# EECS-317 Data Management and Information Processing

## Lecture 1 – Course Logistics & Modeling Data with Tables

Steve Tarzia

Spring 2019

Northwestern

# What is/are Data?

# EECS-317 in a nutshell

- Learn how to handle real-world, *complex, messy* data with **SQL relational databases:**
  - A powerful foundational technology
  - Like a filesystem, but better
    - (easy queries, indexing, concurrency, crash tolerance)
- Roughly speaking "Data Science" is:
  - Data management (*this course!)*
  - Statistics (eg., IEMS-304 Statistical Methods for Data Mining, or EECS-349)
  - Visualization (eg., PSYCH-245 Presenting Ideas and Data)

You'll learn to answer questions (about the past) using complex data sets

# Data are not just numbers

- "Simple" data sets are just arrays or matrices of numbers:
  - Time-series of stock price data
  - matrix of pixel colors in an image
  - 3D "matrix" of atmospheric temperatures in a weather simulation.

- Complex data also represent *relationships*
  - For example, the course scheduling information at Northwestern
  - It's not just a sequence of numbers.
  - It's is a complex web of students, professors, courses, classrooms, grades, etc.
  - This course will teach you how to handle such data.

# Things you *cannot* do with Excel and Matlab

- Model complex data relationships
  - Spreadsheets and matrices are very limiting formats
  - Every row has a fixed number of attributes (columns)
  - Can't model one-to-many and many-to-many relationships
  - You can try using multiple spreadsheet tabs or multiple matrices for different types of data, but linking them is difficult
- Enforce data integrity constraints
  - Spreadsheet cells can have all kinds of weird data
  - Matlab matrices cannot easily handle anything other than numbers
- Keep data and analysis separate

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | | | | | | | |
| 2 | | | | | | | |
| 3 | **Company Name** | **Invoice Date** | **Delivery Date** | **Amounts** | | | |
| 4 | Jenny | 01.09.2007 | 1900/01/00 | 2057 | | | |
| 5 | | 01.11.2007 | 1900/01/00 | 2669 | | | |
| 6 | **Jenny Total** | | | **2669** | | | |
| 7 | Sam | 1900/01/01 | 1900/01/00 | 1426 | | | |
| 8 | | 1998/01/01 | 01.01.1998 | 1185 | | | |
| 9 | | 1998/01/01 | 1900/01/00 | 2359 | | | |
| 10 | | 1998/01/01 | 01.06.1998 | 1886 | | | |
| 11 | | 1998/01/01 | 1900/01/00 | 2359 | | | |
| 12 | | 2000/07/01 | 01.07.2000 | 2486 | | | |
| 13 | **Sam Total** | | | **9342** | | | |
| 14 | | | | | | | |
| 15 | | | | **Page 1** | | | |
| 16 | | | | | | | |
| 17 | Peter | 2000/01/01 | 1900/01/00 | 2385 | | | |
| 18 | | 1975/04/01 | 1900/01/00 | 0 | | | |
| 19 | | 2000/04/01 | 1900/01/00 | 0,000 | | | |
| 20 | | 2005/06/01 | 1900/01/00 | 7 293.07 | | | |
| 21 | | 1993/07/01 | 1900/01/00 | 42 717.42 | | | |
| 22 | | 1993/07/01 | 01.07.1993 | 55 872.63 | | | |
| 23 | | 01.08.2000 | 1900/01/00 | 40 176.80 | | | |
| 24 | | 01.09.2000 | 1900/01/00 | 1585 | | | |
| 25 | | 01.10.2001 | 1900/01/00 | 1384 | | | |
| 26 | | 01.10.2004 | 01.10.2004 | 01518 | | | |

