THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

6. Web Apps

CSCI 2541 Database Systems & Team Projects

Gabe - 2022

Based on slides by Wood

Timeline

Today
- Data in web apps
- Shopping Cart Mini Project overview – start this week!
  - Project assigned *long before* this is due
- ER diagram practice
- Lab: Flask sessions

Monday
- Midterm review

Wednesday - Midterm
- Both periods! Double the fun!
  - Multiple choice & flask/DB programming
- Requires laptop
- DSS: both lecture and lab!

Web App Data

Databases are great for storing permanent data!

How else can we keep data in our application?

Simplest example: Let's make a website that stores a counter of how many times the page has been loaded

SQL counter

```
DROP TABLE IF EXISTS mydata;
CREATE TABLE mydata (
  key       varchar(32) not null PRIMARY KEY,
  value      integer not null
);


INSERT INTO mydata (key, value) VALUES ("x", 1);
```

```
from flask import Flask
# Import the sqlite3 module
import sqlite3
app = Flask('app')



@app.route('/')
def hello_world():
  # open a new connection each time route is loaded
  connection = sqlite3.connect("myDatabase.db")
  connection.row_factory = sqlite3.Row
  cursor = connection.cursor()
  cursor.execute("SELECT * FROM myData where key = ?", ("x",))
  data = cursor.fetchone()
  x = data["value"]
  x += 1
  cursor.execute("UPDATE myData SET value = ? WHERE key = ?", (x, "x",))
  connection.commit()
  connection.close()
  return f'x ={x}'

app.run(host='0.0.0.0', port=8080)
```

Python counter

```
from flask import Flask
from datetime import datetime
app = Flask('app')

x = 1

@app.route('/')
def hello_world():
  global x
  x += 1
 return f"x = {x}"

app.run(host='0.0.0.0', port=8080)
```

Much simpler… but also much less powerful!

How does this differ from the DB case?

Data Storage

Database

- Persistent, reliable storage
- Both storage and data analysis
- Data schema enforces consistency

Application data

- Only available at runtime; lost if app crashes
- Flexible/simple, but can get messy
- Global across all users

Session data (more in lab!)

- Data specific to a single user, may be kept client side
- Otherwise similar to Application data

Shopping Cart

The website should be able to display products being sold in several categories. A user visiting your web store can search for products (i.e., search for a specific item name and display that item) or display all items in a certain category. The website should display the available quantity for each product.

Only a logged in user can add products to a shopping cart and then checkout to complete a purchase and buy the products. To "buy" a product means to reduce the quantity from that product with the quantity that was "bought" (i.e. your database should be updated to reflect the reduction in quantity of items after checkout, not when added to the cart).

A logged in user's shopping cart can be viewed, edited, checked out or deleted. A logged in user can also see her order history including the products and total cost of the order.

What do you need to do?

Design a database schema

Insert some dummy data

Think about routes you will need

Make templates to display categories and products

Use session data to store products in cart

Update DB on checkout

Check for edge cases / consistency!

Flask routes?

The website should be able to display products being sold in several categories. A user visiting your web store can search for products (i.e., search for a specific item name and display that item) or display all items in a certain category. The website should display the available quantity for each product.

Only a logged in user can add products to a shopping cart and then checkout to complete a purchase and buy the products. To "buy" a product means to reduce the quantity from that product with the quantity that was "bought" (i.e. your database should be updated to reflect the reduction in quantity of items after checkout, not when added to the cart).

A logged in user’s shopping cart can be viewed, edited, checked out or deleted. A logged in user can also see her order history including the products and total cost of the order.

