BZAN 6354

Lecture 3

February 5, 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

- Quick Review
- 2.4: Relationships
- 2.5: Cardinality and Participation
- 2.6: Deletion Constraints
- 10 minute break
- 10.2: Data Definition Language for multiple tables

Review: Data vs. Information

Data is raw/unformatted/unorganized

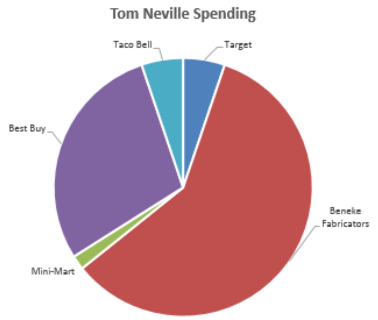
```
12012012,345844475,2295,2213,140223
12012012,345844475,1245,25100,115123
12012012,427658847,1154,885,57625
12052012,345844475,3011,754,114369
12062012,427658847,9584,10001,47624
12082012,427658847,2295,2523,45101
12122012,345844475,9584,12245,101217
12152012,345844475,1154,1300,99917
12192012,345844475,1154,907,113462
12192012,427658847,2224,1085,44016
12192012,427658847,1154,975,43041
12222012,427658847,2224,1085,41956
12231012,427658847,3030,122,41834
12262012,427658847,2295,1850,39984
12272012,427658847,1199,1925,38059
12272012,427658847,2224,1085,36974
12292012,427658847,9999,2000,34974
```

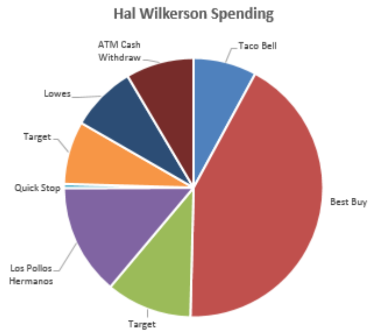
Data in context = Information

Date	Cust_ID	Vend_ID	Charge	Balance
12-01-2012	345-84-4475	2295	$22.13	$1,402.23
12-01-2012	345-84-4475	1245	$251.00	$1,151.23
12-01-2012	427-65-8847	1154	$8.85	$576.25
12-05-2012	345-84-4475	3011	$7.54	$1,143.69
12-06-2012	427-65-8847	9584	$100.01	$476.24
12-08-2012	427-65-8847	2295	$25.23	$451.01
12-12-2012	345-84-4475	9584	$122.45	$1,012.17
12-15-2012	345-84-4475	1154	$13.00	$999.17
12-19-2012	345-84-4475	1154	$9.07	$1,134.62
12-19-2012	427-65-8847	2224	$10.85	$440.16
12-19-2012	427-65-8847	1154	$9.75	$430.41
12-22-2012	427-65-8847	2224	$10.85	$419.56
12-23-1012	427-65-8847	3030	$1.22	$418.34
12-26-2012	427-65-8847	2295	$18.50	$399.84

Vend_ID	Vendor
1154	Taco Bell
1199	Lowes
1245	Beneke Fabricators
2224	Los Pollos Hermanos
2295	Target
3011	Mini-Mart
3030	Quick Stop
9584	Best Buy
9999	ATM Cash Withdraw

Cust_ID	Customer
345-84-4475	Tom Neville
427-65-8847	Hal Wilkerson





Knowledge = Information analyzed, visualized, etc. to help make decisions and predictions

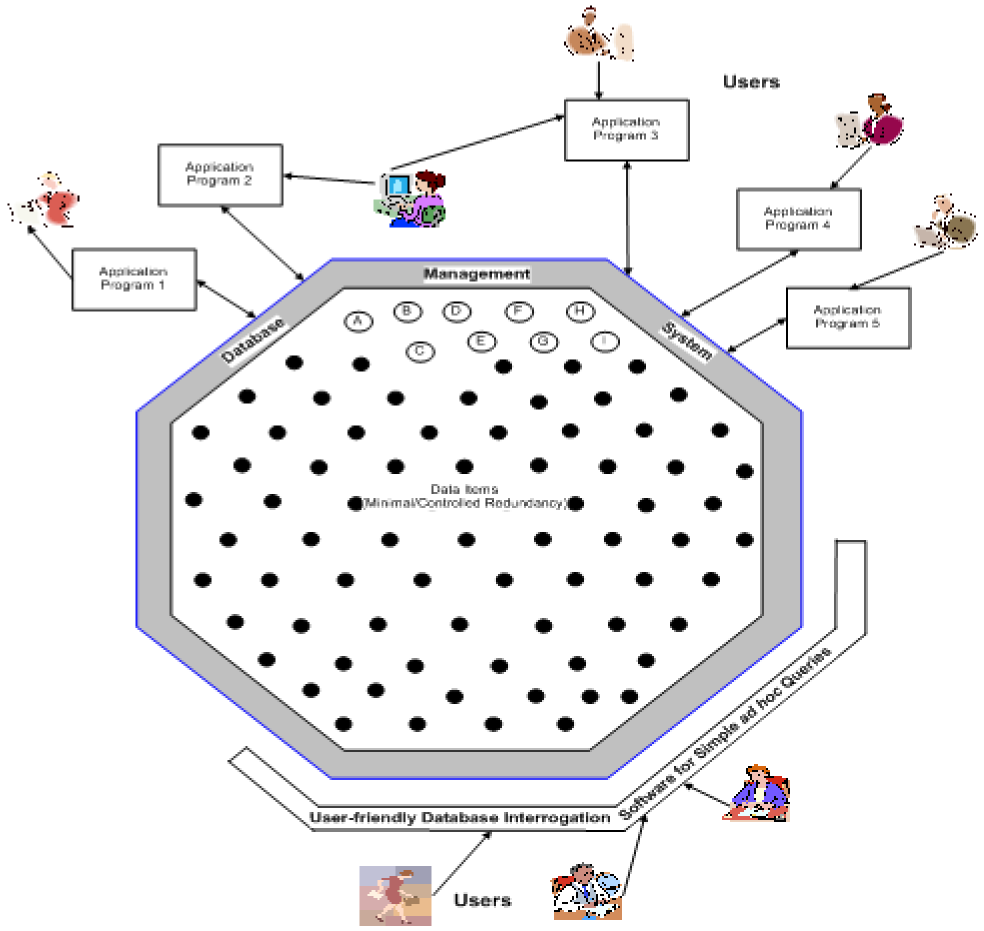
Review: What is a Database Management System

- A DBMS Facilitates data access in a database without burdening a user with the details of how the data is physically organized

ORACLE®

Microsoft® SQL Server®

MySQL®



Review: What is a model?

- Simplified expression of observed or unobservable reality used to perceive relationships in the outside world.

- All models are wrong, but some are useful
  - Box, George. E. P., and Draper, N. R., (1987), *Empirical Model Building and Response Surfaces*, John Wiley & Sons, New York, NY.

- If a model was perfectly correct, it would be the real thing!

Review: Data Models

Domains

Spots: {Yes, No}　　Weight: {800-2200}

Sex: {M,F} ?　　Color: ?

Entity Type (table)

Attributes

Entity Instances (records)

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

Values

Entity Class: Animal (these attributes could be used to describe any animal)

Relationships: Not shown here, but each horse is associated with an owner, a stable, etc.

Review: SQL DDL

- Remember: in the CREATE statement we provide a comma delimited list of table elements:
  - Column definition (i.e., attribute name)
  - Constraint definition

- Constraints can be placed on the same line as the definition of the attribute or on a line by themselves

```
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,
 CONSTRAINT chk_weight CHECK (weight >= 800 AND weight <=2200),
 CONSTRAINT chk_sex CHECK (sex IN ('M','F'))
);
```

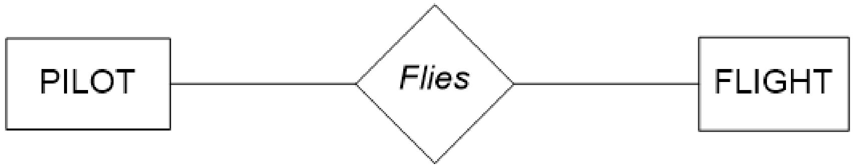
Module 2.4 (2.3.3 in the book)

Relationships

- Describe unary, binary, ternary, quaternary, and “n-ary” degrees of connectivity

Let’s start with binary (not unary)

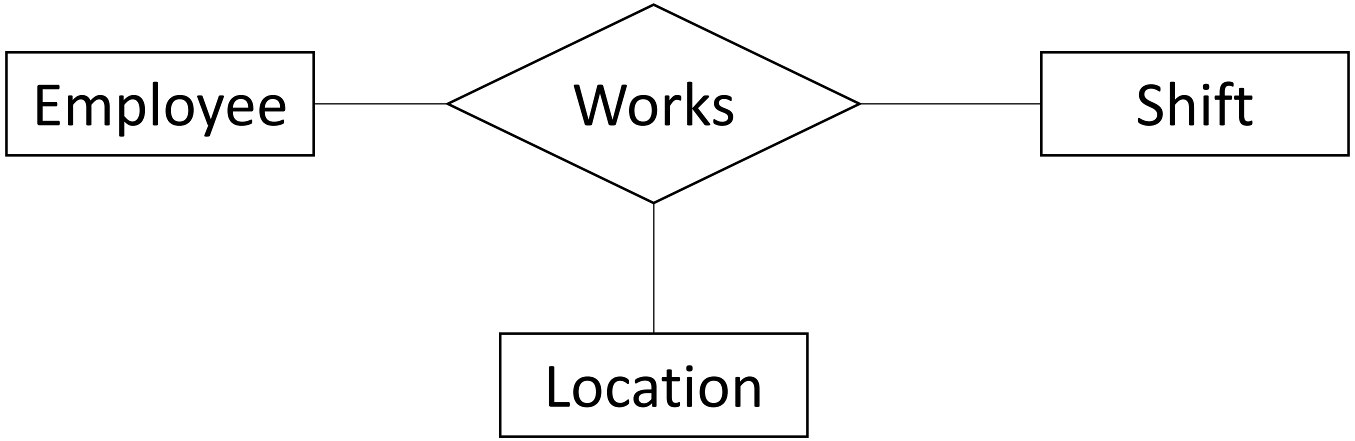
- Simplest relationship
- Degree n=2



- **Captain Smith** flies flight **DL3412**
- **Captain Johnson** flies flight **UA443**
- **Major Tom** flies flight **DB2016**

Ternary relationships

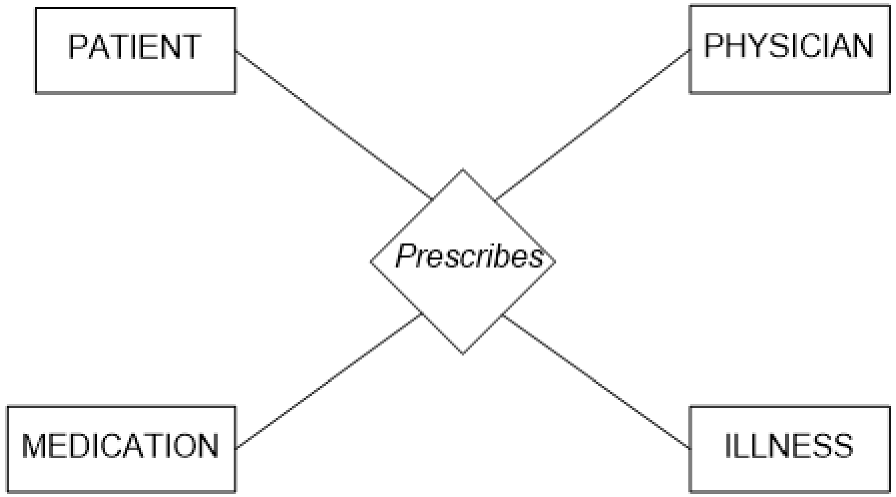
- Degree n=3



- **John** works **Morning** shift at the **Downtown** location
- **Kate** works **Morning** shift at the **Downtown** location
- **Tim** works **Afternoon** shift at the **Downtown** location
- **Ryan** works **Afternoon** shift at the **Midtown** location
- **Claudia** works **Evening** shift at the **Campus** location

Quaternary relationships

- Degree n=4



- **Dr. Fields** prescribes **Naproxin** to treat **Sharon Moore** for **muscle pain**.
- **Dr. Fields** prescribes **Ibuprofen** to treat **Michelle Li** for a **headache**.
- **Dr. Smith** prescribes **Naproxin** to treat **Barb Metzger** for **muscle pain**.
- **Dr. Sai** prescribes **acetaminophen** to treat **Michelle Li** for a **headache**.
- **Dr. Smith** prescribes **Ibuprofen** to treat **Jeff Camm** for **blood pressure**

N-ary relationship

- Unary (1), binary (2), ternary (3), quaternary (4)

- I suppose we could keep going…
  - Quintary (5), hexary (6), septary (7), octary (8)…
  - …but no one ever does

- We can generally refer to relationships of any size as n-ary
  - The term “n” is the number of degrees in the relationship
  - Typically anything above ternary/quaternary is referred to as “n-ary”

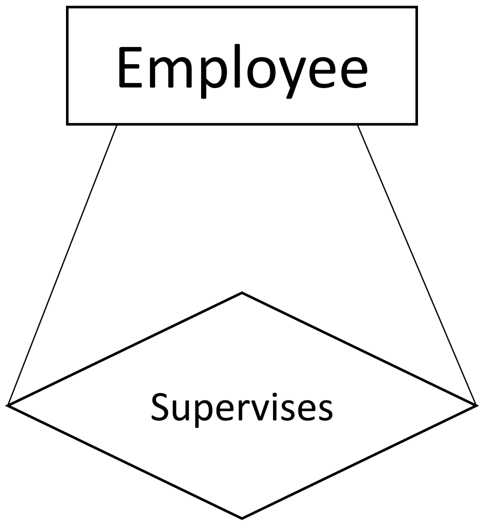
- In two weeks (module 6) we will expand on the meaning of this concept

But we haven’t talked about unary yet!

