06 | 外键和连接：如何做关联查询？

2021-03-20 朱晓峰

MySQL 必知必会

进入课程 >

讲述：朱晓峰

时长 12:51　大小 11.78M

你好，我是朱晓峰。今天我来和你聊一聊关联查询的问题。

在实际的数据库应用开发过程中，我们经常需要把 2 个或 2 个以上的表进行关联，以获取需要的数据。这是因为，为了提高存取效率，我们会把不同业务模块的信息分别存放在不同的表里面。但是，从业务层面上看，我们需要完整全面的信息为经营决策提供数据支撑。

就拿咱们的超市项目来说，数据库里面的销售流水表一般只保存销售必需的信息（比如商品编号、数量、价格、金额和会员卡号等）。但是，在呈现给超市经营者的统计报表里面，只包括这些信息是不够的，比如商品编号、会员卡号，这些数字经营者就看不懂。因此，必须要从商品信息表提取出商品信息，从会员表中提取出会员的相关信息，这样才能

形成一个完整的报表。**这种把分散在多个不同的表里的数据查询出来的操作，就是多表查询。**

不过，这种查询可不简单，我们需要建立起多个表之间的关联，然后才能去查询，同时还需要规避关联表查询中的常见错误。具体怎么做呢？我来借助实际的项目给你讲一讲。

在我们项目的进货模块，有这样 2 个数据表，分别是进货单头表（importhead）和进货单明细表（importdetails），我们每天都要对这两个表进行增删改查的操作。

进货单头表记录的是整个进货单的总体信息：

listnumber（进货单号）	supplierid（供货商编号）	stocknumber（仓库编号）	importtype（进货类别）	totalquantity（总计数量）	totalvalue（总计金额）	recorder（录入人编号）	recordingdate（录入时间）
1234	1	1	1	30	1110	1	2020-12-20

进货单明细表记录了每次进货的商品明细信息。一条进货单头数据记录，对应多条进货商品的明细数据，也就是所谓的一对多的关系。具体信息如下表所示：

listnumber（进货单号）	itemnumber（商品编号）	quantity（进货数量）	Importprice（进货价格）	importvalue（进货金额）
1234	1	10	89	890
1234	2	20	12	240

现在我们需要查询一次进货的所有相关数据，包括进货单的总体信息和进货商品的明细，这样一来，我们就需要把 2 个表关联起来，那么，该怎么操作呢？

在 MySQL 中，为了把 2 个表关联起来，会用到 2 个重要的功能：外键（FOREIGN KEY）和连接（JOIN）。外键需要在创建表的阶段就定义；连接可以通过相同意义的字段把 2 个表连接起来，用在查询阶段。

