Mapping Inheritance – Table per Type and Table per Concrete Type

Torben Jensen
Developer/Cloud Architect

Overview

Table-per-Hierarchy

Table-per-Type

Table-per-Concrete-Type

Table-per-Type

All types mapped to individual tables

Separate table for mapping to base type

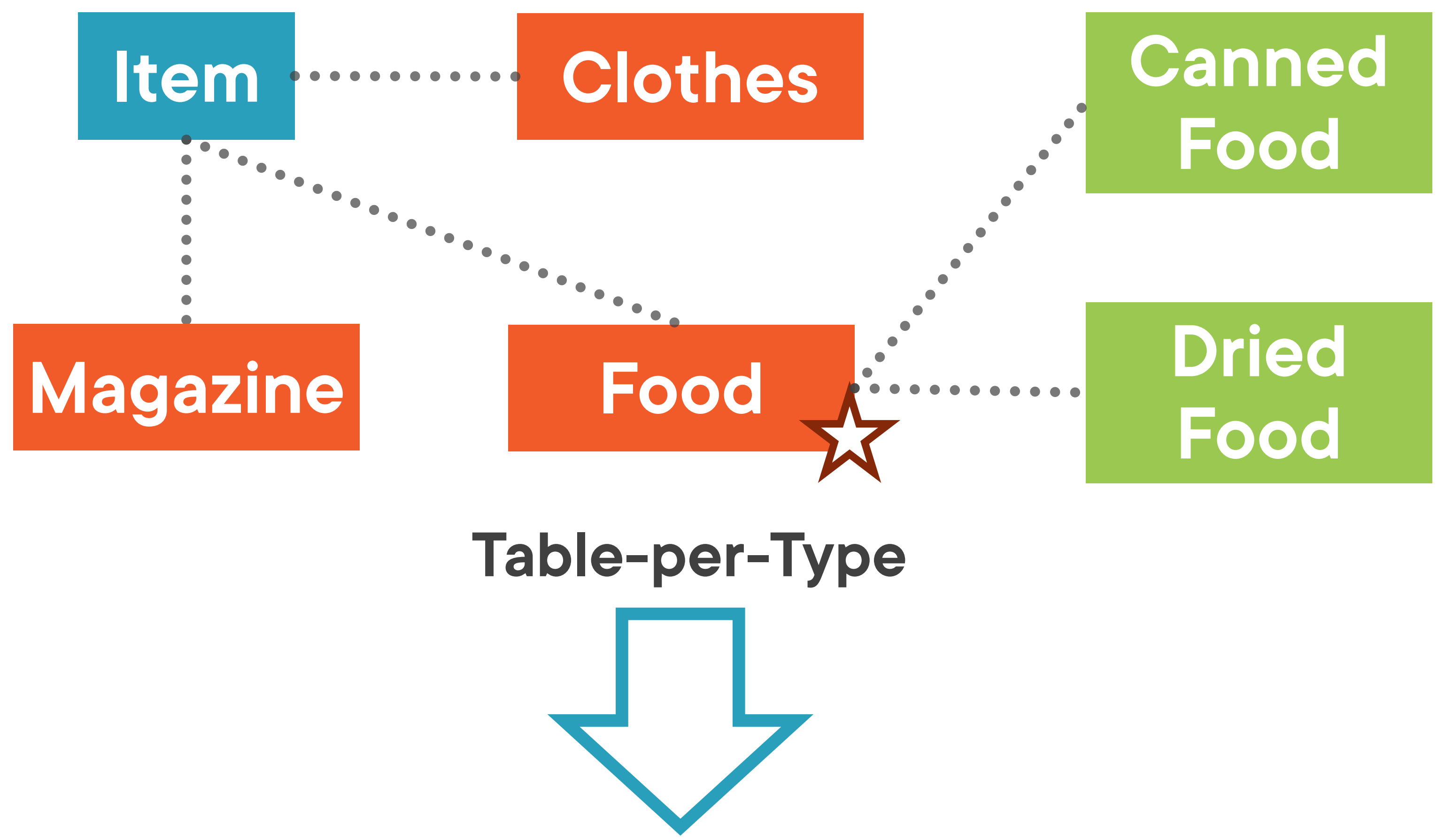
Properties of derived types contained in their specific mapping tables

