intra-mart WebPlatform/AppFramework Ver.7.2

アプリケーション共通マスタ API ガイドライン

2010/04/01 初版

＜＜　変更履歴　＞＞

変更年月日	変更内容
2010/04/01	初版

<< 目次 >>

1 はじめに

1.1 背景

永続化させたいデータは必要に応じて検索、登録、更新および削除される。最も簡単な方法は要求分析によって抽出されたモデルをデータベースのテーブルとして表現することである。モデルの内容へのアクセスはデータベース内のそれぞれのテーブルに対して SQL を発行すればよい。しかし、実際の業務では次のような要求があがる場合が多い。

- あるデータの変更前の情報が欲しい。
  - 過去の伝票を発行した時点での商品の単価を見たい
  - ユーザのある時点における所属部署を知りたい、等
- ある時点から有効になるデータを前もって登録しておきたい。
  - 新しい組織構成の情報を組織変更前にあらかじめ登録しておきたい
  - 入社予定の社員をあらかじめ登録しておきたい、等
- 同一のデータを複数の言語で登録または表示をしたい。
  - 部門一覧を、閲覧するユーザの母国語で表示したい
  - 各国語向けに商品名を登録したい、等
- データベース上のテーブル名またはカラム名を独自のものに変更したい。
  - カラム名が、使用するデータベースの予約語の制限を受けている
  - テーブル名が既に別のアプリケーションで使用されている、等
- 参照先のデータが削除された場合の対処をしたい。
  - ある組織が削除された場合、ユーザの所属情報も同時に削除したい
  - ある商品が伝票で参照されている場合、その商品は削除できないようにしたい、等

これらの要求はアプリケーションの種類によらない共通的なものとしてあげられる。そのため何らかの共通的な仕組が必要となる。

1.2 目的

本書では次のようなことを目的とする。

- 地域や言語によって表現が異なるデータおよび時間によって内容が変化するデータに対してその整合性を保持しながらアクセスする仕組みを提供する。
- データストアの実装方法に依存しないアクセス方法を提供する。

1.3 本書で扱う範囲

本書はアプリケーション共通マスタの基盤となる技術の仕様について述べている。各種の制限事項や動作環境等についてはこれらに依存する。

本書はアプリケーション共通マスタのインタフェースや外部仕様について説明している。また、コンポーネントの使われ方やその手順などの内部仕様についても一部提示しているが、内部の実装などについては触れていない。

本書では説明を補強するために複数のサンプルを提示している。これらは各節や章における仕様を理解することを第一の目的としているものであり、プログラムやシステムとしての最適解を示しているものではない。そのため、一部の例外処理などは意図的に実装していないものもある。

本書では読者が上流工程に関係する場合でも有用な情報が含まれているが、プロジェクト管理や業務分析などの詳細な方法については触れていない。

本書では JavaTM 2 Standard Edition (J2SE)や JavaTM 2 Enterprise Edition (J2EE)の詳細については触れていない。

本書では ER 図の書式やデータベースの正規化の詳細については触れていない。

1.4 本書の対象読者

本書は以下の読者を対象としている。

- 設計者に対してはどのようなインタフェース仕様にするべきか、コンポーネントをどのように分割するべきかを提示している。
- 実装者に対してはコンポーネントを作成するための情報やその仕様について提示している。

プロジェクト管理者や業務分析者などにとっては、本書は特に明示的な情報を提示していない。具体的な分析、管理を行う場合の参考にとどまる。

また、以下の技術や手法に対してある程度の理解があることが望ましい。

- JavaTM 2 Standard Edition 1.4[1]
- JavaTM 2 Enterprise Edition 1.3[2]
- UML
- デザインパターン
- XML 1.1[3]
- ER 図
- データベースの正規化

1.5 構成

本書は以下のような構成をしている。

- 「2 データ構造」では言語や時間の概念を組み込んだ場合のデータの考え方と、そのデータ構造を表現するために拡張されたER図について説明している。
- 「3 API」では言語や時間の概念を組み込んだデータに対してアクセスするための標準的なAPIとその動作原理について説明している。
- 「4 開発者の役割」ではデータ構造をもとに開発者が実際に行うべき作業について説明している。

2 データ構造

ER 図はデータ構造を表現する有効な手段であるが、国際化や期間化といった概念については別に考える必要がある。しかし、国際化や期間化を考慮したER図を描こうとすると非常に煩雑になりやすい。また、国際化や期間化の考え方は共通であるため、同様なパターンがER図の中に何度も現れるためやや冗長となる。

ここでは国際化や期間化を考慮したデータの構造の説明とその内容を表現するための表記法を記す。ここで使用する表記法は従来のER図を独自に拡張したものである。

2.1 エンティティ

エンティティはデータの構造を表すものである。エンティティで定義された構造をもつ実際のデータを本書ではインスタンスと呼ぶ。1つのエンティティに対して複数のインスタンスが存在する。

エンティティは国際化可能、または期間化可能であるかどうかを指定することができる。

エンティティは複数の属性を持つ。属性はエンティティの構造を満たすデータの詳細内容である。属性はプライマリキーと従属属性のいずれかに分類される。プライマリキーはエンティティにおけるインスタンスを一意に識別することができるものである。

2.1.1 国際化

従属属性はそれぞれ国際化の対象であるかどうかを決めることができる。ある従属属性が「国際化の対象である」ということは、その属性を複数の言語で登録することが可能であり、国際化対応したアプリケーションでは必要に応じて表示内容を切り替えることができることを意味する。国際化の対象である属性は、たとえば日本語環境と英語環境のいずれでもそれぞれの言語で表示することが可能となる。ある従属属性が国際化の対象である場合、そのエンティティは国際化可能でなければならない。

2.1.2 期間化

従属属性はそれぞれ期間化の対象であるかどうかを決めることができる。ある従属属性が「期間化の対象である」ということは、その属性は日時によって異なる可能性があり、期間化対応したアプリケーションでは指定された時点におけるデータを表示することができることを意味する。期間化の対象である属性は、たとえば名称などが変更された場合でも過去の時点におけるデータを表示したり、既存のデータを修正することなく変更される内容をあらかじめ登録しておくことが可能となる。ある従属属性が期間化の対象である場合、そのエンティティは期間化可能でなければならない。

期間として扱うことができる日時は以下の範囲に収まる必要がある。

1582/10/15 00:00:00 ≦ 期間として扱うことができる日時 ＜ 9999/12/31 23:59:59

2.1.3 エンティティの拡張

エンティティには独自の従属属性を追加することも可能である。この追加された従属属性もそれぞれ国際化または期間化の対象であるかどうかを指定できる。

2.1.4 エンティティの表記法

本書ではエンティティの表記を「図 2-1 エンティティの表記」のようにしている。

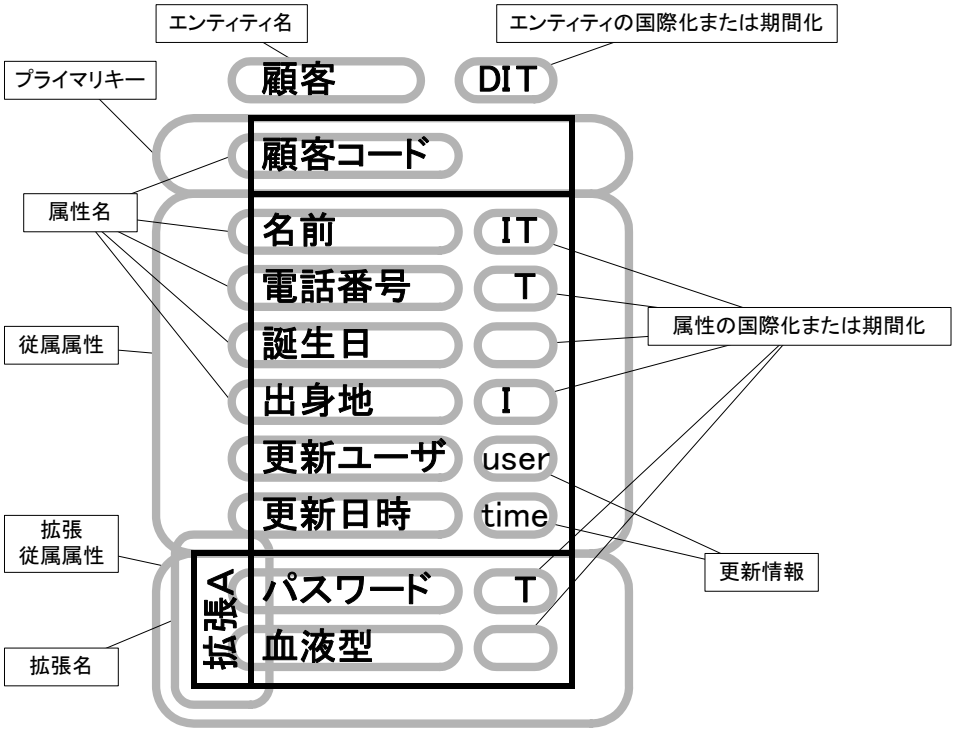


図 2-1 エンティティの表記

「図 2-1 エンティティの表記」におけるそれぞれの内容を以下に示す。

- エンティティ名
  エンティティを一意に識別する名前を定義する。実際には英数字およびアンダースコア("_")の組合せでなければならないが、本書では便宜上日本語で記述する場合もある。
- 属性名
  エンティティに含まれる属性の名前を定義する。属性名は同一のエンティティ内で一意である必要がある。属性はプライマリキー、従属属性または拡張従属属性のいずれかに分類される。実際には英数字およびアンダースコア("_")の組合せでなければならないが、本書では便宜上日本語で記述する場合もある。
- エンティティの国際化または期間化
  エンティティ名と同一行の右端にこのエンティティが国際化または期間化が可能であるかが定義されている。「I」は国際化のみ可能な属性を含むことが可能であることを、「T」は期間化のみできる属性を含むことが可能であることを、「D」は期間化かつ国際化できる属性を含むことが可能であることを示す。
  - 何も書かれていない場合
    このエンティティでは国際化または期間化のいずれも不可能であることを意味する。
  - 「I」と書かれている場合
    このエンティティでは国際化は可能であるが期間化は不可能であることを意味する。
  - 「T」と書かれている場合
    このエンティティでは国際化は不可能であるが期間化は可能であることを意味する。
  - 「D」と書かれている場合
    このエンティティは国際化かつ期間化された従属属性から構成されていることを意味する。
  - 「IT」と書かれている場合
    このエンティティでは国際化または期間化のいずれも可能であることを意味する。ただし、国際化と期間化が同時に含まれる従属属性は存在しない。
  - 「DT」と書かれている場合
    このエンティティでは国際化または期間化のいずれも可能であることを意味する。ただし、国際化のみ可能である従属属性は存在しない。

- ◆ 「DIT」と書かれている場合
  このエンティティでは国際化または期間化のいずれも可能であることを意味する。
- ◆ 「DI」という表記は存在しない
  期間化かつ国際化が可能である属性を含むことができる場合、期間化のみ可能である属性を含むことが可能でなくてはならない。

■ 属性の国際化または期間化
属性名と同一行の右端にその属性が国際化または期間化の対象であるかが定義されている。

- ◆ 何も書かれていない場合
  この属性は国際化または期間化のいずれの対象でもないことを意味する。
- ◆ 「IT」と書かれている場合
  この属性は国際化かつ期間化の対象であることを意味する。この場合、エンティティは国際化かつ期間化されていなければならない(エンティティ名の右端に「D」が含まれている)。
- ◆ 「I」と書かれている場合
  この属性は国際化の対象であることを意味する。この場合、エンティティは国際化可能でなければならない(エンティティ名の右端に「I」が含まれている)。
- ◆ 「T」と書かれている場合
  この属性は期間化の対象であることを意味する。この場合、エンティティは期間化可能でなければならない(エンティティ名の右端に「T」が含まれている)。

■ 更新情報
新規登録または更新時に、データの登録および更新をしたユーザとそのタイムスタンプを保存する属性であることを意味する。これらの設定は任意であるが、片方のみ定義することはできない。

- ◆ 「user」と書かれている場合
  この属性はデータの登録および更新をしたユーザが保存される属性であることを意味する。この指定がされた属性の型はStringである必要がある。属性の型については「2.1.5 属性の型」を参照すること。
- ◆ 「time」と書かれている場合
  この属性はデータの登録および更新をした時点のタイムスタンプが保存される属性であることを意味する。この指定がされた属性の型はDateである必要がある。属性の型については「2.1.5 属性の型」を参照すること。

■ 拡張名
独自の従属属性をエンティティに追加する場合、従属属性のグループを識別するための名前を定義する。1つのエンティティに複数の従属属性グループを追加することができる。拡張名は同一のエンティティ内で一意である必要がある。実際には英数字およびアンダースコア("_")の組合せでなければならないが、本書では便宜上日本語で記述する場合もある。

2.1.5 属性の型

エンティティの属性に指定できる型を「表 2-1 属性の型」に示す。

表 2-1 属性の型

型	意味
String	文字列
Date	日時
Locale	言語
Decimal	整数
Float	浮動小数

「表 2-1 属性の型」はエンティティにおける概念的な型であり、リレーショナルデータベースで扱う型ともJava言語で扱う型とも異なる。

2.1.6 属性スコープ

属性はそれぞれ国際化または期間化されているかどうかによって以下の種類に分類できる。

- 国際化も期間化もされていない
- 国際化のみされている
- 期間化のみされている
- 国際化も期間化もされている

本書ではこの分類を属性スコープと呼んでいる。ある 2 つの属性スコープが一致しているとは、2 つの属性の国際化および期間化の可否が完全に一致していることを意味する。

本書では属性スコープを「表 2-2 属性スコープ名」に示すように分類する。

表 2-2 属性スコープ名

属性スコープ名	内容
基本属性スコープ	属性は国際化も期間化もされていない
国際化属性スコープ	属性は国際化のみされている
期間化属性スコープ	属性は期間化飲みされている
期間国際化属性スコープ	属性は国際化かつ期間化されている

2.2 リレーションシップ

リレーションシップはエンティティ間に存在する関連を指定するものである。

リレーションシップは必ず以下に示される 2 つのエンティティの間に定義される。

- ソースエンティティ
  参照元のエンティティ。ターゲットエンティティを参照する。
- ターゲットエンティティ
  参照先のエンティティ。ソースエンティティから参照される。

ソースエンティティとターゲットエンティティは同一のエンティティであってもよい。

リレーションシップは次のような制約を持たせることが可能である。

- 外部キー制約（必須）
- 国際化制約（任意）
- 期間化制約（任意）
- ライフタイム制約（任意）
- ライフタイム期間（任意）

2.2.1 リレーションシップの表記法

本書ではリレーションシップの表記を「図 2-2 リレーションシップの表記」のようにしている。

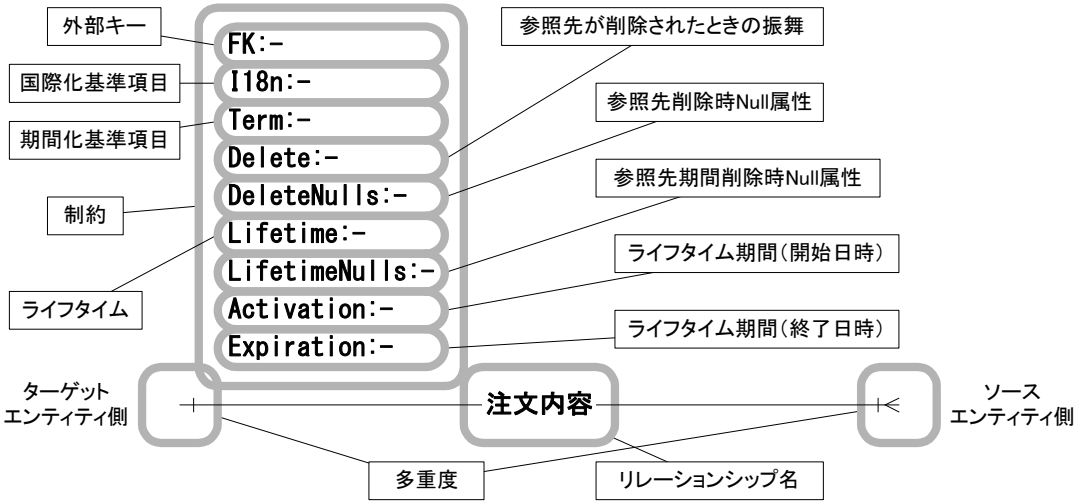


図 2-2 リレーションシップの表記

「図 2-2 リレーションシップの表記」におけるそれぞれの内容を以下に示す。

- リレーションシップ名
  リレーションシップを一意に識別する名前を定義する。実際には英数字およびアンダースコア("_")の組合せでなければならないが、本書では便宜上日本語で記述する場合もある。
- 制約
  - 外部キー(FK)
    参照先を決定するための属性を表す。詳細は「2.2.5.1 外部キー制約」を参照すること。
  - 国際化基準項目(I18n)
    参照先のどの言語(ロケール)のデータを参照するかを決定する属性を表す。詳細は「2.2.5.2 国際化制約」を参照すること。
  - 期間化基準項目(Term)
    参照先のどの時点のデータを参照するかを決定する属性を表す。詳細は「2.2.5.3 期間化制約」を参照すること。
  - 参照先削除時処理(Delete)
    参照先が削除された場合の処理を表す。詳細は「2.2.6 参照先削除時における整合性の維持」を参照すること。
  - 参照先削除時Null属性(DeleteNulls)
    参照先のインスタンスが削除されたときにnullにする属性を表す。DeleteにNullが指定された場合のみ指定できる。詳細は「2.2.6.2 参照元の外部キー情報をnullに設定」を参照すること。
  - ライフタイム(Lifetime)
    参照先の期間が一部削除された場合の処理を表す。詳細は「2.2.5.4 ライフタイム制約」を参照すること。
  - 参照先期間削除時Null属性(LifetimeNulls)
    参照先の期間が一部削除されたときにnullにする属性を表す。LifetimeにNullが指定された場合のみ指定できる。詳細は「2.2.7.2 参照元の外部キー情報をnullに設定」を参照すること。
  - ライフタイム期間(ActivationおよびExpiration)
    詳細は「2.2.5.5 ライフタイム期間」を参照すること。
- 多重度

2.2.2 外部キー

外部キーはソースエンティティがターゲットエンティティのどのインスタンスを参照するかを決定するものである。外

部キーはターゲットエンティティのプライマリキーと同じ構造を持たなければならないが、属性名は一致させる必要はない。ターゲットエンティティのプライマリキーが複数の属性から構成される場合、外部キーの指定順も同じ順番である必要がある。

1 つのリレーションシップにおいて複数の属性で外部キーとする場合、以下のいずれかの方法で指定されていなければならない。

- 外部キーとなるすべての属性が基本属性スコープの属性である。
- 外部キーとなるすべての属性が国際化属性スコープの属性である。
- 外部キーとなるすべての属性が期間化属性スコープの属性である。
- 外部キーとなるすべての属性が期間国際化属性スコープの属性である。
- 外部キーとなる属性のうちいくつかが国際化属性スコープの属性であり、残りのものは基本属性スコープの属性である。
- 外部キーとなる属性のうちいくつかが期間化属性スコープの属性であり、残りのものは基本属性スコープの属性である。
- 外部キーとなる属性のうちいくつかが期間国際化属性スコープの属性であり、残りのものは基本属性スコープまたは期間化属性スコープの属性である。

拡張従属属性は外部キーとして指定することはできない。

2.2.2.1 外部キースコープ

外部キーはさまざまな属性スコープに含まれる属性から構成される。ある外部キーにおいて属性スコープが存在する範囲を、本書では外部キースコープと呼ぶ。

外部キースコープに含まれる属性スコープのうち、もっとも詳細な情報を持つ属性スコープに含まれる属性を外部キースコープの最下層の属性と呼んでいる。

外部キースコープ、外部キーに含まれる属性スコープ、外部キースコープの最下層の属性を「表 2-3 外部キースコープ一覧」にまとめる。

表 2-3 外部キースコープ一覧

外部キーに含まれる属性スコープ	外部キースコープに含まれる属性スコープ	最下層の属性スコープ
基本属性スコープ	基本属性スコープ	基本属性スコープ
国際化属性スコープ	基本属性スコープ 国際化属性スコープ	国際化属性スコープ
期間化属性スコープ	基本属性スコープ 期間化属性スコープ	期間化属性スコープ
期間国際化属性スコープ	基本属性スコープ 期間化属性スコープ 期間国際化属性スコープ	期間国際化属性スコープ
基本属性スコープ 国際化属性スコープ	基本属性スコープ 国際化属性スコープ	国際化属性スコープ
基本属性スコープ 期間化属性スコープ	基本属性スコープ 期間化属性スコープ	期間化属性スコープ
基本属性スコープ 期間国際化属性スコープ	基本属性スコープ 期間化属性スコープ 期間国際化属性スコープ	期間国際化属性スコープ
基本属性スコープ 期間化属性スコープ 期間国際化属性スコープ	基本属性スコープ 期間化属性スコープ 期間国際化属性スコープ	期間国際化属性スコープ
期間化属性スコープ	基本属性スコープ	期間国際化属性スコープ

期間国際化属性スコープ	期間化属性スコープ 期間国際化属性スコープ	

ある 2 つの外部キースコープが一致しているということは、外部キースコープの最下層の属性スコープが一致しているということである。この場合、外部キーが完全に同一の属性スコープを含んでいなくてもよい。

2.2.3 国際化基準項目

ターゲットエンティティのインスタンスの中でどの言語（ロケール）のデータを対象の参照先とするのかを決定したい場合、ソースエンティティから決定される属性の 1 つを基準にする。この基準となる属性を本書では国際化基準項目と呼ぶ。

国際化基準項目には以下の 2 種類がある。

- 外部キーと同一エンティティ内で指定する方法
- 他のエンティティで指定する方法

国際化基準項目は値に null が設定可能なものでなければならない。国際化基準項目に null が設定されている場合、対象とする言語（ロケール）を指定していないものとみなす。

国際化基準項目に指定された属性の型は Locale でなければならない。

拡張従属属性は国際化基準項目として指定することはできない。

2.2.3.1 外部キーと同一エンティティ内で国際化基準項目を指定する方法

外部キーが定義されているエンティティと同一のエンティティ上に国際化基準項目を定義する場合、外部キースコープの最下層の属性スコープに存在する属性だけが国際化基準項目として指定できる。

- 外部キーが基本属性スコープの属性のみで構成されている場合、基本属性スコープの属性のみが国際化基準項目として指定できる。
- 外部キーに国際化属性スコープの属性が含まれている場合、国際化属性スコープの属性のみが国際化基準項目として指定できる。
- 外部キーに期間化属性スコープの属性が含まれているが期間国際化属性スコープの属性が含まれていない場合、期間化属性スコープの属性のみが国際化基準項目として指定できる。
- 外部キーに期間国際化属性スコープの属性が含まれている場合、期間国際化属性スコープの属性のみが国際化基準項目として指定できる。

2.2.3.2 他のエンティティで国際化基準項目を指定する方法

外部キーが定義されているエンティティと異なるエンティティ上に国際化基準項目を定義する場合、該当するエンティティまでのリレーションシップのルートを指定する。この場合、その間のリレーションシップはすべてソースエンティティからターゲットエンティティへ向かう方向にしかたどれない。

また、これらのリレーションシップにおいて国際化制約および期間化制約およびライフタイム制約が指定されている場合、それらの決定項目（国際化基準項目および期間化基準項目）はそのリレーションシップに関連付けられているソースエンティティとターゲットエンティティ内でのみ解決できるものでなければならない。つまり、このようなリレーションシップでは国際化基準項目や期間化基準項目はソースエンティティ内で定義されていなければならない。

国際化基準項目が定義されるエンティティと外部キーが定義されているエンティティが異なる場合、国際化基準項目が定義される参照先までの間のリレーションシップとエンティティは以下のような条件を満たす必要がある。

- ソースエンティティから見た外部エンティティへのリレーションシップに含まれる外部キースコープの最下層の属性スコープは、ソースエンティティから見たターゲットエンティティへのリレーションシップに含まれる外部キースコープの最下層の属性スコープと一致していなければならない。
- 国際化基準項目が定義されているエンティティへのリレーションシップの場合（そこから先に別のリレーションシップをたどる必要がない場合）、国際化基準項目となる属性が一意に決定できる制約がリレーションシップに指定されている必要がある。
  - 参照先のエンティティにおいて基本属性スコープの属性は常に国際化基準項目として指定できる。
  - 参照先のエンティティにおいて国際化属性スコープの属性を国際化基準項目として指定する場合、リレーションシップには国際化基準項目が指定されている必要がある。
  - 参照先のエンティティにおいて期間化属性スコープの属性を国際化基準項目として指定する場合、リレーションシップには期間化基準項目が指定されている必要がある。
  - 参照先のエンティティにおいて期間国際化属性スコープの属性を国際化基準項目として指定する場合、リレーションシップには国際化基準項目と期間化基準項目の両方が指定されている必要がある。
- 国際化基準項目が定義されていないエンティティへのリレーションシップの場合（そこから先に別のリレーションシップをたどる必要がある場合）、そこから先のエンティティを検索するための外部キーは以下のいずれかの条件を満たす必要がある。
  - 参照先のエンティティにおいて基本属性スコープの属性は常に外部キーとして指定できる。
  - 参照先のエンティティにおいて国際化属性スコープの属性が外部キーとして指定されている場合、リレーションシップには国際化基準項目が指定されている必要がある。
  - 参照先のエンティティにおいて期間化属性スコープの属性が外部キーとして指定されている場合、リレーションシップには期間化基準項目が指定されている必要がある。
  - 参照先のエンティティにおいて期間国際化属性スコープの属性が外部キーとして指定されている場合、リレーションシップには国際化基準項目と期間化基準項目の両方が指定されている必要がある。

2.2.4 期間化基準項目

ターゲットエンティティのインスタンスの中でいつの時点のデータを対象の参照先とするのかを決定したい場合、ソースエンティティの属性の 1 つを基準にする。この基準となる属性を本書では期間化基準項目と呼ぶ。

期間化基準項目には以下の 2 種類がある。

- 外部キーと同一エンティティ内で指定する方法
- 他のエンティティで指定する方法

期間化基準項目は値に null が設定可能なものでなければならない。期間化基準項目に null が設定されている場合、対象とする日時を指定していないものとみなす。

期間化基準項目に指定された属性の型は Date でなければならない。

拡張従属属性は期間化基準項目として指定することはできない。

2.2.4.1 外部キーと同一エンティティ内で期間化基準項目を指定する方法

外部キーが定義されているエンティティと同一のエンティティ上に期間化基準項目を定義する場合、外部キースコープの最下層の属性スコープに存在する属性だけが期間化基準項目として指定できる。

- 外部キーが基本属性スコープの属性のみで構成されている場合、基本属性スコープの属性のみが期間化基準項目として指定できる。
- 外部キーに国際化属性スコープの属性が含まれている場合、国際化属性スコープの属性のみが期間化基準項目として指定できる。
- 外部キーに期間化属性スコープの属性が含まれているが期間国際化属性スコープの属性が含まれてい

ない場合、期間化属性スコープの属性のみが期間化基準項目として指定できる。

- 外部キーに期間国際化属性スコープの属性が含まれている場合、期間国際化属性スコープの属性のみが期間化基準項目として指定できる。

2.2.4.2 他のエンティティで期間化基準項目を指定する方法

外部キーが定義されているエンティティと異なるエンティティ上に期間化基準項目を定義する場合、該当するエンティティまでのリレーションシップのルートを指定する。この場合、その間のリレーションシップはすべてソースエンティティからターゲットエンティティへ向かう方向にしかたどれない。

また、これらのリレーションシップにおいて国際化制約および期間化制約が指定されている場合、それらの決定項目（国際化基準項目および期間化基準項目）はそのリレーションシップに関連付けられているソースエンティティとターゲットエンティティ内でのみ解決できるものでなければならない。つまり、このようなリレーションシップでは国際化基準項目や期間化基準項目はソースエンティティ内で定義されていなければならない。

期間化基準項目が定義されるエンティティと外部キーが定義されているエンティティが異なる場合、期間化基準項目が定義される参照先までの間のリレーションシップとエンティティは以下のような条件を満たす必要がある。

- ソースエンティティから見た外部エンティティへのリレーションシップに含まれる外部キースコープの最下層の属性スコープは、ソースエンティティから見たターゲットエンティティへのリレーションシップに含まれる外部キースコープの最下層の属性スコープと一致していなければならない。
- 期間化基準項目が定義されているエンティティへのリレーションシップの場合（そこから先に別のリレーションシップをたどる必要がない場合）、期間化基準項目となる属性が一意に決定できる制約がリレーションシップに指定されている必要がある。
  - 参照先のエンティティにおいて基本属性スコープの属性は常に期間化基準項目として指定できる。
  - 参照先のエンティティにおいて国際化属性スコープの属性を期間化基準項目として指定する場合、リレーションシップには国際化基準項目が指定されている必要がある。
  - 参照先のエンティティにおいて期間化属性スコープの属性を期間化基準項目として指定する場合、リレーションシップには期間化基準項目が指定されている必要がある。
  - 参照先のエンティティにおいて期間国際化属性スコープの属性を期間化基準項目として指定する場合、リレーションシップには国際化基準項目と期間化基準項目の両方が指定されている必要がある。
- 期間化基準項目が定義されていないエンティティへのリレーションシップの場合（そこから先に別のリレーションシップをたどる必要がある場合）、そこから先のエンティティを検索するための外部キーは以下のいずれかの条件を満たす必要がある。
  - 参照先のエンティティにおいて基本属性スコープの属性は常に外部キーとして指定できる。
  - 参照先のエンティティにおいて国際化属性スコープの属性が外部キーとして指定されている場合、リレーションシップには国際化基準項目が指定されている必要がある。
  - 参照先のエンティティにおいて期間化属性スコープの属性が外部キーとして指定されている場合、リレーションシップには期間化基準項目が指定されている必要がある。
  - 参照先のエンティティにおいて期間国際化属性スコープの属性が外部キーとして指定されている場合、リレーションシップには国際化基準項目と期間化基準項目の両方が指定されている必要がある。

2.2.5 リレーションシップの制約

エンティティの属性にはさまざまな値が入り、その内容は比較的自由である。しかしリレーションシップが定義されている場合、エンティティの属性や期間などは整合性を保つためにさまざまな制限を受ける。

2.2.5.1 外部キー制約

ソースエンティティのインスタンスの外部キーに値がすべて入る場合、外部キーによって決定されるターゲットエンティティのインスタンスが必ず存在する必要がある。

2.2.5.2 国際化制約

ターゲットエンティティが国際化されている場合、どの言語（ロケール）を対象の参照先とするのかを決定するためには国際化基準項目を指定する必要がある。参照先の言語（ロケール）を特に決定する必要がない場合は何も指定しない。

「2.2.5.3 期間化制約」で示す期間化基準項目も同時に指定する場合、両者とも同一エンティティ内の項目であり、かつ国際化と期間化の対象となる属性スコープが一致している必要がある [1]。

以下の条件をすべて満たす場合のみこの制約は指定できる。

- ターゲットエンティティが国際化されている。
- ライフタイム制約（「2.2.5.4 ライフタイム制約」を参照）が指定されていない。

2.2.5.3 期間化制約

ターゲットエンティティが期間化されている場合、いつの時点のデータを参照先とするのかを決定するためには期間化基準項目を指定する必要がある。どの時点のデータであるかを特に決定する必要がない場合は何も指定しない。

「2.2.5.3 期間化制約」で示す期間化基準項目も同時に指定する場合、両者とも同一エンティティ内の項目であり、かつ国際化と期間化の対象となる属性スコープが一致している必要がある [2]。

以下の条件をすべて満たす場合のみこの制約は指定できる。

- ターゲットエンティティが期間化されている。
- ライフタイム制約（「2.2.5.4 ライフタイム制約」を参照）が指定されていない。

2.2.5.4 ライフタイム制約

ソースエンティティのインスタンスとターゲットエンティティのインスタンスの間における期間の関連付けを指定する。両者の期間が関連付けられている場合、ソースエンティティで管理することができる期間は、ターゲットエンティティのインスタンスで定義されている期間内でなければならない。

以下の条件をすべて満たす場合のみ、この制約は指定できる。

- ソースエンティティとライフタイム期間（「2.2.5.5 ライフタイム期間」を参照）の関係について以下のいずれかの条件が満たされている。
  - ソースエンティティは期間化されているが、ライフタイム期間は指定されていない（この場合、ソースエンティティのインスタンスで管理することができる期間は、ターゲットエンティティのインスタンスで定義されている期間内でなければならない）。
  - ソースエンティティは期間化されていないが、ライフタイム期間は指定されている（この場合、ライフタイム期間で指定される期間は、ターゲットエンティティのインスタンスで定義されている期間内でなければならない）。
- 外部キー（「2.2.5.1 外部キー制約」を参照）について以下のいずれかの条件が満たされている。
  - 外部キーはソースエンティティのプライマリキーの一部である。
  - 外部キーの一部またはすべてが期間化されている属性である。
- 外部キーの一部またはすべては国際化または期間国際化されていない。
- ターゲットエンティティが期間化されている。
- 国際化制約（「2.2.5.2 国際化制約」を参照）が指定されていない。

---

[1] たとえば、期間化基準項目は期間化も国際化もされていないが、国際化項目は国際化の対象である、といった指定はできない。

[2] たとえば、期間化基準項目は期間化も国際化もされていないが、国際化項目は国際化の対象である、といった指定はできない。

- 期間化制約(「2.2.5.3 期間化制約」を参照)が指定されていない。

2.2.5.5 ライフタイム期間

ライフタイム制約を指定するとき、ソースエンティティが期間化されていない場合はライフタイム期間を指定する必要がある。この場合、ソースエンティティのインスタンスの期間となる属性を明示的に指定する。この属性は以下の 2 種類を指定する。

- Activation
  有効期間の開始日時に該当する属性名を指定する。ここで指定された属性の型は Date にしなければならない。
- Expiration
  有効期間の終了日時に該当する属性名を指定する。ここで指定された属性の型は Date にしなければならない。

ある日時が以下の範囲内である場合、その日時は有効期間内であると判断される。

Activation で決定される日時 ≦ 調査対象の日時 ＜ Expiration で決定される日時

ライフタイム期間で指定される範囲は、ターゲットエンティティのインスタンスで定義されている期間内でなければならない。

有効開始日時と有効終了日時は同一のエンティティに定義されている必要があり、属性スコープも同じでなければならない[3]。

有効開始日時と有効終了日時は値に null が設定不可能なものでなければならない。

ライフタイム期間を指定する方法は以下の 2 種類がある。

- 外部キーと同一エンティティ内で指定する方法
- 他のエンティティで指定する方法

2.2.5.5.1 外部キーと同一エンティティ内でライフタイム期間を指定する方法

外部キーが定義されているエンティティと同一のエンティティ上にライフタイム期間を定義する場合、外部キーのみで一意に決定できる属性だけがライフタイム期間として指定できる。

- 外部キーが国際化も期間化もされていない場合、以下のいずれかの属性がライフタイム期間として指定できる。
  - 国際化も期間化もされていない属性
- 外部キーが国際化のみされている場合、以下のいずれかの属性がライフタイム期間として指定できる。
  - 国際化も期間化もされていない属性
  - 国際化のみされている属性
- 外部キーが期間化のみされている場合、以下のいずれかの属性がライフタイム期間として指定できる。
  - 国際化も期間化もされていない属性
  - 期間化のみされている属性
- 外部キーが国際化かつ期間化されている場合、以下のいずれかの属性がライフタイム期間として指定できる。
  - 国際化も期間化もされていない属性
  - 期間化のみされている属性

[3] たとえば、有効開始日時は期間化も国際化もされていないが、有効終了日時は国際化の対象である、といった指定はできない。

- 期間化かつ国際化されている属性

2.2.5.5.2 他のエンティティでライフタイム期間を指定する方法

外部キーが定義されているエンティティと異なるエンティティ上にライフタイム期間を定義する場合、該当するエンティティまでのリレーションシップのルートを指定する。この場合、その間のリレーションシップはすべてソースエンティティからターゲットエンティティへ向かう方向にしかたどれない。また、これらのリレーションシップにおいて国際化制約、期間化制約およびライフタイム制約が指定されている場合、それらの決定項目（国際化基準項目、期間化基準項目およびライフタイム期間）はそのリレーションシップに関連付けられているソースエンティティとターゲットエンティティ内でのみ解決できるものでなければならない。

ライフタイム期間が定義されるエンティティと外部キーが定義されているエンティティが異なる場合、ライフタイム期間が定義される参照先までの間のリレーションシップとエンティティは以下のような条件を満たす必要がある。

- ソースエンティティから見た外部エンティティへのリレーションシップに含まれる外部キースコープは、ソースエンティティから見たターゲットエンティティへのリレーションシップに含まれる外部キースコープの範囲に収まっていなければならない。
- ライフタイム期間が定義されているエンティティへのリレーションシップの場合（そこから先に別のリレーションシップをたどる必要がない場合）、ライフタイム期間となる属性が一意に決定できる制約がリレーションシップに指定されている必要がある。
  - 国際化も期間化もされていない属性は常にライフタイム期間として指定できる。
  - 国際化のみされている属性をライフタイム期間として指定する場合、リレーションシップには国際化基準項目が指定されている必要がある。
  - 期間化のみされている属性をライフタイム期間として指定する場合、リレーションシップには期間化基準項目が指定されている必要がある。
  - 期間化かつ国際化されている属性をライフタイム期間として指定する場合、リレーションシップには国際化基準項目と期間化基準項目の両方が指定されている必要がある。
- ライフタイム期間が定義されていないエンティティへのリレーションシップの場合（そこから先に別のリレーションシップをたどる必要がある場合）、他のエンティティを検索するための外部キーは以下のいずれかの条件を満たす必要がある。
  - 国際化も期間化もされていない属性は常に外部キーとして指定できる。
  - 国際化のみされている属性が外部キーとして指定されている場合、リレーションシップには国際化基準項目が指定されている必要がある。
  - 期間化のみされている属性が外部キーとして指定されている場合、リレーションシップには期間化基準項目が指定されている必要がある。
  - 期間化かつ国際化されている属性が外部キーとして指定されている場合、リレーションシップには国際化基準項目と期間化基準項目の両方が指定されている必要がある。

2.2.6 参照先削除時における整合性の維持

あるリレーションシップが定義されている場合、参照先が単純に削除されると「2.2.5 リレーションシップの制約」で述べた整合性制約が維持できなくなる。この制約を維持するため、参照先が削除された場合になんらかの処理を行う必要がある。

ここでは制約を維持するために以下に示すもののうちいずれか 1 つを必ず指定する必要がある。ここで指定する内容の詳細は「2.4.1 ターゲットエンティティのインスタンスの削除」を参照すること。

- 参照元を削除（Cascade）
- 参照元の外部キー情報を null に設定（Null）
- 例外（Exception）

"Null"を指定する場合、「2.2.5.1 外部キー制約」で説明されている外部キーの一部またはすべてがソースエンティティの従属属性でなければならない。外部キーがすべてソースエンティティのプライマリキーである場合、この制約に"Null"を指定することはできない。

2.2.6.1 参照元を削除

参照先を削除する前に参照元となるデータを先に削除する。こうすることにより、整合性制約を維持したまま参照先も削除することが可能となる。

2.2.6.2 参照元の外部キー情報をnullに設定

参照先を削除する前に参照元となるデータの外部キー情報を null にする。ここでいう外部キー情報とは、以下のうちのいずれかを指す。

- 外部キー
- 国際化基準項目（国際化制約が設定されているときのみ）
- 期間化基準項目（期間化制約が設定されているときのみ）

これらの情報がすべて設定されている（null 以外の値が入っている）場合、なんらかのデータを参照しているものとして扱う。これらの情報の一部でも null に設定されている場合、何のデータも参照していないことと同じ扱いとなる。

この処理を指定する場合、「2.2.5.1 外部キー制約」で定義される外部キーのうちのどの従属属性をnullとするのかを詳細に決定する必要がある。「図 2-2 リレーションシップの表記」では"DeleteNulls"で指定される値が該当する。ここで指定する従属属性は複数であってもかまわないが、外部キースコープの最下層の属性スコープに存在する従属属性でなければならない。

「2.2.5.2 国際化制約」で定義された国際化制約や「2.2.5.3 期間化制約」で定義された期間化制約が指定されているときにこの処理を指定する場合、国際化基準項目や期間化基準項目はソースエンティティ内で定義されていなければならない。国際化基準項目や期間化基準項目が他のエンティティで定義されている場合、この処理は指定できない。

2.2.6.3 例外

参照先を削除しようとした場合例外を発生させる。この場合、アプリケーションが明示的に参照元のデータを削除してから参照先のデータを削除しなければならない。

2.2.7 参照先期間更新時の制約の維持

リレーションシップにライフタイム制約が定義されている場合、参照先の期間に変更があると「2.2.5.4 ライフタイム制約」で述べたライフタイム制約による整合性が維持できなくなる場合がある。つまり、参照元の期間が参照先の期間に含まれなくなる箇所が出る場合がある。この制約を維持するため、参照先の期間が変更された場合になんらかの処理を行う必要がある。

ここでは制約を維持するために以下の方法を用意する。ここで指定する内容の詳細は「2.4.2 ターゲットエンティティのインスタンスの期間の削除」および「2.4.3 ソースエンティティのインスタンスの期間の追加または変更」を参照すること。

- 参照元の期間を削除（Cascade）
- 参照元の外部キー情報を null に設定（Null）
- 例外（Exception）

2.2.7.1 参照元の期間を削除

参照先の期間を削除する前に参照元となるデータの期間を先に削除する。こうすることにより、ライフタイムの整合性制約を維持したまま参照先の期間も削除することが可能となる。

「2.2.5.5 ライフタイム期間」で述べたライフタイム期間を設定している場合、この処理は指定できない。

2.2.7.2 参照元の外部キー情報をnullに設定

参照先の期間を削除する前に参照元となるデータの期間情報の外部キー情報を null にする。外部キー情報がすべて設定されている(null 以外の値が入っている)場合、なんらかのデータを参照しているものとして扱う。これらの情報の一部でも null に設定されている場合、何のデータも参照していないことと同じ扱いとなる。

この処理を指定する場合、外部キーのうちのどの属性をnullとするのかを詳細に決定する必要がある。ここで決定できる属性は「2.2.5.1 外部キー制約」で指定される外部キーとして指定されている従属属性の一部またはすべてであり、かつ期間化されている従属属性でなければならない。

「2.2.5.5 ライフタイム期間」で述べたライフタイム期間を設定している場合、この処理は指定できない。

2.2.7.3 例外

参照先の期間を削除しようとした場合、参照元の期間が参照先の期間に含まれない部分が生じる場合は例外を発生させる。この場合、アプリケーションが明示的に参照元の期間を削除してから参照先の期間を削除しなければならない。

2.2.8 多重度

リレーションシップではターゲットエンティティのインスタンスの立場から見るとソースエンティティのインスタンスは 0 以上存在する。一方、ソースエンティティの立場から見るとターゲットエンティティは存在しないか 1 つだけ存在するかのいずれかである。これらの多重度は「図 2-3 多重度の表記」のように表記される。

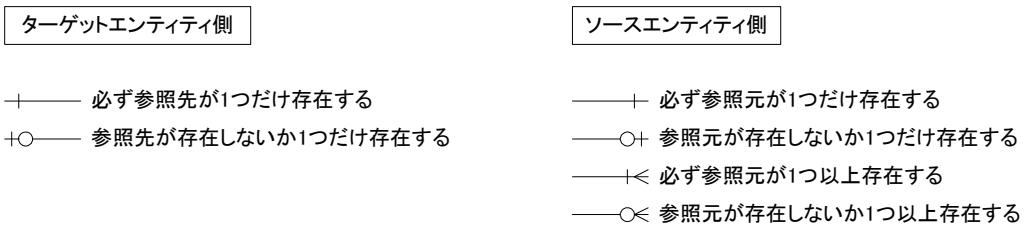


図 2-3 多重度の表記

注意:アプリケーション共通マスタでは多重度の表記は便宜上のものであり、システム上は、ターゲットエンティティ側は「何も参照していないか 1 つだけ参照している」、ソースエンティティ側は「どこからも参照されていないか 1 つ以上から参照されている」として扱われる。

2.3 アプリケーションセグメント

複数のエンティティ、リレーションシップおよび拡張従属属性による集合はある意味を持つ単位で分割することができる。本書ではこの分割単位をアプリケーションセグメントと呼ぶ。

アプリケーションセグメントは以下の集合体として定義される。

- エンティティ
- リレーションシップ
- 拡張従属属性

リレーションシップは通常、同一アプリケーションセグメント内のエンティティを結合するが、片方または両方のエンティティがリレーションシップとは異なるアプリケーションセグメントで定義されていてもよい。

拡張従属属性が追加されるエンティティもまた、同一アプリケーションセグメントまたは他のアプリケーションセグメント内のいずれで定義されていてもよい。

アプリケーションセグメントの例を「図 2-4 アプリケーションセグメントの例」に示す。

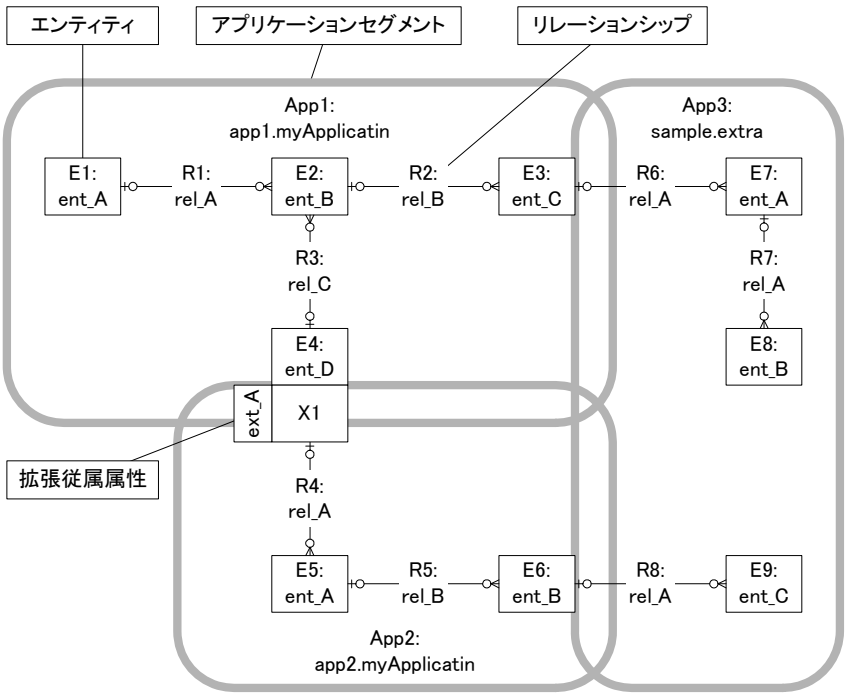


図 2-4 アプリケーションセグメントの例

「図 2-4 アプリケーションセグメントの例」では「表 2-4 アプリケーションセグメントの例の内容」に示すような定義がされている。

表 2-4 アプリケーションセグメントの例の内容

アプリケーションセグメント			名前	備考
App1 app1.myApplication	エンティティ	E1	ent_A	
		E2	ent_B	
		E3	ent_C	
		E4	ent_D	
	リレーションシップ	R1	rel_A	
		R2	rel_B	
		R3	rel_C	
App2 app2.myApplication	エンティティ	E5	ent_A	
		E6	ent_B	
	リレーションシップ	R4	rel_A	アプリケーションセグメント App1 のエンティティ E4 に対する参照
		R5	rel_B	
	拡張従属属性	X1	ext_A	アプリケーションセグメント App1 のエンティティ E4 に対する拡張
App3 sample.extra	エンティティ	E7	ent_A	
		E8	ent_B	
		E9	ent_C	