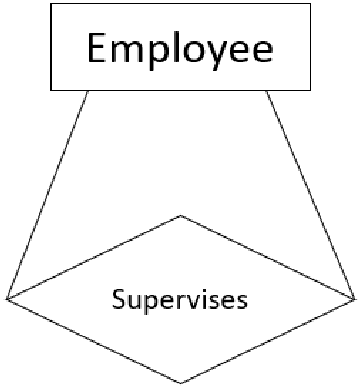
- What’s unary mean?
- It’s kind of a special type of relationship

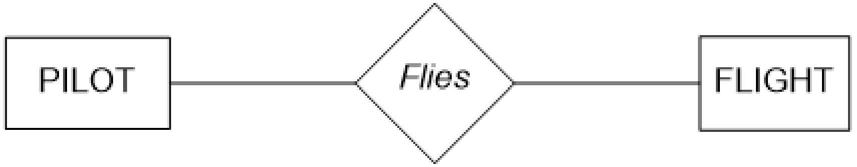
Unary relationships

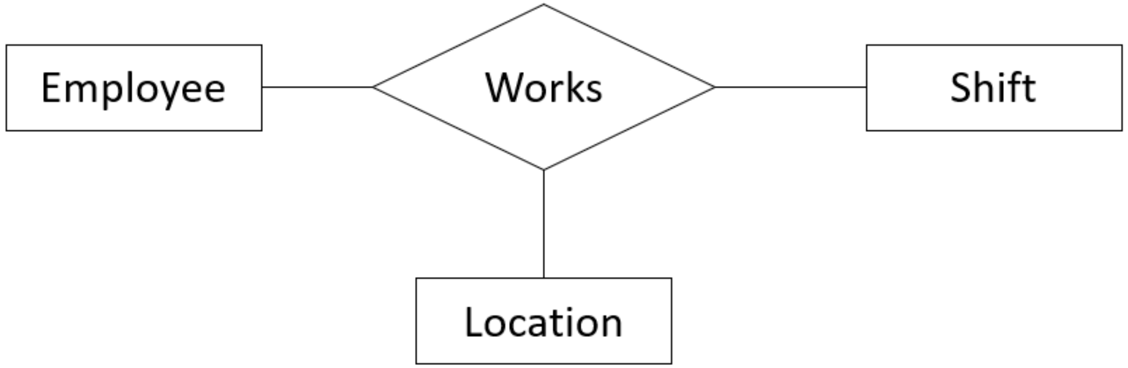
- Degree n=1
- Also known as a “recursive relationship”
- A “manager” is not a totally different entity…
  ...a manager is just an employee that supervises
  other employees
- In relationships with 2+ degrees, multiple entities interact
- With unary relationships, one entity interacts with itself

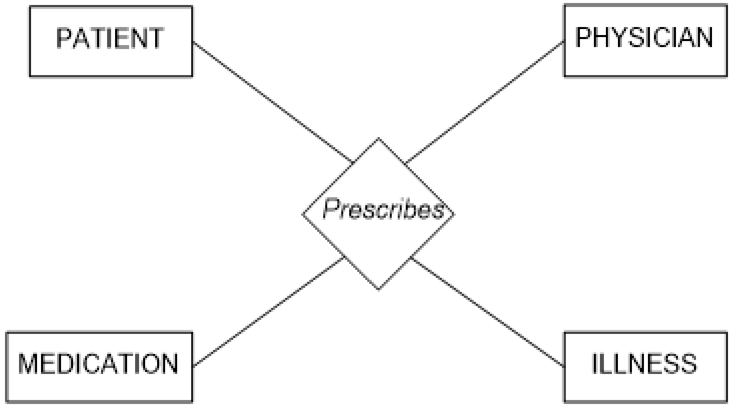


Unary, binary, ternary, and quaternary









Module 2.4 (2.3.3 in the book)
Relationships

- Describe unary, binary, ternary, quaternary, and “n-ary” degrees of connectivity

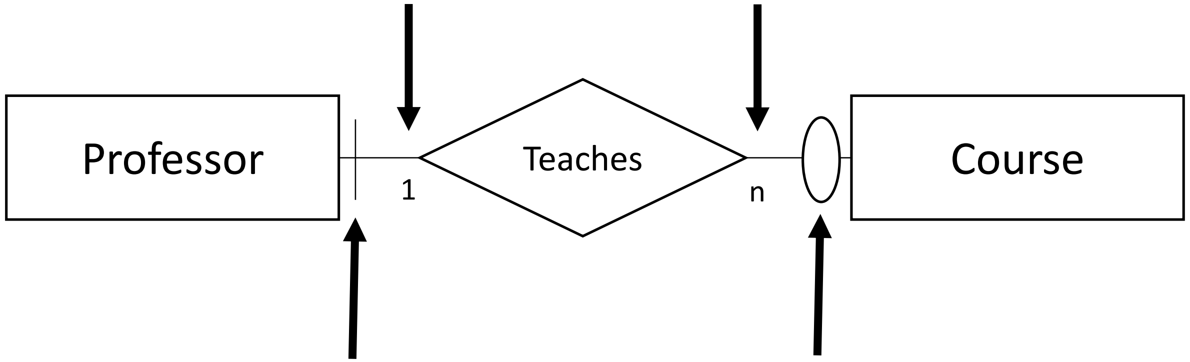
Module 2.5 (2.3.4 in the book)

Cardinality and Participation

- Describe 1:1, 1:n, n:1 and m:n cardinalities
- Describe the minimum cardinality, partial participation, and total participation constraints

Structural Constraints of a Relationship Type

- Two structural constraints define a relationship type:
  - **Cardinality**: Maximum number of instances of an entity type that relate to a single instance of an associated entity type through a relationship



  - **Participation**: Is it required that each instance of an entity participate in the relationship?

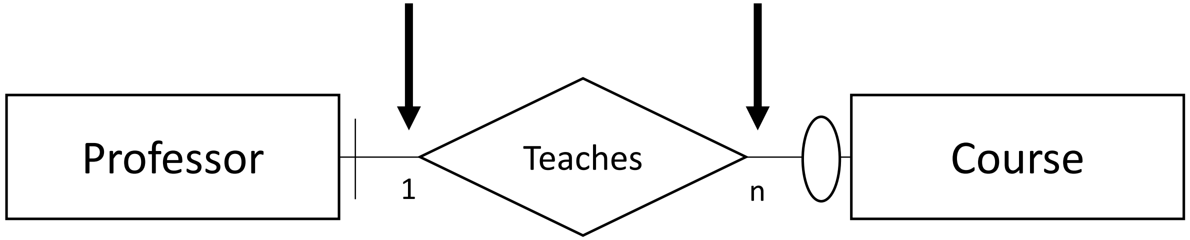
Cardinality

- m:n – An entity instance in set A is associated with up to “m” (many) entity instances in set B. An entity instance in set B is associated with up to “n” (many) entity instances in set A.

- 1:n – An entity instance in set A is associated with up to “n” (many) entity instances in set B. An entity instance in set B is related to no more than 1 entity instance in set A.

- n:1 – The reverse of 1:n (these are often combined).

- 1:1 - An entity instance in set A is associated with no more than 1 entity instance in set B. An entity instance in set B is related to no more than 1 entity instance in set A

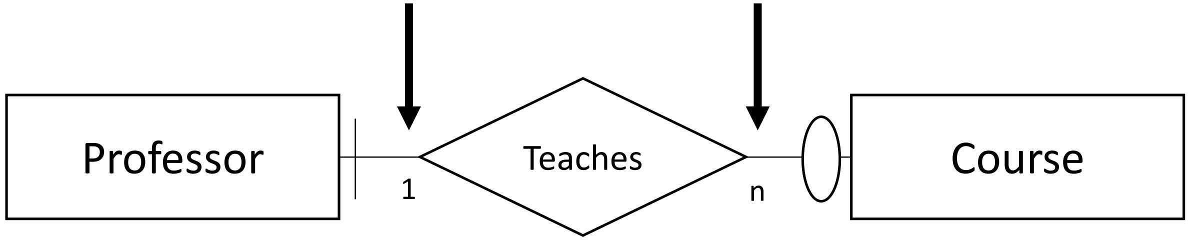


Cardinality – Parent-Child Relationships

- In 1:n or n:1 relationships, the entity that has a cardinality of “many” is considered the “parent”

- In a m:n relationship, either/both sides could be considered the “parent”, because there are two underlying 1:n and n:1 relationships
    - We’ll explore this idea more in the next set of lectures

- In a 1:1 relationship, neither side is clearly the parent based on cardinality



Cardinality – Parent-Child Relationships

- **One** coach (Parent) has **many** players
  - 1:n
  - **One** football coach coaches **many** players
  - Each player has one coach

- **Many** children have **one** mother (Parent)
  - n:1
  - **Many** children are raised by **one** mother
  - Each child has one mother

- **Many** shoppers visit **many** stores
  - m:n (1:n + n:1)
  - **One** shopper visits **many** stores +
    Many shoppers visit **one** store

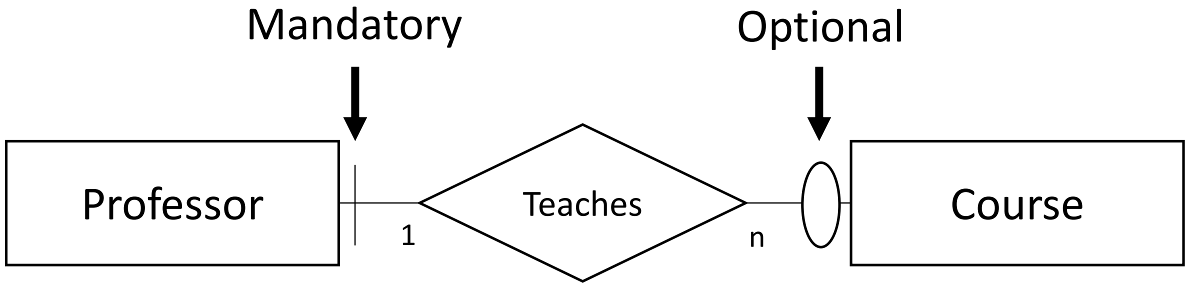
- **One** department has **one** department chair
  - 1:1
  - **One** faculty (Dr. Johnson) is chair of **one** department (DISC)
  - A faculty member cannot chair two departments, and a department cannot have two faculty members as chair!

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

Participation

- Is it required that each instance of an entity participate in the relationship?
  - Yes = Total Participation (Mandatory Participation, or “Existence dependency”)
  - No = Partial Participation (Optional Participation)

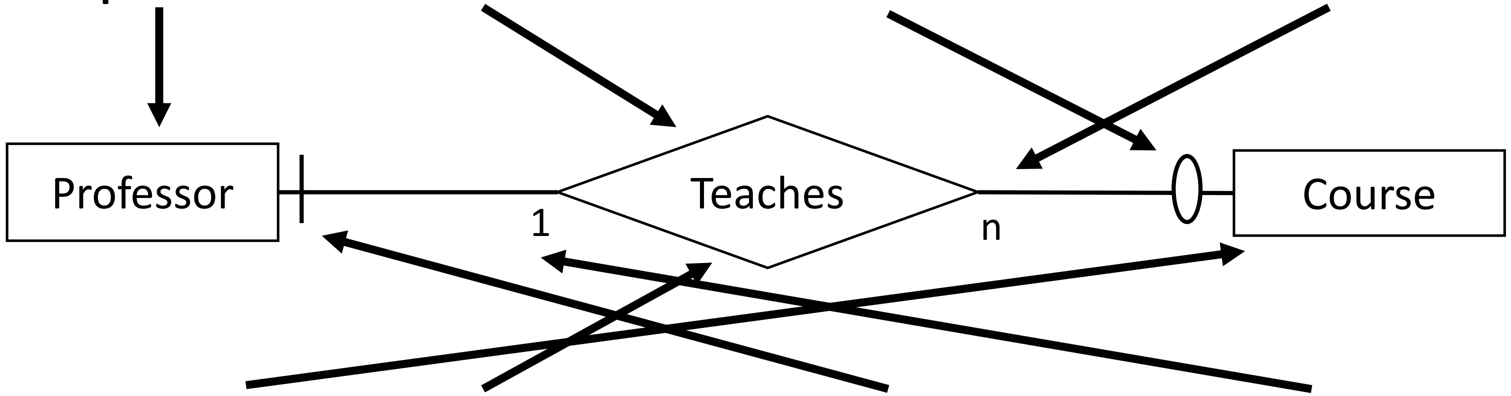
- A professor teaches between 0 and n classes – Partial/Optional
- A course is taught by between 1 and 1 professors – Total/Mandatory



If the course doesn’t participate, it doesn’t exist

Reading ER Diagrams

- For now, we will use the “Look across” method
  - A professor teaches a min of zero and max of n courses

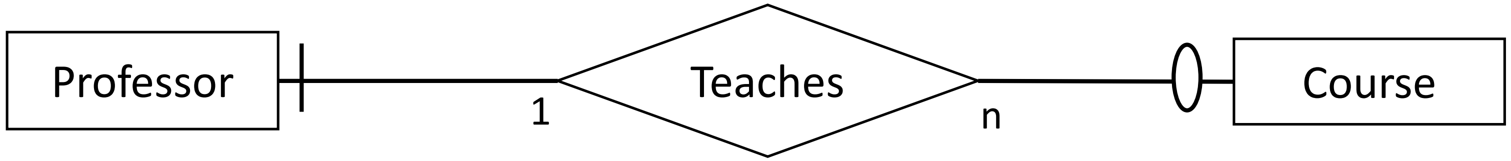


  - A course is taught by a min of 1 and a max of 1 professors

- We “look across” the relationship to determine participation and cardinality of an entity

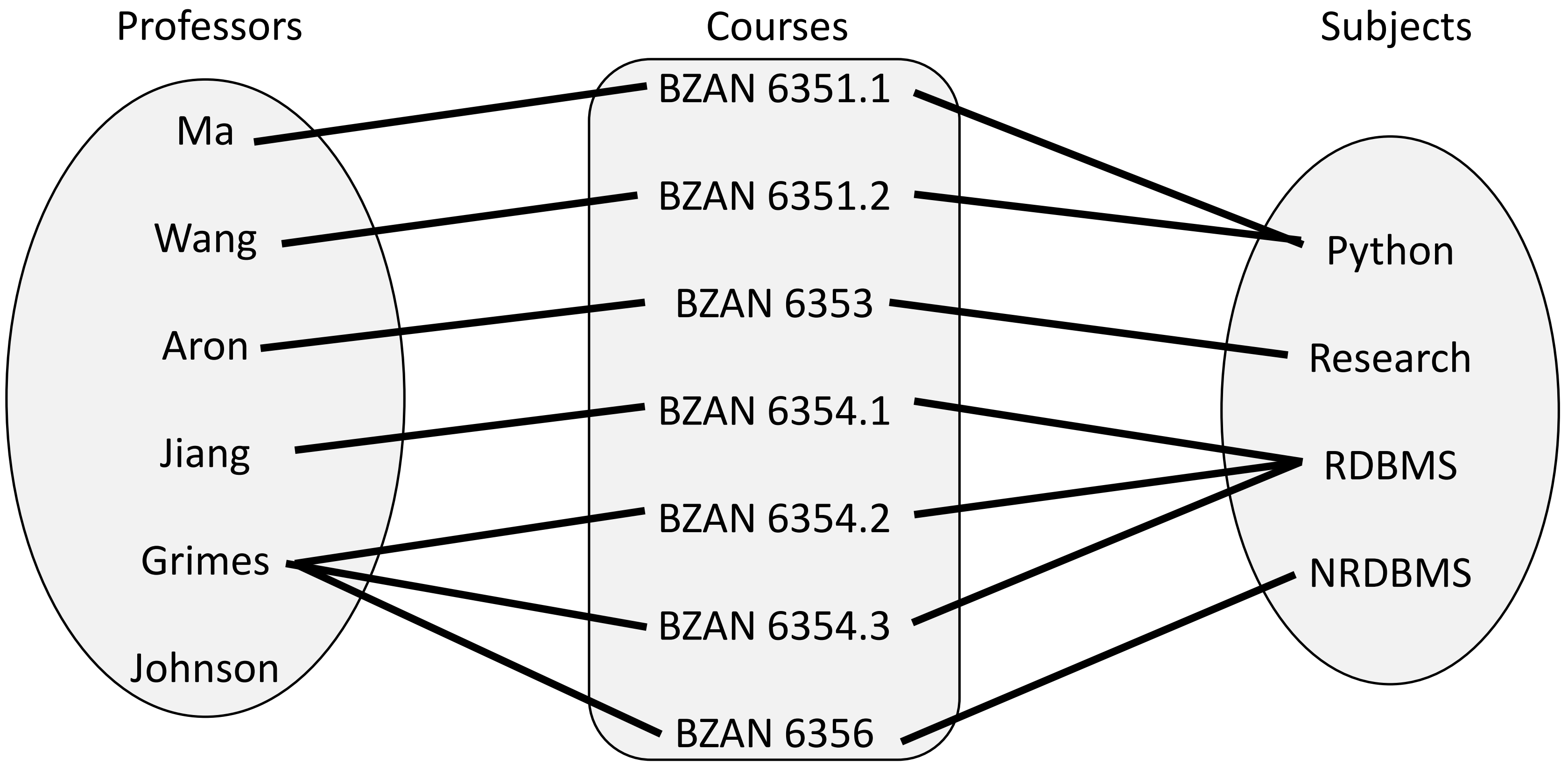
Reading ER Diagrams

- In this case we would say:
  - The relationship (Teaches) has a cardinality of **1:N**
  - Professor has a cardinality of **Many**
  - Course has a cardinality of **One**
  - Professor has **Optional** participation
  - Course has **Mandatory** participation
  - Professor is the **Parent** because it has the cardinality of Many