Derived types store foreign key that maps to base type

Table joins lead to poor performance

No longer recommended

Interpreting Our Model with Table-per-Type



Id	Description	UnitPrice	UnitWeight
1027	Canned Tomato Soup	10	5
1028	Hiking Shoes	40	15
1029	Dried Tomato Soup	7	3
1030	Knotting Rope	3	2
1031	Climber's Quarterly	15	2

Id	Fabric
1028	Leather

Id	PublicationFrequency
1031	Quarterly

Id	DateOfExpiry	ProductionDate
1029	29-11-2032	29-11-2021

Id	CanningMaterial	DateOfExpiry	ProductionDate
1027	Steel	29-11-2025	29-11-2021

Demo

Configuring and interacting with Table-per-Type

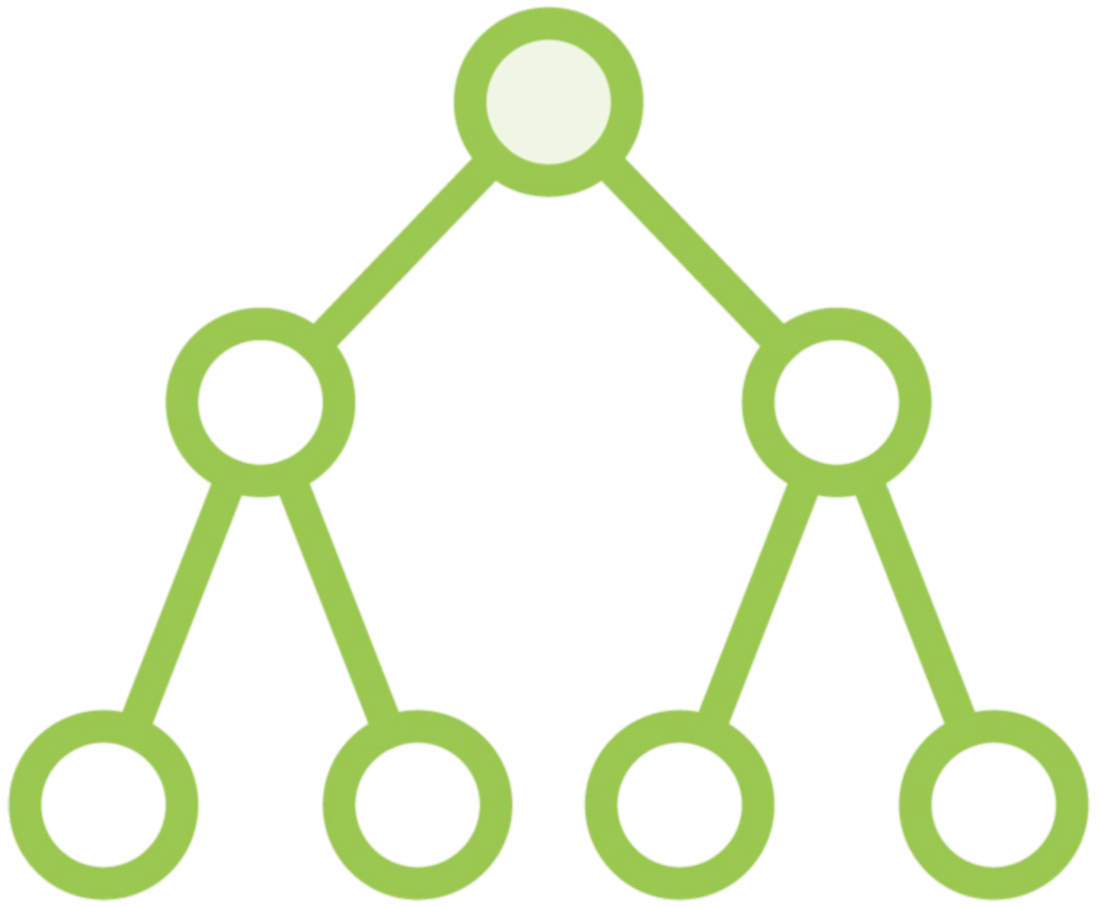
What Did We Just Do?

