7. Learning from Data Model

Fall 2018

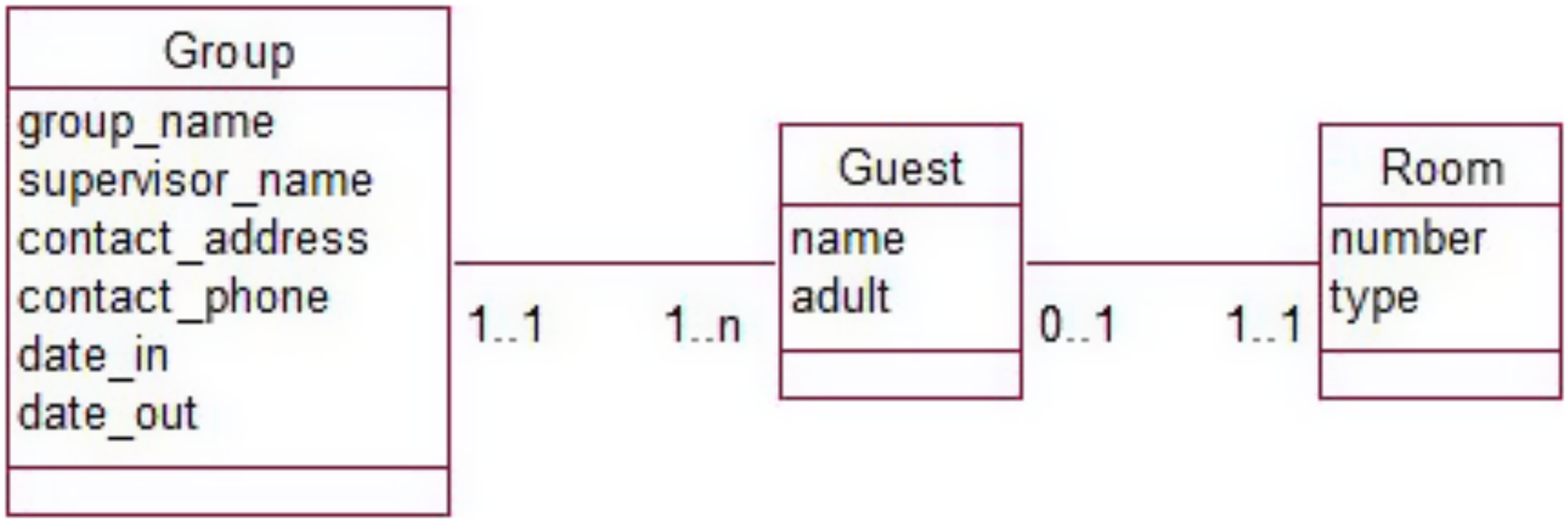
- We look more closely at the data model to see how it can further our understanding of a database system.
- A data model is a precise description of the data stored for a real-world problem,
- The data model is neither a complete nor an exact description of a real situation. It will always be based on definitions and assumptions, and it has a finite scope.
- It is a model of the relationships among the data items that are being stored about a problem, but it is not a complete model of the real problem itself.
- Constraints on money, time, and expertise will always mean that problems will need to be scoped and assumptions made in order to extract the essential elements. It is crucial that the definitions and assumptions are clearly expressed so that the client and the analyst are not talking at cross-purposes.
- We look at how the initial data model can be used to discover where definitions and scope may need to be more rigorously expressed.

Today's Lecture

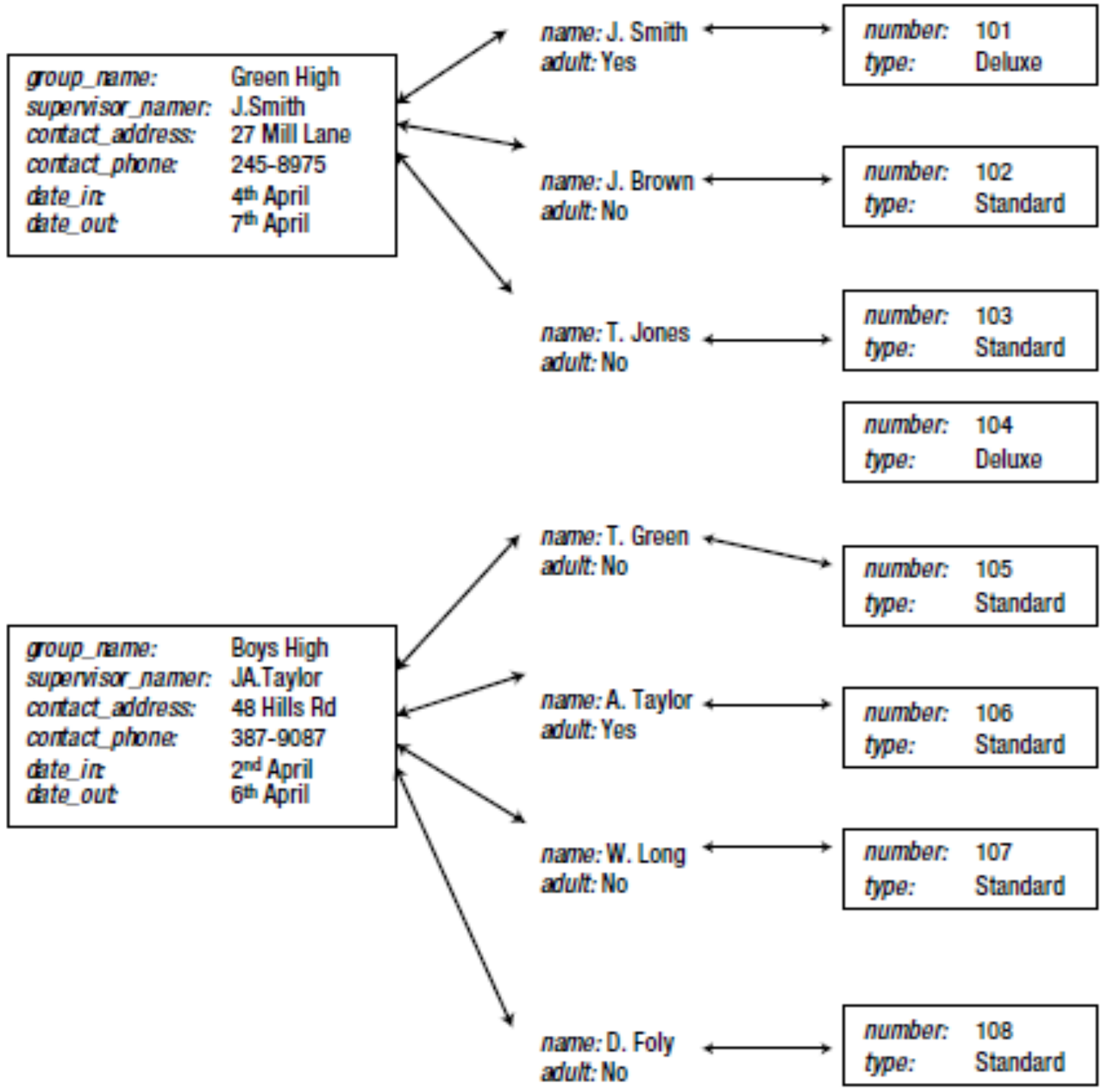
- Review of Data Model
- Optionality: Should It Be 0 or 1?
- A Cardinality of 1: Might It Occasionally Be Two?
- A Cardinality of 1: How About Historical Data?
- A Many-Many: Are We Missing Anything?
- Exercises
- Summary

Review of Data Models

- We will revisit the essential aspects of a data model by way of an example that will highlight some additional features.