	リレーションシップ	R6	rel_A	アプリケーションセグメント App1 のエンティティ E3 に対する参照
		R7	rel_B	
		R8	rel_C	アプリケーションセグメント App2 のエンティティ E6 に対する参照

アプリケーションセグメントはそれぞれ一意となるアプリケーション名を持つ。アプリケーション名は英数字およびアンダースコア("_")の組合せとそれらを区切るピリオド(".")で決定される。

1 つのアプリケーションセグメントに含まれるエンティティ、リレーションシップおよび拡張従属属性はそれぞれ一意となる名前を持つ。これらの名前は英数字およびアンダースコア("_")の組合せで決定される。アプリケーションセグメントが異なればこれらは重複した名前であってもよい。たとえば、「図 2-4 アプリケーションセグメントの例」ではエンティティ名ent_Aは重複しているが、これらはアプリケーションセグメントが異なっているため問題ない。

2.4 整合性

エンティティ間にリレーションシップが定義されている場合、それぞれのエンティティのインスタンス間ではリレーションシップの定義に従って整合性を保つ必要がある。

2.4.1 ターゲットエンティティのインスタンスの削除

ソースエンティティのインスタンスが参照しているターゲットエンティティのインスタンスが削除された場合、以下のいずれかの処理によって整合性を保つ必要がある。

- ソースエンティティのインスタンスも同時に削除する(Cascade)。
  「図 2-5 同時削除の例(インスタンス削除時)」の例では会社コードが"compA"である会社情報を削除したとき、その構成部門である"orgn0001"および"orgn0002"の組織も同時に削除されている。
- ソースエンティティのインスタンスの外部キーに該当する属性をnullにする(Null)。
  この設定はターゲットエンティティのプライマリキーに対応するソースエンティティの外部キーのすべてまたは一部が従属属性(プライマリキー以外の属性)であるときのみ指定できる。この場合、nullにするべき属性も指定する必要がある。「図 2-6 削除時null設定の例(インスタンス削除時)」の例では分類コードをnullに設定するように指定されている(NullKeys)。「図 2-6 削除時null設定の例(インスタンス削除時)」の例では分類コードが"groupA"である分類情報を削除したとき、その情報を参照している"item0001"および"item0003"の商品の分類コードがnullに変更されている。
- 例外を投げる(Exception)。
  「図 2-7 削除時例外の例(インスタンス削除時)」の例ではユーザコードが"user0001"であるユーザ情報を削除しようとしているが、その情報を参照している"order0001"および"order0003"の注文が存在しているため例外を投げている。

ここで述べている「ターゲットエンティティのインスタンスの削除」には次のような場合も含まれる。

- ソースエンティティのインスタンスが特定の時点におけるターゲットエンティティのインスタンスを参照しており、ターゲットエンティティのインスタンスからその時点が含まれる期間のデータが削除された場合(ターゲットエンティティのインスタンス自体は残っている)
- ソースエンティティのインスタンスがターゲットエンティティのインスタンスの特定の言語のデータを参照しており、ターゲットエンティティのインスタンスからその言語に該当するデータが削除された場合(ターゲットエンティティのインスタンス自体は残っている)

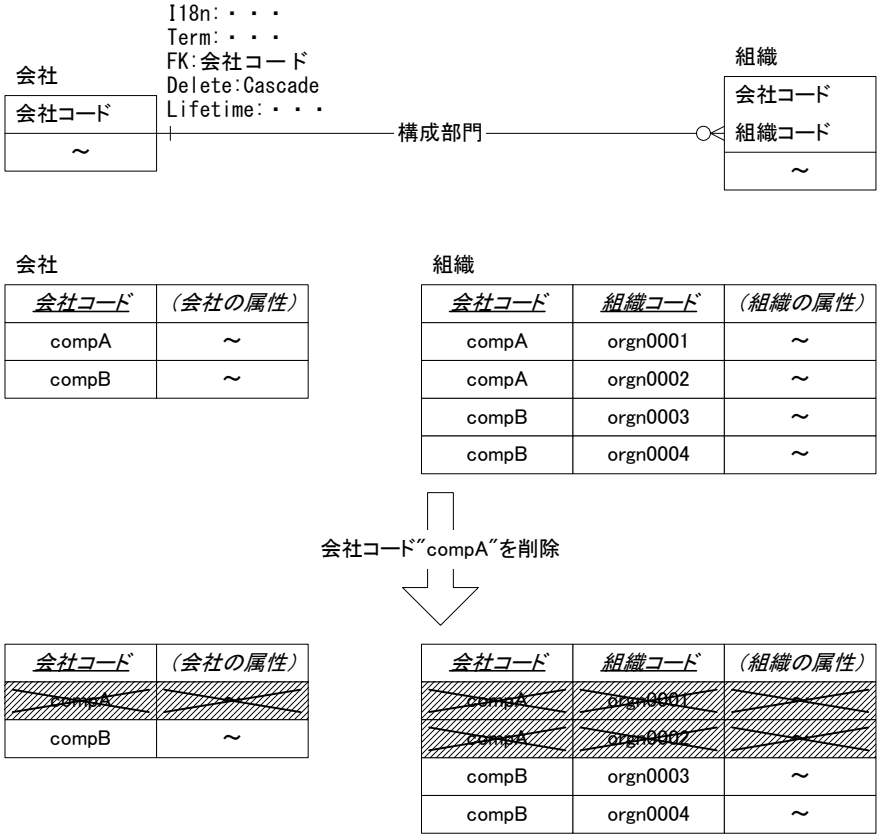


図 2-5 同時削除の例（インスタンス削除時）

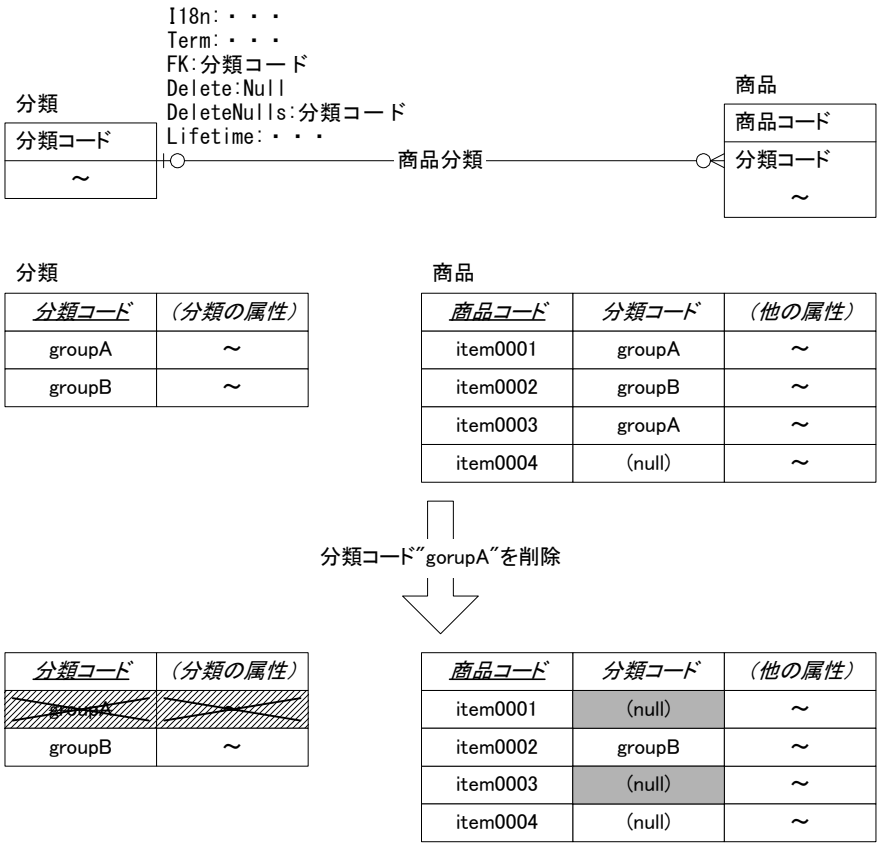


図 2-6 削除時 null 設定の例（インスタンス削除時）

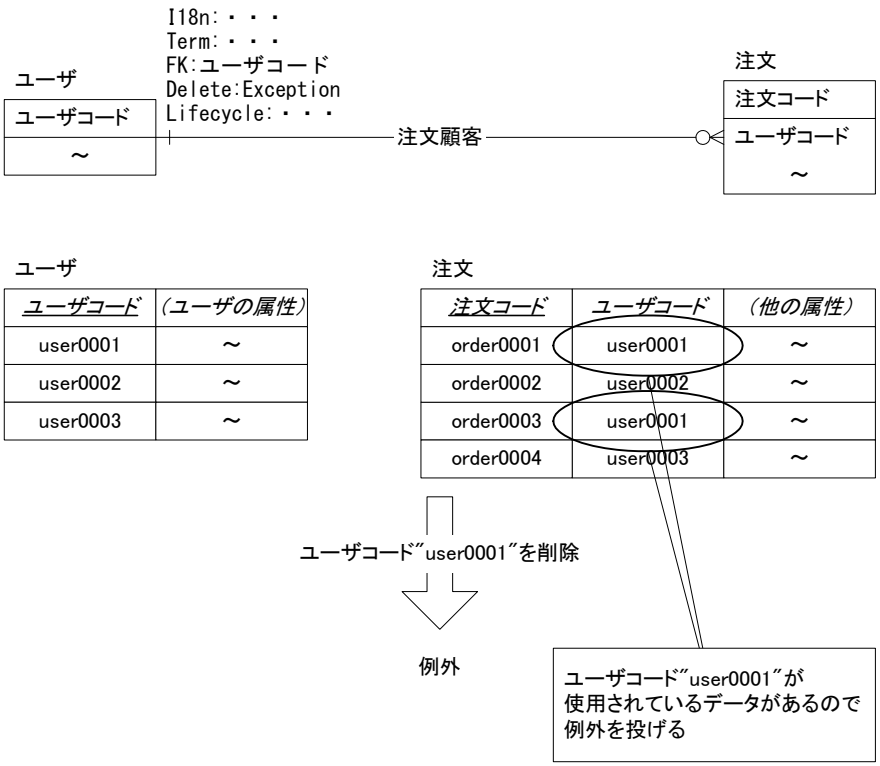
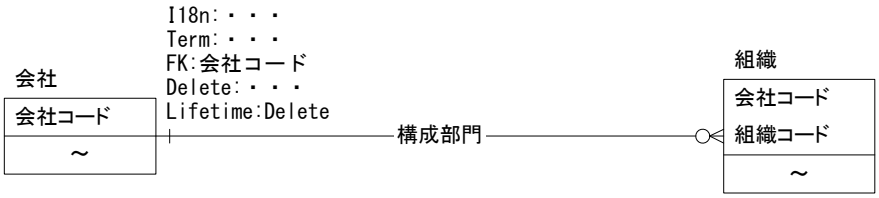


図 2-7 削除時例外の例(インスタンス削除時)

2.4.2 ターゲットエンティティのインスタンスの期間の削除

ソースエンティティとターゲットエンティティの間にライフタイムが指定されているとき、ターゲットエンティティのインスタンスの期間が一部削除された場合、以下のいずれかの処理によって整合性を保つ必要がある。

- ソースエンティティのインスタンスの該当する期間も同時に削除する(Cascade)。
  「図 2-8 同時削除の例(期間削除時)」の例では会社コードが"compA"である会社情報から 2005/01/01 以降の情報を削除したとき、その構成部門である"orgn0001"および"orgn0002"の組織も該当する期間の情報が同時に削除されている。ライフタイム期間が設定されている場合、この処理は指定できない。
- ソースエンティティのインスタンスの該当する期間の外部キーに該当する属性をnullにする(Null)。このとき、必要に応じて期間を分割する。
  「図 2-9 削除時null設定の例(期間削除時)」の例では分類コードが"groupA"である分類情報から 2005/01/01 以降の情報を削除したとき、その情報を参照している"item0001"および"item0002"の商品の該当する期間の分類コードがnullに変更されている。このとき、"item0001"の商品の 2004/12/01/から 2005/01/31 のデータは途中からnullに変更されることになる。そのため、2004/12/01/から 2004/12/31 までは変わらないが 2005/01/01 から 2005/01/31 までのデータを新規に追加し、分類コードをnullとする。分類コードや期間を除けば、新規に追加されるデータは 2004/12/01/から 2005/01/31 のデータからコピーされたものである。"item0002"の商品についても同様である。ライフタイム期間が設定されている場合、この処理は指定できない。
- 例外を投げる(Exception)。
  「図 2-10 削除時例外の例(期間削除時)」の例では会社コードが"compA"である会社情報から 2005/01/01 以降の情報を削除しようとしているが、その構成部門である"orgn0001"および"orgn0002"でその期間に該当する情報が存在しているため例外を投げている。



会社

会社コード	*期間*	*(他の属性)*
compA	2004/04/01～ 2004/09/30	～
	2004/10/01～ 2005/03/31	～

組織

会社コード	*組織コード*	*期間*	*(他の属性)*
compA	orgn0001	2004/04/01～ 2004/09/30	～
		2004/10/01～ 2004/11/30	～
		2004/12/01～ 2005/01/31	～
		2004/02/01～ 2005/03/31	～
compA	orgn0002	2004/04/01～ 2004/09/30	～
		2004/10/01～ 2005/03/31	～

会社"compA"から2005/01/01～2005/03/31の期間を削除

会社

会社コード	*期間*	*(他の属性)*
compA	2004/04/01～ 2004/09/30	～
	2004/10/01～ 2004/12/31	～

組織

会社コード	*組織コード*	*期間*	*(他の属性)*
compA	orgn0001	2004/04/01～ 2004/09/30	～
		2004/10/01～ 2004/11/30	～
		2004/12/01～ 2004/12/31	～
		~~2004/02/01～ 2005/03/31~~	
compA	orgn0002	2004/04/01～ 2004/09/30	～
		2004/10/01～ 2004/12/31	～

図 2-8 同時削除の例(期間削除時)

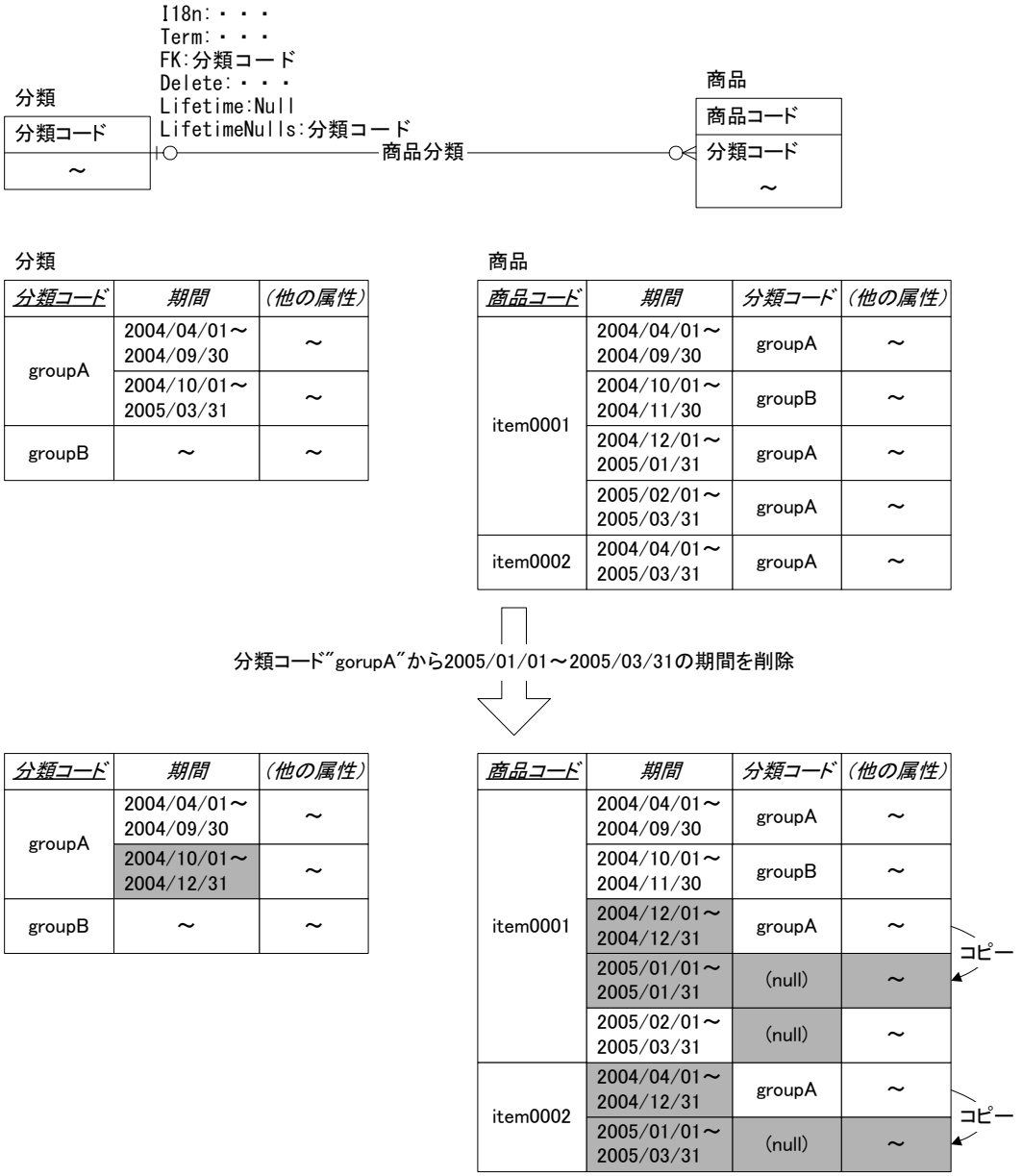


図 2-9 削除時 null 設定の例(期間削除時)

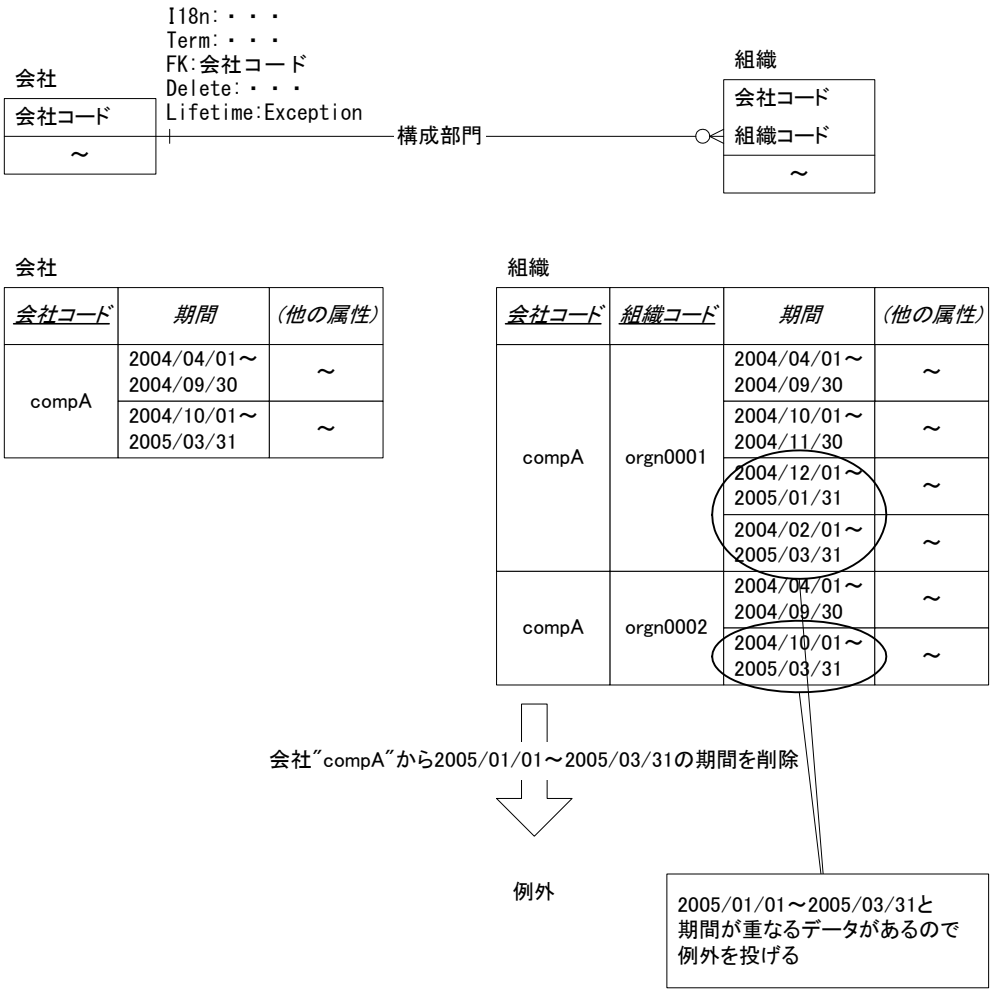


図 2-10 削除時例外の例(期間削除時)

2.4.3 ソースエンティティのインスタンスの期間の追加または変更

ソースエンティティとターゲットエンティティの間にライフタイムが指定されているとき、ターゲットエンティティのインスタンスの期間と矛盾するような期間をソースエンティティのインスタンスの期間に追加または変更しようとした場合、例外を投げる。

「図 2-11 期間の追加または変更時例外の例」の例では組織に対して 2005/04/01 以降のデータを追加しようとしているが、会社では 2005/03/31 までしか情報が有効でないため例外を投げている。この例では組織の期間を単純に延長しようとした場合も同様の例外が発生する。

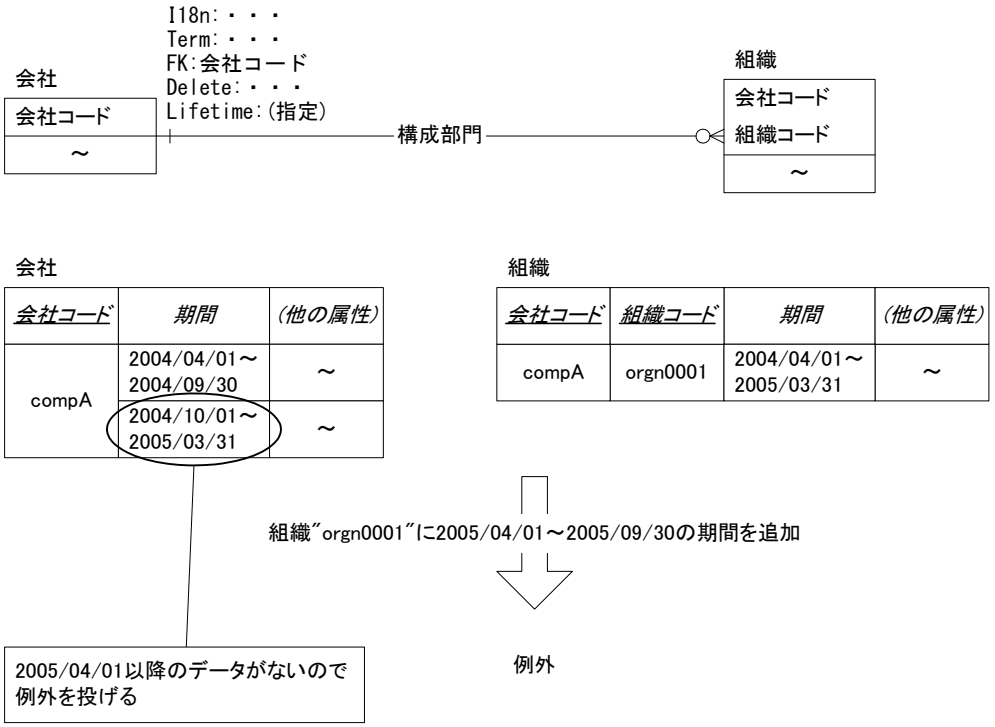


図 2-11 期間の追加または変更時例外の例

2.5 リレーショナルデータベースへのマッピング

ここまでに説明してきたエンティティとリレーションシップは概念的なものであり、実際のデータストアには依存しない抽象的なものとなっている。本節ではエンティティをリレーショナルデータベースに置き換える方法について説明する。

2.5.1 エンティティからテーブル

1 つのエンティティにつき、実際のエンティティは最小で1つ、最大で 4 つに分割される。これらは次のように分類できる。

- 基本テーブル
  エンティティにおいて期間化も国際化もされていない情報を扱う。
- 国際化テーブル
  エンティティにおいて国際化のみされている情報を扱う。
- 期間化テーブル
  エンティティにおいて期間化のみされている情報を扱う。
- 期間国際化テーブル
  エンティティにおいて期間化かつ国際化されている情報を扱う。

また、あるエンティティを拡張したエンティティも最小で 1 つ、最大で 4 つに分割される。

- 拡張基本テーブル
  拡張エンティティにおいて期間化も国際化もされていない情報を扱う。
- 拡張国際化テーブル
  拡張エンティティにおいて国際化のみされている情報を扱う。
- 拡張期間化テーブル
  拡張エンティティにおいて期間化のみされている情報を扱う。
- 拡張期間国際化テーブル

拡張エンティティにおいて期間化かつ国際化されている情報を扱う。

それぞれのテーブルの構成は「表 2-5 実テーブルの構成」のようになっている必要がある。

表 2-5 実テーブルの構成

テーブルの種別	テーブルの プライマリキー	プライマリキー以外のカラム
基本テーブル	プライマリキー	国際化と期間化のいずれの対象でもない従属属性 最終更新者(*) 最終更新日時(*)
国際化テーブル	プライマリキー ロケール(*)	国際化の対象であるが期間化の対象ではない従属属性
期間化テーブル	プライマリキー 期間 ID(*)	有効開始日時(*) 有効終了日時(*) 国際化の対象ではないが期間化の対象である従属属性
期間国際化テーブル	プライマリキー 期間 ID(*) ロケール(*)	国際化かつ期間化の対象である従属属性
拡張基本テーブル	プライマリキー	国際化と期間化のいずれの対象でもない拡張従属属性
拡張国際化テーブル	プライマリキー ロケール(*)	国際化の対象であるが期間化の対象ではない拡張従属属性
拡張期間化テーブル	プライマリキー 期間 ID(*)	国際化の対象ではないが期間化の対象である拡張従属属性
拡張期間国際化テーブル	プライマリキー 期間 ID(*) ロケール(*)	国際化かつ期間化の対象である拡張従属属性

「表 2-5 実テーブルの構成」において、以下のような意味がある。

- 期間 ID
  基本テーブルのレコード毎に対して一意となることが保証されている ID。期間国際化テーブルの期間 ID、拡張期間化テーブルの期間 ID および期間国際化テーブルの期間 ID に指定する値は期間化テーブルに存在しなければならない。
- 有効開始日時と有効終了日時
  このデータが有効である期間を表す。有効となる期間は「有効開始日時≦有効期間＜有効終了日時」となる関係がある。期間化テーブル内では、対応する基本テーブルのレコードが同じである場合、この有効期間が重なってはならない。
- ロケール
  システムで扱うことができる言語のロケール名を表す。
- 最終更新者と最終更新日時
  エンティティに該当するデータを最後に登録または更新したユーザと日時を表す。更新情報が定義されているときのみ定義する。

これらのテーブルのうち、基本テーブルのみ必須であり他のテーブルはエンティティが対応していなければ省略できる。ただし期間国際化テーブルが存在する場合、期間化テーブルの存在は必須である。

また、各拡張テーブルが存在する場合、以下のようにそれぞれに該当するテーブルが存在する必要がある。

- 拡張期間化テーブル --- 期間化テーブル
- 拡張国際化テーブル --- 国際化テーブル

■ 拡張期間国際化テーブル --- 期間国際化テーブル

エンティティ定義から実際のテーブルに変換している例を「図 2-12 エンティティからテーブルへのマッピング」に示す。

エンティティ定義

	組織	I T D
	会社コード	
	組織コード	
	組織名	I T
	略称	I T
	電話番号	T
	内線番号	T
	所在地	I
	国名	I
	創立日	
	組織区分	
アプリケーションA	代表者名	I T
	副代表者名	I T
	FAX番号	T
	FAX内線	T
	創立者	I
	備考	I
	略式ID	
	管理コード	

基本テーブルへのマッピング
テーブル名：tab_orgn_base

会社コード（PK）	comp_cd
組織コード（PK）	orgn_cd
創立日	foundation_day
組織区分	orgn_type
（更新者）	update_user_cd
（更新日時）	update_time_stamp

拡張基本テーブルへのマッピング
テーブル名：my_ext_orgn_base

会社コード（PK）	comp_cd
組織コード（PK）	orgn_cd
略式ID	short_id
管理コード	management_cd

期間化テーブルへのマッピング
テーブル名：tab_orgn_term

会社コード（PK）	comp_cd
組織コード（PK）	orgn_cd
（期間ID）（PK）	term_id
（有効開始日）	start_date
（有効終了日）	end_date
電話番号	tel_no
内線番号	tel_extension_no

拡張期間化テーブルへのマッピング
テーブル名：my_ext_orgn_term

会社コード（PK）	comp_cd
組織コード（PK）	orgn_cd
（期間ID）（PK）	term_id
FAX番号	fax_no
FAX番号	fax_extension_no

国際化テーブルへのマッピング
テーブル名：tab_orgn_i18n

会社コード（PK）	comp_cd
組織コード（PK）	orgn_cd
（ロケール）（PK）	locale_cd
所在地	address
国名	country_name

拡張国際化テーブルへのマッピング
テーブル名：my_ext_orgn_i18n

会社コード（PK）	comp_cd
組織コード（PK）	orgn_cd
（ロケール）（PK）	locale_cd
創立者	founder_name
備考	note

期間国際化テーブルへのマッピング

会社コード（PK）	comp_cd
組織コード（PK）	orgn_cd
（期間ID）（PK）	term_id
（ロケール）（PK）	locale_cd
組織名	orgn_name
略称	short_name

拡張期間国際化テーブルへのマッピング

会社コード（PK）	comp_cd
組織コード（PK）	orgn_cd
（期間ID）（PK）	term_id
（ロケール）（PK）	locale_cd
代表者名	representative
副代表者名	vice_representative

図 2-12 エンティティからテーブルへのマッピング

「図 2-12 エンティティからテーブルへのマッピング」の例ではプライマリキー、ロケールおよび期間IDはすべてのテーブルで同一のカラム名をつけているが、これらはエンティティごとに異なっていてもかまわない。ただし、それぞれに対応するカラムの型とサイズは完全に一致していなければならない。

各種のテーブルの関連を通常のER図に置き換えると「図 2-13 テーブル間の関連」のようになる。

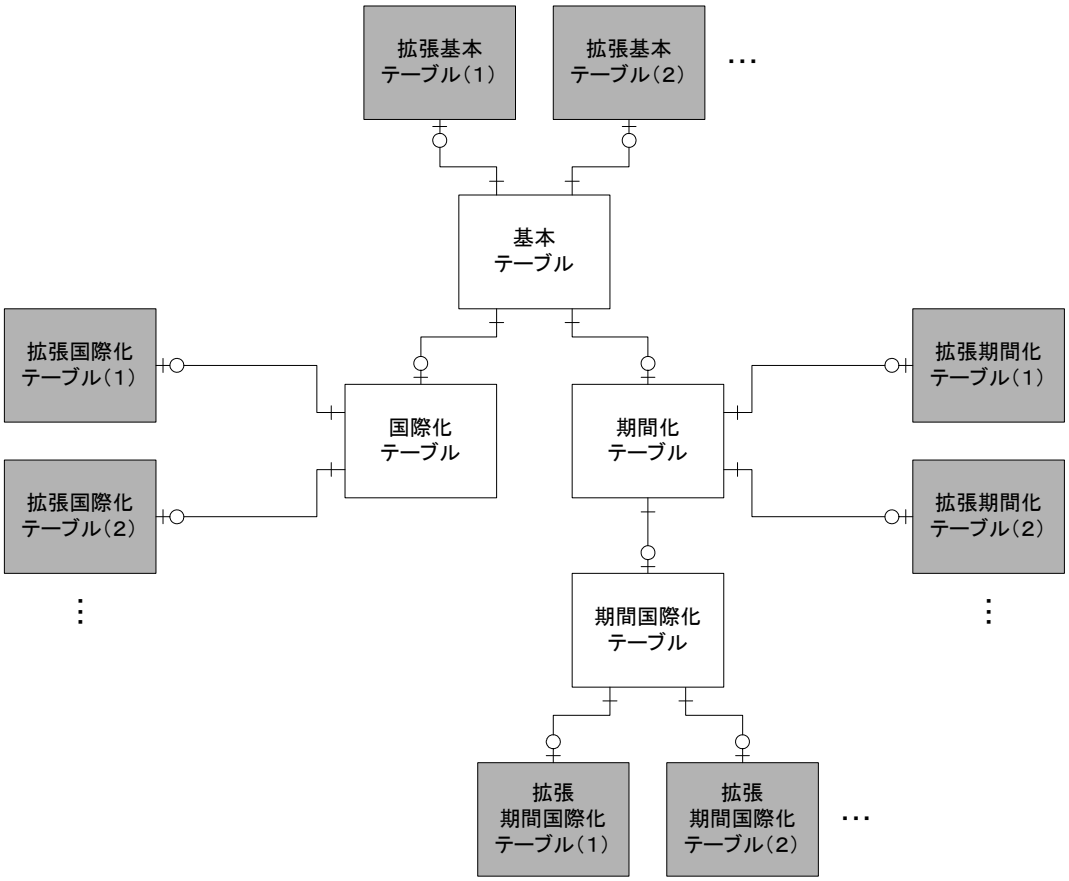


図 2-13 テーブル間の関連

2.5.1.1 基本テーブル

基本テーブルはエンティティの情報を元に以下のようにしてマッピングされる。

- エンティティのすべてのプライマリキーに1対1で対応するカラムが基本テーブルに存在する。
- エンティティの属性のうち国際化も期間化もされていないすべての従属属性に1対1で対応するカラムが基本テーブルに存在する。
- 基本テーブルでは、エンティティのプライマリキーに該当するカラムの組合せがデータベース上のプライマリキー制約を持つ。

2.5.1.2 国際化テーブル

国際化テーブルはエンティティの情報を元に以下のようにしてマッピングされる。

- エンティティのすべてのプライマリキーに1対1で対応するカラムが国際化テーブルに存在する。このカラムは基本テーブルに対する外部キーとなる。
- 言語を決定するカラムがただひとつ国際化テーブルに存在する。
- エンティティの属性のうち国際化のみされているすべての従属属性に1対1で対応するカラムが国際化テーブルに存在する。
- 国際化テーブルでは、エンティティのプライマリキーに該当するカラムおよび言語を決定するカラムの組合せがデータベース上のプライマリキー制約を持つ。

2.5.1.3 期間化テーブル

期間化テーブルはエンティティの情報を元に以下のようにしてマッピングされる。

- エンティティのすべてのプライマリキーに1対1で対応するカラムが期間化テーブルに存在する。このカラムは基本テーブルに対する外部キーとなる。

- 期間 ID を決定するカラムがただひとつ期間化テーブルに存在する。
- エンティティの属性のうち期間化のみされているすべての従属属性に1対1で対応するカラムが期間化テーブルに存在する。
- 期間化テーブルでは、エンティティのプライマリキーに該当するカラムおよび期間 ID を決定するカラムの組合せがデータベース上のプライマリキー制約を持つ。

2.5.1.4 期間国際化テーブル

期間国際化テーブルはエンティティの情報を元に以下のようにしてマッピングされる。

- エンティティのすべてのプライマリキーに1対1で対応するカラムが期間国際化テーブルに存在する。
- 期間 ID を決定するカラムがただひとつ期間国際化テーブルに存在する。
- 上記の 2 つのカラムの組み合わせが期間化テーブルに対する外部キーとなる。
- 言語を決定するカラムがただひとつ期間国際化テーブルに存在する。
- エンティティの属性のうち期間化かつ国際化されているすべての従属属性に1対1で対応するカラムが期間国際化テーブルに存在する。
- 期間国際化テーブルでは、エンティティのプライマリキーに該当するカラム、期間 ID を決定するカラムおよび言語を決定するカラムの組合せがデータベース上のプライマリキー制約を持つ。

2.5.1.5 拡張基本テーブル

拡張基本テーブルはエンティティおよび拡張エンティティの情報を元に以下のようにしてマッピングされる。

- エンティティのすべてのプライマリキーに1対1で対応するカラムが拡張基本テーブルに存在する。
- 拡張エンティティの属性のうち国際化も期間化もされていないすべての従属属性に1対1で対応するカラムが拡張基本テーブルに存在する。
- 拡張基本テーブルでは、エンティティのプライマリキーに該当するカラムの組合せがデータベース上のプライマリキー制約を持つ。このカラムは基本テーブルに対する外部キーとなる。

2.5.1.6 拡張国際化テーブル

拡張国際化テーブルはエンティティおよび拡張エンティティの情報を元に以下のようにしてマッピングされる。

- エンティティのすべてのプライマリキーに1対1で対応するカラムが拡張国際化テーブルに存在する。
- 言語を決定するカラムがただひとつ拡張国際化テーブルに存在する。
- 拡張エンティティの属性のうち国際化のみされているすべての従属属性に1対1で対応するカラムが拡張国際化テーブルに存在する。
- 拡張国際化テーブルでは、エンティティのプライマリキーに該当するカラムおよび言語を決定するカラムの組合せがデータベース上のプライマリキー制約を持つ。このカラムは国際化テーブルに対する外部キーとなる。

2.5.1.7 拡張期間化テーブル

拡張期間化テーブルはエンティティおよび拡張エンティティの情報を元に以下のようにしてマッピングされる。

- エンティティのすべてのプライマリキーに1対1で対応するカラムが拡張期間化テーブルに存在する。
- 期間 ID を決定するカラムがただひとつ拡張期間化テーブルに存在する。
- 拡張エンティティの属性のうち期間化のみされているすべての従属属性に1対1で対応するカラムが拡張期間化テーブルに存在する。
- 拡張期間化テーブルでは、エンティティのプライマリキーに該当するカラムおよび期間 ID を決定するカラムの組合せがデータベース上のプライマリキー制約を持つ。このカラムは期間化テーブルに対する外部キーとなる。

2.5.1.8 拡張期間国際化テーブル

拡張期間国際化テーブルはエンティティおよび拡張エンティティの情報を元に以下のようにしてマッピングされる。

- エンティティのすべてのプライマリキーに1対1で対応するカラムが拡張期間国際化テーブルに存在する。
- 期間 ID を決定するカラムがただひとつ拡張期間国際化テーブルに存在する。
- 言語を決定するカラムがただひとつ拡張期間国際化テーブルに存在する。
- 拡張エンティティの属性のうち期間化かつ国際化されているすべての従属属性に1対1で対応するカラムが拡張期間国際化テーブルに存在する。
- 拡張期間国際化テーブルでは、エンティティのプライマリキーに該当するカラム、期間 ID を決定するカラムおよび言語を決定するカラムの組合せがデータベース上のプライマリキー制約を持つ。このカラムは期間国際化テーブルに対する外部キーとなる。

2.5.2 属性とカラムの型変換

属性の型とリレーショナルテーブルの型は「表 2-6 属性型のリレーショナルテーブルとの対応」に示すような対応をとる必要がある。

表 2-6 属性型のリレーショナルテーブルとの対応

エンティティの属性の型	リレーショナルデータベースに対応可能なカラムタイプ
String	VARCHAR
Decimal	DECIMAL または VARCHAR
Date	VARCHAR
Float	DECIMAL または VARCHAR
Locale	VARCHAR

3 API

アプリケーション共通マスタに対する API は次のパッケージに分かれる。

- アクセサ
  データの値や保持
- モデル
  エンティティのデータと Java インスタンスのマッピング
- マネージャ
  業務ごとに定義されたデータアクセス処理

これらのパッケージの依存関係を「図 3-1 APIのパッケージの依存関係」に示す。

図 3-1 API のパッケージの依存関係

3.1 アクセサ

アクセサパッケージはデータの値やその変更内容を保持するインタフェースやクラスを持つパッケージである。

データストアに対応する値の保持に対する責任は以下のようなインタフェースが受け持つ。

- BaseAccessor
  国際化も期間化もされていない属性の情報を持つ。
- InternationalAccessor
  国際化のみされている属性の情報を持つ。
- TerminableAccessor
  期間化のみされている属性の情報を持つ。
- TerminableInternationalAccessor
  期間化および国際化されている属性の情報を持つ。

期間化や国際化の管理に対する責任は以下のようなインタフェースが受け持つ。

- InternationalAccessorMap
  指定された言語に対応する InternationalAccessor を管理する。
- TerminableAccessorMap
  指定された期間に対応する TerminableAccessor を管理する。期間の追加、削除、整合性の管理なども行う。
- TerminableInternationalAccessorMap
  指定された期間および言語に対応する TerminableInternationalAccessor を管理する。

拡張エンティティの値に対する責任は以下のようなインタフェースが受け持つ。

- ExtendedBaseAccessor
  拡張エンティティの属性のうち国際化も期間化もされていない属性の情報を持つ。
- ExtendedInternationalAccessor
  拡張エンティティの属性のうち国際化のみされている属性の情報を持つ。

- ExtendedTerminableAccessor
  拡張エンティティの属性のうち期間化のみされている属性の情報を持つ。
- ExtendedTerminableInternationalAccessor
  拡張エンティティの属性のうち国際化および期間化されている属性の情報を持つ。

拡張アクセサの管理に対する責任は以下のようなインタフェースが受け持つ。

- ExtendedBaseAccessorMap
  指定された拡張名に対応する ExtendedBaseAccessor を管理する。
- ExtendedInternationalAccessorMap
  指定された拡張名に対応する ExtendedInternationalAccessor を管理する。
- ExtendedTerminableAccessorMap
  指定された拡張名に対応する ExtendedTerminableAccessor を管理する。
- ExtendedTerminableInternationalAccessorMap
  指定された拡張名に対応する ExtendedTerminableInternationalAccessor を管理する。

これらのインタフェースの関連を「図 3-2 アクセサのクラス図（インタフェース）」に示す。

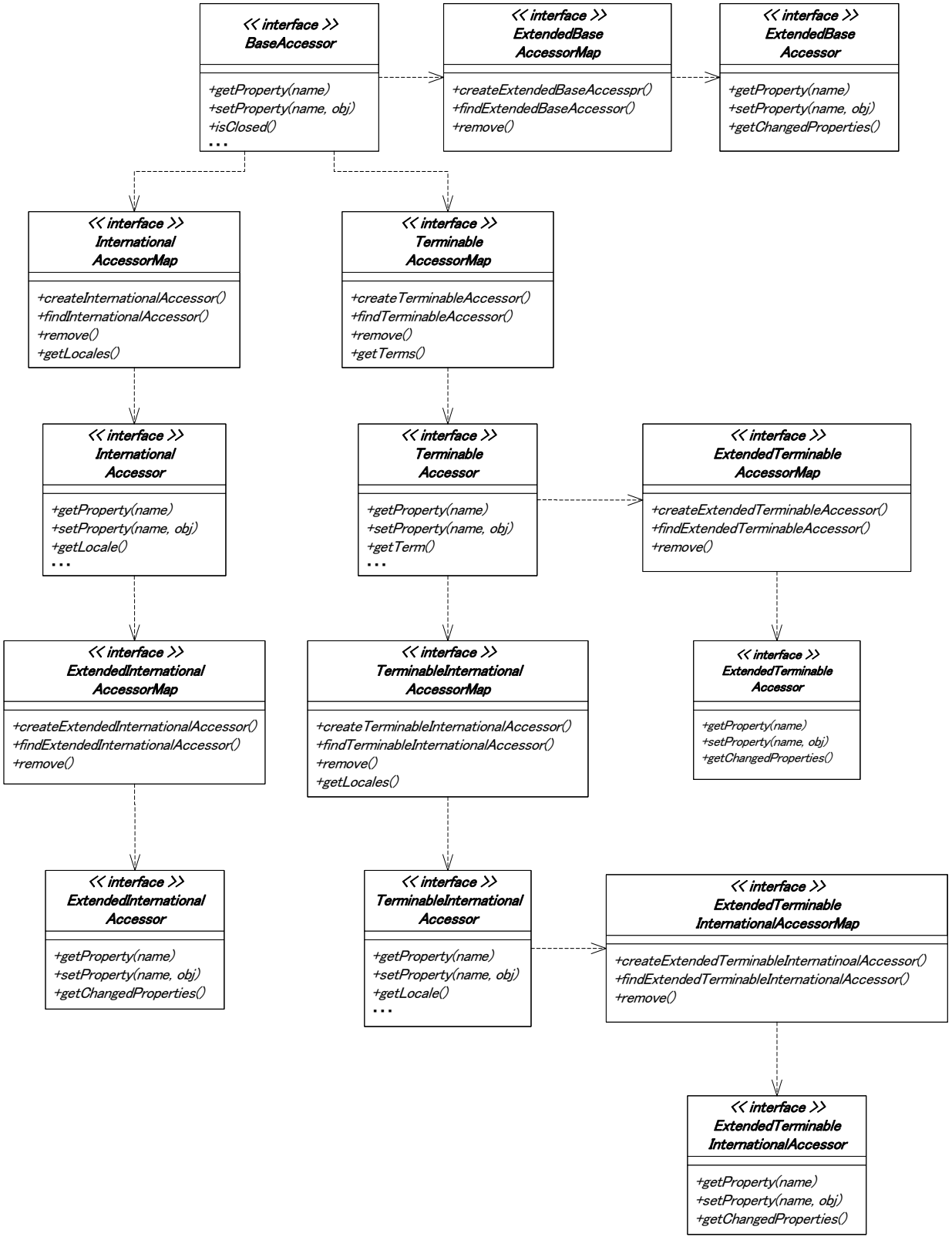


図 3-2 アクセサのクラス図（インタフェース）

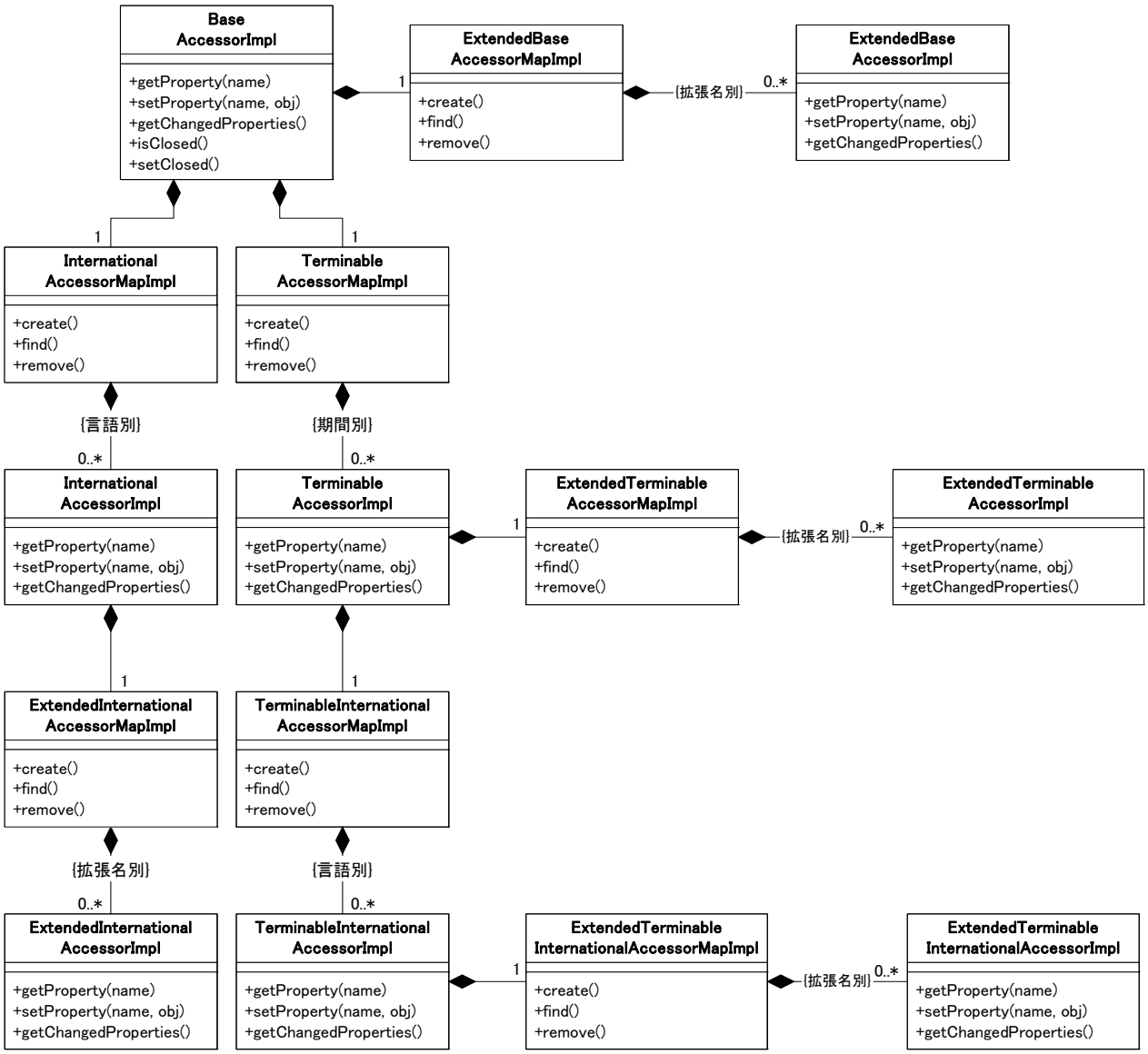


図 3-3 アクセサのクラス図（実装例）

アクセサは複数の属性を管理している。それぞれの属性は属性名で識別される。初期状態では属性は値が設定されているものと値が設定されていないものがある。ここでいう「値が設定されていない」というのは「null が設定されている」とは意味が違うことに注意する。アクセサはデータストアの値を常にすべて持っているとは限らない。データの通信量を減らしたりするため、処理時に必要となる属性の値のみを持っている場合もある。そのため、検索時に不要とされた属性についてはアクセサに値が設定されていない場合がある。一方、ある属性に該当するデータストア上の値が null（もしくはそれに該当する値）として設定されており、かつその属性が初期状態としてアクセサに登録されている場合、アクセサはその値を null として認識する。この場合、getProperty メソッドで取得される値は null である。

