MUNI
FI

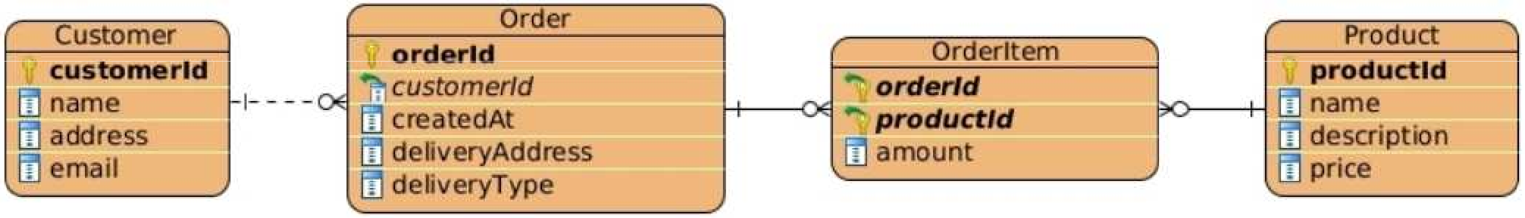
PB007 Week 07

Samuel Sabo

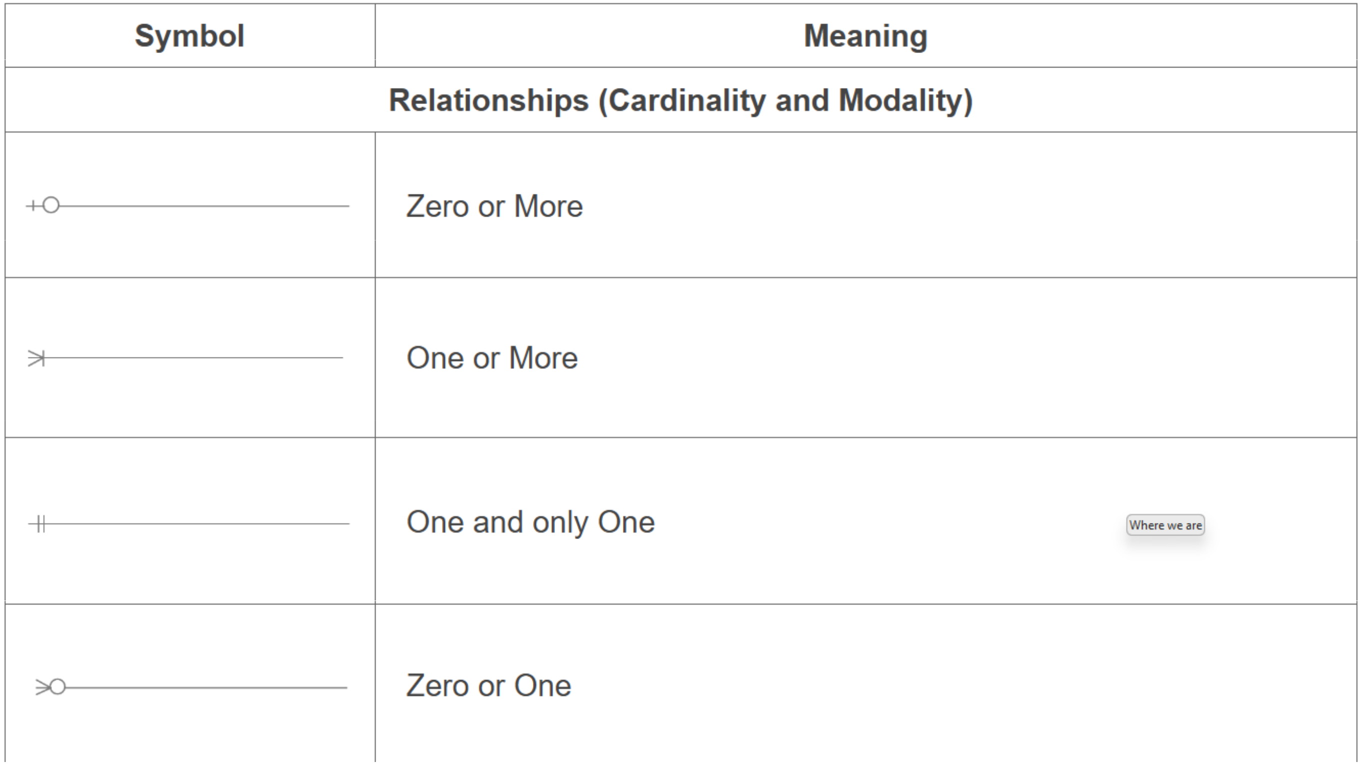
FACULTAS ARTIS INFORMATICÆ • UNIVERSITAS MASARYKIANA •

Entity-Relationship Diagram

- Entity-Relationship Diagram (ERD) is a data model representing the logical structure of a database
- Its main components are:
  - Entities (Entity types)
  - Relations (Relation types)
  - Attributes (Attribute types)



Notation



Symbol	Meaning
Relationships (Cardinality and Modality)	
+○—	Zero or More
>|—	One or More
‖—	One and only One
>○—	Zero or One

Where we are

Class diagram vs. ERD

Class diagram

- models both data and operations
- classes relate to different types of relationships (associations, dependencies, generalization, aggregation, composition)
- usually represent business domain concepts

