前書き

データベースという言葉を知ったのは，大学 3 年生の授業であった．当時，工学部情報科学科の学生であった私は，必修科目である「データベース」の講義を受けていたが，他の授業と同様，ただ漫然と講義を受けていた．単位を修得をしたものの，データベースの意義を理解していなかった．

その後，縁があってデータベース研究者として名を馳せた先生の研究室に入り，情報検索の研究活動を行っていたが... まさかデータベースの教科書を自分が書くとは思ってもいなかった．

目次

第 1 章

データベースを使わない世界

データベースと聞いて何を思い浮かべるだろうか．大抵の人は，データベースといえば「大きなデータの集まり」といったイメージを持つのではないだろうか．このイメージはあながち間違いではない．

さて，本講義のようにわざわざ科目を立ててまで，データベースについて学ぶことはあるのだろうか？結論としては，大規模データに携わる IT エンジニアやデータ分析者を目指す人であれば，「大いにあり」である．1960 年代から今日に至るまで，データベース技術は盛んに研究開発が行われてきた．大きなデータの集まりを扱うには，対処しなければならない問題が思った以上に数多く存在するのである．

本講義では 10 数回にわたってデータベース技術について解説するが，この第 1 講ではデータベースのことはいったん横に置いておいておく．今回は（割と）大きなデータを扱うときに遭遇する問題について考えてみよう．

1.1　ケース 1: 販売履歴の記録をはじめる

以下は，山畑さんという架空の人物のお話である．

> 山畑さんは家族で小さな小売店を営んでいる．個人経営ながら山畑さんのお店は繁盛している．とはいえ，街には大手チェーン小売店が進出してきており，このまま順調に経営を続けられるか，不安が募っている．何か手を打たなければならない．

2020 年の 4 月，山畑さんは念願のショッピングサイトを立ち上げた．言うまでもない．ショッピングサイトを立ち上げたのは，オンラインの場にも顧客獲得の機会を求めるためだ．サイトは順調に立ち上がり，注文もポツポツ入ってきている．

ところで，最近「データサイエンス」なるものが世間の注目を集めているらしい．データを活かせばビジネスチャンスが広がるとのことだ．山畑さんは，Excel シートに記録を取り始めた販売履歴を分析してみようと思い立った．

山畑さんが使っている Excel シートには，「いつ，誰が，何を，いくらで購入したか」の情報が記録されている．ショッピングサイトは立ち上がったばかりであり，Excel シートには 200 行しかデータが入っていない．しかし，今後データが貯まっていけば，売り上げを増やすための課題が見えるかもしれない．いずれがっつりとデータ分析をやるためにも，山畑さんは手持ちのデータを用いて分析の練習に取り組むことにした．

Q1. データの確認

こちらの URL（`https://dbnote.hontolab.org/data/purchase_small.xlsx`）から，上のケース 1 で山畑さんがデータ分析の練習に使おうとしている Excel ファイル（`purchase_small.xlsx`）をダウンロードし，中身を確認しなさい．

なお，Excel シートの各列の意味は以下の通り：

- purchased_at: 購買（販売）日時
- customer: 商品を購入した人物の氏名
- gender: 商品を購入した人物の性別
- product: 購入された商品名
- sale: 販売価格

Q2. あの人は何回買い物をしている？

「岡田 真綾」という人物が何回買い物をしていたかを数えなさい．

Q3. 商品 X を購入しているのは誰？

Excel のオートフィルタ機能を使って,「ビタミン補助剤」を購入している人をリストアップしなさい.

Q4. 総売上金額

Excel 関数の `SUM` を用いて，現時点での総売上金額を計算しなさい.

Q5. 最も売れた商品は？

Excel のピボットテーブル機能を使って，集計期間中に

- 最も購買回数が多かった商品
- 最も売上金額の合計が大きかった商品

をそれぞれ求めなさい.

1.2　ケース 2: サイトの認知度向上につき，得られるデータも膨大に!?

現時点では手持ちのデータは少ないものの，販売履歴データの分析に将来性を感じた山畑さん．販売履歴データを有効活用できるよう，ショッピングサイト運営により力を入れる決意を固めたのであった.

以下は，山畑さんのその後の話（架空の話）である.

> ショッピングサイト立ち上げ以降，順調に利用者数も増えていった．やはりメディアに取り上げられたのが大きかったのだろう．あのタイミングでサイトの認知度が一気に高まり，サイトの利用者数や利用頻度も加速度的に増えていった．それに伴い，サイト運営に関わるスタッフも増員した.
>
> 販売履歴の管理は，当初は山畑さんが一人で担当していたが，さすがに一人では対応しきれなくなった．そこで，ある時点から数名体制で販売履歴の記録を行うことになった．これまで販売履歴の管理に使ってきた Excel シートをクラウドストレージに置き，記録担当スタッフの PC 間で同期を取る仕組みを

導入．同じ Excel ファイルの上で，スタッフ全員で販売履歴を記録できるようにしたのである．

2 年後．サイト事業は軌道に乗った．十分な量の販売履歴データが蓄積されたと判断した山畑さんは，いよいよ大規模な販売履歴データの分析に取りかかることを決意した．立ち上げ当初は 200〜300 行しかなかった Excel シートであったが，シートを開きその行数を数えてみると…なんとその数 90 万行以上! データの量に小躍りした山畑さんは，Excel シートの扱いに詳しいスタッフと共に，意気揚々とデータ分析に取りかかったのであった．

Q6. データの再確認

こちらの URL（`https://dbnote.hontolab.org/data/purchase_large.xlsx`）から，上のケース 2 で山畑さんが分析しようとしている Excel ファイル `purchase_large.xlsx` をダウンロードしなさい．またダウンロードしたファイルを用いて下記課題（演習 1 と同じ）に取り組み，データ分析上の課題（困ったこと）を議論しなさい．以下，課題 1 の内容を再掲する．

- 「岡田 真綾」という人物が何回買い物をしていたかを数えよ
- 「ビタミン補助剤」を購入している人をリストアップせよ
- 総売上金額を計算せよ
- 集計期間中に「最も購買回数が多かった商品」「最も売上金額の合計が大きかった商品」を求めよ

もし，`purchase_large.xlsx` ファイルがうまく開けない場合は，こちらの URL（`https://dbnote.hontolab.org/data/purchase_medium.xlsx`）からダウンロードできる `purchase_medium.xlsx` を用いなさい．なお，ダウンロードできる Excel シートの構造はケース 1 で用いた `purchase_small.xlsx` と同じである．

