基于分布式中间件的SQL改造指南

个人介绍

- 孙正方
- 开源分布式中间件DBLE核心研发
- 邮箱：sunzhengfang@actionsky.com

问题的由来

1. 怎么区分SQL能支持？

2. 按照什么原则调整SQL？

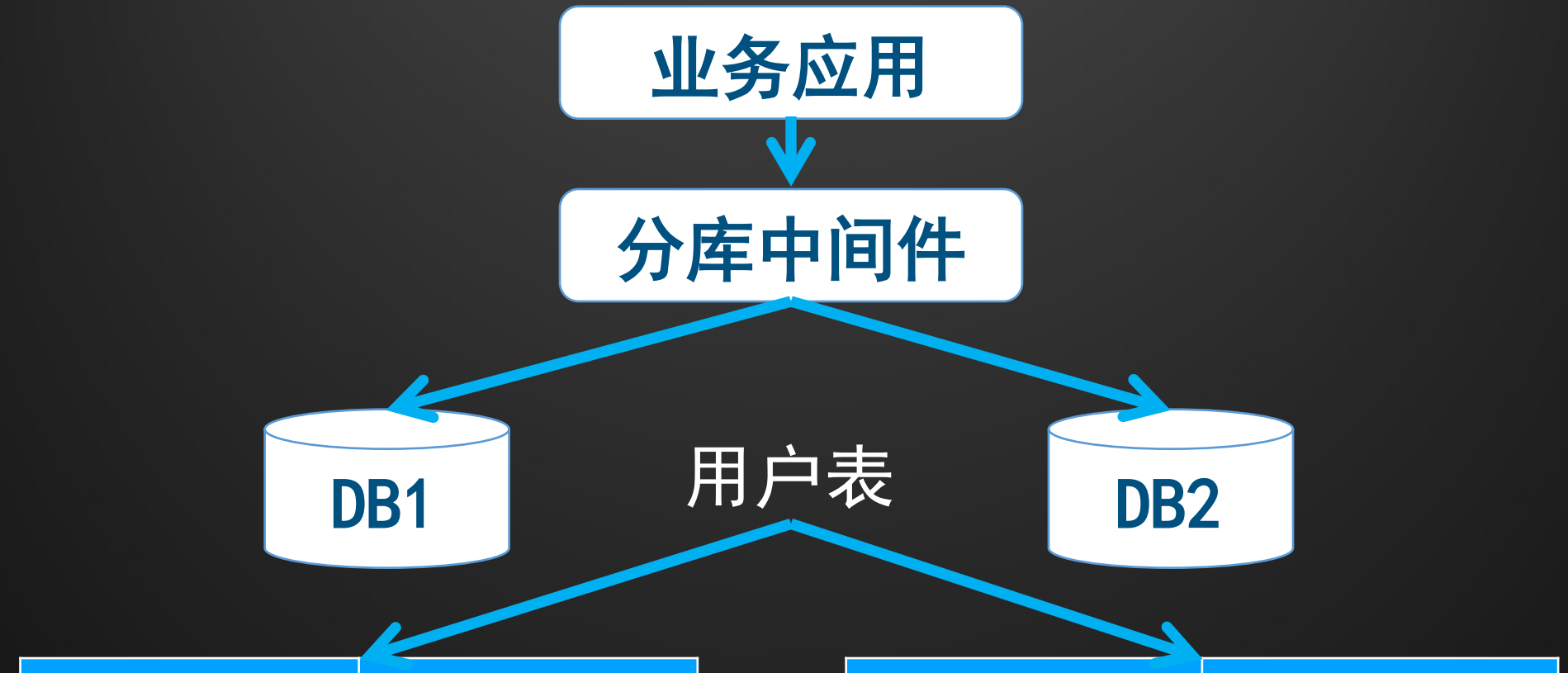
目录

一. 简单SQL处理

1. 中间件如何处理简单SQL?

2. 简单SQL有什么注意要点?

