

# RDB, GROUP BY & JOIN

---

# Contents

---

- webの仕組み(復習)
- RDBの構造
- テーブルの追加
  - ユーザテーブル
  - いいねテーブル
- いいねボタンの追加とdb操作
- いいね数の表示
- 課題発表 -> P2Pタイム

# rules...

---

- 授業中は常にエディタを起動！
- 考えたことや感じたことはslackのガヤチャンネルでガンガン発信！
- 質問はslackへ！ 他の人の質問にも目を通そう！（同じ質問があるかも）
- 演習時，できた人はスクショなどslackに貼ってアウトプット！
- まずは打ち間違いを疑おう！
  - `{'";` など
- 書いたら保存しよう！（よく忘れる！）
  - `command + s`
  - `ctrl + s`

# PHPの準備

---

以下3点ができているか確認しよう！

- XAMPPの起動確認
- <http://localhost/>のアクセス確認
- サンプルフォルダを「htdocs」フォルダに入れる

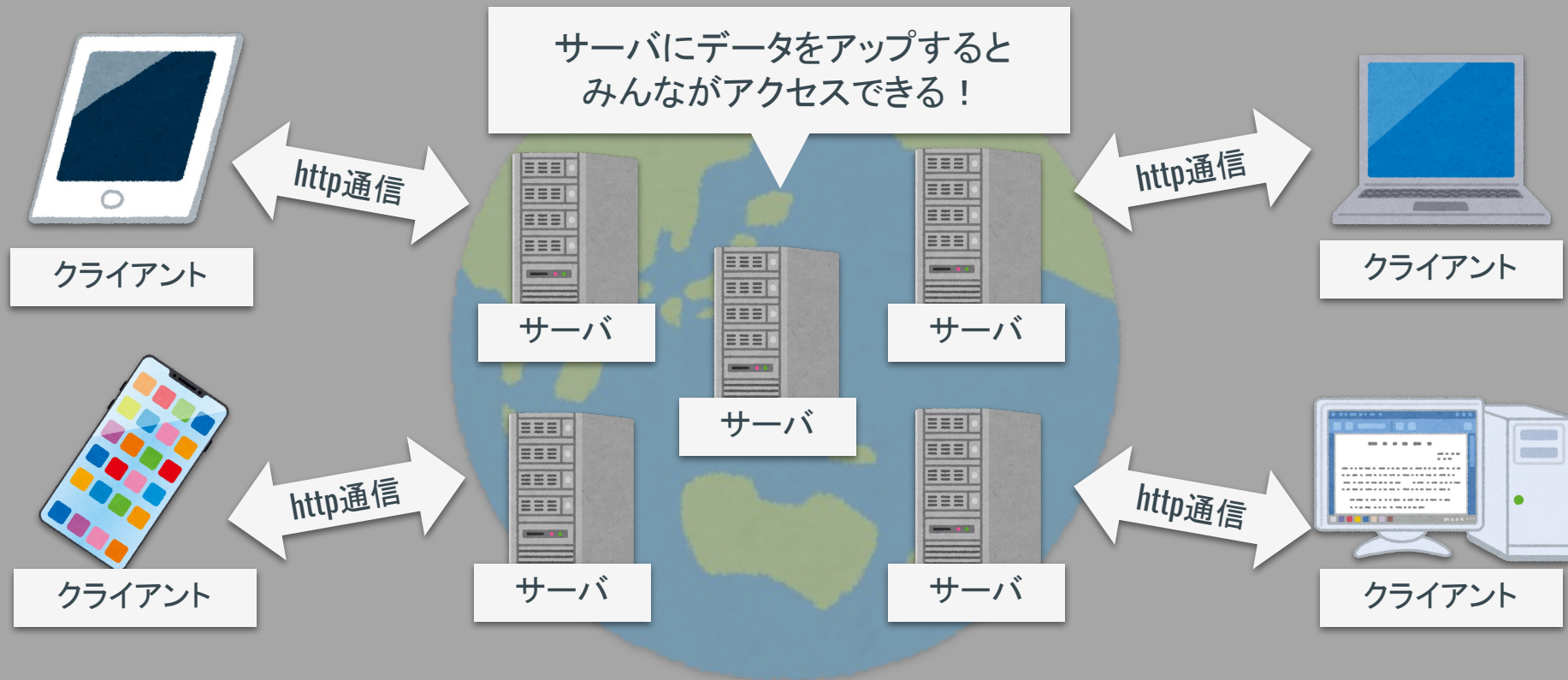
# Goal

---

- RDB構造を把握！
- RDBの考え方を知る！
- 複数テーブルを操作！

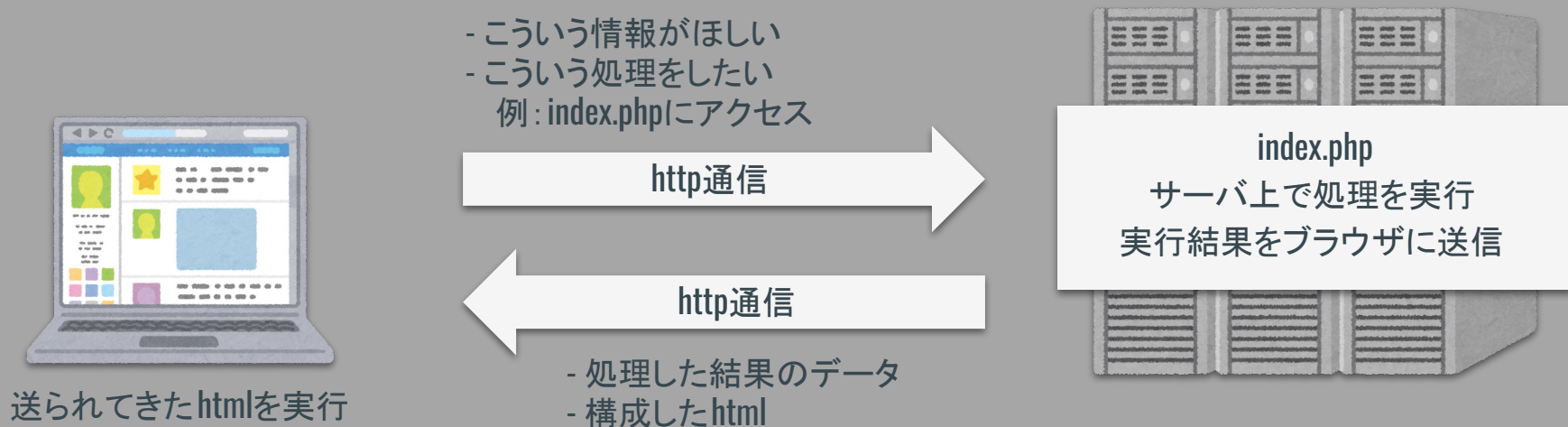
webの仕組み

# 雑なwebの仕組み



# サーバサイド言語の動き方

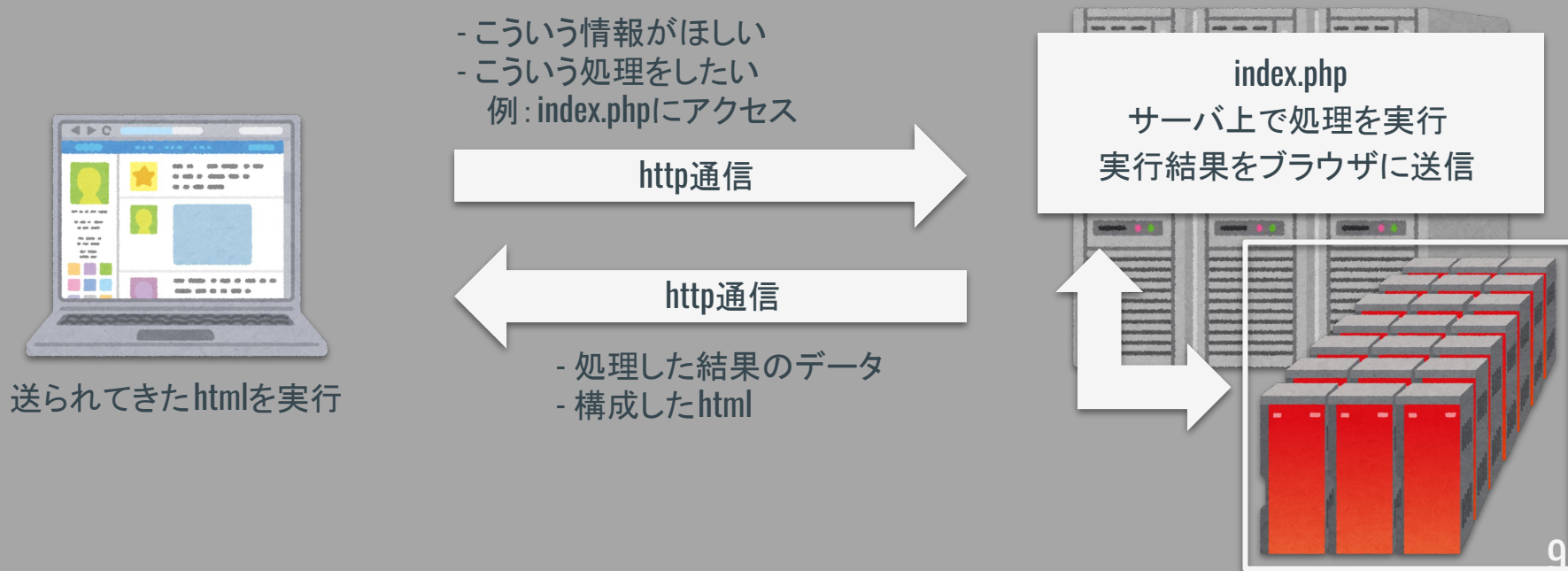
