作业六：MySQL Optimization 1 & 2

519021910913 黄喆敏

请你根据上课内容，针对你在E-BookStore项目中的数据库设计，详细回答下列问题：

1. 你认为在你的数据库中应该建立什么样的索引？为什么？

   答：

   - 所有表的主键均为自增主键，mySQL会自动对主键建立索引。
   - 对于book_description表，我们可以对书籍简介的**前缀**做索引，因为我们需要根据书籍简介搜索图书，并且书籍简介太长，使用前缀索引可以减少索引所占空间。但是我们注意到，书籍简介的全文搜索是通过遍历数据库后，**由全文搜索引擎建立索引**，因此最终此处不对description建立索引。
   - 对于book表，我们对title和author分别建立索引。这是因为我们需要分别根据书的标题和书的作者搜索图书，因此建立多重索引是不合适的，会使索引失效。此外，book表中title和author字段均不太长，因此不需要用前缀索引。
   - 对于user表，我们对用户昵称做索引，因为管理员会根据用户昵称查询用户，进行用户管理。
   - 对于order表，我们建立订单时间和userId的多重索引。因为普通用户需要根据时间范围查询订单，管理员需要根据时间范围，查询所有用户的订单。为了满足这两个要求，使索引生效，我们将订单时间放在多重索引的第一列，将userId放在第二列。
   - 对于order_item表，我们对orderId建立索引，因为我们需要根据订单Id，查询每一个订单项。
   - 对于cart_item表，我们建立userId和购物车项是否生效的多重索引。因为在浏览购物车时，我们需要根据userId查询购物车项；在下单时，我们需要根据userId和是否生效，找出用户选中的购物车项。因此将userId作为多重索引的第一列，是否生效作为第二列。

2. 你的数据库中每个表中的字段类型和长度是如何确定的？为什么？

   答：

   - 对于**书籍余量等字段**，我们均用数值类型，而不用varchar类型，因为数值类型更快、所用内存更小；此外，我们**使用mediumint类型代替int**。unsigned mediumint足够存书籍余量，且所占空间比int小25%。
   - 对于**字符串类型的字段**，如果大小小于8KB，则用varchar代替blob存储。因此对于数据简介等字段，我们采用varchar存储；对于不同的varchar类型字段，我们根据实际情况定义最大长度。例如对于用户名等字段，我们定义其最大长度为31，因此使用varchar(31)即可。
   - 对于数据封面等**图片字段**，我们使用blob类型存储。
   - 对于订单时间等**时间字段**，我们采用datetime类型存储。
   - 对于用户是否禁用等**布尔类型字段**，在mySQL中，我们采用tinyint(1)类型存储。
   - 对于价格等**浮点型字段**，在我们的系统中，不会存在价格超过千万量级的订单，因此使用decimal(10,2)存储，精确到小数点后两位。
   - 表中的每个字段均不存在空的情况，因此我们都加上not null约束。这样可以加快索引效率，且不需要判断是否为null的额外标志位。
   - 对于多表查询使用到的列，例如book表中的id和order_item中的book_id，我们将它们设置为相同的类型，这样可以加快join操作的速度。

3. 你认为在我们大二上课时讲解ORM映射的Person例子时，每个用户的邮箱如果只有一个，是否还有必要像上课那样将邮箱专门存储在一张表中，然后通过外键关联？为什么？

答： 没有必要。因为分表的作用在于**减少数据冗余，同时提高查询效率**。在之前的例子中，用户和邮箱是一对多的关系，生成多条记录，因此存储到多张表，可以减少数据冗余；而现在用户和邮箱是一对一的关系，存储在一张表，不会产生数据冗余；同时，采用多表关联查询效率较慢，因此邮箱与其余信息存一张表，可以提高查询效率。

此外，结合ORM映射中所讲的Person例子，我认为也没有必要邮箱专门存储在一张表中。因为之前给的例子是通过一个event，查找所有的person对应的信息。因此可能会有向用户邮箱发信息的业务场景，即**查询用户时需要返回邮箱这个字段**。因此，不单独存在一张表更为合适。

4. 你认为主键使用自增主键和UUID各自的优缺点是什么?

