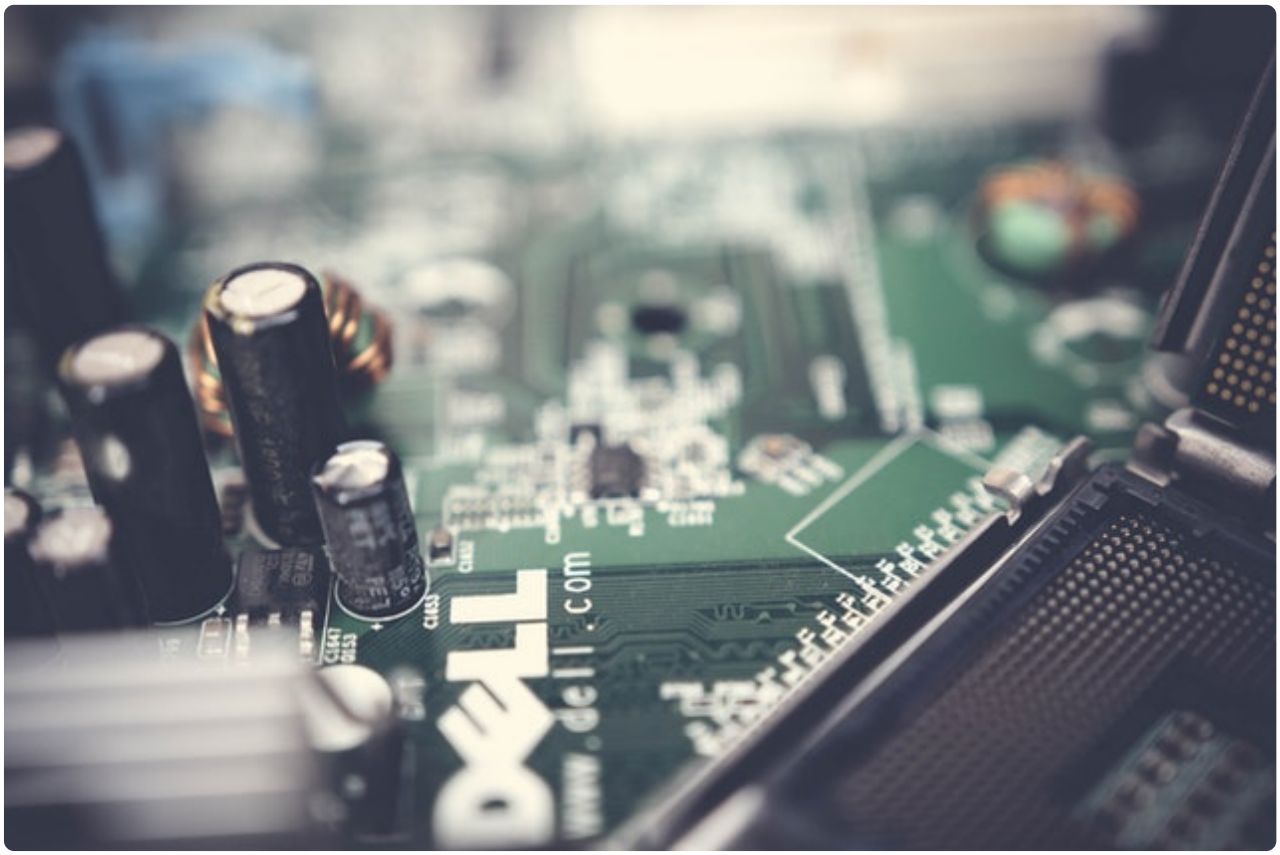
34 为慕课网设计高可用数据库系统

更新时间：2020-05-20 09:33:49



> 成功＝艰苦的劳动＋正确的方法＋少谈空话。——爱因斯坦

想必大家都已经对慕课网很熟悉了，这确实是一个比较“复杂”的系统，它里面定义了很多功能。在不看这一节后面的内容之前，思考一下，如果让你给慕课网业务系统设计数据表，你会怎样设计呢？当然，我还是会建议你，不要总想着“一步到位”。系统是在不断迭代的，功能也是逐渐添加的，初期的设计，不应该是很复杂的。

1. 慕课网业务概述

同样，在开始设计数据表之前，先来理清楚我们需要做什么，即需求是什么样的。当然，几乎每个系统都有它自己的特殊之处，我们也需要在“动手”之前梳理出来。好的，那下面就开始“慕课网”之旅吧。

