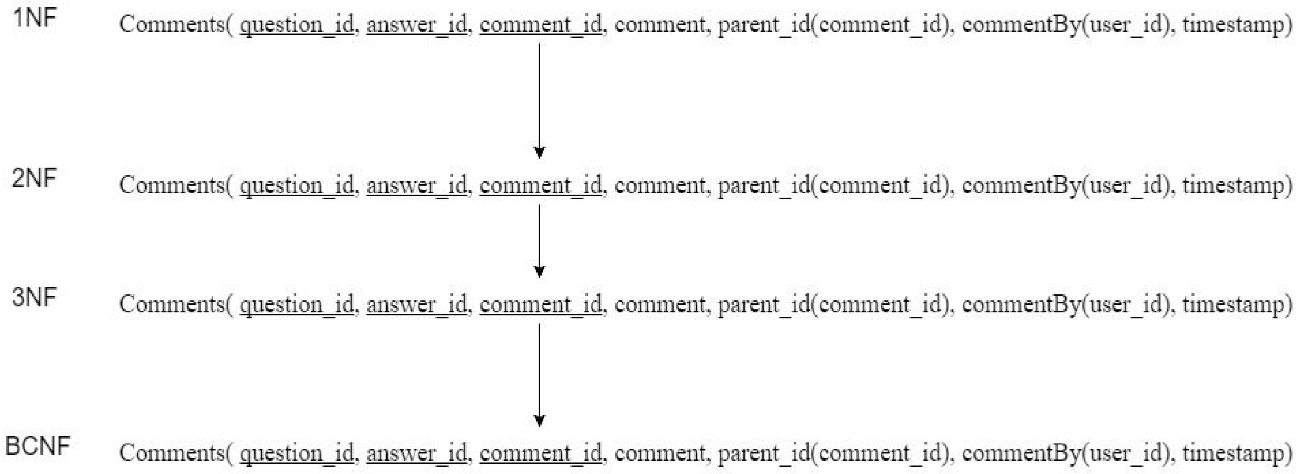
1NF Answer(answer_id, question_id, answerBy(user_id), answer, timestamp, viewers(user_id), bookmarkedBy(user_id))

2NF Bookmark(answer_id, question_id, bookmarkedBy(user_id)) A_view(answer_id, question_id, viewers(user_id)) Answer(answer_id, question_id, answerBy(user_id), answer, timestamp)

3NF Bookmark(answer_id, question_id, bookmarkedBy(user_id)) A_view(answer_id, question_id, viewers(user_id)) Answer(answer_id, question_id, answerBy(user_id), answer, timestamp)

BCNF Bookmark(answer_id, question_id, bookmarkedBy(user_id)) A_view(answer_id, question_id, viewers(user_id)) Answer(answer_id, question_id, answerBy(user_id), answer, timestamp)

g. Comment table



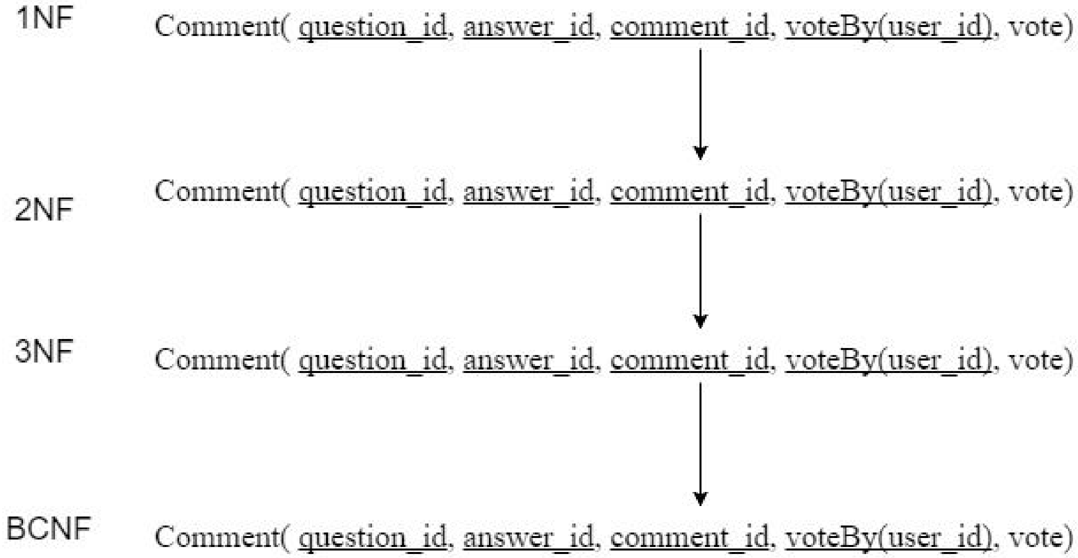
1NF Comments(question_id, answer_id, comment_id, comment, parent_id(comment_id), commentBy(user_id), timestamp)

