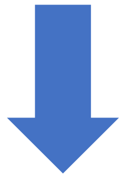
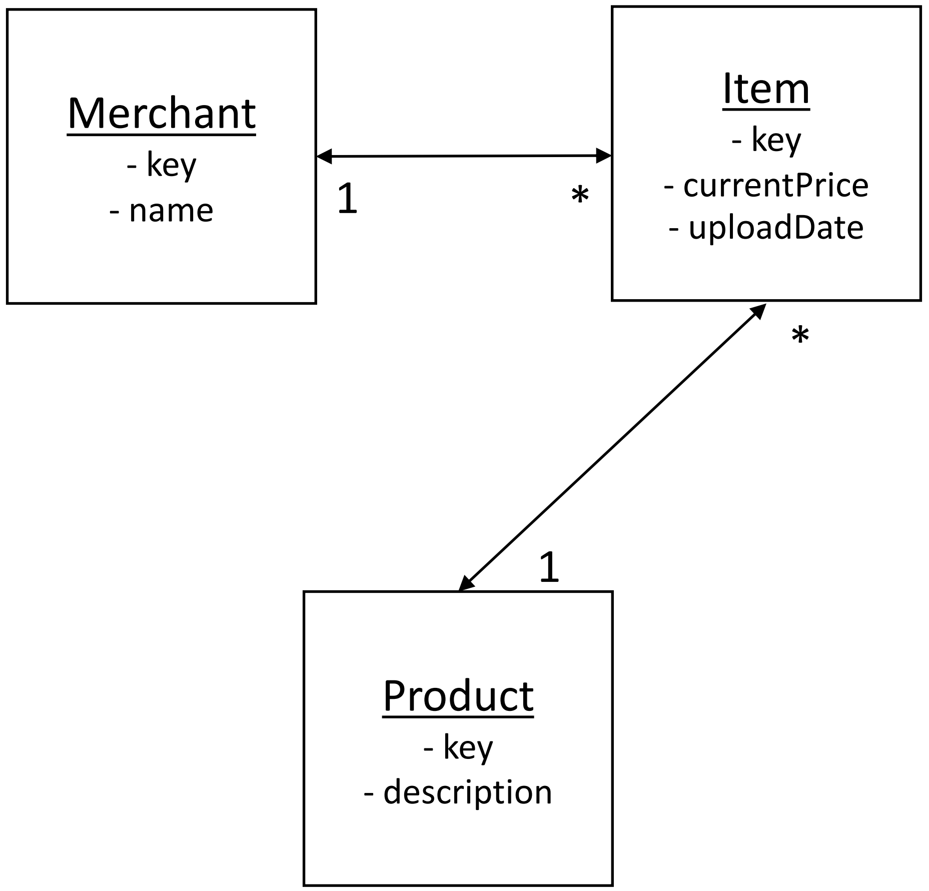
The Ecommerce Example: Data Model

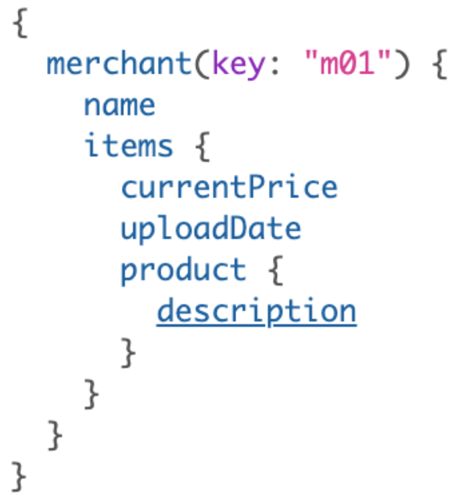
- A merchant can sell many items
- An item is sold by exactly one merchant
- An item represents exactly one product
- A product can be referenced by many items



