

Table of Contents

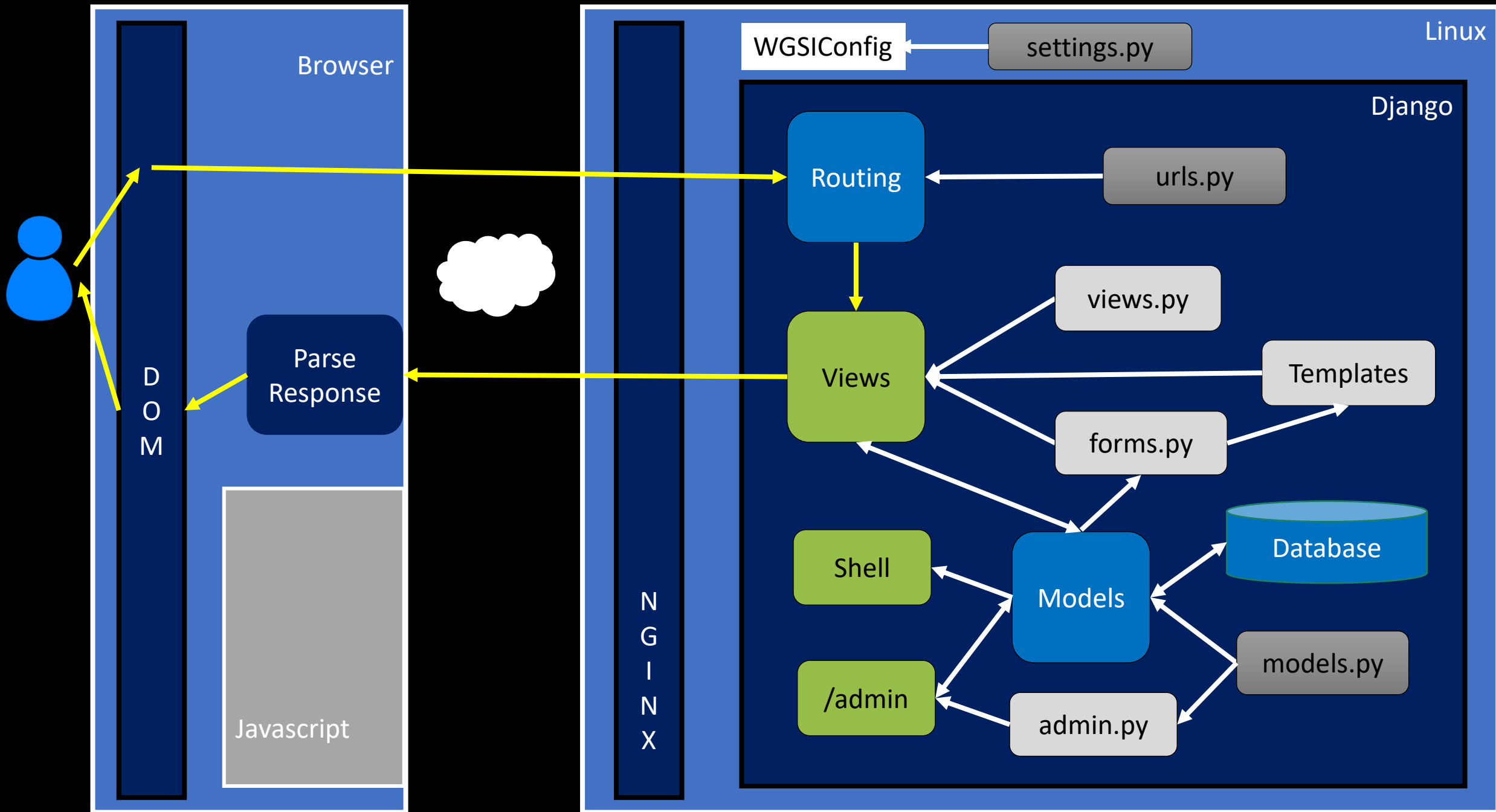
This slide deck consists of slides used in 2 lecture videos in Week 1. Below is a list of shortcut hyperlinks for you to jump into specific sections.

- (page 2) [Week 1: Many-to-Many Overview](#)
- (page 11) [Week 1: A Simple Many-To-Many Examples in Django](#)

