

Modeling Entity Inheritance



Cătălin Tudose

Java Champion, PhD in Computer Science, Java and Web Technologies Expert

www.catalintudose.com | www.linkedin.com/in/catalin-tudose-847667a1

Overview

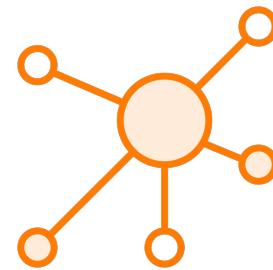
- Inherit from entities and non-entities**
- Working with mapping strategies**
- Working with converters**



Abstract Entity Classes



Annotated with the `@Entity` annotation



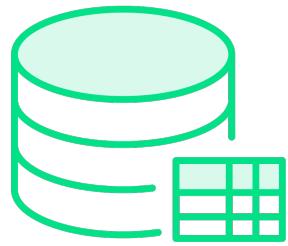
Mapped as an entity



Target of queries



Entities Inheriting from Non-entities



Define state and mapping information common to multiple entities



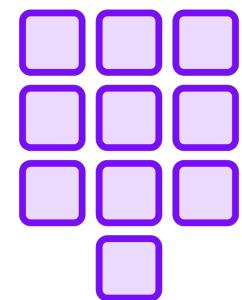
The mapped superclass cannot be queried



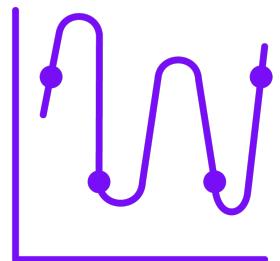
Only unidirectional relationships



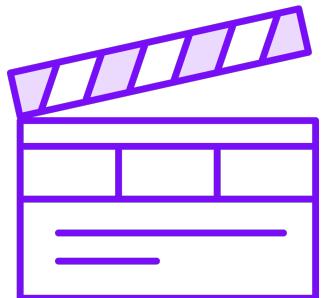
Annotations to Inherit from Non-entities



