Copyright © 2012 NTT DATA INTRAMART CORPORATION

目次

- 1. 改訂情報
- 2. IM-共通マスタについて
  - 2.1. IM-共通マスタ
    - 2.1.1. 目的
    - 2.1.2. 統合化の範囲と要件
    - 2.1.3. データ構造
    - 2.1.4. 複数会社対応
    - 2.1.5. オートコンプリート
- 3. 構造
  - 3.1. ユーザ
    - 3.1.1. プロファイルとアカウント
  - 3.2. ユーザ分類
    - 3.2.1. データ構造
  - 3.3. 会社と組織
    - 3.3.1. データ構造
  - 3.4. 会社グループ
    - 3.4.1. データ構造
  - 3.5. パブリックグループ
    - 3.5.1. データ構造
  - 3.6. プライベートグループ
    - 3.6.1. データ構造
    - 3.6.2. プライベートグループの情報
  - 3.7. 品目
    - 3.7.1. データ構造
  - 3.8. 法人・取引先
    - 3.8.1. データ構造
  - 3.9. 法人グループ
    - 3.9.1. データ構造
  - 3.10. 通貨
    - 3.10.1. データ構造
  - 3.11. その他
- 4. API
  - 4.1. IM-共通マスタとアプリケーションデータの整合性について
    - 4.1.1. IM-共通マスタ間の整合性の確保
    - 4.1.2. テーブル拡張の実現
  - 4.2. マネージャ
    - 4.2.1. マネージャの取得
    - 4.2.2. マネージャの使用
    - 4.2.3. マネージャによる検索
  - 4.3. メソッド命名則
    - 4.3.1. 記法について
    - 4.3.2. コンストラクタについて
    - 4.3.3. 基本構造と所属構造
    - 4.3.4. 内包構造
    - 4.3.5. 期間操作
  - 4.4. リスナー
    - 4.4.1. リスナーの種類
    - 4.4.2. リスナーの動作

- 4.5. 補足事項
- 5. 付録
  - 5.1. キャッシュ
    - 5.1.1. 概要
    - 5.1.2. 会社一覧キャッシュ
    - 5.1.3. 会社一覧のリソースグループ設定キャッシュ
    - 5.1.4. 会社一覧のポリシー設定キャッシュ
    - 5.1.5. プロファイル画像キャッシュ
  - 5.2. IM-LogicDesigner のフロートリガで IM-共通マスタ を使用する
    - 5.2.1. 設定手順

改訂情報

変更年月日	変更内容
2012-10-01	初版作成
2014-04-01	第2版 「*検索画面* 」を修正しました。 「*メンテナンス画面* 」を修正しました。
2014-08-01	第3版 「*データ構造* 」に制約を追記しました。
2015-12-01	第4版 「*キャッシュ* 」を追記しました。 「*認可設定のキャッシュ* 」を追記しました。
2018-04-01	第5版 「*付録*」に「*IM-LogicDesigner のフロートリガで IM-共通マスタ を使用する*」を追加
2018-08-01	第6版 「*キャッシュ*」に「*プロファイル画像キャッシュ*」を追加
2019-04-01	第7版 「*会社グループ*」の会社グループ内包の記述を修正しました。

IM-共通マスタについて

IM-共通マスタ

目的

会社・組織、取引先、ユーザ、その他のマスタを統合化することで、イントラマートユーザのマスタメンテナンスにかかる負担を軽減するとともに、イントラマートシステム全体を統合的に構築できる環境を整備します。

統合化の範囲と要件

IM-共通マスタにはさまざまな要件を実現するために設計・実装されています。
その内容を【表：統合化の範囲と要件】に示します。

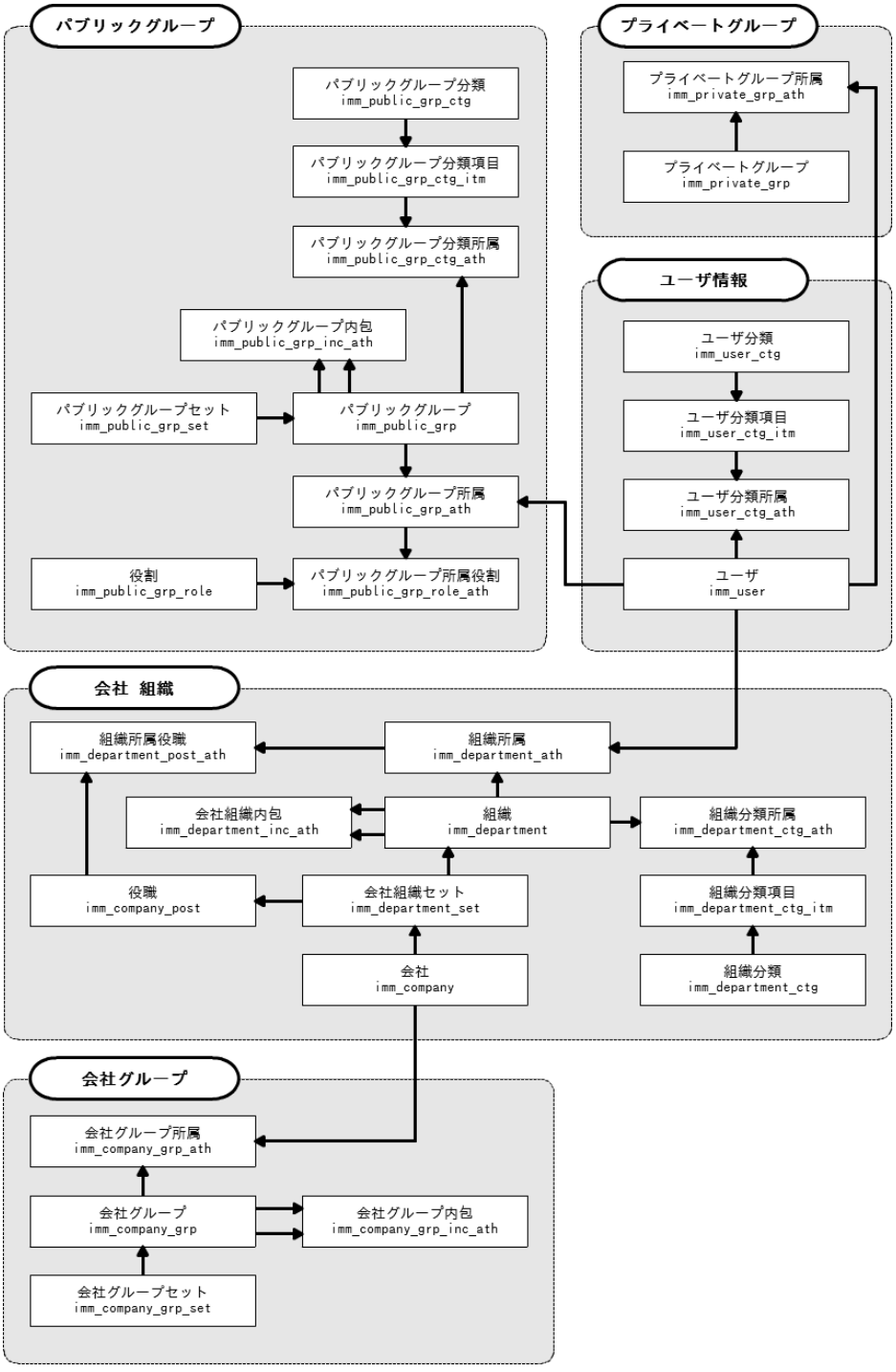
No.	要件	方法
1	組織やパブリックグループを指定階層以下の集合として扱うことができること。（例：.ある組織以下の社員すべてを検索したい）	内包型のデータ構造で実現する。
2	画面の性格によっては自社のみのデータを表示したり、顧客一覧を抽出したりしたい。	「分類」、「分類詳細」、「分類」の各テーブルを使用して組織を分類する、または、法人、取引先テーブルを使用して自社と取引先を明確に分けることで実現する。
3	多数の取引先が混在する環境で、互いの可視範囲を設定したい。	「分類」、「分類詳細」、「分類」の各テーブルを使用して実現する。
4	多数の取引先が混在する環境で、販売チャネル毎に顧客一覧抽出したい。	「分類」、「分類詳細」、「分類」の各テーブルを使用して実現する。
5	会社やパブリックグループの階層構成のバージョン管理を行いたい。	組織セットの期間化、パブリックグループセットの期間化を行うことで実現する。
6	ユーザ名や組織名などが変更された場合でも過去の情報を残したい、または変更される予定の情報をあらかじめ登録しておきたい。	対象となるテーブルに対する期間化されたテーブルを使用して実現する。
7	組織名などの表示名を言語によって変更したい。	対象となるテーブルの各言語に国際化した情報を使用して実現する。
8	「品目」の情報（名称など）を期間および言語により変更したい。	「品目」に対して期間化、国際化されたテーブルを使用して実現する。
9	複数の異なる視点から商品を検索したい。	ツリー構造のカテゴリを作成することにより実現する。
10	会社やユーザを任意に分類し、その分類を使用してユーザや会社の情報を取り扱いたい	ユーザ分類や組織分類テーブルを使用して分類を任意に定義し、ユーザや組織を分類することで実現する。
11	業務によって組織構成の考え方を変えたい。業務領域に応じて組織の構成を柔軟に取り扱いたい	同一会社内に複数の組織セットを定義することで実現する
12	アプリケーションで役職の上下関係を判断したい	役職にRankを設定し、上下関係を表すことで実現する
13	パブリックグループのメンバーを係などパブリックグループ内の属性を設けて管理したい	役割テーブルを使用してパブリックグループ内の役割を定義することで実現する
14	会社情報をさらに大きなグループで管理し、情報の統制や権限の管理などに活用したい	会社グループ関連のテーブルを使用し、会社のグループを定義できる。会社情報を会社グループと関連付けて管理することで実現する
15	取引先の企業や組織を法人や法人のグループとして取り扱いたい	法人関連のテーブルを使用して法人を定義し、 取引先を法人に関連付けることで実現する

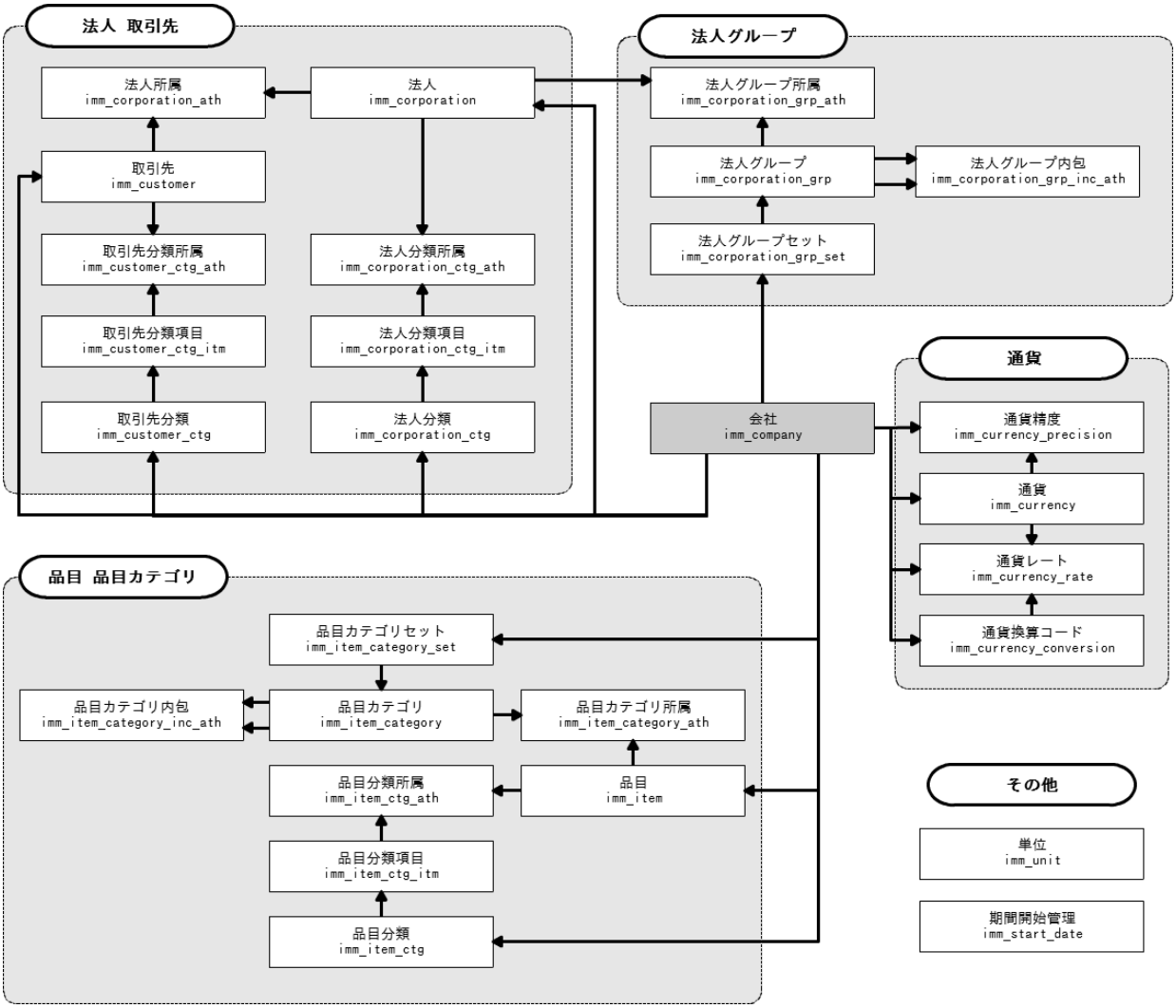
No.	要件	方法
16	会社や組織に所属する各種情報を、他の会社から閲覧できないように制限したい。	会社ごとに認可設定を行うことでアクセスの制限を実現する

【表：統合化の範囲と要件】

データ構造

IM-共通マスタはさまざまなテーブルから構成されています。
これらのテーブルの関連概要を【図：データ構成（全体）】に示します。





【図：データ構成（全体）】

エンティティの概観

IM-共通マスタの基本的な設計要素として期間化と国際化への対応があります。
期間化を行うことで「昔の帳票を出力したい」、「新しい組織情報をあらかじめ登録しておきたい」といったような要件に対応するためには情報の期間管理を行う必要があります。
また国際化情報を持つことで「ログインしているユーザが使用している言語で組織名を表示したい」といった要件に対応することができます。

従来のアプリケーション共通マスタではそれぞれ要件の対応のためにテーブルを複数定義して使用していましたが、IM-共通マスタでは１つのテーブル上で同様の情報を保持できるようにしました。

IM-共通マスタのエンティティの構造は概念によって大きく5種に大別できます。

項目
- 基本構造
- 内包構造
- 所属構造
- 期間属性
- 分類構造

基本構造

基本構造はユーザや組織といった基本的にそれだけで意味を成すマスタ情報を表します。
基本構造は期間化・国際化の対応内容によって以下の4種類の構造に分類されます。

- *期間国際化構造*

- *国際化構造*
- *期間化構造*
- *非期間国際化構造*

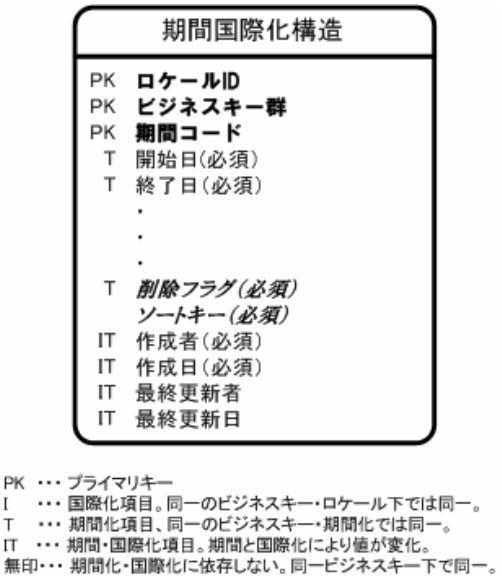
それぞれの特徴について順に説明した後、*期間の取り扱い方について* についても説明します。

期間国際化構造

期間化と国際化に対応する構造です。
具体的には下記のようなルールに沿って設計されています。

- 言語を表すロケールIDを持つ（PK）
- 単一の論理データを表すビジネスキー群を持つ（PK）
- 期間を表す期間コードを持つ（PK）。
  期間コードは特定の開始日から終了日の期間を表す（開始終了日付も同テーブルに保持）。
- 削除フラグ、ソートキーを持っている。必須の項目だが、期間化・国際化されない（同一ビジネスキーの全てのレコードが同じ値）。
- 作成者・作成日付を持つ
- 最終更新者・最終更新日を持つ

> **コラム**
> 一つの論理エンティティを表す為に必要なレコード数 は 国際化数 × 期間化数 です。



【図：期間国際化構造】

国際化構造

期間管理される必要が無く国際化情報を含むエンティティは以下のように設計されています。

- 言語を表すロケールIDを持つ（PK）。
- 単一の論理データを表すビジネスキー群を持つ（PK）。
- 削除フラグ、ソートキーを持っている。必須の項目だが、国際化されない（同一ビジネスキーの全てのレコードが同じ値）。
- 最終更新者・最終更新日を持つ。

> **コラム**
> 一つの論理エンティティを表す為には国際化した数の分だけレコードが必要になります。

PK ・・・ プライマリキー
I ・・・ 国際化項目。同一のビジネスキー・ロケール下では同一。
T ・・・ 期間化項目、同一のビジネスキー・期間化では同一。
IT ・・・ 期間・国際化項目。期間と国際化により値が変化。
無印・・・ 期間化・国際化に依存しない。同一ビジネスキー下で同一。

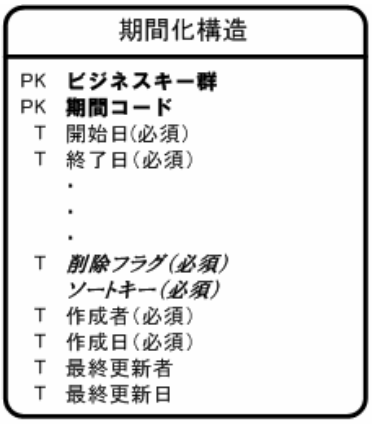
【図：国際化構造】

期間化構造

国際化構造とは対照に国際化されないエンティティを表すテーブルは以下のように設計されています。

- 単一の論理データを表すビジネスキー群を持つ（PK）。
- 期間を表す期間コードを持つ（PK）。
  期間コードは特定の開始日から終了日の期間を表す（開始終了日付も同テーブルに保持）。
- 削除フラグ、ソートキーを持っている。
  必須の項目だが、期間化されない（同一ビジネスキーの全てのレコードが同じ値）。
- 作成者・作成日付を持つ。
- 最終更新者・最終更新日を持つ。

> コラム
> 一つの論理エンティティを表す為には期間化した数の分だけレコードが必要になります。



PK ・・・ プライマリキー
I ・・・ 国際化項目。同一のビジネスキー・ロケール下では同一。
T ・・・ 期間化項目、同一のビジネスキー・期間化では同一。
IT ・・・ 期間・国際化項目。期間と国際化により値が変化。
無印・・・ 期間化・国際化に依存しない。同一ビジネスキー下で同一。

