BZAN 6354

Live Lecture 5

February 19, 2024

Dr. Mark Grimes, Ph.D.
gmgrimes@bauer.uh.edu

UNIVERSITY of HOUSTON
C. T. BAUER COLLEGE of BUSINESS
Department of Decision & Information Sciences

Agenda

- Administration
- Module 3.2 – Decomposed Design-Specific ERD models
- Module 3.3 – Data Modeling Errors
- Building on the MGHH Model
- Break
- Module 6.1 – The Relational Data Model
- Module 6.2 – Characteristics of a Relation
- Module 6.3 – Data Integrity Constraints ← We may not get quite this far, we'll see…

Assignment 1

- Grades are posted
  - Average 23 / 25
- Overall good work
- I left brief comments on most submissions - if you want detailed feedback send me an email
  - Attributes like “Phone Number” should be text, not numeric
  - Attributes like “Age” should be derived, not stored (Date of Birth should be stored)
  - Neatness counts!
- Any questions/thoughts on the assignment?

Assignment 2

- Was posted last week, but you need some knowledge from today
  - Weak entities and partial keys
- Creating an ERD for “Shampooch Doggy Spa”
  - Different than “Dave’s Dog Wash”
- Just like with assignment 1, there is a walkthrough video to help you along the way / let you check your work
- Due one week from today – February 26

Exam 1

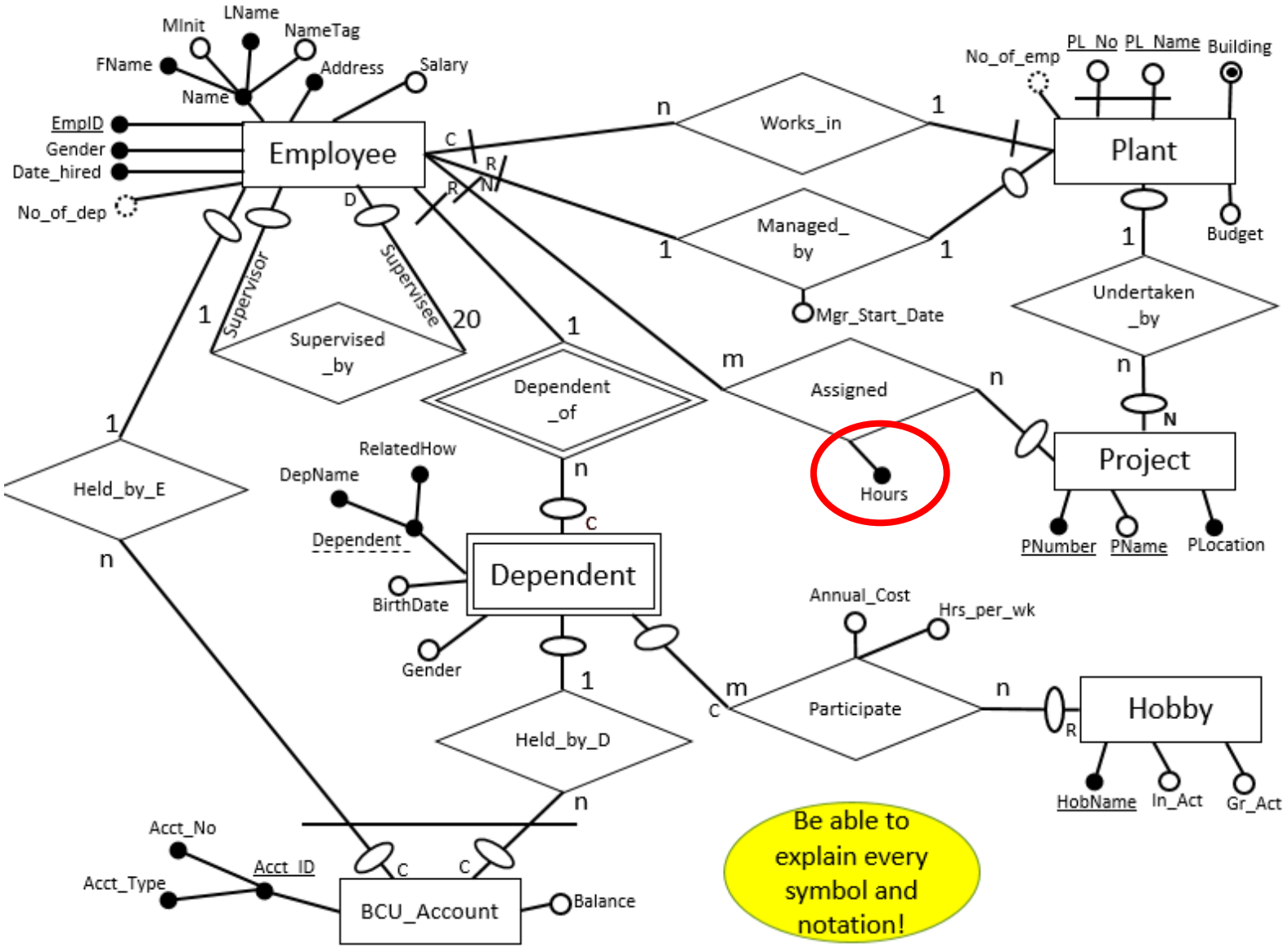
- Coming up in two weeks!
  - Learning objectives from each module are useful as a “study guide”
  - Weekly Progress Quizzes are good practice

- Will be about
  - 1/3 Multiple Choice
  - 1/3 Fill in the blank/matching/etc.
  - 1/3 Drawing an ERD

Module 3.2.4
Decomposed Design-Specific ERD

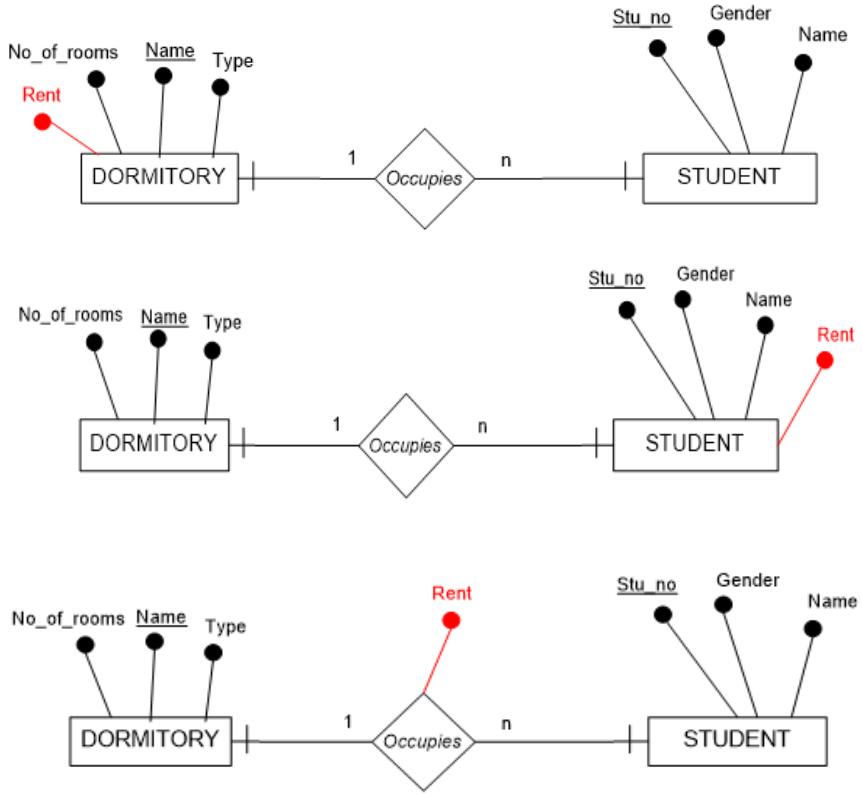
- Attribute Placement
- Weak entities
- Capturing Multi-value attributes
- Modeling m:n relationships

Attribute Placement



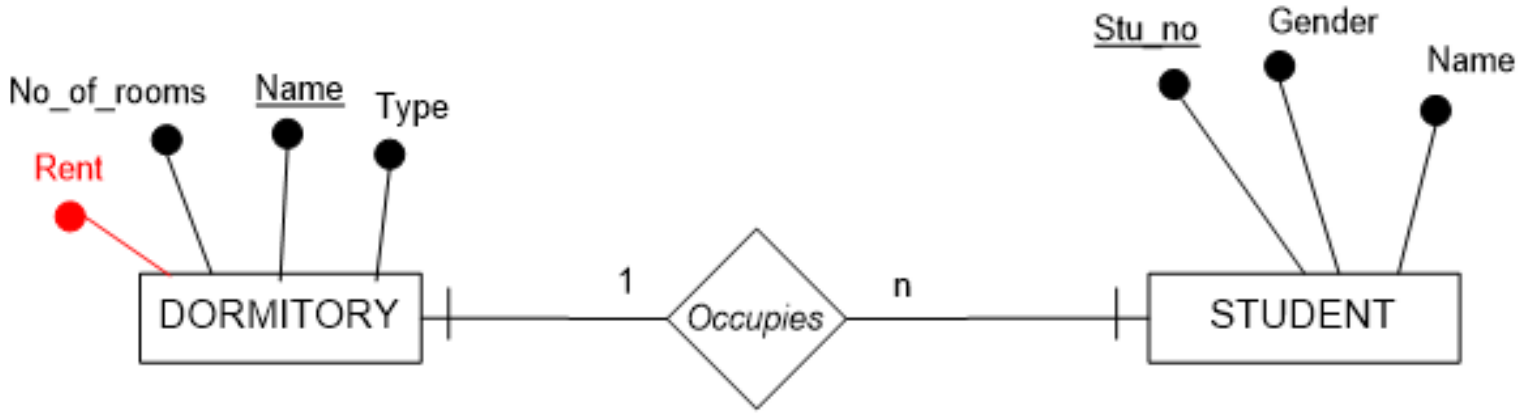
Attribute Placement

- Attributes may not only be part of an entity, but may also be part of the relationship
- Rent on a dorm room is the same no matter who is in it
- A student pays the same rent no matter what dorm room they have
- Rent varies based on student and room type



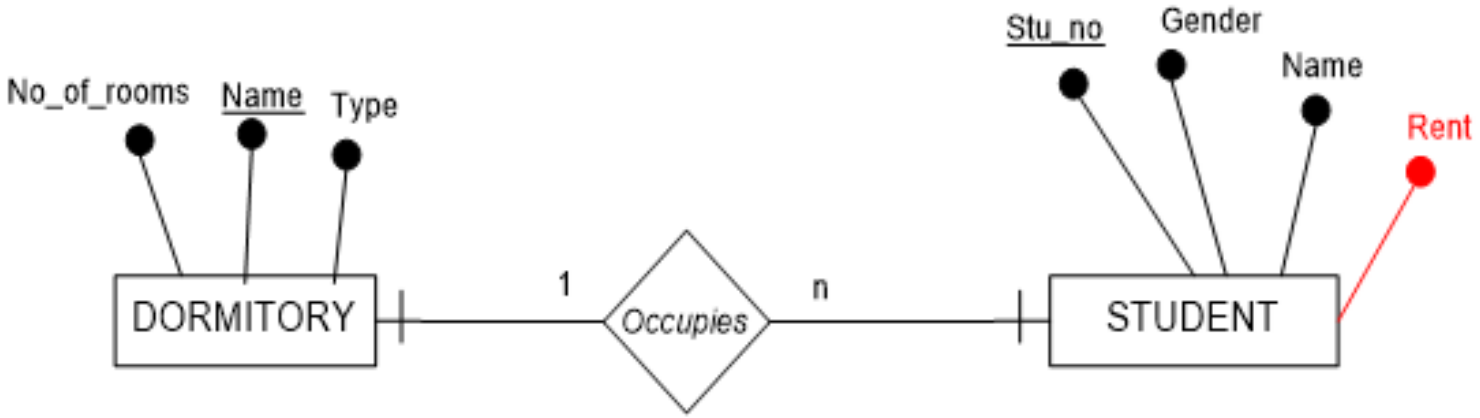
Attribute Placement

- Room A costs $1,200 per semester
  - Room A is really nice
- Room B costs $1,000 per semester
  - Room B is a normal dorm room
- Room C costs $800 per semester
  - Room C is a smaller room in an older dormitory



Attributes in a relationship

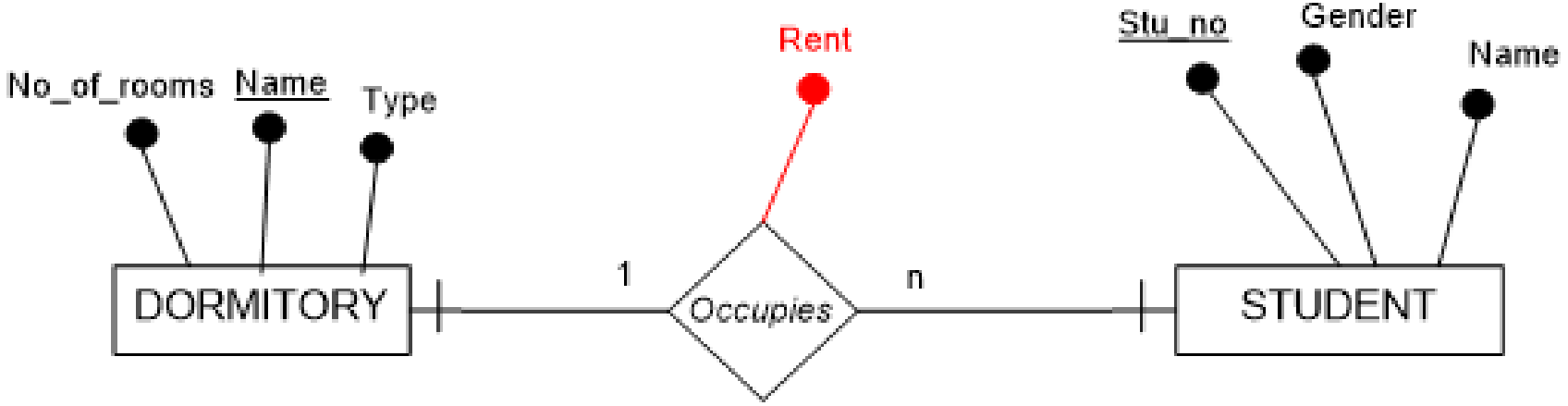
- Doug pays $600 per semester, no matter what room
  - Maybe scholarship? Financial need?
- Eugene pays $1,000 per semester, no matter what room
  - Perhaps a discount for having good grades
- Fiona pays $1,400 per semester, no matter what room
  - Normal rate



Attributes in a relationship

- Rate depends both on how nice the room is and the student's details
- Which you pick depend on your.... Business Rules

	Room A	Room B	Room C
Doug	$800	$700	$500
Eugene	$1,400	$1,200	$800
Fiona	$1,600	$1,400	$1,000

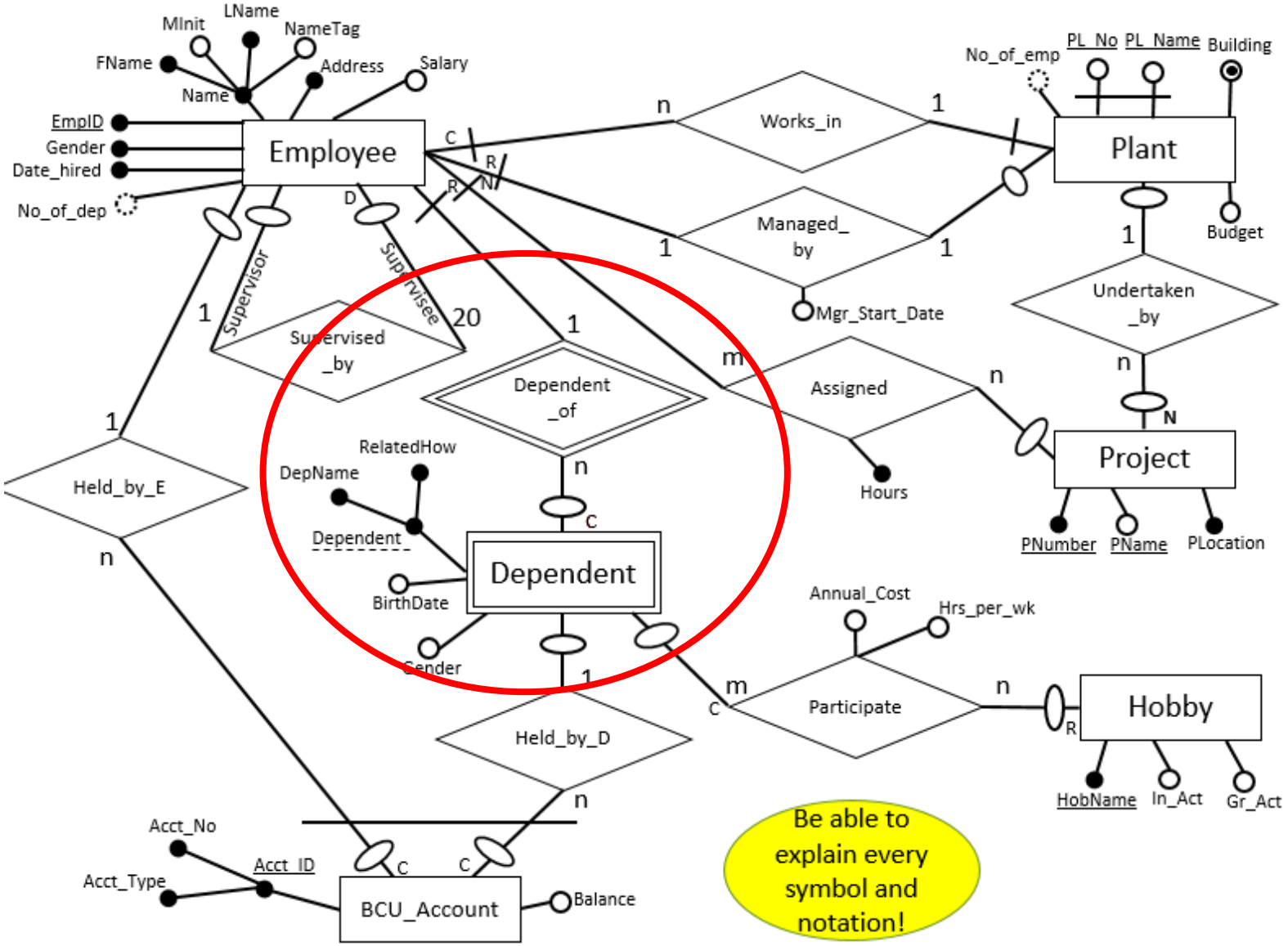


Base/Strong vs. Weak Entity Types

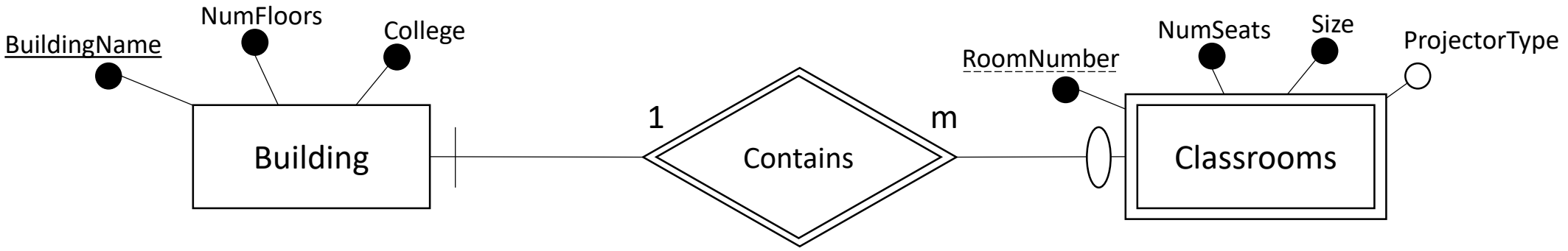
- Base (or strong) entity types are those where the entities have independent existence (i.e., each entity of this entity type can be uniquely identified)
  - A base entity type has a unique identifier.

- Weak entity types are those where entities do not have an independent existence
  - Duplicate entity instances may be present
  - A weak entity type does not have a unique identifier
  - Must be related to a base (strong) entity to be identified – known as an "identifying relationship"
  - A weak entity type has a "**partial key**" – also known as a "discriminator"

Base/Strong vs. Weak Entity Types



Weak entity type: Classroom as an example

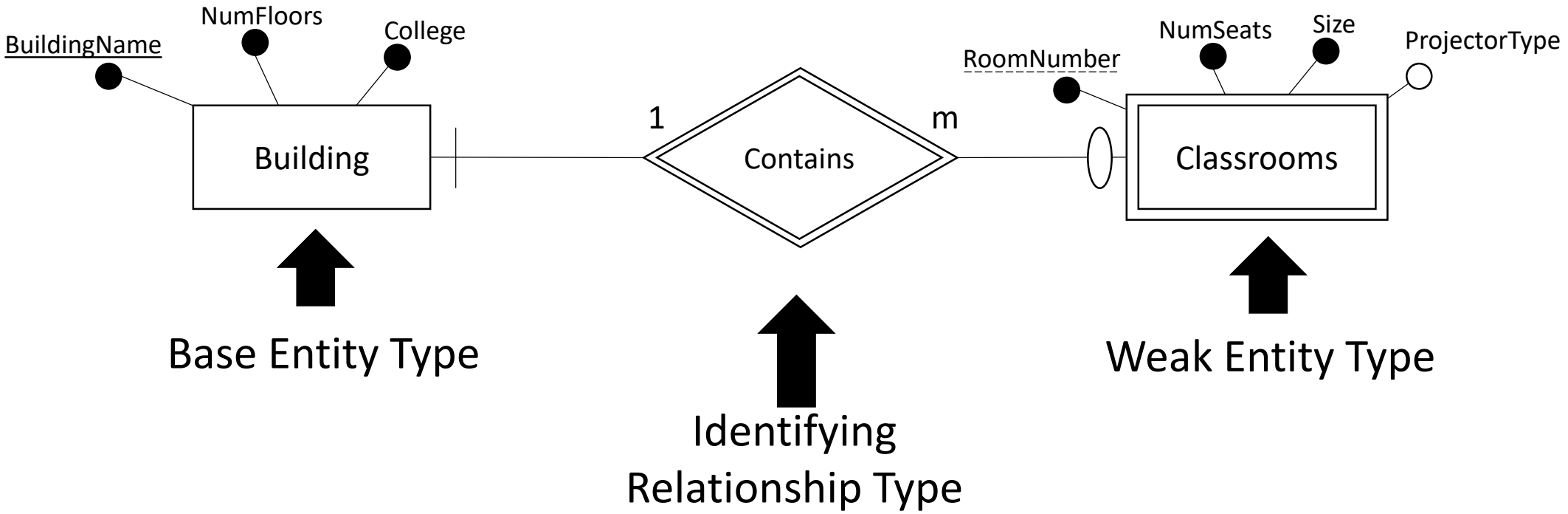


Building		
BuildingName	NumFloors	College
Melcher	3	Business
CBB	5	Business
Cemo	2	Business

Classrooms			
RoomNumber	NumSeats	Size	ProjectorType
110	66	900	Epson MX324
111	66	900	Epson MX324
112	66	900	ViewSonix P32
108	25	400	
110	20	450	Epson Slidemaster
214	120	2200	
226	80	1100	ViewSonix P32
109	85	1100	
111	120	1350	Epson MX324
112	80	900	

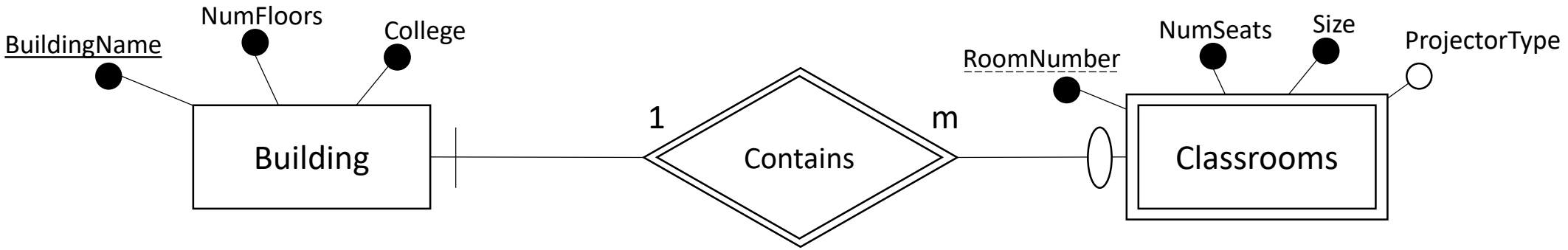
Note: A similar example with “apartments” is on page 53 of the book

Base/Strong vs. Weak Entity Types



- Double square = weak entity
- Double diamond = identifying relationship

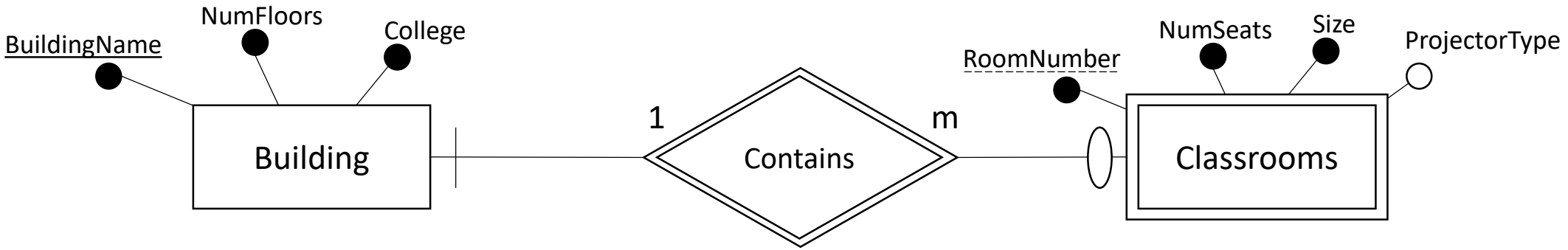
Partial key (Discriminator) Defined



- An attribute, atomic or composite, in a weak entity type, which in conjunction with a unique identifier(s) of the parent entity type(s) in the identifying relationship type(s), uniquely identifies weak entities is called the partial key (discriminator) of a weak entity type.

