# **SNSシステム開発 - タイムライン機能その２**

前回、投稿機能が完成しました。**タイムライン**の開発のつづきで、表示機能を実装していきます。

## **概要**

このページでは、タイムラインの作成を題材に以下について学習していきます。

1. 投稿の表示機能

## **投稿表示**

投稿表示の要件は以下でした。

* ユーザーは投稿の全てを見ることができる
  + 各投稿には投稿したユーザ名と画像が表示されている

投稿を表示するために必要なデータをピックアップするとこの通りです。

* 投稿ユーザ名
* 投稿ユーザプロフィール画像
* 投稿内容
* 投稿日時

これらの情報は別々のテーブルに保存されていましたね。

|  |  |
| --- | --- |
| 投稿ユーザ名 | usersテーブル |
| 投稿ユーザプロフィール画像 | usersテーブル |
| 投稿内容 | feedsテーブル |
| 投稿日時 | feedsテーブル |

複数のテーブルから一度に情報を取り出ししたい場合はテーブル結合を使います。

# **テーブル結合**

テーブル結合とは、「外部キー」と「主キー」を結合条件として、複数テーブルから一気にデータを取得することいいます。

つまり、テーブル結合を利用すれば、今回の場合、feedsテーブルの中には、user\_idを保存しているので、複数のテーブルを合体して表示させることができます。

### **どんな時に使うか**

別のテーブルにある情報を紐づけたい時に使います。

今回のような投稿内容とユーザ情報を紐づけるときや、購入システムで、ユーザが購入した情報と商品単価を掛け合わせて合計を出す時、にも使えます。

### **結合 JOIN**

結合には内部結合、外部結合の２種類があります。

外部結合は、

SELECT \* FROM `テーブル1` **LEFT (OUTER) JOIN `テーブル2` ON 条件**

SELECT \* FROM `テーブル1` **RIGHT (OUTER) JOIN `テーブル2` ON 条件**

内部結合は、

SELECT \* FROM `テーブル1` (**INNER) JOIN `テーブル2` ON 条件**

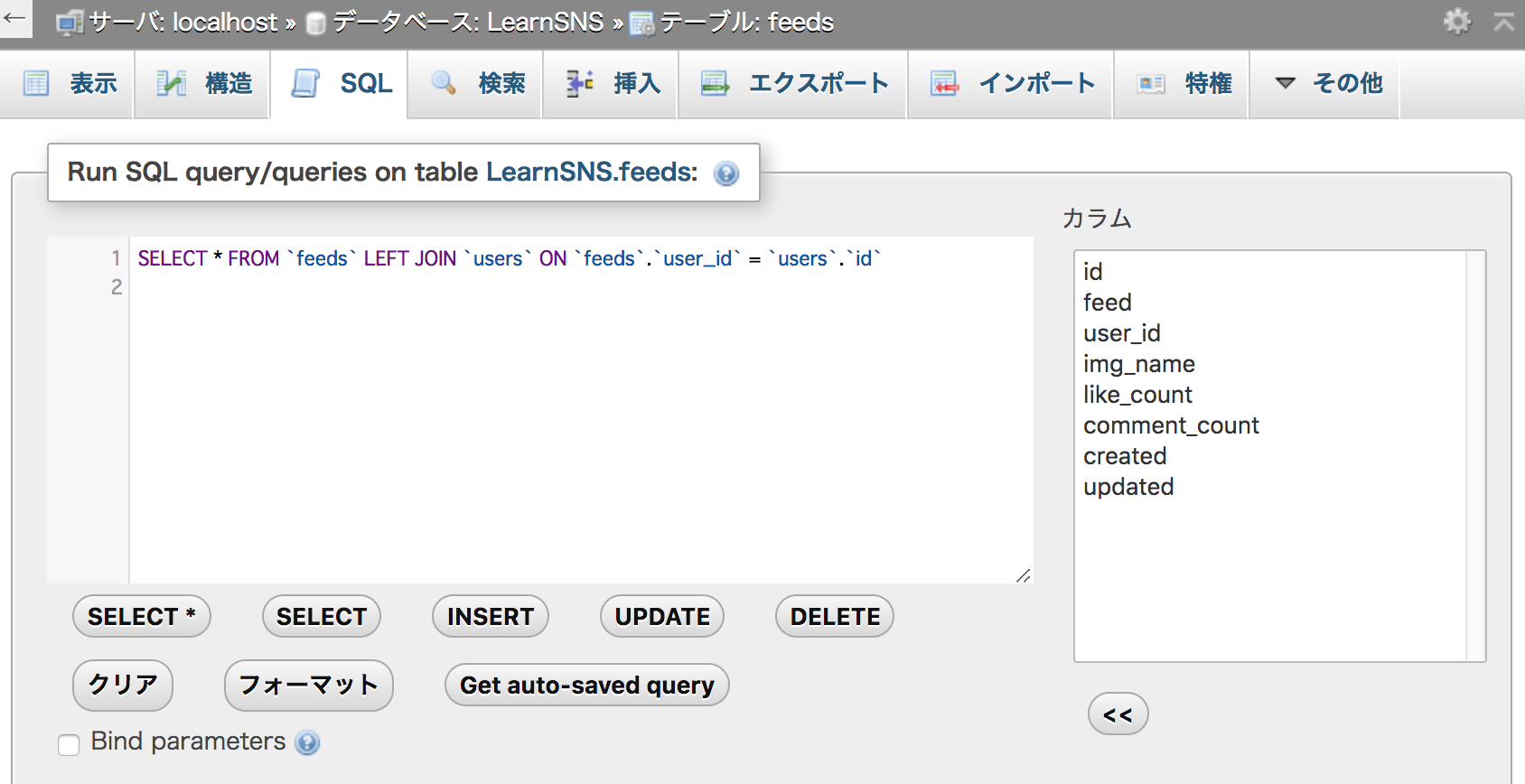
と書きます。

INNERとOUTERは省略できます。

LEFT JOINは、左のテーブル１のレコードを全て表示し、右のテーブル２のレコードを紐付けします。右のテーブル２に一致するレコードがなかった場合、右のテーブルの中身はNULLとなります。RIGHT JOINはその逆です。それに対して、INNER JOINは、２つのテーブルから条件が成立するレコードのみが取り出せます。

phpMyAdminで以下のSQL文を入力してみましょう。

* [SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) \* FROM `feeds` LEFT JOIN `users` ON `feeds`.`user\_id` = `users`.`id`



