MySQL数据库 – node使用

王红元 coderwhy

目录
content

查询数据的问题

- 前面我们学习的查询语句，查询到的结果通常是一张表，比如查询手机+品牌信息：

```
SELECT * FROM products LEFT JOIN brand ON products.brand_id = brand.id;
```

id	brand	title	price	score	voteCnt	url	pid	brand_id	id(1)	name	website	phoneRank
1	华为	华为nova 3（全网通）	2699	6.7	65	http://detail.zol.com.cn/cell_phone/index1185512.shtml	1185512	100	100	华为	www.huawei.com	2
2	华为	华为P20 Pro（6GB RAM/全网通）	4488	8.3	103	http://detail.zol.com.cn/cell_phone/index1207038.shtml	1207038	100	100	华为	www.huawei.com	2
3	华为	华为P20（全网通）	3388	8.4	127	http://detail.zol.com.cn/cell_phone/index1175779.shtml	1175779	100	100	华为	www.huawei.com	2
4	华为	华为nova 3i（4GB RAM/全网通）	1999	7.0	9	http://detail.zol.com.cn/cell_phone/index1222100.shtml	1222100	100	100	华为	www.huawei.com	2
5	华为	华为Mate 10（4GB RAM/全网通）	3399	8.3	125	http://detail.zol.com.cn/cell_phone/index1165672.shtml	1165672	100	100	华为	www.huawei.com	2
6	华为	华为nova 3e（全网通）	1999	8.8	71	http://detail.zol.com.cn/cell_phone/index1207608.shtml	1207608	100	100	华为	www.huawei.com	2
7	华为	华为nova 2s（4GB RAM/全网通）	2399	7.5	97	http://detail.zol.com.cn/cell_phone/index1204363.shtml	1204363	100	100	华为	www.huawei.com	2
8	华为	华为Mate 10 Pro（全网通）	3599	8.7	92	http://detail.zol.com.cn/cell_phone/index1181128.shtml	1181128	100	100	华为	www.huawei.com	2
9	华为	华为畅享8（3GB RAM/全网通）	1099	5.3	28	http://detail.zol.com.cn/cell_phone/index1208286.shtml	1208286	100	100	华为	www.huawei.com	2
10	华为	华为P10（VTR-AL00/全网通）	3488	7.2	395	http://detail.zol.com.cn/cell_phone/index1160028.shtml	1160028	100	100	华为	www.huawei.com	2
11	华为	华为畅享8 Plus（全网通）	1499	5.8	11	http://detail.zol.com.cn/cell_phone/index1210102.shtml	1210102	100	100	华为	www.huawei.com	2
12	华为	华为麦芒7（全网通）	2399	7.6	5	http://detail.zol.com.cn/cell_phone/index1227167.shtml	1227167	100	100	华为	www.huawei.com	2
13	华为	华为Mate 9（MHA-AL00/4GB RAM/全网通）	1788	7.8	449	http://detail.zol.com.cn/cell_phone/index1143413.shtml	1143413	100	100	华为	www.huawei.com	2
14	华为	华为nova 3i（6GB RAM/全网通）	2199	7.0	9	http://detail.zol.com.cn/cell_phone/index1224424.shtml	1224424	100	100	华为	www.huawei.com	2
15	华为	华为Mate RS保时捷版（全网通）	9999	6.1	16	http://detail.zol.com.cn/cell_phone/index1210089.shtml	1210089	100	100	华为	www.huawei.com	2
16	华为	华为nova 2（PIC-AL00/全网通）	1228	8.0	209	http://detail.zol.com.cn/cell_phone/index1170042.shtml	1170042	100	100	华为	www.huawei.com	2
17	华为	华为麦芒6（全网通）	2199	6.1	57	http://detail.zol.com.cn/cell_phone/index1182274.shtml	1182274	100	100	华为	www.huawei.com	2
18	华为	华为P9（EVA-AL00/标准版/全网通）	1448	7.8	648	http://detail.zol.com.cn/cell_phone/index404275.shtml	404275	100	100	华为	www.huawei.com	2
19	华为	华为nova（CAZ-AL10/高配版/全网通）	988	6.9	198	http://detail.zol.com.cn/cell_phone/index1154505.shtml	1154505	100	100	华为	www.huawei.com	2
20	华为	华为畅享8e（全网通）	999	4.8	4	http://detail.zol.com.cn/cell_phone/index1210103.shtml	1210103	100	100	华为	www.huawei.com	2
21	华为	华为麦芒5（MLA-AL10/高配版/全网通）	1099	6.8	136	http://detail.zol.com.cn/cell_phone/index1148829.shtml	1148829	100	100	华为	www.huawei.com	2
22	华为	华为P10 Plus（VKY-AL00/6GB RAM全网通）	2488	7.5	186	http://detail.zol.com.cn/cell_phone/index1163813.shtml	1163813	100	100	华为	www.huawei.com	2
23	华为	华为Mate 9 Pro（4GB RAM/全网通）	2799	8.0	136	http://detail.zol.com.cn/cell_phone/index1159578.shtml	1159578	100	100	华为	www.huawei.com	2

