SQL 基礎(6) JOIN 句 - データの結合

作成日: 2016/02/22
作成者: 西村

更新履歴

更新日	更新概要	作業者
2016/02/22	・　新規作成	西村
	・	
	・	
	・	
	・	

はじめに

この資料では、下記のような JOIN によるテーブル(データ)の結合について簡単に説明します。

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN

サンプルのデータ

この資料では、下記のテーブルをもとに各クエリの結果がどうなるかを示します。

[社員テーブル]

id	姓	名	性別	部署 id
1	山田	太郎	男	1
2	福岡	幸子	女	2
3	東京	次郎	男	3
4	青森	花子	女	1
5	岩手	誠司	男	1
6	山口	三郎	男	999

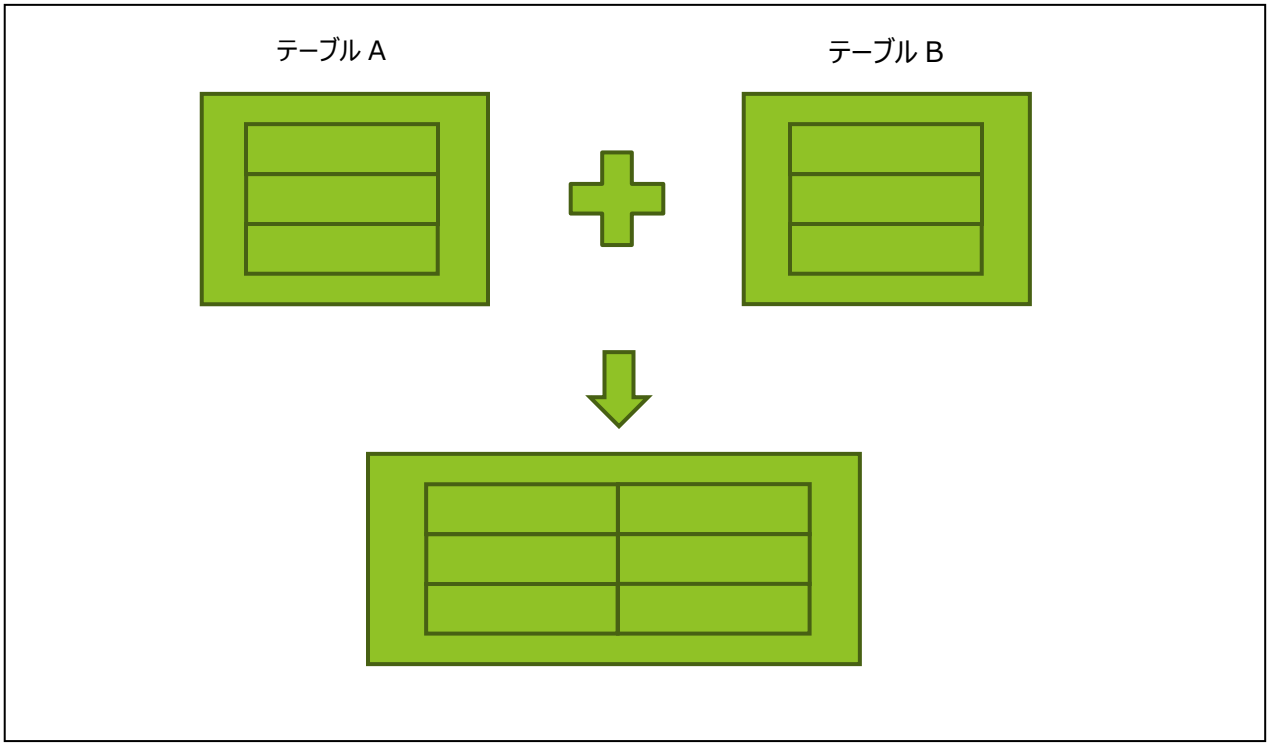
[部署テーブル]

id	部署名
1	開発部
2	技術部
3	総務部
4	経理部

JOIN ?

JOIN は、データベース内の複数のデータを結合する(くっつける)機能です。

たとえば「別のテーブルになっている社員のデータと部署名のデータを一度に取得したい」などデータ同士をくっつけて一度に取得したい場合に使います。



上記のイメージの通り、テーブル同士が横でくっつく感じです。

※ 縦でくっつかせたい場合は「UNION」という別の機能を使います。

基本的な書き方

基本的な書き方は下記のようになります。

```
SELECT 列名
FROM テーブル名
JOIN の種類 別のテーブル名 ON 結合条件
```

JOIN の種類は後述します。

結合条件は、「2 つのデータを何にもとづいてくっつけるか」を DB に教えるための条件です。WHERE と同等の式を書くことができます。(社員.部署 id = 部署.id など。AND や OR を使うこともできます)

JOIN は複数使うことができます。(同じテーブルを何度も使うこともできます)

```
SELECT 列名
FROM テーブル名
JOIN の種類 別のテーブル名 1 ON 結合条件
JOIN の種類 別のテーブル名 2 ON 結合条件
JOIN の種類 別のテーブル名 3 ON 結合条件
```