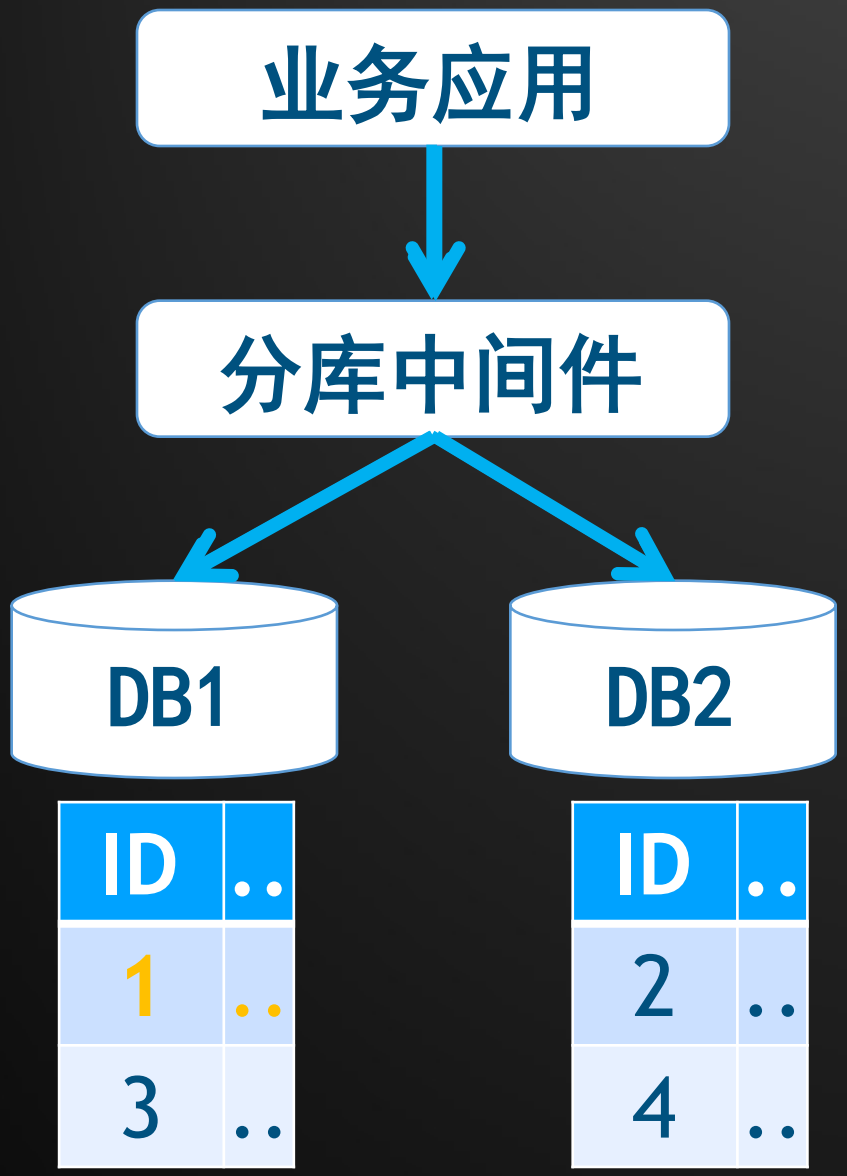
简单SQL - 基本原理



ID	NAME
1	张三
3	王五

ID	NAME
2	李四
4	赵六

简单SQL - 基本原理



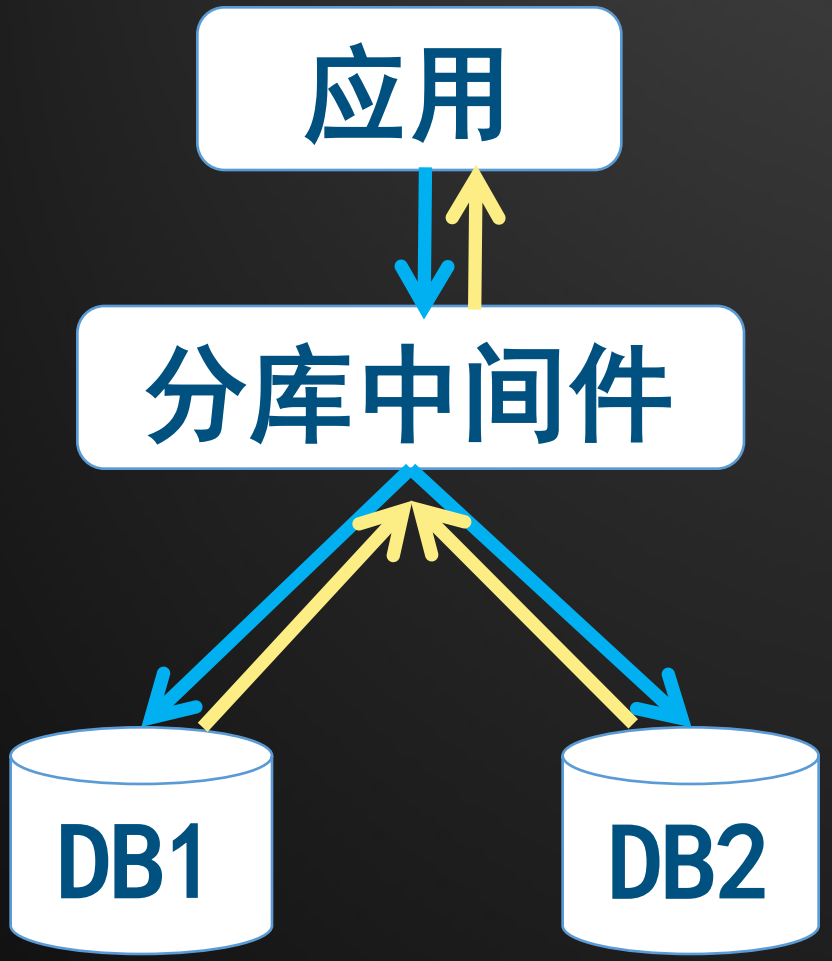
SELECT * FROM 用户表 WHERE ID = 1

用户表 ID =1 -> DB1

DB1:

SELECT * FROM 用户表 WHERE ID = 1

简单SQL - 广播



无分片条件查询：
SELECT * FROM 用户表

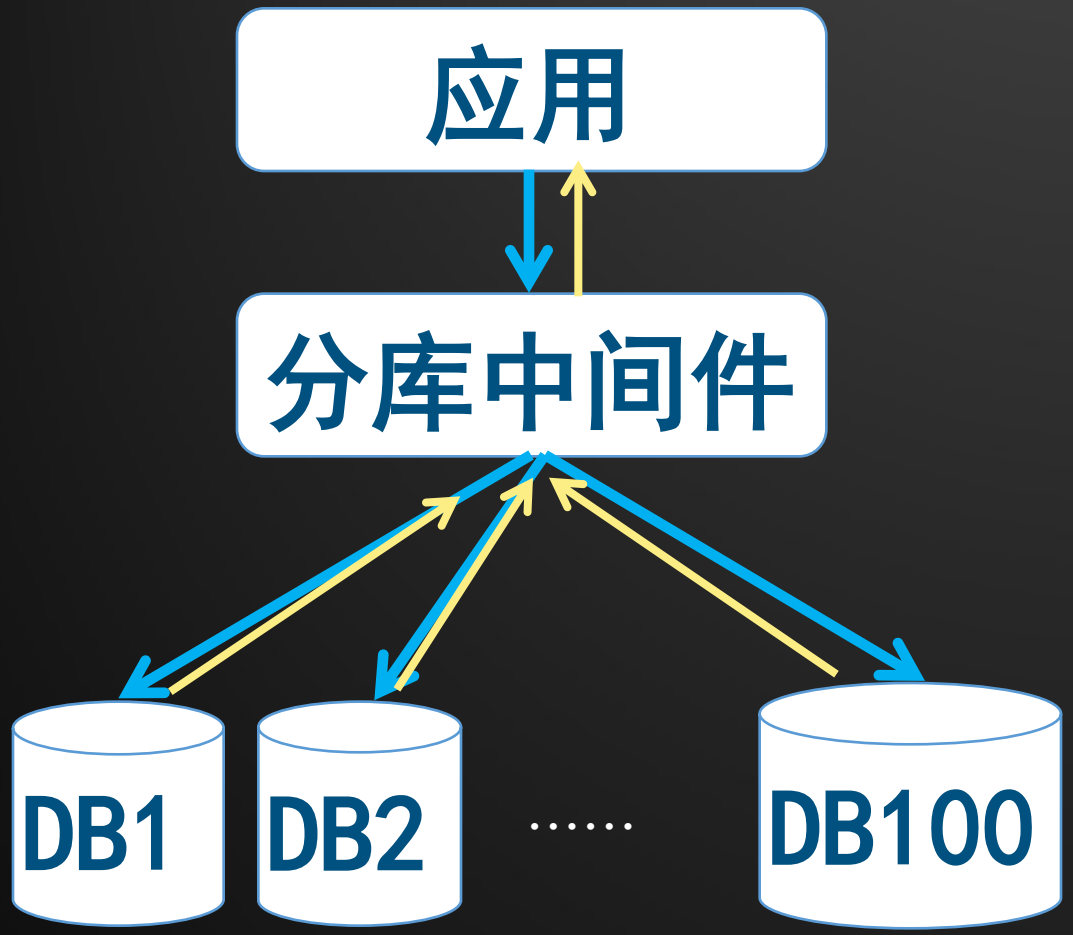
最后一个DB返回结果后
中间件才能向应用返回结果

一. 简单SQL

1. 中间件如何处理简单SQL?

2. 简单SQL有什么注意要点?