Unwanted Rows

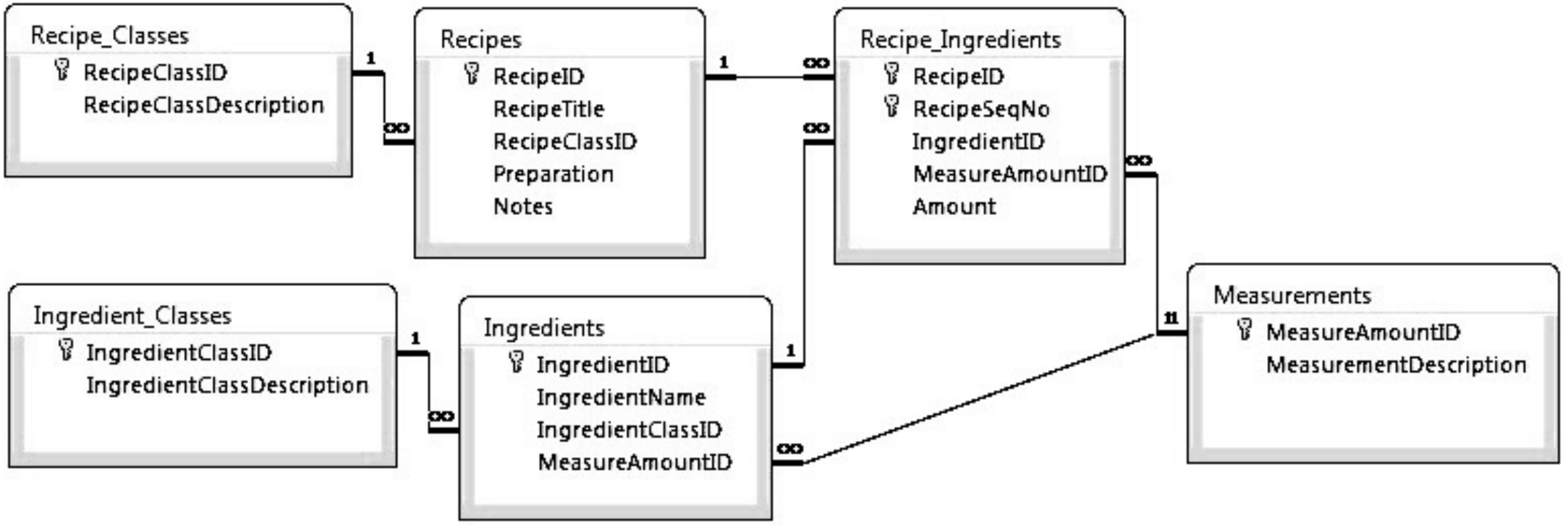
Gaps that need to be filled in

Unwanted Rows

Numbers not working

Dates not working

# SQL database example




- This is the data *schema* – how it's organized, not the recipe data itself.
- First design the structure of the data, then fill it in.

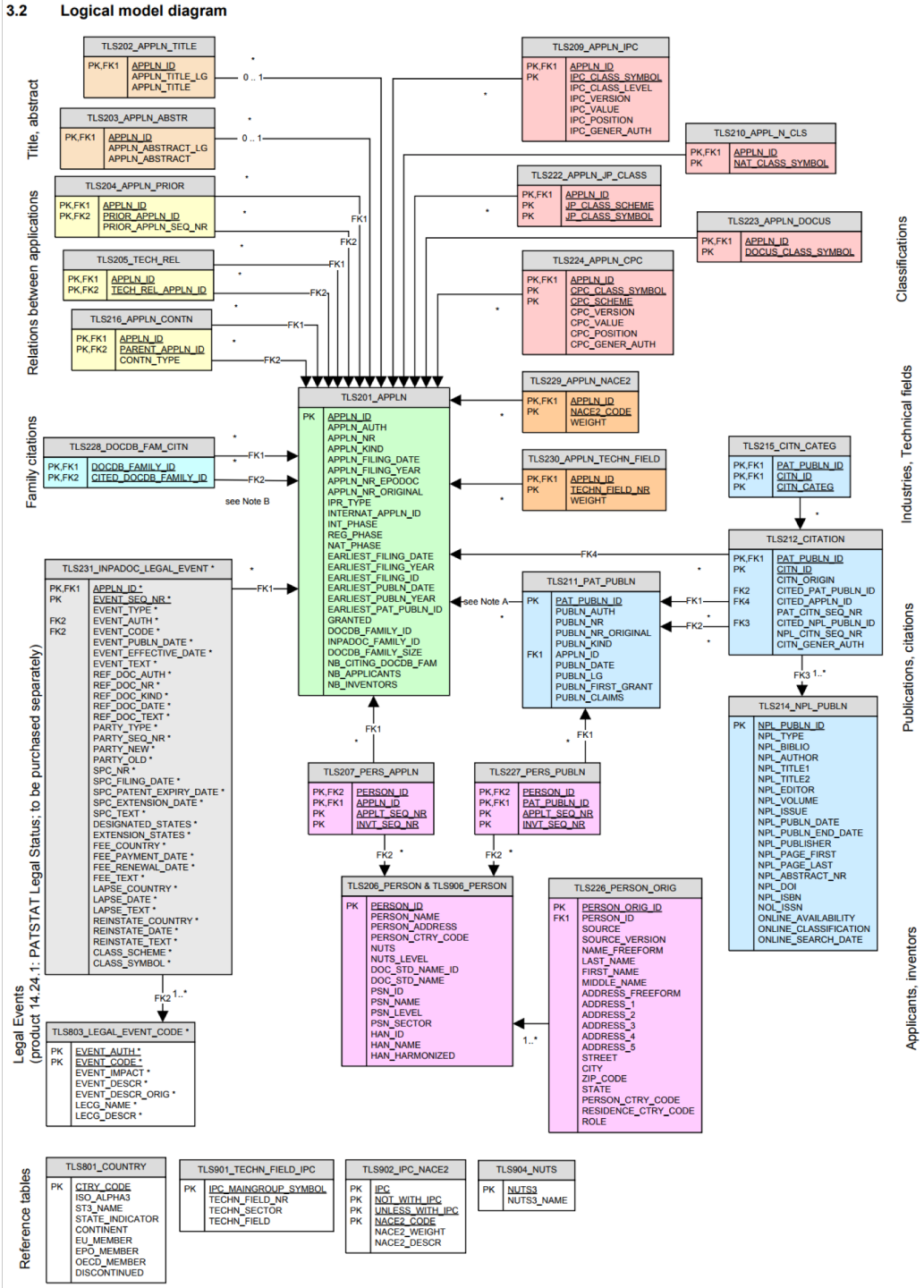
# Questions to be answered from the recipe DB

