데이터베이스 구성

Enterprise applications

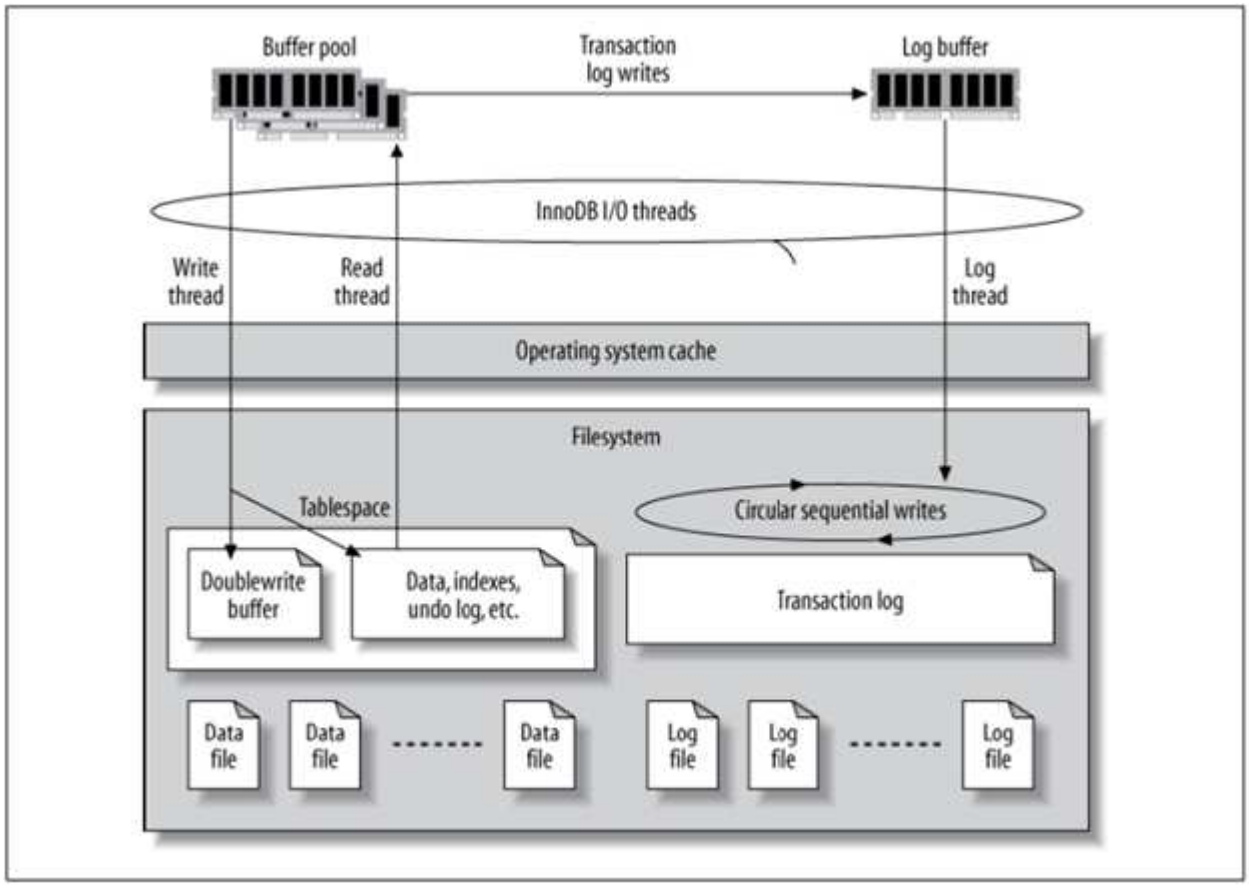
NetApp
February 10, 2026

목차

데이터베이스 구성

