```
쿼리 완료, 영향을 받은 행 1671168개(1분 35.54초)
```

큰 테이블에서 DDL 작업을 실행하기 전에 다음과 같이 작업이 빠른지 또는 느린지 확인하십시오:

1. 테이블 구조를 복제합니다.
2. 복제된 테이블을 소량의 데이터로 채웁니다.
3. 복제된 테이블에서 DDL 작업을 실행합니다.
4. "영향을 받는 행" 값이 0인지 여부를 확인합니다. 0이 아닌 값은 작업이 테이블 데이터를 복사한다는 의미이므로 특별한 계획이 필요할 수 있습니다. 예를 들어, 예정된 다운타임 기간 동안 또는 각 복제본 서버에서 한 번에 하나씩 DDL 작업을 수행할 수 있습니다.

참고

DDL 작업과 관련된 MySQL 처리를 더 잘 이해하려면 DDL 작업 전후에 `InnoDB`와 관련된 성능 스키마 및 정보 스키마 테이블을 검토하여 물리적 읽기, 쓰기, 메모리 할당 횟수 등을 확인하세요.

성능 스키마 단계 이벤트를 사용하여 테이블 변경 진행 상황을 모니터링할 수 있습니다. 15.16.1절, "성능 스키마를 사용하여 InnoDB 테이블에 대한 ALTER TABLE 진행 상황 모니터링"을 참조하십시오.

동시 DDL 작업으로 인한 변경 사항을 기록한 다음 마지막에 해당 변경 사항을 적용하는 데 일부 처리 작업이 필요하기 때문에 온라인 DDL 작업은 다른 세션에서 테이블 액세스를 차단하는 테이블 복사 메커니즘보다 전체적으로 시간이 더 오래 걸릴 수 있습니다. 원시 성능의 감소는 테이블을 사용하는 애플리케이션의 응답성 향상과 균형을 이룹니다. 테이블 구조를 변경하는 기술을 평가할 때는 웹 페이지 로드 시간과 같은 요소를 기반으로 최종 사용자의 성능 인식을 고려해야 합니다.

15.12.3 온라인 DDL 공간 요구 사항

온라인 DDL 작업에 대한 디스크 공간 요구 사항은 아래에 설명되어 있습니다. 이 요구 사항은 즉시 수행되는 작업에는 적용되지 않습니다.

- 임시 로그 파일:

  임시 로그 파일은 온라인 DDL 작업으로 인덱스가 생성되거나 테이블이 변경될 때 동시 DDL을 기록합니다. 임시 로그 파일은 `innodb_sort_buffer_size` 값에 의해 지정된 최대값까지 필요에 따라 확장됩니다. `innodb_online_alter_log_max_size`. 작업 시간이 오래 걸리고 동시 DML이 테이블을 너무 많이 수정하여 임시 로그 파일의 크기가 다음 값을 초과하는 경우를 초과하면 온라인 DDL 작업이 `DB_ONLINE_LOG_TOO_BIG` 오류와 함께 실패하고 커밋되지 않은 동시

DML 작업이 롤백됩니다. `innodb_online_alter_log_max_size`를 크게 설정하면 온라인 DDL 작업 중에 더 많은 DML을 허용하지만, DDL 작업이 끝날 때 테이블이 잠겨 기록된 DML을 적용하는 기간이 길어집니다.

`innodb_sort_buffer_size` 변수는 임시 로그 파일 읽기 버퍼와 쓰기 버퍼의 크기도 정의합니다.

- 임시 정렬 파일:

  테이블을 재구성하는 온라인 DDL 작업은 인덱스 생성 중에 임시 정렬 파일을 MySQL 임시 디렉터리(Unix의 경우 `$TMPDIR`, Windows의 경우 `%TEMP%` 또는 `--tmpdir`로 지정한 디렉터리)에 씁니다. 임시 정렬 파일은 원본 테이블이 포함된 디렉터리에는 생성되지 않습니다. 각 임시 정렬 파일은 데이터 한 열을 저장할 수 있을 만큼 충분히 크며, 각 정렬 파일은 데이터가 최종 테이블 또는 인덱스에 병합될 때 제거됩니다. 임시 정렬 파일과 관련된 작업은 다음과 같습니다.

테이블의 데이터 양에 인덱스를 더한 만큼의 임시 공간이 필요합니다. 온라인 DDL 작업이 데이터 디렉터리가 있는 파일 시스템에서 사용 가능한 모든 디스크 공간을 사용하는 경우 오류가 보고됩니다.

MySQL 임시 디렉터리가 정렬 파일을 저장할 수 있을 만큼 충분히 크지 않은 경우 `tmpdir`을 다른 디렉터리로 설정합니다. 또는 `innodb_tmpdir`을 사용하여 온라인 DDL 작업을 위한 별도의 임시 디렉터리를 정의합니다. 이 옵션은 대용량 임시 정렬 파일로 인해 발생할 수 있는 임시 디렉터리 오버플로를 방지하기 위해 도입되었습니다.

- 중간 테이블 파일:

  테이블을 재구성하는 일부 온라인 DDL 작업은 원본 테이블과 동일한 디렉터리에 임시 중간 테이블 파일을 생성합니다. 중간 테이블 파일에는 원본 테이블 크기와 동일한 공간이 필요할 수 있습니다. 중간 테이블 파일 이름은 `#sql-ib` 접두사로 시작하며 온라인 DDL 작업 중에만 잠깐 나타납니다.

  `innodb_tmpdir` 옵션은 중간 테이블 파일에는 적용되지 않습니다.

15.12.4 온라인 DDL 메모리 관리

보조 인덱스를 생성하거나 재구축하는 온라인 DDL 작업은 인덱스 생성의 여러 단계 동안 임시 버퍼를 할당합니다. `innodb_ddl_buffer_size` 변수는 온라인 DDL 작업의 최대 버퍼 크기를 정의합니다. 기본 설정은 1048576바이트(1MB)입니다. 이 설정은 온라인 DDL 작업을 실행하는 스레드에 의해 생성된 버퍼에 적용됩니다. 적절한 버퍼 크기 제한을 정의하면 보조 인덱스를 만들거나 재구성하는 온라인 DDL 작업에서 발생할 수 있는 메모리 부족 오류를 방지할 수 있습니다. DDL 스레드당 최대 버퍼 크기는 최대 버퍼 크기를 DDL 스레드 수로 나눈 값입니다(`innodb_ddl_buffer_size/innodb_ddl_threads`).

15.12.5 온라인 DDL 작업을 위한 병렬 스레드 구성

보조 인덱스를 생성하거나 재구축하는 온라인 DDL 작업의 워크플로에는 다음이 포함됩니다:

- 클러스터된 인덱스 스캔 및 임시 정렬 파일에 데이터 쓰기
- 데이터 정렬
- 임시 정렬 파일에서 정렬된 데이터를 보조 인덱스로 로드하기

클러스터된 인덱스를 스캔하는 데 사용할 수 있는 병렬 스레드 수는 `innodb_parallel_read_threads` 변수에 의해 정의됩니다. 기본 설정은 4입니다. 최대 설정은 모든 세션의 최대 수인 256입니다. 클러스터된 인덱스를 스캔하는 실제 스레드 수는 `innodb_parallel_read_threads` 설정에 정의된 수 또는 스캔할 인덱스 하위 트리 수 중 더 작은 수입니다. 스레드 제한에 도달하면 세션은 단일 스레드 사용으로 되돌아갑니다.

데이터를 정렬하고 로드하는 병렬 스레드 수는 `innodb_ddl_threads`에 의해 제어됩니다.
변수를 설정합니다. 기본 설정은 4

입니다. 다음과 같은 제한 사항이 적

용됩니다:

- 가상 열을 포함하는 인덱스 구축에는 병렬 스레드가 지원되지 않습니다.
- 병렬 스레드는 전체 텍스트 인덱스 생성에 지원되지 않습니다.
- 공간 인덱스 생성에는 병렬 스레드가 지원되지 않습니다.
- 가상 열로 정의된 테이블에서는 병렬 스캔이 지원되지 않습니다.
- 전체 텍스트 인덱스로 정의된 테이블에서는 병렬 스캔이 지원되지 않습니다.
- 공간 인덱스로 정의된 테이블에서는 병렬 스캔이 지원되지 않습니다.

15.12.6 온라인 DDL로 DDL 문 간소화하기

온라인 DDL이 도입되기 전에는 많은 DDL 작업을 하나의 `ALTER TABLE` 문으로 결합하는 것이 일반적이었습니다. 각 `ALTER TABLE` 문은 테이블을 복사하고 다시 작성해야 했기 때문에 동일한 테이블을 한 번에 여러 번 변경하는 것이 더 효율적이었는데, 테이블에 대한 한 번의 재작성 작업으로 모든 변경을 수행할 수 있었기 때문입니다. 단점은 DDL 작업이 포함된 SQL 코드를 유지 관리하고 다른 스크립트에서 재사용하기가 더 어렵다는 것이었습니다. 특정 변경 사항이 매번 다른 경우, 약간씩 다른 시나리오마다 복잡한 `ALTER TABLE`을 새로 작성해야 할 수도 있습니다.

온라인에서 수행할 수 있는 DDL 작업의 경우 효율성을 유지하면서 스크립팅 및 유지 관리를 쉽게 하기 위해 개별 `ALTER TABLE` 문으로 분리할 수 있습니다. 예를 들어 다음과 같은 복잡한 문이 있을 수 있습니다:

```
ALTER TABLE t1 ADD INDEX i1(c1), ADD UNIQUE INDEX i2(c2),
  CHANGE c4_old_name c4_new_name INTEGER UNSIGNED;
```

와 같이 독립적으로 테스트하고 수행할 수 있는 더 간단한 부분으로 세분화할 수 있습니다:

```
ALTER TABLE t1 ADD INDEX i1(c1);
ALTER TABLE t1 ADD UNIQUE INDEX i2(c2);
ALTER TABLE t1 CHANGE c4_old_name c4_new_name INTEGER UNSIGNED NOT NULL;
```

여전히 여러 부분으로 구성된 `ALTER TABLE` 문을 사용할 수 있습니다:

- 인덱스 생성 후 해당 인덱스를 사용하는 외래 키 제약 조건과 같이 특정 순서로 수행해야 하는 작업입니다.
- 그룹으로 성공 또는 실패하려는 동일한 특정 `LOCK` 절을 모두 사용하는 작업입니다.
- 온라인에서 수행할 수 없는 작업, 즉 여전히 테이블 복사 방법을 사용하는 작업입니다.
- 특수한 시나리오에서 정확한 이전 버전과의 호환성을 위해 필요한 경우 테이블 복사 동작을 강제로 수행하기 위해 `ALGORITHM=COPY` 또는 `old_alter_table=1`을 지정하는 작업입니다.

15.12.7 온라인 DDL 장애 조건

온라인 DDL 작업의 실패는 일반적으로 다음 조건 중 하나로 인해 발생합니다:

- `ALGORITHM` 절은 특정 유형의 DDL 작업 또는 스토리지 엔진과 호환되지 않는 알고리즘을 지정합니다.
- `LOCK` 절은 특정 유형의 DDL 작업과 호환되지 않는 낮은 수준의 잠금(공유 또는 없음)을 지정합니다.
- 테이블에 대한 독점 잠금을 기다리는 동안 타임아웃이 발생하며, 이는 DDL 작업의 초기 및 최종 단계에서 잠시 필요할 수 있습니다.
- 인덱스를 생성하는 동안 MySQL이 디스크에 임시 정렬 파일을 쓰는 동안 `tmpdir` 또는 `innodb_tmpdir` 파일 시스템의 디스크 공간이 부족합니다. 자세한 내용은 섹션 15.12.3, "온라인 DDL 공간 요구 사항"을 참조하세요.
- 이 작업은 시간이 오래 걸리고 동시 DML이 테이블을 너무 많이 수정하여 임시 온라인 로그의 크기가

`innodb_online_alter_log_max_size` 구성 옵션의 값을 초과합니다. 이 조건에서는 `DB_ONLINE_LOG_TOO_BIG` 오류가 발생합니다.

- 동시 DML은 원래 테이블 정의에서는 허용되지만 새 테이블 정의에서는 허용되지 않는 테이블 변경을 수행합니다. 이 작업은 MySQL이 동시 DML 문의 모든 변경 사항을 적용하려고 할 때 맨 마지막에만 실패합니다. 예를 들어, 중복 값을 다음에 삽입할 수 있습니다.
  를 열에 삽입하거나 해당 열에 기본 키 인덱스를 생성하는 동안 열에 `NULL` 값을 삽입할 수 있습니다. 동시 DML에 의해 수행된 변경 사항이 우선 적용되며 `ALTER TABLE` 작업은 효과적으로 롤백됩니다.

15.12.8 온라인 DDL 제한

온라인 DDL 작업에는 다음과 같은 제한 사항이 적용됩니다:

- 임시 테이블에 인덱스를 만들 때 테이블이 복사됩니다.
- 테이블에 ON...CASCADE 또는 ON...SET NULL 제약 조건이 있는 경우 ALTER TABLE 절 LOCK=NONE은 허용되지 않습니다.
- 인플레이스 온라인 DDL 작업이 완료되기 전에 테이블에 메타데이터 잠금이 있는 트랜잭션이 커밋 또는 롤백될 때까지 기다려야 합니다. 온라인 DDL 작업은 실행 단계에서 테이블에 대한 독점 메타데이터 잠금이 잠시 필요할 수 있으며, 테이블 정의를 업데이트할 때 작업의 마지막 단계에서는 항상 메타데이터 잠금이 필요합니다. 따라서 테이블에 메타데이터 잠금이 있는 트랜잭션은 온라인 DDL 작업을 차단할 수 있습니다. 테이블에 메타데이터 잠금을 보유한 트랜잭션은 온라인 DDL 작업 전 또는 작업 중에 시작되었을 수 있습니다.
  장기 실행
  또는 테이블에 메타데이터 잠금이 있는 비활성 트랜잭션으로 인해 온라인 DDL 작업이 시간 초과될 수 있습니다.
- 인플레이스 온라인 DDL 작업을 실행할 때 ALTER TABLE 문을 실행하는 스레드는 다른 연결 스레드에서 동일한 테이블에서 동시에 실행된 DML 작업의 온라인 로그를 적용합니다. DML 작업이 적용될 때 중복 키 입력 오류(오류 1062 (23000))가 발생할 수 있습니다: 중복 항목)이 발생할 수 있습니다. 중복 항목이 일시적이고 나중에 온라인 로그의 항목에 의해 되돌릴 수 있는 경우에도 마찬가지입니다. 이는 트랜잭션 중에 제약 조건이 유지되어야 하는 InnoDB의 외래 키 제약 조건 검사의 개념과 유사합니다.
- InnoDB 테이블에 대한 OPTIMIZE TABLE은 테이블을 재구성하고 인덱스 통계를 업데이트하고 클러스터된 인덱스에서 사용되지 않는 공간을 확보하기 위해 ALTER TABLE 작업에 매핑됩니다. 보조 인덱스는 기본 키에 표시된 순서대로 키가 삽입되기 때문에 효율적으로 생성되지 않습니다. 일반 테이블 및 분할된 InnoDB 테이블을 재구축하기 위한 온라인 DDL 지원이 추가되어 테이블 최적화가 지원됩니다.
- 임시 열(DATE, DATETIME 또는 TIMESTAMP)을 포함하고 ALGORITHM=COPY를 사용하여 재구축되지 않은 MySQL 5.6 이전에 생성된 테이블은 ALGORITHM=INPLACE를 지원하지 않습니다. 이 경우 ALTER TABLE ... ALGORITHM=INPLACE 연산은 다음과 같은 오류를 반환한다:

```
오류 1846 (0A000): 알고리즘=인플레이스가 지원되지 않습니다.
Reason: 열 유형을 INPLACE로 변경할 수 없습니다. ALGORITHM=COPY를 사용해 보십시오.
```

- 일반적으로 테이블 재구성을 수반하는 대규모 테이블의 온라인 DDL 작업에는 다음과 같은 제한 사항이 적용됩니다:
  - 온라인 DDL 작업을 일시 중지하거나 온라인 DDL 작업에 대한 I/O 또는 CPU 사용량을 조절할 수 있는 메커니즘이 없습니다.
  - 온라인 DDL 작업의 롤백은 작업이 실패할 경우 많은 비용이 발생할 수 있습니다.

- 온라인 DDL 작업을 오래 실행하면 복제 지연이 발생할 수 있습니다. 온라인 DDL 작업은 복제본에서 실행되기 전에 소스에서 실행을 완료해야 합니다. 또한 소스에서 동시에 처리된 DDL은 복제본의 DDL 작업이 완료된 후에만 복제본에서 처리됩니다.

대규모 테이블에서 온라인 DDL 작업을 실행하는 것과 관련된 자세한 내용은 15.12.2절. "온라인 DDL 성능 및 동시성"을 참조하십시오.

15.13 InnoDB 저장 데이터 암호화

InnoDB는 테이블별 파일 테이블 스페이스, 일반 테이블 스페이스, `mysql` 시스템 테이블 스페이스, 다시 실행 로그, 실행 취소 로그.

스키마 및 일반 테이블 스페이스에 대한 암호화 기본값을 설정할 수 있으며, 이를 통해 DBA는 해당 스키마 및 테이블 스페이스에서 생성된 테이블의 암호화 여부를 제어할 수 있습니다.

이 섹션의 다음 항목에서는 InnoDB 미사용 데이터 암호화 기능 및 성능에 대해 설명합니다.

저장 데이터 암호화 정보

InnoDB는 마스터 암호화 키와 테이블스페이스 키로 구성된 2계층 암호화 키 아키텍처를 사용합니다. 테이블스페이스가 암호화되면 테이블스페이스 키가 암호화되어 테이블스페이스 헤더에 저장됩니다. 애플리케이션 또는 인증된 사용자가 암호화된 테이블스페이스 데이터에 액세스하려는 경우 InnoDB는 마스터 암호화 키를 사용하여 테이블스페이스 키의 암호를 해독합니다. 해독된 버전의 테이블스페이스 키는 변경되지 않지만 마스터 암호화 키는 필요에 따라 변경할 수 있습니다. 이 작업을 *마스터 키 로테이션이라고 합니다*.

저장 데이터 암호화 기능은 마스터 암호화 키 관리를 위해 키링 구성 요소 또는 플러그인에 의존합니다.

모든 MySQL 에디션은 server 호스트에 로컬로 있는 파일에 키링 데이터를 저장하는 component_keyring_file 컴포넌트 및 keyring_file 플러그인을 제공합니다.

MySQL Enterprise Edition은 추가적인 키링 구성 요소와 플러그인을 제공합니다:

- `component_keyring_encrypted_file`: 키링 데이터를 서버 호스트에 로컬로 암호화된 비밀번호로 보호된 파일에 저장합니다.
- 키링_암호화된_파일: 키링 데이터를 서버 호스트에 로컬로 암호화된 비밀번호로 보호된 파일에 저장합니다.
- `keyring_okv`: KMIP 호환 백엔드 키링 스토리지 제품과 함께 사용하기 위한 KMIP 1.1 플러그인입니다. 지원되는 KMIP 호환 제품에는 오라클 키 볼트, 젬알토 키시큐어, 탈레스 보메트릭 키 관리 서버, 포네틱스 키 오케스트레이션과 같은 중앙 집중식 키 관리 솔루션이 포함됩니다.

- `keyring_aws`: 키 생성을 위한 백엔드로 Amazon Web Services 키 관리 서비스(AWS KMS)와 통신하고 키 저장을 위해 로컬 파일을 사용합니다.
- 키링_해시콥: 백엔드 스토리지를 위해 해시코프 볼트와 통신합니다.

경고

암호화 키 관리를 위해 구성 요소_키링_파일 및 `component_keyring_encrypted_file` 컴포넌트와 키링 파일 및 키링 암호화 파일 플러그인은 규정 준수 솔루션으로 의도되지 않았습니다. PCI, FIPS 등의 보안 표준에서는 키 관리 시스템을 사용하여 키 볼트 또는 하드웨어 보안 모듈(HSM)에서 암호화 키를 보안, 관리 및 보호하도록 요구합니다.

안전하고 강력한 암호화 키 관리 솔루션은 보안 및 다양한 보안 표준 준수를 위해 매우 중요합니다. 미사용 데이터 암호화 기능에서 중앙 집중식 키 관리 솔루션을 사용하는 경우, 이 기능을 'MySQL Enterprise TDE(투명 데이터 암호화)'라고 합니다.

미사용 데이터 암호화 기능은 고급 암호화 표준(AES) 블록 기반 암호화 알고리즘을 지원합니다. 테이블스페이스 키 암호화에는 ECB(전자 코드북) 블록 암호화 모드를, 데이터 암호화에는 CBC(암호 블록 체인) 블록 암호화 모드를 사용합니다.

저장 데이터 암호화 기능에 대해 자주 묻는 질문은 섹션 A.17, "MySQL 8.2 FAQ: InnoDB 저장 데이터 암호화".

암호화 전제 조건

- 키링 구성 요소 또는 플러그인은 시작 시 설치 및 구성해야 합니다. 조기 로딩은 `InnoDB` 스토리지 엔진을 초기화하기 전에 구성 요소 또는 플러그인을 사용할 수 있도록 합니다. 키링 설치 및 구성 지침은 섹션 6.4.4, "MySQL 키링"을 참조하세요. 이 지침은 선택한 컴포넌트 또는 플러그인이 활성화되어 있는지 확인하는 방법을 보여줍니다.

  한 번에 하나의 키링 컴포넌트 또는 플러그인만 활성화해야 합니다. 여러 개의 키링 컴포넌트 또는 플러그인을 활성화하는 것은 지원되지 않으며 결과가 예상과 다를 수 있습니다.

중요

암호화된 테이블스페이스가 MySQL 인스턴스에 생성되면 암호화된 테이블스페이스를 생성할 때 로드된 키링 구성 요소 또는 플러그인이 시작 시에도 계속 로드되어야 합니다. 그렇게 하지 않으면 서버를 시작할 때와 `InnoDB` 복구 중에 오류가 발생합니다.

- 프로덕션 데이터를 암호화할 때는 마스터 암호화 키의 손실을 방지하기 위한 조치를 취해야 합니다. *마스터 암호화 키를 분실하면 암호화된 테이블스페이스 파일에 저장된 데이터를 복구할 수 없습니다.* 구성 요소_키링_파일 또는 구성 요소_키링_암호화된_파일 구성 요소를 사용하는 경우 또는 키링_파일 또는

keyring_encrypted_file 플러그인을 사용하여 첫 번째 암호화된 테이블스페이스를 생성한 직후, 마스터 키 로테이션 전, 마스터 키 로테이션 후에 키링 데이터 파일의 백업을 만듭니다. 용
각 구성 요소의 구성 파일은 데이터 파일 위치를 나타냅니다. keyring_file_data 구성 옵션은 keyring_file 플러그인의 키링 데이터 파일 위치를 정의합니다. keyring_encrypted_file_data 구성 옵션은 keyring_encrypted_file 플러그인에 대한 키링 데이터 파일 위치를 정의합니다. keyring_okv 또는 keyring_aws 플러그인을 사용하는 경우 필요한 구성을 수행했는지 확인하세요. 자세한 내용은 6.4.4절. "MySQL 키링"을 참조하세요.

스키마 및 일반 테이블 스페이스에 대한 암호화 기본값 정의

default_table_encryption 시스템 변수는 스키마 및 일반 테이블스페이스에 대한 기본 암호화 설정을 정의합니다. ENCRYPTION 절이 명시적으로 지정되지 않은 경우 CREATE TABLESPACE 및 CREATE SCHEMA 작업은 default_table_encryption 설정을 적용합니다.

ALTER SCHEMA 및 ALTER TABLESPACE 작업에는 default_table_encryption 설정이 적용되지 않습니다. 기존 스키마 또는 일반 테이블스페이스의 암호화를 변경하려면 ENCRYPTION 절을 명시적으로 지정해야 합니다.

default_table_encryption 변수는 개별 클라이언트 연결에 대해 설정하거나 SET 구문을 사용하여 전역적으로 설정할 수 있습니다. 예를 들어, 다음 문은 기본 스키마 및 테이블 공간 암호화를 전역적으로 사용하도록 설정합니다:

```
mysql> SET GLOBAL default_table_encryption=ON;
```

스키마의 기본 암호화 설정은 기본 암호화를 사용하여 정의할 수도 있습니다.
절을 사용하여 스키마를 생성하거나 변경할 수 있습니다:

```
mysql> CREATE SCHEMA test 기본 암호화 = 'Y';
```

스키마를 생성할 때 DEFAULT ENCRYPTION 절을 지정하지 않으면 default_table_encryption 설정이 적용됩니다. 기존 스키마의 기본 암호화를 변경하려면 DEFAULT ENCRYPTION 절을 지정해야 합니다. 그렇지 않으면 스키마는 현재 암호화 설정을 유지합니다.

기본적으로 테이블은 테이블이 생성된 스키마 또는 일반 테이블 스페이스의 암호화 설정을 상속합니다. 예를 들어 암호화 사용 스키마에서 생성된 테이블은 기본적으로 암호화됩니다. 이 동작을 통해 DBA는 스키마 및 일반 테이블 공간 암호화 기본값을 정의하고 적용하여 테이블 암호화 사용을 제어할 수 있습니다.

암호화 기본값은 table_encryption_privilege_check 시스템 변수를 활성화하여 적용합니다. table_encryption_privilege_check를 활성화하면 기본_table_encryption 설정과 다른 암호화 설정으로 스키마 또는 일반 테이블 공간을 만들거나 변경할 때 또는 기본 스키마 암호화와 다른 암호화 설정으로 테이블을 만들거나 변경할 때 권한 확인이 수행됩니다. 기본값인 table_encryption_privilege_check를 비활성화하면 권한 확인이 수행되지 않으며 앞서 언급한 작업을 경고와 함께 진행할 수 있습니다.

table_encryption_privilege_check가 활성화된 경우 기본 암호화 설정을 재정의하려면 table_encryption_admin 권한이 필요합니다. DBA는 이 권한을 부여하여 사용자가 스키마 또는 일반 테이블 스페이스를 만들거나 변경할 때 기본값인 default_table_encryption 설정에서 벗어나거나 테이블을 만들거나 변경할 때 기본 스키마 암호화에서 벗어나도록 할 수 있습니다. 이 권한은 테이블을 만들거나 변경할 때 일반 테이블 공간의 암호화를 벗어나는 것을 허용하지 않습니다. 테이블은 해당 테이블이 있는 일반 테이블 스페이스와 동일한 암호화 설정을 가져야 합니다.

테이블별 파일 테이블 공간 암호화

테이블별 파일 테이블스페이스는 테이블이 생성되는 스키마의 기본 암호화를 상속하며, CREATE TABLE 문에서 ENCRYPTION 절을 명시적으로 지정하지 않는 한 테이블이 생성되는 스키마의 기본 암호화를 상속합니다.

```
mysql> CREATE TABLE t1 (c1 INT) ENCRYPTION = 'Y';
```

기존 파일 단위 테이블 테이블 스페이스의 암호화를 변경하려면 ENCRYPTION 절을 지정해야 합니다.

```
mysql> ALTER TABLE t1 ENCRYPTION = 'Y';
```

TABLE_ENCRYPTION_PRIVIED_CHECK가 활성화되어 있는 경우 기본 스키마 암호화와 다른 설정으로 ENCRYPTION 절을 지정하려면 TABLE_ENCRYPTION_ADMIN 권한이 필요합니다. 스키마 및 일반 테이블스페이스에 대한 암호화 기본값 정의를 참조하십시오.

일반 테이블 공간 암호화

기본_테이블_암호화 변수는 새로 생성된 일반 테이블스페이스의 암호화를 결정하며, CREATE TABLESPACE 문에 ENCRYPTION 절을 명시적으로 지정하지 않는 한 이 변수는 암호화를 결정합니다.

```
mysql> CREATE TABLESPACE `ts1` ADD DATAFILE 'ts1.ibd' ENCRYPTION = 'Y' Engine=InnoDB;
```

기존 일반 테이블스페이스의 암호화를 변경하려면 ENCRYPTION 절을 지정해야 합니다.

```
mysql> ALTER TABLESPACE ts1 ENCRYPTION = 'Y';
```

table_encryption_privilege_check가 활성화된 경우, 기본_table_encryption 설정과 다른 설정으로 ENCRYPTION 절을 지정하려면 TABLE_ENCRYPTION_ADMIN 권한이 필요합니다. 스키마 및 일반 테이블 스페이스에 대한 암호화 기본값 정의를 참조하세요.

이중 쓰기 파일 암호화

MySQL 8.2에서 InnoDB는 암호화된 테이블 스페이스에 속하는 이중 쓰기 파일 페이지를 자동으로 암호화합니다. 별도의 조치가 필요하지 않습니다. 이중 쓰기 파일 페이지는 연결된 테이블 스페이스의 암호화 키를 사용하여 암호화됩니다. 테이블스페이스 데이터 파일에 기록된 동일한 암호화된 페이지가 중복 쓰기 파일에도 기록됩니다. 암호화되지 않은 테이블스페이스에 속하는 이중 쓰기 파일 페이지는 암호화되지 않은 상태로 유지됩니다.

복구하는 동안 암호화된 이중 쓰기 파일 페이지의 암호화가 해제되고 손상 여부를 확인합니다.

mysql 시스템 테이블 공간 암호화

MySQL 시스템 테이블 스페이스에는 MySQL 시스템 데이터베이스 및 MySQL 데이터 사전 테이블이 포함되어 있습니다. 기본적으로 암호화되지 않습니다. mysql 시스템 테이블스페이스에 대한 암호화를 사용하도록 설정하려면 ALTER TABLESPACE 문에 테이블스페이스 이름과 ENCRYPTION 옵션을 지정합니다.

```
mysql> ALTER TABLESPACE mysql ENCRYPTION = 'Y';
```

mysql 시스템 테이블 스페이스에 대한 암호화를 사용하지 않으려면 ALTER TABLESPACE 문을 사용하여 ENCRYPTION = 'N'을 설정합니다.

```
mysql> ALTER TABLESPACE mysql ENCRYPTION = 'N';
```

mysql 시스템 테이블스페이스에 대한 암호화를 사용하거나 사용하지 않도록 설정하려면 CREATE TABLESPACE가 필요합니다.
인스턴스의 모든 테이블에 대한 권한(*.*에 테이블 공간 만들기)이 필요합니다.

로그 암호화 다시 실행

재실행 로그 데이터 암호화는 innodb_redo_log_encrypt 구성 옵션을 사용하여 활성화할 수 있습니다. 재실행 로그 암호화는 기본적으로 비활성화되어 있습니다.

테이블스페이스 데이터와 마찬가지로, 재실행 로그 데이터 암호화는 재실행 로그 데이터가 디스크에 기록될 때 발생하고 복호화는 디스크에서 재실행 로그 데이터를 읽을 때 발생합니다. 재실행 로그 데이터를 메모리로 읽으면 암호화되지 않은 형태가 됩니다. 재실행 로그 데이터는 테이블스페이스 암호화 키를 사용하여 암호화 및 복호화됩니다.

innodb_redo_log_encrypt를 활성화하면 디스크에 있는 암호화되지 않은 재실행 로그 페이지는 암호화되지 않은 상태로 유지되며, 새 재실행 로그 페이지는 암호화된 형태로 디스크에 기록됩니다. 마찬가지로, innodb_redo_log_encrypt가 비활성화되면 디스크에 있는 암호화된 재실행 로그 페이지는 암호화된 상태로 유지되고 새 재실행 로그 페이지는 암호화되지 않은 형태로 디스크에 기록됩니다.

테이블 공간 암호화 키를 포함한 재실행 로그 암호화 메타데이터는 재실행 로그 파일의 헤더에 가장 최근 체크포인트 LSN으로 저장됩니다. 암호화 메타데이터가 포함된 재실행 로그 파일이 제거되면 재실행 로그 암호화가 비활성화됩니다.

재실행 로그 암호화를 활성화하면 키링 구성 요소나 플러그인 없이 또는 암호화 키가 없는 정상적인 재시작은 불가능합니다. InnoDB는 시작 중에 재실행 페이지를 스캔할 수 있어야 하는데, 재실행 로그 페이지가 암호화된 경우 스캔이 불가능하기 때문입니다. 키링 구성 요소 또는 플러그인 또는 암호화 키가 없으면 재실행 로그가 없는 강제 시작(`SRV_FORCE_NO_LOG_REDO`) 만 가능합니다. 섹션 15.21.3, "InnoDB 강제 복구"를 참조하세요.

로그 암호화 실행 취소

실행 취소 로그 데이터 암호화는 `innodb_undo_log_encrypt` 구성 옵션을 사용하여 활성화할 수 있습니다. 실행 취소 로그 암호화는 실행 취소 테이블스페이스에 있는 실행 취소 로그에 적용됩니다. 15.6.3.4절. "테이블스페이스 실행 취소"를 참조하십시오. 실행 취소 로그 데이터 암호화는 기본적으로 비활성화되어 있습니다.

테이블스페이스 데이터와 마찬가지로 실행 취소 로그 데이터 암호화는 실행 취소 로그 데이터가 디스크에 기록될 때 발생하고, 복호화는 실행 취소 로그 데이터가 디스크에서 읽힐 때 발생합니다. 실행 취소 로그 데이터가 메모리로 읽혀지면 암호화되지 않은 형태가 됩니다. 실행 취소 로그 데이터는 테이블스페이스 암호화 키를 사용하여 암호화 및 복호화됩니다.

innodb_undo_log_encrypt를 활성화하면 디스크에 있는 암호화되지 않은 실행 취소 로그 페이지는 암호화되지 않은 상태로 유지되며, 새 실행 취소 로그 페이지는 암호화된 형태로 디스크에 기록됩니다. 마찬가지로 innodb_undo_log_encrypt가비활성화되면 디스크에 있는 암호화된 실행 취소 로그 페이지는 암호화된 상태로 유지되며, 새 실행 취소 로그 페이지는 암호화되지 않은 형태로 디스크에 기록됩니다.

테이블스페이스 암호화 키를 포함한 실행 취소 로그 암호화 메타데이터는 실행 취소 로그 파일의 헤더에 저장됩니다.

참고

실행 취소 로그 암호화를 비활성화하면 서버는 암호화된 실행 취소 로그 데이터가 포함된 실행 취소 테이블스페이스가 잘릴 때까지 실행 취소 로그 데이터를 암호화하는 데 사용된 키링 구성 요소 또는 플러그인을 계속 요구합니다. (암호화 헤더는 실행 취소 테이블스페이스가 잘릴 때만 실행 취소 테이블스페이스에서 제거됩니다.) 실행 취소 테이블스페이스 잘라내기에 대한 자세한 내용은 실행 취소 테이블스페이스 잘라내기를 참조하십시오.

마스터 키 회전

마스터 암호화 키는 주기적으로 그리고 키가 유출되었다고 의심될 때마다 교체해야 합니다.

마스터 키 회전은 원자적인 인스턴스 수준 작업입니다. 마스터 암호화 키가 회전될 때마다 MySQL 인스턴스의 모든 테이블스페이스 키가 다시 암호화되어 각각의 테이블스페이스 헤더에 다시 저장됩니다. 원자적인 작업으로, 회전 작업이 시작되면 모든 테이블스페이스 키에 대해 재암호화가 성공해야 합니다. 서버 장애로 인해 마스터 키 회전이 중단된 경우, 서버 재시작 시 `InnoDB는` 작업을 앞으로 롤백합니다. 자세한 내용은 암호화 및 복구를 참조하세요.

마스터 암호화 키를 회전하면 마스터 암호화 키만 변경되고 테이블스페이스 키가 다시 암호화됩니다. 연결된 테이블스페이스 데이터의 암호를 해독하거나 다시 암호화하지는 않습니다.

마스터 암호화 키를 회전하려면 ENCRYPTION_KEY_ADMIN 권한(또는 더 이상 사용되지 않는 SUPER 권한)이 필요합니다.

마스터 암호화 키를 회전하려면 실행합니다:

```
mysql> ALTER INSTANCE ROTATE INNODB MASTER KEY;
```

ALTER INSTANCE ROTATE INNODB MASTER KEY는 동시 DML을 지원합니다. 그러나 테이블스페이스 암호화 작업과 동시에 실행할 수 없으며, 동시 실행으로 인해 발생할 수 있는 충돌을 방지하기 위해 잠금이 수행됩니다. ALTER INSTANCE ROTATE INNODB MASTER KEY 작업이 실행 중인 경우 테이블스페이스 암호화 작업이 진행되기 전에 이 작업이 완료되어야 하며, 그 반대의 경우도 마찬가지입니다.

암호화 및 복구

암호화 작업 중에 서버 장애가 발생하면 서버가 다시 시작될 때 작업이 롤포워드됩니다. 일반 테이블스페이스의 경우 암호화 작업은 마지막으로 처리된 페이지의 백그라운드 스레드에서 다시 시작됩니다.

마스터 키 로테이션 중에 서버 장애가 발생하면 서버 재시작 시 InnoDB가 작업을 계속합니다.

스토리지 엔진 초기화 전에 키링 구성 요소 또는 플러그인을 로드해야 InnoDB 초기화 및 복구 활동이 테이블스페이스 데이터에 액세스하기 전에 테이블스페이스 헤더에서 테이블스페이스 데이터 페이지의 암호를 해독하는 데 필요한 정보를 검색할 수 있습니다. (암호화 전제 조건 참조).

InnoDB 초기화 및 복구가 시작되면 마스터 키 로테이션 작업이 다시 시작됩니다. 서버 장애로 인해 일부 테이블스페이스 키는 이미 새 마스터 암호화를 사용하여 암호화되어 있을 수 있습니다.
키입니다. InnoDB는 각 테이블스페이스 헤더에서 암호화 데이터를 읽고, 데이터에 테이블스페이스 키가 이전 마스터 암호화 키를 사용하여 암호화된 것으로 표시되면, InnoDB는 키링에서 이전 키를 검색하여 테이블스페이스 키의 암호를 해독하는 데 사용합니다. 그런 다음 InnoDB는 새 마스터 암호화 키를 사용하여 테이블스페이스 키를 다시 암호화하고 다시 암호화된 테이블스페이스 키를 테이블스페이스 헤더에 다시 저장합니다.

암호화된 테이블 스페이스 내보내기

테이블 스페이스 내보내기는 파일 단위 테이블 스페이스에 대해서만 지원됩니다.

암호화된 테이블스페이스를 내보내면 InnoDB는 테이블스페이스 키를 암호화하는 데 사용되는 *전송 키*를 생성합니다. 암호화된 테이블스페이스 키와 전송 키는 *tablespace_name*.cfp 파일에 저장됩니다. 이 파일은 암호화된 테이블스페이스 파일과 함께 다음을 수행하기 위해 필요합니다.
가져오기 작업을 수행합니다. 가져오기 시 InnoDB는 전송 키를 사용하여 *tablespace_name*.cfp 파일에서 테이블 스페이스 키를 해독합니다. 관련 정보는 15.6.1.3절. "InnoDB 테이블 가져오기"를 참조하세요.

암호화 및 복제

- ALTER INSTANCE ROTATE INNODB MASTER KEY 문은 원본 및 복제본이 테이블 공간 암호화를 지원하는 MySQL 버전을 실행하는 복제 환경에서만 지원됩니다.
- 성공적인 ALTER INSTANCE ROTATE INNODB MASTER KEY 문은 복제본의 복제를 위해 바이너리 로그에 기록됩니다.
- ALTER INSTANCE ROTATE INNODB MASTER KEY 문이 실패하면 바이너리 로그에 기록되지 않으며 복제본에 복제되지 않습니다.
- 키링 구성 요소 또는 플러그인이 원본에는 설치되어 있지만 복제본에는 설치되어 있지 않은 경우 ALTER INSTANCE ROTATE INNODB MASTER KEY 작업의 복제가 실패합니다.
- 원본과 복제본 모두에 키링 파일 또는 키링 암호화된 파일 플러그인이 설치되어 있지만 복제본에 키링 데이터 파일이 없는 경우, 키링 파일 데이터가 메모리에 캐시되어 있지 않다고 가정하고 복제된 ALTER INSANCE ROTATE INNODB MASTER KEY 문이 복제본에 키링 데이터 파일을 생성합니다. ALTER INSTANCE ROTATE INNODB MASTER KEY는 사용 가능한 경우 메모리에 캐시된 키링 파일 데이터를 사용합니다.

암호화된 테이블 스페이스 및 스키마 식별하기

정보 스키마 `INNODB_TABLESPACES` 테이블에는 암호화된 테이블 스페이스를 식별하는 데 사용할 수 있는 `ENCRYPTION` 열이 포함되어 있습니다.

```
mysql> SELECT SPACE, NAME, SPACE_TYPE, ENCRYPTION FROM INFORMATION_SCHEMA.INNODB_TABLESPACES
       WHERE ENCRYPTION='Y'\G
*************************** 1. 행 *************************** 공
     간: 4294967294
      NAME: mysql
SPACE_TYPE: 일반 암호화: Y
*************************** 2. 행 ***************************
```

```
     공간: 2
      NAME: test/t1
SPACE_TYPE: 단일 암호화: Y
*************************** 3. 행 ***************************
     SPACE: 3
      이름: ts1
SPACE_TYPE: 일반 암호화: Y
```

CREATE TABLE 또는 ALTER TABLE 문에 ENCRYPTION 옵션이 지정되면 INFORMATION_SCHEMA.TABLES의 CREATE_OPTIONS 열에 기록됩니다. 이 열을 쿼리하여 암호화된 파일 단위 테이블 공간에 있는 테이블을 식별할 수 있습니다.

```
mysql> SELECT TABLE_SCHEMA, TABLE_NAME, CREATE_OPTIONS FROM INFORMATION_SCHEMA.TABLES
       WHERE CREATE_OPTIONS LIKE '%ENCRYPTION%';
+--------------+------------+--------------------+
| 테이블 스키마 | 테이블 이름 | create_옵션 |
+--------------+------------+--------------------+
| 테스트| t1|    암호화="Y" |
+--------------+------------+--------------------+
```

특정 스키마 및 테이블과 연결된 테이블 스페이스에 대한 정보를 검색하려면 정보 스키마 INNODB_TABLESPACES 테이블을 쿼리합니다.

```
mysql> SELECT SPACE, NAME, SPACE_TYPE FROM INFORMATION_SCHEMA.INNODB_TABLESPACES WHERE NAME='test/t1';
+-------+---------+ --------------+
| 공간 | 이름 |     공간 유형 |
+-------+---------+ --------------+
      |3 | test/t1 | 싱글        |
+-------+---------+ --------------+
```

정보 스키마 SCHEMATA 테이블을 쿼리하여 암호화 사용 스키마를 식별할 수 있습니다.

```
mysql> SELECT SCHEMA_NAME, DEFAULT_ENCRYPTION FROM INFORMATION_SCHEMA.SCHEMATA
       WHERE DEFAULT_ENCRYPTION='YES';
+-------------+ ---------------------+
| 스키마_이름 | 기본_암호화 |
+-------------+ ---------------------+
| 테스트| 예     |
+-------------+ ---------------------+
```

생성 스키마 표시에는 기본 암호화 절도 표시됩니다.

암호화 진행 상황 모니터링

성능 스키마를 사용하여 일반 테이블스페이스 및 mysql 시스템 테이블스페이스 암호화 진행 상황을 모니터링할 수 있습니다.

스테이지/이노드/테이블스페이스 변경(암호화) 스테이지 이벤트 계측기 보고서 일반 테이블스페이스 암호화 작업에 대한 WORK_ESTIMATED 및 WORK_COMPLETED 정보입니다.

다음 예에서는 일반 테이블 스페이스 또는 mysql 시스템 테이블 스페이스 암호화 진행률을 모니터링하기 위해 stage/innodb/alter 테이블 스페이스(암호화) 스테이지 이벤트 계측기 및 관련 소비자 테이블을 활성화하는 방법을 설명합니다. 성능 스키마 스테이지 이벤트 계측기 및 관련 소비자에 대한 자세한 내용

은 27.12.5절. "성능 스키마 스테이지 이벤트 테이블"을 참조하십시오.

1. 스테이지/인노드/테이블스페이스 변경(암호화) 도구를 활성화합니다:

```
mysql> USE performance_schema;
mysql> UPDATE setup_instruments SET ENABLED = 'YES'
       여기서 이름은 'stage/innodb/alter tablespace (암호화)'와 유사합니다;
```

2. 이벤트_스테이지_현재, 이벤트_스테이지_역사, 이벤트_스테이지_역사_길이가 포함된 스테이지 이벤트 소비자 테이블을 사용하도록 설정합니다.

```
mysql> UPDATE setup_consumers SET ENABLED = 'YES' WHERE NAME LIKE '%stages%';
```

3. 테이블 공간 암호화 작업을 실행합니다. 이 예에서는 ts1이라는 일반 테이블스페이스가 암호화됩니다.

```
mysql> ALTER TABLESPACE ts1 ENCRYPTION = 'Y';
```

4. 성능 스키마 events_stages_current 테이블을 쿼리하여 암호화 작업의 진행 상황을 확인합니다. WORK_ESTIMATED는 테이블 공간의 총 페이지 수를 보고합니다. WORK_COMPLETED는 처리된 페이지 수를 보고합니다.

```
mysql> SELECT EVENT_NAME, WORK_ESTIMATED, WORK_COMPLETED FROM events_stages_current;
+----------------------------------------+----------------+----------------------+
| 이벤트_이름                              | 작업_완료 | 작업_추정 |
+----------------------------------------+----------------+----------------------+
| STAGE/INNODB/ALTER TABLE스페이스(암호화)                  |1056            |1407 |
+----------------------------------------+----------------+----------------------+
```

암호화 작업이 완료된 경우 events_stages_current 테이블은 빈 집합을 반환합니다. 이 경우, 완료된 작업에 대한 이벤트 데이터를 보기 위해 events_stages_history 테이블을 확인할 수 있습니다. 예를 들어

```
mysql> SELECT EVENT_NAME, WORK_COMPLETED, WORK_ESTIMATED FROM events_stages_history;
+----------------------------------------+----------------+----------------------+
| 이벤트_이름                              | 작업_완료 | 작업_추정 |
+----------------------------------------+----------------+----------------------+
| STAGE/INNODB/ALTER TABLESPACE (암호화)                    |1407 |1407       |1407 |
+----------------------------------------+----------------+----------------------+
```

암호화 사용 참고 사항

- 암호화 옵션을 사용하여 기존 파일 단위 테이블 테이블 스페이스를 변경할 때는 적절하게 계획을 세우십시오. 파일 단위 테이블 공간에 있는 테이블은 COPY 알고리즘을 사용하여 재구성됩니다. INPLACE 알고리즘은 일반 테이블스페이스 또는 mysql 시스템 테이블스페이스의 ENCRYPTION 속성을 변경할 때 사용됩니다. INPLACE 알고리즘은 일반 테이블스페이스에 있는 테이블에 대한 동시 DDL을 허용합니다. 동시 DDL은 차단됩니다.
- 일반 테이블 스페이스 또는 mysql 시스템 테이블 스페이스가 암호화되면 테이블 스페이스에 있는 모든 테이블이 암호화됩니다. 마찬가지로 암호화된 테이블 공간에서 생성된 테이블도 암호화됩니다.
- 정상 작동 중에 서버가 종료되거나 중지되는 경우 이전에 구성한 것과 동일한 암호화 설정을 사용하여 서버를 다시 시작하는 것이 좋습니다.
- 첫 번째 마스터 암호화 키는 첫 번째 새 테이블스페이스 또는 기존 테이블스페이스가 암호화될 때 생성됩니다.
- 마스터 키 회전은 테이블스페이스 키를 다시 암호화하지만 테이블스페이스 키 자체는 변경하지 않습니다. 테이블 스페이스 키를 변경하려면 암호화를 비활성화했다가 다시 활성화해야 합니다. 파일 단위 테이블 스페이스의 경우 테이블 스페이스를 다시 암호화하는 것은 테이블을 다시 작성하는 ALGORITHM=COPY 작업입니다. 일반 테이블 스페이스 및 mysql 시스템 테이블 스페이스의 경우, 테이블 스페이스에 있는 테이블을 다시 빌드할 필요가 없는 ALGORITHM=INPLACE 작업입니다.

- 압축 및 암호화 옵션을 모두 사용하여 테이블을 생성하는 경우 테이블 공간 데이터가 암호화되기 전에 압축이 수행됩니다.
- 키링 데이터 파일(키링_파일_데이터 또는 키링_암호화된_파일_데이터로 명명된 파일)이 비어 있거나 누락된 경우, `ALTER INSTANCE ROTATE INNODB MASTER KEY`를 처음 실행하면 마스터 암호화 키가 만들어집니다.
- `component_keyring_file` 또는 `component_keyring_encrypted_file` 컴포넌트를 제거해도 기존 키링 데이터 파일은 제거되지 않습니다. `keyring_file` 또는 `keyring_encrypted_file` 플러그인을 제거해도 기존 키링 데이터 파일은 제거되지 않습니다.
- 키링 데이터 파일은 테이블스페이스 데이터 파일과 같은 디렉터리에 두지 않는 것이 좋습니다.

- 런타임 또는 서버를 재시작할 때 키링_파일_데이터 또는 키링_암호화된_파일_데이터 설정을 수정하면 이전에 암호화된 테이블스페이스에 액세스할 수 없게 되어 데이터가 손실될 수 있습니다.
- 전체 텍스트 인덱스를 추가할 때 암시적으로 생성되는 InnoDB 전체 텍스트 인덱스 테이블에 대해 암호화가 지원됩니다. 관련 정보는 InnoDB 전체 텍스트 인덱스 테이블을 참조하십시오.

암호화 제한

- 고급 암호화 표준(AES)은 유일하게 지원되는 암호화 알고리즘입니다. InnoDB 테이블스페이스 암호화는 테이블스페이스 키 암호화에 ECB(전자 코드북) 블록 암호화 모드를 사용하고 데이터 암호화에 CBC(암호 블록 체인) 블록 암호화 모드를 사용합니다. CBC 블록 암호화 모드에서는 패딩이 사용되지 않습니다. 대신 InnoDB는 암호화할 텍스트가 블록 크기의 배수인지 확인합니다.
- 암호화는 테이블별 파일 테이블 스페이스, 일반 테이블 스페이스 및 mysql 시스템 테이블 스페이스에만 지원됩니다. InnoDB 시스템 테이블스페이스를 비롯한 다른 테이블스페이스 유형에는 암호화가 지원되지 않습니다.
- 암호화된 파일 단위 테이블 테이블 스페이스, 일반 테이블 스페이스 또는 mysql 시스템 테이블 스페이스에서 암호화를 지원하지 않는 테이블 스페이스 유형으로 테이블을 이동하거나 복사할 수 없습니다.
- 암호화된 테이블 공간에서 암호화되지 않은 테이블 공간으로 테이블을 이동하거나 복사할 수 없습니다. 그러나 암호화되지 않은 테이블 공간에서 암호화된 테이블 공간으로 테이블을 이동하는 것은 허용됩니다. 예를 들어 암호화되지 않은 테이블별 파일 또는 일반 테이블 공간에서 암호화된 일반 테이블 공간으로 테이블을 이동하거나 복사할 수 있습니다.
- 기본적으로 테이블 스페이스 암호화는 테이블 스페이스의 데이터에만 적용됩니다. innodb_redo_log_encrypt 및 innodb_undo_log_encrypt를 활성화하여 로그 재실행 및 실행 취소 데이터를 암호화할 수 있습니다. 로그 암호화 다시 실행 및 로그 암호화 실행 취소를 참조하세요. 바이너리 로그 파일 및 릴레이 로그 파일 암호화에 대한 자세한 내용은 17.3.2절, "바이너리 로그 파일 및 릴레이 로그 파일 암호화"를 참조한다.
- 암호화된 테이블 공간에 있거나 이전에 암호화된 테이블 공간에 있던 테이블의 저장소 엔진을 변경하는 것은 허용되지 않습니다.

15.14 InnoDB 시작 옵션 및 시스템 변수

- 참 또는 거짓인 시스템 변수는 서버 시작 시 이름을 지정하여 사용하도록 설정하거나 --skip- 접두사를 사용하여 사용하지 않도록 설정할 수 있습니다. 예를 들어, InnoDB 적응형 해시 인덱스를 활성화 또는 비활성화하려면 명령줄에서 --innodb-adaptive-hash-index 또는 --skip-innodb- adaptive-hash-index를 사용하거나, 옵션 파일에서 innodb_adaptive_hash_index 또는 skip_innodb_adaptive_hash_index를 사용할 수 있습니다.

- 일부 변수 설명은 변수의 "활성화" 또는 "비활성화"를 나타냅니다. 이러한 변수는 `SET` 문을 사용하여 `ON` 또는 `1`로 설정하여 활성화하거나 `OFF` 또는 `0`으로 설정하여 비활성화할 수 있습니다. 부울 변수는 시작할 때 1과 `0`뿐만 아니라 `ON`, `TRUE`, `OFF` 및 `FALSE`(대소문자 구분 없음) 값으로 설정할 수 있습니다. 섹션 4.2.2.4, "프로그램 옵션 수정자"를 참조하세요.
- 숫자 값을 취하는 시스템 변수는 명령줄에서 *`--var_name=값으로`* 지정하거나 옵션 파일에서 *`var_name=값으로`* 지정할 수 있습니다.
- 많은 시스템 변수는 런타임에 변경할 수 있습니다(섹션 5.1.9.2, "동적 시스템 변수" 참조).
- `GLOBAL` 및 `SESSION` 변수 범위 수정자에 대한 자세한 내용은 `SET` 문 설명서를 참조하세요.
- 특정 옵션은 `InnoDB` 데이터 파일의 위치 및 레이아웃을 제어합니다. 15.8.1절, 'InnoDB 시작 구성'에서 이러한 옵션을 사용하는 방법을 설명합니다.
- 처음에는 사용하지 않을 수도 있는 일부 옵션은 머신 용량과 데이터베이스 워크로드에 따라 `InnoDB` 성능 특성을 조정하는 데 도움이 됩니다.

- 옵션 및 시스템 변수 지정에 대한 자세한 내용은 4.2.2절, "프로그램 옵션 지정하기"를 참조하십시오.

표 15.24 InnoDB 옵션 및 변수 참조

이름	Cmd-Line	옵션 파일	시스템 변수	상태 변수	변수 범위	동적
daemon_memcYaeched_enableY_beisnlog	예				글로벌	아
daemon_memcYaeched_engineY_leibs_name	예				글로벌	니
daemon_memcYaeched_engineY_leibs_path	예				글로벌	오
daemon_memcYaeched_option	예				글로벌	아
daemon_memcYaeched_r_batchY_essize	예				글로벌	니
daemon_memcYaeched_w_batcYhe_ssize	예				글로벌	오
foreign_key_checks			예		글로벌	아
			예		모두	니
			예			오
						아
						니
						오
						예
innodb	예	예				
innodb_adaptivYee_sflushing	예	예			글로벌	예
innodb_adaptivYee_sflushing_lw	예				글로벌	예
m예	예				글로벌	예
innodb_adaptivYee_shash_index예	예				글로벌	아
innodb_adaptivYee_shash_index_Ypeasrts	예				글로벌	니
innodb_adaptivYee_smax_sleep_Ydeeslay	예				글로벌	오
						예
innodb_api_disYaebsle_rowlock innodb_api_enYabelse_binlog innodb_api_enYabelse_m	d l i	n n o d			b_api _trxY _lees vel	예 예 예 예 예

예 예 예 예 예

아니요

글로벌

예 아니요

글로벌

예 아니요

글로벌

예 예

글로벌

글로벌

innodb_autoexYteensd_incremen Yt es		예	글로벌	예
innodb_autoincY_elosck_mode	예	예	글로벌	아니요
innodb_backgrYouensd_drop_listY_ es pty Innodb_버퍼_풀_바이트_data		예 예 예	글로벌	예 아니오
Innodb_버퍼_풀_바이트_dirty		예	글로벌	아니 오
innodb_버퍼 _Ypeosol_chunk_sizYees			글로벌 글로벌	
innodb_buffer_Ypeosol_de bug	예	예	글로벌	아니요
innodb_버퍼 _Ypeosol_dump_at_Ysehsutdown		i o n d n b	_버 퍼	_Ypeosol_dump_no Ywes innodb_버퍼 _Ypeosol_dump_pc

Yt es innodb_버퍼_pool_dump_status	예		글로벌	예 예
innodb_버퍼_Ypeosol_filename Yes	예		글로벌	예 예
innodb_버퍼_Ypeosol_in_core_fYiles	예	예	글로벌	예 예
innodb_버퍼_Ypeosol_load_aboYrtes			글로벌	예 예
innodb_buffer_Ypeosol_load_at_sYtea srtup	예		글로벌	예 예
innodb_buffer_Ypeosol_load_nowYes	예		글로벌	예 예
Innodb_buffer_pool_load_status	예	예	글로벌	예 예
Innodb_buffer_pool_pages_data	예	예	글로벌	예 예
Innodb_buffer_pool_pages_dirty	예	예	글로벌	아니
Innodb_buffer_pool_pages_flushed	예	예	글로벌	오 아
Innodb_buffer_pool_pages_free	예	예	글로벌	니오
		예	글로벌	아니
			글로벌	오 아
			글로벌	니오
			글로벌	아니
			글로벌	오 아
			글로벌	니오
			글로벌	아니
			글로벌	오 아
			글로벌	니오
			글로벌	
			글로벌	

이름	Cmd-Line	옵션 파일	시스템 변수	상태 변수	변수 범위	동적
Innodb_버퍼풀_페이지_랫치				예	글로벌	아니요
Innodb_버퍼풀_페이지_기타				예	글로벌	아니요
Innodb_버퍼풀_페이지_총계				예	글로벌	아니요
Innodb_버퍼풀_읽기_앞				예	글로벌	아니요
Innodb_버퍼풀_읽기_앞_퇴거				예	글로벌	아니요
Innodb_버퍼풀_읽기_요청				예	글로벌	아니요
Innodb_버퍼풀_읽기				예	글로벌	아니요
Innodb_버퍼풀_사이즈조정_상태				예	글로벌	아니요
				예		
				예		
				예		
				예		
innodb_buffer_Ypeosol_size	예	예			글로벌	예
Innodb_버퍼 풀_대기_무료				예 예	글로벌	아
Innodb_버퍼 풀_쓰기_요청			예		글로벌	니
innodb_changeY_ebsuffer_max_sYizees					글로벌	오
						아
						니
						오
						예
innodb_changeY_ebsuffering	예	예			글로벌	예
innodb_checkpYoeinst_disabled	예	예			글로벌	예
innodb_checksYuems_algorithm			예		글로벌	예
예 innodb_commitY_ecsoncurrency			예		글로벌	예
예						
innodb_compreYsess_debug			i o _ t			fiYlee_spath
innodb_concurYreenscy_tic			n d d a			innodb_data_
kets			n b a _			fsyncs

innodb_data_hYomese_dir	예	예		글로벌	예 예
	예	예		글로벌	아니
	예	예	예	글로벌	오 아
				글로벌	니오
	예	예		글로벌	
				글로벌	
Innodb_data_pending_fsyncs			예	글로벌	아니요
Innodb_data_pending_reads			예	글로벌	아니요
Innodb_data_pending_writes			예	글로벌	아니요
Innodb_data_read			예	글로벌	아니요
Innodb_data_reads			예	글로벌	아니요
Innodb_data_writes			예	글로벌	아니요
Innodb_data_written			예	글로벌	아니요
Innodb_dblwr_pages_written			예	글로벌	아니요
Innodb_dblwr_writes			예	글로벌	아니요
			예	글로벌	아니요
			예		
			예		
			예		
			예		
			예		
			예		
innodb_ddl_buYffes_size	예	예		둘 다	예
innodb_ddl_logY_ecsrash_reset_Ydesug		예		글로벌	예
innodb_ddl_thrYeaedss	예	예		예 예 예 예	모두 글로벌 글
innodb_deadloYcke_sdetect	예			예 예	로벌 글로벌
innodb_dedicatYeeds_server	예				글로벌
innodb_defaultY_reosw_format innodb_directoYries	예				
	예				

예 예 아니오 예 아니오

innodb_disableY_essort_file_cachYees			예		글로벌	예
innodb_doubleYweriste		예	예		글로벌	예
innodb_doubleYweriste_batch_sizYees			예		글로벌	아니요
innodb_doubleYweriste_dir		예	예		글로벌	아니요

이름	Cmd-Line	옵션 파일	시스템 변수	상태 변수	변수 범위	동적
innodb_doubleYweriste_files		예	예		글로벌	아
innodb_doubleYweriste_pages		예	예		글로벌	니
innodb_fast_shYuetdsown		예	예		글로벌	오
innodb_file_perY_etasble		예	예		글로벌	아
innodb_fill_factYoer s		예	예		글로벌	니
		예	예		글로벌	오
						예
						예
						예
innodb_flush_loYge_sat_timeout		예	예 예		글로벌	예
innodb_flush_loYge_sat_trx_commYeits					글로벌	예
innodb_flush_mYeetshod		예	예		글로벌	아
innodb_flush_nYeeigshbors		예	예		글로벌	니
innodb_flush_sYyensc		예	예		글로벌	요
innodb_flushingY_easvg_loops		예	예		글로벌	예
		예	예		글로벌	예
						예
innodb_force_lYoaeds_corrupted			예		글로벌	아니요
Yes						
innodb_force_rYeecos very		예	예		글로벌	아
innodb_fsync_tYheresshold		예	예		글로벌	니
innodb_ft_aux_table			예		글로벌	오
innodb_ft_cachYee_ssize		예	예		글로벌	예
			예		글로벌	예
						아
						니

오

innodb_ft_enabYlesdiag_print	예	예		글로벌	예
innodb_ft_enabYlesstopword	예	예		모두	예
innodb_ft_maxY_teosken_size	예	예		글로벌	아
innodb_ft_min_Ytoeksen_size	예	예		글로벌	니
					오
					아
					니
					오
innodb_ft_numY_wesord_optimizeYes		예		글로벌	예
innodb_ft_resuYlt_ecsache_limit		예		글로벌	예
예		예		글로벌	예
innodb_ft_servYere_sstopword_taYbeles					
innodb_ft_sort_Ypells_degree	예	예 예		글로벌	아니요
innodb_ft_totalY_ceasche_size	예			글로벌	아니요
innodb_ft_userY_setsopword_tabYlees		예		모두	예 아
Innodb_have_atomic_builtins			예	글로벌	니요
innodb_idle_fluYsehs_pct	예	예		글로벌	예
innodb_io_capaYceist	예	예		글로벌	예
y innodb_io_capaYceisty_max	예	예		글로벌	예
innodb_limit_opYteimsistic_insert_Yd eesbug		예		글로벌	예
innodb_lock_wYaiet_stimeout	예	예 예		모두	예
innodb_log_buYffes_size	예			글로벌	예
innodb_log_cheYcekspoint_fuzzyY_neosw		예		글로벌	예
innodb_log_cheYcekspoint_now		예		글로벌	예
예					
innodb_log_cheYcekss	u	m		s	

예	예	글	예		
		로			
		벌			
innodb_log_com예스	Yepsressed_pag	예		글로벌	예
innodb_log_file	Y_essize	예	예 예	글로벌	아니요
innodb_log_file oup	Yse_sin_gr	예		글로벌	아니요
innodb_log_gro	Yueps_home_dir	예	예 예	글로벌	아
innodb_log_spi Ymes	Yne_scpu_abs_lw			글로벌	니
					오
					예

이름	Cmd-Line	옵션 파일	시스템 변수	상태 변수	변수 범위	동적
innodb_log_spiYne_scpu_pct_hwY mes			예		글로벌	예 예
innodb_log_waYite_sfor_flush_spYin e_shwm innodb_log_waits			예	예 예	글로벌	아니
innodb_log_wriYtee_sahead_sizeYes			예		글로벌	오 예
Innodb_log_write_requests					글로벌	아니
					글로벌	오
					글로벌	
innodb_log_wriYteers_threads		예	예		글로벌	예 아
innodb_log_writes				예		
innodb_lru_scaYne_sdepth		예	예		글로벌	니오
					글로벌	예
innodb_max_dYirteys_pages_pct 예			예 예		글로벌 글로	예
					벌	예
innodb_max_dYirteys_pages_pct_ Ylewsm						
innodb_max_pYuregse _lag		예	예		글로벌	예
innodb_max_pYuregse_lag_dela y예			예		글로벌	예
innodb_max_uYndeos_log_ size		예	예		글로벌	예
innodb_merge_Ytehsreshold_set_Yaells _debug			예		글로벌	예
innodb_monitoYr_edsis able		예	예		글로벌	예
innodb_monitoYr_eesn able		예	예		글로벌	예
innodb_monitoYr_erse set		예	예	예	글로벌	예
innodb_monitoYr_erseset _all		예	예		글로벌	예
innodb_num_open_files		예	예		글로벌	예
innodb_numa_Yinetesrle ave					글로벌	아
innodb_old_bloYceks_ pct		예	예		글로벌	니
innodb_old_bloYceks_ti me		예	예		글로벌	오
					글로벌	아

니오 예 예

innodb_online_Yaeltser_log_max_Yseisze		예			글로벌	예
innodb_open_fYile ess	예	예			글로벌	예
innodb_optimizYee_sfulltext_onlyYes		예			글로벌	예 아
Innodb_os_log_fsyncs				예	글로벌	니오
Innodb_os_log_pending_fsyncs						
Innodb_os_log_pending_writes				예	글로벌	아니
Innodb_os_log_witten				예		
					글로벌	오 아
				예	글로벌	니오
				예		
					글로벌	
innodb_page_cYleeasners	예	예 예			글로벌	아니요
Innodb_page_size				예	글로벌	아니요
innodb_page_sYizees						
Innodb_pages_created	예			예	글로벌	아니요
Innodb_pages_read					글로벌	아니요
Innodb_pages_written				예	글로벌	아니요
				예		
					글로벌	아니요
innodb_paralleYl_eresad_threads Yes		예			세션	예
innodb_print_aYll_edseadl ocks	예	예			글로벌	예
innodb_print_dYdle_slog s	예	예			글로벌	예
innodb_purge_Ybeastch_ size	예	예			글로벌	예
innodb_purge_Yrseesg_truncate_Yfrees quency		예			글로벌	예
innodb_purge_이더시 드	예	예			글로벌	아니요

이름	Cmd-Line	옵션 파일	시스템 변수	상태 변수	변수 범위	동적
innodb_randomY_ersead_ahead	예		예 예		글로벌	예
innodb_read_aYhesad_threshold					글로벌	예
예						
innodb_read_ioY_ethsreads	예		예		글로벌	아니요
innodb_redo_loYge_sarchive_dirs	Yes		예		글로벌	예
innodb_redo_loYge_scapacity	예		예		글로벌	예
Innodb_redo_log_capacity_resized				예	글로벌	아니요
Innodb_redo_log_checkpoint_lsn				예	글로벌	아니요
Innodb_redo_log_current_lsn				예	글로벌	아니요
Innodb_redo_log_enabled				예	글로벌	아니요
				예		
				예		
innodb_redo_loYge_sencrypt	예		예		글로벌	예
Innodb_redo_log_flushed_to_disk_lsn				예	글로벌	아니요
Innodb_redo_log_logical_size				예	글로벌	아니요
Innodb_redo_log_physical_size				예	글로벌	아니요
Innodb_redo_log_read_only				예	글로벌	아니요
Innodb_redo_log_resize_status				예	글로벌	아니요
Innodb_redo_log_uuid				예	글로벌	아니요
				예		
				예		
innodb_replicatYioens_delay	예		예		글로벌	예
innodb_rollbackY_eosn_timeout						
innodb_rollbackY_essegments	예		예		글로벌	아
	예		예		글로벌	니
						오
						예
Innodb_row_lock_current_waits			l	d	ow_	Innodb_rows_deleted
Innodb_row_lock_time			n	b	lock	Innodb_rows_inserte
Innodb_row_lock_time_avg			n	_	_wa	d Innodb_rows_read
Innodb_row_lock_time_max			o	r	its	Innodb_rows_update

d			예	글로벌	아
innodb_segmeYnte_sreserve_fac					
Ytoers			예	글로벌	니
			예	글로벌	요
			예	글로벌	아
			예	글로벌	니
			예	글로벌	요
		예			
			예	글로벌	아
			예	글로벌	니
			예	글로벌	요
			예	글로벌	아
			예	글로벌	니
			예	글로벌	요
			예	글로벌	아
			예	글로벌	니
				글로벌	요
					아
					니
					요
					아
					니
					요
					예
innodb_sort_buYffeesr_size	예	예 예		글로벌	아
innodb_spin_wYaeit_sd					
elay	예			글로벌	니
					오
					예
innodb_spin_wYaeit_spause_mulYt		예		글로벌	예
ipelsier					

innodb_stats_aYuetos_recalc		예	예		글로벌	예
innodb_stats_inYcelusde_delete_Ym easrked		예	예		글로벌	예
innodb_stats_m	예측 방법	예	예		글로벌	예
innodb_stats_oYne_smetadata		예	예		글로벌	예
innodb_stats_p	예측 스텐트	예	예		글로벌	예
innodb_stats_p	예어스 스텐트		예		글로벌	예
sampYles	페이지		예		글로벌	예
innodb_stats_trYaenssient_samplYee_	페이지					
innodb-상태 파일		예	예			

이름	Cmd-Line	옵션 파일	시스템 변수	상태 변수	변수 범위	동적
innodb_status_Yoeustput		예	예		글로벌	예
innodb_status_Yoeustput_locks		예	예		글로벌	예
innodb_strict_mYoedse		예	예		모두	예
innodb_sync_aYrreasy_size		예	예		글로벌	아
innodb_sync_dYeebsug		예	예		글로벌	니
innodb_sync_sYpeins_loops		예	예		글로벌	오
innodb_system_rows_deleted				예	글로벌	아니요
innodb_system_rows_inserted				예	글로벌	아니요
innodb_system_rows_read				예	글로벌	아니요
innodb_table_loYcekss		예	예		둘 다	예
innodb_temp_dYaetas_파일_경로		예	예		글로벌	아니요
innodb_temp_tYabelsespaces_dir		예	예		글로벌	아니요
innodb_thread_Yceosncurrency		예	예		글로벌	예
innodb_thread_Yselesep_delay		예	예		글로벌	예
innodb_tmpdir		예	예		모두	예
Innodb_truncated_status_writes				예	글로벌	아니요
innodb_undo_dYireesctory		예	예	예	예	글로벌
innodb_undo_lYoge_sencrypt		예		예		글로벌
innodb_undo_lYoge_struncate		예				글로벌
innodb_undo_tYabelsespaces		예				

아니요 예 예 예

Innodb_undo_tablespaces_active		예	글로벌	아니요
Innodb_undo_tablespaces_explicit		예	글로벌	아니요
Innodb_undo_tablespaces_implicit		예	글로벌	아니요
Innodb_undo_tablespaces_total		예	글로벌	아니요
		예		
innodb_use_fdYateass ync	예	예 예	글로벌 글로	예
innodb_use_naYtievse_aio	예		벌	아 니 요
innodb_validateY_etsablespace_pYaetshs		예	글로벌	아니요
innodb_version		예	글로벌	아니요
innodb_write_ioY_etshreads 고유	예	예	글로벌	아
_체크		예	모두	니 오 예

InnoDB 명령 옵션

- `--innodb[=값]`

명령줄 형식	`--innodb[=값]`
사용 중단	예
유형	열거형
기본값	켜기
유효한 값	끄기 켜기 FORCE

서버가 InnoDB 지원으로 컴파일된 경우 InnoDB 스토리지 엔진의 로드를 제어합니다. 이 옵션은 3상태 형식이며, 가능한 값은 OFF, ON 또는 FORCE입니다. 5.6.1절. "플러그인 설치 및 제거"를 참조하세요.