- How many steps are in the “Chocolate chip cookie” recipe?
- What are the titles of the recipes that have seafood ingredients?
- Do any recipes use the same ingredient twice?
- Which recipe has the greatest number of steps?
- Etc.

# PATSTAT: European Patent Office's International Patent Database

- 29 cross-referenced tables
- 6 DVDs of data
- 119GB of CSV files after unzipping

- This example has both complex structure and lots of data entries.



# Difficulties in plain Python, R, C++, Java, etc.

- Working with data that is larger than the computer's RAM (*scalability*)
- Keeping your data around after your program finishes (*persistence*)
- Efficiently searching through lots of data (*indexing*)
- Easily filtering and summarizing data (*querying*)
- Sharing data between multiple applications (*concurrency*)

*Computation* and *data management* typically use different tools.

- Many systems use both SQL *and* a general-purpose language.

# The Goal: Easy & Clean Descriptive Analytics

Answer a wide variety of complex questions using the same database:

- Where did our 10 biggest customers live in 2007?

```
SELECT customer.name, customer.city, customer.province FROM
  customer JOIN order ON order.customer == customer.id
           JOIN order_item ON order_item.order == order.id
           JOIN product ON order_item.product = product.id
  WHERE order.placed >= “2007-01-01” AND order.placed < “2008-01-01”
  GROUP BY customer.id
  ORDER BY SUM(order_item.qty * product.price) DESC
  LIMIT 10;
```

This is code in the SQL language.

- How many widgets are left in stock?
- What is the average price of the chairs we sell?

# Database Management Systems (DBMSs)

- A DBMS is a data management software that allows users to define databases, load them with data, and query them.
  - Eg., Oracle, MS SQL Server, MySQL, PostgreSQL, (SQLite, Access)
- Often run on a remote, multi-user server
  - Typically you need to know the hostname and have a username and password.
- May be connected to one or more software applications or may stand alone.
- Client libraries exist for every common programming language
  - But you usually query the database using the SQL language
- You purchase/download a *DBMS*, then use it to create your own *databases*.

# Course Outline

- SQL relational databases:
  - Structured Query Language (SQL)
    - Select, create table, update, delete
    - Joining tables
    - Subqueries & temporary tables
    - Indexes and execution plans
  - Data modelling
    - One-to-many, many-to-many relationships
    - Integrity & foreign key constraints
- Getting data from the real world:
  - Numeric formats
    - Binary, integers, floats, precision
    - Dates and times
  - Text encodings
    - ASCII, UTF-8, special characters
  - Organizing data in files
    - CSV, XML, JSON
  - Messy data
    - Missing entries, fuzzy matching
  - Regular expressions
  - Data APIs
  - Web scraping

# Prerequisites

- Very few.
- You should have done some programming in some language.
- I assume you have used spreadsheets.

**NOTE:**

- Computer Science and Computer Engineering students will not get credit for this course, except as an unrestricted elective.
  - They should take EECS-339 instead.