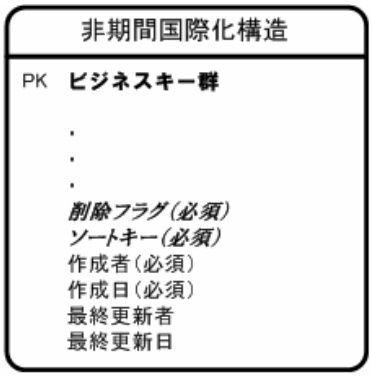
【図：期間化構造】

非期間国際化構造

期間化にも国際化にも対応しないエンティティを表すテーブルは以下のように設計されています。

- 単一の論理データを表すビジネスキー群を持つ（PK）
- 削除フラグ、ソートキーを持っている。必須。
- 最終更新者・最終更新日を持つ

コラム
レコードで一つの論理エンティティを表すことが出来ます。



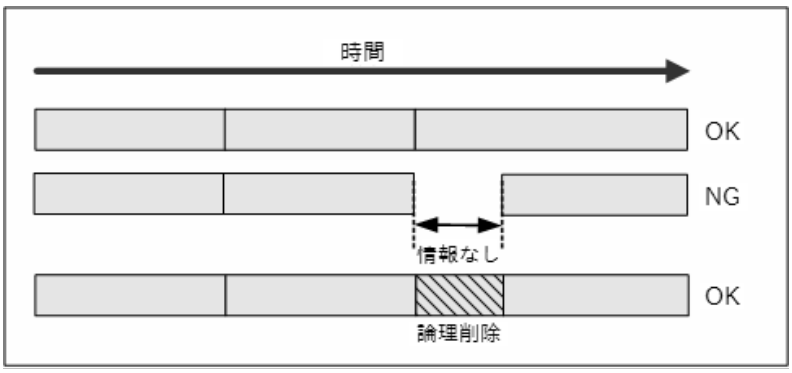
PK ・・・ プライマリキー
I ・・・ 国際化項目。同一のビジネスキー・ロケール下では同一。
T ・・・ 期間化項目。同一のビジネスキー・期間化では同一。
IT ・・・ 期間・国際化項目。期間と国際化により値が変化。
無印・・・ 期間化・国際化に依存しない。同一ビジネスキー下で同一。

【図：非期間国際化構造】

期間の取り扱い方について

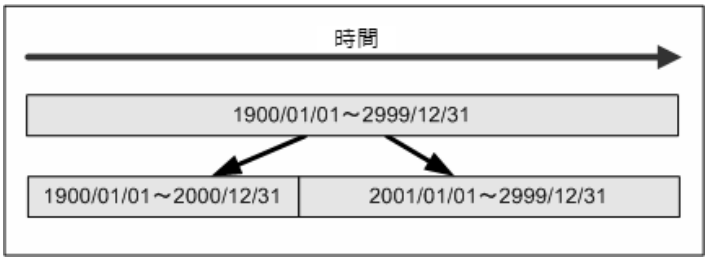
期間の取り扱い方については下記の制約がありますので注意してください。

- 基本構造の期間は、システムの開始日から終了日まで全ての期間が定義されている必要があります。
  システムの開始日と終了日はシステムで管理されており、専用のAPIを使用して取得することが出来ます。
  期間化されている基本構造では、このシステムの開始日から終了日までの全期間のデータが必ず必要です。また重複や間断のない連続した期間である必要があります。
- 隣接した期間は物理的には前方の期間の終了日と後方の期間の開始日は同じ値で保持しています。
- 特定の期間で情報を無効にしたい場合は、削除フラグが定義されていますので、該当の期間を論理削除状態とすることで取り扱います。
  （【図：期間情報の例】参照）



【図：期間情報の例】

- 特に期間指定無く新たな論理エンティティを作成する場合はシステム開始日～システム終了日の期間で作成します。
- 期間化に対応した情報は常に全期間分存在しているため、新しい期間を作成する場合は既存の期間を任意の日付で分割して新しい期間として使用します。
  分割して生成された直後のデータは期間（コード、開始日、終了日）を除いて分割元と全く同じ情報になります。



【図：期間の分割】

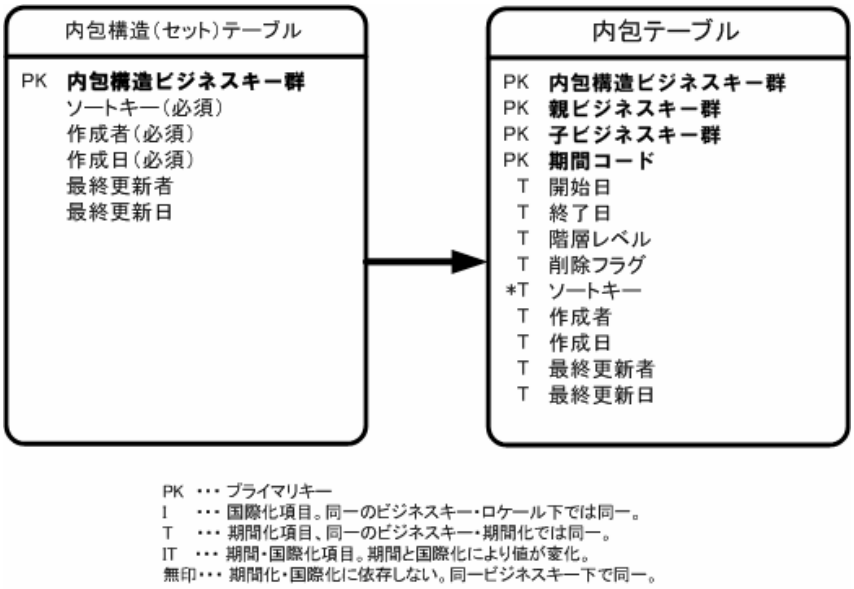
- ある期間の開始日・終了日を変更する場合、その期間の前後のデータについても間断や多重が発生しないように適切に隣接す

る期間を伸縮しなければなりません。

- 以下の情報については、期間化された情報のみを保持するため、全期間分のデータを持ちません。
  - 内包情報
  - 所属役職、所属役割
  - 分類所属

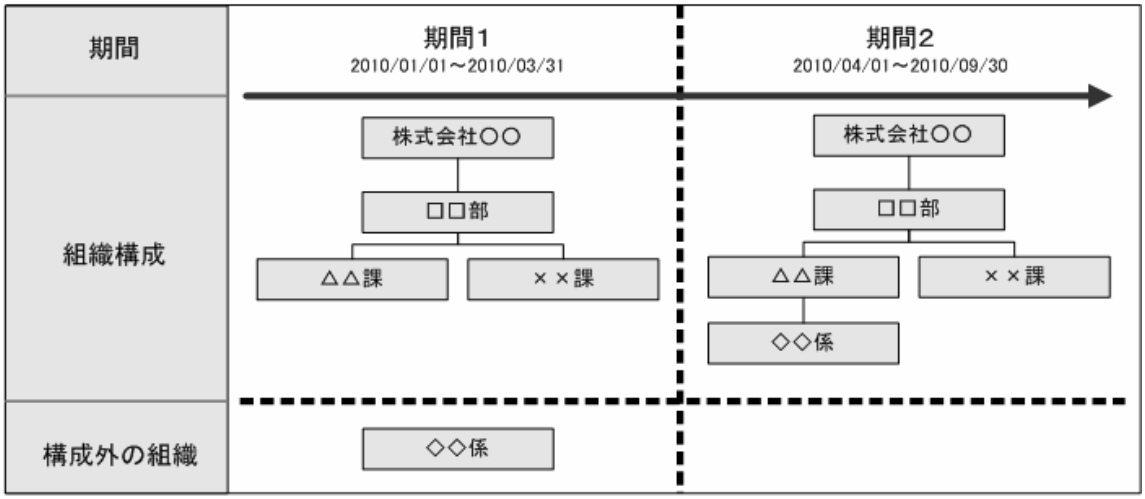
> **コラム**
>
> IM-共通マスタで用意しているAPIを使用すると、上記の条件に従ったチェックや以下のようなデータの操作が自動的に実行されます。
> IM-共通マスタ情報を追加・更新・削除する場合などは特にAPIを使用して行ってください。
>
> - ソートキーや削除フラグなど、期間化・国際化に依存しない項目についての整合性の維持
> - 国際化で新しい言語が追加・削除された場合の整合性の維持
> - 論理エンティティの期間を変更した場合に隣接する期間を自動調整する。
>   また隣接する期間を超えて期間を延ばした場合には隣接期間を削除する。
> - システム開始日や終了日を含む期間を縮めた場合に発生する期間の空きを自動補完する

内包構造



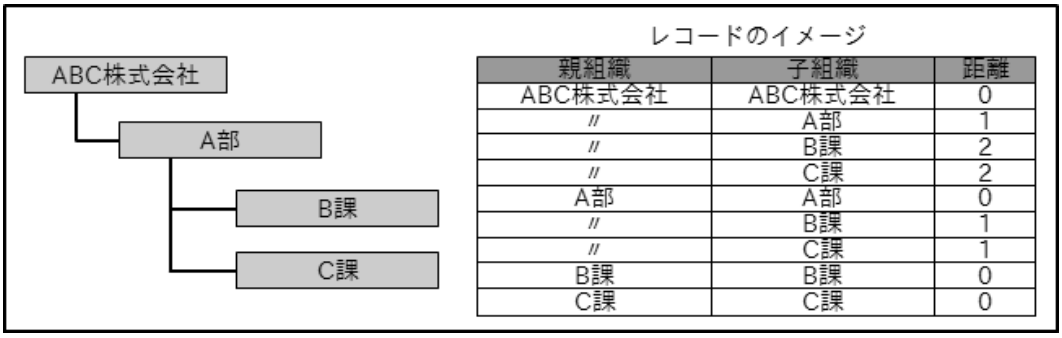
【図：内包構造】

組織構成のように再帰的に階層化された情報を構成する場合に、その関係を記録する情報群です。
期間化する場合は期間ごとの構成情報を管理します。（【図：内包構造の期間化】参照）
内包情報の期間化は階層のトップノード（原則としてセットと呼ばれ、これを表すテーブルが存在します）単位に管理され、期間毎に配下のツリー構成情報を管理します。



【図：内包構造の期間化】

階層構造を保持するために内包構造エンティティでは階層を成す要素間のすべての関係を、階層の距離とともに保持しています。
（【図：内包関係レコードのイメージ】参照）



レコードのイメージ

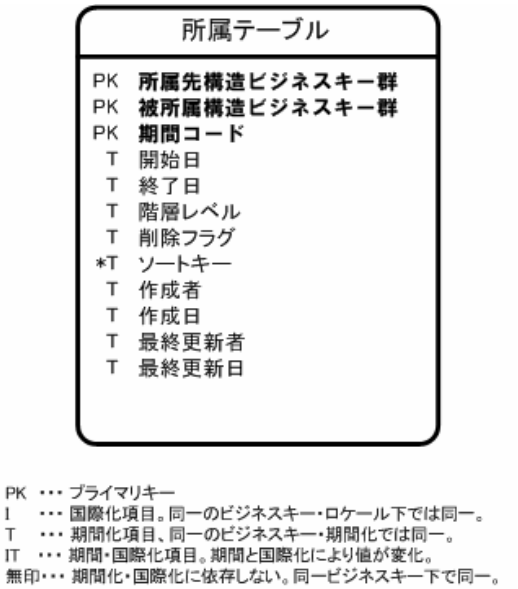
親組織	子組織	距離
ABC株式会社	ABC株式会社	0
〃	A部	1
〃	B課	2
〃	C課	2
A部	A部	0
〃	B課	1
〃	C課	1
B課	B課	0
C課	C課	0

【図：内包関係レコードのイメージ】

内包は階層構造のみ管理しており、実際階層化されている実体の情報とは原則的には独立しています。
たとえば会社組織の情報であれば、組織の情報は組織の情報として内包からは独立して存在しており、期間も組織は組織で独立して管理します。
一方で組織構成の情報は内包によって組織とは独立して管理します。
そのほかは基本構造と同様です。

> **コラム**
>
> IM-共通マスタでは以下のテーブルが該当します。
>
> - 組織内包
> - パブリックグループ内包
> - 品目カテゴリ内包
> - 法人グループ内包
> - 会社グループ内包

所属構造



【図：所属構造】

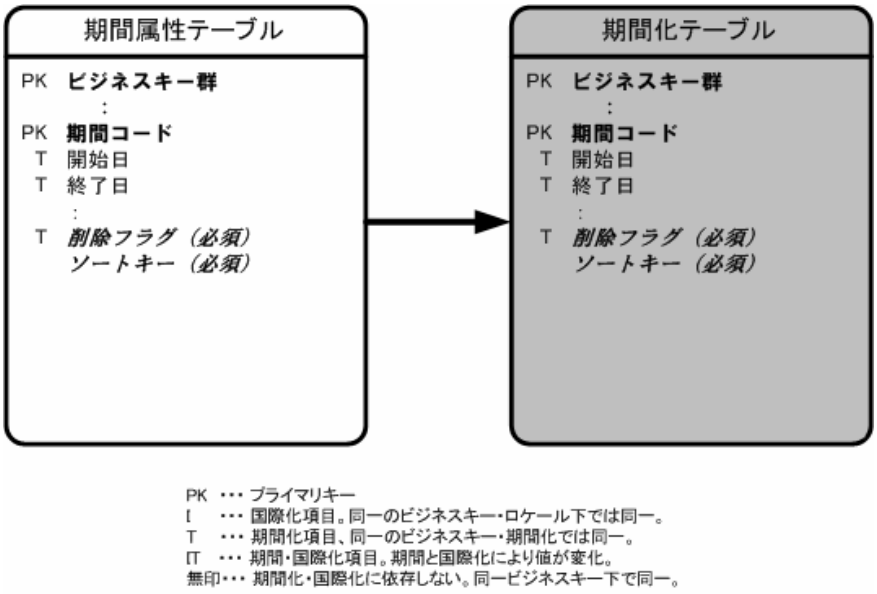
ある基本構造のエンティティから別の基本構造のエンティティへの所属関係を表します。
内包と異なり再帰構造ではない n 対 m の関係になります。
それ以外は基本構造と同様です。

コラム

IM-共通マスタでは以下のテーブルが該当します。

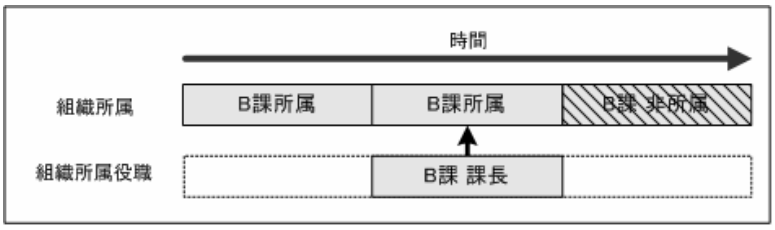
- 組織所属
- パブリックグループ所属
- 品目カテゴリ所属
- 法人所属
- 会社グループ所属
- 法人グループ

期間属性



【図：期間属性（図中左）】

特定の期間に対して付与される属性情報です。
このエンティティ自体は期間情報を保持せず、他のエンティティの期間情報を参照します。
たとえば組織所属情報は期間化されており、ユーザが組織に所属している間を期間化して保持しています。
ユーザの役職は所属情報で定義された期間に対して、組織所属役職情報を定義することでその期間役職に就いていることを表します。



【図：組織所属の例】

コラム

IM-共通マスタでは以下のテーブルが該当します。

- 役職所属
- 役割所属
- 各種分類所属

分類構造

特定のエンティティを任意に分類するための情報構造です。

特に分類自身は期間管理されません。
分類される側のエンティティの期間に依存します。

> **コラム**
>
> IM-共通マスタでは以下のテーブルが該当します。
>
> - ユーザ分類、ユーザ分類項目、ユーザ分類所属
> - 組織分類、組織分類項目、組織分類所属
> - パブリックグループ分類、パブリックグループ分類項目、パブリックグループ分類所属
> - 品目分類、品目分類項目、品目分類所属
> - 法人分類、法人分類項目、法人分類所属
> - 取引先分類、取引先分類項目、取引先分類所属

複数会社対応

複数会社対応の概要

複数会社対応とは、以下の状態を管理するための機能です。

- アクターごとに、参照可能な会社を管理する。
- アクターごとに、編集可能な会社を管理する。

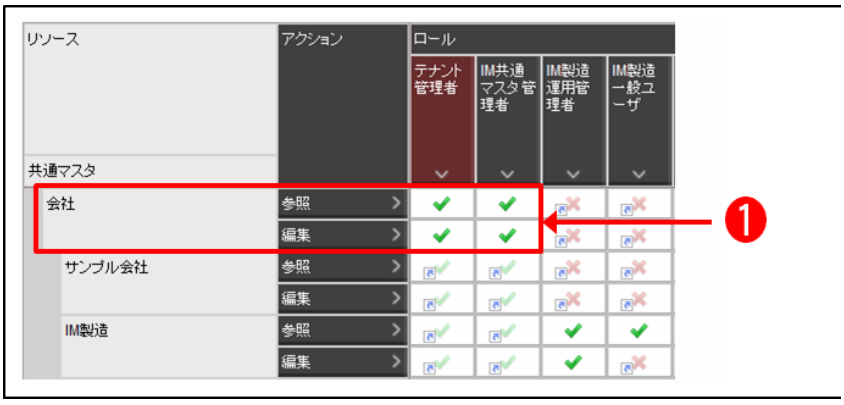
> **コラム**
> いずれの場合も会社で情報が絞り込まれます（組織情報で絞り込まれることはありません）。

> **注意**
> ユーザが所属する会社のみ参照できる、ということではありませんので注意してください。

アクター

アクターは全部で3種類です。

- IM-共通マスタ管理者
  - 全ての会社の編集を行えるリソースを持つユーザです。
  - 会社の作成、変更、削除をおこなうことができます。
  - 会社のリソースに対して認可で管理権、参照権を設定できます。



【図：IM-共通マスタ管理者の認可設定イメージ】

1. テナント管理者、IM共通マスタ管理者は、会社全体に対して、参照権と編集権が与えられている。

   →すべての会社に対して編集可能。

   →すべての会社に対して、そこに属する情報を参照することが可能。

- 会社管理者

- IM-共通マスタ管理者より、認可の会社リソースに対して管理権を与えられたユーザです。
- 各会社の組織構成、役職、所属ユーザを登録、変更、削除できます。
- 会社情報の変更、削除はできません。
- 会社管理者のロールの追加、および認可設定が必要です。

【図：会社管理者の認可設定イメージ】

1. 会社管理者を作成する場合、会社ごとにロールの作成が必要。ユーザに対して、ロールの付与も行う。
2. IM製造運用管理者（会社管理者）に、IM製造に対して、参照権と編集権を与える。

   →IM製造のみ編集可能。

   →IM製造に属する情報のみ参照可能。

- 一般ユーザ
  - 認可の会社リソースに対して参照権を与えられたユーザです。
  - 参照権をもつ会社の組織構成、役職、所属ユーザを参照できます。

【図：一般ユーザの認可設定イメージ】

1. 一般ユーザ用のロールは、作成してもしなくてもよい。既存のロールに対して権限を与えてもよい。
2. IM製造一般ユーザに、IM製造に対して、参照権を与える。

   →IM製造を含む会社の編集は不可。

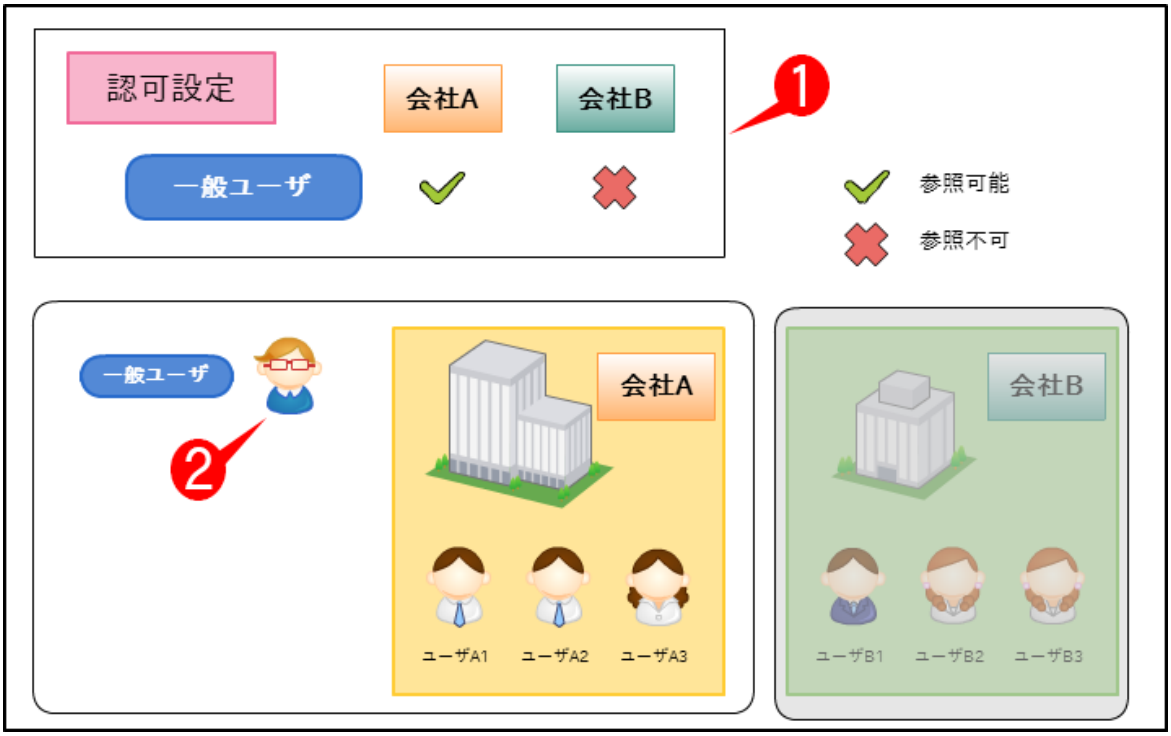
   →IM製造に属する情報のみ参照可能。

認可制御（アクセス権）

複数会社対応における認可制御は、会社ごとにアクセス権を設定し、その設定情報をもとに制御する仕組みです。
具体的な例としては次のとおりです

- 共通検索画面において会社の絞り込みを行う際に、全ての会社を検索対象とするか、参照可能な会社のみを検索対象とするかを、ログインユーザが持つロールをもとに制御する。