- Groups consist of a number of guests, and each guest has a room. Rooms are for one guest only, and they may not all be full. Some possible instances of these objects and relationships are



- What is the definition of a *group* for this hostel?
- It is not a set of people who all know each other and feel as though they belong together, but a set of people with the same arrival and departure dates and common contact information.
- What would we do if A. Taylor and W. Long need to leave the Boys High group a day early? How could we record this information?
- If it is essential, however, that the system needs to record that these two groups of Boys High people are somehow “together,” the data would need to be modeled differently.
- The data model also tells us that a guest must belong to a group. What else does this tell us about the definition of a group?
- We can have a group with just one guest.

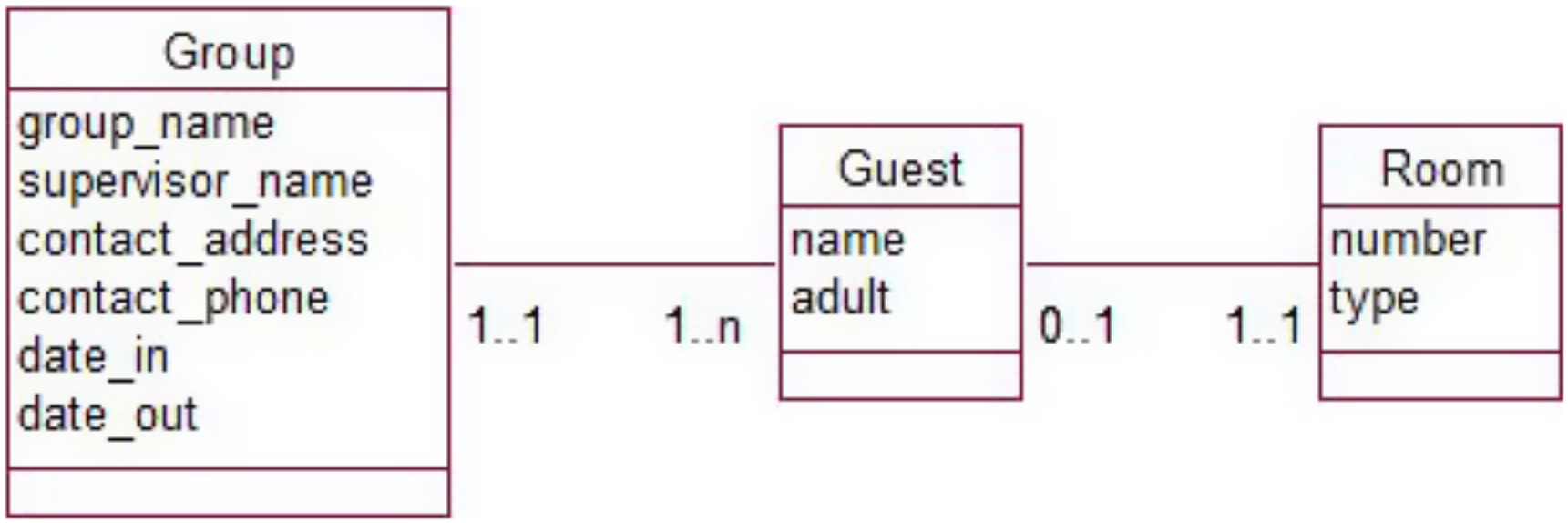
A group is a set of guests with common contact information and with identical arrival and departure dates. A separate group will need to be formed for each different set of arrival and departure dates. A group can have one or more guests associated with it. We can have a group with just one guest.

The questions we will look at only apply to relationships between two classes, but they can open up a great deal of discussion about the problem. As more is understood about a problem, what we learn from the data model can be reflected in the use cases.

- **Optionality**: Should it be 0 or 1?
- **Cardinality of 1**: Might it occasionally be 2?
- **Cardinality of 1**: What about historical data?
- **Many-Many**: Are we missing anything?

Optionality: Should It Be 0 or 1?

The optionality of one end of a relationship is the smallest number of objects that can be associated with an object at the other end. This is usually 0 or 1. For example reading the relationship between guest and room from left to right, we have that a particular guest must be associated with a room (optionality 1), whereas reading the relationship from right to left, we see that a particular room may have no related guest (optionality 0).

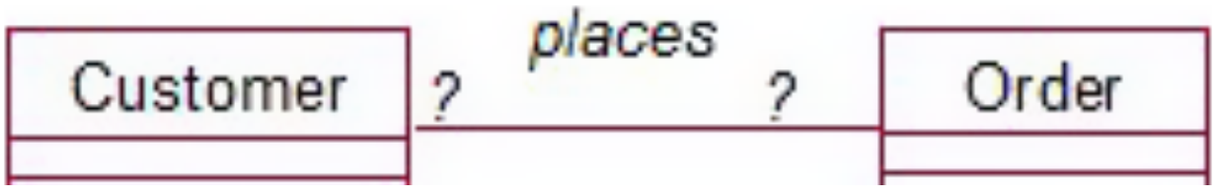


Student Course Example

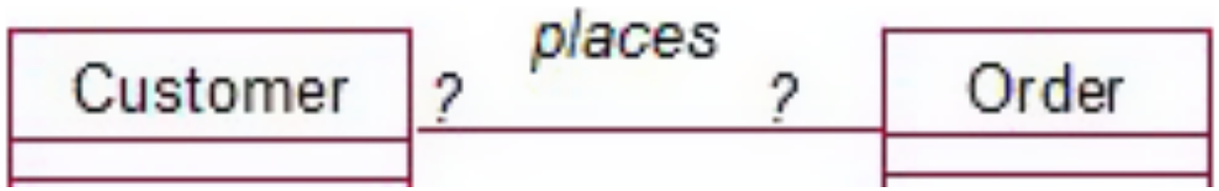
- A student can enroll in many courses, and a course can have many students enrolled in it. What about the optionalities? Can a student be enrolled in no courses? Our normal conversational definition of a student is someone who is studying or is formally enrolled in a course.
- What is our definition of student for this database?
- We can accommodate this situation by expanding our definition of a student to include people accepted by and/or registered with the university.