1.1 慕课网都定义了哪些业务

想要理解慕课网的系统设计，最好的方式当然是在浏览器中打开并到处“点一点”。这里，我将整个系统的需求设计分为五类，下面，依次对这五类进行讲解。

- 字典信息
    - 课程分类字典，例如：免费课程、实战课程、专栏、手记等等
    - 课程分级字典，例如：初级、中级、高级等等
    - 内容分类字典，例如：Java、Python、SpringBoot、SpringCloud 等等

    - 内容格式字典，例如：markdown、txt 等等
    - 支付方式字典，例如：微信支付、支付宝支付、银联支付等等

- 用户相关信息

    - 用户信息，例如：姓名、年龄、邮箱、手机号、职业等等
    - 用户粉丝，即关注当前用户的用户
    - 用户消息，即推送给用户的消息

- 课程相关信息

    - 课程信息，例如：名称、分类、时长、学习人数、评分等等
    - 课程内容信息，每一个课程都会有对应的内容数据
    - 课程问题，用户学习课程时可以针对课程提问题

- 用户与课程相关的信息

    - 用户收藏课程信息，针对喜欢的课程，用户可以收藏
    - 用户课程购物车，想要购买的课程可以加入到购物车中
    - 用户课程订单，购买课程记录
    - 用户课程，用户学习的课程
    - 课程评价，针对学习的课程，可以进行评价

- 其他

    - 手记，类似于博客
    - 推送消息，即人为设定的在某个时间段内推送给所有（种子）用户的消息

可以看出，慕课网的业务设计还是非常复杂的，且这里面有很多处涉及到表与表之间的关联。另外，除了这里的设计，你还知道慕课网的其他业务定义吗 ？ 或者，你可以对现有的系统做扩展吗 ？

1.2 设计慕课网业务数据表应该注意些什么

慕课网最有特色的当然是它的视频课程，但是怎么存储这些视频数据（内容都可以称之为数据）呢 ？ 可以直接使用 MySQL 的 blob（mediumblob、longblob）数据类型吗 ？ 答案是否定的。虽然 blob 被设计为存储二进制大对象，但是，如果真的这么去做，且数据量很大，那么，会严重降低数据库的性能。

通常，对于文件来说，我们都会把它存放在文件服务器或 CDN 中，在实际存储（MySQL 表）时只会存储文件对应的 URL。所以，视频课程的内容存储的应该是 url，而不是二进制流数据。

2. 慕课网业务表设计

根据需求的描述，我们可以把慕课网的数据表分为两类：字典表和业务表（除字典之外，都可以叫做业务）。且几乎每个业务系统的设计思想和建议都是类似（因为毕竟都是在建 MySQL 表）的，所以，这里不再赘述，只是去解释说明表设计。

首先，还是先要去创建一个独立的数据库，将来设计的所有表也都存储在这个库中。与业务名（这并不是规定，你可以认为这是个习惯）保持一致，我们把这个库叫做 `imooc`，建库语句如下：

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

2.1 字典表设计

根据业务需求，我们一共需要给慕课网系统设计5张字典表。由于字典表仅仅是 id 到描述（说明）的映射，所以，它们几乎都是一样的，我也就不再分开说明各个字典表。字典表建表语句如下：

```
-- 课程分类字典表
CREATE TABLE IF NOT EXISTS `imooc`.`i_course_type` (
  `id` tinyint(4) NOT NULL AUTO_INCREMENT COMMENT '自增主键',
  `course_type` varchar(64) NOT NULL DEFAULT '' COMMENT '课程类型: 免费课程、实战课程、专栏、手记等等',
  `remark` varchar(200) NOT NULL DEFAULT '' COMMENT '其他说明',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='课程分类字典表';

-- 课程分级字典表
CREATE TABLE IF NOT EXISTS `imooc`.`i_course_level` (
  `id` tinyint(4) NOT NULL AUTO_INCREMENT COMMENT '自增主键',
  `course_level` varchar(64) NOT NULL DEFAULT '' COMMENT '课程分级: 初级、中级、高级等等',
  `remark` varchar(200) NOT NULL DEFAULT '' COMMENT '其他说明',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='课程分级字典表';

-- 内容分类字典表
CREATE TABLE IF NOT EXISTS `imooc`.`i_content_type` (
  `id` tinyint(4) NOT NULL AUTO_INCREMENT COMMENT '自增主键',
  `content_type` varchar(64) NOT NULL DEFAULT '' COMMENT '内容类型: Java、Python、SpringCloud等等',
  `remark` varchar(200) NOT NULL DEFAULT '' COMMENT '其他说明',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='内容分类字典表';

-- 内容格式字典表
CREATE TABLE IF NOT EXISTS `imooc`.`i_content_format` (
  `id` tinyint(4) NOT NULL AUTO_INCREMENT COMMENT '自增主键',
  `content_format` varchar(64) NOT NULL DEFAULT '' COMMENT '内容格式: markdown、txt等等',
  `remark` varchar(200) NOT NULL DEFAULT '' COMMENT '其他说明',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='内容格式字典表';

-- 支付方式字典表
CREATE TABLE IF NOT EXISTS `imooc`.`i_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';
```