InnoDB를 사용하지 않으려면 --innodb=OFF 또는 --skip-innodb를 사용합니다. 이 경우 기본 스토리지 엔진이 InnoDB이므로 --default-storage- engine 및 --default-tmp-storage-engine을 사용하여 영구 및 임시 테이블 모두에 대해 기본값을 다른 엔진으로 설정하지 않으면 서버가 시작되지 않습니다.

InnoDB 스토리지 엔진은 더 이상 비활성화할 수 없으며 --innodb=OFF 및 --skip- innodb 옵션은 더 이상 사용되지 않으며 아무런 영향을 미치지 않습니다. 이 옵션을 사용하면 경고가 표시됩니다. 이러한 옵션은 향후 MySQL 릴리스에서 제거될 예정입니다.

- --innodb-status-file

명령줄 형식	--innodb-status-file[={OFF|ON}]
유형	부울
기본값	꺼짐

innodb-status-file startup 옵션은 InnoDB가 데이터 디렉터리에 innodb_status.*pid*라는 파일을 생성하고 약 15초마다 이 파일에 엔진 상태 표시 출력을 기록할지 여부를 제어합니다.

innodb_status.*pid* 파일은 기본적으로 생성되지 않습니다. 이 파일을 생성하려면 -- innodb-status-file 옵션을 사용하여 mysqld를 시작합니다. 서버가 정상적으로 종료되면 InnoDB는 이 파일을 제거합니다. 비정상 종료가 발생하면 상태 파일을 수동으로 제거해야 할 수 있습니다.

innodb-status-file 옵션은 일시적으로 사용하기 위한 것으로, 엔진 INNODB 상태 출력 생성은 성능에 영향을 미칠 수 있고 시간이 지남에 따라 innodb_status.*pid* 파일이 상당히 커질 수 있기 때문입니다.

관련 정보는 섹션 15.17.2, 'InnoDB 모니터 사용'을 참조하세요.

- --skip-innodb

InnoDB 스토리지 엔진을 비활성화합니다. innodb에 대한 설명을 참조하세요.

InnoDB 시스템 변수

- DAEMON_EMCACHED_ENABLE_BINLOG

명령줄 형식	--daemon-memcached-enable-binlog[={OFF|ON}]
시스템 변수	DAEMON_EMCACHED_ENABLE_BINLOG
범위	글로벌

동적	아니요
SET_VAR 힌트 적용	아니요
유형	부울
기본값	꺼짐

소스 서버에서 이 옵션을 활성화하면 MySQL 바이너리 로그와 함께 InnoDB 멤캐시드 플러그인 (daemon_memcached)을 사용할 수 있습니다. 이 옵션은 서버를 시작할 때만 설정할 수 있습니다. 또한 소스 서버에서 --log-bin 옵션을 사용하여 MySQL 바이너리 로그를 활성화해야 합니다.

자세한 내용은 섹션 15.20.7, "InnoDB 멤캐시드 플러그인 및 복제"를 참조하세요.

- DAEM_EMCACHED_ENGINE_LIB_NAME

명령줄 형식	--daemon-memcached-engine-lib- name=파일_이름
시스템 변수	DAEM_EMCACHED_ENGINE_LIB_NAME
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	파일 이름
기본값	innodb_engine.so

InnoDB 멤캐시드 플러그인을 구현하는 공유 라이브러리를 지정합니다.

자세한 내용은 섹션 15.20.3, "InnoDB 멤캐시드 플러그인 설정"을 참조하세요.

- DAEMON_EMCACHED_ENGINE_LIB_PATH

명령줄 형식	--daemon-memcached-engine-lib- 경로=dir_name
시스템 변수	DAEMON_EMCACHED_ENGINE_LIB_PATH
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	디렉토리 이름
기본값	NULL

InnoDB 멤캐시드 플러그인을 구현하는 공유 라이브러리가 포함된 디렉터리 경로입니다. 기본값은 MySQL 플러그인 디렉터리를 나타내는 NULL입니다. MySQL 플러그인 디렉터리 외부에 있는 다른 스토리지 엔진에 대한 멤캐시드 플러그인을 지정하지 않는 한 이 매개변수를 수정할 필요가 없습니다.

자세한 내용은 섹션 15.20.3, "InnoDB 멤캐시드 플러그인 설정"을 참조하세요.

- DAEMON_MEMCACHED_OPTION

명령줄 형식	--데몬-멤캐시드-옵션=옵션
시스템 변수	DAEMON_MEMCACHED_OPTION
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	문자열

기본값	

시작 시 공백으로 구분된 멤캐시드 옵션을 기본 멤캐시드 메모리 객체 캐싱 데몬에 전달하는 데 사용됩니다. 예를 들어, memcached가 수신 대기하는 포트를 변경하거나, 최대 동시 연결 수를 줄이거나, 키-값 쌍의 최대 메모리 크기를 변경하거나, 오류 로그에 대한 디버깅 메시지를 활성화할 수 있습니다.

사용법에 대한 자세한 내용은 15.20.3절, 'InnoDB memcached 플러그인 설정하기'를 참조하세요. 멤캐시드 옵션에 대한 자세한 내용은 멤캐시드 매뉴얼 페이지를 참조하세요.

- DAEMON_MEMCACHED_R_BATCH_SIZE

명령줄 형식	--daemon-memcached-r-batch-size=#
시스템 변수	DAEMON_MEMCACHED_R_BATCH_SIZE
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	1
최소값	1
최대 값	1073741824

멤캐시드 읽기 작업(get 작업)을 수행하기 전에 수행할 멤캐시드 읽기 작업의 수를 지정합니다. 커밋하여새 트랜잭션을 시작합니다. daemon_memcached_w_batch_size에 대응하는 값입니다.

이 값은 기본적으로 1로 설정되어 있으므로 SQL 문을 통해 테이블에 대한 모든 변경 사항이 멤캐시된 작업에 즉시 표시됩니다. 멤캐시드 인터페이스를 통해서만 기본 테이블에 액세스하는 시스템에서 잦은 커밋으로 인한 오버헤드를 줄이려면 이 값을 늘릴 수 있습니다. 값을 너무 크게 설정하면 실행 취소 또는 재실행 데이터의 양으로 인해 장기 실행 트랜잭션과 마찬가지로 약간의 스토리지 오버헤드가 발생할 수 있습니다.

자세한 내용은 섹션 15.20.3, "InnoDB 멤캐시드 플러그인 설정"을 참조하세요.

- DAEMON_MEMCACHED_W_BATCH_SIZE

명령줄 형식	--daemon-memcached-w-batch-size=#
시스템 변수	DAEMON_MEMCACHED_W_BATCH_SIZE
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	1
최소값	1
최대 값	1048576

새 트랜잭션을 시작하기 위해 COMMIT을 수행하기 전에 추가, 설정, incr과 같은 멤캐시드 쓰기 작업을 수행할 멤캐시드 쓰기 작업의 수를 지정합니다. daemon_memcached_r_batch_size에 대응하는 값입니다.

이 값은 저장 중인 데이터가 중단 시 보존해야 할 중요한 데이터이므로 즉시 커밋해야 한다는 가정 하에 기본적으로 1로 설정됩니다. 중요하지 않은 데이터를 저장할 때는 이 값을 늘려 잦은 커밋으로 인한 오버헤드를 줄일 수 있지만, 예기치 않은 종료가 발생하면 커밋되지 않은 마지막 N-1개의 쓰기 작업이 손실될 수 있습니다.

자세한 내용은 섹션 15.20.3, "InnoDB 멤캐시드 플러그인 설정"을 참조하세요.

- `INNODB_ADAPTIVE_FLUSHING`

명령줄 형식	`--innodb-adaptive-flushing[={OFF|ON}]`
시스템 변수	`INNODB_ADAPTIVE_FLUSHING`
범위	글로벌
동적	예

SET_VAR 힌트 적용	아니요
유형	부울
기본값	켜기

워크로드에 따라 InnoDB 버퍼 풀의 더티 페이지 플러시 속도를 동적으로 조정할지 여부를 지정합니다. 플러시 속도를 동적으로 조정하는 것은 I/O 활동의 폭주를 피하기 위한 것입니다. 이 설정은 기본적으로 활성화되어 있습니다. 자세한 내용은 섹션 15.8.3.5, "버퍼 풀 플러시 구성"을 참조하세요. 일반적인 I/O 튜닝에 대한 조언은 섹션 8.5.8, "InnoDB 디스크 I/O 최적화"를 참조하세요.

- INNODB_ADAPTIVE_FLUSHING_LWM

명령줄 형식	--innodb-adaptive-flushing-lwm=#
시스템 변수	INNODB_ADAPTIVE_FLUSHING_LWM
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	10
최소값	0
최대 값	70

적응형 플러싱이 활성화되는 재실행 로그 용량의 백분율을 나타내는 저수위 표시를 정의합니다. 자세한 내용은 섹션 15.8.3.5, "버퍼 풀 플러싱 구성"을 참조하십시오.

- innodb_adaptive_hash_index

명령줄 형식	--innodb-adaptive-hash-index[={OFF|ON}]
시스템 변수	innodb_adaptive_hash_index
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	부울
기본값	켜기

InnoDB 적응형 해시 인덱스의 활성화 또는 비활성화 여부. 워크로드에 따라 쿼리 성능을 개선하기 위해 적응형 해시 인덱싱을 동적으로 사용 또는 사용하지 않도록 설정하는 것이 바람직할 수 있습니다. 적응형 해시 인덱스가 모든 워크로드에 유용하지 않을 수 있으므로 실제 워크로드를 사용하여 적응형 해시 인덱스를 사용하거나 사용하지 않도록 설정한 상태에서 벤치마크를 수행하십시오. 자세한 내용은 섹션 15.5.3, "적응형 해시 인덱스"를 참조하십시오.

이 변수는 기본적으로 활성화되어 있습니다. 서버를 다시 시작하지 않고도 `SET GLOBAL` 문을 사용하여 이 매개변수를 수정할 수 있습니다. 런타임에 설정을 변경하려면 전역 시스템 변수를 설정할 수 있는 충분한 권한이 필요합니다. 섹션 5.1.9.1, "시스템 변수 권한"을 참조하세요. 또한 다음을 사용할 수도 있습니다. 서버 시작 시 `--skip-innodb-adaptive-hash-index`를 사용하여 비활성화합니다.

적응형 해시 인덱스를 비활성화하면 해시 테이블이 즉시 비워집니다. 해시 테이블이 비워져 있는 동안에도 정상적인 작업을 계속할 수 있으며, 해시 테이블을 사용하던 쿼리를 실행하면 대신 인덱스 B-트리에 직접 액세스합니다. 적응형 해시 인덱스가 다시 활성화되면 정상 작동 중에 해시 테이블이 다시 채워집니다.

- `innodb_adaptive_hash_index_parts`

명령줄 형식	--innodb-adaptive-hash-index-parts=#
시스템 변수	innodb_adaptive_hash_index_parts
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	숫자
기본값	8
최소값	1
최대 값	512

적응형 해시 인덱스 검색 시스템을 분할합니다. 각 인덱스는 특정 파티션에 바인딩되며, 각 파티션은 별도의 래치로 보호됩니다.

적응형 해시 인덱스 검색 시스템은 기본적으로 8개 부분으로 분할되어 있습니다. 최대 설정은 512입니다.

관련 정보는 섹션 15.5.3, "적응형 해시 인덱스"를 참조하세요.

- innodb_adaptive_max_sleep_delay

명령줄 형식	--innodb-adaptive-max-sleep-delay=#
시스템 변수	innodb_adaptive_max_sleep_delay
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	150000
최소값	0
최대 값	1000000
단위	마이크로초

현재 워크로드에 따라 InnoDB가 innodb_thread_sleep_delay 값을 자동으로 위 또는 아래로 조정할 수 있도록 허용합니다. 0이 아닌 값을 사용하면 자동화된 동적 innodb_adaptive_max_sleep_delay 옵션에 지정된 최대 값까지 innodb_thread_sleep_delay 값을 조정할 수 있습니다. 이 값은 마이크로초를 나타냅니다. 이 옵션은 16개 이상의 InnoDB 스레드가 있는 바쁜 시스템에서 유용할 수 있습니다. (실제로는 수백 또는 수천 개의 동시 연결이 있는 MySQL 시스템에 가장 유용합니다.)

자세한 내용은 15.8.4절, 'InnoDB의 스레드 동시성 구성'을 참조하세요.

- innodb_api_bk_commit_interval

명령줄 형식	--innodb-api-bk-commit-interval=#
시스템 변수	innodb_api_bk_commit_interval
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	5

최소값	1
최대 값	1073741824
단위	초

InnoDB 멤캐시드 인터페이스를 사용하는 유휴 연결을 자동 커밋하는 빈도(초)입니다. 자세한 내용은 섹션 15.20.6.4, "InnoDB memcached 플러그인의 트랜잭션 동작 제어"를 참조하세요.

- INNODB_API_DISABLE_ROWLOCK

명령줄 형식	--innodb-api-disable-rowlock[={OFF|ON}]
시스템 변수	INNODB_API_DISABLE_ROWLOCK
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	부울
기본값	꺼짐

이 옵션을 사용하면 InnoDB 멤캐시가 DML 작업을 수행할 때 행 잠금을 비활성화할 수 있습니다. 기본적으로 innodb_api_disable_rowlock은 비활성화되어 있으므로, 멤캐시드는 get 및 set 연산에 대해 행 잠금을 요청합니다. innodb_api_disable_rowlock을 활성화하면, 멤캐시드는 행 잠금 대신 테이블 잠금을 요청합니다.

innodb_api_disable_rowlock은 동적이지 않습니다. mysqld 명령줄에서 지정하거나 MySQL 구성 파일에 입력해야 합니다. 구성은 플러그인이 설치될 때 적용되며, 이는 MySQL 서버가 시작될 때 발생합니다.

자세한 내용은 섹션 15.20.6.4, "InnoDB 멤캐시드 플러그인의 트랜잭션 동작 제어"를 참조하세요.

- innodb_api_enable_binlog

명령줄 형식	--innodb-api-enable-binlog[={OFF|ON}]
시스템 변수	innodb_api_enable_binlog
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	부울
기본값	꺼짐

MySQL 바이너리 로그와 함께 InnoDB 멤캐시드 플러그인을 사용할 수 있습니다. 자세한 내용은 InnoDB 멤캐시드 바이너리 로그 활성화를 참조하세요.

- INNODB_API_ENABLE_MDL

명령줄 형식	`--innodb-api-enable-mdl[={OFF|ON}]`
시스템 변수	`INNODB_API_ENABLE_MDL`
범위	글로벌
동적	아니요
`SET_VAR` 힌트 적용	아니요

유형	부울
기본값	꺼짐

InnoDB 멤캐시드 플러그인에서 사용하는 테이블을 잠궈서 SQL 인터페이스를 통해 DDL이 삭제하거나 변경할 수 없도록 합니다. 자세한 내용은 15.20.6.4절, "InnoDB 멤캐시드 플러그인의 트랜잭션 동작 제어"를 참조하세요.

- INNODB_API_TRX_LEVEL

명령줄 형식	--innodb-api-trx-level=#
시스템 변수	INNODB_API_TRX_LEVEL
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	0
최소값	0
최대 값	3

멤캐시드 인터페이스에서 처리되는 쿼리의 트랜잭션 격리 수준을 제어합니다. 익숙한 이름에 해당하는 상수는 다음과 같습니다:

- 0 = 커밋되지 않은 읽기
- 1 = 읽기 커밋
- 2 = 반복 읽기 가능
- 3 = 직렬화 가능

자세한 내용은 섹션 15.20.6.4, "InnoDB 멤캐시드 플러그인의 트랜잭션 동작 제어"를 참조하세요.

- innodb_autoextend_increase

명령줄 형식	--innodb-autoextend-increase=#
시스템 변수	innodb_autoextend_increase
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	64

최소값	1
최대 값	1000
단위	메가바이트

InnoDB 시스템 테이블스페이스 파일이 가득 차면 자동 확장되는 크기를 확장하기 위한 증분 크기(메가바이트)입니다. 기본값은 64입니다. 관련 정보는 시스템 테이블 스페이스 데이터 파일 구성 및 시스템 테이블 스페이스 크기 조정을 참조하십시오.

innodb_autoextend_increase 설정은 테이블별 파일 테이블 스페이스에 영향을 주지 않습니다. 파일 또는 일반 테이블스페이스 파일입니다. 이러한 파일은 파일 확장자가

innodb_autoextend_increment 설정. 초기 확장은 소량으로 이루어지며, 그 이후에는 4MB 단위로 확장이 이루어집니다.

- INNODB_AUTOINC_LOCK_MODE

명령줄 형식	--innodb-autoinc-lock-mode=#
시스템 변수	INNODB_AUTOINC_LOCK_MODE
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	2
유효한 값	0 1 2

자동 증가 값 생성에 사용할 잠금 모드입니다. 허용되는 값은 기존, 연속 또는 인터리브에 대해 각각 0, 1 또는 2입니다.

기본 설정은 행 기반 복제와의 호환성을 위해 2(인터리브)입니다.

각 잠금 모드의 특성에 대해서는 InnoDB AUTO_INCREMENT 잠금 모드를 참조하십시오.

- innodb_background_drop_list_empty

명령줄 형식	--innodb-background-drop-list-empty[={OFF|ON}]
시스템 변수	innodb_background_drop_list_empty
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	부울
기본값	꺼짐

innodb_background_drop_list_empty 디버그 옵션을 활성화하면 백그라운드 드롭 목록이 비워질 때까지 테이블 생성을 지연시켜 테스트 케이스 실패를 방지하는 데 도움이 됩니다. 예를 들어 테스트 케이스 A가 백그라운드 드롭 목록에 테이블 t1을 배치하면 테스트 케이스 B는 백그라운드 드롭 목록이 비워질 때까지 기다렸다가 테이블 t1을 생성합니다.

- innodb_buffer_pool_chunk_size

명령줄 형식	--innodb-버퍼-풀-청크-크기=#

InnoDB 시스템 변수

시스템 변수	`innodb_buffer_pool_chunk_size`
범위	글로벌
동적	아니요
`SET_VAR` 힌트 적용	아니요
유형	Integer
기본값	`134217728`
최소값	`1048576`

최대 값	innodb_버퍼_풀_사이즈 / innodb_버퍼_풀_인스턴스
단위	바이트

innodb_buffer_pool_chunk_size는 InnoDB 버퍼 풀 크기 조정 작업을 위한 청크 크기를 정의합니다.

크기 조정 작업 중에 모든 버퍼 풀 페이지가 복사되는 것을 방지하기 위해 작업은 "청크" 단위로 수행됩니다. 기본적으로 innodb_buffer_pool_chunk_size는 128MB(134217728 바이트)입니다. 청크에 포함된 페이지 수는 innodb_page_size의 값에 따라 달라집니다.
innodb_buffer_pool_chunk_size는 1MB(1048576바이트) 단위로 늘리거나 줄일 수 있습니다.

innodb_buffer_pool_chunk_size 값을 변경할 때는 다음 조건이 적용됩니다:

- innodb_버퍼_풀_청크_크기 * innodb_버퍼_풀_인스턴스
  가 버퍼 풀이 초기화될 때 현재 버퍼 풀 크기보다 크면 innodb_buffer_pool_chunk_size는 innodb_buffer_pool_size / innodb_buffer_pool_instances로 잘립니다.
- 버퍼 풀 크기는 항상 innodb_buffer_pool_chunk_size의 배수이거나 같아야 합니다.
  * innodb_buffer_pool_instances. innodb_buffer_pool_chunk_size를 변경하면 innodb_buffer_pool_size는 자동으로 innodb_buffer_pool_chunk_size * innodb_buffer_pool_instances와 같거나 그 배수인 값으로 반올림됩니다. 조정은 버퍼 풀이 초기화될 때 발생합니다.

중요

이 값을 변경하면 버퍼 풀의 크기가 자동으로 커질 수 있으므로 innodb_buffer_pool_chunk_size를 변경할 때 주의해야 합니다. innodb_buffer_pool_chunk_size를 변경하기 전에 innodb_buffer_pool_size에 미치는 영향을 계산하여 결과 버퍼 풀 크기가 허용 가능한지 확인합니다.

잠재적인 성능 문제를 방지하기 위해 청크 수(innodb_buffer_pool_size / innodb_buffer_pool_chunk_size)는 1000을 초과하지 않아야 합니다.

innodb_buffer_pool_size 변수는 동적이므로 서버가 온라인 상태인 동안 버퍼 풀의 크기를 조정할 수 있습니다. 그러나 버퍼 풀 크기는 다음과 같거나 그 배수여야 합니다.
innodb_버퍼풀_청크_크기 * innodb_버퍼풀_인스턴스이며, 이 변수 설정 중 하나를 변경하려면 서버를 다시 시작해야 합니다.

자세한 내용은 섹션 15.8.3.1, 'InnoDB 버퍼 풀 크기 구성'을 참조하세요.

- innodb_buffer_pool_debug

명령줄 형식	--innodb-buffer-pool-debug[={OFF|ON}]

시스템 변수	innodb_buffer_pool_debug
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	부울
기본값	꺼짐

이 옵션을 활성화하면 버퍼 풀의 크기가 1GB 미만인 경우 여러 버퍼 풀 인스턴스가 허용되며
, 다음과 같은 최소 버퍼 풀 크기 제약 조건은 무시됩니다.

innodb_buffer_pool_instances. innodb_buffer_pool_debug 옵션은 디버깅 지원이 WITH_DEBUG CMake 옵션을 사용하여 컴파일된 경우에만 사용할 수 있습니다.

- innodb_buffer_pool_dump_at_shutdown

명령줄 형식	--innodb-buffer-pool-dump-at-shutdown[={OFF|ON}]
시스템 변수	innodb_buffer_pool_dump_at_shutdown
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	부울
기본값	켜기

다음 재시작 시 워밍업 프로세스를 단축하기 위해 MySQL 서버가 종료될 때 InnoDB 버퍼 풀에 캐시된 페이지를 기록할지 여부를 지정합니다. 일반적으로 innodb_buffer_pool_load_at_startup과 함께 사용됩니다. innodb_buffer_pool_dump_pct 옵션은 덤프할 가장 최근에 사용된 버퍼 풀 페이지의 비율을 정의합니다.

innodb_buffer_pool_dump_at_shutdown 및 innodb_buffer_pool_load_at_startup은 기본적으로 활성화되어 있습니다.

자세한 내용은 섹션 15.8.3.6, '버퍼 풀 상태 저장 및 복원'을 참조하세요.

- innodb_buffer_pool_dump_now

명령줄 형식	--innodb-buffer-pool-dump-now[={OFF|ON}]
시스템 변수	innodb_buffer_pool_dump_now
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	부울
기본값	꺼짐

InnoDB 버퍼 풀에 캐시된 페이지를 즉시 기록합니다. 일반적으로 innodb_buffer_pool_load_now와 함께 사용됩니다.

innodb_buffer_pool_dump_now를 활성화하면 기록 작업이 트리거되지만 변수 설정은 변경되지 않으며, 항상 OFF 또는 0으로 유지됩니다. 덤프 트리거 후 버퍼 풀 덤프 상태를 보려면 Innodb_buffer_pool_dump_status 변수를 쿼리합니다.

innodb_buffer_pool_dump_now를 활성화하면 덤프 작업이 트리거되지만 변수 설정은 변경되지 않으며, 항상 OFF 또는 0으로 유지됩니다. 덤프 트리거 후 버퍼 풀 덤프 상태를 보려면 Innodb_buffer_pool_dump_status 변수를 쿼리합니다.

자세한 내용은 섹션 15.8.3.6, '버퍼 풀 상태 저장 및 복원'을 참조하세요.

- INNODB_BUFFER_POOL_DUMP_PCT

명령줄 형식	--innodb-버퍼-풀-덤프-팩트=#
시스템 변수	INNODB_BUFFER_POOL_DUMP_PCT
범위	글로벌
동적	예

SET_VAR 힌트 적용	아니요
유형	Integer
기본값	25
최소값	1
최대 값	100

각 버퍼 풀에서 가장 최근에 사용된 페이지를 읽고 덤프할 페이지의 비율을 지정합니다. 범위는 1~100입니다. 기본값은 25입니다. 예를 들어, 각각 100페이지씩 4개의 버퍼 풀이 있고 innodb_buffer_pool_dump_pct가25로 설정되어 있으면 각 버퍼 풀에서 가장 최근에 사용된 25페이지가 덤프됩니다.

- innodb_buffer_pool_파일명

명령줄 형식	--innodb-buffer-pool-파일명=파일명
시스템 변수	innodb_buffer_pool_파일명
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	파일 이름
기본값	ib_buffer_pool

innodb_buffer_pool_dump_at_shutdown 또는 innodb_buffer_pool_dump_now에서 생성된 테이블스페이스 ID 및 페이지 ID 목록을 보관하는 파일 이름을 지정합니다. 테이블스페이스 ID와 페이지 ID는 space, page_id 형식으로 저장됩니다. 기본적으로 파일 이름은 ib_buffer_pool이며 InnoDB 데이터 디렉터리에 위치합니다. 기본 위치가 아닌 다른 위치는 데이터 디렉터리를 기준으로 지정해야 합니다.

파일 이름은 런타임에 SET 문을 사용하여 지정할 수 있습니다:

```
SET GLOBAL innodb_buffer_pool_filename='file_name';
```

시작 시, 시작 문자열 또는 MySQL 구성 파일에서 파일 이름을 지정할 수도 있습니다. 시작 시 파일 이름을 지정할 때는 파일이 존재해야 하며, 그렇지 않으면 InnoDB에서 해당 파일 또는 디렉터리가 없다는 시작 오류를 반환합니다.

자세한 내용은 섹션 15.8.3.6, '버퍼 풀 상태 저장 및 복원'을 참조하세요.

- INNODB_BUFFER_POOL_IN_CORE_FILE

명령줄 형식	--innodb-buffer-pool-in-core-file[={OFF|ON}]
시스템 변수	INNODB_BUFFER_POOL_IN_CORE_FILE

범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	부울
기본값	켜기

innodb_buffer_pool_in_core_file 변수를 비활성화하면 InnoDB 버퍼 풀 페이지를 제외하여 코어 파일의 크기를 줄일 수 있습니다. 이 변수를 사용하려면 core_file 변수가 활성화되어 있어야 하며 운영 체제에서 madvise() 에 대한 MADV_DONTDUMP 비-POSIX 확장자를 지원해야 합니다,

은 Linux 3.4 이상에서 지원됩니다. 자세한 내용은 섹션 15.8.3.7, '코어 파일에서 버퍼 풀 페이지 제외하기'를 참조하세요.

- innodb_buffer_pool_instances

명령줄 형식	--innodb-버퍼-풀-인스턴스=#
시스템 변수	innodb_buffer_pool_instances
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	Integer
기본값(Windows, 32비트 플랫폼)	(자동 크기 조정)
기본값(기타)	8(또는 innodb_buffer_pool_size < 1GB인 경우 1)
최소값	1
최대 값	64

InnoDB 버퍼 풀이 분할되는 영역의 수입니다. 버퍼 풀이 수 기가바이트 범위인 시스템의 경우, 버퍼 풀을 별도의 인스턴스로 나누면 여러 스레드가 캐시된 페이지를 읽고 쓸 때 경합을 줄여 동시성을 향상시킬 수 있습니다. 버퍼 풀에 저장되거나 버퍼 풀에서 읽혀지는 각 페이지는 해싱 함수를 사용하여 버퍼 풀 인스턴스 중 하나에 무작위로 할당됩니다. 각 버퍼 풀은 자체 프리 리스트, 플러시 리스트, LRU 및 버퍼 풀에 연결된 기타 모든 데이터 구조를 관리하며 자체 버퍼 풀 뮤텍스에 의해 보호됩니다.

이 옵션은 innodb_buffer_pool_size를 1GB 이상으로 설정할 때만 적용됩니다. 총 버퍼 풀 크기는 모든 버퍼 풀로 나뉩니다. 최상의 효율성을 위해 각 버퍼 풀 인스턴스가 최소 1GB가 되도록 innodb_buffer_pool_instances와 innodb_buffer_pool_size의 조합을 지정합니다.

32비트 Windows 시스템의 기본값은 다음 값에 따라 달라집니다.
아래에 설명된 대로 innodb_buffer_pool_size를 설정합니다:

- innodb_buffer_pool_size가 1.3GB보다 큰 경우, innodb_buffer_pool_instances의 기본값은 각 청크에 대한 개별 메모리 할당 요청과 함께 innodb_buffer_pool_size/128MB입니다. 1.3GB는 32비트 Windows에서 단일 버퍼 풀에 필요한 연속 주소 공간을 할당하지 못할 위험이 큰 경계로 선택되었습니다.
- 그렇지 않으면 기본값은 1입니다.

다른 모든 플랫폼에서 innodb_buffer_pool_size가 1GB 이상인 경우 기본값은 8입니다. 그렇지 않으면 기본값은 1입니다.

관련 정보는 섹션 15.8.3.1, "InnoDB 버퍼 풀 크기 구성"을 참조하세요.

- innodb_buffer_pool_load_abort

명령줄 형식	--innodb-buffer-pool-load-abort[={OFF|ON}]
시스템 변수	innodb_buffer_pool_load_abort
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	부울

기본값	꺼짐

다음에 의해 트리거된 InnoDB 버퍼 풀 콘텐츠 복원 프로세스를 중단합니다. innodb_buffer_pool_load_at_startup 또는 innodb_buffer_pool_load_now.

innodb_buffer_pool_load_abort를 활성화하면 중단 작업이 트리거되지만 변수 설정은 변경되지 않으며, 항상 OFF 또는 0으로 유지됩니다. 중단 작업을 트리거한 후 버퍼 풀 로드 상태를 보려면 Innodb_buffer_pool_load_status 변수를 쿼리합니다.

자세한 내용은 섹션 15.8.3.6, '버퍼 풀 상태 저장 및 복원'을 참조하세요.

- innodb_buffer_pool_load_at_startup

명령줄 형식	--innodb-buffer-pool-load-at-startup[={OFF|ON}]
시스템 변수	innodb_buffer_pool_load_at_startup
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	부울
기본값	켜기

MySQL 서버가 시작될 때 이전에 보유했던 동일한 페이지를 로드하여 InnoDB 버퍼 풀을 자동으로 워밍업하도록 지정합니다. 일반적으로 innodb_buffer_pool_dump_at_shutdown과 함께 사용됩니다.

innodb_buffer_pool_dump_at_shutdown 및 innodb_buffer_pool_load_at_startup은 기본적으로 활성화되어 있습니다.

자세한 내용은 섹션 15.8.3.6, '버퍼 풀 상태 저장 및 복원'을 참조하세요.

- INNODB_BUFFER_POOL_LOAD_NOW

명령줄 형식	--innodb-buffer-pool-load-now[={OFF|ON}]
시스템 변수	INNODB_BUFFER_POOL_LOAD_NOW
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	부울
기본값	꺼짐

서버 재시작을 기다리지 않고 데이터 페이지를 로드하여 InnoDB 버퍼 풀을 즉시 워밍업합니다. 벤치마킹 중에 캐시 메모리를 알려진 상태로 되돌리거나 보고서 또는 유지 관리를 위해 쿼리를 실행한 후 MySQL 서버가 정상적인 워크로드를 재개할 수 있도록 준비하는 데 유용할 수 있습니다.

innodb_buffer_pool_load_now를 활성화하면 로드 작업이 트리거되지만 변수 설정은 변경되지 않으며, 항상 OFF 또는 0으로 유지됩니다. 로드를 트리거한 후 버퍼 풀 로드 진행률을 보려면 Innodb_buffer_pool_load_status 변수를 쿼리합니다.

자세한 내용은 섹션 15.8.3.6, '버퍼 풀 상태 저장 및 복원'을 참조하세요.

- innodb_buffer_pool_size

명령줄 형식	--innodb-버퍼-풀-크기=#

시스템 변수	innodb_buffer_pool_size
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	134217728
최소값	5242880
최대값(64비트 플랫폼)	2**64-1
최대값(32비트 플랫폼)	2**32-1
단위	바이트

InnoDB가 테이블 및 인덱스 데이터를 캐시하는 메모리 영역인 버퍼 풀의 크기(바이트)입니다. 기본값은 134217728 바이트(128MB)입니다. 최대 값은 CPU에 따라 다릅니다.
아키텍처에서 최대값은 32비트 시스템에서 4294967295 (2^{32} -1), 64비트 시스템에서 18446744073709551615 (2^{64} -1)입니다. 32비트 시스템에서는 CPU 아키텍처 및 운영 체제에 따라 명시된 최대 크기보다 실제 최대 크기가 더 낮을 수 있습니다. 버퍼 풀의 크기가 1GB보다 큰 경우 innodb_buffer_pool_instances를 1보다 큰 값으로 설정하면 사용량이 많은 서버에서 확장성을 향상시킬 수 있습니다.

버퍼 풀이 클수록 동일한 테이블 데이터에 두 번 이상 액세스하는 데 필요한 디스크 I/O가 줄어듭니다. 전용 데이터베이스 서버에서는 버퍼 풀 크기를 시스템 물리적 메모리 크기의 80%로 설정할 수 있습니다. 버퍼 풀 크기를 구성할 때 다음과 같은 잠재적인 문제에 유의하고 필요한 경우 버퍼 풀의 크기를 축소할 준비를 하세요.

- 물리적 메모리를 차지하기 위한 경쟁으로 인해 운영 체제에서 페이징이 발생할 수 있습니다.
- InnoDB는 버퍼 및 제어 구조를 위해 추가 메모리를 예약하므로 총 할당된 공간은 지정된 버퍼 풀 크기보다 약 10% 더 큽니다.
- 버퍼 풀의 주소 공간은 연속적이어야 하며, 이는 특정 주소에서 로드되는 DLL이 있는 Windows 시스템에서 문제가 될 수 있습니다.
- 버퍼 풀을 초기화하는 데 걸리는 시간은 버퍼 풀의 크기에 대략 비례합니다. 버퍼 풀이 큰 인스턴스에서는 초기화 시간이 상당히 길어질 수 있습니다. 초기화 기간을 줄이려면 서버 종료 시 버퍼 풀 상태를 저장하고 서버 시작 시 복원할 수 있습니다. 15.8.3.6절. "버퍼 풀 상태 저장 및 복원"을 참조하세요.

버퍼 풀 크기를 늘리거나 줄일 때 작업은 청크 단위로 수행됩니다. 청크 크기는 innodb_buffer_pool_chunk_size 변수에 의해 정의되며, 기본값은 128MB입니다.

버퍼 풀 크기는 항상 innodb_buffer_pool_chunk_size *

innodb_buffer_pool_instances의 배수이거나 같아야 합니다. 버퍼 풀 크기를 innodb_buffer_pool_chunk_size * innodb_buffer_pool_instances의 배수나 같지 않은 값으로 변경하면 버퍼 풀 크기는 자동으로 innodb_buffer_pool_chunk_size * innodb_buffer_pool_instances의 배수 또는 같은 값으로 조정됩니다.

innodb_buffer_pool_size를 동적으로 설정할 수 있으므로 서버를 다시 시작하지 않고도 버퍼 풀의 크기를 조정할 수 있습니다. Innodb_buffer_pool_resize_status 상태 변수는 온라인 버퍼 풀 크기 조정 작업의 상태를 보고합니다. 자세한 내용은 15.8.3.1절, "InnoDB 버퍼 풀 크기 구성"을 참조하세요.

innodb_dedicated_server가 활성화된 경우, 명시적으로 정의되지 않은 경우 innodb_buffer_pool_size 값이 자동으로 구성됩니다. 자세한 내용은 15.8.12절, "전용 MySQL 서버에 대한 자동 구성 활성화"를 참조하세요.

- innodb_change_buffer_max_size

명령줄 형식	--innodb-change-buffer-max-size=#
시스템 변수	innodb_change_buffer_max_size
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	25
최소값	0
최대 값	50

버퍼 풀의 총 크기 대비 백분율로 표시되는 InnoDB 변경 버퍼의 최대 크기입니다. 삽입, 업데이트 및 삭제 활동이 많은 MySQL 서버의 경우 이 값을 늘리거나 보고에 사용되는 데이터가 변경되지 않는 MySQL 서버의 경우 이 값을 줄일 수 있습니다. 자세한 내용은 섹션 15.5.2, "버퍼 변경"을 참조하세요. 일반적인 I/O 튜닝에 대한 조언은 8.5.8절. "InnoDB 디스크 I/O 최적화"를 참조하세요.

- innodb_change_buffering

명령줄 형식	--innodb-change-buffering=값
시스템 변수	innodb_change_buffering
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	열거형
기본값	모두
유효한 값	없음 삽입 삭제 변경 사항 삭제 제거 모두

I/O 작업이 순차적으로 수행될 수 있도록 보조 인덱스에 대한 쓰기 작업을 지연시키는 최적화 기능인 변경 버퍼링을 InnoDB가 수행할지 여부입니다. 허용되는 값은 다음 표에 설명되어 있습니다. 값을 숫자로 지정할

수도 있습니다.

표 15.25 innodb_change_buffering에 허용되는 값

가치	숫자 값	설명
없음	0	어떤 작업도 버퍼링하지 마세요.
인서트	1	버퍼 삽입 작업.
삭제	2	버퍼 삭제 마킹 작업, 엄밀히 말하면 인덱스 레코드를 마킹하는 쓰기입니다.

가치	숫자 값	설명
		퍼지 중 나중에 삭제할 수 있도록 작동합니다.
변경 사항	3	버퍼 삽입 및 삭제 표시 작업.
퍼지	4	백그라운드에서 발생하는 물리적 삭제 작업을 버퍼링합니다.
모두	5	기본값입니다. 버퍼 삽입, 삭제 표시 작업 및 제거.

자세한 내용은 섹션 15.5.2, "버퍼 변경"을 참조하십시오. 일반적인 I/O 튜닝에 대한 조언은 8.5.8절. "InnoDB 디스크 I/O 최적화"를 참조하세요.

- `innodb_change_buffering_debug`

명령줄 형식	`--innodb-change-buffering-debug=#`
시스템 변수	`innodb_change_buffering_debug`
범위	글로벌
동적	예
`SET_VAR` 힌트 적용	아니요
유형	Integer
기본값	`0`
최소값	`0`
최대 값	`2`

`InnoDB` 변경 버퍼링에 대한 디버그 플래그를 설정합니다. 값이 1이면 모든 변경 내용을 변경 버퍼에 강제로 저장합니다. 값이 2이면 병합 시 예기치 않은 종료가 발생합니다. 기본값 0은 변경 버퍼링 디버그 플래그가 설정되어 있지 않음을 나타냅니다. 이 옵션은 디버깅 지원이 `WITH_DEBUG CMake` 옵션을 사용하여 컴파일된 경우에만 사용할 수 있습니다.

- `innodb_checkpoint_disabled`

명령줄 형식	`--innodb-checkpoint-disabled[={OFF|ON}]`
시스템 변수	`innodb_checkpoint_disabled`
범위	글로벌
동적	예
`SET_VAR` 힌트 적용	아니요
유형	부울
기본값	꺼짐

이 옵션은 전문가 디버깅용으로만 사용되는 디버그 옵션입니다. 이 옵션은 체크포인트를 비활성화하여 의

도적인 서버 종료 시 항상 `InnoDB` 복구가 시작되도록 합니다. 일반적으로 서버 종료 후 복구가 필요한 재실행 로그 항목을 작성하는 DML 작업을 실행하기 전에 짧은 간격으로만 활성화해야 합니다. 이 옵션은 디버깅 지원이 `WITH_DEBUG CMake` 옵션을 사용하여 컴파일된 경우에만 사용할 수 있습니다.

- `innodb_checksum_algorithm`

명령줄 형식	`--innodb-checksum-algorithm=value`

시스템 변수	innodb_checksum_알고리즘
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	열거형
기본값	crc32
유효한 값	CRC32 STRICT_CRC32 INNODB STRICT_INNODB 없음 strict_none

InnoDB 테이블스페이스의 디스크 블록에 저장된 체크섬을 생성하고 검증하는 방법을 지정합니다. innodb_checksum_algorithm의 기본값은 crc32입니다.

innodb 값은 이전 버전의 MySQL과 역호환됩니다. crc32 값은 수정된 모든 블록에 대한 체크섬을 계산하고 각 디스크 읽기에 대한 체크섬을 검사하는 데 더 빠른 알고리즘을 사용합니다. 이 알고리즘은 한 번에 64비트 블록을 검사하므로 한 번에 8비트 블록을 검사하는 innodb 체크섬 알고리즘보다 빠릅니다. none 값은 블록 데이터를 기반으로 값을 계산하는 대신 체크섬 필드에 상수 값을 씁니다. 의 블록은 테이블스페이스는 이전, 새 및 체크섬 값이 없는 값을 혼합하여 사용할 수 있으며 데이터가 수정됨에 따라 점진적으로 업데이트됩니다. 테이블스페이스의 블록이 crc32 알고리즘을 사용하도록 수정되면 이전 버전의 MySQL에서는 관련 테이블을 읽을 수 없습니다.

엄격한 형태의 체크섬 알고리즘은 테이블스페이스에서 유효하지만 일치하지 않는 체크섬 값을 발견하면 오류를 보고합니다. 테이블스페이스를 처음 설정할 때는 새 인스턴스에서만 엄격한 설정을 사용하는 것이 좋습니다. 엄격한 설정은 디스크 읽기 중에 모든 체크섬 값을 계산할 필요가 없으므로 다소 빠릅니다.

다음 표는 none, innodb 및 crc32 옵션 값과 엄격한 옵션 값의 차이점을 보여줍니다. none, innodb 및 crc32는 각 데이터 블록에 지정된 유형의 체크섬 값을 기록하지만 호환성을 위해 검증할 때 다른 체크섬 값을 허용합니다.
읽기 작업 중 블록을 확인합니다. 엄격한 설정은 유효한 체크섬 값도 허용하지만, 일치하지 않는 유효한 체크섬 값이 발견되면 오류 메시지를 출력합니다. 엄격한 형식을 사용하면 인스턴스의 모든 InnoDB 데이터 파일이 동일한 innodb_checksum_algorithm 값으로 생성되는 경우 검증을 더 빠르게 수행할 수 있습니다.

표 15.26 허용된 innodb_checksum_알고리즘 값

가치	생성된 체크섬(쓰기 시)	허용된 체크섬(읽기 시)
없음	상수입니다.	`none`, `innodb` 또는 `crc32`에서 생성된 체크섬 중 하나.
innodb	`InnoDB`의 원본 알고리즘을 사용하여 소프트웨어에서 계산된 체크섬입니다.	`none`, `innodb` 또는 `crc32`에서 생성된 체크섬 중 하나.
crc32	하드웨어 지원을 통해 계산된 `crc32` 알고리즘을 사용하여 계산된 체크섬입니다.	`none`, `innodb` 또는 `crc32`에서 생성된 체크섬 중 하나.

가치	생성된 체크섬(쓰기 시)	허용된 체크섬(읽기 시)
strict_none	상수	none, innodb 또는 crc32 에서 생성된 체크섬 중 하나. 유효하지만 일치하지 않는 체크섬이 발견되면 InnoDB는 오류 메시지를 인쇄합니다.
strict_innodb	InnoDB의 원본 알고리즘을 사용하여 소프트웨어에서 계산된 체크섬입니다.	none, innodb 또는 crc32 에서 생성된 체크섬 중 하나. 유효하지만 일치하지 않는 체크섬이 발견되면 InnoDB는 오류 메시지를 인쇄합니다.
strict_crc32	하드웨어 지원을 통해 계산된 crc32 알고리즘을 사용하여 계산된 체크섬입니다.	none, innodb 또는 crc32 에서 생성된 체크섬 중 하나. 유효하지만 일치하지 않는 체크섬이 발견되면 InnoDB는 오류 메시지를 인쇄합니다.

- innodb_cmp_per_index_enabled

명령줄 형식	--innodb-cmp-per-index-enabled[={OFF|ON}]
시스템 변수	innodb_cmp_per_index_enabled
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	부울
기본값	꺼짐

정보 스키마 INNODB_CMP_PER_INDEX 테이블에서 인덱스별 압축 관련 통계를 사용하도록 설정합니다. 이러한 통계는 수집하는 데 비용이 많이 들 수 있으므로 InnoDB 압축 테이블과 관련된 성능 튜닝 중에 개발, 테스트 또는 복제본 인스턴스에서만 이 옵션을 활성화하세요.

자세한 내용은 26.4.8절, "정보 스키마 INNODB_CMP_PER_INDEX 및 INNODB_CMP_PER_INDEX_RESET 테이블" 및 15.9.1.4절, "런타임에 InnoDB 테이블 압축 모니터링"을 참조하세요.

- innodb_commit_concurrency

명령줄 형식	--innodb-commit-concurrency=#

시스템 변수	innodb_commit_concurrency
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	0
최소값	0

최대 값	1000

동시에 커밋할 수 있는 스레드 수입니다. 기본값인 0은 원하는 수의 트랜잭션이 동시에 커밋할 수 있도록 허용합니다.

innodb_commit_concurrency의 값은 런타임에 0에서 0이 아닌 값으로 또는 그 반대로 변경할 수 없습니다. 이 값은 0이 아닌 값에서 다른 값으로 변경할 수 있습니다.

- innodb_compress_debug

명령줄 형식	--innodb-compress-debug=value
시스템 변수	innodb_compress_debug
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	열거형
기본값	없음
유효한 값	없음 ZLIB LZ4 lz4hc

각 테이블에 대해 COMPRESSION 속성을 정의할 필요 없이 지정된 압축 알고리즘을 사용하여 모든 테이블을 압축합니다. 이 옵션은 디버깅 지원이 WITH_DEBUG CMake 옵션을 사용하여 컴파일된 경우에만 사용할 수 있습니다.

관련 정보는 섹션 15.9.2, "InnoDB 페이지 압축"을 참조하세요.

- INNODB_COMPRESSION_FAILURE_THRESHOLD_PCT

명령줄 형식	--innodb-compression-failure-threshold-pct=#
시스템 변수	innodb_compression_failure_threshold_pct
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	5
최소값	0
최대값	100

테이블의 압축 실패율 임계값을 백분율로 정의하며, 이 임계값을 초과하면 MySQL이 압축 페이지 내에 패딩을 추가하기 시작하여 비용이 많이 드는 압축 실패를 방지합니다. 이 임계값을 통과하면 MySQL은 각각

의 새 압축 페이지 내에 추가 여유 공간을 남겨두기 시작하여 innodb_compression_pad_pct_max에 지정된 페이지 크기의 백분율까지 여유 공간의 양을 동적으로 조정합니다. 값이 0이면 압축 효율을 모니터링하고 패딩 양을 동적으로 조정하는 메커니즘이 비활성화됩니다.

자세한 내용은 섹션 15.9.1.6, "OLTP 워크로드용 압축"을 참조하십시오.

- innodb_compression_level

명령줄 형식	--innodb-compression-level=#
시스템 변수	innodb_compression_level
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	6
최소값	0
최대 값	9

InnoDB 압축 테이블 및 인덱스에 사용할 zlib 압축 수준을 지정합니다. 값이 클수록 압축 중에 CPU 오버헤드가 증가하지만 저장 장치에 더 많은 데이터를 저장할 수 있습니다. 저장 공간이 중요하지 않거나 데이터를 특별히 압축할 필요가 없을 것으로 예상되는 경우 값을 낮추면 CPU 오버헤드를 줄일 수 있습니다.

자세한 내용은 섹션 15.9.1.6, "OLTP 워크로드용 압축"을 참조하십시오.

- INNODB_COMPRESSION_PAD_PCT_MAX

명령줄 형식	--innodb-compression-pad-pct-max=#
시스템 변수	INNODB_COMPRESSION_PAD_PCT_MAX
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	50
최소값	0
최대 값	75

압축된 테이블 또는 인덱스가 업데이트되어 데이터가 다시 압축될 수 있을 때 페이지 내에서 데이터 및 수정 로그를 재구성할 수 있는 공간을 확보하기 위해 각 압축 페이지 내에서 여유 공간으로 예약할 수 있는 최대 비율을 지정합니다. innodb_compression_failure_threshold_pct가 0이 아닌 값으로 설정되어 있고 압축 실패율이 커트라인을 통과하는 경우에만 적용됩니다.

자세한 내용은 섹션 15.9.1.6, "OLTP 워크로드용 압축"을 참조하십시오.

- innodb_concurrency_tickets

명령줄 형식	--innodb-concurrency-tickets=#
시스템 변수	innodb_concurrency_tickets

범위	글로벌
동적	예
`SET_VAR` 힌트 적용	아니요
유형	Integer
기본값	`5000`
최소값	`1`

최대 값	4294967295

InnoDB에 동시에 들어갈 수 있는 스레드 수를 결정합니다. 스레드 수가 이미 동시성 제한에 도달한 경우 InnoDB에 들어가려고 할 때 스레드가 대기열에 배치됩니다.
스레드에 InnoDB 진입이 허용되면 innodb_concurrency_tickets 값과 동일한 수의 "티켓"이 주어지며, 스레드는 티켓을 모두 소진할 때까지 InnoDB에 자유롭게 출입할 수 있습니다. 그 시점 이후에는 스레드가 다음에 InnoDB에 들어가려고 할 때 다시 동시성 검사(및 대기열 가능성)의 대상이 됩니다. 기본값은 5000입니다.

innodb_concurrency_tickets 값이 작으면 몇 개의 행만 처리하면 되는 소규모 트랜잭션이 많은 행을 처리하는 대규모 트랜잭션과 공정하게 경쟁하게 됩니다. 단점
값이 작으면 대규모 트랜잭션이 완료되기 전에 대기열을 여러 번 반복해야 하므로 작업을 완료하는 데 필요한 시간이 길어집니다.

innodb_concurrency_tickets 값이 크면 대규모 트랜잭션은 대기열 끝에서 위치를 기다리는 데 소요되는 시간이 줄어들고(innodb_thread_concurrency로 제어됨) 행을 검색하는 데 더 많은 시간이 소요됩니다. 또한 대규모 트랜잭션은 작업을 완료하기 위해 큐를 통과하는 횟수가 적습니다. 큰 innodb_concurrency_tickets 값의 단점은 동시에 실행되는 큰 트랜잭션이 너무 많으면 작은 트랜잭션이 실행되기 전에 더 오랜 시간 대기하여 고갈될 수 있다는 것입니다.

innodb_thread_concurrency 값이 0이 아닌 경우, 더 큰 트랜잭션과 더 작은 트랜잭션 사이의 최적의 균형을 찾기 위해 innodb_concurrency_tickets 값을 위 또는 아래로 조정해야 할 수 있습니다. 엔진 INNODB 상태 표시 보고서에는 현재 대기열을 통과하는 트랜잭션의 실행에 남은 티켓 수가 표시됩니다. 이 데이터는 정보 스키마 INNODB_TRX 테이블의 TRX_CONCURRENCY_TICKETS 열에서도 얻을 수 있습니다.

자세한 내용은 15.8.4절, 'InnoDB의 스레드 동시성 구성'을 참조하세요.

- INNODB_DATA_FILE_PATH

명령줄 형식	--innodb-데이터-파일-경로=파일_이름
시스템 변수	INNODB_DATA_FILE_PATH
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	문자열
기본값	ibdata1:12M:자동 확장

InnoDB 시스템 테이블스페이스 데이터 파일의 이름, 크기 및 속성을 정의합니다.
innodb_data_file_path에 값을 지정하지 않으면 기본 동작은 12MB보다 약간 큰 단일 자동 확장 데

이터 파일(ibdata1)을 생성하는 것입니다.

데이터 파일 사양의 전체 구문에는 파일 이름, 파일 크기, 자동 확장 속성 및 최대 속성이 포함됩니다:

```
file_name:file_size[:autoextend[:max:max_file_size]]
```

파일 크기는 크기 값에 K, M 또는 G를 추가하여 킬로바이트, 메가바이트 또는 기가바이트로 지정합니다. 데이터 파일 크기를 킬로바이트 단위로 지정하는 경우 1024의 배수로 지정합니다. 그렇지 않으면 KB 값은 가장 가까운 메가바이트(MB) 경계로 반올림됩니다. 파일 크기의 합은 최소한 12MB보다 약간 커야 합니다.

추가 구성 정보는 시스템 테이블스페이스 데이터 파일 구성을 참조하십시오. For
크기 조정 지침은 시스템 테이블 스페이스 크기 조정을 참조하세요.

- INNODB_DATA_HOME_DIR

명령줄 형식	--innodb-데이터-홈-디렉터=디렉터_이름
시스템 변수	INNODB_DATA_HOME_DIR
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	디렉토리 이름

InnoDB 시스템 테이블스페이스 데이터 파일의 디렉터리 경로의 공통 부분입니다. 기본값은 MySQL 데이터 디렉터리입니다. 이 설정은 절대 경로로 정의되지 않는 한 innodb_data_file_path 설정과 연결됩니다 .

innodb_data_home_dir의 값을 지정할 때는 후행 슬래시가 필요합니다. 예를 들어

```
[mysqld]
innodb_data_home_dir = /path/to/myibdata/
```

이 설정은 테이블별 파일 테이블 스페이스의 위치에는 영향을 주지 않습니다.

관련 정보는 섹션 15.8.1, "InnoDB 시동 구성"을 참조하세요.

- INNODB_DDL_버퍼_크기

명령줄 형식	--innodb-ddl-buffer-size=#
시스템 변수	INNODB_DDL_버퍼_크기
범위	글로벌, 세션
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	1048576
최소값	65536
최대 값	4294967295
단위	바이트

DDL 작업의 최대 버퍼 크기를 정의합니다. 기본 설정은 1048576바이트(약 1MB)입니다. 보조 인덱스를 생성하거나 재구성하는 온라인 DDL 작업에 적용됩니다. 섹션 15.12.4, "온라인 DDL 메모리 관리"를 참조하십시오. DDL 스레드당 최대 버퍼 크기는 최대 버퍼 크기를 DDL 스레드 수로 나눈 값입니다(innodb_ddl_buffer_size/innodb_ddl_threads).

- innodb_ddl_log_crash_reset_debug

명령줄 형식	`--innodb-ddl-log-crash-reset-debug[={OFF|ON}]`
시스템 변수	`innodb_ddl_log_crash_reset_debug`
범위	글로벌
동적	예
`SET_VAR` 힌트 적용	아니요
유형	부울

기본값	꺼짐

이 디버그 옵션을 활성화하면 DDL 로그 충돌 주입 카운터를 1로 재설정할 수 있습니다. 이 옵션은 디버깅 지원이 `WITH_DEBUG CMake` 옵션을 사용하여 컴파일된 경우에만 사용할 수 있습니다.

- `innodb_ddl_threads`

명령줄 형식	`--innodb-ddl-threads=#`
시스템 변수	`innodb_ddl_threads`
범위	글로벌, 세션
동적	예
`SET_VAR` 힌트 적용	아니요
유형	Integer
기본값	`4`
최소값	`1`
최대 값	`64`

인덱스 생성의 정렬 및 빌드 단계에 대한 최대 병렬 스레드 수를 정의합니다. 보조 인덱스를 만들거나 다시 빌드하는 온라인 DDL 작업에 적용됩니다. 관련 정보는 섹션 15.12.5, "온라인 DDL 작업을 위한 병렬 스레드 구성" 및 섹션 15.12.4, "온라인 DDL 메모리 관리"를 참조하십시오.

- `innodb_deadlock_detect`

명령줄 형식	`--innodb-deadlock-detect[={OFF|ON}]`
시스템 변수	`innodb_deadlock_detect`
범위	글로벌
동적	예
`SET_VAR` 힌트 적용	아니요
유형	부울
기본값	켜기

이 옵션은 교착 상태 감지를 비활성화하는 데 사용됩니다. 동시성이 높은 시스템에서 교착 상태 감지는 수많은 스레드가 동일한 잠금을 기다릴 때 속도 저하를 유발할 수 있습니다. 때로는 교착 상태 감지를 비활성화하고 교착 상태 발생 시 트랜잭션 롤백을 위해 `innodb_lock_wait_timeout` 설정에 의존하는 것이 더 효율적일 수 있습니다.

관련 정보는 섹션 15.7.5.2, "교착 상태 감지"를 참조하세요.

- `innodb_dedicated_server`

명령줄 형식	`--innodb-dedicated-server[={OFF|ON}]`

시스템 변수	innodb_dedicated_server
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	부울
기본값	꺼짐

innodb_dedicated_server를 활성화하면 InnoDB는 다음 변수를 자동으로 구성합니다:

- innodb_buffer_pool_size
- innodb_redo_log_capacity.

참고

innodb_log_file_size 및 innodb_log_files_in_group은 더 이상 사용되지 않으며 innodb_redo_log_capacity로 대체됩니다(섹션 15.6.5, "로그 재실행" 참조).

- innodb_flush_method

MySQL 인스턴스가 사용 가능한 모든 시스템 리소스를 사용할 수 있는 전용 서버에 상주하는 경우에만 innodb_dedicated_server를 활성화하는 것을 고려하세요. MySQL 인스턴스가 다른 애플리케이션과 시스템 리소스를 공유하는 경우 innodb_dedicated_server를 활성화하는 것은 권장되지 않습니다.

자세한 내용은 섹션 15.8.12, '전용 MySQL 서버에 대한 자동 구성 활성화'를 참조하세요.

- INNODB_기본_행_형식

명령줄 형식	--innodb-default-row-format=value
시스템 변수	INNODB_기본_행_형식
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	열거형
기본값	다이나믹
유효한 값	리던던트 컴팩트 다이나믹

innodb_default_row_format 옵션은 InnoDB 테이블 및 사용자가 만든 임시 테이블의 기본 행 형식을 정의합니다. 기본 설정은 DYNAMIC입니다. 허용되는 다른 값으로는 COMPACT 및 REDUNDANT가 있습니다. 시스템 테이블 스페이스에서 사용하도록 지원되지 않는 COMPRESSED 행 형식은 기본값으로 정의할 수 없습니다.

새로 생성된 테이블은 다음과 같은 경우 innodb_dault_row_format에 정의된 행 형식을 사용합니다. ROW_FORMAT 옵션이 명시적으로 지정되지 않았거나 ROW_FORMAT=DEFAULT가 사용된 경우.

ROW_FORMAT 옵션을 명시적으로 지정하지 않거나 ROW_FORMAT=DEFAULT를사용하는 경우 테이블을 재구축하는 모든 작업은 테이블의 행 형식을 innodb_default_row_format에 정의된 형식으로 자동 변경합니다. 자세한 내용은 테이블의 행 형식 정의하기를 참조하세요.

쿼리를 처리하기 위해 서버에서 생성한 내부 `InnoDB` 임시 테이블은 `innodb_dault_row_format` 설정에 관계없이 `DYNAMIC` 행 형식을 사용합니다.

- `innodb_directories`

명령줄 형식	`--innodb-디렉토리=디알_이름`
시스템 변수	`innodb_directories`
범위	글로벌
동적	아니요

SET_VAR 힌트 적용	아니요
유형	디렉토리 이름
기본값	NULL

시작 시 테이블스페이스 파일을 검색할 디렉터리를 정의합니다. 이 옵션은 서버가 오프라인 상태에서 테이블스페이스 파일을 새 위치로 이동하거나 복원할 때 사용됩니다. 또한 절대 경로를 사용하여 생성되었거나 데이터 디렉터리 외부에 있는 테이블스페이스 파일의 디렉터리를 지정하는 데에도 사용됩니다.

충돌 복구 중 테이블스페이스 검색은 innodb_directories 설정을 사용하여 재실행 로그에서 참조된 테이블스페이스를 식별합니다. 자세한 내용은 충돌 복구 중 테이블스페이스 검색을 참조하십시오.

기본값은 NULL이지만, InnoDB가 시작 시 검사할 디렉터리 목록을 작성할 때 innodb_data_home_dir, innodb_undo_directory 및 datadir로 정의된 디렉터리는 항상 innodb_directories 인수 값에 추가됩니다. 이러한 디렉터리는 innodb_directories 설정이 명시적으로 지정되었는지 여부에 관계없이 추가됩니다.

innodb_directories는 시작 명령 또는 MySQL 옵션 파일에서 옵션으로 지정할 수 있습니다. 일부 명령 해석기는 세미콜론(;)을 특수 문자로 해석하기 때문에 인수 값을 따옴표로 묶어야 합니다. (예를 들어, 유닉스 셸은 이를 명령 종결자로 취급합니다.)

시작 명령:

```
mysqld --innodb-directories="directory_path_1;directory_path_2"
```

MySQL 옵션 파일입니다:

```
[mysqld]
innodb_directories="directory_path_1;directory_path_2"
```

와일드카드 표현식은 디렉터리를 지정하는 데 사용할 수 없습니다.

innodb_directories 스캔은 지정된 디렉터리의 하위 디렉터리도 검색합니다. 중복된 디렉터리와 하위 디렉터리는 스캔할 디렉터리 목록에서 삭제됩니다.

자세한 내용은 섹션 15.6.3.6, '서버가 오프라인일 때 테이블스페이스 파일 이동'을 참조하세요.

- innodb_disable_sort_파일_cache

명령줄 형식	--innodb-disable-sort-file-cache[={OFF|ON}]
시스템 변수	innodb_disable_sort_파일_cache
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요

유형	부울
기본값	꺼짐

병합 정렬 임시 파일에 대한 운영 체제 파일 시스템 캐시를 비활성화합니다. 그 결과 이러한 파일은 `O_DIRECT`에 해당하는 파일로 열리게 됩니다.

- `innodb_doublewrite`

명령줄 형식	`--innodb-doublewrite=값`
시스템 변수	`innodb_doublewrite`

범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	열거형
기본값	켜기
유효한 값	켜기 끄기 DETECT_AND_REPAIR DETECT_ONLY

innodb_doublewrite 변수는 이중 쓰기 버퍼링을 제어합니다. 대부분의 경우 이중 쓰기 버퍼링은 기본적으로 활성화되어 있습니다.

서버를 시작할 때 innodb_doublewrite를 ON 또는 OFF로 설정하여 각각 이중 쓰기 버퍼링을 활성화 또는 비활성화할 수 있습니다. DETECT_AND_RECOVER는 ON과 동일합니다. 이 설정을 사용하면 이중 쓰기 버퍼가 완전히 활성화되어 복구 중에 불완전한 페이지 쓰기를 수정하기 위해 데이터베이스 페이지 콘텐츠가 이중 쓰기 버퍼에 액세스된다는 점을 제외하고는 이중 쓰기 버퍼가 완전히 활성화됩니다. DETECT_ONLY를 사용하면 메타데이터만 이중 쓰기 버퍼에 기록됩니다. 데이터베이스 페이지 콘텐츠는 이중 쓰기 버퍼에 기록되지 않으며 복구 시 이중 쓰기 버퍼를 사용하여 불완전한 페이지 쓰기를 수정하지 않습니다. 이 경량 설정은 불완전한 페이지 쓰기만 감지하기 위한 것입니다.

MySQL 8.2는 이중 쓰기 버퍼를 활성화하는 innodb_doublewrite 설정에 대한 동적 변경을 ON, DETECT_AND_RECOVER 및 DETECT_ONLY 사이에서 지원합니다. MySQL은 이중 쓰기 버퍼를 활성화하는 설정에서 OFF로 또는 그 반대로의 동적 변경을 지원하지 않습니다.

이중 쓰기 버퍼가 원자 쓰기를 지원하는 Fusion-io 장치에 있는 경우, 이중 쓰기 버퍼는 자동으로 비활성화되고 데이터 파일 쓰기는 대신 Fusion-io 원자 쓰기를 사용하여 수행됩니다. 그러나 innodb_doublewrite 설정은 전역 설정이라는 점에 유의하십시오. 이중 쓰기 버퍼가 비활성화되면 Fusion-io 하드웨어에 상주하지 않는 파일을 포함한 모든 데이터 파일에 대해 이 기능이 비활성화됩니다. 이 기능은 Fusion-io 하드웨어에서만 지원되며 Fusion-
io NVMFS를 지원합니다. 이 기능을 최대한 활용하려면 innodb_flush_method 설정의 O_DIRECT를 권장합니다.

