

Day3

複数テーブルの参照

技術部データ基盤チーム 財津大夏 / GMO PEPABO inc.

2022.07.14 データエンジニアリング研修 基礎編 Day3

GMOペパボ®

カリキュラム目標と概要

- **Day1: 扱いやすいデータの集合の形を理解できる**
 - データを構造化するための知識の導入
 - **Day2: 初歩的な SQL を使ってデータベースからデータを参照できる**
 - データを参照するために必要な基礎的な知識の導入
 - **Day3: 複数テーブルのデータを組み合わせて参照できる**
 - リレーショナルデータベースからデータを参照するための知識の導入
 - **Day4: データを要約・可視化して情報や知識を取り出すことができる**
 - データを実際の施策や判断に利用するために必要な知識の導入
- ➡ 各日のハンズオンを通して手を動かしながら知識の解釈を高める

テーブルとは

テーブルは正規化されたデータ集合

- 正規化とはデータの重複を無くすこと
 - データを操作する際の対象を減らし不整合などの発生機会を減らす
- 正規化された形を「正規形」という

Day2 データベースとデータの参照

GMO ベバポ

余談: データウェアハウスとリレーショナルデータベースの違い

- BigQuery などのデータウェアハウスは列志向が多い
 - 特定列に対する集計処理が得意
 - 例) 1億行10列のテーブルで列Aの平均を算出 → 列ごとにデータを保存しているので列Aのみ走査
- MySQL, PostgreSQLなどのリレーショナルデータベースは行志向
 - 特定行に対する操作が得意
 - 行の特定を高速に行う仕組み(インデックス)がある
 - 構造的に全列を走査するので列方向の集計処理はリソース効率的に向いていない
- 他にも...
 - Primary Key, Foreign Key(行の一意性を担保する仕組み)の有無^{*1}
 - データをどの程度正規化して保存するかの違い^{*2}

*1) *2) Day 3 で補足します。

正規形には段階がある

- 非正規形
- 第 1 正規形
- 第 2 正規形
- 第 3 正規形

Day1 データの集合の形

GMO ベイパボ

tidy data

- 次のように整理すると意味と構造を一致させられる
 - 1つの観測が1つの行に対応する
 - 1つの変数が1つの列に対応する
 - 1つの観測の単位の類型が1つの表に対応する

⇒ 統計学の「tidy data」

⇒ データベースの「正規形」「テーブル」

person	treatment	result
John Smith	a	—
Jane Doe	a	16
Mary Johnson	a	3
John Smith	b	2
Jane Doe	b	11
Mary Johnson	b	1

画像は [Wickham, H. \(2014\). Tidy Data. Journal of Statistical Software, 59\(10\), 1-23. https://doi.org/10.18637/jss.v059.i10](https://doi.org/10.18637/jss.v059.i10) より引用

非正規形

- 1行の中にデータが繰り返し現れる形
- 単純な行列で表現できないので原則*1データベースのテーブルにはならない

架空の EC プラットフォーム の販売履歴

オーナー氏名	売上振込先	店舗名	商品名	単価	個数	合計額
山田太郎	X銀行	Aのお店	赤いブローチ	1000	2	2000
			青いブローチ	1500	1	1500
田中花子	Y銀行	Bのお店	ピアス	5000	1	5000
		Cのお店	ぬいぐるみ	3000	1	3000

*1: BigQuery では「ネストされたフィールド」「繰り返しフィールド」で表現できる。

<https://cloud.google.com/blog/ja/products/data-analytics/bigquery-explained-working-joins-nested-repeated-data>

第1 正規形

- ・ 繰り返し現れるデータを別々の行に独立させた形
- ・ 他の列の値から求められる列(単価 * 個数 = 合計額)を削除した形

販売履歴テーブル

オーナー氏名	売上振込先	店舗名	商品名	単価	個数
山田太郎	X銀行	Aのお店	赤いブローチ	1000	2
山田太郎	X銀行	Aのお店	青いブローチ	1500	1
田中花子	Y銀行	Bのお店	ピアス	5000	1
田中花子	Y銀行	Cのお店	ぬいぐるみ	3000	1

第 2 正規形

- 第 1 正規形に加えて、主キー*¹以外の列のうち主キーの一部だけで決まるもの*²を別テーブルに分離した形
 - この例では A のお店のオーナー氏名が更新された際の更新レコード数を減らせる

販売履歴テーブル

店舗名	商品名	単価	個数
Aのお店	赤いブローチ	1000	2
Aのお店	青いブローチ	1500	1
Bのお店	ピアス	5000	1
Cのお店	ぬいぐるみ	3000	1

店舗テーブル

店舗名	オーナー氏名	売上振込先
Aのお店	山田太郎	X銀行
Bのお店	田中花子	Y銀行
Cのお店	田中花子	Y銀行

*1: レコードを一意に特定できる列。この例では店舗名, 商品名, オーナー氏名の値がユニークであることを仮定している。

*2: 部分関数従属という。

第3 正規形

- 第2 正規形に加えて、主キー以外の列のうち列同士で依存関係があるもの^{*1}を別テーブルに切り出した形
 - この例では田中花子の売上振込先が変更された場合の更新レコード数を減らせる

