Building Relationships Between Entities

Andrew Bancroft

@andrewcbancroft www.andrewcbancroft.com

Overview

Understand why building Relationships is good

Introduce the kinds of Relationships that can be created between Entities

Explain handling of deletions

Update data model with additional Employee Entity

Build Relationship between ShoutOut and Employee

Why Build Relationships?

Separation of concerns

Separation of Concerns

ShoutOut

- from
- message
- shoutCategory
- sentDate
- sentToEmployeeFirstName
- sentToEmployeeLastName
- sentToEmployeeDepartment
- ...

Separation of Concerns

ShoutOut

- from
- message
- shoutCategory
- sentDate

Why Build Relationships?

Separation of concerns

Expressive data model

Expressive Data Model

ShoutOut

- from
- message
- shoutCategory
- sentDate
- toEmployee

```
shoutOut.toEmployee.firstName
```

Employee

- firstName
- lastName
- department
- shoutOuts

```
employee.shoutOuts
```

Why Build Relationships?

Separation of concerns

Expressive data model

Efficiency and performance

Efficiency and Performance

ShoutOut

- from
- message
- shoutCategory
- sentDate
- toEmployee

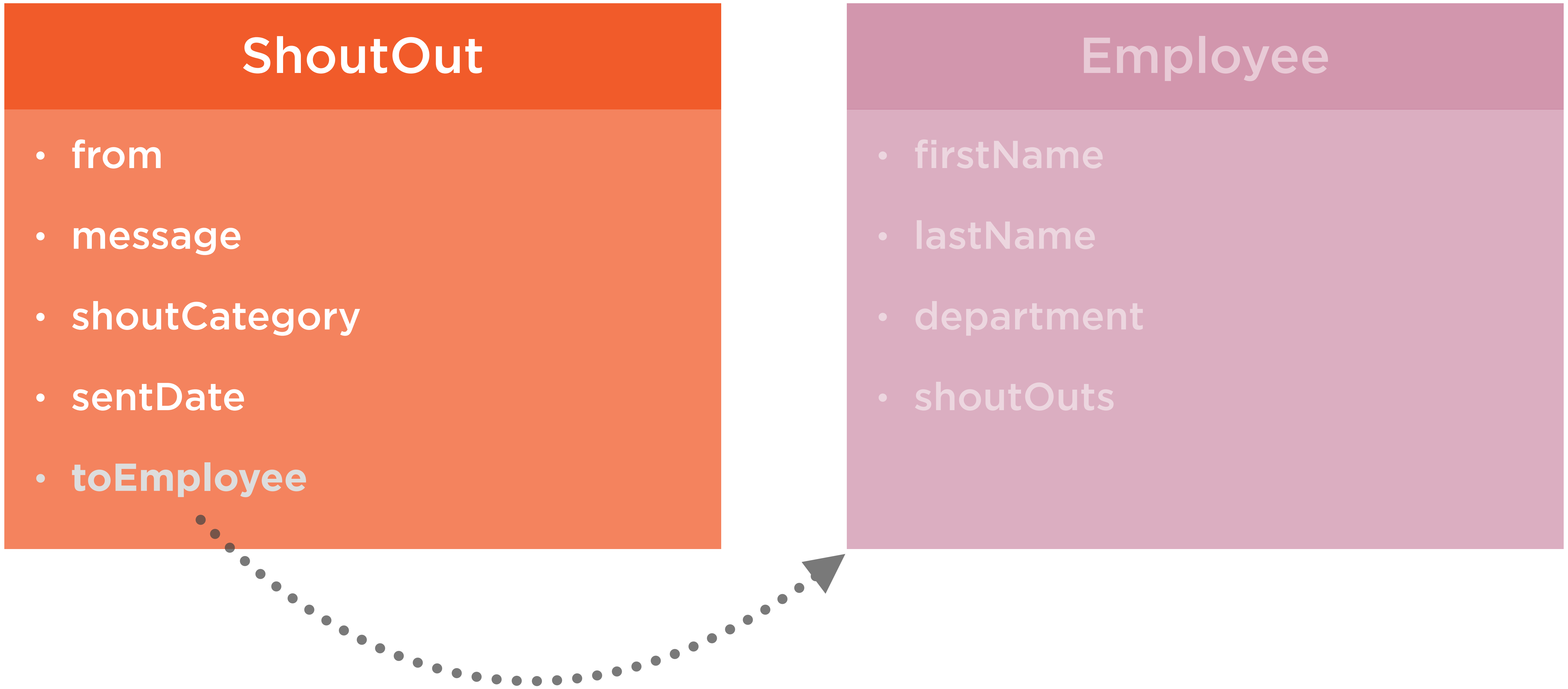
✓

Employee

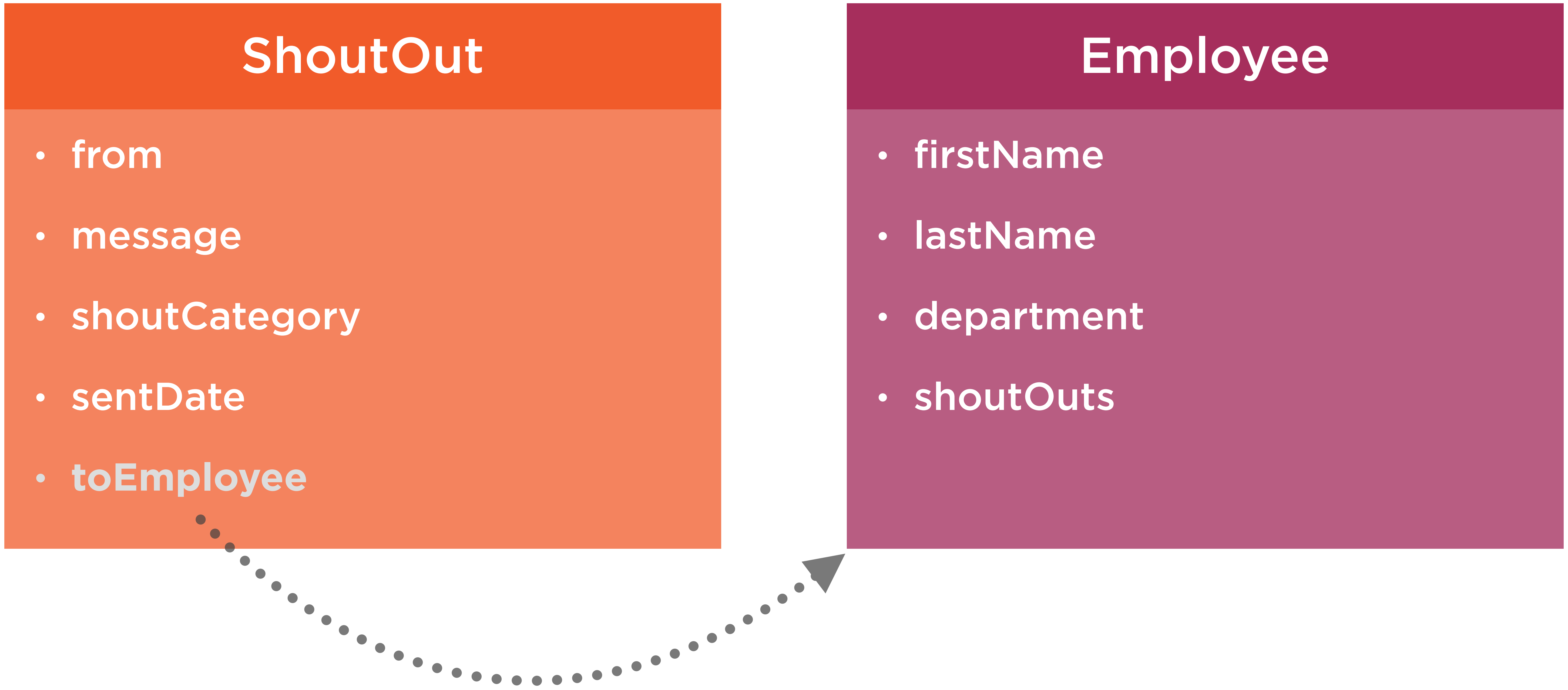
- firstName
- lastName
- department
- shoutOuts

