**データベース設計：**

システム導入後正常に稼働するため、導入するまでの手順を整理する必要があります。

システム開発の進め方：

要件定義―＞設計―＞開発―＞導入・運用

要件定義：

* 課題や要望を受けて必要な機能を考える。
* お客さんがシステムをどのような目的で使われるのかを理解して、問題を解決するようなシステムを想像する。

設計：

* 要件定義を基に、保存する必要のあるデータを考える
* 保存するデータの実体（Entity）とその属性を抽出する
* データ同士の関係を考える。リレーションシップは三つの種類があります
  + 1対1、1対多、多対多
  + 多対多を別のテーブルに表す（本・読者・評価）
* システム構成・データベースで必要なテーブル・カラムとデータ型等を決める。
* 設計を図で表す

開発：

* 設計を基に、システムを構築する

導入・運用：

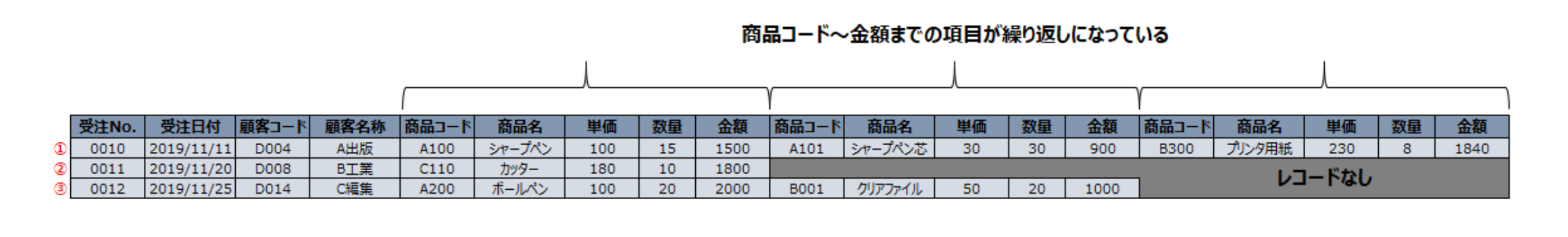
* 開発したシステムを導入する前にテストを行う
* 業務に導入して、運用開始する