※ 言語によらず, ファイル(プログラム)はサーバ上に存在





# DBの動き方

サーバ上のプログラム(PHPなど)がDBにアクセスして処理を実行！



# RDBの構造

# RDB (Relational Database) とは

---

複雑なデータが必要なとき

- 一つのテーブルから取得できる情報は限られている.
- ECで購入したユーザの情報と商品の情報を両方取得したい場合は. . . ??

複数のテーブル同士を連携(relation)させるッ！！

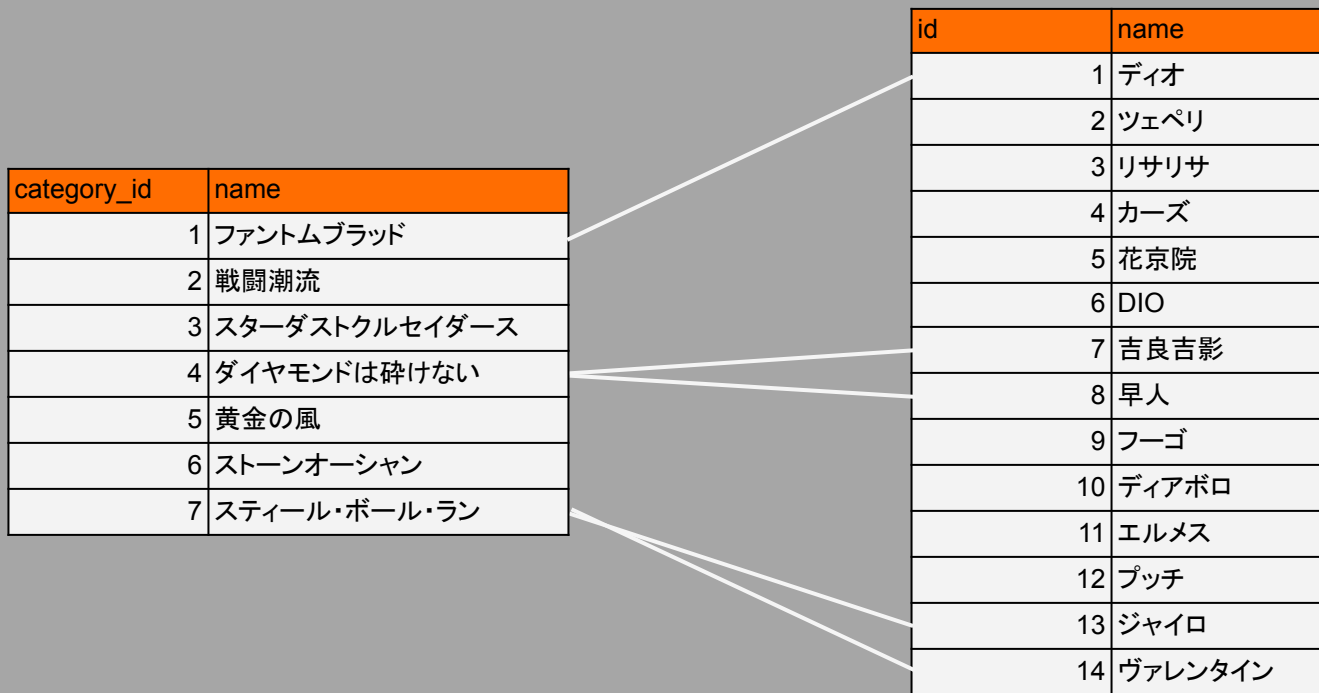
- テーブルの設計段階で連携させたいidのカラムをつくっておく！
- 両方のテーブルを関連付けるテーブルを「SQLで」作成する！
- 大事なのはデータの構造を見極めること！！

