「データベース」の講義について

- 授業の目的（シラバスより）

  データベースとは、**実世界**から収集されたデータを、複数のアプリケーションから**共有**できるよう統合したものである。本講義では、現在、最も広く使われている**リレーショナルデータベース**を中心に、データベースに関する基本的な概念を講義するとともに、データベースの操作言語として標準的な**SQL**の実習も行ない、データベースに対する理解を深める。

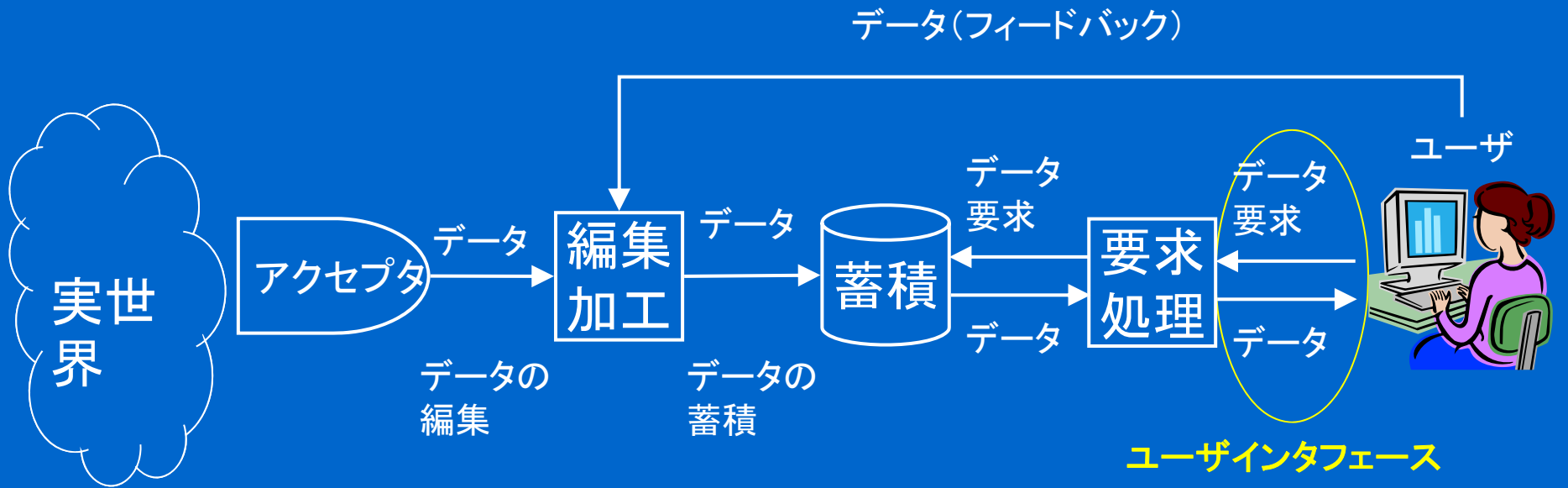
  到達目標：データベースの基礎的な概念と基本的なデータ操作方法について説明できる。

データベースの意義

- 情報科学分野での意義
  - ➢「**データ構造**とアルゴリズム」: アルゴリズムの特徴を議論するには、取り扱うデータの構造が重要
- 実社会での意義
  - ➢実社会では、プログラム（アルゴリズム）でなく、**データが重要**である場合がある
    - 経済・金融（株価、為替レートなど）、地球環境（天候、気象、災害など）、報道・出版（ニュース、新聞、書籍、学術誌）、自然科学（実験・観測、遺伝情報など）
    - ウェブ（Wikipedia, twitter, facebookなど）
    - センサーデータ（気象衛星、GPS, 地球観測衛星など）
  - ➢実社会の問題はデータベースと機械学習で対応可能?
    - 例　IBMのWatson

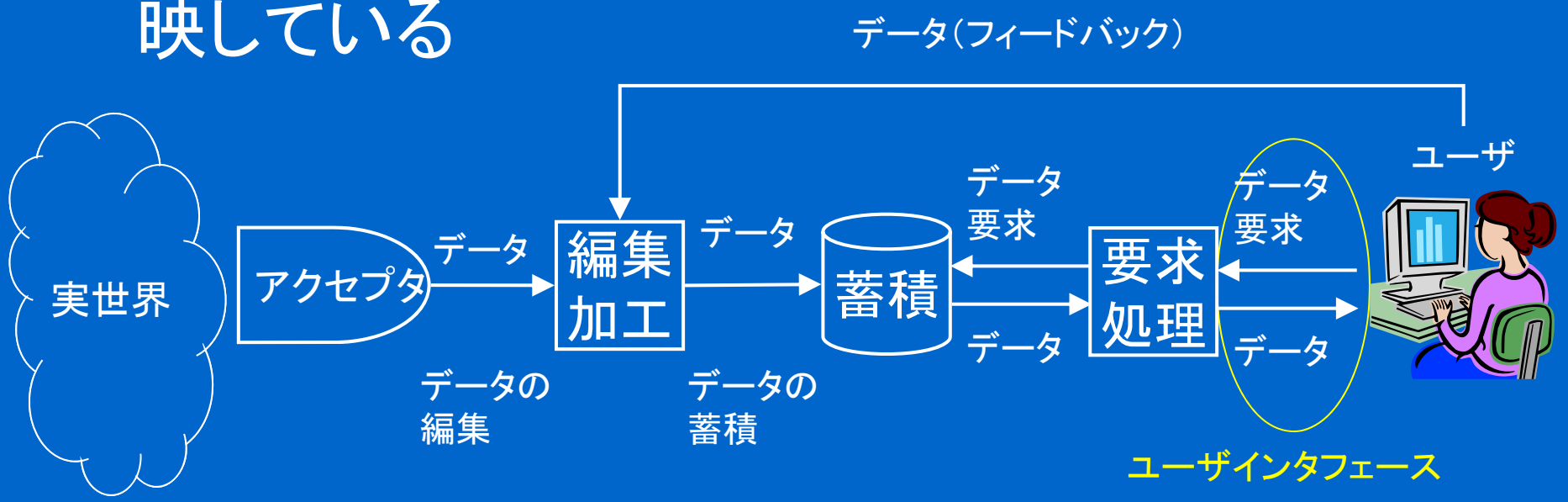
データとは？

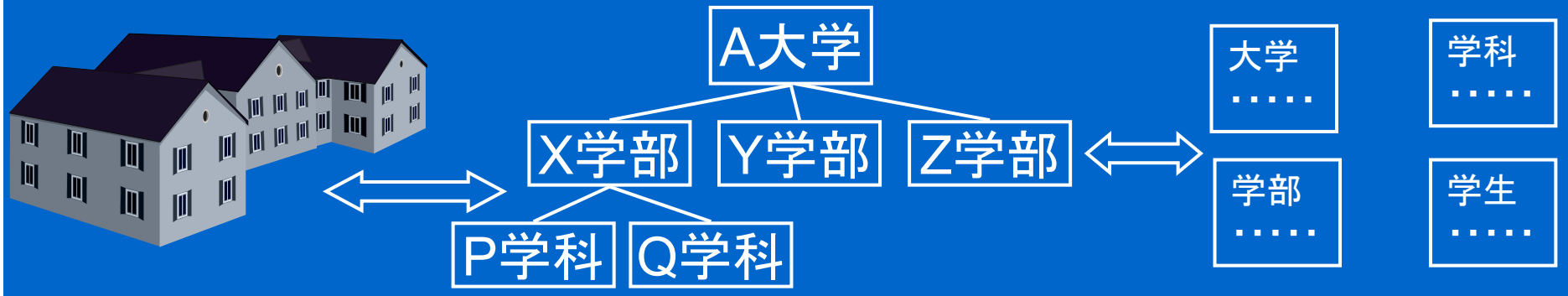
- データベースの世界では、「データ」とは単なる数値や文字列ではなく、**実世界の事物の持つ性質を反映**したものを指す
- データは、計算機上（通常、ディスク）に取り込まれるまでにユーザの要求に応じた編集・加工が成される