Mapped model using Table-per-Type

Using the UseTptMappingStrategy method which was introduced in EF Core 7

Explored generated SQL statements

Table-per-Concrete-Type



All concrete types mapped to individual tables

No mapping table for abstract base types

Tables contain all properties of mapped entity

Performance

Addresses performance problems with Table-per-Type

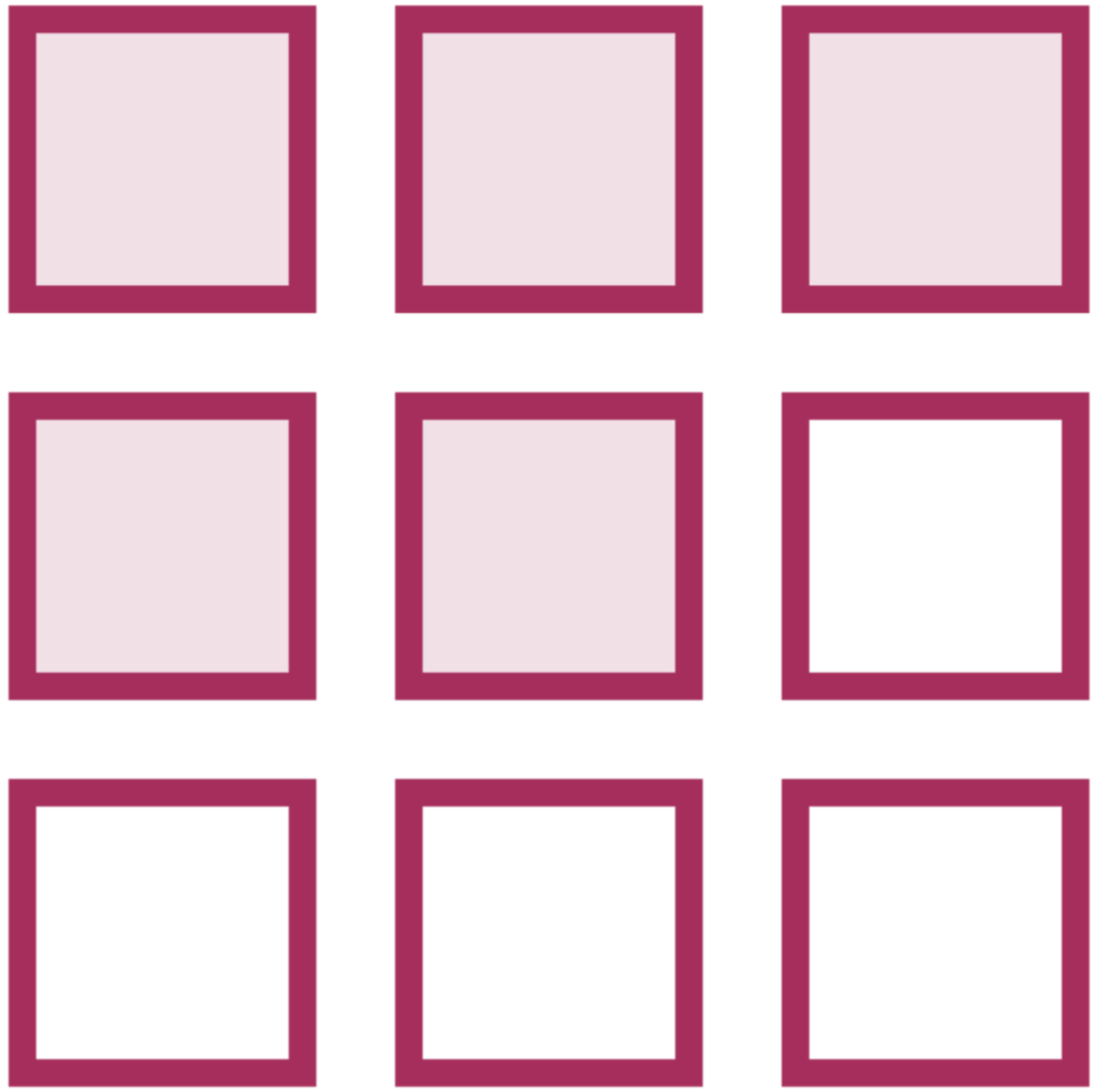
Does not require joins

Data is unioned instead

Faster than Table-per-Type for querying several entity types

Much faster than Table-per-Type for querying leaf type entities

