BZAN 6354

Lecture 2

January 29, 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

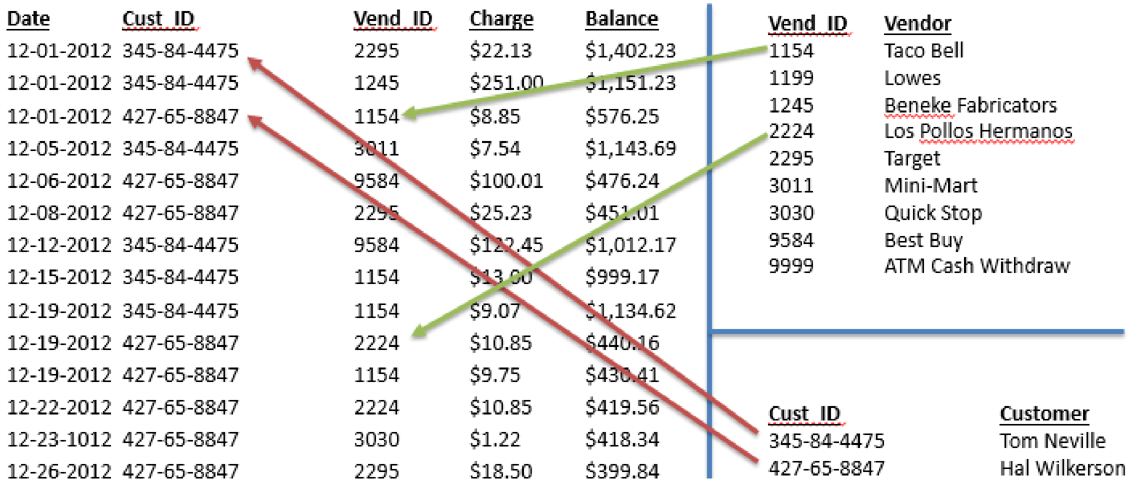
- Quick Review
- 1.5: Characteristics of Database Systems
- 1.6: Data Models
- 2.1: Conceptual data modeling
- 2.2: ER Grammar
- 2.3: Entities and attributes
- 10 minute break
- 10.1: Introduction to Data Definition Language (DDL) and Structured Query Language (SQL)

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

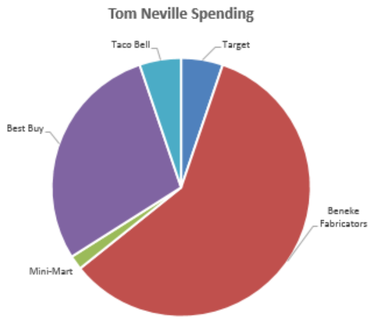


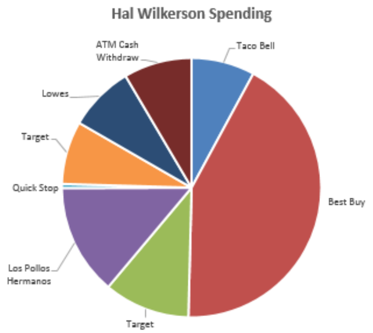
Date	Cust_ID	Vend_ID	Charge	Balance
12-01-2012	345-84-4475	2295	$22.13	$1,402.23
12-01-2012	345-84-4475	1245	$251.00	$1,151.23
12-01-2012	427-65-8847	1154	$8.85	$576.25
12-05-2012	345-84-4475	3011	$7.54	$1,143.69
12-06-2012	427-65-8847	9584	$100.01	$476.24
12-08-2012	427-65-8847	2295	$25.23	$451.01
12-12-2012	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

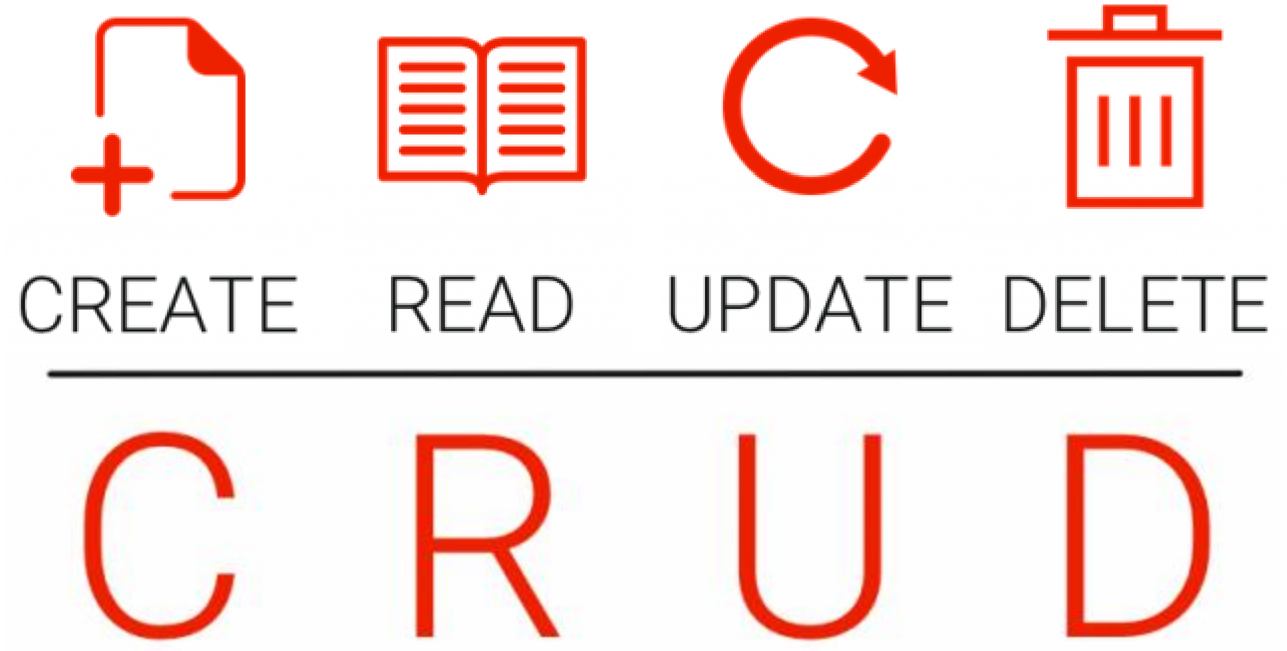
Knowledge = Information analyzed, visualized, etc. to help make decisions and predictions





Review: What are the four actions of data management?

- An unfortunate abbreviation:



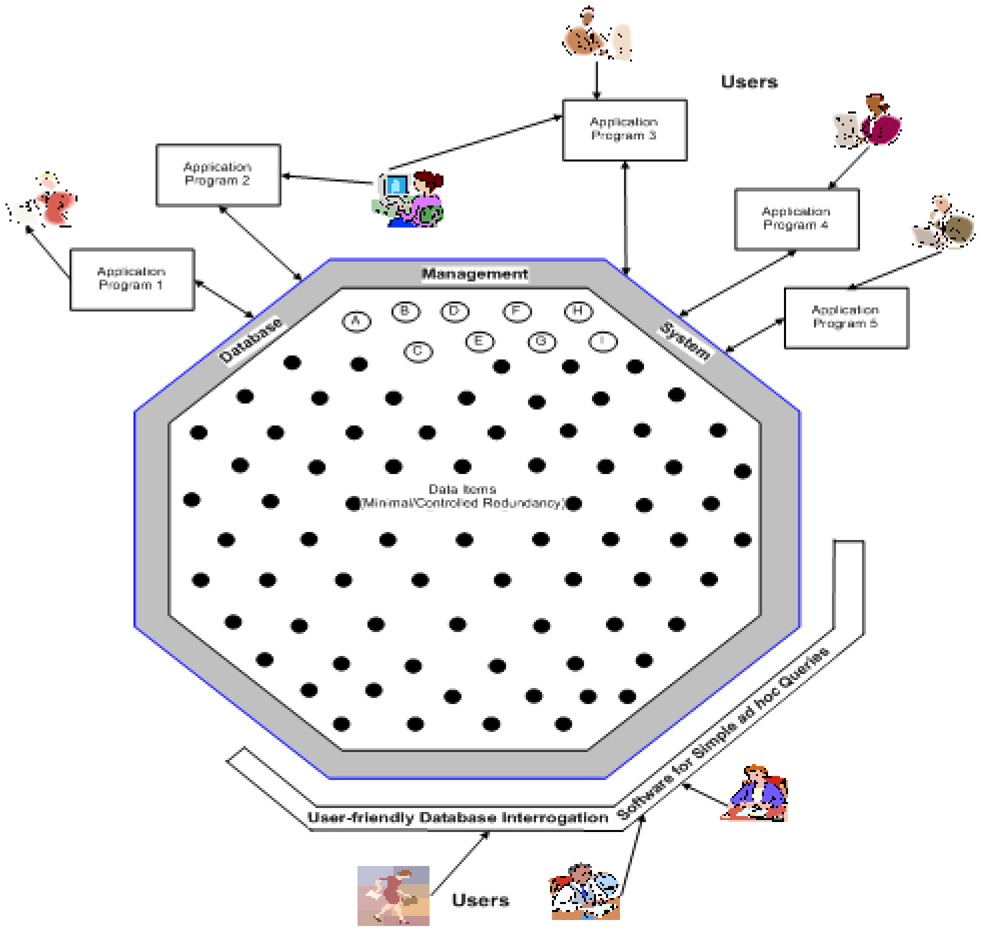
Review: What is a Database Management System

- 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 are the three levels of the ANSI/SPARC three schema architecture?

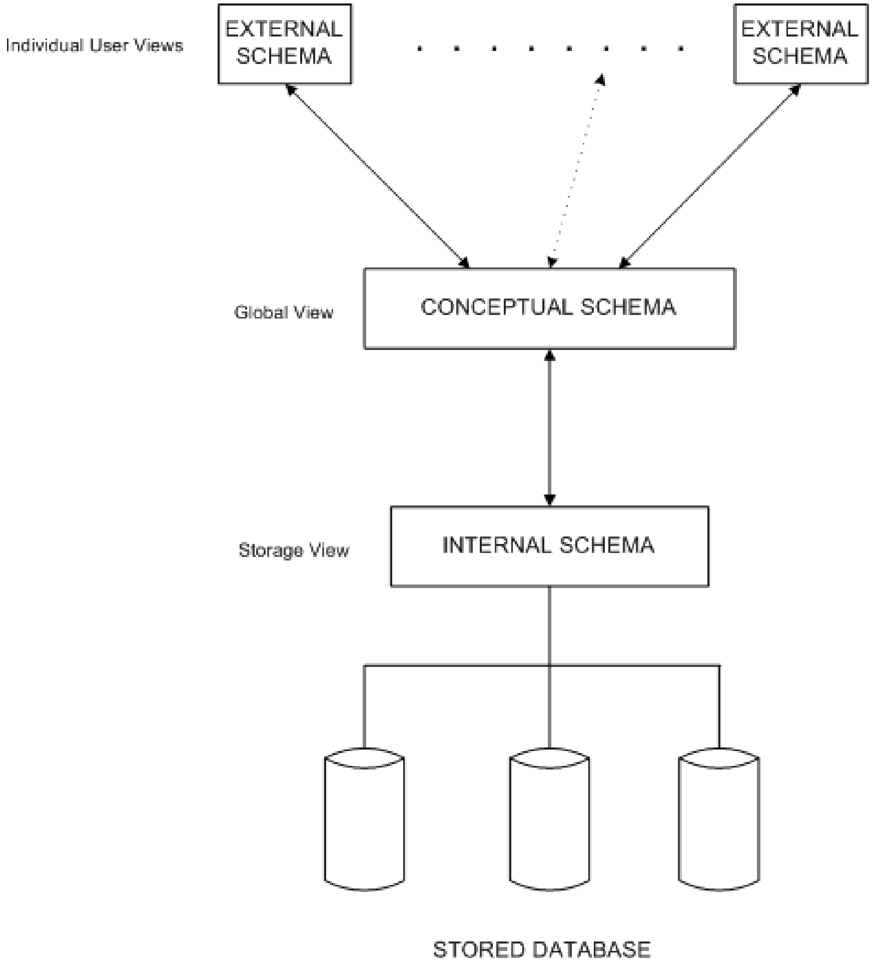
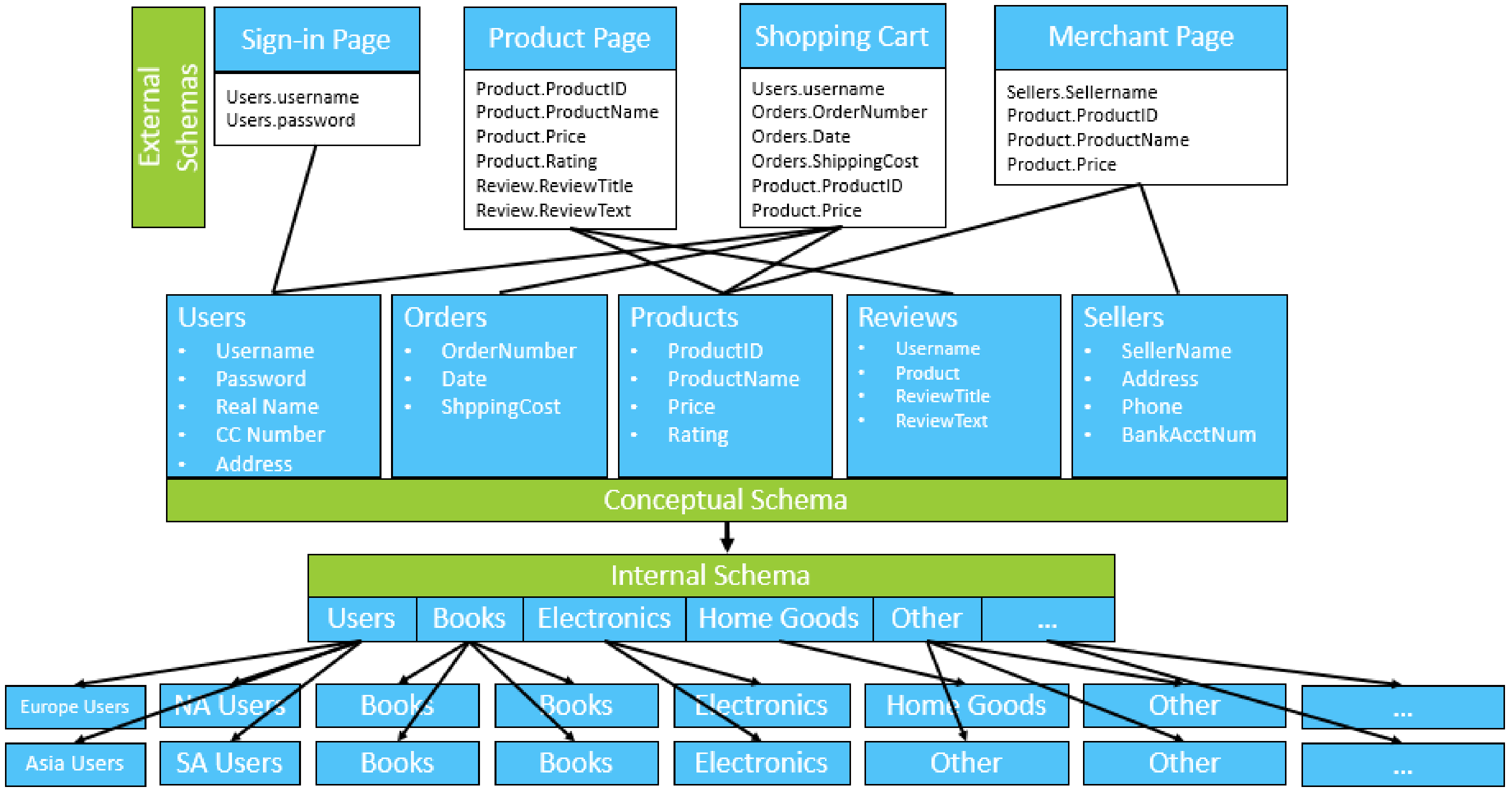


Figure 1.2 The ANSI/SPARC three-schema Architecture

Review: What is a schema?

A map of your data



External Schemas

Sign-in Page
Users.username
Users.password

Product Page
Product.ProductID
Product.ProductName
Product.Price
Product.Rating
Review.ReviewTitle
Review.ReviewText

Shopping Cart
Users.username
Orders.OrderNumber
Orders.Date
Orders.ShippingCost
Product.ProductID
Product.Price

Merchant Page
Sellers.Sellername
Product.ProductID
Product.ProductName
Product.Price

Users
- Username
- Password
- Real Name
- CC Number
- Address

Orders
- OrderNumber
- Date
- ShppingCost

Products
- ProductID
- ProductName
- Price
- Rating

Reviews
- Username
- Product
- ReviewTitle
- ReviewText

Sellers
- SellerName
- Address
- Phone
- BankAcctNum

Conceptual Schema

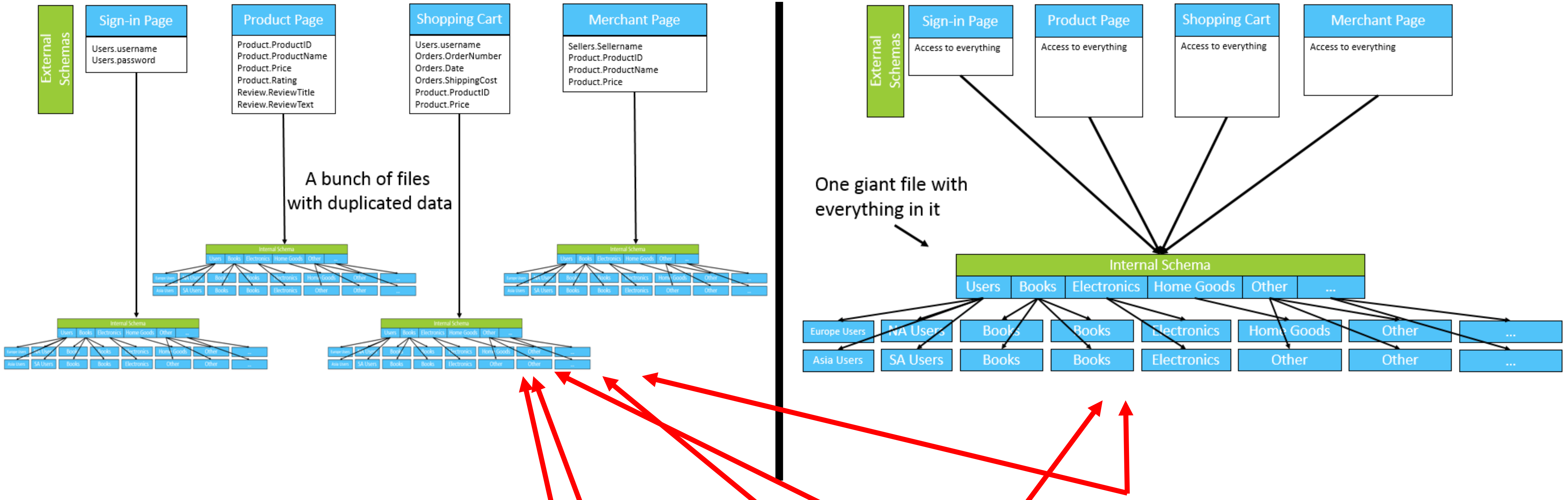
Internal Schema

Users | Books | Electronics | Home Goods | Other | ...

Europe Users | NA Users | Books | Books | Electronics | Home Goods | Other | ...

Asia Users | SA Users | Books | Books | Electronics | Other | Other | ...

Review: Describe these limitations and problems of a “two schema” file system architecture



External Schemas

Sign-in Page
Users.username
Users.password

Product Page
Product.ProductID
Product.ProductName
Product.Price
Product.Rating
Review.ReviewTitle
Review.ReviewText

Shopping Cart
Users.username
Orders.OrderNumber
Orders.Date
Orders.ShippingCost
Product.ProductID
Product.Price

Merchant Page
Sellers.Sellername
Product.ProductID
Product.ProductName
Product.Price

A bunch of files with duplicated data

External Schemas

Sign-in Page
Access to everything

Product Page
Access to everything

Shopping Cart
Access to everything

Merchant Page
Access to everything

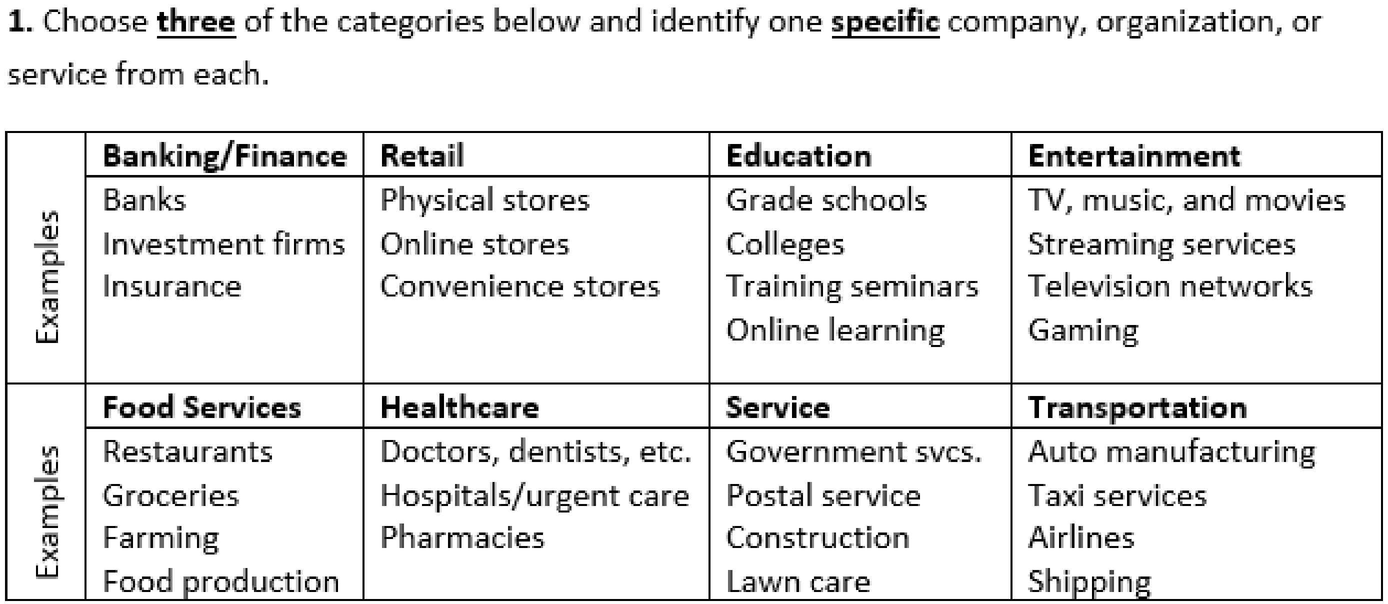
One giant file with everything in it

Limitations: Lack of data integrity, Lack of standards, Lack of flexibility/maintainability

Problems: Lack of integration, Lack of program-data independence

Assignment 1

- Posted to Canvas - Due on Monday, February 5 at 6:00 PM



1. Choose **three** of the categories below and identify one **specific** company, organization, or service from each.

	Banking/Finance	**Retail**	**Education**	**Entertainment**
Examples	Banks Investment firms Insurance	Physical stores Online stores Convenience stores	Grade schools Colleges Training seminars Online learning	TV, music, and movies Streaming services Television networks Gaming
	Food Services	**Healthcare**	**Service**	**Transportation**
Examples	Restaurants Groceries Farming Food production	Doctors, dentists, etc. Hospitals/urgent care Pharmacies	Government svcs. Postal service Construction Lawn care	Auto manufacturing Taxi services Airlines Shipping

- 2 entities, 5 attributes each, and 3-7 instances each.
- Data dictionary, ERD showing relationship, cardinality and participation
- Business rules

Module 1.5

Characteristics of Database Systems

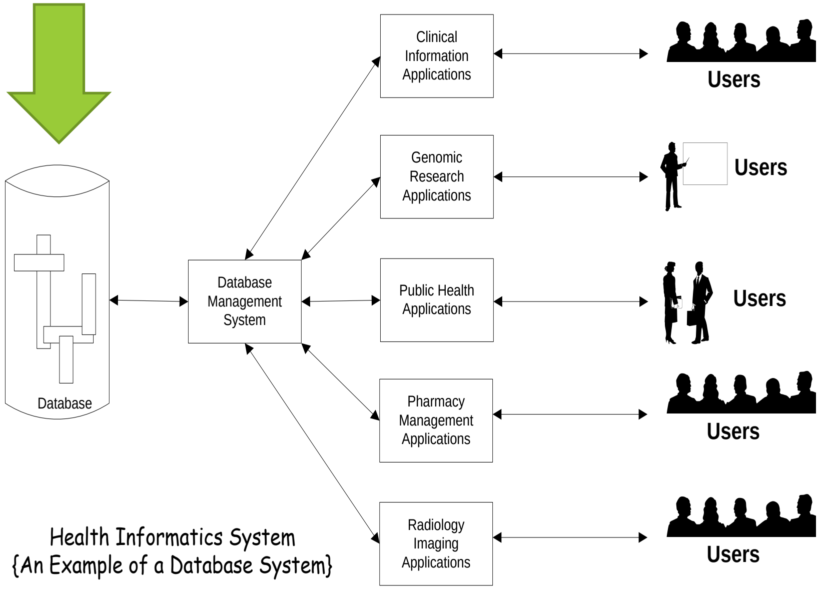
- What is the difference in the “database” and the “DBMS”?