관련 정보는 섹션 15.6.4, '이중 쓰기 버퍼'를 참조하세요.

- INNODB_DOUBLEWRITE_BATCH_SIZE

명령줄 형식	--innodb-doublewrite-batch-size=#

시스템 변수	INNODB_DOUBLEWRITE_BATCH_SIZE
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	0
최소값	0
최대 값	256

일괄적으로 기록할 이중 쓰기 페이지 수를 정의합니다. 자세한 내용

은 섹션 15.6.4, "이중 쓰기 버퍼"를 참조하십시오.

- innodb_doublewrite_dir

명령줄 형식	--innodb-doublewrite-dir=dir_name
시스템 변수	innodb_doublewrite_dir
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	디렉토리 이름

이중 쓰기 파일의 디렉터리를 정의합니다. 디렉터리가 지정되지 않은 경우 기본적으로 데이터 디렉터리가 되는 innodb_data_home_dir 디렉터리에 이중 쓰기 파일이 생성됩니다.

자세한 내용은 섹션 15.6.4, '이중 쓰기 버퍼'를 참조하세요.

- innodb_doublewrite_files

명령줄 형식	--innodb-doublewrite-files=#
시스템 변수	innodb_doublewrite_files
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	innodb_buffer_pool_instances * 2
최소값	2
최대 값	256

이중 쓰기 파일 수를 정의합니다. 기본적으로 각 버퍼 풀 인스턴스에 대해 두 개의 이중 쓰기 파일이 생성됩니다.

이중 쓰기 파일은 최소 두 개입니다. 최대 이중 쓰기 파일 수는 버퍼 풀 인스턴스 수의 두 배입니다. (버퍼 풀 인스턴스 수는 innodb_buffer_pool_instances 변수에 의해 제어됩니다.)

자세한 내용은 섹션 15.6.4, '이중 쓰기 버퍼'를 참조하세요.

- innodb_doublewrite_pages

명령줄 형식	--innodb-doublewrite-pages=#
시스템 변수	innodb_doublewrite_pages
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요

유형	Integer
기본값	`innodb_write_io_threads` 값
최소값	`innodb_write_io_threads` 값
최대 값	`512`

일괄 쓰기에 대한 스레드당 최대 이중 쓰기 페이지 수를 정의합니다. 값을 지정하지 않으면 innodb_dblwr_pages가 `innodb_write_io_threads` 값으로 설정됩니다.

자세한 내용은 섹션 15.6.4, '이중 쓰기 버퍼'를 참조하세요.

- INNODB_EXTEND_AND_초기화

명령줄 형식	--innodb=extend-and-초기화[={OFF|ON}]
시스템 변수	INNODB_EXTEND_AND_초기화
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	부울
기본값	켜기

Linux 시스템에서 테이블별 파일 및 일반 테이블 공간에 공간을 할당하는 방법을 제어합니다.

이 옵션을 활성화하면 InnoDB는 새로 할당된 페이지에 NULL을 씁니다. 비활성화하면 물리적으로 NULL을 쓰지 않고 공간을 예약하는 posix_fallocate() 호출을 사용하여 공간이 할당됩니다.

자세한 내용은 섹션 15.6.3.8, 'Linux에서 테이블 공간 할당 최적화'를 참조하세요.

- innodb_fast_shutdown

명령줄 형식	--innodb-fast-shutdown=#
시스템 변수	innodb_fast_shutdown
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	1
유효한 값	0 1 2

InnoDB 종료 모드입니다. 값이 0이면 InnoDB는 종료하기 전에 느린 종료, 전체 제거 및 변경 버퍼 병합을 수행합니다. 값이 1(기본값)이면 InnoDB는 이러한 작업을 건너뜁니다.
작업을 수행하며, 이 프로세스를 빠른 종료라고 합니다. 값이 2이면 InnoDB는 MySQL이 충돌한 것처럼 로그를 플러시하고 콜드 종료합니다. 커밋된 트랜잭션은 손실되지 않지만 충돌 복구 작업으로 인해 다음 시작이 더 오래 걸립니다.

느린 종료는 상당한 양의 데이터가 여전히 버퍼링되어 있는 극단적인 경우에는 몇 분, 심지어 몇 시간이 걸릴 수도 있습니다. MySQL 주요 릴리스 간에 업그레이드 또는 다운그레이드하기 전에 느린 종료 기술을 사용하여 업그레이드 프로세스에서 파일 형식이 업데이트될 경우에 대비하여 모든 데이터 파일이 완전히 준비되도록 하세요.

데이터가 손상될 위험이 있는 경우 가장 빠르게 종료하려면 긴급 상황 또는 문제 해결 상황에서 `innodb_fast_shutdown=2`를 사용합니다.

- `innodb_fil_make_page_dirty_debug`

명령줄 형식	`--innodb-fil-make-page-dirty-debug=#`
시스템 변수	`innodb_fil_make_page_dirty_debug`
범위	글로벌
동적	예

3203

SET_VAR 힌트 적용	아니요
유형	Integer
기본값	0
최소값	0
최대 값	2**32-1

기본적으로 innodb_fil_make_page_dirty_debug를 테이블 스페이스의 ID로 설정하면 테이블 스페이스의 첫 번째 페이지가 즉시 더티 처리됩니다. innodb_saved_page_number_debug가 기본값이 아닌 값으로 설정된 경우, innodb_fil_make_page_dirty_debug를 설정하면 지정된 페이지가 더럽힙니다. innodb_fil_make_page_dirty_debug 옵션은 디버깅 지원이 WITH_DEBUG CMake 옵션을 사용하여 컴파일된 경우에만 사용할 수 있습니다.

- innodb_file_per_table

명령줄 형식	--innodb-file-per-table[={OFF|ON}]
시스템 변수	innodb_file_per_table
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	부울
기본값	켜기

innodb_file_per_table을 활성화하면 기본적으로 테이블이 파일별 테이블 공간에 생성됩니다. 비활성화하면 기본적으로 시스템 테이블 스페이스에 테이블이 생성됩니다. 테이블별 파일 테이블스페이스에 대한 자세한 내용은 섹션 15.6.3.2, "테이블별 파일 테이블스페이스"를 참조하십시오. InnoDB 시스템 테이블 스페이스에 대한 자세한 내용은 섹션 15.6.3.1, "시스템 테이블 스페이스"를 참조하십시오.

innodb_file_per_table 변수는 시작 시 명령줄에 지정하거나 옵션 파일에 지정한 SET GLOBAL 문을 사용하여 런타임에 구성할 수 있습니다. 런타임에 구성하려면 전역 시스템 변수를 설정할 수 있는 충분한 권한이 필요하며(5.1.9.1절, "시스템 변수 권한" 참조) 모든 연결의 작동에 즉시 영향을 줍니다.

파일 단위 테이블 공간에 있는 테이블을 잘라내거나 삭제하면 확보된 공간이 운영 체제로 반환됩니다.
시스템에 있는 테이블 잘라내기 또는 삭제하기
테이블스페이스는 시스템 테이블스페이스의 공간만 해제합니다. 시스템 테이블스페이스에서 해제된 공간은 InnoDB 데이터에 다시 사용할 수 있지만 시스템 테이블스페이스 데이터 파일은 축소되지 않으므로 운영 체제로 반환되지는 않습니다.

innodb_file_per-table 설정은 임시 테이블 생성에 영향을 주지 않으며, 임시 테이블은 세션 임시 테이블 공간에 생성됩니다. 15.6.3.5절, "임시 테이블 스페이스"를 참조하십시오.

- innodb_fill_factor

명령줄 형식	--innodb-fill-factor=#
시스템 변수	innodb_fill_factor
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	100
최소값	10

최대 값	100

InnoDB는 인덱스를 생성하거나 재구축할 때 대량 로드를 수행합니다. 이 인덱스 생성 방법을 "정렬된 인덱스 빌드"라고 합니다.

innodb_fill_factor는 정렬된 인덱스 빌드 중에 채워지는 각 B-tree 페이지의 공간 비율을 정의하며, 나머지 공간은 향후 인덱스 성장을 위해 예약합니다. 예를 들어, innodb_fill_factor를 80으로 설정하면 향후 인덱스 증가를 위해 각 B-트리 페이지의 공간 중 20%가 예약됩니다. 실제 백분율은 다를 수 있습니다. innodb_fill_factor 설정은 엄격한 제한이 아닌 힌트로 해석됩니다.

innodb_fill_factor를 100으로 설정하면 클러스터된 인덱스 페이지의 1/16 공간이 향후 인덱스 증가를 위해 여유 공간으로 남습니다.

innodb_fill_factor는 B-트리 리프 페이지와 비리프 페이지 모두에 적용됩니다. 텍스트 또는 블롭 항목에 사용되는 외부 페이지에는 적용되지 않습니다.

자세한 내용은 섹션 15.6.2.3, '정렬된 인덱스 빌드'를 참조하세요.

- innodb_flush_log_at_timeout

명령줄 형식	--innodb-flush-log-at-timeout=#
시스템 변수	innodb_flush_log_at_timeout
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	1
최소값	1
최대 값	2700
단위	초

N초마다 로그를 쓰고 플러시합니다. innodb_flush_log_at_timeout을 사용하면 플러시를 줄이고 바이너리 로그 그룹 커밋의 성능에 영향을 미치지 않도록 플러시 사이의 타임아웃 기간을 늘릴 수 있습니다. innodb_flush_log_at_timeout의 기본 설정은 초당 한 번입니다.

- innodb_flush_log_at_trx_commit

명령줄 형식	--innodb-flush-log-at-trx-commit=#
시스템 변수	innodb_flush_log_at_trx_commit
범위	글로벌

동적	예
SET_VAR 힌트 적용	아니요
유형	열거형
기본값	1
유효한 값	0 1

	2

커밋 작업에 대한 엄격한 ACID 준수와 커밋 관련 I/O 작업을 재배치하고 일괄적으로 수행할 때 가능한 더 높은 성능 사이의 균형을 제어합니다. 기본값을 변경하여 더 나은 성능을 얻을 수 있지만 충돌이 발생하면 트랜잭션이 손실될 수 있습니다.

- 완전한 ACID 준수를 위해서는 기본 설정인 1이 필요합니다. 트랜잭션을 커밋할 때마다 로그가 기록되고 디스크에 플러시됩니다.
- 0으로 설정하면 초당 한 번씩 로그가 기록되고 디스크에 플러시됩니다. 로그가 플러시되지 않은 트랜잭션은 충돌 시 손실될 수 있습니다.
- 2로 설정하면 트랜잭션이 커밋될 때마다 로그가 작성되고 초당 한 번씩 디스크에 플러시됩니다. 로그가 플러시되지 않은 트랜잭션은 크래시 시 손실될 수 있습니다.
- 설정 0과 2의 경우, 초당 1회 플러시가 100% 보장되지는 않습니다. DDL 변경 및 기타 내부 InnoDB 활동으로 인해 innodb_flush_log_at_trx_commit 설정과 무관하게 로그가 플러시될 수 있으며, 스케줄링 문제로 인해 플러시 빈도가 더 낮아질 수도 있습니다. 로그가 초당 한 번씩 플러시되는 경우, 크래시 발생 시 최대 1초의 트랜잭션이 손실될 수 있습니다. 로그가 초당 한 번보다 더 자주 또는 덜 자주 플러시되는 경우 손실될 수 있는 트랜잭션의 양은 그에 따라 달라집니다.
- 로그 플러싱 빈도는 innodb_flush_log_at_timeout에 의해 제어되며, 이를 통해 로그 플러싱 빈도를 N초(여기서 N은 1 ... 2700, 기본값은 1)로 설정할 수 있습니다. 그러나 예기치 않은 mysqld 프로세스 종료로 인해 최대 N초의 트랜잭션이 지워질 수 있습니다.
- DDL 변경 및 기타 내부 InnoDB 활동과 독립적으로 로그를 플러시합니다. innodb_flush_log_at_trx_commit 설정.
- InnoDB 크래시 복구는 innodb_flush_log_at_trx_commit 설정에 관계없이 작동합니다. 트랜잭션은 완전히 적용되거나 완전히 지워집니다.

트랜잭션과 함께 InnoDB를 사용하는 복제 설정의 내구성과 일관성을 위해:

- 바이너리 로깅이 활성화된 경우 sync_binlog=1로 설정합니다.
- 항상 innodb_flush_log_at_trx_commit=1로 설정합니다.

예기치 않은 중단에 가장 탄력적으로 대응하는 복제본의 설정 조합에 대한 자세한 내용은 17.4.2절. "복제본의 예기치 않은 중단 처리하기"를 참조하세요.

주의

많은 운영 체제와 일부 디스크 하드웨어는 디스크 플러시 작업을 속입니다. 실제로는 플러시가 수행되지 않았는데도 mysqld에 플러시가 수행되었다고 표시할 수 있

습니다. 이 경우 권장 설정으로도 트랜잭션의 내구성이 보장되지 않으며, 최악의 경우 정전으로 인해 `InnoDB` 데이터가 손상될 수 있습니다. SCSI 디스크 컨트롤러 또는 디스크 자체에 배터리 백업 디스크 캐시를 사용하면 파일 플러시 속도가 빨라지고 작업이 더 안전해집니다. 하드웨어 캐시에서 디스크 쓰기 캐싱을 비활성화할 수도 있습니다.

- `innodb_flush_method`

명령줄 형식	`--innodb-flush-method=값`
시스템 변수	`innodb_flush_method`
범위	글로벌

동적	아니요
SET_VAR 힌트 적용	아니요
유형	문자열
기본값(유닉스)	fsync
기본값(Windows)	언버퍼링
유효한 값(유닉스)	fsync O_DSYNC littlesync nosync O_DIRECT O_Direct_NO_FSYNC
유효한 값(Windows)	언버퍼링 정상

I/O 처리량에 영향을 줄 수 있는 InnoDB 데이터 파일 및 로그 파일로 데이터를 플러시하는 데 사용되는 방법을 정의합니다.

유닉스 계열 시스템에서는 기본값이 fsync입니다. Windows에서는 기본값이 언버퍼링됩니다.

참고

MySQL 8.2에서는 innodb_flush_method 옵션을 숫자로 지정할 수 있습니다.

유닉스 계열 시스템용 innodb_flush_method 옵션은 다음과 같습니다:

- fsync 또는 0: InnoDB는 fsync() 시스템 호출을 사용하여 데이터 및 로그 파일을 모두 플러시합니다.
- O_DSYNC 또는 1: InnoDB는 로그 파일을 열고 플러시할 때 O_SYNC를 사용하고, 데이터 파일을 플러시할 때 fsync()를 사용합니다. 여러 종류의 Unix에서 문제가 발생했기 때문에 InnoDB는 O_DSYNC를 직접 사용하지 않습니다.
- littlesync 또는 2: 이 옵션은 내부 성능 테스트에 사용되며 현재 지원되지 않습니다. 사용자 책임하에 사용하세요.
- nosync 또는 3: 이 옵션은 내부 성능 테스트에 사용되며 현재 지원되지 않습니다. 사용자 책임하에 사용하세요.
- O_DIRECT 또는 4: InnoDB는 데이터 파일을 열기 위해 O_DIRECT(또는 Solaris의 경우

directio()를 사용하며, 데이터와 로그 파일을 모두 플러시하기 위해 fsync()를 사용합니다. 이 옵션은 일부 GNU/Linux 버전, FreeBSD 및 Solaris에서 사용할 수 있습니다.

- O_DIRECT_NO_FSYNC: InnoDB는 I/O를 플러시하는 동안 O_DIRECT를 사용하지만 fsync() 함수는 건너뜁니다. 시스템 호출을 호출합니다.

  MySQL은 새 파일을 만든 후, 파일 크기를 늘린 후, 파일을 닫은 후 fsync()를 호출하여 파일 시스템 메타데이터 변경 사항이 동기화되도록 합니다. 각 쓰기 작업 후에는 여전히 fsync() 시스템 호출을 건너뜁니다.

  재실행 로그 파일과 데이터 파일이 서로 다른 저장 장치에 있는 경우, 배터리가 없는 장치 캐시에서 데이터 파일 쓰기가 플러시되기 전에 예기치 않은 종료가 발생하는 경우 데이터 손실이 발생할 수 있습니다.

백업됩니다. 재실행 로그 파일과 데이터 파일에 서로 다른 저장 장치를 사용 중이거나 사용하려는 경우, 데이터 파일이 배터리로 백업되지 않는 캐시가 있는 장치에 있는 경우, 대신 O_DIRECT를 사용하세요.

fdatasync() 시스템 호출을 지원하는 플랫폼에서 innodb_use_fdatasync 변수는 fsync()를 사용하는 innodb_flush_method 옵션을 대신하여 fdatasync()를 사용할 수 있도록 허용합니다. 후속 데이터 검색에 필요한 경우가 아니라면 fdatasync() 시스템 호출은 파일 메타데이터의 변경 사항을 플러시하지 않으므로 잠재적인 성능 이점을 제공합니다.

Windows 시스템용 innodb_flush_method 옵션은 다음과 같습니다:

- 버퍼링되지 않음 또는 0: InnoDB는 시뮬레이션된 비동기 I/O와 버퍼링되지 않은 I/O를 사용합니다.
- 정상 또는 1: InnoDB는 시뮬레이션된 비동기 I/O 및 버퍼링된 I/O를 사용합니다.

각 설정이 성능에 미치는 영향은 하드웨어 구성 및 워크로드에 따라 다릅니다. 특정 구성을 벤치마크하여 어떤 설정을 사용할지 또는 기본 설정을 유지할지 결정하세요.
Innodb_data_fsyncs 상태 변수를 검사하여 각 설정에 대한 전체 fsync() 호출 횟수(또는 innodb_use_fdatasync가 활성화된 경우 fdatasync() 호출 횟수)를 확인합니다. 워크로드에서 읽기 및 쓰기 작업의 혼합은 설정의 성능에 영향을 미칠 수 있습니다. 예를 들어, 하드웨어 RAID 컨트롤러와 배터리 지원 쓰기 캐시가 있는 시스템에서 O_DIRECT는 InnoDB 버퍼 풀과 운영 체제 파일 시스템 캐시 간의 이중 버퍼링을 방지하는 데 도움이 될 수 있습니다. InnoDB 데이터 및 로그 파일이 SAN에 있는 일부 시스템에서는 SELECT 문이 대부분인 읽기 위주의 워크로드에 대해 기본값 또는 O_DSYNC가 더 빠를 수 있습니다. 항상 프로덕션 환경을 반영하는 하드웨어 및 워크로드를 사용하여 이 매개변수를 테스트하세요. 일반적인 I/O 튜닝에 대한 조언은 섹션 8.5.8, "InnoDB 디스크 I/O 최적화"를 참조하십시오.

innodb_dedicated_server가 활성화된 경우, 명시적으로 정의되지 않은 경우 innodb_flush_method 값이 자동으로 구성됩니다. 자세한 내용은 15.8.12절, "전용 MySQL 서버에 대한 자동 구성 활성화"를 참조하세요.

- innodb_flush_neighbors

명령줄 형식	--innodb-flush-이웃=#
시스템 변수	innodb_flush_neighbors
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	열거형
기본값	0

유효한 값	0 1 2

`InnoDB` 버퍼 풀에서 페이지를 플러시할 때 다른 더티 페이지도 같은 범위에서 플러시할지 여부를 지정합니다.

- 0으로 설정하면 `innodb_flush_neighbors`가 비활성화됩니다. 같은 범위의 더티 페이지는 플러시되지 않습니다.
- 1로 설정하면 연속된 더티 페이지를 동일한 범위로 플러시합니다.
- 2로 설정하면 더러운 페이지가 같은 정도로 플러시됩니다.

테이블 데이터가 기존 HDD 저장 장치에 저장되어 있는 경우, 이러한 인접 페이지를 한 번의 작업으로 플러시하면 플러시하지 않을 때보다 I/O 오버헤드(주로 디스크 탐색 작업의 경우)가 줄어듭니다.

개별 페이지를 서로 다른 시간에 기록할 수 있습니다. SSD에 저장된 테이블 데이터의 경우 검색 시간은 중요한 요소가 아니므로 이 옵션을 0으로 설정하여 쓰기 작업을 분산할 수 있습니다. 관련 정보는 섹션 15.8.3.5, "버퍼 풀 플러싱 구성"을 참조하십시오.

- innodb_flush_sync

명령줄 형식	--innodb-flush-sync[={OFF|ON}]
시스템 변수	innodb_flush_sync
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	부울
기본값	켜기

기본적으로 활성화되어 있는 innodb_flush_sync 변수를 사용하면 체크포인트에서 발생하는 I/O 활동 버스트 중에 innodb_io_capacity 설정이 무시됩니다. innodb_io_capacity 설정에 정의된 I/O 속도를 준수하려면 innodb_flush_sync를 비활성화합니다.

innodb_flush_sync 변수 구성에 대한 자세한 내용은 15.8.7절, "InnoDB I/O 용량 구성"을 참조하세요.

- INNODB_FLUSHING_AVG_LOOPS

명령줄 형식	--innodb-flushing-avg-loops=#
시스템 변수	INNODB_FLUSHING_AVG_LOOPS
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	30
최소값	1
최대 값	1000

InnoDB가 이전에 계산된 플러싱 상태의 스냅샷을 유지하는 반복 횟수로, 워크로드 변화에 따라 적응형 플러싱이 얼마나 빨리 반응하는지를 제어합니다. 값을 늘리면 워크로드 변화에 따라 플러시 작업 속도가 부드럽고 점진적으로 변경됩니다. 값을 낮추면 적응형 플러싱이 워크로드 변화에 빠르게 적응하여 워크로드가 갑자기 증가하거나 감소할 경우 플러싱 작업이 급증할 수 있습니다.

관련 정보는 섹션 15.8.3.5, "버퍼 풀 플러싱 구성하기"를 참조하세요.

- INNODB_FORCE_LOAD_CROSSED

명령줄 형식	`--innodb-force-load-corrupted[={OFF|ON}]`
시스템 변수	`INNODB_FORCE_LOAD_CROSSED`
범위	글로벌
동적	아니요
`SET_VAR` 힌트 적용	아니요
유형	부울

기본값	꺼짐

InnoDB가 시작 시 손상된 것으로 표시된 테이블을 로드할 수 있도록 허용합니다. 문제 해결 중에만 사용하며, 그렇지 않으면 액세스할 수 없는 데이터를 복구하는 데 사용합니다. 문제 해결이 완료되면 이 설정을 비활성화하고 서버를 다시 시작합니다.

- innodb_force_recovery

명령줄 형식	--innodb-force-recovery=#
시스템 변수	innodb_force_recovery
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	0
최소값	0
최대 값	6

일반적으로 심각한 문제 해결 상황에서만 변경되는 크래시 복구 모드입니다. 가능한 값은 0에서 6까지입니다. 이러한 값의 의미와 innodb_force_recovery에 대한 중요한 정보는 15.21.3절, "InnoDB 복구 강제 수행"을 참조하십시오.

경고

긴급 상황에서만 이 변수를 0보다 큰 값으로 설정하여 InnoDB를 시작하고 테이블을 덤프할 수 있도록 하세요. 안전 조치로 InnoDB는 INSERT, UPDATE 또는 DELETE를 방지합니다. innodb_force_recovery가 0보다 클 때. innodb_force_recovery를 4 이상으로 설정하면 InnoDB가 읽기 전용 모드로 전환됩니다.

이러한 제한으로 인해 복제 관리 명령이 오류와 함께 실패할 수 있습니다. 복제는 복제 상태 로그를 InnoDB 테이블에 저장하므로 복제 관리 명령이 실패할 수 있습니다.

- innodb_fsync_threshold

명령줄 형식	--innodb-fsync-threshold=#
시스템 변수	innodb_fsync_threshold
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer

기본값	0
최소값	0
최대 값	2**64-1

기본적으로 InnoDB는 새 로그 파일 또는 테이블 스페이스 파일과 같은 새 데이터 파일을 만들 때 파일을 디스크에 플러시하기 전에 운영 체제 캐시에 완전히 쓰므로 한 번에 많은 양의 디스크 쓰기 작업이 발생할 수 있습니다. 운영 체제 캐시에서 데이터를 더 작고 주기적으로 플러시하도록 하려면 innodb_fsync_threshold 변수를 사용하여
임계값(바이트 단위)을 입력합니다. 바이트 임계값에 도달하면 운영 체제의 콘텐츠가
캐시가 디스크에 플러시됩니다. 기본값 0을 사용하면 파일이 캐시에 완전히 쓰여진 후에만 데이터를 디스크에 플러시하는 기본 동작이 강제로 적용됩니다.

임계값을 지정하여 더 작은 규모의 주기적 플러시를 강제하는 것은 여러 MySQL 인스턴스가 동일한 저장 장치를 사용하는 경우에 유용할 수 있습니다. 예를 들어, 새 MySQL 인스턴스와 관련 데이터 파일을 생성하면 디스크 쓰기 활동이 급증하여 동일한 저장 장치를 사용하는 다른 MySQL 인스턴스의 성능이 저하될 수 있습니다. 임계값을 구성하면 이러한 쓰기 활동의 급증을 방지하는 데 도움이 됩니다.

- `INNODB_FT_aux_table`

시스템 변수	`INNODB_FT_aux_table`
범위	글로벌
동적	예
`SET_VAR` 힌트 적용	아니요
유형	문자열

전체 텍스트 인덱스가 포함된 `InnoDB` 테이블의 정규화된 이름을 지정합니다. 이 변수는 진단 목적으로 사용되며 런타임에만 설정할 수 있습니다. 예를 들어

```
SET GLOBAL innodb_ft_aux_table = 'test/t1';
```

이 변수를 *`db_name/table_name`* 형식의 이름으로 설정하면 `INFORMATION_SCHEMA` 테이블 `INNODB_FT_INDEX_TABLE`, `INNODB_FT_INDEX_CACHE`, `INNODB_FT_CONFIG`, `INNODB_FT_DELETED` 및 `INNODB_FT_BEING_DELETED`는 지정된 테이블의 검색 인덱스에 대한 정보를 표시합니다.

자세한 내용은 섹션 15.15.4, "InnoDB 정보_SCHEMA 전체 텍스트 인덱스 테이블"을 참조하세요.

- `INNODB_FT_CACHE_SIZE`

명령줄 형식	`--innodb-ft-cache-size=#`
시스템 변수	`INNODB_FT_CACHE_SIZE`
범위	글로벌
동적	아니요
`SET_VAR` 힌트 적용	아니요
유형	Integer
기본값	`8000000`
최소값	`1600000`
최대 값	`80000000`
단위	바이트

`InnoDB FULLTEXT` 인덱스를 생성하는 동안 구문 분석된 문서를 메모리에 보관하는 `InnoDB FULLTEXT` 검색 인덱스 캐시에 할당된 메모리(바이트 단위)입니다. 인덱스 삽입 및 업데이트는 `innodb_ft_cache_size` 크기 제한에 도달할 때만 디스크에 커밋됩니다.

innodb_ft_cache_size는 테이블 단위로 캐시 크기를 정의합니다. 모든 테이블에 대한 전역 제한을 설정하려면 `innodb_ft_total_cache_size`를 참조하세요.

자세한 내용은 InnoDB 전체 텍스트 인덱스 캐시를 참조하세요.

- INNODB_FT_ENABLE_DIAG_PRINT

명령줄 형식	--innodb-ft-enable-diag-print[={OFF|ON}]
시스템 변수	INNODB_FT_ENABLE_DIAG_PRINT

범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	부울
기본값	꺼짐

추가 전체 텍스트 검색(FTS) 진단 출력을 사용하도록 설정할지 여부입니다. 이 옵션은 주로 고급 FTS 디버깅을 위한 것으로 대부분의 사용자에게는 적합하지 않습니다. 출력은 오류 로그에 인쇄되며 다음과 같은 정보를 포함합니다:

- FTS 인덱스 동기화 진행률(FTS 캐시 제한에 도달한 경우). 예를 들어

```
테이블 테스트용 FTS 동기화, 삭제된 개수: 100 크기: 10000 바이트 동기화 단어:
100
```

- FTS는 진행 상황을 최적화합니다. 예를 들어

```
FTS 최적화 테스트 시작
FTS_OPTIMIZE: 최적화 "mysql"
FTS_OPTIMIZE: 처리된 "mysql"
```

- FTS 인덱스 빌드 진행률. 예를 들어

```
처리된 문서 수입니다: 1000
```

- FTS 쿼리의 경우 쿼리 구문 분석 트리, 단어 가중치, 쿼리 처리 시간 및 메모리 사용량이 인쇄됩니다. 예를 들어

```
FTS 검색 처리 시간: 1초: 100밀리초: 행 10000 전체 검색 메모리: 245666(바
이트), 행: 10000
```

- INNODB_FT_ENABLE_STOPWORD

명령줄 형식	--innodb-ft-enable-stopword[={OFF|ON}]
시스템 변수	INNODB_FT_ENABLE_STOPWORD
범위	글로벌, 세션
동적	예
SET_VAR 힌트 적용	아니요
유형	부울
기본값	켜기

인덱스가 생성될 때 스톱워드 세트가 InnoDB 전체 텍스트 인덱스와 연결되도록 지정합니다. innodb_ft_user_stopword_table 옵션이 설정되어 있으면 해당 테이블에서 스톱워드를 가져옵니다. 그렇지 않으면, innodb_ft_server_stopword_table 옵션이 설정된 경우, 해당 테이블에서 스톱워

드를 가져옵니다. 그렇지 않으면 기본 제공되는 기본 스톱워드 세트가 사용됩니다.

자세한 내용은 섹션 12.9.4, "전체 텍스트 중지 단어"를 참조하세요.

- `INNODB_FT_MAX_TOKEN_SIZE`

명령줄 형식	`--innodb-ft-max-token-size=#`
시스템 변수	`INNODB_FT_MAX_TOKEN_SIZE`
범위	글로벌
동적	아니요
`SET_VAR` 힌트 적용	아니요

유형	Integer
기본값	84
최소값	10
최대 값	84

InnoDB 전체 텍스트 인덱스에 저장되는 단어의 최대 문자 길이입니다. 이 값에 제한을 설정하면 실제 단어가 아니며 검색어가 될 가능성이 없는 긴 키워드나 임의의 문자 모음을 생략하여 인덱스의 크기를 줄여 쿼리 속도를 높일 수 있습니다.

자세한 내용은 섹션 12.9.6, 'MySQL 전체 텍스트 검색 미세 조정'을 참조하세요.

- INNODB_FT_MIN_TOKEN_SIZE

명령줄 형식	--innodb-ft-min-token-size=#
시스템 변수	INNODB_FT_MIN_TOKEN_SIZE
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	3
최소값	0
최대 값	16

InnoDB 전체 텍스트 인덱스에 저장되는 단어의 최소 길이입니다. 이 값을 늘리면 영어 단어 "a" 및 "to"와 같이 검색 컨텍스트에서 중요하지 않은 일반적인 단어를 생략하여 인덱스의 크기를 줄여 쿼리 속도를 높일 수 있습니다. 한중일(중국어, 일본어, 한국어) 문자 집합을 사용하는 콘텐츠의 경우 값을 1로 지정합니다.

자세한 내용은 섹션 12.9.6, 'MySQL 전체 텍스트 검색 미세 조정'을 참조하세요.

- INNODB_FT_NUM_WORD_OPTIMIZE

명령줄 형식	--innodb-ft-num-word-optimize=#
시스템 변수	INNODB_FT_NUM_WORD_OPTIMIZE
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	2000
최소값	1000
최대 값	10000

InnoDB 전체 텍스트 인덱스에 대한 각 OPTIMIZE TABLE 작업 중에 처리할 단어 수입니다. 전체 텍

스트 검색 인덱스가 포함된 테이블에 대량 삽입 또는 업데이트 작업을 수행하려면 모든 변경 사항을 통합하기 위해 상당한 인덱스 유지 관리가 필요할 수 있으므로, 일련의 `OPTIMIZE TABLE` 문을 수행하여 각 문이 마지막 중단 지점부터 다시 시작할 수 있습니다.

자세한 내용은 섹션 12.9.6, 'MySQL 전체 텍스트 검색 미세 조정'을 참조하세요.

- `INNODB_FT_RESULT_CACHE_LIMIT`

명령줄 형식	`--innodb-ft-result-cache-limit=#`

시스템 변수	INNODB_FT_RESULT_CACHE_LIMIT
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	2000000000
최소값	1000000
최대 값	2**32-1
단위	바이트

전체 텍스트 검색 쿼리 또는 스레드당 InnoDB 전체 텍스트 검색 쿼리 결과 캐시 제한(바이트 단위로 정의됨)입니다. 중간 및 최종 InnoDB 전체 텍스트 검색 쿼리 결과는 메모리에서 처리됩니다. 매우 큰 InnoDB 전체 텍스트 검색 쿼리 결과(예: 수백만 또는 수억 개의 행)의 경우 과도한 메모리 소비를 방지하려면 innodb_ft_result_cache_limit을 사용하여 전체 텍스트 검색 쿼리 결과 캐시에 크기 제한을 설정합니다. 메모리는 전체 텍스트 검색 쿼리가 처리될 때 필요에 따라 할당됩니다. 결과 캐시 크기 제한에 도달하면 쿼리가 허용된 최대 메모리를 초과했음을 나타내는 오류가 반환됩니다.

모든 플랫폼 유형 및 비트 크기에 대한 innodb_ft_결과_캐시_제한의 최대값은 2**32-1입니다.

- innodb_ft_서버_스톱워드_테이블

명령줄 형식	--innodb-ft-server-stopword-table=db_name/table_name
시스템 변수	innodb_ft_서버_스톱워드_테이블
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	문자열
기본값	NULL

이 옵션은 모든 InnoDB 테이블에 대해 고유한 InnoDB 전체 텍스트 인덱스 중지어 목록을 지정하는 데 사용됩니다. 특정 InnoDB 테이블에 대한 고유한 중지어 목록을 구성하려면 innodb_ft_user_stopword_table을 사용합니다.

innodb_ft_server_stopword_table을 중지어 목록이 포함된 테이블의 이름으로 *db_name/table_name* 형식으로 설정합니다.

innodb_ft_server_stopword_table을 구성하기 전에 스톱워드 테이블이 존재해야 하며, 전체 텍스트 인덱스를 생성하기 전에 innodb_ft_enable_stopword가 활성화되어 있어야 하고, innodb_ft_server_stopword_table 옵션이 구성되어 있어야 합니다.

검색어 테이블은 값이라는 단일 VARCHAR 열을 포함하는 InnoDB 테이블이어야 합니다. 자세한 내용은 섹션 12.9.4, "전체 텍스트 스톱워드"를 참조하세요.

- INNODB_FT_SORT_PLL_DEGREE

명령줄 형식	--innodb-ft-sort-pll-degree=#
시스템 변수	INNODB_FT_SORT_PLL_DEGREE
범위	글로벌

동적	아니요
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	2
최소값	1
최대 값	32

검색 인덱스를 구축할 때 InnoDB 전체 텍스트 인덱스에서 텍스트를 인덱싱하고 토큰화하는 데 병렬로 사용되는 스레드 수입니다.

관련 정보는 섹션 15.6.2.4, "InnoDB 전체 텍스트 인덱스"를 참조하세요.
innodb_sort_buffer_size.

- INNODB_FT_총_캐시_크기

명령줄 형식	--innodb-ft-total-cache-size=#
시스템 변수	INNODB_FT_총_캐시_크기
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	640000000
최소값	32000000
최대 값	1600000000
단위	바이트

모든 테이블에 대한 InnoDB 전체 텍스트 검색 인덱스 캐시에 할당된 총 메모리(바이트)입니다. 각각 전체 텍스트 검색 인덱스가 있는 수많은 테이블을 생성하면 사용 가능한 메모리의 상당 부분이 소모될 수 있습니다. innodb_ft_total_cache_size는 모든 전체 텍스트 검색 인덱스에 대한 전역 메모리 제한을 정의하여 과도한 메모리 소비를 방지합니다. 인덱스 작업으로 인해 전역 제한에 도달하면 강제 동기화가 트리거됩니다.

자세한 내용은 InnoDB 전체 텍스트 인덱스 캐시를 참조하세요.

- INNODB_FT_USER_STOPWORD_TABLE

명령줄 형식	--innodb-ft-user-stopword-table=db_name/table_name
시스템 변수	INNODB_FT_USER_STOPWORD_TABLE

범위	글로벌, 세션
동적	예
SET_VAR 힌트 적용	아니요
유형	문자열

기본값	NULL

이 옵션은 특정 테이블에 고유한 InnoDB 전체 텍스트 인덱스 스톱워드 목록을 지정하는 데 사용됩니다. 모든 InnoDB 테이블에 대해 고유한 스톱워드 목록을 구성하려면 innodb_ft_server_stopword_table을 사용합니다.

innodb_ft_user_stopword_table을 중지어 목록이 포함된 테이블의 이름으로 *db_name/table_name* 형식으로 설정합니다.

innodb_ft_user_stopword_table을 구성하기 전에 스톱워드 테이블이 존재해야 하며, 전체 텍스트 인덱스를 생성하기 전에 innodb_ft_enable_stopword가 활성화되어 있어야 하고 innodb_ft_user_stopword_table이 구성되어 있어야 합니다.

검색어 테이블은 값이라는 단일 VARCHAR 열을 포함하는 InnoDB 테이블이어야 합니다. 자세한 내용은

섹션 12.9.4, "전체 텍스트 스톱워드"를 참조하세요.

- INNODB_IDLE_FLUSH_PCT

명령줄 형식	--innodb-idle-flush-pct=#
시스템 변수	INNODB_IDLE_FLUSH_PCT
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	100
최소값	0
최대 값	100

InnoDB가 유휴 상태일 때 페이지 플러시를 제한합니다. innodb_idle_flush_pct 값은 InnoDB에서 사용할 수 있는 초당 I/O 작업 수를 정의하는 innodb_io_capacity 설정의 백분율입니다. 자세한 내용은 유휴 기간 동안 버퍼 플러시 제한을 참조하세요.

- innodb_io_capacity

명령줄 형식	--innodb-io-capacity=#
시스템 변수	innodb_io_capacity
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요

유형	Integer
기본값	200
최소값	100
최대값(64비트 플랫폼)	2**64-1
최대값(32비트 플랫폼)	2**32-1

innodb_io_capacity 변수는 버퍼 풀에서 페이지를 플러시하고 변경 버퍼에서 데이터를 병합하는 등 InnoDB 백그라운드 작업에서 사용할 수 있는 초당 I/O 작업 수(IOPS)를 정의합니다.

innodb_io_capacity 변수 구성에 대한 자세한 내용은 15.8.7절, "InnoDB I/O 용량 구성"을 참조하세요.

- INNODB_IO_CAPACITY_MAX

명령줄 형식	--innodb-io-capacity-max=#
시스템 변수	INNODB_IO_CAPACITY_MAX
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	설명 보기
최소값	100
최대값(32비트 플랫폼)	2**32-1
최대값(유닉스, 64비트 플랫폼)	2**64-1
최대값(Windows, 64비트 플랫폼)	2**32-1

플러싱 활동이 뒤처지면 InnoDB는 더 높은 비율의 I/O 초당 작업 수(IOPS)를 초과할 수 있습니다. innodb_io_capacity_max 변수는 이러한 상황에서 InnoDB 백그라운드 작업에서 수행되는 최대 IOPS 수를 정의합니다.

innodb_io_capacity_max 변수 구성에 대한 자세한 내용은 15.8.7절, "InnoDB I/O 용량 구성"을 참조하세요.

- innodb_limit_optimistic_insert_debug

명령줄 형식	--innodb-limit-optimistic-insert-debug=#
시스템 변수	innodb_limit_optimistic_insert_debug
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	0
최소값	0
최대 값	2**32-1

B-트리 페이지당 레코드 수를 제한합니다. 기본값 0은 제한이 적용되지 않음을 의미합니다. 이 옵션은 디버깅 지원이 WITH_DEBUG CMake 옵션을 사용하여 컴파일된 경우에만 사용할 수 있습니다.

- innodb_lock_wait_timeout

명령줄 형식	--innodb-lock-wait-timeout=#
시스템 변수	innodb_lock_wait_timeout

범위	글로벌, 세션
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	50
최소값	1

최대 값	1073741824
단위	초

InnoDB 트랜잭션이 포기하기 전에 행 잠금을 기다리는 시간(초)입니다. 기본값은 50초입니다. 다른 InnoDB 트랜잭션에 의해 잠긴 행에 액세스하려는 트랜잭션은 다음 오류가 발생하기 전에 최대 이 시간 동안 행에 대한 쓰기 액세스를 기다립니다:

```
오류 1205 (HY000): 잠금 대기 시간 초과; 트랜잭션을 다시 시작해보십시오.
```

잠금 대기 시간 초과가 발생하면 전체 트랜잭션이 아닌 현재 문이 롤백됩니다. 전체 트랜잭션을 롤백하려면 --innodb-rollback-on-timeout 옵션으로 서버를 시작합니다. 섹션 15.21.5, "InnoDB 오류 처리"도 참조하세요.

사용자 피드백을 빠르게 표시하거나 나중에 처리할 수 있도록 업데이트를 대기열에 넣으려면 대화형 애플리케이션이나 OLTP 시스템의 경우 이 값을 낮출 수 있습니다. 다른 대규모 삽입 또는 업데이트 작업이 완료될 때까지 기다리는 데이터 웨어하우스의 변환 단계와 같이 장기간 실행되는 백엔드 작업의 경우 이 값을 늘릴 수 있습니다.

innodb_lock_wait_timeout은 InnoDB 행 잠금에 적용됩니다. MySQL 테이블 잠금은 InnoDB 내부에서 발생하지 않으며 이 시간 제한은 테이블 잠금 대기에는 적용되지 않습니다.

innodb_deadlock_detect가 활성화된 경우(기본값) 교착 상태에는 잠금 대기 시간 제한 값이 적용되지 않는데, 이는 InnoDB가 교착 상태를 즉시 감지하고 교착 상태의 트랜잭션 중 하나를 롤백하기 때문입니다. innodb_deadlock_detect를 비활성화하면 InnoDB는 교착 상태 발생 시 트랜잭션 롤백을 위해 innodb_lock_wait_timeout에 의존합니다. 15.7.5.2절, "교착 상태 감지"를 참조하세요.

innodb_lock_wait_timeout은 런타임에 SET GLOBAL 또는 SET SESSION 문으로 설정할 수 있습니다. GLOBAL 설정을 변경하려면 전역 시스템 변수를 설정할 수 있는 충분한 권한이 필요하며 (5.1.9.1절, "시스템 변수 권한" 참조), 작업에 영향을 줍니다.
이후 연결되는 모든 클라이언트의 수입니다. 모든 클라이언트는 다음에 대한 세션 설정을 변경할 수 있습니다.
해당 클라이언트에만 영향을 미치는 innodb_lock_wait_timeout을 설정합니다.

- INNODB_LOG_BUFFER_SIZE

명령줄 형식	--innodb-log-buffer-size=#
시스템 변수	INNODB_LOG_BUFFER_SIZE
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	16777216

최소값	1048576
최대 값	4294967295

InnoDB가 디스크의 로그 파일에 쓰는 데 사용하는 버퍼의 크기(바이트)입니다. 기본값은 16MB입니다. 로그 버퍼가 크면 트랜잭션이 커밋되기 전에 로그를 디스크에 쓸 필요 없이 대용량 트랜잭션을 실행할 수 있습니다. 따라서 많은 행을 업데이트, 삽입 또는 삭제하는 트랜잭션이 있는 경우 로그 버퍼를 크게 설정하면 디스크 I/O를 절약할 수 있습니다. 관련 정보는 메모리 구성 및 섹션 8.5.4, "InnoDB 재실행 로깅 최적화"를 참조하십시오. 일반적인 I/O 튜닝에 대한 조언은 섹션 8.5.8, "InnoDB 디스크 I/O 최적화"를 참조하십시오.

- innodb_log_checkpoint_fuzzy_now

명령줄 형식	--innodb-log-checkpoint-fuzzy-now[={OFF|ON}]
시스템 변수	innodb_log_checkpoint_fuzzy_now
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	부울
기본값	꺼짐

이 디버그 옵션을 활성화하면 InnoDB가 퍼지 체크포인트를 작성하도록 강제합니다. 이 옵션은 디버깅 지원이 WITH_DEBUG CMake 옵션을 사용하여 컴파일된 경우에만 사용할 수 있습니다.

- innodb_log_checkpoint_now

명령줄 형식	--innodb-log-checkpoint-now[={OFF|ON}]
시스템 변수	innodb_log_checkpoint_now
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	부울
기본값	꺼짐

이 디버그 옵션을 활성화하면 InnoDB가 강제로 체크포인트를 작성합니다. 이 옵션은 디버깅 지원이 WITH_DEBUG CMake 옵션을 사용하여 컴파일된 경우에만 사용할 수 있습니다.

- innodb_log_checksums

명령줄 형식	--innodb-log-checksums[={OFF|ON}]
시스템 변수	innodb_log_checksums
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	부울
기본값	켜기

다시 실행 로그 페이지에 대한 체크섬을 활성화 또는 비활성화합니다.

innodb_log_checksums=ON으로 설정하면 재실행 로그 페이지에 CRC-32C 체크섬 알고리즘이 활성화됩니다. 언제
innodb_log_checksums가 비활성화되면 재실행 로그 페이지 체크섬 필드의 내용이 무시됩니다. 재실행 로그 헤더 페

이지 및 재실행 로그 체크포인트 페이지의 체크섬은 비활성화되지 않습니다.

- innodb_log_compressed_pages

명령줄 형식	--innodb-log-compressed-pages[={OFF|ON}]
시스템 변수	innodb_log_compressed_pages
범위	글로벌

동적	예
SET_VAR 힌트 적용	아니요
유형	부울
기본값	켜기

재압축된 페이지의 이미지를 재실행 로그에 기록할지 여부를 지정합니다. 압축된 데이터가 변경되면 재압축이 발생할 수 있습니다.

복구 중에 다른 버전의 zlib 압축 알고리즘을 사용할 경우 발생할 수 있는 손상을 방지하기 위해 innodb_log_compressed_pages는 기본적으로 활성화되어 있습니다. zlib 버전이 변경되지 않는 것이 확실하다면, 압축 데이터를 수정하는 워크로드에 대한 재실행 로그 생성을 줄이기 위해 innodb_log_compressed_pages를 비활성화할 수 있습니다.

innodb_log_compressed_pages 활성화 또는 비활성화의 효과를 측정하려면, 동일한 워크로드에서 두 설정의 재실행 로그 생성을 비교합니다. 재실행 로그 생성을 측정하는 옵션으로는 엔진 INNODB 상태 표시 출력의 로그 섹션에서 로그 시퀀스 번호(LSN)를 관찰하거나, 재실행 로그 파일에 기록된 바이트 수에 대한 Innodb_os_log_written 상태를 모니터링하는 방법이 있습니다.

관련 정보는 섹션 15.9.1.6, "OLTP 워크로드용 압축"을 참조하십시오.

- INNODB_LOG_FILE_SIZE

명령줄 형식	--innodb-log-file-size=#
사용 중단	예
시스템 변수	INNODB_LOG_FILE_SIZE
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	50331648
최소값	4194304
최대 값	512GB / innodb_log_files_in_group
단위	바이트

참고

innodb_log_file_size 및 innodb_log_files_in_group이 innodb_redo_log_capacity로 대체되었습니다(섹션 15.6.5, "로그 재실행" 참조).

로그 그룹에 있는 각 로그 파일의 바이트 단위 크기입니다. 로그 파일의 합산 크기 (`innodb_log_file_size` * `innodb_log_files_in_group`)는 512GB보다 약간 작은 최대값을 초과할 수 없습니다. 예를 들어 255GB 로그 파일 쌍은 한도에 근접하지만 한도를 초과하지는 않습니다. 기본값은 48MB입니다.

일반적으로 로그 파일의 결합 크기는 서버가 워크로드 활동의 최고점과 최저점을 부드럽게 처리할 수 있을 만큼 충분히 커야 하며, 이는 종종 다음과 같은 작업을 수행할 수 있는 충분한 재실행 로그 공간이 있음을 의미합니다.

는 한 시간 이상의 쓰기 활동을 처리합니다. 값이 클수록 버퍼 풀에서 체크포인트 플러시 작업이 덜 필요하므로 디스크 I/O를 절약할 수 있습니다. 로그 파일이 클수록 크래시 복구 속도도 느려집니다.

최소 innodb_log_file_size는 4MB입니다.

관련 정보는 로그 구성 다시 실행을 참조하십시오. 일반적인 I/O 튜닝에 대한 조언은 섹션 8.5.8, "InnoDB 디스크 I/O 최적화"를 참조하세요.

innodb_dedicated_server가 활성화된 경우, 명시적으로 정의되지 않은 경우 `innodb_log_file_size` 값이 자동으로 구성됩니다. 자세한 내용은 15.8.12절, "전용 MySQL 서버에 대한 자동 구성 활성화"를 참조하세요.

- `INNODB_LOG_FILES_IN_GROUP`

명령줄 형식	`--innodb-log-files-in-group=#`
사용 중단	예
시스템 변수	`INNODB_LOG_FILES_IN_GROUP`
범위	글로벌
동적	아니요
`SET_VAR` 힌트 적용	아니요
유형	Integer
기본값	`2`
최소값	`2`
최대 값	`100`

참고

`innodb_log_file_size` 및 innodb_log_files_in_group이 `innodb_redo_log_capacity`로 대체되었습니다(섹션 15.6.5, "로그 재실행" 참조).

로그 그룹에 있는 로그 파일 수입니다. `InnoDB`는 순환 방식으로 파일에 기록합니다.
기본(및 권장) 값은 2입니다. 파일 위치는 `innodb_log_group_home_dir`로 지정됩니다. 로그 파일의 합산 크기(`innodb_log_file_size` * `innodb_log_files_in_group`)는 최대 512GB까지 가능합니다.

관련 정보는 로그 구성 다시 실행을 참조하세요.

innodb_dedicated_server가 활성화된 경우, 명시적으로 정의되지 않은 경우 `innodb_log_files_in_group`이 자동으로 구성됩니다. 자세한 내용은 15.8.12절, "전용 MySQL 서버에 대한 자동 구성 활성화"를 참조하세요.

- innodb_log_group_home_dir

명령줄 형식	--innodb-log-group-home-dir=dir_name
시스템 변수	innodb_log_group_home_dir
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	디렉토리 이름

InnoDB 재실행 로그 파일의 디렉터리 경로입니다.

관련 정보는 로그 구성 다시 실행을 참조하세요.

- INNODB_LOG_SPIN_CPU_ABS_LWM

명령줄 형식	--innodb-log-spin-cpu-abs-lwm=#
시스템 변수	INNODB_LOG_SPIN_CPU_ABS_LWM
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	80
최소값	0
최대 값	4294967295

플러시 재실행을 기다리는 동안 사용자 스레드가 더 이상 회전하지 않는 최소 CPU 사용량을 정의합니다. 이 값은 CPU 코어 사용량의 합계로 표시됩니다. 예를 들어 기본값인 80은 단일 CPU 코어의 80%입니다. 멀티 코어 프로세서가 있는 시스템에서 값 150은 하나의 CPU 코어 사용량 100%에 두 번째 CPU 코어 사용량 50%를 더한 값입니다.

관련 정보는 섹션 8.5.4, 'InnoDB 재실행 로깅 최적화'를 참조하세요.

- INNODB_LOG_SPIN_CPU_PCT_HWM

명령줄 형식	--innodb-log-spin-cpu-pct-hwm=#
시스템 변수	INNODB_LOG_SPIN_CPU_PCT_HWM
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	50
최소값	0
최대 값	100

플러시 재실행을 기다리는 동안 사용자 스레드가 더 이상 회전하지 않는 최대 CPU 사용량을 정의합니다. 이 값은 모든 CPU 코어의 총 처리 능력을 합한 값의 백분율로 표시됩니다. 기본값은 50%입니다. 예를 들어 CPU 코어 2개 사용 100%는 CPU 코어가 4개인 서버에서 CPU 코어를 합친 처리 능력의 50%입니다.

innodb_log_spin_cpu_pct_hwm 변수는 프로세서 선호도를 고려합니다. 예를 들어 서버에 48개의 코어가 있지만 mysqld 프로세스가 4개의 CPU 코어에만 고정되어 있는 경우, 나머지 44개의 CPU 코어는 무시됩니다.

관련 정보는 섹션 8.5.4, 'InnoDB 재실행 로깅 최적화'를 참조하세요.

- INNODB_LOG_WAIT_FOR_FLUSH_SPIN_HWM

명령줄 형식	`--innodb-log-wait-for-flush-spin-hwm=#`
시스템 변수	`INNODB_LOG_WAIT_FOR_FLUSH_SPIN_HWM`
범위	글로벌
동적	예
`SET_VAR` 힌트 적용	아니요
유형	Integer

관련 정보는 로그 구성 다시 실행을 참조하세요.

기본값	400
최소값	0
최대값(64비트 플랫폼)	2**64-1
최대값(32비트 플랫폼)	2**32-1
단위	마이크로초

플러시된 재실행을 기다리는 동안 사용자 스레드가 더 이상 회전하지 않는 최대 평균 로그 플러시 시간을 정의합니다. 기본값은 400마이크로초입니다.

관련 정보는 섹션 8.5.4, 'InnoDB 재실행 로깅 최적화'를 참조하세요.

- innodb_log_write_ahead_size

명령줄 형식	--innodb-log-write-ahead-size=#
시스템 변수	innodb_log_write_ahead_size
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	8192
최소값	512(로그 파일 블록 크기)
최대 값	innodb_page_size와 같음
단위	바이트

재실행 로그의 쓰기 미리 쓰기 블록 크기를 바이트 단위로 정의합니다. "읽기-온-쓰기"를 방지하려면 운영 체제 또는 파일 시스템 캐시 블록 크기와 일치하도록 innodb_log_write_ahead_size를 설정합니다. 기본 설정은 8192바이트입니다. 읽기-온-쓰기는 재실행 로그의 쓰기 미리 쓰기 블록 크기와 운영 체제 또는 파일 시스템 캐시 블록 크기가 일치하지 않아서 재실행 로그 블록이 운영 체제 또는 파일 시스템에 완전히 캐시되지 않을 때 발생합니다.

innodb_log_write_ahead_size의 유효한 값은 InnoDB 로그 파일 블록 크기(2^n)의 배수입니다. 최소값은 InnoDB 로그 파일 블록 크기(512)입니다. 최소값을 지정하면 미리 쓰기가 수행되지 않습니다. 최대값은 innodb_page_size 값과 같습니다. innodb_log_write_ahead_size 값을 innodb_page_size 값보다 큰 값으로지정하면 innodb_log_write_ahead_size 설정이 innodb_page_size 값으로 잘립니다.

운영 체제 또는 파일 시스템 캐시 블록 크기에 비해 innodb_log_write_ahead_size 값을 너무 낮게 설정하면 "읽기-온-쓰기"가 발생합니다. 값을 너무 높게 설정하면 한 번에 여러 블록이 쓰여지기 때문에 로그 파일 쓰기에 대한 fsync 성능에 약간의 영향을 미칠 수 있습니다.

관련 정보는 섹션 8.5.4, 'InnoDB 재실행 로깅 최적화'를 참조하세요.

- innodb_log_writer_threads

명령줄 형식	--innodb-log-writer-threads[={OFF|ON}]
시스템 변수	innodb_log_writer_threads
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	부울

기본값	켜기

로그 버퍼에서 시스템 버퍼로 재실행 로그 레코드를 쓰고 시스템 버퍼를 재실행 로그 파일로 플러시하기 위한 전용 로그 작성자 스레드를 활성화합니다. 전용 로그 작성자 스레드는 동시성이 높은 시스템에서 성능을 향상시킬 수 있지만 동시성이 낮은 시스템에서는 전용 로그 작성자 스레드를 비활성화하는 것이 더 나은 성능을 제공합니다.

자세한 내용은 섹션 8.5.4, 'InnoDB 재실행 로깅 최적화'를 참조하세요.

- `INNODB_LRU_SCAN-DEPTH`

명령줄 형식	`--innodb-lru-scan-depth=#`
시스템 변수	`INNODB_LRU_SCAN-DEPTH`
범위	글로벌
동적	예
`SET_VAR` 힌트 적용	아니요
유형	Integer
기본값	`1024`
최소값	`100`
최대값(64비트 플랫폼)	`2**64-1`
최대값(32비트 플랫폼)	`2**32-1`

`InnoDB` 버퍼 풀의 플러시 작업에 대한 알고리즘 및 휴리스틱에 영향을 미치는 매개변수입니다. 주로 I/O 집약적인 워크로드를 튜닝하는 성능 전문가가 관심을 가질 수 있습니다. 버퍼 풀 인스턴스별로 페이지 클리너 스레드가 플러시할 더티 페이지를 찾기 위해 버퍼 풀 LRU 페이지 목록의 어느 정도까지 스캔할지 지정합니다. 이 작업은 초당 한 번 수행되는 백그라운드 작업입니다.

일반적으로 기본값보다 작은 설정이 대부분의 워크로드에 적합합니다. 필요한 것보다 훨씬 높은 값은 성능에 영향을 줄 수 있습니다. 일반적인 워크로드에서 여유 I/O 용량이 있는 경우에만 값을 늘리는 것을 고려하세요. 반대로 쓰기 집약적인 워크로드로 인해 I/O 용량이 포화 상태인 경우, 특히 버퍼 풀이 큰 경우에는 값을 줄이세요.

`innodb_lru_scan_depth`를 조정할 때는 낮은 값으로 시작하여 여유 페이지가 거의 표시되지 않는 것을 목표로 설정을 상향 조정합니다. 또한, 버퍼 풀 인스턴스 수를 변경할 때 `innodb_lru_scan_depth` * `innodb_buffer_pool_instances`는 페이지 클리너 스레드가 매초 수행하는 작업량을 정의하므로, 이 값을 조정하는 것도 고려하세요.

관련 정보는 섹션 15.8.3.5, "버퍼 풀 플러싱 구성"을 참조하십시오. 일반적인 I/O 튜닝에 대한 조언은 섹션 8.5.8, "InnoDB 디스크 I/O 최적화"를 참조하세요.

- innodb_max_dirty_pages_pct

명령줄 형식	--innodb-max-dirty-pages-pct=#
시스템 변수	innodb_max_dirty_pages_pct
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	숫자
기본값	90
최소값	0

최대 값	99.99

InnoDB는 더티 페이지의 비율이 이 값을 초과하지 않도록 버퍼 풀에서 데이터를 플러시하려고 시도합니다.

innodb_max_dirty_pages_pct 설정은 플러싱 활동의 목표를 설정합니다. 플러싱 속도에는 영향을 미치지 않습니다. 플러시 속도 관리에 대한 자세한 내용은 15.8.3.5절. "버퍼 풀 플러시 구성"을 참조하세요.

관련 정보는 섹션 15.8.3.5, "버퍼 풀 플러싱 구성"을 참조하십시오. 일반적인 I/O 튜닝에 대한 조언은 섹션 8.5.8, "InnoDB 디스크 I/O 최적화"를 참조하세요.

- INNODB_MAX_DIRTY_PAGES_PCT_LWM

명령줄 형식	--innodb-max-dirty-pages-pct-lwm=#
시스템 변수	INNODB_MAX_DIRTY_PAGES_PCT_LWM
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	숫자
기본값	10
최소값	0
최대 값	99.99

더티 페이지 비율을 제어하기 위해 사전 플러싱이 활성화되는 더티 페이지의 비율을 나타내는 낮은 물 표시를 정의합니다. 값이 0이면 사전 플러싱 동작이 완전히 비활성화됩니다. 구성된 값은 항상 innodb_max_dirty_pages_pct 값보다 낮아야 합니다. 자세한 내용은 섹션 15.8.3.5, "버퍼 풀 플러시 구성"을 참조하세요.

- INNODB_MAX_PURGE_LAG

명령줄 형식	--innodb-max-purge-lag=#
시스템 변수	INNODB_MAX_PURGE_LAG
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	0
최소값	0
최대 값	4294967295

원하는 최대 퍼지 지연을 정의합니다. 이 값을 초과하면 퍼지가 따라잡을 수 있는 시간을 확보하기 위해 INSERT, UPDATE 및 DELETE 작업에 지연이 적용됩니다. 기본값은 0으로, 최대 퍼지 지연이 없고 지연

이 발생하지 않습니다.

자세한 내용은 섹션 15.8.9, "퍼지 구성"을 참조하세요.

- `innodb_max_purge_lag_delay`

명령줄 형식	`--innodb-max-purge-lag-delay=#`
시스템 변수	`innodb_max_purge_lag_delay`
범위	글로벌

동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	0
최소값	0
최대 값	10000000
단위	마이크로초

innodb_max_purge_lag 임계값을 초과할 때 부과되는 지연의 최대 지연을 마이크로초 단위로 지정합니다. 지정된 innodb_max_purge_lag_delay 값은 innodb_max_purge_lag 공식에 의해 계산된 지연 기간의 상한입니다.

자세한 내용은 섹션 15.8.9, "퍼지 구성"을 참조하세요.

- innodb_max_undo_log_size

명령줄 형식	--innodb-max-undo-log-size=#
시스템 변수	innodb_max_undo_log_size
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	1073741824
최소값	10485760
최대 값	2**64-1
단위	바이트

실행 취소 테이블스페이스의 임계값 크기를 정의합니다. 실행 취소 테이블스페이스가 임계값을 초과하는 경우 innodb_undo_log_truncate가 활성화될 때 잘라내도록 표시될 수 있습니다. 기본값은 1073741824 바이트 (1024 MiB)입니다.

자세한 내용은 실행 취소 테이블 공간 잘라내기를 참조하십시오.

- innodb_merge_threshold_set_all_debug

명령줄 형식	--innodb-merge-threshold-set-all-debug=#
시스템 변수	innodb_merge_threshold_set_all_debug
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요

유형	Integer
기본값	50
최소값	1
최대 값	50

현재 딕셔너리 캐시에 있는 모든 인덱스에 대해 현재 MERGE_THRESHOLD 설정을 재정의하는 인덱스 페이지에 대한 페이지 전체 백분율 값을 정의합니다. 이 옵션은 디버깅 지원이 WITH_DEBUG CMake 옵션을 사용하여 컴파일된 경우에만 사용할 수 있습니다. 관련 정보는 섹션 15.8.11, "인덱스 페이지에 대한 병합 임계값 구성"을 참조하세요.

- innodb_monitor_disable

명령줄 형식	--innodb-monitor-disable={카운터| 모듈| 패턴|모두}
시스템 변수	innodb_monitor_disable
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	문자열

이 변수는 스위치 역할을 하여 InnoDB 메트릭 카운터를 비활성화합니다. 카운터 데이터는 정보 스키마 INNODB_METRICS 테이블을 사용하여 쿼리할 수 있습니다. 사용법에 대한 자세한 내용은 15.15.6절, "InnoDB 정보 스키마 메트릭 테이블"을 참조하세요.

innodb_monitor_disable='latch'는 SHOW ENGINE INNODB MUTEX에 대한 통계 수집을 비활성화합니다. 자세한 내용은 13.7.7.16절 "SHOW ENGINE 문"을 참고한다.

- innodb_monitor_enable

명령줄 형식	--innodb-monitor-enable={카운터| 모듈|패턴|모두}
시스템 변수	innodb_monitor_enable
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	문자열

이 변수는 스위치 역할을 하여 InnoDB 메트릭 카운터를 활성화합니다. 카운터 데이터는 정보 스키마 INNODB_METRICS 테이블을 사용하여 쿼리할 수 있습니다. 사용법에 대한 자세한 내용은 15.15.6절, "InnoDB 정보 스키마 메트릭 테이블"을 참조하세요.

innodb_monitor_enable='latch'는 SHOW ENGINE INNODB MUTEX에 대한 통계 수집을 활성화합니다. 자세한 내용은 13.7.7.16절 "SHOW ENGINE 문"을 참고한다.

- INNODB_MONITOR_RESET

명령줄 형식	--innodb-monitor-reset={카운터| 모듈|패턴|모두}
시스템 변수	INNODB_MONITOR_RESET

범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	열거형
기본값	NULL
유효한 값	카운터 모 듈 패턴

	모두

이 변수는 스위치 역할을 하여 InnoDB 메트릭 카운터의 카운트 값을 0으로 재설정합니다. 카운터 데이터는 정보 스키마 INNODB_METRICS 테이블을 사용하여 쿼리할 수 있습니다. 사용법에 대한 자세한 내용은 15.15.6절, "InnoDB 정보 스키마 메트릭 테이블"을 참조하세요.

innodb_monitor_reset='latch' SHOW ENGINE에서 보고한 통계를 INNODB MUTEX로 재설정합니다. 자세한 내용은 섹션 13.7.7.16, "SHOW ENGINE 문"을 참고한다.

- INNODB_MONITOR_RESET_ALL

명령줄 형식	--innodb-monitor-reset-all={카운터| 모듈|패턴|모두}
시스템 변수	INNODB_MONITOR_RESET_ALL
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	열거형
기본값	NULL
유효한 값	카운터 모 듈 패턴 모두

이 변수는 스위치 역할을 하여 InnoDB 메트릭 카운터의 모든 값(최소, 최대 등)을 재설정합니다. 카운터 데이터는 정보 스키마 INNODB_METRICS 테이블을 사용하여 쿼리할 수 있습니다. 사용법에 대한 자세한 내용은 15.15.6절, "InnoDB 정보 스키마 메트릭 테이블"을 참조하세요.

- innodb_numa_interleave

명령줄 형식	--innodb-numa-interleave[={OFF|ON}]
시스템 변수	innodb_numa_interleave
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	부울
기본값	꺼짐

InnoDB 버퍼 풀 할당을 위한 NUMA 인터리브 메모리 정책을 활성화합니다. innodb_numa_interleave가 활성화되면 mysqld 프로세스에 대한 NUMA 메모리 정책은 MPOL_INTERLEAVE로 설정됩니다. InnoDB 버퍼 풀이 할당된 후 NUMA 메모리 정책은 MPOL_DEFAULT로 다시 설정됩니다. innodb_numa_interleave 옵션을 사용할 수 있으려면 NUMA가 활성화된 Linux 시스템에서 MySQL을 컴파일해야 합니다.

CMake는 현재 플랫폼이 NUMA를 지원하는지 여부에 따라 기본 WITH_NUMA 값을 설정합니다. 자세한 내용은 섹션 2.8.7, "MySQL 소스 구성 옵션"을 참조하세요.

- INNODB_OLD_BLOCKS_PCT

명령줄 형식	--innodb-old-blocks-pct=#
시스템 변수	INNODB_OLD_BLOCKS_PCT
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	37
최소값	5
최대 값	95

이전 블록 하위 목록에 사용되는 InnoDB 버퍼 풀의 대략적인 비율을 지정합니다. 값 범위는 5~95입니다. 기본값은 37(즉, 풀의 3/8)입니다. 종종 innodb_old_blocks_time과 함께 사용됩니다.

자세한 내용은 섹션 15.8.3.3, "버퍼 풀 스캔 방지 기능 만들기"를 참조하세요. 버퍼 풀 관리, LRU 알고리즘 및 퇴거 정책에 대한 자세한 내용은 섹션 15.5.1, "버퍼 풀"을 참조하세요.

- INNODB_OLD_BLOCKS_TIME

명령줄 형식	--innodb-old-blocks-time=#
시스템 변수	INNODB_OLD_BLOCKS_TIME
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	1000
최소값	0
최대 값	2**32-1
단위	밀리초

0이 아닌 값은 전체 테이블 스캔 중과 같이 짧은 기간 동안만 참조되는 데이터로 인해 버퍼 풀이 채워지는 것을 방지합니다. 이 값을 높이면 전체 테이블 스캔이 버퍼 풀에 캐시된 데이터를 방해하지 않도록 보호할 수 있습니다.

