

# SQL 実践入門

# SQL 実践入門

高速でわかりやすいクエリの書き方

ミック  
[著]

## 実行計画を読み解き ボトルネックを解消する

条件分岐／集約とカット／ループ

集合指向／CASE式／ウィンドウ関数／相関サブクエリ

結合／Nested Loops／Hash／順序と連番

更新／データモデル／インデックススキャン

大量のデータを自在に処理するノウハウ

高速でわかりやすい  
クエリの書き方

ミック  
[著]

技術評論社

技術評論社

技術評論社

---

## 本書の構成

本書は全10章と2つのAppendixにより構成されています。

---

### 第1章：DBMSのアーキテクチャ——この世にただ飯はあるか

本書の導入として、RDBの内部的な動作に関するモデルを理解します。データキャッシュやワーキングメモリといったメモリ機構とストレージのしくみ、そして何より、SQLのパフォーマンスを理解するためのキー概念である実行計画とそれを構築するオプティマイザの概念を理解します。

---

### 第2章：SQLの基礎——母国語を話すごとく

SQLの基本構文を理解します。検索と更新、分岐、集約、行間比較といったSQLでデータ操作を行うための手段を確認します。本章で学ぶSQL文のさまざまな道具が、第3章以降でパフォーマンスを向上させる際の強力な武器となります。

---

### 第3章：SQLにおける条件分岐——文から式へ

SQLにおいて条件分岐を表現する強力な武器であるCASE式が、パフォーマンス改善においても重要な役割を持っていることを、実行計画を読み解くことで明らかにします。

---

### 第4章：集約とカット——集合の世界

SQLが体現する集合指向というパラダイムがもたらす考え方の変化を、GROUP BY句や集約関数の使い方を通して体感します。同時に、前章で学習したCASE式と集合指向との組み合わせが、どのようにパフォーマンスへ貢献するのかを理解します。

---

### 第5章：ループ——手続き型の呪縛

RDBのパフォーマンスを劣化させる理由の一つが、SQLの集合指向の世界に無理に手続き型のパラダイムを持ち込むことにあります。本章では、その典型的な症状である「ループ依存症」について考察します。

---

### 第6章：結合——結合を制する者はSQLを制す

SQLのパフォーマンスが遅いとき、そこにはほぼ必ずと言ってよいほど結合が関係しています。Nested Loops、Hash、Sort Mergeといった結合アルゴリズムの実行計画を読み解くことで、RDBがどのように結合を最適化しようとするかを取り上げます。

---

## 第7章：サブクエリ——困難は分割するべきか

---

問題を小さな規模に分割し、ステップ・バイ・ステップで解決へと至るサブクエリのアプローチは、手続き型に近いものです。これに頼りすぎた場合に引き起こされる性能問題——サブクエリ・パラノイアについて考察し、いかにしてそれを解消するかを論じます。

---

## 第8章：SQLにおける順序——甦る手続き型

---

伝統的に手続き型と相容れないパラダイムを持っていると思われていたSQLですが、近年、再び手続き型の考え方を取り入れる変化が起きています。その動きを象徴するのがウィンドウ関数です。この強力な表現力を持つ関数がSQLにもたらした革命的な進歩を中心に、SQLにおける行の順序を意識したプログラミングを考えます。

---

## 第9章：更新とデータモデル——盲目のスーパーソルジャー

---

パフォーマンスを改善する一番良い手段——それは実は、SQL文を変えないことではなく、データモデルを変えることです。ときに忘れられがちな「コロンブスの卵」が持つ利点と欠点を明らかにします。

---

## 第10章：インデックスを使いこなす——秀才の弱点

---

パフォーマンスを語るうえで避けて通ることのできないインデックスの利用方法について論じます。どのような条件下で有効に作用するかをインデックスの構造から理解し、そのために必要なデータモデルおよびユーザーインタフェースの設計についても取り上げます。

