

# hw6:MySQL Optimization 1 & 2

---

## hw6:MySQL Optimization 1 & 2

Q1

Q2

Q3

Q4

Q5

### 数据库更新方案

book

cart

order

user

userAuth

表之间的关系

### reference

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？
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？
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？
4. 你认为主键使用自增主键和UUID各自的优缺点是什么？
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。

-请提交包含上述问题答案的文档，文档中附上你更新的数据库的设计方案，包括库结构、表结构和表与表之间的关联

评分标准：

-上述每个问题 1 分，答案不唯一，只要你的说理合理即可视为正确。

## Q1

---

下划线部分为我针对原则做的数据库改进。

### 1. 主键适合作为索引的场景及其原因：

1. 一般按照经常查找的列来建立索引。虽然主键（一般是ID）是用户不知道的信息，但是如果有 其他的表关联到这张表，用它的主键关联。用主键做索引就很需要了。因此如果有这样的情况就使用主键作为索引，比如OrderItem表关联Order 表的主键，因此Order表应该使用主键作为索引。
2. innodb 在硬盘中也是按照主键的顺序存储的，这样有利于顺序读写。
3. 如果不知道其他字段是否唯一，不如使用递增或者UUID生成的保证唯一性的主键。

4. 主键的设置方式一般有自增或者UUID。自增算法简单，并且可以得知数据的插入顺序。而UUID一般运用在分布式中。如果用服务器集群处理大量请求时，基于整数递增可能导致重复，因为插入的是两张不同机器上的表，但是UUID用机器IP + timestamp + 进程PID+当前对象ID+ 随机数，凑成一个32位的16进制数（16byte）几乎不会重复。

- uuid 缺陷：uuid太大了，索引比较耗费空间。不可能知道插入顺序，如果需要知道顺序可以在前面加入前缀。
- 平衡：是需要性能(自增)，还是需要分布式不能出错（UUID），看需求进行选择。

2. 如果有多个条件查找可以考虑建立复合索引。如果是复合索引：不可以有太多列，索引里面包含NULL的值越少越好。
3. 数据少的时候就没有必要建立索引。对于较小的表，建立索引的意义不大。但是如果是太大的表，如果每个元素的都被使用，顺序读写的效率反而比索引高。
4. 索引大小不能太大，比如用book的introduction做索引就不合适，可以选前面10个字符。因为这样会导致索引占用内存太多，会涉及大量页的换进换出导致性能降低。
5. 索引最好每个索引都定位到唯一的元素，这样使用主键做索引就比较好。
6. 拆开表

## Q2

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？

1. 能用数字存储的尽量不用字符串存储，因为这样能存储的数据量更大。
2. 每个数据尽量都使用NOT NULL。因为NULL会比NOT NULL多使用一个字节表示该字段是否空。
3. 关于字符串的类型选择：

一般在保存少量字符串的时候，我们会选择CHAR或者VARCHAR，

- char长度固定，即每条数据占用等长字节空间，适合用在身份证号码、手机号码等定。超过255字节的只能用varchar或者text。在我的实现中，书籍的ISBN是固定格式的，使用char格式。
- varchar可变长度，可以设置最大长度；适合用在长度可变的属性。字符串尽量使用VARCHAR，参数根据最大长度确定，因为VARCHAR可以动态存储，可以压缩空间。CHAR就没有动态存储，每个数据都会占用固定长度的空间。我的实现中，不确定长度的字符串使用varchar

保存较大文本时(>8KB)，通常会选择使用TEXT或者BLOB

- blob。blob和text二者之间的主要差别是BLOB能用来保存二进制数据，比如照片；而TEXT只能保存字符数据，比如一遍文章或日记
- text不设置长度，当不知道属性的最大长度时，适合用text，能用varchar的地方不用text。当text列的内容很多的时候，text列的内容会保留一个指针在记录中，这个指针指向了磁盘中的一块区域，当对这个表进行select \*的时候，会从磁盘中读取text的值，影响查询的性能，而varchar不会存在这个问题。在我的实现中，由于book表中的introduciton是字符不确定长度，并且很长，因此选用了text类型。