- What about a person who has contacted the university and asked to be sent information about enrollment?
- “Exactly who are these people you call students?” are considered right at the start of the analysis process.
- How careful consideration of the details of even the most simple data model can lead to important questions about much wider aspects of the problem.
- How we define a course? What data might we want to keep about a course?

Customer Order Example

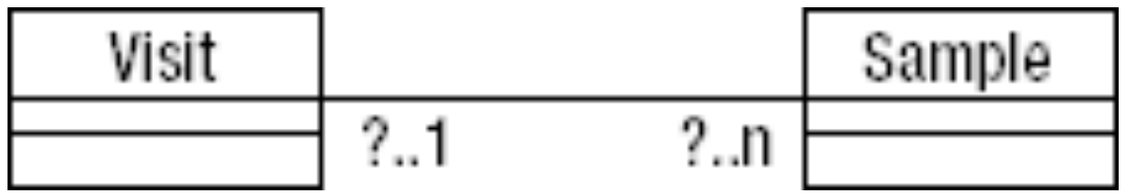


- We keep information on customers and the orders they place. Our first instinct is to say that customers can place many orders and each order is placed by one customer.
- What about the optionalities?
- Can a customer be associated with no orders?
- *Anyone who has ever placed an order and other people who are to be sent catalogs*
- “Do you want to be able to identify people who have previously placed orders but who are now fed up with being sent catalogs?”



- We want to know whether each order must have an associated customer.
- If an order arrives in the mail with no name or address,
- We can insist that every order must have a customer (optionality 1).
- We are not trying to solve any of these issues just now. We are simply using the data model to make us think clearly

Insect Example



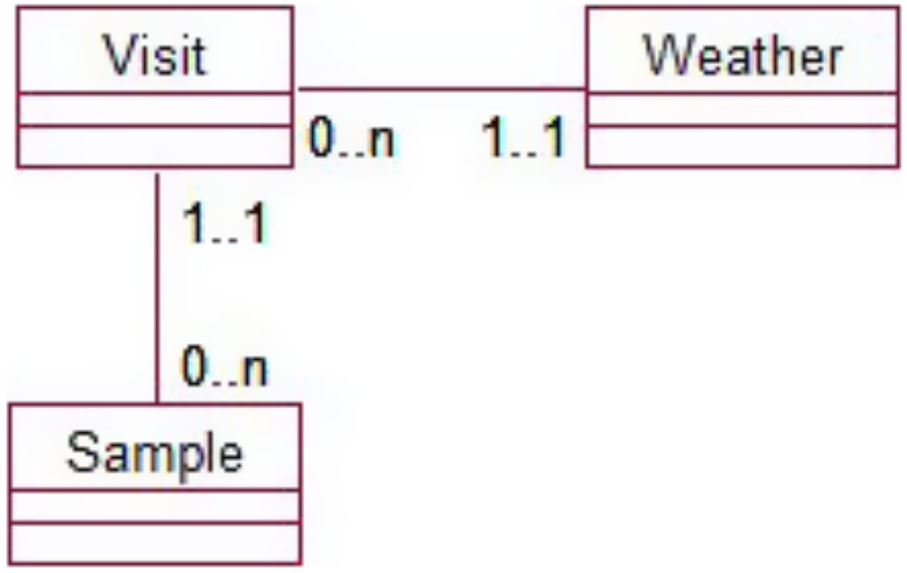
- Farms were visited and several samples of insects were collected. A Visit object would contain information about the date and conditions of a particular visit and would be associated with several *Sample* objects. Each sample object would contain information about the number of insects collected.
- Whether a sample *must* be associated with a visit?
- whether each visit *must* have an associated sample?

A Cardinality of 1: Might It Occasionally Be Two?

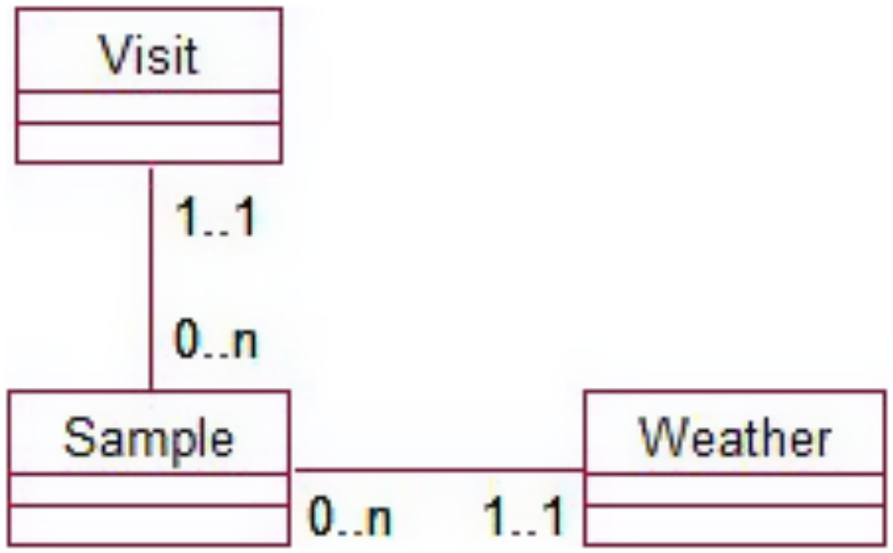
- Think carefully about different scenarios to ensure that the database will be able to cope adequately with all the data that may eventuate. Some “exceptions” are really complications that have been overlooked. Real life and real problems are always complicated.
- How to deal with “exceptions” that do not warrant a complete overhaul of the problem but nevertheless are likely to turn up during the lifetime of the database.

Insect Example

- To record the weather conditions consistently, the scientist may decide to choose from one of a number of categories. Introducing a Weather class with objects for the different conditions (e.g. fine, overcast, raining) can ensure that this information is recorded consistently.



- The conditions under which each individual sample is collected may be vital. In this case, it might be more sensible to associate each sample with its own weather condition.
- This latter solution may be overkill when the majority of visits have stable weather conditions.
- If the weather changes markedly, we will create another visit. This way all visits have a single associated weather type, and we can cope with the "exceptional" case by redefining what we mean by a visit.
- *A visit is a time spent on a farm during constant weather conditions on a single day. It is possible to have more than one visit to a farm per day.*



Sports Club Example

- A local sports club may want to keep a list of its membership and the team for which each member currently plays (SeniorB, JuniorA, Veteran, etc.). One way to model this data is shown.