@MappedSuperclass



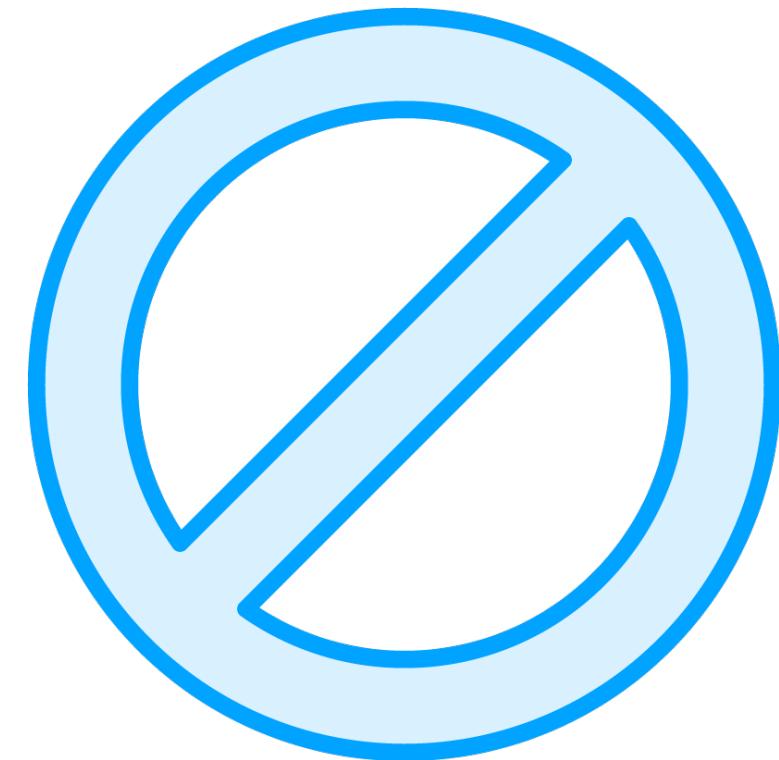
@AttributeOverride



@AssociationOverride



Extending Non-entities with Non-persistent State



Non-persistent state

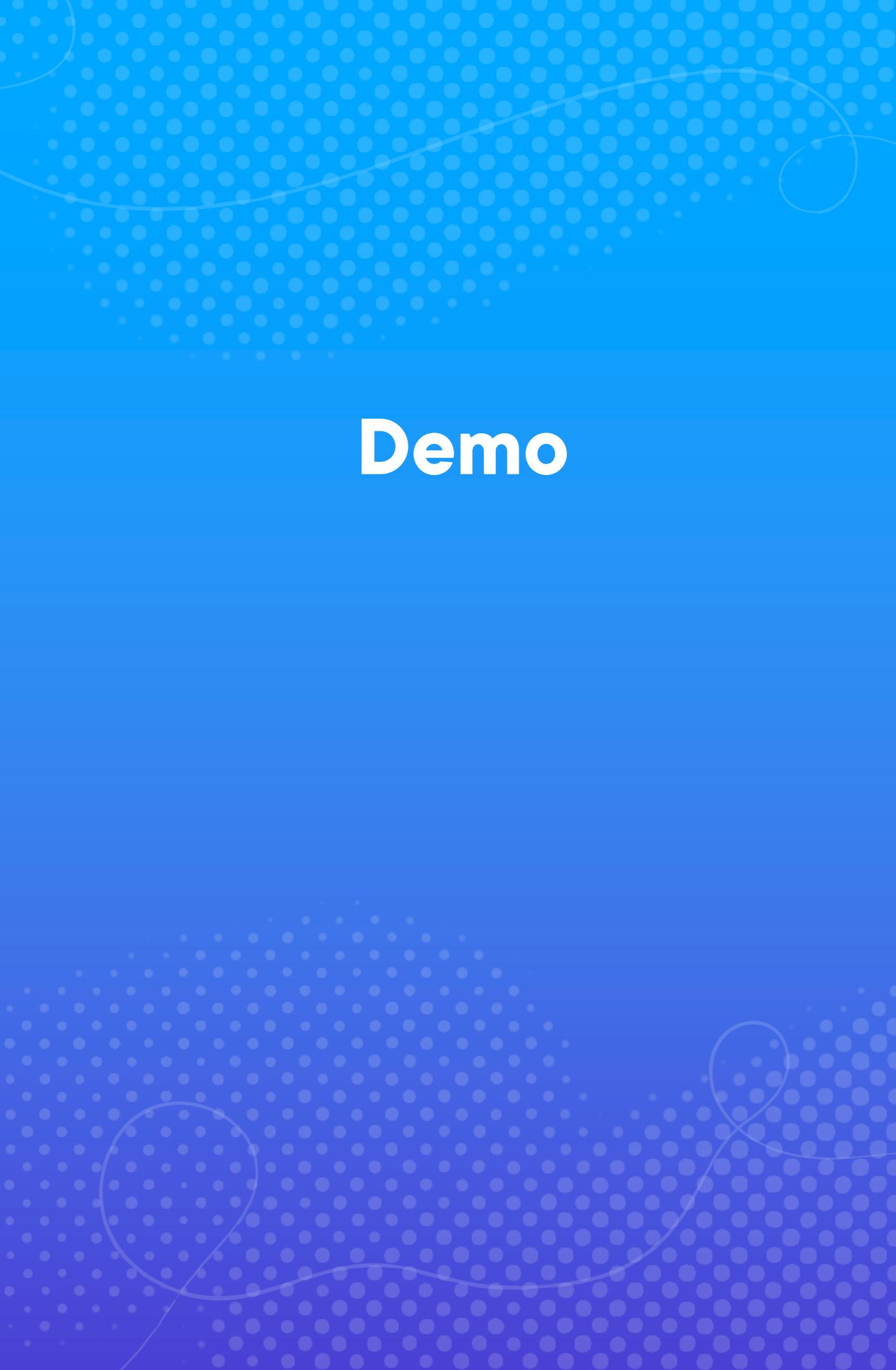


Inherit behavior



Annotations are ignored





Demo

Extend one entity



Demo

Extend one non-entity





Mapping Strategies



Mapping Strategies