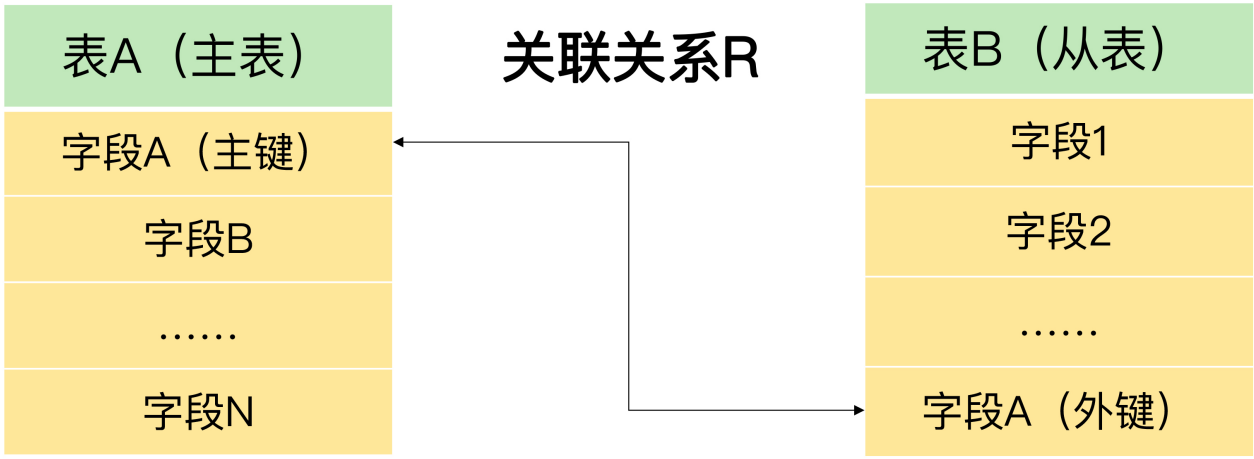
接下来，我就先和你聊聊外键。

如何创建外键？

我先来解释一下什么是外键。

假设我们有 2 个表，分别是表 A 和表 B，它们通过一个公共字段“id”发生关联关系，我们把这个关联关系叫做 R。如果“id”在表 A 中是主键，那么，表 A 就是这个关系 R 中的主表。相应的，表 B 就是这个关系中的从表，表 B 中的“id”，就是表 B 用来引用表 A 中数据的，叫外键。所以，**外键就是从表中用来引用主表中数据的那个公共字段**。

为了方便你理解，我画了一张图来展示：



如图所示，在关联关系 R 中，公众字段（字段 A）是表 A 的主键，所以表 A 是主表，表 B 是从表。表 B 中的公共字段（字段 A）是外键。

在 MySQL 中，外键是通过外键约束来定义的。外键约束就是约束的一种，它必须在从表中定义，包括指明哪个是外键字段，以及外键字段所引用的主表中的主键字段是什么。MySQL 系统会根据外键约束的定义，监控对主表中数据的删除操作。如果发现要删除的主表记录，正在被从表中某条记录的外键字段所引用，MySQL 就会提示错误，从而确保了关联数据不会缺失。

外键约束可以在创建表的时候定义，也可以通过修改表来定义。我们先来看看外键约束定义的语法结构：

复制代码

```
[CONSTRAINT <外键约束名称>] FOREIGN KEY 字段名
REFERENCES <主表名> 字段名
```

你可以在创建表的时候定义外键约束：

复制代码

```
CREATE TABLE 从表名
(
  字段名 类型,
  ...
-- 定义外键约束，指出外键字段和参照的主表字段
CONSTRAINT 外键约束名
FOREIGN KEY （字段名） REFERENCES 主表名 （字段名）
)
```

当然，你也可以通过修改表来定义外键约束：

复制代码

```
ALTER TABLE 从表名 ADD CONSTRAINT 约束名 FOREIGN KEY 字段名 REFERENCES 主表名 （字
```

一般情况下，表与表的关联都是提前设计好了的，因此，会在创建表的时候就把外键约束定义好。不过，如果需要修改表的设计（比如添加新的字段，增加新的关联关系），但没有预先定义外键约束，那么，就要用修改表的方式来补充定义。

下面，我就来讲一讲怎么创建外键约束。

先创建主表 demo.importhead：

复制代码

```
CREATE TABLE demo.importhead (
    listnumber INT PRIMARY KEY,
    supplierid INT,
    stocknumber INT,
    importtype INT,
    importquantity DECIMAL(10 , 3 ),
    importvalue DECIMAL(10 , 2 ),
    recorder INT,
    recordingdate DATETIME
);
```

