22. RDB に関する知識 I

1. 科目の概要

現在におけるデータベースの主流である関係データベース(RDB)に関して、その基本的な構成、機能と役割、特徴を解説する。また実際の開発や運用に必要な設計手法や正規化手法、分析手法について説明を加える。

2. 習得ポイント

本科目の学習により習得することが期待されるポイントは以下の通り。

習得ポイント	説　明	シラバスの対応コマ
I-22-1. データベースの意義、位置付けと利用方法	データベースを利用することのメリット、データベース管理システム(DBMS)の位置づけとOSSの実装、データベースの構造など、データベースを利用するための基本的な概念について説明する。	1,2
I-22-2. RDBMSの特徴、関係モデルとSQL	データ構造モデルとしての関係モデルを説明したうえで、関係モデルに基づく関係データベース管理システム(RDBMS)を説明し、データベース言語SQLの概要を紹介する。	1,2
I-22-3. 集合演算の概念	関係データベースの基礎となる関係モデルについて、その演算に関する概念を説明する。和、積、差、直積といった通常の集合演算と、射影、結合、選択、商といった関係代数独自の演算について解説を行う。	4
I-22-4. トランザクションの概念、排他制御とロック	データベースの特性とデータベース設計に不可欠なトランザクションの概念と特徴について解説し、確実なトランザクションを実現するための排他制御とロックの概念およびその実装方法を説明する。	3
I-22-5. 関係データベースの構成要素	関係データベースの構成要素である表、行、カラムなどの意味について説明する。またレコードを一意に特定するための主キー、複数属性からなる主キー、外部キーなど、リレーションスキーマに関するいくつかのトピックを紹介する。	4
I-22-6. 整合性を保つ方法と各種の演算	関係データベースを構成する各要素の間で整合性を保持する方法や、整合性が保たれるための条件を説明する。またリレーショナルデータモデルに対して行われる各種の演算を、演算結果の例を示しながら分かりやすく説明する。	4
I-22-7. DOA(データ中心分析)の意義、目的、内容	従来のプロセス中心設計に代わるデータベース設計手法として着目されているデータ中心分析DOA (Data Oriented Approach)の概要と、その意義、目的、具体的な手順について概説する。また関連する話題としてデータモデルおよび3層スキーマについても触れる。	5
I-22-8. データベースの設計手順	データベース設計方法と設計手順を解説する。業務分析、概念設計、論理設計、物理設計とデータベース設計の手順を進める中で、正規化が重要な役割を担うことを示す。また分析の手順としてトップダウンアプローチおよびボトムアップアプローチがあることを説明する。	6
I-22-9. ERモデルの具体的な考え方と作成手順、記述法	関係データベース設計の重要な表現法であるERモデル(Entity-Relational Model)について、基本的な考え方、データの表現方法や構成要素、具体的な作成手順を解説する。またERD (Entity-Relational Diagram)を用いたデータ設計手順について説明する。	7
I-22-10. データベース正規化の具体的な考え方と手順	関係データベース設計の必須技術である「正規化」について、具体的な考え方と正規形の種類、関数従属といった基本的な概念を説明する。また正規化の手順を示し、各手順におけるポイントや正規化を崩すケースなどについて述べる。	8

【学習ガイダンスの使い方】

1. 「習得ポイント」により、当該科目で習得することが期待される概念・知識の全体像を把握する。
2. 「シラバス」、「IT 知識体系との対応関係」、「OSS モデルカリキュラム固有知識」をもとに、必要に応じて、従来の IT 教育プログラム等との相違を把握した上で、具体的な講義計画を考案する。
3. 習得ポイント毎の「学習の要点」と「解説」を参考にして、講義で使用する教材等を準備する。

3. IT 知識体系との対応関係

「22. RDB に関する知識Ⅰ」と IT 知識体系との対応関係は以下の通り。

科目名	基本レベル（Ⅰ）								応用レベル（Ⅱ）						
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
22. RDBに関するスキル	<データベースの基礎理論>	<RDBMSの基本知識>	<トランザクションの基本概念>	<データベースの構成要素>	<DOAの内容概要>	<データベース設計の基本理論>	<ERモデル>	<正規化の手順と方法>	<データベースインデックス>	<データベースの物理構造>	<SQLによるデータベースアクセス>	<SQL実践演習ワークショップ>	<代表的なオープンソースRDBMS製品>	<データベース設計構築の実践>	<データベース構築>

[シラバス：http://www.ipa.go.jp/software/open/ossc/download/Model_Curriculum_05_22.pdf]

<IT 知識体系上の関連部分>

分野		科目名	1	2	3	4	5	6	7	8	9	10	11	12	13
組織関連事項と情報システム	1	IT-IAS 情報保証と情報セキュリティ	IT-IAS1. 基礎的な問題	IT-IAS2. 情報セキュリティの仕組み（対策）	IT-IAS3. 運用上の問題	IT-IAS4. ポリシー	IT-IAS5. 攻撃	IT-IAS6. 情報セキュリティ分野	IT-IAS7. フォレンジック（情報証拠）	IT-IAS8. 情報の状態	IT-IAS9. 情報セキュリティサービス	IT-IAS10. 脅威分析モデル	IT-IAS11. 脆弱性		
	2	IT-SP 社会的な観点とプロフェッショナルとしての課題	IT-SP1. プロフェッショナルとしてのコミュニケーション	IT-SP2. コンピュータの歴史	IT-SP3. コンピュータを取り巻く社会環境	IT-SP4. チームワーク	IT-SP5. 知的財産権	IT-SP6. コンピュータの法的問題	IT-SP7. 組織の中のIT	IT-SP8. プロフェッショナルとしての倫理的な問題と責任	IT-SP9. プライバシーと個人の自由				
応用技術	3	IT-IM 情報管理	IT-IM1. 情報管理の概念と基礎 [22-Ⅰ-1]	IT-IM2. データベース問合わせ言語 [22-Ⅰ-1]	IT-IM3. データアーキテクチャ [22-Ⅰ-2, 4, 8]	IT-IM4. データモデリングとデータベース設計 [22-Ⅰ-3, 4, 5]	IT-IM5. データと情報の管理	IT-IM6. データベースの応用分野							
	4	IT-WS Webシステムとその技術	IT-WS1. Web技術	IT-WS2. 情報アーキテクチャ	IT-WS3. デジタルメディア	IT-WS4. Web開発	IT-WS5. 脆弱性	IT-WS6. ソーシャルソフトウェア							
ソフトウェアの方法と技術	5	IT-PF プログラミング基礎	IT-PF1. 基本データ構造	IT-PF2. プログラミングの基本的構成要素	IT-PF3. オブジェクト指向プログラミング	IT-PF4. アルゴリズムと問題解決	IT-PF5. イベント駆動プログラミング	IT-PF6. 再帰							
	6	IT-IPT 技術を統合するためのプログラミング	IT-IPT1. システム間通信	IT-IPT2. データ割り当てと交換	IT-IPT3. 統合的コーディング	IT-IPT4. スクリプティング手法	IT-IPT5. ソフトウェアセキュリティの実現	IT-IPT6. 種々の問題	IT-IPT7. プログラミング言語の概要						
	7	CE-SWE ソフトウェア工学	CE-SWE0. 歴史と概要	CE-SWE1. ソフトウェアプロセス	CE-SWE2. ソフトウェアの要求と仕様	CE-SWE3. ソフトウェアの設計	CE-SWE4. ソフトウェアのテストと検証	CE-SWE5. ソフトウェアの保守	CE-SWE6. ソフトウェア開発・保守ツールと環境	CE-SWE7. ソフトウェアプロジェクト管理	CE-SWE8. 言語翻訳	CE-SWE9. ソフトウェアのフォールトトレランス	CE-SWE10. ソフトウェアの構成管理	CE-SWE11. ソフトェアの標準化	
	8	IT-SIA システムインテグレーションとアーキテクチャ	IT-SIA1. 要求仕様	IT-SIA2. 調達/手配	IT-SIA3. インテグレーション	IT-SIA4. プロジェクト管理	IT-SIA5. テストと品質保証	IT-SIA6. 組織の特性	IT-SIA7. アーキテクチャ						
システム基盤	9	IT-NET ネットワーク	IT-NET1. ネットワークの基礎	IT-NET2. ルーティングとスイッチング	IT-NET3. 物理層	IT-NET4. セキュリティ	IT-NET5. アプリケーション分野	IT-NET6. ネットワーク管理							
	10	CE-NWK テレコミュニケーション	CE-NWK0. 歴史と概要	CE-NWK1. 通信ネットワークのアーキテクチャ	CE-NWK2. 通信ネットワークのプロトコル	CE-NWK3. LANとWAN	CE-NWK4. クライアントサーバコンピューティング	CE-NWK5. データのセキュリティと整合性	CE-NWK6. ワイヤレスコンピューティングとモバイルコンピューティング	CE-NWK7. データ通信	CE-NWK8. 組込み機器向けネットワーク	CE-NWK9. 通信技術とネットワーク概要	CE-NWK10. 性能評価	CE-NWK11. ネットワーク管理	CE-NWK12. 圧縮と伸張
			CE-NWK13. クラスタシステム	CE-NWK14. インターネットアプリケーション	CE-NWK15. 次世代インターネット	CE-NWK16. 放送									
	11	IT-PT プラットフォーム技術	IT-PT1. オペレーティングシステム	IT-PT2. アーキテクチャと機構	IT-PT3. コンピュータインフラストラクチャ	IT-PT4. デプロイメントソフトウェア	IT-PT5. ファームウェア	IT-PT6. ハードウェア							
	12	CE-OPS オペレーティングシステム	CE-OPS0. 歴史と概要	CE-OPS1. 並行性	CE-OPS2. スケジューリングとディスパッチ	CE-OPS3. メモリ管理	CE-OPS4. セキュリティと保護	CE-OPS5. ファイル管理	CE-OPS6. リアルタイムOS	CE-OPS7. OSの概要	CE-OPS8. 設計の原則	CE-OPS9. デバイス管理	CE-OPS10. システム性能評価		
コンピュータアーキテクチャとハードウェア	13	CE-CAO コンピュータのアーキテクチャと構成	CE-CAO0. 歴史と概要	CE-CAO1. コンピュータアーキテクチャの基礎	CE-CAO2. メモリシステムの構成とアーキテクチャ	CE-CAO3. インタフェースと通信	CE-CAO4. デバイスサブシステム	CE-CAO5. CPUアーキテクチャ	CE-CAO6. 性能・コスト評価	CE-CAO7. 分散・並列処理	CE-CAO8. コンピュータによる計算	CE-CAO9. 性能向上			
複数領域にまたがるもの	14	IT-ITF IT基礎	IT-ITF1. ITの一般的なテーマ	IT-ITF2. 組織の問題	IT-ITF3. ITの歴史	IT-ITF4. IT分野（学科）とそれに関連のある分野（学科）	IT-ITF5. 応用領域	IT-ITF6. IT分野における数学と統計学の活用							
	15	CE-ESY 組込みシステム	CE-ESY0. 歴史と概要	CE-ESY1. 低電力コンピューティング	CE-ESY2. 高信頼性システムの設計	CE-ESY3. 組込み用アーキテクチャ	CE-ESY4. 開発環境	CE-ESY5. ライフサイクル	CE-ESY6. 要件分析	CE-ESY7. 仕様定義	CE-ESY8. 構造設計	CE-ESY9. テスト	CE-ESY10. プロジェクト管理	CE-ESY11. 並行設計（ハードウェア、ソフトウェア	CE-ESY12. 実装
			CE-ESY13. リアルタイムシステム設計	CE-ESY14. 組込みマイクロコントローラ	CE-ESY15. 組込みプログラム	CE-ESY16. 設計手法	CE-ESY17. ツールによるサポート	CE-ESY18. ネットワーク型組込みシステム	CE-ESY19. インタフェースシステムと混合信号システム	CE-ESY20. センサ技術	CE-ESY21. デバイスドライバ	CE-ESY22. メンテナンス	CE-ESY23. 専門システム	CE-ESY24. 信頼性とフォールトトレランス	