# Questions about course content?

# Course Logistics

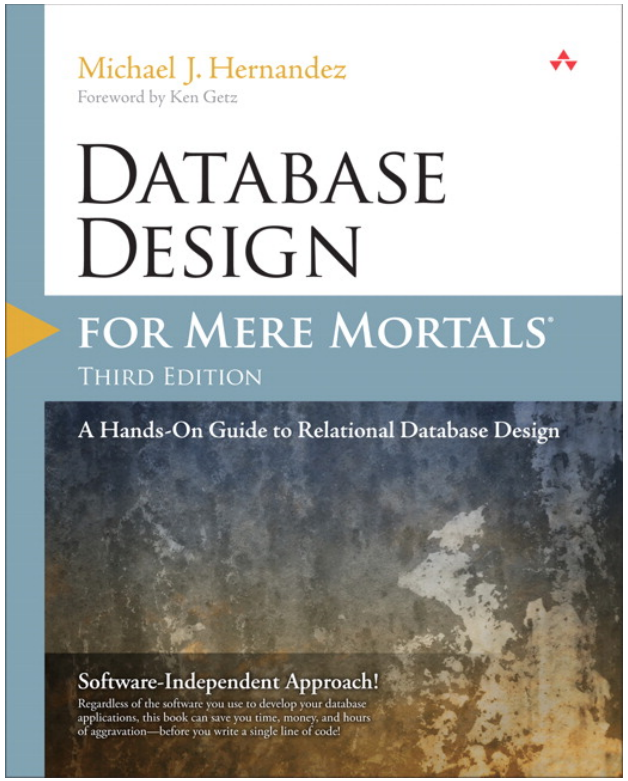
- All materials and HW submission will be on **Canvas**
- Lecture slides and videos will be posted (Panopto section in Canvas)
- Ask your questions on **Piazza** (not by email)
- TA is Panitan Wongse-Ammat
- Peer Mentors (like TAs):
  - Amanda Demopoulos, Moli Mesulam, Keren Zhou, Tianhao Zhang
- Exams will be *open book* and *open notes* (no sharing books or papers)
  - Midterm Exam is Thursday May 2nd during class.
  - Final Exam is Thurs June 6th or Friday June 14th?

# Office hours

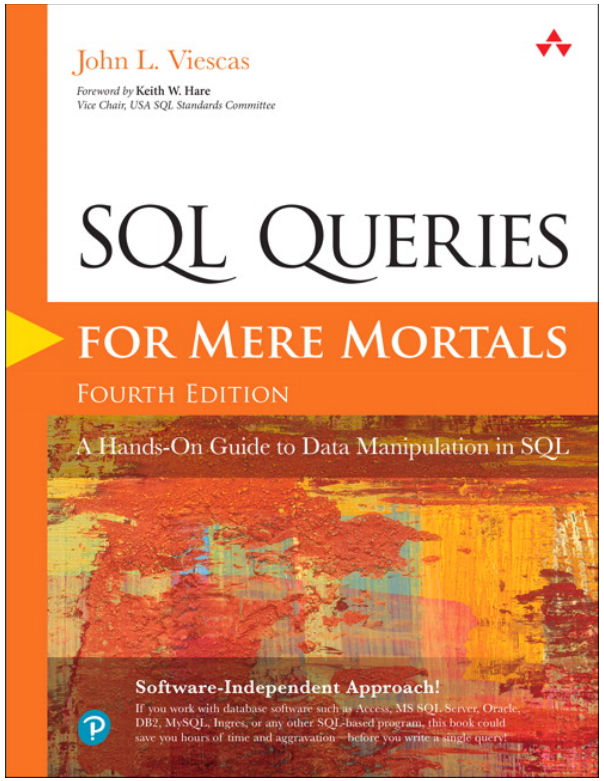
- TA/PM office hours in Mudd 3303:
  - Monday: 1:30-3:30pm
  - Tuesday: 12-2pm
  - Wednesday: 1:30-5pm
- Prof. Tarzia's office hours (in Mudd 3225):
  - Mon 1-3pm, Tues 3:30-4:30pm,
    Wed 3-5pm, Thurs 3:30-4:30pm

# Optional Reference Books

Hernandez “Database Design for Mere Mortals” ($30 on Amazon)




Viescas & Hernandez “SQL Queries for Mere Mortals” ($32 on Amazon)



# Grading

- Homework ($6 \times 6.67\% = 40\%$), including a final project.
- Midterm exam (25%)
- Final exam (35%)