- We should still ask questions about the maximum number of teams with which a member might be associated.
- “Can a member play for more than one team and, if so, do we care?”.
- The relationship plays for means a player’s main team rather than just any team they may play for.
- The data model clearly does not **allow** for historical records to be kept.
- The situation where injury or sickness necessitates a member of one team filling in for another team for a particular match. How will this affect the data model? This is a question of scope.

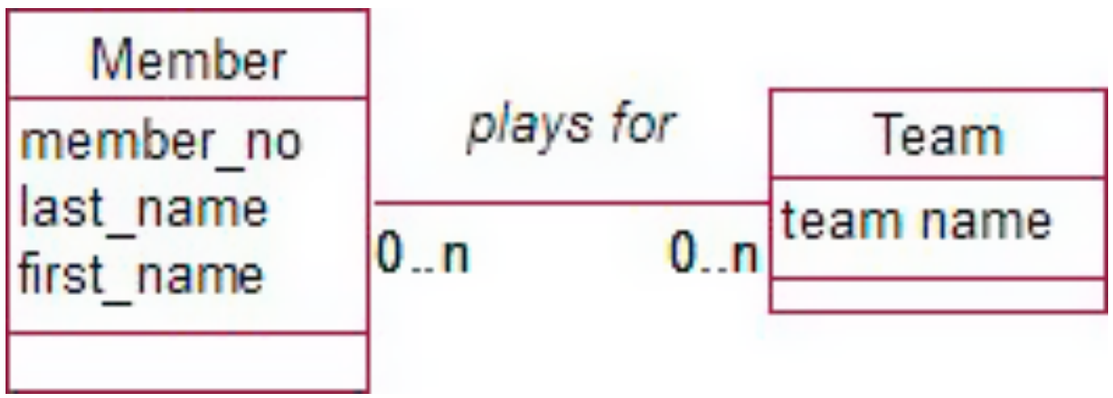
A Cardinality of 1: What About Historical Data?

- We have had a number of examples of relationships with a cardinality of 1 at one end.
- We have been careful to add the word currently because over time a room will have many guests and a player many teams.
- "Do we want our system to keep track of previous guests or previous team affiliations?"
- A sports club will find its system just fine for the first season but may get a surprise when the next year's teams replace the previous ones, which are then lost forever.

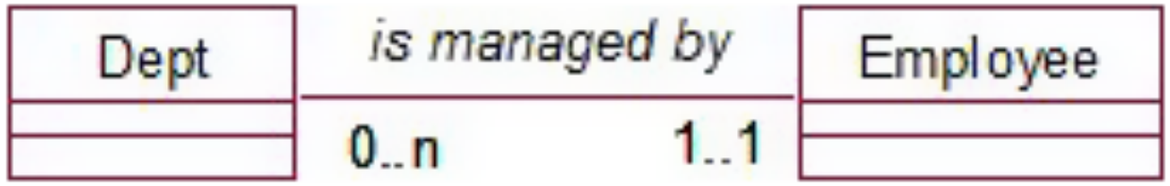
Sports Club Example

member_no	last_name	first_name	team
152	Abell	Walt	SeniorB
103	Anderson	James	JuniorA
276	Avery	Graeme	JuniorA
287	Brown	Bill	JuniorA
298	Burns	Lance	Veteran

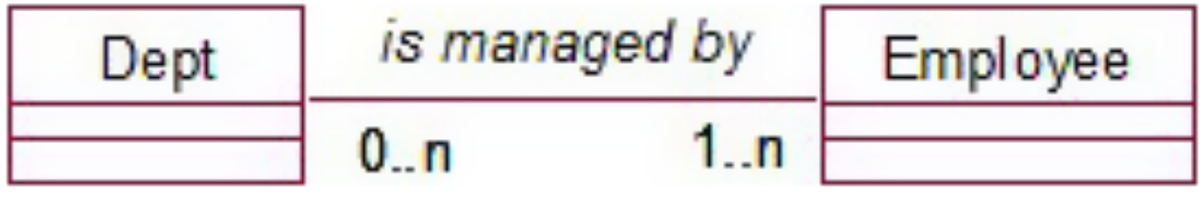
- The following season when Bill Brown graduates to the SeniorB team, his previous association with the JuniorA team will be lost. If the historical data are important, the problem must be remodeled to reflect the fact that members will be associated with many teams over time.



Departments Example

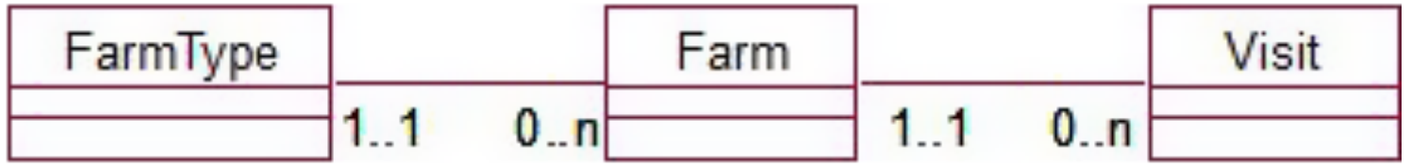


- "Do we want to keep track of former managers?"
- If we want to know who was in charge when something went wrong last year, we will need to keep a history.



Insect Example

- We need to know that the main objective of this long-term project was to see how the numbers of insects change as farming methods evolve over the years. The farms selected represented different farming types (organic, cropping, etc.).
- Throughout the duration of the project, each farm was visited several times to collect samples.



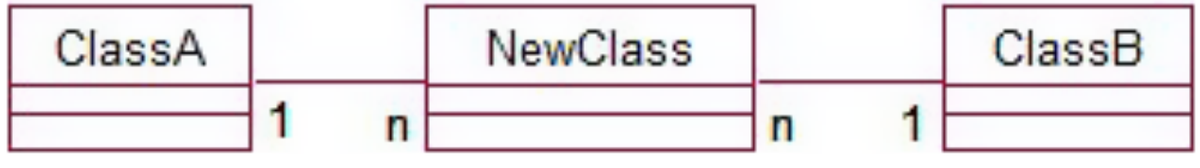
- This was only because the farming types had not changed during the time the project had been running. Throughout the duration of the project, each farm was visited several times to collect samples.
- When a farm did eventually change, the previous farming type would be lost.
- A farm can only be associated with one farming type at a time.
- “Might the type change over time, and is it important for the system to record that historical data?”

A Many–Many: Are We Missing Anything?