Charles Severance
www.dj4e.com

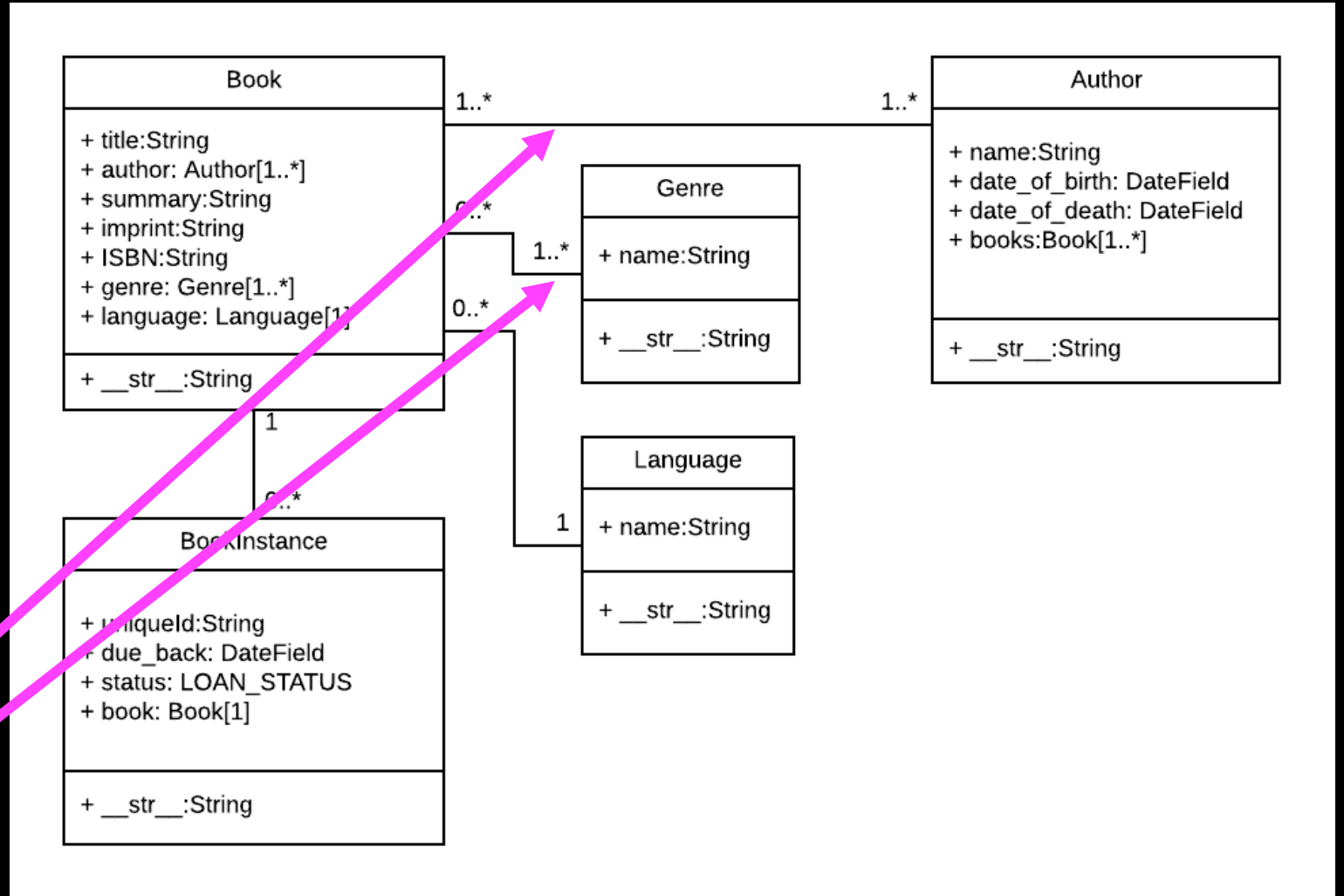
Data Modelling

Many to Many



Many-to-Many

"The relationships between Book / Genre and Book / Author are many-to-many"



DJ-04-Many-To-Many

Search Sheet

HomeInsertPage LayoutFormulasDataReviewViewShare

G13

fx

	A	B	C	D	E	F	G	H	
1	Title	ISBN	Lang	Genre	Author				
2	Wisdom of Crowds	385721706	en	Think	James Surowiecki				
3	Introduction to Networking	9781511654944	en	Tech	Charles Severance				
4	Introducción a las Redes	9781523627516	es	Tech	Fernando Tardio, Charles Severance				
5	Raspberry Pi	9781624311390	en	Tech, Kids	Kristen Fontiachairo, Charles Severance				
6	Where Wizards Stay Up Late	0684812010	en	Tech, Think	Katy Hafner, Matthew Lyon				
7									
8									

Sheet1

+

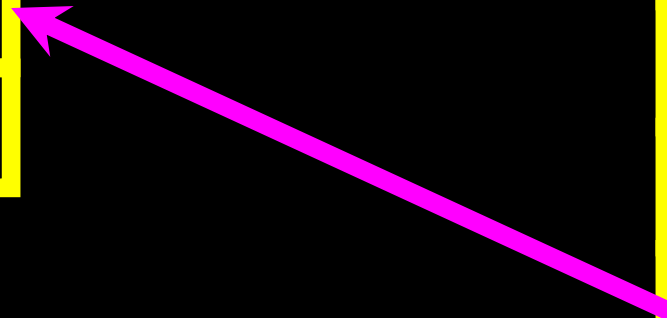
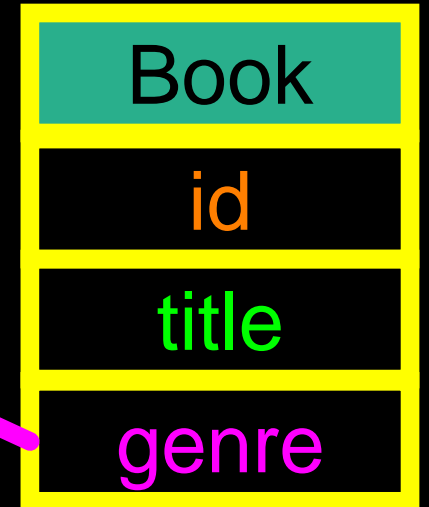
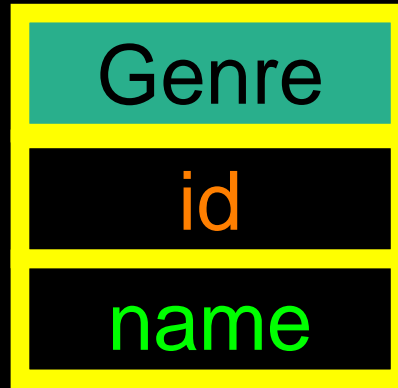
Ready100%

Many-To-Many

Title	ISBN	Genre	Author
Wisdom of Crowds	385721706	Think	James Surowiecki
Introduction to Networking	9781511654944	Tech	Charles Severance
Introducción a las Redes	9781523627516	Tech	Fernando Tardio, Charles Severance
Raspberry PI	9781624311390	Tech, Kids	Kristen Fontichiaro, Charles Severance
Where Wizards Stay Up Late	0684812010	Tech, Think	Katy Hafner, Matthew Lyon



One-To-Many?

