# Pattern of Enterprise Application Architecture
## Chapter 3



# リレーショナルデータベースへのマッピング

WR

WR at Csus4.net
http://www.csus4.net/d/

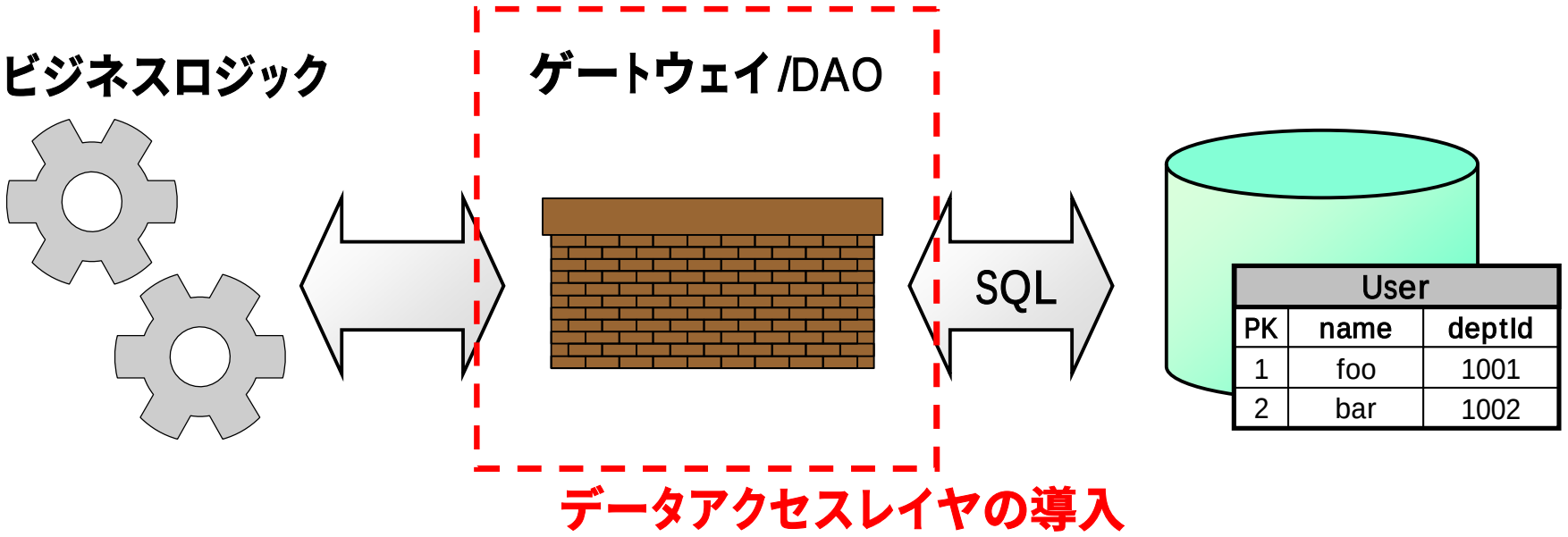
# 日本語版 目次

# 3.1 アーキテクチャに関するパターン

- ゲートウェイ/DAOによるレイヤ化
- ゲートウェイの構成と分類
- ドメインロジック/ビジネスロジックとの関係
- ビューとクエリー
- ドメインモデルの永続化

# ゲートウェイ/DAOによるレイヤ化

P187-188．The Case of the Missing Element

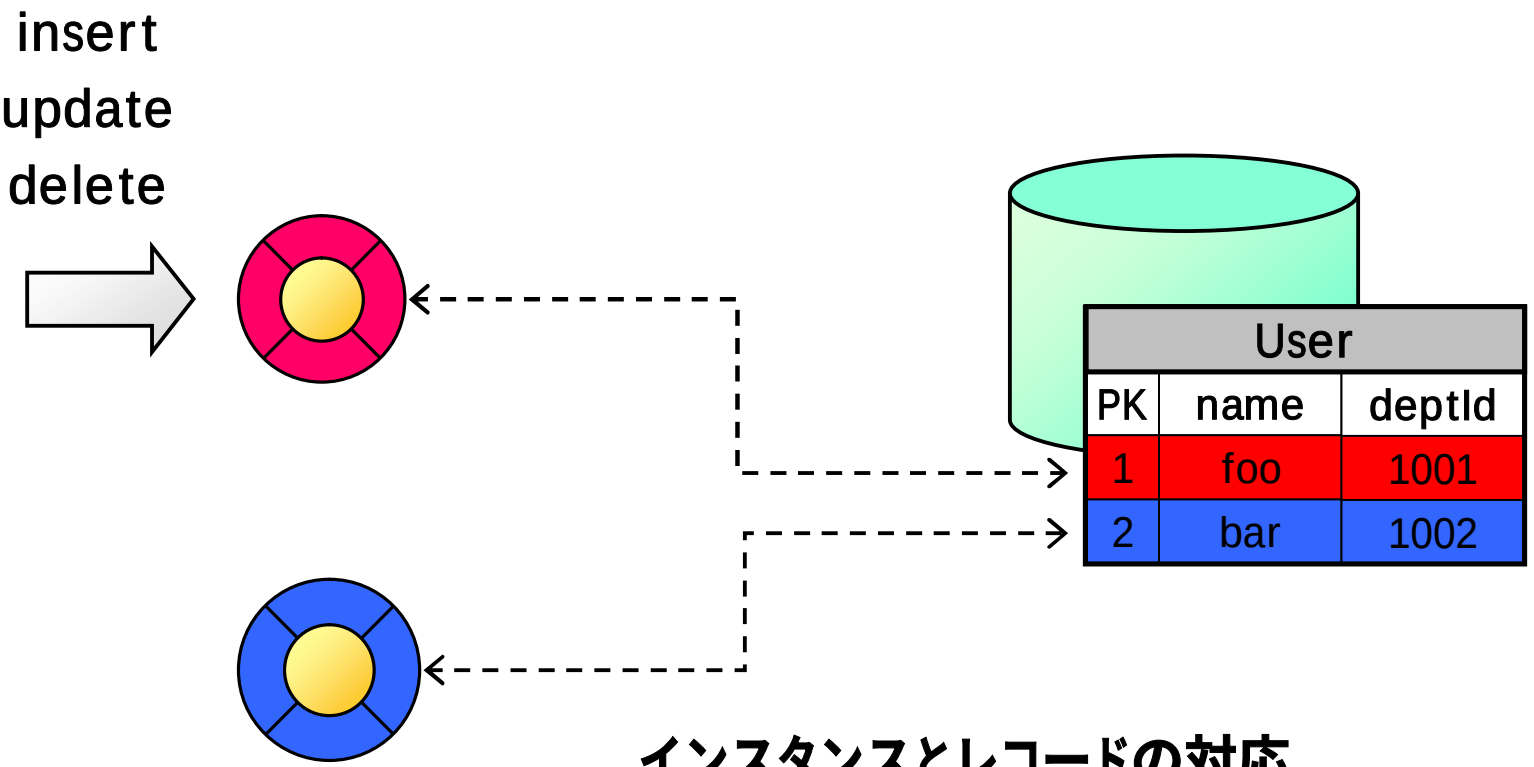



- メリット
  - SQLの隠蔽/分離（プログラム・プログラマ）
  - データアクセスレイヤへのSQLアクセス処理の一元化

# ゲートウェイの構成と分類

- 行データゲートウェイ
  - RDBレコード1つに1インスタンス
  - オブジェクト指向の考え方に適合
- テーブルデータゲートウェイ
  - RDBテーブル1つに1インスタンス
  - レコードセットに適合

