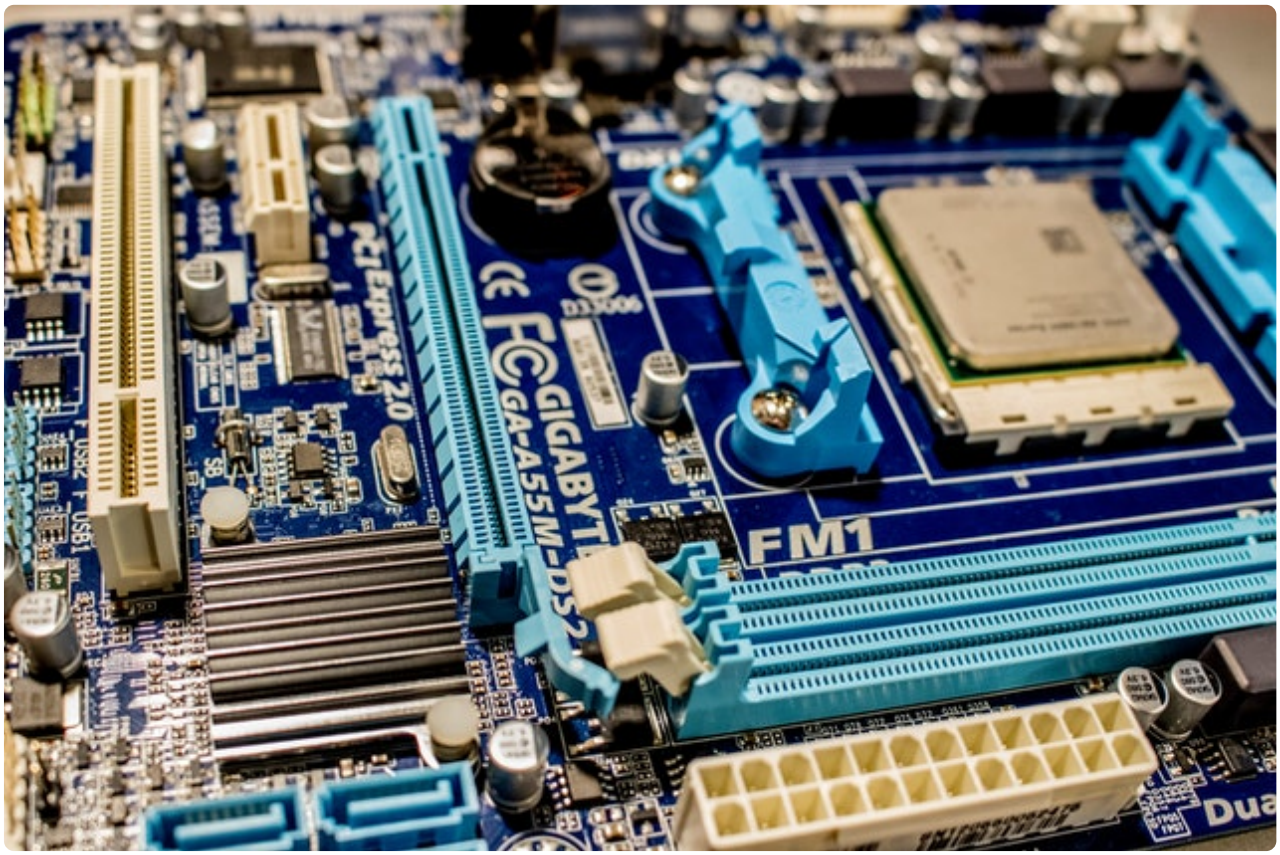
33 为大型电商平台设计高可用数据库系统

更新时间：2020-05-21 16:16:13



> 劳动是一切知识的源泉。——陶铸

电商业务可能是除了工作之外，我们接触最多的业务系统了，例如：京东、天猫、淘宝等等。浏览商品、加入购物车、购买结算等等都是电商平台中最基本的业务，那么，如果让你设计一个高可用的电商业务系统（数据表），你会怎么做呢 ？ 当然，你不需要考虑的 “非常全面”，毕竟业务系统都是在不断迭代（升级）中的，不要总想着一次性把 “所有的东西” 都设计出来。

