分布式数据库系统设计
项目技术报告

组长：周孟莹（19110240001）

组员：方睿钰 （19210240002）、章苏尧 （19212010032）

本报告撰写人：周孟莹

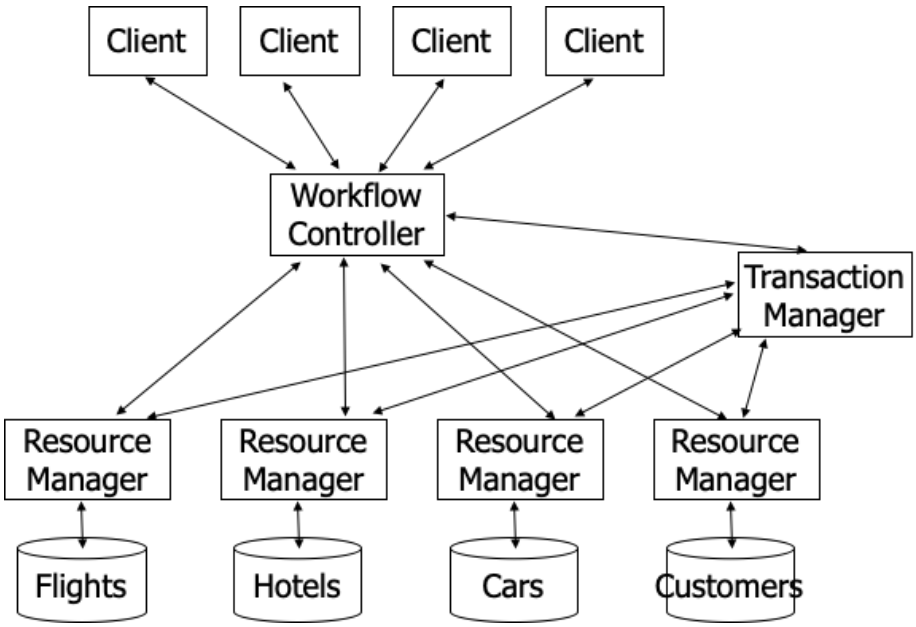
分工：

成员	分工	工作量
周孟莹 19110240001	负责协调整体项目、实现细化 Client 类、设计测试用例、撰写测试用例说明	33%
方睿钰 19210240002	实现 WorkflowController 类、编写 ReadMe 文件、撰写测试用例说明	33%
章苏尧 19212010032	实现 ResourceManager 类、调试测试用例、撰写 server 启动脚本和测试脚本	33%

一、项目设计与整体架构

本项目实现的是一个简单的分布式旅游预订系统，其中主要的数据主体为 Customer、Flight、Car、Room。Customer 可在系统上对其他三个主体进行预订和取消预订操作，另每个主体都设有增删改查的操作。

作为分布式数据系统，从设想上来看系统的各个组件以及底层数据存储都可在不同的机器上存在，因此本项目组件间的接口调用均采用 JavaRMI 模拟远程调用。本项目组件主要分为 WorkflowController（WC），TransactionManager（TM），ResourceManager（RM）三大部分，其中 WC 负责最外层的用户调用，通过与 TM 和 RM 的交互实现业务逻辑，使得 TM 和 RM 对用户透明；TM 负责管理事务，保证事务的 ACID 特性，负责事务的流程、处理错误和异常、事务回滚等；RM 则是最底层接触数据的部分，每个数据主体都由对应的资源管理器进行访问操作，实现增删改查，确保对数据持久性。