✗

Efficiency and Performance

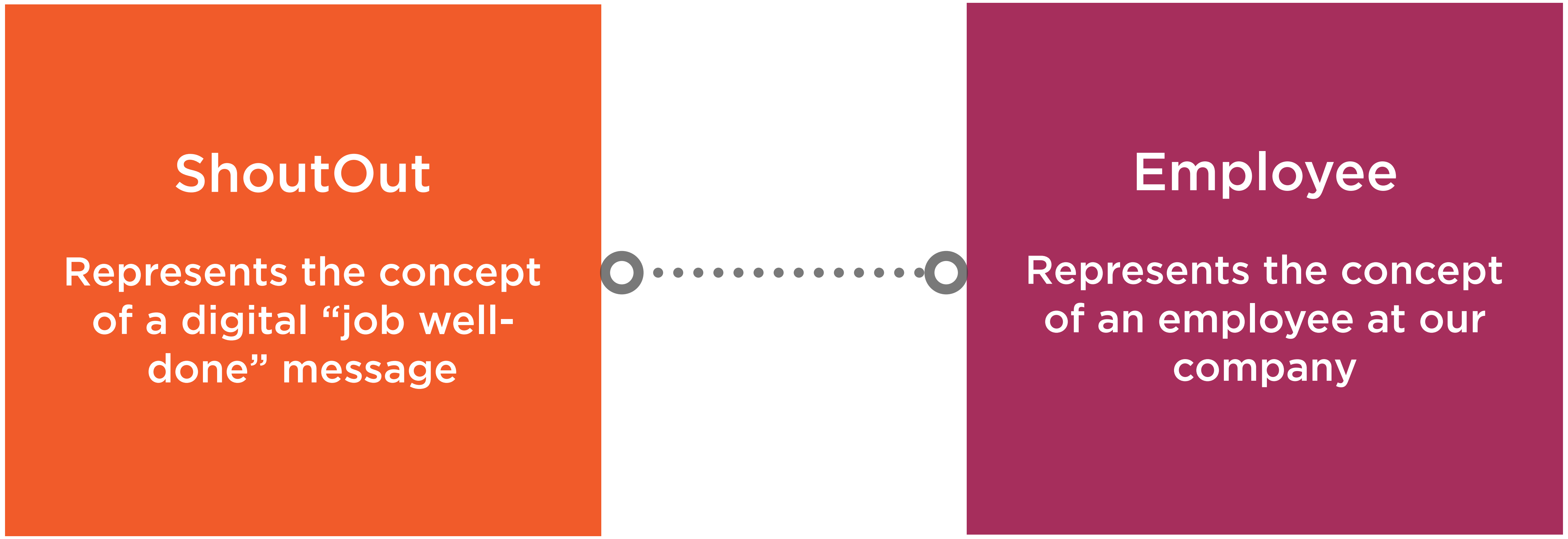


Efficiency and Performance



Describing Relationships

Relationships Between Entities



ShoutOut

Represents the concept of a digital "job well-done" message

Employee

Represents the concept of an employee at our company

Relationship Questions

What is the source Entity?

What is the destination Entity?

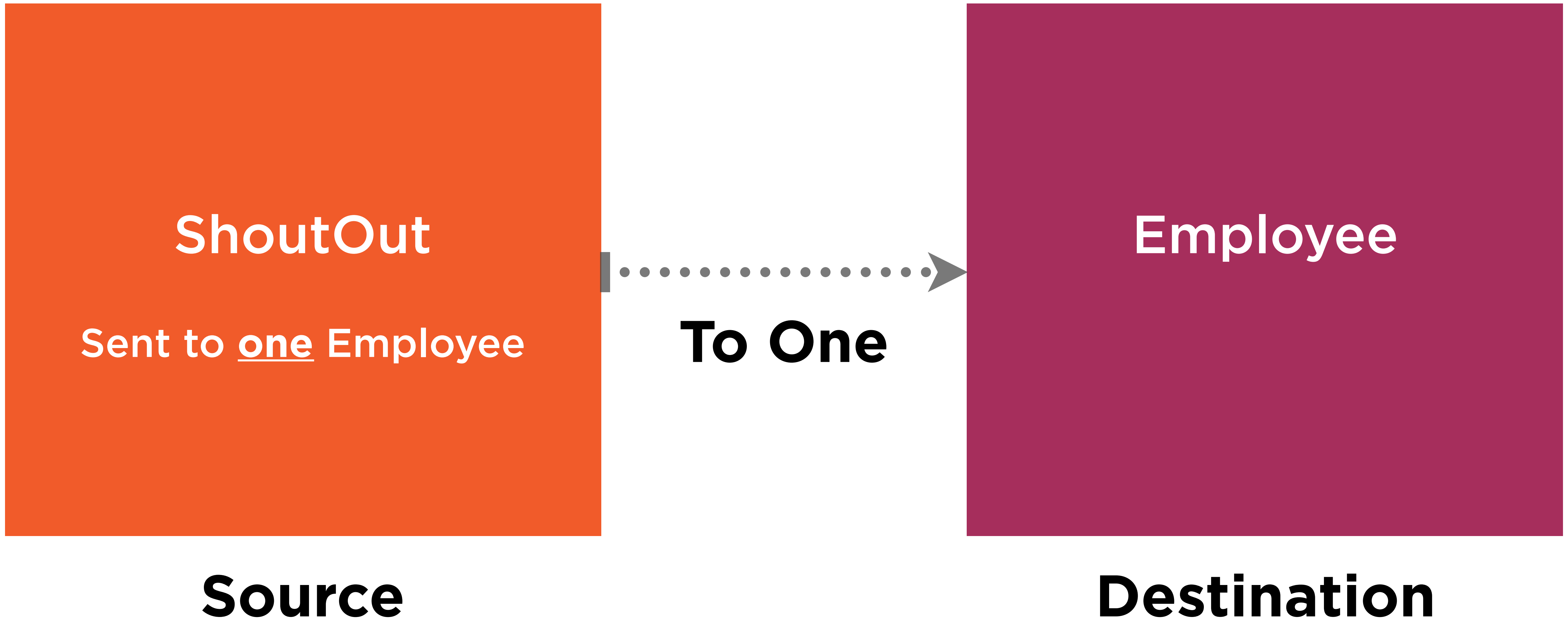
What is the cardinality? To-one or to-many?

- If to-many, is there min/max number of objects involved in the relationship?

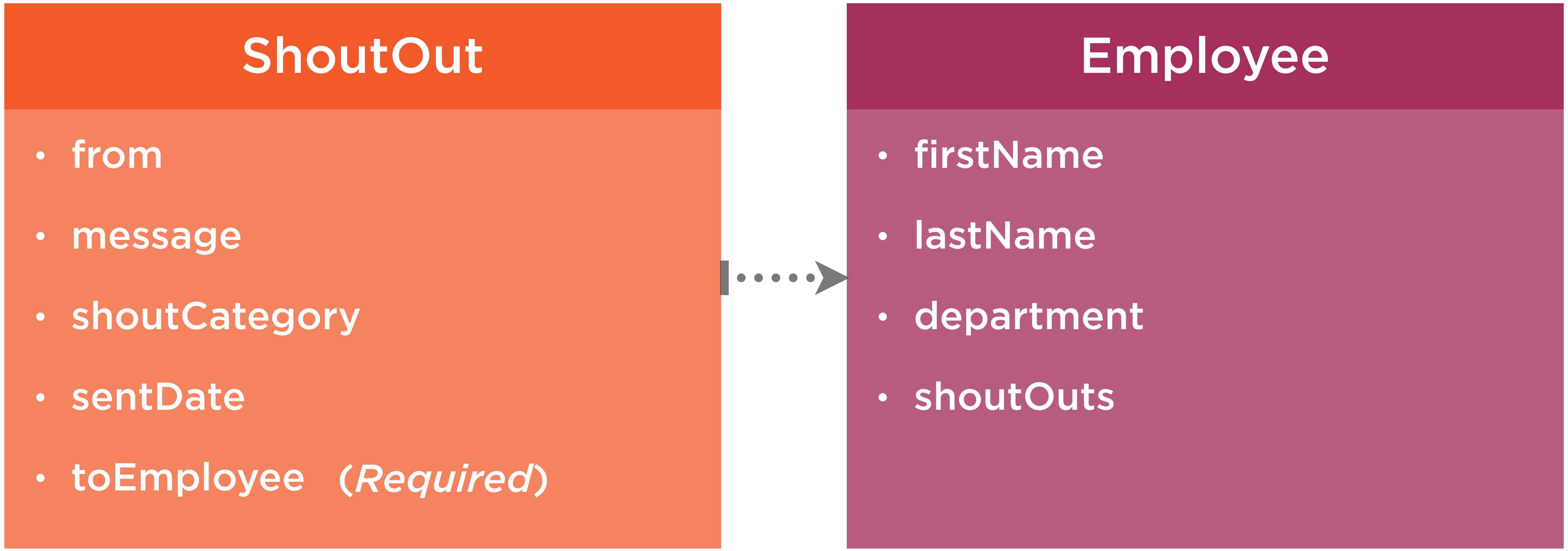
Is the relationship optional?