- Primary key (solid underline) for Building table: BuildingName
- Partial key (dotted underline) for Classrooms table: RoomNumber
- To uniquely identify a classroom , you must specify **BOTH** the BuildingName AND RoomNumber
- This is the Identifying relationship

A reminder

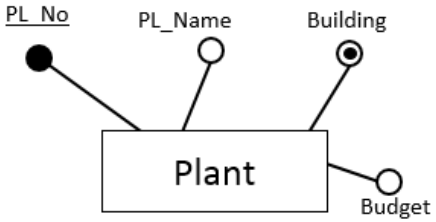


- Database design is both a SCIENCE and an ART

- There are other ways to model this depending on:
  - Business rules
  - Performance considerations
  - Personal preference

Two issues that change the way you look at ERDs

- Multi-value attributes don't really exist*
  - Exist in conceptual models, but most DBMS will only allow for a single value
  - Can be solved by creating a 1:m or m:n relationship



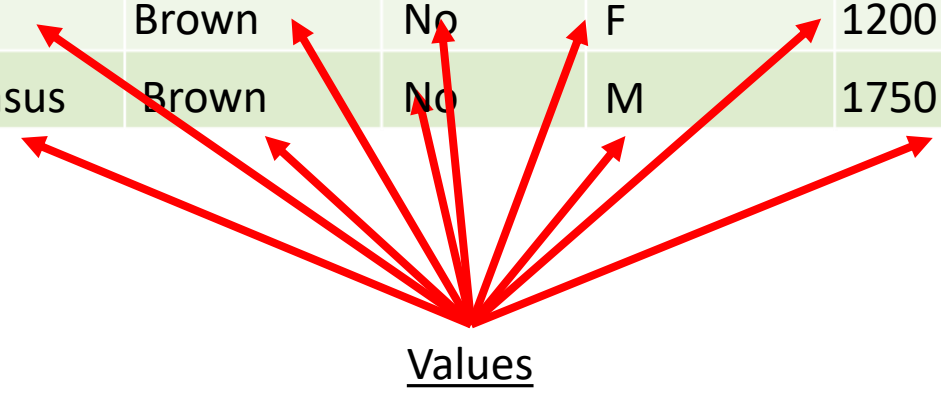
- M:N relationships don't really exist*
  - Remember, they are really two 1:m and m:1 relationships

* They exist CONCEPTUALLY, but cannot be technically implemented in a database

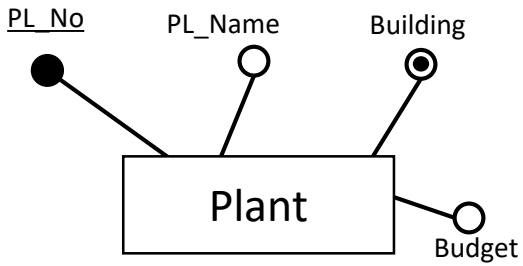
Modeling a multi-value attribute

- We understand values as being at the intersection of a column and a row
- …but what about when we have “multi-value” attributes?

Horses				
Name	Color	Spots	Sex	Weight
Sam	Brown	No	F	1500
Erica	Yellow	Yes	F	920
John	Grey	No	M	1800
Trotty	Brown	Yes	M	1300
Rio	Grey	No	F	1700
Robin	Yellow	No	M	1100
Katy	Brown	No	F	1200
Pegasus	Brown	No	M	1750



Modeling a multi-value attribute

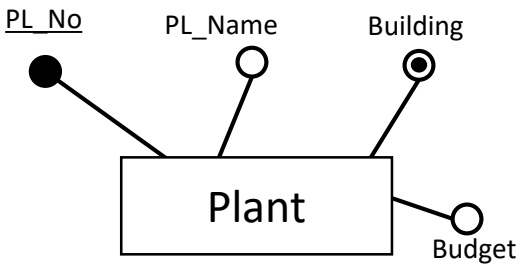


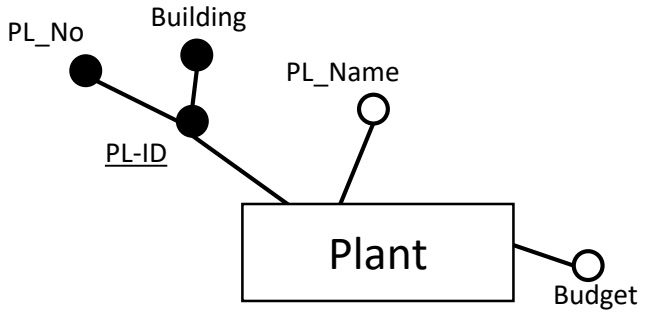
Plant			
PL_No	PL_Name	Budget	Building
10	Underwood	3000000	1
			2
			3
11	Garnett	3000000	1
			2
12	Belmont	3500000	1
13	Vanderbilt	3500000	1
			2

- This is basically what we have been saying, but this functionally does not work
- We have rows of data with required attributes that are not populated

- Note: We can ignore the derived attributes for now

Modeling a multi-value attribute – Option #1





Plant

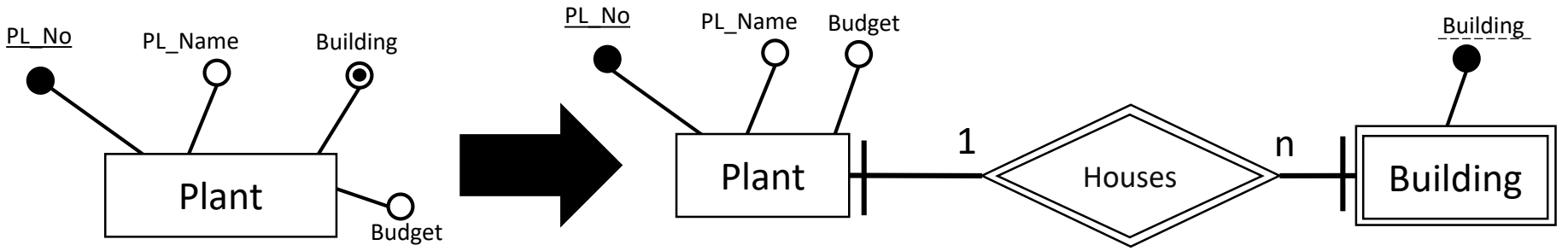
PL_No	PL_Name	Budget	Building
10	Underwood	3000000	1
			2
			3
11	Garnett	3000000	1
			2
12	Belmont	3500000	1
13	Vanderbilt	3500000	1
			2

Plant

PL_No	Building	PL_Name	Budget
10	1	Underwood	3000000
10	2	Underwood	3000000
10	3	Underwood	3000000
11	1	Garnett	3000000
11	2	Garnett	3000000
12	1	Belmont	3500000
13	1	Vanderbilt	3500000
13	2	Vanderbilt	3500000

Technically works, but not a good solution - Data Redundancy!

Modeling a multi-value attribute – Option #2 (correct!)



Plant

PL_No	PL_Name	Budget	Building
10	Underwood	3000000	1
			2
			3
11	Garnett	3000000	1
			2
12	Belmont	3500000	1
13	Vanderbilt	3500000	1
			2

Plant

PL_No	PL_Name	Budget
10	Underwood	3000000
11	Garnett	3000000
12	Belmont	3500000
13	Vanderbilt	3500000

Building

PL_No	Building
10	1
10	2
10	3
11	1
11	2
12	1
13	1
13	2

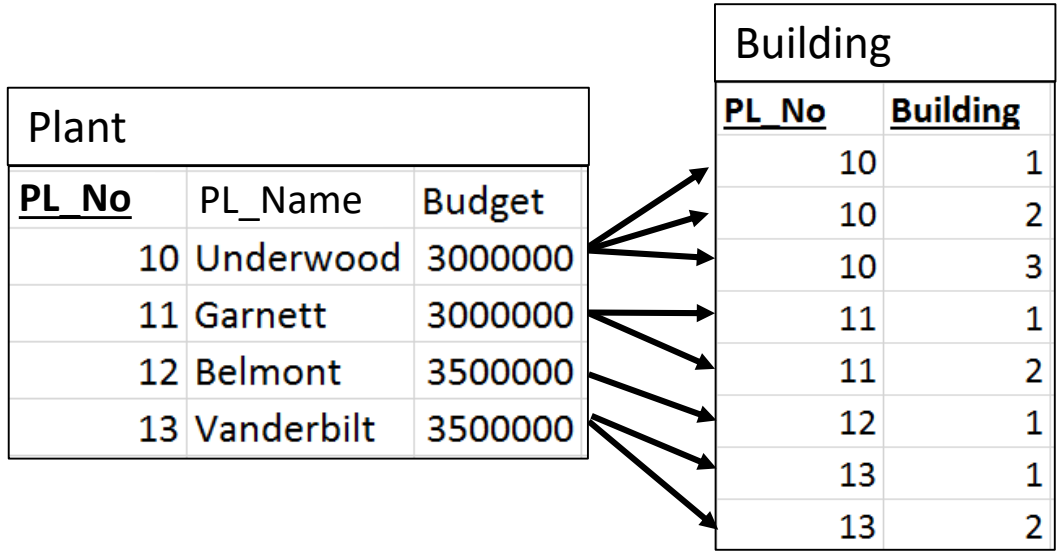
A bit more complex, but solves our data redundancy problems!

?

Together, PL_No and Building are a composite Primary Key → Foreign Key Partial Key

Important note

- While this is "correct" for most business → purposes as it reduces redundancy…
  - Data is "normalized"
- …there is more "computational complexity" in the "correct" approach
- For data mining/data warehouse applications, this may be desirable →
  - Pro: Faster/Less complex
  - Con: Less efficient use of storage
  - Con: More difficult to update…
    - …but we infrequently "update" in a data warehouse, so perhaps not an issue!
- What you choose depends on the business's needs!



Plant

PL_No	PL_Name	Budget
10	Underwood	3000000
11	Garnett	3000000
12	Belmont	3500000
13	Vanderbilt	3500000

Building

PL_No	Building
10	1
10	2
10	3
11	1
11	2
12	1
13	1
13	2

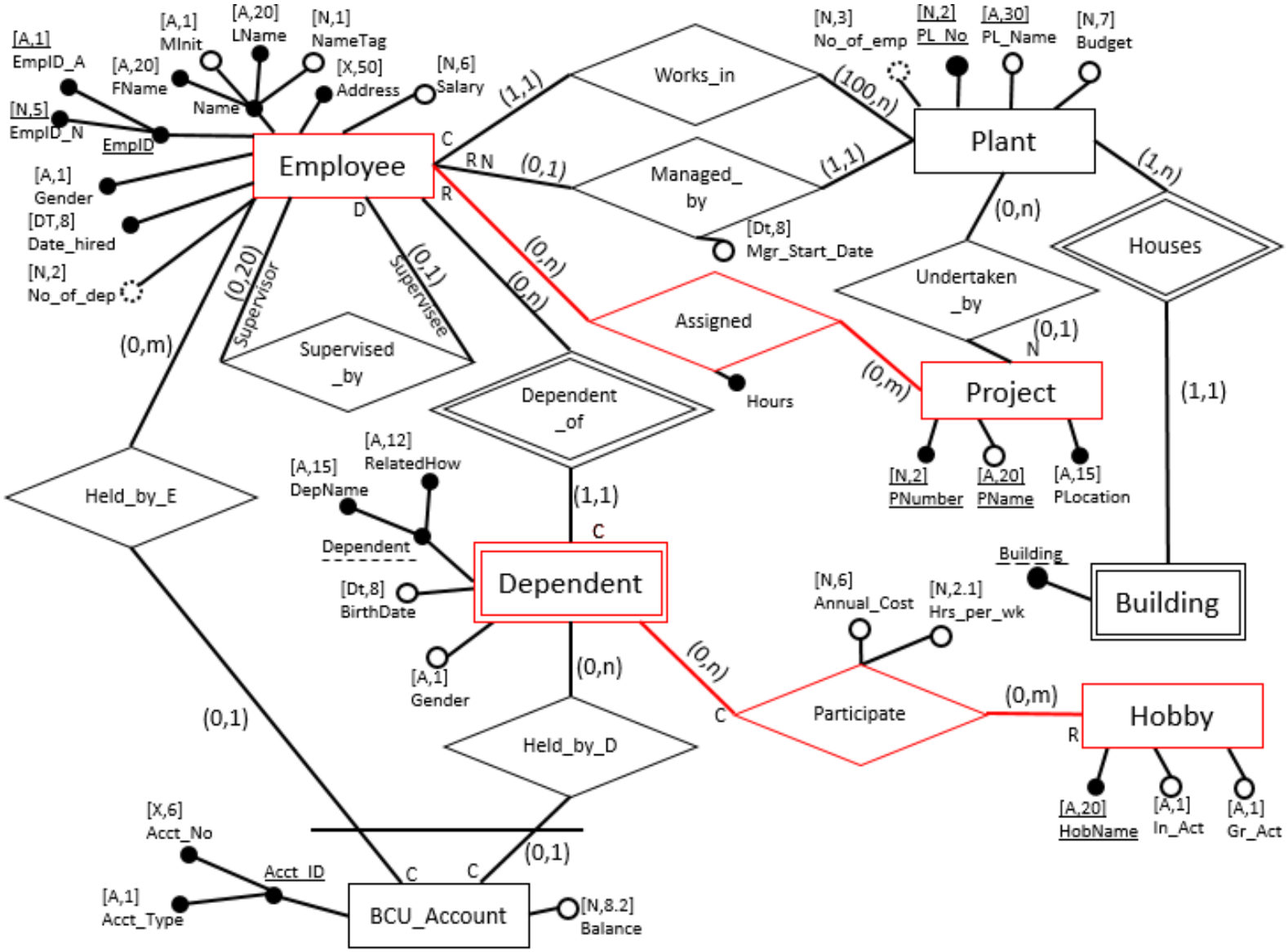
Plant

PL_No	Building	PL_Name	Budget
10	1	Underwood	3000000
10	2	Underwood	3000000
10	3	Underwood	3000000
11	1	Garnett	3000000
11	2	Garnett	3000000
12	1	Belmont	3500000
13	1	Vanderbilt	3500000
13	2	Vanderbilt	3500000

Modeling a many to many relationship

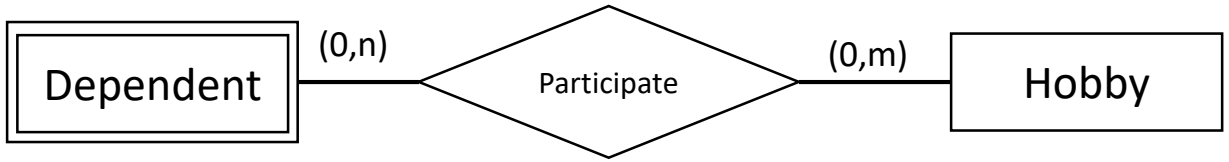
We have two

- Employee-Project
  - Many employees are assigned to a project
  - A project has many employees assigned to it
- Dependent-Hobby
  - A dependent may have many hobbies
  - A hobby may have many dependents that participate



Let’s do dependent-hobby

- Ignore for the moment that dependent is a weak entity
  - Note: I am leaving out many attributes for simplicity of the example – they have not really disappeared



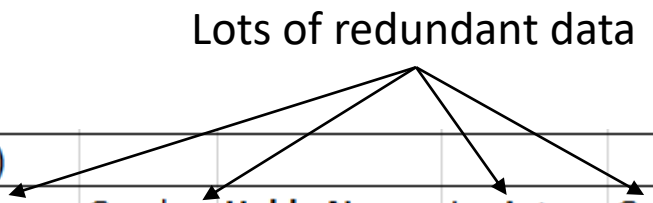
Dependents		
Name	Birthdate	Gender
Mark	5/11/1945	M
Jill	5/4/1976	F
Norm	3/1/1984	M
Mike	1/31/1992	M
Sue	9/2/1990	F

Hobby		
HobbyName	In_Act	Gr_Act
Soft Ball	O	G
Flag Football	O	G
Knitting	I	I
Cycling	O	I
Movies	I	G

- Do employee-project on your own for practice
  - It’s in the book on page 116

Let’s do dependent-hobby

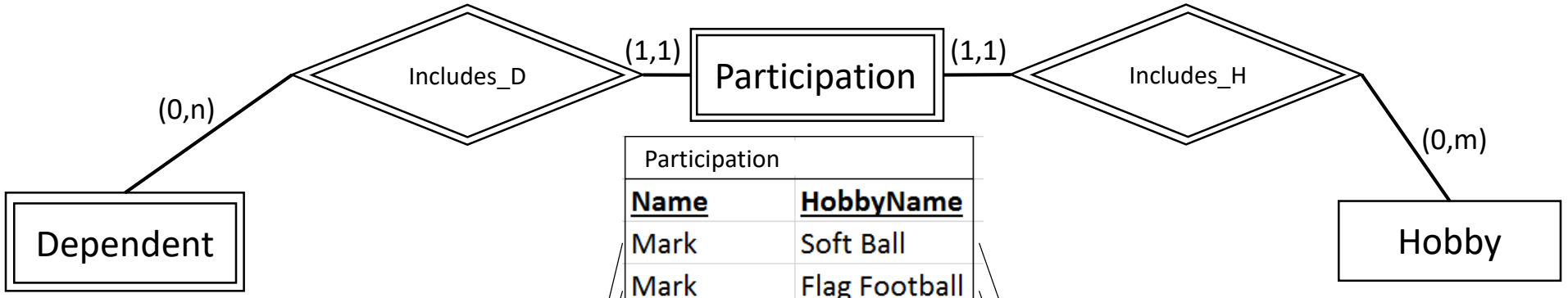
- Name-HobbyName makes for a fine key value
- What’s wrong here?



Participates (relationship)					
Name	Birthdate	Gender	**HobbyName**	In_Act	Gr_Act
Mark	5/11/1945	M	Soft Ball	O	G
Mark	5/11/1945	M	Flag Football	O	G
Mark	5/11/1945	M	Cycling	O	I
Jill	5/4/1976	F	Knitting	I	I
Jill	5/4/1976	F	Movies	I	G
Norm	3/1/1984	M	Soft Ball	O	G
Mike	1/31/1992	M	Soft Ball	O	G
Mike	1/31/1992	M	Flag Football	O	G
Mike	1/31/1992	M	Knitting	I	I
Mike	1/31/1992	M	Cycling	O	I
Mike	1/31/1992	M	Movies	I	G
Sue	9/2/1990	F	Soft Ball	O	G
Sue	9/2/1990	F	Cycling	O	I
Sue	9/2/1990	F	Movies	I	G

Let’s do dependent-hobby

- Decomposed the M:N relationship into two 1:M relationships
- The Participation entity is a **gerund**



Participation

Name	**HobbyName**
Mark	Soft Ball
Mark	Flag Football
Mark	Cycling
Jill	Knitting
Jill	Movies
Norm	Soft Ball
Mike	Soft Ball
Mike	Flag Football
Mike	Knitting
Mike	Cycling
Mike	Movies
Sue	Soft Ball
Sue	Cycling
Sue	Movies

Dependents

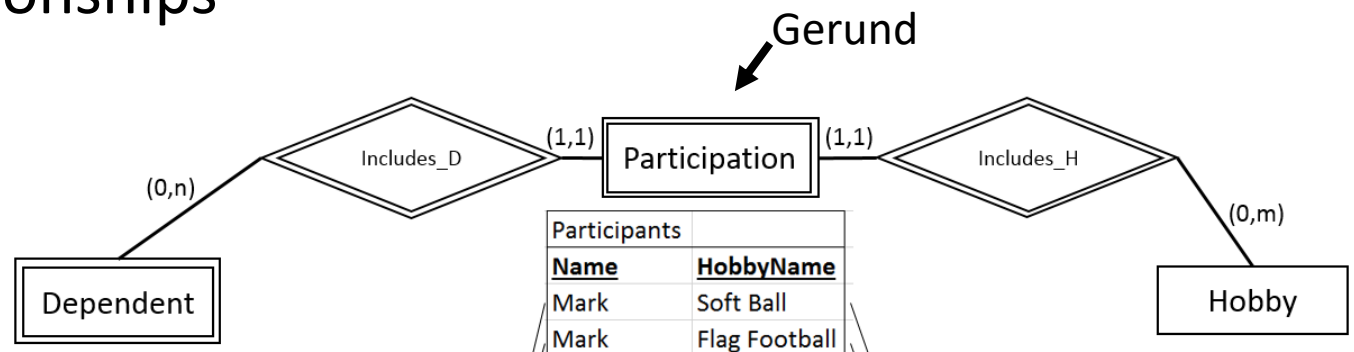
Name	Birthdate	Gender
Mark	5/11/1945	M
Jill	5/4/1976	F
Norm	3/1/1984	M
Mike	1/31/1992	M
Sue	9/2/1990	F

Hobby

HobbyName	In_Act	Gr_Act
Soft Ball	O	G
Flag Football	O	G
Knitting	I	I
Cycling	O	I
Movies	I	G

The Gerund

- Gerunds are a product of decomposing m:n relationships
  - Also called a “composite entity” or “bridge entity”
  - Takes the primary key from each participating entity to create a set of 1:n and m:1 relationships



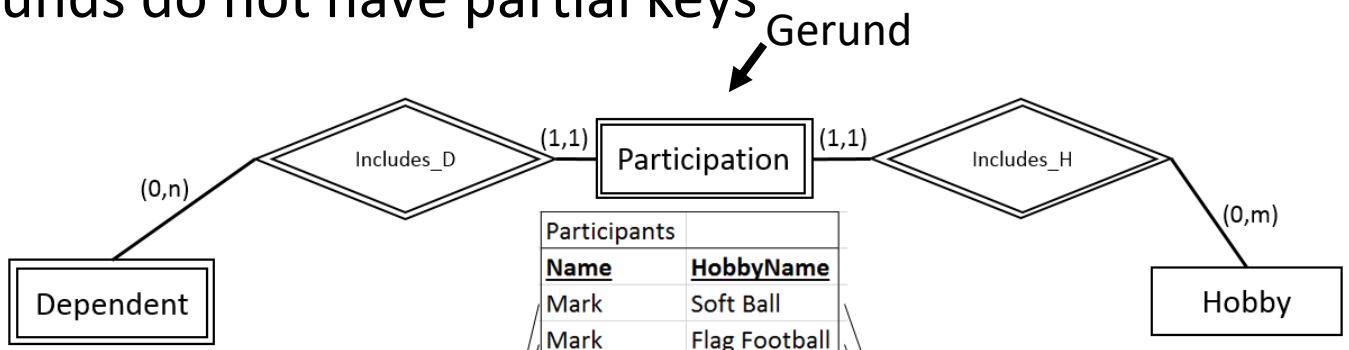
Participants	
Name	**HobbyName**
Mark	Soft Ball
Mark	Flag Football
Mark	Cycling
Jill	Knitting
Jill	Movies
Norm	Soft Ball
Mike	Soft Ball
Mike	Flag Football
Mike	Knitting
Mike	Cycling
Mike	Movies
Sue	Soft Ball
Sue	Cycling
Sue	Movies

Dependents		
Name	Birthdate	Gender
Mark	5/11/1945	M
Jill	5/4/1976	F
Norm	3/1/1984	M
Mike	1/31/1992	M
Sue	9/2/1990	F

Hobby		
HobbyName	In_Act	Gr_Act
Soft Ball	O	G
Flag Football	O	G
Knitting	I	I
Cycling	O	I
Movies	I	G

The Gerund

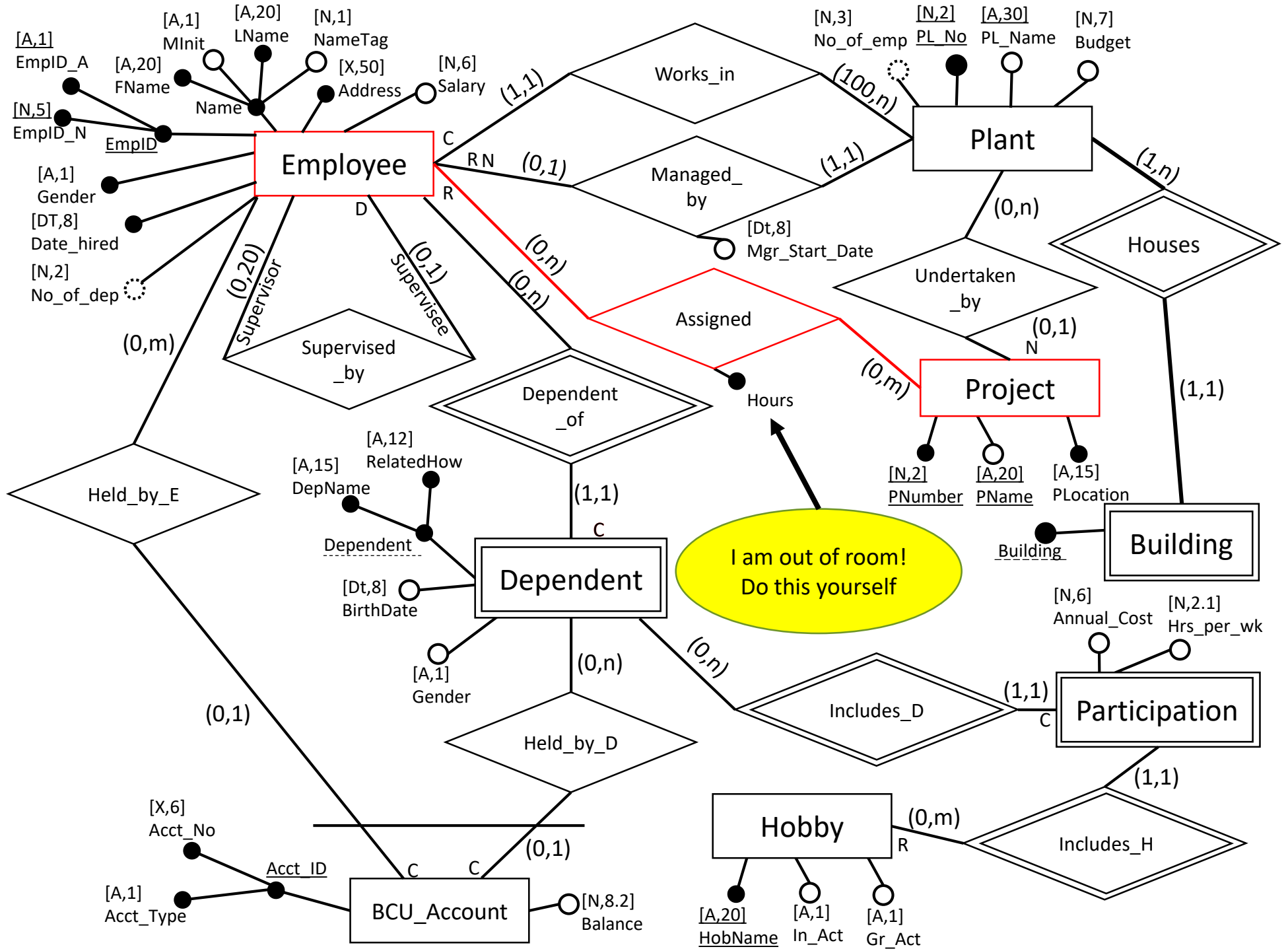
- Looks and acts like a weak entity, but not the same
  - **Weak entities** are a product of the **business rules**
  - **Gerunds** are a product **of decomposing m:n relationships**
    - Gerunds do not have partial keys



Participants	
Name	**HobbyName**
Mark	Soft Ball
Mark	Flag Football
Mark	Cycling
Jill	Knitting
Jill	Movies
Norm	Soft Ball
Mike	Soft Ball
Mike	Flag Football
Mike	Knitting
Mike	Cycling
Mike	Movies
Sue	Soft Ball
Sue	Cycling
Sue	Movies

Dependents		
Name	Birthdate	Gender
Mark	5/11/1945	M
Jill	5/4/1976	F
Norm	3/1/1984	M
Mike	1/31/1992	M
Sue	9/2/1990	F

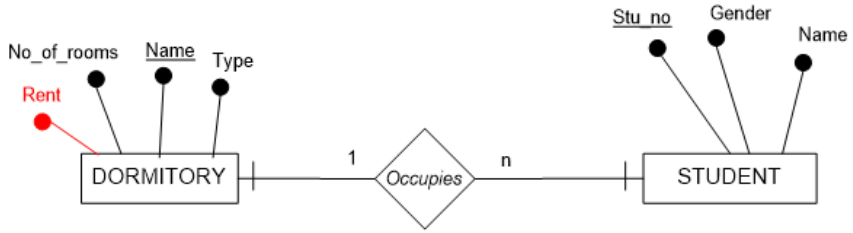
Hobby		
HobbyName	In_Act	Gr_Act
Soft Ball	O	G
Flag Football	O	G
Knitting	I	I
Cycling	O	I
Movies	I	G