IM-共通マスタにおける認可の制御イメージを【図：認可の制御イメージ】に示します。



【図：認可の制御イメージ】

1. ロール「一般ユーザ」に対して、会社Aのみ参照可能となる認可設定を行う
2. 一般ユーザのロールを持つユーザは、会社Bを検索対象にできない

認可設定のキャッシュ

intra-mart Accel Platform 2015 Winter(Lydia) 以降から、会社毎のアクセス権への参照を高速化するため、IM-共通マスタの認可設定がデフォルトでキャッシュされるよう設定されています。キャッシュされる認可設定は以下のとおりです。

- 会社一覧のリソースグループ設定
- 会社一覧のポリシー設定

設定内容については、「キャッシュ」を参照してください。

各画面における制御

IM-共通マスタの画面には大きく2種類あり、検索画面とメンテナンス画面で制御が異なります。
検索画面ごとの仕様やパラメータについては、「IM-共通マスタ 検索画面仕様書」を参照してください。

検索画面

- 会社の参照権を判断します。
- ユーザ検索や組織検索などは認可による認証をおこない、参照できる会社のみ対象とします。
- 全会社参照権をもつユーザ（アクターがIM-共通マスタ管理者）
  - 認可による認証をおこない、パラメータに指定された全会社を対象とします。
  - パラメータに会社コードを指定しなかった場合は全会社を対象とします。
- 全会社参照権をもたないユーザ （アクターが会社管理者、または一般ユーザ）
  - 認可による認証をおこない、パラメータに指定された会社のうち、参照できる会社のみ対象とします。
  - パラメータに会社コードを指定しなかった場合は参照できる会社のみを対象とします。

メンテナンス画面

- 会社の編集権を判断します。
- ユーザ検索は認可による認証をおこないません。
- 全会社の編集権をもつユーザ （アクターがIM-共通マスタ管理者）
  - 認可による認証をおこない、全ての会社を編集対象とします。

- 全会社編集権をもたないユーザ （アクターが会社管理者）
  - 認可による認証をおこない、会社管理者が編集可能な会社のみを対象とします。
  - それ以外の会社は、権限がないため編集も参照もできません。

インポート／エクスポート

インポート、エクスポートは認可による制御は行われません。
インポート、エクスポートでの認可制御は手動で行います。

オートコンプリート

IM-共通マスタにおけるオートコンプリートは、ユーザコード、ユーザ名、ユーザ名カナの一部を入力することで、該当するユーザの一覧をテキストボックスの下に表示する機能です。
オートコンプリートの詳細は「APIリスト(Javadoc)」を参照してください。

検索仕様

IM-共通マスタにおけるオートコンプリートの検索仕様は以下のとおりです。

- ユーザを対象とします。
- ユーザコード、ユーザ名、ユーザ名カナを前方一致で検索します。
- ログインユーザのタイムゾーンに即したシステム日時で有効なユーザを検索します。
- ログインユーザが持つロケール情報でユーザを検索します。
- 暗黙条件として会社（複数可）を指定することができます。
  その場合、指定した会社に所属するユーザを対象とします。
  指定した会社が存在しない場合はリストに表示されません。

認可制御

オートコンプリートには認可制御があります。
認可制御の有無と暗黙条件の指定有無により、オートコンプリートの動作が変わります。

- 認可制御なし
  - 暗黙条件指定なし
    - 認可による認証を行わず、すべてのユーザをオートコンプリートの候補とします。
    - 会社に所属していないユーザも候補とします。
  - 暗黙条件指定あり
    - 認可による認証を行わず、暗黙条件に指定された会社に所属するユーザをオートコンプリートの候補とします。
    - 会社に所属していないユーザは対象外です。
    - 指定したすべての会社が無効、または未存在の場合は、候補が出力されません。
- 認可制御あり
  - 暗黙条件指定なし
    - 認可による認証を行い、ログインユーザが参照可能な会社に所属するユーザをオートコンプリートの候補とします。
    - 会社に所属していないユーザは対象外です。
  - 暗黙条件指定あり
    - 暗黙条件に指定された会社のうち、ログインユーザが参照可能な会社のみを対象とします。
    - 上記で対象となった会社に所属するユーザをオートコンプリートの候補とします。
    - 会社に所属していないユーザは対象外です。
    - 指定したすべての会社が無効、または未存在の場合は、候補が出力されません。

実装方法と暗黙条件の指定

実装サンプルは以下のリストになります。

```
<script>
function getCompanys() {
var criteria = {
   "company" : [
         {"company_cd" : "comp_sample_01"}
   ]
};
return criteria;
}
</script>

<imart type="imuiAutocomplete"
      process="config"
      target="jp.co.intra_mart.master.autocomplete.user.AuthzUserAutocompleteProcessor"
      onAjaxParameterExtend="getCompanys"
      limit="7"
      delay="0"
      minLength="0" />
```

【リスト：オートコンプリート】

タグの各要素に指定する内容は以下のとおりです。

- process
  config固定です
- target
  認可考慮の有無で指定するクラスを変更します。
  - 認可考慮あり ： jp.co.intra_mart.master.autocomplete.user.AuthzUserAutocompleteProcessor
  - 認可考慮なし ： jp.co.intra_mart.master.autocomplete.user.UserAutocompleteProcessor
- onAjaxParameterExtend
  この要素の指定は任意です。要素の使用有無によって記述が変わります。
  - 使用しない場合
    - 要素の記述は不要です。
    - 要素指定時に必要なJSメソッドの定義も不要です。
  - 使用する場合
    - 暗黙条件を取得するためのJSメソッドを指定します。メソッド名は任意です。
    - 要素に指定したメソッド名でJSメソッドを定義します。
    - 返却形式はObjectです。中身の詳細は以下の表のとおりです。
    - 返却形式が誤っている場合はオートコンプリートの候補が出力されません。
    - 返却値のcompanyの配列が長さ0の場合、暗黙条件指定が無いものとして検索します。

プロパティ名	型	必須	初期値	備考
company	Array	×	－	－
└配列インデックス	Object	×	－	－
└company_cd	String	○	－	－

```
{
  "company" : [
    { "company_cd" : 会社コード } , { "company_cd" :  会社コード } ...
  ]
}
```

構造

IM-共通マスタの構造の詳細を説明します。
各テーブルの物理的な定義は、「 IM-共通マスタ テーブルER図 」を参照してください。

ユーザ

ユーザはintra-martで扱う個人情報です。期間化と国際化が可能であり、会社組織やパブリックグループ、プライベートグループ、ロールに所属させる事が出来ます。

プロファイルとアカウント

intra-martではユーザを表現する情報としてプロファイルとアカウントの２種類があります。
プロファイルは組織への所属や人事情報など業務で扱われるユーザの情報です。
アカウントは主にログインやセキュリティなど、システムを対象としたユーザです。
IM-共通マスタで取り扱うユーザはプロファイルに該当します。

> **コラム**
> プロファイルとアカウントはそれらのユーザコードが一致する場合、同一のユーザであるとみなされます。

ユーザ分類

ユーザ分類エンティティを使用してユーザを任意の観点で分類し、管理することが出来ます。
ユーザ分類は以下の3つのテーブルで構成されています。

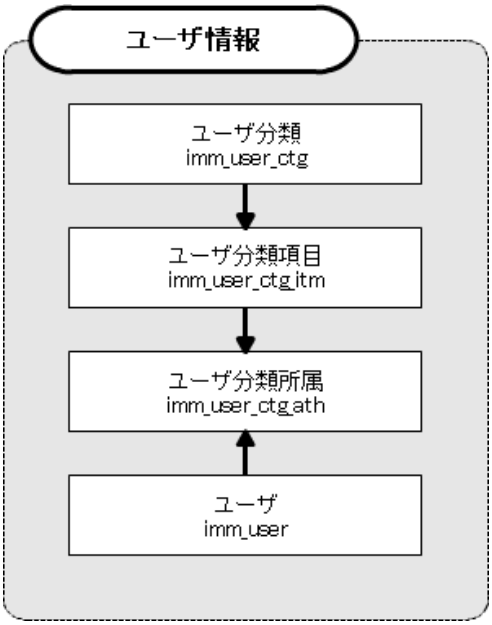
1. ユーザ分類
2. ユーザ分類項目
3. ユーザ分類所属

ユーザ分類は分類する観点を表し、ユーザ分類項目はその観点で分類できる項目を定義します。
ユーザ分類所属は分類項目とユーザを結びつける情報で、これが実際にユーザがどの分類に分類されるかを表しています。

またIM-共通マスタで提供している共通の検索画面では、これらの分類を使用して暗黙的・明示的に絞り込んで検索することが出来るようになっています。

データ構造

ユーザ、ユーザ分類に関連するER図を【図：ユーザ及びユーザ分類関連のER図】に示します。



【図：ユーザ及びユーザ分類関連のER図】

【図：ユーザ及びユーザ分類関連のER図】において各エンティティは次のような役割・制約があります。

- ユーザ分類
  - ユーザ分類の種類（例えば契約形態、勤務形態等）を管理します。
  - 無効化すると、配下の分類項目も連動して無効化します。
    また分類項目が無効化されると、該当のユーザ分類所属エンティティが物理削除されます。
- ユーザ分類項目
  - ユーザ分類の種類ごとに定義されている詳細（正社員、契約社員等）を管理します。
  - 属するユーザ分類エンティティが無効化されると、連動してこのエンティティも無効化状態になります。
- ユーザ分類所属
  - ユーザが分類項目に分類されていることを表します。ユーザの期間毎に分類項目を設定します。
  - 期間は分類するユーザの期間と一致します。ユーザ分類所属の期間コードは対象のユーザの期間コードと同一です。
  - 関連付けられているユーザの期間と無効化状態が連動します。
    関連づいているユーザの期間が無効化された場合、該当のユーザ分類所属も無効化されます。
    ユーザの期間が有効化された場合も連動して有効化します。
  - ユーザ分類項目が無効化される際、このエンティティの関連するレコードは物理削除されます。
  - ユーザが物理削除される際、このエンティティの関連するレコードも物理削除されます。
- ユーザ
  - ユーザの個人情報を保持する期間国際化構造のエンティティです。
  - ユーザの性別は期間化も国際化もされません。

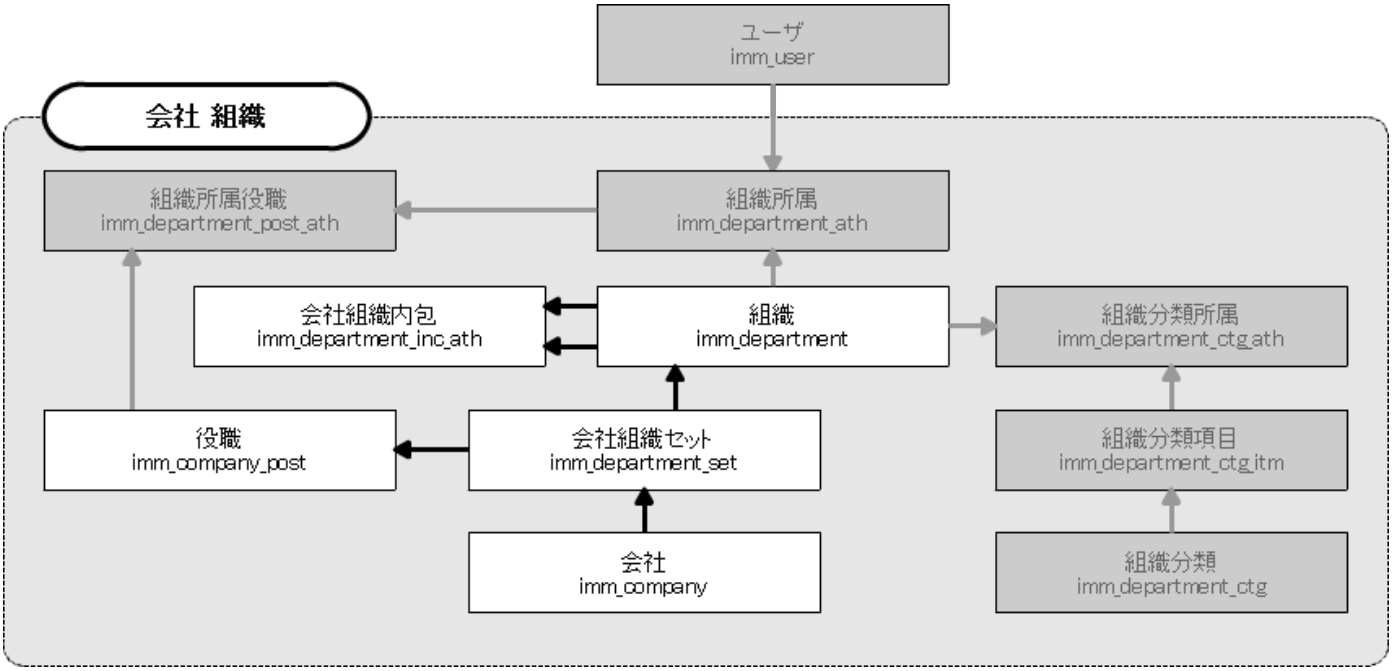
会社と組織

IM-共通マスタでは会社やその組織についての情報を扱うことができます。
取り扱える情報には以下のようなものがあります。

- 会社や組織そのものの情報
- 組織構成の情報
- 業務観点毎に異なる組織構成情報の使用
- 組織構成の履歴管理
- 会社ごとで扱う役職
- 会社または組織に所属するユーザ
- 任意のユーザのある時点における主所属
- 会社ごとで扱う組織構成の分類
- 会社ごとにアクセス可能な情報の管理

会社と組織の構成

会社および組織の構成に関連するER図を【図：会社・組織の構成に関連するER図】に示します。



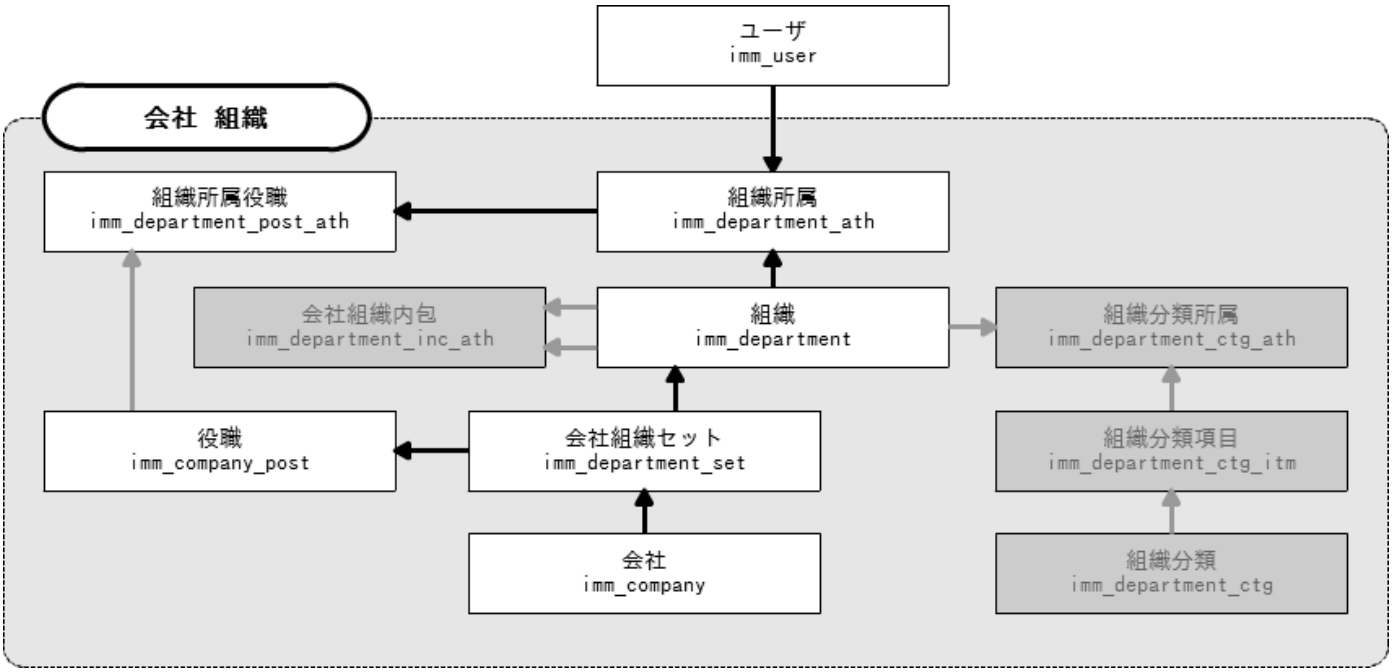
【図：会社・組織の構成に関連するER図】

【図：会社・組織の構成に関連するER図】において各エンティティは次のような役割・制約があります。

- 会社
  - 会社の概念を管理します。会社自身の詳細情報は組織エンティティで管理します。
  - 内包構造のエンティティの一部です。
- 会社組織セット
  - 業務観点毎に異なる組織構成が必要な場合に、単一の会社において複数の組織セットの定義を可能にします。
    会社を作成した際に自動的にデフォルトの組織セットが作成されます。
    特に複数の組織構成を必要としなければ、単純にこのデフォルトのセットが使用されます。
  - 業務観点によって通常の組織構成とは異なる構成を定義したい場合、組織セットを増やして同一の会社に複数の組織構成を定義することが出来ます。
    異なる組織セット配下に作成される組織はそれぞれ異なる組織になります。
    期間や保持している情報もすべてそれぞれ独自に保持しています。
  - 会社が物理削除されると関連する組織セットも削除されます。
  - 内包構造のエンティティの一部です。
- 組織
  - 会社や会社内に存在する組織の詳細情報を管理します。
  - 組織の属する組織セットや会社が削除されると組織もあわせて削除されます。
- 会社組織内包
  - 会社内の組織構成の詳細情報と、組織構成の履歴を管理します。組織セットを先頭としたツリー構成の情報を格納しています。
- 役職
  - 会社内で定義されている組織セット毎に役職の情報を管理するエンティティです。
  - 期間国際化の基本構造です。
  - 定義されている会社組織セットや会社を物理削除すると、役職もあわせて物理削除されます。
  - ランクを設定することが出来、その役職の地位の高さを表します。
    - 値が小さいほど地位が高いものとして扱います。
    - ランクは期間化も国際化もされません。