How to handle changes and deletions?

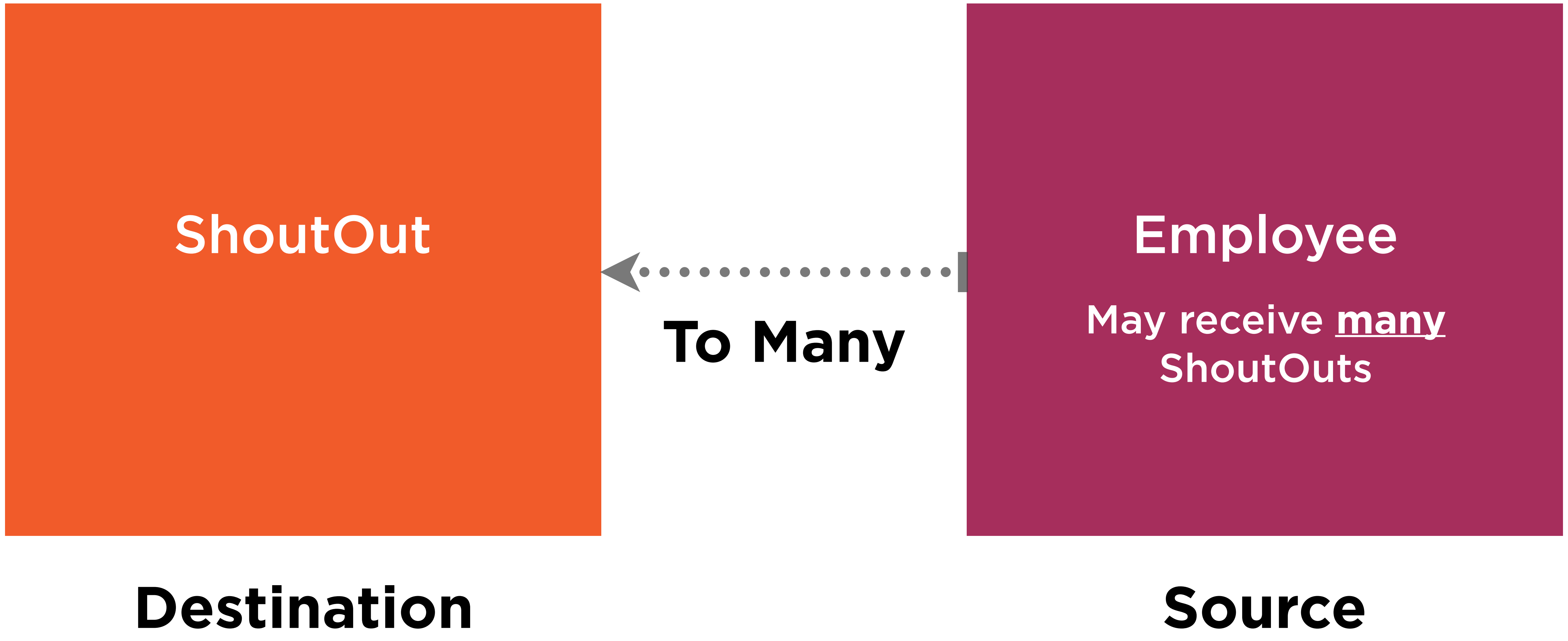
Source, Destination, and Cardinality



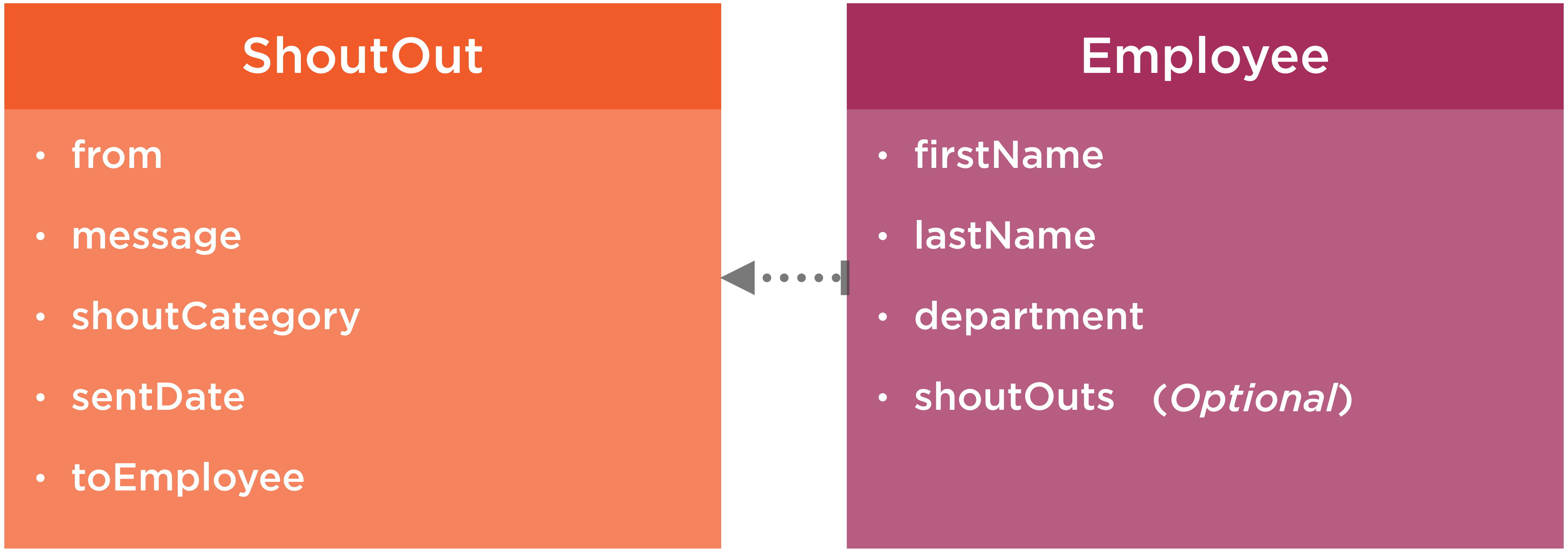
Required or Optional



Source, Destination, and Cardinality

