『これだけは知っておきたいデータベースの常識』（藤本壱　技術評論社）より



# 2.3 テーブルの正規化

リレーショナルデータベースでデータを管理する際には、「**正規化**」という作業を行って、データを効率良く管理できるように、複数のテーブルに分割します。この節では、基本的な正規化の手順を解説します。

## 正規化するテーブルの例

この節では、正規化の例として、「商品の販売伝票管理」のデータベースを考えます。

1 枚の伝票には、商品の販売先／日付や、販売した商品の商品名／単価／個数といった情報があります。このような伝票が多数あると想定して、そのデータをテーブルにすることを考えます。

この場合、「**1 枚 1 枚の伝票を、テーブルの 1 つの行に対応させる**」というのが、分かりやすい考え方になるでしょう。1 つの販売先に対して、一度に複数の商品を販売することもありますので、1 つの行に複数の商品が入った形の表ができます（**表 2.3**）。

例えば、伝票番号 1 番の行は、**図 2.25** のような伝票を、テーブルの行にしたものと考えることができます。

表 2.3 正規化するテーブルの例

| 伝票番号 | 日付 | 販売先番号 | 販売先名 | 販売先住所 | 商品番号 | 商品名 | 単価 | 個数 | 金額 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 4/1 | 1001 | 山田産業（株） | 東京都渋谷区… | 101 | B5 ノート | 120 | 10 | 1,200 |
| | | | | | 201 | A4 ファイル | 200 | 10 | 2,000 |
| 2 | 4/15 | 1002 | （株）田中商事 | 神奈川県横浜市… | 301 | ボールペン | 100 | 20 | 2,000 |
| 3 | 4/25 | 1001 | 山田産業（株） | 東京都渋谷区… | 301 | ボールペン | 100 | 15 | 1,500 |
| | | | | | 302 | 消しゴム | 80 | 10 | 800 |
| 4 | 5/5 | 1003 | （株）高橋建設 | 東京都新宿区… | 201 | A4 ファイル | 200 | 5 | 1,000 |
| | | | | | 101 | B5 ノート | 120 | 10 | 1,200 |
| | | | | | 302 | 消しゴム | 80 | 20 | 1,600 |

図 2.25 表 2.3 の伝票番号 1 番の行の元になった伝票

| 日付 | 20XX年4月1日 |
|---|---|
| 販売先番号 | 1001 |
| 販売先名 | 山田産業(株) |
| 販売先住所 | 東京都渋谷区… |

| 伝票番号 | 1 |
|---|---|

| 商品番号 | 商品名 | 単価 | 個数 | 金額 |
|---|---|---|---|---|
| 101 | B5 ノート | 120 | 10 | 1,200 |
| 201 | A4 ファイル | 200 | 10 | 2,000 |

## 1 つの行／列に 1 つの値を入れる――第 1 正規形

**表 2.3** のテーブルでは、商品名／単価／個数の列で、1 つの行に複数の値が入っているところがあります。例えば、伝票番号 1 番の行では、商品名／単価／個数に 2 つずつの値が入っています。

しかし、リレーショナルデータベースでは、**テーブルの 1 つの行／列には 1 つの値を入れることが必要**です。そこで、「伝票番号」～「販売先住所」の列と、「商品名」～「金額」の列とを、**別々のテーブルに分離**します。ここでは、分離後のそれぞれのテーブルを「伝票メイン」と「伝票サブ」と名付けることにします。

ただ、単純に分離するだけだと、伝票サブテーブルのそれぞれの行が、伝票メインのどの行と結びついているかが分かりません。そこで、伝票サブテーブルにも「伝

票番号」の列を作り、伝票サブテーブルの各行が、伝票メインテーブルのどの行に結びつくかを表すようにします。

また、**表 2.3** の「金額」列の値は、「単価」列と「個数」列の値を掛け算して求めたものです。このように、他の列の値から計算で求められる列のことを、「**導出列**」または「**導出項目**」と呼びます。

導出列は、データとして保存しておかなくても、その都度計算で求めることができ、保存しておく必要はありません。導出列に値を保存しておくことは、ディスクの容量を無駄に消費することにつながります。そこで、導出列をテーブルから取り除きます。