アクセサの getProperty メソッドでは引数に指定された属性名に該当する属性の値を取得する。値が設定されていない場合、getProperty メソッドは例外を返す。

アクセサの setProperty メソッドでは第一引数に指定された属性名に該当する属性の値を第二引数に指定された値に設定する。アクセサは setProperty メソッドによって属性の値が変更された場合、その内容を管理しておく。複数の属性が変更された場合でも、どの属性が変更されたのかという情報を内部で保持する。

getPropertyメソッドで取得する値やsetPropertyメソッドで設定する値は「3.1.2 属性とプロパティの型変換」で指定されている型でなければならない。

アクセサの実装クラスは「3.1.1.1 BaseAccessor」から「3.1.1.11 TerminableInternationalAccessorMap」までに記述している内容を実装する必要がある。

3.1.1 アクセサの構造

3.1.1.1 BaseAccessor

BaseAccessorはエンティティのインスタンスにおける基本的なデータを管理する。BaseAccessorで実装するべき内容を以下に示す。

- getPropertyメソッドは引数に指定された属性名に該当するプライマリキーおよび従属属性の値を取得する。属性名はプライマリキーまたは基本属性スコープの属性である必要がある。以下の場合は例外を返す。
  - エンティティに存在しない属性名を指定した場合
  - 拡張エンティティに存在する属性名を指定した場合
  - 引数に渡す属性として基本属性スコープ以外の属性を指定した場合
  - 検索時に取得対象としなかった属性を指定した場合
- setPropertyメソッドは第一引数に指定された属性名に該当する従属属性の値を第二引数に指定された値に設定する。基本属性スコープの従属属性のものである必要がある。以下の場合は例外を返す。
  - エンティティに存在しない属性名を指定した場合
  - 拡張エンティティに存在する属性名を指定した場合
  - 第一引数にプライマリキーに該当する属性名を指定した場合
  - 第一引数に基本属性スコープ以外の従属属性の属性名を指定した場合
  - 第一引数に更新情報属性の属性名を指定した場合
  - isClosedメソッドがtrueを返す場合
- getInternationalAccessorMapメソッドは言語とこのアクセサに関連するInternationalAccessorのマッピング情報を持つ適切なInternationalAccessorMapを返す。
- getTerminableAccessorMapメソッドは期間とこのアクセサに関連するTerminableAccessorのマッピング情報を持つ適切なTerminableAccessorMapを返す。
- getExtendedBaseAccessorMapメソッドは拡張名とこのアクセサに関連するExtendedBaseAccessorのマッピング情報を持つ適切なExtendedBaseAccessorMapを返す。
- setClosedメソッドによってこのアクセサがクローズされた状態にする。
- isClosedメソッドはこのアクセサがクローズされた状態である場合はtrueを返す。そうでなければfalseを返す。

3.1.1.2 InternationalAccessor

InternationalAccessor はエンティティのインスタンスにおいて国際化のみされている基本的なデータを管理する。InternationalAccessorで実装するべき内容を以下に示す。

- getPropertyメソッドは引数に指定された属性名に該当する従属属性の値を取得する。属性名は国際化属性スコープの属性である必要がある。以下の場合は例外を返す。
  - エンティティに存在しない属性名を指定した場合
  - 拡張エンティティに存在する属性名を指定した場合
  - 引数に渡す属性として国際化属性スコープ以外の従属属性を指定した場合
  - 検索時に取得対象としなかった属性を指定した場合
- setPropertyメソッドは第一引数に指定された属性名に該当する従属属性の値を第二引数に指定された値に設定する。属性名は国際化属性スコープの従属属性である必要がある。以下の場合は例外を返す。

- エンティティに存在しない属性名を指定した場合
- 拡張エンティティに存在する属性名を指定した場合
- 第一引数に国際化属性スコープ以外の従属属性の属性名を指定した場合
- BaseAccessor の isClosed メソッドが true を返す場合

- getExtendedInternationalAccessorMap メソッドは拡張名とこのアクセサに関連する ExtendedInternationalAccessor のマッピング情報を持つ適切な ExtendedInternationalAccessorMap を返す。
- getBaseAccessor メソッドはこのアクセサを含んでいる BaseAccessor を返す。

3.1.1.3 TerminableAccessor

TerminableAccessor はエンティティのインスタンスにおいて期間化のみされている基本的なデータを管理する。TerminableAccessor で実装するべき内容を以下に示す。

- getProperty メソッドは引数に指定された属性名に該当する従属属性の値を取得する。属性名は期間化属性スコープの属性である必要がある。以下の場合は例外を返す。
  - エンティティに存在しない属性名を指定した場合
  - 拡張エンティティに存在する属性名を指定した場合
  - 引数に渡す属性として期間化属性スコープ以外の従属属性を指定した場合
  - 検索時に取得対象としなかった属性を指定した場合
- setProperty メソッドは第一引数に指定された属性名に該当する従属属性の値を第二引数に指定された値に設定する。属性名は期間化のみされている従属属性のものである必要がある。以下の場合は例外を返す。
  - エンティティに存在しない属性名を指定した場合
  - 拡張エンティティに存在する属性名を指定した場合
  - 第一引数に期間化属性スコープ以外の従属属性の属性名を指定した場合
  - BaseAccessor の isClosed メソッドが true を返す場合
- getTerminableInternationalAccessorMap メソッドは期間とこのアクセサに関連する TerminableInternationalAccessor のマッピング情報を持つ適切な TerminableInternationalAccessorMap を返す。
- getExtendedTerminableAccessorMap メソッドは拡張名とこのアクセサに関連する ExtendedTerminableAccessor のマッピング情報を持つ適切な ExtendedTerminableAccessorMap を返す。
- getBaseAccessor メソッドはこのアクセサを含んでいる BaseAccessor を返す。
- getTerm メソッドはこのアクセサで示されるデータの有効期間をあらわす適切な Term を返す。

3.1.1.4 TerminableInternationalAccessor

TerminableInternationalAccessor はエンティティのインスタンスにおいて期間化かつ国際化されている基本的なデータを管理する。TerminableInternationalAccessor で実装するべき内容を以下に示す。

- getProperty メソッドは引数に指定された属性名に該当する従属属性の値を取得する。属性名は期間国際化属性スコープの属性である必要がある。以下の場合は例外を返す。
  - エンティティに存在しない属性名を指定した場合
  - 拡張エンティティに存在する属性名を指定した場合
  - 引数に渡す属性として期間国際化属性スコープ以外の従属属性を指定した場合
  - 検索時に取得対象としなかった属性を指定した場合
- setProperty メソッドは第一引数に指定された属性名に該当する従属属性の値を第二引数に指定された値に設定する。属性名は国際化かつ期間化されている従属属性のものである必要がある。以下の場合は例外を返す。
  - エンティティに存在しない属性名を指定した場合
  - 拡張エンティティに存在する属性名を指定した場合

- ◆ 第一引数に期間国際化属性スコープ以外の従属属性の属性名を指定した場合
- ◆ BaseAccessor の isClosed メソッドが true を返す場合

■ getExtendedTerminableInternationalAccessorMap メソッドは拡張名とこのアクセサに関連する ExtendedTerminableInternationalAccessor のマッピング情報を持つ適切な ExtendedTerminableInternationalMap を返す。

■ getTerminableAccessor メソッドはこのアクセサを含んでいる TerminableAccessor を返す。

3.1.1.5 ExtendedBaseAccessor

ExtendedBaseAccessor は拡張エンティティのインスタンスにおける基本的なデータを管理する。ExtendedBaseAccessor で実装するべき内容を以下に示す。

■ getProperty メソッドは引数に指定された属性名に該当する従属属性の値を取得する。属性名は基本属性スコープの属性である必要がある。以下の場合は例外を返す。

- ◆ 該当する拡張エンティティに存在しない属性名を指定した場合
- ◆ 引数に渡す属性として基本属性スコープ以外の属性を指定した場合
- ◆ 検索時に取得対象としなかった属性を指定した場合

■ setProperty メソッドは第一引数に指定された属性名に該当する従属属性の値を第二引数に指定された値に設定する。属性名は基本属性スコープ以外の従属属性のものである必要がある。以下の場合は例外を返す。

- ◆ 該当する拡張エンティティに存在しない属性名を指定した場合
- ◆ 第一引数に渡す属性として基本属性スコープ以外の属性を指定した場合
- ◆ isClosed メソッドが true を返す場合

■ getBaseAccessor メソッドはこのアクセサの拡張元となる BaseAccessor を返す。

3.1.1.6 ExtendedInternationalAccessor

ExtendedInternationalAccessor は拡張エンティティのインスタンスにおいて国際化のみされている基本的なデータを管理する。ExtendedInternationalAccessor で実装するべき内容を以下に示す。

■ getProperty メソッドは引数に指定された属性名に該当する従属属性の値を取得する。属性名は国際化属性スコープの属性である必要がある。以下の場合は例外を返す。

- ◆ 該当する拡張エンティティに存在しない属性名を指定した場合
- ◆ 引数に渡す属性として国際化属性スコープ以外の属性を指定した場合
- ◆ 検索時に取得対象としなかった属性を指定した場合

■ setProperty メソッドは第一引数に指定された属性名に該当する従属属性の値を第二引数に指定された値に設定する。属性名は国際化属性スコープの従属属性である必要がある。以下の場合は例外を返す。

- ◆ 該当する拡張エンティティに存在しない属性名を指定した場合
- ◆ 第一引数に渡す属性として国際化属性スコープ以外の属性を指定した場合
- ◆ isClosed メソッドが true を返す場合

■ getInternationalAccessor メソッドはこのアクセサの拡張元となる InternationalAccessor を返す。

3.1.1.7 ExtendedTerminableAccessor

ExtendedTerminableAccessor は拡張エンティティのインスタンスにおいて期間化のみされている基本的なデータを管理する。ExtendedTerminableAccessor で実装するべき内容を以下に示す。

■ getProperty メソッドは引数に指定された属性名に該当する従属属性の値を取得する。属性名は期間化属性スコープの属性である必要がある。以下の場合は例外を返す。

- ◆ 該当する拡張エンティティに存在しない属性名を指定した場合
- ◆ 引数に渡す属性として期間化属性スコープ以外の属性を指定した場合
- ◆ 検索時に取得対象としなかった属性を指定した場合

■ setProperty メソッドは第一引数に指定された属性名に該当する従属属性の値を第二引数に指定された値

に設定する。属性名は国際化も期間化もされていない従属属性のものである必要がある。以下の場合は例外を返す。

- ◆ 該当する拡張エンティティに存在しない属性名を指定した場合
- ◆ 第一引数に渡す属性として期間化属性スコープ以外の属性を指定した場合
- ◆ isClosed メソッドが true を返す場合

■ getTerminableAccessor メソッドはこのアクセサの拡張元となる TerminableAccessor を返す。

3.1.1.8 ExtendedTerminableInternationalAccessor

ExtendedTerminableInternationalAccessorは拡張エンティティのインスタンスにおいて期間化かつ国際化されている基本的なデータを管理する。ExtendedTerminableInternationalAccessor で実装するべき内容を以下に示す。

■ getProperty メソッドは引数に指定された属性名に該当する従属属性の値を取得する。属性名は期間国際化属性スコープの属性である必要がある。以下の場合は例外を返す。

- ◆ 該当する拡張エンティティに存在しない属性名を指定した場合
- ◆ 引数に渡す属性として期間国際化属性スコープ以外の属性を指定した場合
- ◆ 検索時に取得対象としなかった属性を指定した場合

■ setProperty メソッドは第一引数に指定された属性名に該当する従属属性の値を第二引数に指定された値に設定する。属性名は国際化も期間化もされていない従属属性のものである必要がある。以下の場合は例外を返す。

- ◆ 該当する拡張エンティティに存在しない属性名を指定した場合
- ◆ 第一引数に渡す属性として期間国際化属性スコープ以外の属性を指定した場合
- ◆ isClosed メソッドが true を返す場合

■ getTerminableInternationalAccessor メソッドはこのアクセサの拡張元となる TerminableInternationalAccessor を返す。

3.1.1.9 InternationalAccessorMap

InternationalAccessorMapは言語（ロケール）をキーとしInternationalAccessorを値とした独自のMap構造を持つ。

InternationalAccessorMap で実装するべき内容を以下に示す。

■ createInternationalAccessorメソッドは指定された言語に該当するInternationalAccessorを生成する。このとき、引数に既存の言語を指定した場合は該当する言語に対応する情報が上書きされる。
intra-martで扱う言語と既存のInternationalAccessorが「図 3-4 既存の言語対応データ（メモリ上）」のようになっている場合、たとえば英語の情報を新規に作成した場合は既存のデータInternationalAccessor(2)は上書きされる。

■ createInternationalAccessor メソッドは、以下の場合例外を throw する。

- ◆ BaseAccessor の isClosed メソッドが true を返す場合

■ findInternationalAccessorメソッドは引数に指定された言語に該当するInternationalAccessorを取得する。指定された言語に該当する国際化情報がデータストアに存在しない場合、nullを返す。
intra-martで扱う言語と既存のInternationalAccessorが「図 3-4 既存の言語対応データ（メモリ上）」のようになっている場合、たとえば日本語を指定するとInternationalAccessor(1)が取得されるが、スペイン語のデータを取得することはできない。

■ removeメソッドは指定された言語に該当するInternationalAccessorを削除する。
intra-martで扱う言語と既存のInternationalAccessorが「図 3-4 既存の言語対応データ（メモリ上）」のようになっている場合、たとえば英語のデータを削除した場合、図 3-5 削除後の言語対応データ（メモリ上）のようになる。

■ remove メソッドは、以下の場合例外を throw する。

- ◆ BaseAccessor の isClosed メソッドが true を返す場合

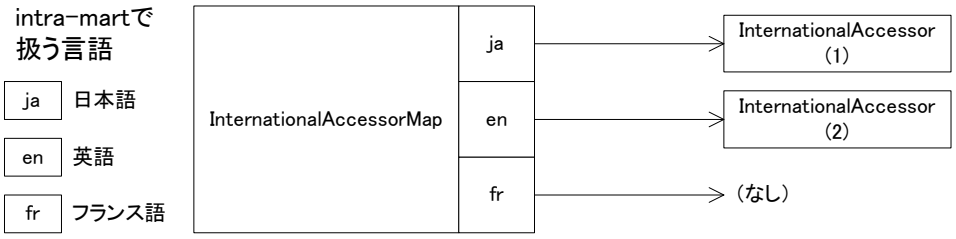


図 3-4 既存の言語対応データ(メモリ上)

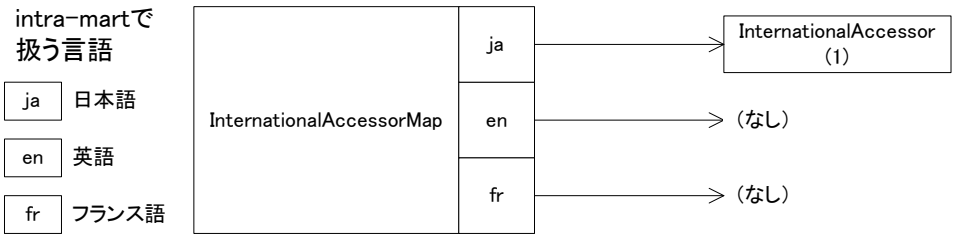


図 3-5 削除後の言語対応データ(メモリ上)

InternationalAccessorMapはintra-martで扱うすべての言語を持つ必要があるが、言語に関連付けられたInternationalAccessorは、検索時に指定がない限りすべて持つ必要はない。この場合、アプリケーションはデータストア上に言語に関連するデータが存在するかどうかを確認することができるが、検索時に指定した言語(あるいはリレーションシップで指定された言語)に関連する言語以外のデータが取得できるとは限らない。「図 3-6 実際のデータとメモリ上のデータ」の例では、データストア上には日本語および英語に関連するデータが存在していることはわかるが、実際にデータを取得できるのは日本語のデータのみである。

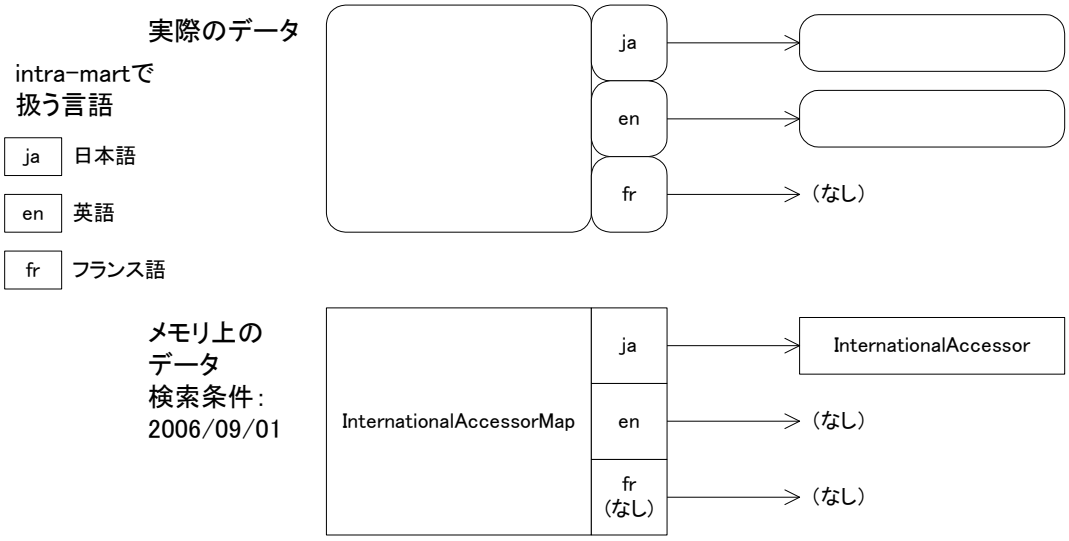


図 3-6 実際のデータとメモリ上のデータ

3.1.1.10 TerminableAccessorMap

TerminableAccessorMap は期間をキーとし TerminableAccessor を値とした独自の Map 構造を持つ。TerminableAccessorMap は期間が互いに重なり合うことがないようにその内容を管理する。

TerminableAccessorMap で実装するべき内容を以下に示す。

- createTerminableAccessor メソッドは指定された期間に該当する TerminableAccessor を生成する。

- createTerminableAccessor メソッドは、以下の場合例外を throw する。
  - ◆ 既存の期間と一部でも重なる部分がある場合
    既存のTerminableAccessorと期間が「図 3-7 既存の期間対応データ（メモリ上）」のようになっている場合、たとえば 2005/7/1 から 2006/2/1 までを期間とするデータを追加することはできるが、始まりが 2005/4/1 よりも前のデータ、または終了が 2006/4/1 以降のデータは登録できない。
  - ◆ BaseAccessor の isClosed メソッドが true を返す場合
- findTerminableAccessorメソッドは指定された日時に該当するTerminableAccessorを取得する。
  既存のTerminableAccessorと期間が「図 3-7 既存の期間対応データ（メモリ上）」のようになっている場合、たとえば 2006/11/15 時点のデータとしてTerminableAccessor(2)が取得されるが、2005/11/15 時点のデータを取得することはできない。
- removeメソッドは指定された期間に該当するTerminableAccessorを削除する。このとき、前後の有効期間を調節する。既存のTerminableAccessorと期間が「図 3-7 既存の期間対応データ（メモリ上）」のようになっている場合、たとえば 2007/4/1 から 2008/4/1 直前までのデータを削除した場合、前後の期間の開始および終了となる時点が調節される（「図 3-8 削除後の期間対応データ（メモリ上）」を参照）。
- remov メソッドは、以下の場合例外を throw する。
  - ◆ BaseAccessor の isClosed メソッドが true を返す場合

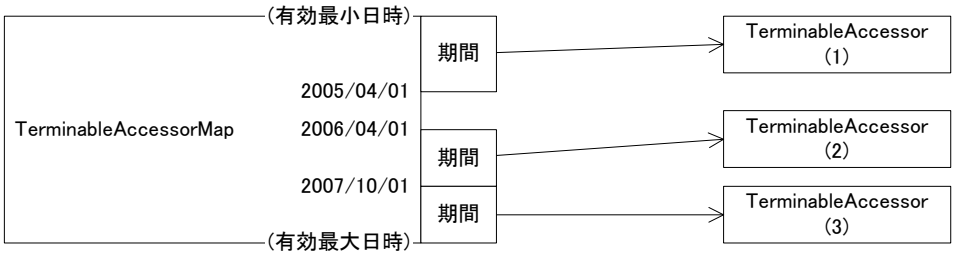


図 3-7 既存の期間対応データ（メモリ上）

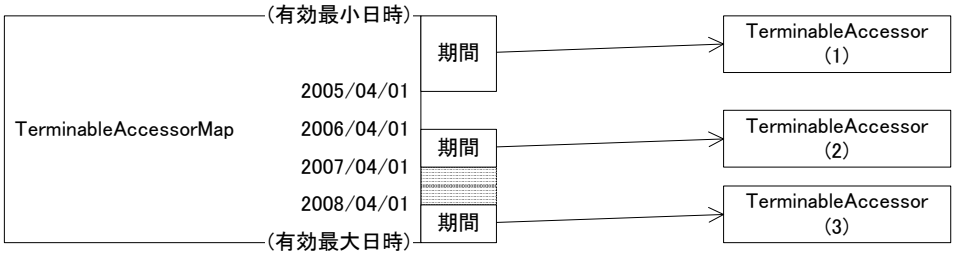


図 3-8 削除後の期間対応データ（メモリ上）

TerminableAccessorMapはデータストアに存在するすべての期間を持つ必要があるが、期間に関連付けられたTerminableAccessorは、検索時に指定がない限りすべて持つ必要はない。この場合、アプリケーションはデータストア上に存在する期間は確認することができるが、検索時に指定した日付（あるいはリレーションシップで指定された日付）に関連する期間以外のデータが取得できるとは限らない。データストアから取得していないTerminableAccessorに対してTerminableAccessorMapのfindTerminableAccessorメソッドでアクセスしようとした場合、TerminableAccessorMapは例外をthrowする。「図 3-9 実際のデータとメモリ上のデータ」の例では、データストア上には2005/4/1まで、2006/4/1から2007/10/1まで、2007/10/1以降という期間が存在していることはわかるが、実際にデータを取得できるのは 2006/4/1 から 2007/10/1 までのデータのみである。

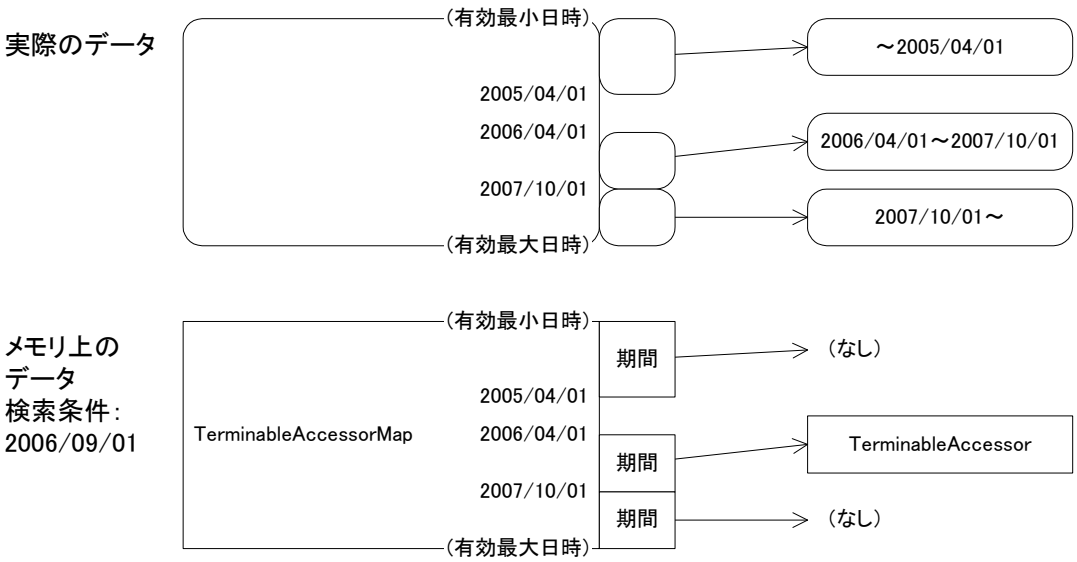


図 3-9 実際のデータとメモリ上のデータ

3.1.1.11 TerminableInternationalAccessorMap

TerminableInternationalAccessorMap は言語(ロケール)をキーとし TerminableInternationalAccessor を値とした独自の Map 構造を持つ。

TerminableInternationalAccessorMap で実装するべき内容を以下に示す。

- createTerminableInternationalAccessor メソッドは指定された言語(ロケール)に該当する TerminableInternationalAccessorを生成する。このとき、引数に既存の言語を指定した場合は該当する言語に対応する情報が上書きされる。
  intra-martで扱う言語と既存のTerminableInternationalAccessorが「図 3-10 既存の言語対応データ(メモリ上)」のようになっている場合、たとえば英語の情報を新規に作成した場合は既存のデータTerminableInternationalAccessor(2)は上書きされる。
- createTerminableInternationalAccessor メソッドは、以下の場合例外を throw する。
  - BaseAccessor の isClosed メソッドが true を返す場合
- findTerminableInternationalAccessorメソッドは引数に指定された言語(ロケール)に該当する TerminableInternationalAccessorを取得する。指定された言語(ロケール)に該当する国際化情報がデータストアに存在しない場合、nullを返す。intra-martで扱う言語と既存のTerminableInternationalAccessorが「図 3-10 既存の言語対応データ(メモリ上)」のようになっている場合、たとえば日本語を指定するとTerminableInternationalAccessor(1)が取得されるが、スペイン語のデータを取得することはできない。
- removeメソッドは指定された言語(ロケール)に該当するTerminableInternationalAccessorを削除する。intra-martで用意されている言語と既存のTerminableInternationalAccessorが「図 3-10 既存の言語対応データ(メモリ上)」のようになっている場合、たとえば英語のデータを削除した場合、図 3-11 削除後の言語対応データ(メモリ上)のようになる。
- remove メソッドは、以下の場合例外を throw する。
  - BaseAccessor の isClosed メソッドが true を返す場合

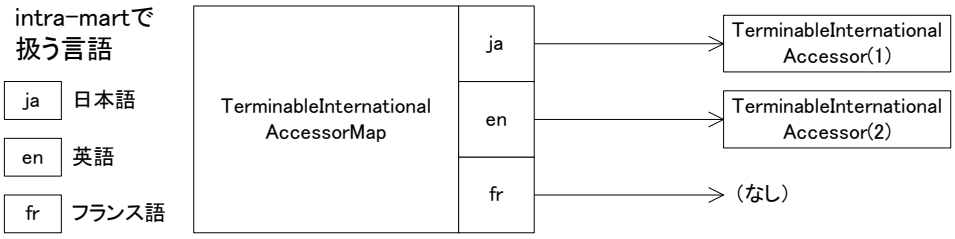


図 3-10 既存の言語対応データ（メモリ上）

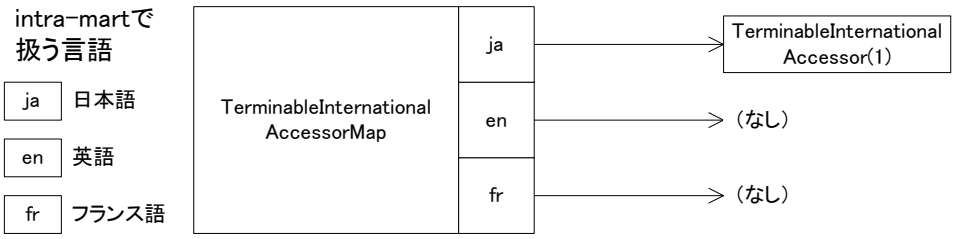


図 3-11 削除後の言語対応データ（メモリ上）

TerminableInternationalAccessorMapはintra-martで扱う言語を持つ必要があるが、言語に関連付けられたTerminableInternationalAccessorは、検索時に指定がない限りすべて持つ必要はない。この場合、アプリケーションはデータストア上に言語に関連するデータが存在するかどうかを確認することができるが、検索時に指定した言語（あるいはリレーションシップで指定された言語）に関連する言語以外のデータが取得できるとは限らない。「図 3-12 実際のデータとメモリ上のデータ」の例では、データストア上には日本語および英語に関連するデータが存在していることはわかるが、実際にデータを取得できるのは日本語のデータのみである。

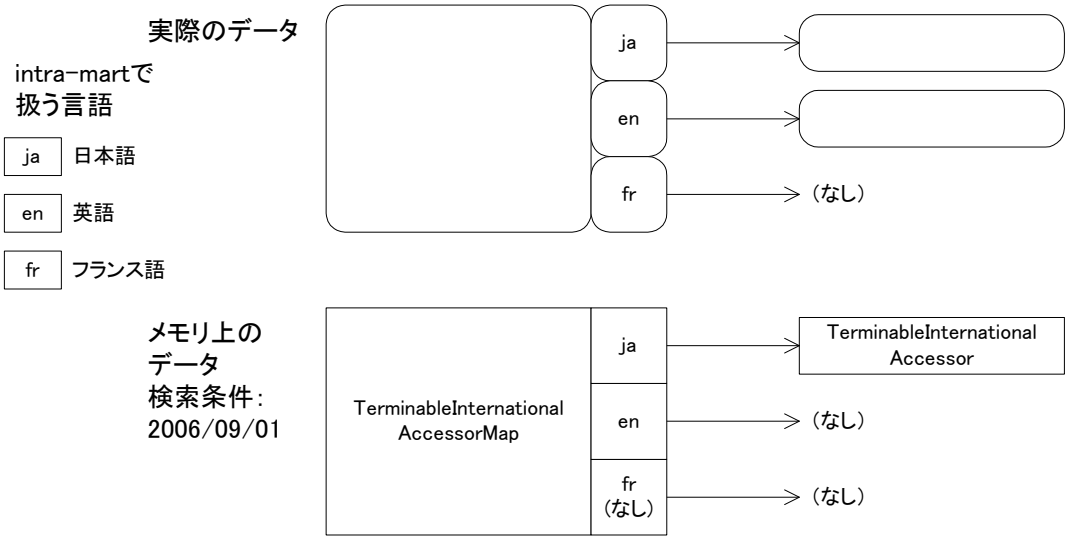


図 3-12 実際のデータとメモリ上のデータ

3.1.1.12 ExtendedBaseAccessorMap

ExtendedBaseAccessorMap は拡張名をキーとし ExtendedBaseAccessor を値とした独自の Map 構造を持つ。

ExtendedBaseAccessorMap で実装するべき内容を以下に示す。

- createExtendedBaseAccessor メソッドは指定された拡張名に該当する ExtendedBaseAccessor を生成する。
- createExtendedBaseAccessor メソッドは、以下の場合例外を throw する。
  - 設定ファイルに定義されていない拡張名を指定された場合

- ◆ 基本属性を持たない拡張エンティティの拡張名を指定された場合
- ◆ BaseAccessor の isClosed メソッドが true を返す場合

- findExtendedBaseAccessor メソッドは指定された拡張名に該当する ExtendedBaseAccessor を取得する。指定された拡張名に該当する拡張情報が存在しない場合は null を返す。
- findExtendedBaseAccessor メソッドは、以下の場合例外を throw する。
  - ◆ 設定ファイルに定義されていない拡張名を指定された場合
  - ◆ 基本属性を持たない拡張エンティティの拡張名を指定された場合
- remove メソッドは指定された拡張名に該当する ExtendedBaseAccessor を削除する。
- remove メソッドは、以下の場合例外を throw する。
  - ◆ BaseAccessor の isClosed メソッドが true を返す場合

3.1.1.13 ExtendedInternationalAccessorMap

ExtendedInternationalAccessorMap は拡張名をキーとし ExtendedInternationalAccessor を値とした独自の Map 構造を持つ。

ExtendedInternationalAccessorMap で実装するべき内容を以下に示す。

- CreateExtendedInternationalAccessor メソッドは指定された拡張名に該当する ExtendedInternationalAccessor を生成する。
- createExtendedInternationalAccessor メソッドは、以下の場合例外を throw する。
  - ◆ 設定ファイルに定義されていない拡張名を指定された場合
  - ◆ 国際化属性を持たない拡張エンティティの拡張名を指定された場合
  - ◆ BaseAccessor の isClosed メソッドが true を返す場合
- findExtendedInternationalAccessor メソッドは指定された拡張名に該当する ExtendedInternationalAccessor を取得する。指定された拡張名に該当する拡張情報が存在しない場合は null を返す。
- findExtendedInternationalAccessor メソッドは、以下の場合例外を throw する。
  - ◆ 設定ファイルに定義されていない拡張名を指定された場合
  - ◆ 国際化属性を持たない拡張エンティティの拡張名を指定された場合
- remove メソッドは指定された拡張名に該当する ExtendedInternationalAccessor を削除する。
- remove メソッドは、以下の場合例外を throw する。
  - ◆ BaseAccessor の isClosed メソッドが true を返す場合

3.1.1.14 ExtendedTerminableAccessorMap

ExtendedTerminableAccessorMap は拡張名をキーとし ExtendedTerminableAccessor を値とした独自の Map 構造を持つ。

ExtendedTerminableAccessorMap で実装するべき内容を以下に示す。

- createExtendedTerminableAccessor メソッドは指定された拡張名に該当する ExtendedTerminableAccessor を生成する。
- createExtendedTerminableAccessor メソッドは、以下の場合例外を throw する。
  - ◆ 設定ファイルに定義されていない拡張名を指定された場合
  - ◆ 期間化属性を持たない拡張エンティティの拡張名を指定された場合
  - ◆ BaseAccessor の isClosed メソッドが true を返す場合
- findExtendedTerminableAccessor メソッドは指定された拡張名に該当する ExtendedTerminableAccessor を取得する。指定された拡張名に該当する拡張情報が存在しない場合は null を返す。
- findExtendedTerminableAccessor メソッドは、以下の場合例外を throw する。
  - ◆ 設定ファイルに定義されていない拡張名を指定された場合
  - ◆ 期間化属性を持たない拡張エンティティの拡張名を指定された場合

- remove メソッドは指定された拡張名に該当する ExtendedTerminableAccessor を削除する。
- remove メソッドは、以下の場合例外を throw する。
  - BaseAccessor の isClosed メソッドが true を返す場合

3.1.1.15 ExtendedTerminableInternationalAccessorMap

ExtendedTerminableInternationalAccessorMap は拡張名をキーとし ExtendedTerminableInternationalAccessor を値とした独自の Map 構造を持つ。

ExtendedTerminableInternationalAccessorMap で実装するべき内容を以下に示す。

- createExtendedTerminableInternationalAccessor メソッドは指定された拡張名に該当する ExtendedTerminableInternationalAccessor を生成する。
- createExtendedTerminableInternationalAccessor メソッドは、以下の場合例外を throw する。
  - 設定ファイルに定義されていない拡張名を指定された場合
  - 期間国際化属性を持たない拡張エンティティの拡張名を指定された場合
  - BaseAccessor の isClosed メソッドが true を返す場合
- findExtendedTerminableInternationalAccessor メソッドは指定された拡張名に該当する ExtendedTerminableInternationalAccessor を取得する。指定された拡張名に該当する拡張情報が存在しない場合は null を返す。
- findExtendedTerminableInternationalAccessor メソッドは、以下の場合例外を throw する。
  - 設定ファイルに定義されていない拡張名を指定された場合
  - 基本属性を持たない拡張エンティティの拡張名を指定された場合
- remove メソッドは指定された拡張名に該当する ExtendedTerminableInternationalAccessor を削除する。
- remove メソッドは、以下の場合例外を throw する。
  - BaseAccessor の isClosed メソッドが true を返す場合

3.1.2 属性とプロパティの型変換

属性の型とアクセサに渡すプロパティのJavaの型は「表 3-1 属性型のJavaの型との対応」に示すような対応をとる必要がある。

表 3-1 属性型の Java の型との対応

エンティティの属性の型	アクセサに渡すプロパティの型
String	java.lang.String
Decimal	java.lang.String、 または java.lang.Integer、 または java.lang.Long または java.math.BigDecimal
Date	java.lang.String、 または java.util.Date
Float	java.lang.String、 または java.lang.Float、 または java.lang.Double または java.math.BigDecimal
Locale	java.lang.String、 または java.util.Locale

3.2 モデル

モデルパッケージはエンティティの情報を取得したり更新したりするインタフェースやクラスを持つパッケージである。

3.2.1 モデル

モデルはエンティティの 1 レコードに該当する内容を持つ。モデルは国際化、期間化および拡張に対してアクセサとほとんど同様の構造を持つ。アクセサとの違いは、プロパティに対して取得や設定を行う直接のメソッドを持たないことである(「図 3-13 モデルのクラス図(インタフェース:1)」および「図 3-14 モデルのクラス図(実装例:1)」を参照)。

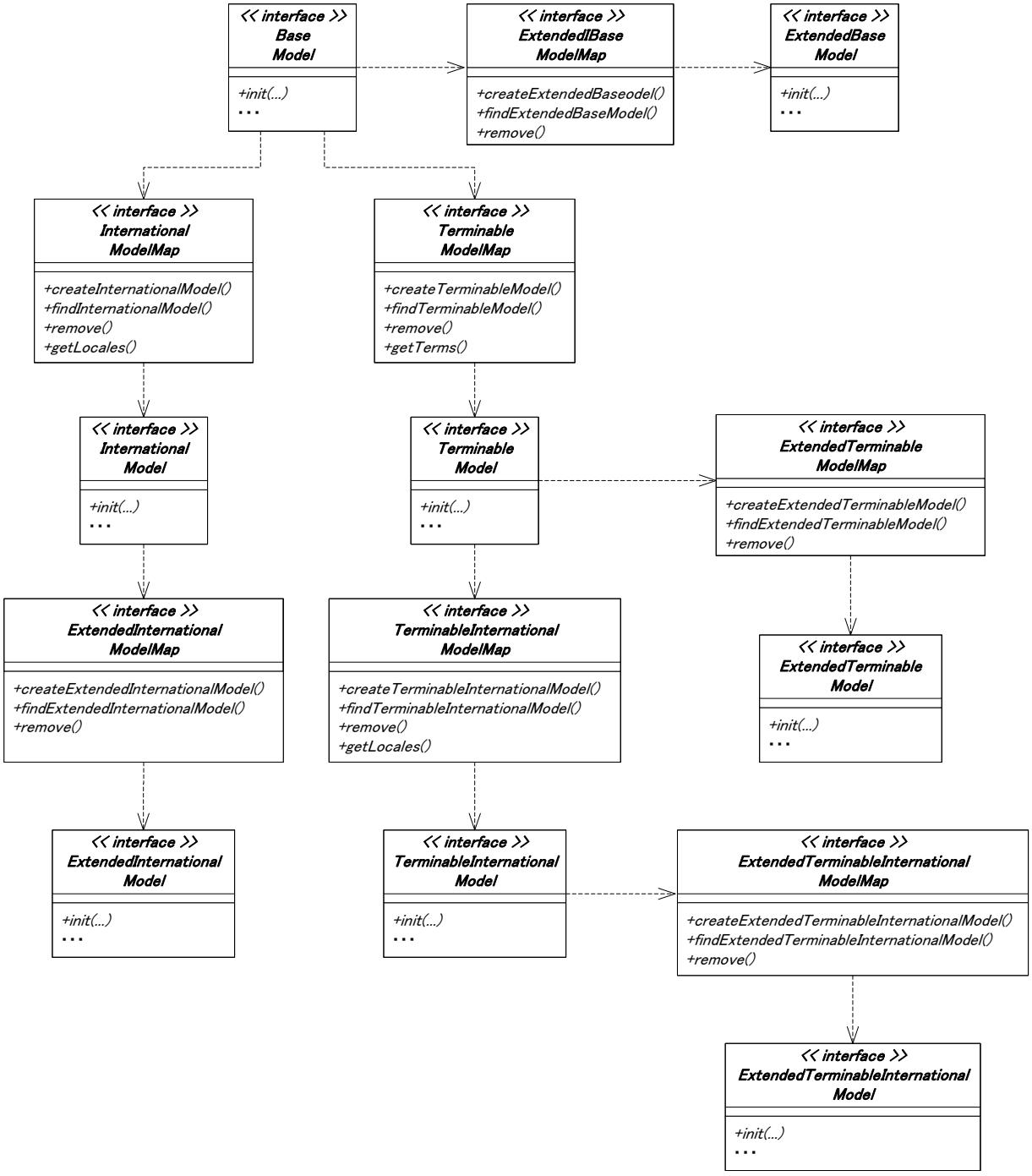


図 3-13 モデルのクラス図(インタフェース:1)

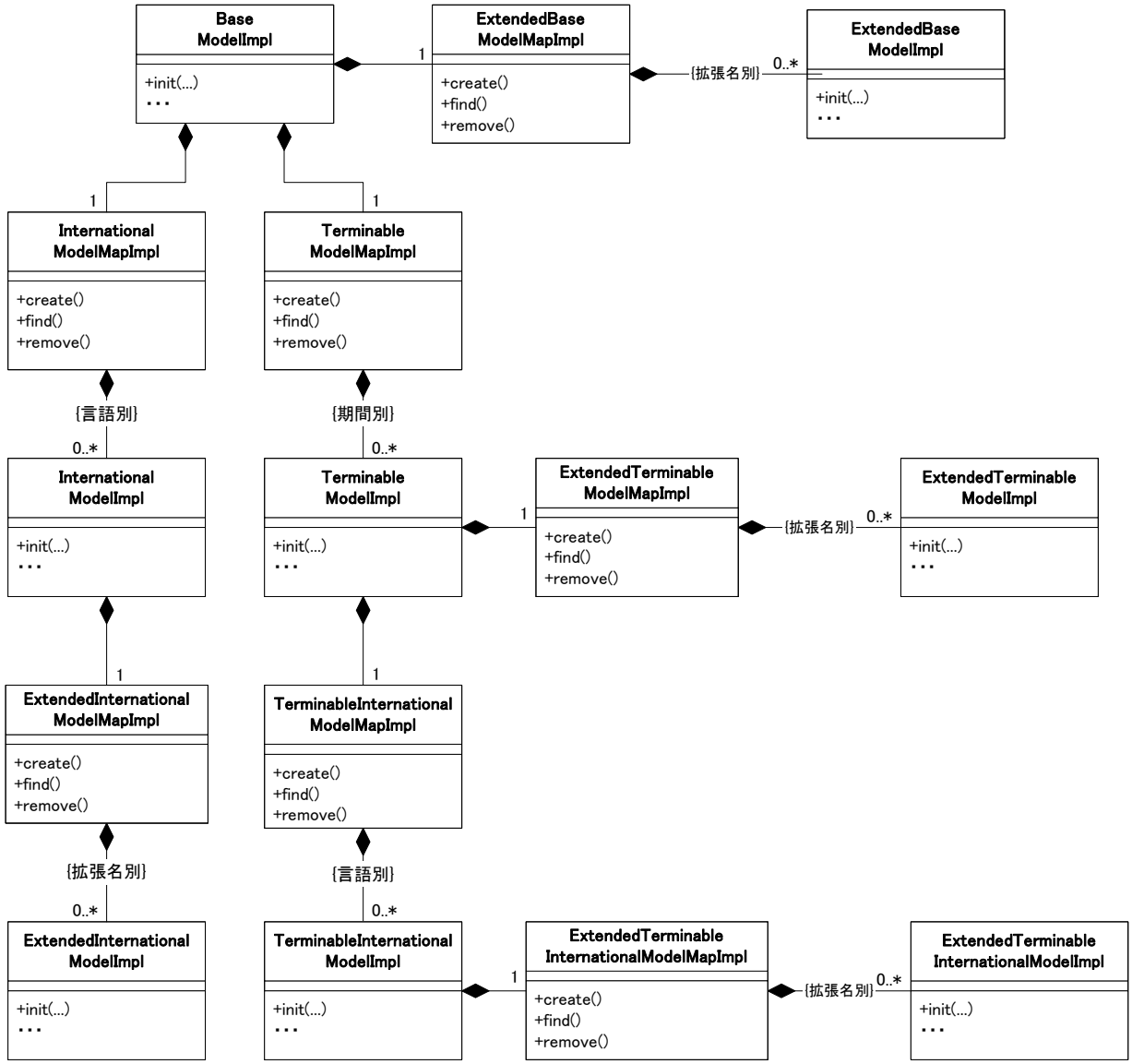


図 3-14 モデルのクラス図（実装例：1）

モデルはエンティティの構造にしたがって自由にプロパティにアクセスするためのメソッドを追加できる。これらのメソッドはアクセサに対して処理を委譲する（「図 3-15 モデルのクラス図（実装例：2）」を参照）。

