Mongo DB vs. MySQL

Pokémon

岡　　亮
田部　悠介
永山　裕人
和久井　拓

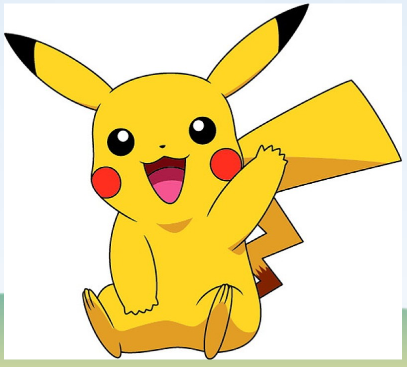
Mongo DB vs MySQL
Pokémon

▶ おか りょう
たなべ ゆうすけ
ながやま ゆうと
わくい たく

fppt.com

What's Pokêmon ?

- Pokêmon is fictional creatures
- Pokêmon have various elements
  - type(タイプ),moves(わざ)that can be acquired
  - weight, height and more …
    (omitted in this report)
- relationship between Pokêmon and elements
  - each Pokêmon has 1 or 2 types.
  - each Pokêmon can acquire many moves



Example:Blaziken(バシャーモ)

element of pokemon:

no, name, type1, type2

No. 257　バシャーモ

タイプ 1	タイプ 2
ほのお	かくとう

弱点

みず	じめん
ひこう	エスパー

30かいだての　ビルを　ジャンプで　とびこす　ちょうやくりょく。ほのお　のパンチが　あいてを　やきつくす。

覚える技

技名	タイプ	分類	威力	命中	PP
フレアドライブ	ほのお	物理	120	100	15
ほのおのパンチ	ほのお	物理	75	100	15
とびひざげり	かくとう	物理	130	90	10

element of move:

name,type,tech,power,acc,PP

how to make MySQL

- pokemon(id,name,type1,type2)
- move(id,name,type,tech,power,acc,pp)
- pokemon_move(pokemon,move)

pokemons

no	name	type1	type2
001	フシギダネ	3	8
002	フシギソウ	3	8
003	フシギバナ	3	8
004	ヒトカゲ	2	null
005	リザード	2	null
006	リザードン	2	11