上記のような作業を行うと、**表 2.4** ／**表 2.5** のようなテーブルが 2 つできます。これを「**第 1 正規形**」と呼びます。

❖ 表 2.4　表 2.3 の第 1 正規形（伝票のメインのテーブル）

| 伝票番号 | 日付 | 販売先番号 | 販売先名 | 販売先住所 |
|---|---|---|---|---|
| 1 | 4/1 | 1001 | 山田産業（株） | 東京都渋谷区… |
| 2 | 4/15 | 1002 | （株）田中商事 | 神奈川県横浜市… |
| 3 | 4/25 | 1001 | 山田産業（株） | 東京都渋谷区… |
| 4 | 5/5 | 1003 | （株）高橋建設 | 東京都新宿区… |

❖ 表 2.5　表 2.3 の第 1 正規形（伝票のうち、商品に関するテーブル）

| 伝票番号 | 商品番号 | 商品名 | 単価 | 個数 |
|---|---|---|---|---|
| 1 | 101 | B5 ノート | 120 | 10 |
| 1 | 201 | A4 ファイル | 200 | 10 |
| 2 | 301 | ボールペン | 100 | 20 |
| 3 | 301 | ボールペン | 100 | 15 |
| 3 | 302 | 消しゴム | 80 | 10 |
| 4 | 201 | A4 ファイル | 200 | 5 |
| 4 | 101 | B5 ノート | 120 | 10 |
| 4 | 302 | 消しゴム | 80 | 20 |

### 導出列を取り除く順序

本書では、**第 1 正規形**を作る際に、**導出列**も取り除くようにしています。一方、導出列は**第 3 正規形**を作る時点で取り除くようにしている文献もあります。

## 主キーに部分関数従属する列を別テーブルに分ける――第2正規形

第1正規形では、テーブルの分割がまだ十分ではありません。さらに正規化を進めて、「**第2正規形**」という形にします。

### 主キーとは

第2正規形を考える際には、「**主キー**」と「**関数従属**」が重要な概念になります。まず、主キーについて解説します。

リレーショナルデータベースでは、「テーブル内の個々の行を他の行と区別できる」ということが重要になる場面が多いです。そこで、「テーブル内の個々の行を識別するための列」を決めることが一般的です。

このような列のことを、「**主キー**」(しゅキー)と呼びます(英語では「Primary Key」と呼びます)。主キーにする列は、1つの列であることが多いです。一方、複数の列の値を組み合わせて主キーとすることもあります。

**主キーにする列は、他の行と値が重複してはなりません。**また、nullにすることもできません。複数の列を組み合わせて主キーにする場合は、各列の値の組み合わせが、他の行と重複しないようにする必要があります。

例として、ここまでで作った第1正規形の2つのテーブルを考えてみます。まず、伝票メインを見てみます。このテーブルでは「伝票番号」の列があります。2枚の伝票に同じ番号を付けることはない(はず)なので、伝票番号の値で、個々の行を識別することができます。したがって、伝票番号は主キーになります(**図2.26**)。

一方、伝票サブのテーブルでは、「伝票番号」の列だけでは主キーにはなりません。例えば、テーブルの1行目と2行目は、どちらも「伝票番号」列の値が1で、同じになっています。

ここで、「伝票番号」列と「商品番号」列の組み合わせを考えてみます。これなら、テーブルの各行で、組み合わせが同じになる行は存在しません。したがって、伝票サブテーブルの主キーは、「『伝票番号』列と『商品番号』列の組み合わせ」と考えることができます(**図2.27**)。

❖ 図 2.26　伝票メインテーブルでは「伝票番号」列が主キー

| 伝票番号 | 日付 | 販売先番号 | 販売先 | 販売先住所 |
|---|---|---|---|---|
| 1 | 4/1 | 1001 | 山田産業(株) | 東京都渋谷区… |
| 2 | 4/15 | 1002 | (株)田中商事 | 神奈川県横浜市… |
| … | … | … | … | … |

主キー

❖ 図 2.27　伝票サブテーブルでは「伝票番号」列と「商品番号」列の組み合わせが主キー

| 伝票番号 | 商品番号 | 商品名 | 単価 | 個数 |
|---|---|---|---|---|
| 1 | 101 | B5 ノート | 120 | 10 |
| 1 | 201 | A4 ファイル | 200 | 10 |
| … | … | … | … | … |

主キー

## 主キーと候補キー

主キーになり得る列（または列の組み合わせ）は、1通りとは限らず、複数存在することがあります。このような、「主キーの候補になり得る列（または列の組み合わせ）」のことを、「**候補キー**」（英語では「Candidate Key」）と呼びます。