# 行データゲートウェイの概念




インスタンスとレコードの対応

# 行データゲートウェイのバリエーション

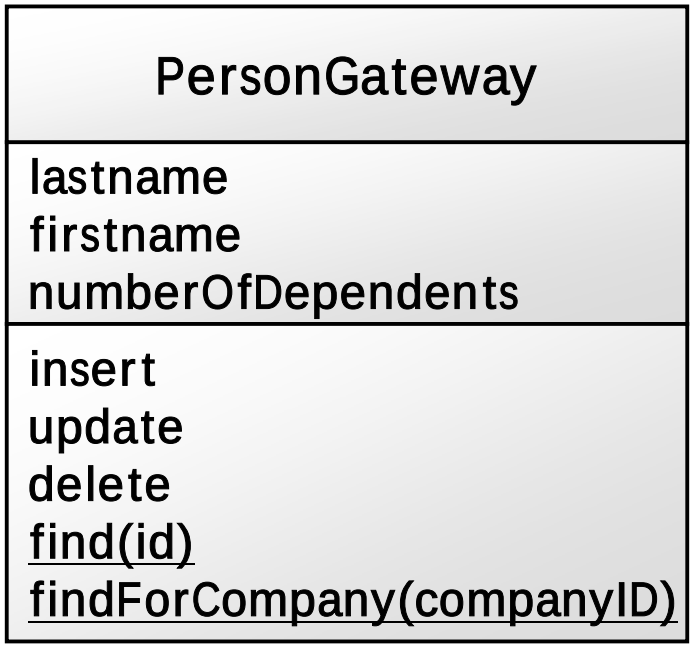
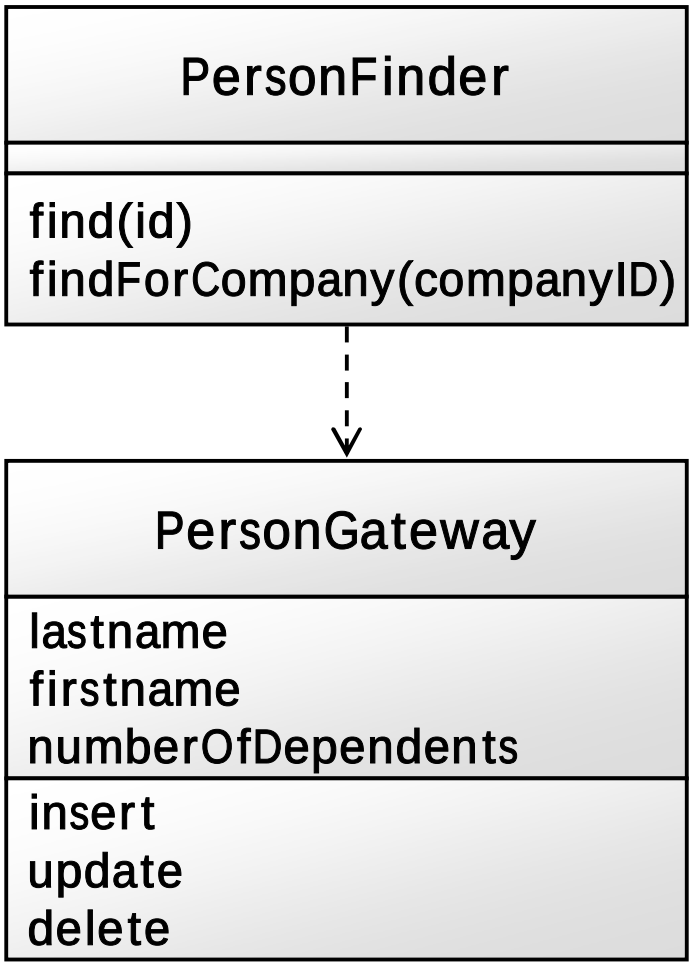
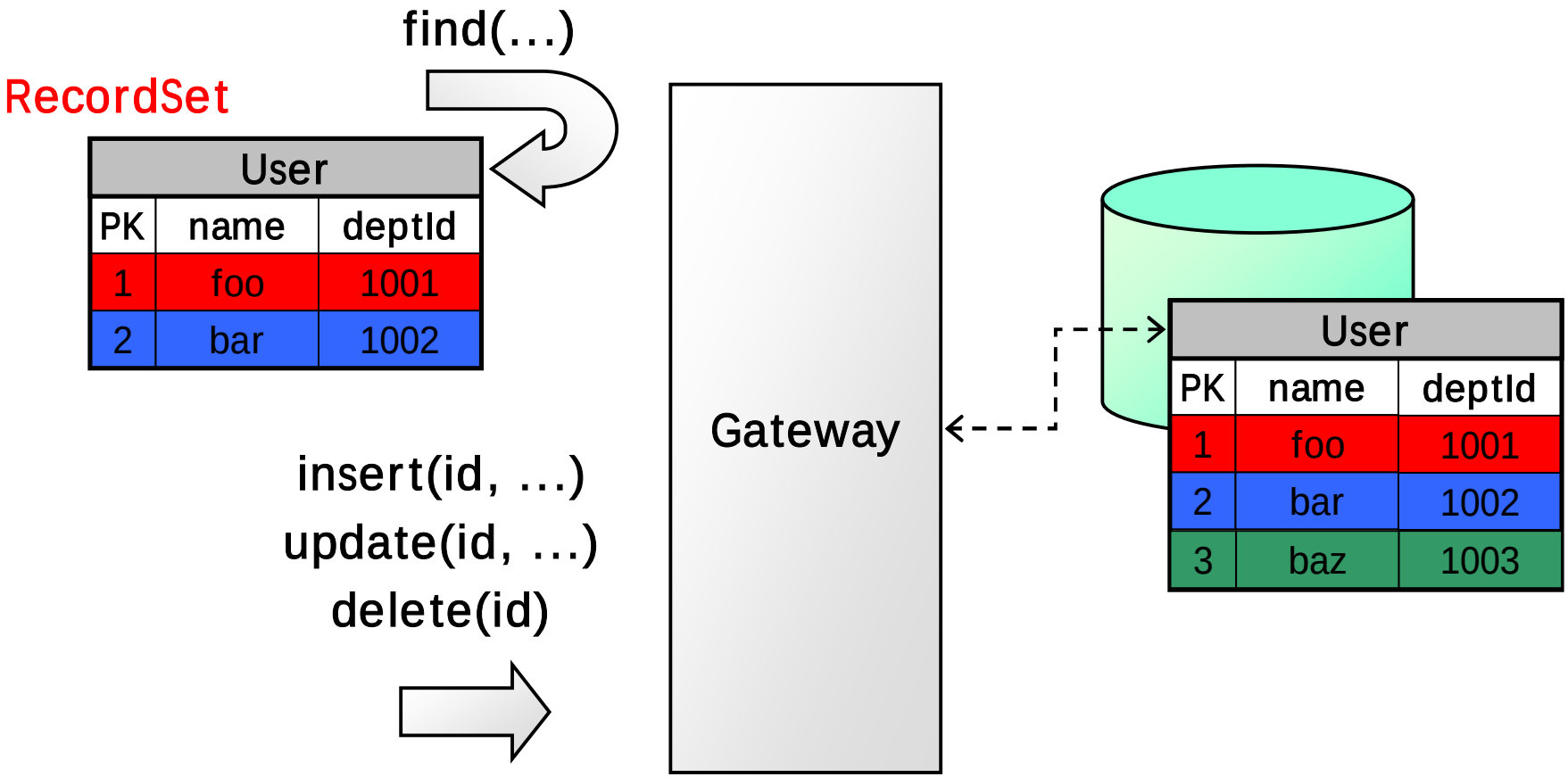



図3.1 (P36)




10.2章 (P162)

# テーブルデータゲートウェイの概念



# テーブルデータゲートウェイ

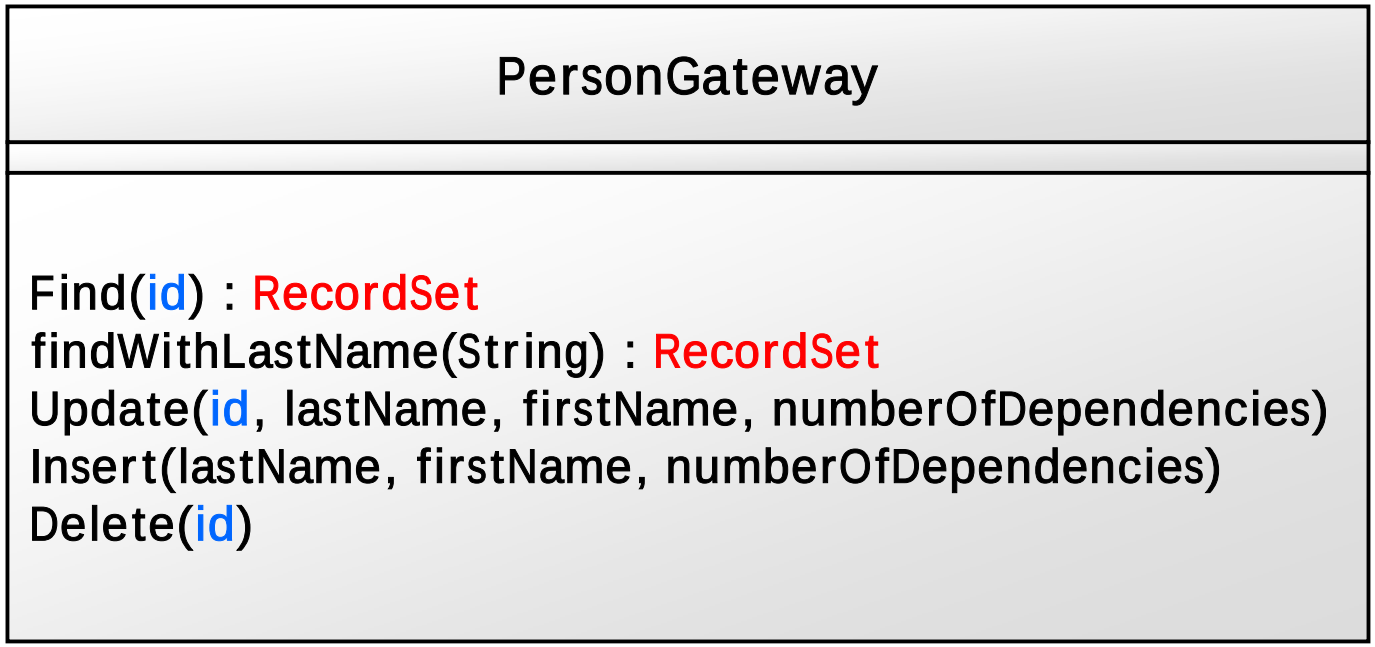
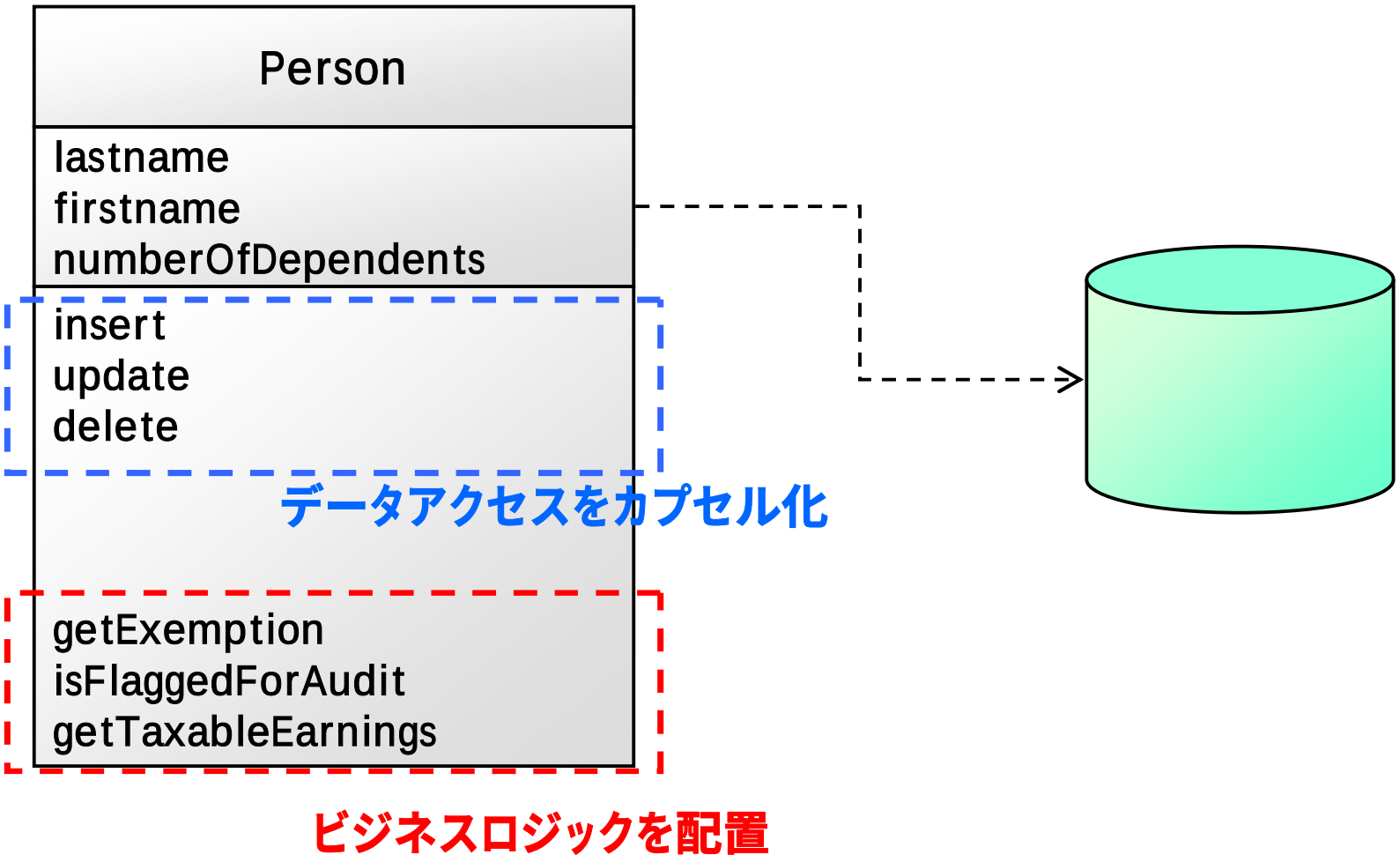