---

## Appendix A：PostgreSQLのインストールと起動

---

学習用の実行環境として、PostgreSQL 9.3のインストールおよび起動手順を解説します。

---

## Appendix B：演習問題の解答

---

各章末の演習問題の解答および解説を行います。

## SQL実践入門——高速でわかりやすいクエリの書き方●目次

|                      |     |
|----------------------|-----|
| はじめに .....           | iii |
| 謝辞 .....             | iv  |
| サンプルコードのダウンロード ..... | iv  |
| 本書の構成 .....          | v   |

## 第1章

## DBMSのアーキテクチャ——この世にただ飯はあるか ... 1

|     |                                    |    |
|-----|------------------------------------|----|
| I.1 | <b>DBMSのアーキテクチャ概要</b> .....        | 2  |
|     | クエリ評価エンジン .....                    | 4  |
|     | バッファマネージャ .....                    | 4  |
|     | ディスク容量マネージャ .....                  | 4  |
|     | トランザクションマネージャとロックマネージャ .....       | 4  |
|     | リカバリマネージャ .....                    | 5  |
| I.2 | <b>DBMSとバッファ</b> .....             | 6  |
|     | この世にただ飯はあるか .....                  | 6  |
|     | DBMSと記憶装置の関係 .....                 | 7  |
|     | HDD .....                          | 7  |
|     | メモリ .....                          | 8  |
|     | バッファの活用による速度向上 .....               | 8  |
|     | メモリ上の2つのバッファ .....                 | 10 |
|     | データキャッシュ .....                     | 11 |
|     | ログバッファ .....                       | 11 |
|     | メモリの性質がもたらすトレードオフ .....            | 12 |
|     | 揮発性とは .....                        | 12 |
|     | 揮発性の問題点 .....                      | 13 |
|     | システム特性によるトレードオフ .....              | 14 |
|     | データキャッシュとログバッファのサイズ .....          | 14 |
|     | 検索と更新、大事なのはどっち .....               | 16 |
|     | もう一つのメモリ領域「ワーキングメモリ」 .....         | 16 |
|     | いつ使われるか .....                      | 16 |
|     | 不足すると何が起きるのか .....                 | 18 |
| I.3 | <b>DBMSと実行計画</b> .....             | 19 |
|     | 権限委譲の功罪 .....                      | 19 |
|     | データへのアクセス方法はどう決まるのか .....          | 20 |
|     | パーサ(parser) .....                  | 21 |
|     | オプティマイザ(optimizer) .....           | 21 |
|     | カタログマネージャ(catalog manager) .....   | 21 |
|     | プラン評価(plan evaluation) .....       | 22 |
|     | オプティマイザとうまく付き合う .....              | 22 |
|     | 適切な実行計画が作成されるようにするには .....         | 23 |
| I.4 | <b>実行計画がSQL文のパフォーマンスを決める</b> ..... | 24 |
|     | 実行計画の確認方法 .....                    | 25 |
|     | テーブルフルスキャンの実行計画 .....              | 26 |
|     | 操作対象のオブジェクト .....                  | 27 |
|     | オブジェクトに対する操作の種類 .....              | 27 |



|                                   |    |
|-----------------------------------|----|
| <b>Column</b> 実行計画の「実行コスト」と「実行時間」 | 28 |
| 操作の対象となるレコード数                     | 29 |
| <b>インデックススキャンの実行計画</b>            | 30 |
| 操作の対象となるレコード数                     | 30 |
| 操作対象のオブジェクトと操作                    | 31 |
| <b>簡単なテーブル結合の実行計画</b>             | 32 |
| オブジェクトに対する操作の種類                   | 34 |
| <b>I.5 実行計画の重要性</b>               | 34 |
| 第1章のまとめ                           | 36 |
| 演習問題1                             | 36 |
| <b>Column</b> いろいろなキャッシュ          | 37 |