简单SQL - 注意事项



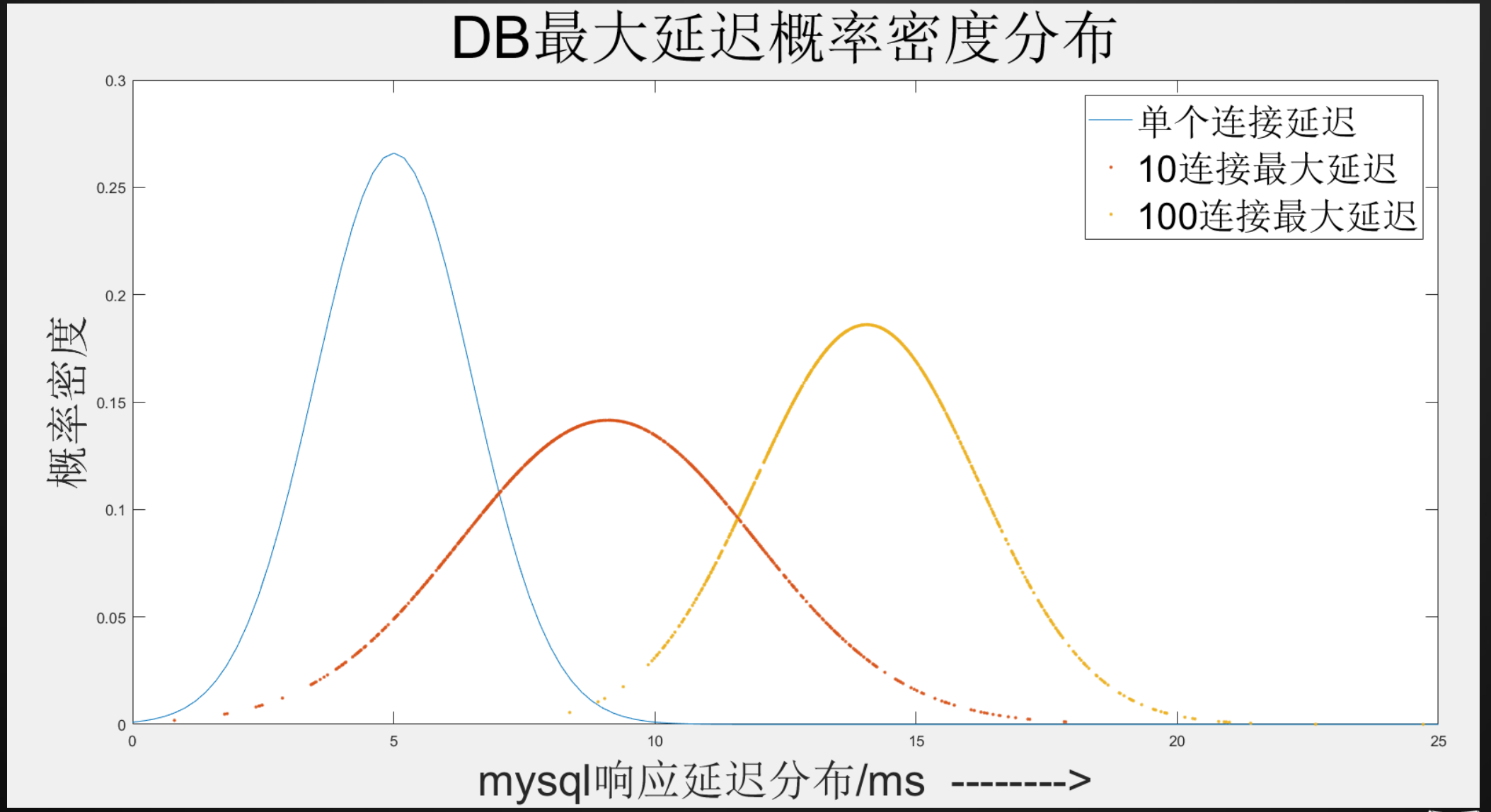
无条件查询：

SELECT * FROM 用户表

用户表分布在DB1-100中

下发 X 100

简单SQL - 注意事项



简单SQL - 注意事项

- 内存：流式返回，无影响
- 延迟：等待最后一个节点

添加分片条件，否则影响延迟

简单SQL-小结

- 中间件中单表查询属于简单SQL
- 广播SQL高并发对延迟有影响
- 使用分片条件查询能降低影响

目录

二. 普通分片表关联

- 中间件怎么处理普通分片表关联?
- 分片表关联有什么注意要点?