実世界の事物とデータの関連

- データベースのデータは実世界の事物を反映している



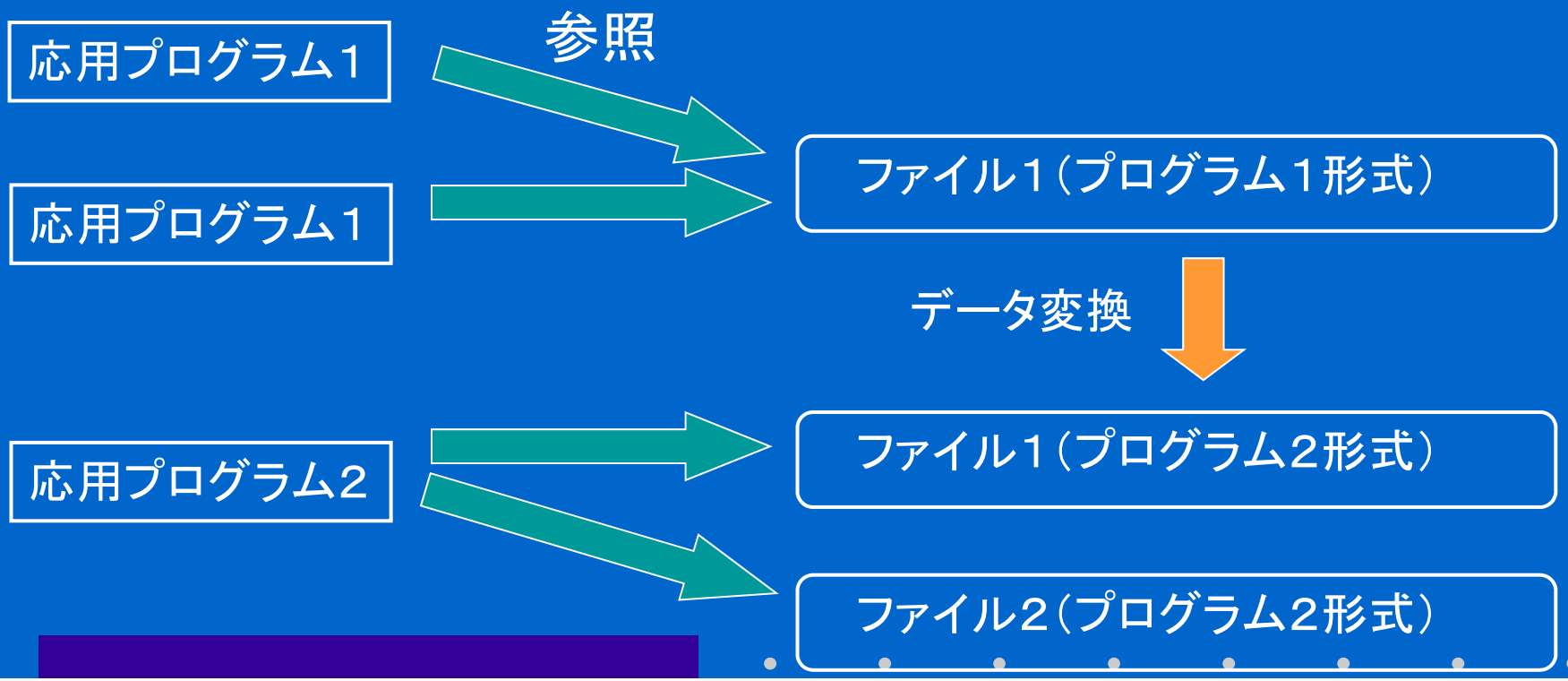


データと情報

- データは実世界の事物の性質を反映しているが、それだけでは「**情報**」(ユーザにとって価値のあるデータ)にはならない
- データと情報の区別は、厳密にはユーザが持っている知識に依存する(ユーザが既に知っているデータは、そのユーザに取って情報とはならない)

ファイル

- ファイルは**応用プログラム**（データを参照するプログラム）に依存する
- ある応用プログラムで作成したファイルを、別の応用プログラムから参照するにはデータ変換が必要
- 同じファイルを同時に複数の応用プログラムから更新するときは、応用プログラム側に相互排除等の更新制御手続きが必要



データ管理における問題点の例

① データを入れたファイルを置いているディスクを変更したら、ファイルにアクセスできなくなった

② データを複数のファイルに分散して入れていたが、更新を繰り返すうちに、各ファイルのデータ間の関係がわからなくなった

③ 同一のデータを多人数で利用しているうちに、ファイルの使用容量が増加した

④ 同一のデータを複数の別のファイルに入れており、相互に変換する必要がある

⑤ ファイル中に機密情報が含まれているため、ファイルのアクセスを制限すると、機密でない情報もアクセスが制限される

データ管理における問題点の原因

① ディスクの構成の変更によりファイルのパス名が変わった（データのアクセスがファイルのアクセス法に依存）→**データ独立性の未達成**

② データ間の関係を管理する方法がない→**データー貫性の欠如**

③ 個々のユーザが自分用にファイルのコピーを持っていた→**重複データを抑制できない**

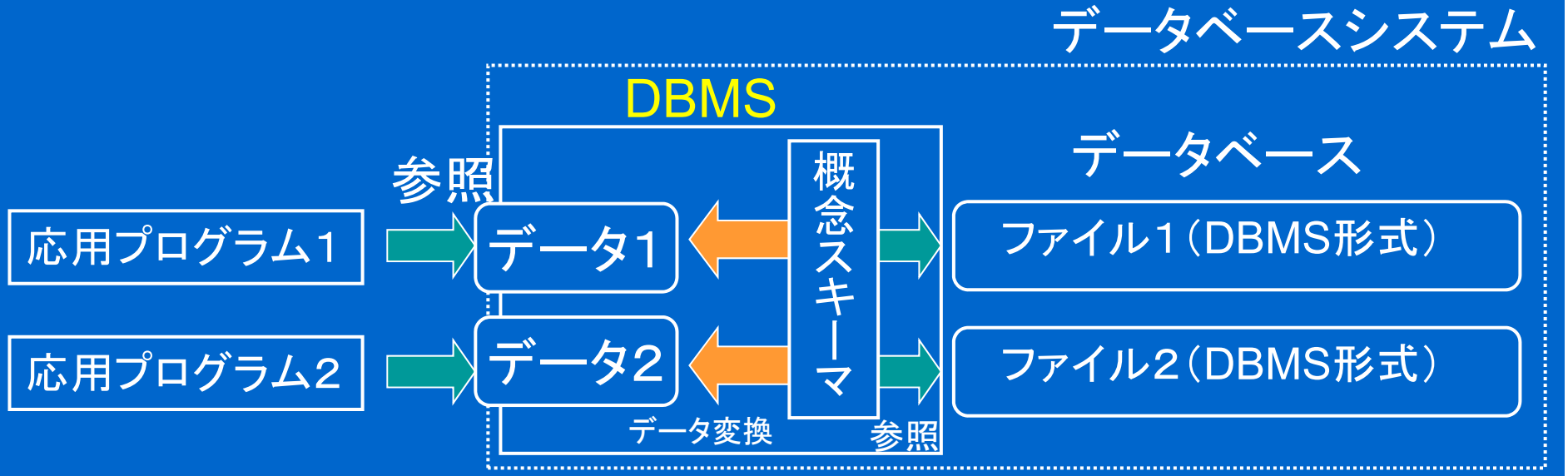
④ 同一のデータでもソフトウェアごとにファイルの形式が異なる→**データの標準化の未達成**