## 第2章

# SQLの基礎——母国語を話すがごとく

|                                 |    |
|---------------------------------|----|
| <b>2.1 SELECT文</b>              | 40 |
| SELECT句とFROM句                   | 42 |
| WHERE句                          | 42 |
| WHERE句のさまざまな条件指定                | 43 |
| WHERE句は巨大なベン図                   | 44 |
| INでOR条件を簡略化する                   | 47 |
| NULL——何もないとはどういうことか             | 48 |
| <b>Column</b> SELECT文は手続き型言語の関数 | 49 |
| GROUP BY句                       | 50 |
| グループ分けするメリット                    | 51 |
| ホールケーキを全部1人で食べたい人は?             | 53 |
| HAVING句                         | 54 |
| ORDER BY句                       | 55 |
| ビューとサブクエリ                       | 56 |
| ビューの作り方                         | 57 |
| 無名のビュー                          | 57 |
| サブクエリを使った便利な条件指定                | 58 |
| <b>2.2 条件分岐、集合演算、ウィンドウ関数、更新</b> | 60 |
| SQLと条件分岐                        | 60 |
| CASE式の構文                        | 60 |
| CASE式の動作                        | 61 |
| SQLで集合演算                        | 62 |
| UNIONで和集合を求める                   | 62 |
| INTERSECTで積集合を求める               | 64 |
| EXCEPTで差集合を求める                  | 64 |
| ウィンドウ関数                         | 65 |
| トランザクションと更新                     | 68 |
| INSERTでデータを挿入する                 | 69 |
| DELETEでデータを削除する                 | 71 |
| UPDATEでデータを更新する                 | 72 |
| 第2章のまとめ                         | 75 |
| 演習問題2                           | 75 |

## 第3章

## SQLにおける条件分岐——文から式へ..... 77

|     |                                |    |
|-----|--------------------------------|----|
| 3.1 | <b>UNIONを使った冗長な表現</b>          | 78 |
|     | UNIONによる条件分岐の簡単なサンプル.....      | 79 |
|     | UNIONを使うと実行計画が冗長になる.....       | 80 |
|     | UNIONを安易に使うべからず.....           | 81 |
|     | WHERE句で条件分岐させるのは素人.....        | 82 |
|     | SELECT句で条件分岐させると実行計画もすっきり..... | 82 |
| 3.2 | <b>集計における条件分岐</b>              | 84 |
|     | 集計対象に対する条件分岐.....              | 85 |
|     | UNIONによる解.....                 | 85 |
|     | UNIONの実行計画.....                | 86 |
|     | 集計における条件分岐もやはりCASE式.....       | 86 |
|     | CASE式の実行計画.....                | 87 |
|     | 集約の結果に対する条件分岐.....             | 87 |
|     | UNIONで条件分岐させるのは簡単だが.....       | 88 |
|     | UNIONの実行計画.....                | 89 |
|     | CASE式による条件分岐.....              | 90 |
|     | CASE式による条件分岐の実行計画.....         | 90 |
| 3.3 | <b>それでもUNIONが必要なのです</b>        | 91 |
|     | UNIONを使わなければ解けないケース.....       | 91 |
|     | UNIONを使ったほうがパフォーマンスが良いケース..... | 92 |
|     | UNIONによる解.....                 | 93 |
|     | ORを使った解.....                   | 95 |
|     | INを使った解.....                   | 96 |
| 3.4 | <b>手続き型と宣言型</b>                | 97 |
|     | 文ベースと式ベース.....                 | 98 |
|     | 宣言型の世界へ跳躍しよう.....              | 98 |
|     | 第3章のまとめ.....                   | 99 |
|     | 演習問題3.....                     | 99 |

## 第4章

## 集約とカット——集合の世界..... 101

|     |                                   |     |
|-----|-----------------------------------|-----|
| 4.1 | <b>集約</b>                         | 102 |
|     | 複数行を1行にまとめる.....                  | 103 |
|     | CASE式とGROUP BYの応用.....            | 106 |
|     | 集約・ハッシュ・ソート.....                  | 108 |
|     | 合わせ技1本.....                       | 109 |
| 4.2 | <b>カット</b>                        | 113 |
|     | あなたは肥り過ぎ？ 痩せ過ぎ？——カットとパーティション..... | 114 |
|     | パーティション.....                      | 115 |
|     | BMIによるカット.....                    | 117 |
|     | PARTITION BY句を使ったカット.....         | 119 |

|              |     |
|--------------|-----|
| 第4章のまとめ..... | 121 |
| 演習問題4.....   | 121 |

