你好，我是朱晓峰。今天，我们来聊一聊数据表设计的范式。

在超市项目的设计阶段，超市经营者把他们正在用的 Excel 表格给到我们，要求我们把这些数据存储到超市管理系统的数据库中。为了方便你理解，我挑选了 1 个有代表性的表来举例说明。

进货单表（import）：

listnumber（单号）	supplierid（供货商编号）	Suppliername（供货商名称）	stock（仓库）	barcode（条码）	goodsname（名称）	property（属性）	quantity（数量）	price（进货价格）	importvalue（进货金额）
3478	1	出版社	仓库	0001	书	16开/本	10	45	450
3478	1	出版社	仓库	0002	地图	张	5	9.9	49.5
3490	2	文具厂	卖场	0003	笔	10支/包	1	3.5	3.5
3492	1	出版社	仓库	0002	地图	张	10	9.5	95

这个表中的字段很多，包含了各种信息，表里的数据量也很惊人。我们刚拿到这个表的时候，光是打开表这个操作，就需要等十几秒。

仔细一看，发现表里重复的数据非常多：比如第一行和第二行的数据，同样是 3478 号单据，供货商编号、供货商名称和仓库，这 3 个字段的信息完全相同。可是这 2 条数据的后半部分又不相同，因此，并不能认为它们是冗余数据而删除。

其实，造成这种问题的原因是这张表的设计非常不合理，大量重复导致表变得庞大，效率极低。

在我们的工作场景中，这种由于数据表结构设计不合理，而导致的数据重复的现象并不少见，往往是系统虽然能够运行，承载能力却很差，稍微有点流量，就会出现内存不足、CUP 使用率飙升的情况，甚至会导致整个项目失败。

所以，**高效可靠的设计是提升数据库工作效率的关键**。那该怎么设计呢？有没有什么可以参考的设计规范呢？自然是有了。

接下来，我就带你重新设计一下刚刚的进货单表，在这个过程中给你具体介绍一下数据表设计的三大范式，分别是第一范式（1NF）、第二范式（2NF）和第三范式（3NF），这些范式可以帮助我们设计出简洁高效的数据表，进而提高系统的效率。

我先来介绍一下最基本的第一范式。

第一范式

我们对这张进货单表重新设计的第一步，就是要把所有的列，也就是字段，都确认一遍，确保每个字段只包含一种数据。如果各种数据都混合在一起，就无法通过后面的拆解，把重复的数据去掉。

其实，这就是第一范式所要求的：**所有的字段都是基本数据字段，不可进一步拆分。**

在我们的这张表里，“property”这一字段可以继续拆分。其他字段已经都是基本数据字段，不能再拆了。

经过优化，我们把“property”这一字段，拆分成“specification（规格）”和“unit（单位）”，这 2 个字段如下：

listnumber	supplierid	Supplier	stock	barcode	goodsname	spcificaton	unit	quantity	importprice	importvalue
3478	1	出版社	仓库	0001	书	16开	本	10	45	450
3478	1	出版社	仓库	0002	地图	NULL	张	5	9.9	49.5
3490	2	铅笔厂	卖场	0003	笔	10支	包	1	3.5	3.5
3492	1	出版社	卖场	0002	地图	NULL	张	10	9.5	95

这样处理之后，字段多了一个，但是每一个字段都成了不可拆分的最小信息单元，我们就可以在这个表的基础之上，着手进行进一步的优化了。这就要用到数据表设计的第二范式了。

第二范式

通过观察，我们可以发现，这个表的前 2 条记录的前 4 个字段完全一样。那可不可以通过拆分，把它们变成一条记录呢？当然是可以的，而且为了优化，必须要进行拆分。

具体怎么拆分呢？第二范式就告诉了我们拆分的原则。

第二范式要求，在满足第一范式的基础上，**还要满足数据表里的每一条数据记录，都是可唯一标识的。而且所有字段，都必须完全依赖主键，不能只依赖主键的一部分。**

根据这个要求，我们可以对表进行重新设计。

重新设计的第一步，就是要确定这个表的主键。通过观察发现，字段“listnumber”+“barcode”可以唯一标识每一条记录，可以作为主键。确定好了主键以后，我们判断一下，哪些字段完全依赖主键，哪些字段只依赖于主键的一部分。同时，把只依赖于主键一部分的字段拆分出去，形成新的数据表。

首先，进货单明细表里面的“goodsname”“specification”“unit”这些信息是商品的属性，只依赖于“barcode”，不完全依赖主键，可以拆分出去。我们把这 3 个字段加上它们所依赖的字段“barcode”，拆分形成一个新的数据表“**商品信息表**”。