1. 电商业务概述

可以肯定的说，电商平台是日常工作、生活中遇到的比较复杂的业务系统，这体现在它面向的是广大的用户群体（用户应用由于面对各种各样的人群，自然也就会有各种各样的需求）。在实际的做设计之前，一定要仔细斟酌这里面的业务思想、业务需求，最好当然是能够 “预见未来”，以使得在不改变数据表结构的前提下做升级迭代。

1.1 电商平台通常都定义了哪些业务

大家基本上对电商平台（你应该有过网购的经历）都不会很陌生，所以，我直接进入主题。我们当前要设计的电商平台，包含以下的几个功能点（你当然可以在此基础上做扩充，并完善它们所需要的表设计）：

- 用户相关信息
    - 用户基本信息，例如：姓名、年龄、邮箱、电话（用户的基本信息收集的越详细越好，思考下这是为什么

呢？）
    - 用户收货地址信息，例如：省、市、收件人、电话等等（可以做区县、镇、乡等等级别的扩展）
- 商品相关信息
    - 商品基本信息，例如：商品名、商品类型（食品、家居、生鲜等等）、商品价格、商品库存等等
    - 商品厂商信息，例如：厂商名称、厂商 logo、厂商审核状态等等
- 用户与商品之间的“互动”信息
    - 收藏关注，用户可以收藏商品，也可以收藏厂商（大部分电商系统会“重命名”为店铺）
    - 购物车，用户可以把自己想要购买的商品加入到购物车中，同时可以指定个数
    - 购买订单，用户可以购买商品，并记录支付信息
    - 评价反馈，对于已完成（这是前提条件）的订单，用户可以发表对商品的评论
- 优惠空间
    - 折扣活动，自定义开始和结束时间，对应于某种商品采取打折促销

可以看到，我们当前定义的功能点足够支撑一个简单电商平台的运营了，接下来，就需要根据这里提出的业务需求去设计数据表。另外，需要知道，对于绝大多数的业务系统而言，数据表是最难完成的工作（因为业务基本都是对表的 CRUD，而这些通常就是简单的 if、else、for 逻辑代码）。

1.2 设计电商业务数据表应该注意些什么

其实，设计 MySQL 表需要注意的地方不仅仅是针对于“某一个”业务系统，这些建议或者意见放在任意一个系统中基本（肯定是会有一些特殊的场景需要有特殊的手段去应对）都是成立的。下面，我将总结一些表设计的注意事项：

- 除非是数据报表，否则，每张表都需要定义主键，且最好是自增的
- 主键最好使用 bigint 类型，以便业务膨胀
- 谨慎使用外键，除非主表与母表“都不大”
- 对于可能会增加选项的枚举不要使用 enum 类型
- 所有的字段都应该是 NOT NULL 的，且最好定义默认值
- 谨慎使用 TEXT、BLOB 等大数据类型
- 单张表的数据列不应该太多，最好不要超过50个（不具有特殊含义）
- 单张表的索引个数不应该太多，最好不要超过5个
- 选择“足够用”的字段类型就可以，不要浪费存储空间
- 不要偷懒，表和列都要给出注释信息，否则，只能去代码中理解它的含义
- 选择合适的字符集，无 emoji 使用 utf8，有 emoji 使用 utf8mb4，但是肯定不建议使用 emoji
- 时间字段选择 timestamp 或 datetime 都可以
- 与用户相关的数据表一定要涵盖 user_id 逻辑外键
- 库名、表名、字段名都应该是小写

在工作、学习中，你会遇到各种各样的问题，自然也就会有一些自己的见解和总结，如果能对我这里的“注意事项”进行一些补充（当然，也可以提出一些意见和建议），那就最好不过了。