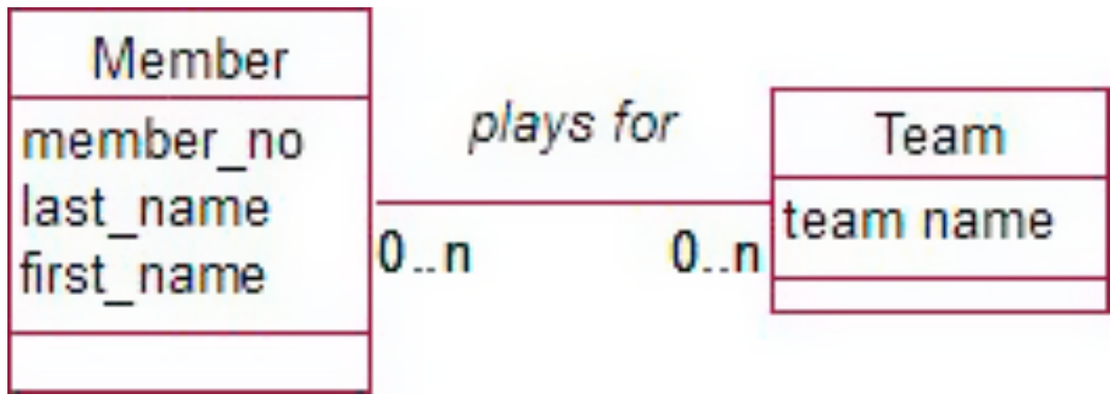
- If we widen the scope of some of the examples to include historical data, a number of 1-Many relationships will become Many-Many relationships (i.e., departments may have many managers, members many teams, and farms many types over a length of time).
- We need to keep some additional information about a Many-Many relationship.
- The historical data will not be of much use without a date attached somewhere. But where will the date go?

We need to ask the question:

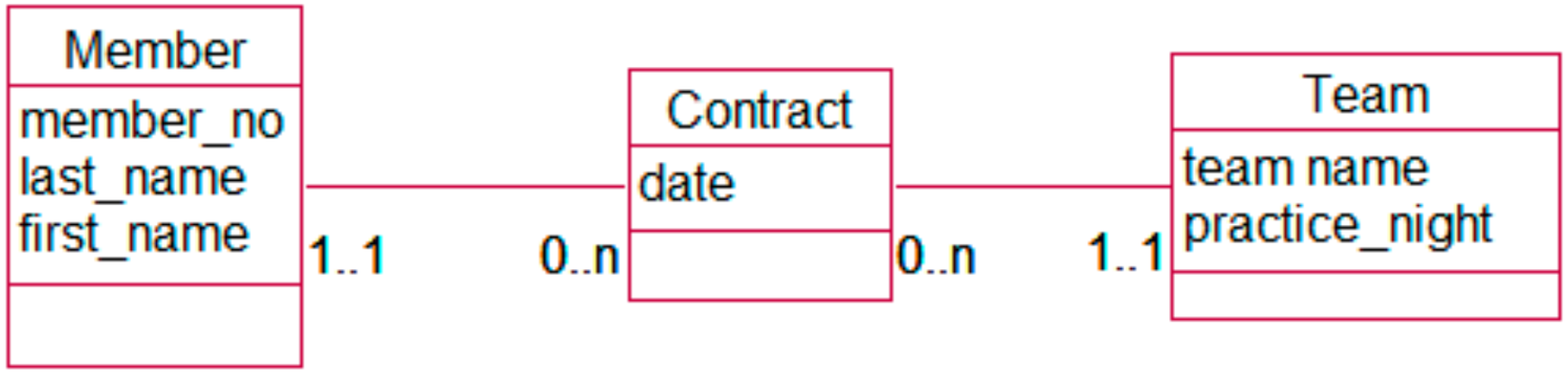
- *Is there any data that we need to record that depend on particular instances of each of the classes in our Many-Many relationship?*
- *Is there any data that depend on a particular player and a particular team?*
  - *Yes—the dates that player played for that team*



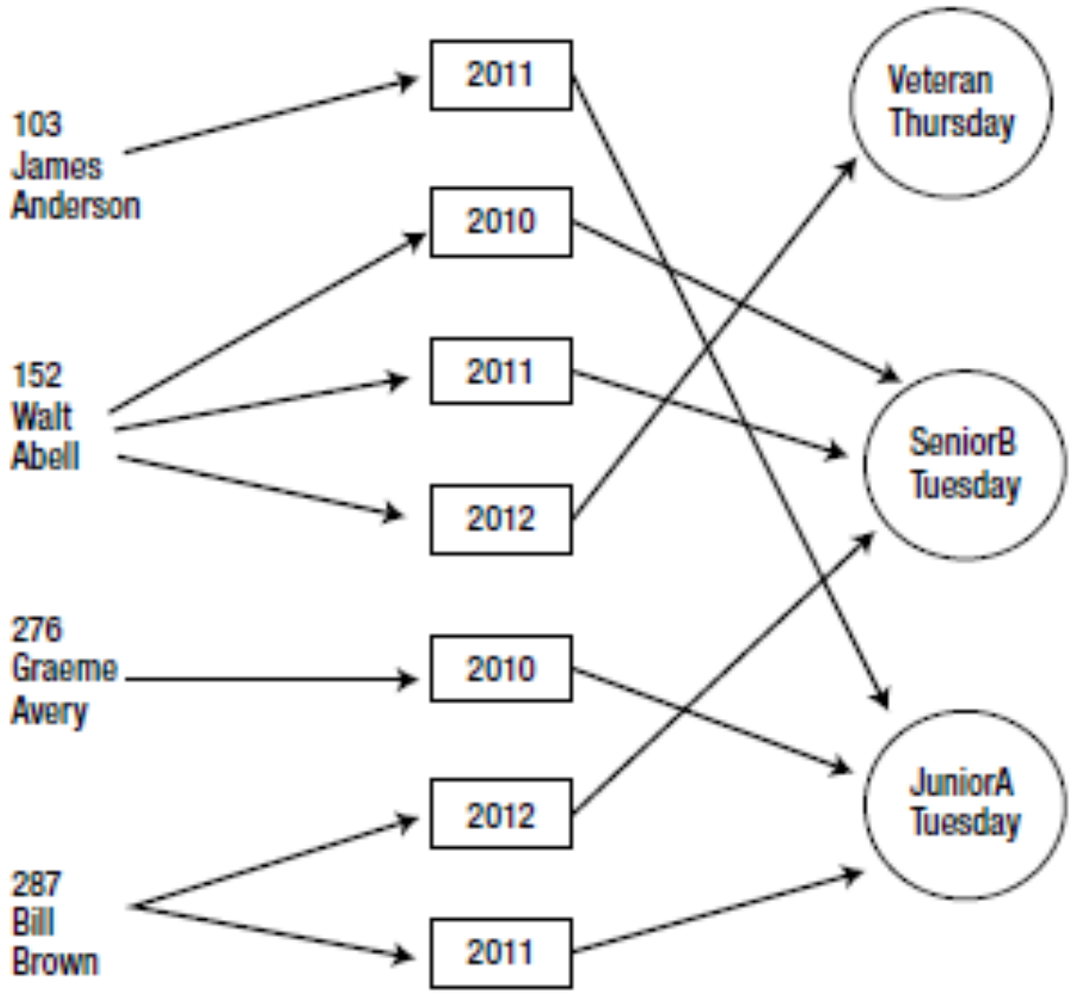
Sports Club Example



- The date that a particular member plays for a particular team cannot live in the **Member** class (because a member will play for many different teams over time) nor can it live in the **Team** class.
- Introduces a new intermediate class, **Contract**.