关于这些表，我仍有两点需要说明：

- 一定要给表和列加上合适且详尽的注释，特别是它们很相似时，例如 `i_course_type` 和 `i_content_type` 表就很相似
- `i_payment_type` 表中“缺失”了 `remark` 字段，这是因为对于支付方式不需要做特殊的说明。当然，如果你觉得仍是需要的，自行加上就好了

2.2 业务表设计

业务表相比于字典表要复杂得多，它毕竟要定义很多属性（数据列）去描述业务逻辑。为了方便理清业务，我这里将整个业务系统的相关表分为三类：用户相关信息表、课程相关信息表、用户与课程相关信息表。下面，我也就围绕这三个类别依次对其中的业务表进行介绍讲解。

2.2.1 用户相关信息表

对于一个系统来说，最重要的当然是需要有用户，那么，我们先来定义用户表。建表语句如下：

```
-- 用户表
CREATE TABLE IF NOT EXISTS `imooc`.`i_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, lecturer-讲师, 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',
  `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';
```

可以发现，这张用户表几乎与“电商”的用户表是一样的。这很好理解，无论使用哪个系统，用户的信息几乎都是一成不变的。所以，用户表大多都是相似的。当然，在“有理有据”的情况下，尽可能多的收集用户信息当然更好。

有了用户表，与 “用户相关” 的一系列表当然也都可以创建了。用户之间是可以互相关注的，在慕课网中叫做 “粉丝”，我们也就把这张表叫做“用户粉丝表”。建表语句如下：

```
-- 用户粉丝表
CREATE TABLE IF NOT EXISTS `imooc`.`i_user_follower` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '自增主键',
  `user_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '用户 id',
  `follower_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '粉丝 id',
  `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`),
  KEY `follower_id_idx` (`follower_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='用户粉丝表';
```

我们经常收到慕课网发送的推送消息，这里面涉及到推送的功能。不过，我并不打算去介绍它的功能实现逻辑。消息推送需要一张“消息表”，用于存储想要推送的消息，我们先来看一看这张表都包含什么（即建表语句）。

```
-- 推送消息表
CREATE TABLE IF NOT EXISTS `imooc`.`i_push_message` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '自增主键',
  `user_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '创建消息用户 id',
  `message` varchar(128) NOT NULL DEFAULT '' COMMENT '消息内容',
  `is_done` boolean NOT NULL DEFAULT 0 COMMENT '是否完成推送: 0-未完成, 1-已完成',
  `push_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' COMMENT '推送时间',
  `latest_push_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' COMMENT '最晚推送时间',
  `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='推送消息表';
```

is_done 字段非常重要，在插入记录时，它的值应该是 false。当完成推送之后，需要修改为 true。另外，还需要关注 push_time 和 latest_push_time，前者标识推送消息的时间；后者标识当发生某种“未知”原因，消息并未按时推送，最晚的可以推送消息的时间。即这张表里使用 latest_push_time 标识消息过期。

推送消息发出之后，会记录在一张表中，以便用户可以随时查看，这也是与用户相关的最后一张数据表。建表语句如下：

```
-- 用户消息表
CREATE TABLE IF NOT EXISTS `imooc`.`i_user_message` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '自增主键',
  `user_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '用户 id',
  `message` varchar(128) NOT NULL DEFAULT '' COMMENT '消息内容',
  `create_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00' COMMENT '创建时间',
  PRIMARY KEY (`id`),
  KEY `user_id_idx` (`user_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='用户消息表';
```

