Jugando con el ORM de Django

Federico Martínez
Santiago Avendaño

¿Qué es un ORM?

¿Qué es un ORM?

Object

Relational

Mapper

¿Qué es un ORM?


NOBODY LIKES ME!
I WISH I COULD LIKE YOU, CHARLIE BROWN, BUT I CAN'T...

SQL

En esta charla:

- Conceptos de bases de datos (relacionales)
- ORM de Django
- Problemas comunes

Y por qué hablar de bases de datos relacionales?

Bases de datos relacionales

Bases de datos relacionales

- Implementación de la teoría del álgebra relacional

Bases de datos relacionales

- Implementación de la teoría del álgebra relacional
- Usamos *tablas* que tiene *registros*

Bases de datos relacionales

- Implementación de la teoría del álgebra relacional
- Usamos *tablas* que tienen *registros*
- Estan buenísimas

¿Qué es una tabla?

¿Qué es una tabla?

Col 1	Col 2	Col 3	Col 4
A	B	C	D
X	Y	Z	W

¿Qué es una tabla?(en el fondo)

¿Qué es una tabla?(en el fondo)

SELECT * FROM tabla

SELECT * FROM tabla

Leer todo el archivo!

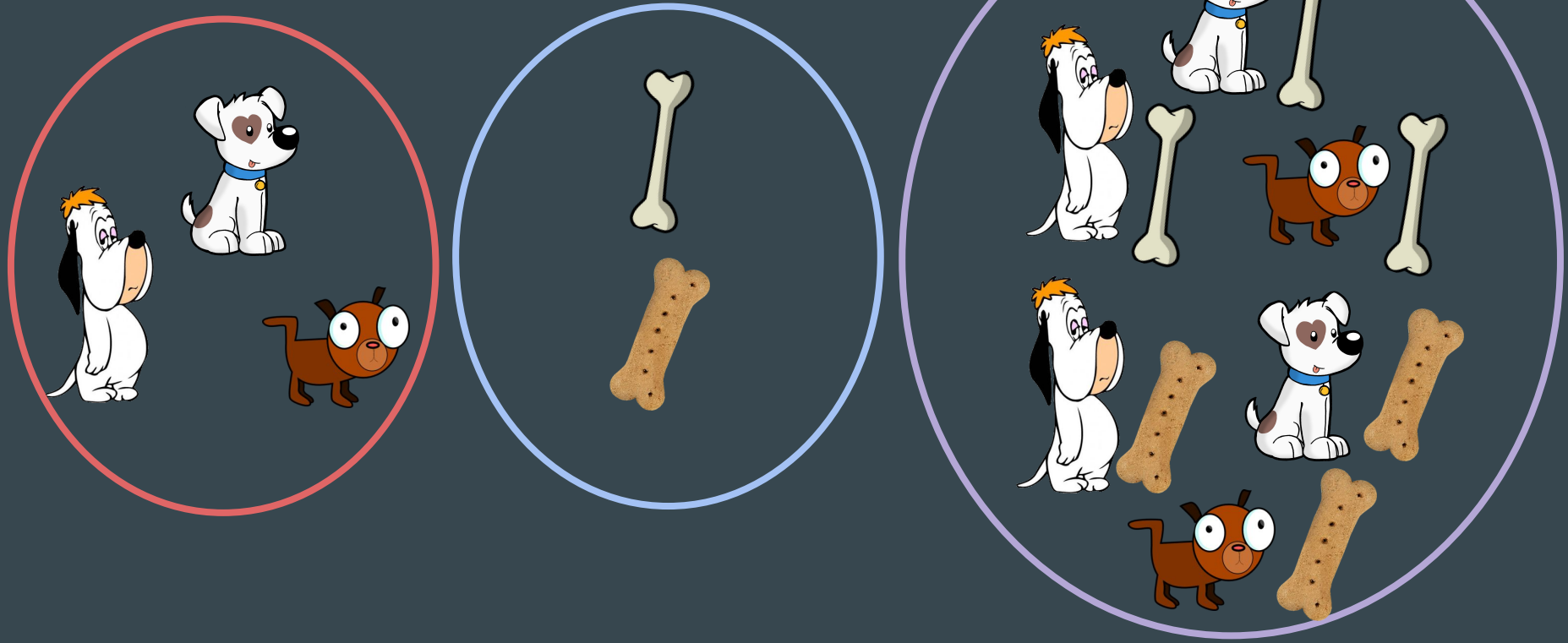
SELECT * FROM tabla WHERE col = 23

SELECT * FROM tabla WHERE col = 23

Leer todo el archivo!