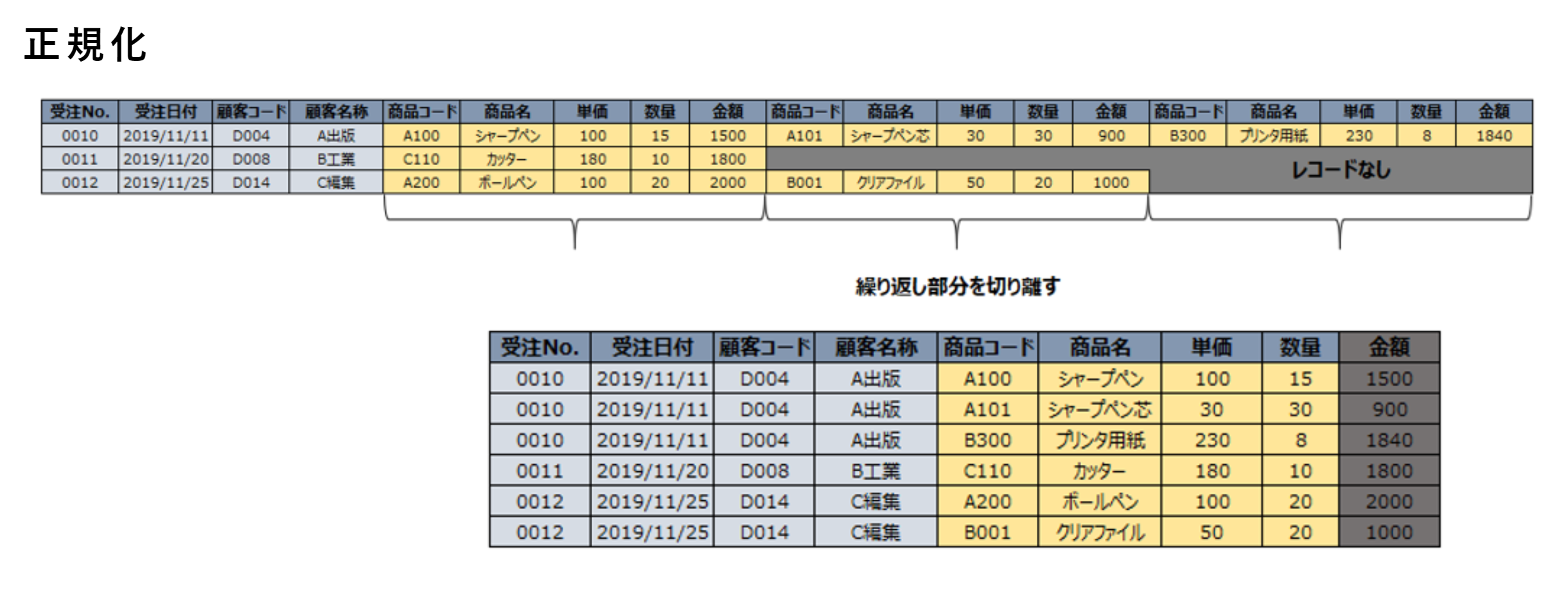
**正規化：**

* データベース内のデータを整理する手順
* それによって、重複のデータを減らすことが出来る
* データベースを管理・維持しやすくなる
* 汎用性・他のシステムとの連携・データ移行することが容易になる