⑤ データのアクセス制限がファイル単位でしかできない→**データの機密保護の未達成**

➔**データベースの導入により解決**

データベース管理システム（DataBase Management System）

- 応用プログラムはファイルを直接参照できず、**データベース管理システム**(DBMS)を通して参照する
- 応用プログラムごとのデータ変換はDBMSが行う



データモデリングとは？

- データベースを開発する過程では、実世界の事物をDBMS（データベース管理システム）で扱えるデータの形式で記述する必要があり、この過程をデータモデリングと呼ぶ

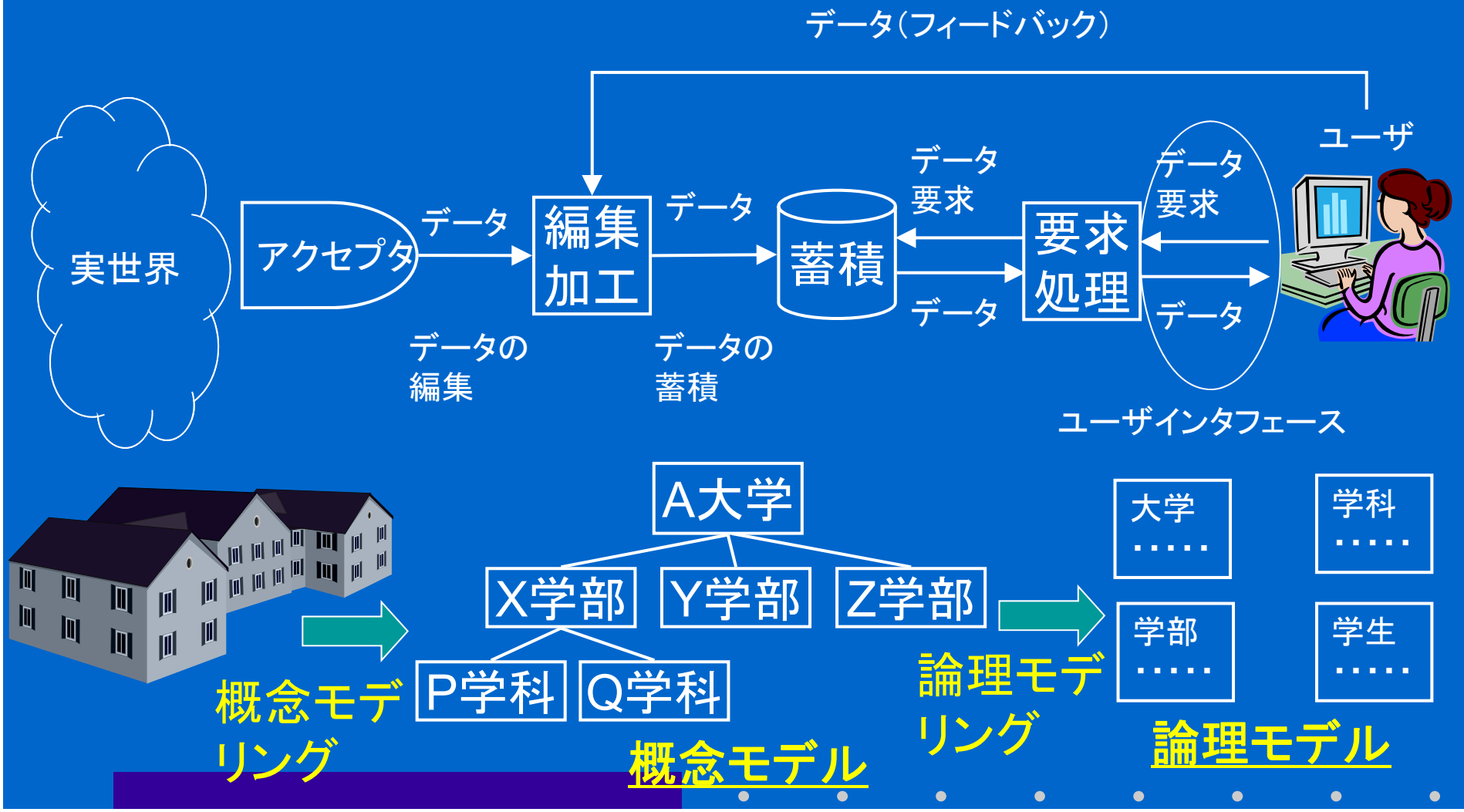
データモデリングは、次の2段階で行われる

1. 概念モデリング(Conceptual Modeling)
   - 実世界の事物とそれらの相互の関連をもとに、それを表現した概念モデルを作成する
2. 論理モデリング(Logical Modeling)
   - 概念モデリングで作られた概念モデルをもとに，計算機上で実際にデータベースを設計するための論理モデルを作成する.

データモデリングの記号系

- 概念モデリング： 概念モデル記述言語
- 論理モデリング： データモデル: Data Model

2段階のモデリング（概念モデルと論理モデル）



概念モデル(Conceptual Model)

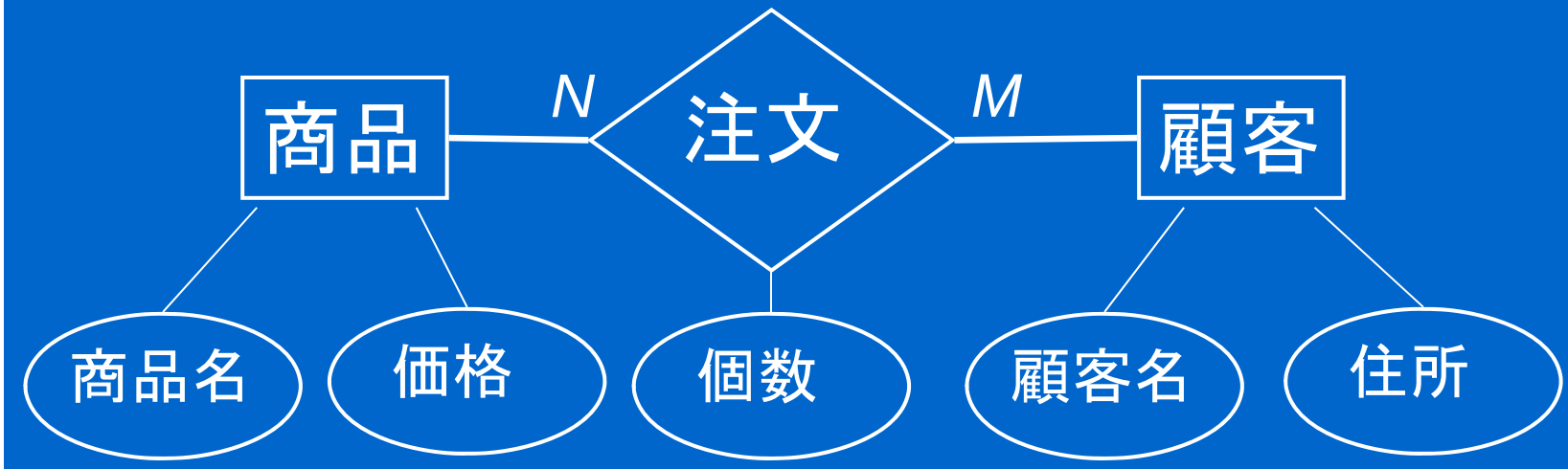
- 実世界の事物を忠実に反映するモデル
- 計算機上で直ちに実装可能かどうかは問わない
- 概念モデルを表現する記号系としては、概念モデル記述言語(Conceptual Modeling Description Language)が使われる

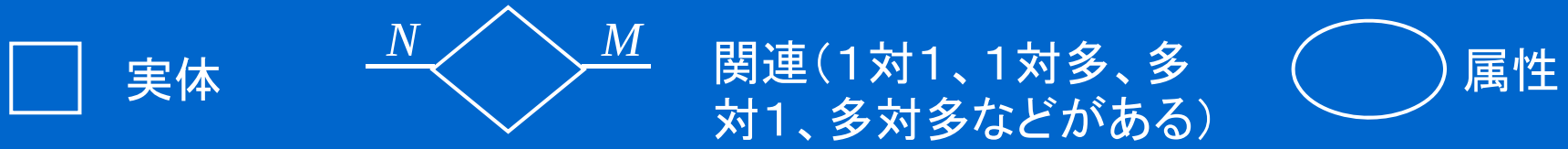
概念モデル記述言語 (Conceptual Modeling Description Language)