이전 하위 목록에 삽입된 블록이 새 하위 목록으로 이동하기 전에 첫 번째 액세스 후 해당 위치에 있어야 하는 시간(밀리초)을 지정합니다. 값이 0이면 이전 하위 목록에 삽입된 블록은 삽입 후 얼마나 빨리 액세스가 발생했는지에 관계없이 처음 액세스할 때 즉시 새 하위 목록으로 이동합니다. 값이 0보다 크면 블록은 첫 번째 액

세스 후 최소 해당 밀리초 후에 액세스가 발생할 때까지 이전 하위 목록에 남아 있습니다. 예를 들어 값이 1000이면 블록은 첫 번째 액세스 후 1초 동안 이전 하위 목록에 머물다가 새 하위 목록으로 이동할 수 있는 자격을 얻게 됩니다.

기본값은 1000입니다.

이 변수는 종종 `innodb_old_blocks_pct`와 함께 사용됩니다. 자세한 내용은 15.8.3.3절, "버퍼 풀 스캔 저항성 만들기"를 참조한다. 버퍼 풀 관리, LRU 알고리즘 및 퇴출 정책에 대한 자세한 내용은 15.5.1절, "버퍼 풀"을 참조한다.

- INNODB_ONLINE_ALTER_LOG_MAX_SIZE

명령줄 형식	--innodb-online-alter-log-max-size=#
시스템 변수	INNODB_ONLINE_ALTER_LOG_MAX_SIZE
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	134217728
최소값	65536
최대 값	2**64-1
단위	바이트

InnoDB 테이블에 대한 온라인 DDL 작업 중에 사용되는 임시 로그 파일의 크기에 대한 상한을 바이트 단위로 지정합니다. 이러한 로그 파일은 생성되는 인덱스 또는 변경되는 테이블마다 하나씩 있습니다. 이 로그 파일은 DDL 작업 중에 테이블에 삽입, 업데이트 또는 삭제된 데이터를 저장합니다. 임시 로그 파일은 필요 시 innodb_sort_buffer_size 값에 따라 innodb_online_alter_log_max_size에 지정된 최대값 까 지 확장됩니다. 임시 로그 파일이
가 상한 크기 제한을 초과하면 테이블 변경 작업이 실패하고 커밋되지 않은 모든 동시 DML 작업이 롤백됩니다. 따라서 이 옵션의 값이 크면 한 번에 더 많은 DML이 발생할 수 있습니다.
온라인 DDL 작업뿐만 아니라 테이블이 잠겨 있을 때 DDL 작업이 끝날 때 로그의 데이터를 적용하기 위한 기간도 연장합니다.

- innodb_open_files

명령줄 형식	--innodb-open-files=#
시스템 변수	innodb_open_files
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	-1(자동 크기 조정을 의미하며, 이 리터럴 값을 할당하지 않음)
최소값	10

최대 값	2147483647

InnoDB가 한 번에 열 수 있는 최대 파일 수를 지정합니다. 최소값은 10입니다. innodb_file_per_table이 비활성화되어 있으면 기본값은 300이고, 그렇지 않으면 기본값 300 또는 table_open_cache 설정 중 더 높은 값이 사용됩니다.

innodb_open_files 제한은 SELECT innodb_set_open_files_limit(*N*) 문을 사용하여 런타임에 설정할 수 있으며, 여기서 N은 원하는 innodb_open_files 제한입니다(예시):

```
mysql> SELECT innodb_set_open_files_limit(1000);
```

이 문은 새 한도를 설정하는 저장 프로시저를 실행합니다. 프로시저가 성공하면 새로 설정된 한도 값을 반환하고, 그렇지 않으면 실패 메시지가 반환됩니다.

SET 문을 사용하여 innodb_open_files를 설정하는 것은 허용되지 않습니다. 런타임에 innodb_open_files를 설정하려면 위에서 설명한 SELECT innodb_set_open_files_limit(*N*) 문을 사용합니다.

innodb_open_files=default 설정은지원되지 않습니다. 정수 값만 허용됩니다.

비-LRU 관리 파일이 전체 innodb_open_files 한도를 소모하지 않도록 비-LRU 관리 파일은 이 한도의 90%로 제한되며, 이 중 10%는 LRU 관리 파일을 위해 예약되어 있습니다.

- innodb_optimize_fulltext_only

명령줄 형식	--innodb-optimize-fulltext-only[={OFF|ON}]
시스템 변수	innodb_optimize_fulltext_only
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	부울
기본값	꺼짐

InnoDB 테이블에서 테이블 최적화 작동 방식을 변경합니다. 전체 텍스트 인덱스가 있는 InnoDB 테이블에 대한 유지 관리 작업 중에 일시적으로 사용하도록 설정됩니다.

기본적으로 테이블 최적화는 테이블의 클러스터된 인덱스에서 데이터를 재구성합니다. 이 옵션을 활성화하면 테이블 최적화에서는 테이블 데이터 재구성을 건너뛰고 대신 InnoDB 전체 텍스트 인덱스에 대해 새로 추가, 삭제 및 업데이트된 토큰 데이터를 처리합니다. 자세한 내용은 InnoDB 전체 텍스트 인덱스 최적화를 참조하십시오.

- innodb_page_cleansers

명령줄 형식	--innodb-page-cleaners=#

InnoDB 시스템 변수

시스템 변수	innodb_page_cleansers
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	4
최소값	3231

최대 값	64

버퍼 풀 인스턴스에서 더티 페이지를 플러시하는 페이지 클리너 스레드의 수입니다. 페이지 클리너 스레드는 플러시 목록 및 LRU 플러시를 수행합니다. 페이지 클리너 스레드가 여러 개 있는 경우, 각 버퍼 풀 인스턴스에 대한 버퍼 풀 플러시 작업이 유휴 페이지 클리너 스레드로 전송됩니다. 기본값은 4이며, 페이지 클리너 스레드 수가 버퍼 풀 인스턴스 수를 초과하는 경우 innodb_page_cleaners는자동으로 innodb_buffer_pool_instances와 같은 값으로 설정됩니다.

워크로드가 버퍼 풀 인스턴스에서 데이터 파일로 더티 페이지를 플러시할 때 쓰기-IO에 묶여 있고 시스템 하드웨어에 사용 가능한 용량이 있는 경우, 페이지 클리너 스레드 수를 늘리면 쓰기-IO 처리량을 개선하는 데 도움이 될 수 있습니다.

멀티스레드 페이지 클리너 지원은 종료 및 복구 단계까지 확장됩니다.

setpriority() 시스템 호출은 지원되는 Linux 플랫폼에서 사용되며, mysqld 실행 사용자가 페이지 플러싱이 현재 워크로드에 보조를 맞출 수 있도록 페이지 클리너 스레드에 다른 MySQL 및 InnoDB 스레드보다 우선순위를 부여할 권한이 있는 경우 이 InnoDB 시작 메시지로 setpriority() 지원 여부를 알 수 있습니다:

```
[참고] InnoDB: mysqld 실행 사용자에게 권한이 있는 경우 페이지 클리너 스레드 우선순위
를 변경할 수 있습니다. setpriority()의 매뉴얼 페이지를 참조하세요.
```

서버 시작 및 종료가 systemd에 의해 관리되지 않는 시스템의 경우, /etc/security/limits.conf에서 mysqld 실행 사용자 권한을 구성할 수 있습니다. 예를 들어, mysql 사용자로 mysqld를 실행하는 경우 /etc/ security/limits.conf에 다음 줄을 추가하여 mysql 사용자에게 권한을 부여할 수 있습니다:

```
mysql               hard    nice       -20
mysql               소프트    nice       -20
```

시스템드 관리형 시스템의 경우, 로컬화된 시스템드 구성 파일에 LimitNICE=-20을 지정하여 동일한 효과를 얻을 수 있습니다. 예를 들어, /etc/systemd/system/mysqld.service.d/override.conf에 override.conf라는 파일을 생성하고 이 항목을 추가합니다:

```
[서비스] 제한 나
이스=-20
```

override.conf를 만들거나 변경한 후 systemd 구성을 다시 로드한 다음 systemd에 MySQL 서비스를 다시 시작하라고 지시합니다:

```
systemctl 데몬-로드

systemctl restart mysqldRPM 플랫폼
systemctl restart mysql   데비안 플랫폼
```

로컬라이즈된 systemd 구성 파일을 사용하는 방법에 대한 자세한 내용은 MySQL용 systemd 구성하기를 참조하세요.

`mysqld` 실행 사용자에게 권한을 부여한 후 `cat` 명령을 사용하여 구성된 나이스를 확인합니다.
제한을 설정합니다:

```
cat /proc/mysqld_pid/limits | grep nice
최대 좋은                     우선순위184467440737095516 18446744073709551596
```

- `innodb_page_size`

명령줄 형식	`--innodb-page-size=#`
시스템 변수	`innodb_page_size`
범위	글로벌
동적	아니요

SET_VAR 힌트 적용	아니요
유형	열거형
기본값	16384
유효한 값	4096 8192 16384 32768 65536

InnoDB 테이블스페이스의 페이지 크기를 지정합니다. 값은 바이트 또는 킬로바이트 단위로 지정할 수 있습니다. 예를 들어 16킬로바이트 페이지 크기 값은 16384, 16KB 또는 16k로 지정할 수 있습니다.

innodb_page_size는 MySQL 인스턴스를 초기화하기 전에만 구성할 수 있으며 이후에는 변경할 수 없습니다. 값을 지정하지 않으면 기본 페이지 크기를 사용하여 인스턴스가 초기화됩니다. 섹션 15.8.1, "InnoDB 시작 구성"을 참조하세요.

32KB 및 64KB 페이지 크기 모두에서 최대 행 길이는 약 16000바이트입니다. innodb_page_size가 32KB 또는 64KB로 설정된 경우 ROW_FORMAT=COMPRESSED는 지원되지 않습니다. innodb_page_size=32KB의 경우, 범위 크기는 2MB입니다. innodb_page_size=64KB의 경우, 범위 크기는 4MB입니다. 32KB 또는 64KB 페이지 크기를 사용하는 경우 innodb_log_buffer_size를 최소 16M(기본값)으로 설정해야 합니다.

기본 16KB 이상의 페이지 크기는 광범위한 워크로드, 특히 테이블 스캔과 관련된 쿼리 및 대량 업데이트가 포함된 DML 작업에 적합합니다. 단일 페이지에 많은 행이 포함되어 경합이 문제가 될 수 있는 소규모 쓰기가 많은 OLTP 워크로드에는 이보다 작은 페이지 크기가 더 효율적일 수 있습니다. 일반적으로 작은 블록 크기를 사용하는 SSD 저장 장치에서도 더 작은 페이지가 더 효율적일 수 있습니다. InnoDB 페이지 크기를 저장 장치 블록 크기에 가깝게 유지하면 디스크에 다시 쓰여지는 변경되지 않은 데이터의 양을 최소화할 수 있습니다.

첫 번째 시스템 테이블스페이스 데이터 파일(ibdata1)의 최소 파일 크기는 innodb_page_size 값에 따라 달라집니다. 자세한 내용은 innodb_data_file_path 옵션 설명을 참조하세요.

특정 InnoDB 페이지 크기를 사용하는 MySQL 인스턴스는 다른 페이지 크기를 사용하는 인스턴스의 데이터 파일이나 로그 파일을 사용할 수 없습니다.

일반적인 I/O 튜닝 조언은 섹션 8.5.8, "InnoDB 디스크 I/O 최적화"를 참조하세요.

- innodb_parallel_read_threads

명령줄 형식	--innodb-parallel-read-threads=#
시스템 변수	innodb_parallel_read_threads

범위	세션
동적	예
`SET_VAR` 힌트 적용	아니요
유형	Integer
기본값	`4`
최소값	`1`

최대 값	256

병렬 클러스터 인덱스 읽기에 사용할 수 있는 스레드 수를 정의합니다. 파티션의 병렬 스캔도 지원됩니다. 병렬 읽기 스레드를 사용하면 테이블 검사 성능을 향상시킬 수 있습니다. InnoDB는 CHECK TABLE 작업 중에 클러스터된 인덱스를 두 번 읽습니다. 두 번째 읽기는 병렬로 수행할 수 있습니다. 이 기능은 보조 인덱스 스캔에는 적용되지 않습니다. 병렬 클러스터 인덱스 읽기가 수행되려면 innodb_parallel_read_threads 세션 변수를 1보다 큰 값으로 설정해야 합니다. 병렬 클러스터 인덱스 읽기를 수행하는 데 사용되는 실제 스레드 수는 innodb_parallel_read_threads 설정 또는 스캔할 인덱스 하위 트리 수 중 더 작은 값에 의해 결정됩니다. 스캔 중에 버퍼 풀로 읽은 페이지는 버퍼 풀 LRU 목록의 맨 끝에 보관되므로 여유 버퍼 풀 페이지가 필요할 때 빠르게 삭제할 수 있습니다.

병렬 읽기 스레드의 최대 수(256개)는 모든 클라이언트 연결의 총 스레드 수입니다. 스레드 제한에 도달하면 연결은 단일 스레드 사용으로 되돌아갑니다.

- innodb_print_all_deadlock

명령줄 형식	--innodb-print-all-deadlock[={OFF|ON}]
시스템 변수	innodb_print_all_deadlock
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	부울
기본값	꺼짐

이 옵션을 활성화하면 InnoDB 사용자 트랜잭션의 모든 교착 상태에 대한 정보가 mysqld 오류 로그에 기록됩니다. 그렇지 않으면 마지막 교착 상태에 대한 정보만 표시됩니다.
SHOW ENGINE INNODB STATUS 명령을 실행합니다. InnoDB는 상태를 즉시 감지하고 트랜잭션 중 하나를 롤백하기 때문에 간헐적인 InnoDB 교착상태가반드시 문제가 되는 것은 아닙니다.
을 자동으로 실행합니다. 애플리케이션에 롤백을 감지하고 작업을 다시 시도할 수 있는 적절한 오류 처리 로직이 없는 경우 이 옵션을 사용하여 교착 상태가 발생하는 이유를 해결할 수 있습니다. A
교착 상태가 많으면 여러 테이블에 대해 DML 또는 SELECT ... FOR UPDATE 문을 실행하는 트랜잭션을 재구성하여 각 트랜잭션이 동일한 순서로 테이블에 액세스하여 교착 상태를 피해야 함을 나타낼 수 있습니다.

관련 정보는 섹션 15.7.5, "InnoDB의 교착 상태"를 참조하세요.

- innodb_print_ddl_logs

명령줄 형식	`--innodb-print-ddl-logs[={OFF|ON}]`
시스템 변수	`innodb_print_ddl_logs`
범위	글로벌
동적	예
`SET_VAR` 힌트 적용	아니요
유형	부울
기본값	꺼짐

이 옵션을 활성화하면 MySQL이 DDL 로그를 `stderr`에 기록합니다. 자세한 내용은 DDL 로그 보기를 참조하세요.

- INNODB_PURGE_BATCH_SIZE

명령줄 형식	--innodb-purge-batch-size=#
시스템 변수	INNODB_PURGE_BATCH_SIZE
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	300
최소값	1
최대 값	5000

기록 목록에서 구문 분석 및 처리를 일괄 처리하는 실행 취소 로그 페이지의 수를 정의합니다. 다중 스레드 제거 구성에서 코디네이터 제거 스레드는 다음을 나눕니다.

innodb_purge_batch_size를 innodb_purge_threads로 반환하고 각 제거 스레드에 해당 페이지 수를 할당합니다. 또한 innodb_purge_batch_size 변수는 실행 취소 로그를 128번 반복할 때마다 제거가 해제하는 실행 취소 로그 페이지 수를 정의합니다.

innodb_purge_batch_size 옵션은 innodb_purge_threads 설정과 함께 고급 성능 튜닝을 위한 것입니다. 대부분의 사용자는 innodb_purge_batch_size를 기본값에서 변경할 필요가 없습니다.

관련 정보는 섹션 15.8.9, "퍼지 구성"을 참조하세요.

- innodb_purge_threads

명령줄 형식	--innodb-purge-threads=#
시스템 변수	innodb_purge_threads
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	4
최소값	1
최대 값	32

InnoDB 제거 작업에 전용되는 백그라운드 스레드 수입니다. 값을 늘리면 퍼지 스레드가 추가로 생성되므로 여러 테이블에서 DML 작업이 수행되는 시스템에서 효율성이 향상될 수 있습니다.

관련 정보는 섹션 15.8.9, "퍼지 구성"을 참조하세요.

- INNODB_PURGE_RSEG_TRUNCATE_FREQUENCY

명령줄 형식	`--innodb-purge-rseg-truncate-frequency=#`
시스템 변수	`INNODB_PURGE_RSEG_TRUNCATE_FREQUENCY`
범위	글로벌
동적	예
`SET_VAR` 힌트 적용	아니요
유형	Integer

기본값	128
최소값	1
최대 값	128

퍼지 시스템이 롤백 세그먼트를 해제하는 빈도를 퍼지가 호출되는 횟수 측면에서 정의합니다. 실행 취소된 테이블스페이스는 해당 롤백 세그먼트가 해제될 때까지 잘릴 수 없습니다. 일반적으로 퍼지 시스템은 퍼지가 128회 호출될 때마다 롤백 세그먼트를 한 번씩 해제합니다. 기본값은 128입니다. 이 값을 줄이면 퍼지 스레드가 롤백 세그먼트를 해제하는 빈도가 증가합니다.

innodb_purge_rseg_truncate_frequency는 다음과 함께 사용하도록 설계되었습니다. `innodb_undo_log_truncate`. 자세한 내용은 실행 취소 테이블 공간 잘라내기를 참조하십시오.

- `INNODB_RANDOM_READ_Ahead`

명령줄 형식	`--innodb-random-read-ahead[={OFF|ON}]`
시스템 변수	`INNODB_RANDOM_READ_Ahead`
범위	글로벌
동적	예
`SET_VAR` 힌트 적용	아니요
유형	부울
기본값	꺼짐

`InnoDB` I/O 최적화를 위한 랜덤 읽기 선행 기술을 활성화합니다.

다양한 유형의 미리 읽기 요청에 대한 성능 고려 사항에 대한 자세한 내용은 다음을 참조하세요. 섹션 15.8.3.4, "InnoDB 버퍼 풀 프리페칭 구성(읽기 앞서)"을 참조하세요. 일반적인 I/O 튜닝 조언은 섹션 8.5.8, "InnoDB 디스크 I/O 최적화"를 참조하세요.

- `INNODB_READ_Ahead_THRESHOLD`

명령줄 형식	`--innodb-read-ahead-threshold=#`
시스템 변수	`INNODB_READ_Ahead_THRESHOLD`
범위	글로벌
동적	예
`SET_VAR` 힌트 적용	아니요
유형	Integer
기본값	56
최소값	0
최대 값	64

`InnoDB`가 페이지를 버퍼 풀로 미리 가져오는 데 사용하는 선형 미리 읽기의 민감도를 제어합니다. `InnoDB`가 한 범위(64페이지)에서 최소 `innodb_read_ahead_threshold` 페이지 이상을 순차적으로 읽으면

다음 범위 전체에 대해 비동기 읽기를 시작합니다. 허용되는 값의 범위는 0~64입니다. 값이 0이면 미리 읽기가 비활성화됩니다. 기본값이 56인 경우 InnoDB는 다음 범위에 대한 비동기 읽기를 시작하려면 한 범위에서 최소 56페이지를 순차적으로 읽어야 합니다.

미리 읽기 메커니즘을 통해 읽은 페이지 수와 이러한 페이지 중 액세스되지 않고 버퍼 풀에서 제거된 페이지 수를 알면 innodb_read_ahead_threshold 설정을 미세 조정할 때 유용할 수 있습니다. 엔진 상태 표시 출력은 Innodb_buffer_pool_read_ahead의 카운터 정보 및 Innodb_buffer_pool_read_ahead_evicted 글로벌 상태 변수를 보고하는

읽기 요청에 의해 버퍼 풀로 가져온 페이지 수와 액세스하지 않고 버퍼 풀에서 퇴거된 페이지 수를 각각 나타냅니다. 상태 변수는 마지막 서버 재시작 이후의 전역 값을 보고합니다.

또한 `SHOW ENGINE INNODB STATUS`는 미리 읽은 페이지가 읽혀지는 속도와 해당 페이지가 액세스되지 않고 제거되는 속도도 보여줍니다. 초당 평균은 SHOW `ENGINE INNODB STATUS`를 마지막으로 호출한 이후 수집된 통계를 기반으로 하며, `SHOW ENGINE INNODB STATUS` 출력의 버퍼 풀 및 메모리 섹션에 표시됩니다.

자세한 내용은 섹션 15.8.3.4, "InnoDB 버퍼 풀 프리페칭 구성(읽기 앞서)"을 참조하십시오. 일반적인 I/O 튜닝에 대한 조언은 8.5.8절, "InnoDB 디스크 I/O 최적화"를 참조하세요.

- `innodb_read_io_threads`

명령줄 형식	`--innodb-read-io-threads=#`
시스템 변수	`innodb_read_io_threads`
범위	글로벌
동적	아니요
`SET_VAR` 힌트 적용	아니요
유형	Integer
기본값	`4`
최소값	`1`
최대 값	`64`

`InnoDB`에서 읽기 작업을 위한 I/O 스레드 수입니다. 쓰기 스레드에 대응하는 값은 `innodb_write_io_threads`입니다. 자세한 내용은 섹션 15.8.5, "백그라운드 InnoDB I/O 스레드 수 구성"을 참조하세요. 일반적인 I/O 튜닝 조언은 섹션 8.5.8, "InnoDB 디스크 I/O 최적화"를 참조하십시오.

참고

Linux 시스템에서 `innodb_read_io_threads`에 대한 기본 설정으로 여러 MySQL 서버(일반적으로 12개 이상)를 실행합니다, `innodb_write_io_threads` 및 Linux `aio-max-nr` 설정이 시스템 제한을 초과할 수 있습니다. 이상적으로는 `aio-max-nr` 설정을 늘리는 것이 좋으며, 해결 방법으로 MySQL 중 하나 또는 둘의 설정을 줄일 수 있습니다.
변수를 사용합니다.

- `innodb_read_only`

명령줄 형식	`--innodb-read-only[={OFF|ON}]`
시스템 변수	`innodb_read_only`

범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	부울

기본값	꺼짐

읽기 전용 모드에서 InnoDB를 시작합니다. 데이터베이스 애플리케이션 또는 데이터 세트를 읽기 전용 미디어에 배포하는 데 사용됩니다. 데이터 웨어하우스에서 여러 인스턴스 간에 동일한 데이터 디렉터리를 공유하기 위해 사용할 수도 있습니다. 자세한 내용은 15.8.2절. "읽기 전용 작업을 위한 InnoDB 구성"을 참조하세요.

이전에는 innodb_read_only 시스템 변수를 활성화하면 InnoDB 스토리지 엔진에 대해서만 테이블을 생성하고 삭제할 수 없었습니다. MySQL 8.2부터 innodb_read_only를 활성화하면 모든 스토리지 엔진에 대해 이러한 작업을 방지할 수 있습니다. 모든 스토리지 엔진에 대한 테이블 생성 및 삭제 작업은 mysql 시스템 데이터베이스의 데이터 사전 테이블을 수정하지만, 이러한 테이블은 InnoDB 스토리지 엔진을 사용하므로 innodb_read_only가 활성화되어 있으면 수정할 수 없습니다. 데이터 사전 테이블을 수정해야 하는 다른 테이블 작업에도 동일한 원칙이 적용됩니다. 예시

- innodb_read_only 시스템 변수가 활성화된 경우, InnoDB를 사용하는 데이터 사전의 통계 테이블을 업데이트할 수 없기 때문에 ANALYZE TABLE이 실패할 수 있습니다. 키 분포를 업데이트하는 분석 테이블 작업의 경우 테이블 자체를 업데이트하는 경우에도 실패가 발생할 수 있습니다(예: MyISAM 테이블인 경우). 업데이트된 분포 통계를 얻으려면 information_schema_stats_expiry=0을 설정합니다.
- 데이터 사전에 저장된 스토리지 엔진 지정을 업데이트하기 때문에 *tbl_name* 테이블 엔진=엔진_이름 변경이 실패합니다.

또한 MySQL 시스템 데이터베이스의 다른 테이블은 MySQL의 InnoDB 스토리지 엔진을 사용합니다.
8.2. 이러한 테이블을 읽기 전용으로 설정하면 테이블을 수정하는 작업이 제한됩니다. 예시:

- 부여 테이블이 InnoDB를 사용하기 때문에 CREATE USER 및 GRANT와 같은 계정 관리 문이 실패합니다.
- mysql.plugin 시스템 테이블이 InnoDB를 사용하기 때문에 INSTALL PLUGIN 및 UNINSTALL PLUGIN 플러그인 관리 문이 실패합니다.
- mysql.func 시스템 테이블이 InnoDB를 사용하기 때문에 로드 가능한 함수 관리 문인 CREATE FUNCTION 및 DROP FUNCTION이 실패합니다.

- innodb_redo_log_archive_dirs

명령줄 형식	--innodb-redo-log-archive-dirs
시스템 변수	innodb_redo_log_archive_dirs
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	문자열

기본값	NULL

재실행 로그 아카이브 파일을 만들 수 있는 레이블이 지정된 디렉터리를 정의합니다. 세미콜론으로 구분된 목록으로 여러 개의 레이블이 지정된 디렉터리를 정의할 수 있습니다. 예를 들어

```
innodb_redo_log_archive_dirs='label1:/backups1;label2:/backups2'
```

레이블은 허용되지 않는 콜론(:)을 제외한 모든 문자 문자열을 사용할 수 있습니다. 빈 레이블도 허용되지만 이 경우에도 콜론(:)은 필수입니다.

경로를 지정해야 하며 디렉터리가 존재해야 합니다. 경로에는 콜론(':')을 포함할 수 있지만 세미콜론(;)은 허용되지 않습니다.

- innodb_redo_log_capacity

명령줄 형식	--innodb-redo-log-capacity=#
시스템 변수	innodb_redo_log_capacity
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	정수
기본값	104857600
최소값	8388608
최대 값	137438953472
단위	바이트

재실행 로그 파일이 차지하는 디스크 공간의 양을 정의합니다.

이 변수는 innodb_log_files_in_group 및 innodb_log_file_size 변수를 대체합니다. innodb_redo_log_capacity 설정이 정의된 경우 innodb_log_files_in_group 및 innodb_log_file_size 설정은 무시되며, 그렇지 않은 경우 이 설정이 innodb_redo_log_capacity 설정을 계산하는 데 사용됩니다. (innodb_log_files_in_group * innodb_log_file_size = innodb_redo_log_capacity). 이러한 변수가 설정되지 않은 경우, 재실행 로그 용량은 innodb_redo_log_capacity 기본값으로 설정됩니다.

자세한 내용은 섹션 15.6.5, '로그 다시 실행'을 참조하세요.

- innodb_redo_log_encrypt

명령줄 형식	--innodb-redo-log-encrypt[={OFF|ON}]
시스템 변수	innodb_redo_log_encrypt
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	부울
기본값	꺼짐

InnoDB 저장 데이터 암호화 기능을 사용하여 암호화된 테이블에 대한 재실행 로그 데이터의 암호화를 제어합니다. 재실행 로그 데이터 암호화는 기본적으로 비활성화되어 있습니다. 자세한 내용은 재실행 로그 암호화를 참조하십시오.

- innodb_replication_delay

명령줄 형식	--innodb-replication-delay=#

시스템 변수	innodb_replication_delay
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	0
최소값	0
최대 값	4294967295

단위	밀리초

innodb_thread_concurrency에 도달한 경우 복제 서버에서 복제 스레드 지연 시간(밀리초)입니다.

- innodb_rollback_on_timeout

명령줄 형식	--innodb-rollback-on-timeout[={OFF|ON}]
시스템 변수	innodb_rollback_on_timeout
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	부울
기본값	꺼짐

InnoDB는 기본적으로 트랜잭션 시간 초과 시 마지막 문만 롤백합니다. innodb- 롤백-온-타임아웃을 지정하면 트랜잭션 타임아웃이 발생하면 InnoDB가 전체 트랜잭션을 중단하고 롤백합니다.

자세한 내용은 섹션 15.21.5, "InnoDB 오류 처리"를 참조하세요.

- innodb_rollback_segments

명령줄 형식	--innodb-rollback-segments=#
시스템 변수	innodb_rollback_segments
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	128
최소값	1
최대 값	128

innodb_rollback_segments는 각 실행 취소 테이블 스페이스에 할당된 롤백 세그먼트 수와 실행 취소 레코드를 생성하는 트랜잭션에 대한 글로벌 임시 테이블 스페이스를 정의합니다. 각 롤백 세그먼트가 지원하는 트랜잭션의 수는 InnoDB 페이지 크기와 각 트랜잭션에 할당된 실행 취소 로그 수에 따라 달라집니다. 자세한 내용은 섹션 15.6.6, "실행 취소 로그"를 참조하세요.

관련 정보는 섹션 15.3, "InnoDB 다중 버전 관리"를 참조하십시오. 테이블 스페이스 실행 취소에 대한 자세한 내용은 15.6.3.4절. "테이블 스페이스 실행 취소"를 참조하십시오.

- innodb_saved_page_number_debug

	명령줄 형식	`--innodb-saved-page-number-debug=#`
	시스템 변수	`innodb_saved_page_number_debug`
	범위	글로벌
	동적	예
	`SET_VAR` 힌트 적용	아니요
3240	유형	Integer

기본값	0
최소값	0
최대 값	2**23-1

페이지 번호를 저장합니다. innodb_fil_make_page_dirty_debug 옵션을 설정하면 innodb_saved_page_number_debug로 정의된 페이지가 더럽혀집니다. innodb_saved_page_number_debug 옵션은 디버깅 지원이 WITH_DEBUG CMake 옵션을 사용하여 컴파일된 경우에만 사용할 수 있습니다.

- INNODB_SEGMENT_RESERVE_FACTOR

명령줄 형식	--innodb-segment-reserve-factor=#
시스템 변수	INNODB_SEGMENT_RESERVE_FACTOR
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	숫자
기본값	12.5
최소값	0.03
최대 값	40

빈 페이지로 예약된 테이블스페이스 파일 세그먼트 페이지의 비율을 정의합니다. 이 설정은 테이블별 파일 및 일반 테이블 스페이스에 적용할 수 있습니다. innodb_segment_reserve_factor 기본 설정은 12.5%이며, 이는 이전 MySQL 릴리스에서 예약된 페이지의 비율과 동일합니다.

자세한 내용은 예약된 파일 세그먼트 페이지의 비율 구성을 참조하세요.

- innodb_sort_buffer_size

명령줄 형식	--innodb-sort-buffer-size=#
시스템 변수	innodb_sort_buffer_size
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	1048576
최소값	65536
최대 값	67108864
단위	바이트

이 변수는 온라인 DDL 작업 중 동시 DML을 기록할 때 임시 로그 파일이 확장되는 양과 임시 로그 파일 읽

기 버퍼 및 쓰기 버퍼의 크기를 정의합니다.

자세한 내용은 섹션 15.12.3, '온라인 DDL 공간 요구 사항'을 참조하세요.

- INNODB_SPIN_WAIT_DELAY

명령줄 형식	--innodb-spin-wait-delay=#
시스템 변수	INNODB_SPIN_WAIT_DELAY
범위	글로벌

동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	6
최소값	0
최대 값	1000

스핀 잠금에 대한 폴링 사이의 최대 지연 시간입니다. 이 메커니즘의 하위 수준 구현은 하드웨어와 운영 체제의 조합에 따라 다르므로 지연이 고정된 시간 간격과 일치하지 않습니다.

innodb_spin_wait_pause_multiplier 변수와 함께 사용하면 스핀 잠금 폴링 지연 시간을 더욱 효과적으로 제어할 수 있습니다.

자세한 내용은 섹션 15.8.8, "스핀 잠금 폴링 구성하기"를 참조하세요.

- innodb_spin_wait_pause_multiplier

명령줄 형식	--innodb-spin-wait-pause-multiplier=#
시스템 변수	innodb_spin_wait_pause_multiplier
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	50
최소값	1
최대 값	100

스레드가 뮤텍스 또는 rw-잠금을 획득하기 위해 대기할 때 발생하는 스핀-대기 루프에서 PAUSE 명령의 수를 결정하는 데 사용되는 승수 값을 정의합니다.

자세한 내용은 섹션 15.8.8, "스핀 잠금 폴링 구성"을 참조하세요.

- innodb_stats_auto_recalc

명령줄 형식	--innodb-stats-auto-recalc[={OFF|ON}]
시스템 변수	innodb_stats_auto_recalc
범위	글로벌
동적	예

SET_VAR 힌트 적용	아니요
유형	부울
기본값	켜기

테이블의 데이터가 크게 변경된 후 InnoDB가영구 통계를 자동으로 다시 계산하도록 합니다. 임계값은 테이블 행의 10%입니다. 이 설정은 innodb_stats_persistent 옵션이 활성화된 경우 생성된 테이블에 적용됩니다. 자동 통계 재계산은 CREATE TABLE 또는 ALTER TABLE에서 STATS_PERSISTENT=1을 지정하여 구성할 수도 있습니다.

TABLE 문을 사용합니다. 통계를 생성하기 위해 샘플링되는 데이터의 양은 innodb_stats_perpetistent_sample_pages 변수.

자세한 내용은 섹션 15.8.10.1, '영구 옵티마이저 통계 매개변수 구성'을 참조하세요.

- innodb_stats_include_delete_marked

명령줄 형식	--innodb-stats-include-delete-marked[={OFF|ON}]
시스템 변수	innodb_stats_include_delete_marked
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	부울
기본값	꺼짐

기본적으로 InnoDB는 통계를 계산할 때 커밋되지 않은 데이터를 읽습니다. 테이블에서 행을 삭제하는 커밋되지 않은 트랜잭션의 경우 InnoDB는 행 추정치 및 인덱스 통계를 계산할 때 삭제 표시가 있는 레코드를 제외하므로, 테이블에서 동시에 운영 중인 다른 트랜잭션에 대해 READ UNCOMMITTED 이외의 트랜잭션 격리 수준을 사용하여 최적의 실행 계획이 되지 않을 수 있습니다. 이 시나리오를 방지하려면 innodb_stats_include_delete_marked를 활성화하여 InnoDB가 영구 옵티마이저 통계를 계산할 때 삭제 표시가 있는 레코드를 포함하도록 할 수 있습니다.

innodb_stats_include_delete_marked를 활성화하면 테이블 분석은 통계를 다시 계산할 때 삭제 표시가 있는 레코드를 고려합니다.

innodb_stats_include_delete_marked는 모든 InnoDB 테이블에 영향을 미치는 전역 설정입니다. 퍼시스턴트 옵티마이저 통계에만 적용됩니다.

관련 정보는 섹션 15.8.10.1, '영구 옵티마이저 통계 매개변수 구성'을 참조하세요.

- innodb_stats_method

명령줄 형식	--innodb-stats-method=값
시스템 변수	innodb_stats_method
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	열거형
기본값	널스_평등

유효한 값	NULLS_EQUAL NULLS_UNEQUAL nulls_ignored

서버가 InnoDB 테이블의 인덱스 값 분포에 대한 통계를 수집할 때 NULL 값을 처리하는 방식입니다. 허용되는 값은 nulls_equal, nulls_unequal 및 nulls_ignored입니다. nulls_equal의 경우, 모든 NULL 인덱스 값은 동일한 것으로 간주되며 NULL 값의 수와 동일한 크기의 단일 값 그룹을 형성합니다. nulls_unequal의 경우, NULL 값은 다음과 같이 간주됩니다.

같지 않으며, 각 NULL은 크기 1의 고유한 값 그룹을 형성합니다. nulls_ignored의 경우, NULL 값은 무시됩니다.

테이블 통계를 생성하는 데 사용되는 방법은 8.3.8절 "InnoDB 및 MyISAM 인덱스 통계 수집"에 설명된 대로 옵티마이저가 쿼리 실행을 위해 인덱스를 선택하는 방식에 영향을 줍니다.

- INNODB_STATS_ON_METADATA

명령줄 형식	--innodb-stats-on-metadata[={OFF|ON}]
시스템 변수	INNODB_STATS_ON_METADATA
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	부울
기본값	꺼짐

이 옵션은 옵티마이저 통계가 비영구적으로 구성된 경우에만 적용됩니다. innodb_stats_persistent가 비활성화되어 있거나 개별 테이블이 STATS_PERSISTENT=0으로 생성 또는 변경된 경우 옵티마이저 통계는 디스크에 지속되지 않습니다. 자세한 내용은 15.8.10.2절. "비영구 옵티마이저 통계 매개변수 구성"을 참조하세요.

innodb_stats_on_metadata를 활성화하면 InnoDB는 SHOW TABLE STATUS와 같은 메타데이터 문이나 정보 스키마 TABLES 또는 STATISTICS 테이블에 액세스할 때 비영구 통계를 업데이트합니다. (이러한 업데이트는 테이블 분석에서 발생하는 것과 유사합니다.) 이 설정을 비활성화하면 InnoDB는 이러한 작업 중에 통계를 업데이트하지 않습니다. 이 설정을 비활성화하면 테이블 또는 인덱스 수가 많은 스키마의 액세스 속도가 향상될 수 있습니다. 또한 InnoDB 테이블을 포함하는 쿼리에 대한 실행 계획의 안정성을 향상시킬 수 있습니다.

설정을 변경하려면 SET GLOBAL *innodb_stats_on_metadata=mode* 문을 실행합니다. 여기서 *모드는* ON 또는 OFF(또는 1 또는 0)입니다. 설정을 변경하려면 전역 시스템 변수를 설정할 수 있는 충분한 권한이 필요하며(5.1.9.1절. "시스템 변수 권한" 참조), 모든 연결의 작동에 즉시 영향을 미칩니다.

- INNODB_STATS_PERSISTENT

명령줄 형식	--innodb-stats-persistent[={OFF|ON}]
시스템 변수	INNODB_STATS_PERSISTENT
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	부울
기본값	켜기

InnoDB 인덱스 통계가 디스크에 유지되는지 여부를 지정합니다. 그렇지 않으면 통계가 자주 다시 계산되어

쿼리 실행 계획이 변경될 수 있습니다. 이 설정은 테이블이 생성될 때 각 테이블과 함께 저장됩니다. 테이블을 생성하기 전에 전역 수준에서 `innodb_stats_persistent`를 설정하거나, `CREATE TABLE` 및 `ALTER TABLE` 문의 `STATS_PERSISTENT` 절을 사용하여 시스템 전체 설정을 재정의하고 개별 테이블에 대한 영구 통계를 구성할 수 있습니다.

자세한 내용은 섹션 15.8.10.1, '영구 옵티마이저 통계 매개변수 구성'을 참조하세요.

- `innodb_stats_perpetistent_sample_pages`

명령줄 형식	--innodb-stats-persistent-sample-pages=#
시스템 변수	innodb_stats_perpetistent_sample_pages
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	20
최소값	1
최대 값	18446744073709551615

인덱스된 열의 카디널리티 및 기타 통계(예: 분석 테이블에서 계산된 통계)를 추정할 때 샘플링할 인덱스 페이지 수입니다. 이 값을 늘리면 인덱스 통계의 정확도가 향상되어 쿼리 실행 계획이 개선될 수 있지만 다음과 같은 비용이 발생합니다.

InnoDB 테이블에 대한 테이블 분석을 실행하는 동안 I/O가 증가했습니다. 자세한 내용은 섹션 15.8.10.1, "영구 옵티마이저 통계 매개변수 구성"을 참조하십시오.

참고

innodb_stats_persistent_sample_page에 높은 값 설정하기를 사용하면 분석 테이블 실행 시간이 길어질 수 있습니다. 분석 테이블이 액세스하는 데이터베이스 페이지 수를 추정하려면 15.8.10.3절. "InnoDB 테이블의 분석 테이블 복잡성 추정"을 참조하십시오.

innodb_stats_perpetual_sample_pages는 테이블에 대해 innodb_stats_perpetual이 활성화된 경우에만 적용되며, innodb_stats_perpetual이 비활성화된 경우에는 innodb_stats_transient_sample_pages가 대신 적용됩니다.

- innodb_stats_transient_sample_pages

명령줄 형식	--innodb-stats-transient-sample-pages=#
시스템 변수	innodb_stats_transient_sample_pages
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	8
최소값	1
최대 값	18446744073709551615

테이블 분석에서 계산된 것과 같이 인덱싱된 열의 카디널리티 및 기타 통계를 추정할 때 샘플링할 인덱스 페이지 수입니다. 기본값은 8입니다.
값은 인덱스 통계의 정확도를 향상시켜 쿼리 실행 계획을 개선할 수 있습니다.
InnoDB 테이블을 열거나 통계를 다시 계산할 때 I/O 증가로 인한 비용이 발생합니다. 자세한 내용은 섹션 15.8.10.2, '비영구 옵티마이저 통계 매개변수 구성'을 참조하세요.

참고

innodb_stats_transient_sample_pages의 값을 높게 설정하면 분석 테이블 실행 시간이 길어질 수 있습니다. 숫자를 추정하려면

> 분석 테이블이 액세스하는 데이터베이스 페이지의 수는 섹션 15.8.10.3, 'InnoDB 테이블에 대한 분석 테이블 복잡성 추정'을 참조하십시오.

테이블에 대해 innodb_stats_persistent가 비활성화된 경우에만 innodb_stats_transient_sample_pages가 적용되며, innodb_stats_persistent가 활성화된 경우에는 innodb_stats_persistent_sample_page가 대신 적용됩니다. innodb_stats_sample_pages를 대체합니다. 자세한 내용은 15.8.10.2절, "비영구 옵티마이저 통계 매개변수 구성"을 참조하세요.

- INNODB_STATUS_OUTPUT

명령줄 형식	--innodb-status-output[={OFF|ON}]
시스템 변수	INNODB_STATUS_OUTPUT
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	부울
기본값	꺼짐

표준 InnoDB 모니터에 대한 주기적 출력을 활성화 또는 비활성화합니다. 또한 innodb_status_output_lock과 함께 사용하여 InnoDB 잠금 모니터에 대한 주기적 출력을 활성화 또는 비활성화합니다. 자세한 내용은 섹션 15.17.2, "InnoDB 모니터 활성화"를 참조하세요.

- INNODB_STATUS_OUTPUT_LOCKS

명령줄 형식	--innodb-status-output-locks[={OFF|ON}]
시스템 변수	INNODB_STATUS_OUTPUT_LOCKS
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	부울
기본값	꺼짐

InnoDB 잠금 모니터를 활성화 또는 비활성화합니다. 활성화하면 InnoDB 잠금 모니터가 엔진 INNODB 상태 표시 출력 및 MySQL 오류 로그에 인쇄되는 주기적 출력에 잠금에 대한 추가 정보를 인쇄합니다. InnoDB 잠금 모니터에 대한 주기적 출력은 표준 InnoDB 모니터 출력의 일부로 인쇄됩니다. 따라서 InnoDB 잠금 모니터가 MySQL 오류 로그에 주기적으로 데이터를 인쇄하려면 표준 InnoDB 모니터를 활성화해야 합니다. 자세한 내용은 섹션 15.17.2, "InnoDB 모니터 활성화"를 참조하세요.

- innodb_strict_mode

명령줄 형식	`--innodb-strict-mode[={OFF|ON}]`
시스템 변수	`innodb_strict_mode`
범위	글로벌, 세션
동적	예
`SET_VAR` 힌트 적용	아니요
유형	부울

기본값	켜기

innodb_strict_mode를 활성화하면 InnoDB는 유효하지 않거나 호환되지 않는 테이블 옵션을 확인할 때 경고 대신 오류를 반환합니다.

키 블록 크기, 행 형식, 데이터 디렉토리, 임시 및 테이블 공간 옵션은 서로 및 다른 설정과 호환됩니다.

또한 테이블을 만들거나 변경할 때 행 크기 검사를 활성화하면 선택한 페이지 크기에 비해 레코드가 너무 커서 INSERT 또는 UPDATE가 실패하는 것을 방지할 수 있습니다.

mysqld를 시작할 때 명령줄에서 또는 MySQL 구성 파일에서 innodb_strict_mode를 활성화 또는 비활성화할 수 있습니다. 또한 런타임에 SET [GLOBAL|SESSION] *innodb_strict_mode=mode* 구문을 사용하여 *innodb_strict_mode*를 활성화 또는 비활성화할 수 있으며, 여기서 *모드는* ON 또는 OFF 중 하나입니다. GLOBAL 설정을 변경하려면 전역 시스템 변수를 설정할 수 있는 충분한 권한이 필요하며(5.1.9.1절, "시스템 변수 권한" 참조), 이후 연결되는 모든 클라이언트의 작동에 영향을 줍니다. 모든 클라이언트는 innodb_strict_mode에 대한 세션 설정을 변경할 수 있으며, 이 설정은 해당 클라이언트에만 영향을 줍니다.

이 시스템 변수의 세션 값을 설정하는 것은 제한된 작업입니다. 세션 사용자에게는 제한된 세션 변수를 설정할 수 있는 충분한 권한이 있어야 합니다. 5.1.9.1절. "시스템 변수 권한"을 참조하세요.

- innodb_sync_array_size

명령줄 형식	--innodb-sync-array-size=#
시스템 변수	innodb_sync_array_size
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	1
최소값	1
최대 값	1024

뮤텍스/잠금 대기 배열의 크기를 정의합니다. 값을 늘리면 스레드를 조정하는 데 사용되는 내부 데이터 구조가 분할되어 대기 스레드 수가 많은 워크로드에서 동시성을 높일 수 있습니다. 이 설정은 MySQL 인스턴스를 시작할 때 구성해야 하며 이후에는 변경할 수 없습니다. 일반적으로 768보다 큰 대기 스레드를 자주 생성하는 워크로드의 경우 이 값을 늘리는 것이 좋습니다.

- innodb_sync_spin_loops

명령줄 형식	`--innodb-sync-spin-loops=#`
시스템 변수	`innodb_sync_spin_loops`
범위	글로벌
동적	예
`SET_VAR` 힌트 적용	아니요
유형	Integer
기본값	`30`
최소값	`0`

최대 값	4294967295

스레드가 일시 중단되기 전에 스레드가 InnoDB 뮤텍스가 해제될 때까지 기다리는 횟수입니다.

- innodb_sync_debug

명령줄 형식	--innodb-sync-debug[={OFF|ON}]
시스템 변수	innodb_sync_debug
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	부울
기본값	꺼짐

InnoDB 스토리지 엔진에 대한 동기화 디버그 검사를 활성화합니다. 이 옵션은 디버깅 지원이 WITH_DEBUG CMake 옵션을 사용하여 컴파일된 경우에만 사용할 수 있습니다.

- innodb_table_locks

명령줄 형식	--innodb-table-locks[={OFF|ON}]
시스템 변수	innodb_table_locks
범위	글로벌, 세션
동적	예
SET_VAR 힌트 적용	아니요
유형	부울
기본값	켜기

자동 커밋이 0이면 InnoDB는 테이블 잠금을 준수하고, MySQL은 테이블 잠금에서 반환하지 않습니다. WRITE에서 다른 모든 스레드가 테이블에 대한 잠금을 모두 해제할 때까지 기다립니다. innodb_table_locks의 기본값은 1이며, 이는 autocommit = 0인 경우 LOCK TABLES가 InnoDB가 내부적으로 테이블을 잠그게 함을 의미합니다.

innodb_table_locks = 0은 테이블 잠금 ...으로 명시적으로 잠긴 테이블에는 영향을 미치지 않습니다. WRITE. 읽기 또는 쓰기를 위해 잠긴 테이블에는 영향을 미칩니다 (예 : 트리거를 통해 암시적으로 또는 LOCK TABLES ... WRITE 로 암시적으로 (예: 트리거를 통해) 또는 LOCK TABLES ... READ.

관련 정보는 섹션 15.7, "InnoDB 잠금 및 트랜잭션 모델"을 참조하세요.

- INNODB_TEMP_DATA_FILE_PATH

명령줄 형식	`--innodb-temp-data-file-` 경로=파일_이름
시스템 변수	`INNODB_TEMP_DATA_FILE_PATH`
범위	글로벌
동적	아니요
`SET_VAR` 힌트 적용	아니요
유형	문자열

기본값	ibtmp1:12M:자동 확장

전역 임시 테이블 스페이스 데이터 파일의 상대 경로, 이름, 크기 및 속성을 정의합니다. 글로벌 임시 테이블 스페이스는 사용자가 만든 임시 테이블의 변경 사항에 대한 롤백 세그먼트를 저장합니다.

innodb_temp_data_file_path에 값을 지정하지 않으면 기본 동작은 innodb_data_home_dir 디렉터리에 ibtmp1이라는 이름의 단일 자동 확장 데이터 파일을 생성하는 것입니다. 초기 파일 크기는 12MB보다 약간 큽니다.

글로벌 임시 테이블스페이스 데이터 파일 사양의 구문에는 파일 이름, 파일 크기, 자동 확장 및 최대 속성이 포함됩니다:

```
file_name:file_size[:autoextend[:max:max_file_size]]
```

글로벌 임시 테이블스페이스 데이터 파일은 다른 InnoDB 데이터 파일과 같은 이름을 가질 수 없습니다. 글로벌 임시 테이블스페이스 데이터 파일을 만들지 못하거나 오류가 발생하면 치명적인 것으로 간주되어 서버 시작이 거부됩니다.

파일 크기는 크기 값에 K, M 또는 G를 추가하여 KB, MB 또는 GB로 지정합니다. 파일 크기의 합계는 12MB보다 약간 커야 합니다.

개별 파일의 크기 제한은 운영 체제에 따라 결정됩니다. 대용량 파일을 지원하는 운영 체제에서는 파일 크기가 4GB를 초과할 수 있습니다. 글로벌 임시 테이블 스페이스 데이터 파일에 원시 디스크 파티션을 사용하는 것은 지원되지 않습니다.

자동 확장 및 최대 속성은 마지막에 지정된 데이터 파일에 대해서만 사용할 수 있습니다. innodb_temp_data_file_path 설정을 변경합니다. 예를 들어

```
[mysqld]
innodb_temp_data_file_path=ibtmp1:50M;ibtmp2:12M:autoextend:max:500M
```

자동 확장 옵션을 사용하면 여유 공간이 부족할 때 데이터 파일의 크기가 자동으로 증가합니다. 자동 확장 증분은 기본적으로 64MB입니다. 증분을 수정하려면 innodb_autoextend_increment 변수 설정을 변경합니다.

글로벌 임시 테이블스페이스 데이터 파일의 디렉터리 경로는 innodb_data_home_dir 및 innodb_temp_data_file_path로 정의된 경로를 연결하여 형성됩니다.

읽기 전용 모드로 InnoDB를 실행하기 전에 innodb_temp_data_file_path를 데이터 디렉터리 외부의 위치로 설정합니다. 경로는 데이터 디렉터리에 상대적이어야 합니다. 예를 들어

```
--innodb-temp-data-file-path=../../../tmp/ibtmp1:12M:autoextend
```

자세한 내용은 글로벌 임시 테이블 스페이스를 참조하세요.

- innodb_temp_tablespaces_dir

명령줄 형식	`--innodb-temp-tablespaces-dir=dir_name`
시스템 변수	`innodb_temp_tablespaces_dir`
범위	글로벌
동적	아니요
`SET_VAR` 힌트 적용	아니요
유형	디렉토리 이름

기본값	#innodb_temp

InnoDB가 시작할 때 세션 임시 테이블스페이스 풀을 생성하는 위치를 정의합니다. 기본 위치는 데이터 디렉터리의 #innodb_temp 디렉터리입니다. 정규화된 경로 또는 데이터 디렉터리를 기준으로 한 경로가 허용됩니다.

세션 임시 테이블 스페이스에는 항상 사용자가 생성한 임시 테이블과 InnoDB를 사용하여 옵티마이저가 생성한 내부 임시 테이블이 저장됩니다. (이전에는 내부 임시 테이블의 온디스크 저장소 엔진이 internal_tmp_disk_storage_engine 시스템 변수에 의해 결정되었지만 더 이상 지원되지 않습니다. 온디스크 내부 임시 테이블용 스토리지 엔진을 참조하세요.)

자세한 내용은 세션 임시 테이블 스페이스를 참조하세요.

- innodb_thread_concurrency

명령줄 형식	--innodb-thread-concurrency=#
시스템 변수	innodb_thread_concurrency
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	0
최소값	0
최대 값	1000

InnoDB 내에서 허용되는 최대 스레드 수를 정의합니다. 기본값인 0은 무한 동시성(제한 없음)으로 해석됩니다. 이 변수는 동시성이 높은 시스템에서 성능을 튜닝하기 위한 것입니다.

InnoDB는 InnoDB 내부의 스레드 수를 innodb_thread_concurrency 제한 이하로 유지하려고 시도합니다. 이 제한에 도달하면 대기 중인 스레드를 위해 추가 스레드가 "선입선출(FIFO)" 대기열에 배치됩니다. 잠금을 기다리는 스레드는 동시에 실행 중인 스레드 수에 포함되지 않습니다.

올바른 설정은 워크로드 및 컴퓨팅 환경에 따라 다릅니다. MySQL 인스턴스가 다른 애플리케이션과 CPU 리소스를 공유하거나 워크로드 또는 동시 사용자 수가 증가하는 경우 이 변수를 설정하는 것을 고려하세요. 다양한 값을 테스트하여 최상의 성능을 제공하는 설정을 결정합니다. innodb_thread_concurrency는 동적 변수이므로 라이브 테스트 시스템에서 다양한 설정으로 실험할 수 있습니다. 특정 설정의 성능이 좋지 않은 경우 innodb_thread_concurrency를 0으로 빠르게 다시 설정할 수 있습니다.

다음 가이드라인을 참고하여 적절한 설정을 찾고 유지하세요:

- 워크로드의 동시 사용자 스레드 수가 지속적으로 적고 성능에 영향을 미치지 않는 경우, `innodb_thread_concurrency=0`(제한 없음)을 설정합니다.

- 워크로드가 지속적으로 과중하거나 때때로 급증하는 경우, 최상의 성능을 제공하는 스레드 수를 찾을 때까지 `innodb_thread_concurrency` 값을 설정하고 조정합니다. 예를 들어, 일반적으로 시스템의 스레드 수가 40~100개라고 가정합니다.
  50명의 사용자로 시작하지만 주기적으로 60명, 70명 또는 그 이상으로 증가합니다. 테스트를 통해 동시 사용자 80명으로 제한해도 성능이 대체로 안정적으로 유지되는 것을 확인했습니다. 이 경우 `innodb_thread_concurrency`를 80으로 설정합니다.

- `InnoDB`가 사용자 스레드에 특정 개수 이상의 가상 CPU(예: 20개 가상 CPU)를 사용하지 않도록 하려면 `innodb_thread_concurrency`를 이 수로 설정(또는
  성능 테스트에 따라 더 낮음). MySQL을 다른 애플리케이션으로부터 격리하는 것이 목표인 경우,

mysqld 프로세스를 가상 CPU에만 바인딩하는 것을 고려하세요. 그러나 독점 바인딩은 mysqld 프로세스가 지속적으로 사용 중이 아닌 경우 하드웨어 사용량이 최적화되지 않을 수 있다는 점에 유의하십시오. 이 경우 mysqld 프로세스를 가상 CPU에 바인딩하되 다른 애플리케이션이 가상 CPU의 일부 또는 전부를 사용하도록 허용할 수 있습니다.

참고

운영 체제 관점에서 볼 때 리소스 관리 솔루션을 사용하여 애플리케이션 간에 CPU 시간을 공유하는 방법을 관리하는 것이 mysqld 프로세스를 바인딩하는 것보다 더 나을 수 있습니다. 예를 들어, 다른 중요 프로세스가 실행 중이 *아닐* 때는 특정 애플리케이션에 가상 CPU 시간의 90%를 할당하고, 다른 중요 프로세스가 실행 *중일* 때는 이 값을 40%로 다시 조정할 수 있습니다.

- 경우에 따라 최적의 innodb_thread_concurrency 설정이 가상 CPU 수보다 작을 수 있습니다.
- innodb_thread_concurrency 값이 너무 높으면 시스템 내부 및 리소스에 대한 경합이 증가하여 성능이 저하될 수 있습니다.
- 시스템을 정기적으로 모니터링하고 분석하세요. 워크로드, 사용자 수 또는 컴퓨팅 환경이 변경되면 innodb_thread_concurrency 설정을 조정해야 할 수 있습니다.

값이 0이면 InnoDB 내부의 쿼리와 큐 카운터의 쿼리가 비활성화됩니다.
엔진 이노드 상태 표시 출력의 행 작업 섹션을 선택합니다.

관련 정보는 15.8.4절, 'InnoDB의 스레드 동시성 구성'을 참조하세요.

- innodb_thread_sleep_delay

명령줄 형식	--innodb-thread-sleep-delay=#
시스템 변수	innodb_thread_sleep_delay
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	10000
최소값	0
최대 값	1000000
단위	마이크로초

InnoDB 대기열에 참여하기 전에 InnoDB 스레드가 절전 모드로 전환되는 시간(마이크로초)입니다. 기본값은 10000입니다. 값이 0이면 절전이 비활성화됩니다. innodb_adaptive_max_sleep_delay를 허용할 수 있는 가장 높은 값으로 설정하면 현재 스레드 스케줄링 활동에 따라 innodb_thread_sleep_delay를

자동으로 위 또는 아래로 조정합니다. 이 동적 조정은 시스템 부하가 적거나 최대 용량에 가깝게 작동하는 시간 동안 스레드 스케줄링 메커니즘이 원활하게 작동하도록 도와줍니다.

자세한 내용은 15.8.4절, 'InnoDB의 스레드 동시성 구성'을 참조하세요.

- `innodb_tmpdir`

명령줄 형식	`--innodb-tmpdir=dir_name`
시스템 변수	`innodb_tmpdir`
범위	글로벌, 세션

동적	예
SET_VAR 힌트 적용	아니요
유형	디렉토리 이름
기본값	NULL

온라인 ALTER TABLE 중에 생성된 임시 정렬 파일의 대체 디렉터리를 정의하는 데 사용됩니다.
테이블을 다시 작성하는 연산입니다.

테이블을 재구축하는 온라인 ALTER TABLE 작업은 원래 테이블과 동일한 디렉터리에 *중간* 테이블 파일도 생성합니다. innodb_tmpdir 옵션은 중간 테이블 파일에는 적용되지 않습니다.

유효한 값은 MySQL 데이터 디렉터리 경로 이외의 모든 디렉터리 경로입니다. 값이 NULL(기본값)이면 임시 파일이 MySQL 임시 디렉터리(Unix의 경우 $TMPDIR, %TEMP
또는 --tmpdir 구성 옵션으로 지정한 디렉터리). 디렉터리가 지정된 경우 디렉터리의 존재 여부와 사용 권한은 SET 문을 사용하여 innodb_tmpdir을 구성한 경우에만 확인됩니다. 디렉터리 문자열에 심볼릭 링크가 제공된 경우 심볼릭 링크는 다음과 같다.
를 확인하여 절대 경로로 저장합니다. 경로는 512바이트를 초과하지 않아야 합니다. innodb_tmpdir이 잘못된 디렉터리로 설정된 경우 온라인 ALTER TABLE 작업은 오류를 보고합니다. innodb_tmpdir은 온라인 ALTER TABLE 작업의 경우에만 MySQL tmpdir 설정을 재정의합니다.

innodb_tmpdir을 구성하려면 FILE 권한이 필요합니다.

innodb_tmpdir 옵션은 tmpfs 파일 시스템에 있는 임시 파일 디렉터리의 오버플로를 방지하기 위해 도입되었습니다. 이러한 오버플로는 테이블을 재구축하는 온라인 ALTER TABLE 작업 중에 생성된 대용량 임시 정렬 파일로 인해 발생할 수 있습니다.

복제 환경에서는 모든 서버의 운영 체제 환경이 동일한 경우에만 innodb_tmpdir 설정 복제를 고려하십시오. 그렇지 않으면 테이블을 재구성하는 온라인 ALTER TABLE 작업을 실행할 때 innodb_tmpdir 설정을 복제하면 복제 실패가 발생할 수 있습니다.
서버 운영 환경이 다른 경우 각 서버에서 innodb_tmpdir을 개별적으로 구성하는 것이 좋습니다.

자세한 내용은 섹션 15.12.3, "온라인 DDL 공간 요구 사항"을 참조하십시오. 온라인 ALTER TABLE 작업에 대한 자세한 내용은 15.12절. "InnoDB 및 온라인 DDL"을 참조하세요.

- innodb_trx_purge_view_update_only_debug

명령줄 형식	--innodb-trx-purge-view-update-only-debug[={OFF|ON}]
시스템 변수	innodb_trx_purge_view_update_only_debug
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	부울
기본값	꺼짐

제거 보기를 업데이트하는 동안 삭제 표시된 레코드의 제거를 일시 중지합니다. 이 옵션은 제거 보기가 업데이트되었지만 아직 제거가 수행되지 않은 상황을 인위적으로 만듭니다. 이 옵션은 디버깅 지원이 `WITH_DEBUG CMake` 옵션을 사용하여 컴파일된 경우에만 사용할 수 있습니다.

- `innodb_trx_rseg_n_slots_debug`

명령줄 형식	`--innodb-trx-rseg-n-slots-debug=#`

시스템 변수	innodb_trx_rseg_n_slots_debug
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	0
최소값	0
최대 값	1024

실행 취소 로그 세그먼트의 여유 슬롯을 찾는 trx_rsegf_undo_find_free 함수에 대해 TRX_RSEG_N_SLOTS를 지정된 값으로 제한하는 디버그 플래그를 설정합니다. 이 옵션은 디버깅 지원이 WITH_DEBUG CMake 옵션을 사용하여 컴파일된 경우에만 사용할 수 있습니다.

- innodb_undo_directory

명령줄 형식	--innodb-undo-directory=dir_name
시스템 변수	innodb_undo_directory
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	디렉토리 이름

InnoDB가 실행 취소 테이블스페이스를 생성하는 경로입니다. 일반적으로 실행 취소 테이블스페이스를 다른 저장 장치에 배치하는 데 사용됩니다.

기본값은 없습니다(NULL). innodb_undo_directory 변수가 정의되지 않은 경우 데이터 디렉터리에 실행 취소 테이블스페이스가 생성됩니다.

MySQL 인스턴스가 초기화될 때 생성되는 기본 실행 취소 테이블스페이스 (innodb_undo_001 및 innodb_undo_002)는 항상 innodb_undo_directory 변수에 의해 정의된 디렉터리에 있습니다.

다른 경로가 지정되지 않은 경우 CREATE UNDO TABLESPACE 구문을 사용하여 생성된 실행 취소 테이블스페이스는 innodb_undo_directory 변수에 의해 정의된 디렉터리에 생성됩니다.

자세한 내용은 섹션 15.6.3.4, '테이블 공간 실행 취소'를 참조하세요.

- innodb_undo_log_encrypt

명령줄 형식	--innodb-undo-log-encrypt[={OFF|ON}]

시스템 변수	innodb_undo_log_encrypt
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	부울
기본값	꺼짐

InnoDB 저장 데이터 암호화 기능을 사용하여 암호화된 테이블에 대한 실행 취소 로그 데이터의 암호화를 제어합니다. 별도의 실행 취소 테이블 공간에 있는 실행 취소 로그에만 적용됩니다. 섹션 15.6.3.4, "테이블 스페이스 실행 취소"를 참조하십시오. 시스템 테이블스페이스에 있는 실행 취소 로그 데이터에는 암호화가 지원되지 않습니다. 자세한 내용은 로그 암호화 실행 취소를 참조하세요.

- innodb_undo_log_truncate

명령줄 형식	--innodb-undo-log-truncate[={OFF|ON}]
시스템 변수	innodb_undo_log_truncate
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	부울
기본값	켜기

이 옵션을 활성화하면 innodb_max_undo_log_size에 정의된 임계값을 초과하는 실행 취소 테이블스페이스는 잘림을 위해 표시됩니다. 실행 취소 테이블스페이스만 잘릴 수 있습니다. 시스템 테이블스페이스에 있는 실행 취소 로그의 잘림은 지원되지 않습니다. 잘라내기를 수행하려면 실행 취소 테이블스페이스가 두 개 이상 있어야 합니다.

innodb_purge_rseg_truncate_frequency 변수를 사용하여 실행 취소된 테이블 스페이스의 잘림을 신속하게 처리할 수 있습니다.

자세한 내용은 실행 취소 테이블 공간 잘라내기를 참조하십시오.

- innodb_undo_tablespaces

명령줄 형식	--innodb-undo-tablespaces=#
사용 중단	예
시스템 변수	innodb_undo_tablespaces
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	2
최소값	2
최대 값	127

InnoDB에서 사용하는 실행 취소 테이블 공간의 수를 정의합니다. 기본값 및 최소값은 2입니다.

참고

innodb_undo_tablespaces 변수는 더 이상 사용되지 않으며, MySQL 8.2에서는 이 변수를 설정해도 아무런 효과가 없습니다. 향후 MySQL 릴리스에서 이 변수를 제거해야 합니다.

자세한 내용은 섹션 15.6.3.4, '테이블 공간 실행 취소'를 참조하세요.

- INNODB_USE_FDATASYNC

명령줄 형식	--innodb-use-fdatasync[={OFF|ON}]
시스템 변수	INNODB_USE_FDATASYNC
범위	글로벌
동적	예
SET_VAR 힌트 적용	아니요
유형	부울

기본값	꺼짐

`fdatasync()` 시스템 호출을 지원하는 플랫폼에서 `innodb_use_fdatasync` 변수를 활성화하면 운영 체제 플러시 시 `fsync()` 시스템 호출 대신 `fdatasync()` 사용을 허용할 수 있습니다. 후속 데이터 검색에 필요한 경우가 아니라면 `fdatasync()` 호출은 파일 메타데이터의 변경 사항을 플러시하지 않으므로 잠재적인 성능 이점을 제공합니다.

`fsync`, `O_DSYNC` 및 `O_DIRECT`와 같은 `innodb_flush_method` 설정의 하위 집합은 `fsync()` 시스템 호출을 사용합니다. 이러한 설정을 사용할 때 `innodb_use_fdatasync` 변수를 사용할 수 있습니다.

- `INNODB_USE_NATIVE_AIO`

명령줄 형식	`--innodb-use-native-aio[={OFF|ON}]`
시스템 변수	`INNODB_USE_NATIVE_AIO`
범위	글로벌
동적	아니요
`SET_VAR` 힌트 적용	아니요
유형	부울
기본값	켜기

Linux 비동기 I/O 하위 시스템을 사용할지 여부를 지정합니다. 이 변수는 Linux 시스템에만 적용되며 서버가 실행되는 동안에는 변경할 수 없습니다. 일반적으로 이 옵션은 기본적으로 사용하도록 설정되어 있으므로 구성할 필요가 없습니다.

Windows 시스템에서 InnoDB가 제공하는 비동기 I/O 기능을 Linux 시스템에서도 사용할 수 있습니다. (다른 유닉스 계열 시스템은 계속 동기식 I/O 호출을 사용합니다.) 이 기능은 일반적으로 `SHOW ENGINE INNODB STATUS\G` 출력에 보류 중인 읽기/쓰기가 많이 표시되는 I/O가 많은 시스템의 확장성을 개선합니다.

많은 수의 `InnoDB` I/O 스레드로 실행하는 경우, 특히 동일한 서버 시스템에서 이러한 인스턴스를 여러 개 실행하는 경우 Linux 시스템에서 용량 제한을 초과할 수 있습니다. 이 경우 다음과 같은 오류가 발생할 수 있습니다:

```
다시: 지정된 최대 이벤트가 사용자의 사용 가능한 이벤트 한도를 초과합니다.
```

일반적으로 이 오류는 `/proc/sys/fs/aio-max-nr`에 더 높은 제한을 작성하여 해결할 수 있습니다.

