



Mysql JOIN 的高阶使用

装睡鹿先生

2025-01-16

👁 119

🕒 阅读7分钟

📁 专栏：SQL

智能总结 ⓘ

复制

重新生成

文章主要介绍了 Mysql 中 Inner Join、Left Join、Right Join、Full Join 与 Union 的高阶用法。包括多表连接、与复杂条件结合、处理结果集、与子查询结合、模拟 Full Join 等，还通过丰富的 SQL 示例展示了在不同场景下的应用，强调掌握这些用法能高效处理复杂数据需求，提升数据处理效率。

关联问题: [如何优化JOIN效率](#) [Union能否合并多表](#) [Full Join替代方案是啥](#)

基于该文章内容继续向AI提问

在数据库操作中，JOIN 操作和 UNION 操作是数据关联与合并的重要工具。它们不仅有基础的使用方式，还存在许多高阶技巧，能帮助我们更高效地处理复杂的数据需求。接下来，让我们深入探究 Inner Join、Left Join、Right Join、Full Join 与 Union 的高阶使用，并通过丰富的 SQL 示例来加深理解。

一、Inner Join 的高阶用法

Inner Join 用于返回两个表中连接字段匹配的行，是最常用的连接方式之一。其高阶用法主要体现在多表连接以及与复杂条件的结合上。

1. 多表 Inner Join

假设有三个表：orders（订单表）包含 order_id、customer_id、order_date；customers（客户表）包含 customer_id、customer_name；products（产品表）包含 product_id、product_name、price；order_items（订单项表）包含 order_id、product_id、quantity。

现在要找出每个订单的详细信息，包括客户名称、产品名称、价格和数量，SQL 语句如下：

▼ vbnet 代码解读 复制代码

```
1  SELECT
2      o.order_id,
3      c.customer_name,
4      p.product_name,
5      p.price,
6      oi.quantity
7  FROM
8      orders o
9  JOIN
10     customers c ON o.customer_id = c.customer_id
11  JOIN
12     order_items oi ON o.order_id = oi.order_id
13  JOIN
14     products p ON oi.product_id = p.product_id;
```

通过这种多表 Inner Join，可以将多个相关表的数据整合在一起，满足复杂的业务查询需求。

2. Inner Join 与复杂条件

有时，我们不仅需要基于连接字段进行匹配，还需要其他复杂条件。例如，找出订单日期在特定时间段内且产品价格高于某个值的订单详细信息：



装睡鹿先生

java @rh

85

文章

6.3k

阅读

31

粉丝

关注

私信

目录

收起 ^

2. Right Join 与复杂逻辑结合

四、Full Join 的高阶用法

1. 模拟 Full Join

2. Full Join 处理复杂业务场景

五、Union 的高阶用法

1. Union 与结果集筛选

2. Union 与排序

3. Union 与其他操作结合

六、总结

相关推荐

避开陷阱：Java 中影响 finally 块执行的...

91阅读 · 3点赞

MVCC如何应对MySQL并发问题

75阅读 · 3点赞

JVM类加载器(附面试题)

48阅读 · 0点赞

InnoDB 索引与 Online DDL 的结合：业...

78阅读 · 0点赞

面试官：JDBC 是如何打破双亲委派模...

347阅读 · 7点赞

精选内容

程序员的保命技能——流程编排，你一定...

五阳 · 3.0k阅读 · 27点赞

Zookeeper （42） Zookeeper的启动命...

Victor356 · 29阅读 · 0点赞

Zookeeper （41） 如何安装和配置Zook...

