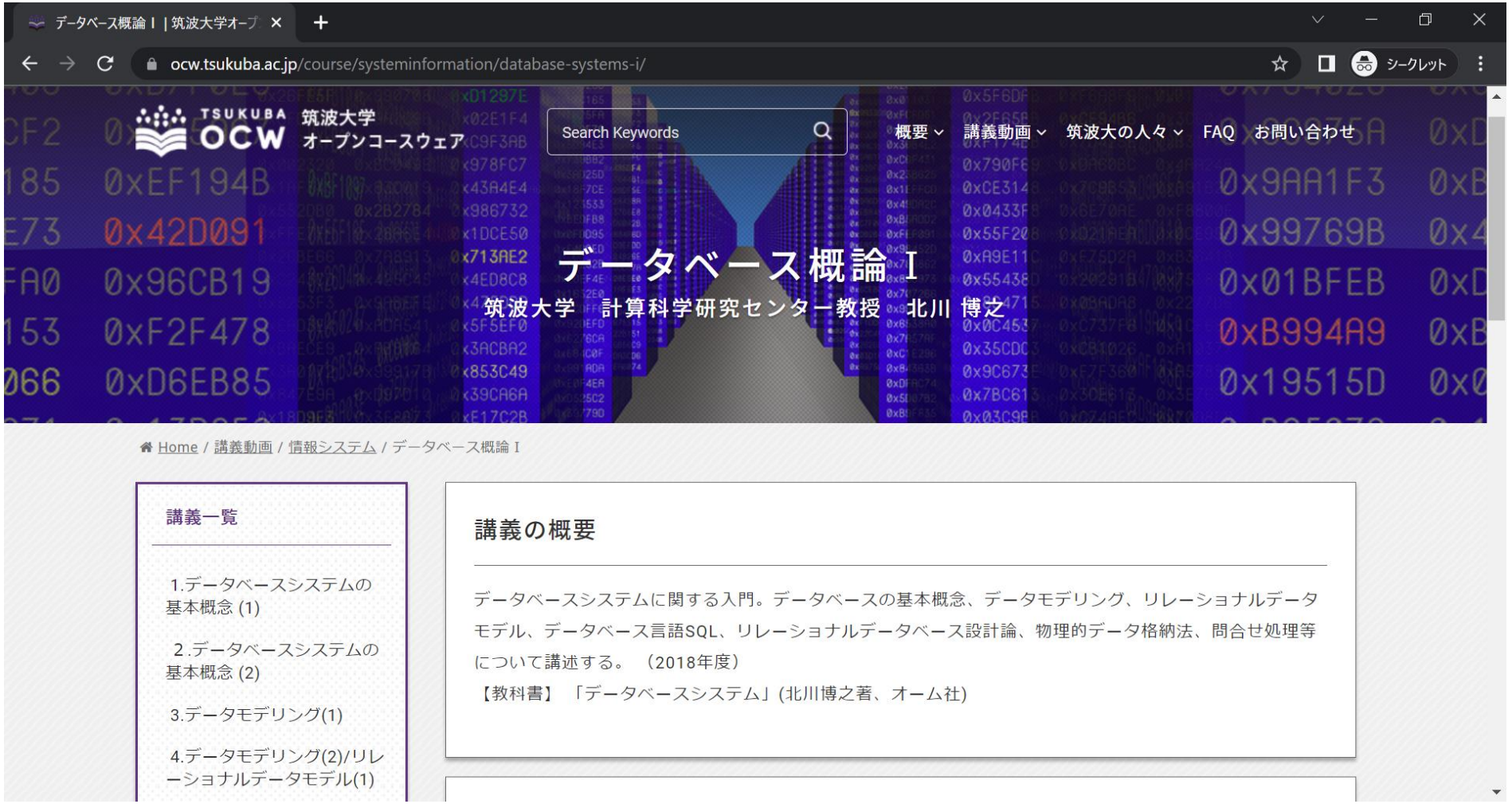
データベースの概要

データベースと情報検索　第2回

横浜市立大学　坂巻顕太郎
kentaro.sakamaki@gmail.com

情報学的な観点でのデータベース論

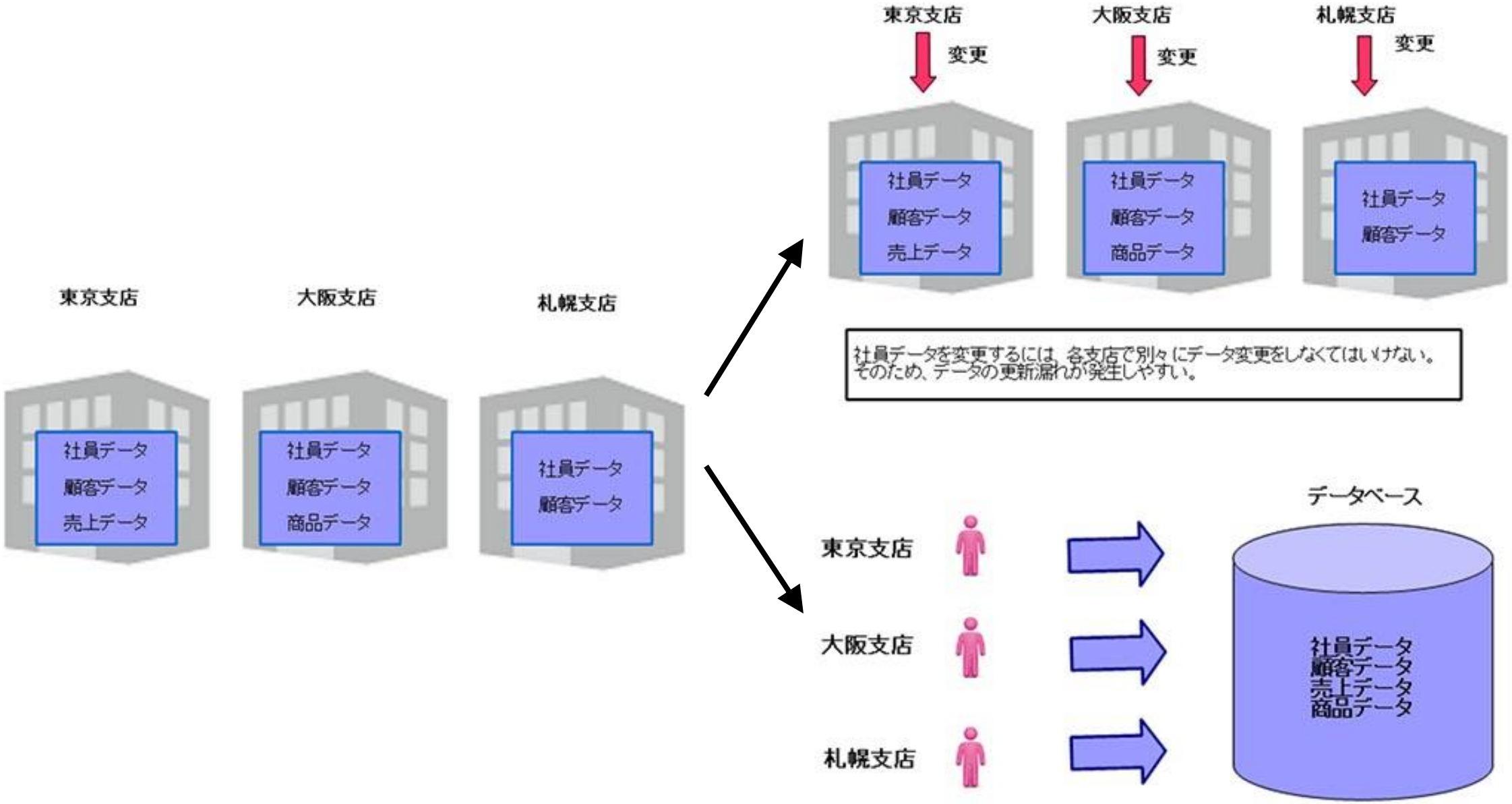
https://ocw.tsukuba.ac.jp/course/systeminformation/database-systems-i/