分片表关联 – 逻辑关系

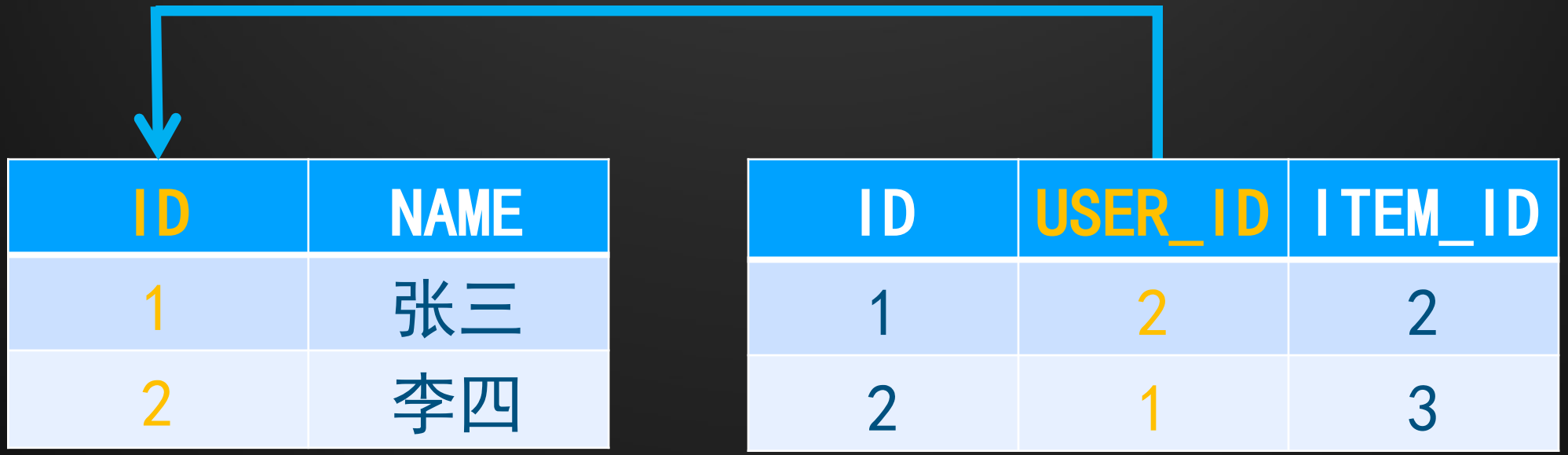
需求：查询一个用户的订单列表

用户表　　订单表

分片表关联 – 逻辑关系

查询一个用户的订单列表 逻辑上存在从属关系

订单 从属于 用户



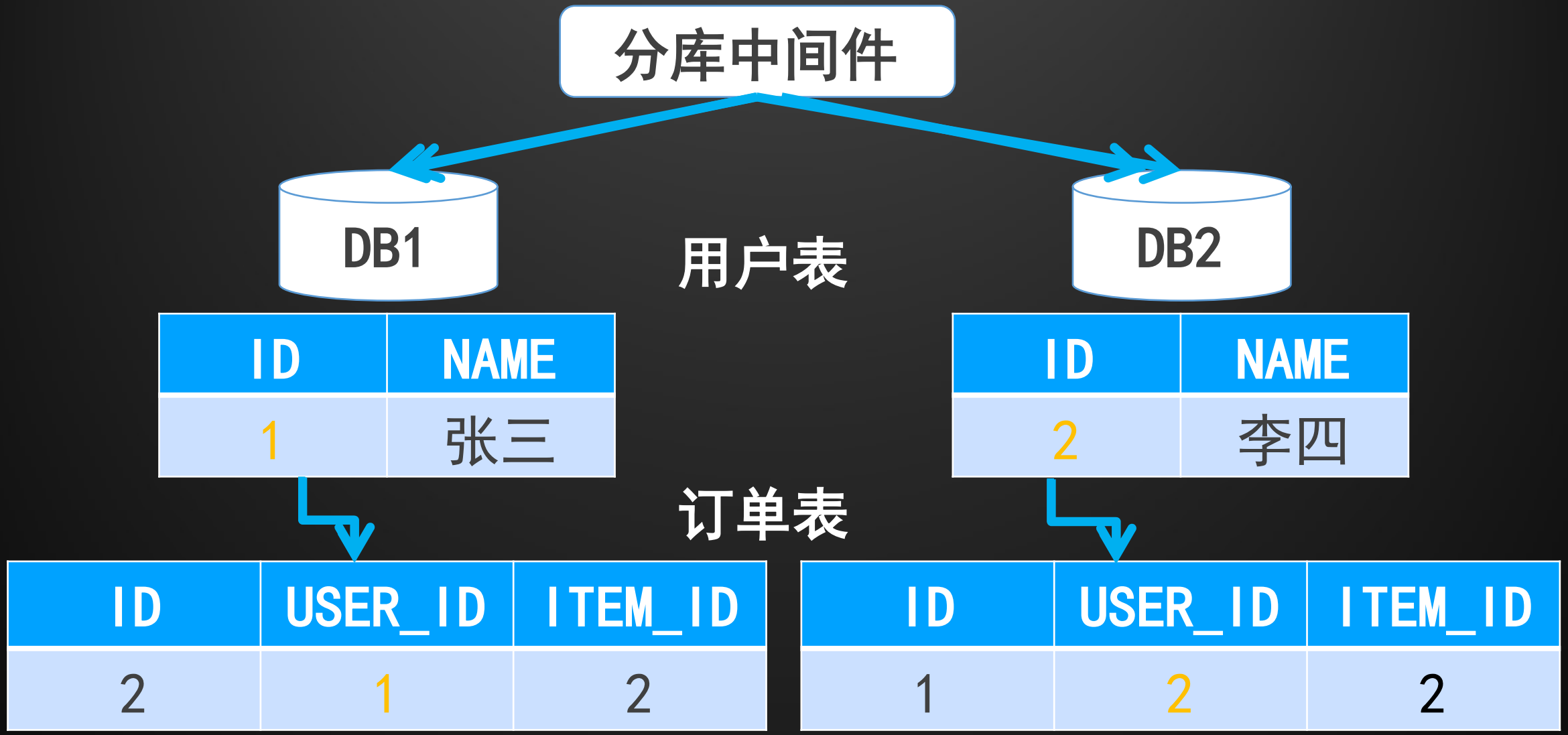
ID	NAME
1	张三
2	李四

用户表

ID	USER_ID	ITEM_ID
1	2	2
2	1	3

订单表

分片表关联 – 存储分布



分片表关联 – ER场景

用户表

ID	NAME
1	张三

订单表

ID	USER_ID	ITEM_ID
2	1	2
...	1	...

. . . . . .

```
SELECT O.*, U.NAME
FROM
      用户表 U,
      订单表 O
WHERE
      O.USER_ID = U.ID
      AND U.ID = 1
```

二. 普通分片表关联

- 中间件怎么处理普通分片表关联?
- 分片表关联有什么注意要点?