図3.2 (P36)

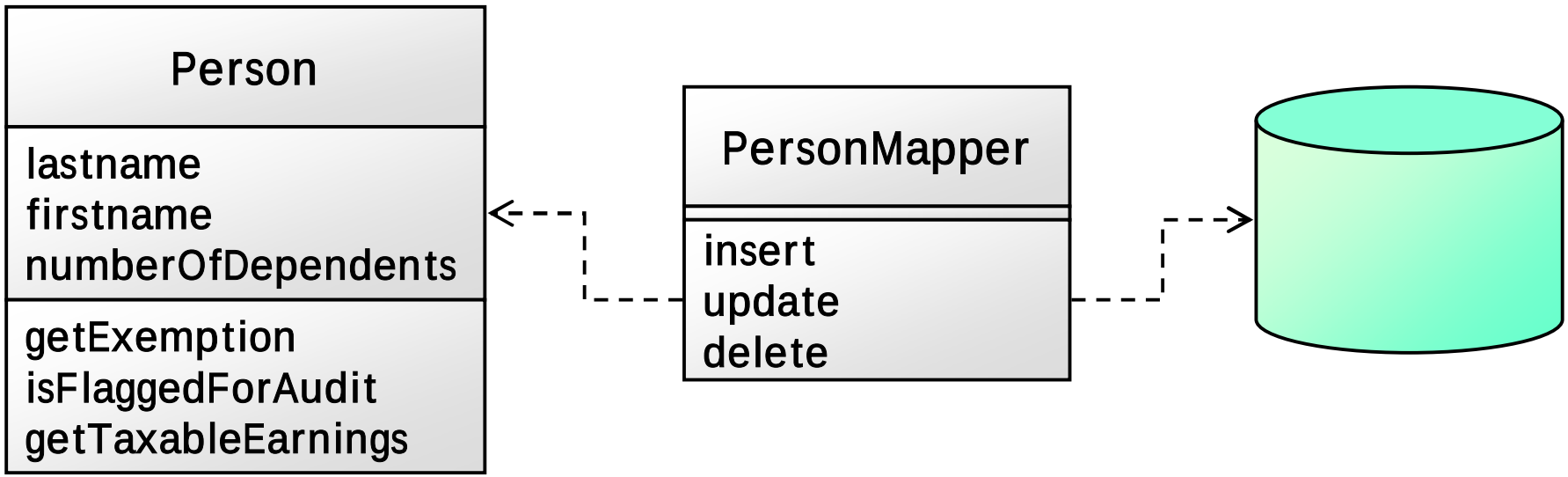
# ドメインロジックとゲートウェイ

- テーブルモジュール
  - → テーブルデータゲートウェイ
- ドメインモデル
  - ロジックがシンプルかつ、テーブルとオブジェクトの関係がシンプル → アクティブレコード
  - ロジックが複雑かつ、テーブルのオブジェクトの関係が複雑 → データマッパー
- 多種のゲートウェイを併用してよい
  - しかし、混乱を招く恐れあり
  - 併用せざるを得ないケースもある

# アクティブレコード


Person
lastname
firstname
numberOfDependents
insert
update
delete
データアクセスをカプセル化
getExemption
isFlaggedForAudit
getTaxableEarnings
ビジネスロジックを配置

# データマッパー


Person
lastname
firstname
numberOfDependents
getExemption
isFlaggedForAudit
getTaxableEarnings
PersonMapper
insert
update
delete

# ドメインモデルの永続化

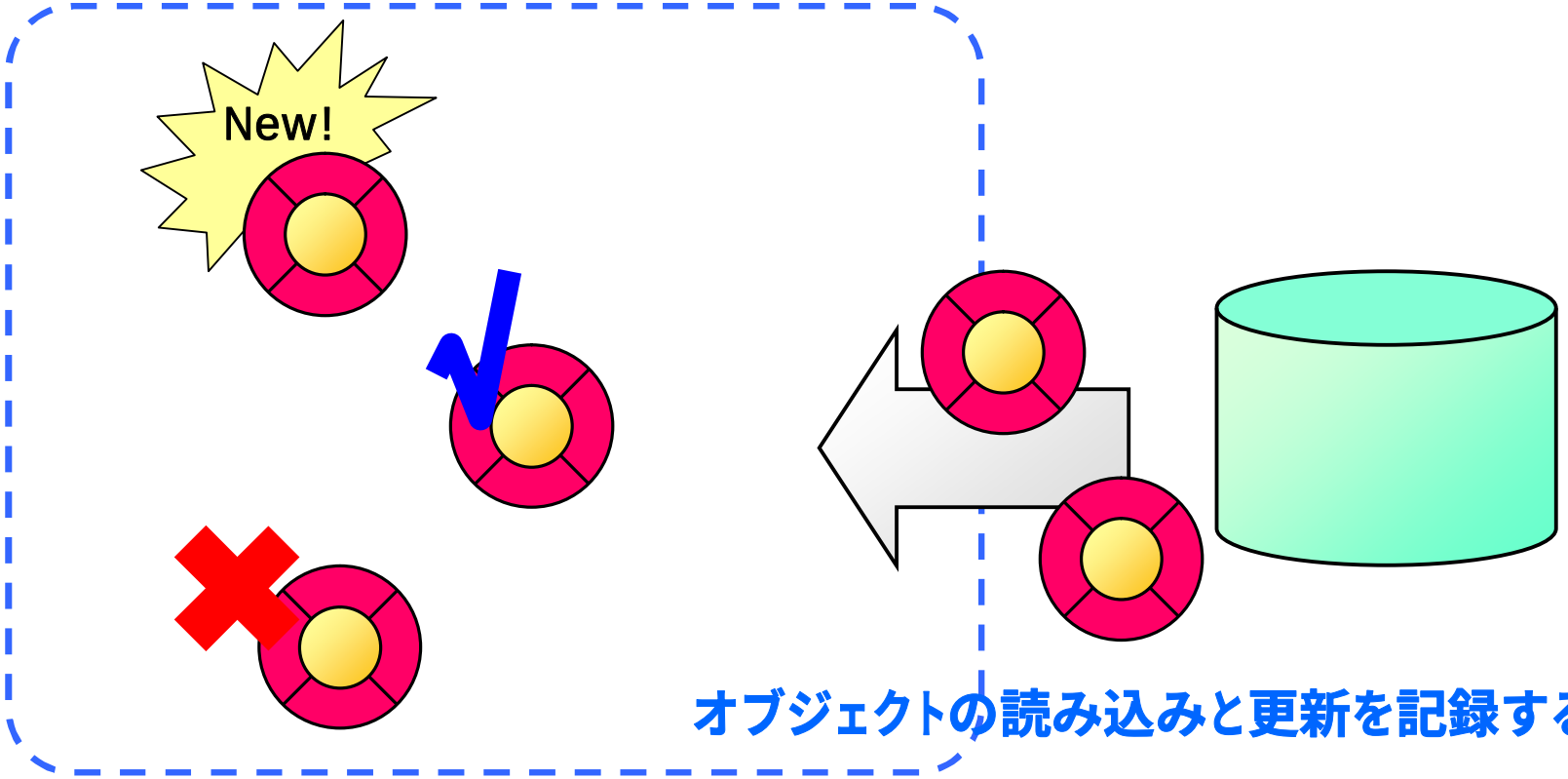
- OODBの利用
  - 最もシンプルなドメインモデルの永続化方法
  - 長所
    - 生産性の向上
  - 短所
    - リスク
- O-Rマッピングツールの利用