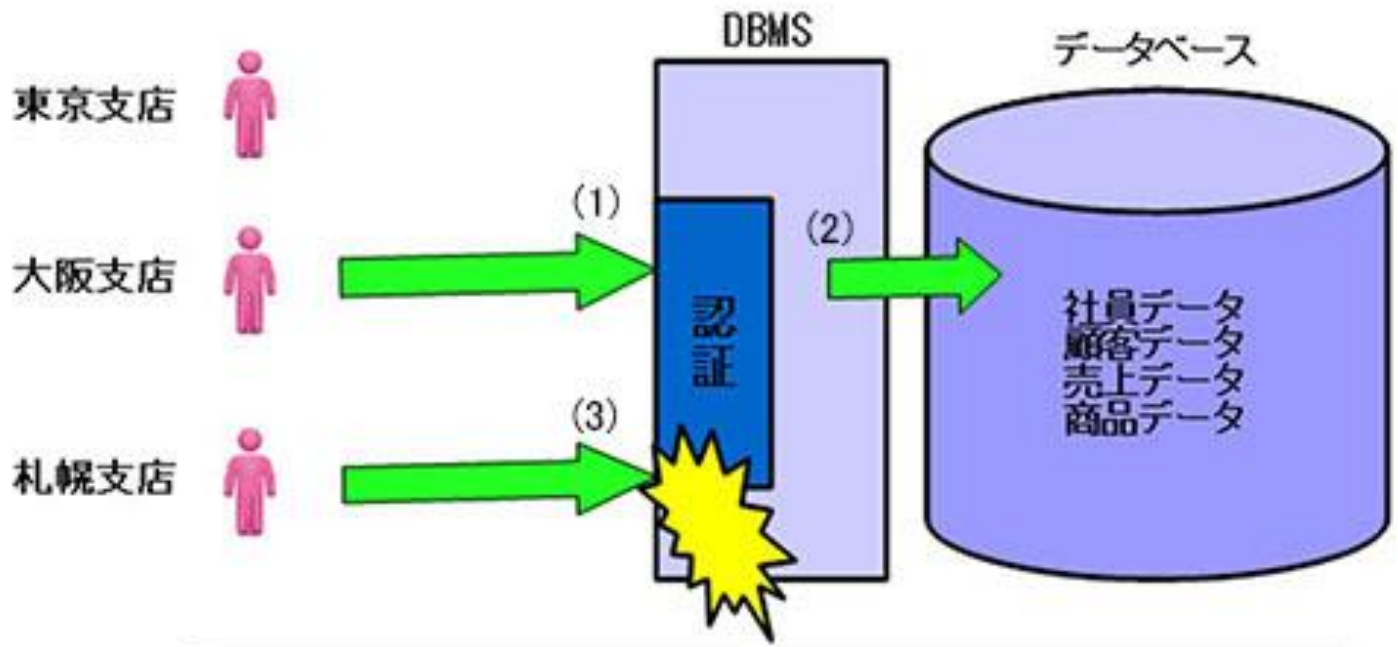
データベースの概要

- データベース
  - データを共有利用するために1つにまとめたデータの集合体
    - 顧客データ、商品データなど、企業活動で生じる様々なデータの集合体
  - 階層型、ネットワーク型、リレーショナル型などがある

- データベース管理システム（DBMS: Database Management System）
  - 複数でデータベースを利用する際に適切な管理をするための仕組み
    - ユーザーからのデータベースに対する要求は必ずDBMSを介して行われる
  - データを処理する前に、順序制御、権限確認などを行う


東京支店
大阪支店
札幌支店
社員データ
顧客データ
売上データ
社員データ
顧客データ
商品データ
社員データ
顧客データ
東京支店
変更
大阪支店
変更
札幌支店
変更
社員データ
顧客データ
売上データ
社員データ
顧客データ
商品データ
社員データ
顧客データ
社員データを変更するには、各支店で別々にデータ変更をしなくてはいけない。
そのため、データの更新漏れが発生しやすい。
東京支店
大阪支店
札幌支店
データベース
社員データ
顧客データ
売上データ
商品データ

データベース管理

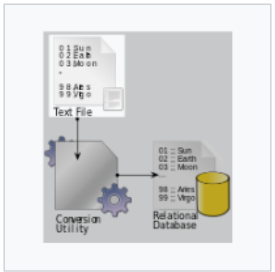

DBMS
データベース
東京支店
大阪支店
札幌支店
(1)
(2)
(3)
認証
社員データ
顧客データ
売上データ
商品データ
(1)DBMSがアクセス権限があるかチェックする
(2)アクセス権限がある場合は、データベースにアクセスできる。
(3)アクセス権限がない場合は、アクセス不可。

データモデルのタイプ [編集]

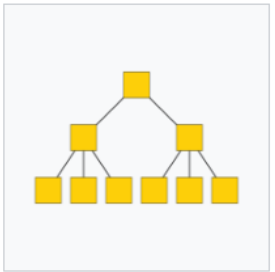
データベース・モデル [編集]

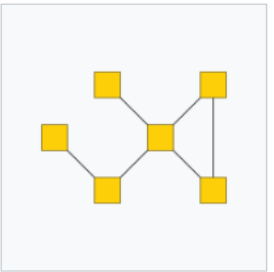
データベース・モデル (en:database model) は、どのようにデータベースが構造化され、使われるかを記述する理論または仕様である[12]。いくつかのそのようなモデルが提案されてきた。広く知られたモデルは以下を含む:

