Write Ahead Log System

---

id: 519021910861

name: huidong xu

简介

运行

将命令放在 "cmd.in" 文件中并直接从该文件中读取输入、

```
./wal-sys.py -reset < cmd.in
```

知识点回顾 - **Checkpoint**

- Recovery rules
  a. travel from end to start.
  b. mark all transaction's log record **w/o CNT** log and append **ABORT log**.
  c. **UNDO ABORT logs** from **end to start**.
  d. **REDO CMT logs** from **start to end**.
- Why **Checkpoint**
  a. avoid log size from growing too large.
  b. avoid recovery from a blank state.

- How **Checkpoint**
  CKRT_ROOT: points to the last checkpointed log file, log content before can be safely discard.
    a. wait till no actions are in progress.
    b. write a **CKRT** record to log, contains a list of all actions in process and pointers to their most recent previous log records.
    c. save all files.
    d. atomically save checkpoint by updating checkpoint root to new CKRT record.

Checkpoints

我将命令写入文件 `cmd.in` 中，每次运行 `./wal-sys.py -reset < cmd.in`。

Q1:During checkpoint, wal-sys divides actions into three types: "PENDING", "COMMITED" and "DONE", what is the meaning of these types?

"PENDING" 就是还没有 "commit"， "COMMITED" 就是已经"commit"但还没有 "end"， "DONE" 就是已经"end"了。

Q2: What is the relationship between the action categories during checkpoing("PENDING", "COMMITED" and "DONE") and action categories during recovery("Winners", "Losers", and "Done")?

checkpoint 分为三种情况：

- PENDING：事务在 checkpoint 之前没有 `commit`。
- COMMITED：事务在 checkpoint 之前 `commit` 但没有 `end`。
- DONE：事务在 checkpoint 之前 `end`。

recovery 也分为三种情况：

- WINNER：事务在 `crash` 之前 `commit`了，但没有 `end`。
- LOSER：事务在 `crash` 之前没有 `commit` 且没有 `end`。
- DONE：事务在 `crash` 之前 `end`。

且 checkpoint 必然发生在 crash 之前，两者关联即为：

1. 如果在 checkpoint 前已经 `end` 了，则在 checkpoint 中是 DONE 在 recovery 中也是 DONE。
2. 如果在 checkpoint 前已经 `commit`，且在 crash 前没有 `end`，则在 checkpoint 中是 COMMITED，在 recovery 中是 WINNER。
3. 如果在 checkpoint 前已经 `commit`，且在 checkpoint 和 crash 之间已经 `end`，则在 checkpoint 中是 COMMITED，在 recovery 中是 DONE。
4. 如果在 checkpoint 前还没有 `commit`，则在 checkpoint 中是 PENDING，在 recovery 中是 LOSER。

Q3:How many lines were rolled back? What is the advantage of using checkpoints?

仅在 checkpoint 之后行会被回滚。在本次 checkpoint时，我们有如下结果：

```
PENDING: id : 3 id: 2 COMMITED: DONE: id : 1
```

即我们会创建指针分别指向 id 为 2 和 3 的 actions 的最近记录，并以此为开始向后进行恢复。

```
commit 2
devit_account 3 studentC 100
```

回滚行数为 8 行。

```
The log was rolled back 8 lines
```

这样做可以避免 log 文件过大（安全删除 CKRT_ROOT 之前的内容），从而缩短数据库恢复的时间。而且还可以避免从未知状态中的恢复。

Q4:Does the second run of the recovery procedure restore "DB" to the same state as the first run? What is this property called?

是的，这种属性叫做持久性（Durability）。持久性即指一个事务一旦被提交，它对数据库中数据的改变就是永久性的，数据库故障不应该对其产生影响。

Q5: Compare the `action_ids` of "Winners", "Losers", and "Done" from the second recovery with those from the first. The lists are different. How does the recovery procedure guarantee the property from Q4 even though the recovery procedure can change?

第一次恢复

```
Winners: id : 2 Losers: id : 3 Done: id : 1
```

第二次恢复

```
Winners:         Losers: id : 3 Done: id : 1 id : 2
```

第一次恢复时在 LOG 文件中增加了一行

```
type: END action_id: 2
```

这相当于将进程 2 "end" 了，因此可以将其从 "Winner" 中移到 "Done"，在第二次恢复不再考虑其影响。

Q6: Wal-sys has a hitherto unmentioned option: if you type wal-sys -undo it will perform undo logging and undo recovery. Try the above sequences again with undo logging to see what changes.

相当于一个 "UNDO-only" Logging，因此要将所有改变（无论是否 commit）都直接写入数据库，在恢复时根据 "commit" 决定是否 "UNDO"。