JOIN, WITH

1日目	2日目	3日目	4日目	5日目	6日目	7日目
8日目	9日目	10日目	11日目	12日目	13日目	14日目
15日目	16日目	17日目	18日目	19日目	20日目	21日目

SQLのコツ

ここからは、テーブルの結合処理があるため、また一段と難しくなります!!
SQLを理解するには、以下のように**1つずつ分解して考える**ことをお勧めします

例）AテーブルとBテーブルと結合して、Aテーブルのnameカラムが**A**で始まるものを取り出し、Bテーブルのageで昇順に並び替え、10行取り出す。

- AテーブルとBテーブルを結合する(**INNER JOIN, LEFT JOIN 等**)
- Aテーブルのnameカラムが**A**で始まるものに絞り込む(**name LIKE “A%”**)
- Bテーブルのageで並び替える(**ORDER BY age**)
- 10行取り出す(**LIMIT 10**)

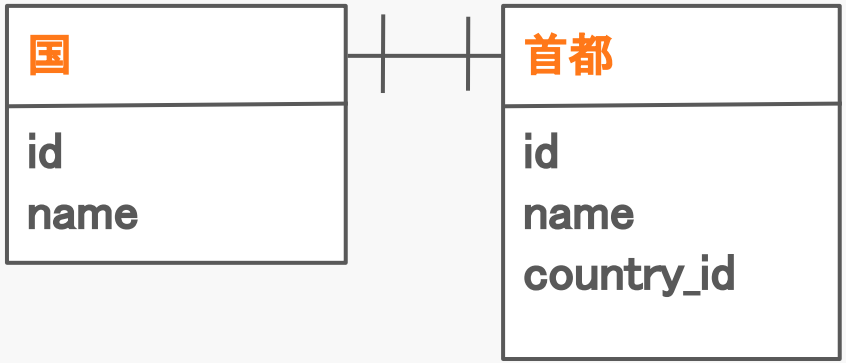
テーブルの結合について

テーブルの結合について

RDBMSでは、特定のカラム（列）の値を用いて、同じ値のレコード同士を紐づけることができる。
これをテーブルの結合という。

1対1の結合

複数のテーブルが、1レコードにつき1レコードが紐づく結合を1対1(One To One)結合という（下の例では、国に対して首都が1レコードに1レコードが紐づく）

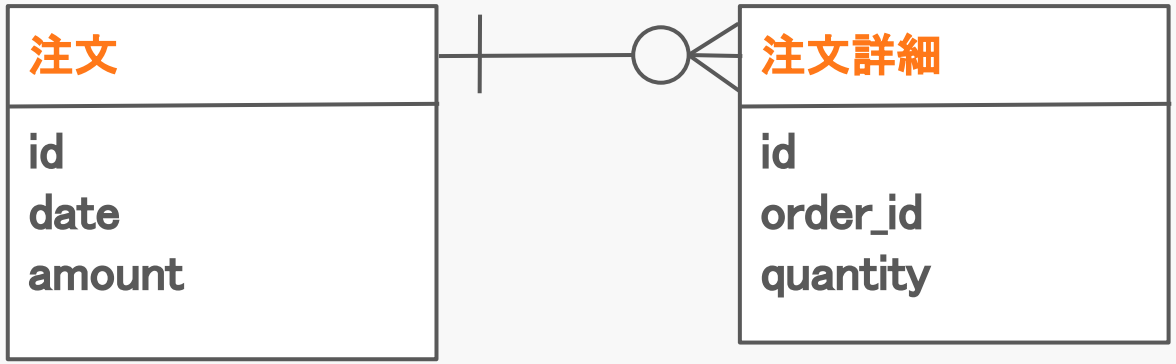


id	name
1	日本
2	中国

id, country_id	name
1	東京
2	北京

1対多の結合

複数のテーブルが、一方のテーブルの1レコードにつきもう片方のテーブルの複数レコードに紐づく結合を1対多(One To Many)結合という(下の例では、注文に対して複数の注文詳細が紐づく)