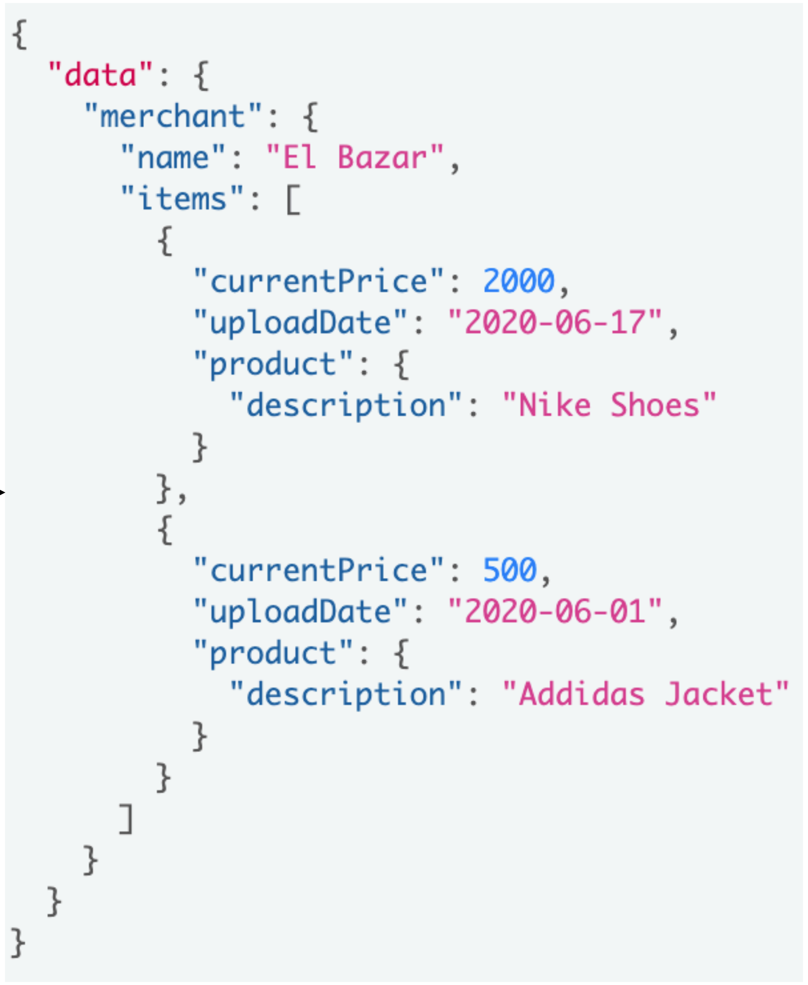
•Therefore a product can be sold by different merchants at various prices



The Ecommerce Example: Query

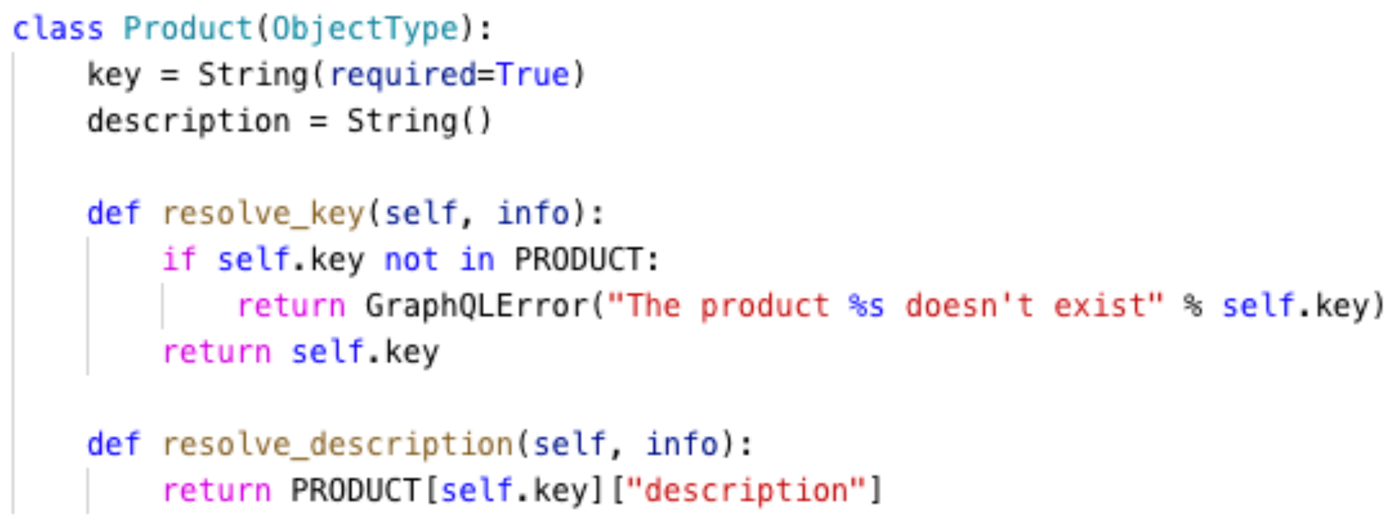


```
{
  merchant(key: "m01") {
    name
    items {
      currentPrice
      uploadDate
      product {
        description
      }
    }
  }
}
```



```
{
  "data": {
    "merchant": {
      "name": "El Bazar",
      "items": [
        {
          "currentPrice": 2000,
          "uploadDate": "2020-06-17",
          "product": {
            "description": "Nike Shoes"
          }
        },
        {
          "currentPrice": 500,
          "uploadDate": "2020-06-01",
          "product": {
            "description": "Addidas Jacket"
          }
        }
      ]
    }
  }
}
```