Flask routes

homepage
- login
- cart/checkout
- categories

Categories
- links to products of that category

Profile
- past orders

Cart
- contents + optional deletion + total $

Products - list

Flask routes

Potential solutions:

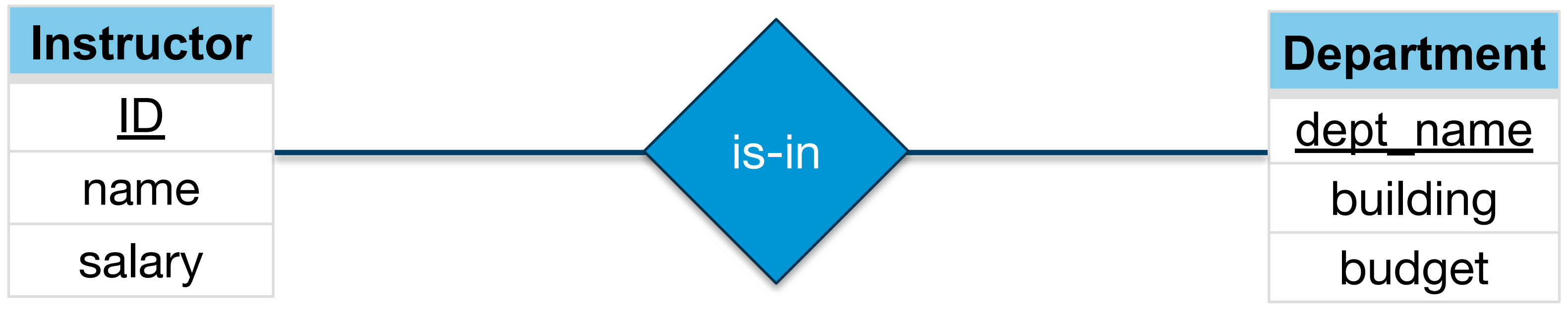
```
/categories
/products
/category/<catid>
/product/<prodid>

/login
/cart
/checkout
/orders
```

ER Diagram Syntax

Entities, Relationships, Attributes

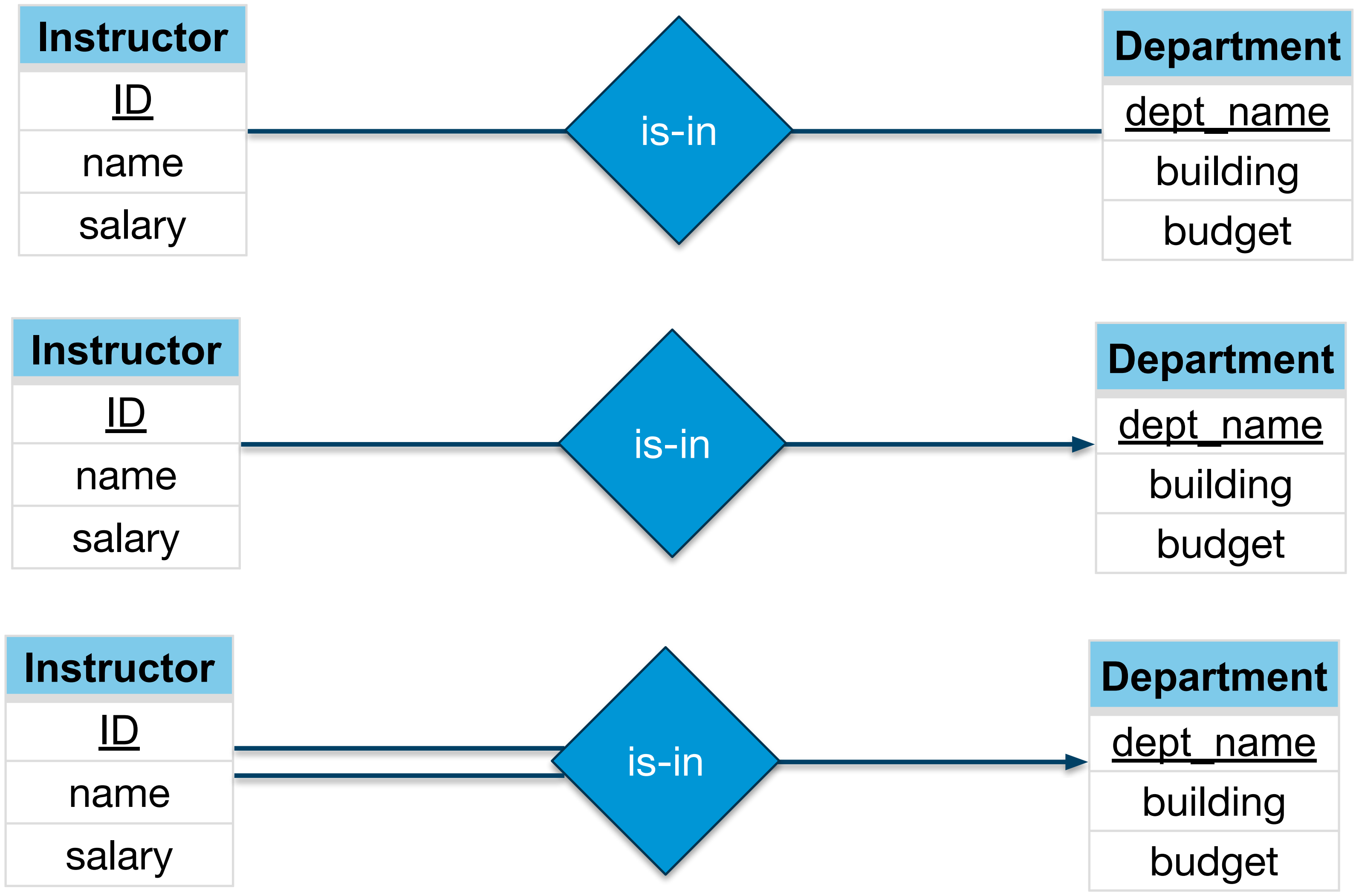
- Cardinality (arrows) and participation (single/double line)