# Tentative Homework

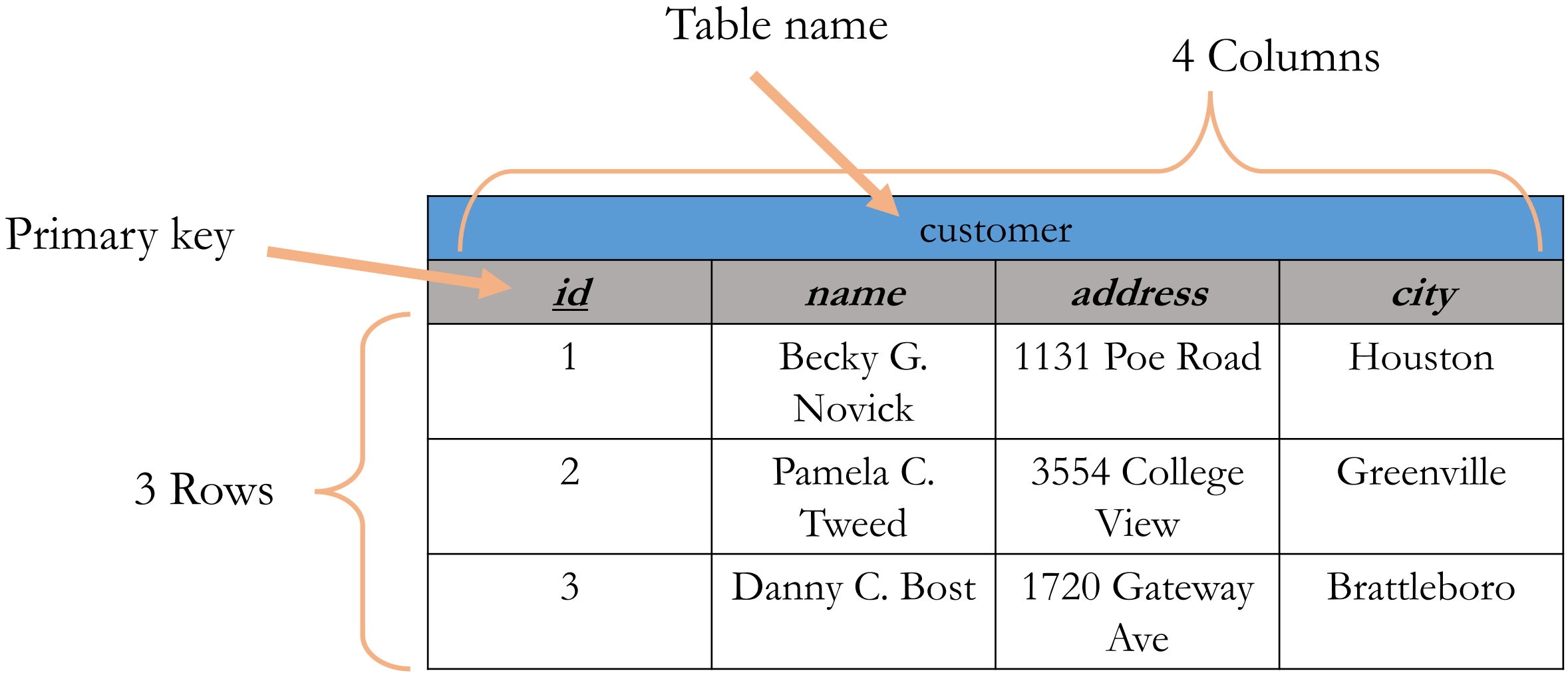
- HW 1, 2, 3: SQL queries
  - Write SQL queries to perform analytics on a small, local database (SQLite).
  - Wire more advanced queries.
  - Connect to a large, remote database.
- HW 4: Getting data
  - Numeric types, regular expressions
- HW 5, 6: Database design project
  - Design a data model from scratch to model a data domain of your choice.
  - Load data & perform queries.

# Questions about logistics?

# Why use a relational database?

- **Scalability** – work with data larger than computer's RAM
- **Persistence** – keep data around after your program finishes
- **Indexing** – efficiently sort & search along various dimensions
- **Integrity** – restrict data type, disallow duplicate entries
- **Deduplication** – save space, keep common data consistent
- **Concurrency** – multiple users or applications can read/write
- **Security** – different users can have access to specific data
- **"Researchability"** – SQL allows you to concisely express analysis

# **Tables** are the main concept in relational DBs




| customer | | | |
|---|---|---|---|
| ***id*** | ***name*** | ***address*** | ***city*** |
| 1 | Becky G. Novick | 1131 Poe Road | Houston |
| 2 | Pamela C. Tweed | 3554 College View | Greenville |
| 3 | Danny C. Bost | 1720 Gateway Ave | Brattleboro |

# DB design process answers these questions:

- What tables do we need?
  - How to logically separate the data?
- What columns?
  - Data types for columns?
  - How will rows be uniquely identified?
  - Are some columns optional?
- How will tables be linked?

| customer | | | |
|---|---|---|---|
| ***id*** | ***name*** | ***address*** | ***city*** |
| 1 | Becky G. Novick | 1131 Poe Road | Houston |
| 2 | Pamela C. Tweed | 3554 College View | Greenville |
| 3 | Danny C. Bost | 1720 Gateway Ave | Brattleboro |

# Sometimes we start with one redundant table and break it down to reflect the logical components

| staff | | | | | |
|---|---|---|---|---|---|
| ***id*** | ***name*** | ***department*** | ***building*** | ***room*** | ***faxNumber*** |
| 11 | Bob | Industrial Eng. | Tech | 100 | 1-1000 |
| 20 | Betsy | Computer Sci. | Ford | 100 | 1-5003 |
| 21 | Fran | Industrial Eng. | Tech | 101 | 1-1000 |
| 22 | Frank | Chemistry | Tech | 102 | 1-1000 |
| 35 | Sarah | Physics | Mudd | 200 | 1-2005 |
| 40 | Sam | Materials Sci. | Cook | 10 | 1-3004 |
| 54 | Pat | Computer Sci. | Ford | 102 | 1-5003 |