2. 电商业务表设计

理解了电商的业务思想和注意事项，我们就可以去着手设计数据表了。这里，我把业务中所有的数据表分为两类：字典表和业务表。字典表是与具体的业务无关的，例如地域字典、支付方式字典等等；业务表则是与具体的业务系统强关联的。

由于电商业务是一个独立的系统，我们最好单独创建一个库用于存储它的所有数据表。如下所示，创建 e_commerce 库：

```
-- 创建数据库 e_commerce
CREATE DATABASE IF NOT EXISTS `e_commerce`;
```

2.1 字典表设计

我一共设计了三张字典表：地域字典表、支付方式字典表、商品分类字典表。对于字典表来说，它们是通用的，即可以把它们应用在其他的业务系统中。首先，我们先去创建“地域字典表”，建表语句如下：

```
-- 地域字典表
CREATE TABLE IF NOT EXISTS `e_commerce`.`district_dict` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT COMMENT '记录id, 主键, 自增',
  `district_id` int(11) NOT NULL COMMENT '地域 id',
  `parent_district_id` int(11) NOT NULL COMMENT '父地区关联, 如果是一级区域, 父地区 id 为0',
  `district_name` varchar(48) NOT NULL COMMENT '地区名称',
  `district_level` tinyint(4) NOT NULL COMMENT '区域层次级别: 0是省; 1是市',
  PRIMARY KEY (`id`),
  UNIQUE KEY `district_id_idx` (`district_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='地域字典表';
```

从表设计和注释中可以看出，我们的“地域字典”包含省和市，这基本已经够用了。如果想要“更深”的层次，可以自行扩展到区县、镇、乡等等。需要注意，`district_id` 这个字段可以唯一的标识一个城市，它的值可以使用代码生成，也可以人为指定。

“支付方式”指的是用户在订单中选择的结算方式，常见的有：微信支付、支付宝支付、银联支付（中国银行、建设银行）等等。把这些支付方式硬编码在业务代码中肯定不会是比较好的选择，所以，我们需要一张 “支付方式字典表”。

```
-- 支付方式字典表
CREATE TABLE IF NOT EXISTS `e_commerce`.`e_payment_type` (
  `id` tinyint(4) NOT NULL AUTO_INCREMENT COMMENT '自增主键',
  `payment_method` varchar(64) NOT NULL DEFAULT '' COMMENT '支付方式',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='支付方式字典表';
```

这张表非常简单，没有额外多余的字段（需要的话可以自行填充）。与之类似，每一种商品也都会（至少）属于一个分类，例如：食品、家居、生鲜、酒水等等。我们也需要为此创建一张“商品分类字典表”。

```
-- 商品分类字典表
CREATE TABLE IF NOT EXISTS `e_commerce`.`e_goods_type` (
  `id` tinyint(4) NOT NULL AUTO_INCREMENT COMMENT '自增主键',
  `goods_type` varchar(64) NOT NULL DEFAULT '' COMMENT '商品类型',
  `remark` varchar(200) NOT NULL DEFAULT '' COMMENT '其他说明',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='商品分类字典表';
```

字典表当然是为业务表服务的，例如：对于地域信息而言，业务中可能会有多处场景需要设置。在没有 “地域字典表” 之前，只能重复填充所有的地域信息。同时，字典表与业务也并不是绑定关系，它们可以应用于任何其他业务中。