   答：

   （1）自增主键优点

   - 自增主键为数字型，占用空间小，便于排序。
   - 相比于UUID，自增主键**易读性更强**，便于调试。
   - 建立索引更方便，且查询、排序效率较高。

   （2）自增主键缺点

   - 自增主键不适用于**分布式系统**。当有大量的请求需要处理（例如双十一时），不同服务器的数据库可能会生成相同的自增主键，导致主键不唯一。
   - 自增主键**业务安全性较弱**，攻击者可以通过观察主键大小、范围，来获取销量等信息。

   （3）UUID优点

   - UUID**保证全局唯一**，适用于分布式系统，同时在进行**分表分库操作**/重新建表时较为方便。
   - UUID**安全性较强**，适用于需要脱敏的数据。
   - 相比于自增主键，UUID可以**预先生成**，方便数据操作。

   （4）UUID缺点

   - UUID是一个128位的标识符，**占用空间远大于自增主键**。
   - UUID**不利于维护索引**。MySQL采用B+树建立索引，因此采用UUID会导致**索引占用空间过大**；自增主键插入时只需要维护最大节点即可，而UUID具有随机性，插入时可能需要维护多个节点，**建立索引时间较长**。
   - 根据UUID的生成策略，**读取出的数据是无序的**，不方便进行范围搜索。
   - UUID之间比较大小为字符串比较，比自增主键慢。

5. 请你搜索参考文献，总结InnoDB和MyISAM两种存储引擎的主要差异，并详细说明你的E-BookStore项目应该选择哪种存储引擎。

   答：两者差异如下：

   - **InnoDB 支持事务，MyISAM 不支持事务。**InnoDB提供事务支持事务，外部键等高级数据库功能；MyISAM强调的是性能，每次查询具有原子性，**其执行速度比 InnoDB 更快**，但是不提供事务支持。
   - **InnoDB 支持外键，而 MyISAM 不支持。**对一个包含外键的 InnoDB 表转为 MyISAM 会失败。

- InnoDB 最小的锁粒度是**行锁**，MyISAM 最小的锁粒度是**表锁**。一个更新语句会锁住整张表，导致其他查询和更新都会被阻塞，因此并发访问受限；同时需要注意，InnoDB的行锁也不是绝对的。若执行一个SQL语句时MySQL不能确定要扫描的范围，InnoDB同样会锁住全表。
- **InnoDB 是聚集索引，MyISAM 是非聚集索引。**聚集索引的文件存放在主键索引的叶子节点上，因此 InnoDB 必须要有主键，通过主键索引效率很高；而 MyISAM 是非聚集索引，数据文件是分离的，索引保存的是数据文件的指针。
- **InnoDB需要更多的内存和存储空间。**它会在内存中建立其专用的缓冲池用于高速缓冲数据和索引；**MyISAM可被压缩，存储空间较小。**它支持三种不同的存储格式：静态表、动态表、压缩表。
- InnoDB 不保存表的具体行数，执行 `select count(*) from table` 时需要全表扫描；MyISAM 用一个变量保存了整个表的行数，执行上述语句时只需要读出该变量即可，速度很快；
- InnoDB规定，如果没有设定主键或者非空唯一索引，就会**自动生成一个用户不可见的6字节的主键**，数据是主索引的一部分，附加索引保存的是主索引的值；MyISAM**允许没有任何索引和主键的表存在**，索引都是保存行的地址。
- 对于AUTO_INCREMENT类型的字段，InnoDB中必须包含只有该字段的索引，但是在MyISAM表中，可以和其他字段一起建立联合索引。

我认为我的项目应该采用**InnoDB存储引擎**，理由如下：

- 书店下单时，采用了事务机制进行控制。MyISAM的一个缺陷是不支持事务处理，而InnoDB支持事务。因此采用InnoDB。
- 数据库设计时，例如order、order_item分成了两张表，并通过外键连接。MyISAM不支持外键，因此必须采用InnoDB。
- MyISAM更适合读多写少的场景，而书店下订单会有大量的写操作，因此采用InnoDB性能更好。
- MyISAM的一个缺陷是**崩溃后无法安全恢复**。而书店中存在用户订单，金额等重要信息，若数据库奔溃后无法恢复，则会带来不可预计的后果。因此采用InnoDB，**可以利用事务日志进行数据恢复**，更加方便。

