Beijing Forest Studio

北京理工大学信息系统及安全对抗实验中心

MySQL事务机制

MySQL Transaction

陈骋　硕士研究生

2018年10月28日

内容提要

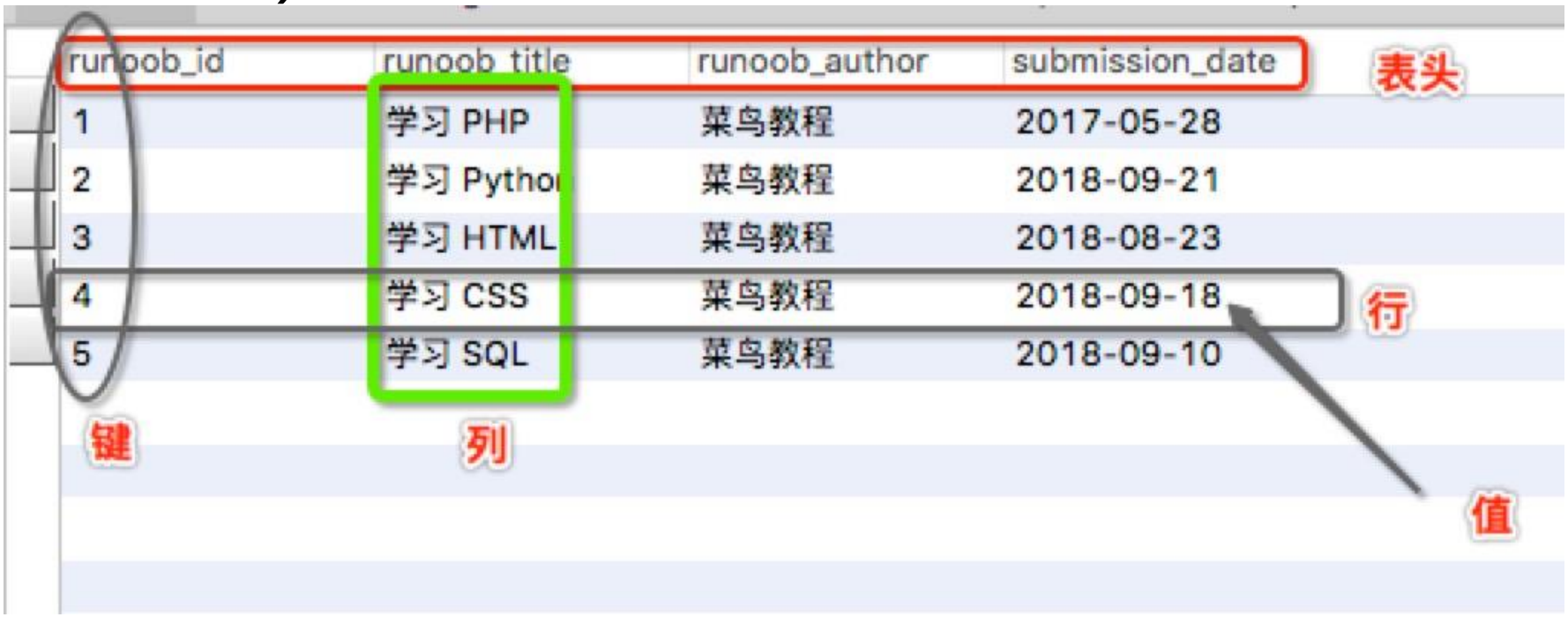
- 背景简介
- 基本概念
- MVCC机制
- 应用总结
- 参考文献

背景简介

- 预期收获
  - 1. 了解MySQL事务机制
  - 2. 理解事务机制的实现原理
  - 3. 了解分布式环境下事务的应用

背景简介

- MySQL 是最流行的关系型数据库管理系统，在WEB应用方面 MySQL 是最好的RDBMS(Relational Database Management System：关系数据库管理系统)应用软件之一。



背景简介

- 何为事务机制?
  - 事务是MySQL中innoDB引擎中特有的功能
  - 事务是用来保证一组数据操作的完整性和一致性


```
Begin;
Update 库存;
Insert into 订单;
Update 付款;
Commit;
```