**第一正規形：**

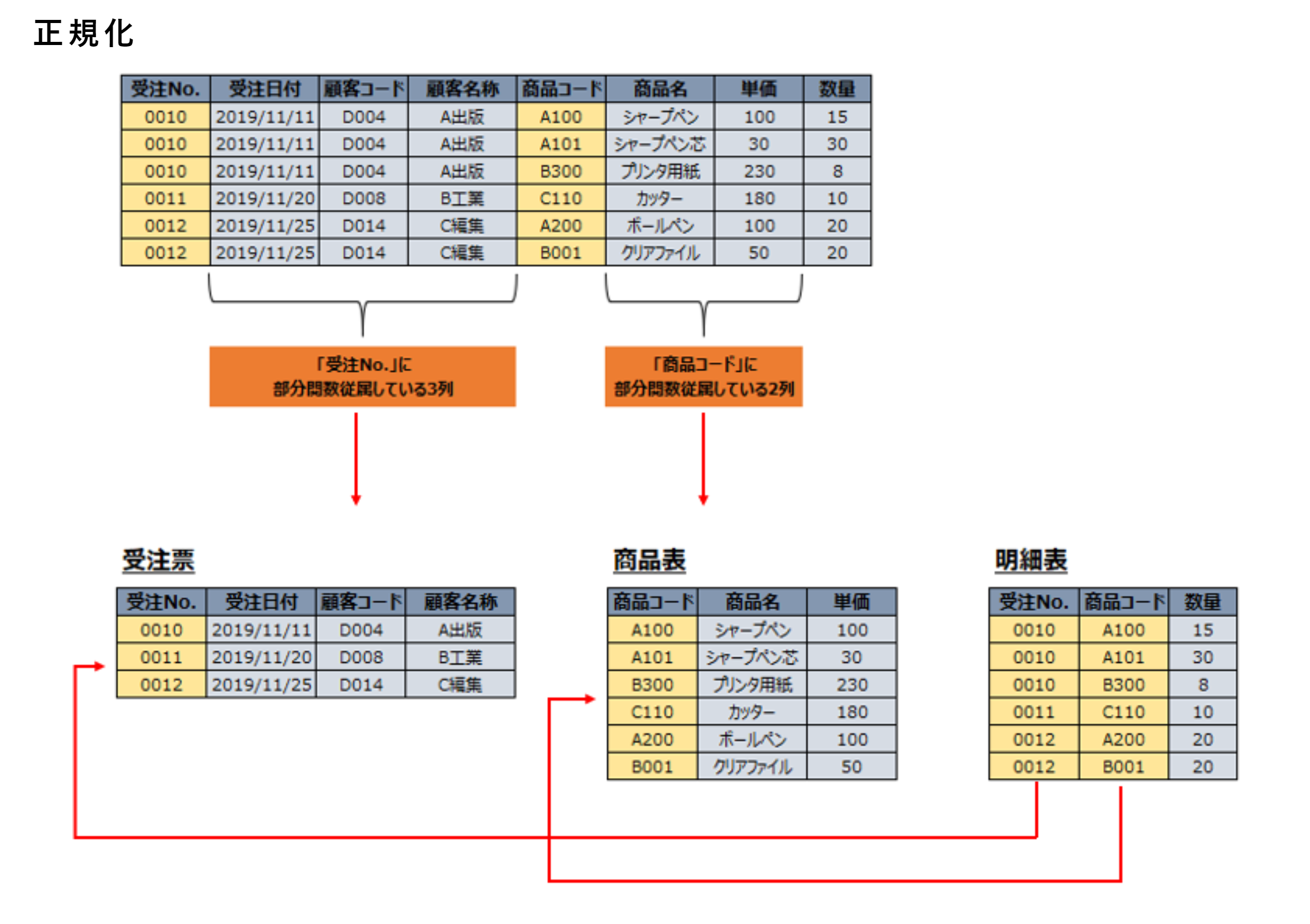
1. 1つのデータで繰り返し出て来る項目が排除して、別の行として独立する





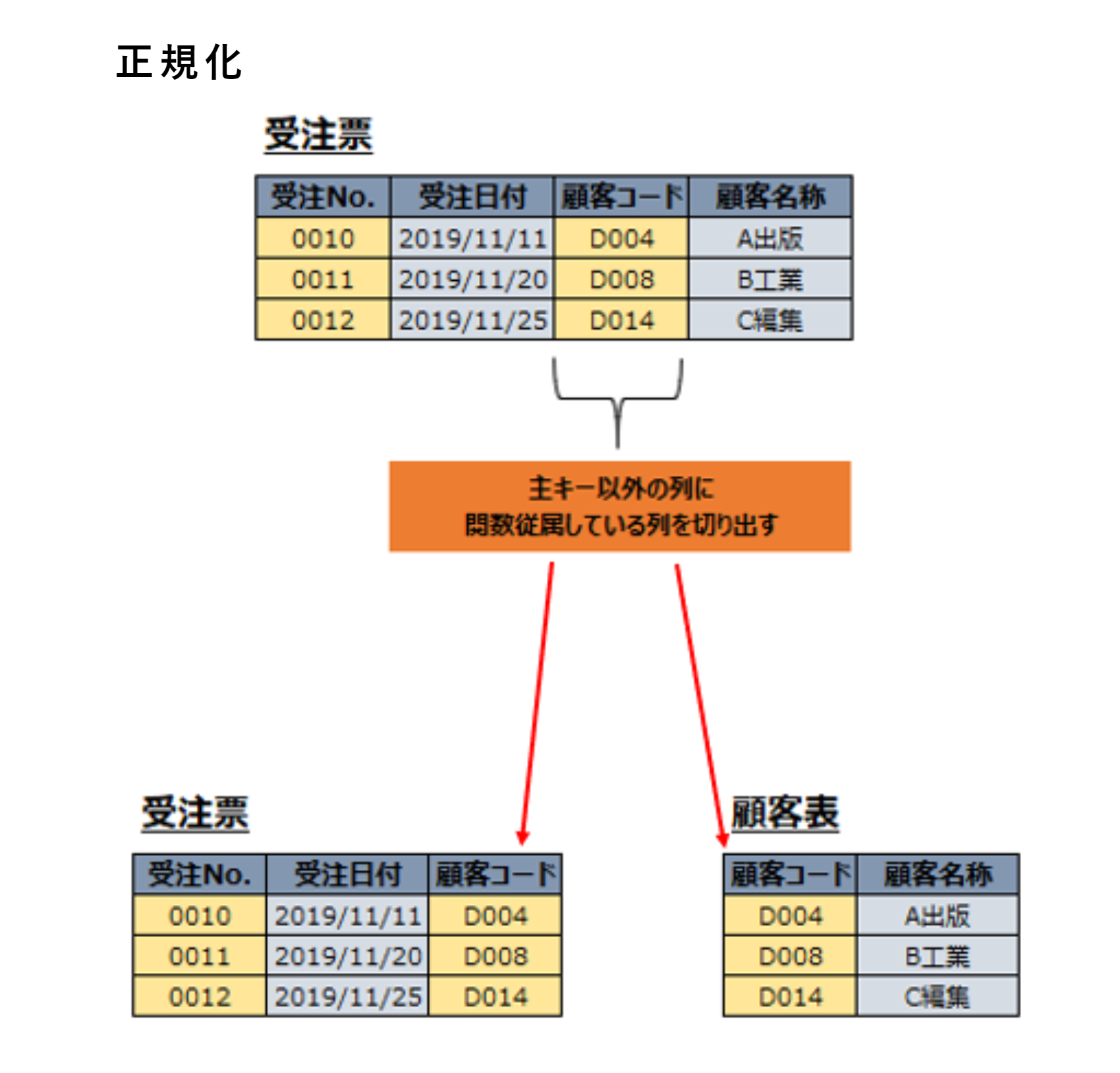
**第二正規形：**

1. 別の種類の項目を分割する、従属関係を取り除く
2. レコードを特定出来る主キーカラムと従属関係のカラムを別のテーブルに分ける
3. 登録と編集が行いやすくなる



**第三正規形：**

1. 主キーと従属関係あるカラム以外の従属関係を更に別のテーブルに分ける



**名前の仕方：**

* 英語を使う
* 小文字にする
* 省略を避ける
* 複数形にする

**データベースを置く場所：**

1. On-premise