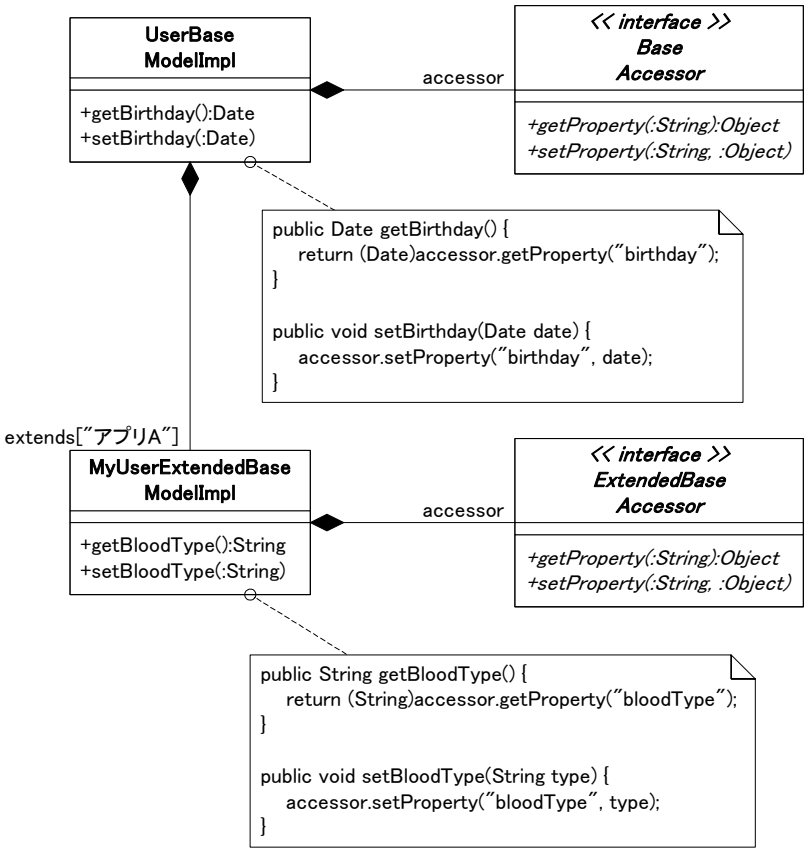


図 3-15 モデルのクラス図(実装例:2)

モデルとアクセサの委譲関係を「表 3-2 モデルとアクセサの委譲関係」に示す。

表 3-2 モデルとアクセサの委譲関係

モデル	アクセサ
BaseModel	BaseAccessor
InternationalModel	InternationalAccessor
TerminableModel	TerminableAccessor
TerminableInternationalModel	TerminableInternationalAccessor
InternationalModelMap	InternationalAccessorMap
TerminableModelMap	TerminableAccessorMap
TerminableInternationalModelMap	TerminableInternationalAccessorMap
ExtendedBaseModel	ExtendedBaseAccessor
ExtendedInternationalModel	ExtendedInternationalAccessor
ExtendedTerminableModel	ExtendedTerminableAccessor
ExtendedTerminableInternationalModel	ExtendedTerminableInternationalAccessor
ExtendedBaseModelMap	ExtendedBaseAccessorMap
ExtendedInternationalModelMap	ExtendedInternationalAccessorMap
ExtendedTerminableModelMap	ExtendedTerminableAccessorMap
ExtendedTerminableInternationalModelMap	ExtendedTerminableInternationalAccessorMap

モデルに追加するメソッドの方針を「3.2.1.1 すべてのモデルに共通する構造」から「3.2.1.5 期間国際化モデルの構造」に示す。「図 3-16 モデルのクラス図(実装例:3)」もあわせて参照すること。

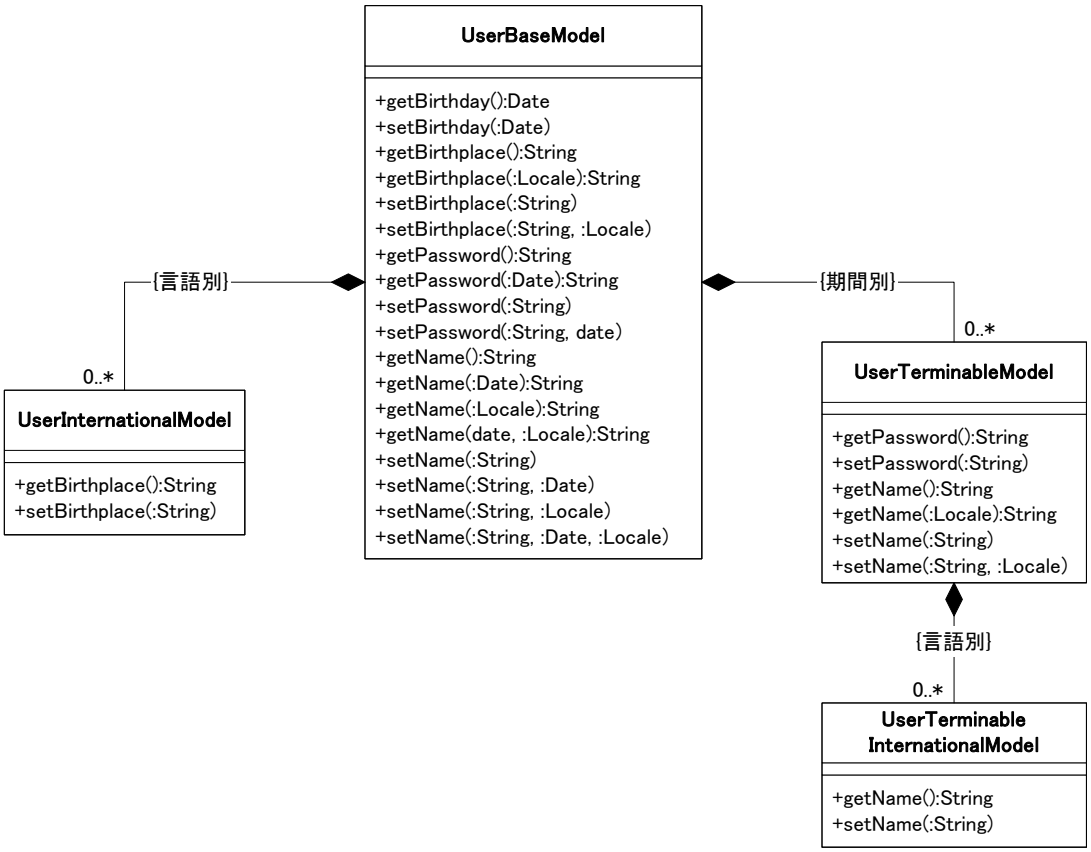


図 3-16 モデルのクラス図（実装例：3）

3.2.1.1 すべてのモデルに共通する構造

すべてのモデル（～Model）において共通するメソッドの方針を以下に記す。

- JavaBeans となるように実装する。
- そのモデルで管理するプロパティ（ロケールや有効期間は除く）に対するgetterメソッドを用意する。このメソッドは～AccessorのgetPropertyメソッドに処理を委譲する。「図 3-15 モデルのクラス図（実装例：2）」の例では、誕生日に該当するgetterメソッドとしてgetBirthdayが用意されている。この内部では、アクセサに対して属性名”birthday”に該当するプロパティを取得している。
- そのモデルで管理するプロパティ（ロケールや有効期間は除く）に対するsetterメソッドを用意する。ただし、プライマリキーである場合はこのメソッドは用意しない。このメソッドは～AccessorのsetPropertyメソッドに処理を委譲する。「図 3-15 モデルのクラス図（実装例：2）」の例では、誕生日に該当するsetterメソッドとしてsetBirthdayが用意されている。この内部では、アクセサに対して属性名”birthday”に該当するプロパティを設定している。
- プライマリキーに該当するプロパティに対する getter メソッドは BaseModel のみに実装する。

アクセサとモデルで同一のプロパティに対する型は一致させる必要はないが、その場合は型のミスマッチに起因する不具合に注意する必要がある。たとえば、数量のプロパティをアクセサでは java.lang.Integer で扱い、モデルではこのプロパティを int で扱うこととしその getter メソッドを public int getQuantity()（内部では Integer.intValue() などを使用）とすることはできるが、新規にモデルを生成してこのメソッドを呼び出したときは java.lang.NullPointerException が発生する場合がある。

3.2.1.2 基本モデルの構造

基本モデル（～BaseModel）におけるメソッドの方針を以下に記す。

- InternationalModel で管理されるプロパティに対する getter と setter メソッドを用意する。

- ◆ 引数に言語(ロケール)をとるものは、指定された言語(ロケール)に対する InternationalModel の該当プロパティにアクセスする。
- ◆ 引数に言語(ロケール)をとらないものは、BaseModel の init メソッドで渡されたデフォルト言語(ロケール)に対する InternationalModel の該当プロパティにアクセスする。

■ TerminableModel で管理されるプロパティに対する getter と setter メソッドを用意する。

- ◆ 引数に期間をとるものは、指定された期間に対する TerminableModel の該当プロパティにアクセスする。
- ◆ 引数に期間をとらないものは、BaseModel の init メソッドで渡されたデフォルト日時に対する TerminableModel の該当プロパティにアクセスする。

■ TerminableInternationalModel で管理されるプロパティに対する getter と setter メソッドを用意する。

- ◆ 引数に期間と言語(ロケール)の両者をとるものは、指定された期間と指定された言語(ロケール)に対する TerminableInternationalModel の該当プロパティにアクセスする。
- ◆ 引数に期間のみをとり言語(ロケール)をとらないものは、指定された期間と BaseModel の init メソッドで渡されたデフォルト言語(ロケール)に対する TerminableInternationalModel の該当プロパティにアクセスする。
- ◆ 引数に言語(ロケール)のみをとり期間をとらないものは、指定された言語(ロケール)と BaseModel の init メソッドで渡されたデフォルト日時に対する TerminableInternationalModel の該当プロパティにアクセスする。
- ◆ 引数に期間も言語(ロケール)もとらないものは、BaseModel の init メソッドで渡されたデフォルト日時とデフォルト言語(ロケール)に対する TerminableInternationalModel の該当プロパティにアクセスする。

■ 指定された拡張名に対する ExtendedBaseModel を取得するメソッドを用意する。

3.2.1.3 国際化モデルの構造

国際化(～InternationalModel)モデルにおけるメソッドの方針を以下に記す。

■ 指定された拡張名に対する ExtendedInternationalModelMap を取得するメソッドを用意する。

3.2.1.4 期間化モデルの構造

期間化モデル(～TerminableModel)におけるメソッドの方針を以下に記す。

■ TerminableInternationalModel で管理されるプロパティに対する getter と setter メソッドを用意する。

- ◆ 引数に言語(ロケール)をとるものは、指定された言語(ロケール)に対する TerminableInternationalModel の該当プロパティにアクセスする。
- ◆ 引数に言語(ロケール)をとらないものは、BaseModel の init メソッドで渡されたデフォルト言語(ロケール)に対する TerminableInternationalModel の該当プロパティにアクセスする。

■ 指定された拡張名に対する ExtendedTerminableModelMap を取得するメソッドを用意する。

3.2.1.5 期間国際化モデルの構造

期間国際化モデル(～TerminableInternationalModel)におけるメソッドの方針を以下に記す。

■ 指定された拡張名に対する ExtendedTerminableInternationalModelMap を取得するメソッドを用意する。

3.2.1.6 すべての拡張モデルに共通する構造

すべての拡張モデル(Extended～Model)において共通するメソッドの方針を以下に記す。

■ JavaBeans となるように実装する。

■ その拡張モデルで管理するプロパティに対するgetterメソッドを用意する。このメソッドは属性名をプロパティ名として扱うJavaBeansのgetterメソッドである必要がある。このメソッドはExtended～Accessorのgetpropertyメソッドに処理を委譲する。「図 3-15 モデルのクラス図(実装例:2)」の例では、血液型に該当するgetterメソッドとしてgetBloodTypeが用意されている。この内部では、アクセサに対して属性名"bloodType"に該当するプロパティを取得している。

- そのモデルで管理するプロパティに対するsetterメソッドを用意する。このメソッドは属性名をプロパティ名として扱うJavaBeansのsetterメソッドである必要がある。このメソッドはExtended～AccessorのsetPropertyメソッドに処理を委譲する。「図 3-15 モデルのクラス図（実装例：2）」の例では、血液型に該当するsetterメソッドとしてsetBloodTypeが用意されている。この内部では、アクセサに対して属性名”bloodType”に該当するプロパティを設定している。

3.2.2 マッパ

マッパはエンティティの内容をモデルのインスタンスに対応させる。データストアからのデータの取得や反映はマッパを通じて行う。

3.2.2.1 構成

マッパは以下の要素から構成される。

- MappaFactory
  マッパを生成する。
- Mapper
  データストアとのやり取りを行う。

MapperFactory、Mapper、BaseModelの関連を「図 3-17 マッパのクラス図」に示す。

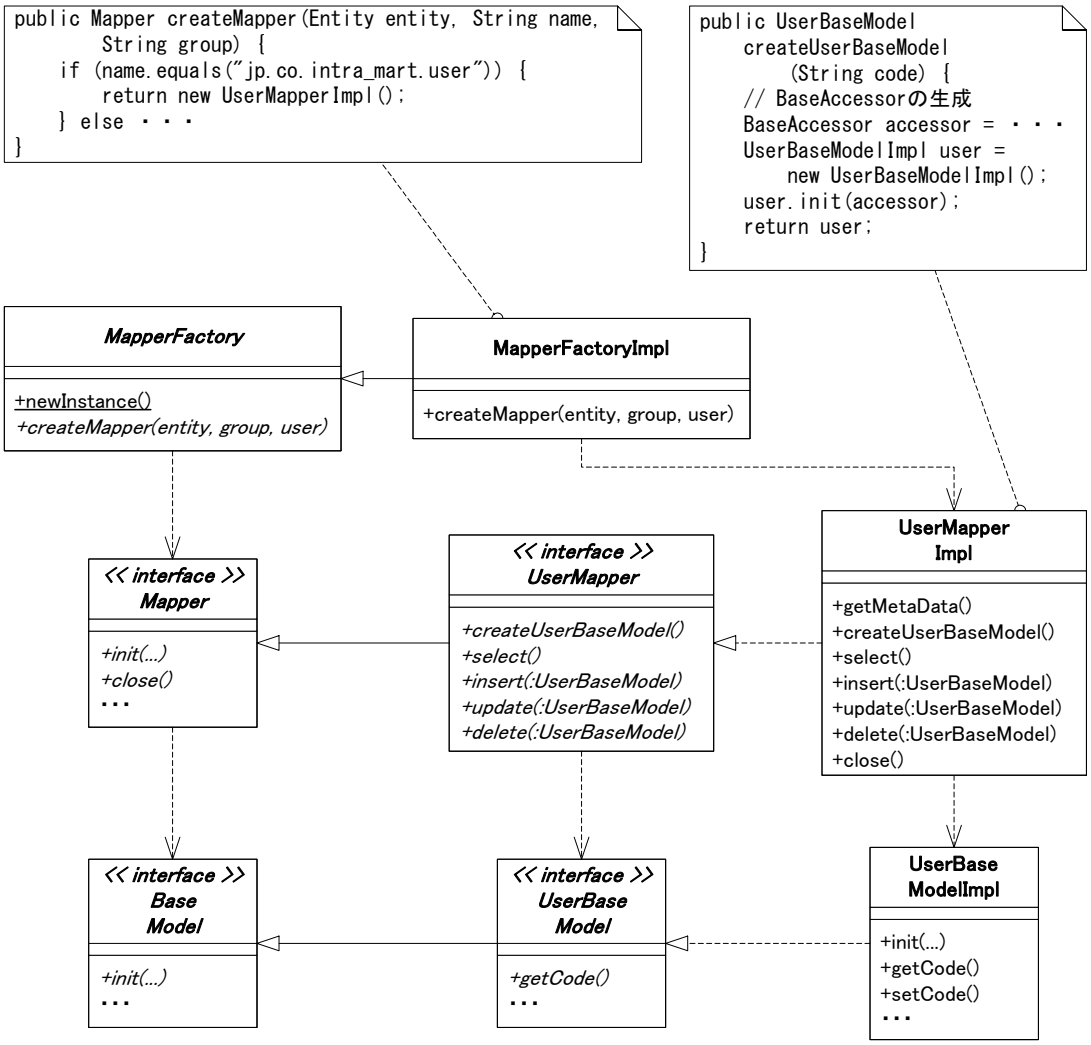


図 3-17 マッパのクラス図

3.2.2.1.1 MapperFactory

MapperFactory は Mapper を生成することを目的としている。MapperFactory はシングルトンであり、同一の Java VM 上では最大 1 つだけ存在する。

Jp.co.intra_mart.foundation.datastore.common.model.MapperFactoryは抽象クラスである。MapperFactoryの具象クラスのインスタンスを取得する場合、MapperFactoryのnewInstanceメソッドを使用する。このメソッドはjp.co.intra_mart.foundation.datastore.applicationで取得されるリソースバンドルのプロパティ”factory”で決定されるMapperFactoryのサブクラスのインスタンスを返す。Intra-martを標準でインストールした場合、この設定は既に行われている。このときのMapperFactoryは「3.4 設定」で説明されている設定情報からMapperのクラス名を取得し、そのインスタンスを生成するような実装がされている。

3.2.2.1.2 Mapper

Mapper はデータストアとやり取りをすることを目的としている。Mapper 自身はデータストアとやり取りするために Accessor を提供し、エンティティを定義した開発者が作成した Model と関連付ける。このようにすることでアプリケーションは Mapper と Model の知識だけでデータストアにアクセスすることができる。

Mapper は以下の条件をすべて満たしている必要がある。

- インタフェースjp.co.intra_mart.foundation.datastore.common.model.Mapperを実装している具象クラスである。
- デフォルトコンストラクタ（public で引数が 1 つもないコンストラクタ）が存在する。

Mapper はエンティティごとに用意する必要がある。この場合 Mapper インタフェースを直接実装してもよいが、エンティティに対応したマッパのインタフェースと実装を別にすることを推奨する。こうすることで、データストアの形式が変わった場合でも設定を変更するだけでよく、マッパを利用するアプリケーションには修正を加える必要がない。

データストアとしてintra-martで管理されているリレーショナルデータベースを使用する場合、intra-martに標準で用意されているjp.co.intra_mart.foundation.datastore.common.model.IntramartDBMapperを継承したマッパを利用することを推奨する（「図 3-18 IntramartDBMapperの利用」を参照）。

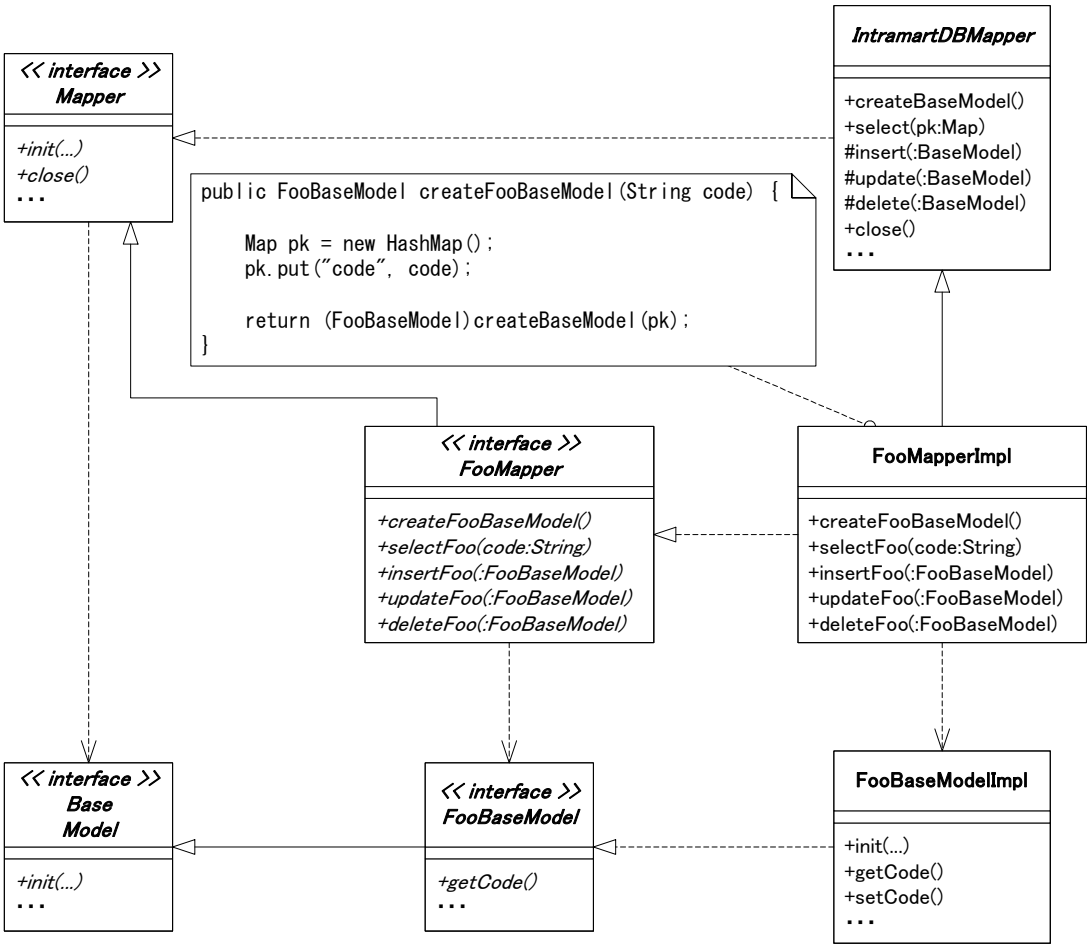


図 3-18 IntramartDBMapper の利用

「図 3-18 IntramartDBMapperの利用」ではエンティティFooに対してマッパFooMapperとモデルFooBaseModelをインタフェースとして用意している。アプリケーションはこれらのインタフェースを利用するように製造する。一方、マッパの実装であるFooMapperImplはFooMapperインタフェースを実装すると同時にIntramartDBMapperを継承している。FooMapperインタフェースではFooBaseModelに特化したメソッドを用意することでアプリケーションからは余計なダウンキャストが必要なくなるようになっている。また、FooMapperImplの実際の処理はIntramartDBMapperに処理を依頼することで簡略化されている。

3.2.2.2 マッパの利用

マッパを利用してデータストアにアクセスする場合、次のような手順をとる。

(1) トランザクションを開始する。
(2) MapperFactory を取得する。
(3) MapperFactory から Mapper を取得する。
(4) Mapper から取得される BaseModel を通じてデータストアにアクセスする。
(5) Mapper をクローズする。
(6) トランザクションを終了する。

この様子を「図 3-19 マッパの利用」に示す。

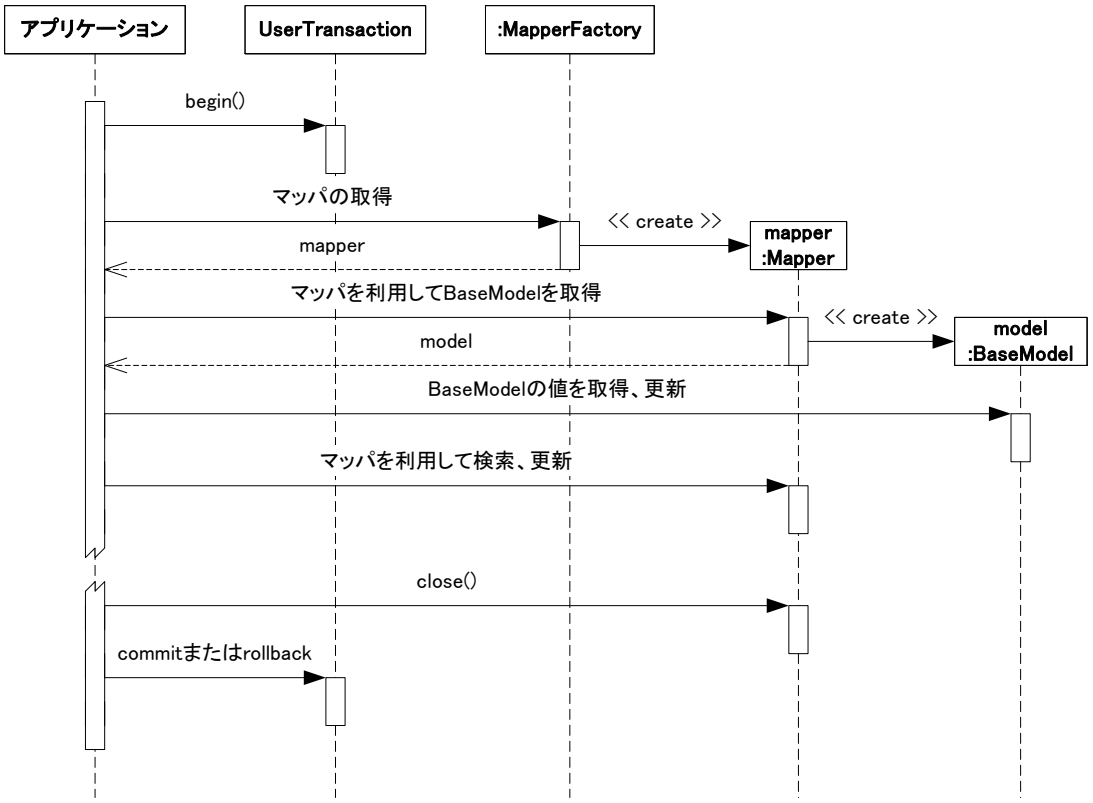


図 3-19 マッパの利用

3.2.2.3 マッパの取得

マッパを取得するとき、次のような手順をとる。

(1) MapperFactory の newInstance メソッドを呼び、MapperFactory の具象クラスのインスタンスを取得する。

(2) (1)で取得した MapperFactory の createMapper メソッドでマッパを取得する。

この様子を「図 3-20 マッパの取得」に示す。

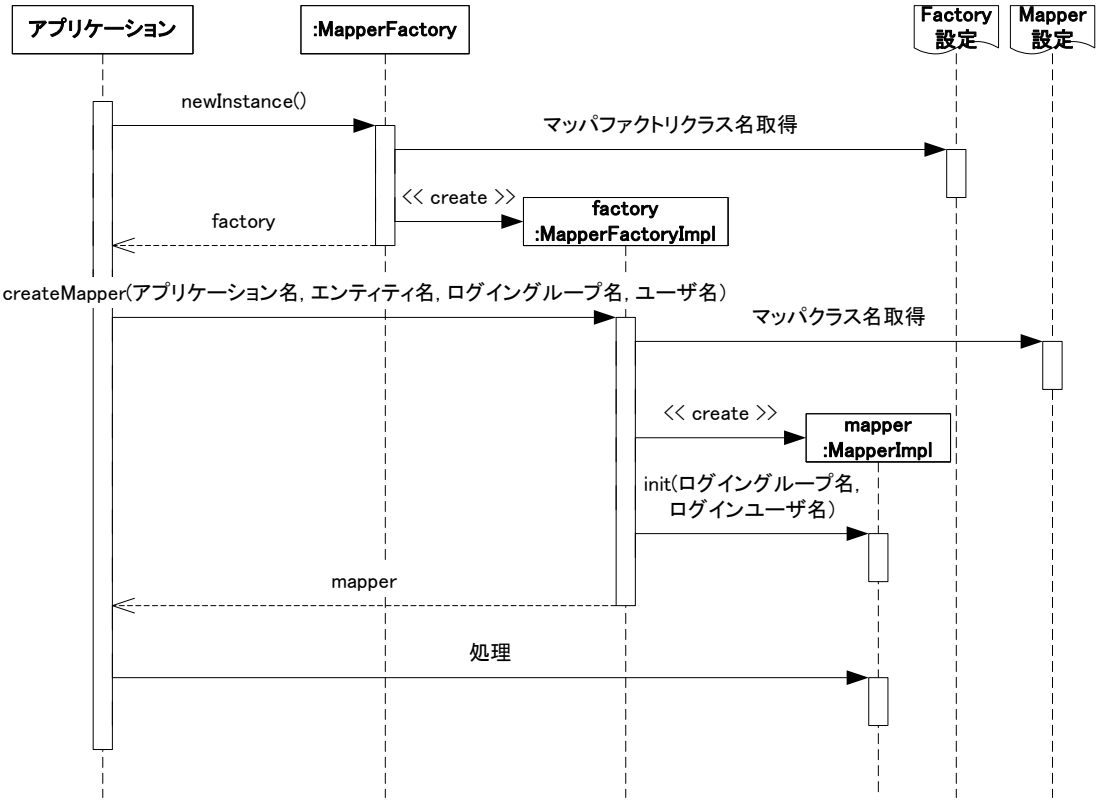


図 3-20 マッパの取得

3.2.2.4 モデルの検索

検索を行うためにはデータストアに対して直接アクセスするか、またはマッパの select メソッドを利用する。Select メソッドを利用した場合、指定したプライマリキーに該当する BaseModel が返される。マッパの select メソッドで検索を行う場合は以下の手順に従う。

(1) 検索対象の BaseModel のプライマリキーに該当する Map を生成する。

(2) 検索対象のマッパの select メソッドに対して Map を引数に渡して検索する。

Selectメソッドによる検索の様子を「図 3-21 selectメソッドによる検索」に示す。

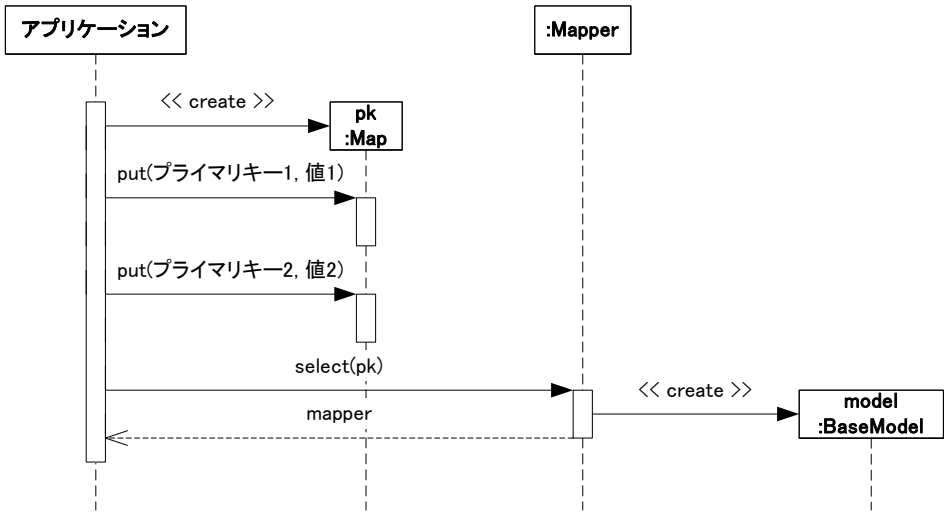


図 3-21 select メソッドによる検索

「図 3-21 selectメソッドによる検索」でselectメソッドの引数に渡すjava.util.Mapは、キーにプライマリキーの属性名、値に検索する値を指定する必要がある。このMapがプライマリキーとして不完全な場合は例外が返される。

3.2.2.5 モデルの生成

データストアに値するデータのやり取りする最小単位は基本モデルである。データストアに新規にデータを追加する場合、マッパを通じて基本モデルを生成しなければならない。

3.2.2.5.1 基本モデルの生成

基本モデルの生成はアプリケーションから明示的に行ってはならない。代わりに、マッパのcreateBaseModelメソッドを使用して新規に基本モデルを生成する。このメソッドは引数にプライマリキーに該当するマップを渡す。マップは項目名をキーとする値を持っている必要がある。マップを通じて基本モデルを新規に生成する様子の概要を「図 3-22 モデルの生成(概要)」に示す。

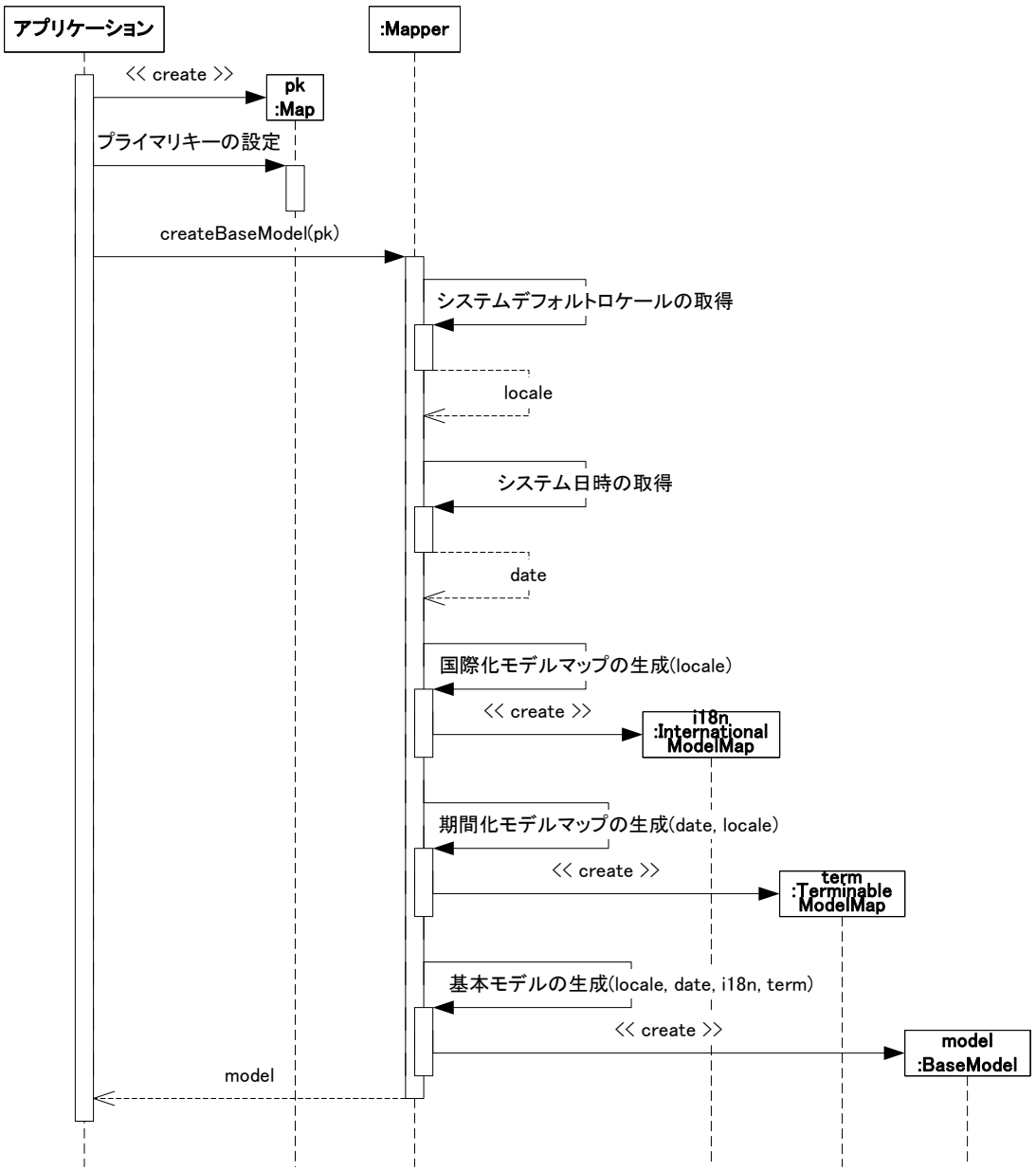


図 3-22 モデルの生成(概要)

Mapper の createBaseModel メソッドによって生成される BaseModel は以下のような常態になっている。

- BaseModel に対応する BaseAccessor はプライマリキーに該当する属性に対してのみ値が入っている。
- 国際化モデルが設定されている場合、ログインユーザのデフォルトの言語（ロケール）に該当する InternationalModel のみ存在する。
- 期間化モデルが設定されている場合、有効最小日時から有効最大日時までを対象とする期間に該当する TerminableModel のみ存在する。
- 期間国際化モデルが設定されている場合、有効最小日時から有効最大日時までを対象とする期間であり、かつログインユーザのデフォルトの言語（ロケール）に該当する TerminableInternationalModel のみ存在する。
- 基本モデルに対応する BaseAccessor 、国際化モデルに対応する InternationalAccessor、期間化モデルに対応する TerminableAccessor および期間国際化モデルに対応する TerminableInternationalAccessor は、各アクセサの setProperty メソッドによって属性に値が設定される前にそれぞれのアクセサの getProperty メソッドが呼び出された場合、jp.co.intra_mart.foundation.datastore.exception.DataNotFoundException を throw する。ただし、BaseAccessor ではプライマリキーに対応する属性は最初から設定されているため、この場合のみ正しい値を返す。

「図 3-22 モデルの生成（概要）」において“国際化モデルマップの生成”、“期間化モデルマップの生成”、期間化モデルマップ生成中の期間国際化モデルマップの生成および“基本モデルの生成”とある部分の詳細を、それぞれ「図 3-23 国際化モデルの生成」、「図 3-24 期間化モデルの生成」、「図 3-25 期間国際化モデルの生成」および「図 3-26 基本モデルの生成」に示す。

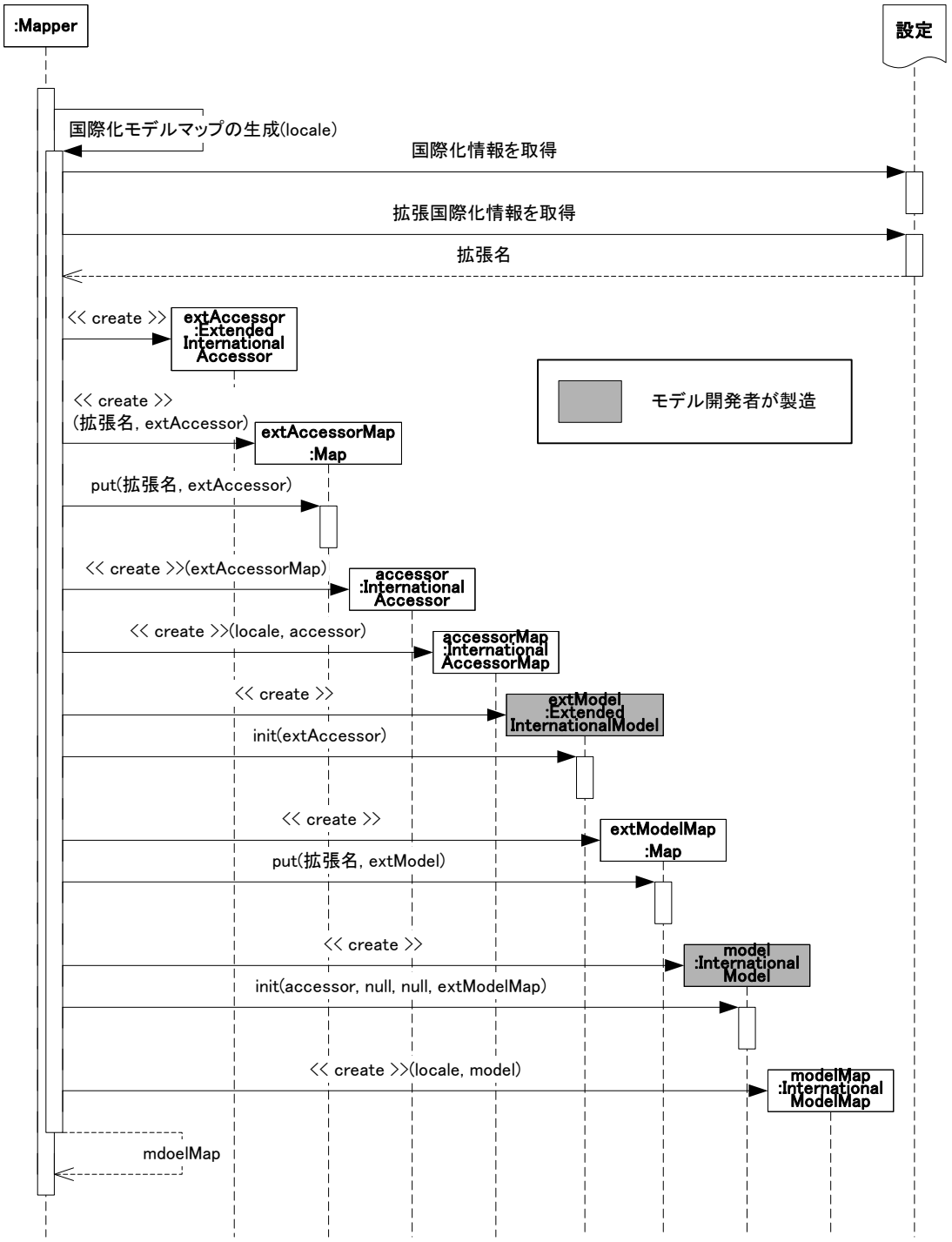


図 3-23 国際化モデルの生成

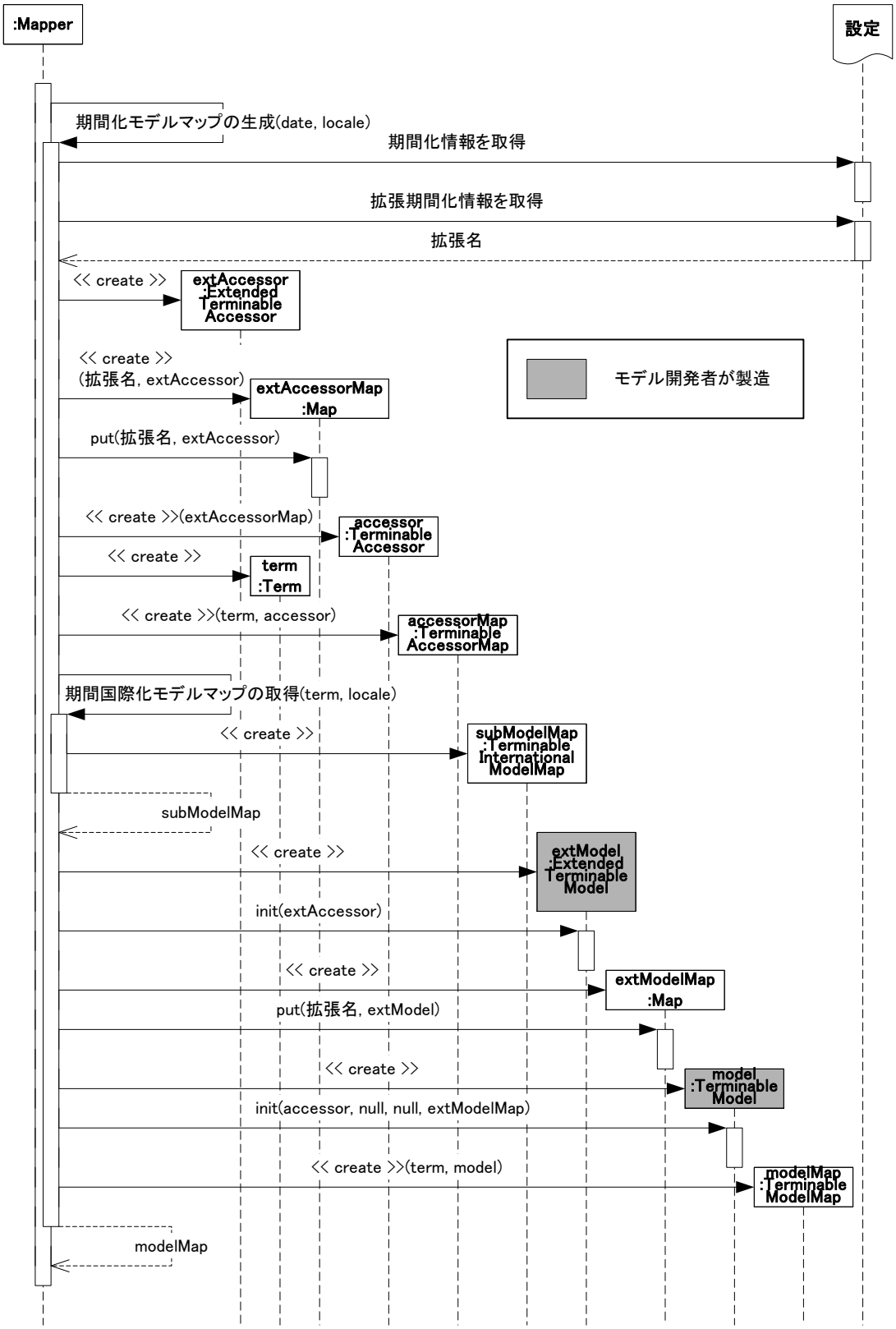


図 3-24 期間化モデルの生成

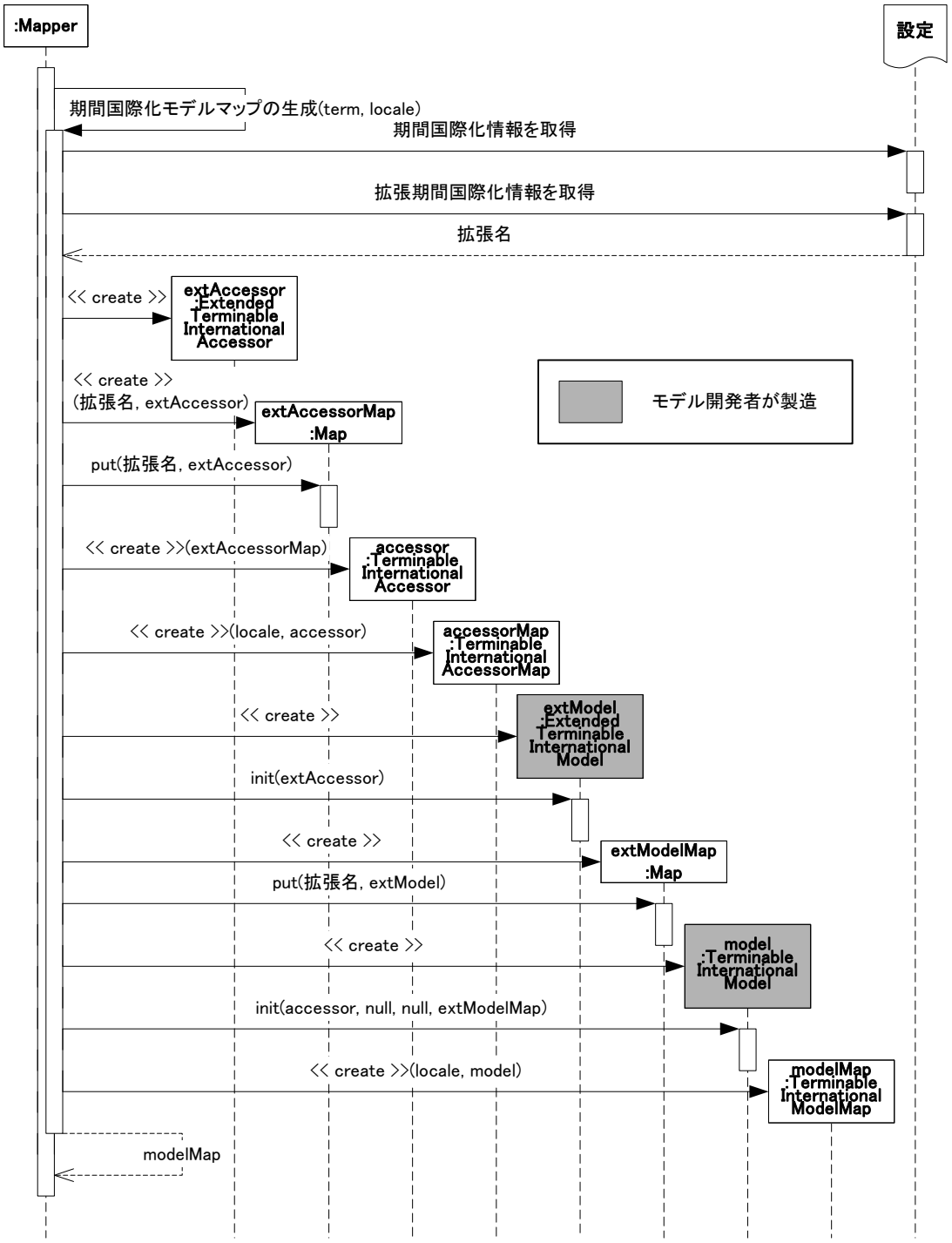


図 3-25 期間国際化モデルの生成

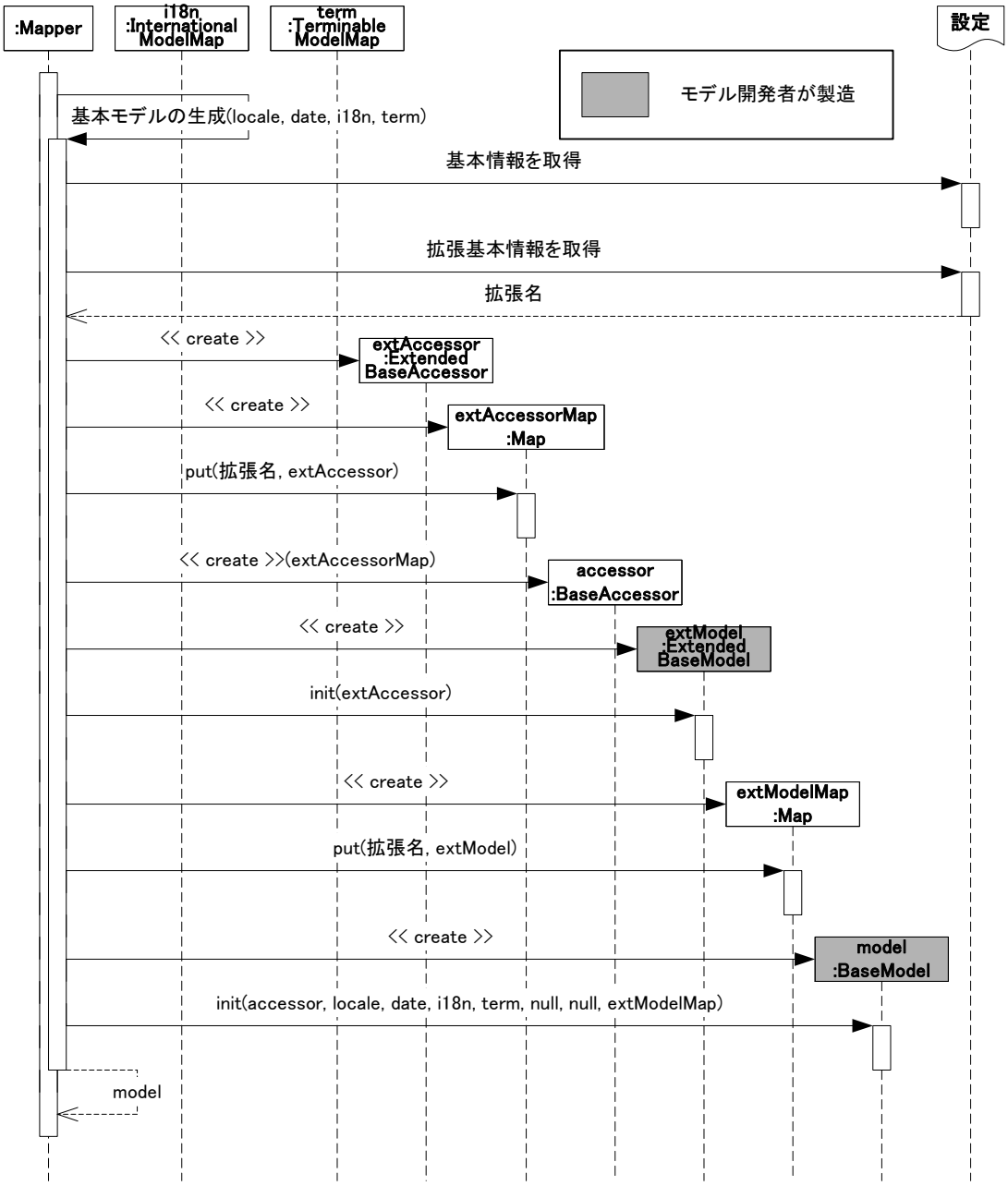


図 3-26 基本モデルの生成

3.2.2.6 モデルの挿入

エンティティに対する新しいインスタンスの挿入は基本モデルの単位で行う。挿入するときは次のような手順をとる。

1. マッパを通じて基本モデルを新規に生成する。
2. 基本モデルに対してデータを設定する。
3. データを設定した基本モデルを、マッパを通じて登録する。

この概要を「図 3-27 基本モデルの挿入(概要)」に示す。

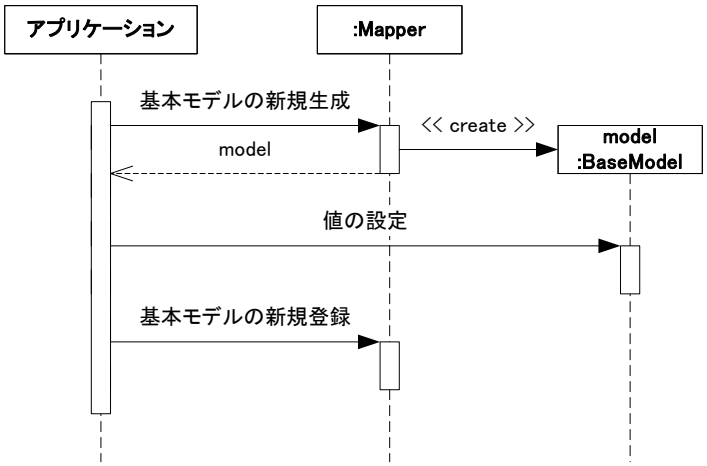


図 3-27 基本モデルの挿入（概要）

3.2.2.6.1 基本モデルの生成

基本モデルの生成はマッパを通じて行う。基本モデル生成の詳細については「3.2.2.5 モデルの生成」を参照すること。

3.2.2.6.2 基本モデルの挿入

基本モデルを挿入するときはマッパのinsertメソッドを使用する。このメソッドは引数に基本モデルを指定する。この基本モデルは、「3.2.2.6.1 基本モデルの生成」で生成され、値が設定された基本モデルである。この基本モデルは同一マッパのインスタンスのcreateBaseModelメソッドで生成されたものでなければならない。同じ種類であっても別インスタンスのマッパから生成された基本モデルをinsertメソッドの引数に指定することはできない。同様に、同一インスタンスのマッパであってもcreateBaseModelメソッド以外の方法（selectメソッド等）によって生成された基本モデルもinsertメソッドの引数に指定することはできない。

