Data Model Design **Checklist**

PRETTY PRINTED

Data model design can be a little complicated to explain, so I made this checklist so that if you answer all of the questions here, you're more likely to end up with a good data model.

A good data model helps achieve three things:

Simple Code | **Performance** | **Data Integrity**

Not all items apply to your data model, but you should be able to quickly go through the checklist and check your models. You don't need to follow my advice for each item, because each data model is different, but if you do things differently, you should at least know why.

Questions to Ask About Your Models (the classes defined in what is usually models.py that represent database tables)

- [ ] **Did your write down all of your models and draw connections between them?**

  If you aren't experienced with creating data models, then having a visual representation helps to understand what you're trying to build.

- [ ] **Does each model represent a single object?**

  Database tables normally represent either real-world things or single concepts. They don't usually represent actions.

- [ ] **Do you have consistent names for your tables?**

  Having consistent naming conventions for things like the use of capital letters, underscores, and singular/plural will make it a lot easier to understand what's going on in your code.

Does each model have a primary key?

In SQLAlchemy, you need to create your own primary key. In Django, it's created for you by default.

If a model is related to another model, is there a foreign key?

In all frameworks, you'll have to directly specify any relationships that exist between tables.

If you have a many-to-many relationship, do you have a third table for the combination of the two main model?

Django creates this third table for you by combining the names of the two models, but in SQLAlchemy, you'll have to create it yourself.

Are there two or more models that are very similar?

It might help to combine them into a single model and then have a field called type or category in the model to differentiate betweeen the types.

Will each model be used in the code you have planned to write?

You don't want to create extra models that stay around just in case you'll need them in the future. Focus on the models you need now, and if you need more in the future, you can always add them in.

Questions to Ask About the Fields Within your Models (attributes on your classes)

☐ Should the field have a constraint?

If a field should never be null, use things like null/nullable=False. If a field is unique, use unique=True. If the field represents a connection to another table, be sure to specify that it is a foreign key.

☐ Can you avoid using nullable fields?

If you believe a field should allow nulls, are you sure it won't be used in any calculations or comparisons? If you want to use the field in a calculation or comparison, consider using a default value like zero or an empty string so you don't have any issues when you write queries.

☐ Does each field appear just once in all your models?

If not, there should be a relationship between two models so you can retrieve the data instead of storing it twice.

☐ Does each field make sense on the model it belongs to?

If not, like the question before, you want to move it to another model and then create a relationship.

☐ Does each field have the best possible data type?

Try to be as specific as possible when choosing a data type. If you are working with decimal numbers, use the decimal type instead of float. If you are working with a long, unbounded list of text, use text instead of string or char.

☐ Are there any fields that can be derived or calculated from other fields?

Generally, you don't want to store any data you can calculate on the fly in the database. This will make your queries more flexible when you go to write them.

Do your fields have consistent names?

Like your model names, consistent names will make it easier to write and understand your code. It may not seem like a big deal, but it's very important.

Can any field broken into multiple fields?

Is one field doing too much work? For things like addresses, postal codes can stand alone. For phone numbers, country codes and area codes can stand alone.