Entity Relationship Diagrams

DB Practice CSC 330 Spring 2022

Marcus Birkenkrahe

March 26, 2024

README

- This is a notebook for learning and practicing modeling relational databases using Entity Relationship Diagrams, keys and bridge tables
- Though we won't get more deeply into database design, understanding relationships, you need to understand this to understand `JOIN` commands when querying relational databases
- The notebook also contains two practice exercises for you to complete on your own in a self-guided lab session.
- **Modeling** is an essential IT design skill. You can model:
  - Technical processes (e.g. with UML)
  - Human processes (e.g. with BPMN)
  - Algorithms (e.g. with pseudocode or flowcharts)
  - Entity relationships (e.g. with ERDs)

ERD notations

- Relational database design relies on table relationships
- This is especially important for `JOIN` operations
- There are two dominant notations for ERD[1]

[1] Notation reflects priorities, e.g. readibility vs. detail. It is surprising that there aren't more popular notations! Notation must faithfully represent the modeling standard. You can in fact become famous with notation - Feynman diagrams are an example: they are a diagrammatic language for complicated integrals that represent elementary particle interactions.

Chen notation

- **Notation Example**

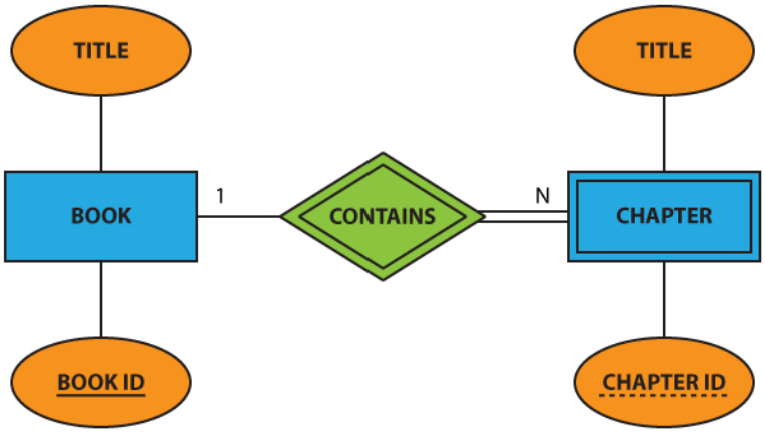

Figure 1: ERD Chen notation example (Source: Dybka, 2014)

- `BOOK` and `CHAPTER` are two entities (tables)
- A `BOOK` fully `CONTAINS` 1...N `CHAPTER` entities
- The `CHAPTER` is a fully dependent child of `BOOK`
- The `CHAPTER` totally participates in the `BOOK`
- The `BOOK` has attributes `TITLE` and `BOOK_ID`
- `BOOK_ID` is a *Primary Key* (`PK`) of `BOOK`
- The `CHAPTER` has attributes `TITLE` and `CHAPTER_ID`
- The `CHAPTER_ID` is a *Primary Key* (`PK`) of `CHAPTER`

