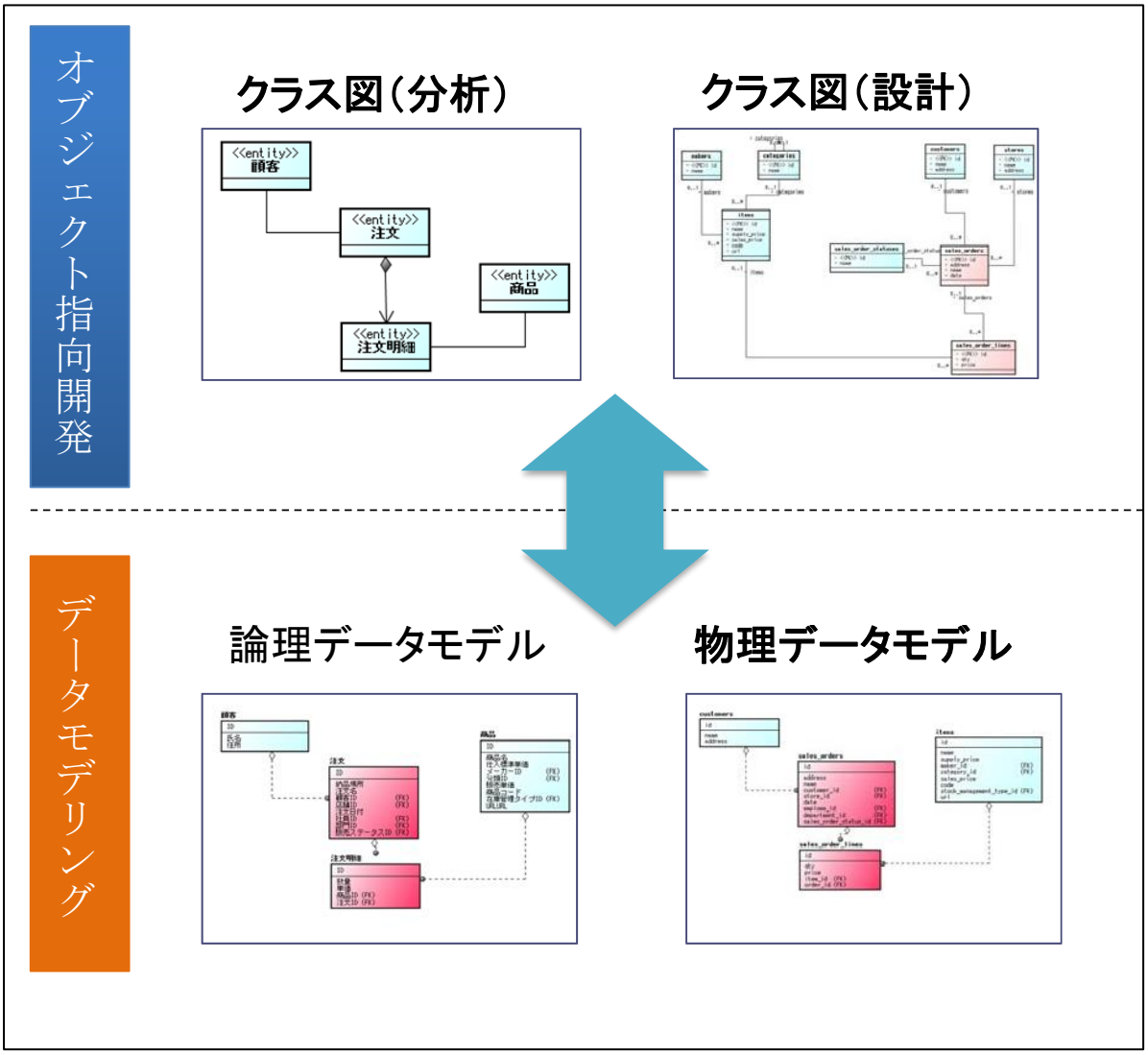
データモデリングしよう

データモデリングしよう

オブジェクト指向分析設計がかなり現場でも使われるようになってきましたが、それでも、データモデリングは重要です。

オブジェクト指向はプログラミング言語から来た概念を設計に援用しようとしています。データモデリングは、アプリケーションを超えた寿命を持つデータをモデリングしようとしています。

両者は、補い合ってひとつの「意味論」の中で協調して使われても「よい」ものなのです。オブジェクト指向至上主義でも、データモデリング至上主義でも、現実の開発はうまく行きません。オブジェクト指向言語とリレーショナルデータベースをつなぐのは、DAO や Hibernate のようなパターンやツールを使うこともありますが、逆に、これまで蓄えられた業務モデリングの知見を利用しながらオブジェクト指向設計をすることも可能です。

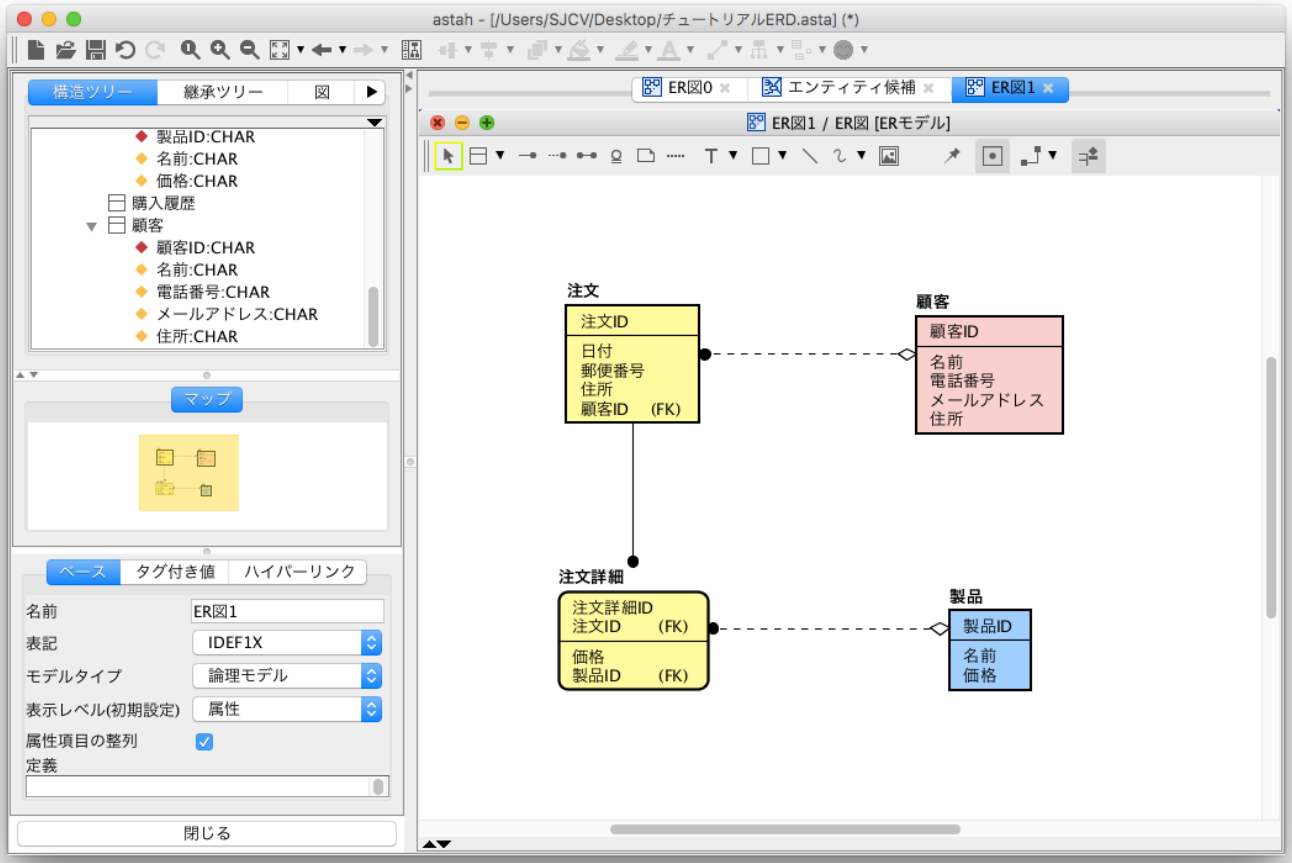


ER 図を使ってみよう

使用できる製品: astah* professional

デモ動画 : https://www.youtube.com/watch?v=lKfuAutufZA&feature=youtu.be

ER 図のIDEF1X とIE の両記法をサポート。「リソース」、「イベント」、「サマリー」のエンティティカテゴリや、階層的な「ドメイン」の活用（エンティティにドラッグ &ドロップできます）、論理・物理名の交換、SQL 出力などに対応しています。マインドマップ、UML 図要素、データフロー図要素からER エンティティへの変換、又その逆の連携も実現しています。



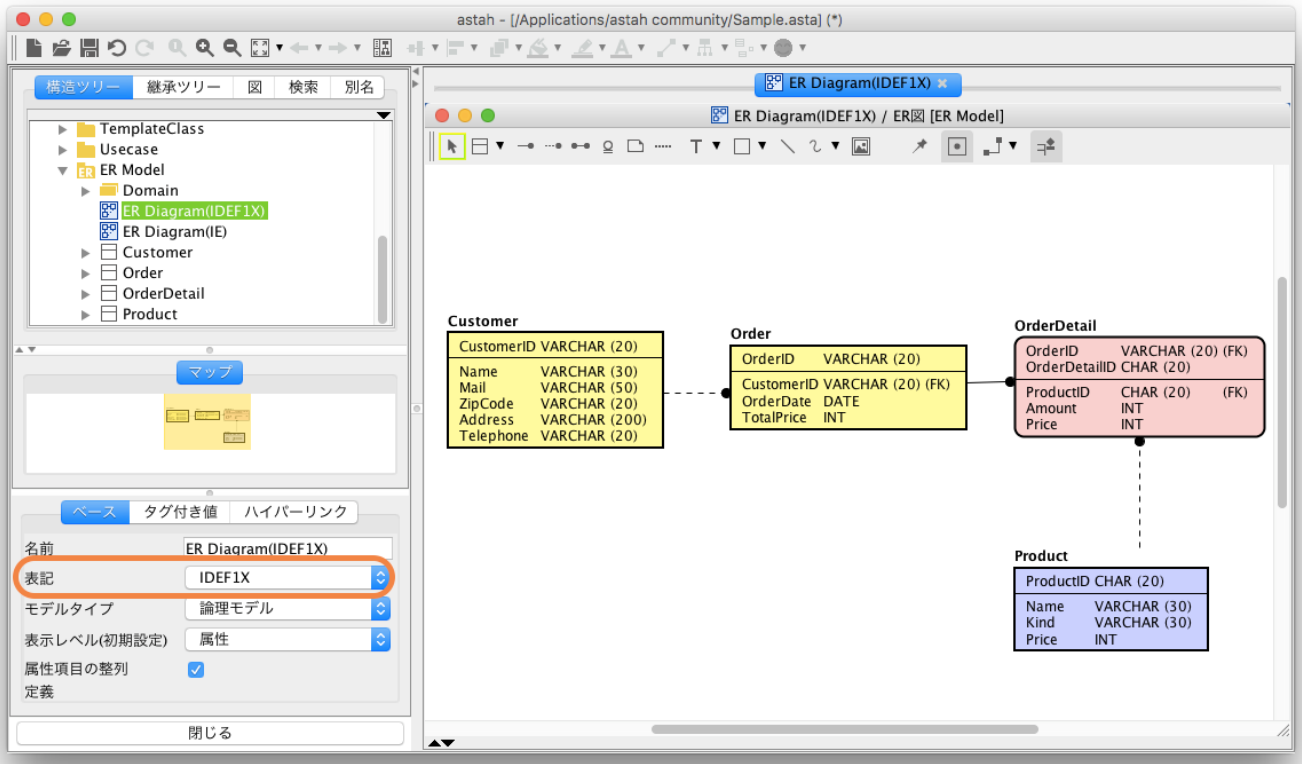
astah*の ER 図の主な機能は以下の通りです。

- IDEF1X、IE によるER 図の作成
- エンティティ定義書出力
- DB リバース（プラグイン対応、サポート対象外）
- マインドマップからエンティティへの変換
- SQL 出力（SQL-92 準拠）
- ドメインの対応
- 表示レベル設定
- 物理名の対応