マップを通じて基本モデルを挿入する様子を「図 3-28 モデルの挿入」に示す。

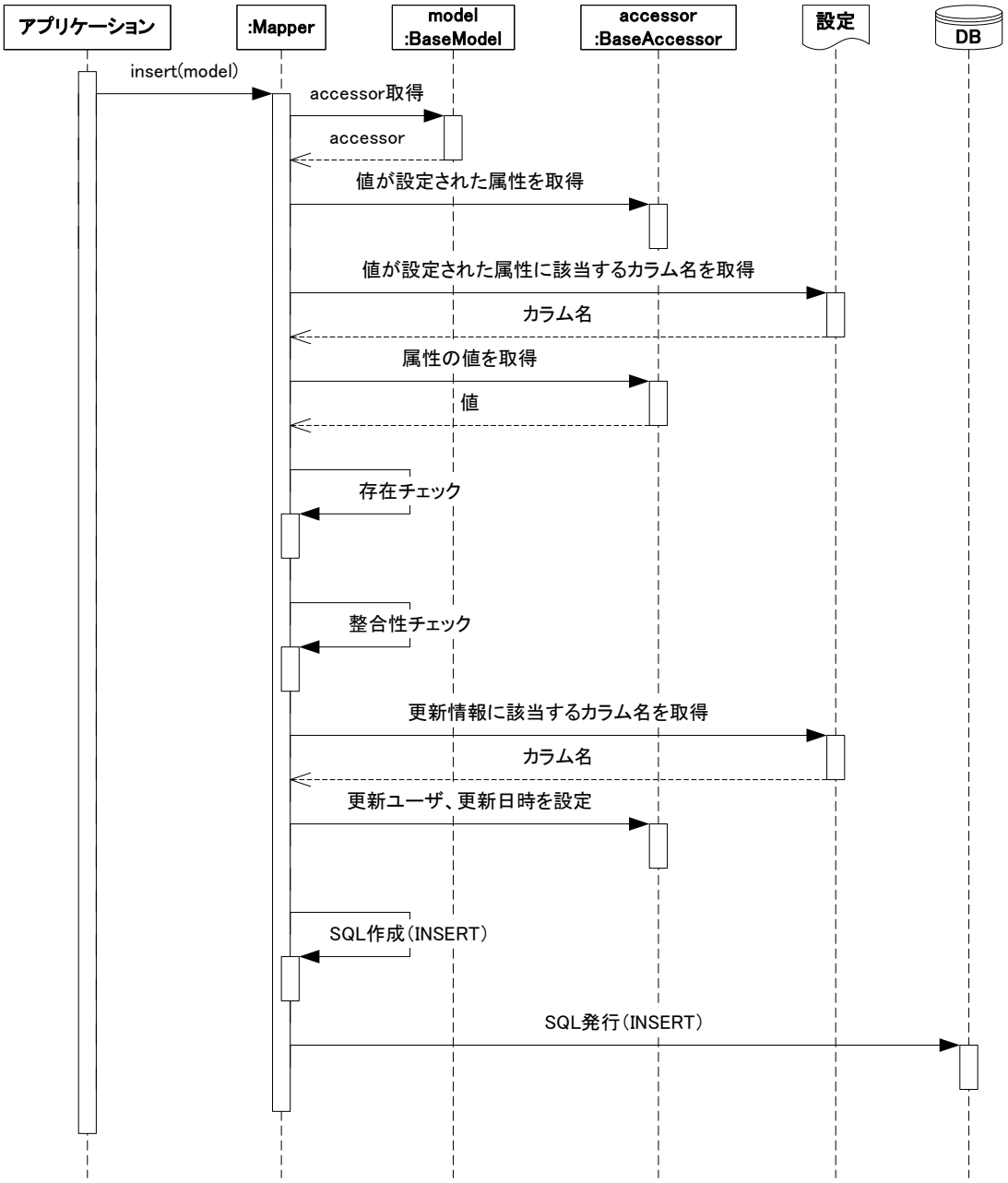


図 3-28 モデルの挿入

「図 3-28 モデルの挿入」ではデータストアがリレーショナルデータベースであることを想定しているが、他の種類のデータストアであってもかまわない。その場合、SQL作成とSQL発行の部分がデータストアに特化した方法で基本モデルの内容を永続化することになる。

「図 3-28 モデルの挿入」で行っている存在チェックについては「3.2.2.6.3 存在チェック」を、整合性チェックについては「3.2.2.6.4 リレーションシップによる参照先の整合性のチェック（挿入時）」を参照すること。

基本モデルを挿入するサンプルを「リスト 3-1 挿入のサンプルプログラム」に示す。

リスト 3-1 挿入のサンプルプログラム

```
1: UserTransaction ut = ・・・;
2: ut.begin();
```

```
 3: MapperFactory factory = MapperFactory.newInstance();
 4: UserMapper mapper =
 5:       (UserMapper)factory.createMapper("jp.co.intra_mart.app", "User",
 6:                                        "DEFAULT", "admin");
 7: UserBaseModel user = mapper.createUserBaseModel("u0002");
 8: user.setBirthday(new Date("1990/02/02"));
 9: UserInternationalModel userI18n =
10:       user.getUserInternationalModel(Locale.JAPANESE);
11: userI18n.setBirthPlace("日本");
12: TerminableModelMap userTermMap = user.getTerminableModelMap();
13: UserTerminableModel userTerm =
14:      (UserTerminableModel)userTermMap.createTerminableModel(new Term());
15: userTerm.setPassword("xxxxxx");
16: UserTerminableInternationalModel userTermI18n =
17:       userTerm.getUserTerminableInternationalModel(Locale.JAPANESE);
18: userTermI18n.setName("ユーザ 0002");
19: mapper.insert(user);
20: mapper.close();
21: ut.commit();
```

3.2.2.6.3 存在チェック

基本モデルを新規に挿入するとき、同一のプライマリキーで既にデータが登録されていないかどうかをチェックする必要がある。この処理の概要を「図 3-29 挿入時の存在チェック」に示す。

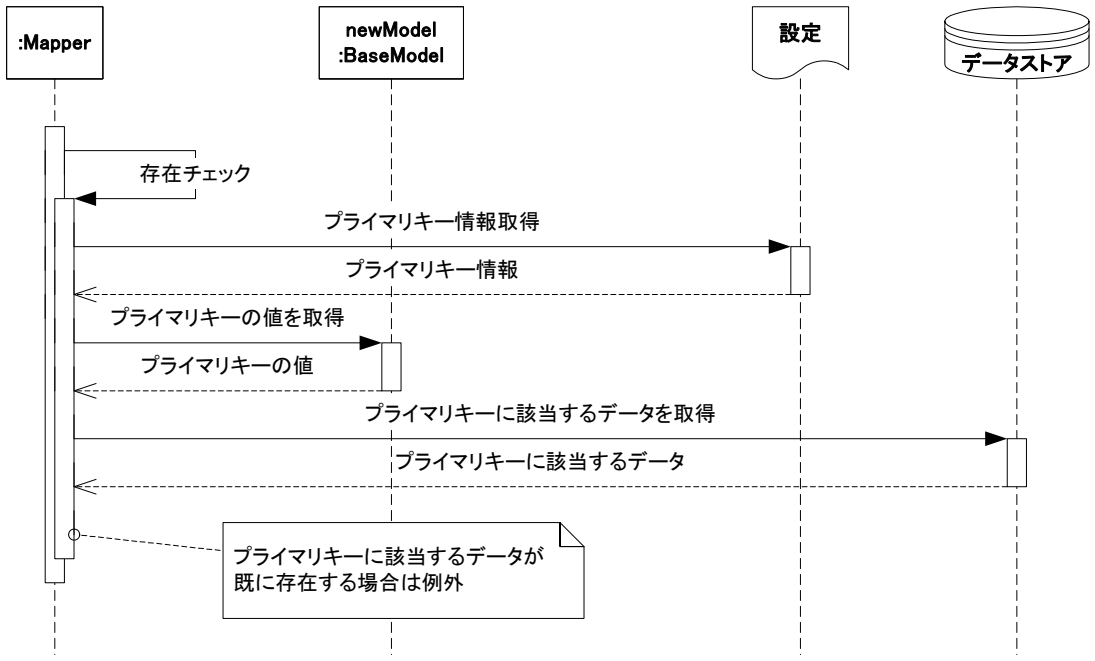


図 3-29 挿入時の存在チェック

存在チェックの実際の処理はデータストアの種類と Mapper の実装に依存する。

3.2.2.6.4 リレーションシップによる参照先の整合性のチェック(挿入時)

以下のような条件をすべて満たす場合、データ挿入時には整合性チェックを行う。

- 自分自身がソースエンティティとなる(自分自身もしくは他のエンティティを参照する)ようなリレーションシップが存在する
- 外部キーに該当する項目すべてに値(null 以外)が設定されている

上記の条件をすべて満たす場合、参照先のエンティティには外部キーに該当するデータが存在する必要がある。このときに行われる整合性のチェック内容を「図 3-30 挿入時の整合性チェック(共通)」に示す。

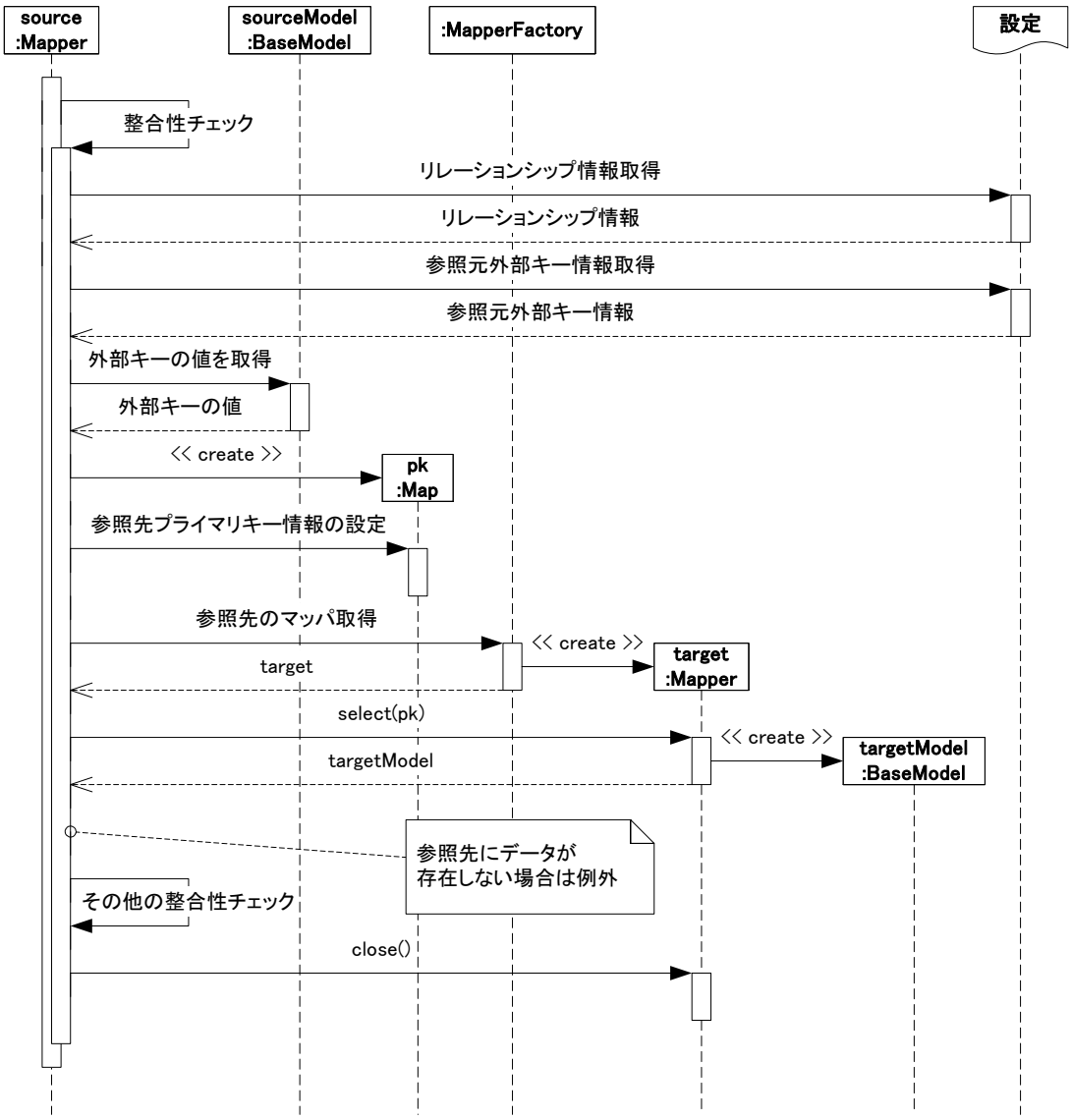


図 3-30 挿入時の整合性チェック(共通)

「図 3-30 挿入時の整合性チェック(共通)」では参照先のデータの存在チェックに参照先のMapperのselect(java.util.Map)メソッドを明示的に使用しているが、Mapperの実装によっては最適化のために他の方法で存在チェックをしている場合もある。この方法はデータストアの種類とMapperの実装に依存する。

リレーションシップで以下の項目が定義されている場合、「図 3-30 挿入時の整合性チェック(共通)」で示した整合性検査を行った後さらに詳細なチェックを行う。

- 国際化基準項目(I18n)
- 期間化基準項目(Term)
- 期間の関連付け(Lifetime)

チェック内容の一覧を「表 3-3 挿入時の整合性チェック(項目別)」に示す。

表 3-3 挿入時の整合性チェック(項目別)

リレーションシップ			チェック
国際化基準項目 （I18n） ○:あり、×:なし	期間化基準項目 （Term） ○:あり、×:なし	ライフタイム （Lifetime） ○:none 以外、×:none	
×	×	―	他の整合性チェックは不要
○	×	―	パターン 1
×	○	―	パターン 2
○	○	―	パターン 3
―	―	○	パターン 4

以下にそれぞれのパターンにおける個別のチェック内容の詳細を記述する。

3.2.2.6.4.1 パターン 1（国際化のみ）

ターゲットエンティティの基本モデルに、国際化基準項目で指定される言語（ロケール）に該当する国際化モデルが存在するかチェックする。チェックする様子を「図 3-31 挿入時の整合性チェック（国際化）」に示す。

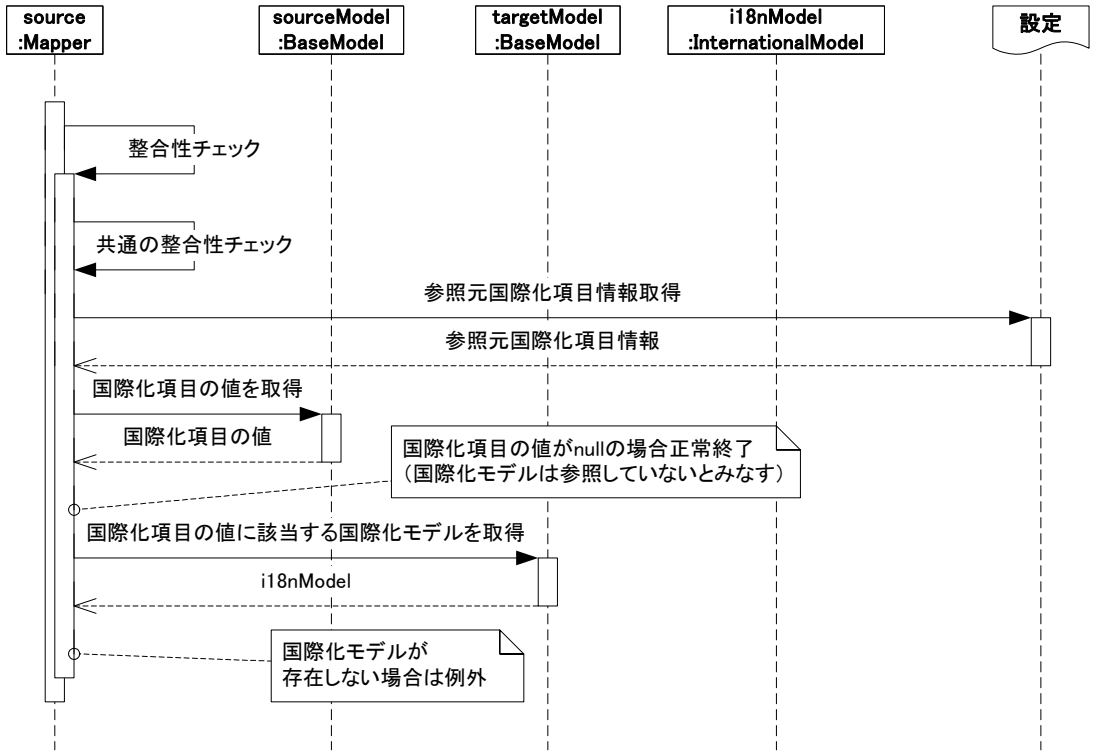


図 3-31 挿入時の整合性チェック（国際化）

3.2.2.6.4.2 パターン 2（期間化のみ）

ターゲットエンティティの基本モデルに、期間化基準項目で指定される日時に該当する期間化モデルが存在するかチェックする。チェックする様子を「図 3-32 挿入時の整合性チェック（期間化）」に示す。

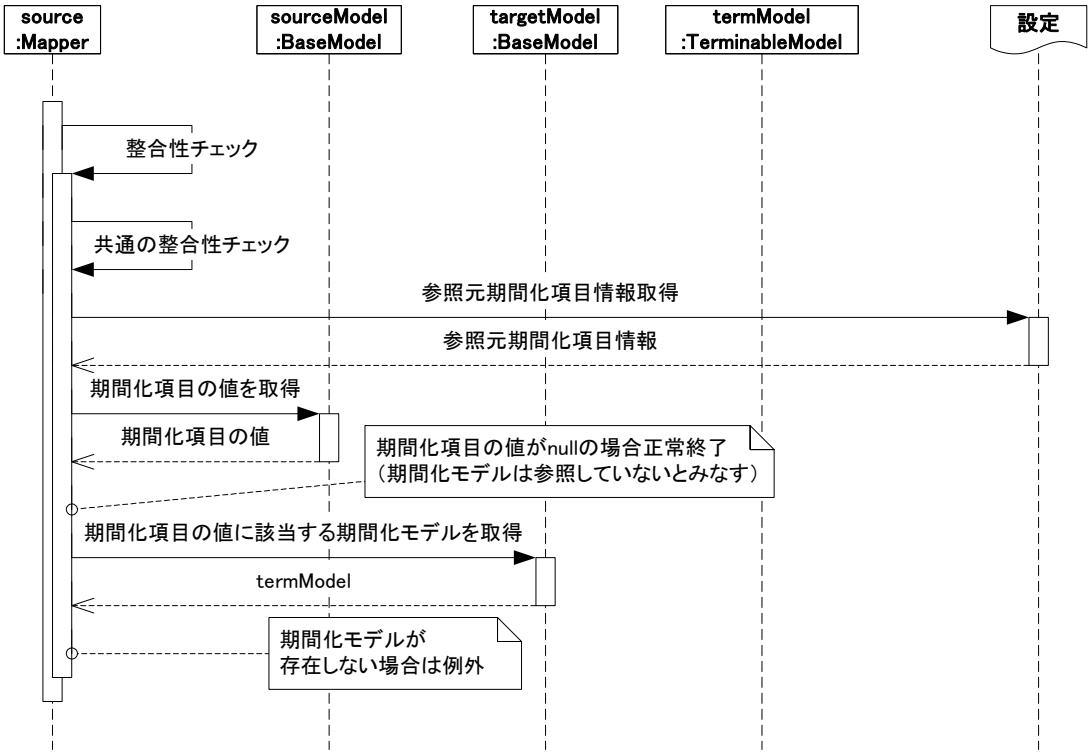


図 3-32 挿入時の整合性チェック（期間化）

3.2.2.6.4.3 パターン 3（国際化＋期間化）

ターゲットエンティティの基本モデルに、期間化基準項目で指定される日時に該当する期間化モデルが存在し、その期間化モデルに国際化基準項目で指定される言語（ロケール）に該当する期間国際化モデルが存在するかチェックする。チェックする様子を「図 3-33 挿入時の整合性チェック（国際化＋期間化）」に示す。

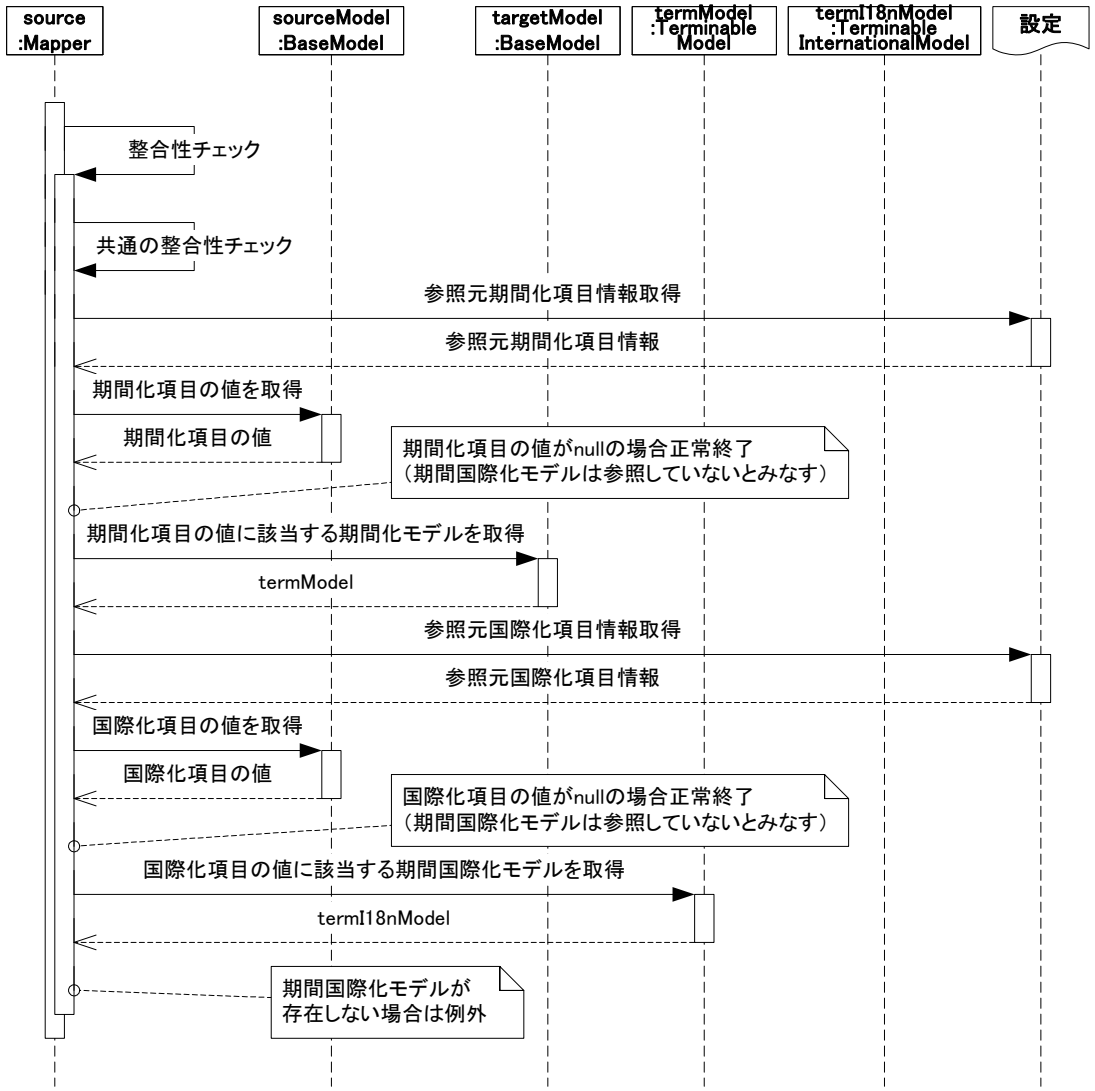


図 3-33 挿入時の整合性チェック（国際化＋期間化）

3.2.2.6.4.4 パターン 4（ライフタイム）

挿入する基本モデル内に存在するすべての期間化モデルの期間が、ターゲットエンティティの基本モデル内に存在するすべての期間化モデルの期間範囲内であるかチェックする。チェックする様子を「図 3-34 挿入時の整合性チェック（ライフタイム）」に示す。

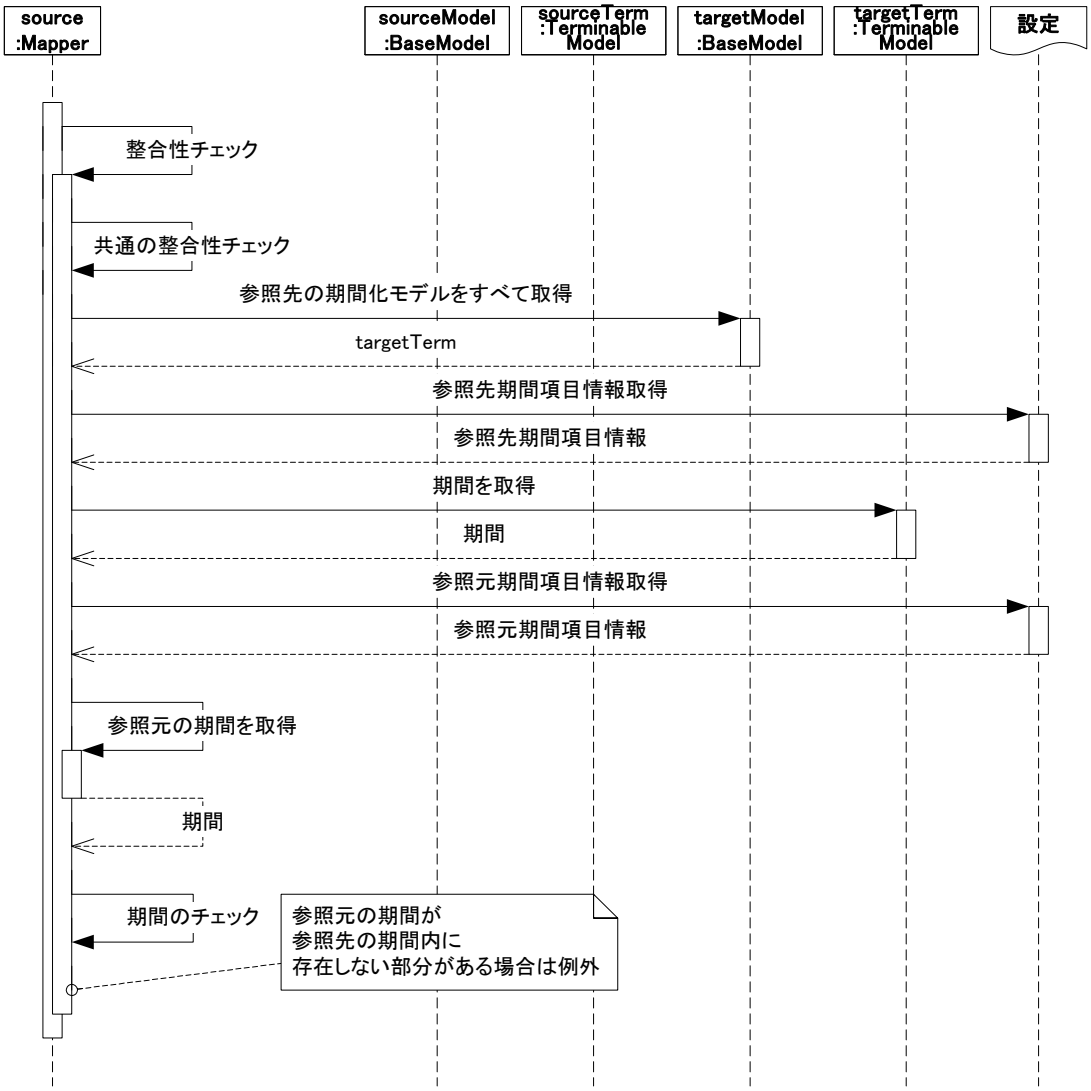


図 3-34 挿入時の整合性チェック(ライフタイム)

参照元の期間を取得する場合、ソースエンティティの期間設定状況によって取得方法が異なる。これらは以下のパターンに分けられる。

- ソースエンティティの期間化による管理
  ソースエンティティが期間化されていて、ライフタイム期間が設定されていない場合。
- ソースエンティティのライフタイム期間による管理
  ソースエンティティが期間化されてなく、ソースエンティティにライフタイム期間に該当する属性が設定されている場合。
- ソースエンティティの参照先のライフタイム期間による管理
  ソースエンティティが期間化されてなく、ソースエンティティからたどれる他のエンティティにライフタイム期間に該当する属性が設定されている場合。

3.2.2.6.4.4.1 ソースエンティティの期間化による管理

ソースエンティティが期間化されていて、ライフタイム期間が設定されていない場合、ソースエンティティで設定する期間はすべてターゲットエンティティで設定されている期間内であるかどうかチェックする必要がある。「図 3-35 参照元の期間の取得(ソースエンティティが期間化されている場合)」を参照すること。

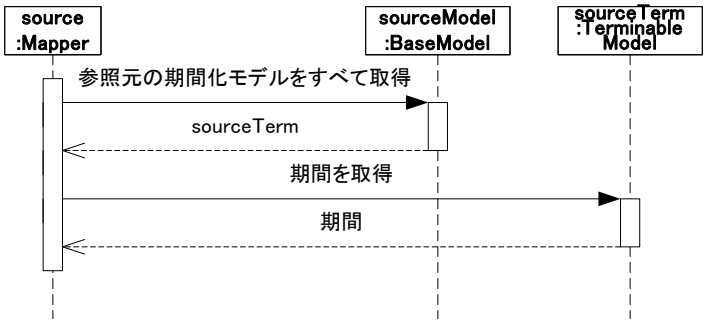


図 3-35 参照元の期間の取得（ソースエンティティが期間化されている場合）

ソースエンティティの期間化によってライフタイムが管理されている場合の整合性の例を「図 3-36 ソースエンティティの期間化によるライフタイム整合性の例」に示す。

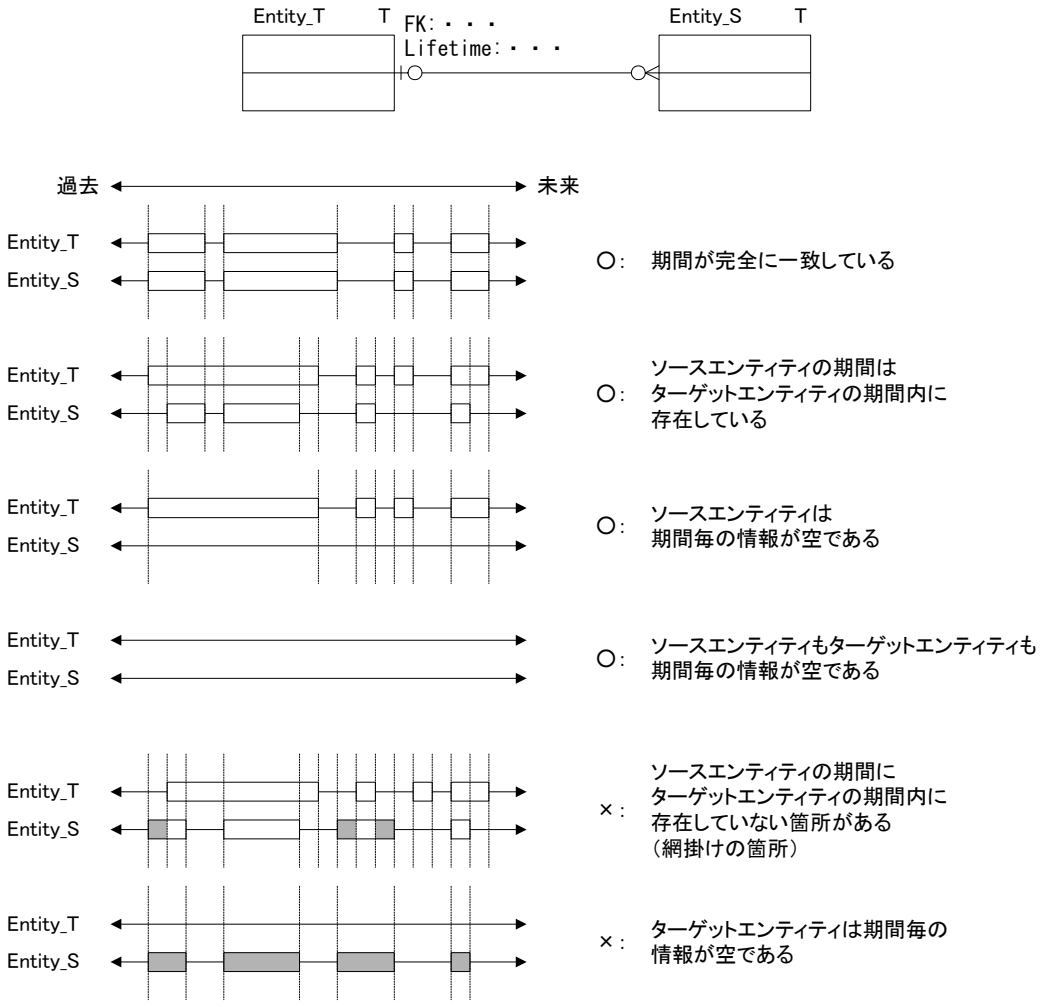


図 3-36 ソースエンティティの期間化によるライフタイム整合性の例

3.2.2.6.4.4.2 ソースエンティティのライフタイム期間による管理

ソースエンティティが期間化されてなく、ソースエンティティにライフタイム期間に該当する属性が設定されている場合、ソースエンティティのライフタイム期間で設定された期間はターゲットエンティティで設定されている期間内であるかどうかチェックする必要がある。「図 3-37 参照元の期間の取得（ソースエンティティに有効期間が指定されている場合）」を参照すること。

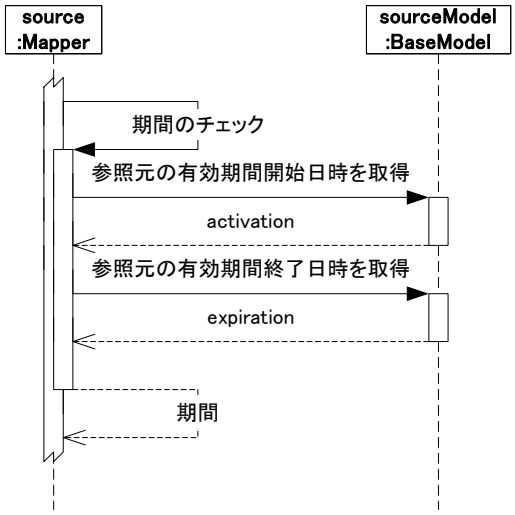


図 3-37 参照元の期間の取得(ソースエンティティに有効期間が指定されている場合)

ソースエンティティの有効期間によってライフタイムが管理されている場合の整合性の例を「図 3-38 ソースエンティティの有効期間によるライフタイム整合性の例」に示す。

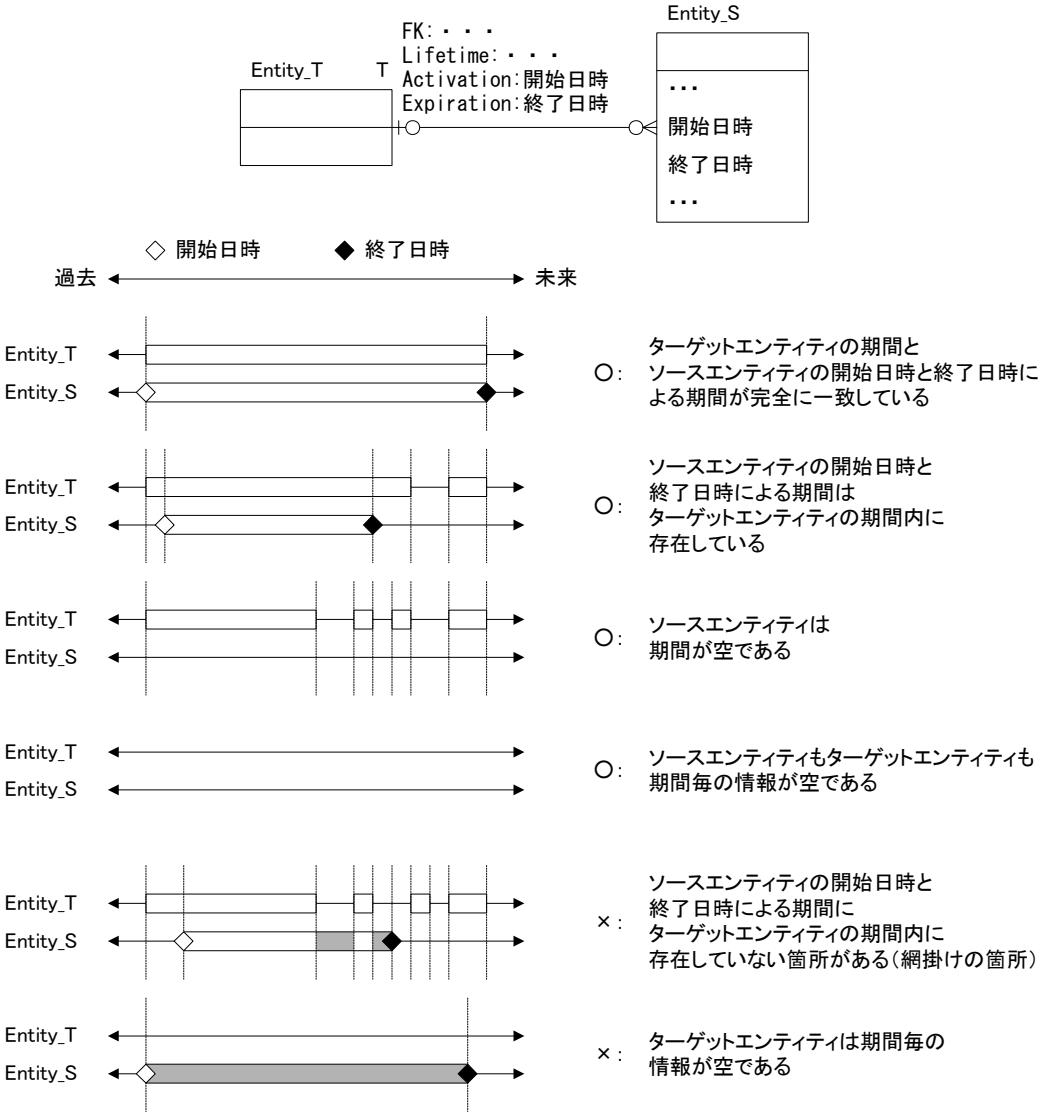


図 3-38 ソースエンティティの有効期間によるライフタイム整合性の例

3.2.2.6.4.4.3 ソースエンティティの参照先のライフタイム期間による管理

ソースエンティティが期間化されてなく、ソースエンティティからたどれる他のエンティティにライフタイム期間に該当する属性が設定されている場合、他のエンティティのライフタイム期間で設定された期間はターゲットエンティティで設定されている期間内であるかどうかチェックする必要がある。「図 3-39 参照元の期間の取得（ソースエンティティの参照先に有効期間が指定されている場合）」を参照すること。

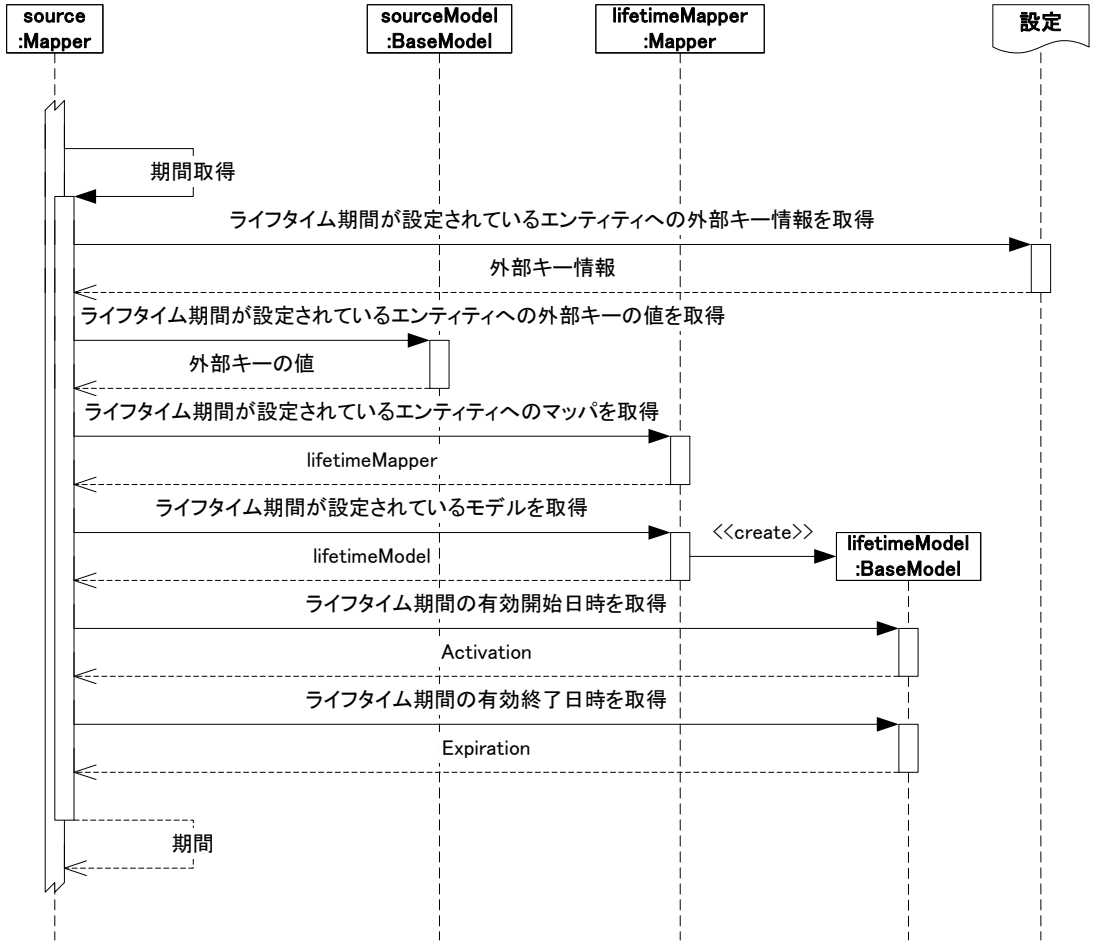


図 3-39 参照元の期間の取得（ソースエンティティの参照先に有効期間が指定されている場合）

ソースエンティティの有効期間によってライフタイムが管理されている場合の整合性の例を「図 3-40 ソースエンティティの参照先の有効期間によるライフタイム整合性の例」に示す。