ER Diagram Syntax

Entities, Relationships, Attributes

- Cardinality (arrows) and participation (single/double line)

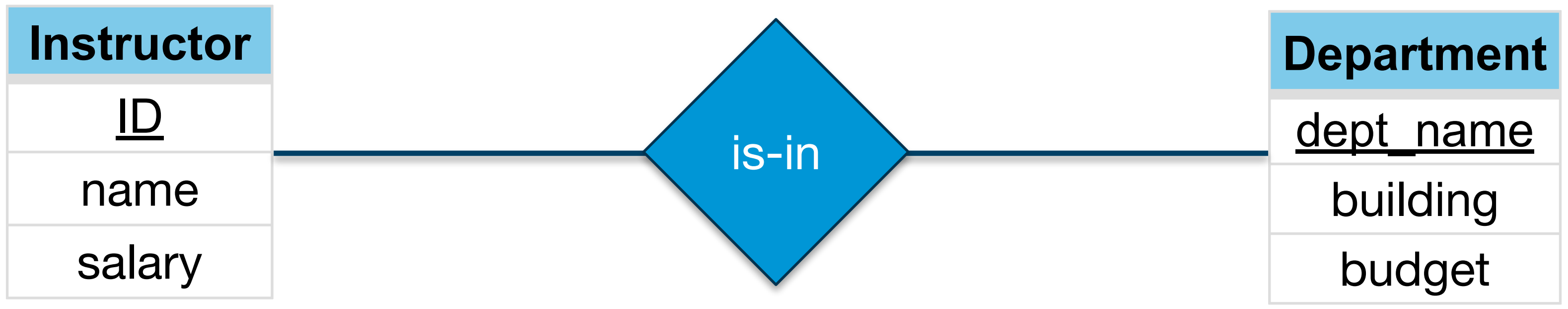


ER Diagram Syntax

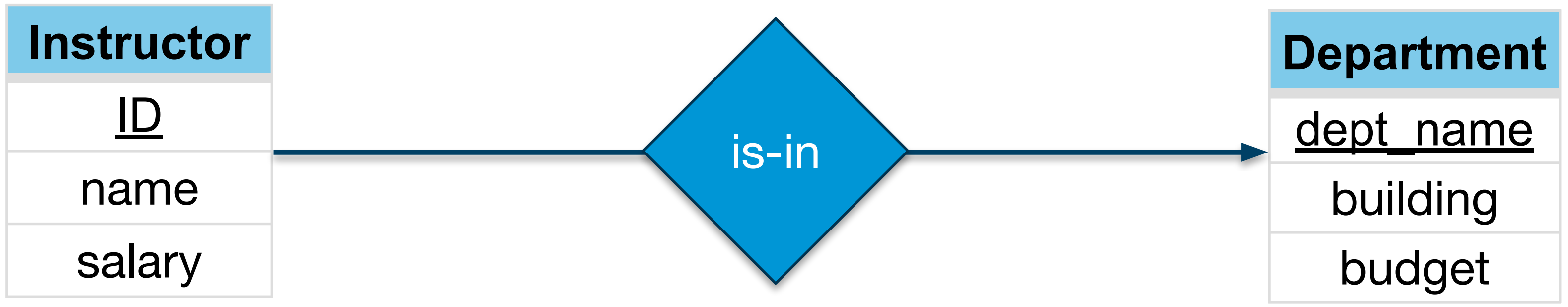
Entities, Relationships, Attributes

- Cardinality (arrows) and participation (single/double line)