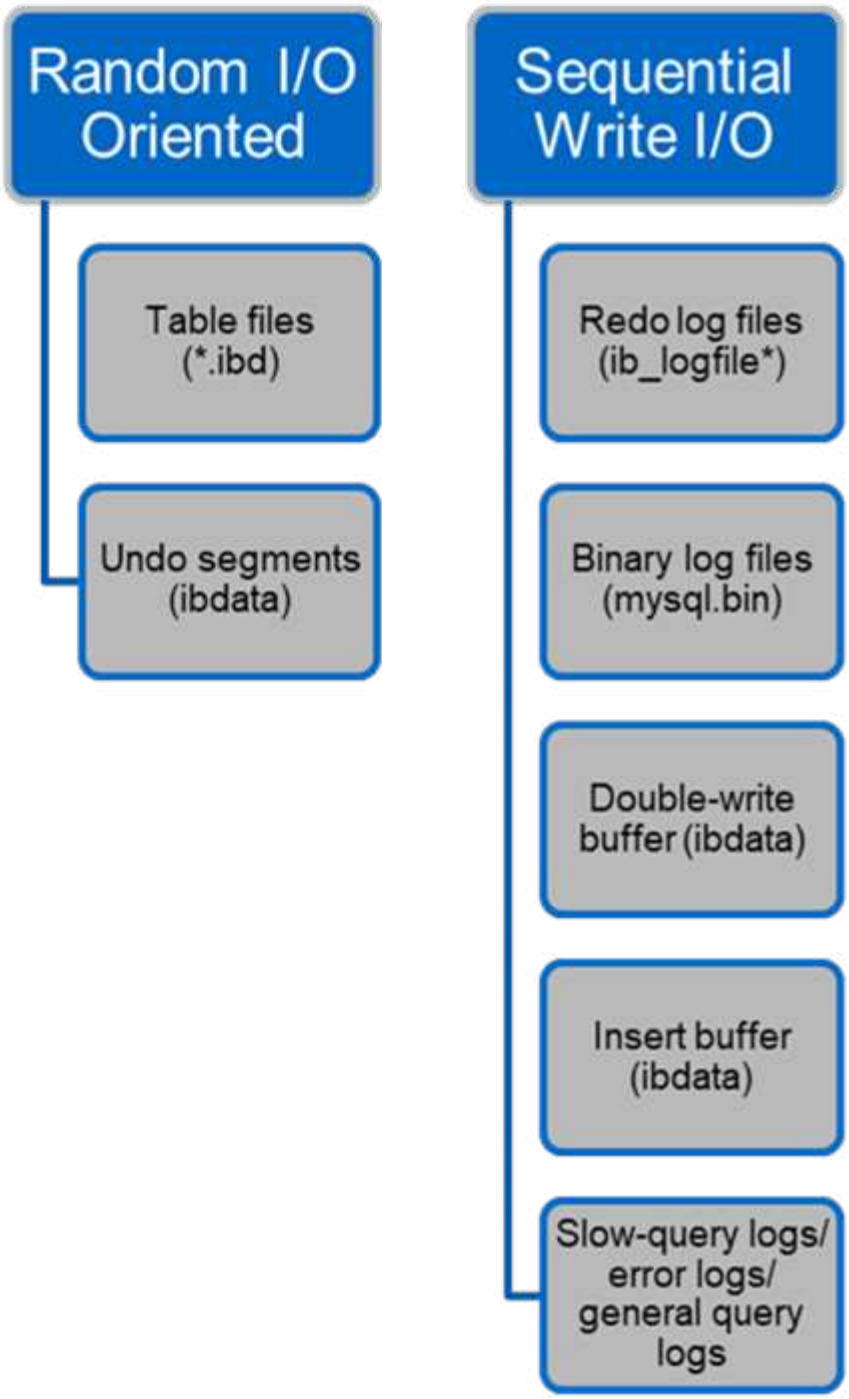
파일 구조

InnoDB는 스토리지와 MySQL 서버 사이의 중간 계층 역할을 하며 데이터를 드라이브에 저장합니다.