Crow's foot notation

- **Notation summary**

  1. A box represents an **entity**, e.g. `book`

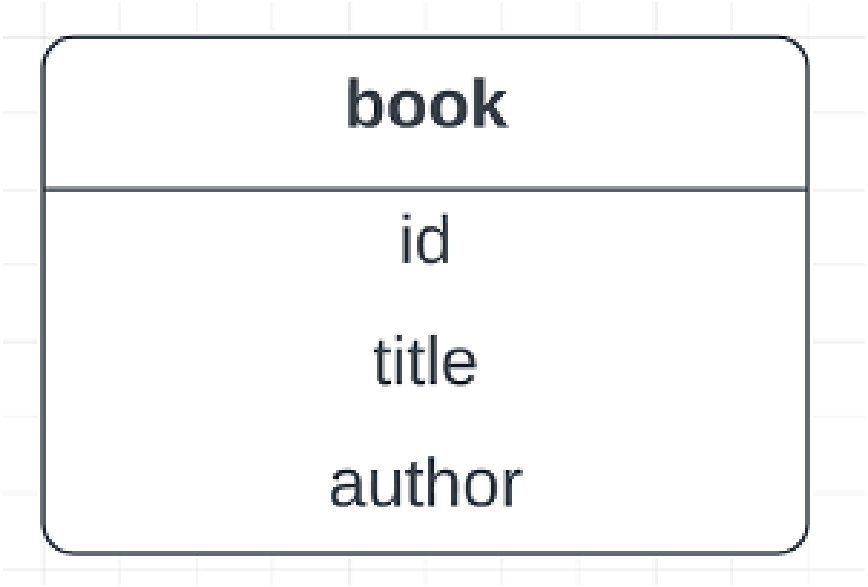

Figure 2: Entity with attributes

2. An entity has **attributes**, e.g. `book_id`, `book_title` etc.
3. The attributes can be overloaded with additional properties like *Primary Key*, *Foreign Key*, and they have types like *integer*, *text* etc.
4. Lines between entities represent a (binary) **relationship**
5. Relationships have two indicators: **maximum** (aka multiplicity) and **minimum**
6. Multiplicity = *maximum* number of associations between the entities
7. Example 1: "A book has one and only one ISBN."
8. Example 2: "A book has many chapters."
9. Example 3: "A book has at least one chapter."
10. Example 4: "A book has no or many new editions."

- Realistic example: bank tables
- What are some example for table relationships in the

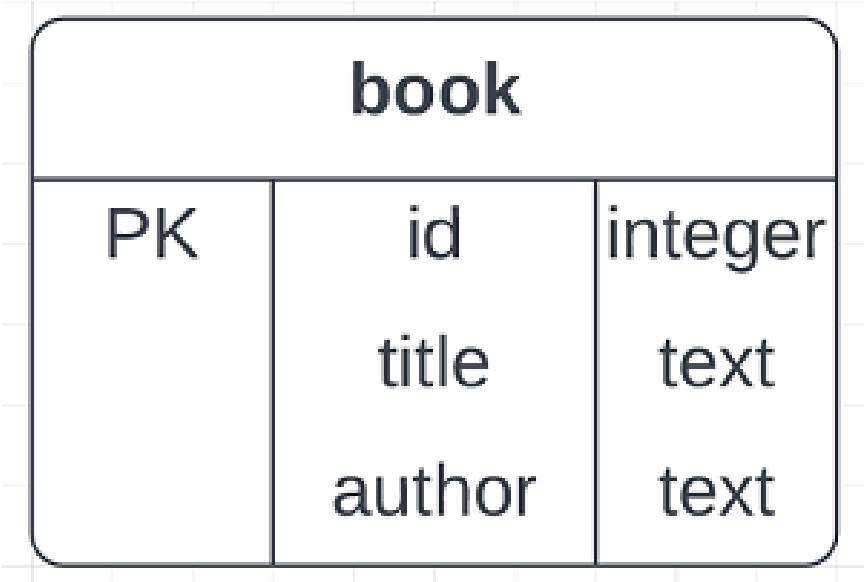

Figure 3: Entity with overloaded attributes

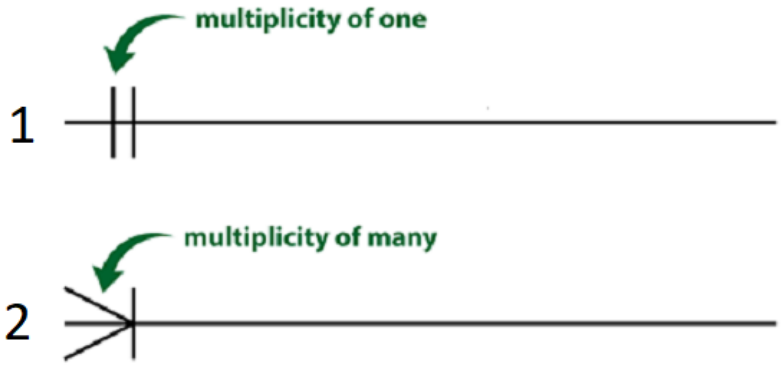

Figure 4: Multiplicity (Source: Dybka, 2016)