二、文件目录结构

- src
  - lockmgr: 已提供的实现好的锁管理器
  - test: Java 编写实现的测试用例
    - data: 运行生成的数据文件夹
    - TestFileObject.java: [TODO]
    - TestManager.java: 测试主程序
    - Makefile
    - 其它 Java 程序: 测试用例
  - transaction: 主要实现部分
    - data: 运行生成的数据文件夹
    - exceptions: 自定义的可能异常
    - models: 数据实体定义，包括接口 ResourceItem 与类 Car, Customer, Flight, Hotel, Reservation, ReservationKey
    - Client.java: 脚本测试时调用的客户端
    - MyClient.java: 默认简单客户端
    - ResourceManager.java: RM 接口
    - ResourceManagerImpl.java: RM 接口实现
    - RMTable.java: 数据表
    - TransactionManager.java: TM 接口（主要修改文件）
    - TransactionManagerImpl.java: TM 接口实现（主要修改文件）
    - Utils.java: 工具类，用于简化代码
    - WorkflowController.java: WC 接口
    - WorkflowControllerImpl.java: WC 接口实现（主要修改文件）
    - Makefile
- doc
  - 测试用例说明
  - 项目分工说明
  - 项目技术报告-周孟莹
  - 项目技术报告-方睿钰
  - 项目技术报告-章苏尧
- README.md
- run_server.sh
- run_test.sh
- stop_server.sh

三、Client 实现和操作

Client 类是用户（系统使用者）直接接触的类。用户通过 client 来创建 WC，并通过 WC 来获取 TM 和 RM 访问。

Client 是通过 Workflow Controller Interface 来调用 resource，业务接口包括下表：

组件启动接口		
Start	启动一个事务 id	start
Commit	提交一个事务	commit
Abort	终止一个事务	abort
基本业务逻辑接口		
Add*	添加 XX	addFlight, addRooms, addCars, newCustomer
Query*	查询 XX	queryFlight, queryFlightPrice, queryRooms, queryRoomsPrice, queryCars, queryCarsPrice, queryCustomerBill
Delete*	删除 XX	deleteFlight, addRooms, deleteRooms, addCars, deleteCars, deleteCustomer
Reserve*	预订 XX	reserveFlight, reserveCar, reserveRoom
异常模拟接口		
Die*	XX 崩溃	dieNow, dieRMAfterEnlist, dieRMBeforePrepare, dieRMAfterPrepare, dieTMBeforeCommit, dieTMAfterCommit, dieRMBeforeCommit, dieRMBeforeAbort

操作 client 表面看来与集中式数据库的访问没差别，例如：

```
int xid = wc.start();
System.out.println("Flight 347 has " +
        wc.queryFlight(6, "347") +
        " seats.");

if (!wc.addFlight(xid, "347", 230, 999)) {
    System.err.println("Add flight failed");
}
if (!wc.addRooms(xid, "SFO", 500, 150)) {
    System.err.println("Add room failed");
}

System.out.println("Flight 347 has " +
        wc.queryFlight(xid, "347") +
        " seats.");
```

```
if (!wc.reserveFlight(xid, "John", "347")) {
        System.err.println("Reserve flight failed");
}
System.out.println("Flight 347 now has " +
                wc.queryFlight(xid, "347") +
                " seats.");

if (!wc.commit(xid)) {
        System.err.println("Commit failed");
}
```

四、 测试用例设计

基于 Client 类的实现基础上，我们设计了几类测试用例来测试整体系统的可行性、可靠性、鲁棒性。测试的用例类型如下，详细内容可以参考测试用例文档：

- 1. 组件启动（无测试用例）
- 2. 组件启动功能测试 Basic*.java
  - 2.1. 正常运行测试
  - 2.2. 异常测试
- 3. 基本业务逻辑测试 CRUD*.java
  - 3.1. 正常运行测试
  - 3.2. 异常测试
- 4. ACID 性质测试 ACID*.java
  - 4.1. TM 宕机
  - 4.2. RM 宕机
  - 4.3. WC 宕机
- 5. 并行锁测试 Lock*.java
  - 5.1. 正常运行测试
  - 5.2. 死锁测试