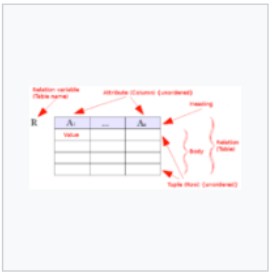
- フラット・モデル
  これは、厳密にはデータモデルとして認められないかもしれない。フラット（またはテーブル）モデルは、与えられたカラムの全要素が、同じような値であり、そして1つの行の全要素が互いに関連していると想定される、データ要素の単一の2次元配列で構成される。
- 階層型データモデル：このモデルにおけるデータは、それぞれ同じレベルのリストに特定の順序でレコードを保持するネスト化と並べ替えフィールドを記述するそれぞれのレコードへの単純な上昇リンクを暗示する、ツリー構造に組織化される。
- ネットワーク型データモデル：このモデルは、レコードとセットと呼ばれる、2つの基本的概念を使うデータを組織化する。レコードはフィールドを含み、セットはレコード間の、1は所有者、多はメンバーである、1対多の関連を定義する。
- リレーショナル・モデルは、一階述語論理に基づくデータベース・モデルである。その中核アイデアは、とりうる値と値の組み合わせへの制約を記述する、有限個の述語変数を持つ述語の集合としてデータベースを記述することである[12]:468,467。
- スタースキーマは、データ・ウエアハウス・スキーマの最もシンプルなスタイルである。スタースキーマは、いくつかの「事実テーブル」（おそらく1つのみであり、その名前を正当化する）がどんな数の「次元テーブル」を参照する。スタースキーマは、重要な雪形スキーマの特別なケースと考えられる。

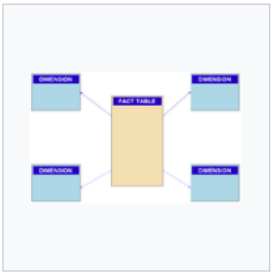


フラット・モデル


階層型データモデル


ネットワーク型データモデル


リレーショナル・モデル



スタースキーマ

リレーション（リレーショナルデータ）のイメージ

社員番号	社員名	給与	部門番号
7899	鈴木	300,000	10
7651	加藤	250,000	20
7565	佐藤	180,000	30
7422	山田	200,000	10

・「社員番号」「社員名」「給与」「部門番号」のデータを管理している。

・この例では、各列の組み合わせによる一行が、一人の社員のデータとして意味をもっている。

関係データベース

45の言語版

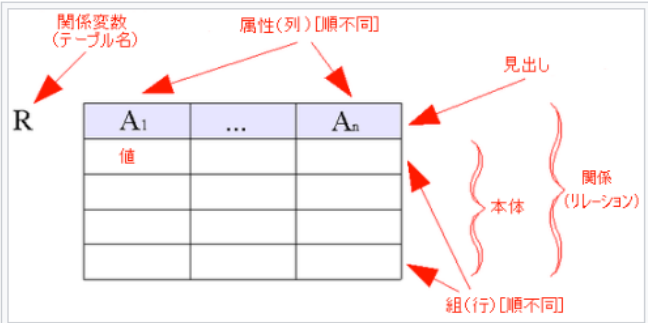
出典: フリー百科事典『ウィキペディア（Wikipedia）』

関係データベース（かんけいデータベース、**リレーショナルデータベース**、英: relational database）は、関係モデル（リレーショナルデータモデル、後述）にもとづいて設計、開発されるデータベースである。関係データベースを管理するデータベース管理システム (DBMS) を関係データベース管理システム (RDBMS) と呼ぶ。

Oracle Database、Microsoft SQL Server、MySQL、PostgreSQL、DB2、FileMaker、H2 Database などがRDBMSである[1] 。

関係モデル [編集]

詳細は「関係モデル」を参照



関係モデルの概念

関係モデルはIBMのエドガー・F・コッドによって考案された[2]現在もっとも広く用いられているデータモデルである。データベースの利用者は、クエリ（問い掛け）をデータベースに与え、データを検索したり、変更することができる。

データは表に似た構造で管理されるが、関係（リレーション）と呼ぶ概念でモデル化される。関係（リレーション）は組（タプル、表における行に相当する）、属性（アトリビュート、表における列に相当する）、定義域（ドメイン）、候補キー（主キー）、外部キーなどによって構成される。SQLなどに代表されるデータベース言語（問い合わせ言語）を用いて、関係に対して制限・射影・結合・和・差・交わりなどの関係代数演算（集合演算を含む）ないし関係論理演算を行うことで結果を取り出す。

関係を複数持つことも可能で、互いを関連させることも可能である。

例 [編集]

例えばある食品を扱う（架空の）通信販売会社における顧客管理データベースでは、顧客リストと物品販売リストは別々のデータ群である

リレーショナルデータの結合

「正規化」の必要性

- 社員表に部門データも含める
  - 3つしかない「所在地」の名称が入っている「部門表のデータ」を3拠点で計1万人のデータを保存する社員表に含めると、無駄に大きな表になる
- 所在地の名称を変える
  - 社員表に所在地のデータが含まれる場合、その拠点にいる社員全員のデータを変更する処理を行う必要があり、無駄な処理になる
- 「正規化」
  - 上記のような問題を避けるために、「データの重複を省きながら」、1つの表を複数の表に細かく分け、必要に応じてジョインできるようにする

関係の正規化

33の言語版

ページ　ノート　閲覧　編集　履歴表示

出典: フリー百科事典『ウィキペディア（Wikipedia）』

関係の正規化（かんけいのせいきか）は、関係データベース (リレーショナル・データベース) において、関係（リレーション）を正規形と呼ばれる形式に準拠させることにより、データの一貫性の維持と効率的なデータアクセスを可能にする関係設計を導くための方法である。正規形には様々なものが存在するが、いずれにせよ、正規化を行うことにより、データの冗長性と不整合が起きる機会を減らすことができる。

多くの関係データベース管理システム (RDBMS) は、論理的なデータベース設計とデータを格納する物理的な実装方法とが十分に分離されていないので、完全に正規化されたデータベースへのクエリ（検索質問）はパフォーマンスが良くないことがある。このような場合、パフォーマンスを向上させるためにデータの一貫性の低下と引き換えにあえて非正規化されることもある。

正規形の定義の解釈 [編集]

ある正規形であるためには、ある時点でたまたま関係（表、テーブル）中にあるすべての組（タプル、行）の値がその定義に当てはまるだけでは十分でなく、過去及び将来において、その関係中の組に増減があっても、定義から外れることがないように属性（列、カラム）が定義されていることを要する。

注意すべきは、実際の関係データベース管理システムでは、属性の定義域（ドメイン）に合致する限りで関係の中にどんな値からなる組でも入れることができるが、ここでの議論は「関係にはそれぞれの属性に対応した現実の事象を表す組としてシステム要件上あり得るものだけが入る」という暗黙の制約が仮定されていることである。言い換えれば、正規形の定義では各属性の値

目次 [非表示]

SQLによるリレーションの操作のイメージ

トランザクション

- トランザクション
  - データベースにおいて「作業を完了するための一連の作業」の単位のこと

- トランザクション制御文
  - COMMITやROLLBACK

トランザクションの必要性

〈注文表〉

注文番号	販売日	顧客名	製品番号	注文数
1	15-02-20	A社	2	100
2	15-03-19	B社	1	40
3	15-08-04	A社	1	30
4	15-12-06	C社	2	50
5	16-06-15	D社	1	100