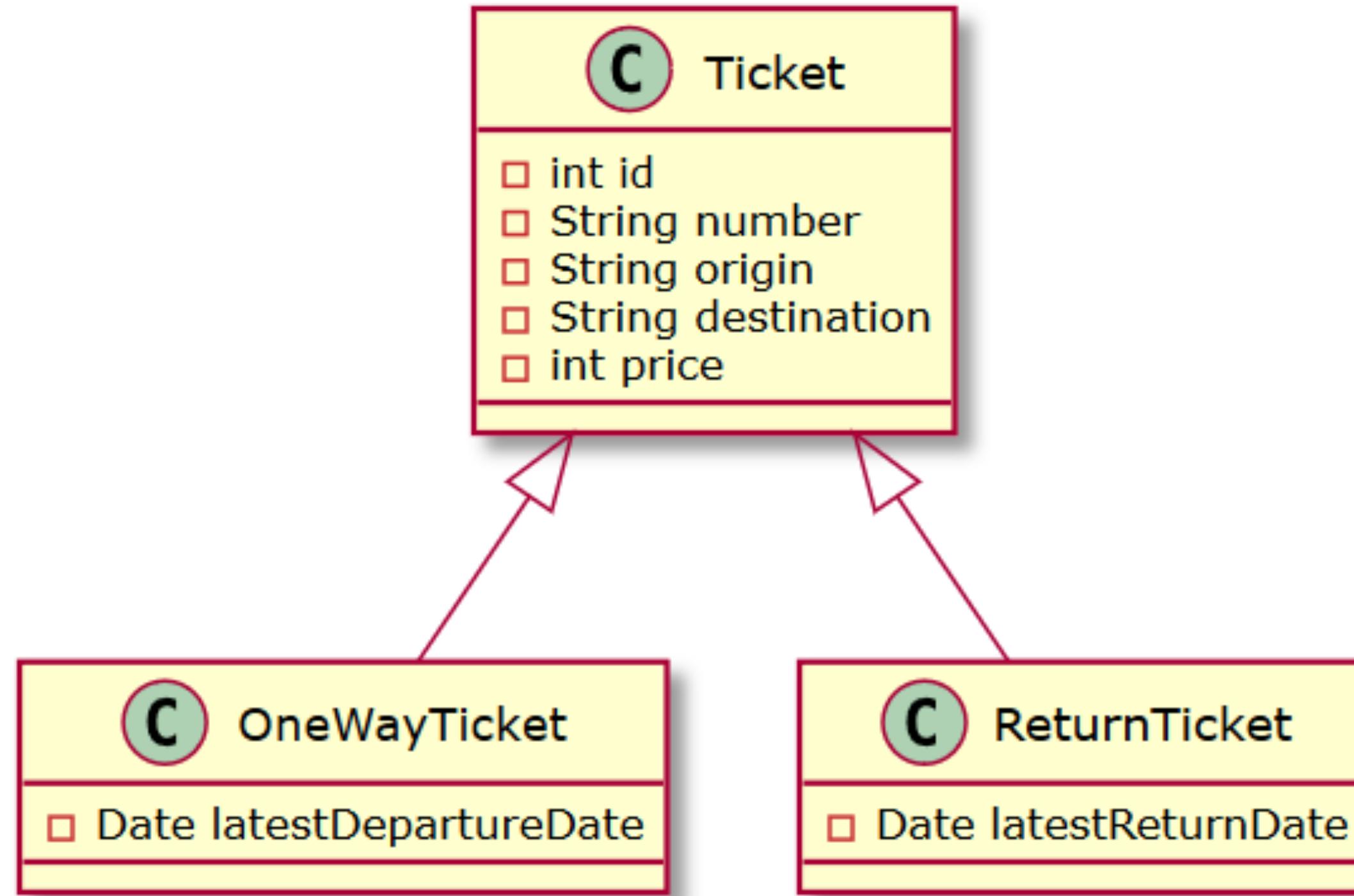
Single table per class hierarchy

Joined subclass strategy

Table per concrete entity class



The Classes Hierarchy

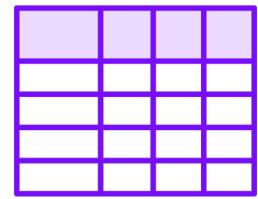


Single Table per Class Hierarchy

TICKETS	
● ID : number	«generated»
TICKET_NUMBER:	text
ORIGIN:	text
DESTINATION :	text
PRICE:	number
LATEST_DEPARTURE_DATE:	date
LATEST_RETURN_DATE:	date