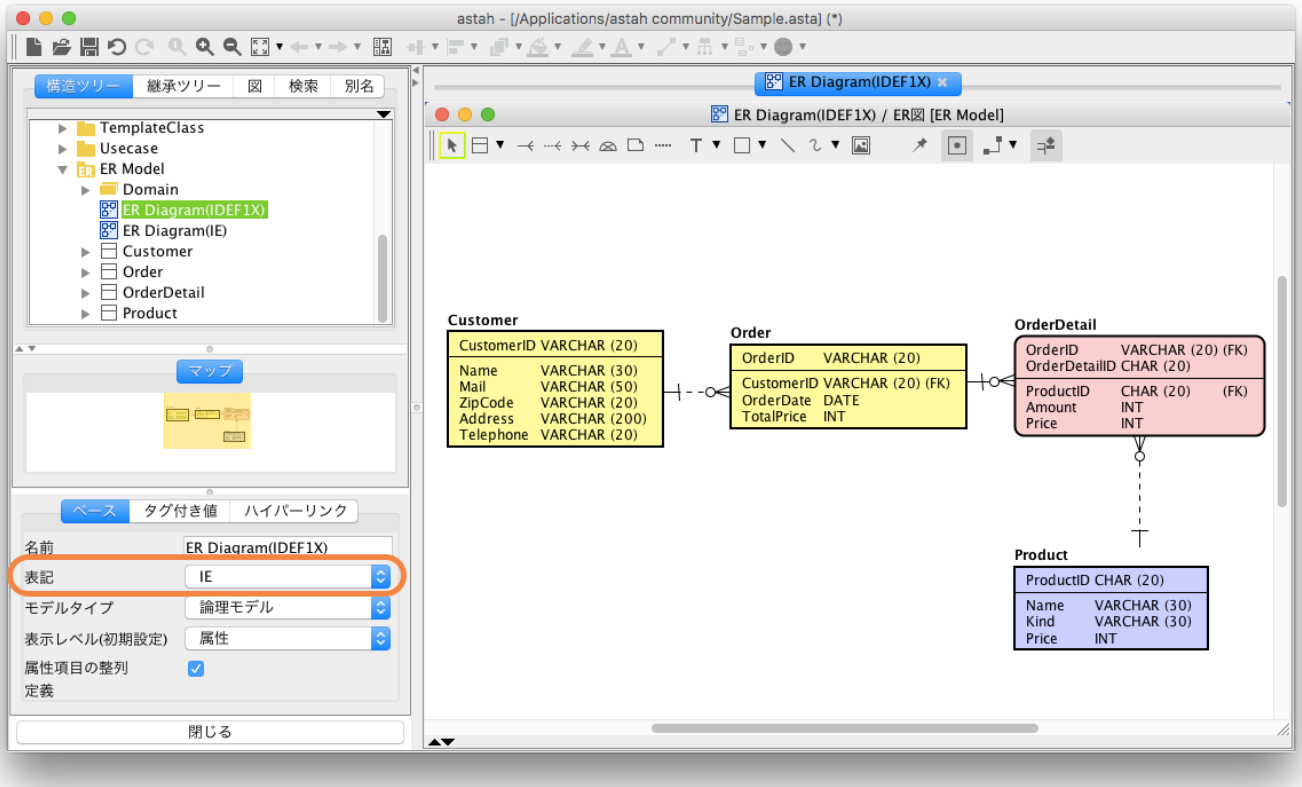
IDEF1X、IE による ER 図を作成しよう

astah*のER 図は 2 つの表記法に対応しており、すぐに切り替えられます。

[IDEF1X 表記]



[IE 表記]



エンティティの種類について考えてみよう

エンティティとは、システム(データベース)の管理対象となりうるもので人、モノ、イベント、履歴などがあります。
エンティティには、リソース系、イベント系、サマリー系などがあり、以下のように分類されます。

エンティティ種別	主な例
リソース系	会社、顧客、商品、製品、工場、カネなど。
イベント	会員登録・変更・削除、発注、出荷など。
サマリー系	購入履歴、出荷履歴など。

エンティティの抽出のコツがあり、以下の手順を踏むのもよいでしょう。

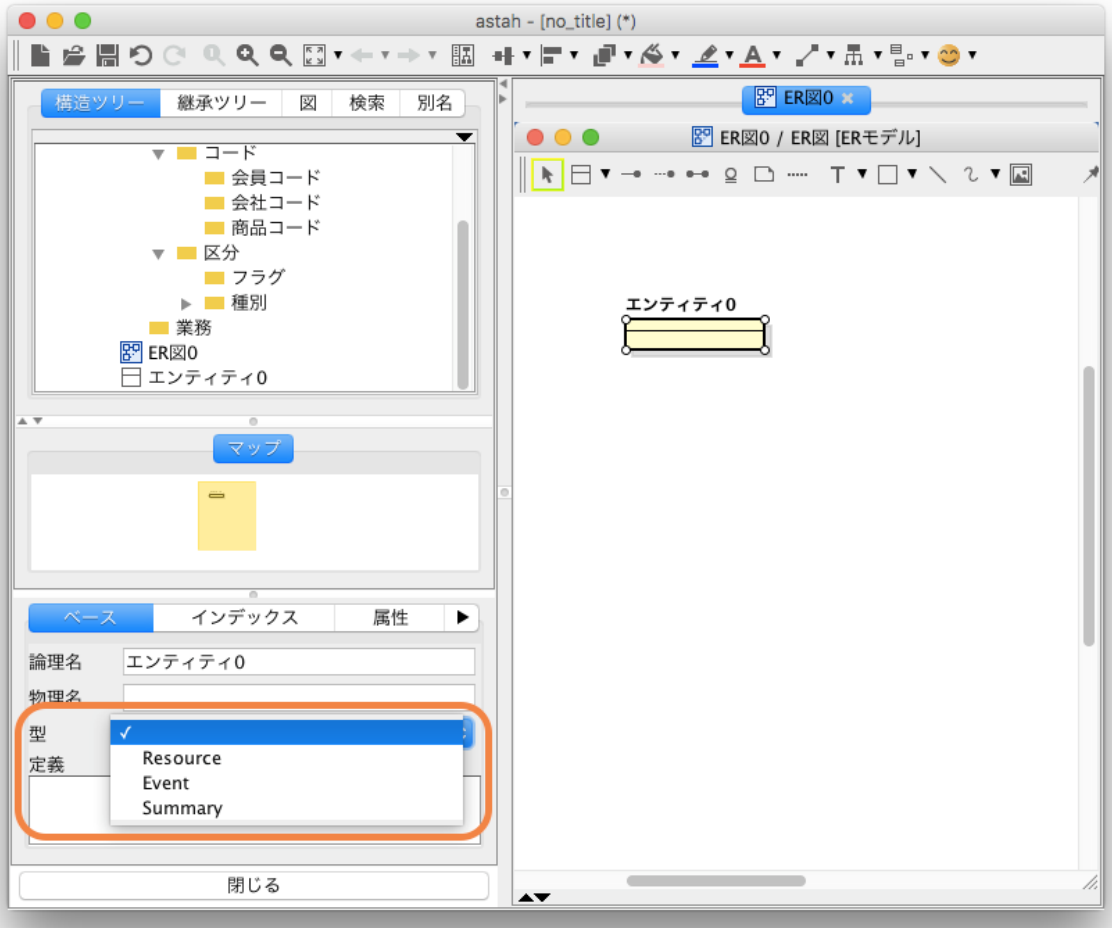
(1) エンティティ候補をヒト、モノなどのカテゴリーで抽出(マインドマップで行うのもよい)

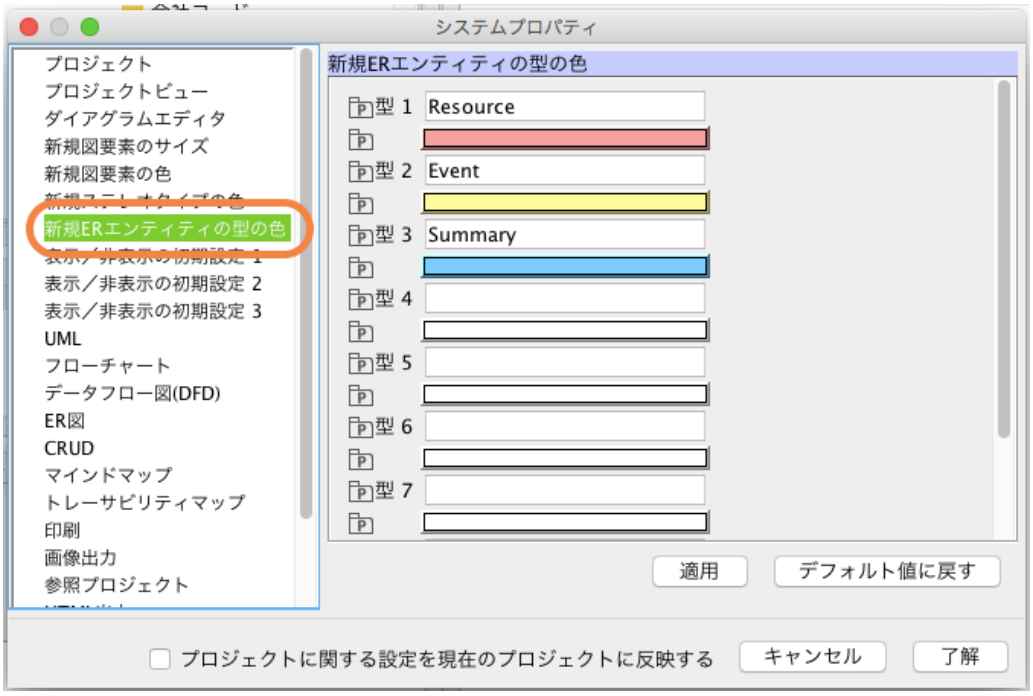
・そこから、リソース系エンティティの抽出

(2)業務フローを書く。(フローチャートやDFD、アクティビティ図で行うのもよい)

・そこから、イベント系エンティティの抽出

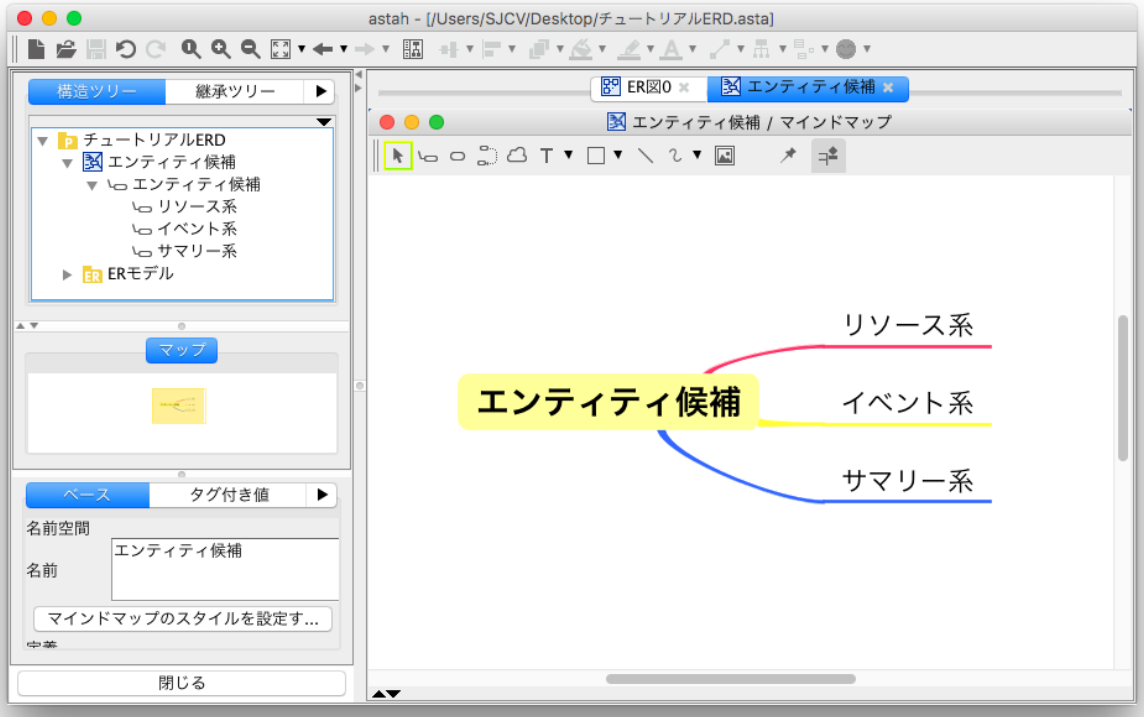
また、astah*ではエンティティのプロパティビューから、リソース系、イベント系、サマリー系の指定や、[システムプロパティ]-[新規 ER エンティティの型の色]で系統ごとに色を指定できるので、ER 図を書いてモデリングするときに便利です。



