3NF DECOMP

attributes: {userid, username, password, fname, lname, isbn, title, author, category, description, year, reviewId, body, liked, created_date, follows, wishlisted}

Functional Dependencies:

userid -> username, password, fname, lname
isbn -> title, author, category, description, year
reviewId -> body, liked, created_date, userId, isbn, author
userId, isbn -> reviewId, like, body, created_date
authorId -> author
userId, isbn -> wishlisted, read, rating

Step 1:
userId -> username
userId -> password
userId -> fname
userId -> lname

isbn -> title
isbn -> author
isbn -> category
isbn -> description
isbn -> year

reviewId -> body
reviewId -> liked
reviewId -> created_date
reviewId -> userId
reviewId -> isbn
reviewId -> author

userId, isbn -> reviewId
userId, isbn -> like
userId, isbn -> body
userId, isbn -> created_date
userId, isbn -> wishlisted
userId, isbn -> read
userId, isbn -> rating

authorId -> author

Step 2:
userId -> username
userId -> password
userId -> fname
userId -> lname

isbn -> title
isbn -> author
isbn -> category
isbn -> description
isbn -> year

reviewId -> body
reviewId -> liked
reviewId -> created_date
reviewId -> userId
reviewId -> isbn
reviewId -> author

userId, isbn -> reviewId
userId+ = {userId, username, password, fname, lname} X
isbn+ = {isbn, title, author, category, description, year} X

userId, isbn -> like X
userId, isbn -> body X
userId, isbn -> created_date X
userId, isbn -> wishlisted X

authorId -> author

Step 3:
userId -> username
userId -> password
userId -> fname
userId -> lname

isbn -> title

isbn -> author
isbn -> category
isbn -> description
isbn -> year

reviewId -> body
reviewId -> liked
reviewId -> created_date
reviewId -> userId
reviewId -> isbn
reviewId -> author

userId, isbn -> reviewId
userId, isbn -> like {userId, isbn}+ = {userId, isbn, username, password, fname, lname, title, author, category, description, year, reviewId, like} X
  userId, isbn -> reviewId implies that any userId, isbn can determine the same RHS as reviewId.

userId, isbn -> body X
userId, isbn -> created_date X
userId, isbn -> wishlisted

authorId -> author

Minimal Cover:
userId -> username
userId -> password
userId -> fname
userId -> lname
isbn -> title
isbn -> author
isbn -> category
isbn -> description
isbn -> year
reviewId -> body
reviewId -> liked
reviewId -> created_date
reviewId -> userId
reviewId -> isbn
reviewId -> author

userId, isbn -> reviewId
userId, isbn -> wishlisted
userId, isbn -> read
userId, isbn -> liked
authorId -> author

merge LHS:

userId -> username, password, fname, lname
isbn -> title, author, category, description, year
reviewId -> body, liked, created_date, userId, isbn
userId, isbn -> reviewId, wishlisted, read, liked
authorId -> author

Create Tables:

user(userId, username, password, fname, lname)
book(isbn, title, authorId, category, description, year)
review(reviewId, body, liked, created_date, userId, isbn)
wishlist(userId, isbn, wishlisted, read, liked)
author(authorId, author)
CANDIDATE KEY reviewId

Final Tables:
user(userId, username, password, fname, lname)
book(isbn, title, authorId, category, description, year)
review(reviewId, body, liked, created_date, userId, isbn)
wishlist(userId, isbn, wishlisted)
author(authorId, author)

Caveats:
books can have multiple authors
Not every user is going to have a review
Putting likes, wishlist, and read in the same table will result in a sparse table

In order to simplify our queries and account for the issue of multiple authors, we chose to use the tables created from the ER diagram.