

CCT395, Week 9

Review Databases and Objects

Yuri Takhteyev
University of Toronto
November 3, 2010



This presentation is licensed under Creative Commons Attribution License, v. 3.0. To view a copy of this license, visit <http://creativecommons.org/licenses/by/3.0/>. This presentation incorporates images from the Crystal Clear icon collection by Everaldo Coelho, available under LGPL from <http://everaldo.com/crystal/>.

Announcements

- Proposals & projects
- Schedule & readings
- The exam
- Next class

João's Coffee

The Entities

Coffee shop

Representative

Shipper

Estate

Coffee

Batch

Warehouse

Exchange Rate

Invoice

And there is probably more!

Divide and Conquer

Grouping João's concerns:

- Getting customers the right coffee
- Tracking deliveries
- Tracking money

Divide and Conquer

Getting the customers the right coffee

- Ordering the right coffee
- Delivering the coffee to the right customers

The Coffee

- Estate
 - name, description
- Region
 - name, description
- Variety
 - name, description
- Request
 - quantity, what kind, who requested
- Shipment
 - quantity, what kind of coffee

Region

name
description

Estate

name
description

Variety

name
description

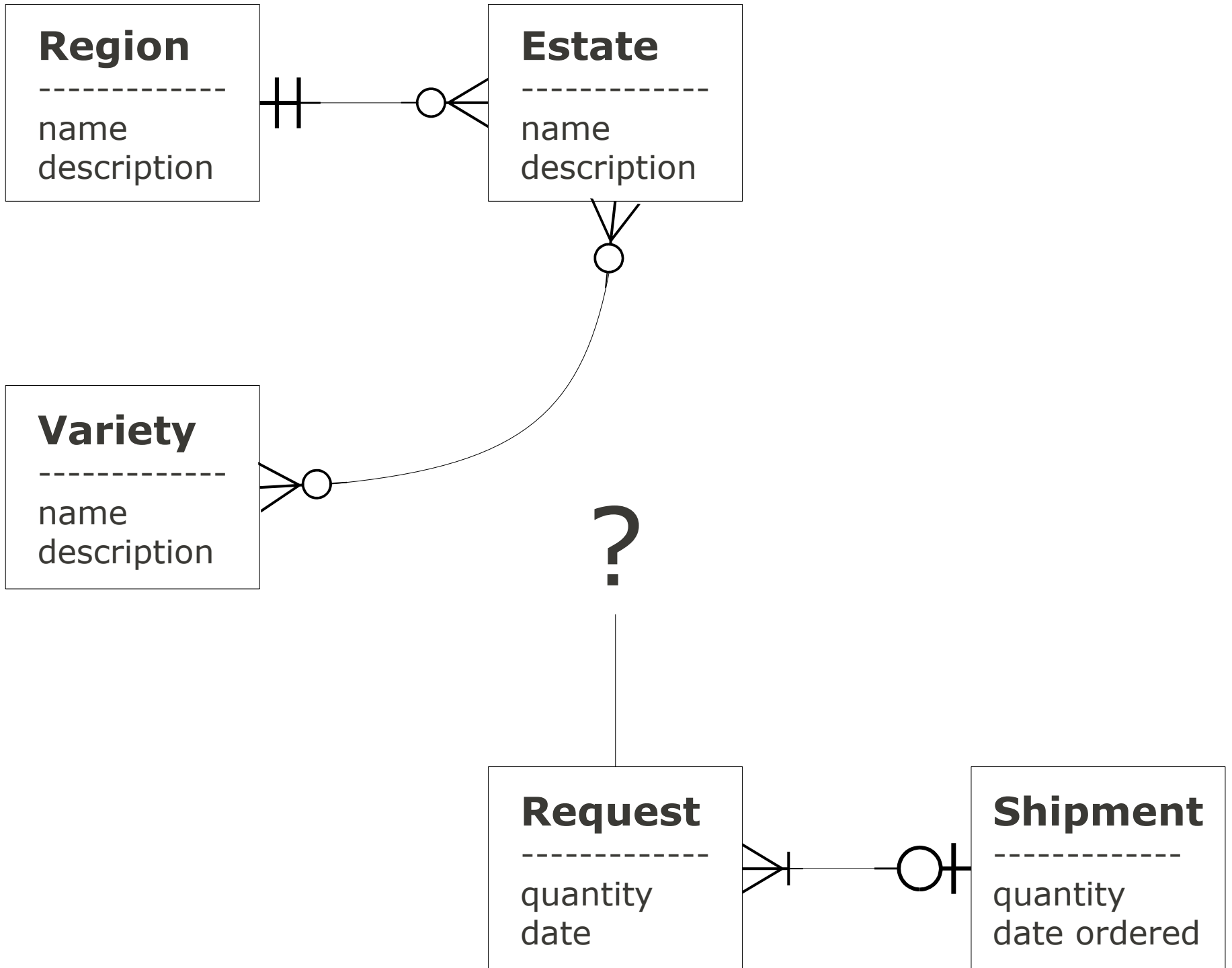
Request

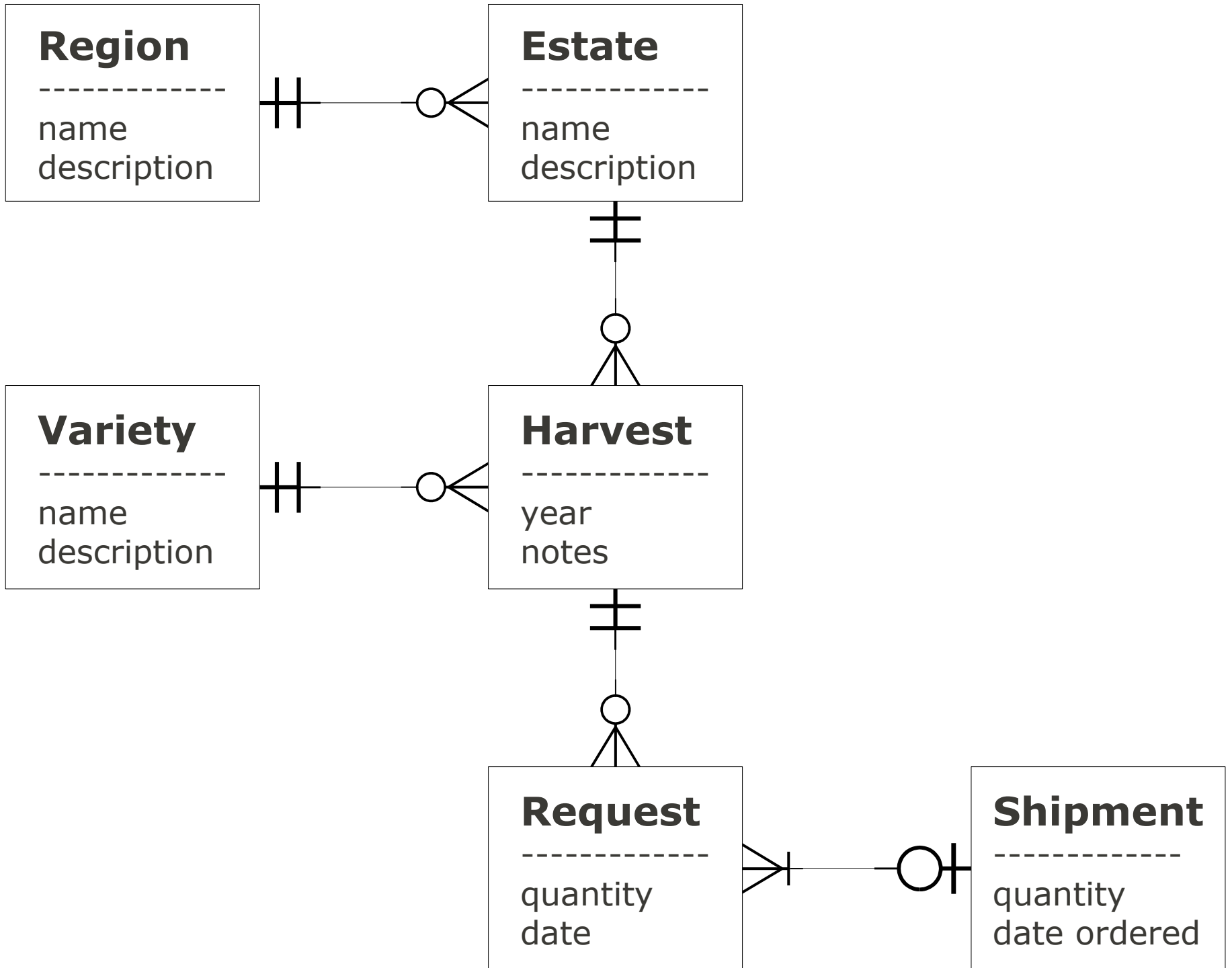
quantity
date

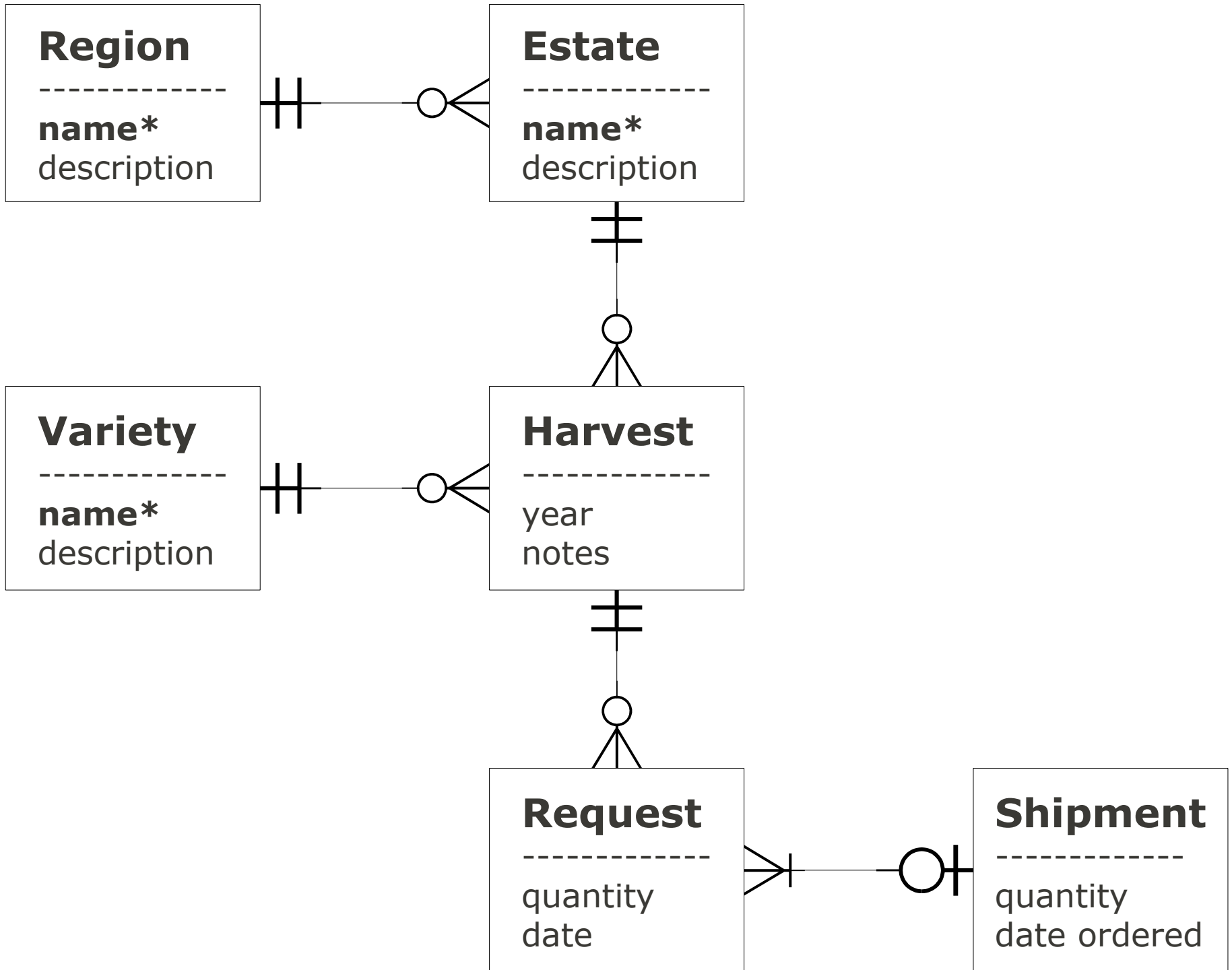
Shipment

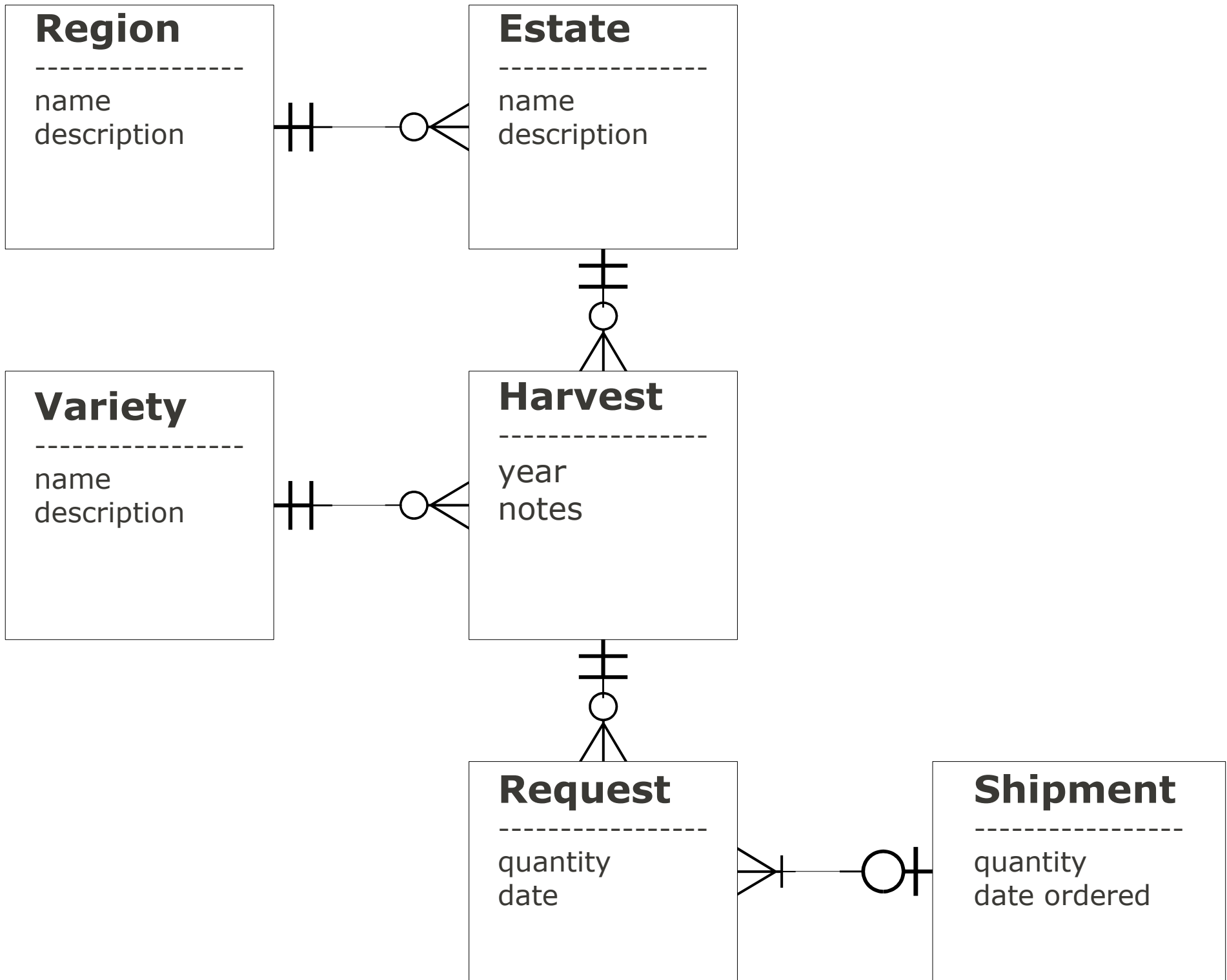
quantity
date ordered

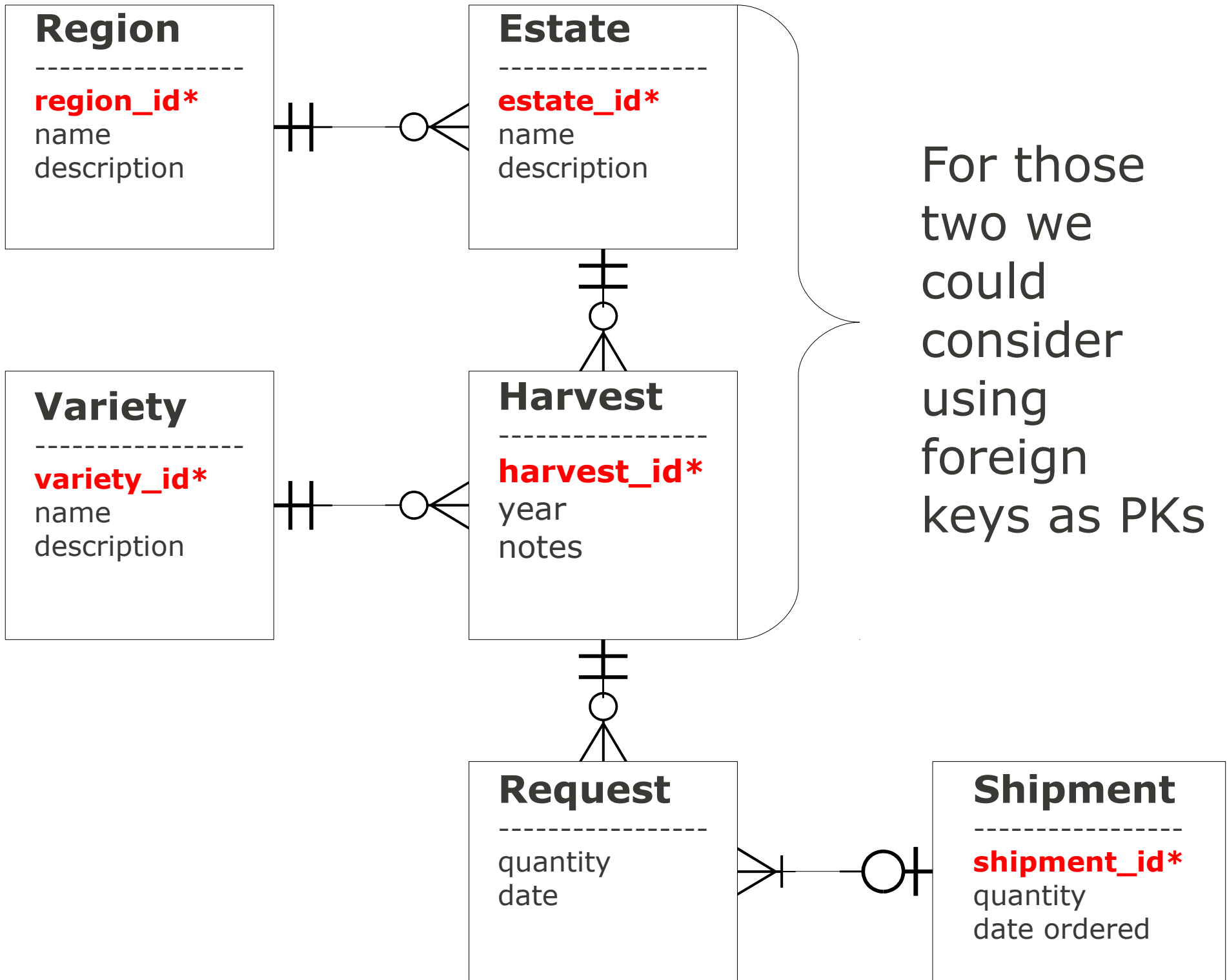
let's leave for later
whose request it is