pokemon_move

pokeomn_no	move_id
019	1
020	1
025	1
026	1
029	1
030	1

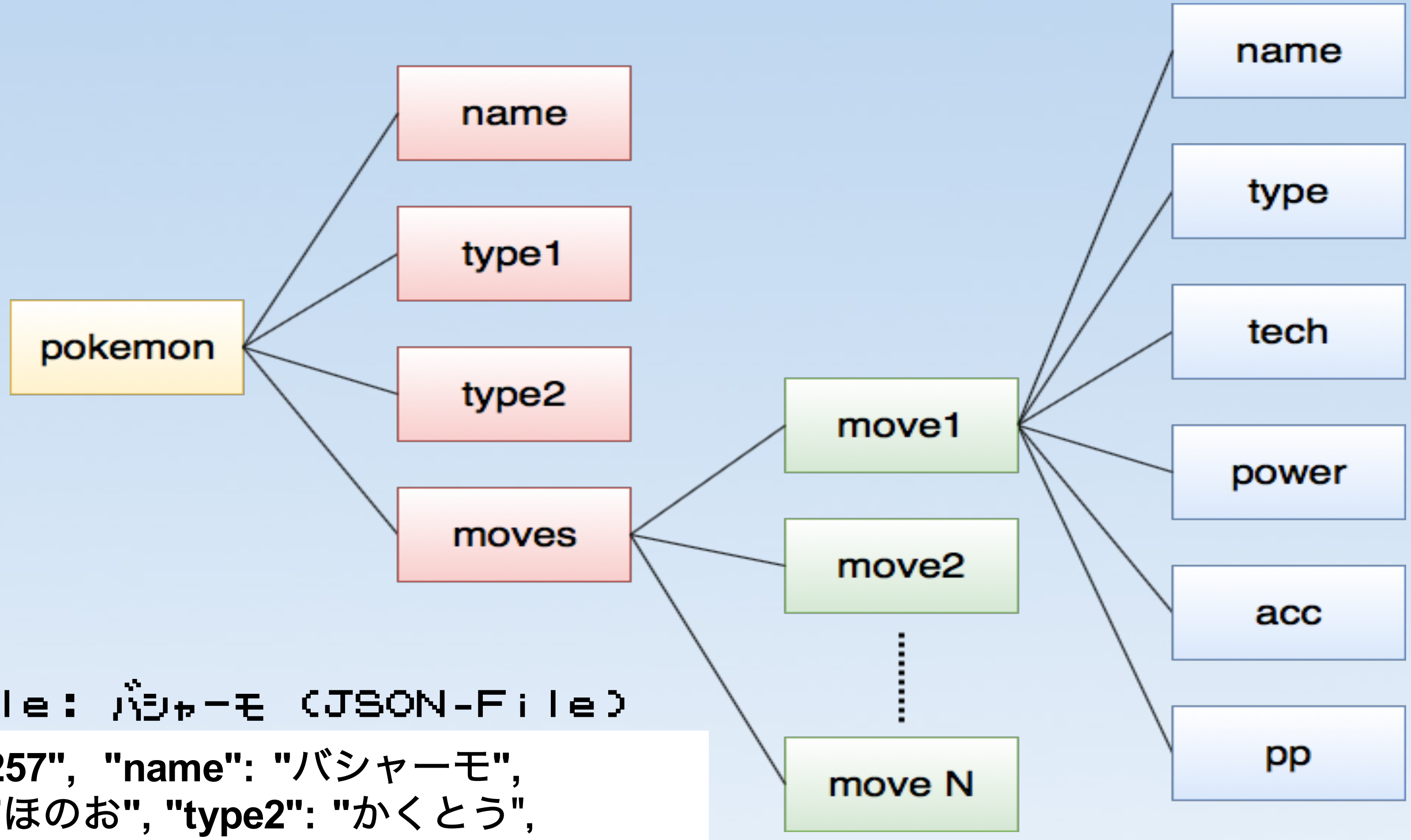
moves

id	name	type	tech	power	acc	pp
1	10まんボルト	4	とくしゅ	90	100	15
2	あおいほのお	2	とくしゅ	130	85	5
3	あくうせつだん	15	とくしゅ	100	95	5
4	あくのはどう	16	とくしゅ	80	100	15
5	アシストパワー	11	とくしゅ	20	100	10
6	アシッドボム	8	とくしゅ	40	100	20

types

id	name
1	ノーマル
2	ほのお
3	みず
4	でんき
5	くさ
6	こおり

how to make Mongo DB



Example: バシャーモ (JSON-File)

```
{ "no": "257", "name": "バシャーモ",
"type1": "ほのお", "type2": "かくとう",
        “move”: [ { “m_name”: “ひのこ”,“m_type”: “ほのお”,“m_tech”: “特殊”
                  ,"m_power": 40,"m_acc": 100, "m_pp": 25},
                  {"m_name": "いびき","m_type": "ノーマル", "m_tech": "特殊”
                  ,"m_power": 50,"m_acc": 100, "m_pp": 15},
                  …..]}
```

Query 1.

Show only Fire Pokémon

MySQL

Mongo DB

query

```
select p.no, p.name
from pokemons p,types t
where(p.type1=t.id
or p.type2=t.id)
and t.name='ほのお';
```

```
db.pokemon.find(
{{$or:[{type1:"ほのお"},
{type2:"ほのお"}]},
{_id:0,type1:0,
type2:0,move:0});
```

result

004	ヒトカゲ
005	リザード
006	リザードン
037	ロコン
038	キュウコン
059	ウインディ

...

```
{"no":"004","name": "ヒトカゲ"}
{"no":"005","name": "リザード"}
{"no":"006","name":"リザードン"}
{"no":"037","name":  "ロコン"}
{"no":"038","name":"キュウコン"}
{"no":"059","name":"ウインディ"}
...
```

Query 2.

Fire Pokémon with Fighting Moves

MySQL

Mongo DB

query

```
select p.no, p.name
from pokemons p,
pokemon_move pm,
types pt,types mt,moves m
where (p.type1=t.id
or p.type2 = t.id)
and t.name=‘ほのお’ and
p.no=pm.pokemon_no and
pm.move_id=m.id and
m.type = mt.id
and mt.name = ‘かくとう’;
```

```
db.pokemon.find(
{$and:[{$or:[{"type1":"ほのお"}
,{"type2":"ほのお"}]}
,{"move.m_type":"かくとう"}]}
,{_id:0,type1:0
,type2:0,move:0});
```

result

```
004 | ヒトカゲ
005 | リザード
006 | リザードン
058 | ガーディ
059 | ウインディ
077 | ポニータ
...
```

```
{“no”:”004”,”name”: “ヒトカゲ”}
{“no”:”005”,”name”: “リザード”}
{“no”:”006”,”name”:”リザードン”}
{“no”:”059”,”name”:“ウインディ”}
{“no”:”058”,”name”: ”ガーディ”}
{“no”:”077”,”name”: ”ポニータ”}
...
```

Query 3.

