

CoreData

大容量データ マイグレーション

Yuki Yasoshima

Ubiregi Inc.

twitter : @yaso_san
objective-audio.jp

CoreDataの マイグレーションは 大量にメモリを消費する

マイグレーションの仕組み

- **Lightweight Migration**
- **Custom Migration**

Three Stage Migration



Stage 1

オブジェクトの生成とコピー

**Source
Context**

Version 1

古いバージョン

**Destination
Context**

Version 2

新しいバージョン

MigrationManagerには
古いバージョンと新しいバージョンの
ManagedObjectContextが作られる

Source
Context

Destination
Context

Version 1

Version 2



オブジェクトを
コピーして移し換える

Source Context

Destination Context

Entity

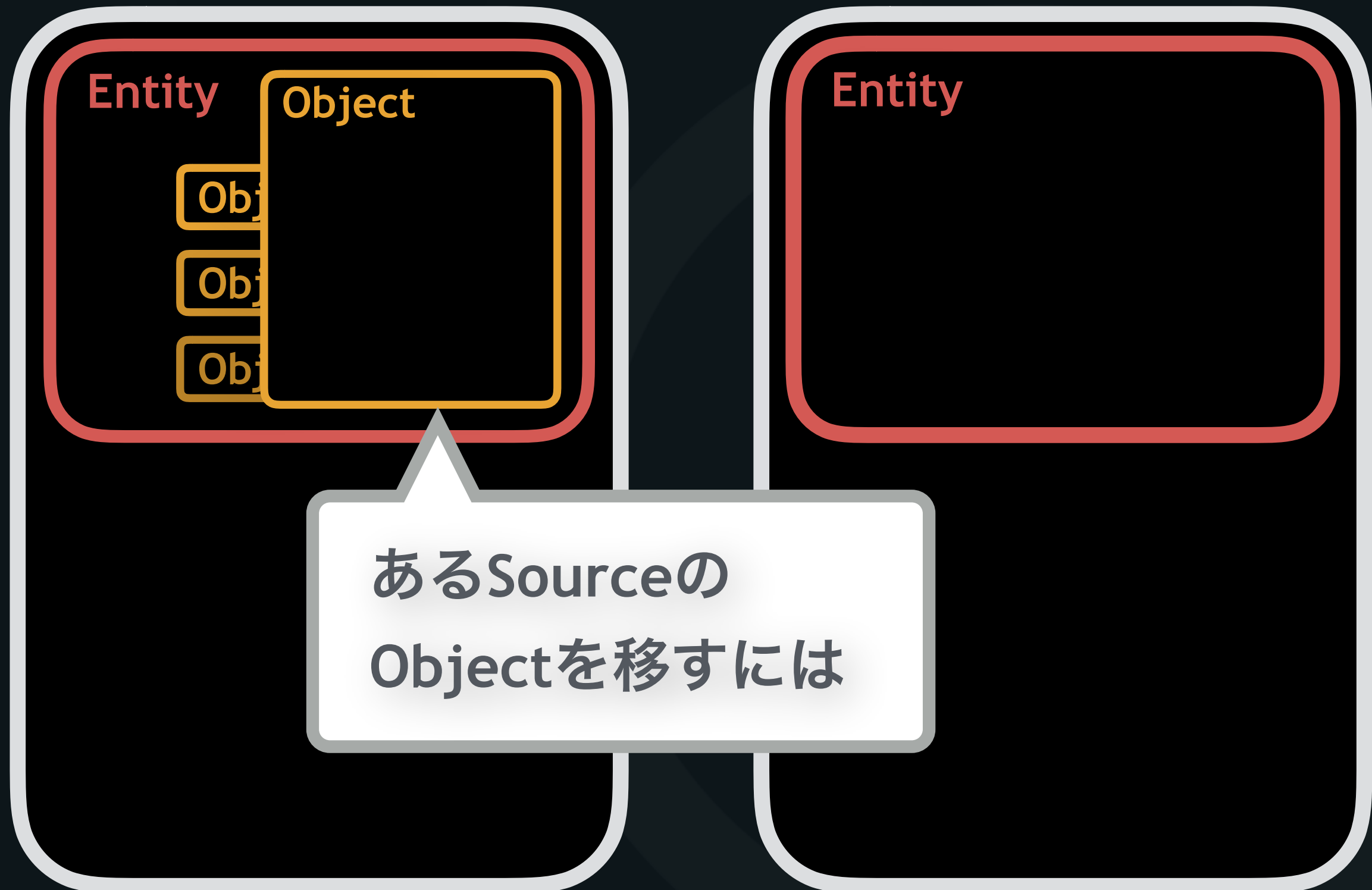
Object Object
Object Object
Object Object