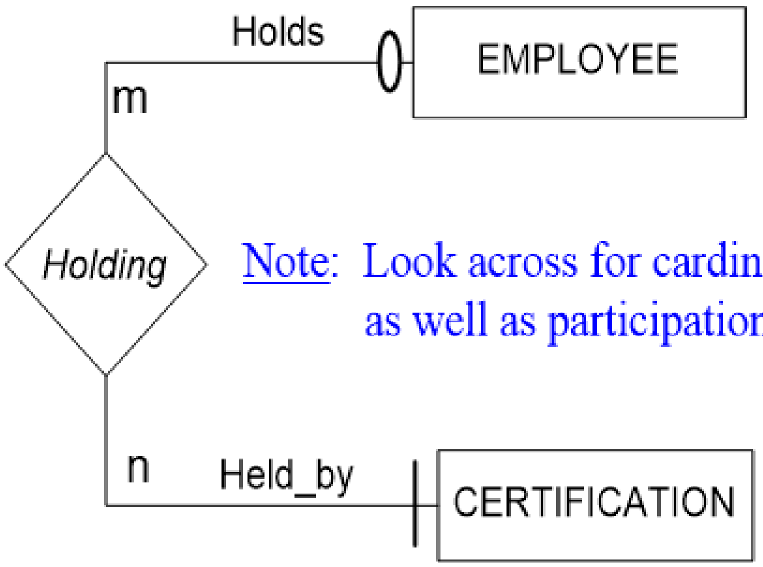
Instance Diagrams

- Good to help with understanding, not practical on a large scale
- Similar to an “object diagram” in universal modeling language (UML)

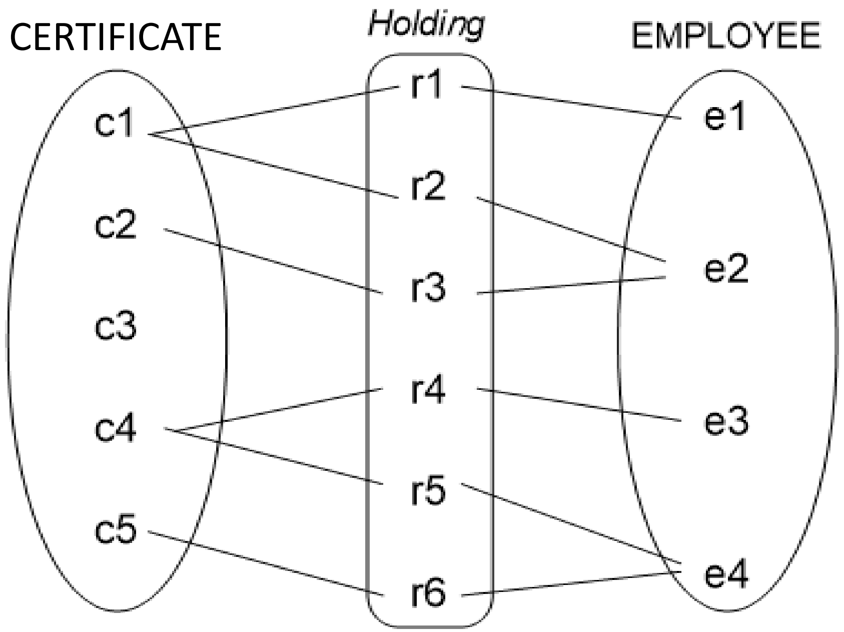


Example 1

- Cardinality Constraint → m:n
- Participation Constraint → Employee: Total (Mandatory)
  Certification: Partial (Optional)

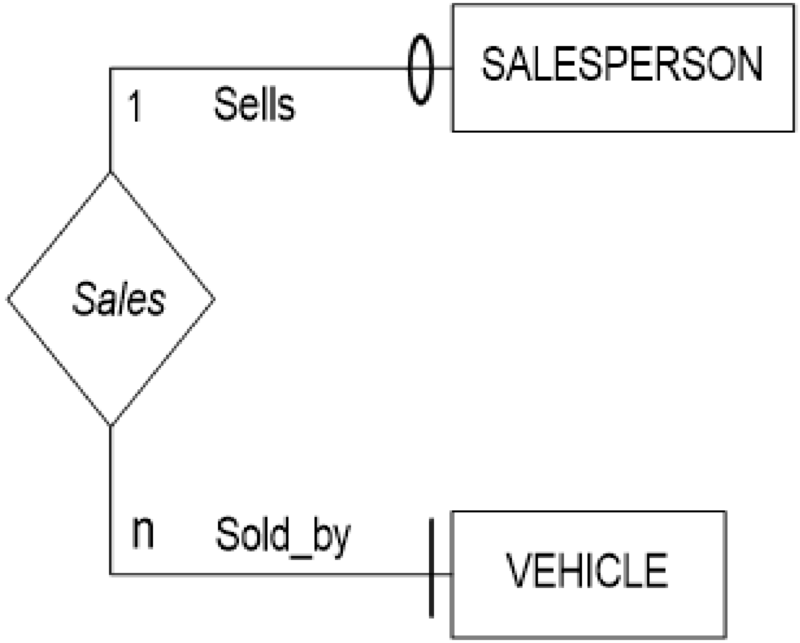


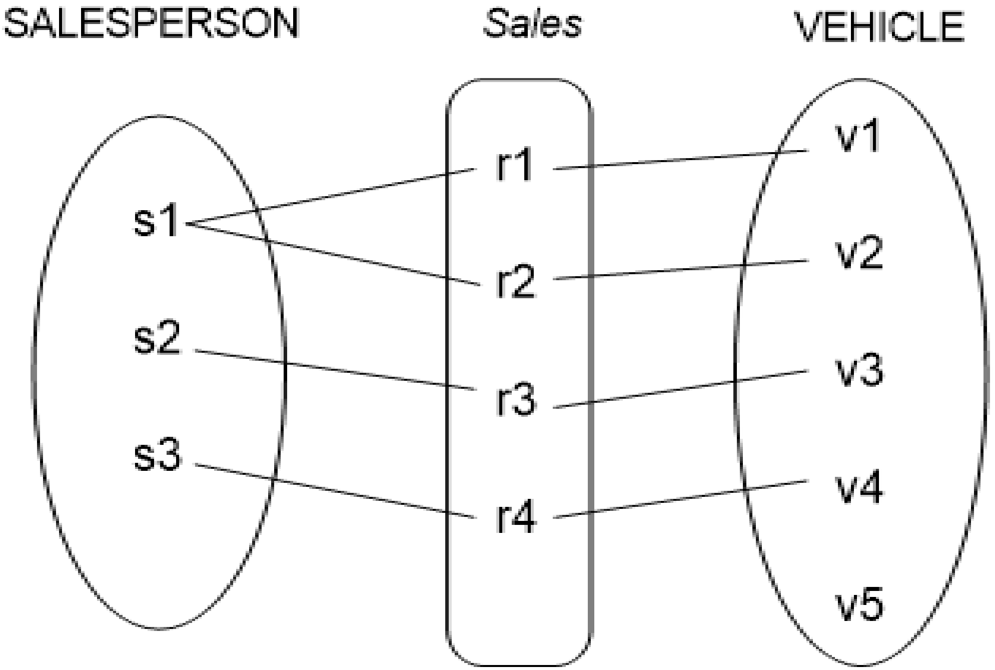
Note: Look across for cardinality constraint as well as participation constraint.



Example 2

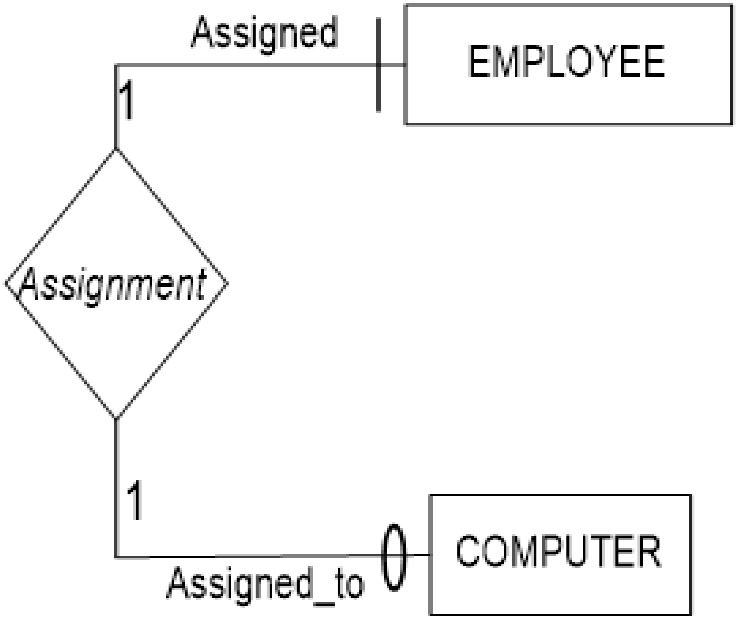
- Cardinality Constraint → 1:n
- Participation Constraint → Salesperson: Total (Mandatory)
  Vehicle: Partial (Optional)

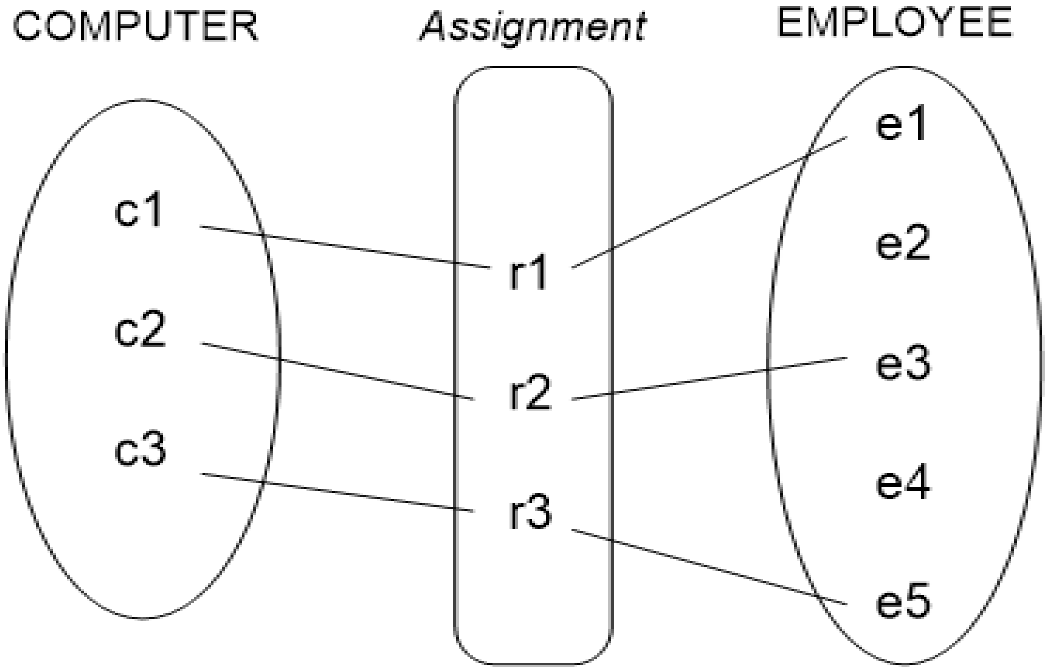




Example 3

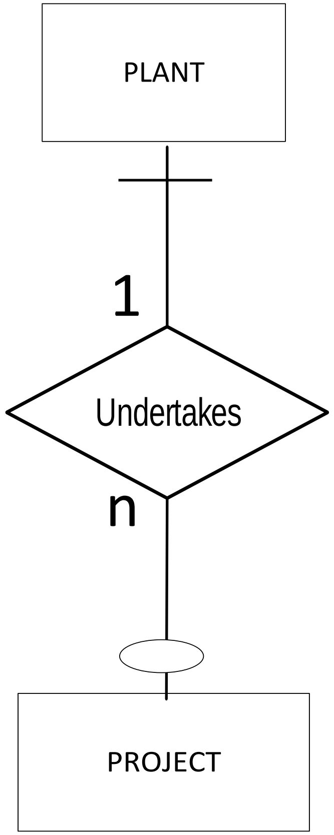
- Cardinality Constraint → 1:1
- Participation Constraint → Employee: Partial (Optional)
  Computer: Total (Mandatory)





Let’s work through one

Bearcat Incorporated is a manufacturing company that has several plants in the US. Plants are responsible for undertaking projects. A certain plant might undertake several projects but a project is always under the control of just one plant. Some plants do not undertake any projects at all.

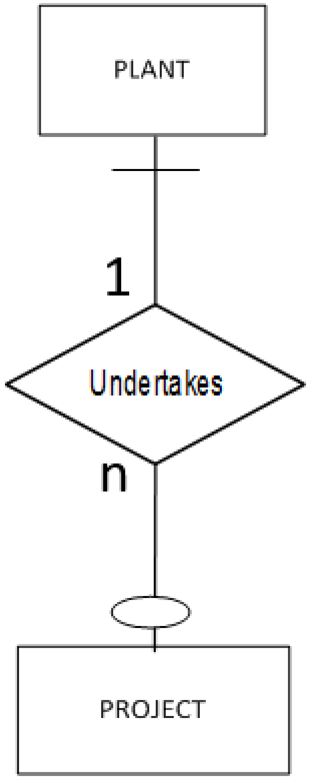


An Instantiation of a 1:n relationship

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_num	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



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

We will discuss in much more detail what a "foreign key" is in future lectures – but for now just know we join rows from two tables together based on a common value of the foreign key and a candidate key to which it refers

Tip: Carefully read and study the examples

- Much of this is easy to “glaze over” and read without understanding
- Go back and really look at each diagram until they make sense
- There are more symbols/notations and concepts that build on this coming up in modules 3 and 6

Module 2.5 (2.3.4 in the book)

Cardinality and Participation

- Describe 1:1, 1:n, n:1 and m:n cardinalities
- Describe the minimum cardinality, partial participation, and total participation constraints

Module 2.6 (2.3.7 in the book)
Deletion Constraints

- Describe deletion constraints and the restrict (R), cascade (C), set null (N), and set default (D) rules

Deletion Rules

- When entities are related, there are implications to deleting data

- Deletion of an instance from a child entity in a m:n or 1:n relationship **requires no action**
  - Deletion of an instance from the child entity in a m:n or 1:n relationship type can have implications that requires attention; however, the solution is unconnected to deletion rules.

- Deletion of an instance from a parent entity in a relationship **requires some type of action**:
  - In order to maintain consistency of data in the entity types participating in the specific relationship, the action may have to be in the parent entity or the child entity

Deletion Rules

- Deletion of an instance from a **child entity type** in a m:n or 1:n relationship requires no action
- When we remove a player (child) from the team, no impact to the coach-player relationship

Deletion Rules

- Deletion of an instance from a **parent entity type** in a relationship requires some type of action
- When we remove the coach (parent) from the team, what is the team going to do now?
  - Refuse to let him go? Stop playing? Play with no coach? Find a new coach?

Deletion Constraints

- **Restrict Rule (R):** When an attempt is made to delete an entity instance from a parent entity in a relationship, the deletion should be disallowed if child entity instances are related to the parent entity instance. The Restrict rule is imposed on the parent entity type in the relationship.
  - When a deletion constraint is not specified, the restrict rule is usually implied by default.

- **Cascade Rule (C):** When an attempt is made to delete an entity instance from a parent entity in a relationship, if all child entity instances related to this parent in this relationship should also be deleted along with the deletion of the parent entity instance. The Cascade rule applies and is imposed on the child entity type in the relationship.

Deletion Constraints (Continued)

- **Set Null Rule (N):** When an attempt is made to delete an entity instance from a parent entity in a relationship, all child entity instances related to this parent in this relationship should be retained but no longer referenced to this parent. The Set Null rule applies and is imposed on the child entity type in the relationship

- **Set Default Rule (D):** When an attempt is made to delete a parent entity instance in a relationship, all child entity instances related to this parent in this relationship should be retained but the child instances should be changed to reference a predefined default parent. The Set Default" rule applies and is imposed on the child entity type in the relationship.