会社と組織への所属

ユーザが所属する会社および組織に関連するER図を【図：会社・組織とユーザの所属に関連するER図】に示します。



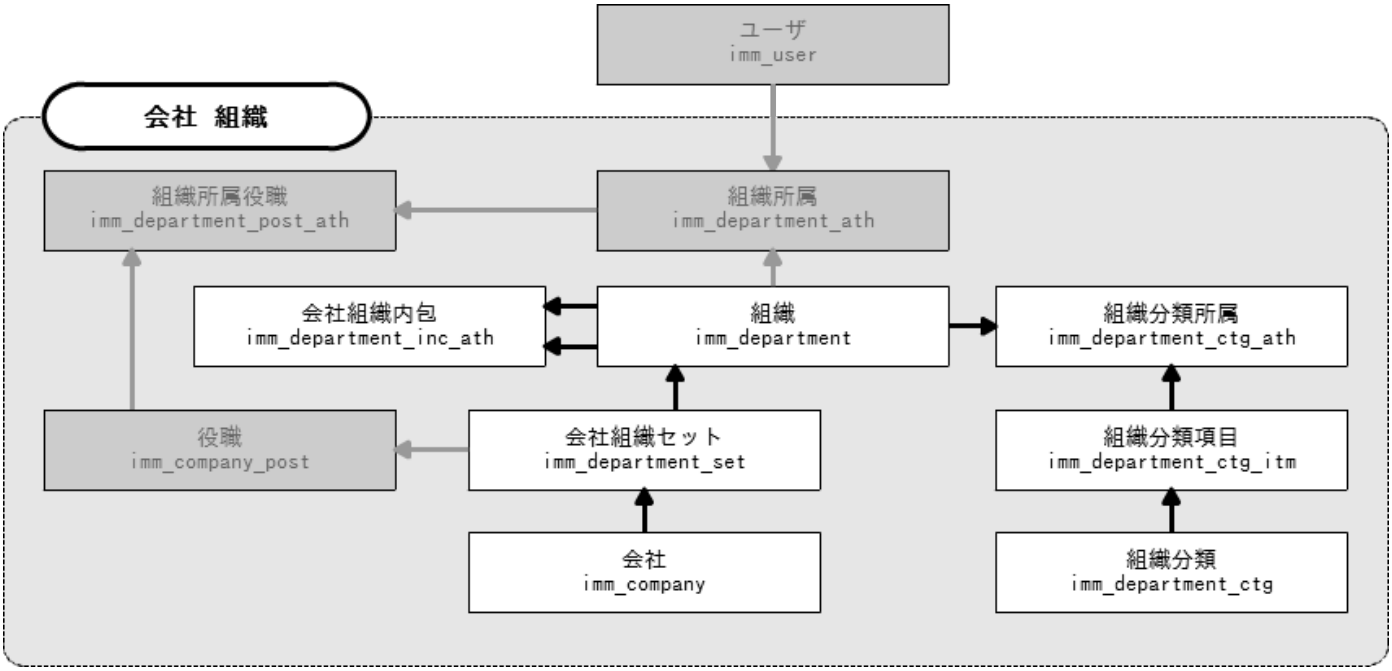
【図：会社・組織とユーザの所属に関連するER図】

【図：会社・組織とユーザの所属に関連するER図】において各エンティティは次のような役割・制約があります。

- ユーザ
  - ユーザを管理します。詳細は「*ユーザ*」を参照してください。
- 会社
  - 会社を管理します。詳細は「*会社と組織の構成*」を参照してください。
- 組織
  - 組織の詳細情報を管理します。詳細は「*会社と組織の構成*」を参照してください。
- 役職
  - 役職を管理します。詳細は「*会社と組織の構成*」を参照してください。
- 組織所属
  - ユーザがどの期間、どの組織に所属するかを管理します。
    主所属がどの組織であるかも管理します。
  - 所属構造のエンティティです。
  - 主所属は何れの会社であるかに関わらずデフォルトのセット中では重複する期間で設定することはできません。
    同一会社内の異なる組織セットの間等では重複して設定することができます。
  - ユーザが物理削除された際に、あわせて組織所属役職の関連レコードも削除されます。
  - 役職が物理削除された際に、あわせて組織所属役職の関連レコードも削除されます。
- 組織所属役職
  - ユーザが組織に所属する期間について、どの役職に就いているかを管理します。複数の役職を兼任することが出来ます。
  - 期間属性構造です。組織所属を参照します。
  - 期間コード、開始日、終了日、削除フラグが組織所属と連動します。組織所属が無効化されると、関連する組織所属役職も無効化されます。有効化した場合も同様です。組織所属の期間が変更されると開始日、終了日が連動して更新されます。
  - ユーザが物理削除された際に、あわせて組織所属役職の関連レコードも削除されます。
  - 役職が物理削除された際に、あわせて組織所属役職の関連レコードも削除されます。

会社と組織の分類

会社および組織の分類に関連するER図を【図：会社・組織の分類に関連するER図】に示します。



【図：会社・組織の分類に関連するER図】において各エンティティは次のような役割・制約があります。

- 組織分類
  - 組織分類の種類を会社ごとに管理します。
  - 無効化すると、配下の分類項目も連動して無効化します。
    また分類項目が無効化されると、該当のユーザ分類所属エンティティが物理削除されます。
- 組織分類項目
  - 分類の種類ごとに定義されている詳細を会社ごとに管理します。
  - 組織分類エンティティが無効化された場合、連動してこのエンティティの関連するレコードも無効化状態になります。
- 組織分類所属
  - 組織が分類項目に分類されていることを表します。
  - 組織が期間化されている場合はその期間単位に分類します。
  - 期間は分類する組織の期間と一致します。
    組織分類所属の期間コードは対象の組織の期間コードと同一です。
  - 関連付けられている組織の期間と無効化状態が連動します。
    関連づいている組織の期間が無効化された場合、該当の組織分類所属も無効化されます。
    組織の期間が有効化された場合も連動して有効化します。
  - 組織分類項目が無効化される際、このエンティティの関連するレコードは物理削除されます。
  - 組織が物理削除される際、このエンティティの関連するレコードも物理削除されます。
  - 組織分類項目が物理削除される際、このエンティティの関連するレコードも物理削除されます。

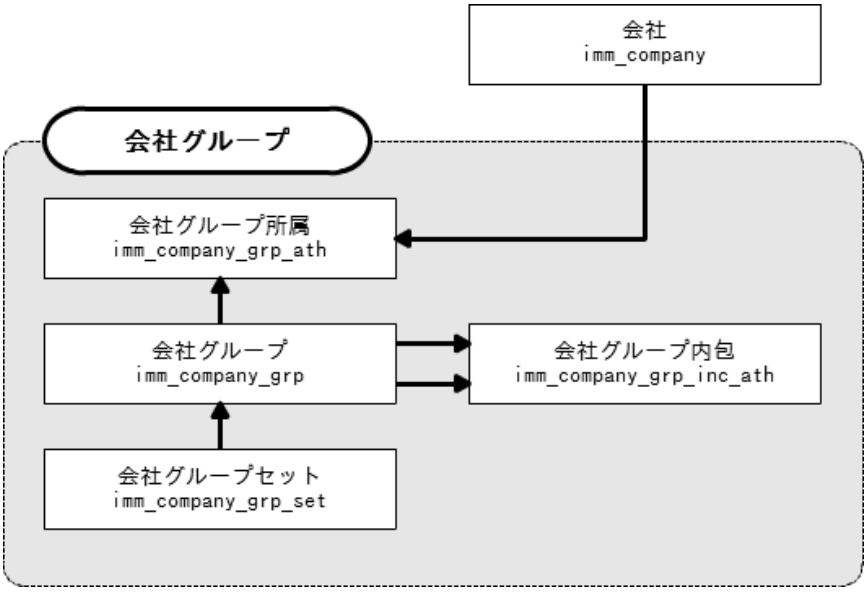
会社グループ

IM-共通マスタでは会社を会社グループに関連付けて、会社情報をさらに大きなグループ管理します。
会社を階層構造で管理することのできる情報です。

データ構造

会社グループの構成

会社グループの構成に関連するER図を【図：会社グループの構成に関連するER図】に示します。



【図：会社グループの構成に関連するER図】

【図：会社グループの構成に関連するER図】において各エンティティは次のような役割・制約があります。

- 会社グループセット
  - 会社の集合の概念を管理します。
    会社自身の詳細情報は組織エンティティで管理します。
  - 内包構造の一部です。
- 会社グループ
  - 期間化国際化の基本構造です。
  - 会社グループセットや会社グループセット内に存在する会社グループの詳細情報を管理します。
  - 会社グループセットが物理削除される際、このエンティティの関連レコードも物理削除されます。
- 会社グループ内包
  - 内包構造のエンティティです。
  - 会社グループセット内の会社グループ構成の情報を管理します。
  - 会社グループセットが物理削除される際、このエンティティの関連レコードも物理削除されます。
- 会社グループ所属
  - 会社がどの会社グループに所属するかを管理します。
  - 所属構造のエンティティです。
  - 会社が物理削除された場合、このエンティティの関連レコードも物理削除されます。
  - 会社グループが物理削除された場合、このエンティティの関連レコードも物理削除されます。
- 会社
  - 会社を管理します。詳細は「*会社と組織の構成* 」を参照してください。

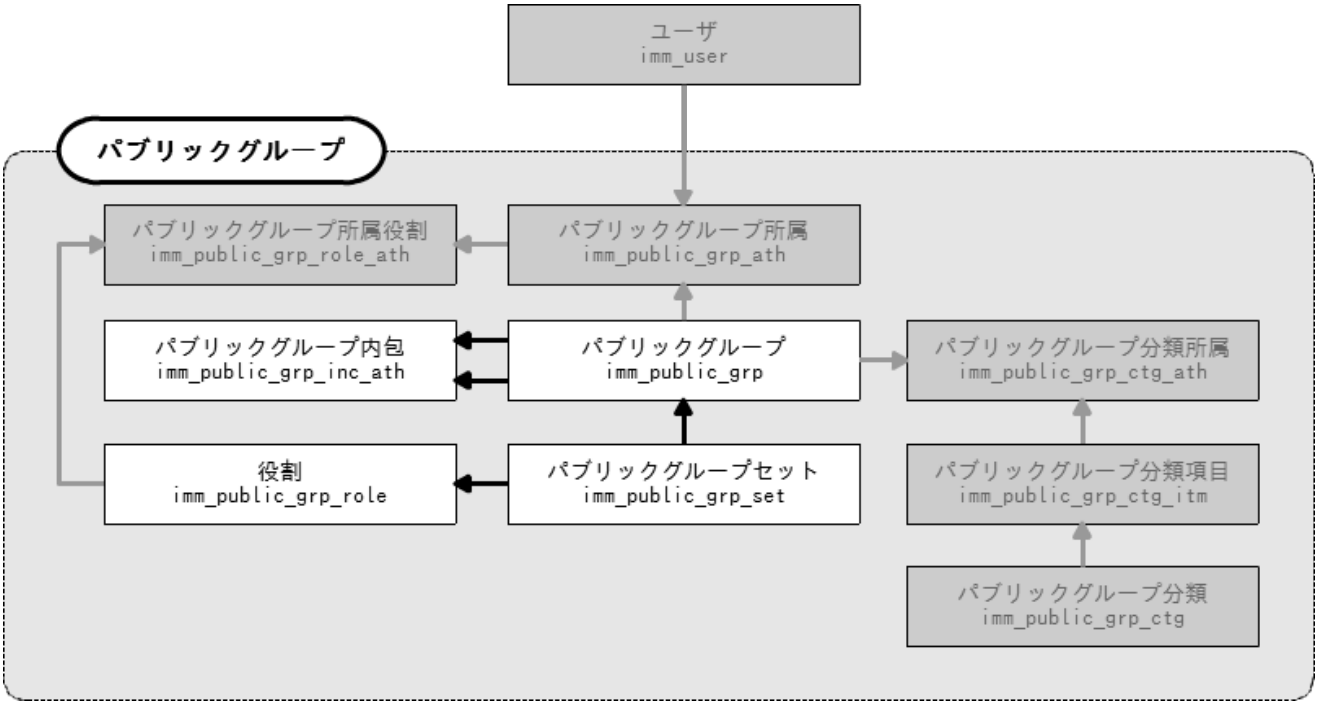
パブリックグループ

「*会社と組織* 」では会社や組織を扱うときの情報について説明しました。
しかし、そのような概念には当てはまらない団体も存在する場合があります（サークル、非営利団体等）。
IM-共通マスタではこのような団体をパブリックグループとして情報を扱うことができます。
取り扱える情報には以下のようなものがあります。

- パブリックグループそのものの情報
- グループ構成の情報
- グループ構成の履歴管理
- グループに所属するユーザ
- グループの分類情報
- グループにおけるユーザの役割

パブリックグループの構成

パブリックグループセットおよびパブリックグループの構成に関連するER図を【図：パブリックグループの構成に関連するER図】に示します。



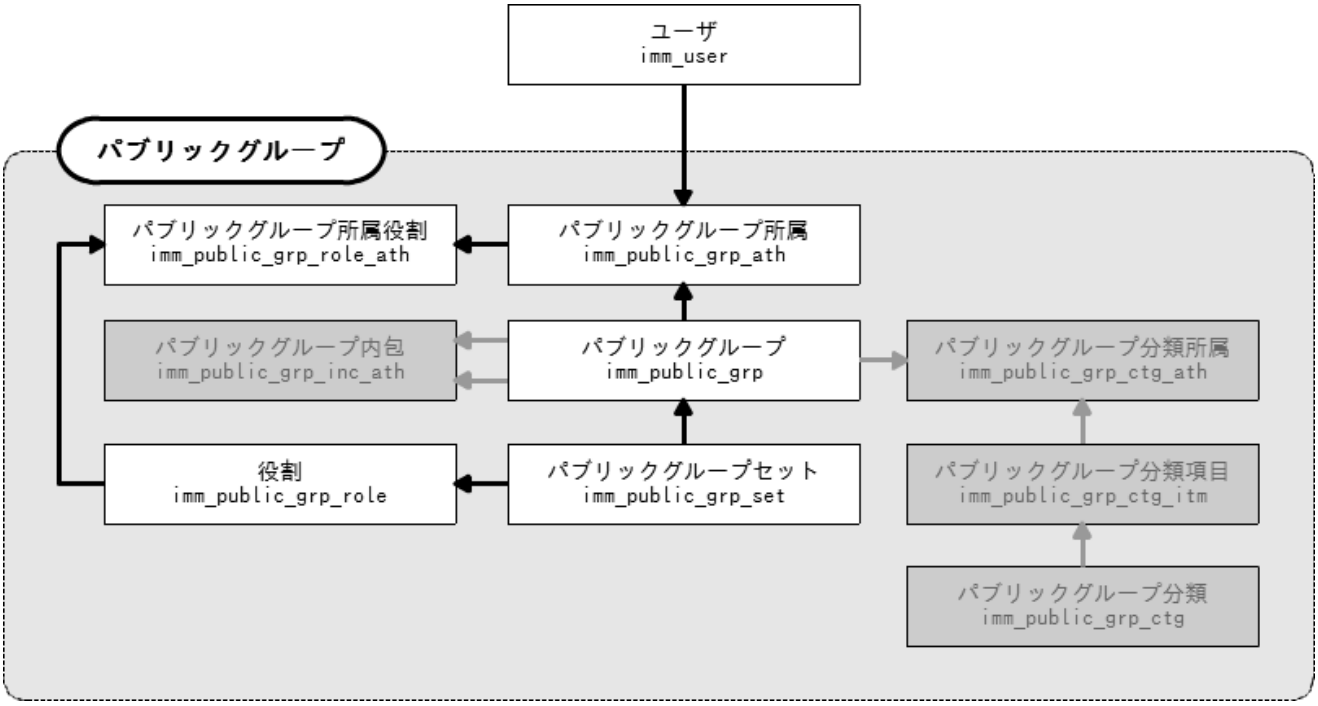
【図：パブリックグループの構成に関連するER図】

【図：パブリックグループの構成に関連するER図】において各エンティティは次のような役割・制約があります。

- パブリックグループセット
  - パブリックグループの集合の概念を管理します。
    パブリックグループセット自身の詳細情報はパブリックグループエンティティで管理します。
  - 内包構造の一部です。
- パブリックグループ
  - 期間化国際化の基本構造です。
  - パブリックグループセットやパブリックグループセット内に存在するパブリックグループの詳細情報を管理します。
  - パブリックグループセットが物理削除される際、このエンティティの関連レコードも物理削除されます。
- パブリックグループ内包
  - 内包構造のエンティティです。
  - パブリックグループセット内のパブリックグループ構成の情報を管理します。
  - パブリックグループセットが物理削除される際、このエンティティの関連レコードも物理削除されます。
- 役割
  - パブリックグループ内で定義されているパブリックグループセット毎に役割の情報を管理するエンティティです。
  - 期間国際化の基本構造です。
  - 定義されているパブリックグループセットを物理削除すると、このエンティティの関連レコードもあわせて物理削除されます。
  - ランクを設定することが出来、その役割の地位の高さを表します。
    - 値が小さいほど地位が高いものとして扱います。
    - ランクは期間化も国際化もされません。

パブリックグループへの所属

ユーザが所属するパブリックグループセットおよびパブリックグループに関連するER図を【図：パブリックグループとユーザの所属に関連するER図】に示します。



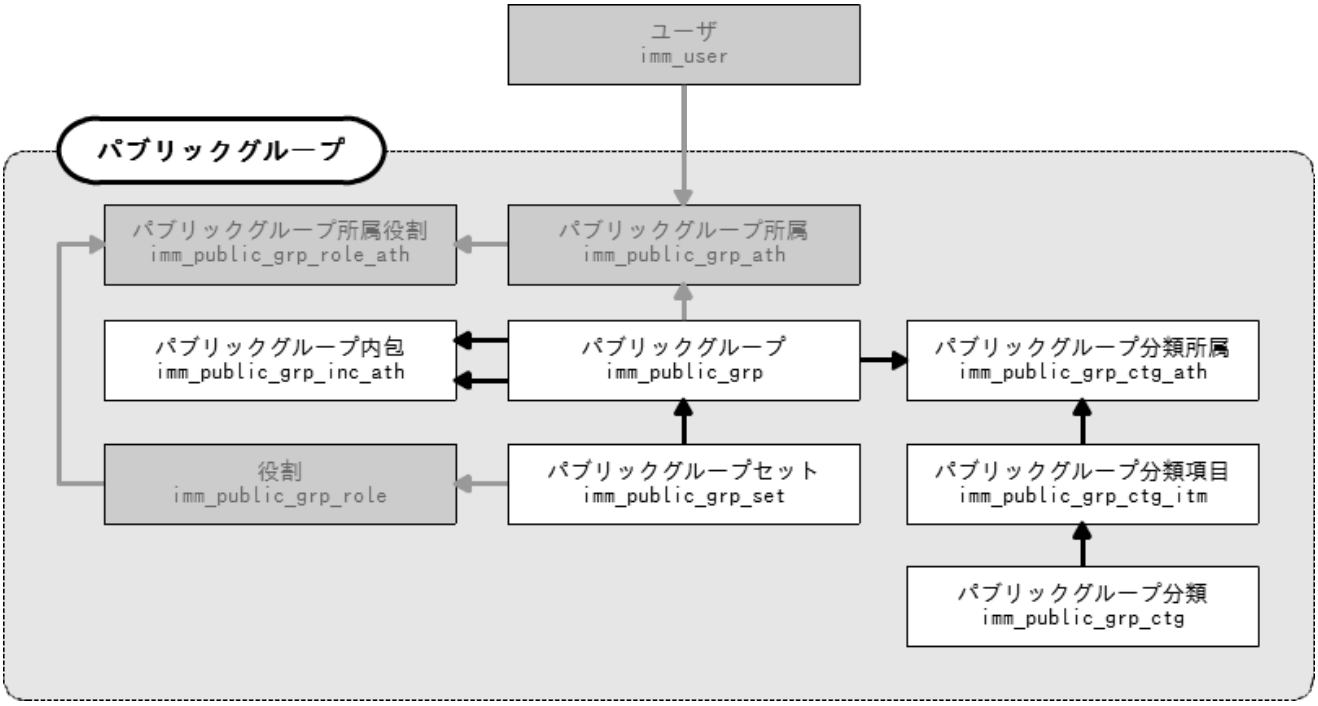
【図：パブリックグループとユーザの所属に関連するER図】

【図：パブリックグループとユーザの所属に関連するER図】において各エンティティは次のような役割・制約があります。

- ユーザを管理します。詳細は「*ユーザ*」を参照してください。
- パブリックグループ
  - パブリックグループの詳細情報を管理します。詳細は「*パブリックグループの構成*」を参照してください。
- 役割
  - 役割を管理します。詳細は「*パブリックグループの構成*」を参照してください。
- パブリックグループ所属
  - ユーザがどの期間、どのパブリックグループに所属するかを管理します。
  - 所属構造のエンティティです。
  - ユーザが物理削除された際に、あわせてこのエンティティの関連レコードも削除されます。
  - 役割が物理削除された際に、あわせてこのエンティティの関連レコードも削除されます。
- パブリックグループ所属役割
  - ユーザがパブリックグループに所属する期間について、どの役割に就いているかを管理します。
    複数の役割を兼任することが出来ます。
  - 期間属性構造です。組織所属を参照します。
  - 期間コード、開始日、終了日、削除フラグがパブリックグループ所属と連動します。
    パブリックグループ所属が無効化されると、関連するパブリックグループ所属役職も無効化されます。有効化した場合も同様です。パブリックグループ所属の期間が変更されると開始日、終了日が連動して更新されます。
  - ユーザが物理削除された際に、あわせてパブリックグループ所属役職の関連レコードも削除されます。
  - 役割が物理削除された際に、あわせてパブリックグループ所属役職の関連レコードも削除されます。