- Why is it important that a database is “self describing”?

What is a database?

- Database systems were created to overcome the limitations of the old “file system” way of doing things

- Database: An integrated set of files
  - We still use files – but the DBMS is a system for managing the files and data contained within


Database
Database Management System
Clinical Information Applications
Genomic Research Applications
Public Health Applications
Pharmacy Management Applications
Radiology Imaging Applications
Users
Users
Users
Users
Users

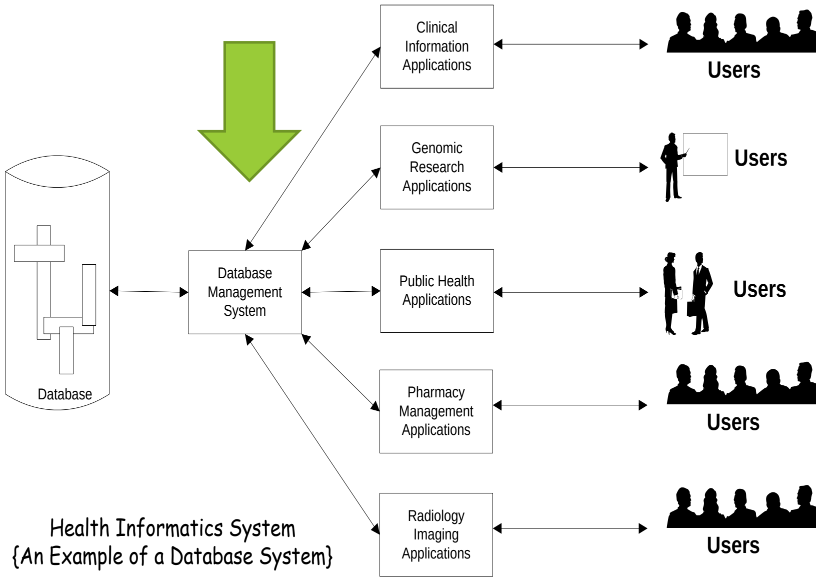
Health Informatics System
{An Example of a Database System}

What is a database?

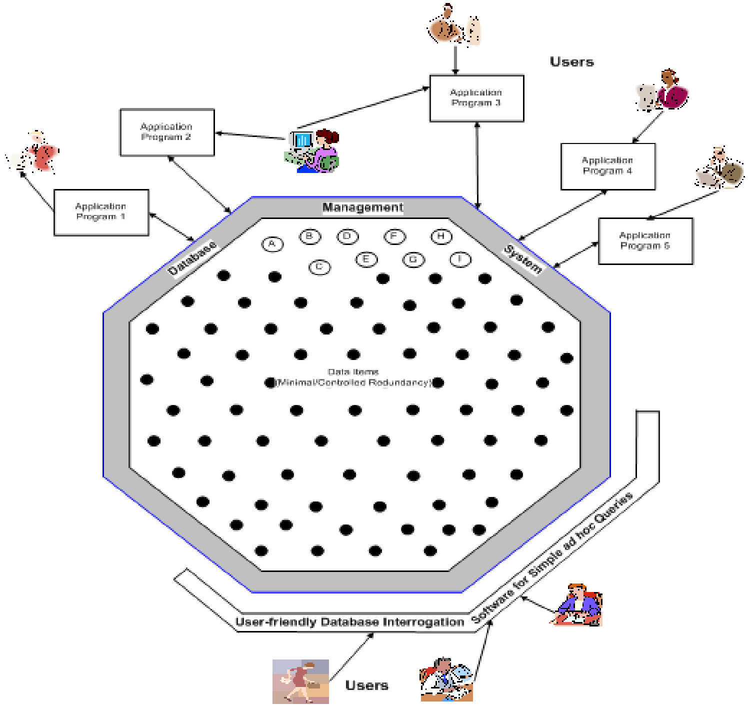
- A **Database** includes data and metadata
  - A database is **self-describing** in that the metadata is recorded within the database **(i.e., the schemas)**, <u>not in application programs</u>.
  - Data consists of **recorded facts** that have implicit meaning.
  - Viewed through the lens of **metadata**, the meaning of data becomes explicit.
  - A database is a collection of files whose records are logically related to one another.
  - **Integration of data is the responsibility of the DBMS instead of the programmer.**

What is a database management system?

- DBMS: A collection of tools (software) that facilitate the process of defining, constructing, and manipulating data in a database.
- Rather than interacting with the data directly, the DMBS provides users and applications a method for "asking" for the data


Database
Database Management System
Clinical Information Applications
Genomic Research Applications
Public Health Applications
Pharmacy Management Applications
Radiology Imaging Applications
Users
Users
Users
Users
Users

Health Informatics System
{An Example of a Database System}



Components of a DBMS

- A **data dictionary**
  - The metadata about your data
- One or more **query languages** (i.e., SQL)
- A **data manipulation language** (SQL, PL/SQL) for accessing the database
- A **data definition language** (SQL) to define the structure of data
- Tools for generating **reports**
- DBMS **utilities**
  - User security, importing data, data conversion, backup/restore, performance monitoring, reorganizing/indexing data,

Module 1.5
Characteristics of Database Systems

- What is the difference in the “database” and the “DBMS”?
- Why is it important that a database is “self describing”?

Module 1.6

Data Models

- What is a data model?

- What are the differences in conceptual, logical, and physical data models? Who is the intended audience for each

What is a model?

- 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!

What is a model?

- Simplified expression of observed or unobservable reality used to perceive relationships in the outside world.
  - A model is an approximation & entails assumptions

- Examples:
  - Model aircraft in wind tunnel testing
  - Mathematical models (econometric, optimization, etc.)
  - Computing models (e.g., analog models/directed graphs)
  - Data models, Process models, etc.

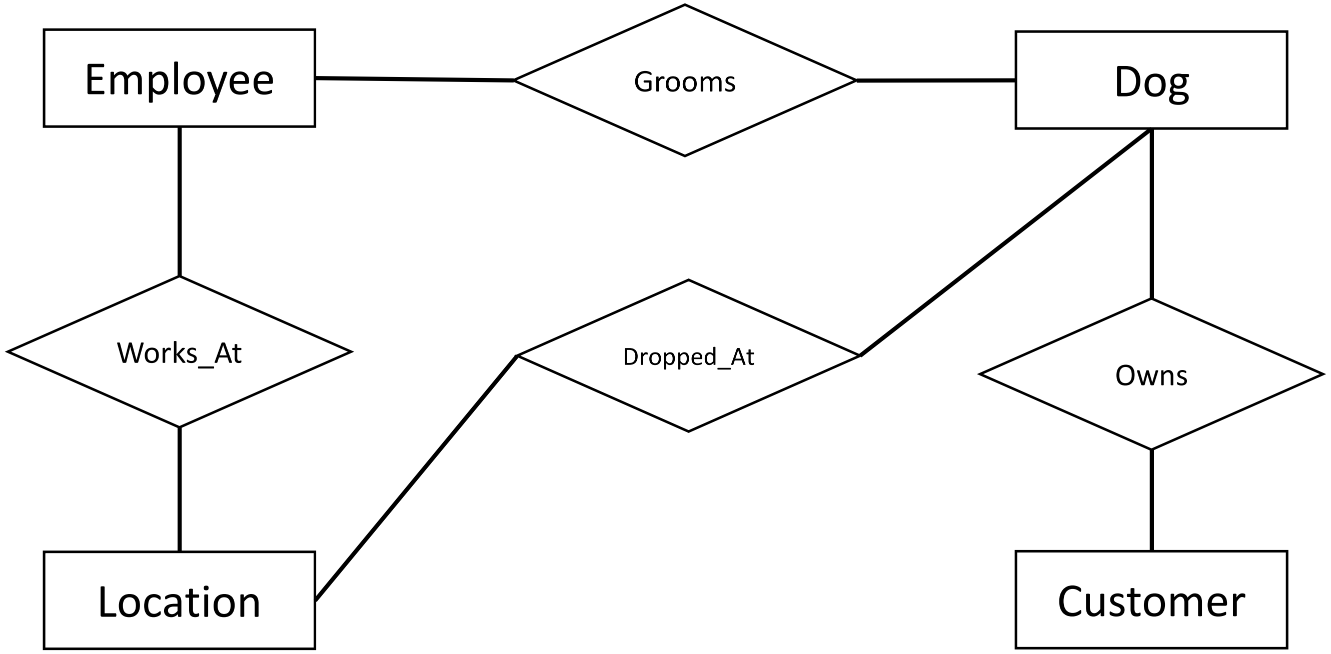
- A blue print for designing databases

Data modeling stages

- Conceptual modeling
  - Product: Conceptual schema
- Logical model/design
  - Product: Logical schema
- Physical design
  - Product: Physical/Internal schema

Conceptual modeling: Dog grooming service

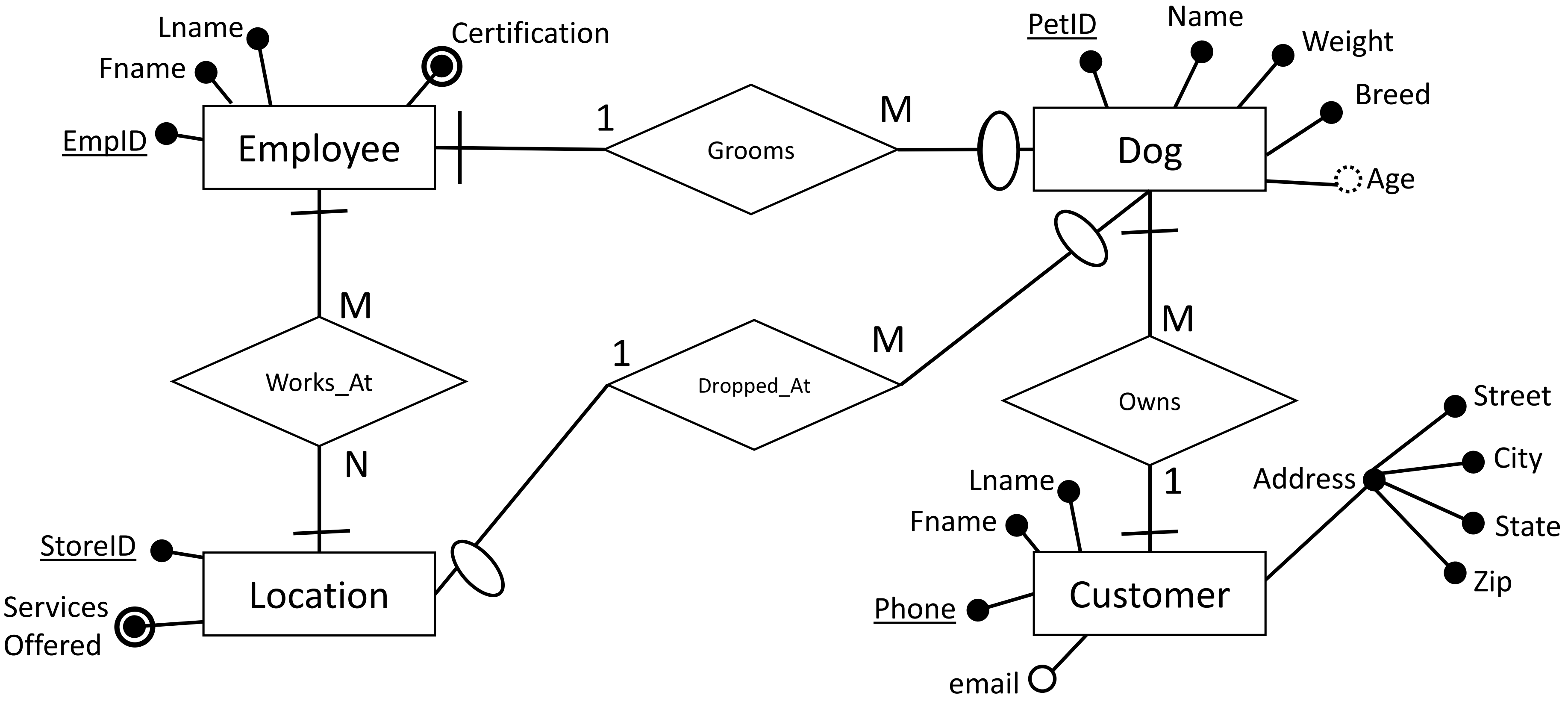
Customers of Dave's Dog Wash (DDW) own dogs. Employees of DDW groom dogs. There are multiple Locations of DDW that employees can work at. Dogs can be dropped off at any location.



Conceptual modeling: Dog grooming service

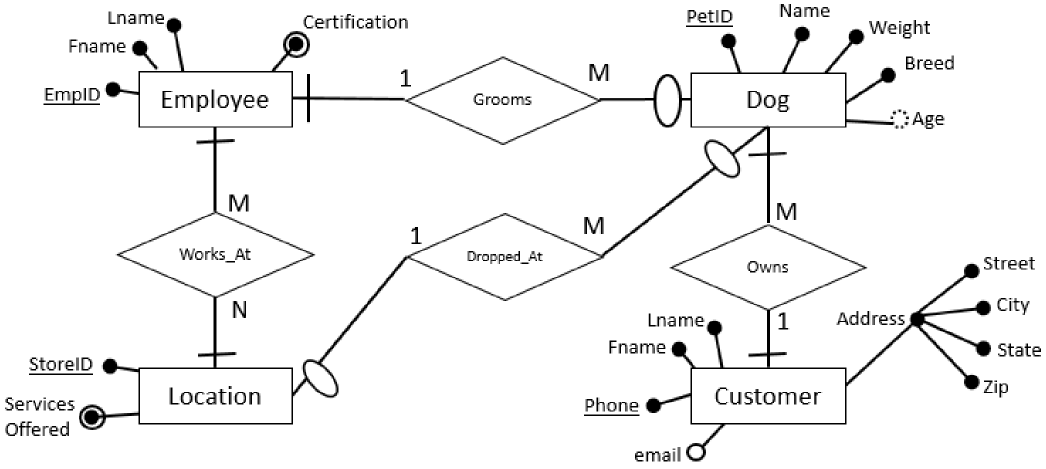
Customers of Dave’s Dog Wash (DDW) own dogs. Employees of DDW groom dogs. There are multiple Locations of DDW that employees can work at. Dogs can be dropped off at any location.

Customers, Dogs, Employees, and Locations also have **attributes** that describe them!



Logical Schema

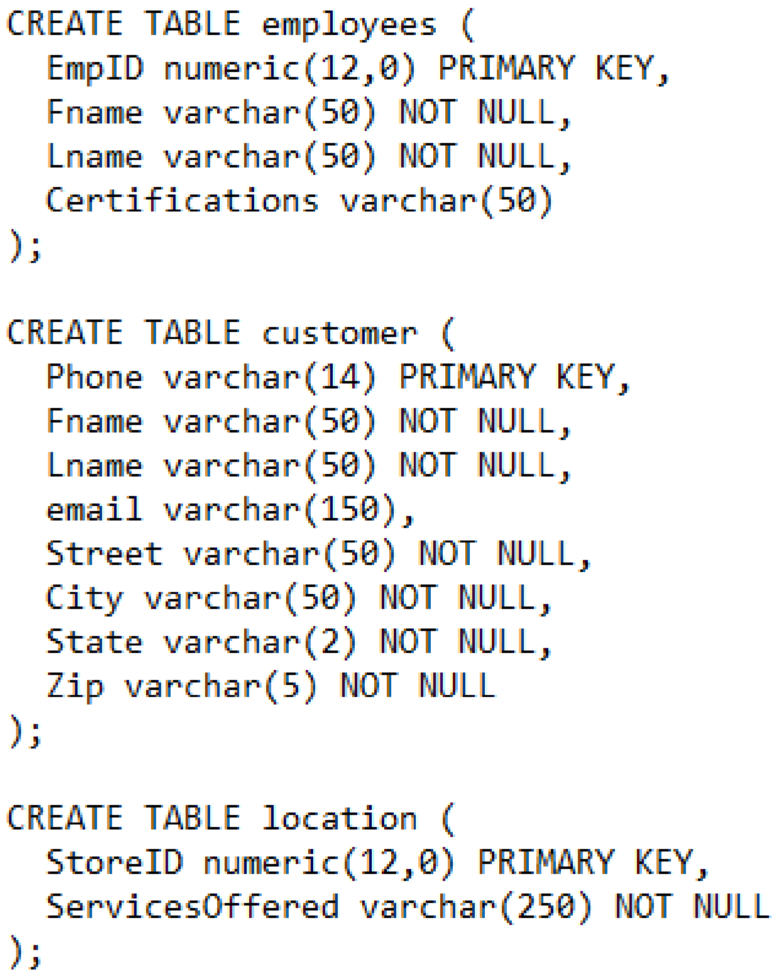
Conceptual schema



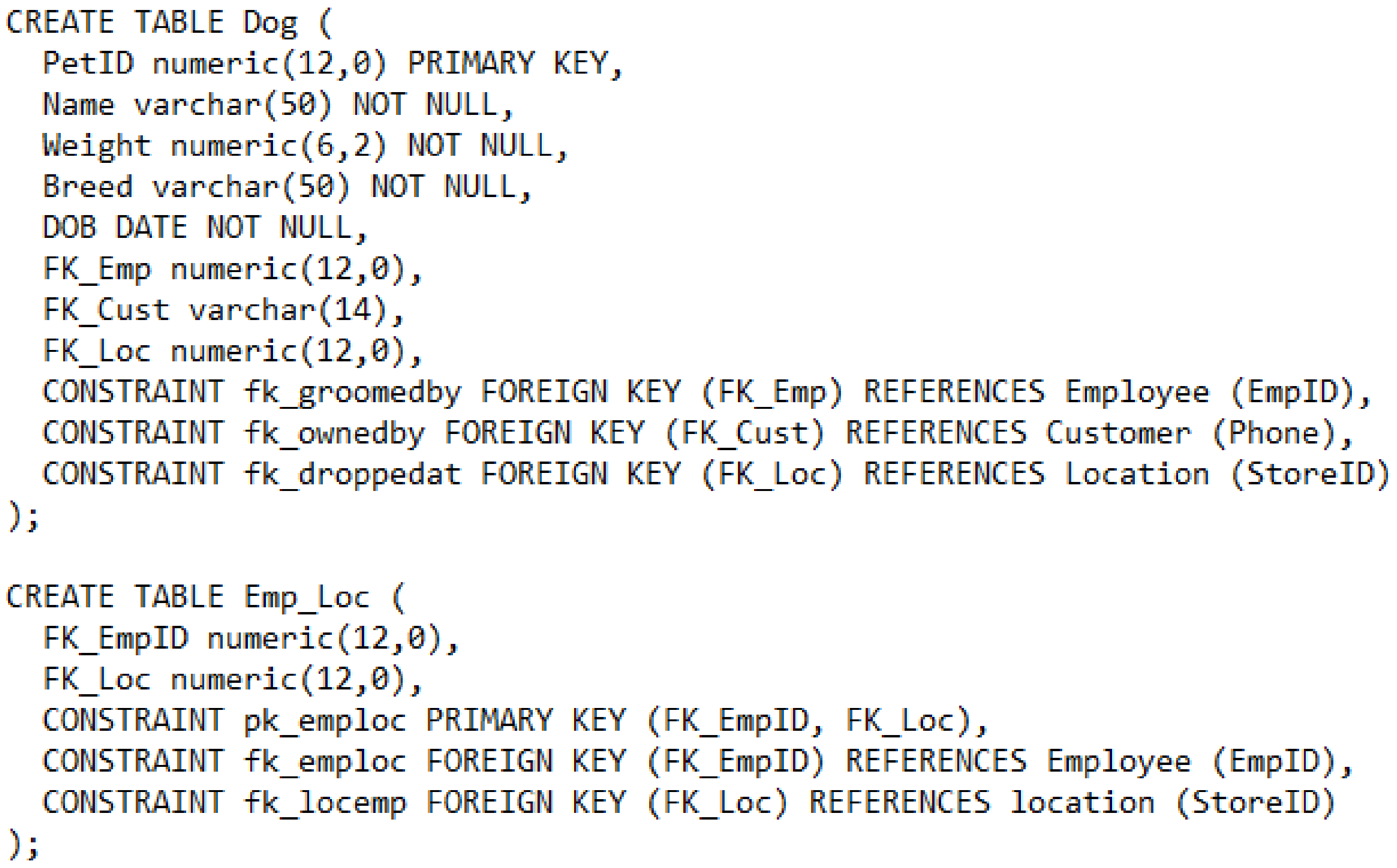
Logical schema

```
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
```

SQL Code to create the physical schema



```
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)
);
```

Module 1.6
Data Models

- What is a data model?

