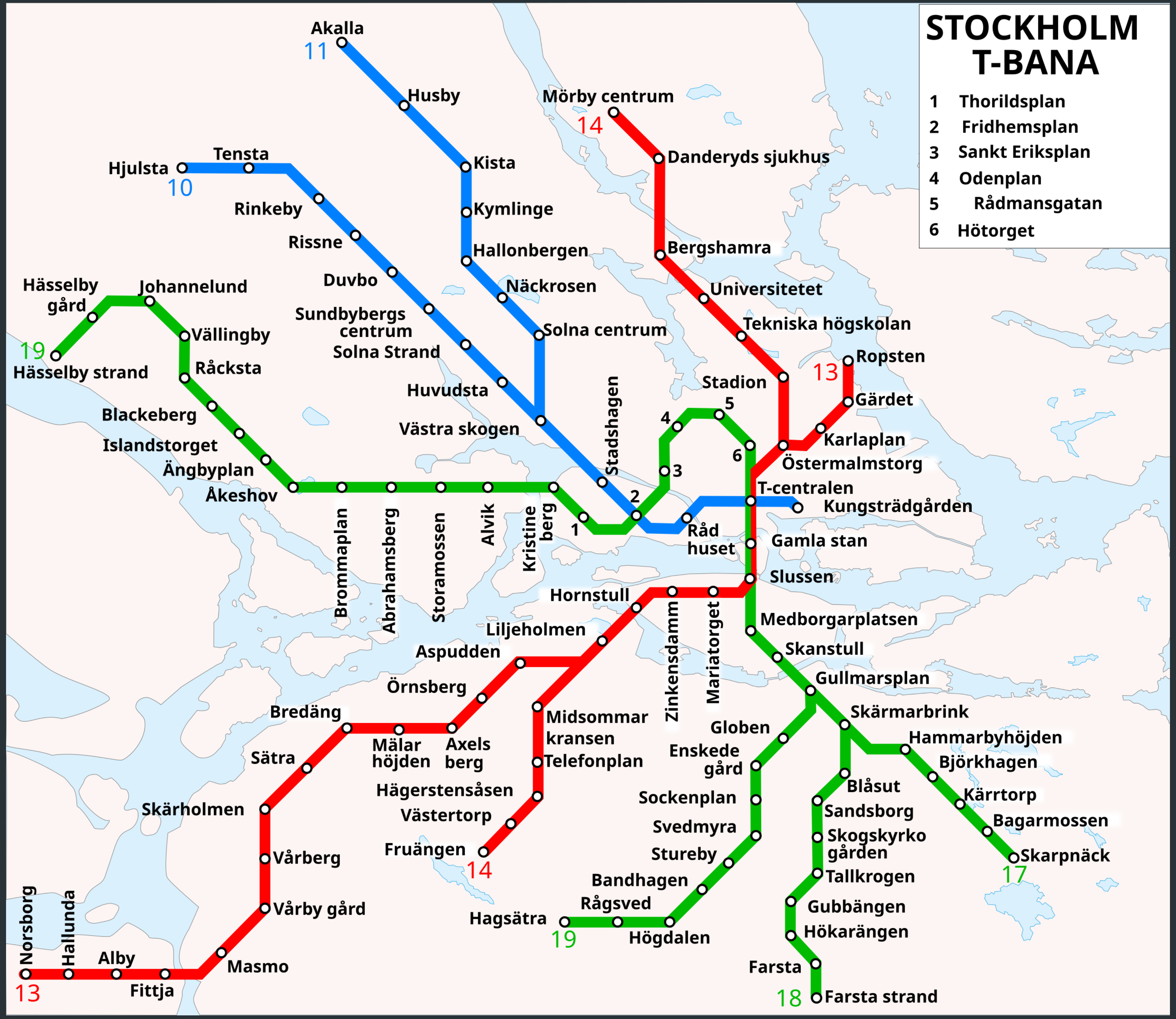
kokchun giang

creating a **conceptual** model of a business requirements is the first step in data modeling