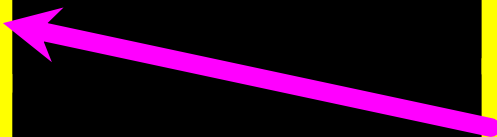
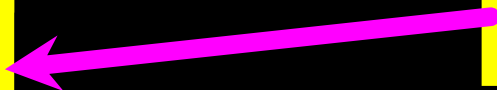


Genre
41
Think

Genre
43
Tech

Book
3
Wisdom of Crowds
genre_id

Book
5
Networking
genre_id



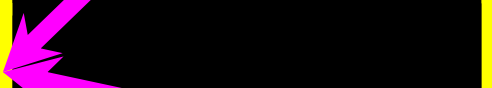
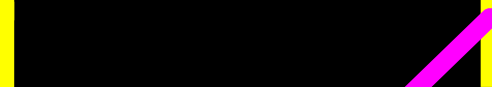
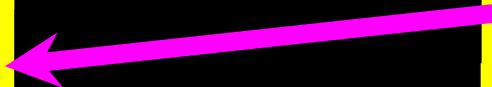
Multiple Genre Columns?

Genre
42
Kids

Genre
43
Tech

Book
4
Raspberry Pi
genre_id_01
genre_id_02

Book
5
Networking
genre_id_01
genre_id_02



Multiple Genre Columns?

Genre
43
Tech

Genre
43
Tech

NO!!!!!!!!!!

Book
4
Raspberry Pi
genre_id_01
genre_id_02

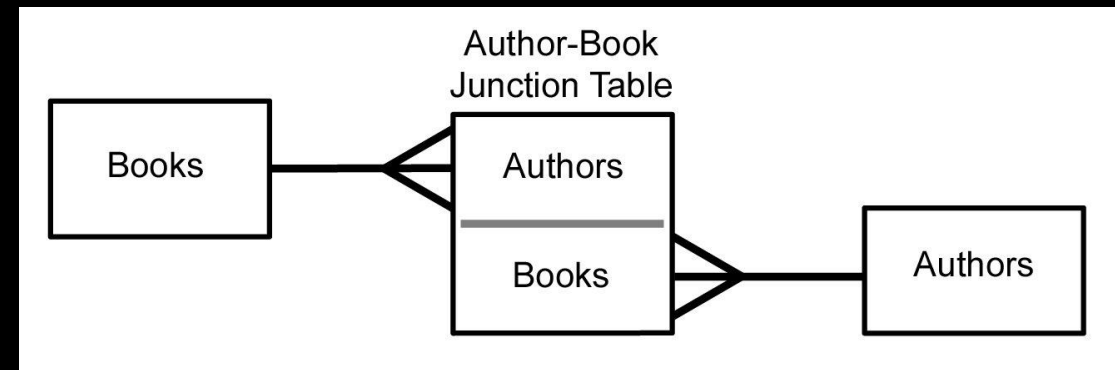
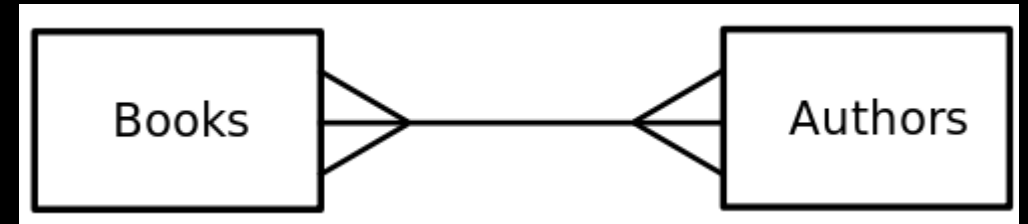
Book
5
Networking
genre_id_01
genre_id_02

Many-To-Many

`one_to_many + one_to_many = many_to_many`

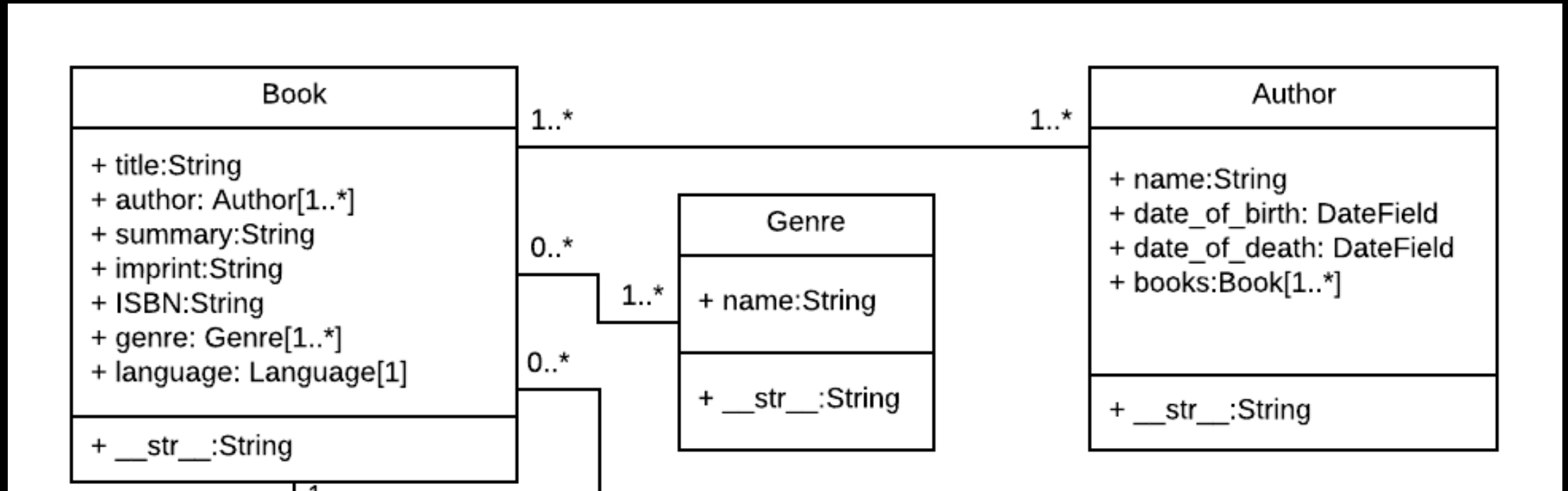
Many to Many

- Sometimes we need to model a relationship that is many to many.
- We need to add a “connection” table with two foreign keys.
- There is usually no separate primary key.
- We need two one-to-many relationships to capture a many-to-many