Deletion Rules

- Restrict: Do not allow coach (Parent) to be deleted
- Cascade: Delete all the players (Children) as well
- Set Null: Players (Children) exist, but without a coach (Parent)
- Set Default: A new "default" coach (Parent) is defined

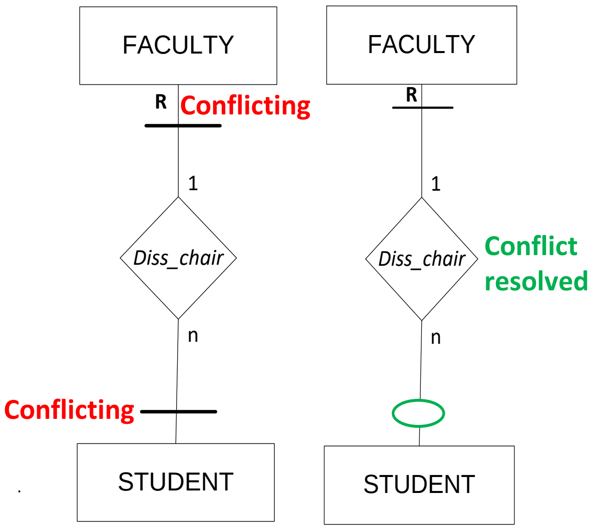
Default Coach
Khator

NO!

Don't Leave me!

RESTRICTED!

Deletion conflicts in a 1:n relationship



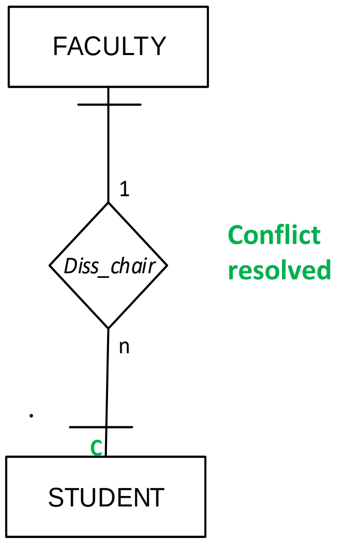
(a1)

(a2)

(a) The restrict Rule

When a parent entity in a relationship is deleted, if the deletion of the parent should be prohibited even if one child entity related to this parent in present, then the **restrict (R)** rule is used.

Restrict Rule:
A faculty member can never leave because they are required to chair a student, and we restrict deletion if they have a student.
Resolved by making it optional to chair a student.

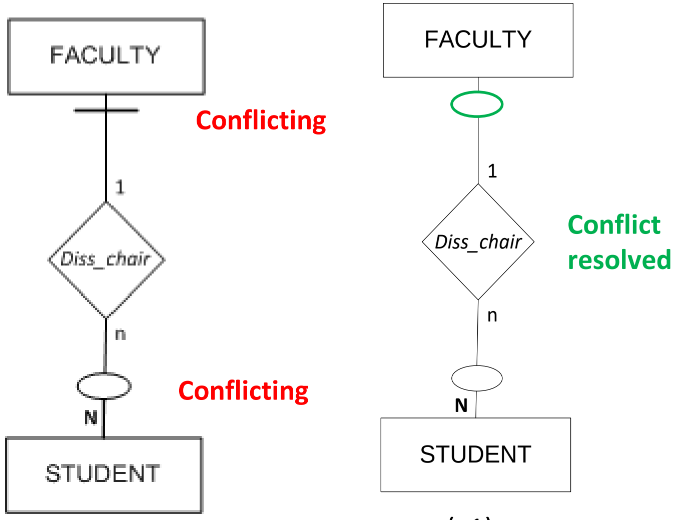


(b) The cascade Rule

When a parent entity in a relationship is deleted, if all child entities related to this parent should be deleted, then the **cascade rule (C)** applies.

Cascade Rule:
A faculty member can leave, but when s/he does all students they are chairing will be deleted (probably not good)

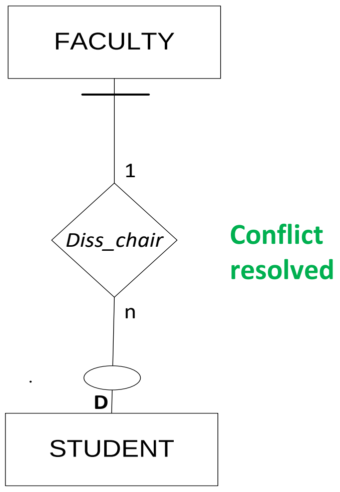
Deletion conflicts in a 1:n relationship



(c2) (c1)

(c) The "set null" Rule

When a parent entity in a relationship is deleted, if all child entities related to this parent in this relationship should be retained but no longer referenced to this parent, the **"set null" (N)** rule applies.



(d) The "set default" Rule

When a parent entity in a relationship is deleted, if all child entities related to this parent in this relationship should be retained, no longer referenced to this parent, but should be referenced to a predefined default parent, the **"set default" (D)** rule applies.

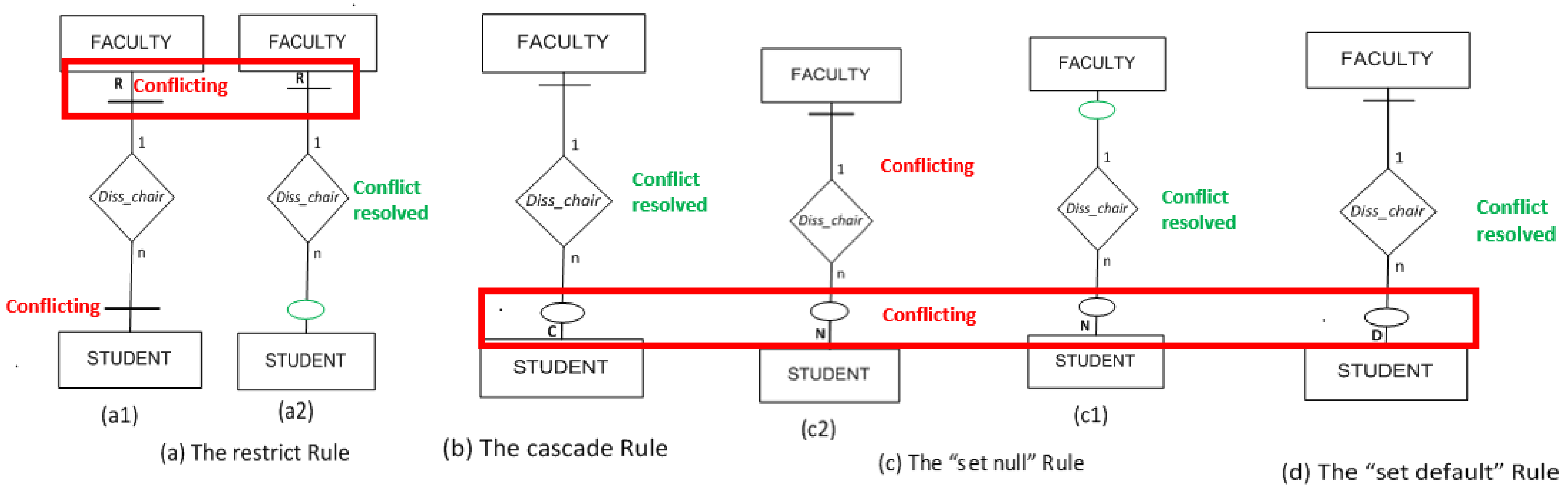
Set Null Rule:
A faculty member who is chairing a student cannot leave because any students they are chairing first have to find a new faculty.

Resolved by not requiring students to have a faculty chair.

Set Default Rule:
A faculty member can leave, and when they do any students they are chairing get reassigned to a default faculty chair.

Deletion conflicts in a 1:n relationship

- Note that the restrict rule applies to the parent, while cascade, set null, and set default apply to the children



Deletion constraints in m:n and 1:1

- Same concept, but…
- Remember that m:n is fundamentally two 1:m relationships – both sides can be considered the "parent" – so the constraints work in both directions
- In 1:1 relationships, where neither side is the "parent", special consideration must be given to deletion constraints
- We will discuss and practice more on this next week.

Module 2.6 (2.3.7 in the book)
Deletion Constraints

- Describe deletion constraints and the restrict (R), cascade (C), set null (N), and set default (D) rules

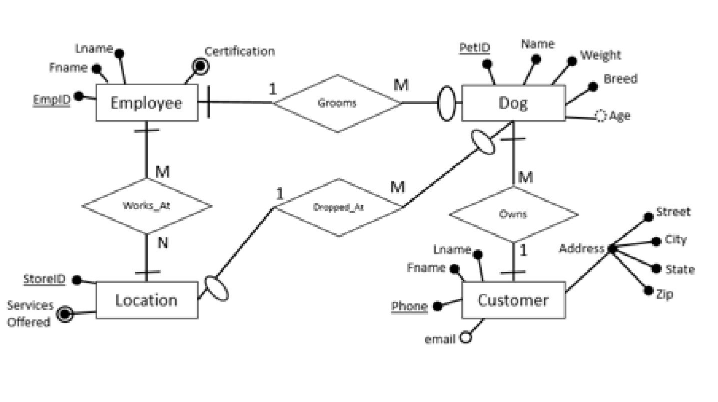
Break

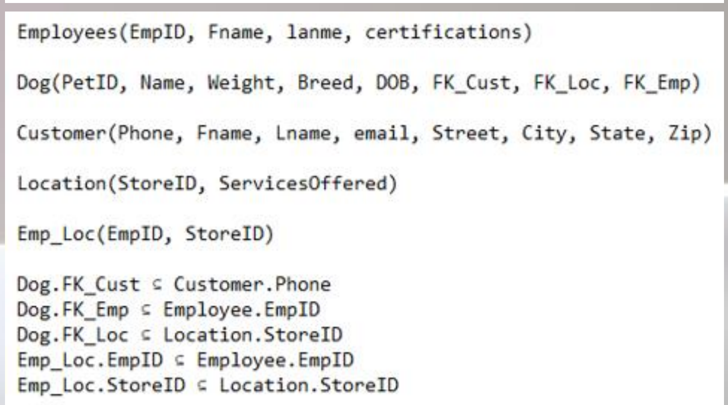
Module 10.2

Implementing Databases – Multiple Tables

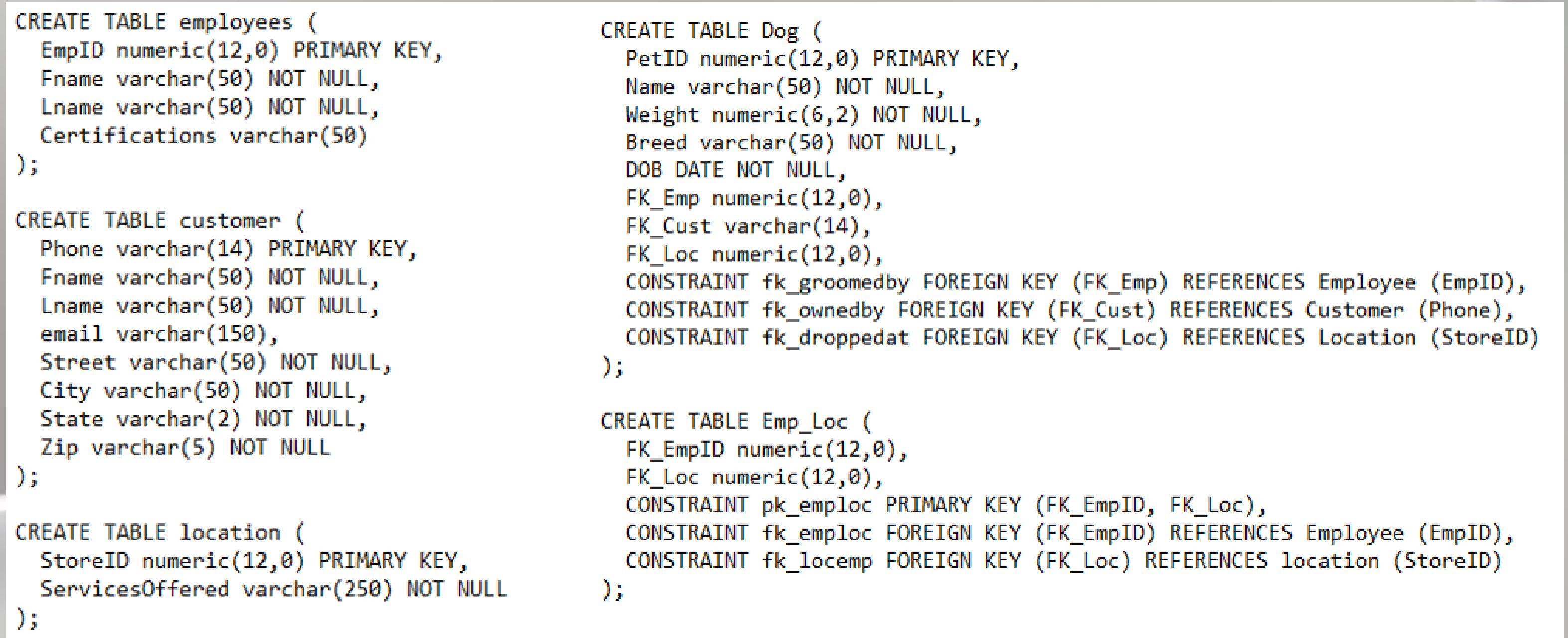
Next step in transforming business rules into an actual DB

ERD → Design Specific ERD → Logical Schema → Physical Schema





```
Employees(EmpID, Fname, lanme, certifications)

Dog(PetID, Name, Weight, Breed, DOB, FK_Cust, FK_Loc, FK_Emp)

Customer(Phone, Fname, Lname, email, Street, City, State, Zip)

Location(StoreID, ServicesOffered)

Emp_Loc(EmpID, StoreID)

Dog.FK_Cust ⊆ Customer.Phone
Dog.FK_Emp ⊆ Employee.EmpID
Dog.FK_Loc ⊆ Location.StoreID
Emp_Loc.EmpID ⊆ Employee.EmpID
Emp_Loc.StoreID ⊆ Location.StoreID
```



```
CREATE TABLE employees (
  EmpID numeric(12,0) PRIMARY KEY,
  Fname varchar(50) NOT NULL,
  Lname varchar(50) NOT NULL,
  Certifications varchar(50)
);

CREATE TABLE customer (
  Phone varchar(14) PRIMARY KEY,
  Fname varchar(50) NOT NULL,
  Lname varchar(50) NOT NULL,
  email varchar(150),
  Street varchar(50) NOT NULL,
  City varchar(50) NOT NULL,
  State varchar(2) NOT NULL,
  Zip varchar(5) NOT NULL
);

CREATE TABLE location (
  StoreID numeric(12,0) PRIMARY KEY,
  ServicesOffered varchar(250) NOT NULL
);
```

```
CREATE TABLE Dog (
  PetID numeric(12,0) PRIMARY KEY,
  Name varchar(50) NOT NULL,
  Weight numeric(6,2) NOT NULL,
  Breed varchar(50) NOT NULL,
  DOB DATE NOT NULL,
  FK_Emp numeric(12,0),
  FK_Cust varchar(14),
  FK_Loc numeric(12,0),
  CONSTRAINT fk_groomedby FOREIGN KEY (FK_Emp) REFERENCES Employee (EmpID),
  CONSTRAINT fk_ownedby FOREIGN KEY (FK_Cust) REFERENCES Customer (Phone),
  CONSTRAINT fk_droppedat FOREIGN KEY (FK_Loc) REFERENCES Location (StoreID)
);

CREATE TABLE Emp_Loc (
  FK_EmpID numeric(12,0),
  FK_Loc numeric(12,0),
  CONSTRAINT pk_emploc PRIMARY KEY (FK_EmpID, FK_Loc),
  CONSTRAINT fk_emploc FOREIGN KEY (FK_EmpID) REFERENCES Employee (EmpID),
  CONSTRAINT fk_locemp FOREIGN KEY (FK_Loc) REFERENCES location (StoreID)
);
```

Where we left off last week…

- We created a table to collect data about horses including:
  - Name (Unique)
  - Color (With a domain and Default of “UNK”)
  - Spots (With a domain and Default of “UNK”)
  - Sex (With a domain and required)
  - Weight (With a domain)
  - Owner (Which we added with the ALTER command)

- Let’s recreate this table and also include an attribute for the owner’s phone number.