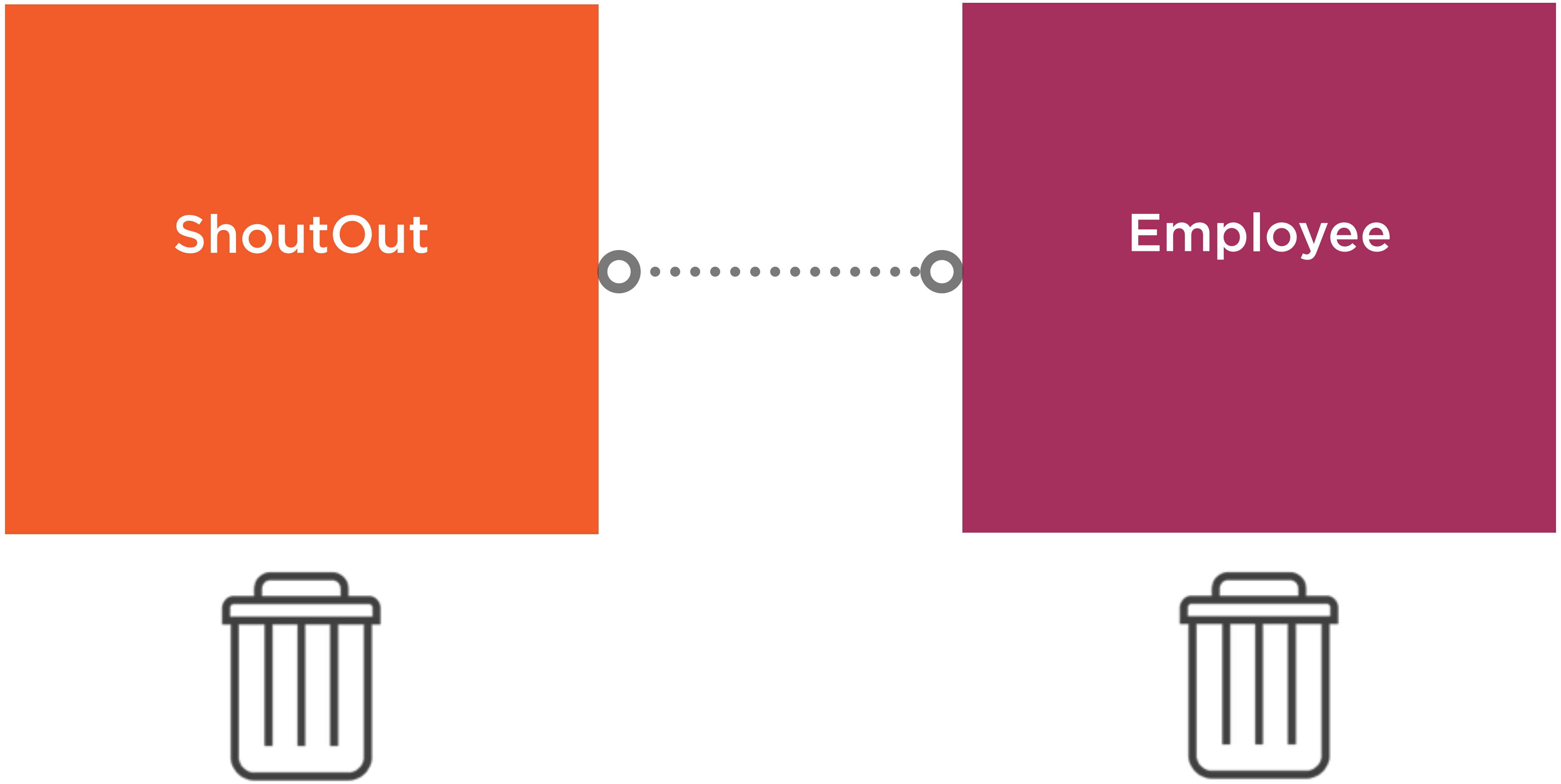


Required or Optional



Handling Deletions Between Related Entities

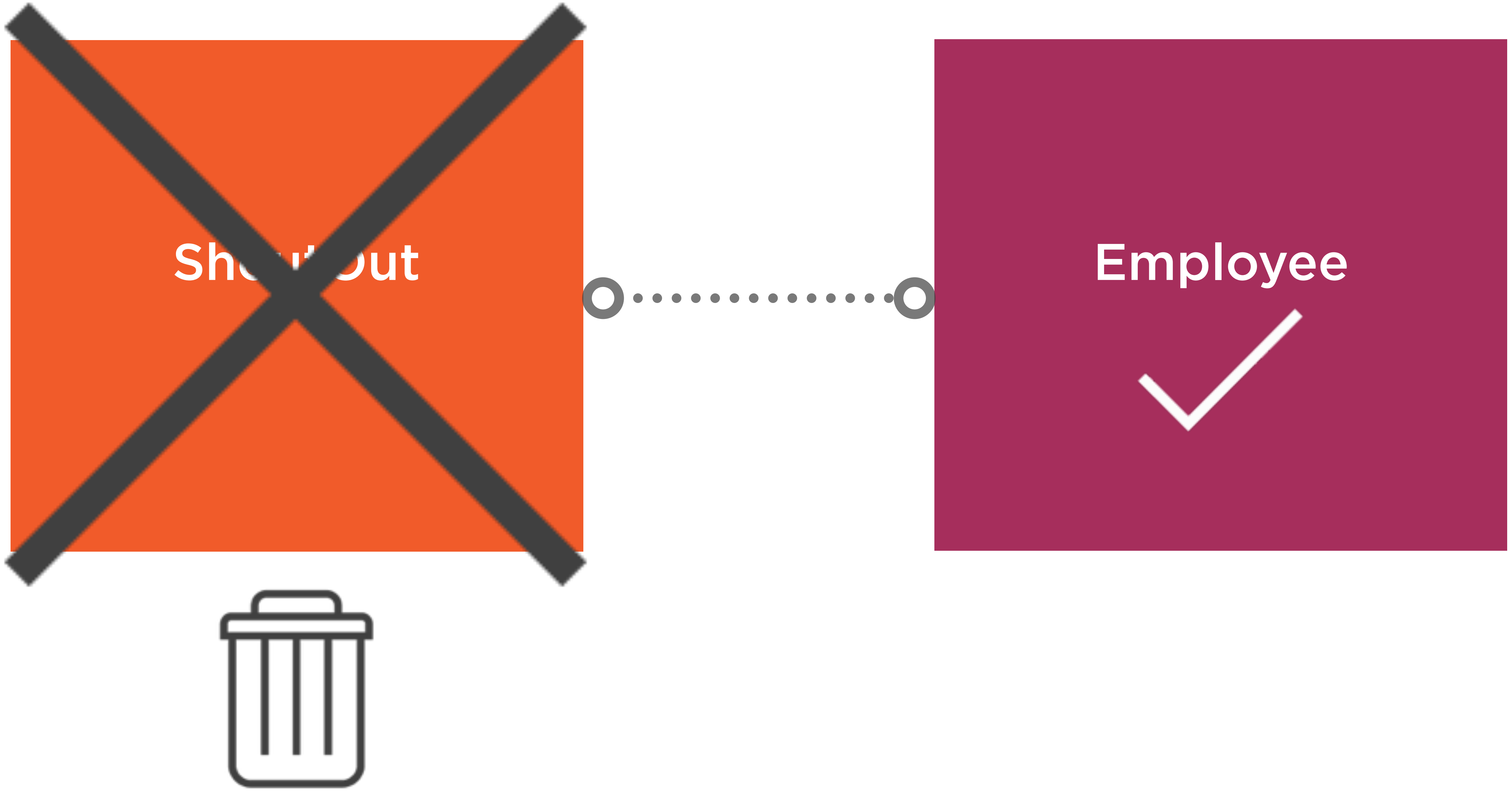
Handling Deletions Between Related Entities



