《Illuminate\Database 常用查询方式及底层逻辑》

`Illuminate\Database`（Eloquent ORM / 查询构建器）提供了多种查询方式，以下是常用查询方法及其对应的原始 SQL 语句，帮助理解底层执行逻辑：

一、基础查询

1. 查询所有数据

```
// 查询构建器
DB::table('users')->get();
// Eloquent模型
User::all();
// 对应SQL
SELECT * FROM `users`;
```

2. 指定字段查询

```
// 查询构建器
DB::table('users')->select('id', 'name', 'email')->get();
// Eloquent模型
User::select('id', 'name', 'email')->get();
// 对应SQL
SELECT `id`, `name`, `email` FROM `users`;
```

3. 条件查询（WHERE）

```
// 查询构建器
DB::table('users')->where('status', 1)->get();
// Eloquent模型
User::where('status', 1)->get();
```

```
// 对应SQL
SELECT * FROM `users` WHERE `status` = 1;
```

二、条件组合查询

1. AND 多条件

```
// 查询构建器
DB::table('users')
    ->where('status', 1)
    ->where('age', '>', 18)
    ->get();
// Eloquent模型
User::where('status', 1)->where('age', '>', 18)->get();
// 对应SQL
SELECT * FROM `users` WHERE `status` = 1 AND `age` > 18;
```

2. OR 条件

```
// 查询构建器
DB::table('users')
    ->where('status', 1)
    ->orWhere('age', '>', 18)
    ->get();
// Eloquent模型
User::where('status', 1)->orWhere('age', '>', 18)->get();
// 对应SQL
SELECT * FROM `users` WHERE `status` = 1 OR `age` > 18;
```

3. 嵌套条件（AND + OR）

```
// 查询构建器
DB::table('users')
    ->where(function ($query) {
        $query->where('status', 1)->where('age', '>', 18);
```

```
    })
    ->orWhere(function ($query) {
        $query->where('level', '>', 3)->where('vip', 1);
    })
    ->get();
// Eloquent模型
User::where(function ($query) {
        $query->where('status', 1)->where('age', '>', 18);
    })
    ->orWhere(function ($query) {
        $query->where('level', '>', 3)->where('vip', 1);
    })
    ->get();
// 对应SQL
SELECT * FROM `users`
WHERE (`status` = 1 AND `age` > 18)
OR (`level` > 3 AND `vip` = 1);
```

三、联表查询（JOIN）

1. INNER JOIN

```
// 查询构建器
DB::table('articles')
    ->join('users', 'articles.user_id', '=', 'users.id')
    ->select('articles.*', 'users.name as author')
    ->get();
// Eloquent关联（with）
Article::with('user')->get();
// 对应SQL（查询构建器）
SELECT `articles`.*, `users`.`name` as `author`
FROM `articles`
INNER JOIN `users` ON `articles`.`user_id` = `users`.`id`;
// 对应SQL（Eloquent with，预加载）
SELECT * FROM `articles`;
SELECT * FROM `users` WHERE `users`.`id` IN (1, 2, 3, ...); // 根据文章user_id查询
```

2. LEFT JOIN

```
// 查询构建器
DB::table('users')
   ->leftJoin('profiles', 'users.id', '=', 'profiles.user_id')
   ->select('users.*', 'profiles.phone')
   ->get();
// 对应SQL
SELECT `users`.*, `profiles`.`phone`
FROM `users`
LEFT JOIN `profiles` ON `users`.`id` = `profiles`.`user_id`;
```

四、聚合查询

1. 统计总数

```
// 查询构建器
DB::table('users')->count();
// Eloquent模型
User::count();
// 对应SQL
SELECT COUNT(*) AS aggregate FROM `users`;
```