그러나 OS의 비동기 I/O 하위 시스템 문제로 인해 InnoDB가 시작되지 않는 경우 `innodb_use_native_aio=0`으로 서버를 시작할 수 있습니다. InnoDB가 `tmpdir` 위치, `tmpfs` 파일 시스템, `tmpfs`에서 AIO를 지원하지 않는 Linux 커널의 조합과 같은 잠재적인 문제를 감지하는 경우 이 옵션은 시작 중에 자동으로 비활성화될 수도 있습니다.

자세한 내용은 섹션 15.8.6, 'Linux에서 비동기 I/O 사용'을 참조하세요.

- innodb_validate_tablespace_paths

명령줄 형식	--innodb-validate-tablespace-paths[={OFF|ON}]
시스템 변수	innodb_validate_tablespace_paths
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	부울
기본값	켜기

테이블스페이스 파일 경로 유효성 검사를 제어합니다. InnoDB는 시작 시 테이블스페이스 파일이 다른 위치로 이동된 경우 데이터 사전에 저장된 테이블스페이스 파일 경로와 비교하여 알려진 테이블스페이스 파일의 경로를 검증합니다. innodb_validate_tablespace_paths 변수를 사용하면 테이블스페이스 경로 유효성 검사를 비활성화할 수 있습니다. 이 기능은 테이블스페이스 파일이 이동되지 않는 환경을 위한 것입니다. 경로 유효성 검사를 비활성화하면 테이블스페이스 파일이 많은 시스템에서 시작 시간이 향상됩니다.

경고

테이블스페이스 파일을 이동한 후 테이블스페이스 경로 유효성 검사를 비활성화한 상태로 서버를 시작하면 정의되지 않은 동작이 발생할 수 있습니다.

자세한 내용은 섹션 15.6.3.7, "테이블 공간 경로 유효성 검사 비활성화"를 참조하십시오.

- innodb_version

  InnoDB 버전 번호입니다. MySQL 8.2에서는 InnoDB에 대한 별도의 버전 번호가 적용되지 않으며 이 값은 서버의 버전 번호와 동일합니다.

- innodb_write_io_threads

명령줄 형식	--innodb-write-io-threads=#
시스템 변수	innodb_write_io_threads
범위	글로벌
동적	아니요
SET_VAR 힌트 적용	아니요
유형	Integer
기본값	4
최소값	1
최대 값	64

InnoDB에서 쓰기 작업을 위한 I/O 스레드 수입니다. 기본값은 4입니다. 읽기 스레드에 대응하는 값은 innodb_read_io_threads입니다. 자세한 내용은 섹션 15.8.5, "백그라운드 InnoDB I/O 스레드 수 구성"을 참조하세요. 일반적인 I/O 튜닝에 대한 조언은 섹션 8.5.8, "InnoDB 디스크 I/O 최적화"를 참조하십시오.

참고

Linux 시스템에서 innodb_read_io_threads에 대한 기본 설정으로 여러 MySQL 서버(일반적으로 12개 이상)를 실행합니다, innodb_write_io_threads 및 Linux aio-max-nr 설정이 시스템 제한을 초과할 수 있습니다. 이상적으로는 aio-max-nr 설정을 늘리는 것이 좋으

며, 해결 방법으로 MySQL 중 하나 또는 둘의 설정을 줄일 수 있습니다.

변수를 사용합니다.

또한 바이너리 로그를 디스크에 동기화하는 것을 제어하는 `sync_binlog`의 값도 고려하세요.

일반적인 I/O 튜닝 조언은 섹션 8.5.8, "InnoDB 디스크 I/O 최적화"를 참조하세요.

15.15 InnoDB 정보_화학 테이블

이 섹션에서는 `InnoDB INFO_SCHEMA` 테이블에 대한 정보 및 사용 예제를 제공합니다.

InnoDB INFO_SCHEMA 테이블은 InnoDB 스토리지 엔진의 다양한 측면에 대한 메타데이터, 상태 정보 및 통계를 제공합니다. INFO_SCHEMA 데이터베이스에서 SHOW TABLES 문을 실행하여 InnoDB INFO_SCHEMA 테이블 목록을 확인할 수 있습니다:

```
mysql> SHOW TABLES FROM INFORMATION_SCHEMA LIKE 'INNODB%';
```

테이블 정의는 섹션 26.4, "INFORMATION_SCHEMA InnoDB 테이블"을 참조하세요. MySQL INFORMATION_SCHEMA 데이터베이스에 관한 일반적인 정보는 26장, *INFORMATION_SCHEMA 테이블을* 참조하세요.

15.15.1 압축에 대한 InnoDB 정보_SCHEMA 테이블

압축에 대한 두 쌍의 InnoDB INFO_SCHEMA 테이블은 압축이 전반적으로 얼마나 잘 작동하는지에 대한 인사이트를 제공할 수 있습니다:

- INNODB_CMP 및 INNODB_CMP_RESET은 압축 작업 횟수 및 압축 수행에 소요된 시간에 대한 정보를 제공합니다.
- INNODB_CMPMEM 및 INNODB_CMPMEM_RESET은 압축을 위해 메모리가 할당되는 방식에 대한 정보를 제공합니다.

15.15.1.1 INNODB_CMP 및 INNODB_CMP_RESET

INNODB_CMP 및 INNODB_CMP_RESET 테이블은 압축된 테이블과 관련된 작업에 대한 상태 정보를 제공하며, 이는 15.9절 "InnoDB 테이블 및 페이지 압축"에 설명되어 있습니다.
페이지 크기 열은 압축된 페이지 크기를 보고합니다.

이 두 테이블의 내용은 동일하지만 INNODB_CMP_RESET에서 읽으면 압축 및 압축 해제 작업에 대한 통계가 초기화됩니다. 예를 들어, 60분마다 INNODB_CMP_RESET의 출력을 아카이브하면 매 시간마다 통계를 볼 수 있습니다. 모니터링하는 경우
의 출력을 확인하면(INNODB_CMP_RESET을 읽지 않도록 주의하세요), InnoDB가 시작된 이후 누적 통계를 확인할 수 있습니다.

테이블 정의는 26.4.6절, "INFORMATION_SCHEMA INNODB_CMP 및 INNODB_CMP_RESET 테이블"을 참조하세요.

15.15.1.2 INNODB_CMPMEM 및 INNODB_CMPMEM_RESET

INNODB_CMPMEM 및 INNODB_CMPMEM_RESET 테이블은 버퍼 풀에 상주하는 압축 페이지에 대한 상태 정보를 제공합니다. 압축 테이블 및 버퍼 풀 사용에 대한 자세한 내용은 섹션 15.9, "InnoDB 테이블 및 페이지 압축"을 참조하세요. INNODB_CMP 및 INNODB_CMP_RESET 테이블은 압축에 대한 보다 유용한 통계를 제공합니다.

내부 세부 정보

InnoDB는 버디 얼로케이터 시스템을 사용하여 1KB에서 16KB까지 다양한 크기의 페이지에 할당된 메모리를 관리합니다. 여기에 설명된 두 테이블의 각 행은 단일 페이지 크기에 해당합니다.

INNODB_CMPMEM과 INNODB_CMPMEM_RESET 테이블의 내용은 동일하지만, INNODB_CMPMEM_RESET에서 읽으면 재배치 작업에 대한 통계가 재설정됩니다. 예를 들어 60분마다 INNODB_CMPMEM_RESET의 출력을 아카이브하면 시간별 통계가 표시됩니다. INNODB_CMPMEM_RESET을 읽지 않고대신 INNODB_CMPMEM의 출력을 모니터링했다면, InnoDB가 시작된 이후 누적 통계가 표시될 것입니다.

테이블 정의는 26.4.7절, "INFORMATION_SCHEMA INNODB_CMPMEM 및 INNODB_CMPMEM_RESET 테이블"을 참조하세요.

15.15.1.3 압축 정보 스키마 테이블 사용

예제 15.1 압축 정보 스키마 테이블 사용

다음은 압축된 테이블이 포함된 데이터베이스의 샘플 출력입니다(15.9절, "InnoDB 테이블 및 페이지 압축", `INNODB_CMP`, `INNODB_CMP_PER_INDEX` 및 `INNODB_CMPMEM` 참조).

다음 표는 가벼운 워크로드에서 `INFORMATION_SCHEMA.INNODB_CMP`의 내용을 보여줍니다. 버퍼 풀에 포함된 유일한 압축 페이지 크기는 8K입니다. 통계가 재설정된 시점 이후 페이지 압축 또는 압축 해제에 소요된 시간은 1초 미만이며, 이는 `COMPRESS_TIME` 및 `UNCOMPRESS_TIME` 열이 0이기 때문입니다.

페이지 크기	압축 작업	압축 작업 확인	압축 시간	압축 해제 작업	압축 해제 시간
1024	0	0	0	0	0
2048	0	0	0	0	0
4096	0	0	0	0	0
8192	1048	921	0	61	0
16384	0	0	0	0	0

`INNODB_CMPMEM`에 따르면 버퍼 풀에는 6169개의 압축된 8KB 페이지가 있습니다. 할당된 다른 유일한 블록 크기는 64바이트입니다. 버퍼 풀에 압축되지 않은 페이지가 존재하지 않는 압축된 페이지의 블록 설명자에는 `INNODB_CMPMEM`에서 가장 작은 `PAGE_SIZE`가 사용됩니다. 이러한 페이지가 5910개 있음을 알 수 있습니다. 간접적으로 259개(6169-5910)의 압축 페이지도 압축되지 않은 형태로 버퍼 풀에 존재한다는 것을 알 수 있습니다.

다음 표는 가벼운 워크로드에서 `INFORMATION_SCHEMA.INNODB_CMPMEM`의 내용을 보여줍니다
. 압축 페이지에 대한 메모리 할당자의 조각화로 인해 일부 메모리를 사용할 수 없습니다: `합계(페이지_크기*페이지_자유)=6784`. 이는 작은 메모리 할당 요청이 다음과 같은 16K 블록부터 시작하여
더 큰 블록을 분할하여 수행되기 때문입니다.
버디 할당 시스템을 사용하여 메인 버퍼 풀에서 할당된 블록입니다. 할당된 일부 블록이 인접한 더 큰 여유 블록을 형성하기 위해 재배치(복사)되었기 때문에 조각화가 이렇게 낮습니다.
이 `SUM(PAGE_SIZE*RELOCATION_OPS)` 바이트 복사에 소요된 시간은 1초 미만입니다.
`(합계(재배치_시간)=0)`.

페이지 크기	사용된 페이지	페이지 무료	재배치 작업	재배치 시간
64	5910	0	2436	0
128	0	1	0	0
256	0	0	0	0
512	0	1	0	0
1024	0	0	0	0
2048	0	1	0	0
4096	0	1	0	0
8192	6169	0	5	0
16384	0	0	0	0

15.15.2 InnoDB 정보_SCHEMA 트랜잭션 및 잠금 정보

하나의 정보 스키마 테이블과 두 개의 성능 스키마 테이블을 통해 다음을 모니터링할 수 있습니다.
InnoDB 트랜잭션 및 잠재적인 잠금 문제를 진단합니다:

- INNODB_TRX: 이 정보 스키마 테이블은 트랜잭션 상태(예: 실행 중인지 또는 잠금을 기다리는 중인지), 트랜잭션이 시작된 시점, 트랜잭션이 실행 중인 특정 SQL 문을 포함하여 현재 InnoDB 내에서 실행 중인 모든 트랜잭션에 대한 정보를 제공합니다.

- DATA_LOCKS: 이 성능 스키마 테이블에는 각 보류 잠금과 보류 잠금이 해제되기를 기다리며 차단된 각 잠금 요청에 대한 행이 포함되어 있습니다:

  - 잠금을 보유한 트랜잭션의 상태(INNODB_TRX.TRX_STATE가 실행 중, 잠금 대기, 롤백 또는 커밋)에 관계없이 각 잠금에 대해 하나의 행이 있습니다.

  - 다른 트랜잭션이 잠금을 해제하기를 기다리는 InnoDB의 각 트랜잭션은 정확히 하나의 차단 잠금 요청에 의해 차단됩니다(INNODB_TRX.TRX_상태는 LOCK WAIT). 이 차단 잠금 요청은 호환되지 않는 모드에서 다른 트랜잭션이 보유한 행 또는 테이블 잠금에 대한 것입니다. 잠금 요청은 항상 요청을 차단하는 보류된 잠금의 모드와 호환되지 않는 모드(읽기 대 쓰기, 공유 대 독점)를 갖습니다.

    차단된 트랜잭션은 다른 트랜잭션이 커밋하거나 롤백하여 요청된 잠금을 해제할 때까지 진행할 수 없습니다. 차단된 모든 트랜잭션에 대해 데이터_잠금에는 트랜잭션이 요청하고 대기 중인 각 잠금을 설명하는 행이 하나씩 포함됩니다.

- DATA_LOCK_WAITS: 이 성능 스키마 테이블은 어떤 트랜잭션이 특정 잠금을 기다리고 있는지, 또는 특정 트랜잭션이 어떤 잠금을 기다리고 있는지를 나타냅니다. 이 테이블에는 차단된 각 트랜잭션에 대해 하나 이상의 행이 포함되어 있으며, 해당 트랜잭션이 요청한 잠금과 해당 요청을 차단하고 있는 모든 잠금을 나타냅니다. REQUESTING_ENGINE_LOCK_ID 값은 트랜잭션이 요청한 잠금을 참조하고, BLOCKING_ENGINE_LOCK_ID 값은 (다른 트랜잭션이 보유한) 잠금을 참조합니다.
  를 설정하여 첫 번째 트랜잭션이 진행되지 않도록 합니다. 특정 차단된 트랜잭션의 모든 행은 data_lock_waits는 요청 엔진 잠금 ID는 동일하고 차단 엔진 잠금 ID는 다른 값을 갖습니다.

앞의 테이블에 대한 자세한 내용은 26.4.28절, "정보 스키마 INNODB_TRX 테이블", 27.12.13.1절, "데이터 잠금 테이블" 및 27.12.13.2절, "데이터 잠금 대기 테이블"을 참조하세요.

15.15.2.1 InnoDB 트랜잭션 및 잠금 정보 사용

이 섹션에서는 성능 스키마에 노출된 잠금 정보 사용에 대해 설명합니다.
데이터 잠금 및 데이터 잠금 대기 테이블.

차단 거래 식별

어떤 트랜잭션이 다른 트랜잭션을 차단하는지 식별하는 것이 도움이 될 때가 있습니다. InnoDB 트랜잭션 및 데이터 잠금에 대한 정보가 포함된 테이블을 사용하면 어떤 트랜잭션이 다른 트랜잭션을 대기 중인지, 어떤 리소스가 요청되고 있는지 확인할 수 있습니다. (이 테이블에 대한 설명은 섹션 15.15.2, "InnoDB 정보_SCHEMA 트랜잭션 및 잠금 정보"를 참조하세요.)

세 개의 세션이 동시에 실행되고 있다고 가정해 보겠습니다. 각 세션은 MySQL 스레드에 해당하며 하나의 트랜잭션을 차례로 실행합니다. 이러한 세션이 다음 문을 실행했지만 아직 트랜잭션을 커밋하지 않은 경우의 시스템 상태를 고려해 보겠습니다:

- 세션 A:

```
시작;
SELECT a FROM t FOR UPDATE;
SELECT SLEEP(100);
```

- 세션 B:

```
SELECT b FROM t FOR UPDATE;
```

- 세션 C:

```
SELECT c FROM t FOR UPDATE;
```

이 시나리오에서는 다음 쿼리를 사용하여 대기 중인 트랜잭션과 이를 차단하는 트랜잭션을 확인합니다:

```
SELECT
  r.trx_id waiting_trx_id,
  r.trx_mysql_thread_id waiting_thread,
  r.trx_query waiting_query,
  b.trx_id blocking_trx_id,
  b.trx_mysql_thread_id blocking_thread,
  b.trx_query blocking_query
FROMperformance_schema .data_lock_waits w INNER
JOIN information_schema.innodb_trx b
  ON b.trx_id = w.blocking_engine_transaction_id
INNER JOIN information_schema.innodb_trx r
  ON r.trx id = w.requesting engine transaction id;
```

또는 더 간단하게는 `sys` 스키마 `innodb_lock_waits` 뷰를 사용합니다:

```
SELECT
  대기_TRX_ID, 대기_PID, 대
  기_쿼리, 블로킹_TRX_ID, 블
  로킹_PID, 블로킹_쿼리
FROM sys.innodb lock waits;
```

차단 쿼리에 대해 NULL 값이 보고되는 경우 발급 세션이 유휴 상태가 된 후 차단 쿼리 식별을 참조하세요.

대기 중인 TRX ID	대기 스레드	대기 쿼리	TRX ID 차단	스레드 차단	쿼리 차단
A4	6	SELECT b FROM t FOR UPDATE	A3	5	SELECT SLEEP(100)
A5	7	SELECT c FROM t FOR UPDATE	A3	5	SELECT SLEEP(100)
A5	7	SELECT c FROM t FOR UPDATE	A4	6	SELECT b FROM t FOR UPDATE

앞의 표에서 '대기 중인 쿼리' 또는 '차단 중인 쿼리' 열을 통해 세션을 식별할 수 있습니다. 보시다시피

- 세션 B(trx id `A4`, 스레드 `6`)와 세션 C(trx id `A5`, 스레드 `7`)는 모두 세션 A(trx id `A3`, 스레드 `5`)를 기다리고 있습니다.
- 세션 C는 세션 A와 마찬가지로 세션 B를 기다리고 있습니다.

정보 스키마 `INNODB_TRX` 테이블과 성능 스키마 `data_locks` 및 `data_lock_waits` 테이블에서 기초 데이터를 확인할 수 있습니다.

다음 표는 `INNODB_TRX` 테이블의 일부 샘플 내용을 보여줍니다.

트렉스 ID	TRX 상태	TRX 시작	TRX 요청 잠금 ID	TRX 대기 시작	TRX 무게	TRX MYSQL 스레드 ID	trx 쿼리

A3	러닝	2008-01-1 16:44:54	5NULL	NULL	2	5	SELECT SLEEP(100

트렉스 ID	TRX 상태	TRX 시작	TRX 요청 잠금 ID	TRX 대기 시작	TRX 무게	TRX MYSQL 스레드 ID	trx 쿼리
A4	잠금 대기	2008-01-16:45:09	5A4:1:3:2	2008-01-16:45:09	52	6	SELECT b FROM t FOR UPDATE
A5	잠금 대기	2008-01-16:45:14	5A5:1:3:2	2008-01-16:45:14	52	7	SELECT c FROM t FOR UPDATE

다음 표는 데이터_잠금 테이블의 일부 샘플 내용을 보여줍니다.

잠금 ID	TRX ID 잠금	잠금 모드	잠금 유형	잠금 스키마	잠금 테이블	잠금 색인	데이터 잠금
A3:1:3:2	A3	X	기록	테스트	t	기본	0x0200
A4:1:3:2	A4	X	기록	테스트	t	기본	0x0200
A5:1:3:2	A5	X	기록	테스트	t	기본	0x0200

다음 표는 데이터_잠금_대기 테이블의 일부 샘플 내용을 보여줍니다.

TRX ID 요청	요청된 잠금 ID	TRX ID 차단	차단 잠금 ID
A4	A4:1:3:2	A3	A3:1:3:2
A5	A5:1:3:2	A3	A3:1:3:2
A5	A5:1:3:2	A4	A4:1:3:2

발급 세션이 유휴 상태가 된 후 차단 쿼리 식별하기

차단 트랜잭션을 식별할 때 쿼리를 실행한 세션이 유휴 상태인 경우 차단 쿼리에 대해 NULL 값이 보고됩니다. 이 경우 다음 단계에 따라 차단 쿼리를 확인합니다:

1. 차단 트랜잭션의 프로세스 목록 ID를 식별합니다. sys.innodb_lock_waits 테이블에서 차단 트랜잭션의 프로세스 목록 ID는 blocking_pid 값입니다.

2. blocking_pid를 사용하여 MySQL 성능 스키마 스레드 테이블을 쿼리하여 차단 트랜잭션의 THREAD_ID를 확인합니다. 예를 들어, blocking_pid가 6이면 이 쿼리를 실행합니다:

```
SELECT THREAD_ID FROM performance_schema.threads WHERE PROCESSLIST_ID = 6;
```

3. THREAD_ID를 사용하여 성능 스키마 events_statements_current 테이블을 쿼리하여 스레드에서 마지막으로 실행한 쿼리를 확인합니다. 예를 들어 THREAD_ID가 28이면 이 쿼리를 실행합니다:

```
SELECT THREAD_ID, SQL_TEXT FROM performance_schema.events_statements_current
WHERE THREAD_ID = 28\G
```

4. 스레드에서 실행한 마지막 쿼리로 잠금이 유지되는 이유를 파악하기에 정보가 충분하지 않은 경우 성능 스키마 `events_statements_history` 테이블을 쿼리하여 스레드에서 실행한 마지막 10개의 문을 볼 수 있습니다.

```
SELECT THREAD_ID, SQL_TEXT FROM performance_schema.events_statements_history
WHERE THREAD_ID = 28 ORDER BY EVENT_ID;
```

InnoDB 트랜잭션과 MySQL 세션의 상관 관계 파악

때때로 내부 `InnoDB` 잠금 정보를 MySQL에서 유지 관리하는 세션 수준 정보와 연관시키는 것이 유용할 수 있습니다. 예를 들어, 특정 `InnoDB` 트랜잭션 ID에 대해 잠금을 유지하여 다른 트랜잭션을 차단할 수 있는 해당 MySQL 세션 ID와 세션의 이름을 알고 싶을 수 있습니다.

다음은 `INFORMATION_SCHEMA INNODB_TRX` 테이블과 성능 스키마 `data_locks` 및 `data_lock_waits` 테이블의 출력으로, 어느 정도 로드된 시스템에서 가져온 것입니다. 보시다시피, 여러 트랜잭션이 실행되고 있습니다.

다음 데이터_잠금 및 데이터_잠금_대기 테이블에서 이를 확인할 수 있습니다:

- 트랜잭션 `77F`(`INSERT` 실행)는 트랜잭션 `77E`, `77D`, 77B가 커밋되기를 기다리고 있습니다.
- 트랜잭션 `77E`(`INSERT` 실행)는 트랜잭션 `77D` 및 77B가 커밋되기를 기다리고 있습니다.
- 트랜잭션 `77D`(`INSERT` 실행)가 트랜잭션 77B가 커밋되기를 기다리고 있습니다.
- 트랜잭션 `77B`(`INSERT` 실행)가 트랜잭션 77A가 커밋되기를 기다리고 있습니다.
- 트랜잭션 77A가 실행 중이며 현재 `SELECT`를 실행 중입니다.
- 트랜잭션 `E56`(`INSERT` 실행)이 트랜잭션 E55가 커밋되기를 기다리고 있습니다.
- 트랜잭션 `E55`(`INSERT` 실행)가 트랜잭션 19C가 커밋되기를 기다리고 있습니다.
- 트랜잭션 19C가 실행 중이며 현재 `INSERT`를 실행 중입니다.

참고

에 표시된 쿼리 간에 불일치가 있을 수 있습니다.
`INFORMATION_SCHEMA PROCESSLIST` 및 `INNODB_TRX` 테이블. 의 경우
설명은 섹션 15.15.2.3, "InnoDB 트랜잭션 및 잠금 정보의 지속성 및 일관성"을 참조하세요.

다음 표는 워크로드가 많은 시스템을 실행하는 시스템에 대한 프로세스 목록 테이블의 내용을 보여줍니다.

ID	USER	호스트	DB	COMMAND	시간	상태	정보
384	root	localhost	테스트	쿼리	10	업데이트	t2 값에 삽입 ...
257	root	localhost	테스트	쿼리	3	업데이트	t2 값에 삽입 ...
130	root	localhost	테스트	쿼리	0	업데이트	t2 값에 삽입 ...
61	root	localhost	테스트	쿼리	1	업데이트	t2 값에 삽입 ...
8	root	localhost	테스트	쿼리	1	업데이트	t2 값에 삽

T

ID	USER	호스트	DB	COMMAND	시간	상태	정보
2	root	localhost	테스트	수면	566		NULL

다음 표는 워크로드가 많은 시스템을 실행하는 시스템에 대한 `INNODB_TRX` 테이블의 내용을 보여줍니다.

트렉스 ID	TRX 상태	TRX 시작	TRX 요청 잠금 ID	TRX 대기 시작	TRX 무게	TRX MYSQL 스레드 ID	trx 쿼리
77F	잠금 대기	2008-01-1 13:10:16	577F	2008-01-1 13:10:16	51	876	INSERT INTO t09 (D, b, c) 값 ...
77E	잠금 대기	2008-01-1 13:10:16	577E	2008-01-1 13:10:16	51	875	INSERT INTO t09 (D, b, c) 값 ...
77D	잠금 대기	2008-01-1 13:10:16	577D	2008-01-1 13:10:16	51	874	INSERT INTO t09 (D, b, c) 값 ...
77B	잠금 대기	2008-01-1 13:10:16	577B:733:1	22:0108-01-1 13:10:16	54	873	INSERT INTO t09 (D, b, c) 값 ...
77A	러닝	2008-01-1 13:10:16	5NULL	NULL	4	872	SELECT b, c FROM t09 WHERE ...
E56	잠금 대기	2008-01-1 13:10:06	5E56:743:6	:22008-01-1 13:10:06	55	384	t2 값에 삽입 ...
E55	잠금 대기	2008-01-1 13:10:06	5E55:743:3	82:0208-01-1 13:10:13	5965	257	t2 값에 삽입 ...
19C	러닝	2008-01-1 13:09:10	5NULL	NULL	2900	130	t2 값에 삽입 ...
E15	러닝	2008-01-1 13:08:59	5NULL	NULL	5395	61	t2 값에 삽입 ...
51D	러닝	2008-01-1 13:08:47	5NULL	NULL	9807	8	t2 값에 삽입 ...

다음 표는 워크로드가 많은 시스템을 실행하는 시스템에 대한 `data_lock_waits` 테이블의 내용을 보여줍

니다.

TRX ID 요청	요청된 잠금 ID	TRX ID 차단	차단 잠금 ID
77F	77F:806	77E	77E:806

TRX ID 요청	요청된 잠금 ID	TRX ID 차단	차단 잠금 ID
77F	77F:806	77D	77D:806
77F	77F:806	77B	77B:806
77E	77E:806	77D	77D:806
77E	77E:806	77B	77B:806
77D	77D:806	77B	77B:806
77B	77B:733:12:1	77A	77A:733:12:1
E56	E56:743:6:2	E55	E55:743:6:2
E55	E55:743:38:2	19C	19C:743:38:2

다음 표는 워크로드가 많은 시스템을 실행하는 데이터 잠금 테이블의 내용을 보여줍니다.

잠금 ID	TRX ID 잠금	잠금 모드	잠금 유형	잠금 스키마	잠금 테이블	잠금 색인	데이터 잠금
77F:806	77F	AUTO_INC	테이블	테스트	t09	NULL	NULL
77E:806	77E	AUTO_INC	테이블	테스트	t09	NULL	NULL
77D:806	77D	AUTO_INC	테이블	테스트	t09	NULL	NULL
77B:806	77B	AUTO_INC	테이블	테스트	t09	NULL	NULL
77B:733:1	27:71B	X	기록	테스트	t09	기본	최고 의사 기록
77A:733:1	27:71A	X	기록	테스트	t09	기본	최고 의사 기록
E56:743:6	:E256	S	기록	테스트	t2	기본	0, 0
E55:743:6	:E255	X	기록	테스트	t2	기본	0, 0
E55:743:3	8E:525	S	기록	테스트	t2	기본	1922, 1922
19C:743:3	81:92C	X	기록	테스트	t2	기본	1922, 1922

15.15.2.2 InnoDB 잠금 및 잠금 대기 정보

트랜잭션이 테이블의 행을 업데이트하거나 `SELECT FOR UPDATE`로 잠그면 `InnoDB`는 해당 행에 대한 잠금 목록 또는 큐를 설정합니다. 마찬가지로 `InnoDB`는 테이블 수준의 잠금을 위해 테이블에 잠금 목록을 유지 관리합니다. 두 번째 트랜잭션이 행을 업데이트하거나 호환되지 않는 모드에서 이전 트랜잭션에 의해 이미 잠긴 테이블을 잠그고자 하는 경우, `InnoDB`는 해당 행에 대한 잠금 요청을 해당 큐에 추가합니다. 트랜잭션이 잠금을 획득하려면 해당 행 또는 테이블에 대해 이전에 잠금 대기열에 입력된 호환되지 않는 모든 잠금 요청을 제거해야 합니다(해당 잠금을 보유하거나 요청하는 트랜잭션이 커밋하거나 롤백할 때 발생).

트랜잭션에는 서로 다른 행이나 테이블에 대한 잠금 요청이 얼마든지 있을 수 있습니다. 언제든지 한 트랜잭션이 다른 트랜잭션이 보유한 잠금을 요청할 수 있으며, 이 경우 해당 트랜잭션은 다른 트랜잭션에 의해 차단됩니

다. 요청하는 트랜잭션은 차단 잠금을 보유한 트랜잭션이 커밋 또는 롤백할 때까지 기다려야 합니다. 트랜잭션이 잠금을 기다리지 않으면 실행 중인 상태입니다. 트랜잭션이 잠금을 기다리는 중이면 잠금 대기 상태입니다. (`INFORMATION_SCHEMA INNODB_TRX` 테이블은 트랜잭션 상태 값을 나타냅니다.)

성능 스키마 데이터_잠금 테이블은 각 잠금 대기 트랜잭션에 대해 하나 이상의 행을 보유하며, 진행을 방해하는 모든 잠금 요청을 나타냅니다. 또한 이 테이블에는 특정 행 또는 테이블에 대해 보류 중인 잠금 대기열의 각 잠금을 설명하는 행이 하나씩 포함되어 있습니다. 성능

스키마 data_lock_waits 테이블은 트랜잭션이 이미 보유하고 있는 잠금 중 다른 트랜잭션이 요청한 잠금을 차단하는 잠금을 보여줍니다.

15.15.2.3 InnoDB 트랜잭션 및 잠금 정보의 지속성 및 일관성

트랜잭션 및 잠금 테이블(정보 스키마 INNODB_TRX 테이블, 성능 스키마 data_locks 및 data_lock_waits 테이블)에 노출된 데이터는 빠르게 변화하는 데이터를 엿볼 수 있게 해줍니다. 이는 애플리케이션에서 시작한 업데이트가 발생할 때만 데이터가 변경되는 사용자 테이블과는 다릅니다. 기본 데이터는 내부 시스템에서 관리하는 데이터이며 매우 빠르게 변경될 수 있습니다:

- INNODB_TRX, data_locks 및 data_lock_waits 간에 데이터가 일관되지 않을 수 있습니다. 테이블.

  data_locks 및 data_lock_waits 테이블은 InnoDB 스토리지 엔진의 라이브 데이터를 노출하여 INNODB_TRX 테이블의 트랜잭션에 대한 잠금 정보를 제공합니다. 잠금 테이블에서 검색된 데이터는 SELECT가 실행될 때 존재하지만, 클라이언트가 쿼리 결과를 소비할 때쯤에는 사라지거나 변경될 수 있습니다.

  data_lock을 data_lock_waits와 조인하면 더 이상 존재하지 않거나 아직 존재하지 않는 data_lock의 상위 행을 식별하는 data_lock_waits의 행을 표시할 수 있습니다.

- 트랜잭션 및 잠금 테이블의 데이터가 트랜잭션 및 잠금 테이블의 데이터와 일치하지 않을 수 있습니다. INFORMATION_SCHEMA PROCESSLIST 테이블 또는 성능 스키마 스레드 테이블.

  예를 들어, InnoDB 트랜잭션 및 잠금 테이블의 데이터를 PROCESSLIST 테이블의 데이터와 비교할 때는 주의해야 합니다. 예를 들어 INNODB_TRX와 PROCESSLIST를 조인하는 SELECT를 한번만 실행하더라도 해당 테이블의 내용은 일반적으로 일관되지 않습니다. INNODB_TRX가 PROCESSLIST에 없는 행을 참조하거나 INNODB_TRX.TRX_QUERY에 표시된 트랜잭션의 현재 실행 중인 SQL 쿼리가 PROCESSLIST.INFO에 있는 것과 다를 수 있습니다.

15.15.3 InnoDB INFO_SCHEMA 스키마 개체 테이블

InnoDB에서 관리하는 스키마 개체에 대한 메타데이터는 InnoDB INFORMATION_SCHEMA 테이블을 사용하여 추출할 수 있습니다. 이 정보는 데이터 사전에서 가져옵니다. 일반적으로 이러한 유형의 정보는 섹션 15.17, "InnoDB 모니터"의 기술을 사용하여 InnoDB 모니터를 설정하고 SHOW ENGINE INNODB STATUS 문에서 출력을 구문 분석하여 얻을 수 있습니다. InnoDB INFORMATION_SCHEMA 테이블 인터페이스를 사용하면 SQL을 사용하여 이 데이터를 쿼리할 수 있습니다.

InnoDB INFO_SCHEMA 스키마 개체 테이블에는 아래 나열된 테이블이 포함됩니다.

```
innodb_datafiles
innodb_tablestats
innodb_foreign
innodb_columns
innodb_indexes
innodb_fields
innodb_tablespaces
innodb_tablespaces_brief
innodb_foreign_cols
innodb_tables
```

테이블 이름은 제공된 데이터 유형을 나타냅니다:

- `INNODB_TABLES`는 `InnoDB` 테이블에 대한 메타데이터를 제공합니다.
- `INNODB_COLUMNS`는 `InnoDB` 테이블 열에 대한 메타데이터를 제공합니다.
- `INNODB_INDEXES`는 `InnoDB` 인덱스에 대한 메타데이터를 제공합니다.
- `INNODB_FIELDS`는 `InnoDB` 인덱스의 주요 열(필드)에 대한 메타데이터를 제공합니다.
- `INNODB_TABLESTATS`는 인메모리 데이터 구조에서 파생된 `InnoDB` 테이블에 대한 저수준 상태 정보 보기를 제공합니다.

- INNODB_DATAFILES는 InnoDB 테이블별 파일 및 일반 테이블 공간에 대한 데이터 파일 경로 정보를 제공합니다.
- INNODB_TABLESPACES는 테이블별, 일반 및 실행 취소 테이블 공간에 대한 InnoDB 파일에 대한 메타데이터를 제공합니다.
- INNODB_TABLESPACES_BRIEF는 InnoDB 테이블 스페이스에 대한 메타데이터의 하위 집합을 제공합니다.
- INNODB_FOREIGN은 InnoDB 테이블에 정의된 외래 키에 대한 메타데이터를 제공합니다.
- INNODB_FOREIGN_COLS는 다음에 정의된 외래 키의 열에 대한 메타데이터를 제공합니다. InnoDB 테이블.

InnoDB INFO_SCHEMA 스키마 객체 테이블은 TABLE_ID, INDEX_ID, SPACE 등의 필드를 통해 조인할 수 있으므로, 연구 또는 모니터링하려는 객체에 대해 사용 가능한 모든 데이터를 쉽게 검색할 수 있습니다.

각 테이블의 열에 대한 자세한 내용은 InnoDB INFO_SCHEMA 설명서를 참조하세요.

예 15.2 InnoDB INFO_SCHEMA 스키마 개체 테이블

이 예제에서는 단일 인덱스(i1)가 있는 간단한 테이블(t1)을 사용하여 InnoDB INFO_SCHEMA 스키마 객체 테이블에 있는 메타데이터 유형을 설명합니다.

1. 테스트 데이터베이스 및 테이블 t1을 만듭니다:

```
mysql> 데이터베이스 생성 테스트;

mysql> USE 테스트;

mysql> CREATE TABLE t1 (
       col1 INT,
       col2 CHAR(10),
       col3 VARCHAR(10))
       ENGINE = InnoDB;

mysql> CREATE INDEX i1 ON t1(col1);
```

2. 테이블 t1을 생성한 후 INNODB_TABLES를 쿼리하여 test/t1의 메타데이터를 찾습니다:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_TABLES WHERE NAME='test/t1' \G
*************************** 1. 행 ***************************
     TABLE_ID: 71
         이름: test/t1 플
         래그: 1
       N_COLS: 6
        공간: 57
   ROW_FORMAT: Compact
ZIP_PAGE_SIZE: 0
 instant_cols: 0
```

테이블 t1의 TABLE_ID는 71입니다. FLAG 필드는 테이블 형식 및 스토리지 특성에 대한 비트 수준 정보를 제공합니다. 6개의 열이 있으며, 이 중 3개의 열은 InnoDB에서 생성한 숨겨진 열입니다(DB_ROW_ID, DB_TRX_ID 및 DB_ROLL_PTR). 테이블의 SPACE ID는 57입니다(값이 0이면 테이블이 시스템 테이블

스페이스에 있음을 나타냄). ROW_FORMAT은 Compact입니다. ZIP_PAGE_SIZE는 압축 행 형식의 테이블에만 적용됩니다. INSTANT_COLS는 ALTER TABLE ...을 사용하여 첫 번째 인스턴트 열을 추가하기 전의 테이블 내 열 수를 표시합니다.
알고리즘=인스턴트로 열을 추가합니다.

3. INNODB_TABLES의 TABLE_ID 정보를 사용하여 테이블의 열에 대한 정보를 INNODB_COLUMNS 테이블에 쿼리합니다.

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_COLUMNS where TABLE_ID = 71\G
*************************** 1. 행 ***************************
```

```
     TABLE_ID: 71
         NAME: col1
          POS: 0
        MTYPE: 6
       PRTYPE: 1027
          LEN: 4
  HAS_기본값: 0 기본값:
NULL
*************************** 2. 행 ***************************
     TABLE_ID: 71
         이름: col2 위치
          : 1
        MTYPE: 2
       PRTYPE: 524542
          LEN: 10
  HAS_기본값: 0 기본값:
NULL
*************************** 3. row ***************************
     TABLE_ID: 71
         이름: col3 위치
          : 2
        MTYPE: 1
       PRTYPE: 524303
          LEN: 10
  HAS_기본값: 0 기본값:
NULL
```

INNODB_COLUMNS는 TABLE_ID 및 칼럼 NAME 외에도 각 칼럼의 서수 위치(POS)(0에서 시작하여 순차적으로 증가), 칼럼 MTYPE 또는 "기본 유형"(6 = INT, 2 = CHAR, 1 = VARCHAR), PRTYPE 또는 "정확한 유형"(MySQL 데이터 유형, 문자 집합 코드 및 null 가능성을 나타내는 비트가 있는 이진 값) 및 칼럼 길이(LEN)를 제공합니다. HAS_DEFAULT 및 DEFAULT_VALUE 열은 ALTER TABLE ... 알고리즘=INSTANT로 칼럼 추가.

4. 다시 한 번 INNODB_TABLES의 TABLE_ID 정보를 사용하여 테이블 t1과 연결된 인덱스에 대한 정보를 INNODB_INDEXES에 쿼리합니다.

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_INDEXES WHERE TABLE_ID = 71 \G
*************************** 1. 행 ***************************
       INDEX_ID: 111
           NAME: GEN_CLUST_INDEX
       TABLE_ID: 71
           TYPE: 1
       N_FIELDS: 0
        PAGE_NO: 3
            공간: 57
merge_threshold: 50
*************************** 2. 행 ***************************
       INDEX_ID: 112
           NAME: i1
       TABLE_ID: 71
           TYPE: 0
       N_FIELDS: 1
        PAGE_NO: 4
            공간: 57
merge_threshold: 50
```

INNODB_INDEXES는 두 개의 인덱스에 대한 데이터를 반환합니다. 첫 번째 인덱스는 테이블에 사용자 정의 클러스터 인덱스가 없는 경우 InnoDB에서 생성하는 클러스터 인덱스인 GEN_CLUST_INDEX입니다. 두 번째 인덱스(i1)는 사용자 정의 보조 인덱스입니다.

INDEX_ID는 인스턴스의 모든 데이터베이스에서 고유한 인덱스의 식별자입니다. TABLE_ID는 인덱스가 연결된 테이블을 식별합니다. 인덱스 TYPE 값은 인덱스의 유형을 나타냅니다(1 = 클러스터된 인덱스, 0 = 보조 인덱스). N_FILEDS 값은 인덱스를 구성하는 필드 수입니다. PAGE_NO는 인덱스 B-트리의 루트 페이지 번호이며, SPACE는 인덱스가 있는 테이블 스페이스의 ID입니다. 0이 아닌 값은 인덱스가 시스템 테이블 스페이스에 상주하지 않음을 나타냅니다. MERGE_THRESHOLD는 인덱스 페이지의 데이터 양에 대한 백분율 임계값을 정의합니다. 인덱스 페이지의 데이터 양이 이 값 아래로 떨어지면

(기본값은 50%) 행이 삭제되거나 업데이트 작업으로 행이 단축되는 경우입니다,
InnoDB는 인덱스 페이지를 인접한 인덱스 페이지와 병합하려고 시도합니다.

5. INNODB_INDEXES의 INDEX_ID 정보를 사용하여 인덱스 i1의 필드에 대한 정보를 INNODB_FIELDS에 쿼리합니다.

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_FIELDS where INDEX_ID = 112 \G
*************************** 1. 행 ***************************
INDEX_ID: 112
    NAME: col1
     POS: 0
```

INNODB_FIELDS는 인덱싱된 필드의 이름과 인덱스 내에서 해당 필드의 서수 위치를 제공합니다. 인덱스(i1)가 여러 필드에 정의된 경우, INNODB_FIELDS는 인덱싱된 각 필드에 대한 메타데이터를 제공합니다.

6. INNODB_TABLES의 SPACE 정보를 사용하여 테이블의 테이블 스페이스에 대한 정보를 INNODB_TABLESPACES 테이블에 쿼리합니다.

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_TABLESPACES WHERE SPACE = 57 \G
*************************** 1. 행 ***************************
          SPACE: 57
          이름: test/t1
          플래그: 16417
    ROW_FORMAT: 동적 PAGE_SIZE:
     16384
 ZIP_PAGE_SIZE: 0
    SPACE_TYPE: 단일
 FS_BLOCK_SIZE: 4096
     file_size: 114688
할당된 크기: 98304
자동 확장 크기: 0
서버 버전: 8.1.0
 공간_버전: 1 암호화
    : N
         상태: 정상
```

테이블스페이스의 SPACE ID와 연결된 테이블의 NAME 외에도, INNODB_TABLESPACES는 테이블스페이스 형식 및 스토리지 특성에 대한 비트 레벨 정보인 테이블스페이스 FLAG 데이터를 제공합니다. 또한 테이블스페이스 ROW_FORMAT, PAGE_SIZE 및 기타 여러 테이블스페이스 메타데이터 항목도 제공됩니다.

7. 다시 한 번 INNODB_TABLES의 SPACE 정보를 사용하여 테이블 스페이스 데이터 파일의 위치를 INNODB_DATAFILES에쿼리합니다.

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_DATAFILES WHERE SPACE = 57 \G
*************************** 1. 행 ***************************
SPACE: 57
 경로: ./test/t1.ibd
```

데이터 파일은 MySQL의 데이터 디렉터리 아래 테스트 디렉터리에 있습니다. 테이블별 파일 테이블 스페이스가 CREATE TABLE 문의 DATA DIRECTORY 절을 사용하여 MySQL 데이터 디렉터

리 외부 위치에 생성된 경우 테이블 스페이스 PATH는 정규화된 디렉터리 경로가 됩니다.

8. 마지막 단계로 t1 테이블에 행을 삽입하고(TABLE_ID = 71) INNODB_TABLESTATS 테이블의 데이터를 확인합니다. 이 테이블의 데이터는 MySQL 옵티마이저가 InnoDB 테이블을 쿼리할 때 사용할 인덱스를 계산하는 데 사용됩니다. 이 정보는 인메모리 데이터 구조에서 파생됩니다.

```
mysql> INSERT INTO t1 VALUES(5, 'abc', 'def');
쿼리 확인, 영향을 받은 행 1개(0.06초)

mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_TABLESTATS where TABLE_ID = 71 \G
*************************** 1. 행 ***************************
         TABLE_ID: 71
             이름: test/t1
```

```
통계_초기화: 초기화됨 NUM_ROWS: 1
 클러스터_인덱스_크기: 1
 OTHER_INDEX_SIZE: 0
       수정된_카운터: 1
          AUTOINC: 0
        REF_COUNT: 1
```

STATS_INITIALIZED 필드는 테이블에 대한 통계가 수집되었는지 여부를 나타냅니다. NUM_ROWS는 테이블의 현재 예상 행 수입니다. CLUST_INDEX_SIZE 및 OTHER_INDEX_SIZE 필드는 각각 테이블의 클러스터된 인덱스와 보조 인덱스를 저장하는 디스크의 페이지 수를 보고합니다. MODIFIED_COUNTER 값은 DML 연산 및 외래 키의 캐스케이드 연산에 의해 수정된 행 수를 표시합니다. AUTOINC 값은 자동 증가 기반 연산에 대해 발행될 다음 번호입니다. 테이블 t1에 정의된 자동 증가 열이 없으므로 값은 0입니다. REF_COUNT 값은 카운터입니다. 카운터가 0에 도달하면 테이블 메타데이터가 테이블 캐시에서 제거될 수 있음을 의미합니다.

예 15.3 외래 키 정보_SCHEMA 스키마 개체 테이블

INNODB_FOREIGN 및 INNODB_FOREIGN_COLS 테이블은 외래 키 관계에 대한 데이터를 제공합니다. 이 예제에서는 외래 키 관계가 있는 상위 테이블과 하위 테이블을 사용하여 INNODB_FOREIGN 및 INNODB_FOREIGN_COLS 테이블에 있는 데이터를 보여 줍니다.

1. 상위 및 하위 테이블로 테스트 데이터베이스를 만듭니다:

```
mysql> 데이터베이스 생성 테스트;

mysql> USE 테스트;

mysql> CREATE TABLE parent (id INT NOT NULL,
       PRIMARY KEY (id)) ENGINE=INNODB;

mysql> CREATE TABLE child (id INT, parent_id INT,
       INDEX par_ind (parent_id),
       CONSTRAINT fk1
       FOREIGN KEY (parent_id) REFERENCES parent(id)
       ON DELETE CASCADE) ENGINE=INNODB;
```

2. 부모 테이블과 자식 테이블이 생성된 후 INNODB_FOREIGN을 쿼리하여 테스트/자식 및 테스트/부모 외래 키 관계에 대한 외래 키 데이터를 찾습니다:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_FOREIGN \G
*************************** 1. 행 ***************************
      ID: test/fk1
FOR_NAME: 테스트/자식
REF_NAME: 테스트/부모
  N_COLS: 1
    TYPE: 1
```

메타데이터에는 자식 테이블에 정의된 CONSTRAINT의 이름이 지정된 외래 키 ID(fk1)가 포함됩니다. FOR_NAME은 외래 키가 정의된 하위 테이블의 이름입니다. REF_NAME은 상위 테이블("참조" 테이블)의 이름입니다. N_COLS는
외래 키 인덱스의 열 수입니다. TYPE은 외래 키 열에 대한 추가 정보를 제공하는 비트 플래그를 나타내는 숫자 값입니다. 이 경우 TYPE 값은 1이며, 이는 외래 키에 대해 ON DELETE CASCADE 옵션이 지정되

었음을 나타냅니다. `TYPE` 값에 대한 자세한 내용은 `INNODB_FOREIGN` 테이블 정의를 참조하십시오.

3. 외래 키 ID를 사용하여 INNODB_FOREIGN_COLS를 쿼리하여 외래 키의 열에 대한 데이터를 확인합니다.

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_FOREIGN_COLS WHERE ID = 'test/fk1' \G
*************************** 1. 행 ***************************
          ID: test/fk1
FOR_COL_NAME: parent_id
REF_COL_NAME: id
         POS: 0
```

FOR_COL_NAME은 하위 테이블의 외래 키 열 이름이고 REF_COL_NAME은 상위 테이블에서 참조된 열 이름입니다. `POS` 값은 외래 키 인덱스 내에서 키 필드의 서수 위치(0부터 시작)입니다.

예 15.4 InnoDB INFO_SCHEMA 스키마 객체 테이블 조인

이 예제에서는 3개의 `InnoDB INFO_SCHEMA` 스키마 객체 테이블(`INNODB_TABLES`, `INNODB_TABLESPACES` 및 `INNODB_TABLESTATS`)을 조인하여 직원 샘플 데이터베이스의 테이블에 대한 파일 형식, 행 형식, 페이지 크기 및 인덱스 크기 정보를 수집하는 방법을 설명합니다.

다음 테이블 별칭은 쿼리 문자열을 단축하는 데 사용됩니다:

- `INFORMATION_SCHEMA.INNODB_TABLES`: a
- `INFORMATION_SCHEMA.INNODB_TABLESPACES`: b
- `INFORMATION_SCHEMA.INNODB_TABLESTATS`: c

`IF()` 제어 흐름 함수는 압축된 테이블을 설명하는 데 사용됩니다. 테이블이 압축된 경우 인덱스 크기는 `PAGE_SIZE`가 아닌 `ZIP_PAGE_SIZE`를 사용하여 계산됩니다. 바이트 단위로 보고되는 `CLUST_INDEX_SIZE` 및 OTHER_INDEX_SIZE는 `1024*1024`로 나누어 메가바이트(MB) 단위의 인덱스 크기를 제공합니다. MB 값은 `ROUND()` 함수를 사용하여 소수점 이하 자릿수로 반올림됩니다.

```
mysql> SELECT a.NAME, a.ROW_FORMAT,
        page_size := IF(a.ROW_FORMAT='압축
         ',
          b.ZIP_PAGE_SIZE, b.PAGE_SIZE)
          AS page_size,
         ROUND((@page_size * c.CLUST_INDEX_SIZE)
          /(1024*1024)) AS pk_mb,
         ROUND((@page_size * c.OTHER_INDEX_SIZE)
          /(1024*1024)) AS secidx_mb
       FROM INFORMATION_SCHEMA.INNODB_TABLES a
       INNER JOIN INFORMATION_SCHEMA.INNODB_TABLESPACES b on a.NAME = b.NAME
       INNER JOIN INFORMATION_SCHEMA.INNODB_TABLESTATS c on b.NAME = c.NAME
       WHERE a.NAME LIKE 'employees/%'
       ORDER BY a.NAME DESC;
+----------------------+------------+-----------+-------+-----------------+
| NAME|                  ROW_FORMAT | page_size | pk_mb | secidx_mb |
+----------------------+------------+-----------+-------+-----------------+


| 직원/직함                  | 동적       |     16384 |    20 |         11 |
| 직원/급여                  | 동적       |     16384 |    93 |         34 |
| 직원/직원                  | 동적       |     16384 |    15 |          0 |
| 직원/부서_관리자           | 동적       |     16384 |     0 |          0 |
| employees/dept_emp         | 동적       |     16384 |    12 |         10 |
| 직원/부서 | 동적                                |16384 |0 |0 |
+----------------------+------------+-----------+-------+-----------------+
```

15.15.4 InnoDB 정보_화학 풀텍스트 인덱스 테이블

다음 표는 전체 텍스트 인덱스에 대한 메타데이터를 제공합니다:

```
mysql> SHOW TABLES FROM INFORMATION_SCHEMA LIKE 'INNODB_FT%';
+-----------------------------------------+
| Tables_in_INFORMATION_SCHEMA (INNODB_FT%) |
+-----------------------------------------+
| innodb_ft_config                        |
| innodb_ft_being_deleted                 |
| innodb_ft_deleted                       |
| innodb_ft_default_stopword              |
| innodb_ft_index_table                   |
| innodb_ft_index_cache                   |
+-----------------------------------------+
```

테이블 개요

- INNODB_FT_CONFIG: InnoDB 테이블에 대한 전체 텍스트 인덱스 및 관련 처리에 대한 메타데이터를 제공합니다.

- INNODB_FT_BEING_DELETED: INNODB_FT_DELETED 테이블의 스냅샷을 제공하며, OPTIMIZE TABLE 유지보수 작업 중에만 사용됩니다. OPTIMIZE TABLE을 수행하면 INNODB_FT_BEING_DELETED 테이블이 비워지고, INNODB_FT_DELETED 테이블에서 DOC_ID 값이 제거됩니다. INNODB_FT_BEING_DELETED의 내용은 일반적으로 다음과 같습니다.
  인덱스의 수명이 짧기 때문에 이 테이블은 모니터링 또는 디버깅에 유용성이 제한됩니다. 전체 텍스트 인덱스가 있는 테이블에서 최적화 테이블을 실행하는 방법에 대한 자세한 내용은 12.9.6절. "MySQL 전체 텍스트 검색 미세 조정"을 참조하세요.

- innodb_ft_deleted: InnoDB 테이블의 전체 텍스트 인덱스에서 삭제된 행을 저장합니다. InnoDB FULLTEXT 인덱스에 대한 DML 작업 중 비용이 많이 드는 인덱스 재구성을 방지하기 위해 새로 삭제된 단어에 대한 정보를 별도로 저장하고, 텍스트 검색 시 검색 결과에서 필터링하며, InnoDB 테이블에 대해 OPTIMIZE TABLE 문을 실행할 때만 기본 검색 인덱스에서 제거합니다.

- INNODB_FT_DEFAULT_STOPWORD: InnoDB 테이블에 전체 텍스트 인덱스를 생성할 때 기본적으로 사용되는 중지어 목록을 보유합니다.

  INNODB_FT_DEFAULT_STOPWORD 테이블에 대한 자세한 내용은 12.9.4절. "전체 텍스트 중지 단어"를 참조하세요.

- INNODB_FT_INDEX_TABLE: InnoDB 테이블의 전체 텍스트 인덱스에 대한 텍스트 검색을 처리하는 데 사용되는 반전된 인덱스에 대한 정보를 제공합니다.

- INNODB_FT_INDEX_CACHE: 전체 텍스트 인덱스에 새로 삽입된 행에 대한 토큰 정보를 제공합니다. DML 작업 중 비용이 많이 드는 인덱스 재구성을 피하기 위해 새로 인덱싱된 단어에 대한 정보는 별도로 저장되며, OPTIMIZE TABLE이 실행되거나 서버가 종료될 때, 또는 캐시 크기가 innodb_ft_cache_size 또는 innodb_ft_total_cache_size 시스템 변수에 정의된 한도를 초과할 때만 기본 검색 인덱스와 결합됩니다.

참고

INNODB_FT_DEFAULT_STOPWORD 테이블을 제외하고 이러한 테이블은 처음에 비어 있습니다. 이들 테이블을 쿼리하기 전에 innodb_ft_aux_table 시스템 변수의 값을 전체 텍스트 인덱스가 포함된 테이블의 이름(데이터베이스 이름 포함)으로 설정합니다(예: test/기사).

예 15.5 InnoDB 전체 텍스트 인덱스 정보_화학 테이블 예제

이 예제에서는 전체 텍스트 인덱스가 있는 테이블을 사용하여 전체 텍스트에 포함된 데이터를 보여줍니다. 인덱스 정보_SCHEMA 테이블을 생성합니다.

1. 전체 텍스트 인덱스가 있는 테이블을 만들고 일부 데이터를 삽입합니다:

```
mysql> CREATE TABLE articles (
         id INT UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY KEY,
         title VARCHAR(200),
         본문 텍스트,
         전체 텍스트(제목, 본문)
       ) 엔진=InnoDB;

mysql> INSERT INTO articles (title,body) VALUES
       ('MySQL 튜토리얼','DBMS는 데이터베이스 ...'),
       ('How To Use MySQL Well','당신이 겪은 후 ...'), ('Optimizing MySQL','이
       튜토리얼에서는 ...'),
       ('1001 MySQL 트릭','1. mysqld를 루트로 실행하지 마십시오. 2. ...'),
       ('MySQL vs. YourSQL','다음 데이터베이스 비교에서 ...................'),
       ('MySQL Security','제대로 구성된 경우, MySQL .............');
```

2. innodb_ft_aux_table 변수를 전체 텍스트 인덱스가 있는 테이블 이름으로 설정합니다. 이 변수가 설정되어 있지 않으면 INNODB_FT_DEFAULT_STOPWORD를 제외한 InnoDB FULLTEXT 정보 스키마 테이블이 비어 있습니다.

```
mysql> SET GLOBAL innodb_ft_aux_table = 'test/articles';
```

3. 전체 텍스트 인덱스에 새로 삽입된 행에 대한 정보를 표시하는 INNODB_FT_INDEX_CACHE 테이블을 쿼리합니다. DML 작업 중 비용이 많이 드는 인덱스 재구성을 피하기 위해 새로 삽입된 행에 대한 데이터는 OPTIMIZE TABLE이 실행될 때까지(또는 서버가 종료되거나 캐시 제한을 초과할 때까지) FULLTEXT 인덱스 캐시에 남아 있습니다.

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_FT_INDEX_CACHE LIMIT 5;
+------------+--------------+-------------+-----------+---------+ -----------------+
| 단어 |       첫_문서_ID | 마지막_문서_ID | 문서_수 | 문서_ID | 위치 |
+------------+--------------+-------------+-----------+---------+ -----------------+
| 1001       |            5 |           5 |         1 |       5 |         0 |
| after      |            3 |           3 |         1 |       3 |        22 |
| 비교       |            6 |           6 |         1 |       6 |        44 |
| 구성됨     |            7 |           7 |         1 |       7 |        20 |
| 데이터베이스 |2 |6 |2 |2 |  31 |
+------------+--------------+-------------+-----------+---------+ -----------------+
```

4. innodb_optimize_fulltext_only 시스템 변수를 활성화하고 FULLTEXT 인덱스가 포함된 테이블에서 OPTIMIZE TABLE을 실행합니다. 이 작업은 FULLTEXT 인덱스 캐시의 내용을 기본 FULLTEXT 인덱스로 플러시합니다. innodb_optimize_fulltext_only는 InnoDB 테이블에서 OPTIMIZE TABLE 문이 작동하는 방식을 변경하며, FULLTEXT 인덱스가 있는 InnoDB 테이블의 유지보수 작업 중에 일시적으로 사용하도록 되어 있습니다.

```
mysql> SET GLOBAL innodb_optimize_fulltext_only=ON;

mysql> OPTIMIZE TABLE articles;
+---------------+----------+----------+---------------+
| 표 | 연산       | 메시지 유형 | 메시지 텍스트 |
+---------------+----------+----------+---------------+
| test.articles | 최적화 | 상태| 확인     |
+---------------+----------+----------+---------------+
```

5. INNODB_FT_INDEX_TABLE 테이블을 쿼리하여 기본 전체 텍스트의 데이터에 대한 정보를 확인합니다. 인덱스에 전체 텍스트 인덱스 캐시에서 방금 플러시된 데이터에 대한 정보를 포함합니다.

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_FT_INDEX_TABLE LIMIT 5;
+------------+--------------+-------------+-----------+---------+ -----------------+
| 단어 |       첫_문서_ID | 마지막_문서_ID | 문서_수 | 문서_ID | 위치 |
+------------+--------------+-------------+-----------+---------+ -----------------+
| 1001       |            5 |           5 |         1 |       5 |         0 |
| after      |            3 |           3 |         1 |       3 |        22 |
| 비교       |            6 |           6 |         1 |       6 |        44 |
| 구성됨     |            7 |           7 |         1 |       7 |        20 |
| 데이터베이스 |2 |6 |2 |2 |  31 |
+------------+--------------+-------------+-----------+---------+ -----------------+
```

최적화 테이블 작업이 FULLTEXT 인덱스 캐시를 플러시했기 때문에 이제 INNODB_FT_INDEX_CACHE 테이블이 비어 있습니다.

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_FT_INDEX_CACHE LIMIT 5;
빈 세트 (0.00초)
```

6. 시험/아카이브 테이블에서 일부 레코드를 삭제합니다.

```
mysql> DELETE FROM test.articles WHERE id < 4;
```

7. INNODB_FT_DELETED 테이블을 쿼리합니다. 이 테이블은 전체 텍스트 인덱스에서 삭제된 행을 기록합니다. DML 작업 중 비용이 많이 드는 인덱스 재구성을 피하기 위해 새로 삭제된 레코드에 대한 정보는 별도로 저장되어 텍스트 검색을 수행할 때 검색 결과에서 필터링되고 OPTIMIZE TABLE을 실행할 때 기본 검색 인덱스에서 제거됩니다.

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_FT_DELETED;
+--------+
```

```
| DOC_ID |
+---------+
|2 |
|3 |
|4 |
+---------+
```

8. 테이블 최적화를 실행하여 삭제된 레코드를 제거합니다.

```
mysql> OPTIMIZE TABLE articles;
+---------------+----------+----------+---------------+
| 표 | 연산 | 메시지 유형 | 메시지 텍스트 |
+---------------+----------+----------+---------------+
| test.articles | 최적화 | 상태| 확인 |
+---------------+----------+----------+---------------+
```

이제 INNODB_FT_DELETED 테이블이 비어 있어야 합니다.

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_FT_DELETED;
빈 세트 (0.00초)
```

9. INNODB_FT_CONFIG 테이블을 쿼리합니다. 이 테이블에는 전체 텍스트 인덱스 및 관련 처리에 대한 메타데이터가 포함되어 있습니다:

   - optimize_checkpoint_limit: 최적화 테이블 실행이 중지되는 시간(초)입니다.
   - SYNCED_DOC_ID: 발급할 다음 DOC_ID입니다.
   - stopword_table_name: 사용자 정의 스톱워드 테이블의 *데이터베이스/테이블* 이름입니다. 사용자 정의 사용자 정의 중지 단어 테이블이 없는 경우 VALUE 열은 비어 있습니다.
   - 사용_스톱워드: 스톱워드 테이블을 사용할지 여부를 나타냅니다. 전체 텍스트 인덱스가 생성됩니다.

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_FT_CONFIG;
+---------------------------+----------+
| 키 | 값 |
+---------------------------+----------+
| 최적화_체크포인트_제한 | 180 |
| 동기화_문서_ID| 8 |
| stopword_table_name | |
| 사용_스톱워드| 1 |
+---------------------------+----------+
```

10. 이 기능은 일시적으로만 사용하도록 설정되므로 innodb_optimize_fulltext_only를 비활성화합니다:

```
mysql> SET GLOBAL innodb_optimize_fulltext_only=OFF;
```

15.15.5 InnoDB 정보_화학 버퍼 풀 테이블

InnoDB INFO_SCHEMA 버퍼 풀 테이블은 버퍼 풀 상태 정보와 InnoDB 버퍼 풀 내의 페이지에 대한 메타데이터를 제공합니다.

InnoDB INFO_SCHEMA 버퍼 풀 테이블에는 아래 나열된 테이블이 포함됩니다:

```
mysql> SHOW TABLES FROM INFORMATION_SCHEMA LIKE 'INNODB_BUFFER%';
+-----------------------------------------------+
| Tables_in_INFORMATION_SCHEMA (INNODB_BUFFER%) |
+-----------------------------------------------+
| innodb_buffer_page_lru                        |
| innodb_buffer_page                            |
| innodb_buffer_pool_stats                      |
+-----------------------------------------------+
```

테이블 개요

- `innodb_buffer_page`: `InnoDB` 버퍼 풀의 각 페이지에 대한 정보를 보유합니다.

- INNODB_BUFFER_PAGE_LRU: InnoDB 버퍼 풀의 페이지에 대한 정보, 특히 어떤 페이지를 퇴거할지 결정하는 LRU 목록에서 페이지가 어떻게 정렬되는지에 대한 정보를 보유합니다.
  버퍼 풀이 가득 차게 됩니다. INNODB_BUFFER_PAGE_LRU 테이블은 INNODB_BUFFER_PAGE 테이블과 동일한 열을 갖지만, BLOCK_ID 열 대신 LRU_POSITION 열이 있다는 점만 다릅니다.

- innodb_buffer_pool_stats: 버퍼 풀 상태 정보를 제공합니다. 대부분의 동일한 정보는 SHOW ENGINE INNODB STATUS 출력에서 제공되거나 InnoDB 버퍼 풀 서버 상태 변수를 사용하여 얻을 수 있습니다.

경고

INNODB_BUFFER_PAGE 또는 INNODB_BUFFER_PAGE_LRU 테이블을 쿼리하면 성능에 영향을 미칠 수 있습니다. 성능에 미치는 영향을 알고 있고 허용 가능한 수준이라고 판단하지 않는 한 프로덕션 시스템에서 이러한 테이블을 쿼리하지 마세요. 프로덕션 시스템의 성능에 영향을 미치지 않으려면 조사하려는 문제를 재현하고 테스트 인스턴스에서 버퍼 풀 통계를 쿼리하세요.

예 15.6 INNODB_BUFFER_PAGE 테이블에서 시스템 데이터 쿼리하기

이 쿼리는 테이블 이름에 슬래시 또는 마침표가 포함되어 있거나 테이블 이름에 사용자 정의 테이블을 나타내는 TABLE_NAME 값이 NULL인 페이지를 제외하여 시스템 데이터가 포함된 페이지의 대략적인 수를 제공합니다.

```
mysql> SELECT COUNT(*) FROM INFORMATION_SCHEMA.INNODB_BUFFER_PAGE
       여기서 table_name은 null이거나 (instr(table_name, '/') = 0 및 instr(table_name, '.') = 0);
+-----------+
| COUNT(*)  |
+-----------+
      |1516 |
+-----------+
```

이 쿼리는 시스템 데이터가 포함된 대략적인 페이지 수, 총 버퍼 풀 페이지 수, 시스템 데이터가 포함된 페이지의 대략적인 비율을 반환합니다.

```
mysql> SELECT
       (정보_schema.innodb_버퍼_페이지에서 count(*)를 선택합니다.
       여기서 table_name이 null이거나 (instr(table_name, '/') = 0 및 instr(table_name, '.') = 0)
       ) AS system_pages,
       (
       셀렉트 카운트(*)
       에서 정보_schema.innodb_버퍼_페이지
       ) AS total_pages,
       (
       SELECT ROUND((시스템_페이지/총페이지) * 100)
       ) AS 시스템_페이지_퍼센티지;
+-------------+------------+--------------------------+
| 시스템 페이지 | 총 페이지 | 시스템 페이지 비율 |
+-------------+------------+--------------------------+
           |295          |8192 |4 |
+-------------+------------+--------------------------+
```

버퍼 풀에 있는 시스템 데이터의 유형은 `PAGE_TYPE` 값을 쿼리하여 확인할 수 있습니다. 예를 들어, 다음 쿼리는 시스템 데이터가 포함된 페이지 중 8개의 고유한 `PAGE_TYPE` 값을 반환합니다:

```
mysql> SELECT DISTINCT PAGE_TYPE FROM INFORMATION_SCHEMA.INNODB_BUFFER_PAGE
       여기서 table_name은 null이거나 (instr(table_name, '/') = 0 및 instr(table_name, '.') = 0);
+--------------------+
| 페이지 유형          |
+--------------------+
| 시스템               |
| IBUF_BITMAP        |
| 알 수 없음           |
| 파일_공간_헤더 |
```

```
| INODE              |
| UNDO_LOG           |
| 할당됨             |
+--------------------+
```

예 15.7 INNODB_BUFFER_PAGE 테이블에서 사용자 데이터 쿼리하기

이 쿼리는 사용자 데이터가 포함된 페이지의 대략적인 개수를 계산하여 `TABLE_NAME` 값이 `NULL`이 아니며 `'%INNODB_TABLES%'`와 같지 않습니다.

```
mysql> SELECT COUNT(*) FROM INFORMATION_SCHEMA.INNODB_BUFFER_PAGE
       여기서 table_name이 null이 아니고 table_name이 '%innodb_tables%'와 같지 않습니다;
+----------+
| COUNT(*) |
+----------+
      |7897 |
+----------+
```

이 쿼리는 사용자 데이터가 포함된 대략적인 페이지 수, 버퍼 풀 페이지의 총 수, 사용자 데이터가 포함된 페이지의 대략적인 비율을 반환합니다.

```
mysql> SELECT
       (정보_schema.innodb_버퍼_페이지에서 count(*)를 선택합니다.
       여기서 table_name이 null이 아니고 (instr(table_name, '/') > 0 또는 instr(table_name, '.') >입
       니다.
       ) AS user_pages,
       (
       셀렉트 카운트(*)
       FROM information_schema.INNODB_BUFFER_PAGE
       ) AS total_pages,
       (
       SELECT ROUND((user_pages/total_pages) * 100)
       ) AS user_page_percentage;
+------------+-------------+ ------------------------+
| 사용자 페이지 | 총 페이지 | 사용자 페이지 퍼센트 |
+------------+-------------+ ------------------------+
        |7897         |8192                    |96 |
+------------+-------------+ ------------------------+
```

```
mysql> SELECT DISTINCT TABLE_NAME FROM INFORMATION_SCHEMA.INNODB_BUFFER_PAGE
       여기서 table_name이 null이 아니고 (instr(table_name, '/') > 0 또는 instr(table_name, '.') >입
       니다.
       그리고 테이블 이름이 '`mysql`.`innodb_%'와 같지 않아야 합니다;
+--------------------------+
| TABLE_NAME               |
+--------------------------+
| ``직원``.`급여``
| `직원`.`직원` |
+--------------------------+
```

인덱스 페이지에 대한 정보를 보려면 인덱스 이름을 사용하여 `INDEX_NAME` 열을 쿼리합니다. 예를 들어, 다음 쿼리는 `employees.salaries` 테이블에 정의된 `emp_no` 인덱스에 대한 페이지 수와 페이지의 총 데이터 크기를 반환합니다:

```
mysql> SELECT INDEX_NAME, COUNT(*) AS Pages,
ROUND(SUM(SUM(IF(COMPRESSED_SIZE = 0, @@GLOBAL.innodb_page_size, COMPRESSED_SIZE))/1024/1024)
AS '총 데이터(MB)'
에서 정보_schema.innodb_버퍼_페이지
WHERE INDEX_NAME='emp_no' AND TABLE_NAME = '`employees`.`salaries`';
+------------+-------+-------------------+
| 인덱스 이름 | 페이지 | 총 데이터(MB) |
+------------+-------+-------------------+
|            emp_no| 1609           |25 |
+------------+-------+-------------------+
```

이 쿼리는 인덱스에 정의된 모든 인덱스에 대한 페이지 수와 페이지의 총 데이터 크기를 반환합니다.

