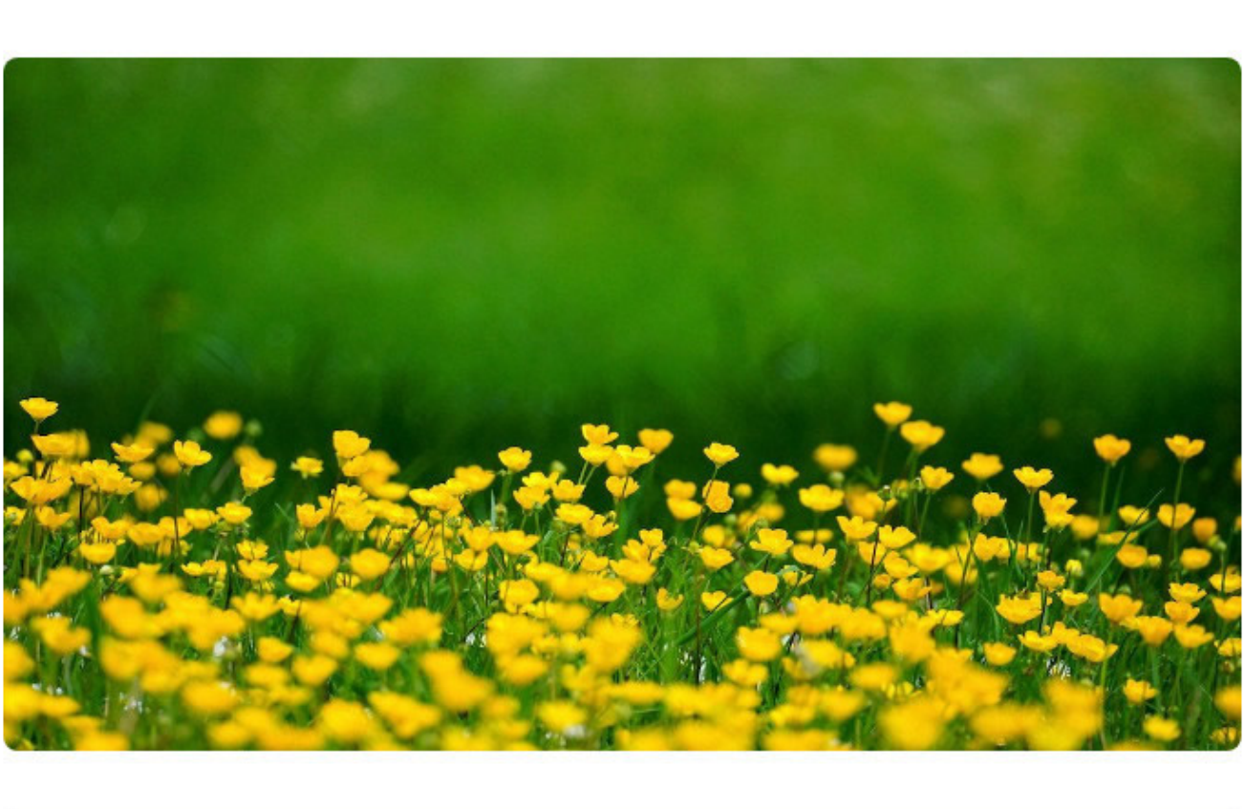
27 | 手把手带你设计一个完整的连锁超市信息系统数据库（上）

朱晓峰 2021-05-15



00:00 1.0x
讲述：朱晓峰 大小：13.86M 时长：15:07

你好，我是朱晓峰。

从创建第一个 MySQL 数据库开始到现在，我们已经学完了 MySQL 的核心操作。最后这两节课，我想带你实际设计一个超市信息系统的数据库。毕竟，设计数据库很考验我们综合运用单个技术点的能力。所以，通过这个项目，我不仅会带你把前面的内容串联起来，而且还会教你设计数据库的通用思路。