[A,1]
EmpID_A
[A,1]
MInit
[A,20]
LName
[N,1]
NameTag
[A,20]
FName
[X,50]
Address
[N,6]
Salary
Name
[N,5]
EmpID_N
EmpID
Employee
[A,1]
Gender
[DT,8]
Date_hired
[N,2]
No_of_dep
Works_in
(1,1)
(100,n)
[N,3]
No_of_emp
[N,2]
PL_No
[A,30]
PL_Name
[N,7]
Budget
Plant
C
R N
(0,1)
Managed_
by
(1,1)
[Dt,8]
Mgr_Start_Date
R
D
(0,n)
(1,n)
Houses
(0,n)
Undertaken
_by
(0,1)
N
(0,20)
Supervisor
(0,1)
Supervisee
(0,n)
Assigned
Supervised
_by
(0,m)
Hours
(0,m)
Project
Dependent
_of
[N,2]
PNumber
[A,20]
PName
[A,15]
PLocation
(1,1)
[A,12]
RelatedHow
[A,15]
DepName
Held_by_E
(1,1)
C
Building
Building
Dependent
[Dt,8]
BirthDate
I am out of room!
Do this yourself
[N,6]
Annual_Cost
[N,2.1]
Hrs_per_wk
[A,1]
Gender
(0,n)
(0,n)
Includes_D
(1,1)
Participation
C
(0,1)
Held_by_D
(1,1)
[X,6]
Acct_No
Hobby
(0,m)
R
Includes_H
Acct_ID
C
C
(0,1)
[A,1]
Acct_Type
BCU_Account
[N,8.2]
Balance
[A,20]
HobName
[A,1]
In_Act
[A,1]
Gr_Act

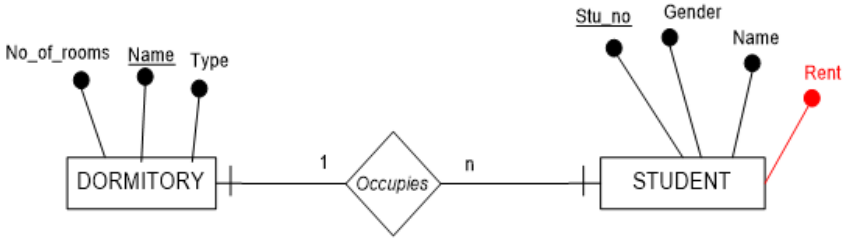
Recall when we talked about this:

- Attributes may not only be part of an entity, but may also be part of the relationship

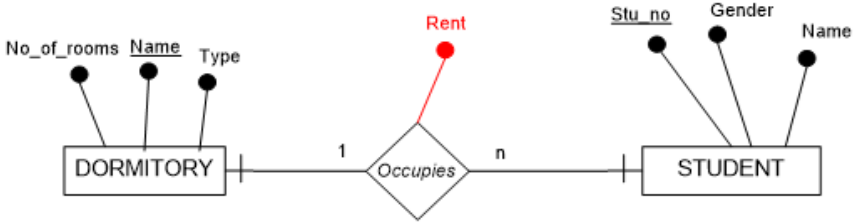
- Rent on a dorm room is the same no matter who is in it



- A student pays the same rent no matter what dorm room they have

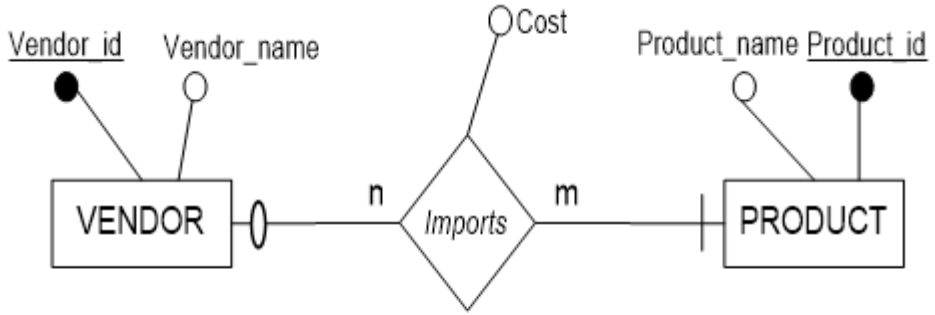


- Rent varies based on student and room type



The gerund is also how we capture attributes of a relationship

Another case of attributes in a relationship



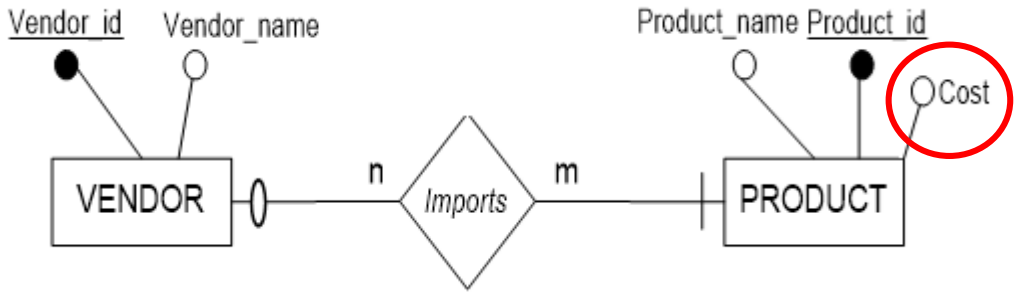
Note: Since the cardinality constraint of *Imports* is m:n, Cost cannot be an attribute of either VENDOR or PRODUCT – **Cost must remain as an attribute of *Imports***

VENDOR	
Vendor_id	Vendor_name
V3	Buffet Inc.
V5	Gates Inc.
V7	Jobs Inc.

IMPORTS		
Vendor_id	Product_id	Cost
V3	P11	7
V5	P11	10
V5	P17	17
V7	P17	19
V7	P23	13

PRODUCT	
Product_name	Product_id
Soup	P11
Noodles	P13
Chocolate	P17
Nuts	P19
Coffee	P23

What if we moved cost to product?

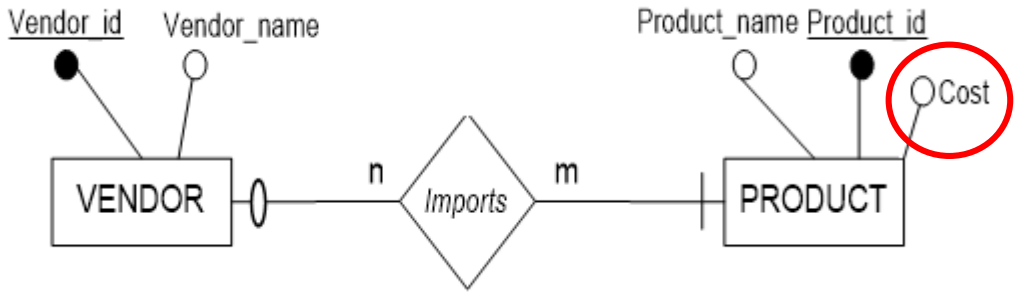


VENDOR	
Vendor_id	Vendor_name
V3	Buffet Inc.
V5	Gates Inc.
V7	Jobs Inc.

IMPORTS	
Vendor_id	Product_id
V3	P11
V5	P11
V5	P17
V7	P17
V7	P23

PRODUCT		
Product_name	Product_id	Cost
Soup	P11	7
Noodles	P13	7
Chocolate	P17	17
Nuts	P19	19
Coffee	P23	13

What if we moved cost to product?



- Now products must cost the same regardless of the vendor – does this reflect reality?

VENDOR	
Vendor_id	Vendor_name
V3	Buffet Inc.
V5	Gates Inc.
V7	Jobs Inc.

IMPORTS	
Vendor_id	Product_id
V3	P11
V5	P11
V5	P17
V7	P17
V7	P23

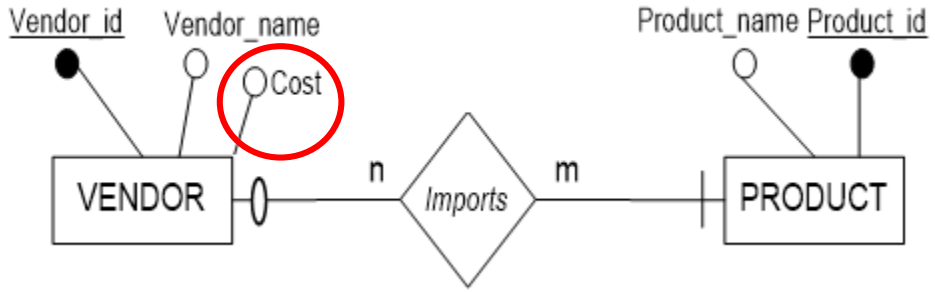
PRODUCT		
Product_name	Product_id	Cost
Soup	P11	7
Noodles	P13	7
Chocolate	P17	17
Nuts	P19	19
Coffee	P23	13

Cost as part of product

- A box of girl scout cookies costs the same $6 regardless of the vender (troop) you buy from

- In commodities markets, a barrel of oil or bushel of grain costs the same regardless of who you buy from

What if we moved cost to vendor?



VENDOR

Vendor_id	Vendor_name	Cost
V3	Buffet Inc.	7
V5	Gates Inc.	10
V7	Jobs Inc.	19

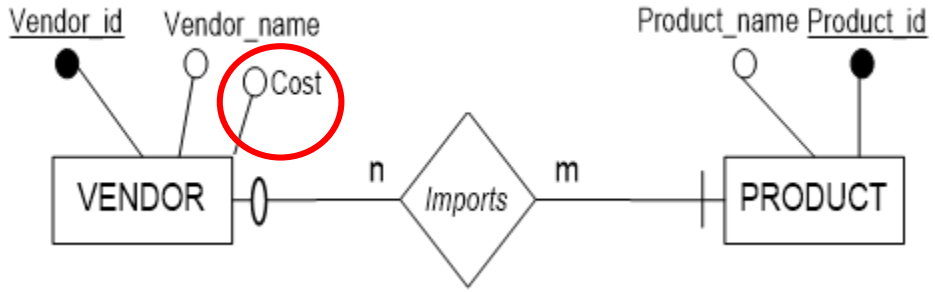
IMPORTS

Vendor_id	Product_id
V3	P11
V5	P11
V5	P17
V7	P17
V7	P23

PRODUCT

Product_name	Product_id
Soup	P11
Noodles	P13
Chocolate	P17
Nuts	P19
Coffee	P23

What if we moved cost to vendor?



- Now all products from a vendor cost the same regardless of the product – does this reflect reality?

VENDOR		
Vendor_id	Vendor_name	Cost
V3	Buffet Inc.	7
V5	Gates Inc.	10
V7	Jobs Inc.	19

IMPORTS	
Vendor_id	Product_id
V3	P11
V5	P11
V5	P17
V7	P17
V7	P23

PRODUCT	
Product_name	Product_id
Soup	P11
Noodles	P13
Chocolate	P17
Nuts	P19
Coffee	P23

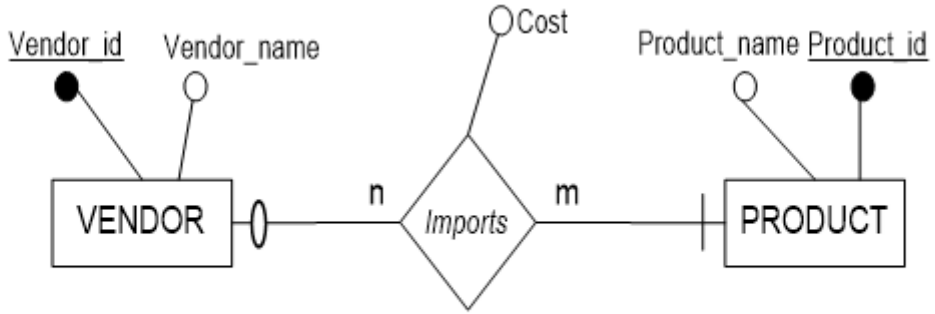
Cost as part of vendor

- At a farmers market, it might be the case that all products from a vendor cost the same, regardless of what the product is

- From a vending machine, all items might cost the same, regardless of what the product is

The most normal case though…

- Cost depends on both the vendor and the product, therefore



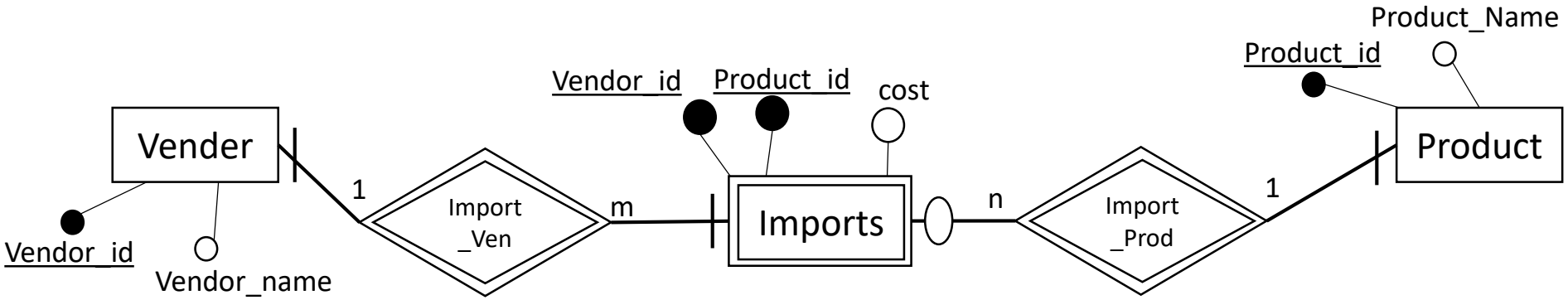
Note: Since the cardinality constraint of *Imports* is m:n, Cost cannot be an attribute of either VENDOR or PRODUCT – **Cost must remain as an attribute of *Imports***

VENDOR	
Vendor_id	Vendor_name
V3	Buffet Inc.
V5	Gates Inc.
V7	Jobs Inc.

IMPORTS		
Vendor_id	Product_id	Cost
V3	P11	7
V5	P11	10
V5	P17	17
V7	P17	19
V7	P23	13

PRODUCT	
Product_name	Product_id
Soup	P11
Noodles	P13
Chocolate	P17
Nuts	P19
Coffee	P23

The decomposed model of this



Note: Since the cardinality constraint of *Imports* is m:n, Cost cannot be an attribute of either VENDOR or PRODUCT – **Cost must remain as an attribute of *Imports***

VENDOR	
Vendor_id	Vendor_name
V3	Buffet Inc.
V5	Gates Inc.
V7	Jobs Inc.

IMPORTS		
Vendor_id	Product_id	Cost
V3	P11	7
V5	P11	10
V5	P17	17
V7	P17	19
V7	P23	13

PRODUCT	
Product_name	Product_id
Soup	P11
Noodles	P13
Chocolate	P17
Nuts	P19
Coffee	P23

Module 3.2.4
Decomposed Design-Specific ERD

- Attribute Placement
- Weak entities
- How do we handle multi-value attributes?
- How do we handle m:n relationships?

Module 3.3
Modeling Errors

- What are semantic errors?
- What are syntactical errors?

Modeling errors

- Errors may be in:
  - Syntax: using the symbols incorrectly
    - Relatively easy to spot
  - Semantics: not accurately reflecting the business rules
    - More difficult to spot, because these are often judgment calls

- How do we know what is an entity vs. an attribute vs. a value?
  - Often trial and error + experience

- It can be difficulty to know exactly how something should be modeled until you start modeling it and fail
  - Was this anyone's assignment 1 experience?

Consider this story:

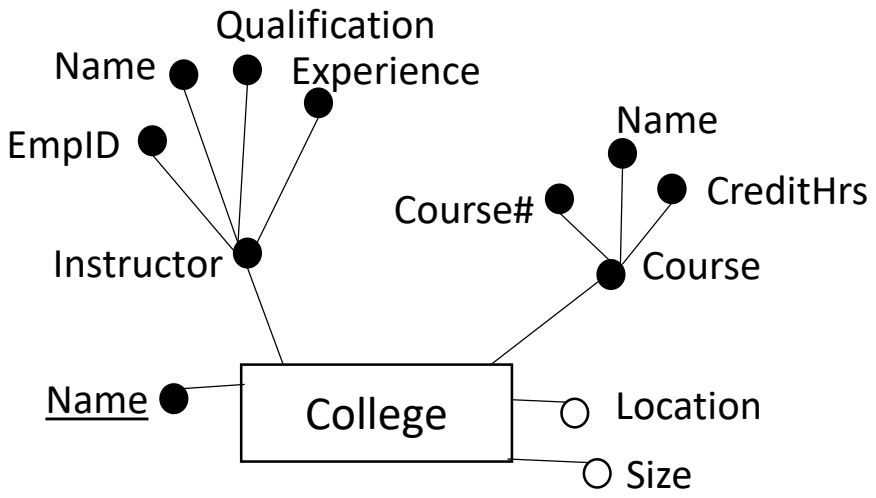
- There are several colleges in the university. Each college has a **Name**, **Location**, and **Size**. A college offers many **courses**, which are described by Course#, Name, and Credit Hours. The college also has several **instructors**. Instructors teach courses, but not all instructors are scheduled to teach during all terms. All courses must be taught by an instructor. Instructors are capable of teaching a multiple courses offered by the college. Instructors have a unique Employee ID and their Name, Qualification, and Experience are also recorded.

- Note: This is a simplified version of the vignette on page 120.

Working through the two vignettes presented on pages 120-133 is a good exercise/exam practice

Consider this story:

- College
  - Name
  - Location
  - Size
  - Courses
    - Course#
    - Name
    - CreditHrs
  - Instructor
    - Name
    - EmpID
    - Qualification
    - Experience



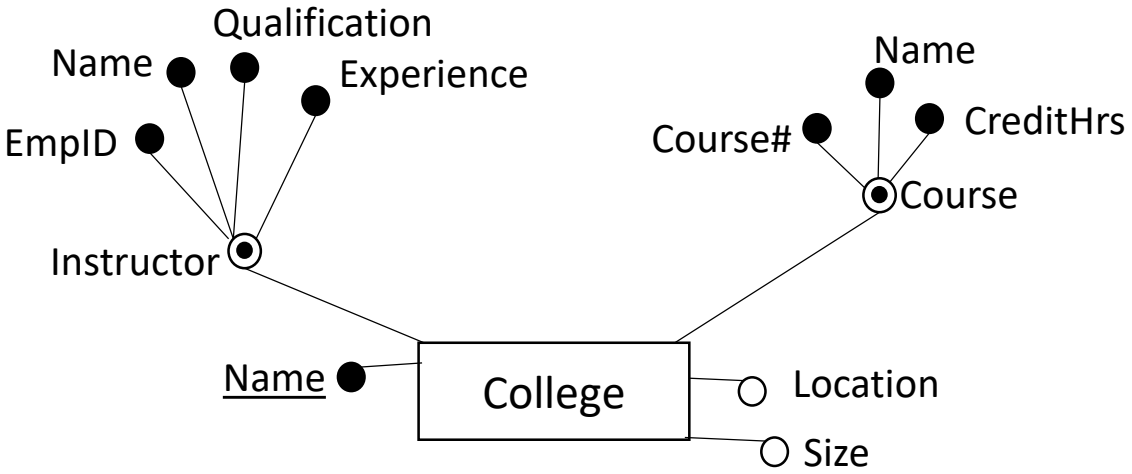
This is syntactically OK

(**But incorrect**)

Semantic Problem: college can have Multiple Instructors and Courses

Consider this story:

- College
  - Name
  - Location
  - Size
  - Courses
    - Course#
    - Name
    - CreditHrs
  - Instructor
    - Name
    - EmpID
    - Qualification
    - Experience



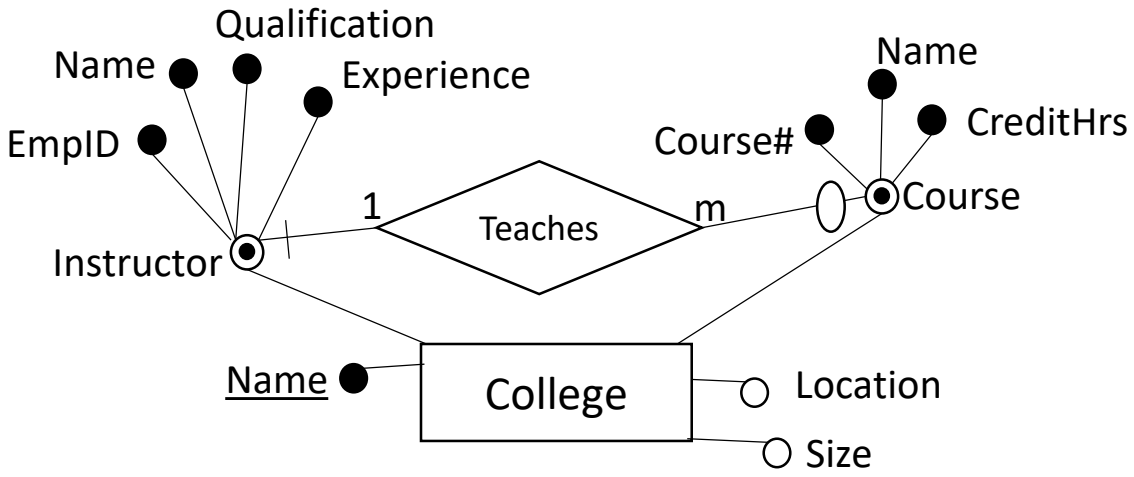
This is syntactically OK

(**But incorrect**)

Semantic Problem: Instructors teach courses

Consider this story:

- College
  - Name
  - Location
  - Size
  - Courses
    - Course#
    - Name
    - CreditHrs
  - Instructor
    - Name
    - EmpID
    - Qualification
    - Experience



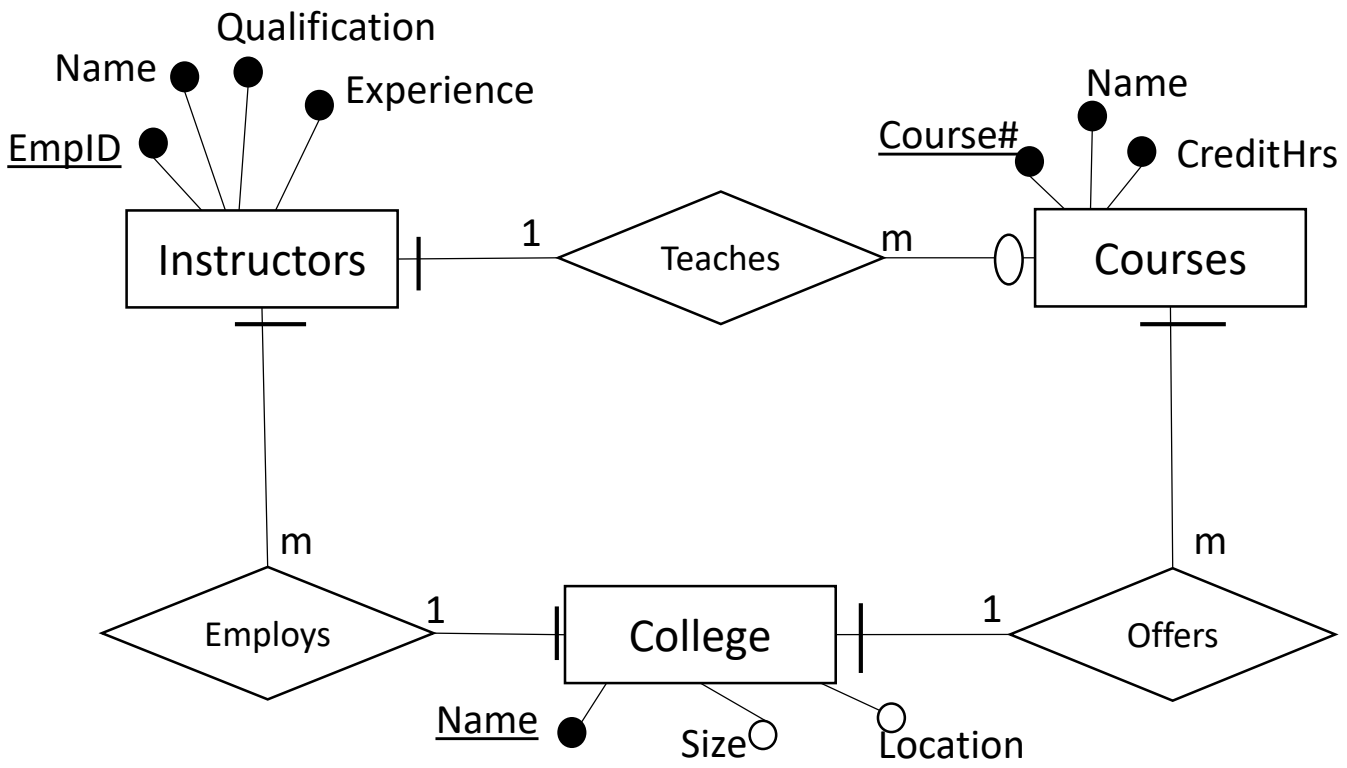
This is syntactically Incorrect

How do we properly model multi-value attributes?