## 第5章

### ループ——手続き型の呪縛..... 123

#### 5.1 ループ依存症..... 124

|                                  |     |
|----------------------------------|-----|
| Q.「先生、なぜSQLにはループがないのですか?」.....   | 124 |
| A.「ループなんてないほうがいいな、と思ったからです」..... | 125 |
| それでもループは回っている.....               | 125 |

#### 5.2 ぐるぐる系の恐怖..... 127

|                            |     |
|----------------------------|-----|
| ぐるぐる系の欠点.....              | 130 |
| SQL実行のオーバーヘッド.....         | 131 |
| 並列分散がやりにくい.....            | 133 |
| データベースの進化による恩恵を受けられない..... | 133 |
| ぐるぐる系を速くする方法はあるか.....      | 134 |
| ぐるぐる系をガツン系に書き換える.....      | 134 |
| 個々のSQLを速くする.....           | 135 |
| 処理を多量化する.....              | 135 |
| ぐるぐる系の利点.....              | 136 |
| 実行計画が安定する.....             | 136 |
| 処理時間の見通り精度が(相対的には)高い.....  | 137 |
| トランザクション制御が容易.....         | 138 |

#### 5.3 SQLではループをどう表現するか..... 138

|                                 |     |
|---------------------------------|-----|
| ポイントはCASE式とウィンドウ関数.....         | 138 |
| Column 相関サブクエリによる対象レコードの制限..... | 141 |
| ループ回数の上限が決まっている場合.....          | 142 |
| 近似する郵便番号を求める.....               | 143 |
| ランキングの問題に読み替え可能.....            | 144 |
| ウィンドウ関数でスキャン回数を減らす.....         | 147 |
| Column インデックスオンリースキャン.....      | 148 |
| ループ回数が不定の場合.....                | 149 |
| 隣接リストモデルと再帰クエリ.....             | 150 |
| 入れ子集合モデル.....                   | 155 |

#### 5.4 バイアスの功罪..... 158

|              |     |
|--------------|-----|
| 第5章のまとめ..... | 161 |
| 演習問題5.....   | 161 |

## 第6章

### 結合——結合を制する者はSQLを制す..... 163

#### 6.1 機能から見た結合の種類..... 165

|                       |     |
|-----------------------|-----|
| クロス結合——すべての結合の母体..... | 165 |
| Column 自然結合の構成.....   | 166 |
| クロス結合の動作.....         | 167 |

