

まえがき

本書は、データベースシステム開発における設計と運用管理を初めて学ぶ人の入門書として書かれている。大学の学部レベルの情報系学生向けの教科書として、また、データベースシステムの開発や運用管理に携わる企業の技術者向けの参考書として読まれることを想定している。特に、経済産業省が認定する国家試験「情報処理技術者試験」の一つである「データベーススペシャリスト試験 (DB)」の受験者向けの参考書として活用していただけるよう内容を配慮した構成となっている。

記述にあたっては、具体的な例を多く示し、また、図表を用いて丁寧に説明している。

1章では、データベースシステム概要について述べている。データベースシステム開発および運用管理の導入部分となるため、情報の蓄積やデータ構造に関する基本的な事項の確認と、ハードウェアとしてのディスク構造やファイルシステムに関する事項をまとめた。また、リレーショナルデータベース (RDB) システムの構成について触れた。

2章では、リレーショナルデータベース管理システム (RDBMS) の重要な基本機能について詳細に説明した。

3章では、データベースの実装・テストに関して、具体的な操作例を含めて説明した。RDBMSとしては、代表的なオープンソースソフトウェアであるMySQLを用いた。

4章では、データベースシステムの運用管理に関して、データベース技術者および管理者が知っておくべきことをハードウェア面およびソフトウェア面においてそれぞれ説明した。

5章では、データベースシステム開発における応用技術についてまとめるとともに、最近の技術的な話題についても触れた。

6章では、データベース関連技術として、意思決定支援やオブジェクト指向とともに、Web との連携に関する技術について説明した。

姉妹編の『リレーショナルデータベースの実践的基礎』(巻末の参考文献1))と併せて読めば、データベース技術に関して、基礎から応用までひと通りマスターでき、データベーススペシャリスト試験対策として活用できる。その受験対策のために、2～6章までの各章末に、過去の情報処理技術者試験テクニカルエンジニア

(データベース) (平成 21 年度以降はデータベーススペシャリストに改称) に出題された関連問題を掲載している。各章の内容を学んだうえで、問題を解くことにチャレンジしてほしい。

なお、本書の特徴として、教科書としてご採用の先生方には、コロナ社を通じて、スライド教材や演習問題解説などをご提供する仕組みを備えている。

2010 年 8 月

速水 治夫, 納富 一宏

本書姉妹編の内容梗概と特徴

(1) 『リレーショナルデータベースの実践的基礎』

データベースがなぜ必要なのかを理解し、リレーショナルデータベースを実践的に使用できる実力が備わる内容になっている。特に、リレーショナルモデルの論理設計および SQL を具体例で学ぶことができる。SQL の実行例はオープンソースのデータベース管理システムである MySQL を使用して示している。

(2) 『Web データベースの構築技術』

本書では、写真やイラストなどのマルチメディアデータを Web ブラウザから登録したり検索したりする Web データベースを実際に構築しながら、そのために必要な技術を学ぶ。具体的には、オープンソースで自由に使える Apache, MySQL, PHP を利用して、一連の Web データベースプログラミングを基礎から学ぶ。また、画像処理やセッション管理など高度な内容も扱う。これらにより、マルチメディアデータを扱う Web データベースを独自に設計・開発するスキルの習得が可能である。

【特徴】

本書と (1) を併せて学ぶことによって、「情報処理技術者試験」の「データベーススペシャリスト試験 (DB)」の出題範囲を理解することができる。

また、両著書の特徴として、教科書としてご採用の先生方には、コロナ社を通じて、講義を進めるための PowerPoint の資料などの教材をご提供する仕組みを備えている。

目 次

1 章 データベースシステム概要

1.1	データベースと情報システム	1
1.1.1	データの収集・蓄積・抽出	1
1.1.2	データの構造化	4
1.1.3	リレーショナル代数	5
1.1.4	ファイルシステム	6
1.1.5	ファイルアクセス方式	8
1.1.6	ファイル編成	9
1.1.7	記憶効率とアクセス効率（検索効率）	11
1.2	リレーショナルデータベース（RDB）利用の概要	12
1.2.1	RDB システムの基本構造	12
1.2.2	RDB システム構築の関係者とその役割	14