`employees.salaries` 테이블:

```
mysql> SELECT INDEX_NAME, COUNT(*) AS Pages,
       ROUND(SUM(SUM(IF(COMPRESSED_SIZE = 0, @@GLOBAL.innodb_page_size,
       COMPRESSED_SIZE))/1024/1024)
       AS '총 데이터(MB)'
       에서 정보_schema.innodb_버퍼_페이지
       WHERE TABLE_NAME = ``employees`.`salaries``
       GROUP BY INDEX_NAME;
+------------+-------+-------------------+
| 인덱스 이름 | 페이지 | 총 데이터(MB) |
+------------+-------+-------------------+
|            emp_no| 1608           |25 |
             | 기본| 6086           |95 |
+------------+-------+-------------------+
```

예 15.9 INNODB_BUFFER_PAGE_LRU 테이블에서 LRU_POSITION 데이터 쿼리하기

INNODB_BUFFER_PAGE_LRU 테이블에는 InnoDB 버퍼 풀의 페이지에 대한 정보, 특히 버퍼 풀이 가득 차면 어떤 페이지를 버퍼 풀에서 제거할지 결정하는 페이지 정렬 방식에 대한 정보가 들어 있습니다. 이 페이지에 대한 정의는 INNODB_BUFFER_PAGE와 동일하지만 이 테이블에는 BLOCK_ID 열 대신 LRU_POSITION 열이 있다는 점이 다릅니다.

이 쿼리는 employees.employees 테이블의 페이지가 차지하는 LRU 목록의 특정 위치에서 위치 수를 계산합니다.

```
mysql> SELECT COUNT(LRU_POSITION) FROM INFORMATION_SCHEMA.INNODB_BUFFER_PAGE_LRU
       WHERE TABLE_NAME=``employees`.`employees`' AND LRU_POSITION < 3072;
+---------------------+
| count(lru_position) |
+---------------------+
|                 548 |
+---------------------+
```

예 15.10 INNODB_BUFFER_POOL_STATS 테이블 쿼리하기

INNODB_BUFFER_POOL_STATS 테이블은 엔진 INNODB 상태 표시 및 InnoDB 버퍼 풀 상태 변수와 유사한 정보를 제공합니다.

```
mysql> SELECT * FROM information_schema.INNODB_BUFFER_POOL_STATS \G
*************************** 1. 행 ***************************
                         POOL_ID: 0
                       POOL_SIZE: 8192
                    FREE_BUFFERS: 1
                  데이터베이스_페이지: 8173
              old_database_pages: 3014
            수정된_데이터베이스_페이지: 0
              pending_decompress: 0
                   PENDING_READS: 0
               PENDING_FLUSH_LRU: 0
              PENDING_FLUSH_LIST: 0
                pages_made_young: 15907
            pages_not_made_young: 3803101
          pages_made_young_rate: 0
       pages_made_not_young_rate: 0
                number_pages_read: 3270
            number_pages_created: 13176
            number_pages_written: 15109
                 pages_read_rate: 0
              pages_create_rate: 0
              pages_written_rate: 0
                number_pages_get: 33069332
                        HIT_RATE: 0
    young_make_per_thousand_gets: 0
NOT_YOUNG_MAKE_PER_THOUSAND_GETS: 0
```

```
                번호_페이지_읽기_앞: 2713
       number_read_ahead_evicted: 0
                 READ_Ahead_RATE: 0
         READ_Ahead_EVICTED_RATE: 0
                    LRU_IO_TOTAL: 0
                  lru_io_current: 0
                uncompress_total: 0
             uncompress_current: 0
```

비교를 위해, 동일한 데이터 세트를 기반으로 한 엔진 INNODB 상태 표시 출력과 InnoDB 버퍼 풀 상태 변수 출력은 아래와 같습니다.

엔진 INNODB 상태 표시 출력에 대한 자세한 내용은 15.17.3절. "InnoDB 표준 모니터 및 잠금 모니터 출력"을 참조하세요.

```
mysql> SHOW ENGINE INNODB STATUS \G
...
----------------------------
버퍼 풀 및 메모리
----------------------------
할당된 총 대용량 메모리 137428992 할당된 사전
메모리 579084 버퍼 풀     크기8192
여유 버퍼             1
데이터베이스 페이지 8173 이
전 데이터베이스 페이지 3014
수정된 데이터베이스 페이지 0
보류 중인 읽기 0
쓰기 보류 중입니다: LRU 0, 플러시 목록 0, 단일 페이지 0 페이지가
영 15907이 아닌 영 3803101로 만들어졌습니다.
0.00 영/초, 0.00 비영/초
페이지 읽기 3270, 생성 13176, 쓰기 15109
0.00 읽기/초, 0.00 생성/초, 0.00 쓰기/초
마지막 인쇄 이후 버퍼 풀 페이지를 가져올 수 없습니다.
페이지 앞 읽기 0.00초, 액세스 없이 퇴거 0.00초, 무작위 앞 읽기 0.00초 LRU len: 8173,
unzip_LRU len: 0
I/O 합계[0]:cur[0], 압축 해제 합계[0]:cur[0]
...
```

상태 변수에 대한 설명은 5.1.10절. "서버 상태 변수"를 참조하세요.

```
mysql> SHOW STATUS LIKE 'Innodb_buffer%';
+---------------------------------------+-----------------+
| 변수_이름| 값                                 |
+---------------------------------------+-----------------+
| Innodb_버퍼_풀_덤프_상태                  | 시작되지 않음   |
| Innodb_버퍼_풀_로드_상태                  | 시작되지 않음   |
| Innodb_버퍼_풀_크기조정_상태              | 시작되지 않음   |
| Innodb_버퍼_풀_페이지_데이터              | 8173          |
| Innodb_버퍼_풀_바이트_데이터              | 133906432     |
| Innodb_버퍼_풀_페이지_dirty               | 0             |
| Innodb_버퍼_풀_바이트_dirty               | 0             |
| Innodb_버퍼_풀_페이지_플로시드            | 15109         |
| Innodb_버퍼_풀_페이지_무료                | 1             |
| Innodb_버퍼_풀_페이지_misc                | 18            |
| Innodb_버퍼_풀_페이지_총계                | 8192          |
| Innodb_버퍼_풀_read_ahead_rnd             | 0             |
| Innodb_버퍼_풀_read_ahead                 | 2713          |
| Innodb_buffer_pool_read_ahead_evicted | 0             |
| Innodb_버퍼_풀_읽기_요청                  | 33069332      |
| Innodb_버퍼_풀_읽기                       | 558           |
| Innodb_버퍼_풀_대기_무료                  | 0             |
|
                                        Innodb_buffer_p
ool_write_requests| 11985961              |
+---------------------------------------+-----------------+
```

15.15.6 InnoDB 정보_화학 메트릭 테이블

INNODB_METRICS 테이블은 InnoDB 성능 및 리소스 관련 카운터에 대한 정보를 제공합니다.

INNODB_METRICS 테이블 열은 아래와 같습니다. 열에 대한 설명은 26.4.21절, "INFORMATION_SCHEMA INNODB_METRICS 테이블"을 참조하세요.

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME="dml_inserts" \G
*************************** 1. 행 ***************************
           NAME: dml_inserts
      서브시스템: dml 카
          운트: 46273
      최대_수: 46273
```

```
      MIN_COUNT: NULL
      평균_수: 492.2659574468085
    COUNT_RESET: 46273
MAX_COUNT_RESET: 46273
MIN_COUNT_RESET: NULL
AVG_COUNT_RESET: NULL
   TIME_ENABLE: 2014-11-28 16:07:53
  TIME_DISABLE: NULL
   TIME_ELAPSED: 94
     TIME_RESET: NULL
         상태: 활성화됨
           유형: 상태_카운터
        코멘트: 삽입된 행 수
```

카운터 활성화, 비활성화 및 재설정하기

다음 변수를 사용하여 카운터를 활성화, 비활성화 및 재설정할 수 있습니다:

- `innodb_monitor_enable`: 카운터를 활성화합니다.

  ```
  SET GLOBAL innodb_monitor_enable = [카운터명|모듈명|패턴|모두];
  ```

- `INNODB_MONITOR_DISABLE`: 카운터를 비활성화합니다.

  ```
  SET GLOBAL innodb_monitor_disable = [카운터명|모듈명|패턴|모두];
  ```

- `INNODB_MONITOR_RESET`: 카운터 값을 0으로 초기화합니다.

  ```
  SET GLOBAL innodb_monitor_reset = [카운터명|모듈명|패턴|모두];
  ```

- `INNODB_MONITOR_RESET_ALL`: 모든 카운터 값을 초기화합니다. 카운터를 사용하기 전에 비활성화해야 합니다. `innodb_monitor_reset_all`.

  ```
  SET GLOBAL innodb_monitor_reset_all = [카운터명|모듈명|패턴|모두];
  ```

카운터 및 카운터 모듈은 MySQL 서버 구성 파일을 사용하여 시작할 때 활성화할 수도 있습니다. 예를 들어, 로그 모듈인 `metadata_table_handles_opened` 및 `metadata_table_handles_closed` 카운터를 활성화하려면 MySQL 서버 구성 파일의 `[mysqld]` 섹션에 다음 줄을 입력합니다.

```
[mysqld]
innodb_monitor_enable = 로그, 메타데이터_테이블_핸들_열림, 메타데이터_테이블_핸들_닫힘
```

구성 파일에서 여러 카운터 또는 모듈을 활성화하는 경우, 위와 같이 `innodb_monitor_enable` 변수 뒤에 카운터 및 모듈 이름을 쉼표로 구분하여 지정합니다. 구성 파일에는 `innodb_monitor_enable` 변수만 사용할 수 있습니다.

`innodb_monitor_disable` 및 `innodb_monitor_reset` 변수는 명령줄에서만 지원됩니다.

참고

각 카운터는 어느 정도의 런타임 오버헤드를 추가하므로 프로덕션 서버에서는 특정 문제를 진단하거나 특정 기능을 모니터링할 때 카운터를 보수적으로 사용하세요. 카운터를 보다 광범위하게 사용하려면 테스트 또는 개발 서버를 사용하는 것이 좋습니다.

카운터

사용 가능한 카운터 목록은 변경될 수 있습니다. 정보 스키마 `INNODB_METRICS` 쿼리하기 테이블을 사용하여 MySQL 서버 버전에서 카운터를 사용할 수 있습니다.

기본적으로 활성화된 카운터는 `SHOW ENGINE INNODB STATUS` 출력에 표시되는 카운터와 일치합니다. `SHOW ENGINE INNODB STATUS` 출력에 표시되는 카운터는 시스템 수준에서 항상 활성화되어 있지만 `INNODB_METRICS` 테이블에 대해서는 비활성화할 수 있습니다. 카운터 상태는 영구적이지 않습니다. 달리 구성하지 않는 한 서버가 다시 시작되면 카운터는 기본 활성화 또는 비활성화 상태로 되돌아갑니다.

카운터 추가 또는 제거의 영향을 받는 프로그램을 실행하는 경우, 릴리스 노트를 검토하고 INNODB_METRICS 테이블을 쿼리하여 업그레이드 프로세스의 일부로 해당 변경 사항을 식별하는 것이 좋습니다.

```
mysql> SELECT name, subsystem, status FROM INFORMATION_SCHEMA.INNODB_METRICS ORDER BY NAME;
+----------------------------------------+--------------------+-----------------+
                                         | 이름|                 하위 시스템| 상태 |
+----------------------------------------+--------------------+-----------------+
| 적응형 해시 페이지 추가 |                 적응형 해시 인덱스 | 비활성화 |
| 적응형 해시 페이지 제거 |                 적응형 해시 인덱스 | 비활성화 |
| 적응형 해시 행 추가 |                    적응형 해시 인덱스 | 비활성화 |
| 적응형 해시 행 삭제됨_no_해시 항목 |      적응형 해시 인덱스 | 사용 안 함 |
| 적응형 해시 행 제거 |                    적응형 해시 인덱스 | 사용 안 함 |
| 적응형 해시 행 업데이트 |                 적응형 해시 인덱스 | 비활성화 |
| 적응형 해시 검색 |                    적응형 해시 인덱스 | 활성화됨 |
| 적응형_해시 검색_비트리 |                 적응형_해시 인덱스 | 활성화됨 |
|                                        buffer_data_reads| 버퍼|       활성화 |
|                                        buffer_data_written| buffer| 활성화 |
| 버퍼_                                  플러시_적응형| 버퍼|       비활성화 |
                                         | 버퍼_플러시_어댑티브_AVG_패스| 버퍼|     비활성화 |
                                         | 버퍼_플러시_적응형_평균_시간_최고| 버퍼|  비활성화 |
                                         | 버퍼_플러시_어댑티브_평균_시간_슬롯| 버퍼| 비활성화 |
                                         | 버퍼_플러시_적응형_평균_시간_스레드| 버퍼|         비활성화 |
                                         | 버퍼_플러시_적응형_페이지| 버퍼| 비활성화 |
| 버퍼                                   플러시_적응형_총_페이지| 버퍼|     비활성화 |
                                         | 버퍼_플러시_평균_페이지_속도| 버퍼|       비활성화 |
| 버퍼 플러시                            AVG 패스 | 버퍼 |        비활성화 |
                                         | 버퍼_플러시_평균_시간| 버퍼|     비활성화 |
                                         | 버퍼_플러시_백그라운드| 버퍼|     비활성화 |
                                         | 버퍼_플러시_백그라운드_페이지| 버퍼|       비활성화 |
                                         | 버퍼_플러시_백그라운드_총_페이지| 버퍼|    비활성화 |
                                         | 버퍼_플러시_배치| 버퍼| 비활성화 |
| 버퍼_플러시_배치_넘버_스캔| 버퍼|       비활성화 |
                                         | 버퍼_플러시_배치_페이지| 버퍼|    비활성화 |
| 버퍼 플러시                            배치 스캔| 버퍼|          비활성화 |
|                                        buffer_flush_batch_scanned_per_call| 버퍼|    비활성화 |
| 버퍼_                                  플러시_배치_총_페이지| 버퍼|       비활성화 |
                                         | 버퍼_플러시_LSN_AVG_RATE| 버퍼|       비활성화 |
                                         | 버퍼_플러시_이웃| 버퍼| 비활성화 |
                                         | 버퍼_플러시_이웃_페이지| 버퍼|    비활성화 |
                                         | 버퍼_플러시_이웃_총_페이지| 버퍼|       비활성화 |
                                         | 버퍼_플러시_엔_투_플러시_바이_에이징| 버퍼|         비활성화 |
|                                        buffer_flush_n_to_flush_by_dirty_page| 버퍼| 비활성화 |
                                         | 버퍼_플러시_n_to_플러시_요청된| 버퍼|     비활성화 |
|                                        buffer_flush_pct_for_dirty| 버퍼|     비활성화 |
| 버퍼_플러시_PCT_FOR_LSN| 버퍼|       비활성화 |
                                         | 버퍼_플러시_동기화| 버퍼|       비활성화 |
|                                        buffer_flush_sync_pages| 버퍼|        비활성화 |
```

버퍼	플러시 동기화 총 페이지	버퍼	비활성화
버퍼	플러시 동기화 대기	버퍼	비활성화
	버퍼_LRU_배치_이벤트	버퍼	비활성화
	버퍼_LRU_배치_ 플러시	버퍼	비활성화
	buffer_LRU_batch_evict_pages	버퍼	비활성화
	buffer_LRU_batch_evict_total_pages	버퍼	비활성화
	buffer_LRU_batch_flush_avg_pass	버퍼	비활성화
	buffer_LRU_batch_flush_avg_time_est	buffer	비활성화
	buffer_LRU_batch_flush_avg_time_slot	buffer	비활성화
	buffer_LRU_batch_flush_avg_time_thread	버퍼	비활성화
	buffer_LRU_batch_flush_pages	버퍼	비활성화
	buffer_LRU_batch_flush_total_pages	버퍼	비활성화
	buffer_LRU_batch_num_scan	버퍼	비활성화
	버퍼_LRU_배치_스캔	버퍼	비활성화
	buffer_LRU_batch_scanned_per_call	버퍼	비활성화
	buffer_LRU_get_free_loops	버퍼	비활성화
	buffer_LRU_get_free_search	버퍼	비활성화
	buffer_LRU_get_free_waits	버퍼	비활성화
	buffer_LRU_search_num_scan	버퍼	비활성화
	buffer_LRU_search_scanned	버퍼	비활성화
	buffer_LRU_search_scanned_per_call	버퍼	비활성화
	buffer_LRU_single_flush_실패 횟수	버퍼	비활성화
	buffer_LRU_single_flush_num_scan	버퍼	비활성화
	buffer_LRU_single_flush_scanned	버퍼	비활성화

```
| buffer_LRU_single_flush_scanned_per_call| 버퍼| 비활성화 |
| buffer_LRU_unzip_search_num_scan| 버퍼| 비활성화 |
| buffer_LRU_unzip_search_scanned| 버퍼| 비활성화 |
| buffer_LRU_unzip_search_scanned_per_call| buffer| 비활성
화 |
| buffer_pages_created| buffer| 활성화 |
| 버퍼_페이지_읽기| 버퍼| 활성화 |
| 버퍼_페이지_쓰기| 버퍼| 활성화 |
| buffer_page_read_blob| buffer_page_io| 비활성화 |
| buffer_page_read_fsp_hdr| buffer_page_io| 비활성화 |
| buffer_page_read_ibuf_bitmap| buffer_page_io| disabled |
| buffer_page_read_ibuf_free_list| buffer_page_io|
disabled |
| buffer_page_read_index_ibuf_leaf| buffer_page_io| 비활성
화 |
| buffer_page_read_index_ibuf_non_leaf| buffer_page_io| 비활성
화 |
| buffer_page_read_index_inode| buffer_page_io| disabled |
| buffer_page_read_index_leaf| buffer_page_io| 비활성화 |
| buffer_page_read_index_non_leaf| buffer_page_io| 비활성
화 |
| buffer_page_read_other| buffer_page_io| 비활성화 |
| buffer_page_read_rseg_array| buffer_page_io| disabled |
| buffer_page_read_system_page| buffer_page_io| disabled |
| buffer_page_read_trx_system| buffer_page_io| disabled |
| buffer_page_read_undo_log| buffer_page_io| 비활성화 |
| buffer_page_read_xdes| buffer_page_io| 비활성화 |
| buffer_page_read_zblob| buffer_page_io| disabled |
| buffer_page_read_zblob2| buffer_page_io| disabled |
| buffer_page_written_blob| buffer_page_io| disabled |
| buffer_page_written_fsp_hdr| buffer_page_io| disabled |
| buffer_page_written_ibuf_bitmap| buffer_page_io|
disabled |
| buffer_page_written_ibuf_free_list| buffer_page_io|
disabled |
| buffer_page_written_index_ibuf_leaf| buffer_page_io|
disabled |
| buffer_page_written_index_ibuf_non_leaf| buffer_page_io|
비활성화 |
| buffer_page_written_index_inode| buffer_page_io|
disabled |
| buffer_page_written_index_leaf| buffer_page_io| disabled |
| buffer_page_written_index_non_leaf| buffer_page_io| 비활성
화 |
| buffer_page_written_on_log_no_waits| buffer_page_io|
disabled |
| buffer_page_written_on_log_waits| buffer_page_io|
disabled |
| buffer_page_written_on_log_wait_loops| buffer_page_io|
disabled |
| buffer_page_written_other| buffer_page_io| 비활성화 |
| buffer_page_written_rseg_array| buffer_page_io| disabled |
| buffer_page_written_system_page| buffer_page_io|
disabled |
| buffer_page_written_trx_system| buffer_page_io| disabled |
| buffer_page_written_undo_log| buffer_page_io| disabled |
| buffer_page_written_xdes| buffer_page_io| disabled |
| buffer_page_written_zblob| buffer_page_io| disabled |
| buffer_page_written_zblob2| buffer_page_io| disabled |
| buffer_pool_bytes_data| buffer| 활성화 |
```

	buffer_pool_bytes_dirty	buffer	활성화
	buffer_pool_pages_data	buffer	활성화
	buffer_pool_pages_dirty	버퍼	활성화
	buffer_pool_pages_free	buffer	활성화
	buffer_pool_pages_misc	buffer	활성화
	buffer_pool_pages_total	buffer	활성화
	buffer_pool_reads	버퍼	활성화
	buffer_pool_read_ahead	버퍼	활성화
	buffer_pool_read_ahead_evicted	버퍼	활성화
	buffer_pool_read_requests	버퍼	활성화
	buffer_pool_size	서버	활성화
	buffer_pool_wait_free	buffer	활성화
	buffer_pool_write_requests	buffer	활성화
	압축_패드_감소	압축	비활성화
	압축_패드_증분	압축	비활성화
압축_페이지_압축	압축	비활성화	
	압축_페이지_해제	압축	비활성화
CPU-N	CPU	비활성화	
CPU_STIME_ABS	CPU	비활성화	
CPU_STIME_PCT	CPU	비활성화	
CPU_UTIME_ABS	CPU	비활성화	
CPU_UTIME_PCT	CPU	비활성화	
dblwr_async_requests	dblwr	disabled	
dblwr_flush_requests	dblwr	비활성화됨	
dblwr_flush_wait_events	dblwr	disabled	
dblwr_sync_requests	dblwr	disabled	
ddl_백그라운드_드롭_테이블	ddl	비활성화	

DDL_LOG_FILE_ALTER_TABLE	ddl	disabled
DDL_ONLINE_CREATE_INDEX	ddl	disabled
DDL_PENDING_ALTER_TABLE	ddl	disabled
DDL_SORT_FILE_ALTER_TABLE	ddl	disabled
dml_deletes	dml	활성화
dml_inserts	dml	활성화
dml_reads	dml	disabled
DML_SYSTEM_DELETES	dml	활성화
DML_SYSTEM_INSERT	dml	활성화
DML_SYSTEM_READS	dml	활성화
dml_system_updates	dml	활성화
dml_updates	dml	활성화
FILE_NUM_OPEN_FILES	file_system	활성화
ibuf_merges	변경_버퍼	활성화
ibuf_merges_delete	변경_버퍼	활성화
ibuf_merges_delete_mark	변경_버퍼	활성화
ibuf_merges_discard_delete	변경_버퍼	활성화
ibuf_merges_discard_delete_mark	변경_버퍼	활성화
IBUF_MERGES_DISCARD_INSERT	변경_버퍼	활성화
ibuf_merges_insert	변경_버퍼	활성화
ibuf_size	변경_버퍼	활성화
icp_attempt	icp	disabled
icp_match	icp	disabled
icp_no_match	icp	disabled
ICP_OUT_OF_RANGE	icp	disabled
index_page_discards	index	disabled
index_page_merge_시도	index	disabled
index_page_merge_successful	index	disabled
index_page_reorg_시도	index	disabled
index_page_reorg_successful	index	disabled
index_page_splits	index	disabled
innodb_activity_count	서버	활성화
innodb_background_drop_table_usec	서버	disabled
innodb_dblwr_pages_witten	서버	활성화
innodb_dblwr_writes	서버	활성화
INNODB_DICT_LRU_COUNT	서버	disabled
INNODB_DICT_LRU_USEC	서버	disabled
INNODB_IBUF_MERGE_USEC	서버	disabled
innodb_master_active_loops	서버	disabled
innodb_master_idle_loops	서버	disabled
innodb_master_purge_usec	서버	disabled
innodb_master_thread_sleeps	서버	disabled
innodb_mem_validate_usec	서버	disabled
innodb_page_size	서버	활성화
INNODB_RWLOCK_SX_OS_WAITS	서버	활성화
INNODB_RWLOCK_SX_SPIN_ROUNDS	서버	활성화
INNODB_RWLOCK_SX_SPIN_WAITS	서버	활성화
INNODB_RWLOCK_S_OS_WAITS	서버	활성화
INNODB_RWLOCK_S_SPIN_ROUNDS	서버	활성화
INNODB_RWLOCK_S_SPIN_WAITS	서버	활성화
INNODB_RWLOCK_X_OS_WAITS	서버	활성화
INNODB_RWLOCK_X_SPIN_ROUNDS	서버	활성화
INNODB_RWLOCK_X_SPIN_WAITS	서버	활성화
lock_deadlock	잠금	활성화
LOCK_DEADLOCK_FALSE_POSITIVES	잠금	활성화
LOCK_DEADLOCK_ROUNDS	잠금	활성화
LOCK_RE_C_GRANT_ATTEMPS	잠금	활성화
LOCK_RE_C_LOCKS	잠금	disabled
LOCK_RE_C_LOCK_CREATED	잠금	disabled
lock_rec_lock_removed	잠금	disabled
LOCK_RE_C_LOCK_REQUESTS	잠금	disabled
LOCK_RE_C_LOCK_WAITS	잠금	disabled
잠금_해제_시도	잠금	활성화
LOCK_ROW_LOCK_CURRENT_WAITS	잠금	활성화
LOCK_ROW_LOCK_TIME	잠금	활성화
LOCK_ROW_LOCK_TIME_AVG	잠금	활성화
LOCK_ROW_LOCK_TIME_MAX	잠금	활성화
LOCK_ROW_LOCK_WAITS	잠금	활성화
LOCK_SCHEDULE_REFRESH	잠금	활성화
LOCK_TABLE_LOCKS	잠금	disabled
LOCK_TABLE_LOCK_CREATED	잠금	disabled
lock_table_lock_removed	잠금	disabled

LOCK_TABLE_LOCK_WAITS	잠금	disabled
LOCK_THREADS_WAITING	잠금	활성화
lock_timeouts	잠금	활성화
로그_체크포인트	로그	disabled
로그_통화_마진	로그	disabled
log_flusher_no_waits	로그	disabled
log_flusher_waits	로그	disabled
로그_플러셔_대기_루프	로그	disabled
LOG_FLUSH_AVG_TIME	로그	disabled
LOG_FLUSH_LSN_AVG_RATE	로그	disabled
LOG_FLUSH_MAX_TIME	로그	disabled
log_flush_notifier_no_waits	로그	disabled
log_flush_notifier_waits	로그	disabled
로그_플러시_알림_대기_루프	로그	disabled
LOG_FLUSH_TOTAL_TIME	로그	disabled
로그_자유공간	로그	disabled
로그_풀_블록_쓰기	로그	disabled
로그_LSN_아카이브	로그	disabled
log_lsn_buf_dirty_pages_added	로그	disabled
로그_LSN_BUF_풀_최근_약정	로그	disabled
LOG_LSN_BUF_POOL_OLDEST_LWM	로그	disabled
로그_LSN_체크포인트_연령	로그	disabled
LOG_LSN_CURRENT	로그	disabled
로그_LSN_마지막_체크포인트	로그	disabled
LOG_LSN_LAST_FLUSH	로그	disabled
로그_최대_수정_연령_동기화	로그	disabled
로그_최대_수정_연령_동기화	로그	disabled
로그_다음_파일	로그	disabled
로그온_버퍼_공간_대기_없음	로그	disabled
로그온_버퍼_공간_대기	로그	disabled
로그온_버퍼_공간_대기_루프	로그	disabled
로그온_파일_공간_대기_없음	로그	disabled
로그온_파일_공간_대기	로그	disabled
로그온_파일_공간_대기_루프	로그	disabled
로그온_플러시_대기_없음	로그	disabled
로그온_플러시_대기	로그	disabled
로그온_플러시_대기_루프	로그	disabled
로그온_최근_닫힘_대기_루프	로그	disabled
로그온_최근_기록_대기_루프	로그	disabled
로그온_쓰기_대기_없음	로그	disabled
로그온_쓰기_대기	로그	disabled
로그온_쓰기_대기_루프	로그	disabled
log_padded	로그	disabled
로그_부분_블록_쓰기	로그	disabled
log_waits	로그	활성화
log_writer_no_waits	로그	disabled
log_writer_on_archiver_waits	로그	disabled
로그_라이터_온_파일_공간_대기	로그	disabled
log_writer_waits	로그	disabled
로그_작성기_대기_루프	로그	disabled
로그_쓰기	로그	활성화
로그_쓰기_알림기_대기_없음	로그	disabled
로그_쓰기_알림기_대기	로그	disabled
로그_쓰기_알림_대기_루프	로그	disabled
로그_쓰기_요청	로그	활성화
로그_쓰기_투_파일_요청_간격	로그	disabled
메타데이터_테이블_핸들_닫힘	메타데이터	disabled
메타데이터_테이블_핸들_열림	메타데이터	disabled
메타데이터_테이블_참조_수	메타데이터	disabled
module_cpu	cpu	disabled
module_dblwr	dblwr	disabled
MODULE_PAGE_TRACK	페이지_트랙	disabled
OS_DATA_FSYNC	os	활성화
OS_DATA_READS	os	활성화
OS_DATA_WRITES	os	활성화
OS_LOG_BYTE_WRITEN	os	활성화
OS_LOG_FSYNC	os	활성화
OS_LOG_PENDING_FSYNCS	os	활성화
OS_LOG_PENDING_WRITES	os	활성화
OS_PENDING_READS	os	disabled
OS_PENDING_WRITES	os	disabled
페이지_트랙_체크포인트_부분_ 플러시_요청	페이지_트랙	disabled

```
| 페이지_트랙_풀_블록_쓰기                          | 페이지_트랙          | disabled |
| 페이지_트랙_부분_블록_쓰기                        | 페이지_트랙          | disabled |
| 페이지_트랙_리셋                                  | 페이지_트랙          | disabled |
| PURGE_DEL_MARK_RECORDS                            | 퍼지                 | disabled |
| PURGE_DML_DELAY_USEC                              | 퍼지                 | disabled |
| 퍼지_인보크                                       | 퍼지                 | disabled |
| PURGE_RESUME_COUNT                                | 퍼지                 | disabled |
| PURGE_STOP_COUNT                                  | 퍼지                 | disabled |
| PURGE_TRUNCATE_HISTORY_COUNT                      | 퍼지                 | disabled |
| PURGE_TRUNCATE_HISTORY_USEC                       | 퍼지                 | disabled |
| purge_undo_log_pages                              | 퍼지                 | disabled |
| PURGE_UP_EXIST_OR_EXTERNAL_RECORDS                | 퍼지                 | disabled |
| 샘플링된 페이지 읽기                              | 샘플링               | disabled |
| sampled_pages_skipped                             | 샘플링               | disabled |
| TRX_ACTIVE_TANSACTIONS                            | 트랜잭션             | disabled |
| TRX_ALLOCATIONS                                   | 트랜잭션             | disabled |
| TRX_COMMIT_INSERT_업데이트                        | 트랜잭션             | disabled |
| TRX_NL_RO_COMMITS                                 | 트랜잭션             | disabled |
| TRX_ON_LOG_NO_WAITS                               | 트랜잭션             | disabled |
| TRX_ON_LOG_WAITS                                  | 트랜잭션             | disabled |
| TRX_ON_LOG_WAIT_LOOPS                             | 트랜잭션             | disabled |
| trx_rollbacks                                     | 트랜잭션             | disabled |
| TRX_ROLLBACKS_SAFEPOINT                           | 트랜잭션             | disabled |
| TRX_ROLLBACK_ACTIVE                               | 트랜잭션             | disabled |
| TRX_RO_COMMITS                                    | 트랜잭션             | disabled |
| TRX_RSEG_CURRENT_SIZE                             | 트랜잭션             | disabled |
| TRX_RSEG_HISTORY_LEN                              | 트랜잭션             |  활성화  |
| TRX_RW_COMMITS                                    | 트랜잭션             | disabled |
| TRX_UNDO_SLOTS_CACHED                             | 트랜잭션             | disabled |
| TRX_UNDO_SLOTS_USED                               | 트랜잭션             | disabled |
| undo_truncate_count                               | 실행 취소            | disabled |
| undo_truncate_done_logging_count                  | 실행 취소            | disabled |
| undo_truncate_start_logging_count                 | 실행 취소            | disabled |
| undo_truncate_usec                                | 실행 취소            | disabled |
+---------------------------------------------------+----------------------+----------+

314행 세트(0.00초)
```

카운터 모듈

각 카운터는 특정 모듈과 연관되어 있습니다. 모듈 이름을 사용하여 특정 서브시스템의 모든 카운터를 활성화, 비활성화 또는 재설정할 수 있습니다. 예를 들어, `dml` 하위 시스템과 연결된 모든 카운터를 활성화하려면 `module_dml`을 사용합니다.

```
mysql> SET GLOBAL innodb_monitor_enable = module_dml;

mysql> SELECT name, subsystem, status FROM INFORMATION_SCHEMA.INNODB_METRICS
       WHERE subsystem ='dml';
+-------------+-----------+ -----------+
| 이름 |        하위 시스템 | 상태
+-------------+-----------+ -----------+
| dml_reads | dml          | 활성화 | 사
| dml_inserts | dml
| dml_deletes | dml        용
| dml_updates | dml
+-------------+-----------+ -- 활성화 | 사
                           용
                           | 활성화 | 사
                           용
                           | 활성화 | 사
                           용
```

모듈 이름은 `innodb_monitor_enable` 및 관련 변수와 함께 사용할 수 있습니다. 모듈 이름과 해당 서브시스템 이름은 아래에 나열되어 있습니다.

- module_adaptive_hash (서브시스템 = adaptive_hash_index)
- 모듈_버퍼 (서브시스템 = 버퍼)
- 모듈_버퍼_페이지 (서브시스템 = buffer_page_io)
- module_compress (서브시스템 = 압축)
- MODULE_DDL (서브시스템 = DDL)
- 모듈_ml (서브시스템 = dml)

- 모듈_파일 (서브시스템 = 파일_시스템)
- module_ibuf_system (서브시스템 = change_buffer)
- module_icp (서브시스템 = icp)
- 모듈_인덱스 (서브시스템 = 인덱스)
- module_innodb (서브시스템 = innodb)
- 모듈_잠금 (서브시스템 = 잠금)
- module_log (서브시스템 = 로그)
- 모듈_메타데이터(서브시스템 = 메타데이터)
- module_os (서브시스템 = os)
- 모듈_퍼지(서브시스템 = 퍼지)
- module_trx (서브시스템 = 트랜잭션)
- module_undo (서브시스템 = 실행 취소)

예 15.11 INNODB_METRICS 테이블 카운터로 작업하기

이 예제에서는 카운터 활성화, 비활성화 및 재설정, INNODB_METRICS 테이블의 카운터 데이터 쿼리를 보여줍니다.

1. 간단한 InnoDB 테이블을 만듭니다:

```
mysql> USE 테스트;
데이터베이스 변경

mysql> CREATE TABLE t1 (c1 INT) ENGINE=INNODB;
쿼리 확인, 영향을 받는 행 0개(0.02초)
```

2. dml_inserts 카운터를 활성화합니다.

```
mysql> SET GLOBAL innodb_monitor_enable = dml_inserts;
쿼리 확인, 영향을 받는 행 0개(0.01초)
```

dml_inserts 카운터에 대한 설명은 INNODB_METRICS 테이블:

```
mysql> SELECT NAME, COMMENT FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME="dml_inserts";
+-------------+---------------------------+
| 이름| 댓글     |
+-------------+---------------------------+
| dml_inserts | 삽입된 행 수 |
+-------------+---------------------------+
```

3. INNODB_METRICS 테이블에서 dml_inserts 카운터 데이터를 쿼리합니다. DML 작업이 수행되지 않았으므로 카운터 값은 0 또는 NULL입니다. TIME_ENABLED 및 TIME_ELAPSED 값은 카운터가 마지막으로 활성화된 시점과 그 시점으로부터 경과한 시간(초)을 나타냅니다.

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME="dml_inserts" \G
*************************** 1. 행 ***************************
           NAME: dml_inserts
      서브시스템: dml 카운
          트: 0
      최대_수: 0 최소_수:
      NULL AVG_COUNT: 0
    COUNT_RESET: 0
MAX_COUNT_RESET: 0
MIN_COUNT_RESET: NULL
AVG_COUNT_RESET: NULL
```

```
   TIME_ENABLE: 2014-12-04 14:18:28
  TIME_DISABLE: NULL
   TIME_ELAPSED: 28
     TIME_RESET: NULL
         상태: 활성화됨
           유형: 상태_카운터
        코멘트: 삽입된 행 수
```

4. 테이블에 세 행의 데이터를 삽입합니다.

```
mysql> INSERT INTO t1 values(1);
쿼리 확인, 영향을 받은 행 1개(0.00초)

mysql> INSERT INTO t1 values(2);
쿼리 확인, 영향을 받은 행 1개(0.00초)

mysql> INSERT INTO t1 values(3);
쿼리 확인, 영향을 받은 행 1개(0.00초)
```

5. INNODB_METRICS 테이블에서 dml_inserts 카운터 데이터를 다시 쿼리합니다. 이제 카운트, MAX_COUNT, AVG_COUNT, COUNT_RESET을 포함한 여러 카운터 값이 증가했습니다. 이러한 값에 대한 설명은 INNODB_METRICS 테이블 정의를 참조하세요.

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME="dml_inserts"\G
*************************** 1. 행 ***************************
           NAME: dml_inserts
      서브 시스템: dml 카운
          트: 3
      최대_수: 3 최소_수:
      NULL
       AVG_COUNT: 0.046153846153846156
    COUNT_RESET: 3
MAX_COUNT_RESET: 3
MIN_COUNT_RESET: NULL
AVG_COUNT_RESET: NULL
   TIME_ENABLE: 2014-12-04 14:18:28
  TIME_DISABLE: NULL
   TIME_ELAPSED: 65
     TIME_RESET: NULL
         상태: 활성화됨
              유형: 상태_카운터
        코멘트: 삽입된 행 수
```

6. dml_inserts 카운터를 재설정하고 INNODB_METRICS 테이블에 dml_inserts 카운터 데이터를 다시 쿼리합니다. 이전에 보고된 %_RESET 값(예: COUNT_RESET 및 MAX_RESET)은 0으로 다시 설정됩니다. 카운터가 활성화된 시점부터 누적적으로 데이터를 수집하는 COUNT, MAX_COUNT, AVG_COUNT와 같은 값은 재설정의 영향을 받지 않습니다.

```
mysql> SET GLOBAL innodb_monitor_reset = dml_inserts;
쿼리 확인, 영향을 받은 행 0개(0.00초)

mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME="dml_inserts"\G
*************************** 1. 행 ***************************
           NAME: dml_inserts
      서브 시스템: dml 카운
          트: 3
      최대_수: 3 최소_수:
      NULL
       AVG_COUNT: 0.03529411764705882
    COUNT_RESET: 0
MAX_COUNT_RESET: 0
MIN_COUNT_RESET: NULL
AVG_COUNT_RESET: 0
   TIME_ENABLE: 2014-12-04 14:18:28
  TIME_DISABLE: NULL
   TIME_ELAPSED: 85
     시간 재설정: 2014-12-04 14:19:44
         상태: 활성화됨
              유형: 상태_카운터
```

```
        코멘트: 삽입된                                                    행 수
```

7. 모든 카운터 값을 초기화하려면 먼저 카운터를 비활성화해야 합니다. 카운터를 비활성화하면 `STATUS` 값을 비활성화합니다.

```
mysql> SET GLOBAL innodb_monitor_disable = dml_inserts;
쿼리 확인, 영향을 받은 행 0개(0.00초)

mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME="dml_inserts"\G
*************************** 1. 행 ***************************
           NAME: dml_inserts
      서브 시스템: dml 카운
          트: 3
      최대_수: 3 최소_수:
      NULL
       AVG_COUNT: 0.030612244897959183
    COUNT_RESET: 0
MAX_COUNT_RESET: 0
MIN_COUNT_RESET: NULL
AVG_COUNT_RESET: 0
   시간_사용 가능: 2014-12-04 14:18:28
  시간_비활성화됨: 2014-12-04 14:20:06
   TIME_ELAPSED: 98
     TIME_RESET: NULL
         상태: 비활성화됨
              유형: 상태_카운터
        코멘트: 삽입된 행 수
```

참고

카운터 및 모듈 이름에 와일드카드 일치가 지원됩니다. 예를 들어 전체 `dml_inserts` 카운터 이름을 지정하는 대신 `dml_i%`를 지정할 수 있습니다. 와일드카드 일치를 사용하여 한 번에 여러 카운터 또는 모듈을 활성화, 비활성화 또는 재설정할 수도 있습니다. 예를 들어, `dml_`로 시작하는 모든 카운터를 활성화, 비활성화 또는 재설정하려면 `dml_%`를 지정합니다.

8. 카운터를 비활성화한 후 모든 카운터 값을 재설정하려면 `innodb_monitor_reset_all` 옵션. 모든 값이 0 또는 NULL로 설정됩니다.

```
mysql> SET GLOBAL innodb_monitor_reset_all = dml_inserts;
쿼리 확인, 영향을 받은 행 0개(0.00초)

mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME="dml_inserts"\G
*************************** 1. 행 ***************************
           NAME: dml_inserts
      서브시스템:  dml
          카운트: 0
      MAX_COUNT: NULL
      MIN_COUNT: NULL
      AVG_COUNT: NULL
    카운트_리셋: 0
MAX_COUNT_RESET: NULL
MIN_COUNT_RESET: NULL
AVG_COUNT_RESET: NULL
   TIME_ENABLED: NULL
  TIME_DISABLED: NULL
   TIME_ELAPSED: NULL
     TIME_RESET: NULL
         상태: 비활성화됨
           유형: 상태_카운터
        코멘트: 삽입된 행 수
```

15.15.7 InnoDB INFORMATION_SCHEMA 임시 테이블 정보 테이블

`INNODB_TEMP_TABLE_INFO`는 `InnoDB` 인스턴스에서 활성 상태인 사용자가 생성한 `InnoDB` 임시 테이블에 대한 정보를 제공합니다. 옵티마이저가 사용하는 내부 `InnoDB` 임시 테이블에 대한 정보는 제공하지 않습니다.

```
mysql> SHOW TABLES FROM INFORMATION_SCHEMA LIKE 'INNODB_TEMP%';
```

```
+-------------------------------------------+
| Tables_in_INFORMATION_SCHEMA (INNODB_TEMP%) |
+-------------------------------------------+
| innodb_temp_table_info                    |
+-------------------------------------------+
```

테이블 정의는 섹션 26.4.27, "정보 스키마 INNODB_TEMP_TABLE_INFO 테이블"을 참조하세요.

예제 15.12 INNODB_TEMP_TABLE_INFO

이 예는 INNODB_TEMP_TABLE_INFO 테이블의 특성을 보여줍니다.

1. 간단한 InnoDB 임시 테이블을 만듭니다:

```
mysql> CREATE TEMPORARY TABLE t1 (c1 INT PRIMARY KEY) ENGINE=INNODB;
```

2. 임시 테이블 메타데이터를 보려면 INNODB_TEMP_TABLE_INFO를 쿼리하세요.

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_TEMP_TABLE_INFO\G
*************************** 1. 행 ***************************
            TABLE_ID: 194
                NAME: #SQL7A79_1_0
              N_COLS: 4
               공간: 182
```

TABLE_ID는 임시 테이블의 고유 식별자입니다. NAME 열은 시스템에서 생성된 임시 테이블의 이름을 표시하며, 이 이름 앞에 "#sql"이 붙습니다. InnoDB는 항상 3개의 숨겨진 테이블 열(DB_ROW_ID, DB_TRX_ID 및 DB_ROLL_PTR)을 생성하므로 열 수(N_COLS)는 1이 아닌 4입니다.

3. MySQL을 다시 시작하고 INNODB_TEMP_TABLE_INFO를 쿼리합니다.

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_TEMP_TABLE_INFO\G
```

서버가 종료될 때 INNODB_TEMP_TABLE_INFO 및 해당 데이터가 디스크에 지속되지 않기 때문에 빈 집합이 반환됩니다.

4. 새 임시 테이블을 만듭니다.

```
mysql> CREATE TEMPORARY TABLE t1 (c1 INT PRIMARY KEY) ENGINE=INNODB;
```

5. 임시 테이블 메타데이터를 보려면 INNODB_TEMP_TABLE_INFO를 쿼리하세요.

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_TEMP_TABLE_INFO\G
*************************** 1. 행 ***************************
            TABLE_ID: 196
                NAME: #SQL7B0E_1_0
              N_COLS: 4
               공간: 184
```

스페이스 ID는 서버가 시작될 때 동적으로 생성되므로 다를 수 있습니다.

15.15.8 INFORMATION_SCHEMA.FILES에서 InnoDB 테이블스페이스 메타데이터 가져오기

정보 스키마 파일 테이블은 테이블별 파일 테이블 공간, 일반 테이블 공간, 시스템 테이블 공간, 임시 테이블 테이블 공간 및 실행 취소 테이블 공간(있는 경우)을 포함한 모든 `InnoDB` 테이블 공간 유형에 대한 메타데이터를 제공합니다.

이 섹션에서는 InnoDB 관련 사용 예제를 제공합니다. 정보 스키마 파일 테이블에서 제공하는 데이터에 대한 자세한 내용은 26.3.15절. "정보 스키마 파일 테이블"을 참조하세요.

참고

INNODB_TABLESPACES 및 INNODB_DATAFILES 테이블은 InnoDB 테이블 스페이스에 대한 메타데이터도 제공하지만 데이터는 테이블별 파일, 일반 및 실행 취소 테이블 공간으로 제한됩니다.

이 쿼리는 InnoDB 테이블 스페이스와 관련된 정보 스키마 FILES 테이블의 필드에서 InnoDB 시스템 테이블 스페이스에 대한 메타데이터를 검색합니다. InnoDB와 관련이 없는 FILES 열은 항상 NULL을 반환하며 쿼리에서 제외됩니다.

```
mysql> SELECT FILE_ID, FILE_NAME, FILE_TYPE, TABLESPACE_NAME, FREE_EXTENTS,
       TOTAL_EXTENTS, EXTENT_SIZE, INITIAL_SIZE, MAXIMUM_SIZE, AUTOEXTEND_SIZE, DATA_FREE, 상태 엔진을
       INFORMATION_SCHEMA.FILES에서 테이블스페이스_이름이 'innodb_system' \G처럼 선택한다.
*************************** 1. 행 ***************************
        FILE_ID: 0
      파일 이름: ./ibdata1 파일
      유형: 테이블스페이스
테이블스페이스_이름: innodb_system
   FREE_EXTENTS: 0
  TOTAL_EXTENTS: 12
    범위_크기: 1048576
   initial_size: 12582912
   maximum_size: null
자동 확장 크기: 67108864
      DATA_FREE: 4194304 엔진:
         정상
```

이 쿼리는 InnoDB 테이블별 파일 및 일반 테이블 스페이스에 대한 FILE_ID(스페이스 ID에 해당) 및 FILE_NAME(경로 정보 포함)을 검색합니다. 테이블별 파일 및 일반 테이블 스페이스의 파일 확장자는 .ibd입니다.

```
mysql> SELECT FILE_ID, FILE_NAME FROM INFORMATION_SCHEMA.FILES
       WHERE FILE_NAME LIKE '%.ibd%' ORDER BY FILE_ID;
    +---------+----------------------------------------+
    | 파일_ID | 파일_이름                              |
    +---------+----------------------------------------+

    |       2 | ./mysql/plugin.ibd                     |
    |       3 | ./mysql/servers.ibd                    |
    |       4 | ./mysql/help_topic.ibd                 |
    |       5 | ./mysql/help_category.ibd              |
    |       6 | ./mysql/help_relation.ibd              |
    |       7 | ./mysql/help_keyword.ibd               |
    |       8 | ./mysql/시간대_영역_이름.ibd           |
    |       9 | ./mysql/time_zone.ibd                  |
    |      10 | ./mysql/시간대_영역_전환.ibd           |
    |      11 | ./mysql/시간대_영역_전환_type.ibd      |
    |      12 | ./mysql/시간대_영역_리프_second.ibd    |
    |      13 | ./mysql/innodb_table_stats.ibd         |
    |      14 | ./mysql/innodb_index_stats.ibd         |
    |      15 | ./mysql/slave_relay_log_info.ibd       |
    |      16 | ./mysql/slave_master_info.ibd          |
    |      17 | ./mysql/slave_worker_info.ibd          |
    |      18 | ./mysql/gtid_executed.ibd              |
    |      19 | ./mysql/server_cost.ibd                |
    |      20 | ./mysql/engine_cost.ibd                |
    |      21 | ./sys/sys_config.ibd                   |
    |      23 | ./test/t1.ibd                          |
    |      26 | /home/user/test/test/t2.ibd            |
```

이 쿼리는 InnoDB 글로벌 임시 테이블스페이스에 대한 FILE_ID 및 FILE_NAME을 검색합니다. 글로벌 임시 테이블스페이스 파일 이름 앞에는 ibtmp가 붙습니다.

```
mysql> SELECT FILE_ID, FILE_NAME FROM INFORMATION_SCHEMA.FILES
       WHERE FILE_NAME LIKE '%ibtmp%';
+---------+------------+
| 파일_ID | 파일_이름 |
+---------+------------+
|22     | ./ibtmp1 |
+---------+------------+
```

기
마찬가지로 InnoDB 실행 취소 테이블스페이스 파일 이름 앞에는 취소가 붙습니다. 다음 쿼리는 InnoDB의 FILE_ID 및 FILE_NAME은 테이블 스페이스를 실행 취소합니다.

```
mysql> SELECT FILE_ID, FILE_NAME FROM INFORMATION_SCHEMA.FILES
       WHERE FILE_NAME LIKE '%undo%';
```

15.16 MySQL 성능 스키마와 InnoDB 통합

이 섹션에서는 InnoDB와 성능 스키마의 통합에 대해 간략하게 소개합니다. 포괄적인 성능 스키마 설명서는 27장, *MySQL 성능 스키마를* 참조하세요.

MySQL 성능 스키마 기능을 사용하여 특정 내부 InnoDB 작업을 프로파일링할 수 있습니다. 이러한 유형의 튜닝은 주로 성능 병목 현상을 극복하기 위한 최적화 전략을 평가하는 전문 사용자를 위한 것입니다. DBA는 용량 계획에 이 기능을 사용하여 일반적인 워크로드에서 특정 CPU, RAM 및 디스크 스토리지 조합으로 인해 성능 병목 현상이 발생하는지 확인하고, 발생한다면 시스템 일부의 용량을 늘려 성능을 개선할 수 있는지 판단할 수 있습니다.

이 기능을 사용하여 InnoDB 성능을 검사합니다:

- 일반적으로 성능 스키마 기능을 사용하는 방법을 잘 알고 있어야 합니다. 예를 들어, 계측기 및 소비자를 사용하도록 설정하는 방법과 performance_schema 테이블을 쿼리하여 데이터를 검색하는 방법을 알고 있어야 합니다. 소개 개요는 섹션 27.1, "성능 스키마 빠른 시작"을 참조하십시오.

- InnoDB에서 사용할 수 있는 성능 스키마 계측기에 대해 잘 알고 있어야 합니다. InnoDB 관련 계측기를 보려면 setup_instruments 테이블에서 'innodb'가 포함된 계측기 이름을 쿼리하면 됩니다.

```
mysql> SELECT *
       FROM performance_schema.setup_instruments
       WHERE NAME LIKE '%innodb%';
+---------------------------------------------------------+---------+ -------------+
| 이름|                                                    활성화 | 시간 |
+---------------------------------------------------------+---------+ -------------+
| 대기/동기화/뮤텍스/innodb/commit_cond_mutex               | 아니요    | 아니요    |
| 대기/동기화/뮤텍스/innodb/innobase_공유_뮤텍스            | 아니요    | 아니요    |
| 대기/동기화/뮤텍스/innodb/autoinc_mutex                  | 아니요    | 아니요    |
| 대기/동기화/뮤텍스/innodb/buf_pool_mutex                 | 아니요    | 아니요    |
| 대기/동기화/뮤텍스/innodb/buf_pool_zip_mutex             | 아니요    | 아니요    |
| 대기/동기화/뮤텍스/innodb/캐시_마지막_읽기_뮤텍스          | 아니요    | 아니요    |
| 대기/동기화/뮤텍스/innodb/dict_foreign_err_mutex         | 아니요    | 아니요    |
| 대기/동기화/뮤텍스/innodb/dict_sys_mutex                 | 아니요    | 아니요    |
| 대기/동기화/뮤텍스/innodb/recalc_pool_mutex              | 아니요    | 아니요    |
...
| wait/io/file/innodb/innodb_data_file                    | 예       | 예    |
| wait/io/file/innodb/innodb_log_file                     | 예       | 예    |
| wait/io/file/innodb/innodb_temp_file                    | 예       | 예    |
| STAGE/INNODB/ALTER TABLE (END)                          | 예       | 예    |
| 스테이지/INODB/테이블 변경(플러시)                        | 예       | 예    |
| STAGE/INNODB/ALTER TABLE(삽입)                          | 예       | 예    |
| 스테이지/인노드/테이블 변경(로그 적용 인덱스)              | 예       | 예    |
| 스테이지/인노드/변경 테이블(로그 적용 테이블)              | 예       | 예    |
| stage/innodb/alter table (병합 정렬)                    | 예       | 예    |
| 스테이지/인노드/알터 테이블(읽기 PK 및 내부 정렬)          | 예       | 예    |
| stage/innodb/버퍼 풀 로드                               | 예       | 예    |
```

```
| memory/innodb/buf_buf_pool                                | 아니요    | 아니요    |
| memory/innodb/dict_stats_bg_recalc_pool_t                 | 아니요    | 아니요    |
| memory/innodb/dict_stats_index_map_t                      | 아니요    | 아니요    |
| memory/innodb/dict_stats_n_diff_on_level                  | 아니요    | 아니요    |
| 메모리/이노드/기타                                        | 아니요    | 아니요    |
| memory/innodb/row_log_buf                                 | 아니요    | 아니요    |
| memory/innodb/row_merge_sort                              | 아니요    | 아니요    |
| memory/innodb/std                                         | 아니요    | 아니요    |
| 메모리/이노드/동기화/디버그_래치                          | 아니요    | 아니요    |
| memory/innodb/trx_sys_t::rw_trx_ids                       | 아니요    | 아니요    |
...
+-----------------------------------------------------------+---------+ -------------+
```

```
세트당 155행(0.00초)
```

계측된 InnoDB 개체에 대한 추가 정보를 보려면 계측된 개체에 대한 추가 정보를 제공하는 성능 스키마 인스턴스 테이블을 쿼리할 수 있습니다. InnoDB와 관련된 인스턴스 테이블은 다음과 같습니다:

- mutex_instances 테이블
- rwlock_instances 테이블
- cond_instances 테이블
- file_instances 테이블

참고

InnoDB 버퍼 풀과 관련된 뮤텍스 및 RW-락은 이 적용 범위에 포함되지 않으며, SHOW ENGINE INNODB MUTEX 명령의 출력에도 동일하게 적용됩니다.

예를 들어, 파일 I/O 계측을 실행할 때 성능 스키마에 표시되는 계측된 InnoDB 파일 개체에 대한 정보를 보려면 다음 쿼리를 실행할 수 있습니다:

```
mysql> SELECT *
       FROM performance_schema.file_instances
       WHERE EVENT_NAME LIKE '%innodb%'\G
*************************** 1. 행 ***************************
 FILE_NAME: /home/dtprice/mysql-8.2/data/ibdata1
EVENT_NAME: wait/io/file/innodb/innodb_data_file
OPEN_COUNT: 3
*************************** 2. 행 ***************************
 FILE_NAME: /home/dtprice/mysql-8.2/data/#ib_16384_0.dblwr
EVENT_NAME: wait/io/file/innodb/innodb_dblwr_file
OPEN_COUNT: 2
*************************** 3. 행 ***************************
 FILE_NAME: /home/dtprice/mysql-8.2/data/#ib_16384_1.dblwr
EVENT_NAME: wait/io/file/mysql-8.2/innodb_dblwr_file
OPEN_COUNT: 2
...
```

- InnoDB 이벤트 데이터를 저장하는 performance_schema 테이블에 대해 잘 알고 있어야 합니다. InnoDB 관련 이벤트와 관련된 테이블은 다음과 같습니다:
  - 대기 이벤트를 저장하는 대기 이벤트 테이블입니다.
  - 요약 테이블은 시간 경과에 따라 종료된 이벤트에 대한 집계된 정보를 제공합니다. 요약 테이블에는 I/O 작업에 대한 정보를 집계하는 파일 I/O 요약 테이블이 포함됩니다.
  - 스테이지 이벤트 테이블은 InnoDB ALTER TABLE 및 버퍼 풀 로드 작업에 대한 이벤트 데이터를 저장합니다. 자세한 내용은 15.16.1절, '성능 스키마를 사용하여 InnoDB 테이블의 ALTER TABLE 진행 상황 모니터링' 및 '성능 스키마를 사용하여 버퍼 풀 로드 진행 상황 모니터링'을 참조하십시오.

InnoDB 관련 객체에만 관심이 있는 경우, 이러한 테이블을 쿼리할 때 필요에 따라 WHERE EVENT_NAME LIKE '%innodb%' 또는 WHERE NAME LIKE '%innodb%' 절을 사용하세요.

15.16.1 성능 스키마를 사용하여 InnoDB 테이블에 대한 테이블 변경 진행 상황 모니터링

성능 스키마를 사용하여 InnoDB 테이블에 대한 테이블 변경 진행 상황을 모니터링할 수 있습니다.

ALTER TABLE의 여러 단계를 나타내는 7가지 단계 이벤트가 있습니다. 각 단계 이벤트는 전체 ALTER TABLE에 대한 작업 완료 및 작업 예상 총계를 보고합니다.

연산이 여러 단계로 진행됨에 따라 계산됩니다. WORK_ESTIMATED는 ALTER TABLE이 수행하는 모든 작업을 고려하는 수식을 사용하여 계산되며, ALTER TABLE 처리 중에 수정될 수 있습니다. WORK_COMPLETED 및 WORK_ESTIMATED 값은 ALTER TABLE이 수행한 모든 작업을 추상적으로 표현한 값입니다.

발생 순서대로 테이블 변경 단계 이벤트에는 다음이 포함됩니다:

- STAGE/INNODB/ALTER TABLE(PK 읽기 및 내부 정렬): 이 단계는 ALTER TABLE이 읽기-기본 키 단계에 있을 때 활성화됩니다. 이 단계는 기본 키의 예상 페이지 수로 설정된 WORK_COMPLETED=0 및 WORK_ESTIMATED로 시작합니다. 이 단계가 완료되면 WORK_ESTIMATED가 기본 키의 실제 페이지 수로 업데이트됩니다.
- STAGE/INNODB/ALTER TABLE(병합 정렬): 이 단계는 ALTER TABLE 연산으로 추가된 각 인덱스에 대해 반복됩니다.
- STAGE/INNODB/ALTER TABLE(삽입): 이 단계는 인덱스가 추가될 때마다 반복됩니다. 테이블 변경 작업을 수행합니다.
- STAGE/INNODB/ALTER TABLE(로그 적용 인덱스): 이 단계는 ALTER TABLE이 실행되는 동안 생성된 DML 로그를 적용하는 단계입니다.
- STAGE/INNODB/ALTER TABLE(플러시): 이 단계가 시작되기 전에, 플러시 목록의 길이에 따라 보다 정확한 예상값으로 WORK_ESTIMATED가 업데이트됩니다.
- STAGE/INNODB/ALTER TABLE(로그 적용 테이블): 이 단계에는 ALTER TABLE이 실행되는 동안 생성된 동시 DML 로그의 적용이 포함됩니다. 이 단계의 지속 시간은 은 테이블 변경 범위에 따라 다릅니다. 이 단계는 테이블에서 동시 DML이 실행되지 않은 경우 즉시 실행됩니다.
- STAGE/INNODB/ALTER TABLE (END): ALTER TABLE이실행되는 동안 테이블에서 실행된 DML을 다시 적용하는 등 플러시 단계 이후에 나타난 모든 남은 작업을 포함합니다.

참고

InnoDB ALTER TABLE 단계 이벤트는 현재 공간 인덱스 추가를 설명하지 않습니다.

성능 스키마를 사용한 테이블 모니터링 예제 변경

다음 예제에서는 테이블 변경 진행 상황을 모니터링하기 위해 stage/innodb/alter table% 스테이지 이벤트 계측기 및 관련 소비자 테이블을 활성화하는 방법을 설명합니다. 성능 스키마 스테이지 이벤트 계측기 및 관련 소비자에 대한 자세한 내용은 섹션 27.12.5, "성능 스키마 스테이지 이벤트 테이블"을 참조하십시오.

1. 스테이지/인노드/변경% 악기를 활성화합니다:

```
mysql> UPDATE performance_schema.setup_instruments
       SET ENABLED = 'YES'
       여기서 이름은 'stage/innodb/alter%'와 같습니다;
쿼리 확인, 영향을 받는 행 7개(0.00초) 행이 일치
했습니다: 7 변경됨: 7 경고: 0
```

2. 이벤트_스테이지_현재, 이벤트_스테이지_역사, 이벤트_스테이지_역사_길이를 포함하는 스테이지 이벤트 소비자 테이블을 사용하도록 설정합니다.

```
mysql> UPDATE performance_schema.setup_consumers
       SET ENABLED = 'YES'
       여기서 이름은 '%단계%'와 같습니다;
쿼리 확인, 영향을 받는 행 3개(0.00초) 행이 일치
했습니다: 3 변경됨: 3 경고: 0
```

3. ALTER TABLE 작업을 실행합니다. 이 예에서는 employees 샘플 데이터베이스의 employees 테이블에 middle_name 열을 추가합니다.

```
mysql> ALTER TABLE employees.employees ADD COLUMN middle_name varchar(14) AFTER first_name;
```

```
쿼리 완료, 영향을 받은 행 0개(9.27초) 레코드:
0 중복: 0 경고: 0
```

4. 성능 스키마 events_stages_current 테이블을 쿼리하여 ALTER TABLE 작업의 진행 상황을 확인합니다. 표시되는 단계 이벤트는 현재 진행 중인 ALTER TABLE 단계에 따라 다릅니다. WORK_COMPLETED 열에는 완료된 작업이 표시됩니다. WORK_ESTIMATED 열은 남은 작업의 예상치를 제공합니다.

```
mysql> SELECT EVENT_NAME, WORK_COMPLETED, WORK_ESTIMATED
       FROM performance_schema.events_stages_current;
+----------------------------------------------------+----------------+ ------------------------+
| 이벤트_이름                                          | 작업_완료 | 작업_추정 |
+----------------------------------------------------+----------------+ ------------------------+
| 스테이지/인노드/알터 테이블(읽기 PK 및 내부 정렬)                          |280             |1245 |
+----------------------------------------------------+----------------+ ------------------------+
한 세트에 1행(0.01초)
```

ALTER TABLE 작업이 완료된 경우 events_stages_current 테이블은 빈 집합을 반환합니다. 이 경우, 완료된 작업에 대한 이벤트 데이터를 보려면 events_stages_history 테이블을 확인하면 됩니다. 예를 들어

```
mysql> SELECT EVENT_NAME, WORK_COMPLETED, WORK_ESTIMATED
       FROM performance_schema.events_stages_history;
+----------------------------------------------------+----------------+ ------------------------+
| 이벤트_이름                                          | 작업_완료 | 작업_추정 |
+----------------------------------------------------+----------------+ ------------------------+
| 스테이지/인노드/알터 테이블(읽기 PK 및 내부 정렬)          |            886 |           1213 |
| 스테이지/INODB/테이블 변경(플러시)                    |           1213 |           1213 |
| 스테이지/인노드/변경 테이블(로그 적용 테이블)           |           1597 |           1597 |
| STAGE/INNODB/ALTER TABLE (END)                     |           1597 |           1597 |
| 스테이지/인노드/변경 테이블(로그 적용 테이블)           |           1981 |           1981 |
+----------------------------------------------------+----------------+ ------------------------+
한 세트에 5행(0.00초)
```

위와 같이 ALTER TABLE 처리 중에 WORK_ESTIMATED 값이 수정되었습니다. 초기 단계 완료 후 예상 작업량은 1213입니다. ALTER TABLE 처리가 완료되면 WORK_ESTIMATED는 실제 값인 1981로 설정됩니다.

15.16.2 성능 스키마를 사용한 InnoDB Mutex 대기 시간 모니터링

뮤텍스는 코드에서 특정 시간에 하나의 스레드만 공통 리소스에 액세스할 수 있도록 강제하기 위해 사용되는 동기화 메커니즘입니다. 서버에서 실행 중인 두 개 이상의 스레드가 동일한 리소스에 액세스해야 하는 경우 스레드는 서로 경쟁합니다. 뮤텍스에 대한 잠금을 획득한 첫 번째 스레드는 다른 스레드가 잠금이 해제될 때까지 기다리게 합니다.

계측되는 InnoDB 뮤텍스의 경우, 뮤텍스 대기는 성능 스키마를 사용하여 모니터링할 수 있습니다. 예를 들어, 성능 스키마 테이블에서 수집된 대기 이벤트 데이터는 대기 횟수가 가장 많은 뮤텍스 또는 총 대기 시간이 가장 긴 뮤텍스를 식별하는 데 도움이 될 수 있습니다.

다음 예제에서는 InnoDB 뮤텍스 대기 계측기를 활성화하는 방법, 연결된 소비자를 활성화하는 방법, 대기 이벤트 데이터를 쿼리하는 방법을 설명합니다.