事务的ACID

- **原子性**(A)：事务是最小单位，不可再分
- **一致性**(C)：事务要求所有的语句操作的时候，必须保证同时成功或者同时失败
- **隔离性**(I)：事务A和事务B之间具有隔离性
- **持久性**(D)：是事务的保证，事务终结的标志(内存的数据持久到硬盘文件中)

背景简介

脏读(Drity Read)：某个事务已更新一份数据，另一个事务在此时读取了同一份数据，由于某些原因，前一个RollBack了操作，则后一个事务所读取的数据就会是不正确的。

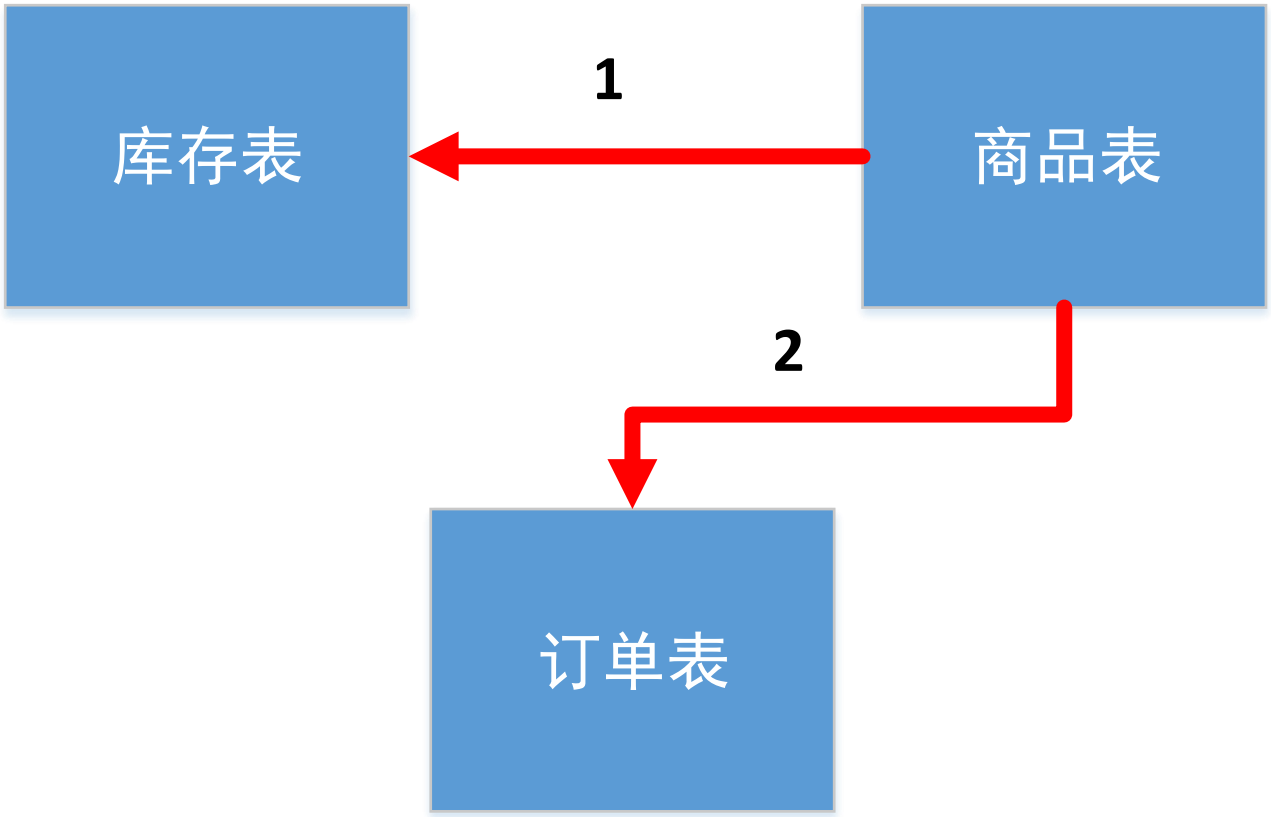
不可重复读(Non-repeatable read):在一个事务的两次查询之中数据不一致，这可能是两次查询过程中间另一个事务更新的原有的数据。

