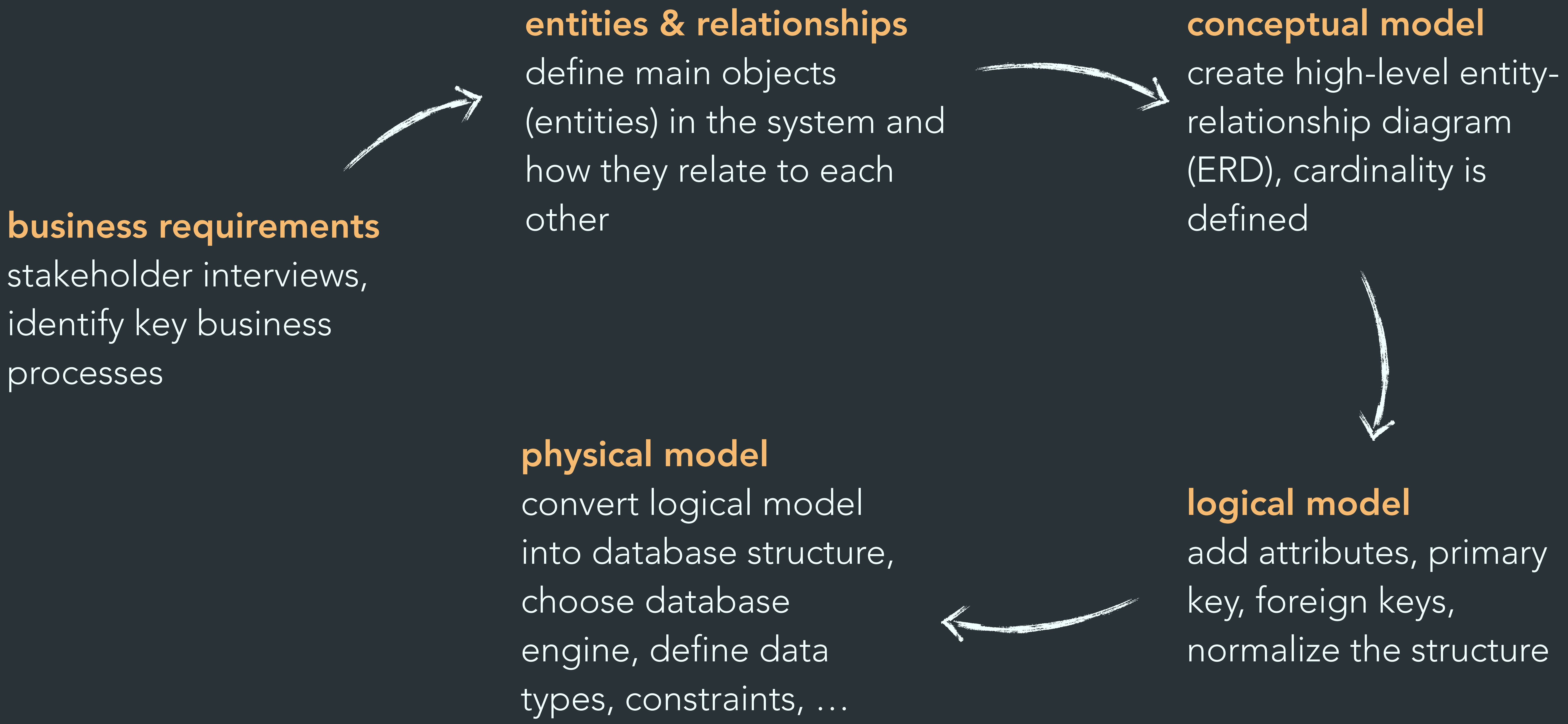
kokchun giang

going from a conceptual business-centric modeling to **logical** and **physical** modeling that are more technical

