Week#14 Logging & Recovery in SQLite

Sangeun Chae

2018314760

# INTRODUCTION

데이터베이스 저널이란, 데이터베이스가 예상하지 못한 이벤트 (Ex: 예상하지 못한 shutdown) 가 발생할 때 복구를 위해 필요한 시스템이다. SQLite에서 지원하는 journaling mode는 총 6가지가 있다. Delete (default), Truncate, Memory, Persist, WAL 그리고 None(Off). 이번 랩에서는 6가지의 journaling mode중, Delete (default) mode와 WAL mode에 대해 알아보고자 한다.

**Delete (default) Journaling mode**

Delete journaling mode는 SQLite에서 지원하는 default journaling mode이다. Delete mode는 하나의 트랜잭션 마다 rollback을 위한 journal 파일을 매번 생성 및 삭제한다. 즉, journal 파일을 생성하여 기존 database의 “old” 한 데이터를 journal 파일에 저장한다. Rollback operation 수행 시, journal 파일에 저장되어 있는 “old” 데이터를 데이터베이스에 복구한다.

**WAL Journaling mode**

WAL journaling mode는 SQLite에서 지원하는 journaling mode중에 하나이다. WAL journaling mode는 다른 journal mode와는 다르게, “old”한 데이터를 journal 파일에다 저장하는 방식이 아닌, 새롭게 insert되거나 update되는 데이터를 log 파일에 저장한다. 이후 batch 형태로 한 번에 데이터를 commit 하는 방식을 따른다. WAL (Log) file은 DB Connection이 처음 이뤄질 때 생성되고, last connection 이 close될 때 제거된다. 하지만 정상 종료하지 않고 예외상황이 발생하면 WAL 파일을 남게 되고, 다음에 DB open 시 이 정보를 활용해 DB를 원상 복구한다.

# METHODS

전반적인 실험은 pytpcc benchmark를 통해서 이루어진다. 벤치마크를 실행하기 전에, 우선 TPC-C database를 load하고, load 된 데이터를 기준으로 벤치마크를 실행하여 성능을 비교한다. 벤치마크를 실행할 때는 journal mode를 delete mode와 WAL mode로 변화를 주어 각각 실험을 진행한다. 각각의 journal mode를 통해서 나온 결과를 redirection을 통해 따로 파일로 저장하고, 실험이 모두 종료되면 각각의 결과를 비교 및 대조하여 분석한다.

# Performance Evaluation

## Experimental Setup

Specify the experimental setup (e.g., OS, Linux version, kernel version, CPU spec, DRAM size, storage devices, etc.) and benchmark setup (e.g., database size, # of concurrent threads, running time).

|  |  |
| --- | --- |
| **Type** | **Specification** |
| OS | Ubuntu 18.04.5 |
| CPU | Intel® Xeon® Gold 5125 CPU 2.50GHz (10 Core, 40 Threads) |
| Memory | 64GB |
| Kernel | Linux 4.19.108 |
| Storage | SSD 860 PRO 512GB (SATA) |

Table 1: System Setup

|  |  |
| --- | --- |
| **Type** | **Configuration** |
| Benchmark Type | Pytpcc benchmark |
| Warehouse | 10 |
| Duration | 1800s |
| Journal mode | Delete, WAL |

Table 2: Benchmark Setup

## Experimental Results

Delete journaling mode와 WAL journaling mode에 대한 transaction per throughput과 throughput average는 다음과 같다.

Figure 1: Throughput per Journal mode

Figure 2: Average throughput per journal mode

**[Figure 1]**을 통해 확인할 수 있듯이, WAL journal mode가 전반적으로 transaction per throughput 이 Delete journal mode에 비해 높은 것을 알 수 있다. WAL journal mode와 Delete Journal mode의 차이는 journal file에 저장하는 데이터 종류에서 발생한다. WAL journal mode에서는 새롭게 insert되는 데이터나 update되는 데이터를 journaling file (WAL file)에 저장한다. 하지만 Delete journal mode는 “old”한 데이터를 journal file에 저장하고, 새롭게 insert 되거나 update되는 데이터는 데이터베이스에 저장한다. 데이터베이스에서도 데이터가 파일로 저장이 되기 때문에, delete journal mode의 경우에는 disk I/O (File I/O)가 두 번 일어나게 된다. 하지만 WAL journaling mode의 경우에는 새롭게 저장하는 데이터를 WAL file에 저장하기 때문에, commit이 일어나기 전까지는 데이터베이스 파일에 접근하지 않는다. 따라서 commit이 일어나는 경우가 아닐 때는 disk I/O (File I/O)가 한 번만 일어나게 되고, commit이 일어나게 되더라도 batch형태로 한 번에 데이터베이스에 쓰기가 일어나기 때문에 file open system call과 file close system call, file create system call, file delete system call이 불리는 횟수가 delete journal mode에 비해 현저히 적게 일어난다. 따라서 **[Figure 1]** 과 **[Figure 2]**에서 같은 결과가 도출되게 된다.

[**The root cause of the performance gap between delete mode and wal is Number of I/O occurs]**

# Conclusion

데이터베이스의 성능에 영향을 주는 요소 중, 가장 빈번하게 발생하는 성능 저하 요인은 I/O 횟수의 증가이다. 이번 랩에서 살펴본, Delete (default) Journaling mode과 WAL journaling mode는 I/O 횟수에서 큰 차이가 있다. WAL journaling mode는 특정 상황이 아니면, Disk I/O가 한 번 밖에 일어나지 않는다. 하지만 Delete Journaling mode의 경우에는 Disk I/O가 데이터가 새롭게 insert되거나 update될 때 마다 두 번씩 일어난다. 따라서 성능적인 부분에서는 WAL journaling mode가 Delete (default) Journaling mode에 비해 월등이 뛰어남을 알 수 있다.

# REFERENCES

1. <https://github.com/meeeejin/SWE3033-F2021/tree/main/week-14>
2. <https://www.sqlite.org/pragma.html#pragma_journal_mode>