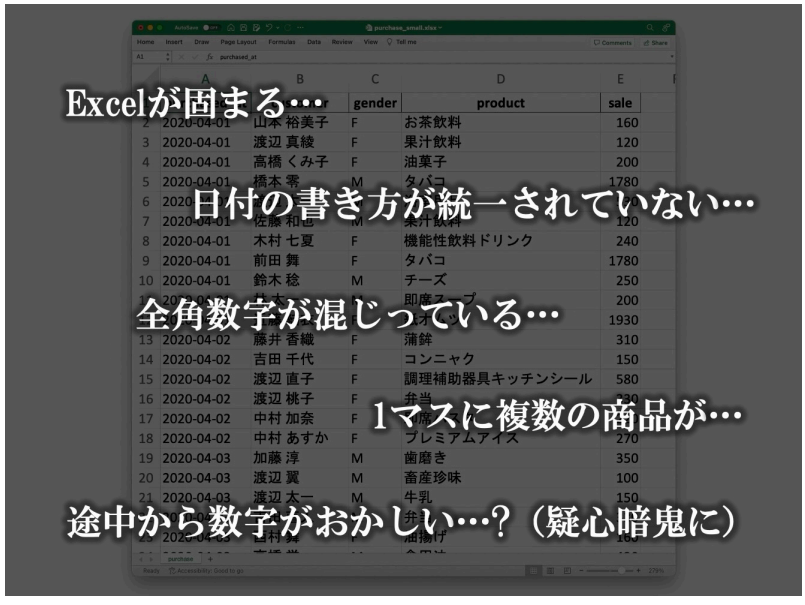

図 1.1: 大きな表データを複数人で Excel で扱うときの悲劇.

1.3　表計算ソフトでデータ管理を行ったときに起きる悲劇

ケース 1 および 2 で用いた Excel ファイルは，販売履歴データの集まりであった．一般的な認識からすると，このようなデータの集まりは「データベース」ということになるだろう．

ところで，上記演習，とりわけケース 2 に取り組んでみてイライラしなかっただろうか？数万件，数十万件ある表データを Excel で扱おうとすると，さまざまな不都合が生じる（図 1.1）．これは，本来 Excel は個人用の表計算アプリケーションであって，大規模データの管理や処理を前提として設計されていないためである．

では，大規模なデータを管理・処理するためにはどうすればよいだろうか？そのための技術こそが「データベース」である．以降，本書では**大規模データを効率よく管理・処理するための「データベース」技術** について学習する．

第 2 章

データベースの概念

今日，データベースは様々な業務やアプリケーションで利用されている．例えば，Amazon[*1]や楽天市場[*2]といったオンラインショッピングサイトにおいては，購買履歴管理や在庫管理を行うためにデータベースが用いられている．Instagram[*3]や Tiktok[*4]といった，多数のユーザがコンテンツを投稿するソーシャルメディアにおいてもデータベースは欠かせない．個人用ブログのような比較的小規模なウェブサイトでも，記事データを管理するためにデータベースが用いられている（例: WordPress[*5]．データを語る上で，データベースはなくてはならないものである．

本章では，データベースの概念を説明した後，最も代表的なデータベースであり，かつ本書の主要テーマでもある**関係データベース**の概要について述べる．

2.1 データベースとは

データを収集・利用する文化が根付いてきたこともあり，データベースという言葉が市民権を得つつある．しかし，「データベース」という語が指す意味は，一般人と IT 屋とは大きく異なる．一般人にとってのデータベースとは，「データの集まり」くらいの意味である．一方，IT 屋にとってのデータベースは，

[*1] Amazon. `https://www.amazon.co.jp/`
[*2] 楽天市場. `https://www.rakuten.co.jp/`
[*3] Instagram. `https://www.instagram.com/`
[*4] Tiktok. `https://www.tiktok.com/`
[*5] WordPress. `https://ja.wordpress.org/`

> “複数の応用目的での共有を意図して，組織的にかつ永続的に格納されたデータ群（北川博之著「データベースシステム」[1] より）”

あるいは

> “データの正しさを管理する主体によって体系的に整理され，計算機によって永続的に格納されたデータの集まり（吉川正俊著「データベースの基礎」[2] より）”

を意味する．特に「データの正しさを組織的に管理する」という概念が重要である．

データベースを扱うためのシステムは，**データベース管理システム（database management system, DBMS）**と呼ばれる．本来データベースとは DBMS によって管理されるデータ群を指すが，DBMS そのもの（あるいは DBMS とそれによって管理されるデータ群）をデータベース（DB）と呼ぶこともある．

増え続ける多様かつ大量のデータと付き合っていくためには，データのうまい管理・処理方法が必要になる．場当たり的にデータを作っては Excel シートやフォルダ（ディレクトリ）に突っ込んでおくというやり方は，扱うデータが増え，データを使う人間やアプリケーションが増えるにつれて，やがて破綻する．IT 屋が考えるデータベースは，こういった事態を未然に防ぐための強力なツールである．

2.2　データ処理の際に求められる機能

大量のデータを管理・処理する際には，以下のような要件が求められる．

多様かつ大規模なデータの管理

ビジネスをはじめとする現場では，多種多様かつ大量のデータが時々刻々と発生している．例えば，2020 年 1 月 1 日から 2020 年 12 月 31 日までの 12 か月間に，Amazon.co.jp で購買された商品の数は 5 億点以上にのぼるとされている [3].

小規模な表データであれば，Excel のような表計算ソフトでも対応できる．しかし，データの規模が大きくなり，さらに表データの登録・更新を担当する人員が増えていくと…データの管理が破綻するのは想像に難くない（そもそも Excel は 1 つの表につき最大 100 万行程度しか扱えない）．Excel のようなスプレッドシートでは，購買データのように多様かつ大規模なデータを効率よく集積，管理することは