# This is called *Normalization*

| staff | | |
|---|---|---|
| ***id*** | ***name*** | ***department*** |
| 11 | Bob | 1 |
| 20 | Betsy | 2 |
| 21 | Fran | 1 |
| 22 | Frank | 4 |
| 35 | Sarah | 5 |
| 40 | Sam | 7 |
| 54 | Pat | 2 |

| department | | |
|---|---|---|
| ***id*** | ***name*** | ***building*** |
| 1 | Industrial Eng. | 1 |
| 2 | Computer Sci. | 2 |
| 4 | Chemistry | 1 |
| 5 | Physics | 4 |
| 7 | Materials Sci. | 5 |

| building | | |
|---|---|---|
| ***id*** | ***name*** | ***faxNumber*** |
| 1 | Tech | 1-1000 |
| 2 | Ford | 1-5003 |
| 4 | Mudd | 1-2005 |
| 5 | Cook | 1-3004 |
| 6 | Garage | 1-6001 |

- Removes redundancy
  - Save space
  - Edit values in one place, so duplicates don't become inconsistent
- Tables can be populated separately
- **But**, you are adding a new *id* column for each table

# Tables

- Represent objects, events, or relationships
  - Each of its rows must be uniquely identifiable
  - Has attributes that the DB will store in columns
  - Can refer to rows in other tables
- *Objects*: people, places, or things
- *Events*: usually associated with a specific time. Can recur.
- *Relationships*: associations

Designing a set of tables is called *data modelling*, and it's best learned by example.

# Database **Schema** defines the data's structure

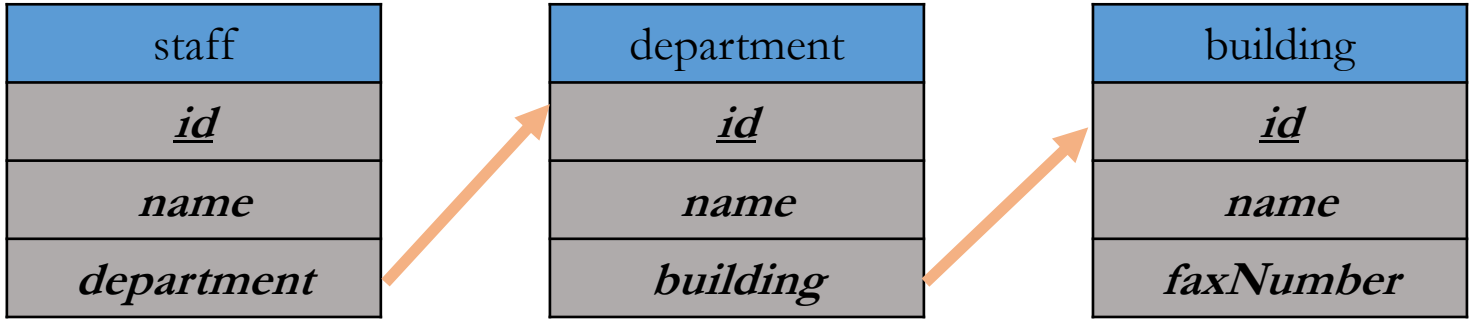
- Also called a data model
- It's *metadata* – data about data
- Defines the tables, including:
  - Columns in each table (both the name and *type*)
  - Primary Key for each table
  - Foreign Keys that link tables

| staff | | | |
|---|---|---|---|
| ***id*** | ***name*** | ***room*** | ***depart-ment*** |
| 11 | Bob | 100 | 1 |
| 20 | Betsy | 100 | 2 |
| 21 | Fran | 101 | 1 |
| 22 | Frank | 102 | 4 |
| 35 | Sarah | 200 | 5 |
| 40 | Sam | 10 | 7 |
| 54 | Pat | 102 | 2 |

