Introduction to Data and Database Modeling

WMDD 4921

Contents adapted by Jordan Miller from

Beginning Database Design: From Novice to Professional, Second Edition

by Clare Churcher Published by Apress, 2012

Relational Database Design and Implementation, 4th Edition

by Jan L. Harrington Published by Morgan Kaufmann, 2016

What is Data?

We know that data is an important part of web and mobile development. But what is it exactly?

Data in Business

Most businesses rely on data for their operation. Bad database design can seriously impact the success of a business.

Problems we would like to avoid through good database design:

- Not able to make best use of available data (examples?)
- Data is lost or destroyed
- Data contains errors or inaccuracies

In order to avoid these problems, we must design the database with a very clear understanding of **what the data is and how it will be used**.

Beginning Database Design: From Novice to Professional, Second Edition by Clare Churcher Published by Apress, 2012

Importance of Data

Bad data practices can negatively impact a business. An interesting example is the failed attempt to expand the major U.S. retailer **Target** into Canada: https://www.canadianbusiness.com/the-last-days-of-target-canada/

"One of the most important decisions [regarding the expansion] concerned technology—the systems that allow the company to order products ... process goods through warehouses and get them onto store shelves. Target was again seeking to do the impossible: It was going to set up and run [it's new inventory management system] in roughly two years. "

Importance of Data

From https://www.canadianbusiness.com/the-last-days-of-target-canada/

After their new inventory system was up and running:

"Items … were stalled—products weren't fitting into shipping containers as expected… Other items weren't able to fit properly onto store shelves … the underlying cause of the breakdown: ***The data contained within the company's supply chain software, which governs the movement of inventory, was riddled with flaws."***

Importance of Data

From https://www.canadianbusiness.com/the-last-days-of-target-canada/

"Product dimensions would be in inches, not centimetres or entered in the wrong order: width by height by length, instead of, say, length by width by height. Sometimes the wrong currency was used. Item descriptions were vague. Important information was missing."

Importance of Data

From https://www.canadianbusiness.com/the-last-days-of-target-canada/

"An employee … might have ordered 1,000 toothbrushes and mistakenly entered into [the software] that the shipment would arrive in a case pack containing 10 boxes of 100 toothbrushes each. But the shipment might actually be configured differently—four larger boxes of 250 toothbrushes, for example. As a result, that shipment ***wouldn't exist*** *within the distribution centre's software and couldn't be processed."*

Importance of Data

From https://www.canadianbusiness.com/the-last-days-of-target-canada/

"A technology team was finally able to install … [a] feature to catch bad data … If an employee entered a [product code] that was short one digit, for example, the system wouldn't allow that purchase order to proceed…"

What Exactly is a Database?

Specifically, what should we be able to do with a database?

Databases

- " …[the] fundamental concept [of] databases: There are things in a business environment about which we need to store data, and those things are related to one another in a variety of ways.

  In fact, to be considered a database, the place where data are stored must contain **not only the data** but also information about the **relationships between those data**." Harrington 2016 (emphasis mine)

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

Database Design

"To be useful, the data in a database must be accurate, complete, and organized in such a way that data can be retrieved when needed and in the format desired." Harrington 2016

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

Database Design

Your first instinct might be to store data in a **list** or **table,** or maybe a collection of **files and folders** with descriptive names.

However, for the data to be easily usable, extendable, and updateable, we might find that we need something more complex than a simple list or spreadsheet. **Databases help us organize the data and define how data are related to one another.**

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

What is a Database?

Using a database management system (DBMS) to interact with our database, we should be able to easily do the following:

- Retrieve data using queries that leverage logical relationships between data.

  Example: retrieve all students registered in a particular course, or retrieve all courses that a particular student is registered in.

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

What is a Database?

Using a database management system (DBMS) to interact with our database, we should be able to easily do the following:
Continued

- Add, delete, modify data without disorganizing or damaging existing data
- Perform operations (such as adding, updating, or retrieving) ***quickly***. This is non-trivial, since a database could contain millions or billions of entries. The structure used to store the data must be optimized for speed!