パブリックグループの分類

パブリックグループセットおよびパブリックグループの分類に関連するER図を【図：パブリックグループの分類に関連するER図】に示します。



【図：パブリックグループの分類に関連するER図】

【図：パブリックグループの分類に関連するER図】において各エンティティは次のような役割・制約があります。

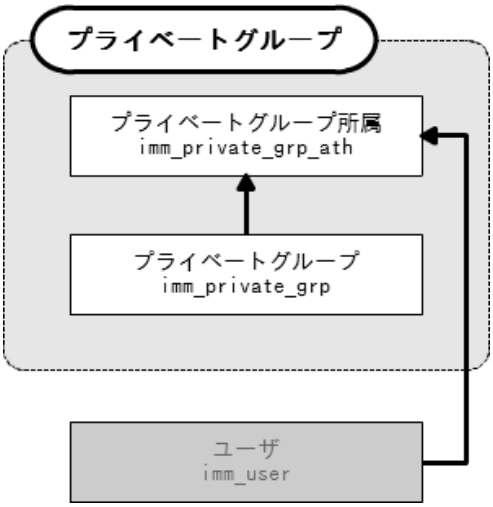
- パブリックグループ分類
  - パブリックグループ分類の種類（例えば公開ゾーンなど）を管理します。
  - 無効化すると、配下の分類項目も連動して無効化します。また分類項目が無効化されると、関連する分類所属エンティティが物理削除されます。
- パブリックグループ分類項目
  - 分類の種類ごとに定義されている詳細（イントラネットゾーン、エクストラネットゾーンなど）を管理します。
  - 属する組織分類エンティティが無効化されると、連動してこのエンティティも無効化状態になります。
- パブリックグループ分類所属
  - パブリックグループが分類項目に分類されていることを表します。
  - パブリックグループが期間化されている場合はその期間単位に分類します。
  - 期間は分類するパブリックグループの期間と一致します。
    パブリックグループ分類所属の期間コードは対象のパブリックグループの期間コードと同一です。
  - 関連付けられているパブリックグループの期間と無効化状態が連動します。
    関連づいているパブリックグループの期間が無効化された場合、該当のパブリックグループ分類所属も無効化されます。
    パブリックグループの期間が有効化された場合も連動して有効化します。
  - パブリックグループ分類項目が無効化される際、このエンティティの関連するレコードは物理削除されます。
  - パブリックグループが物理削除される際、このエンティティの関連するレコードも物理削除されます。
  - パブリックグループ分類項目が物理削除される際、このエンティティの関連するレコードも物理削除されます。
- パブリックグループ
  - パブリックグループセットやパブリックグループセット内に存在するパブリックグループの詳細情報を管理します。詳細は「*パブリックグループの構成*」を参照してください。

プライベートグループ

会社やパブリックグループはシステムで決定する団体ですが、個人でグループを管理したい場合はプライベートグループを利用することができます。

データ構造

プライベートグループに関連するER図を【図：プライベートグループに関連するER図】に示します。



【図：プライベートグループに関連するER図】

【図：プライベートグループに関連するER図】において各エンティティは次のような役割・制約があります。

- ユーザ
  - ユーザを管理します。詳細は「ユーザ」を参照してください。
- プライベートグループ
  - プライベートグループの詳細情報を管理します。
  - 非期間国際化の基本構造です。
  - 削除フラグもありません。
- プライベートグループ所属
  - ユーザがどのプライベートグループに所属するかを管理します。
  - 所属構造ですが、期間化はされていません。
  - ユーザが物理削除された場合、このエンティティの関連レコードも物理削除されます。

プライベートグループの情報

プライベートグループは会社やパブリックグループと同様に団体を扱う情報です。
しかし、プライベートグループは個人で扱えることを最優先の目的としているため、簡素な管理体系となっています。
そのためプライベートグループは会社やパブリックグループとは以下の点で異なります。

- 階層構造を持ちません。
- バージョンによる管理がされません。
- 期間化や国際化がされていません。

> **コラム**
>
> ユーザはプライベートグループを複数定義することが可能です。
> 複数のプライベートグループのグループコードが同じでも、プライベートグループを定義したユーザが異なれば別のプライベートグループとして扱われます。

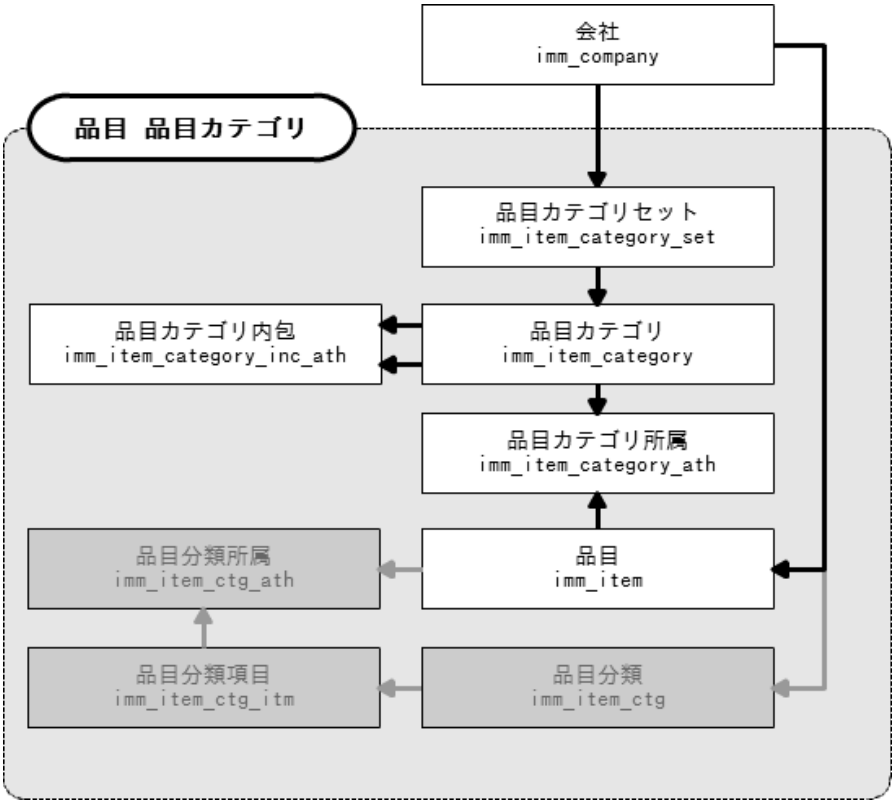
品目

IM-共通マスタでは品目に関連する情報を扱うことができます。
基本的には拡張情報を追加して使用することを想定しており、初期状態では取り扱える情報はほとんどありません。
取り扱える情報には以下のようなものがあります。

- 品目についての情報（名称、コードなどの基本的な情報）
- 品目を階層的にカテゴライズして整理・管理する情報
- 品目に付与する属性的な情報である分類情報
- 会社ごとにアクセス可能な情報の管理

品目・品目カテゴリ

品目カテゴリに関するER図を【図：品目カテゴリに関連するER図】に示します。



【図：品目カテゴリに関連するER図】

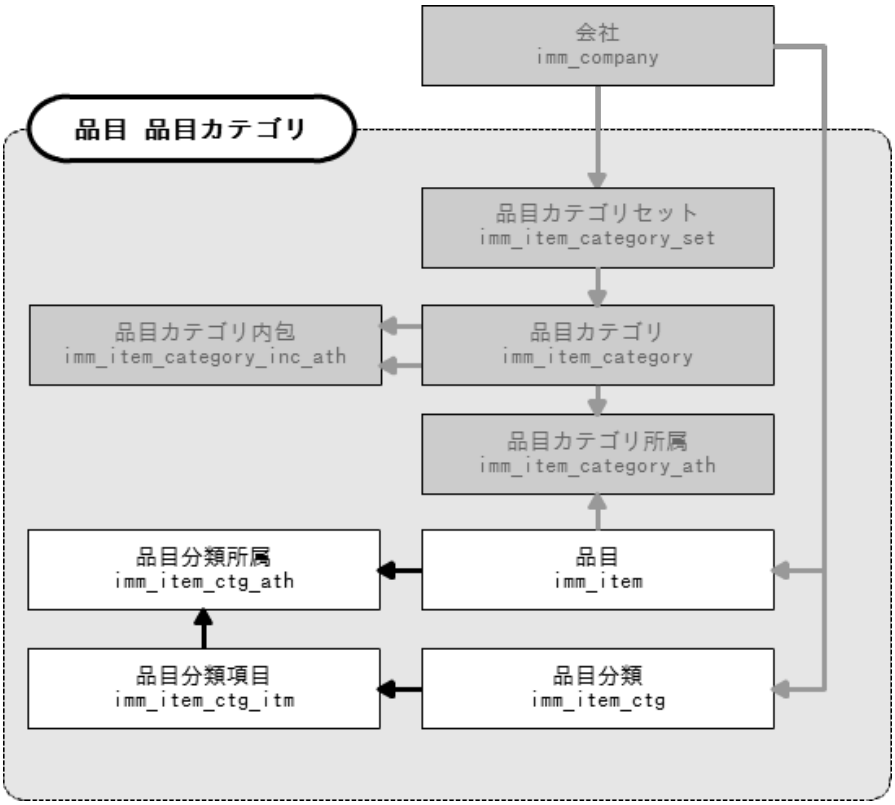
【図：品目カテゴリに関連するER図】において各エンティティは次のような役割・制約があります。

- 品目カテゴリセット
  - 品目をカテゴリ分けする際カテゴリ群として管理します。その品目カテゴリ群を代表するエンティティです。
  - 内包構造のエンティティの一部です。
  - 品目カテゴリセットは会社ごとに定義します。
  - 会社が物理削除されると、このエンティティの関連レコードも物理削除されます。
- 品目カテゴリ内包
  - 階層化された品目カテゴリの構造を管理します。
  - 内包構造のエンティティです。
  - 品目カテゴリセットが物理削除されると、このエンティティの関連レコードも物理削除されます。
- 品目カテゴリ
  - 品目をカテゴライズするためのカテゴリをあわらすエンティティです。
  - 期間国際化された基本構造エンティティです。
  - 品目カテゴリセットが物理削除されると、このエンティティの関連レコードも物理削除されます。
  - 品目カテゴリは会社ごとに定義します。
  - 会社が物理削除されると、このエンティティの関連レコードも物理削除されます。
- 品目カテゴリ所属
  - 品目がどのカテゴリに所属するかを管理します。
  - 所属構造のエンティティです。
  - 品目が物理削除された場合、このエンティティの関連レコードも物理削除されます。
  - 品目カテゴリが物理削除された場合、このエンティティの関連レコードも物理削除されます。
- 品目
  - 品目の情報を管理します。
  - 期間国際化された基本構造エンティティです。

- 品目は会社毎に定義します。
- 会社が物理削除されると、このエンティティの関連レコードも物理削除されます。

品目分類の構成

品目分類の構成に関するER図を【図：品目分類に関連するER図】に示します。



【図：品目分類に関連するER図】

【図：品目分類に関連するER図】において各エンティティは次のような役割・制約があります。

- 品目分類
  - 品目分類の種類を管理します。
  - 無効化すると、配下の分類項目も連動して無効化します。また分類項目が無効化されると、該当の品目分類所属エンティティが物理削除されます。
  - 品目分類は会社ごとに定義します。
  - 会社が物理削除されると、このエンティティの関連レコードも物理削除されます。
- 品目分類項目
  - 分類の種類ごとに定義される項目を管理します。
  - 品目分類エンティティが無効化された場合、連動してこのエンティティの関連するレコードも無効化状態になります。
  - 品目分類が物理削除される際、このエンティティの関連レコードも物理削除されます。
  - 品目分類項目は会社ごとに定義します。
- 品目分類所属
  - 品目が分類項目に分類されていることを表します。
  - 品目が期間化されている場合はその期間単位に分類します。
  - 期間は分類する品目の期間と一致します。品目分類所属の期間コードは対象の品目の期間コードと同一です。
  - 関連付けられている品目の期間と無効化状態が連動します。関連づいている品目の期間が無効化された場合、該当の品目分類所属も無効化されます。品目の期間が有効化された場合も連動して有効化します。
  - 品目分類項目が無効化される際、このエンティティの関連するレコードは物理削除されます。
  - 品目が物理削除される際、このエンティティの関連するレコードも物理削除されます。
  - 品目分類項目が物理削除される際、このエンティティの関連するレコードも物理削除されます。
  - 品目分類所属は会社ごとに定義します。
- 品目

- 品目を管理します。詳細は「*品目・品目カテゴリ* 」を参照してください。

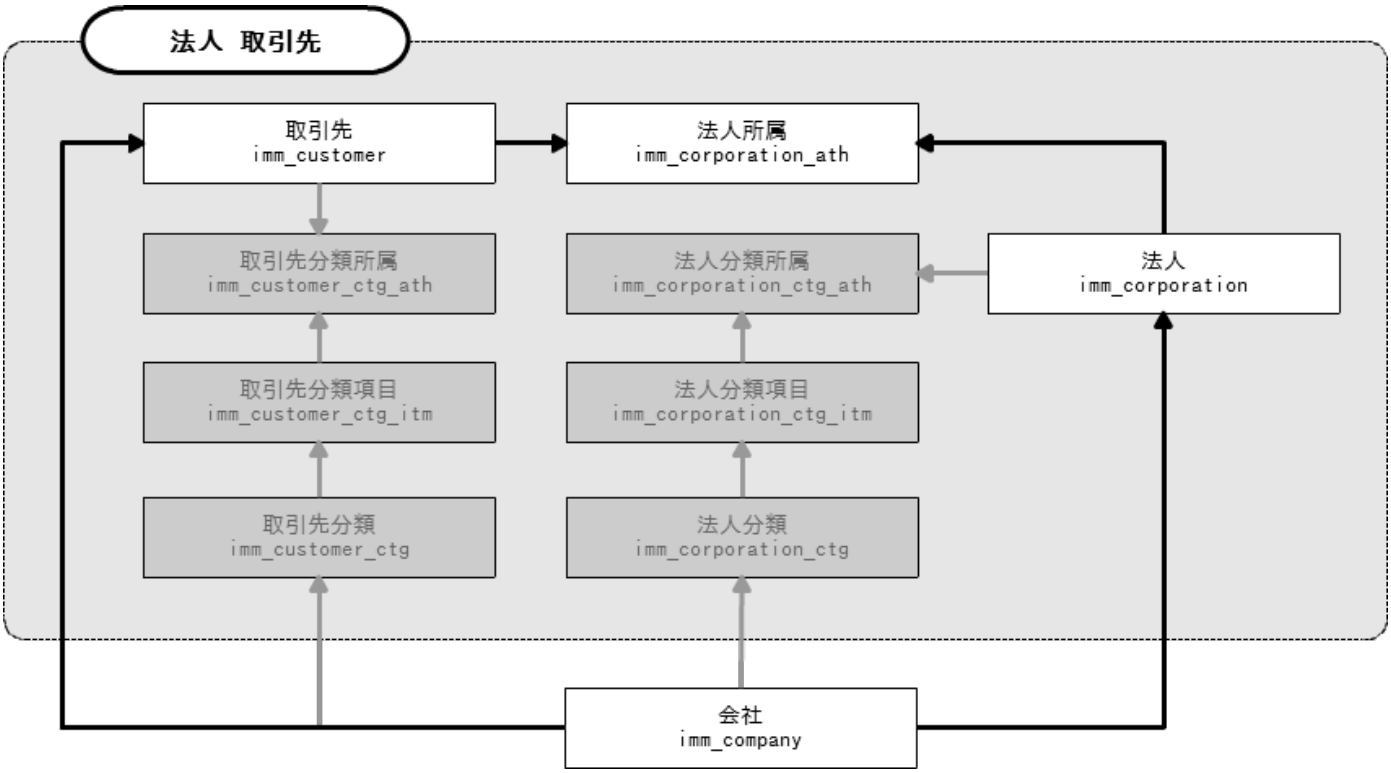
法人・取引先

IM-共通マスタでは取引先を法人に関連付けて管理します。
取引先は法人に所属させることのできる基本構造の情報です。

データ構造

法人・取引先

法人の構成に関するER図を【図：法人に関連するER図】に示します。



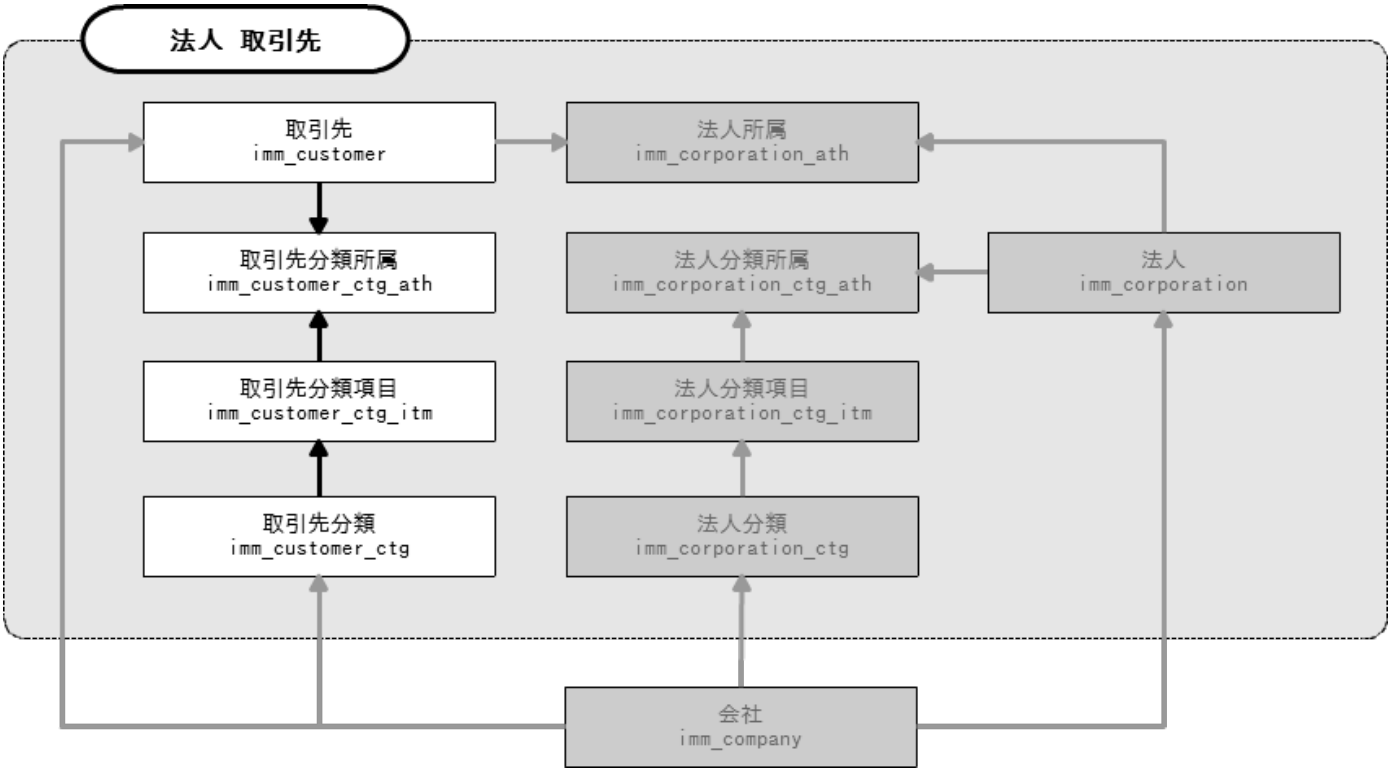
【図：法人に関連するER図】

【図：法人に関連するER図】において各エンティティは次のような役割・制約があります。

- 法人
  - 特定の法人を会社ごとに管理します。
  - 期間国際化された基本構造エンティティです。
- 法人所属
  - 取引先がどの法人に所属するかを管理します。
  - 所属構造のエンティティです。
  - 取引先が物理削除された場合、このエンティティの関連レコードも物理削除されます。
  - 法人が物理削除された場合、このエンティティの関連レコードも物理削除されます。
- 取引先
  - 取引先を会社ごとに管理します。
  - 期間国際化された基本構造エンティティです。