为什么选择超市项目呢？一方面呢，超市的场景与我们的日常生活密切相关，你比较容易理解其中的业务逻辑，另一方面，超市的业务又相当复杂，几乎能用到我们学到的所有知识点，利于我们对前面学过的内容进行整合。

今天，我就带你一起，从需求分析开始入手，一直到容灾和备份，完成一个全流程的连锁超市数据库设计。这节课，我会主要给你讲解需求分析、ER 模型、分库分表和数据库设计方案。我们做项目，首先要从大处着眼，把握方向，这样才不容易出现大的偏差。下节课，我们再设计具体的细节，比如创建数据表、外键约束，设计灾备方案和备份数据等。

在开始设计之前，咱们得先了解一下项目背景。

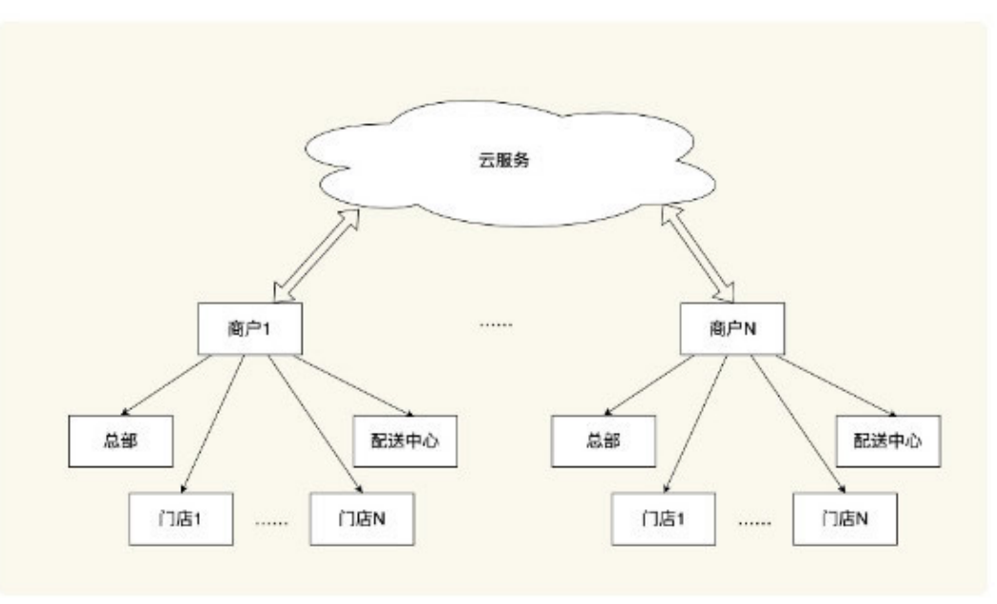
随着互联网使用的日益广泛，传统的桌面管理系统的数据不能互通、资源利用率低等弊端越来越明显，显然不能满足用户的需求了。因此，我们需要开发一款基于云服务的连锁超市管理信息系统。具体的要求是：

1. 基于浏览器，无需安装，账号开通即可使用，方便快捷；
2. 数据部署在云端，由运营商负责维护，安全可靠；
3. 用户无需自备服务器，只需租用信息服务，资源利用率高。

知道了具体要求，那该怎么进行设计呢？下面我带你来分析一下。

如何设计数据结构？

用户账号开通即可使用，所以必然要设计分层的数据结构；数据要部署在云端，所以必然要使用云。根据这些要求，我们可以设计一个基于云服务的 2 层数据结构，这个结构的示意图如下所示：