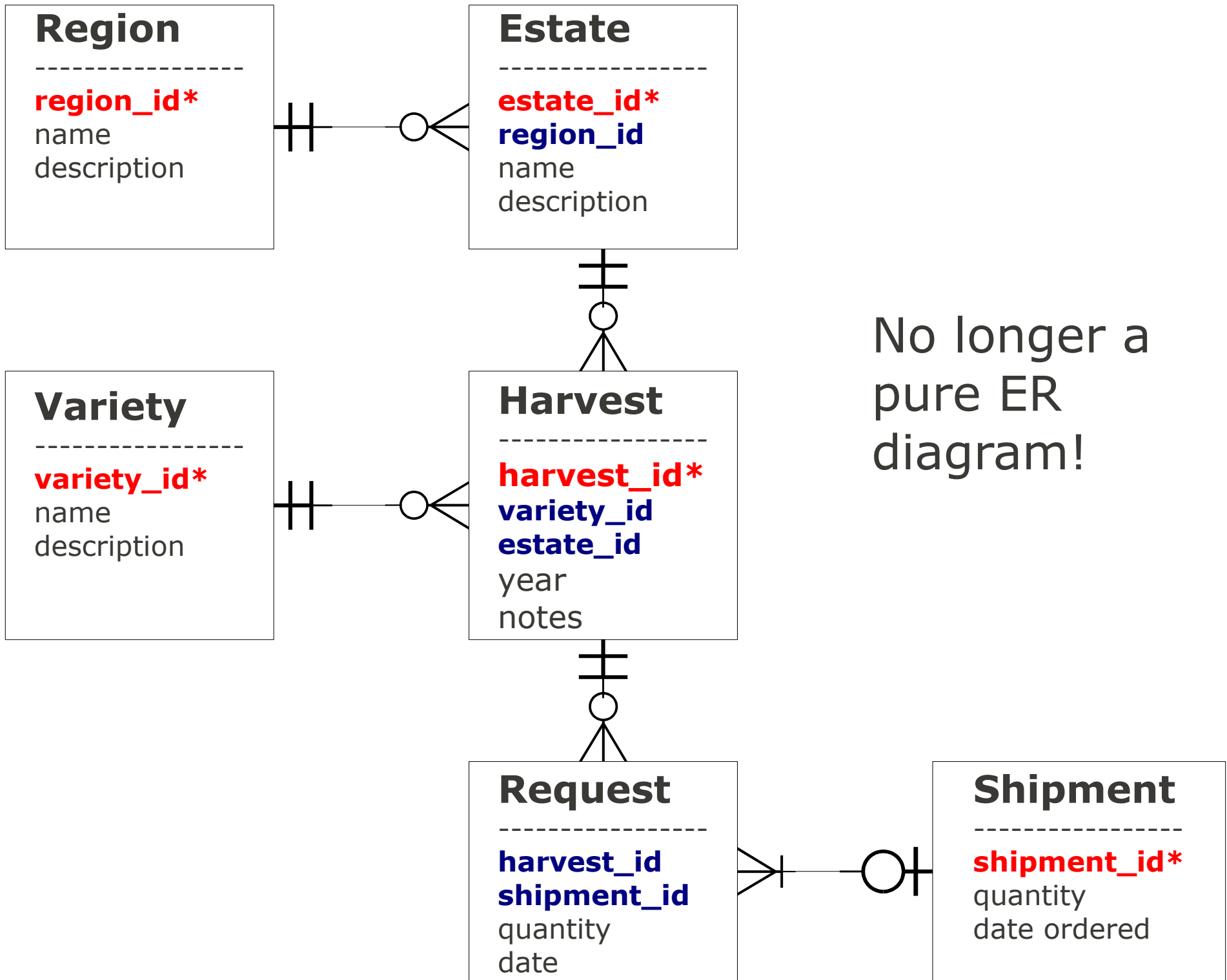












No longer a pure ER diagram!