取引先分類

取引先分類に関するER図を【図：取引先分類に関するER図】に示します。



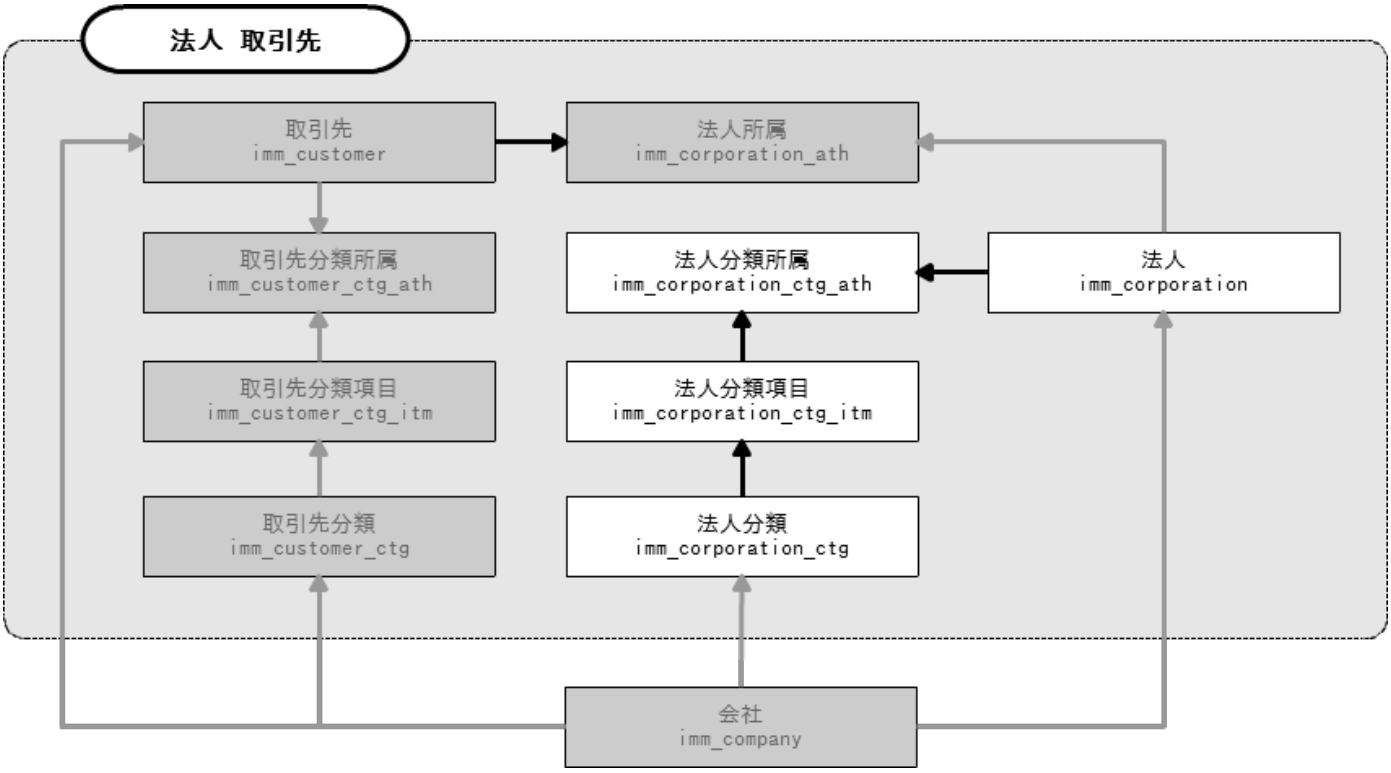
【図：取引先分類に関するER図】

【図：取引先分類に関するER図】において各エンティティは次のような役割・制約があります。

- 取引先分類
  - 取引先分類の種類を管理します。
  - 無効化すると、配下の分類項目も連動して無効化します。
    また分類項目が無効化されると、該当の取引先分類所属エンティティが物理削除されます。
- 取引先分類項目
  - 分類の種類ごとに定義される項目を管理します。
  - 取引先分類エンティティが無効化された場合、連動してこのエンティティの関連するレコードも無効化状態になります。
  - 取引先分類が物理削除される際、このエンティティの関連レコードも物理削除されます。
- 取引先分類所属
  - 取引先が分類項目に分類されていることを表します。
  - 取引先が期間化されている場合はその期間単位に分類します。
  - 期間は分類する取引先の期間と一致します。
    取引先分類所属の期間コードは対象の取引先の期間コードと同一です。
  - 関連付けられている取引先の期間と無効化状態が連動します。
    関連づいている取引先の期間が無効化された場合、該当の取引先分類所属も無効化されます。
    取引先の期間が有効化された場合も連動して有効化します。
  - 取引先分類項目が無効化される際、このエンティティの関連するレコードは物理削除されます。
  - 取引先が物理削除される際、このエンティティの関連するレコードも物理削除されます。
  - 取引先分類項目が物理削除される際、このエンティティの関連するレコードも物理削除されます。
- 取引先
  - 取引先を管理します。詳細は「*法人・取引先*」を参照してください。

法人分類

法人分類に関するER図を【図：法人分類に関するER図】に示します。



【図：法人分類に関するER図】

【図：法人分類に関するER図】において各エンティティは次のような役割・制約があります。

- 法人分類
  - 法人分類の種類を管理します。
  - 無効化すると、配下の分類項目も連動して無効化します。
    また分類項目が無効化されると、該当の法人分類所属エンティティが物理削除されます。
- 法人分類項目
  - 分類の種類ごとに定義される項目を管理します。
  - 法人分類エンティティが無効化された場合、連動してこのエンティティの関連するレコードも無効化状態になります。
  - 法人分類が物理削除される際、このエンティティの関連レコードも物理削除されます。
- 法人分類所属
  - 法人が分類項目に分類されていることを表します。
  - 法人が期間化されている場合はその期間単位に分類します。
  - 期間は分類する法人の期間と一致します。
    法人分類所属の期間コードは対象の法人の期間コードと同一です。
  - 関連付けられている法人の期間と無効化状態が連動します。
    関連づいている法人の期間が無効化された場合、該当の法人分類所属も無効化されます。
    法人の期間が有効化された場合も連動して有効化します。
  - 法人分類項目が無効化される際、このエンティティの関連するレコードは物理削除されます。
  - 法人が物理削除される際、このエンティティの関連するレコードも物理削除されます。
  - 法人分類項目が物理削除される際、このエンティティの関連するレコードも物理削除されます。
- 法人
  - 法人を管理します。詳細は「*法人・取引先*」を参照してください。

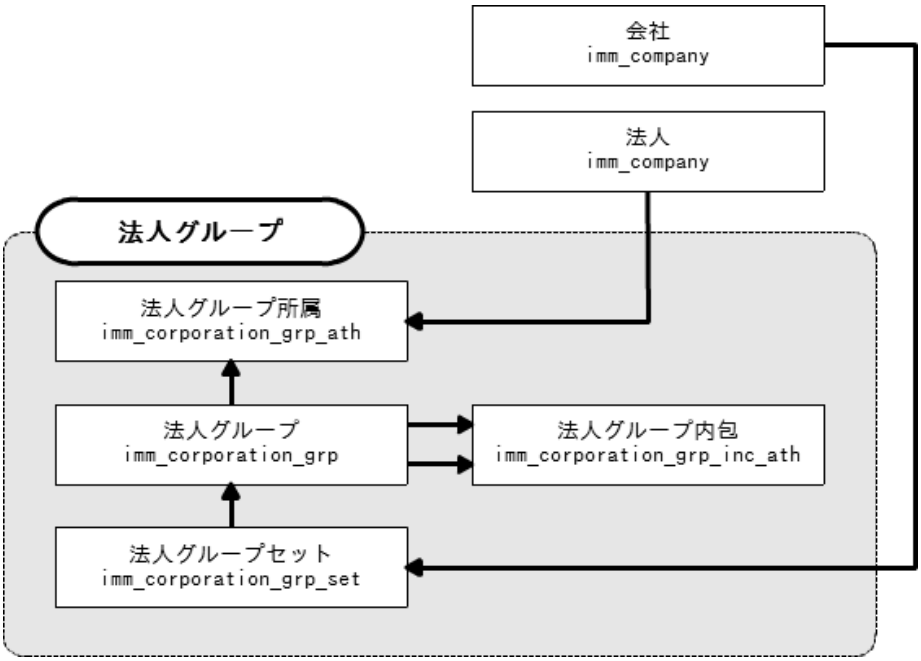
法人グループ

IM-共通マスタでは法人を法人グループに関連付けて管理します。
法人を階層構造で管理する、会社別に管理することのできる情報です。

データ構造

法人グループ

法人グループの構成に関連するER図を【図：法人グループの構成に関連するER図】に示します。



【図：法人グループの構成に関連するER図】

【図：法人グループの構成に関連するER図】において各エンティティは次のような役割・制約があります。

- 法人グループセット
  - 法人の集合の概念を管理します。法人自身の詳細情報は法人エンティティで管理します。
  - 内包構造の一部です。
  - 会社が物理削除されると、このエンティティの関連レコードも物理削除されます。
- 法人グループ
  - 期間化国際化の基本構造です。
  - 法人グループセットや法人グループセット内に存在する法人グループの詳細情報を管理します。
  - 法人グループセットが物理削除される際、このエンティティの関連レコードも物理削除されます。
- 法人グループ内包
  - 内包構造のエンティティです。
  - 法人グループセット内の法人グループ構成の情報を管理します。
  - 法人グループセットが物理削除される際、このエンティティの関連レコードも物理削除されます。
- 法人グループ所属
  - 法人がどの法人グループに所属するかを管理します。
  - 所属構造のエンティティです。
  - 法人が物理削除された場合、このエンティティの関連レコードも物理削除されます。
  - 法人グループが物理削除された場合、このエンティティの関連レコードも物理削除されます。
- 法人
  - 法人を管理します。詳細は「*法人・取引先* 」を参照してください。

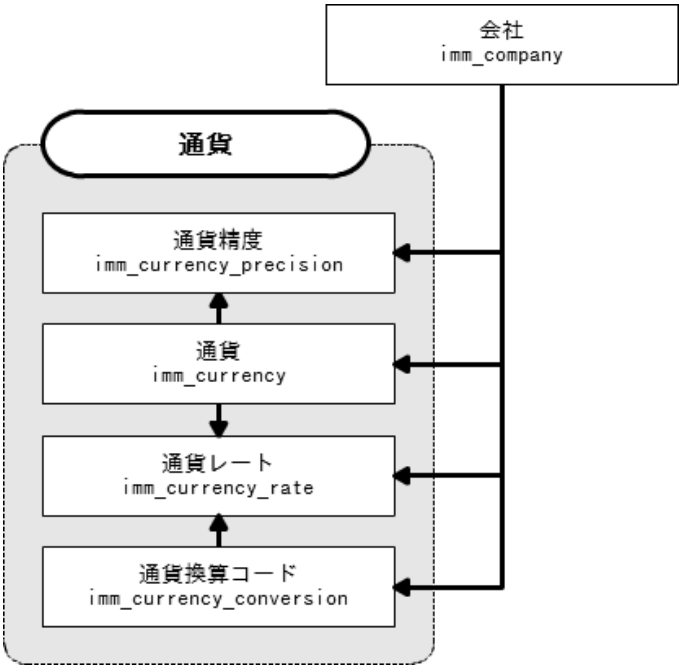
通貨

IM-共通マスタでは、各国で使用される通貨を管理することができます。
ある通貨を別の通貨に換算するための情報も含まれています。

データ構造

法人グループ

通貨に関するER図を【図：通貨に関するER図】に示します。



【図：通貨に関するER図】

【図：通貨に関するER図】において、各エンティティは次のような役割・制約があります。

- 通貨
  - 通貨の種類を管理します。
  - 無効化すると、配下の通貨精度、通貨レートも連動して無効化します。
  - 会社が物理削除されると、このエンティティの関連レコードも物理削除されます。
- 通貨換算コード
  - 通貨レートの所属を管理します。
  - 通貨換算コードエンティティが無効化された場合、連動してこのエンティティの関連するレコードも無効化状態になります。
  - 通貨を物理削除した際、このエンティティの関連レコードも物理削除されます。
  - 会社が物理削除されると、このエンティティの関連レコードも物理削除されます。
- 通貨レート
  - 2つの通貨の換算レートを管理します。
  - 通貨を物理削除した際、このエンティティの関連レコードも物理削除されます。
  - 通貨換算コードを物理削除した際、このエンティティの関連レコードも物理削除されます。
  - 会社が物理削除されると、このエンティティの関連レコードも物理削除されます。
- 通貨精度
  - 通貨の有効桁数を管理します。
  - 通貨換算コードエンティティが無効化された場合、連動してこのエンティティの関連するレコードも無効化状態になります。
  - 通貨を物理削除した際、このエンティティの関連レコードも物理削除されます。
  - 会社が物理削除されると、このエンティティの関連レコードも物理削除されます。

その他

ここまでで説明した以外でIM-共通マスタで用意しているテーブルについて説明します。
IM-共通マスタで取り扱っている残りのテーブルのER図を【図：その他のテーブルについてのER図】に示します。

【図：その他のテーブルについてのER図】

【図：その他のテーブルについてのER図】において各エンティティは次のような役割・制約があります。

- 単位
  - システムで用いる単位を保持する管理テーブルです。
- 期間開始管理
  - システム開始日付を保持する管理テーブルです。

API

IM-共通マスタとアプリケーションデータの整合性について

IM-共通マスタに登録、更新および削除を行う場合、データの整合性を保つ必要があります。
また、IM-共通マスタは比較的汎用的に設計されていますが、独自に拡張したい場合もあります。

この場合、直接データベースにデータを登録、更新および削除したり、テーブル構造を変更したりすると、同じ情報を共有するアプリケーションへ想定外の影響を及ぼしたり、様々な不具合を併発してしまう可能性が考えられます。

これらの問題を解決するため、intra-mart ではIM-共通マスタに対する API を用意しています。
API を利用すると、以下の利点があります。

- IM-共通マスタ間の整合性の考慮が必要なくなる
- テーブルを拡張したい場合に、拡張したい対象のテーブルと同期して独自に拡張した情報を取り扱うことが可能になる

> **注意**
>
> 各アプリケーションおよびテンプレートは、IM-共通マスタで使用しているテーブルに対してデータを直接登録、更新および削除してはいけません。
> IM-共通マスタで使用しているテーブルのデータに変更する処理を行う場合、必ずこの API で行う必要があります。

IM-共通マスタ間の整合性の確保

IM-共通マスタはデータベース上の複数のテーブルから構成されますが、APIを利用することで整合性の考慮は不要になります。
例えば組織の登録、更新および削除時には組織テーブルと組織内包テーブル等、複数のテーブルにアクセスする必要がありますが、APIを利用すると1回の呼び出しで全てのテーブルが更新されます。
また、組織テーブルや組織内包テーブル内に保持している期間の操作をする必要がある場合、その期間の整合性を維持するためにテーブル内の複数のレコードを更新する必要がありますが、これもAPIを使用することによって整合性が確保されます。

テーブル拡張の実現

テーブルの情報を拡張したい場合は、APIが登録・更新・削除などを行った際に、それぞれのタイミングで呼び出される「リスナー」を設定しておくことで、IM-共通マスタの内容変更に対し同期を取って拡張情報を処理することが出来るようになります。

この仕組みにより、 intra-mart のマスタメンテナンス画面を使用してIM-共通マスタに対する変更を行っても、インストールされているすべてのプロダクトに対してデータの整合性が維持されます。
アプリケーションからIM-共通マスタのデータを登録および編集する場合でも、APIを使用することで同様にデータの整合性を保つことが出来るようになります。

マネージャ

IM-共通マスタのデータを変更または取得するとき、マネージャと呼ばれるAPI群を使用します。
IM-共通マスタのどのデータを扱うかによって使用するマネージャが変わります。
【表：マネージャー覧】にその一覧を示します。
APIの詳細は「APIリスト(Javadoc)」を参照してください。

取り扱う情報	スクリプト開発モデル API	J2EE開発モデル API
ユーザ	IMMUserManager	jp.co.intra_mart.foundation.master.user.UserManager

取り扱う情報	スクリプト開発モデル API	J2EE開発モデル API
会社グループ	IMMCompanyGroupManager	jp.co.intra_mart.foundation.master.company_group.CompanyGroupManager
会社組織	IMMCompanyManager	jp.co.intra_mart.foundation.master.company.CompanyManager
パブリックグループ	IMMPublicGroupManager	jp.co.intra_mart.foundation.master.public_group.PublicGroupManager
プライベートグループ	IMMPrivateGroupManager	jp.co.intra_mart.foundation.master.private_group.PrivateGroupManager
品目	IMMItemManager	jp.co.intra_mart.foundation.master.item.ItemManager
品目カテゴリ	IMMItemCategoryManager	jp.co.intra_mart.foundation.master.item_category.ItemCategoryManager
取引先	IMMCustomerManager	jp.co.intra_mart.foundation.master.customer.CustomerManager
法人	IMMCorporationManager	jp.co.intra_mart.foundation.master.corporation.CorporationManager
法人グループ	IMMCorporationGroupManager	jp.co.intra_mart.foundation.master.corporation_group.CorporationGroupManager
通貨	IMMCurrencyManager	jp.co.intra_mart.foundation.master.currency.CurrencyManager