all **models** are wrong, but some are useful - George Box



railway map is a model with some simplifications

not drawn to scale

don’t represent exact geographical positions

straight lines or fixed angles

remove unnecessary details

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, …

the reason for doing **conceptual modeling**

bridge the gap between business and technical teams

help stakeholders understand how data is structured

ensure business rules correctly reflected in the database

foundation for database design

business requirements for ezecream could look like this

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

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

Customer entity

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

Product entity

Customer can place one or more Orders

Order entity

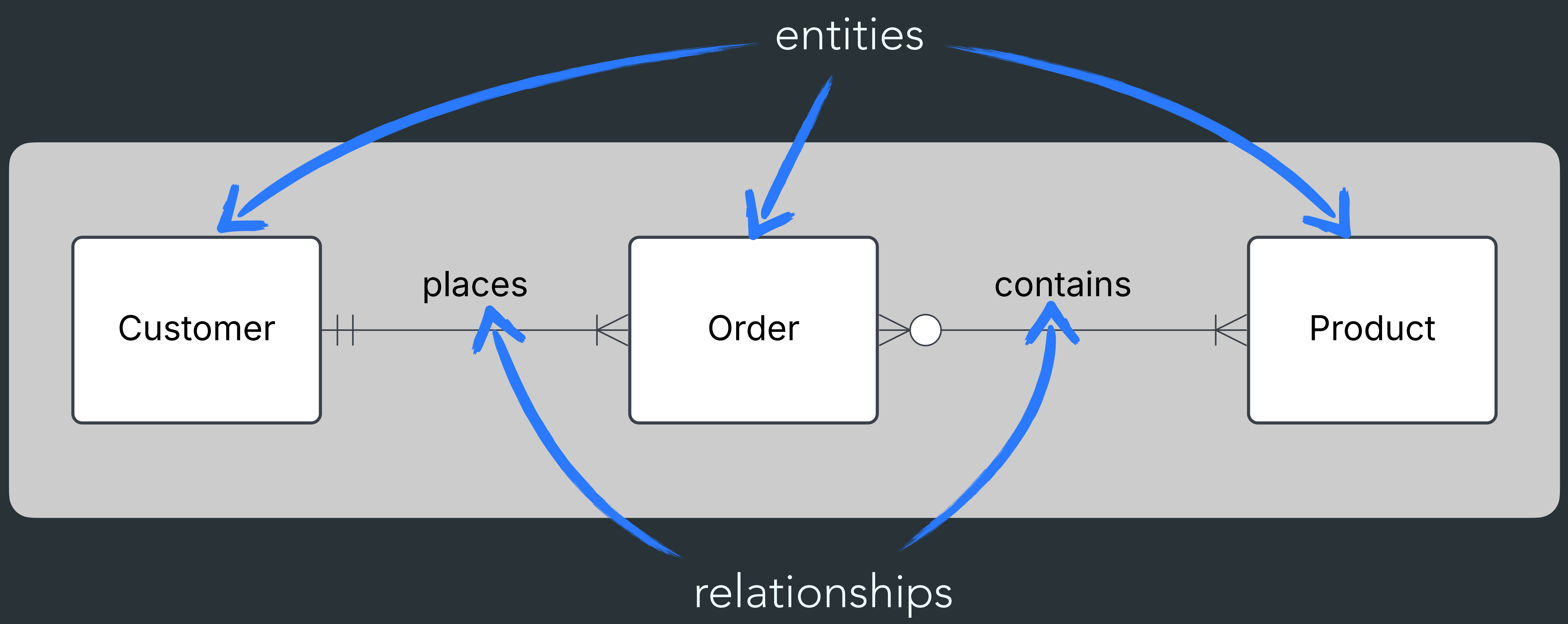
each order should contain one or more ice cream flavors

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



Customer can place one or more Orders

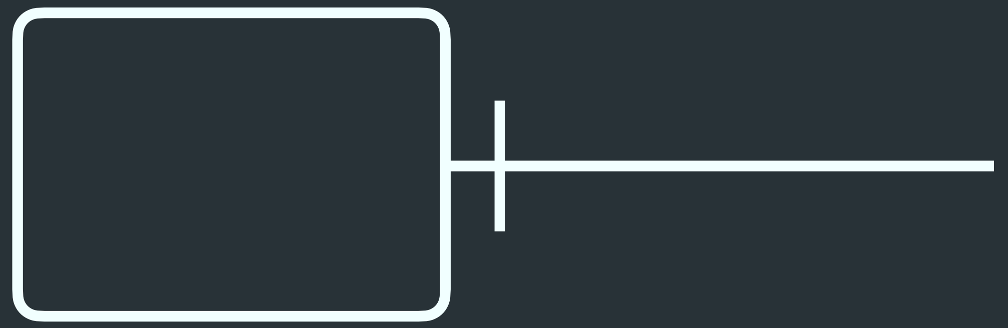
an Order can be placed by one and only one Customer