Semantic Problems have been resolved

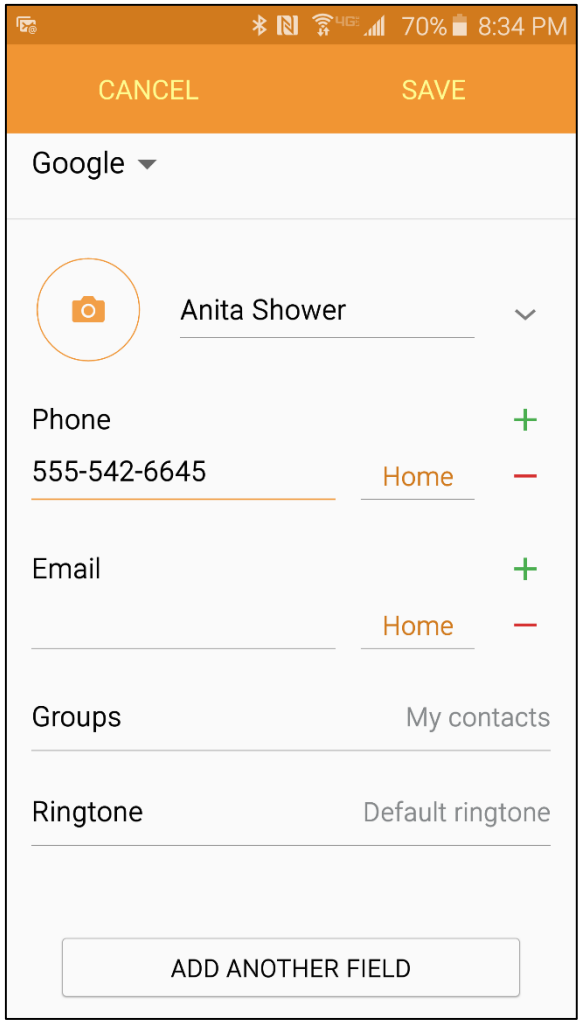
Consider this story:

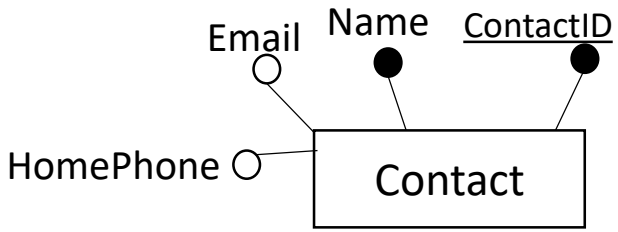
- College
  - Name
  - Location
  - Size
  - Courses
    - Course#
    - Name
    - CreditHrs
  - Instructor
    - Name
    - EmpID
    - Qualification
    - Experience



This is syntactically correct and semantic Problems have been resolved

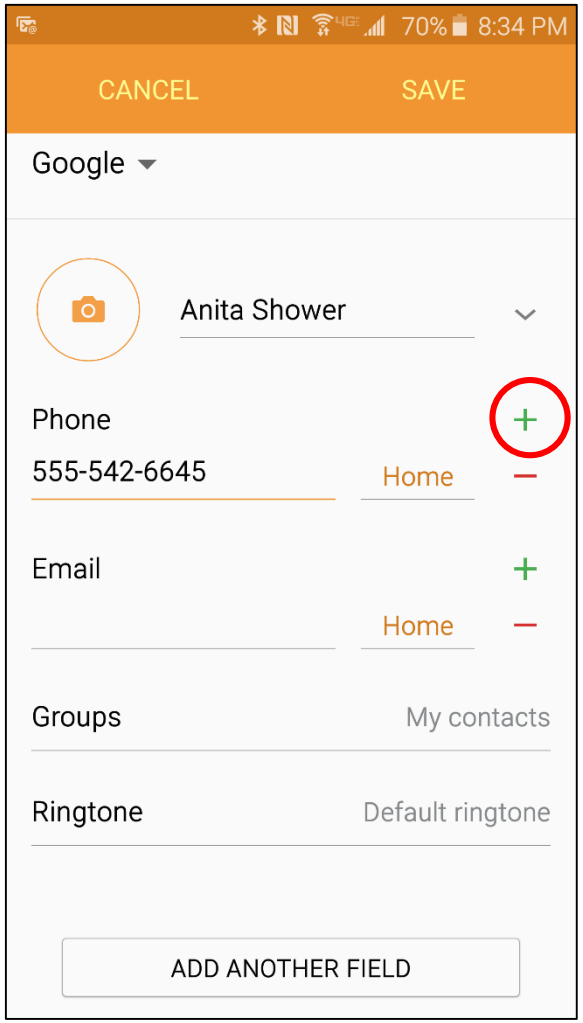
Consider this story: Contacts list on my phone

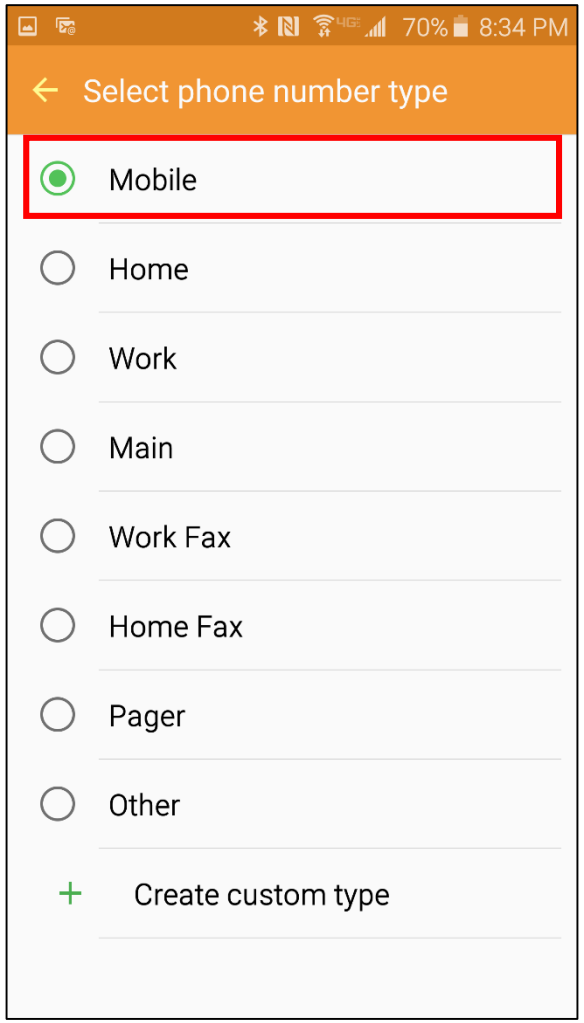


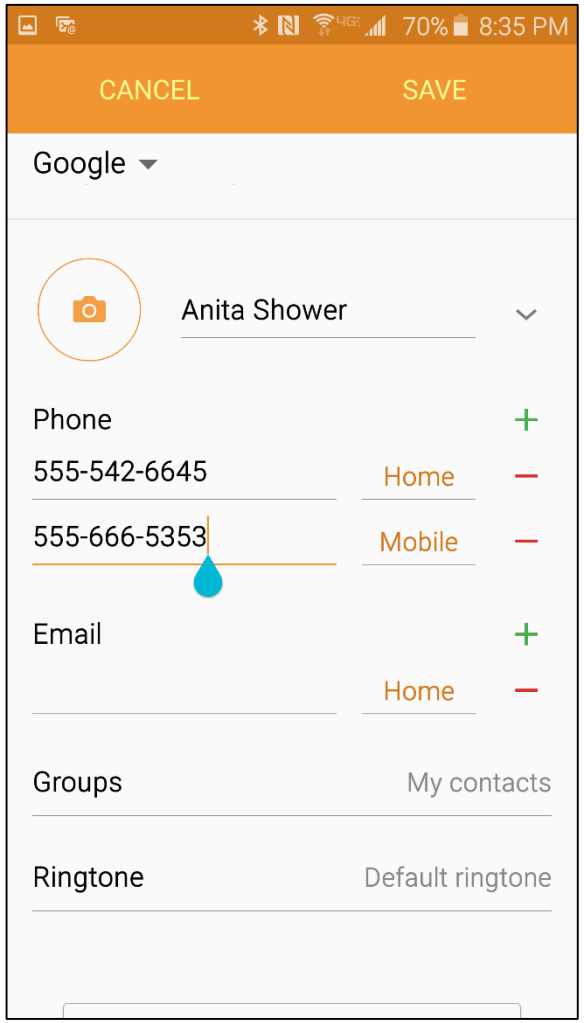


This seems reasonable and is syntactically correct, but does it capture all the business rules?

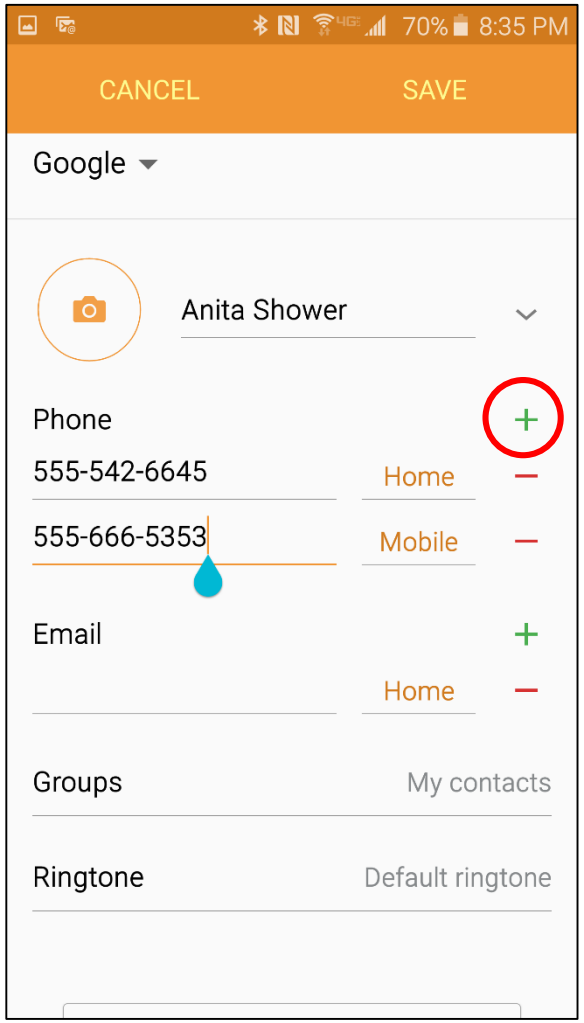
Consider this story: Contacts list on my phone

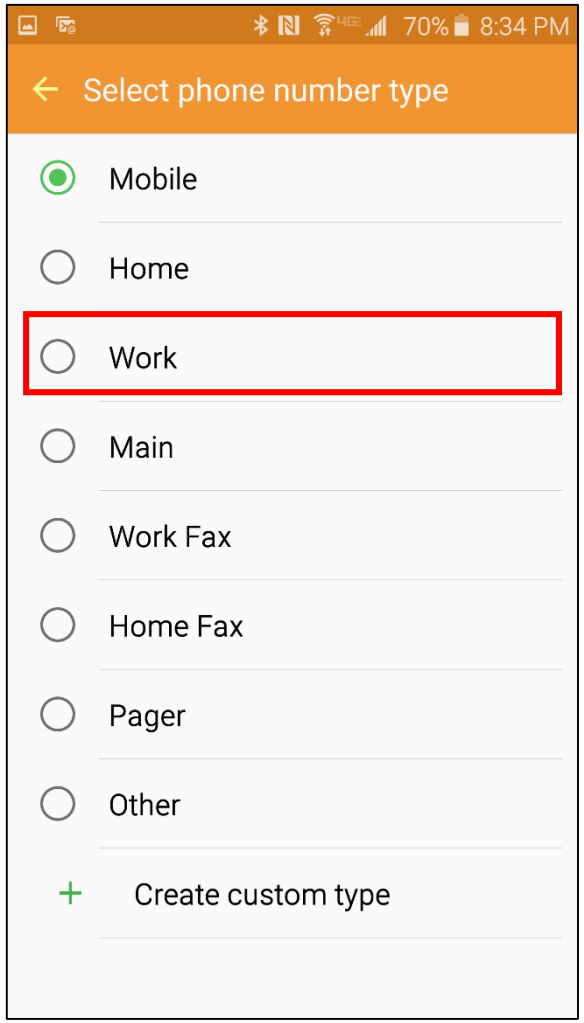


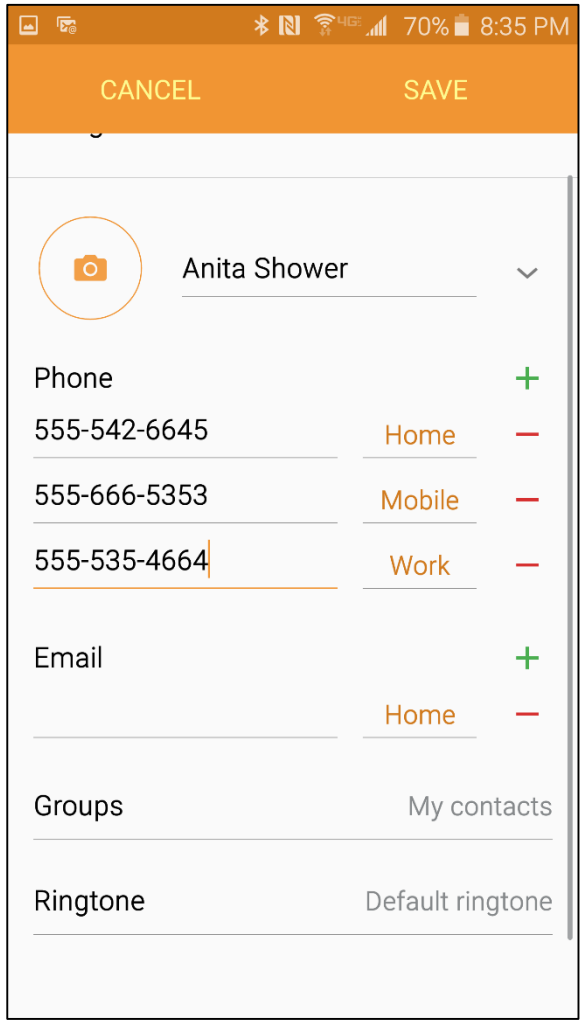




Contacts list on my phone

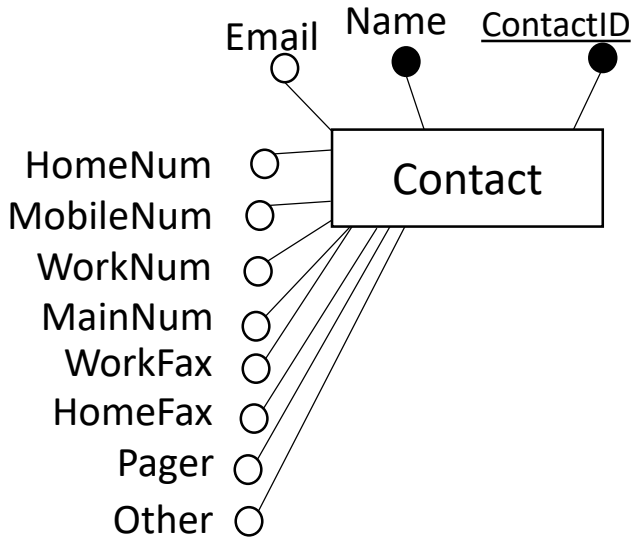




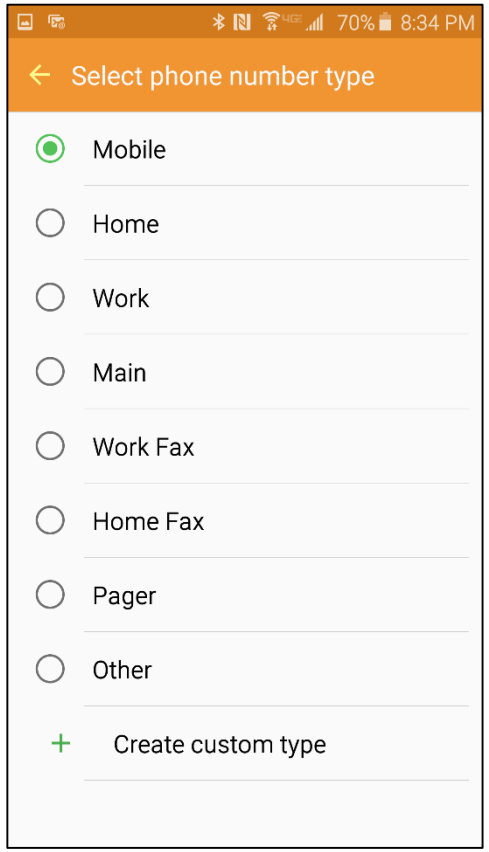


Perhaps this is the data structure…

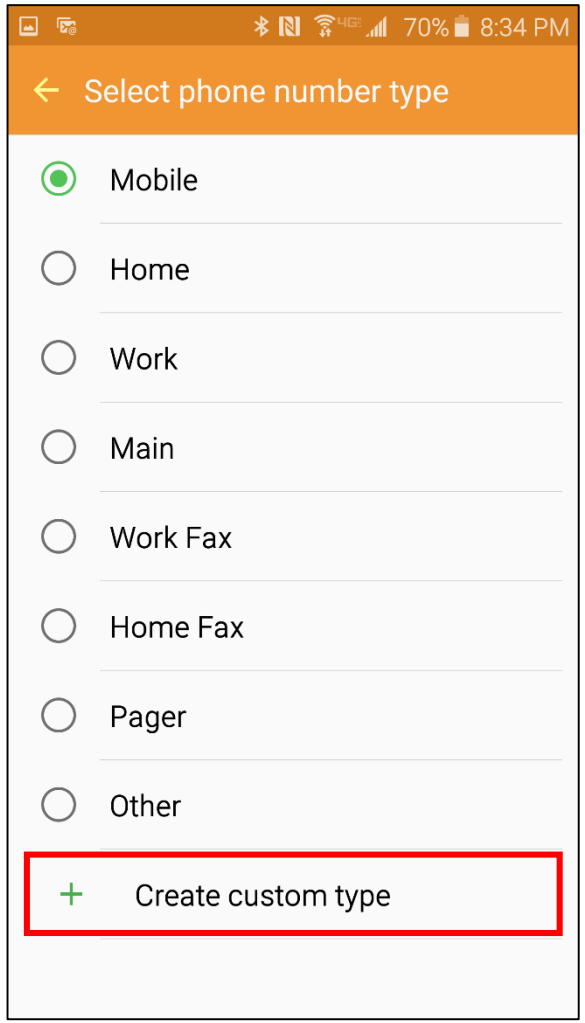
Name	HomeNum	MobileNum	WorkNum	MainNum	WorkFax	HomeFax	Pager	Other
Anita Shower	555-123-2134							
Jane Smith	555-234-4344	555-234-3434	555-334-3434					
Tom Powers		555-345-3523						
Dave Rogers	555-234-3995		555-334-3434					

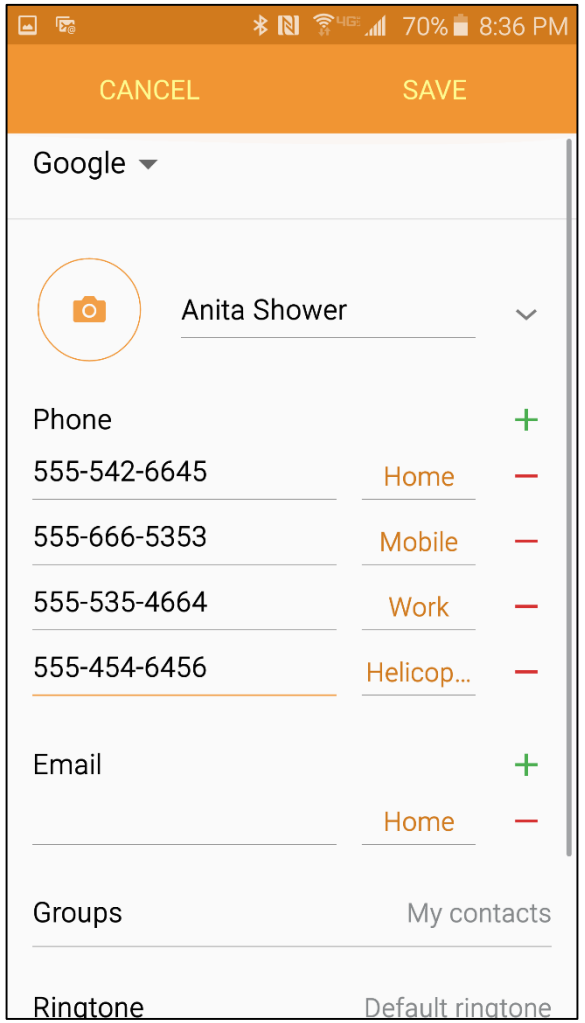


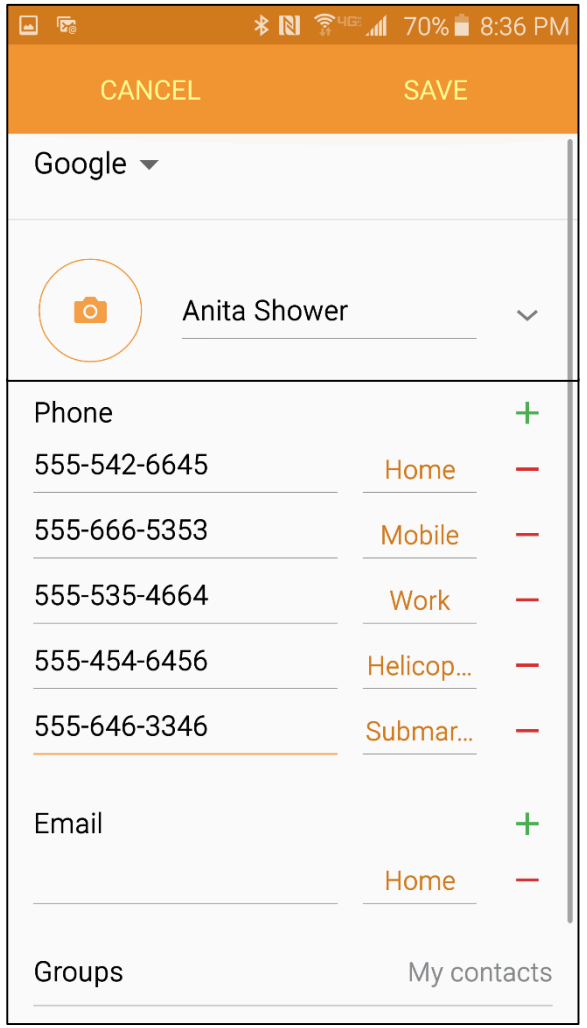
This seems reasonable and is syntactically correct, but does it capture all the business rules?



I can create any type of number





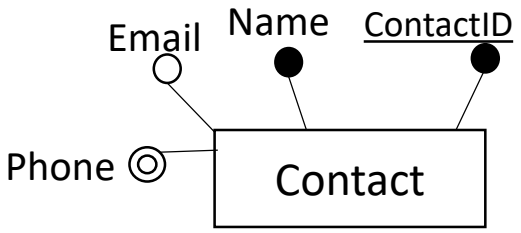


How should this be modeled?

- Does this data model still make sense?

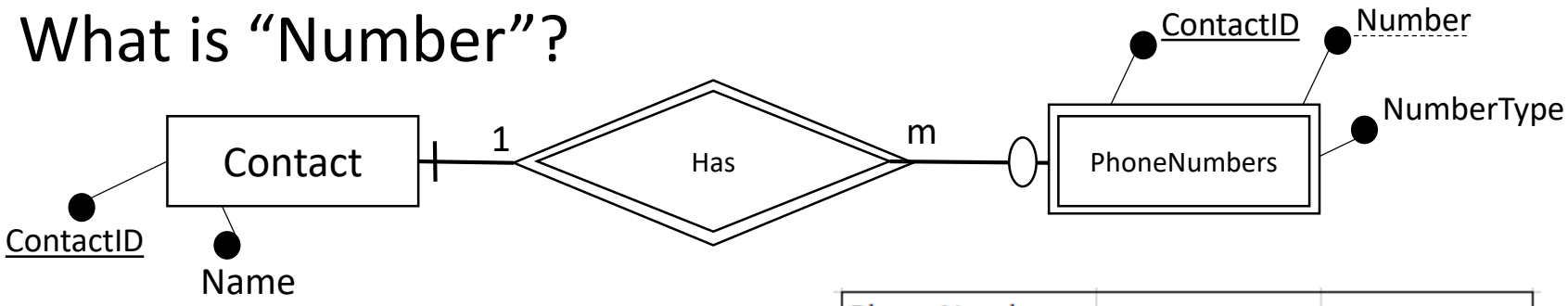
Name	HomeNum	MobileNum	WorkNum	MainNum	WorkFax	HomeFax	Pager	Other
Anita Shower	555-123-2134							
Jane Smith	555-234-4344	555-234-3434	555-334-3434					
Tom Powers		555-345-3523						
Dave Rogers	555-234-3995		555-334-3434					

- Perhaps we do not have multiple phone number attributes, but rather phone number is… a multi-value attribute!



…a multi-value attribute

- What type of entity is “PhoneNumbers”?
- What is “Number”?



Contacts

ContactID	Name
101	Anita Shower
102	Jane Smith
103	Tom Powers
104	Dave Rogers

PhoneNumbers

ContactID	Number	NumberType
101	555-123-2134	Home
101	555-343-5235	Mobile
101	555-266-6463	Helicopter
101	555-234-2133	Submarine
102	555-234-4344	Home
102	555-234-5522	Mobile
102	555-334-3434	Work
103	555-345-3523	Mobile
104	555-234-3995	Home
104	555-334-3434	Work

Assignment 1

- This was one of the goals of assignment 1 – to look at a system and infer what the schema might look like.

- As an outsider looking in, you will likely be wrong. Why?
  - You only see the external schema – many of the details and business rules are hidden from you

- It can sometimes be difficult to determine if something should be an entity, attribute, or value
  - An entity is a group of related attributes
  - Imagine what an instance of the entity would be
  - Try to create values for each attribute for each instance you think of

Module 3.3
Modeling Errors

- What are semantic errors?
- What are syntactical errors?

Let’s apply what we have learned to MGHH!

MG's Horse Habitat – Horses and Customers

Horses

MG's Horse Habitat (MGHH) is a ranch located just outside of Houston, Texas. When horses arrive at MGHH their name (which must be unique) and sex (M or F) is always recorded. Other descriptive attributes of the horse including whether they have spots (Yes or No), color (Black, White, Brown, Grey, Red, or Yellow), and weight are optional. If color and spots are not recorded at the time of arrival a default value of "UNK" is recorded. The value of weight must between 800 and 2200 pounds. If a value of weight is not recorded at the time of arrival the value is left blank until the horse can be weighed by a vet.

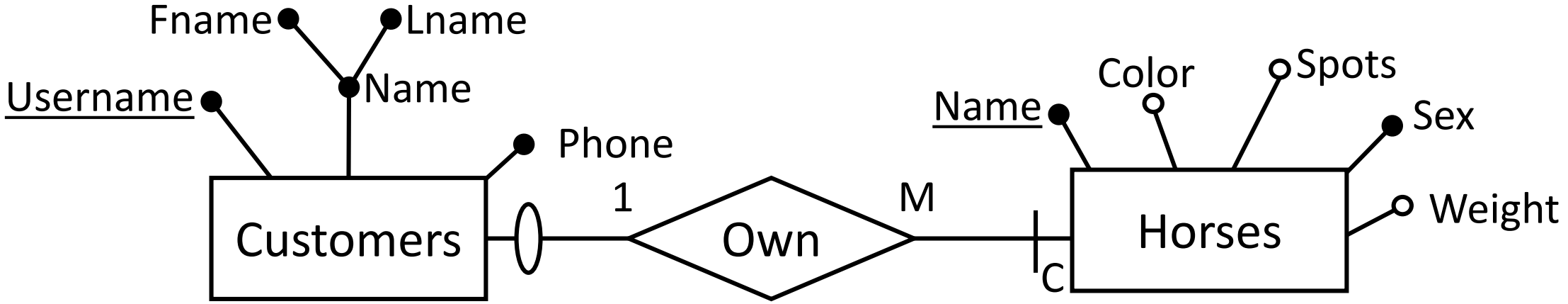
Customers

When a customer brings their horse to MGHH or purchases a horse from MGHH, their name, phone number, and a unique username (usually first initial + last name) are entered into the system.

Relationship

Some horses are the property of MGHH, while other horses that stay at the habitat are owned by customers. That is to say, not all horses have owners. All customers of MGHH own at least one horse and might own several horses. If a customer leaves MGHH, all horses they own should be deleted from the database.

MG’s Horse Habitat – Horses and Customers



Semantic Integrity Constraints:
Color: {Black, White, Brown, Grey, Red, Yellow}, Default: 'UNK'
Spots: {Yes, No}, Default: 'UNK’
Sex: {M, F}
Weight: {800-2200}

Let’s have a fresh start by recreating all we’ve done so far…

- The next few slides have the EXACT same SQL we’ve done so far, but in case you want everything in one easy to find spot, we can start over from here!