分片表关联 – 注意要点

- 确保表格之间存在ER关系
- 下发到MySQL节点执行，广播类型注意延迟

分片表关联 – 小结

- 有从属关系使用ER关系
- ER关系表关联整体下发SQL
- 同样需要注意分片条件问题

如果场景变得更加复杂，或者失去逻辑从属关系会怎么样？

目录

一. 简单SQL处理

二. 普通分片表关联

三. 跨库表关联

- 跨库表关联下发的流程
- 改造注意点

三. 跨库表关联

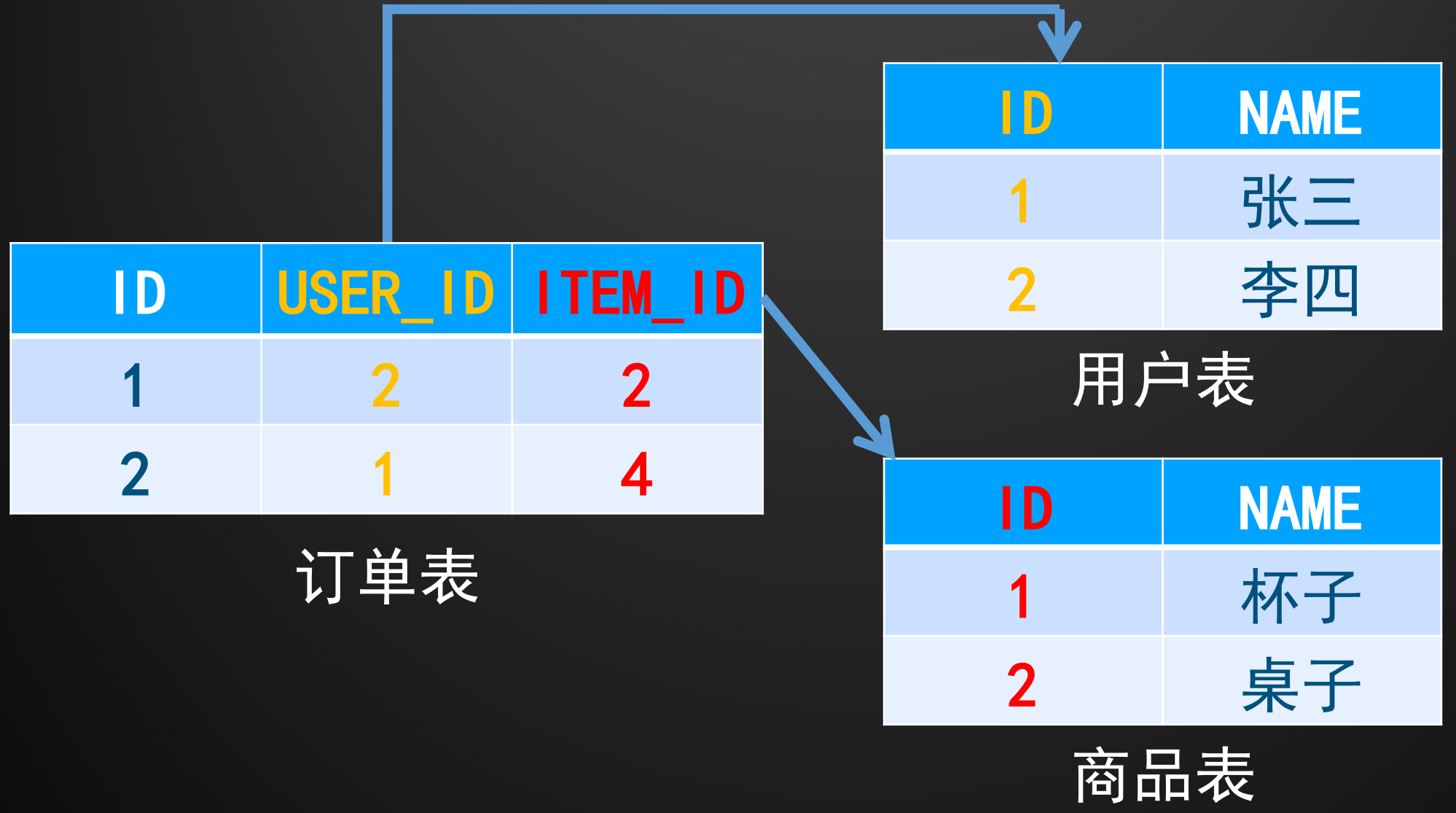
- 中间件怎么处理更复杂的场景？
- 中间件执行跨库表关联注意事项？

跨库表关联 – 复杂场景

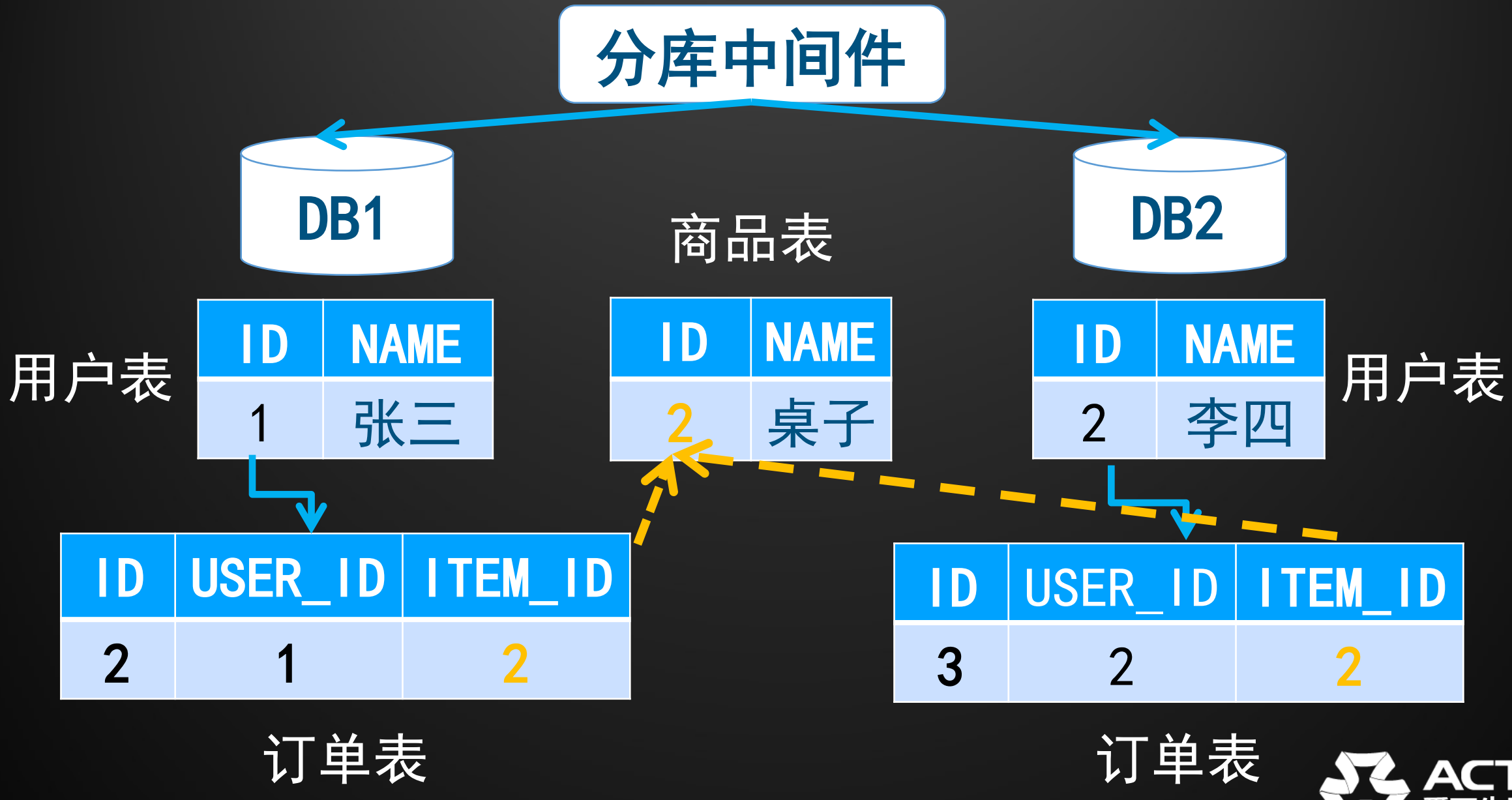
需求：查询一个用户买过的商品列表

用户表　　商品表　　订单表

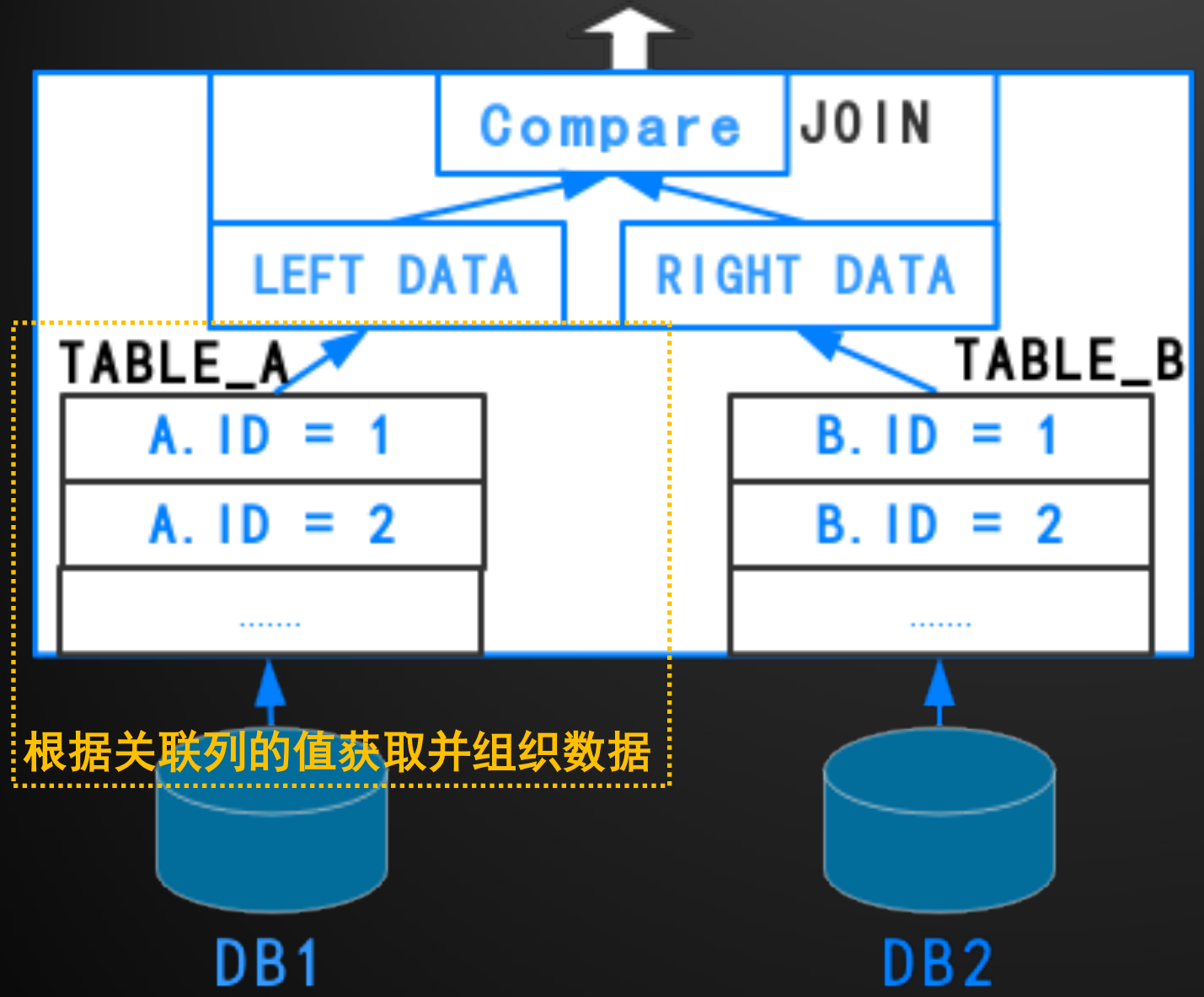
跨库表关联 – 复杂场景



跨库表关联 – 存储分布



跨库表关联 - 跨库关联的实现

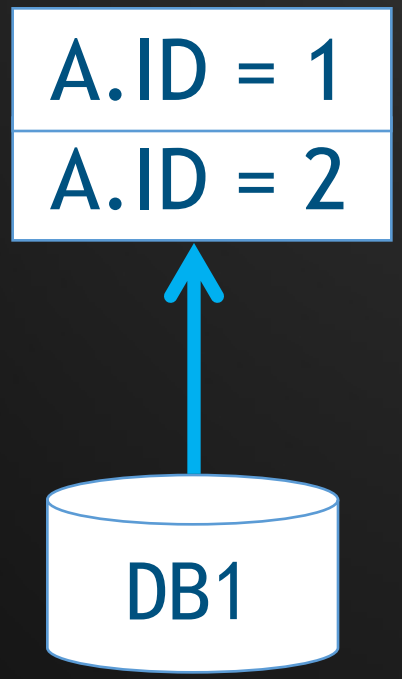



```
SELECT *
FROM
      TABLE_A A,
      TABLE_B B
WHERE
      A.ID = B.ID
```