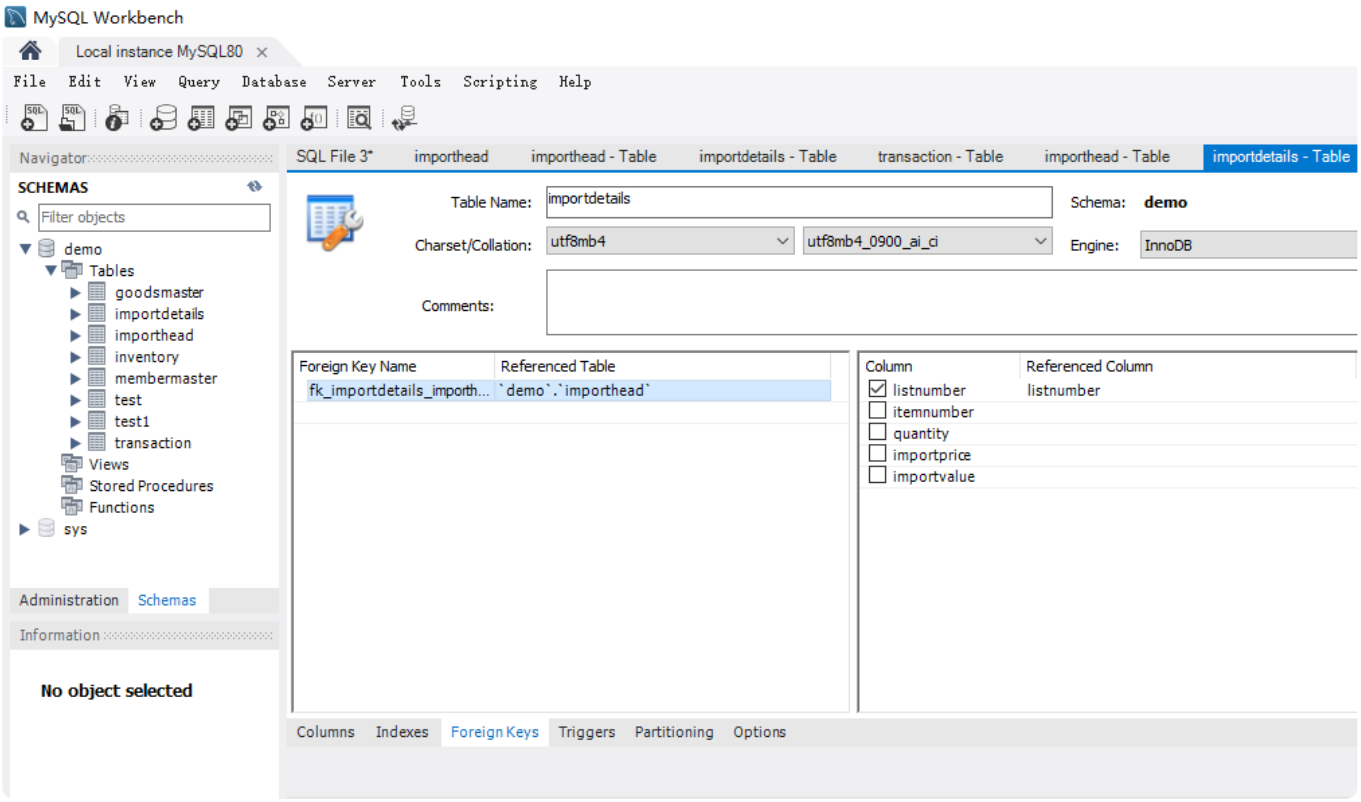
然后创建从表 demo.importdetails，并且给它定义外键约束：

复制代码

```
CREATE TABLE demo.importdetails
(
  listnumber INT,
  itemnumber INT,
  quantity DECIMAL(10,3),
  importprice DECIMAL(10,2),
  importvalue DECIMAL(10,2),
  -- 定义外键约束，指出外键字段和参照的主表字段
  CONSTRAINT fk_importdetails_importhead
  FOREIGN KEY (listnumber) REFERENCES importhead (listnumber)
);
```

运行这个 SQL 语句，我们就在创建表的同时定义了一个名字叫“fk_importdetails_importhead”的外键约束。同时，我们声明，这个外键约束的字段“listnumber”引用的是表 importhead 里面的字段“listnumber”。

外键约束创建好之后，我们可以通过 Workbench，来查看外键约束是不是创建成功了：



当然，我们也可以通过 SQL 语句来查看，这里我们要用到 MySQL 自带的、用于存储系统信息的数据库：information_schema。我们可以查看外键约束的相关信息：