- 実体ー関連モデル(Entity-Relationship Model; ERM)
  - 実体(entity, 実世界に存在する事物に対応)と関連(relationship, 実体間に成り立つ対応関係)により記述.
  - 実体と関連は, それらの持つ特徴や性質を表す属性(attribute)を持っている
- 意味的データモデル (Semantic Data Model)
  - 関数型データモデル(FDM: Functional Data Model), 一般化意味データモデル(GSM: Generalized Semantic Data Model)
- CADの形状モデル
  - ワイヤーフレーム, サーフェイス, ソリッド
- UML (Unified Modeling Language)
- 自然言語

実体ー関連モデル（ERM: Entity Relationship Model）

- **実体(entity)**：実世界に存在する事物に対応
- **関連(relationship):**実体と実体の間にある関係
- **属性(attribute)**：実体や関連の持つ性質（名前や数など）





実体ー関連モデルの例（Chenの記法）

実体－関連モデルの例（Crow's Foot記法）(1)

(a) Crow's Foot記法による表記（関連には属性がない）

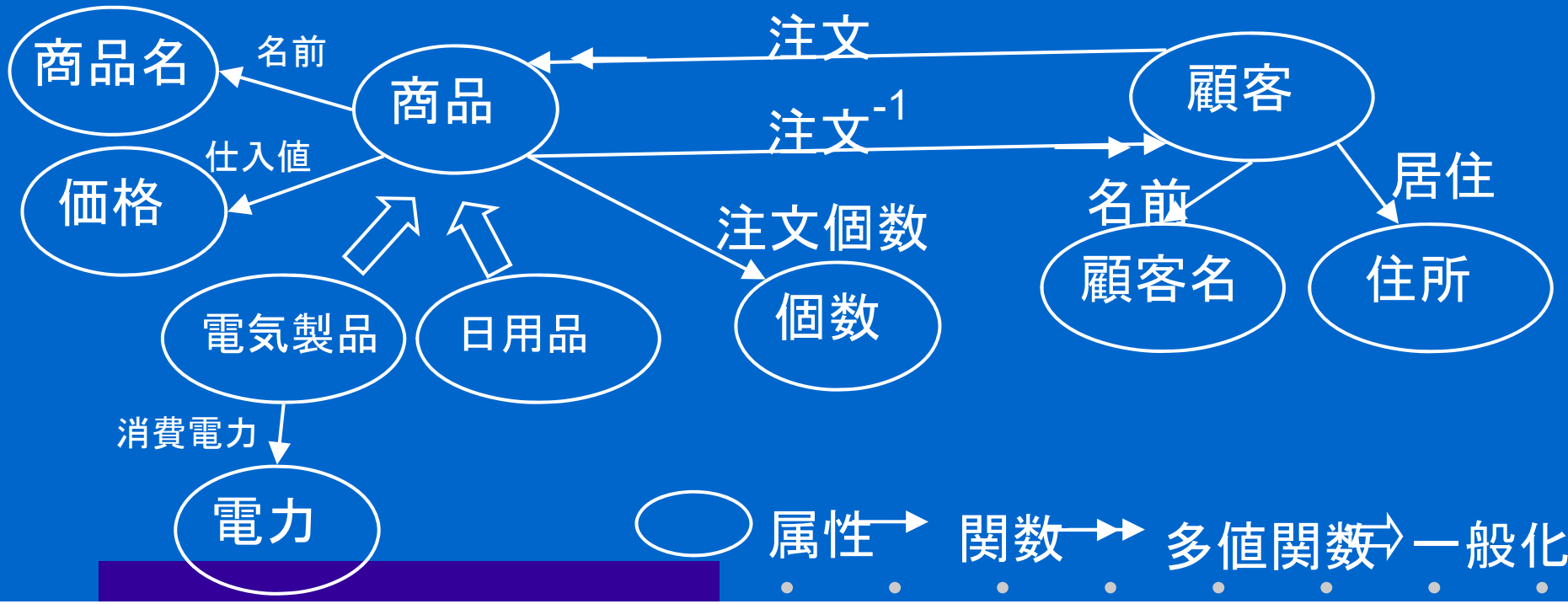
(b) 参考：関連に対応する実体を作成し属性を付与

実体－関連モデルの例（Crow’s Foot記法）(2)

(c) 実体間の対応関係の個数の表記例

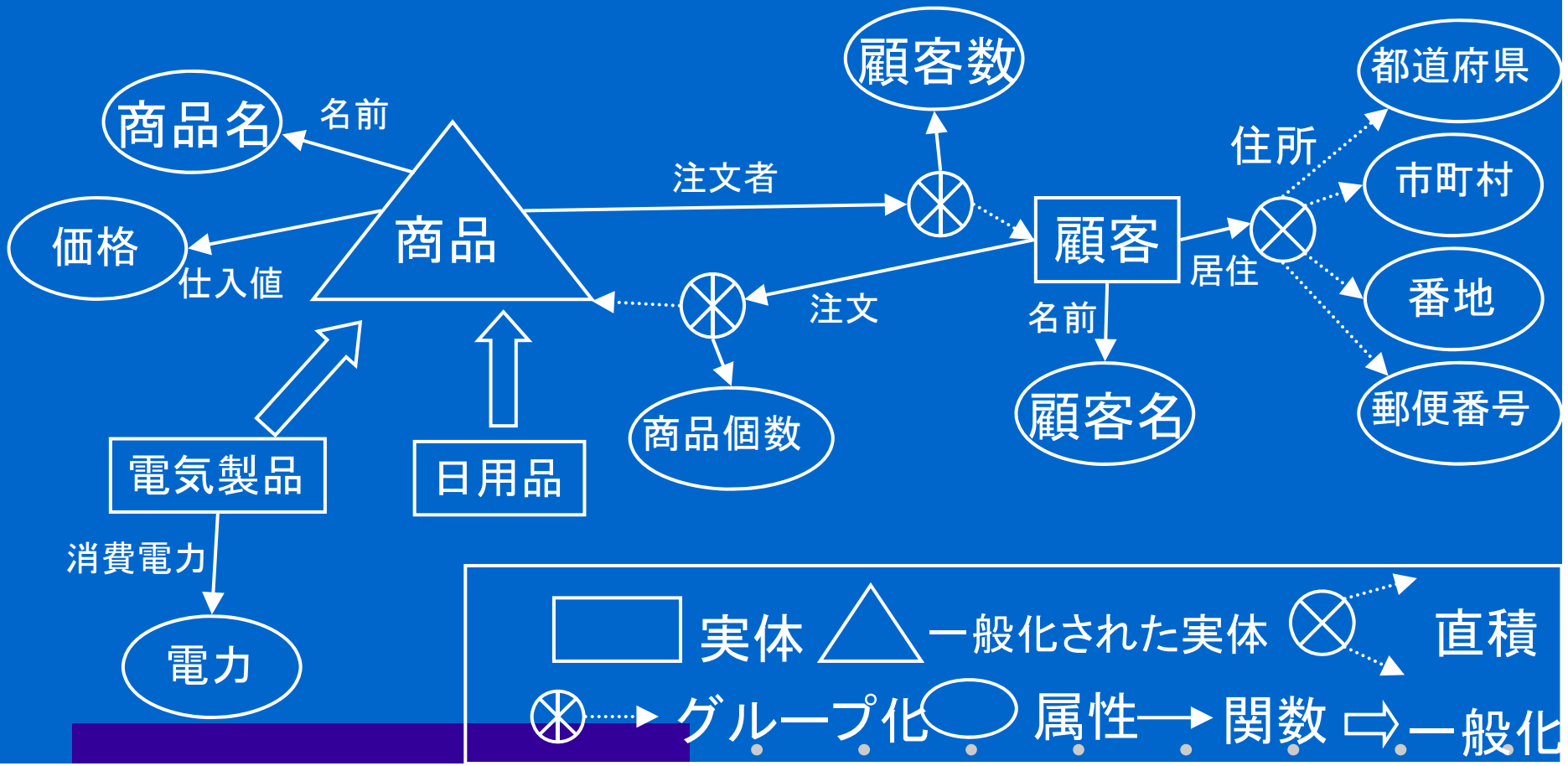
関数型データモデル (FDM: Functional Data Model)