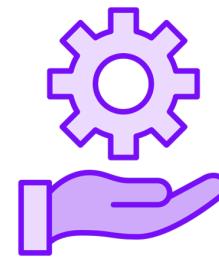
Single Table per Class Hierarchy



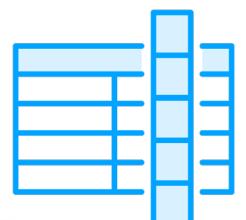
All classes in the hierarchy mapped to a single table



Discriminator column



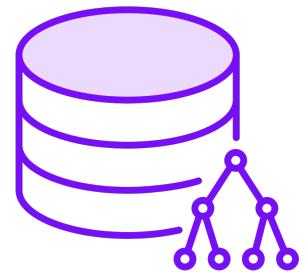
Good support for polymorphic relationships



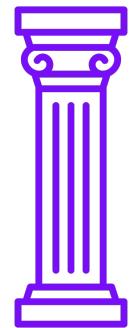
Requires nullable columns



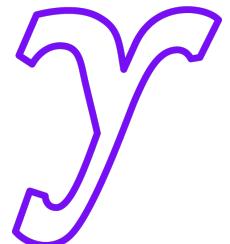
Annotations for Mapping Strategies



@Inheritance



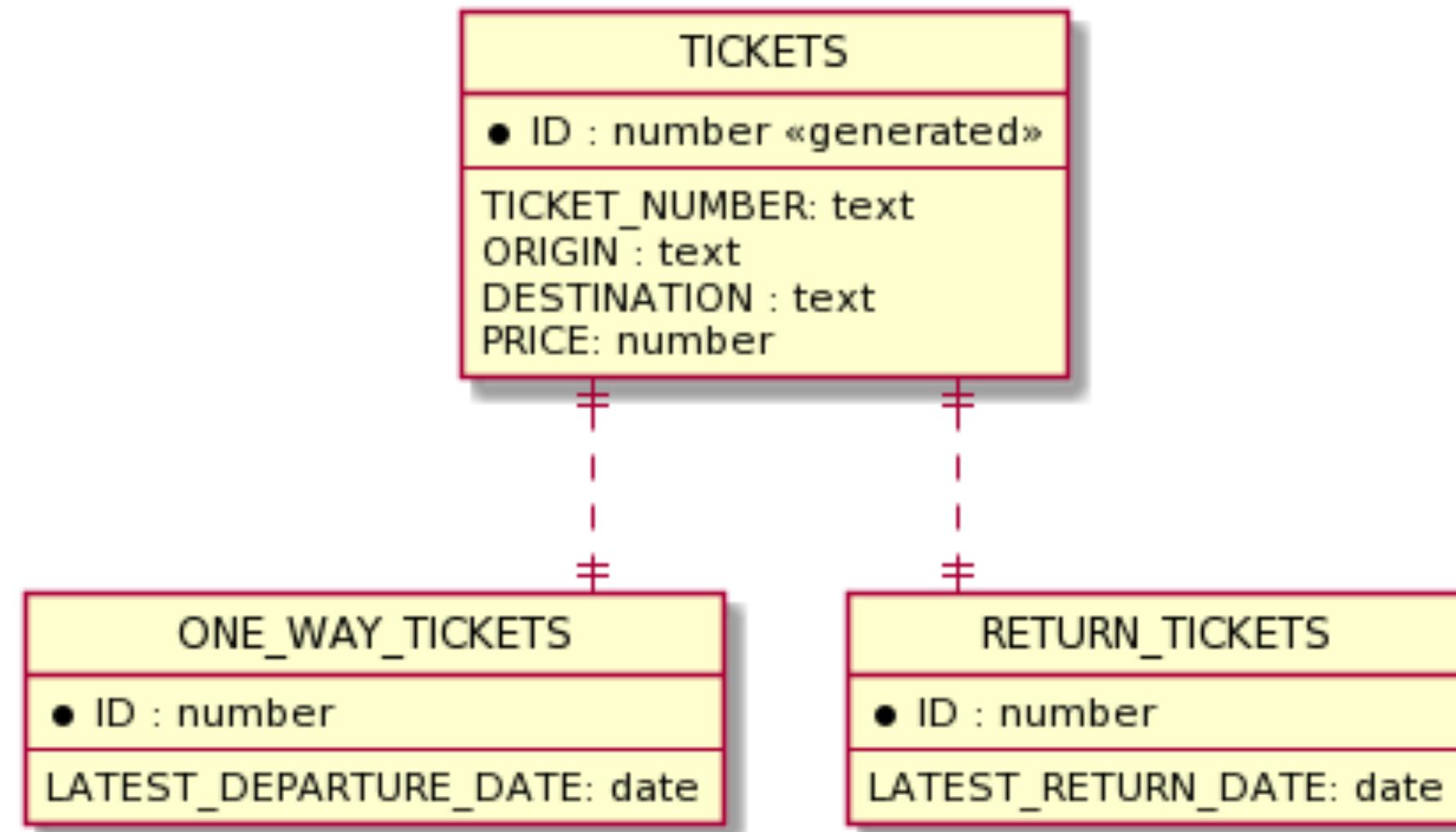
@DiscriminatorColumn



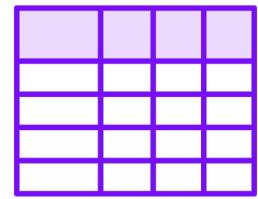
@DiscriminatorValue



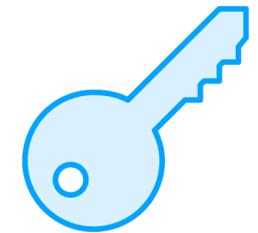
Joined Subclass Strategy