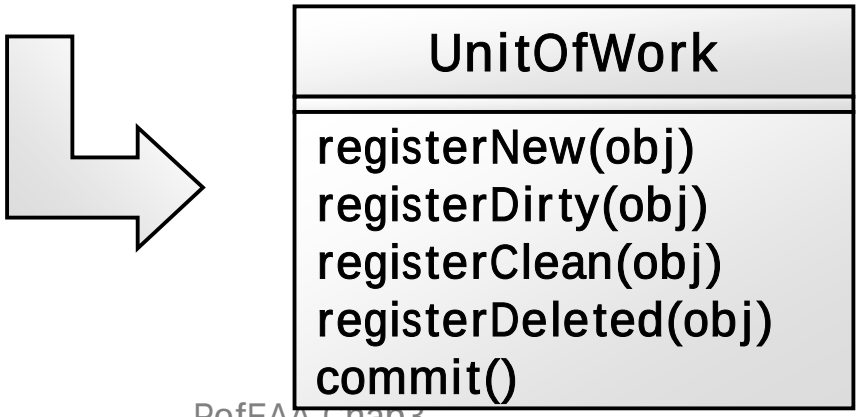
# 3.2 振る舞いに関する問題

- 読み込みと保存に関する

- ユニットオブワーク
- 一意マッピング
- レイジーロード

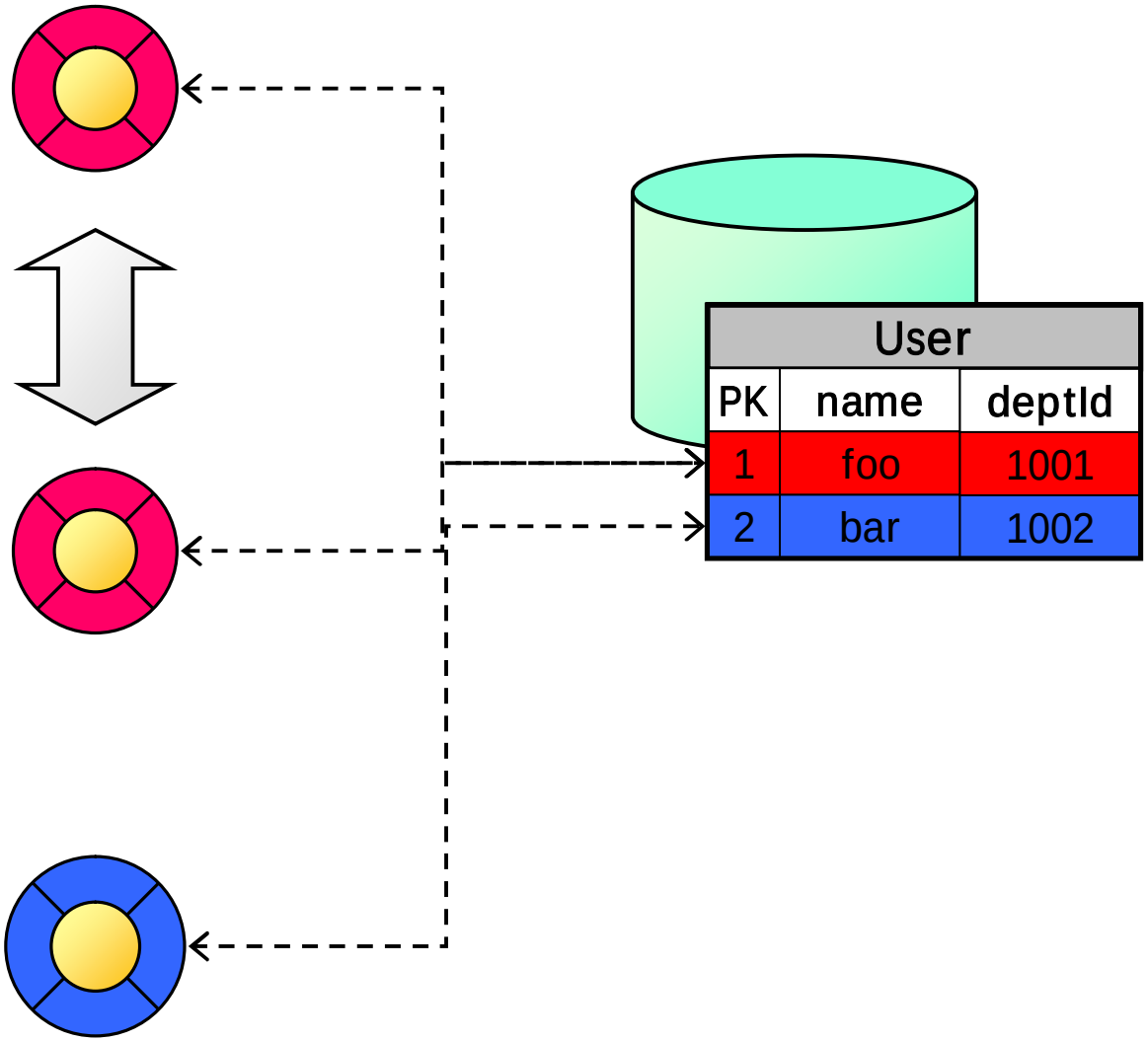
# ユニットオブワークの概念



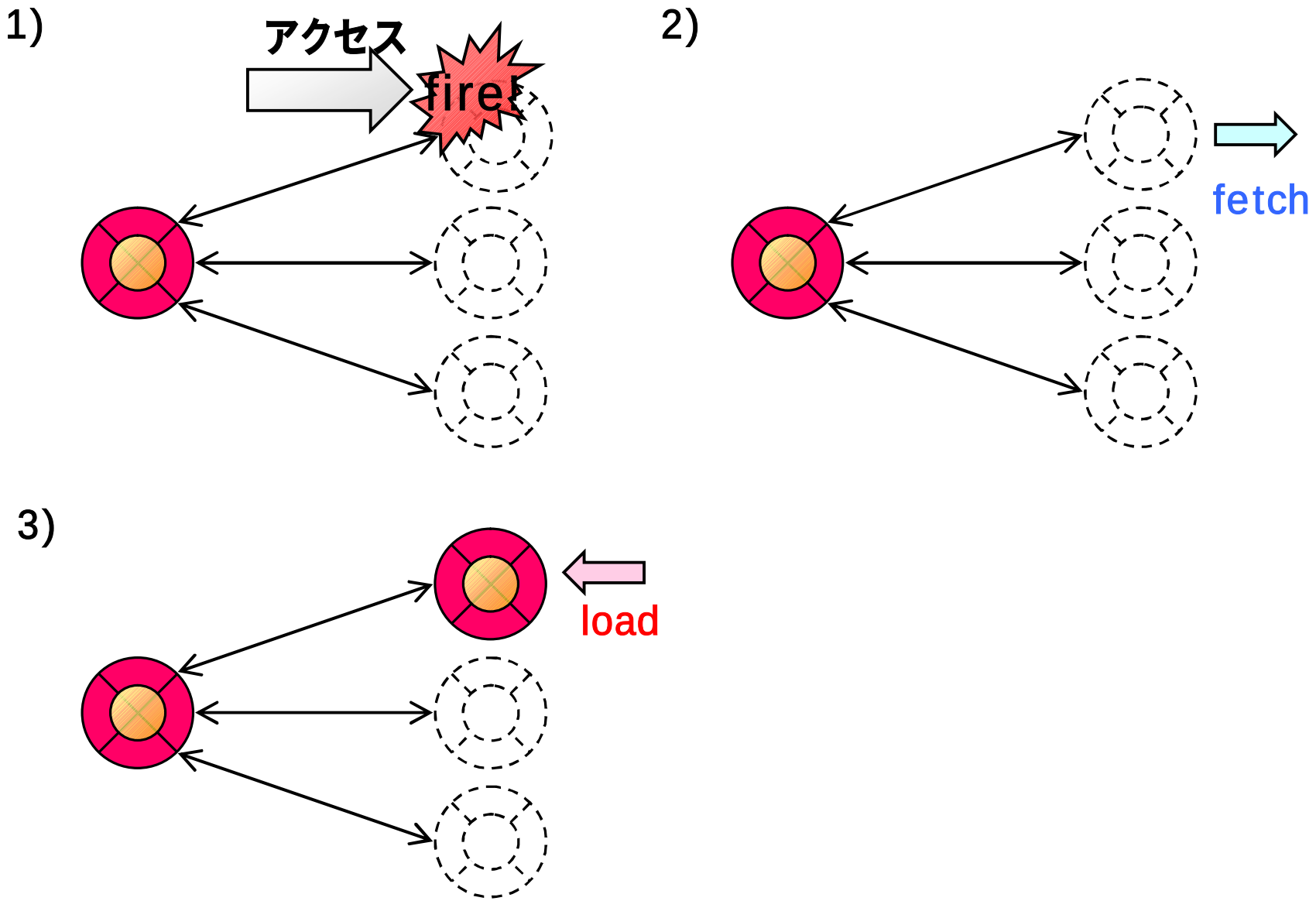



# 一意マッピングの概念

同一レコードに対して2つのオブジェクトが存在すべきでない



# レイジーロードの概念



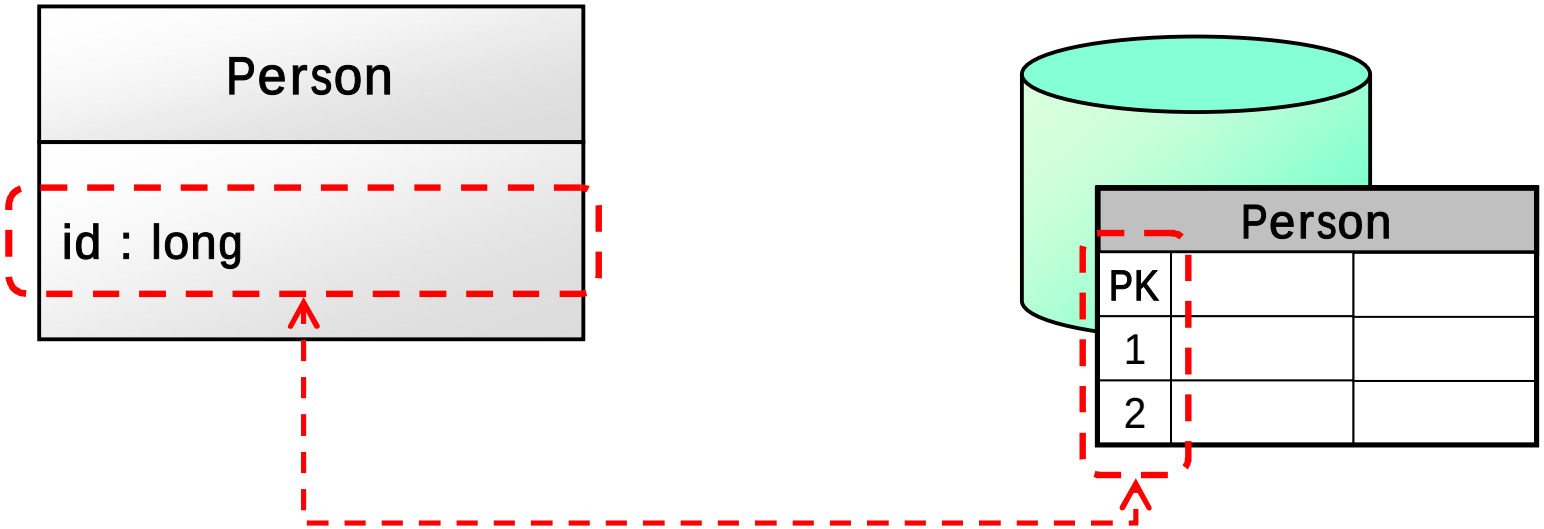
# データの読み込み

- findメソッド
  - 実装方法
    - staticメソッドとして実装する
    - Finderクラスを設ける
- 読み込みの効率化
  - 経験則
    - 1度に複数行を取得するほうが良い
    - ジョインを使用するほうが良い
  - 実際問題・・・
    - DBAに相談せよ
    - プロファイルを取得せよ

# 3.3 構造的なマッピングに関するパターン

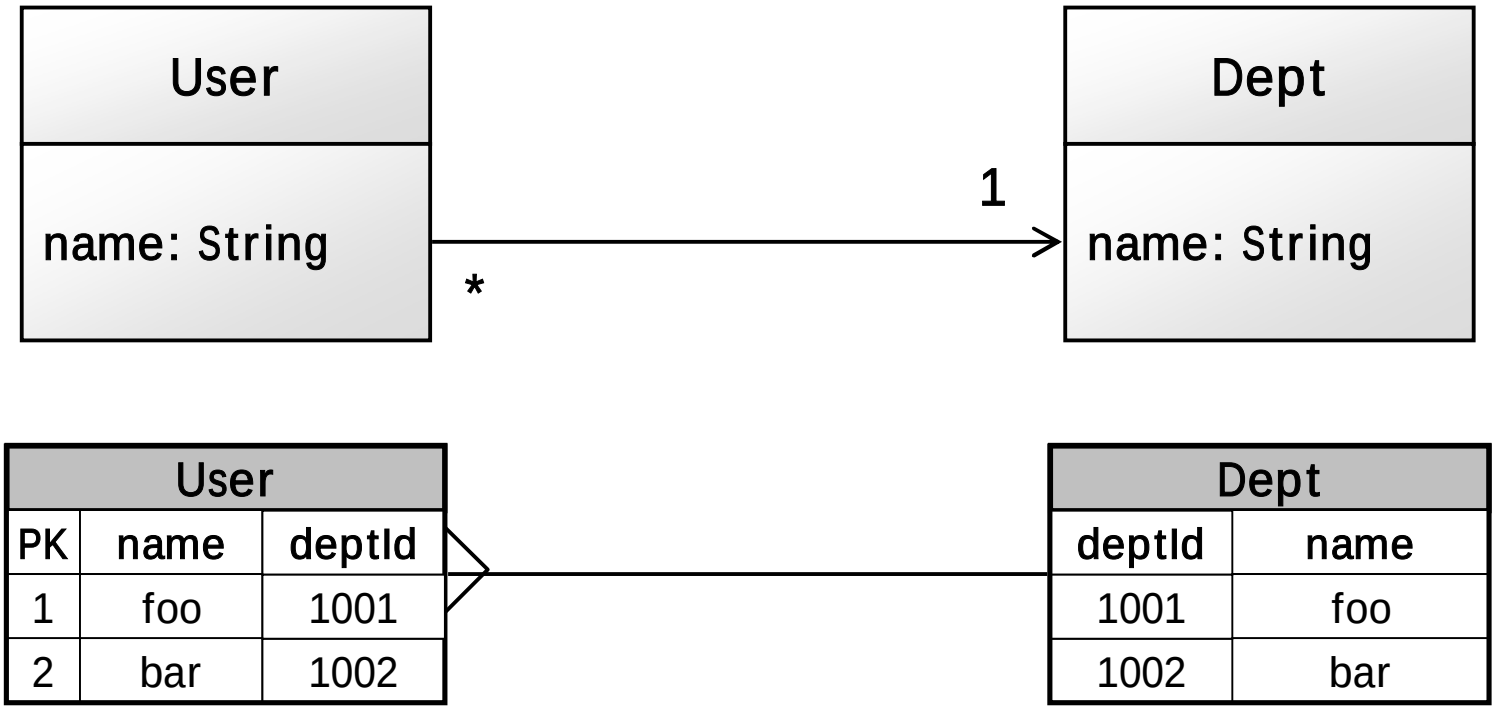
- 関係のマッピング
  - 一意フィールド
  - 外部キーマッピング
  - 関連テーブルマッピング
  - 順序を持ったコレクション