2 章 リレーショナルデータベース管理システムの基本機能

2.1	データベースの定義と操作	15
2.1.1	データベース定義機能	15
2.1.2	データ操作機能	17
2.2	トランザクションと同時実行制御	19
2.2.1	マルチタスク	19
2.2.2	トランザクションサポート機能	20
2.2.3	トランザクションの考え方	21
2.2.4	同時実行制御機能	23
2.3	データベースの回復機能	30
2.3.1	ロールバック	30
2.3.2	ロールフォワード	31
2.3.3	チェックポインティング	32
2.4	セキュリティとインテグリティ	34
2.4.1	セキュリティ機能	34
2.4.2	インテグリティ機能	35
2.5	DBMS のその他の機能	35
2.5.1	データベース管理機能	35

2.5.2 ユーティリティ機能	36
2.5.3 アプリケーション開発支援機能	36
2.6 演習問題	37

3章 データベースの実装・テスト

3.1 DBMS の選定と導入	45
3.1.1 オープンソースソフトウェアとフリーソフト	45
3.1.2 DBMS の選定	46
3.1.3 DBMS の導入	47
3.1.4 Web アプリケーション開発および関連ツール	48
3.2 物理データベースの設計	49
3.2.1 概念データモデル	51
3.2.2 論理データモデル	51
3.2.3 物理データモデル	52
3.2.4 スキーマモデル	52
3.2.5 物理データベース	54
3.3 データベースの実装	54
3.3.1 SQL 概要	54
3.3.2 データベースの管理	56
3.3.3 ユーザの管理	59
3.3.4 テーブルの管理	62
3.3.5 クエリの発行	65
3.4 テストと移行	67
3.4.1 データベースのテスト	68
3.4.2 モニタリング	68
3.4.3 データベースの移行	69
3.5 演習問題	74

4章 データベースシステムの運用管理

4.1 データベースシステムの運用計画	75
4.1.1 情報戦略と運用計画	75
4.1.2 管理部門と人員の配置	76
4.1.3 管理作業項目	76
4.2 データベースシステムの運用と保守	77
4.2.1 システムの監視	77
4.2.2 ハードウェアの保守と増設	80
4.2.3 性能確保	88

4.3 データベースシステムの管理	90
4.3.1 バックアップとリカバリ	90
4.3.2 バックアップ方法	91
4.3.3 バックアップメディア	92
4.3.4 インテグリティとセキュリティ	92
4.3.5 データベース監査への対応	93
4.4 性能チューニング	94
4.4.1 テーブル設計の見直し	94
4.4.2 インデックスの利用と見直し	94
4.4.3 ハードウェアアクセスの見直し	99
4.5 ユーザサポート	99
4.6 演習問題	101

5章 データベースシステム開発における応用技術

5.1 クライアント／サーバシステムと DBMS	104
5.1.1 コンピュータシステム構成の変遷	104
5.1.2 クライアント／サーバシステム (C/S システム)	105
5.2 分散データベース	106
5.2.1 集中型と分散型のシステム構成	106
5.2.2 分散データベースにおける透過性	108
5.2.3 分散トランザクション制御	108
5.2.4 レプリケーション	111
5.2.5 クラウドコンピューティング	112
5.3 高可用性システム	112
5.3.1 ハイ・アベイラビリティ	112
5.3.2 HA クラスタ	113
5.4 データベースセキュリティ	114
5.5 演習問題	118

6章 データベース関連技術

6.1 意思決定支援	124
6.1.1 データウェアハウス	124
6.1.2 データマイニング	125
6.2 オブジェクト指向とデータベース	125
6.2.1 オブジェクト指向とは	125
6.2.2 オブジェクト指向プログラミング	126