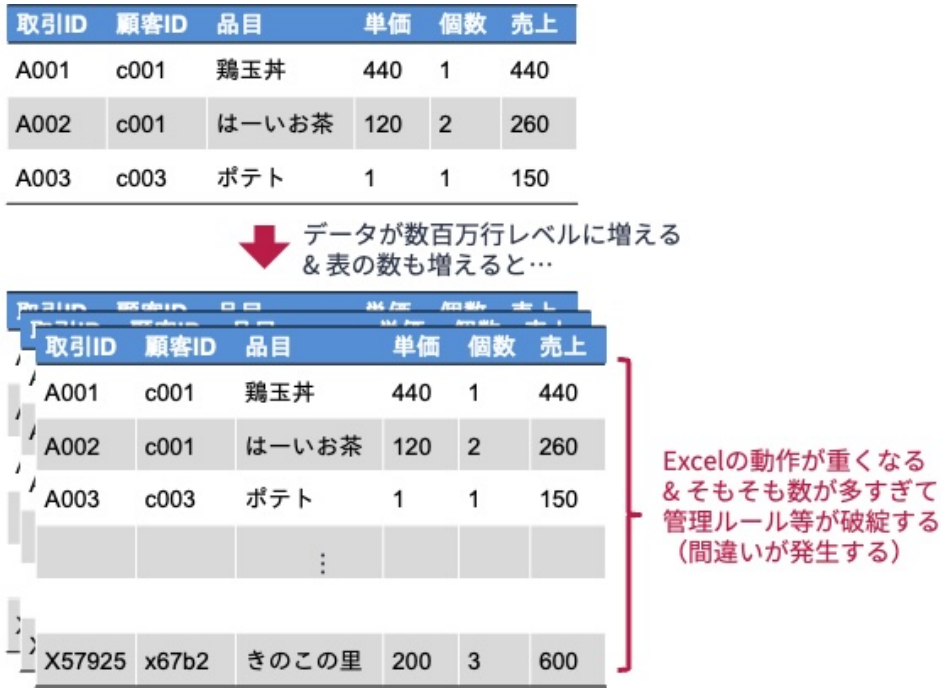

図 2.1: 管理する表データの規模や数が増えると破綻する．

困難極まりない（図 2.1）．

データの正しさの保証

管理するデータが大量かつ多様になってくると，データを正しく保つことが難しくなってくる．管理対象となるデータに誤りが混入すると，大変なことになる．例えば，図 2.2 のように，購買データの中に

- 現実にはあり得ない売り上げ金額が記録されている
- 入力形式が異なる日付が混在している
- 取扱商品リストにないはずの商品が購買履歴に含まれている

といったことが起きると，オンラインショッピング事業においては一大事である．それゆえ，データ管理においては，格納されるデータの正しさ，データ間の矛盾のなさを保証する機能が求められる．

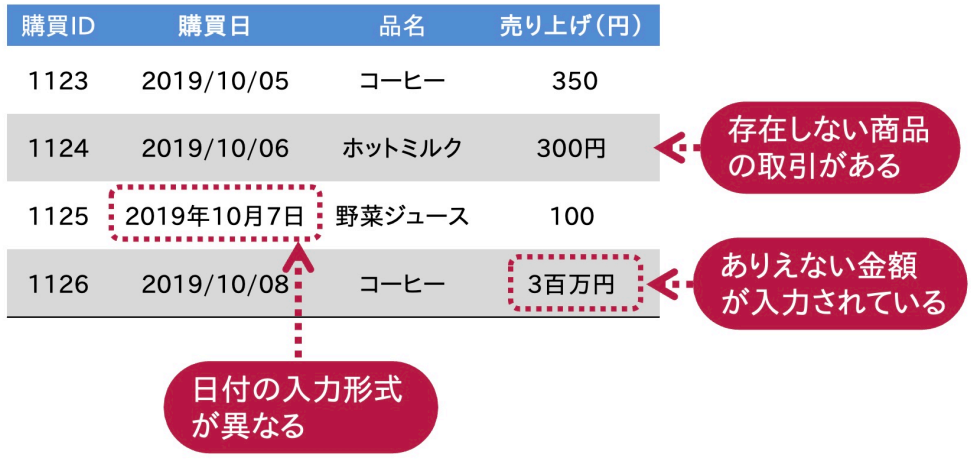

図 2.2: バッドデータ（not ビッグデータ）の例.

高速で効率的なデータ処理

データが大量に格納できたとしても，対象となるデータを高速に処理できなければ使い物にならない．たとえ数百万件の書籍情報を格納しているシステムがあっても，ニーズを満たす書籍リストの検索結果を出力するのに数分待たされるようでは，そのようなシステムは使われない．

ユーザが増えるに従って，システムにかかる負荷も増える．例えば，Amazon.co.jp では毎分平均 900 件以上の商品取引が行われている [4]．このように大量のデータのやりとりが発生するケースにおいても，高速にデータ処理されることが求められる（図 2.3）.

同時実行（並列処理）

関連して，大勢の人が同時にデータを検索，追加，更新，削除しても，システムが問題なく動作することも重要である．

例えば，X さん，A さん，B さんがとあるネット銀行の口座を利用しているとしよう．X さんの口座残高は 100 万円であるとする．振り込みがあった際，振込先口座の残高の計算は

$$\text{残高} = \text{振り込み時の「振込先口座」の残高} + \text{振込額}$$

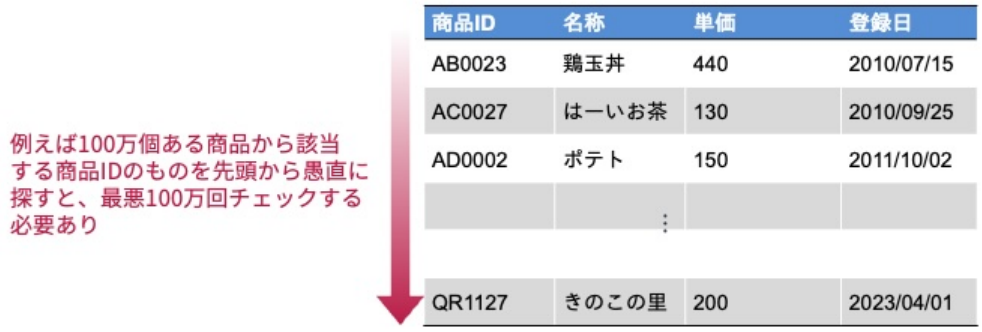

図 2.3: 線形探索では大規模データに対する高速なアクセスは難しい.

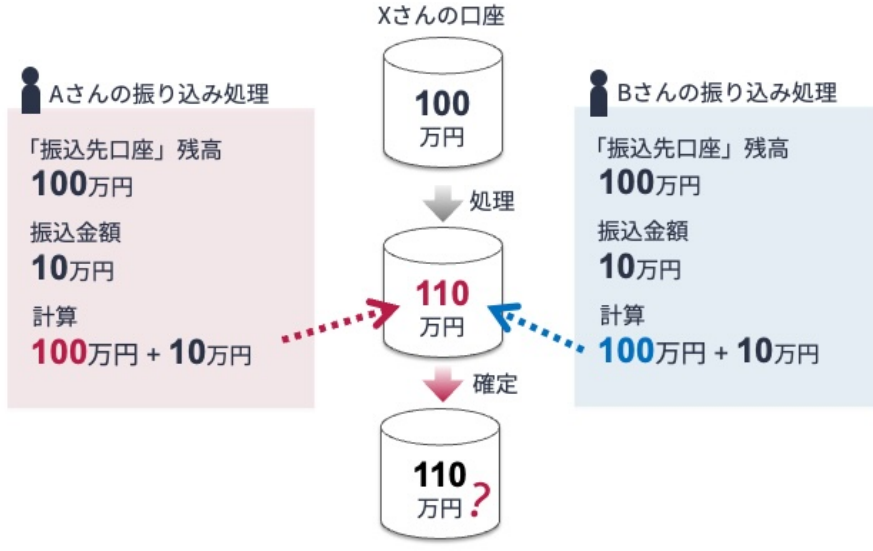

図 2.4: 複数の処理要求を矛盾なく処理できていない例.