   1. 弱点：機器購入、維持、障害対応、セキュリティ対応する必要がある
   2. 利点：システムを自由に変更出来る
2. Cloud
   1. 運用の手間を減らせる
   2. 利用した分だけの費用を払う
   3. 初期運用のコストが減る
   4. 自由度が比較的に低い

データベース運用にかかるコストの種類：

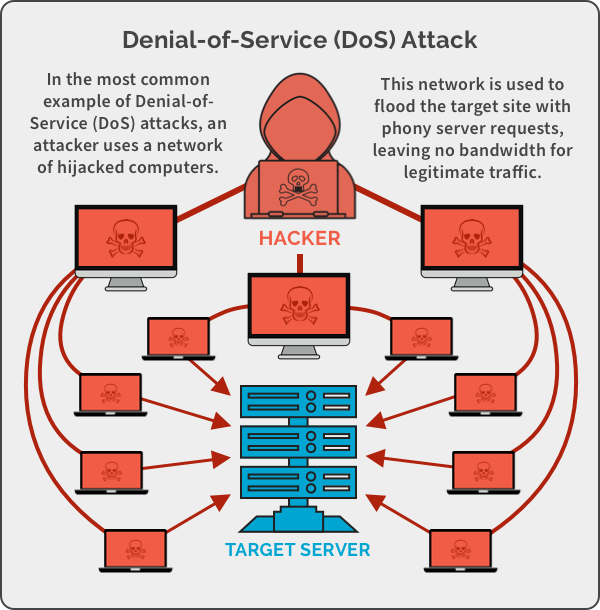
1. 初期費用
2. 経営費用（毎月かかる費用）

* 初期費用と経営費用のバランスが重要
* 初期コストが低い理由で設備を購入しても、経営費用が高くなる可能性もあります
* 運用する時、ユーザーと権限設定が必要
* データベースを監視するためにローグと稼働率の記録を残す
* 定期的にバックアップするのが重要
* 秘密データを暗号化して保存する