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

Database Design

Let's start the discussion of what makes a well-designed database by first talking about **bad database design**;

- **Unnecessary duplication of data**
  - Example: creating a new customer every time a person orders a product. (What happens if the same person orders many products?)
- **Data inconsistency**
  - Example: Suppose we were to add a student to a course, but we accidentally mistype their student number by adding an extra digit. Later, when we try to submit their grades, the college can't find anyone with that number. **How could this be avoided?**

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

Database Design

Let's start the discussion of what makes a well-designed database by first talking about **bad database design**;

- **Unnecessary duplication of data**
  - Example: creating a new customer every time a person orders a product. (What happens if the same person orders many products?)
- **Data inconsistency**
  - Example: Suppose we were to add a student to a course, but we accidentally mistype their student number by adding an extra digit. Later, when we try to submit their grades, the college can't find anyone with that number. **How could this be avoided?**

A well-designed database should eliminate as much duplication as possible, or at least make sure that when data must be duplicated, they are entered correctly.

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

Database Design

Bad database design practices continued

- **Insertion Anomalies:**
  - Example: Suppose data about each student is stored within the courses they are registered in. What happens if we try to create a student that is registered in a course that doesn't exist in the database? **How do we fix this problem?**
  - **Deletion Anomalies:**
    Example: suppose we have an inventory system where customers are stored as part of the data for products they've ordered. Suppose we then delete a product from the database; **what happens to the customers who've ordered that product?**
- **Bad Identifiers**
  - Suppose a customer ID code is created by combining their postal code and last initial. **What problems could arise from this?**

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

Database Design

Bad database design practices continued

- **Duplication and Modification:** Suppose we have a table of employees, and each one has their work phone number listed; since they all work at the same place, it's the same number for each employee. But what happens if the work phone number changes?

Database Design

Caveat: when talking about "bad practices" or "best practices", there are almost always exceptions. Implementing the aforementioned "bad practices" is sometimes appropriate, necessary, and optimal. We'll talk more about those cases later in the course when we learn about **non-relational data** and **no-SQL databases.**

(This is a good time to mention, as an aside, that one should always understand their business problem thoroughly and develop a good solution based on that analysis rather than simply doing what everyone else is doing.)

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

Data Modeling

"The formal way in which you express data [in] a database management system (DBMS) is known as a **data model**." Harrington 2016

In this class, we will start by using a **relational data model.** This is the most popular type of data model, although there are others. (more on this later.)

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

Data Modeling: Use Cases

We could start modeling the data that we will be using in our application by creating an **initial description of the problem.**

We can start doing this by creating **use cases**, which are descriptions of how users will use our application. They help us document all the things that we should be able to use the data for.

Example: music discography application

Beginning Database Design: From Novice to Professional, Second Edition by Clare Churcher Published by Apress, 2012

Discography Browser Application

Use Cases:

- The user can choose a band/artist from a list and see a description of that band with a list of albums they've released.
- They can choose an album to see the year it came out and a list of songs on the album.
- They can choose a song and see which albums it belongs to.

Constraints: There can be no collaboration albums (meaning no albums with more than one artist), but there *can* be compilation albums (meaning a song can exist on more than one album). There are no singles (meaning no songs without an album).

Beginning Database Design: From Novice to Professional, Second Edition by Clare Churcher Published by Apress, 2012

Data Modeling: Entities

The next step in designing a database will be identifying **entities.**

"An entity is something about which we store data." Harrington

Can you think of some examples of entities for the Discography Browser application?

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

Data Modeling: Entities

The next step in designing a database will be identifying **entities.**

"An entity is something about which we store data." Harrington

Can you think of some examples of entities for the Discography Browser application?

Maybe: Artist, Album, Song

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

Data Modeling: Entities

Entities have **attributes** that describe them.

What are some attributes of the entities you came up with in the previous example?

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

Data Modeling: Entities

Entities have **attributes** that describe them.

What are some attributes of the entities you came up with in the previous example?

For the Album entity, maybe:

- **Artist that recorded the album**
- **Name of the album**
- **List of songs on the album**

Data Modeling: Entities

Instances of entities are data in the database that belong to a particular entity type. For example, instances of the **Album** entity are **Dark Side of the Moon, Jagged Little Pill,** and **1989.**

- When we store an **instance** of an entity in a database, **we actually only store its attributes**! Each group of attributes represents a single instance of an entity.
  Example: creating **dog** as an instance of the **animal** entity with the following attributes: **name:'dog', numberOflegs:4, type:'mammal'**

Data Modeling: Entities

Since we may have many instances of a given entity (for example, a **student entity** might have the instances **Lin, Jason, Harpreet, Livia**) we need each instance to have an **entity identifier** so that we can uniquely tell them apart. (Can you think of situations in which we might need a unique identifier? Why might just the **name** not be enough?)

