論理設計　論理データモデル(多対多の分解、キーの整理)

# 論理設計ですること

概念設計で作成したER図はあくまで概念の世界における理想的なエンティティ構造を表しているにすぎない為、このままの姿でデータベースに格納できるとは限りません。そこで、利用する予定の**データベースが扱いやすい構造にエンティティを変形する作業が論理設計**です。

具体的にどのような変形を行えばよいのか、論理設計の流れを確認しておきましょう。

概念設計

(概念データモデル)

商品

従業員

店舗

論理設計

(論理データモデル)

商品

従業員

職種

正規化

キーの

整理

多対多の

分解

# 多対多の分解

リレーショナルデータベースでは「多対多」の関係をうまく扱うことが出来ません。そこで、２つのエンティティの対応を格納した**中間テーブル**を追加することによって、「多対多」を**２つの「１対多」の関係に変換**します。

多対多

商品

中間

１対多

原材料

多対１

多対多の関係だとテーブル同士の

リレーションが出来ない

中間テーブルを持つことにより

各テーブルの整合性が保たれる

商品

原材料

ハンズオン　商品と原材料の関係を整理する

１．前回、ハンズオンで使用した「database2.drawio」ファイルを開く

https://app.diagrams.net/

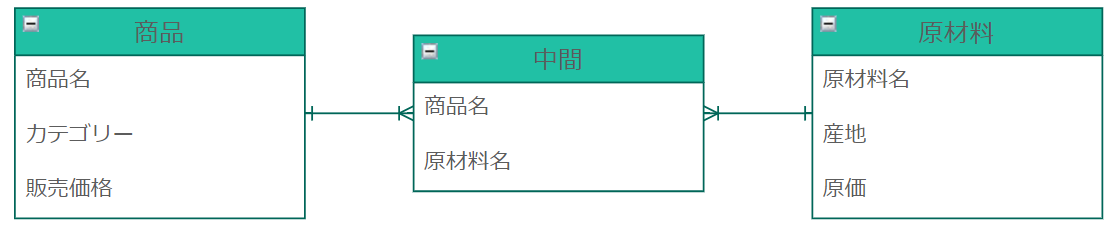
２．商品エンティティと原材料エンティティは「多対多」の関係なので

中間エンティティを作成して、属性に「商品名」と「原材料名」を持たせる

３．商品エンティティと中間エンティティは「1対多」のカーディナリティを

中間エンティティと原材料エンティティは「多対１」のカーディナリティの

リレーションに変形を行う



# キーの整理

エンティティとキーについて整理と確認をします。特に重要なのが**主キー**です。**主キーを持たないエンティティには、管理をしやすくするために人工的な主キーを追加**します。

主キーが備えるべき３つの特性を押さえておきましょう。

|  |  |
| --- | --- |
| 特性 | 説明 |
| 非NULL性 | 必ず何らかの値を持っている |
| 一意性 | 他の値と重複しない |
| 不変性 | 一度決定されたら値が変化することがない  (主キーは一貫して同じ１行を指し示す) |

