Understand the

N+1 Query

Problem in Django

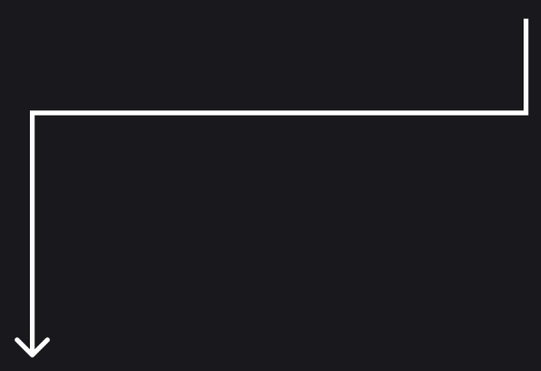
N+1 Query Problem

The N+1 query problem is a common issue that arises in django ORMs. In this problem, each related object (foreign key, many to many etc.), fetch the additional query and that impact the performance of database queries.

#pythonworld

Let’s understand the problem by example

Suppose i have two models Author and Books, and each author could have mutiple books, so that’s why we have foreign key relation with Author in Book model



```python
from django.db import models

class Author(models.Model):
    name = models.CharField(max_length=100)

class Book(models.Model):
    title = models.CharField(max_length=100)
    author = models.ForeignKey(Author, on_delete=models.CASCADE)
```

This will return all the book objects

```python
from .models import Book

books = Book.objects.all()

for book in books:
    print(book.title, book.author.name)
```

If we loop the books objects and call the author name, so here it will fetch the additional query to get the author name and same things it do for all object for eg. if i have 5 book objects, so N would be N=5 so 5+1 (N+1) = 6, it means 5 additional query will perform, and this will slow the database queries.

How to fix this N+1 query problem

To fix the n+1 query problem, we can use **select_related()** or **prefetch_related()** methods.

By select_related() method

```python
from .models import Book

books = Book.objects.select_related('author').all()

for book in books:
    print(book.title, book.author.name)
```

By select_related() method it fetch related objects in a single query by performing a SQL JOIN. It works for ForeignKey and OneToOne relationships.

By prefetch_related() method

```python
from .models import Book

books = Book.objects.prefetch_related('author').all()

for book in books:
    print(book.title, book.author.name)
```

By prefetch_related() method it fetch related objects in a in separate queries and then perform the JOIN operation in Python. It works for ForeignKey , OneToOne and ManyToMany relationships.

Conclusion

By using select_related() or prefetch_related(), you can significantly reduce the number of database queries and avoid the N+1 query problem in your Django applications.

#pythonworld

For more django
related content

Subscribe

To My Youtube
Channel

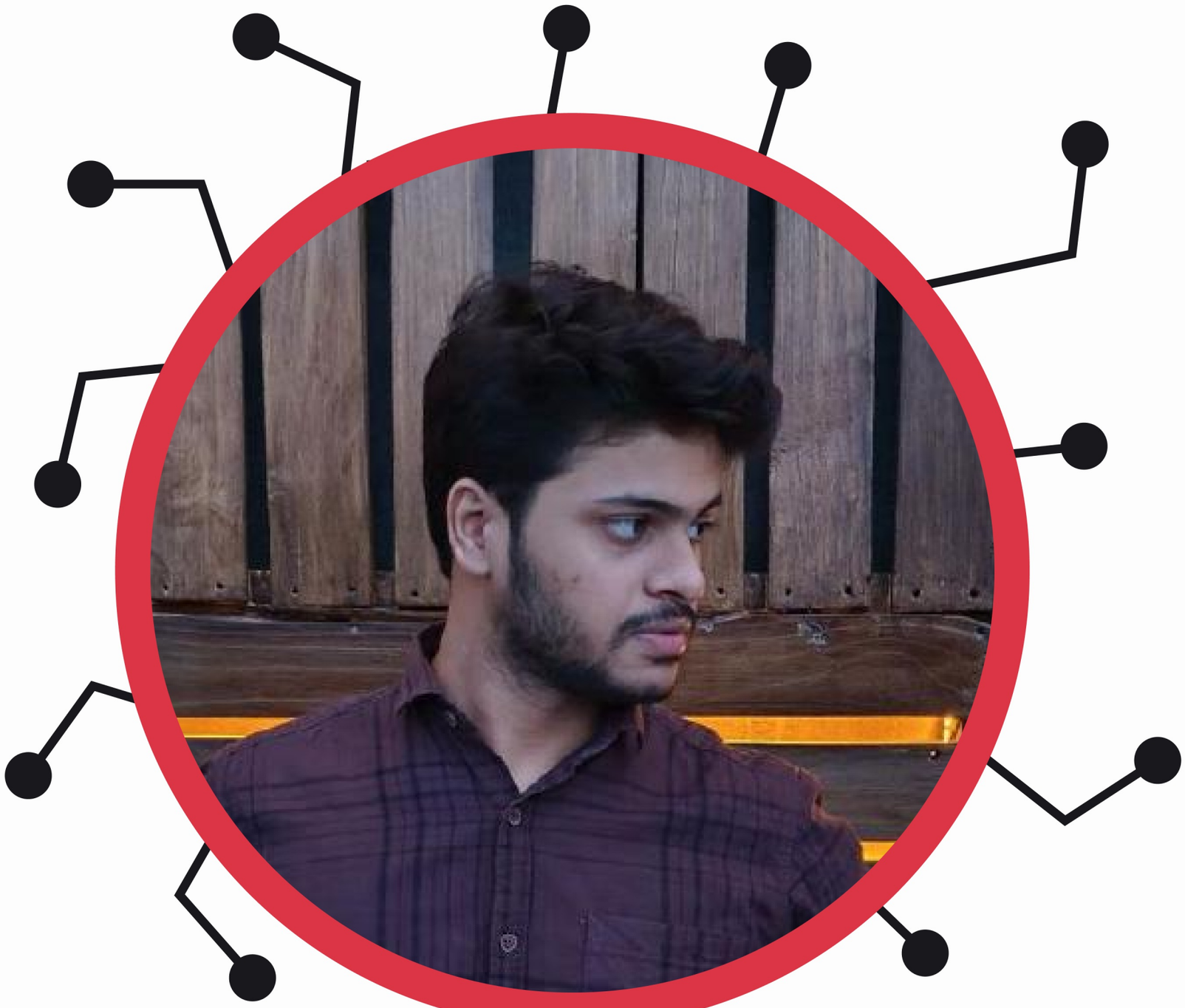
Link in bio

#pythonworld

DID YOU FIND THIS

HELPFUL

Let me know in the comment



Aashish Kumar

Software Engineer

Follow for more ❤️