我来解释一下图中展示的内容。

首先，你可以看到，所有的系统资源和服务都部署在云端。

其次，我们来看下数据结构层面，主要有 2 层。

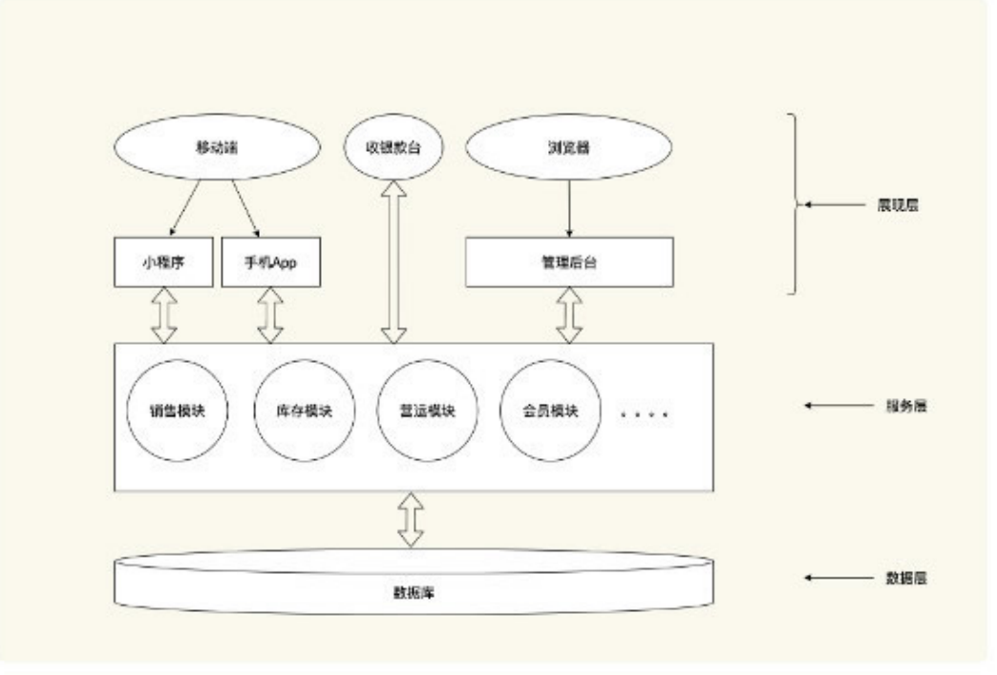
1. 第一层是商户。每个入驻的商户都有一个组号，所有与这个商户有关的数据，通过这个组号进行识别。
2. 第二层是分支机构。分支机构从属于商户，相同商户的分支机构有相同的组号。分支机构分为几种，包括总部、门店、配送中心等。门店又分为直营店和加盟店。每个分支机构有一个机构编号，同一分支机构的数据，有相同的组号和机构编号。

这样一来，新商户只需要开通账号，分配一个新的组号，就可以使用了。组号用于隔离商户之间的数据，使商户之间互相不影响。

最后，数据由我们统一进行运维，安全性有保障。商户自己不需要采购服务器，只需租用服务，资源的利用率更高。

系统的整体结构设计思路有了，那具体在应用层面如何实现呢？

我先用一张图来展示具体的应用构成：



这个图展示了应用的 3 个层级。

1. 展现层：包括门店收款机 App、移动端的手机 App、小程序，以及通过浏览器访问的后台管理部分。
2. 服务层：包括云端的销售模块、库存模块、营运模块、会员模块等。
3. 数据层：MySQL 数据库。

门店收款 App、移动端的手机 App，小程序等与数据库设计无关，我就不多说了。下面我重点介绍一下后台管理部分下面的服务层和数据层的相关内容。

服务层包括了销售、库存、营运、会员等管理模块。下面我就以库存管理中的盘点模块为例，详细介绍一下。因为这个模块比较简单，容易理解。

盘点，简单来说，就是把超市仓库里的商品都数一遍，然后跟电脑里的库存数比对，看看有没有不对的地方。实际数出来的库存数量叫做盘存数量，电脑中的库存数量叫做结存数量，比对的结果叫做盈亏数量。要是盘存数量比结存数量多，叫盘盈，否则叫做盘亏。