[https://en.wikipedia.org/wiki/Many-to-many_\(data_model\)](https://en.wikipedia.org/wiki/Many-to-many_(data_model))

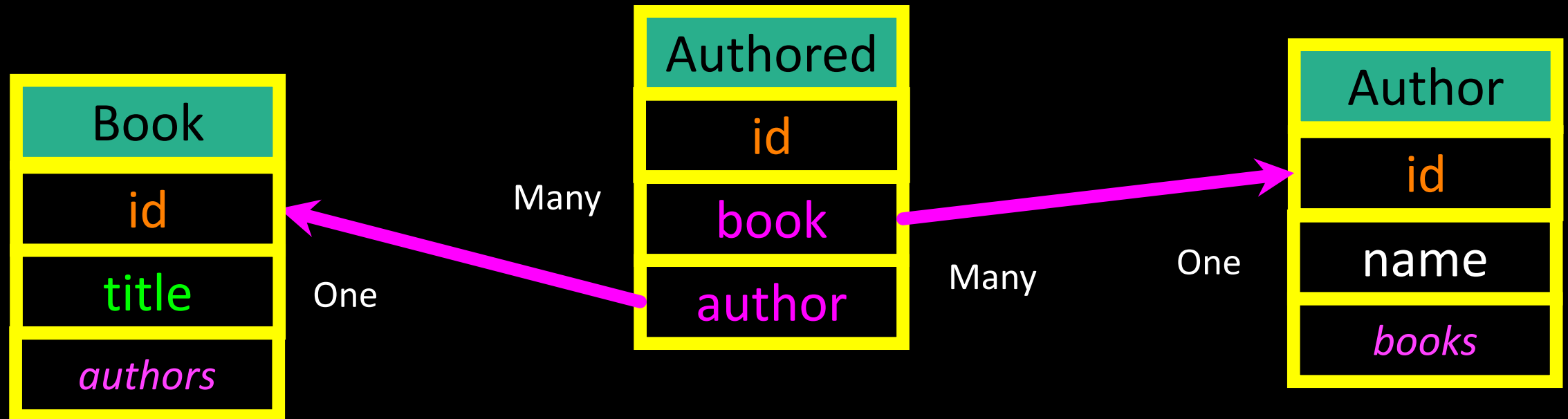
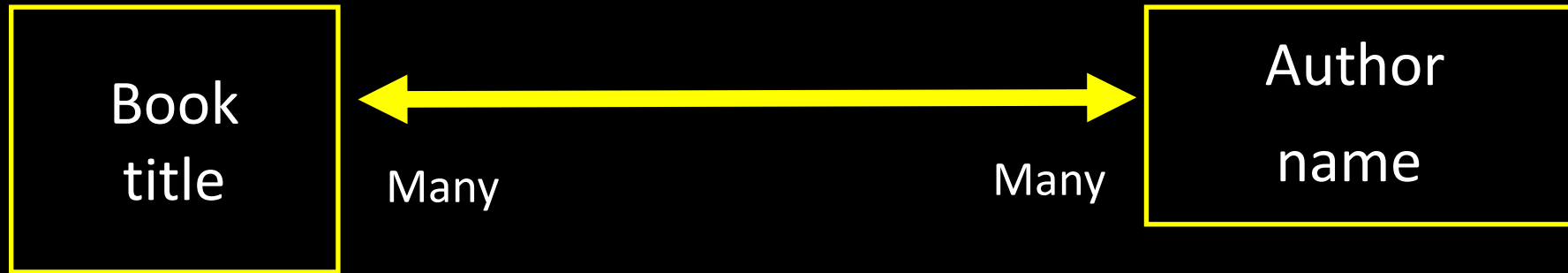
Many-to-Many in Locallibrary



Legend:

1..* Many with a minimum of 1

0..* Many with a minimum of 0



Django calls this the "through" table

```
from django.db import models

class Book(models.Model):
    title = models.CharField(max_length=200)
    authors = models.ManyToManyField('Author', through='Authored')

class Author(models.Model):
    name = models.CharField(max_length=200)
    books = models.ManyToManyField('Book', through='Authored')

class Authored(models.Model):
    book = models.ForeignKey(Book, on_delete=models.CASCADE)
    author = models.ForeignKey(Author, on_delete=models.CASCADE)
```

<https://github.com/csev/dj4e-samples/blob/master/samples/bookmany/models.py>

```
dj4e-samples$ python3 manage.py makemigrations
```

Migrations for 'bookmany':

bookmany/migrations/0001_initial.py

- Create model Author
- Create model Authored
- Create model Book
- Add field book to authored
- Add field books to author

```
dj4e-samples$ python3 manage.py migrate
```

Operations to perform:

Apply all migrations: admin, auth, autos, bookmany, bookone, contenttypes, favs, favsql, forums, gview, many, myarts, pics, rest, sessions, social_django, tracks, users

Running migrations:

Applying bookmany.0001_initial... OK

```
dj4e-samples$
```

Remember that makemigrations only "does something" when you create or alter a models.py file. The migrate only "does something" when there are migrations that are not yet applied to the database. Also an application must be added to settings.py before these commands see the models.py file for an application.