Entity

SourceContextに
オブジェクトをフェッチ

Source Context

Destination Context



Source Context

Entity

Object

Obj

Obj

Obj

Destination Context

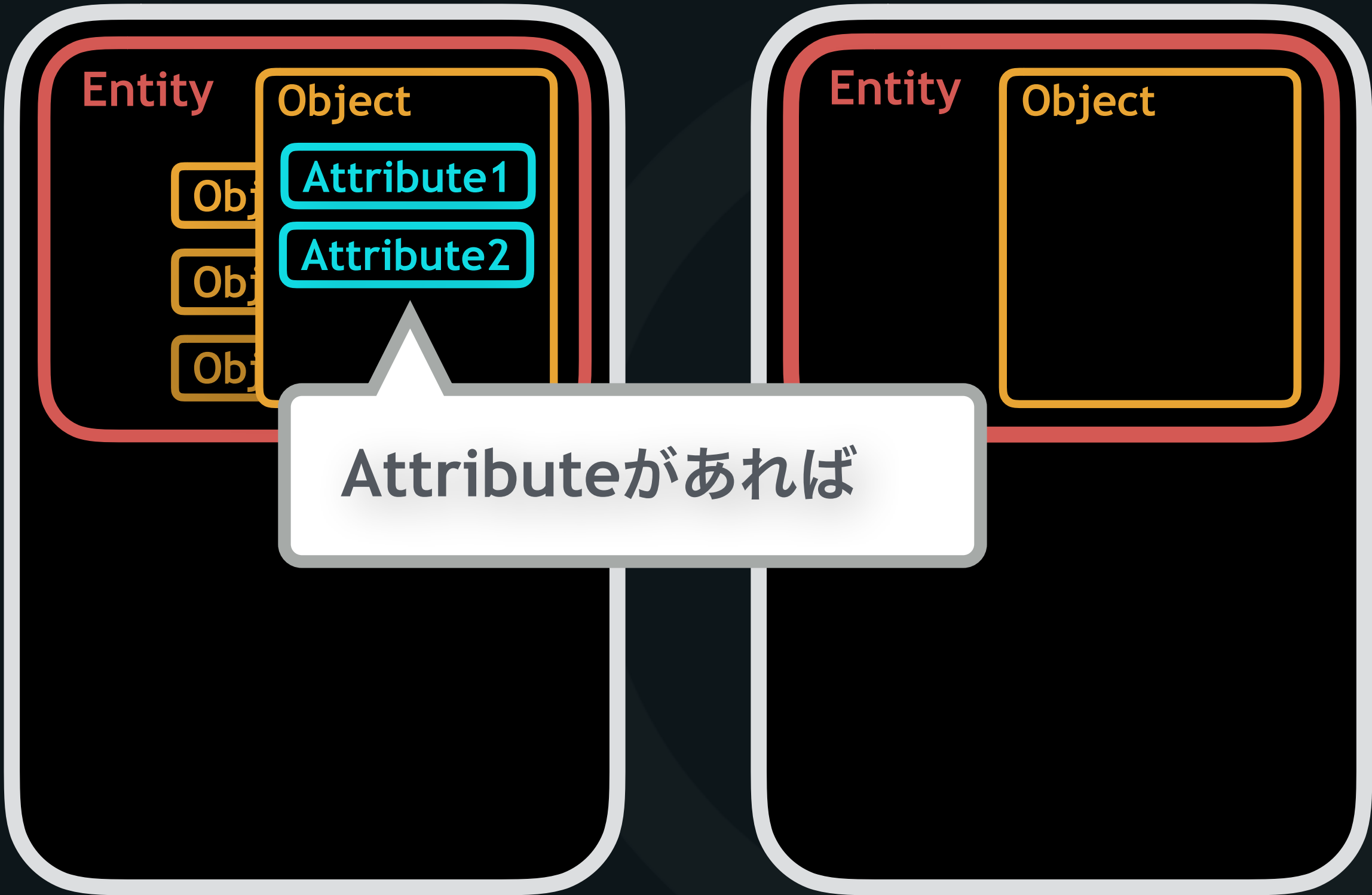
Entity

Object

DestinationContextに
オブジェクトを生成

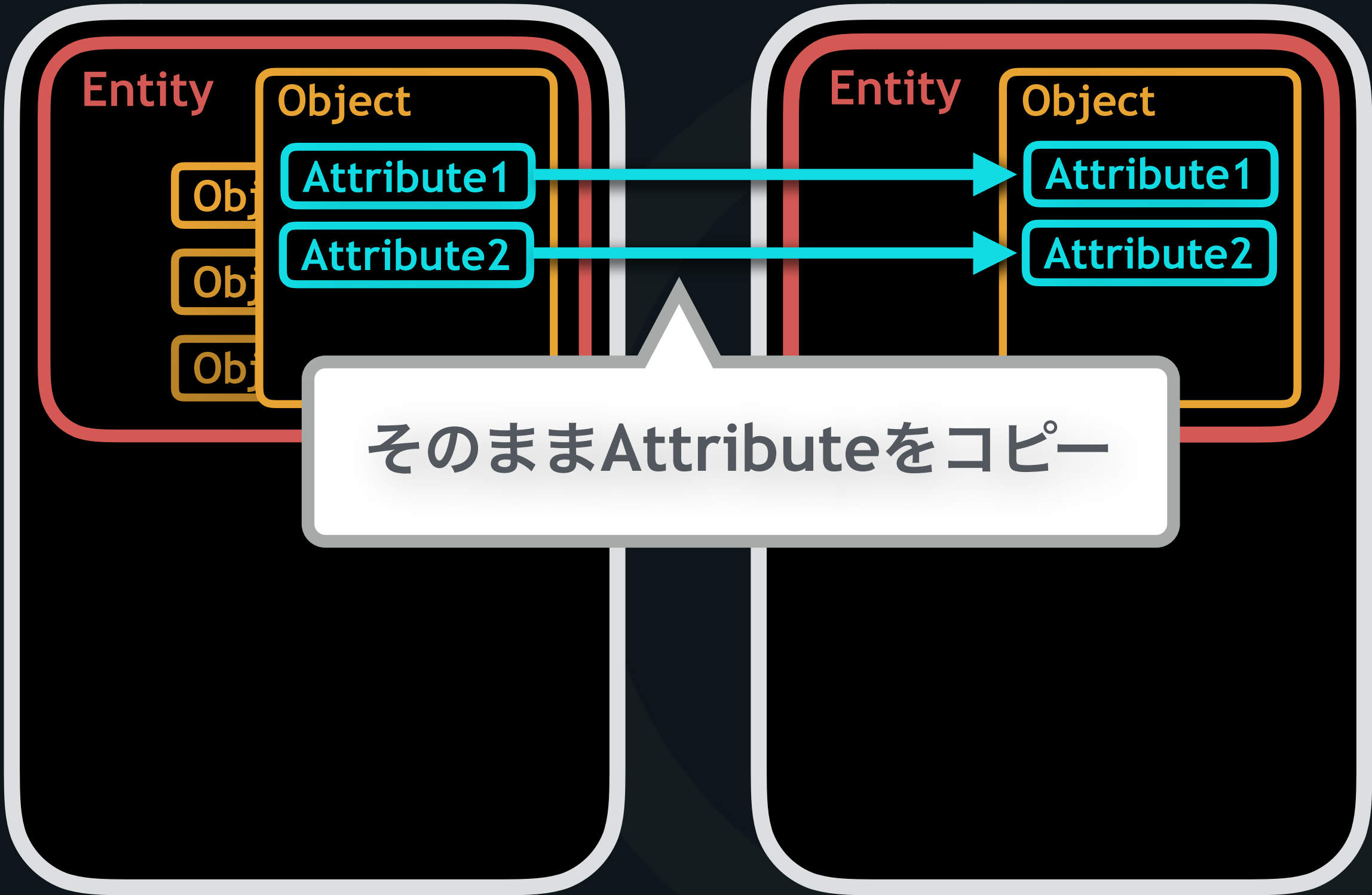
Source
Context

Destination
Context



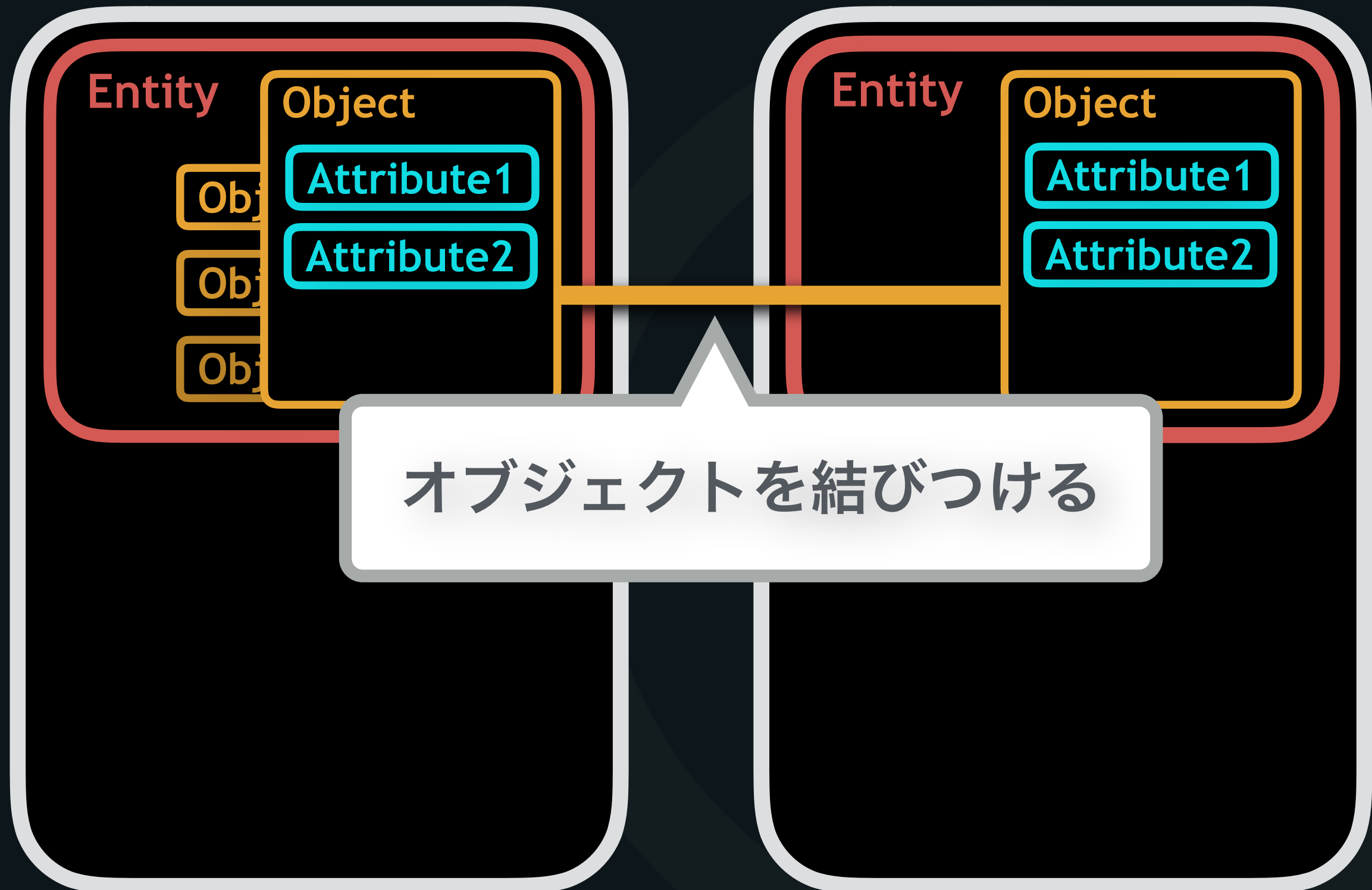
Source
Context

Destination
Context