an Order contains one or more Products

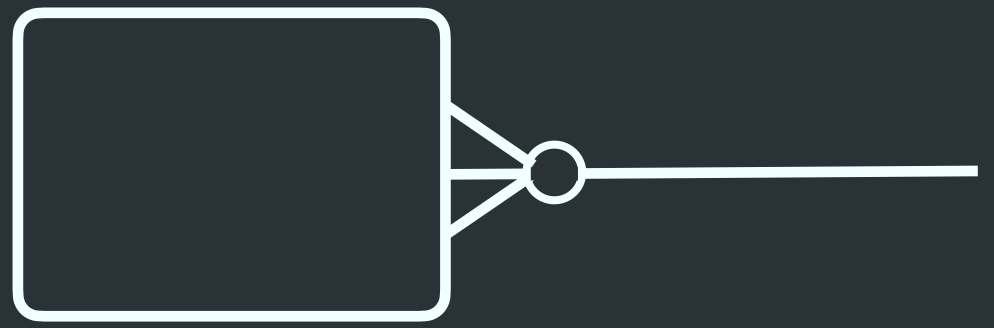
a Product is contained by 0 or more Orders

lets break down the cardinality symbols in **crows foot notation**

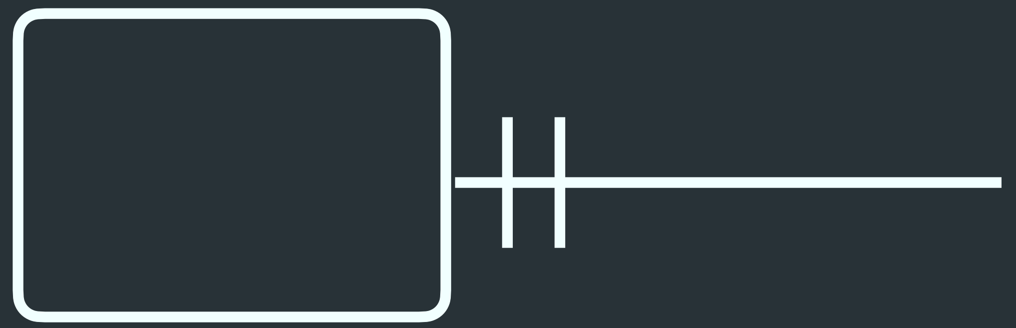
zero or one



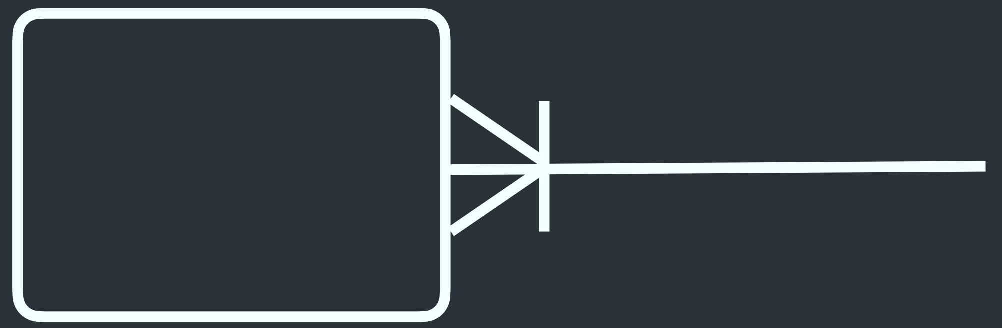