# データ構造

One to Many  
Many to Many

# One to Many (カテゴリテーブルとユーザテーブル)

- 一つのカテゴリには複数のユーザが在籍するが、ユーザの所属は一つ。



# One to Many (カテゴリテーブルとユーザテーブル)

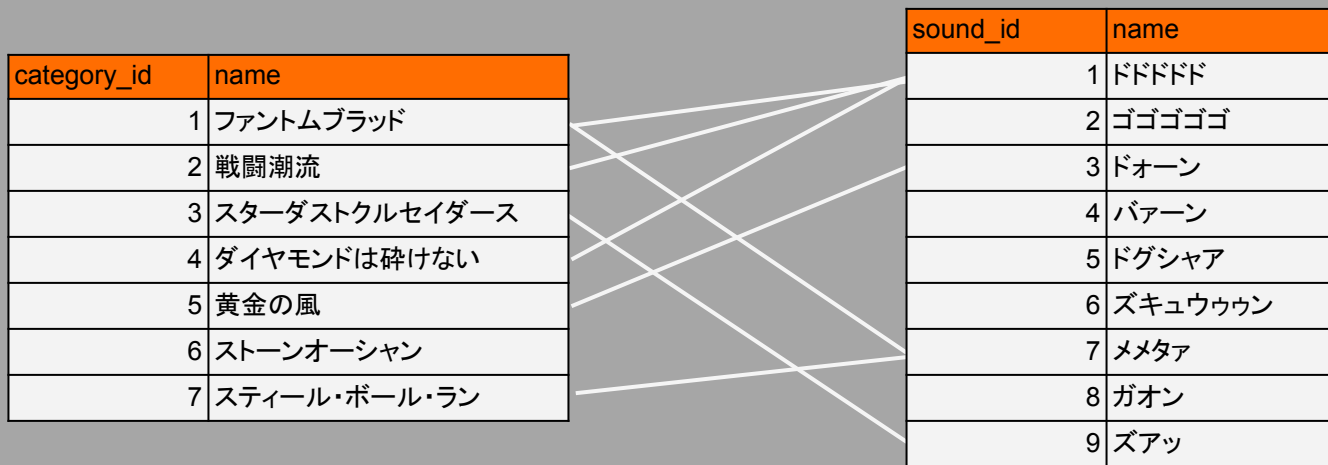
- 「多」の側に対応するidを入れる構造にすればOK

category_id	name
1	ファントムブラッド
2	戦闘潮流
3	スターダストクルセイダース
4	ダイヤモンドは砕けない
5	黄金の風
6	ストーンオーシャン
7	スティール・ボール・ラン

user_id	name	category_id
1	ディオ	1
2	ツェペリ	1
3	リサリサ	2
4	カーズ	2
5	花京院	3
6	DIO	3
7	吉良吉影	4
8	早人	4
9	フーゴ	5
10	ディアボロ	5
11	エルメス	6
12	プッチ	6
13	ジャイロ	7
14	ヴァレンタイン	7