- 実体ー関連モデルでの実体と属性を区別せずにどちらも属性で表現
- 属性間の関連は方向性のある**関数(function)**で表現(逆方向の関連は、逆関数で表す)



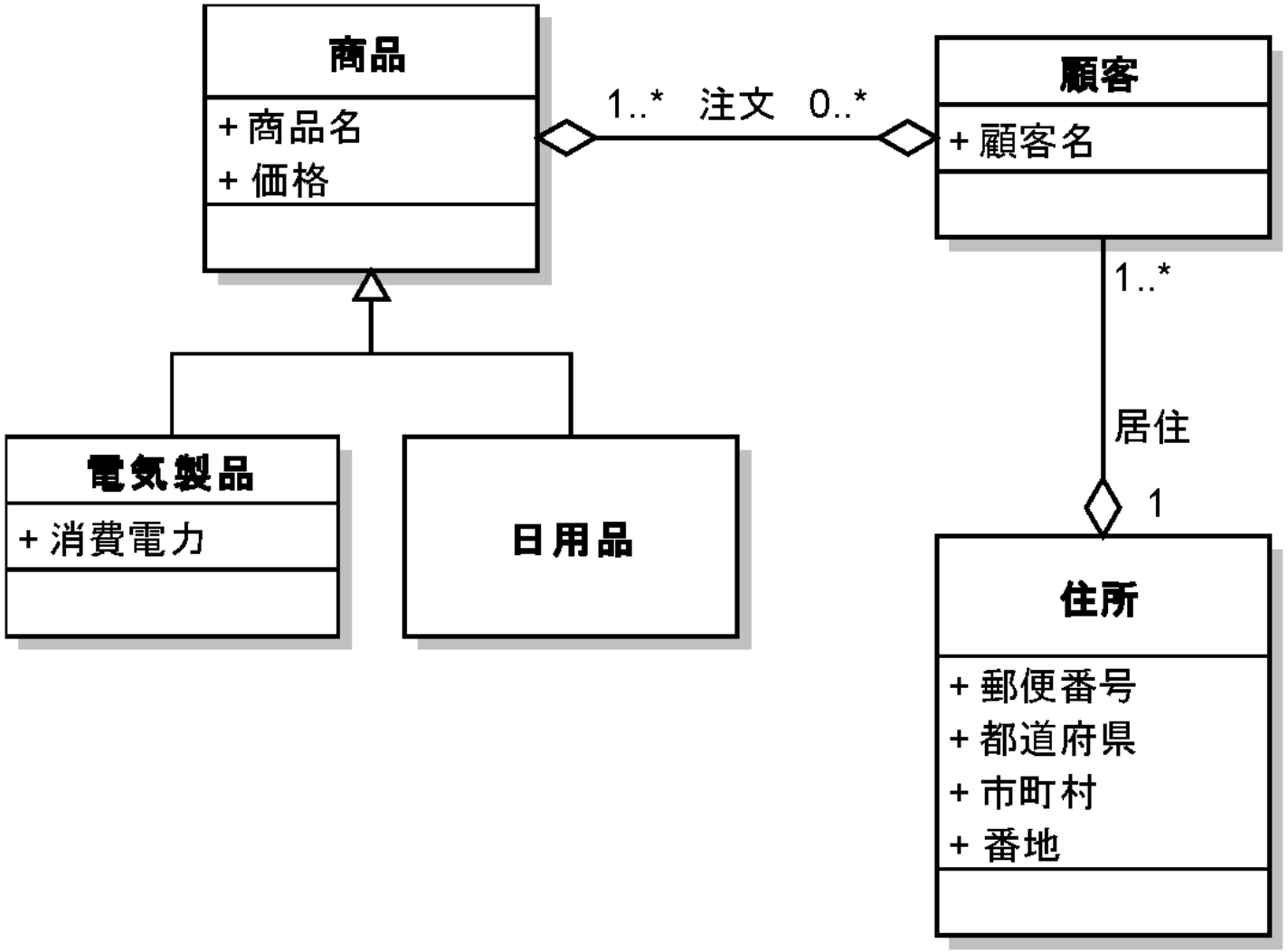
一般化意味データモデル
(GSM: Generalized Semantic Data Model)

- 記述能力が高く、多様な表現が可能



UML (Unified Modeling Language)

- 本来はオブジェクト指向プログラミング用だが、クラス図は概念モデルの記述に利用可能



論理モデル(Logical Model)

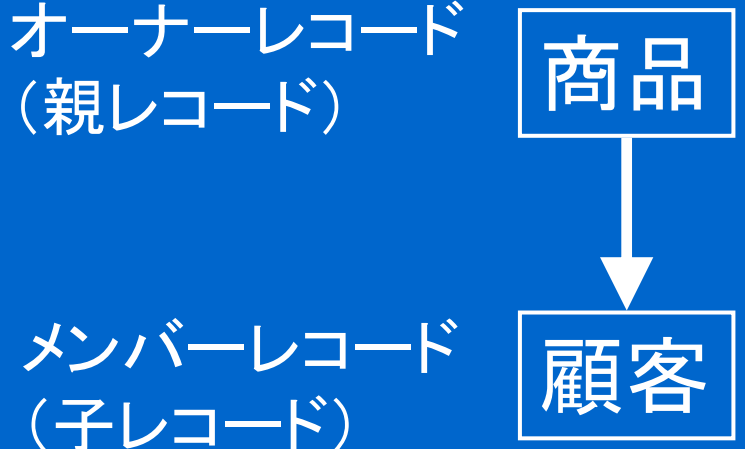
- 概念モデルをもとに、実際に計算機上で実装するために作成される。
- 記述能力よりも，2次記憶上で表現しやすく、アクセス効率が高い形式であることが重要である。
- 論理モデルを表現するための記号系としては、**データモデル(Data Model)**が使われる。