- What are the differences in conceptual, logical, and physical data models? Who is the intended audience for each

Module 2.1

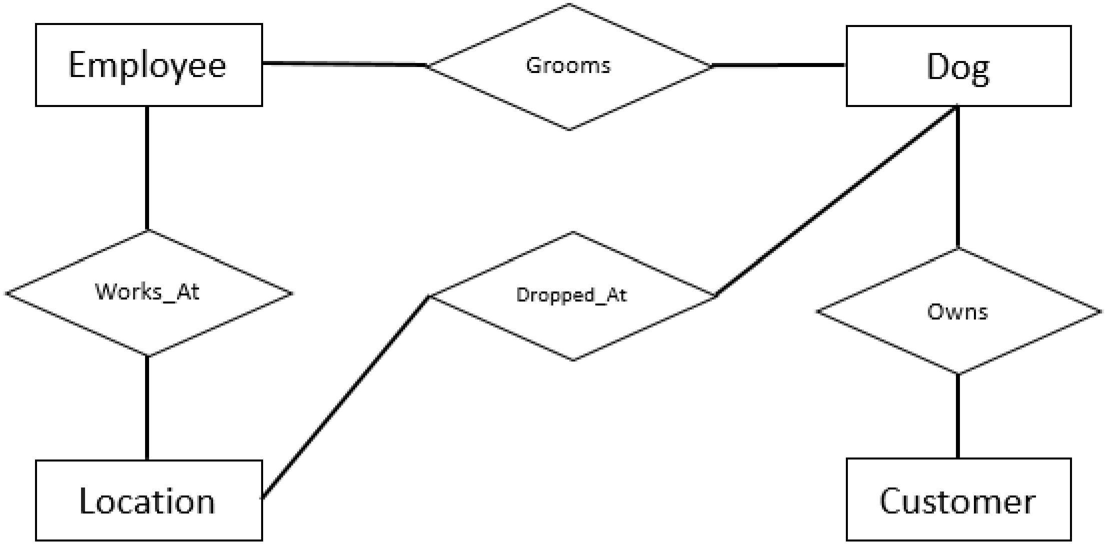
Conceptual Data Modeling Framework

- What is an entity relationship (ER) model?

- Who uses an ER model and why?

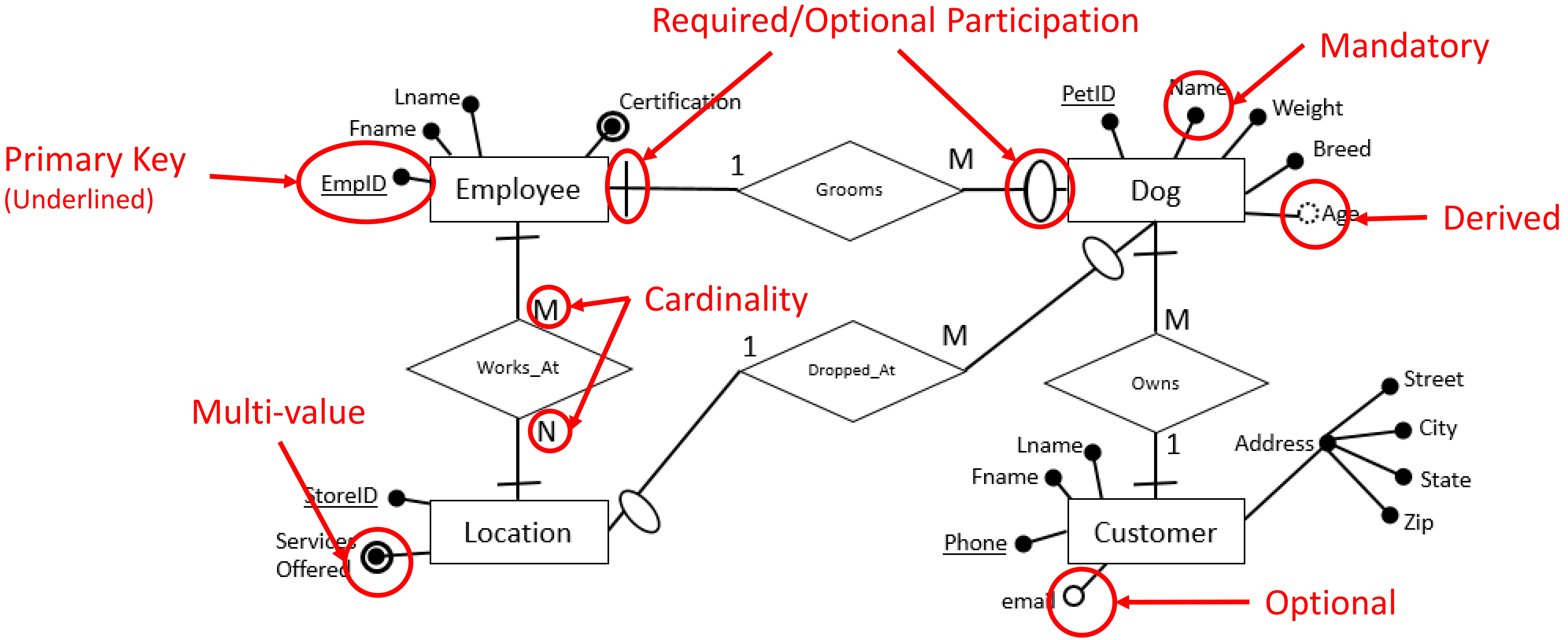
Let’s get precise

- Up to this point, we have talked about data modeling and relationships using pretty imprecise terms



Let's get precise

- By using specific symbols and notations (grammar), we can accurately represent business rules in our diagrams.



Entity-Relationship (ER) Model

- **Modeling grammar** for conceptual data modeling originally proposed by Peter Chen in 1976
  - The most widely accepted data modeling grammar for conceptual design
  - Chosen by ANSI in 1988 as the standard model for Information Resource Directory Systems (IRDSs)

- ER modeling grammar obeys the properties of a semantic data modeling technique:
  - Expressiveness
  - Simplicity
  - Minimality
  - Unique interpretation
  - Formality

ER model: The Purpose

- Communication/presentation device used by an analyst to interact with the **end-user community**

- A design tool at the highest level of abstraction to convey a **deeper level understanding to the database designer**

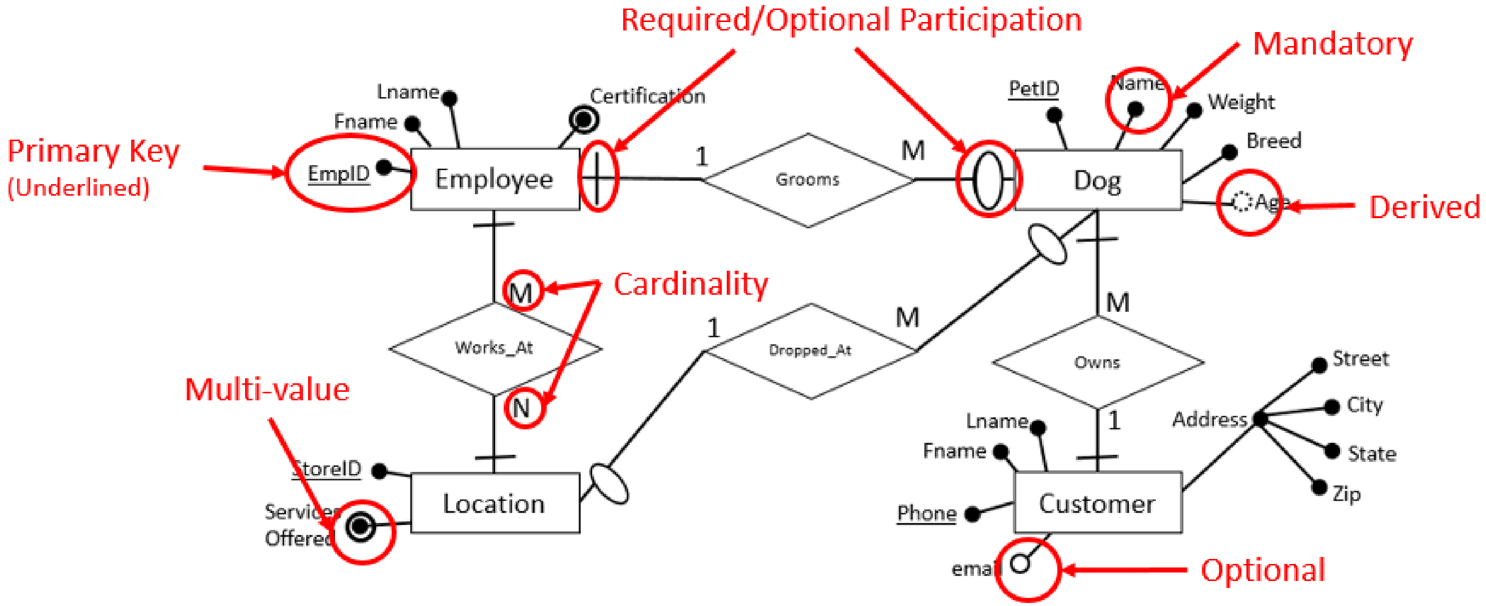
ER model: The Purpose

- From the users: What are the business rules?

- To the techies: How should the business rules be implemented in the database?

The ER Model includes

- An ER diagram that portrays **entity types**, **attributes**, and **relationships** among entity types



- Semantic integrity constraints that reflect the **business rules** about data not captured in the ER diagram
  - The things circled in red
  - Sometimes cannot represent graphically, so we write them out

What is a business rule?

- **A statement of a specific condition or procedure** relevant to the universe of interest (application domain) being modeled

- Business rules may be explicitly stated, but are often **implied** in the requirements specification and must be inferred
  - Problem: People often don’t mean exactly what they explicitly say
  - Problem: When you infer things you may get them wrong

- The process of developing business rules from the requirements specification is not quite scientific, but it can be systematic.
  - Go through the spec step-by-step
  - An iterative process

- Systematic analysis will also facilitate identification of ambiguities which, when clarified by the user community, will yield additional business rules and also facilitate correction of other business rules

Enforcing business rules

- Business rules might be defined in the application, the database, or both!

Why should we define the constraints in the database?

Hint:

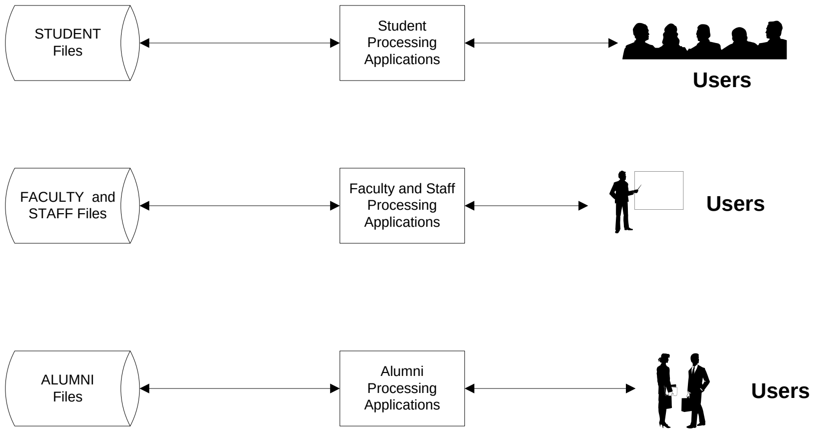

STUDENT Files
Student Processing Applications
Users
FACULTY and STAFF Files
Faculty and Staff Processing Applications
Users
ALUMNI Files
Alumni Processing Applications
Users

Figure 1.1 An example of a file processing environment

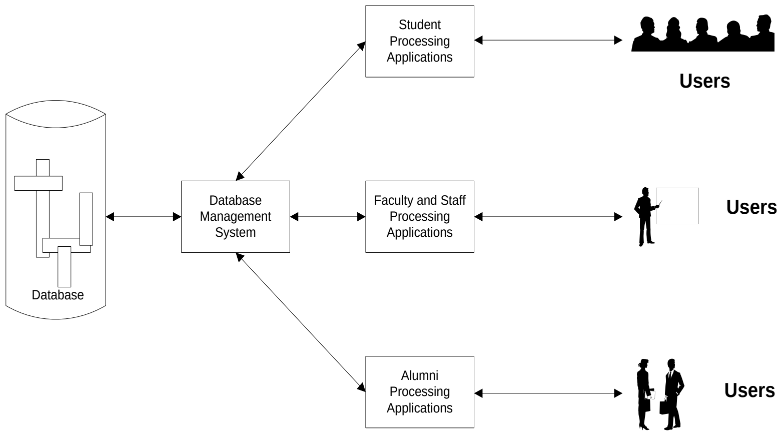

Database
Database Management System
Student Processing Applications
Users
Faculty and Staff Processing Applications
Users
Alumni Processing Applications
Users

Figure 1.6 An example of a database system

Module 2.1

Conceptual Data Modeling Framework

- What is an entity relationship (ER) model?
- Who uses an ER model and why?

Module 2.2
ER Grammar

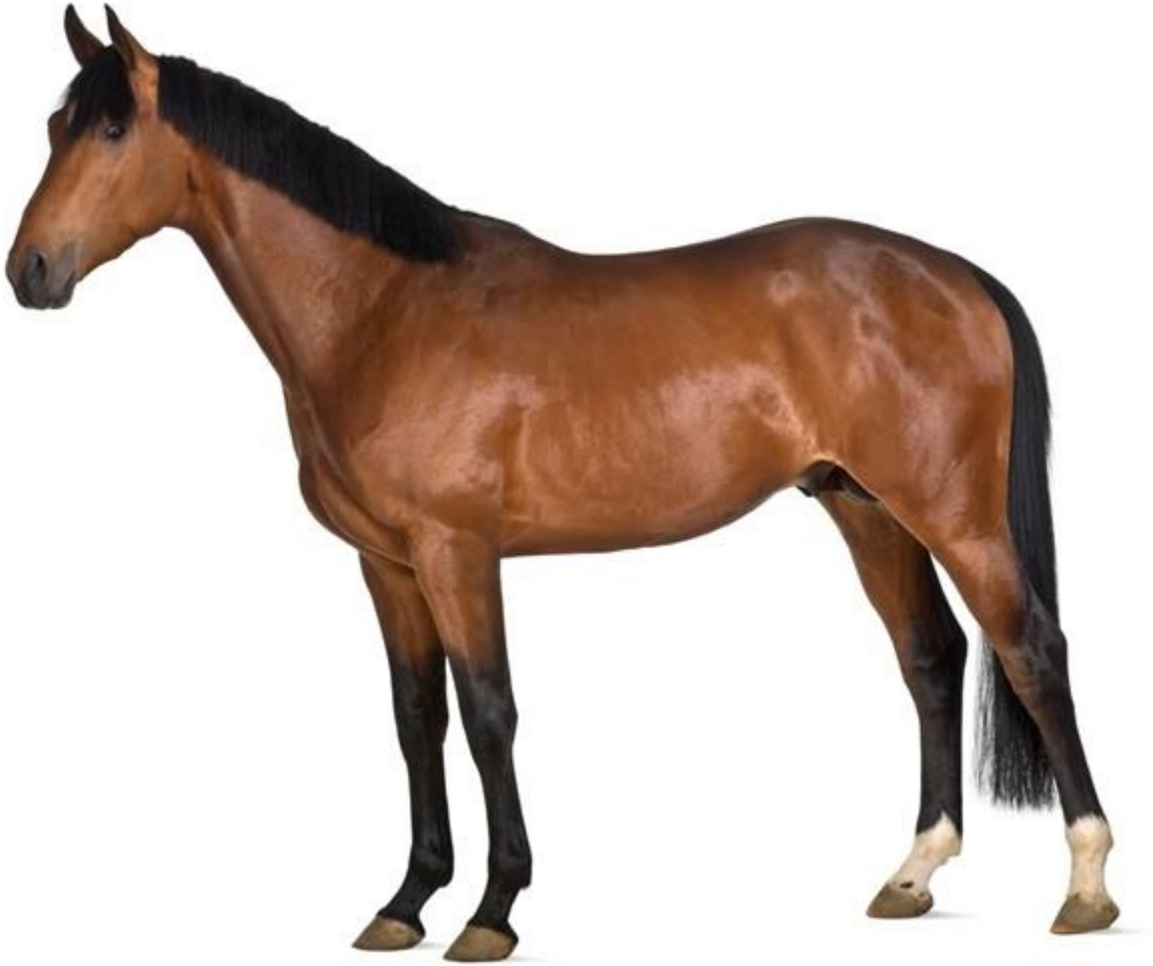
- At the end of this discussion, you should be able to describe the following:
  - Entity Type
  - Entity Instance
  - Entity Class
  - Attribute
  - Value
  - Domain
  - Relationship

ER Modeling primitives

- We’ve said entities are made up of collections of attributes

- There is more nuance to this:
  - Entity Type - Conceptual representation of an object
  - Entity Instance - An occurrence of an entity type
  - Entity Class - A set of entity types that have shared properties
  - Attribute - Describes an entity
  - Value – Data value of an attribute for a specific instance
  - Domain – Possible values for an attributes
  - Relationship – How an entity type is associated with other entity types

Exercise

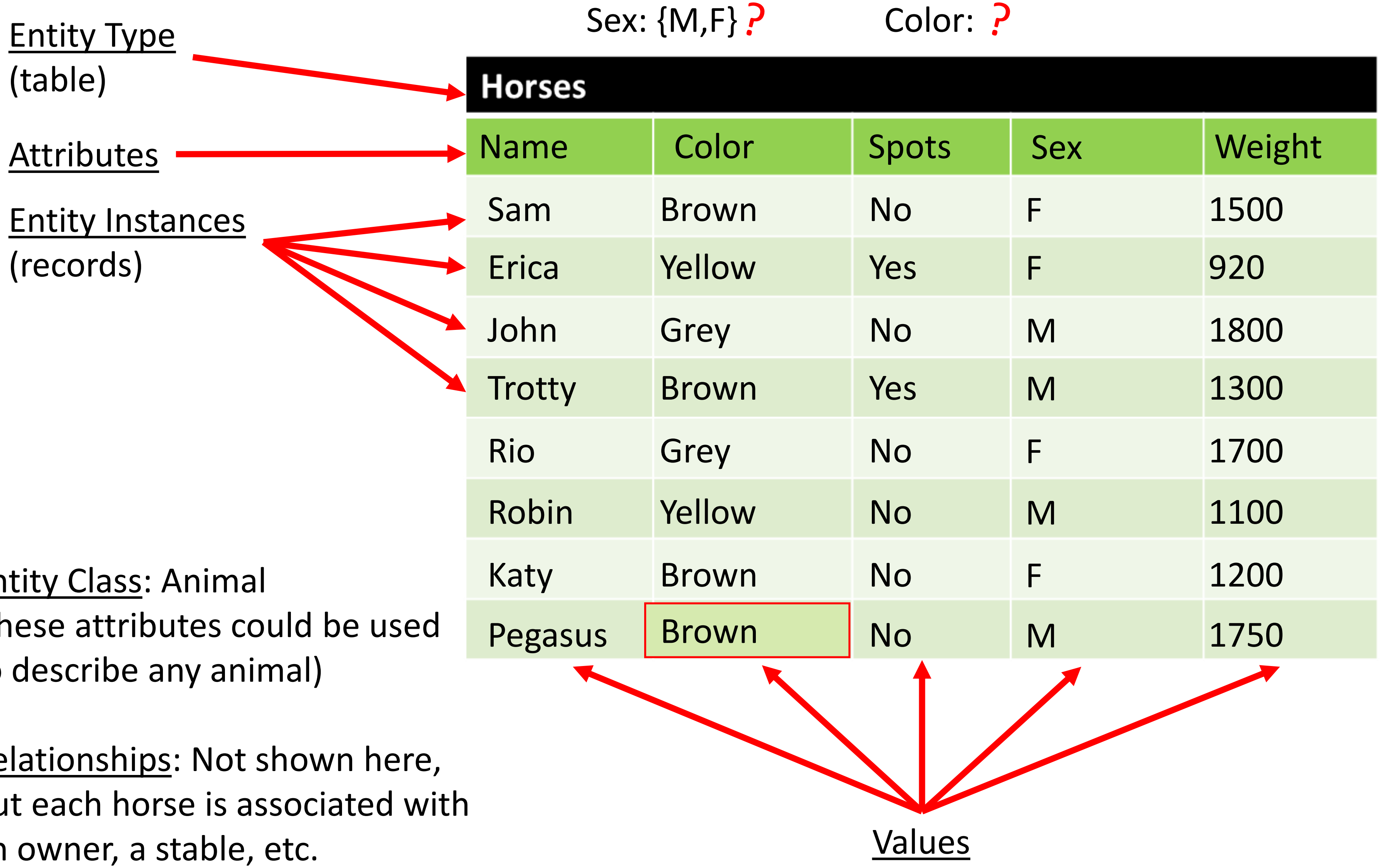


Data Model

Domains

Spots: {Yes, No} Weight: {800-2200}

Sex: {M,F} ? Color: ?



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

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.

Data Dictionary

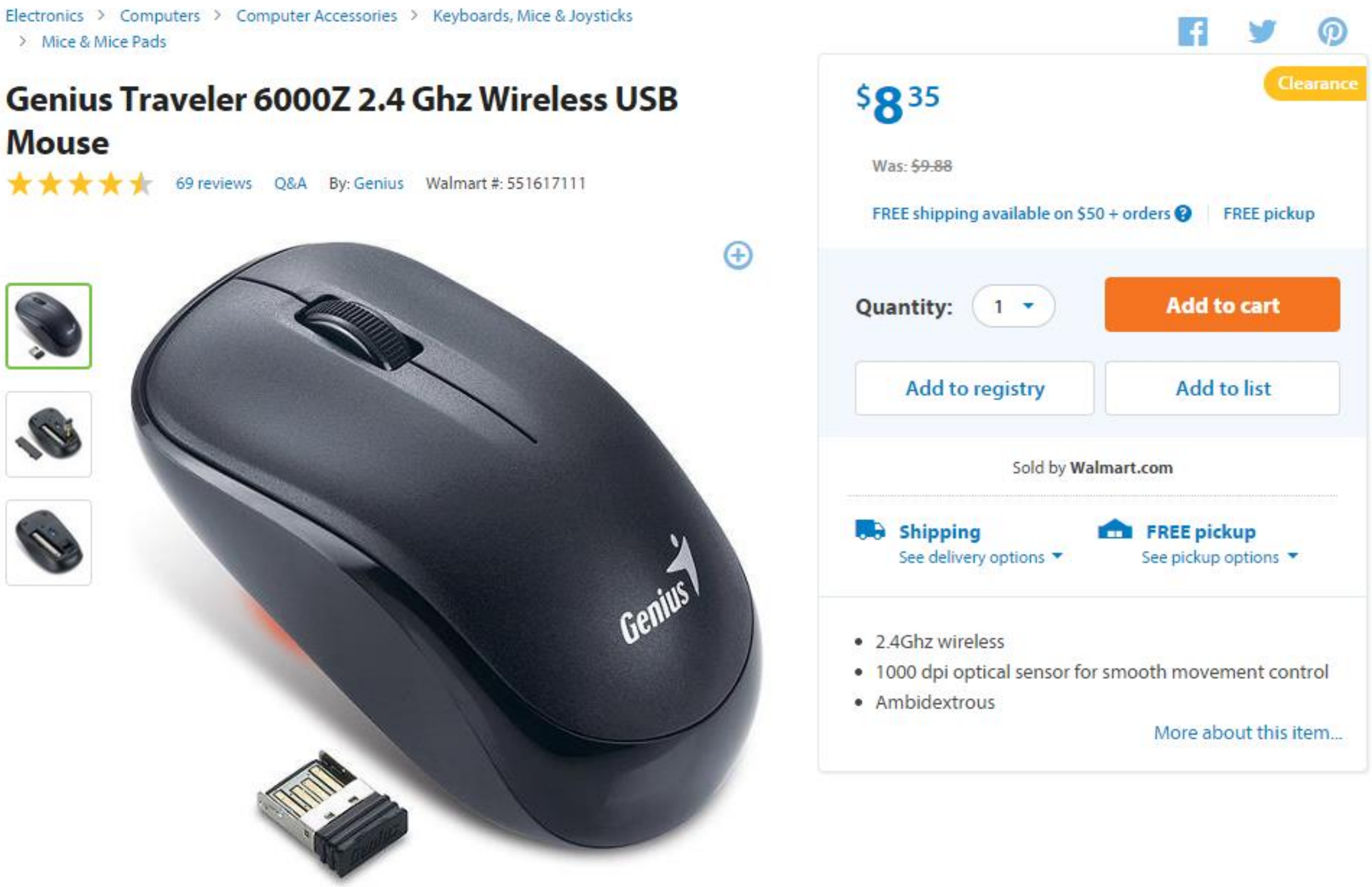
Table	Columns	Rows	Primary Key
Horses	5	800	HorseName
Owners	3	500	OwnerID
Stables	3	10	StableName

Metadata describing
The tables in the database (Schema)

Metadata describing the
Data in the tables

Column	Table	Data Type	Length
HorseName	Horses	Text	20
Color	Horses	Text	10
Spots	Horses	Binary	1
Sex	Horses	Text	1
Weight	Horses	Numeric	8
OwnerID	Owners	Numeric	8
Name	Owners	Text	40
Phone	Owners	Text	12
StableName	Stables	Text	50
Address	Stables	Text	50
Phone	Stables	Text	12

E-Commerce data – What attributes do you see here?



