16章 2つのテーブルを結合しよう

16章2つのテーブルを結合しよう

SQLで2つのテーブルを結合する方法と、結合の種類について学びます。

本章の目標

- テーブル結合とは何かを知ること
- SQLで複数のテーブルを結合する方法を知ること
- テーブル結合の種類を知ること

16章テーブル結合の必要性

これまではusersテーブルだけを使い、データベース操作を学んできました。しかし、実際のWebアプリケーションでは、複数のテーブルを連携させてデータを管理する場合もあります。

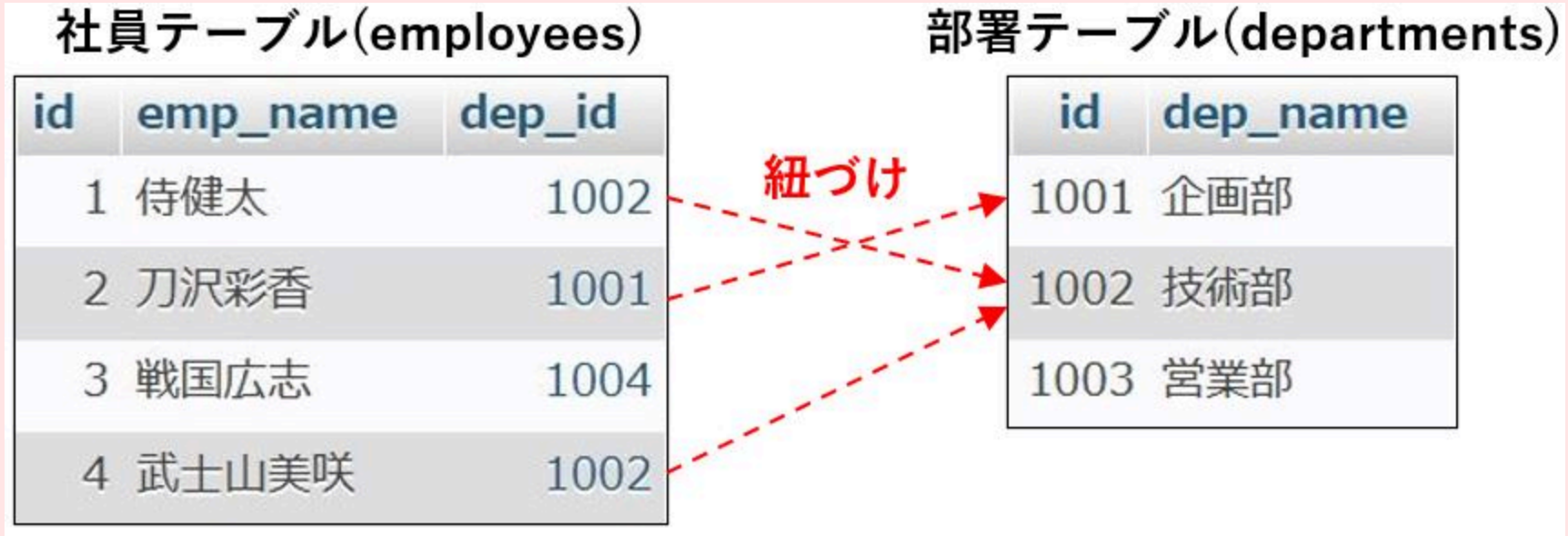
本章では、SQLで2つのテーブルを結合する方法を学びます。

16章テーブル結合とは何かを知ろう

SELECT文と併用して必要なデータを効率的に抽出することができます。

たとえば、社員テーブルemployeesと部署テーブルdepartmentsを考えてみましょう。

各社員はどこかの部署に所属するため、部署IDdep_idで社員と部署を紐づけます。



16章テーブル結合の活用シーン

これまでのSELECT文では、1つのテーブルのデータしか取得しませんでした。

しかし、社員がどの部署に所属しているかを把握したいなら、両テーブルを参照する必要があります。

このようなケースで役立つテクニックが、テーブル結合です。

16章JOIN句でテーブルを結合しよう

SQLでテーブルを結合するときには、SELECT文にJOIN句を組み合わせて使います。

JOIN句は、複数テーブルのデータを結合して取得するためのキーワードです。

ただし、元のテーブルの構造自体を変えるわけではありません。あくまでも、SELECT文でのデータ取得時に、複数テーブルを組み合わせる機能です。

まずは、JOIN句の基本的な使い方を学び、実際に使ってみましょう。

16章JOIN句の使い方

SELECT文でJOIN句を使う場合の最も基本的な書き方は、次のとおりです。

```
SELECT カラム名 FROM テーブルA JOIN テーブルB ON 結合条件;
```