振込元口座の残高の計算は

残高 = 振り込み時の「振込元口座」の残高 - 振込額

となる．これはは至極当然の処理であるが，次のような状況を考えてみよう．

ある日，A さんと B さんが X さんの口座に一秒の狂いもなく，**まったく同時**に 10 万円送金したとする．この際，前述の計算式をそのまま適用すると，図 2.4 のように誤った処理が行われてしまう．A さんと B さんが振込みを行った後，X さん

の口座の残高は 110 万円になりました – こんなことが起きたら大変である．この例における問題点は，たとえ振込み処理が同時に発生したとしても，A さんの振り込み処理を待ってから B さんの処理を行うべきであったという点である．

このように，データ処理の内容によっては，別の処理が完了したことを保証してから次の処理を行う必要がある．さらに，処理の途中で何らかのエラーが起きた場合は，すべての処理をキャンセルして最初の状態に戻すことが求められる．同時にアクセスがあった場合でも，データを矛盾なく処理できることが重要である．

アクセス権限のコントロール

データによっては，誰でも自由に閲覧してよいものもあれば，特定の立場のユーザしかアクセスできないようにすべきものも存在する．多種多様なデータのやりとりが発生する環境においては，ユーザの属性ごとにデータの閲覧，作成，更新，削除といったアクセス権限を制御する機能が必要となる．

2.3　データベースを用いるメリット

一人で扱えるほどデータの規模が小さければ，Excel などの表計算用のソフトウェアでデータ管理しても問題はない．しかし，扱うデータが多様かつ大規模になると，要求されるデータ処理の質が変わる．そのため，データの管理方法や処理方法を見直す必要がある．

本稿で学ぶデータベースを用いれば，**データの一元管理**が可能となり，データの管理や利活用において様々な恩恵を受けられる．具体的には，データベースが備える以下のような特性あるいは関連技術を用いることで，前節で述べた「データ処理に求められる要件」をクリアすることができる．

- 大規模なデータの管理：「**物理的データ格納方式**」の工夫によって対応
- データの正しさの保証：「**一貫性制約**」によって対応
- 高速で効率的なデータ処理：「**索引づけ**」「**問い合わせ最適化**」などで対応
- 同時実行，データの共同利用：「**トランザクション**」などで対応
- アクセス権のコントロール：「**ロール管理**」によって対応

2.4 データのモデリング

モデリングとは？

科学やビジネスの世界では，モデルあるいはモデリングという用語がしばしば登場する．ビジネスプロセスモデル，物理モデル，統計モデルなど，世の中には様々なモデルが存在する．モデルとは，複雑な仕組みや現象，状態などを表現・分析・操作しやすくするために，本質的でない要素を取り除き，関心のある側面のみを抽出し抽象化したものである．**モデリング**とはモデル化，つまりモデルを作る行為である．

データモデリング

データベースで何らかのデータを扱う場合，まずデータをモデリングする．世の中に存在するデータは多種多様であり，データを統一的に整理し，計算機で処理しやすい形に抽象化，すなわちモデリングする必要がある．

データモデリングとは，データベース化すべき情報を取捨選択し，対象とするデータとその操作に関する枠組み（**データモデル; data model**）を設計する行為である．対象とする事象やアプリケーションに応じて適切なデータモデルを設計することで，データモデルに従って実データを格納し，操作することが可能となる．

一般に，データモデルは以下の 3 つの要素を含む：

- データの構造
- データの制約条件
- データの操作

本講義の主要テーマである**関係データベース（relational database）**は，**関係データモデル（relational data model）**[*6]に基づき設計されたデータベースである．関係データモデルの詳細については，次章で説明する．

データベース管理システムで扱われるデータモデルとしては，関係データモデルのほかにも以下のようなものがある：

- ネットワークモデル [5]

[*6] 関係モデル（relational model）と呼ぶこともある．

- 階層型データモデル [6]
- オブジェクト指向モデル [7]
- キー・バリュー（key-value）モデル [8]
- グラフデータモデル [9]

2.5　関係データモデル（導入）

関係データモデルは，最も代表的なデータモデルである．1970 年に Edgar F. Codd 氏により提案されたもので，単純ながらその背後には強力な数学的基盤をもつ [10]．関係データモデルでは，あらゆるデータを**表（table）**としてモデル化する．

図 2.5 は，関係データモデルを用いて構築された，授業の履修状況・成績を管理する関係データベースの例である．この関係データベースには，以下の 3 つの表[*7]が存在する．

- 学生テーブル：学籍番号，氏名，入学年度，所属といった学生に関する情報を格納
- 科目テーブル：科目 ID，科目名，開講年度といった科目に関する情報を格納
- 履修テーブル：どの学生が何の科目を履修し，どのような成績であったかに関する情報を格納

この例だけ見ると「なんだ，関係データモデルとは単なる表なのか」と思われたかもしれないが，**ただの表ではない**．関係データモデルに基づいて表現された（表）データは，あらかじめ定義された「データの構造」「データの制約」「データの操作」に関する規則に従ってデータが作られ，関係データベース内に格納される．例えば，以下のような規則が考えられる：

- 学生テーブルは「学生 ID」「姓」「名」「入学年度」「所属」という見出しをもつ
- 履修テーブルには，科目名や学生の氏名を格納しない
- 履修テーブルの成績には「優」「良」「可」「不可」のいずれかしか登録でき

[*7] テーブルと呼ぶこともある．

学生

学生ID	姓	名	入学年度	所属
s00001	川澄	桜	2023	A学部
s00002	山畑	滝子	2024	B学部
s00003	田辺	通	2024	C学部
		⋮		

科目

科目ID	科目名	開講年度
c0001	線形代数	2023
c0002	線形代数	2024
c0003	統計学入門	2024
	⋮	

履修

科目ID	学生ID	成績
c0001	s00001	不可
c0002	s00001	良
c0002	s00002	優
c0003	s00003	可
	⋮	

図 2.5: 関係データベースの例.