商品エンティティ、原材料エンティティともに主キーと呼べるものはありません。

商品名や原材料名で判別できなくはないですが、商品名が決まっていなかったり、変更されたりする可能性がありますので主キーには適していません。

ハンズオン　商品と原材料に主キーを設定する

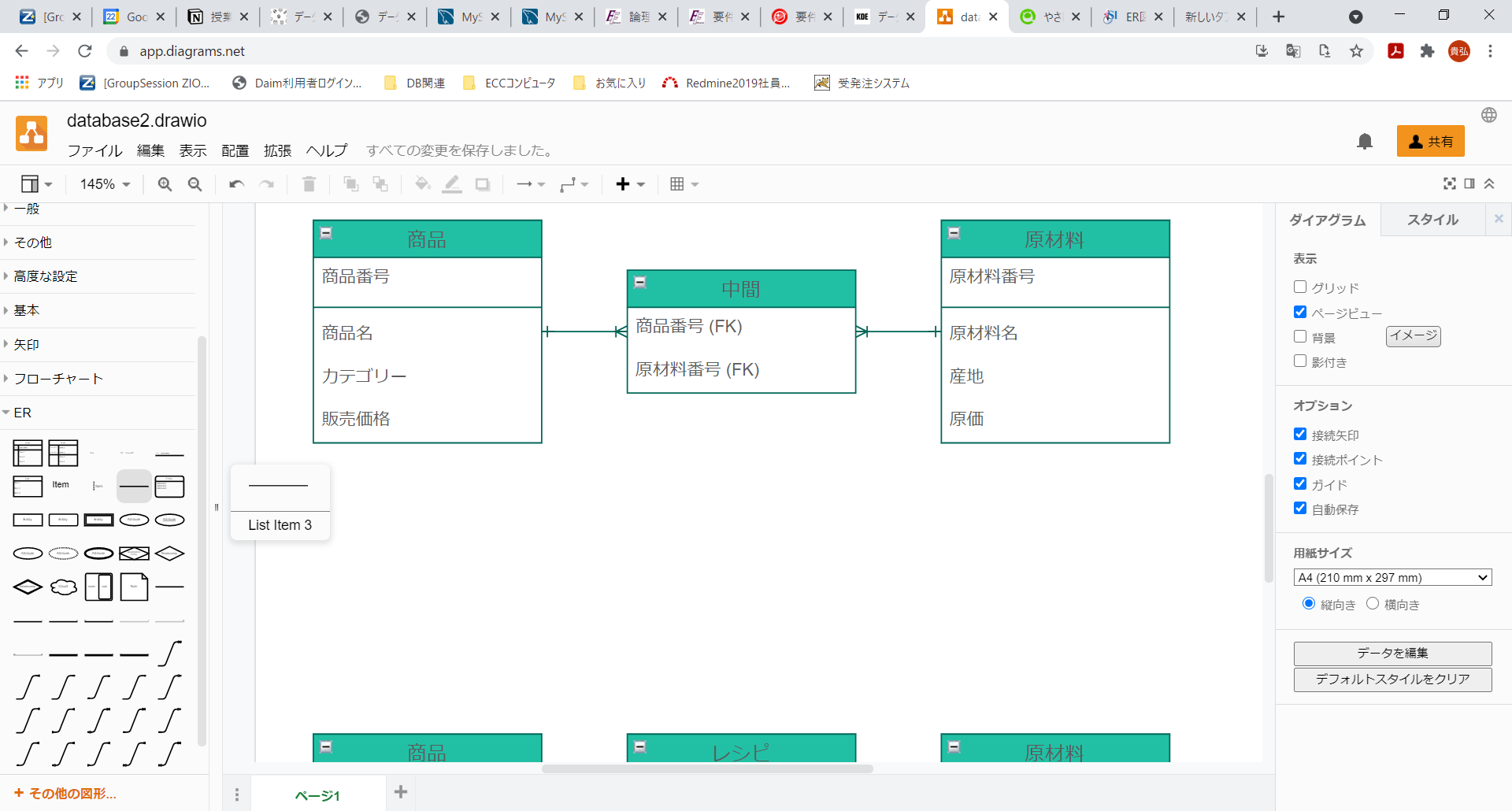
１．商品エンティティと原材料エンティティにそれぞれ「商品番号」と「原材料番号」の

