具体表的结构设计

用户/商店表

用户表

FIELD	User_id	Password	Balance	Token	Terminal	Address
类型	String	String	Integer	String	String	String
	主键；唯一键	非空	非空	非空	非空	

调整了原有用户表的结构，为收货发货做准备。当一个用户有多个收货地址时，可以考虑新增`user-address`表，满足一对多的关系。

用户-商店关系表

FIELD	User_id	store_id
类型	String	String
	外键	唯一，非空

在数据库设计之初，我们在用户表中新增一列`username`，但是为了后面与test测试接口统一，将其删去。`username`和`user_id`的两个字段更符合真实场景中的设计逻辑，`user_id`可以是用户注册是的邮箱，手机号等，而`username`才是用户昵称等用户名。`store_id`也同理；

商店表(商店-图书）关系表

FIELD	Store_id	Book_id	Stock_level	Price
类型	String	String	Integer	Integer
	外键	外键	非空	非空

订单总表

- 我们没有选择使用status_code：0，1，2，去将待发货订单，已发货待收货订单，已收获订单进行区分，而是新增了purchase_time和三张不同的订单类型的表，更好的支持用户查看交易时间，发货时间和收获时间。
- 逻辑上，订单表中的price与(商店-图书）关系表中的price字段的含义一致，都是具有时间变化性的价格，即不同的商家针对不同的书目有着自己的定价和折扣，更好的模拟真实的交易过程(如建议零售价，原价和购物券的叠加等)

待付款订单

当用户下单之后，订单会按照这个格式插入该表中。order_id为主码，区别不同的订单，buyer_id,store_id,price是为了方面查找订单数据。commit_time是为了记录提交订单时间和用于判断订单是否超时使用。

FIELD	Order_id	buyer_id	Store_id	Price	commit_time
类型	String	String	String	Integer	DateTime
	主键	外键	外键		订单提交时间

待发货订单

当用户付款之后，订单会按照这个格式插入该表中。order_id为主码，区别不同的订单，buyer_id,store_id,price是为了方面查找订单数据。purchase_time是为了记录付款订单时间。

FIELD	Order_id	buyer_id	Store_id	Price	Purchase_time
类型	String	String	String	Integer	DateTime
	主键	外键	外键		订单付款时间

待收货订单

当卖家发货之后后，订单会按照这个格式插入该表中。order_id为主码，区别不同的订单，buyer_id,store_id,price是为了方面查找订单数据。send_time是为卖家发货时间，receive_time是记录买家的收货时间。

FIELD	Order_id	buyer_id	Store_id	Price	send_time	Receive_time
类型	String	String	String	Integer	DateTime	DateTime
	主键	外键	外键		订单发货时间	订单收货时间

这里新增一个收货时间字段。这样加的目的由于我们功能中不涉及对收货订单的评价，不需要另外增加待评价订单。通过receive_time区分已发货订单和未发货订单，并在查询用户订单功能中支持查询已收货和已发货未收货订单。

已取消订单

new_order_detail包括用户的所有订单，也应包含new_order_cancel。new_order_cancel与new_order_undelivered等的并集为new_order。这样的添加使得订单的结构更加完善。对应需要修改手动删除，自动删除以及查询订单历史的函数。手动删除订单和自动删除后将对应订单插入该表。

FIELD	Order_id	buyer_id	Store_id	Price	CANCEL_time
类型	String	String	String	Integer	DateTime
	主键	外键	外键		

订单明细表

对于订单类型区分有订单总表和订单明细表的原因：

1.0版本

FIELD	order_id	Book_id	Count	Price
类型	String	String	Integre	Integer
	主键	外键		

针对查询优化的2.0版本--应用驱动优化

由于在查询用户历史订单的功能应用中遇到之前的表结构查询效率较慢的问题，需要查询一个buyer的所有订单时，要在usr_store里查user买过的store的store_id, 再在store查买过的book_id 再在new_order_detail查book_id对应的具体信息。

故在new_order_detail中添加冗余属性store_id和buyer_id。

这样的修改方便用户查询所有订单信息，也方便卖家查自己店铺的所有订单信息。

FIELD	order_id	Book_id	user_id	store_id	Count	Price
类型	String	String	String	String	Integer	Integer
	主键	外键				

Book表(与book.db的schema基本保持一致)

- 在这里，书的概念要做详细说明，book_id不是表示的每一本书，而是每一类同名的书，如A店的《当代数据库管理系统》的book_id是1，B店的《当代数据库管理系统》这本书的book_id也是1，他们的book_id相同，但是不同店，甚至是同一个店不同的交易定价，这些书可以拥有属于自己的不同的价格。比如11.11那天0点A店的《当代数据库管理系统》这本书可以是30元，11.11日0点B店为29.99元。
- 在后续的进一步完善中，还可以使A店的《当代数据库管理系统》拥有不同的价格，在这里暂未实现
- 字段的text类型是为了全文索引功能的支持。

FIELD	book_id	Title	Author	Publisher	Original_title	translator	Pub_year	Pages	Original_price	Binding	Isbn	Author_intro	book_int
类型	Integer	Text	Text	Text	Text	Text	Text	Text	Integer	Text	Text	Text	Text
	主键，自增	非空											