ない

- 履修テーブルに現れる科目 ID および学生 ID は，必ず学生テーブルと科目テーブルに存在する
- 学生テーブル，科目テーブルの各テーブルにおいて，同じ科目 ID は存在しない

このような規則に従い，授業の履修状況や成績を管理するための情報が，複数のテーブルに分割され格納される．

次章以降では，

- このような規則，すなわち関係データモデルをどう設計するのか
- 一見ただの表にしか見えない関係データモデルが，どのように数学的に定式化されているか
- 関係データモデルを用いると，なぜ大規模データの管理が効率的になるのか

などについて詳しく述べていく．

2.6　クイズ

Q1. メジャーなデータベース管理システム

ウェブ検索エンジンを用いて，世の中にあるメジャーなデータベース管理システムを調べなさい．また，調べたデータベース管理システムを「関係データベースを扱うもの」と「そうでないもの」に分類せよ．

Q2. 線形探索

100 万件ある商品リストの中に特定の商品が含まれているかを確認したい．商品リストの先頭から末尾まで順に商品名を確認していくと，平均で何回（何件）の確認で商品の有無を確認できるか？

Q3. データベース管理システムの利用例

普段利用しているサービスのうち，データベース管理システムを用いていると思われるものを 3 つピックアップしなさい．

Q4. CSV/TSV ファイル

CSV ファイルおよび TSV ファイルとは何かを調べなさい．また，（データベース管理システムでデータを管理する場合と比較して）CSV/TSV ファイルに存在する欠点を挙げなさい．

Q5. チューリング賞

関係データモデルを提唱した Edgar F. Codd 氏は，計算機科学分野のノーベル賞といわれるチューリング賞の受賞者である．Codd 氏以外で，データベースに関する功績でチューリング賞を受賞した人をピックアップしなさい．

第 3 章

関係データモデル

関係データベース（relational database）とは，関係データモデルにもとづき表現されたデータの集まりである．前章で述べたように，関係データモデルとは，表によってデータを表現するデータモデルである．一見すると単純な表にしか見えない関係データモデルは数学によって定式化されており，強固な理論的基盤を有している．関係データモデルを用いることで，データから冗長性を排除し，データを正しく管理することが可能となる．

以下では，数学的な観点から関係データモデルについて述べる．少々堅い説明にはなるが，関係データベースという計算機科学技術が数学に支えられていることを知るためにも，眠気を我慢して読んでもらいたい．なお，本講では数学における「集合と写像」の知識を使う．

3.1　関係データモデルのデータ構造

表 3.1 は，ある小売店で取り扱っている商品のデータを関係データモデルによって表現したものである．この表において，各行は小売店で取り扱っている商品に対応し，各列には各商品に関するデータが記述されている．表の左上にある「商品」はこの表の名前を示しており，**関係名**と呼ばれる．関係データモデル上では，各行は**タプル（tuple）**と呼ばれる．表の 1 行目には，「商品」がどのようなデータを持つかを示す見出しが付けられている．各見出しのことを**属性（attribute）**，各タプルにおけるある属性の値を**属性値**と呼ぶ．例えば，表中の 2 行目に対応するタプルは

表 3.1: ある小売店の商品に関するデータを収めた関係

商品

商品 ID	名称	単価	登録日
P1	はーいお茶	130	2020/07/15
P2	午前の紅茶	130	2020/09/25
P3	健康麦茶	150	2021/02/16
⋮	⋮	⋮	⋮
P1000	きのこの里	200	2023/01/08

$$(\text{“}P1\text{”}, \text{“はーいお茶”}, 130, \text{“2020/07/15”}) \tag{3.1}$$

であるが，このタプルの属性「単価」の属性値は 130 となる．

3.1.1　数学における「関係」

ところで，関係データベースあるいは関係データモデルの「関係」とは一体何だろうか．**関係（relation）** とは数学上の概念である．集合 $S_1, S_2, ..., S_n$ が与えられたとき，それらの直積集合 $S_1 \times ... \times S_n$ とは

$$S_1 \times ...S_n = \{(x_1, ..., x_n) \mid x_1 \in S_1, ..., x_n \in S_n\} \tag{3.2}$$

と表現されるものである．（数学上の概念である）関係とは，このような直積集合の部分集合を意味する．正確に書くと，集合 $S_1, S_2, ..., S_n$ が与えられたとき，直積集合 $S_1 \times ... \times S_n$ の部分集合を $S_1, S_2, ..., S_n$ 上の **n 項関係**と呼ぶ.

例えば，$A = \{2, 3\}$，$B = \{-2, -3\}$ が与えられたとき，直積 $A \times B$ は集合 A および B の要素のすべての組み合わせであるから

$$A \times B = \{(2, -2), (2, -3), (3, -2), (3, -3)\} \tag{3.3}$$

となる．そして，その部分集合の 1 つである

$$\{(2, -3), (3, -2)\} \tag{3.4}$$

は集合 A および B 上の 2 項関係となる．このような（n 項）関係の概念を使って，関係データモデルは理論構築されている

3.1.2 関係データモデルにおける「関係」

関係データモデルにおける関係は「数学におけるの関係」を拡張したものになっており，以下の 2 つから構成される．- 関係スキーマ- インスタンス

関係スキーマ（relation schema）とは，関係の名前と属性の集合，および（後述する）一貫性制約の情報を示すものである．関係スキーマは

$$(\boldsymbol{R}(A_1, ..., A_n), \{\sigma_1, \sigma_2, ..., \sigma_m\}) \tag{3.5}$$

の形式で記述される．ここで

- $\boldsymbol{R}$ は関係名
- $A_1, ..., A_n$ は属性
- $\sigma_1, ..., \sigma_m$ は一貫性制約

に対応する．一貫性制約が自明な場合やそれを考慮しないときは，関係スキーマを

$$\boldsymbol{R}(A_1, ..., A_n) \tag{3.6}$$

のように簡潔に表記することもある．先の表「商品」の例の場合，関係スキーマは

$$\text{商品（商品 } ID, \text{名称, 単価, 登録日）} \tag{3.7}$$

と記述する．

関係スキーマに記された各属性には，属性が取り得る値の集合を定義することになる．この集合は**ドメイン（domain; 定義域）**と呼ばれ，属性 A のドメインは $Dom(A)$ と記す．例えば，先の表「商品」の例では，属性「単価」は金額を表すので，そのドメイン Dom(単価) は 0 以上の整数値が想定される．これを数学的に記すと，以下のようになる:

$$Dom(\text{単価}) = \{x \mid x \in \mathbb{N}\} \tag{3.8}$$

同様に，属性「商品 ID」「名称」「登録日」についても，以下のようにドメインを

定義できる．

$$Dom(\text{商品 } ID) = \{x \mid x \in P \text{ から始まる文字列集合 }\} \tag{3.9}$$

$$Dom(\text{名称}) = \{x \mid x \in \text{文字列集合 }\} \tag{3.10}$$

$$Dom(\text{登録日}) = \{x \mid x \in \text{年月日集合（ただし西暦表記）}\} \tag{3.11}$$

インスタンス（instance） は，関係スキーマ $\boldsymbol{R}(A_1, ..., A_n)$ におけるドメイン $Dom(A_1), ..., Dom(A_n)$ の直積の部分集合である．仮にインスタンスを R とすると，

$$R \subset Dom(A_1) \times ... \times Dom(A_n) \tag{3.12}$$

である．インスタンスは，上で説明した数学上の関係に相当する（あらためて「数学における関係」の節を確認してみよう）．また，インスタンス R の要素がタプルとなる．

例に戻って，定義と照らし合わせてみよう．先の表「商品」の関係スキーマは商品 (商品 ID, 名称, 単価, 登録日) であった．各属性のドメインの定義も与えたが，その定義から，例えば属性「単価」については

$$Dom(\text{単価}) = \{1, 2, 3, ...10000, ...\} \tag{3.13}$$

のようにあらゆる自然数を取り得る．また，属性「名称」については

$$Dom(\text{名称}) = \{\text{“あ”}, \text{“い”}, ..., \text{“はーいお茶”}, ...\} \tag{3.14}$$

のようにあらゆる文字列を取り得る．

各属性の直積 $Dom(\text{商品 } ID) \times Dom(\text{名称}) \times Dom(\text{単価}) \times Dom(\text{登録日})$ には，

$$(\text{“}P001\text{”}, \text{“こんな商品はありません”}, 100 \text{ 兆}, \text{“2020/01/01”}) \tag{3.15}$$

のように，実際に商品としてありえないタプルも含めて，様々なタプルが要素として入っている．なぜなら，直積とは集合（ドメイン）の要素のすべての組み合わせであるためである．ドメインの直積の部分集合をインスタンスとすることは，取り得るタプルの候補から実際にデータとして扱われるタプルの集合を選択することに相当する．

3.2　非正規関係と第 1 正規形

関係データモデルにおけるドメインとは属性が取り得る値の集合を意味するが，ドメインの要素はそれ以上分解不可能な値を意味する**原子値（atomic value）** で

図 3.1: 非正規関係

あることを想定している．原子値の例としては，文字列，整数，実数，真偽値，日付などが挙げられ，どれもデータの基本単位と呼べるものである．逆に原子値で「ない」例としては，タプルや集合，関係などが挙げられる．ドメインとして原子値以外の値をとることを許す関係データを**非正規関係（unnormalized relation）**と呼ぶ．逆に，ドメインとして原子値しかとらない関係は**第 1 正規形（first normal form; 1NF）である**という．

例えば，表 3.1 の (a)(b) はともに商品の購買履歴を表しているが，表 (a) は属性「購入品目」に集合要素が入ってしまっている（例えば，{“はーいお茶”, “きのこの里”, “のど飴”}）．表 (a) の属性「購入品目」のドメインが原子値以外の値（つまり文字列の集合）を許してしまっていることから，表 (a) は非正規関係と見なせる．

一方，表 (b) は表 (a) の購買品目の値が原子値（文字列の集合ではなく文字列）になるように設計されているため，第 1 正規形と見なせる．表 (a) では 1 行で表現できていた 1 回分の購買データが，表 (b) では複数行を使って表現されている．表 (b) のほうがデータ構造が簡潔になるため，データが扱いやすい．

3.3　一貫性制約

一貫性制約（integrity constraint） とは，データベースが対象とする実世界を反映するように設定される，データが満たすべき規則である．関係データモデルにおいては，代表的な一貫性制約として以下がある：

- ドメイン制約
- キー制約
- 参照制約
- データ従属性

関係データモデルを扱うデータベースを設計する際には，上記一貫性制約を踏まえて**関係スキーマの定義や（複数の関係スキーマへの）分解** を行う．

3.3.1　ドメイン制約

ドメイン制約（domain constraint） とは，関係 $\boldsymbol{R}(A_1, ..., A_n)$ に含まれるタプルの各成分は，対応する属性のドメインの要素でなければならないという制約である．これは属性の定義ですでに触れたことである．ドメイン制約では，各属性のドメインのデータ型（例: 整数，実数，文字列，日付）に加えて，値の取り得る範囲を指定することもある．例えば，

先の例でとりあげた関係

$$\text{商品}\ (\text{商品}\ ID, \text{名称}, \text{単価}, \text{登録日}) \tag{3.16}$$

において，属性「単価」や「登録日」のドメインは

$$Dom(\text{単価}) = \{x \mid x \in \mathbb{N}\} \tag{3.17}$$

$$Dom(\text{登録日}) = \{x \mid x \in \text{年月日集合（ただし西暦表記）}\} \tag{3.18}$$

のように定義されていた．これらはドメイン制約である．これら制約にもとづき，

- 「単価」属性の値は**必ず**自然数
- 「登録日」属性の値は**必ず**西暦表記の年月日集合

でなければならない．よって，表 3.2 に記したの関係「商品 (*)」は関係「商品（商

表 3.2: ドメイン制約に違反する関係

商品 (*)

商品 ID	**名称**	**単価**	**登録日**
P1	はーいお茶	130	2020/07/15
P2	午前の紅茶	130	2020/09/25
P3	健康麦茶	150	令和 3 年 2 月 16 日
⋮	⋮	⋮	⋮
P1000	きのこの里	2 百	2023/01/08

品 ID, 名称, 単価, 登録日)」のインスタンスには**なれない**．なぜなら，

- 商品 ID が P1 の商品の登録日は年月はあるが日が抜けている
- 商品 ID が P2 の商品の登録日は和暦で表現されている
- 商品 ID が P1000 の商品の単価は漢数字（文字列）で表現されている

からである．

3.3.2 キー制約

まず，キー制約の前提となるキーの概念について説明する．関係 $\boldsymbol{R}$ における**超キー（super key）** とは，関係 $\boldsymbol{R}$ における属性の集合のうち，それらの属性値が決まればおのずと関係 $\boldsymbol{R}$ のタプルが唯一ひとつに決まる（タプルを一意に特定できる）ものを指す．