zero or many



one and only one



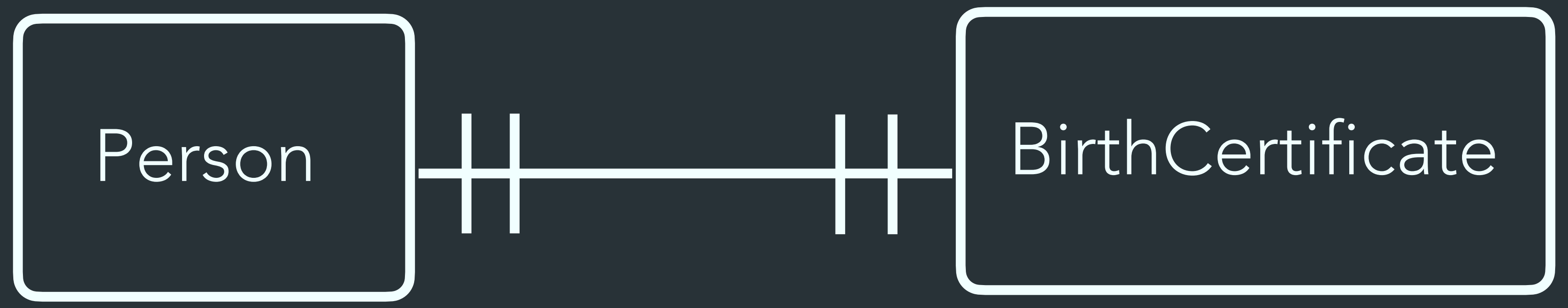
one or many



mapping **cardinality** between two entities

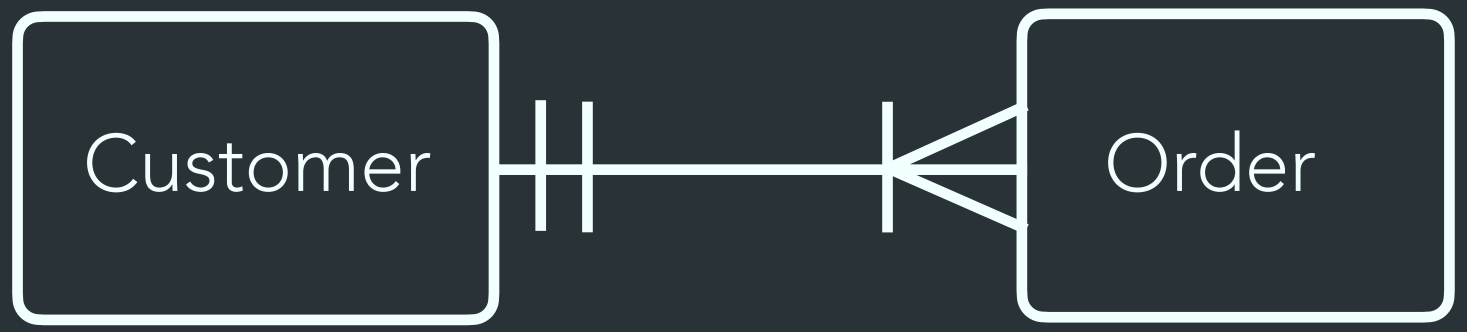
cardinality is how many instances of one entity that can be associated with how many instances of another entity

