### 临时文件概述

SQLite3中有9种临时文件，它们是：

1. 回滚日志文件(Rollback journal)
2. 主日志文件(Master journals)
3. Write-ahead Log(WAL) files
4. 共享内存文件(Shared-memory files)
5. Statement journals
6. 临时数据库(TEMP databases)
7. Materializations of views and subqueries
8. 临时索引(Transient indices)
9. VACUUM使用的临时数据库(Transient databases used by VACUUM)

### 回滚日志

回滚日志用来实现原子提交(atomic commit)和回滚能力(rollback capabilities).

回滚日志文件一般和数据库文件在同一目录，并且拥有相同的文件名，除了后缀有"-journal".

回滚日志一般在第一次使用事务的时候创建，在事务提交或回滚后删除。

回滚日志是实现原子提交和回滚能力的基础。如果没有回滚日志，SQlite就不能回滚一个没有完成的事务。

如果没有回滚日志，当程序在操作数据库过程中崩溃或机器断电时整个数据库就会出错。

当程序在操作数据库过程中崩溃或机器断电时,回滚日志会保留在磁盘上。下一次打开数据库文件时，会使用回滚日志中的日志还原数据库文件到未完成的事务之前的状态。

[exclusive locking mode](https://www.sqlite.org/pragma.html#pragma\_locking\_mode)

PRAGMA locking\_mode=EXCLUSIVE

当设置参数，使用EXCLUSIVE的locking\_mode时,回滚日志不会被删除，知道退出EXCLUSIVE的locking\_mode

[jorunal\_mode\_pragma](https://www.sqlite.org/pragma.html#pragma\_journal\_mode)

PRAGMA schema.journal\_mode = DELETE | TRUNCATE | PERSIST | MEMORY | WAL | OFF

回滚日志的创建和删除还随参数jorunal\_mode\_pragma控制，默认的模式时DELETE，

出现在事务的起始点，消失在事务的结束点

若PRAGMA locking\_mode=EXCLUSIVE; // 默认为normal

则不会在事务结束点删除回滚日志

若pragma journal\_mode=persist; // 默认为delete

也不会在事务结束点删除回滚日志

若pragma journal\_mode=off;则不会产生回滚日志

存储于磁盘上

### Write-Ahead Log(WAL)文件

当jorunal\_mode\_pragma设置为WAL模式时，使用Write-Ahead Log文件替换回滚日志。

回滚日志文件和数据库文件有相同的文件名，除了后缀多了-wal。

WAL文件在第一次连接上数据库文件时创建，最后一次断开和数据库的连接时删除。当最后一次断开数据库的连接不正常，就会保留下来，知道下一次打开数据库文件时清理内容。

### 共享内存文件

在WAL模式

存储于磁盘上

Master Journal主日志

在一个数据库连接使用attach命令接入N个外部数据库的情况下，事务开始点产生的回滚日志

存储于磁盘上

### Statement Journal Files

针对单个SQL语句的回滚日志

比如，我们执行一条UPDATA语句会改变数据库文件中100条记录的内容，但是实际执行过程中改变50条记录内容后UPDATE语句发生问题无法执行了。这时Statement Journal文件就用来撤销前面50条记录的修改，让数据库还原到执行UPDATE语句之前的状态。

Statement Jorunal只会在使用UPDATE和INSERT语句时创建，这些语句可能改变数据库的多条记录内容，并且在执行中间发生问题。

Statement Journal会随机产生一个名字

存储于磁盘上

### TEMP Databases

通过"CREATE TEMP TABLE"创建的临时表只针对当前环境可见。这些临时表以及和它相关联的索引、触发器、视图会单独存储在一个临时文件中，当断开和数据库的连接时会自动删除。

使用PRAGMA synchronous=OFF

使用PRAGMA journal\_mode=PERSIST

操作速度较快，一般用于临时日志的记录操作

Materializations of views and subqueries

Queries that contain subqueries must sometime evaluate the subqueries separately and store the results in a temporary table, then use the content of the temporary table to evaluate the outer query. We call this "materializing" the subquery. The query optimizer in SQLite attempts to avoid materializing, but sometimes it is not easily avoidable. The temporary tables created by materialization are each stored in their own separate temporary file, which is automatically deleted at the conclusion of the query. The size of these temporary tables depends on the amount of data in the materialization of the subquery, of course.

\*SQL优化点：\*要避免创建

存储视图、子查询等临时表

SELECT \* FROM ex1 WHERE ex1.a IN (SELECT b FROM ex2);

子查询语句 SELECT b FROM ex2执行后的结果会存储在一个零时表中.

为了避免创建临时表，可以写成：

SELECT \* FROM ex1 WHERE EXISTS(SELECT 1 FROM ex2 WHERE ex2.b=ex1.a);

### Transient indices

临时索引

瞬时索引，比如

ORDER BY从句

GROUP BY从句

DISTINCT聚合查询关键词

复合SELECT从句joined by UNION, EXCEPT或INTERSECT

### Transient databases used by VACUUM

使用VACUUM压缩数据库文件时使用的临时文件

6-9中的几种临时文件通过参数[PRAGMA temp\_store = 0 | DEFAULT | 1 | FILE | 2 | MEMORY;](https://www.sqlite.org/pragma.html#pragma\_temp\_store)来控制存储位置。

|SQLITE\_TEMP\_STORE| PRAGMA temp\_store| Storage used for TEMP tables and indices

|---|---|---|

|0 |any |file

|1 |0 |file

|1 |1 |file

|1 |2 |memory

|2 |0 |memory

|2 |1 |file

|2 |2 |memory

|3 |any |memory