One-to-Many Relationships using Flask SQLAlchemy

Pratap Sharma

August 22nd, 2023 - 59 views / 4 mins read

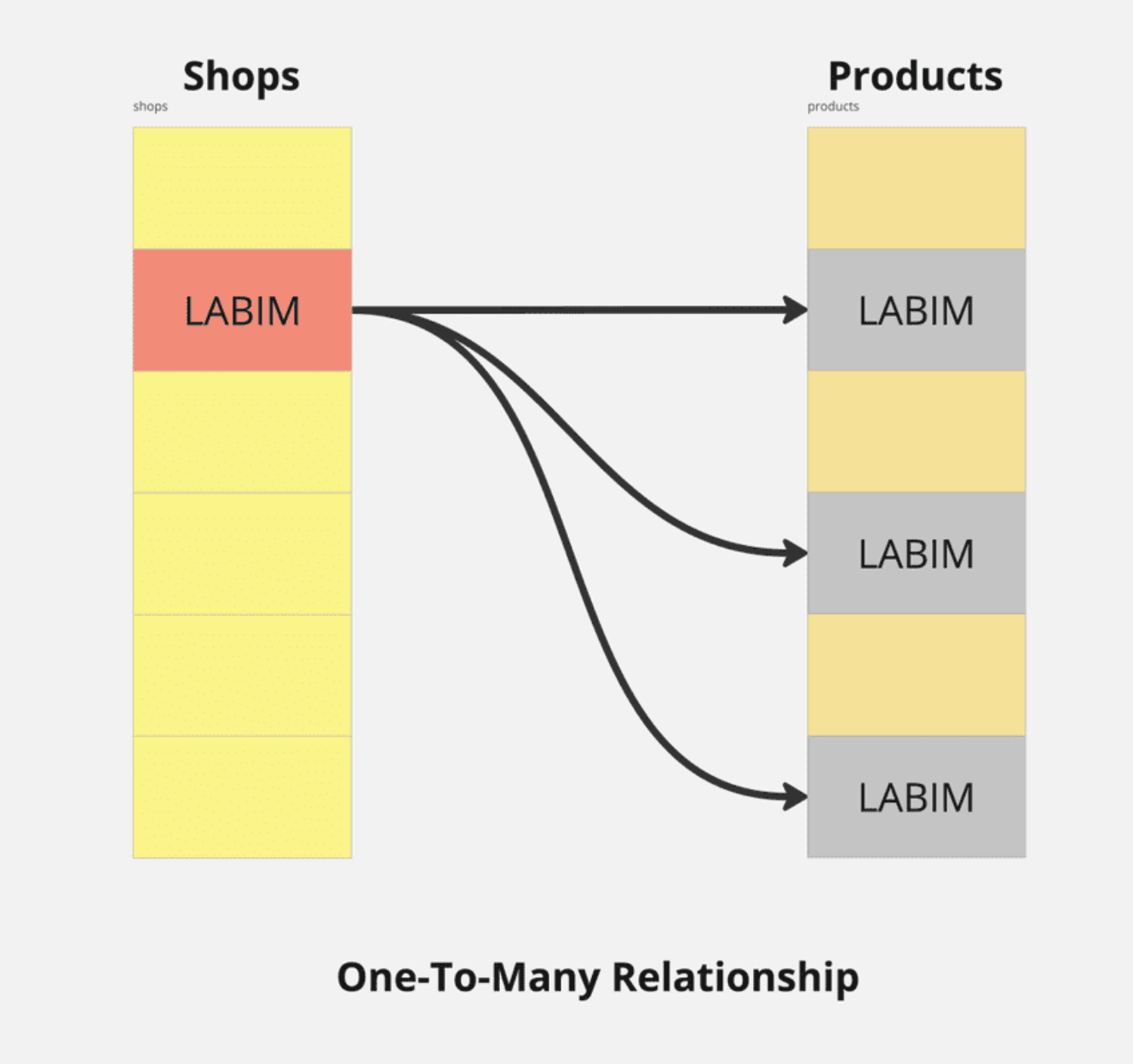
flask sql alchemy

In this blog we will learn how to envision a one-to-many relationships between two tables in SQLAlchemy.