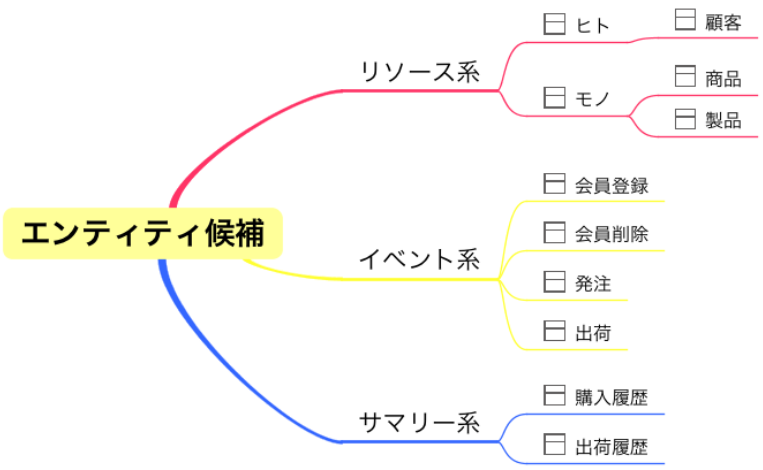


マインドマップからエンティティに変換しよう

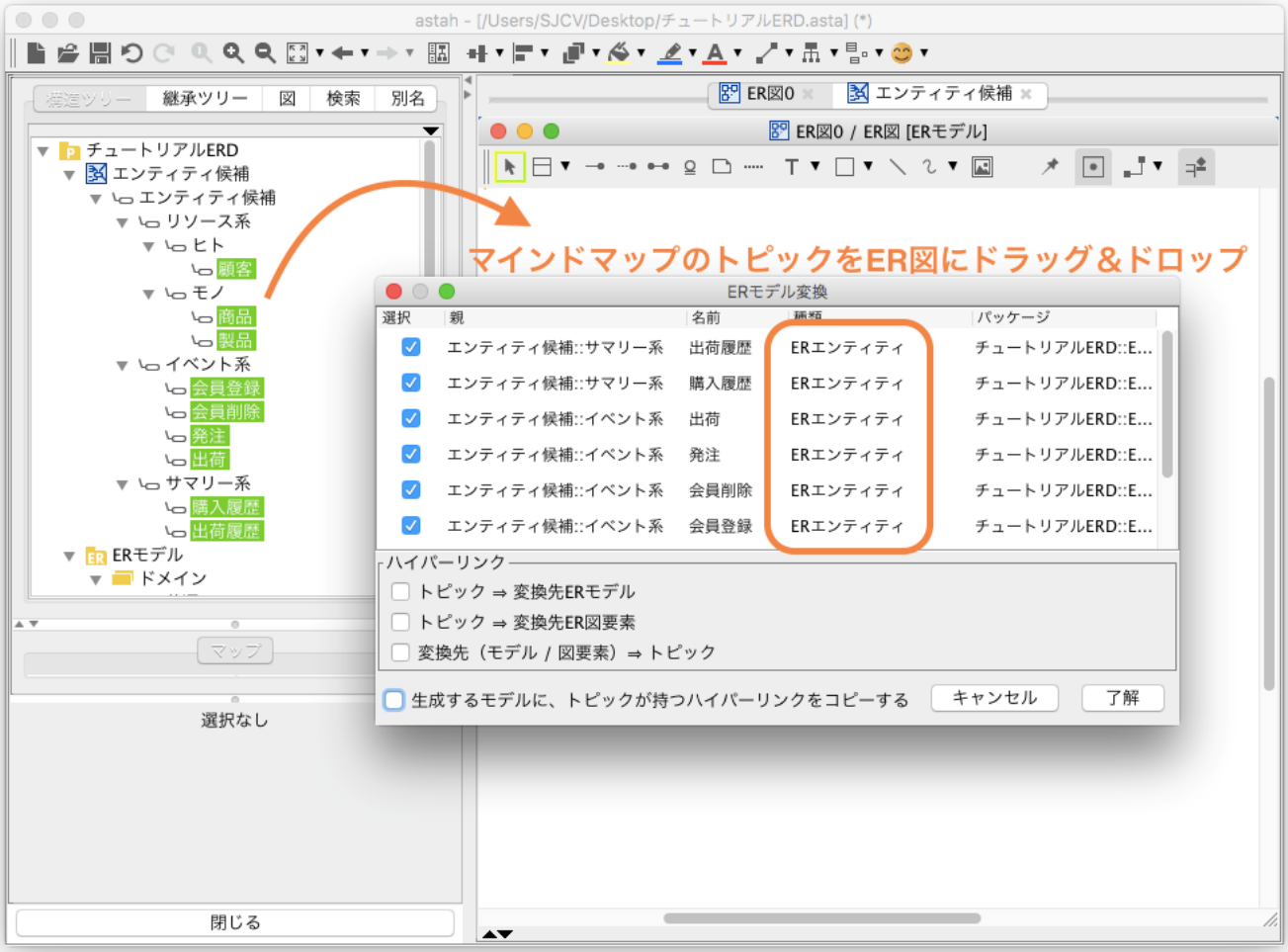
マインドマップでエンティティを抽出するために、以下のようなテンプレートを用意するのもよいでしょう。



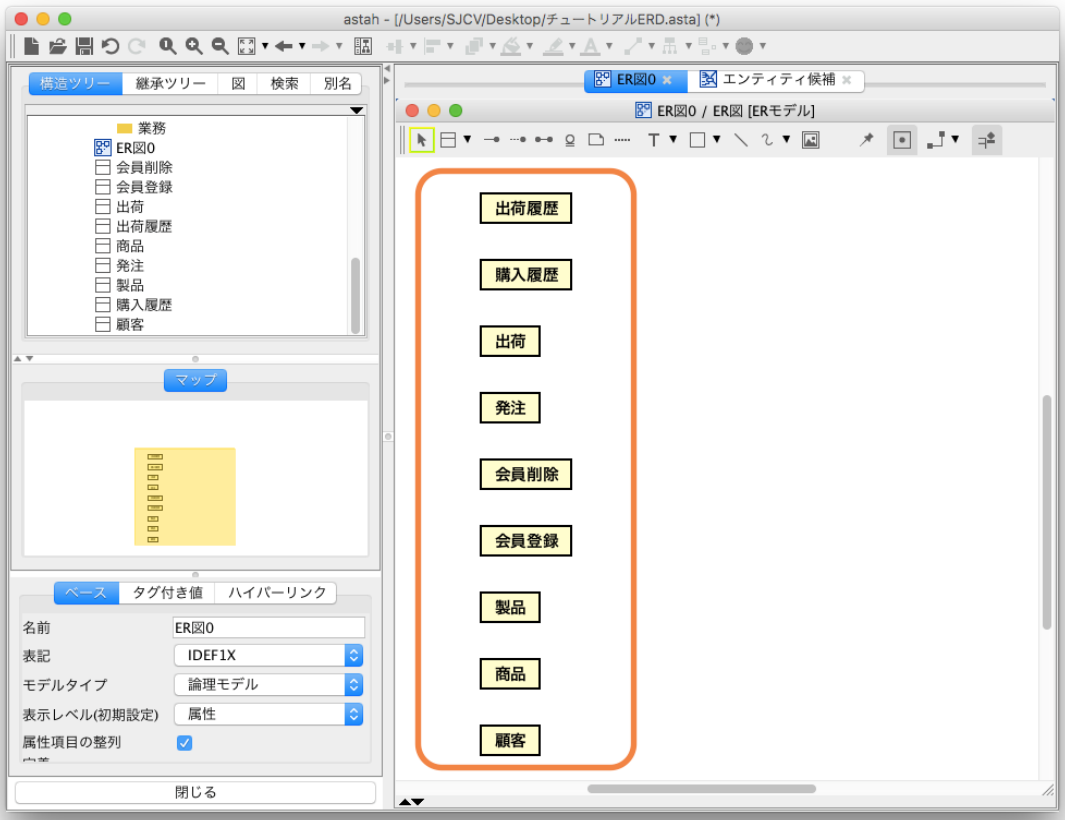
ショッピングサイト構築という想定でエンティティを挙げてみました。また、エンティティ候補には、ER エンティティのアイコンも付加してみました。



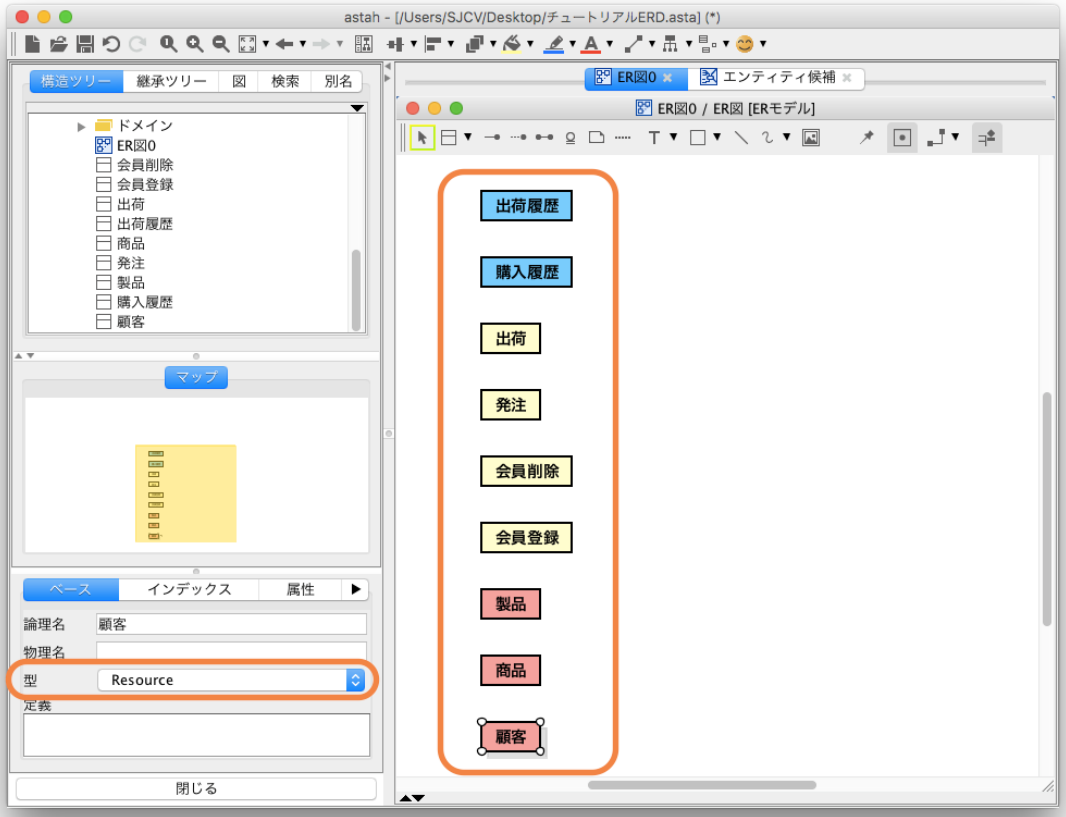
さて、新規に ER 図を作成し、このマインドマップのトピックを、構造ツリーから ER 図にドラッグ&ドロップします。事前に図の表示レベルをエンティティにしておくとよいでしょう。(詳しくは次の章で説明します。)



以下のようにER エンティティが作成されました。



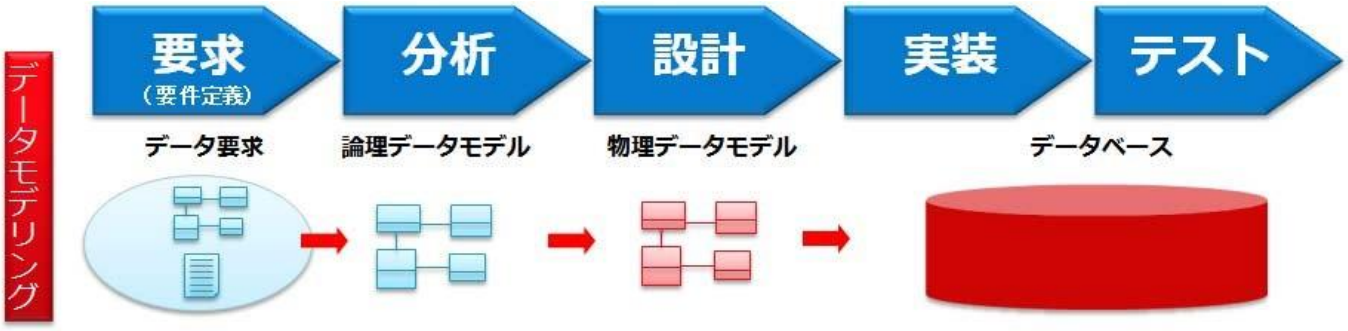
次にリソース系、イベント系、サマリー系を設定してみます。



システムプロパティの色が設定されていることがわかりますね。色付けすることにより、コミュニケーションも容易になり、より分かりやすくなります。あとは、続けて属性やリレーションシップの設計などを行い、設計を詰めていくとよいでしょう。

