Best Practices in ASP.NET Core: Entities, Validation, and View Models

THE BIG PICTURE

Shawn Wildermuth
MICROSOFT MVP, SPEAKER AND INSTRUCTOR

@shawnwildermuth http://wilderminds.com

Overview

Introduce Best Practices

- Separate entities and persistence
- Make sense of the role of View Models
- Introduce Validation

Overview

Don't Take This Course Unless:

- You already know C#
- You understand ASP.NET Core
- You've stored data in a database
- You want to learn best practices

What You'll Learn

- Understand Entity Design
- Make sense of View Models
- Effective use of Validation
- How View Models are used in APIs
- Client and server model differences

What You'll Use

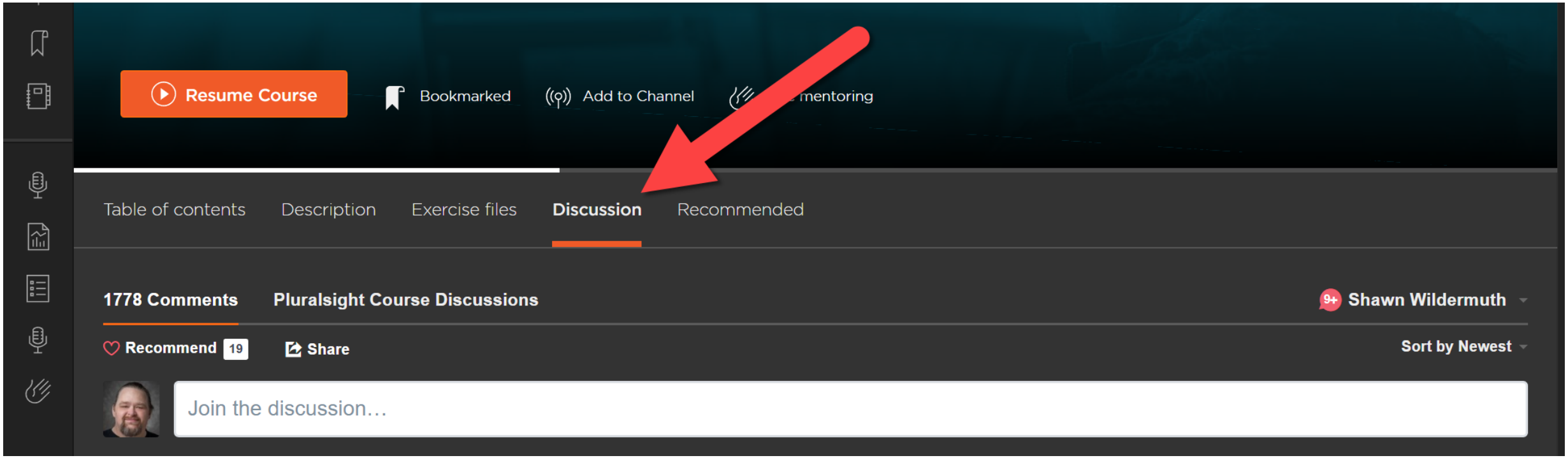
Visual Studio

ASP.NET Core

Entity Framework Core

- Though, concepts are agnostic

Questions During the Course?



Real World Entities

define: Entity

Something that exists in itself, actually or potentially, concretely or abstractly, physically or not that can be uniquely identified in some way.