4. OSS モデルカリキュラム固有の知識

OSS モデルカリキュラム固有の知識は特になく、各回の内容は IT 知識体系と共通した RDB に関する内容を扱う。

科目名	第1回	第2回	第3回	第4回	第5回	第6回	第7回	第8回
22. RDB に関する知識Ⅰ	（1）データベースとは （2）情報システムにおける関係データベースの役割とメリット （3）データベースの仕組みと構造	（1）RDBMS とは （2）RDBMS の機能	（1）トランザクションとは （2）排他制御とロック	（1）データベースの構成要素 （2）参照整合性	（1）DOA（Data Oriented Approach） （2）データモデル	（1）データベースの設計手順 （2）分析の進め方	（1）ER モデル（Entity-Relationship Model） （2）ERD を用いたデータ設計手順 （3）ERD の効果	（1）正規化 （2）正規化の手順

（網掛け部分は IT 知識体系で学習できる知識を示し、それ以外は OSS モデルカリキュラム固有の知識を示している）

スキル区分	OSS モデルカリキュラムの科目	レベル
RDB 分野	22　RDB に関する知識 I	基本
習得ポイント	I-22-1. データベースの意義、位置付けと利用方法	
対応する コースウェア	第 1 回（データベースの基礎理論） 第 2 回（RDBMS の基本知識）	

I-22-1. データベースの意義、位置付けと利用方法

データベースを利用することのメリット、データベース管理システム(DBMS)の位置づけと OSS の実装、データベースの構造など、データベースを利用するための基本的な概念について説明する。

【学習の要点】

* データを格納する際、さまざまなメリットから、データベースとして格納するのが一般的である。データベースを利用すると、一貫性を保ちながらデータを一元管理することができる。
* データベース管理システム(DBMS)とは、データベースを運用、管理するソフトウェアである。
* アプリケーションから DBMS に指示を与えることで、データベースに格納されたデータをアプリケーションで利用することができる。
* 近年は、OSS の DBMS も広く利用されるようになっている。

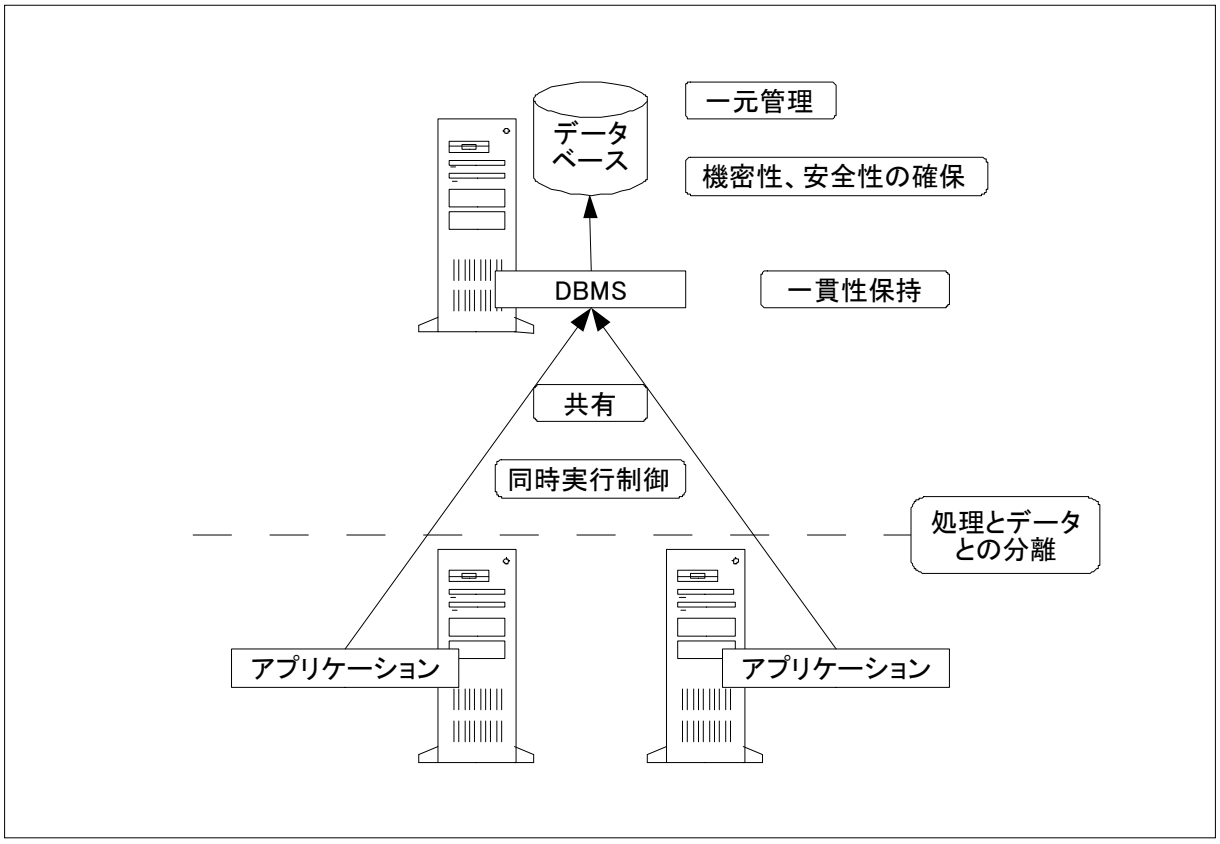

図 I-22-1.　データベースのメリット

【解説】

1) データベース

データベースとは、構造化された(相互に関連するデータを整理・統合し、検索性を考慮された)電子データの集合である。データベースを利用するメリットには以下のようなものがある。

* 処理とデータとを分離独立させて扱える(保守が容易)。
* データを一元管理できる。
* データを共有できる。
* さまざまなデータを関連付けて格納できる。
* 登録するデータに制約をつけられる。
* データの一貫性(整合性)を確保できる。
* 複数のデータアクセスを同時に処理できる。
* データの機密性を確保しやすい。
* データを安全に格納する。

2) データベース管理システム

データベース管理システム(DBMS)とは、データベースを運用、管理し、データへのアクセス要求に応答するシステムおよびソフトウェアであり、上記メリットを実現するため、以下のような機能を備える。

* データ一貫性保持(トランザクション管理)
* データ制約
* 同時実行制御
* 障害回復

3) 関係データベース管理システムと OSS の実装

DBMS の種類として、ネットワーク型 DBMS、階層型 DBMS、オブジェクト指向 DBMS、XML DBMS などがあるが、現在の主流は関係データベース管理システム(RDDBMS)といわれるものである。
代表的な OSS の RDBMS としては、MySQL、PostgreSQL、Firebird、SQLite が挙げられる。

4) データベースのディスク上の配置

通常、DBMS はデータベースを OS 上の独自形式ファイルとして作成、保持する。MySQL では通常、データベース単位で専用のディレクトリが作成され、データベース中の要素はそのディレクトリ配下のファイルとして作成される。

5) アプリケーションからのデータベース利用方法

アプリケーションから DBMS に指示を与え、DBMS ではそれを解釈して OS 上のファイルにアクセスし、結果をアプリケーションに返す。MySQL、PostgreSQL、Firebird はクライアント/サーバ型の構成をとるが、SQLite はデータベースにアクセスするためのライブラリとしての形態をとる。