# Many to Many (カテゴリテーブルと擬音語テーブル)

- カテゴリは複数の擬音語が登場し、擬音語も複数のカテゴリで使われる。



# Many to Many (カテゴリテーブルと擬音語テーブル)

category_id	name
1	ファントムブラッド
2	戦闘潮流
3	スターダストクルセイダース
4	ダイヤモンドは砕けない
5	黄金の風
6	ストーンオーシャン
7	スティール・ボール・ラン

sound_id	name
1	ドドドドド
2	ゴゴゴゴゴ
3	ドォーン
4	バァーン
5	ドグシャア
6	ズキュウウン
7	メメタァ
8	ガオン
9	ズアッ

category_id	sound_id
1	1
1	2
1	7
7	7
3	9
1	6

何部に

どの音

このテーブルを「中間テーブル」と呼ぶ

7	5
---	---



# 今回やること

---

前回までに作成したtodoリストにいいね機能(Many to Manyの関係)を追加！

各ユーザは複数のtodoにいいねし, 各todoは複数のユーザからいいねされる.

- ユーザテーブルといいねテーブルを作成
- いいねボタン追加
- いいねボタンクリック時の処理
- いいね数を表示

# テーブルの追加

## 【課題2】ユーザ管理機能の作成

ユーザ管理テーブル(← 必ず作成, DBはこれまでのものを使用)

- テーブル名: `users_table`
- カラム名など

	#	名前	データ型	照合順序	属性	NULL	デフォルト値	コメント	その他	操作
<input type="checkbox"/>	1	<b>id</b> 	int(12)			いいえ	なし		AUTO_INCREMENT	 変更  削除  その他
<input type="checkbox"/>	2	<b>username</b>	varchar(128)	utf8mb4_unicode_ci		いいえ	なし			 変更  削除  その他
<input type="checkbox"/>	3	<b>password</b>	varchar(128)	utf8mb4_unicode_ci		いいえ	なし			 変更  削除  その他
<input type="checkbox"/>	4	<b>is_admin</b>	int(1)			いいえ	なし			 変更  削除  その他
<input type="checkbox"/>	5	<b>is_deleted</b>	int(1)			いいえ	なし			 変更  削除  その他
<input type="checkbox"/>	6	<b>created_at</b>	datetime			いいえ	なし			 変更  削除  その他
<input type="checkbox"/>	7	<b>updated_at</b>	datetime			いいえ	なし			 変更  削除  その他

# 新しくテーブルを作成

## いいねテーブル

- テーブル名: `like_table`
- カラム名など

	#	名前	データ型	照合順序	属性	NULL	デフォルト値	コメント	その他	操作
<input type="checkbox"/>	1	<b>id</b> 	int(12)			いいえ	なし		AUTO_INCREMENT	 変更  削除  その他
<input type="checkbox"/>	2	<b>user_id</b>	int(12)			いいえ	なし			 変更  削除  その他
<input type="checkbox"/>	3	<b>todo_id</b>	int(12)			いいえ	なし			 変更  削除  その他
<input type="checkbox"/>	4	<b>created_at</b>	datetime			いいえ	なし			 変更  削除  その他

# いいねボタンの設置

# やりたいこと

---

いいねボタンをクリックしたら...