Entity-relationship model

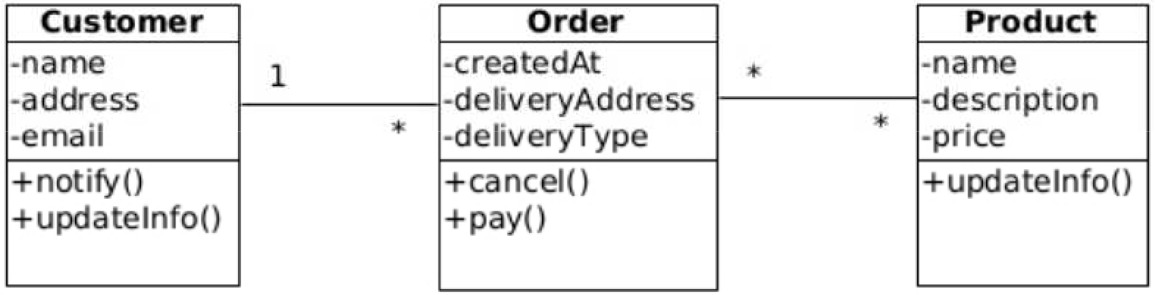
- models data only
- contains only simple relationships
- represents database tables

Object-Relational Mapping (ORM)

- **Object-Relational Mapping (ORM)** is a technique for conversion of data between relational database and object-oriented language.
  - relational database ⇔ object-oriented language
  - (persistent) class ⇔ entity type (table)
  - object ⇔ entity (table row)
  - class attribute ⇔ entity attribute (table column)
  - association/aggregation/composition ⇔ relation (connection via foreign keys)
  - inheritance ⇔ 1:1 mapping, merge to superclass, propagation to subclasses
- Note:
  - not all classes must be persistent

Object-Relational Mapping (ORM)

Class diagram:



Customer
-name
-address
-email
+notify()
+updateInfo()