the **data modeling** journey for transactional data



business requirements
stakeholder interviews, identify key business processes

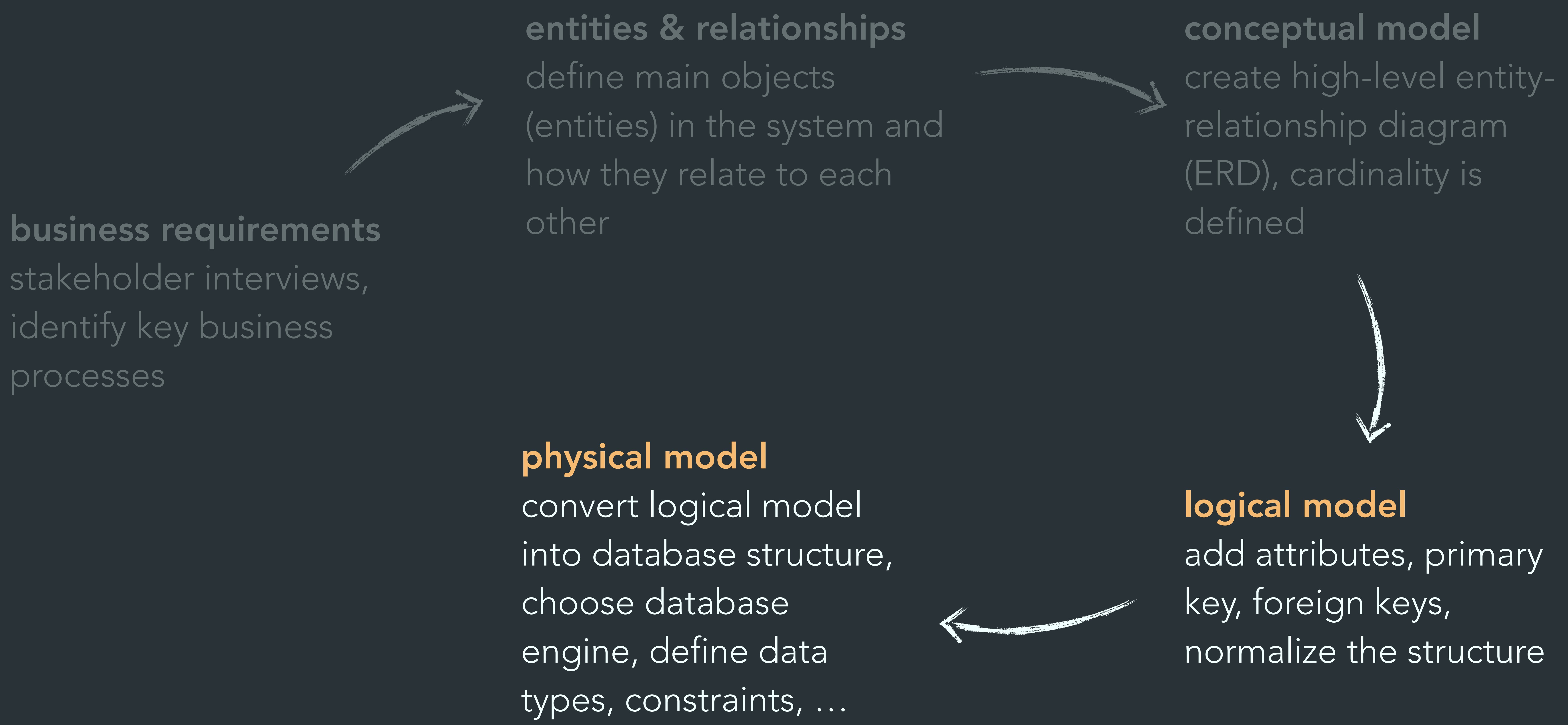
entities & relationships
define main objects (entities) in the system and how they relate to each other

conceptual model
create high-level entity-relationship diagram (ERD), cardinality is defined

logical model
add attributes, primary key, foreign keys, normalize the structure

physical model
convert logical model into database structure, choose database engine, define data types, constraints, …