2.2 业务表设计

对于每一个业务系统而言，用户表都是必不可少的基本表。这里我先给出 “用户表” 的建表语句，之后再对它进行解释说明。如下所示：

```
-- 用户表
CREATE TABLE IF NOT EXISTS `e_commerce`.`e_user` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '自增主键',
  `user_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '用户 id',
  `user_name` varchar(128) NOT NULL DEFAULT ' ' COMMENT '用户名',
  `user_age` tinyint(4) NOT NULL DEFAULT '0' COMMENT '用户年龄',
  `user_status` enum('normal','paused','deleted') NOT NULL DEFAULT 'normal' COMMENT '用户状态: normal-正常, paused-暂停/锁定, deleted-删除',
  `user_type` varchar(20) NOT NULL DEFAULT 'normal' COMMENT '用户类型: normal-普通用户, superadmin-超管, admin-普通管理员',
  `user_real_name` varchar(50) NOT NULL DEFAULT '' COMMENT '用户真实姓名',
  `user_email` varchar(100) NOT NULL DEFAULT '' COMMENT '用户邮箱',
  `user_mobile` varchar(20) NOT NULL DEFAULT '' COMMENT '用户手机号',
  `user_company` varchar(50) NOT NULL DEFAULT '' COMMENT '用户公司',
  `user_department` varchar(45) NOT NULL DEFAULT '' COMMENT '用户部门',
  `user_duty` varchar(100) NOT NULL DEFAULT '' COMMENT '用户具体职责',
  `remark` varchar(200) NOT NULL DEFAULT '' COMMENT '其他说明',
  `last_login_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' COMMENT '上次登录时间',
  `expire_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' COMMENT '过期时间(默认一年)',
  `create_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' COMMENT '创建时间',
  `update_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' COMMENT '更新时间',
  PRIMARY KEY (`id`),
  UNIQUE KEY `user_id_idx` (`user_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='用户信息表';
```

可以看到，用户表非常 “详细”，它包含的数据列非常多。之所以需要获取 “如此多的” 用户信息，是为了可以对用户进行分析，并使用 “推荐系统” 给用户推荐商品或促销信息。这里需要特别注意的是 `last_login_time` 和 `expire_time` 字段，前者代表上一次访问系统的时间；而后者代表用户多久没有登录系统会被暂停使用。

用户注册之后，可以创建收货地址，所以，我们需要一个 “收货地址表”。建表语句如下：

```
-- 收货地址
CREATE TABLE IF NOT EXISTS `e_commerce`.`e_address` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '自增主键',
  `user_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '用户 id',
  `district_id` int(11) NOT NULL DEFAULT '0' COMMENT '地域 id',
  `detailed` varchar(200) NOT NULL DEFAULT '' COMMENT '详细的门牌地址',
  `name` varchar(128) NOT NULL DEFAULT ' ' COMMENT '收件人',
  `mobile` varchar(20) NOT NULL DEFAULT '' COMMENT '手机号',
  `create_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' COMMENT '创建时间',
  `update_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' COMMENT '更新时间',
  PRIMARY KEY (`id`),
  KEY `user_id_idx` (`user_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='用户地址表';
```

其中，`district_id` 则直接引用了 `district_dict` 字典表中的 `district_id` 字段。另外，需要注意，不应该给 `user_id` 创建唯一索引，因为一个用户可以创建多个收货地址。

电商平台当然可以上架很多 “商品”，而这些商品一定会属于某一个生产厂商或者叫店铺。所以，为了 “上架” 商品，我们需要两张表：商品生产厂商信息表、商品表。“商品生产厂商信息表” 比较简单，只需要记录厂商的基本信息就可以了。建表语句如下所示：

```sql
-- 商品厂商信息
CREATE TABLE IF NOT EXISTS `e_commerce`.`e_company` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '自增主键',
  `company_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '商品生产厂商 id',
  `company_name` varchar(64) NOT NULL DEFAULT '' COMMENT '商品生产厂商名称',
  `company_logo_url` varchar(1024) NOT NULL DEFAULT '' COMMENT '商品生产厂商 logo 地址',
  `company_audit` enum('review', 'approve','reject') NOT NULL DEFAULT 'review' COMMENT '审核状态: review-审核中, approve-通过, reject-拒绝',
  `remark` varchar(200) NOT NULL DEFAULT '' COMMENT '厂商说明',
  `create_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' COMMENT '创建时间',
  `update_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' COMMENT '更新时间',
  PRIMARY KEY (`id`),
  UNIQUE KEY `company_id_idx` (`company_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='商品生产厂商信息表';
```