2NF Comments(question_id, answer_id, comment_id, comment, parent_id(comment_id), commentBy(user_id), timestamp)

3NF Comments(question_id, answer_id, comment_id, comment, parent_id(comment_id), commentBy(user_id), timestamp)

BCNF Comments(question_id, answer_id, comment_id, comment, parent_id(comment_id), commentBy(user_id), timestamp)

h. Vote table



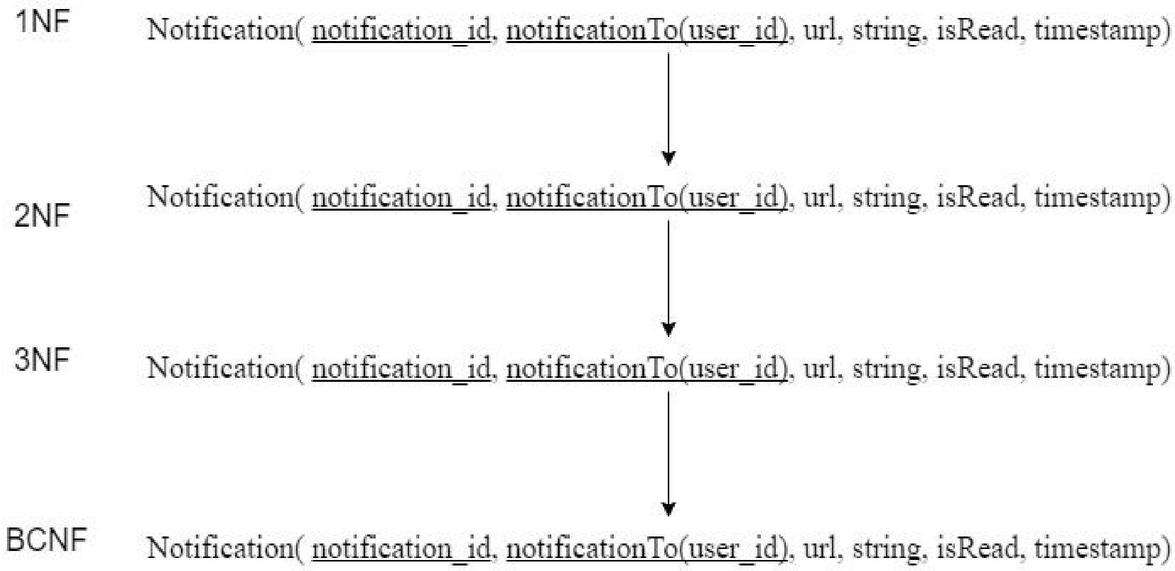
1NF Comment(question_id, answer_id, comment_id, voteBy(user_id), vote)

2NF Comment(question_id, answer_id, comment_id, voteBy(user_id), vote)

3NF Comment(question_id, answer_id, comment_id, voteBy(user_id), vote)

BCNF Comment(question_id, answer_id, comment_id, voteBy(user_id), vote)

i. Notification table



1NF Notification(notification_id, notificationTo(user_id), url, string, isRead, timestamp)

2NF Notification(notification_id, notificationTo(user_id), url, string, isRead, timestamp)

3NF Notification(notification_id, notificationTo(user_id), url, string, isRead, timestamp)

BCNF Notification(notification_id, notificationTo(user_id), url, string, isRead, timestamp)