データモデリングのプロセスを使用してみよう

データモデリングのプロセスの流れは、次の図のように、分析、設計、実装フェーズで、論理データモデルから物理データモデルを作成するのが主流です。

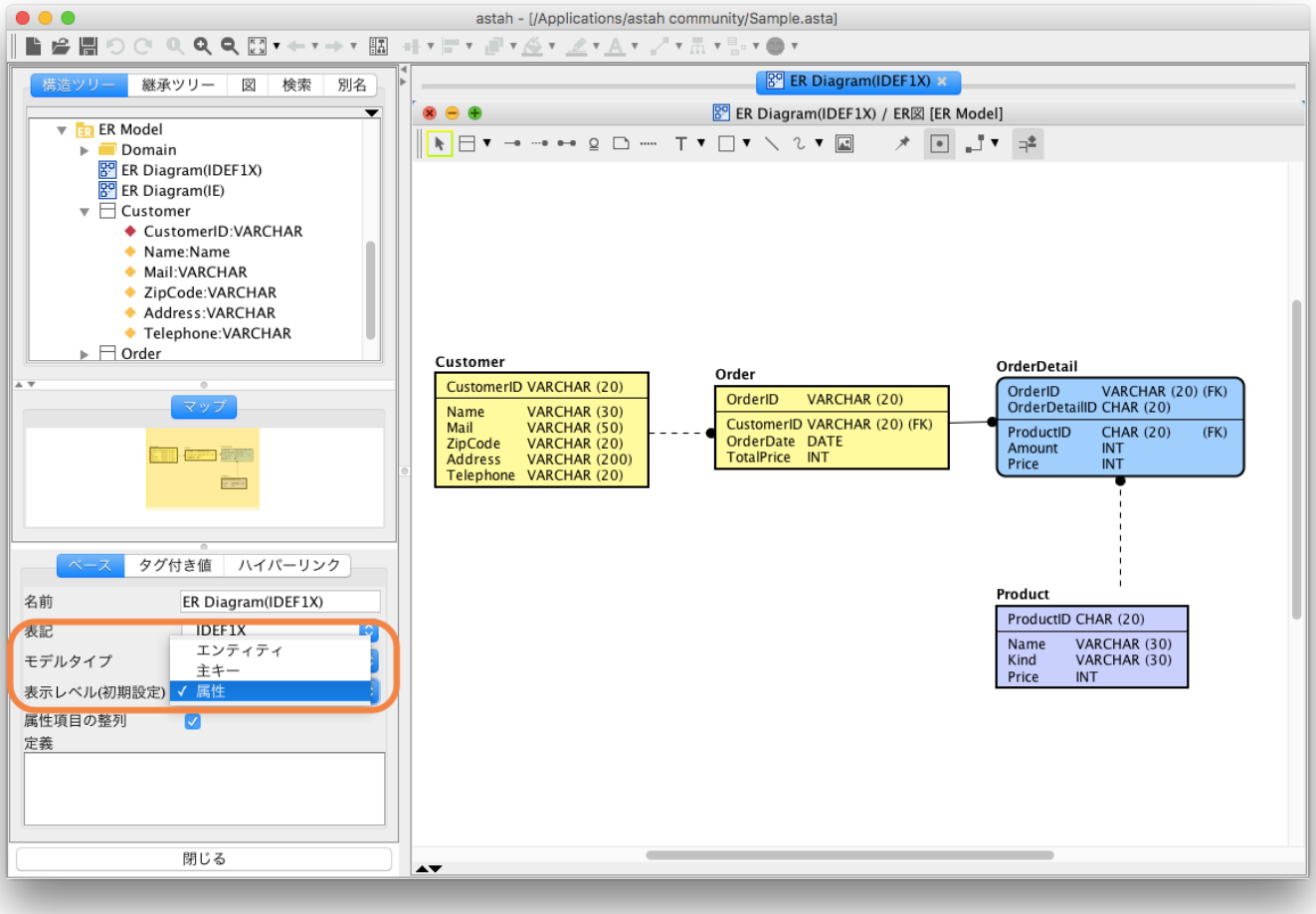


出典:アーキテクタス 細川努「UML による一気通貫DB システム設計」

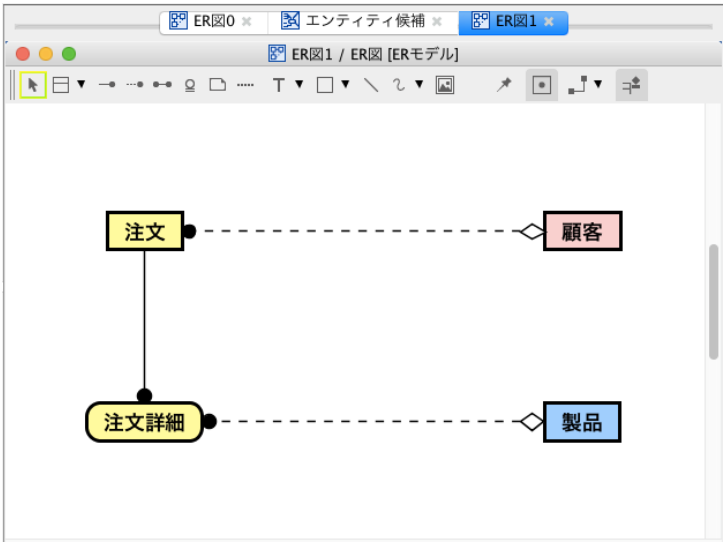
エンティティの表示レベルを設定してみよう

astah*では、図の表示レベル(エンティティ、主キー、属性)を設定でき、これらのプロセスに則ったデータモデリングが可能です。論理よりではエンティティレベルで、物理モデルに近づくにつれて主キーレベル、属性レベルで設計するとよいでしょう。操作方法は、ER 図のプロパティビューから表示レベル(初期設定)で変更します。

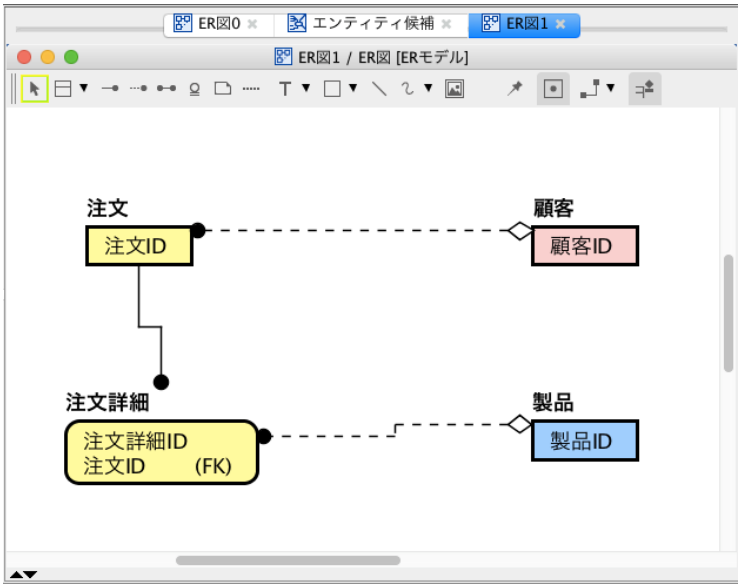
ちなみに、この設定は、設定後にER 図で新規作成したER エンティティのみ反映されます。既存のER エンティ ティの表示レベルを変更したい場合は、ER エンティティのポップアップメニューから変更してください。



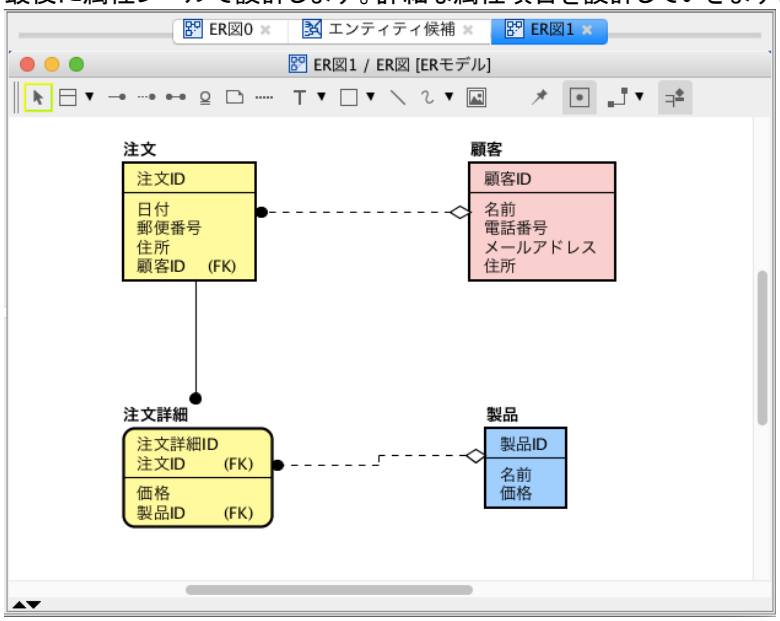
まず、エンティティレベルで設計してみます。エンティティとなりうるモデルを抽出するにとどめておきます。