- Each contract is for exactly one team and exactly one member.
- Each member can have many contracts as can each team.


103
James
Anderson
152
Walt
Abell
276
Graeme
Avery
287
Bill
Brown
2011
2010
2011
2012
2010
2012
2011
Veteran
Thursday
SeniorB
Tuesday
JuniorA
Tuesday

member_no	last_name	first_name
152	Abell	Walt
103	Anderson	James
276	Avery	Graeme
287	Brown	Bill
298	Burns	Lance

Member

team_name	practice_night
JuniorA	Tuesday
SeniorB	Tuesday
Veteran	Thursday
Under 18	Monday

Team

member	team	year
103	JuniorA	2011
276	JuniorA	2010
287	JuniorA	2011
287	SeniorB	2012
152	SeniorB	2010
152	Veteran	2012
152	SeniorB	2011

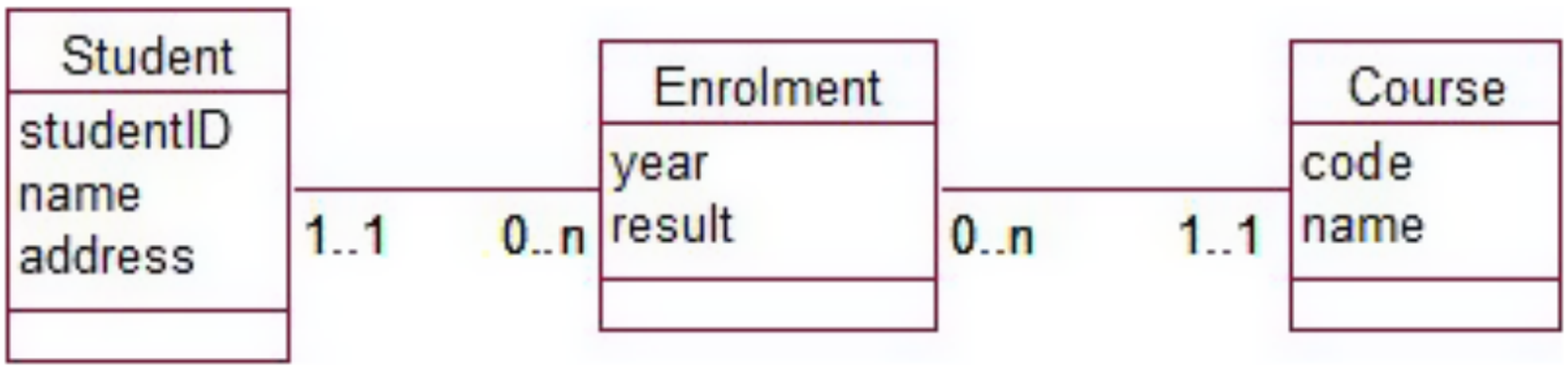
Contract

Student Course Example

- The Many-Many relationship of students enrolling in courses isn't just a historical problem, although we clearly will want to know when the student completed the course.

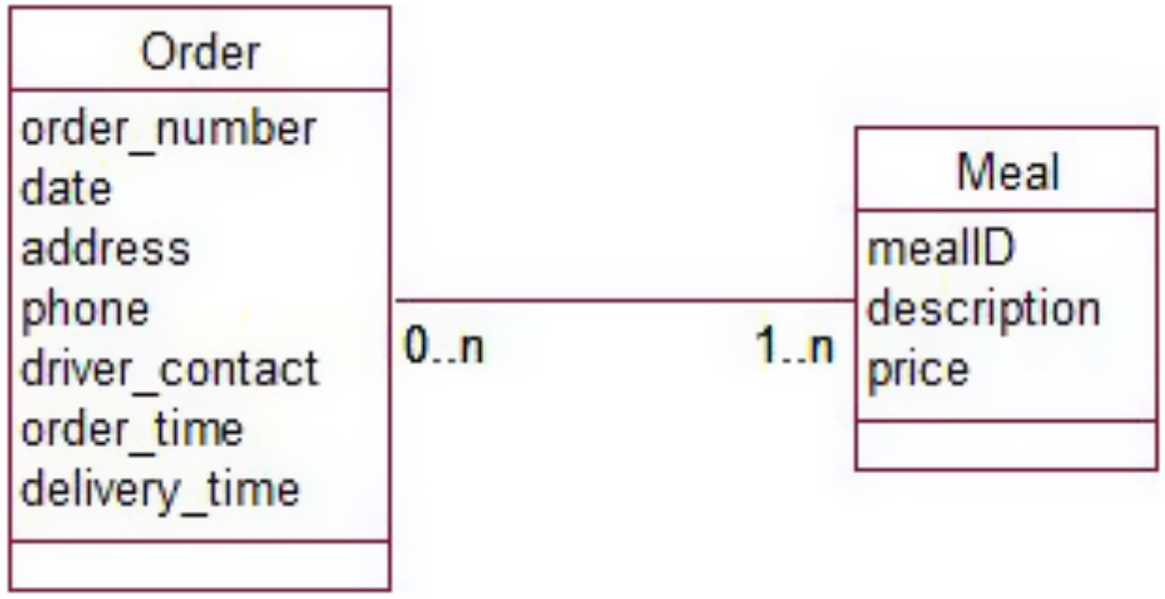
  Are there any data that I want to keep that are specific to a particular student and his or her enrollment in a particular course?

- One obvious piece of data that fits the preceding criteria is the **result** or **grade**.
- A student and a course can each have many enrollments, and a particular enrollment is for exactly one student and one course.