**データベースを守るための知識：**

**システムに起きる問題：**

* 物理的脅威：
  + - * 災害・盗難・老朽等で機器に故障起きるリスク
      * 自然障害の対策：
        + UPS, 耐震対策
      * 盗難の対策：
        + 部屋の施錠、入退管理、防犯カメラ等
      * 機器老朽の対策：
        + データのバックアップ、予備装置等
* 技術的脅威：
  + DoS攻撃・盗聴・SQL injection等
  + 不正アクセス：
    - 権限がない人がシステムに侵入すること
  + Virus
  + DoS(Denial of Service) 攻撃：
    - 大量なデータを送信することによって、サーバーに負担をかける攻撃
  + SQL Injection：
    - 外部から不正なSQL文を入力すること
    - 対策として、ユーザーが入力するデータそのまま保存せず、データを変換して保存する
    - WAF(Web Application Firewall)を導入する



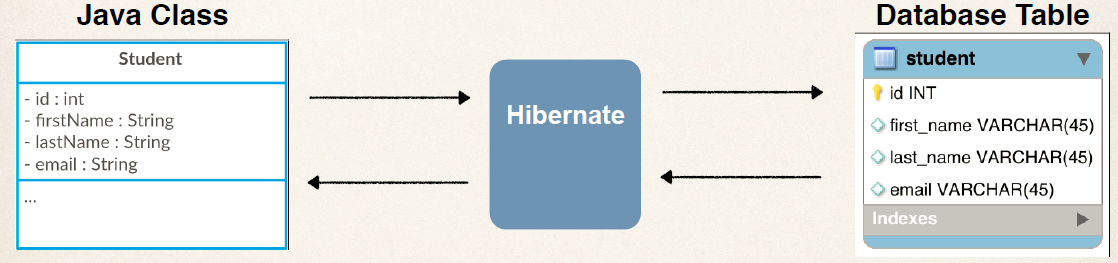
* 人的脅威
  + 誤操作、紛失、置き忘れ、Social Engineering
* エラーが発生する時、エラーの履歴（ローグ）を確認する
* 主なエラー：
  + SQL文法の誤り
  + リソース不足
  + データベースに接続出来ない
  + データロックやタイムアウト
* スロークエリを特定する
* スロークエリが引き起こす問題：
  + 処理の時間が長くなる
  + 他のサーバーに影響を与える
  + サーバーに負荷がかかる
* スロークエリの対策：
  + SQL文を修正する
  + インデックス（索引）を使う
* 検索条件に頻繫に用いるカラムのインデックスを作成することでデータ取得が早くなる
  + データ量が多く、インデックスを行うカラムに格納するデータの種類が多い場合は効果的
  + 索引の弱点：
    - データ編集する時に索引を更新しなければならないので、更新の速度が落ちる
    - ディスク量を消費する
* 処理の性能を上げる方法：
  + パソコンの性能を上げる・高性能な製品に置き換える（スケールアップ）
  + 端末の台数を増やして、処理を分散する（スケールアウト）
  + データベースを複製して、データを同期する