Electronics > Computers > Computer Accessories > Keyboards, Mice & Joysticks > Mice & Mice Pads

Genius Traveler 6000Z 2.4 Ghz Wireless USB Mouse

69 reviews Q&A By: Genius Walmart #: 551617111

$8.35 Clearance

Was: ~~$9.88~~

FREE shipping available on $50 + orders | FREE pickup

Quantity: 1

Sold by **Walmart.com**

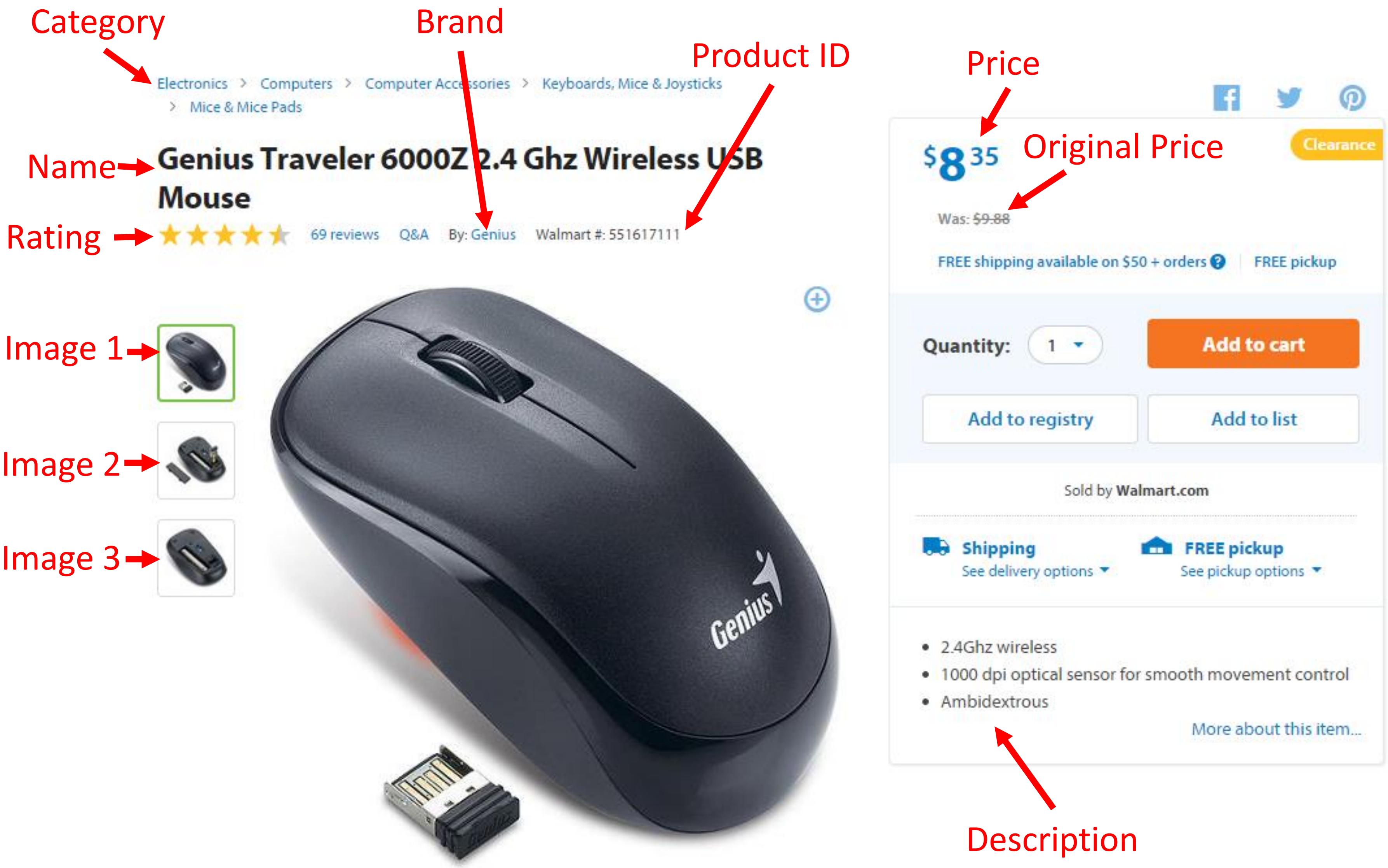
Shipping See delivery options

FREE pickup See pickup options

- 2.4Ghz wireless
- 1000 dpi optical sensor for smooth movement control
- Ambidextrous

More about this item...

E-Commerce data – What attributes do you see here?



Electronics > Computers > Computer Accessories > Keyboards, Mice & Joysticks > Mice & Mice Pads

Genius Traveler 6000Z 2.4 Ghz Wireless USB Mouse

★★★★½ 69 reviews Q&A By: Genius Walmart #: 551617111

$8.35 Clearance

Was: ~~$9.88~~

FREE shipping available on $50 + orders | FREE pickup

Quantity: 1 | Add to cart

Add to registry | Add to list

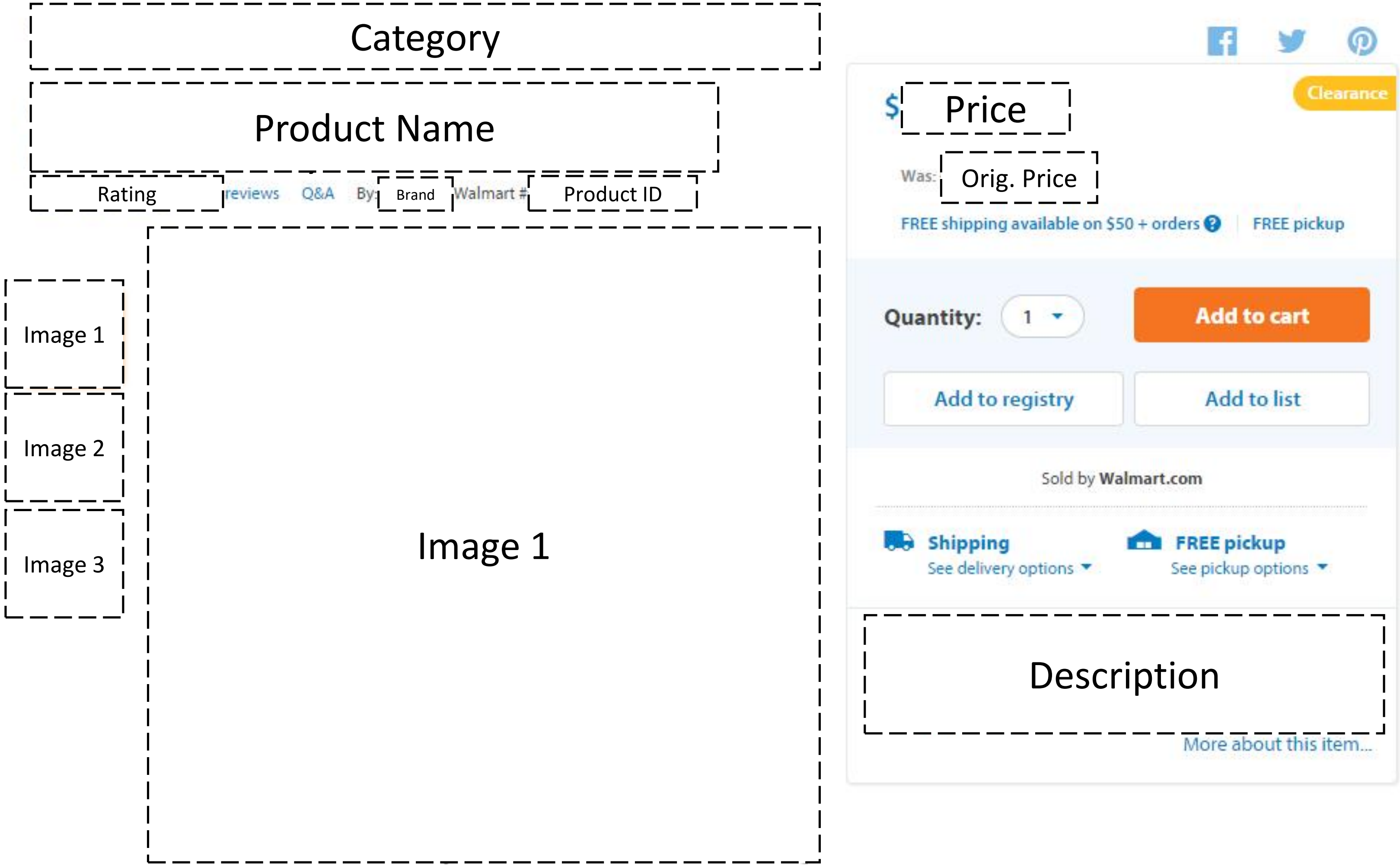
Sold by **Walmart.com**

Shipping See delivery options | **FREE pickup** See pickup options

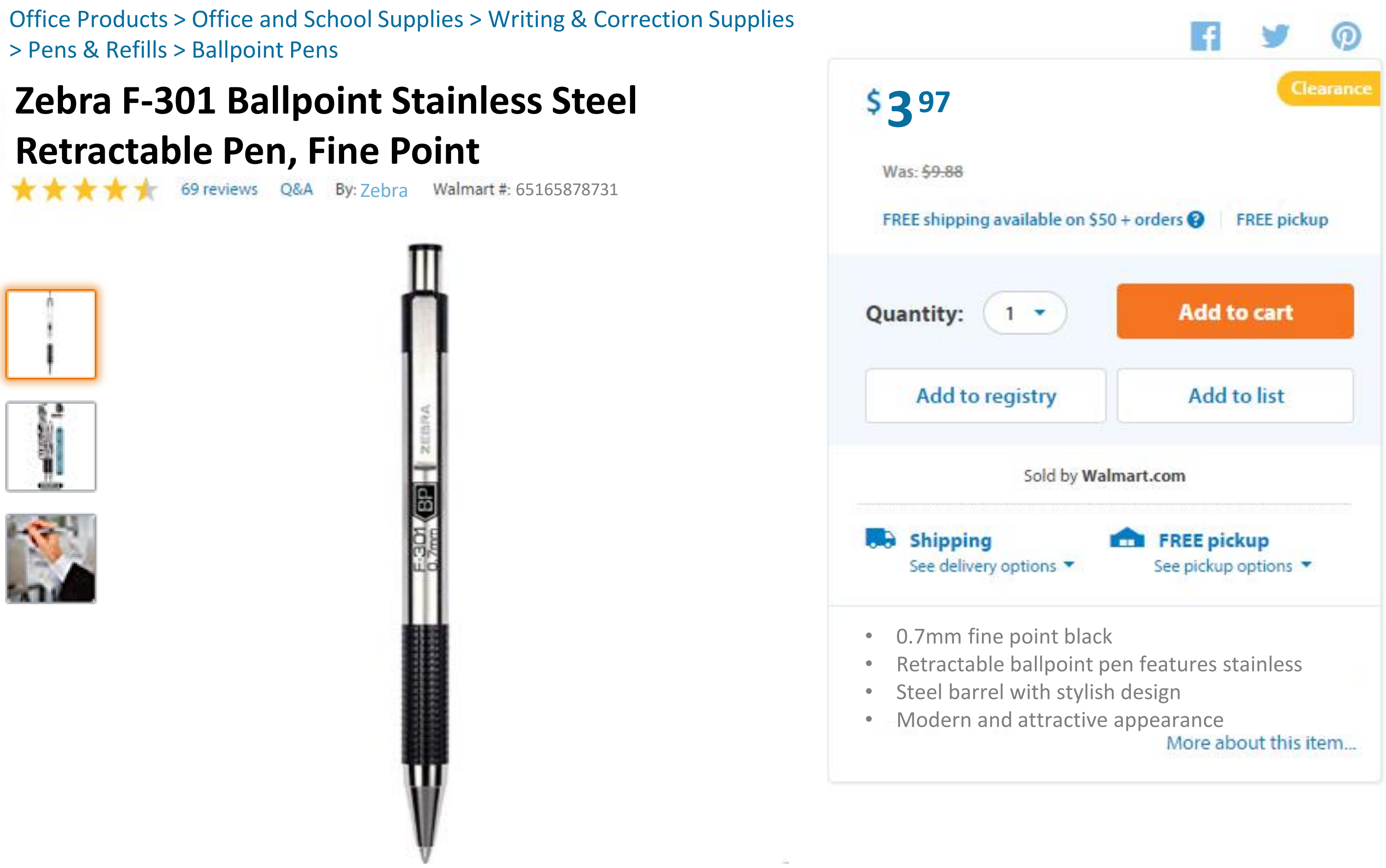
- 2.4Ghz wireless
- 1000 dpi optical sensor for smooth movement control
- Ambidextrous

More about this item...

E-Commerce data – What attributes do you see here?



E-Commerce data – What attributes do you see here?



Office Products > Office and School Supplies > Writing & Correction Supplies > Pens & Refills > Ballpoint Pens

Zebra F-301 Ballpoint Stainless Steel Retractable Pen, Fine Point

★★★★½ 69 reviews Q&A By: Zebra Walmart #: 65165878731

Clearance

$ 3 97

Was: ~~$9.88~~

FREE shipping available on $50 + orders ? | FREE pickup

Quantity: 1 ▾ Add to cart

Add to registry Add to list

Sold by **Walmart.com**

Shipping See delivery options ▾ **FREE pickup** See pickup options ▾

- 0.7mm fine point black
- Retractable ballpoint pen features stainless
- Steel barrel with stylish design
- Modern and attractive appearance

More about this item...

Module 2.2

ER Grammar

- At the end of this discussion, you should be able to describe the following:
    - Entity Type
    - Entity Instance
    - Entity Class
    - Attribute
    - Value
    - Domain
    - Relationship

Horses

Name	Color	Spots	Sex	Weight
Sam	Brown	No	F	1500
Amy	Yellow	Yes	F	920
Dave	Grey	No	M	1800
Ed	Yellow	No	M	1100
Sally	Black	No	F	1200
Sarah	Grey	No	F	1700
Tom	Red	Yes	M	1500
Jim	Red	No	M	1400
Joan	Brown	No	F	1250

Module 2.3

Entities and Attributes

- Describe the characteristics of an attribute:
  - Name Type Classification
  - Category Source Domain
  - Value Optionality Role

The Entity

- Entity type
  - ▫ Conceptual representation of an object type
  - ▫ **A set of related attributes**
  - ▫ Can have relationships with other entity types

- Entity instance: An occurrence of an entity type

- Entity class: A set of entity types that have shared properties

- An entity may be strong or weak

The Attribute

- Describes an entity – and does so in many ways

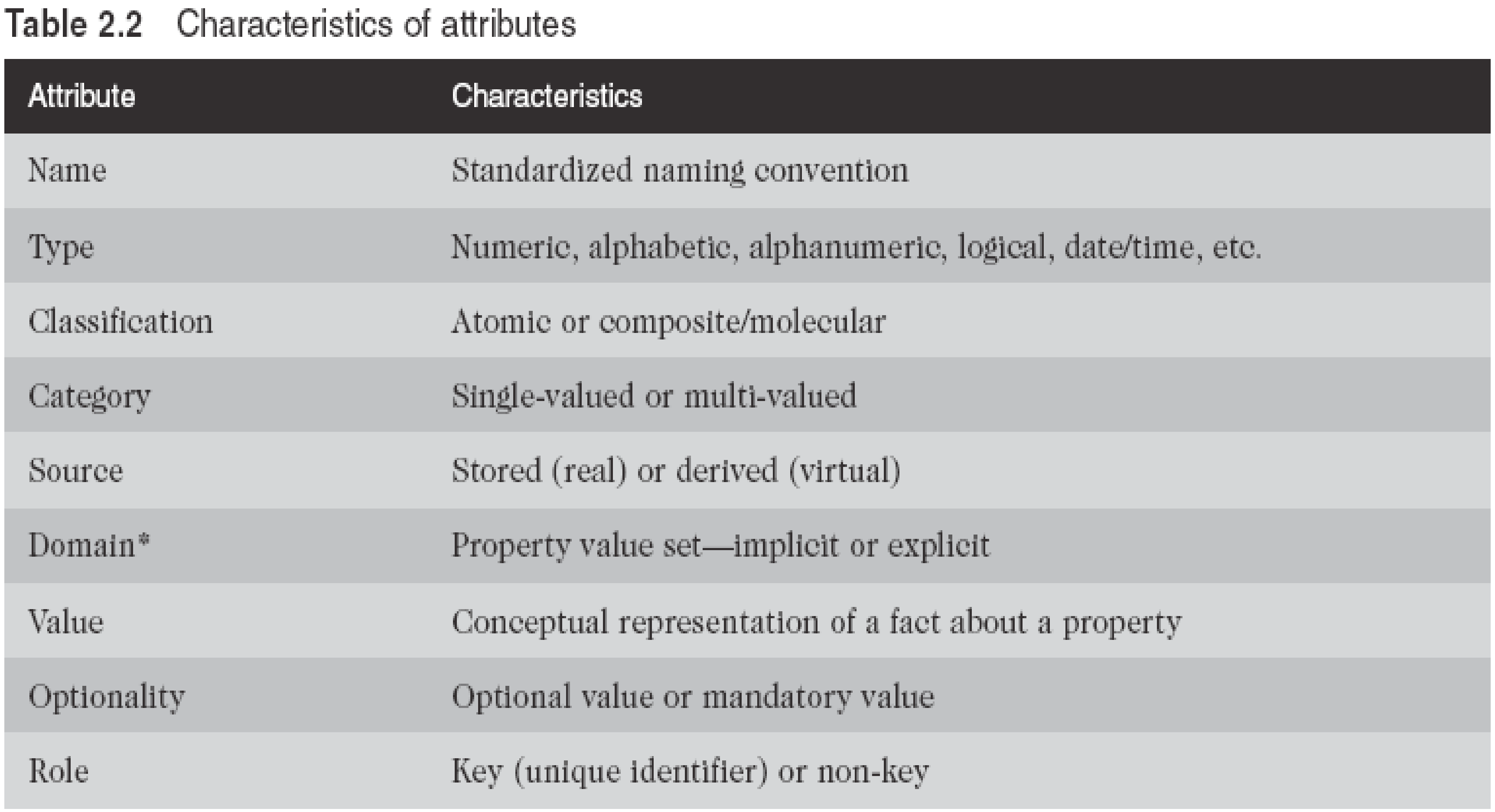


Table 2.2 Characteristics of attributes

Attribute	Characteristics
Name	Standardized naming convention
Type	Numeric, alphabetic, alphanumeric, logical, date/time, etc.
Classification	Atomic or composite/molecular
Category	Single-valued or multi-valued
Source	Stored (real) or derived (virtual)
Domain*	Property value set—implicit or explicit
Value	Conceptual representation of a fact about a property
Optionality	Optional value or mandatory value
Role	Key (unique identifier) or non-key

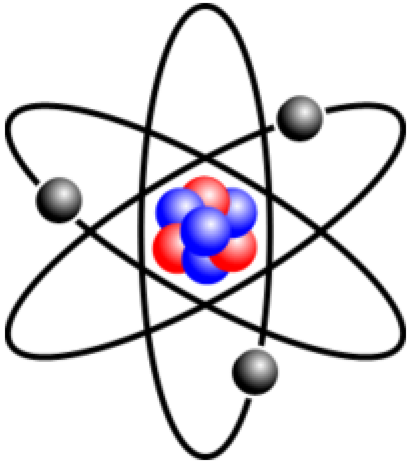
Attribute Type

- Numeric
  - ONLY numbers - useful for doing mathematical operations
  - **Even if a value only contains numbers (credit card, phone, etc.), it you would not do math operations it is typically better to store the value as an alphanumeric data type!**
- Alphanumeric
  - Letters and numbers – 55 West Main Street
  - Numbers that are not useful for math operations (Phone, CC, SSN, etc.)
- Date/time
  - 2:14 AM, 8/29/1997

- Different DBMS have different data types
  - Different types of numeric (integer, floating point, etc.)
  - Different sizes of alphanumeric
  - “Blob” data types for binary data

Attribute Classification: Atomic or Composite

- Just as an atom can be divided no further, an atomic attribute can be (meaningfully) divided no further
  - Last name: Grimes
  - Hair color: Black
  - Height: 72



Attribute Classification: Atomic or Composite

- A composite attribute can be meaningfully divided into smaller attributes
  - Address -> Street address + city + state + zip
    - Street address -> Street name + house number
  - Name -> Salutation + First name + Middle name + Last name
  - Date of Birth -> Month + Day + Year

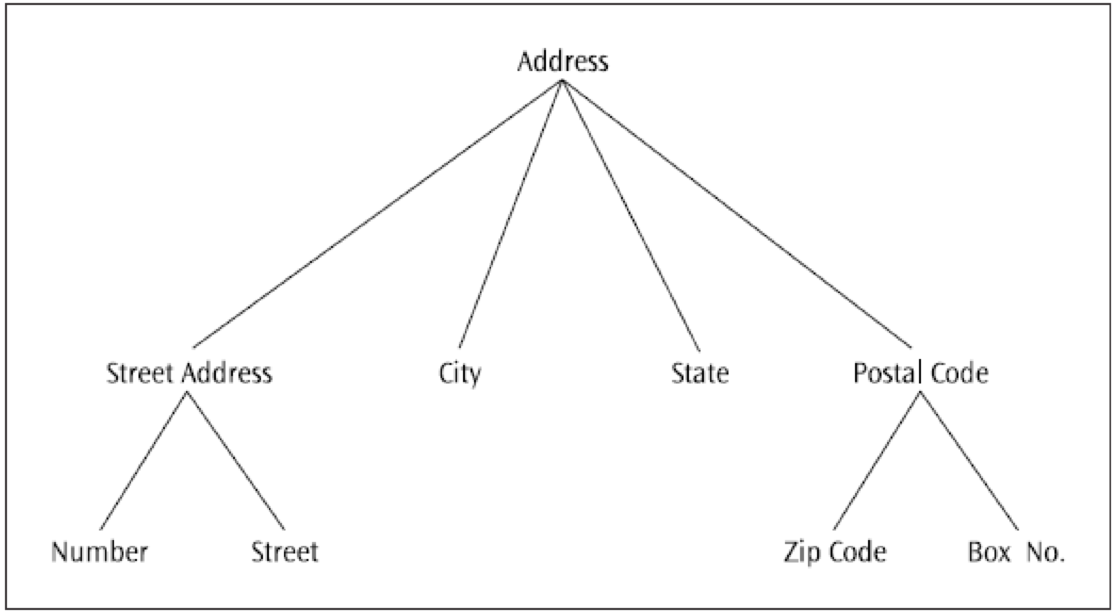
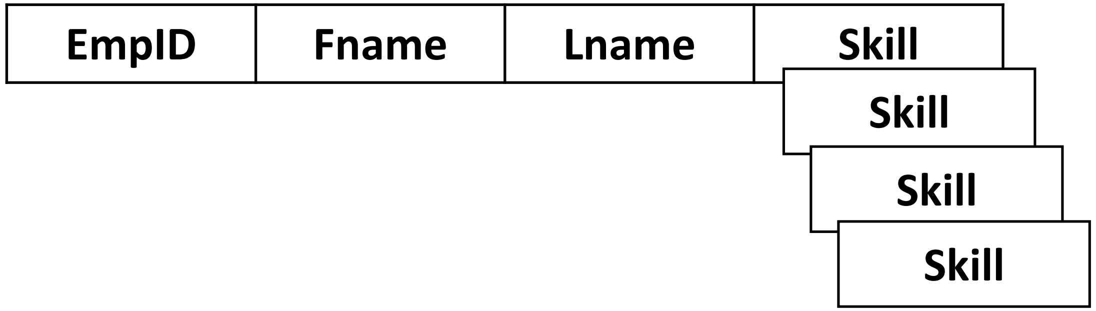


Figure 2.1 An example of a composite attribute hierarchy

Attribute Category: Single or Multi value

- An employee has one employee ID, one first name, one last name, etc…

- An employee may have multiple skills
  - Database design
  - C# Programming
  - Java programming
  - Basket weaving



Attribute Source: Stored or Derived

- An attribute’s value can be stored
  - Things that don’t generally change, such as name

- or-

- An attribute’s value can be derived
  - Things that change with time, such as years of service
    - Derived by subtracting start date from today’s date
  - Why would we want to do this?