FROM句のテーブルAと、結合したいテーブルBをJOIN句でつなぎ、ON句に結合条件を書きます。

16章ON句の書き方

ON句には、結合したい2テーブルを紐づけるカラムを指定します。

たとえば、「部署テーブルの部署IDid」と「社員テーブルの部署IDdep_id」を紐づける場合、以下のように指定します。

```
ON departments.id = employees.dep_id;
```

複数テーブルの場合は、「テーブル名.カラム名」のようにテーブル名も明記しましょう。
「id」のようにカラム名が同じケースでも区別できるためです。

16章JOIN句を使ってみよう

実際にJOIN句を使ってみましょう。本章では、社員テーブルと部署テーブルを作成して、JOIN句を実践で繰り返し使います。

社員テーブル(employees)

id	emp_name	dep_id
1	侍健太	1002
2	刀沢彩香	1001
3	戦国広志	1004
4	武士山美咲	1002

部署テーブル(departments)

id	dep_name
1001	企画部
1002	技術部
1003	営業部

16章テーブル結合を実行してみよう

では実際に、JOIN句を使ってみましょう。今回は、社員テーブルと部署テーブルを結合し、部署IDが一致したレコードのみを取得してみます。

```
SELECT * FROM employees JOIN departments
ON departments.id = employees.dep_id;
```

16章結合結果の確認

中身を見ると、左側に社員テーブルの3カラム、右側に部署テーブルの2カラムが表示されています。

このように、複数テーブルのカラムを1テーブルとして取得するのが結合です。

社員テーブル

id	emp_name	dep_id
1	侍健太	1002
2	刀沢彩香	1001
4	武士山美咲	1002

部署テーブル

id	dep_name
1002	技術部
1001	企画部
1002	技術部

なお、結合条件を「部署IDが一致するもの」としたため、社員テーブルにしかない「1004」は除外されました。

16章ほかのキーワードを併用する場合

これまでの章ではデータを検索するWHERE句、並び替えるORDER BY句、グループ化するGROUP BY句を学んできました。

JOIN句は、これらのキーワードとの併用も可能です。

併用する場合、JOIN句のテーブル結合が優先されます。

```
SELECT * FROM employees JOIN departments
ON departments.id = employees.dep_id
WHERE departments.dep_name = '技術部';
```