盘点操作是超市库存管理模块中的一个重要环节，是掌握实际库存的唯一办法。盘点盈亏数据也是衡量超市管理水平的重要指标。

盘点作业一般都在晚上门店营业结束以后进行。这也很好理解，毕竟，在白天营业的过程中，商品不断被顾客取走，又不断得到补充，库存处于一种变化状态，无法获取准确数据。

下面我来介绍下盘点的步骤。

1. 先生成一张盘点表，把当前系统中的库存数量提取出来，获得结存数量；
2. 把员工实际数出来的库存数据录入盘点表中，获得盘存数量；
3. 计算盈亏数量，公式是“盈亏数量 = 盘存数量 - 结存数量”；
4. 数据确认无误后，验收盘点表，并调整当前库存：库存数量 = 库存数量 + 盈亏数量。

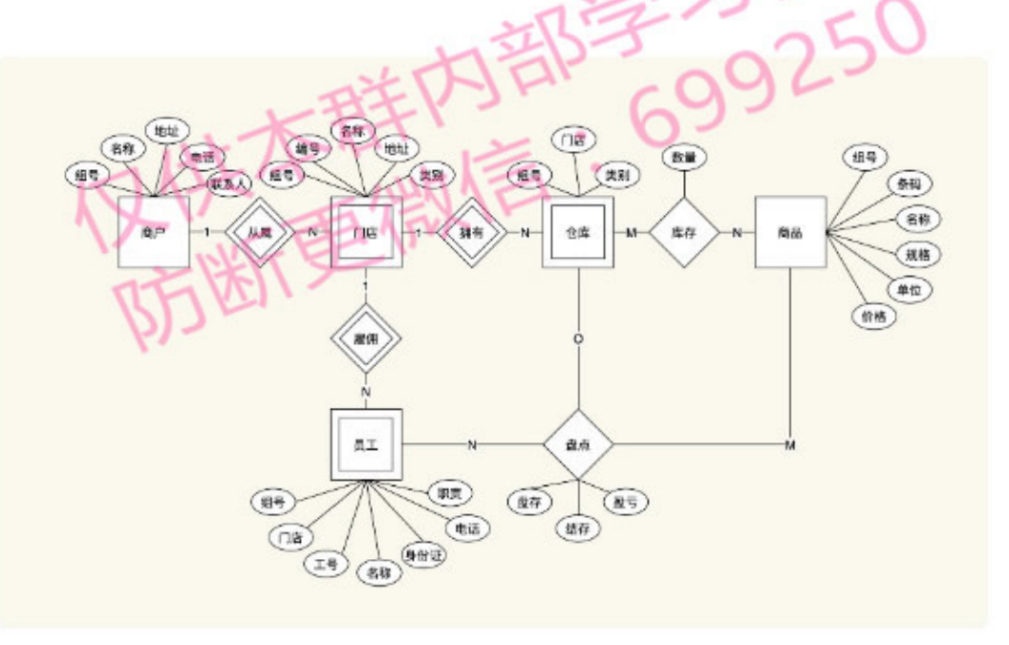
经过这些操作，系统中的库存数量与实际的库存数量就一致了，盘点盈亏也被记录下来了，体现在日报等报表中，超市经营者可以进行查看，为经营决策提供依据。

介绍完了盘点业务，现在回到数据库设计的主题上来，看看如何把盘点业务用数据表的形式表现出来。

盘点业务都是在门店进行，由员工实际操作，对仓库中的商品进行清点。因此，盘点业务模块中肯定要包含员工、门店、仓库、商品等实体。这个时候，我们就可以使用 ER 模型这个工具，来理清盘点模块的业务逻辑。