the **data modeling** journey for transactional data



business requirements
stakeholder interviews, identify key business processes

entities & relationships
define main objects (entities) in the system and how they relate to each other

conceptual model
create high-level entity-relationship diagram (ERD), cardinality is defined

logical model
add attributes, primary key, foreign keys, normalize the structure

physical model
convert logical model into database structure, choose database engine, define data types, constraints, …

remember the **business requirements** for ezecream

customers should be able to browse and order ice cream flavors online

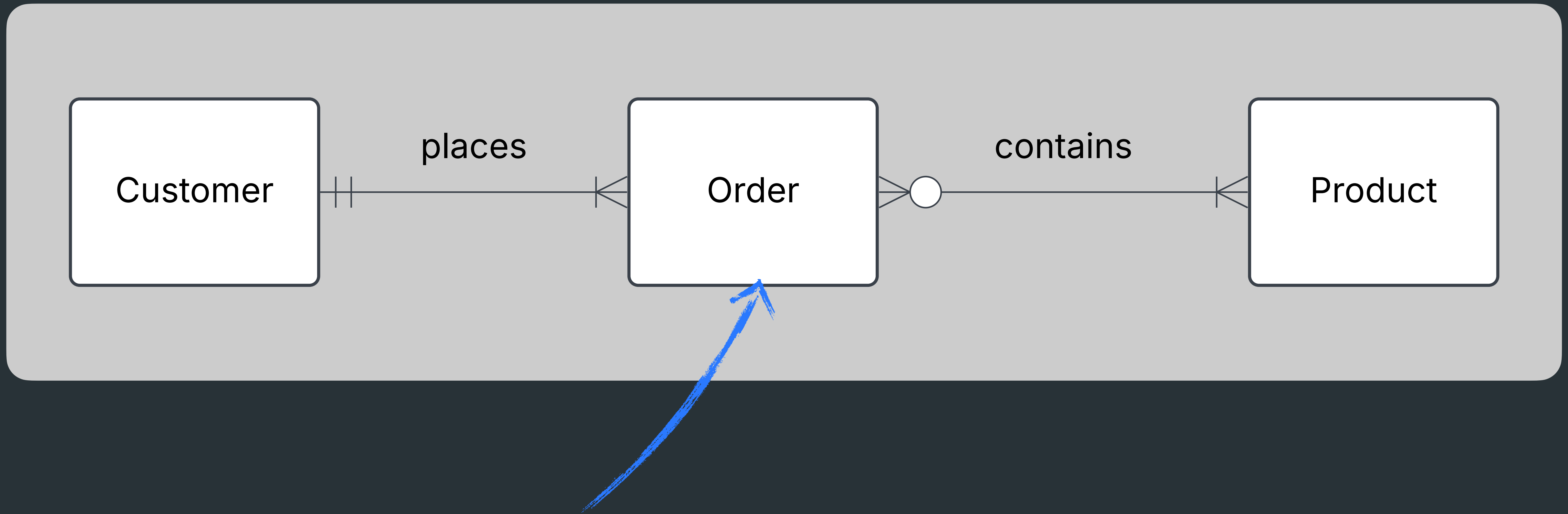
each order should contain one or more ice cream flavors