Denormalized Data

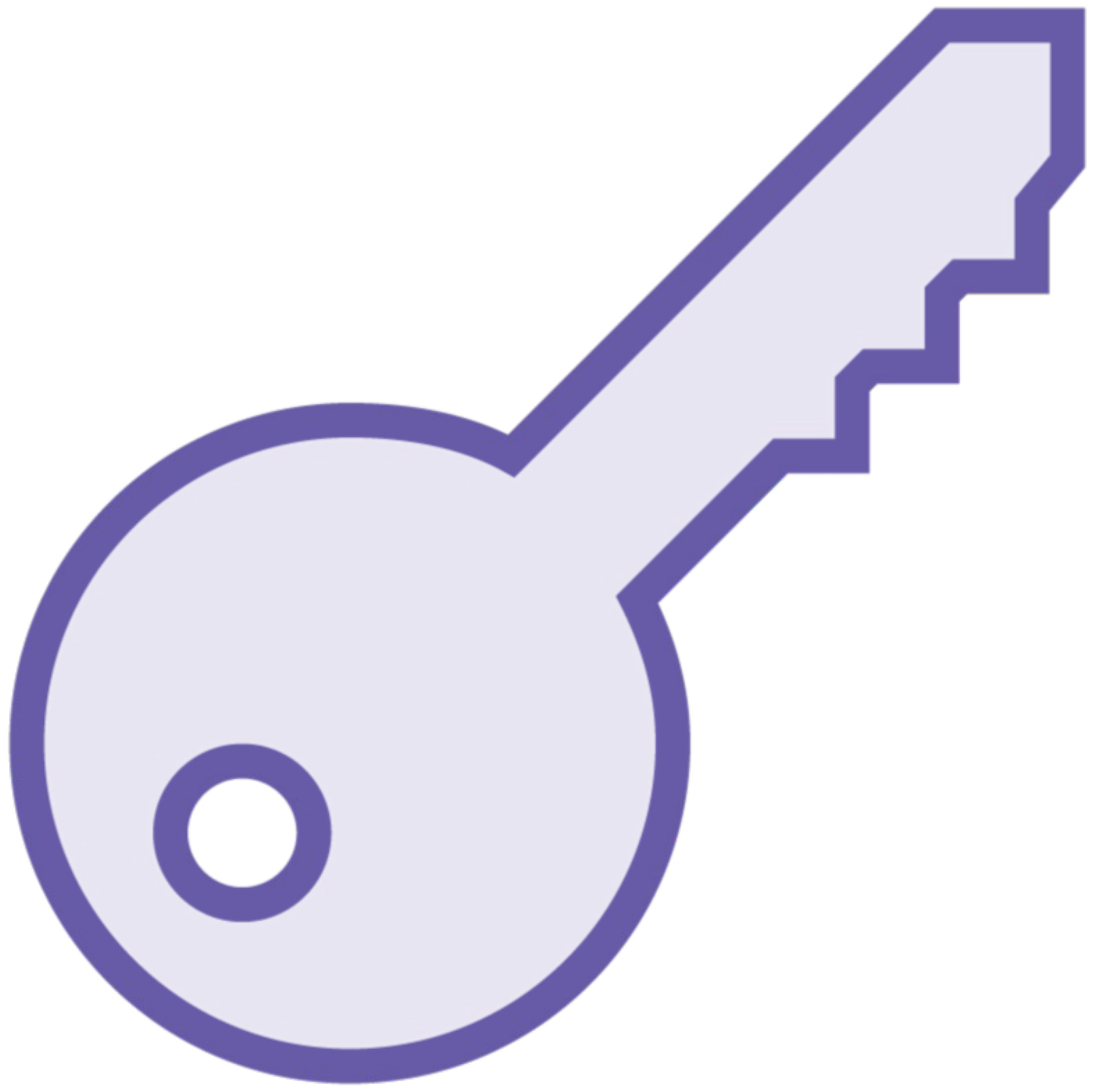


Denormalized data

All entities in type hierarchy must have unique keys

Table-per-Type and Table-per-Hierarchy can use Identity columns for primary keys

Primary Keys



We cannot use Identity columns as primary key for Table-per-Concrete-Type

Ids are generated using sequences

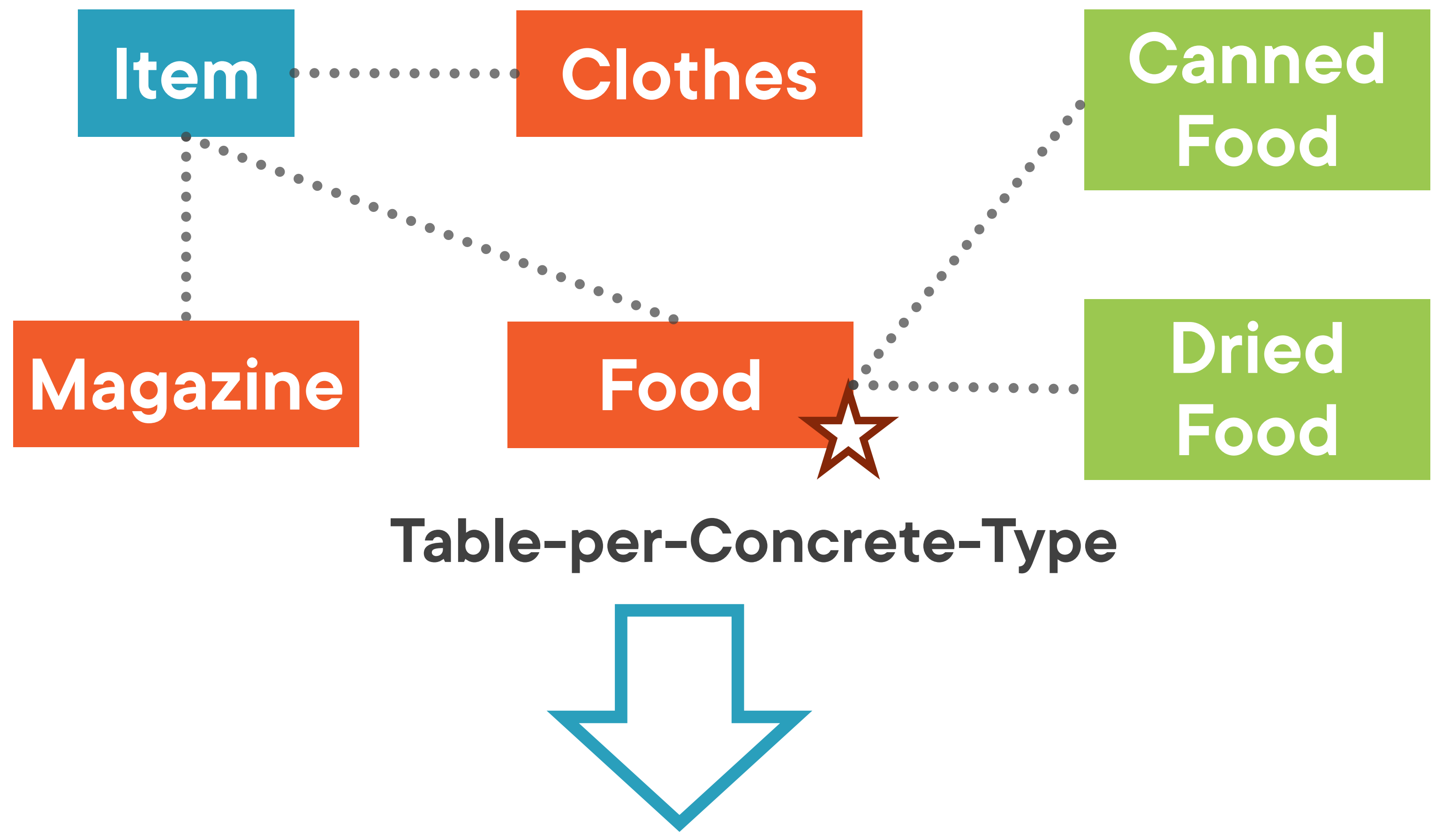
Cannot create foreign keys to base type

We can reference base type entities

But it is not a proper foreign key relationship

EF Core guarantees valid key values

Our Model with Table-per-Concrete-Type



Database Tables with Table-per-Concrete-Type

CannedFoodItems

Id	Description	UnitPrice	UnitWeight	CanningMaterial	DateOfExpiry	ProductionDate
1027	Canned Tomato Soup	10	5	Steel	29-11-2025	29-11-2021

ClothesItems

Id	Description	UnitPrice	UnitWeight	Fabric
1028	Hiking Shoes	40	15	Leather

DriedFoodItems

Id	Description	UnitPrice	UnitWeight	DateOfExpiry	ProductionDate
1029	Dried Tomato Soup	7	3	29-11-2032	29-11-2021

MagazineItems

Id	Description	UnitPrice	UnitWeight	PublicationFrequency
1031	Climber's Quarterly	15	2	Quarterly