### 関数従属（完全関数従属と部分関数従属）

ここで、伝票サブのテーブルをもう一度見てみます。このテーブルの主キーは、前述したように「伝票番号」列と「商品番号」列の組み合わせです。

次に、主キーではない残りの3つの列を見てみます。主キーでない列のことを、「**非キー列**」と呼びます。

非キー列のうち、「個数」列の値は、伝票ごとの個々の商品の個数を指しています。したがって、「個数」列の値は、「伝票番号と商品番号の組み合わせ」（つまり主キーの値）によって、値が決まると言えます。

このように、「**ある列（または列の組み合わせ）によって別の列の値が決まる**」ということを、「**関数従属**」と呼びます。今の例で言えば、個数は伝票番号と商品番号に関数従属していると言えます。

また、「列Bが列Aに関数従属する」ということを、記号では「A → B」と表し

ます。また、列Xが複数の列A、B……に関数従属するする場合は、「{A, B, ……}→X」と表します。例えば、個数と伝票番号／商品番号の関係を記号で表すと、「{伝票番号, 商品番号}→個数」となります。

一方、「商品名」と「単価」の列は、伝票番号には関係なく、商品番号だけで決まります。つまり、商品名と単価は、商品番号に関数従属していると言えます（**図2.28**）。

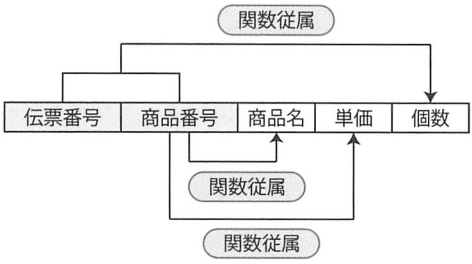
ある列が複数の列に関数従属し、かつそのうちの一部の列だけには関数従属しない場合、そのような関数従属のことを「**完全関数従属**」と呼びます。

例えば、**図2.28**の個数は、伝票番号と商品番号の組み合わせに関数従属します。しかし、伝票番号のみや、商品番号のみには関数従属しません。したがって、個数は伝票番号と商品番号の組み合わせに完全関数従属しています。

一方、ある列が複数の列に関数従属しているように見えて、実際はそのうちの一部の列にしか関数従属していない場合、そのような関数従属を「**部分関数従属**」と呼びます。

例えば、**図2.28**の商品名と単価は、伝票番号と商品番号の組み合わせに関数従属しているように見えなくもありません。しかし、図にもあるように、実際には商品番号にのみ関数従属しています。したがって、商品名と単価は、伝票番号と商品番号の組み合わせに部分関数従属していると言えます。

❖ 図2.28 伝票サブテーブルの各列の関数従属




### 主キーに部分関数従属する列を別のテーブルに分ける――第2正規形

ここまでに何度か述べたように、伝票サブテーブルの主キーは、伝票番号と商品番号の組み合わせです。また、前の項で述べたことから、個数は主キー（伝票番号と商品番号の組み合わせ）に完全関数従属していて、商品名と単価は主キーに部分

関数従属している、と言えます。

テーブルの中に、主キーに部分関数従属している列があると、「データをうまく追加することができない」などの困ったことが起こります。このような、**データの追加／変更／削除を行う際に起こる問題を総称して、「更新時異常」と呼びます。**

例えば、今取り上げている伝票管理のデータベースで、新たな商品として「シャープペンシル」を扱うとします。ところが、そのシャープペンシルの情報（商品名と単価）だけを入力しようと思っても、入力できるテーブルがありません。

伝票サブテーブルには商品名と単価の列がありますが、それだけでなく伝票番号と個数も入力する必要があります。そのため、新たな商品が増えても、その情報をすぐに入力することができません。

この問題は、**「主キーに完全関数従属している列と、部分関数従属している列とを、別々のテーブルに分ける」**という手法を取ることで解決することができます。

今取り上げている伝票サブテーブルの例だと、主キーに完全関数従属している列は、個数の列です。一方、主キーに部分関数従属している列は、商品名と単価の列です。そこで、伝票サブテーブルを２つのテーブルに分け、１つには主キー（伝票番号と商品番号）と個数、もう１つには商品番号／商品名／単価を入れるようにします。また、商品のテーブルからは、重複する行を取り除きます（**図 2.29**）。

このようにすることで、**図 2.29** の商品テーブルは、商品の情報だけを含むテーブルになります。したがって、取り扱う商品が増えて、その商品をまだ販売していないとしても、その商品の情報だけをテーブルに追加することができるようになります。