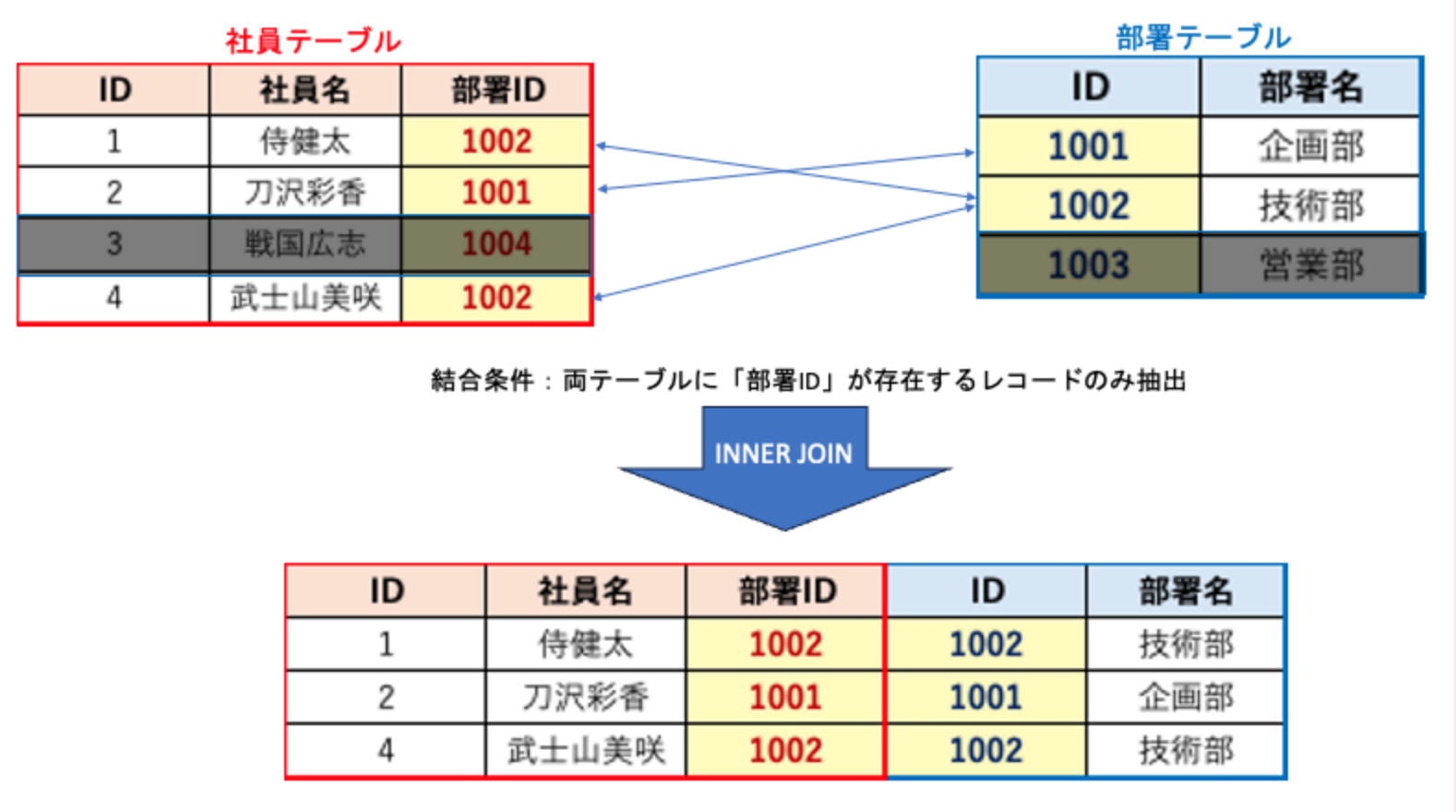
16章JOIN句の種類を知っておこう

JOIN句には5種類あり、それぞれテーブルの結合方法が異なります。

種類	機能
INNER JOIN（内部結合）	両テーブルで合致するレコードのみを結合
LEFT OUTER JOIN（左外部結合）	左テーブル基準で結合
RIGHT OUTER JOIN（右外部結合）	右テーブル基準で結合
FULL OUTER JOIN（完全外部結合）	両テーブルをそれぞれ基準にして結合
CROSS JOIN（交差結合）	両テーブルのレコードを総当たりで結合

16章INNER JOIN（内部結合）

つまり、左テーブルと右テーブルの両テーブルに存在するレコードだけが出力されます。



16章INNER JOINを使ってみよう

下記SQL文を実行すると、社員テーブルにも部署テーブルにも存在するレコードだけが出力されます。

```
SELECT * FROM employees INNER JOIN departments
ON departments.id = employees.dep_id;
```