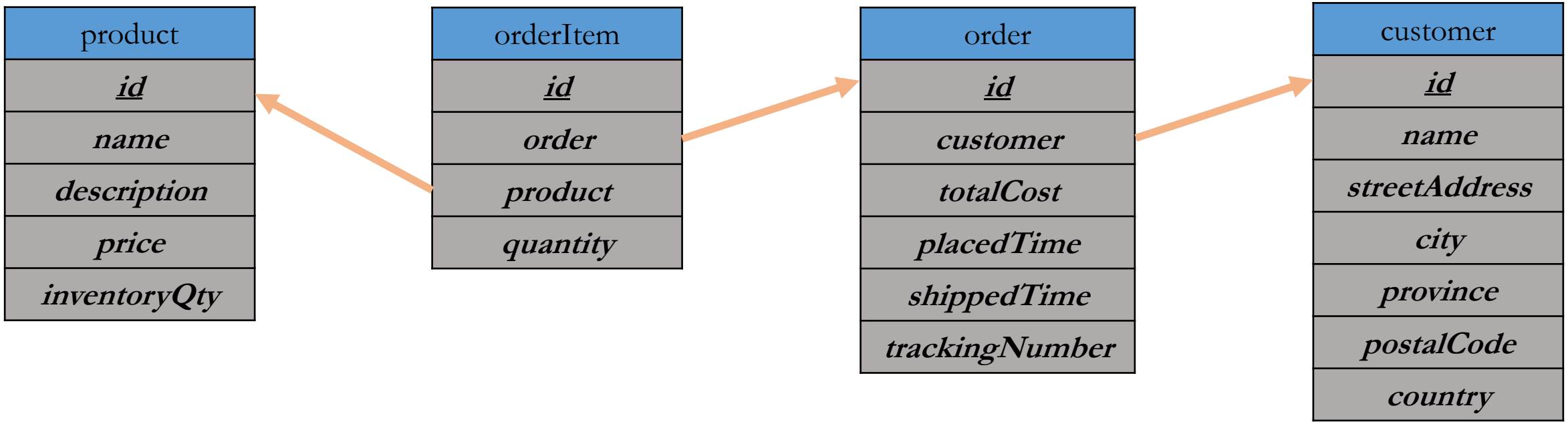
| department | | |
|---|---|---|
| ***id*** | ***name*** | ***building*** |
| 1 | Industrial Eng. | 1 |
| 2 | Computer Sci. | 2 |
| 4 | Chemistry | 1 |
| 5 | Physics | 4 |
| 7 | Materials Sci. | 5 |

| building | | |
|---|---|---|
| ***id*** | ***name*** | ***faxNumber*** |
| 1 | Tech | 1-1000 |
| 2 | Ford | 1-5003 |
| 4 | Mudd | 1-2005 |
| 5 | Cook | 1-3004 |
| 6 | Garage | 1-6001 |

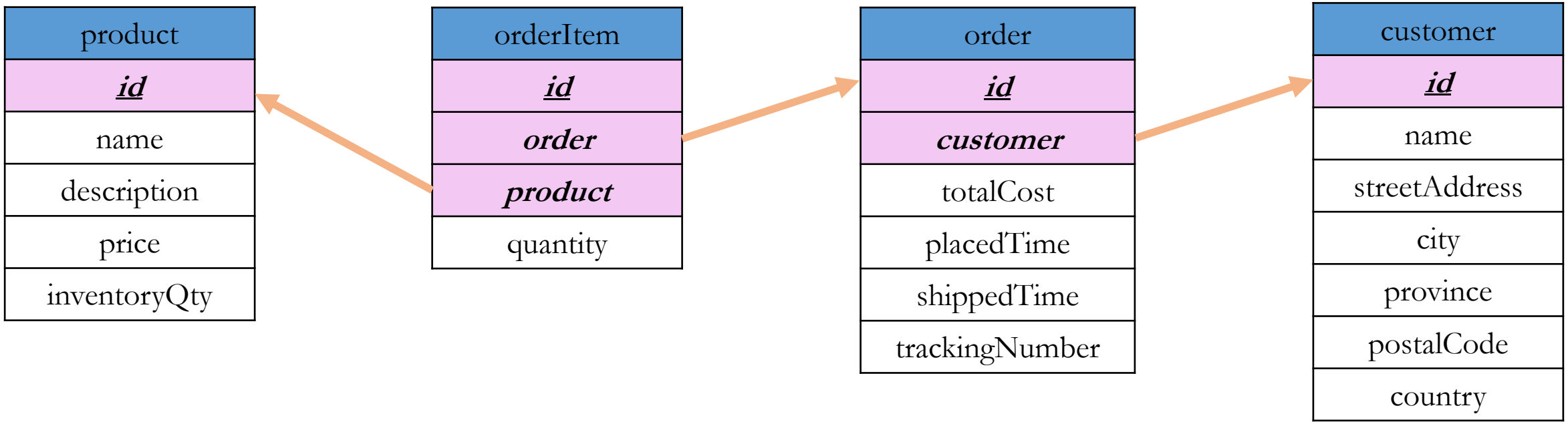
# DB Design diagram:



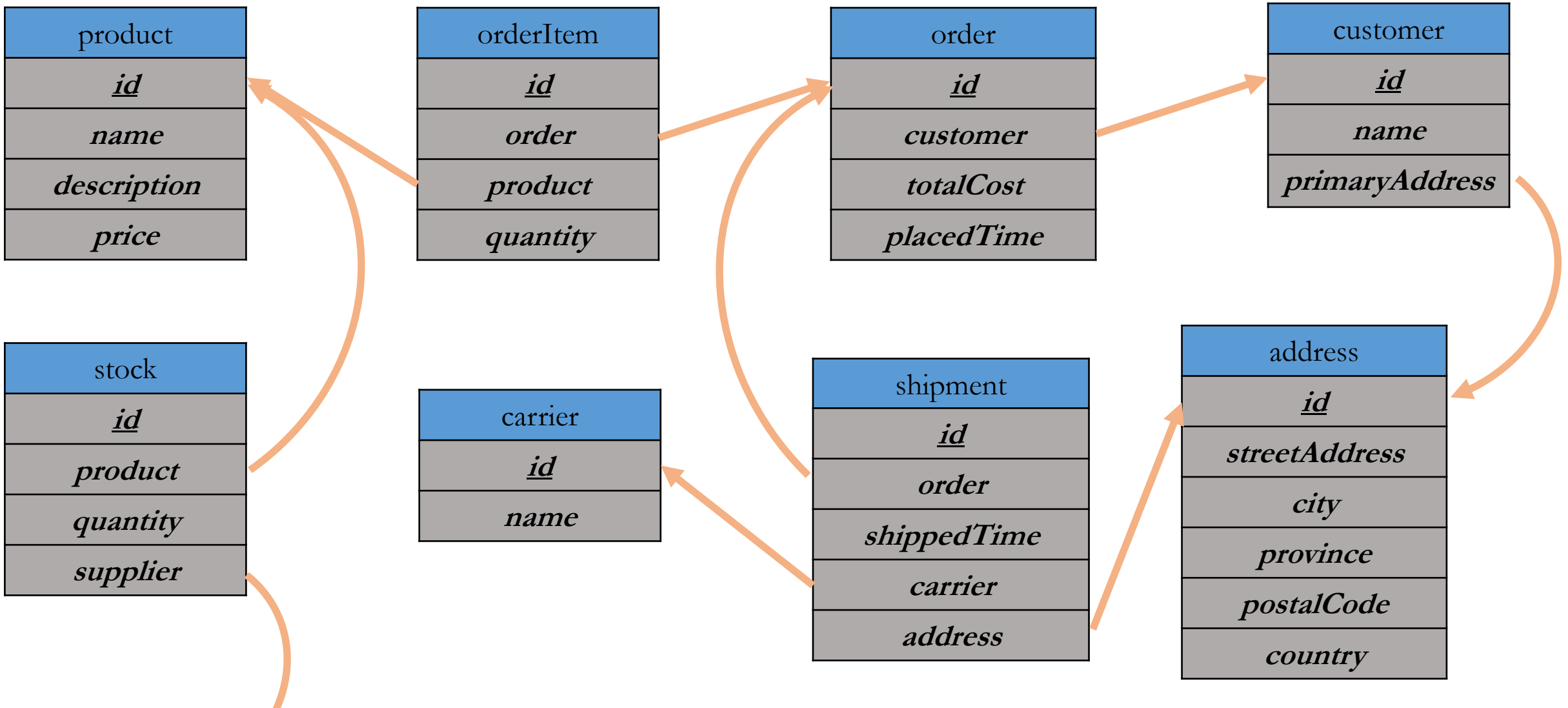
# Online retail example



# Some columns are just internal references

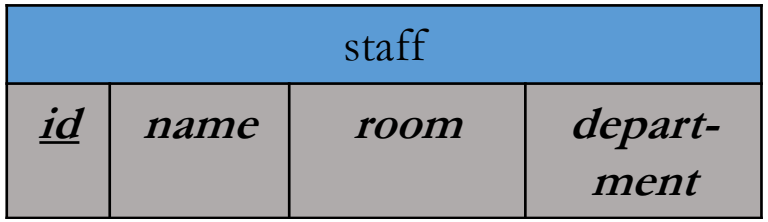


# Can make the model more complex



# Basic steps

- Create table:
  - Table has a name
  - Table has certain named & *typed* columns.
- Add rows to table
  - Each row gives exactly one value to each column (except optional columns can take a null or empty value in a row).
- Write queries to fetch data from the table.



# Recap

- *Complex data* are more than just streams of numbers!
- *Data model* or *schema* defines the data's structure
  - It's a list of *tables*, each with a fixed number of *columns*
  - Data *rows* are added after the data model is designed.
- These are called *Relational* or *SQL* databases.
- Can represent much more complex data than a simple spreadsheet.

## TODO:

- Maybe buy or print a SQL book.
- Download DB browser for SQLite http://sqlitebrowser.org/
  - Open and view the sample databases on Canvas (.sqlite files)