Source Context

Destination Context



Source Context

Destination Context

EntityA

- Object A
- Object A
- Object A

EntityA

- Object A
- Object A
- Object A

EntityB

- Object B
- Object B
- Object B

EntityB

- Object B
- Object B
- Object B

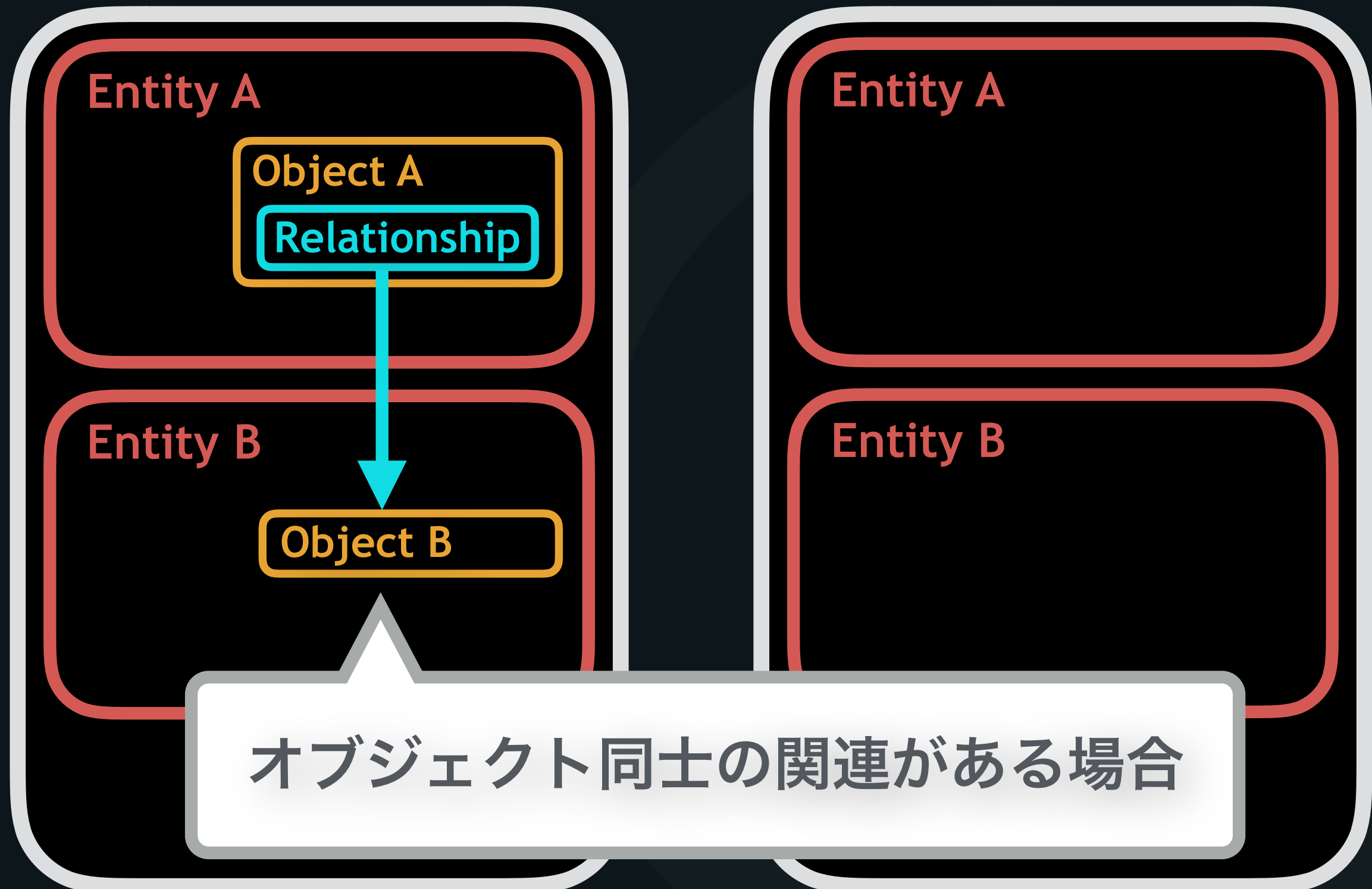
全オブジェクト繰り返し

Stage 2

関連をセット

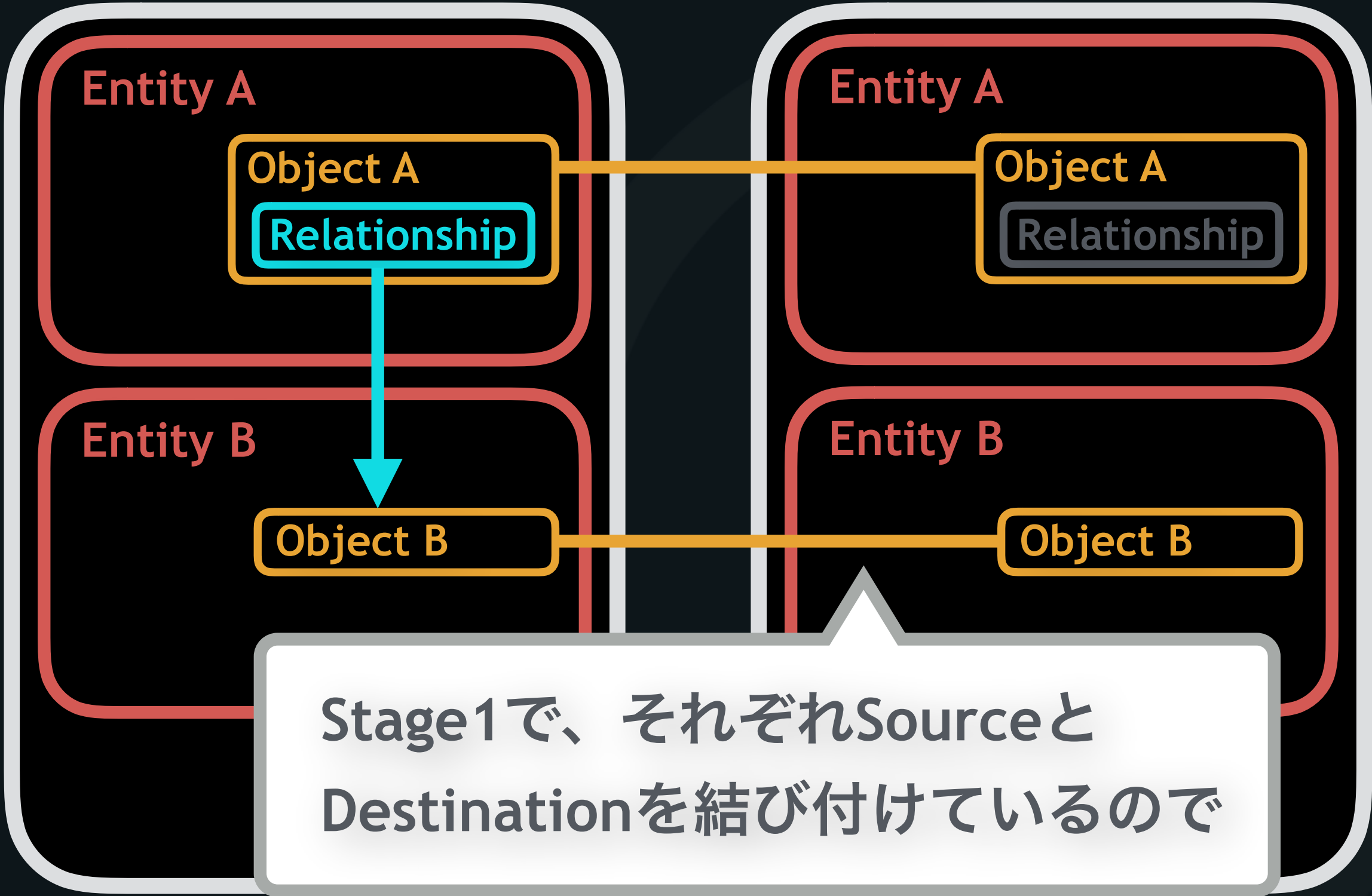
Source Context

Destination Context



