モバイル
プログラミング2

M.Katsube

2016/12/05
日本工学院八王子専門学校

教材をDLしてください

- data.zip
  https://git.io/v1Wo2

※短縮URLのアクセス先は以下
https://github.com/katsube/neec/tree/master/mobileprogramming2/20161205

本日の予定

午前

- MySQL基礎
  - 正規化
  - 複数テーブルを利用したSQL
- レポートの案内

午後

- PHP演習
  - ガチャ開発 (MySQL版 その2)

本日もペアプロです！

前回休んだ人

(゜∀゜)ノシ

PC借りた人

(°∀°)ﾉｼ

まずは追いつこう

1. GitHubの資料見てね
   http://github.com/katsube/neec

2. 環境構築

3. 環境構築で困ったらすぐに聞いてください

アンケート
（出席カード）

アンケート

1. 提出＝出席 (授業終了までに限る)
   未提出＝**欠席**
2. 学籍番号、名前が確認できない場合は**欠席**
3. わからない場合は、どこが理解できなかったか記入

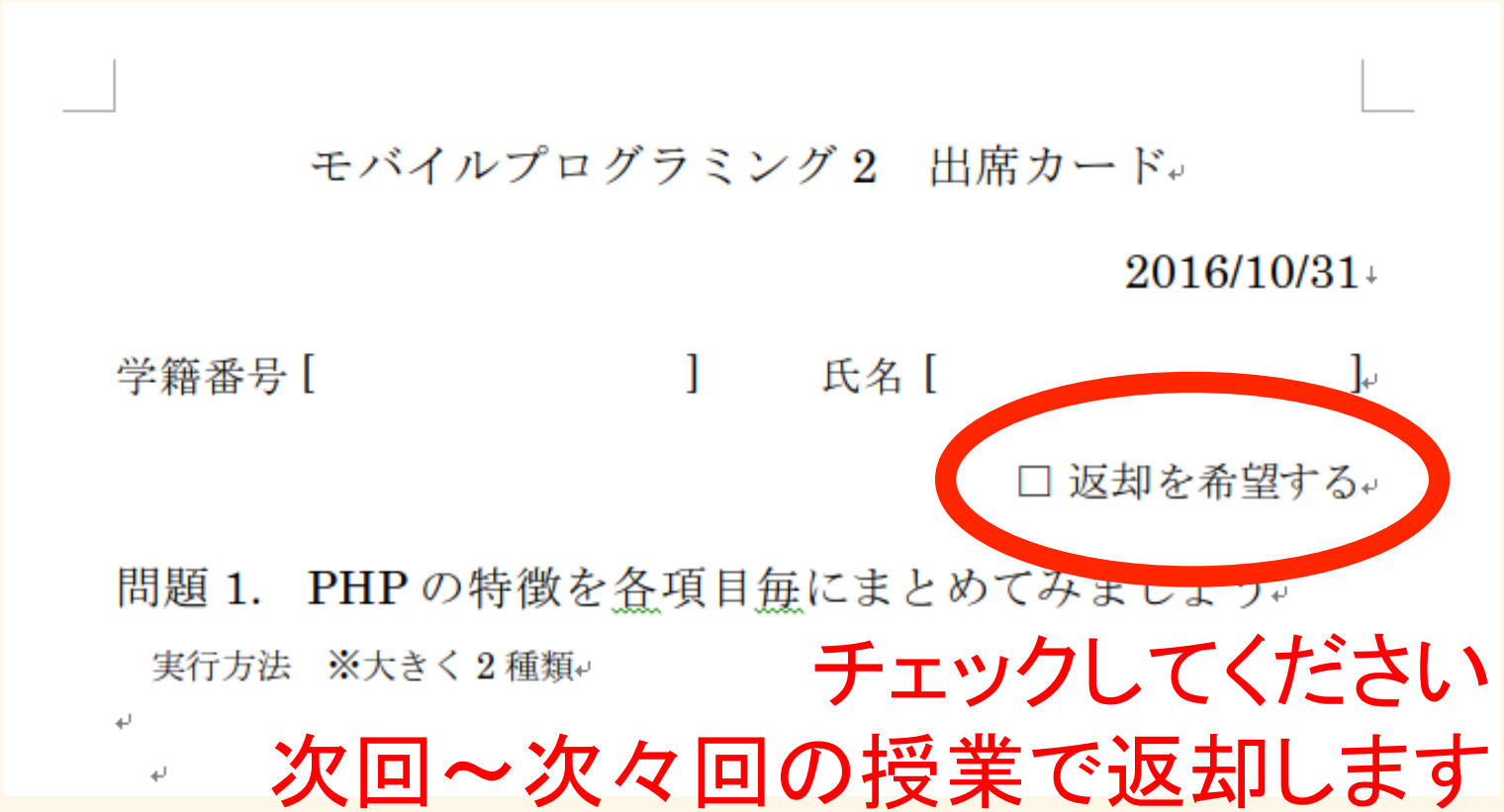
アンケート

1. 「白紙提出」「授業を聞いていたと判断できない」場合は個別にヒアリングを行います。