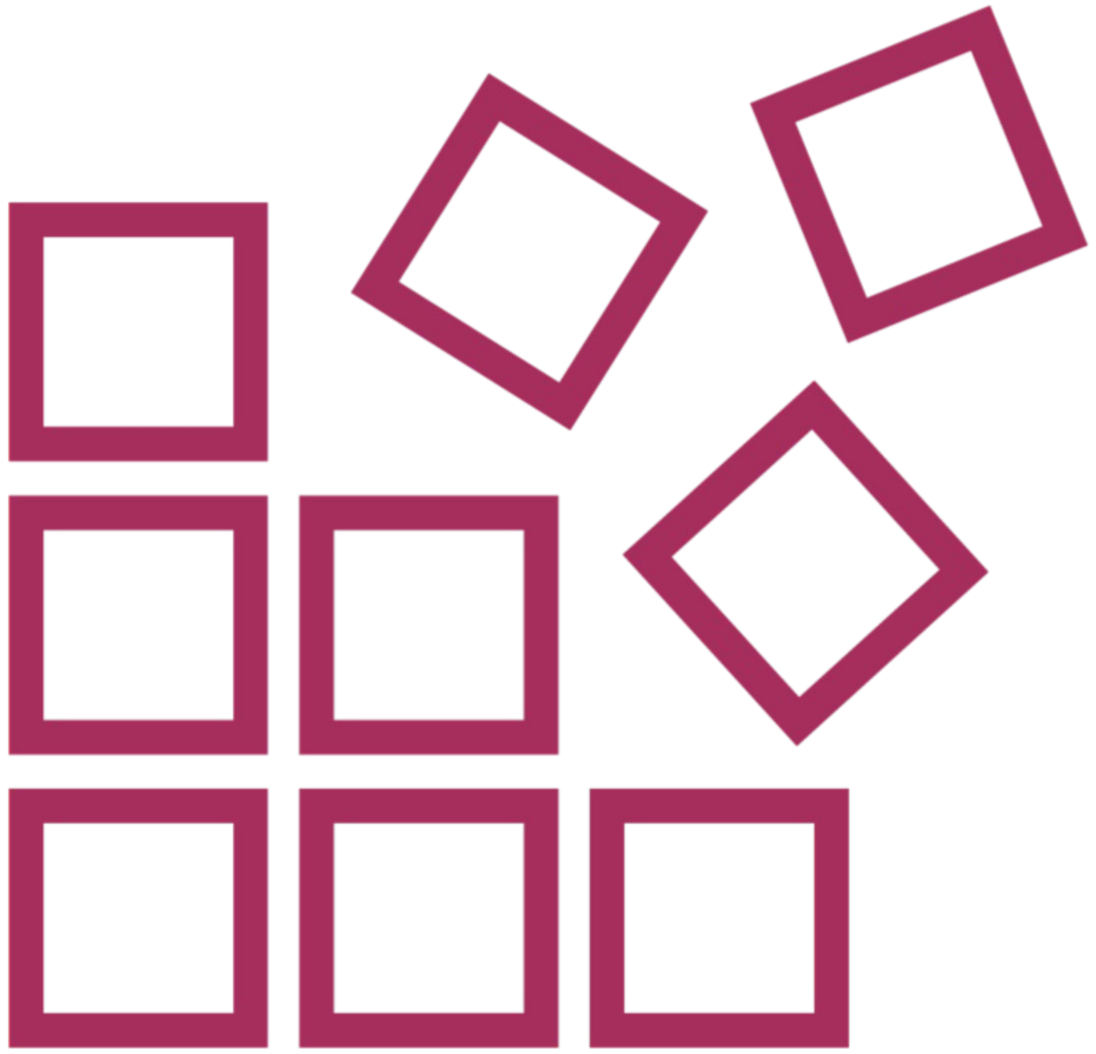
Items

Id	Description	UnitPrice	UnitWeight
1030	Knotting Rope	3	2

Demo

Configuring and interacting with Table-per-Concrete-Type

What Did We Just Do?



Mapped model using Table-per-Concrete-Type

Created denormalized version of Table-per-Type schema

Explored generated SQL statements

Improved performance

Which Mapping to Choose?

Table-per-Type more similar to our .NET type hierarchy

Table-per-Hierarchy leads to empty columns

Mapping table may become cluttered

Table-per-Concrete type denormalizes

Can be difficult to use on existing databases

Which Mapping to Choose?