- Drop our existing tables:

```
DROP TABLE horses;
DROP TABLE customers;
```

Recreate the Customers and Horses tables with all constraints

```
CREATE TABLE customers
(username varchar(50) CONSTRAINT pk_customers PRIMARY KEY,
 Fname    varchar(50) CONSTRAINT nn_fname NOT NULL,
 Lname    varchar(50) CONSTRAINT nn_lname NOT NULL,
 Phone    varchar(14) CONSTRAINT nn_Phone NOT NULL
);

CREATE TABLE horses
(Name   varchar(50) CONSTRAINT pk_horse PRIMARY KEY,
 Color  varchar(50) DEFAULT 'UNK' CONSTRAINT chk_color CHECK (color IN
 ('Black','White','Brown','Grey','Red','Yellow', 'UNK')),
 Spots  varchar(3) DEFAULT 'UNK',
 Sex    varchar(1) CONSTRAINT nn_sex NOT NULL,
 Weight integer,
 owner varchar(50),
 CONSTRAINT chk_weight CHECK (weight >= 800 AND weight <=2200),
 CONSTRAINT chk_sex CHECK (sex IN ('M','F')),
 CONSTRAINT fk_cust FOREIGN KEY (owner) REFERENCES customers (username) ON DELETE CASCADE
);
```

INSERTing all the data we've worked with so far:

```
INSERT INTO customers (username, fname, lname, phone) VALUES ('mgrimes', 'Marvin', 'Grimes', '(218) 330-8004');
INSERT INTO customers (username, fname, lname, phone) VALUES ('canderson', 'Christine', 'Anderson', '(555) 523-9989');
INSERT INTO customers (username, fname, lname, phone) VALUES ('tswift', 'Tina', 'Swift', '(555) 424-1313');
INSERT INTO customers (username, fname, lname, phone) VALUES ('jisbell', 'Jason', 'Isbell', '(615) 555-5555');
INSERT INTO customers (username, fname, lname, phone) VALUES ('ssimpson', 'Sam', 'Simpson', '(615) 387-9682');

INSERT INTO horses (name, color, spots, sex, weight, owner) VALUES ('Sam', 'Brown', 'No', 'F', 1500, 'mgrimes');
INSERT INTO horses (name, color, spots, sex, weight, owner) VALUES ('Erica', 'Yellow', 'Yes', 'F', 920, 'canderson');
INSERT INTO horses (name, color, spots, sex, weight, owner) VALUES ('John', 'Grey', 'No', 'M', 1800, 'mgrimes');
INSERT INTO horses (name, color, spots, sex, weight, owner) VALUES ('Trotty', 'Brown', 'Yes', 'M', 1300, 'mgrimes');
INSERT INTO horses (name, color, spots, sex, weight, owner) VALUES ('Rio', 'Grey', 'No', 'F', 1700, 'tswift');
INSERT INTO horses (name, color, spots, sex, weight, owner) VALUES ('Robin', 'Yellow', 'No', 'M', 1100, 'jisbell');
INSERT INTO horses (name, color, spots, sex, weight, owner) VALUES ('Katy', 'Brown', 'No', 'F', 1200, 'jisbell');
INSERT INTO horses (name, color, spots, sex, weight, owner) VALUES ('Pegasus', 'Brown', 'No', 'M', 1750, 'mgrimes');
INSERT INTO horses (name, color, spots, sex, weight, owner) VALUES ('Sammy', 'Black', 'Yes', 'M', 2200, 'mgrimes');
INSERT INTO horses (name, color, spots, sex, weight, owner) VALUES ('Pinky', 'Red', 'No', 'M', 1050, 'tswift');
INSERT INTO horses (name, color, spots, sex, weight, owner) VALUES ('Hulk', 'Grey', 'No', 'M', 2050, 'mgrimes');
INSERT INTO horses (name, color, spots, sex, weight, owner) VALUES ('Pat', 'White', 'No', 'F', 1400, 'mgrimes');
INSERT INTO horses (name, color, spots, sex, weight, owner) VALUES ('Betty', 'White', 'Yes', 'F', 1250, 'tswift');
INSERT INTO horses (name, color, spots, sex, weight, owner) VALUES ('Shamrock', 'Black', 'No', 'M', 1400, 'ssimpson');
```

Now on to the new stuff!

MG's Horse Habitat – Horses and Medications

While at MGHH, medications may be ordered for horses to treat a variety of illnesses, or to ensure their continued health.

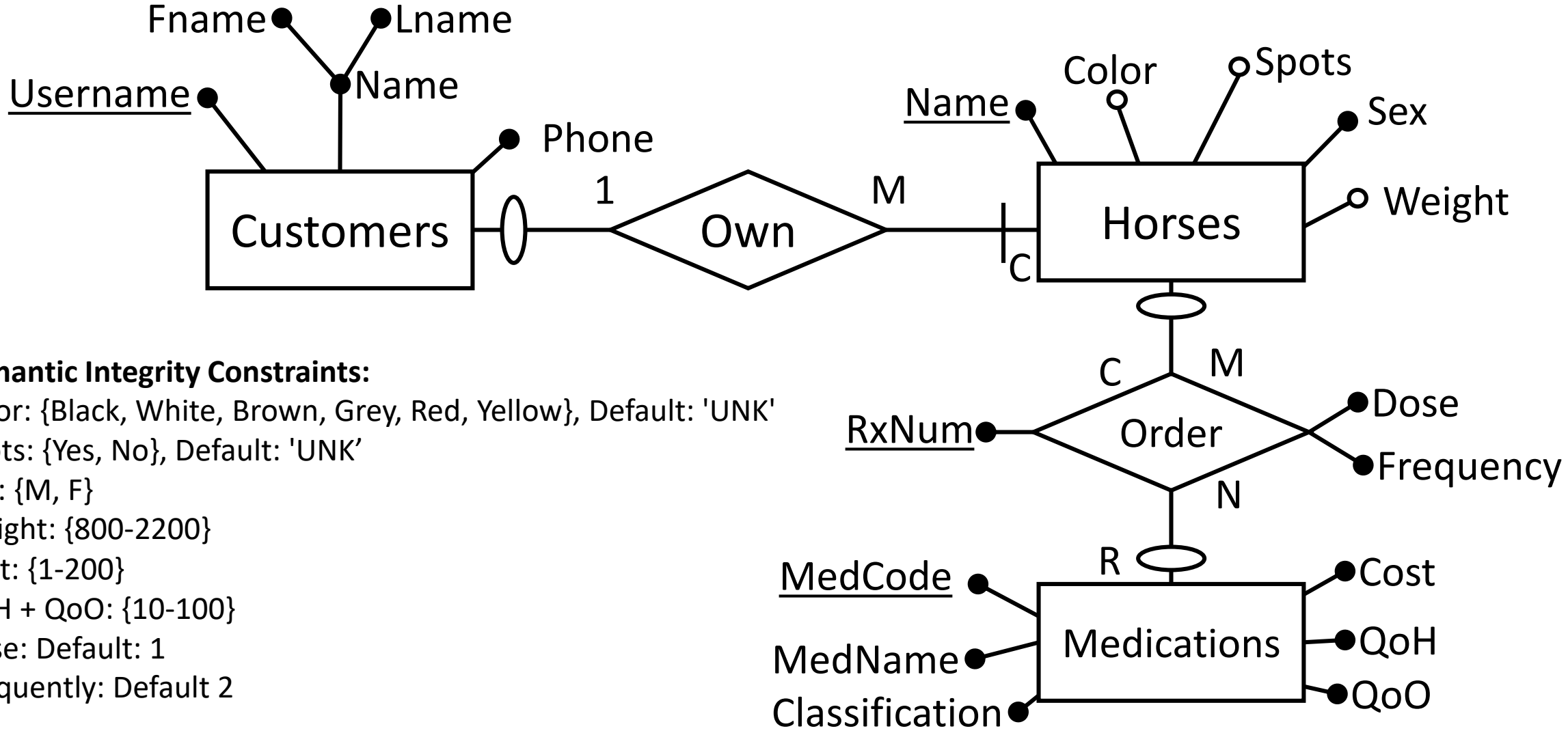
Medications are identified by a medication code (MedCode) but also have a commonly used name (for example Ivermectin or Ibuprofen). Each medication has a cost per dose that ranges from $1 to $200 and a classification that describes its use (NSAID, Antibiotic, Sedative, etc…).

MGHH keeps track of quantity on hand (QoH) as well as quantity on order (QoO). The value of QoH and QoO together must be between 10 and 100—that is, if QoH gets down to 10, then some units should be ordered to replenish the stock, but not to a quantity greater than 100.

When medications are ordered for horses, a dose and frequency should be recorded. If dose and frequency are not provided, a default dose of "1" should be used, with a frequency of "2" times per day. Orders are uniquely identified by a prescription number (RxNum).

If a horse is deleted from the database, all records of their medication orders should be deleted. A medication that has been ordered for a horse cannot be deleted from the database.

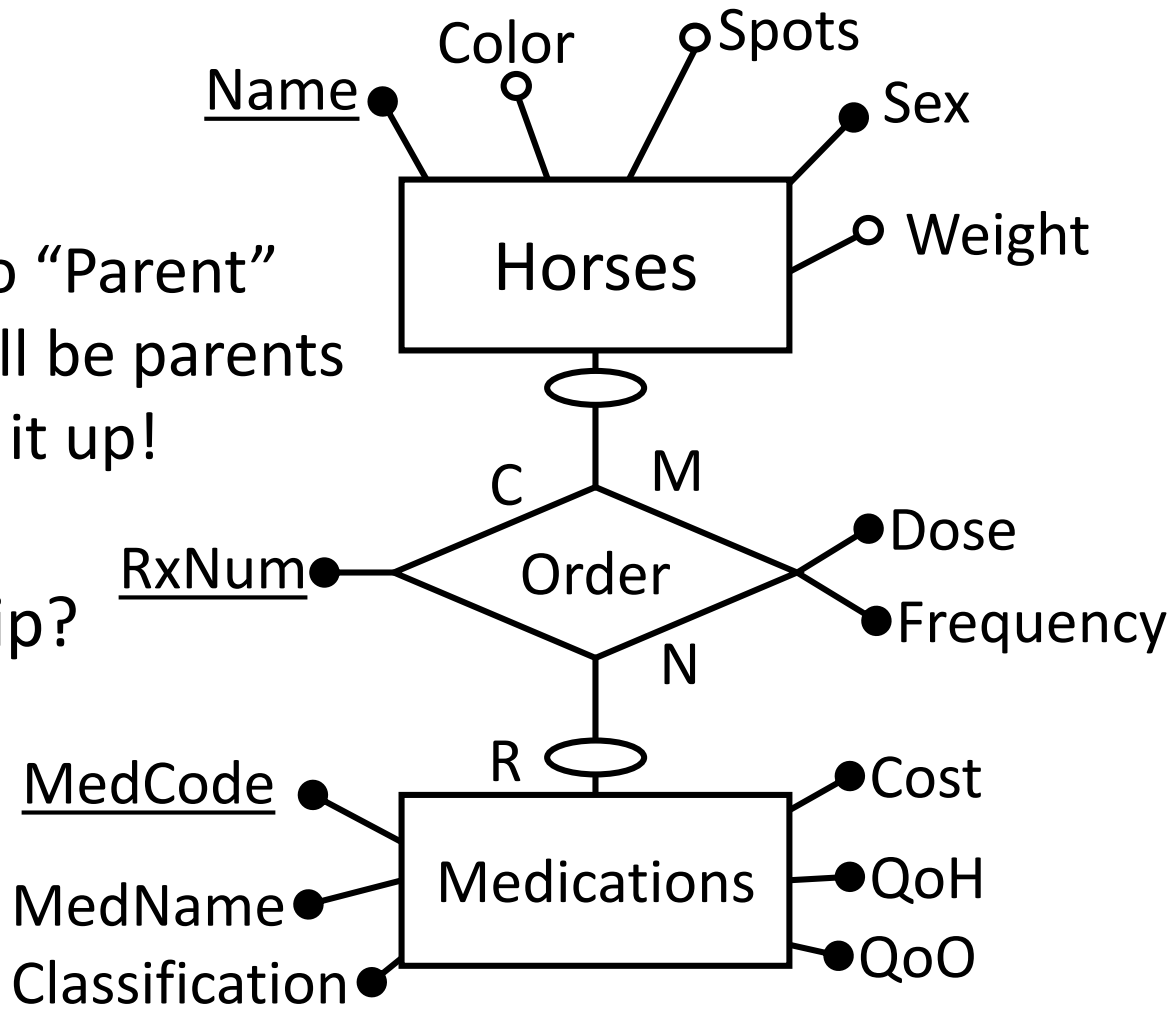
MG's Horse Habitat – Horses and Medications



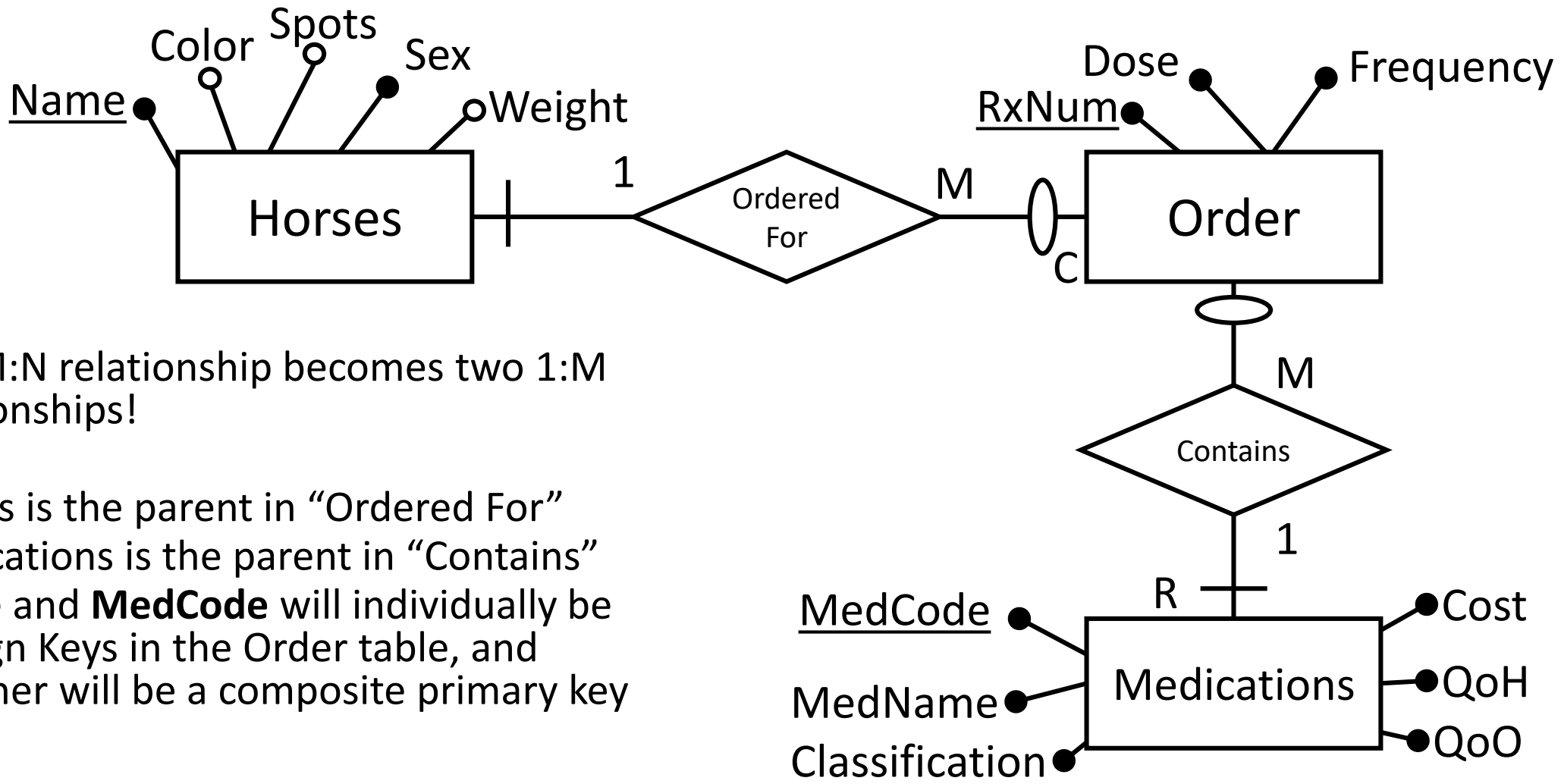
Semantic Integrity Constraints:
Color: {Black, White, Brown, Grey, Red, Yellow}, Default: 'UNK'
Spots: {Yes, No}, Default: 'UNK'
Sex: {M, F}
Weight: {800-2200}
Cost: {1-200}
QoH + QoO: {10-100}
Dose: Default: 1
Frequently: Default 2

MG's Horse Habitat – Horses and Medications

- Which entity is the "Parent" in this relationship?
  - Either One?
  - Neither One?
  - Both?
  - All of the above are correct – there is no "Parent" in a M:N relationship, because BOTH will be parents in the two 1:M Relationships that make it up!
- How do we model this M:N Relationship?
  - Using a Gerund to decompose into two 1:M relationships



MG's Horse Habitat – Horses and Medications

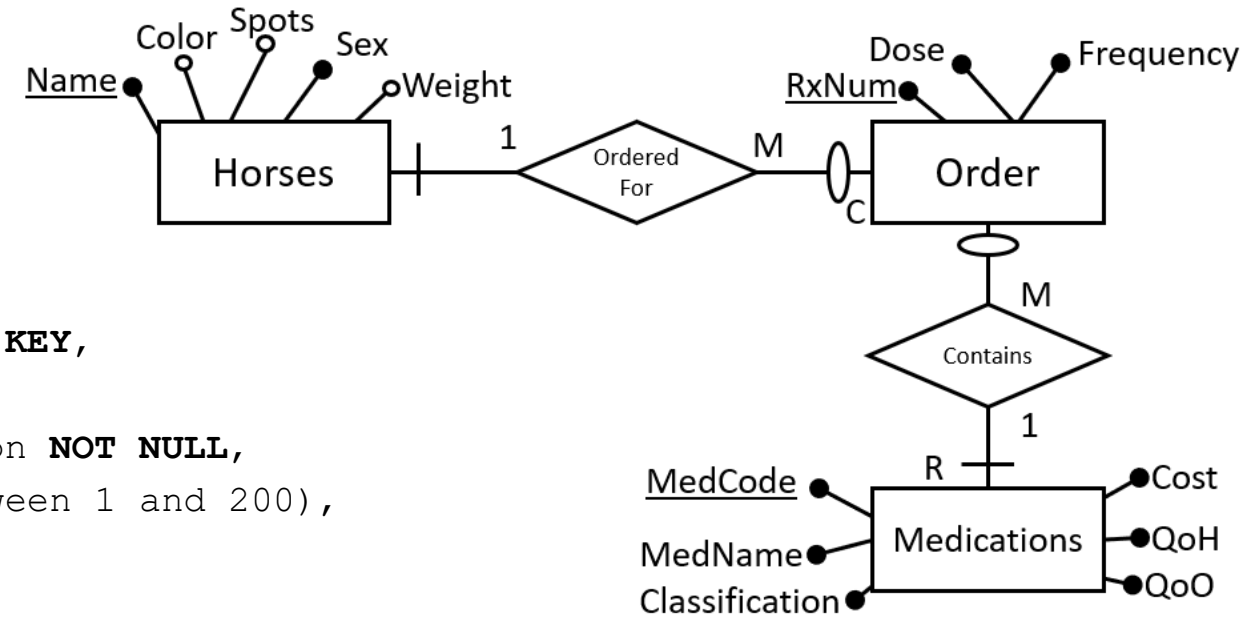


- The M:N relationship becomes two 1:M relationships!

- Horses is the parent in "Ordered For"
- Medications is the parent in "Contains"
- **Name** and **MedCode** will individually be Foreign Keys in the Order table, and together will be a composite primary key

MG's Horse Habitat – Horses and Medications

- First we need to CREATE the Medications table:




```
CREATE TABLE medications
(medcode varchar(50) CONSTRAINT pk_medications PRIMARY KEY,
 medname varchar(50) CONSTRAINT nn_medname NOT NULL,
 classification varchar(50) CONSTRAINT nn_classification NOT NULL,
 cost    float(32) CONSTRAINT chk_cost CHECK (cost between 1 and 200),
 QoH     integer,
 QoO     integer,
 CONSTRAINT chk_qty CHECK ((QoH + QoO) BETWEEN 10 AND 100)
);
```

- Note that we must create the Medications table before the Order table, since Order will reference Medications

MG’s Horse Habitat – Horses and Medications

- Let’s INSERT some horse medications:

```
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Flux','Flunixin Meglumine','NSAID',27,43,0);
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Bute','Phenylbutazone','NSAID',19,84,0);
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Oxi','Oxibuzone','NSAID',12,55,0);
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Mel','Meloxicam','NSAID',6,72,0);
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Pen','Penicillin','Antibiotic',38,73,0);
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Doxy','Doxycycline','Antibiotic',81,89,0);
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('TMPS','Trimethoprim Sulfa','Antibiotic',27,50,0);
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Enro','Enrofloxacin','Antibiotic',36,78,0);
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Xyl','Xylazine ','Sedative',22,28,0);
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Acep','Acepromazine','Sedative',33,50,0);
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Fen','Fenbendazole','Dewormer',52,15,25);
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Meb','Mebendazole','Dewormer',81,25,0);
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Pyr','Pyrantel Embonate','Dewormer',11,29,0);
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Iver','Ivermectin','Dewormer',39,21,0);
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Clen','Clenbuterol','Respiratory',132,11,20);
```

MG's Horse Habitat – Horses and Medications

- Note that we cannot insert a combined value greater that 100

```
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Pres','Prednisone','Corticosteroid',47,48,70);
```

SQL Error [2290] [23000]: ORA-02290:
check constraint
(GMGRIMES.CHK_QTY) violated

- …But we can if we fix this error by reducing the Quantity on Order (QoO):

```
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Pres','Prednisone','Corticosteroid',47,48,7);
```

- We also cannot create a medication with a cost greater than 200

```
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Dexa','Dexamethasone ','Corticosteroid',247,18,82);
```

SQL Error [2290] [23000]: ORA-02290:
check constraint
(GMGRIMES.CHK_COST) violated

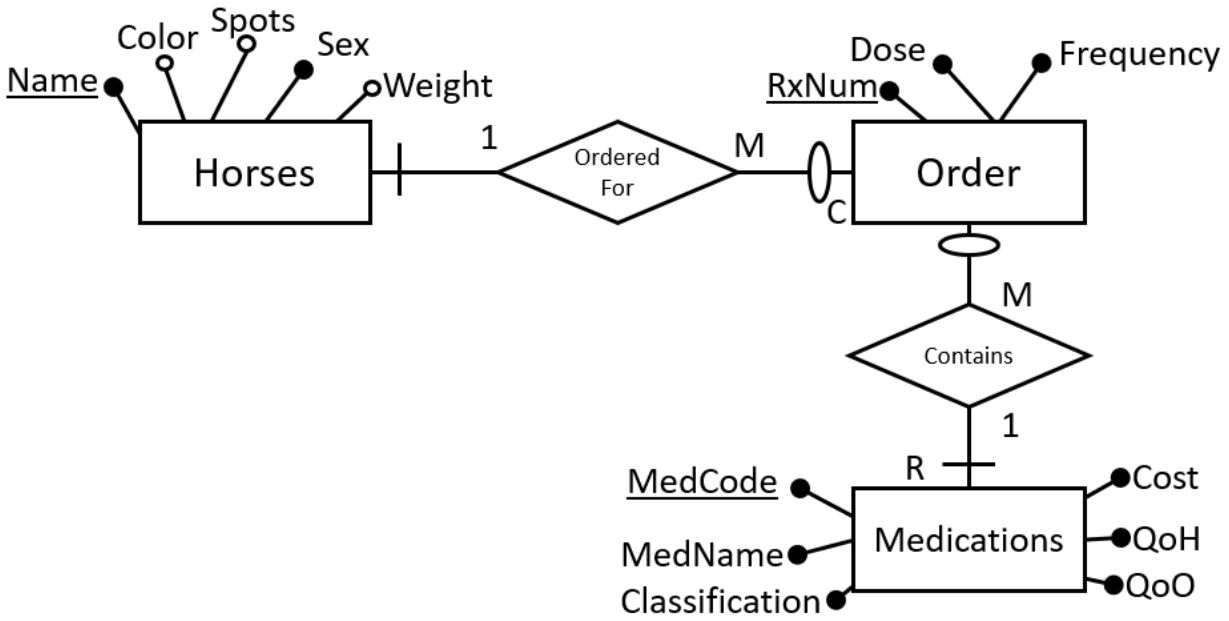
- Fixed:

```
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Dexa','Dexamethasone ','Corticosteroid',47,18,82);
```

MG's Horse Habitat – Horses and Medications

- Now let's CREATE the Order table

For your notes – there is an error in this DDL

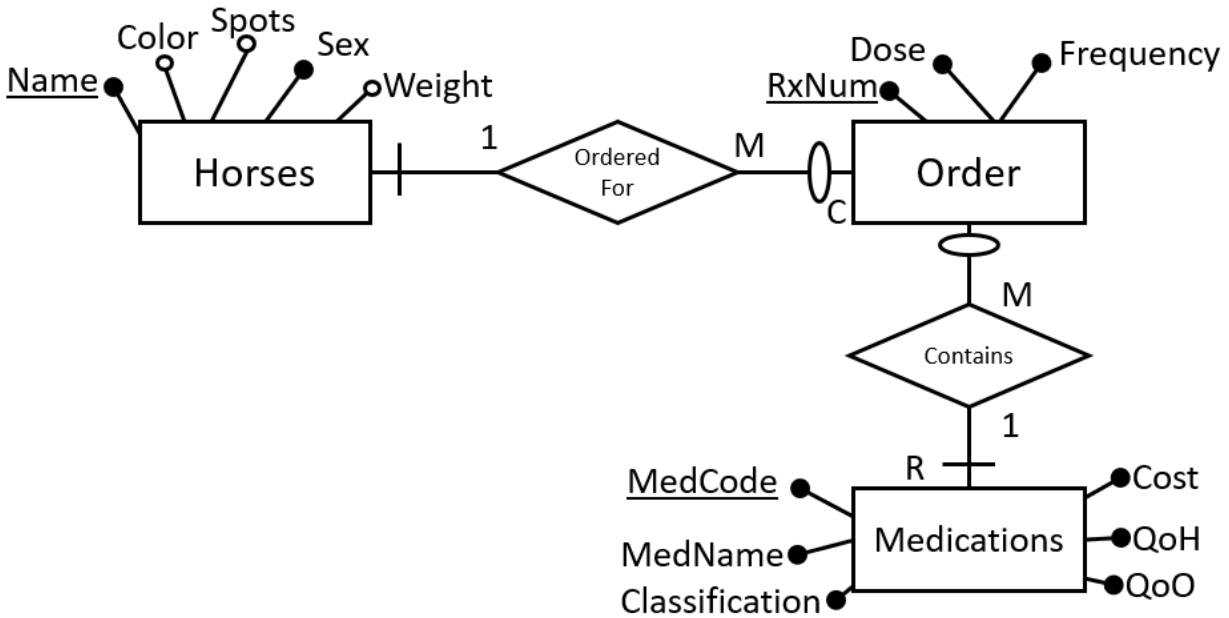



```
CREATE TABLE order
(RxNum integer CONSTRAINT pk_order PRIMARY KEY,
 ORD_Medcode varchar (50),
 ORD_Horsename varchar(50),
 Dose integer DEFAULT 1,
 Frequency integer DEFAULT 2,
 CONSTRAINT fk_meds FOREIGN KEY (ORD_Medcode) REFERENCES medications (medcode),
 CONSTRAINT fk_horses FOREIGN KEY (ORD_Horsename) REFERENCES horses (name) ON DELETE CASCADE
);
```

SQL Error [903] [42000]: ORA-00903: invalid table name

MG's Horse Habitat – Horses and Medications

- You cannot call the table "order" because "order" is a reserved keyword in SQL
  - We can just rename this table to be "orders" instead:




```
CREATE TABLE orders
(RxNum integer CONSTRAINT pk_orders PRIMARY KEY,
 ORD_Medcode varchar (50),
 ORD_Horsename varchar(50),
 Dose integer DEFAULT 1,
 Frequency integer DEFAULT 2,
 CONSTRAINT fk_meds FOREIGN KEY (ORD_Medcode) REFERENCES medications (medcode),
 CONSTRAINT fk_horses FOREIGN KEY (ORD_Horsename) REFERENCES horses (name) ON DELETE CASCADE
);
```