1

データベースシステム概要

データの蓄積と抽出はコンピュータを利用するうえで基本となる操作である。基本操作をモデル化するうえでデータベースという概念が考えられた。コンピュータの2次記憶装置[†]であるハードディスクとの入出力を行うことがデータベースの操作を行うことに相当する。効率よくこれらの操作を実行するために汎用的な実行環境が必要となり、データベース管理システムという形態へと発展した。データベースを活用したシステム形態は多様であり、それらのひな形となるシステム構築を可能にする基盤部分がデータベース管理システムである。

1.1 データベースと情報システム

1.1.1 データの収集・蓄積・抽出

われわれがコンピュータにデータ (**data**) を収集・蓄積する場合、後々、それらある条件に基づいて検索し取り出す (抽出する) ことが前提である。ただ単にデータを集めただけでそれで終わりということはありません。効率よく検索 (抽出) するためには、収集したデータを整理・整頓して蓄積し、すぐに見つけ出せるようあらかじめ見出しをつけたり、データの形式をそろえておくことがとても重要である。こうしたデータの収集先、蓄積先となり、加えて抽出もともなるデータの保管場所に存在するデータの集合体のことをデータベース (**database**) と呼ぶ。データの収集と蓄積について図 1.1 に示す。

この図に示した基本的な構成を持つシステムが情報システム (**information system**) である。データベースを有する情報システムがデータベースシステムである。なお、これらについては1.2節で詳しく述べる。

情報システムは言葉のとおり「情報処理を行うためのシステム」を意味するわけだが、そこで扱う情報やデータとはどのような性質を持つものであろうか。情報やデータを記憶したり、収集したり、整理・整頓したり、検索したりすることができるということについて少し考えてみよう。この基本的な性質が満たされる (保証さ

[†] 補助記憶装置とも呼ぶ。

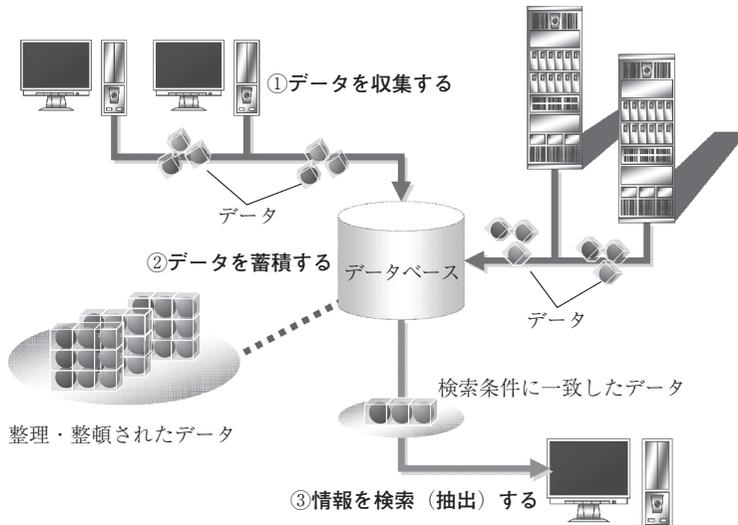


図 1.1 データの収集と蓄積

れる)ことが情報処理やシステム管理運用を考えるうえで大切である。情報システムが稼働しているか、または運用されている期間すべてにおいて、システム内の全データが保全されることの重要性を把握しなければならない。こうした意味で、以下、データの基本的性質について考えてみる。

(1) **データの基本的性質** データとは、人間やコンピュータが扱いやすい形で存在しなければならない。また、これらのデータが表現するものは、事実や概念や指令といった身近で具体的な**情報 (information)**を形式化したものだと考える