盘点模块的 ER 模型

我先把模型直接展示给你，一会儿我再带你具体分析一下。



首先，我们来分析下模型中的实体和关系。

这个 ER 模型中包括了 5 个实体，分别是：

1. 商户
2. 门店
3. 员工
4. 商品
5. 仓库

其中，商户和商品是强实体，门店、仓库和员工是弱实体。

这个 ER 模型中还包含了 5 个关系，我们按照 1 对多和多对多来分下类。

1 对多：

1. 商户与门店的从属关系
2. 门店与员工的雇佣关系
3. 门店与仓库的拥有关系

多对多：

1. 仓库与商品的库存关系
2. 仓库、商品和员工参与的盘点关系

接下来，我们再分析一下这 5 个实体各自的属性。

1. 商户：组号、名称、地址、电话、联系人。
2. 门店：组号、门店编号、名称、地址、电话、类别。
3. 员工：组号、门店编号、工号、名称、身份证、电话、职责。
4. 仓库：组号、门店编号、仓库编号、类别。
5. 商品：组号、条码、名称、规格、单位、价格。

除此之外，还有 2 个多对多关系的属性。

1. 仓库与商品的库存关系：库存数量。
2. 仓库、商品和员工参与的盘点关系：盘存数量、结存数量和盈亏数量。

通过建立 ER 模型，我们理清了业务逻辑。接下来，我们就可以把盘点业务中这些实体和关系落实到实际的数据表了。

ER 模型转换成数据表

你还记得在第 23 讲里学的转换规则吗？强实体和弱实体转换成独立的数据表，多对多的关系转换成独立的数据表，1 对多的关系转换成外键约束。

首先，我们把强实体转换成独立的数据表。

商户表（demo.enterprice）：

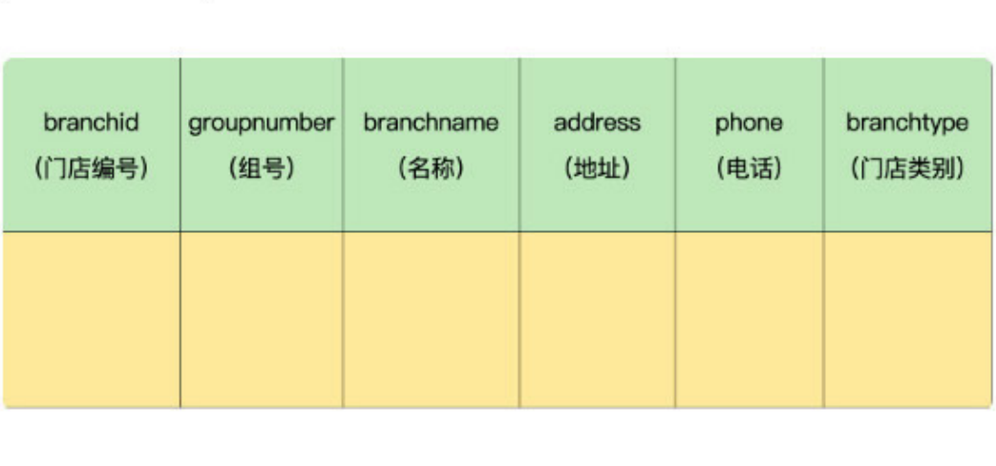
groupnumber （组号）	groupname （名称）	address （地址）	phone （电话）	contactor （联系人）

商品信息表（demo.goodsmaster）：

groupnumber （组号）	itemnumber （商品编号）	barcode （条码）	goodsname （名称）	specification （规格）	unit （单位）	salesprice （价格）

接着，我们把弱实体转换成独立的数据表。

门店表（demo.branch）：



branchid （门店编号）	groupnumber （组号）	branchname （名称）	address （地址）	phone （电话）	branchtype （门店类别）

员工表（demo.employee）：

employeeid （员工编号）	groupnumber （组号）	branchid （门店编号）	workno （工号）	name （名称）	pid （身份证号）	phone （电话）	duty （职责）

仓库表（demo.stockmaster）：

stockid （仓库编号）	groupnumber （组号）	branchid （门店编号）	stocknumber （仓库号）	stockkind （仓库类别）

第三步，把多对多的关系转换成独立的数据表。

库存表（demo.inventory）：

id （编号）	groupnumber （组号）	branchid （门店编号）	stockid （仓库编号）	itemnumber （商品编号）	itemquantity （库存数量）