スキル区分	OSS モデルカリキュラムの科目	レベル
RDB 分野	22　RDB に関する知識 I	基本
習得ポイント	I-22-2. RDBMS の特徴、関係モデルと SQL	
対応する コースウェア	第 1 回（データベースの基礎理論） 第 2 回（RDBMS の基本知識）	

I-22-2. RDBMS の特徴、関係モデルと SQL

データ構造モデルとしての関係モデルを説明したうえで、関係モデルに基づく関係データベース管理システム(RDBMS)を説明し、データベース言語 SQL の概要を紹介する。

【学習の要点】

＊ DBMS の種類にはいろいろあるが、RDBMS はその中で主流をなす。

＊ RDBMS は、データ構造を表の形式で視覚化できる「関係モデル」というデータベースモデルに基づいている。

＊ RDBMS においては、データへアクセスするための命令として、SQL という言語が事実上の標準となっており、SQL は、DDL、DML、DCL の 3 つに大別される。

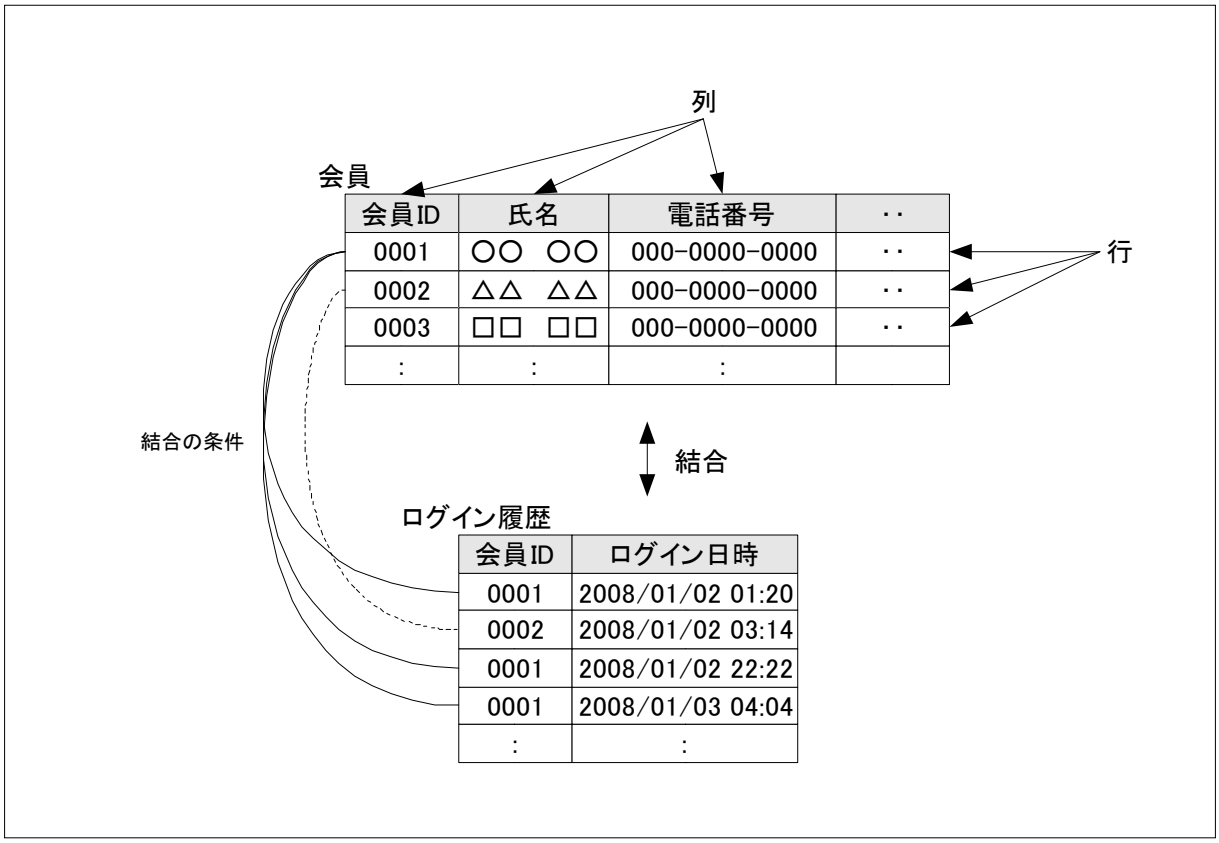

図 I-22-2.　関係モデル

【解説】

1) 関係データベース管理システム

関係モデルに基づく DBMS を、関係データベース管理システム(RDBMS)という。RDBMS においては、SQL というデータベース言語が事実上の標準となっている。

2) 関係モデル(リレーショナルモデル)

データベースにおけるデータの構造をモデル化したものをデータベースモデルという。関係モデルはデータベースモデルの一種である。関係モデルにおいては、データ構造は表の形式で視覚化することが可能である(図参照)。

* 各属性(例えば会員情報の ID、氏名、電話番号等)をデータ型(数値型、文字列型等)の定義された列として表現する。
* 各属性の値を持った 1 件のデータの組(例えば 1 人分の会員情報)を行として表現する。
* 複数件のデータ(例えば複数人の会員情報)を行の集合で表現する。

関係モデルにおいては、指定条件を満たす行の参照/変更/削除、といった形でデータベースにアクセスする。また、表の演算により、複数の異なる表のデータを関連づけることができる。(例えば、会員マスタ表とログイン履歴表とを、会員 ID が等しいという条件で結合することで、氏名とログイン日時とを関連づけることができる。)

3) SQL(Structured Query Language)

データベースへのアクセス方法としては、DBMS に対しデータベース言語(データアクセス等のための一連のコンピュータ言語)を与える方法が一般的である。SQL は関係モデルに基づくデータベース言語の一種で、ANSI や ISO で規格化されている。ほとんどの RDBMS 製品では SQL をサポートしているが、製品により SQL 規格への対応に違いがあり注意を要する。SQL の個々の命令は、機能により以下の 3 つに大別できる。

* データ定義言語 (Data Definition Language:DDL)
  データの構造を定義する命令の集合。
  - データベースや表を作成する CREATE 文
  - データベースや表の定義を変更する ALTER 文
  - データベースや表を削除する DROP 文
* データ操作言語 (Data Manipulation Language:DML)
  データへのアクセス(参照や変更など)を行う命令の集合。
  - 表データを抽出する SELECT 文
  - 表に行を挿入する INSERT 文
  - 表データを更新する UPDATE 文
  - 表から行を削除する DELETE 文
* データ制御言語 (Data Control Language: DCL)
  データに対するアクセス制御を行う命令の集合。
  - データベース利用者へ権限を付与する GRANT 文

  などがある。

スキル区分	OSS モデルカリキュラムの科目	レベル
RDB 分野	22　RDB に関する知識 I	基本
習得ポイント	I-22-3. 集合演算の概念	
対応する コースウェア	第 4 回（データベースの構成要素）	

I-22-3. 集合演算の概念

関係データベースの基礎となる関係モデルについて、その演算に関する概念を説明する。和、積、差、直積といった通常の集合演算と、射影、結合、選択、商といった関係代数独自の演算について解説を行う。

【学習の要点】

＊ 関係モデルにおけるデータの演算は、関係代数で定義されている。

＊ SQL によるデータ加工の命令は、関係代数の演算に基づいており、SQL の命令の理解には、関係代数の演算の理解が重要となる。

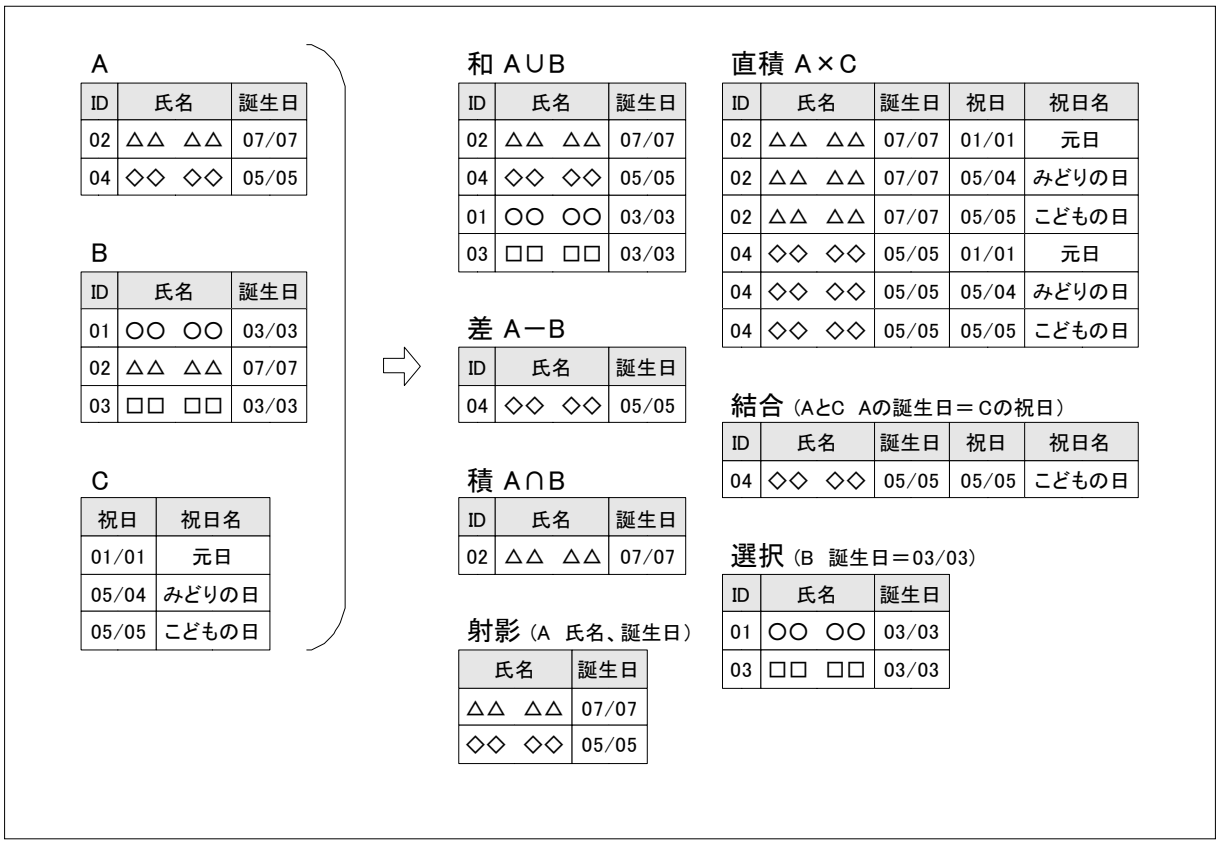

図 I-22-3.　表の演算の例

【解説】

1) SQL と関係代数

関係モデルでは、行を(列名、データ型、列の値)の組み合わせを要素とする集合としてとらえ、表を行という要素の集合としてとらえる。これらのデータの演算は、関係代数という演算体系において定義されており、SQL におけるデータ加工の命令も、関係代数に基づいている。