复制代码

```
mysql> SELECT
    ->     constraint_name, -- 表示外键约束名称
    ->     table_name, -- 表示外键约束所属数据表的名称
    ->     column_name, -- 表示外键约束的字段名称
    ->     referenced_table_name, -- 表示外键约束所参照的数据表名称
    ->     referenced_column_name -- 表示外键约束所参照的字段名称
    -> FROM
    ->     information_schema.KEY_COLUMN_USAGE
    -> WHERE
    ->     constraint_name = 'fk_importdetails_importhead';
+-----------------------------+---------------+-------------+-----------------
| CONSTRAINT_NAME             | TABLE_NAME    | COLUMN_NAME | REFERENCED_TABLE
+-----------------------------+---------------+-------------+-----------------
| fk_importdetails_importhead | importdetails | listnumber  | importhead
+-----------------------------+---------------+-------------+-----------------
1 row in set (0.05 sec)
```

通过查询，我们可以看到，外键约束所在的表是“importdetails”，外键字段是“listnumber”，参照的主表是“importhead”，参照的主表字段是“listnumber”。这样，通过定义外键约束，我们已经建立起了 2 个表之间的关联关系。

关联关系建立起来之后，如何才能获取我们需要的数据呢？这时，我们就需要用到连接查询了。

连接

在 MySQL 中，有 2 种类型的连接，分别是内连接（INNER JOIN）和外连接（OUTER JOIN）。

> 内连接表示查询结果只返回符合连接条件的记录，这种连接方式比较常用；
>
> 外连接则不同，表示查询结果返回某一个表中的所有记录，以及另一个表中满足连接条件的记录。

下面我先来讲一下内连接。

在 MySQL 里面，关键字 JOIN、INNER JOIN、CROSS JOIN 的含义是一样的，都表示内连接。我们可以通过 JOIN 把两个表关联起来，来查询两个表中的数据。

我借助一个小例子，来帮助你理解。

咱们的项目中有会员销售的需求，所以，我们的流水表中的数据记录，既包括非会员的普通销售，又包括会员销售。它们的区别是，会员销售的数据记录包括会员编号，而在非会员销售的数据记录中，会员编号为空。

来看一下项目中的销售表（demo.trans)。实际的销售表比较复杂，为了方便你理解，我把表进行了简化，并且假设业务字段 cardno 是会员信息表的主键。简化以后的结构如下所示：

transactionnumber（流水单号）	Itemnumber（商品编号）	price（价格）	quantity（数量）	cardno（会员卡号）	transdate（交易时间）
1	1	89	1	10000001	2020-12-01
2	2	12	1	NULL	2020-12-02

再看下简化后的会员信息表（demo.membermaster）：

cardno（会员卡号）	membername（会员名称）	memberphone（会员电话）	memberpid（会员身份证号）	sex（会员性别）
10000001	张三	13512345678	110123199901017890	男

这两个表之间存在关联关系，表 demo.trans 中的字段“cardno”是这个关联关系中的外键。

我们可以通过内连接，查询所有会员销售的流水记录：

复制代码

```
mysql> SELECT
    ->     a.transactionno,
    ->     a.itemnumber,
    ->     a.quantity,
    ->     a.price,
```

```
    ->     a.transdate,
    ->     b.membername
    -> FROM
    ->     demo.trans AS a
    ->         JOIN
    ->     demo.membermaster AS b ON (a.cardno = b.cardno);
+---------------+------------+----------+-------+---------------------+-------
| transactionno | itemnumber | quantity | price | transdate           | member
+---------------+------------+----------+-------+---------------------+-------
|             1 |          1 |    1.000 | 89.00 | 2020-12-01 00:00:00 | 张三
+---------------+------------+----------+-------+---------------------+-------
1 row in set (0.00 sec)
```

可以看到，我们通过公共字段“cardno”把两个表关联到了一起，查询出了会员消费的数据。

在这里，关键字 JOIN 与关键字 ON 配对使用，意思是查询满足关联条件“demo.trans 表中 cardno 的值与 demo.membermaster 表中的 cardno 值相等”的两个表中的所有记录。

知道了内连接，我们再来学习下外连接。跟内连接只返回符合连接条件的记录不同的是，外连接还可以返回表中的所有记录，它包括两类，分别是左连接和右连接。

> 左连接，一般简写成 LEFT JOIN，返回左边表中的所有记录，以及右表中符合连接条件的记录。
>
> 右连接，一般简写成 RIGHT JOIN，返回右边表中的所有记录，以及左表中符合连接条件的记录。