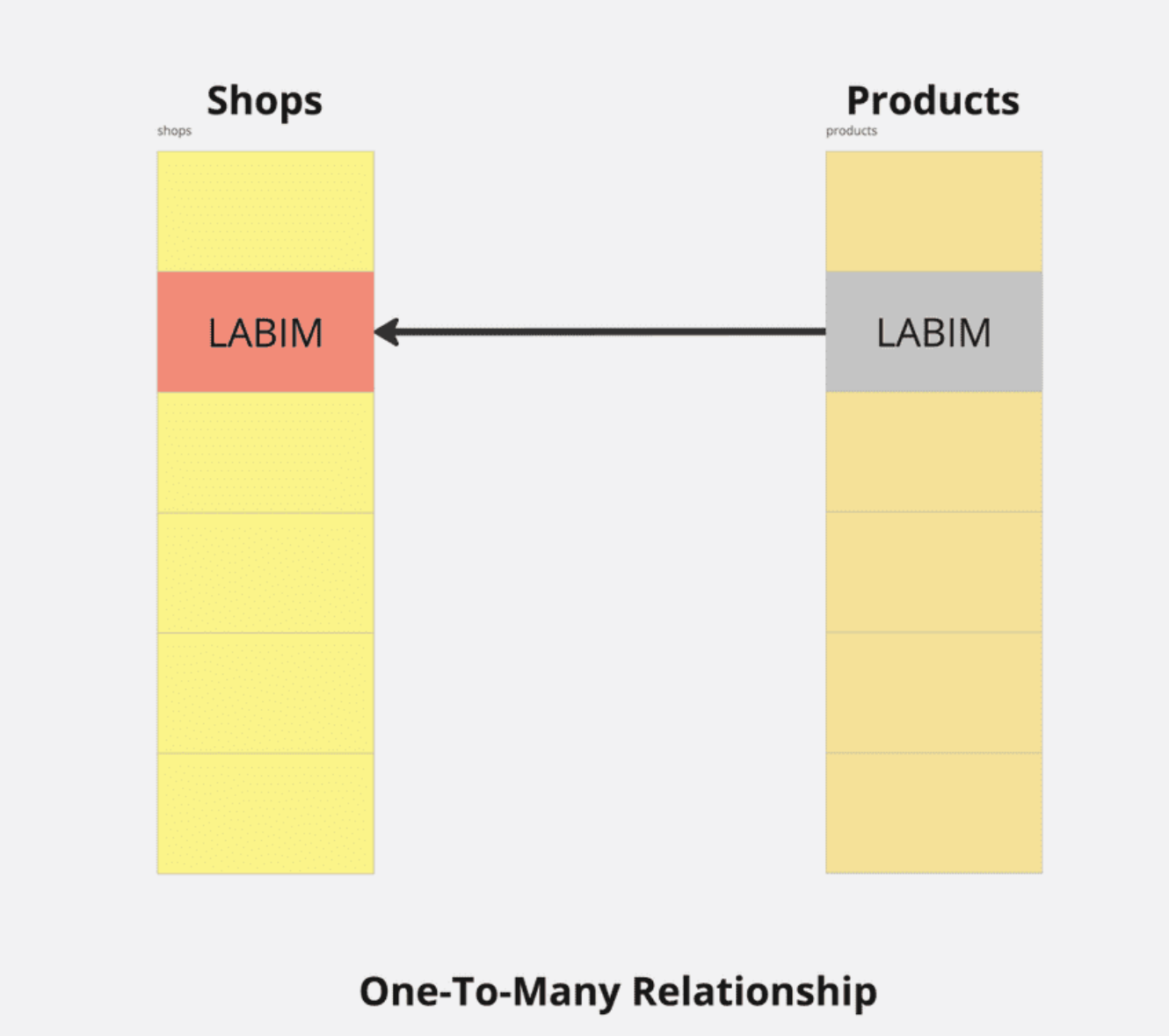
A one-to-many (1:N) relationship occurs when, for one instance of entity A, there can be zero, one, or many instances of entity B. Conversely, for one instance of entity B, there exists zero or one instance of entity A.