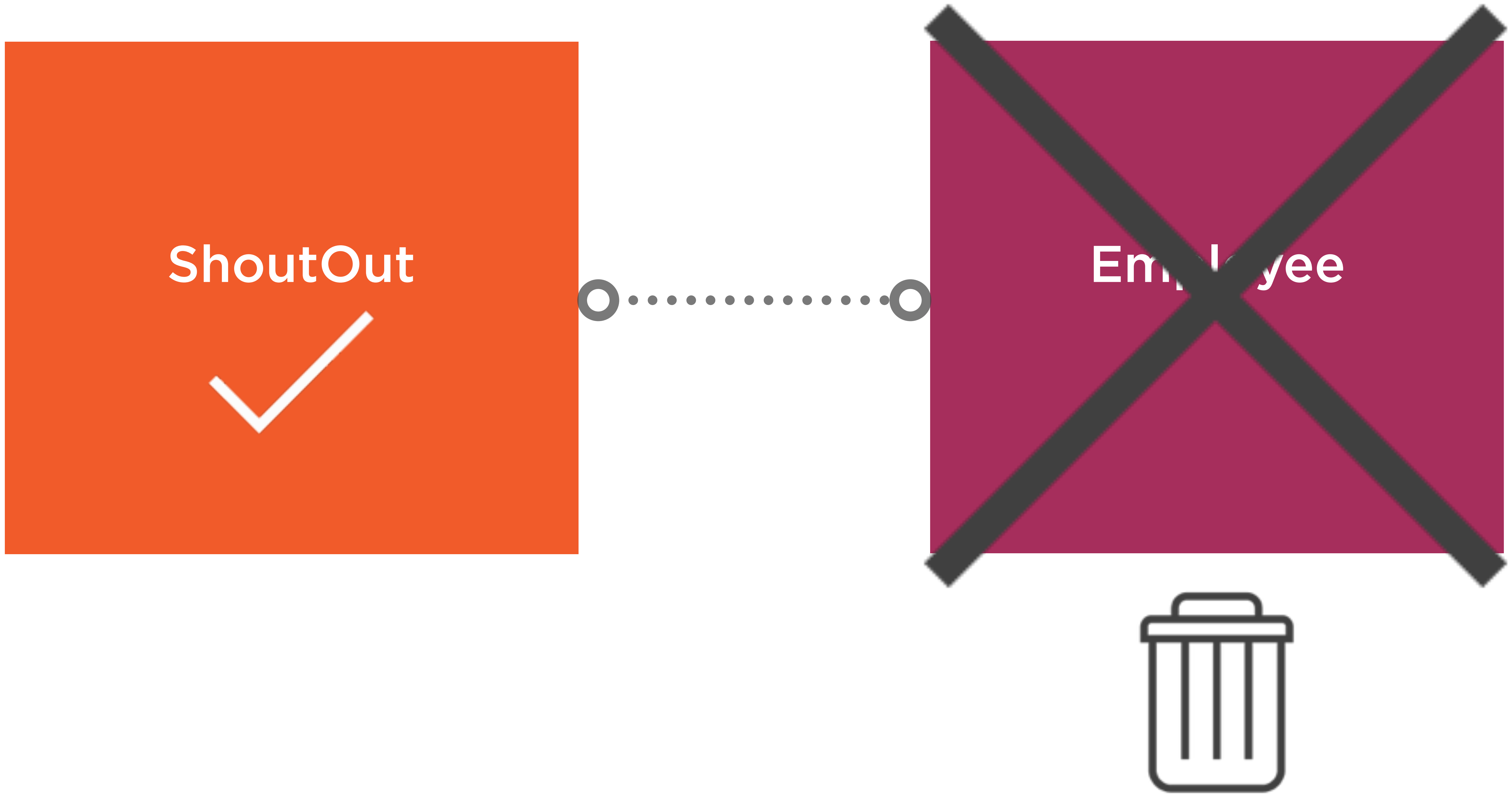
Delete Rules

No Action	Nullify
Cascade	Deny

No Action Upon Delete



No Action Upon Delete

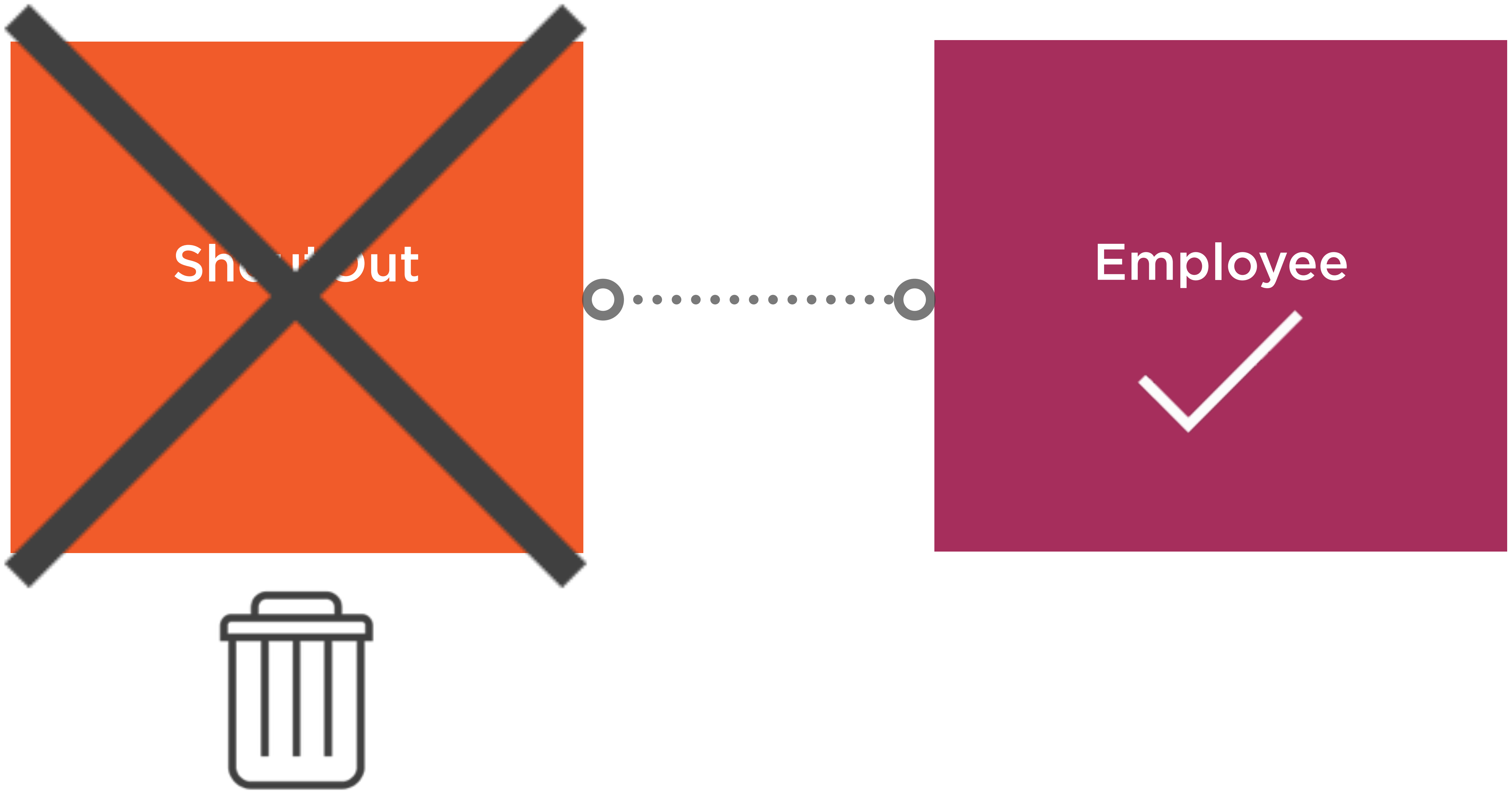


No Action Delete Rule

Could “orphan” objects in the persistent store

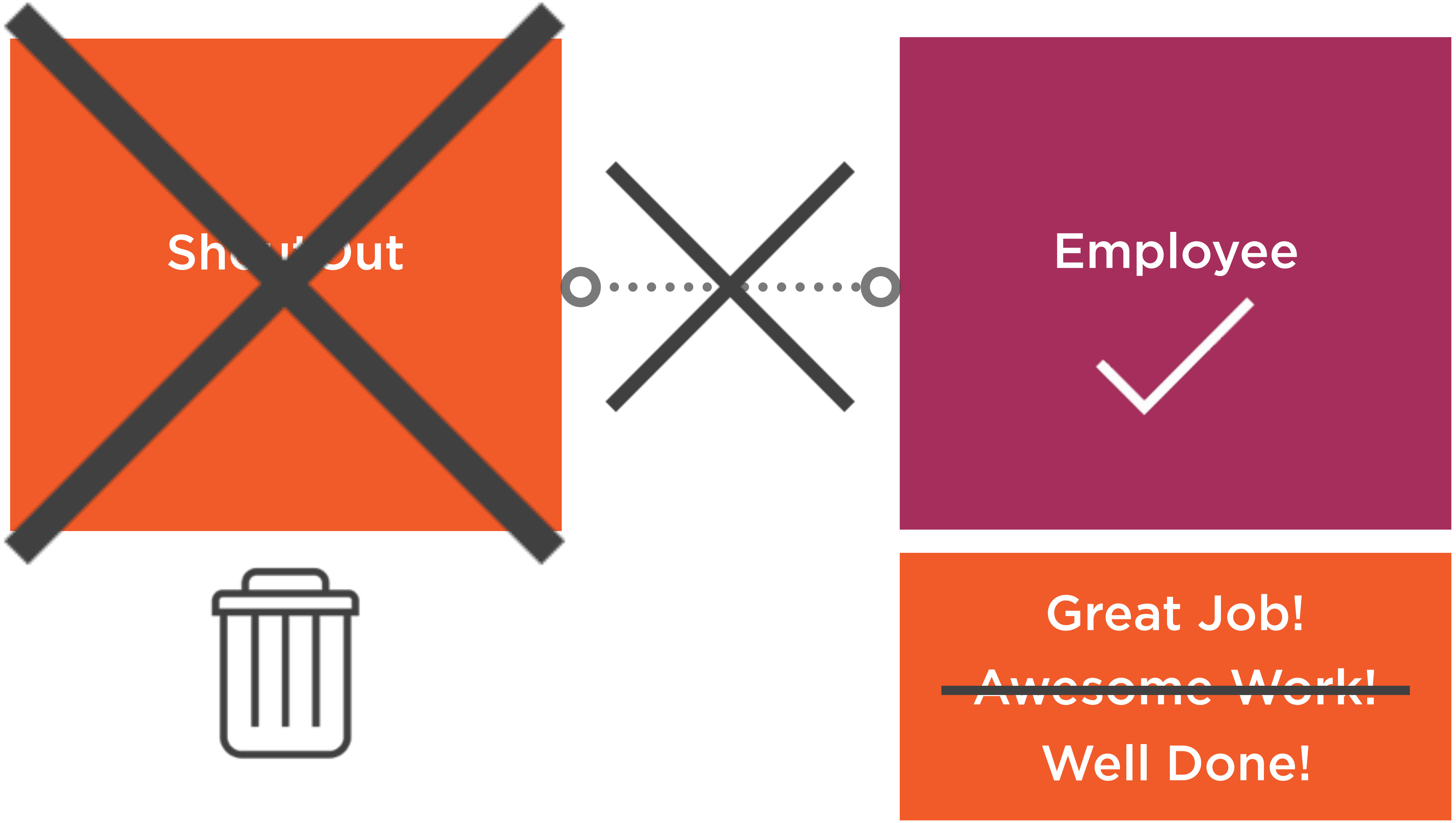
Requires manual handling of removing related objects (if necessary)