2. 最大值 / 最小值 / 平均值

```
// 查询构建器
DB::table('users')->max('age');
DB::table('users')->min('age');
DB::table('users')->avg('age');
// Eloquent模型
User::max('age');
User::min('age');
User::avg('age');
// 对应SQL
SELECT MAX(`age`) AS aggregate FROM `users`;
```

```
SELECT MIN(`age`) AS aggregate FROM `users`;
SELECT AVG(`age`) AS aggregate FROM `users`;
```

五、排序与分页

1. 排序（ORDER BY）

```
// 查询构建器
DB::table('users')->orderBy('created_at', 'desc')->get();
// Eloquent模型
User::orderBy('created_at', 'desc')->get();
// 对应SQL
SELECT * FROM `users` ORDER BY `created_at` DESC;
```

2. 分页（LIMIT + OFFSET）

```
// 查询构建器
DB::table('users')->paginate(10);
// Eloquent模型
User::paginate(10);
// 对应SQL（分页第一页）
SELECT * FROM `users` LIMIT 10 OFFSET 0;
// 分页第二页
SELECT * FROM `users` LIMIT 10 OFFSET 10;
```

六、分组与过滤（GROUP BY + HAVING）

```
// 查询构建器
DB::table('articles')
    ->select('user_id', DB::raw('COUNT(*) as article_count'))
    ->groupBy('user_id')
    ->having('article_count', '>', 5)
    ->get();
// Eloquent模型
Article::select('user_id', DB::raw('COUNT(*) as article_count'))
```

```
    ->groupBy('user_id')
    ->having('article_count', '>', 5)
    ->get();
// 对应SQL
SELECT `user_id`, COUNT(*) as article_count
FROM `articles`
GROUP BY `user_id`
HAVING `article_count` > 5;
```

七、存在性查询（EXISTS）

```
// 查询构建器
DB::table('users')
    ->whereExists(function ($query) {
        $query->select(DB::raw(1))
            ->from('articles')
            ->whereRaw('articles.user_id = users.id');
    })
    ->get();
// Eloquent模型
User::whereExists(function ($query) {
        $query->select(DB::raw(1))
            ->from('articles')
            ->whereRaw('articles.user_id = users.id');
    })
    ->get();
// 对应SQL
SELECT * FROM `users`
WHERE EXISTS (
    SELECT 1 FROM `articles` WHERE articles.user_id = users.id
);
```

八、子查询

```
// 查询构建器
$subQuery = DB::table('articles')->select('user_id')->where('status', 1);
$users = DB::table('users')->whereIn('id', $subQuery)->get();
```

```
// Eloquent模型
$subQuery = Article::select('user_id')->where('status', 1);
$users = User::whereIn('id', $subQuery)->get();
// 对应SQL
SELECT * FROM `users`
WHERE `id` IN (
  SELECT `user_id` FROM `articles` WHERE `status` = 1
);
```

九、原生 SQL 片段

```
// 查询构建器
DB::table('users')
  ->whereRaw('YEAR(created_at) = ?', [2024])
  ->orderByRaw('RAND()')
  ->limit(10)
  ->get();
// Eloquent模型
User::whereRaw('YEAR(created_at) = ?', [2024])
  ->orderByRaw('RAND()')
  ->limit(10)
  ->get();
// 对应SQL
SELECT * FROM `users`
WHERE YEAR(created_at) = 2024
ORDER BY RAND()
LIMIT 10;
```

核心技巧：查看实际执行的 SQL

通过`toSql()`方法可获取查询对应的 SQL 语句（不执行查询）：

```
// 查看SQL
$sql = User::where('status', 1)->orderBy('created_at', 'desc')->toSql();
echo $sql; // 输出：SELECT * FROM `users` WHERE `status` = ? ORDER BY `created_at`
DESC
// 查看绑定参数
```

```
$bindings = User::where('status', 1)->getBindings();
print_r($bindings); // 输出：[1]
```

以上覆盖了`Illuminate\Database`的主流查询方式，通过对比原始 SQL 可更清晰理解 ORM / 查询构建器的底层逻辑，便于调试和优化性能。

> （注：文档部分内容可能由 AI 生成）