1

*

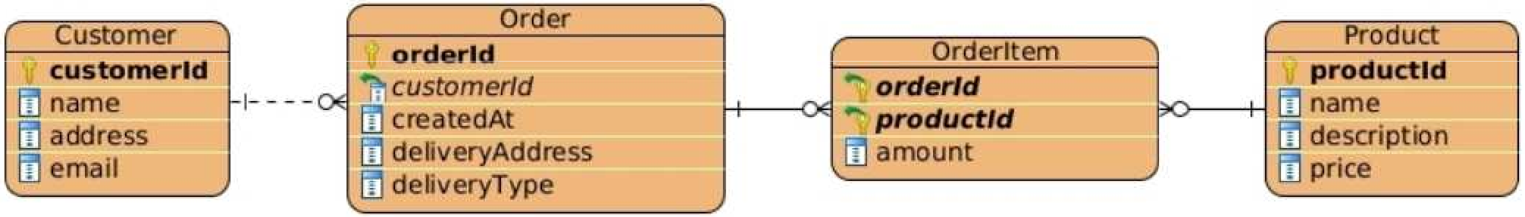
Order
-createdAt
-deliveryAddress
-deliveryType
+cancel()
+pay()

*

*

Product
-name
-description
-price
+updateInfo()

ERD:



Customer
customerId
name
address
email

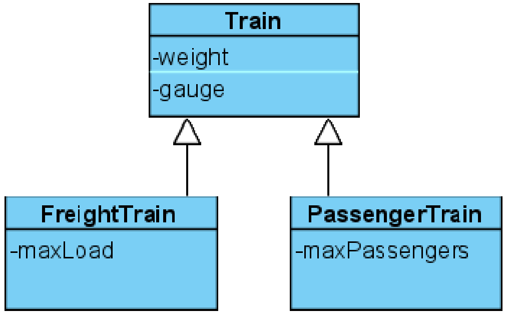
Order
orderId
customerId
createdAt
deliveryAddress
deliveryType

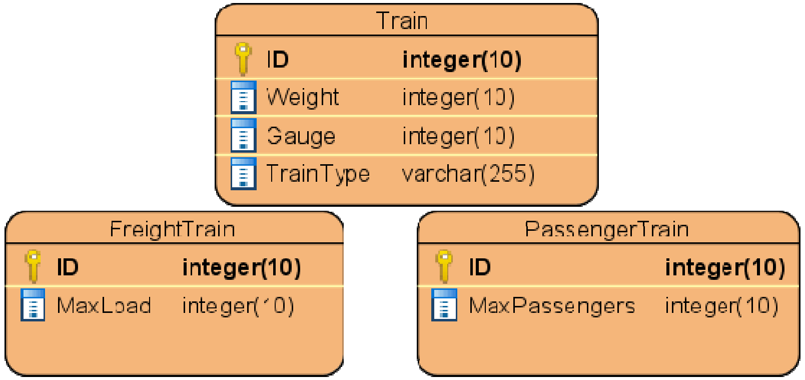
OrderItem
orderId
productId
amount

Product
productId
name
description
price

ORM - Inheritance - 1:1 mapping

- Each class becomes a table
- An attribute (*TrainType*) differentiates the subclass type
- One object instance in multiple tables
  - more difficult data access





Train

ID	**integer(10)**
Weight	integer(10)
Gauge	integer(10)
TrainType	varchar(255)

FreightTrain

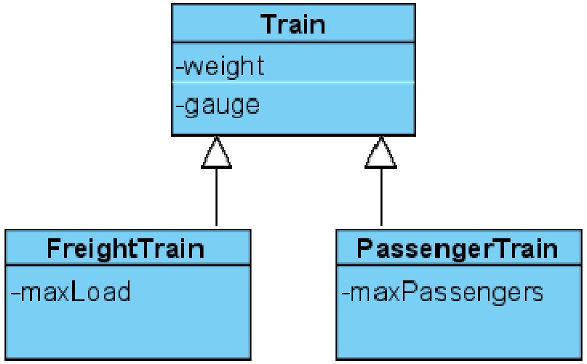
ID	**integer(10)**
MaxLoad	integer(10)