MG's Horse Habitat – Horses and Medications

- Let's INSERT some orders for medications:

```
INSERT INTO ORDERS (RxNum, ord_medcode, ord_horsename, dose, frequency) values (101,'Flux','Sam',1,2);
INSERT INTO ORDERS (RxNum, ord_medcode, ord_horsename, dose, frequency) values (102,'Pen','Sam',1,1);
INSERT INTO ORDERS (RxNum, ord_medcode, ord_horsename, dose, frequency) values (103,'Iver','John',2,1);
INSERT INTO ORDERS (RxNum, ord_medcode, ord_horsename, dose, frequency) values (104,'Iver','Trotty',2,1);
INSERT INTO ORDERS (RxNum, ord_medcode, ord_horsename, dose, frequency) values (105,'Doxy','Trotty',1,3);
INSERT INTO ORDERS (RxNum, ord_medcode, ord_horsename, dose, frequency) values (106,'Pres','Shamrock',1,1);
INSERT INTO ORDERS (RxNum, ord_medcode, ord_horsename, dose, frequency) values (107,'Dexa','Shamrock',2,1);
INSERT INTO ORDERS (RxNum, ord_medcode, ord_horsename, dose, frequency) values (108,'Dexa','Betty',2,1);
INSERT INTO ORDERS (RxNum, ord_medcode, ord_horsename, dose, frequency) values (109,'Xyl','Hulk',1,1);
INSERT INTO ORDERS (RxNum, ord_medcode, ord_horsename, dose, frequency) values (110,'Enro','Erica',3,2);
```

MG's Horse Habitat – Horses and Medications

- You cannot INSERT an order for a medication that does not exist:

```
INSERT INTO ORDERS (RxNum, ord_medcode, ord_horsename, dose, frequency) values (999,'NottaMed','Sam',1,2);
```

SQL Error [2291] [23000]: ORA-02291:
integrity constraint
(GMGRIMES.FK_MEDS) violated -
parent key not found

- Likewise you cannot INSERT an order for a horse that does not exist:

```
INSERT INTO ORDERS (RxNum, ord_medcode, ord_horsename, dose, frequency) values (888, 'Iver','NottaHorse',1,2);
```

SQL Error [2291] [23000]: ORA-02291:
integrity constraint
(GMGRIMES.FK_HORSES) violated -
parent key not found

- Remember: All values of a **Foreign Key** must be in the domain of values of the **Candidate Key** to which it refers

MG’s Horse Habitat – Horses and Medications

- You cannot DELETE a medication that has been ordered for a horse:

```
DELETE FROM Medications WHERE medcode = 'Iver’;
```

SQL Error [2292] [23000]: ORA-02292:
integrity constraint
(GMGRIMES.FK_MEDS) violated -
child record found

- But you CAN delete a medication that has NOT been ordered for a horse

```
DELETE FROM Medications WHERE medcode = 'Bute';
```

MG's Horse Habitat – Horses and Medications

- If you delete a horse (Parent), all records of that horses orders (Child) are also deleted (because of the CASCADE constraint applied to the foreign key)

	NAME	COLOR	SPOTS	SEX	WEIGHT	OWNER
1	Sam	Brown	No	F	1,500	mgrimes
2	Erica	Yellow	Yes	F	920	canderson
3	John	Grey	No	M	1,800	mgrimes
4	Trotty	Brown	Yes	M	1,300	mgrimes
5	Rio	Grey	No	F	1,700	tswift
6	Robin	Yellow	No	M	1,100	jisbell
7	Katy	Brown	No	F	1,200	jisbell
8	Pegasus	Brown	No	M	1,750	mgrimes
9	Sammy	Black	Yes	M	2,200	mgrimes
10	Pinky	Red	No	M	1,050	tswift
11	Hulk	Grey	No	M	2,050	mgrimes
12	Pat	White	No	F	1,400	mgrimes
13	Betty	White	Yes	F	1,250	tswift
14	Shamrock	Black	No	M	1,400	ssimpson

	RXNUM	ORD_MEDCODE	ORD_HORSENAME	DOSE	FREQUENCY
1	101	Flux	Sam	1	2
2	102	Pen	Sam	1	1
3	103	Iver	John	2	1
4	104	Iver	Trotty	2	1
5	105	Doxy	Trotty	1	3
6	106	Pres	Shamrock	1	1
7	107	Dexa	Shamrock	2	1
8	108	Dexa	Betty	2	1
9	109	Xyl	Hulk	1	1
10	110	Enro	Erica	3	2

```
DELETE FROM Horses WHERE name = 'Shamrock';
```

	NAME	COLOR	SPOTS	SEX	WEIGHT	OWNER
1	Sam	Brown	No	F	1,500	mgrimes
2	Erica	Yellow	Yes	F	920	canderson
3	John	Grey	No	M	1,800	mgrimes
4	Trotty	Brown	Yes	M	1,300	mgrimes
5	Rio	Grey	No	F	1,700	tswift
6	Robin	Yellow	No	M	1,100	jisbell
7	Katy	Brown	No	F	1,200	jisbell
8	Pegasus	Brown	No	M	1,750	mgrimes
9	Sammy	Black	Yes	M	2,200	mgrimes
10	Pinky	Red	No	M	1,050	tswift
11	Hulk	Grey	No	M	2,050	mgrimes
12	Pat	White	No	F	1,400	mgrimes
13	Betty	White	Yes	F	1,250	tswift

	RXNUM	ORD_MEDCODE	ORD_HORSENAME	DOSE	FREQUENCY
1	101	Flux	Sam	1	2
2	102	Pen	Sam	1	1
3	103	Iver	John	2	1
4	104	Iver	Trotty	2	1
5	105	Doxy	Trotty	1	3
6	108	Dexa	Betty	2	1
7	109	Xyl	Hulk	1	1
8	110	Enro	Erica	3	2

- This deletes ONE row from the Horses table and TWO rows from the Orders tables

MG's Horse Habitat – Horses and Medications

- If we want to see all our orders for medications, we can use the INNER JOIN introduced in Lecture 3 (we will talk much more about this after the exam)
  - Remember, all joins are binary (only two tables), so we first join Horses to Orders, then join that resulting table to Medications

```
SELECT * FROM Horses INNER JOIN orders ON name=ord_horsename
INNER JOIN medications ON ord_medcode = medcode;
```

	NAME	COLOR	SPOTS	SEX	WEIG	OWNER	RXNUM	ORI	ORD_	DOSI	FREQI	MEDCODE	MEDNAME	CLASSIFICATION	COST	QOH	QOO
1	Sam	Brown	No	F	1,500	mgrimes	101	Flux	Sam	1	2	Flux	Flunixin Meglumine	NSAID	27	43	0
2	Sam	Brown	No	F	1,500	mgrimes	102	Pen	Sam	1	1	Pen	Penicillin	Antibiotic	38	73	0
3	Trotty	Brown	Yes	M	1,300	mgrimes	105	Doxy	Trotty	1	3	Doxy	Doxycycline	Antibiotic	81	89	0
4	Erica	Yellow	Yes	F	920	canderson	110	Enro	Erica	3	2	Enro	Enrofloxacin	Antibiotic	36	78	0
5	Hulk	Grey	No	M	2,050	mgrimes	109	Xyl	Hulk	1	1	Xyl	Xylazine	Sedative	22	28	0
6	Trotty	Brown	Yes	M	1,300	mgrimes	104	Iver	Trotty	2	1	Iver	Ivermectin	Dewormer	39	21	0
7	John	Grey	No	M	1,800	mgrimes	103	Iver	John	2	1	Iver	Ivermectin	Dewormer	39	21	0
8	Shamrock	Black	No	M	1,400	ssimpson	106	Pres	Shamrock	1	1	Pres	Prednisone	Corticosteroid	47	48	7
9	Shamrock	Black	No	M	1,400	ssimpson	107	Dexa	Shamrock	2	1	Dexa	Dexamethasone	Corticosteroid	47	18	82
10	Betty	White	Yes	F	1,250	tswift	108	Dexa	Betty	2	1	Dexa	Dexamethasone	Corticosteroid	47	18	82

MG's Horse Habitat – Horses and Medications

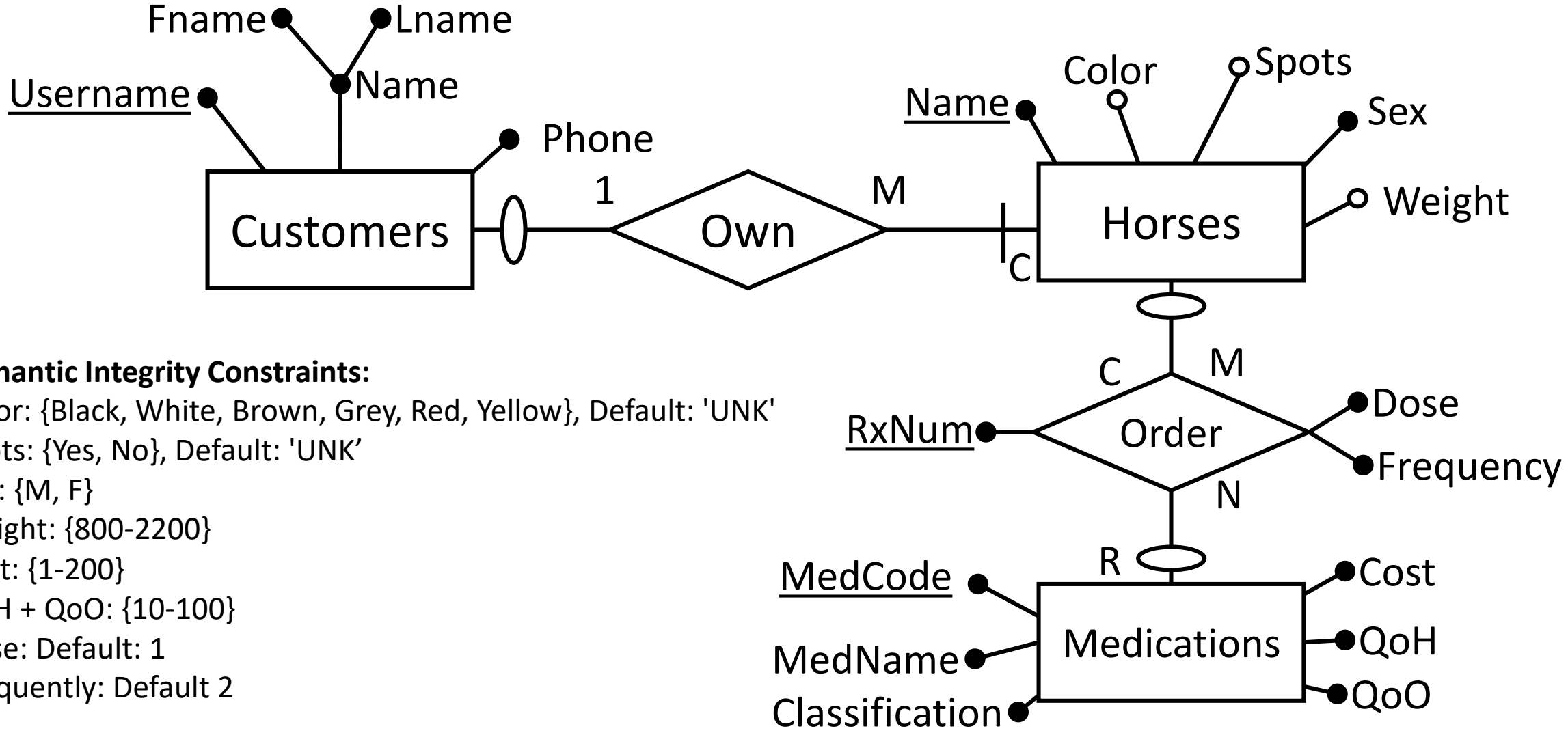
	NAME	COLOR	SPOTS	SEX	WEIG	OWNER	RXNUM	ORI	ORD_	DOSI	FREQ	MEDCODE	MEDNAME	CLASSIFICATION	COST	QOH	QOO
1	Sam	Brown	No	F	1,500	mgrimes	101	Flux	Sam	1	2	Flux	Flunixin Meglumine	NSAID	27	43	0
2	Sam	Brown	No	F	1,500	mgrimes	102	Pen	Sam	1	1	Pen	Penicillin	Antibiotic	38	73	0
3	Trotty	Brown	Yes	M	1,300	mgrimes	105	Doxy	Trotty	1	3	Doxy	Doxycycline	Antibiotic	81	89	0
4	Erica	Yellow	Yes	F	920	canderson	110	Enro	Erica	3	2	Enro	Enrofloxacin	Antibiotic	36	78	0
5	Hulk	Grey	No	M	2,050	mgrimes	109	Xyl	Hulk	1	1	Xyl	Xylazine	Sedative	22	28	0
6	Trotty	Brown	Yes	M	1,300	mgrimes	104	Iver	Trotty	2	1	Iver	Ivermectin	Dewormer	39	21	0
7	John	Grey	No	M	1,800	mgrimes	103	Iver	John	2	1	Iver	Ivermectin	Dewormer	39	21	0
8	Shamrock	Black	No	M	1,400	ssimpson	106	Pres	Shamrock	1	1	Pres	Prednisone	Corticosteroid	47	48	7
9	Shamrock	Black	No	M	1,400	ssimpson	107	Dexa	Shamrock	2	1	Dexa	Dexamethasone	Corticosteroid	47	18	82
10	Betty	White	Yes	F	1,250	tswift	108	Dexa	Betty	2	1	Dexa	Dexamethasone	Corticosteroid	47	18	82

- …There is a lot of extra information here, so we can "project" a subset of attributes we are interested in (again, we will talk much more about this after the exam) :

```
SELECT Horses.Name, medications.MEDNAME, orders.dose, orders.FREQUENCY  FROM Horses
INNER JOIN orders ON name=ord_horsename
INNER JOIN medications ON ord_medcode = medcode;
```

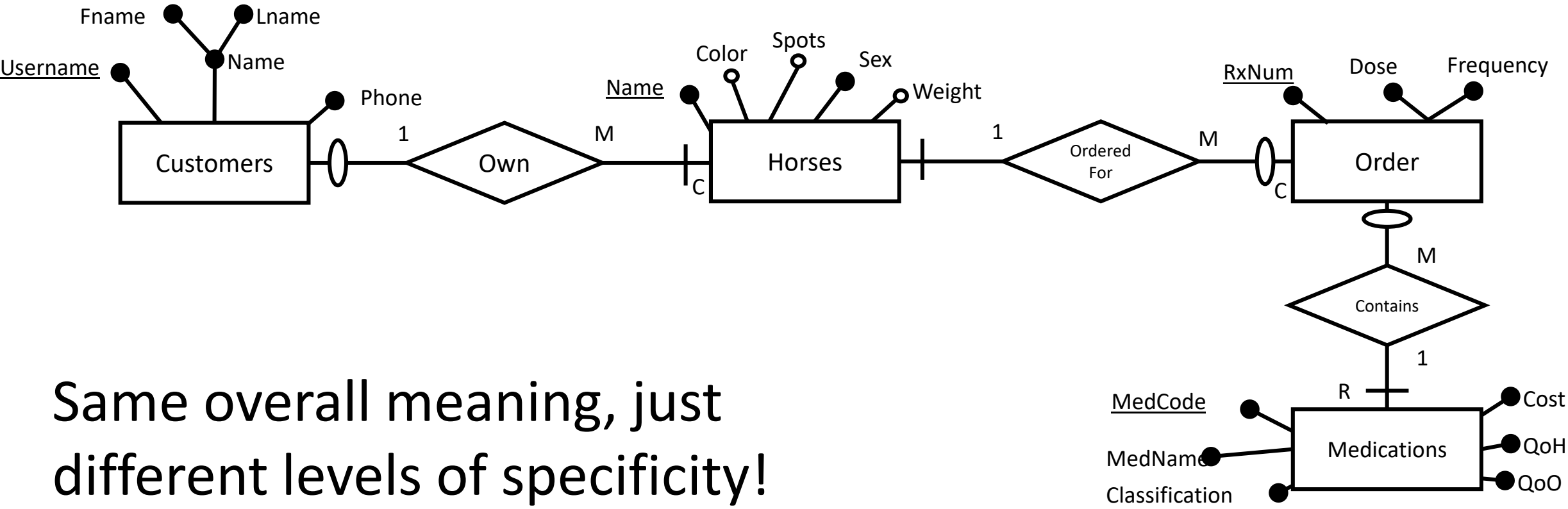
	NAME	MEDNAME	DOSE	FREQUENCY
1	Sam	Flunixin Meglumine	1	2
2	Sam	Penicillin	1	1
3	Trotty	Doxycycline	1	3
4	Erica	Enrofloxacin	3	2
5	Hulk	Xylazine	1	1
6	Trotty	Ivermectin	2	1
7	John	Ivermectin	2	1
8	Shamrock	Prednisone	1	1
9	Betty	Dexamethasone	2	1
10	Shamrock	Dexamethasone	2	1

Our current Conceptual ERD (and semantic Integrity Constraints)



Semantic Integrity Constraints:
Color: {Black, White, Brown, Grey, Red, Yellow}, Default: 'UNK'
Spots: {Yes, No}, Default: 'UNK'
Sex: {M, F}
Weight: {800-2200}
Cost: {1-200}
QoH + QoO: {10-100}
Dose: Default: 1
Frequently: Default 2

Our current decomposed ERD



Same overall meaning, just different levels of specificity!

Final DDL for now (Creating tables)

```
CREATE TABLE customers
(username varchar(50) CONSTRAINT pk_customers PRIMARY KEY,
 Fname    varchar(50) CONSTRAINT nn_fname NOT NULL,
 Lname    varchar(50) CONSTRAINT nn_lname NOT NULL,
 Phone    varchar(14) CONSTRAINT nn_Phone NOT NULL
);

CREATE TABLE horses
(Name   varchar(50) CONSTRAINT pk_horse PRIMARY KEY,
 Color  varchar(50) DEFAULT 'UNK' CONSTRAINT chk_color CHECK (color IN
 ('Black','White','Brown','Grey','Red','Yellow', 'UNK')),
 Spots  varchar(3) DEFAULT 'UNK',
 Sex    varchar(1) CONSTRAINT nn_sex NOT NULL,
 Weight integer,
 owner varchar(50),
 CONSTRAINT chk_weight CHECK (weight >= 800 AND weight <=2200),
 CONSTRAINT chk_sex CHECK (sex IN ('M','F')),
 CONSTRAINT fk_cust FOREIGN KEY (owner) REFERENCES customers (username) ON DELETE CASCADE
);

CREATE TABLE medications
(medcode varchar(50) CONSTRAINT pk_medications PRIMARY KEY,
 medname varchar(50) CONSTRAINT nn_medname NOT NULL,
 classification varchar(50) CONSTRAINT nn_classification NOT NULL,
 cost    float(32) CONSTRAINT chk_cost CHECK (cost between 1 and 200),
 QoH     integer,
 QoO     integer,
 CONSTRAINT chk_qty CHECK ((QoH + QoO) BETWEEN 10 AND 100)
);

CREATE TABLE orders
(RxNum integer CONSTRAINT pk_orders PRIMARY KEY,
 ORD_Medcode varchar (50),
 ORD_Horsename varchar(50),
 Dose integer DEFAULT 1,
 Frequency integer DEFAULT 2,
 CONSTRAINT fk_meds FOREIGN KEY (ORD_Medcode) REFERENCES medications (medcode),
 CONSTRAINT fk_horses FOREIGN KEY (ORD_Horsename) REFERENCES horses (name) ON DELETE CASCADE
);
```

Final DDL for now (Inserting data)

```
INSERT INTO customers (username, fname, lname, phone) VALUES ('mgrimes', 'Marvin', 'Grimes', '(218) 330-8004');
INSERT INTO customers (username, fname, lname, phone) VALUES ('canderson', 'Christine', 'Anderson', '(555) 523-9989');
INSERT INTO customers (username, fname, lname, phone) VALUES ('tswift', 'Tina', 'Swift', '(555) 424-1313');
INSERT INTO customers (username, fname, lname, phone) VALUES ('jisbell', 'Jason', 'Isbell', '(615) 555-5555');
INSERT INTO customers (username, fname, lname, phone) VALUES ('ssimpson', 'Sam', 'Simpson', '(615) 387-9682');

INSERT INTO horses (name, color, spots, sex, weight, owner) VALUES ('Sam', 'Brown', 'No', 'F', 1500, 'mgrimes');
INSERT INTO horses (name, color, spots, sex, weight, owner) VALUES ('Erica', 'Yellow', 'Yes', 'F', 920, 'canderson');
INSERT INTO horses (name, color, spots, sex, weight, owner) VALUES ('John', 'Grey', 'No', 'M', 1800, 'mgrimes');
INSERT INTO horses (name, color, spots, sex, weight, owner) VALUES ('Trotty', 'Brown', 'Yes', 'M', 1300, 'mgrimes');
INSERT INTO horses (name, color, spots, sex, weight, owner) VALUES ('Rio', 'Grey', 'No', 'F', 1700, 'tswift');
INSERT INTO horses (name, color, spots, sex, weight, owner) VALUES ('Robin', 'Yellow', 'No', 'M', 1100, 'jisbell');
INSERT INTO horses (name, color, spots, sex, weight, owner) VALUES ('Katy', 'Brown', 'No', 'F', 1200, 'jisbell');
INSERT INTO horses (name, color, spots, sex, weight, owner) VALUES ('Pegasus', 'Brown', 'No', 'M', 1750, 'mgrimes');
INSERT INTO horses (name, color, spots, sex, weight, owner) VALUES ('Sammy', 'Black', 'Yes', 'M', 2200, 'mgrimes');
INSERT INTO horses (name, color, spots, sex, weight, owner) VALUES ('Pinky', 'Red', 'No', 'M', 1050, 'tswift');
INSERT INTO horses (name, color, spots, sex, weight, owner) VALUES ('Hulk', 'Grey', 'No', 'M', 2050, 'mgrimes');
INSERT INTO horses (name, color, spots, sex, weight, owner) VALUES ('Pat', 'White', 'No', 'F', 1400, 'mgrimes');
INSERT INTO horses (name, color, spots, sex, weight, owner) VALUES ('Betty', 'White', 'Yes', 'F', 1250, 'tswift');
INSERT INTO horses (name, color, spots, sex, weight, owner) VALUES ('Shamrock', 'Black', 'No', 'M', 1400, 'ssimpson');

INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Flux','Flunixin Meglumine','NSAID',27,43,0);
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Bute','Phenylbutazone','NSAID',19,84,0);
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Oxi','Oxibuzone','NSAID',12,55,0);
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Mel','Meloxicam','NSAID',6,72,0);
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Pen','Penicillin','Antibiotic',38,73,0);
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Doxy','Doxycycline','Antibiotic',81,89,0);
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('TMPS','Trimethoprim Sulfa','Antibiotic',27,50,0);
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Enro','Enrofloxacin','Antibiotic',36,78,0);
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Xyl','Xylazine ','Sedative',22,28,0);
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Acep','Acepromazine','Sedative',33,50,0);
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Fen','Fenbendazole','Dewormer',52,15,25);
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Meb','Mebendazole','Dewormer',81,25,0);
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Pyr','Pyrantel Embonate','Dewormer',11,29,0);
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Iver','Ivermectin','Dewormer',39,21,0);
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Clen','Clenbuterol','Respiratory',132,11,20);
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Pres','Prednisone','Corticosteroid',47,48,7);
INSERT INTO Medications (medcode, medname, classification, cost, qoh, qoo) values ('Dexa','Dexamethasone ','Corticosteroid',47,18,82);

INSERT INTO ORDERS (RxNum, ord_medcode, ord_horsename, dose, frequency) values (101,'Flux','Sam',1,2);
INSERT INTO ORDERS (RxNum, ord_medcode, ord_horsename, dose, frequency) values (102,'Pen','Sam',1,1);
INSERT INTO ORDERS (RxNum, ord_medcode, ord_horsename, dose, frequency) values (103,'Iver','John',2,1);
INSERT INTO ORDERS (RxNum, ord_medcode, ord_horsename, dose, frequency) values (104,'Iver','Trotty',2,1);
INSERT INTO ORDERS (RxNum, ord_medcode, ord_horsename, dose, frequency) values (105,'Doxy','Trotty',1,3);
INSERT INTO ORDERS (RxNum, ord_medcode, ord_horsename, dose, frequency) values (106,'Pres','Shamrock',1,1);
INSERT INTO ORDERS (RxNum, ord_medcode, ord_horsename, dose, frequency) values (107,'Dexa','Shamrock',2,1);
INSERT INTO ORDERS (RxNum, ord_medcode, ord_horsename, dose, frequency) values (108,'Dexa','Betty',2,1);
INSERT INTO ORDERS (RxNum, ord_medcode, ord_horsename, dose, frequency) values (109,'Xyl','Hulk',1,1);
INSERT INTO ORDERS (RxNum, ord_medcode, ord_horsename, dose, frequency) values (110,'Enro','Erica',3,2);
```

Break

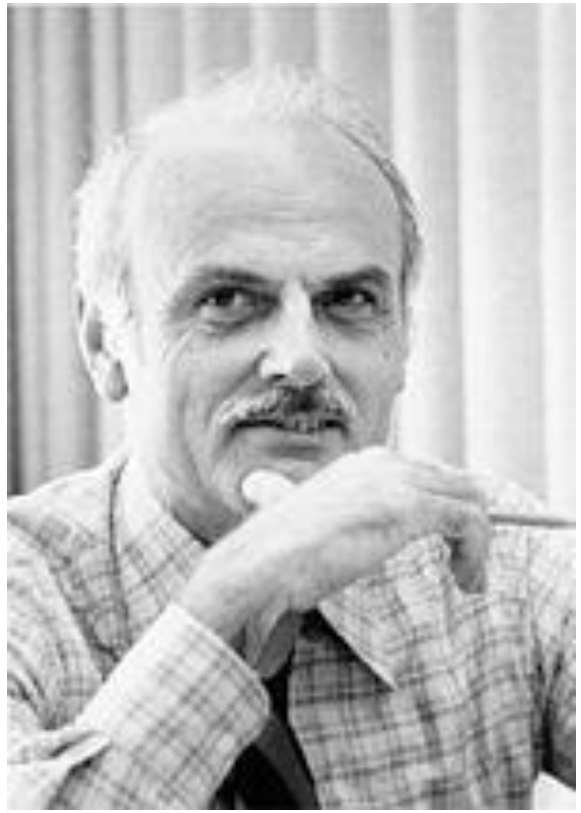
Module 6.1
The Relational Data Model

- Why do we care about logical data modeling?

- Be able to explain terms as used in Relational Data Models
  - Relation
  - Tuple
  - Cardinality
  - Degrees

The Relational Data Model

- Edgar F. Codd in 1970 used the concept of mathematical relations to define the relational data model
- Database: collection of relations
- Relation: two-dimensional table
- Tuple: row of related data values in the table
- Attribute: column in the table
- Domain: set of possible atomic values of an attribute
- Degree of the relation: number of attributes in a relation
- Cardinality of the relation: number of tuples in a relation



A bold, and true, statement:

- The foundation of modern database technology is without question the relational model; it is that foundation that makes the field a science. Thus any book on the fundamentals of database technology that does not include a thorough coverage of the relational model is by definition shallow. Likewise any claim to expertise in the database field can hardly be justified if the claimant does not understand the relational model in depth.