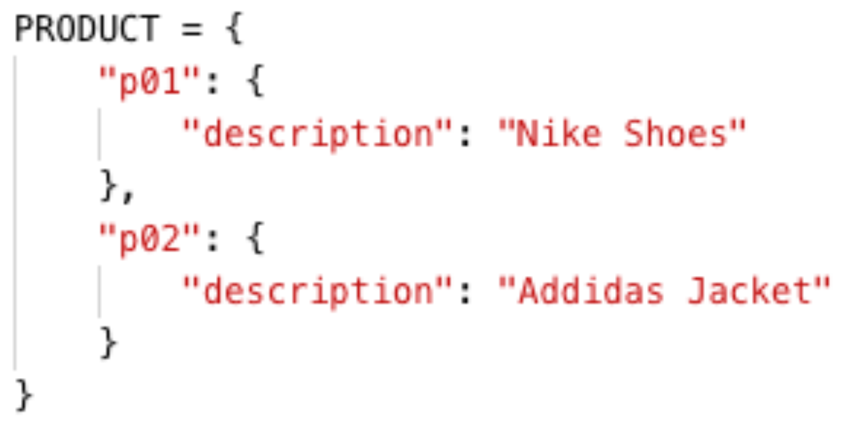
The Ecommerce Example: Product Type



```python
class Product(ObjectType):
    key = String(required=True)
    description = String()

    def resolve_key(self, info):
        if self.key not in PRODUCT:
            return GraphQLError("The product %s doesn't exist" % self.key)
        return self.key

    def resolve_description(self, info):
        return PRODUCT[self.key]["description"]
```



```python
PRODUCT = {
    "p01": {
        "description": "Nike Shoes"
    },
    "p02": {
        "description": "Addidas Jacket"
    }
}
```

The Ecommerce Example: Item Type

```python
class Item(ObjectType):
    key = String(required=True)
    product = Field(Product)
    upload_date = Date()
    current_price = Int()

    def resolve_key(self, info):
        if self.key not in ITEM:
            return GraphQLError("The item %s doesn't exist" % self.key)
        return self.key

    def resolve_current_price(self, info):
        return ITEM[self.key]["price"]

    def resolve_upload_date(self, info):
        return ITEM[self.key]["date"]

    def resolve_product(self, info):
        return Product(key=ITEM[self.key]["product"])
```

```python
ITEM = {
    "i01" : {
        "product": "p01",
        "price": "2000",
        "date": datetime(2020, 6, 17)
    },
    "i02" : {
        "product": "p02",
        "price": "500",
        "date": datetime(2020, 6, 1)
    }
}
```