- To make sure we are all at the same point, I suggest to first run the command:

  ```
  DROP TABLE Horses;
  ```

Recreating the Horses table / schema

```
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),
 phone varchar(14),
 CONSTRAINT chk_weight CHECK (weight >= 800 AND weight <=2200),
 CONSTRAINT chk_sex CHECK (sex IN ('M','F'))
);
```

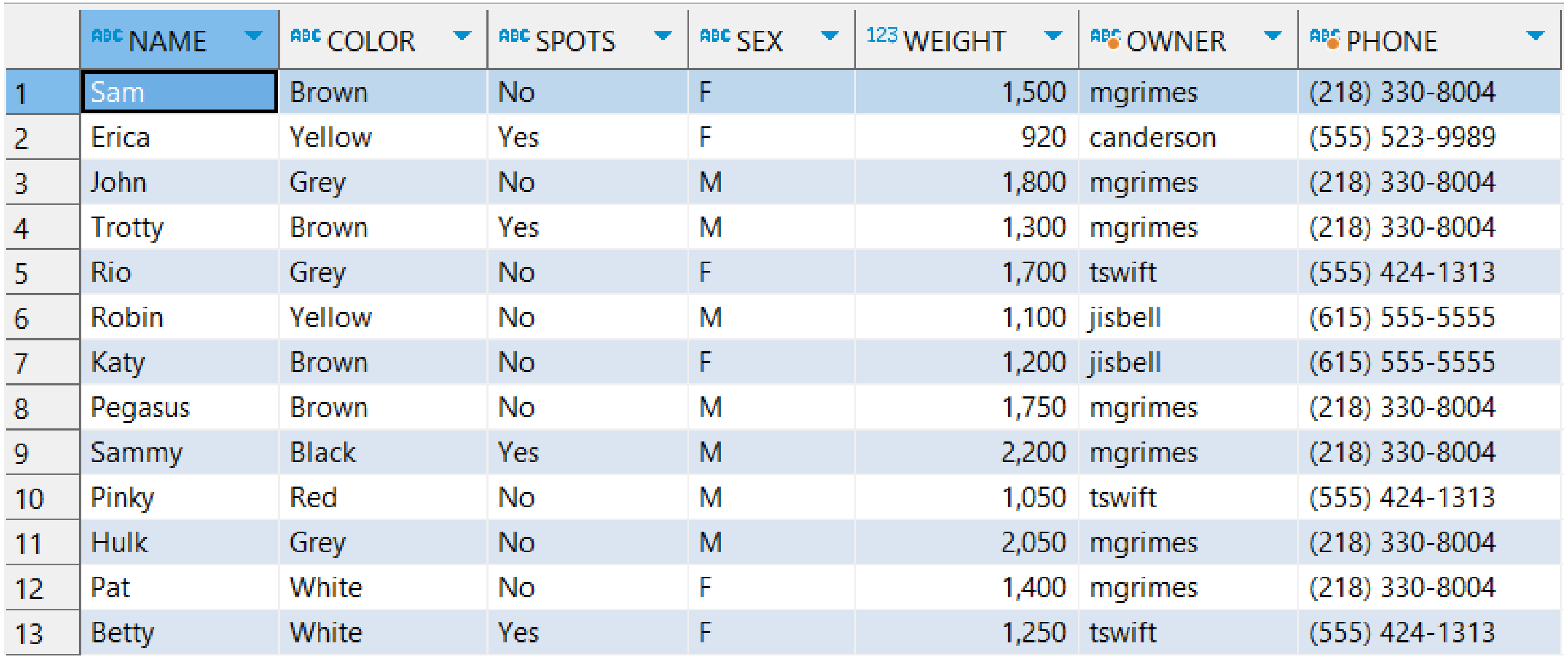
Inserting data for Horses

- Including owner name and phone number

```
INSERT INTO horses (name, color, spots, sex, weight, owner, phone) VALUES ('Sam', 'Brown', 'No', 'F', 1500, 'mgrimes', '(218) 330-8004');
INSERT INTO horses (name, color, spots, sex, weight, owner, phone) VALUES ('Erica', 'Yellow', 'Yes', 'F', 920, 'canderson', '(555) 523-9989');
INSERT INTO horses (name, color, spots, sex, weight, owner, phone) VALUES ('John', 'Grey', 'No', 'M', 1800, 'mgrimes', '(218) 330-8004');
INSERT INTO horses (name, color, spots, sex, weight, owner, phone) VALUES ('Trotty', 'Brown', 'Yes', 'M', 1300, 'mgrimes', '(218) 330-8004');
INSERT INTO horses (name, color, spots, sex, weight, owner, phone) VALUES ('Rio', 'Grey', 'No', 'F', 1700, 'tswift', '(555) 424-1313');
INSERT INTO horses (name, color, spots, sex, weight, owner, phone) VALUES ('Robin', 'Yellow', 'No', 'M', 1100, 'jisbell', '(615) 555-5555');
INSERT INTO horses (name, color, spots, sex, weight, owner, phone) VALUES ('Katy', 'Brown', 'No', 'F', 1200, 'jisbell', '(615) 555-5555');
INSERT INTO horses (name, color, spots, sex, weight, owner, phone) VALUES ('Pegasus', 'Brown', 'No', 'M', 1750, 'mgrimes', '(218) 330-8004');
INSERT INTO horses (name, color, spots, sex, weight, owner, phone) VALUES ('Sammy', 'Black', 'Yes', 'M', 2200, 'mgrimes', '(218) 330-8004');
INSERT INTO horses (name, color, spots, sex, weight, owner, phone) VALUES ('Pinky', 'Red', 'No', 'M', 1050, 'tswift', '(555) 424-1313');
INSERT INTO horses (name, color, spots, sex, weight, owner, phone) VALUES ('Hulk', 'Grey', 'No', 'M', 2050, 'mgrimes', '(218) 330-8004');
INSERT INTO horses (name, color, spots, sex, weight, owner, phone) VALUES ('Pat', 'White', 'No', 'F', 1400, 'mgrimes', '(218) 330-8004');
INSERT INTO horses (name, color, spots, sex, weight, owner, phone) VALUES ('Betty', 'White', 'Yes', 'F', 1250, 'tswift', '(555) 424-1313');
```

SELECTing to see where we are so far…

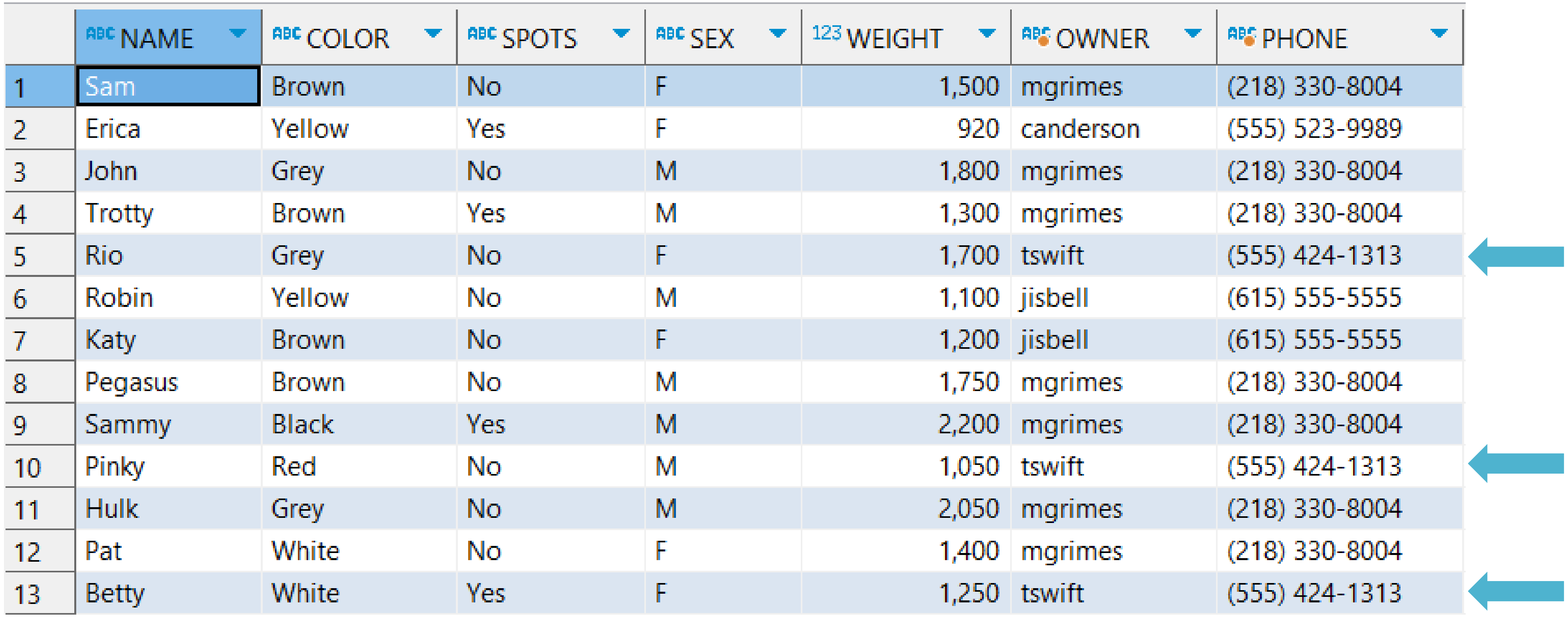
- `SELECT * FROM Horses;`



	NAME	COLOR	SPOTS	SEX	WEIGHT	OWNER	PHONE
1	Sam	Brown	No	F	1,500	mgrimes	(218) 330-8004
2	Erica	Yellow	Yes	F	920	canderson	(555) 523-9989
3	John	Grey	No	M	1,800	mgrimes	(218) 330-8004
4	Trotty	Brown	Yes	M	1,300	mgrimes	(218) 330-8004
5	Rio	Grey	No	F	1,700	tswift	(555) 424-1313
6	Robin	Yellow	No	M	1,100	jisbell	(615) 555-5555
7	Katy	Brown	No	F	1,200	jisbell	(615) 555-5555
8	Pegasus	Brown	No	M	1,750	mgrimes	(218) 330-8004
9	Sammy	Black	Yes	M	2,200	mgrimes	(218) 330-8004
10	Pinky	Red	No	M	1,050	tswift	(555) 424-1313
11	Hulk	Grey	No	M	2,050	mgrimes	(218) 330-8004
12	Pat	White	No	F	1,400	mgrimes	(218) 330-8004
13	Betty	White	Yes	F	1,250	tswift	(555) 424-1313

Do you see any problem with this? Data Redundancy

- If someone owns more than one horse, we are storing their phone number multiple times
  - This is inefficient use of storage
  - If we need to update their phone number, we have to do so in multiple places
  - Storing multiple copies of the same data makes it possible to have data integrity problems



	NAME	COLOR	SPOTS	SEX	WEIGHT	OWNER	PHONE
1	Sam	Brown	No	F	1,500	mgrimes	(218) 330-8004
2	Erica	Yellow	Yes	F	920	canderson	(555) 523-9989
3	John	Grey	No	M	1,800	mgrimes	(218) 330-8004
4	Trotty	Brown	Yes	M	1,300	mgrimes	(218) 330-8004
5	Rio	Grey	No	F	1,700	tswift	(555) 424-1313
6	Robin	Yellow	No	M	1,100	jisbell	(615) 555-5555
7	Katy	Brown	No	F	1,200	jisbell	(615) 555-5555
8	Pegasus	Brown	No	M	1,750	mgrimes	(218) 330-8004
9	Sammy	Black	Yes	M	2,200	mgrimes	(218) 330-8004
10	Pinky	Red	No	M	1,050	tswift	(555) 424-1313
11	Hulk	Grey	No	M	2,050	mgrimes	(218) 330-8004
12	Pat	White	No	F	1,400	mgrimes	(218) 330-8004
13	Betty	White	Yes	F	1,250	tswift	(555) 424-1313

Let’s start over again…

- We have identified a problem with our design, and we are not too far in, so lets just drop everything and start over:

```
DROP TABLE horses;
```

- Note that I will do this a lot in this class to keep things clean and simple, but it is not “normal” to frequently drop and recreate tables on a database that is being used since all your data would be lost. It would be more common to fix problems using the ALTER command to make changes like this.

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.

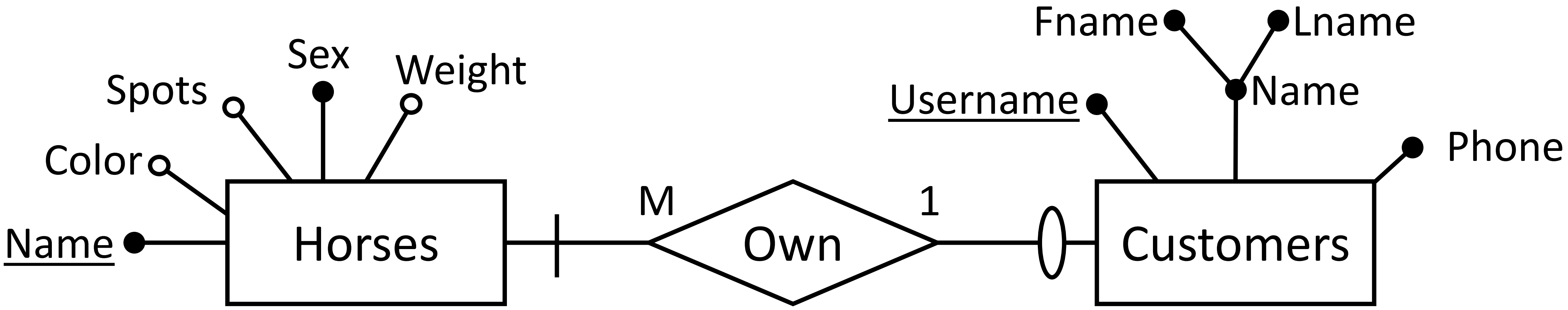
Customers

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.

MG’s Horse Habitat – Horses and Customers



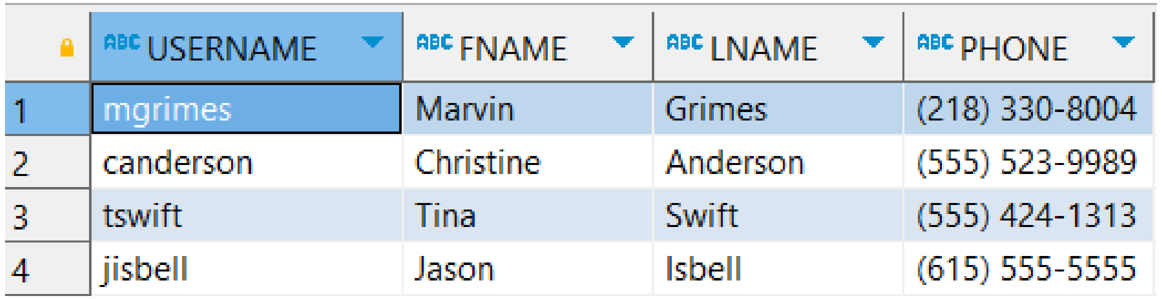
Semantic Integrity Constraints:
Color: {Black, White, Brown, Grey, Red, Yellow}, Default: 'UNK'
Spots: {Yes, No}, Default: 'UNK’
Sex: {M, F}
Weight: {800-2200}

Which entity is the “Parent” in this relationship?