データモデル

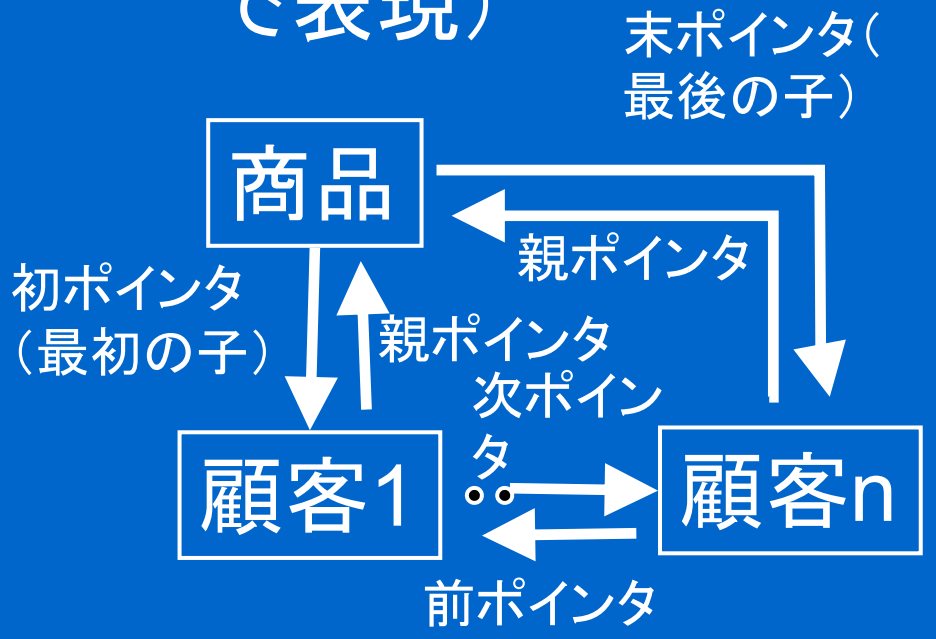
- ハイアラキカルデータモデル（階層データモデル：Hiearachical Data Model）
  - データ間の関係を階層的な表現（親子関係）で記述
  - IMS（IBM社）などのDBMSがある
- ネットワークデータモデル(Network Data Model)
  - ハイアラキカルデータモデルの親子関係を、子データが複数の親データを持てるように拡張
  - CODASYL(Conference on Data Systems Languages)で提唱されたデータモデル
  - IDS（General Electric社のC. Bachmanにより開発）などのDBMSがある
- リレーショナルデータモデル（関係データモデル：Relational Data Model）
  - 次回に詳しく説明するので、ここでは例のみ示す