幻读(Phantom Read):在一个事务的两次查询中数据笔数不一致，例如有一个事务查询了几列(Row)数据，而另一个事务却在此时插入了新的几列数据，先前的事务在接下来的查询中，就会发现有几列数据是它先前所没有的。

背景简介

- 事务的隔离级别
  - 读未提交：read uncommitted （事物A和事物B，事物A未提交的数据，事物B可以读取到）
  - 读已提交：read committed（事物A和事物B，事物A提交的数据，事物B才能读取到）
  - 可重复读：repeatable read（事务A和事务B，事务A提交之后的数据，事务B读取不到）
  - 串行化：serializable（事务A和事务B，事务A在操作数据库时，事务B只能排队等待）

背景简介



背景简介

- 1、Begin;
  4、Update 库存 set 数量 = 数量-1 where id = 1;
  。 。 。
  7、Commit;

- 2、Begin;
  3、查看库存信息;
  5、查看库存信息;
  6、Update 库存 set 数量 = 数量-1 where id = 1;
  。 。 。
  8、查看库存信息
  9、Commit;

基本概念

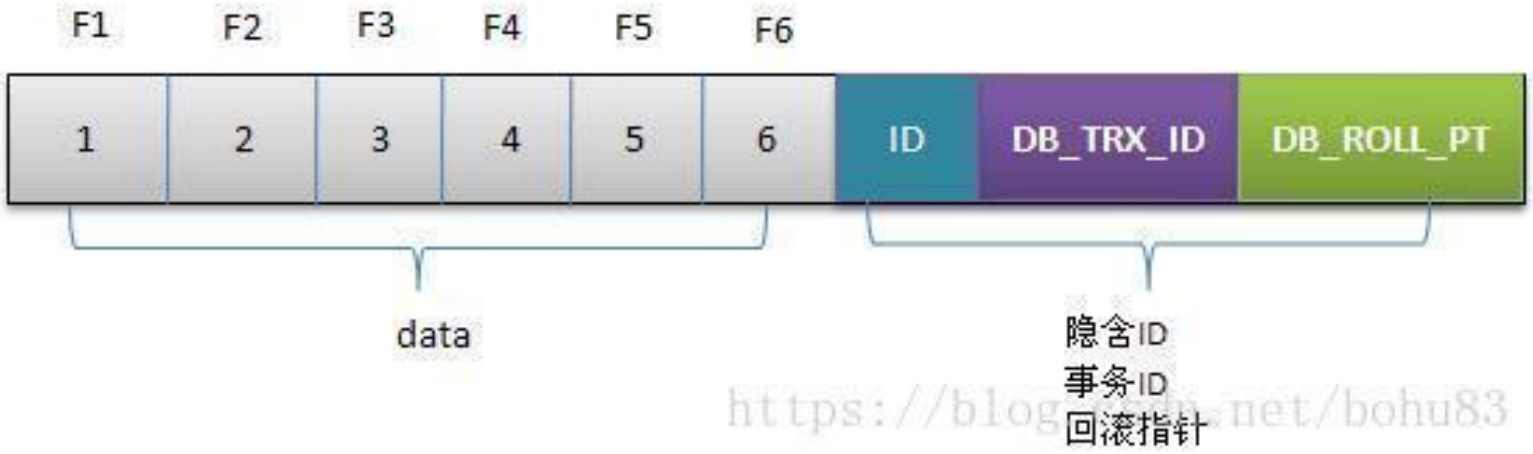
隔离级别	脏读	不可重复读	幻读
读未提交	可能	可能	可能
读已提交	不可能	可能	可能
可重复读	不可能	不可能	对Innodb不可能
串行化	不可能	不可能	不可能

事务隔离性的实现

- MVCC机制
- 可以认为MVCC是行锁的一个变种，大都实现了非阻塞的读操作，写操作也只锁定了必要的行。
- MVCC的实现，是通过保存数据在某个时间点的快照来实现的，根据事务开始时间的不同，每个事务对于同一张表，同一时刻看到的数据可能是不同的。