Let’s first create Customers

```
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
);
```

```
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');
```



	USERNAME	FNAME	LNAME	PHONE
1	mgrimes	Marvin	Grimes	(218) 330-8004
2	canderson	Christine	Anderson	(555) 523-9989
3	tswift	Tina	Swift	(555) 424-1313
4	jisbell	Jason	Isbell	(615) 555-5555

Now let’s recreate Horses with a FOREIGN KEY

```
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)
);
```

This is where the magic happens!

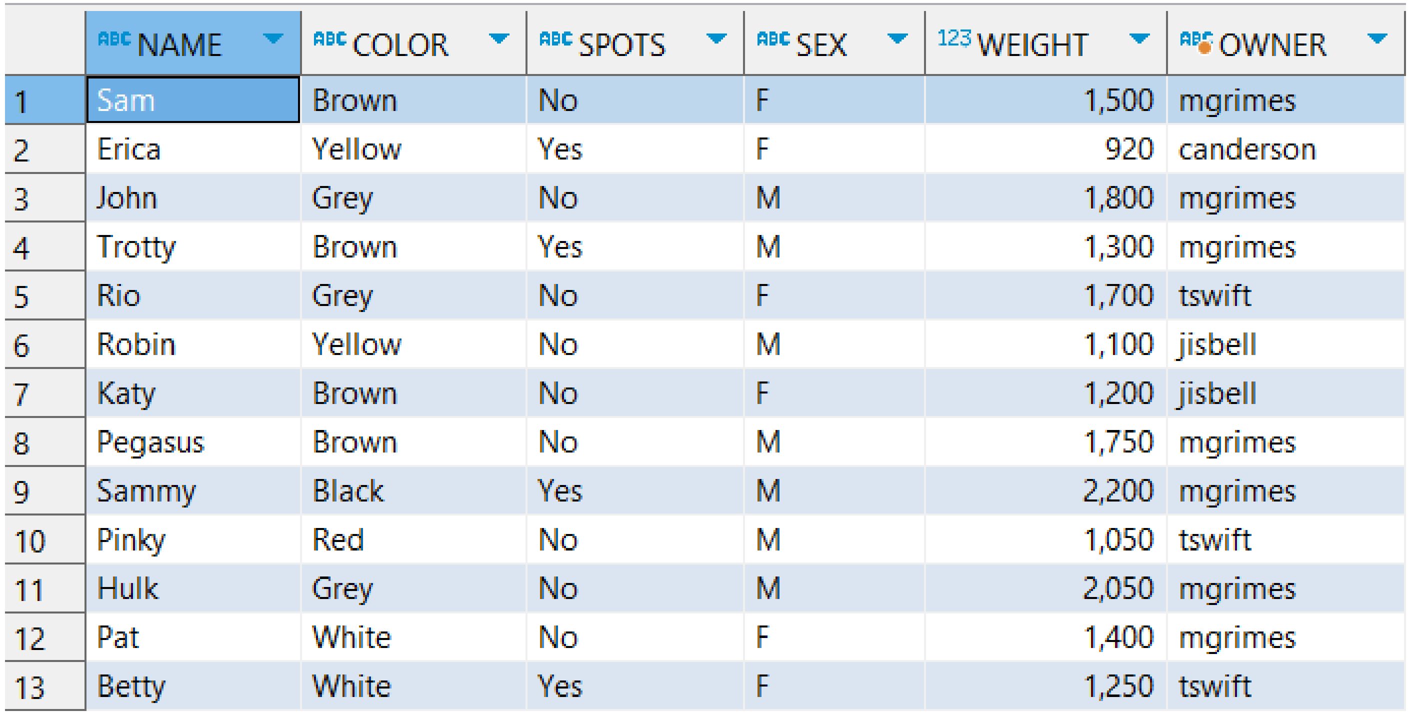
INSERT our Horses data back in (with owner, but not phone)

```
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');
```

So now we have Horses and Customers…

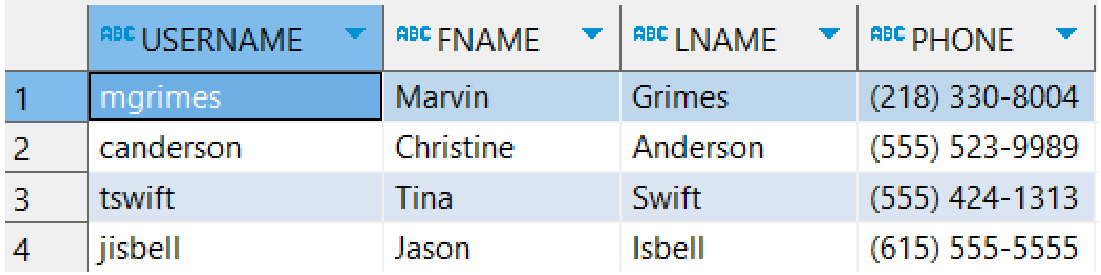
- Now we have two tables, but how to bring the data together?

```
SELECT * FROM Horses;
```



	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

```
SELECT * FROM Customers;
```

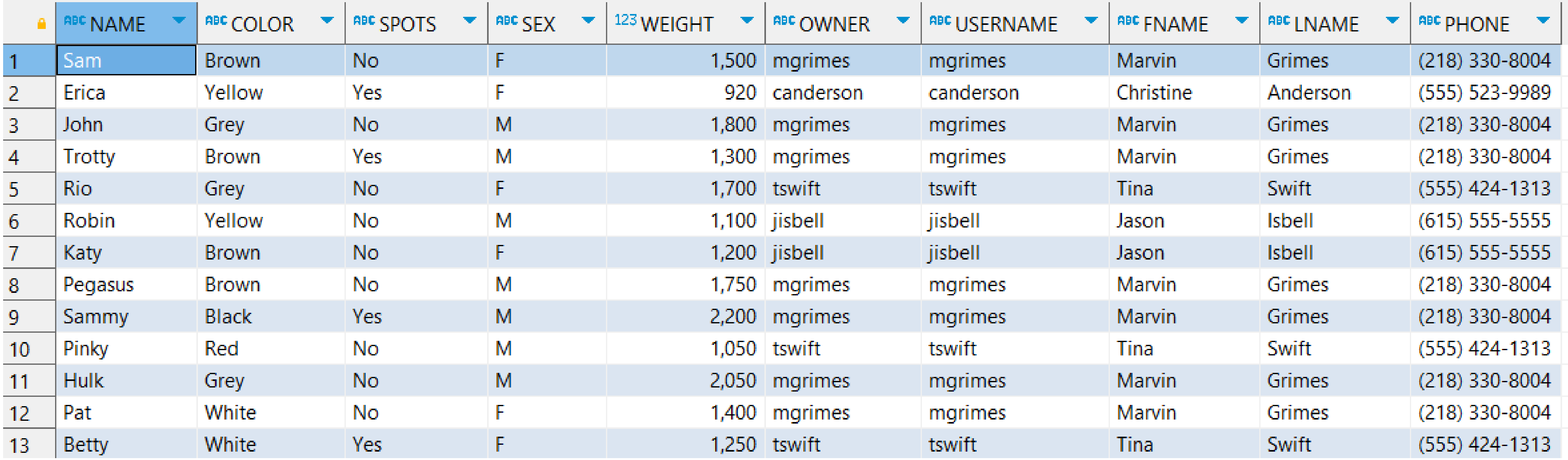


	USERNAME	FNAME	LNAME	PHONE
1	mgrimes	Marvin	Grimes	(218) 330-8004
2	canderson	Christine	Anderson	(555) 523-9989
3	tswift	Tina	Swift	(555) 424-1313
4	jisbell	Jason	Isbell	(615) 555-5555

Joining tables together

- We will have a lot more discussion about JOINs later, but for now… Introducing the INNER JOIN!

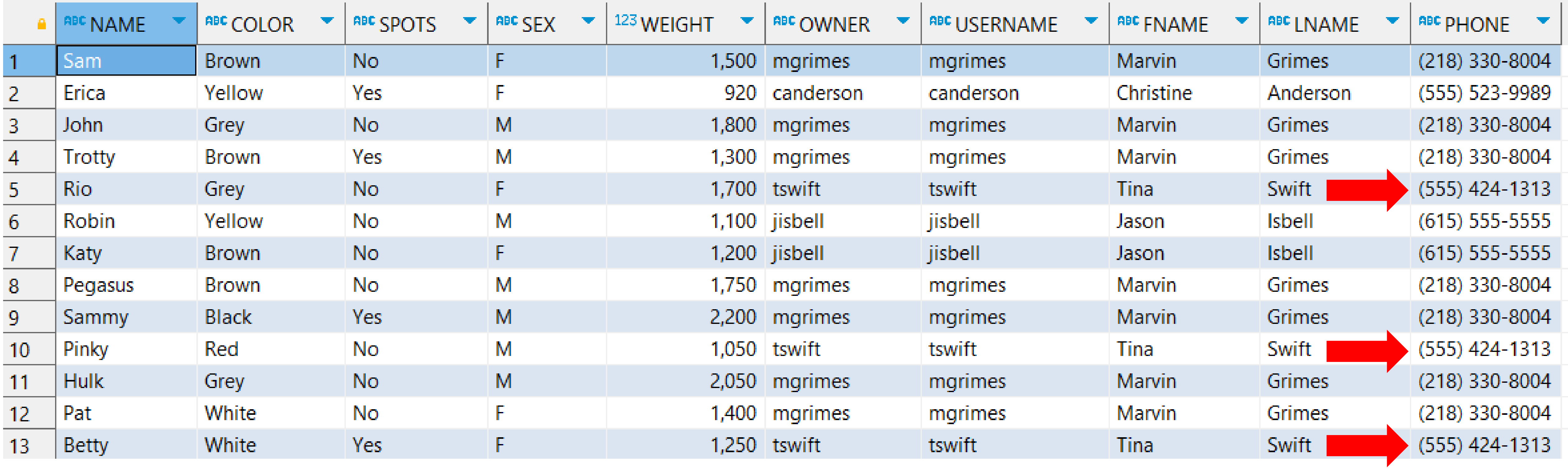
```
SELECT * FROM Horses INNER JOIN Customers ON Horses.owner = Customers.username;
```



	NAME	COLOR	SPOTS	SEX	WEIGHT	OWNER	USERNAME	FNAME	LNAME	PHONE
1	Sam	Brown	No	F	1,500	mgrimes	mgrimes	Marvin	Grimes	(218) 330-8004
2	Erica	Yellow	Yes	F	920	canderson	canderson	Christine	Anderson	(555) 523-9989
3	John	Grey	No	M	1,800	mgrimes	mgrimes	Marvin	Grimes	(218) 330-8004
4	Trotty	Brown	Yes	M	1,300	mgrimes	mgrimes	Marvin	Grimes	(218) 330-8004
5	Rio	Grey	No	F	1,700	tswift	tswift	Tina	Swift	(555) 424-1313
6	Robin	Yellow	No	M	1,100	jisbell	jisbell	Jason	Isbell	(615) 555-5555
7	Katy	Brown	No	F	1,200	jisbell	jisbell	Jason	Isbell	(615) 555-5555
8	Pegasus	Brown	No	M	1,750	mgrimes	mgrimes	Marvin	Grimes	(218) 330-8004
9	Sammy	Black	Yes	M	2,200	mgrimes	mgrimes	Marvin	Grimes	(218) 330-8004
10	Pinky	Red	No	M	1,050	tswift	tswift	Tina	Swift	(555) 424-1313
11	Hulk	Grey	No	M	2,050	mgrimes	mgrimes	Marvin	Grimes	(218) 330-8004
12	Pat	White	No	F	1,400	mgrimes	mgrimes	Marvin	Grimes	(218) 330-8004
13	Betty	White	Yes	F	1,250	tswift	tswift	Tina	Swift	(555) 424-1313

Joining tables together

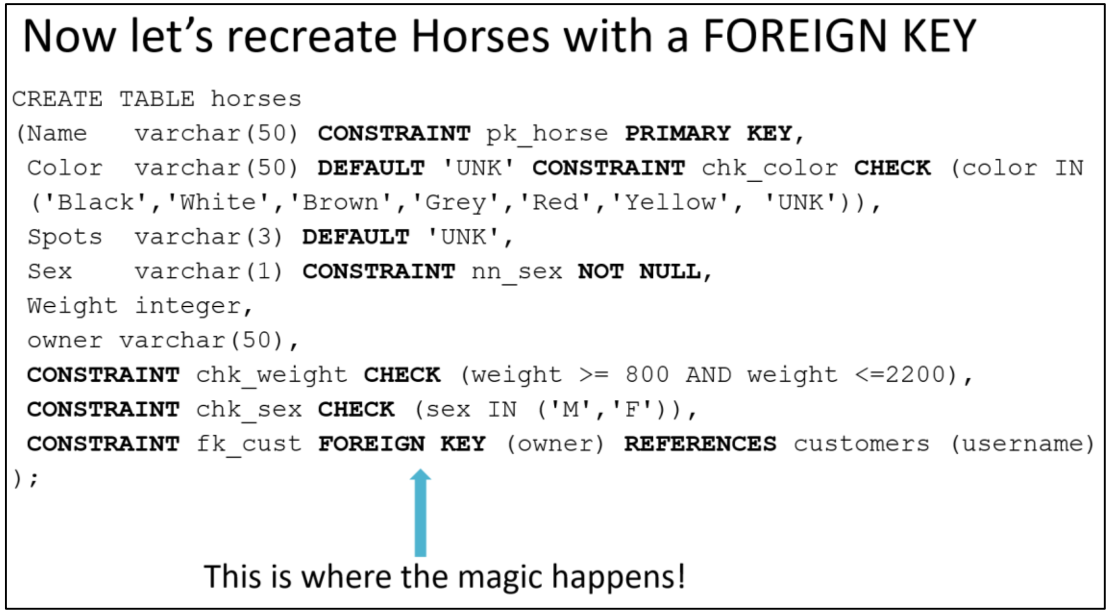
- Note that we still see the repetition of phone number, HOWEVER, this is only in the presentation of the data, the actual value is only stored once
    - No data integrity problems because even though we see the value multiple times in this view, the value is stored only once in the Customers table!



	NAME	COLOR	SPOTS	SEX	WEIGHT	OWNER	USERNAME	FNAME	LNAME	PHONE
1	Sam	Brown	No	F	1,500	mgrimes	mgrimes	Marvin	Grimes	(218) 330-8004
2	Erica	Yellow	Yes	F	920	canderson	canderson	Christine	Anderson	(555) 523-9989
3	John	Grey	No	M	1,800	mgrimes	mgrimes	Marvin	Grimes	(218) 330-8004
4	Trotty	Brown	Yes	M	1,300	mgrimes	mgrimes	Marvin	Grimes	(218) 330-8004
5	Rio	Grey	No	F	1,700	tswift	tswift	Tina	Swift	(555) 424-1313
6	Robin	Yellow	No	M	1,100	jisbell	jisbell	Jason	Isbell	(615) 555-5555
7	Katy	Brown	No	F	1,200	jisbell	jisbell	Jason	Isbell	(615) 555-5555
8	Pegasus	Brown	No	M	1,750	mgrimes	mgrimes	Marvin	Grimes	(218) 330-8004
9	Sammy	Black	Yes	M	2,200	mgrimes	mgrimes	Marvin	Grimes	(218) 330-8004
10	Pinky	Red	No	M	1,050	tswift	tswift	Tina	Swift	(555) 424-1313
11	Hulk	Grey	No	M	2,050	mgrimes	mgrimes	Marvin	Grimes	(218) 330-8004
12	Pat	White	No	F	1,400	mgrimes	mgrimes	Marvin	Grimes	(218) 330-8004
13	Betty	White	Yes	F	1,250	tswift	tswift	Tina	Swift	(555) 424-1313

So… what is a FOREIGN KEY?