Requiring that each new instance have a unique identifier is what we would call a **constraint** on the database. Constraints help us maintain consistency and accuracy.

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

Data Modeling: Single and Multi-valued Attributes

In a **relational database**, attributes must be **single-valued**, which means each attribute can only have one value.

Suppose the **customer entity** has the attribute **phone number**. But suppose a given customer has both a **cell number** and a **work number**. You may be tempted to store **two values** for the **phone number** attribute, but we're not allowed to do that.

How do we resolve this apparent conflict? Does this mean the customer has to choose just one?

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

Data Modeling: Single and Multi-valued Attributes

In this case, we could **create a new entity** to represent **phone numbers**. Each phone number instance would include an **attribute** representing the owner of that number.

Just as there is no limit to how many customers we can have, there is also no limit to the number of phone numbers we can have, even for a single customer.

This is where you start to get a sense of how the relational model works!

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

Data Modeling: Single and Multi-valued Attributes

But why do you think only single-valued attributes are allowed? Can you come up with any examples to show why multi-value attributes might not be desirable? How would you implement multiple values for a given attribute?

" As a general rule, if you run across a multivalued attribute, this is a major hint that you need another entity. " **Harrington**

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

Entities and Collections

Despite the restrictions on multi-valued attributes, we will find ourselves designing databases that have collections of items.

For example, suppose we're designing a database for an online shop. Naturally, we'll want to have an inventory of products.

Which should be an entity in our database: the inventory, or the product?

Entities and Collections

Despite the restrictions on multi-valued attributes, we will find ourselves designing databases that have collections of items.

For example, suppose we're designing a database for an online shop. Naturally, we'll want to have an inventory of products.

Which should be an entity in our database: the inventory, or the product?

If we were to choose **Inventory** as the entity, that would mean the **products** attribute would be multi-valued, which we don't want! (Maybe BOTH should be entities if there's a possibility of having more than one inventory!)

Documenting Entities

We'll use diagrams of entities and relationships, called **ER diagrams** or **ERD**s, to visualize our databases. ERDs are a good way to visualize your database as you're designing it.

A few different types of diagrams can be used for this:

- **Chen ERDs** (simple, but can get cluttered)
- **Information Engineering/IE/Crow's Feet**
- **Unified Modeling Language (UML)** (often used for object-oriented systems)

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

Documenting Entities

We'll use diagrams of entities and relationships, called **ER diagrams** or **ERD**s, to visualize our databases. ERDs are a good way to visualize your database as you're designing it.

A few different types of diagrams can be used for this:

- **Chen ERDs** (simple, but can get cluttered)
- **Information Engineering/IE/Crow's Feet**
- **Unified Modeling Language (UML)** (often used for object-oriented systems)

These are similar and we'll use both.