2) 関係代数の基本的な演算

関係代数で定義されている基本的な演算を、RDBMS の用語に置き換えて表現すると、以下のようになる。(図参照)

＊ 通常の集合演算

以下の演算は、表を行という要素の集合としてとらえた場合の、通常の集合演算となる。

- 和 A∪B

  表 A に属するまたは表 B に属する行で構成される表。
  型適合(A と B とで列数、列名、各列のデータ型が一致)の場合に可能な演算。
  両方に属する行を含む。同じ行の重複は取り除く。

- 差 A－B

  表 A に属するかつ表 B に属さない行で構成される表。
  型適合の場合に可能な演算。

- 積 A∩B

  表 A に属するかつ表 B に属する行で構成される表。
  型適合の場合に可能な演算。「共通」「交わり」とも呼ばれる。
  積は、差を用いて、A∩B＝A－(A－B) で表現される。

- 直積 A×B

  表 A に属する行と表 B に属する行との全ての組み合わせで構成される表。
  A×B の要素となる各行は、A に属する行と B に属する行との組み合わせとなる。
  「デカルト積」とも呼ばれる。

- 商 A÷B

  C×B＝A となるような表 C。
  商は関係代数では基本的な演算だが、SQL では対応した命令が存在しない。

＊ 関係代数独自の演算

- 選択

  表の行のうち、指定した条件に一致する行で構成される表。
  「制限」とも呼ばれる。

- 結合

  表の直積演算の結果となる表に、選択演算を適用した結果の表。
  表 A と表 B との直積となる表に、選択演算を適用した結果の表を、A と B との結合という。

- 射影

  表の各行から指定した列を取り出した行で構成される表。
  列は複数指定も可能。

スキル区分	OSS モデルカリキュラムの科目	レベル
RDB 分野	22　RDB に関する知識 I	基本
習得ポイント	I-22-4. トランザクションの概念、排他制御とロック	
対応する コースウェア	第 3 回（トランザクションの基本概念）	

I-22-4. トランザクションの概念、排他制御とロック

データベースの特性とデータベース設計に不可欠なトランザクションの概念と特徴について解説し、確実なトランザクションを実現するための排他制御とロックの概念およびその実装方法を説明する。

【学習の要点】

＊ データベースの特性であるデータの整合性を保持するために、トランザクションの適切な処理が不可欠である。

＊ 排他制御とロックの実装により、データベースの整合性を保持しながら、同時に複数のトランザクションを処理することが出来る。

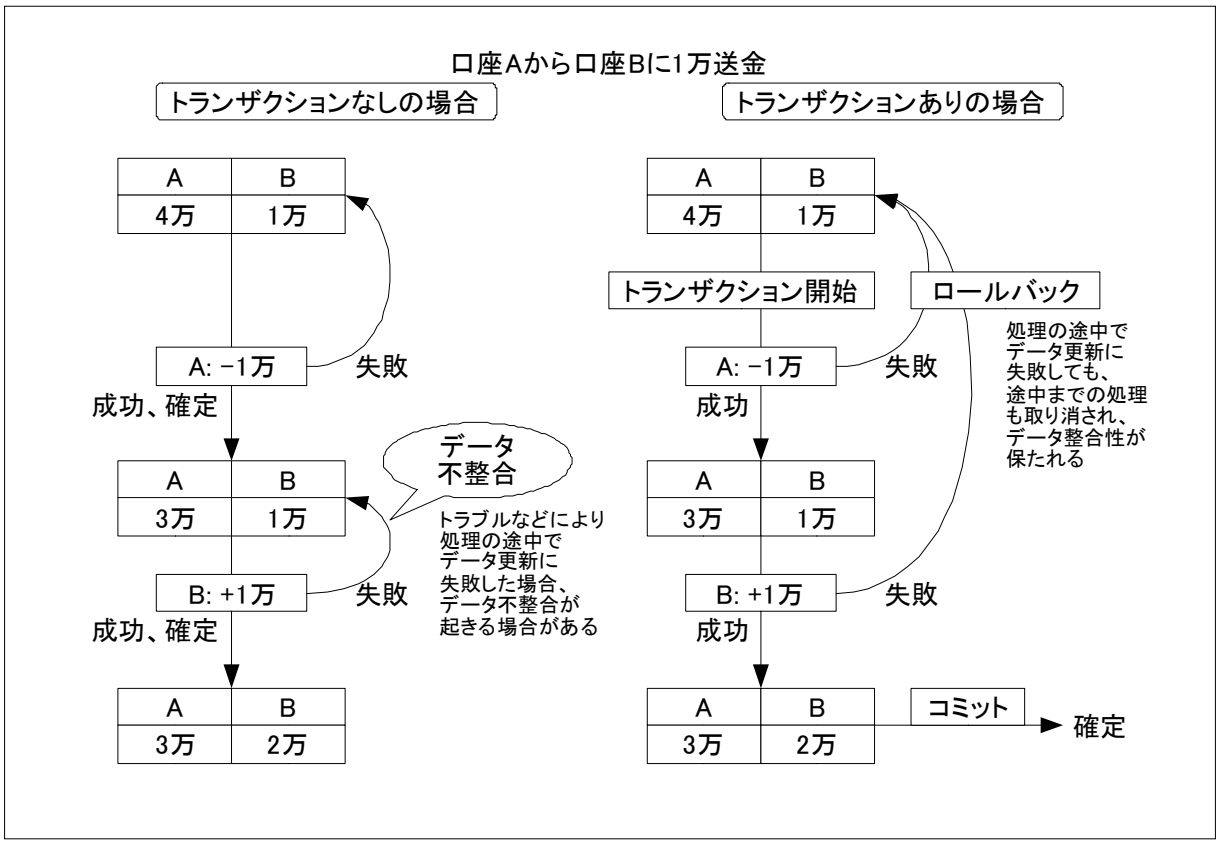

図 I-22-4.　トランザクションの概念

【解説】

1) トランザクションとは

データベースを操作する上での不可分な処理のまとまりをトランザクションという。例えば口座 A から口座Bに送金する場合、Aからの出金という処理とBへの入金という処理とは不可分であり、片方のみ処理された状態ではデータに不整合が起こる(図参照)。このようなデータ不整合を防ぐため、DBMS ではトランザクションの適切な処理が必要になる。

2) ACID 特性

DBMS がトランザクションを処理するための必要不可欠な要素として、以下の 4 つの性質が挙げられ、頭文字をとって ACID 特性と呼ばれる。

* 原子性(Atomicity)
  トランザクション内の全処理を確定する(コミット)か、全処理を取り消す(ロールバック)かのどちらかであること。
* 一貫性(Consistency)
  トランザクションの前後でデータベースの整合性が保たれていること。
* 独立性(Isolation)
  トランザクション内の一部の処理が行われた中間的な状態が、外部からは見えないこと。
* 耐久性(Durability)
  一旦確定(コミット)されたトランザクションは今後取り消し(ロールバック)されることが無いこと。

3) ロックと排他制御

データベースの特定のデータへのアクセスを一時的に制限することをロックと呼ぶ。あるデータへのアクセス要求があった場合、そのデータをロックし、他からの書き換えを禁止するなどデータアクセスを制限することを、排他制御と呼ぶ。各トランザクションに対して排他制御を適用することで、トランザクション内でのデータ一貫性を実現することができる。

4) ロックの種類と特徴、実装

ロックはその対象範囲やアクセス制限方法により、いくつかの種類がある。MySQL の場合、次のようなロックが実装されている(作成する表の種類により、使用できるロックは限られる)。

* テーブルレベルロック
  データを変更する処理において、対象データを含む表すべてをロックする。
* 行レベルロック
  データを変更する処理において、対象データを含む行をロックする。

テーブルレベルロックと比べ、行レベルロックは、多数のスレッドで異なる行をアクセスする際にロックコンフリクトが少ない、ロールバックの変更が少ない、メモリ消費が大きい、表の大部分で使用されると処理速度が遅い、などの違いがある。

スキル区分	OSS モデルカリキュラムの科目	レベル
RDB 分野	22　RDB に関する知識 I	基本
習得ポイント	I-22-5. 関係データベースの構成要素	
対応する コースウェア	第 4 回（データベースの構成要素）	

I-22-5. 関係データベースの構成要素

関係データベースの構成要素である表、行、カラムなどの意味について説明する。またレコードを一意に特定するための主キー、複数属性からなる主キー、外部キーなど、リレーションスキーマに関するいくつかのトピックを紹介する。