- like\_tableに「誰が」「何に」「いいねをしたのか」を追加

実装の方針

- todo\_read.phpにいいねボタンを設置
- GETでtodoのidとユーザのidを送信する
- 受け取り側のファイル(like\_create.php)で受け取ったデータをdbに登録

## likeのリンクをつくる(todo\_read.php)

---

```
// ユーザid取得
$user_id = $_SESSION['id'];

// タグ生成部分

$output .= "<td>
<a href='like_create.php?user_id={$user_id}&todo_id={$record[\"id\"]}'>
like</a>
        </td>";

$output .= "<td>          // 編集ボタン
        <a href='todo_edit.php?id={$record[\"id\"]}'>edit</a>
        </td>";

...
// user_idはログイン時にセッション変数に保存している値を使用している。
```

# いいねボタンの確認(todo\_read.php)

DB連携型todoリスト（一覧画面）

[入力画面](#) [logout](#)

deadline	todo	
2020-06-19	良い食材を買う	<a href="#">like</a> <a href="#">edit</a> <a href="#">delete</a>
2020-06-07	PHP選手権準備	<a href="#">like</a> <a href="#">edit</a> <a href="#">delete</a>
2020-06-12	紅茶を買う	<a href="#">like</a> <a href="#">edit</a> <a href="#">delete</a>
2020-06-05	関数を動かす	<a href="#">like</a> <a href="#">edit</a> <a href="#">delete</a>
2020-06-09	sessionを使う	<a href="#">like</a> <a href="#">edit</a> <a href="#">delete</a>

こんな感じ！！



# いいねボタンクリック の動作

# いいね追加の処理

# いいねデータ受取時の処理 (like\_create.php)

---

- データの受取
- データベース接続
- テーブルへの登録
- 完了したら一覧画面へ移動

## いいねデータ受取時の処理 (like\_create.php)

---

```
// まずはこれ
var_dump($_GET);
exit();

// 関数ファイルの読み込み
include('functions.php');

// GETデータ取得
$user_id = $_GET['user_id'];
$todo_id = $_GET['todo_id'];

// DB接続
$pdo = connect_to_db();
```

## いいねデータ受取時の処理(like\_create.php)

```
$sql = 'INSERT INTO like_table(id, user_id, todo_id, created_at)
      VALUES(NULL, :user_id, :todo_id, sysdate())'; // SQL作成

$stmt = $pdo->prepare($sql);
$stmt->bindValue(':user_id', $user_id, PDO::PARAM_INT);
$stmt->bindValue(':todo_id', $todo_id, PDO::PARAM_INT);
$status = $stmt->execute(); // SQL実行
    exec_query($stmt);
if ($status == false) {
    // エラー処理
} else {
    header('Location:todo_read.php');
}
```

## いいね追加の動作確認

---

いいねボタンをクリックして、データが入っていればOK！

				id	user_id	todo_id	created_at
<input type="checkbox"/>	 編集	 コピー	 削除	1	1	10	2020-06-10 11:20:59

ボタン作成 -> 動作確認まで練習！

いいね済かどうかで条件分岐



# すでにいいねしていればいいねを取り消す

---

このままだと...

- 連打すれば無限にいいねできてしまう...
- いいねしている状況であれば, いいねを取り消す処理にしたい!

実装の方針

- いいねボタンをクリックしたtodoとuserでテーブルを検索
- データが1件以上存在すれば削除のSQLを作成して実行
- データが存在しなければ登録のSQL(前項で作成したもの)を実行

## いいねしているかどうかのチェック(like\_create.php)

```
// いいね状態のチェック (COUNTで件数を取得できる！)
$sql = 'SELECT COUNT(*) FROM like_table
        WHERE user_id=:user_id AND todo_id=:todo_id';
$stmt = $pdo->prepare($sql);
$stmt->bindValue(':user_id', $user_id, PDO::PARAM_INT);
$stmt->bindValue(':todo_id', $todo_id, PDO::PARAM_INT);
$status = $stmt->execute();
if ($status == false) {
    // エラ 処理
} else { exec_query($stmt);
    $like_count = $stmt->fetch();
    var_dump($like_count[0]);           // データの件数を確認しよう！
    exit();
}
```

## いいね有無で条件分岐(like\_create.php)

---

```
// いいねしていれば削除, していなければ追加のSQLを作成
if ($like_count[0] != 0) {
    $sql = 'DELETE FROM like_table
           WHERE user_id=:user_id AND todo_id=:todo_id';
} else {
    $sql = 'INSERT INTO like_table(id, user_id, todo_id, created_at)
           VALUES(NULL, :user_id, :todo_id, sysdate())';
}

// INSERTのSQLは前項で使ったものと同じ！
// 以降（SQL実行部分と一覧画面への移動）は変更なし！
// SQL文は1行にまとめる
```