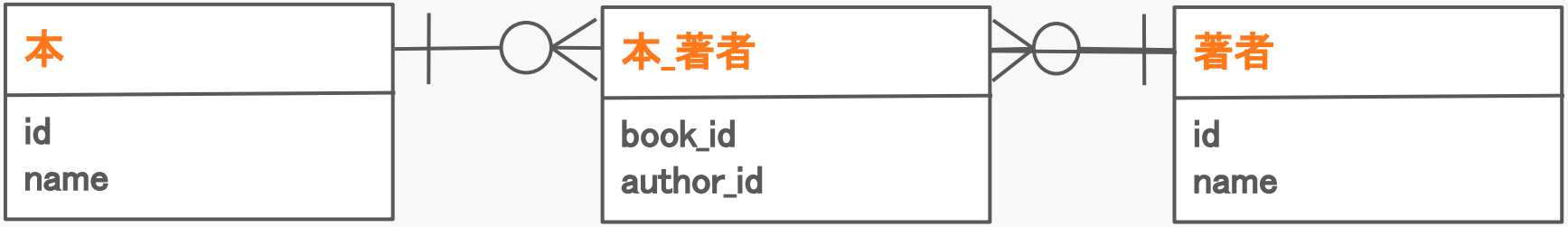


id	date	amount
1	20210101	5

id	order_id	quantity
1	1	1
2	1	2
3	1	3

多対多の結合

複数のテーブルで**各テーブルが、互いのテーブルに対して複数のレコードに紐づく**結合を多対多(Mny To Many)結合という(下の例では、本に対して本_著者が**1対多で紐づき**、本_著者と著者が**多対1で紐づく**。結果、複数の著者に対して複数の本が紐づき、複数の本に対して複数の著者に紐づく。)



id	name
1	緑の本
2	青の本

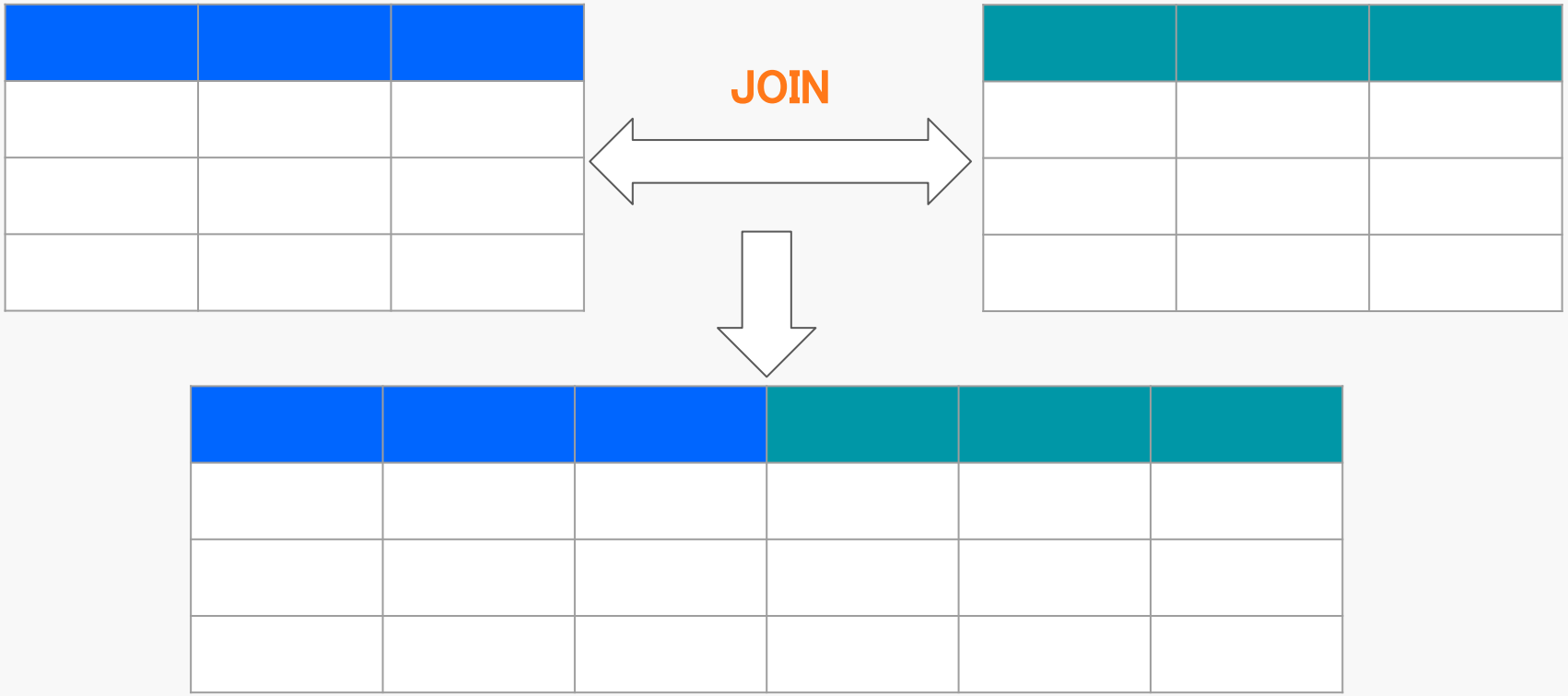
book_id	author_id
1	1
1	2
2	1
2	2

id	name
1	田中義男
2	小原信二

JOIN
(INNER JOIN, OUTER JOIN)