  - バリューオブジェクト
  - シリアライズLOBとXML
- 継承

# 一意フィールド




- テーブルのキーをオブジェクトのフィールドにマッピングする

# 外部キーマッピング


User
name: String
*
1
Dept
name: String
User
PK
name
deptId
1
foo
1001
2
bar
1002
Dept
deptId
name
1001
foo
1002
bar

# 関連テーブルマッピング


Employees

| ID | |
|---|---|
| 1 | |
| 2 | |

skill - employees

| empID | skillID |
|---|---|
| 1 | 1001 |
| 2 | 1002 |

Skills

| ID | name |
|---|---|
| 1001 | foo |
| 1002 | bar |

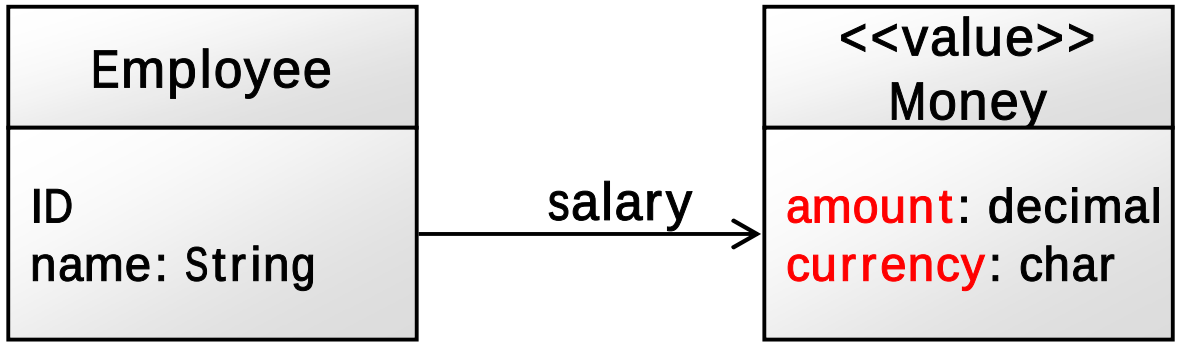
# 順序を持ったコレクション

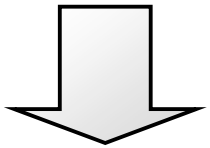
- 順序を持ったコレクション
  - OO言語では一般的かつ有用
  - RDBでは取り扱いが難しい
  - → 順序を持たないコレクションに利用を検討すべき
  - → もしくは、常にあるソート順でデータを取得するようにする