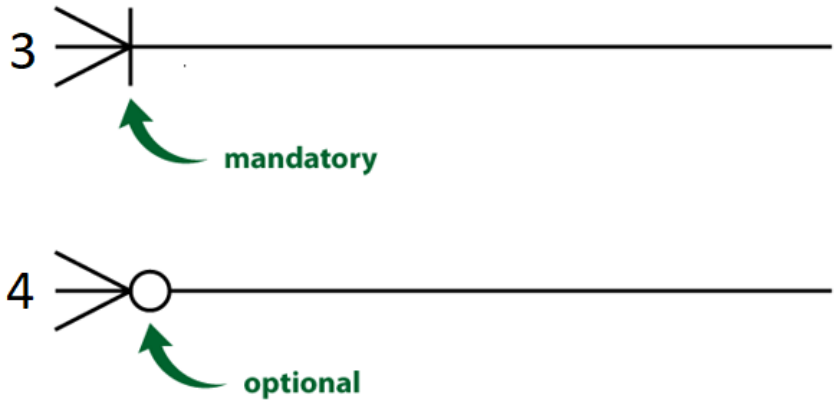

Figure 5: Minimum (Source: Dybka, 2016)

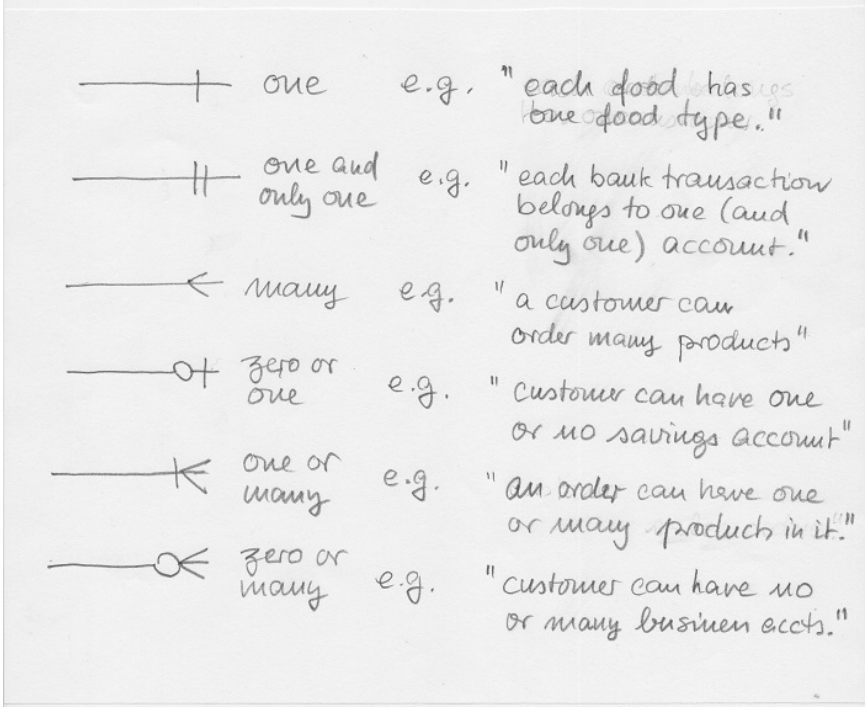

Figure 6: ERD crow's foot notation summary

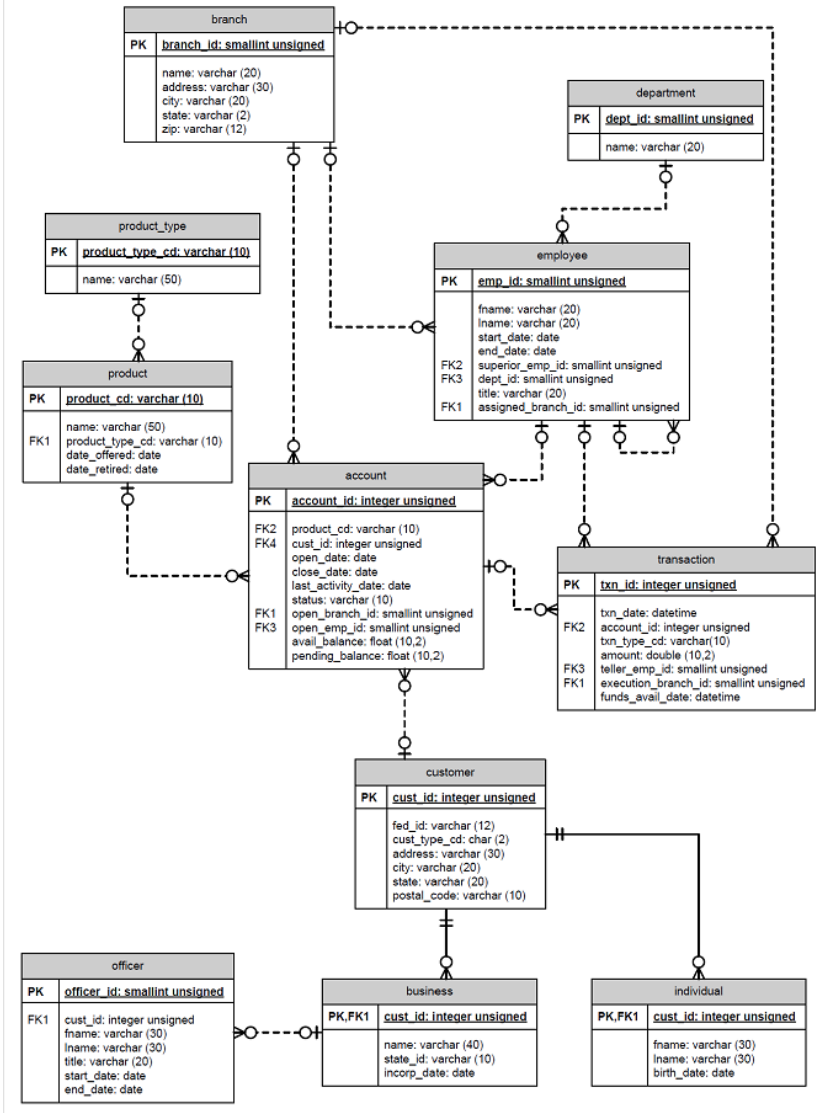

Figure 7: ERD of a bank (Source: Beaulieu, 2008)

Food database revisited

Create diagram with `draw.io`

- ☐ Open draw.io - you can choose where to save your diagrams
- ☐ If you choose Google Drive, you need to sign in to authorize as shown in the image.