販売履歴テーブル

店舗名	商品名	単価	個数
Aのお店	赤いブローチ	1000	2
Aのお店	青いブローチ	1500	1
Bのお店	ピアス	5000	1
Cのお店	ぬいぐるみ	3000	1

店舗テーブル

店舗名	オーナー氏名
Aのお店	山田太郎
Bのお店	田中花子
Cのお店	田中花子

オーナーテーブル

オーナー氏名	売上振込先
山田太郎	X銀行
田中花子	Y銀行

*1: 推移的関数従属という。

余談: データウェアハウスとリレーショナルデータベースの違い その 2

- データウェアハウスは第 1 正規形または非正規形が多い
 - 行の更新が少ない
 - 集計データはひとつのテーブルに格納されている方が分かりやすい
 - テーブル結合が不要なので一般的にクエリ速度が速くなる
- リレーショナルデータベースは第 3 正規形が多い
 - 行の更新が頻繁にあるため不整合を防ぎたい
 - テーブル同士の関連を扱う仕組みがある

Day2 データベースとデータの参照

GMO ベイパホ

余談: データウェアハウスとリレーショナルデータベースの違い

- BigQuery などのデータウェアハウスは列志向が多い
 - 特定列に対する集計処理が得意
 - 例) 1億行10列のテーブルで列Aの平均を算出 → 列ごとにデータを保存しているので列Aのみ走査
- MySQL, PostgreSQLなどのリレーショナルデータベースは行志向
 - 特定行に対する操作が得意
 - 行の特定を高速に行う仕組み(インデックス)がある
 - 構造的に全列を走査するので列方向の集計処理はリソース効率的に向いていない
- 他にも...
 - Primary Key, Foreign Key(行の一意性を担保する仕組み)の有無¹⁾
 - データをどの程度正規化して保存するかの違い²⁾

¹⁾ *2) Day 3 で解説します。

12

テーブルの結合

テーブルを跨いでデータを参照する

- 正規化によって複数テーブルに分離されたデータを参照するにはテーブル結合が必要
 - 例)オーナーごとの販売回数を集計するには販売履歴テーブルと店舗テーブルが必要

販売履歴テーブル

店舗名	商品名	単価	個数
Aのお店	赤いブローチ	1000	2
Aのお店	青いブローチ	1500	1
Bのお店	ピアス	5000	1
Cのお店	ぬいぐるみ	3000	1

店舗テーブル

店舗名	オーナー氏名
Aのお店	山田太郎
Bのお店	田中花子
Cのお店	田中花子

オーナーテーブル

オーナー氏名	売上振込先
山田太郎	X銀行
田中花子	Y銀行

SQL でテーブルを跨いでデータを参照する

- ・ 「JOIN」: FROM 句の中で 2 つのテーブルを結合する演算
 - ・ 結合方法によっていくつか種類がある

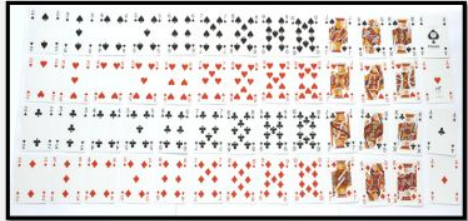
```
SELECT
  オーナー氏名,
  COUNT(*) AS _count
FROM
  販売履歴テーブル
  LEFT OUTER JOIN 店舗テーブル ON 店舗テーブル.店舗名 = 販売履歴テーブル.店舗名
GROUP BY
  オーナー氏名;
```

3 種類の結合方法

- CROSS JOIN (クロス結合)
- INNER JOIN (内部結合)
- OUTER JOIN (外部結合)

$R \times S$

Ranks = {A, K, Q, J, 10, 9, 8, 7, 6, 5, 4, 3, 2}
Suits = {♠, ♥, ♦, ♣}
Ranks × Suits = {
 (A, ♠), (A, ♥), (A, ♦), (A, ♣),
 (K, ♠), ...,
 (2, ♠), (2, ♥), (2, ♦), (2, ♣)
}



By Trainler - Own work, CC BY 3.0,
<https://commons.wikimedia.org/w/index.php?curid=7104281>

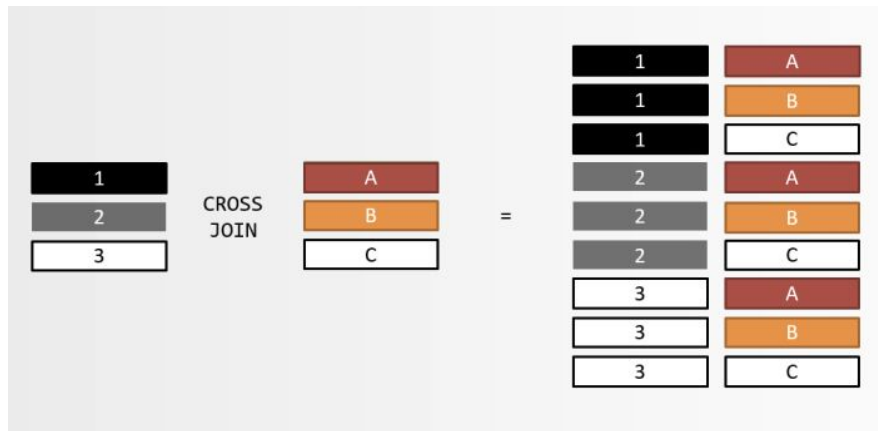
画像は <https://blog.jooq.org/say-no-to-venn-diagrams-when-explaining-joins/> より引用
Copyright 2009-2016 by Data Geekery GmbH. Slides licensed under CC BY SA 3.0