1. 사용 가능한 `InnoDB` 뮤텍스 대기 계측기를 보려면 성능 스키마를 쿼리하세요.
   `설정_인스트루먼트` 테이블. 모든 `InnoDB` 뮤텍스 대기 계측기는 기본적으로 비활성화되어 있습니다.

```
mysql> SELECT *
       FROM performance_schema.setup_instruments
       WHERE NAME LIKE '%wait/synch/mutex/innodb%';
+-----------------------------------------------------------+---------+--------------+
| 이름                                                      | 활성화  | 시간         |
+-----------------------------------------------------------+---------+--------------+
| 대기/동기화/뮤텍스/innodb/commit_cond_mutex               | 아니요  | 아니요       |
| 대기/동기화/뮤텍스/innodb/innobase_공유_뮤텍스            | 아니요  | 아니요       |
| 대기/동기화/뮤텍스/innodb/autoinc_mutex                   | 아니요  | 아니요       |
| 대기/동기화/뮤텍스/innodb/autoinc_persisted_mutex         | 아니요  | 아니요       |
| 대기/동기화/뮤텍스/innodb/buf_pool_flush_state_mutex      | 아니요  | 아니요       |
| 대기/동기화/뮤텍스/innodb/buf_pool_LRU_list_mutex         | 아니요  | 아니요       |
```

```
| 대기/동기화/뮤텍스/innodb/buf_pool_free_list_mutex| NO| NO    |
| 대기/동기화/뮤텍스/innodb/buf_pool_zip_free_mutex| 아니요| 아니요          |
| 대기/동기화/뮤텍스/innodb/buf_pool_zip_해시_뮤텍스| 아니요| 아니요|
| 대기/동기화/뮤텍스/innodb/buf_pool_zip_mutex| 아니요| 아니요     |
| wait/synch/mutex/innodb/cache_last_read_mutex| NO| NO     |
| 대기/동기화/뮤텍스/인노드/딕트_외국_오류_뮤텍스| 아니요| 아니요        |
| 대기/동기화/mutex/innodb/dict_persist_dirty_tables_mutex | NO| NO    |
| 대기/동기화/뮤텍스/innodb/dict_sys_mutex| 아니요| 아니요          |
| 대기/동기화/뮤텍스/innodb/recalc_pool_mutex| 아니요| 아니요      |
| 대기/동기화/뮤텍스/인노드b/필_시스템뮤텍스| 아니요| 아니요            |
| 대기/동기화/뮤텍스/innodb/flush_list_mutex| 아니요| 아니요        |
| 대기/동기화/뮤텍스/innodb/fts_bg_threads_mutex| 아니요| 아니요   |
| 대기/동기화/뮤텍스/innodb/fts_delete_mutex| 아니요| 아니요        |
| 대기/동기화/뮤텍스/innodb/fts_optimize_mutex| 아니요| 아니요     |
| 대기/동기화/mutex/innodb/fts_doc_id_mutex| 아니요| 아니요       |
| wait/synch/mutex/innodb/log_flush_order_mutex| NO| NO     |
| 대기/동기화/뮤텍스/innodb/해시 테이블 뮤텍스| 아니요| 아니요          |
| wait/synch/mutex/innodb/ibuf_bitmap_mutex| NO| NO          |
| 대기/동기화/뮤텍스/innodb/ibuf_mutex| 아니요| 아니요             |
| 대기/동기화/뮤텍스/innodb/ibuf_pessimistic_insert_mutex| 아니요| 아니요    |
| 대기/동기화/뮤텍스/innodb/로그_sys_mutex| 아니요| 아니요           |
| 대기/동기화/뮤텍스/innodb/로그_sys_쓰기_뮤텍스| 아니요| 아니요       |
| wait/synch/mutex/innodb/mutex_list_mutex| NO| NO           |
| 대기/동기화/뮤텍스/인노드비/페이지_지퍼_스탯_퍼_인덱스_뮤텍스| 아니요| 아니요      |
| 대기/동기화/뮤텍스/인노드b/퍼지_sys_pq_mutex| 아니요| 아니요        |
| 대기/동기화/뮤텍스/innodb/recv_sys_mutex| 아니요| 아니요          |
| 대기/동기화/mutex/innodb/recv_writer_mutex| NO| NO           |
| 대기/동기화/뮤텍스/innodb/redo_rseg_mutex| 아니요| 아니요         |
| 대기/동기화/뮤텍스/innodb/noredo_rseg_mutex| NO| NO           |
| 대기/동기화/뮤텍스/innodb/rw_lock_list_mutex| 아니요| 아니요     |
| 대기/동기화/뮤텍스/innodb/rw_lock_mutex| 아니요| 아니요           |
| 대기/동기화/뮤텍스/innodb/srv_dict_tmpfile_mutex| 아니요| 아니요          |
| 대기/동기화/뮤텍스/innodb/srv_innodb_monitor_mutex| 아니요| 아니요         |
| 대기/동기화/뮤텍스/innodb/srv_misc_tmpfile_mutex| NO| NO      |
| 대기/동기화/뮤텍스/인노드b/srv_모니터_파일_뮤텍스| 아니요| 아니요     |
| 대기/동기화/뮤텍스/innodb/buf_dblwr_mutex| 아니요| 아니요         |
| 대기/동기화/뮤텍스/인노드비/트렉스_undo_뮤텍스| 아니요| 아니요        |
| 대기/동기화/뮤텍스/innodb/trx_pool_mutex| NO| NO             |
| wait/synch/mutex/innodb/trx_pool_manager_mutex| NO| NO    |
| 대기/동기화/뮤텍스/인노드b/srv_sys_mutex| 아니요| 아니요          |
| 대기/동기화/뮤텍스/innodb/잠금_뮤텍스| NO| NO                |
| 대기/동기화/뮤텍스/innodb/잠금_대기_뮤텍스| 아니요| 아니요          |
| 대기/동기화/뮤텍스/인노드/트렉스뮤텍스| 아니요| 아니요              |
| 대기/동기화/뮤텍스/인노드b/srv_쓰레드_뮤텍스| 아니요| 아니요         |
| 대기/동기화/뮤텍스/innodb/rtr_active_mutex| 아니요| 아니요       |
| 대기/동기화/뮤텍스/innodb/rtr_match_mutex| 아니요| 아니요         |
| 대기/동기화/뮤텍스/인노드/rtr_경로_뮤텍스| 아니요| 아니요           |
```

```
| 대기/동기화/뮤텍스/innodb/rtr_ssn_mutex| 아니요| 아니요          |
| 대기/동기화/뮤텍스/인노드/트렉스_시스템_뮤텍스| 아니요| 아니요        |
| wait/synch/mutex/innodb/zip_pad_mutex| NO| NO              |
| 대기/동기화/뮤텍스/innodb/마스터_키_ID_뮤텍스| 아니요| 아니요      |
+-----------------------------------------------------+---------+--------------+
```

2. 일부 InnoDB 뮤텍스 인스턴스는 서버 시작 시 생성되며, 서버 시작 시 연결된 계측기가 활성화된 경우에만 계측됩니다. 모든 InnoDB 뮤텍스 인스턴스가 계측되고 활성화되도록 하려면, MySQL 구성 파일에 다음 performance-schema-instrument 규칙을 추가하세요:

```
performance-schema-instrument='wait/synch/mutex/innodb/%=ON'
```

모든 InnoDB 뮤텍스에 대한 대기 이벤트 데이터가 필요하지 않은 경우, MySQL 구성 파일에 성능 스키마-인스트루먼트 규칙을 추가하여 특정 인스트루먼트를 비활성화할 수 있습니다.

예를 들어, 전체 텍스트 검색과 관련된 InnoDB 뮤텍스 대기 이벤트 도구를 비활성화하려면 다음 규칙을 추가합니다:

```
performance-schema-instrument='wait/synch/mutex/innodb/fts%=OFF'
```

참고

wait/synch/mutex/innodb/fts%와 같이 접두사가 긴 규칙이 wait/synch/ mutex/innodb/%와 같이 접두사가 짧은 규칙보다 우선합니다.

구성 파일에 성능 스키마-인스트루먼트 규칙을 추가한 후 서버를 다시 시작합니다. 전체 텍스트 검색과 관련된 뮤텍스를 제외한 모든 InnoDB 뮤텍스가 활성화됩니다. 확인하려면 setup_instruments 테이블을 쿼리합니다. 활성화한 인스트루먼트에 대해 ENABLED 및 TIMED 열이 YES로 설정되어 있어야 합니다

```
.

mysql> SELECT *
       FROM performance_schema.setup_instruments
       WHERE NAME LIKE '%wait/synch/mutex/innodb%';
+-----------------------------------------------------+---------+ --------------+
| 이름| 활성화 | 시간 |
+-----------------------------------------------------+---------+ --------------+
| 대기/동기화/뮤텍스/innodb/commit_cond_mutex | 예 | 예 |
| 대기/동기화/뮤텍스/innodb/innobase_공유_뮤텍스 | 예 | 예 |
| 대기/동기화/뮤텍스/innodb/autoinc_mutex | 예 | 예 |
...
| 대기/동기화/뮤텍스/innodb/마스터_키_ID_뮤텍스 | 예 | 예 |
+-----------------------------------------------------+---------+ --------------+
세트 49행(0.00초)
```

3. setup_consumers 테이블을 업데이트하여 대기 이벤트 소비자를 사용하도록 설정합니다. 대기 이벤트 소비자는 기본적으로 비활성화되어 있습니다.

```
mysql> UPDATE performance_schema.setup_consumers
       SET enabled = 'YES'
       WHERE 이름 같은 'events_waits%'; 쿼
리 확인, 영향을 받는 행 3개(0.00초) 행이 일치했
습니다: 3 변경됨: 3 경고: 0
```

설정_소비자 테이블을 쿼리하여 대기 이벤트 소비자가 활성화되었는지 확인할 수 있습니다. 이벤트를 기다리는 현재, 이벤트 대기 기록 및 이벤트 대기 기록이 긴 소비자를 활성화해야 합니다.

```
mysql> SELECT * FROM performance_schema.setup_consumers;
+--------------------------------+------------+
| 이름| 활성화 |
+--------------------------------+------------+
| 이벤트_스테이지_현재| 아니요 |
| 이벤트_스테이지_역사| 아니요 |
| events_stages_history_long| NO |
| events_statements_current| 예 |
| events_statements_history| 예 |
| events_statements_history_long| 아니요 |
| 이벤트_대기_현재| 예 |
| 이벤트_거래_현재| 예 |
| 이벤트_대기_기록| 예 |
| events_transactions_history| 예 |
| events_waits_history_long| 예 |
| 이벤트_거래_기록_길이 | 아니요 |
| 글로벌_인스트루먼트| 예 |
| 스레드_인스트루먼트| 예 |
| statements_digest| 예 |
+--------------------------------+------------+
한 세트에 15행(0.00초)
```

4. 계측기와 소비자가 활성화되면 모니터링하려는 워크로드를 실행합니다. 이 예제에서는 `mysqlslap` 로드 에뮬레이션 클라이언트를 사용하여 워크로드를 시뮬레이션합니다.

```
$> ./mysqlslap --auto-generate-sql --concurrency=100 --iterations=10
       --number-of-queries=1000 --number-char-cols=6 --number-int-cols=6;
```

5. 대기 이벤트 데이터를 쿼리합니다. 이 예에서 대기 이벤트 데이터는 `events_waits_summary_global_by_event_name` 테이블에서 쿼리되며, 이 테이블은 `events_waits_current`, `events_waits_history` 및 `events_waits_history_long` 테이블에 있는 데이터를 집계합니다. 데이터는 이벤트를 생성한 상품의 이름인 이벤트 이름(`EVENT_NAME`)을 기준으로 요약됩니다. 요약된 데이터에는 다음이 포함됩니다:

   - `COUNT_STAR`

     요약된 대기 이벤트의 수입니다.

   - `SUM_TIMER_WAIT`

     요약된 시간 제한 대기 이벤트의 총 대기 시간입니다.

   - `MIN_TIMER_WAIT`

     요약된 시간 제한 대기 이벤트의 최소 대기 시간입니다.

   - `AVG_TIMER_WAIT`

     요약된 시간 제한 대기 이벤트의 평균 대기 시간입니다.

   - `MAX_TIMER_WAIT`

     요약된 시간 제한 대기 이벤트의 최대 대기 시간입니다.

   다음 쿼리는 상품 이름(`EVENT_NAME`), 대기 이벤트 수(`COUNT_STAR`), 해당 상품에 대한 이벤트의 총 대기 시간(`SUM_TIMER_WAIT`)을 반환합니다. 대기 시간은 기본적으로 피코초(1조분의 1초) 단위로 측정되므로 대기 시간을 1000000000 로 나누어 밀리초 단위로 표시합니다. 데이터는 요약된 대기 이벤트 수(`COUNT_STAR`)에 따라 내림차순으로 표시됩니다. `ORDER BY` 절을 조정하여 총 대기 시간별로 데이터 순서를 지정할 수 있습니다.

```
mysql> SELECT EVENT_NAME, COUNT_STAR, SUM_TIMER_WAIT/1000000000 SUM_TIMER_WAIT_MS
       FROM performance_schema.events_waits_summary_global_by_vent_name
       WHERE SUM_TIMER_WAIT > 0 AND EVENT_NAME LIKE 'wait/synch/mutex/innodb/%'
       ORDER BY COUNT_STAR DESC;
+-----------------------------------------------------------+------------+--------------------------+
| 이벤트_이름                                                | 카운트_스타 | 합계_타이머_대기_ms |
+-----------------------------------------------------------+------------+--------------------------+
| wait/synch/mutex/innodb/trx_mutex                         |     201111 |                  23.4719 |
| wait/synch/mutex/innodb/fil_system_mutex                  |      62244 |                   9.6426 |
| wait/synch/mutex/innodb/redo_rseg_mutex                   |      48238 |                   3.1135 |
| wait/synch/mutex/innodb/log_sys_mutex                     |      46113 |                   2.0434 |
| wait/synch/mutex/innodb/trx_sys_mutex                     |      35134 |                1068.1588 |
| wait/synch/mutex/innodb/lock_mutex                        |      34872 |                1039.2589 |
| wait/synch/mutex/innodb/log_sys_write_mutex               |      17805 |                1526.0490 |
| wait/synch/mutex/innodb/dict_sys_mutex                    |      14912 |                1606.7348 |
| wait/synch/mutex/innodb/trx_undo_mutex                    |      10634 |                   1.1424 |
```

```
| wait/synch/mutex/innodb/rw_lock_list_mutex                |        8538 |             0.1960 |
| wait/synch/mutex/innodb/buf_pool_free_list_mutex          |        5961 |             0.6473 |
| wait/synch/mutex/innodb/trx_pool_mutex                    |        4885 |          8821.7496 |
| wait/synch/mutex/innodb/buf_pool_LRU_list_mutex           |        4364 |             0.2077 |
| wait/synch/mutex/innodb/innobase_공유_mutex               |        3212 |             0.2650 |
| wait/synch/mutex/innodb/flush_list_mutex                  |        3178 |             0.2349 |
| wait/synch/mutex/innodb/trx_pool_manager_mutex            |        2495 |             0.1310 |
| wait/synch/mutex/innodb/buf_pool_flush_state_mutex        |        1318 |             0.2161 |
| wait/synch/mutex/innodb/log_flush_order_mutex             |        1250 |             0.0893 |
| wait/synch/mutex/innodb/buf_dblwr_mutex                   |         951 |             0.0918 |
| wait/synch/mutex/innodb/recalc_pool_mutex                 |         670 |             0.0942 |
```

```
| wait/synch/mutex/innodb/dict_persist_dirty_tables_mutex   |         345 |             0.0414 |
| wait/synch/mutex/innodb/lock_wait_mutex                   |         303 |             0.1565 |
| 대기/동기화/뮤텍스/INNODB/오토인크_뮤텍스                 |         196 |             0.0213 |
| wait/synch/mutex/innodb/autoinc_persisted_mutex           |         196 |             0.0175 |
| wait/synch/mutex/innodb/purge_sys_pq_mutex                |         117 |             0.0308 |
| 대기/동기화/뮤텍스/INNOD/B/SRV_SYS_MUTEX                  |          94 |             0.0077 |
| wait/synch/mutex/innodb/ibuf_mutex                        |          22 |             0.0086 |
| wait/synch/mutex/innodb/recv_sys_mutex                    |          12 |             0.0008 |
| wait/synch/mutex/innodb/srv_innodb_monitor_mutex          |           4 |             0.0009 |
| wait/synch/mutex/innodb/recv_writer_mutex                 |1 |0       .0005 |
+-----------------------------------------------------------+-------------+ ---------------------------+
```

참고

앞의 결과 집합에는 시작 프로세스 중에 생성된 대기 이벤트 데이터가 포함되어 있습니다. 이 데이터를 제외하려면, 앞의 결과 집합에서 `events_waits_summary_global_by_event_name` 테이블을 시작 직후와 워크로드를 실행하기 전에 잘라내야 합니다. 그러나 잘라내기 작업 자체는 무시할 수 있는 양의 대기 이벤트 데이터를 생성할 수 있습니다.

```
mysql> TRUNCATE performance_schema.events_waits_summary_global_by_vent_name;
```

15.17 InnoDB 모니터

`InnoDB` 모니터는 `InnoDB` 내부 상태에 대한 정보를 제공합니다. 이 정보는 성능 튜닝에 유용합니다.

15.17.1 InnoDB 모니터 유형

`InnoDB` 모니터에는 두 가지 유형이 있습니다:

- 표준 `InnoDB` 모니터는 다음 유형의 정보를 표시합니다:
  - 메인 백그라운드 스레드에서 수행되는 작업
  - 세마포어 대기
  - 가장 최근의 외래 키 및 교착 상태 오류에 대한 데이터
  - 트랜잭션 대기 잠금
  - 활성 트랜잭션이 보유한 테이블 및 레코드 잠금
  - 보류 중인 I/O 작업 및 관련 통계
  - 버퍼 및 적응형 해시 인덱스 통계 삽입
  - 로그 데이터 다시 실행
  - 버퍼 풀 통계
  - 행 작업 데이터

- `InnoDB` 잠금 모니터는 표준 `InnoDB` 모니터 출력의 일부로 추가 잠금 정보를 인쇄합니다.

15.17.2 InnoDB 모니터 활성화

`InnoDB` 모니터가 주기적 출력을 사용하도록 설정된 경우, `InnoDB`는 약 15초마다 출력을 `mysqld` 서버 표준 오류 출력(`stderr`)에 기록합니다.

InnoDB는 잠재적인 버퍼 오버플로우를 방지하기 위해 모니터 출력을 `stdout` 또는 고정 크기 메모리 버퍼가 아닌 `stderr`로 전송합니다.

Windows에서는 달리 구성하지 않는 한 stderr이기본 로그 파일로 이동합니다. 출력을 오류 로그가 아닌 콘솔 창으로 보내려면 콘솔 창에 있는 명령 프롬프트에서 `--console` 옵션을 사용하여 서버를 시작합니다. 자세한 내용은 Windows의 기본 오류 로그 대상을 참조하세요.

Unix 및 Unix와 유사한 시스템에서는 달리 구성하지 않는 한 일반적으로 stderr이터미널로 전달됩니다. 자세한 내용은 유닉스 및 유닉스와 유사한 시스템의 기본 오류 로그 대상을 참조하세요.

출력 생성으로 인해 약간의 성능 저하가 발생하므로 실제로 모니터 정보를 확인하려는 경우에만 `InnoDB` 모니터를 사용하도록 설정해야 합니다. 또한 모니터 출력이 오류 로그로 향하는 경우 나중에 모니터를 비활성화하는 것을 잊어버리는 경우 로그가 상당히 커질 수 있습니다.

참고

문제 해결을 지원하기 위해 `InnoDB`는 특정 조건에서 표준 `InnoDB` 모니터 출력을 일시적으로 활성화합니다. 자세한 내용은 섹션 15.21, "InnoDB 문제 해결"을 참조하십시오.

`InnoDB` 모니터 출력은 타임스탬프와 모니터 이름이 포함된 헤더로 시작됩니다. 예를 들어

```
=====================================
2014-10-16 18:37:29 0x7fc2a95c1700 INNODB 모니터 출력
=====================================
```

잠금 모니터는 추가 잠금 정보를 추가하여 동일한 출력을 생성하기 때문에 표준 `InnoDB` 모니터의 헤더(`INNODB 모니터 출력`)는 잠금 모니터에도 사용됩니다.

`innodb_status_output` 및 `innodb_status_output_locks` 시스템 변수는 표준 `InnoDB` 모니터 및 `InnoDB` 잠금 모니터를 활성화하는 데 사용됩니다.

`InnoDB` 모니터를 활성화 또는 비활성화하려면 `PROCESS` 권한이 필요합니다.

표준 InnoDB 모니터 활성화

`innodb_status_output` 시스템 변수를 `ON`으로 설정하여 표준 `InnoDB` 모니터를 활성화합니다.

```
설정 글로벌 innodb_status_output=ON;
```

표준 `InnoDB` 모니터를 비활성화하려면 `innodb_status_output`을 `OFF`로 설정합니다.

서버를 종료하면 `innodb_status_output` 변수가 기본값인 `OFF`로 설정됩니다.
값입니다.

InnoDB 잠금 모니터 활성화

`InnoDB` 잠금 모니터 데이터는 `InnoDB` 표준 모니터 출력과 함께 인쇄됩니다. `InnoDB` 잠금 모니터 데이터를 주기적으로 인쇄하려면 `InnoDB` 표준 모니터와 `InnoDB` 잠금 모니터를 모두 사용하도록 설정해야 합니다.

InnoDB 잠금 모니터를 활성화하려면 innodb_status_output_locks 시스템 변수를 ON으로 설정합니다. InnoDB 잠금 모니터 데이터를 주기적으로 출력하려면 InnoDB 표준 모니터와 InnoDB 잠금 모니터를 모두 활성화해야 합니다:

```
설정 글로벌 innodb_status_output=ON;
GLOBAL innodb_status_output_locks=ON으로 설정합니다;
```

InnoDB 잠금 모니터를 비활성화하려면 innodb_status_output_locks를 OFF로 설정합니다. 설정 innodb_status_output을 OFF로 설정하여 InnoDB 표준 모니터도 비활성화합니다.

서버를 종료하면 innodb_status_output 및 innodb_status_output_locks 변수는 기본값인 OFF로 설정됩니다.

참고

엔진 INNODB 상태 표시를 위한 InnoDB 잠금 모니터를 활성화하려면 다음과 같이 하세요. 출력을 사용하도록 설정한 경우에는 innodb_status_output_locks만 활성화하면 됩니다.

주문형 표준 InnoDB 모니터 출력 얻기

주기적인 출력을 위해 표준 InnoDB 모니터를 활성화하는 대신, 클라이언트 프로그램으로 출력을 가져오는 SHOW ENGINE INNODB STATUS SQL 문을 사용하여 필요에 따라 표준 InnoDB 모니터 출력을 얻을 수 있습니다. mysql 대화형 클라이언트를 사용하는 경우 일반적인 세미콜론 문 종결자를 \G로 바꾸면 출력을 더 쉽게 읽을 수 있습니다:

```
mysql> SHOW ENGINE INNODB STATUS\G
```

InnoDB 잠금 모니터가 활성화된 경우 엔진 INNODB 상태 표시 출력에 InnoDB 잠금 모니터 데이터도 포함됩니다.

표준 InnoDB 모니터 출력을 상태 파일로 지시하기

표준 InnoDB 모니터 출력을 활성화하고 시작 시 -- innodb-status-file 옵션을 지정하여 상태 파일로 보낼 수 있습니다. 이 옵션을 사용하면 InnoDB는 데이터 디렉터리에 innodb_status.*pid*라는 파일을 생성하고 약 15초마다 출력을 이 파일에 씁니다.

InnoDB는 서버가 정상 종료되면 상태 파일을 제거합니다. 비정상 종료가 발생하면 상태 파일을 수동으로 제거해야 할 수 있습니다.

innodb-status-file 옵션은 출력 생성이 성능에 영향을 미칠 수 있고 시간이 지남에 따라 innodb_status.*pid* 파일이 상당히 커질 수 있으므로 일시적으로 사용하기 위한 것입니다.

15.17.3 InnoDB 표준 모니터 및 잠금 모니터 출력

잠금 모니터는 추가 잠금 정보를 포함한다는 점을 제외하면 표준 모니터와 동일합니다. 주기적 출력을 위해 두 모니터 중 하나를 활성화하면 동일한 출력 스트림이 켜지지만 잠금 모니터가 활성화된 경우 스트림에 추가 정보가 포함됩니다. 예를 들어 표준 모니터와 잠금 모니터를 활성화하면 단일 출력 스트림이 켜집니다. 잠금 모니터를 비활성화할 때까지 스트림에는 추가 잠금 정보가 포함됩니다.

표준 모니터 출력은 쇼 엔진 이노드 상태를 사용하여 제작할 경우 1MB로 제한됩니다.
문을 사용할 수 없습니다. 이 제한은 서버 표준 오류 출력(stderr)에 기록된 출력에는 적용되지 않습니다. 표

준 모니터 출력 예시:

```
mysql> SHOW ENGINE INNODB STATUS\G
*************************** 1. 행 *************************** 입력합
  니다: InnoDB
  이름:
상태:
=====================================
2018-04-12 15:14:08 0x7f971c063700 INNODB 모니터 출력
```

```
=====================================
지난 4초 동안 계산된 초당 평균값입니다.




---------------------
백그라운드 스레드
---------------------
srv_master_thread 루프: 15 srv_active, 0 srv_shutdown, 1122 srv_idle
srv_master_thread 로그 플러시 및 쓰기: 0
------------
SEMAPHORES
------------
OS 대기열 정보: 예약 카운트 24 OS 대기열 정보:
신호 카운트 24
RW 공유 회전 4회, 라운드 8회, OS 대기 4회
RW-excl 회전 2, 60회전, OS 대기 2
RW-sx 회전 0, 라운드 0, OS 대기 0
대기당 스핀 라운드: 2.00 RW 공유, 30.00 RW-excl, 0.00 RW-sx
-----------------------------
최신 외래 키 오류
```

```
------------------------------
2018-04-12 14:57:24 0x7f97a9c91700 트랜잭션:
트랜잭션 7717, 활성 0초 사용 중인 mysql 테이블
삽입 중 1, 잠김 1
4 잠금 구조체, 힙 크기 1136, 3 행 잠금, 로그 항목 실행 취소 3
MySQL 스레드 id 8, OS 스레드 핸들 140289365317376, 쿼리 id 14 localhost root update INSERT INTO
child VALUES (NULL, 1), (NULL, 2), (NULL, 3), (NULL, 4), (NULL, 5), (NULL, 6)
test`.`child` 테이블에 대한 외래 키 제약 조건이 실패했습니다:
,
  CONSTRAINT `child_ibfk_1` FOREIGN KEY (`parent_id`) REFERENCES `parent` (`id`) ON DELETE
  CASCADE ON UPDATE CASCADE
인덱스 par_ind 튜플에서 하위 테이블에 추가하려고 합니다: 데이터
튜플: 필드 2개;
 0: LEN 4; HEX 80000003; ASC     ;;
 1: LEN 4; HEX 80000003; ASC     ;;

그러나 상위 테이블 `test`.`parent`의 PRIMARY 인덱스에서 가장
가까운 일치 항목을 찾을 수 있는 것은 레코드입니다:
물리적 기록: n_fields 3; 컴팩트 포맷; 정보 비트 0 0: 렌 4; 헥스
 80000004; asc                    ;;
 1: LEN 6; HEX 000000001E19; ASC       ;;
 2: LEN 7; HEX 81000001110137; ASC       7;;
---------------
트랜잭션
---------------
Trx ID 카운터 7748
trx의 n:o < 7747에 대한 퍼지 완료 n:o < 0 상태: 실행 중이지만 유휴 상태 히스토리
목록 길이 19
각 세션의 트랜잭션 목록입니다:
---트랜잭션 421764459790000, 시작되지 않음
0 잠금 구조체, 힙 크기 1136, 0 행 잠금
---트랜잭션 7747, ACTIVE 23초 인덱스 시작 23초 사용 중인
mysql 테이블 읽기 1, 잠김 1
잠금 대기 2 잠금 구조체, 힙 크기 1136, 1 행 잠금
MySQL 스레드 id 9, OS 스레드 핸들 140286987249408, 쿼리 id 51 로컬 호스트 루트 업데이트 DELETE FROM t
WHERE i = 1
------- trx 이 잠금이 부여되기를 23초 동안 기다렸습니다:
레코드 록 스페이스 id 4 페이지 번호 4 n 비트 72 인덱스 `test`.`t` 테이블의 GEN_CLUST_INDEX trx id
7747 lock_mode X 대기 중
레코드 잠금, 힙 번호 3 물리적 레코드: n_fields 4; 컴팩트 포맷; 정보 비트 0 0: 렌 6; 헥스
 000000000202; asc                     ;;
 1: LEN 6; HEX 000000001E41; ASC      A;;
 2: LEN 7; HEX 820000008B0110; ASC        ;;
 3: LEN 4; HEX 80000001; ASC     ;;

TABLE LOCK 테이블 `test`.`t` trx id 7747 잠금 모드 IX
레코드 록 스페이스 id 4 페이지 번호 4 n 비트 72 인덱스 `test`.`t` 테이블의 GEN_CLUST_INDEX trx id
7747 lock_mode X 대기 중
레코드 잠금, 힙 번호 3 물리적 레코드: n_fields 4; 컴팩트 포맷; 정보 비트 0 0: 렌 6; 헥스
 000000000202; asc                     ;;
 1: LEN 6; HEX 000000001E41; ASC      A;;
 2: LEN 7; HEX 820000008B0110; ASC        ;;
 3: LEN 4; HEX 80000001; ASC     ;;
----------
파일 입출
력
----------
```

```
I/O 스레드 0 상태: I/O 요청 대기 중(버퍼 스레드 삽입) I/O 스레드 1 상태: I/O
요청 대기 중(로그 스레드)
I/O 스레드 2 상태: I/O 요청 대기 중 (읽기 스레드) I/O 스레드 3 상태
: I/O 요청 대기 중 (읽기 스레드) I/O 스레드 4 상태: I/O 요청 대기
중 (읽기 스레드) I/O 스레드 5 상태: I/O 요청 대기 중 (읽기 스레드)
I/O 스레드 6 상태: I/O 요청 대기 중 (쓰기 스레드) I/O 스레드 7 상태:
I/O 요청 대기 중 (쓰기 스레드) I/O 스레드 8 상태: I/O 요청 대기 중
(쓰기 스레드) I/O 스레드 9 상태: I/O 요청 대기 중 (쓰기 스레드)
일반 aio 읽기 보류 중입니다: [0, 0, 0, 0] , aio 쓰기: [0, 0, 0, 0] , ibuf
 aio 읽기:, 로그 I/O:, 동기화 I/O:
보류 중인 플러시(fsync) 로그: 0; 버퍼 풀: 0
```

```
833개의 OS 파일 읽기, 605개의 OS 파일 쓰기, 208개의 OS fsync
0.00 읽기/초, 0 평균 바이트/읽기, 0.00 쓰기/초, 0.00 fsyncs/초

-------------------------------------------------
삽입 버퍼 및 적응형 해시 인덱스

Ibuf: 크기 1, 자유 목록 길이 0, 세그 크기 2, 0 병합된 작업을 병
합합니다:
 삽입 0, 마크 0 삭제, 삭제 0 폐기된 작업:
 삽입 0, 마크 삭제 0, 삭제 0
해시 테이블 크기 553253, 노드 힙에 버퍼가 0개 있음 해시 테
이블 크기 553253, 노드 힙에 버퍼가 1개 있음 해시 테이블 크
기 553253, 노드 힙에 버퍼가 3개 있음 해시 테이블 크기
553253, 노드 힙에 버퍼가 0개 있음 해시 테이블 크기
553253, 노드 힙에 0 버퍼 있음 해시 테이블 크기 553253,
노드 힙에 0 버퍼 있음 해시 테이블 크기 553253, 노드 힙에
0 버퍼 있음 해시 테이블 크기 553253, 노드 힙에 0 버퍼 있음
해시 테이블 크기 553253, 노드 힙에 0 버퍼 있음
0.00 해시 검색/초, 0.00 비해시 검색/초
---
LOG
---
로그 시퀀스 번호 19643450 로그 버퍼 할당 최
대 19643450 로그 버퍼 완료 최대 19643450
--------------------------------
로그 기록 최대                    19643450
로그가 최대                       19643450
까지 플러시됨                     더티  페이
지가 최대 19643450까지 추가됨 페이지가 최대
19643450까지 플러시됨
마지막 체크포인트                 19643450
129 로그 I/O 완료, 0.00 로그 I/O/초

버퍼 풀 및 메모리

할당된 총 대용량 메모리 2198863872 할당된 사전
메모리 409606 버퍼 풀         크기131072
------------------------------
여유 버퍼             130095
데이터베이스 페이지------973
이전 데이터베이스 페이지
0 수정된 데이터베이스 페
이지                  0
보류 중인 읽기        0
쓰기 보류 중입니다: LRU 0, 플러시 목록 0, 단일 페이지 0 페이지가
영 0, 영이 아님 0
0.00 영/초, 0.00 비영/초
페이지 읽기 810, 생성 163, 쓰기 404
0.00 읽기/초, 0.00 생성/초, 0.00 쓰기/초
버퍼 풀 적중률 1000 / 1000, 영 만들기 비율 0 / 1000이 아닌 0 / 1000
페이지 앞 읽기 0.00초, 액세스 없이 퇴거 0.00초, 무작위 앞 읽기 0.00초 LRU len: 973,
unzip_LRU len: 0
I/O 합계[0]:cur[0], 압축 해제 합계[0]:cur[0]
```

개별 버퍼 풀 정보

---버퍼 풀 0

버퍼 풀 크기

65536 사용 가능한 버퍼 크기

65043

데이터베이스 페이지 491

이전 데이터베이스 페이지

0 수정된 데이터베이스 페

이지 0

보류 중인 읽기 0

쓰기 보류 중입니다: LRU 0, 플러시 목록 0, 단일 페이지 0 페이지가

영 0, 영이 아닌 0으로 만들어짐

0.00 영/초, 0.00 비영/초

페이지 읽기 411, 생성 80, 쓰기 210

0.00 읽기/초, 0.00 생성/초, 0.00 쓰기/초

버퍼 풀 적중률 1000 / 1000, 영 만들기 비율 0 / 1000이 아닌 0 / 1000

페이지 앞 읽기 0.00초, 액세스 없이 퇴거 0.00초, 무작위 앞 읽기 0.00초 LRU len: 491,

unzip_LRU len: 0

I/O 합계[0]:cur[0], 압축 해제 합계[0]:cur[0]

---버퍼 풀 1

버퍼 풀 크기 65536

```
여유 버퍼             65052
데이터베이스 페이지     482
이전 데이터베이스 페이지
0 수정된 데이터베이스 페
이지                  0
보류 중인 읽기        0
쓰기 보류 중입니다: LRU 0, 플러시 목록 0, 단일 페이지 0 페이지가
영 0, 영이 아님 0
0.00 영/초, 0.00 비영/초
페이지 읽기 399, 생성 83, 작성 194
0.00 읽기/초, 0.00 생성/초, 0.00 쓰기/초
마지막 인쇄 이후 버퍼 풀 페이지를 가져올 수 없습니다.
페이지 앞 읽기 0.00초, 액세스 없이 퇴거 0.00초, 무작위 앞 읽기 0.00초 LRU len: 482,
unzip_LRU len: 0
I/O 합계[0]:cur[0], 압축 해제 합계[0]:cur[0]
행 작업
----------
InnoDB 내부 쿼리 0개, 대기열에 있는 쿼리 0개
InnoDB 내에서 열린 읽기 뷰 0개
프로세스 ID=5772, 메인 스레드 ID=140286437054208 , 상태=잠자기 행 수 삽입
57, 업데이트 354, 삭제 4, 읽기 4421
0.00 삽입/초, 0.00 업데이트/초, 0.00 삭제/초, 0.00 읽기/초
INODB 모니터 출력 종료
============================
```

표준 모니터 출력 섹션

표준 모니터에서 보고하는 각 메트릭에 대한 설명은 MySQL 데이터베이스용 Oracle Enterprise Manager 사용자 가이드의 메트릭 장을 참조하십시오.

- 상태

  이 섹션에는 타임스탬프, 모니터 이름 및 초당 평균의 기준이 되는 초 수가 표시됩니다. 초 수는 현재 시간과 InnoDB 모니터 출력이 마지막으로 인쇄된 시간 사이의 경과 시간입니다.

- 백그라운드 스레드

  srv_master_thread 줄에는 주 백그라운드 스레드에서 수행한 작업이 표시됩니다.

- SEMAPHORES

  이 섹션에서는 세마포어를 대기 중인 스레드와 스레드가 뮤텍스 또는 rw-lock 세마포어에서 스핀 또는 대기를 필요로 한 횟수에 대한 통계를 보고합니다. 많은 수의 스레드가 세마포어를 대기하는 것은 디스크 I/O 또는 InnoDB 내부의 경합 문제의 결과일 수 있습니다. 경합은 쿼리의 병렬 처리량이 많거나 운영 체제 스레드 스케줄링에 문제가 있기 때문일 수 있습니다. 이러한 상황에서는 innodb_thread_concurrency 시스템 변수를 기본값보다 작게 설정하면 도움이 될 수 있습니다. 대기 라인당 스핀 라운드 수는 뮤텍스에 대한 OS 대기당 스핀록 라운드 수를 보여줍니다.

  뮤텍스 메트릭은 쇼 엔진 INNODB 뮤텍스에서 보고합니다.

- 최신 외래 키 오류

  이 섹션에서는 가장 최근에 발생한 외래 키 제약 조건 오류에 대한 정보를 제공합니다. 해당 오류가 발생하지 않은 경우에는 표시되지 않습니다. 내용에는 실패한 문과 실패한 제약 조건에 대한 정보 및 참조 및 참조 테이블이 포함됩니다.

- 최근 감지된 교착 상태

  이 섹션에서는 가장 최근 교착 상태에 대한 정보를 제공합니다. 교착 상태가 발생하지 않은 경우에는 표시되지 않습니다. 내용에는 관련된 트랜잭션, 각 트랜잭션이 시도한 문이 표시됩니다.
  실행할 트랜잭션, 해당 트랜잭션에 필요한 잠금, 교착 상태를 풀기 위해 `InnoDB`가 롤백하기로 결정한 트랜잭션에 대한 정보를 제공합니다. 이 섹션에서 보고되는 잠금 모드는 섹션 15.7.1, "InnoDB 잠금"에 설명되어 있습니다.

- 트랜잭션

  이 섹션에서 잠금 대기가 보고되면 애플리케이션에 잠금 경합이 있을 수 있습니다. 이 출력은 트랜잭션 교착 상태의 원인을 추적하는 데 도움이 될 수도 있습니다.

- 파일 입출력

  이 섹션에서는 InnoDB가 다양한 유형의 I/O를 수행하는 데 사용하는 스레드에 대한 정보를 제공합니다. O. 이 중 처음 몇 개는 일반적인 InnoDB 처리 전용입니다. 또한 보류 중인 I/O 작업에 대한 정보와 I/O 성능에 대한 통계도 표시됩니다.

  이러한 스레드 수는 innodb_read_io_threads 및 innodb_write_io_threads 매개 변수에 의해 제어됩니다. 섹션 15.14, "InnoDB 시작 옵션 및 시스템 변수"를 참조하십시오.

- 삽입 버퍼 및 적응형 해시 인덱스

  이 섹션에서는 InnoDB 삽입 버퍼(변경 버퍼라고도 함) 및 적응형 해시 인덱스의 상태를 표시합니다.

  관련 정보는 섹션 15.5.2, "버퍼 변경" 및 섹션 15.5.3, "적응형 해시 인덱스"를 참조하세요.

- LOG

  이 섹션에는 InnoDB 로그에 대한 정보가 표시됩니다. 여기에는 현재 로그 시퀀스 번호, 로그가 디스크에 얼마나 많이 플러시되었는지, InnoDB가 마지막으로 체크포인트를 수행한 위치가 포함됩니다. (섹션 15.11.3, "InnoDB 체크포인트" 참조) 이 섹션에는 보류 중인 쓰기 및 쓰기 성능 통계에 대한 정보도 표시됩니다.

- 버퍼 풀 및 메모리

  이 섹션에서는 읽고 쓴 페이지에 대한 통계를 제공합니다. 이 숫자를 통해 현재 쿼리가 수행 중인 데이터 파일 I/O 작업의 수를 계산할 수 있습니다.

  버퍼 풀 통계에 대한 설명은 InnoDB 표준 모니터를 사용하여 버퍼 풀 모니터링하기를 참조하세요. 버퍼 풀 작동에 대한 자세한 내용은 섹션 15.5.1, "버퍼 풀"을 참조하십시오.

- 행 작업

  이 섹션에서는 각 행 작업 유형에 대한 행 작업 수 및 성능 속도를 포함하여 메인 스레드가 수행하는 작업을 보여줍니다.

15.18 InnoDB 백업 및 복구

이 섹션에서는 InnoDB 백업 및 복구와 관련된 주제를 다룹니다.

- InnoDB에 적용할 수 있는 백업 기술에 대한 자세한 내용은 섹션 15.18.1, "InnoDB 백업"을 참조하

세요.

- 특정 시점 복구, 디스크 장애 또는 손상으로부터의 복구 및 방법에 대한 자세한 내용은 다음을 참조하세요. InnoDB는 크래시 복구를 수행합니다(15.18.2절. "InnoDB 복구"를 참조하세요).

15.18.1 InnoDB 백업

안전한 데이터베이스 관리의 핵심은 정기적인 백업입니다. 데이터 볼륨, *MySQL* 서버 수, 데이터베이스 워크로드에 따라 다음과 같은 백업 기술을 단독으로 또는 조합하여 사용할 수 있습니다. *MySQL Enterprise Backup*을 사용한 핫 백업, 파일을 복사하는 콜드 백업

MySQL 서버를 종료하고, 데이터 볼륨이 작거나

스키마 객체의 구조. 핫 백업 및 콜드 백업은 실제 데이터 파일을 복사하는 물리적 백업으로, `mysqld` 서버에서 직접 사용하여 더 빠르게 복원할 수 있습니다.

MySQL 엔터프라이즈 백업을 사용하는 것이 `InnoDB` 데이터 백업에 권장되는 방법입니다.

참고

InnoDB는 타사 백업 도구를 사용하여 복원된 데이터베이스를 지원하지 않습니다.

핫 백업

MySQL 엔터프라이즈 백업 구성 요소의 일부인 `mysqlbackup` 명령을 사용하면 데이터베이스의 일관된 스냅샷을 생성하면서 운영 중단을 최소화하여 `InnoDB` 테이블을 포함하여 실행 중인 MySQL 인스턴스를 백업할 수 있습니다. `mysqlbackup`이 `InnoDB`를 복사하는 경우 테이블에대한 읽기 및 쓰기를 계속할 수 있습니다. MySQL 엔터프라이즈 백업은 압축된 백업 파일을 생성하고 테이블 및 데이터베이스의 하위 집합을 백업할 수도 있습니다. MySQL 바이너리 로그와 함께 사용자는 특정 시점 복구를 수행할 수 있습니다. MySQL Enterprise 백업은 MySQL Enterprise 구독의 일부입니다. 자세한 내용은 섹션 30.2, 'MySQL Enterprise 백업 개요'를 참조하세요.

콜드 백업

MySQL 서버를 종료할 수 있는 경우 `InnoDB`에서 테이블을 관리하는 데 사용하는 모든 파일로 구성된 물리적 백업을 만들 수 있습니다. 다음 절차를 따르세요:

1. MySQL 서버를 느리게 종료하고 오류 없이 중지되는지 확인합니다.
2. 모든 `InnoDB` 데이터 파일(`ibdata` 파일 및 `.ibd` 파일)을 안전한 장소에 복사합니다.
3. 모든 `InnoDB` 재실행 로그 파일(*`#ib_redoN`* 파일)을 안전한 곳에 복사합니다.
4. 내 `.cnf` 구성 파일을 안전한 곳에 복사합니다.

mysqldump를 사용한 논리적 백업

물리적 백업 외에도 `mysqldump`를 사용하여 테이블을 덤프하여 논리적 백업을 정기적으로 생성하는 것이 좋습니다. 바이너리 파일은 사용자가 인지하지 못하는 사이에 손상될 수 있습니다. 덤프된 테이블은 사람이 읽을 수 있는 텍스트 파일로 저장되므로 테이블 손상을 더 쉽게 발견할 수 있습니다. 또한 형식이 더 간단하기 때문에 심각한 데이터 손상이 발생할 가능성도 적습니다. `mysqldump`에는 다른 클라이언트를 잠그지 않고 일관된 스냅샷을 만들 수 있는 `--single-transaction` 옵션도 있습니다. 섹션 7.3.1, "백업 정책 설정"을 참조하세요.

복제는 `InnoDB` 테이블과 함께 작동하므로 MySQL 복제 기능을 사용하여 고가용성이 필요한 데이터베이스 사이트에 데이터베이스 사본을 보관할 수 있습니다. 섹션 15.19, "InnoDB 및 MySQL 복제"를 참조하세요.

15.18.2 InnoDB 복구

이 섹션에서는 `InnoDB` 복구에 대해 설명합니다. 주제는 다음과 같습니다:

- 특정 시점 복구
- 데이터 손상 또는 디스크 장애 복구
- InnoDB 충돌 복구
- 충돌 복구 중 테이블 공간 검색

특정 시점 복구

물리적 백업이 수행된 시점부터 현재까지의 `InnoDB` 데이터베이스를 복구하려면 백업을 수행하기 전이라도 바이너리 로깅을 활성화한 상태로 MySQL 서버를 실행해야 합니다. To

백업을 복구한 후 특정 시점 복구를 수행하면 백업이 이루어진 후 발생한 바이너리 로그의 변경 사항을 적용할 수 있습니다. 7.5절. "특정 시점(증분) 복구"를 참조하세요.

데이터 손상 또는 디스크 장애 복구

데이터베이스가 손상되거나 디스크 장애가 발생하면 백업을 사용하여 복구를 수행해야 합니다. 손상된 경우 먼저 손상되지 않은 백업을 찾습니다. 기본 백업을 복구한 후 `mysqlbinlog` 및 `mysql`을 사용하여 바이너리 로그 파일에서 특정 시점 복구를 수행하여 백업이 이루어진 후 발생한 변경 사항을 복원합니다.

데이터베이스가 손상된 경우에는 하나 또는 몇 개의 손상된 테이블을 덤프, 삭제하고 다시 생성하는 것으로 충분합니다. 테이블이 손상되었는지 확인하기 위해 `CHECK TABLE` 문을 사용할 수 있지만, `CHECK TABLE`은 당연히 가능한 모든 종류의 손상을 감지할 수는 없습니다.

경우에 따라 명백한 데이터베이스 페이지 손상은 실제로 운영 체제가 자체 파일 캐시를 손상시켰기 때문이며 디스크의 데이터는 괜찮을 수 있습니다. 먼저 컴퓨터를 재시작해 보는 것이 가장 좋습니다.
이렇게 하면 데이터베이스 페이지 손상으로 보이는 오류가 제거될 수 있습니다. `InnoDB` 일관성 문제로 인해 MySQL을 시작하는 데 여전히 문제가 있는 경우, 데이터를 덤프할 수 있는 복구 모드에서 인스턴스를 시작하는 단계는 15.21.3절. 'InnoDB 복구 강제 실행'을 참조하세요.

InnoDB 충돌 복구

예기치 않은 MySQL 서버 종료에서 복구하려면 MySQL 서버를 다시 시작하기만 하면 됩니다. `InnoDB`는 자동으로 로그를 확인하고 데이터베이스를 현재로 롤포워드합니다. `InnoDB`는 충돌 시점에 존재했던 커밋되지 않은 트랜잭션을 자동으로 롤백합니다.

`InnoDB` 충돌 복구는 여러 단계로 구성됩니다:

- 테이블 공간 검색

  테이블 공간 검색은 `InnoDB`가 재실행 로그 적용이 필요한 테이블 공간을 식별하는 데 사용하는 프로세스입니다. 충돌 복구 중 테이블스페이스 검색을 참조하십시오.

- 로그 애플리케이션 다시 실행

  로그 적용 재실행은 연결을 수락하기 전에 초기화 중에 수행됩니다. 종료 또는 충돌 시 모든 변경 사항이 버퍼 풀에서 테이블 스페이스(`ibdata*` 및 `*.ibd` 파일)로 플러시되는 경우, 재실행 로그 적용이 건너뜁니다. `InnoDB`는 또한 시작 시 재실행 로그 파일이 누락된 경우에도 재실행 로그 적용을 건너뜁니다.

  - 현재 최대 자동 증가 카운터 값은 값이 변경될 때마다 재실행 로그에 기록되므로 충돌에 안전합니다. 복구 중에 `InnoDB`는 재실행 로그를 스캔하여 카운터 값 변경 사항을 수집하고 변경 사항을 인메모리 테이블 개체에 적용합니다.

    InnoDB가 자동 증가 값을 처리하는 방법에 대한 자세한 내용은 섹션 15.6.1.6, "InnoDB에서

AUTO_INCREMENT 처리" 및 InnoDB AUTO_INCREMENT 카운터 초기화를 참조하세요.

  - 인덱스 트리 손상이 발생하면 InnoDB는 재실행 로그에 손상 플래그를 기록하여 손상 플래그가 크래시로부터 안전하도록 합니다. 또한 InnoDB는 각 체크포인트의 엔진 프라이빗 시스템 테이블에 인메모리 손상 플래그 데이터를 씁니다. 복구 중에 InnoDB는 두 위치에서 손상 플래그를 읽고 결과를 병합한 후 인메모리 테이블 및 인덱스 개체를 손상된 것으로 표시합니다.

  - 일부 데이터 손실이 허용되는 경우에도 복구 속도를 높이기 위해 재실행 로그를 제거하는 것은 권장되지 않습니다. 재실행 로그 제거는 innodb_fast_shutdown을 0 또는 1로 설정한 상태에서 새로 종료한 후에만 고려해야 합니다.

- 완료되지 않은 거래 롤백

불완전한 트랜잭션은 예기치 않은 종료 또는 빠른 종료 시점에 활성 상태였던 트랜잭션입니다. 불완전한 트랜잭션을 롤백하는 데 걸리는 시간은 서버 부하에 따라 트랜잭션이 중단되기 전 활성 상태였던 시간의 3~4 배가 될 수 있습니다.

롤백 중인 트랜잭션은 취소할 수 없습니다. 극단적인 경우, 트랜잭션을 롤백하는 데 매우 오랜 시간이 걸릴 것으로 예상되는 경우, `innodb_force_recovery` 설정을 `3` 이상으로 설정하여 `InnoDB`를 시작하는 것이 더 빠를 수 있습니다. 섹션 15.21.3, "InnoDB 복구 강제 실행"을 참조하세요.

- 버퍼 병합 변경

  인덱스 페이지가 버퍼 풀로 읽혀질 때 변경 버퍼(시스템 테이블 스페이스의 일부)의 변경 사항을 보조 인덱스의 리프 페이지에 적용합니다.

- 퍼지

  활성 트랜잭션에 더 이상 표시되지 않는 삭제 표시된 레코드를 삭제합니다.

재실행 로그 적용 이후의 단계는 재실행 로그에 의존하지 않으며(쓰기 로깅을 제외하고) 일반 처리와 병렬로 수행됩니다. 이 중 불완전한 트랜잭션의 롤백만 충돌 복구에 특화되어 있습니다. 삽입 버퍼 병합과 퍼지는 정상 처리 중에 수행됩니다.

로그 적용을 다시 실행한 후 `InnoDB`는 다운타임을 줄이기 위해 가능한 한 빨리 연결을 수락하려고 시도합니다. 충돌 복구의 일환으로, `InnoDB`는 서버가 종료될 때 커밋되지 않았거나 `XA` 준비 상태에 있던 트랜잭션을 롤백합니다. 롤백은 새 연결의 트랜잭션과 병렬로 실행되는 백그라운드 스레드에 의해 수행됩니다. 롤백 작업이 완료될 때까지 새 연결에서 복구된 트랜잭션과 잠금 충돌이 발생할 수 있습니다.

대부분의 경우, 활동량이 많은 도중에 MySQL 서버가 예기치 않게 종료되더라도 복구 프로세스가 자동으로 진행되므로 DBA는 아무런 조치를 취할 필요가 없습니다. 하드웨어 오류 또는 심각한 시스템 오류로 인해 `InnoDB` 데이터가 손상된 경우 MySQL이 시작을 거부할 수 있습니다. 이 경우 섹션 15.21.3, "InnoDB 복구 강제 실행"을 참조하세요.

바이너리 로그 및 `InnoDB` 크래시 복구에 대한 자세한 내용은 5.4.4절. "바이너리 로그"를 참조하세요.

충돌 복구 중 테이블 공간 검색

복구 중에 `InnoDB`가 마지막 체크포인트 이후에 작성된 재실행 로그를 발견하면 해당 재실행 로그를 영향을 받는 테이블스페이스에 적용해야 합니다. 복구 중에 영향을 받는 테이블스페이스를 식별하는 프로세스를 *테이블스페이스 검색이라고* 합니다.

테이블스페이스 검색은 시작 시 테이블스페이스 파일을 검색할 디렉터리를 정의하는 `innodb_directories` 설정에 의존합니다. `innodb_directories` 기본 설정은 NULL이지만, InnoDB가 테이블스페이스 파일을 빌드할 때 `innodb_data_home_dir`,

innodb_undo_directory 및 datadir로 정의된 디렉터리는 항상 innodb_directories 인수 값에 추가됩니다.

시작 시 검사할 디렉터리 목록입니다. 이러한 디렉터리는 innodb_directories 설정이 명시적으로 지정되었는지 여부와 관계없이 추가됩니다. 절대 경로로 정의된 테이블스페이스 파일 또는 innodb_directories 설정에추가된 디렉터리 외부에 있는 테이블스페이스 파일은 innodb_directories 설정에 추가해야 합니다. 재실행 로그에서 참조된 테이블스페이스 파일이 이전에 발견되지 않은 경우 복구가 종료됩니다.

15.19 InnoDB 및 MySQL 복제

복제본의 스토리지 엔진이 원본의 스토리지 엔진과 동일하지 않은 방식으로 복제를 사용할 수 있습니다. 예를 들어 원본의 InnoDB 테이블에 대한 수정 내용을 복제본의 MyISAM 테이블에 복제할 수 있습니다. 자세한 내용은 17.4.4절. "다른 소스 및 복제본 스토리지 엔진으로 복제 사용"을 참조하세요.

복제본 설정에 대한 자세한 내용은 17.1.2.6절. "복제본 설정"을 참조하세요.
섹션 17.1.2.5, '데이터 스냅샷 방법 선택하기' 참조. 소스 또는 기존 복제본을 삭제하지 않고 새 복제본을 만들려면 MySQL Enterprise 백업 제품을 사용하세요.

소스에서 실패한 트랜잭션은 복제에 영향을 미치지 않습니다. MySQL 복제는 MySQL이 데이터를 수정하는 SQL 문을 작성하는 바이너리 로그를 기반으로 합니다. 실패한 트랜잭션(예: 외래 키 위반 또는 롤백으로 인해)은 바이너리 로그에 기록되지 않습니다.
를 사용하면 복제본으로 전송되지 않습니다. 섹션 13.3.1, "트랜잭션 시작, 커밋 및 롤백 문"을 참조하세요.

원본의 `InnoDB` 테이블에 대한 **복제 및 CASCADE.** Cascading 작업은 외래 키 관계를 공유하는 테이블이 원본과 복제본 모두에서 `InnoDB`를 사용하는 경우에만 복제본에서 실행됩니다.
복제본입니다. 이는 문 기반 복제를 사용하든 행 기반 복제를 사용하든 마찬가지입니다. 복제를 시작한 다음 `InnoDB`가기본 스토리지 엔진으로 정의된 원본에 다음 `CREATE TABLE` 문을 사용하여 두 개의 테이블을 생성한다고 가정합니다:

```
CREATE TABLE fc1 (
    i int 기본 키,
    j INT
);

CREATE TABLE fc2 (
    m int 기본 키,
    n INT,
    FOREIGN KEY ni (n) REFERENCES fc1 (i)
        ON DELETE CASCADE
);
```

복제본에 기본 저장소 엔진으로 `MyISAM`이 정의되어 있는 경우 복제본에 동일한 테이블이 생성되지만 `MyISAM` 저장소 엔진을 사용하며 `FOREIGN KEY` 옵션은 무시됩니다. 이제 원본의 테이블에 몇 개의 행을 삽입합니다:

```
source> INSERT INTO fc1 VALUES (1, 1), (2, 2);
쿼리 확인, 영향을 받는 행 2개(0.09초)
기록: 2개 중복: 0 경고: 0

source> INSERT INTO fc2 VALUES (1, 1), (2, 2), (3, 1);
쿼리 확인, 영향을 받는 행 3개(0.19초) 레코드: 3개
중복: 0 경고: 0
```

이 시점에서 원본과 복제본 모두에서 다음과 같이 테이블 `fc1`에는 2행이 포함되고 테이블 fc2에는 3행이 포함됩니다:

```
source> SELECT * FROM fc1;
+---+------+
| i | j    |
+---+------+
| 1 |1 |
| 2 |2 |
+---+------+
2 세트 내 행(0.00초)

source> SELECT * FROM fc2;
+---+------+
| m | n    |
+---+------+
| 1 |1 |
| 2 |2 |
| 3 |1 |
+---+------+
3 세트 내 행(0.00초)

replica> SELECT * FROM fc1;
+---+------+
| i | j    |
+---+------+
| 1 |1 |
```

```
| 2 |2 |
+---+------+
2 세트 내 행(0.00초)

replica> SELECT * FROM fc2;
+---+------+
| m | n    |
+---+------+
| 1 |1 |
| 2 |2 |
| 3 |1 |
+---+------+
3 세트 내 행(0.00초)
```

이제 소스에 대해 다음 DELETE 문을 수행한다고 가정해 보겠습니다:

```
source> DELETE FROM fc1 WHERE i=1;
쿼리 확인, 영향을 받은 행 1개(0.09초)
```

캐스케이드로 인해 소스의 테이블 fc2에는 이제 행이 1개만 포함됩니다:

```
source> SELECT * FROM fc2;
+---+---+
| m | n |
+---+---+
| 2 | 2 |
+---+---+
1행 1세트(0.00초)
```

그러나 복제본에서는 fc1에 대한 삭제가 fc2에서 행을 삭제하지 않기 때문에 캐스케이드가 복제본에 전파되지 않습니다. 복제본의 fc2 복사본에는 원래 삽입된 모든 행이 여전히 포함됩니다:

```
replica> SELECT * FROM fc2;
+---+---+
| m | n |
+---+---+
| 1 | 1 |
| 3 | 1 |
| 2 | 2 |
+---+---+
한 세트에 3줄(0.00초)
```

이 차이는 계단식 삭제가 InnoDB에서 내부적으로 처리되기 때문입니다.
스토리지 엔진을 사용하므로 변경 사항이 기록되지 않습니다.

15.20 InnoDB 멤캐시드 플러그인

참고

InnoDB 멤캐시드 플러그인은 더 이상 사용되지 않으며, 향후 MySQL 버전에서 지원이 제거될 예정입니다.

InnoDB 멤캐시드 플러그인(daemon_memcached)은 InnoDB 테이블에서 데이터를 자동으로 저장하고 검색하는 통합 멤캐시드 데몬을 제공하여 MySQL 서버를 빠른 "키-값 저장소"로 전환합니다. SQL로 쿼리를 공식화하는 대신 간단한 get, set, incr 작업을 사용하여 SQL 구문 분석 및 쿼리 최적화 계획 구성과 관련된 성능 오버헤드를 피할 수 있습니다. 또한 기존 데이터베이스 소프트웨어의 장점인 편의성, 복잡한 쿼리, 대

량 작업 등을 위해 SQL을 통해 동일한 InnoDB 테이블에 액세스할 수도 있습니다.

이 "NoSQL 스타일" 인터페이스는 멤캐시드 API를 사용하여 데이터베이스 작업 속도를 높이고, InnoDB가 버퍼 풀 메커니즘을 사용하여 메모리 캐싱을 처리할 수 있도록 합니다. 추가, 설정, 인크루트 등의 멤캐시드 작업을 통해 수정된 데이터는 InnoDB 테이블에 디스크에 저장됩니다. 조합 멤캐시드의 단순성과 InnoDB의 신뢰성 및 일관성을 결합하여 사용자에게 다음과 같은 장점을 제공합니다.

섹션 15.20.1, "InnoDB 멤캐시드 플러그인의 이점"에서 설명한 대로 두 가지를 모두 지원합니다. 아키텍처 개요는 섹션 15.20.2, "InnoDB 멤캐시드 아키텍처"를 참조하세요.

15.20.1 InnoDB 멤캐시드 플러그인의 장점

이 섹션에서는 `daemon_memcached` 플러그인의 장점에 대해 간략하게 설명합니다. `InnoDB` 테이블과 멤캐시드의 조합은 둘 중 하나를 단독으로 사용하는 것보다 이점을 제공합니다.

- `InnoDB` 스토리지 엔진에 직접 액세스하면 SQL의 구문 분석 및 계획 오버헤드를 피할 수 있습니다.
- MySQL 서버와 동일한 프로세스 공간에서 멤캐시를 실행하면 요청을 주고받는 데 따른 네트워크 오버헤드를 피할 수 있습니다.
- 멤캐시드 프로토콜을 사용하여 쓰여진 데이터는 MySQL SQL 계층을 거치지 않고 `InnoDB` 테이블에 투명하게 기록됩니다. 중요하지 않은 데이터를 업데이트할 때 쓰기 빈도를 제어하여 더 높은 원시 성능을 달성할 수 있습니다.
- 멤캐시드 프로토콜을 통해 요청된 데이터는 MySQL SQL 계층을 거치지 않고 `InnoDB` 테이블에서 투명하게 쿼리됩니다.
- 동일한 데이터에 대한 후속 요청은 `InnoDB` 버퍼 풀에서 제공됩니다. 버퍼 풀은 인메모리 캐싱을 처리합니다. `InnoDB` 구성 옵션을 사용하여 데이터 집약적인 작업의 성능을 조정할 수 있습니다.
- 데이터는 애플리케이션 유형에 따라 비정형 또는 정형 데이터일 수 있습니다. 데이터에 대한 새 테이블을 만들거나 기존 테이블을 사용할 수 있습니다.
- `InnoDB`는 여러 열 값을 하나의 멤캐시된 항목 값으로 구성하고 분해하는 작업을 처리할 수 있으므로 애플리케이션에 필요한 문자열 구문 분석 및 연결의 양을 줄일 수 있습니다. 예를 들어 문자열 값 `2|4|6|8`을 멤캐시드 캐시에 저장하고 `InnoDB`가 구분 문자를 기준으로 값을 분할한 다음 그 결과를 4개의 숫자 열에 저장하도록 할 수 있습니다.
- 메모리와 디스크 간의 전송이 자동으로 처리되므로 애플리케이션 로직이 간소화됩니다.
- 데이터는 충돌, 중단 및 손상으로부터 보호하기 위해 MySQL 데이터베이스에 저장됩니다.
- 보고, 분석, 임시 쿼리, 대량 로딩, 다단계 트랜잭션 계산, 유니온 및 교차 등의 집합 연산, 기타 SQL의 표현력과 유연성에 적합한 연산을 위해 SQL을 통해 기본 `InnoDB` 테이블에 액세스할 수 있습니다.
- 소스 서버에서 `daemon_memcached` 플러그인을 MySQL 복제와 함께 사용하면 고가용성을 보장할 수 있습니다.
- 멤캐시와 MySQL의 통합은 인메모리 데이터를 영구적으로 만들 수 있는 방법을 제공하므로 더 중요한 종류의 데이터에 사용할 수 있습니다. 데이터 손실에 대한 걱정 없이 애플리케이션에서 더 많은 `add`, `incr` 및 유사한 쓰기 작업을 사용할 수 있습니다. 캐시된 데이터에 대한 업데이트를 잃지 않고 멤캐시드 서버를 중지 및 시작할 수 있습니다. 예기치 않은 중단에 대비하기 위해 `InnoDB` 충돌 복구, 복제 및 백업 기능을

활용할 수 있습니다.

- InnoDB가 빠른 기본 키 조회를 수행하는 방식은 멤캐시된 단일 항목 쿼리에 자연스럽게 적합합니다. daemon_memcached 플러그인이 사용하는 직접적이고 낮은 수준의 데이터베이스 액세스 경로는 동등한 SQL 쿼리보다 키-값 조회에 훨씬 더 효율적입니다.
- 복잡한 데이터 구조, 바이너리 파일 또는 코드 블록을 저장 가능한 문자열로 변환할 수 있는 memcached의 직렬화 기능은 이러한 객체를 데이터베이스로 가져오는 간단한 방법을 제공합니다.
- SQL을 통해 기본 데이터에 액세스할 수 있으므로 보고서를 생성하고, 여러 키에 걸쳐 검색 또는 업데이트하고, 멤캐시된 데이터에서 AVG(), MAX() 등의 함수를 호출할 수 있습니다. 이러한 모든 작업은 멤캐시를 단독으로 사용하면 비용이 많이 들거나 복잡합니다.
- 시작할 때 데이터를 멤캐시에 수동으로 로드할 필요가 없습니다. 애플리케이션에서 특정 키를 요청하면 데이터베이스에서 값이 자동으로 검색되고 InnoDB 버퍼 풀을 사용하여 메모리에 캐시됩니다.

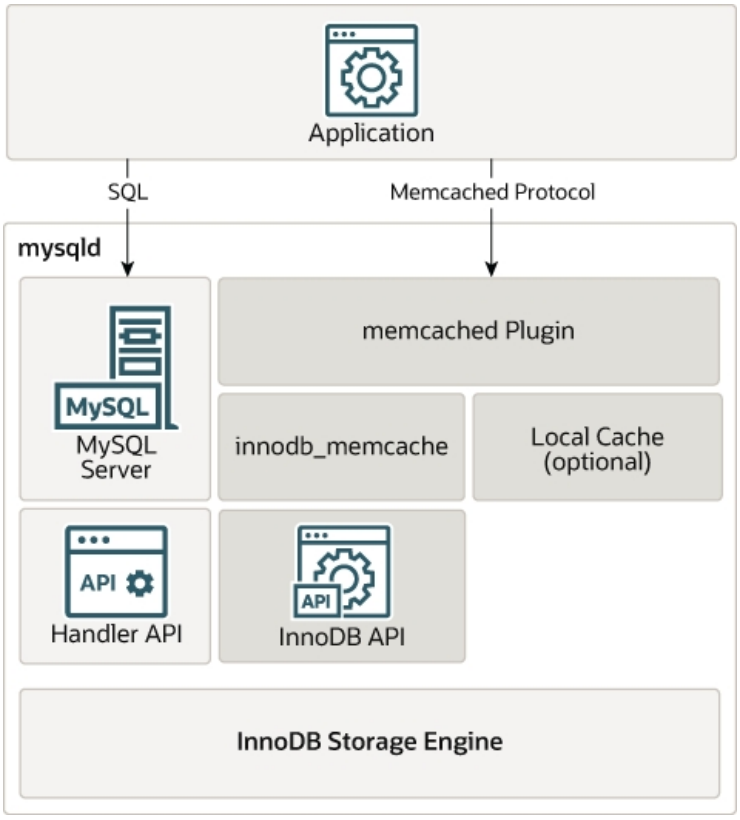
- 멤캐시드는 CPU를 상대적으로 적게 사용하고 메모리 공간을 쉽게 제어할 수 있기 때문에 동일한 시스템에서 MySQL 인스턴스와 함께 편안하게 실행할 수 있습니다.
- 데이터 일관성은 일반 `InnoDB` 테이블에 사용되는 메커니즘에 의해 적용되므로 키가 누락된 경우 데이터베이스를 쿼리하기 위해 오래된 멤캐시 데이터나 폴백 로직에 대해 걱정할 필요가 없습니다.