Entities are Nouns

Person

Invoice

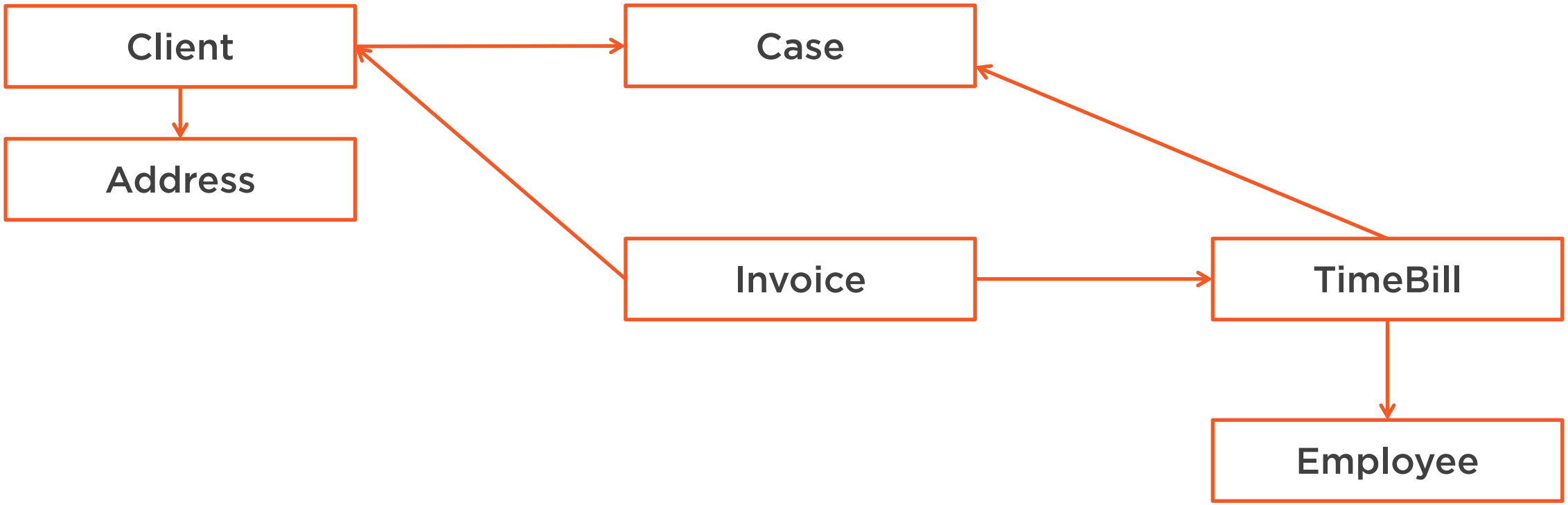
Customer

Address

Product

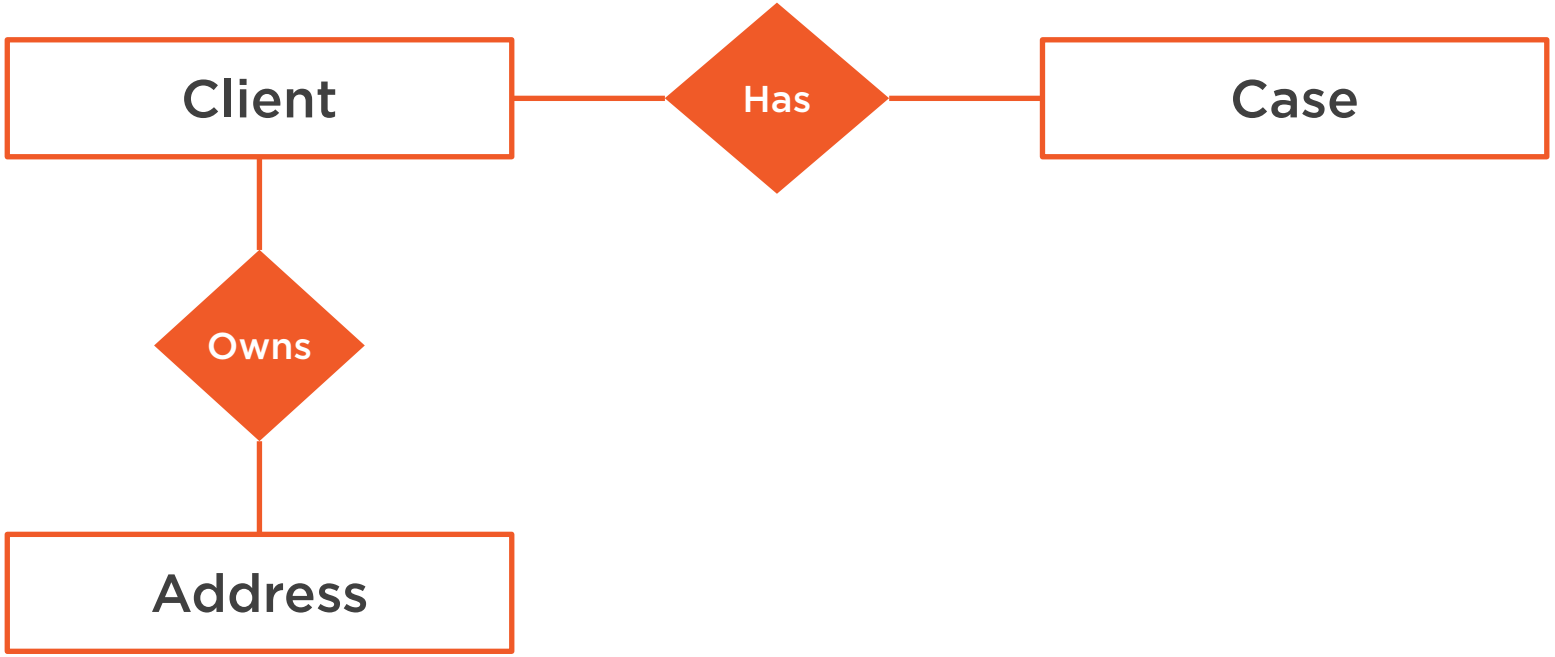
Log

Our Example

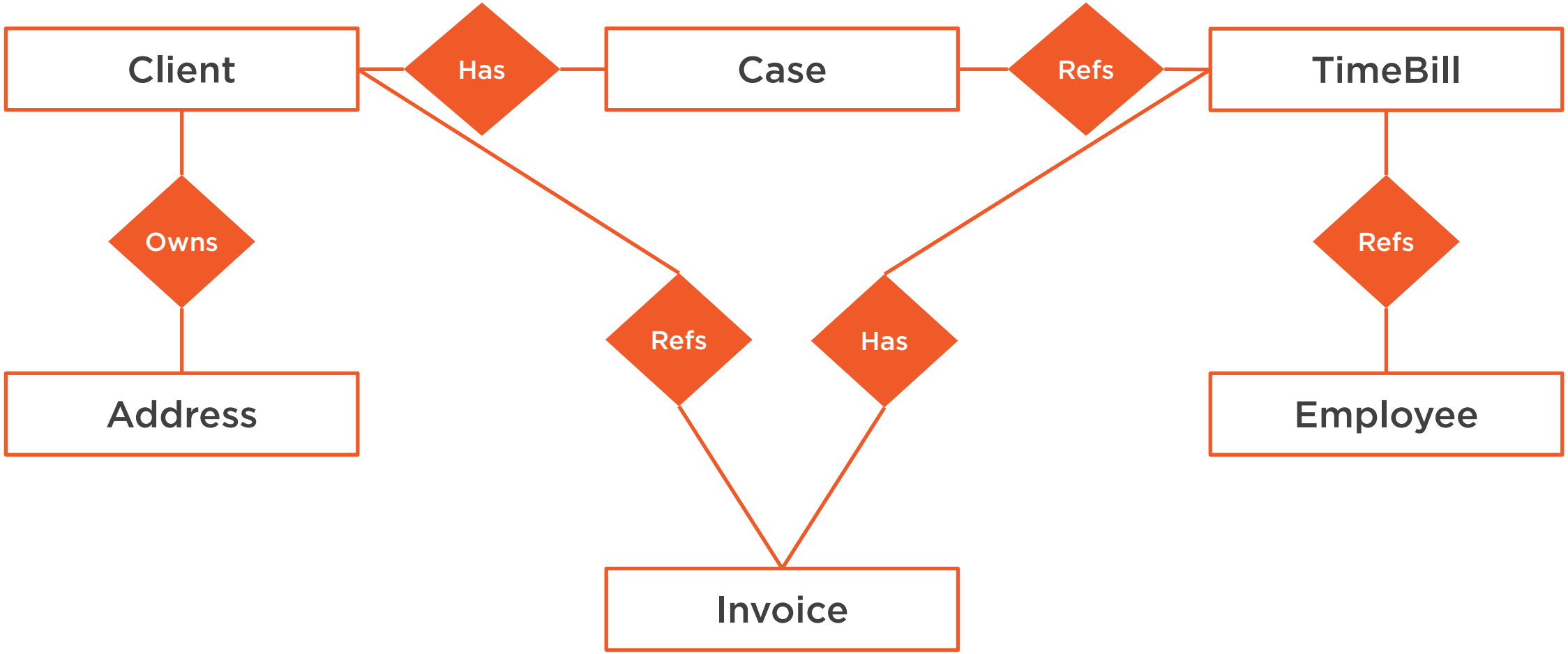


Entity Relational Model