Attribute Domain: Implicit or Explicit

- Explicit Domain Constraints
  - Sex: [M, F]
  - Student_type: [Fr, So, Jr, Sr, Gr]

- Implicit Domain Constraints
  - Age: [1 – 120]
  - Salary [17,000 – 3,000,000]

- Whether it is required or optional to have a value at all

Attribute Role: Key (unique) or non-key

- Unique Identifier
  - An attribute (atomic or composite) whose values are distinct for each entity instance in the entity set
    - Employee ID, SSN, ISBN, UPC, etc…

- Key attribute
  - Attribute that is a constituent part of a unique identifier
  - A key attribute is a proper subset of a unique identifier

- Non-key
  - Any attribute that is not a constituent part of (subset of) a unique identifier
    - Name, weight, classification

Entity & Attribute Data Integrity Constraints

- Data integrity constraints are rules that govern behavior of data at all times in a database
  - Technical expressions of business rules

- They must be preserved across all three tiers of data modeling – conceptual, logical and physical

- Some constraints cannot be expressed explicitly in an ERD and are therefore carried forward in textual form (i.e., semantic integrity constraints)

Module 2.3

Entities and Attributes

- Describe the characteristics of an attribute:
  - Name Type Classification
  - Category Source Domain
  - Value Optionality Role

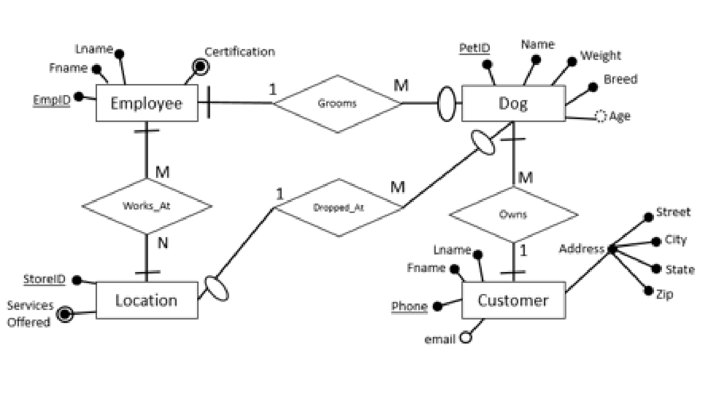
10 minute break

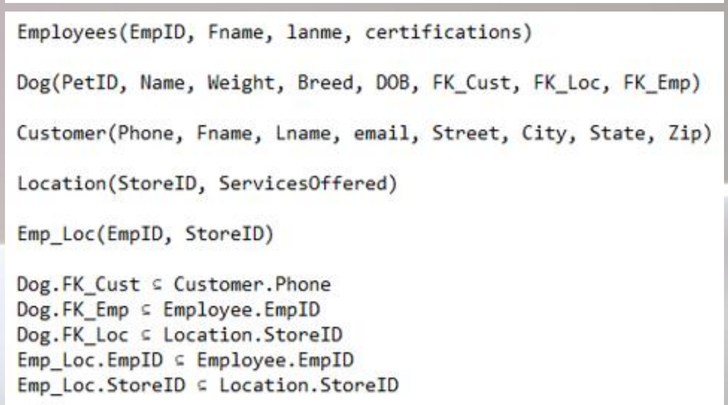
Module 10.1

Implementing Databases

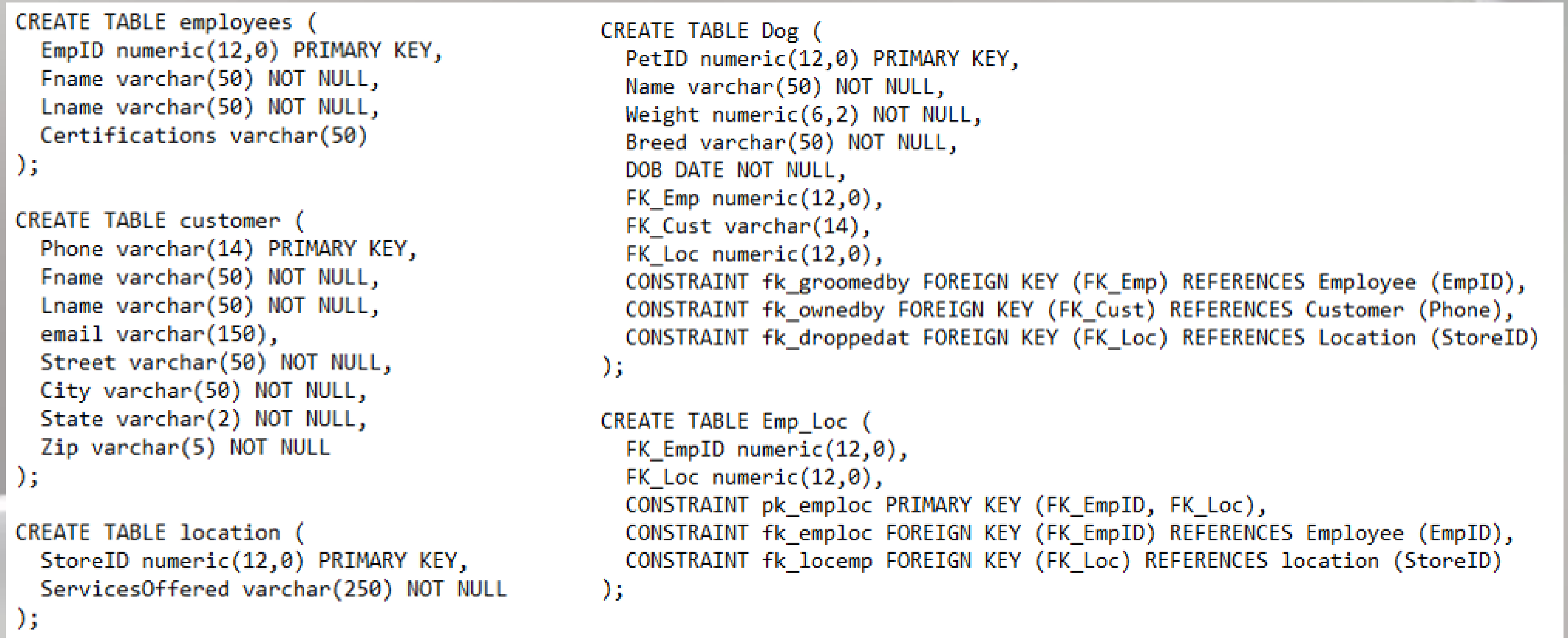
Next step in transforming business rules into an actual DB

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)
);
```

SQL Client – DBeaver (Community Edition)

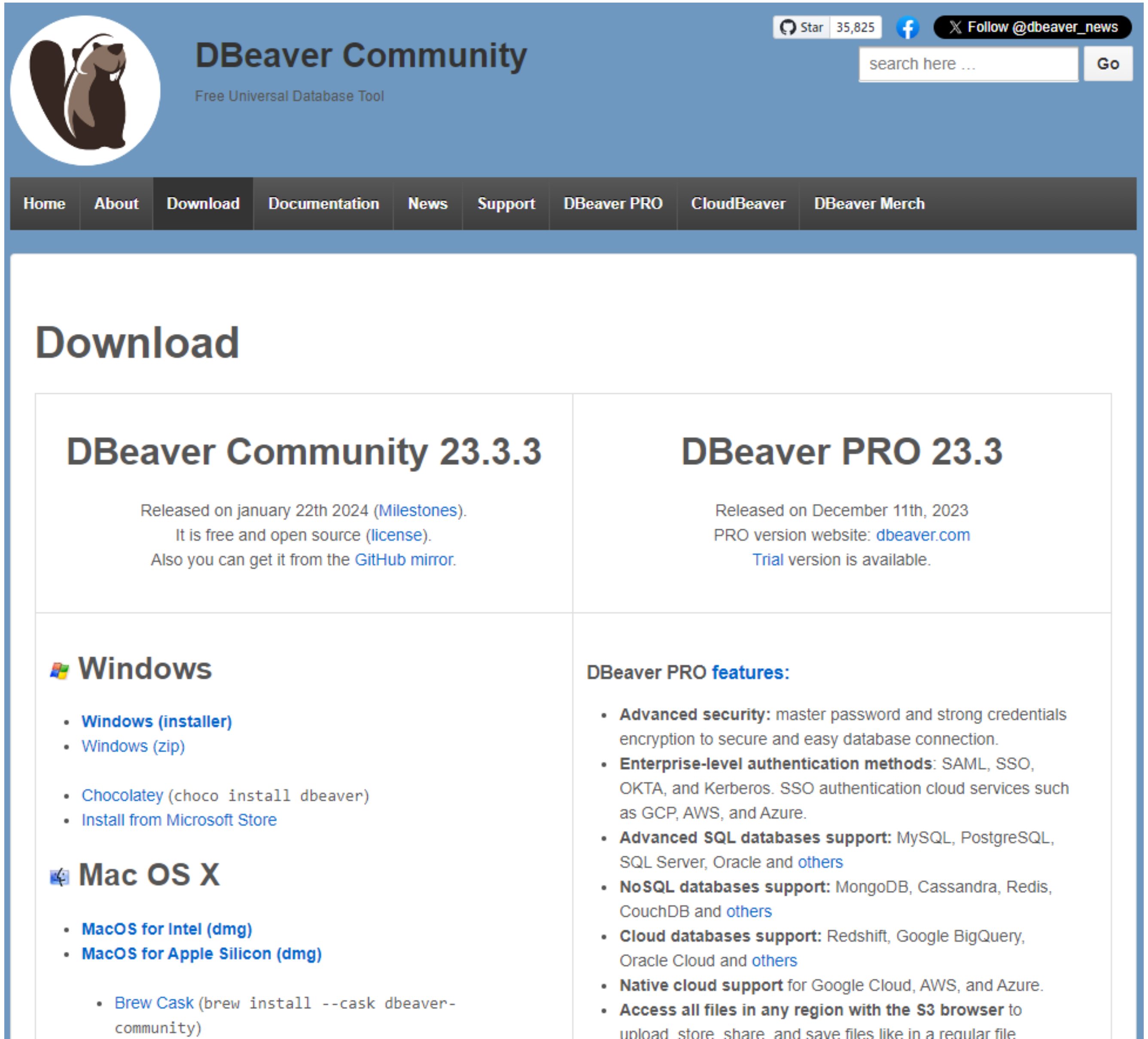
- There are many SQL clients you could use:
  - Oracle SQL developer
  - SQuirreL
  - SQL Workbench
  - TOAD

- We have previously used Oracle SQL Developer in the class…
  - It requires creating an account with Oracle
  - It is kind of clunky

- Trying out DBeaver for this semester
  - It automatically downloads drivers for many different DBMSs
  - If you want to use something else that's fine, but I might not be able to troubleshoot any connection issues you have…

SQL Client – DBeaver (Community Edition)

- https://dbeaver.io/
- Go to Download
- Select your OS from the Community edition section
- Install it accepting the defaults
- If you are asked if you want to create a sample database, just say no.



DBeaver Community

Free Universal Database Tool

Download

DBeaver Community 23.3.3

Released on january 22th 2024 (Milestones).
It is free and open source (license).
Also you can get it from the GitHub mirror.

DBeaver PRO 23.3

Released on December 11th, 2023
PRO version website: dbeaver.com
Trial version is available.

Windows

- Windows (installer)
- Windows (zip)

- Chocolatey (`choco install dbeaver`)
- Install from Microsoft Store

Mac OS X

- MacOS for Intel (dmg)
- MacOS for Apple Silicon (dmg)

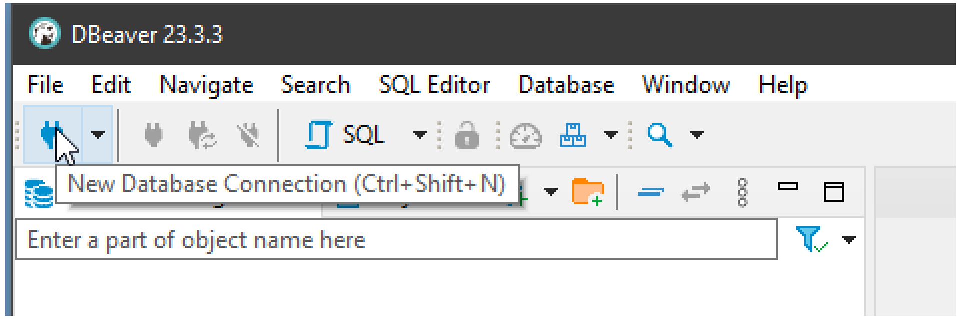
  - Brew Cask (`brew install --cask dbeaver-community`)

DBeaver PRO features:

- **Advanced security:** master password and strong credentials encryption to secure and easy database connection.
- **Enterprise-level authentication methods**: SAML, SSO, OKTA, and Kerberos. SSO authentication cloud services such as GCP, AWS, and Azure.
- **Advanced SQL databases support:** MySQL, PostgreSQL, SQL Server, Oracle and others
- **NoSQL databases support:** MongoDB, Cassandra, Redis, CouchDB and others
- **Cloud databases support:** Redshift, Google BigQuery, Oracle Cloud and others
- **Native cloud support** for Google Cloud, AWS, and Azure.
- **Access all files in any region with the S3 browser** to upload, store, share, and save files like in a regular file

SQL Client – DBeaver (Community Edition)

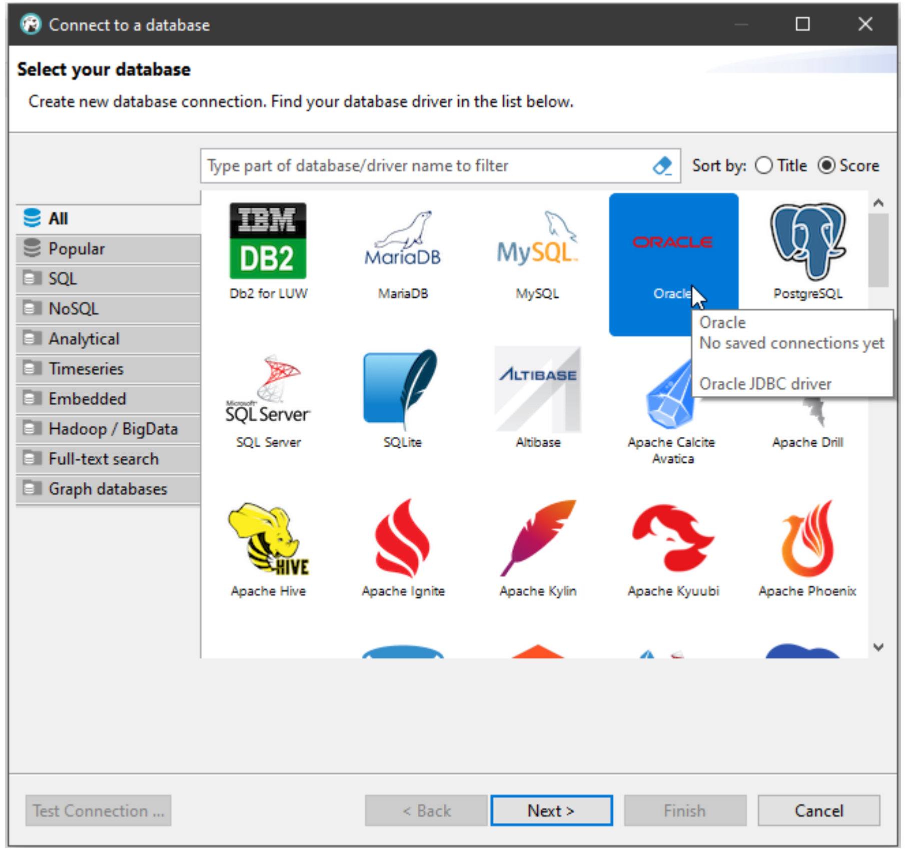
- Click “New Database Connection”



New Database Connection (Ctrl+Shift+N)

Enter a part of object name here

- From “All” select Oracle


Connect to a database

Select your database

Create new database connection. Find your database driver in the list below.

Type part of database/driver name to filter

Sort by: ○ Title ● Score

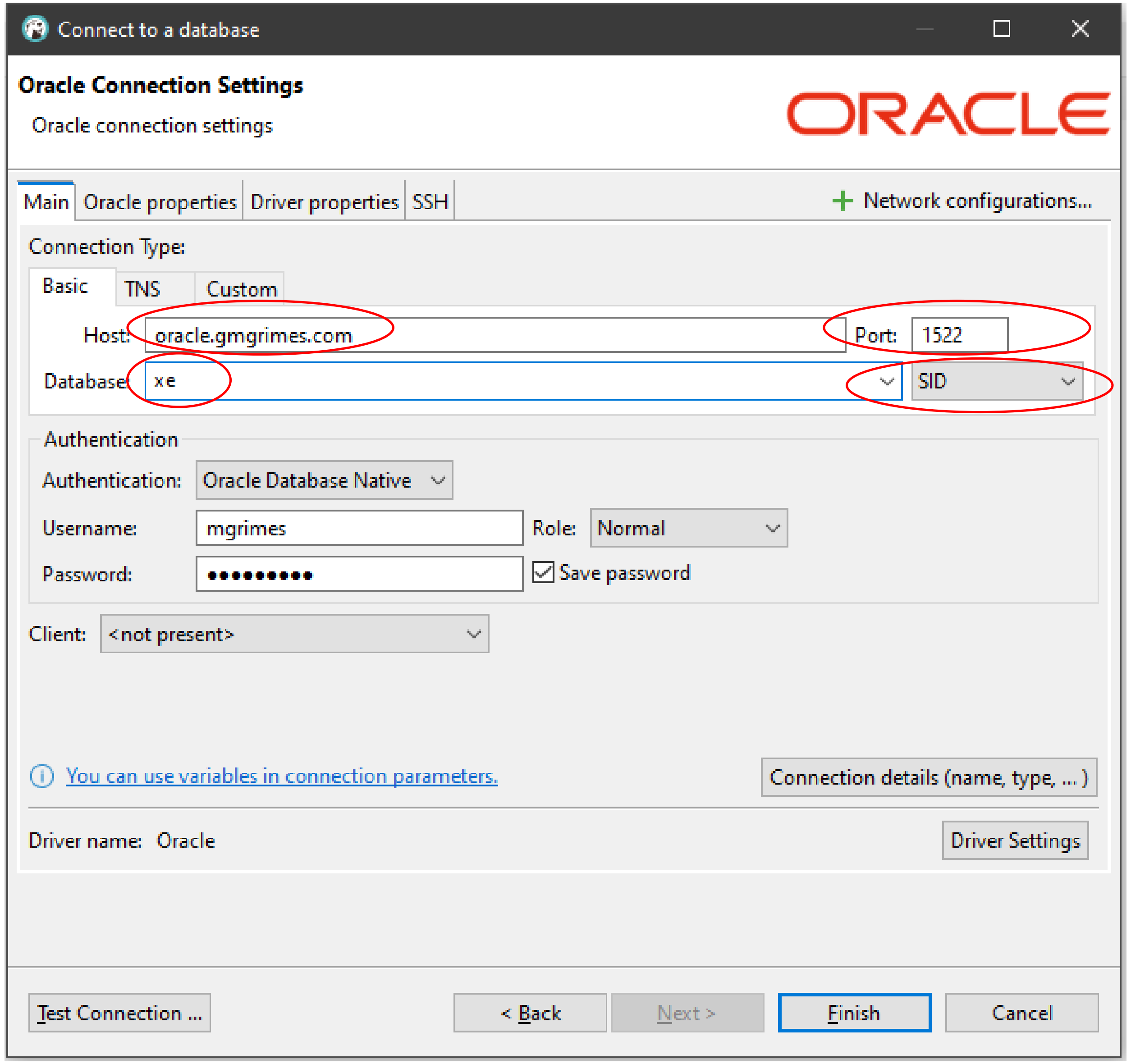
Oracle
No saved connections yet

Oracle JDBC driver

SQL Client – DBeaver (Community Edition)

- In the connect screen
  - Host: oracle.gmgrimes.com
  - Port: 1522 (Change from 1521)
  - Database: xe
  - Change “Service name” to “SID”

- Username and password will be provided by me



Connect to a database

Oracle Connection Settings

Oracle connection settings

Connection Type: Basic

Host: oracle.gmgrimes.com

Port: 1522

Database: xe

SID

Authentication

Authentication: Oracle Database Native

Username: mgrimes

Role: Normal

Password: ••••••••••

☑ Save password

Client: <not present>

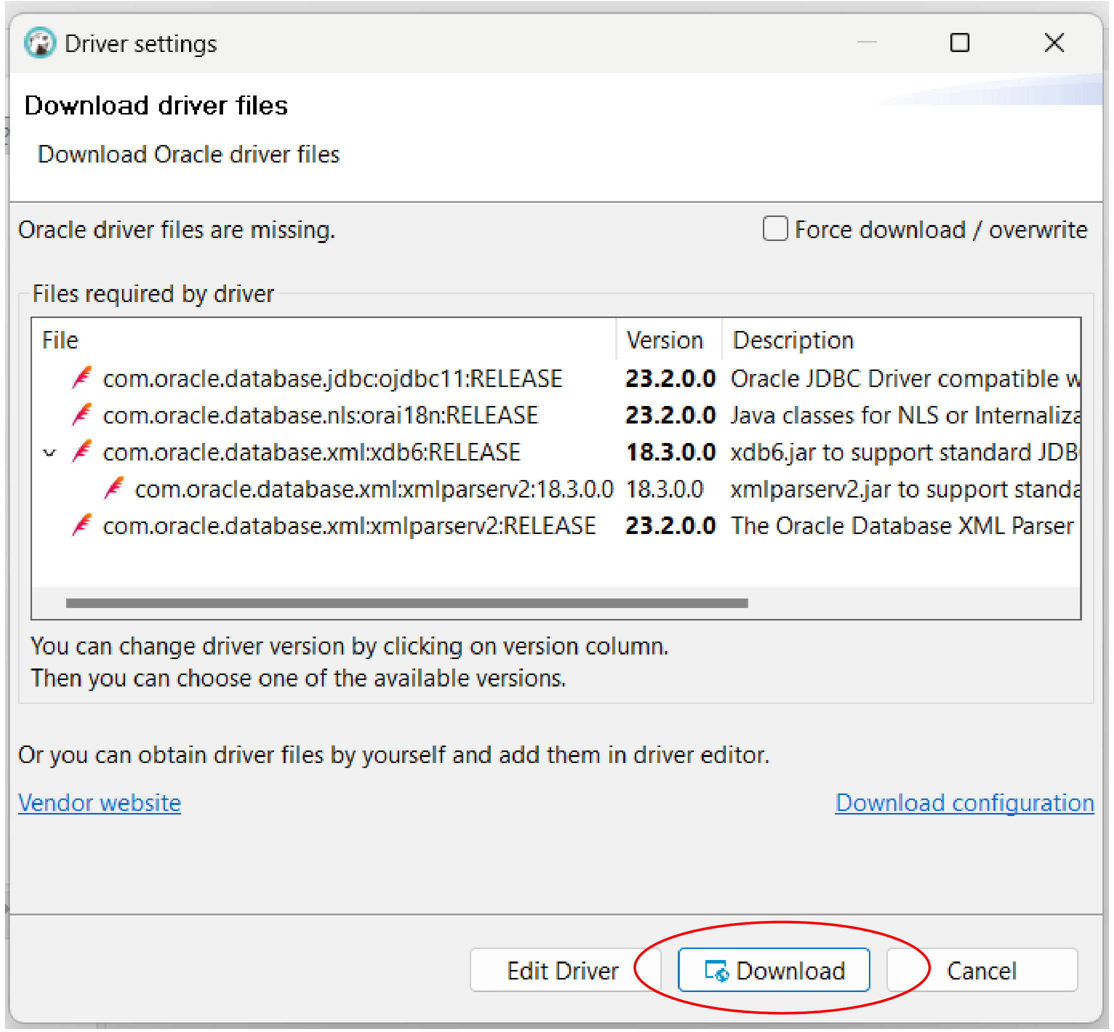
You can use variables in connection parameters.

Driver name: Oracle

SQL Client – DBeaver (Community Edition)

- If you are asked to send anonymous usage data, I would suggest not, but either is ok…

- You will probably see a screen like this the first time you try to connect – just click Download.



Download driver files

Download Oracle driver files

Oracle driver files are missing.

□ Force download / overwrite

Files required by driver

File	Version	Description
com.oracle.database.jdbc:ojdbc11:RELEASE	**23.2.0.0**	Oracle JDBC Driver compatible w
com.oracle.database.nls:orai18n:RELEASE	**23.2.0.0**	Java classes for NLS or Internaliza
com.oracle.database.xml:xdb6:RELEASE	**18.3.0.0**	xdb6.jar to support standard JDB
com.oracle.database.xml:xmlparserv2:18.3.0.0	18.3.0.0	xmlparserv2.jar to support standa
com.oracle.database.xml:xmlparserv2:RELEASE	**23.2.0.0**	The Oracle Database XML Parser

You can change driver version by clicking on version column.
Then you can choose one of the available versions.

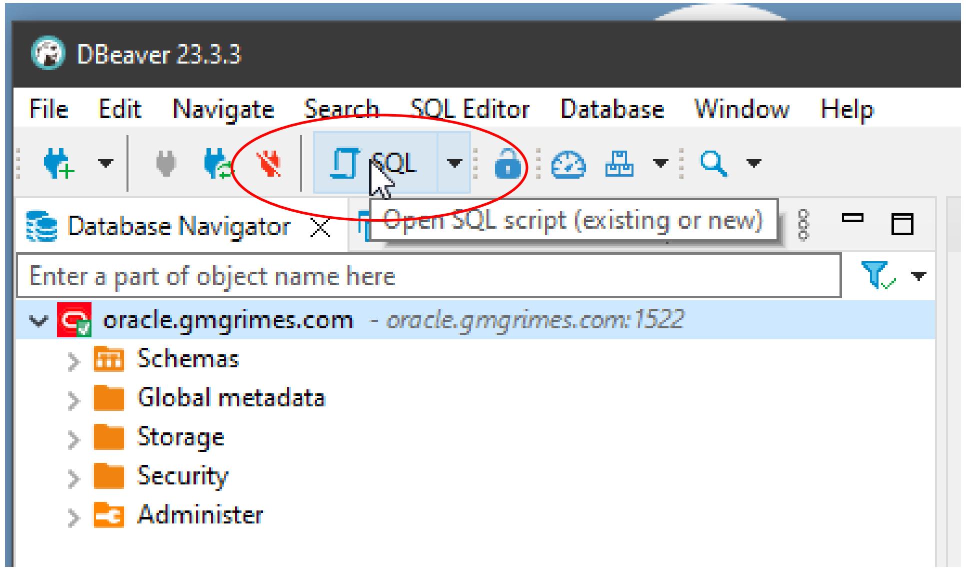
Or you can obtain driver files by yourself and add them in driver editor.

Vendor website

Download configuration

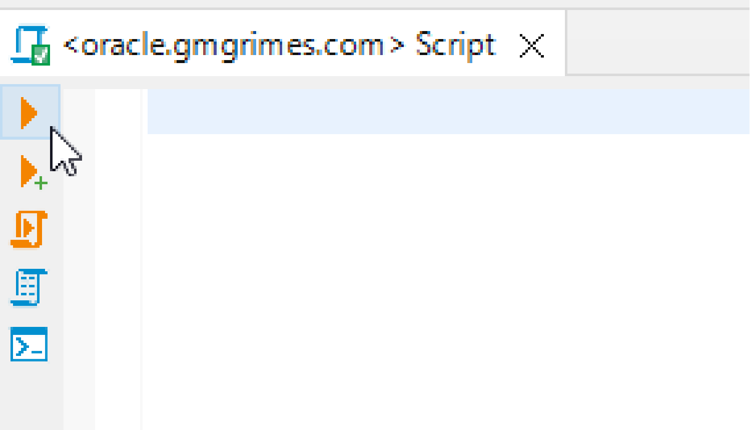
SQL Client – DBeaver (Community Edition)

- Click on the "SQL" button to start writing SQL queries



- To run a query, hit <ctrl> + <enter> or click the orange play button

- To run multiple queries at once his the Execute SQL Script button (third one) →



Database Creation

- The principle tasks include
  - Creation and modification of the database tables and other related structures
  - Enforcement of integrity constraints
  - Population of the database tables

- Three major constructs of DDL
  - CREATE
  - ALTER
  - DROP

Notes about SQL

- Capitalization and spacing do not matter
- Can all be on one line or across multiple lines
- SQL statements end with a semicolon (;)
- Important that parentheses and quotation marks match
- Text attributes are enclosed in single quotes
- When giving examples, square brackets mean [optional]

A sneak peak at INSERT and SELECT

- We’ll talk more about this later, but so we can see the fruits of our labor:

- To insert data into a table:
  - ▫ INSERT INTO *table* VALUES (*comma delimited list of values*);

  or
  - ▫ INSERT INTO *table* (*comma delimited list of attributes)* VALUES (*comma delimited list of values*);

- To select data from a table:
  - ▫ SELECT * FROM *table*;

  or
  - ▫ SELECT *Att1, Att2, Att3* FROM *table*;

CREATE TABLE Syntax

- CREATE TABLE table_name (comma delimited list of table-elements);

- table_name is a user supplied name for the TABLE

- Each table-element in the list is either:
  - Column definition (i.e., attribute name)
  - Constraint definition

- The basic syntax for a column-definition is of the form:
  column_name representation [default-definition] [column-constraint list]
  - Column_name is a user supplied name for a COLUMN
  - Representation specifies the data type
  - The [optional] default-definition specifies a default value. Absent an explicit default definition, the implicit assumption is NULL.
  - The [optional] column-constraint list specifies constraint-definitions. The syntax for constraint-definition is: [CONSTRAINT constraint_name] constraint-definition

Creating the Horses table / schema (no constraints) (CRUD)

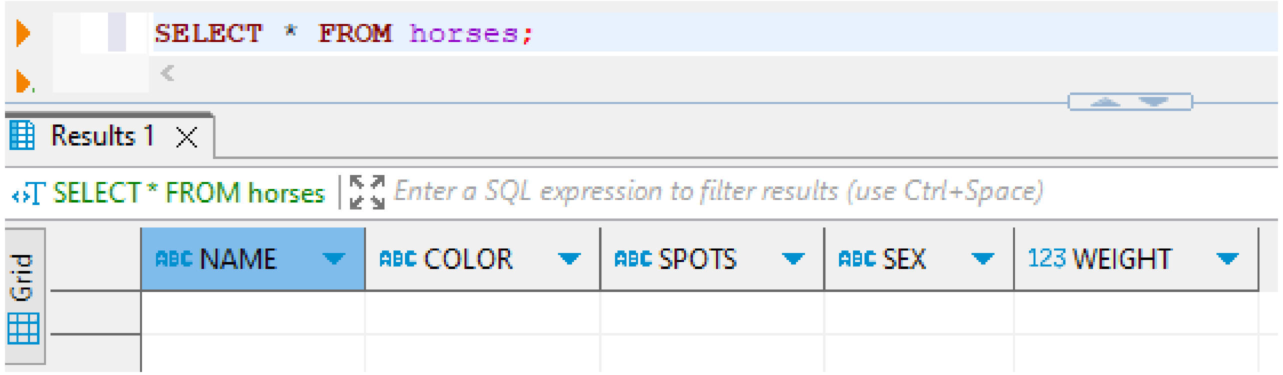
- To create a basic version of our Horses table:

```
CREATE TABLE horses
(Name       varchar(50),
 Color      varchar(50),
 Spots      varchar(3),
 Sex        varchar(1),
 Weight     integer);
```

- You can see the table with:

```
SELECT * FROM horses;
```

- Currently no data there! →



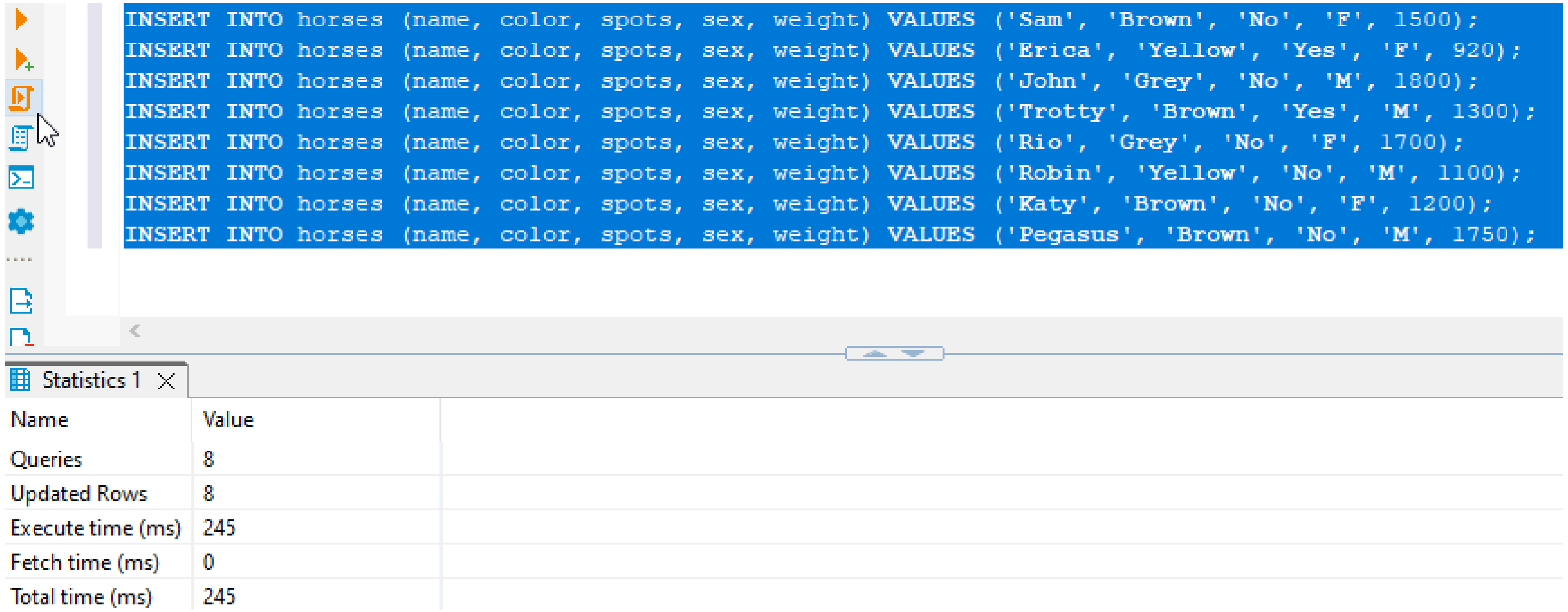
```
SELECT * FROM horses;
```

ABC NAME	ABC COLOR	ABC SPOTS	ABC SEX	123 WEIGHT

Inserting data into the Horses table (CRUD)

```
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('Sam', 'Brown', 'No', 'F', 1500);
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('Erica', 'Yellow', 'Yes', 'F', 920);
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('John', 'Grey', 'No', 'M', 1800);
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('Trotty', 'Brown', 'Yes', 'M', 1300);
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('Rio', 'Grey', 'No', 'F', 1700);
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('Robin', 'Yellow', 'No', 'M', 1100);
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('Katy', 'Brown', 'No', 'F', 1200);
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('Pegasus', 'Brown', 'No', 'M', 1750);
```

To run multiple SQL statements at once click this “Execute SQL Script” button →



```
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('Sam', 'Brown', 'No', 'F', 1500);
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('Erica', 'Yellow', 'Yes', 'F', 920);
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('John', 'Grey', 'No', 'M', 1800);
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('Trotty', 'Brown', 'Yes', 'M', 1300);
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('Rio', 'Grey', 'No', 'F', 1700);
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('Robin', 'Yellow', 'No', 'M', 1100);
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('Katy', 'Brown', 'No', 'F', 1200);
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('Pegasus', 'Brown', 'No', 'M', 1750);
```

Name	Value
Queries	8
Updated Rows	8
Execute time (ms)	245
Fetch time (ms)	0
Total time (ms)	245

General form of SELECT FROM WHERE (CRUD)

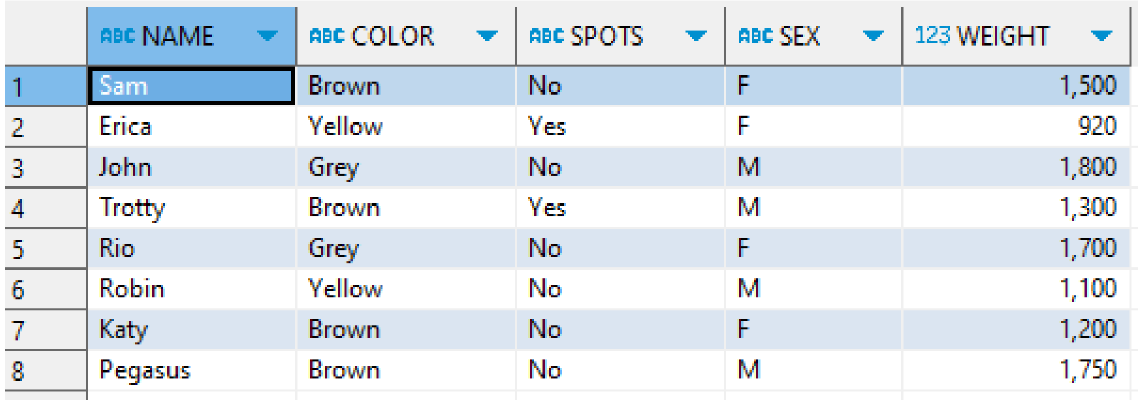
-
  ```
  SELECT <column list>
      FROM <table list>
      WHERE <condition>
  ```
  ← Optional

- <column list> is a list of column names (attributes) whose values are to be projected

- <table list> is a list of the table names (relations) required to process the query

- <condition> is a conditional (Boolean) expression that identifies the rows to be retrieved by the query.

Running some queries against the Horses table (CRUD)



	NAME	COLOR	SPOTS	SEX	WEIGHT
1	Sam	Brown	No	F	1,500
2	Erica	Yellow	Yes	F	920
3	John	Grey	No	M	1,800
4	Trotty	Brown	Yes	M	1,300
5	Rio	Grey	No	F	1,700
6	Robin	Yellow	No	M	1,100
7	Katy	Brown	No	F	1,200
8	Pegasus	Brown	No	M	1,750

- Show me all data about all horses:
  ```
  SELECT * FROM Horses;
  ```
- Show me all data about Female horses
  ```
  SELECT * FROM Horses WHERE sex = 'F';
  ```
- Show me all data about Brown horses
  ```
  SELECT * FROM Horses WHERE color = 'Brown';
  ```
- Show me all data about horses that weigh over 1,500 pounds
  ```
  SELECT * FROM Horses WHERE weight > 1500;
  ```
- Show me all data about horses that weigh 1,500 pounds or more
  ```
  SELECT * FROM Horses WHERE weight >= 1500;
  ```
- Show me all data about horses that are Brown AND weigh 1,500 pounds or more
  ```
  SELECT * FROM Horses WHERE weight >= 1500 AND color='Brown';
  ```

…but now some problems…

- Based on our previous discussion, we have some DATA INTEGRITY CONTRAINTS that should be applied:
  - Horses should be uniquely identified by their names (is this the best long-term decision, though?)
  - Colors should have a domain of reasonable values (Black, White, Brown, Grey, Yellow, Red)
  - Weight should be between 800 and 2200
  - Sex should either be M or F and is required
  - If a value is not provided for Color or Spots, a default of “UNK” should be provided (for unknown)

- We are able to insert multiple horses with the same name – how to tell them apart?:

```
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('Sam', 'Black', 'Yes', 'M', 2200);
```

- We are able to insert horses with any value for Color

```
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('Pinky', 'Pink', 'No', 'M', 1050);
```

- We are able to insert horses with absurd values for weight

```
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('Hulk', 'Grey', 'No', 'M', 91050);
```

- We are able to insert horses without a value for sex

```
INSERT INTO horses (name, color, spots, weight) VALUES ('Pat', 'White', 'No', 1400);
```

Dropping the Horses table (CRU**D**)

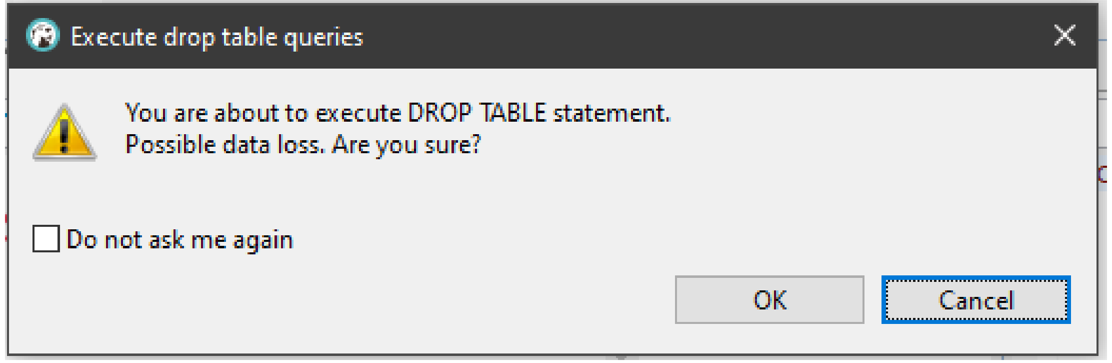
- Let’s get rid of what we’ve done so far:

```
DROP TABLE Horses;
```

- Dropping a table gets rid of the table and all the data contained in the table
  - Very fast and few “guardrails”
  - As we will see later in the semester, drops can also “Cascade” to other tables in some cases
  - Improper use can be a “resume generating event” – be careful!

- DBeaver does very kindly give us a warning, but this is coming from DBeaver, NOT from Oracle!

- Not every client will do this, and you will not get a warning like this if “directly connected”



Execute drop table queries

You are about to execute DROP TABLE statement.
Possible data loss. Are you sure?

□ Do not ask me again

Recreating the Horses table / schema (with constraints) (CRUD)

- 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'))
);
```

Put the original data back into the Horses table… (CRUD)

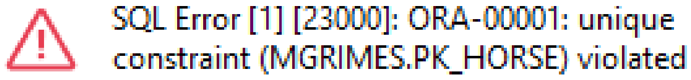
```
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('Sam', 'Brown', 'No', 'F', 1500);
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('Erica', 'Yellow', 'Yes', 'F', 920);
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('John', 'Grey', 'No', 'M', 1800);
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('Trotty', 'Brown', 'Yes', 'M', 1300);
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('Rio', 'Grey', 'No', 'F', 1700);
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('Robin', 'Yellow', 'No', 'M', 1100);
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('Katy', 'Brown', 'No', 'F', 1200);
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('Pegasus', 'Brown', 'No', 'M', 1750);
```

(Note: These are just the same insert statements from a few slides back)

Data integrity problems resolved!