(1)INSERT

〈在庫表〉

製品番号	製品名	在庫数
1	A	200
2	B	300

(2)UPDATE

障害発生！

(1)注文番号5で、注文表へ新しい注文を追加
　→この時点でCOMMIT

(2)在庫表の製品Aの在庫数を減らす(UPDATE)前に障害が発生

→注文表と在庫表の数に不整合が発生する
　本来は在庫表の数は200-100＝100個であるべき

データの不整合を回避するための「ロック」

<在庫表>

(1)Aさん

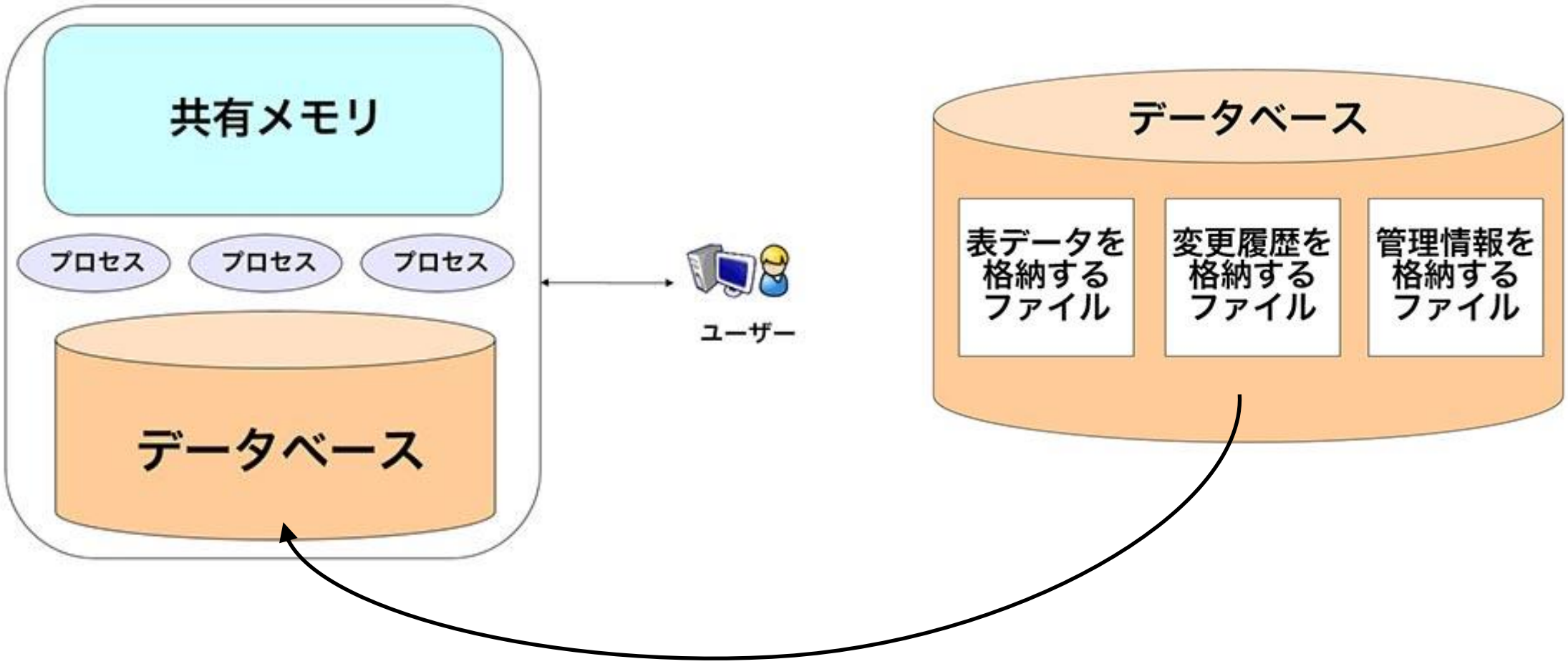
製品番号	製品名	注文数
1	A	300 → 200
2	B	100

(2)Bさん

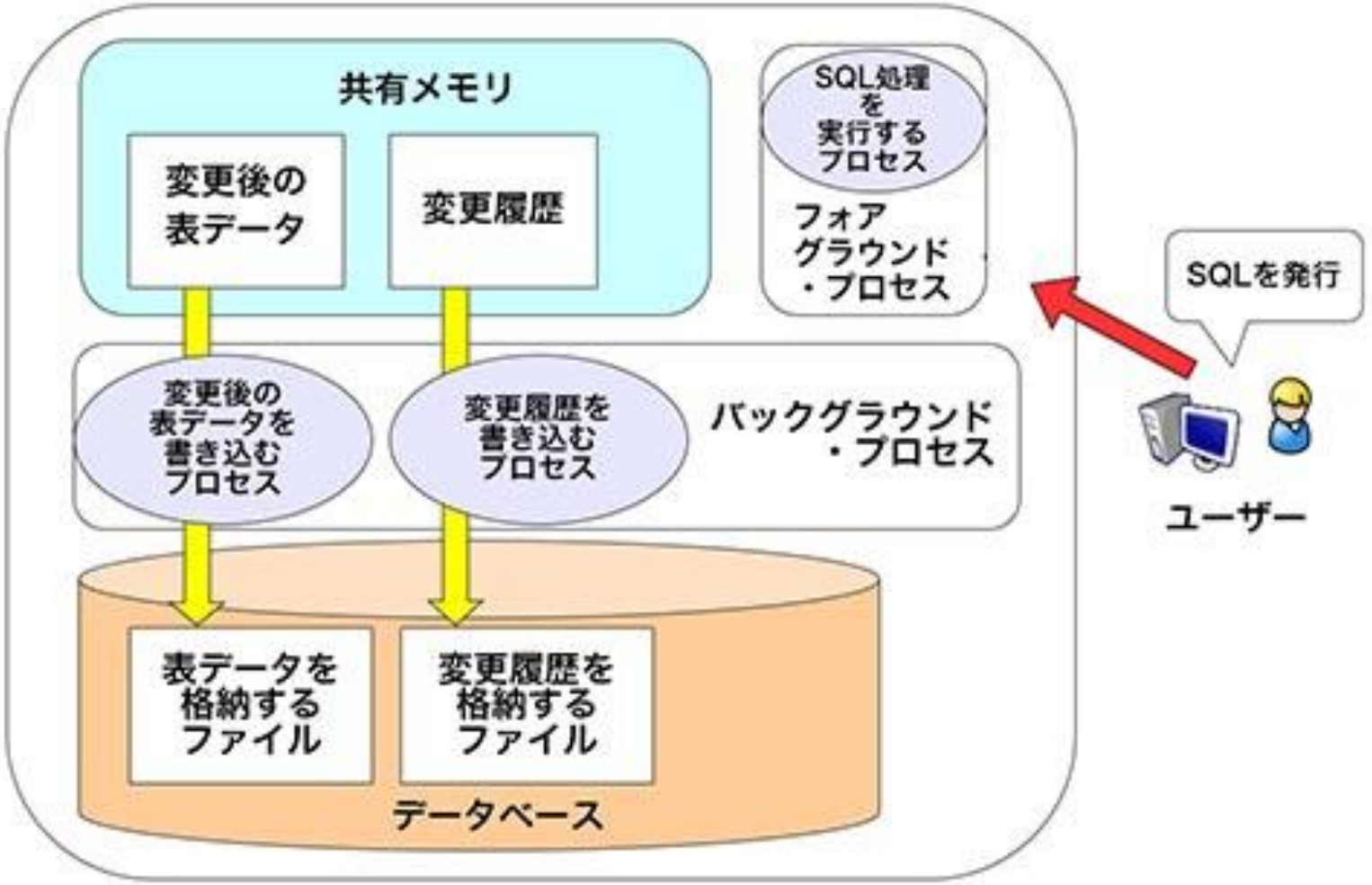
(1)ユーザーAが製品Aの在庫数を300から200に変更。
　このとき製品Aの行にはロックが取得される。