【表：マネージャ一覧】

また、各マネージャが使用するエンティティおよびエンティティを構成するテーブルを【表：マネージャが使用するエンティティとテーブルの一覧】に示します。

マネージャ	エンティティ	テーブル
UserManager	ユーザ	imm_user
	ユーザ分類	imm_user_ctg
	ユーザ分類項目	imm_user_ctg_itm
	ユーザ分類所属	imm_user_ctg_ath
CompanyGroupManager	会社グループ	imm_company_grp
	会社グループセット	imm_company_grp_set
	会社グループ内包	imm_company_grp_inc_ath
	会社グループ所属	imm_company_grp_ath
CompanyManager	会社	imm_company

	組織	imm_department
	会社組織セット	imm_department_set
	会社組織内包	imm_department_inc_ath
	役職	imm_company_post
	組織所属役職	imm_department_post_ath
	組織所属	imm_department_ath
	組織分類	imm_department_ctg
	組織分類項目	imm_department_ctg_itm
	組織分類所属	imm_department_ctg_ath
PublicGroupManager	パブリックグループセット	imm_public_grp_set
	パブリックグループ	imm_public_grp
	パブリックグループ内包	imm_public_grp_inc_ath
	パブリックグループ所属	imm_public_grp_ath
	役割	imm_public_grp_role
	パブリックグループ所属役割	imm_public_grp_role_ath
	パブリックグループ分類	imm_public_grp_ctg
	パブリックグループ分類項目	imm_public_grp_ctg_itm
	パブリックグループ分類所属	imm_public_grp_ctg_ath
PrivateGroupManager	プライベートグループ	imm_private_grp
	プライベートグループ所属	imm_private_grp_ath
ItemManager	品目	imm_item
ItemCategoryManager	品目カテゴリ	imm_item_category
	品目カテゴリセット	imm_item_category_set
	品目カテゴリ内包	imm_item_category_inc_ath
	品目カテゴリ所属	imm_item_category_ath
CustomerManager	取引先	imm_customer
CorporationManager	法人	imm_corporation
	法人所属	imm_corporation_ath
CorporationGroupManager	法人グループセット	imm_corporation_grp_set
	法人グループ	imm_corporation_grp
	法人グループ内包	imm_corporation_grp_inc_ath
	法人グループ所属	imm_corporation_grp_ath
CurrencyManager	通貨	imm_currency
	通貨精度	imm_currency_precision
	通貨換算コード	imm_currency_conversion
	通貨レート	imm_currency_rate
AppCommonManager	期間開始管理	imm_start_date
	期間終了管理	—

【表：マネージャが使用するエンティティとテーブルの一覧】

項目
- マネージャの取得
  - J2EE開発モデルの場合
  - スクリプト開発モデルの場合
- マネージャの使用
  - J2EE開発モデルの場合
  - スクリプト開発モデルの場合
- マネージャによる検索
  - J2EE開発モデルの場合
    - 詳細情報を取得する場合
    - 一覧情報を取得する場合
  - スクリプト開発モデルの場合
    - 詳細情報を取得する場合
    - 一覧情報を取得する場合

マネージャの取得

IM-共通マスタではさまざまなマネージャを用意していますが、マネージャの取得方法は同様です。
以下にJ2EE開発モデルおよびスクリプト開発モデルにおけるマネージャの取得方法について説明します。

J2EE開発モデルの場合

マネージャはコンストラクタを使用してインスタンスを生成できます。
この際、検索などの際にデフォルトとして使用するロケールや基準日などを引数に指定してインスタンスを生成できます。
【リスト：CompanyManagerの取得例（JavaEE開発モデル）】に会社に関連するマネージャ（CompanyManager）を取得する例を示します。

```
/* マネージャの取得 */
CompanyManager manager = new CompanyManager();
```

【リスト：CompanyManagerの取得例（JavaEE開発モデル）】

スクリプト開発モデルの場合

スクリプト開発モデルではマネージャは定義済みのオブジェクトとして定義されています。
マネージャを取得する場合、それぞれのオブジェクトのコンストラクタを呼び出すことでインスタンスを取得できます。
これらのコンストラクタは、検索などの際にデフォルトとして使用するロケールや基準日などを引数に指定してインスタンスを生成できます。

【リスト：CompanyManagerの取得例（スクリプト開発モデル）】に会社に関連するマネージャ（CompanyManager）を取得する例を示します。

```
  // ・・・
/* マネージャの取得 */
var manager = new IMMCompanyManager();
  // ・・・
```

【リスト：CompanyManagerの取得例（スクリプト開発モデル）】

マネージャの使用

マネージャの各メソッドを呼び出すことでIM-共通マスタ対してデータの登録や削除などが行えます。
これらのメソッドはトランザクションが有効となっている状態で呼び出す必要があります。
以下にJava EE開発モデルおよびスクリプト開発モデルにおけるマネージャの使用方法について説明します。

J2EE開発モデルの場合

Java EE開発モデルのマネージャのメソッド（get～ を除きます）はユーザトランザクション内で実行されることが前提です。
【リスト：CompanyManagerの使用例（Java EE開発モデル）】に会社に関連するマネージャ（CompanyManager）を使用して会社情報を新規に登録するときの例を示します。

```
/* マネージャの取得 */
CompanyManager manager = new CompanyManager();

/* 会社組織情報の作成 */
Department department = new Department();
department.setDepartmentCd("dept_1");
department.setDepartmentSetCd("dept_set_1");
department.setCompanyCd("comp_1");

// ・・・中略・・・

/* マネージャを使用して登録 */
manager.setDepartment(department);
```

【リスト：CompanyManagerの使用例（Java EE開発モデル）】

スクリプト開発モデルの場合

スクリプト開発モデルのマネージャのメソッド（get～ を除きます）はユーザトランザクション内で実行されることが前提です。
【リスト：CompanyManagerの使用例（スクリプト開発モデル）】に会社に関連するマネージャ（CompanyManager）を使用して会社情報を新規に登録するときの例を示します。

```
  // ・・・
/* トランザクションの開始 */
Transaction.begin(function() {
    // ・・・
  /* マネージャの取得 */
  var manager = new IMMCompanyManager();
  var department = new Object();
  department.termCd = null;
    // ・・・
  /* 組織情報の生成 */
    // ・・・
  manager.setDepartment(department); // マネージャを使用して登録
});
  // ・・・
```

【リスト：CompanyManagerの使用例（スクリプト開発モデル）】

マネージャによる検索

いくつかのマネージャにはデータを取得するためのメソッド（詳細については「APIリスト(Javadoc)」を参照してください）が提供されています。
このメソッドは大きく分けて以下の２つに分類されます。

- 一覧を取得するメソッド

  検索条件に一致するデータを配列として取得します。
  取得されたデータには、基準日を含む期間のログインユーザのデフォルト言語（または指定された言語）に一致する情報が設定されます。
- 詳細を取得するメソッド

  検索条件に一致するデータを１件だけ取得します。
  期間化されているエンティティの場合、取得されたデータには、基準日を含む期間のすべての国際化情報が設定されます。

次項ではJavaEE開発モデルとスクリプト開発モデルそれぞれで、会社組織に関連するマネージャ（CompanyManager）を使用し

て組織情報を検索・取得する場合の例を示します。

J2EE開発モデルの場合

詳細情報を取得する場合

Java EE開発モデルのマネージャの詳細取得メソッドを使用すると、結果はDTO（Data Transfer Object）として返却されます。
【リスト：CompanyManagerによる詳細検索の例（Java EE開発モデル）】に会社に関連するマネージャ（CompanyManager）を使用して組織の詳細情報を取得するときの例を示します。

```
/* マネージャの取得 */
CompanyManager manager = new CompanyManager();

/* キーの作成 */
DepartmentBizKey bizkey = new DepartmentBizKey();
bizkey.setCompanyCd("comp_1");
bizkey.setDepartmentCd("dept_1");
bizkey.setDepartmentSetCd("dept_set_1");

/* マネージャを使用して詳細情報を取得 */
Department department = manager.getDepartment(bizkey, new Date());
  // ・・・
```

【リスト：CompanyManagerによる詳細検索の例（Java EE開発モデル）】

一覧情報を取得する場合

Java EE開発モデルのマネージャの一覧取得メソッドを使用すると、結果はDTOの配列として返却されます。
【リスト：CompanyManagerによる一覧検索の例（Java EE開発モデル）】に会社に関連するマネージャ（CompanyManager）を使用して組織に所属するユーザの一覧を取得するときの例を示します。

```
/* マネージャの取得 */
CompanyManager manager = new CompanyManager();

/* 検索条件の作成 */
AppCmnSearchCondition condition = new AppCmnSearchCondition();
condition.setLogicalOperetor(LogicalOperator.OR);
condition.setSortDirection(SortDirection.ASC);
condition.addCondition("department_cd", "dept_1", Operator.EQ);

/* マネージャを使用して検索 */
DepartmentListNode[] node = manager.listDepartment(condition, new Date(), Locale.JAPAN);
  // ・・・
```

【リスト：CompanyManagerによる一覧検索の例（Java EE開発モデル）】

スクリプト開発モデルの場合

詳細情報を取得する場合

スクリプト開発モデルのマネージャの詳細取得メソッドを使用すると、結果は処理結果オブジェクトに格納されて返却されます。
【リスト：CompanyManagerによる詳細検索の例（スクリプト開発モデル）】に会社に関連するマネージャ（CompanyManager）を使用して組織の詳細情報を取得するときの例を示します。

```
  // ・・・
/* マネージャの取得 */
var companyManager = new IMMCompanyManager();

/* ビジネスキーオブジェクトの作成 */
var key = new Object();
key.departmentCd = "dept_1";
key.departmentSetCd =  "dept_set_1";
key.companyCd = "comp_1";

var res = companyManager.getDepartment(key, new Date(), false);
  // ・・・
```

【リスト：CompanyManagerによる詳細検索の例（スクリプト開発モデル）】

一覧情報を取得する場合

スクリプト開発モデルのマネージャの一覧取得メソッドを使用すると、結果は処理結果オブジェクトに配列として格納されて返却されます。
【リスト：CompanyManagerによる一覧検索の例（スクリプト開発モデル）】に会社に関連するマネージャ（CompanyManager）を使用して組織に所属するユーザの一覧を取得するときの例を示します。

```
  // ・・・
/* 検索条件の作成 */
var condition =  new AppCmnSearchCondition();
condition.setLogicalOperetor(AppCmnSearchCondition.OR);
condition.addLike("name","サンプル会社",AppCmnSearchCondition.PARTIAL);
condition.addOrder("sort_key");

var result;

/* マネージャの取得 */
var manager = new IMMCompanyManager();

/* 会社組織の検索処理 */
result = manager.listDepartment(condition, new Date(), "ja", 0, 20,true);

if (!result.error) {
  /* 正しく結果が取得できた */
  for ( i = 0; i < result.data.length; i++) {
    result.data[i].code = result.data[i].departmentCd;
    // ・・・
  }
}else{
  /* エラー処理 */
  // ・・・
}
  // ・・・
```

【リスト：CompanyManagerによる一覧検索の例（スクリプト開発モデル）】

メソッド命名則

ここではマネージャの持つメソッドの命名規則について概説します。
APIリストには多数のメソッドが列挙されていますが、概ねここに説明する命名規則に従っています。

> **コラム**
> 名前付けのパターンによって機能が決まっているため、命名則を把握しておくことで機能の概要を知ることが出来ます。

項目

- 記法について
- コンストラクタについて
- 基本構造と所属構造
  - 検索系
    - list*
    - search*
    - total*
    - count*
  - 詳細取得
    - get
  - 更新系
    - set*
    - remove*
- 内包構造
  - 検索
    - listTreeRoot
    - searchTreeRoot
    - totalTreeRoot
    - countTreeRoot
    - List<A>with<B>UpTree
    - Search<A>with<B>UpTree
    - Total<A>with<B>UpTree
    - Count<A>with<B>UpTree
  - 取得
    - getAbsoluteBranch, getBranch
    - getAbsoluteUpBranch, getUpBranch
    - getAbsoluteParent, getParent
    - getAbsoluteIsolation, getIsolation
    - getAbsoluteChildren, getChildren
    - getFullPathListNode
  - 更新
    - set<A>Inclusion
- 期間操作
  - 取得
    - get<A>Term
    - get<A>TermList
  - 更新
    - mergeBackwordTerm<A>
    - mergeForwardTerm<A>
    - moveTerm*
    - separateTerm*

記法について

以下エンティティの構造毎に説明しますが、一部特殊な記述があります。
その内容を【表：特殊な記述について】にて説明します。

記号	意味

記号	意味
*	ワイルドカードを表します。 list* という表記はlist のほかlistDepartmentやlistCompanyといった名称群の全体を表しています。
<A>	何らかの名を表します。 一箇所で<A>, <B>, <C>と出てきた場合はそれぞれ異なるエンティティ名を指すと捉えて下さい。

【表：特殊な記述について】

コンストラクタについて

IM-共通マスタのマネージャクラスのコンストラクタは一律同様の引数を受け取ります。
以下に、IMー共通マスタのマネージャクラスが受け取るコンストラクタ引数それぞれについて説明します。

- 更新ユーザコード

  マネージャを使用して登録や変更など更新系の処理を行う場合にここで指定されたユーザコードが使用されます。
  全てのテーブルに存在する更新ユーザ、または登録ユーザのカラムに設定されます。
- デフォルトロケール

  検索や取得など参照系の処理に対して影響します。
  マネージャから取得した結果のデフォルトのロケールがここで指定したロケールになります。
  更新系処理の場合は必要なロケール全てに対し更新を行うため、この値は影響しません。

基本構造と所属構造

検索系

list*

検索して該当する結果の一覧を取得します。結果はListNodeを継承した型になります。
後述するsearch*と動作の内容が似ていますが、listは指定されたLocaleの情報が存在しないエンティティでも結果に返します。
以下の命名パターンがあります。

- list<A>
  エンティティAに対して検索します。
- list<A>with<B>
  関連のあるエンティティAとBに対して、エンティティBを条件にしてエンティティAを検索します。
- list<A>with<B>On<C>
  関連のあるエンティティAとBとCに対して、エンティティBとエンティティCへの関連が存在するエンティティAを検索します。

search*

検索して該当する結果の一覧を取得します。結果はListNodeを継承した型になります。
list*と動作の内容が似ていますが、searchは指定されたLocaleの情報が存在しないエンティティは結果に含まれません。
以下の命名パターンがあります。

- search<A>
  エンティティAに対して検索します。
- search<A>with<B>
  関連のあるエンティティAとBに対して、エンティティBを条件にしてエンティティAを検索します。
- search<A>with<B>On<C>
  関連のあるエンティティAとBとCに対して、エンティティBとエンティティCへの関連が存在するエンティティAを検索します。

total*

指定された条件に一致する件数を返します。
list同様Localeに関係なく対象のビジネスキーが存在する件数を返します。

- total<A>
  エンティティAに対して検索し、件数を返します。
- total<A>with<B>
  関連のあるエンティティAとBに対して、エンティティBを条件にしてエンティティAを検索し、件数を返します。
- total<A>with<B>On<C>
  関連のあるエンティティAとBとCに対して、エンティティBとエンティティCへの関連が存在するエンティティAを検索し、件数を返します。

count*

指定された条件に一致する件数を返します。
search同様指定Locale情報の存在するエンティティの件数を返します。

- count<A>
  エンティティAに対して検索し、件数を返します。
- count<A>with<B>
  関連のあるエンティティAとBに対して、エンティティBを条件にしてエンティティAを検索し、件数を返します。
- count<A>with<B>On<C>
  関連のあるエンティティAとBとCに対して、エンティティBとエンティティCへの関連が存在するエンティティAを検索し、件数を返します。

詳細取得

get

- get<A>
  エンティティAに対して、指定の基準日における全ロケールのデータを単一のオブジェクトとして取得して返します。

更新系

set*

引数に渡された情報でデータベースを更新します。キーを比較し、すでにテーブルに存在している場合は更新、そうでない場合は登録処理を行います。

remove*

引数に渡されたビジネスキー情報でデータベースからエンティティを削除します。
このメソッドは指定のビジネスキーを持つ全期間の情報を物理削除します。
論理削除状態にしたい場合はset*を使って削除フラグを更新してください。

内包構造

検索

listTreeRoot

指定の検索条件で内包構造のルート（トップ階層）を検索し、一覧を返します。
指定された言語情報が存在しなくても結果を返す点は基本構造のlistと同様です。

searchTreeRoot

指定の検索条件で内包構造のルート（トップ階層）を検索し、一覧を返します。

指定された言語情報が存在しない場合結果に含めないのは基本構造のsearchと同様です。

totalTreeRoot

指定の検索条件で内包構造のルート（トップ階層）を検索し、該当件数を返します。
指定された言語情報が存在しなくてもカウントする点は基本構造のtotalと同様です。

countTreeRoot

指定の検索条件で内包構造のルート（トップ階層）を検索し、該当件数を返します。
指定された言語情報が存在しない場合カウントしない点は基本構造のcountと同様です。

List<A>with<B>UpTree

エンティティBの特定のノードとその上位ノードを条件に、エンティティAから該当するノードに所属する一覧を返します。
指定された言語情報が存在しなくても結果を返す点は基本構造のlistと同様です。

Search<A>with<B>UpTree

エンティティBの特定のノードとその上位ノードを条件に、エンティティAから該当するノードに所属する一覧を返します。
指定された言語情報が存在しない場合結果に含めないのは基本構造のsearchと同様です。

Total<A>with<B>UpTree

エンティティBの特定のノードとその上位ノードを条件に、エンティティAから該当するノードに所属する件数を返します。
指定された言語情報が存在しなくてもカウントする点は基本構造のtotalと同様です。

Count<A>with<B>UpTree

エンティティBの特定のノードとその上位ノードを条件に、エンティティAから該当するノードに所属する件数を返します。
指定された言語情報が存在しない場合カウントしない点は基本構造のcountと同様です。

取得

getAbsoluteBranch, getBranch

内包構造において、指定したノードから下位の部分ツリーを取得します。
getAbsoluteBranchは指定された言語情報の有無にかかわらず、存在するノードは取得して返します。
getBranchは指定の言語が存在しないノードは返しません。

getAbsoluteUpBranch, getUpBranch

内包構造において、指定したノードから上位の部分ツリーを取得します。
getAbsoluteUpBranchは指定された言語情報の有無にかかわらず、存在するノードは取得して返します。
getUpBranchは指定の言語が存在しないノードは返しません。

getAbsoluteParent, getParent

内包構造において、特定のノードの親ノードの一覧を返します。
getAbsoluteParentは指定された言語情報の有無にかかわらずノードの情報を取得して返します。
getParentは指定の言語が存在しないノードは返しません。

getAbsoluteIsolation, getIsolation

内包構造において、指定した基準日において構成に属していないノードを取得します。
getAbsoluteIsolationは指定された言語情報の有無にかかわらずノード情報を取得して返します。
getIsolationは指定の言語が存在しないノードは返しません。

getAbsoluteChildren, getChildren

内包構造において、特定のノードの配下の子ノードの一覧を返します。
getAbsoluteChildrenは指定された言語情報の有無にかかわらずノードの情報を取得して返します。
getChildrenは指定の言語が存在しないノードは返しません。

getFullPathListNode

ノードの記述名(descriptionプロパティ)を、内包構造をパス形式で表現した名称に置き換えます。

更新

set<A>Inclusion

内包する側のエンティティAと内包される側のエンティティA'と、内包する期間の期間コードを指定して、内包情報を更新・追加します。

期間操作

取得

get<A>Term

エンティティAについて指定されたビジネスキーと、基準日で街頭する期間情報（単一）を取得します。

get<A>TermList

エンティティAについて指定されたビジネスキーに定義されている期間の一覧を取得します。

更新

mergeBackwordTerm<A>

ビジネスキーと期間コードを指定して、特定の期間をその直後の期間とマージします。
直後の期間が削除される形で結合されます。

mergeForwardTerm<A>

ビジネスキーと期間コードを指定して、特定の期間をその直前の期間とマージします。
直前の期間が削除される形で結合されます。

moveTerm*

特定の期間コードに対して、開始日と終了日を指定し、期間を変更します。
隣接する期間は自動的に伸縮されます。隣接する期間を超えて伸ばしたり移動させたりした隣接する期間が伸縮するだけで入れ替わりが起こることはありません。
隣接する期間がつぶされた期間は自動的に削除されます。

separateTerm*

特定の期間を特定の日付で分割します。
分割した直後は分割されたどちらの期間も開始日・終了日以外は同じ内容になります。
分割された後半の期間の期間コードが新たに振られます。

