HW9

A

聚簇索引的数据在磁盘上是连续存储的，因此在进行范围查找时是顺序读，执行效率高。

B

我选择以 LONGBLOB 类型来存储，因为我的图书封面的图片转为 BASE64 之后，大小超过了 8K，也就是超过了 VARCHAR 的存储上限，而 LONGBLOB 支持对较大文本的存储，因此选用 LONGBLOB 的类型来存储。

另外一方面由于不需要在图片上进行索引，因此 VARCHAR 不是必要的。

C

```SQL
CREATE INDEX BookIndex
ON books (book_name, price)
```

我建立了一个基于书名和价格的复合索引，首先因为这两个列的数据不可能为空，其次他们是在查找时的高频率查找条件。用户可以基于该索引进行基于书名的单字段查询，也可以进行书名和价格区间的多字段查询，通过复合索引的方式加速了查找的速度。

字段顺序：将 book_name 放在前面是由于 book_name 是最常被查询的字段，放在前面可以使得更多查询支持索引查找，更大程度地提高查找性能。

字段的升降序：采用了默认的 ASC（升序），这是由于我在呈现书本列表时优先将书本按价格从低到高排序。

D

选用自增主键好，在我的项目中仅存在单服务节点，不存在多节点时基于整数递增可能导致的 ID 重复，且可使用自增主键可以根据主键的 ID 来判断订单的产生顺序。而 UUID 太大，索引较花费时间，尤其是我的业务需要将订单表和其他表格进行 Join 操作，此时性能较低。

E

1. InnoDB 支持事务，MyISAM 不支持
2. 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
3. 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
4. InnoDB 支持**外键**，MyISAM 不支持
5. InnoDB 不支持 FULLTEXT 类型的索引
6. MyISAM 是默认引擎，InnoDB 需要指定
7. 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。
8. 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
9. 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%'**）