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

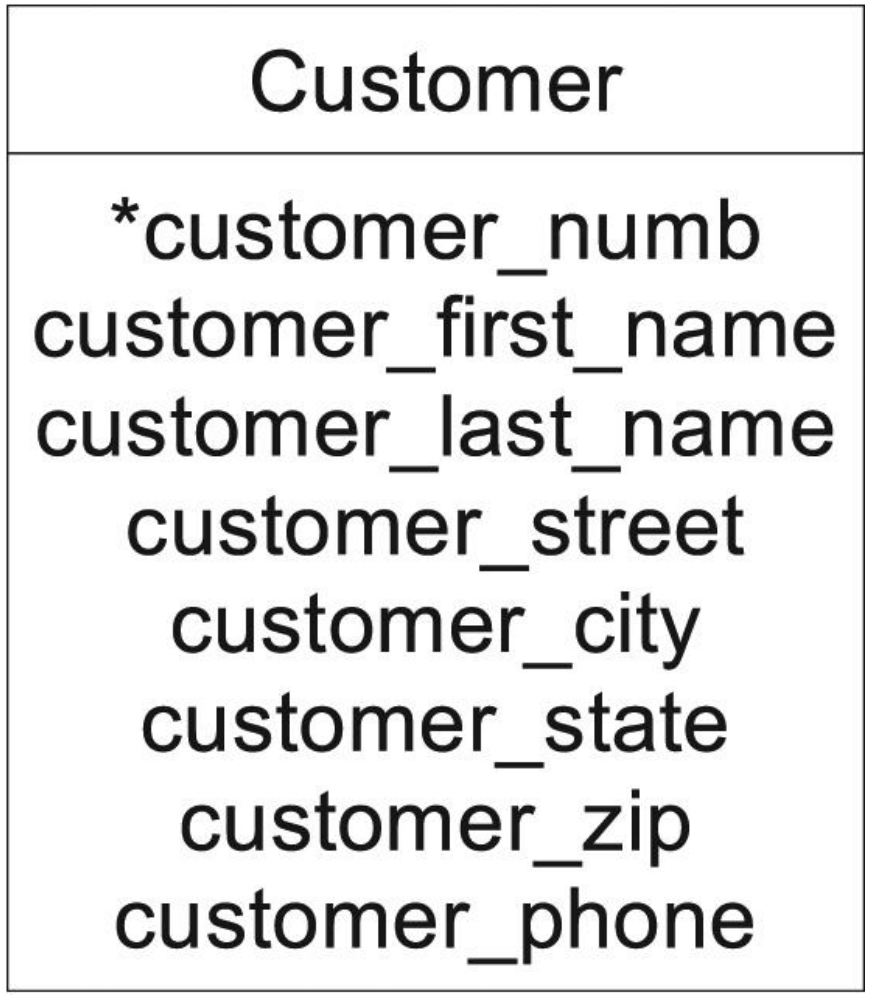
Documenting Entities

By creating an ERD for our database, we're also creating a **logical schema,** which is the large-scale logical plan for a database. (independent of how it will be implemented in the database.)

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

Documenting Entities

IE/UML diagrams: The entity name is specified in a rectangle at the top, with attributes as a list underneath. The entity identifier has an * beside it.



Documenting Entities: Domain

Each attribute has a **domain**, which is the type of data that is allowed and the permissible values. **Examples?**

We can add the domain to our UML entity after each attribute:

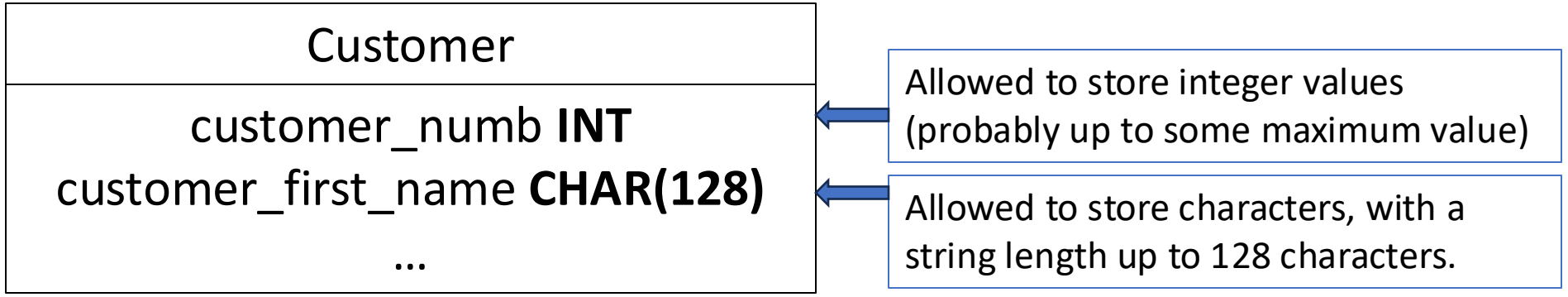
Customer
customer_numb **INT** customer_first_name **CHAR(128)** …

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

Documenting Entities: Domain

Each attribute has a **domain**, which is the type of data that is allowed and the permissible values. **Examples?**

We can add the domain to our UML entity after each attribute:

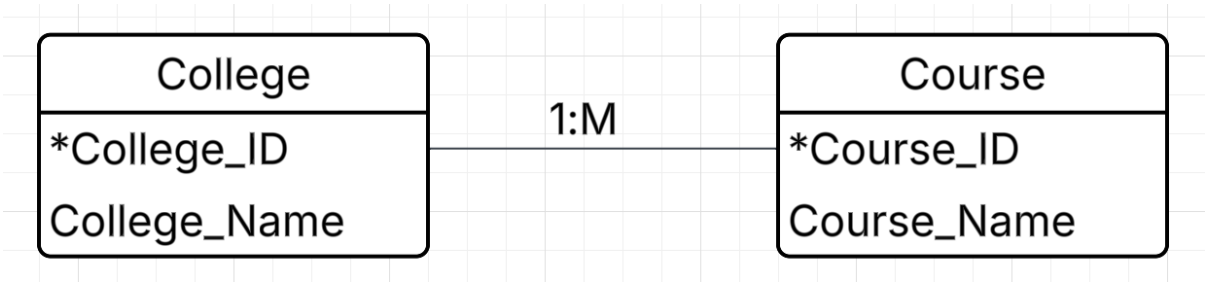


Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

Documenting Relationships

Within our diagrams, we will document the relationships between **entity instances.** Unless otherwise specified, relationships are optional. (Meaning not every instance need have these relationships.)

- One-to-one (1:1)
- One-to-many (1:M)
- Many-to-many: (M:M)



Documenting Relationships

Cardinality: How many instances of an entity are related to an instance of another entity.

- Example: **one** student is registered in **many** courses.
- Or: **many** students are registered in a course, and each student is registered in **many** courses

Ordinality: whether or not a relationship is optional or mandatory

- Example: a course *must* have an instructor.
- Example: a customer doesn't necessarily have any orders.

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

Documenting Relationships: 1:1

One-to-one: In this type of relationship, a pair of entity instances are related to each other and not any other instances of the same types.

Try to think of some examples; however, note that this relationship is rare, and you may be able to express a one-to-one relationship between two entities as a single entity with an attribute.

Documenting Relationships: 1:M

One-to-Many

Example: A CPU can only be installed in one computer, but a computer can have more than one CPU. This means the **entity of CPU** and the **entity of Computer** are in a **one-to-many** relationship.

Documenting Relationships: M:M

Many-to-Many

Can you think of examples?

Documenting Relationships: M:M

Many-to-Many

Can you think of examples?

We will soon see that M:M relationships present problems for relational DBMSs, but we'll get to that later.

Weak Entities

In the previously-discussed relationships, participation between entities is often optional. However, not all relationships are optional.

Consider two entities: **customer** and **order**. A customer instance can exist without an order instance, but the reverse is not true. An order instance ***must*** have a customer instance to exist. We therefore say that the **order** entity is ***weak.***

Identifying and Non-Identifying Relationships

An **identifying relationship** is one in which the **child** entity depends on the **parent** and cannot exist without it.

Examples:

- can an **order** for a product exist without a **customer**?
- can a **song** exist without an **album**?

Identifying and Non-Identifying Relationships

An **identifying relationship** is one in which the **child** entity depends on the **parent** and cannot exist without it.

Examples:

- can an **order** for a product exist without a **customer**?
- can a **song** exist without an **album**?

In the above examples, which entities are weak and which are strong? How does this affect whether the relationship is identifying or non-identifying?

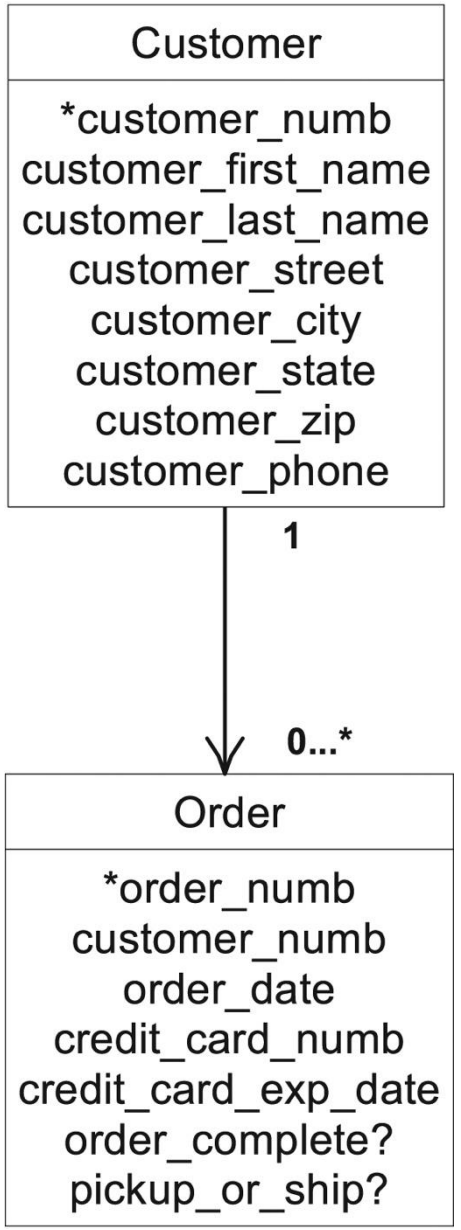
Identifying and Non-Identifying Relationships