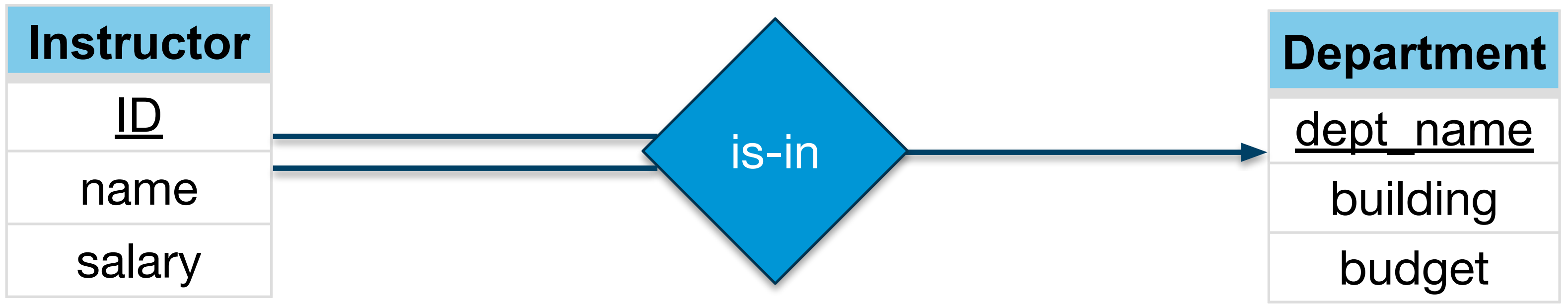
many-to-many, optionally related



many-to-one, optionally related



many-to-one, instructors must be in dept

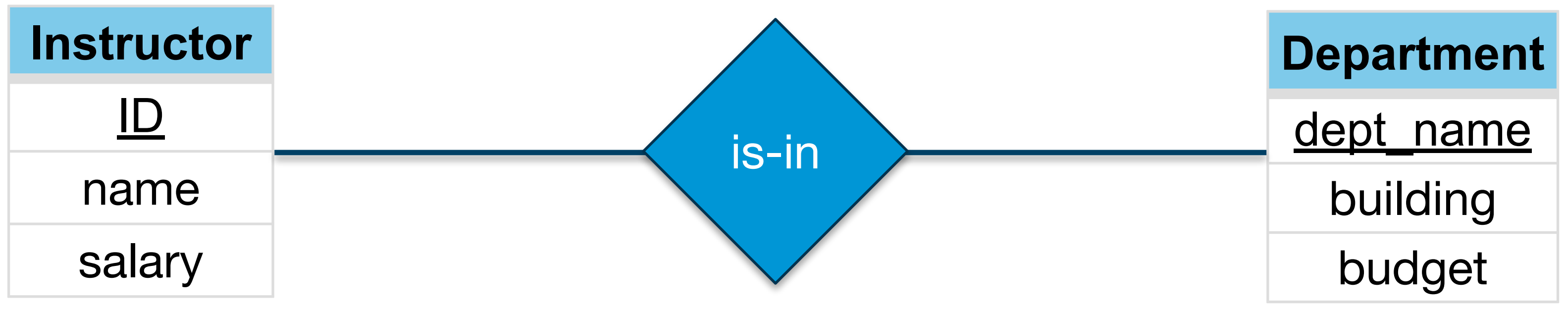


ER Diagram Syntax

Entities, Relationships, Attributes

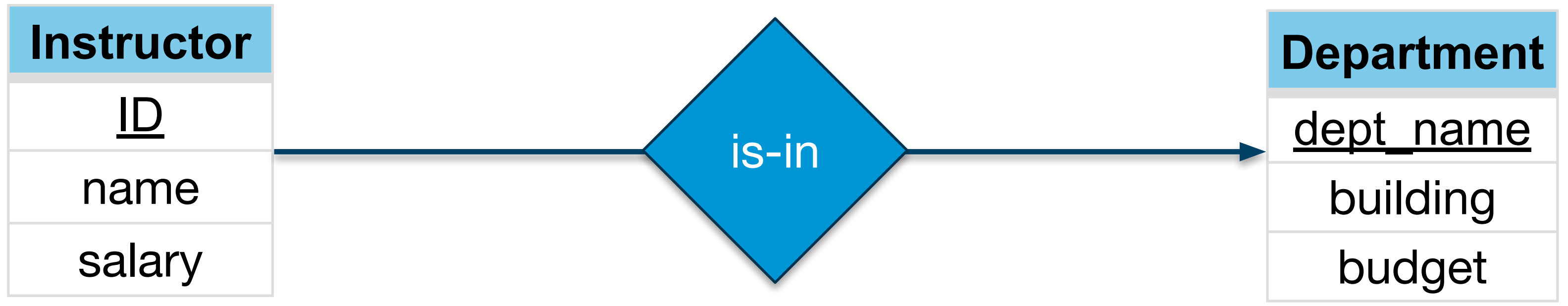
- Cardinality (arrows) and participation (single/double line)