Nullifying Relationships Between Entities

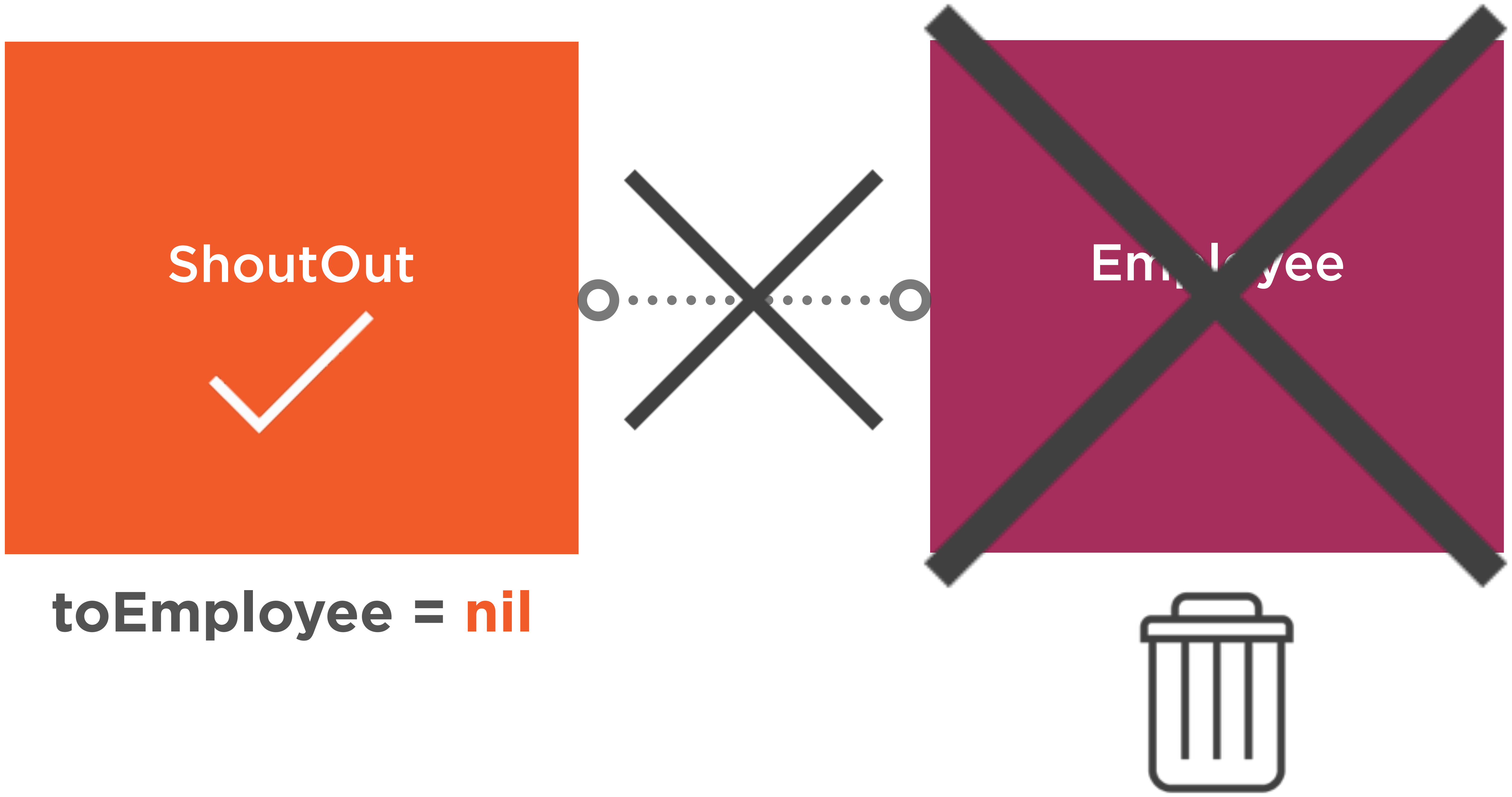


nil + Swift *Optional*

Nullifying Relationships Between Entities



Nullifying Relationships Between Entities



toEmployee = nil

Cascade Delete Rule

Use when keeping related Entity instances in persistent store doesn’t make sense with source Entity removed

Cascade Delete Rule

Use when keeping related Entity instances in persistent store doesn't make sense with source Entity removed

Cascade Delete Rule

Use when keeping related Entity instances in persistent store doesn’t make sense with source Entity removed

Cascade Delete Rule

Use when keeping related Entity instances in persistent store doesn’t make sense with source Entity removed

Cascade Delete Rule

Use when keeping related Entity instances in persistent store doesn’t make sense with source Entity removed

Cascade Delete Rule

Use when keeping related Entity instances in persistent store doesn’t make sense with source Entity removed

“Cascade” because it removes destination Entity instances from persistent store when source deleted

Cascade Delete Rule

Use when keeping related Entity instances in persistent store doesn't make sense with source Entity removed

"Cascade" because it removes destination Entity instances from persistent store when source deleted

Cascade Delete Rule

Use when keeping related Entity instances in persistent store doesn’t make sense with source Entity removed

“Cascade” because it removes destination Entity instances from persistent store when source deleted

Cascade Delete Rule

Use when keeping related Entity instances in persistent store doesn’t make sense with source Entity removed