|                              |     |
|------------------------------|-----|
| クロス結合が実務で使われない理由             | 168 |
| うっかりクロス結合                    | 169 |
| <b>内部結合</b> ——何の「内部」なのか      | 170 |
| 内部結合の動作                      | 170 |
| 内部結合と同値の関連サブクエリ              | 171 |
| <b>外部結合</b> ——何の「外部」なのか      | 172 |
| 外部結合の動作                      | 173 |
| <b>外部結合と内部結合の違い</b>          | 174 |
| <b>自己結合</b> ——自己とは誰のことか      | 174 |
| 自己結合の動作                      | 175 |
| 自己結合の考え方                     | 176 |
| <b>6.2 結合のアルゴリズムとパフォーマンス</b> | 177 |
| <b>Nested Loops</b>          | 178 |
| Nested Loopsの動作              | 178 |
| 駆動表の重要性                      | 179 |
| Nested Loopsの落とし穴            | 183 |
| <b>Hash</b>                  | 184 |
| Hashの動作                      | 184 |
| Hashの特徴                      | 186 |
| Hashが有効なケース                  | 186 |
| <b>Sort Merge</b>            | 187 |
| Sort Mergeの動作                | 187 |
| Sort Mergeの特徴                | 188 |
| Sort Mergeが有効なケース            | 188 |
| <b>意図せぬクロス結合</b>             | 188 |
| Nested Loopsが選択される場合         | 190 |
| クロス結合が選択される場合                | 190 |
| 意図せぬクロス結合を回避するには             | 191 |
| <b>6.3 結合が遅いなど感じたら</b>       | 193 |
| ケース別の最適な結合アルゴリズム             | 193 |
| そもそも実行計画の制御は可能なのか？           | 194 |
| DBMSごとの実行計画制御の状況             | 194 |
| 実行計画をユーザが制御することによるリスク        | 195 |
| 揺れるよ揺れる、実行計画は揺れるよ            | 195 |
| 第6章のまとめ                      | 197 |
| 演習問題6                        | 197 |
| <b>第7章</b>                   |     |
| <b>サブクエリ</b> ——困難は分割するべきか    | 199 |
| <b>7.1 サブクエリが引き起こす弊害</b>     | 201 |
| <b>サブクエリの問題点</b>             | 201 |
| サブクエリの計算コストが上乘せされる           | 201 |
| データのI/Oコストがかかる               | 201 |
| 最適化を受けられない                   | 201 |
| <b>サブクエリ・パラノイア</b>           | 202 |
| サブクエリを使った場合                  | 203 |
| 相関サブクエリは解にならない               | 206 |
| ウィンドウ関数で結合をなくせ！              | 207 |
| 長期的な視野でのリスクマネジメント            | 208 |



|     |                                     |     |
|-----|-------------------------------------|-----|
|     | アルゴリズムの変動リスク                        | 209 |
|     | 環境起因の遅延リスク                          | 210 |
|     | サブクエリ・パノミア——応用版                     | 211 |
|     | サブクエリ・パノミア再び                        | 211 |
|     | 行間比較でも結合は必要ない                       | 213 |
|     | 困難は分割するな                            | 215 |
| 7.2 | <b>サブクエリの積極的意味</b>                  | 215 |
|     | 結合と集約の順序                            | 216 |
|     | 2つの解                                | 218 |
|     | 結合の対象行数                             | 219 |
|     | 第7章のまとめ                             | 221 |
|     | 演習問題7                               | 221 |
|     | <b>第8章</b>                          |     |
|     | <b>SQLにおける順序——甦る手続き型</b>            | 223 |
| 8.1 | <b>行に対するナンバリング</b>                  | 225 |
|     | 主キーが1列の場合                           | 225 |
|     | ウィンドウ関数を利用する                        | 226 |
|     | 相関サブクエリを利用する                        | 226 |
|     | 主キーが複数列から構成される場合                    | 227 |
|     | ウィンドウ関数を利用する                        | 228 |
|     | 相関サブクエリを利用する                        | 228 |
|     | グループごとに連番を振る場合                      | 229 |
|     | ウィンドウ関数を利用する                        | 229 |
|     | 相関サブクエリを利用する                        | 230 |
|     | ナンバリングによる更新                         | 230 |
|     | ウィンドウ関数を利用する                        | 231 |
|     | 相関サブクエリを利用する                        | 232 |
| 8.2 | <b>行に対するナンバリングの応用</b>               | 232 |
|     | 中央値を求める                             | 232 |
|     | 集合指向的な解                             | 233 |
|     | 手続き型の解①——世界の中心を目指せ                  | 235 |
|     | 手続き型の解②——2マイナス1は1                   | 237 |
|     | ナンバリングによりテーブルを分割する                  | 239 |
|     | 断絶区間を求める                            | 239 |
|     | 集合指向的な解——集合の境界線                     | 240 |
|     | 手続き型の解——「1行あと」との比較                  | 242 |
|     | テーブルに存在するシーケンスを求める                  | 244 |
|     | 集合指向的な解——再び、集合の境界線                  | 244 |
|     | 手続き型の解——再び、「1行あと」との比較               | 245 |
| 8.3 | <b>シーケンスオブジェクト・IDENTITY列・採番テーブル</b> | 250 |
|     | シーケンスオブジェクト                         | 250 |
|     | シーケンスオブジェクトの問題点                     | 251 |
|     | シーケンスオブジェクトそのものに起因する性能問題            | 251 |
|     | シーケンスオブジェクトそのものに起因する性能問題への対策        | 253 |
|     | 連番をキーに使うことに起因する性能問題                 | 253 |
|     | 連番をキーに使うことに起因する性能問題への対策             | 255 |
|     | IDENTITY列                           | 255 |