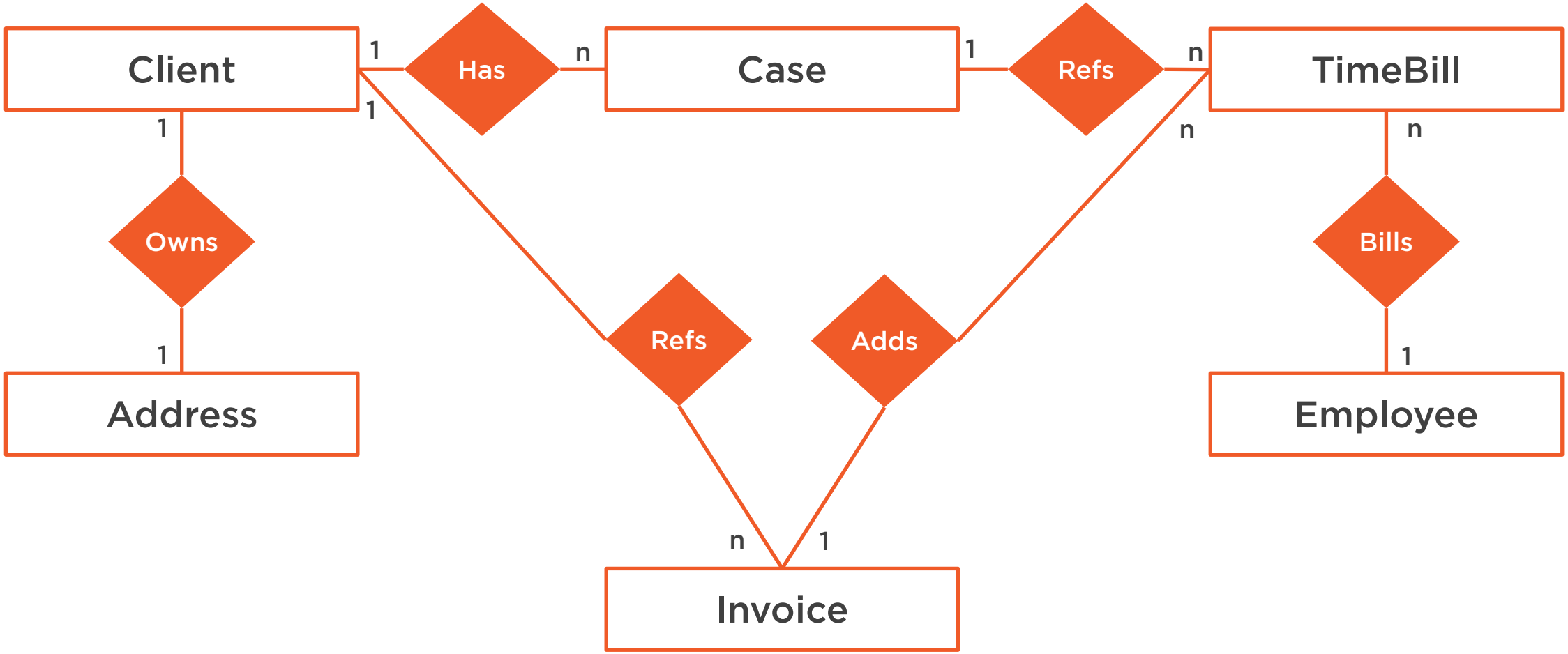
Entity Relational Model



Entity Relational Model



Entity Relational Model



Entity Composition

Entities need to be uniquely identifiable;

that’s why Entity Keys exist.