(2)ユーザーBが製品Aの在庫数を変更しようとしたが、
　ユーザーAがロックをかけているため待機状態になる。

リレーショナルデータベースの構成要素



基本的なプロセス

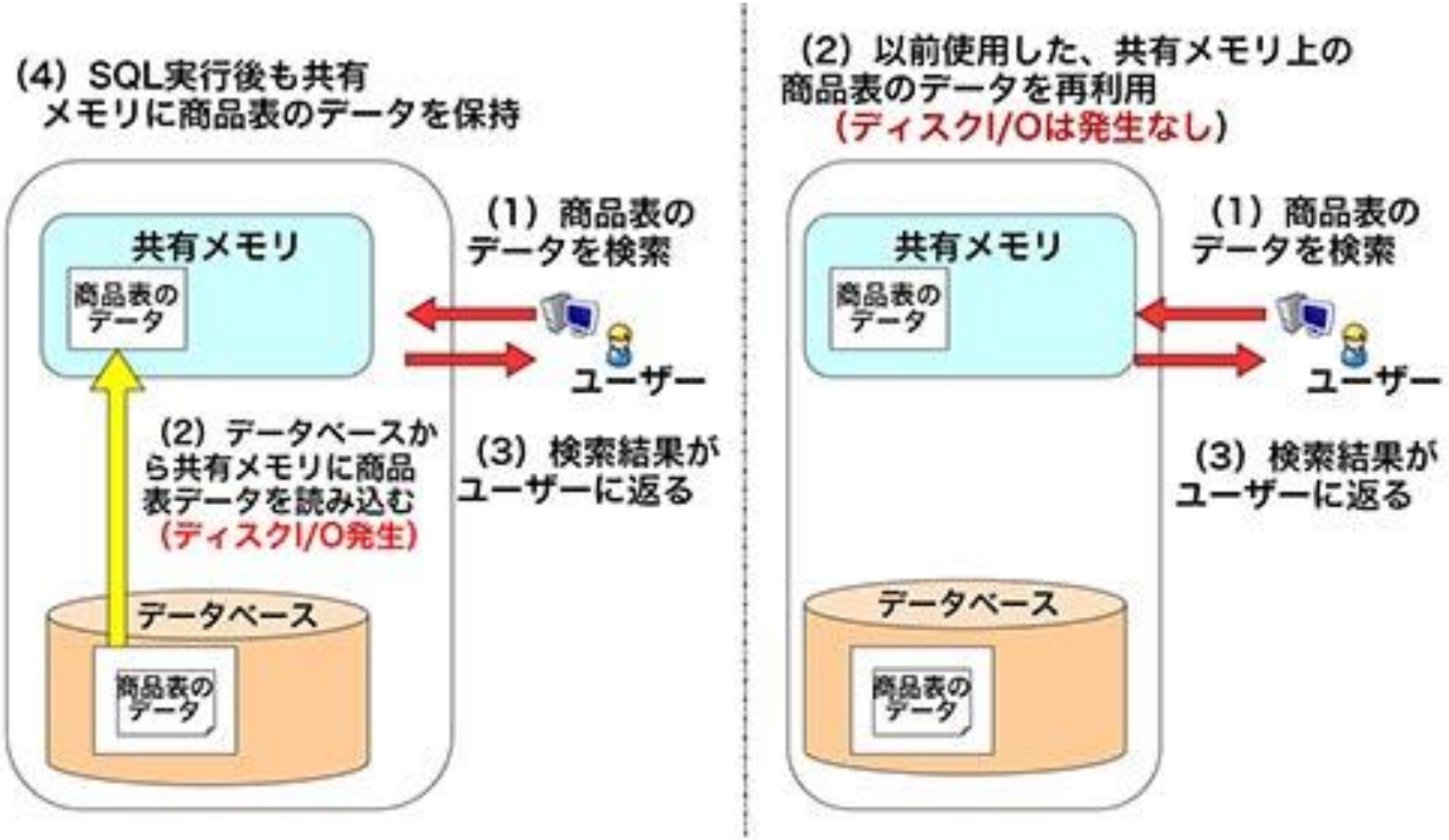

共有メモリ
変更後の
表データ
変更履歴
SQL処理
を
実行する
プロセス
フォア
グラウンド
・プロセス
SQLを発行
変更後の
表データを
書き込む
プロセス
変更履歴を
書き込む
プロセス
バックグラウンド
・プロセス
ユーザー
表データを
格納する
ファイル
変更履歴を
格納する
ファイル
データベース

共有メモリ上での処理

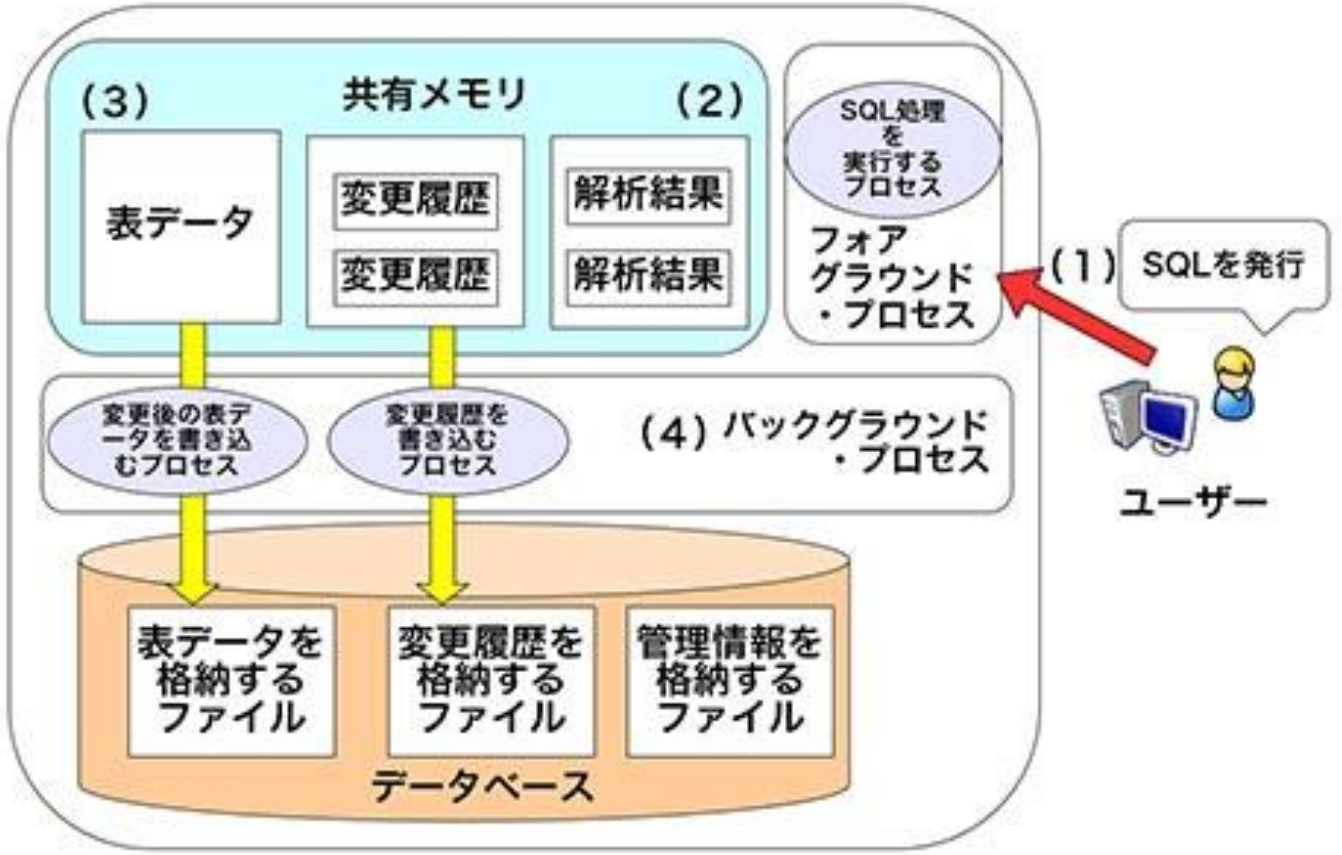
- 共有メモリ
  - SQL処理に必要なデータをユーザー間で共有利用するための領域