many-to-many, optionally related



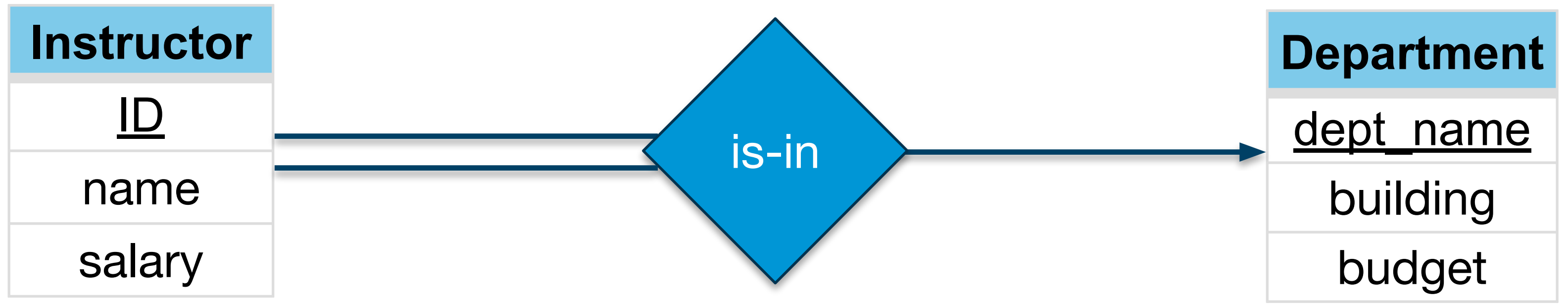
SQL: 3 relations

many-to-one, optionally related



SQL: foreign key in instructor, merge is-in into instructor

many-to-one, instructors must be in dept



SQL: **not null** on foreign key to dept

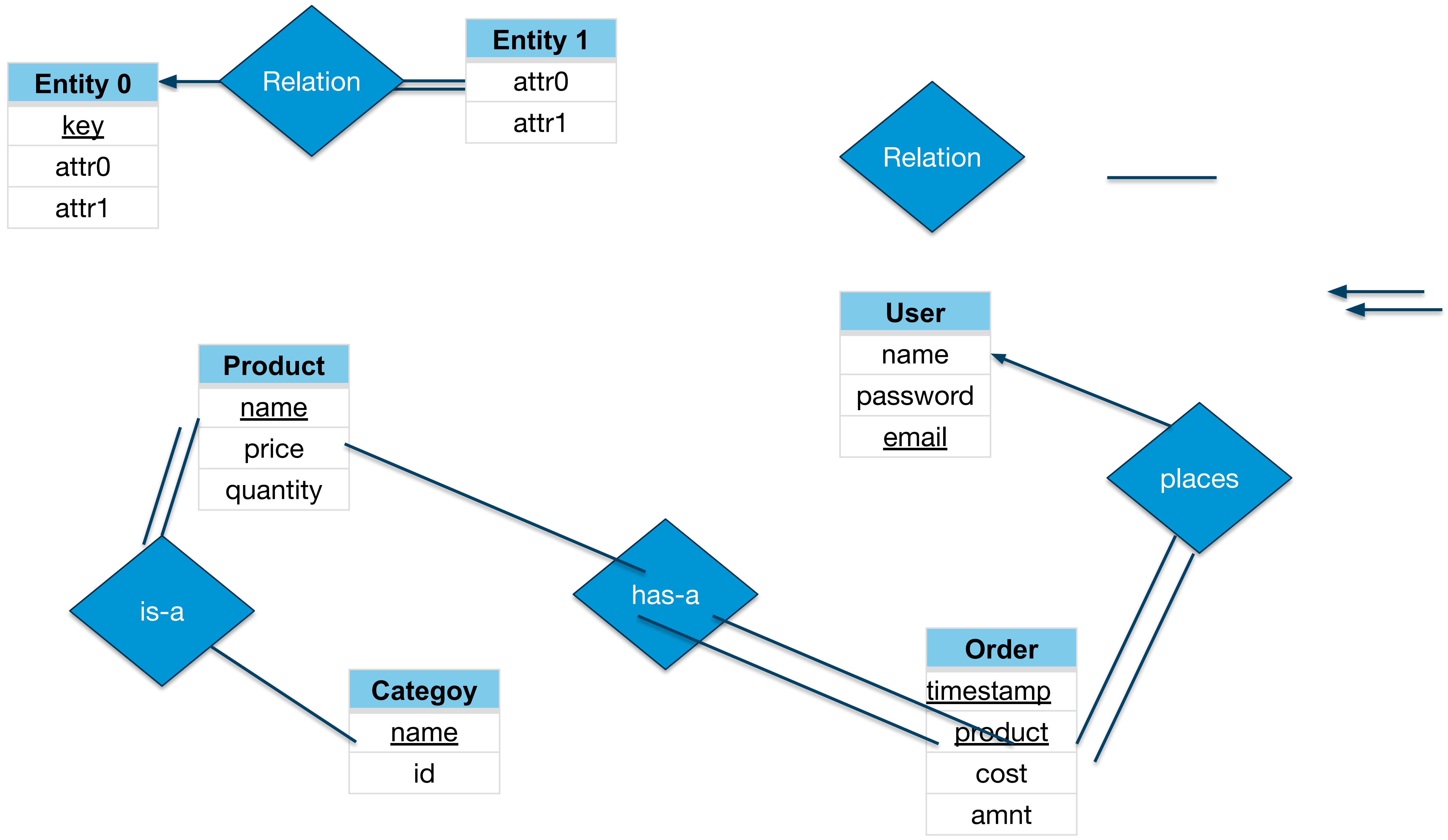
ER Diagram?