当我们需要查询全部流水信息的时候，就会用到外连接，代码如下：

复制代码

```
SELECT
    a.transactionno,
    a.itemnumber,
    a.quantity,
    a.price,
    a.transdate,
    b.membername
FROM demo.trans AS a
LEFT JOIN demo.membermaster AS b -- LEFT JOIN，以demo.transaction为主
ON (a.cardno = b.cardno);
```

可以看到，我用到了 LEFT JOIN，意思是以表 demo.trans 中的数据记录为主，这个表中的数据记录要全部出现在结果集中，同时给出符合连接条件（a.cardno=b.cardno) 的表 demo.membermaster 中的字段 membername 的值。

我们也可以使用 RIGHT JOIN 实现同样的效果，代码如下：

复制代码

```
SELECT
    a.transactionno,
    a.itemnumber,
    a.quantity,
    a.price,
    a.transdate,
    b.membername
FROM
    demo.membermaster AS b
        RIGHT JOIN
    demo.trans AS a ON (a.cardno = b.cardno); -- RIGHT JOIN，顺序颠倒了，还是以de
```

其实，这里就是把顺序颠倒了一下，意思是一样的。运行之后，我们都能得到下面的结果：

复制代码

```
mysql> SELECT
    ->     a.transactionno,
    ->     a.itemnumber,
    ->     a.quantity,
    ->     a.price,
    ->     a.transdate,
    ->     b.membername
    -> FROM
    ->     demo.trans AS a
    ->         LEFT JOIN   -- 左连接
    ->     demo.membermaster AS b ON (a.cardno = b.cardno);
+---------------+------------+----------+-------+---------------------+-------
| transactionno | itemnumber | quantity | price | transdate           | member
+---------------+------------+----------+-------+---------------------+-------
|             1 |          1 |    1.000 | 89.00 | 2020-12-01 00:00:00 | 张三
|             2 |          2 |    1.000 | 12.00 | 2020-12-02 00:00:00 | NULL
+---------------+------------+----------+-------+---------------------+-------
2 rows in set (0.00 sec)

```

```
mysql> SELECT
    ->     a.transactionno,
    ->     a.itemnumber,
    ->     a.quantity,
    ->     a.price,
    ->     a.transdate,
    ->     b.membername
    -> FROM
    ->     demo.membermaster AS b
    ->         RIGHT JOIN   -- 右连接
    ->     demo.trans AS a
    ->     ON (a.cardno = b.cardno);
+---------------+------------+----------+-------+---------------------+-------
| transactionno | itemnumber | quantity | price | transdate           | member
+---------------+------------+----------+-------+---------------------+-------
|             1 |          1 |    1.000 | 89.00 | 2020-12-01 00:00:00 | 张三
|             2 |          2 |    1.000 | 12.00 | 2020-12-02 00:00:00 | NULL
+---------------+------------+----------+-------+---------------------+-------
2 rows in set (0.00 sec)
```

通过关联查询，销售流水数据里就补齐了会员的名称，我们也就获取到了需要的数据。

关联查询的误区

有了连接，我们就可以进行 2 个表的关联查询了。你可能会有疑问：关联查询必须在外键约束的基础上，才可以吗?

其实，在 MySQL 中，外键约束不是关联查询的必要条件。很多人往往在设计表的时候，觉得只要连接查询就可以搞定一切了，外键约束太麻烦，没有必要。如果你这么想，就进入了一个误区。

下面我就以超市进货的例子，来实际说明一下，为什么这种思路不对。

假设一次进货数据是这样的：供货商编号是 1，进货仓库编号是 1。我们进货的商品编号是 1234，进货数量是 1，进货价格是 10，进货金额是 10。

我先插入单头数据：

复制代码

```
INSERT INTO demo.importhead
(
listnumber,
```

```
supplierid,
stocknumber
)
VALUES
(
1234,
1,
1
);
```

运行成功后，查看一下表的内容：

复制代码

```
mysql> SELECT *
    -> FROM demo.importhead;
+------------+------------+-------------+------------+----------+-------------
| listnumber | supplierid | stocknumber | importtype | quantity | importprice
+------------+------------+-------------+------------+----------+-------------
|       1234 |          1 |           1 |          1 |     NULL |        NULL
+------------+------------+-------------+------------+----------+-------------
1 row in set (0.00 sec)
```