MVCC机制

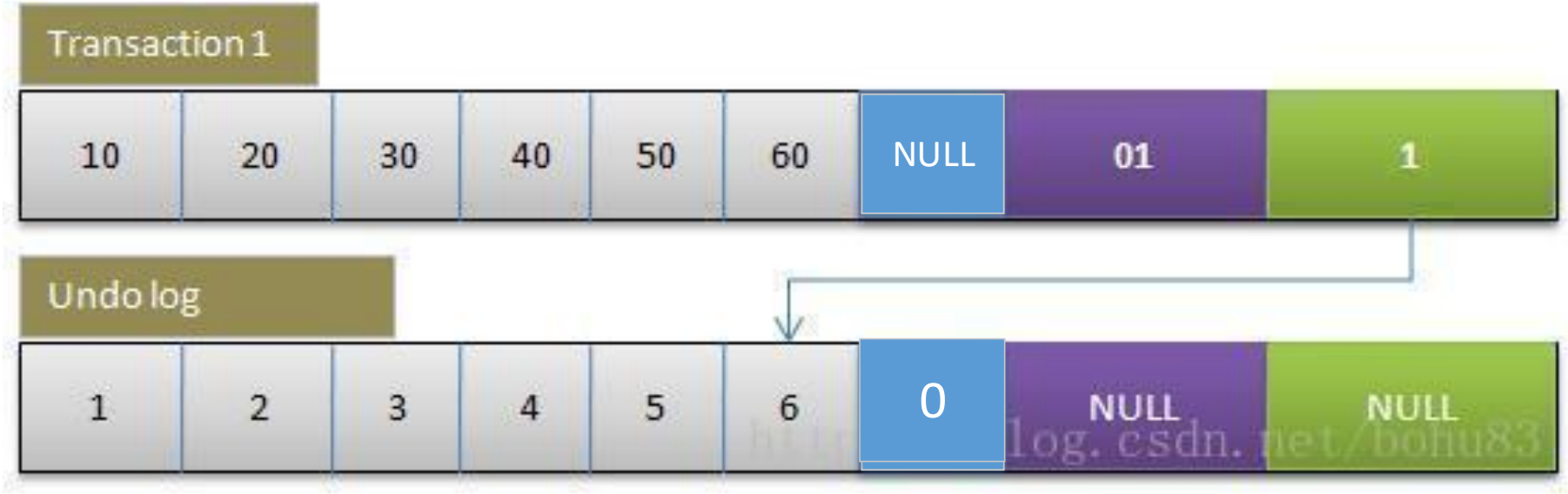
- InnoDB的 MVCC ，是通过在每行记录的后面保存两个隐藏的列来实现的。这两个列， 一个保存了行的创建时间，一个保存了行的过期时间（或删除时间）， 当然存储的并不是实际的时间值，而是系统版本号。



- F1~F6是某行列的名字，1~6是其对应的数据。后面三个隐含字段分别对应该行的删除和创建事务号和回滚指针，假如这条数据是刚INSERT的，可以认为三个字段为空。

