

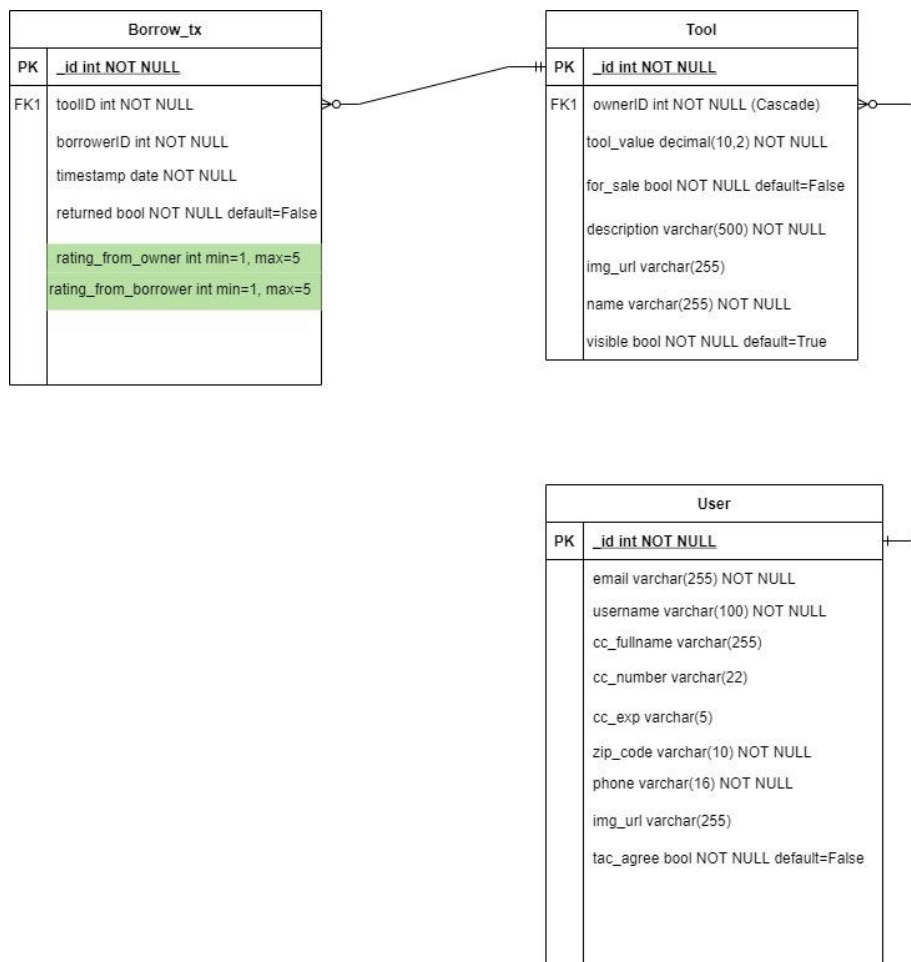
# Design Database

SWDV-691 Chris Patterson

**Your database technology choice:** After several false starts with DynamoDB, I finally decided to rethink my choice in database. I feel that MySQL will probably be a better fit for this project.

**Explanation for why this is the appropriate choice:** I settled on MySQL for several reasons. I have some experience with SQL, it's free, and it seems to integrate well with Django, which I plan on using for the back-end. I've got the database up and running in AWS and I'm able to access it with Django.

**Each table/structure you will store in your database:** Based on feedback, the database design has changed somewhat. My DB will contain three tables – User, Tool, and Borrow\_tx. I imagine that most of the "User" object will be constructed by an authenticator when a user signs up for an account. The App will handle "Tool" objects. The Borrow\_tx is a table to track tool borrowing transactions (Green items are a stretch feature).



**Explanation for each table/structure in your database that contextualizes its role in your application:** These are pretty self-explanatory, but the Users can either be the owner or borrower of the tool.

The table IDs are not named, because Django and MySQL auto-generate and auto-increment the IDs for each table. The ID is referenced by <table>.id .

.A Tool must have one owner. A tool cannot exist without an owner User. If a User owner is deleted from the DB, the Tool needs to be deleted as well (Cascade on delete).