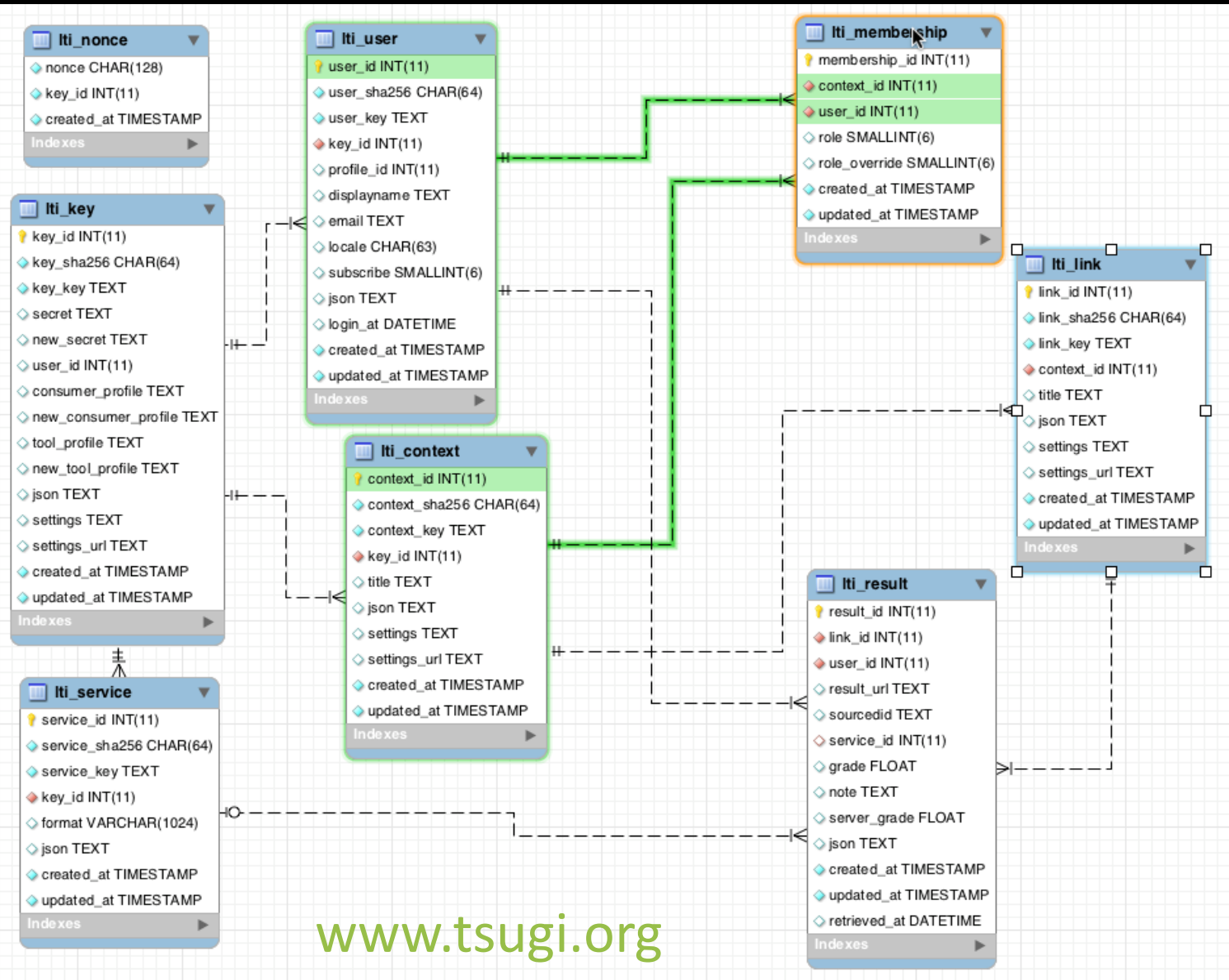

```

dj4e-samples$ python3 manage.py shell
>>> from bookmany.models import Book, Author, Authored
>>> b1 = Book(title='Networking')
>>> b1.save()
>>> b2 = Book(title='Raspberry')
>>> b2.save()
>>> a1 = Author(name='Fontichiaro')
>>> a1.save()
>>> a2 = Author(name='Severance')
>>> a2.save()
>>> Authored(book=b1, author=a2).save()
>>> Authored(book=b2, author=a1).save()
>>> Authored(book=b2, author=a2).save()
>>> b1.authors.values()
<QuerySet [{'id': 2, 'name': 'Severance'}]>
>>> b2.author_set.values()
<QuerySet [{'id': 1, 'name': 'Fontichiaro'}, {'id': 2, 'name': 'Severance'}]>
>>> a1.books.values()
<QuerySet [{'id': 2, 'title': 'Raspberry'}]>
>>> a2.book_set.values()
<QuerySet [{'id': 1, 'title': 'Networking'}, {'id': 2, 'title': 'Raspberry'}]>
>>> quit()
dj4e-samples$
```

