モバイル
プログラミング2

M.Katsube

2016/11/28
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/v1k5G

※短縮URLの飛び先は以下
https://github.com/katsube/neec/tree/master/mobileprogramming2/20161128

本日の予定

午前

- MySQL基礎
  - インデックス
  - トランザクション
- PHP/MySQL 連携
  - 接続
  - SQLの実行

午後

- PHP演習
  - ガチャ開発 (MySQL版)

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

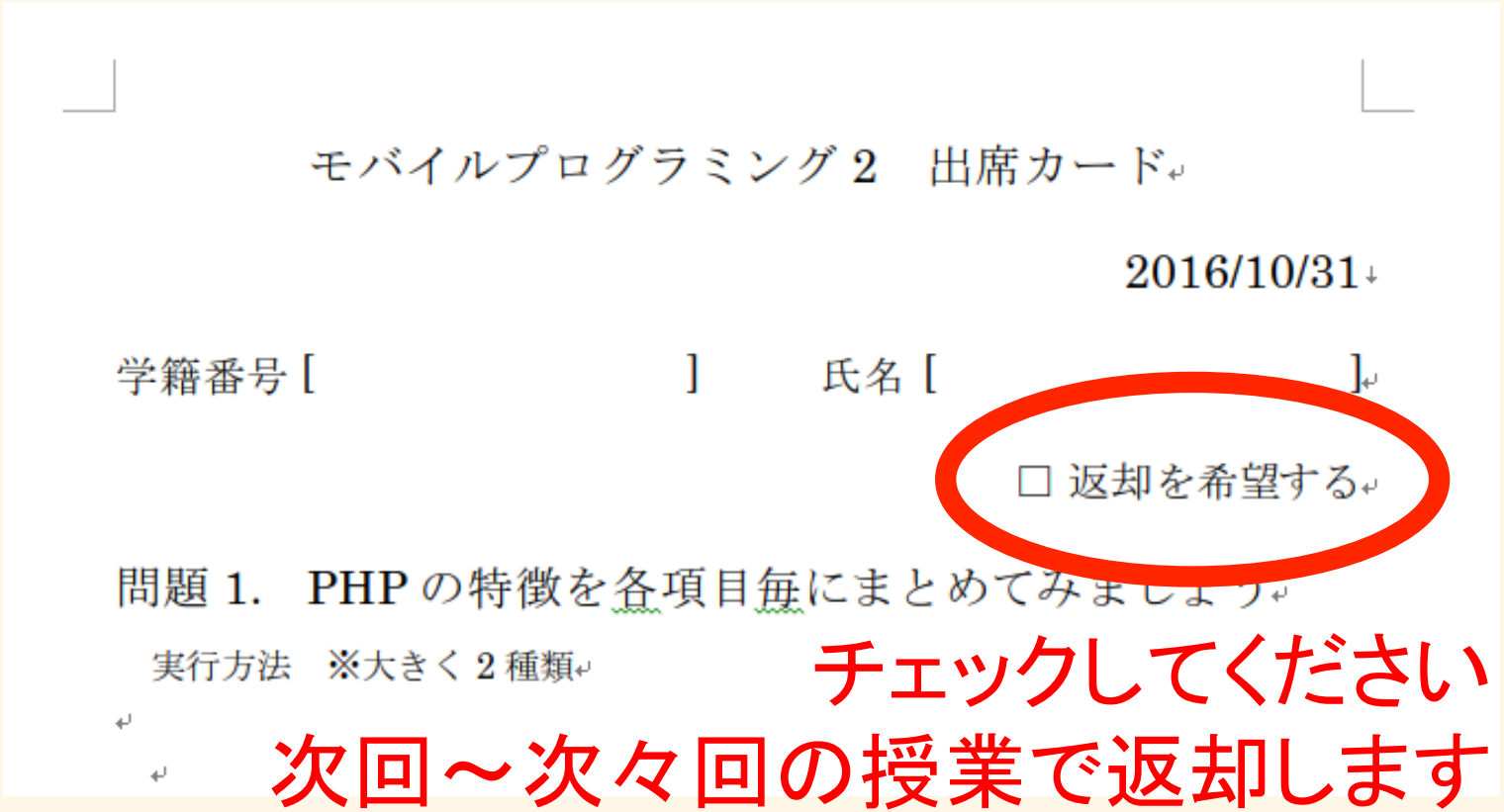
アンケート

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

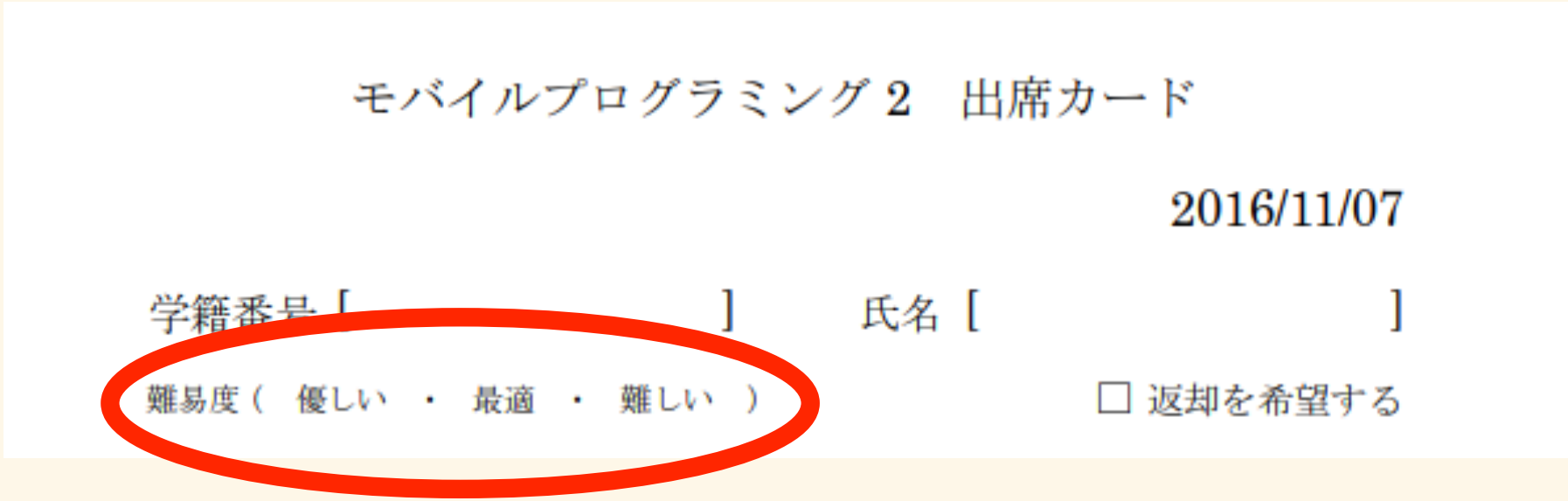
実行方法　※大きく 2 種類

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

1. チャット関連
    1. クエリーを表示する方法
    2. ログの表示
    3. 改行したい
2. おすすめのエディター
3. ポケモンやりました？
4. なぜ工学院の先生になったんですか？

クエリーを表示

```
<form action="disp.php">
   <input type="text" name="foo">
   <input type="submit">
</form>
```

```
<?php
print $_REQUEST['foo'];
```

この段階でわからないようであれば、早めに言ってくださいね。
どんどんついて来れなくなっちゃいます。

ログの表示

```
$fp = fopen('foo.log', 'r');
while( ($buff = fgets($fp)) !== false ){
    // $buffにファイルの内容が入っている
}
fclose($fp);
```

11月7日の復習をする、
GitHubのsample/BBS/BBS.class.phpを参照してください。

改行したい

行末で改行

 pはparagraph(段落)