one-to-one

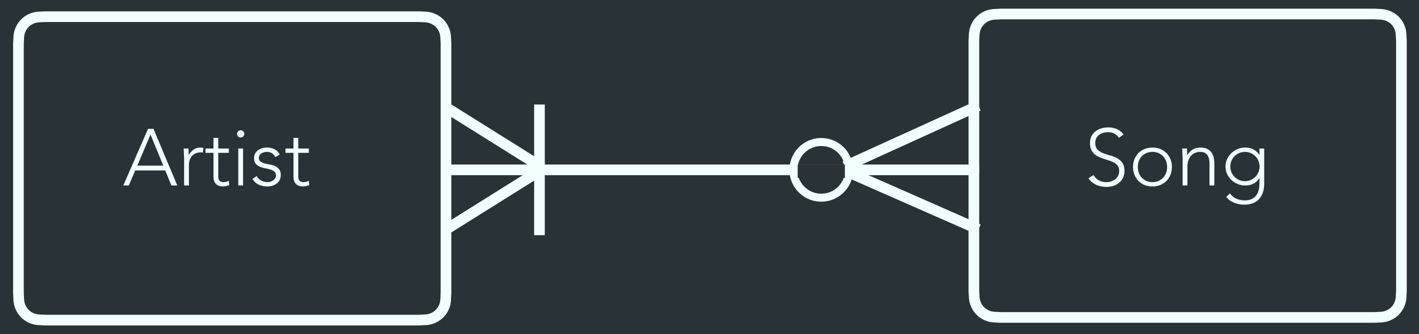


one-to-one is uncommon

one-to-many

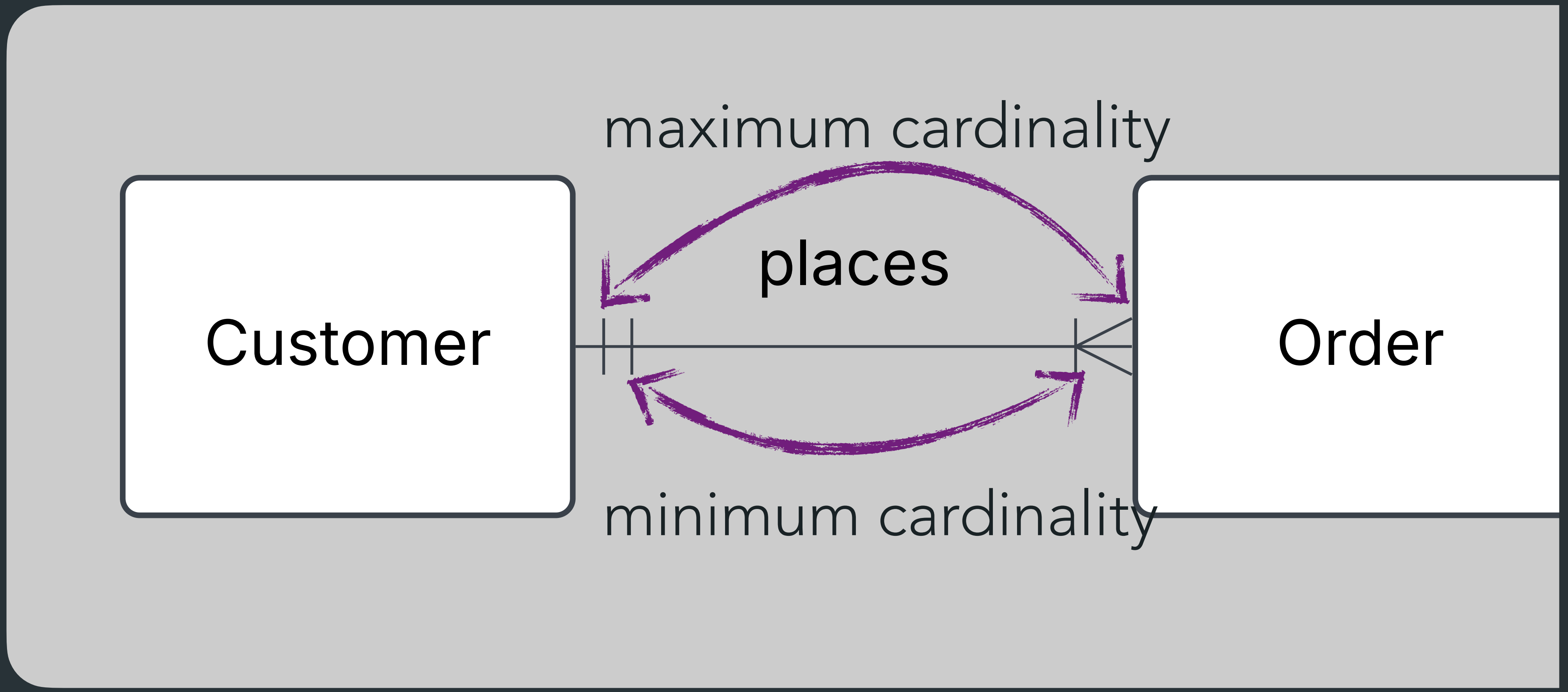


many-to-many



many-to-many can't be implemented directly

minimum and maximum **cardinalities**



Customer can place **one or more** Orders

an Order can be placed by **one and only one** Customer

this is a **one-to-many** relationship

entities, relationships and cardinalities will affect the implementation of the database tables