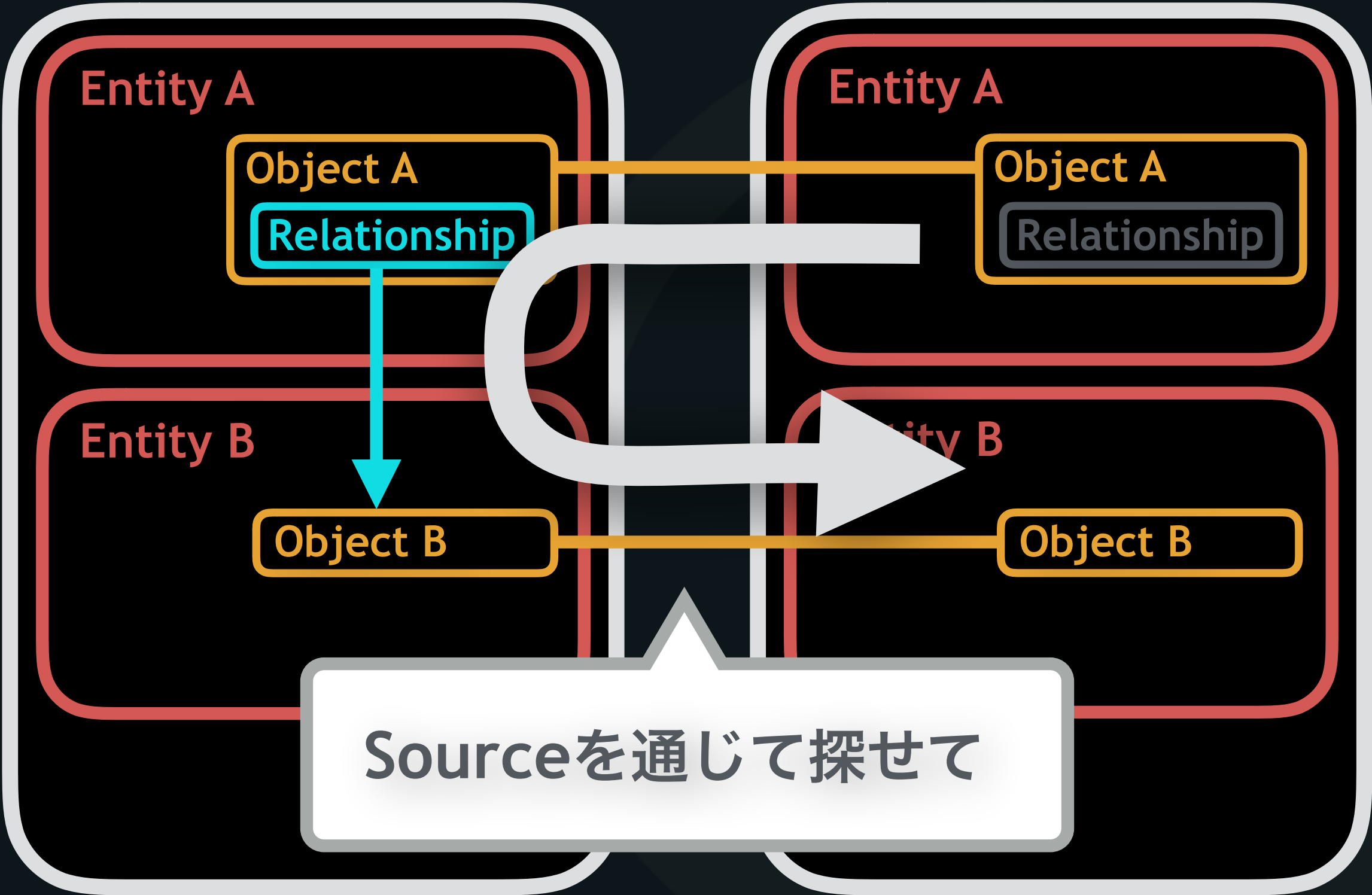
Source
Context

Destination
Context



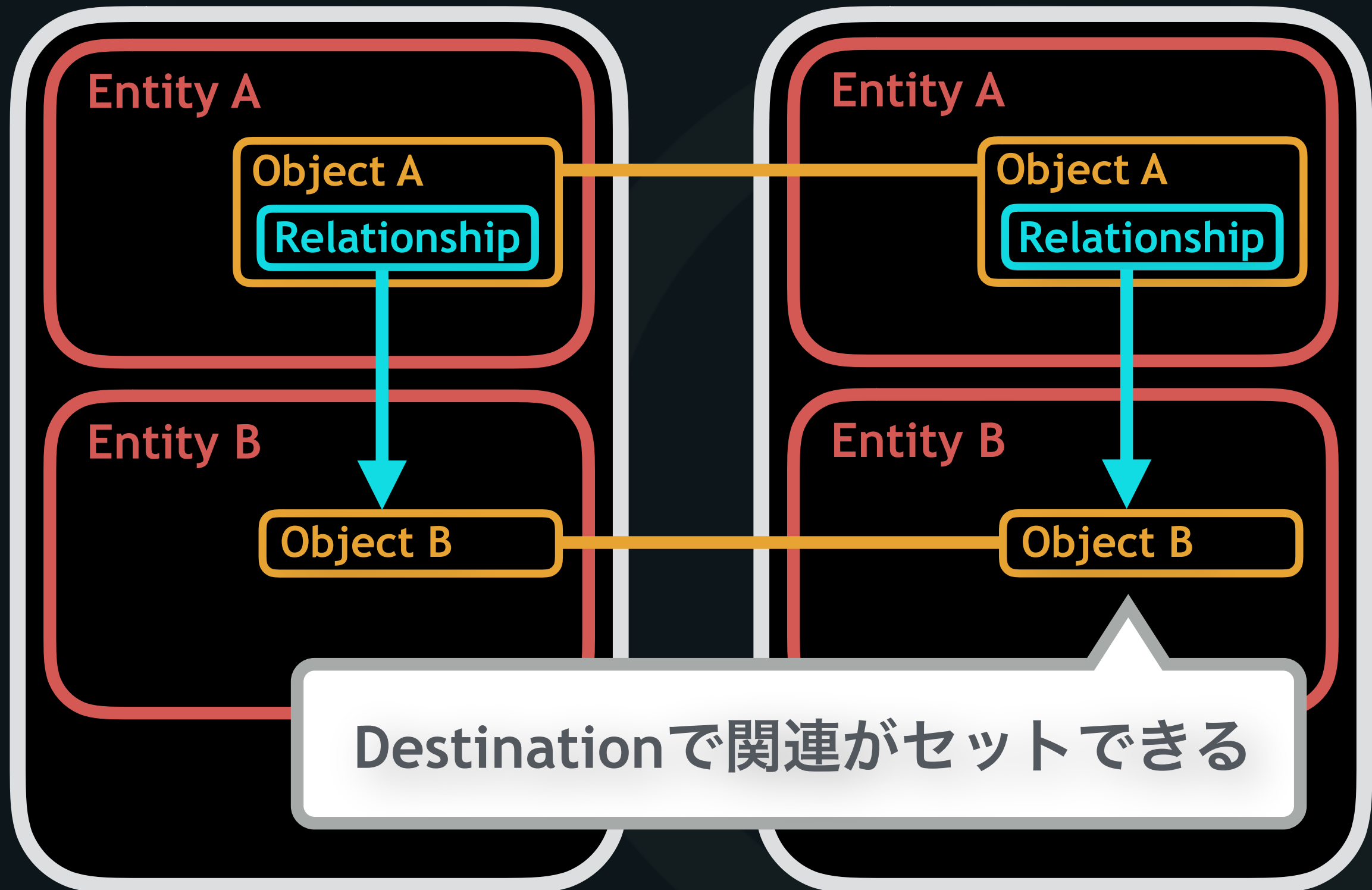
Source
Context

Destination
Context



Source Context

Destination Context



Stage 3

検証

なぜ、メモリを消費するのか？

データを引き継ぐために
すべてのオブジェクトを
メモリに載せているから

大容量のデータの
マイグレーションを
終わらせるには

Core Data Model Versioning and Data Migration Programming Guide

Customizing the Migration Process