2.2.2 课程相关信息表

除了用户之外，慕课网最核心的是“课程”，课程又可以分为两个部分：课程基本信息和课程内容（想一想为什么要分成这两个部分）。我们先来创建存储课程基本信息的“课程表”：

```
-- 课程表
CREATE TABLE IF NOT EXISTS `imooc`.`i_course` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '自增主键',
  `user_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '课程所属用户',
  `course_name` varchar(128) NOT NULL DEFAULT '' COMMENT '课程名称',
  `course_level` tinyint(4) NOT NULL DEFAULT '0' COMMENT '课程分级',
  `course_type_id` tinyint(4) NOT NULL DEFAULT '0' COMMENT '课程分类',
  `content_type_str` varchar(200) NOT NULL DEFAULT '[]' COMMENT '内容分类, 支持多分类',
  `duration` int(11) NOT NULL DEFAULT '0' COMMENT '课程时长',
  `deleted` enum('normal', 'deleted') NOT NULL DEFAULT 'normal' COMMENT '状态: normal-正常, deleted-下线',
  `user_count` int(11) NOT NULL DEFAULT '0' COMMENT '学习用户数',
  `course_grade` decimal(5,2) NOT NULL COMMENT '课程评分, 最多支持两位小数',
  `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 (`id`),
  KEY `user_id_idx` (`user_id`),
  KEY `course_type_idx` (`course_type_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='课程表';
```

这张表的大部分字段都很好理解，我们需要特别关注的是 content_type_str。虽然内容分类可以用 id（数字）存储，但是我这里使用的字符类型，原因是课程可以属于很多类，例如：Java、SpringBoot、SpringCloud 等等。而 content_type_str 这个字段值存储的实际是 JSON 数组，例如：[1, 2, 3]，其中的每一个数字代表 i_content_type 表的主键。

课程的基本信息已经有了，现在需要存储课程的内容信息。建表语句如下：

```
-- 课程内容表
CREATE TABLE IF NOT EXISTS `imooc`.`i_course_content` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '自增主键',
  `user_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '课程所属用户',
  `course_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '课程 id',
  `chapter_num` int(11) NOT NULL DEFAULT '0' COMMENT '章编号',
  `section_num` int(11) NOT NULL DEFAULT '0' COMMENT '节编号',
  `content_url` varchar(1024) NOT NULL DEFAULT '' COMMENT '内容地址',
  `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 `course_id_idx` (`course_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='课程内容表';
```

可以看到，我们只会存储课程内容的 url 地址，而不会把课程的二进制内容存储下来。另外，这张表里面也冗余存储了课程的所属用户，你知道这是为什么吗？

对于每一门课程，用户都可以在学习过程中提问，为了存储这些问题，我们需要一张 “课程问题表”。建表语句如下：

```
-- 课程问题表
CREATE TABLE IF NOT EXISTS `imooc`.`i_course_problem` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '自增主键',
  `user_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '提问用户 id',
  `course_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '课程 id',
  `problem_title` varchar(100) NOT NULL DEFAULT '' COMMENT '问题标题',
  `problem_content` varchar(500) NOT NULL DEFAULT '' COMMENT '问题内容',
  `related_problem_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '关联问题的 id(问题回复)',
  `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 `course_id_idx` (`course_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='课程问题表';
```

当对问题的解答存在疑惑时，我们可以继续追问。所以，这张表里设计了 `related_problem_id` 字段，它与 `i_course_problem` 表自身进行关联。

“手记” 类似于博客，以文字的形式面向用户，我们同样使用 MySQL 表来存储其中的内容。这张表比较简单，没有太多的关联字段，建表语句如下所示：

```
-- 手记表
CREATE TABLE IF NOT EXISTS `imooc`.`i_blog` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '自增主键',
  `user_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '手记所属用户',
  `blog_title` varchar(128) NOT NULL DEFAULT '' COMMENT '手记标题',
  `blog_content` text NOT NULL COMMENT '手记内容',
  `content_format_id` tinyint(4) NOT NULL DEFAULT '0' COMMENT '内容格式',
  `content_type_str` varchar(200) NOT NULL DEFAULT '[]' COMMENT '内容分类, 支持多分类',
  `support_count` int(11) NOT NULL DEFAULT '0' COMMENT '赞个数',
  `nonsupport_count` int(11) NOT NULL DEFAULT '0' COMMENT '踩个数',
  `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`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='手记表';
```

关于这张表，我使用 text 类型（`blog_content`）去存储手记的内容。用户保存时，直接对这个字段赋值就可以；但是，如果修改手记的内容，我会怎样做呢（代码逻辑是怎样的呢）？