Victor356 · 30阅读 · 0点赞

kubernetes 核心技术-Controller 控制器

難釋懷 · 27阅读 · 1点赞

kubernetes 核心技术-Label

難釋懷 · 26阅读 · 1点赞

找对属于你的技术圈子

回复「进群」加入官方微信群



AI
助手

▼

vbnet

[代码解读](#)

[复制代码](#)

```
1  SELECT
2      o.order_id,
3      c.customer_name,
4      p.product_name,
5      p.price,
6      oi.quantity
7  FROM
8      orders o
9  JOIN
10     customers c ON o.customer_id = c.customer_id
11 JOIN
12     order_items oi ON o.order_id = oi.order_id
13 JOIN
14     products p ON oi.product_id = p.product_id
15 WHERE
16     o.order_date BETWEEN '2023 - 01 - 01' AND '2023 - 06 - 30'
17     AND p.price > 100;
```

这里结合了 BETWEEN 条件，进一步筛选出符合条件的数据。

二、Left Join 的高阶用法

Left Join 返回左表中的所有行以及右表中连接字段匹配的行。高阶应用中，常涉及对结果集的处理以及与子查询的结合。

1. Left Join 与结果集处理

假设我们要统计每个客户的订单数量，包括没有下过订单的客户，SQL 语句如下：

▼

vbnet

[代码解读](#)

[复制代码](#)

```
1  SELECT
2      c.customer_id,
3      c.customer_name,
4      COUNT(o.order_id) AS order_count
5  FROM
6      customers c
7  LEFT JOIN
8      orders o ON c.customer_id = o.customer_id
9  GROUP BY
10     c.customer_id, c.customer_name;
```

通过 Left Join，即使某个客户没有订单，在结果集中也会有对应的记录，order_count 为 0。

2. Left Join 与子查询

假设我们有一个 high_value_orders 子查询，用于找出订单总金额大于 1000 的订单。现在要找出每个客户的信息以及他们的高价值订单数量，SQL 语句如下：

▼

vbnet

[代码解读](#)

[复制代码](#)

```
1  SELECT
2      c.customer_id,
3      c.customer_name,
4      COUNT(hvo.order_id) AS high_value_order_count
5  FROM
6      customers c
7  LEFT JOIN
8      (
9          SELECT
10             order_id,
11             customer_id
12          FROM
13             orders o
14          JOIN
15             order_items oi ON o.order_id = oi.order_id
16          JOIN
17             products p ON oi.product_id = p.product_id
18          GROUP BY
19             o.order_id
20          HAVING
21             SUM(p.price * oi.quantity) > 1000
22      ) hvo ON c.customer_id = hvo.customer_id
23  GROUP BY
24     c.customer_id, c.customer_name;
```

这里通过 Left Join 将客户表与子查询结果进行连接，统计每个客户的高价值订单数量。

三、Right Join 的高阶用法

Right Join 与 Left Join 相反，返回右表中的所有行以及左表中连接字段匹配的行。虽然使用频率相对较低，但在特定场景下也有重要作用。

1. Right Join 用于特定数据筛选

假设我们有一个 special_products 表，记录了一些特殊产品，现在要找出购买了这些特殊产品的客户信息以及订单详情，SQL 语句如下：

▼

sql

🔗代码解读 复制代码

```
1  SELECT
2      c.customer_name,
3      o.order_date,
4      sp.product_name
5  FROM
6      special_products sp
7  RIGHT JOIN
8      order_items oi ON sp.product_id = oi.product_id
9  RIGHT JOIN
10     orders o ON oi.order_id = o.order_id
11  RIGHT JOIN
12     customers c ON o.customer_id = c.customer_id;
```

通过 Right Join，以特殊产品表为基准，找出与之相关的订单和客户信息。

2. Right Join 与复杂逻辑结合

在某些复杂业务场景中，可能需要结合多种条件和函数。例如，找出购买了特殊产品且订单金额在一定范围内的客户信息，并计算每个客户的平均订单金额：

▼

sql

🔗代码解读 复制代码

```
1  SELECT
2      c.customer_id,
3      c.customer_name,
4      AVG(p.price * oi.quantity) AS average_order_amount
5  FROM
6      special_products sp
7  RIGHT JOIN
8      order_items oi ON sp.product_id = oi.product_id
9  RIGHT JOIN
10     orders o ON oi.order_id = o.order_id
11  RIGHT JOIN
12     customers c ON o.customer_id = c.customer_id
13  WHERE
14      p.price * oi.quantity BETWEEN 500 AND 2000
15  GROUP BY
16      c.customer_id, c.customer_name;
```