一番手っ取り早いのは"HTML 改行" などでググってください。
授業の範囲としては10月10日前後です。

エディター

- ~~男なら~~サーバエンジニアなら**Vi、Vim**
  - Vi, Emacsは基本教養
  - viはほぼすべてのサーバに入っているので知識としては必須
- 最近なら
  - VisualStudio Code
  - ATOM (https://atom.io/)
- 統合環境
  - Eclipse
  - PHPStorm (有料)

なぜ先生になったの？

- 大きく2つ
  - 自己研鑽
  - 後輩の育成
- 長い人生、一度は講師業をやっても面白そう。
- 退学した学校に、ゲーム開発者として戻るのはネタとして面白そうだ。

復習

PHP開発例

GitHubのサンプル参照

- BBS
  https://git.io/vXbcr

※短縮URLの飛び先は以下
https://github.com/katsube/neec/tree/master/sample/BBS

次回の開発デーは12月19日

- それまでに必ず「チャット」を完成させて授業に挑んでください。
  - この日に提出した物は成績に反映します。
- Chat開発に関して質問があれば、Issue上で受け付けます。

復習

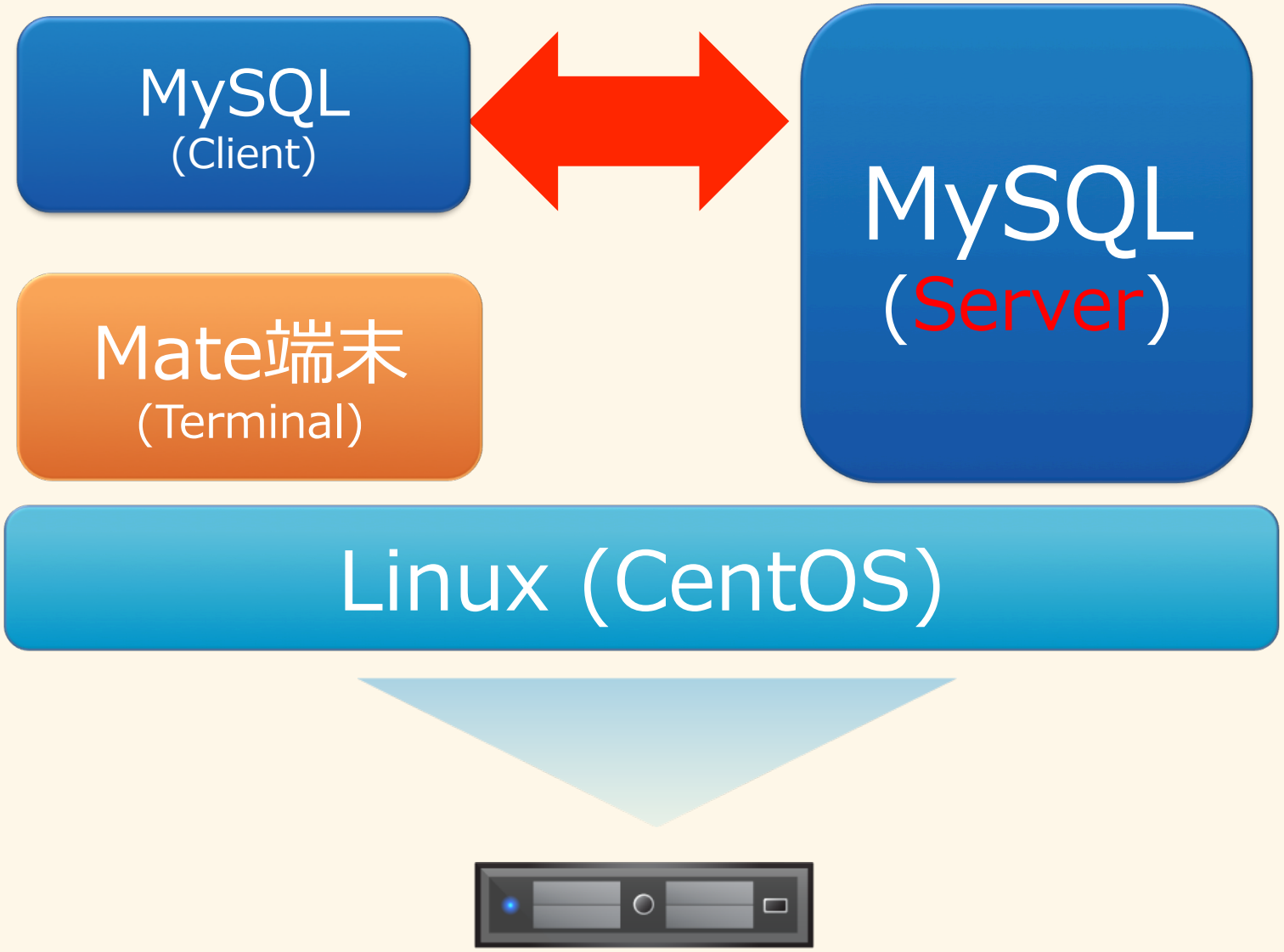
MySQL基礎

MySQLにログイン

```
$ mysql –u root –p
Enter password:
```

- **-u** ログインするユーザー名を指定
- **-p** パスワードを入力する
- パスワードは「H@chiouji1」

イメージ



データベースを表示する

```
mysql> show databases;

+--------------------------+
 | Database                 |
+--------------------------+
 | information_schema       |
 | mysql                    |
 | performance_schema       |
 | sys                      |
+--------------------------+
4 rows in set (0.01 sec)
```