  Dale, C.J., Darwin, H., Foundation for Object/Relational Databases: The Third Manifesto

The Motivation for Logical Data Modeling

- Completion of **conceptual** modeling phase results in a picture of data requirements at high level of abstraction

- During conceptual modeling, we are not constrained by technology limitations that will be used for implementation
  - We got to use things like multi-value attributes and m:n relationships

- Conceptual schema may contain constructs not directly compatible with technology intended for implementation
  - Like multi-value attributes and m:n relationships!

The Motivation for Logical Data Modeling

- Further refinement may be required to eliminate data redundancy in design
  - Getting rid of multi-value attributes
  - Decomposing m:n relationships into 1:m using gerunds

- **Transforming conceptual schema to something that is more compatible with implementation technology of choice** is achieved via logical data modeling

- Logical data modeling phase serves as transition from technology-independent conceptual schema to technology-dependent design that can actually be implemented

- The next step will be generating the actual SQL code to create the entities, attributes, and relationships

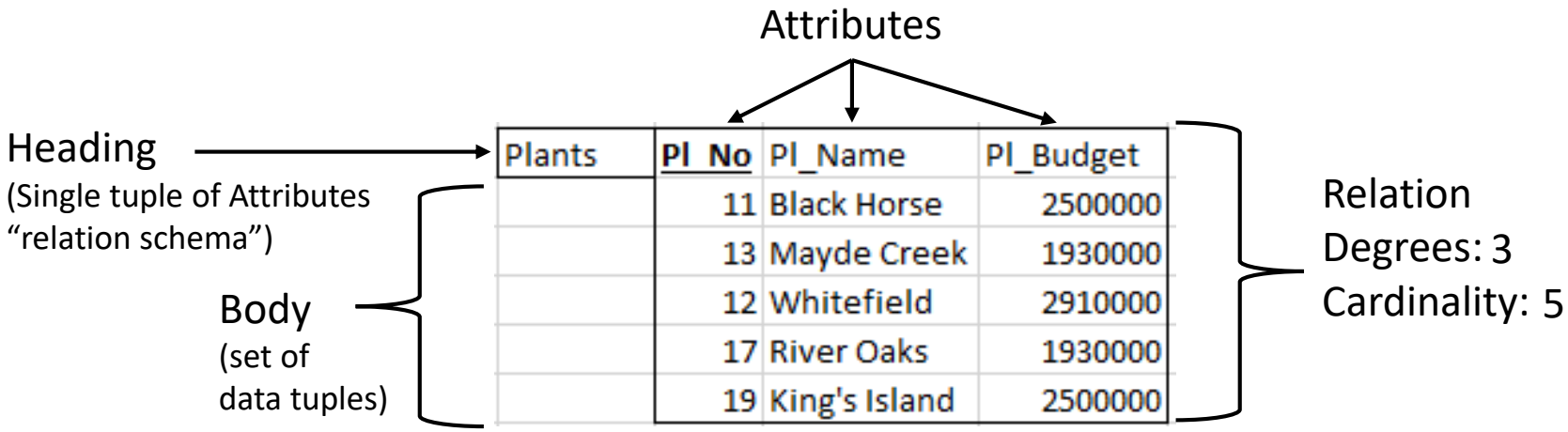
Warning

- This may be confusing at first!

- Conceptual modeling uses many of the same concepts/terms as logical modeling, but in a slightly different (but really the same) way
  - Cardinality
  - Degrees of relationships
  - “Relations” are different than “relationships”
    - But a relationship is a relation…

Relation: two-dimensional table

- Two components of a Relation
  - Heading – a single tuple listing the attributes (Relation Schema)
  - Body – collection of data tuples

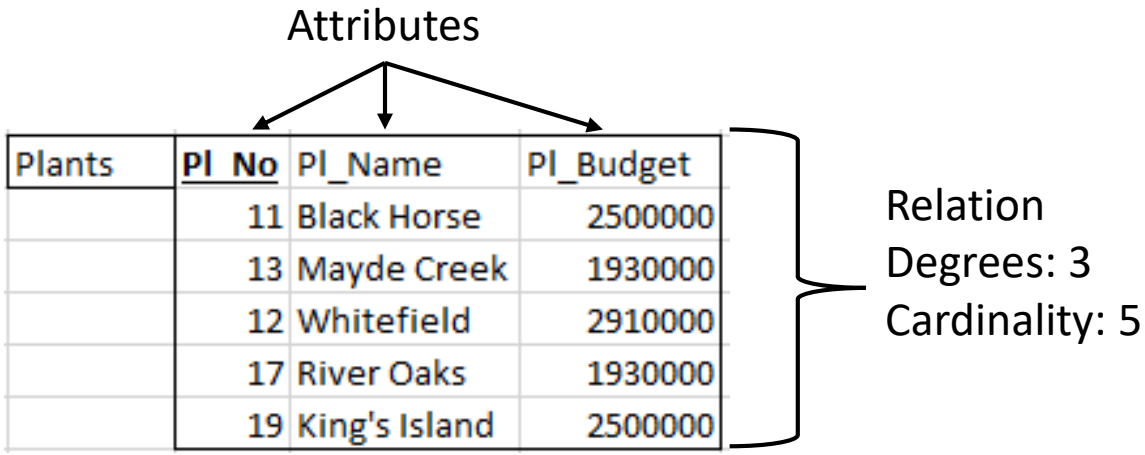


Plants	Pl_No	Pl_Name	Pl_Budget
	11	Black Horse	2500000
	12	Whitefield	2910000
	17	River Oaks	1930000
	19	King's Island	2500000

Relation
Degrees: 3
Cardinality: 4

Degree of the relation: number of attributes

- Degree is the number of attributes (columns) in a relation
  - But why?

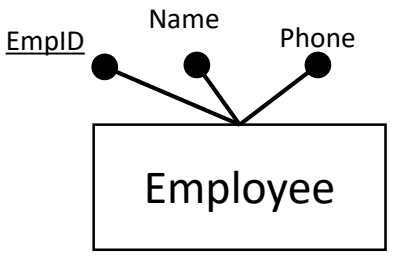


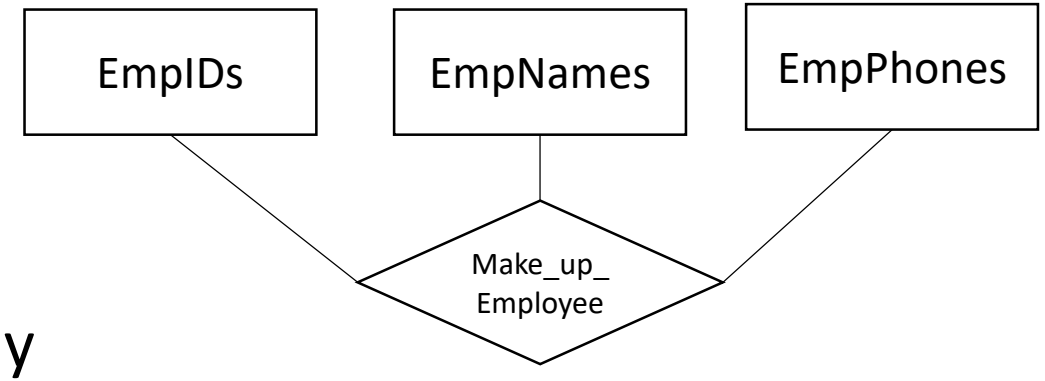
Degree of the relation: number of attributes

- Degree is the number of attributes (columns) in a relation
  - But why?

- Recall that an entity is a collection of related attributes

- Imagine this: each attribute of an entity can be considered an entity that is associated with the other attributes…

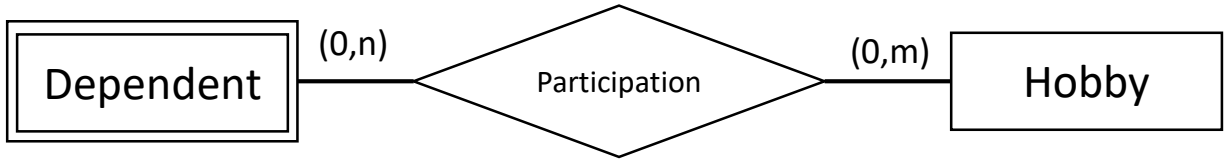




- We would never model it this way
  - …but this is how degree describes the number of attributes

Degree of the relation: number of attributes

- A working example - remember the relationship between hobbies and dependents?

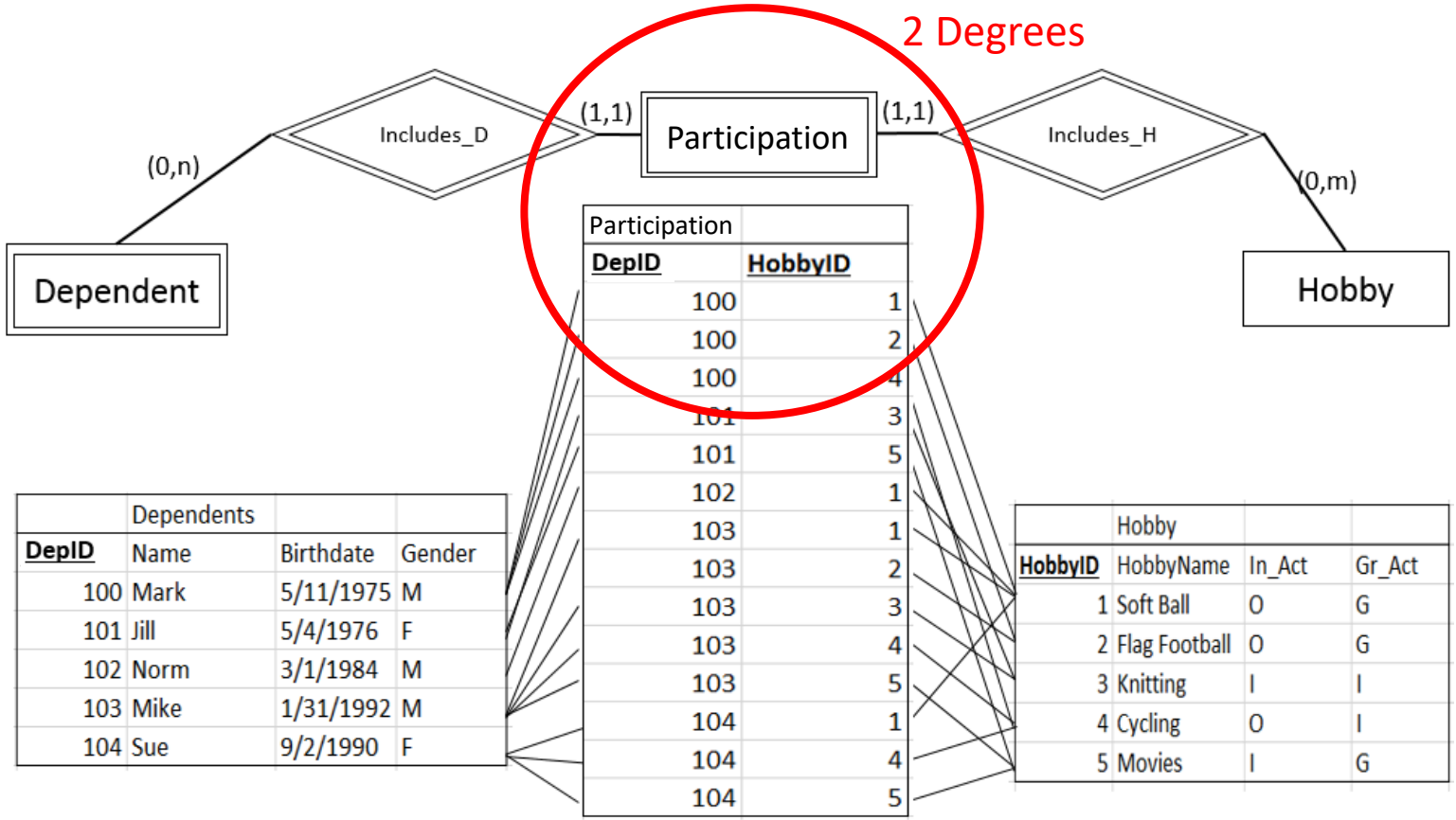


- How many degrees in the participation relationship?

- Since we cannot have m:n relationships in our logical model we decomposed this…

Degree of the relation: number of attributes

- When we decompose the m:n relationship, how many degrees are in the participants relation?



Participation

DepID	HobbyID
100	1
100	2
100	4
101	3
101	5
102	1
103	1
103	2
103	3
103	4
103	5
104	1
104	4
104	5

Dependents

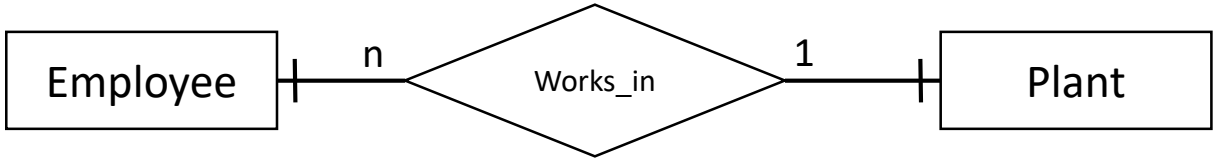
DepID	Name	Birthdate	Gender
100	Mark	5/11/1975	M
101	Jill	5/4/1976	F
102	Norm	3/1/1984	M
103	Mike	1/31/1992	M
104	Sue	9/2/1990	F

Hobby

HobbyID	HobbyName	In_Act	Gr_Act
1	Soft Ball	O	G
2	Flag Football	O	G
3	Knitting	I	I
4	Cycling	O	I
5	Movies	I	G

Degree of the relation: number of attributes

- Note: while **it is not necessary to do so** (and should generally not be done unless there are attributes of the relationship), 1:n and 1:1 relationships can also be decomposed using a gerund.



Employees

EmpID_N	Fname	Minit	Lname	NameTag	PL_No
12345	Adam	B	Christie		10
22233	Danny	E	Francisco		10
32112	Greg		Hernandez	1	10
43210	Ivana	J	Klink		11
52525	Greg		Hernandez	2	11

Plants

PL_No	PL_Name	Budget
10	Underwood	3000000
11	Garnett	3000000
12	Belmont	3500000
13	Vanderbilt	3500000

2 Degrees

Employees

EmpID_N	Fname	Minit	Lname	NameTag
12345	Adam	B	Christie	
22233	Danny	E	Francisco	
32112	Greg		Hernandez	1
43210	Ivana	J	Klink	
52525	Greg		Hernandez	2

Works_in

EmpID_N	PL_No
12345	10
22233	10
32112	10
43210	11
52525	11

Plants

PL_No	PL_Name	Budget
10	Underwood	3000000
11	Garnett	3000000
12	Belmont	3500000
13	Vanderbilt	3500000

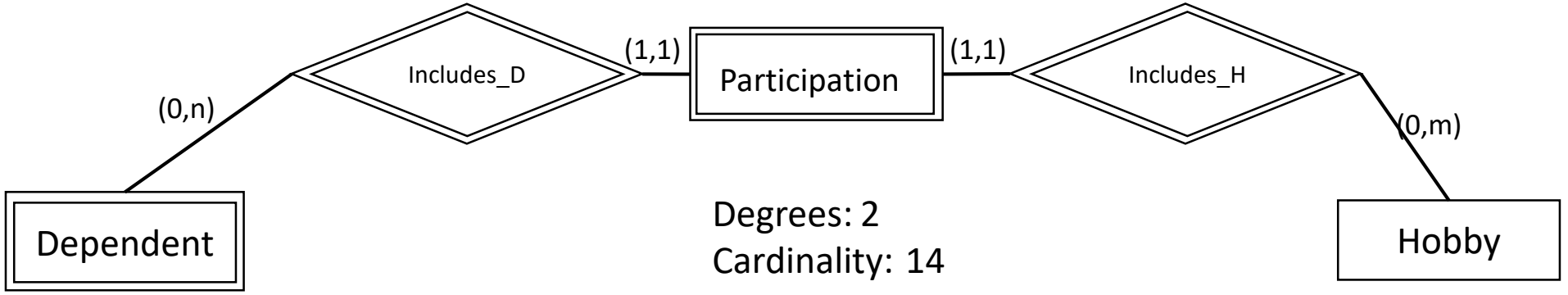
Cardinality of the relation: number of tuples

- Cardinality is the number of tuples (rows) in a relation

- When we say things like:
  - “many students may take a class”

- We really mean
  - “Any number of students up to number of students we have enrolled at the university may take a class”
  - …which is the cardinality of the students relation
    (Obviously other business rules may limit the number of student that may take a class – like the size of the classroom!)

Cardinality of the relation: number of tuples



Degrees: 2
Cardinality: 14

Degrees: 4
Cardinality: 5

	Dependents		
DepID	Name	Birthdate	Gender
100	Mark	5/11/1975	M
101	Jill	5/4/1976	F
102	Norm	3/1/1984	M
103	Mike	1/31/1992	M
104	Sue	9/2/1990	F

Participation	
DepID	**HobbyID**
100	1
100	2
100	4
101	3
101	5
102	1
103	1
103	2
103	3
103	4
103	5
104	1
104	4
104	5

Degrees: 4
Cardinality: 5

	Hobby		
HobbyID	HobbyName	In_Act	Gr_Act
1	Soft Ball	O	G
2	Flag Football	O	G
3	Knitting	I	I
4	Cycling	O	I
5	Movies	I	G

Cardinality of the relation: number of tuples

- Knowing the cardinality of a relation is important for understanding how must data is present…however, even more important is that it allows us to understand how large/complex join operations will be
- What is the <u>maximum</u> cardinality of the participants relation?

5 * 5 = Max cardinality of 25

Degrees: 4
Cardinality: 5

	Dependents		
DepID	Name	Birthdate	Gender
100	Mark	5/11/1975	M
101	Jill	5/4/1976	F
102	Norm	3/1/1984	M
103	Mike	1/31/1992	M
104	Sue	9/2/1990	F

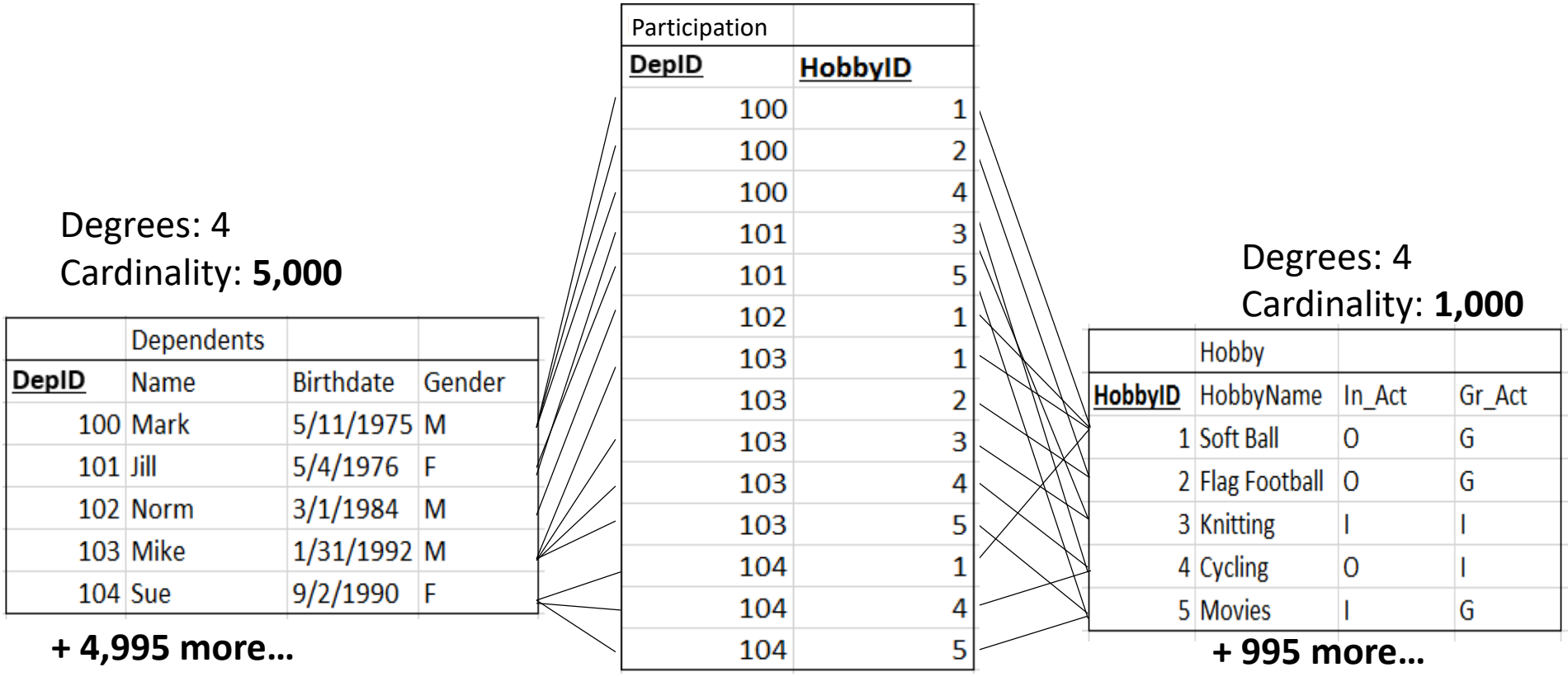
Participation	
DepID	**HobbyID**
100	1
100	2
100	4
101	3
101	5
102	1
103	1
103	2
103	3
103	4
103	5
104	1
104	4
104	5

Degrees: 4
Cardinality: 5

	Hobby		
HobbyID	HobbyName	In_Act	Gr_Act
1	Soft Ball	O	G
2	Flag Football	O	G
3	Knitting	I	I
4	Cycling	O	I
5	Movies	I	G

Cardinality of the relation: number of tuples

- What if there were 5,000 dependents and 1,000 hobbies?
  - Max cardinality of 5,000,000
- …and each hobby met once per month (12 times)?
  - Maximum cardinality of 60,000,000 (5,000 x 1,000 x 12)!
- We can reduce cardinality prior to joining in order to reduce computational complexity!



Module 6.1
The Relational Data Model

- Why do we care about logical data modeling?

- Be able to explain terms as used in Relational Data Models
  - Relation
  - Tuple
  - Cardinality
  - Degrees

Module 6.2
Characteristics of a Relation

- Be able to describe characteristics of a relation

Characteristics of a Relation

- A relation is a mathematical term that resembles a two-dimensional table
- Has a heading, which is a tuple of attributes, also known as the relation schema
- Has a body, which is made up of many tuples of data containing the same attributes
- Attributes of relation schema have unique names
- Values of an attribute in a relation come from same domain

Characteristics of a Relation (continued)

- Order of arrangement of tuples does not matter
- Order of attributes does not matter
- Each attribute value in tuple is atomic; hence, composite and multi-valued attributes are not allowed in a relation
- Derived attributes are not captured in relation schema
- All tuples in relation must be distinct (that is to say every relation schema must have unique identifier)
  - **If every tuple is distinct, then at a minimum the combination of the values of every attribute would be a unique identifier, right? We'll come back to this idea soon.**

Module 6.2
Characteristics of a Relation

- Be able to describe characteristics of a relation
- Describe three set operators

Module 6.3
Data Integrity Constraints

- Be able to define: Super keys, candidate keys, key attributes, non-key attributes, primary keys, and alternate keys
- What is the entity integrity constraint?
- What is the referential integrity constraint?
- What is the foreign key constraint?

A “Key” topic in your Database education as we talk about

- Super Keys
- Candidate Keys
- Primary Keys
- Alternate Keys
- Key Attributes
- Non-Key Attributes

Not to mention those we’ve already talked about:

- Partial Keys
- Foreign Keys

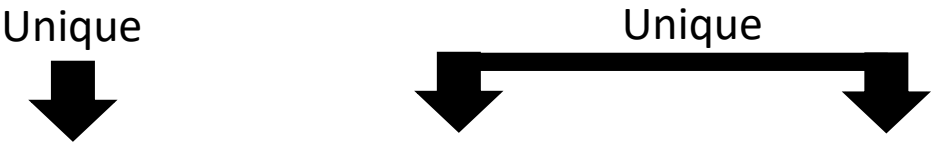
Data Integrity Constraints

- Rules that govern the behavior of data in a database
- Are technical expressions of business rules that emerge from user requirement specifications for database application
- Prevail across all tiers of data modeling – conceptual, logical, and physical
- Are considered to be part of the schema
  - Declared along with structural design of data model and hold for all valid states of a database and at all levels (conceptual, external, and internal)

The Concept of Unique Identifiers

- Superkey:
  - A set of one or more attributes, which taken collectively, uniquely identifies a tuple of a relation {uniqueness property}
- Candidate Key:
  - A superkey with no proper subset that uniquely identifies a tuple of a relation {uniqueness property + irreducibility}
- Primary Key:
  - A candidate key with no missing values for the constituent attributes {uniqueness property + irreducibility + entity integrity constraint}
- Alternate Key:
  - Any candidate key that is not serving the role of the primary key

A Sample Relation Instance: Prescription A



Rx_rx#	Rx_pat#	Rx_medcode	Rx-dosage
A100	7642	PCN	3
A103	4678	TYL	2
A102	4772	CLR	2
A101	6742	ASP	2
A104	4772	ZAN	3
A105	7456	CLR	2
A107	2222	TYL	2
A106	4772	VAL	2
A108	7384	CLR	3
A109	7384	ZAN	2
A110	7642	VAL	2

Prescription A

Superkey: one or more attributes which together can uniquely identify a tuple

- Rx_rx#

- (Rx_rx#, Rx_pat#)
- (Rx_rx#, Rx_medcode)
- (Rx_rx#, Rx_dosage)

- (Rx_pat#, Rx_medcode)
- (Rx_pat#, Rx_medcode, Rx_dosage)
- (Rx_pat#, Rx_medcode, Rx_rx#)

- (Rx_rx#, Rx_pat#, Rx_dosage)
- (Rx_rx#, Rx_medcode, Rx_dosage)
- (Rx_rx#, Rx_pat#, Rx_medcode, Rx_dosage)

Unique

Unique

Rx_rx#	Rx_pat#	Rx_medcode	Rx-dosage
A100	7642	PCN	3
A103	4678	TYL	2
A102	4772	CLR	2
A101	6742	ASP	2
A104	4772	ZAN	3
A105	7456	CLR	2
A107	2222	TYL	2
A106	4772	VAL	2
A108	7384	CLR	3
A109	7384	ZAN	2
A110	7642	VAL	2

Prescription A

Superkey: one or more attributes which together can uniquely identify a tuple

- In general if K is a superkey, then any superset of K (i.e., any set that contains K) is also a superkey

- Superkeys uniquely define a tuple, but there are lots of them – so let's do better

Unique

Unique

Rx_rx#	Rx_pat#	Rx_medcode	Rx-dosage
A100	7642	PCN	3
A103	4678	TYL	2
A102	4772	CLR	2
A101	6742	ASP	2
A104	4772	ZAN	3
A105	7456	CLR	2
A107	2222	TYL	2
A106	4772	VAL	2
A108	7384	CLR	3
A109	7384	ZAN	2
A110	7642	VAL	2

Prescription A

Candidate key (uniqueness + irreducibility)

- A superkey with no proper subset that uniquely identifies a tuple

- Rx_rx# ← Yes, Candidate key

- (Rx_rx#, Rx_pat#) ← No, the proper subset Rx_rx# uniquely identifies a tuple
- (Rx_rx#, Rx_medcode) ← No, the proper subset Rx_rx# uniquely identifies a tuple
- (Rx_rx#, Rx_dosage) ← No, the proper subset Rx_rx# uniquely identifies a tuple

- (Rx_pat#, Rx_medcode) ← Yes, Candidate key
- (Rx_pat#, Rx_medcode, Rx_dosage) ← No, why?
- (Rx_pat#, Rx_medcode, Rx_rx#) ← No, why?