人工主キーを属性に追加します

２．中間エンティティの属性を「商品番号」と「原材料番号」に変更します

３．中間エンティティの「商品番号」と「原材料番号」の内容は商品、原材料エンティティに存在するものなので外部キーを表す(FK)を追記する。

４．主キーを判別させるために、List Item3を主キーの下にドラッグ＆ドロップする

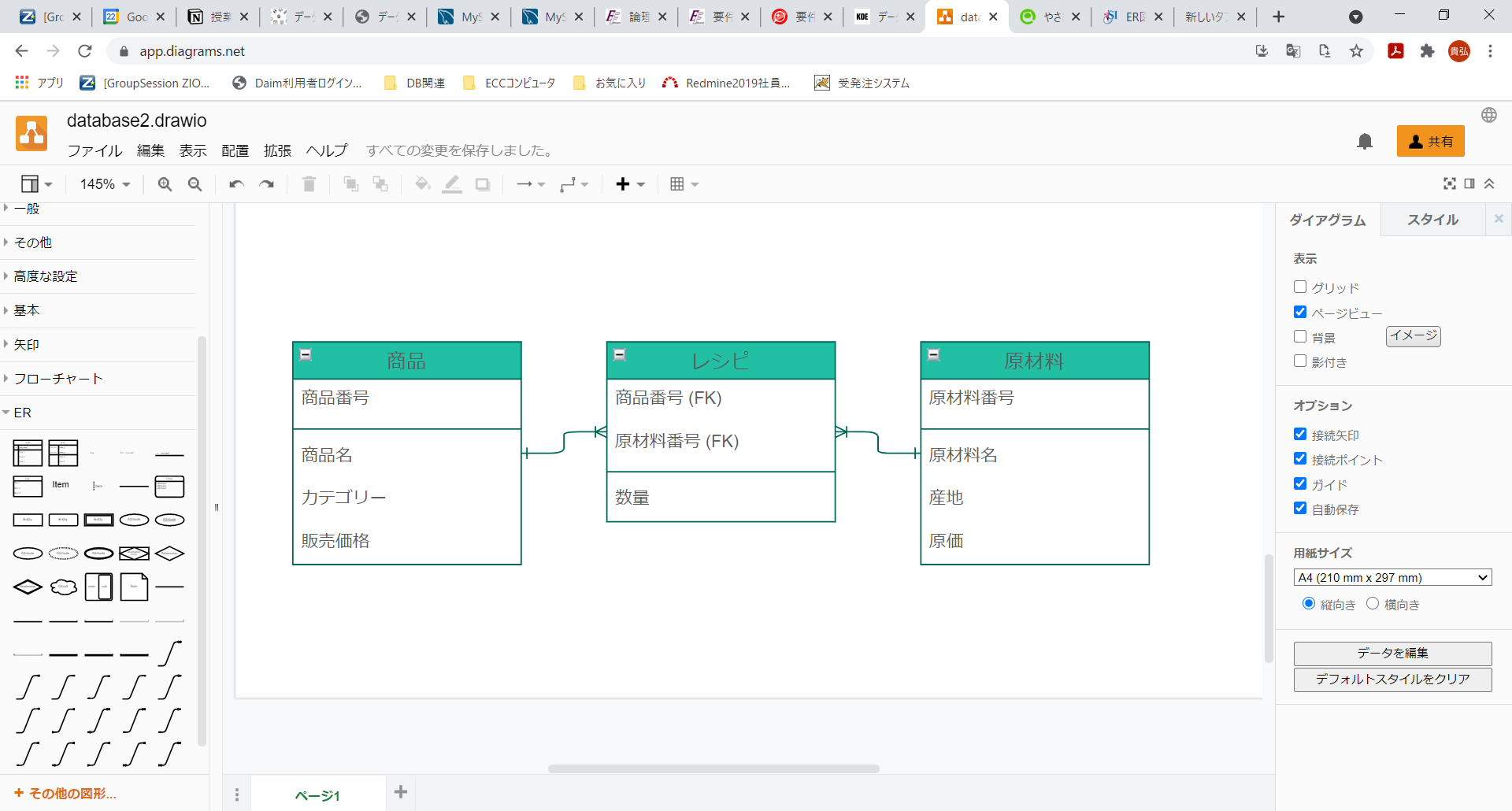


中間エンティティを持つことにより、**一つの商品は複数の原材料を使用して、一つの原材料は複数の商品から利用されるER図を描くことが出来ます**。つまり中間エンティティには「商品がどの原材料を使うのか」が管理されています。