The website should be able to display products being sold in several categories. A user visiting your web store can search for products (i.e., search for a specific item name and display that item) or display all items in a certain category. The website should display the available quantity for each product.

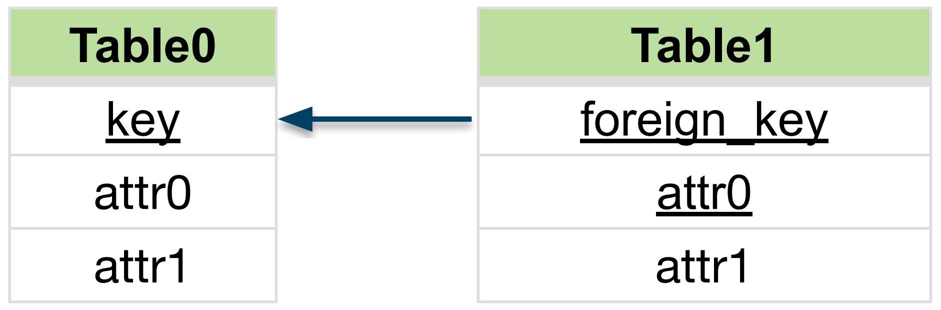
Only a logged in user can add products to a shopping cart and then checkout to complete a purchase and buy the products. To "buy" a product means to reduce the quantity from that product with the quantity that was "bought" (i.e. your database should be updated to reflect the reduction in quantity of items after checkout, not when added to the cart).

A logged in user's shopping cart can be viewed, edited, checked out or deleted. A logged in user can also see her order history including the products and total cost of the order.