   - よほどのことがなければ呼び出されません
   - 大人としての自覚を持って授業に望んで下さい。

2. 一人では解決できないことがあれる場合、自分から聞きにくるように。

アンケート

- 返却を希望する場合



モバイルプログラミング 2　出席カード

2016/10/31

学籍番号 [　　]　　氏名 [　　]

☐ 返却を希望する

問題 1.　PHP の特徴を各項目毎にまとめてみましょう

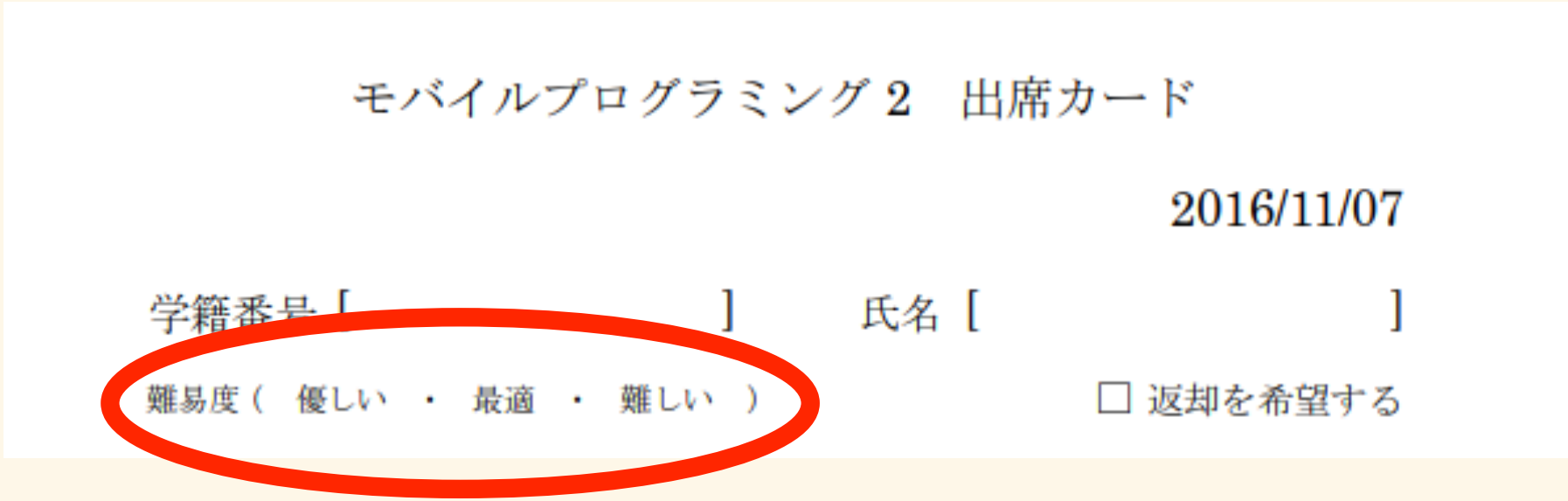
実行方法　※大きく 2 種類

チェックしてください

次回～次々回の授業で返却します

アンケート

- 難易度に○をつける



モバイルプログラミング２　出席カード

2016/11/07

学籍番号［　　］　　氏名［　　］

難易度（　優しい　・　最適　・　難しい　）　　□ 返却を希望する

○をつけてください。
様子を見て難易度を調整します。

前回の
アンケートに
答えるコーナー

質問

Q. 完成品のソースコードを公開して

A. その物ずばりのコードは原則公開しません。

私がこの学校に通っていたころ、同じ方針の先生がいました。
「なぜコードを公開しないのか」について考えてみてください。学生の内は答えにたどり着けない人が多いと思います。就職して数年経ったら改めて考えてると良いかもしれません。