ただし、「原材料をどれぐらい使うのか」が管理出来てないので「数量」の属性を追加します。また、数量が管理出来れば商品を作るレシピになるので中間エンティティを

「レシピ」エンティティに変更します。

５．中間エンティティに「数量」属性を追加して、エンティティ名を「レシピ」に変更する



正規化

論理設計におけるもっとも中心的な作業は、**正規化**(normalization)作業です。**正規化とは、矛盾したデータを格納できないよう、テーブルを複数に分割していく作業**です。

テーブルが正規化出来ていないと、データを更新するときに複数の関連個所を正確に更新しなければならず、間違えるとデータの整合性が失われます。人間は忘れたり間違えたりする生き物なので、常に複数の箇所に対して正確に更新できるとは限りません。

整合性が崩れにくい優れたテーブル設計の原則は「**１つの事実は１箇所に**」です。正規化という手法を用いて正しくテーブルを分割することにより、ヒューマンエラーを起こしにくい優れたテーブル構造を構築することが出来ます。

# 正規化の段階

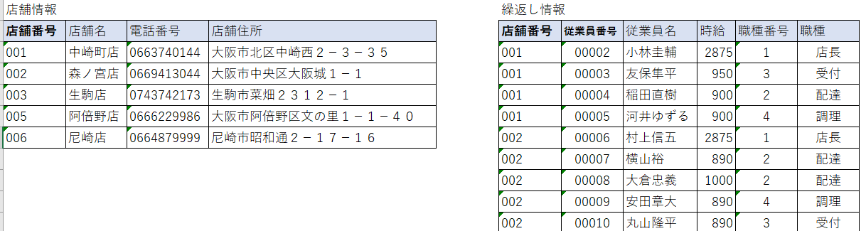
正規化によってテーブルが適切に分割された状態を**正規形**（normalized form）と言います。

システム開発を行う場合、どの程度正規化されているかによって第１正規形から第３正規形までに分けられます。

非正規形



第１正規形



第２正規形



第３正規形



更新時にデータの整合性が崩れにくい

# 非正規形

非正規形とは、その名の通り正規化が全く行われていない状態を表します。

テーブルの列が結合されていたり、１つのフィールドに複数の値を格納していたり、

様々な形式がありますが、どれもリレーショナルデータベースの基本構造が守られていない状態です。

○リレーショナルデータベースの基本構造

フィールド

列(データの要素)

行(１件のデータ)

|  |  |  |  |
| --- | --- | --- | --- |
| 社員番号 | 氏名 | 年齢 | 出身地 |
| 0101 | 菅原　拓磨 | 31 | 福岡県 |
| 0104 | 大江　岳人 | 30 | 京都府 |
| 0108 | 立花　いずみ | 29 | 千葉県 |

横１行が１つのデータに対応して、縦１列はデータの要素に対応する。

表は「テーブル」、列は「カラム」、行は「レコード」とも呼ばれ、行と列の交差する１つのデータを表す領域をフィールドと呼びます。

しっかりと概念設計を行っていれば、非正規形になっている可能性はほとんどありません。なぜなら、**非正規形の構造は通常のER図では表現できない**からです。

しかし、**ボトムアップアプローチで帳票や画面からデータベース設計を行う場合は、非正規形から始める必要があります**。学習環境では、商品と店舗・従業員について非正規形から正規化を進めていきます。