The Ecommerce Example: Merchant Type

```python
class Merchant(ObjectType):
    key = String(required=True)
    name = String()
    items = List(Item)

    def resolve_key(self, info):
        if self.key not in MERCHANT:
            return GraphQLError("The merchant %s doesn't exist" % self.key)
        return self.key

    def resolve_name(self, info):
        return MERCHANT[self.key]["name"]

    def resolve_items(self, info):
        return [Item(key=p_code) for p_code in MERCHANT[self.key]["items"]]
```

```python
MERCHANT = {
    "m01": {
        "name": "El Bazar",
        "items": ["i01", "i02"]
    }
}
```

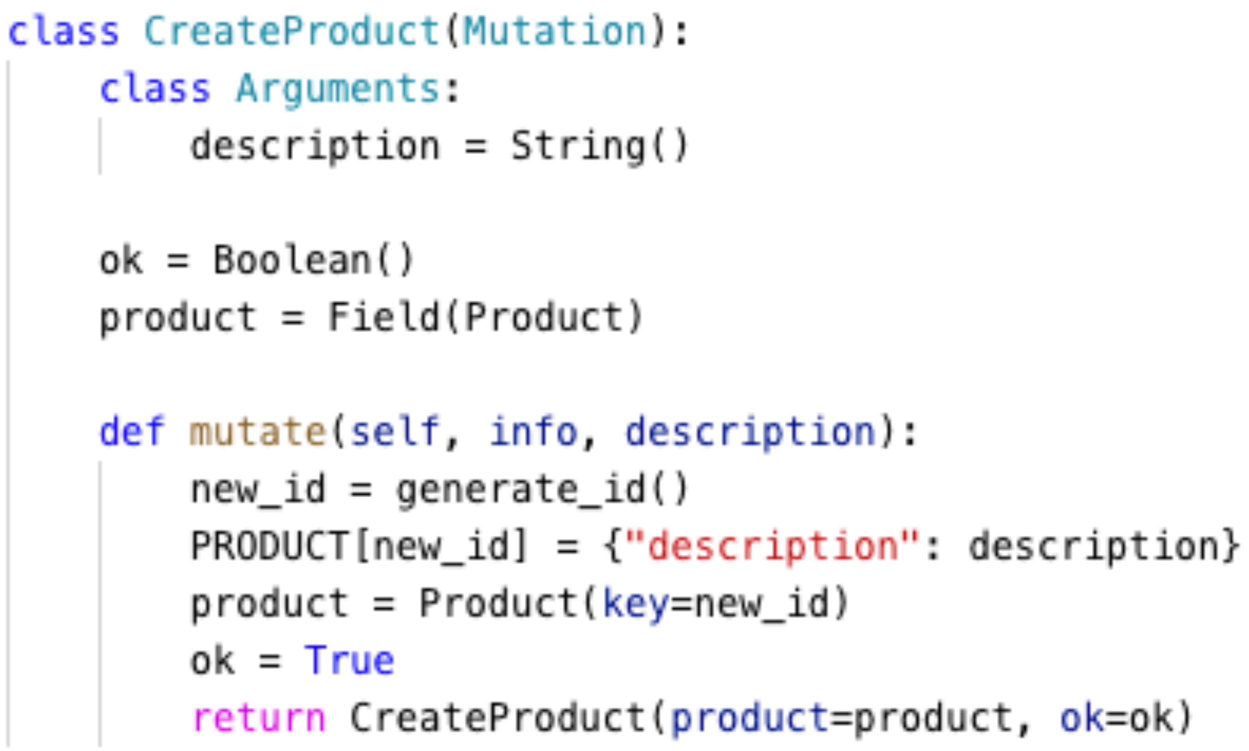
The Ecommerce Example: Mutation

```
mutation EcommerceExample {
  createProduct (description: "A new product"){
    product{
      key
    }
    ok
  }
}
```

$\longleftrightarrow$

```
{
  "data": {
    "createProduct": {
      "product": {
        "key": "p03"
      },
      "ok": true
    }
  }
}
```

The Ecommerce Example: Creating a Product



```python
class CreateProduct(Mutation):
    class Arguments:
        description = String()

    ok = Boolean()
    product = Field(Product)

    def mutate(self, info, description):
        new_id = generate_id()
        PRODUCT[new_id] = {"description": description}
        product = Product(key=new_id)
        ok = True
        return CreateProduct(product=product, ok=ok)
```