可以看到，我们有了一个进货单头，单号是 1234，供货商是 1 号供货商，进货仓库是 1 号仓库。

接着，我们向进货单明细表中插入进货明细数据：

复制代码

```
INSERT INTO demo.importdetails
(
listnumber,
itemnumber,
quantity,
importprice,
importvalue
)
VALUES
(
1234,
1,
1,
10,
10
);
```

运行成功，查看一下表的内容：

复制代码

```
mysql> SELECT *
    -> FROM demo.importdetails;
+------------+------------+----------+-------------+-------------+
| listnumber | itemnumber | quantity | importprice | importvalue |
+------------+------------+----------+-------------+-------------+
|       1234 |          1 |    1.000 |       10.00 |       10.00 |
+------------+------------+----------+-------------+-------------+
1 row in set (0.00 sec)
```

这样，我们就有了 1234 号进货单的明细数据：进货商品是 1 号商品，进货数量是 1 个，进货价格是 10 元，进货金额是 10 元。

这个时候，如果我删除进货单头表的数据，就会出现只有明细、没有单头的数据缺失情况。我们来看看会发生什么：

复制代码

```
DELETE FROM demo.importhead
WHERE listnumbere = 1234;
```

运行这条语句，MySQL 会提示错误，因为数据删除违反了外键约束。看到了吗？MySQL 阻止了数据不一致的情况出现。

不知道你有没有注意我插入数据的顺序：为什么我要先插入进货单头表的数据，再插入进货单明细表的数据呢？其实，这是因为，如果我先插入数据到从表，也就是进货单明细表，会导致 MySQL 找不到参照的主表信息，会提示错误，因为添加数据违反了外键约束。

你可能会不以为然，觉得按照信息系统的操作逻辑，生成一张进货单的时候，一定是先生成单头，再插入明细。同样，删除一张进货单的时候，一定是先删除明细，再删除单头。

要是你这么想，可能就会“中招”了。原因很简单，既然我们把进货数据拆成了 2 个表，这就决定了无论是数据添加，还是数据删除，都不能通过一条 SQL 语句实现。实际工作中，什么突发情况都是有可能发生的。你认为一定会完成的操作，完全有可能只执行了一部分。

我们曾经就遇到过这么一个问题：用户月底盘点的时候，盘点单无法生成，系统提示“有未处理的进货单”。经过排查，发现是进货单数据发生了数据缺失，明细数据还在，对应的单头数据却被删除了。我们反复排查之后，才发现是缺少了防止数据缺失的机制。最后通过定义外键约束，解决了这个问题。

所以，虽然你不用外键约束，也可以进行关联查询，但是有了它，MySQL 系统才会保护你的数据，避免出现误删的情况，从而提高系统整体的可靠性。

现在来回答另外一个问题，为什么在 MySQL 里，没有外键约束也可以进行关联查询呢？原因是外键约束是有成本的，需要消耗系统资源。对于大并发的 SQL 操作，有可能会不适合。比如大型网站的中央数据库，可能会因为外键约束的系统开销而变得非常慢。所以，MySQL 允许你不使用系统自带的外键约束，在应用层面完成检查数据一致性的逻辑。也就是说，即使你不用外键约束，也要想办法通过应用层面的附加逻辑，来实现外键约束的功能，确保数据的一致性。

总结

这节课，我给你介绍了如何进行多表查询，我们重点学习了外键和连接。

外键约束，可以帮助我们确定从表中的外键字段与主表中的主键字段之间的引用关系，还可以确保从表中数据所引用的主表数据不会被删除，从而保证了 2 个表中数据的一致性。

连接可以帮助我们对 2 个相关的表进行连接查询，从 2 个表中获取需要的信息。左连接表示连接以左边的表为主，结果集中要包括左边表中的所有记录；右连接表示连接以右边的表为主，结果集中要包括右边表中的所有记录。