INNER JOIN

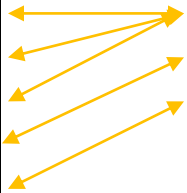
INNER JOIN は「内部結合」と呼び、2 つのテーブル間の行で「くっつくことができた」内の行だけ結果として表示する結合方式です。

(例) 社員の部署 id と部署の id が一致するもので INNER JOIN する場合

```
SELECT 社員.id, 社員.姓, 社員.部署 id,部署.部署名
FROM 社員
INNER JOIN 部署 ON 社員.部署 id = 部署.id;
```

[社員]

id	姓	…	**部署 id**
1	山田	…	1
2	福岡	…	1
3	東京	…	1
4	青森	…	2
5	岩手	…	3
6	山口	…	999



[部署]

id	部署名
1	開発部
2	技術部
3	総務部
4	経理部

[結果]

id	姓	**部署 id**	部署名
1	山田	1	開発部
2	福岡	1	開発部
3	東京	1	開発部
4	青森	2	技術部
5	岩手	3	総務部

http://sqlfiddle.com/#!9/d440ed/2

- 社員側にある部署 id = 999 の行は部署側に id = 999 の行がないので結果から取りのぞかれます。
- 部署側にある id = 4 の行は社員側に部署 id = 4 の行がないので結果から取りのぞかれます。

LEFT JOIN

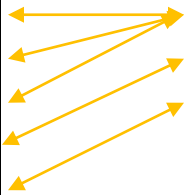
LEFT JOIN(LEFT OUTER JOIN)は「左外部結合」と呼び、2 つのテーブル間の行で「くっつくことができた」内の行 + くっつくことができなかったの左側のテーブルの行を結果として出すという結合方式です。

(例) 社員の部署 id と部署の id が一致するもので LEFT JOIN する場合

```
SELECT 社員.id, 社員.姓, 社員.部署 id,部署.部署名
FROM 社員
LEFT JOIN 部署 ON 社員.部署 id = 部署.id;
```

[社員]

id	姓	…	**部署 id**
1	山田	…	1
2	福岡	…	1
3	東京	…	1
4	青森	…	2
5	岩手	…	3
6	山口	…	999



[部署]

id	部署名
1	開発部
2	技術部
3	総務部
4	経理部

[結果]

id	姓	**部署 id**	部署名
1	山田	1	開発部
2	福岡	1	開発部
3	東京	1	開発部
4	青森	2	技術部
5	岩手	3	総務部
6	山口	999	(NULL)

http://sqlfiddle.com/#!9/d440ed/3

- 社員側にある部署 id = 999 の行は部署側に id = 999 の行がありませんが、左側をベースにするため結果に含まれます。(部署名は結合できなかったので NULL で出てきます)
- 部署側にある id = 4 の行は user 側に部署 id = 4 の行がなかったので結果から取りのぞかれます。

RIGHT JOIN

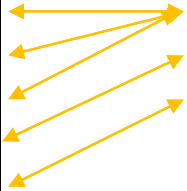
RIGHT JOIN(RIGHT OUTER JOIN)は「右外部結合」と呼び、2 つのテーブル間の行で「くっつくことができた」内の行 + くっつくことができなかったの右側のテーブルの行を結果として出すという結合方式です。

(例) 社員の部署 id と部署の id が一致するもので RIGHT JOIN する場合

```
SELECT 社員.id, 社員.姓, 社員.部署 id,部署.部署名
FROM 社員
RIGHT JOIN 部署 ON 社員.部署 id = 部署.id;
```

[社員]

id	姓	…	**部署 id**
1	山田	…	1
2	福岡	…	1
3	東京	…	1
4	青森	…	2
5	岩手	…	3
6	山口	…	999



[部署]

id	部署名
1	開発部
2	技術部
3	総務部
4	経理部

[結果]

id	姓	**部署 id**	部署名
1	山田	1	開発部
2	福岡	1	開発部
3	東京	1	開発部
4	青森	2	技術部
5	岩手	3	総務部
(NULL)	(NULL)	(NULL)	経理部

http://sqlfiddle.com/#!9/d440ed/6

- 社員側にある部署 id = 999 の行は部署側に id = 999 の行なかったので結果から取りのぞかれます。

- 部署側にある id = 4 の行は社員側に部署 id = 4 の行がありませんが、右側をベースにするため結果に含まれます。
  (社員から表示しようとしていた列は社員と結合できなかったので NULL で出てきます)

左側、右側？

SQL での「左側」は「FROM に書いてあるテーブル」、「右側」は「○○ JOIN に書いてあるテーブル」のことです。

```
SELECT 社員.id, 社員.姓, 社員.部署 id,部署.部署名
FROM 社員
LEFT JOIN 部署 ON 社員.部署 id = 部署.id;
```