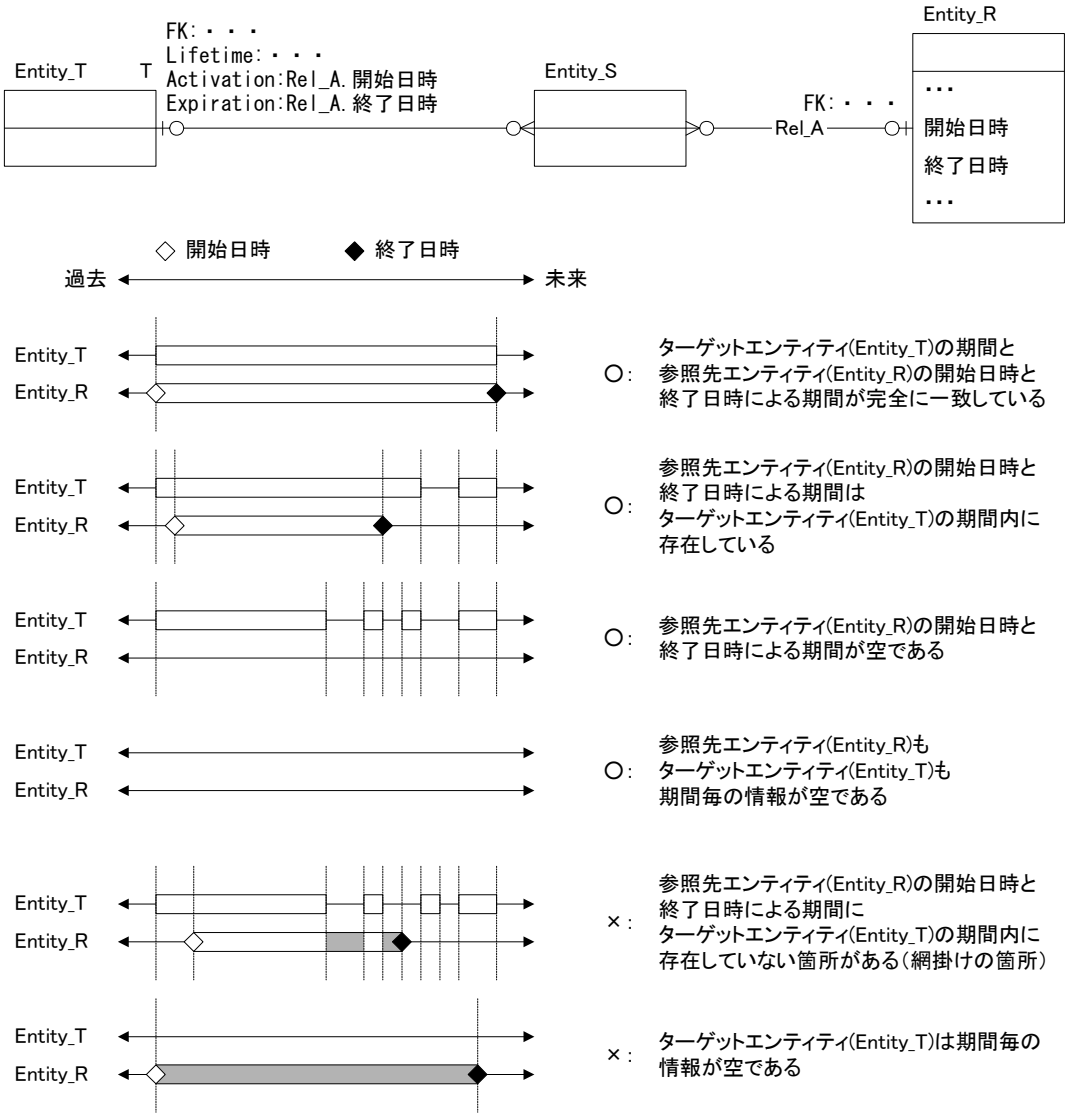


図 3-40 ソースエンティティの参照先の有効期間によるライフタイム整合性の例

3.2.2.7 モデルの更新

エンティティに既に存在するインスタンスの更新は基本モデルの単位で行う。既存の基本モデルに対して国際化モデル期間化モデルなどを追加する場合でも基本モデルの更新とみなす。更新するときは次のような手順をとる。

(1) マッパを通じて既存の基本モデルを取得する。
(2) 基本モデルに対して情報を更新する。
(3) 新たにデータを設定した基本モデルを、マッパを通じて更新する。

この概要を「図 3-41 基本モデルの更新(概要)」に示す。

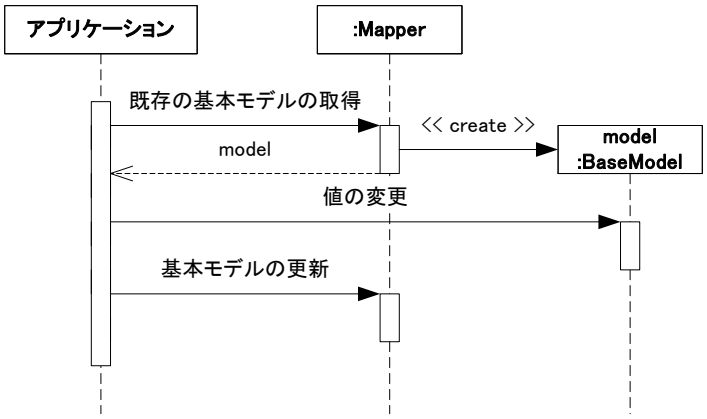


図 3-41 基本モデルの更新(概要)

3.2.2.7.1 基本モデルの更新

基本モデルを更新するときはマッパの update メソッドを使用する。このメソッドは引数に基本モデルを指定する。この基本モデルは、この基本モデルはデータストア上に実際に存在するデータを同一マッパのインスタンスによって(select メソッド等によって)取得されたものでなければならない。同じ種類であっても別インスタンスのマッパから取得された基本モデルを update メソッドの引数に指定することはできない。また、同一インスタンスのマッパから取得された基本モデルであっても、createBaseModel メソッドによって生成されたものや、delete メソッドなどによってデータストア上から削除されたものは update メソッドの引数に指定することはできない。

マップを通じて基本モデルを更新する様子を「図 3-42 モデルの更新」に示す。

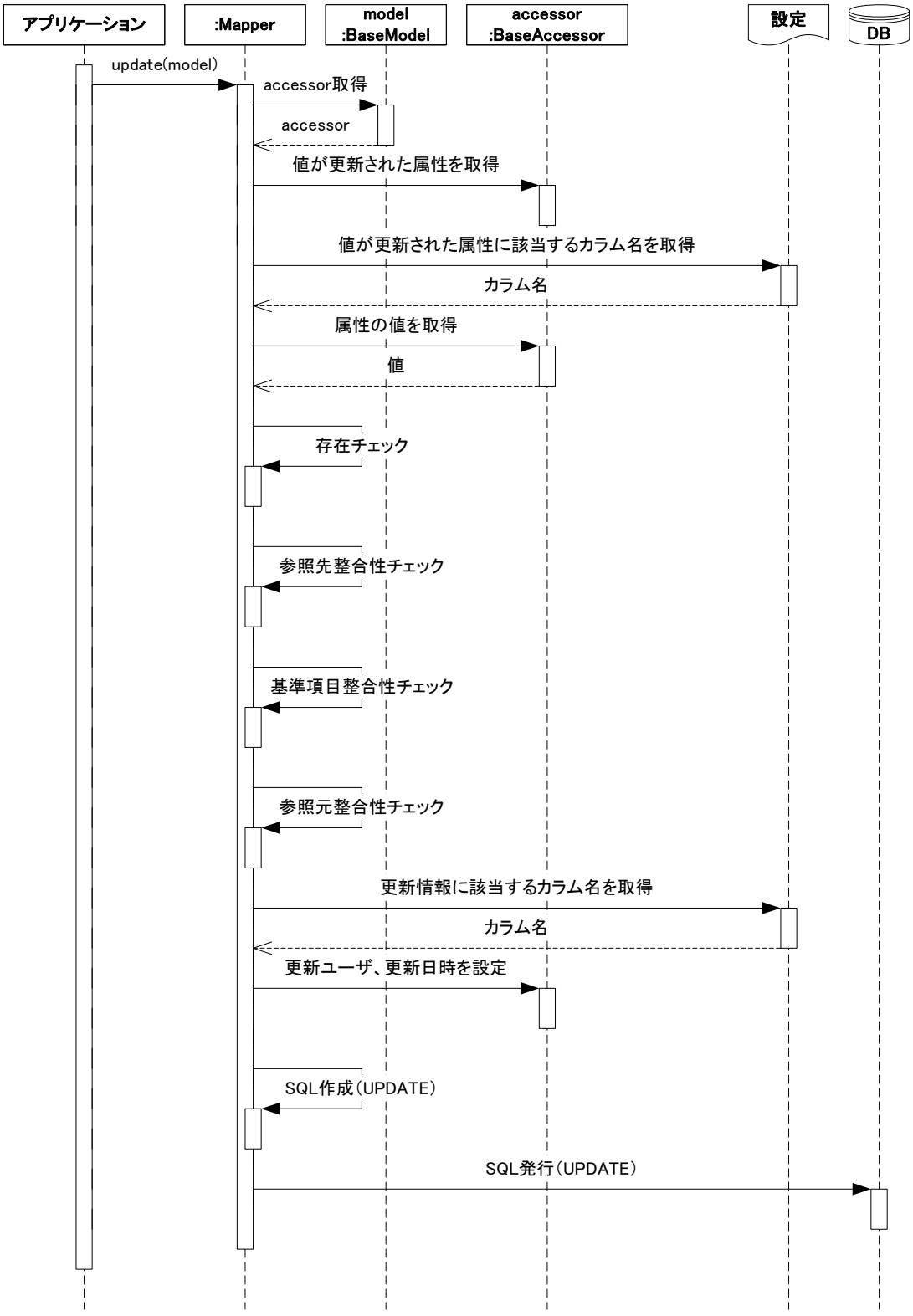


図 3-42 モデルの更新

「図 3-42 モデルの更新」ではデータストアがリレーショナルデータベースであることを想定しているが、他の種類のデータストアであってもかまわない。その場合、SQL作成とSQL発行の部分がデータストアに特化した方法で基本モデルの内容を永続化することになる。

「図 3-42 モデルの更新」で行っている存在チェックについては「3.2.2.7.2 更新時の存在チェック」を、参照先整

合性チェックについては「3.2.2.7.3 リレーションシップによる参照先の整合性のチェック(更新時)」を、基準項目整合性チェックについては「3.2.2.7.4 基準項目またはライフタイム期間による参照先の整合性チェック(更新時)」を、参照元整合性チェックについては「3.2.2.7.5 リレーションシップによる参照元の整合性のチェック(更新時)」を参照すること。

基本モデルを更新するサンプルを「リスト 3-2 更新のサンプルプログラム」に示す。

リスト 3-2 更新のサンプルプログラム

```
1: UserTransaction ut = ・・・;
 2: ut.begin;
 3: MapperFactory factory = MapperFactory.newInstance();
 4: UserMapper mapper =
 5:       (UserMapper)factory.createMapper("jp.co.intra_mart", "User",
 6:                                        "DEFAULT", "admin");
 7: UserBaseModel user = mapper.selectUserBaseModel("u0002");
 8: TerminableModelMap userTermMap = user.getTerminableModelMap();
 9: UserTerminableModel userTerm =
10:      (UserTerminableModel)userTermMap.createTerminableModel(new Term());
11: userTerm.setPassword("xxxxxx");
12: UserTerminableInternationalModel userTermI18n =
13:       userTerm.getUserTerminableInternationalModel(Locale.JAPANESE);
14: userTermI18n.setName("新ユーザ 0002");
15: mapper.updateUser(user);
16: mapper.close();
17: ut.commit();
```

3.2.2.7.2 更新時の存在チェック

既存の基本モデルを更新するとき、同一のプライマリキーで既にデータが登録されているかどうかをチェックする必要がある。この処理内容を「図 3-43 更新時の存在チェック」に示す。

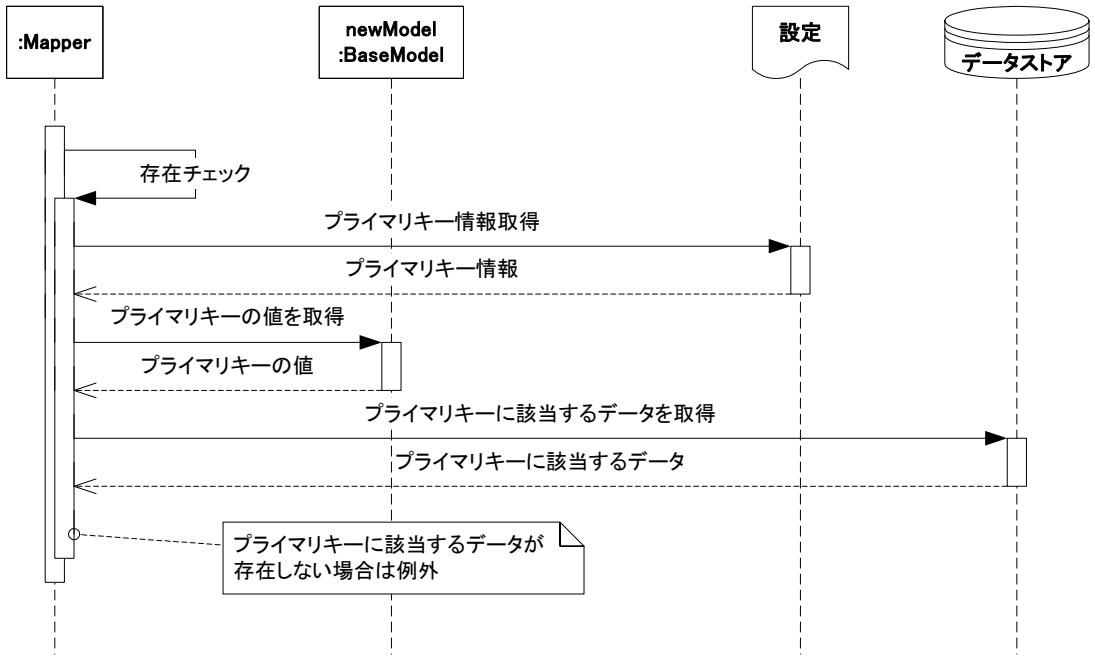


図 3-43 更新時の存在チェック

3.2.2.7.3 リレーションシップによる参照先の整合性のチェック（更新時）

以下のような条件をすべて満たす場合、データ更新時には整合性チェックを行う。

- 自分自身がソースエンティティとなる（自分自身もしくは他のエンティティを参照する）ようなリレーションシップが存在する
- 外部キーに該当する項目すべてに値（null 以外）が設定されている

上記の条件をすべて満たす場合、参照先のエンティティには外部キーに該当するデータが存在する必要がある。このときに行われる整合性のチェック内容は「図 3-30 挿入時の整合性チェック（共通）」とまったく同様である。

リレーションシップで以下の項目が定義されている場合、挿入時と同様の整合性検査を行った後さらに詳細なチェックを行う。

- 国際化基準項目（I18n）
- 期間化基準項目（Term）
- 期間の関連付け（Lifetime）

これらの検査内容も挿入時と同様である。詳細については「3.2.2.6.4 リレーションシップによる参照先の整合性のチェック（挿入時）」を参照すること。

3.2.2.7.4 基準項目またはライフタイム期間による参照先の整合性チェック（更新時）

自分自身がターゲットエンティティにもソースエンティティにもならないリレーションシップが自分自身に以下のいずれかの項目を定義している場合、データ更新時には整合性チェックを行う。

- 国際化基準項目（「2.2.3.2 他のエンティティで国際化基準項目を指定する方法」を参照）
- 期間化基準項目（「2.2.4.2 他のエンティティで期間化基準項目を指定する方法」を参照）
- ライフタイム期間（「2.2.5.5.2 他のエンティティでライフタイム期間を指定する方法」を参照）

上記のいずれかの条件を満たす場合、該当するリレーションシップのターゲットエンティティには外部キーに該当するデータが存在し、それぞれの項目の条件を満たしている必要がある。このときに行われる整合性のチェック内容を「図 3-44 基準項目の整合性チェック」に示す。

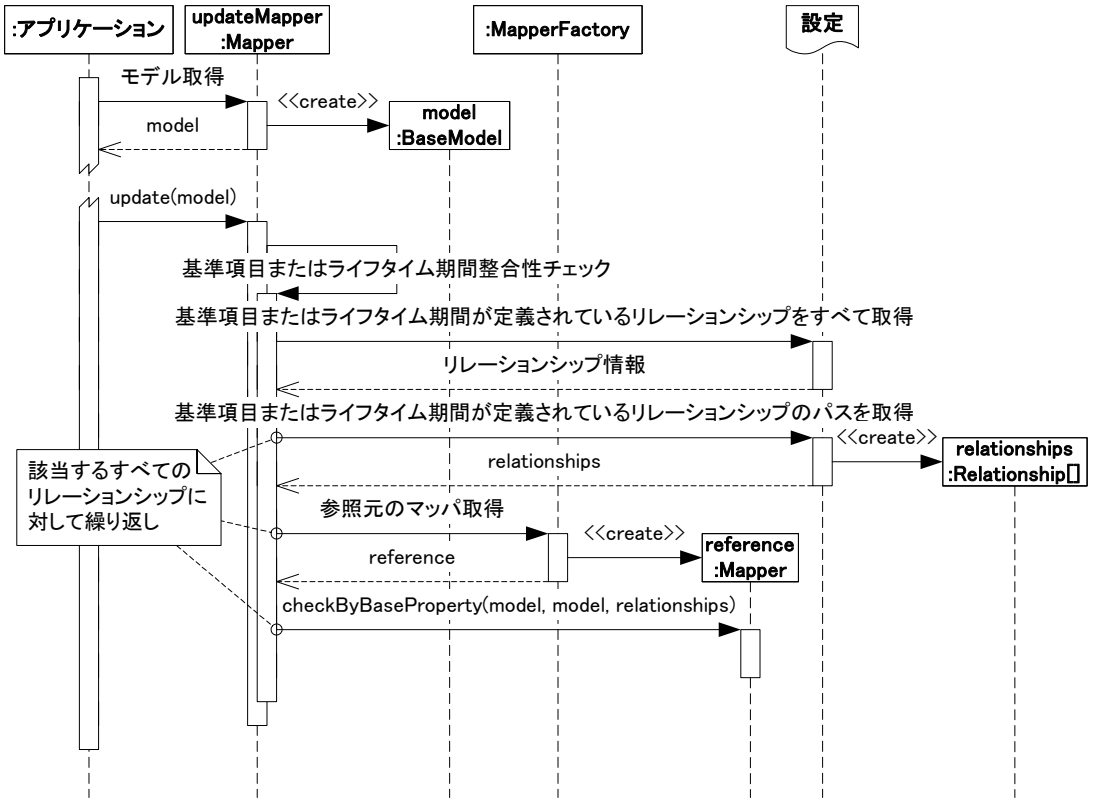


図 3-44 基準項目の整合性チェック

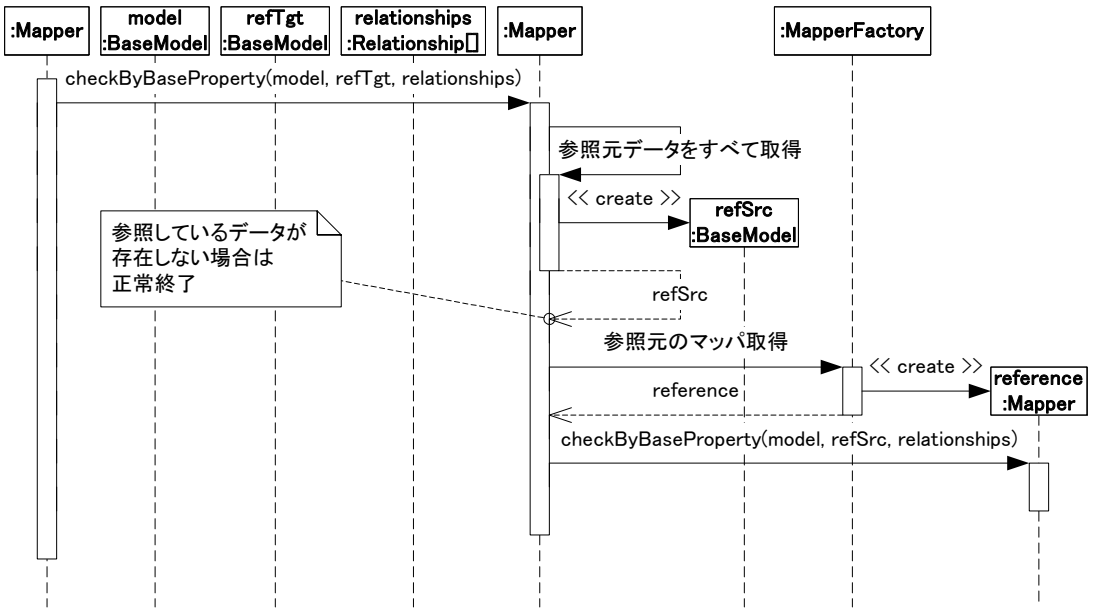


図 3-45 基準項目の整合性チェック(リレーションシップ中)

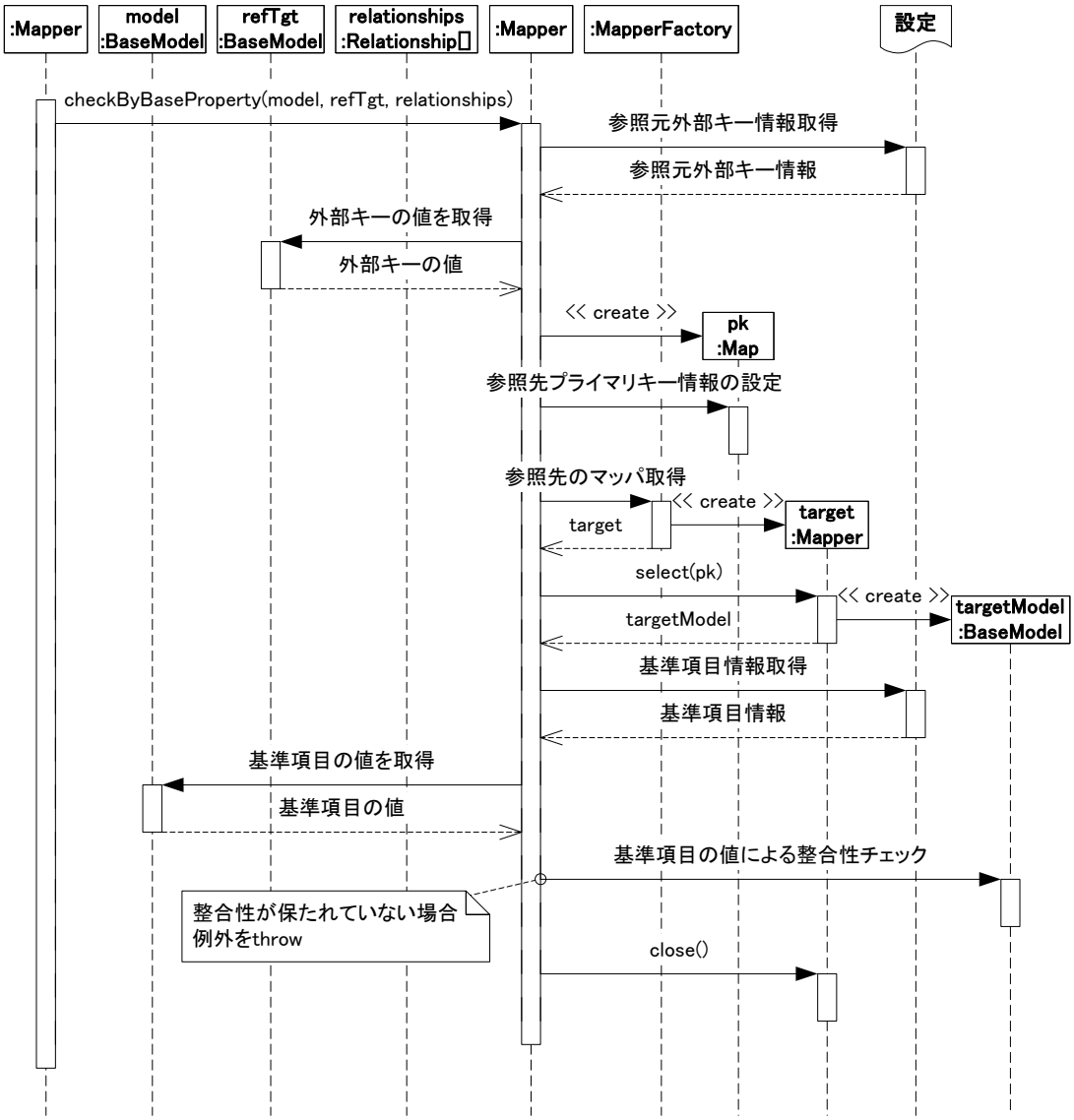


図 3-46 基準項目の整合性チェック(ソースエンティティ)

基準項目の値によるチェック内容の詳細については「3.2.2.6.4.1 パターン 1(国際化のみ)」から「3.2.2.6.4.4 パターン 4(ライフタイム)」を参照すること。

3.2.2.7.5 リレーションシップによる参照元の整合性のチェック(更新時)

以下のような条件をすべて満たす場合、データ更新時には整合性チェックを行う。

- 自分自身がターゲットエンティティとなる(自分自身もしくは他のエンティティから参照される)ようなリレーションシップが存在する
- 以下のいずれかの変更が行われている
  - 上記のリレーションシップが国際化されており、国際化モデルを削除した
  - 上記のリレーションシップが期間化されており、期間化モデルの期間を変更した
  - 上記のリレーションシップが期間化かつ国際化されており、期間化モデルの期間を変更した、または期間国際化モデルを削除した
  - 上記のリレーションシップにライフタイムが設定されており、期間化モデルの期間を変更した

上記の条件をすべて満たす場合、外部キーによって自分自身が参照されているようなデータが参照元のエンティティに存在するかどうか調べる必要がある。この様な条件を満たすデータを取得する内容を「図 3-47 更新時の

参照元データの取得(共通)」に示す。

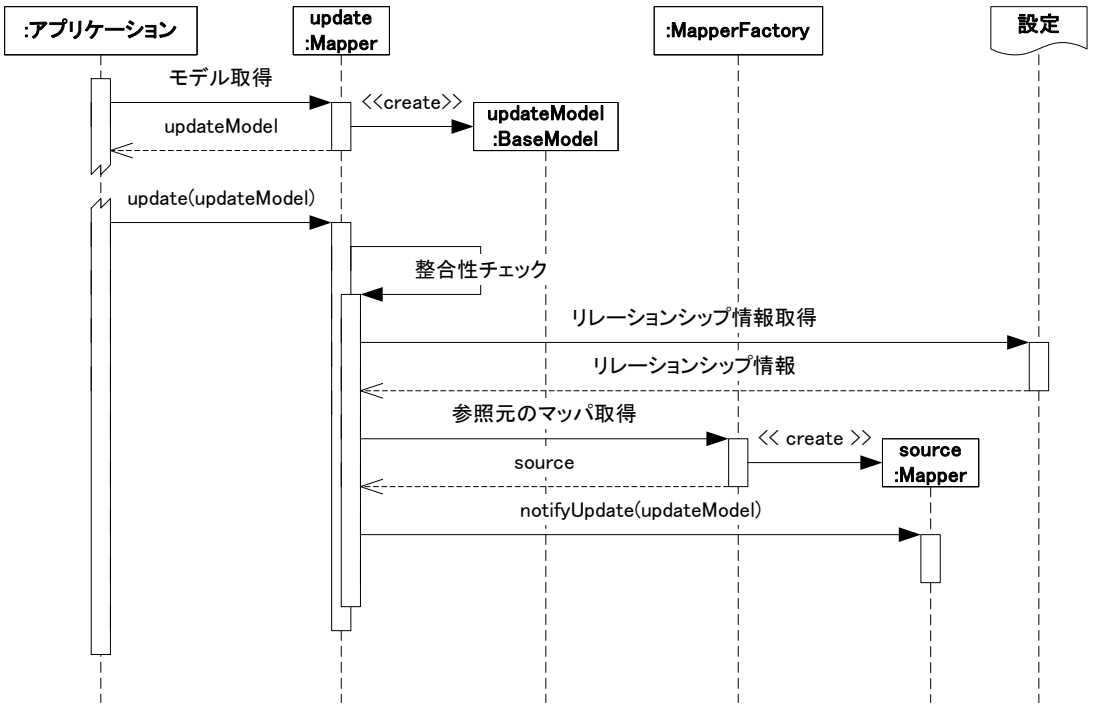


図 3-47 更新時の参照元データの取得(共通)

「図 3-47 更新時の参照元データの取得(共通)」で参照元の基本モデルが 1 つ以上取得された場合、取得した基本モデルすべてに対してさらに詳細なチェックを行う。チェック内容の一覧を「表 3-4 更新時の参照元の整合性チェック(項目別)」に示す。

表 3-4 更新時の参照元の整合性チェック(項目別)

リレーションシップ			チェック
国際化基準項目 (I18n) ○:あり、×:なし	期間化基準項目 (Term) ○:あり、×:なし	ライフタイム (Lifetime) ○:あり、×:なし	
×	×	―	整合性チェック不要
○	×	―	パターン 1
×	○	―	パターン 2
○	○	―	パターン 3
―	―	○	パターン 4

以下にそれぞれのパターンにおける個別のチェック内容の詳細を記述する。

3.2.2.7.5.1 パターン 1(国際化のみ)

ソースエンティティの基本モデルの国際化基準項目で指定される言語(ロケール)に該当する国際化モデルが、更新しようとしている基本モデルに存在するかチェックする。

このチェックは国際化基準項目で指定される言語が設定されている場合のみ行う。国際化基準項目が設定されていない場合、外部キーに該当するエンティティがあっても何も参照していないものとみなす。以下のいずれかの条件を満たす場合、国際化基準項目が設定されているものとみなされる。

- 国際化基準項目がソースエンティティ内の属性として定義されている場合、その属性に null 以外の値が

設定されていれば国際化基準項目が設定されているとみなす。

- 国際化基準項目がソースエンティティとは別のエンティティで定義されている場合、リレーションシップをたどった結果として取得した国際化基準項目に null 以外の値が設定されていれば国際化基準項目が設定されているとみなす。

チェックする様子を「図 3-48 更新時の参照元整合性チェック(国際化)」に示す。

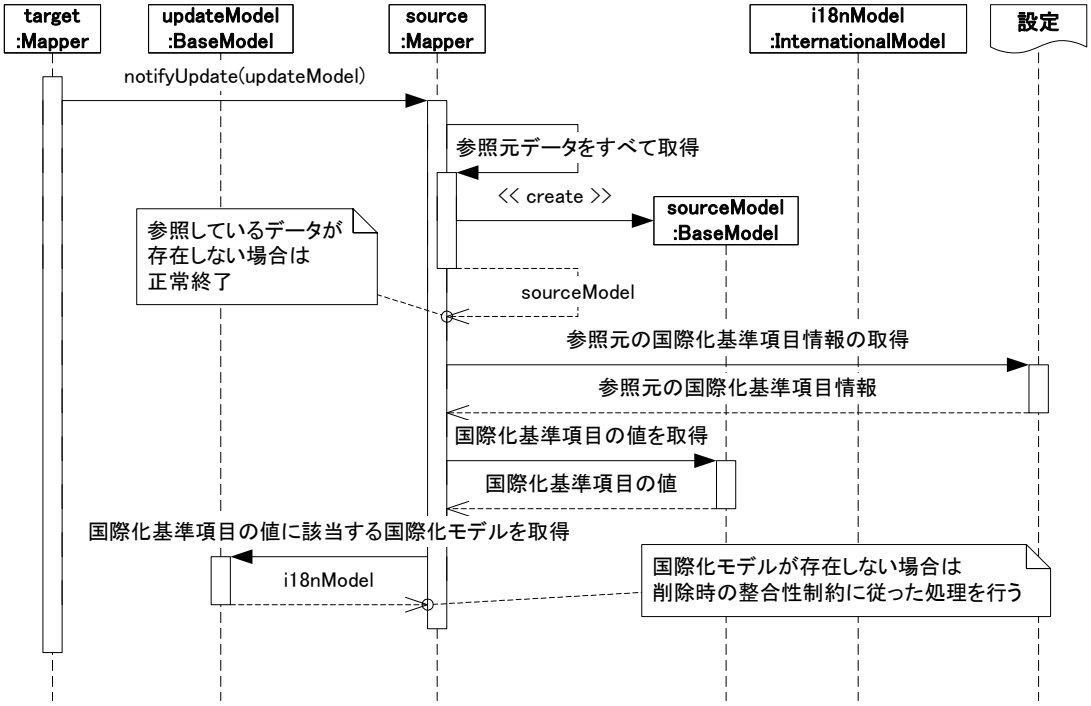


図 3-48 更新時の参照元整合性チェック(国際化)

「図 3-48 更新時の参照元整合性チェック(国際化)」では、更新しようとしている基本モデルに対象となる国際化モデルが存在しない場合の処理は、リレーションシップで定義された削除時の整合性制約に従うようになっている。詳細については「2.4.1 ターゲットエンティティのインスタンスの削除」を参照すること。

ソースエンティティのインスタンスも同時に削除する(Cascade)場合、以下の条件にしたがって参照元のデータを更新または削除する。

- 参照元の国際化基準項目が基本モデルに定義されている場合、参照元の基本モデルを削除する。
- 参照元の国際化基準項目が国際化モデルに定義されている場合、参照元の国際化モデルを削除する。
- 参照元の国際化基準項目が期間化モデルに定義されている場合、参照元の期間化モデルおよび期間国際化モデルを削除する。
- 参照元の国際化基準項目が期間国際化モデルに定義されている場合、参照元の期間国際化モデルを削除する。

ソースエンティティのインスタンスの外部キーに該当する属性を null にする(Null)場合、該当する国際化基準項目の値を null にする。

3.2.2.7.5.2 パターン 2(期間化のみ)

ソースエンティティの基本モデルの期間化基準項目で指定される日時に該当する期間化モデルが、更新しようとしている基本モデルに存在するかチェックする。

このチェックは期間化基準項目で指定される日時が設定されている場合のみ行う。期間化基準項目が設定されていない場合、外部キーに該当するエンティティがあっても何も参照していないものとみなす。以下のいずれかの条件を満たす場合、期間化基準項目が設定されているものとみなされる。

- 期間化基準項目がソースエンティティ内の属性として定義されている場合、その属性に null 以外の値が設定されていれば期間化基準項目が設定されているとみなす。
- 期間化基準項目がソースエンティティとは別のエンティティで定義されている場合、リレーションシップをたどった結果として取得した期間化基準項目に null 以外の値が設定されていれば期間化基準項目が設定されているとみなす。

チェックする様子を「図 3-49 更新時の参照元整合性チェック(期間化)」に示す。

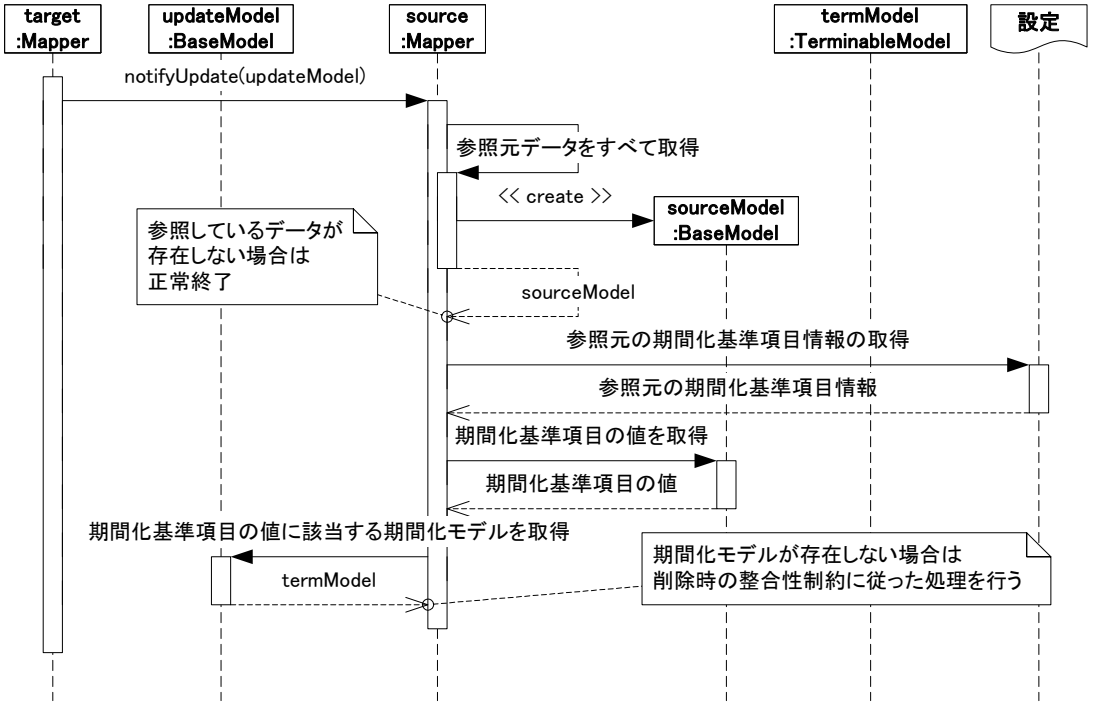


図 3-49 更新時の参照元整合性チェック(期間化)

「図 3-49 更新時の参照元整合性チェック(期間化)」では、更新しようとしている基本モデルに対象となる期間化モデルが存在しない場合の処理は、リレーションシップで定義された削除時の整合性制約に従うようになっている。詳細については「2.4.1 ターゲットエンティティのインスタンスの削除」を参照すること。

ソースエンティティのインスタンスも同時に削除する(Cascade)場合、以下の条件にしたがって参照元のデータを更新または削除する。

- 参照元の期間化基準項目が基本モデルに定義されている場合、参照元の基本モデルを削除する。
- 参照元の期間化基準項目が国際化モデルに定義されている場合、参照元の国際化モデルを削除する。
- 参照元の期間化基準項目が期間化モデルに定義されている場合、参照元の期間化モデルおよび期間国際化モデルを削除する。
- 参照元の期間化基準項目が期間国際化モデルに定義されている場合、参照元の期間国際化モデルを削除する。

ソースエンティティのインスタンスの外部キーに該当する属性を null にする(Null)場合、該当する期間化基準項目の値を null にする。

3.2.2.7.5.3 **パターン 3（国際化＋期間化）**

ソースエンティティの基本モデルの期間化基準項目で指定される日時と国際化基準項目で指定される言語（ロケール）に該当する期間国際化モデルが、更新しようとしている基本モデルに存在するかチェックする。

このチェックは期間化基準項目で指定される日時および国際化基準項目で指定される言語（ロケール）が両方とも設定されている場合のみ行う。期間化基準項目または国際化基準項目が設定されていない場合、外部キーに該当するエンティティがあっても何も参照していないものとみなす。以下のいずれかの条件を満たす場合、期間化基準項目および国際化基準項目が両方とも設定されているものとみなされる。

- 期間化基準項目および国際化基準項目がソースエンティティ内の属性として定義されている場合、その属性に両方とも null 以外の値が設定されていれば期間化基準項目および国際化基準項目が設定されているとみなす。
- 期間化基準項目および国際化基準項目がソースエンティティとは別のエンティティで定義されている場合、リレーションシップをたどった結果として取得した期間化基準項目および国際化基準項目に両方とも null 以外の値が設定されていれば期間化基準項目および国際化基準項目が設定されているとみなす。

チェックする様子を「図 3-50 更新時の参照元整合性チェック（国際化＋期間化）」に示す。

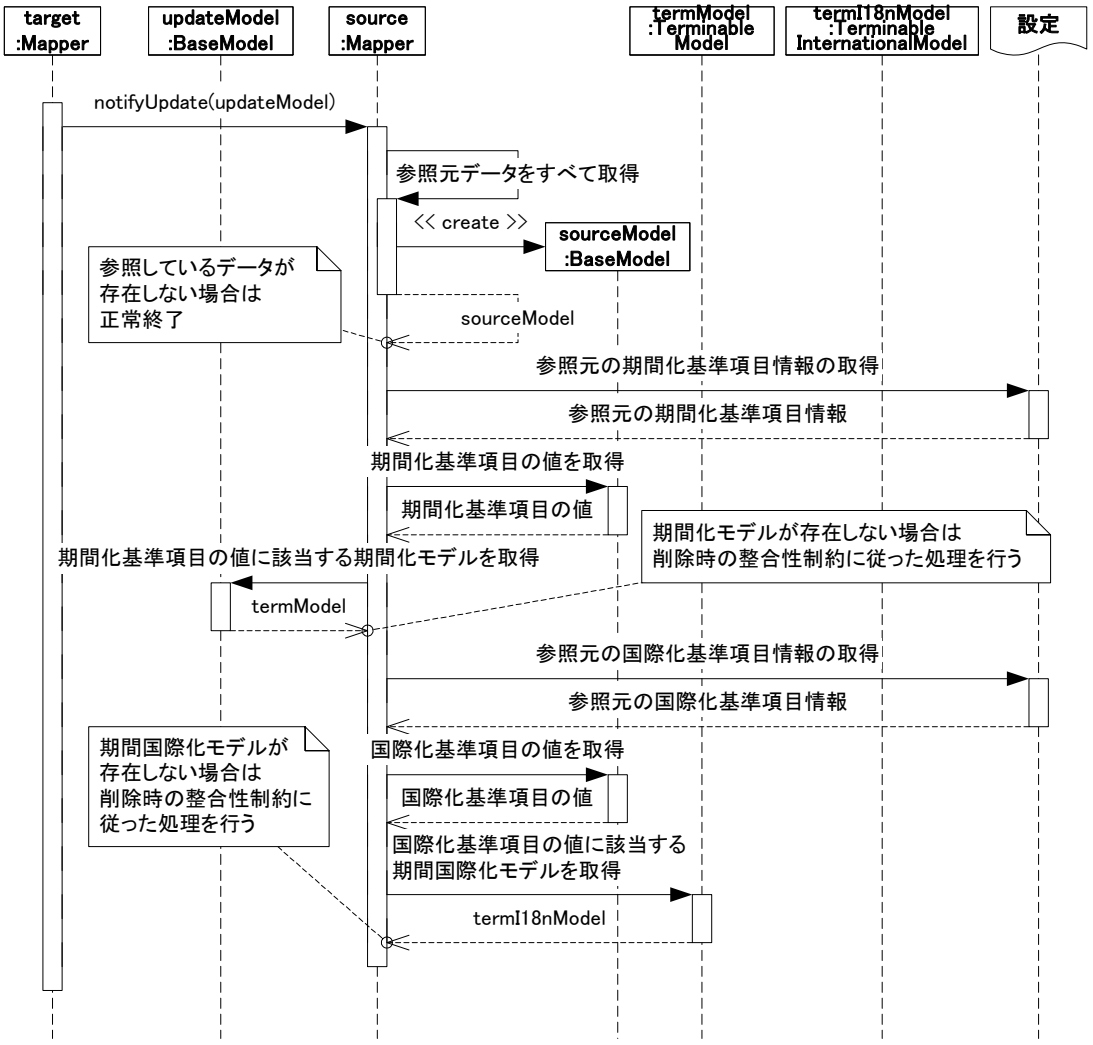


図 3-50 更新時の参照元整合性チェック（国際化＋期間化）

「図 3-50 更新時の参照元整合性チェック（国際化＋期間化）」では、更新しようとしている基本モデルに対象とな

る期間化モデルが存在しない場合の処理は、リレーションシップで定義された削除時の整合性制約に従うようになっている。詳細については「2.4.1 ターゲットエンティティのインスタンスの削除」を参照すること。

ソースエンティティのインスタンスも同時に削除する（Cascade）場合、以下の条件にしたがって参照元のデータを更新または削除する。

- 参照元の期間化基準項目および国際化基準項目が両方とも基本モデルに定義されている場合、参照元の基本モデルを削除する。
- 参照元の期間化基準項目および国際化基準項目が国際化モデルに定義されている場合、参照元の国際化モデルを削除する。
- 参照元の期間化基準項目および国際化基準項目が期間化モデルに定義されている場合、参照元の期間化モデルおよび期間国際化モデルを削除する。
- 参照元の期間化基準項目および国際化基準項目が期間国際化モデルに定義されている場合、参照元の期間国際化モデルを削除する。

ソースエンティティのインスタンスの外部キーに該当する属性を null にする（Null）場合、以下の条件にしたがって参照元のデータを更新する。

- 参照元の期間化基準項目および国際化基準項目が両方とも基本モデルに定義されている場合、参照元の基本モデルの期間化基準項目および国際化基準項目の値を null にする。
- 参照元の期間化基準項目および国際化基準項目が国際化モデルに定義されている場合、参照元の国際化モデルの期間化基準項目および国際化基準項目の値を null にする。
- 参照元の期間化基準項目および国際化基準項目が期間化モデルに定義されている場合、参照元の期間化モデルおよび期間国際化モデルの期間化基準項目および国際化基準項目の値を null にする。
- 参照元の期間化基準項目および国際化基準項目が期間国際化モデルに定義されている場合、参照元の期間国際化モデルの期間化基準項目および国際化基準項目の値を null にする。