Mapping Modelを分けて
マイグレーションを
forループで分割して実行

マイグレーション分割の壁

マイグレーション分割の壁 1

分割する数だけ
Mapping Modelが必要

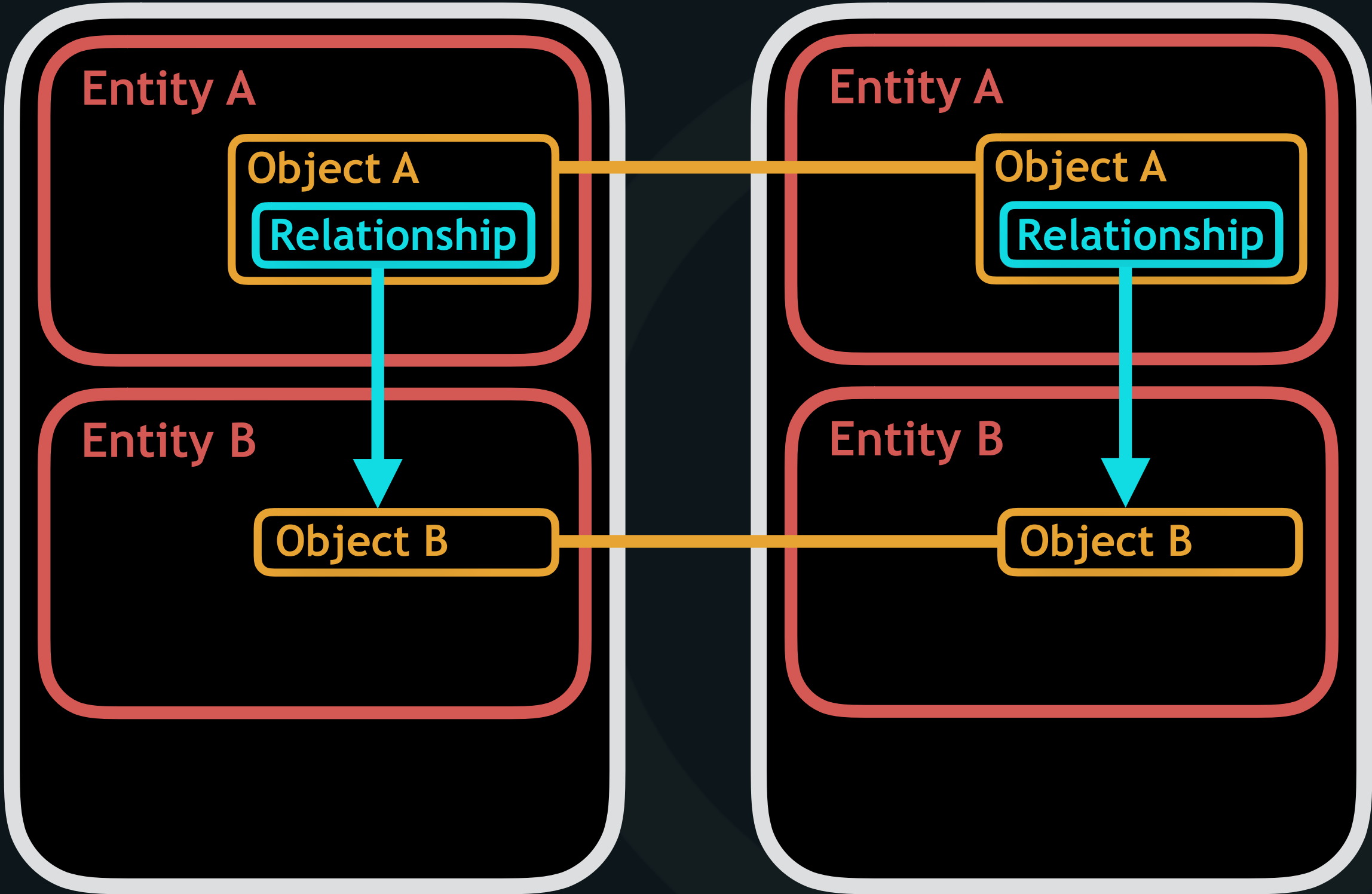
作るのも管理も大変

マイグレーション分割の壁 2

SourceとDestinationの
オブジェクトの結びつきは
分割すると引き継げない

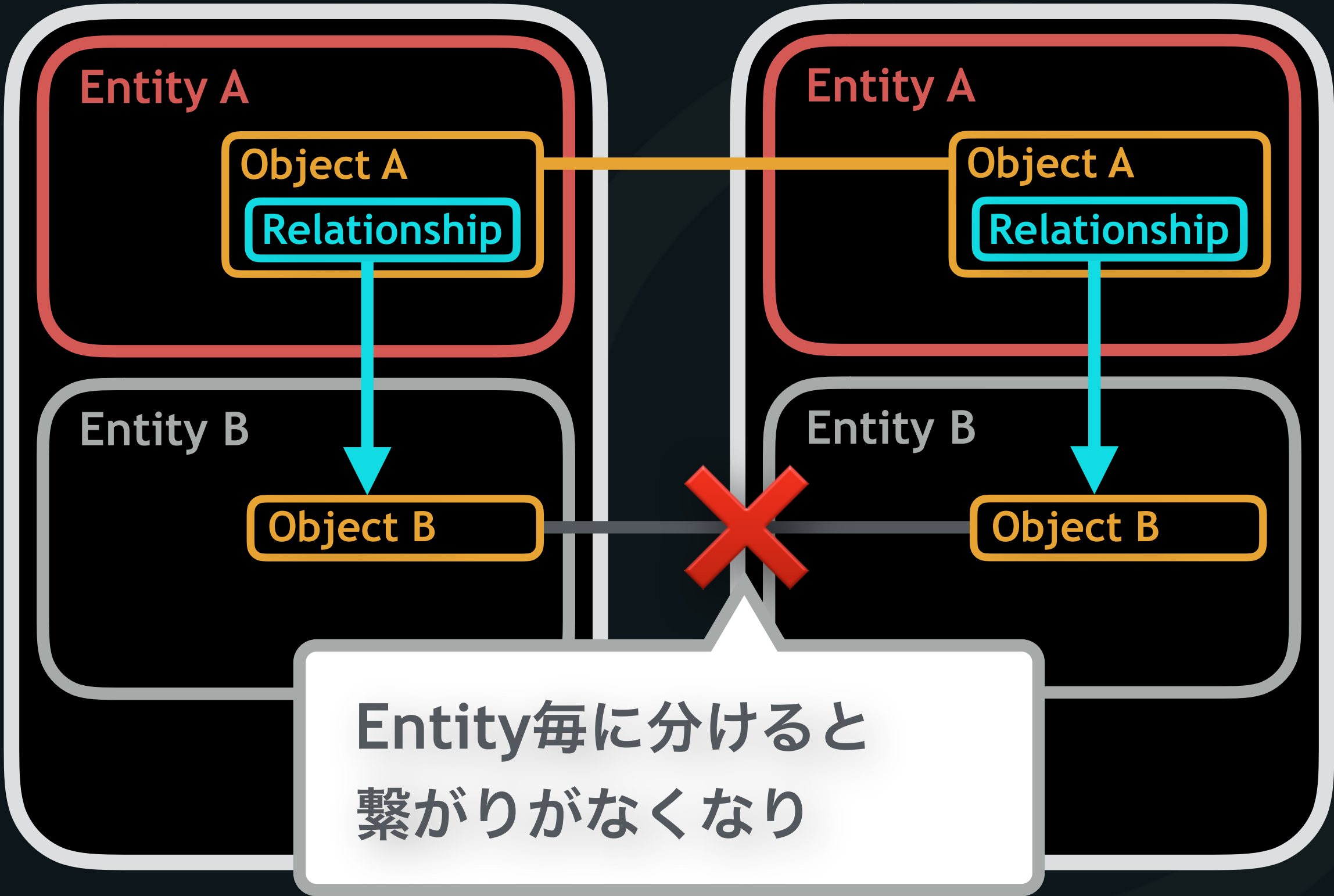
Source Context

Destination Context



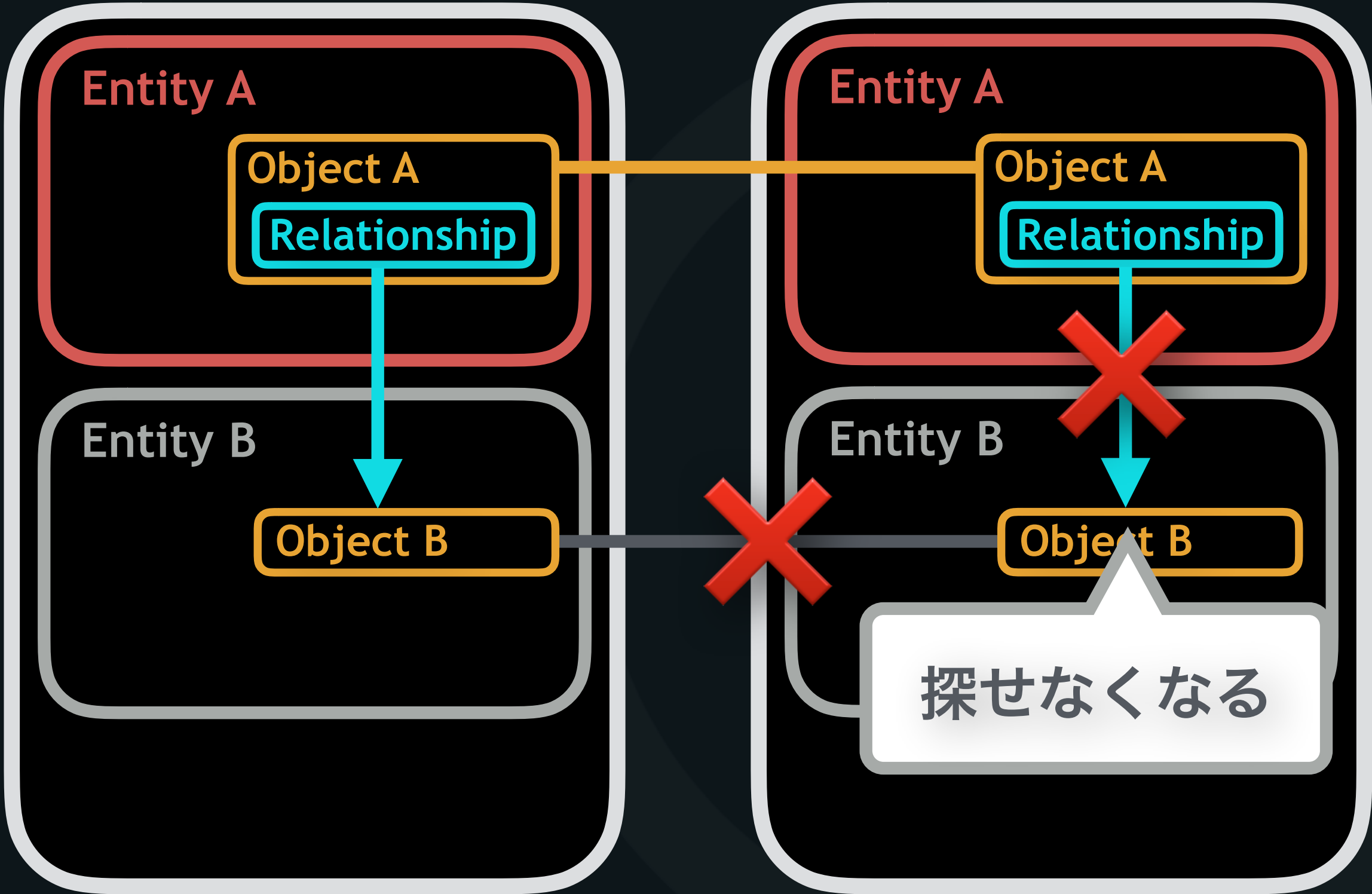
Source
Context

Destination
Context



Source Context