変数名

コンピュータにおけるデータの1次記憶装置には、**メモリ (memory)**があり、1 byte を最小単位として、メモリすべてに番地が付されている。ゆえに、番地でデータを区別することができるわけだが、メモリに記憶されているデータをどのように読み出すかは、読み出し側のプログラムの挙動に依存する。よって、データとして表現されている値が文字列なのか数値 (バイナリ) であるのか、もし数値 (バイナリ) であるなら、それが音声データなのか画像データなのかといった情報を便宜的に扱いやすくするための表現手段が変数に**変数名 (variable name)**をつけることなのである。こうしたことに注目して考えてみると、プログラミング言語で表現されているプログラムは、コンピュータが扱うデータをユーザの都合に応じて扱いやすくするために、変数への命名法など、表現上の何らかの工夫が必要であることがわかる。どのような場合も情報への意味づけが重要だと考えるべきであろう。

ことができる。形式化するというこの意味は、現実の世界をコンピュータの中に「情報として」取り入れるための根本的な手段である。その原理を理解することがコンピュータを活用するうえで重要である。

(2) **属性と構造化** データベースに格納されるデータは、データの種類としての**属性 (attribute)**を持つ。これらの属性は変数の**型 (type)**として表現され、他と区別される。リレーショナルデータベース (**relational database : RDB**)[†]はデータの属性間の**関係 (relationship)**を定義することで、データ相互の関連性を明確に表現することができる。こうした表現形式を取り入れることで、データへの意味づけを明示化するとともに、データの構造化を目指している。

(3) **データ構造の拡張** 構造化されたデータの集合全体を一つのデータ構造とみなすことができる。この操作を繰り返していくことで相互関係をさらに拡張することができる。この様子を図 1.2 に示す。

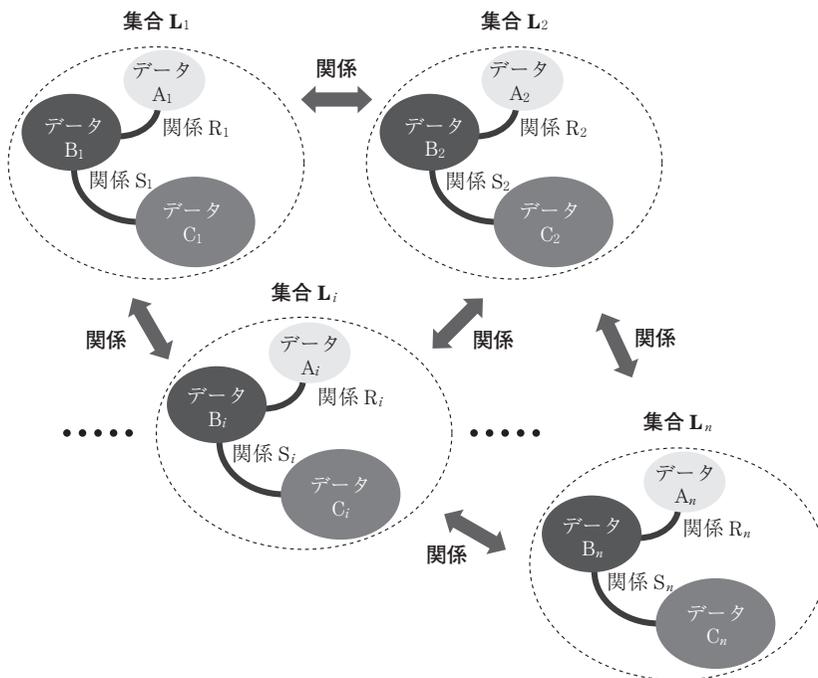


図 1.2 集合間の相互関係の拡張

「データを構造化すること」、「データを大量に集積すること」、「それぞれの関連づけを行って構造を拡張すること」などの基本的な操作がコンピュータによるデータ処理、ひいてはソフトウェア開発における基盤となる発想につながっている。