A Relation

harvest (**harvest_id***, **variety_id**,
estate_id, year, notes)

~~**harvest** (**harvest_id***, year, notes)~~

A Relation

```
harvest (  
    harvest_id*,  
    variety_id,  
    estate_id,  
    year,  
    notes  
)
```


SQL

```
create table harvest (  
    harvest_id int,  
    variety_id int,  
    foreign key (variety_id)  
        references variety(variety_id) ,  
    estate_id int,  
    foreign key (estate_id)  
        references estate(estate_id) ,  
    year year,  
    notes varchar(200) ,  
    primary key (harvest_id) ,  
);
```

to be continued

Projects

- PHP Resources
 - <http://www.w3schools.com/php/default.asp>
 - the Menagerie Web example
- Get the **database** done first
- Do **basic** features before the advanced
 - The 6th bullet point of section 2.2 is now optional
- Advanced features
 - just do **two**
- Don't hesitate to ask questions!

Advanced Features

- Login
 - Store the password in the database
 - Ideally, use `md5()` to store a “hash” of a function
 - Have the user enter a username and password
 - Check what the user's password should be
 - Compare this with what they entered
 - If using `md5()`, compare the hashes.
 - If you can, use PHP sessions to remember the user. Otherwise, just have them enter the password every time.

Advanced Features

- Enabling Modifications
 - Do an INSERT or UPDATE based on the form data
- Backup
 - mysqldump is the simplest option
- XML Output
 - See examples from last week's class
 - Make sure your XML validates:
 - http://www.w3schools.com/xml/xml_validator.asp

Advanced Features

- Transactions
 - Chapter 13 in SQL – we are *not* covering this
- Stored Procedures
 - SQL chapter 14. Again, we are *not* covering this
- Publicly Available Data
 - I.e., using “real” data

Objects

Models

Relational:

- good for retrieval and updates

Document:

- good for exchange of data

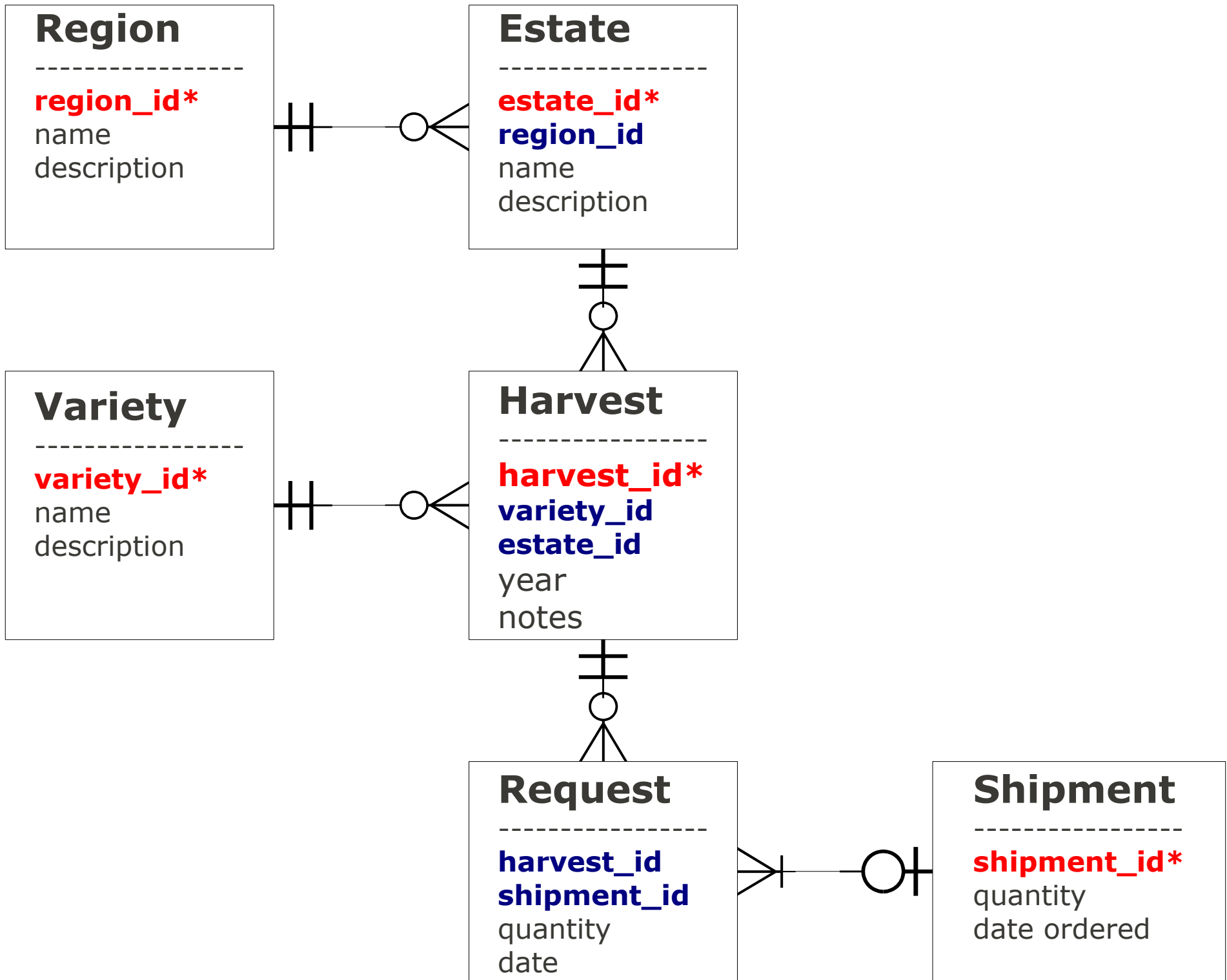
Objects:

- good for active manipulation
- good for changing software

Encapsulation

“Need to know” basis

(fewer assumptions →
easier to make changes)



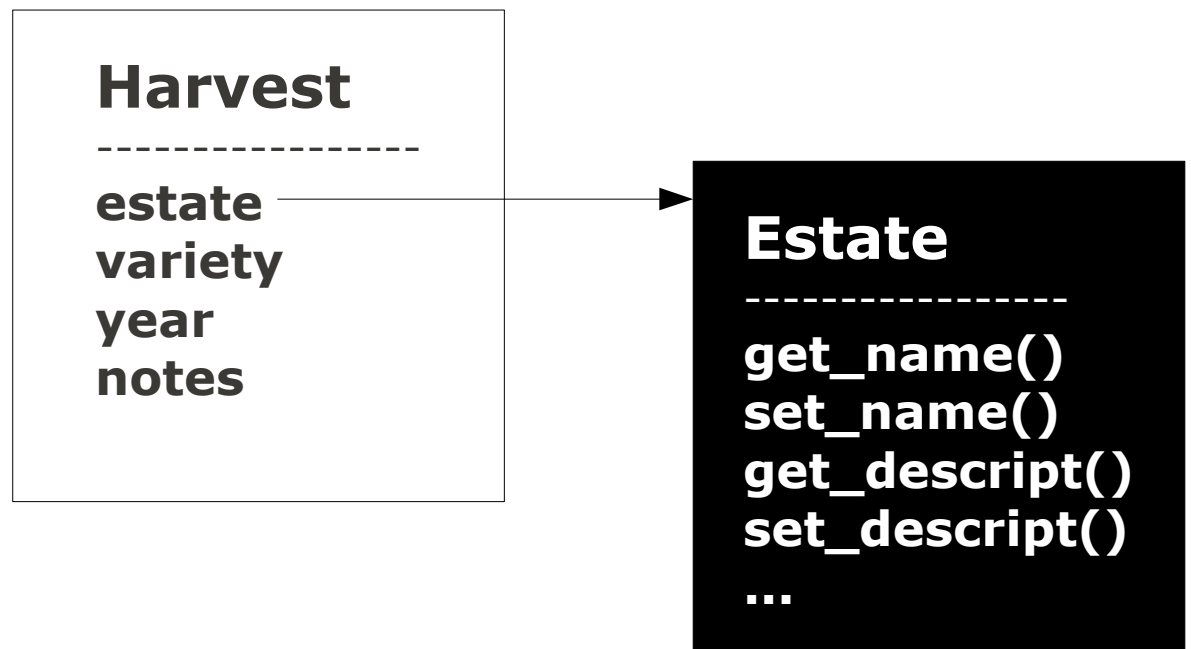
Messages

aka "methods"