the system should store order details, including order date and total price

customers should provide their name, contact details, and delivery address

each ice cream flavor should have a name, price, and availability status

a **conceptual ERD** for ezecream using crows foot notation

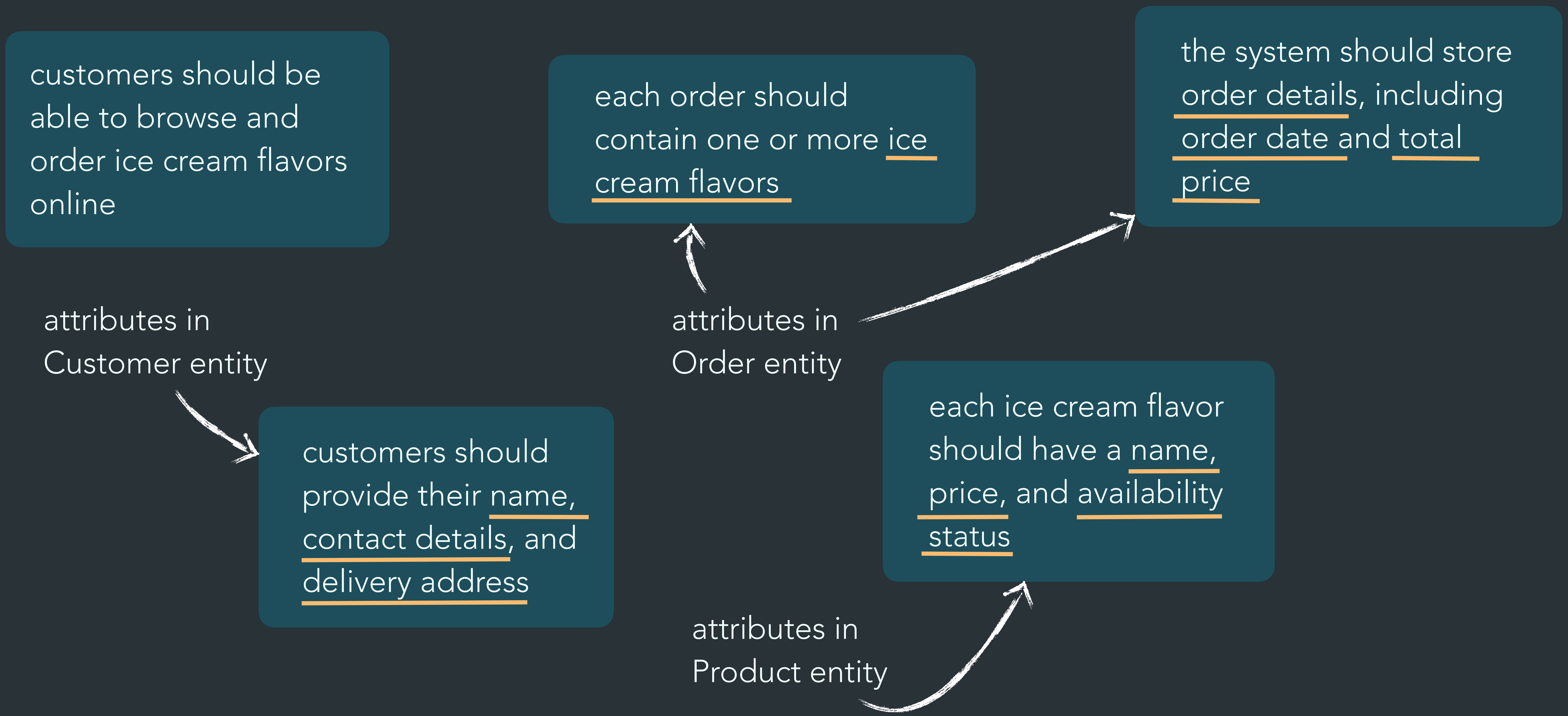


composite entity
an entity that resolves many-to-many relationship between two entities

replaces a many-to-many relationship with two one-to-many relationships

it acts as a bridge table with foreign keys that references primary keys in the related tables