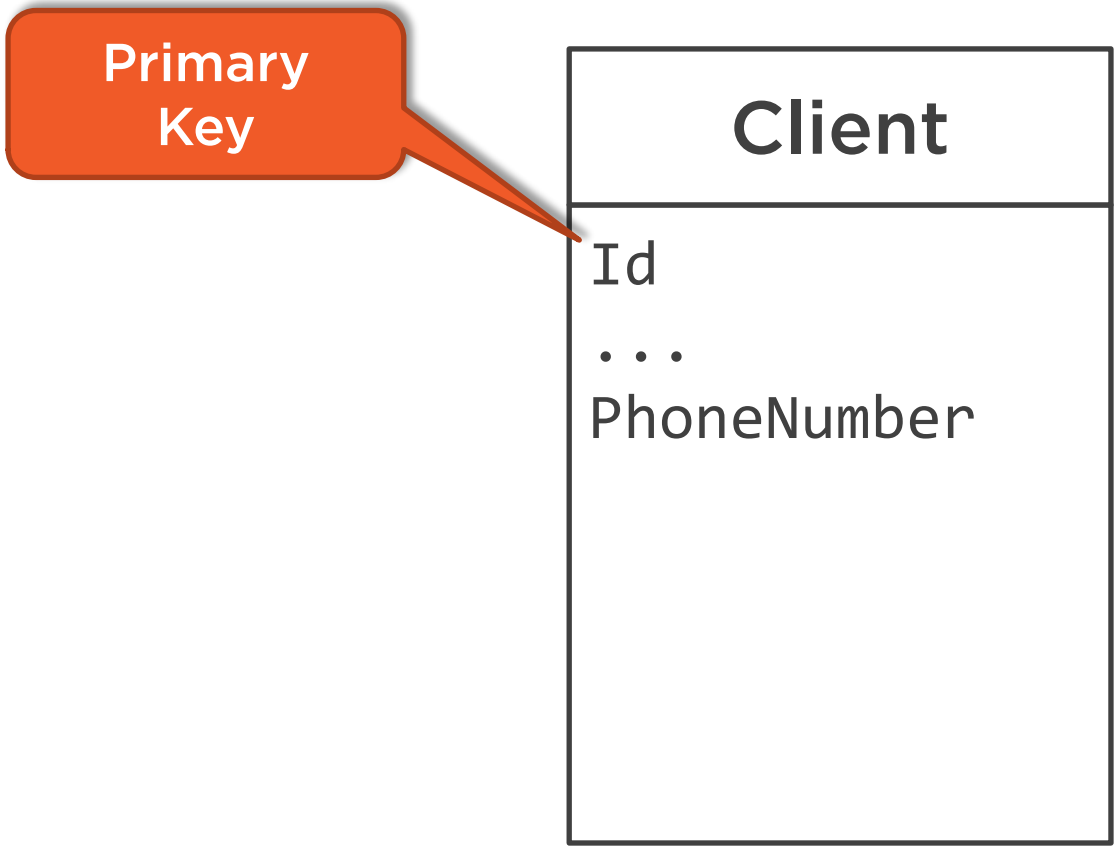
Entity Keys

Primary Key: Unique Identifying Key

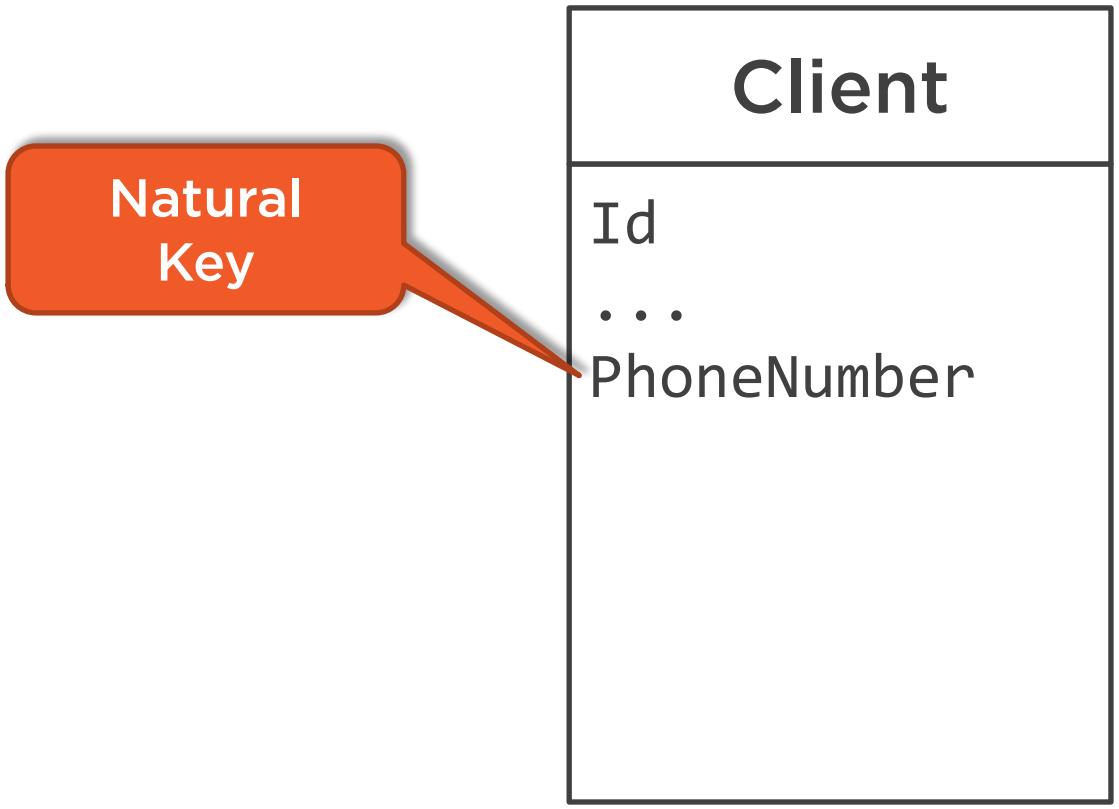
Foreign Key: Attribute that connects to another entity's primary key

Natural Key: Existing, real-world identifier (e.g. Social Security #)