MySQL I/O는 두 가지 유형으로 분류됩니다.

- 랜덤 파일 I/O
- 순차적 파일 I/O



데이터 파일을 무작위로 읽고 덮어써서 IOPS가 높아집니다. 따라서 SSD 스토리지를 사용하는 것이 좋습니다.

재실행 로그 파일과 바이너리 로그 파일은 트랜잭션 로그입니다. 순차적으로 작성되므로 쓰기 캐시가 있는 HDD에서 우수한 성능을 얻을 수 있습니다. 순차적 읽기는 복구 시 발생하지만, 일반적으로 로그 파일 크기가 데이터 파일보다 작고 순차 읽기가 랜덤 읽기(데이터 파일에서 발생)보다 빠르므로 성능 문제가 발생하는 경우는 거의 없습니다.

이중 쓰기 버퍼는 InnoDB의 특별한 기능입니다. InnoDB는 먼저 플러시된 페이지를 이중 쓰기 버퍼에 쓴 다음 데이터 파일의 올바른 위치에 페이지를 씁니다. 이 프로세스는 페이지 손상을 방지합니다. 이중 쓰기 버퍼가 없으면 드라이브에 쓰기 프로세스 중에 전원 오류가 발생하면 페이지가 손상될 수 있습니다. 이중 쓰기 버퍼에 쓰는 작업이 순차적이기 때문에 HDD에 맞게 고도로 최적화되어 있습니다. 복구 시 순차적 읽기가 발생합니다.

ONTAP NVRAM은 이미 쓰기 보호를 제공하고 있기 때문에 이중 쓰기 버퍼링이 필요하지 않습니다. MySQL에는 매개 변수가 있습니다. `skip_innodb_doublewrite`이중 쓰기 버퍼를 사용하지 않도록 설정합니다. 이 기능은 성능을 크게 향상시킬 수 있습니다.

INSERT 버퍼는 InnoDB의 특별한 기능이기도 합니다. 고유하지 않은 보조 인덱스 블록이 메모리에 없으면 InnoDB는 임의의 I/O 작업을 방지하기 위해 INSERT 버퍼에 엔트리를 삽입합니다. 주기적으로 삽입 버퍼가 데이터베이스의 보조 인덱스 트리에 병합됩니다. 삽입 버퍼는 I/O 요청을 동일한 블록에 병합하여 I/O 작업 수를 줄입니다. 랜덤 I/O 작업은 순차적일 수 있습니다. 인서트 버퍼는 또한 HDD에 대해 고도로 최적화되어 있습니다. 순차적 쓰기와 읽기는 정상 작업 중에 발생합니다.

실행 취소 세그먼트는 임의 I/O 방향입니다. 다중 버전 동시성(MVCC)을 보장하려면 InnoDB가 실행 취소 세그먼트에 이전 영상을 등록해야 합니다. 실행 취소 세그먼트에서 이전 이미지를 읽으려면 임의 읽기가 필요합니다. 반복 가능한 읽기(예: mysqldump - 단일 트랜잭션)로 긴 트랜잭션을 실행하거나 긴 쿼리를 실행하면 임의 읽기가 발생할 수 있습니다. 따라서 SSD에 실행 취소 세그먼트를 저장하는 것이 더 좋습니다. 짧은 트랜잭션이나 쿼리만 실행할 경우 랜덤 읽기는 문제가 되지 않습니다.

*NetApp는 InnoDB I/O 특성으로 인해 다음과 같은 스토리지 디자인 레이아웃을 권장합니다.

- 단일 볼륨으로 MySQL의 랜덤 및 순차적 I/O 지향 파일을 저장합니다
- MySQL의 순수 순차 I/O 중심 파일을 저장하는 또 다른 볼륨입니다

또한 이 레이아웃은 데이터 보호 정책 및 전략을 설계하는 데 도움이 됩니다.