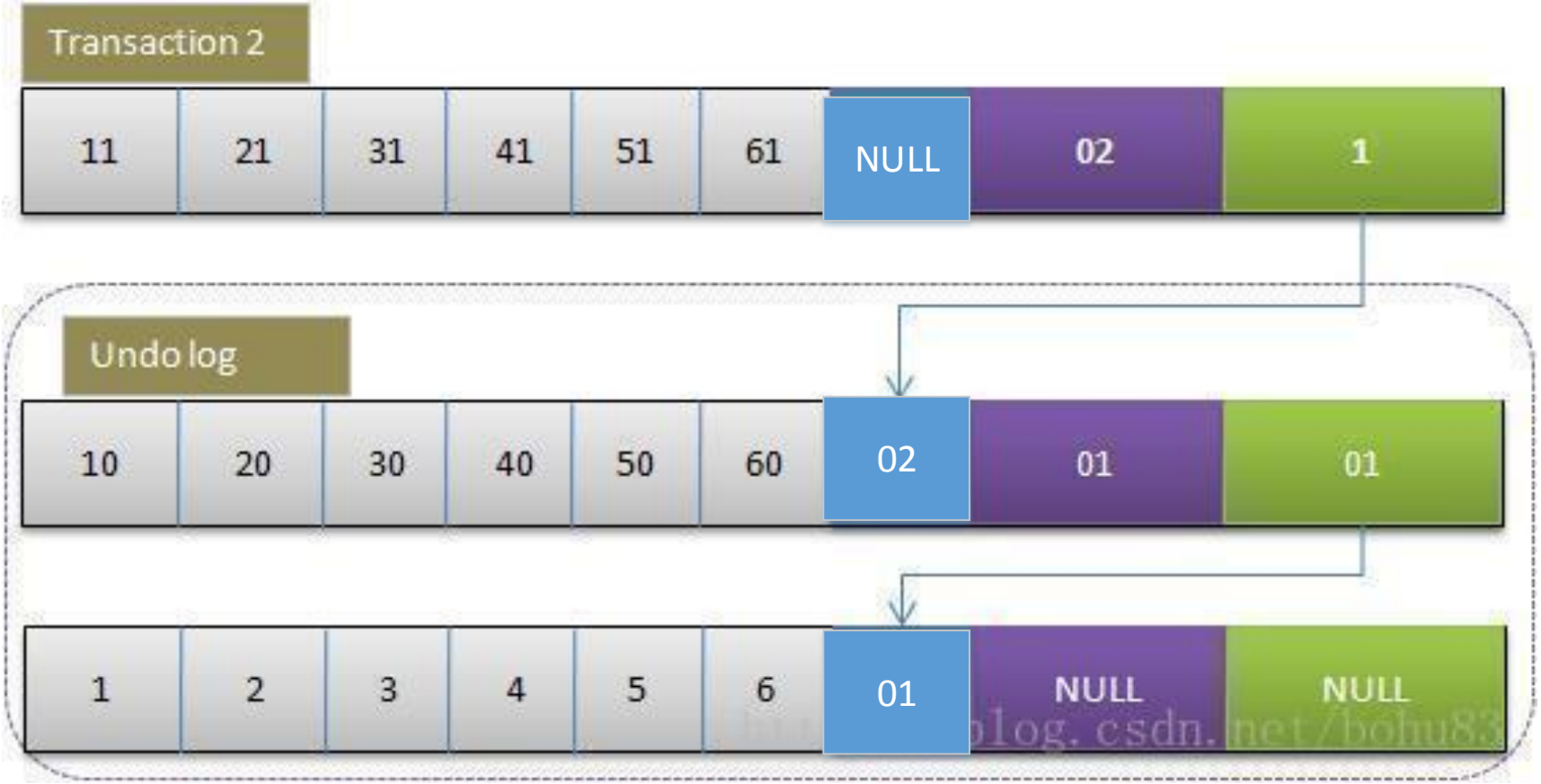
MVCC机制

- 事务1更改该行的各字段的值
- 把该行修改前的值Copy到undo log，即图中下面的行
- 修改当前行的值，填写事务编号，使回滚指针指向undo log中的修改前的行。