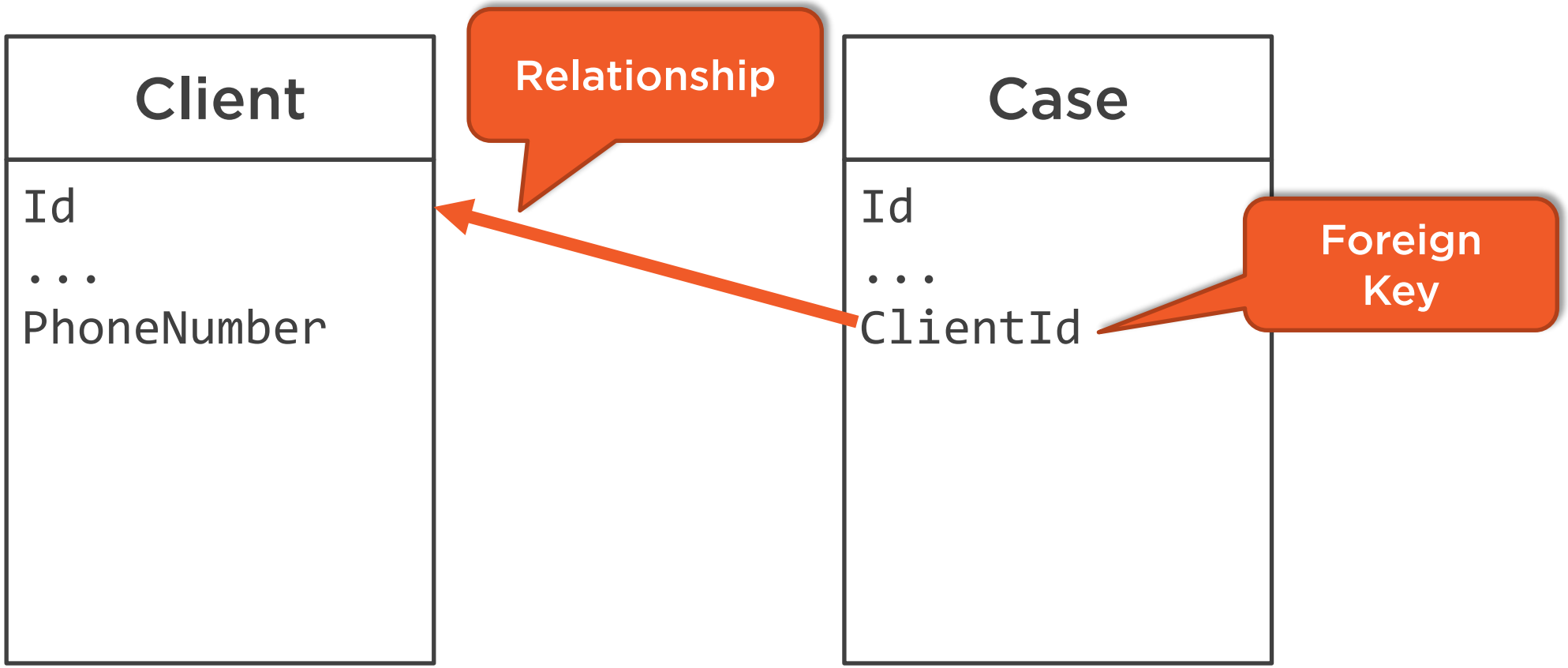
Entity Keys



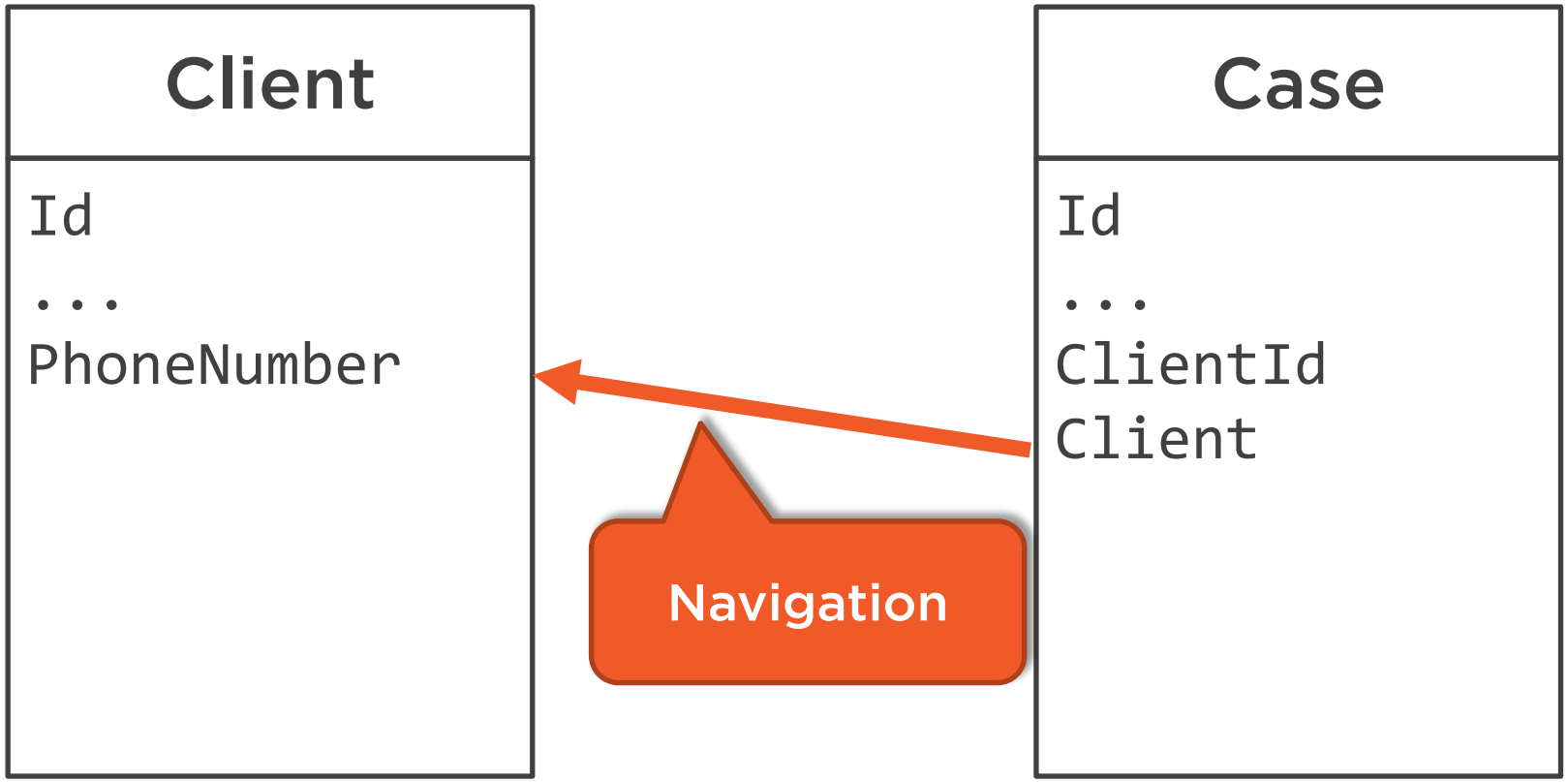
Entity Keys



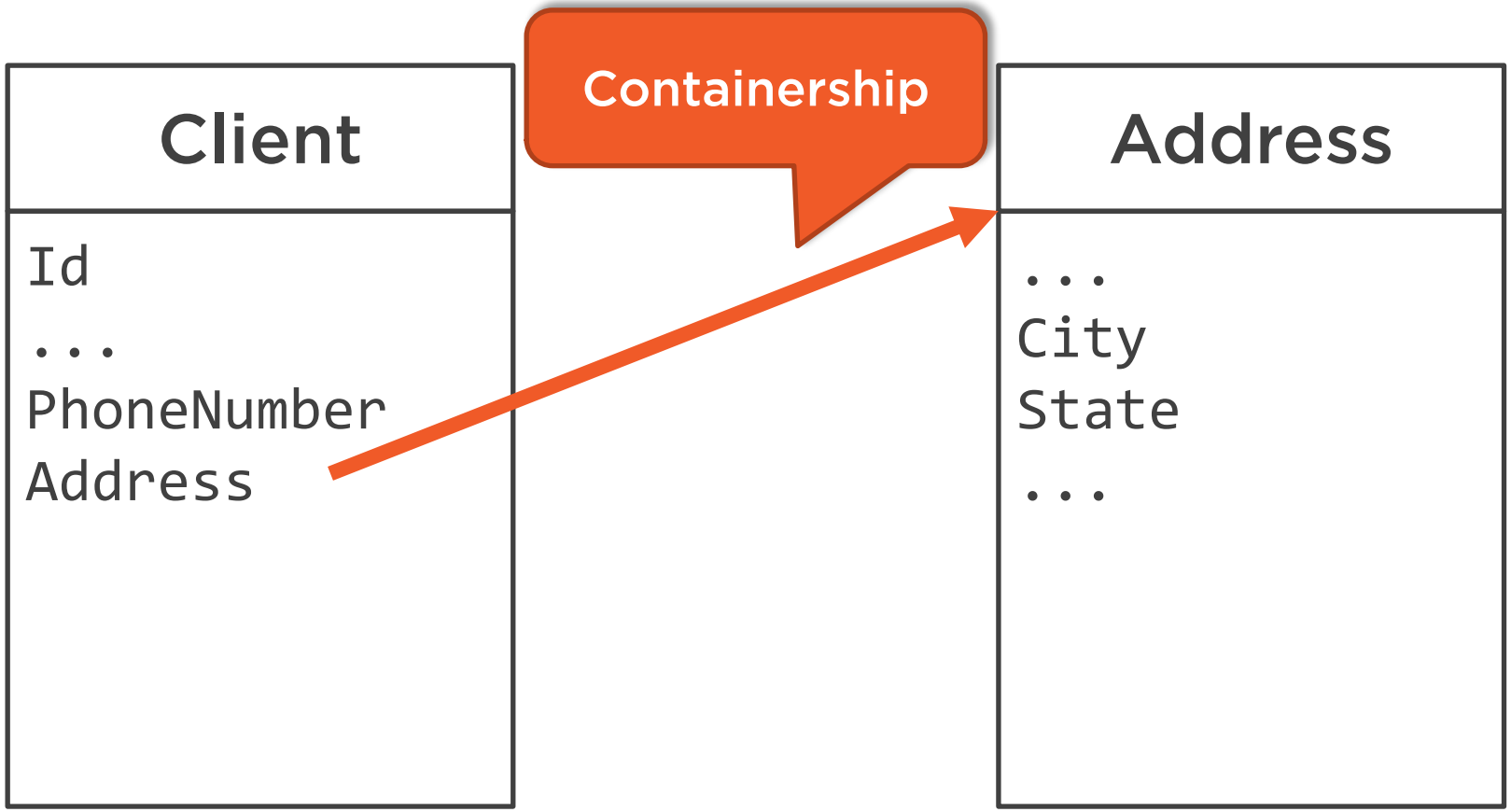
Entity Keys


Client
Id
...
PhoneNumber
Relationship
Case
Id
...
ClientId
Foreign Key

Entity Keys



Owning Relationships



Attributes

Client
Id Name Address PhoneNumber Email IsContracted IsClosed

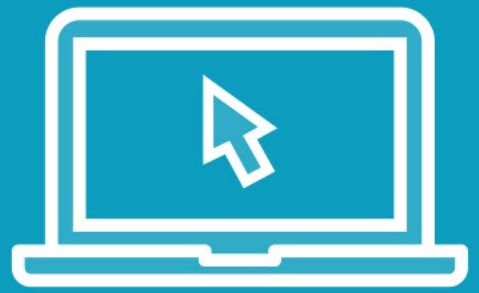
Attribute Types

Simple: Single scalar types (e.g. birthday, name, isRegistered)

Composite: Complex types owned by entity (e.g. Address)

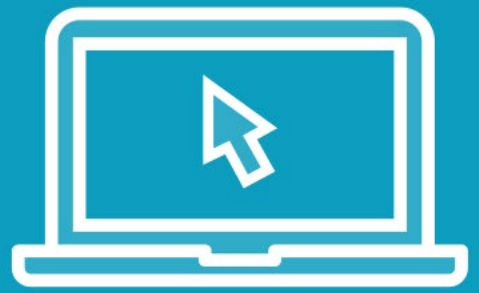
Collections: Multi-valued attributes (e.g. phone numbers)