구성 매개 변수

NetApp은 최적의 성능을 얻기 위해 몇 가지 중요한 MySQL 구성 매개 변수를 권장합니다.

매개 변수	값
InnoDB_log_file_size입니다	256M
InnoDB_flush_log_at_TRx_commit입니다	2
inoDB_doublewrite입니다	0
InnoDB_flush_method 를 참조하십시오	fsync를 참조하십시오
InnoDB_buffer_pool_size입니다	11g
InnoDB_IO_capacity의 약어입니다	8192
InnoDB_buffer_pool_instances입니다	8
InnoDB_LRU_scan_depth 를 선택합니다	8192
open_file_limit를 선택합니다	65535

이 섹션에 설명된 매개 변수를 설정하려면 MySQL 구성 파일(my.cnf)에서 변경해야 합니다. NetApp 모범 사례는 사내에서 수행된 테스트의 결과입니다.

InnoDB_log_file_size입니다

InnoDB 로그 파일 크기에 적합한 크기를 선택하는 것은 쓰기 작업과 서버 충돌 후 적절한 복구

시간을 갖는 데 중요합니다.

많은 트랜잭션이 파일에 로그인되기 때문에 로그 파일 크기는 쓰기 작업에 중요합니다. 레코드가 수정되면 변경 내용이 테이블스페이스에 즉시 다시 기록되지 않습니다. 대신 변경 내용이 로그 파일 끝에 기록되고 페이지가 더티(dirty)로 표시됩니다. InnoDB는 로그를 사용하여 랜덤 I/O를 순차적 I/O로 변환합니다

로그가 가득 차면 로그 파일의 공간을 확보하기 위해 더티 페이지가 테이블스페이스에 순서대로 기록됩니다. 예를 들어 트랜잭션 중에 서버가 충돌하고 쓰기 작업이 로그 파일에만 기록된다고 가정합니다. 서버가 다시 가동되기 전에 로그 파일에 기록된 변경 내용이 재생되는 복구 단계를 거쳐야 합니다. 로그 파일에 있는 항목이 많을수록 서버가 복구하는 데 더 오래 걸립니다.

이 예에서 로그 파일 크기는 복구 시간과 쓰기 성능에 모두 영향을 줍니다. 로그 파일 크기에 적합한 숫자를 선택할 때는 복구 시간과 쓰기 성능의 균형을 맞춥니다. 일반적으로 128M과 512M 사이의 모든 것이 좋은 가치가 있습니다.

InnoDB_flush_log_at_TRx_commit입니다

데이터에 변경 사항이 있을 때 변경 사항이 스토리지에 즉시 기록되지 않습니다.

대신 로그 버퍼에 데이터가 기록됩니다. 로그 버퍼는 InnoDB가 로그 파일에 기록된 버퍼 변경 사항에 할당하는 메모리의 일부입니다. InnoDB는 트랜잭션이 커밋될 때, 버퍼가 가득 찰 때 또는 초당 한 번씩 이벤트가 먼저 발생하는 경우 버퍼를 로그 파일로 플러시합니다. 이 프로세스를 제어하는 구성 변수는 innodb_flush_log_at_TRx_commit입니다. 값 옵션은 다음과 같습니다.