[†] 関係データベースとも呼ぶ。

データベースはこうした考え方に基づいて実装されている。

1.1.2 データの構造化

構造化データ (structured data) とは、構造を持つデータのことであり、形式的な操作の対象を表現する手段として有用である。先に述べたとおり、コンピュータプログラムの処理を記述する際に、データ形式がそろっていることは非常に重要である。RDB では、**表 (テーブル: table)** を定義することでデータの構造化を実現する。構造化データを表現する個々の属性は、グループ中のメンバーの特性を列挙する形で定義づけを行うものである。

データの構造化が重要であるという考え方は古くから存在し、プログラミング言語におけるデータ定義表現の一つとして実現されている。例えば、Pascal 言語におけるレコード型の定義や C 言語における構造体の定義などがこれに相当する。また、この考え方は、そのまま**オブジェクト指向 (object oriented)** における**クラス (class)** にも受け継がれている。オブジェクト指向については、6 章で扱う。

特に、システム実装などプログラミングが主体となる場面、あるいはシステム運用管理における場面では、ストレージ (ディスク装置) へのデータのストア、リストアが重要であり、その際に用いられる表の概念は、ファイル処理やファイル編成を考えるうえで一つの指針を示している。従来、こうした背景のもとにリレーショナルモデルにおける定式化が加わっているため、表の構成要素の名称にもさまざま

C 言語における集合と構造体

C 言語は最もポピュラーなプログラミング言語の一つである。多くの手続き型プログラミング言語 (procedural programming language) であればどれも同じだが、頻繁に使われるデータ表現方法として配列 (array) がある。同一種類のデータが多数集まっているデータ構造であり、通常はゼッケン番号に見立てた添え字番号で個々のデータを区別する。名前ではなく番号で区別するところがおもしろい。種類が同じなので属性がそろっているわけである。

配列の添え字番号はメモリの番地に相当すると考えれば、基本的な区別手段であるともいえる。また、同じ配列でも、PHP 言語などで利用される「連想配列」では、添え字番号ではなく、名前 (文字列) で区別する。

これに対して、構造体 (structure) というデータ表現方法もあり、こちらは種類が異なるものを一つに集めた構造をしている。さらにこれらを組み合わせた構造体配列なるものもある。このようにプログラミングの世界でもデータの集合や相互関係による構造化と拡張という考え方が大切な基本になっている。

な用語が混在していると考えられる。構造化データを表のアナロジーでとらえることも同様である。

個々のデータは定義された表の形式を持ち、扱うすべてのデータが同一の構造を持つことになる。構造化データの具体的な要素の一つ一つを組（**タプル**：**tuple**）、行（**ロー**：**row**）、またはレコード（**record**）と呼ぶ。表を形づくる項目となる個々の構成要素を属性（**attribute**）、列（**カラム**：**column**）、またはフィールド（**field**）と呼ぶ。フィールドはデータを定義する属性であるか、あるいは表を形づくる項目のことであると考えればよい。

このように、表を構成する行と列にはさまざまな呼び方があるわけだが、これは表というデータ構造をどのように意味づけし解釈するかという点にかかわっていることがその理由である。

例えば、リレーショナルデータベースのオリジナルでは、「タプル」と「属性」という名称がよく使われるが、SQLでは、「行」と「列」が使われる。また、スプレッドシート（spreadsheet）ソフトウェア（いわゆる表計算ソフト）では、「行」と「列」、ファイル処理のプログラミングでは、「レコード」と「フィールド」がより一般的に用いられるようである。このように、分野が異なれば、ほぼ同一の概念に対する呼称がその発想の由来に従って異なっている。本書でも、「表」を構成する要素として、これらの用語を適宜言い換えて用いることにする。

1.1.3 リレーショナル代数

データ集合は表としての構造化がなされることで操作や演算を考えることができる。これらをリレーショナル代数（**relational algebra**）と呼ぶ。リレーショナル代数には、大きく分けて集合演算と関係演算とがある。集合演算としては、和（**union**）、差（**difference**）、積（**intersection**）、直積[†]（**direct product**）の四つが挙げられる。また、関係演算としては、選択（**selection**）、射影（**projection**）、結合（**join**）、商（**division**）の四つが挙げられる。これらを表 1.1 にまとめる。