## いいね削除の動作確認 (like\_create.php)

<div>←T→</div>						id	user_id	todo_id	created_at	
<input type="checkbox"/>		編集		コピー		削除	1	1	10	2020-06-10 11:34:45
<input type="checkbox"/>		編集		コピー		削除	2	1	16	2020-06-10 11:34:54



いいねしたあとでもう一度クリックし、  
データが削除されていればOK！

<div>←T→</div>					id	user_id	todo_id	created_at
<div><div><div></div><div></div></div></div>	<div><div><div></div><div></div></div></div>	編集	<div><div><div></div><div></div></div></div>	コピー	2	1	16	2020-06-10 11:34:54

動作確認までやってみよう！

# いいね数の表示

# 一覧画面のいいねボタン横にいいね数を表示

---

一覧画面で何件いいねされているかがわからない...

- いいねしたらいいねの件数が表示されるように！

実装の方針

- todoごとのいいね数を集計
- 取得したいいね数をtodoのテーブルに結合させる
- いいね数をいいねボタンに表示する

各todoのいいね数を集計



# 【解説】GROUP BY

---

GROUP BYを使うと集計ができる！

- 例 (phpmyadminで実行しよう！)
- `SELECT todo_id, COUNT(id) AS cnt FROM like_table GROUP BY todo_id;`
- 「like\_table」の「todo\_id」ごとに「idの数」を「cnt」というカラム名で表示

## 注意点

- `COUNT()`で件数を取得しているが、集計関数以外だとエラーになる
- その他の集計関数
  - `MIN()`, `MAX()`, `SUM()`, ...など

# いいね数の集計を試みる

---

まずは記述したSQLをphpmyadminで実行しよう！

todo_id	cnt
10	1
15	1
16	1

「どのタスクに」「何件」  
いいねがついているかが確認できる！

【Point】このSQL文は集計結果のテーブルを返す！！

# テーブルの結合

# テーブルを結合して情報をまとめる

---

集計はできたので、いいねボタンに件数を表示したい！

- 画面の表示はtodo\_tableからデータなので、集計結果を組み込めない...
- テーブルを結合させることで、集計結果もまとめて表示できる！

todo\_read.phpのSQL部分を変更する

- SQLの方針
  - 「todo\_table」と「集計結果のテーブル」をつなげる
  - 「todo\_tableのid」と「集計結果のテーブルのtodo\_id」を対応させる
  - 「集計結果のテーブル」は前項で取得したアレ

# 結合前

## todo\_tableとGROUP BYしたテーブル

←T→			id	todo	deadline	created_at	updated_at
<input type="checkbox"/>		編集		コピー		削除	4  良い食材を買う 2020-06-19 2020-06-04 11:47:54 2020-06-05 12:01:56
<input type="checkbox"/>		編集		コピー		削除	7  PHP選手権準備 2020-06-07 2020-06-04 11:48:37 2020-06-04 11:48:37
<input type="checkbox"/>		編集		コピー		削除	10  紅茶を買う 2020-06-12 2020-06-04 11:49:20 2020-06-04 11:49:20
<input type="checkbox"/>		編集		コピー		削除	15  関数を動かす 2020-06-05 2020-06-05 10:54:39 2020-06-05 10:54:39
<input type="checkbox"/>		編集		コピー		削除	16  sessionを使う 2020-06-09 2020-06-09 10:48:12 2020-06-09 10:48:12

todo_id	cnt
10	1
15	1
16	1

## 【解説】JOIN

---

JOINを使うとテーブルの結合ができる！

- 例
- `SELECT * FROM todo_table LEFT OUTER JOIN result_table ON todo_table.id = result_table.todo_id`
- 「todo\_table」と「result\_table」を結合する
- 「ON」の後でどのカラムを対応させるのかを決定する
- 今回は「todo\_tableのid」と「result\_tableのtodo\_id」が対応