Later, we will see that a weak entity that is in an identifying relationship with a strong entity will have a unique identifier that contains some reference to the identifier of its parent.

ERD Diagram Relationships

UML:

- 1: One and only one (mandatory)
- 1…*: One or more (mandatory)
- 0…1: Zero or one (optional)
- 0…*: Zero, one, or more

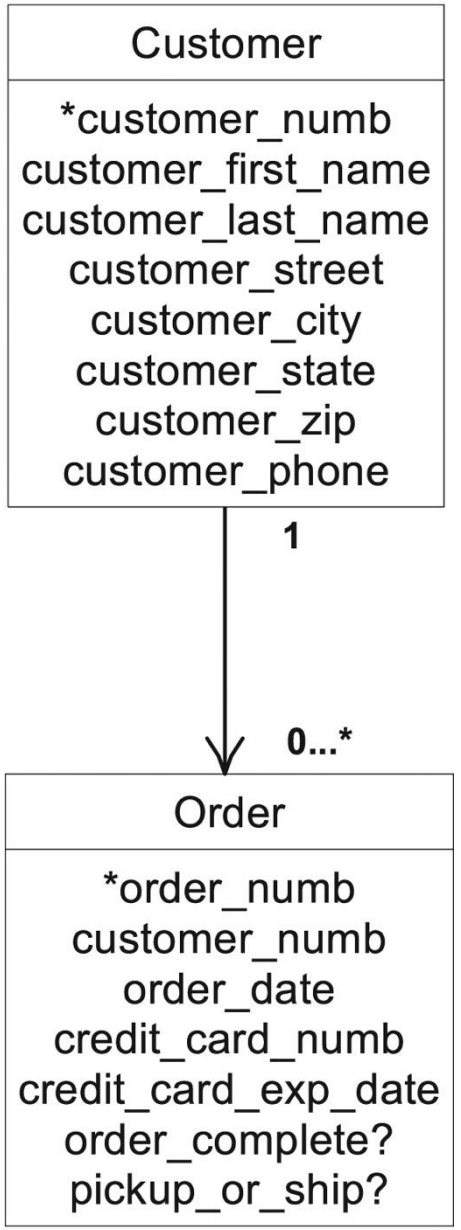


ERD Diagram Relationships

UML:

- 1: One and only one (mandatory)
- 1…*: One or more (mandatory)
- 0…1: Zero or one (optional)
- 0…*: Zero, one, or more