ハンズオン　非正規形の情報を確認する

１．「商品情報正規化.xlsx」を開いて非正規形段階の商品情報を確認する。

２．「店舗・従業員情報正規化.xlsx」を開いて非正規形段階の店舗・従業員情報を確認する。

# 関数従属性

正規化を理解するためには、関数従属性を理解する必要があります。

関数従属性とは、**関係（リレーション）の中でXの値が決まるとYの値が決まる**ことです。

関数従属性には、いくつか種類があります。

|  |  |
| --- | --- |
| 種類 | 説明 |
| 完全関数従属性 | X(候補キー)の全ての値が決まると、Yの値が決まる状態の時、  YはXに完全関数従属している。 |
| 部分関数従属性 | X(候補キー)の一部の値が決まると、Yの値が決まる状態の時、  YはXに部分関数従属している。 |
| 推移関数従属性 | Xの値が決まると、Yの値が決まり、Yが決まることでZの値が決まる状態の時、ZはXに推移関数従属している。 |

※データべース設計に関連しない関数従属性については省略

・完全関数従属と部分関数従属の例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **商品番号** | **原材料番号** | 商品名 | 原材料名  数量は商品番号と原材料番号が決まると  内容が確定するので完全関数従属している。 | 数量 |
| 0001 | 00014 | マルゲリータ | トマト | ３ |

原材料名は原材料番号が決まると

内容が確定するので部分関数従属している。

・推移関数従属の例

|  |  |  |  |
| --- | --- | --- | --- |
| **従業員番号** | 従業員名 | 勤務店舗 | 店舗名 |
| 00002 | 小林圭輔 | 001 | 中崎町店 |