【学習の要点】

* 関係データベースでは、データを表の形式で保持する。
* 関係データベースでは、目的に応じて複数の表に対して演算を行うことで、求めたい表を得ることができる。これが関係データベースの最大の特徴である。
* 表の作成の際は、関係スキーマとして、表の各列の名前やデータ型以外にも様々な条件を設定できる。

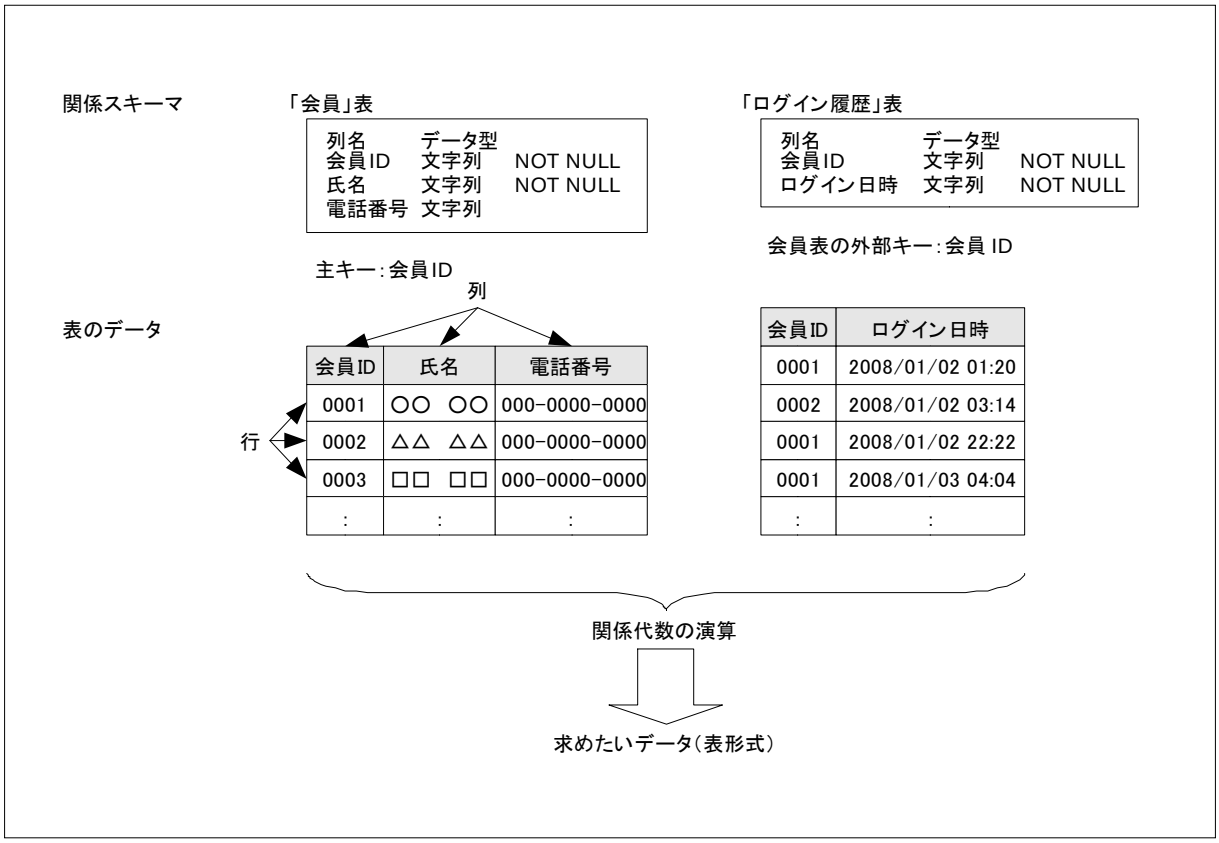

図 I-22-5.　関係データベースの特徴

【解説】

1) 関係データベースの構成要素

関係データベースでは、データを表形式の構造で格納し、行や列は次のような意味を持つ。

＊ 列（カラム、フィールド）
データベースに保存するデータの最小単位。名前とデータ型(数値型、文字列型等)を持つ。

＊ 行（レコード）
表の各列の値の 1 組の集合。

例として、複数の口座名義と残高のデータを管理することを考える。口座名義のデータ型を文字列型、残高のデータ型を数値型とし、(口座名義、残高)の組を属性として持つ 1 件分のデータとすると、各行が 1 件分のデータ、各列が各属性(口座名義または残高)となる表として表現できる。

2) 関係スキーマ（リレーションスキーマ）

関係データベースを設定するための枠組みを関係スキーマという。関係スキーマには、表を構成する各列の列名やデータ型の定義のほか、以下のようなものを設定できる。

＊ デフォルト値
列の既定値。表の行を挿入する場合、列の値が指定されていなければ、デフォルト値が挿入される。

＊ NOT NULL
NOT NULL を設定した列で、NULL(列の値の設定されていない状態)を許可しない。表の行を更新/挿入する場合、NOT NULL を設定した列では、NULL をとることは出来ない。

＊ 一意キー（UNIQUE キー）
一意キーを設定した列の値(または列の値の組み合わせ)が、表内で一意である(同じ値を持つ行が 2 つ以上存在しない)。表の行を更新/挿入する場合、一意キーを設定した列(または列の組み合わせ)では、既存の値と同じ値をとることは出来ない。(NULL は許可される。)

＊ 主キー（プライマリキー）
表内の行を一意に識別するための列(または列の組み合わせ)。表ごとに 1 つまで指定できる。表の行を更新/挿入する場合、主キーとして設定として指定された列(または列の組み合わせ)では、既存の値と同じ値をとることはできず、また、NULL も許可されない。

＊ 外部キー
外部キーを設定した列の値(または列の値の組み合わせ)が、参照される他の表の列(または列の組み合わせ)に存在する。参照される表の列(または列の組み合わせ)を親キーという。親キーは参照される表の主キーまたは一意キーでなければならない。表の行を更新/挿入する場合、外部キーとして指定された列(または列の組み合わせ)では、参照される表の親キーに存在しない値をとることは出来ない。(NULL は許可される。)

＊ 索引（インデックス）
表の検索を高速化するため、表とは別に作成されるデータやファイルの仕組み。

スキル区分	OSS モデルカリキュラムの科目	レベル
RDB 分野	22　RDB に関する知識 I	基本
習得ポイント	I-22-6. 整合性を保つ方法と各種の演算	
対応する コースウェア	第 4 回（データベースの構成要素）	

I-22-6. 整合性を保つ方法と各種の演算

関係データベースを構成する各要素の間で整合性を保持する方法や、整合性が保たれるための条件を説明する。またリレーショナルデータモデルに対して行われる各種の演算を、演算結果の例を示しながら分かりやすく説明する。

【学習の要点】

＊ RDBMS には、行レベル、表レベル、データベースレベルで、データの整合性を保持する仕組みが備わっている。

＊ 関係スキーマで制約を設定することにより、整合性の保持機能を実装できる。

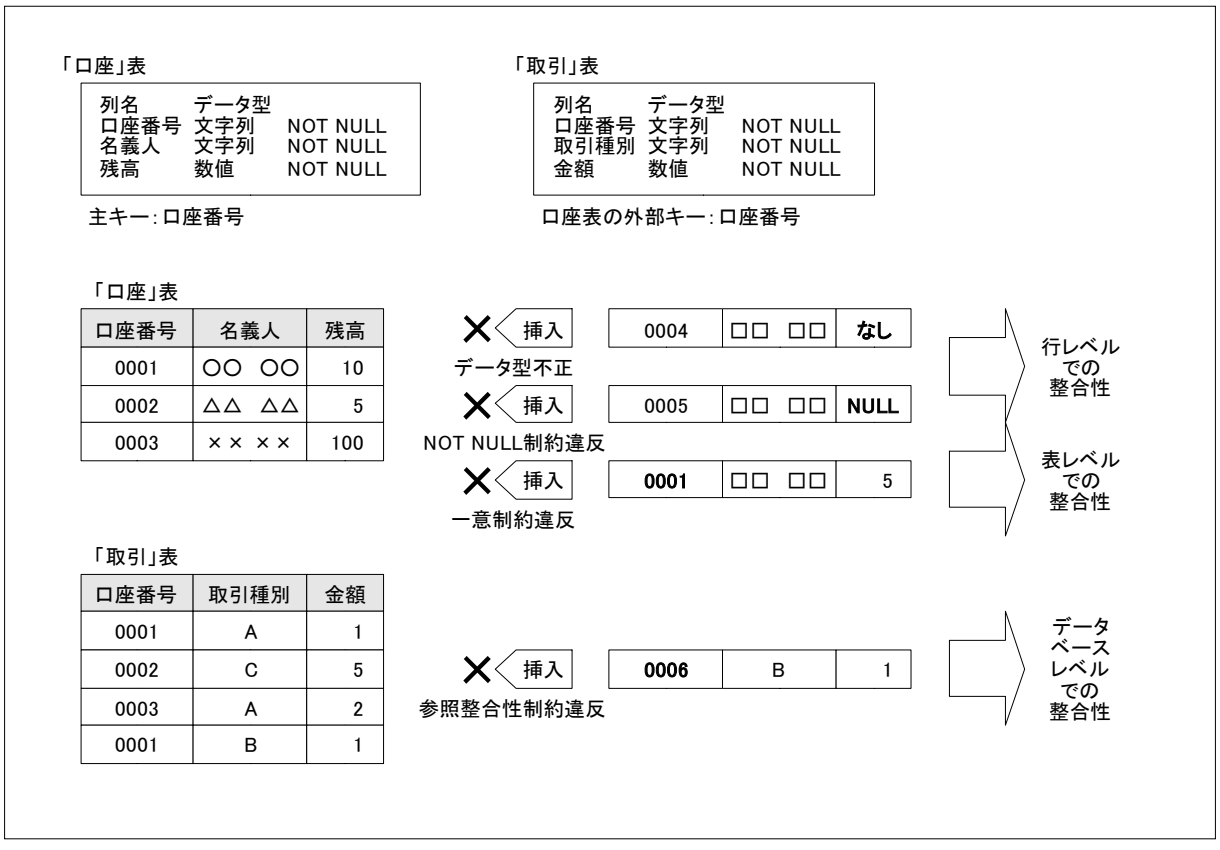

図 I-22-6.　RDBMS でのデータ整合性保持

【解説】

1) 関係データベースでの整合性の保持

RDBMS では、関係スキーマとして制約やキーを設定することで、行レベル、表レベル、データベースレベルでデータの整合性を保持できる仕組みを備えている。

* 行レベルでの整合性 ～ データ型による制約、NOT NULL 制約
* 表レベルでの整合性 ～ 一意キー設定による一意制約
* データベースレベルでの整合性 ～ 外部キー設定による参照整合性制約