なお、単に「JOIN」と記述した場合はINNER JOIN扱いです。

内部結合は、2テーブル間の結びつきが強いケースで役に立ちます。

16章OUTER JOIN（外部結合）

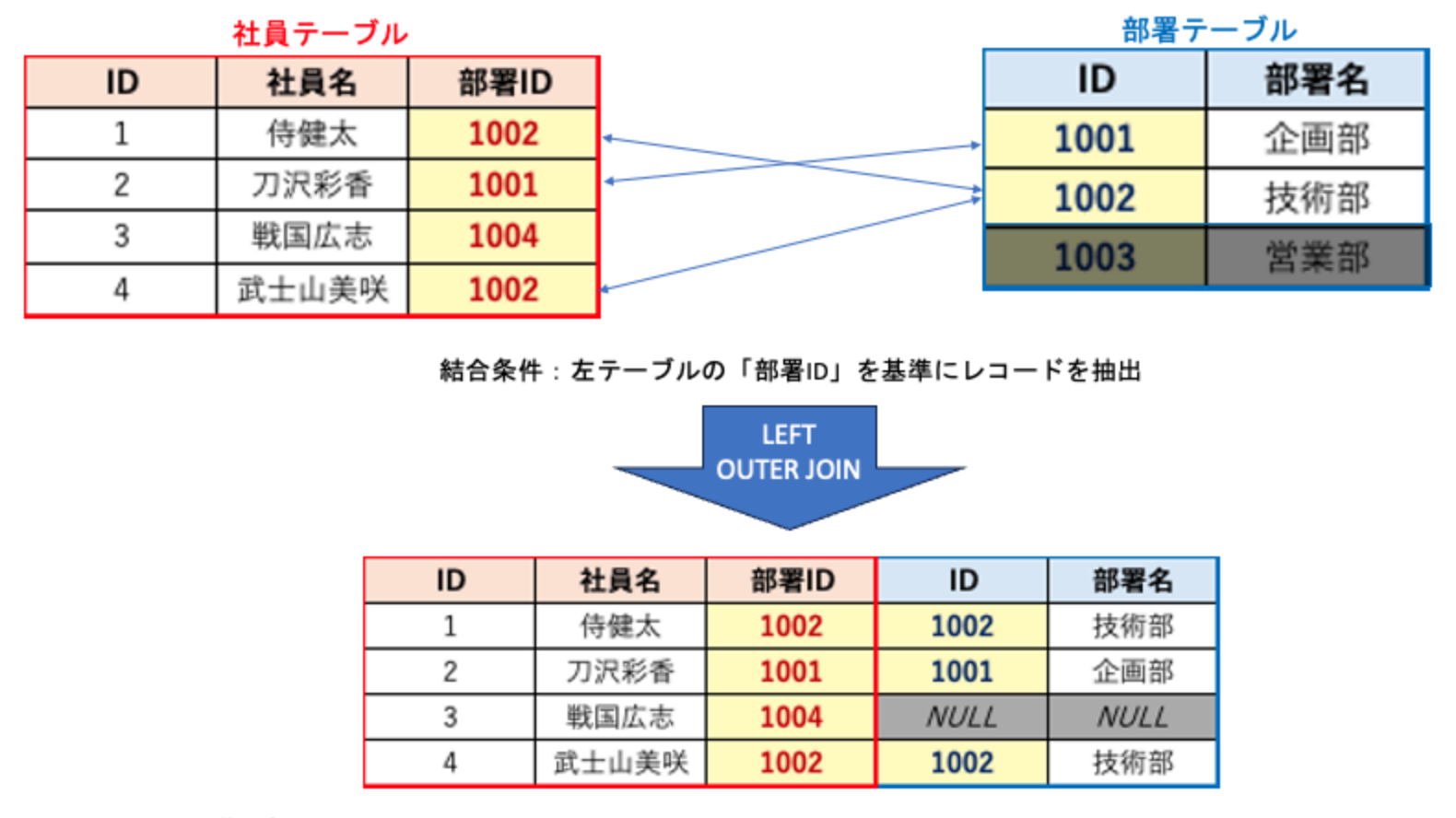
OUTER JOIN（外部結合）は、「INNER JOINの外側」にあるレコードも含めたテーブル結合です。