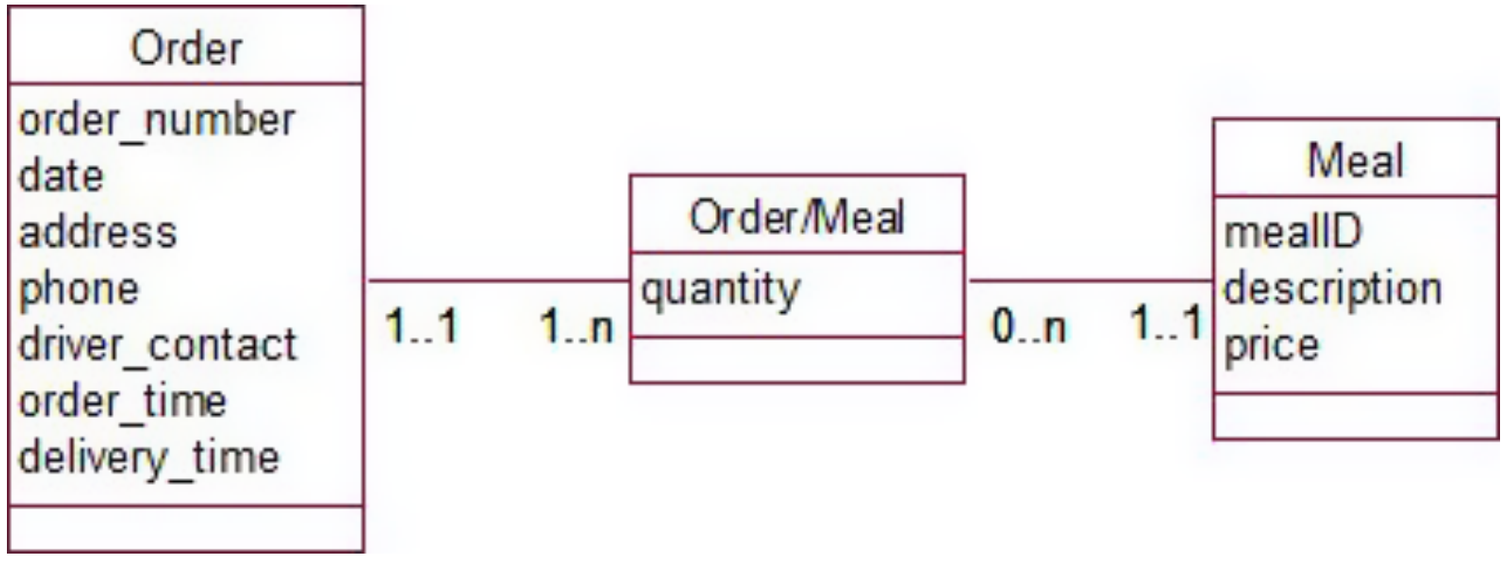


Meal Delivery Example

- The initial data model had a Many-Many relationship between types of meal and orders.



- If a family orders three chicken vindaloos, one hamburger, and one pork fried rice? Where do we put these quantities? The quantity cannot be an attribute in the **Order** class nor in the **Meal** class .



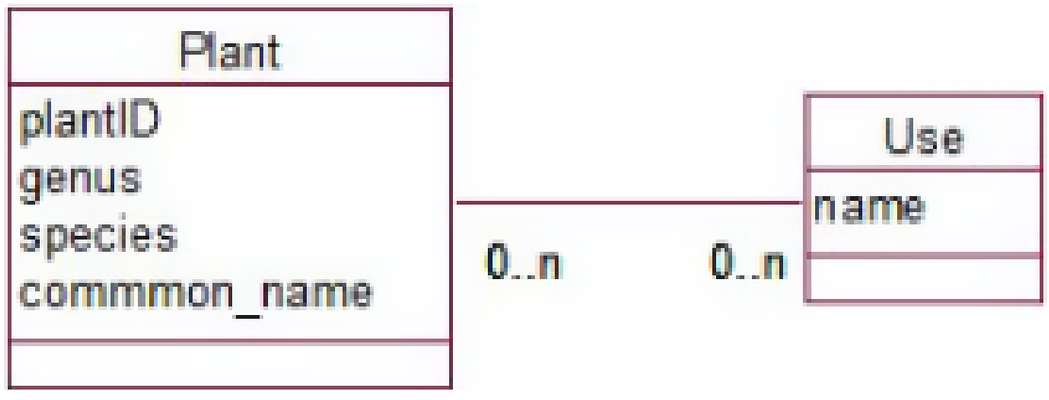
- It can be difficult to come up with a meaningful name for the intermediate class. We could maybe have called the class **Orderline**,
- We can also use this new intermediate class to solve one of the problem of coping with the price of a meal changing over time.
- This will be the price charged for a particular meal on a particular order and will not change when the current price changes in the **Meal** class. This way we have a complete history of the prices for each meal on each order. A price attribute in this intermediate class can allow us to keep historical data and also to deal with “unusual” situations such as specials or discounts..

Are there any data we need to store about a particular meal type on a particular order?

Yes, the quantity of that meal type ordered and the price being charged for that meal type on the order.

When a Many–Many Doesn't Need an Intermediate Class

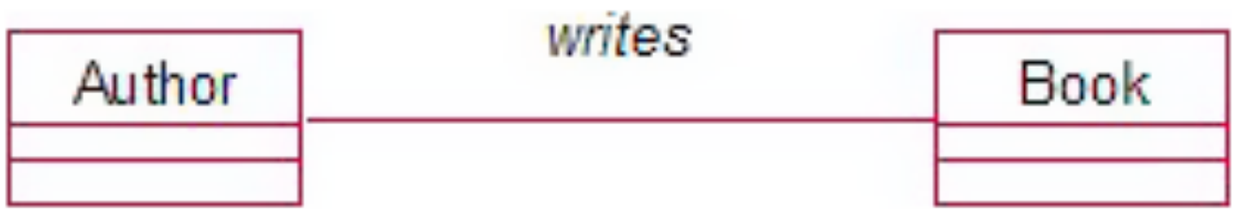
- A few Many-Many relationships contain complete information for a problem without the need for an intermediate class in the data model.
- Problems that involve categories as part of the data often do not require an additional class.



- "Is there any information we want to keep about a particular species and a particular use?"
  - We might want to record whether a particular plant is excellent or just reasonable at hedging.
  - We may want to note how many of a particular species are needed to be sufficient for attracting bees.

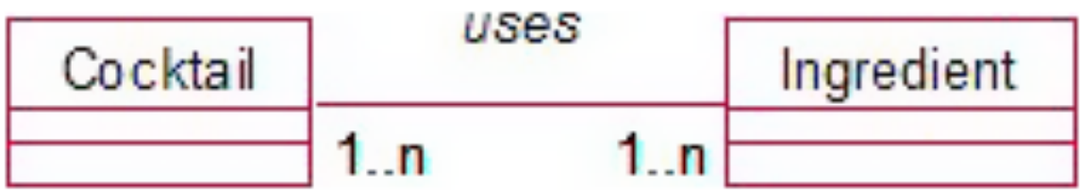
Exercise 7-1

The figure shows a first draft of modeling the situation where a publishing company wants to keep information about authors and books. Consider the possible optionalities at each end of the relationship writes and so determine some possible definitions for a book and an author.



