CMPINF 2110

Spring 2021

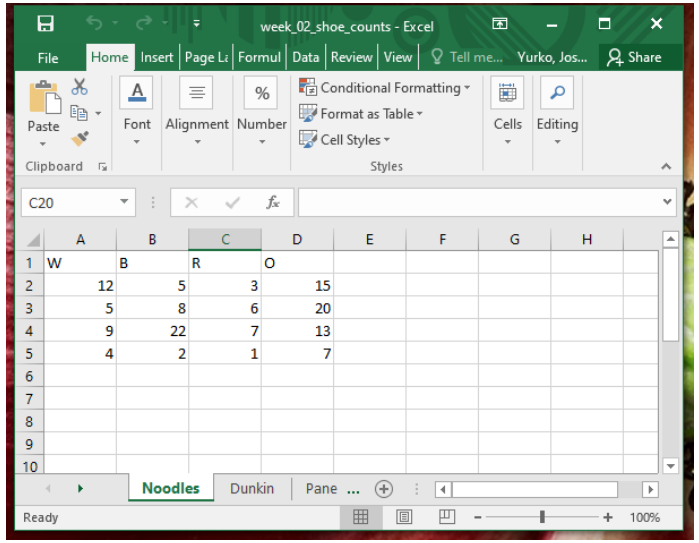
Week 06

Entity Relationship Diagrams and Database design

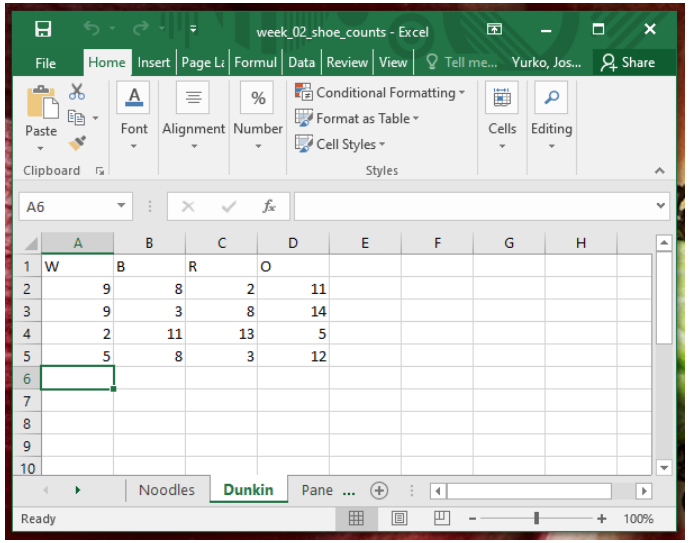
Up to this point we have informally discussed how to structure or design a database

- We have taken a “data first” approach, where we work with the data directly.
- We manipulated the data, reshaping and reorganizing the information.
- Perhaps the exercises seemed a bit confusing at first…let’s review what we did.

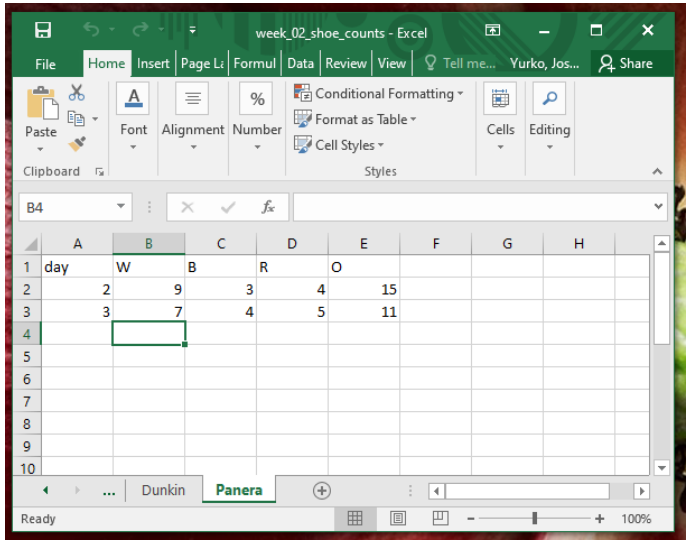
The shoe counting example, started with three separate spread sheets…



	A	B	C	D	E	F	G	H
1	W	B	R	O				
2	12	5	3	15				
3	5	8	6	20				
4	9	22	7	13				
5	4	2	1	7				
6								
7								
8								
9								
10								



	A	B	C	D	E	F	G	H
1	W	B	R	O				
2	9	8	2	11				
3	9	3	8	14				
4	2	11	13	5				
5	5	8	3	12				
6								
7								
8								
9								
10								



	A	B	C	D	E	F	G	H
1	day	W	B	R	O			
2	2	9	3	4	15			
3	3	7	4	5	11			
4								
5								
6								
7								
8								
9								
10								

Each spread sheet (tab in the Excel workbook) corresponded to a different location (store).