質問

Q. 今までに学生に教えたことはある？今回がはじめて？講師が向いていると思う？

今回はこれにマジレスします

適正のある職業とは何か

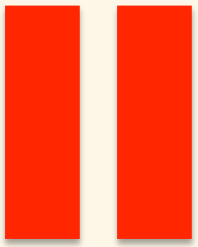
職業適性の考え方

習熟度

投下時間

職業適性の考え方

誰でも時間を投下すれば一定の習熟度にはたどり着ける。



時間を投下しなければ、スキルは身につかない。

職業適性の考え方

好きなことを仕事にする

※間違ってはいないがちょっと足りない

職業適性の考え方

好きなことを
仕事にする

時間を忘れて
集中できることを
仕事にする

職業適性の考え方

1万時間の法則

- 1日8時間投下したとすると1250日。
- およそ3.4年。
  - 10,000hour / 8hour = 1,250 day
  - 1,250day / 365day ≒ 3.4 year

よくいるエンジニア(PG)の一日

08:00
起床

10:00
出社

20:00
退社

01:00
就寝

移動中は専門書を読む、ニュースのチェックなど。ゲームをしてる場合もあり。

それほど忙しくない場合、8時間労働＋1～2時間の残業が多い。
※契約形態にもよる

このうち本当に集中してコードを書けるのは3～4時間前後。ミーティングや雑務もこなす。

移動中は朝と同様。

帰宅後に1時間程度の勉強や最新の動向をチェックするなど。

0.5H

7.0H

0.5H + 1H

平日、1日の間に技術に触れる時間は**9時間**ほど。
※勉強会やイベントに行ったり、土日に集中して勉強することも。

まとめ

- 今までの学校生活で、何時間プログラムに投下しましたか？それは自分で満足行く時間ですか？
- 就職すると1日中コードにまみれた生活を送ることになりますが、楽しそうだと思いますか？

両方イエスならぜひプログラマーの世界に！

復習

MySQL基礎

復習

インデックス

インデックス

- データベースの多くには、大量のデータから目的のレコードをすばやく見つけ出すために「**インデックス**」と呼ばれる仕組みが用意されています。

本の目次をイメージしてください。

実践「インデックス」 その4

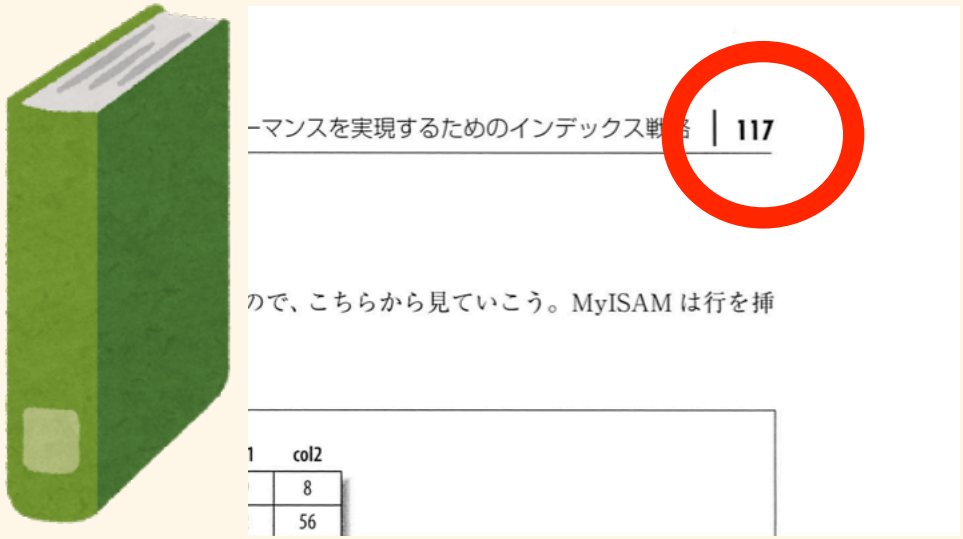
```
mysql> select * from jp_address1
    -> where id='192098300';

mysql> select * from jp_address2
    -> where id='192098300';
```