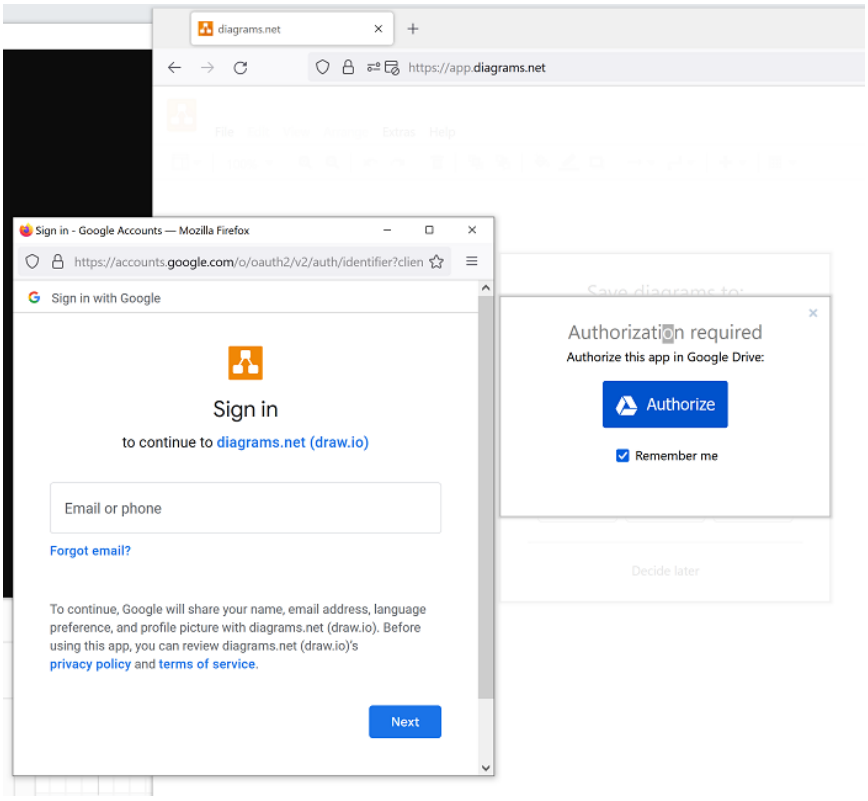

Figure 8: draw.io authorization dialog

- ☐ When you've authorized the storage place, you can create a new diagram. Don't bother with the templates.
- ☐ This short video shows how to create an entity and relationships between them using the crow's foot notation.

Create diagram

- ☐ *Zoom to Width* using the menu (left bottom of screen)
- ☐ Highlight and delete the diagrams on the screen
- ☐ Drag a table template onto the drawing board
- ☐ Draw the 4 entity diagrams for the food database

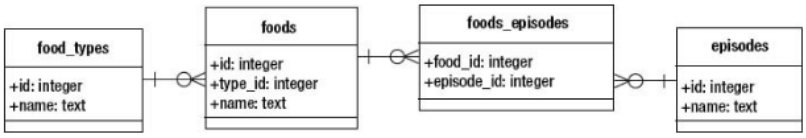

Figure 9: ERD of the food database (Source: Allen/Owens, 2010)

- ☐ Write down the relationships between `food_types` and `food`. Remember that you're not trying to describe all possible relationships but only the relationship for the purpose of this database with its narrow meaning: types of foods shown in episodes of the Seinfeld TV show.

  `food_types` to `food`: each food type has zero or many instances of food in the show.

  `food` to `food_types`: each food on the show is exactly one type of food.

Practice: customer orders

- ☒ Develop an ER diagram with three tables: `customer`, `order`, and `product`
- ☒ Identify suitable attributes
- ☒ The diagram should allow for relationships like
  - "A customer submits an order"
  - "An order contains a product"
  - "An order belongs to a customer"
  - "A product is part of an order."

☒ Use draw.io to draw the diagram

☒ Draw relationships with the correct minimum/maximum

☐ Give your diagram a title

☐ Share the link to your solution with me via Email

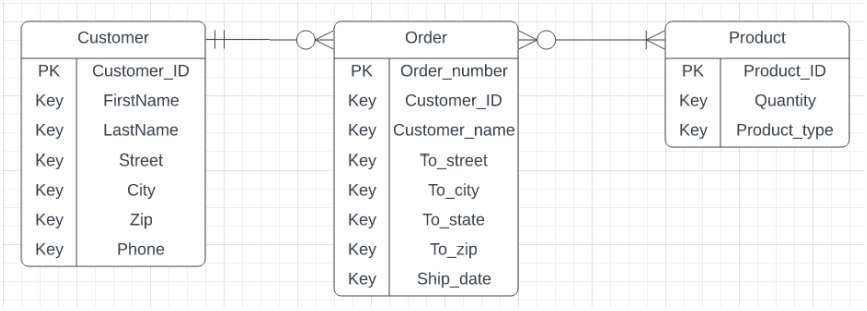

Figure 10: Customer orders products ERD

Practice exercise

☐ Develop an ER diagram with relationships between **four** tables: `Student`, `Course`, `Lecturer`, and `Class` (or meeting).

☐ Identify suitable attributes: make sure that they are

- measurable (you can think of a suitable data type)
- atomic (`address` is composite, `street`, `street number` are atomic)
- attributes (like `name`) and not entities (like `registrar`) or relationships (like `attendance`) themselves

☐ The diagram should allow for relationships like

- "A student is enroled in a course."
- "A class is attended by students."
- "An lecturer offers a course."
- "A class is taught by an lecturer."
- etc.