盘点关系可以转换成 2 个表，分别是盘点单头表和盘点单明细表，这样做是为了满足第三范式的要求，防止冗余。

盘点单头表（demo.invcounthead）：

listnumber （单号）	groupnumber （组号）	branchid （门店编号）	stockid （仓库编号）	recorder （操作员）	recordingdate （盘点时间）

盘点单明细表（demo.invcountdetails）：

listnumber （单号）	groupnumber （组号）	branchid （门店编号）	stockid （仓库编号）	itemnumber （商品编号）	accquant （结存数量）	invquant （盘存数量）	plquant （盈亏数量）

这样一来，我们就把 ER 模型中的实体和多对多的关系，转换成了独立的数据表。

这里你要注意的是，我在盘点单明细表中保留了组号和门店编号。这是因为，虽然这 2 个字段是冗余数据，但是可以加快查询的速度，经过权衡利弊，最后决定还是加上。

在我把 1 对多的关系转换成外键约束之前，我们还要进行一项重要的工作：分库分表。因为外键约束与数据表以及表中的字段有关，分库分表会影响到表和表中的字段。

而且外键约束需要在创建数据表的时候创建，所以咱们下节课和创建数据表一起讲，这节课我们先学习下分库分表。

在前面的课程中，每节课我们都是以具体技术点为核心展开的。而分库分表，只有在进行数据库系统整体设计的阶段才会用到。所以，今天我就结合咱们这个项目的系统设计，来给你具体讲一讲如何进行分库分表。

为什么要分库分表呢？当数据量足够大的时候，即便我们把索引都建好，系统资源调优到极致，仍然有可能遇到运行缓慢、CPU 使用率居高不下的情况。因为单个数据库中单个表的数据量高到一定程度，超过了系统的承载能力。

面对这种情况，我们有 2 种选择：一种是购买更多的资源，增加内存，增加 CPU 的算力，但是这样会增加系统的成本。这个时候，我们就可以用另一种方法，也就是接下来我要讲的分库分表。

如何进行分库分表？

所谓的分库分表，其实就是把大的数据库拆成小数据库，把大表拆成小表，让单一数据库、单一数据表的数据量变小。这样每次查询时，需要扫描的数据量减少了，也就达到了提升查询执行效率的目的。

分库分表又可以分成垂直分表、垂直分库、水平分库和水平分表。

垂直分表

所谓垂直分表，就是把一个有很多字段的表，按照使用频率的不同，拆分成 2 个或多个表。

为了帮助你理解，还是用我们的盘点模块中的表来演示说明一下。

每个商户都有一个自己的商品信息表，数据量比较大。所以，我们可以拆分下这个表。我们把经常使用的字段条码、名称和价格，拆分成商品常用信息表（demo.goods_o）；把剩下的字段，也就是规格和单位拆分成商品不常用信息表（demo.goods_f）。