- 共有メモリ上での処理
  - ディスクI/Oを削減し、パフォーマンスの向上を目的とする
    - SQL処理を行う度に各ファイルへ個別にアクセスすると、ディスクI/Oが頻繁に発生し、パフォーマンスが低下する原因となる

メモリ上でのデータの保持／再利用の仕組み



RDBMSにおけるアーキテクチャ

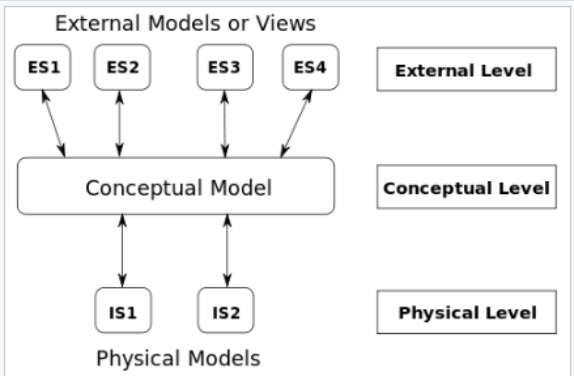


(1) ユーザーが発行したSQLを、フォアグラウンド・プロセスが受け取る
(2) 該当SQLの解析が行われ、共有メモリ上に保存される
(3) 共有メモリ上でSQLが処理される
(4) 変更履歴や変更済データは、随時データベースへ書き込まれる

3つの観点 [編集]

データモデルインスタンスは、1975年のANSIに沿った3つの種類の1つかもしれない[5]。

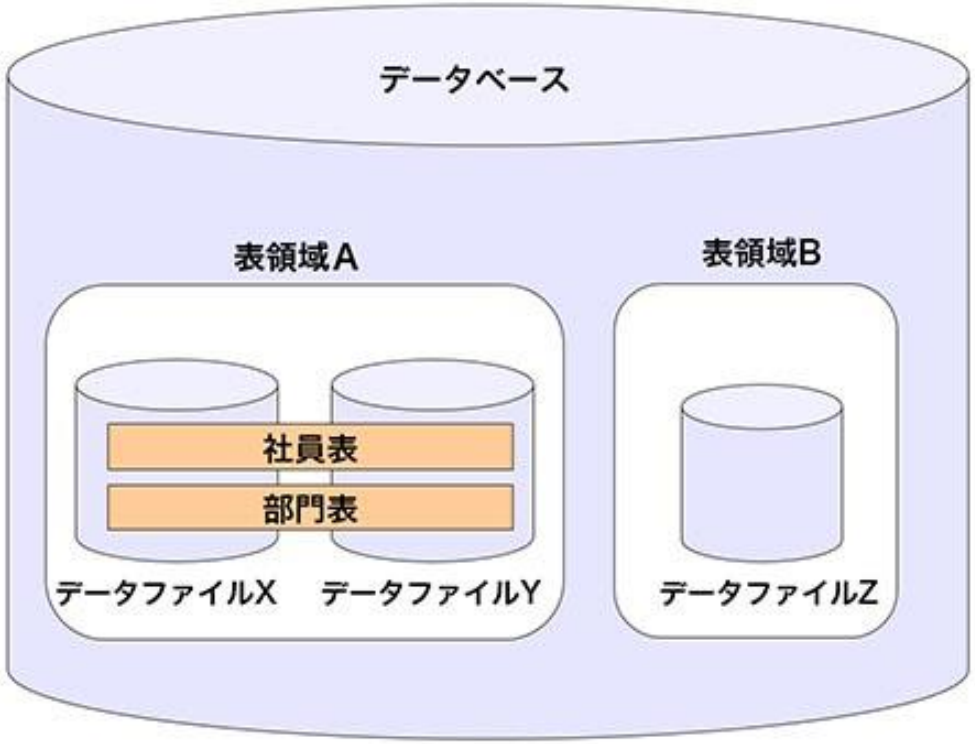
- 概念スキーマ：モデルのスコープである、1つのドメインの意味を記述する。たとえば、それは1つの組織あるいは産業の関心領域のモデルかもしれない。これは、そのドメインにおける重要なものの種類を表現するエンティティ・クラスと、一対のエンティティ・クラス間の関連について関連からなる。概念スキーマは、そのモデルを使って表されうる、事実と命題の種類を特定する。そのセンスで、それは、そのモデルのスコープによって限定される1つのスコープの、1つの人工的'言語'で許される表現を定義する。概念スキーマの利用は、事業ユーザーと共に強力なコミュニケーション・ツールとなるよう進化する。しばしば、「主題領域モデル（SAM）」または「ハイレベル・データモデル（HDM）」と呼ばれるこのモデルは、事業ユーザーが全体的アプリケーション開発または事業体イニシアティブの一部として、コア・データ概念、ルール、および定義をコミュニケートするのに使われる。オブジェクトのいくつかは、少なくかつ主要な概念に焦点を当てるべきである。大変大きな組織や複雑なプロジェクトのため、モデルは2ページ以上にまたがるかもしれないが、1ページにこのモデルを限定しようと試みる必要がある[6]。
- 論理スキーマ：特定のデータ操作技術によって表現されるような、意味論を記述する。これは、他のものの間の、テーブルおよびカラム、オブジェクト指向クラス、およびXMLタグの解説からなる。
- 物理スキーマ：データが格納される物理的手段を記述する。これは、パーティション、CPU、表空間、あるいはそのようなことに係わる。