集合演算では、集合論に基づく、データ集合の組合せに関する演算を扱う。ここでは、一つの集合を表として考える。関係演算では、集合からのデータの抽出に関する操作を扱う。

リレーショナルモデルおよびリレーショナル代数の定式化については、文献1) に詳しく述べられている。具体例も豊富であるので、データベースの基礎理論に関

[†] 直積をデカルト積（Cartesian product）とも呼ぶ。

表 1.1 リレーショナル代数の分類と演算名

分類	演算名
集合演算	和 (union)
	差 (difference)
	積 (intersection)
	直積 (direct product)
関係演算	選択 (selection)
	射影 (projection)
	結合 (join)
	商 (division)

してはそちらを参照されたい。

1.1.4 ファイルシステム

ここでは、コンピュータシステムにおいてデータはどのように記録されるのかという点に注目し、データベースとの関係を概観してみよう。

Windows や Linux など、**オペレーティングシステム (Operating System : OS)**[†]における**ファイル (file)**の管理・制御方式のことを**ファイルシステム (file system)**と呼ぶ。データはすべてファイルシステムの管理下に置かれ、ファイルという単位で保存・管理される。ファイルの読み書きや検索が容易に行えるようにするための蓄積・編成方式であると考えればよい。**ハードディスクドライブ (Hard Disk Drive : HDD)**や CD-ROM, あるいは DVD などの**ストレージデバイス (storage device : 記憶装置)**としてのディスク管理や、**ネットワーク (network)**を利用したデータ共有のためのファイル管理など、さまざまな種類や区分が存在する。

ファイルシステムは、OS の違いによって種類が異なるのが普通である。また、時代とともに絶えず改良や拡張がなされてきており、ハードウェアの進歩とともに変化していくものである。ファイルシステムの具体例を挙げると、Windows では、NTFS や FAT32 などが、また、Linux では ext2, ext3, XFS, ReiserFS などが、さらに、Mac OS では HFS や HFS Plus などがそれぞれ利用されている。Linux など Unix 系 OS の場合、多くの種類のファイルシステムに対応することが可能であるため、他の OS で利用していた HDD をそのまま利用してファイルの読み書きを行うことができる可能性が高い。

従来、コンピュータシステムでは、検索の高速化・効率化を目的に、データの読

[†] コンピュータのハードウェアとアプリケーションソフトとを管理・制御するための基本ソフトウェア。

索引

【あ】

アウトバウンド	116
アカウント	34
アクセス効率	11
アクセス方式	9
アクティブ	113
アトリビュート	51
アプリケーションソフトウェア	7

アプリケーション・フレームワーク	46
アポート	28, 110
アルゴリズム	126
暗号化	116
アンロック	25

【い】

位置透過性	108
一貫性	20
移動透過性	108
インターネット	45, 105, 112, 130
インタフェース	12, 81
インタフェースカード	89
インタプリタ	49, 132
インテグリティ	93
インデックス	88, 94
インバウンド	116
インフラストラクチャ	14
インヘリタンス	128
インポート	36

【う】

運用記録	68
------	----

【え】

エクスポート	36
エンティティ	51

【お】

応用ソフトウェア	7
オーバーヘッド	133
オブジェクト	126
オブジェクト指向	4, 125
オブジェクト指向データベース	129
オブジェクト指向プログラミング言語	126

オブジェクトデータベース	129
オブジェクトデータベース管理システム	129
オブジェクトリレーショナルデータベース	129
オープンソースソフトウェア	45
オペレーティングシステム	6

【か】

概念スキーマ	52
概念データモデル	50, 51
外部キー	16
外部記憶装置	9
外部スキーマ	53
型	3
稼働率	112
カラム	5, 16
関係	3
監査	93
監視	68