リスナー

マネージャはさまざまな更新処理（登録や削除を含みます）を行います。
このとき、マネージャ本来の処理の後に独自の処理を追加することができます。これをリスナーといいます。
リスナーはマネージャの更新処理に対してPluginとして追加することが出来ます。

リスナーの種類

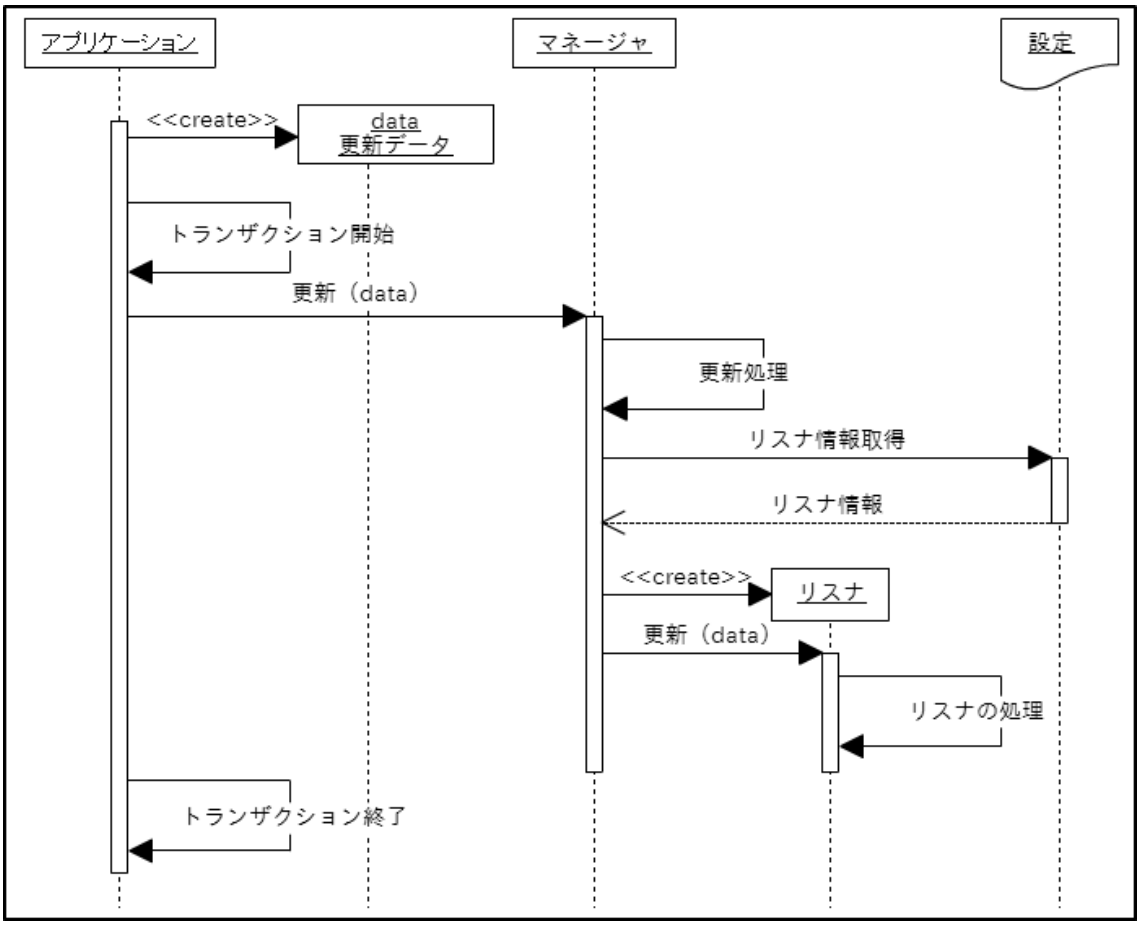
リスナーはIM-共通マスタで提供しているAppCommonManager以外のマネージャに対して追加する事が出来ます。
リスナーはマネージャ毎に呼び出すタイミングが決まっており、大きく分類して以下のようなタイミングで呼び出しを行います。

- 基本となる情報を追加、更新、削除した時点
- 期間を分割、更新した時点
- 内包構造のセットを作成したり、内包関係を追加、削除、更新した時点
- 所属の追加、更新、解除の時点

それぞれ、必要に応じて必要なタイミングの処理を実装する必要があります。
リスナーの具体的な実装方法や、仕様の詳細については「IM-共通マスタ 拡張プログラミングガイド」を参照して下さい。

リスナーの動作

リスナーの動作の概要を【図：リスナーの動作概要】に示します。



【図：リスナーの動作概要】

1. アプリケーションは更新（登録や削除を含みます）用のオブジェクトを生成します。
2. アプリケーションはトランザクションを開始します。
3. アプリケーションはマネージャに対して更新処理を依頼します。
4. マネージャは本来の更新処理を行います。
5. マネージャは設定（Plugin.xml）をもとにリスナー情報を取得します。
6. マネージャは取得できたリスナーに対し、取得できた順に更新用のオブジェクトを渡して呼び出しを行います。
7. リスナーでは更新用のオブジェクトを参照し、独自の処理を行います。
8. マネージャおよびすべてのリスナーの処理が正常終了した場合、アプリケーションはトランザクションをコミットします。

一部でも例外を返した場合、アプリケーションはトランザクションをロールバックする必要があります。

補足事項

略称を持つデータ（会社グループ、組織、パブリックグループ、法人、取引先、品目カテゴリ、品目、法人グループ、通貨精度、通貨換算コード）を登録、更新する場合に、略称が空であるとAPIが自動的に名称をコピーします。

付録

キャッシュ

IM-共通マスタの一部のデータは頻繁にアクセスされます。そこで、IM-共通マスタでは、一部の機能において処理を高速化するためにキャッシュを使用しています。
本章では、IM-共通マスタで利用するキャッシュについて説明します。

項目

- 概要
- 会社一覧キャッシュ
  - キャッシュ内容
  - キャッシュするオブジェクトの単位
  - キャッシュサイズの計算式
- 会社一覧のリソースグループ設定キャッシュ
- 会社一覧のポリシー設定キャッシュ
- プロファイル画像キャッシュ
  - キャッシュ内容
  - キャッシュするオブジェクトの単位
  - キャッシュサイズの計算式

概要

IM-共通マスタでは、一部の処理を高速化するために以下についてキャッシュを使用しています。

項番	名称	導入バージョン	備考
1	*会社一覧キャッシュ*	2015 Winter(Lydia)	
2	*会社一覧のリソースグループ設定キャッシュ*	2015 Winter(Lydia)	
3	*会社一覧のポリシー設定キャッシュ*	2015 Winter(Lydia)	
4	*プロファイル画像キャッシュ*	2018 Summer(Tiffany)	

会社一覧キャッシュサイズのデフォルト値には、会社数を 100 、システムで利用可能な言語数を 3 としてサイジングした値が設定されています。
プロファイル画像キャッシュサイズのデフォルト値には、ユーザ数を 100 、サムネイルサイズを100 × 100と48 × 48の2種としてサイジングした値が設定されています。

デフォルト値のサイジングの詳細については、各キャッシュの計算式を参照してください。

> **コラム**
>
> 設定可能な値については「設定ファイルリファレンス キャッシュ設定」を参照してください。

会社一覧キャッシュ

キャッシュ内容

システム日付時点の会社情報をキャッシュします。
キャッシュサイズは、「IM-共通マスタキャッシュ設定（im-ehcache-config/im_master.xml）」で設定されています。

> **コラム**
>
> このキャッシュは intra-mart Accel Platform 2015 Winter(Lydia) 以降で利用可能です。

キャッシュするオブジェクトの単位

国際化情報毎に一覧でキャッシュします。

例えば、システムで利用可能な言語が３言語だった場合は、一覧を３個キャッシュします。

キャッシュサイズの計算式

キャッシュが行われる対象データのおおまかなサイズは以下の計算式で求めることができます。

キャッシュサイズ ＝ （１） × （２） × （３）

（１） ... 会社数
（２） ... システムで利用可能な言語数
（３） ... 会社情報のバイト数 （約 2500byte）

intra-mart Accel Platform のデフォルト値は以下の計算をもとに設定されています。

100 × 3 × 2,500 ＝ 750,000 （約 732.4KB）
⇒ 750KB

会社一覧のリソースグループ設定キャッシュ

キャッシュ内容やキャッシュするオブジェクトの単位、キャッシュサイズの計算式については、「認可仕様書」-「認可のキャッシュ設定」を参照してください。

会社一覧のポリシー設定キャッシュ

キャッシュ内容やキャッシュするオブジェクトの単位、キャッシュサイズの計算式については、「認可仕様書」-「認可のキャッシュ設定」を参照してください。

プロファイル画像キャッシュ

キャッシュ内容

プロファイル画像のデータURLをキャッシュします。
キャッシュサイズは、「IM-共通マスタキャッシュ設定（im-ehcache-config/im_master.xml）」で設定されています。

> **コラム**
>
> このキャッシュは intra-mart Accel Platform 2018 Summer(Tiffany) 以降で利用可能です。

キャッシュするオブジェクトの単位

ユーザ毎にサムネイル化したプロファイル画像のデータURLをキャッシュします。オリジナルサイズはキャッシュしません。
サムネイルのサイズは 「 設定ファイルリファレンス 」-「 IM-共通マスタ設定 」 で設定を行います。

例えば、設定したサムネイルサイズが3種だった場合は、ユーザ毎に画像のデータURLを3個ずつキャッシュします。

キャッシュサイズの計算式

キャッシュが行われる対象データのおおまかなサイズは以下の計算式で求めることができます。

キャッシュサイズ ＝ （1） × （（2） ＋ （2） ...）

（1） ... ユーザ数
（2） ... 設定したサムネイルサイズ毎のプロファイル画像のデータURL （100 × 100 の場合約40kb、48 × 48 の場合約10kb）

> **コラム**
>
> （2）のプロファイル画像のデータURLは設定したサムネイルサイズ毎の画像の大まかなサイズをすべて足し合わせます。

intra-mart Accel Platform のデフォルト値は以下の計算をもとに設定されています。

100 × （40 ＋ 10） ＝ 5,000 （約 4.8MB）
⇒ 5MB

IM-LogicDesigner のフロートリガで IM-共通マスタ を使用する

IM-LogicDesigner のフロートリガで IM-共通マスタ のカテゴリを使用する方法について説明します。

> **コラム**
>
> この設定は intra-mart Accel Platform 2018 Spring(Skylark) から提供しています。

> **注意**
>
> IM-共通マスタ をトリガに指定してロジックフローを実行する場合、アプリケーションサーバの処理が増加することによって、システム全体の動作に影響を与える可能性があります。

設定手順

1. IM-Propagation の受信側（リスナ）を設定をします。
   %JUGGLING_PROJECT_PATH%/conf 配下に propagation-receivers-config ディレクトリを作成します。
   propagation-receivers-config 配下に im_master.xml を作成します。

   ファイルパス：
   %JUGGLING_PROJECT_PATH%/conf/propagation-receivers-config/im_master.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<propagation-receivers-config
 xmlns="http://www.intra-mart.jp/propagation/receivers-config"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.intra-mart.jp/propagation/receivers-config propagation-receivers-config.xsd">

 <receiver source="jp.co.intra_mart.foundation.master.user.model.User" operationType="DATA_CREATED">
  <decoder
class="jp.co.intra_mart.system.master.user.propagation.receiver.logic.trigger.CreatedUserProfileDecoder" />
  <procedure
class="jp.co.intra_mart.system.master.user.propagation.receiver.logic.trigger.CreatedUserProfileProcedure" />
 </receiver>
 <receiver source="jp.co.intra_mart.foundation.master.user.model.User" operationType="DATA_UPDATED">
  <decoder
class="jp.co.intra_mart.system.master.user.propagation.receiver.logic.trigger.UpdatedUserProfileDecoder" />
  <procedure
class="jp.co.intra_mart.system.master.user.propagation.receiver.logic.trigger.UpdatedUserProfileProcedure" />
 </receiver>

 <receiver source="jp.co.intra_mart.system.master.user.propagation.model.PropagationDepartment"
operationType="DATA_CREATED">
  <decoder
```

```
class="jp.co.intra_mart.system.master.user.propagation.receiver.logic.trigger.CreatedDepartmentUserAttachDecoder"
/>
    <procedure
class="jp.co.intra_mart.system.master.user.propagation.receiver.logic.trigger.CreatedDepartmentUserAttachProcedure
/>
  </receiver>
  <receiver source="jp.co.intra_mart.system.master.user.propagation.model.PropagationDepartment"
operationType="DATA_UPDATED">
    <decoder
class="jp.co.intra_mart.system.master.user.propagation.receiver.logic.trigger.UpdatedDepartmentUserAttachDecoder"
/>
    <procedure
class="jp.co.intra_mart.system.master.user.propagation.receiver.logic.trigger.UpdatedDepartmentUserAttachProcedur
/>
  </receiver>
  <receiver source="jp.co.intra_mart.system.master.user.propagation.model.PropagationDepartment"
operationType="DATA_DELETED">
    <decoder
class="jp.co.intra_mart.system.master.user.propagation.receiver.logic.trigger.DeletedDepartmentUserAttachDecoder"
/>
    <procedure
class="jp.co.intra_mart.system.master.user.propagation.receiver.logic.trigger.DeletedDepartmentUserAttachProcedure
/>
  </receiver>

  <receiver source="jp.co.intra_mart.system.master.user.propagation.model.PropagationPublicGroup"
operationType="DATA_CREATED">
    <decoder
class="jp.co.intra_mart.system.master.user.propagation.receiver.logic.trigger.CreatedPublicGroupUserAttachDecoder"
/>
    <procedure
class="jp.co.intra_mart.system.master.user.propagation.receiver.logic.trigger.CreatedPublicGroupUserAttachProcedure
/>
  </receiver>
  <receiver source="jp.co.intra_mart.system.master.user.propagation.model.PropagationPublicGroup"
operationType="DATA_UPDATED">
    <decoder
class="jp.co.intra_mart.system.master.user.propagation.receiver.logic.trigger.UpdatedPublicGroupUserAttachDecoder"
/>
    <procedure
class="jp.co.intra_mart.system.master.user.propagation.receiver.logic.trigger.UpdatedPublicGroupUserAttachProcedur
/>
  </receiver>
  <receiver source="jp.co.intra_mart.system.master.user.propagation.model.PropagationPublicGroup"
operationType="DATA_DELETED">
    <decoder
class="jp.co.intra_mart.system.master.user.propagation.receiver.logic.trigger.DeletedPublicGroupUserAttachDecoder"
/>
    <procedure
class="jp.co.intra_mart.system.master.user.propagation.receiver.logic.trigger.DeletedPublicGroupUserAttachProcedure
/>
  </receiver>

</propagation-receivers-config>
```

2. IM-Propagation の送信側（トリガ）を設定をします。
   %JUGGLING_PROJECT_PATH%/conf 配下に propagation-senders-config ディレクトリを作成します。
   propagation-senders-config 配下に im_master.xml を作成します。

   ファイルパス：
   %JUGGLING_PROJECT_PATH%/conf/propagation-senders-config/im_master.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<propagation-senders-config
 xmlns="http://www.intra-mart.jp/propagation/senders-config"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.intra-mart.jp/propagation/senders-config propagation-senders-config.xsd">

 <sender source="jp.co.intra_mart.foundation.master.user.model.User" operationType="DATA_CREATED">
  <encoder
class="jp.co.intra_mart.system.master.user.propagation.sender.logic.trigger.SynchronousUserListenerEncoder" />
 </sender>
 <sender source="jp.co.intra_mart.foundation.master.user.model.User" operationType="DATA_UPDATED">
  <encoder
class="jp.co.intra_mart.system.master.user.propagation.sender.logic.trigger.SynchronousUserListenerEncoder" />
 </sender>

 <sender source="jp.co.intra_mart.system.master.user.propagation.model.PropagationDepartment"
operationType="DATA_CREATED">
  <encoder
class="jp.co.intra_mart.system.master.user.propagation.sender.logic.trigger.SynchronousCompanyListenerEncoder"
/>
 </sender>
 <sender source="jp.co.intra_mart.system.master.user.propagation.model.PropagationDepartment"
operationType="DATA_UPDATED">
  <encoder
class="jp.co.intra_mart.system.master.user.propagation.sender.logic.trigger.SynchronousCompanyListenerEncoder"
/>
 </sender>
 <sender source="jp.co.intra_mart.system.master.user.propagation.model.PropagationDepartment"
operationType="DATA_DELETED">
  <encoder
class="jp.co.intra_mart.system.master.user.propagation.sender.logic.trigger.SynchronousCompanyListenerEncoder"
/>
 </sender>

 <sender source="jp.co.intra_mart.system.master.user.propagation.model.PropagationPublicGroup"
operationType="DATA_CREATED">
  <encoder
class="jp.co.intra_mart.system.master.user.propagation.sender.logic.trigger.SynchronousPublicGroupListenerEncoder"
/>
 </sender>
 <sender source="jp.co.intra_mart.system.master.user.propagation.model.PropagationPublicGroup"
operationType="DATA_UPDATED">
  <encoder
class="jp.co.intra_mart.system.master.user.propagation.sender.logic.trigger.SynchronousPublicGroupListenerEncoder"
/>
 </sender>
 <sender source="jp.co.intra_mart.system.master.user.propagation.model.PropagationPublicGroup"
operationType="DATA_DELETED">
  <encoder
class="jp.co.intra_mart.system.master.user.propagation.sender.logic.trigger.SynchronousPublicGroupListenerEncoder"
/>
 </sender>

</propagation-senders-config>
```

3. IM-共通マスタ のリスナを設定をします。
   %JUGGLING_PROJECT_PATH% 直下に plugin ディレクトリが存在しない場合は作成します。
   plugin 配下に使用するプラグインのディレクトリを作成します。

   - jp.co.intra_mart.master.accessor.user_8.0.19
   - jp.co.intra_mart.master.accessor.company_8.0.19
   - jp.co.intra_mart.master.accessor.public_group_8.0.19

   各ディレクトリの配下に plugin.xml を作成します。

ファイルパス：
%JUGGLING_PROJECT_PATH%/plugin/%PLUGIN_ID%/plugin.xml

- jp.co.intra_mart.master.accessor.user_8.0.19 の plugin.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin>
  <extension point="jp.co.intra_mart.foundation.master.accessor.user" >
    <accessor
      name="synchronous"
      id="jp.co.intra_mart.foundation.master.accessor.user"
      version="8.0.19"
      rank="1" >
      <listener class="jp.co.intra_mart.system.master.user.impl.SynchronousUserListener" />
    </accessor>
  </extension>
</plugin>
```

- jp.co.intra_mart.master.accessor.company_8.0.19 の plugin.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin>
  <extension point="jp.co.intra_mart.foundation.master.accessor.company" >
    <accessor
      name="synchronous"
      id="jp.co.intra_mart.foundation.master.accessor.company"
      version="8.0.19"
      rank="1" >
      <listener class="jp.co.intra_mart.system.master.company.impl.SynchronousCompanyListener" />
    </accessor>
  </extension>
</plugin>
```

- jp.co.intra_mart.master.accessor.public_group_8.0.19 の plugin.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin>
  <extension point="jp.co.intra_mart.foundation.master.accessor.public_group" >
    <accessor
      name="synchronous"
      id="jp.co.intra_mart.foundation.master.accessor.public_group"
      version="8.0.19"
      rank="1" >
      <listener class="jp.co.intra_mart.system.master.public_group.impl.SynchronousPublicGroupListener"
/>
    </accessor>
  </extension>
</plugin>
```