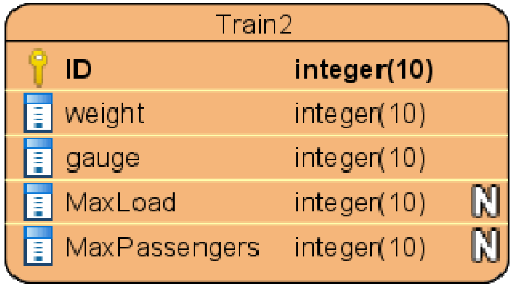
PassengerTrain

ID	**integer(10)**
MaxPassengers	integer(10)

ORM - Inheritance - Merge to superclass

- All attributes in **one table**
- Some attributes will have NULL value
  - breaks 4.NF
- Suitable for "big parents, small children"
  - superclasses with many attributes
  - only a few subclasses with only a few attributes



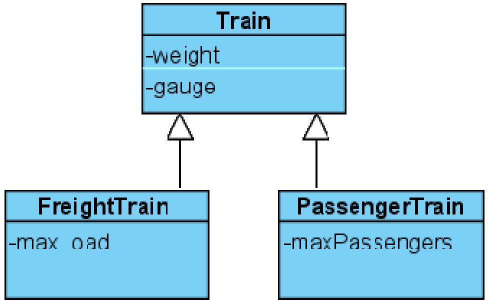


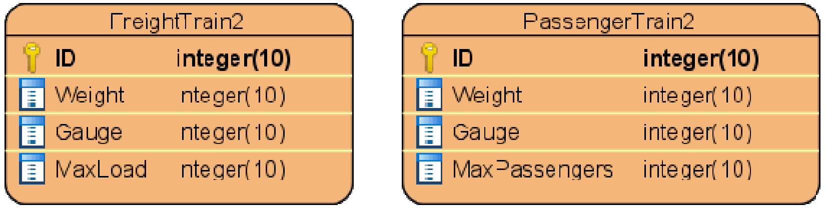
Train2

🔑	**ID**	**integer(10)**	
▤	weight	integer(10)	
▤	gauge	integer(10)	
▤	MaxLoad	integer(10)	N
▤	MaxPassengers	integer(10)	N

ORM - Inheritance - Propagation to subclasses

- Superclass attributes are **copied to subclass tables**
  - many similar tables
- Suitable for "small parents, big children"
  - superclasses with only a few attributes
  - many subclasses with many attributes





FreightTrain2

🔑	**ID**	**integer(10)**
▤	Weight	nteger(10)
▤	Gauge	nteger(10)
▤	MaxLoad	nteger(10)

PassengerTrain2

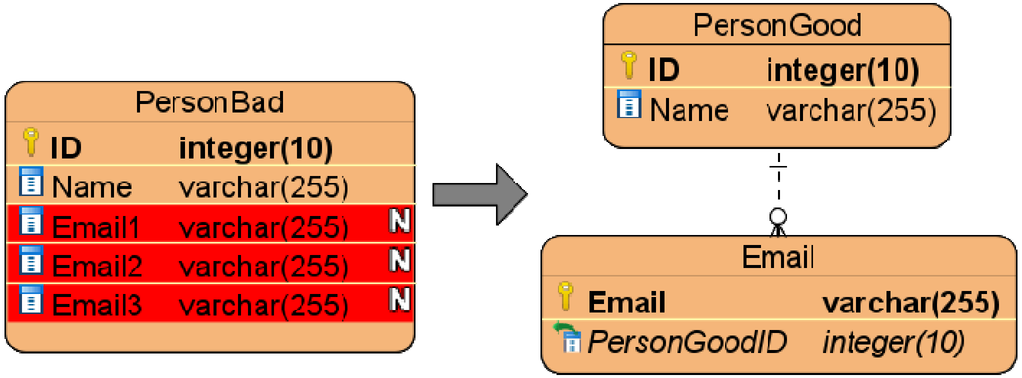
🔑	**ID**	**integer(10)**
▤	Weight	integer(10)
▤	Gauge	integer(10)
▤	MaxPassengers	integer(10)