商品常用信息表：

groupnumber （组名）	itemnumber （商品编号）	barcode （条码）	goodsname （名称）	salesprice （价格）

商品不常用信息表：

groupnumber （组名）	itemnumber （商品编号）	specification （规格）	unit （单位）

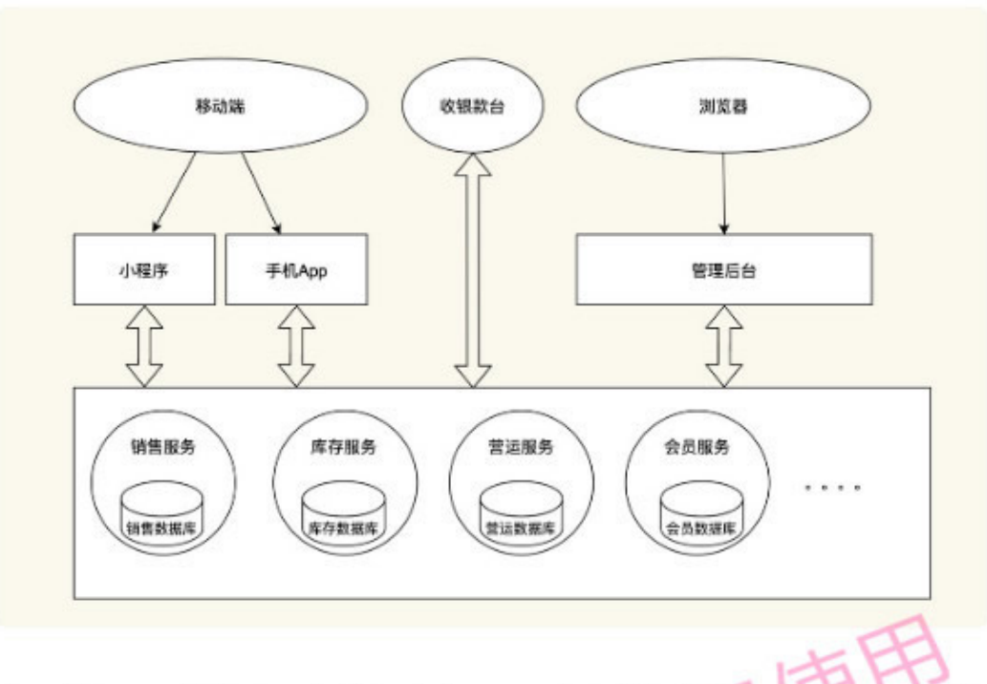
至于商户表、门店表、员工表、仓库表，这些表的数据量有限，不需要拆分。库存表、盘点单头表和盘点单明细表，虽然数据量大，但是评估之后，我们发现字段的使用频率都很高，拆分的价值不大，所以也不需要拆分。

下面我再介绍一下什么是垂直分库。

垂直分库

垂直分库的意思是，把不同模块的数据表分别存放到不同的数据库中。这样做的好处是，每个数据库只保存特定模块的数据，系统只有用到特定模块的数据时，才会访问这个数据库。这样就减少了数据库访问的次数，就相当于是把数据访问的流量分散了。

这个可能不太好理解，我来画一个简单的示意图：



在这个图中，数据不再存储在一个数据库中，而是根据业务模块的不同，分别存储在不同的数据库中，比如销售数据库、库存数据库、营运数据库、会员数据库等。这样一来，业务模块可以主要与自己的数据库进行数据交互。业务内的数据交互多了，业务与业务之间的数据交互就可以大大减少了。

这样做的好处主要有 3 个：

1. 单个数据库的数据量减小了；
2. 单个数据库的访问流量分散了；
3. 系统整体的故障风险减小了。

下面我再介绍一下什么是水平分库和水平分表。

水平分库和水平分表

当垂直分表已经穷尽，垂直分库也不能再拆分的时候，我们还可以做水平分库和水平分表。

水平分表的意思是，把数据表的内容，按照一定的规则拆分出去。

盘点数据会不断累积，数据量越来越大。为了提升系统效率，我们制定了水平分表的策略。

第一步，我们把盘点单头表和盘点单明细表水平拆分：把验收处理过的盘点单头表和盘点单明细表拆分到盘点单头历史表和盘点单明细历史表。

这样做的好处是，盘点单头表和盘点单明细表经常需要进行插入、删除和修改操作，只保留当前正在处理的数据，可以提升效率，避免在一个不断增长的大表中进行 DML 操作。

而盘点单头历史表和盘点单明细历史表中的数据虽然不断增长，但数据不会修改，只进行查询操作。用经常作为筛选条件的字段创建索引，可以大大加快查询的速度。

拆分出来的盘点单头历史表（demo.invcountheadhist）与盘点单头表类似，不同之处是增加了验收人编号（confirmer）和验收日期（confirmationdate）。

盘点单头表历史表：

listnumber （单号）	groupnumber （组号）	branchid （门店编号）	stockid （仓库编号）	recorder （操作员）	recordingdate （盘点时间）	confirmer （验收员）	confirmationdate （验收日期）