- 를 설정합니다 `innodb_flush_log_trx_at_commit=0`, InnoDB는 InnoDB 버퍼 풀에 있는 수정된 데이터를 로그 파일(IB_LOGFILE)에 쓰고 로그 파일(스토리지에 쓰기)을 매초마다 플러시합니다. 그러나 트랜잭션이 커밋되면 아무 작업도 수행하지 않습니다. 전원 장애나 시스템 충돌이 발생한 경우 플러시되지 않은 데이터는 로그 파일에 기록되지 않으므로 복구할 수 없습니다.
- 를 설정합니다 `innodb_flush_log_trx_commit=1`, InnoDB 는 트랜잭션 로그에 로그 버퍼를 쓰고 모든 트랜잭션에 대해 내구성 있는 저장소로 플러시합니다. 예를 들어, 모든 트랜잭션 커밋에 대해 InnoDB는 로그에 쓴 다음 스토리지에 씁니다. 스토리지 속도가 느리면 성능에 부정적인 영향을 줍니다. 예를 들어 초당 InnoDB 트랜잭션 수가 줄어듭니다.
- 를 설정합니다 `innodb_flush_log_trx_commit=2`InnoDB 는 모든 커밋에서 로그 파일에 로그 버퍼를 쓰지만 스토리지에 데이터를 쓰지 않습니다. InnoDB는 1초에 한 번씩 데이터를 플러시합니다. 전원 장애 또는 시스템 충돌이 발생하더라도 로그 파일에서 옵션 2 데이터를 사용할 수 있으며 복구할 수 있습니다.

성과가 주요 목표인 경우 값을 2로 설정합니다. InnoDB는 모든 트랜잭션 커밋이 아니라 1초에 한 번씩 드라이브에 쓰기 때문에 성능이 크게 향상됩니다. 전원 장애 또는 충돌이 발생하면 트랜잭션 로그에서 데이터를 복구할 수 있습니다.

데이터 안전이 주요 목표인 경우 값을 1로 설정하여 모든 트랜잭션 커밋에 대해 InnoDB가 드라이브로 플러시합니다. 그러나 성능에 영향을 줄 수 있습니다.

* NetApp는 성능 향상을 위해 nodb_flush_log_TRx_commit 값을 2로 설정할 것을 권장합니다.

inoDB_doublewrite입니다

시기 `innodb_doublewrite` 이(기본값)을 사용하도록 설정하면 InnoDB는 모든 데이터를 두 번 저장합니다. 먼저 이중 쓰기 버퍼에 데이터를 저장한 다음 실제 데이터 파일에 저장합니다.

을 사용하여 이 매개 변수를 끌 수 있습니다 `--skip-innodb_doublewrite` 벤치마크용 또는 데이터 무결성 또는

가능한 오류보다 최고 성능에 더 관심이 있는 경우. InnoDB는 double-write라는 파일 플러시 기술을 사용합니다. InnoDB는 데이터 파일에 페이지를 쓰기 전에 이중 쓰기 버퍼라는 인접 영역에 페이지를 씁니다. 이중 쓰기 버퍼에 대한 쓰기 및 플러시가 완료된 후 InnoDB는 데이터 파일의 적절한 위치에 페이지를 씁니다. 페이지 쓰기 중에 운영 체제 또는 mysqld 프로세스가 충돌하는 경우 InnoDB는 나중에 충돌 복구 중에 이중 쓰기 버퍼에서 올바른 페이지 복사본을 찾을 수 있습니다.

* NetApp는 이중 쓰기 버퍼를 비활성화 * 할 것을 권장합니다. ONTAP NVRAM은 동일한 기능을 제공한다. 이중 버퍼링은 불필요하게 성능을 저하시킵니다.

InnoDB_buffer_pool_size입니다

InnoDB 버퍼 풀은 모든 튜닝 작업에서 가장 중요한 부분입니다.

InnoDB는 인덱스 캐싱 및 데이터 조정, 적응형 해시 인덱스, 삽입 버퍼 및 내부적으로 사용되는 기타 많은 데이터 구조를 위해 버퍼 풀에 크게 의존합니다. 버퍼 풀은 또한 쓰기 작업을 스토리지에 즉시 수행할 필요가 없도록 데이터의 변경 사항을 버퍼링하여 성능을 개선합니다. 버퍼 풀은 InnoDB의 필수 부분이며 그에 따라 크기를 조정해야 합니다. 버퍼 풀 크기를 설정할 때는 다음 요소를 고려하십시오.