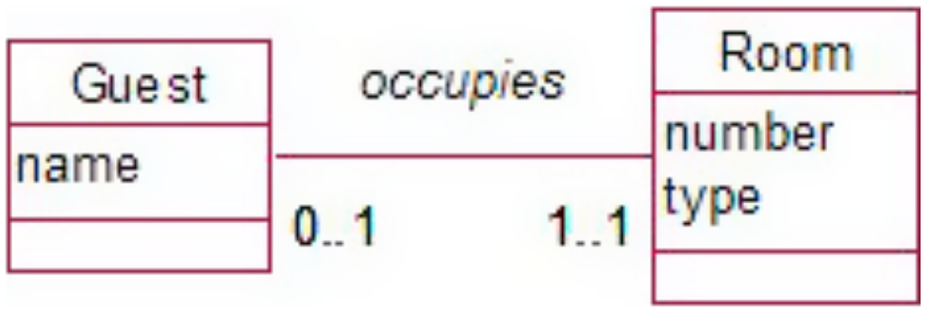
Exercise 7-2

The figure shows a possible data model for cocktail recipes. The Many-Many relationship uses can be navigated in either direction. To find out the ingredients in a Manhattan or to discover the possible uses for that bottle of Vermouth. What is missing?



Exercise 7-3

Part of the data model about guests at a hostel is shown in the figure. How could the model be amended to keep historical information about room occupancy?



Summary

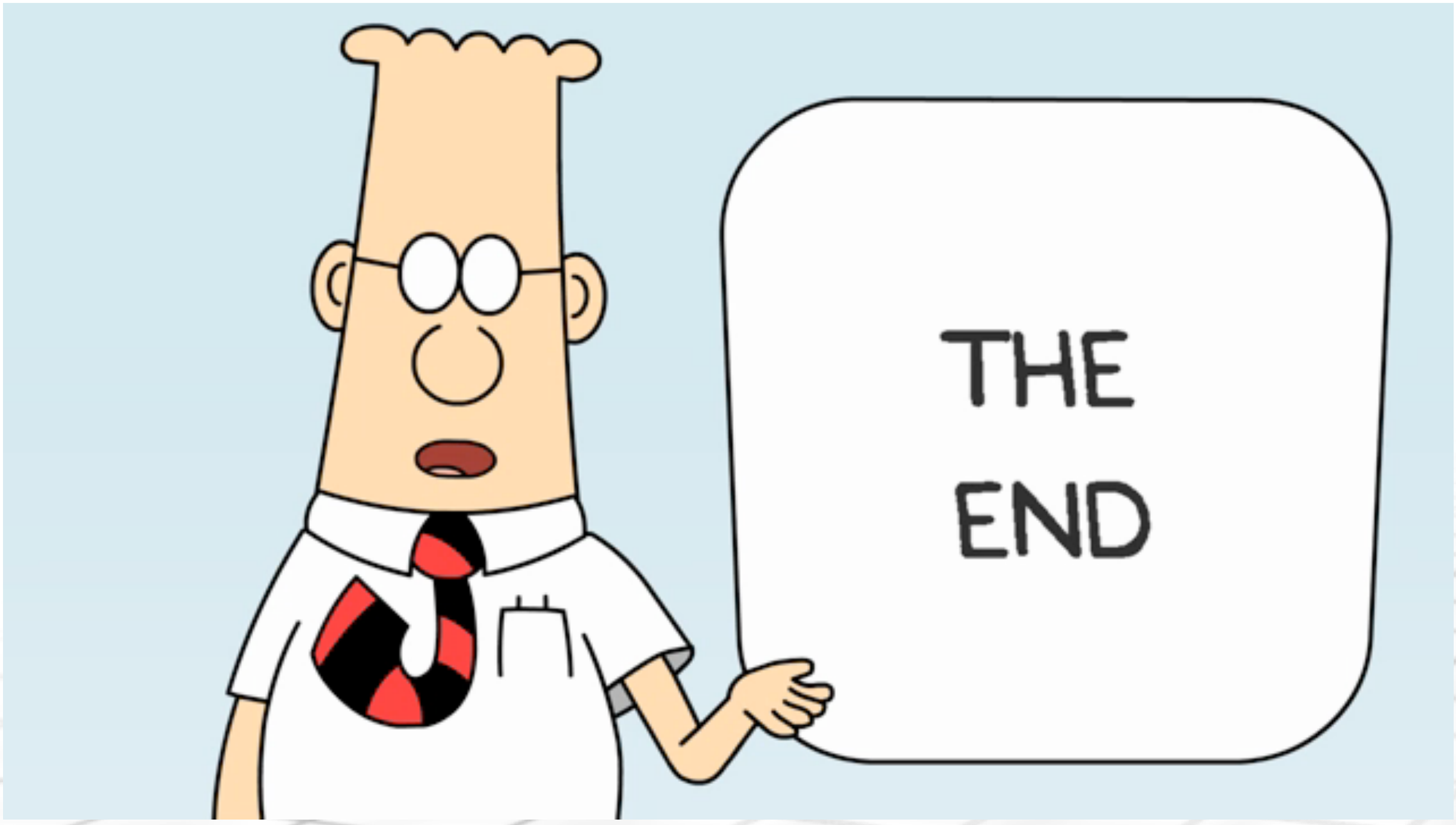
- The resulting clarifications to the problem should eventually be reflected in the use cases and may affect the final model and the eventual implementation.

Optionality: Should it be 0 or 1? Considering whether an optionality should be 0 or 1 might affect definitions of our classes; for example, “Would a student who was not enrolled in any courses still be considered a student for the purposes of our database?”

A cardinality of 1: Might it occasionally be 2? We need to consider whether there might be exceptional cases in which we might want to squeeze two numbers or categories into a box designed for one; for example, “What happens if the weather changes during a visit?” Redefining a class might help out for the exceptional cases, as in, “If the weather changes, we will call it two visits.”

A cardinality of 1: What about historical data? Always consider whether the 1 in a relationship really means “just one at a time.” For example, “A department has one manager. Do we want to know who the previous managers of the department were?” If so, the relationship should be Many-Many.

Many-Many: Are we missing anything? Consider whether there is information we need to record about a particular pairing of objects from each class; for example, “What might we want to know about a particular student and a particular course?” If there is such information (e.g., the grade), introduce a new intermediate class.



출처: metachannels.com

Thank you!