拆分出来的盘点单明细历史表（demo.invcountdetailshist）的字段则与盘点单明细表一样。

盘点单明细历史表：

listnumber （单号）	groupnumber （组号）	branchid （门店编号）	stockid （仓库编号）	itemnumber （商品编号）	accquant （结存数量）	invquant （盘存数量）	plquant （盈亏数量）

第二步，我们把组号大于 500、小于 1000 的商户数据，拆分到另外的数据表里进行保存。这里的数字是我们根据对入驻平台商户的数据量进行评估之后得出的，在实际工作中，你可以根据实际情况来决定。**原则是：确保单个数据表中的数据量适中，不会成为操作的瓶颈。**

这样，我们就完成了对盘点模块中数据表的水平拆分。

接下来，我们来进行水平分库。水平分库的目的是使单个数据库中的数据量不会太大。这样可以确保我们设计出来的数据库，在大数据环境下，也能高效运行。

水平分库的意思与水平分表类似，就是按照一定的规则，把数据库中的数据拆分出去，保存在新的数据库当中。

新的数据库可以在相同的服务器上，也可以在不同的服务器上。比如，我们可以把组号大于 500、小于 1000 的用户数据拆分出来，保存到新的服务器的数据库中。不过，保存到新的服务器，也就意味着增加系统的开销。

因此，我们可以以 500 个商户为单位，每 500 个商户，在相同的服务器上创建一套新的数据库；每 5000 个商户，购置新的服务器。这样，我们就完成了对数据库进行分库的设计。

总结

这节课，我们一起设计了一个基于云服务的连锁超市管理系统数据库。你要重点掌握如何进行需求分析、如何把分析的结果转换成数据库的设计，以及如何在总体设计的阶段通过使用分库分表使设计出来的数据库能够处理大量数据。

在实际项目中，你要重点关注 3 个方面。

- 第一，要充分理解项目需求。有的时候，客户自己也不清楚自己的需求。这个时候，就需要你帮助客户理清思路。你可以把客户的需求用图表等方式整理出来，再跟客户一起讨论。在这个阶段，投入较多的时间是值得的，如果等系统开发完成之后再改，成本就很高了。
- 第二，使用 ER 模型工具来整理思路，可以提高效率，提高设计的质量。
- 第三，要充分考虑到系统投入运行之后的承载能力。如果有可能处理大量数据，就需要考虑分库分表的策略。

最后，还有几点你需要注意下。

1. 分库分表的策略，需要在设计阶段完成。如果缺乏整体分库分表的策略而匆忙上线，遇到瓶颈时再解决，花费的成本要远远高于在设计阶段投入的时间成本。
2. 分库分表的策略，必然带来开发和运维方面成本的提升，因此，你需要在设计阶段就有一个整体规划。比如，在类的设计里面，用正则表达式计算访问的服务器、数据库和数据表的名称。
3. 分库分表一般适用于比较大的项目，如果你开发的应用数据量小，系统规模有限，团队成员不多，不如就用一个服务器、一个数据库。因为分库分表对数据量小的项目没有什么作用，却会大大提升开发的复杂度，增加开发、运维和项目管理的成本。所以，你要综合考虑利与弊。

思考题

假设我有一个数据库 demo，当中有一个商品流水表（demo.trans），示例如下：

transno （流水单号）	itemnumber （商品编号）	goodsname （商品名称）	quantity （数量）	actualvalue （金额）	wechatvalue （微信支付）	cashvalue （现金支付）	groupnumber （组号）	branchnumber （门店编号）
4521	1	笔	1	89	89	0	1	1
245112	1	纸	1	10	10	0	2	2

在设计阶段，如果预见到未来数据量会非常庞大，你会如何制定分库分表策略？

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(1)

bearlu

数据库表的时间必须设置时间还是使用时间戳？

2021-05-15