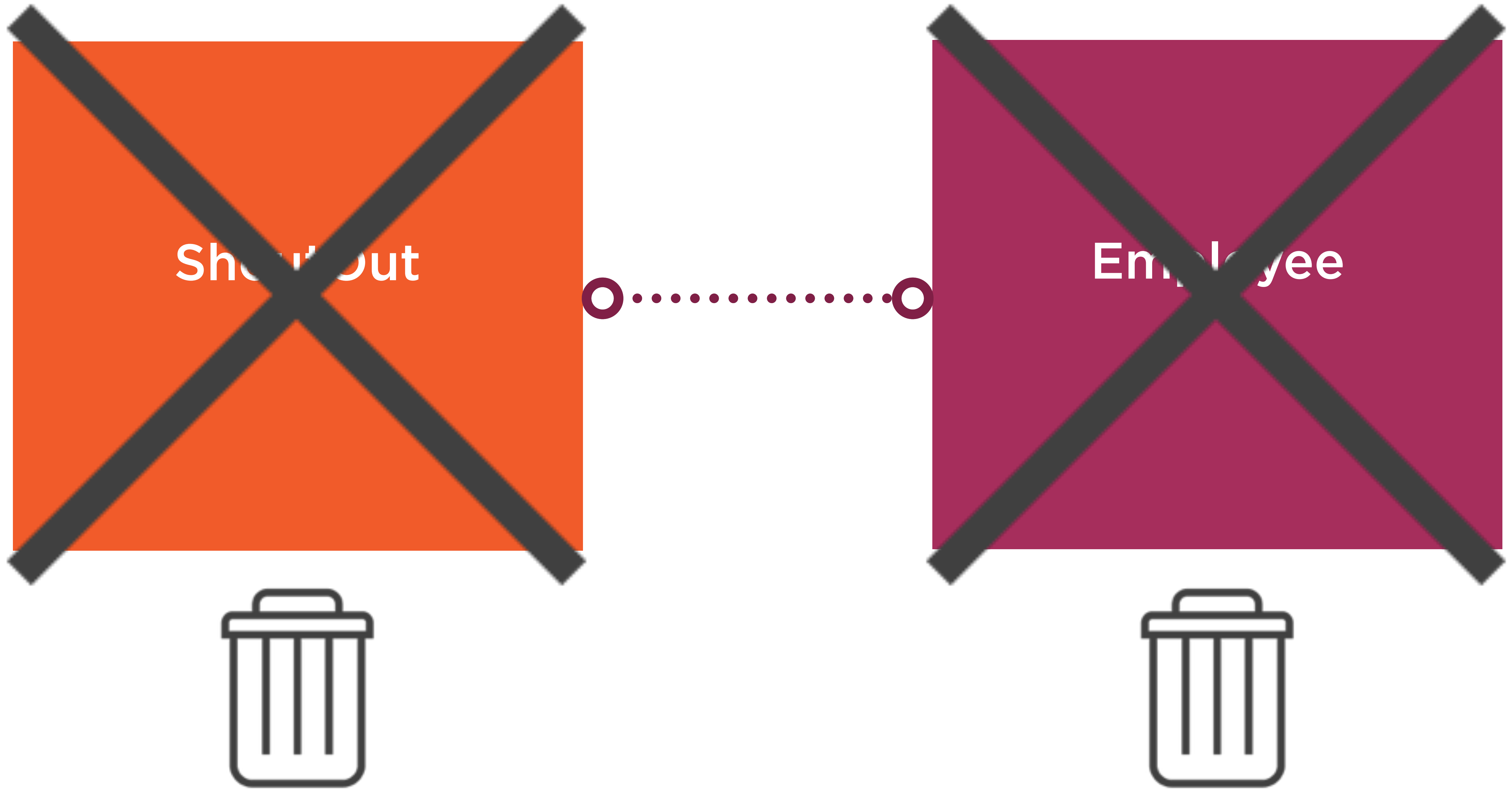
“Cascade” because it removes destination Entity instances from persistent store when source deleted

Cascade Delete Rule

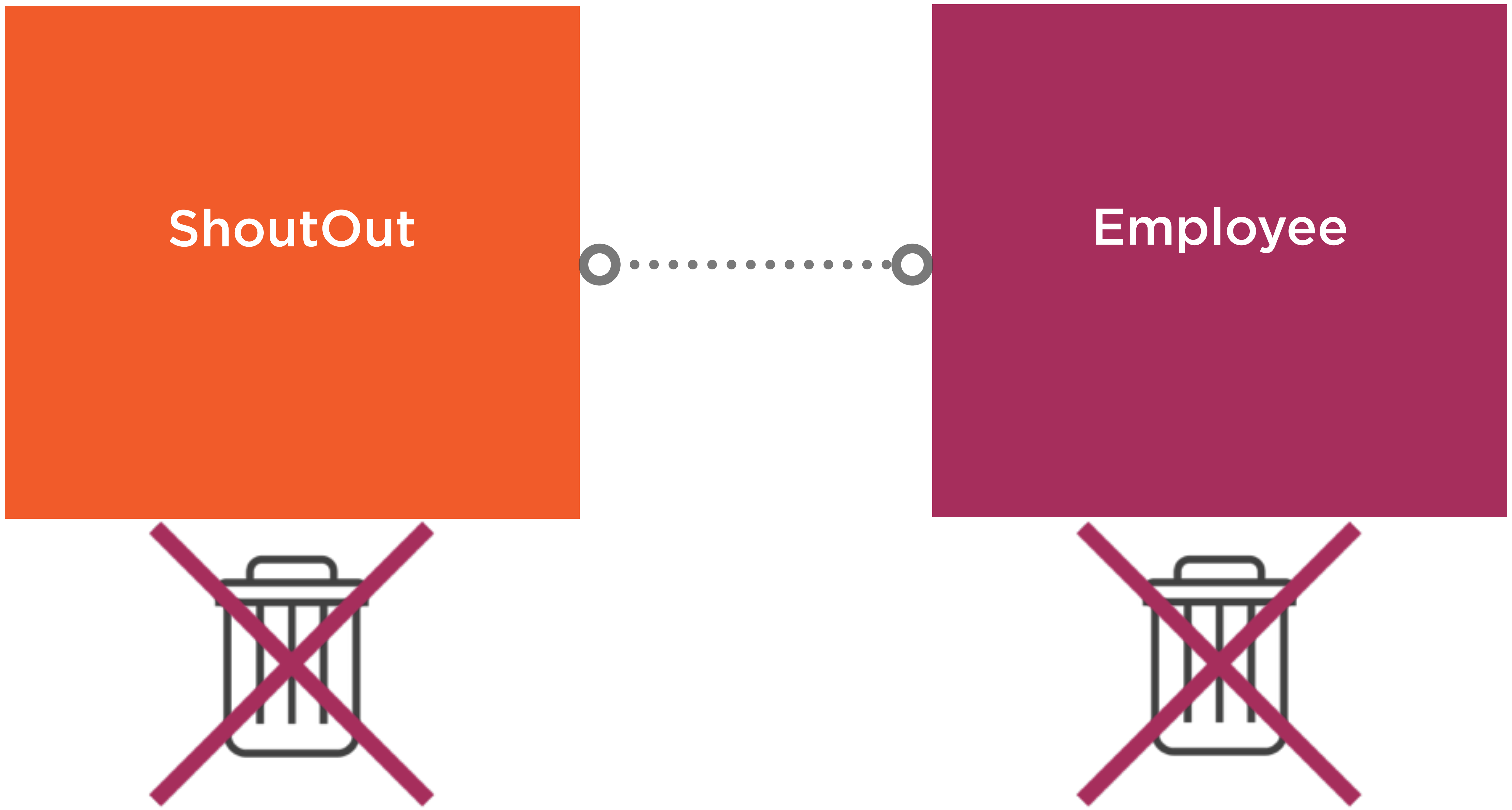
Use when keeping related Entity instances in persistent store doesn’t make sense with source Entity removed

“Cascade” because it removes destination Entity instances from persistent store when source deleted