1. 组件启动（无测试用例）

运行脚本文件`./run_server.sh`，若可正常输出（输出参考 README），则说明系统启动正常。

2. 组件启动功能测试 Basic*.java

2.1. 正常运行测试

- BasicBind.java

  - 测试目的：是否能正常绑定 WC 模块，并且能 start 事务

- BasicCommit.java
  - 测试目的: 是否能正常提交事务
- BasicAbort.java
  - 测试目的: 是否能正常终止

2.2. 异常测试

- BasicXid.java
  - 测试目的: 是否可检测出错误的事务 id(xid)

3. 基本业务逻辑测试 CRUD*.java

主要为 CRUD（create 增, read 读, update 改, delete 删）业务的独立以及组合测试。数据操作对象为 Flight, Room, Car 以及 Customer。可以添加修改删除 Flight, Room, Car, Customer。用户可以预订或者取消预订 Flight, Room, Car。具体包括以下测试用例：

3.1. 正常运行测试

- CRUDCreate.java
  - 测试目的：是否可正常添加 Flight，Room，Car，Customer，以及相关的 reservation
- CRUDRead.java
  - 测试目的：是否可以正常读取 Flight，Room，Car，Customer，以及相关的 reservation
- CRUDUpdate.java
  - 测试目的：是否可以正常修改 Flight，Room，Car，Customer，以及相关的 reservation

- CRUDDelete.java

  - 测试目的：是否可以正常删除 Flight，Room，Car，Customer，以及相关的 reservation

3.2. 异常测试

- CRUDCreateFailParam.java

  - 测试目的：是否可以正常检测出添加过程中的参数一查昂

- CRUDCreateFailInvalidItem.java

  - 测试目的：是否可正常检测出 reservation 中有误的参数

- CRUDDeleteFailSeq.java

  - 测试目的：是否可正常检测出试图删除有 reservation 的 Flight, Room, Car 的操作，并返回错误

- CRUDDeleteFailParam.java

  - 测试目的：是否可正常检测出删除是的参数异常

4. ACID 性质测试 ACID*.java

ACID 这四个性质的目的就是为了保证系统能在异常的出现的情况下，系统能在这些规则条件下保持数据的稳定

4.1. TM 宕机

- ACIDDieTMBeforeCommit.java

  - 测试目的：测试 Atomicity，确保事务 commit 成功前所做的操作无效

- dieTMAfterCommit.java

  - 测试目的：测试 Atomicity & Durability，确保已 commit 的事务对数据改动生效

- ACIDDieTM.java
  - 测试目的：测试 Atomicity，确保未提交的事务无效

4.2. RM 宕机

- ACIDDieRM.java
  - 测试目的：测试 Atomicity & Consistency，确保未提交时 RM 宕机，事务不成功
- ACIDDieRMAfterEnlist.java
  - 测试目的：测试 Atomicity & Consistency，确保 enlist 后 RM 的宕机会使事务失败，并且数据保留
- ACIDDieRMBeforePrepare.java
  - 测试目的：测试 Atomicity，确保 prepare 前 RM 的宕机会使事务失败
- ACIDDieRMAfterPrepare.java
  - 测试目的：测试 Atomicity
- ACIDDieRMBeforeCommit.java
  - 测试目的：测试 Atomicity
- ACIDDieRMBeforeAbort.java
  - 测试目的：测试 Atomicity & Consistency，确保在 abort 前宕机的 RMFlight 相关事务不会执行成功

4.3. WC 宕机

- ACIDDieWC.java
  - 测试目的：测试 Atomicity & Consistency & Durability

5. 并行锁测试 Lock*.java

5.1. 正常运行测试

- LockRR.java
  - 测试目的：测试 Durability
  - 场景：读读共享
- LockRW.java
  - 测试目标：测试 Isolation & Durability
  - 场景：读写等待，写 wait 读
- LockWR.java
  - 测试目的：测试 Isolation & Durability
  - 场景：写读共享，读 wait 写
- LockWW.java
  - 测试目的：测试 Isolation & Durability
  - 场景：写 wait 写