つまり、片方のテーブルにしかないレコードも出力対象となります。

OUTER JOINは、「どちらのテーブルを基準にするか」によってLEFT・RIGHT・FULLの3種類に分けられます。

16章LEFT OUTER JOIN（左外部結合）

つまり左テーブルの全レコードと、それらに紐づく右テーブルのデータが結合されます。



16章LEFT OUTER JOINを使ってみよう

下記のSQL文において、左テーブルはemployees（社員テーブル）です。

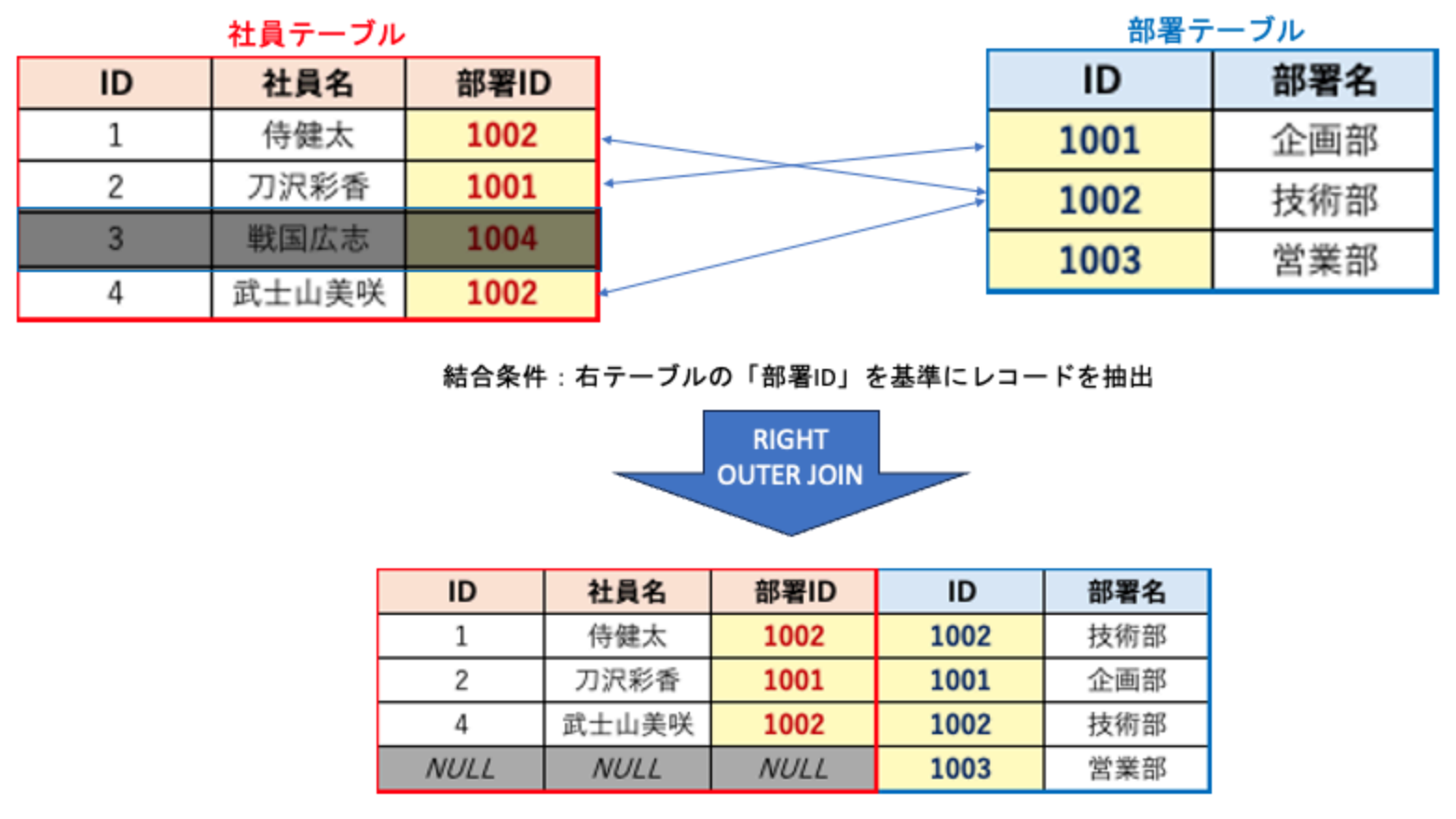
```
SELECT * FROM employees LEFT OUTER JOIN departments
ON departments.id = employees.dep_id;
```