従業員番号が決まると、勤務店舗が決まる。勤務店舗が決まると店舗名の内容が確定する。

この時、店舗名は従業員番号に推移関数従属している。

# 第1正規形

第１正規形とは、**「繰り返し項目」やセル結合などの「複雑な構造」の排除された状態**です。

非正規形を第１正規形に変形するためには、次の３ステップを実行します。

【ステップ１】繰り返し列の部分を別の表に切り出す

【ステップ２】切り出したテーブルの仮の主キーを決める

【ステップ３】主キー列をコピーして複合主キーを構成する

ハンズオン　商品情報を第１正規形にする

１．「【非正規形】商品レシピ一覧」シートをコピーしてバックアップしておく。

シート名を「【第１正規形】商品レシピ一覧」に変更する。

２．【ステップ１】繰り返し列の部分を別の表に切り出す。

一つの商品に対して繰り返して入力されている「原材料番号」「原材料」「産地」「原価」

「個数」「原価小計」を列ごと切り取り右側のスペースにペーストする。

３．【ステップ２】切り出したテーブルの仮の主キーを決める

切り出したテーブルの「原材料番号」列名を太字に設定して仮主キーを設定する。

４．【ステップ３】主キー列をコピーして複合主キーを構成する

元のテーブルの「商品番号」を列ごとコピーして切り分けたテーブルにペーストする

さらに「商品番号」列名を太字に設定して複合主キーを設定する。



# 第２正規形

第２正規形とは、**部分関数従属性項目の分離**を行った状態です。

第１正規形を第２正規形に変形するためには、次の２ステップを実行します。

【ステップ１】複合主キーの一部に関数従属する列を別表に切り出す

【ステップ２】部分関数従属先のキー列をコピーする

ハンズオン　商品情報を第２正規形にする

１．「【第１正規形】商品レシピ一覧」シートをコピーしてバックアップしておく。

シート名を「【第２正規形】商品レシピ一覧」に変更する。

２．【ステップ１】複合主キーの一部に関数従属する列を別表に切り出す

繰返し情報にある情報のうち「原材料番号」のみに関数従属する

「原材料」「産地」「原価」を列ごと切り取り右側のスペースにペーストする。

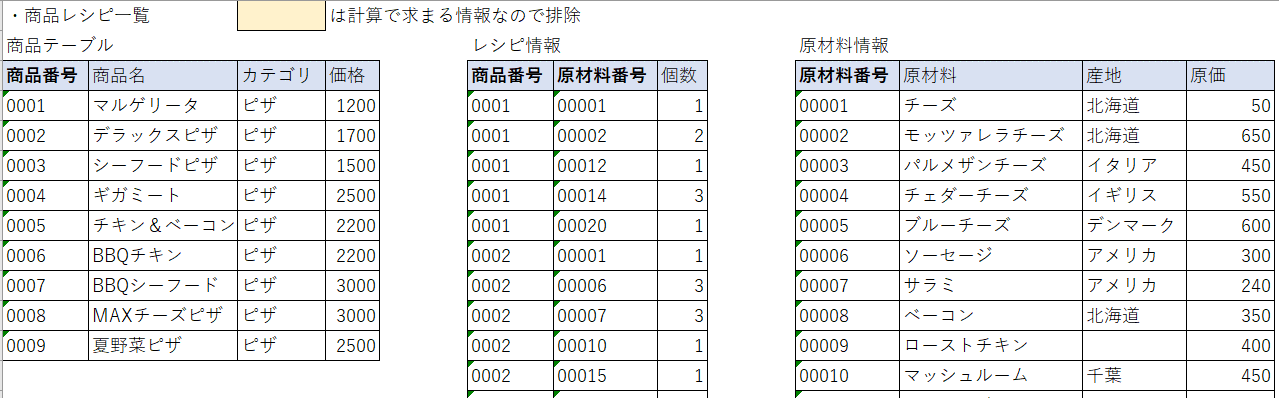
３．【ステップ２】部分関数従属先のキー列をコピーする

元のテーブルの「原材料番号」を列ごとコピーして切り分けたテーブルにペーストする

４．計算で算出できる列「生産原価」「原価小計」を削除する

５．テーブル名をレシピ情報、原材料情報に変更する

６．原材料情報の重複データを削除する



**※商品・原材料・レシピ情報は、この状態で正規化が完成です。理由は後述します。**

# 第3正規形

第３正規形とは、**推移関数従属項目の分離**を行った状態です。

リレーショナルデータベースでは、**第３正規形まで出来ていればOK**です。

第２正規形を第３正規形に変形するためには、次の２ステップを実行します。

【ステップ１】関節的に主キーに関数従属する列を切り出す

【ステップ２】直接関数従属先だった列をコピーする

ハンズオン　店舗・従業員情報を第３正規形にする

先程のハンズオンで行っていた商品情報には推移関数従属はありません。

なので、第2正規形まで行った状態でリレーショナルデータベースに管理できる形になりました。第３正規形のハンズオンでは、店舗・従業員情報を第３正規化してきます。

１．店舗・従業員情報正規化.xlsxを開いて非正規化から第２正規化までの流れを確認する。

２．「【第２正規形】店舗・従業員一覧表」シートをコピーしてバックアップしておく。

シート名を「【第３正規形】店舗・従業員一覧表」に変更する。

３．第２正規化で部分関数従属の分離を行うと複合キーに関数従属する列が無くなりました。「店舗番号」を列ごとコピーして従業員情報にペーストして「勤務店舗」に変更する。