Table-per-Hierarchy is suitable for most cases

Table-per-Type has inferior performance in most cases

Because table joining is needed

Most databases handle empty columns efficiently

Sparse columns improve Table-per-Hierarchy performance

Table-per-Concrete-Type?

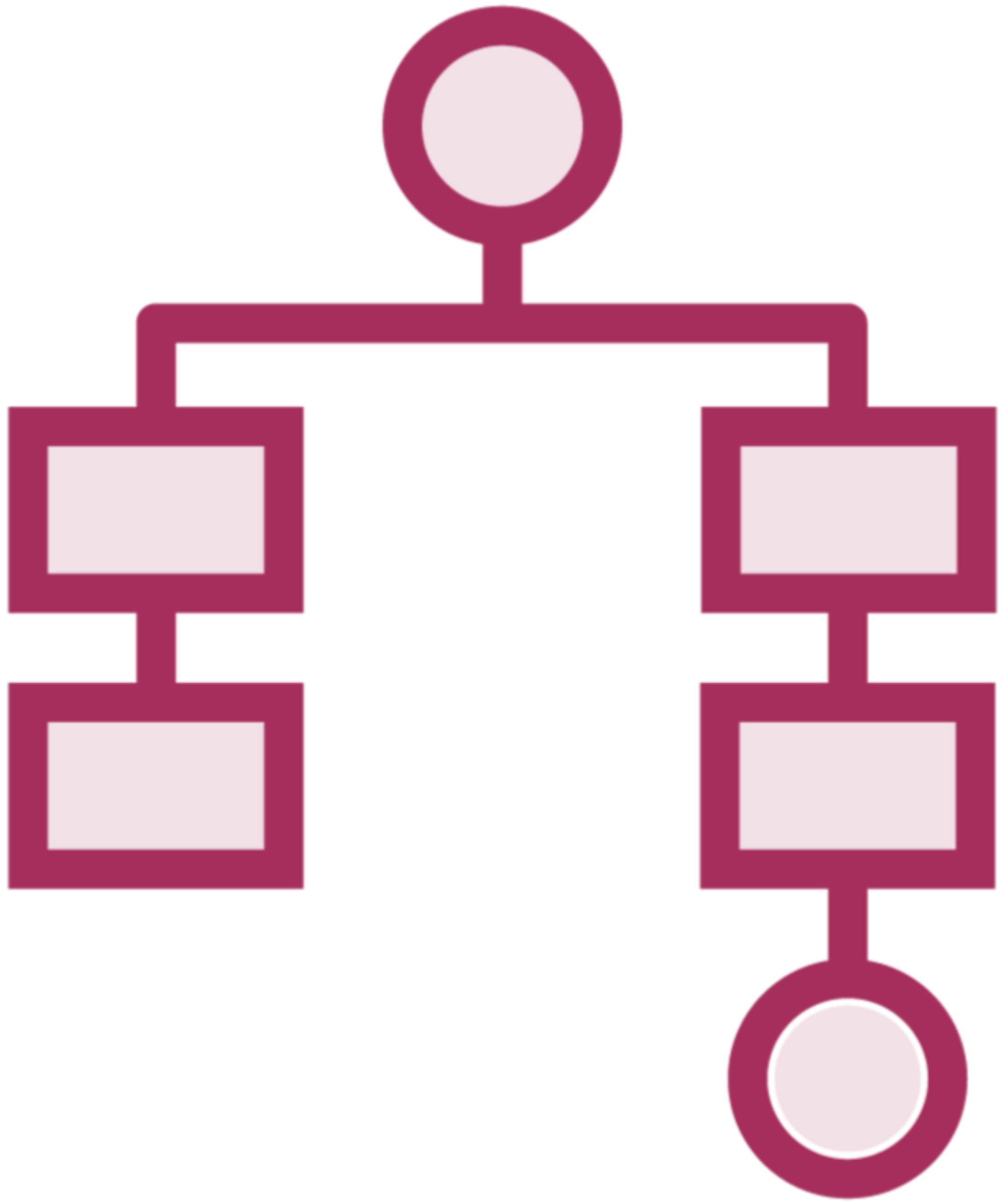


Table-per-Concrete-Type performs like Table-per-Hierarchy

Especially good for querying leaf type entities

Denormalization might be an issue

But it is excellent for single table queries

Not ideal for entities that are already mapped

Best when starting from scratch

Table-per-Hierarchy!