- 전용 InnoDB 전용 시스템의 경우 버퍼 풀 크기를 사용 가능한 RAM의 80% 이상으로 설정합니다.
- MySQL 전용 서버가 아닌 경우 크기를 RAM의 50%로 설정합니다.

InnoDB_flush_method 를 참조하십시오

innodb_flush_method 매개 변수는 InnoDB가 로그 및 데이터 파일을 열고 플러시하는 방법을 지정합니다.

최적화

InnoDB 최적화에서 이 매개 변수를 설정하면 해당되는 경우 데이터베이스 성능이 조정됩니다.

다음 옵션은 InnoDB를 통해 파일을 플러시하는 것입니다.

- `fsync`. InnoDB는 를 사용합니다 `fsync()` 데이터 및 로그 파일을 모두 플러시하기 위한 시스템 호출입니다. 이 옵션이 기본 설정입니다.
- `O_DSYNC`. InnoDB는 를 사용합니다 `O_DSYNC` 로그 파일을 열고 플러시하는 옵션과 데이터 파일을 플러시하기 위한 fsync() 옵션을 선택합니다. InnoDB는 사용하지 않습니다 `O_DSYNC` 유닉스의 많은 변종에서 그것에 문제가 있기 때문에 직접.
- `O_DIRECT`. InnoDB는 를 사용합니다 `O_DIRECT` 옵션(또는 `directio()` Solaris의 경우)를 사용하여 데이터 파일을 열고 사용합니다 `fsync()` 데이터 및 로그 파일을 모두 플러시합니다. 이 옵션은 일부 GNU/Linux 버전, FreeBSD 및 Solaris에서 사용할 수 있습니다.
- `O_DIRECT_NO_FSYNC`. InnoDB는 를 사용합니다 `O_DIRECT` 입출력 플러싱 중에 옵션을 사용할 수 있지만 이 옵션은 를 건너뜁니다 `fsync()` 이후 시스템 호출 이 옵션은 일부 파일 시스템 유형(예: XFS)에는 적합하지 않습니다. 파일 시스템에 가 필요한지 확실하지 않은 경우 `fsync()` 예를 들어 모든 파일 메타데이터를 보존하려면 시스템 호출을 사용합니다 `O_DIRECT` 옵션을 선택합니다.

관찰

NetApp 랩 테스트에서 는 를 선택합니다 `fsync` 기본 옵션은 NFS 및 SAN에서 사용되었으며 에 비해 성능이 탁월했습니다 `O_DIRECT`. 으로 플러시 방법을 사용하는 동안 `O_DIRECT` ONTAP에서는 클라이언트가 4096 블록의 테두리에서 많은 싱글바이트 쓰기를 직렬 방식으로 기록하는 것을 확인했습니다. 이러한 쓰기는 네트워크에서 지연 시간이 증가하고 성능이 저하됩니다.

InnoDB_IO_capacity의 약어입니다

InnoDB 플러그인에서는 MySQL 5.7에서 innodb_io_capacity라는 새 매개 변수가 추가되었습니다.

InnoDB가 수행하는 최대 IOPS 수를 제어합니다(더티 페이지의 플러싱 비율과 삽입 버퍼[ibuf] 배치 크기 포함). innodb_io_capacity 매개 변수는 버퍼 풀에서 페이지를 플러시하거나 변경 버퍼에서 데이터를 병합하는 등 InnoDB 백그라운드 작업에 의한 IOPS의 상한을 설정합니다.

innodb_io_capacity 매개 변수를 시스템이 초당 수행할 수 있는 대략적인 입출력 작업 수로 설정합니다. 가장 좋은 방법은 설정을 가능한 낮게 유지하는 것이지만, 너무 낮게 설정하면 배경 활동이 느려지는 것이 좋습니다. 설정이 너무 높으면 버퍼 풀에서 데이터가 제거되고 캐싱을 위해 너무 빨리 버퍼를 삽입하여 상당한 이점을 얻을 수 있습니다.