☐ Use draw.io to draw the final ERD (see demo video)

- ☐ Draw relationships with the correct minimum/maximum
- ☐ Put `ERD assignment by [Your name] - Pledged` in the title of your diagram
- ☐ Upload a screenshot of your solution to Schoology

Sample solution

- All relationships are subject to **business rules**. E.g. at least one student may have to be enroled in a course, or a lecturer may only be allowed to give a certain number of lectures - or reversely, a lecture may have to have at least one lecturer assigned to it.
- Because business rules change, the relationship structure of the database may also change. The schema, reflected in the entity relationship diagram, reflects this (source diagram at ponyorm).

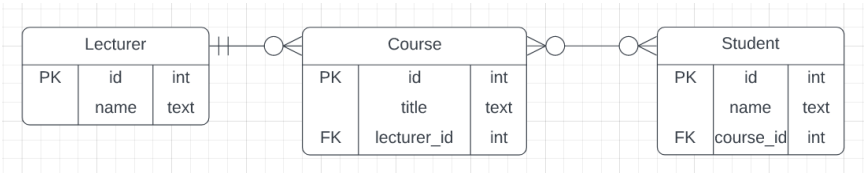

Figure 11: Sample solution (part) for students db

- Important to note:
  - Each lecturer can teach many courses (or none)
  - Each course can only be taught by exactly one lecturer
  - Each course can have many students (or none)
  - Each student can have many courses (or none)

Keys and bridge tables

ERD and relational model

- The ER model maps the relational database model - using the `Lecturer` table as an example.

ER MODEL (ORM)	DATABASE	EXAMPLE
Entity	Table	Lecturer
Entity attribute	Tuple/record/row	("2", "Birkenkrahe")
Attribute	Column	Lecturer.name
Attribute type	Column data type	integer/int

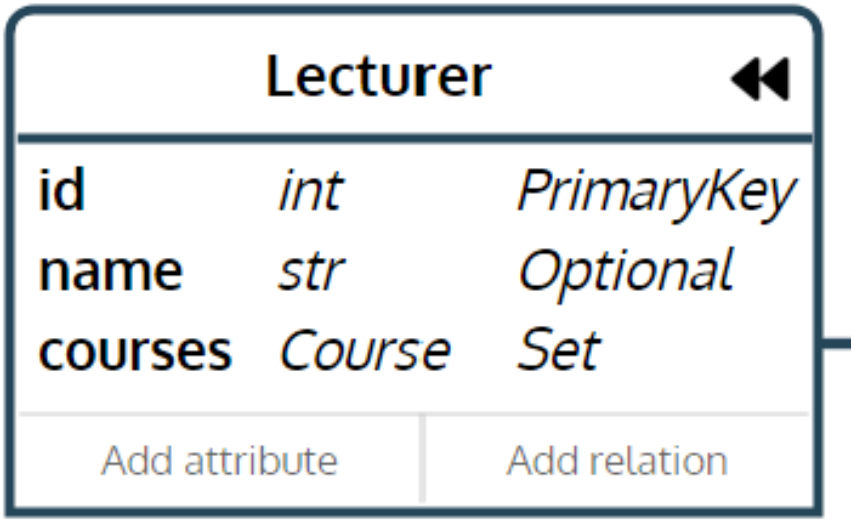

Figure 12: Lecturer table

- Some database design necessities such as normalization (removal of logical dependencies between cells) can be done using ERDs.

PK Definition

- A **primary key** (`PK`) is an attribute (or column) that uniquely identifies every record in a certain table.
- We already marked the potential PKs in the figure 10.
- Primary key rules:
  1. **UNIQUE** (across the database, i.e. all tables)
  2. **unchangeable** (while the table exists)
  3. **NOT NULL** (when data are inserted)

Key candidate identification

- In any table, the tuple of potential primary keys form the **candidate key**.