JOINとは

特定のカラム同士が等しいレコード同士を、テーブル間で結合するSQL
（複数のテーブルを列方向に連結するイメージ）



INNER JOIN(JOIN)について

複数のテーブルを**指定した条件が正しいレコードのみ連結**して、取り出すSQL

INNER JOIN 接続するテーブル名 ON 接続する条件

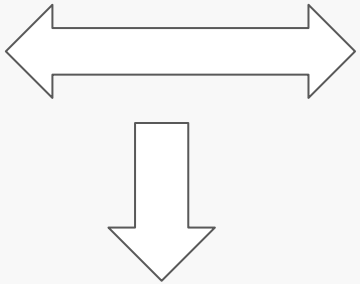
```
SELECT
    or.id,
    ct.customer_name,
    or.order_date
FROM
    orders AS or
    INNER JOIN
        customers AS ct
    ON
    or.customer_id = ct.customer_id;
```

ordersテーブルのid、customersテーブルの
customer_name、ordersテーブルのorder_dateを表示する

ONの直後の条件で、orders(or)のcustomer_idカラムと
customers(ct)のcustomer_idが **等しいもののみ** を連結して取り出す

```
SELECT * FROM orders AS or
    INNER JOIN customers AS ct
    ON or.customer_id = ct.customer_id;
```

order_id	customer_id	name
1	2	Order A
2	3	Order B
3	10	Order C



customer_id	name
2	Customer A
4	Customer B
10	Customer C

order_id	customer_id	name	customer_id	name
1	2	Order A	2	Customer A
3	10	Order C	10	Customer C

LEFT（OUTER）JOINについて

複数のテーブルを結合して、左のテーブルは全てのレコードを取得して、右のテーブルからは**紐づけのできたレコードのみ**を取り出し、それ以外はNULL

```
SELECT
    or.id,
    ct.customer_name,
    or.order_date
FROM
    orders AS or
    LEFT JOIN
        customers AS ct
    ON
    or.customer_id = ct.customer_id;
```

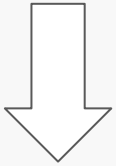
ordersテーブルのid、customersテーブルの
customer_name、ordersテーブルのorder_dateを表示する

左のテーブルorders(or)は **全てのレコードを取得** し、
customers(ct)はor.customer_id=ct.customer_idで紐づけられた
ものだけを取得して **それ以外はNULLとする**

```
SELECT * FROM orders AS or
    LEFT JOIN customers AS ct
    ON or.customer_id = ct.customer_id;
```

order_id	customer_id	name
1	2	Order A
2	3	Order B
3	10	Order C

customer_id	name
2	Customer A
4	Customer B
10	Customer C



order_id	customer_id	name	customer_id	name
1	2	Order A	2	Customer A
2	3	Order B	NULL	NULL
3	10	Order C	10	Customer C

RIGHT（OUTER）JOINについて

複数のテーブルを結合して、右のテーブルは全てのレコードを取得して、左のテーブルからは紐づけのできたレコードのみを取り出し、それ以外はNULL

```
SELECT
    or.id,
    ct.customer_name,
    or.order_date
FROM
    orders AS or
    RIGHT JOIN
        customers AS ct
    ON
    or.customer_id = ct.customer_id;
```

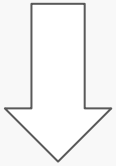
ordersテーブルのid、customersテーブルのcustomer_name、ordersテーブルのorder_dateを表示する

右のテーブルcustomers(cr)は 全てのレコードを取得 し、orders(ct)はor.customer_id=ct.customer_idで紐づけられたものだけを取得して それ以外はNULLとする

```
SELECT * FROM orders AS or
    RIGHT JOIN customers AS ct
    ON or.customer_id = ct.customer_id;
```

order_id	customer_id	name
1	2	Order A
2	3	Order B
3	10	Order C

customer_id	name
2	Customer A
4	Customer B
10	Customer C



order_id	customer_id	name	customer_id	name
1	2	Order A	2	Customer A
NULL	NULL	NULL	4	Customer B
3	10	Order C	10	Customer C

FULL OUTER JOIN*)について

複数のテーブルを結合して、結合できなかった行はNULLと表示する結合方法

*) MySQL(ver. 8.0.28　の時点)は、FULL OUTER JOINを利用できない

```
SELECT
    or.id,
    ct.customer_name,
    or.order_date
FROM
    orders AS or
    FULL OUTER JOIN
        customers AS ct
    ON
    or.customer_id = ct.customer_id;
```