MVCC机制

- 事务2修改该行的值
- 与事务1相同，此时undo log中有两行记录，并且通过回滚指针连在一起。



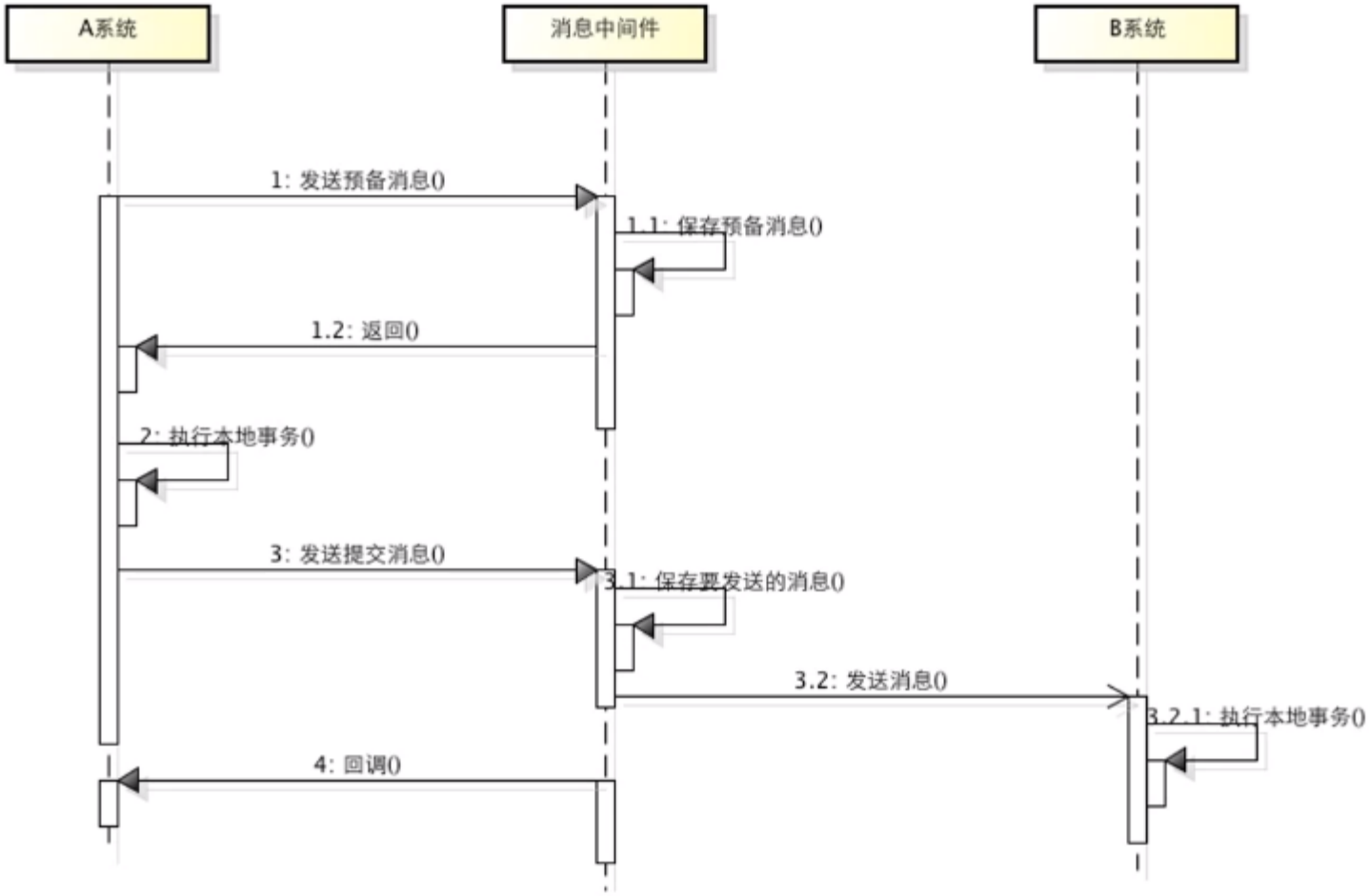
分布式事务

- 两段式和三段式事务
- 基于消息的最终一致性方案
- TCC柔性补偿性方案

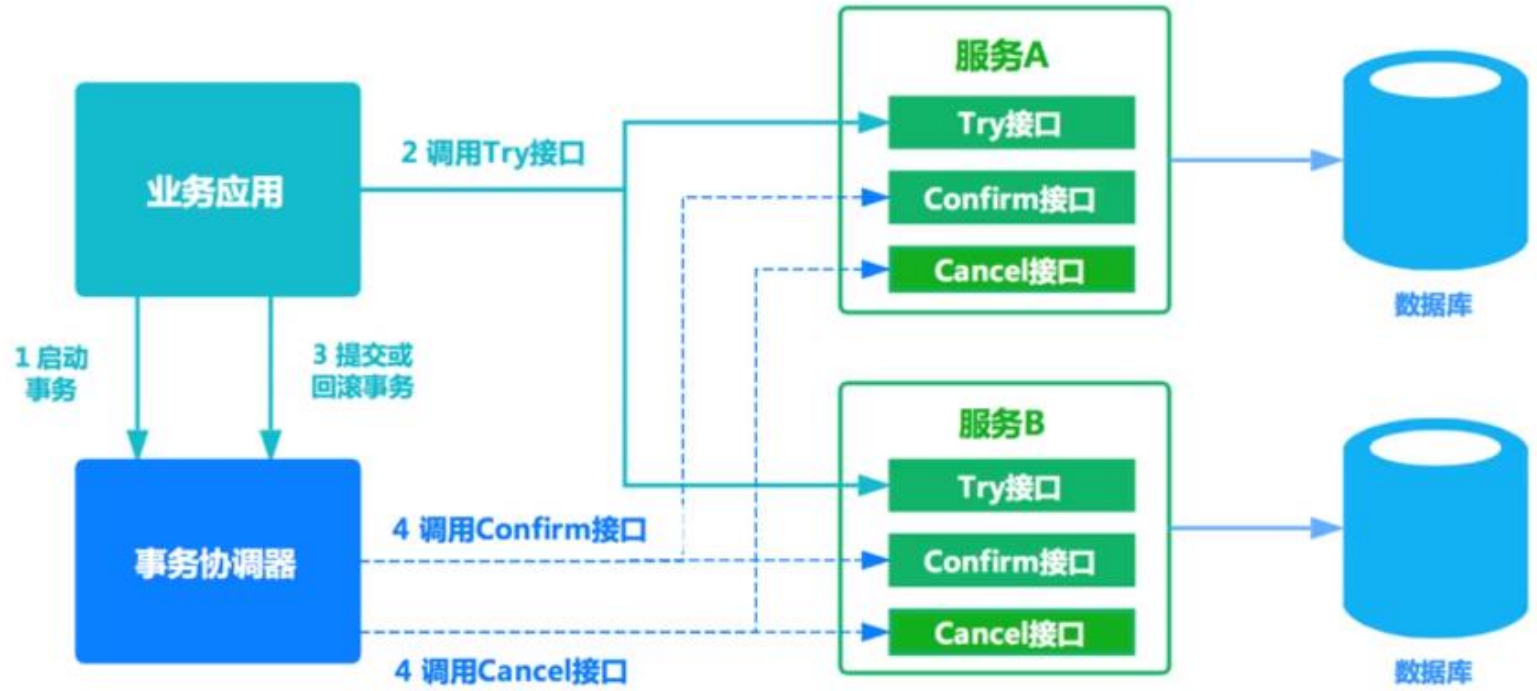
两段式事务


事务
管理器
预备（ Prepare ）
就绪（ Ready ）
资源管理器
预备（ Prepare ）
就绪（ Ready ）
资源管理器
第一阶段
（ PHASE I ）
事务
管理器
提交（ Commit ）
已提交（ Committed ）
资源管理器
提交（ Commit ）
已提交（ Committed ）
资源管理器
第二阶段
（ PHASE II ）

基于消息的一致性方案


A系统
消息中间件
B系统
1: 发送预备消息()
1.1: 保存预备消息()
1.2: 返回()
2: 执行本地事务()
3: 发送提交消息()
3.1: 保存要发送的消息()
3.2: 发送消息()
3.2.1: 执行本地事务()
4: 回调()

TCC补偿性事务实现


业务应用
2 调用Try接口
1 启动事务
3 提交或回滚事务
事务协调器
4 调用Confirm接口
4 调用Cancel接口
服务A
Try接口
Confirm接口
Cancel接口
服务B
Try接口
Confirm接口
Cancel接口
数据库
数据库

参考文献

[1]https://blog.csdn.net/bohu83/article/details/80941162
[2]高性能MySQL
[3]https://blog.csdn.net/w_linux/article/details/79666086
[4]https://blog.csdn.net/tanga842428/article/details/52748531

谢谢！

大成若缺，其用不弊。大盈若冲，其用不穷。大直若屈。大巧若拙。大辩若讷。静胜躁，寒胜热。清静为天下正。