- We are no longer able to insert multiple horses with the same name:

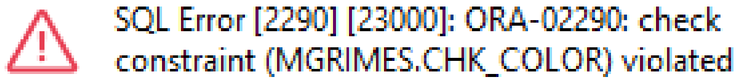
```
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('Sam', 'Black', 'Yes', 'M', 2200);
```



SQL Error [1] [23000]: ORA-00001: unique constraint (MGRIMES.PK_HORSE) violated

- Colors must be in the domain of values we specified

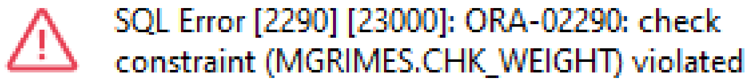
```
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('Pinky', 'Pink', 'No', 'M', 1050);
```



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

- Weight must be in the domain of values we specified

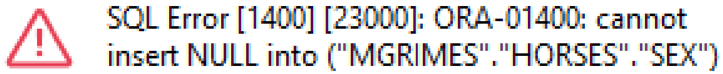
```
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('Hulk', 'Grey', 'No', 'M', 91050);
```



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

- It is required to have a value for sex

```
INSERT INTO horses (name, color, spots, weight) VALUES ('Pat', 'White', 'No', 1400);
```



SQL Error [1400] [23000]: ORA-01400: cannot insert NULL into ("MGRIMES"."HORSES"."SEX")

Data integrity problems resolved!

- These INSERT statements (with appropriately modified values) will work:

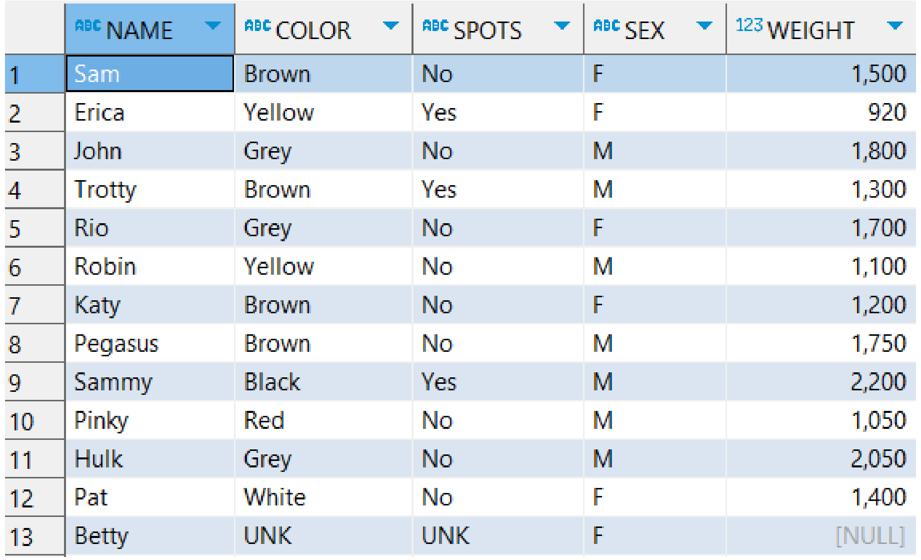
```
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('Sammy', 'Black', 'Yes', 'M', 2200);
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('Pinky', 'Red', 'No', 'M', 1050);
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('Hulk', 'Grey', 'No', 'M', 2050);
INSERT INTO horses (name, color, spots, sex, weight) VALUES ('Pat', 'White', 'No', 'F', 1400);
```

NULL values

- You could also enter a horse with no value for color, spots, or weight:

```
INSERT INTO horses (name, sex) VALUES ('Betty', 'F');
```

  - …perhaps at intake time we don’t care so much about color and spots, and we have to wait for a vet to weigh them, but it’s important to have a value for name and sex!
  - Since we provided a DEFAULT value for Color and Spots we see that (UNK), but since we did not provide a DEFAULT for Weight we get NULL



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

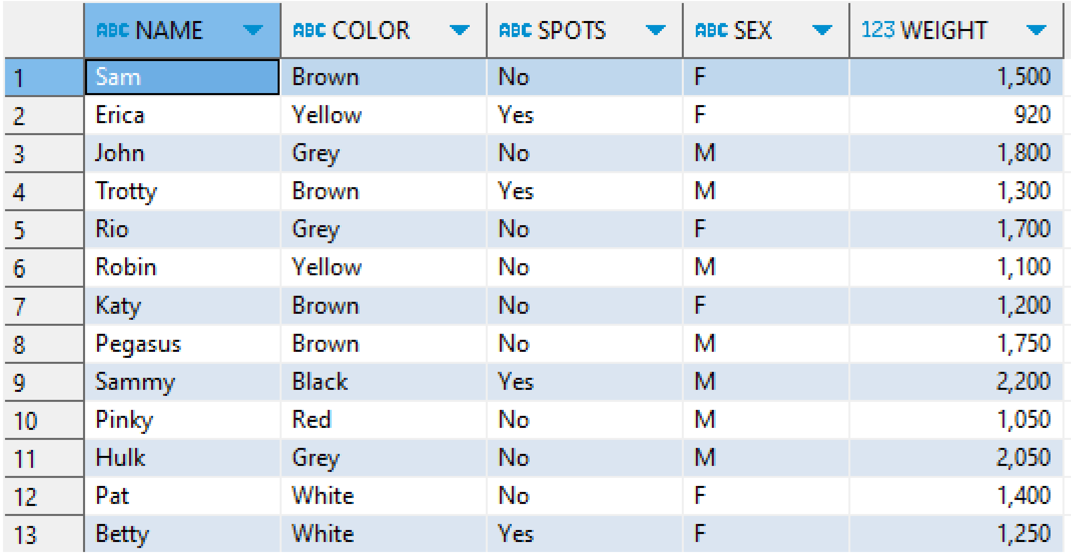
- By default, all attributes are optional EXCEPT for the Primary Key and any attributes with the NOT NULL constraint

- However, as we will see later in the semester, NULL values are problematic are should be avoided as much as possible in your design!

Updating values (CRUD)

- You can update existing values with the update command, which will commonly (almost always) be combined with WHERE to update only a subset of rows:

```
UPDATE horses SET Color='White', Spots='Yes', Weight=1250 WHERE name='Betty';
```



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

- If you do not provide a WHERE clause, all rows will be updated with the provided values, which is almost never what you want
  - It is possible, but uncommon…

Altering your schema (CRUD)

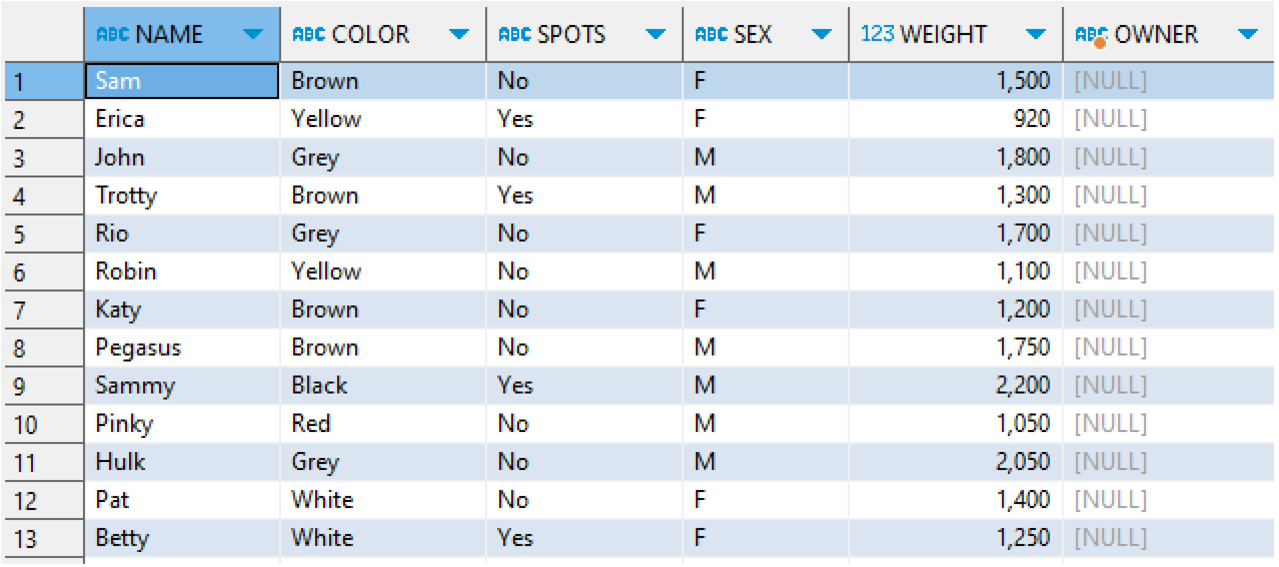
- If your schema changes, you can easily update using the ALTER command:
- ALTER TABLE table_name action;
- table_name is the name of the base TABLE being altered.
- Possible actions are:
    - ADD [COLUMN] column_definition
    - ALTER [COLUMN] column_name { SET default-definition | DROP DEFAULT }
        - (Adds the default-definition or replaces an existing default-definition) or
        - (removes an existing default-definition)
    - DROP [COLUMN] column_name { RESTRICT | CASCADE }
    - ADD CONSTRAINT table_constraint_definition
        - (Permits addition to existing set of constraints, if any)
    - DROP CONSTRAINT constraint_name { RESTRICT | CASCADE }
        - (Removes the named constraint)

Altering your schema (CRUD)

- If your schema changes, you can easily modify the table using the ALTER command:

- Imagine we want to track the name of the owner of each horse. We can add attributes for that:

```
ALTER TABLE horses ADD owner varchar(50);
```

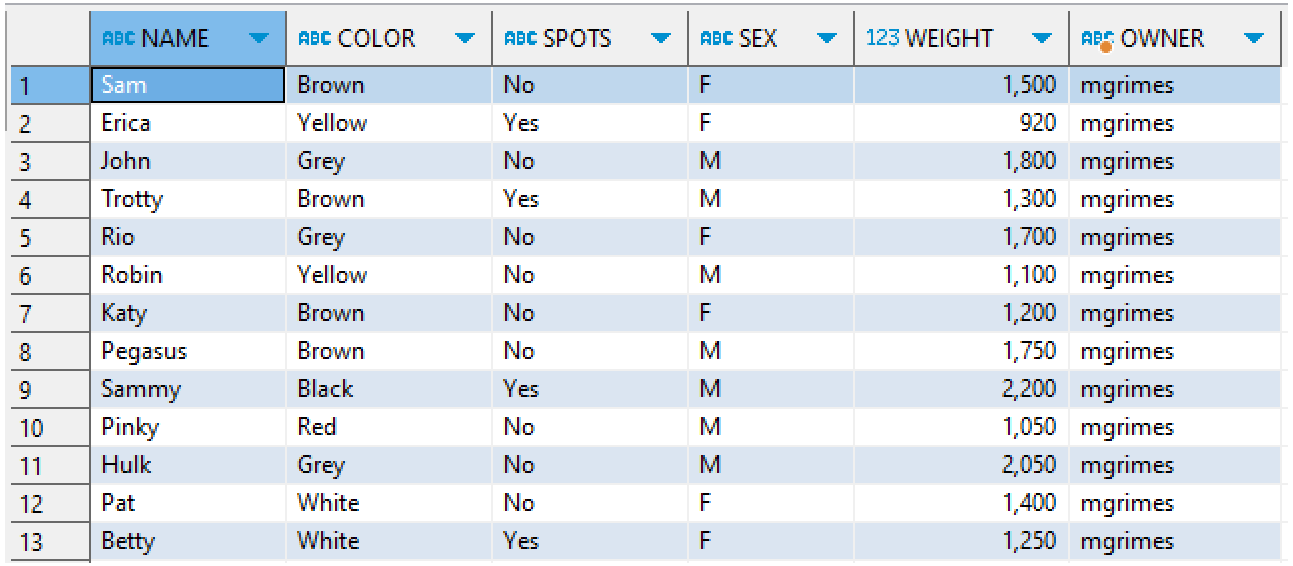


	NAME	COLOR	SPOTS	SEX	WEIGHT	OWNER
1	Sam	Brown	No	F	1,500	[NULL]
2	Erica	Yellow	Yes	F	920	[NULL]
3	John	Grey	No	M	1,800	[NULL]
4	Trotty	Brown	Yes	M	1,300	[NULL]
5	Rio	Grey	No	F	1,700	[NULL]
6	Robin	Yellow	No	M	1,100	[NULL]
7	Katy	Brown	No	F	1,200	[NULL]
8	Pegasus	Brown	No	M	1,750	[NULL]
9	Sammy	Black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

Altering your schema (CRUD)

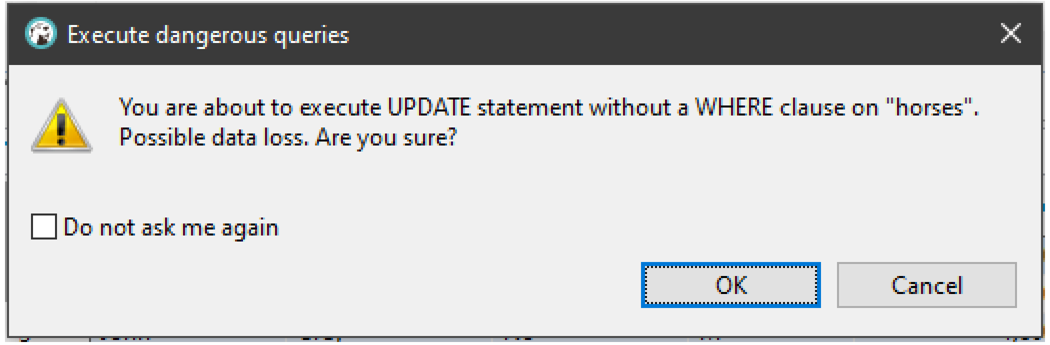
- Now we can update the owner attribute – this would set all horses to have the same owner:

```
UPDATE horses SET owner='mgrimes';
```



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

- Doing this is so uncommon that DBeaver → even warns you



Execute dangerous queries

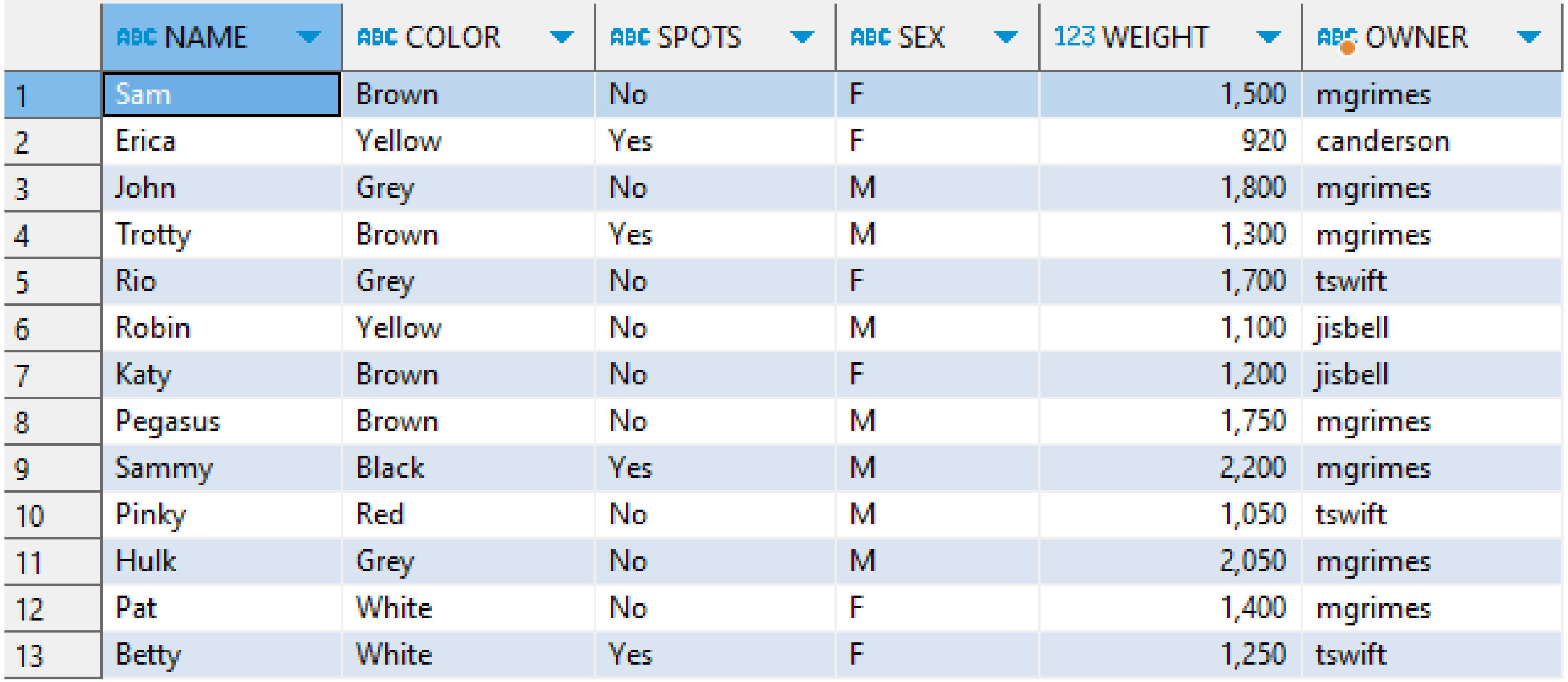
You are about to execute UPDATE statement without a WHERE clause on "horses". Possible data loss. Are you sure?

☐ Do not ask me again

Altering your schema (CRUD)

- Maybe this would be more reasonable:

```
UPDATE horses SET owner='canderson' WHERE name = 'Erica';
UPDATE horses SET owner='tswift' WHERE name IN ('Betty','Pinky','Rio');
UPDATE horses SET owner='jisbell' WHERE name IN ('Katy','Robin');
```



	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

Altering your schema (CRUD)

- If you want to change constraints, you can do so using the ALTER command:

- Imagine we did want to allow Pink as a color – we would DROP the old constraint then add a new CHECK constraint:

```
ALTER TABLE horses DROP CONSTRAINT chk_color;
```

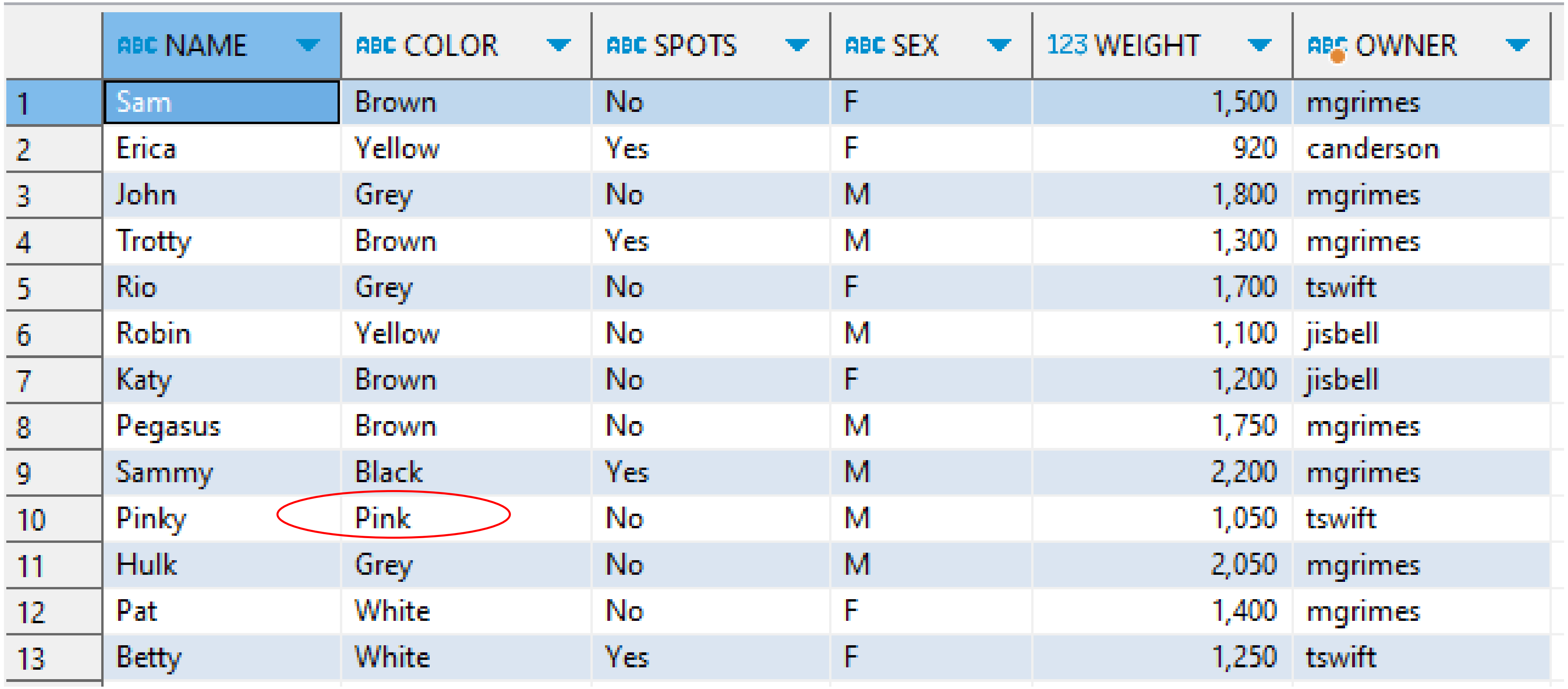
```
ALTER TABLE horses ADD CONSTRAINT chk_color CHECK (color IN
('Black','White','Brown','Grey','Red','Yellow','Pink','UNK'));
```

- Now we could make “Pinky” a pink horse!

```
UPDATE Horses SET color = 'Pink' WHERE name = 'Pinky';
```

Altering your schema (CRUD)

- `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	Pink	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

Altering your schema (CRUD)

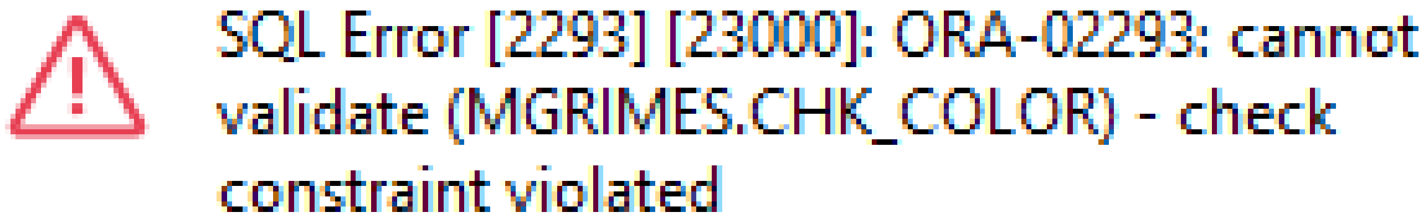
- …but what if our constraint violates data that is already in place?