# 参照整合性

- 参照整合性が更新の邪魔になるケースがある
- 参照整合性に引っかからないように更新処理を実行する方法
  - 整合性チェックをコミット時まで遅延させる
  - 更新のトポロジカルソートを行う

# バリューオブジェクト



| Employees | | | |
|---|---|---|---|
| ID | name | salaryAmount | salaryCurrency |
| 1 | | | |
| 2 | | | |

| Employees | | |
|---|---|---|
| PK | name | moneyID |
| 1 | | 1001 |
| 2 | | 1002 |

| Money | | |
|---|---|---|
| moneyID | amount | currency |
| 1001 | | |
| 1002 | | |

# シリアライズLOB/XML

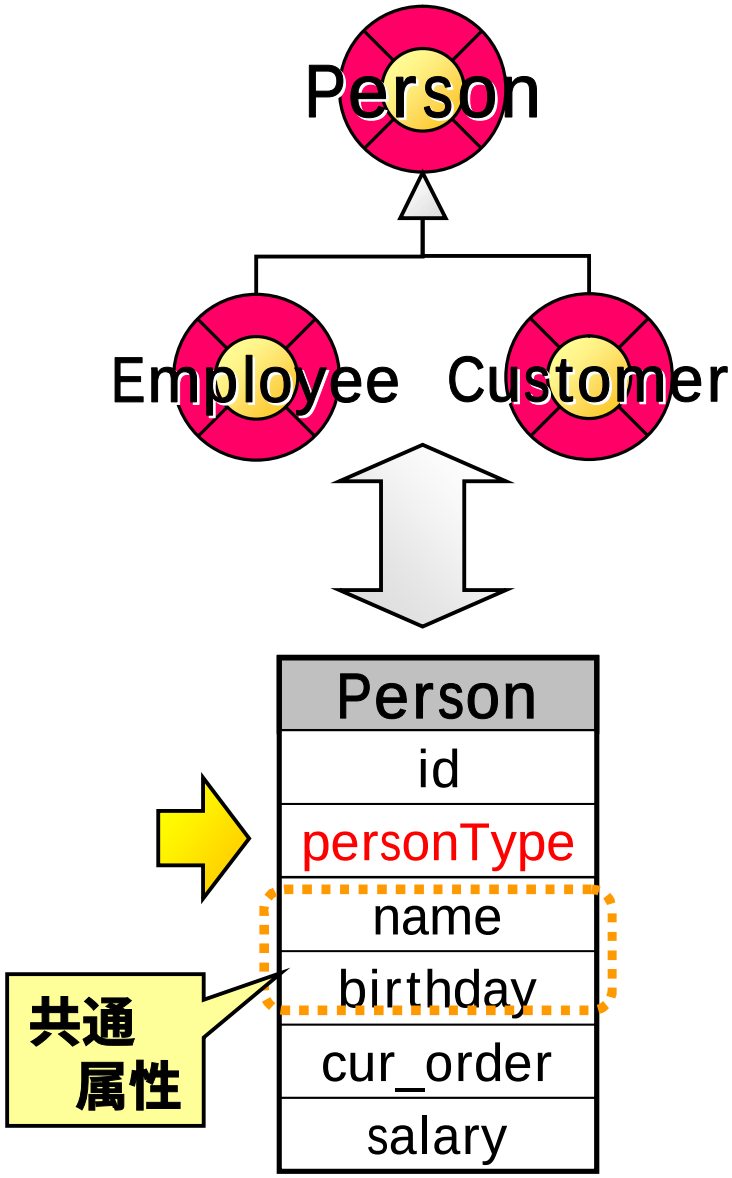
- 利点
  - 階層的データ構造を自然に格納できる
- 欠点
  - クエリのインタフェースがポータブルでない
    - XPathの普及に期待！
- 結論
  - 検索不要な、独立性が高い部分に使用するが吉

# 継承

- シングルテーブル継承
- 具象テーブル継承
- クラステーブル継承

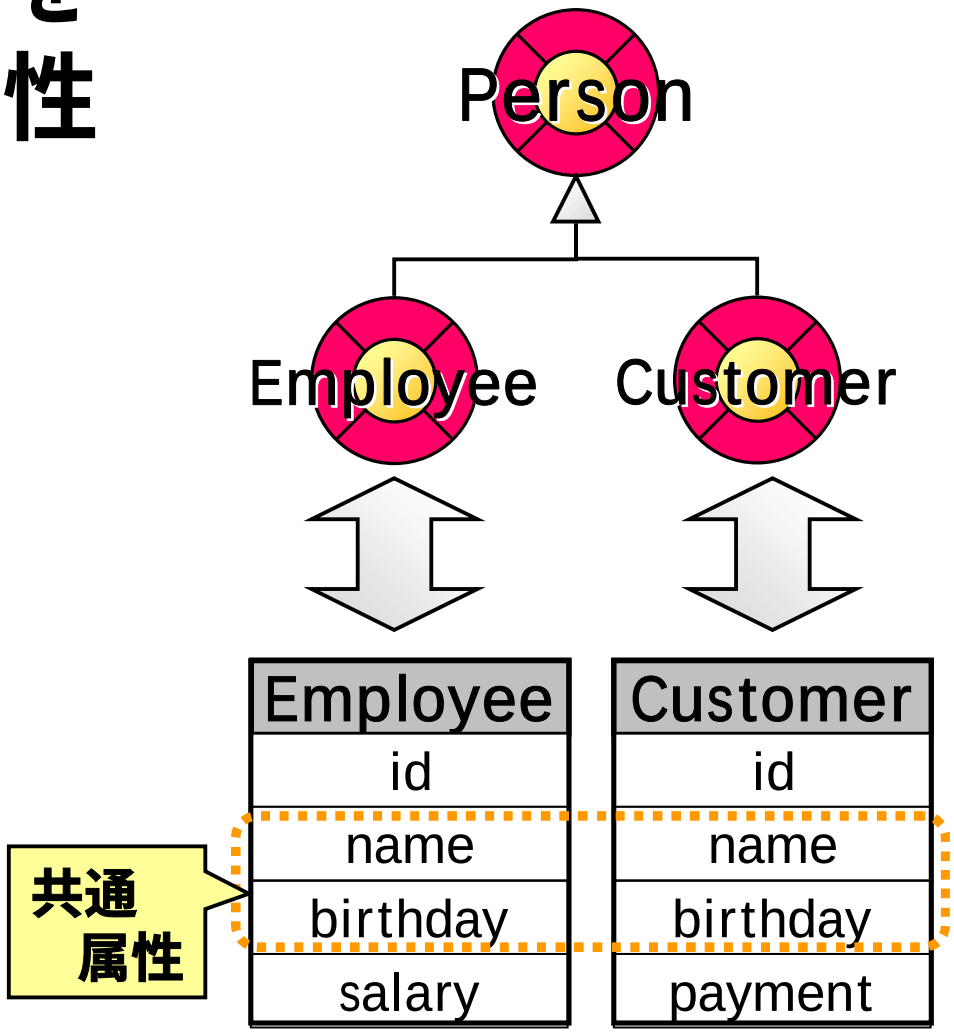
# シングルテーブル継承

- 親Entityと子Entityを単一のテーブルに格納する
- スペースを食う
- パフォーマンス良い
- 継承されたアトリビュートは複製されない
- サブクラス(やスーパークラス)の識別に”type”を用いる