identify the **entities & relationships** from the requirements



customers should be able to browse and order ice cream flavors online

customers should provide their name, contact details, and delivery address

attributes in Customer entity

each order should contain one or more ice cream flavors

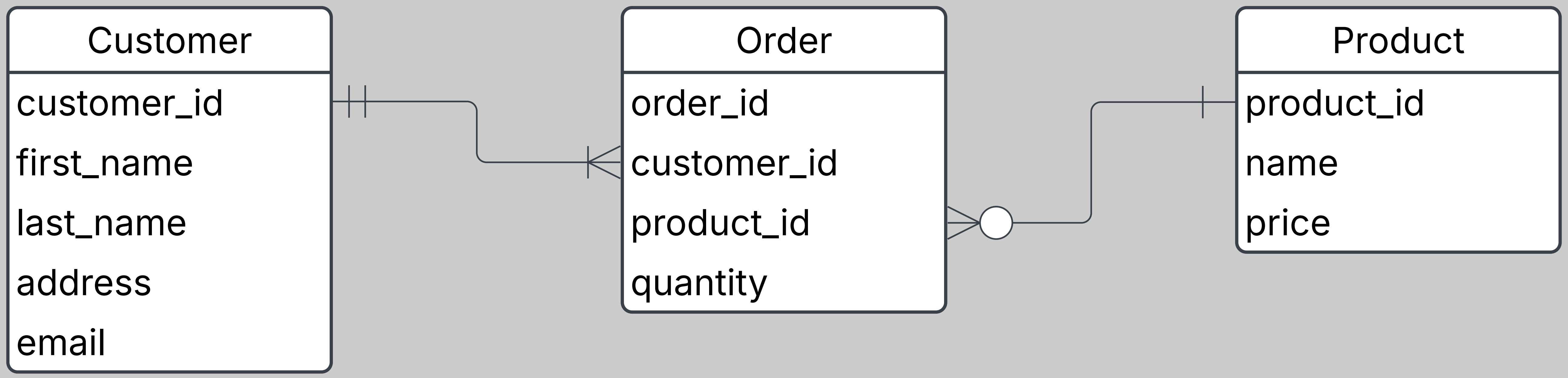