- Imagine we changed our mind and did NOT want to allow Pink as a color, but we already have a pink horse…

-

```
ALTER TABLE horses DROP CONSTRAINT chk_color;

ALTER TABLE horses ADD CONSTRAINT chk_color CHECK (color IN
('Black','White','Brown','Grey','Red','Yellow','UNK'));
```

Pink has been removed



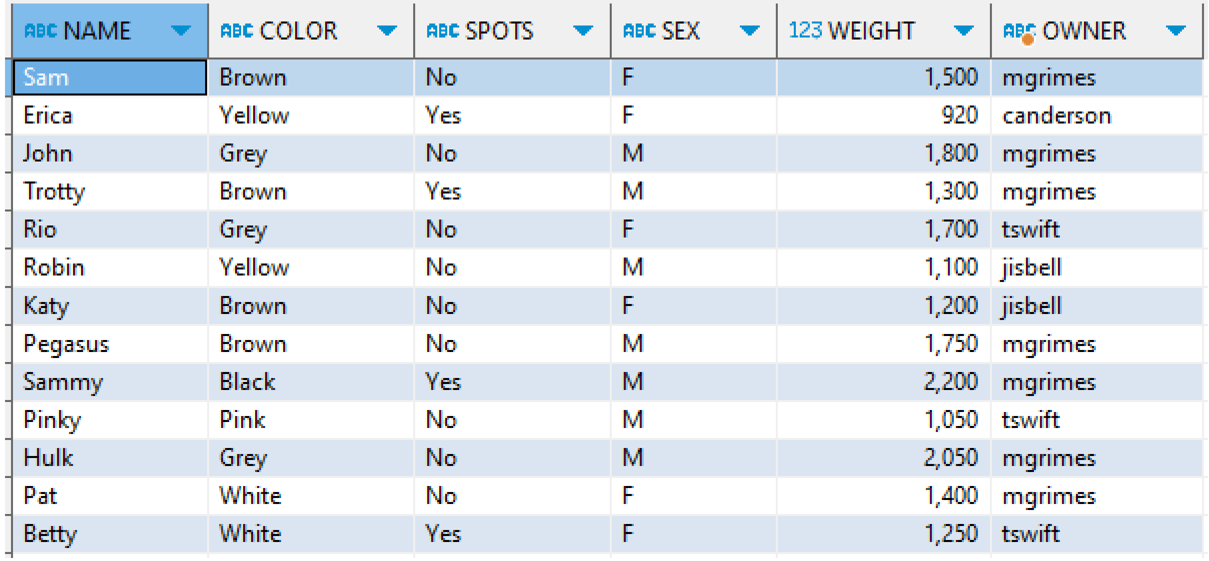
SQL Error [2293] [23000]: ORA-02293: cannot validate (MGRIMES.CHK_COLOR) - check constraint violated

Altering your schema (CRUD)

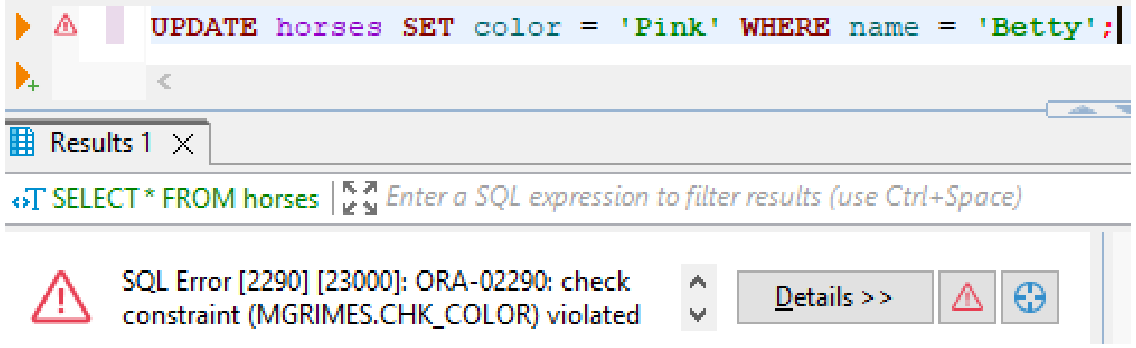
- **This is a business decision** – perhaps it is best to remediate the violations first… OR, if you only want to apply the constraint going forward you could add the “**novalidate**” option:

```
ALTER TABLE horses ADD CONSTRAINT chk_color CHECK (color IN
('Black','White','Brown','Grey','Red','Yellow','UNK')) novalidate;
```

- Now our pink horse can continue to exist, but no more pink horses could be added



NAME	COLOR	SPOTS	SEX	123 WEIGHT	OWNER
Sam	Brown	No	F	1,500	mgrimes
Erica	Yellow	Yes	F	920	canderson
John	Grey	No	M	1,800	mgrimes
Trotty	Brown	Yes	M	1,300	mgrimes
Rio	Grey	No	F	1,700	tswift
Robin	Yellow	No	M	1,100	jisbell
Katy	Brown	No	F	1,200	jisbell
Pegasus	Brown	No	M	1,750	mgrimes
Sammy	Black	Yes	M	2,200	mgrimes
Pinky	Pink	No	M	1,050	tswift
Hulk	Grey	No	M	2,050	mgrimes
Pat	White	No	F	1,400	mgrimes
Betty	White	Yes	F	1,250	tswift



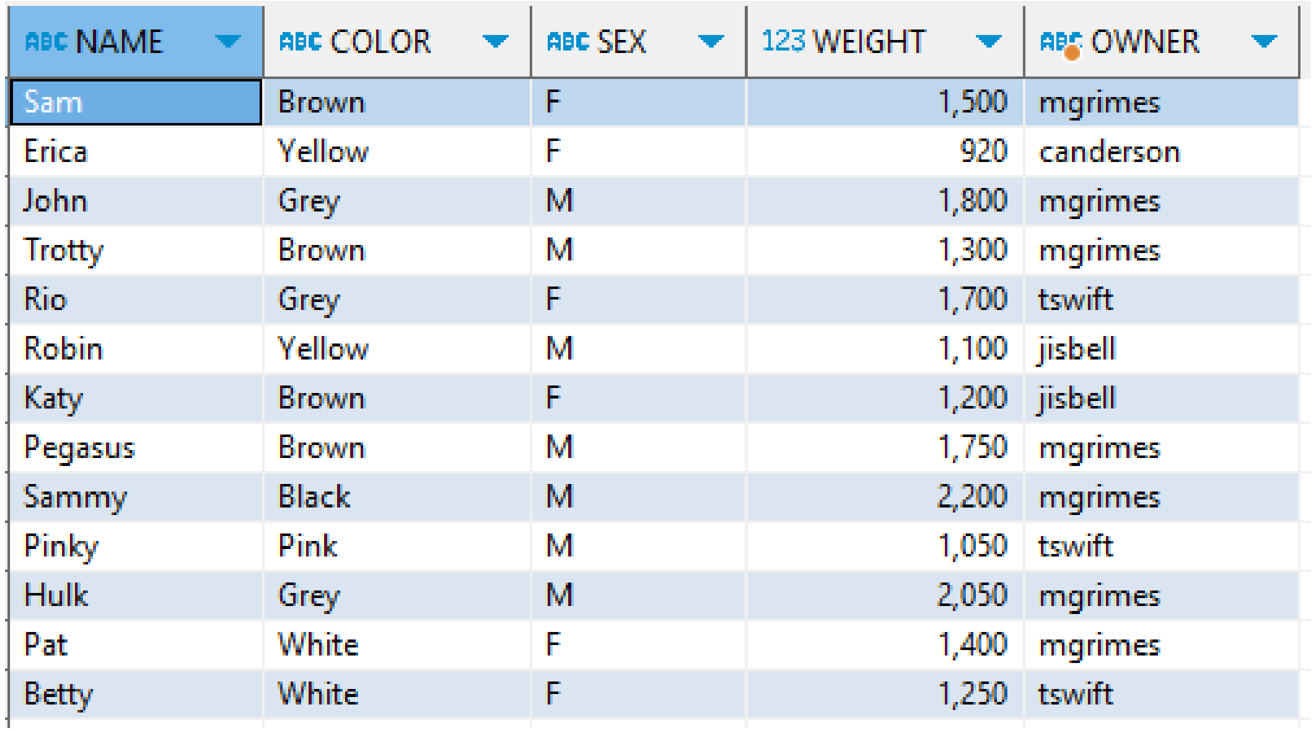
```
UPDATE horses SET color = 'Pink' WHERE name = 'Betty';
```

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

Altering your schema (CRUD)

- We can also get rid of attributes using the alter command
- If we no longer care about spots:

- `ALTER TABLE horses DROP column Spots;`

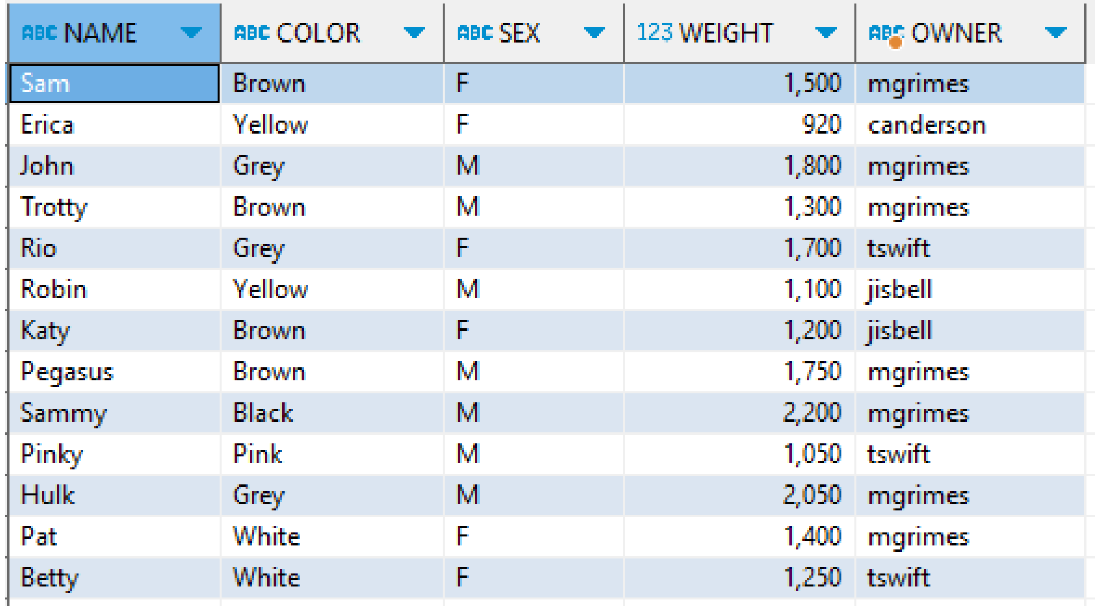


NAME	COLOR	SEX	WEIGHT	OWNER
Sam	Brown	F	1,500	mgrimes
Erica	Yellow	F	920	canderson
John	Grey	M	1,800	mgrimes
Trotty	Brown	M	1,300	mgrimes
Rio	Grey	F	1,700	tswift
Robin	Yellow	M	1,100	jisbell
Katy	Brown	F	1,200	jisbell
Pegasus	Brown	M	1,750	mgrimes
Sammy	Black	M	2,200	mgrimes
Pinky	Pink	M	1,050	tswift
Hulk	Grey	M	2,050	mgrimes
Pat	White	F	1,400	mgrimes
Betty	White	F	1,250	tswift

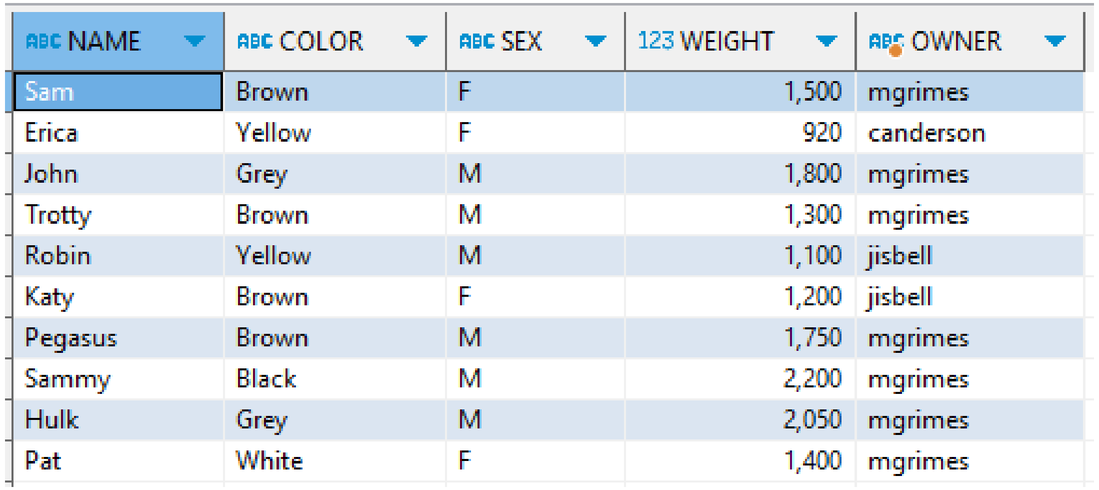
Deleting data (CRU**D**)

- We can delete data that matches a certain criteria using a WHERE clause
- If the owner “tswift” leaves and takes all her horses with her:

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



NAME	COLOR	SEX	WEIGHT	OWNER
Sam	Brown	F	1,500	mgrimes
Erica	Yellow	F	920	canderson
John	Grey	M	1,800	mgrimes
Trotty	Brown	M	1,300	mgrimes
Rio	Grey	F	1,700	tswift
Robin	Yellow	M	1,100	jisbell
Katy	Brown	F	1,200	jisbell
Pegasus	Brown	M	1,750	mgrimes
Sammy	Black	M	2,200	mgrimes
Pinky	Pink	M	1,050	tswift
Hulk	Grey	M	2,050	mgrimes
Pat	White	F	1,400	mgrimes
Betty	White	F	1,250	tswift

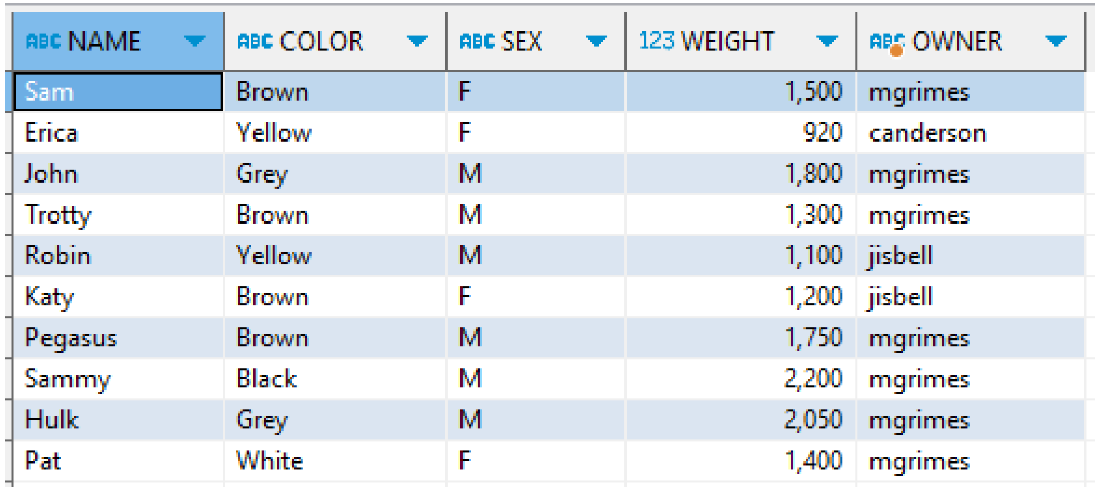


NAME	COLOR	SEX	WEIGHT	OWNER
Sam	Brown	F	1,500	mgrimes
Erica	Yellow	F	920	canderson
John	Grey	M	1,800	mgrimes
Trotty	Brown	M	1,300	mgrimes
Robin	Yellow	M	1,100	jisbell
Katy	Brown	F	1,200	jisbell
Pegasus	Brown	M	1,750	mgrimes
Sammy	Black	M	2,200	mgrimes
Hulk	Grey	M	2,050	mgrimes
Pat	White	F	1,400	mgrimes

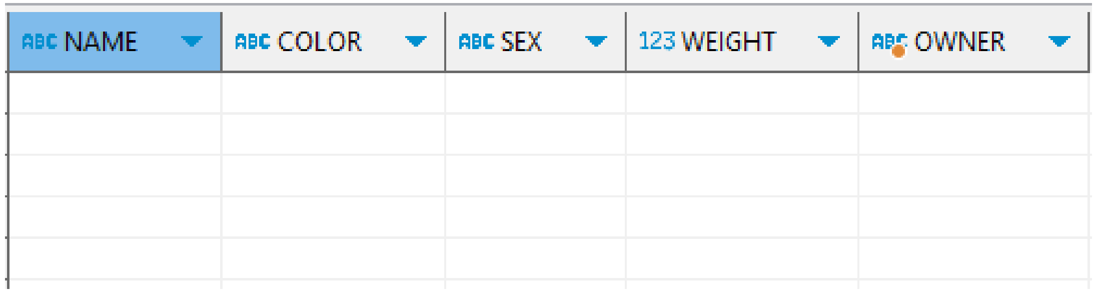
Deleting data (CRUD)

- If we do not use a WHERE, all data will be delete (but the table will remain in the database)

```
DELETE FROM horses;
```



NAME	COLOR	SEX	WEIGHT	OWNER
Sam	Brown	F	1,500	mgrimes
Erica	Yellow	F	920	canderson
John	Grey	M	1,800	mgrimes
Trotty	Brown	M	1,300	mgrimes
Robin	Yellow	M	1,100	jisbell
Katy	Brown	F	1,200	jisbell
Pegasus	Brown	M	1,750	mgrimes
Sammy	Black	M	2,200	mgrimes
Hulk	Grey	M	2,050	mgrimes
Pat	White	F	1,400	mgrimes



NAME	COLOR	SEX	WEIGHT	OWNER

- Finally, as we saw before, we can delete the table altogether with the DROP command, which will delete the schema along with any data that remains:

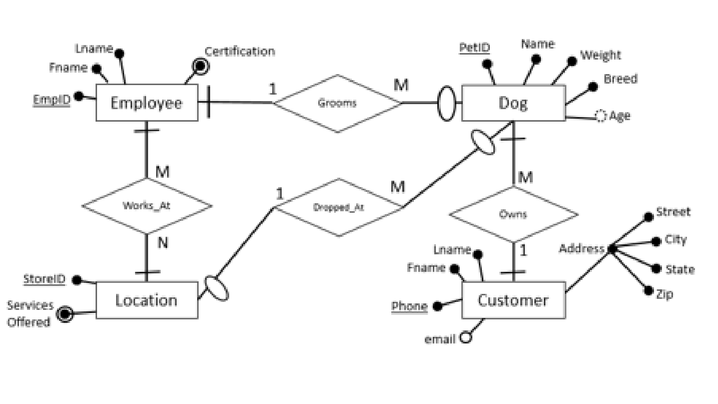
```
DROP TABLE Horses;
```

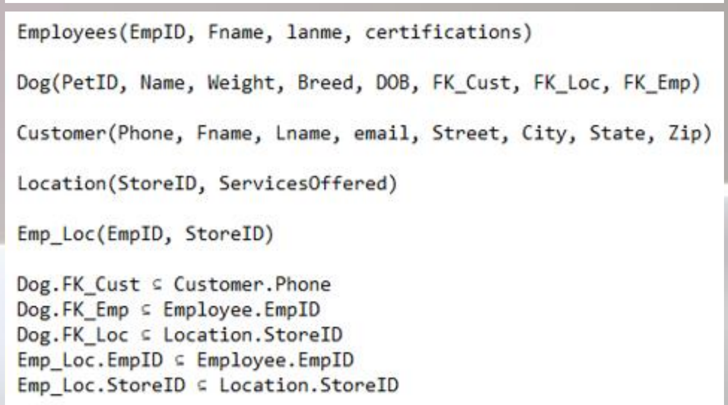
Module 10.1

Implementing Databases

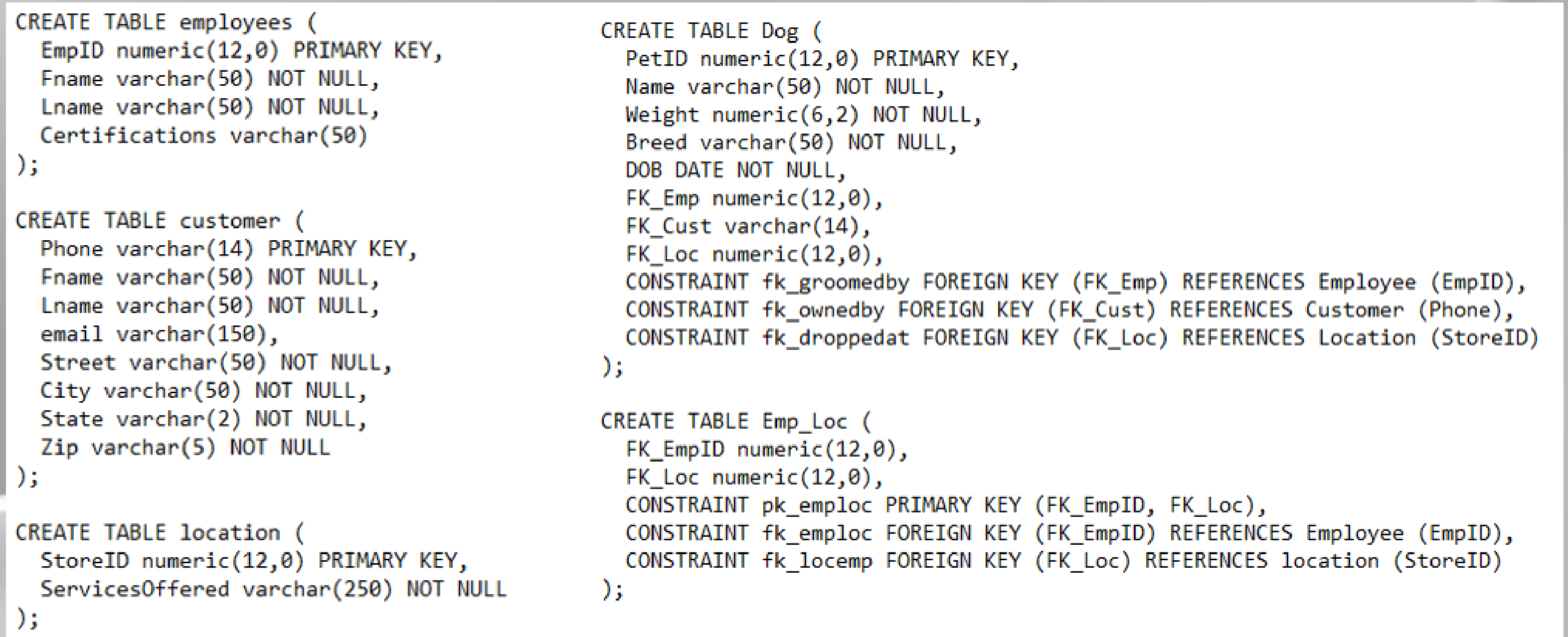
Next step in transforming business rules into an actual DB

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!

https://kahoot.it

- This Kahoot activity serves as a fun review/learning experience
- The same questions are on Canvas as a Progress Quiz
- You MUST complete the Progress Quiz on Canvas - this Kahoot does not count for points!

Go forth and do great things!

- Next week we will talk about relationships – lots of important content
- Make sure you do the Progress Quiz on Canvas before Friday at 6:00 PM
- Assignment 1 is due by 6:00 on Monday, February 5