  Example table:

id	fname	lname	street	city	zip	phone
30014	John	Smith	1014 Main St	Batesville	72501	870-307-4245
30067	John	Smith	2300 College Rd	Batesville	72501	501-444-4287
30333	Jane	Doe	1014 Main St	Conway	72004	877-223-4445

- Names, phones, addresses are not *unique* or *unchangeable*
- Only `id` is designed to fulfil the PK rules
- Can be "randomly" assigned (are there problems with that?)[2]

Example: web sites

- If a website does not let you change your username, it likely uses your `username` as a primary key (*unchangeable*)
- If you're told that a username is already taken, it likely uses your `username` as a primary key (*unique*)
- A website will force you to enter certain information for database reasons if the information is used to create your PK (must be `NOT NULL`)

Foreign Keys

- Foreign keys are primary keys in other (linked) tables in the same database
- Example from the customer example:
- Foreign keys do not need to be unique (a customer could make another order) and there can be multiple foreign keys in one entity - if we want to link information across multiple tables.
- Example: `product_id` in a customer order. Now the order table has two foreign keys, `customer_id` and `product_id`.

[2] Yes - (1) computers cannot generate true random numbers; (2) the basis for the number may over time get exhausted - this happened e.g. with IPv4 addresses (see "IPv4 address exhaustion").

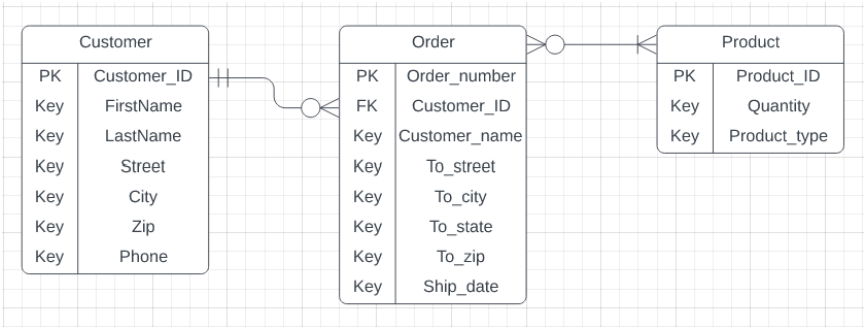

Figure 13: PK and FK in Customer and Order (ERD)

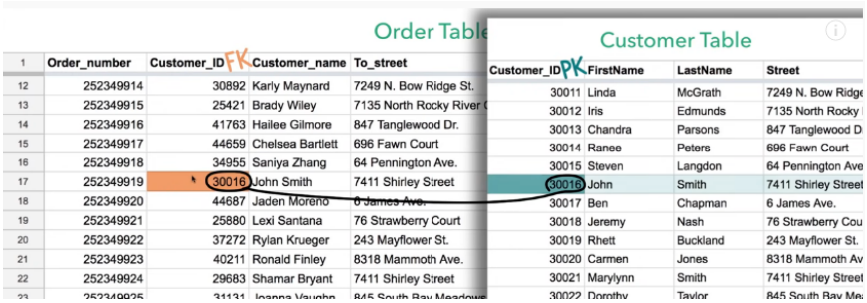

Figure 14: PK and FK in Customer and Order (tables)

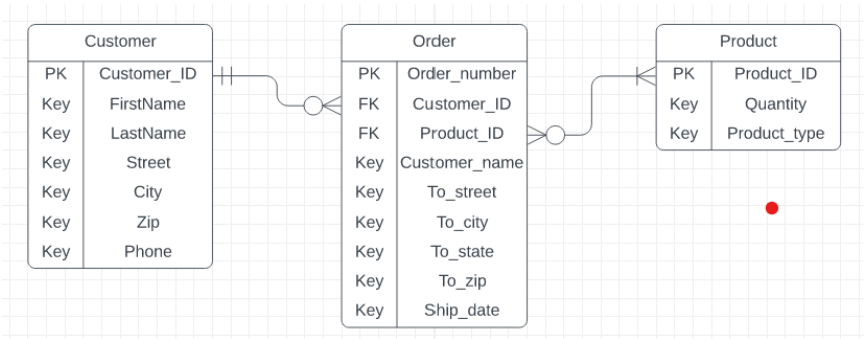

Figure 15: PK and FK in Order and Product (ERD)