These are called **multiplicities.**



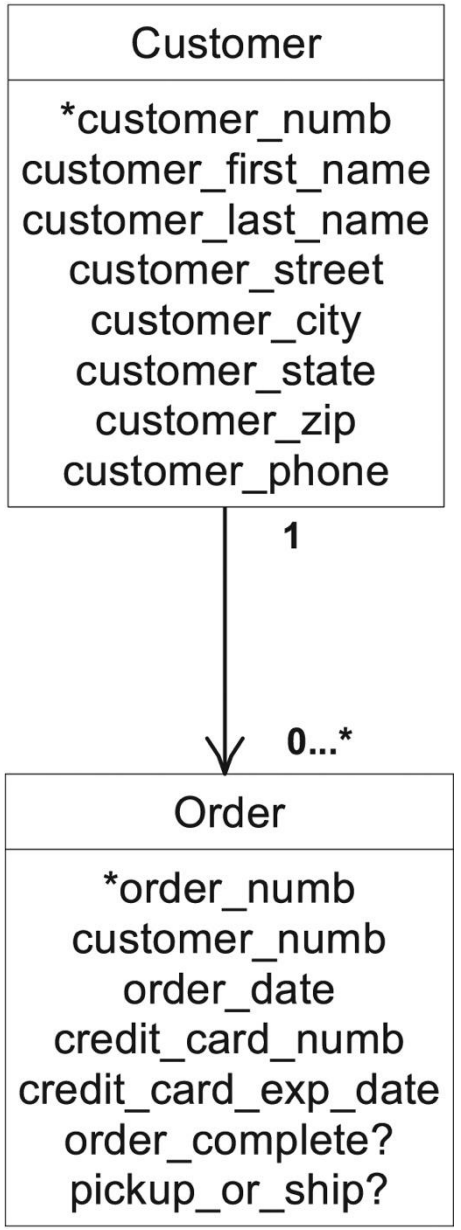
ERD Diagram Relationships

UML:

- 1: One and only one (mandatory)
- 1…*: One or more (mandatory)
- 0…1: Zero or one (optional)
- 0…*: Zero, one, or more

These are called **multiplicities.**

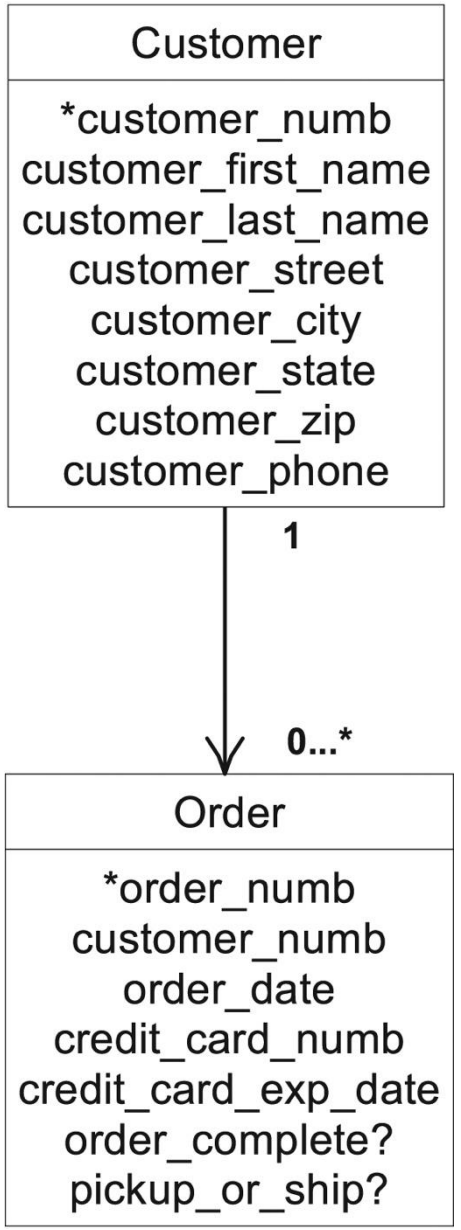
(you can see that multiplicities are related to **ordinality** and **cardinality.)**



ERD Diagram Relationships

UML:

- 1: One and only one (mandatory)
- 1…*: One or more (mandatory)
- 0…1: Zero or one (optional)
- 0…*: Zero, one, or more