the system should store order details, including order date and total price

attributes in Order entity

each ice cream flavor should have a name, price, and availability status

attributes in Product entity

the **logical data model** for ezecream



Customer
customer_id
first_name
last_name
address
email

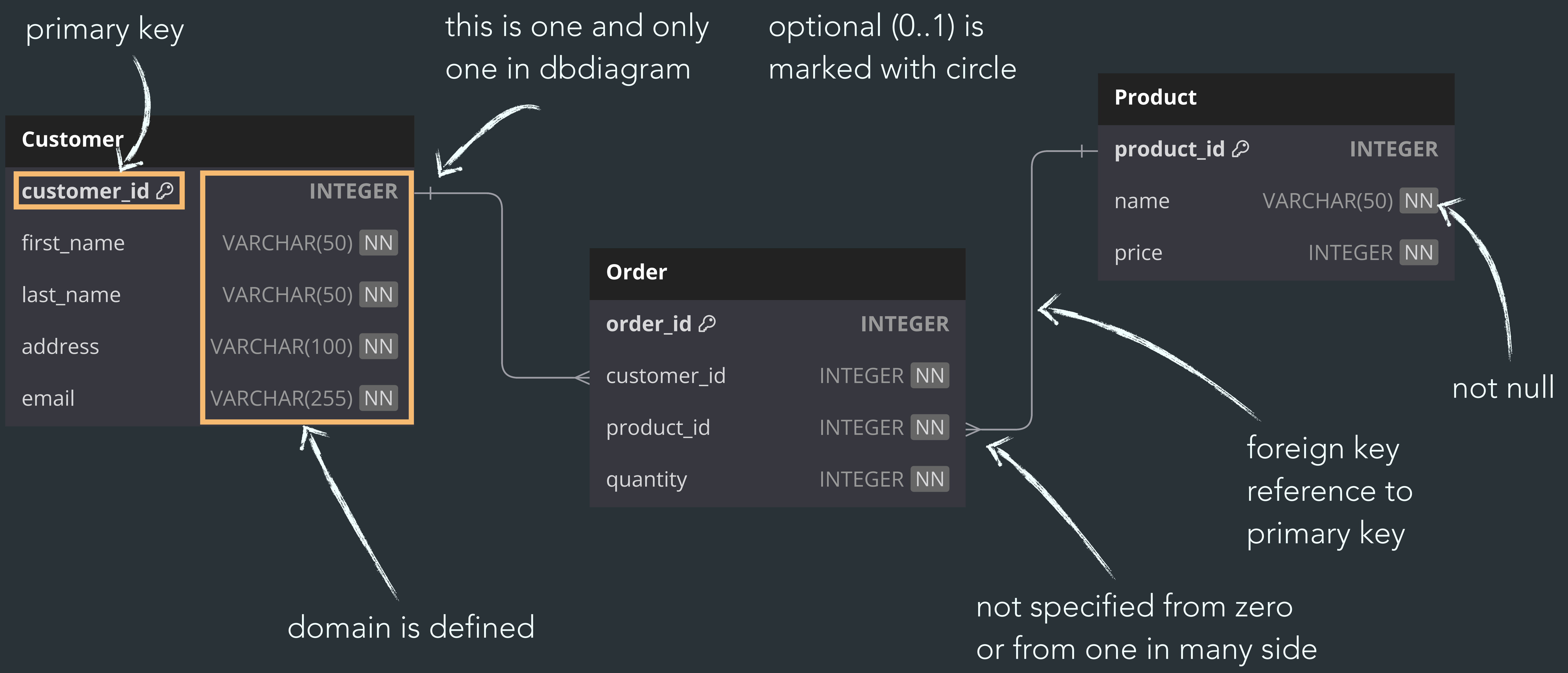
Order
order_id
customer_id
product_id
quantity