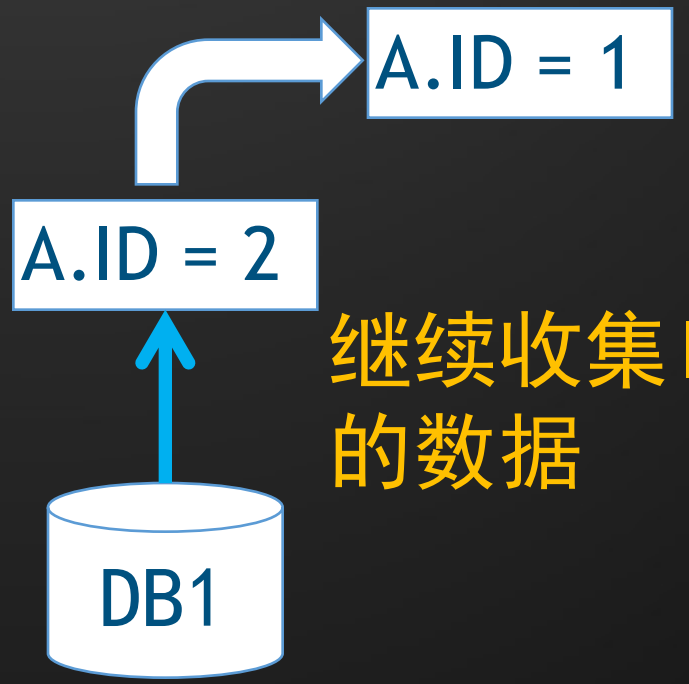
完全按照SQL原有逻辑

跨库表关联 – 跨库关联的实现

出现ID ＞ 1
标志ID = 1
数据收集完毕



ID=1的数据
进入
对比队列



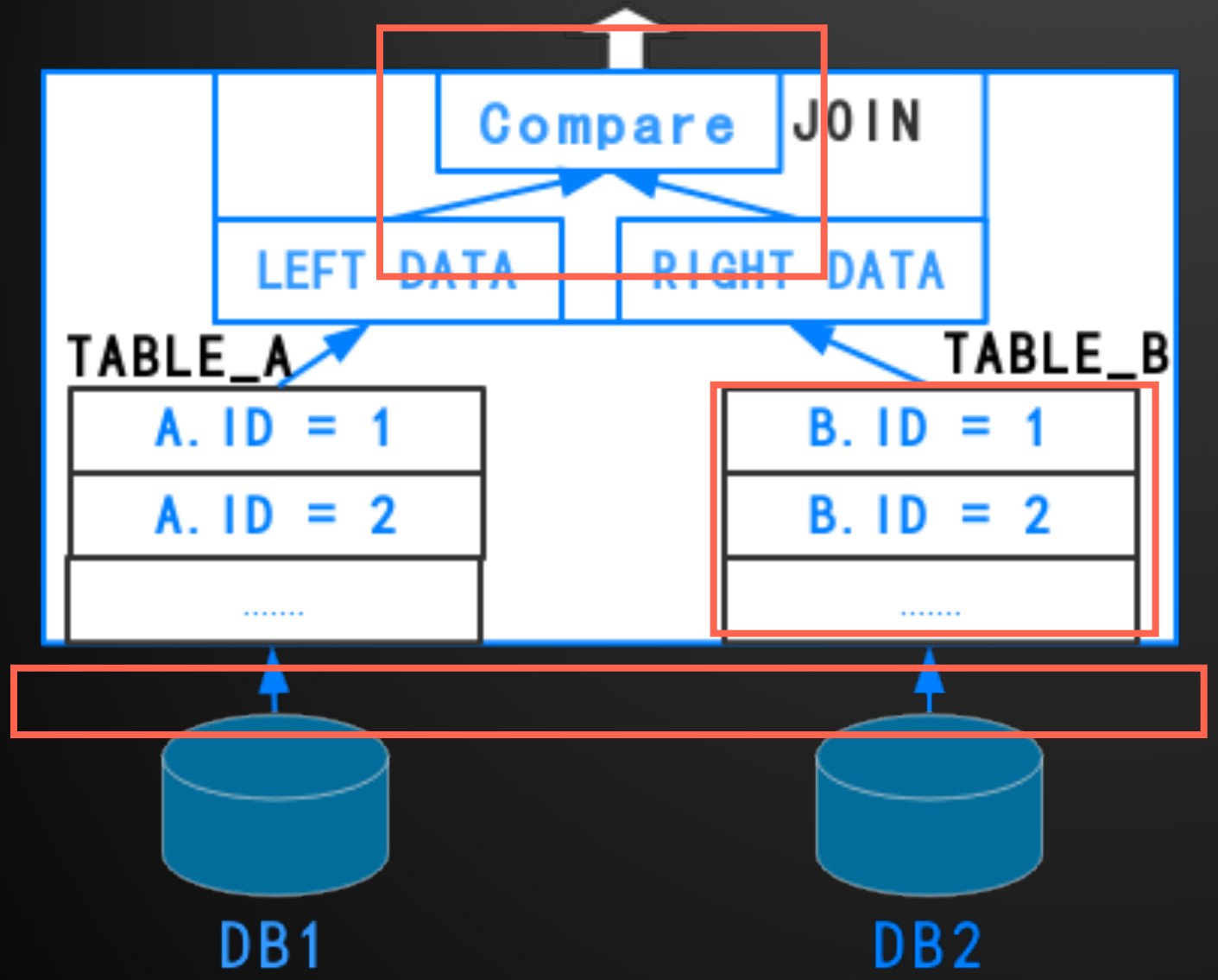
继续收集ID=2
的数据

按照ID排序查询数据

二. 跨库表关联

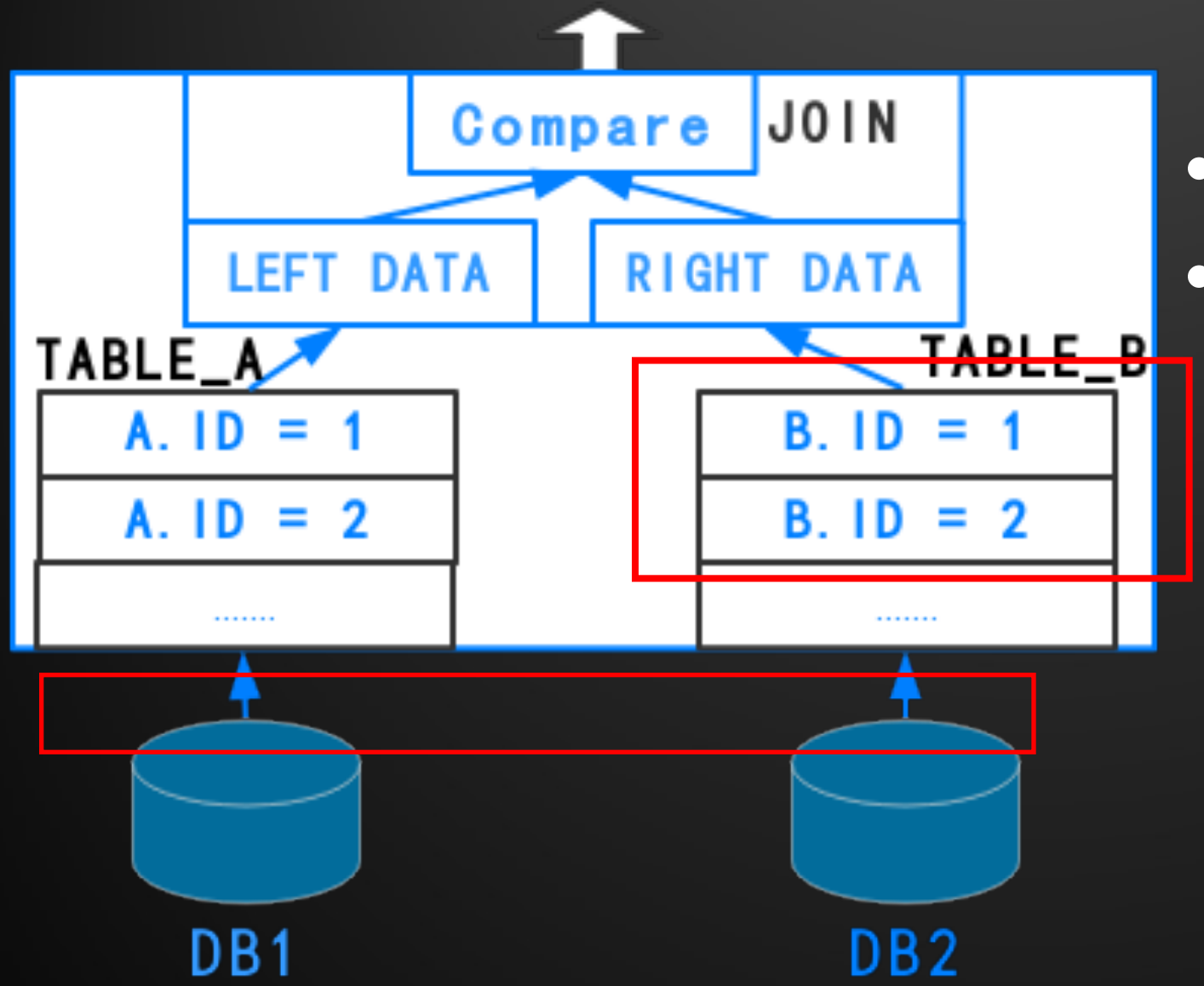
- 中间件怎么处理更复杂的场景？
- 中间件执行跨库表关联注意事项？

跨库表关联 – 资源消耗



- 查询的过程消耗后端连接