A Tool may have 0 or 1 borrower User.

A User may own 0 to N Tools.

A Borrow transaction must have a tool and a borrower.

Some that might not be so evident –

In User:.

- Most actions in the app rely on user data. When a user is deleted, they cannot be affiliated with a currently tool (no outstanding borrow transactions).
- When a user is deleted, any tools they own will be deleted to prevent orphans.
- img\_url = a url to the users profile image. This MAY be handled by authentication, but I need to verify.
- tac\_agree = did the user agree to the terms and conditions? I put this in there to prevent users who did not agree from participating. If the TAC gets updated, then this would get flagged to false for all users until they re-read the TAC.

In Tool:

- A tool must have an owner. An owner can delete a tool, as long as there aren't any outstanding borrow transactions affiliated with the tool.
- A borrower user cannot delete their profile while they are still affiliated with a borrow transaction.
- borrowerID = the user currently borrowing the tool (could be 'none').
- visible = is the tool visible to potential borrowers (owners can 'hide' a Tool).
- for\_sale = if the owner wants to advertise the tool as being available for purchase.

In Borrow\_tx:

- A transaction is created when a user borrows a tool.
- The tool cannot be deleted until it is marked as returned in borrow\_tx.
- A borrowing user cannot delete their profile until the borrow\_tx status is updated to show "returned=true".
- A user cannot borrow a tool unless they have a CC on file.
- The green signifies a stretch feature
- "returned" is a bool that is updated from the owner on whether or not the borrowed tool has been returned.

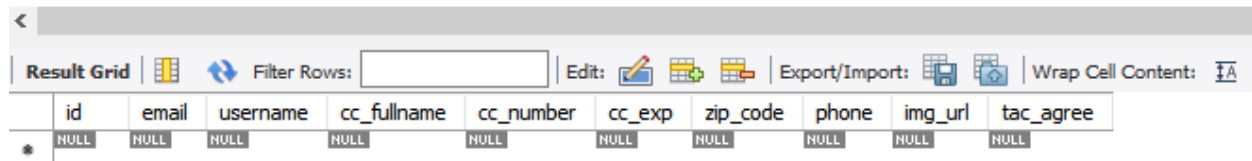
```
class User(models.Model):
    email = models.CharField(max_length=255)
    username = models.CharField(max_length=100)
    cc_fullname = models.CharField(max_length=255, null=True)
    cc_number = models.CharField(max_length=22, null=True)
    cc_exp = models.CharField(max_length=5, null=True)
    zip_code = models.CharField(max_length=10)
    phone = models.CharField(max_length=16)
    img_url = models.CharField(max_length=255, null=True)
    tac_agree = models.BooleanField(blank=False)

class Tool(models.Model):
    ownerID = models.ForeignKey(User, on_delete=models.CASCADE)
    tool_value = models.DecimalField(max_digits=10, decimal_places=2)
    for_sale = models.BooleanField()
    description = models.CharField(max_length=500)
    img_url = models.CharField(max_length=255, null=True)
    name = models.CharField(max_length=255)
    visible = models.BooleanField(default=True)

class Borrow_tx(models.Model):
    # Use "delete" code, logic, or SP to make sure borrower cannot be deleted while borrowing tool,
    # make sure that tool cannot be deleted while borrowed
    toolID = models.ForeignKey(Tool, on_delete=models.RESTRICT)
    borrowerID = models.IntegerField()
    timestamp = models.DateTimeField()
    returned = models.BooleanField(default=False)
```

Above is the data model for the tables.

```
1 • SELECT * FROM diyapp.diyexch_app_user;
```



The screenshot shows a database query result grid. The grid has a header row with column names and a single data row below it. All data cells contain the value 'NULL'. The interface includes a toolbar with options like 'Result Grid', 'Filter Rows', 'Edit', 'Export/Import', and 'Wrap Cell Content'.

	id	email	username	cc_fullname	cc_number	cc_exp	zip_code	phone	img_url	tac_agree
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Above is showing the user table and its columns as created by the app.

I allowed some fields, like credit cards, to remain null so users can still browse the site without putting in a card number. A user cannot borrow a tool without a cc on file. The cards will be dummies for the purpose of the app.