- MySQLの機能

データベースを作成する

```
mysql> create database rpgdb;
```

- SQL構文
- `show databases`で作成されたことを確認してみましょう。

使用するデータベースを指定

```
mysql> use rpgdb;
Database changed
```

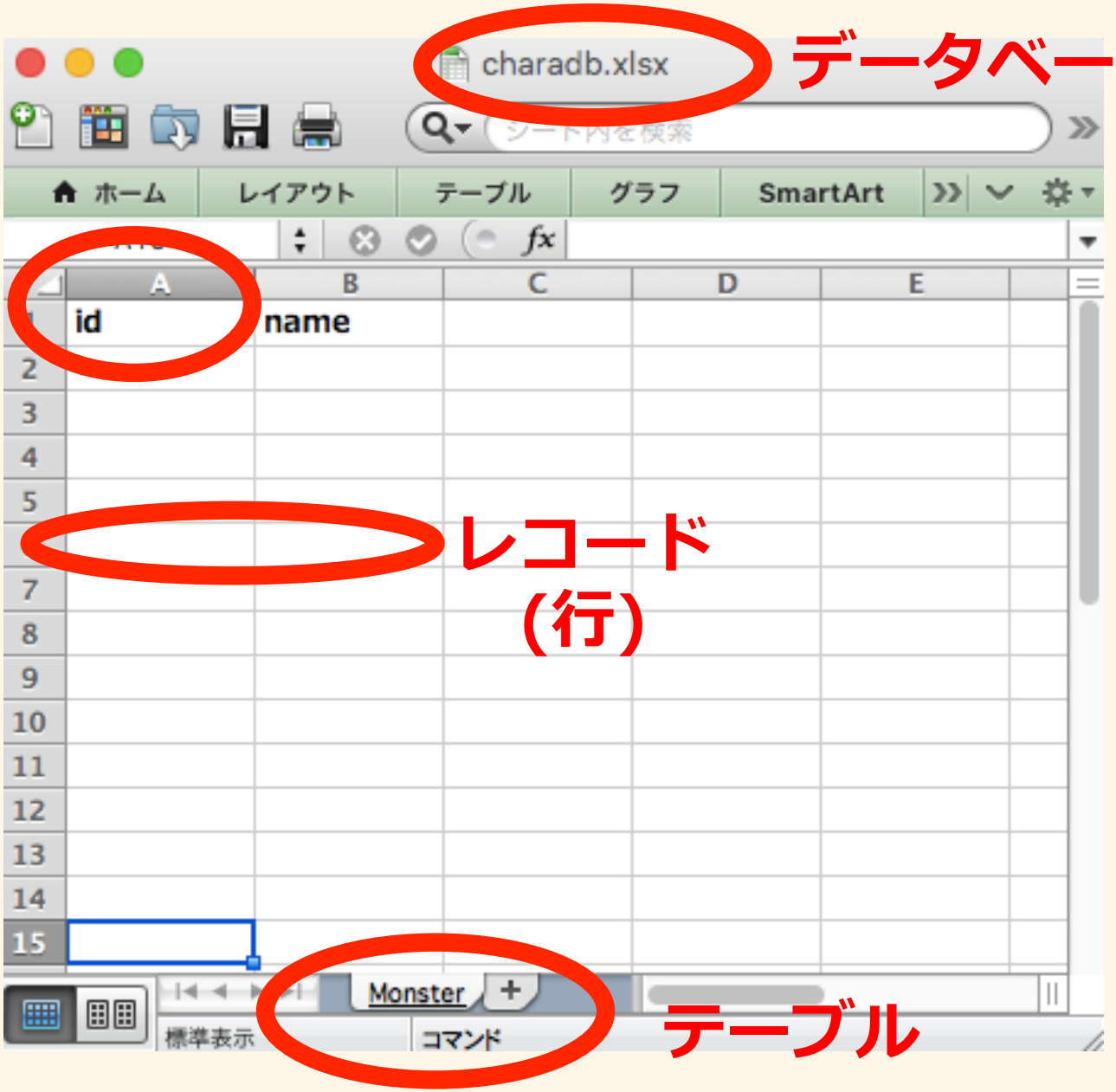
- MySQLの機能
- これから操作するデータベースを指定します。

テーブルを作成する

```
mysql> create table Monster(
    ->   id int ,
    ->   name varchar(8)
    -> );
Query OK, 0 rows affected (0.24 sec)
```

- SQL構文
- id と name の2つの列を持つテーブル「Monster」を作成します。

テーブルってなに？


charadb.xlsx
データベース
ホーム
レイアウト
テーブル
グラフ
SmartArt
カラム
(列)
id
name
レコード
(行)
Monster
標準表示
コマンド
テーブル

テーブルのカラムには型がある

- 数値
  - 整数 (TINYINT, SMALLINT, MIDIUMINT, INT, BIGINT)
    ※ INT UNSIGNED とすると符号なしにできる
  - 小数点 (FLOAT, DOUBLE, DECIMAL, NUMERIC)
  - ビット (BIT)
- 日付と時刻 (YEAR, DATE, TIME, DATETIME, TIMESTAMP)
- 文字列 (CHAR, VARCHAR, TEXT)
- バイナリ (BINARY, BLOB)

※詳細は公式ドキュメントを参照
https://dev.mysql.com/doc/refman/5.6/ja/data-types.html

テーブルの一覧を表示

```
mysql> show tables;
+----------------------+
| Tables_in_charadb |
+----------------------+
| Monster                    |
+----------------------+
```

- MySQLの機能

テーブルの構造を表示

```
mysql> desc Monster;
+---------+---------------+------+-----+----------+--------+
| Field   | Type          | Null | Key | Default | Extra |
+---------+---------------+------+-----+----------+--------+
| id      | int(11)       | YES  |     | NULL    |        |
| name    | varchar(8)    | YES  |     | NULL    |        |
+---------+---------------+------+-----+----------+--------+
```

- MySQLの機能

テーブルにレコードを挿入

```
mysql> insert into Monster
    -> values(1, 'dragon');
Query OK, 1 row affected (0.00 sec)
```