数据库设计方案如下：

```
drop schema if exists bookstore;
create schema bookstore;
use bookstore;

/* Table structure for book_image */
create table `book_image`
(
    `id`    int unsigned auto_increment not null,
    `image` blob                        not null,
    primary key (`id`)
) engine = InnoDB
  charset = utf8;

/* Table structure for book_description */
create table `book_description`
(
```

```
    `id`          int unsigned auto_increment not null,
    `description` varchar(1000)               not null,
    primary key (`id`)
) engine = InnoDB
  charset = utf8;

/* Table structure for book */
drop table if exists `book`;
create table `book`
(
    `id`          int unsigned auto_increment not null,
    `title`       varchar(63)                 not null,
    `author`      varchar(63)                 not null,
    `price`       decimal(10, 2)              not null,
    `isbn`        varchar(15)                 not null,
    `inventory`   mediumint unsigned          not null,
    `image_id`    int unsigned                not null,
    `descript_id` int unsigned                not null,
    `type`        varchar(15)                 not null, # 历史类/文学类...
    `enabled`     tinyint(1) unsigned         not null default 1,
    index (title),
    index (author),
    primary key (`id`),
    foreign key (`image_id`) references book_image (`id`),
    foreign key (`descript_id`) references book_description (`id`)
) engine = InnoDB
  charset = utf8;

/* Table structure for user_auth */
drop table if exists `user_auth`;
create table `user_auth`
(
    `id`       int unsigned auto_increment not null,
    `username` varchar(31)                 not null,
    `password` varchar(31)                 not null,
    `type`     tinyint unsigned            not null default 0, /* 0: user  1: admin */
    primary key (`id`)
) engine = InnoDB
  charset = utf8;

/* Table structure for user */
drop table if exists `user`;
create table `user`
(
    `id`       int unsigned auto_increment   not null,
    `nickname` varchar(31)                   not null,
    `email`    varchar(63)                   not null,
    `enabled`  tinyint(1) unsigned default 1 not null, /* true or false */
    `auth_id`  int unsigned                  not null,
```