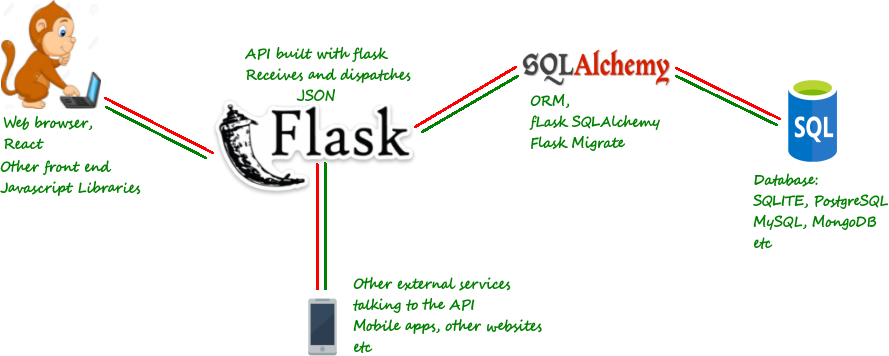
**データベース活用：**

* コマンド操作が基本です
* クライアントソフトの操作が便利で、効率的にデータベースを管理することが出来る
* アプリケーション（Website・スマホアプリ）上で表示するデータ・もらえるデータは裏側でデータベースに保存される
* プログラムからライブラリーやドライバー経由でデータベースを操作する
* クラウドサービスを使う

ORM（Object Relational Mapping）：

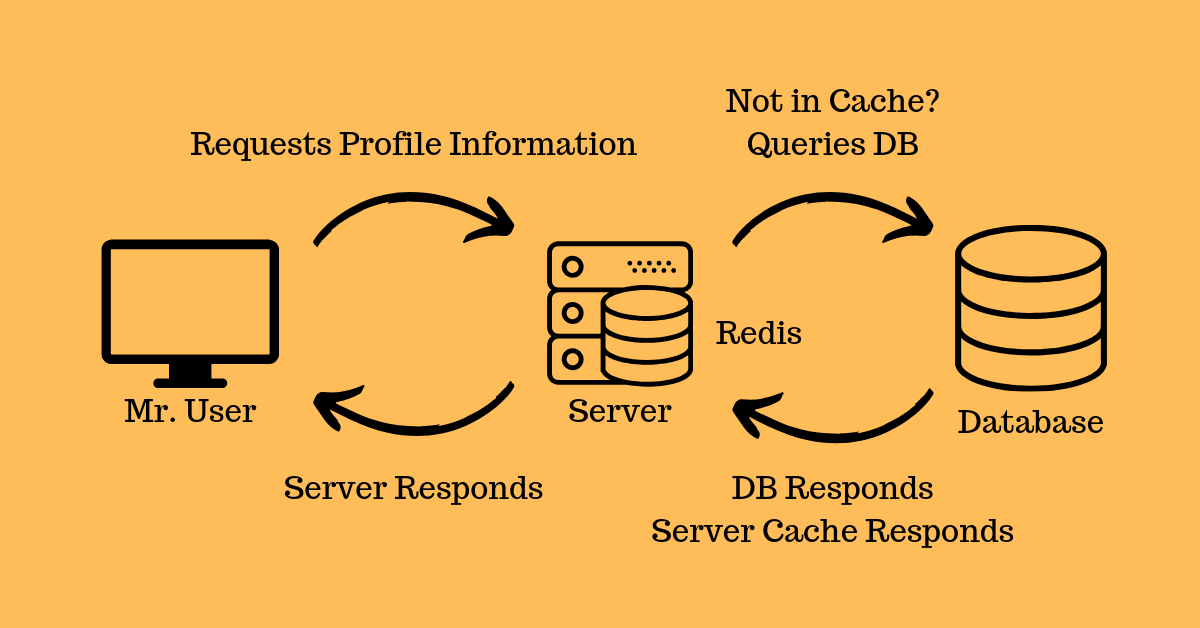
プログラム言語特有の形式でデータベースを扱う仕組み





Cache：

* 一度利用したデータを読み込みの早いディスクに一時的に置いて、次回同じデータを読み込む時に処理が速くさせる仕組みです
* 事例：ブラウザーが画像等の大きいファイルを一時的に保存して、ページの再使用の際にキャッシュを使う
* データベースで変更の頻度が少ないデータや頻繫に読み込むデータにキャシューを使う



* AI Database：
  + 言葉をSQLに変換する機能
  + データの集約・分析によって、SQL文の最適化等