feedsテーブルの右側に、user\_idが一致するユーザ情報が取れていることが確認できます。

ONの後ろにはJOINする条件が入りますが、一意に決まる条件を入れる必要があります。どれを結合するのかデータベースにはわからないためです。裏を返すと、結合したい情報が別のテーブルにある場合、あらかじめJOINするためのキーを作っておく必要があります。

**feedsテーブル定義書（抜粋）**

|  |  |  |
| --- | --- | --- |
| **論理名** | **物理名** | **型** |
| 投稿ID | id | INT |
| 投稿内容 | feed | TEXT |
| 投稿したユーザID | user\_id | INT |
| 投稿した日時 | created | DATETIME |

今回は、投稿したユーザIDをfeedsテーブルに定義してあるので、usersテーブルのIDとfeedsテーブルの投稿したユーザIDで紐づけることができます。

###### テーブル定義は今回のSNS作成ではもともとあったものを使っていますが、自分で設計する際は、データベースの**正規化**をする必要があります。気になる人はググってみましょう。

**補足**

以下に、JOIN句を書く際に知っておくと有用なSQLの書き方を紹介します。

今回のタイムラインでは、以下全て使用して実装します。

### **複数のテーブルがある時のカラム名の書き方**

SELECT \* FROM `feeds` LEFT JOIN `users` ON `feeds`.`user\_id` = `users`.`id`

2つのテーブルが登場しているときは、`テーブル名`**.**`カラム名`のようにピリオド(.)を使って、テーブル名とカラム名をつなぎます。

### **カラムの一部読み出し**

SELECT `feeds`.\*, `users`.`name`, `users`.`img\_name` FROM `feeds` LEFT JOIN `users` ON `f`.`user\_id`=`u`.`id`

全てのカラムは必要ないので、限定して取り出すこともできます。

今回は、feedsテーブル全てと、usersテーブルの一部を読み出します。

### **テーブルのリネーム**

SELECT `f`.\*, `u`.`name`, `u`.`img\_name` FROM `feeds` AS `f` LEFT JOIN `users` AS `u` ON `f`.`user\_id`=`u`.`id`

何度も登場するテーブル名はASを使ってリネームすることで、SQL文を短く書くことができます。

### **順番の指定 ORDER BY**

SELECT `f`.\*, `u`.`name`, `u`.`img\_name` FROM `feeds` AS `f` LEFT JOIN `users` AS `u` ON `f`.`user\_id`=`u`.`id` ORDER BY `f`.`created` DESC