Demo



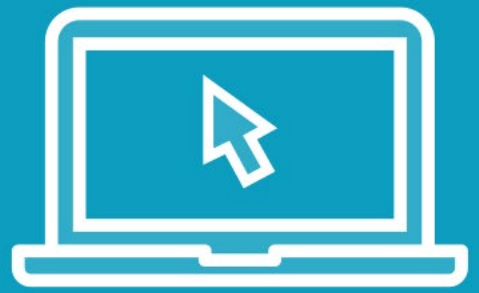
The Project

Demo



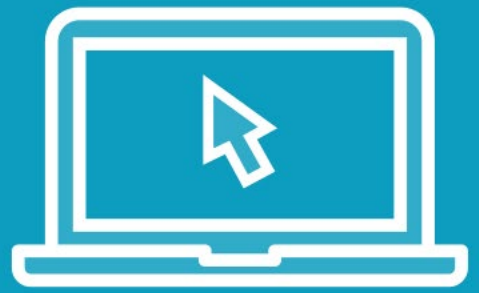
Simple Entities

Demo



Collections

Demo



Owned Entities

Entities not Entity Framework

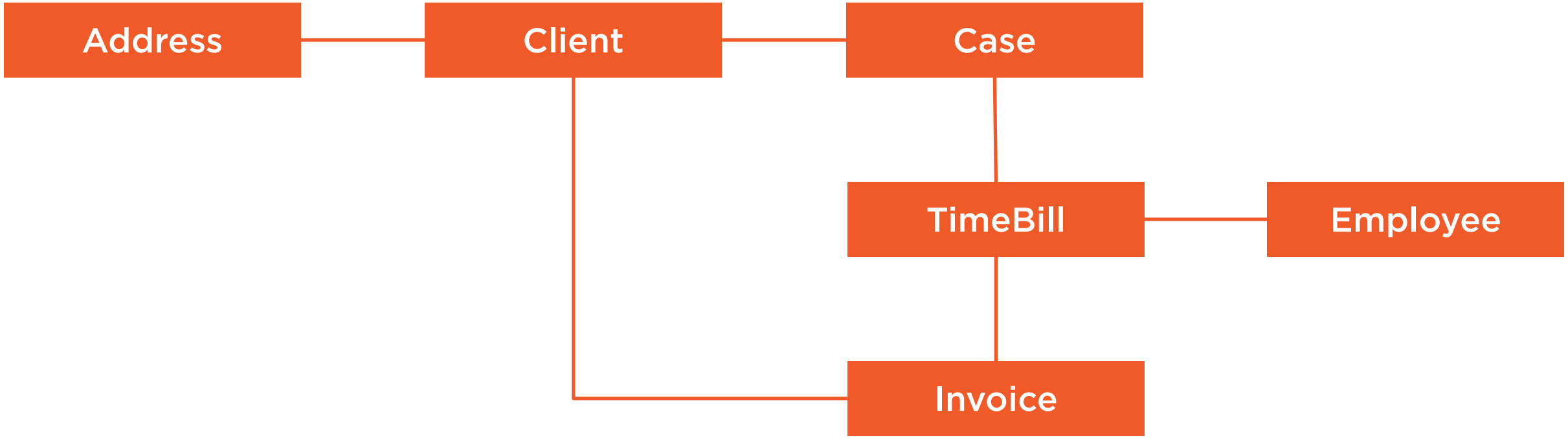
Entities are types of data structures that are persisted between invocations.

E.g. We're not just talking about Entity Framework

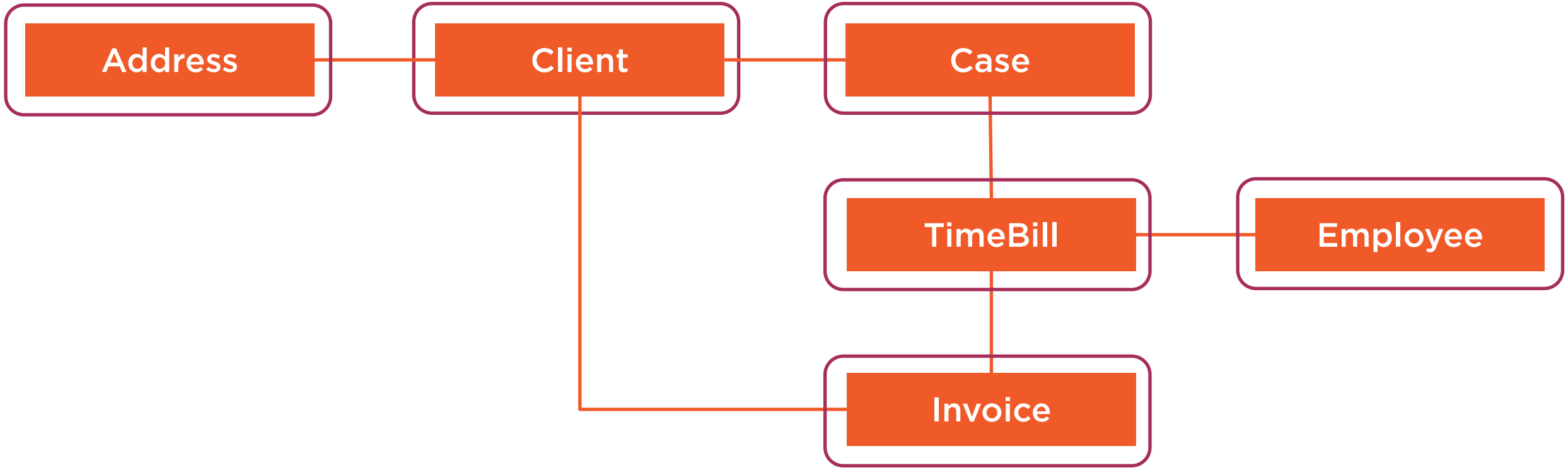
Persistence

- Our example will use SQL/EFCore
- Doesn’t limit your storage choices
  - SQL Server
  - MongoDb
  - CouchDb
  - Postgres
  - Cosmos
  - Etc.