なお、LEFT OUTER JOINは、単にLEFT JOINと書いても構いません。

右テーブル（部署テーブル）に存在しないデータは、「NULL（データなし）」となります。

16章RIGHT OUTER JOIN（右外部結合）

つまり右テーブルの全レコードと、それらに紐づく左テーブルのデータが結合されます。



16章RIGHT OUTER JOINを使ってみよう

下記のSQL文において、右テーブルはdepartments（部署テーブル）です。

```
SELECT * FROM employees RIGHT OUTER JOIN departments
ON departments.id = employees.dep_id;
```

なお、RIGHT OUTER JOINは、単にRIGHT JOINと書いても構いません。

LEFT OUTER JOINとは逆に、左テーブルに存在しないデータがNULLとなります。

16章FULL OUTER JOIN（完全外部結合）

FULL OUTER JOIN（完全外部結合）は、両テーブルを基準にしたテーブル結合です。

つまり左テーブルの全レコードと、右テーブルの全レコードを条件に沿って結合します。

```
SELECT * FROM employees FULL OUTER JOIN departments
ON departments.id = employees.dep_id;
```

ただし、残念ながらMySQLはFULL OUTER JOINに対応していません。

16章FULL OUTER JOINのイメージ

FULL OUTER JOINは、LEFT・RIGHTのOUTER JOINを組み合わせたイメージです。

左テーブルにしかないレコードも、右テーブルにしかないレコードもすべて出力されます。ただし、片方のテーブルに存在しないデータはNULLとなります。

16章FULL OUTER JOINのイメージ

社員テーブル

ID	社員名	部署ID
1	侍健太	1002
2	刀沢彩香	1001
3	戦国広志	1004
4	武士山美咲	1002

部署テーブル

ID	部署名
1001	企画部
1002	技術部
1003	営業部

結合条件：両テーブルの「部署ID」を基準に全レコードを抽出

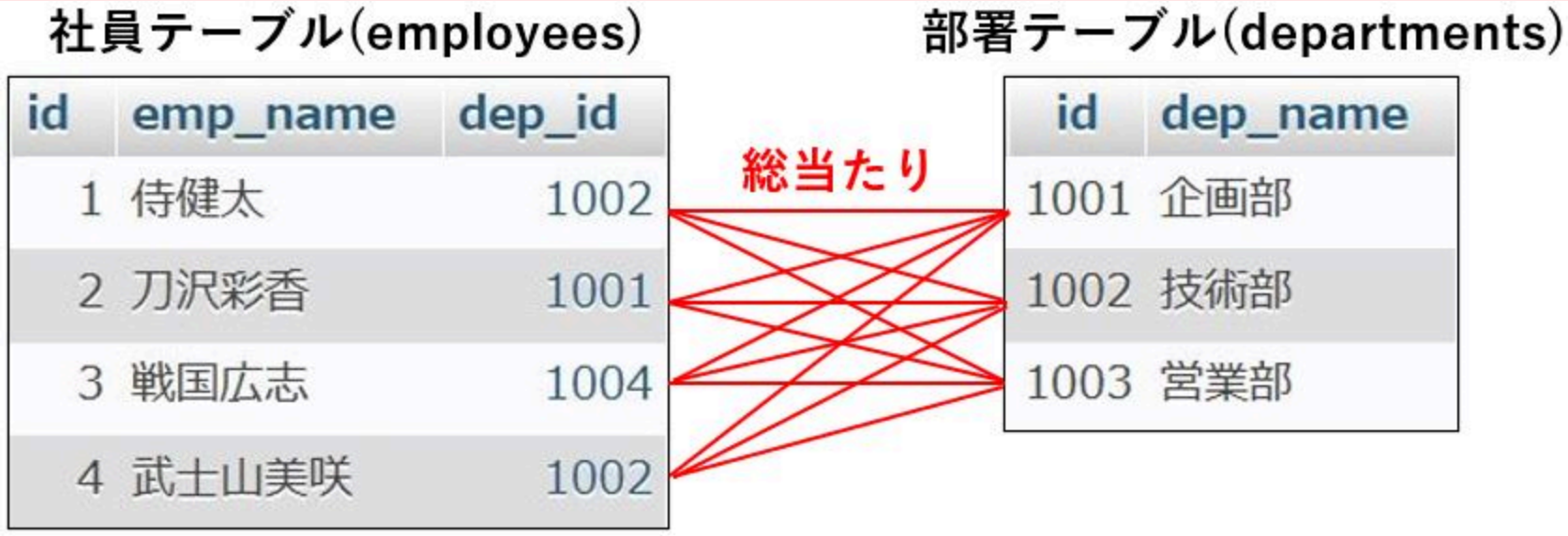
FULL OUTER JOIN

ID	社員名	部署ID	ID	部署名
1	侍健太	1002	1002	技術部
2	刀沢彩香	1001	1001	企画部
3	戦国広志	1004	*NULL*	*NULL*
4	武士山美咲	1002	1002	技術部
NULL	*NULL*	*NULL*	1003	営業部

16章CROSS JOIN（交差結合）

CROSS JOIN（交差結合）は、両テーブルのレコードを総当たりで組み合わせるテーブル結合です。

INNER JOINやOUTER JOINとは、毛色がまったく異なります。



16章CROSS JOINを使ってみよう

以下のように、ON句による結合条件は必要ありません。

```
SELECT * FROM employees CROSS JOIN departments;
```

4人の各社員に対して、企画部・技術部・営業部という部署テーブルのレコードが結合されています。