続いて主キーレベルで設計します。リレーションを張っていってよいでしょう。



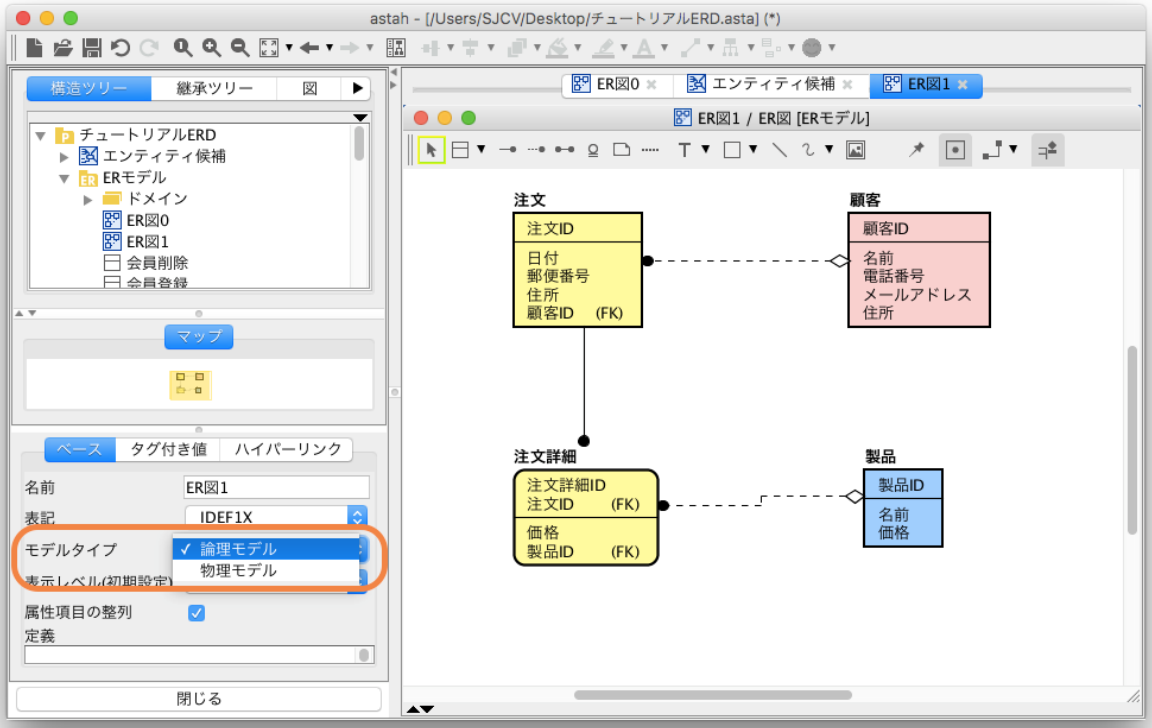
最後に属性レベルで設計します。詳細な属性項目を設計していきます。



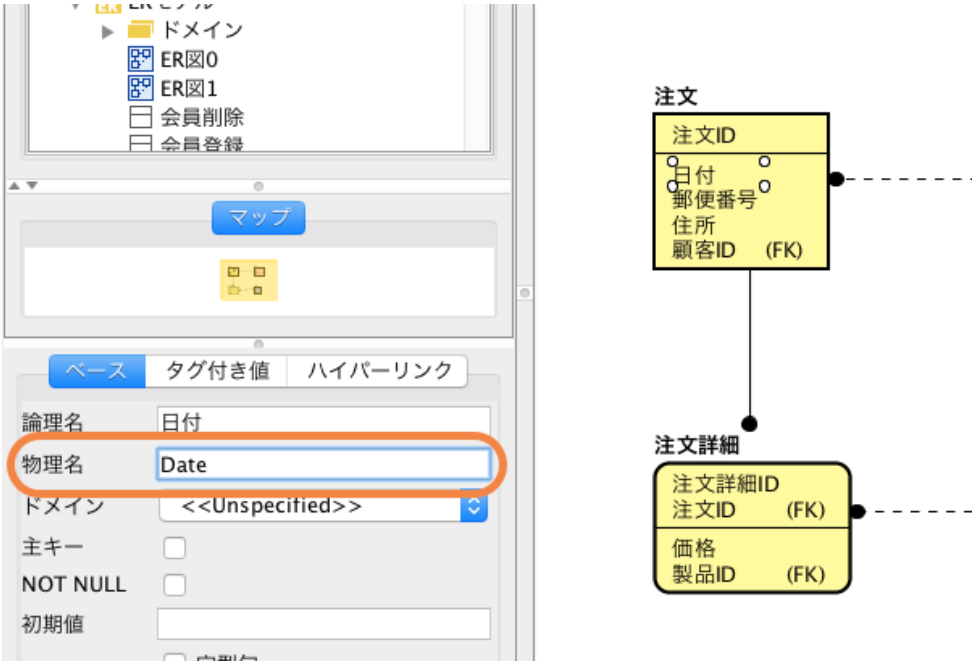
論理名と物理名を使ってみよう

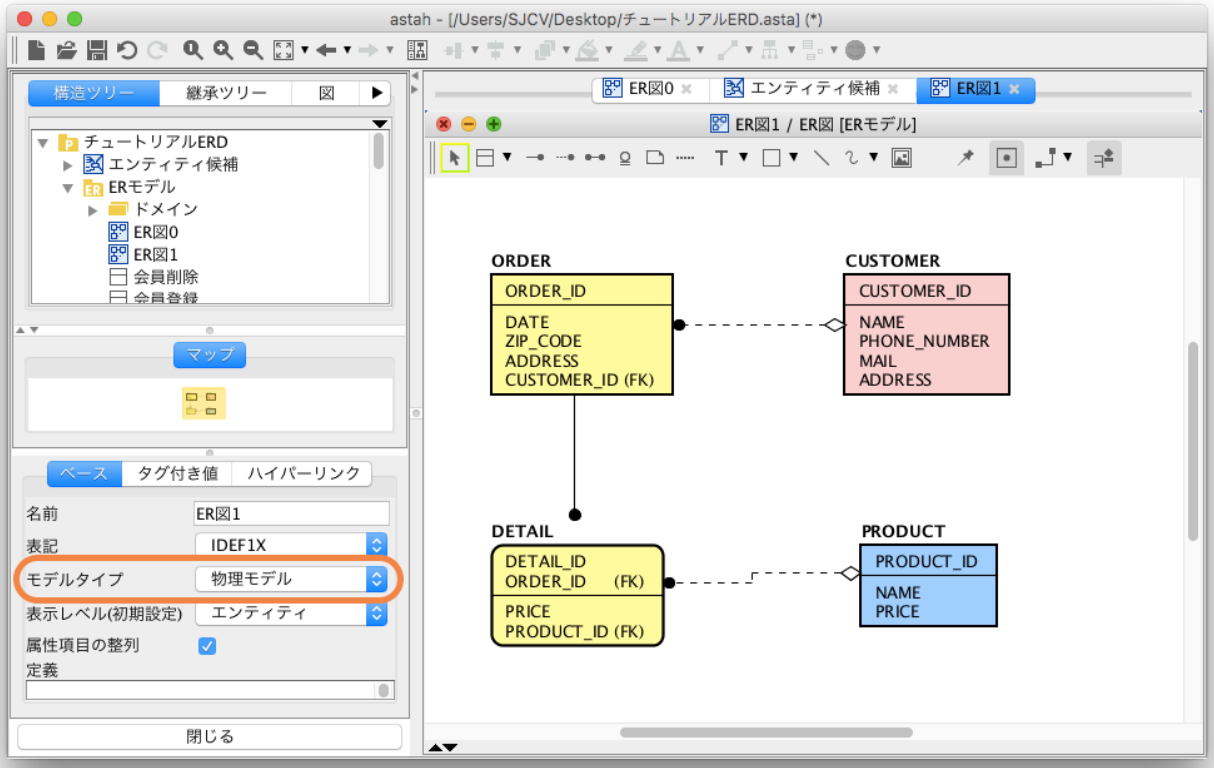
astah*では、エンティティや属性などに論理名と物理名の両方を設定でき、データモデリングのプロセスをサポート します。論理データモデリングでは、エンティティの論理名に”顧客”とつけ、物理データモデリングではテーブル 命名規則に則り、物理名を”CUSTOMER”にするようなことができます。

エンティティだけでなく属性にも物理名を設定できますし、SQL 出力では、論理名、物理名での出力が可能です。論理モデル、物理モデルの切り替えは、ER 図のプロパティビューのモデルタイプから行います。



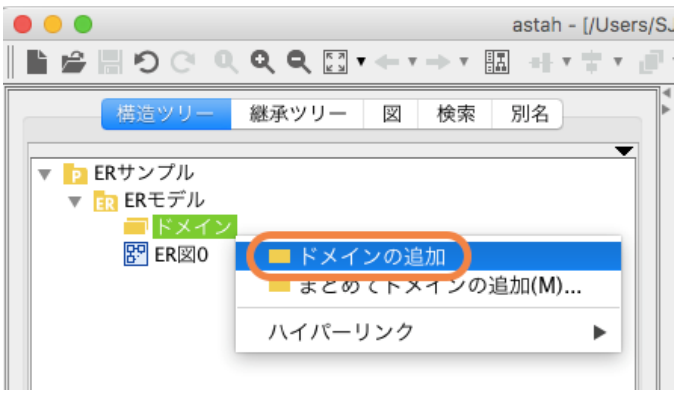
まず、[論理モデル]で設計します。各エンティティや属性、リレーションについて概念を詰めていきます。
次に[物理モデル]で設計します。データベース自体を想定しながらモデリングしていきます。





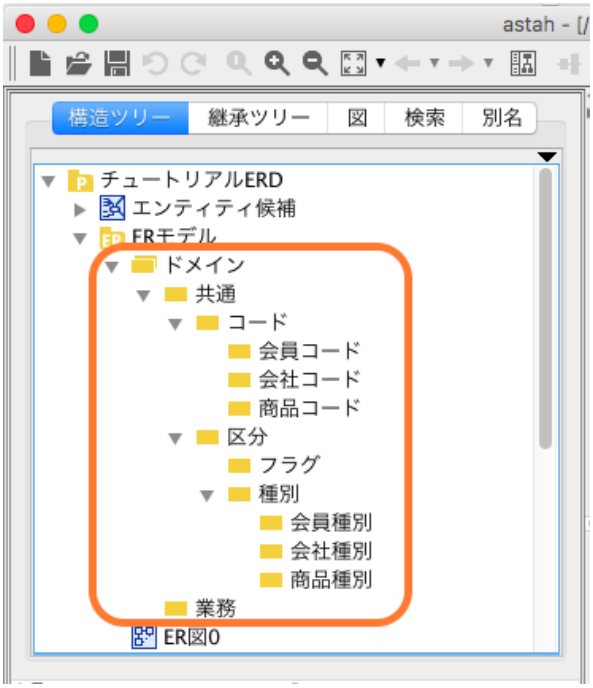
ドメインを活用しよう

ドメインとは、データの型や値/範囲を取り決めるディクショナリのことで、複数のエンティティの属性から設定できます。ドメインを使用したデータモデリングはシステムの内のデータの不統一や不整合をなくすことができ、データモデリングの中でも重要なプロセスです。例えば、クレジットカード番号にハイフンを含めるか含めないかで、CHAR(12)に設計する人やCHAR(15)で設計する人が、ドメインを使用することで、このようなテーブル設計のミスをなくすことが可能です。また、ドメインを使用していることで、後から属性の長さを変更しても、参照先のエンティティに反映されるなど、仕様変更に強い設計になります。astah*では階層的なドメインの機能も兼ね備えています。構造ツリーのドメインのポップアップメニューから追加できます。階層的に作成した場合、親の型や長さ等を引き継ぎますが、子側で型や長さの変更も可能です。

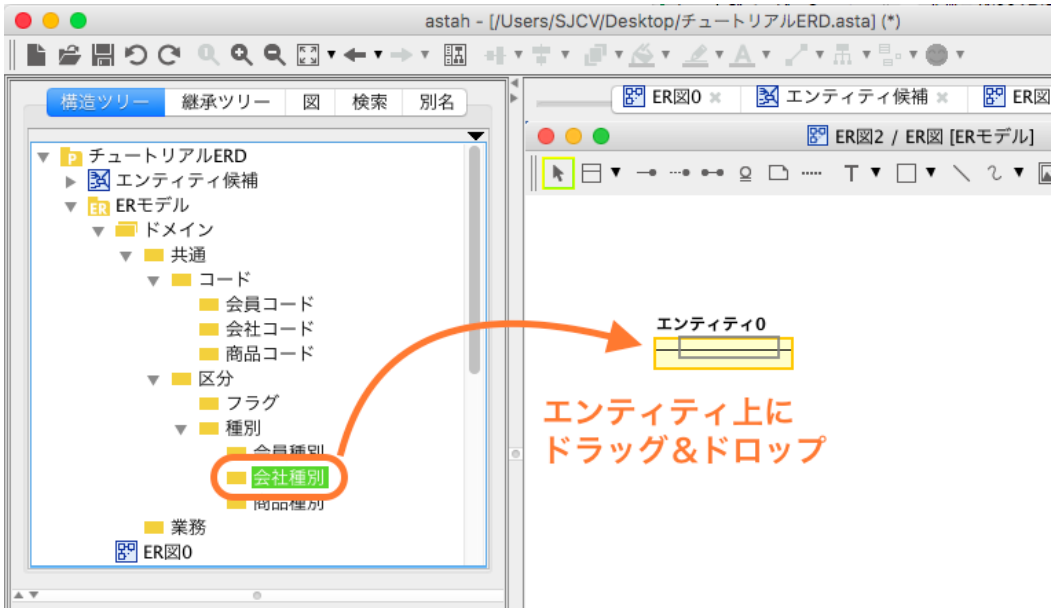


ドメインのグルーピングの仕方は、その設計者の自由に取り決めていいですが、業務に依存しない共通的なドメイ ンは”共通”、業務に依存するドメインは”固有”や”業務”としておくのもいいでしょう。

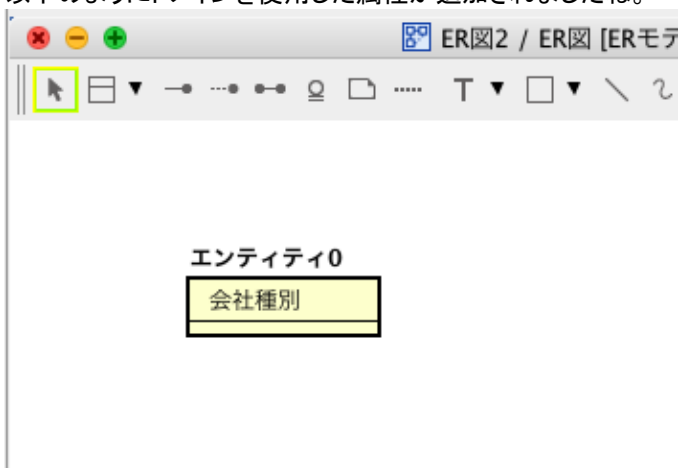
下の例は、”共通”ドメインでいくつかドメインを設計した結果です。



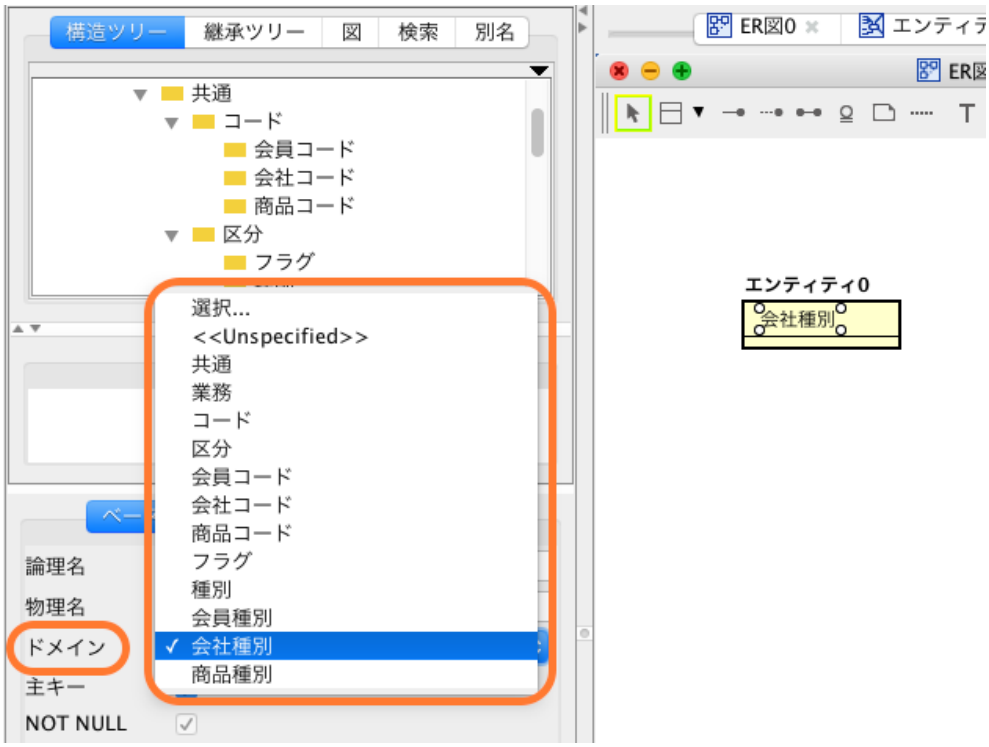
また、作成したドメインをER エンティティにドラッグ&ドロップすることもできます。