Was reorganized into a LONG-FORMAT “tidy” data set!

	day	value	location	shoe
0	1	12	N	W
1	2	5	N	W
2	3	9	N	W
3	4	4	N	W
4	1	5	N	B
5	2	8	N	B
6	3	22	N	B
7	4	2	N	B
8	1	3	N	R
9	2	6	N	R
10	3	7	N	R
11	4	1	N	R
12	1	15	N	O
13	2	20	N	O
14	3	13	N	O

⋮

1 row is one day’s count of the number of shoes of a specific color in a store!

We then “broke up” or decomposed the tidy long-format data set into smaller tables…

- We identified the unique locations or stores:

```
location_info = lf.groupby(['location']).size().reset_index(name='num_rows')
```

```
location_info
```

	location	num_rows
0	D	16
1	N	16
2	P	8

- And added more descriptive information about each store:

	location	location_name	address	street_address	city	state_zip	state	zipcode
0	D	Dunkin Donuts	3907 Forbes Ave, Pittsburgh, PA 15123	3907 Forbes Ave	Pittsburgh	PA 15123	PA	15123
1	N	Noodles & Company	3805 Forbes Ave, Pittsburgh, PA 15123	3805 Forbes Ave	Pittsburgh	PA 15123	PA	15123
2	P	Panera Bread	3800 Forbes Ave, Pittsburgh, PA 15123	3800 Forbes Ave	Pittsburgh	PA 15123	PA	15123

We then “broke up” or decomposed the tidy long-format data set into smaller tables…

- We identified the unique shoe colors:

```
shoe_info = lf.groupby(['shoe']).size().reset_index(name='num_rows')
```

```
shoe_info
```

	shoe	num_rows
0	B	10
1	O	10
2	R	10
3	W	10

- And added more descriptive names for the colors:

	shoe	shoe_color
0	B	Black
1	O	Other
2	R	Red
3	W	White

We then added in unique identifiers to each of the “information tables”

```
location_info['location_id'] = location_info.index + 1
```

```
location_info
```

	location	location_name	street_address	city	state	zipcode	location_id
0	D	Dunkin Donuts	3907 Forbes Ave	Pittsburgh	PA	15123	1
1	N	Noodles & Company	3805 Forbes Ave	Pittsburgh	PA	15123	2
2	P	Panera Bread	3800 Forbes Ave	Pittsburgh	PA	15123	3

```
shoe_info['shoe_id'] = shoe_info.index + 1
```

```
shoe_info
```

	shoe	shoe_color	shoe_id
0	B	Black	1
1	O	Other	2
2	R	Red	3
3	W	White	4

Then, we isolated the count for the shoe color in a store on a given day.

- Used JOINS to bring in the unique IDs:

```
lf_copy = pd.merge( lf, shoe_info.loc[:, ['shoe_id', 'shoe']], on='shoe', how='left').\
merge(location_info.loc[:, ['location_id', 'location']], on='location', how='left')
```

- Isolated just a few of the columns:

```
lf_copy = lf_copy[['day', 'location_id', 'shoe_id', 'value']].copy()
```

```
lf_copy
```

	day	location_id	shoe_id	value
0	1	2	4	12
1	2	2	4	5
2	3	2	4	9
3	4	2	4	4
4	1	2	1	5
5	2	2	1	8
6	3	2	1	22
7	4	2	1	2
8	1	2	3	3
9	2	2	3	6
10	3	2	3	7

⋮

Our tidy dataset was now distributed across 3 separate datasets!