5.2. 死锁测试

- LockDead1.java
  - 测试目的：测试死锁情况的检测
    - 死锁理由：一个用户 A 访问表 A(锁住了表 A),然后又访问表 B；另一个用户 B 访问表 B(锁住了表 B)，然后企图访问表 A；这时用户 A 由于用户 B 已经锁住表 B，它必须等待用户 B 释放表 B 才能继续，同样用户 B 要等用户 A 释放表 A 才能继续，这就死锁就产生了。
- LockDead2.java
  - 测试目的：测试死锁情况的检测

- 死锁理由：用户 A 查询一条纪录，然后修改该条纪录；这时用户 B 修改该条纪录，这时用户 A 的事务里锁的性质由查询的共享锁企图上升到独占锁，而用户 B 里的独占锁由于 A 有共享锁存在所以必须等 A 释放掉共享锁，而 A 由于 B 的独占锁而无法上升的独占锁也就不可能释放共享锁，于是出现了死锁。

测试用例通过 TestManager.java 来进行统一的管理和运行。如果需要一次性运行所有的测试用例，可以直接运行 run_test.sh，运行结果如下：

```
root@c04a7ea5b0f1:/home/DDB/src/test# make clean all test
rm -rf ./data/*
rm -rf ./results/*
rm -f *.class
javac -classpath .. *.java
Note: TestFileObject.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
java -classpath .. -DrmiPort=3345 -DtestClass=BasicXid test.TestManager
Launching test BasicXid
Test BasicXid passed.
java -classpath .. -DrmiPort=3345 -DtestClass=BasicBind test.TestManager
Launching test BasicBind
Test BasicBind passed.
java -classpath .. -DrmiPort=3345 -DtestClass=BasicCommit test.TestManager
Launching test BasicCommit
Test BasicCommit passed.
java -classpath .. -DrmiPort=3345 -DtestClass=BasicAbort test.TestManager
Launching test BasicAbort
Test BasicAbort passed.
java -classpath .. -DrmiPort=3345 -DtestClass=CRUDCreate test.TestManager
Launching test CRUDCreate
Test CRUDCreate passed.
java -classpath .. -DrmiPort=3345 -DtestClass=CRUDRead test.TestManager
Launching test CRUDRead
Test CRUDRead passed.
java -classpath .. -DrmiPort=3345 -DtestClass=CRUDCreateFailInvalidItem test.TestManager
Launching test CRUDCreateFailInvalidItem
Test CRUDCreateFailInvalidItem passed.
java -classpath .. -DrmiPort=3345 -DtestClass=CRUDCreateFailParam test.TestManager
Launching test CRUDCreateFailParam
Test CRUDCreateFailParam passed.
java -classpath .. -DrmiPort=3345 -DtestClass=CRUDDelete test.TestManager
Launching test CRUDDelete
Test CRUDDelete passed.
java -classpath .. -DrmiPort=3345 -DtestClass=CRUDDeleteFailParam test.TestManager
Launching test CRUDDeleteFailParam
Test CRUDDeleteFailParam passed.
java -classpath .. -DrmiPort=3345 -DtestClass=CRUDDeleteFailSeq test.TestManager
Launching test CRUDDeleteFailSeq
Test CRUDDeleteFailSeq passed.
java -classpath .. -DrmiPort=3345 -DtestClass=CRUDUpdate test.TestManager
Launching test CRUDUpdate
Test CRUDUpdate passed.
java -classpath .. -DrmiPort=3345 -DtestClass=ACIDDieRM test.TestManager
Launching test ACIDDieRM
Test ACIDDieRM passed.
java -classpath .. -DrmiPort=3345 -DtestClass=ACIDDieRMAfterEnlist test.TestManager
Launching test ACIDDieRMAfterEnlist
Test ACIDDieRMAfterEnlist passed.
```