- SQL構文
- Monsterテーブルにデータを挿入。

テーブルのデータを表示

```
mysql>select id, name from Monster;
+------+----------+
| id   | name     |
+------+----------+
|    1 | Dragon   |
+------+----------+
1 row in set (0.00 sec)
```

- SQL構文
- Monsterテーブルのデータをすべて表示。

テーブルのデータを表示

```
mysql>select * from Monster;
+------+----------+
| id   | name     |
+------+----------+
|    1 | Dragon   |
+------+----------+
1 row in set (0.00 sec)
```

- SQL構文
- すべての列を対象とする場合、アスタリスク(*)で代用することができる。

MySQLから抜ける

```
mysql> ¥q
$
```

- MySQLの機能
- ¥q または exit と入力することでMySQLから抜けて、元のLinuxの環境に戻ることができます。

MySQL基礎

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1

```
$ mysql –u root –p
Enter password:


mysql> create database jpadr;
mysql> show databases;
```

- 実際に試してみましょう。
- まずはデータベースを作成します。

実践「インデックス」 その2

```
$ mysql –u root –p jpadr <noindex.sql
Enter password:

$ mysql –u root –p jpadr <index.sql
Enter password:
```

- MATE端末(Terminal)をもう一つ開きます
- 大量にデータを入れる「テーブル」を作成し、データをINSERTします。
  - noindex.sql インデックスなし
  - index.sql インデックスあり

実践「インデックス」 その3

```
mysql> use jpadr;
mysql> show tables;
mysql> select count(*) from jp_address1;
mysql> select count(*) from jp_address2;
```

- テーブルがあるか、データが入っているか確認しましょう。

実践「インデックス」 その4

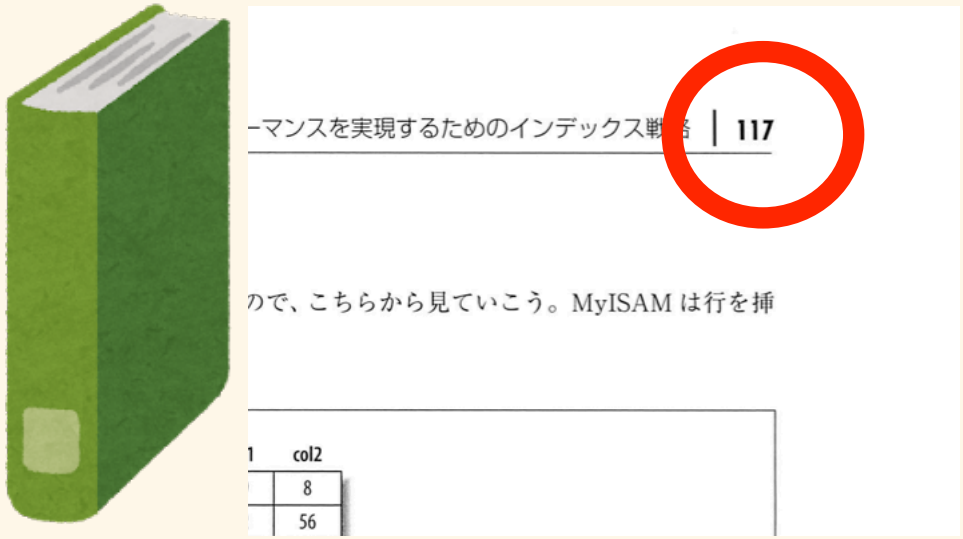
```
mysql> select * from jp_address1
    -> where id='192098300';

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5

```
mysql> select  *
    -> from    jp_address1
    -> where  town_name='片倉町';
```

- 同一の値のあるカラムの検索を早くしたい場合はどうすれば良いでしょう？

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1

```
$ mysql –u root –p
Enter password:


mysql> use jpadr;
```

- 実際に試してみましょう。
- インデックスで作成したデータベースをそのまま使用します。

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)

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 implode(', ', $buff);
    print "¥n";
}
```