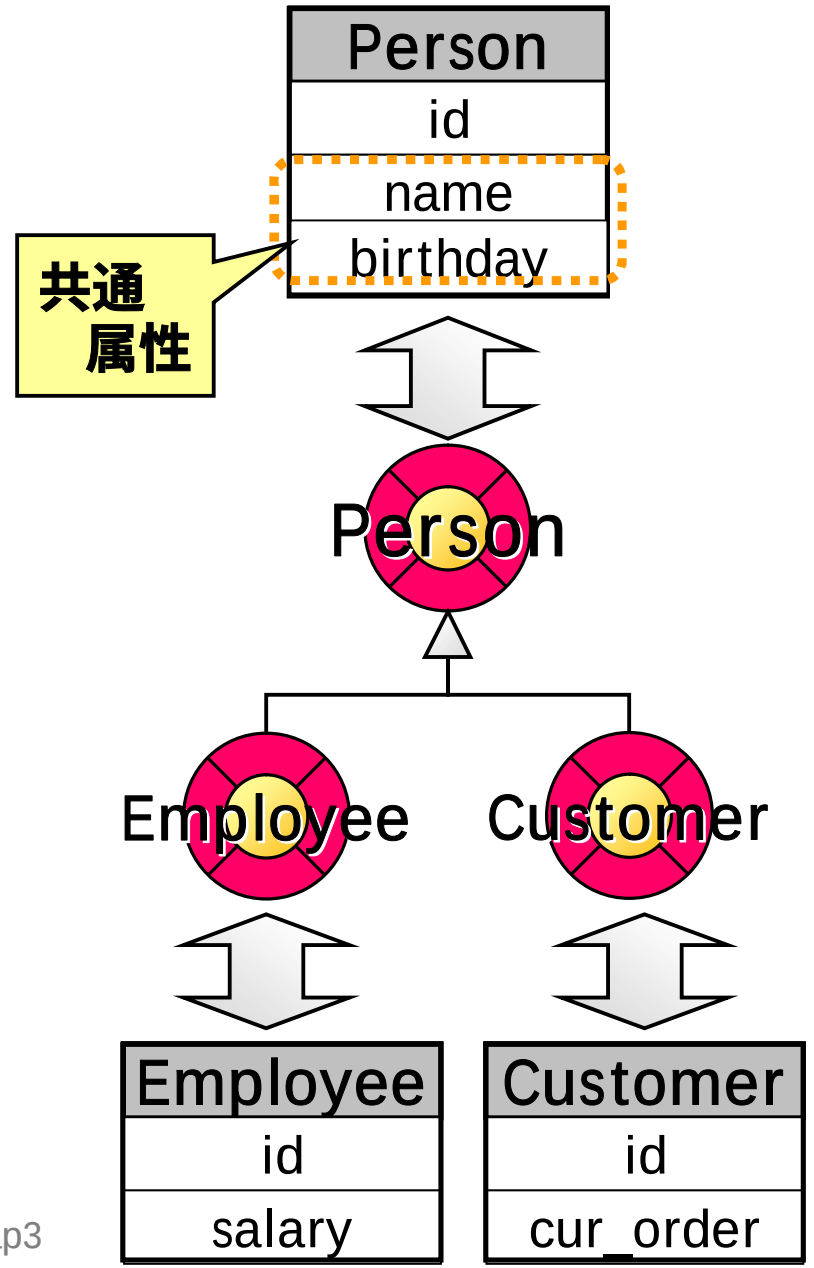
# 具象テーブル継承

- 継承アトリビュートを含め、すべての属性が複製される



# クラステーブル継承

- 継承ツリー間で属性が共有されない



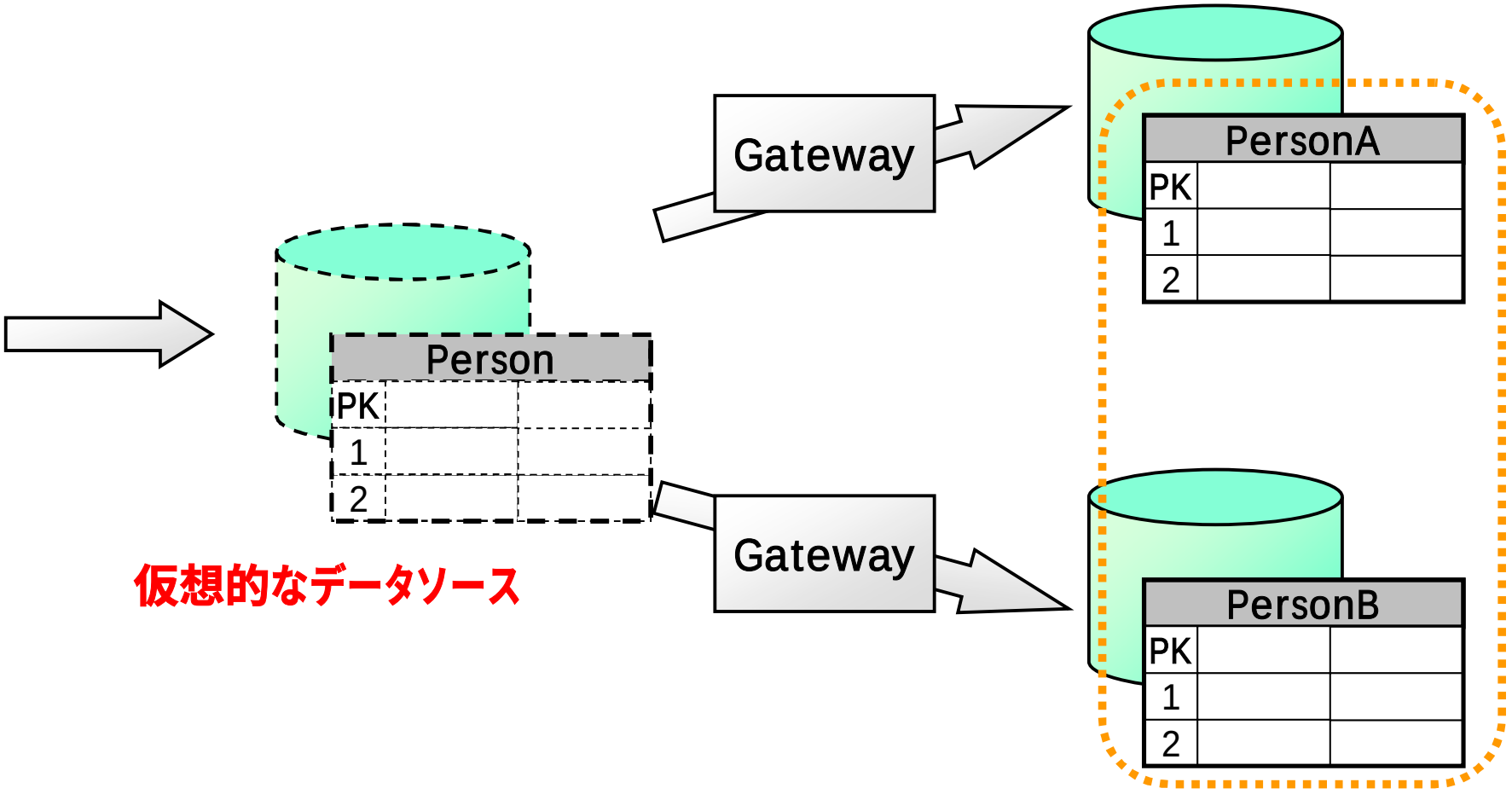
# 多重継承への対処

- WR:**よくわかりません・・・**

# 3.4 マッピングの構築

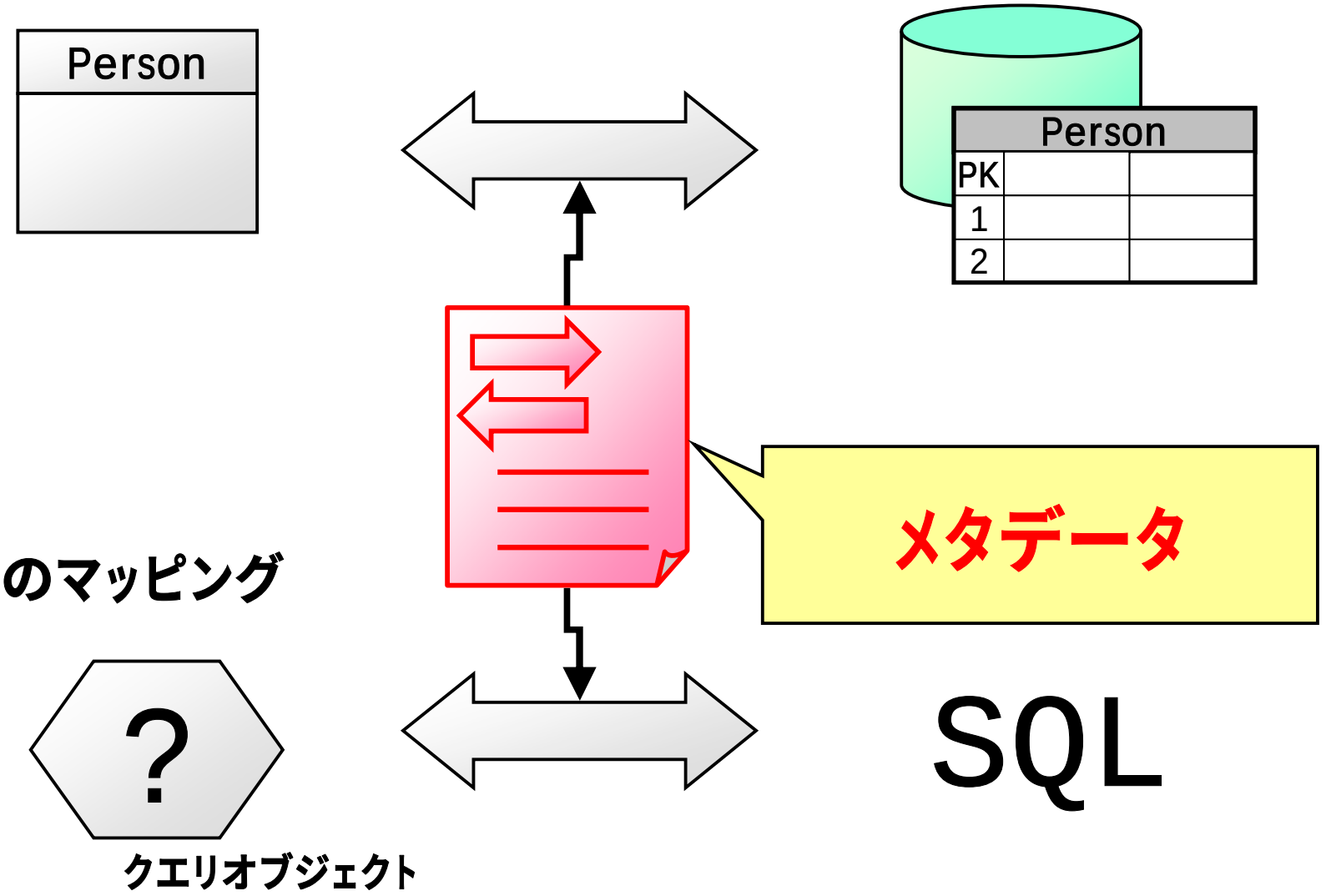
- 3つの状況
  - スキーマを自分で選択する。
  - 変更できない既存のスキーマへのマッピングを行わなくてはならない。
  - 既存のスキーマへのマッピングを行わなければいけないが、このための変更には交渉の余地がある。