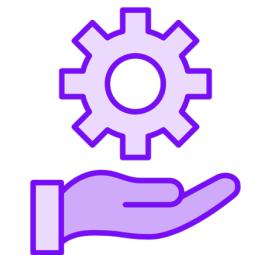
Joined Subclass Strategy



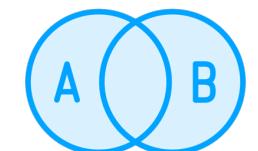
Classes as separate tables



Join through the primary key column



Good support for polymorphic relationships



More join operations between entities



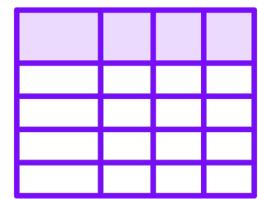
Table per Concrete Class Strategy

ONE_WAY_TICKETS
● ID : number
TICKET_NUMBER: text
ORIGIN : text
DESTINATION : text
PRICE: number
LATEST_DEPARTURE_DATE: date

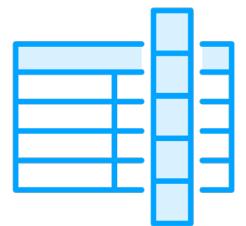
RETURN_TICKETS
● ID : number
TICKET_NUMBER: text
ORIGIN : text
DESTINATION : text
PRICE: number
LATEST_RETURN_DATE: date