CROSS JOIN

- 2つのテーブルの全行の組み合わせ(デカルト積)を返す
- カンマを使っても記述できる

```
FROM  
  Ranks  
  CROSS JOIN Suits;
```

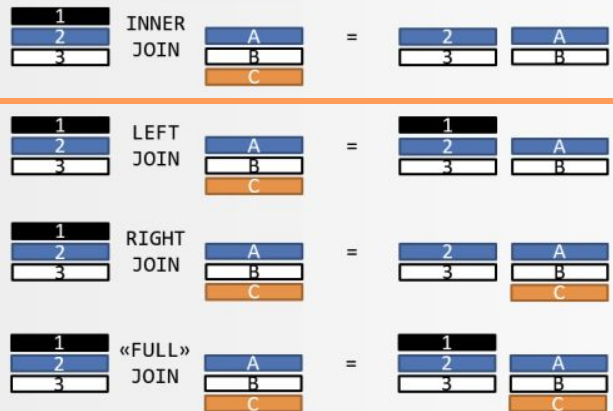
```
FROM  
  Ranks, Suits;
```



INNER JOIN

- 2つのテーブルの全行の組み合わせ(デカルト積)のうち結合条件を満たす行を返す
- 結合条件は ON, USING 句で指定

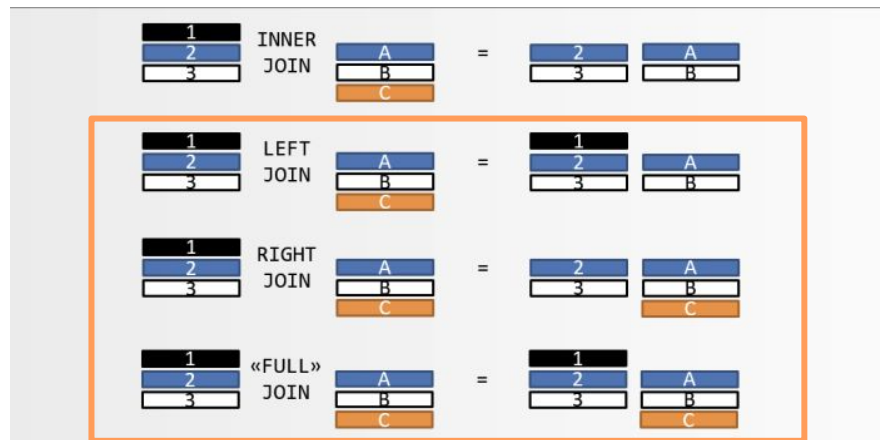
```
FROM
  Numbers
  INNER JOIN Letters
    ON Letters.color = Numbers.color;
# INNER は省略可能
FROM
  Numbers
  JOIN Letters USING (color);
```



OUTER JOIN

- 2つのテーブルの行の組み合わせが、
 - 結合条件を満たす場合はその行を返す
 - 結合条件を満たさない場合も一方または両方のテーブルの全ての行を返す
 - すべての行を返すテーブルによって LEFT/RIGHT/FULL OUTER JOIN の 3 種類

```
FROM
Numbers
LEFT OUTER JOIN Letters
ON Letters.color = Numbers.color;
# OUTER は省略可能
FROM
Numbers
LEFT JOIN Letters USING (color);
```



ハンズオン

ハンズオン

- 以下の内容を取得するクエリを作成してください
 - 📌 (3-1) ユーザーごとに直近半年の間に作成した Issue と Pull Request の数
 - Issue には Pull Request の数を含めてください
 - ユーザーはログイン ID 名で抽出してください
 - 📌 (3-2) suzuri と minne org のリポジトリごとの Workflow の件数
 - 📌 (3-3) suzuri と minne org のリポジトリで Workflow ごとの Run が成功した回数と失敗した回数
 - 📌 (3-4) bigfoot org のリポジトリで Issue ごとの初回/最後のコメント時刻
 - コメントがないものは NULL で構いません

早く終わったら

- Google Cloud Self-Paced Labs をやってみましょう
 - Google BigQuery で SQL を使用して e コマース データセットを操作する
 - <https://www.cloudskillsboost.google/focuses/3618?locale=ja&parent=catalog>
 - BigQuery でのよくある SQL エラーのトラブルシューティング
 - <https://www.cloudskillsboost.google/focuses/3642?locale=ja&parent=catalog>