这里结合了 BETWEEN 条件和聚合函数 AVG，实现复杂的数据筛选和计算。

四、Full Join 的高阶用法

Full Join 返回左表和右表中的所有行。当某行在另一表中没有匹配项时，对应列显示 NULL。由于 MySQL 不直接支持 FULL JOIN，但可以通过 LEFT JOIN 和 UNION 模拟实现。

1. 模拟 Full Join

假设有两个表 table1 和 table2，要实现它们的 Full Join，SQL 语句如下：

▼

vbnet

🔗代码解读 复制代码

```
1  SELECT
2      t1.column1,
3      t1.column2,
4      t2.column3
5  FROM
6      table1 t1
7  LEFT JOIN
8      table2 t2 ON t1.key = t2.key
9  UNION
10 SELECT
11     t1.column1,
12     t1.column2,
13     t2.column3
14  FROM
15     table2 t2
16  LEFT JOIN
17     table1 t1 ON t2.key = t1.key
18  WHERE
19     t1.key IS NULL;
```

前半部分通过 LEFT JOIN 从 table1 出发获取数据，后半部分通过 LEFT JOIN 从 table2 出发获取 table1 中没有匹配项的数据，再通过 UNION 合并结果。

2. Full Join 处理复杂业务场景

在处理复杂业务数据时，Full Join 可以用于整合多个数据源的信息，确保所有数据都被包含在内。例如，有两个部门的员工表 department1_employees 和 department2_employees，要合并两个部门的员工信息，包括可能存在的重复员工（以员工 ID 区分），并显示每个员工所属部门：

▼

sql

代码解读 复制代码

```
1  SELECT
2      e1.employee_id,
3      e1.employee_name,
4      'Department 1' AS department
5  FROM
6      department1_employees e1
7  LEFT JOIN
8      department2_employees e2 ON e1.employee_id = e2.employee_id
9  UNION
10 SELECT
11     e2.employee_id,
12     e2.employee_name,
13     'Department 2' AS department
14 FROM
15     department2_employees e2
16 LEFT JOIN
17     department1_employees e1 ON e2.employee_id = e1.employee_id
18 WHERE
19     e1.employee_id IS NULL;
```

这样可以完整地显示两个部门的所有员工信息以及所属部门。

五、Union 的高阶用法

Union 用于合并两个或多个 SELECT 语句的结果集。其高阶用法主要体现在结果集的筛选、排序以及与其他操作的结合上。

1. Union 与结果集筛选

假设我们有两个查询，分别获取不同年龄段的用户信息，现在要合并结果并筛选出特定城市的用户，SQL 语句如下：

▼

sql

代码解读 复制代码

```
1  (
2      SELECT
3          user_id,
4          user_name,
5          age,
6          city
7      FROM
8          users
9      WHERE
10         age BETWEEN 18 AND 30
11 )
12 UNION
13 (
14     SELECT
15         user_id,
16         user_name,
17         age,
18         city
19     FROM
20         users
21     WHERE
22         age BETWEEN 31 AND 50
23 )
24 WHERE
25     city = 'New York';
```

通过 Union 合并两个年龄段的用户信息，再通过 WHERE 子句筛选出纽约的用户。

2. Union 与排序

有时，我们需要对合并后的结果集进行排序。例如，合并两个产品表的信息，并按价格从高到低排序：

▼

sql

代码解读 复制代码

```
1  (
2      SELECT
3          product_id,
4          product_name,
5          price
6      FROM
7          products1
8  )
9  UNION
10 (
11     SELECT
12         product_id,
13         product_name,
14         price
15     FROM
16         products2
17 )
18 ORDER BY
19     price DESC;
```