Product
product_id
name
price

difference from conceptual model is that logical model also contains attributes

this makes the model more specialized while conceptual model reflects the real world more generally

the **physical data model** for ezecream using dbdiagram



Customer		
customer_id ⚷	**INTEGER**	
first_name	VARCHAR(50)	NN
last_name	VARCHAR(50)	NN
address	VARCHAR(100)	NN
email	VARCHAR(255)	NN

Order		
order_id ⚷	**INTEGER**	
customer_id	INTEGER	NN
product_id	INTEGER	NN
quantity	INTEGER	NN

Product		
product_id ⚷	**INTEGER**	
name	VARCHAR(50)	NN
price	INTEGER	NN

primary key

this is one and only one in dbdiagram

optional (0..1) is marked with circle

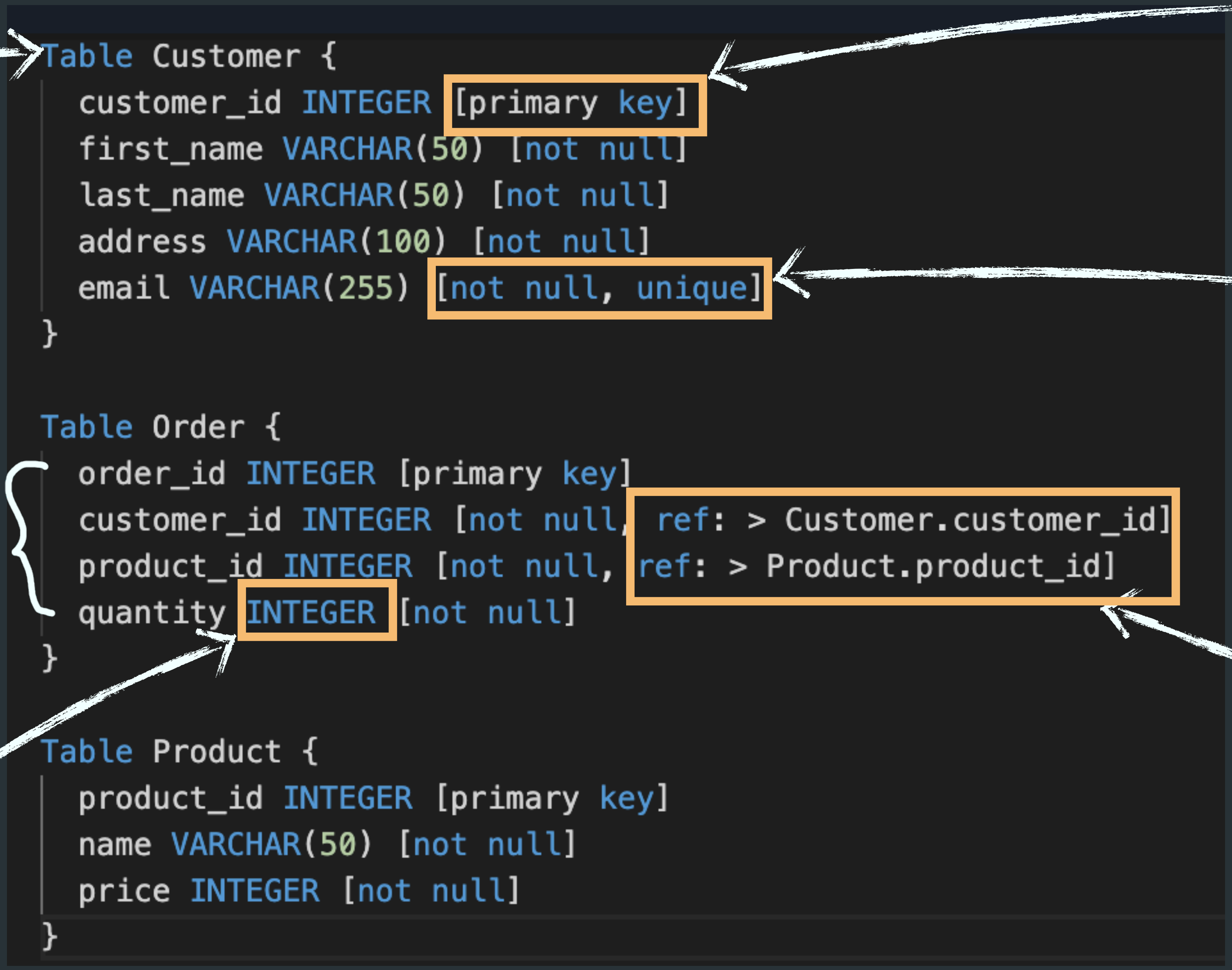
not null

foreign key reference to primary key

not specified from zero or from one in many side

domain is defined

database modeling language **dbml** is used in dbdiagram to do physical data modeling



```
Table Customer {
  customer_id INTEGER [primary key]
  first_name VARCHAR(50) [not null]
  last_name VARCHAR(50) [not null]
  address VARCHAR(100) [not null]
  email VARCHAR(255) [not null, unique]
}

Table Order {
  order_id INTEGER [primary key]
  customer_id INTEGER [not null, ref: > Customer.customer_id]
  product_id INTEGER [not null, ref: > Product.product_id]
  quantity INTEGER [not null]
}

Table Product {
  product_id INTEGER [primary key]
  name VARCHAR(50) [not null]
  price INTEGER [not null]
}
```