以下のようにドメインを使用した属性が追加されましたね。



また、属性のプロパティビューからドメインを変更することもできます。



リレーションを張ってみよう

以下のようなモデルを作成してみましょう。

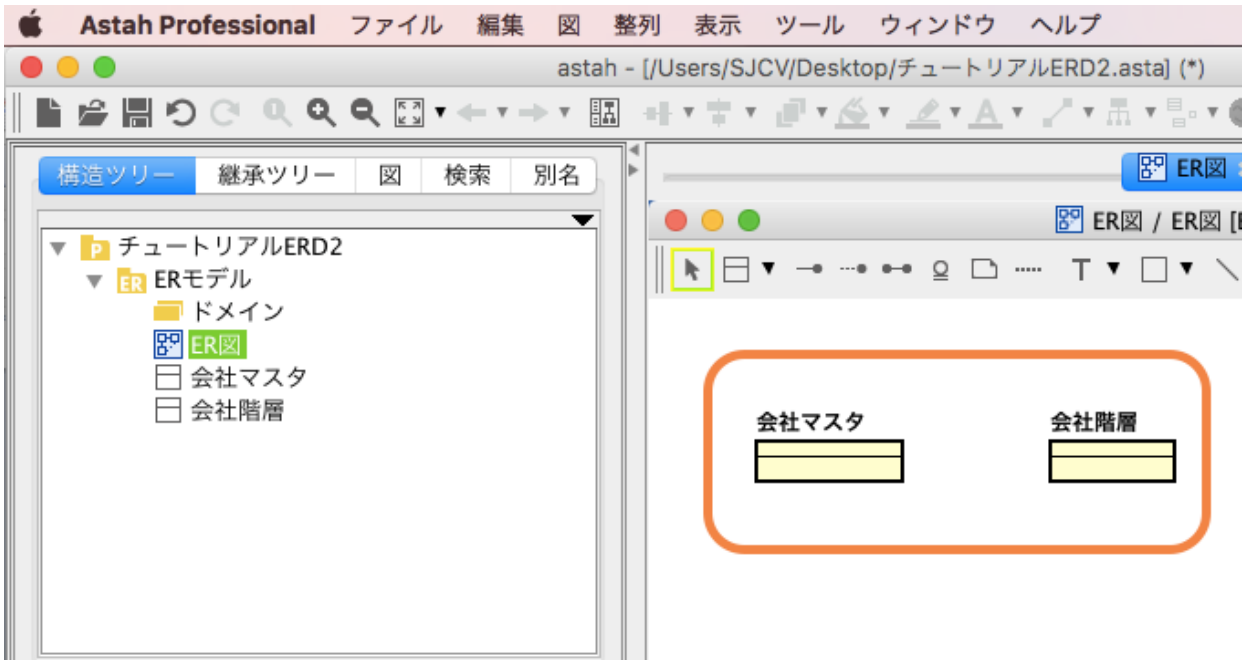
（例1）

【会社マスタ】　　【会社階層】会社コード

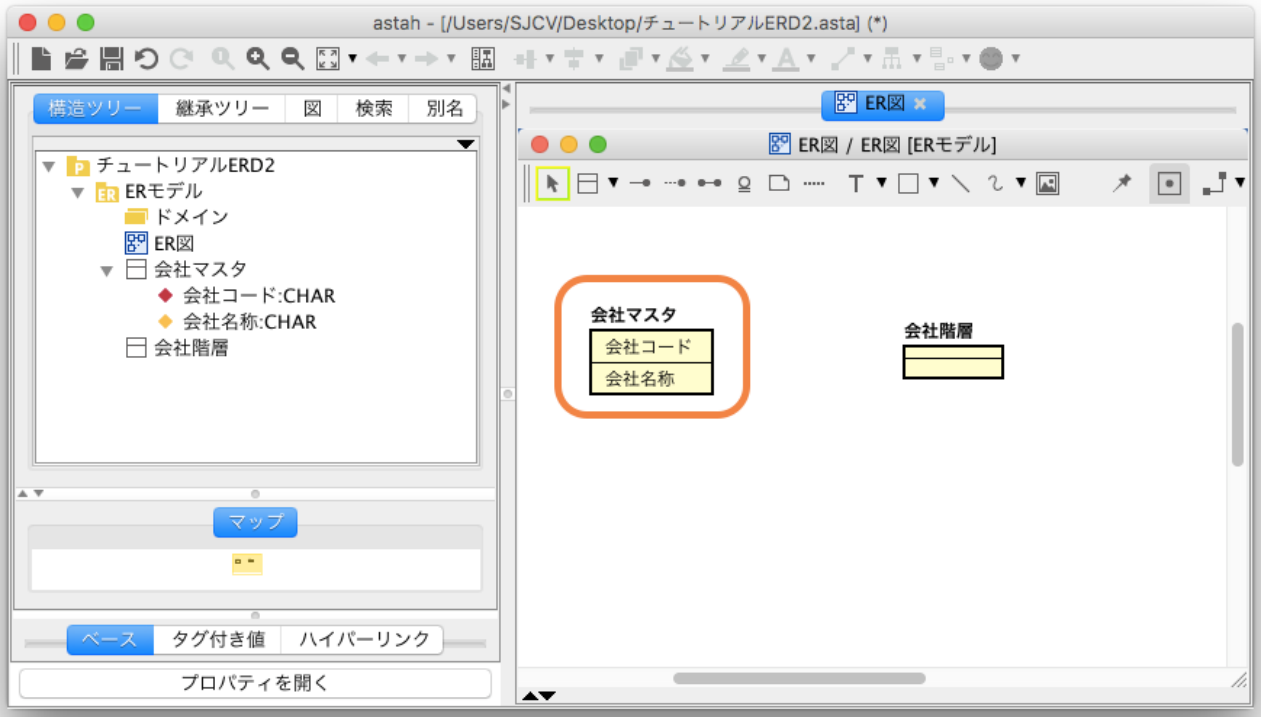
（PK）　　上位会社コード(FK)

会社名称　　下位会社コード(FK)

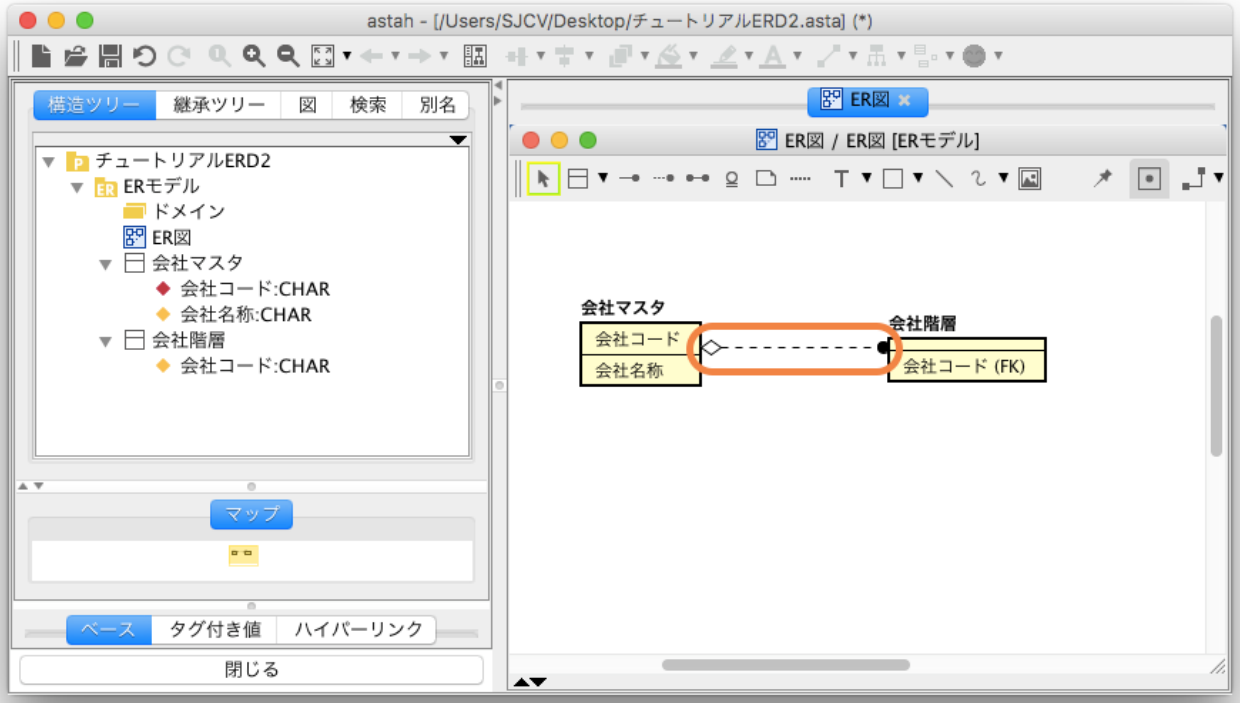
まず、2つの ER エンティティ –”会社マスタ”と”会社階層”を作成します。



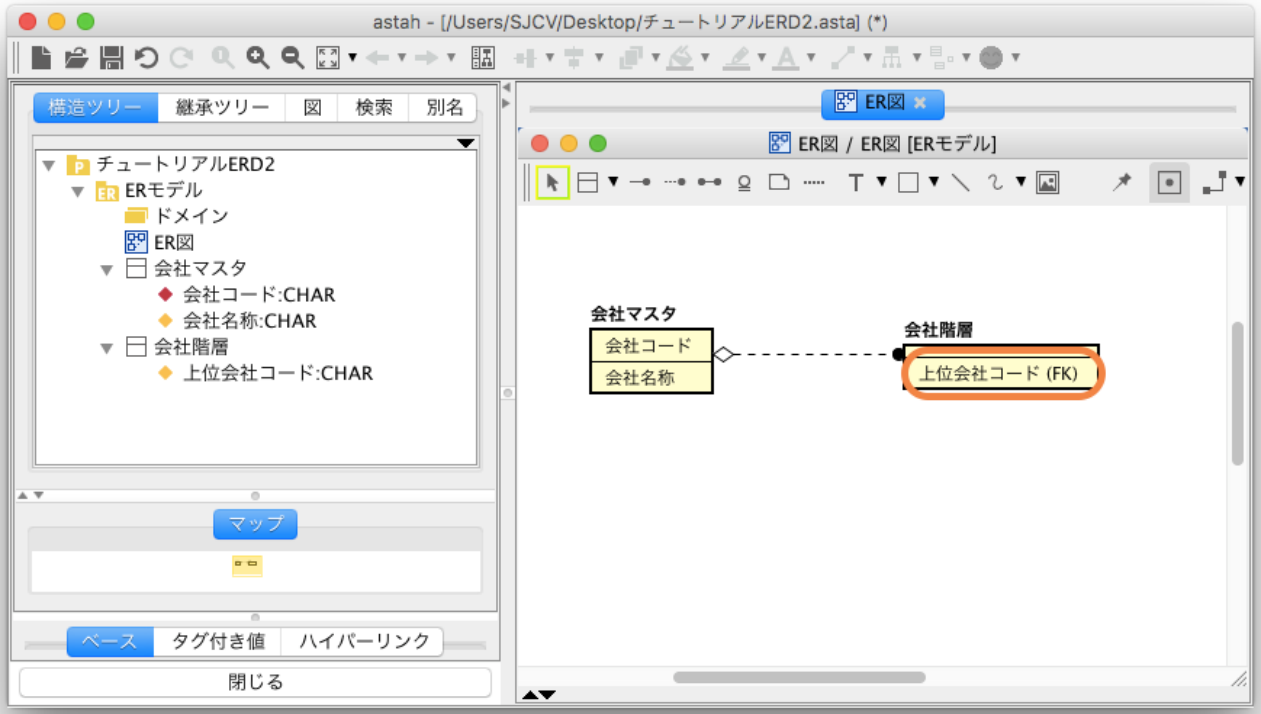
ER エンティティ“会社マスタ”に、主キー”会社コード”、属性”会社名称”を追加します。



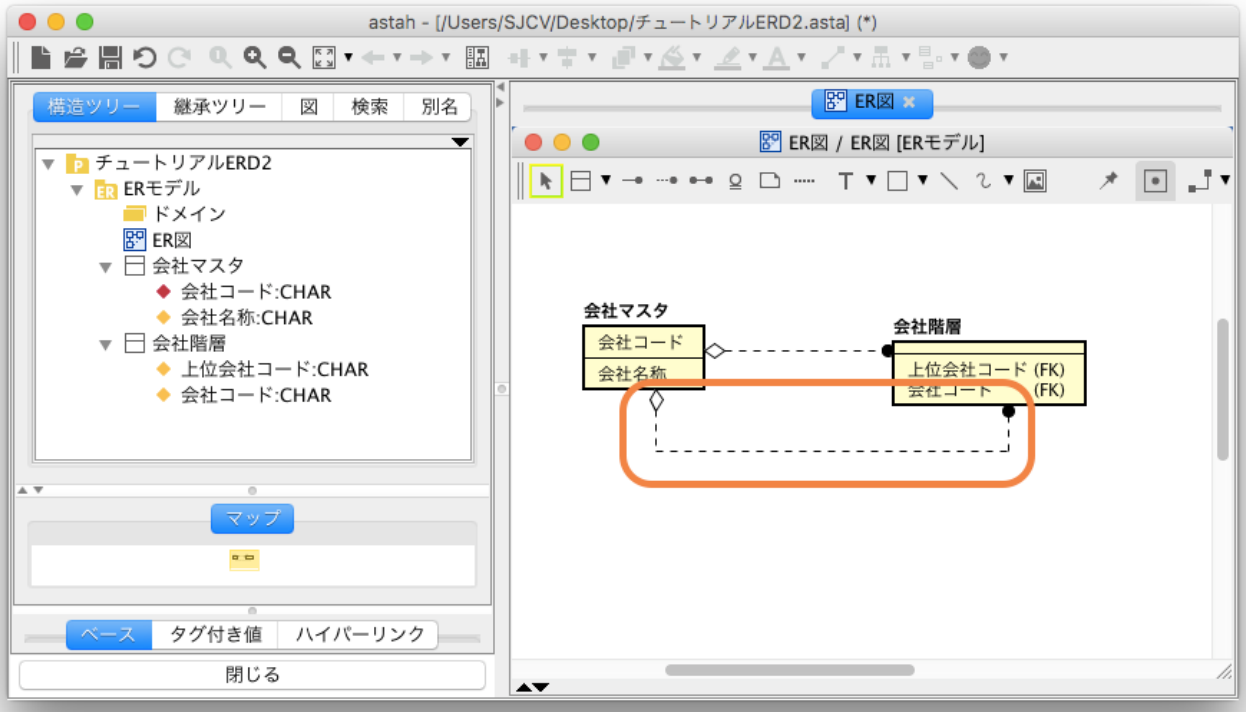
ER エンティティ“会社マスタ”から、ER エンティティ”会社階層”に非依存型リレーションシップを引きます。