【き】

キー	16
記憶効率	11
記憶装置	6
基底クラス	128
基本クラス	128
行	5, 16
業界標準	46
共通鍵暗号	116
共有ロック	25

【く】

クアドコア	86
クエリ	65
組	5
クライアント	105, 130
クライアント/サーバ	13, 35
クライアント/サーバシステム	105
クライアント/サーバ・モデル	105
クラウドコンピューティング	112
クラス	4, 127

【け】

計算量	11
継承	128

軽量言語	37, 49
結合	5
権限	34
検索	18
検索効率	11
原子性	20

【こ】

公開鍵	116
公開鍵暗号	116
高可用性	68, 113
更新	18
構造化データ	4
構造化プログラミング	126
構造体	4
後退復帰	30
候補キー	16
国際標準化機構	54
個人情報	114
コマンド	47
コマンドインタプリタ	55
コマンドラインインタフェース	47
コマンドラインツール	55
コミット	30
コンパイル	132
コンピュータネットワーク	12, 106
コンプライアンス	93
コンフリクト	96

【さ】

差	5
索引	94
削除	17
サーチ	9
サーバ	105, 130
サーバサイドスクリプティング	48, 126, 133
サービス	36, 106, 112, 130
サブクラス	128
サブプログラム	36
差分バックアップ	91

【し】

シェル	57
磁気テープ	9, 92
シーク	9

ハードディスクドライブ 6, 80, 89
 バランス木 95
 パリティ符号 83

【ひ】

光ディスク 92
 ビットマップインデックス 97
 秘密鍵 117
 表 4, 16, 52, 62
 表計算ソフト 5

【ふ】

ファイアウォール 116
 ファイル 6, 49
 ファイルシステム 6, 33
 ファイルシステム階層標準 69
 ファイル編成 7, 10
 フィールド 5, 62
 フェイルオーバー 114
 フェイルバック 114
 復号 116
 複製透過性 108
 不正アクセス 114
 物理スキーマ 53
 物理データベース 54
 物理データモデル 50, 52
 フリーソフトウェア 45
 フルダンプ 91
 フルバックアップ 91
 プレゼンテーション層 135
 プロセッサ 77, 80, 86, 89
 フローチャート 126
 プロパティ 127
 プロンプト 57
 分散DBMS 70
 分散システム 108
 分散処理 105
 分散データベース 106
 分散トランザクション 108

【へ】

並行透過性 108

米国規格協会 54
 並列処理制御 25
 ベースクラス 128
 変数名 2

【ほ】

ポインタ 96
 ポリシー 99, 115

【ま】

マイクロプロセッサ 86
 マスター 111, 136
 待ちグラフ 29
 マッシュアップ 112
 マルチコア 86
 マルチタスク 13, 19
 マルチプロセッサ 86
 マルチメディアデータ 106
 マルチユーザ 13, 19
 マルチユーザモード 91

【み】

ミドルウェア 37, 45, 55, 134
 ミラーリング 83

【め】

メインフレーム 9
 メソッド 127
 メタ 16
 メモリ 2, 85
 メモリモジュール 89
 メンテナンス 91, 100

【も】

モジュール 36
 モニタ 36, 55
 モニタリング 68
 モニタリングツール 77

【ら】

ライブラリ 37, 55
 ランダムアクセス 9

【り】

リカバリ 36, 71, 90
 リストア 91
 リソース 26
 リダイレクト 71
 リーフ 95
 粒度 30
 リレーショナル代数 5
 リレーショナルデータベース 3
 リレーショナルデータベース管理システム 12
 リレーションシップ 51
 リンク 51, 96

【る】

ルータ 14
 ルート 95

【れ】

レコード 5, 62
 列 5, 16
 レプリケーション 111, 136

【ろ】

ロー 5, 16
 ログイン 69
 ログ 30, 68, 88
 ログファイル 32
 ロジック層 135
 ロック 25
 ロック解除 25
 ロックレベル 30
 ロード 9
 ロールバック 30
 ロールフォワード 31
 論理スキーマ 52
 論理データモデル 50, 51