- We could recover the tidy dataset (along with the additional more descriptive pieces of information) by JOINING the 3 datasets together!

```
pd.merge( lf_copy, shoe_info, on='shoe_id', how='left').\
merge( location_info, on='location_id', how='left')
```

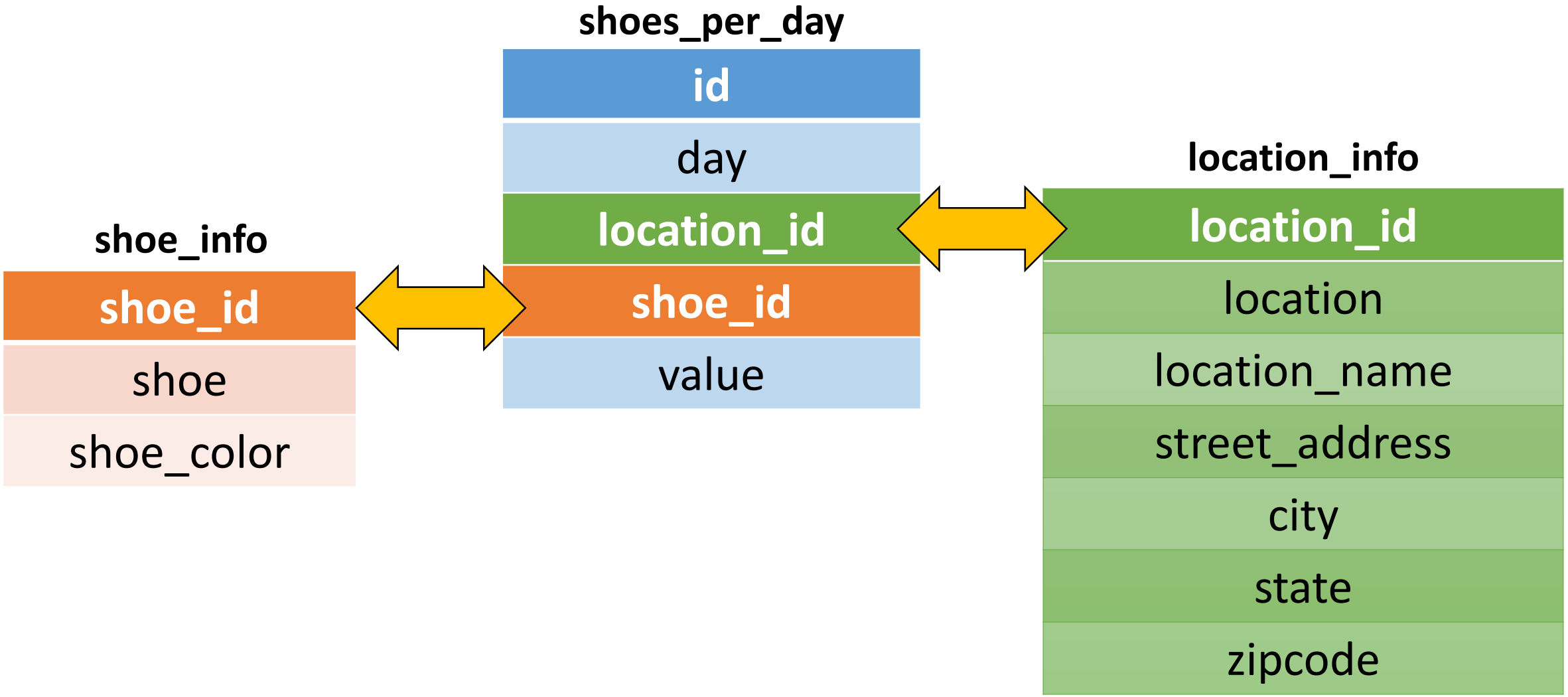
	day	location_id	shoe_id	value	shoe	shoe_color	location	location_name	street_address	city	state	zipcode
0	1	2	4	12	W	White	N	Noodles & Company	3805 Forbes Ave	Pittsburgh	PA	15123
1	2	2	4	5	W	White	N	Noodles & Company	3805 Forbes Ave	Pittsburgh	PA	15123
2	3	2	4	9	W	White	N	Noodles & Company	3805 Forbes Ave	Pittsburgh	PA	15123
3	4	2	4	4	W	White	N	Noodles & Company	3805 Forbes Ave	Pittsburgh	PA	15123
4	1	2	1	5	B	Black	N	Noodles & Company	3805 Forbes Ave	Pittsburgh	PA	15123
5	2	2	1	8	B	Black	N	Noodles & Company	3805 Forbes Ave	Pittsburgh	PA	15123
6	3	2	1	22	B	Black	N	Noodles & Company	3805 Forbes Ave	Pittsburgh	PA	15123
7	4	2	1	2	B	Black	N	Noodles & Company	3805 Forbes Ave	Pittsburgh	PA	15123
8	1	2	3	3	R	Red	N	Noodles & Company	3805 Forbes Ave	Pittsburgh	PA	15123
9	2	2	3	6	R	Red	N	Noodles & Company	3805 Forbes Ave	Pittsburgh	PA	15123
10	3	2	3	7	R	Red	N	Noodles & Company	3805 Forbes Ave	Pittsburgh	PA	15123