Destination Context



マイグレーション分割の壁 3

分割した数だけ時間がかかる

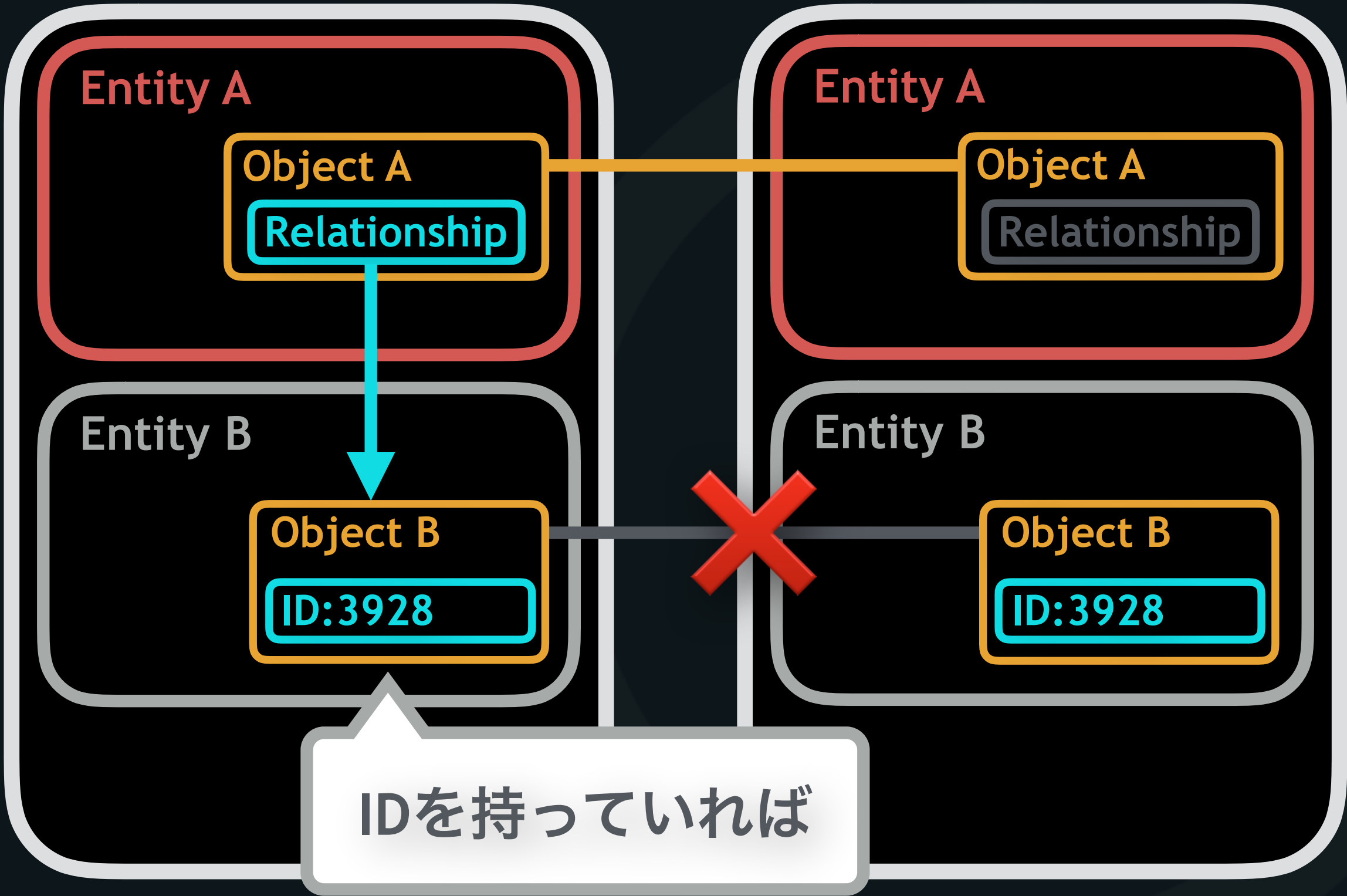
マイグレーション分割の改善

マイグレーション分割の改善 1

オブジェクトに
一意の値をもたせる

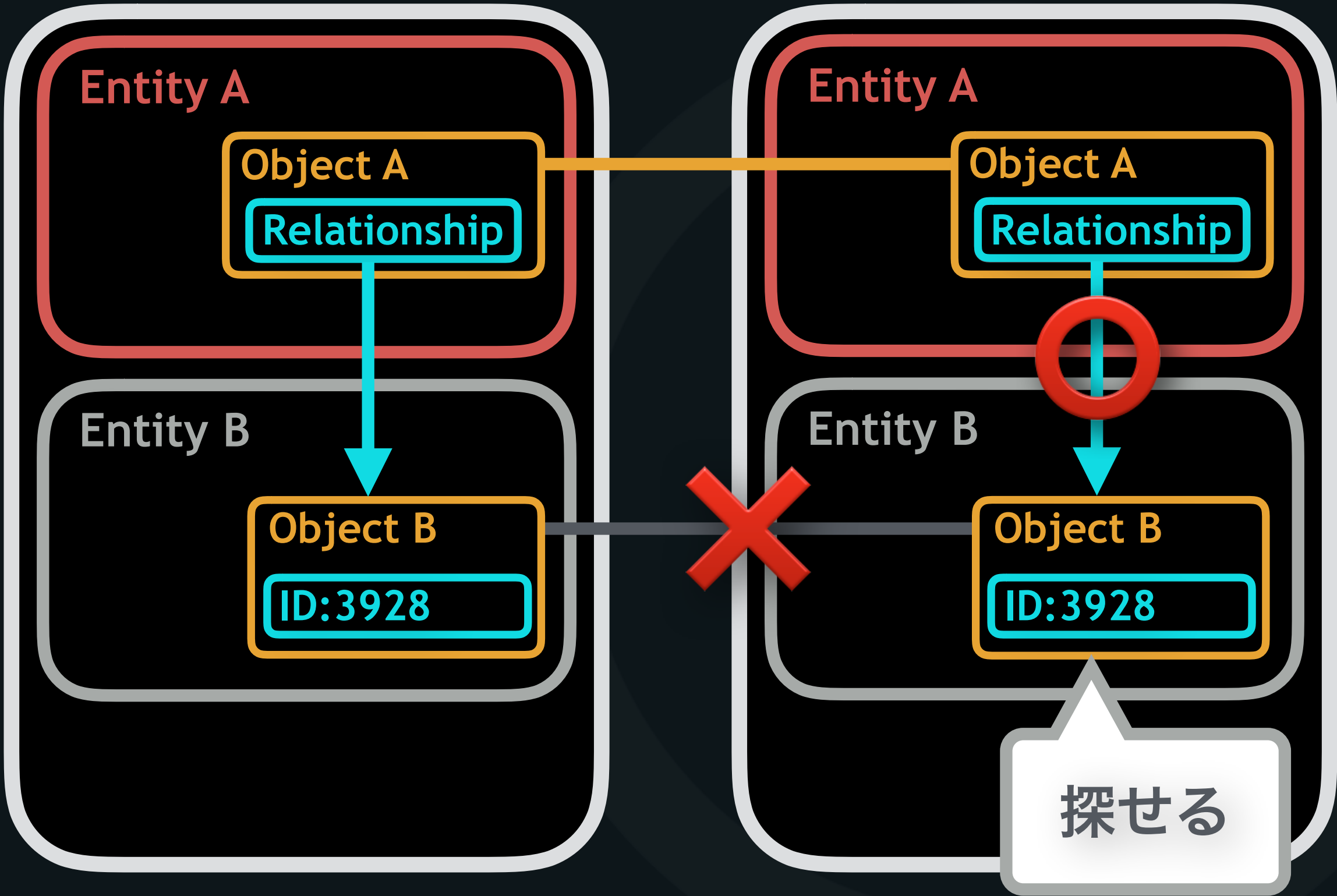
Source
Context

Destination
Context



Source Context

Destination Context



マイグレーション分割の改善 2

グループ分けできる値を持たせて
1つのエンティティ内で分割する

※ データの多いエンティティの場合

マイグレーション分割の改善

例えば、UUIDを持たせておく

3F2504E0-4F89-11D3-9A0C-0305E82C3301

↑ 頭1文字目で16分割できて、他とかぶらない

マイグレーション分割の改善 3

Mapping Modelを
コードで分ける

Mapping Model

ENTITY MAPPINGS

EntityAToEntityA

EntityBToEntityB

Attribute Mappings

Destination Attribute	Value Expression
dummy	Value Expression
group	\$source.group
name	\$source.name

+ -

Relationship Mappings

Destination Relationship	Value Expression
relationship	FUNCTION(\$manager, "destinationInstancesForEntityMa...