↑の場合、FROM 社員 の部分が左側、その次の部署が右側になります。

※JOIN が続く場合は先に出てきたテーブルが左です。

どの JOIN を使ったらいい？

基本的には INNER JOIN と LEFT JOIN しか使うことはほとんどありません。

サンプルのデータで言うと、

- 「部署 ID とくっつかない社員データはいらない(またはありえない)から結果からのぞこう」という場合
  → INNER JOIN
- 「部署 ID とくっつかない社員データがあってもとりあえず全社員出したい」という場合
  → LEFT JOIN

になります。(RIGHT JOIN は順番が変わるだけで LEFT JOIN と同じなのでわざわざ RIGHT JOIN を使う必要がない)

結合する行が複数あった場合

例えば部署テーブルに部署 id=3 の行が 2 件あるなど、結合する行が複数あった場合は結果も同じ行出てきます。

[社員]

id	姓	…	**部署 id**
1	山田	…	1
2	福岡	…	1
3	東京	…	1
4	青森	…	2
5	岩手	…	3
6	山口	…	999

[部署]

id	部署名
1	開発部
2	技術部
3	総務部
4	経理部
3	総務部 2

[結果]

id	姓	**部署 id**	部署名
1	山田	1	開発部
2	福岡	1	開発部
3	東京	1	開発部
4	青森	2	技術部
5	岩手	3	総務部
5	岩手	3	総務部 2

変な行が出てくる…という場合はまず「結合の条件が甘いのではないか」「データがおかしくないか」を確認してみるとよいです。

ON と WHERE

ON の結合条件でできることは WHERE でもできることがあります。

(通常の INNER JOIN の例)

```
SELECT 社員.id, 社員.姓, 社員.部署 id,部署.部署名
FROM 社員
INNER JOIN 部署 ON 社員.部署 id = 部署.id;
```

(INNER JOIN に ON で結合条件を付けずに WHERE に持っていくこともできます)

```
SELECT 社員.id, 社員.姓, 社員.部署 id,部署.部署名
FROM 社員
INNER JOIN 部署
WHERE 社員.部署 id = 部署.id;
```

ただ、WHERE の場合は「全部の行を掛けあわせ(CROSS JOIN)してから抽出する」という動きになるので一般的に遅いですしわかりづらいのでやめましょう。(次のページのような形になります)

id	姓	**部署 id**	部署名	(部署.id)
1	山田	1	開発部	1
1	山田	1	技術部	2
1	山田	1	総務部	3
1	山田	1	経理部	4
2	福岡	2	開発部	1
2	福岡	2	技術部	2
2	福岡	2	総務部	3
2	福岡	2	経理部	4
3	東京	3	開発部	1
3	東京	3	技術部	2
3	東京	3	総務部	3
3	東京	3	経理部	4
4	青森	1	開発部	1
4	青森	1	技術部	2
4	青森	1	総務部	3
4	青森	1	経理部	4
5	岩手	1	開発部	1
5	岩手	1	技術部	2
5	岩手	1	総務部	3
5	岩手	1	経理部	4
6	山口	999	開発部	1
6	山口	999	技術部	2
6	山口	999	総務部	3
6	山口	999	経理部	4

↓

id	姓	**部署 id**	部署名
1	山田	1	開発部
2	福岡	2	技術部
3	東京	3	総務部
4	青森	1	開発部
5	岩手	1	開発部

その他の JOIN の形式

ほとんど使いませんが、下記のような形式もあります。

- FULL OUTER JOIN: くっついた行 + 左側 + 右側 すべて結果に出す
  (例えばサンプルデータの場合、LEFT のみで出る 1 行、RIGHT のみで出る 1 行を含めた 7 行出てきます)
- CROSS JOIN: 結合条件を指定しないで全て結果に出す
  (例えばサンプルデータの場合社員 6 行×部署 4 行を掛けあわせた 24 行出てきます)

※ MySQL では FULL OUTER JOIN はサポートされていません。

JOIN の形式の違い

JOIN の形式の違いは下記の通りです。

JOIN	くっついた行を出す	くっついていない左側を出す	くっついていない右側を出す
INNER JOIN	○	×	×
LEFT JOIN	○	○	×
RIGHT JOIN	○	×	○
FULL OUTER JOIN	○	○	○
CROSS JOIN	全部の行を掛けあわせる(直積と呼びます)		

問題

※ 今回は簡単です

1. FROM 側をベースにデータを結合するには何の JOIN を使いますか？
2. くっつく条件に当てはまる結果だけ取得するためには何の JOIN を使いますか？
3. FROM A ??? JOIN B で B 側をベースにデータを結合するには何の JOIN を使いますか？
4. 結合条件の前に置く文字は何ですか？

(選択肢)

A. LEFT JOIN
B. INNER JOIN
C. RIGHT JOIN
D. ON
E. WHERE

JOIN は開発するときにはよく使うので覚えておくと効率がよくなります。