Customer entity

primary key constraint

not null and unique constraints

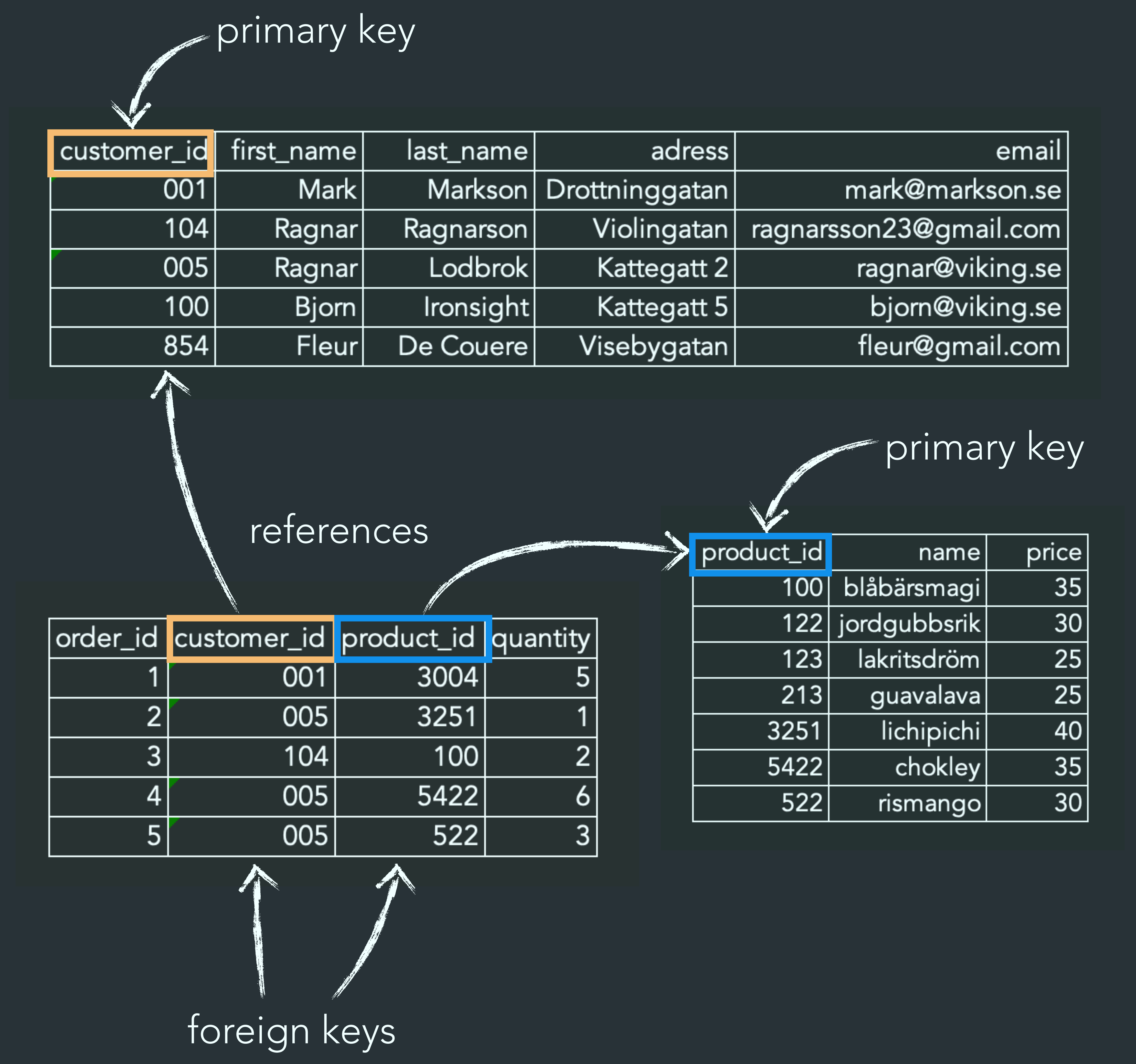
unfortunately unique constraint isn’t implemented in dbdiagram yet

attributes

domain constraint

foreign keys, referencing primary keys in related tables

the **actual tables** could look like this in the end



customer_id	first_name	last_name	adress	email
001	Mark	Markson	Drottninggatan	mark@markson.se
104	Ragnar	Ragnarson	Violingatan	ragnarsson23@gmail.com
005	Ragnar	Lodbrok	Kattegatt 2	ragnar@viking.se
100	Bjorn	Ironsight	Kattegatt 5	bjorn@viking.se
854	Fleur	De Couere	Visebygatan	fleur@gmail.com

order_id	customer_id	product_id	quantity
1	001	3004	5
2	005	3251	1
3	104	100	2
4	005	5422	6
5	005	522	3

product_id	name	price
100	blåbärsmagi	35
122	jordgubbsrik	30
123	lakritsdröm	25
213	guavalava	25
3251	lichipichi	40
5422	chokley	35
522	rismango	30

foreign keys in the bridge table references primary keys in the related tables

can you figure out how much Ragnar Lodbrok needs to pay in total?