- I said “This is where the magic happens”
- The Foreign Key Constraint is how “Referential Integrity” is maintained
    - This is important, and we’ll talk more about it later



Now let’s recreate Horses with a FOREIGN KEY

```
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)
);
```

This is where the magic happens!

- In short: rows of data in two tables can be brought together based on a common value of the **Foreign Key** and the **Candidate Key** it refers to.
- Precisely (**and importantly**): All values of a Foreign Key MUST BE in the domain of value of the Candidate Key to which it refers.

All values of a Foreign Key MUST BE in the domain of value of the Candidate Key to which it refers.

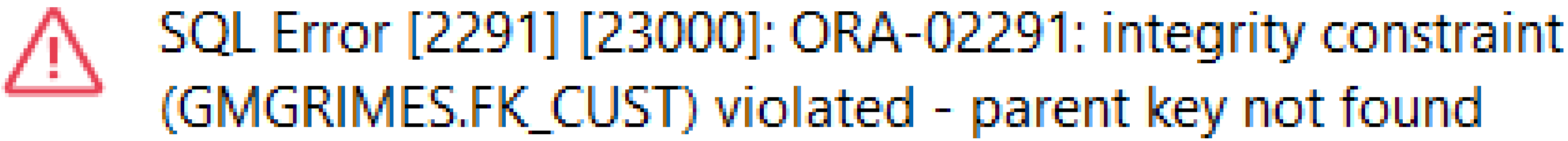
- We cannot have any values for a Foreign Key that are not in the domain of values of the Candidate Key it refers to

- This is how we maintain Referential Integrity
    - All tuples that reference another tuple, must reference a tuple that EXISTS (tuple is essentially a more technical name for "row", and we'll discuss this soon)

- In our current example this means:
    - A horse cannot have an owner that does not exist in the Customers table
    - We cannot delete a customer as long as they have a horse
    - When creating the tables, the Customers table MUST be created BEFORE the Horses table, since Horses refers to Customers

What if we try to INSERT a Horse, with an owner that is not a Customer?

```
INSERT INTO horses (name, color, spots, sex, weight, owner)
VALUES ('Shamrock', 'Black', 'No', 'M', 1400, 'ssimpson’);
```

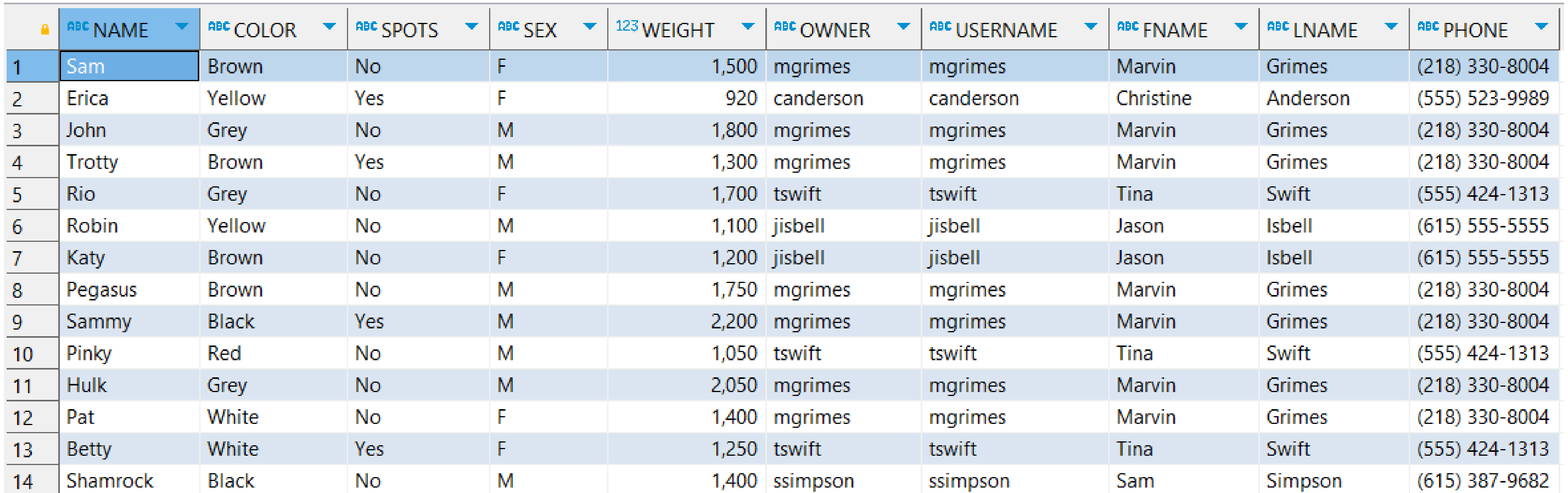


```
SQL Error [2291] [23000]: ORA-02291: integrity constraint
(GMGRIMES.FK_CUST) violated - parent key not found
```

- The constraint “FK_Cust” was violated because the “Parent” key was not found
    - We tried to insert a value for the Foreign Key (horses.owner) that was not in the domain of the Candidate Key (customers.username)

INSERT the Customer (Parent) first, then the Horse (Child)

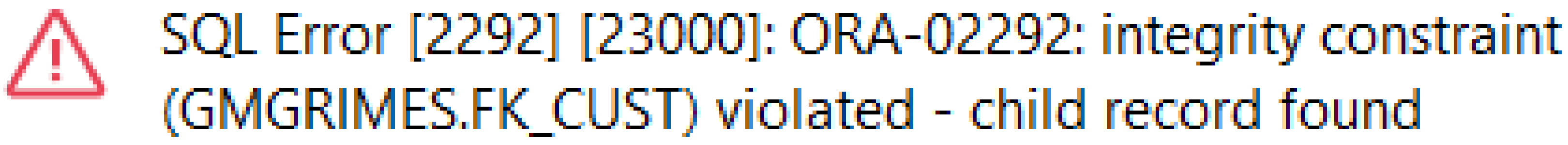
```
INSERT INTO customers (username, fname, lname, phone)
VALUES ('ssimpson', 'Sam', 'Simpson', '(615) 387-9682');

INSERT INTO horses (name, color, spots, sex, weight, owner)
VALUES ('Shamrock', 'Black', 'No', 'M', 1400, 'ssimpson’);

SELECT * FROM Horses INNER JOIN Customers ON Horses.owner = Customers.username;
```



	NAME	COLOR	SPOTS	SEX	WEIGHT	OWNER	USERNAME	FNAME	LNAME	PHONE
1	Sam	Brown	No	F	1,500	mgrimes	mgrimes	Marvin	Grimes	(218) 330-8004
2	Erica	Yellow	Yes	F	920	canderson	canderson	Christine	Anderson	(555) 523-9989
3	John	Grey	No	M	1,800	mgrimes	mgrimes	Marvin	Grimes	(218) 330-8004
4	Trotty	Brown	Yes	M	1,300	mgrimes	mgrimes	Marvin	Grimes	(218) 330-8004
5	Rio	Grey	No	F	1,700	tswift	tswift	Tina	Swift	(555) 424-1313
6	Robin	Yellow	No	M	1,100	jisbell	jisbell	Jason	Isbell	(615) 555-5555
7	Katy	Brown	No	F	1,200	jisbell	jisbell	Jason	Isbell	(615) 555-5555
8	Pegasus	Brown	No	M	1,750	mgrimes	mgrimes	Marvin	Grimes	(218) 330-8004
9	Sammy	Black	Yes	M	2,200	mgrimes	mgrimes	Marvin	Grimes	(218) 330-8004
10	Pinky	Red	No	M	1,050	tswift	tswift	Tina	Swift	(555) 424-1313
11	Hulk	Grey	No	M	2,050	mgrimes	mgrimes	Marvin	Grimes	(218) 330-8004
12	Pat	White	No	F	1,400	mgrimes	mgrimes	Marvin	Grimes	(218) 330-8004
13	Betty	White	Yes	F	1,250	tswift	tswift	Tina	Swift	(555) 424-1313
14	Shamrock	Black	No	M	1,400	ssimpson	ssimpson	Sam	Simpson	(615) 387-9682

What if we try to DELETE a Customer that has a Horse?

```
DELETE FROM customers WHERE username = 'tswift';
```



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

- The constraint “FK_Cust” was violated because the “Parent” (tswift) has one or more child records (all the horses they own)
  - We tried to delete a parent while it has a child

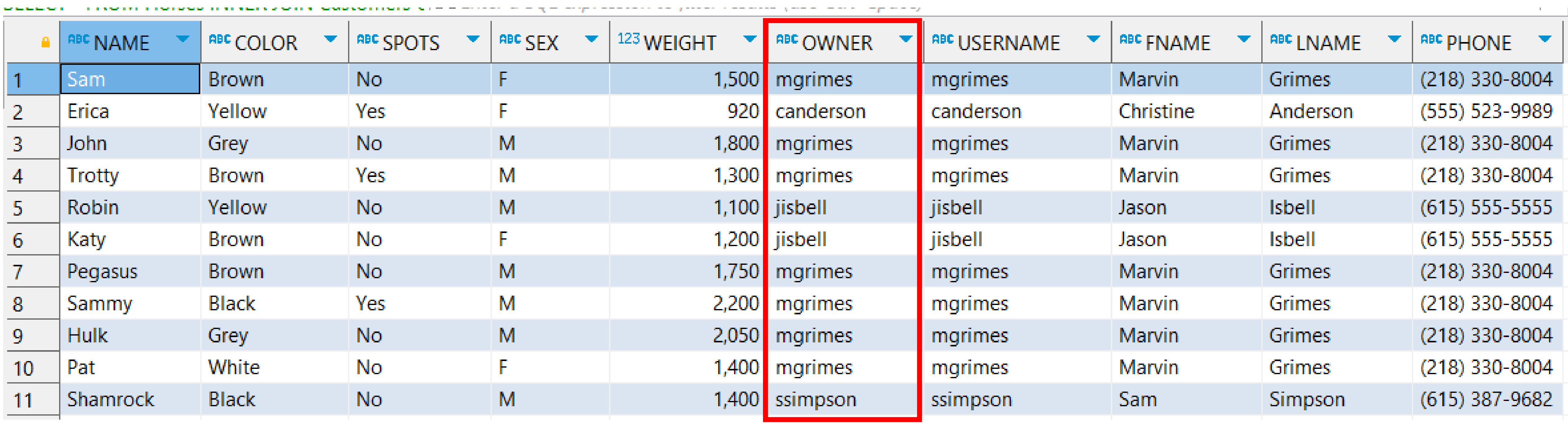
- This is the RESTRICT Deletion Rule in effect!

DELETE the horses first, THEN the customer

```
DELETE FROM horses WHERE owner = 'tswift';
```

```
DELETE FROM customers WHERE username = 'tswift';
```

```
SELECT * FROM Horses INNER JOIN Customers ON Horses.owner = Customers.username;
```



	NAME	COLOR	SPOTS	SEX	WEIGHT	OWNER	USERNAME	FNAME	LNAME	PHONE
1	Sam	Brown	No	F	1,500	mgrimes	mgrimes	Marvin	Grimes	(218) 330-8004
2	Erica	Yellow	Yes	F	920	canderson	canderson	Christine	Anderson	(555) 523-9989
3	John	Grey	No	M	1,800	mgrimes	mgrimes	Marvin	Grimes	(218) 330-8004
4	Trotty	Brown	Yes	M	1,300	mgrimes	mgrimes	Marvin	Grimes	(218) 330-8004
5	Robin	Yellow	No	M	1,100	jisbell	jisbell	Jason	Isbell	(615) 555-5555
6	Katy	Brown	No	F	1,200	jisbell	jisbell	Jason	Isbell	(615) 555-5555
7	Pegasus	Brown	No	M	1,750	mgrimes	mgrimes	Marvin	Grimes	(218) 330-8004
8	Sammy	Black	Yes	M	2,200	mgrimes	mgrimes	Marvin	Grimes	(218) 330-8004
9	Hulk	Grey	No	M	2,050	mgrimes	mgrimes	Marvin	Grimes	(218) 330-8004
10	Pat	White	No	F	1,400	mgrimes	mgrimes	Marvin	Grimes	(218) 330-8004
11	Shamrock	Black	No	M	1,400	ssimpson	ssimpson	Sam	Simpson	(615) 387-9682

Changing the deletion rule from Restrict to Cascade

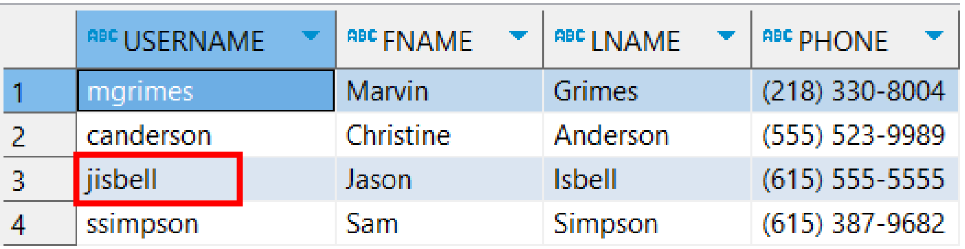
- Instead of Restricting the deletion of a Customer that owns a Horse, we may want to CASCADE the deletion, so that when a Customer is deleted all Horses they own are automatically deleted

```
ALTER TABLE Horses drop CONSTRAINT fk_cust;

ALTER TABLE Horses ADD CONSTRAINT fk_cust FOREIGN KEY (owner)
REFERENCES Customers (Username) ON DELETE CASCADE;
```

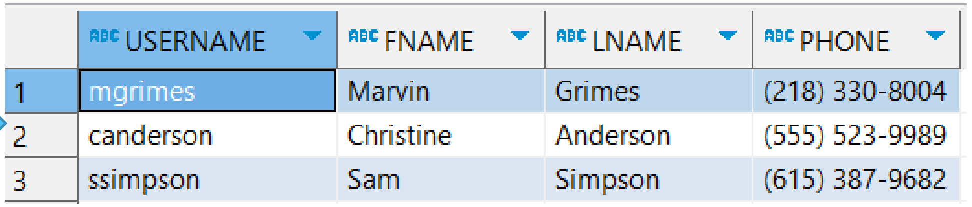
Changing the deletion rule from Restrict to Cascade

- Now when we delete a Customer, Oracle reports that one row was deleted, HOWEVER, on closer inspection we see that not only was the one Customer deleted, but two Horses (owned by that customer) were also deleted!