|               |     |
|---------------|-----|
| 採番テーブル .....  | 256 |
| 第8章のまとめ ..... | 257 |
| 演習問題8 .....   | 257 |

## 第9章

### 更新とデータモデル——盲目のスーパーソルジャー .....

|     |                              |     |
|-----|------------------------------|-----|
| 9.1 | 更新は効率的に .....                | 260 |
|     | NULLの埋め立てを行う .....           | 260 |
|     | 逆にNULLを作成する .....            | 264 |
| 9.2 | 行から列への更新 .....               | 265 |
|     | 1列ずつ更新する .....               | 267 |
|     | 行式で複数列更新する .....             | 268 |
|     | NOT NULL制約がついている場合 .....     | 270 |
|     | UPDATE文を利用する .....           | 271 |
|     | MERGE文を利用する .....            | 272 |
| 9.3 | 列から行への更新 .....               | 274 |
| 9.4 | 同じテーブルの異なる行からの更新 .....       | 276 |
|     | 関連サブクエリを利用する .....           | 278 |
|     | ウィンドウ関数を利用する .....           | 279 |
|     | INSERTとUPDATEはどちらが良いのか ..... | 280 |
| 9.5 | 更新のもたらすトレードオフ .....          | 281 |
|     | SQLで解く方法 .....               | 283 |
|     | SQLに頼らずに解く方法 .....           | 285 |
| 9.6 | モデル変更の注意点 .....              | 286 |
|     | 更新コストが高まる .....              | 286 |
|     | 更新までのタイムラグが発生する .....        | 287 |
|     | モデル変更のコストが発生する .....         | 288 |
| 9.7 | スーパーソルジャー病：類題 .....          | 288 |
|     | 再び、SQLで解くなら .....            | 289 |
|     | 再び、モデル変更で解くなら .....          | 291 |
|     | 初級者よりも中級者のご用心 .....          | 291 |
| 9.8 | データモデルを制する者はシステムを制す .....    | 292 |
|     | 第9章のまとめ .....                | 294 |
|     | 演習問題9 .....                  | 294 |

## 第10章

### インデックスを使いこなす——秀才の弱点 .....

|      |                        |     |
|------|------------------------|-----|
| 10.1 | インデックスと言えばB-tree ..... | 298 |
|------|------------------------|-----|