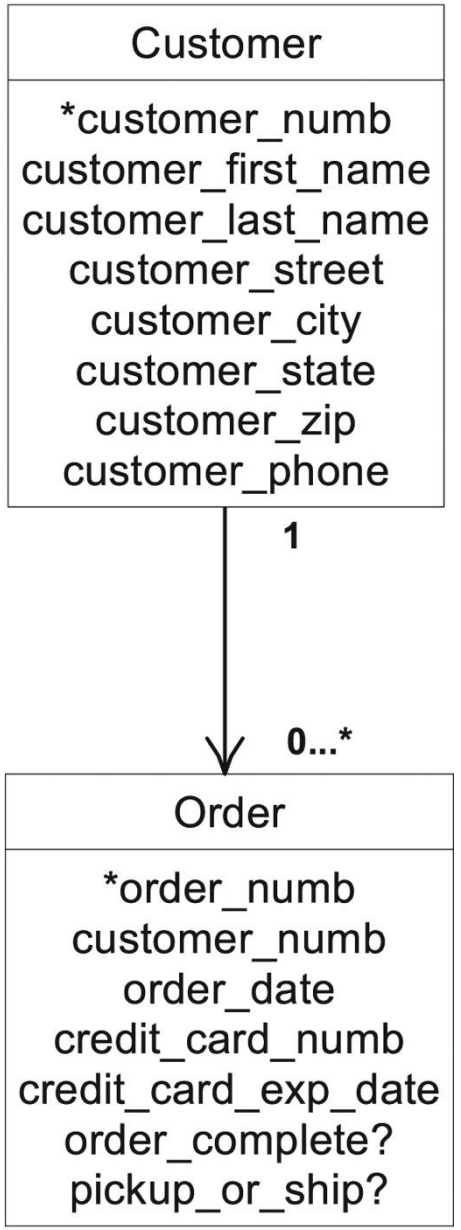
See this article for multiplicities represented in the IE / Crow's Foot convention: https://www.lucidchart.com/pages/ER-diagram-symbols-and-meaning#section-2-heading

ERD Diagram Relationships

UML:

- 1: One and only one (mandatory)
- 1…*: One or more (mandatory)
- 0…1: Zero or one (optional)
- 0…*: Zero, one, or more

Which entity is the **parent** here?



ERD Diagram Software

There are MANY programs for creating ER diagrams, including programs that can convert ERDs into databases, and vice versa. Drawing ERDs by hand on paper is also totally legitimate.

Here are some free browser-based programs I recommend:

- DrawSQL
  - Very simple and easy to learn, but with limited options
- Lucidchart
  - Much more complex and difficult to learn, but no constraints whatsoever

Discography Browser Revisited

The user can choose a band/artist from a list and see a description of that band with a list of albums they've released. They can choose an album to see the year it came out and a list of songs on the album. They can choose a song and see which albums it belongs to.

Constraints: There can be no collaboration albums, but there *can* be compilation albums. There are no singles (meaning no songs without an album).

Draw the ERD: Come up with entities, attributes, and relationships, and include the multiplicities.

Discography Browser Revisited

We've already learned that spreadsheets and tables aren't ideal for storing data because they don't have all the benefits of a database. ***But it turns out that relational databases actually themselves use tables!*** The tables in a relational database correspond to the **entities** in the database! Each **row** in the table corresponds to an **instance** of the entity, and each **column** in the table corresponds to an **attribute** of the **entity.**

Draw some tables to represent the entities in your database, along with columns for their attributes and some rows as example instances.

Many-to-Many Relationships

Earlier, we mentioned that **many-to-many** relationships in databases are **not ideal** for several reasons. For example, when you made tables for your Discography application, you may have created an **Album** table that either had many columns for songs (one column for each song) or maybe one column that contained many songs. You can see how this could create problems when trying to update the name of a song (many copies of it exist in the database that must be updated) or trying to figure out which albums a song belongs to (you have to search every single album to be sure you found all of them.)

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

Many-to-Many Relationships

Futhermore, what if we want to store some additional **information about a relationship** itself? Using the example of an online store, consider the entity **order** which is related to the entity **item** because items are contained in an order. What if a customer wants to order multiple copies of the same item? Where do you store the number of copies?

You can't store it in the **order**, because the order could contain multiple items, and you need to somehow relate the number of copies to the correct item. But you can't store it in the **item** either, because that item could be stored in different orders!

Composite Entities

A **composite entity** represents a relationship between two other entities. It can be used for the following:

- Break a M:M relationship into two 1:M relationships
- Store attributes about the relationship.

Can you think of a way to use this to solve the album/song problem? Try redrawing your ERD and tables for the Discography app to implement a **composite entity** (which will require a **composite table**.)

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016

Business Rules

Designing a database is as much an art as it is a science. There might not be a single right way to do it. However, the rules of the application (or business) that you're modeling will determine what your ERD looks like.

Example: what if we relaxed the constraint that an album can be associated with only one band or artist? This seems like a small, insignificant change, but it would require a redraw of your ERD and (as we will see) a redesign of the database!

Relational Database Design and Implementation, 4th Edition by Jan L. Harrington Published by Morgan Kaufmann, 2016