+ -

+ Add Entity Mapping

Model 1

Model 2

SourceDestination

Entity Mapping

Mapping NameEntityAToEntityA

SourceEntityA

DestinationEntityA

TypeCopy

Custom PolicyClass Name

Source FetchDefault

Filter PredicateFilter Predicate

User Info

Key	Value
-----	-------

+ -

MappingModelをコードで分ける

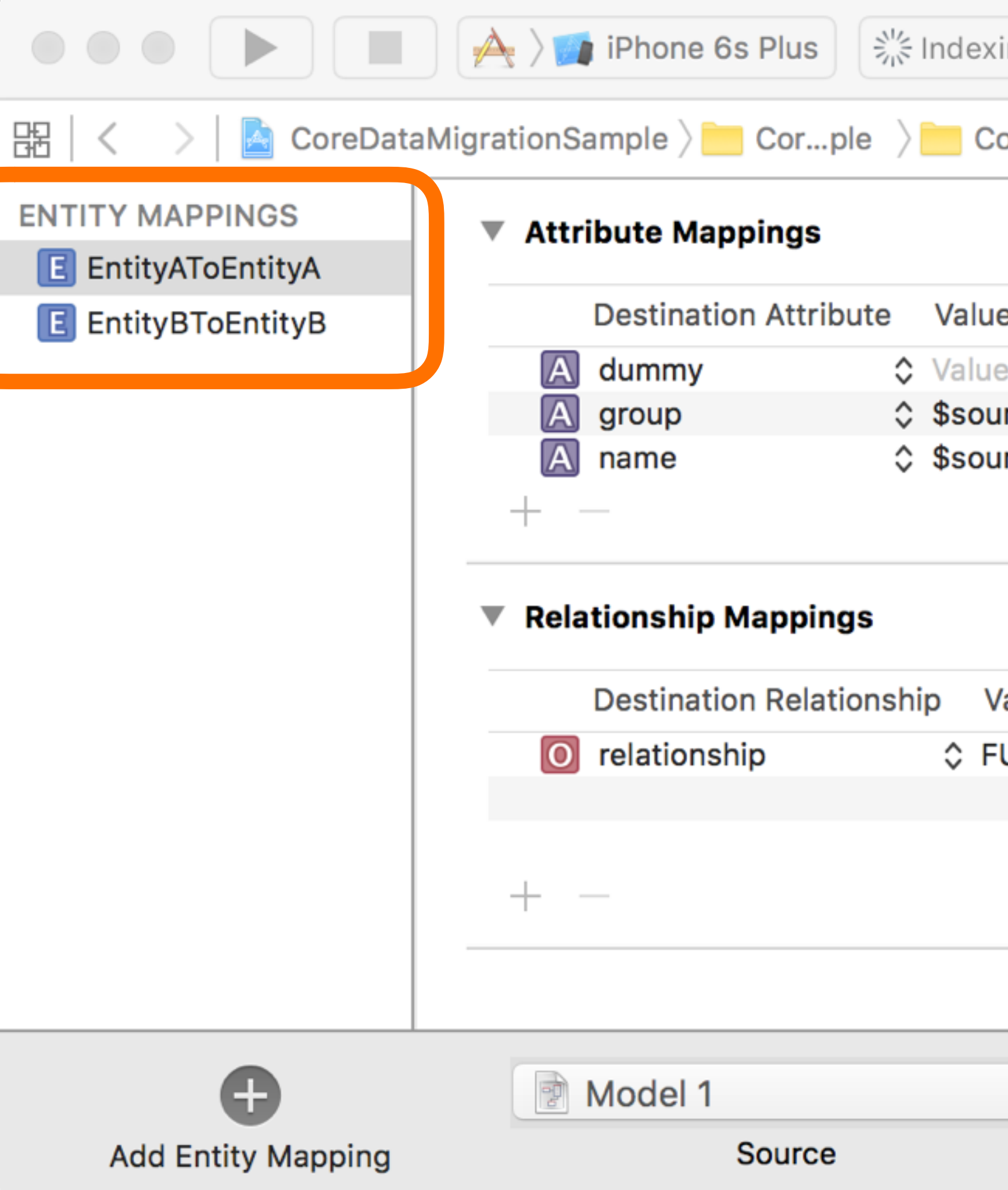
元となるMapping Modelを
一つだけ作っておく

MappingModelをコードで分ける

分割する数だけコード上で
Mapping Modelをコピー

MappingModelをコードで分ける

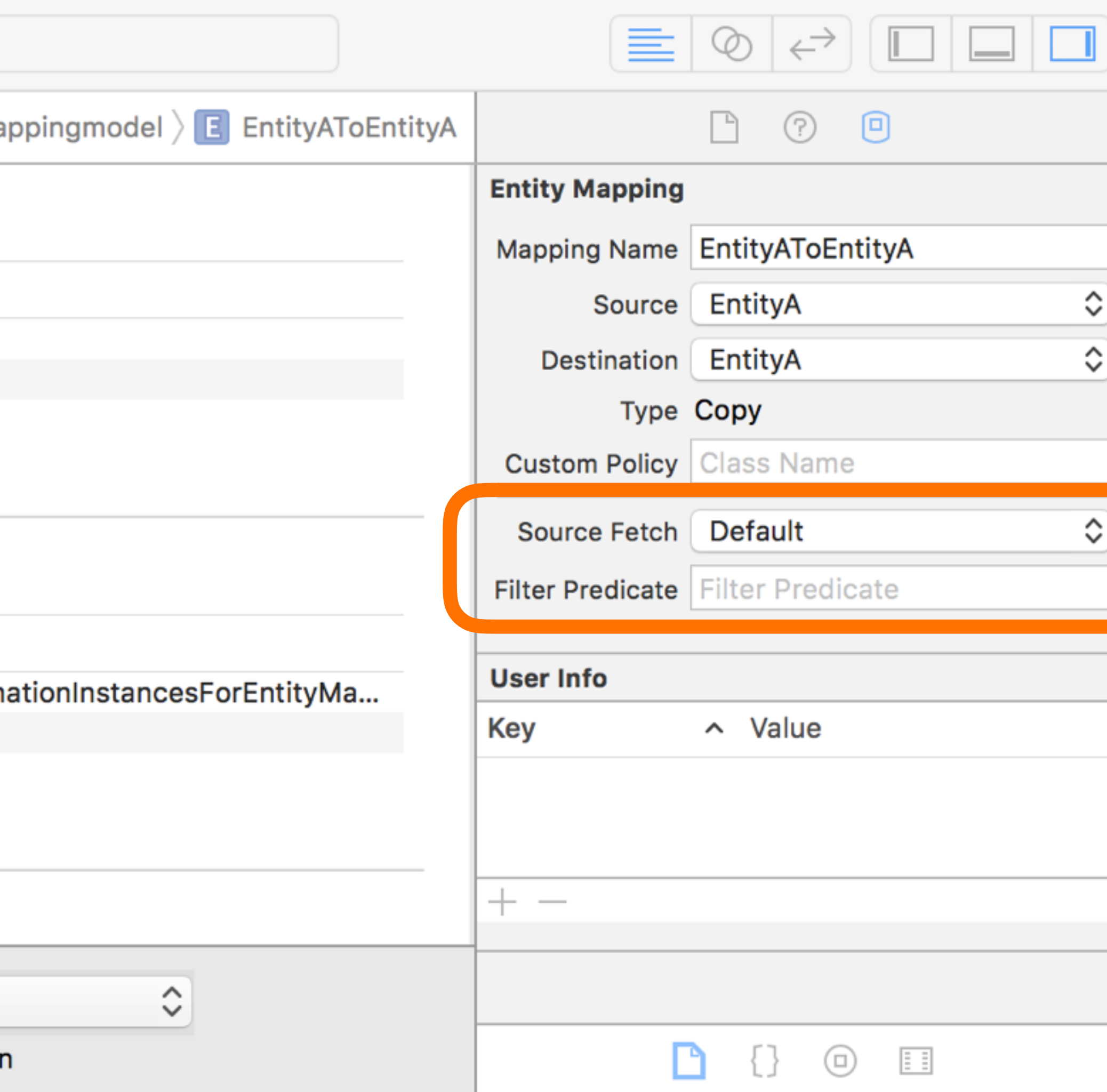
Mapping Model毎に
必要なエンティティだけ残す



MappingModelをコードで分ける

エンティティを更に分けるなら
Filter Predicateを差し替える

sourceExpressionに
NSFetchRequestExpressionをセット



DEMO



マイグレーション分割のポイント

- メモリと時間はトレードオフ
- エンティティ毎のデータ量によって
分け方を考える
- 処理の順番を決めて無駄な関連付けをしない
- 必要の無いデータは引き継がない