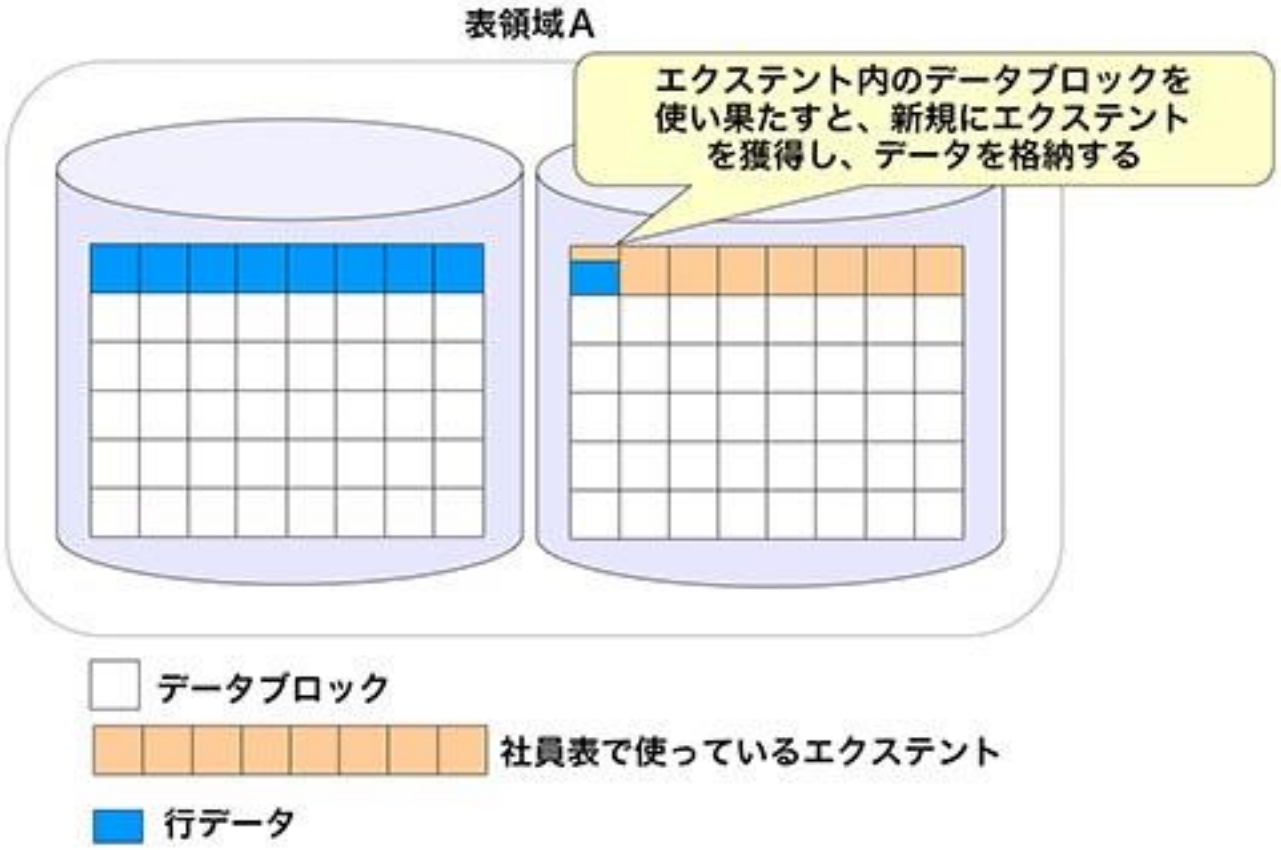


ANSI/SPARCの3層スキーマ・アプローチ（three level architecture）。これは、データモデルは1つの外部モデル（またはビュー）、概念モデル、あるいは物理モデルでありうることを示している。これは、データモデルを見る時の方法のみでなく、それは、特にモデルを比較するとき、1つの使い方の方法である[4]。

ANSIによれば、このアプローチの重要性は、3つの観点がそれぞれ相対的に独立であることを可能にすることである。格納技術は、論理的あるいは概念モデルのいずれにも影響することなく変更できる。テーブル／カラム構造は、概念モデルに（必要なら）影響することなく変更できる。いずれの場合も、もちろん、その構造は他のモデルとの一貫性を残さなければならない。テーブル／カラム構造は、エンティティ・クラスや属性の直接変換からは異なるかもしれないが、しかし、それは究極的に概念エンティティ・クラス構造の目的の外で扱わなくてはならない。多くのソフトウエア開発プロジェクトの初期段階は、概念データモデル（英語版）の設計を強調する。このような設計は、論理データモデル（英語版）で詳細化される。その後段で、このモデルは、物理データモデル（英語版）に変換されるかもしれない。しかしながら、概念モデ