- 内存持有中间结果
- 聚合的过程占用CPU资源

跨库表关联 – 资源消耗



- 添加条件降低数据量
- 使用重复性低的列进行JOIN

增加流式返回的效率

跨库表关联 - 亲和性

MySQL:
优化器计算代价调整顺序

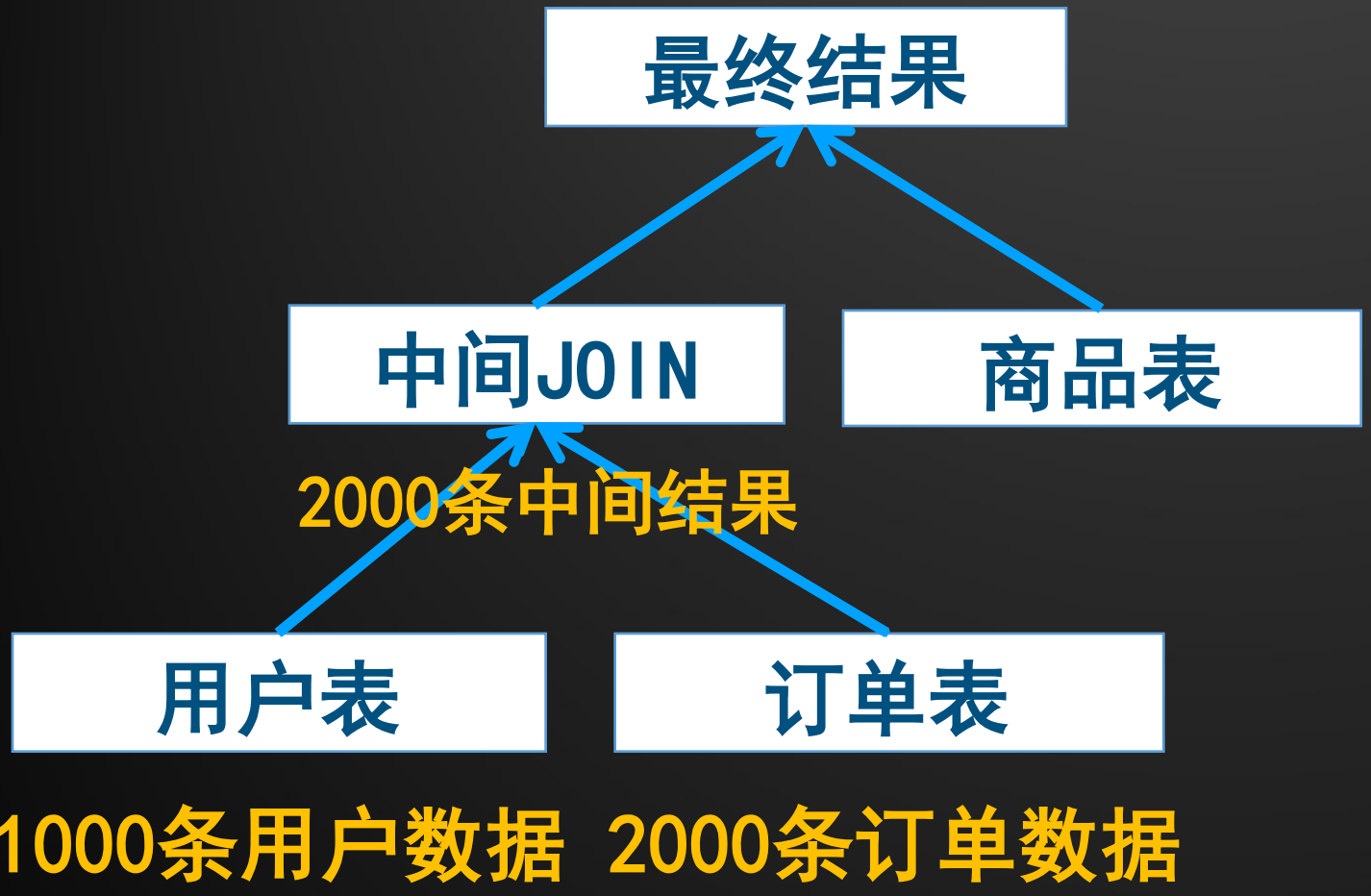
中间件:
不存储数据，无法预知代价

```
SELECT I.*
FROM
        用户表 U,
        商品表 I,
        订单表 O
WHERE
        I.ID = O.ITEM_ID
        AND O.USER_ID = U.ID
```