- Example bank ERD 7:
  - the `employee` table has the FKs `dept_id` and `assigned_branch`, linking each employee to a department and a branch.
  - the `transaction` table logs bank transactions; its FKs are the `account_id` for the account used, the `teller_emp_id` of the teller responsible for the transaction, and the `execution_branch_id` for the executing bank branch.

Composite primary key

- Let's say, an order is sent in two different shipments.

Shipment Table

1	Product_ID	Order_number	ChargeCardTime	PackingTime	ShipOrderDate
13	49225	252349915	6/1/2017 9:13:34	6/2/2017 10:14:46	6/3/2017 11:15:52
14	40807	252349916	6/1/2017 9:14:16	6/2/2017 10:15:02	6/3/2017 11:16:03
15	76342	252349917	6/1/2017 9:14:01	6/2/2017 10:15:26	6/3/2017 11:16:13
16	96893	252349918	6/1/2017 9:14:21	6/2/2017 10:15:39	6/3/2017 11:16:19
17	69246	252349919	6/1/2017 9:14:34	6/2/2017 10:15:41	6/3/2017 11:16:47
18	69253	252349919	6/1/2017 9:14:34	6/2/2017 10:15:45	6/3/2017 11:16:47
19	99002	252349920	6/1/2017 9:15:07	6/2/2017 10:16:07	6/3/2017 11:17:11
20	64382	252349921	6/1/2017 9:15:14	6/2/2017 10:16:07	6/3/2017 11:17:23
21	91514	252349922	6/1/2017 9:15:33	6/2/2017 10:16:28	6/3/2017 11:17:23

Figure 16: Two orders in the shipment table

- We need a composite primary key, because none of the individual IDs will satisfy our rules - none are unique:
  - Someone else might order the same product (`Product_ID`)
  - Products might be in the same order (`Order_ID`)
  - Time and date information of two orders could coincide
- The pair `(Product_ID, Order_ID)` is unique for the shipment - it's a valid **composite primary key**.
- Rules:
  - use the fewest number of attributes possible
  - attributes should be unchangeable
- Alternative: add a `Shipping_ID`. There are arguments for either method.

- In the bank ERD 7 there is one composite key in the table `business`. It consists of the PK `cust_id` in the `customer` table, and the `FK` that points at `cust_id` in the `officer` table.

Bridge tables

- If two tables (like `Student` and `Course` in figure 11) are connected by man-to-many relationships, you need a bridge table to remove ambiguities.
- Otherwise, many details are not accessible: e.g. without the Order table, you would not know how many products a customer bought, or when he made individual purchases.
- With the order table, each time a product is purchased, there is a record in the order table about when and how many products.
- The easiest way to generate a bridge table is by creating an intermediate table with only two columns that together form the composite key.
- For example, for the `student-course` relationship, this could be a table `enrolment` with the primary key `(student_id, course_id)`. Such a table could record all kinds of enrolment data.

☐ You see another example in the bank ERD 7: can you find the bridge table with the composite key?

Object relations mapper

- Ponyorm is a (free) example, and Lucidchart and other apps also offer this option - translate an ERD model into a database.
- The other way around is also possible and common: visualizing an existing database. Recommended for SQLite: DBeaver (open source).

References

- Birkenkrahe (April 5, 2022). Drawing ERD in draw.io [video]. URL: youtu.be/gCranxLqZDI.
- Dybka (August 2, 2014). Chen Notation [blog]. URL: vertabelo.com.

- Dybka (August 31, 2016). Crow's Foot Notation [blog]. URL: vertabelo.com.
- Lucidchart (2017). Entity Relationship Diagram (ERD) Tutorial Part 1 [video]. URL: youtu.be/QpdhBUYk7Kk.