3.2.2.7.5.4 パターン 4（ライフタイム）

基本モデル内に存在するすべての期間化モデルの期間またはライフタイム期間が、更新する基本モデル内に存在するすべての期間化モデルの期間範囲内であるかチェックする。

このチェックはライフタイム制約が設定されている場合は常に行う。

チェックする様子を「図 3-51 更新時の参照元整合性チェック（ライフタイム）」に示す。

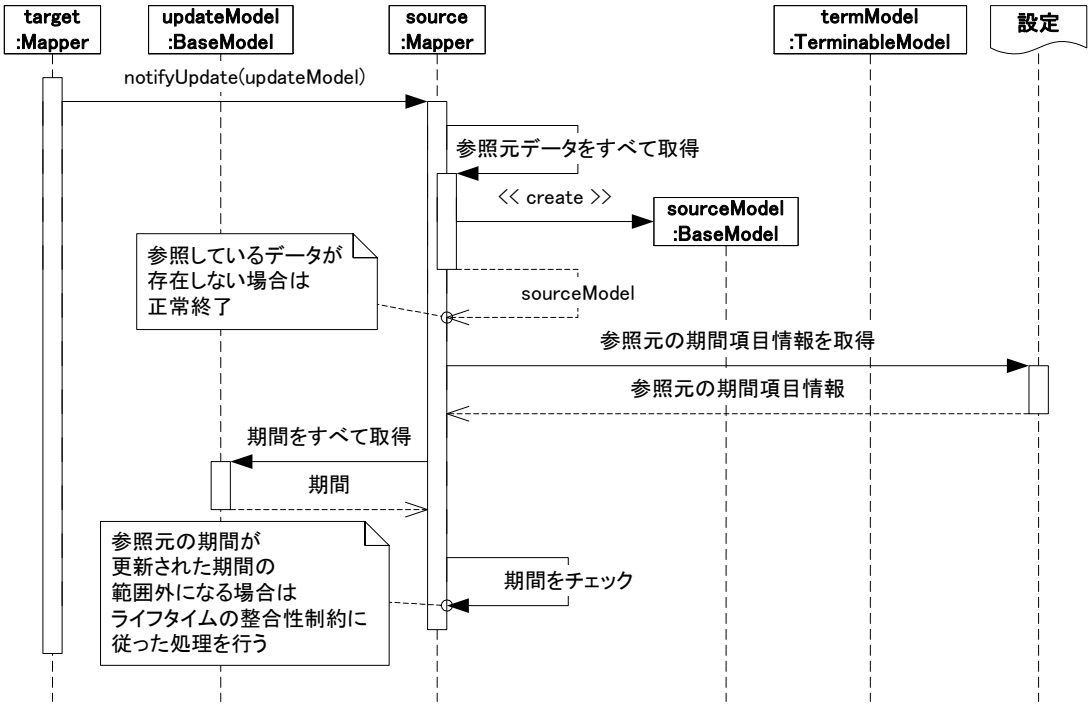


図 3-51 更新時の参照元整合性チェック(ライフタイム)

「図 3-51 更新時の参照元整合性チェック(ライフタイム)」では、参照元の期間が更新しようとしている基本モデル内の期間の範囲外となる場合の処理は、リレーションシップで定義されたライフタイムの整合性制約に従うようになっている。詳細については「2.4.2 ターゲットエンティティのインスタンスの期間の削除」を参照すること。

「図 3-51 更新時の参照元整合性チェック(ライフタイム)」において"期間を取得"となっている箇所は、挿入時と同じである。詳細は「3.2.2.6.4.4 パターン 4(ライフタイム)」の説明を参照すること。

3.2.2.8 モデルの削除

エンティティに既に存在するインスタンスの削除は基本モデルの単位で行う。削除するときは次のような手順をとる。

(1) マッパを通じて既存の基本モデルを取得する。
(2) 取得した基本モデルを、マッパを通じて削除する。

この概要を「図 3-52 基本モデルの削除(概要)」に示す。

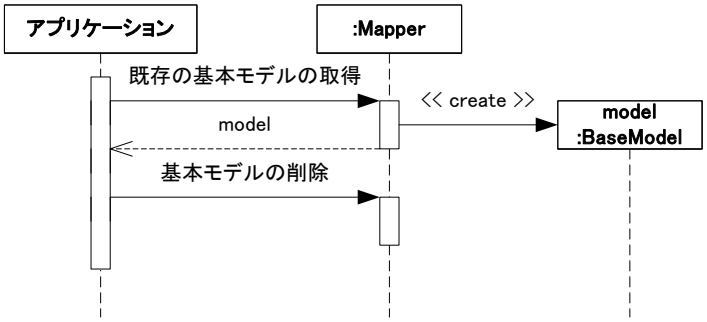


図 3-52 基本モデルの削除(概要)

3.2.2.8.1 **基本モデルの削除**

基本モデルを削除するときはマッパの delete メソッドを使用する。このメソッドは引数に基本モデルを指定する。この基本モデルは、この基本モデルは同一マッパのインスタンスから取得されたものでなければならない。同じ種類であっても別インスタンスのマッパから取得された基本モデルを delete メソッドの引数に指定することはできない。また、同一インスタンスのマッパから取得された基本モデルであっても、createBaseModel メソッドによって生成されたものや、select メソッド等によって取得されたものは delete メソッドの引数に指定することはできない。

マップを通じて基本モデルを削除する様子を「図 3-53 モデルの削除」に示す。

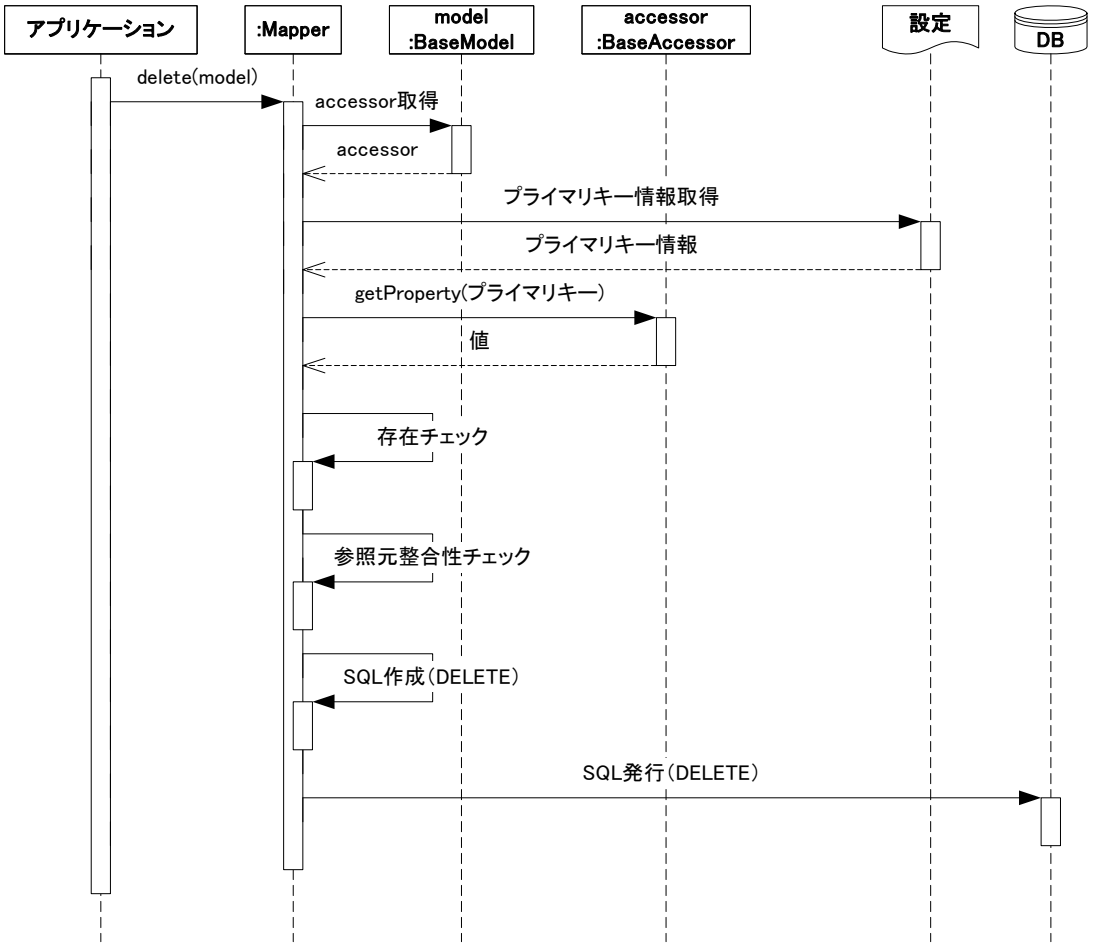


図 3-53 モデルの削除

「図 3-53 モデルの削除」ではデータストアがリレーショナルデータベースであることを想定しているが、他の種類のデータストアであってもかまわない。その場合、SQL作成とSQL発行の部分がデータストアに特化した方法で基本モデルの内容を永続化することになる。

「図 3-53 モデルの削除」で行っている存在チェックについては「3.2.2.8.2 削除時の存在チェック」を、参照元整合性チェックについては「3.2.2.8.4 リレーションシップによる参照元の整合性のチェック(削除時)」を参照すること。

基本モデルを更新するサンプルを「リスト 3-3 削除のサンプルプログラム」に示す。

リスト 3-3 削除のサンプルプログラム

```
1: UserTransaction ut = ・・・;
2: ut.begin;
3: MapperFactory factory = MapperBuilderFactory.newInstance();
4: UserMapper mapper =
5:        (UserMapper)factory.createMapper(“jp.co.intra_mart.app”, “User”,
6:                                                    “DEFAULT”, “admin”);
7: mapper.delete(user);
8: mapper.close();
9: ut.commit();
```

3.2.2.8.2 削除時の存在チェック

既存の基本モデルを更新するとき、同一のプライマリキーで既にデータが登録されているかどうかをチェックする必要がある。このときに行われる整合性のチェック内容は「図 3-43 更新時の存在チェック」とまったく同様である。詳細については「3.2.2.7.2 更新時の存在チェック」を参照すること。

3.2.2.8.3 基準項目またはライフタイム期間による参照先の整合性チェック（削除時）

自分自身がターゲットエンティティにもソースエンティティにもならないリレーションシップが自分自身に以下のいずれかの項目を定義している場合、データ更新時には整合性チェックを行う。

- 国際化基準項目（「2.2.3.2 他のエンティティで国際化基準項目を指定する方法」を参照）
- 期間化基準項目（「2.2.4.2 他のエンティティで期間化基準項目を指定する方法」を参照）
- ライフタイム期間（「2.2.5.5.2 他のエンティティでライフタイム期間を指定する方法」を参照）

この場合の整合性チェックは「3.2.2.7.4 基準項目またはライフタイム期間による参照先の整合性チェック（更新時）」で説明している方法とまったく同様である。詳細についてはそちらを参照すること。

3.2.2.8.4 リレーションシップによる参照元の整合性のチェック（削除時）

自分自身がターゲットエンティティとなる（自分自身もしくは他のエンティティから参照される）ようなリレーションシップが存在する場合、データ削除時には整合性チェックを行う。

削除時の整合性チェックを行う場合、外部キーによって自分自身が参照されているようなデータが参照元のエンティティに存在するかどうか調べる必要がある。この様な条件を満たすデータを取得する方法は「図 3-47 更新時の参照元データの取得（共通）」とまったく同様である。詳細については「3.2.2.7.5 リレーションシップによる参照元の整合性のチェック（更新時）」を参照すること。

上記の様な条件を満たす基本モデルが 1 つ以上取得された場合、取得した基本モデルすべてに対してさらに以下に示すような詳細なチェックを行う。

- 外部キーとなるすべての従属属性が国際化も期間化もされていない場合、取得した基本モデルに対して削除時の整合性制約の処理を適用する。整合性制約の処理としてNullが指定されていた場合、削除時Nullキー項目で指定された属性（「2.2.6.2 参照元の外部キー情報をnullに設定」を参照）に該当するプロパティをすべてnullにする。
- 外部キーとなるすべての従属属性が国際化のみされており、期間化されていない場合、取得した基本モデルの該当する国際化モデルに対して削除時の整合性制約の処理を適用する。整合性制約の処理としてNullが指定されていた場合、取得した国際化モデル内の削除時Nullキー項目で指定された属性（「2.2.6.2 参照元の外部キー情報をnullに設定」を参照）に該当するプロパティをすべてnullにする。
- 外部キーとなるすべての従属属性が期間化のみされており、国際化されていない場合、取得した基本モデルの該当する期間化モデルに対して削除時の整合性制約の処理を適用する。整合性制約の処理としてNullが指定されていた場合、取得した期間化モデル内の削除時Nullキー項目で指定された属性（「2.2.6.2 参照元の外部キー情報をnullに設定」を参照）に該当するプロパティのみをすべてnullにする。

- 外部キーとなるすべての従属属性が国際化かつ期間化されている場合、取得した基本モデルの該当する期間国際化モデルに対して削除時の整合性制約の処理を適用する。整合性制約の処理としてNullが指定されていた場合、取得した期間国際化モデル内の削除時Nullキー項目で指定された属性(「2.2.6.2 参照元の外部キー情報をnullに設定」を参照)に該当するプロパティをすべてnullにする。
- 外部キーとなる従属属性のうちいくつかが国際化のみされており、残りのものは国際化も期間化もされていない場合、取得した基本モデルの該当する国際化モデルに対して削除時の整合性制約の処理を適用する。整合性制約の処理としてNullが指定されていた場合、取得した国際化モデル内の削除時Nullキー項目で指定された属性(「2.2.6.2 参照元の外部キー情報をnullに設定」を参照)に該当するプロパティをすべてnullにする。
- 外部キーとなる従属属性のうちいくつかが期間化のみされており、残りのものは国際化も期間化もされていない場合、取得した基本モデルの該当する期間化モデルに対して削除時の整合性制約の処理を適用する。整合性制約の処理としてNullが指定されていた場合、取得した期間化モデル内の削除時Nullキー項目で指定された属性(「2.2.6.2 参照元の外部キー情報をnullに設定」を参照)に該当するプロパティをすべてnullにする。
- 外部キーとなる従属属性のうちいくつかが期間化かつ国際化されており、残りのものは国際化されていない(期間化されていてもよい)場合、取得した基本モデルの該当する期間国際化モデルに対して削除時の整合性制約の処理を適用する。整合性制約の処理としてNullが指定されていた場合、取得した期間国際化モデル内の削除時Nullキー項目で指定された属性(「2.2.6.2 参照元の外部キー情報をnullに設定」を参照)に該当するプロパティをすべてnullにする。

3.2.2.9 マッパのクローズ

MapperFactory から Mapper を取得したアプリケーションはトランザクションが終了する前にその Mapper の close メソッドを呼ぶ必要がある。Close メソッドが呼ばれた Mapper は isClosed メソッド以外のすべてのメソッドは例外を返す。

3.2.2.10 例外

アプリケーションは Mapper のメソッドを実行したときになんらかの例外が発生した場合、Mapper に対する処理に関連するトランザクションをロールバックしなければならない。

3.3 マネージャ

モデルパッケージのインタフェースやクラスを使用すれば一通りの操作はできる。マネージャは開発者から扱いやすいようにした API の集合である。

マネージャとマッパは似たような役割をするが、以下の点で異なる。

- マッパは 1 つのエンティティに対して 1 つ割り当てられる。
- マネージャは 1 つのドメインに対して1つ割り当てられる。
- マッパは Mapper インタフェースを実装する必要があるが、マネージャは特に制約はない。
- マッパは主に BaseModel を扱うが、マネージャでは外部的には扱わない(内部的に扱う場合はある)。

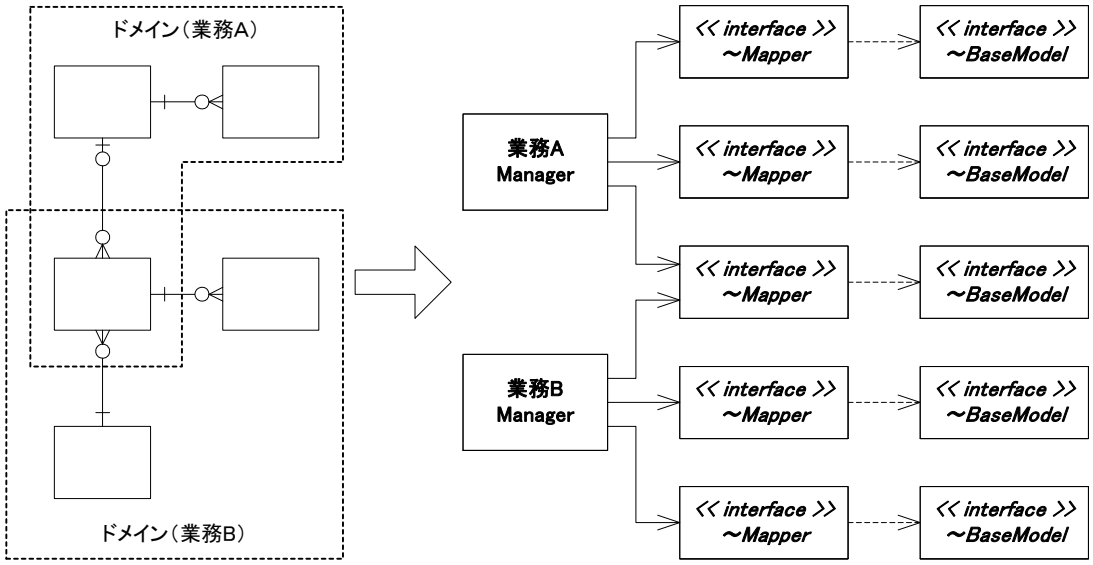
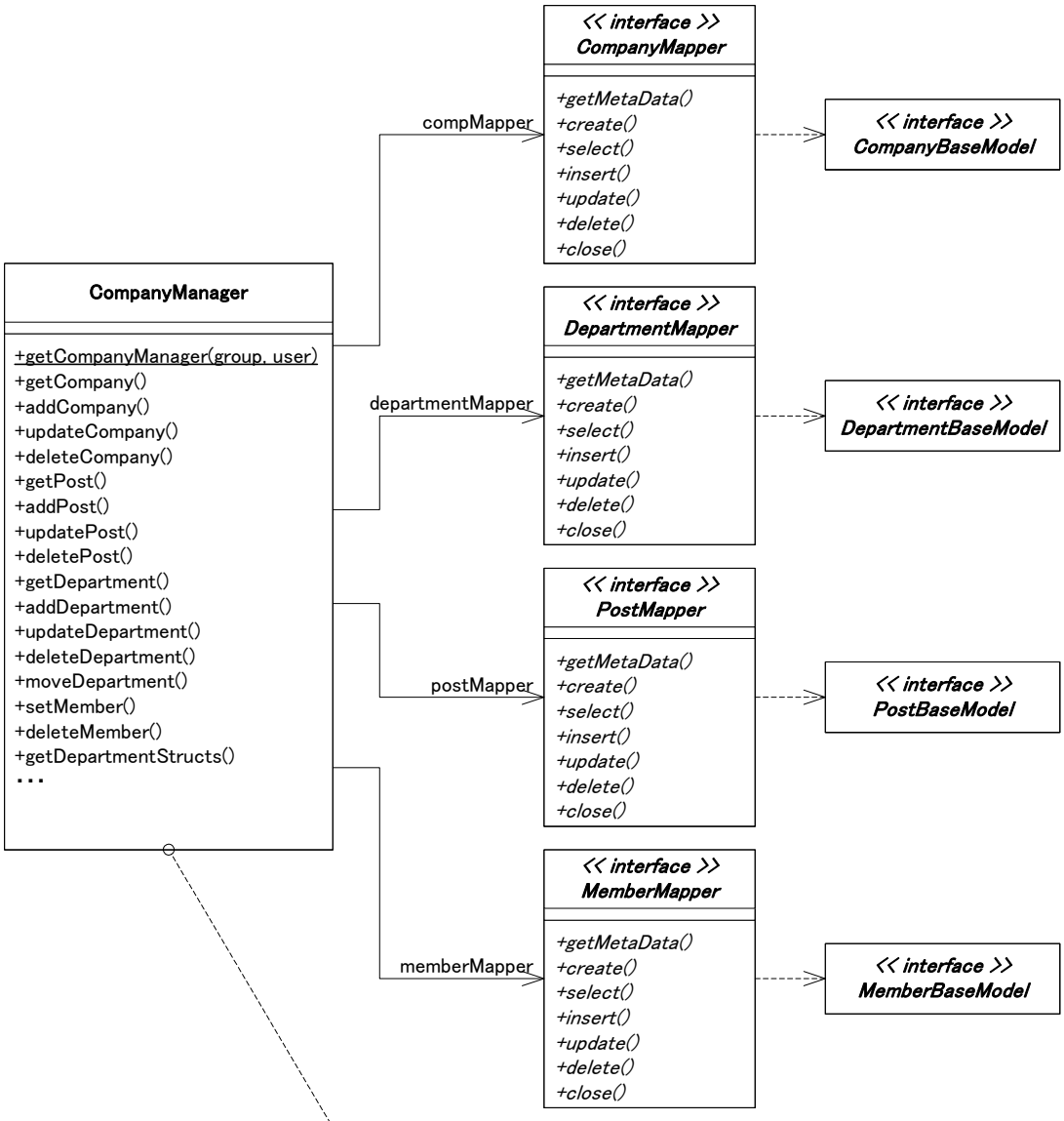


図 3-54 マッパとマネージャ




```
public void setMember(String company, String department, String user) {
    MemberMapper memberMapper = ・・・; // MemberMapperを取得
    MemberBaseModel member = memberMapper.create(company, department, user);
    memberMapper.insert(member);
    memberMapper.close();
}

public void addDepartment(Department dept, String parent, String version) {
    DepartmentMapper departmentMapper= ・・・; // DepartmentMapper を取得
    DepartmentBaseModel departmentModel
        = departmentMapper.create(dept.getCompanyCode(), dept.getDepartmentCode());
    // departmentModelにdeptの情報を設定
    ・・・
    departmentMapper.insert(departmentModel);
    departmentMapper.close();
}
```

図 3-55 会社・組織関連のマネージャの例

3.4 設定

エンティティやそれに対応する Mapper などの設定情報はすべて外部で指定する。Intra-mart に標準で搭載されている MapperFactory はその設定情報を以下の場所に保管しているものとして動作する。

<intra-mart Application Runtime のインストールディレクトリ>/conf/datastore

ここで設定されているMapperFactoryは、上述したディレクトリからの相対パスをもとにしてアプリケーション名を決定している。そのため、実際に設定すべきファイルの位置とファイル名はアプリケーション名に依存する。アプリケーション名に含まれるピリオド(".")はディレクトリの区切りを意味し、ファイル名はアプリケーション名の最後の部分に".xml"を追加したものとなる。この関係は、Javaにおけるパッケージ名とクラス名の関連とほぼ同意である(「表 3-5 アプリケーション名と設定ファイルの例」を参照)。

表 3-5 アプリケーション名と設定ファイルの例

アプリケーション名	設定ファイル
sample.MyApplication	<ApplicationRuntime>/conf/datastore/sample/MyApplication.xml
Myapp	<ApplicationRuntime>/conf/datastore/Myapp.xml
aaa.bbb.ccc.Test	<ApplicationRuntime>/conf/datastore/aaa/bbb/ccc/Test.xml

「表 3-5 アプリケーション名と設定ファイルの例」ではintra-martのApprication Runtimeのインストールディレクトリを"<ApplicationRuntime>"として表現している。

設定ファイルに記述する内容は「6 付録B DTD」を参照すること。

4 開発者の役割

本書で説明している Mapper や Model を利用して期間化や国際化を行う場合、開発者の役割はさまざまなものになる。本書では開発者を以下のような役割に分け、それぞれの役割担当者が行うべき作業を述べる。

- データ構造定義者
  ER 図を作成する。
- インタフェース定義者
  エンティティの定義をもとに Mapper と Model のインタフェースを定義する。
- データストア管理者
  エンティティ定義をもとにデータストア上にエンティティを定義する（データベース上にテーブルを作成する、ファイルシステムにディレクトリを構築するなど）。
- マッパ実装者
  エンティティ定義とインタフェースをもとに Mapper と Accessor を実装する。
- モデル実装者
  Model のインタフェースを実装する。内部では Accessor に処理を委譲するようにする。
- 記述子定義者
  データ構造、（データストアとしてリレーショナルデータベースを利用する場合は）テーブル定義、Mapper と Model の実装をもとに定義ファイルを作成する。
- アプリケーション開発者
  Mapper と Model のインタフェースを利用して開発をする。

これらの役割担当者と生成物の関連を「図 4-1 役割と生成物」に示す。

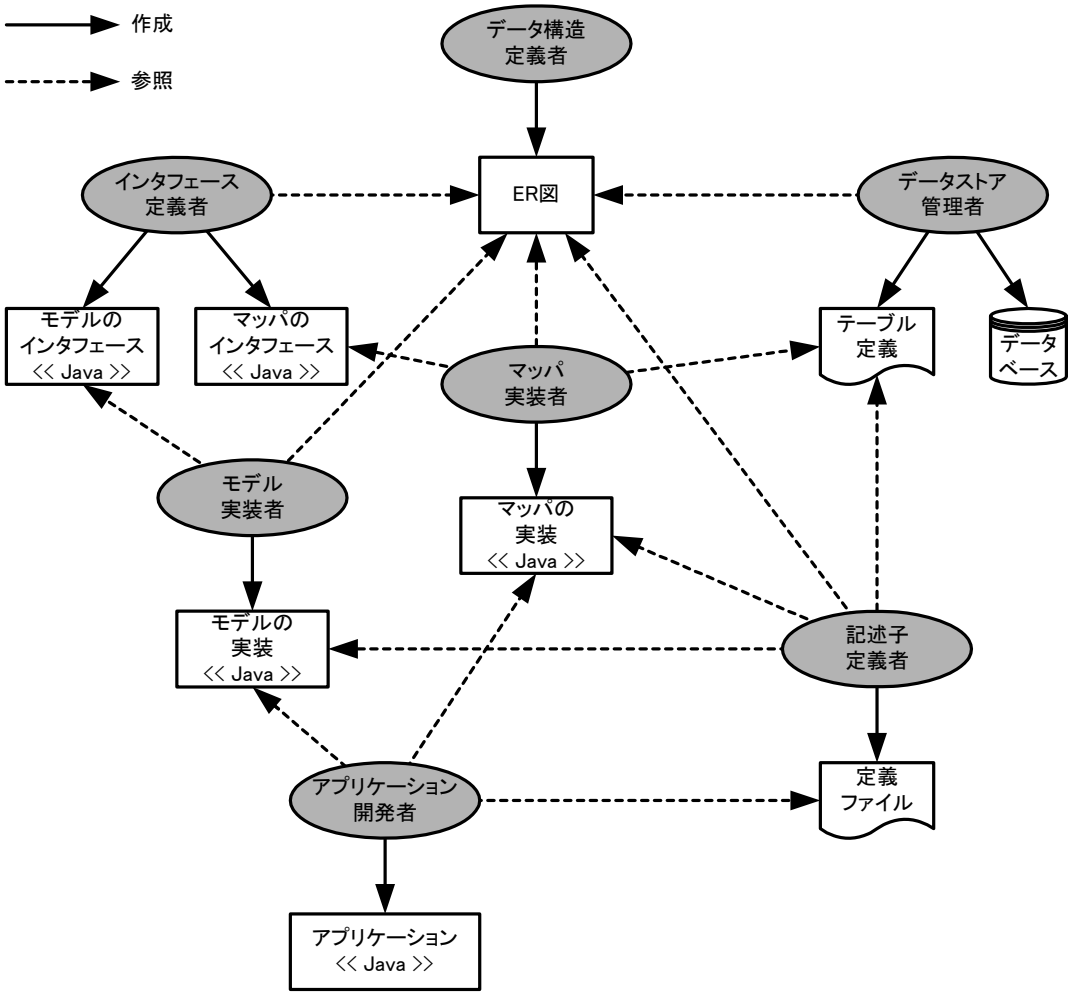


図 4-1 役割と生成物

4.1 データ構造定義者

データ構造定義者は業務要件をもとに「2.1.4 エンティティの表記法」や「2.2.1 リレーションシップの表記法」で述べたような表記法を利用してER図を作成する。また、それぞれの項目に対する属性型も定義する。

4.2 インタフェース定義者

エンティティの定義をもとに Mapper と Model のインタフェースを定義する。
インタフェース定義者は「4.1 データ構造定義者」によって定義されたエンティティをもとにマッパとモデルのインタフェースを定義する。このとき、以下の点に注意する。

- jp.co.intra_mart.foundation.datastore.common.model.Mapper インタフェースを拡張し、このエンティティに対応した Mapper インタフェースを作成する。エンティティの特性に応じた各種メソッドを用意してもよい。
- jp.co.intra_mart.foundation.datastore.common.model.BaseModel インタフェースを拡張し、このエンティティの基本属性スコープの属性に対応する setter および getter メソッドを必要に応じて含んだ BaseModel インタフェースを作成する。ただし、プライマリキーや更新情報に該当する属性の場合はこれらに対応する setter メソッドを定義しないこと。
- jp.co.intra_mart.foundation.datastore.common.model.InternationalModel インタフェースを拡張し、このエンティティの国際化属性スコープの属性に対応する setter および getter メソッドを必要に応じて含んだ InternationalModel インタフェースを作成する。
- jp.co.intra_mart.foundation.datastore.common.model.TerminableModel インタフェースを拡張し、このエン

ティティの期間化属性スコープの属性に対応する setter および getter メソッドを必要に応じて含んだ TerminableModel インタフェースを作成する。

- jp.co.intra_mart.foundation.datastore.common.model.TerminableInternationalModel インタフェースを拡張し、このエンティティの期間化属性スコープの属性に対応する setter および getter メソッドを必要に応じて含んだ TerminableInternationalModel インタフェースを作成する。

インタフェース定義者はまた、「4.1 データ構造定義者」によって定義された拡張エンティティをもとに拡張モデルのインタフェースを定義する。このとき、以下の点に注意する。

- jp.co.intra_mart.foundation.datastore.common.model.ExtendedBaseModel インタフェースを拡張し、この拡張エンティティの基本属性スコープの属性に対応する setter および getter メソッドを必要に応じて含んだ ExtendedBaseModel インタフェースを作成する。
- jp.co.intra_mart.foundation.datastore.common.model.ExtendedInternationalModel インタフェースを拡張し、この拡張エンティティの国際化属性スコープの属性に対応する setter および getter メソッドを必要に応じて含んだ ExtendedInternationalModel インタフェースを作成する。
- jp.co.intra_mart.foundation.datastore.common.model.ExtendedTerminableModel インタフェースを拡張し、この拡張エンティティの期間化属性スコープの属性に対応する setter および getter メソッドを必要に応じて含んだ ExtendedTerminableModel インタフェースを作成する。
- jp.co.intra_mart.foundation.datastore.common.model.ExtendedTerminableInternationalModel インタフェースを拡張し、この拡張エンティティの期間化属性スコープの属性に対応する setter および getter メソッドを必要に応じて含んだ ExtendedTerminableInternationalModel インタフェースを作成する。

4.3 データストア管理者

データストア管理者は「4.1 データ構造定義者」によって定義されたエンティティや拡張エンティティをもとに、システムで実際に利用するデータストア上における内容を定義し、実装する。この内容はデータストアの種類によって異なる[4]。

リレーショナルデータベースを利用する場合、領域や接続ユーザの作成、エンティティに該当するテーブルの定義と作成などがあげられる。この場合の注意点は「2.5 リレーショナルデータベースへのマッピング」を参照すること。また、以下の点に注意する。

- 「2.1.4 エンティティの表記法」で説明されている更新情報が定義されている場合、更新者に該当するカラムはVARCHAR(50)に、更新日時に該当するカラムはVARCHAR(19)にする。
- 「2.5 リレーショナルデータベースへのマッピング」で説明されているロケールに該当するカラムはVARCHAR(20)にする。
- 「2.5 リレーショナルデータベースへのマッピング」で説明されている期間IDに該当するカラムはVARCHAR(20)にする。

4.4 マッパ実装者

マッパ実装者はエンティティ定義、インタフェースおよび利用するデータストアの種類をもとにマッパおよびアクセサを実装する。

Intra-mart で管理されるリレーショナルデータベースを利用する場合、マッパは以下の条件を満たす必要がある。

- jp.co.intra_mart.foundation.datastore.common.model.IntramartDBMapper を継承している。
- 「4.2 インタフェース定義者」によって定義されたマッパのインタフェースを実装している。

[4] intra-mart Version 6.1 では intra-mart で管理されるリレーショナルデータベースのみを対象としている。

IntramartDBMapper を継承したマッパは内部でアクセサを持っているため、マッパ実装者は改めてアクセサを作成する必要はない。

4.5 モデル実装者

モデル実装者は「4.2 インタフェース定義者」によって定義されたモデルおよび拡張モデルのインタフェースを実装する。各getterおよびsetterメソッドはinitメソッドで渡されるアクセサに対して処理を委譲する。このとき、必要に応じて適切なデータ型の変換(クラス型とプリミティブ型の変換等)を行う。

モデルは関連する国際化モデルや期間化モデル、拡張モデルなどを管理する必要がある。これらの煩わしさを少なくするため、モデルを実装する際は以下のクラスを継承することを推奨する。

- jp.co.intra_mart.foundation.datastore.common.model.GenericBaseModel
- jp.co.intra_mart.foundation.datastore.common.model.GenericInternationalModel
- jp.co.intra_mart.foundation.datastore.common.model.GenericTerminableModel
- jp.co.intra_mart.foundation.datastore.common.model.GenericTerminableInternationalModel
- jp.co.intra_mart.foundation.datastore.common.model.ExtendedGenericBaseModel
- jp.co.intra_mart.foundation.datastore.common.model.ExtendedGenericInternationalModel
- jp.co.intra_mart.foundation.datastore.common.model.ExtendedGenericTerminableModel
- jp.co.intra_mart.foundation.datastore.common.model.ExtendedGenericTerminableInternationalModel

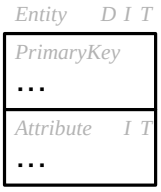
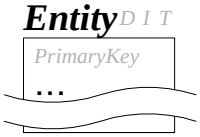
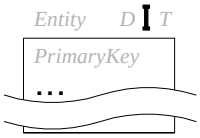
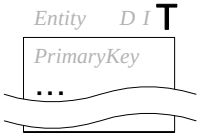
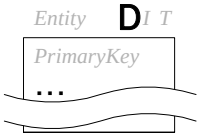
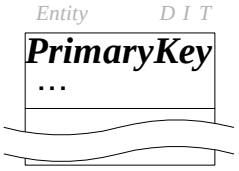
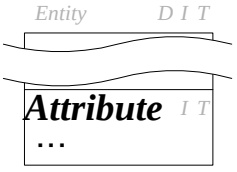
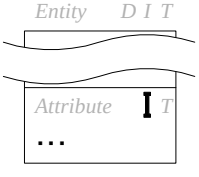
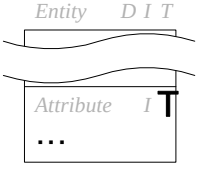
4.6 記述子定義者

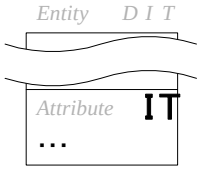
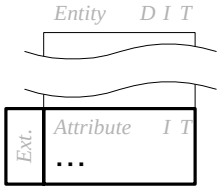
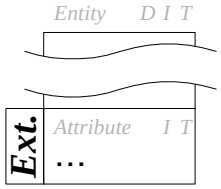
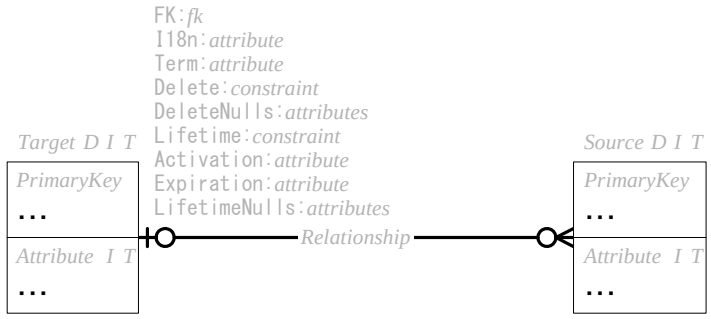
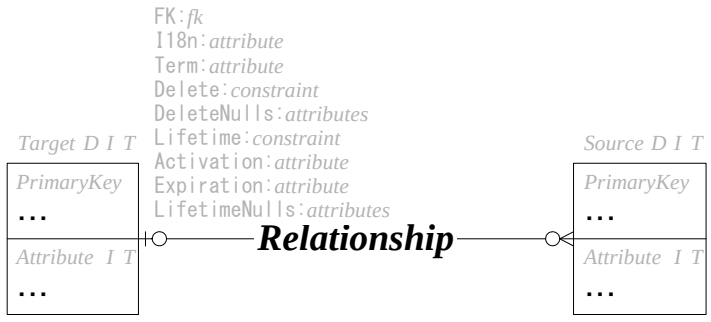
記述子定義者は「4.1 データ構造定義者」によって定義されたエンティティおよびリレーションシップ、(データストアとしてリレーショナルデータベースを利用する場合は)「4.3 データストア管理者」によって定義されたテーブル定義、「4.4 マッパ実装者」によって実装されたマッパおよび「4.5 モデル実装者」によって実装されたモデルをもとに定義ファイルを作成する。この内容については「3.4 設定」および「6 付録B DTD」を参照すること。

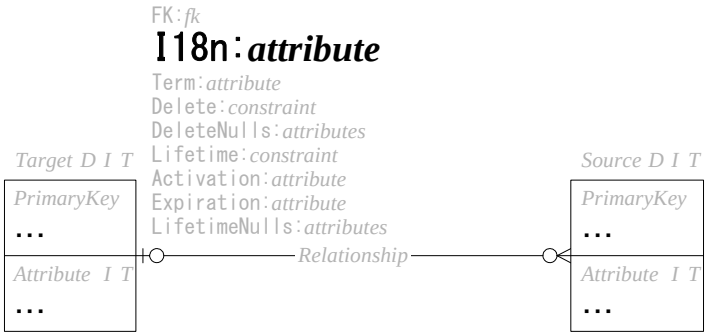
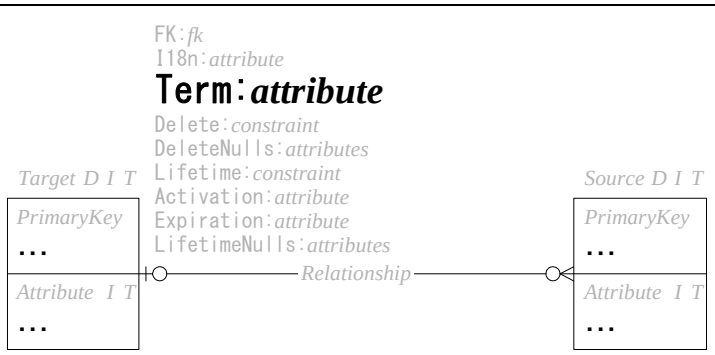
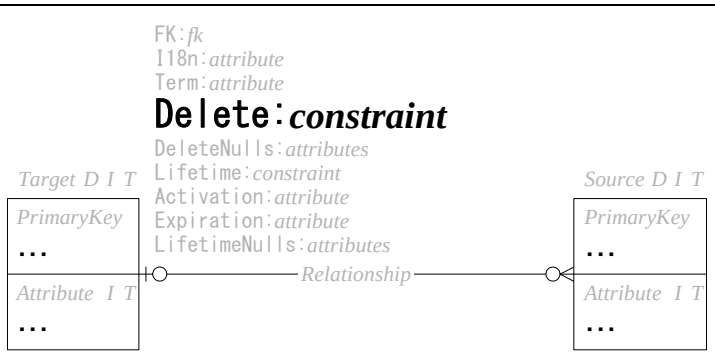
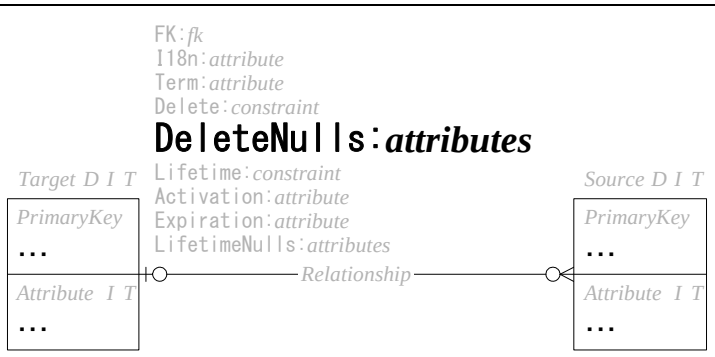
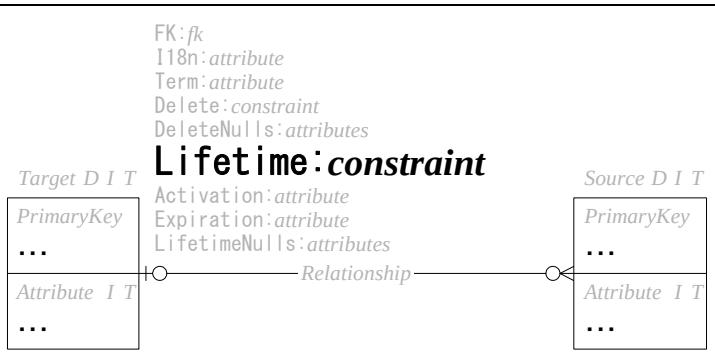
4.7 アプリケーション開発者

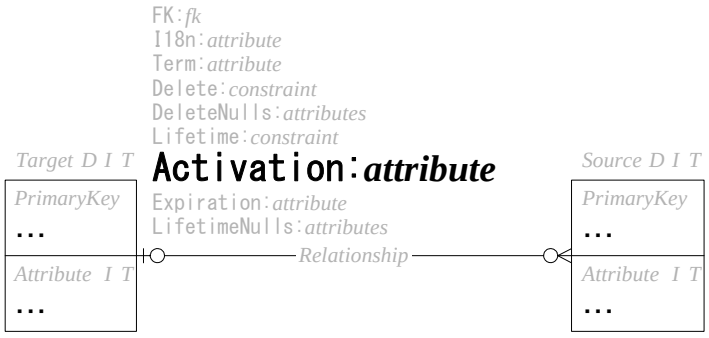
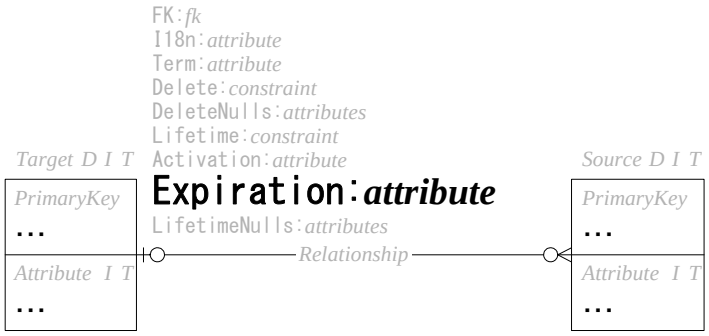
アプリケーション開発者は「4.2 インタフェース定義者」によって定義されたマッパおよびモデルのインタフェースを利用してアプリケーションを開発する。

5 付録 A intra-mart 拡張 ERD 表記法一覧

表記内容	表記
エンティティ: 	
エンティティ名: エンティティを一意に識別する名前	
エンティティの国際化: このエンティティが国際化可能であることを示すフラグ	
エンティティの期間化: このエンティティが期間化可能であることを示すフラグ	
エンティティの期間国際化: このエンティティが期間国際化可能であることを示すフラグ	
プライマリキー: このエンティティのプライマリキー	
従属属性: このエンティティに含まれるプライマリキー以外の属性	
従属属性の国際化: この属性が国際化可能であることを示すフラグ	
従属属性の期間化: この属性が期間化可能であることを示すフラグ	

従属属性の期間国際化: この属性が期間国際化可能であることを示すフラグ	
拡張エンティティ: エンティティに対して拡張して追加される従属属性	
拡張名: 拡張エンティティを一意に識別する名前	
リレーションシップ: 2 つのエンティティ(同一でもよい)間で定義される関連(参照先をターゲットエンティティ、参照元をソースエンティティと呼ぶ)	
リレーションシップ名: リレーションシップを一意に識別する名前	
外部キー制約: ターゲットエンティティのプライマリキーに該当するソースエンティティの属性名	

国際化基準項目: ターゲットエンティティの言語(ロケール)を一意に識別するソースエンティティまたは他のエンティティの属性名	
期間化基準項目: ターゲットエンティティの期間を一意に識別するソースエンティティまたは他のエンティティの属性名	
参照先削除時処理: ターゲットエンティティのデータが削除された場合にソースエンティティのデータに対して行う処理(Cascade、Null、Exception のいずれか)	
参照先削除時 Null 項目: ターゲットエンティティのインスタンスが削除された場合に null にするソースエンティティの属性名(Delete:Null のときのみ指定可能)	
参照先期間変更時処理: ターゲットエンティティの期間が更新された場合にソースエンティティのデータに対して行う処理(Cascade、Null、Exception のいずれか)	

有効期間開始日時項目: ターゲットエンティティの期間内に収めるべき期間の開始日時を示すソースエンティティまたは他のエンティティの属性名	
有効期間終了日時項目: ターゲットエンティティの期間内に収めるべき期間の終了日時を示すソースエンティティまたは他のエンティティの属性名	
参照先期間変更時 Null 項目: ターゲットエンティティの期間内に収めるべき期間の開始日時を示すソースエンティティまたは他のエンティティの属性名	

6 付録 B DTD

6.1 entities

<!ELEMENT entities (entity*, extended-entity*, relationship*, table*,
extended-table*, model*, extended-model*)>

設定ファイルのトップノードは常に<entities>となる。以下の情報を含む。

- entity
  エンティティの定義
- extended-entity
  拡張エンティティの定義
- relationship
  リレーションシップの定義
- table
  データベースにおけるテーブルの定義
- extended-table
  拡張テーブルの定義
- model
  モデルの定義
- extended-model
  拡張モデルの定義

6.2 entity

<!ELEMENT entity (entity-name, attribute+, primary-key, update-info?,
terminable?, international?, terminable-international?)>

エンティティを定義する。以下の情報を含む。

- entity-name
  エンティティ名
- attribute
  エンティティに含まれる属性
- primary-key
  エンティティのプライマリキー
- update-info
  最終更新情報の定義
- terminable
  期間化可能なエンティティかどうかを表すフラグ
- international
  国際化可能なエンティティかどうかを表すフラグ
- terminable-international
  期間国際化可能なエンティティかどうかを表すフラグ

6.3 entity-name

<!ELEMENT entity-name (#PCDATA)>

エンティティ名を定義する。エンティティ名は英数字および"_"の組合せのみが許される。

6.4 attribute

<!ELEMENT attribute (attribute-name, attribute-type, terminable?,
international?, null-acceptable?)>

属性を定義する。以下の情報を含む。

- attribute-name
  属性名
- attribute-type
  属性の型
- terminable
  期間化可能な属性かどうかを表すフラグ
- international
  国際化可能な属性かどうかを表すフラグ
- null-acceptable
  NULL 設定可能な属性かどうかを表すフラグ

6.5 attribute-name

<!ELEMENT attribute-name (#PCDATA)>

属性名を定義する。属性名は英数字および"_"の組合せのみが許される。

6.6 attribute-type

<!ELEMENT attribute-type (#PCDATA)>

属性の型を指定する。指定できる値は「表 2-1 属性の型」で示されているものである。

6.7 terminable

<!ELEMENT terminable (#PCDATA)>

期間化可能であるかどうかを指定する。指定できる値は以下のとおり。

<terminable>True</terminable>	期間化できる
<terminable>False</terminable>	期間化できない

6.8 international

<!ELEMENT international (#PCDATA)>

国際化可能であるかどうかを指定する。指定できる値は以下のとおり。

<international>True</international>	国際化できる
<international>False</international>	国際化できない

6.9 terminable-international

<!ELEMENT terminable-international (#PCDATA)>

期間国際化可能であるかどうかを指定する。指定できる値は以下のとおり。

<terminable-international>True</terminable-international>	期間国際化できる
<terminable-international>False</terminable-international>	期間国際化できない