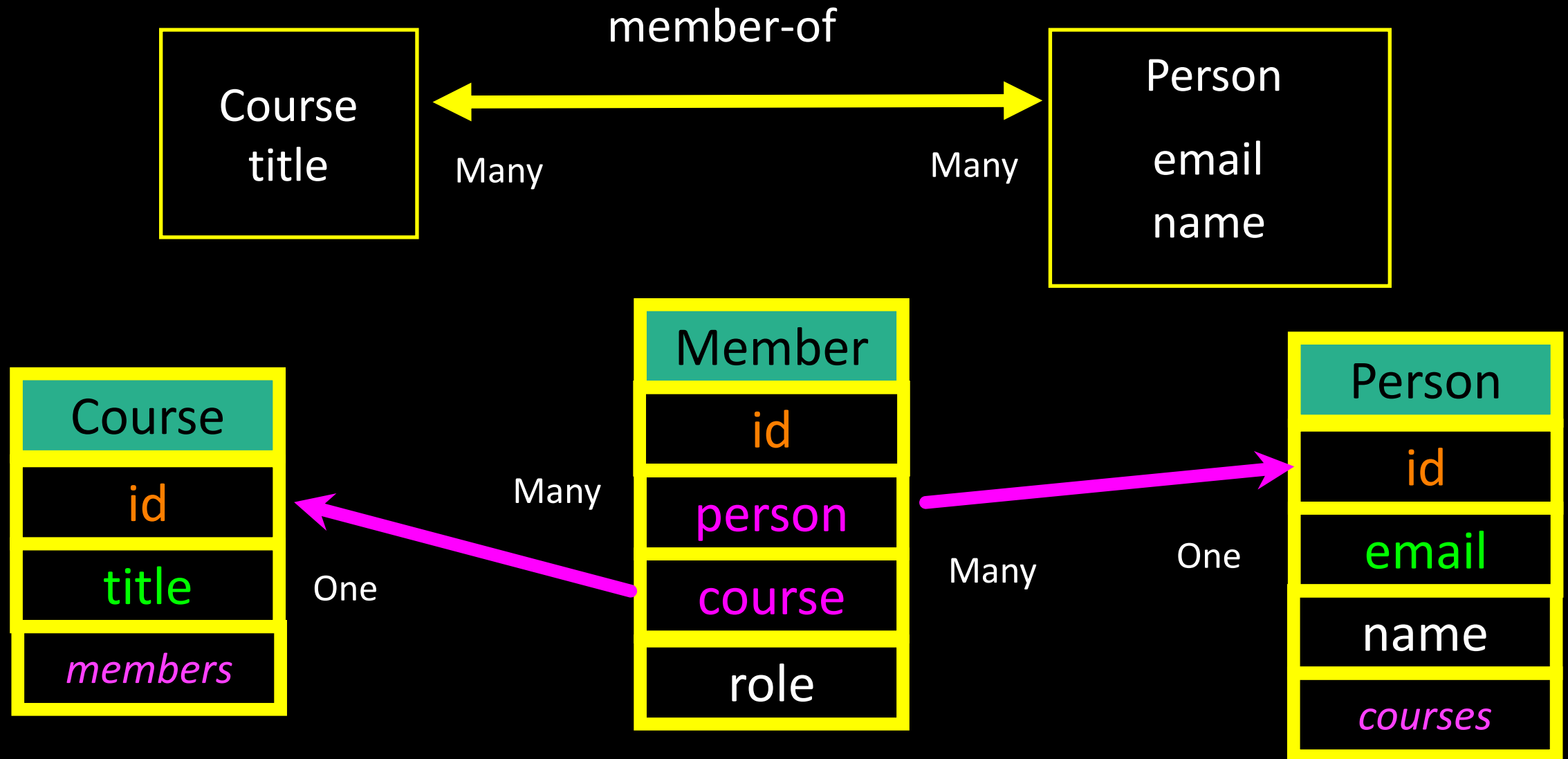


Another Example of Many-Many

Educational Technology



www.tsugi.org



```
from django.db import models

class Person(models.Model):
    email = models.CharField(max_length=128, unique=True)
    name = models.CharField(max_length=128, null=True)

class Course(models.Model):
    title = models.CharField(max_length=128, unique=True)
    members = models.ManyToManyField(Person,
                                     through='Membership', related_name='courses')
```

<https://github.com/csev/dj4e-samples/blob/master/samples/many/models.py>

<https://docs.djangoproject.com/en/3.0/ref/models/fields/#choices>

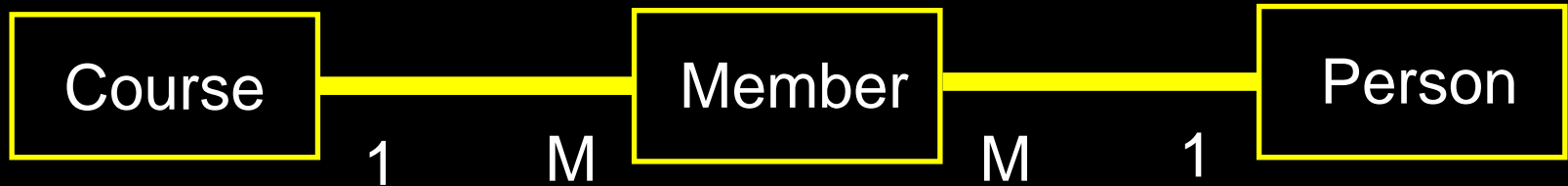
```
class Membership(models.Model):
    person = models.ForeignKey(Person, on_delete=models.CASCADE)
    course = models.ForeignKey(Course, on_delete=models.CASCADE)

    LEARNER = 1
    IA = 1000
    GSI = 2000
    INSTRUCTOR = 5000
    ADMIN = 10000

    MEMBER_CHOICES = (
        ( LEARNER, 'Learner' ),
        ( IA, 'Instructional Assistant' ),
        ( GSI, 'Grad Student Instructor' ),
        ( INSTRUCTOR, 'Instructor' ),
        ( ADMIN, 'Administrator' ),
    )

    role = models.IntegerField(
        choices=MEMBER_CHOICES,
        default=LEARNER,
    )
```

```
>>> from many.models import Person, Course, Membership
>>> p = Person(email='ted@umich.edu')
>>> p.save()
>>> c = Course(title='Woodcraft')
>>> c.save()
>>> c.id
6
>>> c.members.values()
<QuerySet []>
>>> m = Membership(role=Membership.INSTRUCTOR, course=c, person=p)
>>> m.save()
>>> m.id
15
>>> m.course_id
6
>>> c.members.values()
<QuerySet [{'id': 3, 'email': 'ted@umich.edu', 'name': None}]>
>>> p.courses.values()
<QuerySet [{'id': 6, 'title': 'Woodcraft'}]>
```