ハイアラキカルデータモデル（階層データモデル：Hiearachical Data Model）

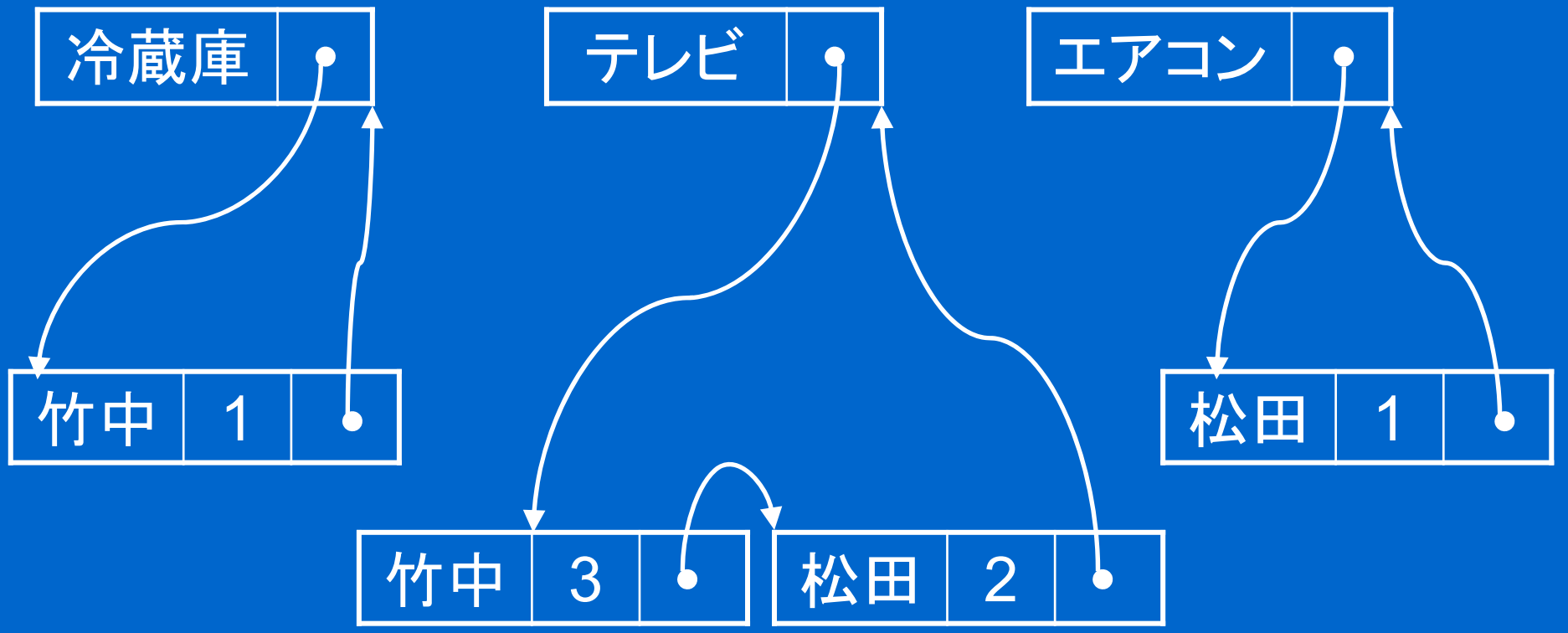
モデル（親子関係で表現）



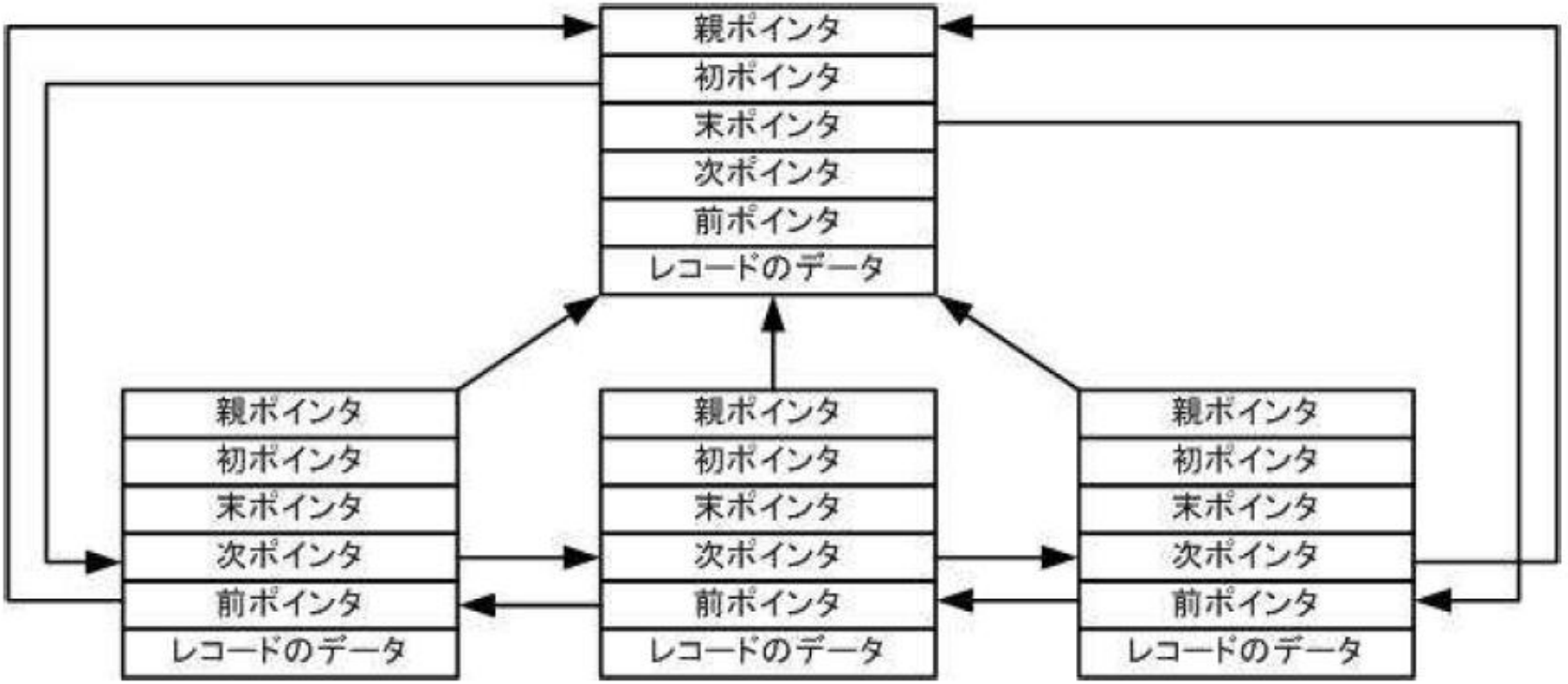
実データ（ポインタで表現）



ハイアラキカルデータモデル (Hierarchical Data Model)の例



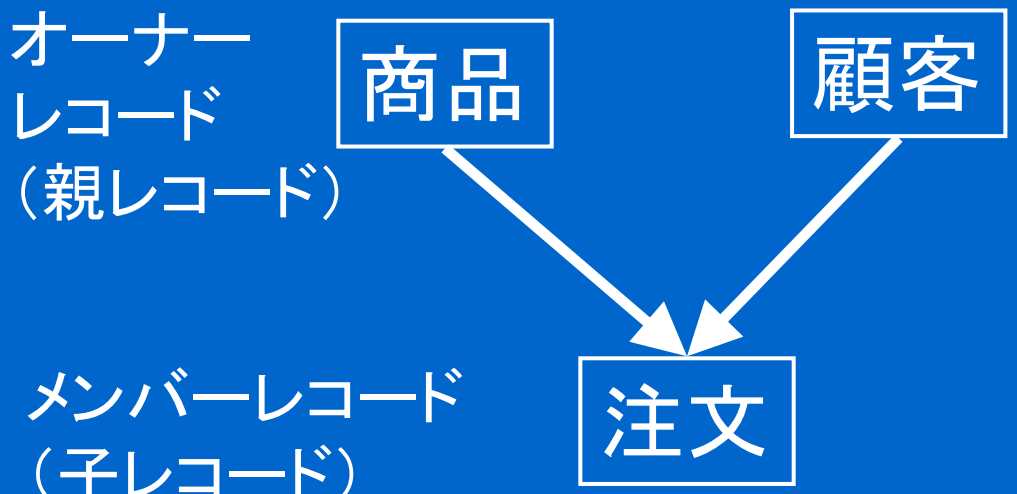
ハイアラキカルデータモデルの実装



ネットワークデータモデル (Network Data Model)

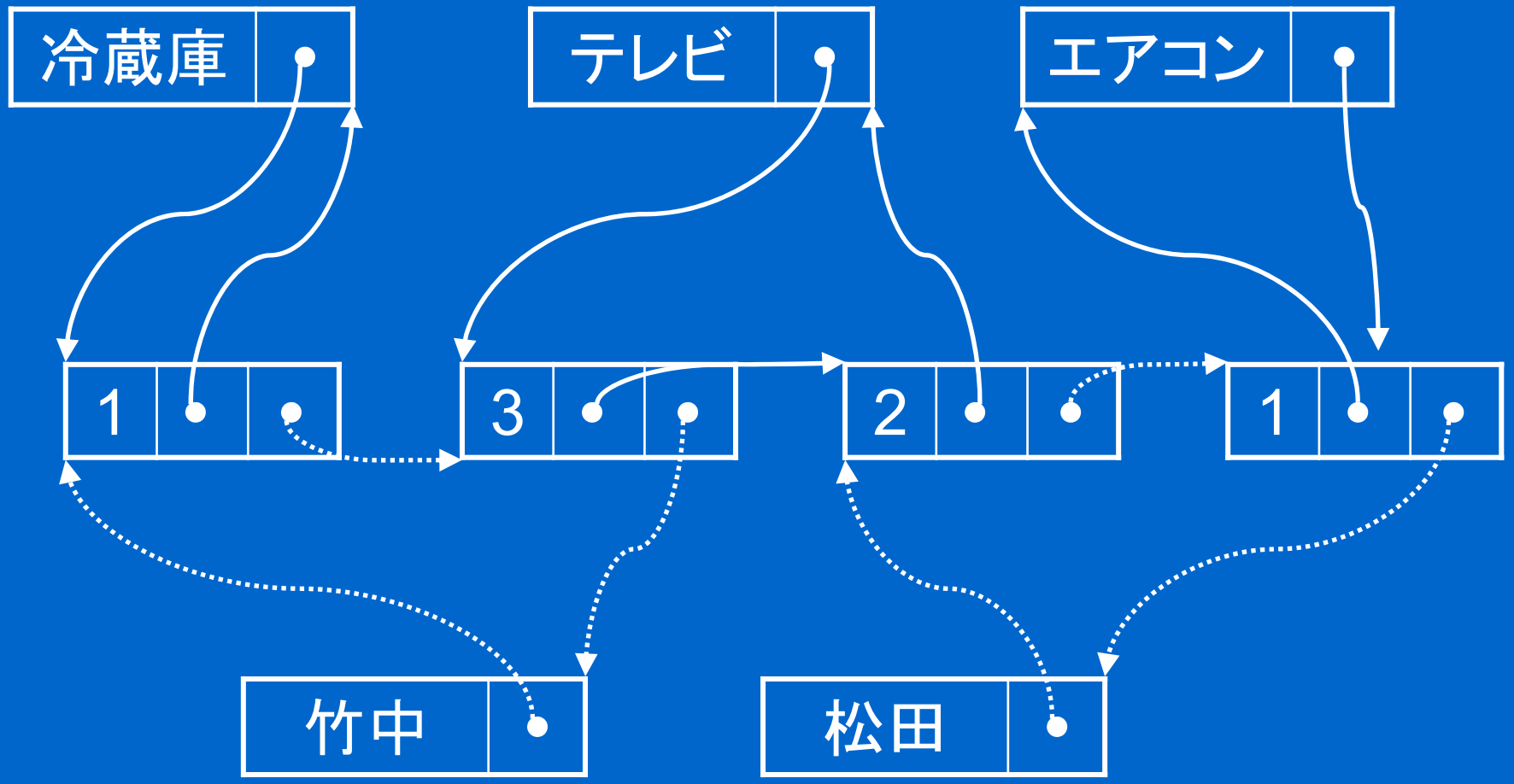
- 親子関連型(set type)と呼ばれる形式で表現
- 複数の親を持つ（実データでは、ハイアラキカルデータモデルのポインタを複数持つ）

モデル



実データ

ネットワークデータモデル(Network Data Model)の例



リレーショナルデータモデル (Relational Data Model)の例

商品

商品番号	商品名
G1	冷蔵庫
G2	テレビ
G3	エアコン

顧客

顧客番号	顧客名
C1	竹中
C2	松田

注文

注文番号	商品番号	顧客番号	個数
O1	G1	C1	1
O2	G2	C1	3
O3	G2	C2	2
O4	G3	C2	1

リレーショナルデータモデルでは、商品番号、顧客番号、注文番号（属性名に下線）という識別子(ID)による参照が必要となることに注意。

データモデリングを2段階に分ける意義(1)

- モデリングの効率向上

  データモデリングには、

  - 実世界のデータを忠実に表現するという目標と、
  - 計算機上で実装可能でアクセス効率が良いデータベースを設計するという目標

  の**2種類の目標**が存在する。

  １回の変換でこれら2つを同時に達成するのは困難であるため、２段階のモデリングによりこの2つの目標を段階的に達成する

データモデリングを2段階に分ける意義(2)

修正の容易さ

- 実世界の変化に伴うデータベースの修正
  - 実世界に変化があっても概念モデルを通じて修正することによりデータベースの修正が容易となる
- データベースの実現上の理由による修正
  - アクセス効率の向上等のデータベースの実現上の理由で、論理モデルを変更する必要が生じても、論理モデリング過程の変更だけですみ、概念モデリングを再度行う必要はない