⋮

Our tidy dataset represents a DATABASE

- The database contains information about specific ENTITIES.
- Each ENTITY consists of ATTRIBUTES (variables, features, columns in a table).
- Our database understands how the ENTITIES are RELATED to each other.

We created a simple diagram to represent the ENTITIES, their ATTRIBUTES, and their RELATIONSHIPS



Let’s now repeat these steps, but following a more formal process

- We will walk through DATA MODELING where we will create:

- CONCEPTUAL data model.

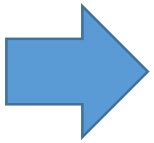
- LOGICAL data model.

- PHYISCAL data model.

Let’s now repeat these steps, but following a more formal process

- We will walk through DATA MODELING where we will create:

- CONCEPTUAL data model.



- Demonstrate an understanding of the high level REQUIREMENTS.
- Provide a GRAPHICAL tool for communicating those requirements with stakeholders.
- Nothing specific to a language or Database Management System (DBMS), programming language, or technology.

- LOGICAL data model.

- PHYISCAL data model.

Let’s now repeat these steps, but following a more formal process

- We will walk through DATA MODELING where we will create:

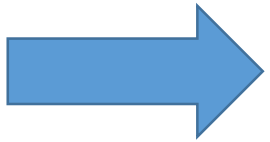
- CONCEPTUAL data model.
- LOGICAL data model.
- PHYISCAL data model.

- More detail than a conceptual model.
- Understand the TABLES and COLUMNS in the tables.
- Understand data types of variables.
- Understand how tables relate to one another.
- Independent of a programming language and technology.

Let’s now repeat these steps, but following a more formal process

- We will walk through DATA MODELING where we will create:

- CONCEPTUAL data model.
- LOGICAL data model.
- PHYISCAL data model.



- Specific to a programming language and technology.
- Creates the tables, columns, and relationships within a specific DBMS, following the logical model.
- We have worked with MySQL as our DBMS for relational data models.

IMPORTANT: iterative process that sometimes requires going back to the beginning!

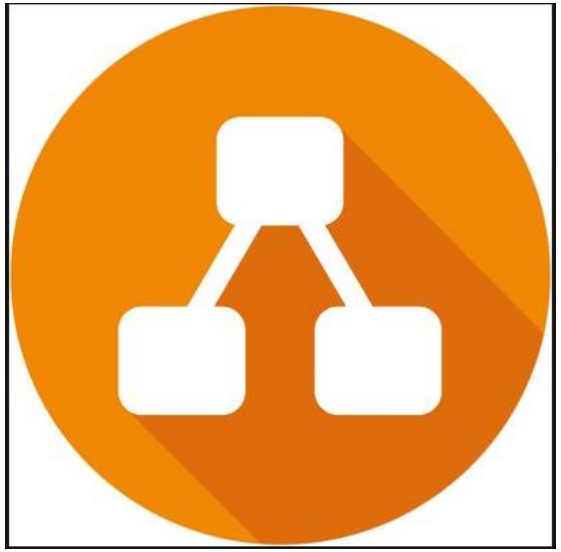
- CONCEPTUAL data model.
- LOGICAL data model.
- PHYISCAL data model.

The second half of the following video provides a nice illustration of the iterative process of designing a database: https://youtu.be/3BZz8R7mqu0?t=1

Data models – making the diagram

- diagrams.net (formerly draw.io) is a free online diagram software.

- Provides nice functionality for creating the ER diagrams.



