# CH17 Database Recovery故障恢复

## 一、Issues and Models for Resilient Operation

### 1. Failures Classification

- 磁盘故障
- 系统崩溃
- 事务故障

### 2. Transaction

#### 2.1 ACID

> Atomic : 原子性，意味着事务中的所有操作要么全部执行成功，要么全部失败回滚，不存在部分执行的情况。
>
> Consistency : 事务执行的结果必须使数据库从一个一致性状态转变到另一个一致性状态。这意味着事务在执行前后，数据库必须保持一致性，不会破坏数据完整性和约束。
>
> Isolated : 隔离性，事务的执行不会受到其他事务的干扰
>
> Durable : 一旦事务提交，其所做的修改将永久保存在数据库中，并且不会因系统故障或崩溃而丢失。即使系统重启或发生故障，已经提交的事务所做的修改也不会丢失。

#### 2.2 Defining

隐式：单条sql语句

显式：

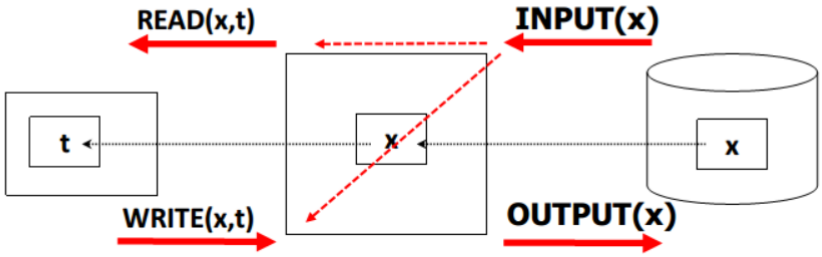
```
BEGIN TRANSACTION
    statements
COMMIT              -- complete

BEGIN TRANSACTION
    statements
ROLLBACK            -- abort
```

#### 2.3 Consistent state

满足所有约束的状态。

一致状态$\rightarrow T \rightarrow$一致状态

### 3. Primitive operations of transaction




**Local address space of transaction (Memory)** **Buffer(Memory)** **Space of Disk**

四种操作。

事务执行之前INPUT，事务执行完毕之后OUTPUT。

# 二、Undo Logging

## 1. definition

用于写内容的文件，只有追加模式。

## 2. Records

START $T_i$　　启动

COMMIT $T_i$　提交

ABORT $T_i$　　回滚

$T_i, X, v$　　事务改变了$X$的值，且事务的原始值为$v$

日志首先写入内存，不记录新值

恢复器：将所有修改的值改为原来的值。

## 3. Write Rules

U1 : 在修改数据写回磁盘之前触发undo log

U2 : 等到所有数据写回磁盘之后再将commit日志写入磁盘

写回顺序：undo log→数据→COMMIT记录

若先写COMMIT，数据在写回磁盘时数据库崩溃，由于undo log中有COMMIT，故原有数据丢失，数据库一致性破坏。

每条Write指令产生一条undo log。

## 4. Recovery rules

对于没有commit但是有start的事务undo日志而言，从最新到最旧，反复执行`write(x, v)`和`output(x)`

恢复过程中发生错误，无关紧要，当DBMS恢复时重新执行即可。

### 5. Checkpointing

commit之前的日志不可删除。

避免检查整个文档。

执行完一部分事务将数据写入磁盘，undo日志写入ckpt，继续接收事务。

逆向扫描时遇到checkpoint结束。

## 三、Redo Logging

### 1. Problem of undo log

过多的磁盘I/O操作。

为了减少I/O操作，让变化的数据先存在主存中，使用redo日志解决数据库崩溃。

### 2. Differences between undo and redo

undo : 取消不完整的事务，忽略已经提交的部分，commit之前数据写回磁盘，存储旧值

redo : 忽略不完整的事务，重复已经提交的部分，commit之后数据写回磁盘，存储新值

### 3. Write Rules

R1 : 修改磁盘中的任意元素之前，所有日志记录和commit记录都得出现在磁盘中

写回顺序：redo log→commit log→ update DB

使用redo log推迟修改。

日志优先数据持久化到硬盘上。

### 4. Recovery rules

不完整的事务由于没有commit，就当没发生过。没有commit，写入abort。

完整的事务，将新值写入内存。 从begin写到commit。

使用Redo log恢复数据 is very *very* **very** *very* slow

从第一条记录写到最后一条commit

### 5. Checkpointing

不接受新事务，直到所有事务执行完毕，刷新所有的日志到磁盘，刷新所有的缓存池到数据盘，日志中写入ckpt。

重新接收新事务。

## 四、Undo/Redo Logging

***flush log： 将log持久化到硬盘

### 1. Comparison

- Undo : 增大磁盘I/O
- Redo : 增大缓冲区

### 2. Undo/Redo Rule

记录$T_i, X, v, w$

旧值$v$，新值$w$

UR1 : 事务修改数据之前，将上述记录写入磁盘

数据X可以在提交之前或之后持久化到硬盘。commit可以在在任意磁盘变化之前或之后写入硬盘

UR2 : 在commit写入log之后立即持久化磁盘。

### 3. Recovery Rule

Redo已经提交的事务，Undo未提交的日志。

***所以到底写不写数据呢？？？

## 五、Protecting against Media Failures

媒体数据错误：非易失性数据丢失

将媒体数据备份N份。

input : 1（循环检查合法数据，直至找到第一份合法数据）{非法数据可检测}

output : N

使用redo日志将数据从备份数据库写回原数据库。