４．繰り返し情報列を削除する。

５．【ステップ１】関節的に主キーに関数従属する列を切り出す

従業員情報にある「職種」は間接的に「従業員番号」と関数従属するので列ごと

切り取り右側のスペースにペーストする。

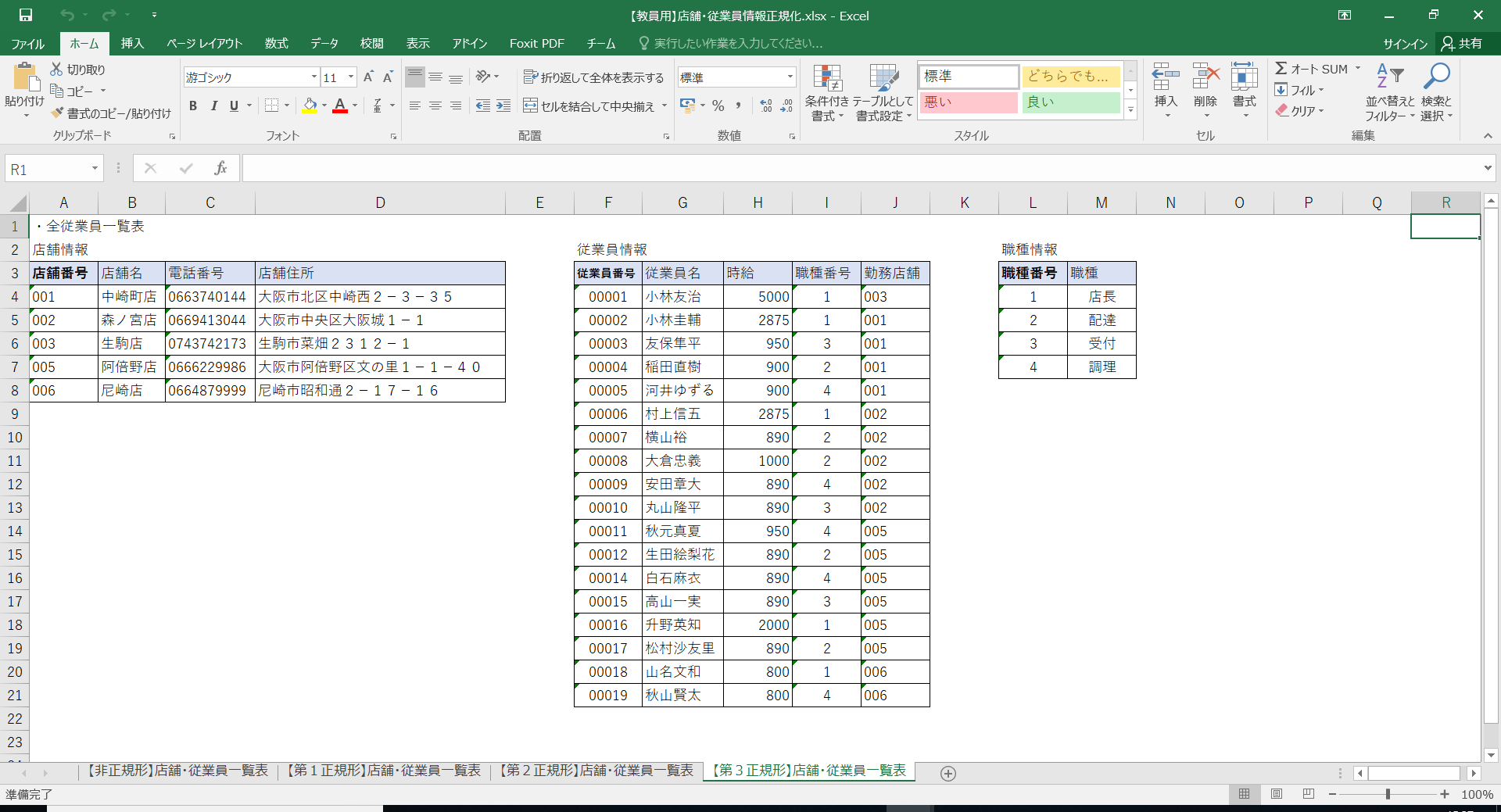
６．職種情報の重複データを削除して人工主キーの「職種番号」を追加する。

７．【ステップ２】直接の関数従属先だった列をコピーする

「職種」は従業員情報の「職種番号」に対して完全関数従属するので「職種番号」を

従業員情報テーブルに追加する。

※L列の再表示してください。



このように正規化を行うことで、リレーショナルデータベースで管理しやすいテータ構造（データモデル）を設計することが出来ます。概念設計をしっかり出来ていると正規化の作業が簡単になる傾向があります。

ハンズオンではボトムアップで正規化を行ったのでここからER図を作成する必要がありますが、ER図の書き起こし方法は同じなので作業は割愛します。

データベース設計で一番大事なことは「要望にきちんと応えたもの作りを作り上げる」ことです。トップダウン、ボトムアップという言葉だけに気を取られることなく、両方の良いととろをうまく利用してデータモデルを完成させましょう。