例えば，表 3.3 は関係

$$\text{連絡先 (学生 } ID\text{, 名前, 学年, 大学 } email\text{, 自宅住所)} \tag{3.19}$$

を表形式で記したもので，各行（タプル）は学生の連絡先情報を示している．大学において学生情報を管理する場合，学生 ID（の値）が決まれば唯一 1 つの学生連絡先情報（行; タプル）を照会できるよう，学生 ID は重複がないように設定される．それゆえ，この関係「連絡先」において属性「学生 ID」は超キーとなる．同様に，属性「大学 email」も超キーとなる．

なお，{ 学生 ID, 名前 }，{ 学生 ID, 名前, 学年 } のように学生 ID を含む属性

表 3.3: 関係「連絡先」

連絡先

学生 ID	名前	学年	大学 email	自宅住所
S1	川澄 桜	4	kawasumi@xxx.ac.jp	名古屋市 xxx
S2	山畑 滝子	3	taki@xxx.ac.jp	岡崎市 xxx
S3	田辺 通	3	t.tanabe@xxx.ac.jp	尾張旭市 xxx
S4	田辺 瑞穂	1	m.tanabe@xxx.ac.jp	尾張旭市 xxx
⋮	⋮	⋮	⋮	⋮
S1000	北 千種	2	kita@xxx.ac.jp	名古屋市 xxx

の集合も超キーになる．例えば，「学生 ID」が `S1`，「名前」が **川澄桜** である行は唯一 1 行に決まるので，属性集合 { 学生 ID, 名前 } は超キーになる．これは属性「学生 ID」が超キーであるから，当たり前である．

一方，属性「住所」は超キーにはなり得ない．学生 ID が `S3` と `S4` の学生の住所が同じ（つまり同じところに住んでいる）であるから，住所を 1 つ指定しても連絡先タプルが一意に特定できないからである．

超キーは複数ありえるが，上の例では属性「学生 ID」のみでタプルを特定できるように，属性集合 学生 ID, 名前 を使ってタプルを特定しようとするのは無駄であろう．そこで，超キーのうち極小（つまり最も小さい部分集合）のものを**候補キー（candiate key）** と定義する．候補キーは単純に**キー（key）** と呼ばれることもある．例えば，上の関係「連絡先」では，属性「学生 ID」や「大学 email」が候補キーとなりえる．

最後に**主キー（primary key）** の概念を導入する．候補キーのうち，値として**未定義や空の値（空値または NULL 値と呼ばれる）** を取る可能性がなく，かつデータベース管理上都合のよいキーの 1 つを主キーと決める．主キー以外の候補キーは**代替キー**と呼ばれる．例えば，先の関係「連絡先」の場合，候補キーは「学生 ID」と「大学 email」であったが，大学 email は「氏名 xxx.ac.jp」のようなものから「学籍番号 xxx.ac.jp」のようなものに変更される可能性があったとしても，学生 ID は一度決めたらほぼ変わらないと思われる．これらを鑑みると，関係「連絡先」においては属性「学生 ID」が主キーとして選ぶのがよい．

表 3.4: 関係「履修」

履修

科目 ID	学生 ID	成績
C1	S1	不可
C2	S1	良
C2	S2	優
C3	S3	可
⋮	⋮	⋮

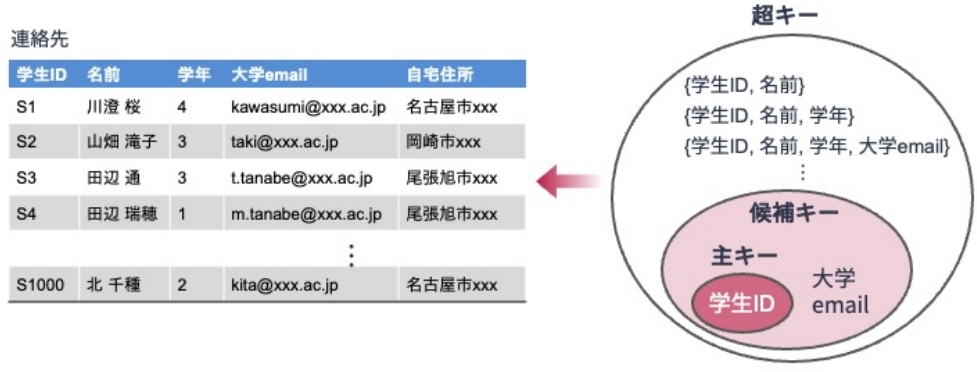

図 3.2: キーの概念の整理

なお，候補キーの定義から主キーは属性の集合をとることができる．このような主キーを**複合主キー（composite primary key）** と呼ぶ．

例えば，表 3.4 は，学生が履修した科目の成績をあらわす関係「履修」を表にしたものである．この表においては，科目 ID あるいは学生 ID だけでは成績が特定できないが，科目 ID と学生 ID の両方が決まればある学生のある科目の成績が特定される．それゆえ，関係「履修」においては { 科目 ID, 学生 ID } の属性ペアが主キーとなる．

関係表と主キー，候補キー，超キーの関係は，図 3.2 のようにまとめることができる．

ここまで，超キー，候補キー，主キーの説明を行ったが，最後に本節の主題であ

るキー制約について述べる．**キー制約（key constraint）** は，関係 $\boldsymbol{R}$ の関係スキーマに対して主キーが設定されたとき，$\boldsymbol{R}$ のインスタンスにおいては

- 主キーに設定された属性の値は重複があってはならない，かつ
- その要素（タプル）は主キーによって一意に特定されなければならない，かつ
- 主キーとなる属性の値は **NULL 値** であってはならない

という制約である．定義は小難しく見えるかもしれないが，キー制約は「ある属性が主キーと設定されたら，それがきちんと主キーの役割を果たすようデータが作られなければならない」ということを意味している．主キーを定義できれば，キー制約を定義できたようなものである．

さて，関係スキーマにおいてキー制約以外の一貫性制約には注目しない場合，関係 $\boldsymbol{R}$ の関係スキーマは以下のように主キーに下線を引いて表す．

$$\boldsymbol{R}(\underline{A_1, A_2}, ..., A_n) \tag{3.20}$$

例えば，図 3.2 の関係「連絡先」であれば，その関係スキーマは以下のように表す．

$$\text{連絡先}\ (\underline{\text{学生}\ ID}, \text{名前}, \text{学年}, \text{大学}\ email, \text{自宅住所}) \tag{3.21}$$

3.3.3　参照整合性制約

（関係データモデルにもとづく）関係データベースでは，対象となる事象のデータを正しく管理するために，**データを複数の関係（表）で管理する**ことがほとんどである．複合主キーの説明の例で使った関係「履修」には，その値だけ見ても具体的に何を意味しているのかが分からない属性「科目 ID」「学生 ID」がある．これらの意味を読み解くためには