2.2.3 用户与课程相关信息表

用户和课程（表）都已经有了，那么，最后，我们来设计用户与课程相关的信息。首先，用户看到喜欢的课程，可以去收藏它，这就需要一张“用户收藏课程表”。建表语句如下：

```
-- 用户收藏课程表
CREATE TABLE IF NOT EXISTS `imooc`.`i_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 '收藏的课程 id',
  `course_type_id` tinyint(4) NOT NULL DEFAULT '0' COMMENT '课程分类',
  `deleted` enum('normal', 'deleted') NOT NULL DEFAULT 'normal' COMMENT '状态: normal-正常, deleted-取消收藏',
  `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='用户收藏课程表';
```

注意到，收藏表中设计了 `deleted` 字段，这里面的含义是：即使用户不再收藏该课程，也不会把收藏记录物理删除。我们可以据此猜测用户的喜好与变化情况。

与“电商”系统类似，课程也是商品，用户可以把喜欢的课程加入到购物车中，之后一并结算。所以，我们需要一张“购物车表”，建表语句如下（可以发现它与“电商购物车”是及其相似的）：

```
-- 用户课程购物车
CREATE TABLE IF NOT EXISTS `imooc`.`i_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',
  `course_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '课程 id',
  `is_purchased` enum('yes', 'no') NOT NULL DEFAULT 'no' COMMENT '状态: yes-已购买, no-未购买',
  `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 `course_id_idx` (`course_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='用户课程购物车';
```

之后，用户就可以对喜欢的课程“下单”，完成购买过程。即我们需要一张订单表，建表语句如下所示（它与“电商订单表”几乎一致，所以，这里我也不再赘述）：

```
-- 用户课程订单表
CREATE TABLE IF NOT EXISTS `imooc`.`i_course_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',
  `course_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '课程 id',
  `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',
  `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 `course_id_idx` (`course_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='用户课程订单表';
```

已经完成购买的课程，就“属于”当前用户了，这同样需要一张表来存储用户的课程数据信息。另外，需要注意，对于免费课程，我们同样会建立订单（目的是逻辑保持一致），只是它的价格为0。

```
-- 用户课程表
CREATE TABLE IF NOT EXISTS `imooc`.`i_user_course` (
  `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',
  `course_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '课程 id',
  `watch_location` varchar(512) NOT NULL DEFAULT '[]' COMMENT '观看位置记录',
  `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 `course_id_idx` (`course_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='用户课程表';
```

需要特别注意的是 `watch_location` 字段，它的存储结构同样是 JSON 数组。其中的数据类似于 [2, 7, 12:21]，代表第二章、第七节、第12分钟21秒（不过，如果你不想这样存储，把这一个字段扩展为三个也是可以的）。

用户观看了课程，可以对课程做出评价，这里，“评价表”也是用户与课程相关的最后一张业务表了。这张表同样比较简单，建表语句如下所示：

```
-- 课程评价表
CREATE TABLE IF NOT EXISTS `imooc`.`i_course_feedback` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '自增主键',
  `user_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '评论用户 id',
  `order_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '订单 id',
  `course_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '课程 id',
  `star_level` tinyint(4) NOT NULL DEFAULT '0' COMMENT '星级',
  `content` varchar(500) NOT NULL DEFAULT '' COMMENT '评价内容',
  `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='课程评价表';
```

3. 总结

给慕课网设计业务系统表是一件很有意思的事情，这是因为我们经常接触但是可能从没想过怎样去实现它。在设计的过程中，你会发现，许多表或许多字段需要经常调整。这是很正常的，毕竟谁也做不到“一次成型”。任何一个优秀的平台或系统，都需要反复雕琢，甚至是很多次试错。所以，不要去抱怨或怀疑自己的设计能力，做事情总是需要循序渐进。

4. 问题

对于当前的系统设计，你觉得合理吗？为什么呢？

用户里面分为讲师和学员，都放在一个表里合理吗？是否需要分开存储呢？

慕课网首页有个功能叫做“就业班”，对于这个功能，你会怎样设计数据表呢？

用户的积分、经验怎么存储呢？

如果让你设计慕课网的优惠券系统，你会怎样设计（数据表和逻辑都需要考虑）呢？

如果我需要一个报表系统，可以查看每天慕课网的点击和消费，应该怎么设计报表和实现呢？

5. 参考资料

《高性能 MySQL（第三版）》

慕课网

MySQL 官方文档：Data Types

MySQL 官方文档：Optimization

MySQL 官方文档：Character Sets and Collations in MySQL

}

33 为大型电商平台设计高可用数据库系统

35 总结、展望与思考