ここまでで述べたように、「非キー列の中で、主キーに部分関数従属する列がある場合は、その列を別のテーブルに分ける」という作業を行います。その結果のことを、**第２正規形**と呼びます。

♥ 図 2.29 主キーに完全関数従属している列と、部分関数従属している列とを、別々のテーブルに分ける

伝票サブ

| 伝票番号 | 商品番号 | 商品名 | 単価 | 個数 |
|---|---|---|---|---|
| 1 | 101 | B5 ノート | 120 | 10 |
| 1 | 201 | A4 ファイル | 200 | 10 |
| 2 | 301 | ボールペン | 100 | 20 |
| 3 | 301 | ボールペン | 100 | 15 |
| 3 | 302 | 消しゴム | 80 | 15 |
| 4 | 201 | A4 ファイル | 200 | 5 |
| 4 | 101 | B5 ノート | 120 | 10 |
| 4 | 302 | 消しゴム | 80 | 20 |

主キーに完全関数従属する列と
部分関数従属する列とを
別々のテーブルに分ける

| 伝票番号 | 商品番号 | 個数 |
|---|---|---|
| 1 | 101 | 10 |
| 1 | 201 | 10 |
| 2 | 301 | 20 |
| 3 | 301 | 15 |
| 3 | 302 | 15 |
| 4 | 201 | 5 |
| 4 | 101 | 10 |
| 4 | 302 | 20 |

伝票サブ

| 商品番号 | 商品名 | 単価 |
|---|---|---|
| 101 | B5 ノート | 120 |
| 201 | A4 ファイル | 200 |
| 301 | ボールペン | 100 |
| 302 | 消しゴム | 80 |

商品

### 外部キー

テーブルを 2 つに分けると、**「片方のテーブルの主キー（または候補キー）の列を、もう片方のテーブルが参照する」**という形が出てきます。

**図 2.29** だと、商品テーブルの「商品番号」の列は、個々の商品を識別するのに使うことができますので、主キーになり得ます。一方、伝票サブテーブルの「商品番号」の列は、商品テーブルの「商品番号」列を参照するのに使う形になります。

このように、他のテーブルの主キー（または候補キー）を参照する列は、**「外部キー」**（英語では「Foreign Key」）に設定することが多いです。外部キーの列には、参照先の列にある値を入れるか、または値を入れない（null）かのどちらかにする必要があります。

**図 2.29** の例だと、伝票サブテーブルの「商品番号」列は、101 ／ 201 ／ 301 ／ 302 のいずれかの値を入れるか、もしくは値を入れないようにする必要があります。

## 推移的に関数従属する列を別のテーブルに分ける――第 3 正規形

第 2 正規形では、まだテーブルの分割が不十分な場合もあります。その際にはさらに分割を進めて、「**第 3 正規形**」にします。

### 「推移的関数従属」について

ここまでで「**関数従属**」という用語が出てきましたが、もう 1 つの関数従属として「**推移的関数従属**」があります。第 3 正規形を考える際には、推移的関数従属がキーワードになります。

55 ページで、最初のテーブルを伝票メインと伝票サブのテーブルに分け、**第 1 正規形**にしました。そして、伝票サブテーブルをさらに分割して、**第 2 正規形**にしました。ところが、伝票メインテーブルは、55 ページ以来手を付けていません。そこで、伝票メインテーブルで、列間の関数従属の関係を考えてみます。

伝票メインテーブルには、伝票番号／日付／販売先番号／販売先／販売先住所の 5 つの列があります。そして、主キーは伝票番号でした。そこで、主キーと、残りの 4 つの列の関係を考えます。

4 つの列のうち、日付と販売先番号は、伝票番号によって決まる値です。したがって、伝票番号と日付、および伝票番号と販売先番号は、関数従属の関係になっています。

一方、販売先と販売先住所は、販売先番号によって決まると考えることができます。つまり、販売先番号と販売先名、および販売先番号と販売先住所は、関数従属の関係です。一方、これら 2 つの列は、主キーである伝票番号とは、直接的には関数従属の関係にはなっていません。販売先番号を介して、間接的に関数従属する関係になっています（**図 2.30**）。

このように、「列 B が列 A に関数従属し、また列 C が列 B に関数従属する」という場合（記号で表すと A → B → C）、「列 C は列 A に**推移的に関数従属**する」と呼びます。