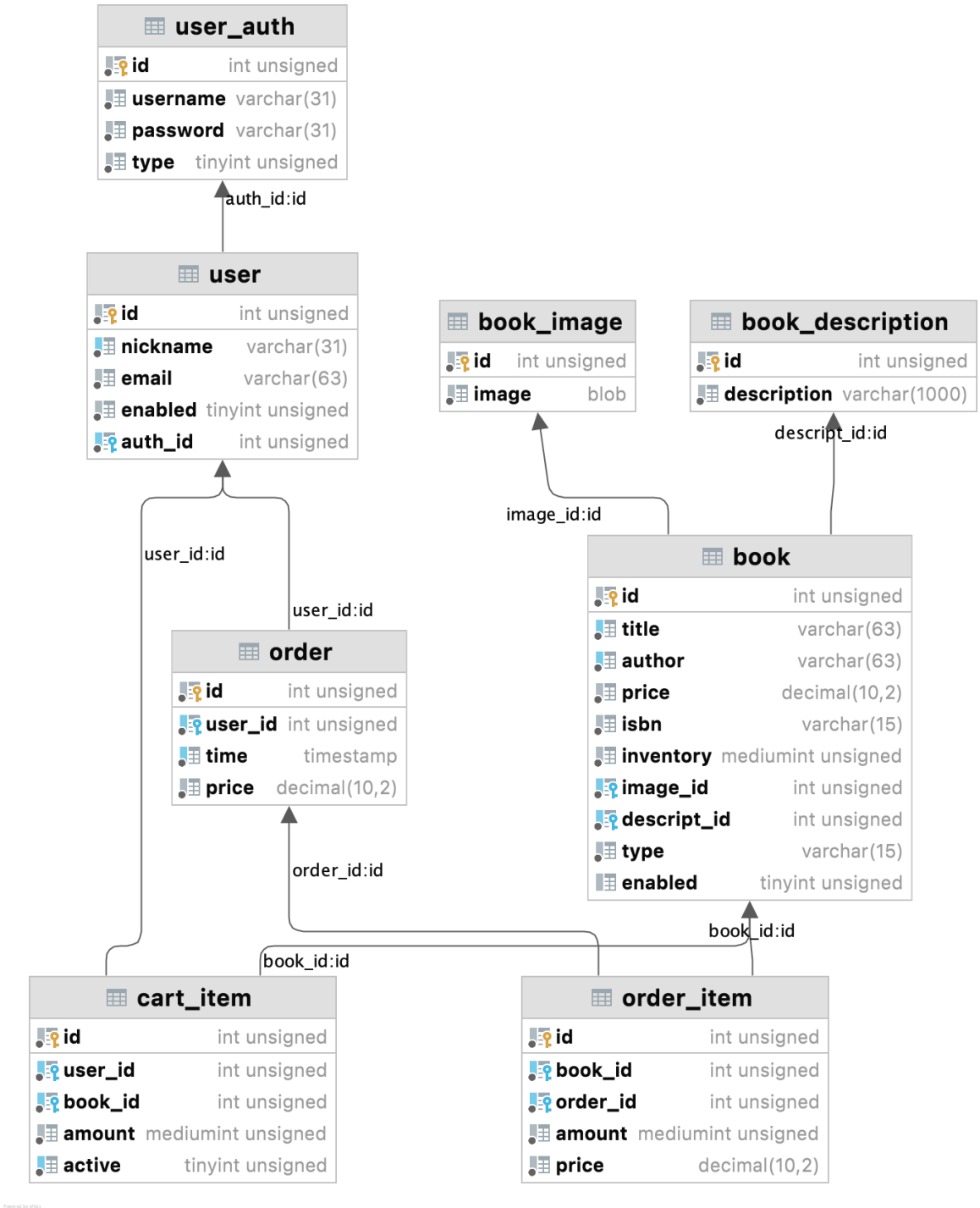
```
    index (`nickname`),
    primary key (`id`),
    foreign key (`auth_id`) references user_auth (`id`)
) engine = InnoDB
  charset = utf8;

/* Table structure for order */
drop table if exists `order`;
create table `order`
(
    `id`      int unsigned auto_increment not null,
    `user_id` int unsigned                not null,
    `time`    datetime                    not null,
    `price`   decimal(10, 2)              not null,
    index (time, user_id),
    primary key (`id`),
    foreign key (`user_id`) references `user` (`id`)
) engine = InnoDB
  charset = utf8;

/* Table structure for order_item */
drop table if exists `order_item`;
create table `order_item`
(
    `id`       int unsigned auto_increment not null,
    `book_id`  int unsigned                not null,
    `order_id` int unsigned                not null,
    `amount`   mediumint unsigned          not null,
    `price`    decimal(10, 2)              not null,
    index (`order_id`),
    primary key (`id`),
    foreign key (`order_id`) references `order` (`id`),
    foreign key (`book_id`) references `book` (`id`)
) ENGINE = InnoDB
  CHARSET = utf8;

/* Table structure for cart_item */
drop table if exists `cart_item`;
create table `cart_item`
(
    `id`      int unsigned auto_increment   not null,
    `user_id` int unsigned                  not null,
    `book_id` int unsigned                  not null,
    `amount`  mediumint unsigned            not null,
    `active`  tinyint(1) unsigned default 1 not null,
    index (`user_id`, `active`),
    PRIMARY KEY (`id`),
    foreign key (`user_id`) references `user` (`id`),
    foreign key (`book_id`) references `book` (`id`)
```

```
) engine = InnoDB
  charset = utf8;
```



Reference:

https://www.zhihu.com/question/20596402

https://www.runoob.com/w3cnote/mysql-different-nnodb-myisam.html

https://zhuanlan.zhihu.com/p/95949946