- 実際にSELECT文を実行し、速度を比較してみましょう。

プライマリーキー

- テーブルの中から一意(ユニーク)なレコードを特定できるカラムを**「プライマリーキー」**または「主キー」と呼びます。



ページ番号のような物です。

実践「インデックス」 その6

```
mysql> alter table jp_address2
    -> add index idx_town(town_name);

mysql> select  *
    -> from    jp_address2
    -> where  town_name='片倉町';
```

- インデックスは、テーブルを作成した後でも張ることができます。
  - インデックスを張ったら、SELECT文で確認してみましょう。
- create index文でも作成できます。
- alter tableはインデックスを張る以外にも、カラムの追加や削除といったテーブルの定義を変更する際にも用います。

実践「インデックス」 その7

```
mysql> show index from jp_address2;
```

- インデックスが貼られているか確認するには show indexを使用します。

通常のインデックスは
巻末の索引をイメージ

その他の情報

- **UNIQUE** インデックス
  - プライマリーキー以外でも、ユニーク(一意)であるカラムに張ることができます。
  - 重複する値をINSERTしようとすると、エラーとなります。

- **FULL TEXT**インデックス
  - 通常、where句でlikeを使用した検索にはインデックスが使用されません。
  - このような全文検索を使用する場合には FULL TEXTインデックスを張っておく必要があります。
  - データベースによってはこのような機能がない場合があります。

その他の情報

- **複合インデックス**
  - インデックスのカラムは1つだけではなく、複数同時に指定することができます。
  - where句で毎回同時に複数カラムを指定する場合に使ってみましょう。

```
mysql> alter table foo
    -> add index indexname(name, age, postcd);
```

トランザクション

トランザクション

```
mysql> delete from foo
    -> ;
```

WHERE句を入力し忘れた＼(^o^)／

トランザクション

銀行口座A

残高100万円

銀行口座B

残高0円

銀行口座Aにある100万円を、
銀行口座Bに振り込みたい

トランザクション

クライアント

銀行口座A

銀行口座B

残高100万円

残高0円

残高を100万円減らす

残高を100万円増やす

残高0円

残高100万円

トランザクション

クライアント

銀行口座A

残高100万円

銀行口座B

残高0円

残高を100万円減らす

残高を100万円増や

障害が発生、
更新ができなかった

残高0円

残高0円

実践「トランザクション」 その2

```
mysql> START TRANSACTION;
mysql> delete from jp_address1;
mysql> select count(*)
    -> from jp_address1;
```

- トランザクションを開始するには「START TRANSACTION」と入力します。
- delete文で削除し、本当に消えているか確認しましょう。

実践「トランザクション」 その3

```
mysql> ROLLBACK;
mysql> select count(*)
    -> from jp_address1;
```

- 「**ROLLBACK**」でSTART TRANSACTIONの地点まで操作を取り消すことができます。

実践「トランザクション」 その4

```
mysql> START TRANSACTION;
mysql> delete from jp_address1;
mysql> COMMIT;
```

- 操作を確定したい場合は「**COMMIT**」と打ちます。

トランザクション

クライアント

銀行口座A

銀行口座B

START TRANSACTION

UPDATE (残高を100万円減らす)

UPDATE (残高を100万円増やす)

COMMIT

トランザクション

クライアント

銀行口座A

残高100万円

銀行口座B

残高0円

START TRANSACTION

UPDATE (残高を100万円減らす)

UPDATE (残高を100万円増やす)

障害が発生、
更新できなかった

ROLLBACK

残高100万円

残高0円

ACID

- **原子性**（Atomicity）
  - 操作の途中ではないことが保証される
  - すべての操作が終わっている、または始まっていない状態であること
- **一貫性**（Consistency）
  - データベースのルールに反する操作が行われるとトランザクションは実行されない（操作前の状態に戻る）
- **独立性**（Isolation）
  - 実行中の操作は、他の操作に影響されない。
- **永続性**（Durability）
  - データベースからトランザクション中の処理が完了したという通知を受けたら、それ以降に元の状態に巻き戻ることはない。

気をつけるポイント

- **リソース食い**
  - データベースの実装にもよりますが、トランザクションは操作が最終的に確定されるまで、最初の情報を保持し続けることになります。
  - そのためのリソースはバカになりません。

- **ロック**
  - トランザクション中、（設定によっては）ロックがかかり他のプロセスは読み込みも含めてできなくなります。
  - ACIDでいう独立性(Isolation)