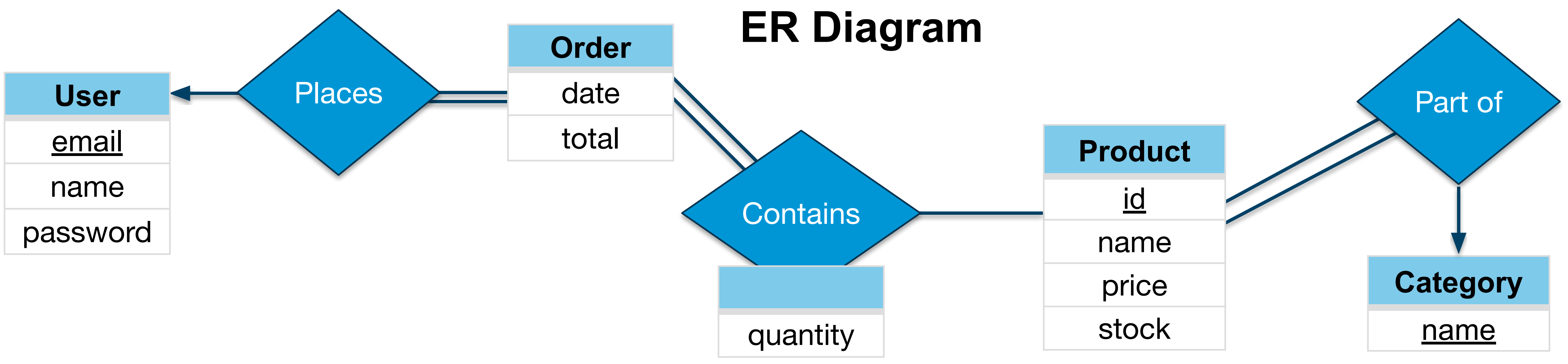
Shopping Cart ER Diagram



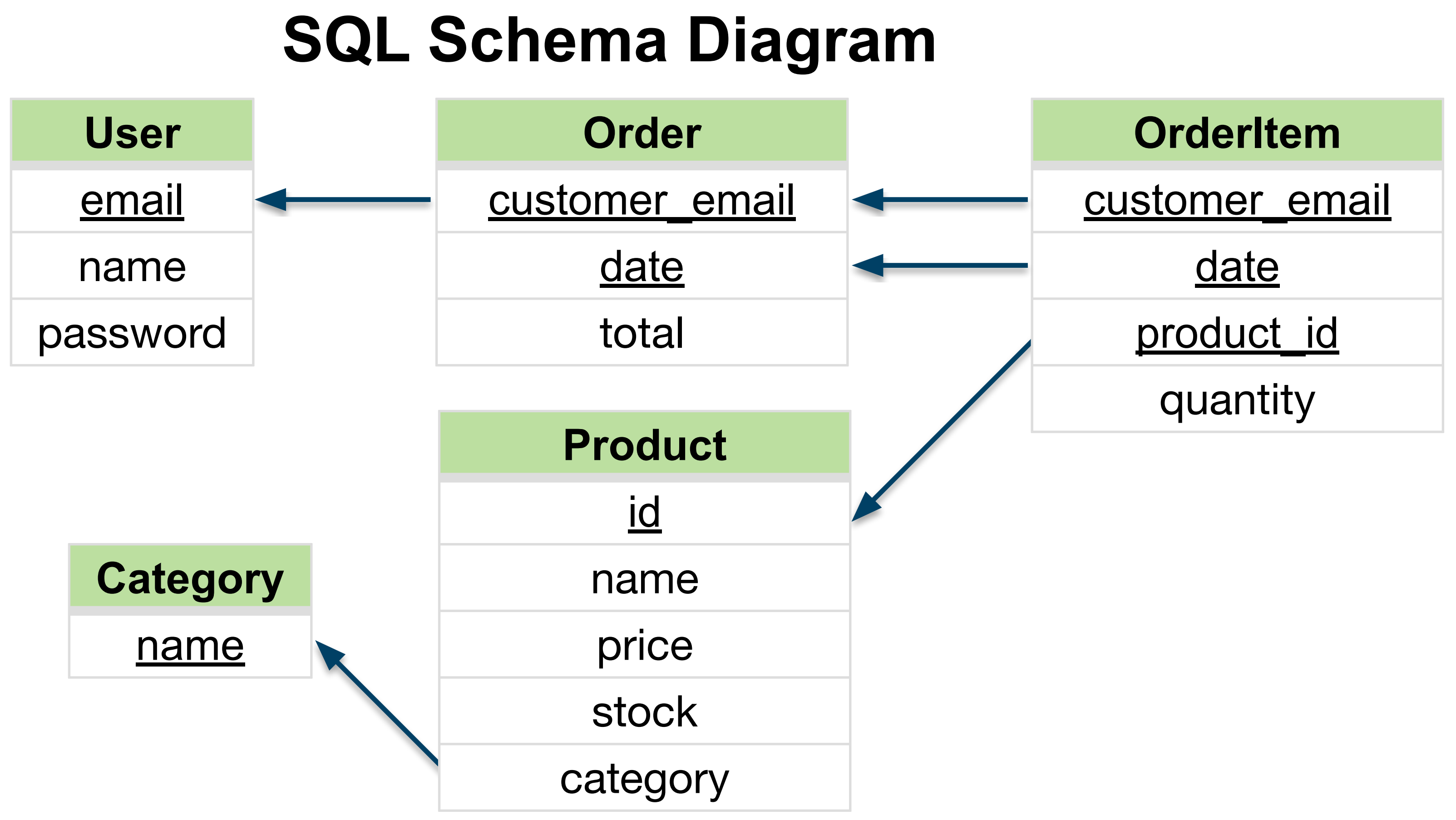
Shopping Cart SQL Diagram



Shopping Cart Diagrams



ER Diagram
User
email
name
password
Places
Order
date
total
Contains
quantity
Product
id
name
price
stock
Part of
Category
name



SQL Schema Diagram
User
email
name
password
Order
customer_email
date
total
OrderItem
customer_email
date
product_id
quantity
Product
id
name
price
stock
category
Category
name