6.10 null-acceptable

<!ELEMENT null-acceptable (#PCDATA)>

null を設定することが可能であるかどうかを指定する。指定できる値は以下のとおり。

<null-acceptable>True</null-acceptable>	null を設定できる
<null-acceptable>False</null-acceptable>	null を設定できない

6.11 primary-key

<!ELEMENT primary-key (attribute-name+)>

プライマリキーを定義する。以下の情報を含む。

- attribute-name
  プライマリキーとなる属性の属性名を指定する。

ここで指定された attribute-name の順番がそのままプライマリキーの順番と一致する。

6.12 update-info

<!ELEMENT update-info (update-user, update-timestamp)>

最終更新情報を示す属性を指定する。以下の情報を含む。

- update-user
  最終更新者を記録するための属性
- update-timestamp
  最終更新日時を記録するための属性

6.13 update-user

<!ELEMENT update-user (attribute-name)>

最終更新者を記録するための属性を指定する。以下の情報を含む。

- attribute-name
  最終更新者を記録する属性の属性名

6.14 update-timestamp

<!ELEMENT update-timestamp (attribute-name)>

最終更新日時を記録するための属性を指定する。以下の情報を含む。

- update-timestamp
  最終更新日時を記録する属性の属性名

6.15 extended-entity

<!ELEMENT extended-entity (extended-entity-name, application-name?,
entity-name, attribute+)>

拡張エンティティを定義する。以下の情報を含む。

- extended-entity-name
  拡張エンティティ名
- application-name
  拡張元となるエンティティが定義されているアプリケーション名（省略された場合はこの拡張エンティティと同一のアプリケーションが指定されたものとみなされる）
- entity-name
  拡張元となるエンティティのエンティティ名
- attribute
  この拡張エンティティに含まれる属性

6.16 application-name

<!ELEMENT application-name (#PCDATA)>

アプリケーション名を指定する。<ROOT>/foo/bar/myconfig.xml に情報が定義されている場合、アプリケーション名は foo.bar.myconfig となる。ここで<ROOT>は定義ファイルを配置するルートディレクトリを意味している。

6.17 extended-entity-name

<!ELEMENT extended-entity-name (#PCDATA)>

拡張エンティティ名を定義する。拡張エンティティ名は英数字および"_"の組合せのみが許される。

6.18 relationship

<!ELEMENT relationship (relationship-name, source, target, foreign-keys,
terminable-key?, international-key?, delete?, ((activation-key,
expiration-key)?, lifetime)?)>

エンティティ間のリレーションリップを定義する。以下の情報が含まれる。

- relationship-name
  リレーションシップ名
- source
  参照元となるエンティティ
- target
  参照先となるエンティティ
- foreign-keys
  外部キー
- terminable-key
  期間化基準項目
- international-key
  国際化基準項目
- delete
  参照先削除時の振る舞い
- activation-key
  有効期間開始日時項目
- expiration-key
  有効期間終了日時項目
- lifetime
  参照先期間変更時の振る舞い

6.19 relationship-name

<!ELEMENT relationship-name (#PCDATA)>

リレーションシップ名を定義する。リレーションシップ名は英数字および"_"の組合せのみが許される。

6.20 source

<!ELEMENT source (application-name?, entity-name)>

参照元となるエンティティを指定する。以下の情報を含む。

- application-name
  参照元となるエンティティが定義されているアプリケーション名（省略された場合はこのリレーションシップと同一のアプリケーションが指定されたものとみなされる）
- entity-name
  参照元となるエンティティのエンティティ名

6.21 target

<!ELEMENT target (application-name?, entity-name)>

参照先となるエンティティを指定する。以下の情報を含む。

- application-name
  参照先となるエンティティが定義されているアプリケーション名（省略された場合はこのリレーションシップと同一のアプリケーションが指定されたものとみなされる）
- entity-name
  参照元となるエンティティのエンティティ名

6.22 foreign-keys

<!ELEMENT foreign-keys (foreign-key+)>

外部キーを指定する。以下の情報を含む。

- foreign-key
  外部キーとなる属性

6.23 foreign-key

<!ELEMENT foreign-key (attribute-name)>

外部キーとなる属性を指定する。以下の情報を含む。

- attribute-name
  外部キーとなる属性の属性名

6.24 terminable-key

<!ELEMENT terminable-key (relationship-path*, attribute-name)>

期間化基準項目となる属性を指定する。以下の情報を含む。

- relationship-path
  期間化基準項目が指定されているエンティティへのパス（省略された場合は自エンティティ内で定義されているものとみなす）
- attribute-name
  期間化基準項目となる属性の属性名

6.25 relationship-path

<!ELEMENT relationship-path (application-name?, relationship-name)>

他のエンティティを指定するときのリレーションシップのパスを指定する。以下の情報が含まれる。

- application-name
  リレーションシップが定義されているアプリケーション名（省略された場合はこのリレーションシップパスと同一のアプリケーションが指定されたものとみなされる）
- relationship-name
  リレーションシップパスで使用するリレーションシップのリレーションシップ名

6.26 international-key

<!ELEMENT international-key (relationship-path*, attribute-name)>

国際化基準項目となる属性を指定する。以下の情報を含む。

- relationship-path
  国際化基準項目が指定されているエンティティへのパス（省略された場合は自エンティティ内で定義されているものとみなす）
- attribute-name
  国際化基準項目となる属性の属性名

6.27 delete

<!ELEMENT delete (delete-type, null-keys?)>

参照先のエンティティのデータが削除されたときの振る舞いを指定する。以下の情報を含む。

- delete-type
  参照先のエンティティのデータが削除されたときの振る舞い
- null-keys
  null にする属性（delete-type に Null が指定された場合のみ指定）

6.28 delete-type

<!ELEMENT delete-type (#PCDATA)>

参照先のデータが削除されたときの振る舞いを指定する。指定できる値は以下のとおり。

<delete-type>Cascade</delete-type>	参照元のデータも削除する
<delete-type>Null</delete-type>	参照元の外部キーに null を設定する
<delete-type>Exception</delete-type>	例外を発生させる

6.29 null-keys

<!ELEMENT null-keys (foreign-key+)>

参照先が削除されたときに null を設定する外部キーを指定する。以下の情報を含む。

- foreign-key
  null を設定する外部キー

6.30 activation-key

<!ELEMENT activation-key (relationship-path*, attribute-name)>

ライフタイム制約の有効期間開始日時を決定する属性を指定する。以下の情報を含む。

- relationship-path
  有効期間開始日時が指定されているエンティティへのパス（省略された場合は自エンティティ内で定義されているものとみなす）
- attribute-name
  有効期間開始日時となる属性の属性名

6.31 expiration-key

<!ELEMENT expiration-key (relationship-path*, attribute-name)>

ライフタイム制約の有効期間終了日時を決定する属性を指定する。以下の情報を含む。

- relationship-path
  有効期間終了日時が指定されているエンティティへのパス（省略された場合は自エンティティ内で定義されているものとみなす）
- attribute-name
  有効期間終了日時となる属性の属性名

6.32 lifetime

<!ELEMENT lifetime (lifetime-type, null-keys?)>

参照先のエンティティの有効期間が更新されたときの振る舞いを指定する。以下の情報を含む。

- lifetime-type
  参照先のエンティティのデータの有効期間が削除されたときの振る舞い
- null-keys
  null にする属性（lifetime-type に Null が指定された場合のみ指定）

6.33 lifetime-type

<!ELEMENT lifetime-type (#PCDATA)>

参照先のデータの期間が部分的または全体が削除されたときの振る舞いを指定する。指定できる値は以下のとおり。

<lifetime-type>Cascade</lifetime-type>	参照元の期間を参照先の期間内に収まるように削除する
<lifetime-type>Null</lifetime-type>	参照元に存在するが参照先には存在しない期間における外部キーに null を設定する
<lifetime-type>Exception</lifetime-type>	例外を発生させる

6.34 table

<!ELEMENT table (application-name?, entity-name, column+, base-table,
international-table?, terminable-table?, terminable-international-table?)>

エンティティをリレーショナルデータベースのテーブルにマッピングするための情報を定義する。以下の情報を含む。

- application-name
  関連付けられるエンティティが定義されているアプリケーション名（省略された場合はこのテーブルと同一のアプリケーションが指定されたものとみなされる）
- entity-name
  ここで定義するテーブルと関連付けられるエンティティのエンティティ名
- column
  テーブルに含まれるカラム
- base-table
  基本テーブル
- international-table
  国際化テーブル
- terminable-table
  期間化テーブル
- terminable-international-table
  期間国際化テーブル

6.35 column

<!ELEMENT column (attribute-name, column-type)>

属性をテーブルのカラムにマッピングするための情報を定義する。以下の情報を含む。

- attribute-name
  マッピングの対象となる属性の属性名
- column-type
  カラムの型

6.36 column-type

<!ELEMENT column-type (#PCDATA)>

カラムの型を定義する。指定できる値は以下のとおり。

VARCHAR	文字列型
DECIMAL	数値型

6.37 base-table

<!ELEMENT base-table (table-name, column-mapping+)>

基本テーブルの情報を定義する。以下の情報を含む。

- table-name
  基本テーブルのテーブル名
- column-mapping
  基本属性とカラムのマッピング情報

6.38 table-name

<!ELEMENT table-name (#PCDATA)>

テーブル名を定義する。

6.39 column-mapping

<!ELEMENT column-mapping (attribute-name, column-name)>

属性とカラムのマッピング情報を定義する。以下の情報を含む。

- attribute-name
  属性名
- column-name
  属性に関連付けられるカラムのカラム名

6.40 column-name

<!ELEMENT column-name (#PCDATA)>

カラム名を定義する。

6.41 international-table

<!ELEMENT international-table (table-name, locale-column, column-mapping+)>

国際化テーブルの情報を定義する。以下の情報を含む。

- table-name
  国際化テーブルのテーブル名
- locale-column
  ロケール ID となるカラム
- column-mapping
  国際化属性とカラムのマッピング情報

6.42 locale-column

<!ELEMENT locale-column (column-name)>

ロケール ID となるカラムを定義する。以下の情報を含む。

- column-name
  ロケール ID となるカラムのカラム名

6.43 terminable-table

<!ELEMENT terminable-table (table-name, term-column, start-column, end-column, column-mapping+)>

期間化テーブルの情報を定義する。以下の情報を含む。

- table-name
  期間化テーブルのテーブル名
- term-column
  期間コードとなるカラム
- start-column
  有効期間開始日時となるカラム
- end-column
  有効期間終了日時となるカラム
- column-mapping
  期間化属性とカラムのマッピング情報

6.44 term-column

<!ELEMENT term-column (column-name)>

期間コードとなるカラムを定義する。以下の情報を含む。

- column-name
  期間コードとなるカラムのカラム名

6.45 start-column

<!ELEMENT start-column (column-name)>

有効期間開始日時となるカラムを定義する。以下の情報を含む。

- column-name
  有効期間開始日時となるカラムのカラム名

6.46 end-column

<!ELEMENT end-column (column-name)>

有効期間終了日時となるカラムを定義する。以下の情報を含む。

- column-name
  有効期間終了日時となるカラムのカラム名

6.47 terminable-international-table

<!ELEMENT terminable-international-table (table-name, term-column,
locale-column, column-mapping+)>

期間国際化テーブルの情報を定義する。以下の情報を含む。

- table-name
  期間国際化テーブルのテーブル名
- term-column
  期間コードとなるカラム
- locale-column
  ロケール ID となるカラム
- column-mapping
  期間国際化属性とカラムのマッピング情報

6.48 extended-table

<!ELEMENT extended-table (application-name?, extended-entity-name, column+,
extended-base-table?, extended-international-table?,
extended-terminable-table?, extended-terminable-international-table?)>

拡張したエンティティと拡張テーブルとして用意されたテーブルをマッピングするための情報を定義する。以下の情報を含む。

- application-name
  関連付けられるエンティティが定義されているアプリケーション名（省略された場合はこのテーブルと同一のアプリケーションが指定されたものとみなされる）
- extended-entity-name
  ここで定義する拡張テーブルと関連付けられる拡張エンティティの拡張エンティティ名
- column
  拡張テーブルに含まれるカラム
- extended-base-table
  拡張基本テーブル
- extended-international-table
  拡張国際化テーブル
- extended-terminable-table
  拡張期間化テーブル
- extended-terminable-international-table
  拡張期間国際化テーブル

6.49 extended-base-table

<!ELEMENT extended-base-table (table-name, primary-key-mapping+,
column-mapping+)>

拡張基本テーブルの情報を定義する。以下の情報を含む。

- table-name
  拡張基本テーブルのテーブル名
- primary-key-mapping
  エンティティのプライマリキーと拡張基本テーブルのプライマリキーのマッピング情報
- column-mapping
  基本属性とカラムのマッピング情報

6.50 primary-key-mapping

<!ELEMENT primary-key-mapping (attribute-name, column-name)>

エンティティのプライマリキーと拡張テーブル内のカラムとのマッピング情報を定義する。以下の情報を含む。

- attribute-name
  プライマリキーとなる属性の属性名
- column-name
  プライマリキーとなるカラムのカラム名

6.51 extended-international-table

<!ELEMENT extended-international-table (table-name, primary-key-mapping+,
locale-column, column-mapping+)>

拡張国際化テーブルの情報を定義する。以下の情報を含む。

- table-name
  拡張国際化テーブルのテーブル名
- primary-key-mapping
  エンティティのプライマリキーと拡張国際化テーブルのプライマリキーのマッピング情報
- locale-column
  ロケール ID となるカラム
- column-mapping
  拡張国際化属性とカラムのマッピング情報

6.52 extended-terminable-table

<!ELEMENT extended-terminable-table (table-name, primary-key-mapping+,
term-column, column-mapping+)>

拡張期間化テーブルの情報を定義する。以下の情報を含む。

- table-name
  拡張期間化テーブルのテーブル名
- primary-key-mapping
  エンティティのプライマリキーと拡張期間化テーブルのプライマリキーのマッピング情報
- term-column
  期間コードとなるカラム
- column-mapping
  拡張期間化属性とカラムのマッピング情報

6.53 extended-terminable-international-table

<!ELEMENT extended-terminable-international-table (table-name,
primary-key-mapping+, term-column, locale-column, column-mapping+)>

拡張期間国際化テーブルの情報を定義する。以下の情報を含む。

- table-name
  拡張期間国際化テーブルのテーブル名
- primary-key-mapping
  エンティティのプライマリキーと拡張期間国際化テーブルのプライマリキーのマッピング情報
- term-column
  期間コードとなるカラム
- locale-column
  ロケール ID となるカラム
- column-mapping
  拡張期間国際化属性とカラムのマッピング情報

6.54 model

<!ELEMENT model (application-name?, entity-name, property+, mapper-class,
base-model, international-model?, terminable-model?,
terminable-international-model?)>

エンティティをモデルやマッパにマッピングするための情報を定義する。以下の情報を含む。

- application-name
  関連付けられるエンティティが定義されているアプリケーション名（省略された場合はこのモデルと同一のアプリケーションが指定されたものとみなされる）
- entity-name
  ここで定義するモデルと関連付けられるエンティティのエンティティ名
- property
  モデルで扱うプロパティ
- mapper-class
  このエンティティに対するマッパのクラス
- base-model
  このエンティティの基本情報に対するモデル
- international-model
  このエンティティの国際化情報に対するモデル
- terminable-model
  このエンティティの期間化情報に対するモデル
- terminable-international-model
  このエンティティの期間国際化情報に対するモデル

6.55 property

<!ELEMENT property (attribute-name, property-type)>

属性をモデルのプロパティにマッピングするための情報を定義する。以下の情報を含む。

- attribute-name
  マッピングの対象となる属性の属性名
- property-type
  プロパティの型

6.56 property-type

<!ELEMENT property-type (#PCDATA)>

プロパティの型を定義する。指定できる値は以下のとおり。

Java.lang.String	文字列型
java.lang.Integer	整数型
java.lang.Long	整数型
java.util.Date	日付型
java.lang.Float	浮動小数点型
java.lang.Double	浮動小数点型
Java.math.BigDecimal	小数点型
java.util.Locale	ロケール

6.57 mapper-class

<!ELEMENT mapper-class (#PCDATA)>

マッパのクラス名を指定する。クラス名はパッケージを含む完全な形式で指定する。

6.58 base-model

<!ELEMENT base-model (base-model-class)>

基本モデルを定義する。以下の情報を含む。

- base-model-class
  基本モデルのクラス名

6.59 base-model-class

<!ELEMENT base-model-class (#PCDATA)>

基本モデルのクラス名を指定する。クラス名はパッケージを含む完全な形式で指定する。

6.60 international-model

<!ELEMENT international-model (international-model-class)>

国際化モデルを定義する。以下の情報を含む。

- international-model-class
  国際化モデルのクラス名

6.61 international-model-class

<!ELEMENT international-model-class (#PCDATA)>

国際化モデルのクラス名を指定する。クラス名はパッケージを含む完全な形式で指定する。

6.62 terminable-model

<!ELEMENT terminable-model (terminable-model-class)>

期間化モデルを定義する。以下の情報を含む。

- terminable-model-class
  期間化モデルのクラス名

6.63 terminable-model-class

<!ELEMENT terminable-model-class (#PCDATA)>

期間化モデルのクラス名を指定する。クラス名はパッケージを含む完全な形式で指定する。

6.64 terminable-international-model

<!ELEMENT terminable-international-model (terminable-international-model-class)>

期間国際化モデルを定義する。以下の情報を含む。

- terminable-international-model-class
  期間国際化モデルのクラス名

6.65 terminable-international-model-class

<!ELEMENT terminable-international-model-class (#PCDATA)>

期間国際化モデルのクラス名を指定する。クラス名はパッケージを含む完全な形式で指定する。

6.66 extended-model

<!ELEMENT extended-model (application-name?, extended-entity-name, property+,
extended-base-model?, extended-international-model?,
extended-terminable-model?, extended-terminable-international-model?)>

拡張したエンティティと拡張モデルをマッピングするための情報を定義する。以下の情報を含む。

- application-name
  関連付けられる拡張エンティティが定義されているアプリケーション名（省略された場合はこのモデルと同一のアプリケーションが指定されたものとみなされる）
- extended-entity-name
  ここで定義する拡張モデルと関連付けられる拡張エンティティの拡張エンティティ名
- property
  拡張モデルで扱うプロパティ
- extended-base-model
  この拡張エンティティの基本情報に対する拡張モデル
- extended-international-model
  この拡張エンティティの国際化情報に対する拡張モデル
- extended-terminable-model
  この拡張エンティティの期間化情報に対する拡張モデル
- extended-terminable-international-model
  この拡張エンティティの期間国際化情報に対する拡張モデル

6.67 extended-base-model

<!ELEMENT extended-base-model (extended-base-model-class)>

拡張基本モデルを定義する。以下の情報を含む。

- extended-base-model-class
  拡張基本モデルのクラス名

6.68 extended-base-model-class

<!ELEMENT extended-base-model-class (#PCDATA)>

拡張基本モデルのクラス名を指定する。クラス名はパッケージを含む完全な形式で指定する。

6.69 extended-international-model

<!ELEMENT extended-international-model (extended-international-model-class)>

拡張国際化モデルを定義する。以下の情報を含む。

- extended-international-model-class
  拡張国際化モデルのクラス名

6.70 extended-international-model-class

<!ELEMENT extended-international-model-class (#PCDATA)>

拡張国際化モデルのクラス名を指定する。クラス名はパッケージを含む完全な形式で指定する。

6.71 extended-terminable-model

<!ELEMENT extended-terminable-model (extended-terminable-model-class)>

拡張期間化モデルを定義する。以下の情報を含む。

- extended-terminable-model-class
  拡張期間化モデルのクラス名

6.72 extended-terminable-model-class

<!ELEMENT extended-terminable-model-class (#PCDATA)>

拡張期間化モデルのクラス名を指定する。クラス名はパッケージを含む完全な形式で指定する。

6.73 extended-terminable-international-model

<!ELEMENT extended-terminable-international-model
(extended-terminable-international-model-class)>

拡張期間国際化モデルを定義する。以下の情報を含む。

- extended-terminable-international-model-class
  拡張期間国際化モデルのクラス名

6.74 extended-terminable-international-model-class

<!ELEMENT extended-terminable-international-model-class (#PCDATA)>

拡張期間国際化モデルのクラス名を指定する。クラス名はパッケージを含む完全な形式で指定する。

7 付録 C ER 定義時の制約

7.1 基準項目

7.1.1 基準項目は外部キーと同一エンティティ上で定義

国際化基準項目または期間化基準項目が外部キーと同一のエンティティ上で定義されている場合、国際化基準項目または期間化基準項目の属性スコープと外部キースコープの最下層の属性スコープは一致していなければならない。

外部キーの属性スコープの組合せ	国際化基準項目または 期間化基準項目の属性スコープ	設定可否
基本属性スコープ	基本属性スコープ	○
	期間化属性スコープ	×
	国際化属性スコープ	×
	期間国際化属性スコープ	×
期間化属性スコープ	基本属性スコープ	×
	期間化属性スコープ	○
	国際化属性スコープ	×
	期間国際化属性スコープ	×
国際化属性スコープ	基本属性スコープ	×
	期間化属性スコープ	×
	国際化属性スコープ	○
	期間国際化属性スコープ	×
期間国際化属性スコープ	基本属性スコープ	×
	期間化属性スコープ	×
	国際化属性スコープ	×
	期間国際化属性スコープ	○
基本属性スコープ 期間化属性スコープ	基本属性スコープ	×
	期間化属性スコープ	○
	国際化属性スコープ	×
	期間国際化属性スコープ	×
基本属性スコープ 国際化属性スコープ	基本属性スコープ	×
	期間化属性スコープ	×
	国際化属性スコープ	×
	期間国際化属性スコープ	×
基本属性スコープ 期間国際化属性スコープ	基本属性スコープ	×
	期間化属性スコープ	×
	国際化属性スコープ	×
	期間国際化属性スコープ	×
基本属性スコープ 期間化属性スコープ 期間国際化属性スコープ	基本属性スコープ	×
	期間化属性スコープ	×
	国際化属性スコープ	×
	期間国際化属性スコープ	×
期間化属性スコープ 期間国際化属性スコープ	基本属性スコープ	×
	期間化属性スコープ	×
	国際化属性スコープ	×
	期間国際化属性スコープ	×

7.1.2 基準項目は外部キーと異なるエンティティ上で定義

国際化基準項目または期間化基準項目が外部キーと異なるエンティティ上で定義される場合、国際化基準項目または期間化基準項目を検索するための外部キーの外部キースコープはターゲットエンティティへの外部キーの外部キースコープに包含されている必要がある。

ターゲットエンティティに対する外部キーの属性スコープの組合せ	国際化基準項目または期間化基準項目が定義されているエンティティを検索するための外部キースコープの最下層の属性スコープ	設定可否
基本属性スコープ	基本属性スコープ	○
	期間化属性スコープ	×
	国際化属性スコープ	×
	期間国際化属性スコープ	×
期間化属性スコープ	基本属性スコープ	○
	期間化属性スコープ	○
	国際化属性スコープ	×
	期間国際化属性スコープ	×
国際化属性スコープ	基本属性スコープ	○
	期間化属性スコープ	×
	国際化属性スコープ	○
	期間国際化属性スコープ	×
期間国際化属性スコープ	基本属性スコープ	○
	期間化属性スコープ	○
	国際化属性スコープ	×
	期間国際化属性スコープ	○
基本属性スコープ 期間化属性スコープ	基本属性スコープ	○
	期間化属性スコープ	○
	国際化属性スコープ	×
	期間国際化属性スコープ	×
基本属性スコープ 国際化属性スコープ	基本属性スコープ	○
	期間化属性スコープ	×
	国際化属性スコープ	○
	期間国際化属性スコープ	×
基本属性スコープ 期間国際化属性スコープ	基本属性スコープ	○
	期間化属性スコープ	○
	国際化属性スコープ	×
	期間国際化属性スコープ	○
基本属性スコープ 期間化属性スコープ 期間国際化属性スコープ	基本属性スコープ	○
	期間化属性スコープ	○
	国際化属性スコープ	×
	期間国際化属性スコープ	○
期間化属性スコープ 期間国際化属性スコープ	基本属性スコープ	○
	期間化属性スコープ	○
	国際化属性スコープ	×
	期間国際化属性スコープ	○

7.1.2.1 例 1：OK

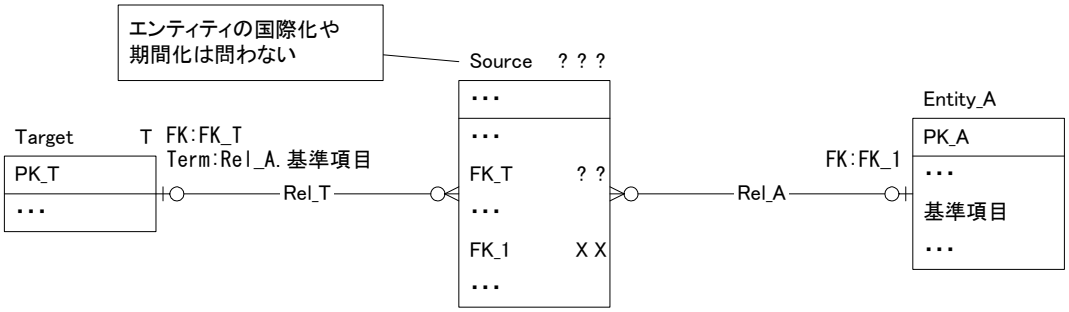


図 付録 C-1 例 1

ソースエンティティ（Source）は期間化基準項目を外部のエンティティ（Entity_A）で指定している。このとき、

- ターゲットエンティティ（Target）へ参照するときの外部キーの外部キースコープは未定
- 基準項目が存在するエンティティへ参照するときの外部キースコープは基本属性スコープ

となっているため、ターゲットエンティティへ参照するときの外部キーの外部キースコープは、基準項目が存在するエンティティへ参照するときの外部キースコープと同じか小さくなるため妥当である。

7.1.2.2 例 2：OK

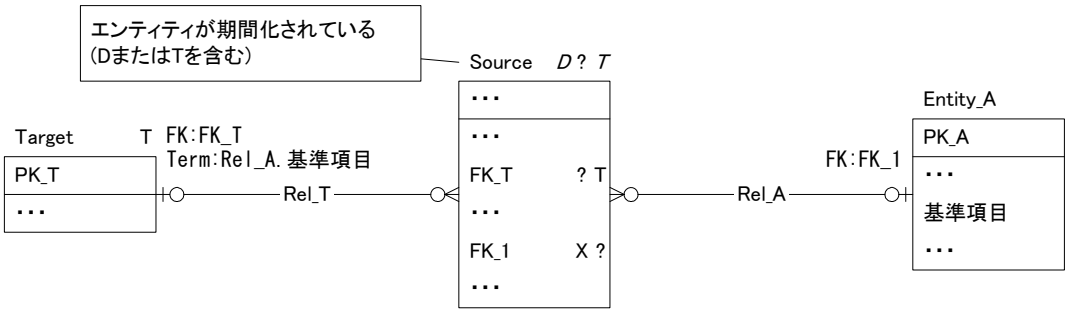


図 付録 C-2 例 2

ソースエンティティ（Source）は期間化基準項目を外部のエンティティ（Entity_A）で指定している。このとき、

- ターゲットエンティティ（Target）へ参照するときの外部キーの外部キースコープは期間化属性スコープまたは期間国際化属性スコープ
- 基準項目が存在するエンティティへ参照するときの外部キースコープは基本属性スコープまたは期間化属性スコープ

となっているため、ターゲットエンティティへ参照するときの外部キーの外部キースコープは、基準項目が存在するエンティティへ参照するときの外部キースコープと同じか小さくなるため妥当である。

7.1.2.3 例 3:OK

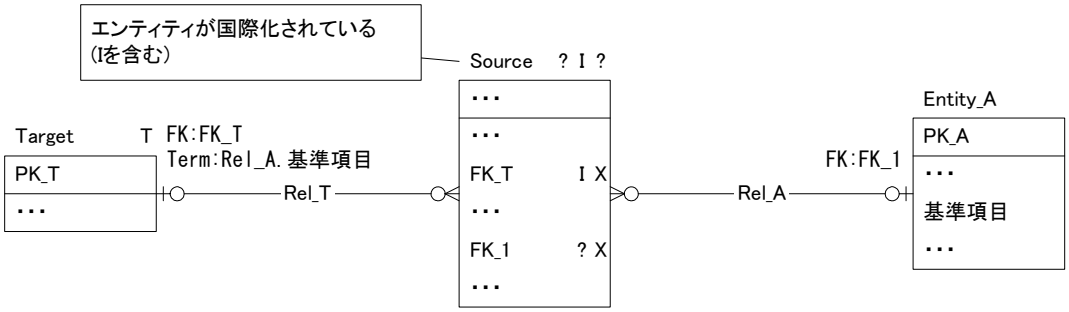


図 付録 C-3 例 3

ソースエンティティ(Source)は期間化基準項目を外部のエンティティ(Entity_A)で指定している。このとき、

- ターゲットエンティティ(Target)へ参照するときの外部キーの外部キースコープは国際化属性スコープ
- 基準項目が存在するエンティティへ参照するときの外部キースコープは基本属性スコープまたは国際化属性スコープ

となっているため、ターゲットエンティティへ参照するときの外部キーの外部キースコープは、基準項目が存在するエンティティへ参照するときの外部キースコープと同じか小さくなるため妥当である。

7.1.2.4 例 4:NG

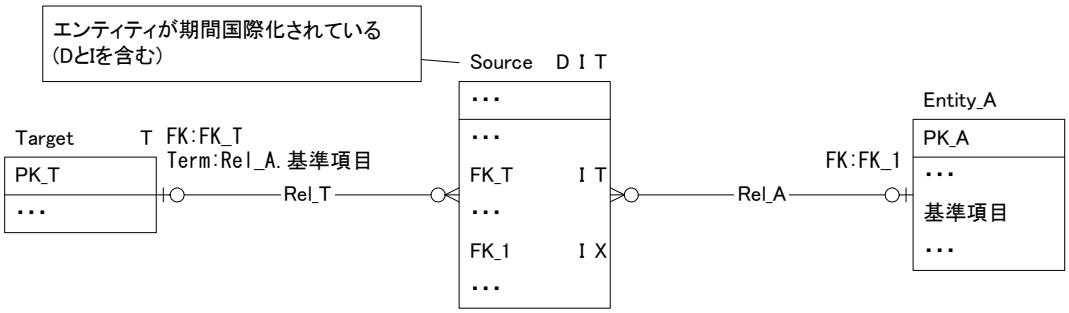


図 付録 C-4 例 4

ソースエンティティ(Source)は期間化基準項目を外部のエンティティ(Entity_A)で指定している。このとき、

- ターゲットエンティティ(Target)へ参照するときの外部キーの外部キースコープは期間国際化属性スコープ
- 基準項目が存在するエンティティへ参照するときの外部キースコープは国際化属性スコープ

となっているため、ターゲットエンティティへ参照するときの外部キーの外部キースコープは、基準項目が存在するエンティティへ参照するときの外部キースコープに含まれないため設定できない。

7.1.2.5 例 5:OK

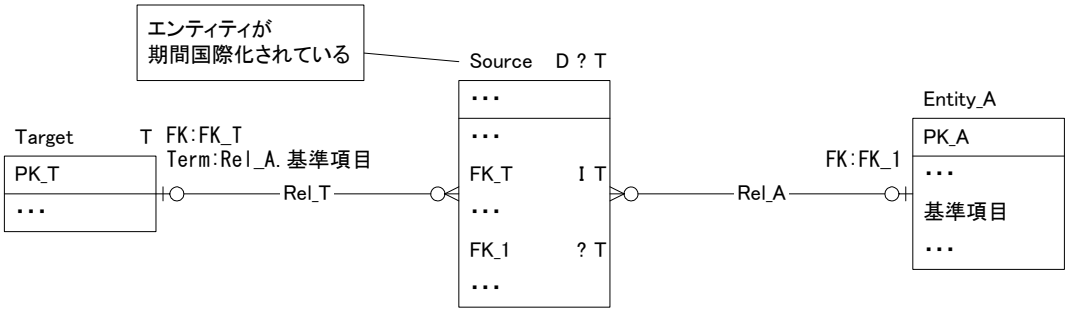


図 付録 C-5 例 5

ソースエンティティ(Source)は期間化基準項目を外部のエンティティ(Entity_A)で指定している。このとき、

- ターゲットエンティティ(Target)へ参照するときの外部キーの外部キースコープは期間国際化属性スコープ
- 基準項目が存在するエンティティへ参照するときの外部キースコープは期間化属性スコープまたは期間国際化属性スコープ

となっているため、ターゲットエンティティへ参照するときの外部キーの外部キースコープは、基準項目が存在するエンティティへ参照するときの外部キースコープと同じか小さくなるため妥当である。

7.1.2.6 例 6:NG

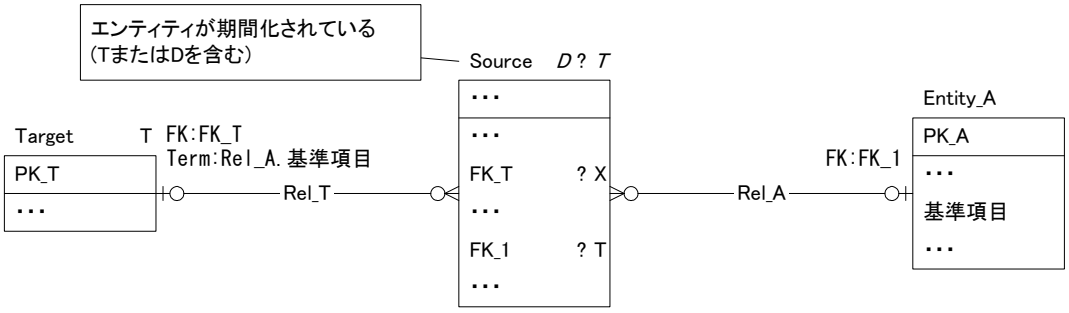


図 付録 C-6 例 6

ソースエンティティ(Source)は期間化基準項目を外部のエンティティ(Entity_A)で指定している。このとき、

- ターゲットエンティティ(Target)へ参照するときの外部キーの外部キースコープは基本属性スコープまたは国際化属性スコープ
- 基準項目が存在するエンティティへ参照するときの外部キースコープは期間化属性スコープまたは期間国際化属性スコープ

となっているため、ターゲットエンティティへ参照するときの外部キーの外部キースコープは、基準項目が存在するエンティティへ参照するときの外部キースコープに含まれないため設定できない。

7.1.2.7 例 7:NG

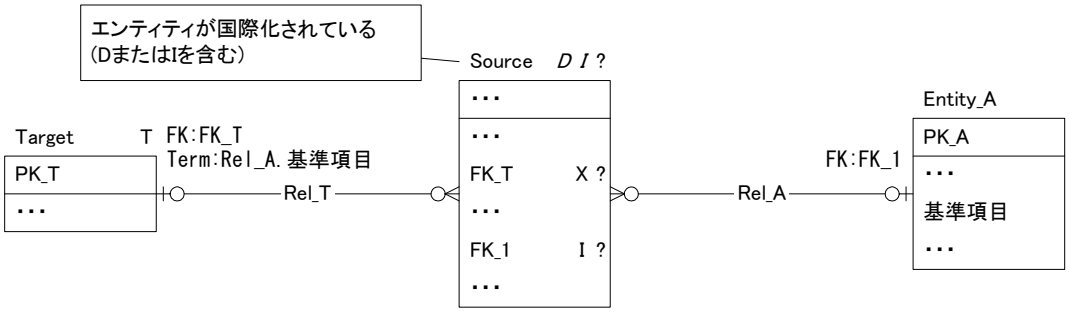


図 付録 C-7 例 7

ソースエンティティ(Source)は期間化基準項目を外部のエンティティ(Entity_A)で指定している。このとき、

- ターゲットエンティティ(Target)へ参照するときの外部キーの外部キースコープは基本属性スコープまたは期間化属性スコープ
- 基準項目が存在するエンティティへ参照するときの外部キースコープは国際化属性スコープまたは期間国際化属性スコープ

となっているため、ターゲットエンティティへ参照するときの外部キーの外部キースコープは、基準項目が存在するエンティティへ参照するときの外部キースコープよりも小さくなることが保障されないため設定できない。

7.1.2.8 例 8:OK

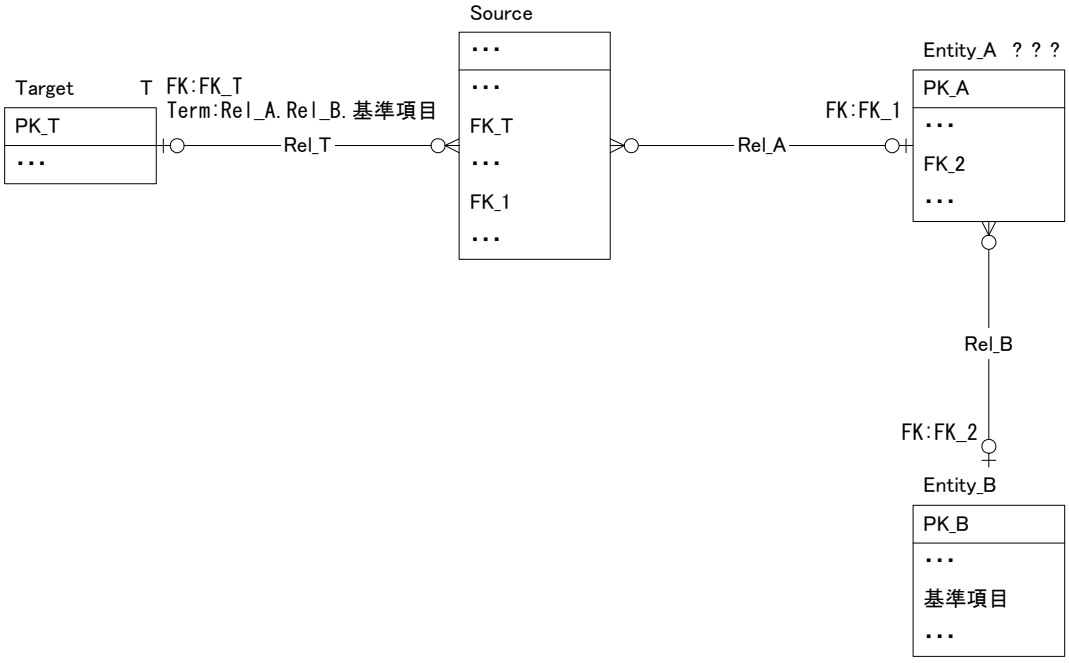


図 付録 C-8 例 8

ソースエンティティ(Source)は期間化基準項目を外部のエンティティ(Entity_B)で指定している。このとき、

- ターゲットエンティティ(Target)へ参照するときの外部キーの外部キースコープは基本属性スコープ
- 中間エンティティ(Entity_A)へ参照するときの外部キーの外部キースコープは基本属性スコープ(つまり、ターゲットエンティティへ参照するときの外部キーの外部キースコープと同一)
- 中間エンティティにおいて、基準項目が存在するエンティティへ参照するときの外部キースコープは基本属性スコープ

となっているため、基準項目への参照は一意に決定し、かつターゲットエンティティへ参照するときの外部キーの

外部キースコープは、基準項目が存在するエンティティへの参照が開始されるときの外部キースコープと同じになるため妥当である。

7.1.2.9 例 9：OK

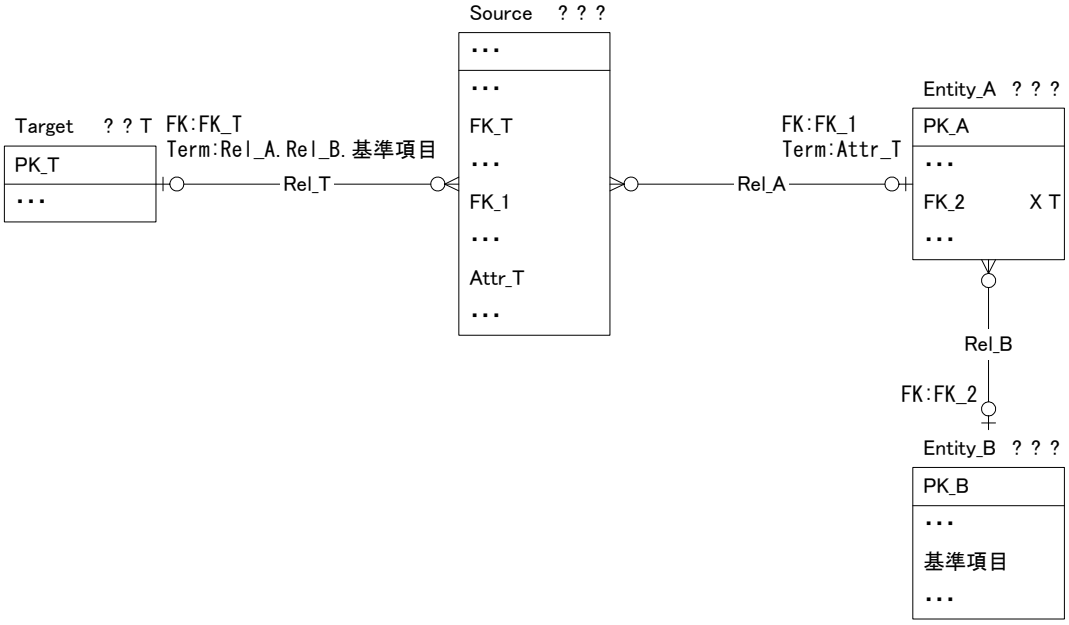


図 付録 C-9 例 9

ソースエンティティ（Source）は期間化基準項目を外部のエンティティ（Entity_B）で指定している。このとき、

- ターゲットエンティティ（Target）へ参照するときの外部キーの外部キースコープは基本属性スコープ
- 中間エンティティ（Entity_A）へ参照するときの外部キーの外部キースコープは基本属性スコープ（つまり、ターゲットエンティティへ参照するときの外部キーの外部キースコープと同一）
- 中間エンティティにおいて、基準項目が存在するエンティティへ参照するときの外部キースコープは期間化属性スコープ（ただし、ソースエンティティから中間エンティティへの参照における期間化基準項目はソースエンティティ内に存在する）

となっているため、基準項目への参照は一意に決定し、かつターゲットエンティティへ参照するときの外部キーの外部キースコープは、基準項目が存在するエンティティへの参照が開始されるときの外部キースコープと同じになるため妥当である。

7.1.2.10 例 10:OK

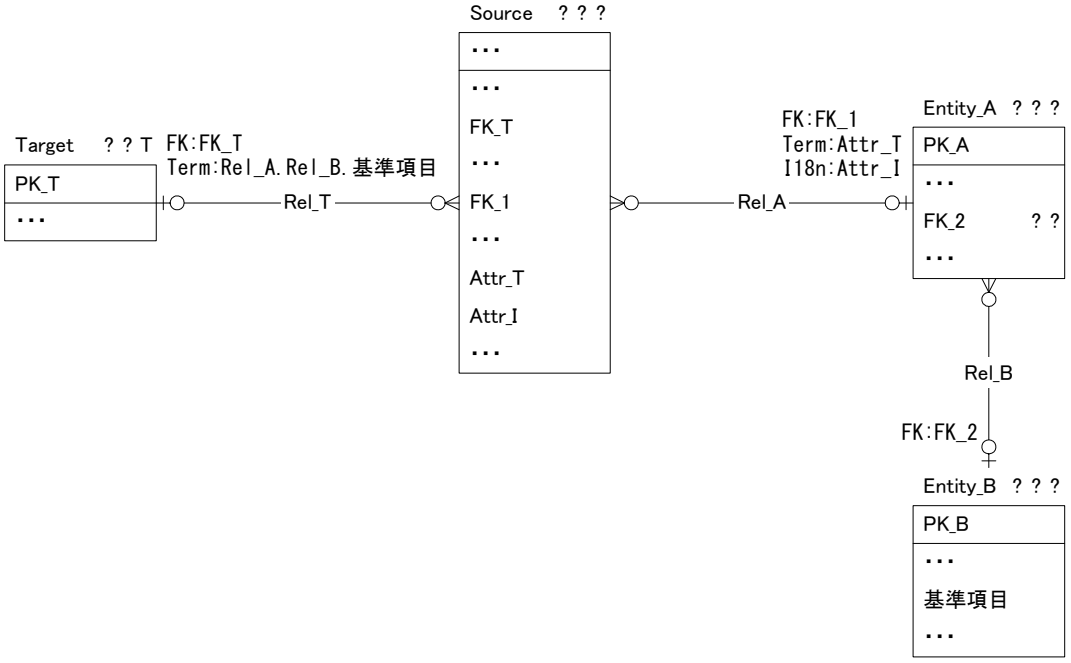


図 付録 C-10 例 10

ソースエンティティ(Source)は期間化基準項目を外部のエンティティ(Entity_B)で指定している。このとき、

- ターゲットエンティティ(Target)へ参照するときの外部キーの外部キースコープは基本属性スコープ
- 中間エンティティ(Entity_A)へ参照するときの外部キーの外部キースコープは基本属性スコープ(つまり、ターゲットエンティティへ参照するときの外部キーの外部キースコープと同一)
- 中間エンティティにおいて、基準項目が存在するエンティティへ参照するときの外部キースコープは期間国際化属性スコープ(ただし、ただし、ソースエンティティから中間エンティティへの参照における期間化基準項目および国際化基準項目はソースエンティティ内に存在する)

となっているため、基準項目への参照は一意に決定し、かつターゲットエンティティへ参照するときの外部キーの外部キースコープは、基準項目が存在するエンティティへの参照が開始されるときの外部キースコープと同じになるため妥当である。

8 付録 D 期間管理について

8.1 有効期間

アプリケーション共通マスタの期間化テーブルで扱う期間は、開始時刻と終了時刻から構成され以下の式を満たす範囲が有効期間として認められます。
開始時間 ≦ 有効期間 ＜ 終了時間

また、システムで有効な期間の範囲は、以下の式で定義されます。
1582/10/15 00:00:00 ≦ 有効期間 ＜ 9999/12/31 23:59:59

アプリケーション共通マスタでは期間を jp.co.intra_mart.foundation.common.structure.Term クラスにより管理されます。Term クラスの getMinDate メソッドによりシステム最小有効時間（1582/10/15 00:00:00）を、getMaxDate メソッドによりシステム最大有効時間（9999/12/31 23:59:59）を取得することが出来ます。

8.2 時刻の切り捨て

Term クラスでは期間を構成する時間の時刻は 00 時 00 分 00 秒に切り捨てられます。ただし、期間を構成する時間にシステム最大有効時間が設定された場合は、時刻の切捨ては行いません。
具体的には、Term クラスで設定される時間は以下のとおりになります。

Term クラスに設定する時間	Term クラス内の時間
1582/10/15 00:00:00	1582/10/15 00:00:00
1582/10/15 23:59:59	1582/10/15 00:00:00
2005/01/01 12:00:00	2005/01/01 00:00:00
9999/12/31 23:59:50	9999/12/31 00:00:00
9999/12/31 23:59:59	9999/12/31 23:59:59

参考資料

[1] Java 2 Platform, Standard Edition (J2SE)
http://java.sun.com/j2se/1.4/
[2] Java 2 Platform, Enterprise Edition (J2EE)
http://java.sun.com/j2ee/1.3/docs/
[3] Extensible Markup Language (XML) 1.1
http://www.w3.org/TR/xml11/

intra-mart WebPlatform/AppFramework Ver.7.2
アプリケーション共通マスタ API ガイドライン

2010/04/01 初版

TEL: 03-5549-2821
FAX: 03-5549-2816
E-MAIL: info@intra-mart.jp
URL: http://www.intra-mart.jp/