- (Rx_rx#, Rx_pat#, Rx_dosage) ← No, why?
- (Rx_rx#, Rx_medcode, Rx_dosage) ← No, why?
- (Rx_rx#, Rx_pat#, Rx_medcode, Rx_dosage) ← No, why?

Key/Non-Key Attributes

- An attribute is always a key attribute, non-key attribute, or candidate key

- Key attribute:
  - Any attribute that is a proper subset of a candidate key
    - Note: A set is a subset of itself, but is not a **proper** subset

- Non-key attribute:
  - Any attribute that is not a subset of a candidate key

- Example:
  - Rx_rx# is not a key attribute of PRESCRIPTION-A since it is not a proper subset of a candidate key.
  - Rx_rx# is not a non-key attribute of PRESCRIPTION-A since it is a subset of a candidate key.
  - Rx_rx# is a candidate key of PRESCRIPTION-A since it is an irreducible superkey of PRESCRIPTION-A.

Key, Non-Key, or candidate key attribute?

- We identified two candidate keys for the prescription-A relation
  - Rx_rx# — Candidate Key
  - (Rx_pat#, Rx_medcode) — Candidate Key

- There are four atomic attributes in the prescription-A relation.
  - Rx_rx# — Candidate Key
  - Rx_pat# — Key Attribute
  - Rx_medcode — Key Attribute
  - Rx_dosage — Non-key Attribute

- An attribute is one and only one of the 3

Unique Unique

Rx_rx#	Rx_pat#	Rx_medcode	Rx-dosage
A100	7642	PCN	3
A103	4678	TYL	2
A102	4772	CLR	2
A101	6742	ASP	2
A104	4772	ZAN	3
A105	7456	CLR	2
A107	2222	TYL	2
A106	4772	VAL	2
A108	7384	CLR	3
A109	7384	ZAN	2
A110	7642	VAL	2

- Candidate keys themselves are not key attributes
  - A set is not a proper subset of itself

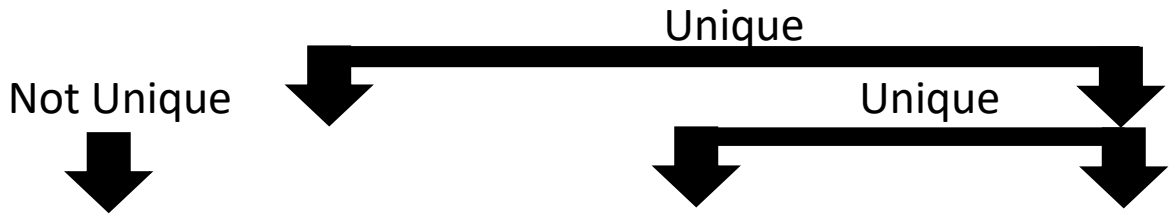
Primary Key

- A primary Key is a candidate key with one additional constraint: It must not be missing (NULL)
  - Entity integrity constraint
- Which to choose?
  - Technically does not matter, but choose the easy one (and one that will never be NULL)
  - Candidate keys not chosen as PK are alternate keys

Unique Unique

Rx_rx#	Rx_pat#	Rx_medcode	Rx-dosage
A100	7642	PCN	3
A103	4678	TYL	2
A102	4772	CLR	2
A101	6742	ASP	2
A104	4772	ZAN	3
A105	7456	CLR	2
A107	2222	TYL	2
A106	4772	VAL	2
A108	7384	CLR	3
A109	7384	ZAN	2
A110	7642	VAL	2

Another example – Prescription B



Rx_rx#	Rx_pat#	Rx_medcode	Rx_dosage
B100	7642	PCN	3
B103	4678	TYL	2
B102	4772	CLR	2
B101	6742	ASP	2
B102	4772	ZAN	**2**
B105	7456	CLR	2
B107	2222	TYL	2
B106	4772	VAL	2
B108	7384	CLR	3
B109	7384	ZAN	2
B100	7642	VAL	2

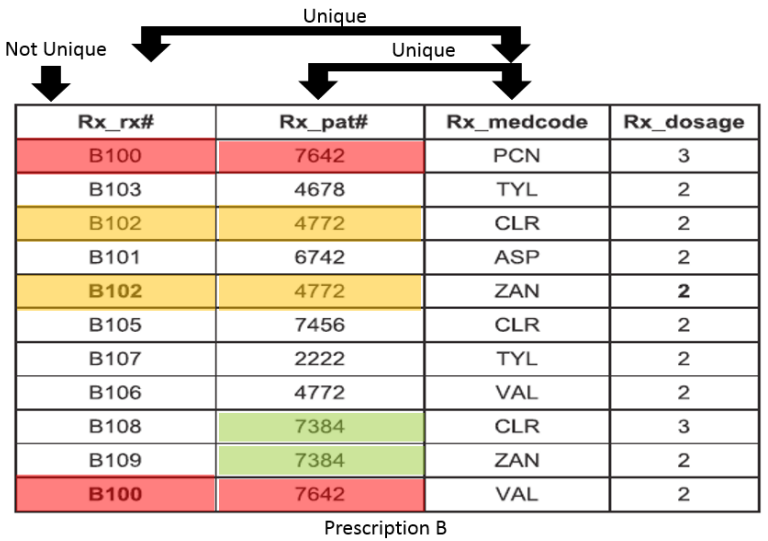
Prescription B

What are the superkeys?

- (Rx_pat#, Rx_medcode)
- (Rx_pat#, Rx_medcode, Rx_rx#)
- (Rx_pat#, Rx_medcode, Rx_dosage#)

- (Rx_rx#, Rx_medcode)
- (Rx_rx#, Rx_medcode, Rx_dosage#)

- (Rx_rx#, Rx_medcode, Rx_pat#, Rx_dosage#)



Rx_rx#	Rx_pat#	Rx_medcode	Rx_dosage
B100	7642	PCN	3
B103	4678	TYL	2
B102	4772	CLR	2
B101	6742	ASP	2
B102	4772	ZAN	**2**
B105	7456	CLR	2
B107	2222	TYL	2
B106	4772	VAL	2
B108	7384	CLR	3
B109	7384	ZAN	2
B100	7642	VAL	2

Prescription B

What are the candidate keys?

- (Rx_pat#, Rx_medcode) ☺
- (Rx_pat#, Rx_medcode, Rx_rx#) ✖
- (Rx_pat#, Rx_medcode, Rx_dosage#) ✖

- (Rx_rx#, Rx_medcode) ☺
- (Rx_rx#, Rx_medcode, Rx_dosage#) ✖

- (Rx_rx#, Rx_medcode, Rx_pat#, Rx_dosage#) ✖

Not Unique

Unique

Unique

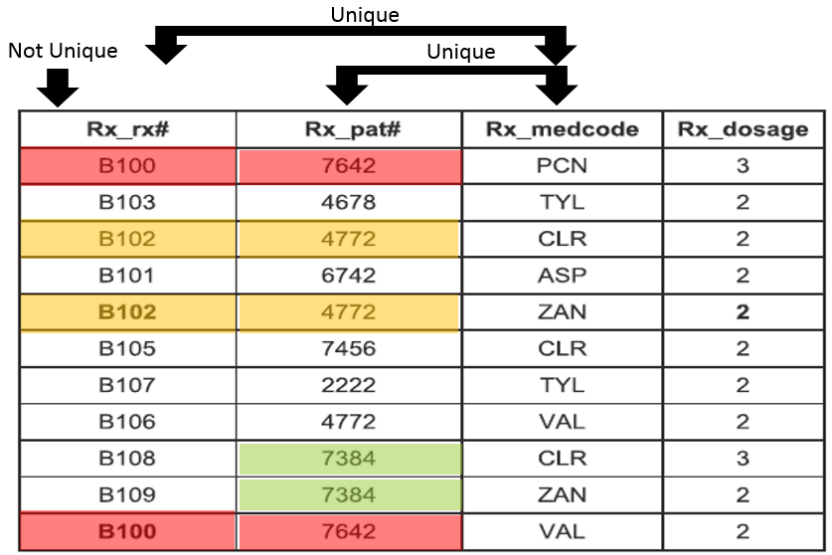
Rx_rx#	Rx_pat#	Rx_medcode	Rx_dosage
B100	7642	PCN	3
B103	4678	TYL	2
B102	4772	CLR	2
B101	6742	ASP	2
B102	4772	ZAN	**2**
B105	7456	CLR	2
B107	2222	TYL	2
B106	4772	VAL	2
B108	7384	CLR	3
B109	7384	ZAN	2
B100	7642	VAL	2

Prescription B

What attributes are keys vs. non-keys?

- We identified two candidate keys for the prescription-B relation
  - (Rx_pat#, Rx_medcode) — Candidate Key
  - (Rx_rx#, Rx_medcode) — Candidate Key

- There are four atomic attributes in the prescription-B relation
  - Rx_rx# — Key Attribute
  - Rx_pat# — Key Attribute
  - Rx_medcode — Key Attribute
  - Rx_dosage — Non-key Attribute



Rx_rx#	Rx_pat#	Rx_medcode	Rx_dosage
B100	7642	PCN	3
B103	4678	TYL	2
B102	4772	CLR	2
B101	6742	ASP	2
B102	4772	ZAN	**2**
B105	7456	CLR	2
B107	2222	TYL	2
B106	4772	VAL	2
B108	7384	CLR	3
B109	7384	ZAN	2
B100	7642	VAL	2

Prescription B

Superkey and Candidate Key (page 287)

PRESCRIPTION (Rx_rx#, Rx_pat#, Rx_medcode, Rx_dosage)

	PRESCRIPTION-A		PRESCRIPTION-B	
	Superkey	Candidate Key	Superkey	Candidate Key
Rx_rx#	Yes	Yes	No	No
Rx_pat#	No	No	No	No
Rx_medcode	No	No	No	No
Rx_dosage	No	No	No	No
Rx_rx#, Rx_pat#	Yes	No	No	No
Rx_rx#, Rx_medcode	Yes	No	Yes	Yes
Rx_rx#, Rx_dosage	Yes	No	No	No
Rx_pat#,Rx_medcode	Yes	Yes	Yes	Yes
Rx_pat#, Rx_dosage	No	No	No	No
Rx_medcode, Rx_dosage	No	No	No	No
Rx_rx#, Rx_pat#, Rx_medcode	Yes	No	Yes	No
Rx_rx#, Rx_pat#, Rx_dosage	Yes	No	No	No
Rx_rx#, Rx_medcode, Rx_dosage	Yes	No	Yes	No
Rx_pat#, Rx_medcode, Rx_dosage	Yes	No	Yes	No
Rx_rx#, Rx_pat#, Rx_medcode, Rx_dosage	Yes	No	Yes	No

Table 6.2 Superkeys and candidate keys in the PRESCRIPTION-A and PRESCRIPTION-B relations

Referential Integrity Constraint

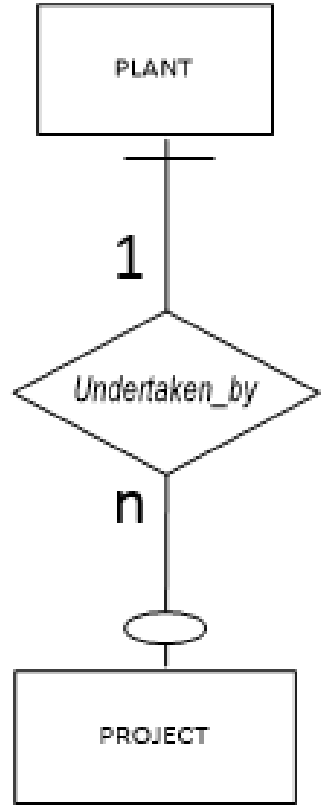
- Key constraints (superkey and candidate key) and entity integrity constraint (primary key) pertain to individual relation schemas
- Referential integrity constraints are specified between two relation schemas (i.e., R1 and R2)
- Specifically, a referential integrity constraint is specified between two relations in order to maintain consistency across tuples of the two relations
- Informal definition: A tuple in one relation that refers to another relation must refer to an existing tuple in that relation.

Foreign Key Constraint

- A special form of referential integrity constraint specification
- Establishes an explicit association between two relation schemas and maintains the integrity of such an association
- Foreign key: An attribute(s) set, A2, in a relation schema R2 that shares the same domain with a **candidate key** (A1) of another relation schema R1; A2 is said to reference or refer to the relation schema R1.
  - Note: R2 is known as the referencing relation and R1 is called the referenced relation. The attribute(s) doing the referencing (A2 in R2) is the foreign key, while the candidate key (**advisably, the primary key**) being referenced (A1 in R1) is the referenced attribute(s).
- Referred to as Inclusion Dependency, this constraint is algebraically expressed as: **R2.{A2} ⊆ R1.{A1}**
- Meaning: Child.foreignkey (R2.{A2}) is inclusion dependent on parent.primarykey (R1.{A1})

Example – Source Schema

- Bearcat Incorporated is a manufacturing company that has several plants in the northeastern part of the United States. These plants are responsible for leading different projects that the company might undertake, depending on a plants' function. A certain plant might even be associated with several projects but a project is always under the control of just one plant. Some plants do not undertake any projects at all.

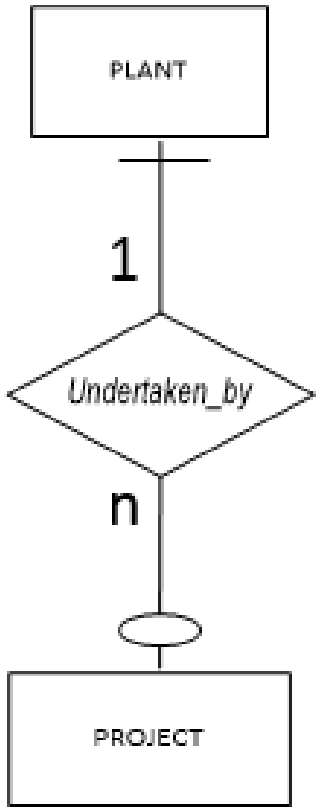


FK in the 1:n relationship

PLANT	Pl_name	Pl_p#	Pl_budget
	Black Horse	11	1230000
	Mayde Creek	13	1930000
	Whitefield	12	2910000
	River Oaks	17	1930000
	King's Island	19	2500000
	Ashton	15	2500000

Foreign Key

PROJECT	Prj_name	Prj_n#	*Prj_pl_p#*
	Solar Heating	41	11
	Lunar Cooling	17	17
	Synthetic Fuel	29	17
	Nitro-Cooling	23	12
	Robot Sweeping	31	11
	Robot Painting	37	19
	Ozone Control	13	19



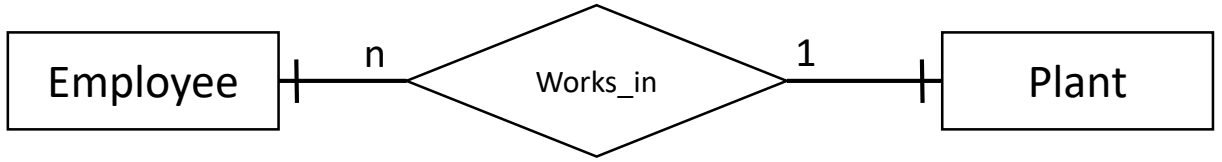
R2.{A2} ⊆ R1.{A1}

Project.Prj_pl_P# ⊆ Plant.Pl_p#

Note: PROJECT.**Prj_pl_p#** is the foreign key referencing PLANT.**Pl_p#**, the primary key of PLANT.

Note: Prj_n# would be a better primary key for the Project entity. We'll talk about why later, but this works for this example

Foreign key placement



- What is the cardinality of the relationship? 1:n
- Which entity is the parent? Plant
- In a 1:M relationship, Foreign key (FK) goes with the "child" side
  - The FK is an attribute in the child entity (employees in this case) that refers to the primary key of the entity on the other side of the relationship

Employees

EmpID_A	**EmpID_N**	Fname	Minit	Lname	NameTag	PL_No
E	12345	Adam	B	Christie		10
E	22233	Danny	E	Francisco		10
C	32112	Greg		Hernandez	1	10
E	43210	Ivana	J	Klink		11
E	52525	Greg		Hernandez	2	11

Plants

PL_No	PL_Name	Budget
10	Underwood	3000000
11	Garnett	3000000
12	Belmont	3500000
13	Vanderbilt	3500000

- Would this even make sense the other way?

Note this is dataset is truncated. In reality every entity in the plant table must have at least 100 employees (as specified in the business rule – the ERD above merely specifies that it must participate)

Note: Employees.PL_No is a FK that refers to Plants.PL_No. All values of Employees.PL_No must be found in the domain of values for Plants.PL_No (i.e., must be a valid plant)

In 1:m The FK must go with the child entity

- If we put the FK with the parent it is INCORRECT:
  - Lots of data redundancy in the Plants table if the FK is there

Employees

EmpID_A	EmpID_N	Fname	Minit	Lname	NameTag
E	12345	Adam	B	Christie	
E	22233	Danny	E	Francisco	
C	32112	Greg		Hernande	1
E	43210	Ivana	J	Klink	
E	52525	Greg		Hernande	2

Plants

PL_No	PL_Name	Budget	EmpID
10	Underwood	3000000	E12345
10	Underwood	3000000	E22233
10	Underwood	3000000	C32112
11	Garnett	3000000	E43210
11	Garnett	3000000	E52525
12	Belmont	3500000	**NULL**
13	Vanderbilt	3500000	**NULL**

Note: In this case, Plants.EmpID is a FK that refers to the composite attribute Employees.EmpID_A + Employees.EmpID_N.

There is now redundancy for the Plants.PL_Name and Plants.Budget attributes, **and** PL_No is no longer unique!

- No redundancy when FK is with Employees (CORRECT):

Employees

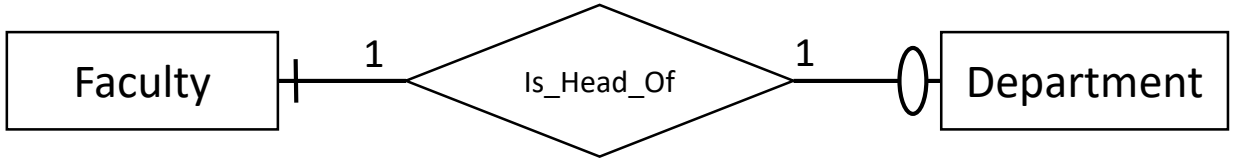
EmpID_A	EmpID_N	Fname	Minit	Lname	NameTag	PL_No
E	12345	Adam	B	Christie		10
E	22233	Danny	E	Francisco		10
C	32112	Greg		Hernandez	1	10
E	43210	Ivana	J	Klink		11
E	52525	Greg		Hernandez	2	11

Plants

PL_No	PL_Name	Budget
10	Underwood	3000000
11	Garnett	3000000
12	Belmont	3500000
13	Vanderbilt	3500000

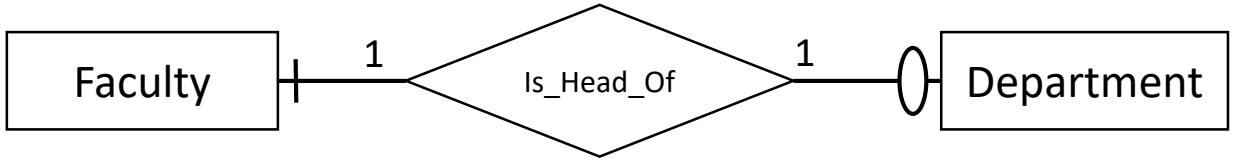
Where do we put the foreign key in a 1:1 relationship?

- There is no clear parent….



In 1:1 we place the FK based on participation

- FK goes with the entity that has total (mandatory) participation



Faculty		
EmpID	Name	Title
112	Chad Larson	Assistant Professor
108	Dusya Vera	Professor
109	Edward Blair	Professor
111	Gerald Lobo	Professor
106	Kris Jacobs	Professor
102	Mark Grimes	Assistant Professor
110	Melanie Rudd	Assistant Professor
103	Michael Parks	Associate Professor
101	Norm Johnson	Professor
104	Praveen Kumar	Professor
107	Steve Werner	Professor
105	Thomas George	Professor

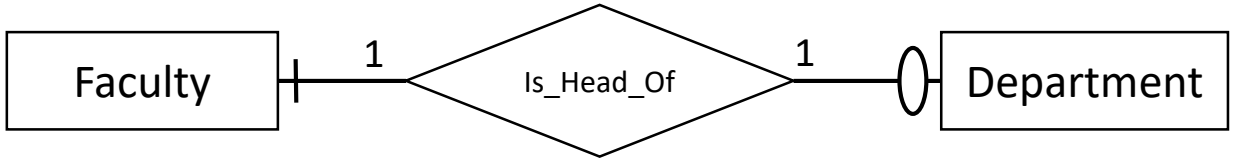
Departments		
DeptID	Deptname	Head
1	Accounting	111
2	DISC	101
3	Finanace	104
4	Management	107
5	Marketing	109

Note: In this case, Departments.Head is a FK that refers to Faculty.EmpID.

CORRECT WAY!

In 1:1 we place the FK based on participation

- Imagine if we did it the other way, it still works, but lots of NULL values!
  - NULL values are difficult to work with – avoid them as much as possible!



Faculty			
EmpID	Name	Title	Head of
112	Chad Larson	Assistant Professor	
108	Dusya Vera	Professor	
109	Edward Blair	Professor	5
111	Gerald Lobo	Professor	1
106	Kris Jacobs	Professor	
102	Mark Grimes	Assistant Professor	
110	Melanie Rudd	Assistant Professor	
103	Michael Parks	Associate Professor	
101	Norm Johnson	Professor	2
104	Praveen Kumar	Professor	3
107	Steve Werner	Professor	4
105	Thomas George	Professor	

Departments	
DeptID	Deptname
1	Accounting
2	DISC
3	Finanace
4	Management
5	Marketing

Note: In this case, Faculty.HeadOf is a FK that refers to Departments.DeptID.

No data redundancy, but lots of nasty NULL values

WRONG WAY!

In 1:1 we place the FK based on participation

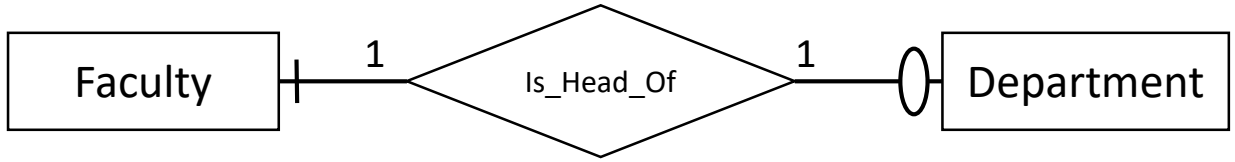
- There are some fringe cases…

- If both sides of a 1:1 have mandatory participation
  - It doesn't matter where you place the FK
  - This would be a pretty rare thing to find

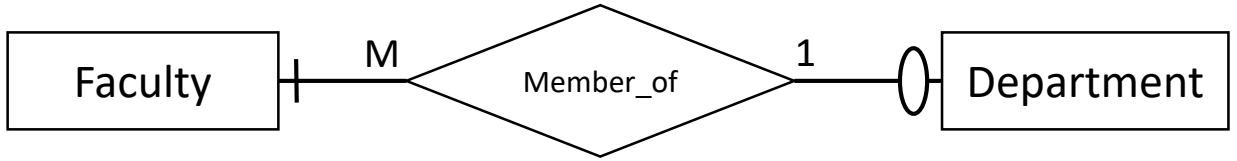
- If both sides of a 1:1 have optional participation
  - No hard and fast rule, but generally would be better to put it with the entity that will have fewer instances
  - Alternatively, you can use a gerund to ensure you will have no NULL values!

Watch out!

- The “is head of” relationships is different than the “member of” relationship:



- Where should the FK go for the “Member of” relationship? Faculty (The Child)



Note: In this case, Faculty.Department is a FK that refers to Departments.DeptID - this is the “Member_of relationship.

Departments.Head is a FK that refers to Faculty.EmpID - this is the “Is_Head_of” relationship.

We have no redundancy and no NULL values with this design

Faculty

EmpID	Name	Title	Department
112	Chad Larson	Assistant Professor	1
108	Dusya Vera	Professor	4
109	Edward Blair	Professor	5
111	Gerald Lobo	Professor	1
106	Kris Jacobs	Professor	3
102	Mark Grimes	Assistant Professor	2
110	Melanie Rudd	Assistant Professor	5
103	Michael Parks	Associate Professor	2
101	Norm Johnson	Professor	2
104	Praveen Kumar	Professor	3
107	Steve Werner	Professor	4
105	Thomas George	Professor	3

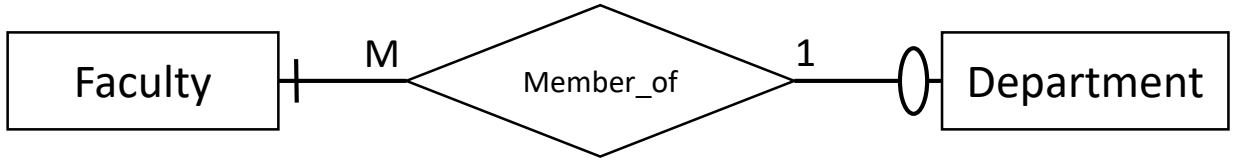
Departments

DeptID	Deptname	Head
1	Accounting	111
2	DISC	101
3	Finanace	104
4	Management	107
5	Marketing	109

CORRECT WAY!

Watch out!

- What if we did it the wrong way?



Faculty

EmpID	Name	Title
112	Chad Larson	Assistant Professor
108	Dusya Vera	Professor
109	Edward Blair	Professor
111	Gerald Lobo	Professor
106	Kris Jacobs	Professor
102	Mark Grimes	Assistant Professor
110	Melanie Rudd	Assistant Professor
103	Michael Parks	Associate Professor
101	Norm Johnson	Professor
104	Praveen Kumar	Professor
107	Steve Werner	Professor
105	Thomas George	Professor

Departments

DeptID	Deptname	Head	EmpID
1	Accounting	111	111
1	Accounting	111	112
2	DISC	101	101
2	DISC	101	102
2	DISC	101	103
3	Finanace	104	104
3	Finanace	104	105
3	Finanace	104	106
4	Management	107	107
4	Management	107	108
5	Marketing	109	109
5	Marketing	109	110

Redundant data for the Deptname and Head attributes **AND** DeptID is no longer unique!

WRONG WAY!

Module 6.3
Data Integrity Constraints

- Be able to define: Super keys, candidate keys, key attributes, non-key attributes, primary keys, and alternate keys
- What is the entity integrity constraint?
- What is the referential integrity constraint?
- What is the foreign key constraint?

Progress Quiz Time!

- The Progress Quiz is available in Canvas
  - You MUST complete the quiz on Canvas by 5:00 on Friday – This in-class activity does not count for points!
  - Each week we will discuss the questions, so for those of you that are in class and keeping up with things, you'll have an extra easy time with it!
- Go to http://kahoot.it and we'll get started momentarily!

Go forth and do great things

- Get to work on assignment 2
- We will continue relational data modeling next week
- Exam 1 is coming up in two weeks – March 4

BZAN 6354

Live Lecture 5

February 19, 2024

Dr. Mark Grimes, Ph.D.
gmgrimes@bauer.uh.edu

UNIVERSITY of HOUSTON
C. T. BAUER COLLEGE of BUSINESS
Department of Decision & Information Sciences