4レコードと3レコードの総当たりで、出力されるのは4×3=12レコードです。

16章CROSS JOINの注意点

ただし総当たりのため、出力されるレコード数が多くなりやすいことに注意しましょう。

仮に100レコード×100レコードの総当たりだと、10,000レコードも出力されてしまいます。

CROSS JOINは、たとえば「重量」と「地域」で送料が変わる配送システムで役に立ちます。重量テーブルと地域テーブルの組み合わせを網羅して、各送料を一覧表示できます。

16章演習

INNER JOINを使って2つのテーブルを結合し、書籍名とその著者名をリストアップしてください。

書籍テーブル（books）と著者テーブル（authors）を作成し、INNER JOINを使って結合してください。

期待する出力結果

book_id	title	author_name
1	HTML入門	著者A
2	データベース基礎編	著者B
3	データベース応用編	著者B

16章演習の解答例

模範解答は以下のとおりです。

```
SELECT book_id, title, author_name
FROM books INNER JOIN authors
ON books.author_id = authors.author_id;
```

今回は、書籍がどの著者によって書かれたかを表示させるために、bookテーブルのauthor_idとauthorsテーブルのauthor_idを関連付けて結合します。

なお、「著者C」に紐づく書籍はbooksテーブルに存在しないため、表示されません。

16章まとめ（1/3）

- **JOIN句**は、複数テーブルのデータを結合して取得するためのキーワード
  - ON句で結合条件を指定する
- WHERE句やORDER BY句、GROUP BY句よりも、JOIN句のテーブル結合が優先される
- **外部キー**は、ほかのテーブルにあるカラムを参照するカラムのこと
  - テーブル作成（CREATE TABLE）の実行時に外部キーを設定できる

```
SELECT カラム名 FROM テーブルA名 JOIN テーブルB名 ON 結合条件;
```

16章まとめ（2/3）

JOIN句には5種類あり、それぞれテーブル結合方法が異なる

種類	機能
INNER JOIN（内部結合）	両テーブルで合致するレコードのみを結合
LEFT OUTER JOIN（左外部結合）	左テーブル基準で結合
RIGHT OUTER JOIN（右外部結合）	右テーブル基準で結合
FULL OUTER JOIN（完全外部結合）	両テーブルをそれぞれ基準にして結合
CROSS JOIN（交差結合）	両テーブルのレコードを総当たりで結合

16章まとめ（3/3）

```sql
-- 内部結合：両テーブルで合致するレコードのみを結合
SELECT * FROM テーブルA INNER JOIN テーブルB ON 結合条件;

-- 左外部結合：左テーブル基準で結合
SELECT * FROM テーブルA LEFT OUTER JOIN テーブルB ON 結合条件;

-- 右外部結合：右テーブル基準で結合
SELECT * FROM テーブルA RIGHT OUTER JOIN テーブルB ON 結合条件;

-- 交差結合：両テーブルのレコードを総当たりで結合
SELECT * FROM テーブルA CROSS JOIN テーブルB;
```