接下来，就可以去创建商品表了（它会引用到两张表：e_goods_type、e_company）：

```sql
-- 商品表
CREATE TABLE IF NOT EXISTS `e_commerce`.`e_goods` (
  `goods_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '商品 id',
  `goods_name` varchar(64) NOT NULL DEFAULT '' COMMENT '商品名称',
  `goods_type` int(11) NOT NULL DEFAULT '0' COMMENT '商品分类',
  `goods_price` decimal(10,4) NOT NULL COMMENT '商品价格, 最多支持四位小数',
  `goods_volume` bigint(20) NOT NULL DEFAULT '0' COMMENT '商品供应量',
  `goods_count` bigint(20) NOT NULL DEFAULT '0' COMMENT '商品库存',
  `goods_icon_url` varchar(1024) NOT NULL DEFAULT '' COMMENT '商品图标地址',
  `landing_page_url` varchar(1024) NOT NULL DEFAULT '' COMMENT '商品信息落地页(详细信息页面)',
  `goods_company_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '商品生产厂商 id',
  `remark` varchar(200) NOT NULL DEFAULT '' COMMENT '其他说明',
  `create_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' COMMENT '创建时间',
  `update_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' COMMENT '更新时间',
  PRIMARY KEY (`goods_id`, `goods_type`),
  KEY `goods_company_id_idx` (`goods_company_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='商品表' PARTITION BY HASH(goods_type) PARTITIONS 16;
```

注意到，商品表是分区表，这是因为它是电商平台的核心业务表，可能会非常大。我这里以 `goods_type` 作为（哈希）分区条件，一共将数据分散到16个分区中。同时，也正是由于 `goods_type` 是分区条件，它必须出现在主键中。

看到喜欢的商品或店铺我们可以收藏下来，这种功能叫做“收藏关注”。由于要保存这些收藏记录，当然也就需要一张数据表。建表语句如下：

```sql
-- 收藏关注表
CREATE TABLE IF NOT EXISTS `e_commerce`.`e_attention` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '自增主键',
  `user_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '用户 id',
  `attention_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '关注的商品或厂商的 id',
  `attention_type` enum('goods', 'company') NOT NULL DEFAULT 'goods' COMMENT '关注类型: 商品或厂商',
  `deleted` enum('normal', 'deleted') NOT NULL DEFAULT 'normal' COMMENT '状态: normal-正常, deleted-删除(取关)',
  `create_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' COMMENT '创建时间',
  `update_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' COMMENT '更新时间',
  PRIMARY KEY (`id`),
  KEY `user_id_idx` (`user_id`),
  UNIQUE KEY `attention_id_type_idx` (`attention_id`, `attention_type`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='收藏关注表';
```

需要注意，这张表既可以存储关注的店铺，也可以存储关注的商品，我这里使用了一个枚举类型的 `attention_type` 来进行区分。同时，还设计了一个“取关（deleted）”的字段，即使用户取关，也要记录下他曾经关注过。

购物车与“收藏关注”是非常类似的概念，它用于方便用户下单购买商品（同时购买很多商品）。购物车表也是比较简单的，记录用户、商品、个数就可以了，建表语句如下：

```
-- 购物车
CREATE TABLE IF NOT EXISTS `e_commerce`.`e_shopping_cart` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '自增主键',
  `user_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '用户 id',
  `goods_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '商品 id',
  `goods_count` int(11) NOT NULL DEFAULT '0' COMMENT '加入购物车个数',
  `create_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' COMMENT '创建时间',
  `update_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' COMMENT '更新时间',
  PRIMARY KEY (`id`),
  KEY `user_id_idx` (`user_id`),
  KEY `goods_id_idx` (`goods_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='购物车表';
```