将brand转成对象

- 但是在真实开发中，实际上红色圈起来的部分应该放入到一个对象中，那么我们可以使用下面的查询方式：
  - 这个时候我们要用 JSON_OBJECT;

```
SELECT products.id as id, products.title as title, products.price as price, products.score as score,
    JSON_OBJECT('id', brand.id, 'name', brand.name, 'rank', brand.phoneRank, 'website', brand.website) as brand
FROM products LEFT JOIN brand ON products.brand_id = brand.id;
```

id	title	price	score	brand
1	华为 nova 3（全网通）	2699	6.7	{"id": 100, "name": "华为", "rank": 2, "website": "www.huawei.com"}
2	华为 P20 Pro（6GB RAM/全网通）	4488	8.3	{"id": 100, "name": "华为", "rank": 2, "website": "www.huawei.com"}
3	华为 P20（全网通）	3388	8.4	{"id": 100, "name": "华为", "rank": 2, "website": "www.huawei.com"}
4	华为 nova 3i（4GB RAM/全网通）	1999	7.0	{"id": 100, "name": "华为", "rank": 2, "website": "www.huawei.com"}
5	华为 Mate 10（4GB RAM/全网通）	3399	8.3	{"id": 100, "name": "华为", "rank": 2, "website": "www.huawei.com"}
6	华为 nova 3e（全网通）	1999	8.8	{"id": 100, "name": "华为", "rank": 2, "website": "www.huawei.com"}
7	华为 nova 2s（4GB RAM/全网通）	2399	7.5	{"id": 100, "name": "华为", "rank": 2, "website": "www.huawei.com"}
8	华为 Mate 10 Pro（全网通）	3599	8.7	{"id": 100, "name": "华为", "rank": 2, "website": "www.huawei.com"}
9	华为畅享 8（3GB RAM/全网通）	1099	5.3	{"id": 100, "name": "华为", "rank": 2, "website": "www.huawei.com"}
10	华为 P10（VTR-AL00/全网通）	3488	7.2	{"id": 100, "name": "华为", "rank": 2, "website": "www.huawei.com"}
11	华为畅享 8 Plus（全网通）	1499	5.8	{"id": 100, "name": "华为", "rank": 2, "website": "www.huawei.com"}
12	华为麦芒 7（全网通）	2399	7.6	{"id": 100, "name": "华为", "rank": 2, "website": "www.huawei.com"}
13	华为 Mate 9（MHA-AL00/4GB RAM/全网通）	1788	7.8	{"id": 100, "name": "华为", "rank": 2, "website": "www.huawei.com"}
14	华为 nova 3i（6GB RAM/全网通）	2199	7.0	{"id": 100, "name": "华为", "rank": 2, "website": "www.huawei.com"}

多对多转成数组

- 在多对多关系中，我们希望查询到的是一个数组：
  - 比如一个学生的多门课程信息，应该是放到一个数组中的；
  - 数组中存放的是课程信息的一个个对象；
  - 这个时候我们要 JSON_ARRAYAGG和JSON_OBJECT结合来使用；

```
SELECT stu.id, stu.name, stu.age,
                    JSON_ARRAYAGG(JSON_OBJECT('id', cs.id, 'name', cs.name)) as courses
FROM students stu
LEFT JOIN students_select_courses ssc ON stu.id = ssc.student_id
LEFT JOIN courses cs ON ssc.course_id = cs.id
GROUP BY stu.id;
```

id	name	age	courses
1	why	18	[{"id": 1, "name": "英语"}, {"id": 3, "name": "数学"}]
2	tom	22	[{"id": null, "name": null}]
3	lilei	25	[{"id": 2, "name": "语文"}, {"id": 3, "name": "数学"}, {"id": 4, "name": "历史"}]
4	lucy	16	[{"id": null, "name": null}]
5	lily	20	[{"id": null, "name": null}]
6	hmm	25	[{"id": null, "name": null}]