## Q3

根据具体需求而定。需要考虑：使用频率、文本长度、一个用户的邮箱个数

1. 一般情况下，邮箱不是很长的文本，而且很多时候作为用户的登陆id，这个时候就使用频率较高，如果是一个用户单一邮箱，就直接存在用户信息的表里，不需要外键关联比较好。因为频繁使用Join操作会带来很大的开销。同时增加一个占用空间比较小的字段，在从磁盘读到内存的时候也不会带来太多占用内存的情况。

2. 特殊情况下，如果使用频率不大，或者极端情况下邮箱的格式特殊，都很长很长（虽然觉得这种情况很少见但是也要考虑），或者一个用户有好几个邮箱，这样如果存在另外一张表的join操作带来的代价比放入内存的性能代价要小的话，放在另外一张表更合适

## Q4

---

1. 主键的设置方式一般有自增或者UUID。
  - 自增优势：
    - 算法简单，并且可以得知数据的插入顺序。
    - 在进行数据库插入时，位置相对固定（B+树中的右下角）增加数据插入效率，减少插入的磁盘IO消耗，每页的空间在填满的情况下再去申请下一个空间，底层物理连续性更好，能更好的支持区间查找
  - 自增主键 缺点：
    1. 安全性不高，容易被得知业务量和数据量等，而且容易被爬取数据。
    2. 高并发时，竞争自增锁会降低数据库的吞吐能力。
    3. 数据迁移时，尤其是发生表格合并时，多个表容易主键冲突。
  - UUID优势：
    1. UUID一般运用在分布式中。如果用服务器集群处理大量请求时，基于整数递增可能导致重复，因为插入的是两张不同机器上的表，但是UUID用机器IP + timestamp + 进程PID+当前对象ID+ 随机数，凑成一个32位的16进制数（16byte）几乎不会重复。使用UUID，生成的ID不仅是表独立的，而且是库独立的。对以后的数据操作很有好处，可以说一劳永逸。
    2. 相对安全，不能简单的从uuid获取信息，但是如果自增，则容易暴露信息。
  - uuid 缺陷：
    - uuid太大了，索引比较耗费空间。不可能知道插入顺序，如果需要知道顺序可以在前面加入前缀。
    - 由于UUID是随机生成的 插入时位置具有一定的不确定性，无序插入，会存在许多内存碎片，造成硬盘使用率低
    - 降低查询速度。
- 平衡：是需要性能(自增)，还是需要分布式不能出错（UUID），看需求进行选择。

## Q5

---

一、InnoDB支持事务，MyISAM不支持，这一点是非常之重要。事务是一种高级的处理方式，如在一些列增删改中只要哪个出错还可以回滚还原，而MyISAM就不可以了。

二、MyISAM适合查询以及插入为主的应用，InnoDB适合频繁修改以及涉及到安全性较高的应用

三、InnoDB支持**外键**，MyISAM不支持

四、MyISAM是默认引擎，InnoDB需要指定

五、InnoDB不支持FULLTEXT类型的索引

六、InnoDB中不保存表的行数，如select count() from table时，InnoDB需要扫描一遍整个表来计算有多少行，但是MyISAM只要简单的读出保存好的行数即可。注意的是，当count()语句包含where条件时MyISAM也需要扫描整个表

七、对于自增长的字段，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

八、清空整个表时，InnoDB是一行一行的删除，效率非常慢。MyISAM则会重建表

九、InnoDB支持行锁（某些情况下还是锁整表，如 **update table set a=1 where user like '%lee%'**

## MySQL中InnoDB和MyISAM的比较

### MyISAM:

每个MyISAM在磁盘上存储成三个文件。第一个文件的名称以表的名称开始，扩展名指出文件类型。.frm文件存储表定义。数据文件的扩展名为.MYD (MYData)。

MyISAM表格可以被压缩，而且它们支持全文搜索。不支持事务，而且也不支持外键。如果事物回滚将造成不完全回滚，不具有原子性。在进行update时进行表锁，并发量相对较小。如果执行大量的SELECT，MyISAM是更好的选择。

MyISAM的索引和数据是分开的，并且索引是有压缩的，内存使用率就对应提高了不少。能加载更多索引，而InnoDB是索引和数据是紧密捆绑的，没有使用压缩从而会造成InnoDB比MyISAM体积庞大不小