很多店铺或商品会在节假日做促销活动，会有一定的打折优惠，这种业务场景叫做“活动”。但是，活动的形式“五花八门”，几乎每个电商平台都不一样。所以，我这里设计一个比较简单的“活动表”来做示例说明，只是标记了在某个时间范围内，某种商品会有折扣。建表语句如下：

```
-- 活动表
CREATE TABLE IF NOT EXISTS `e_commerce`.`e_activity` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '自增主键',
  `activity_name` varchar(128) NOT NULL DEFAULT '' COMMENT '活动名称',
  `start_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' COMMENT '活动开始时间',
  `end_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' COMMENT '活动结束时间',
  `audit_status` enum('review', 'approve','reject') NOT NULL DEFAULT 'review' COMMENT '审批状态: review-审核中, approve-通过, reject-拒绝',
  `audit_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' COMMENT '审批时间',
  `audit_comment` varchar(256) NOT NULL DEFAULT '' COMMENT '审批批注',
  `audit_by_user` bigint(20) NOT NULL DEFAULT '0' COMMENT '审批人',
  `deleted` enum('normal', 'deleted') NOT NULL DEFAULT 'normal' COMMENT '活动状态: normal-正常, deleted-删除',
  `goods_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '商品 id',
  `goods_discount` decimal(10,4) NOT NULL COMMENT '商品折扣, 最多支持四位小数',
  `create_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' COMMENT '创建时间',
  `update_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' COMMENT '更新时间',
  `create_by` bigint(20) NOT NULL DEFAULT '0' COMMENT '创建用户 id',
  `update_by` bigint(20) NOT NULL DEFAULT '0' COMMENT '修改用户 id',
  PRIMARY KEY (`id`),
  KEY `goods_id_idx` (`goods_id`),
  UNIQUE KEY `activity_name_idx` (`activity_name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='活动表';
```

有了用户、商品、活动等等基本的业务表，我们就可以下单去购买商品了，即需要一张订单表。通常，订单表都会设计的非常复杂，包含的信息非常多，主要原因是它与 “钱” 相关（当然，除了数据表记录，业务日志也是少不了的）。订单表建表语句如下：

```
-- 订单表
CREATE TABLE IF NOT EXISTS `e_commerce`.`e_order` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '自增主键',
  `order_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '订单 id',
  `user_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '用户 id',
  `goods_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '商品 id',
  `goods_count` int(11) NOT NULL DEFAULT '0' COMMENT '商品个数',
  `order_price` decimal(10,4) NOT NULL COMMENT '订单价格, 最多支持四位小数',
  `order_status` enum('init', 'waiting', 'timeout','completed') NOT NULL DEFAULT 'init' COMMENT '订单状态: init-初始化, waiting-等待付款, timeout-超时,
 completed-已完成',
  `order_payment_type` tinyint(4) NOT NULL DEFAULT '0' COMMENT '付款方式',
  `activity_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '参与的活动 id',
  `address_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '用户地址 id',
  `remark` varchar(200) NOT NULL DEFAULT '' COMMENT '其他说明(折扣、活动等等说明)',
  `create_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' COMMENT '创建时间',
  `update_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' COMMENT '更新时间',
  PRIMARY KEY (`id`),
  KEY `order_id_idx` (`order_id`),
  KEY `user_id_idx` (`user_id`),
  KEY `goods_id_idx` (`goods_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='订单表';
```

关于订单表，需要注意这样几个地方：

- `user_id` 是 `e_user` 表的逻辑外键
- `goods_id` 是 `e_goods` 表的逻辑外键
- `order_payment_type` 是 `e_payment_type` 表的逻辑外键
- `activity_id` 是 `e_activity` 表的逻辑外键
- `address_id` 是 `e_address` 表的逻辑外键

同时，需要注意这张表的 `order_status` 字段，它是枚举类型，其中有个选项是 timeout，代表的是：用户创建了订单，在一定时间（例如半个小时、一个小时等等）内没有完成支付，这个订单就算做超时不可用了。