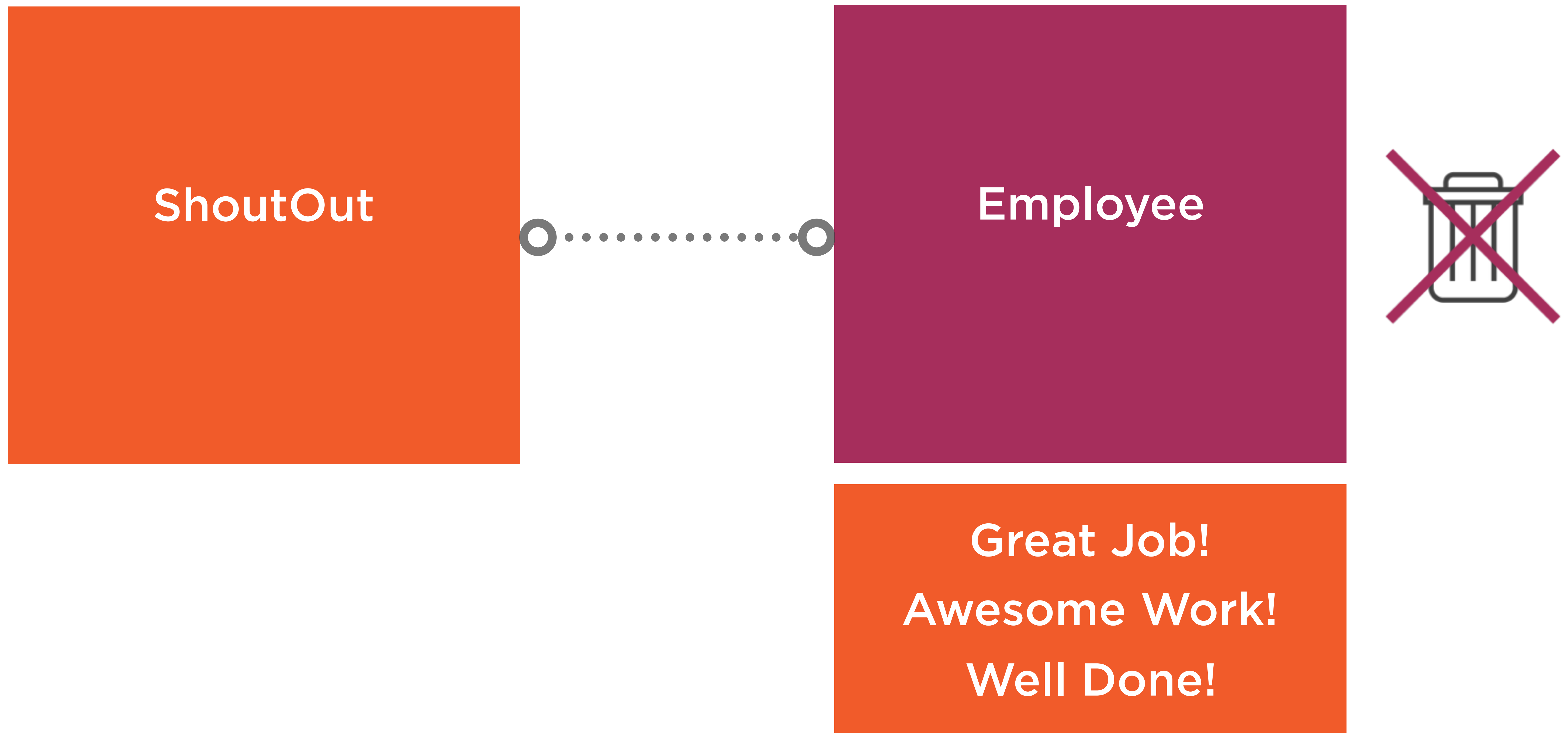
Cascading Deletions Between Entities



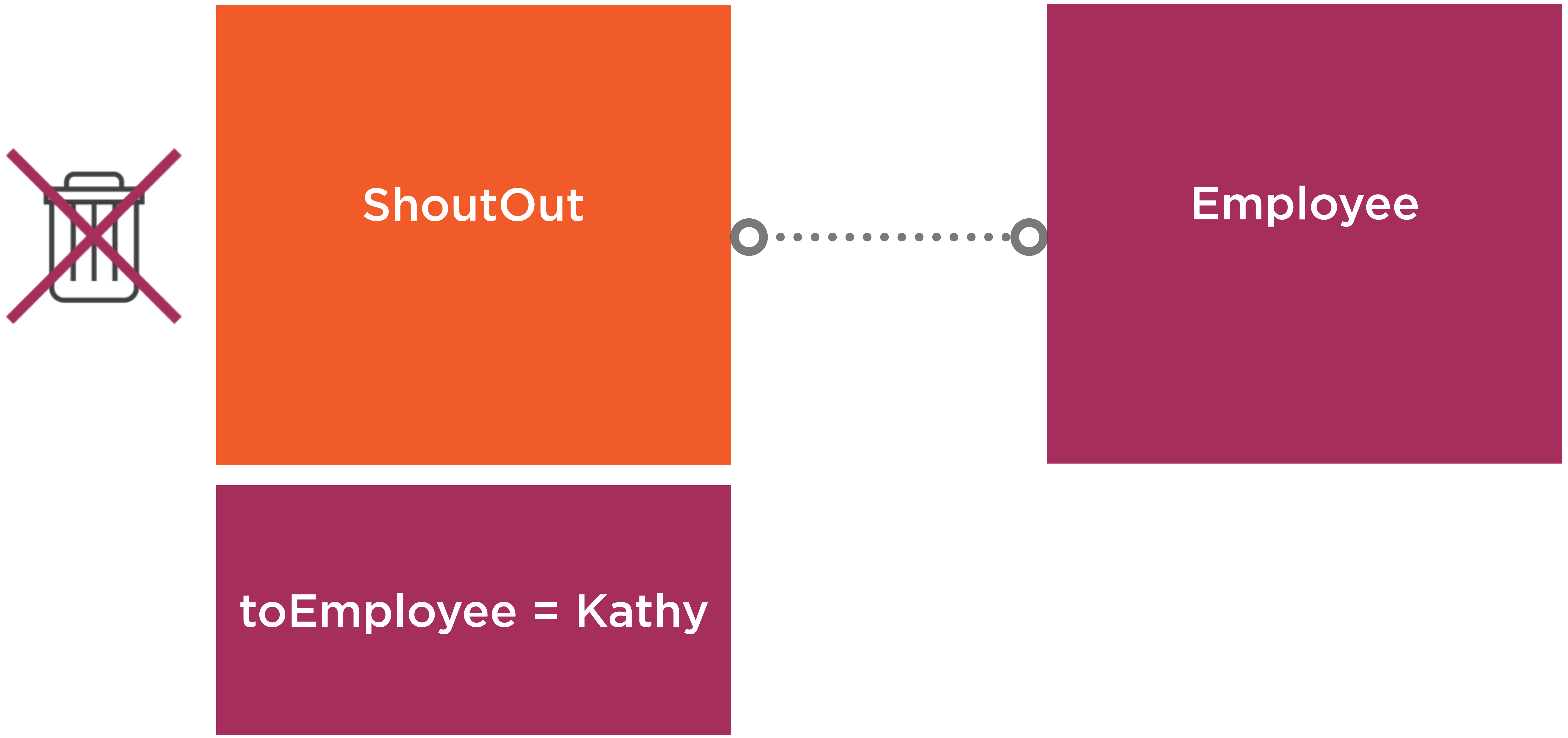
Denying Deletions Between Entities



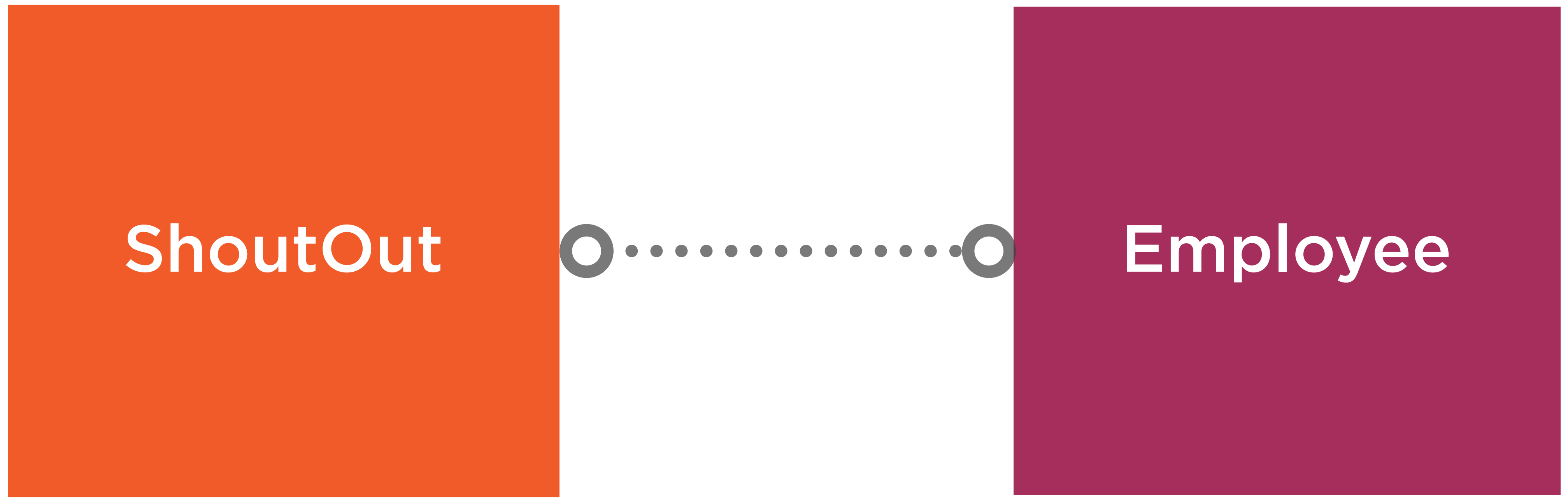
Denying Deletions Between Entities



Denying Deletions Between Entities



Demo



Demo

Create Employee Entity

Add Attributes to Employee

Build relationship between ShoutOut and Employee

Implement NSManagedObject subclass for Employee Entity

Summary

Introduced the kinds of Relationships that can be created between Entities

- To-One
- To-Many
- Optional
- Inverse

Considered handling of changes and deletions

Updated data model with additional Employee Entity

Built Relationship between ShoutOut and Employee