认识mysql2

- **前面我们所有的操作都是在GUI工具中，通过执行SQL语句来获取结果的，那真实开发中肯定是通过代码来完成所有的操作的。**
- **那么如何可以在Node的代码中执行SQL语句来，这里我们可以借助于两个库：**
  - mysql：最早的Node连接MySQL的数据库驱动；
  - mysql2：在mysql的基础之上，进行了很多的优化、改进；
- **目前相对来说，我更偏向于使用mysql2，mysql2兼容mysql的API，并且提供了一些附加功能**
  - 更快/更好的性能；
  - Prepared Statement（预编译语句）：
    - ✓ 提高性能：将创建的语句模块发送给MySQL，然后MySQL编译（解析、优化、转换）语句模块，并且存储它但是不执行，之后我们在真正执行时会给?提供实际的参数才会执行；就算多次执行，也只会编译一次，所以性能是更高的；
    - ✓ 防止SQL注入：之后传入的值不会像模块引擎那样就编译，那么一些SQL注入的内容不会被执行；or 1 = 1不会被执行；
  - 支持Promise，所以我们可以使用async和await语法
  - 等等....
- **所以后续的学习中我会选择mysql2在node中操作数据。**

使用mysql2

- **安装mysql2**

```
npm install mysql2
```

- **mysql2的使用过程如下：**
  - 第一步：创建连接（通过createConnection），并且获取连接对象；
  - 第二步：执行SQL语句即可（通过query）；

```
// 创建连接
const connection = mysql.createConnection({
  host: 'localhost',
  database: 'coderhub',
  user: 'root',
  password: 'Coderwhy888.'
});
```

```
// 执行SQL语句
connection.query('SELECT * FROM products;',
                 (err, results, fields) => {
  console.log(err);
  console.log(results);
  console.log(fields);
  connection.destroy()
})
```

Prepared Statement

- **Prepared Statement（预编译语句）：**
  - 提高性能：将创建的语句模块发送给MySQL，然后MySQL编译（解析、优化、转换）语句模块，并且存储它但是不执行，之后我们在真正执行时会给?提供实际的参数才会执行；就算多次执行，也只会编译一次，所以性能是更高的；
  - 防止SQL注入：之后传入的值不会像模块引擎那样就编译，那么一些SQL注入的内容不会被执行；or 1 = 1不会被执行；

```
const statement = 'SELECT * FROM products WHERE price > ? and brand = ?;';
connection.execute(statement, [1000, '华为'], (err, results) => {
  console.log(results);
});
```

Connection Pools

- **前面我们是创建了一个连接（connection），但是如果我们有多个请求的话，该连接很有可能正在被占用，那么我们是否需要每次一个请求都去创建一个新的连接呢?**
  - 事实上，mysql2给我们提供了连接池（connection pools）；
  - 连接池可以在需要的时候自动创建连接，并且创建的连接不会被销毁，会放到连接池中，后续可以继续使用；
  - 我们可以在创建连接池的时候设置LIMIT，也就是最大创建个数；

```
const pool = mysql.createPool({
  host: 'localhost',
  database: 'coderhub',
  user: 'root',
  password: 'Coderwhy888.',
  connectionLimit: 5
});
```

```
const statement = `
  SELECT * FROM products
    WHERE price > ? and brand = ?;
`;
pool.execute(statement, [1000, '华为'], (err, results) => {
  console.log(results);
});
```

Promise方式

- 目前在JavaScript开发中我们更习惯Promise和await、async的方式，mysql2同样是支持的：

```
const mysql = require('mysql2');

const pool = mysql.createPool({
  host: 'localhost',
  database: 'coderhub',
  user: 'root',
  password: 'Coderwhy888.',
  connectionLimit: 5
});

const statement = 'SELECT * FROM products WHERE price > ? and brand = ?;';
pool.promise().execute(statement, [1000, '华为']).then(([results, fields]) => {
  console.log(results);
});
```