MyISAM缓存在内存的是索引，不是数据。而InnoDB缓存在内存的是数据，相对来说，服务器内存越大，InnoDB发挥的优势越大。

**优点：**查询数据相对较快，适合大量的select，可以全文索引。

**缺点：**不支持事务，不支持外键，并发量较小，不适合大量update

### InnoDB:

这种类型是事务安全的。它与BDB类型具有相同的特性,它们还支持外键。InnoDB表格速度很快。具有比BDB还丰富的特性,因此如果需要一个事务安全的存储引擎，建议使用它。在update时表进行行锁，并发量相对较大。如果你的数据执行大量的INSERT或UPDATE，出于性能方面的考虑，应该使用InnoDB表。

**优点：**支持事务，支持外键，并发量较大，适合大量update

**缺点：**查询数据相对较快，不适合大量的select

对于支持事物的InnoDB类型的表，影响速度的主要原因是AUTOCOMMIT默认设置是打开的，而且程序没有显式调用BEGIN 开始事务，导致每插入一条都自动Commit，严重影响了速度。可以在执行sql前调用begin，多条sql形成一个事物（即使autocommit打开也可以），将大大提高性能。

**基本的差别为：**MyISAM类型不支持事务处理等高级处理，而InnoDB类型支持。

MyISAM类型的表强调的是性能，其执行速度比InnoDB类型更快，但是不提供事务支持，而InnoDB提供事务支持已经外部键等高级数据库功能。

### 其他比较:

MyIASM是IASM表的新版本，有如下扩展:

- 二进制层次的可移植性。
- NULL列索引。
- 对变长行比ISAM表有更少的碎片。
- 支持大文件。
- 更好的索引压缩。

- 更好的键吗统计分布。
- 更好和更快的auto\_increment处理。

1 • show engines;  
2  
3










100%	15:1					
Result Grid		Filter Rows:	Search	Export:		
Engine	Support	Comment	Transactions	XA	Savepo	
ARCHIVE	YES	Archive storage engine	NO	NO	NO	
BLACKHOLE	YES	/dev/null storage engine (anything you write to it...	NO	NO	NO	
MRG_MYISAM	YES	Collection of identical MyISAM tables	NO	NO	NO	
FEDERATED	NO	Federated MySQL storage engine	NULL	NULL	NULL	
MyISAM	YES	MyISAM storage engine	NO	NO	NO	
PERFORMANCE_SCHEMA	YES	Performance Schema	NO	NO	NO	
InnoDB	DEFAULT	Supports transactions, row-level locking, and for...	YES	YES	YES	
MEMORY	YES	Hash based, stored in memory, useful for tempo...	NO	NO	NO	
CSV	YES	CSV storage engine	NO	NO	NO	

在我的实现中，因为需要实现外键、事务。所以选择innodb

## 数据库更新方案















表包括：book, cart,orders, order\_items, user, user\_auth

### book

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G
 book_id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
 author	VARCHAR(50)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 description	VARCHAR(255)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 image	VARCHAR(255)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 inventory	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 isbn	CHAR(13)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 name	VARCHAR(50)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 price	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 type	VARCHAR(15)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>









- `book_id` 是主键设成自增，建立唯一性索引，便于其他表格外键引用，能够提升表格查询性能。因为目前还没有做分布式的系统，不容易有高并发或与其他数据库合并等操作，用自增的方式性能比较好。
- `author` 是作者名，因为名字可变长并且可能多个，故采用 `varchar(50)`。满足了绝大部分书籍作者名称长度要求且节约了存储空间。
- `description` 是书籍的简介。本数据库中，大部分的简介在100字左右。于是为了扩展起见，使用 `varchar(255)`，虽然是比较大的数据，但是上课提到基于性能考虑："不超过 8 KB 的数据不用 BLOB/TEXT 用 VARCHAR"
- `image` 是书籍的图片路径，使用了图床，平均的长度大概在160个字符，扩展起见以及同introduction一样的考虑，使用VARCHAR
- `inventory` 是书籍的库存。由于小型书籍网站的书籍库存不会太大，因此使用 `int` 来存储，节约磁盘空间。
- `isbn` 是书籍的 isbn 编号，经过查阅资料，目前 isbn 的长度均为 13 位，因此用固定长度CHAR的字符串存储。
- `name` 是书籍名称。因为名字可变长并且可能多个，故采用 `varchar(50)`。满足了绝大部分书籍作者名称长度要求且节约了存储空间。
- `price` 是书籍的价格，用 `int` 存储而不是 `float` 或 `double` 存储，避免了加减计算时的精度丢失问题，且满足了小型书籍网站的需求。
- `type` 是书籍种类对应的种类，考虑到种类之后可能有很多，并且可能有大分类下面的小分类，用数字存可能不能表达意思，于是采用字符串。
- 关于分表问题：虽然在首页不需要拿到description 和image这两个稍微大一些的数据。但是由于book是很多类里面的oneToMany的对象，比如orderItem, 这样就使得使用这些对象的时候要加载book,再加载一张分开的表，有很多的join操作，极大的降低了性能。因为这两个数据并没有那么大，比起join产生的数据下降，不分表是合理的。