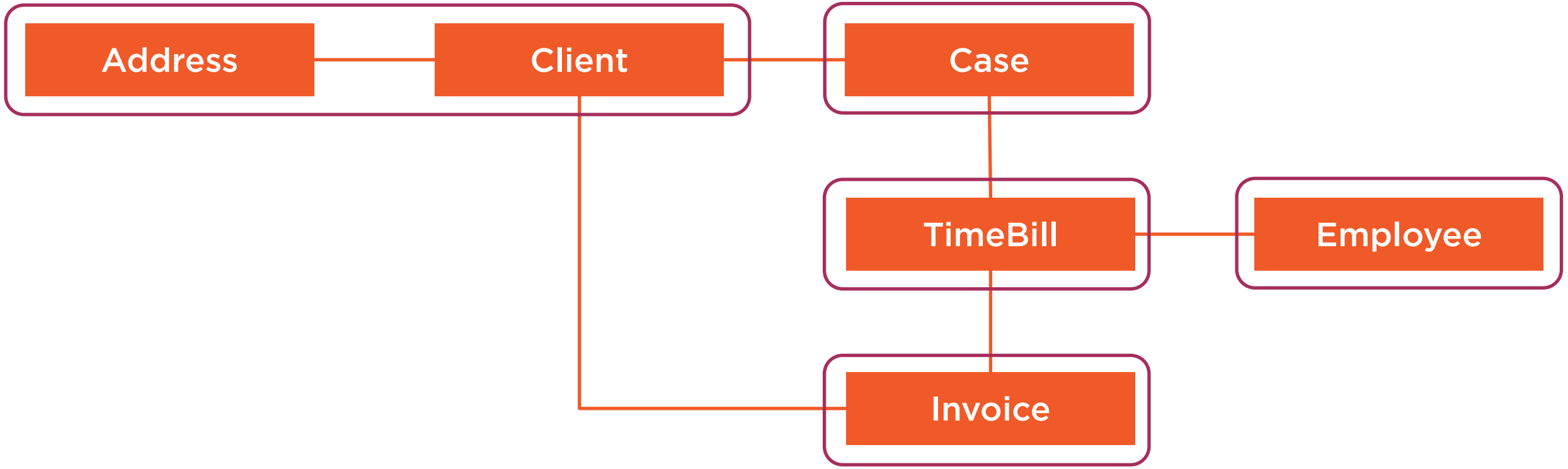
Shapes of Persistence



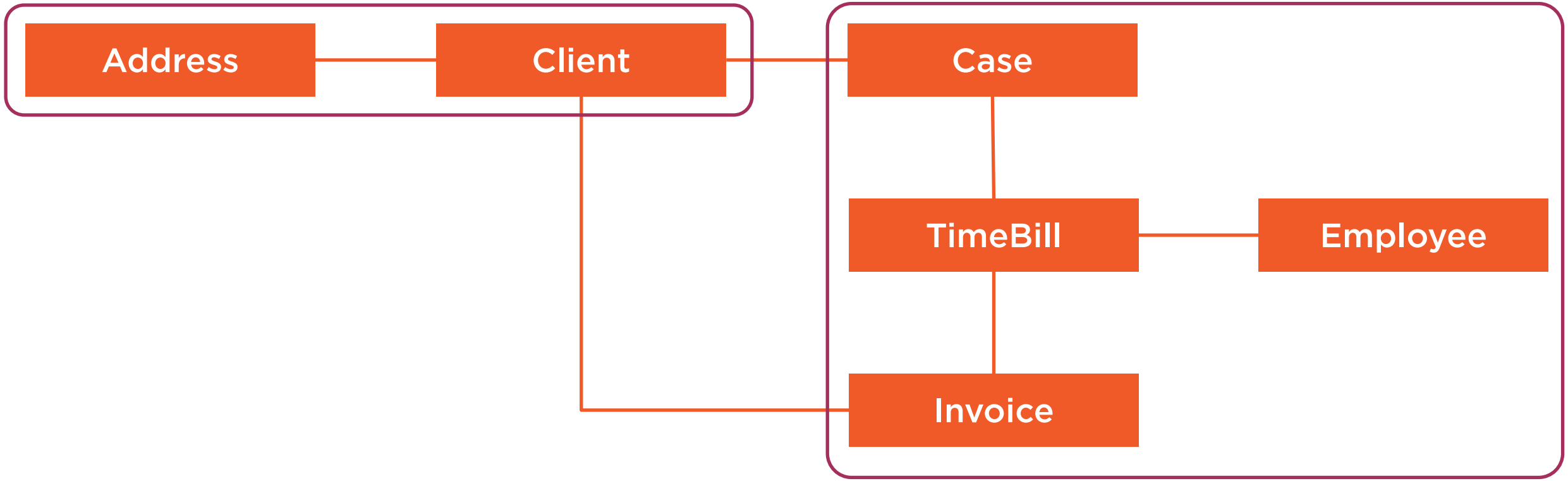
Shapes of Persistence



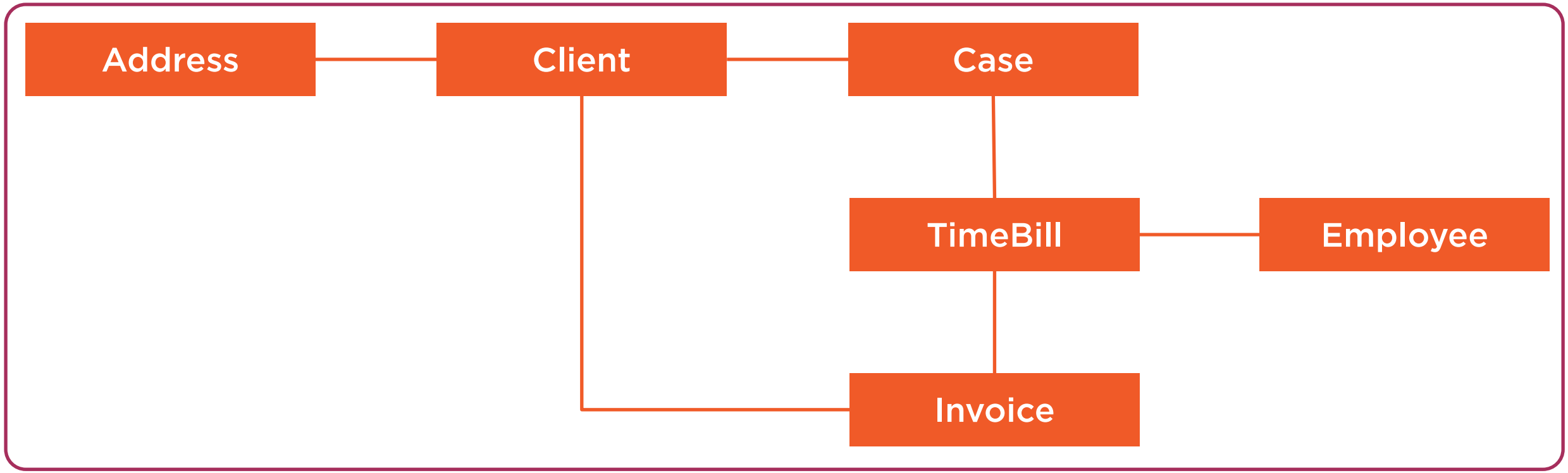
Shapes of Persistence



Shapes of Persistence



Shapes of Persistence



What We've Learned

Entities are persistable data structures that are needed in most apps

Think about entities as a single unit of work instead of 'tables'

It's no longer just SQL; different data needs different stores

Coming Up Next:

Why ViewModels Matter