> **This is a part of REST APIs with Flask and Python in 2023 Course**

To visually represent this relationship, envision entities A and B as generic tables such as **Shop** and **Product**. In a one-to-many relationship, a single record in table **Shop** can be associated with multiple records in **Product**, while a record in **Product** can be linked to just one record in table **Shop**. The following images provides an illustration of a 1:N relationship between Tables **Shop** and **Product**, viewed first from the perspective of table **Shop** and then from the perspective of **Product**.



1. This image illustrates a 1:N relationship between table **Shop** and **Product** from the perspective of table **Shop**, showcasing that one record in table **Shop** is associated with many records in **Product** (Shop:Product = 1:N).



2. This image provides the relationship from the viewpoint of **Product**. In this view, one record in **Product** is linked to one record in table **Shop** (Product:Shop = 1:1).

The most prevalent type of relationship is the one-to-many relationship, which is commonly found in relational databases. In fact, most databases consist of one-to-one relationships and numerous one-to-many relationships. For instance, a Shop can have multiple Products but each product is associated with a single shop. When dealing with instances of two entities, A and B, a one-to-many relationship exists between two instances (Ai and Bi) if Ai can be linked to zero, one, or more instances of entity B, and Bi can be connected to zero or one instance of entity A.

Other examples of one-to-many relationships include the relationship between a daughter and her biological mother. A woman may have zero, one, or more biological daughters, but a daughter can have only one biological mother.

One-to-Many Relationships using SQLAlchemy

Let's have our `ShopModel` and `ProductModel`. Each shop can have multiple products, creating a one-to-many relationship.

```
from flask_sqlalchemy import SQLAlchemy

db = SQLAlchemy()

class ProductModel(db.Model):
    __tablename__ = "products"

    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(80), unique=True, nullable=False)
    price = db.Column(db.Float(precision=2), unique=False, nullable=False)

    shop_id = db.Column(db.Integer, db.ForeignKey('shops.id'),
unique=False, nullable=False)
    shop = db.relationship("ShopModel", back_populates="products")

class ShopModel(db.Model):
    __tablename__ = "shops"

    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(80), unique=True, nullable=False)
    products = db.relationship('ProductModel', back_populates='shop',
lazy='dynamic')
```

What is lazy="dynamic"?

Here's how you can define these models with `lazy="dynamic"`. In this example:

1. Each `ShopModel` has a one-to-many relationship with `ProductModel`. A shop can have multiple products, but each product belongs to only one shop.

2. We've added the `lazy='dynamic'` option to the `products` relationship in the `ShopModel` model. This means that when you access `shops.products`, you get a query object rather than immediately loading all related products.

To obtain a list of ProductModel objects, representing all products associated with a specific ShopModel instance, you can use the following approach:

```
shop.products.all()
```

```
shop.products.filter_by(name=="Rice").first()
```

Learn More

1. Flask and SQLAlchemy: Better Data Management
2. Beginner's Guide to HTTP Methods and Status Codes
3. What is a Docker container?

Please let me know if there's anything else I can add or if there's any way to improve the post. Also, leave a comment if you have any feedback or suggestions.

Up next

Beginner's Guide to HTTP Methods and Status Codes

What is a Docker container?

Author

Hey, I'm Pratap, a full stack software engineer and an instructor. I write about what I know to help viewers like you. If you enjoy my content, please consider supporting what I do!

Buy me a coffee

Become a Patron