- 「科目 ID」がどのような科目のことを指しているのか
- 「学生 ID」がどの学生のことを指しているのか

といった情報を管理する他の関係（表）が必要となる．以下の図は，学生の成績を管理するための関係データベースの例である．この中には関係「履修」も含まれている．

学生

学生ID	姓	名	所属
S1	川澄	桜	A学部
S2	山畑	滝子	B学部
S3	田辺	通	C学部
	⋮		

科目

科目ID	科目名	開講年度
C1	線形代数	2023
C2	線形代数	2024
C3	統計学入門	2024
	⋮	

履修

科目ID	学生ID	成績
C1	S1	不可
C2	S1	良
C2	S2	優
C3	S3	可
	⋮	

図 3.3: 成績管理データベース

上記 3 つの関係表があれば，

- 関係「学生」から「学生 ID」が `S1` のものを探し，
- 関係「科目」から「科目 ID」が `C1` のものを探す

ことで，関係「履修」において「学生 ID」が `S1` で「科目 ID」が `C1` の「成績」が **不可** だった件について，具体的にどんな学生が何の科目を落としてしまった（不可になってしまった）かを把握することができる．このとき，関係「履修」のタプルにある「科目 ID」の `C1` という値が，関係「科目」のインスタンスの属性「科目 ID」に含まれていなければならない（関係「科目」の科目 ID の列にその値が存在しなければならない），ということは自明であろう．関係「履修」の中にはあるが，関係「科目」には存在していなければ，データ管理として破綻していることになる．「学生 ID」の `S1` についても同様である．

上記の例では,

- 関係「履修」の属性「学生 ID」は関係「学生」の主キーである「学生 ID」と
- 関係「履修」の属性「科目 ID」は関係「科目」の主キーである「科目 ID」と

紐付いていることが分かる．言い方を変えると，関係「履修」は属性「学生 ID」および「科目 ID」の値を通して，関係「学生」および「科目」の情報を参照していることになる．

このように，関係スキーマ $\boldsymbol{R_1}(..., FK, ...)$ と $\boldsymbol{R_2}(\underline{PK}, ...)$ が与えられ，$\boldsymbol{R_1}$ におけるタプルの属性 FK の値は必ず $\boldsymbol{R_2}$ におけるいずれかのタプルの主キー PK の値と一致するように設計されているとき，$\boldsymbol{R_1}$ の属性 FK を $\boldsymbol{R_2}$ の PK に対する**外部キー（foreign key）** と呼ぶ．関係スキーマ上では，関係 $\boldsymbol{R_1}$ の属性 FK が関係 $\boldsymbol{R_2}$ の主キー PK に対する外部キーであることを以下のように記す．

$$\boldsymbol{R_1}.FK \subseteq \boldsymbol{R_1}.PK \tag{3.22}$$

先の例では，関係「履修」の属性「学生 ID」および「科目 ID」が外部キーとなるので，関係スキーマとして

$$\text{履修.学生 } ID \subseteq \text{学生.学生 } ID \tag{3.23}$$
$$\text{履修.科目 } ID \subseteq \text{科目.科目 } ID \tag{3.24}$$

と記す．

前置きが長くなったが，本節のテーマである**参照整合性制約（referential constraint）** とは「関係スキーマにおいて外部キーが設定されたとき，そのいかなるインスタンスも上記外部キーの条件を満たさなければいけない」という制約である．

3.3.4　データ従属性

一貫性制約として，ドメイン制約，キー制約，参照制約を挙げてきた．関係データモデルにはこれら以外にも，データ間に成立する制約を数学的に記述する手段がある．これを**データ従属性（data dependency）** と呼ぶ．

データ従属性としては, 関数従属性, 多値従属性, 結合従属性など, 様々なものが提案されている．**関数従属性（functional dependency）** は，データ従属性の中でも最も単純で, かつ実用的なものである．関数従属性は,「関係 $\boldsymbol{R}(..., X, ..., Y, ...)$

において，属性（あるいは属性集合）X の値が決まると属性 Y の値も一意に決まる」という性質である．このことを

$$X \to Y \tag{3.25}$$

と記す．

例えば，関係として

連絡先 (学生 ID, 名前, 学年, 大学 $email$, 郵便番号, 都道府県, 自宅住所) (3.26)

が与えられたとしよう．一般常識から，住所が決まれば住所がその住所が存在する都道府県や郵便番号もひとつに決まる．このことから，関係「連絡先」においては

$$\text{自宅住所} \to \text{郵便番号} \tag{3.27}$$
$$\text{自宅住所} \to \text{都道府県} \tag{3.28}$$

という関数従属性が定義できる．関数従属性はキー制約を一般化したものと捉えることができる．

データ従属性は，データの更新があったときにその影響が最小限になるような関係データベースを設計する上で，極めて重要である．それゆえ，特に関数従属性については別章で取り上げる．

参考文献

[1] 北川博之：データベースシステム, オーム社, 改訂 2 版 (2020).

[2] 吉川正俊：データベースの基礎, オーム社 (2019).

[3] Amazon Inc., : What is ecommerce? Definition, advantages and disadvantages of ecommerce websites and ecommerce business: `https://sell.amazon.co.jp/learn/what-is-ecommerce`.

[4] Amazon Inc., (2021), `https://press.aboutamazon.com/jp/news/`中小企業支援`/2021/10/amazon-`中小規模の販売事業者様の販売状況を発表.

[5] Bachman, C. W.: Data structure diagrams, *ACM SIGMIS Database: the DATABASE for Advances in Information Systems*, Vol. 1, No. 2, pp. 4–10 (1969).

[6] Tsichritzis, D. C. and Lochovsky, F. H.: Hierarchical Data-Base Management: A Survey, *ACM Computing Surveys*, Vol. 8, No. 1, pp. 105–123 (1976).

[7] Atkinson, M., Dewitt, D., Maier, D., Bancilhon, F., Dittrich, K. and Zdonik, S.: *The object-oriented database system manifesto*, pp. 946–954, Morgan Kaufmann Publishers Inc. (1994).

[8] DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P. and Vogels, W.: Dynamo: amazon's highly available key-value store, in *Proceedings of Twenty-First ACM SIGOPS Symposium on Operating Systems Principles (SOSP 2007)*, pp. 205–220, Association for Computing Machinery (2007).

[9] neo4j, : What is a graph database: `https://neo4j.com/docs/getting-started/graph-database/`.

[10] Codd, E. F.: A relational model of data for large shared data banks,

Communications of the ACM, Vol. 13, No. 6, pp. 377–387 (1970).