* NetApp는 NFS에서 이 설정을 사용할 경우 IOPS(Sysbench/FiO)의 테스트 결과를 분석하고 그에 따라 매개변수를 설정할 것을 권장합니다. InnoDB 버퍼 풀에서 원하는 것보다 더 많은 수정 또는 더티 페이지가 표시되지 않는 한 플러싱과 퍼징에 가능한 가장 작은 값을 사용합니다.

작업량에 더 낮은 값이 충분하지 않다는 사실이 입증되지 않는 한 20,000개 이상의 극단적인 값을 사용하지 마십시오.

InnoDB_IO_CAPACITY 매개변수는 플러싱 속도 및 관련 I/O를 조절합니다

이 매개 변수 또는 innodb_io_capacity_max 매개 변수를 너무 높게 설정하여 성능을 심각하게 저하시킬 수 있습니다

InnoDB_LRU_scan_depth 를 선택합니다

를 클릭합니다 `innodb_lru_scan_depth` 매개 변수는 InnoDB 버퍼 풀에 대한 플러시 작업의 알고리즘 및 휴리스틱에 영향을 줍니다.

이 매개 변수는 I/O 집약적인 워크로드를 성능 전문가에 주로 적용됩니다. 각 버퍼 풀 인스턴스에 대해 이 매개 변수는 LRU(Least Recently Used) 페이지 목록에서 페이지 클리너 스레드가 계속 스캔해야 하는 범위를 지정하며 플러시할 더티 페이지를 찾습니다. 이 백그라운드 작업은 초당 한 번 수행됩니다.

값을 위 또는 아래로 조정하여 사용 가능한 페이지 수를 최소화할 수 있습니다. 스캔에 상당한 성능 비용이 들 수 있으므로 값을 필요 이상으로 설정하지 마십시오. 또한 버퍼 풀 인스턴스의 수를 변경할 때 이 매개 변수를 조정하는 것도 고려하십시오. 그 이유는 무엇입니까 `innodb_lru_scan_depth * innodb_buffer_pool_instances` 페이지 클리너 스레드에서 초당 수행하는 작업 양을 정의합니다.

기본값보다 작은 설정은 대부분의 워크로드에 적합합니다. 일반적인 작업 부하에서 여유 I/O 용량이 있는 경우에만 이 값을 높이는 것이 좋습니다. 반면, 쓰기 집약적 워크로드에서 I/O 용량이 포화되면 특히 버퍼 풀이 있는 경우 값을 줄이십시오.

Open_file_limits 를 참조하십시오

를 클릭합니다 `open_file_limits` 매개 변수는 운영 체제에서 mysqld를 열도록 허용하는 파일 수를 결정합니다.

런타임에 이 매개 변수의 값은 시스템에서 허용하는 실제 값이며 서버 시작 시 지정한 값과 다를 수 있습니다. MySQL이 열려 있는 파일 수를 변경할 수 없는 시스템의 값은 0입니다. 효과 `open_files_limit` 값은 시스템 시작 시 지정된 값(있는 경우)과 의 값을 기반으로 합니다 `max_connections` 및 `table_open_cache` 다음 수식을 사용하여 다음을 실행합니다.

- 10 이상 `max_connections` 를 누릅니다 (`table_open_cache` x 2)
- `max_connections` x 5
- 양수인 경우 운영 체제 제한
- 운영 체제 제한이 무한대인 경우: `open_files_limit` 시작 시 값이 지정되고 없는 경우 5,000입니다

서버는 이 네 개의 최대 값을 사용하여 파일 설명자의 수를 가져오려고 시도합니다. 이렇게 많은 설명자를 얻을 수 없는 경우 서버는 시스템에서 허용하는 수만큼 얻기를 시도합니다.

저작권 정보

상표 정보