論理構造のデータの格納

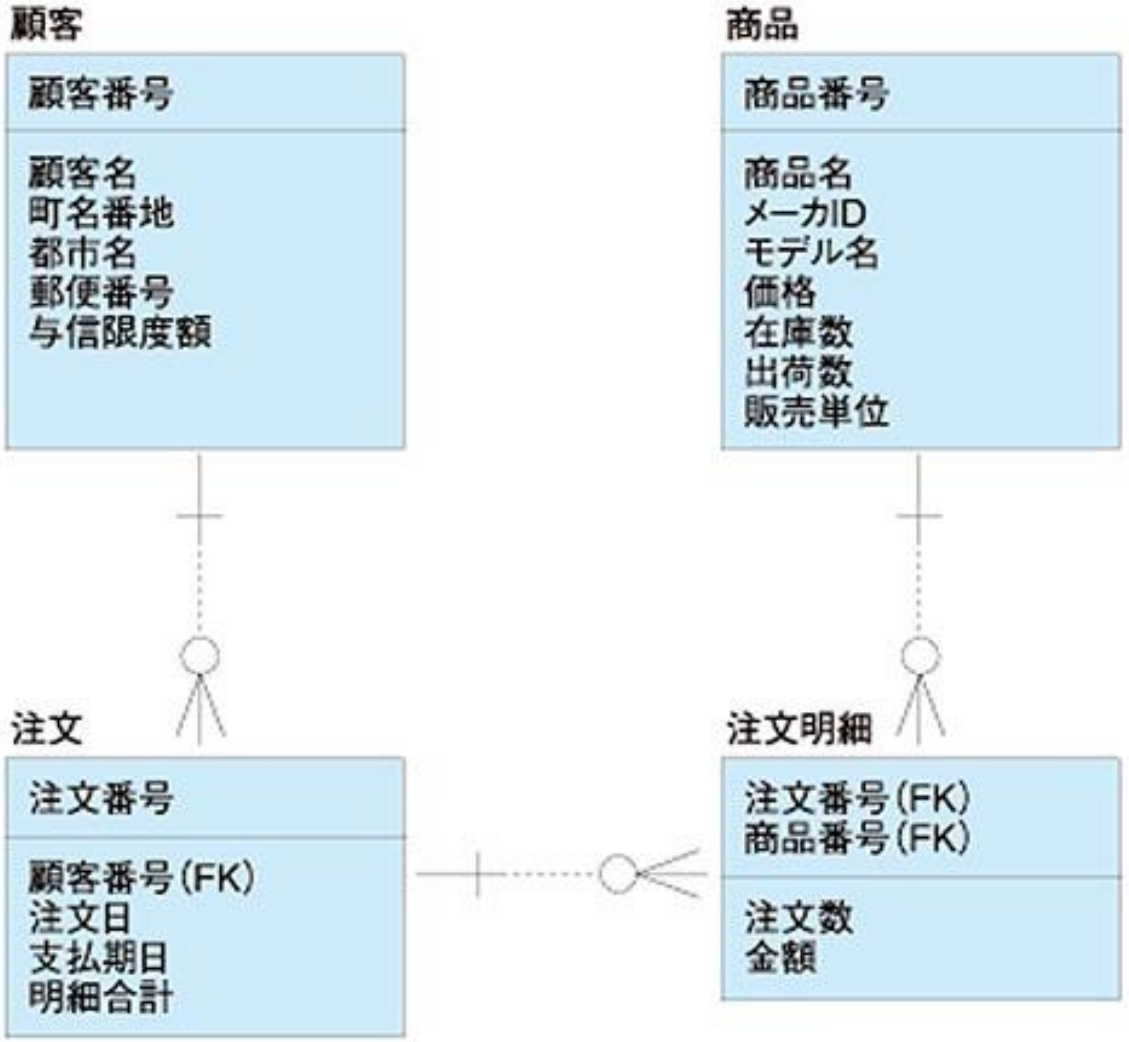




実体関連モデル（ERモデル）

- 実体（entity）
  - 実世界において識別可能な物体や事象
  - （グループ化されたデータ群）
  - 例えば、顧客、商品など
- 属性（attribute）
  - 定義域（domain）
  - （実体（関連）を定義するもの、または実体（関連）に所属するもの）
  - 例えば、顧客の場合、顧客名、住所、電話番号など
- 関連（relationship）
  - 実体間の関係
  - 例えば、会社に顧客が属する、など

ERモデルの例

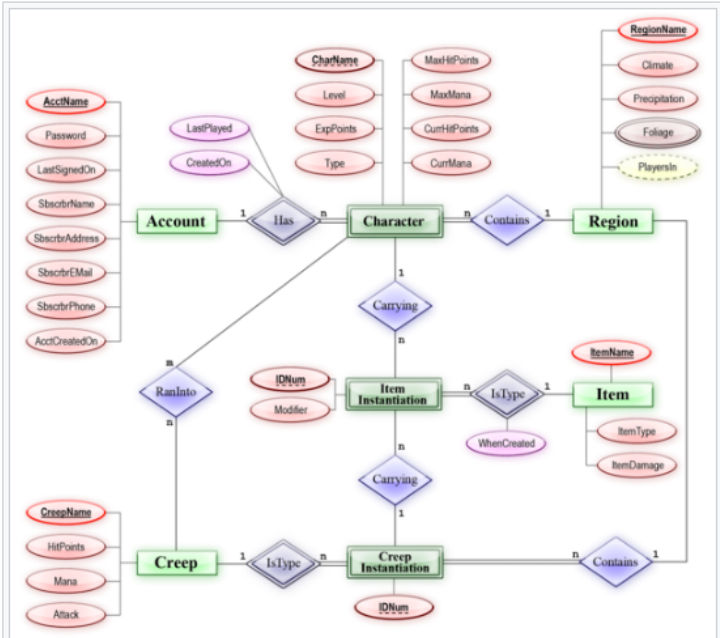

顧客
顧客番号
顧客名
町名番地
都市名
郵便番号
与信限度額
商品
商品番号
商品名
メーカID
モデル名
価格
在庫数
出荷数
販売単位
注文
注文番号
顧客番号(FK)
注文日
支払期日
明細合計
注文明細
注文番号(FK)
商品番号(FK)
注文数
金額

実体関連モデル

35の言語版

出典: フリー百科事典『ウィキペディア（Wikipedia）』

実体関連モデル（じったいかんれんモデル、英: entity-relationship Model、ERM）は、概念的データモデルの高レベルな記述を可能とするモデルの一種である。また、実体関連モデルによって具体的なシステムのデータモデルを図で表現したものを**実体関連図**（英: entity-relationship Diagram、ERD）あるいは**ER図**と呼ぶ。本項ではピーター・チェンの1975年の論文で提唱された技法を中心に解説する[1]。ただし、同様のアイデアはそれ以前から存在し[2]、実体と関連を扱う様々な派生モデルが考案されている。



チェンの記法を使った実体関連図の例

概要 [編集]

ERモデルは、データベース、特に関係データベースを抽象的に表現する手法の1つである。関係データベースは表にデータを格納し、表内の一部のデータは他の表内のデータを指している。例えば個人情報のデータベースでは、ある個人のエントリが当人のいくつかの電話番号のデータを指している構成になっていることがある。ERモデルでは、その個人のエントリが1つの「実体」[3]であり、電話番号も「実体」で、それら実体間に「一件の電話番号を持つ」という「関連」[4]が存在するという見方をする。そういった実体や関連を設計する際に作られる図が実体関連図またはER図である。

三層スキーマをソフトウェア開発に使用する場合、三層それぞれに対応するERモデルが構築される。

関係データベース管理システム

24の言語版

出典: フリー百科事典『ウィキペディア（Wikipedia）』

関係データベース管理システム（かんけいデータベースかんりシステム）または**リレーショナルデータベースマネジメントシステム**（英語: relational database management system、略称：**RDBMS**）は、関係データベース (RDB) の管理システムである。RDB がデータベースの種類を示すのに対して、RDBMS は RDB の実装を示す。標準問い合わせ言語として SQL を用いたアクセスを行うため、相対する言葉として RDBMS 以外のデータベースを意味する NoSQL (Not only SQL) がある。

2007年の時点では、広く知られていてまた広く使われているデータベースのほとんどは関係データベースであったが、その後NoSQLが発展したため必ずしもそうとは言えなくなっている。

SQLを扱うRDBMSが表を使って演算を行う事から、全く異なる用途の表計算ソフトと間違えられる事もあるが、RDBMSでは、データの形式は表に限定されておらず、表計算ソフトのように見た目に分かりやすい表を作る事が目的ではない。あくまでも、必要なデータを必要な時に素早く引き出して他のソフトウェアに提供する事が目的である。

関係データベースをオブジェクトデータベースと融合させたオブジェクト関係データベースなどといったものもある。その管理システムはオブジェクト関係データベース管理システムなどと呼ばれる (ORDBMS)。

商用のRDBMSとしてはOracle DatabaseやIBM DB2などが、オープンソースのRDBMSとしてはMySQLやPostgreSQLなどが、広く知られている。ただし、これらのDBMSを 真のRDBMSと呼んで良いのかどうかについては、後述のとおり、議論の対象となっている。