2) 各種の制約とデータ操作

* データ型による制約
  行の各列の値が、スキーマによって定義されたデータ型以外の値をとることを防ぐ。
  例えば、[口座]表で[残高]列のデータ型を数値型に設定した場合、
  - [口座]表の行の更新または挿入の際、[残高]列の値に'なし'等の文字列を持つことはできない。
* NOT NULL 制約
  行の NOT NULL を設定した列が NULL をとることを防ぐ。
  例えば、[口座]表で[残高]列を NOT NULL に設定した場合、
  - [口座]表の行の更新または挿入の際、[残高]列に NULL を持つことはできない。
* 一意制約
  一意キーを設定した列の値(または列の値の組み合わせ)が、表内で重複することを防ぐ。これにより、一意キーを設定した列の値(または列の値の組み合わせ)が、表内で一意であることを保持する。
  例えば[口座]表で[名義人]列を一意キーに設定した場合、
  - [口座]表の行の更新または挿入の際、[名義人]列において、他の行の[名義人]列と同じ値を持つことはできない。
* 参照整合性制約
  外部キーを設定した列の(または列の組み合わせ)が、参照される他の表の列(または列の組み合わせ)に無い値をとることを防ぐ。これにより、外部キーを設定した列の値(または列の値の組み合わせ)が、親キーを設定した列から参照可能である状態を保持する。
  例えば[取引]表の[口座番号]列を外部キーに設定し、その親キーを[口座]表の[口座番号]列に設定した場合、
  - [取引]表の行の更新または挿入の際、[口座番号]列において、[口座]表の[口座番号]列にない値を持つことはできない。
  - [口座]表のある行において、[口座番号]列の値が[取引]表の[口座番号]列に存在する場合、その行の削除はできない。(あるいは、その行の削除の際、[取引]表においても、当該[口座番号]列を持つ行が削除される。)
  - [口座]表のある行において、[口座番号]列の値が[取引]表の[口座番号]列に存在する場合、その行の[口座番号]列の値の更新はできない。(あるいは、その行の更新の際、[取引]表の当該[口座番号]列の値も同じ値に更新される。)

スキル区分	OSS モデルカリキュラムの科目	レベル
RDB 分野	22　RDB に関する知識 I	基本
習得ポイント	I-22-7. DOA(データ中心分析)の意義、目的、内容	
対応する コースウェア	第 5 回 (DOA の内容概要)	

I-22-7. DOA(データ中心分析)の意義、目的、内容

従来のプロセス中心設計に代わるデータベース設計手法として着目されているデータ中心分析 DOA (Data Oriented Approach)の概要と、その意義、目的、具体的な手順について概説する。また関連する話題としてデータモデリングおよび 3 層スキーマについても触れる。

【学習の要点】

* データベースだけでなく、システムの全体の品質を高めるために、DOA という考え方が重要となる。
* データモデリングを抽象度に応じて 3 種類に分けることで、設計段階に応じたモデリングが可能となる。
* スキーマを 3 層に分けることで、概念スキーマを変更されにくい安定したものとすることができる。

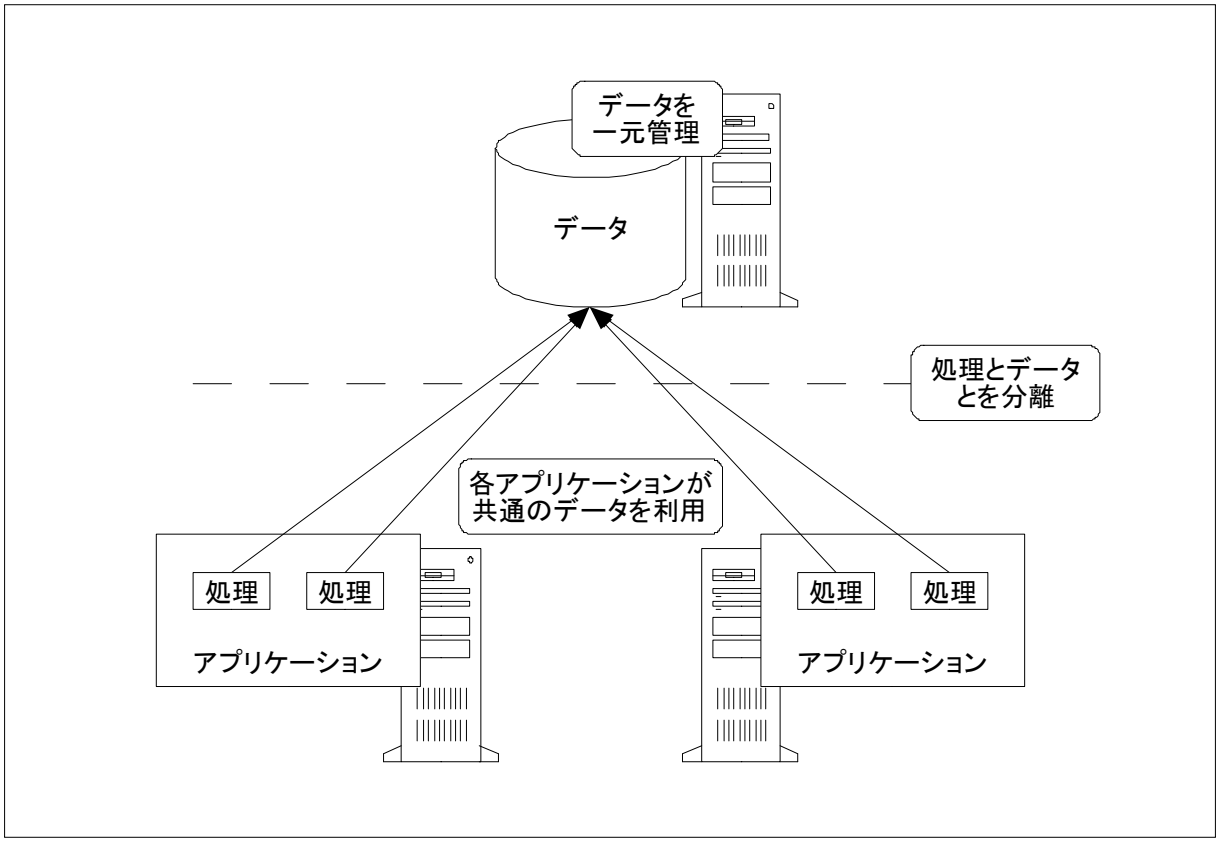

図 I-22-7.　DOA の概念

【解説】

1) DOA(Data Oriented Approach、データ中心アプローチ)とは

DOA とは、データの構造や流れに優先的に着目する分析、設計手法である。DOA が登場する前の従来の手法はPOA(Process Oriented Approach)と呼ばれ、業務処理に優先的に着目し、その後付随データに着目するという設計手法だったが、以下のような問題があった。

* 業務処理に応じた別個のシステムになり、システム間のやりとりが複雑になる傾向がある。
* システムに変更を加えていくとデータに不整合が起こりやすくなる。
* データの共有、一元管理が困難になる。

一般に、システムの業務処理は変更されやすいが、データやデータ構造は変更されにくいという傾向があるため、DOA は POA に比べ、システムの変更に強いと考えられる。

2) データモデリングと ERD (Entity-Relationship Diagram、ER 図)

特定の業務処理に応じたデータベースの論理的な構造を示したモデルを作成することをデータモデリングといい、関係データベースにおいては、ERD という表記手法が広く用いられている。データモデリングは、一般的に、抽象度の高いものから、「概念データモデル」、「論理データモデル」、「物理データモデル」の 3 種類に分類される。

* 概念データモデル
  システムの要件に基づいた大まかなモデル。
* 論理データモデル
  概念データモデルに、入出力データを踏まえてシステムで必要とする属性(列)をすべて付与したモデル。
* 物理データモデル
  実装を意識したモデル。関係データベースの表と 1 対 1 に対応し，データ型や索引までを定義した、スキーマの生成が可能なレベルのもの。

3) 3 層スキーマ

データベースにどのような情報をどのような構造で格納、利用するかという枠組みをスキーマといい、ANSI/SPARC では「3 層スキーマ構造」が提案されている。

* 概念スキーマ
  対象世界を論理的に表現したもの。関係データベースでは表の定義に相当する。
* 外部スキーマ(副スキーマ)
  DBMS の外部からデータにアクセスする立場から見た際のデータ構造を表現したもの。関係データベースでは、ビュー(表を演算した結果となる仮想的な表を定義したもの)の定義に相当する。
* 内部スキーマ(記憶スキーマ)
  データの物理的な格納方法を表現したもの。関係データベースでは、索引やデータファイルの定義に相当する。

スキル区分	OSS モデルカリキュラムの科目	レベル
RDB 分野	22　RDB に関する知識 I	基本
習得ポイント	I-22-8. データベースの設計手順	
対応する コースウェア	第 6 回（データベース設計の基本理論）	