好的，现在只剩下最后一张表了：评价反馈表。这张表也是比较简单的，它需要关联用户表和订单表，且只能是“已完成” 的订单才可以做评价。建表语句如下：

```
-- 评价反馈表
CREATE TABLE IF NOT EXISTS `e_commerce`.`e_feedback` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '自增主键',
  `user_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '用户 id',
  `order_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '订单 id',
  `star_level` tinyint(4) NOT NULL DEFAULT '0' COMMENT '星级',
  `content` varchar(200) NOT NULL DEFAULT '' COMMENT '评价内容',
  `pic_url` varchar(1024) NOT NULL DEFAULT '' COMMENT '晒图地址',
  `remark` varchar(200) NOT NULL DEFAULT '' COMMENT '其他说明(折扣、活动等等说明)',
  `create_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' COMMENT '创建时间',
  `update_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' COMMENT '更新时间',
  PRIMARY KEY (`id`),
  KEY `user_id_idx` (`user_id`),
  KEY `order_id_idx` (`order_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='评价反馈表';
```

设计数据表一定是在深入理解业务思想基础之上的，这是最核心也是最难的开发过程。这不仅仅需要你足够理解 MySQL，还需要你有足够的经验，毕竟 “规则” 并不一定适用所有的场景，要讲求灵活应变。

3. 不适用 MySQL 的存储业务

不一定所有的存储业务都需要使用 MySQL，毕竟 MySQL 也会有很多限制。有些场景下，前期使用 MySQL 比较“顺手”，但是，逐渐地性能越来越差。此时，再去做技术选型、数据迁移等等都会浪费大量的时间和精力。所以，提前做好准备，知道哪些场景可以使用 MySQL，哪些场景又不能使用 MySQL。

3.1 涉及到存储，为什么不用 MySQL

我们应该知道，“大数据量”存储会让 MySQL 的读写性能急剧下降。虽然有分区表、分库分表等等技术方案，但无疑都是非常麻烦且难以维护的。所以，对于数据膨胀快速的业务场景，MySQL 几乎是不适用的。

如果有一些数据，经常需要做插入和删除操作，那么，也并不适合使用 MySQL 存储。虽然可以对数据做标记删除（设计一个 `deleted` 字段即可），但是，大量的无用数据会浪费很多存储空间。

3.2 场景举例说明

报表型数据是“大数据量”的典型代表，通常情况下，电商平台每天需要入库的报表数据在5万行量级，随着业务发展，数据量大概率会进一步膨胀。所以，对于这类业务场景，最好是使用分布式数据库或报表型数据库存储，例如：HBase、Palo 等等。

“浏览记录”也是电商平台的常见业务需求，但是由于这类数据随着时间跨度的增长，价值也会相应降低。所以，通常会以时间单位或数字单位去做保留，例如：近一个月的浏览记录、近100条浏览记录。如果使用 MySQL 来存储，虽然可以应用 `WHERE` 或 `LIMIT` 子句对结果进行限定，但是会浪费大量的存储空间。所以，对于这种类似的需求，使用 Redis 这样的缓存系统是更好的。

4. 总结

对于业务系统的表设计来说，过程都是类似的：首先理清楚业务需求是什么，即需要做什么；接着梳理各个需求之间的关联关系（就像订单与商品一样），即怎样去做；最后再去书写表的创建语句，并验证是否符合要求。所以，表设计的是否合理，除了日常的学习之外，更多的还是经验的积累。

5. 问题

除了当前的业务需求设计，你还能做怎样的扩展呢 ？

随着业务发展，订单表会越来越大，你会怎么解决这个问题呢 ？

对于目前的数据表设计，你觉得合理吗 ？如果不合理，又为什么呢 ？

6. 参考资料

《高性能 MySQL（第三版）》

MySQL 官方文档：Data Types

MySQL 官方文档：Optimization

MySQL 官方文档：Character Sets and Collations in MySQL

}

32 一起探究下事务的实现原理吧

34 为慕课网设计高可用数据库系统