# 【解説】OUTER JOINとINNER JOIN

OUTER JOINは対応するものがない場合NULLで補完される

- 例(上で実行したものと同じ)
- `SELECT * FROM todo_table LEFT OUTER JOIN result_table ON todo_table.id = result_table.todo_id`

id	todo	deadline	created_at	updated_at	todo_id	cnt
4	良い食材を買う	2020-06-19	2020-06-04 11:47:54	2020-06-05 12:01:56	NULL	NULL
7	PHP選手権準備	2020-06-07	2020-06-04 11:48:37	2020-06-04 11:48:37	NULL	NULL
10	紅茶を買う	2020-06-12	2020-06-04 11:49:20	2020-06-04 11:49:20	10	1
15	関数を動かす	2020-06-05	2020-06-05 10:54:39	2020-06-05 10:54:39	15	1
16	sessionを使う	2020-06-09	2020-06-09 10:48:12	2020-06-09 10:48:12	16	1

# 【解説】OUTER JOINとINNER JOIN

---

INNER JOINは対応するものがない場合は削除される

- 例
- `SELECT * FROM todo_table INNER JOIN result_table ON todo_table.id = result_table.todo_id`

id	todo	deadline	created_at	updated_at	todo_id	cnt
10	紅茶を買う	2020-06-12	2020-06-04 11:49:20	2020-06-04 11:49:20	10	1
15	関数を動かす	2020-06-05	2020-06-05 10:54:39	2020-06-05 10:54:39	15	1
16	sessionを使う	2020-06-09	2020-06-09 10:48:12	2020-06-09 10:48:12	16	1



## テーブルを結合して一覧表示(todo\_read.php)

```
//データ表示SQL作成
// $sql = 'SELECT * FROM todo_table';      // <- select文を変更
$sql = 'SELECT * FROM todo_table
        LEFT OUTER JOIN (SELECT todo_id, COUNT(id) AS cnt
                          FROM like_table GROUP BY todo_id) AS likes
        ON todo_table.id = likes.todo_id';
$stmt = $pdo->prepare($sql);                // 変更なし
$status = $stmt->execute();                // 変更なし
```

JOINを使って結合！

## まずはSQL文をphpmyadminで実行！

---

id	todo	deadline	created_at	updated_at	todo_id	cnt
4	良い食材を買う	2020-06-19	2020-06-04 11:47:54	2020-06-05 12:01:56	NULL	NULL
7	PHP選手権準備	2020-06-07	2020-06-04 11:48:37	2020-06-04 11:48:37	NULL	NULL
10	紅茶を買う	2020-06-12	2020-06-04 11:49:20	2020-06-04 11:49:20	10	1
15	関数を動かす	2020-06-05	2020-06-05 10:54:39	2020-06-05 10:54:39	15	1
16	sessionを使う	2020-06-09	2020-06-09 10:48:12	2020-06-09 10:48:12	16	1

「todo\_table」と「集計結果」が結合される！！

## 集計結果を表示(todo\_read.php)

---

// いいねボタン

```
<a href='like_create.php?user_id={$user_id}&todo_id={$record["id"]}'>
  like{$record["cnt"]}      // cntカラムの数値（いいね数）を追加
</a>
```

テーブルを結合したので、  
カラム名「cnt」を指定していいね件数を取れる！！

# いいねボタンの表示確認

クリックすると1増える or 1減る

DB連携型todoリスト（一覧画面）

[入力画面](#) [logout](#)

**deadline**

**todo**

2020-06-19 良い食材を買う [like](#) [edit](#) [delete](#)

2020-06-07 PHP選手権準備 [like](#) [edit](#) [delete](#)

2020-06-12 紅茶を買う [like1](#) [edit](#) [delete](#)

2020-06-05 関数を動かす [like1](#) [edit](#) [delete](#)

2020-06-09 sessionを使う [like1](#) [edit](#) [delete](#)

# 課題

ひきつづきプロトタイプ！  
テーブル設計を見直そう！

締切は厳守！！

# P2Pタイム

まずはチーム内で解決を目指す！

訊かれた人は苦し紛れでも応える！！