这样一来，原来的数据表就被拆分成了两个表。

商品信息表：

barcode	goodsname	spcificaton	unit
0001	书	16开	本
0002	地图	NULL	张
0003	笔	10支	包

进货单表：

listnumber	Supplierid	Supplier	stock	barcode	quantity	importprice	importvalue
3478	1	出版社	仓库	0001	10	45	450
3478	1	出版社	仓库	0002	5	9.9	49.5
3490	2	铅笔厂	卖场	0003	1	3.5	3.5
3492	1	出版社	卖场	0002	10	9.5	95

同样道理，字段“supplierid”“suppliername”“stock”只依赖于“listnumber”，不完全依赖于主键，所以，我们可以把“supplierid”“suppliername”“stock”这 3 个字段拆出去，再加上它们依赖的字段“listnumber”，就形成了一个新的表“进货单头表”。剩下的字段，会组成新的表，我们叫它“**进货单明细表**”。

这样一来，原来的数据表就拆分成了 3 个表。

进货单头表：

listnumber	supplierid	suppliername	stock
3478	1	出版社	仓库
3490	2	铅笔厂	卖场
3492	1	出版社	卖场

进货单明细表：

listnumber	barcode	quantity	importprice	importvalue
3478	0001	10	45	450
3478	0002	5	9.9	49.5
3490	0003	1	3.5	3.5
3492	0002	10	9.5	95

商品信息表：

barcode	goodsname	specification	unit
0001	书	16开	本
0002	地图	NULL	张
0003	笔	10支	包

到这里，我们就按照第二范式的要求，把原先的一个数据表拆分成了 3 个数据表。

现在，我们再来分析一下拆分后的 3 个表，保证这 3 个表都满足第二范式的要求。

在“商品信息表”中，字段“barcode”是有可能存在重复的，比如，用户门店可能有散装称重商品和自产商品，会存在条码共用的情况。所以，所有的字段都不能唯一标识表里的记录。这个时候，我们必须给这个表加上一个主键，比如说是自增字段“itemnumber”。

现在，我们就可以把进货单明细表里面的字段“barcode”都替换成字段“itemnumber”，这就得到了新的进货单明细表和商品信息表。

进货单明细表：

listnumber	itemnumber	quantity	importprice	importvalue
3478	1	10	45	450
3478	2	5	9.9	49.5
3490	3	1	3.5	3.5
3492	2	10	9.5	95

商品信息表：

itemnumber	barcode	goodsname	specification	unit
1	0001	书	16开	本
2	0002	地图	NULL	张
3	0003	笔	10支	包

这样一来，我们拆分后的 3 个数据表中的数据都不存在重复，可以唯一标识。而且，表中的其他字段，都完全依赖于表的主键，不存在部分依赖的情况。所以，拆分后的 3 个数据表就全部满足了第二范式的要求。

第三范式

如果你仔细看的话，会发现，我们的进货单头表，还有数据冗余的可能。因为“suppliername”依赖“supplierid”。那么，这个时候，就可以按照第三范式的原则进行拆分了。

第三范式要求数据表**在满足第二范式的基础上，不能包含那些可以由非主键字段派生出来的字段，或者说，不能存在依赖于非主键字段的字段。**

在刚刚的进货单头表中，字段“suppliername”依赖于非主键字段“supplierid”。因此，这个表不满足第三范式的要求。

那接下来，我们就进一步拆分下进货单头表，把它拆解成供货商表和进货单头表。

供货商表：

supplierid	suppliername
1	出版社
2	铅笔厂

进货单头表：

listnumber	supplierid	stock
3478	1	仓库
3490	2	卖场
3492	1	卖场

这样一来，供货商表和进货单头表中的所有字段，都完全依赖于主键，不存在任何一个字段依赖于非主键字段的情况了。所以，这 2 个表就都满足第三范式的要求了。

但是，在进货单明细表中，quantity * importprice = importvalue，“importprice”“quantity”和“importvalue”这 3 个字段，可以通过任意两个计算出第三个来，这就存在冗余字段。如果严格按照第三范式的要求，现在我们应该进行进一步优化。优化的办法是删除其中一个字段，只保留另外 2 个，这样就没有冗余数据了。

可是，真的可以这样做吗？要回答这个问题，我们就要先了解下实际工作中的业务优先原则。

业务优先的原则

所谓的业务优先原则，就是指一切以业务需求为主，技术服务于业务。**完全按照理论的设计不一定就是最优，还要根据实际情况来决定。**这里我们就来分析一下不同选择的利与弊。

对于 quantity * importprice = importvalue，看起来“importvalue”似乎是冗余字段，但并不会导致数据不一致。可是，如果我们把这个字段取消，是会影响业务的。

因为有的时候，供货商会经常进行一些促销活动，按金额促销，那他们拿来的进货单只有金额，没有价格。而“importprice”反而是通过“importvalue”÷“quantity”计算出来的。因此，如果不保留“importvalue”字段，只有“importprice”和“quantity”的话，经过四舍五入，会产生较大的误差。这样日积月累，最终会导致查询结果出现较大偏差，影响系统的可靠性。