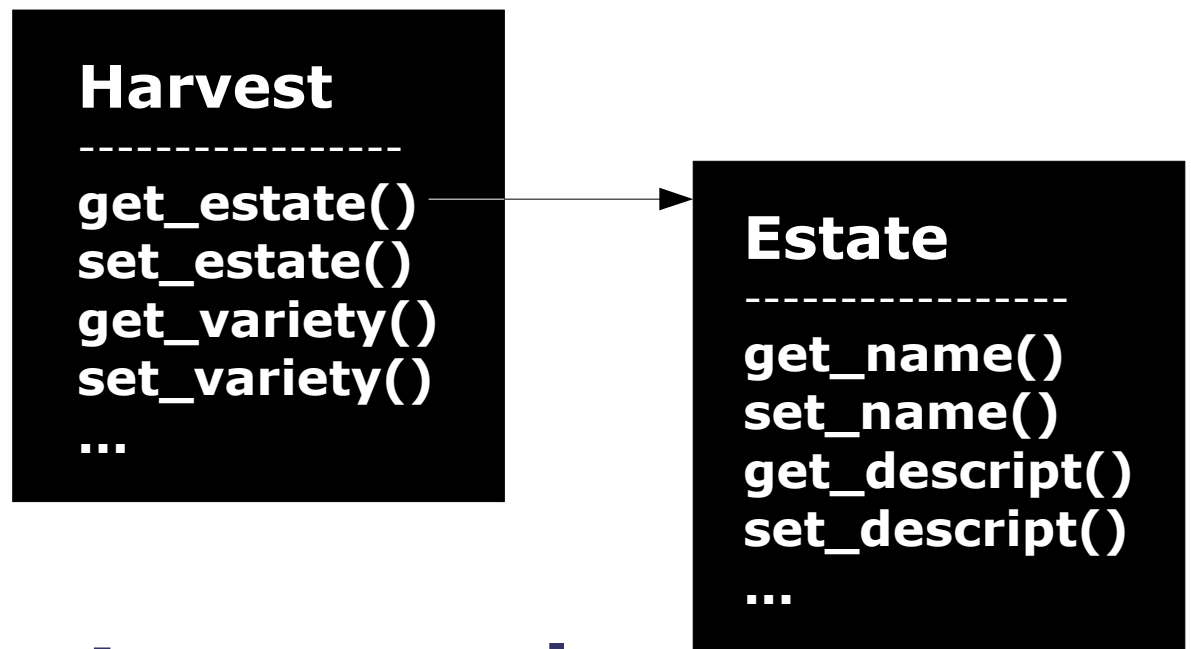
Harvest

get_estate()
set_estate()
get_variety()
set_variety()
...

The Implementation



```
$harvest = $order->getHarvest();  
$estate = $harvest->getEstate();  
$estate->setName("Boa Vista");
```



Object-Oriented Programming (OOP)

PHP with MySQL

```
$query = "select name, species from pet where owner='"  
        . $owner . "'";  
$result = mysql_query($query);  
while ($row = mysql_fetch_array($result)) {  
    echo "<tr>";  
    echo "<td>" . $row['name'] . "</td>";  
    echo "<td>" . $row['species'] . "</td>";  
    echo "</tr>";  
}
```

Don't forget the PHP Tutorial:

<http://www.w3schools.com/php/default.asp>

Updating the Data

```
$query = "update pet set species='"  
        . $species . "' where name='"  
        . $name . "'";  
$result = mysql_query($query);  
echo "ok";
```

Select and Update

```
$query = "select name, species from pet where owner='"  
        . $owner . "'";  
$result = mysql_query($query);  
while ($row = mysql_fetch_array($result)) {  
    $name = $row['name'];  
    echo "Let's change the species of ";  
    echo $name . " from " . $row['species'];  
    echo " to " . $new_species;  
    $update_query = "update pet set species='"  
                    . $new_species . "' where name='"  
                    . $name . "'";  
    $update_result = mysql_query($query);  
}
```


What We Want

```
$pets = $menagerie->get_pets_by_owner($owner);  
while ($pet = $pets->next()) {  
    echo "Let's change the species of ";  
    echo $pet->get_name() . " from " . $pet->get_species();  
    echo " to " . $new_species;  
    $pet->set_name($new_species);  
    $menagerie->save();  
}
```

Or Even

```
$pets = $menagerie->get_pets_by_owner($owner);  
while ($pet = $pets->next()) {  
    echo "Let's change the species of ";  
    echo $pet->name . " from " . $pet->species;  
    echo " to " . $new_species;  
    $pet->name = $new_species;  
    $menagerie->save();  
}
```

1. Easier / more intuitive
2. Encapsulation simplifies reuse

Too good to be true?

Object-Relational Mapping

Option 1:

Map objects to relations
(code → database)

Option 2:

Map relations to objects
(database → code)

Option 2

Manual:

We setup a mapping

```
pet.name → getName($name)  
          setName($name)
```

Automatic:

The software figures it out
("introspection")

Examples

Rails / ActiveRecord (Ruby)

Django (Python)

Hibernate (Java)

Doctrine (PHP)

Questions?