Normal forms

- Normal forms are used to achieve good database design.
- They help with:
  - elimitnation of repetitive data
  - reduce table complexity
  - prevent anomalies (for update, insert, delete)

1NF

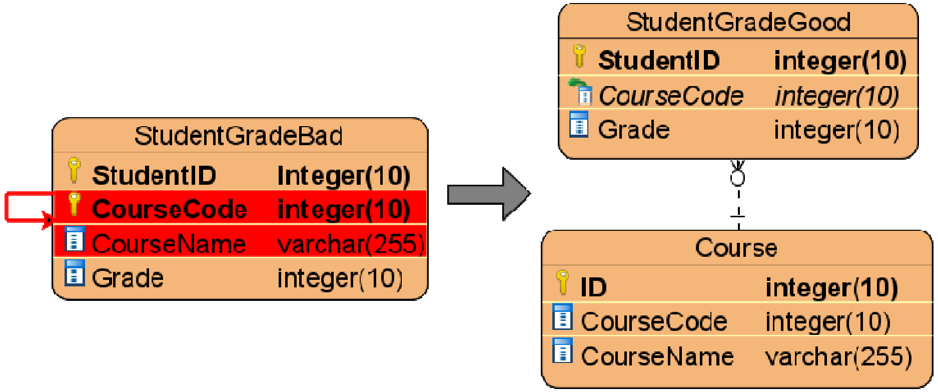
- Satisfies 0. NF
- Each attribute is **atomic**
- Problem: composed attributes



2NF

- Satisfies 1. NF
- No **partial dependency**
  - each non-key attribute is fully dependent on every candidate key
- Problem: inferring non-key attributes

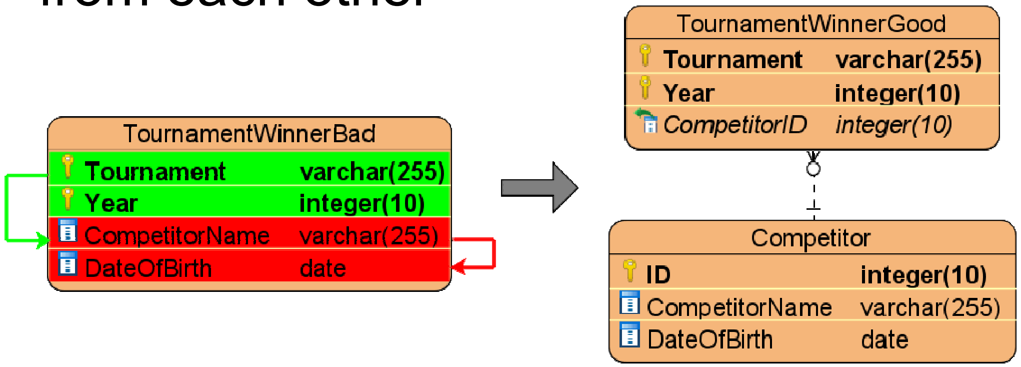
  from only a part of candidate key



3NF

- Satisfies 2. NF
- No **transitive dependency**
  - each non-key attribute is dependent on candidate keys only
  - => non-key attributes are mutually independent
- Problem: inferring non-key attributes

  from each other



TournamentWinnerBad	
Tournament	**varchar(255)**
Year	**integer(10)**
CompetitorName	varchar(255)
DateOfBirth	date

TournamentWinnerGood	
Tournament	**varchar(255)**
Year	**integer(10)**
CompetitorID	*integer(10)*

Competitor	
ID	**integer(10)**
CompetitorName	varchar(255)
DateOfBirth	date

Activity: Game

- Link: source code
- Draw entity/entities for assigned class(es)

Task for this week

- Review the **analytical class diagrams** from the previous session. Fix any problem.
- Create Entity-Relationship Diagram
- Update use case diagram
- Submit this week‘s report in homework vault week07 in format **surname1-surname2-surname3.pdf**