復習

PHP/MySQL 連携

PHPからSQLを実行する～準備

```
<?php
$dsn  = 'mysql:dbname=rpgdb;host=127.0.0.1';
$user = 'root';
$pw   = 'H@chiouji1';

$sql = 'SELECT * FROM Monster';
```

PHPからSQLを実行する～実行

```
//SQLを実行
$dbh = new PDO($dsn, $user, $pw);  //接続
$sth = $dbh->prepare($sql);        //SQL準備
$sth->execute();                   //実行

//結果を取得
while( ($buff = $sth->fetch()) !== false ){
    print $buff['id'];       //←先週と変更しました
    print "¥n";
}
```

MySQL基礎

正規化・結合

※注意※

「正規化」「結合」はRDBを学習する際に大抵の人がつまずきます。

※注意※

復習と実践が必須。
授業外でも手を動かし
練習してみて
くださいね。

Twitterのテーブル構造を考えてみる

Twitterのテーブル構造を考えてみる

Twitterのテーブル構造を考えてみる

名前(表示用)

ユーザー名

ツイート内容

RT数

いいね数

ツイート日時

Twitterのテーブル構造を考えてみる

I12

	A	B	C	D	E	F	G
1	ツイートID	ユーザー名	名前(表示用)	ツイート内容	いいね数	RT数	ツイート日時
2	bigint	varchar(15)	varchar(60)	varchar(255)	int	int	int
3	1234567891	FFXVJP	FINAL FANTASY XV	メキシコで開催された中南米の発売日イベントには、橋本プロデューサーと開発から志田さんが参加！！深夜イベントだったにも関わらず、熱狂的なファンの方達が多く集まってくださり、大盛り上がりだったようです！	468	143	2016/12/1
4	1234567892	FFXVJP	FINAL FANTASY XV	FINAL FANTASY XVは11/29に中国でも同時発売したのですが、そのロンチイベントat上海の写真をゲット！！松田社長、田畑D、下村さんや野末さん達も登壇。ステージ超かっこいいですねぇ。（よーく見ると右端にヒューマンサイズのモーグリが…！）	720	324	2016/12/1
5	1234567893	FFXVJP	FINAL FANTASY XV	ホイミンありがとう！ノクトがバナナ食べてるかと思ったらお前だったのか・・・ #FF15 #FFXV	795	664	2016/11/29
6							

Twitterのテーブル構造を考えてみる

	A	B	C	D	E	F	G
1	ツイートID	ユーザー名	名前(表示用)	ツイート内容	いいね数	RT数	ツイート日時
2	bigint	varchar(15)	varchar(60)	varchar(255)	int	int	int
3	1234567891	FFXVJP	FINAL FANTASY XV	メキシコで開催された中南米の発売日イベントには、橋本プロデューサーと開発から も関わらず、熱狂的 集まってくれたり、大盛 うでした！			1
4	123456789	FFXVJP	FINAL FANTASY XV	FINAL FANTASY XVは1 時発売したのですが、そ at上海の写真をゲット！ D、下村さんや野末さん 超かっこいいですねぇ。 端にヒューマンサイズのモーグリか…！）			1
5	1234567893	FFXVJP	FINAL FANTASY XV	ホイミンありがとう！ノクトがバナナ食べてるかと思ったらお前だったのか・・・ #FF15 #FFXV	795	664	2016/11/29
6							