我借助一个例子来说明下为什么会有偏差。

假设进货金额是 25.5 元，数量是 34，那么进货价格就等于 25.5÷34=0.74 元，但是如果用这个计算出来的进货价格来计算进货金额，那么，进货金额就等于 0.74×34=25.16 元，其中相差了 25.5-25.16=0.34 元。代码如下所示：

```
“importvalue”=25.5元，“quantity”=34，“importprice”=25.5÷34=0.74
“importprice”=0.74元，“quantity”=34，“importvalue”=0.74*34=25.16
误差 = 25.5 - 25.16 = 0.34
```

现在你知道了，在我们这个场景下，“importvalue”是必须要保留的。

那么，换一种思路，如果我们保留“quantity”和“importvalue”，取消“importprice”，这样不是既能节省存储空间，又不会影响精确度吗？

其实不是的。“importprice”是系统的核心指标，涉及成本核算。几乎所有的财务、营运和决策支持模块，都要涉及到成本问题，如果取消“importprice”这个字段，那么系统的运算成本、开发和运维成本，都会大大提高，得不偿失。

所以，本着业务优先的原则，在不影响系统可靠性的前提下，可以容忍一定程度的数据冗余，保留“importvalue”“importprice”和“quantity"。

因此，最后的结果是，我们可以把进货单表拆分成下面的 4 个表：

供货商表：

supplierid	suppliername
1	出版社
2	铅笔厂

进货单头表：

listnumber	supplierid	stock
3478	1	仓库
3490	2	卖场
3492	1	卖场

进货单明细表：

listnumber	itemnumber	quantity	importprice	importvalue
3478	1	10	45	450
3478	2	5	9.9	49.5
3490	3	1	3.5	3.5
3492	2	10	9.5	95

商品信息表：

itemnumber	barcode	goodsname	specification	unit
1	0001	书	16开	本
2	0002	地图	NULL	张
3	0003	笔	10支	包

这样一来，我们就避免了冗余数据，而且还能够满足业务的需求，这样的数据表设计，才是合格的设计。

总结

今天，我们通过具体案例的分析，学习了 MySQL 数据库设计的范式规范，包括第一范式、第二范式和第三范式。

我再给你汇总下 MySQL 数据库规范化设计的三个范式。

- 第一范式：数据表中所有字段都是不可拆分的基本数据项。
- 第二范式：在满足第一范式的基础上，数据表中所有非主键字段，必须完全依赖全部主键字段，不能存在部分依赖主键字段的字段。
- 第三范式：在满足第二范式的基础上，数据表中不能存在可以被其他非主键字段派生出来的字段，或者说，不能存在依赖于非主键字段的字段。

遵循范式的要求，可以减少冗余，结合外键约束，可以防止添加、删除、修改数据时产生数据的不一致问题。

除此之外，我还给你解释了为什么有的时候不能简单按照规范要求设计数据表，因为有的数据看似冗余，其实对业务来说十分重要。这个时候，我们就要遵循业务优先的原则，首先满足业务需求，在这个前提下，再尽量减少冗余。

一般来说，MySQL 的数据库设计满足第三范式，就足够了。不过，第三范式，并不是终极范式，还有BCNF 范式（也叫 BC 范式）、第四范式和第五范式。如果你想进一步研究数据库设计，课下可以看下我分享的链接，拓展下思路。

思考题

假设有这样一个销售流水表，如下所示：

transactionno	itemnumber	goodsname	quantity	price	salesvalue	cardno
1	1	书	1	89	89	10000001

这个表存在数据冗余，应该如何优化设计呢？为什么？

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，也欢迎你分享给你的朋友或同事，我们下节课见。

精选留言(4)

Skrpy

老师除法算错了哦，25.5÷34=0.75 这是精确的啊，可以换个数来举例，哈哈哈😄

2021-05-10 1

floating

销售流水表可以拆分成三张表：
1、transcationnumber、transcationno、itemnumber、salesvalue、cardnumber
2、itemnumber、goodsname、quantity、price
3、cardnumber、cardno

2021-05-13

Michael

思考题：
我的理解是只能把itemnumber和goodsname拆分出去，其他的都拆分不出去。因为我们结账的时候应该是需要记录当时的价格信息的，这个相当于一个是对当时价格的一个快照，所以只有itemnumber和goodsname能拆分，这个的对应关系不会变。

2021-05-07 1

Harry

25.5 ÷ 34 = 0.74 (因为计算结果保留了两位小数)。

所以“importvalue”“importprice”和“quantity”冗余的真正原因是字段精度的问题？

除了三个范式以外，竟然还有其他范式的存在，学习了。

思考题答案：

可以把销售流水表中的商品信息(商品编号、商品名称、数量、单价)和会员信息(会员卡号)单独拆分出来，因为它们不直接依赖于表中的主键字段。

2021-05-04