# 2重のマッピング


Gateway
PersonA
PK
1
2
Person
PK
1
2
Gateway
PersonB
PK
1
2
仮想的なデータソース
似たようなテーブルが存在

# 3.5 メタデータマッピング

**構造のマッピング**




**クエリのマッピング**

**クエリオブジェクト**

# 3.6 データベース接続

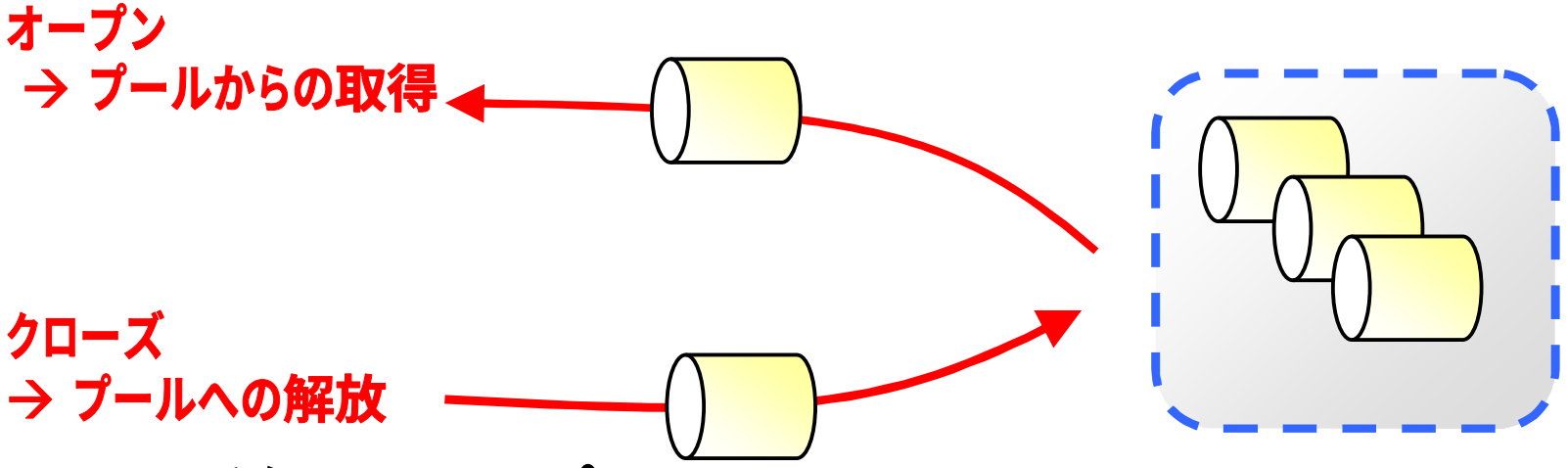
- データベース接続とレコードセット
- データベース接続とコネクションプール
- 接続の管理

# データベース接続とレコードセット

- 切断されたレコードセット
  - 接続を閉じた後に処理される
  - ex. ADO.netの非接続型DataSet
- 接続されたレコードセット
  - 処理される間、接続を維持する必要あり
  - ex. ???

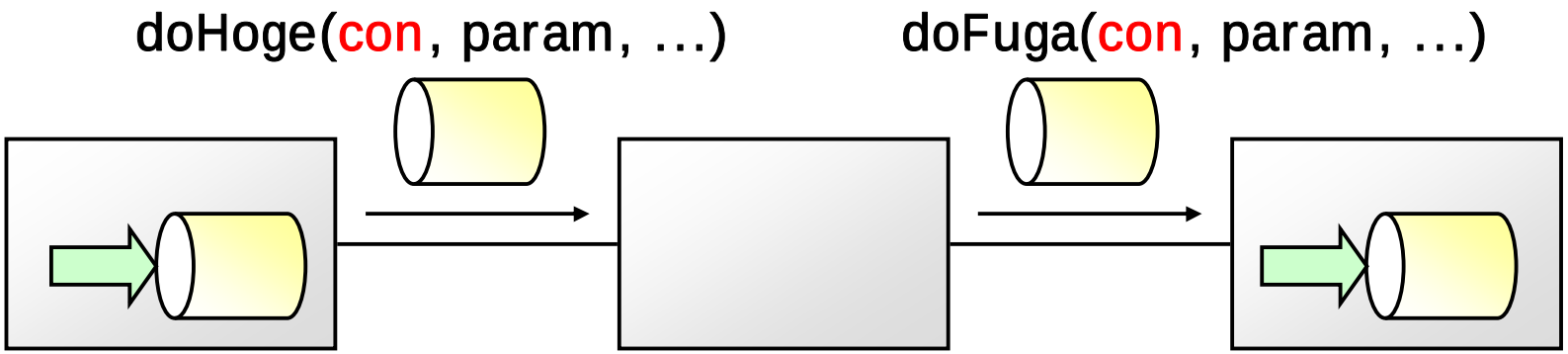
# データベース接続とコネクションプール

- コネクションプールすべき (一般に)
  - 多くのプラットフォームではプールが提供される
- プールのインタフェース
  - 適切にカプセル化されている場合が多い



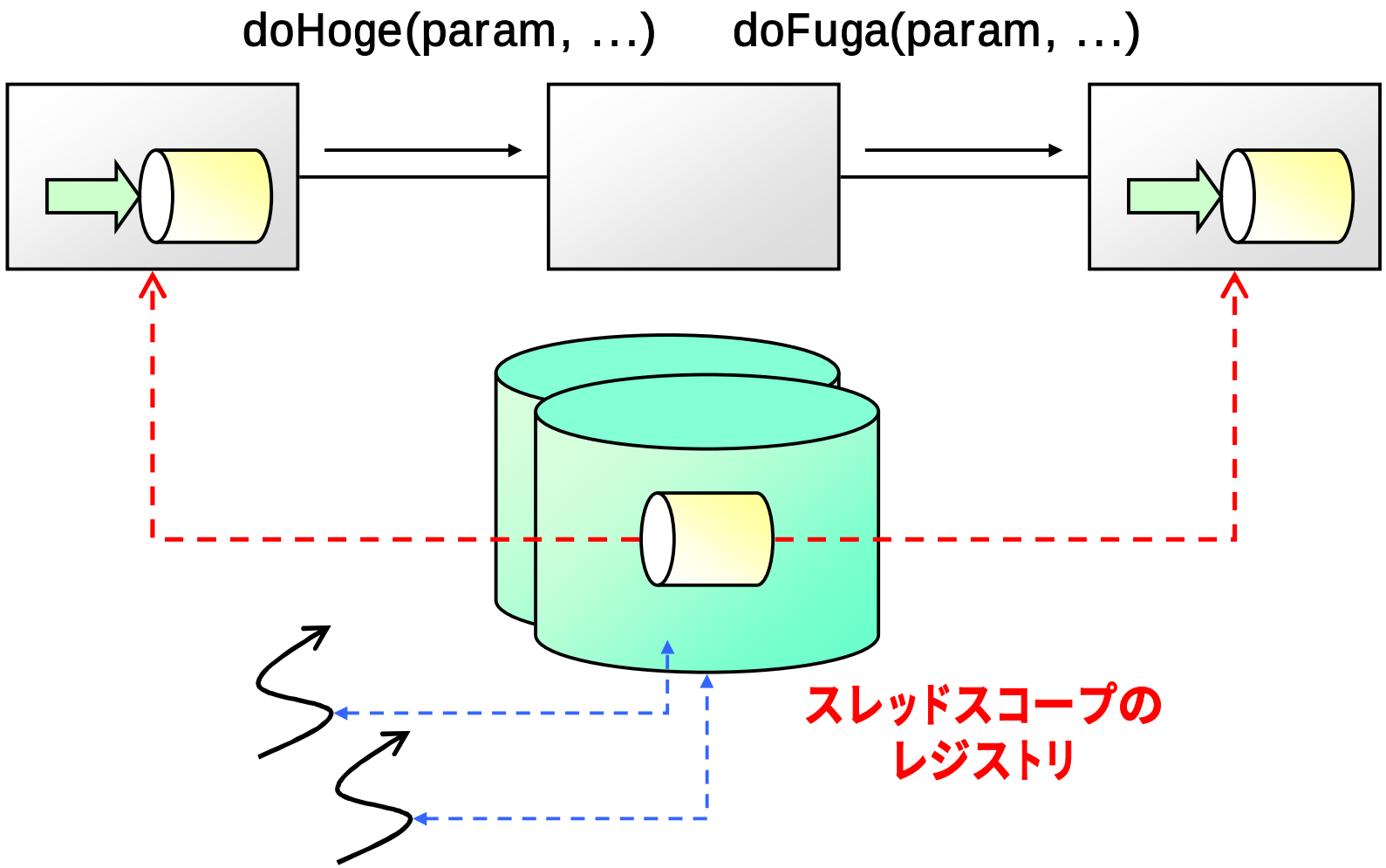

- トランザクションとプール
  - 特定のトランザクションには同一の接続を割り当てる必要がある

# 接続の割り当て法(1)




- 明示的な引数に接続を指定する。
  - コネクションを引き回す格好となる

# 接続の割り当て法(2)



# 接続のクローズ管理

- 方法2つ
  - ガベージコレクションと関連付ける
    - → 推奨しない
  - トランザクションと関連付ける
    - 機能との結びつきがわかりやすく、管理しやすい
      - ユニットオブワーク
    - → 推奨できる

# おわり