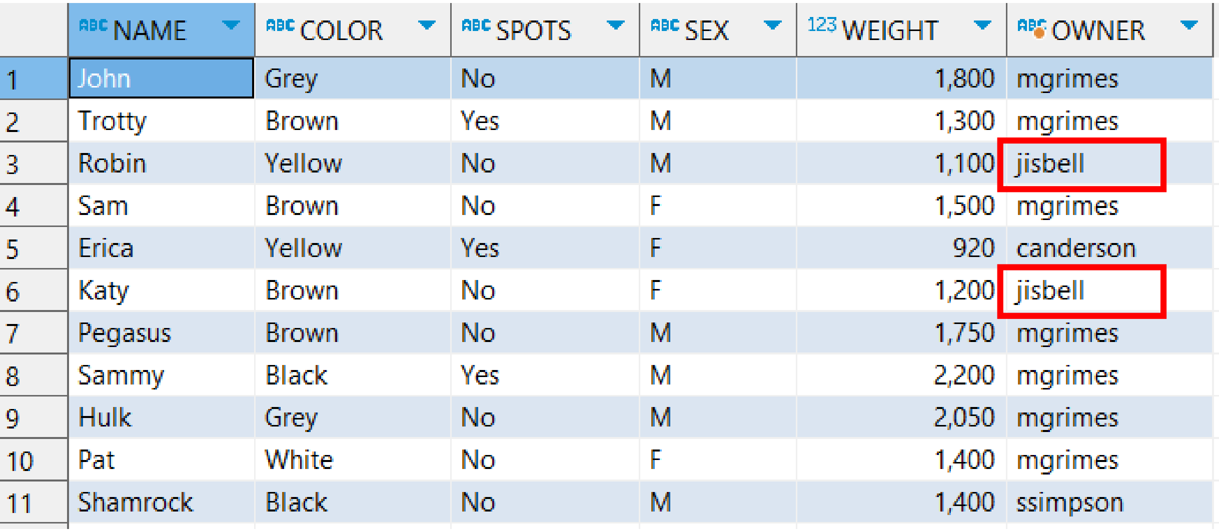
```
DELETE FROM customers WHERE username = 'jisbell';
```



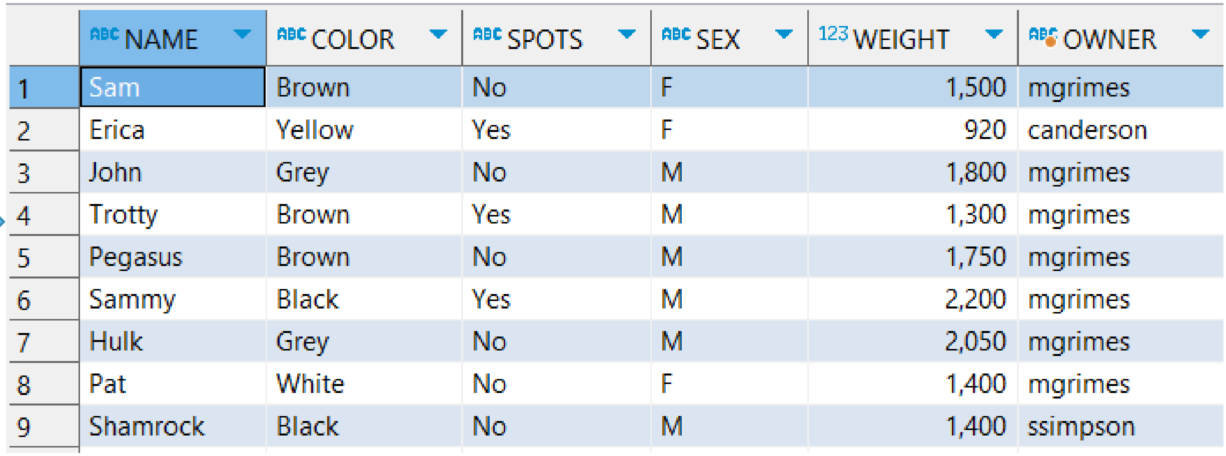
	USERNAME	FNAME	LNAME	PHONE
1	mgrimes	Marvin	Grimes	(218) 330-8004
2	canderson	Christine	Anderson	(555) 523-9989
3	jisbell	Jason	Isbell	(615) 555-5555
4	ssimpson	Sam	Simpson	(615) 387-9682



	USERNAME	FNAME	LNAME	PHONE
1	mgrimes	Marvin	Grimes	(218) 330-8004
2	canderson	Christine	Anderson	(555) 523-9989
3	ssimpson	Sam	Simpson	(615) 387-9682



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

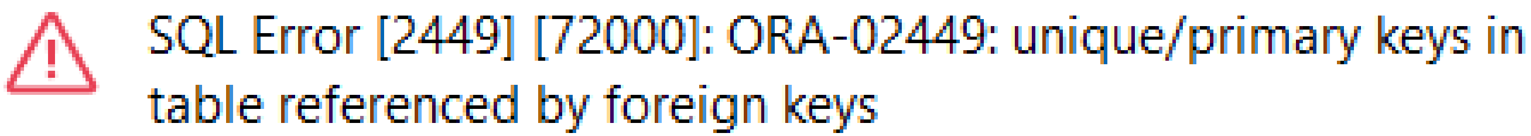


	NAME	COLOR	SPOTS	SEX	WEIGHT	OWNER
1	Sam	Brown	No	F	1,500	mgrimes
2	Erica	Yellow	Yes	F	920	canderson
3	John	Grey	No	M	1,800	mgrimes
4	Trotty	Brown	Yes	M	1,300	mgrimes
5	Pegasus	Brown	No	M	1,750	mgrimes
6	Sammy	Black	Yes	M	2,200	mgrimes
7	Hulk	Grey	No	M	2,050	mgrimes
8	Pat	White	No	F	1,400	mgrimes
9	Shamrock	Black	No	M	1,400	ssimpson

DROPping tables that are in relationships…

- We cannot drop the Customers table as long as the Horses table is referring to it:

```
DROP TABLE Customers;
```



SQL Error [2449] [72000]: ORA-02449: unique/primary keys in table referenced by foreign keys

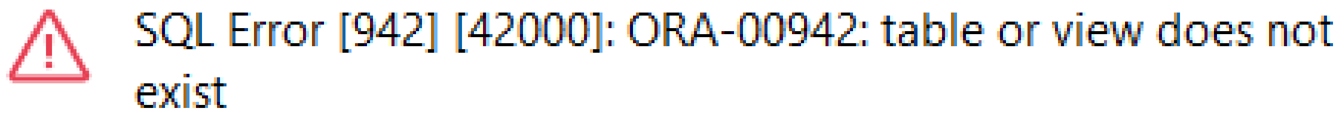
- … but we can drop horses FIRST (which will drop the Foreign Key constraint), then drop customers:

```
DROP TABLE Horses;
DROP TABLE Customers;
```

Similarly, we cannot CREATE a table that references a table that does not exist!

- If the Horses table references the Customers table, we cannot create the Horses table if the Customer table does not already exist!

```
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
);
```



SQL Error [942] [42000]: ORA-00942: table or view does not exist

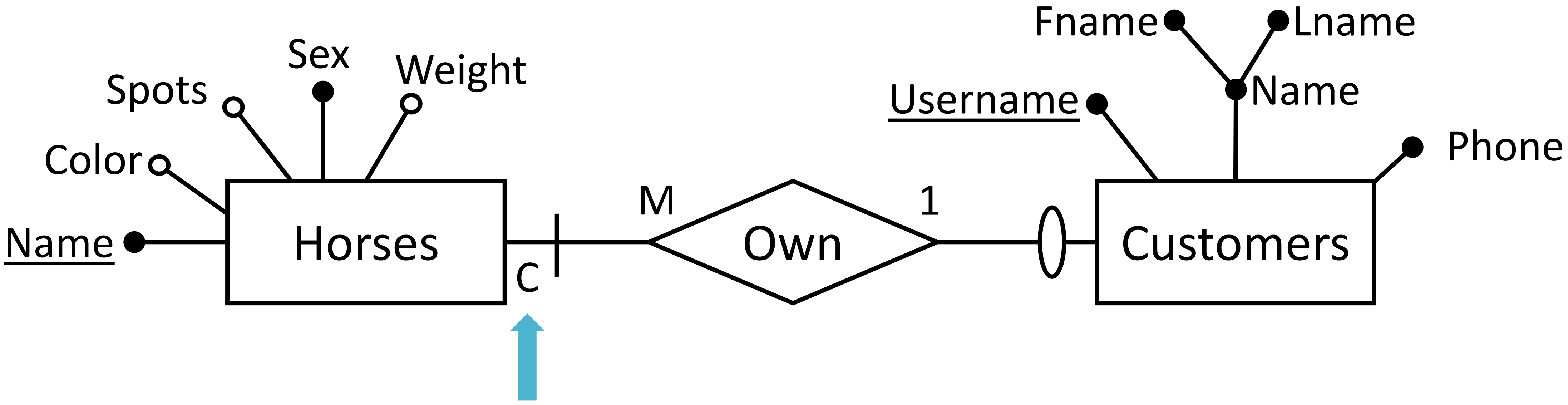
Our final DDL for the day:

```
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');
```

Final ERD for the day (with the CASCADE rule documented)



Semantic Integrity Constraints:

Color: {Black, White, Brown, Grey, Red, Yellow}, Default: 'UNK'

Spots: {Yes, No}, Default: 'UNK’

Sex: {M, F}

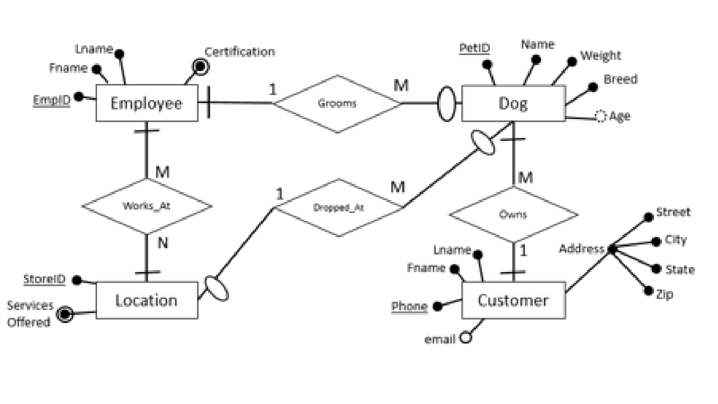
Weight: {800-2200}

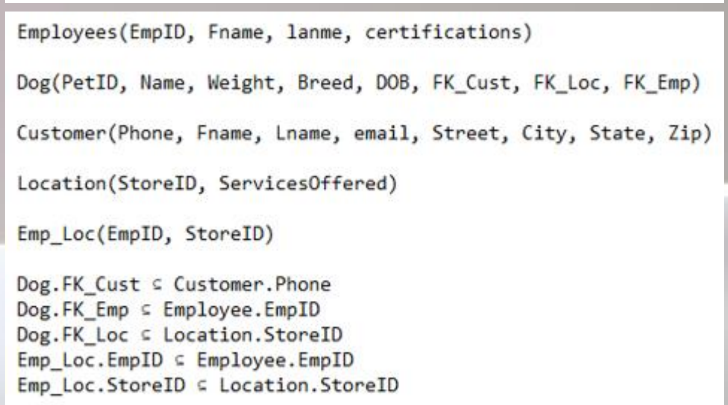
Module 10.2

Implementing Databases – Multiple Tables

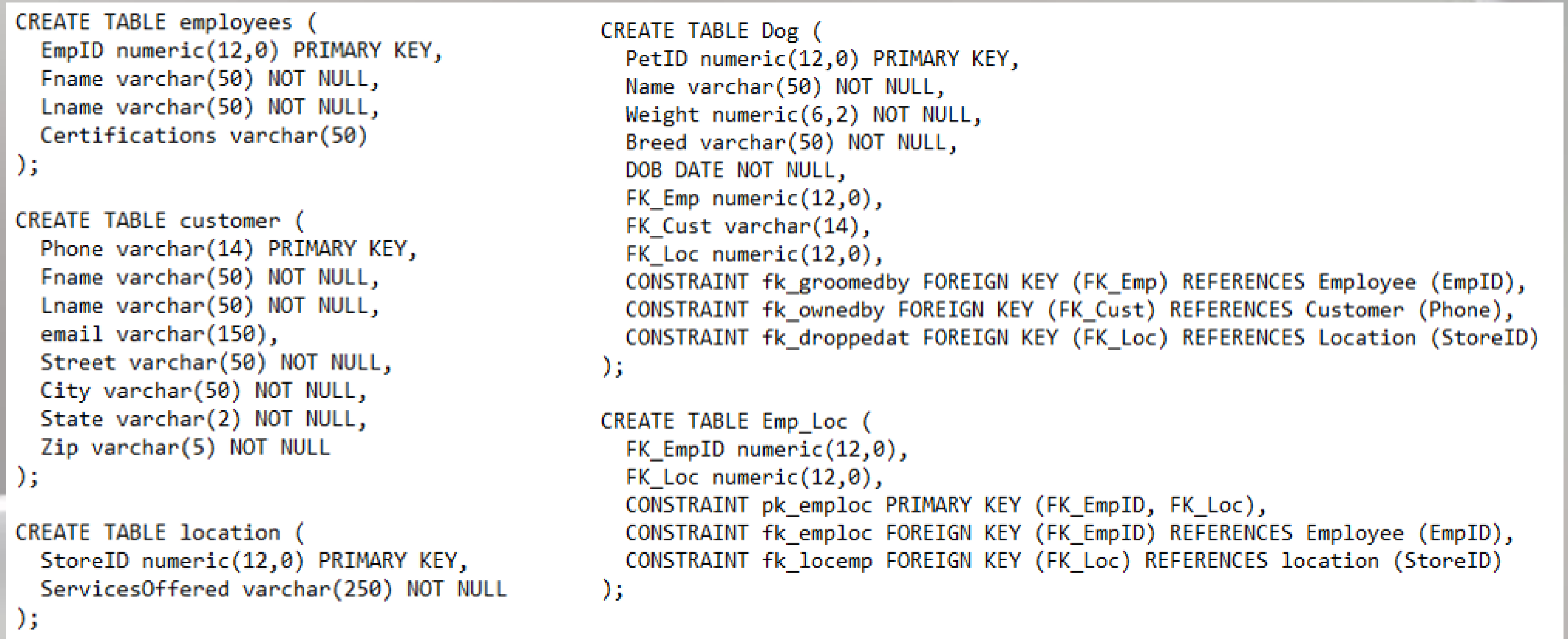
Next step in transforming business rules into an actual DB

ERD → Design Specific ERD → Logical Schema → Physical Schema





```
Employees(EmpID, Fname, lanme, certifications)

Dog(PetID, Name, Weight, Breed, DOB, FK_Cust, FK_Loc, FK_Emp)

Customer(Phone, Fname, Lname, email, Street, City, State, Zip)

Location(StoreID, ServicesOffered)

Emp_Loc(EmpID, StoreID)

Dog.FK_Cust ⊆ Customer.Phone
Dog.FK_Emp ⊆ Employee.EmpID
Dog.FK_Loc ⊆ Location.StoreID
Emp_Loc.EmpID ⊆ Employee.EmpID
Emp_Loc.StoreID ⊆ Location.StoreID
```



```
CREATE TABLE employees (
  EmpID numeric(12,0) PRIMARY KEY,
  Fname varchar(50) NOT NULL,
  Lname varchar(50) NOT NULL,
  Certifications varchar(50)
);

CREATE TABLE customer (
  Phone varchar(14) PRIMARY KEY,
  Fname varchar(50) NOT NULL,
  Lname varchar(50) NOT NULL,
  email varchar(150),
  Street varchar(50) NOT NULL,
  City varchar(50) NOT NULL,
  State varchar(2) NOT NULL,
  Zip varchar(5) NOT NULL
);

CREATE TABLE location (
  StoreID numeric(12,0) PRIMARY KEY,
  ServicesOffered varchar(250) NOT NULL
);
```

```
CREATE TABLE Dog (
  PetID numeric(12,0) PRIMARY KEY,
  Name varchar(50) NOT NULL,
  Weight numeric(6,2) NOT NULL,
  Breed varchar(50) NOT NULL,
  DOB DATE NOT NULL,
  FK_Emp numeric(12,0),
  FK_Cust varchar(14),
  FK_Loc numeric(12,0),
  CONSTRAINT fk_groomedby FOREIGN KEY (FK_Emp) REFERENCES Employee (EmpID),
  CONSTRAINT fk_ownedby FOREIGN KEY (FK_Cust) REFERENCES Customer (Phone),
  CONSTRAINT fk_droppedat FOREIGN KEY (FK_Loc) REFERENCES Location (StoreID)
);

CREATE TABLE Emp_Loc (
  FK_EmpID numeric(12,0),
  FK_Loc numeric(12,0),
  CONSTRAINT pk_emploc PRIMARY KEY (FK_EmpID, FK_Loc),
  CONSTRAINT fk_emploc FOREIGN KEY (FK_EmpID) REFERENCES Employee (EmpID),
  CONSTRAINT fk_locemp FOREIGN KEY (FK_Loc) REFERENCES location (StoreID)
);
```

Progress Quiz Tme

The progress is for this week is available on Canvas

- …but 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

- Remember to do the Progress Quiz on Canvas

- Assignment 1 is due by 6:00 on Monday, February 12

- Next class we will be applying this to the Bearcat Incorporated business case described in chapter 3 of the book

BZAN 6354

Lecture 3

February 5, 2024

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

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