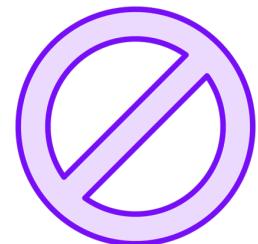
Table per Concrete Class Strategy



Each subclass is mapped to a separate table



All properties mapped to columns of the table for the class



Poor support for polymorphic relationships



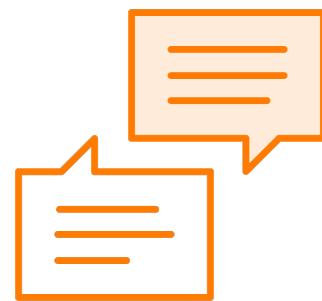
May duplicate the IDs between tables



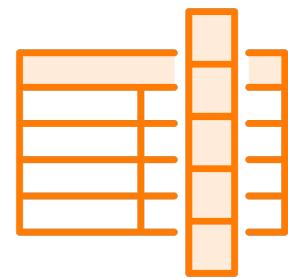
Storing Values Represented Differently



Java boolean \leftrightarrow 0/1, True/False, Yes/No



jakarta.persistence.AttributeConverter



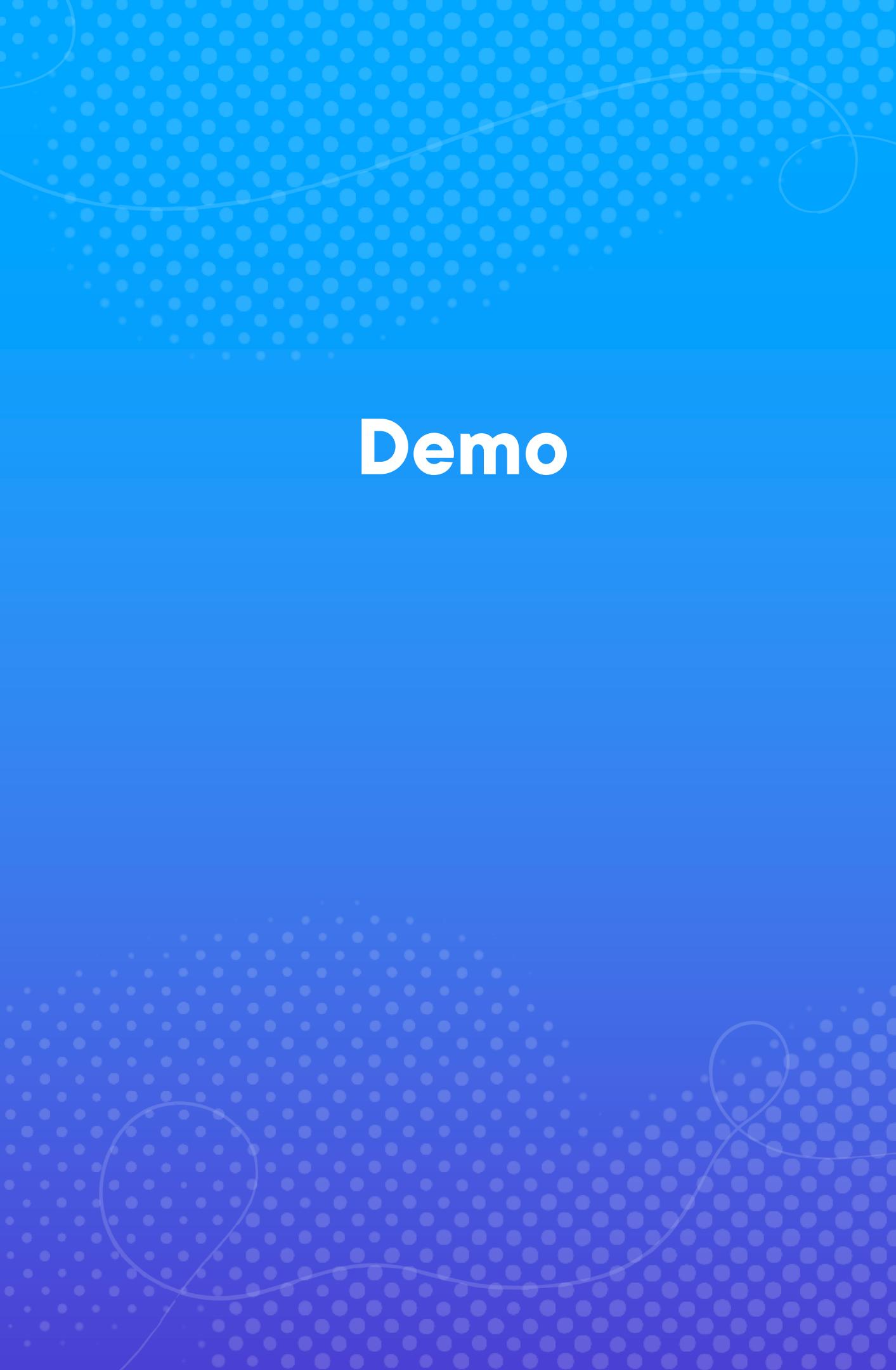
Convert entity attribute state to database column and vice-versa