Joins

Joins



SELECT * FROM tabla1, tabla2

SELECT * FROM tabla1, tabla2

Leer todo el archivo2 por cada registro en archivo1!

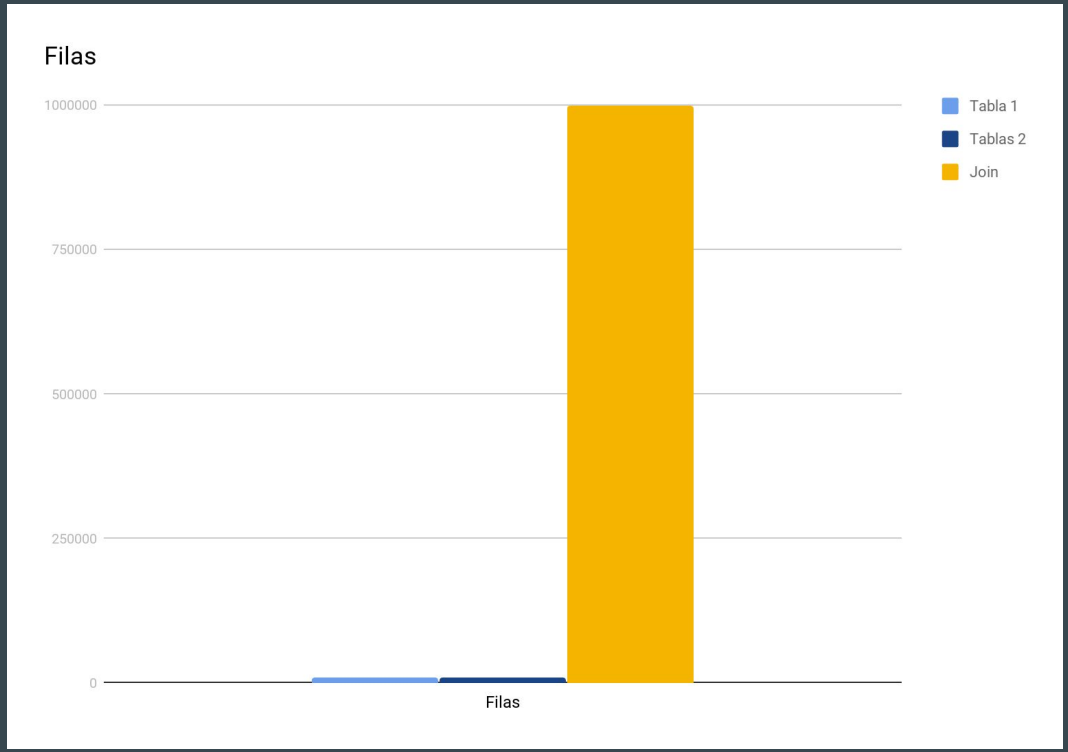
```
SELECT * FROM tabla1
INNER JOIN tabla2
ON tabla1.col1 = tabla2.col2
```

```
SELECT * FROM tabla1
INNER JOIN tabla2
ON tabla1.col1 = tabla2.col2
```

Leer todo el archivo2 por cada registro en archivo1!*

SELECT * FROM tabla1, tabla2

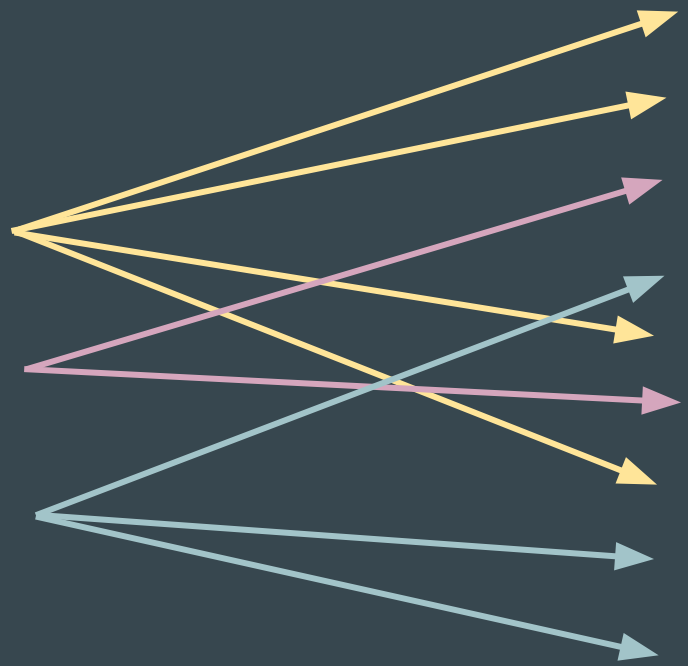
- Tabla 1: 1000
- Tabla 2: 1000
- Join 1000000



Indices

Indices

Col 2
1
2
3



Col 1	Col 2	Col 3	Col 4
	1		
	1		
	2		
	3		
	1		
	2		
	1		
	3		
	3		

Indices

- Búsqueda logarítmica
- Búsqueda por rango
- Claves compuestas

Indices

Che, indexemos todo!