Demo Batch Loading from CSV

<https://github.com/csev/dj4e-samples/tree/master/samples/scripts>

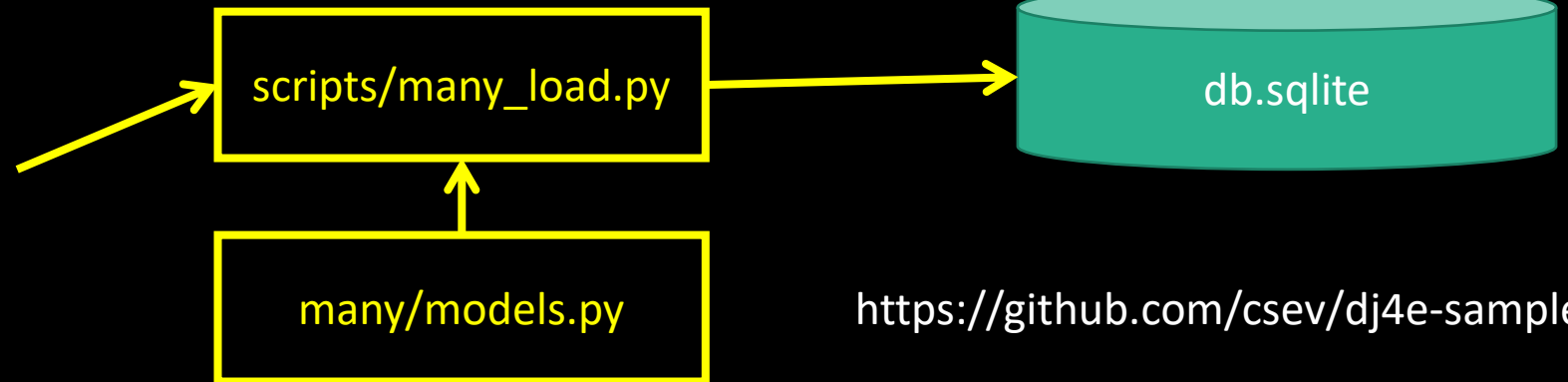
<https://django-extensions.readthedocs.io/en/latest/runscript.html>

Loading Data From A File

- Sometimes we need to pre-load data into our Django database
- This data might come from an API or file
- We need to write a Python program to function like the Django shell

`many/load.csv`

jane@tsugi.org,I,Python
ed@tsugi.org,L,Python
sue@tsugi.org,L,Python
ed@tsugi.org,I,Django
sue@tsugi.org,L,Django
ed@tsugi.org,I,SQL
jane@tsugi.org,L,SQL



<https://github.com/csev/dj4e-samples>

Installing django-extensions

```
dj4e-samples$ pip3 install django-extensions
Requirement already satisfied: django-extensions in
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages
Requirement already satisfied: six>=1.2 in
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages
dj4e-samples$
```

Note that this is installed already in dj4e-samples but for a new project you will need to install it yourself and edit **settings.py**

<https://django-extensions.readthedocs.io/en/latest/runscript.html>

Include Extensions in Project Settings

[dj4e-samples/settings.py](#)

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
  
    [ ... ]  
  
    # Extensions - see requirements.txt  
    'django_extensions',  
    'crispy_forms',  
  
    [ ... ]  
  
    'home.apps.HomeConfig',  
  
    # Sample Applications - don't copy  
    'hello.apps.HelloConfig',  
    'getpost.apps.GetpostConfig',  
    'users.apps.UsersConfig',  
  
    [ ... ]  
]
```

Make a scripts folder

```
dj4e-samples$ mkdir scripts  
dj4e-samples$ touch scripts/__init__.py
```

We place empty `__init__.py` files in folders to indicate to Python that they contain files that hold modules and as such are suitable for importing into a Python application.

<http://effbot.org/pyfaq/what-is-init-py-used-for.htm>

The Data File

```
dj4e-samples$ cat many/load.csv
jane@tsugi.org,I,Python
ed@tsugi.org,L,Python
sue@tsugi.org,L,Python
ed@tsugi.org,I,Django
sue@tsugi.org,L,Django
ed@tsugi.org,I,SQL
jane@tsugi.org,L,SQL
dj4e-samples$
```

[https://en.wikipedia.org/wiki/Cat_\(Unix\)](https://en.wikipedia.org/wiki/Cat_(Unix))

scripts/many_load.py

```
import csv # https://docs.python.org/3/library/csv.html

from many.models import Person, Course, Membership

def run():
    fhand = open('many/load.csv')
    reader = csv.reader(fhand)

    Person.objects.all().delete()
    Course.objects.all().delete()
    Membership.objects.all().delete()

    for row in reader:
        print(row)

        p, created = Person.objects.get_or_create(email=row[0])
        c, created = Course.objects.get_or_create(title=row[2])

        r = Membership.LEARNER
        if row[1] == 'I' : r = Membership.INSTRUCTOR
        m = Membership(role=r, person=p, course=c)
        m.save()
```

```
class Membership(models.Model):
    person = models.ForeignKey(Person, on_delete=models.CASCADE)
    course = models.ForeignKey(Course, on_delete=models.CASCADE)

    LEARNER = 1
    IA = 1000
    GSI = 2000
    INSTRUCTOR = 5000
    ADMIN = 10000

    MEMBER_CHOICES = (
        ( LEARNER, 'Learner' ),
        ( IA, 'Instructional Assistant' ),
        ( GSI, 'Grad Student Instructor' ),
        ( INSTRUCTOR, 'Instructor' ),
        ( ADMIN, 'Administrator' ),
    )

    role = models.IntegerField(
        choices=MEMBER_CHOICES,
        default=LEARNER,
    )

    def __str__(self):
        return "Person " + str(self.person.id) + " <--> Course " + str(self.course.id)
```