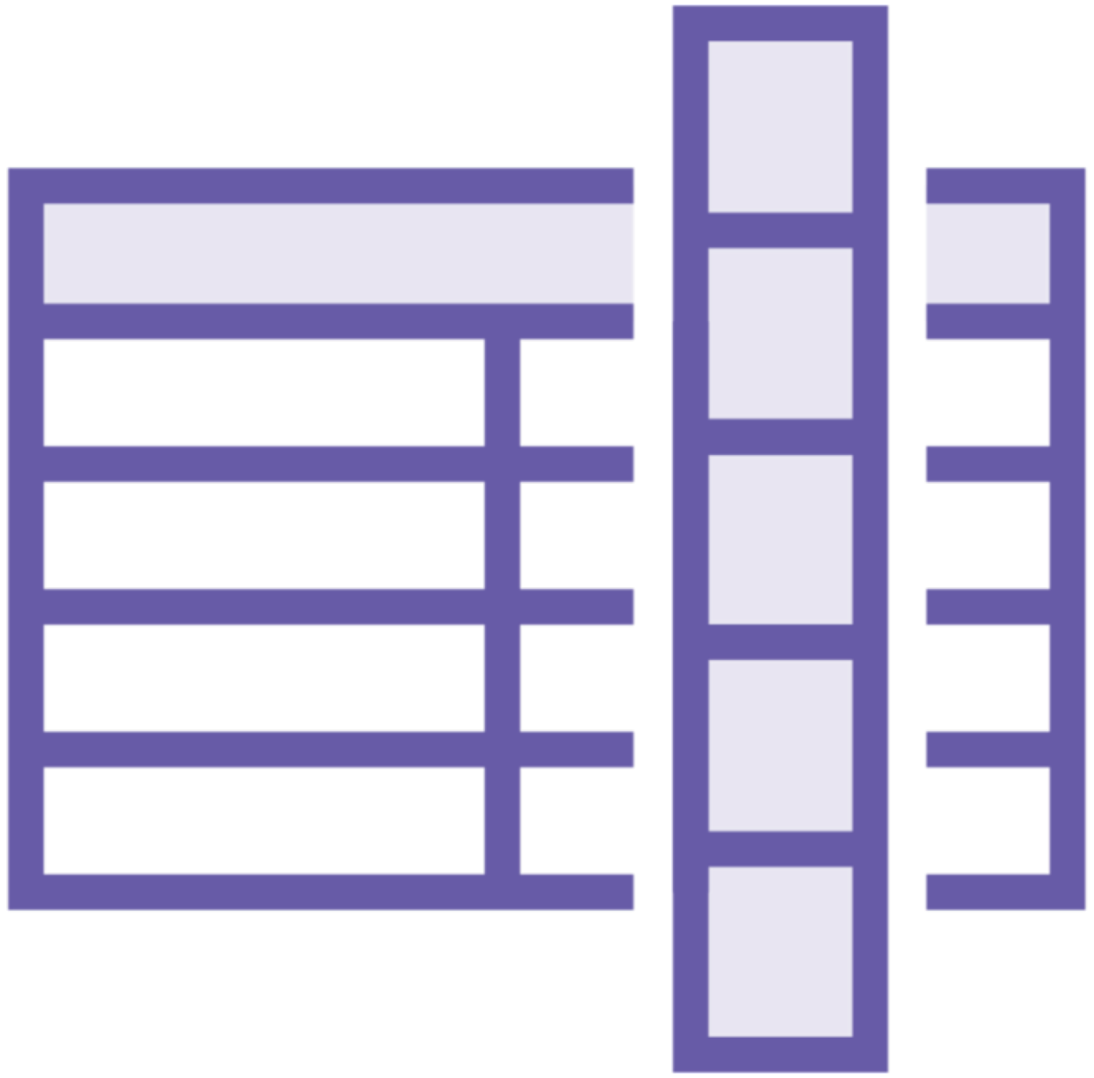


Table-per-Hierarchy works for most cases

Best choice when querying for different types of entites

Table-per-Concrete-Type is good when querying for entities of a single leaf type

Benchmark before deciding

Choice of strategy has long-term implications

Avoid Table-per-Type if possible

Summary

Mapping inheritance

Table-per-Type

Table-per-Concrete-Type

EF Core does magic behind the scenes

Summary

Table-per-Hierarchy for most cases

Table-per-Concrete-Type when mainly querying leaf type entities

Table-per-Type should be avoided