図 2.30 伝票メインテーブルでの列間の従属関係

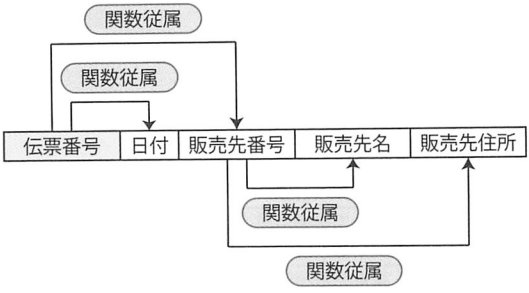
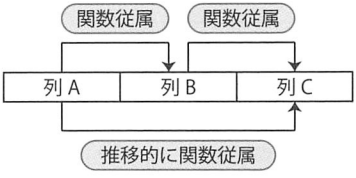



図 2.31 列 C は列 A に推移的に関数従属する




### 主キー以外に関数従属する列を別のテーブルに分ける

**図 2.31** のように、テーブル内に主キーと推移的に関数従属する列がある場合、テーブルにデータを追加するときに困ることが出てきます。

例えば、商品の販売先が新たに増えるとして、そのデータを入力することを考えてみます。今の状況だと、商品の販売先の情報は、伝票メインテーブルに入力するようになっています。しかし、伝票メインテーブルには、販売先だけでなく、商品を販売した日付も入力するようになっていて、販売先だけを入力することができません。

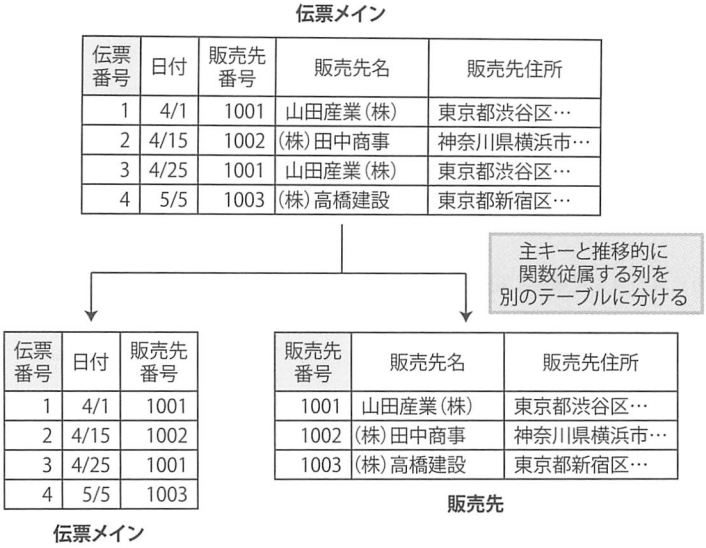
この問題は、主キーと推移的に関数従属する列を別のテーブルに分けることで、解決することができます。今取り上げている例だと、伝票メインテーブルのうち、販売先名と販売先住所の列を、販売先だけのテーブルに分けます。また、伝票メインテーブルと販売先テーブルを結合できるように、販売先テーブルにも販売先番号の列を入れます。

さらに、単にテーブルを分けるだけだと、販売先テーブルに、同じ販売先が何度も出てきます。そこで、重複する行を取り除きます(**図 2.32**)。

このように、**第 2 正規形を元に、主キーと推移的に関数従属する列を別のテーブ**

ルに分けると、その結果が**第3正規形**になります。

❤ 図 2.32　伝票メインテーブルから、販売先に関する行を抜き出して、販売先テーブルに分ける




## その他の正規形

**第3正規形**をより厳密にした正規形として、「**ボイス・コッド正規形**」があります。また、「**第4正規形**」「**第5正規形**」もあります。

ただ、一般的なデータベースでは、第3正規形まで正規化すれば十分であることが多いです。本書では、第3正規形から後の正規形は、解説を省略します。

## 意図的に正規形を崩すこともある

実際にデータベースを運用する際には、**意図的に正規形を崩す**こともあります。

例えば、「単価と個数をかけて金額を求める」という場合、テーブルに単価と個数の列を作っておけば、金額はその都度計算で求めることができます。ただ、金額を頻繁に計算する場合、あらかじめ計算済みの値をテーブルに入れておけば、データ処理のパフォーマンスが上がります。

このように、「あらかじめ計算済みの値をテーブルに入れておく」ということは、

実際のデータベースの運用ではままあることです。ただし、「計算済みの値の列」は導出列（56 ページ参照）にあたり、正規化に反します。

正規化はなるべく行うべきですが、常に完全に正規化しなければならないのではありません。データベースの運用に応じて正規化を崩すのも、テクニックの 1 つです。