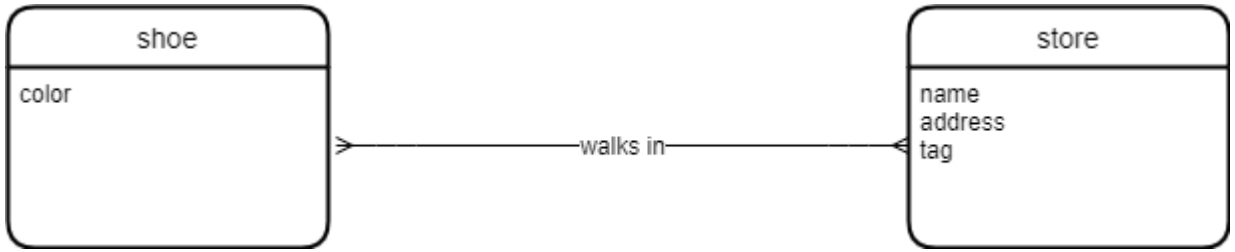
After creating the CONCEPTUAL and LOGICAL data models…

- We will create the PHYSICAL model in MySQL Workbench.

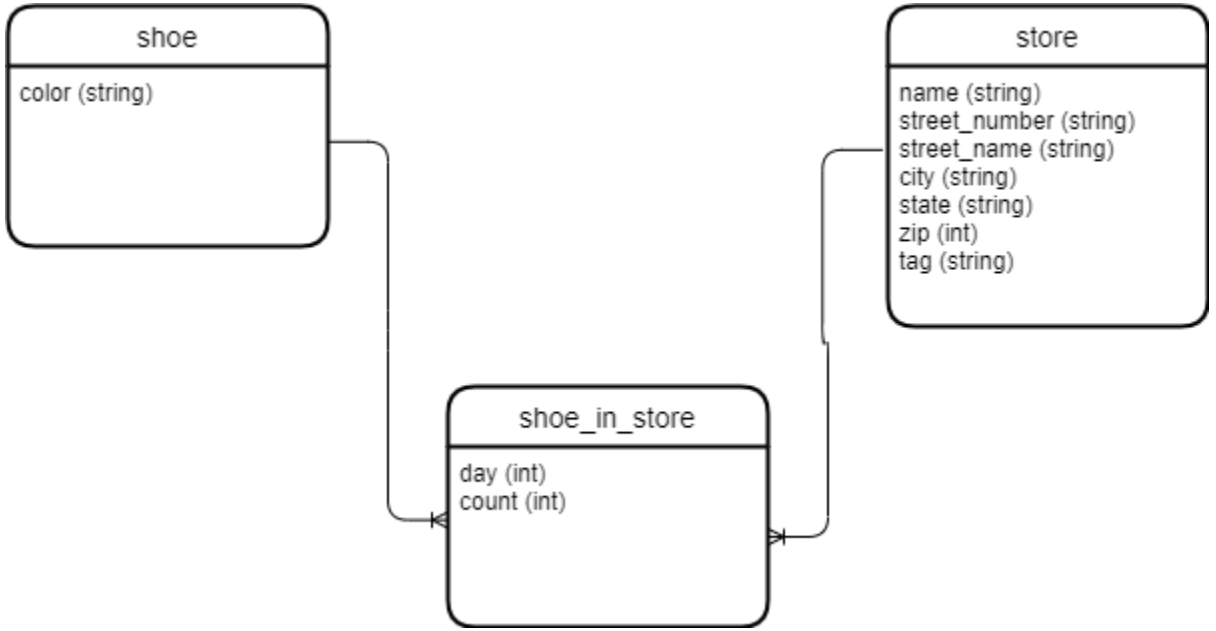
Start by UNDERSTANDING the GOALS

- We want to count the number of specific colors of shoes that enter into different locations or stores.
- We are not interested in the people wearing the shoes in this example, just the shoe color.
- So we do NOT need to store information or ATTRIBUTES about the people!
- However, we are interested in storing information about the stores.

The conceptual data model created in draw.io



The logical data model for the shoes example



Normalization

- We want to prevent anomalies from occurring as we alter or change the data in a database.
- Allow efficient queries to our database.
- RDDI discusses all of the normalization forms in detail.
- We will focus on the first 3 normalization forms.
  - 1NF, 2NF, and 3NF

First normal form

- We want tables as spread sheets.
- We do NOT want multi-valued attributes…each cell must contain a single value.

Second normal form

- Must be in the first normal form.
- All nonkey attributes are functionally dependent on the entire primary key.

Third normal form

- Must be in the second normal form.
- There are no transitive dependencies.