ORDER BY 句を使うと、特定のカラムによって並べ替えができます。ここでは、DESCを使って、`created`カラムを降順に並べる指定をしています。

DESCを指定しないと昇順（ASC）となります。

## **投稿表示実装**

それでは、投稿表示機能を実装します。

### **1.実装設計**

はじめに、要件を実現するための設計を行います。

前処理のPHPではRead処理を行います。

HTMLの表示箇所では、投稿の数だけ表示されるように繰り返し文を実装します。

また、表示させたい名前、プロフィール画像、投稿内容、投稿日時が、各投稿によって変わるようにPHPを実装します。

手順

1. 投稿情報を全て取得する(SQL)
2. 投稿数だけ繰り返す(繰り返し文)
3. 投稿情報に基づいて表示内容を変える

### **2.PHP実装**

では、SELECT文にLEFT JOINを用いて、必要なレコードを取り出してみましょう。

#### **▼練習問題1**

**投稿の表示**に必要な**Read処理**を実装してください。

投稿の表示には配列変数**$feeds**を使います。$feedsにテーブル結合したレコードが全て入るように実装してください。

**※すぐ下に解答プログラムを記載していますが、今までのプログラムを参考に自分の力で実装しましょう。どう確認したら、実装できていることが確認できるかも合わせて考えましょう。**

#### 

#### **▼練習問題1の解答プログラム**

if (!empty($\_POST)) {

$feed = $\_POST['feed'];

if( $feed != '' ) {

// 投稿処理

}

}

// LEFT JOINで全件取得

$sql = 'SELECT `f`.\*, `u`.`name`, `u`.`img\_name` FROM `feeds` AS `f` LEFT JOIN `users` AS `u` ON `f`.`user\_id`=`u`.`id` ORDER BY `f`.`created` DESC';

$stmt = $dbh->prepare($sql);

$stmt->execute();

// 表示用の配列を初期化

$feeds = [];

while (true) {

$record = $stmt->fetch(PDO::FETCH\_ASSOC);

if ($record == false) {

break;

}

$feeds[] = $record;

}

?>

SQLは前項で学んだ通り、JOIN句を用いてfeedsテーブルにusersテーブルの情報を紐付けて全件取得できるようにします。「誰が投稿したのか」が結合条件になるので、feeds.user\_idとusers.idを利用しましょう。

取得しているカラムにも注意しましょう。今回は投稿情報を表示する箇所であるため、ユーザーのパスワードなどの情報は不要です。そのため、ここでは、投稿情報の全て（feeds.\*）とユーザーの名前と画像（users.name, users.img\_name）のみを取得しています。

while文の条件は、「投稿情報が取れなくなるまで」なので、trueに設定してあります。

投稿情報が取れなくなった時点、つまり、$recordがfalseになった時点でbreakするようにしましょう。

fetchはREAD処理で取得できた表の行を1行ずつ取得する処理になります。while文と合わさり、1行取得して$feedsに追加、1行取得して$feedsに追加・・・と繰り返されることになります。

なお、この処理はPOST送信があってもなくても実行される場所に書くことがポイントです。

確認は**var\_dump**で**$feedsを表示**させて、ちゃんとデータが入っているか検証しましょう。

１つ処理を書いたら想像通りになっているか確認しましょう。

細かく確認しながら進めることで、結果的に時間が無駄になりません。

### **3.表示部分のPHP実装**

前項で、$feedsに全ての投稿データが保存されているので、$feedsを使って表示します。

#### 

#### **▼練習問題2**

**HTML内**に**投稿内容、投稿日時、ユーザ名、ユーザプロフィール画像を表示するPHP**を実装してください。

投稿の表示には前項で実装した配列変数**$feeds**を使います。

**※すぐ下に解答プログラムを記載していますが、自分の力で実装してみましょう。**

#### **▼練習問題2の解答プログラム**

….

<input type="submit" value="投稿する" class="btn btn-primary">

</form>

</div>

**<?php foreach($feeds as $feed): ?>**

<div class="thumbnail">

<div class="row">

<div class="col-xs-1">

<img src="**user\_profile\_img/<?php echo $feed['img\_name']; ?>**" width="40">

</div>

<div class="col-xs-11">

**<?php echo $feed['name']; ?>**<br>

<a href="#" style="color: #7F7F7F;">**<?php echo $feed['created']; ?>**</a>

</div>

</div>

<div class="row feed\_content">

<div class="col-xs-12" >

<span style="font-size: 24px;">**<?php echo $feed['feed']; ?>**</span>

</div>

</div>

<div class="row feed\_sub">

<div class="col-xs-12">

<form method="POST" action="" style="display: inline;">

<input type="hidden" name="feed\_id" >

<input type="hidden" name="like" value="like">

<button type="submit" class="btn btn-default btn-xs"><i class="fa fa-thumbs-up" aria-hidden="true"></i>いいね！</button>

</form>

<span class="like\_count">いいね数 : 100</span>

<span class="comment\_count">コメント数 : 9</span>

<a href="#" class="btn btn-success btn-xs">編集</a>

<a href="#" class="btn btn-danger btn-xs">削除</a>

</div>

</div>

</div>

**<?php endforeach; ?>**

<div aria-label="Page navigation">

….

配列の個数回繰り返したい時は、foreachを使うと便利です。

４つの情報が、表示され、投稿するたびに増えて行けばOKです。

もし期待していた表示と違っていたら、前処理のPHPから見直してみましょう。

これで、表示機能は完成です。