Demo

Single table per class hierarchy





Demo

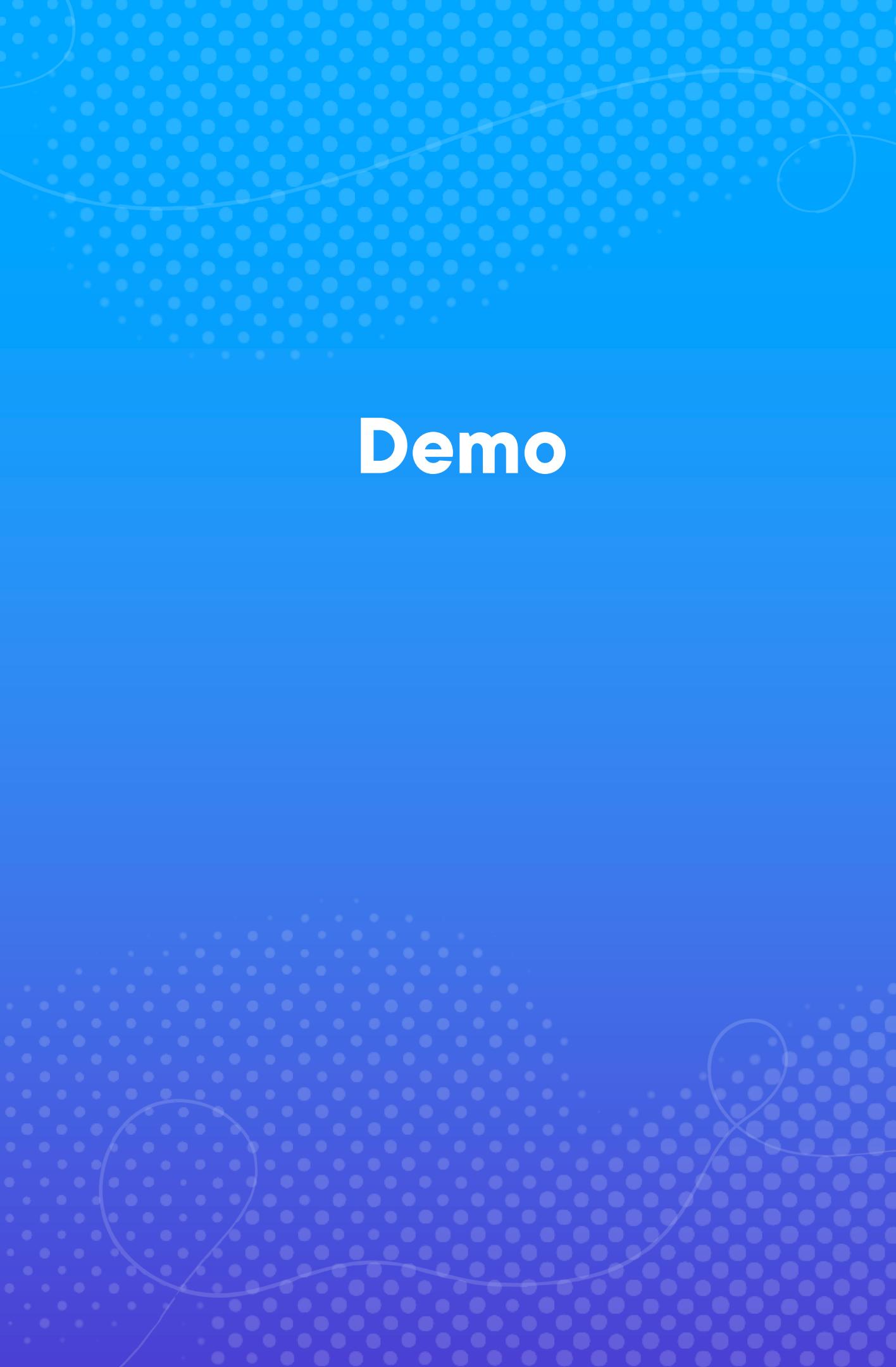
Joined subclass strategy



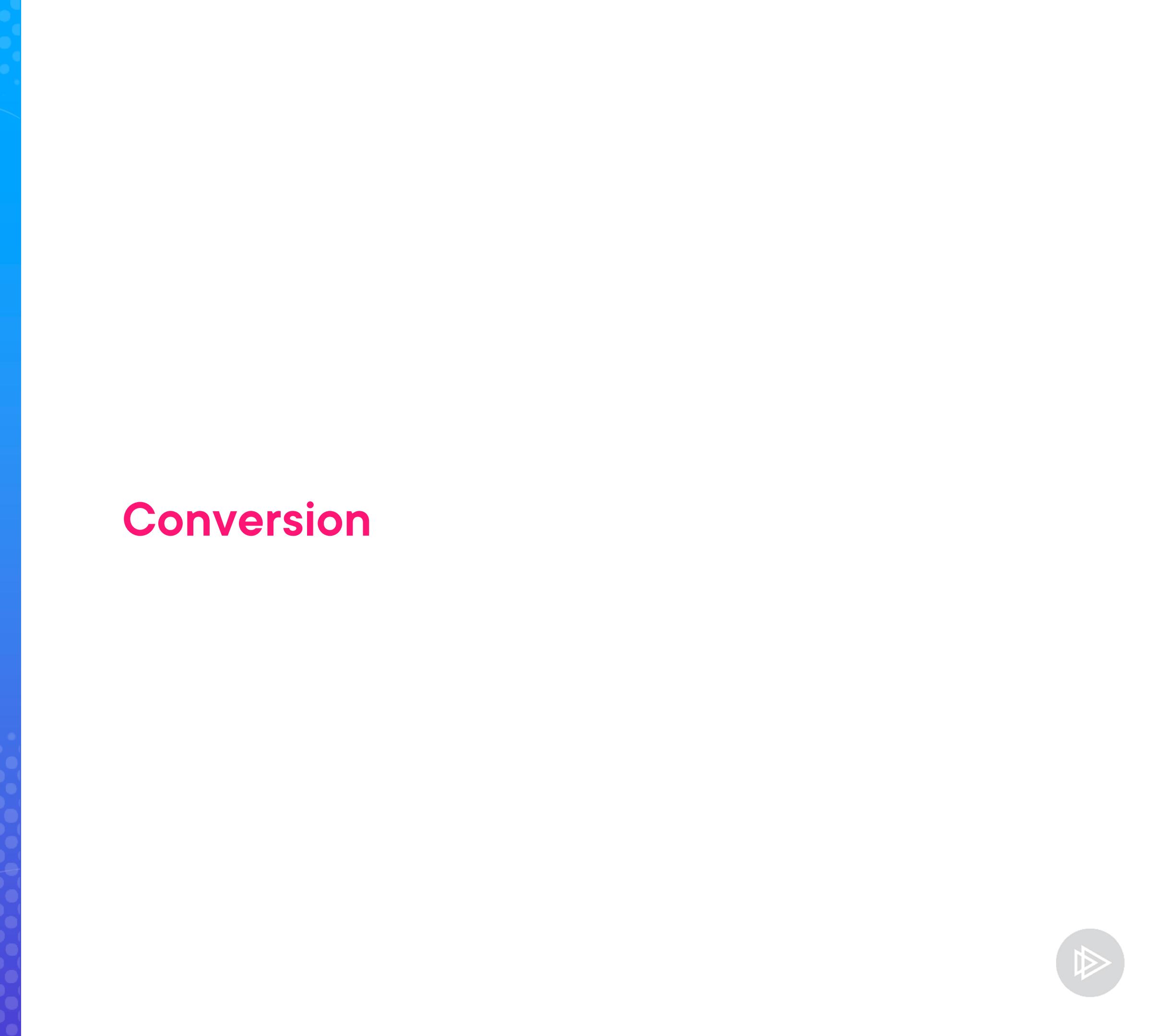
Demo

Table per concrete class strategy





Demo



Conversion



Summary

Build class hierarchies

- Inherited from entities
- Inherited from non-entities

Applied the mapping strategies

- Single table per class hierarchy
- Joined subclass strategy
- Table per concrete class strategy
- Particularities, pluses, minuses

Worked with converters