cart

Column	Datatype		PK	NN	UQ	BIN	UN	ZF	AI	G
 cart_id	INT		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
 author	VARCHAR(50)		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 name	VARCHAR(50)		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 number	INT		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 price	INT		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 book_id	INT		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 user_id	INT		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

- cart\_id 是主键设成自增，建立唯一性索，便于其他表外键引用，能够提升表格查询性能。因为目前还没有做分布式的系统，不容易有高并发与其他数据库合并等操作，用自增的方式性能比较好。
- 在这里对author， name, price 做了冗余处理，因为cart是比较经常访问的，降低范式化的程度可以使得速度更快。
- 这里有外键关联字段：book\_id, user\_id, 外键关联字段要求两者的名字和类型完全一致。





order

Column	Datatype		PK	NN	UQ	BIN	UN	ZF	AI	G
 order_id	INT		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
 order_price	INT		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 user_id	INT		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 date	CHAR(10)		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

date：下单日期，这里的格式是2019-10-01，共10个字符，所以用固定字符的。

其他同上

user





Column	Datatype		PK	NN	UQ	BIN	UN	ZF	AI	G
 user_id	INT	↕	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
 name	VARCHAR(50)	↕	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 type	INT	↕	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 email	VARCHAR(50)	↕	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

name: 用户名,在注册的时候有限制字符少于50个并且可变长, 选择VARCHAR (50)

type: 用户类型, 目前有管理员, 普通用户, 被禁止使用用户。考虑到之后还有别的类型拓展, 所以使用int来记录。

email: 用户邮箱, 平均长度一般少于50, 并且可变长, 选择VARCHAR (50)

## userAuth

Column	Datatype		PK	NN	UQ	BIN	UN	ZF	AI	G
 user_id	INT	↕	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 type	INT	↕	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 name	VARCHAR(50)	↕	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 password	VARCHAR(50)	↕	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

这张表用来校验用户的信息, 因为password只有在登陆的时候使用, 但是user被其他调用需要他的信息的时候一般不需要password,同时userAuth是做外键关联而不是用oneToMany 的方式作为一个必须加载的成员的, 所以就可以分开两张表。

type: user 类型

password: 注册时设置了长度上限, 考虑到正常密码的长度, 如果有那种浏览器自动生成的强密码可能很长, 设置了50。

## 表之间的关系

orderItem 和order由order\_id外键关联。

userAuth 和user由user\_id外键guan'lian

## reference

MySQL中的Text类型

<https://blog.csdn.net/SlowIsFastLemon/article/details/106383776>

Mysql主键UUID和自增主键区别优劣



[https://blog.csdn.net/qg\\_39454048/article/details/82378155?  
spm=1001.2101.3001.6650.3&utm\\_medium=distribute.pc\\_relevant.none-task-blog-2\\_defaultCTRLIST\\_default-  
3.essearch\\_pc\\_relevant&depth\\_1-utm\\_source=distribute.pc\\_relevant.none-task-blog-2\\_defaultCTRLIST\\_default-  
3.essearch\\_pc\\_relevant](https://blog.csdn.net/qg_39454048/article/details/82378155?spm=1001.2101.3001.6650.3&utm_medium=distribute.pc_relevant.none-task-blog-2_defaultCTRLIST_default-3.essearch_pc_relevant&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2_defaultCTRLIST_default-3.essearch_pc_relevant)

MySQL存储引擎InnoDB与Myisam的六大区别

<https://www.runoob.com/w3cnote/mysql-different-innodb-mysam.html>