我汇总了常用的 SQL 语句，你一定要重点掌握。

复制代码

```
1 -- 定义外键约束：
```

```
CREATE TABLE 从表名
(
字段 字段类型
....
CONSTRAINT 外键约束名称
FOREIGN KEY (字段名) REFERENCES 主表名 (字段名称)
);
ALTER TABLE 从表名 ADD CONSTRAINT 约束名 FOREIGN KEY 字段名 REFERENCES 主表名 (字

-- 连接查询
SELECT 字段名
FROM 表名 AS a
JOIN 表名 AS b
ON (a.字段名称=b.字段名称);

SELECT 字段名
FROM 表名 AS a
LEFT JOIN 表名 AS b
ON (a.字段名称=b.字段名称);

SELECT 字段名
FROM 表名 AS a
RIGHT JOIN 表名 AS b
ON (a.字段名称=b.字段名称);
```

刚开始学习 MySQL 的同学，很容易忽略在关联表中定义外键约束的重要性，从而导致数据缺失，影响系统的可靠性。我建议你尽量养成在关联表中定义外键约束的习惯。不过，如果你的业务场景因为高并发等原因，无法承担外键约束的成本，也可以不定义外键约束，但是一定要在应用层面实现外键约束的逻辑功能，这样才能确保系统的正确可靠。

思考题

如果你的业务场景因高并发等原因，不能使用外键约束，在这种情况下，你怎么在应用层面确保数据的一致性呢？

欢迎在留言区写下你的思考和答案，我们一起交流讨论。如果你觉得今天的内容对你有所帮助，欢迎你把它分享给你的朋友或同事，我们下节课见。

提建议

上一篇 05 | 主键：如何正确设置主键？

下一篇 07 | 条件语句：WHERE 与 HAVING有什么不同？

精选留言 (8)

写留言

lesserror
2021-03-20

外键约束，可以简单有效的保证数据的可靠性。适合内部管理系统使用，因为访问量不会太大。如果是面向外界用户使用的应用，外键所带来的性能损耗，可能无法支撑大批用户的同时访问。

但是，数据的可靠性和唯一性才是最重要的，在保证一次对多张表操作的过程中，可以...
展开

作者回复: 在系统开销和使用的功能之间需要做好平衡，既要确保数据可靠性和唯一性，又要确保系统可用

3

ple
2021-03-20

回答问题觉得可以不在数据库里做，使用事物让操作变成原子操作。

2

X-Dragon
2021-03-24

老师讲课很好，实战举例很符合现实，也简洁易懂，小白也看得懂，为老师点个赞👍

作者回复: 谢谢鼓励！

右耳朵猫咪
2021-03-22

文中说“左连接，一般简写成 LEFT JOIN，返回左边表中的所有记录，以及右表中符合连接条件的记录。”感觉不太准确，应该是不管连接条件是否满足，都会返回，如果符合连接条件，就返回右表对应的记录，如果不符合连接条件，就标注为null。

展开

作者回复: 这里的意思是指返回的数据记录，返回左表的全部记录，以及右表中符合连接条件的记录，如果不符合连接条件，MySQL会返回NULL

1

peter
2021-03-20

不考虑应用层面，只考虑数据库层面，没有外键能进行关联查询吗？（或者这样说： 没有外键，在数据库层面用SQL语句能进行关联查询吗？

展开

作者回复: 可以，请看06篇。考虑到数据一致性，在数据表比较多时尽量还是设置外键，查询更方便。

1

Harry
2021-03-22

交作业：

1、用一张宽表来记录主表和从表(不存在连接查询)
2、删除主表数据前检查从表(效率低)

展开


Harry
2021-03-22

“连接可以通过相同意义的字段把 2 个表连接起来”
这里是指外键字段和主键字段的名称、类型必须完全一致吗？

随着表中数据日益剧增，最好不要在数据库层面创建关联关系，因为连接查询会产生巨...

展开


陈启年
2021-03-22

老师，在进行多表关联查询时，我有几个问题：
- join关键字可以叠加多次使用吗？
- left join和right join可以叠加多次使用吗？如果可以的话，那怎么理解他们的含义呢（因为两者是相斥的关系）？
...

展开