这样可以确保合并后的结果集按价格降序排列。

3. Union 与其他操作结合

Union 可以与 JOIN 等操作结合使用，以满足更复杂的业务需求。例如，先通过 Inner Join 获取订单与客户
的关联信息，再通过 Union 合并一些特殊订单信息，并按订单日期排序：

▼ vbnet 代码解读 复制代码

```
1  (  
2      SELECT  
3          o.order_id,  
4          c.customer_name,  
5          o.order_date  
6      FROM  
7          orders o  
8      INNER JOIN  
9          customers c ON o.customer_id = c.customer_id  
10 )  
11 UNION  
12 (  
13     SELECT  
14         special_order_id AS order_id,  
15         special_customer_name AS customer_name,  
16         special_order_date AS order_date  
17     FROM  
18         special_orders  
19 )  
20 ORDER BY  
21     order_date ASC;
```

通过这种方式，可以将常规订单和特殊订单信息整合在一起，并按日期顺序展示。

六、总结

Inner Join、Left Join、Right Join、Full Join 与 Union 在数据库操作中具有强大的功能。掌握它们的高阶用法，能够帮助我们更加灵活、高效地处理各种复杂的数据需求。无论是多表连接、结果集处理，还是与其他操作和条件的结合，这些技巧都能让我们在数据库开发中更加得心应手。希望大家在实际工作中多多运用这些技巧，优化自己的 SQL 代码，提升数据处理效率。

在实际应用中，你可能还会遇到更多独特的业务场景，需要根据具体情况灵活运用这些方法。如果在使用过程中有任何新的发现或问题，欢迎分享交流，让我们一起在数据库的探索中不断进步。

标签： MySQL MyBatis Java 话题： 2024年终总结

本文收录于以下专栏



SQL 专栏目录


sql基本使用

3 订阅 · 8 篇文章

订阅

上一篇 MySQL 中 IN 和 BETWEEN 的高阶... 下一篇 SQL 独特约束与主键约束：高阶玩...

评论 0



登录 / 注册

即可发布评论!



暂无评论数据

为你推荐

子查询和JOIN的用法和区别

云原生melo荣 | 1年前 | 982 | 1 | 评论

后端 面试 数据库

前端也该知道，除了 select 、 from 、 where 之外的另外几个重要的数据库操作

掘金安东尼 | 2年前 | 1.8k | 46 | 2

前端 数据库 数据分析

SQL-JOIN全解析

TrueDei | 3年前 | 161 | 2 | 评论

后端

股票、指数、快照、逐笔... 不同行情数据源的实时关联分析应用

DolphinDB | 1年前 | 1.1k | 1 | 评论

后端

【MySQL必知必会】：联结表

言羽 | 1年前 | 454 | 1 | 评论

掘金·日新计划

MySQL的JOIN到底是怎么玩的

码上遇见你 | 11月前 | 326 | 2 | 评论

后端 面试 程序员

MySQL之连接原理

阿布 | 2年前 | 638 | 1 | 评论

后端 MySQL

MySQL以其他表作为条件更新指定表

ABS_Plastic | 9月前 | 759 | 1 | 评论

MySQL

JOIN 查询性能优化手段 - Runtime Filter

destiny1020 | 1年前 | 1.2k | 3 | 评论

数据库

mysql基础——在多张表格中检索数据

史呆芬 | 1年前 | 1.7k | 2 | 评论

后端

【MySQL】MySQL 连接表的几种方式 JOIN 和 UNION 的区别

sleepyhermit | 1年前 | 310 | 2 | 评论

MySQL

SQL职场必备：掌握数据库技能提升职场竞争力

海拥 | 7月前 | 139 | 1 | 评论

后端

SQL-JOIN 全解析

TrueDei | 4年前 | 553 | 3 | 评论

SQL

如何让Join性能更高效？

迟到的微笑 | 2年前 | 983 | 1 | 评论

后端

如何让JOIN跑得更快？

java奥斯卡 | 2年前 | 1.6k | 25 | 评论

后端 SQL 数据库