名前を変更したい場合、
どう処理すれば良いのだろう？

ユーザー名を変更するには？

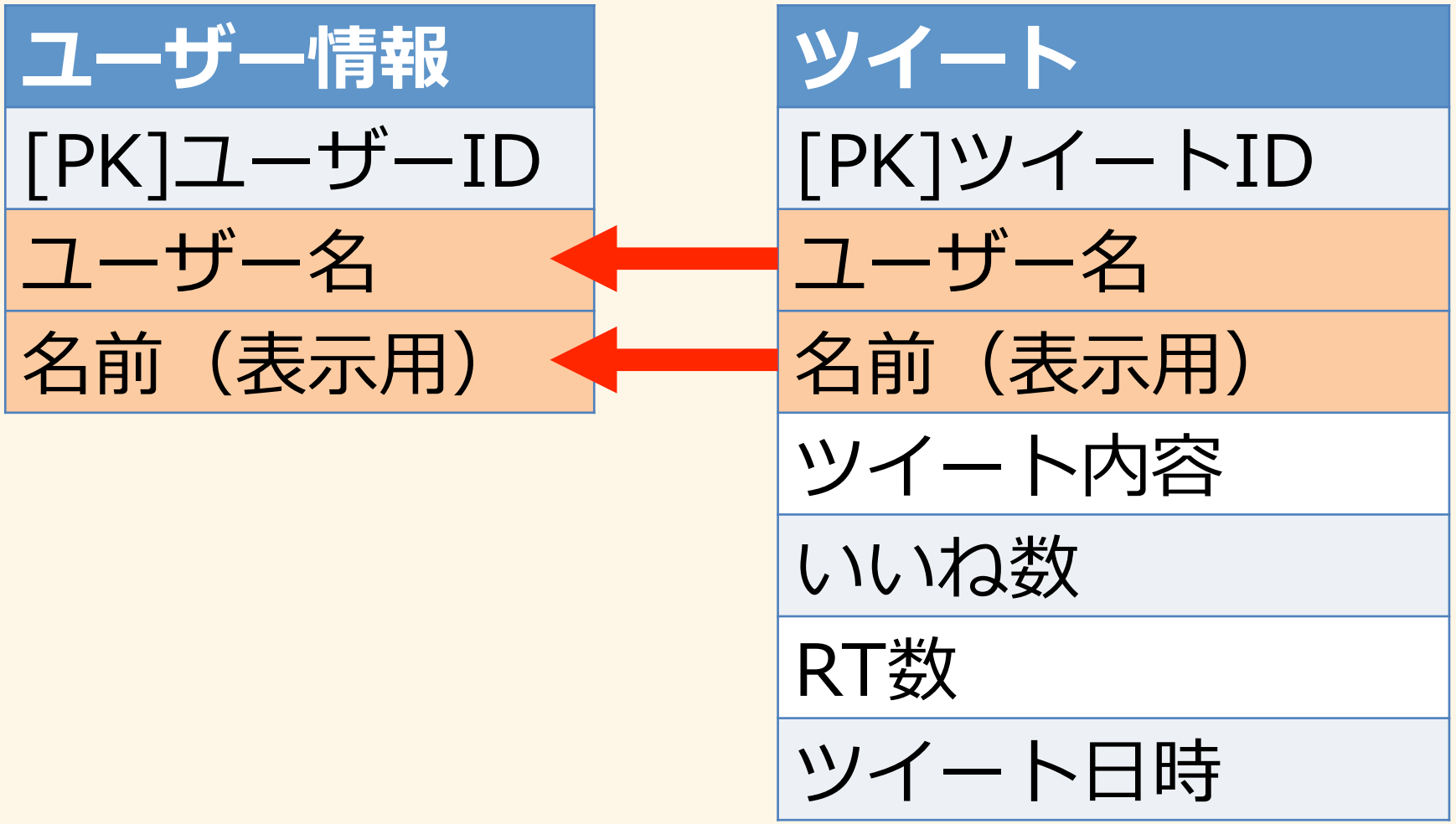
```
mysql> update  tweets
    -> set     name='NewName'
    -> where   name='OldName';
```

- この方法でも実現できます。
- しかし10万ツイートしているユーザーの場合、**最低でも10万レコードに更新処理**を行う必要があります。
- もし**1000人が実行したら1億レコード**…。
- Twitterのユーザー数は3億人以上。