|                                          |            |
|------------------------------------------|------------|
| 万能型のB-tree .....                         | 298        |
| その他のインデックス .....                         | 300        |
| <b>10.2 インデックスを有効活用するには</b> .....        | <b>300</b> |
| カーディナリティと選択率 .....                       | 300        |
| Column クラスティングファクタ .....                 | 301        |
| インデックスの利用が有効かを判断するには .....               | 302        |
| <b>10.3 インデックスによる性能向上が難しいケース</b> .....   | <b>302</b> |
| 絞り込み条件が存在しない .....                       | 303        |
| ほとんどレコードを絞り込めない .....                    | 304        |
| 入力パラメータによって選択率が変動する① .....               | 305        |
| 入力パラメータによって選択率が変動する② .....               | 305        |
| インデックスが使えない検索条件 .....                    | 306        |
| 中間一致、後方一致のLIKE述語 .....                   | 306        |
| 索引列で演算を行っている .....                       | 307        |
| IS NULL述語を使っている .....                    | 307        |
| 否定形を用いている .....                          | 308        |
| <b>10.4 インデックスが使用できない場合どう対処するか</b> ..... | <b>308</b> |
| 外部設計による対処——深くて暗い川を渡れ .....               | 309        |
| UI設計による対処 .....                          | 309        |
| 外部設計による対処の注意点 .....                      | 310        |
| データマートによる対処 .....                        | 311        |
| データマートを採用するときの注意点 .....                  | 312        |
| データ鮮度 .....                              | 312        |
| データマートのサイズ .....                         | 312        |
| データマートの数 .....                           | 313        |
| パッチウィンドウ .....                           | 314        |
| インデックスオンリースキャンによる対処 .....                | 314        |
| Column インデックスオンリースキャンとカラム指向データベース .....  | 317        |
| インデックスオンリースキャンを採用するときの注意点 .....          | 318        |
| DBMSによっては使えないこともある .....                 | 319        |
| 1つのインデックスに含められる列数には限度がある .....           | 319        |
| 更新のオーバーヘッドを増やす .....                     | 319        |
| 定期的なインデックスのリビルドが必要 .....                 | 320        |
| SQL文に新たな列が追加されたら使えない .....               | 320        |
| 第10章のまとめ .....                           | 321        |
| 演習問題10 .....                             | 321        |
| <b>Appendix A</b>                        |            |
| PostgreSQLのインストールと起動 .....               | 323        |
| <b>Appendix B</b>                        |            |
| 演習問題の解答 .....                            | 333        |
| 索引 .....                                 | 347        |
| 著者プロフィール .....                           | 353        |

|                                          |            |
|------------------------------------------|------------|
| 万能型のB-tree .....                         | 298        |
| その他のインデックス .....                         | 300        |
| <b>10.2 インデックスを有効活用するには</b> .....        | <b>300</b> |
| カーディナリティと選択率 .....                       | 300        |
| Column クラスティングファクタ .....                 | 301        |
| インデックスの利用が有効かを判断するには .....               | 302        |
| <b>10.3 インデックスによる性能向上が難しいケース</b> .....   | <b>302</b> |
| 絞り込み条件が存在しない .....                       | 303        |
| ほとんどレコードを絞り込めない .....                    | 304        |
| 入力パラメータによって選択率が変動する① .....               | 305        |
| 入力パラメータによって選択率が変動する② .....               | 305        |
| インデックスが使えない検索条件 .....                    | 306        |
| 中間一致、後方一致のLIKE述語 .....                   | 306        |
| 索引列で演算を行っている .....                       | 307        |
| IS NULL述語を使っている .....                    | 307        |
| 否定形を用いている .....                          | 308        |
| <b>10.4 インデックスが使用できない場合どう対処するか</b> ..... | <b>308</b> |
| 外部設計による対処——深くて暗い川を渡れ .....               | 309        |
| UI設計による対処 .....                          | 309        |
| 外部設計による対処の注意点 .....                      | 310        |
| データマートによる対処 .....                        | 311        |
| データマートを採用するときの注意点 .....                  | 312        |
| データ鮮度 .....                              | 312        |
| データマートのサイズ .....                         | 312        |
| データマートの数 .....                           | 313        |
| パッチウィンドウ .....                           | 314        |
| インデックスオンリースキャンによる対処 .....                | 314        |
| Column インデックスオンリースキャンとカラム指向データベース .....  | 317        |
| インデックスオンリースキャンを採用するときの注意点 .....          | 318        |
| DBMSによっては使えないこともある .....                 | 319        |
| 1つのインデックスに含められる列数には限度がある .....           | 319        |
| 更新のオーバーヘッドを増やす .....                     | 319        |
| 定期的なインデックスのリビルドが必要 .....                 | 320        |
| SQL文に新たな列が追加されたら使えない .....               | 320        |
| 第10章のまとめ .....                           | 321        |
| 演習問題10 .....                             | 321        |
| <b>Appendix A</b>                        |            |
| PostgreSQLのインストールと起動 .....               | 323        |
| <b>Appendix B</b>                        |            |
| 演習問題の解答 .....                            | 333        |
| 索引 .....                                 | 347        |
| 著者プロフィール .....                           | 353        |