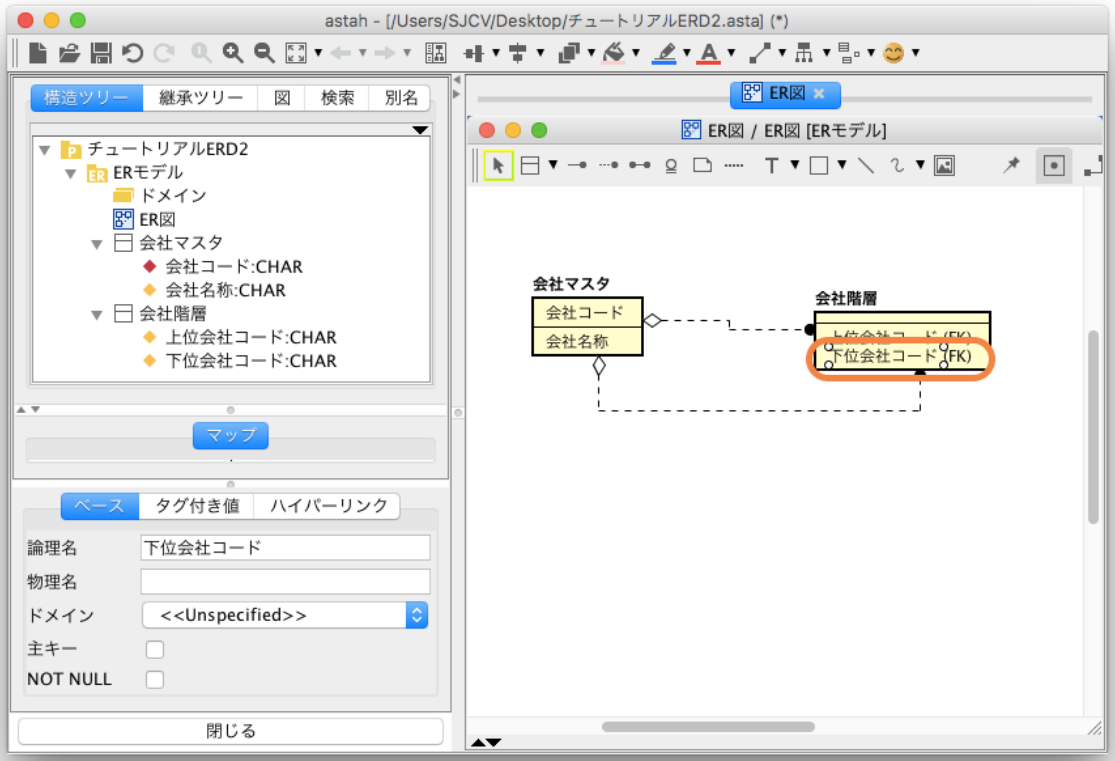
ER エンティティ”会社階層“の属性”会社コード“を、”上位会社コード”にリネームします。



再度、ER エンティティ“会社マスタ”から ER エンティティ”会社階層”に非依存型リレーションシップを引きます。



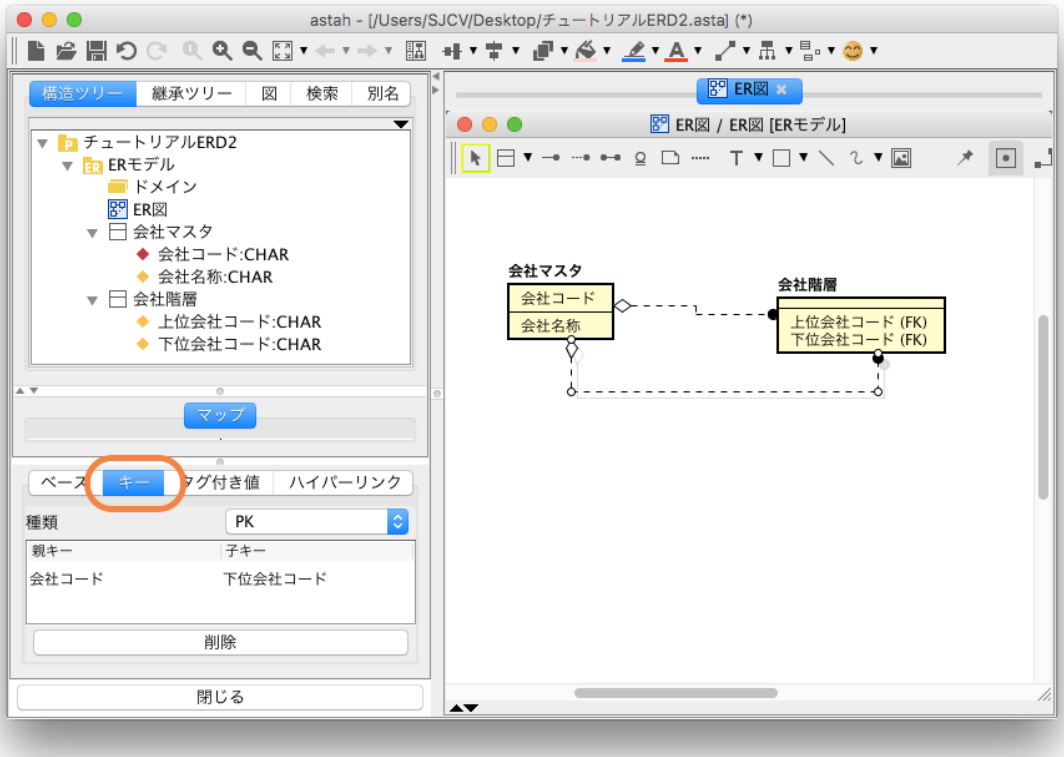
[会社コード(FK)]が追加されるので、[会社下位コード]にリネームします。



これでリレーションも含めたモデルを作成できましたね。

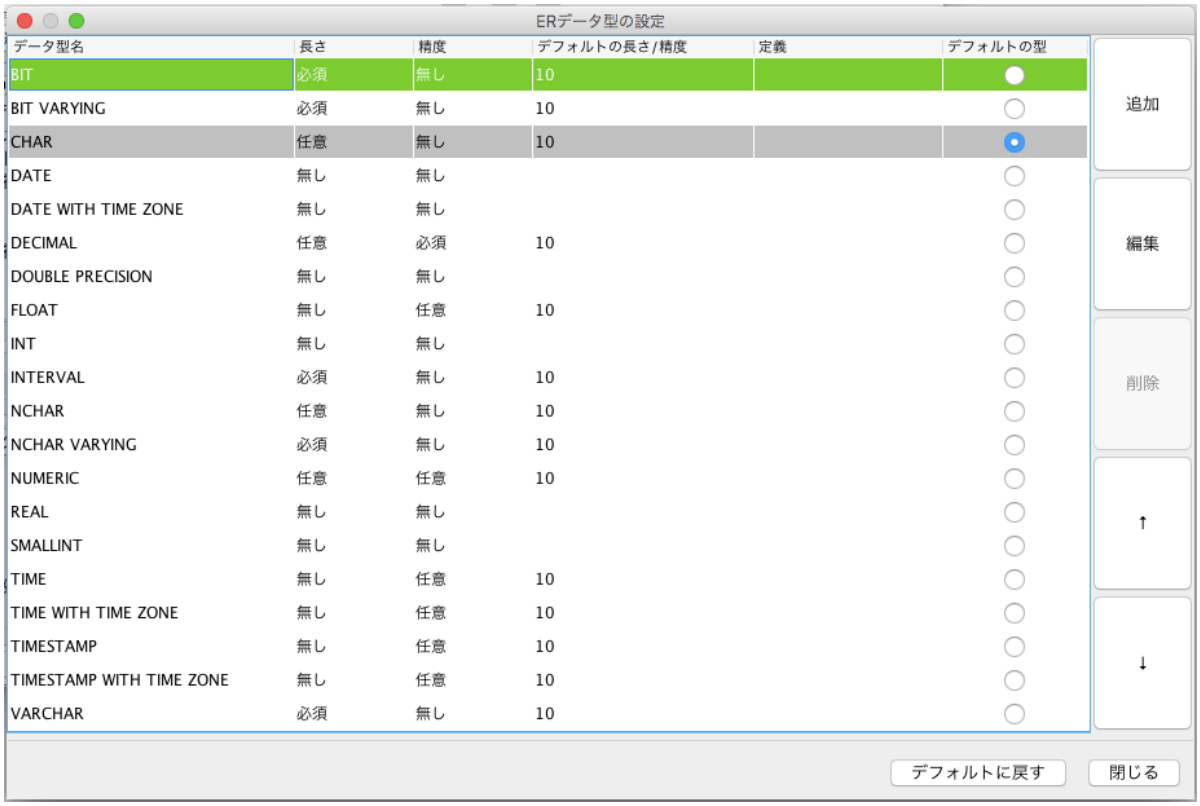
[リレーションシップ作成時のポイント]

親子間で複数のリレーションシップを張り、子エンティティ側のFKを異なる属性にしたい場合は、リレーションシップの[キー]タブからも可能です。

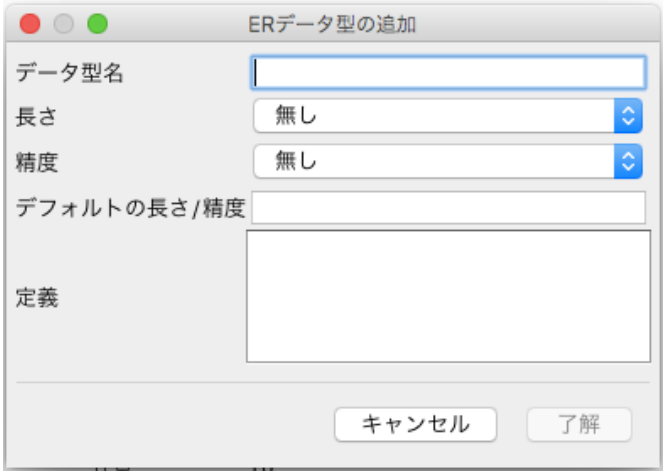


データ型を追加しよう

astah*では、デフォルトで以下のデータ型が定義されています。([ツール] – [ER 図] – [ER データ型の設定])



この一覧にないデータ型は[追加]ボタンを押す事で、設定できます。長さや精度も設定できます。



[物理モデル作成時のポイント]

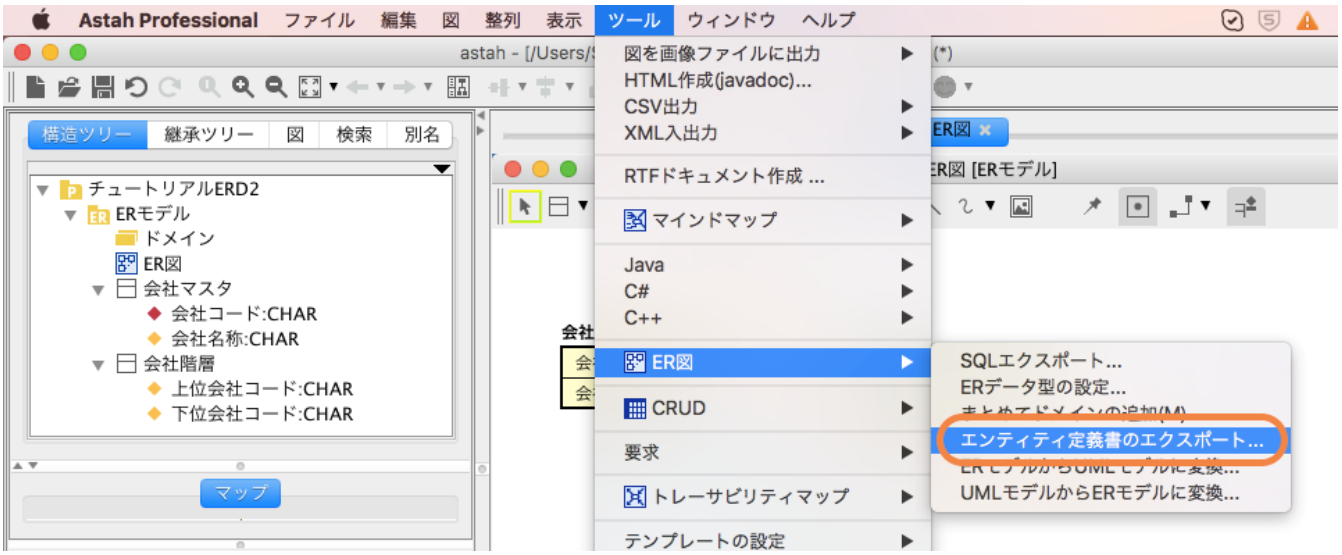
論理モデル作成後、物理モデル作成前に、固有のDBを使用し、astah*デフォルトのデータ型で不十分な場合、先にデータ型を設定しておくと、物理モデルの属性の型を考慮するときにスムーズにモデリングできます。

エンティティ定義書を出力しよう

デモ動画：https://www.youtube.com/watch?v=vAaaj3ud0l0&feature=youtu.be

astah*では、テンプレートを指定して、ER 図のエンティティ定義を EXCEL 形式で出力できます。設計したエンティティをドキュメントに出力して、納品資料として利用したり、チームでレビューしたりするのもよいでしょう。