そんなときに「正規化」1

ユーザー情報＋ツイート
ツイートID
ユーザー名
名前（表示用）
ツイート内容
いいね数
RT数
ツイート日時

そんなときに「正規化」2



※ユーザー情報を管理するテーブルを作成

そんなときに「正規化」3

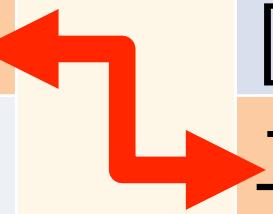
ユーザー情報
[PK]ユーザーID
ユーザー名
名前（表示用）

ツイート
[PK]ツイートID
ツイート内容
いいね数
RT数
ツイート日時

※完全に移しきった状態

そんなときに「正規化」４

ユーザー情報
[PK]ユーザーID
ユーザー名
名前（表示用）



ツイート
[PK]ツイートID
ユーザーID
ツイート内容
いいね数
RT数
ツイート日時

※ツイートテーブルにユーザーIDを追加

正規化とは

I12

	A	B	C	D	E	F	G
1	ツイートID	ユーザー名	名前(表示用)	ツイート内容	いいね数	RT数	ツイート日時
2	bigint	varchar(15)	varchar(60)	varchar(255)	int	int	int
3	1234567891	FFXVJP	FINAL FANTASY XV	メキシコで開催された中南米の発売日イベントには、橋本プロデューサーと開発から志田さんが参加！！深夜イベントだったにも関わらず、熱狂的なファンの方達が多く集まってくださり、大盛り上がりだったよう	468	143	2016/12/1
4	123456789	FFXVJP		FIN 時発 at D、 超 端			/1
5	1234567893	FFXVJP	FINAL FANTASY XV	ホー てるかと思ったらお前だったのか・・・ #FF15 #FFXV	795	664	2016/11/29
6							

表の中で値が重複している箇所を別の表に移すことを**「正規化」**といいます。

正規化

- 正規化とは
  - テーブルの中で、同じ内容が繰り返えされる箇所を、別のテーブルに移すことを「正規化」と呼びます。
- 正規化には段階がある
  - 本日は取り上げませんでしたが、第1～3正規化までの段階が存在します。
  - 第1正規化が完了したデータを「第一正規型」、正規化が全くされていないデータを「非正規型」と呼びます。

SQLで結合する1

```
mysql> select tid, name
    -> from   Tweets, User
    -> where Tweets.uid=User.uid;
```

- 結合には大きく2種類のやり方があります
  - 上記は古くから使われる WHERE句で結合する例
  - もう一つは FROM句で結合する方法

SQLで結合する2

tid	name
1234567891	FFXVJP
1234567892	FFXVJP

↑Tweetsテーブル　↑Userテーブル

ここまでを
実際にやって
みましょう

教材をDLしてください

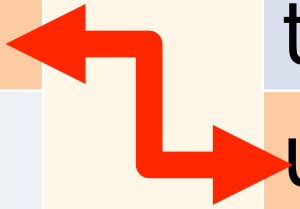
- data.zip
  https://git.io/v1Wo2

※短縮URLのアクセス先は以下
https://github.com/katsube/neec/tree/master/mobileprogramming2/20161205

データ構造

User
uid
name
displayname
register_date

Tweets
tid
uid
tweet
fav
rt
register_date



データをインポート

```
$ mysql –u root –p twitter < twitter.sql
```

- MATE端末(Terminal)を立ち上げ、配布したSQLファイルを実行します。

状態を確認する

```
mysql> show databases;

mysql> use twitter;
mysql> show tables;

mysql> desc Tweets;
mysql> desc User;
```

まずはそれぞれのテーブル内容を確認

```
mysql> select * from User;
mysql> select * from Tweets;
```

- 配布したSQLの内容が正しく入っているか確認してみましょう。

結合してみましょう 1

```
mysql> select tid, name, tweet
    -> from   Tweets, User
    -> where Tweets.uid=User.uid;
```

- 2つのテーブルを結合し、それぞれのテーブルの情報を取り出します。
- ここではWHERE句で結合しています。

結合してみましょう2

```
mysql> select tid, name, tweet
    -> from   Tweets join User
    ->             on Tweets.uid=User.uid;
```

- 今度はFROM句で結合してみます。
  - WHERE句、FROM句、どちらの結合を用いてもかまいません。
  - WHERE句に絞り込みのための条件を記述する場合はFROM句に書いた方が可読性が向上するでしょう。

結合 - JOIN

- **WHERE句**
  - WHERE句で結合するカラムを指定
  - 絞り込む条件と併記することも可能
    - WHERE Tweets.uid = User.uid
      AND Tweets.tid = 1234567891

- **FROM句**
  - FROM句で結合するカラムを指定
  - WHERE句の条件が複雑な場合はこちらがおすすめ。

カラム名がかぶったら？ 1

```
mysql> select tid, register_date
    -> from   Tweets, User
    -> where Tweets.uid=User.uid;

ERROR 1052 (23000): Column
'register_date' in field list is ambiguous
```

- register_dateは両方のテーブルに存在し、どちらもデータの意味が異なります。
- このSQLを実行するとエラーになります。

カラム名がかぶったら？2

```
mysql> select tid, Tweets.register_date
    -> from   Tweets, User
    -> where Tweets.uid=User.uid;
```