I-22-8. データベースの設計手順

データベース設計方法と設計手順を解説する。業務分析、概念設計、論理設計、物理設計とデータベース設計の手順を進める中で、正規化が重要な役割を担うことを示す。また分析の手順としてトップダウンアプローチおよびボトムアップアプローチがあることを説明する。

【学習の要点】

* データベース設計は、正しい手順を踏むことで、品質を高め、仕様変更による手戻りも抑えることができる。
* 論理設計のアプローチには、トップダウンアプローチとボトムアップアプローチという考え方があるが、一長一短であり、うまく組み合わせて使うことが重要である。

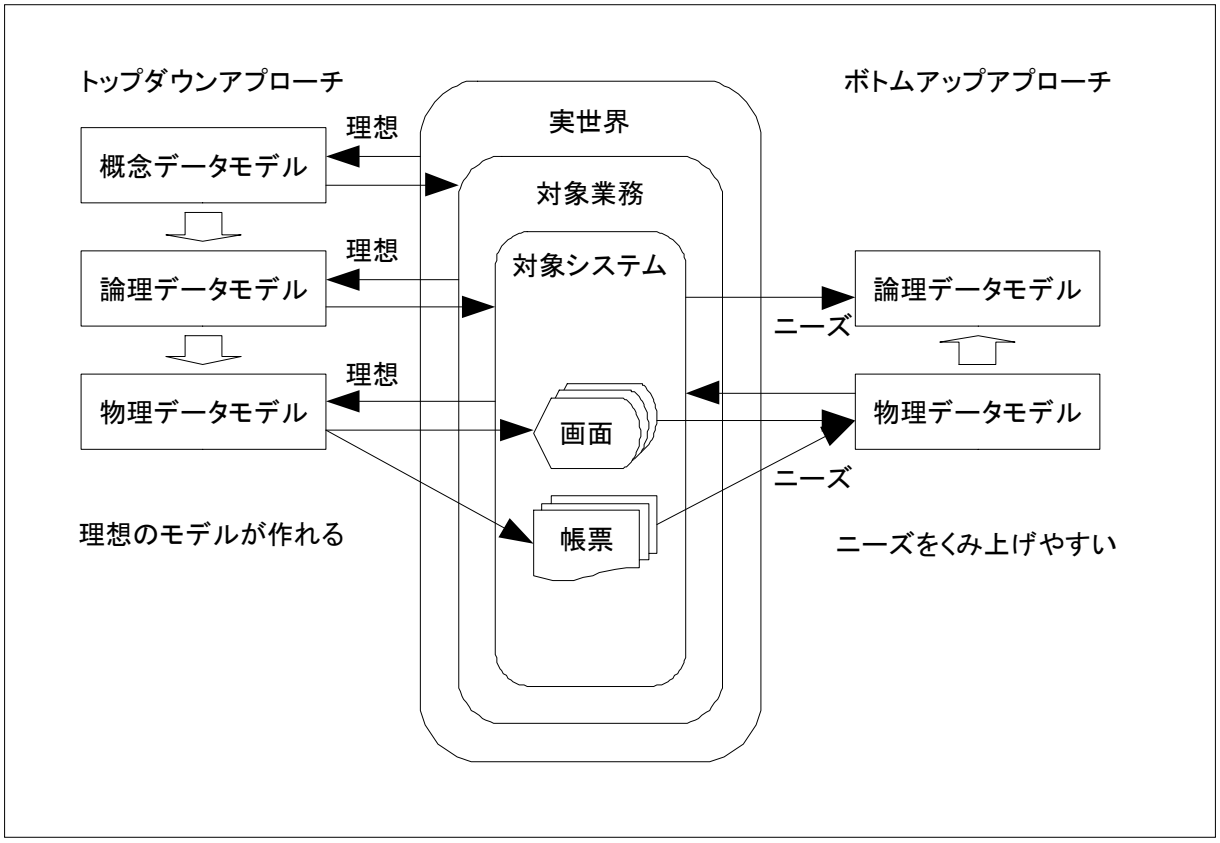

図 I-22-8.　トップダウンアプローチとボトムアップアプローチ

【解説】

1) 関係データベースの設計手法と設計手順

データベースの設計手順として、業務分析→概念設計→論理設計→物理設計という手順を考えることができる。

＊ 業務分析
対象とする業務実態を客観的に把握する。業務の内容(業務・業務機能)、処理方式、業務間を流れる情報量、業務量などを分析結果としてまとめる。

＊ 概念設計
業務分析結果をもとに、業務領域全体を表現する概念データモデルを ERD で作成する。

＊ 論理設計
業務をシステム化する領域としない領域とに分け、システム化する領域を表現する論理データモデルを ERD で作成する。

＊ 物理設計
論理データモデルを基に、利用する RDBMS に合わせた物理データモデルを作成する。保守性や性能等を考慮のうえ、SQL を発行可能なレベルまで詳細に記述する。

2) データベース設計における正規化

論理設計の段階で、正規化(I-22-10 に記載)という作業を行う。正規化は、データの冗長性を減らし、データベースの品質を高める上で重要な作業となる。また、非正規化(同じく I-22-10 に記載)は物理設計の段階で、利用する RDBMS に合わせて行う。

3) 論理設計のアプローチ

データベースの論理設計におけるアプローチには「トップダウン型」と「ボトムアップ型」という考え方がある。両アプローチを適切に併用することが望ましい。

＊ トップダウン型 DB 設計
概念データモデルをもとに具体化を進める形式。概念的なモデルに基づき、将来的な拡張性を含めた理想的な設計を行なえる可能性があるが、逆にユーザニーズを十分反映できない場合もある。

＊ ボトムアップ型 DB 設計
ユーザが利用している画面や帳票を抽象化してモデル化を行なう方法。直近のユーザニーズを的確に反映することができるが、将来的な拡張性、柔軟性にかけたモデルになる可能性がある。

スキル区分	OSS モデルカリキュラムの科目	レベル
RDB 分野	22　RDB に関する知識 I	基本
習得ポイント	I-22-9. ER モデルの具体的な考え方と作成手順、記述法	
対応する コースウェア	第 7 回 (ER モデル)	

I-22-9. ER モデルの具体的な考え方と作成手順、記述法

関係データベース設計の重要な表現法である ER モデル(Entity-Relational Model)について、基本的な考え方、データの表現方法や構成要素、具体的な作成手順を解説する。また ERD (Entity-Relational Diagram)を用いたデータ設計手順について説明する。

【学習の要点】

* ER モデルは、エンティティとリレーションシップに注目したデータモデリングの手法であり、ER モデルを表現した図を ERD という。
* ERD は、関係データベースの設計図として広く利用されている。

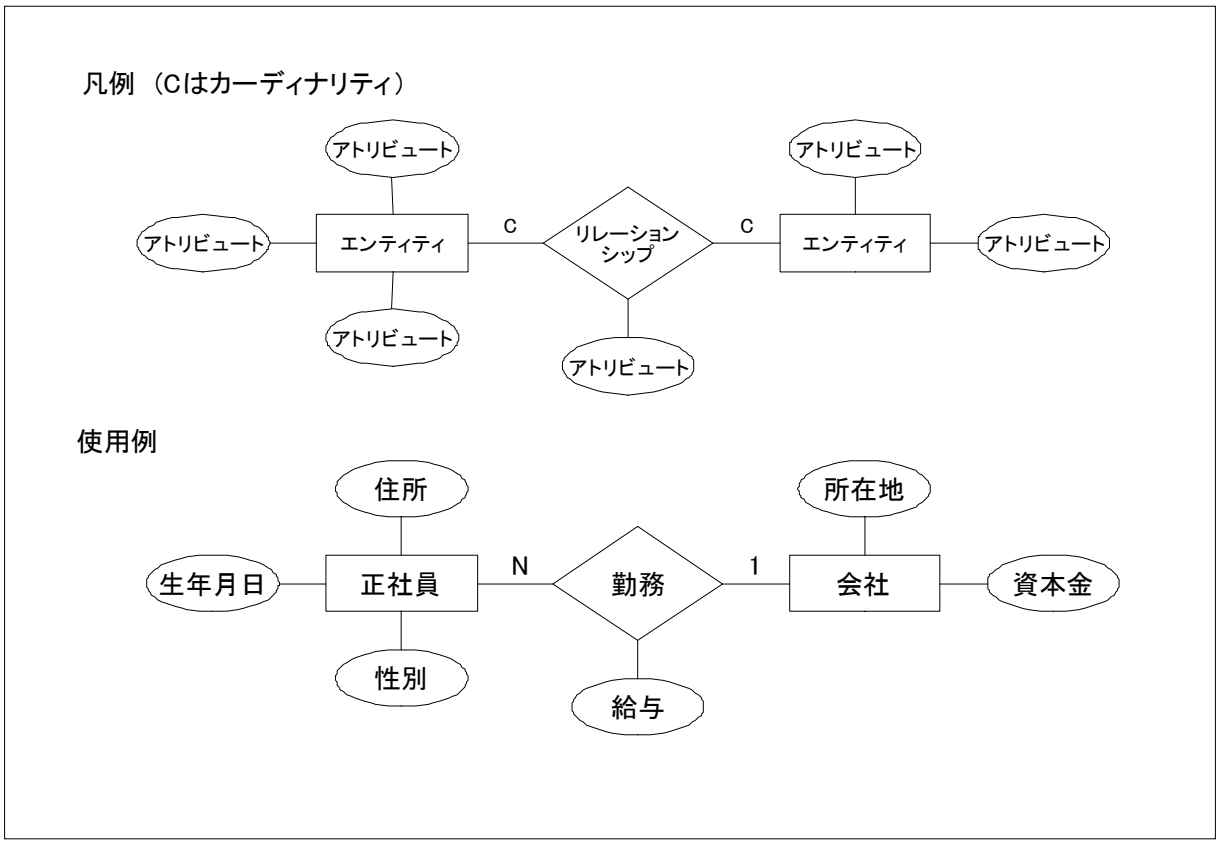

図 I-22-9.　ERD の記法

【解説】

1) ER(Entity-Relationship)モデル

データ構造をエンティティとリレーションシップとで表現するデータモデリングの一種。

＊ エンティティ (Entity)
データベースで管理すべき事象。名詞に相当する。(例: 銀行、支店、振込)