[ツール]-[ER 図]-[エンティティ定義書のエクスポート]を選択します

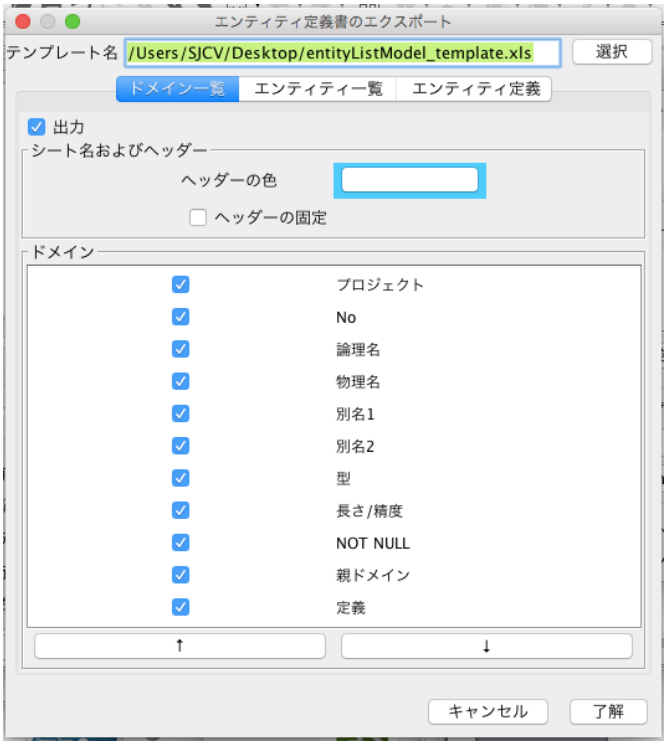


出力ダイアログが表示されます。テンプレートはプロジェクト固有のテンプレートにも変更できます。

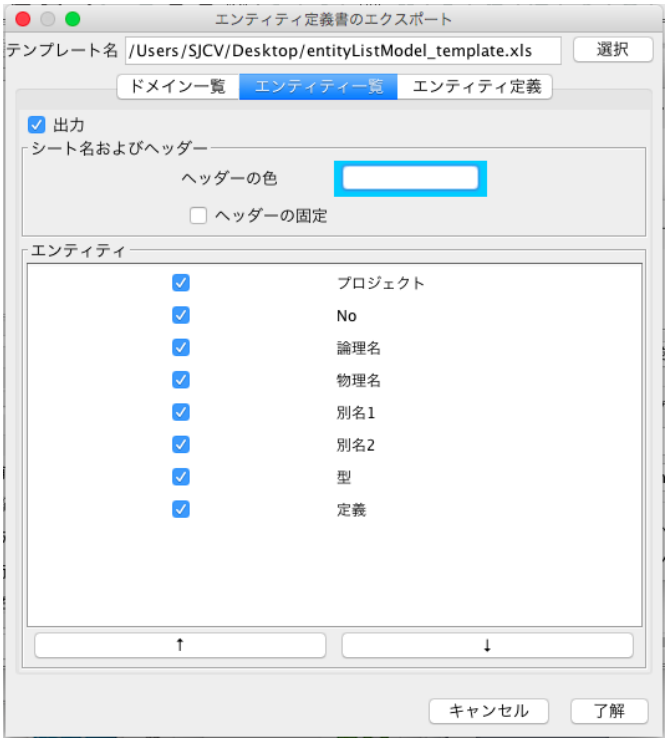
新しいテンプレートボタンをクリックしてみましょう。

以下のようにテンプレートを編集できます。

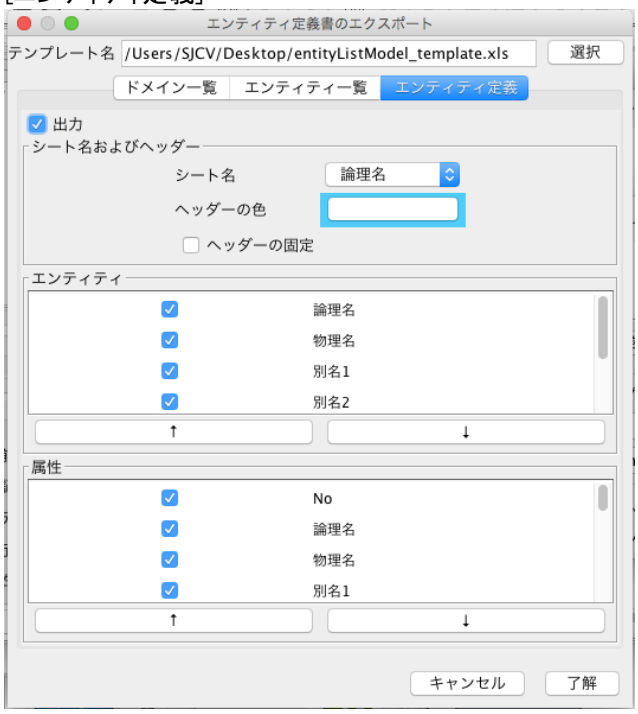
[ドメイン一覧]



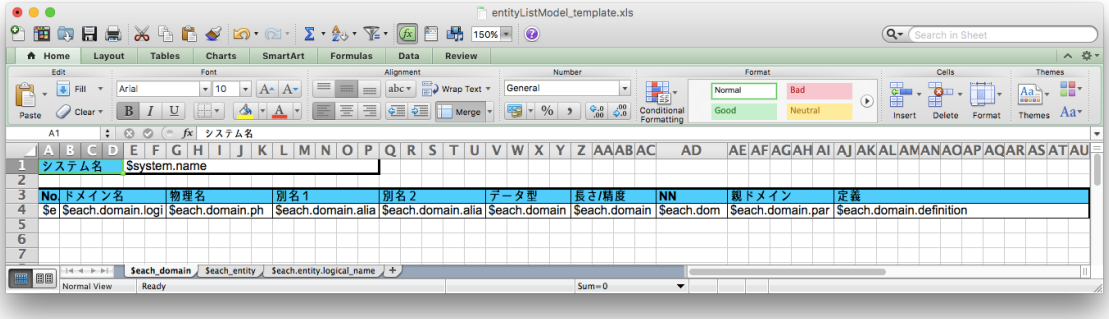
[エンティティー覧]



[エンティティ定義]



新しいテンプレートボタンをクリック後、了解ボタンクリックでテンプレートが作成されます。そのファイルを開いてみましょう。



$each_domain など$から始まるものは、次のようにマッピングされます。

全体	
タブのドメイン一覧	– $each_domain
タブのエンティティー一覧	– $each_entity
タブのエンティティ名	– $each.entity.logical_name
システム名	– $system.name
No.	– $each.row_number

ドメイン	
名前	– $each.domain.logical_name
物理名	– $each.domain.physical_name
別名１	– $each.domain.alias1
別名２	– $each.domain.alias2
データ型	– $each.domain.type
ドメインの長さ/精度	– $each.domain.length_precision
NotNull フラグ	– $each.domain.notnull
親ドメイン	– $each.domain.parent
定義	– $each.domain.definition

エンティティ	
論理名	– $entity.logical_name
物理名	– $entity.physical_name
別名１	– $entity.alias1
別名２	– $entity.alias2
型	– $entity.kind
定義	– $entity.definition

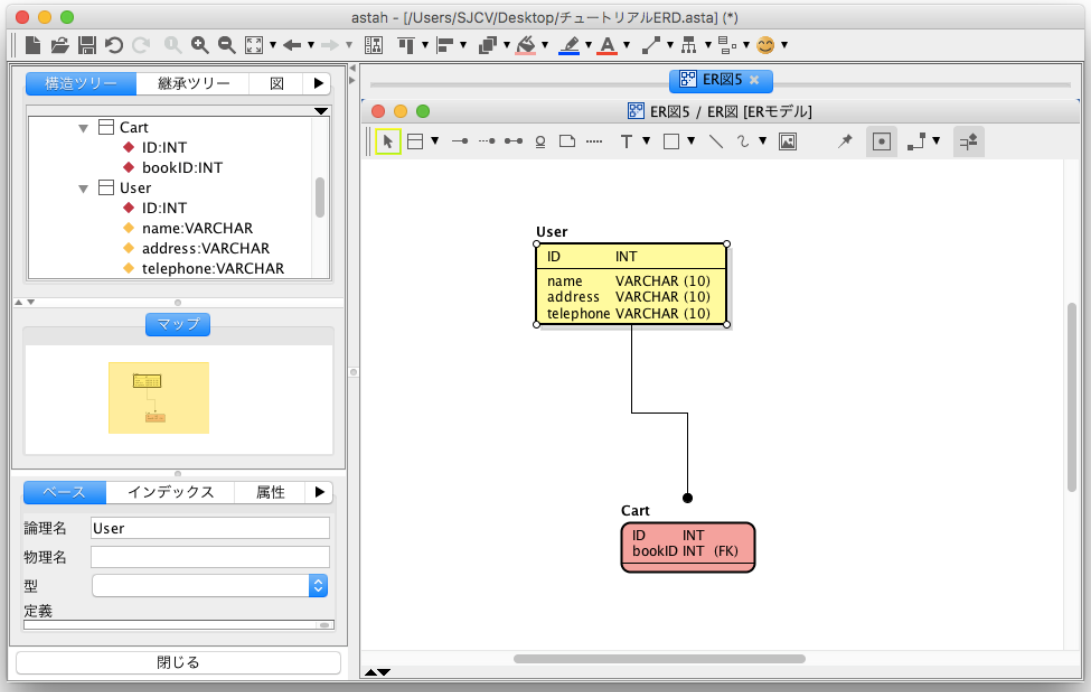
属性	
論理名	– $each.entity.each.attribute.logical_name
物理名	– $each.entity.each.attribute.physical_name
ドメイン名	– $each.entity.each.attribute.domain
主キーフラグ	– $each.entity.each.attribute.pk
外部キーフラグ	– $each.entity.each.attribute.fk
NotNull フラグ	– $each.entity.each.attribute.notnull
参照先	– $each.entity.each.attribute.ref
データ型	– $each.entity.each.attribute.type
長さ/精度	– $each.entity.each.attribute.length_precision
初期値	– $each.entity.each.attribute.initial_value
定義	– $each.attribute.definition
タグ付き値	– $each.entity.each.attribute.each.taggedvalue

出力した EXCEL は、プラグインを利用してインポートすることも可能です。
EXCEL-ER モデルインポートプラグイン

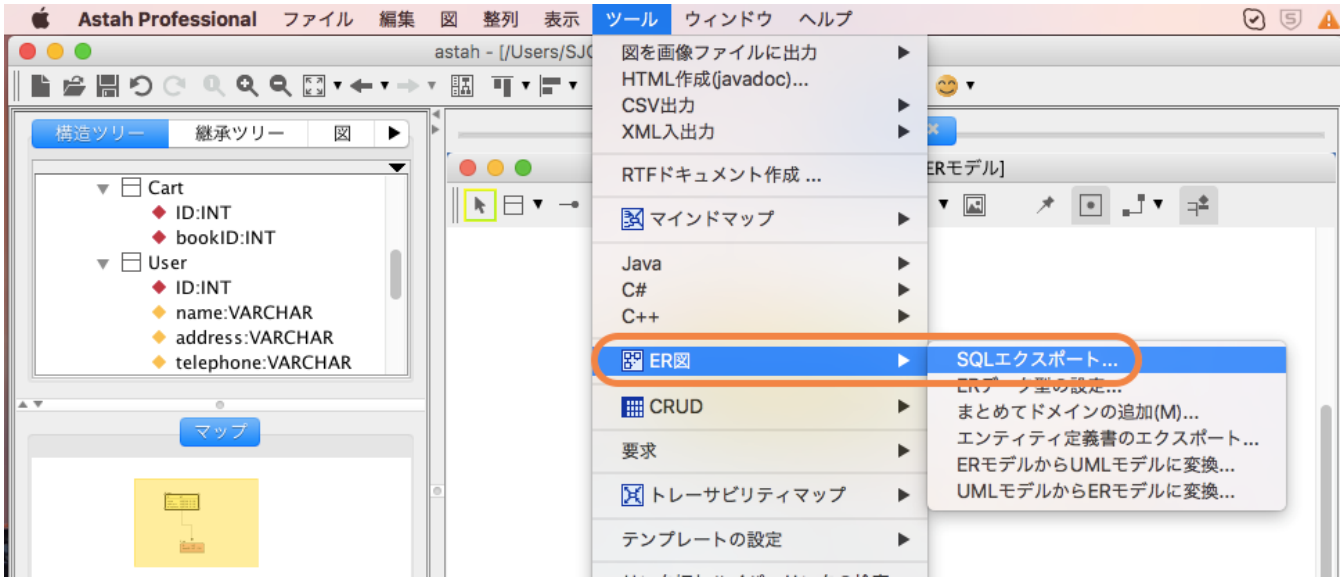
SQL 出力(SQL-92 準拠)してみよう

デモ動画:https://www.youtube.com/watch?v=zSJtEOVstnQ

SQL-92 相当のDDL を出力できます。以下のモデルでSQL を出力してみましょう。



[ツール]-[ER 図]-[SQL 出力]を選択します。



以下のダイアログが表示されますので、オプションボタンを押下します。

次のようなオプションを設定できます。

出力した結果のサンプルです。