- 両方のテーブルに存在するカラムの場合、**カラム名の前にテーブル名を指定**すれば解決します。

テーブルに別名をつける 1

```
mysql> select tid, A.register_date
    -> from   Tweets A, User B
    -> where A.uid=B.uid;
```

- 毎回テーブル名を書くのは非常に面倒なので、SQLでは省略名を定義することができます。
- FROM句でテーブル名に続いて指定します。
- **エイリアス**と呼びます

テーブルに別名をつける 2

```
mysql> select tid, A.register_date
    -> from   Tweets A  join User B
    ->                  on A.uid=B.uid;
```

- FROM句で結合する場合は上記の通り

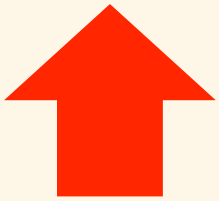
ここまでは、
それほど難しい
(≒ややこしい)
ところでは
ありません。

内部結合
と
外部結合

SQLで結合する

User
uid
name
displayname
register_date

Tweets
tid
uid
tweet
fav
rt
register_date



ユーザーを削除したら
どうなる？

ユーザーを削除して実行 1

```
mysql> delete from User
    -> where uid=2;

mysql> select tid, name, tweet
    -> from   Tweets A join User B
    ->              on A.uid=B.uid;
```

- User.uid=2を消して先程のSQLを実行します。

ユーザーを削除して実行 2

```
mysql> delete from User
    -> where uid=2;

mysql> select tid, name, tweet
    -> from User A right outer join Tweets B
    ->              on A.uid=B.uid;
```

- right outerを指定すると「右外部結合」

ユーザーを削除して実行 3

```
mysql> delete from User
    -> where uid=2;

mysql> select tid, name, tweet
    -> from User A left outer join Tweets B
    ->              on A.uid=B.uid;
```

- left outerを指定すると「左外部結合」

結合 - JOIN

- **内部結合** (inner join)
  - 両方のテーブルに存在するレコードを抽出
  - 等価結合とも言います。
- **外部結合**
  - 右外部結合 (right outer join)
    - 右側のテーブルに存在するレコードのみを抽出
  - 左外部結合 (left outer join)
    - 左側のテーブルに存在するレコードのみを抽出
  - OracleにはWHERE句で記述できる方言がありますが、基本的にFROM句で指定します。

結合 - JOIN

- **自己結合**
  - FROM User A, User B といったように自分自身と結合します。
  - 再起結合とも言います。

- **クロス結合** (cross join)
  - 2つのテーブルを総当りでジョインします。
    - Aテーブルに3レコード、Bテーブルに5レコードあった場合、15レコードが結果として得られます。
  - 直積結合とも言います。

【予告】
レポート

午前の評価

復習

- レポート (50%)
  - 11月後半～12月前半ごろ実施
  - 冬休みまでに提出
- 定期試験 (50%)
  - 授業に出ていれば解ける難易度。ヌルゲーです。

午後の評価

- チャット (50%)
  - 要件を満たしていれば満点
- オンライン対戦 (50%)
  - 要件を満たしている、チームへ貢献していれば満点

レポート

- 指定する**ゲームのテーブル設計**を行ってもらう予定です。
  - 「テーブル設計」の具体的なやり方は次回。
  - 「ゲーム」の指定、提出方法も次回。
    - 基本的にF2Pのスマホゲームの予定
  - 100%正確に行うことは不可能なので、主要な項目についてのみでOK。

- 「テーブル設計」は、本日説明をした「正規化」を行うことが前提です。

レポート

- 締切は以下の予定
  2016年12月26日(月)　16:00必着

- 課題の提示から締切りまで2週間を設定しています。この期間中に必ず提出してください。
  - 締め切り後の提出、未提出の場合はレポート点は0となります。
  - 提出された内容が一定の基準を達しているかで評価を行います。

レポート例

- 「ツムツム」のテーブル設計を行ってください。

- 回答例
  - ユーザー テーブル
    - LINE ID、レベル、経験値、所有コイン、所有ルビー…
  - ツム(マスター) テーブル
    - ツムID、ツム名、必殺技…
  - ツム(所持) テーブル
  - ショップ テーブル

:

:

※予習のすすめ※

- 来週のテーブル設計は**「ディズニーツムツム」**を事例として取り上げる予定。
- 1回以上プレイしておくとスムーズに理解できること受け合いです。