报告

使用关系式数据库

1. **表格结构：**
   - **store 表：**
     - 字段 `book_info` 类型更改为 `text`，支持更大的文本内容。
     - 索引添加在 `store_id` 和 `book_id` 字段上，以优化查询效率。
     - `stock_level` 字段的数据类型为 `int`，以更好地表示库存水平。
   - **user 表：**
     - `token` 和 `terminal` 字段的类型更改为 `varchar(255)`，以支持更长的字符串。
     - 添加索引在 `user_id` 字段上，以优化用户查询和验证登录信息的速度。
   - **user_store 表：**
     - 添加索引在 `user_id` 和 `store_id` 字段上，以提高用户与商店关联信息的查询效率。
   - **new_order 表：**
     - 添加索引在 `order_id` 字段上，以支持订单的快速查询。
     - `user_id` 和 `store_id` 字段设置为可选，以适应可能的情况。
   - **new_order_detail 表：**
     - 添加索引在 `order_id`、`book_id` 字段上，以提高订单细节的查询效率。
2. **数据类型调整：**
   - 对于 `int` 类型的字段，例如 `balance`、`stock_level`、`count` 和 `price`，使用适当的数据类型以提高存储效率和查询速度。
3. **主键设置：**
   - 主键设置在每个表格中的唯一标识符字段上，确保数据唯一性和快速检索。

4. **索引优化：**
    - 添加合适的索引以优化查询性能，减少查询时间。
5. **表之间关联：**
    - 表之间建立关联，如 `user_store` 表中的 `user_id` 和 `store_id`，以支持更复杂的查询和数据关联。

改动理由：

1. **提高访问速度：**
    - 使用关系型数据库的索引机制，优化表格结构和数据类型，以提高数据库查询速度，特别是对于经常进行的查询操作。
2. **便于编写业务逻辑代码：**
    - 关系型数据库提供了更直观的表格结构和查询语言，使得编写业务逻辑代码更为方便和直观。表之间的关联使得数据之间的关系更加清晰。
3. **支持事务处理：**
    - 关系型数据库天然支持事务处理，确保了数据的一致性和可靠性。这对于处理订单、库存等涉及多个表格的业务非常重要。

后端逻辑代码改动

在本次项目中，对后端逻辑代码进行了改动，以适应关系型数据库（MySQL）的使用。对 `be/model` 目录下buyer.py，db_conn.py，seller.py，store.py，user.py相关文件进行了修改，编写了MySQL 相关的代码，使用 `cursor.execute( )` 代码进行sql语句的执行，使用 `self.conn.commit()` 代码进行事务的提交。在 `be/server.py` 中加入了连接本地mysql数据库的代码。

---

book_lx.db导入本地mysql

如果选用的是mysql的话，直接导入会因为content和picture的长度超过限制而无法导入，因此只导入了除了这两列数据之外其他列，在book_lx.db中建立一个新的table，但是只保留除了content和picture的其他列，再在mysql中建立对应的book表格和列，再将数据一一导入

```
import sqlite3
import mysql.connector

original_db_path = r'D:\desktop\lesson\bookstore\fe\data\book_lx.db'

connection = sqlite3.connect(original_db_path)
cursor = connection.cursor()

cursor.execute(
    'CREATE TABLE new_table AS SELECT id, title, author, publisher, original_title,
translator, pub_year, pages, price, currency_unit, binding, isbn, author_intro,
book_intro, tags FROM book;')

connection.commit()

connection.close()

sqlite_file = r'D:\desktop\lesson\bookstore\fe\data\book_lx.db'

mysql_config = {
    'host': 'localhost',
    'user': 'root',
    'password': '',
    'database': 'book_test',
    'charset': 'utf8mb4',  # Explicitly set character set to UTF-8
    'collation': 'utf8mb4_unicode_ci',  # Set collation
    'raise_on_warnings': True
}

sqlite_conn = sqlite3.connect(sqlite_file)
sqlite_cursor = sqlite_conn.cursor()

mysql_conn = mysql.connector.connect(**mysql_config)
mysql_cursor = mysql_conn.cursor()

sqlite_cursor.execute("SELECT * FROM new_table")
sqlite_data = sqlite_cursor.fetchall()

for row in sqlite_data:
    mysql_cursor.execute("""
        INSERT INTO book (id, title, author, publisher, original_title, translator,
pub_year, pages,
                          price, currency_unit, binding, isbn, author_intro,
book_intro, tags)
        VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)
    """, row)

mysql_conn.commit()
mysql_conn.close()
sqlite_conn.close()
```

```
print("Data successfully transferred from SQLite to MySQL.")
```

测试

在bookstore下的window终端运行以下命令进行测试

```
coverage run --timid --branch --source fe,be --concurrency=thread -m pytest -v --
ignore=fe/data
coverage combine
coverage report
coverage html
```

最终的测试覆盖率为45%

```
Name                                Stmts   Miss Branch BrPart  Cover
-------------------------------------------------------------------
be\__init__.py                          0      0      0      0   100%
be\app.py                               3      3      2      0     0%
be\model\__init__.py                    0      0      0      0   100%
be\model\buyer.py                     136    125     48      0     7%
be\model\db_conn.py                    28     21      8      0    25%
be\model\error.py                      23      9      0      0    61%
be\model\seller.py                     49     41     24      0    14%
be\model\store.py                      27      0      2      0   100%
be\model\user.py                      153     56     60      6    59%
be\serve.py                            33      0      0      0   100%
be\view\__init__.py                     0      0      0      0   100%
be\view\auth.py                        37      6      0      0    84%
be\view\buyer.py                       31     23      2      0    24%
be\view\seller.py                      28     19      0      0    32%
fe\__init__.py                          0      0      0      0   100%
fe\access\__init__.py                   0      0      0      0   100%
fe\access\auth.py                      35      5      6      1    85%
fe\access\book.py                      63     27     12      1    55%
fe\access\buyer.py                     36     27      4      0    28%
fe\access\new_buyer.py                  8      2      0      0    75%
fe\access\new_seller.py                 8      2      0      0    75%
fe\access\seller.py                    31     22      2      0    33%
fe\bench\__init__.py                    0      0      0      0   100%
fe\bench\run.py                        13      8      6      0    26%
fe\bench\session.py                    47     37     14      0    20%
fe\bench\workload.py                  125     70     28      2    41%
fe\conf.py                             11      0      0      0   100%
fe\conftest.py                         19      0      0      0   100%
fe\test\gen_book_data.py               49     37     18      0    21%
fe\test\test_add_book.py               36     21     12      0    35%
fe\test\test_add_funds.py              22     11      2      0    54%
fe\test\test_add_stock_level.py        39     24     12      0    33%
fe\test\test_bench.py                   6      0      0      0   100%
fe\test\test_create_store.py           19      6      2      0    71%
fe\test\test_login.py                  27      0      2      0   100%
fe\test\test_new_order.py              39     23      2      0    44%
fe\test\test_password.py               32     17      2      0    50%
fe\test\test_payment.py                59     34      6      0    42%
fe\test\test_register.py               30      0      2      0   100%
-------------------------------------------------------------------
TOTAL                                1302    676    278     10    45%
```