3層スキーマ

ユーザ1

応用プログラム1　応用プログラム2

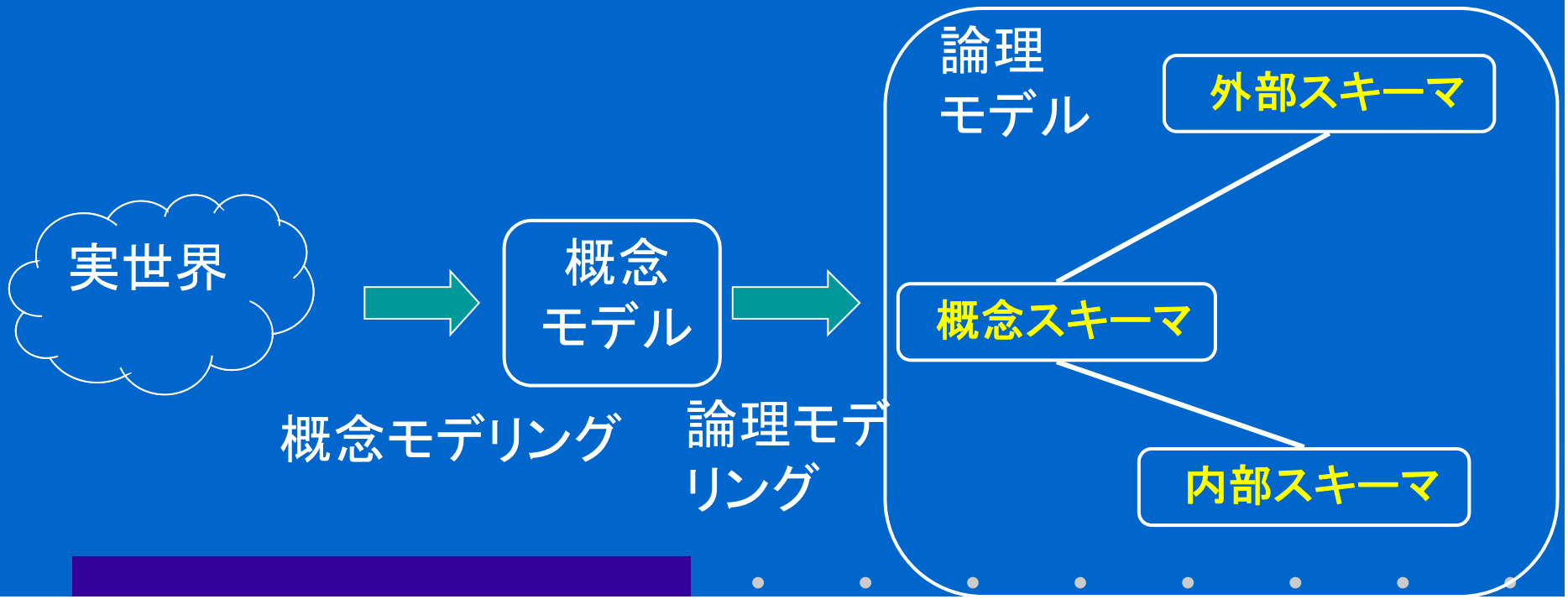
外部スキーマ(ビュー)
・ユーザやプログラマに見せるデータ形式
(ユーザやプログラムごとに機密保持などの理由で異なるように設定できる)
・リレーショナルデータモデルでは、SQLのCREATE VIEW 文で作成

概念スキーマ
・実世界に対応してデータを記述したデータ形式
・リレーショナルデータモデルでは、SQLのCREATE TABLE 文で作成

内部スキーマ
・データベースの内容をディスクに格納するときの物理的な形式
・固定長のページ、B木やAVL木の構造、インデックスの作成など

3層スキーマとデータモデリング

- 次の二つは用語が似ているが対象が異なる
  - ANSI/SPARCモデルの3層スキーマ（内部スキーマ／概念スキーマ／外部スキーマ）
  - データモデリングのモデル（概念モデル／論理モデル）
- 3層スキーマは論理モデルに含まれる



データベース構築の意義

データを単にファイルに格納するのではなく、DBMSを使ってデータベースとしてまとめることには以下のような意義がある

1. データ独立性の達成
2. データの一貫性の保証
3. データ蓄積の効率化（重複データの抑制）
4. データの標準化
5. データの機密保護

データ独立性

- データ独立性とは、データベースの構成の変更が応用プログラムに影響を及ぼさないこと
  - 物理的データ独立性
  - 論理的データ独立性

  の2種類がある
- 大量のデータを効率良く管理するためには必須の概念であり、データベースを構築する意義の一つである

物理的データ独立性

- **内部スキーマと概念スキーマが独立であること、すなわち、内部スキーマの変更が概念スキーマに影響しないこと**
  - 内部スキーマ:DBMSがデータベース中のデータを物理的にディスク上で管理している形式
  - 概念スキーマ:データベース管理者がデータベースを設計・管理するときの形式
- 物理的データ独立性の例:

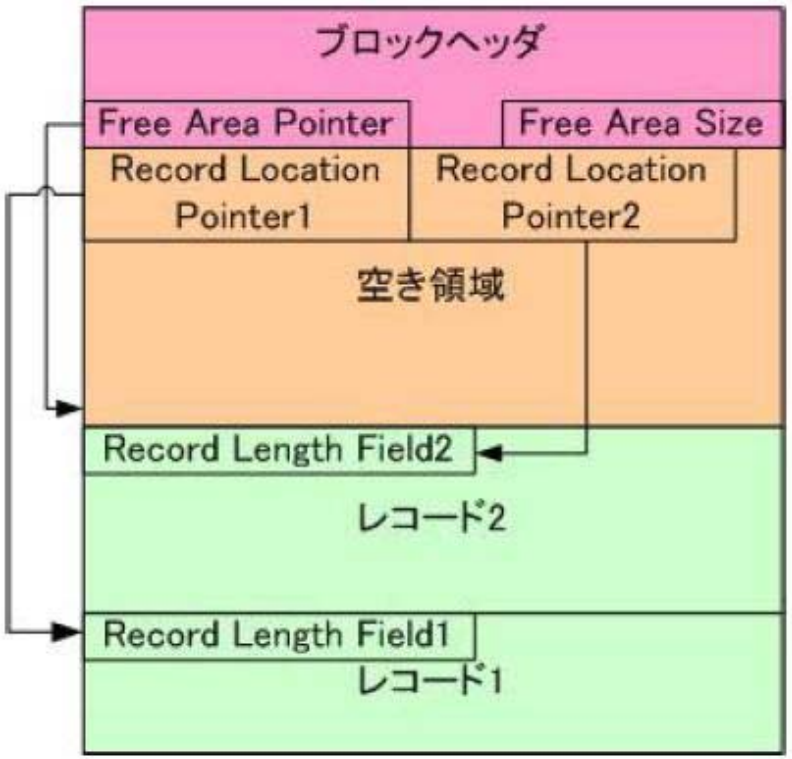
  データベースを置いているディスクや、DBMSの稼働しているOSや計算機、DBMSそのものを変更しても、データベース自体を設計しなおす必要がない

論理的データ独立性

- **概念スキーマと外部スキーマが独立であること、すなわち、概念スキーマの変更が外部スキーマに影響しないこと**
  - 概念スキーマ：データベース管理者がデータベースを設計・管理するときの形式
  - 外部スキーマ：応用プログラムがデータベースを検索・更新する形式
- 論理的データ独立性の例：

  データベースの構成を変更（主にデータ項目の追加・削除）しても、応用プログラムを修正する必要がない

3層スキーマの例



内部スキーマ

商品

商品番号	商品名
G1	冷蔵庫
G2	テレビ
G3	エアコン

顧客

顧客番号	顧客名
C1	竹中
C2	松田

注文

注文番号	商品番号	顧客番号	個数
O1	G1	C1	1
O2	G2	C1	3
O3	G2	C2	2
O4	G3	C2	1

概念スキーマ

注文商品

商品番号	商品名	個数
G1	冷蔵庫	1
G2	テレビ	5
G3	エアコン	1

外部スキーマ（顧客の情報なしに注文個数だけ表示したい）