【わ】

和 5
 ワイルドカード 61, 72
 ワークベンチ 46

【A】

ACID 特性 20
 ANSI 54
 ANSI/SPARC 53
 API 37, 55, 112, 134

【B】

B⁺ tree 96

B⁺ 木 96
 B-tree 95
 B 木 95

【C】

CGI 131
 CLI 47
 CPU 77, 80, 86
 C/S 13

【D】

CUI 47
 DA 76
 DAT 92
 DBA 76
 DBMS 12
 DCL 54
 DDL 54

DML	54			SATA	81
【E】			【L】	SCSI	81
ECC	85	LAN	14, 105, 106	SDK	46
ER 図	51		【M】	SQL	12, 18
ER モデル	51	MAC アドレス	115	SSD	92
e コマース	130	MPU	86	【T】	
【F】			【N】	TSS	104
FHS	69	NA	76	【U】	
【G】			【O】	undo	34
GUI	47	ODBMS	129	【W】	
【H】		OOPL	126	WAN	106
HA	113	ORDB	129	Web	106
HA クラスタ	113	OS	6	Web アプリケーション	48
HDD	6	OSS	45	Web サイト	130
HTML	131	【R】		Web サーバ	131
HTTP	106, 131	RAID	83	Web データベース	48
【I】		RAID0	83	World Wide Web	106
IDE	81	RAID1	83	WWW	106
IDS	116	RAID5	83	【数字】	
I/O	77	RAID6	83	1 次記憶装置	2, 85
IPS	116	RDB	3	1 相コミットメント制御	109
IP アドレス	115	RDBMS	12	2 次記憶装置	89
ISO	54	RDB システム	12	2 相コミットメント制御	109
ISP	106	redo	34	2 相ロッキング・プロトコル	27
i ノード	8	【S】		3 層アーキテクチャ	135
		SA	76	3 層スキーマアーキテクチャ	53
				3 層スキーマモデル	52

— 編著者・著者略歴 —

速水 治夫 (はやみ はるお)

1970年 名古屋大学工学部応用物理学科卒業
1972年 名古屋大学大学院工学研究科博士前期
課程修了(応用物理学専攻)
1972年
～98年 日本電信電話公社(現NTT)勤務
1993年 博士(工学)
1994年
～98年 電気通信大学客員教授(兼務)
1997年 工学院大学非常勤講師(兼務)
1998年 神奈川工科大学教授
現在に至る
2004年 Workflow Management Coalition (WfMC)
Fellow
2007年 情報処理学会フェロー

納富 一宏 (のうとみ かずひろ)

1987年 早稲田大学理工学部電子通信学科卒業
1989年 早稲田大学大学院理工学研究科博士前期
課程修了(電気工学専攻)
1989年
～92年 早稲田大学助手
1994年 早稲田大学大学院理工学研究科博士後期
課程単位取得退学(電気工学専攻)
1994年 神奈川工科大学助手
2003年 博士(工学)(早稲田大学)
2007年 神奈川工科大学准教授
2009年 神奈川工科大学教授
現在に至る

データベースの実装とシステム運用管理

Database Implementation and System Management © Hayami, Notomi 2010

2010年10月8日 初版第1刷発行



検印省略

編著者 速水 治夫
著者 納富 一宏
発行者 株式会社 コロナ社
代表者 牛来真也
印刷所 萩原印刷株式会社

112-0011 東京都文京区千石 4-46-10

発行所 株式会社 コロナ社

CORONA PUBLISHING CO., LTD.

Tokyo Japan

振替 00140-8-14844・電話 (03) 3941-3131 (代)

ホームページ <http://www.coronasha.co.jp>

ISBN 978-4-339-02450-0

(中原) (製本: 愛千製本所)

Printed in Japan



無断複写・転載を禁ずる

落丁・乱丁本はお取替えいたします