```
dj4e-samples$ python3 manage.py runscript many_load
```

```
['jane@tsugi.org', 'I', 'Python']
```

```
['ed@tsugi.org', 'L', 'Python']
```

```
['sue@tsugi.org', 'L', 'Python']
```

```
['ed@tsugi.org', 'I', 'Django']
```

```
['sue@tsugi.org', 'L', 'Django']
```

```
['ed@tsugi.org', 'I', 'SQL']
```

```
['jane@tsugi.org', 'L', 'SQL']
```

```
dj4e-samples$
```

```
for row in reader:  
    print(row)
```

```
p, created = Person.objects.get_or_create(email=row[0])  
c, created = Course.objects.get_or_create(title=row[2])
```

```
r = Membership.LEARNER  
if row[1] == 'I' : r = Membership.INSTRUCTOR  
m = Membership(role=r, person=p, course=c)  
m.save()
```


Many-to-Many in the Django Shell

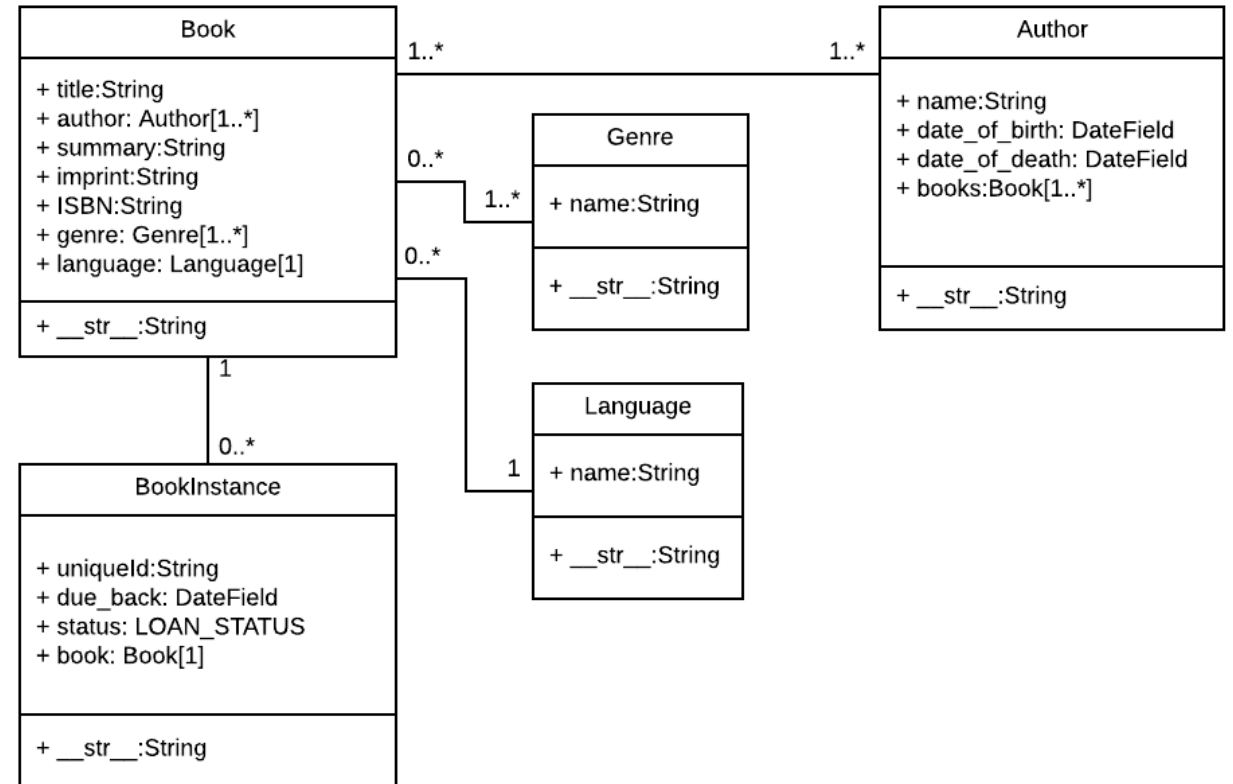
```
dj4e-samples$ python3 manage.py shell
>>> from many.models import Person, Course, Membership
>>> Person.objects.values()
<QuerySet [{'id': 1, 'email': 'jane@tsugi.org', 'name': None},
            {'id': 2, 'email': 'ed@tsugi.org', 'name': None},
            {'id': 3, 'email': 'sue@tsugi.org', 'name': None}]>
>>> x = Person.objects.get(pk=1)
>>> x.email
jane@tsugi.org
>>> x.courses.values()
<QuerySet [{'id': 1, 'title': 'Python'},
            {'id': 3, 'title': 'SQL'}]>
>>> y = Course.objects.get(pk=2)
>>> y.title
'Django'
>>> y.members.values()
<QuerySet [{'id': 2, 'email': 'ed@tsugi.org', 'name': None},
            {'id': 3, 'email': 'sue@tsugi.org', 'name': None}]>
>>>
```

Looking at the "through" table

```
>>> y = Course.objects.get(pk=2)
>>> y.membership_set.all().values()
<QuerySet [
  {'id': 4, 'person_id': 2, 'course_id': 2, 'role': 5000},
  {'id': 5, 'person_id': 3, 'course_id': 2, 'role': 1}
]>
>>>
```

Summary

- Data modelling is both simple and complex
- "Don't allow string data to be replicated"
- We use keys and relationships
 - Primary key
 - Foreign key
- Relationships
 - One-to-Many
 - Many-to-Many



Acknowledgements / Contributions

These slides are Copyright 2019- Charles R. Severance (www.dr-chuck.com) as part of www.dj4e.com and made available under a Creative Commons Attribution 4.0 License. Please maintain this last slide in all copies of the document to comply with the attribution requirements of the license. If you make a change, feel free to add your name and organization to the list of contributors on this page as you republish the materials.

Initial Development: Charles Severance, University of Michigan School of Information

Insert new Contributors and Translators here including names and dates

Continue new Contributors and Translators here