＊ リレーションシップ (Relationship)
エンティティ間の関連。動詞に相当する。(例: 銀行は複数の支店を「持つ」)

＊ アトリビュート
エンティティやリレーションシップの特性や状態。(例: 銀行の「銀行コード」「銀行名」)

2) ERD (Entity-Relationship Diagram、ER 図)

ER モデルを表現した図のこと。ERD にはさまざまな表記法があるが、図では Peter Chen による表記法を示している。

＊ エンティティ
矩形を書き、その中にエンティティの名称を書く。

＊ リレーションシップ
ひし形を書き、その中にリレーションシップの名称を書き、ひし形の頂点を各エンティティと直線で結ぶ。

＊ アトリビュート
楕円を書き、その中にアトリビュートの名称を書き、エンティティやリレーションシップと直線で結ぶ。

3) 関係データベースの設計における ER モデルの適用

関係データベースの設計においては、関係データベースに適した形で ER モデルが利用されている。エンティティを表(の行)、リレーションシップを表と表との結合、アトリビュートを表の列としてあてはめていく。関係データベース向けの ERD 作成ツールも存在し、OSS の DBDesigner4 などがある。

＊ カーディナリティ
リレーションシップは、エンティティの対応関係により、「1 対 1」「1 対多」「多対多」のいずれかに分類できる。例えば、1 つの銀行エンティティが、複数の支店エンティティに対応するときは「1 対多」となる。この対応関係をカーディナリティという。ERD では、エンティティとリレーションシップとを結ぶ直線のそばに「1」「N」を書くことで、カーディナリティを表現する。

＊ リレーションシップが「多対多」の対応関係となる場合、関係データベースとしては、このリレーションシップ自体をエンティティとし、専用の表を作成することが望まれる。これを連関エンティティという。

＊ オプショナリティ
リレーションシップにおいて、一方のエンティティに対応する他方のエンティティが存在する必要があるかどうかという、対応関係の任意性のことをオプショナリティという。

スキル区分	OSS モデルカリキュラムの科目	レベル
RDB 分野	22　RDB に関する知識 I	基本
習得ポイント	I-22-10. データベース正規化の具体的な考え方と手順	
対応する コースウェア	第 8 回（正規化の手順と方法）	

I-22-10. データベース正規化の具体的な考え方と手順

関係データベース設計の必須技術である「正規化」について、具体的な考え方と正規形の種類、関数従属といった基本的な概念を説明する。また正規化の手順を示し、各手順におけるポイントや正規化を崩すケースなどについて述べる。

【学習の要点】

＊ 関係データベースを設計する上で、データの冗長性を減らす正規化という作業が、品質確保のために重要となる。

＊ 正規化には段階があり、一般に、第 1 正規化、第 2 正規化、第 3 正規化の順で行われる。

＊ 実際の設計においては、あえて正規化を崩す(データに冗長性を持たせる)場合もある。

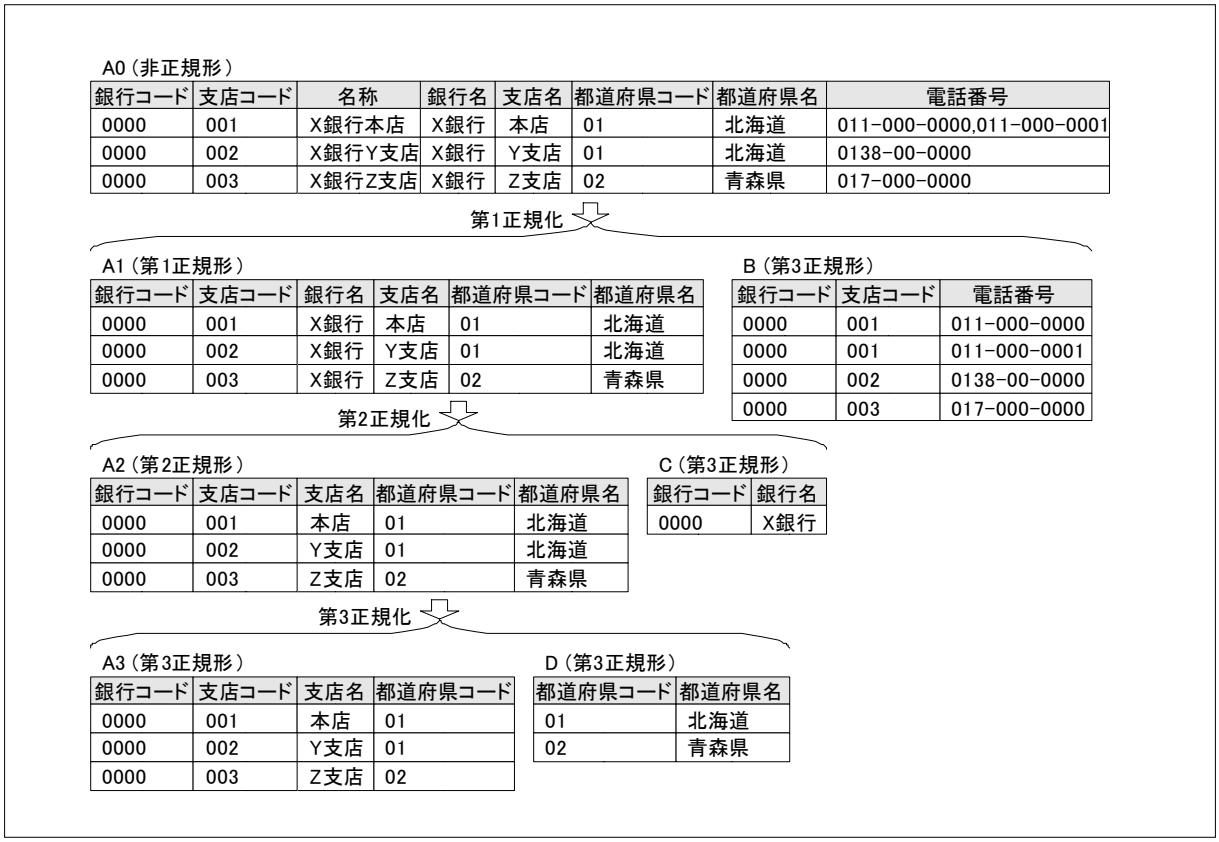

図 I-22-10.　正規化の例

【解説】

1) 正規化とそのメリット

関係データベースにおける正規化とは、データの整合性を保ったまま、データの冗長性を減らす作業のことである。冗長性を減らすことで、関係代数による演算に適した形となり、データの操作やデータベースの保守が容易になり、データベースやアプリケーションの品質を確保することが期待される。正規化の具体的な手順としては、第 1 正規化→第 2 正規化→第 3 正規化の順に行うのが一般的であり、以下、図を例に説明する。

＊ 第 1 正規化

表の各列の値から分割可能な値を排除する作業を第 1 正規化といい、正規化されていない表(非正規形という)に第1正規化を適用した結果を第1正規形という。ここで、分割可能かどうかは、分割した結果のデータが必要かどうかで判断される。第 1 正規形を満たさない表では、関係代数の演算の対象にできない値が存在しうるという問題がある。図の表 A0 は非正規形であり、[名称]列の値は銀行名と支店名に分割され、[銀行名][支店名]列と重複しており、また、[電話番号]列には複数の値が入っているものがある。表 A0 を第 1 正規化することで、表 A1 と表 B とに分離される。

＊ 関数従属

表において、列(または列の組み合わせ)P、Q があり、P の値が定まると Q の値も一意に定まる場合、(P に対して)Q は関数従属であるという。(ここで主キーは、実際に主キーに設定しなくとも、主キーとなりうるものを指す。)

- 部分関数従属　　主キーの一部の列に対して関数従属であること。
- 完全関数従属　　主キー全体に対して関数従属であり、部分関数従属でないこと。
- 推移関数従属

  A に対し B が関数従属、かつ、B に対し C が関数従属の場合、A に対し C は推移関数従属であるという。

＊ 第 2 正規化

表の部分関数従属である列を他の表に分離する作業を第 2 正規化といい、第 1 正規形に第 2 正規化を適用した結果を第 2 正規形という。図の表 A1 は第 1 正規形であり、[銀行コード]列に対し[銀行名]列は関数従属である。[銀行コード]列と[支店コード]列との組み合わせは主キーとなりうるので、[銀行名]列は部分関数従属である。表 A1 を第 2 正規化することで、表 A2 と表 C とに分離される。

＊ 第 3 正規化

表の推移関数従属である列を他の表に分離する作業を第 3 正規化といい、第 2 正規形に第 3 正規化を適用した結果を第 3 正規形という。図の表 A2 は第 2 正規形であり、[都道府県コード]列に対し[都道府県名]列は関数従属である。主キーに対し[都道府県コード]列は関数従属であるので、[都道府県名]は推移関数従属である。表 A2 を第 3 正規化することで、表 A3 と表 D とに分離される。

2) 正規化のデメリットと非正規化

正規化には前述のメリットがある反面、多くの表に分解されるため、RDBMS の実装上の問題で、検索速度の低下等を招く可能性もある。そこで、正規形を敢えて冗長化する場合がある。この作業を非正規化という。