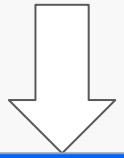
ordersテーブルのid、customersテーブルの
customer_name、ordersテーブルのorder_dateを表示する

両方のテーブルorders(or), customers(ct)から 全てのレコードを取得し、or.customer_id=ct.customer_idで紐づけられなかったものはNULLとする

```
SELECT * FROM orders AS or
    RIGHT JOIN customers AS ct
    ON or.customer_id = ct.customer_id;
```

order_id	customer_id	name
1	2	Order A
2	3	Order B
3	10	Order C

customer_id	name
2	Customer A
4	Customer B
10	Customer C

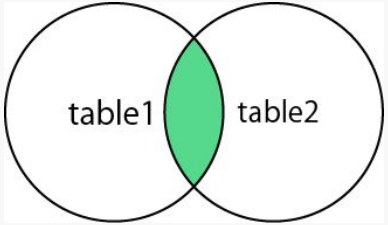


order_id	customer_id	name	customer_id	name
1	2	Order A	2	Customer A
2	3	Order B	NULL	NULL
NULL	NULL	NULL	4	Customer B
3	10	Order C	10	Customer C

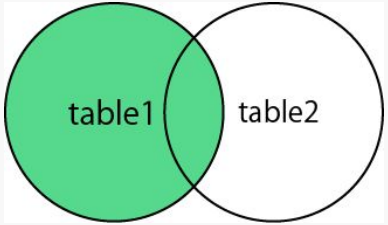
INNER JOIN, LEFT(RIGHT) JOIN, FULL OUTER JOINまとめ

INNER JOIN: 両方のテーブルで紐づいたレコードのみ取得
LEFT JOIN: 左のテーブルからは全レコード、右のテーブルで左に紐づかないものはNULLで取得
RIGHT JOIN: 右のテーブルからは全レコード、左のテーブルで右に紐づかないものはNULLで取得
FULL OUTER JOIN: 左右のテーブルから全レコード、紐づかないものはNULLで取得

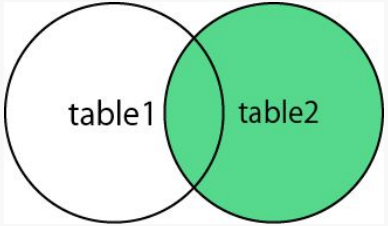
INNER JOIN



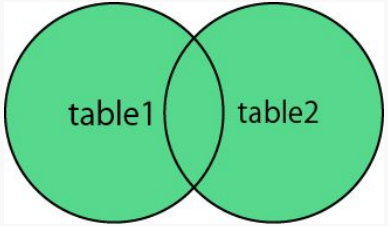
LEFT JOIN



RIGHT JOIN



FULL OUTER JOIN



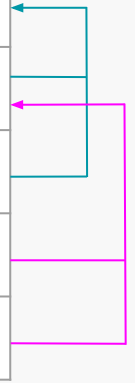
自己結合
(SELF JOIN)

自己結合

同一のテーブルを結合する結合方法

```
SELECT
    e1.firstname, e2.firstname, e1.city
FROM
    employees e1
INNER JOIN
    employees e2  ON e1.manager_id = e2.id
```

id	name	manager_id
1	Taro	NULL
2	Jiro	1
3	Saburo	1
4	Shiro	2
5	Goro	2



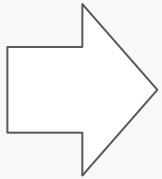
交差結合
(CROSS JOIN)

交差結合(CROSS JOIN)とは

2つのテーブルのデータの全ての組み合わせを取得するSQL

COLOR
Blue
Green

SIZE
Small
Medium
Large



COLOR	SIZE
Blue	Small
Blue	Medium
Blue	Large
Green	Small
Green	Medium
Green	Large

交差結合(CROSS JOIN)の構文

```
SELECT (カラム) FROM　　テーブル1 AS 別名1
CROSS JOIN テーブル2 AS 別名2
(ON 結合条件)
```

古い結合の書き方

```
SELECT (カラム)
FROM　テーブル1 AS 別名1, テーブル2 AS 別名2
(WHERE 条件)
```

WITH

WITHとは*)

WITHの後に記述したSQLの実行結果を、一時的なテーブルに格納する。可読性の高い構文

```
WITH 一時テーブル名1 AS (
    SELECT 〇〇
), 一時テーブル名2 AS (
    SELECT 〇〇
)
SELECT *
FROM テーブル
INNER JOIN 一時テーブル1 ON テーブル.id = 一時テーブル1.id
INNER JOIN 一時テーブル2 ON 一時テーブル1.id =  一時テーブル2.id
```

*) MySQLでは、WITHはversion 8.0以降で利用できるようになった