Fire Pokémon with Fighting Moves (power of move >= 100)

MySQL

Mongo DB

query

```sql
select p.no, p.name
from pokemons p,
pokemon_move pm,
types pt,types mt,moves m
where (p.type1=t.id
or p.type2 = t.id)
and t.name='ほのお' and
p.no=pm.pokemon_no and
pm.move_id=m.id and
m.type = mt.id
and mt.name = 'かくとう' and
m.power >= 100;
```

```
db.pokemon.find(
{$and:[{move:{$elemMatch:
{m_type:"かくとう"
,m_power:{$gte: 100}}}}
,{$or:[{type1:"ほのお"}
,{type2:"ほのお"}]}]}
,{_id:0,type1:0
,type2:0,move:0});
```

result

```
004   ヒトカゲ
005   リザード
006   リザードン
058   ガーディ
059   ウインディ
126   ブーバー
...
```

```
{"no":"004","name": "ヒトカゲ"}
{"no":"005","name": "リザード"}
{"no":"006","name":"リザードン"}
{"no":"059","name":"ウインディ"}
{"no":"058","name": "ガーディ"}
{"no":"126","name": "ブーバー"}
...
```

Consideration about Type Effect

- If we want to calculate type compatibilities ...
  - MySQL is suitable
    - make table about type compatibilities

ほのお is stronger than くさ!

type1	type2	rate
ほのお	でんき	1
ほのお	くさ	2
ほのお	こおり	2
ほのお	かくとう	1

  - Mongo DB is not suitable
    - It requires a lot of descriptions
    - It's hard to search with multi tables

Implement of Type Effect

MySQL

It can be executed easily by using "comps" table!

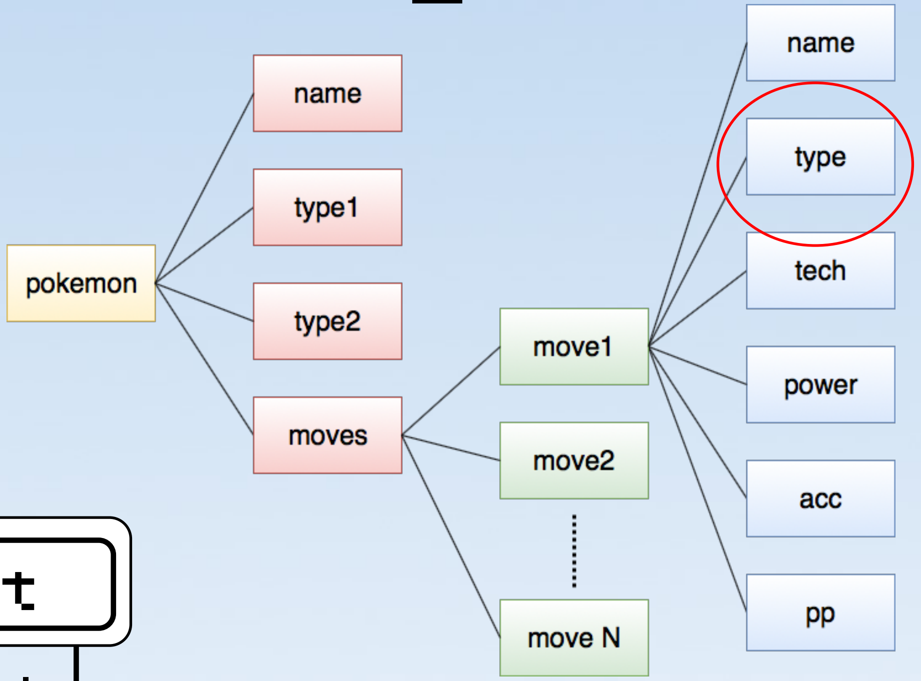
query

```
select te.name
from types t, types te
, comps c
where t.name = 'ほのお'
and t.id=c.m_type
and c.rate=2
and c.p_type=te.id;
```

result

name
くさ
こおり
はがね
むし

Mongo DB



We have to add component about type comp under "type"

Feature of Each Other

- MySQL
  - ◎ Complex queries can be executed
  - ✕ Data structure is strict
    - Data types must be defined beforehand
- Mongo DB
  - ◎ JavaScript notation can be used
    - Searchable from JS-File
  - ◎ Easy to read
  - ✕ Not good at complex queries
  - ✕ Data and queries are redundant