跨库表关联 - 亲和性

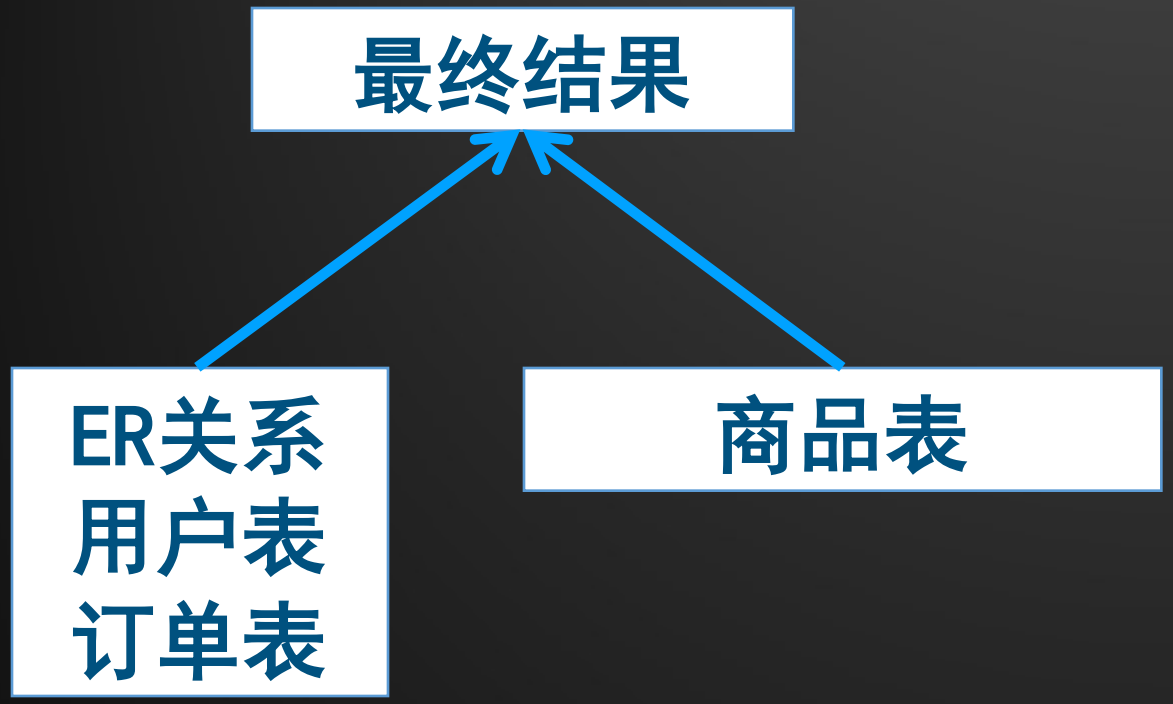

```
SELECT I.*
FROM
        用户表 U,
        商品表 I,
        订单表 O
WHERE
        I.ID = O.ITEM_ID
        AND O.USER_ID = U.ID
```

跨库表关联 - 亲和性




```
SELECT I.*
FROM
      用户表 U,
      订单表 O,
      商品表 I
WHERE
       I.ID = O.ITEM_ID
      AND O.USER_ID = U.ID
```

跨库表关联 - 亲和性




```
SELECT I.*
FROM
        订单表 O,
        用户表 U,
        商品表 I
WHERE
        I.ID = O.ITEM_ID
        AND O.USER_ID = U.ID
```

用户和订单存在ER关系，直接下发效率最高

两次中间件JOIN降低为1次

跨库表关联 - explain

```
mysql> explain select * from test_a a,test_b b,test_c c where a.id = b.id and b.id = c.id;
+-----------------+---------------+-----------------------------------------------------------+
| DATA_NODE       | TYPE          | SQL/REF                                                   |
+-----------------+---------------+-----------------------------------------------------------+
| dn1_0           | BASE SQL      | select `a`.`id` from  `test_a` `a` order by `a`.`id` ASC  |
| merge_1         | MERGE         | dn1_0                                                     |
| dn2_0           | BASE SQL      | select `b`.`id` from  `test_b` `b` order by `b`.`id` ASC  |
| merge_2         | MERGE         | dn2_0                                                     |
| join_1          | JOIN          | merge_1; merge_2                                          |
| order_1         | ORDER         | join_1                                                    |
| shuffle_field_1 | SHUFFLE_FIELD | order_1                                                   |
| dn3_0           | BASE SQL      | select `c`.`id` from  `test_c` `c` order by `c`.`id` ASC  |
| merge_3         | MERGE         | dn3_0                                                     |
| join_2          | JOIN          | shuffle_field_1; merge_3                                  |
| shuffle_field_2 | SHUFFLE_FIELD | join_2                                                    |
+-----------------+---------------+-----------------------------------------------------------+
11 rows in set (0.01 sec)
```

跨库表关联 - 小结

1. 按SQL本来的逻辑结构分别取数据执行
2. 按照JOIN列组织数据进行JOIN
3. SQL编写需使得中间结果尽量少
   - 表限定条件
   - 调整亲和性
   - JOIN列重复低

SQL改造原则 – 总结

1. 简单查询以及ER查询：
   - 分片条件
   - 使用ER关系
2. 跨库JOIN：
   - 表格给分片条件，限定条件
   - 注意亲和性，优化关联顺序
   - 降低JOIN列的重复率

推荐:DBLE公开课

DBLE系列公开课	1.DBLE的基本使用使用	2.DBLE的高级特性	3.DBLE的进阶使用
	1.1 DBLE概述 1.2 DBLE的配置 1.3 DBLE的管理端口	2.1 分布式特性 2.2 后端数据库相关特性 2.3 可靠性/可运维功能 2.4 MySQL的兼容特性	3.1 SQL性能分析 3.2 故障分析 3.3 聊聊性能测试 3.4 运维实施
	4.终章： 现状 / Roadmap / Q&A		

关于开源分布式中间件DBLE

加入开源分布式中间件交流群

- 获取12节**分布式原理解析视频**
- 每天8:30-20:30**即时疑问解答**
- 最新分布式中间件**发版资讯**
- **用户见面会早知道**

技术咨询

地区：华南地区

主题：MySQL数据库自动化运维管理

联系：曹勇 18503063188

Thank You