Pero dónde está todo esto en Django???

Modelos

```
from django.db import models

class Book(models.Model):
  title = models.CharField(max_length=100)
  pages = models.PositiveIntegerField()

  genre = models.ForeignKey(
      Genre, on_delete=models.CASCADE
  )
  authors = models.ManyToManyField(
      Writer, related_name='books'
  )
```

Managers y Querysets

```
all_books = Book.objects.all()
```

```
SELECT *
FROM library_book
```

Filtros

```
Book.objects.filter(
    title__contains='anillos'
)
```

```
SELECT *
FROM library_book
WHERE
  title LIKE '%anillos%';
```

Joins

```
Book.objects.filter(
  authors__user__username='JRR'
)
```

```
SELECT *
FROM library_book
INNER JOIN library_book_authors
ON library_book.id =
library_book_authors.book_id
INNER JOIN library_writer ON
(library_book_authors.writer_id =
library_writer.id)
INNER JOIN auth_user ON
library_writer.user_id=auth_user.id
WHERE auth_user.username = 'JRR'
```

Indices

```
class Book(models.Model):
  ...
  pages = models.PositiveIntegerField(
    db_index=True
  )

class Writer(models.Model):
  class Meta:
    indexes = [
      models.Index(
        fields=['last_name','first_name']),
      )
```

Problemas comunes

Procesar datos en Python

```
pages_list = Book.objects.all()
  .values_list('pages', flat=True)

max_pages = max(pages_list)
```

```
from django.db.models import Max

max_pages = \
Book.objects.all().aggregate(
  Max('pages')
)['pages__max']
```

Muchas queries

```
all_books = Book.objects.all()
for b in all_books:
    print(b.title, b.genre.name)
```

```
SELECT * FROM "library_book"
SELECT * FROM "library_genre" WHERE "library_genre"."id" = 4;
SELECT * FROM "library_genre" WHERE "library_genre"."id" = 4;
SELECT * FROM "library_genre" WHERE "library_genre"."id" = 4;
SELECT * FROM "library_genre" WHERE "library_genre"."id" = 4;
SELECT * FROM "library_genre" WHERE "library_genre"."id" = 5;
SELECT * FROM "library_genre" WHERE "library_genre"."id" = 6;
SELECT * FROM "library_genre" WHERE "library_genre"."id" = 6;
...
```

Muchas queries

```
all_books = Book.objects.all().select_related("genre")
for b in all_books:
    print(b.title, b.genre.name)
```

```
SELECT "library_book"."id", "library_book"."title",
"library_book"."pages", "library_book"."genre_id",
"library_book"."publication_date",
"library_genre"."id", "library_genre"."name"
FROM "library_book"
INNER JOIN "library_genre"
ON ("library_book"."genre_id" = "library_genre"."id");
```

Muchas queries

```
all_books_with_writers = Book.objects.all().prefetch_related('authors')

for b in all_books_with_writers:
    print(b.title, b.authors.all())
```

```
SELECT * FROM "library_book";
```

```
SELECT * FROM "library_writer"
INNER JOIN "library_book_authors"
ON ("library_writer"."id" = "library_book_authors"."writer_id")
WHERE "library_book_authors"."book_id" IN (2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
12, 13, 14, 15) ;
```

Queries innecesarias

```
Book.objects.filter(authors__user__username__contains='J')

Book.objects.filter(authors__user__username__contains='JR')

Book.objects.filter(authors__user__username__contains='JRR')

Book.objects.filter(authors__user__username__contains='JRR ')
```

Procesamientos innecesarios

- Traer datos que no son necesarios

```
Book.objects.all().select_related('genre').values('title', 'genre__name')
```

- Ordenar cosas si no es necesario

```
class Meta:
   ordering = ['last_name']
```

```
SELECT * FROM "library_writer" ORDER BY "library_writer"."last_name" ASC
```

Nos queda tiempo?

Preguntas?

Muchas Gracias!!!