15.20.2 InnoDB 멤캐시드 아키텍처

`InnoDB` 멤캐시드 플러그인은 멤캐시드를 MySQL SQL 계층을 우회하여 `InnoDB` 스토리지 엔진에 직접 액세스하는 MySQL 플러그인 데몬으로 구현합니다.

다음 다이어그램은 애플리케이션이 `daemon_memcached`를 통해 데이터에 액세스하는 방법을 보여줍니다. 플러그인과 비교합니다.

그림 15.4 멤캐시드 서버가 통합된 MySQL 서버



`daemon_memcached` 플러그인의 특징:

- 멤캐시드는 `mysqld`의 데몬 플러그인으로 사용됩니다. mysqld와 memcached는 모두 동일한 프로세스 공간에서 실행되며, 데이터에 매우 짧은 지연 시간으로 액세스할 수 있습니다.
- SQL 파서, 옵티마이저, 심지어 핸들러 API 계층까지 우회하여 `InnoDB` 테이블에 직접 액세스할 수 있습니다.
    - 텍스트 기반 프로토콜과 바이너리 프로토콜을 포함한 표준 멤캐시드 프로토콜. 그리고 `daemon_memcached` 플러그인은 `memcapable` 명령의 55개 호환성 테스트를 모두 통과했습니다.

- 다중 열 지원. 여러 열을 키-값 저장소의 '값' 부분에 매핑할 수 있으며, 열 값은 사용자가 지정한 구분 문자로 구분할 수 있습니다.

- 기본적으로 멤캐시드 프로토콜은 InnoDB에 직접 데이터를 읽고 쓰는 데 사용되며, MySQL은 InnoDB 버퍼 풀을 사용하여 인메모리 캐싱을 관리할 수 있습니다. 기본 설정은 데이터베이스 애플리케이션에 대한 높은 안정성과 가장 적은 놀라움의 조합을 나타냅니다. 예를 들어

기본 설정은 데이터베이스 측에서 커밋되지 않은 데이터 또는 멤캐시드 가져오기 요청에 대해 반환되는 오래된 데이터를 방지합니다.

- 고급 사용자는 모든 데이터가 멤캐시드 엔진(메모리 캐싱)에만 캐시되는 기존 멤캐시드 서버로 시스템을 구성하거나 "멤캐시드 엔진"(메모리 캐싱)과 InnoDB 멤캐시드 엔진(백엔드 영구 스토리지로서의 InnoDB)을 조합하여 사용할 수 있습니다.

- innodb_api_bk_commit_interval, daemon_memcached_r_batch_size 및 daemon_memcached_w_batch_size 구성 옵션을 통해 InnoDB와 멤캐시드 작업 간에 데이터를 주고받는 빈도를 제어할 수 있습니다. 배치 크기 옵션은 안정성을 극대화하기 위해 기본값이 1로 설정되어 있습니다.

- daemon_memcached_option 구성 매개변수를 통해 멤캐시드 옵션을 지정하는 기능. 예를 들어 멤캐시가 수신 대기하는 포트를 변경하거나, 최대 동시 연결 수를 줄이거나, 키-값 쌍의 최대 메모리 크기를 변경하거나, 오류 로그에 대한 디버깅 메시지를 사용하도록 설정할 수 있습니다.

- innodb_api_trx_level 구성 옵션은 memcached가 처리하는 쿼리에 대한 트랜잭션 격리 수준을 제어합니다. memcached에는 트랜잭션 개념이 없지만, 이 옵션을 사용하여 이 플러그인이 사용하는 테이블에서 실행된 SQL 문으로 인한 변경 사항을 memcached가 얼마나 빨리 확인하는지 제어할 수 있습니다. 기본적으로 innodb_api_trx_level은 READ UNCOMMITTED로 설정되어 있습니다.

- innodb_api_enable_mdl 옵션을 사용하면 MySQL 수준에서 테이블을 잠글 수 있으므로 매핑된 테이블이 SQL 인터페이스를 통해 DDL에 의해 삭제되거나 변경될 수 없습니다. 잠금을 설정하지 않으면 테이블을 MySQL 계층에서 삭제할 수 있지만, 멤캐시되거나 다른 사용자가 사용을 중단할 때까지 InnoDB 스토리지에 보관할 수 있습니다. "MDL"은 "메타데이터 잠금"을 의미합니다.

15.20.3 InnoDB 멤캐시드 플러그인 설정하기

이 섹션에서는 MySQL 서버에서 daemon_memcached 플러그인을 설정하는 방법을 설명합니다. 멤캐시드 데몬은 네트워크 트래픽을 피하고 지연 시간을 최소화하기 위해 MySQL 서버와 긴밀하게 통합되어 있으므로 이 기능을 사용하는 각 MySQL 인스턴스에서 이 프로세스를 수행해야 합니다.

참고

daemon_memcached 플러그인을 설정하기 전에 섹션 15.20.5, "InnoDB memcached 플러그인에 대한 보안 고려 사항"을 참조하여 무단 액세스를 방지하는 데 필요한 보안 절차를 이해하시기 바랍니다.

전제 조건

- daemon_memcached 플러그인은 Linux, Solaris 및 macOS 플랫폼에서만 지원됩니다. 다른 운영 체제는 지원되지 않습니다.

- 소스에서 MySQL을 빌드할 때는 `-DWITH_INNODB_MEMCACHED=ON`으로 빌드해야 합니다. 이 빌드 옵션은 MySQL 플러그인 디렉터리(`plugin_dir`)에 `daemon_memcached` 플러그인을 실행하는 데 필요한 공유 라이브러리 2개를 생성합니다:
  - `libmemcached.so`: MySQL에 대한 멤캐시드 데몬 플러그인.
  - `innodb_engine.so`: 멤캐시드용 `InnoDB` API 플러그인.
- 를 설치해야 합니다.
  - 소스에서 MySQL을 빌드하지 않은 경우, 라이브러리 라이브러리는 설치에 포함되지 않습니다. 운영 체제에 맞는 설치 방법을 사용하여 `libevent` 1.4.12 이상을 설치하세요. 예를 들어, 운영 체제에 따라 `apt-get`, `yum` 또는 `port install`을 사용할 수 있습니다. 예를 들어 우분투 리눅스에서는 다음을 사용합니다:

```
sudo apt-get 설치 이벤트-개발
```

- 소스 코드 릴리스에서 MySQL을 설치한 경우 libevent 1.4.12는 패키지와 함께 번들로 제공되며 MySQL 소스 코드 디렉터리의 최상위 레벨에 위치합니다. 번들 버전의 libevent를 사용하는 경우 별도의 조치가 필요하지 않습니다. 로컬 시스템 버전의 libevent를 사용하려면 -DWITH_LIBEVENT 빌드 옵션을 시스템 또는 yes로 설정하여 MySQL을 빌드해야 합니다.

InnoDB 멤캐시드 플러그인 설치 및 구성

1. MYSQL_HOME/share에 있는 innodb_memcached_config.sql 구성 스크립트를 실행하여 InnoDB 테이블과 상호 작용할 수 있도록 daemon_memcached 플러그인을 구성합니다. 이 스크립트는 세 가지 필수 테이블(캐시 정책, 구성 옵션, 컨테이너)과 함께 innodb_memcache 데이터베이스를 설치합니다. 또한 테스트 데이터베이스에 demo_test 샘플 테이블을 설치합니다.

```
mysql> 소스 MYSQL_HOME/share/innodb_memcached_config.sql
```

innodb_memcached_config.sql 스크립트 실행은 한 번만 수행하면 됩니다. 나중에 daemon_memcached 플러그인을 제거했다가 다시 설치해도 테이블은 그대로 유지됩니다.

```
mysql> USE innodb_memcache;
mysql> SHOW TABLES;
+----------------------------+
| Tables_in_innodb_memcache |
+----------------------------+
| 캐시 정책                    |
| config_options             |
| 컨테이너                     |
+----------------------------+

mysql> USE 테스트;
mysql> SHOW TABLES;
+----------------+
| Tables_in_test |
+----------------+
| 데모 테스트       |
+----------------+
```

이 테이블 중 innodb_memcache.containers 테이블이 가장 중요합니다. containers 테이블의 항목은 InnoDB 테이블 열에 대한 매핑을 제공합니다. daemon_memcached 플러그인과 함께 사용되는 각 InnoDB 테이블에는 컨테이너 테이블의 항목이 필요합니다.

innodb_memcached_config.sql 스크립트는 demo_test 테이블에 대한 매핑을 제공하는 컨테이너 테이블에 단일 항목을 삽입합니다. 또한 demo_test 테이블에 단일 데이터 행을 삽입합니다. 이 데이터를 통해 설정이 완료된 후 즉시 설치를 확인할 수 있습니다.

```
mysql> SELECT * FROM innodb_memcache.containers\G
*************************** 1. 행 *************************** 이
                  름: aaa
            DB_SCHEMA: 테스트
             DB_TABLE: DEMO_TEST
          KEY_COLUMNS: C1
        VALUE_COLUMNS: C2
                플래그: C3
           CAS_COLUMN: C4
   EXPIRE_TIME_COLUMN: C5
고유_IDX_NAME_ON_KEY: PRIMARY

mysql> SELECT * FROM test.demo_test;
+----+-----------------+------+-----+----------+
| C1 |                 C2| C3| C4| C5 |
+----+-----------------+------+-----+----------+
| aa | 안녕하세요, 안녕하세요     |8 |0 |0 |
+----+-----------------+------+-----+----------+
```

`innodb_memcache` 테이블 및 `demo_test` 샘플 테이블에 대한 자세한 내용은 15.20.8절, 'InnoDB 멤캐시드 플러그인 내부'를 참조하세요.

2. INSTALL PLUGIN 문을 실행하여 daemon_memcached 플러그인을 활성화합니다:

```
mysql> INSTALL PLUGIN daemon_memcached soname "libmemcached.so";
```

플러그인이 설치되면 MySQL 서버가 다시 시작될 때마다 자동으로 활성화됩니다.

InnoDB 및 멤캐시드 설정 확인

daemon_memcached 플러그인 설정을 확인하려면 텔넷 세션을 사용하여 memcached를 발행합니다. 명령을 사용합니다. 기본적으로 멤캐시드 데몬은 포트 11211에서 수신 대기합니다.

1. test.demo_test 테이블에서 데이터를 검색합니다. demo_test 테이블의 단일 데이터 행의 키 값은 AA입니다.

```
텔넷 로컬호스트 11211
127.0.0.1 시도 중...
로컬 호스트에 연결되었습니다. 이스
케이프 문자는 '^]'입니다. get
AA
값 AA 8 12 헬로,
헬로 종료
```

2. 집합 명령을 사용하여 데이터를 삽입합니다.

```
설정 BB 10 0 16 안
녕, 안녕 저장됨
```

어디에:

- set은 값을 저장하는 명령입니다.
- BB가 핵심입니다.
- 10은 연산에 대한 플래그입니다; 멤캐시드에서는 무시되지만 클라이언트에서 모든 유형의 정보를 나타내는 데 사용할 수 있습니다; 사용하지 않는 경우 0을 지정합니다.
- 0은만료 시간(TTL)이며, 사용하지 않을 경우 0을 지정합니다.
- 16은 제공된 값 블록의 길이(바이트)입니다.
- GOODBYE, GOODBYE는 저장되는 값입니다.

3. MySQL 서버에 연결하고 test.demo_test 테이블을 쿼리하여 삽입된 데이터가 MySQL에 저장되었는지 확인합니다.

```
mysql> SELECT * FROM test.demo_test;
+----+-----------------+------+-----+-----------+
| C1 |                 C2| C3| C4| C5 |
+----+-----------------+------+-----+-----------+
| aa | 여보세요, 여보세요   |    8 |    0 |    0 |
| bb | 안녕, 안녕            |10 |1 |0 |0 |
+----+-----------------+------+-----+-----------+
```

4. 텔넷 세션으로 돌아가서 `BB` 키를 사용하여 이전에 삽입한 데이터를 검색합니다.

```
BB 받기
값 BB 10 16 안녕,
안녕 끝
종료
```

통합 멤캐시드 서버도 종료하는 MySQL 서버를 종료하면 멤캐시드 데이터에 대한 추가 액세스 시도가 연결 오류와 함께 실패합니다. 일반적으로 멤캐시된 데이터도 이 시점에서 사라지며, 데이터를 다시 메모리로 로드하려면 애플리케이션 로직이 필요합니다.

를 다시 시작해야 합니다. 하지만 InnoDB 멤캐시드 플러그인은 이 프로세스를 자동화합니다.

MySQL을 다시 시작하면 get 작업은 이전 memcached 세션에 저장한 키-값 쌍을 다시 한 번 반환합니다. 키가 요청되고 관련 값이 메모리 캐시에 아직 없는 경우, 해당 값은 MySQL test.demo_test 테이블에서 자동으로 쿼리됩니다.

새 테이블 및 열 매핑 만들기

이 예제는 daemon_memcached 플러그인을 사용하여 자체 InnoDB 테이블을 설정하는 방법을 보여줍니다.

1. InnoDB 테이블을 만듭니다. 테이블에는 고유 인덱스가 있는 키 열이 있어야 합니다. 도시 테이블의 키 열은 기본 키로 정의된 city_id입니다. 테이블에는 플래그, CAS 및 만료 값에 대한 열도 포함되어야 합니다. 하나 이상의 값 열이 있을 수 있습니다. 도시 테이블에는 세 개의 값 열(이름, 주, 국가)이 있습니다.

참고

열 이름과 관련된 특별한 요구 사항은 없으며, 유효한 매핑이 innodb_memcache.containers 테이블에 추가되기만 하면 됩니다.

```
mysql> CREATE TABLE city (
       city_id VARCHAR(32),
       name VARCHAR(1024),
       state VARCHAR(1024),
       국가 VARCHAR(1024), 플래그
       INT,
       cas BIGINT UNSIGNED,
       만료 INT,
       기본 키(CITY_ID)
       ) 엔진=InnoDB;
```

2. daemon_memcached 플러그인이 InnoDB 테이블에 액세스하는 방법을 알 수 있도록 innodb_memcache.containers 테이블에 항목을 추가합니다. 이 항목은 innodb_memcache.containers 테이블 정의를 만족해야 합니다. 각 필드에 대한 설명은 섹션 15.20.8, "InnoDB 멤캐시드 플러그인 내부"를 참조하세요.

```
mysql> DESCRIBE innodb_memcache.container;
+----------------------+-------------+------+-----+---------+-------------+
|                      필드| 유형|      Null | 키 | 기본값 | 추가 |
+----------------------+-------------+------+-----+---------+-------------+
| 이름                 | varchar(50) | NO   | PRI | NULL    |             |
| db_schema            | varchar(250) | NO   |     | NULL    |             |
| db_table             | varchar(250) | NO   |     | NULL    |             |
| 키_열                | varchar(250) | NO   |     | NULL    |             |
| 값_열                | varchar(250) | YES  |     | NULL    |             |
| 플래그               | varchar(250) | NO   |     | 0       |             |
| cas_column           | varchar(250) | YES  |     | NULL    |             |
| 만료_시간_열         | varchar(250) | YES  |     | NULL    |             |
| 고유_IDX_이름_온_키 | varchar(250) | NO                || NULL    |             |
+----------------------+-------------+------+-----+---------+-------------+
```

도시 테이블에 대한 innodb_memcache.containers 테이블 항목은 다음과 같이 정의됩니다:

```
mysql> INSERT INTO `innodb_memcache`.`containers` (
       name`, `db_schema`, `db_table`, `key_column`, `value_column`,
       `flags`, `cas_column`, `expire_time_column`, `unique_idx_name_on_key`)
       VALUES ('default', 'test', 'city', 'city_id', 'name|state|country',
       'flags', 'cas', 'expiry','PRIMARY');
```

- 를 containers.name 열에 지정하면 도시 테이블을 daemon_memcached 플러그인과 함께 사용할 기본 InnoDB 테이블로 구성할 수 있습니다.
- 여러 InnoDB 테이블 열(이름, 주, 국가)이 다음에 매핑됩니다.
  "|" 구분 기호를 사용하여 containers.value_columns에 추가합니다.

- 일반적으로 daemon_memcached 플러그인을 사용하는 애플리케이션에서는 innodb_memcache.containers 테이블의 flag, cas_column, expire_time_column 필드가 중요하지 않습니다. 그러나 각각에 대해 지정된 InnoDB 테이블 열이 필요합니다. 데이터를 삽입할 때 이러한 열이 사용되지 않는 경우 0을 지정하세요.

3. innodb_memcache.containers 테이블을 업데이트한 후 daemon_memcache 플러그인을 다시 시작하여 변경 사항을 적용합니다.

```
mysql> UNINSTALL PLUGIN daemon_memcached;

mysql> INSTALL PLUGIN daemon_memcached soname "libmemcached.so";
```

4. 텔넷을 사용하여 memcached set 명령을 사용하여 도시 테이블에 데이터를 삽입합니다.

```
텔넷 로컬호스트 11211
127.0.0.1 시도 중...
로컬 호스트에 연결되었습니다. 이스
케이프 문자는 '^]'입니다. set B
0 0 22 방갈로르|방갈로르|IN
STORED
```

5. MySQL을 사용하여 test.city 테이블을 쿼리하여 삽입한 데이터가 저장되었는지 확인합니다.

```
mysql> SELECT * FROM test.city;
+---------+-----------+-----------+---------+-------+------+--------------+
| 도시 아이디 | 이름 | 주              | 국가 | 플래그 | CAS |    만료일 |
+---------+-----------+-----------+---------+-------+------+--------------+
| b|       방갈로르 | 방갈로르 | in              |0 |3 |0 |
+---------+-----------+-----------+---------+-------+------+--------------+
```

6. MySQL을 사용하여 test.city 테이블에 추가 데이터를 삽입합니다.

```
mysql> INSERT INTO city VALUES ('C','CHENNAI','TAMIL NADU','IN', 0, 0 ,0);
mysql> INSERT INTO city VALUES ('D','DELHI','DELHI','IN', 0, 0, 0);
mysql> INSERT INTO city VALUES ('H','HYDERABAD','TELANGANA','IN', 0, 0, 0);
mysql> INSERT INTO city VALUES ('M','MUMBAI','MAHARASHTRA','IN', 0, 0, 0);
```

참고

사용하지 않는 경우 flag, cas_column 및 expire_time_column 필드에 0 값을 지정하는 것이 좋습니다.

7. 텔넷을 사용하여 memcached get 명령을 실행하여 MySQL을 사용하여 삽입한 데이터를 검색합니다.

```
get H
값 H 0 22 하이데라바드|텔랑가
나|인엔드
```

InnoDB 멤캐시드 플러그인 구성하기

기존의 멤캐시드 구성 옵션은 MySQL 구성 파일 또는 mysqld 시작 문자열에 지정할 수 있으며, daemon_memcached_option 구성 매개변수의 인수로 인코딩됩니다. 멤캐시드 구성 옵션은 플러그인이 로드될 때 적용되며, 이는 MySQL 서버가 시작될 때마다 발생합니다.

예를 들어, 기본 포트 11211 대신 포트 11222에서 멤캐시드가 수신 대기하도록 하려면 `-p11222`를 `daemon_memcached_option` 구성 옵션의 인수로 지정합니다:

```
mysqld .... --daemon_memcached_option="-p11222"
```

다른 멤캐시드 옵션은 `daemon_memcached_option` 문자열에 인코딩할 수 있습니다. 예를 들어 최대 동시 연결 수를 줄이거나 키-값 쌍의 최대 메모리 크기를 변경하거나 오류 로그에 대한 디버깅 메시지를 활성화하는 등의 옵션을 지정할 수 있습니다.

daemon_memcached 플러그인 전용 구성 옵션도 있습니다. 여기에는 다음이 포함됩니다:

- daemon_memcached_engine_lib_name: InnoDB 멤캐시드 플러그인을 구현하는 공유 라이브러리를 지정합니다. 기본 설정은 innodb_engine.so입니다.
- DAEMON_MEMCACHED_ENGINE_LIB_경로: InnoDB 멤캐시드 플러그인을 구현하는 공유 라이브러리가 포함된 디렉터리 경로입니다. 기본값은 플러그인 디렉터리를 나타내는 NULL입니다.
- DAEMON_MEMCACHED_R_BATCH_SIZE: 읽기 작업(get)에 대한 일괄 커밋 크기를 정의합니다. 이 값은 커밋이 발생하는 멤캐시드 읽기 작업의 수를 지정합니다. 모든 get 요청은 데이터가 멤캐시드를 통해 업데이트되었는지 또는 SQL을 통해 업데이트되었는지 여부에 관계없이 InnoDB 테이블에서 가장 최근에 커밋된 데이터에 액세스하도록 기본적으로 daemon_memcached_r_batch_size가 1로 설정되어 있습니다. 값이 1보다 크면 각 get 호출마다 읽기 작업에 대한 카운터가 증가합니다. flush_all 호출은 읽기 카운터와 쓰기 카운터를 모두 초기화합니다.
- DAEMON_EMCACHED_W_BATCH_SIZE: 쓰기 작업(set, replace, add, prepend, incr, decr 등)에 대한 일괄 커밋 크기를 정의합니다. 운영 중단 시 커밋되지 않은 데이터가 손실되지 않고 기초 테이블의 SQL 쿼리가 가장 최근 데이터에 액세스할 수 있도록 daemon_memcached_w_batch_size는 기본적으로 1로 설정되어 있습니다. 값이 1보다 크면 추가, 설정, 인크루트, 디크루트, 삭제 호출이 있을 때마다 쓰기 작업에 대한 카운터가 증가합니다. flush_all 호출은 읽기 및 쓰기 카운터를 모두 초기화합니다.

기본적으로 daemon_memcached_engine_lib_name 또는 daemon_memcached_engine_lib_path는 수정할 필요가 없습니다. 예를 들어 멤캐시에 다른 스토리지 엔진(예: NDB 멤캐시드 엔진)을 사용하려는 경우 이러한 옵션을 구성할 수 있습니다.

daemon_memcached 플러그인 구성 매개변수는 MySQL 구성 파일 또는 mysqld 시작 문자열에 지정할 수 있습니다. 이 매개변수는 daemon_memcached 플러그인을 로드할 때 적용됩니다.

daemon_memcached 플러그인 구성을 변경할 때는 플러그인을 다시 로드하여 변경 사항을 적용합니다. 이렇게 하려면 다음 문을 실행합니다:

```
mysql> UNINSTALL PLUGIN daemon_memcached;

mysql> INSTALL PLUGIN daemon_memcached soname "libmemcached.so";
```

구성 설정, 필수 테이블 및 데이터는 플러그인이 다시 시작될 때 보존됩니다.

플러그인 활성화 및 비활성화에 대한 자세한 내용은 5.6.1절. "플러그인 설치 및 제거"를 참조하세요.

15.20.4 InnoDB 멤캐시 다중 가져오기 및 범위 쿼리 지원

daemon_memcached 플러그인은 다중 가져오기 작업(단일 멤캐시 쿼리에서 여러 키-값 쌍 가져오기)과 범위 쿼리를 지원합니다.

다중 가져오기 연산

단일 멤캐시 쿼리에서 여러 키-값 쌍을 가져오는 기능은 클라이언트와 서버 간의 통신 트래픽을 줄여 읽기 성능을 향상시킵니다. InnoDB의 경우, 이는 트랜잭션과 오픈 테이블 작업의 감소를 의미합니다.

다음 예제에서는 다중 가져오기 지원을 보여 줍니다. 이 예에서는 새 테이블 만들기 및 열 매핑에 설명된 test.city 테이블을 사용합니다.

```
mysql> USE 테스트;
mysql> SELECT * FROM test.city;
+---------+-----------+------------+---------+-------+------+--------------+
| 도시 아이디 | 이름 | 주                | 국가 | 플래그 | CAS |    만료일 |
+---------+-----------+------------+---------+-------+------+--------------+
| B                | 방갈로르 | 방갈로르| in       |     0 |    1 |       0 |
| C              | 첸나이 |    타밀 나두 | IN       |     0 |    0 |       0 |
| D                        | 델리| 델리| IN       |     0 |    0 |       0 |
```

```
| H        | 하이데라바드 | 텔랑가나 | IN           |      0 |     0 |       0 |
| M| MUMBAI|           MAHARASHTRA | IN         |0 |0 |0 |
+---------+----------+------------+---------+-------+------+ --------------+
```

get 명령을 실행하여 도시 테이블에서 모든 값을 검색합니다. 결과는 키-값 쌍 시퀀스로 반환됩니다.

```
텔넷 127.0.0.1 11211
127.0.0.1 시도 중...
127.0.0.1에 연결되었습니다.
이스케이프 문자는 '^]'입니다.
get B C D H M
값 B 0 22 방갈로르|방갈로르|
인 값 C 0 21 첸나이|타밀 나
두|인 값 D 0 14 델리|델리|인
값 H 0 22 하이데라바드|텔랑
가나|인 값 M 0 21 뭄바이|마
하라슈트라|인 끝
```

하나의 get 명령으로 여러 값을 검색하는 경우 @@containers.*name* 표기법을 사용하여 테이블을 전환하여 첫 번째 키의 값을 검색할 수 있지만 후속 키의 테이블은 전환할 수 없습니다. 예를 들어 이 예의 테이블 전환은 유효합니다:

```
get @@aaa.AA BB
VALUE @@aaa.AA 8 12
HELLO, HELLO
값 BB 10 16 안녕,
안녕 끝
```

동일한 get 명령에서 테이블을 다시 전환하여 다른 테이블에서 키 값을 검색하려고 시도하는 것은 지원되지 않습니다.

다중 가져오기 작업으로 검색할 수 있는 키 수에는 제한이 없지만 결과를 저장하는 데는 128MB 메모리 제한이 있습니다.

범위 쿼리

범위 쿼리의 경우, daemon_memcached 플러그인은 다음과 같은 비교 연산자를 지원합니다: <, >, <=연산자 앞에는 @ 기호가 와야 합니다. 범위 쿼리에서 일치하는 키-값 쌍을 여러 개 찾으면 결과가 키-값 쌍 시퀀스로 반환됩니다.

다음 예는 범위 쿼리 지원을 보여줍니다. 이 예에서는 새 테이블 만들기 및 열 매핑에 설명된 test.city 테이블을 사용합니다.

```
mysql> SELECT * FROM test.city;
+---------+----------+------------+---------+-------+------+ --------------+
| 도시 아이디 | 이름 | 주                | 국가 | 플래그 | CAS |   만료일 |
+---------+----------+------------+---------+-------+------+ --------------+
| B        | 방갈로르     | 방갈로르       | IN       |      0 |     1 |       0 |
| C        | 첸나이       | 타밀 나두      | IN       |      0 |     0 |       0 |
| D        | 델리         | 델리           | IN       |      0 |     0 |       0 |
| H        | 하이데라바드 | TELANGANA    | IN       |      0 |     0 |       0 |
| M| MUMBAI|           MAHARASHTRA | IN         |0 |0 |0 |
```

텔넷 세션을 엽니다:

```
텔넷 127.0.0.1 11211
127.0.0.1 시도 중...
127.0.0.1에 연결되었습니다.
이스케이프 문자는 '^]'입니다.
```

B보다 큰 모든 값을 가져오려면 `get @>B`를 입력합니다:

```
get @>B
값 C 0 21 첸나이|타밀 나두|
인 값 D 0 14 델리|델리|인
값 H 0 22 하이데라바드|텔랑
가나|인 값 M 0 21 뭄바이|마
하라슈트라|인 끝
```

M보다 작은 모든 값을 가져오려면 `get @<M`을 입력합니다:

```
get @<M
값 B 0 22 방갈로르|방갈로르|
인 값 C 0 21 첸나이|타밀 나
두|인 값 D 0 14 델리|델리|인
값 H 0 22 하이데라바드|텔랑가
나|인엔드
```

M보다 작거나 M을 포함한 모든 값을 가져오려면 `get @<=M`을 입력합니다:

```
get @<=M
값 B 0 22 방갈로르|방갈로르|
인 값 C 0 21 첸나이|타밀 나
두|인 값 D 0 14 델리|델리|인
가치 H 0 22 하이데라바드|텔랑
가나|인 가치 M 0 21 뭄바이|마
하라슈트라|인
```

B보다 크지만 M보다 작은 값을 가져오려면 `get @>B@<M`을 입력합니다:

```
get @>B@<M
값 C 0 21 첸나이|타밀 나두|
인 값 D 0 14 델리|델리|인
값 H 0 22 하이데라바드|텔랑가
나|인엔드
```

최대 두 개의 비교 연산자를 구문 분석할 수 있는데, 하나는 '보다 작음'(`@<`) 또는 '보다 작거나 같음'(`@<=`) 연산자이고, 다른 하나는 '보다 큼'(`@>`) 또는 '보다 크거나 같음'(`@>=`) 연산자입니다. 추가 연산자는 키의 일부로 간주됩니다. 예를 들어 연산자가 3개인 `get` 명령을 실행하는 경우 세 번째 연산자(`@>C`)는 키의 일부로 간주되며 `get` 명령은 M보다 작고 B@>C보다 큰 값을 검색합니다.

```
get @<M@>B@>C
값 C 0 21 첸나이|타밀 나두|
인 값 D 0 14 델리|델리|인
값 H 0 22 하이데라바드|텔랑가
나|인
```

15.20.5 InnoDB 멤캐시드 플러그인에 대한 보안 고려 사항

주의

프로덕션 서버 또는 MySQL 인스턴스에 민감한 데이터가 포함된 경우 테스트 서

버에 `daemon_memcached` 플러그인을 배포하기 전에 이 섹션을 참조하세요.

멤캐시드는 기본적으로 인증 메커니즘을 사용하지 않으며, 선택 사항인 SASL 인증은 기존 DBMS 보안 조치만큼 강력하지 않으므로, 중요하지 않은 데이터만 `daemon_memcached` 플러그인을 사용하는 MySQL 인스턴스에 보관하고 이 구성을 사용하는 모든 서버를 잠재적 침입자로부터 차단하세요. 인터넷에서 이러한 서버에 대한 멤캐시드 액세스를 허용하지 말고, 방화벽이 설치된 인트라넷 내에서만 액세스를 허용하며, 멤버십을 제한할 수 있는 서브넷에서만 액세스를 허용하는 것이 바람직합니다.

SASL을 사용하여 멤캐시드 암호 보호하기

SASL 지원은 멤캐시드 클라이언트를 통한 인증되지 않은 액세스로부터 MySQL 데이터베이스를 보호할 수 있는 기능을 제공합니다. 이 섹션에서는 `daemon_memcached` 플러그인으로 SASL을 활성화하는 방법을 설명합니다. 단계는 기존 멤캐시드 서버에서 SASL을 사용하도록 설정하는 것과 거의 동일합니다.

SASL은 연결 기반 프로토콜에 인증 지원을 추가하기 위한 표준인 "단순 인증 및 보안 계층"의 약자입니다. memcached는 버전 1.4.3에서 SASL 지원을 추가했습니다.

> SASL 인증은 바이너리 프로토콜에서만 지원됩니다.

멤캐시드 클라이언트는 `innodb_memcache.containers` 테이블에 등록된 `InnoDB` 테이블에만 액세스할 수 있습니다. DBA가 이러한 테이블에 액세스 제한을 설정할 수 있지만, 멤캐시드 애플리케이션을 통한 액세스는 제어할 수 없습니다. 이러한 이유로 `daemon_memcached` 플러그인과 연결된 `InnoDB` 테이블에 대한 액세스를 제어하기 위해 SASL 지원이 제공됩니다.

다음 섹션에서는 SASL이 활성화된 `daemon_memcached`를 빌드, 활성화 및 테스트하는 방법을 보여줍니다. 플러그인.

InnoDB 멤캐시드 플러그인을 사용한 SASL 구축 및 활성화

기본적으로 SASL을 사용하려면 SASL 라이브러리를 사용하여 `memcached`를 빌드해야 하므로 MySQL 릴리스 패키지에는 SASL을 지원하는 `daemon_memcached` 플러그인이 포함되어 있지 않습니다. SASL 지원을 사용하려면 MySQL 소스를 다운로드하고 SASL 라이브러리를 다운로드한 후 `daemon_memcached` 플러그인을 다시 빌드하세요:

1. SASL 개발 및 유틸리티 라이브러리를 설치합니다. 예를 들어 Ubuntu에서는 apt-get을 사용하여 라이브러리를 가져옵니다:

   ```
   sudo apt-get -f 설치 libsasl2-2 sasl2-bin libsasl2-2 libsasl2-dev libsasl2-modules
   ```

2. `cmake` 옵션에 `ENABLE_MEMCACHED_SASL=1`을 추가하여 SASL 기능이 있는 `daemon_memcached` 플러그인 공유 라이브러리를 빌드하세요. 또한 memcached는 *간단한 일반 텍스트 암호 지원을* 제공하므로 테스트가 용이합니다. 간단한 일반 텍스트 암호 지원을 활성화하려면 `ENABLE_MEMCACHED_SASL_PWDB=1` `cmake` 옵션을 지정합니다.

   요약하면 다음 세 가지 `cmake` 옵션을 추가합니다:

   ```
   cmake ... -dwith_innodb_memcached=1 -denable_memcached_sasl=1 -denable_memcached_sasl_pwdb=1
   ```

3. 15.20.3절, "InnoDB 멤캐시드 플러그인 설정"에 설명된 대로 `daemon_memcached` 플러그인을 설치합니다.

4. 사용자 이름 및 비밀번호 파일을 구성합니다. (이 예에서는 멤캐시된 간단한 일반 텍스트 비밀번호 지원을 사용합니다.)

   a. 파일에서 `testname`이라는 사용자를 만들고 비밀번호를 `testpasswd`로 정의합니다:

   ```
   echo "testname:testpasswd:::::::" >/home/jy/memcached-sasl-db
   ```

   b. 멤캐시에 사용자 이름과 비밀번호 파일을 알려주도록 `MEMCACHED_SASL_PWDB` 환경 변수를 구성합니다:

```
내보내기 MEMCACHED_SASL_PWDB=/home/jy/memcached-sasl-db
```

c. 멤캐시드에 일반 텍스트 비밀번호를 사용한다고 알립니다:

```
echo "mech_list: plain" > /home/jy/work2/msasl/clients/memcached.conf
내보내기 SASL_CONF_PATH=/home/jy/work2/msasl/clients
```

5. `memcached -S` 옵션으로 인코딩된 MySQL 서버를 다시 시작하여 SASL을 활성화합니다. `daemon_memcached_option` 구성 매개변수입니다:

```
mysqld ... --daemon_memcached_option="-S"
```

6. 설정을 테스트하려면 SASL 지원 libmemcached와 같은 SASL 지원 클라이언트를 사용하세요.

```
memcp --서버=로컬호스트:11211 --binary --사용자 이름=테스트 이름
  --비밀번호=암호 내파일.txt

memcat --서버=로컬호스트:11211 --binary --사용자 이름=테스트 이름
  --비밀번호=암호 내파일.txt
```

잘못된 사용자 이름이나 비밀번호를 지정하면 멤캐시 오류 인증 실패 메시지와 함께 작업이 거부됩니다. 이 경우 `memcached-sasl-db` 파일에 설정된 일반 텍스트 암호를 검사하여 제공한 자격 증명이 올바른지 확인합니다.

멤캐시드를 사용하여 SASL 인증을 테스트하는 다른 방법도 있지만 위에서 설명한 방법이 가장 간단합니다
.

15.20.6 InnoDB 멤캐시드 플러그인용 애플리케이션 작성하기

일반적으로 `InnoDB` 멤캐시드 플러그인용 애플리케이션을 작성하려면 MySQL 또는 멤캐시드 API를 사용하는 기존 코드를 어느 정도 재작성하거나 수정해야 합니다.

- `daemon_memcached` 플러그인을 사용하면 저전력 컴퓨터에서 실행되는 기존의 많은 멤캐시드 서버 대신 디스크 스토리지와 메모리가 넉넉한 비교적 고성능 컴퓨터에서 실행되는 MySQL 서버와 동일한 수의 멤캐시드 서버를 사용할 수 있습니다. 멤캐시드 API와 함께 작동하는 일부 기존 코드를 재사용할 수도 있지만, 서버 구성이 다르기 때문에 적응이 필요할 수 있습니다.

- `daemon_memcached` 플러그인을 통해 저장된 데이터는 `VARCHAR`, `TEXT` 또는 `BLOB` 열에 저장되며, 숫자 연산을 수행하려면 변환해야 합니다. 애플리케이션 측에서 변환을 수행하거나 쿼리에서 `CAST()` 함수를 사용하여 변환할 수 있습니다.

- 데이터베이스 출신이라면 열이 많은 범용 SQL 테이블에 익숙할 수 있습니다. 멤캐시드 코드가 액세스하는 테이블에는 데이터 값을 저장하는 열이 몇 개 또는 단 하나만 있을 수 있습니다.

- 단일 행 쿼리, 삽입, 업데이트 또는 삭제를 수행하는 애플리케이션의 일부를 조정하여 코드의 중요한 섹션에서 성능을 개선할 수 있습니다. 쿼리(읽기) 및 DML(쓰기) 작업은 모두 `InnoDB` 멤캐시드 인터페이스를 통

해 수행하면 훨씬 더 빨라질 수 있습니다. 일반적으로 쓰기 성능 개선 효과가 읽기 성능 개선 효과보다 크므로 웹사이트의 로깅을 수행하거나 대화형 선택 사항을 기록하는 코드를 조정하는 데 집중할 수 있습니다.

다음 섹션에서는 이러한 사항을 자세히 살펴봅니다.

15.20.6.1 InnoDB 멤캐시드 플러그인을 위한 기존 MySQL 스키마 조정하기

기존 MySQL 스키마 또는 애플리케이션을 `daemon_memcached` 플러그인을 사용하도록 조정할 때 멤캐시드 애플리케이션의 이러한 측면을 고려하세요:

- 멤캐시드 키에는 공백이나 개행이 포함될 수 없습니다. 이러한 문자는 ASCII 프로토콜에서 구분 기호로 사용되기 때문입니다. 공백이 포함된 조회 값을 사용하는 경우 추가(), 설정(), 가져오기() 등의 호출에서 키로 사용하기 전에 공백이 없는 값으로 변환하거나 해시해야 합니다. 이론적으로는 바이너리 프로토콜을 사용하는 프로그램의 키에 이러한 문자가 허용되지만 광범위한 클라이언트와의 호환성을 보장하려면 키에 사용되는 문자를 제한해야 합니다.

- InnoDB 테이블에 짧은 숫자 기본 키 열이 있는 경우 정수를 문자열 값으로 변환하여 멤캐시에 대한 고유 조회 키로 사용하세요. 멤캐시드 서버가 여러 애플리케이션에 사용되거나 둘 이상의 InnoDB 테이블과 함께 사용되는 경우 이름을 수정하여 고유하도록 하는 것이 좋습니다. 예를 들어 숫자 값 앞에 테이블 이름 또는 데이터베이스 이름과 테이블 이름을 앞에 붙입니다.

참고

daemon_memcached 플러그인은 매핑된 맵에서 삽입 및 읽기를 지원합니다.
기본 키로 INTEGER가 정의된 InnoDB 테이블.

- 멤캐시를 사용하여 쿼리하거나 저장한 데이터에는 파티션된 테이블을 사용할 수 없습니다.

- 멤캐시드 프로토콜은 숫자 값을 문자열로 전달합니다. 예를 들어, 기본 InnoDB 테이블에 숫자 값을 저장하기 위해, SUM() 또는 AVG()와 같은 SQL 함수에서 사용할 수 있는 카운터를 구현하기 위해 사용합니다:

  - 예상되는 가장 큰 숫자의 모든 자릿수를 담을 수 있는 충분한 문자(음수 기호, 소수점 또는 둘 다에 적합한 경우 추가 문자)가 있는 VARCHAR 열을 사용합니다.

  - 열 값을 사용하여 산술을 수행하는 모든 쿼리에서 CAST() 함수를 사용하여 값을 문자열에서 정수로 또는 다른 숫자 유형으로 변환합니다. 예를 들어

```
# 알파벳 항목은 0으로 반환됩니다.

SELECT CAST(c2 as 부호 없는 정수) FROM demo_test;

# 0의 숫자 값이 있을 수 있으므로 실격 처리할 수 없습니다.
# 문자열 값을 테스트하여 정수인 값을 찾고 그 값만 평균을 냅니다.

SELECT AVG(cast(c2 as 부호 없는 정수)) FROM demo_test WHERE c2
  BETWEEN '0'과 '9999999999';

# 뷰를 사용하면 쿼리의 복잡성을 숨길 수 있습니다. 결과가 이미 변환되어 있으므로 # 매번 변환 함수와
WHERE 절을 반복할 필요가 없습니다.

CREATE VIEW numbers AS SELECT c1 KEY, CAST(c2 AS UNSIGNED INTEGER) val
  FROM demo_test WHERE c2 BETWEEN '0' and '9999999999';
SELECT SUM(val) FROM numbers;
```

참고

결과 집합의 모든 알파벳 값은 CAST() 호출에 의해 0으로 변환됩니다.

AVG()와 같은 함수를 사용할 때는

결과 집합의 행 수에 숫자가 아닌 값을 필터링하는 `WHERE` 절을 포함합니다.

- 키로 사용되는 `InnoDB` 열의 값이 250바이트보다 클 수 있는 경우 값을 250바이트 미만으로 해시합니다.
- 기존 테이블을 `daemon_memcached` 플러그인과 함께 사용하려면 `innodb_memcache.containers` 테이블에 해당 테이블에 대한 항목을 정의합니다. 해당 테이블을 모든 멤캐시드 요청에 대한 기본값으로 설정하려면 이름 열에 기본값을 지정한 다음 MySQL 서버를 다시 시작하여 변경 사항을 적용합니다. 멤캐시된 데이터의 여러 클래스에 대해 여러 개의 테이블을 사용하는 경우
  entries in the `innodb_memcache.containers` table with `name` values of your choice, then issue a `memcached` request in the form of `get @@name` or `set @@name` within the application to specify the table to be used for subsequent `memcached` requests.

미리 정의된 test.demo_test 테이블이 아닌 다른 테이블을 사용하는 예는 다음을 참조하세요. 예제 15.13, "InnoDB 멤캐시드 애플리케이션에서 자체 테이블 사용". 필요한 테이블 레이아웃은 섹션 15.20.8, "InnoDB 멤캐시드 플러그인 내부"를 참조하십시오.

- 멤캐시된 키-값 쌍으로 여러 InnoDB 테이블열 값을 사용하려면 InnoDB 테이블에 대한 innodb_memcache.containers 항목의 value_columns 필드에 쉼표, 세미콜론, 공백 또는 파이프 문자로 구분된 열 이름을 지정합니다. 예를 들어, value_columns 필드에 col1,col2,col3 또는 col1|col2|col3을 지정합니다.

  멤캐시드 추가 또는 집합 호출에 문자열을 전달하기 전에 파이프 문자를 구분 기호로 사용하여 열 값을 단일 문자열로 연결합니다. 문자열은 올바른 열에 자동으로 압축 해제됩니다. 각 get 호출은 파이프 문자로 구분된 열 값을 포함하는 단일 문자열을 반환합니다. 적절한 애플리케이션 언어 구문을 사용하여 값의 압축을 풀 수 있습니다.

예제 15.13 InnoDB 멤캐시드 애플리케이션에서 자체 테이블 사용

이 예는 다음을 사용하는 샘플 Python 애플리케이션에서 자체 테이블을 사용하는 방법을 보여줍니다. 데이터 조작을 위한 멤캐시드.

이 예제에서는 daemon_memcached 플러그인이 설명된 대로 설치되어 있다고 가정합니다. 섹션 15.20.3, "InnoDB 멤캐시 플러그인 설정". 또한 시스템이 python-memcache 모듈을 사용하는 Python 스크립트를 실행하도록 구성되어 있다고 가정합니다.

1. 인구, 면적, 운전자 측 데이터를 포함한 국가 정보를 저장하는 멀티콜 테이블을 만듭니다('R'은 오른쪽, 'L'은 왼쪽).

```
mysql> USE 테스트;

mysql> CREATE TABLE `multicol` (
        `country` varchar(128) NOT NULL DEFAULT '',
        `인구` varchar(10) 기본값 NULL,
        `area_sq_km` varchar(9) 기본값 NULL,
        `드라이브_사이드` varchar(1) 기본값 NULL,
        `c3` int(11) DEFAULT NULL,
        `c4` bigint(20) 부호 없는 DEFAULT NULL,
        `c5` int(11) DEFAULT NULL,
        기본 키 (`국가`)
        ) 엔진=InnoDB 기본 문자 집합=utf8mb4;
```

2. innodb_memcache.containers 테이블에 레코드를 삽입하여 daemon_memcached가 플러그인은 멀티콜 테이블에 액세스할 수 있습니다.

```
mysql> INSERT INTO innodb_memcache.containers
       (name,db_schema,db_table,key_columns,value_columns,flags,cas_column,expire_time_column
       ,unique_idx_name_on_key)
       가치
       ('bbb','test','multicol','country','population,area_sq_km,drive_side','c3','c4','c5','
       PRIMARY');

mysql> COMMIT;
```

   - 멀티콜 테이블에 대한 innodb_memcache.containers 레코드는 이름 값을 다음과 같이 지정합니다. 테이블 식별자인 'bbb'입니다.

참고

모든 멤캐시드 애플리케이션에 단일 `InnoDB` 테이블이 사용되는 경우, 이름 값을 기본값으로 설정하여 `@@` 표기법을 사용하여 테이블을 전환하지 않도록 할 수 있습니다.

- `db_schema` 열은 멀티콜이 저장된 데이터베이스의 이름인 `test`로 설정됩니다. 테이블이 있습니다.

- `db_table` 열은 `InnoDB` 테이블의 이름인 `multicol`로 설정됩니다.

- 키_컬럼은 고유한 국가 열로 설정됩니다. 국가 열은 다중 콜 테이블 정의에서 기본 키로 정의됩니다.
- 복합 데이터 값을 저장하기 위해 단일 InnoDB 테이블 열 대신 데이터가 세 개의 테이블 열(population, area_sq_km, drive_side)로 나뉩니다. 여러 값 열을 수용하기 위해 쉼표로 구분된 열 목록이 value_columns 필드에 지정됩니다. value_columns 필드에 정의된 열은 값을 저장하거나 검색할 때 사용되는 열입니다.
- flags, expire_time, cas_column 필드의 값은 demo.test 샘플 테이블에 사용된 값을 기반으로 합니다. 이러한 필드는 MySQL이 데이터를 동기화하므로 데이터가 만료되거나 부실해질 염려가 없으므로 일반적으로 daemon_memcached 플러그인을 사용하는 애플리케이션에서는 중요하지 않습니다.
- 고유_idx_이름_온_키 필드는 PRIMARY로 설정되며, 이는 다중 테이블의 고유 국가 열에 정의된 기본 인덱스를 참조합니다.

3. 샘플 Python 애플리케이션을 파일에 복사합니다. 이 예제에서는 샘플 스크립트를 multicol.py라는 파일에 복사합니다.

   샘플 Python 애플리케이션은 멀티콜 테이블에 데이터를 삽입하고 모든 키에 대한 데이터를 검색하며, daemon_memcached 플러그인을 통해 InnoDB 테이블에 액세스하는 방법을 보여줍니다.

```
시스템 가져오기,
OS 멤캐시 가져오기

def connect_to_memcached():
  memc = memcache.Client(['127.0.0.1:11211'], debug=0);
  print "memcached에 연결되었습니다."
  반환 MMC

def banner(message):
  print
  인쇄 "=" * len(message) 인
  쇄 메시지 인쇄
  print "=" * len(message)

country_data = [
("Canada","34820000","9984670","R"),
("USA","314242000","9826675","R"),
("Ireland","6399152","84421","L"),
("UK","62262000","243610","L"),
("Mexico","113910608","1972550","R"),
("Denmark","5543453","43094","R"),
("Norway","5002942","385252","R"),
("UAE","8264070","83600","R"),
("India","1210193422","3287263","L"),
("China","1347350000","9640821","R"),
]

def switch_table(memc,table):
  key = "@@" + table
  print "기본 테이블을 '" '" + key + "'에 대해 GET을 실행하여 '" + table + "'." result =
  memc.get(key)

def insert_country_data(memc):
  banner("멤캐시 인터페이스를 통해 초기 데이터 삽입") for item in
  country_data:
    국가 = item[0] 인구 =
    item[1] 지역 =
    item[2] 드라이브_사이드
    = item[3]

    키 = 국가
    value = "|".join([population,area,drive_side])
    print "Key = " + key
    인쇄 "값 = " + 값
```

```
    if memc.add(키,값):
      print "새 키, 값 쌍을 추가했습니다."
    else:
      print "기존 키의 값을 업데이트합니다." memc.set(키,값)

def QUERY_COUNTRY_DATA(MMC):
  banner("모든 키(국가명)에 대한 데이터 검색 중") for item in
  country_data:
    key = item[0]
    결과 = memc.get(키)
    print "다음은 키 " + 키 + ":"에 대해 데이터베이스에서 검색된 결과입니다." 결과 인쇄
    (m_population, m_area, m_drive_side) = result.split("|")
    print "언패킹된 인구 값입니다: " + m_population
    인쇄 "압축 해제된 면적 값            :        " + m_area
    print "언패킹된 드라이브 측 값입니다: " + m_drive_side if


name == ' main ':

  memc = connect_to_memcached()
  switch_table(memc,"bbb")
  insert_country_data(memc)
  query_country_data(memc)

  sys.exit(0)
```

Python 애플리케이션 노트 샘플:

- 데이터 조작은 멤캐시드 인터페이스를 통해 수행되므로 애플리케이션을 실행하는 데 데이터베이스 권한이 필요하지 않습니다. 필요한 유일한 정보는 멤캐시드 데몬이 수신 대기하는 로컬 시스템의 포트 번호뿐입니다.
- 애플리케이션이 멀티콜 테이블을 사용하도록 하기 위해 `@@` 표기법을 사용하여 더미 가져오기 또는 설정 요청을 수행하는 `switch_table()` 함수가 호출됩니다. 요청의 이름 값은 `innodb_memcache.containers.name` 필드에 정의된 멀티콜 테이블 식별자인 `bbb`입니다.

  실제 애플리케이션에서는 보다 설명적인 이름 값을 사용할 수 있습니다. 이 예에서는 `get @@...` 요청에서 테이블 이름 대신 테이블 식별자를 지정하는 것을 간단히 설명합니다.
- 데이터를 삽입하고 쿼리하는 데 사용되는 유틸리티 함수는 추가 또는 설정 요청을 통해 MySQL로 데이터를 전송하기 위해 파이썬 데이터 구조를 파이프 구분 값으로 변환하는 방법과 `get` 요청에서 반환된 파이프 구분 값의 압축을 푸는 방법을 보여 줍니다. 이 추가 처리는 단일 멤캐시드 값을 여러 MySQL 테이블 열에 매핑할 때만 필요합니다.

4. 샘플 Python 애플리케이션을 실행합니다.

```
파이썬 멀티콜.py
```

성공하면 샘플 애플리케이션이 이 출력을 반환합니다:

```
멤캐시드에 연결되었습니다.
'@@bbb'에 대해 GET을 실행하여 기본 테이블을 'bbb'로 전환합니다.

==============================================
멤캐시드 인터페이스를 통해 초기 데이터 삽입하기
==============================================
키 = 캐나다
값 = 34820000|9984670|R
새 키, 값 쌍이 추가되었습니다. 키
= USA
값 = 314242000|9826675|R
새 키, 값 쌍이 추가되었습니다. 키
= 아일랜드
값 = 6399152|84421|L
새 키, 값 쌍을 추가했습니다.
```

```
키 = 영국
값 = 62262000|243610|L
새 키, 값 쌍이 추가되었습니다. 키
= 멕시코
값 = 113910608|1972550|R
새 키, 값 쌍이 추가되었습니다. 키
= 덴마크
값 = 5543453|43094|R
새 키, 값 쌍이 추가되었습니다. 키
= 노르웨이
값 = 5002942|385252|R
새 키, 값 쌍이 추가되었습니다. 키
= UAE
값 = 8264070|83600|R
새 키, 값 쌍이 추가되었습니다. 키
= 인도
값 = 1210193422|3287263|L
새 키, 값 쌍이 추가되었습니다. 키
= 중국
값 = 1347350000|9640821|R
새 키, 값 쌍을 추가했습니다.

==========================================
모든 키에 대한 데이터 검색(국가 이름)
==========================================
다음은 주요 캐나다에 대한 데이터베이스에서 검색된 결과입니다: 34820000|9984670|R
포장 해제된 인구 값: 34820000 포장 해제된
면적 값                    9984670 포
장 해제된 드라이브 측 값: R
다음은 key USA에 대해 데이터베이스에서 검색된 결과입니다:
314242000|9826675|R
압축 해제된 인구 값: 314242000 압축 해제된
면적 값                    9826675 압축
해제된 드라이브 측 값: R
다음은 아일랜드 키에 대해 데이터베이스에서 검색된 결과입니다: 6399152|84421|L
포장 해제된 인구 값:6399152 포장 해제된
면적 값                    84421 포장
해제된 드라이브 측 값: L
다음은 주요 영국에 대한 데이터베이스에서 검색된 결과입니다:
62262000|243610|L
포장 해제된 인구 값:62262000 포장 해제된
면적 값                    243610 포장
해제된 드라이브 측 값: L
다음은 주요 멕시코에 대한 데이터베이스에서 검색된 결과입니다: 113910608|1972550|R
포장 해제된 인구 값:113910608 포장 해제된
면적 값                    1972550 포장
해제된 드라이브 측 값: R
다음은 주요 덴마크에 대한 데이터베이스에서 검색된 결과입니다: 5543453|43094|R
언패킹된 인구 값:5543453 언패킹된 면적
```

값 : 43094

압축 해제된 드라이브 측 값: R

다음은 주요 노르웨이에 대해 데이터베이스에서 검색된 결과입니다: 5002942|385252|R

포장 해제된 인구 값:5002942 포장 해제된

면적 값 : 385252

압축 해제된 드라이브 측 값: R

다음은 주요 아랍에미리트에 대한 데이터베이스에서 검색된 결과입니다: 8264070|83600|R

포장 해제된 인구 값:면적 값 83600

포장 해제된 드라이브 측 값: R

다음은 주요 인도에 대한 데이터베이스에서 검색된 결과입니다: 1210193422|3287263|L

포장 해제된 인구 값:1210193422 포장 해제된

면적 값 : 3287263 언

패킹된 드라이브 사이드 값: L

다음은 주요 중국에 대한 데이터베이스에서 검색된 결과입니다: 1347350000|9640821|R

```
포장 해제된 인구 값:1347350000 포장 해제된
면적 값                    9640821 포장
해제된 드라이브 측 값: R
```

5. innodb_memcache.containers 테이블을 쿼리하여 앞서 멀티콜 테이블에 삽입한 레코드를 확인합니다. 첫 번째 레코드는 초기 daemon_memcached 플러그인 설정 중에 생성되는 demo_test 테이블의 샘플 항목입니다. 두 번째 레코드는 멀티콜 테이블에 삽입한 항목입니다.

```
mysql> SELECT * FROM innodb_memcache.containers\G
*************************** 1. 행 *************************** 이
                  름: aaa
            DB_SCHEMA: 테스트
             DB_TABLE: 데모 테스트
          KEY_COLUMNS: C1
        VALUE_COLUMNS: C2
                플래그: C3
           CAS_COLUMN: C4
    EXPIRE_TIME_COLUMN: C5
고유_IDX_NAME_ON_KEY: PRIMARY
*************************** 2. 행 *************************** 이
                  름: bbb
            DB_SCHEMA: 테스트
             DB_TABLE: 멀티콜
          key_columns: 국가
        VALUE_COLUMNS: POPULATION, AREA_SQ_KM, DRIVE_SIDE
                FLAGS: C3
           CAS_COLUMN: C4
    EXPEAL_TIME_COLUMN: C5
고유 IDX NAME ON KEY: PRIMARY
```

6. 멀티콜 테이블을 쿼리하여 샘플 Python 애플리케이션에서 삽입된 데이터를 확인합니다. 이 데이터는 MySQL 쿼리에 사용할 수 있으며, SQL을 사용하거나 애플리케이션을 통해(적절한 MySQL 커넥터 또는 API를 사용하여) 동일한 데이터에 액세스하는 방법을 보여줍니다.

```
mysql> SELECT * FROM test.multicol;
+---------+------------+------------+------------+------+------+ -------------+
| 국가 | 인구 | 면적_평방킬로미터 | 드라이브_측면 | C3| C4| C5   |
+---------+------------+------------+------------+------+------+ -------------+
| 캐나다    | 34820000   | 9984670    | R          |    0 |   11 |    0 |
| 중국      | 1347350000 | 9640821    | R          |    0 |   20 |    0 |
| 덴마크    | 5543453    | 43094      | R          |    0 |   16 |    0 |
| 인도      | 1210193422 | 3287263    | L          |    0 |   19 |    0 |
| 아일랜드  | 6399152    | 84421      | L          |    0 |   13 |    0 |
| 멕시코    | 113910608  | 1972550    | R          |    0 |   15 |    0 |
| 노르웨이  | 5002942    | 385252     | R          |    0 |   17 |    0 |
| UAE       | 8264070    | 83600      | R          |    0 |   18 |    0 |
| UK        | 62262000   | 243610     | L          |    0 |   14 |    0 |
| 미국|    314242000 | 9826675| r                    |0   |12 |0 |
                                                -----+------+------+---------+
```

참고

숫자로 처리되는 열의 길이를 정의할 때는 항상 필요한 숫자, 소수점, 부호 문자, 선행 0 등을 포함할 수 있는 충분한 크기를 허용해야 합니다. VARCHAR와 같은 문자열 열의 값이 너무 길면 일부 문자가 제거되어 잘리게 되므로 무의미한 숫자 값이 생성될 수 있습니다.

7. 선택 사항으로 멤캐시된 데이터를 저장하는 InnoDB 테이블에서 보고서형 쿼리를 실행합니다.

   국가 키 열뿐만 아니라 모든 열에서 계산 및 테스트를 수행하여 SQL 쿼리를 통해 보고서를 생성할 수 있습니다. (다음 예제에서는 일부 국가의 데이터만 사용하므로 숫자는 설명용으로만 사용됩니다.) 다음 쿼리는 사람들이 오른쪽으로 운전하는 국가의 평균 인구와 이름이 "U"로 시작하는 국가의 평균 규모를 반환합니다:

```
mysql> SELECT AVG(population) FROM multicol WHERE drive_side = 'R';
```

```
+-------------------+
| 평균(인구)          |
+-------------------+
| 261304724.7142857 |
+-------------------+

mysql> SELECT SUM(area_sq_km) FROM multicol WHERE country LIKE 'U%';
+-----------------+
| 합계(area_sq_km) |
+-----------------+
         |10153885 |
+-----------------+
```

인구 및 area_sq_km 열은 강력한 형식의 숫자 데이터가 아닌 문자 데이터를 저장하므로 AVG() 및 SUM() 같은 함수는 각 값을 다음과 같이 변환하여 작동합니다.

먼저 숫자를 입력합니다. 이 접근 방식은 문자 기반 값을 비교할 때 < 또는 >와 같은 연산자(예: 9 > 1000)에는 *작동하지 않으며, 이는* ORDER BY population DESC와 같은 절에서 예상되지 않습니다. 가장 정확한 유형 처리를 위해 숫자 열을 적절한 유형으로 캐스팅하는 뷰에 대해 쿼리를 수행합니다. 이 기술을 사용하면 데이터베이스 응용 프로그램에서 간단한 SELECT * 쿼리를 실행하면서 형 변환, 필터링 및 순서가 올바른지 확인할 수 있습니다. 다음 예에서는 인구 내림차순으로 상위 3개 국가를 찾기 위해 쿼리할 수 있는 뷰를 보여 주며, 결과는 다중_열 테이블의 최신 데이터를 반영하고 인구 및 면적 수치는 숫자로 처리됩니다.

```
mysql> CREATE VIEW populous_countries AS
       SELECT
       국가,
       cast(population as 부호 없는 정수) population,
       cast(area_sq_km as 부호 없는 정수) area_sq_km,
       drive_side FROM multicol
       ORDER BY CAST(인구는 부호 없는 정수로) DESC LIMIT 3;

mysql> SELECT * FROM populous_countries;
+---------+------------+------------+ ----------------+
| 국가 | 인구 | 면적_sq_km | 드라이브_측면 |
+---------+------------+------------+ ----------------+
| 중국|    1347350000 |9640821                      | R|
| 인도|    1210193422 |3287263                      | L|
|         미국| 314242000 |9826675                  | r|
+---------+------------+------------+ ----------------+

mysql> DESC 인구수_국가;
+------------+---------------------+------+-----+---------+-------------+
| 필드| 유형|  Null | 키 | 기본값 | 추가 |
+------------+---------------------+------+-----+---------+-------------+
| 국가         | varchar(128)        | 아니요 |     |         |        |
| 인구         | bigint(10) 부호 없음  | 예   |     | NULL    |        |
| area_sq_km | int(9) 부호 없음      | 예   |     | NULL    |        |
| 드라이브_측면 | varchar(1)          | YES || NULL |         |        |
+------------+---------------------+------+-----+---------+-------------+
```

15.20.6.2 InnoDB 멤캐시드 플러그인을 위한 멤캐시드 애플리케이션 조정하기

기존 멤캐시드 애플리케이션을 daemon_memcached 플러그인을 사용하도록 조정할 때 MySQL 및 InnoDB 테이블의 이러한 측면을 고려하세요:

- 몇 바이트보다 긴 키 값이 있는 경우 숫자 자동 증가 열을 InnoDB 테이블의 기본 키로 사용하고 멤캐시된 키 값이 포함된 열에 고유한 보조 인덱스를 생성하는 것이 더 효율적일 수 있습니다. 이는 자동 증가

값과 마찬가지로 기본 키 값이 정렬된 순서대로 추가되는 경우 InnoDB가 대규모 삽입에 가장 적합한 성능을 발휘하기 때문입니다. 기본 키 값은 보조 인덱스에 포함되므로 기본 키가 긴 문자열 값인 경우 불필요한 공간을 차지하게 됩니다.

- 멤캐시를 사용하여 여러 종류의 정보를 저장하는 경우 각 데이터 유형에 대해 별도의 InnoDB 테이블을 설정하는 것이 좋습니다. 추가 테이블 식별자를 innodb_memcache.containers 테이블을 저장하고 *@@table_id.key* 표기법을 사용하여 저장 및

다른 테이블에서 항목을 검색합니다. 서로 다른 유형의 정보를 물리적으로 나누면 최적의 공간 활용도, 성능 및 안정성을 위해 각 테이블의 특성을 조정할 수 있습니다. 예를 들어 블로그 게시물이 포함된 테이블에는 압축을 사용하도록 설정하지만 썸네일 이미지가 포함된 테이블에는 압축을 사용하지 않을 수 있습니다. 한 테이블에 중요한 데이터가 포함되어 있기 때문에 다른 테이블보다 더 자주 백업할 수 있습니다. SQL을 사용하여 보고서를 생성하는 데 자주 사용되는 테이블에 보조 인덱스를 추가로 만들 수 있습니다.

- 가급적이면 `daemon_memcached` 플러그인과 함께 사용할 안정적인 테이블 정의 집합을 구성하고 테이블을 영구적으로 제자리에 두는 것이 좋습니다. `innodb_memcache.containers` 테이블에 대한 변경 사항은 다음에 `innodb_memcache.containers` 테이블을 쿼리할 때 적용됩니다. 컨테이너 테이블의 항목은 시작 시 처리되며, `@@` 표기법을 사용하여 인식할 수 없는 테이블 식별자(`containers.name`에 정의된 대로)가 요청될 때마다 참조됩니다. 따라서 새 항목은 연결된 테이블 식별자를 사용하는 즉시 표시되지만 기존 항목에 대한 변경 사항은 서버를 다시 시작해야 적용됩니다.

- 기본 `innodb_only` 캐싱 정책을 사용하면 `add()`, `set()`, `incr()` 등의 호출은 성공할 수 있지만 `'STORED'`를 예상하는 동안 예기치 않은 응답 `'NOT_STORED'`를 받는 등의 디버깅 메시지가 계속 트리거될 수 있습니다. 디버그 메시지는 `innodb_only` 캐싱 정책으로 인해 새 값과 업데이트된 값이 메모리 캐시에 저장되지 않고 `InnoDB` 테이블로 직접 전송되기 때문에 발생합니다.

15.20.6.3 InnoDB 멤캐시드 플러그인 성능 튜닝

멤캐시와 함께 `InnoDB`를 사용하면 즉시 또는 나중에 모든 데이터를 디스크에 쓰기 때문에, 원시 성능은 멤캐시드만 사용하는 것보다 다소 느려질 것으로 예상됩니다. `InnoDB` 멤캐시드 플러그인을 사용하는 경우, 멤캐시드 작업의 튜닝 목표는 동등한 SQL 작업보다 더 나은 성능을 달성하는 데 중점을 두어야 합니다.

벤치마크에 따르면 멤캐시드 인터페이스를 사용하는 쿼리 및 DML 작업(삽입, 업데이트, 삭제)이 기존 SQL보다 더 빠른 것으로 나타났습니다. DML 작업은 일반적으로 더 큰 개선 효과를 볼 수 있습니다. 따라서 쓰기 집약적인 애플리케이션을 먼저 멤캐시드 인터페이스를 사용하도록 조정하는 것을 고려하세요. 또한 안정성이 부족한 빠르고 가벼운 메커니즘을 사용하는 쓰기 집약적인 애플리케이션을 우선적으로 적용하는 것도 고려하세요.

SQL 쿼리 조정

간단한 `GET` 요청에 가장 적합한 쿼리 유형은 단일 절 또는 `WHERE` 절에 `AND` 조건 집합이 있는 쿼리입니다:

```
SQL:
SELECT col FROM tbl WHERE key = 'key_value';

멤캐시드:
키_값 가져오기

SQL:
SELECT col FROM tbl WHERE col1 = val1 및 col2 = val2 및 col3 = val3;

멤캐시드:
# 키를 조회하려면 항상 이 세 가지 값을 알고 있어야 하므로 # 고유한 문자열로
결합하여 키로 사용합니다.
# 키_값 = val1 + ":" + val2 + ":" + val3
get key_value

SQL:
SELECT '키가 존재합니다!' FROM tbl
  WHERE EXISTS (SELECT col1 FROM tbl WHERE KEY = 'key_value') LIMIT 1;

멤캐시드:
# 값을 요청하고 호출이 성공했는지 확인하여 키의 존재를 테스트하고, # 값 자체는 무시합니다. 존재 확인을 위
해 일반적으로 매우 작은
# "1"과 같은 짧은 값입니다. 키_값
가져오기
```

시스템 메모리 사용

최상의 성능을 얻으려면, 대부분의 시스템 RAM이 InnoDB 버퍼 풀에 할당되는 일반적인 데이터베이스 서버로 구성된 시스템에서 innodb_buffer_pool_size 구성 옵션을 통해 daemon_memcached 플러그인을 배포하세요. 멀티 기가바이트 버퍼 풀이 있는 시스템의 경우, 대부분의 작업에 이미 메모리에 캐시된 데이터가 포함되는 경우 처리량을 최대화하기 위해 innodb_buffer_pool_instances 값을 높이는 것을 고려하세요.

중복 I/O 감소

InnoDB에는 충돌 시 높은 안정성과 쓰기 워크로드가 많은 경우의 I/O 오버헤드 사이의 균형을 선택할 수 있는 여러 가지 설정이 있습니다. 예를 들어, innodb_doublewrite를 0으로 설정하고 innodb_flush_log_at_trx_commit을 2로 설정하는 것을 고려해 보십시오. 다른 innodb_flush_method 설정으로 성능을 측정해 보세요.

테이블 작업의 I/O를 줄이거나 조정하는 다른 방법은 섹션 8.5.8, 'InnoDB 디스크 I/O 최적화'를 참조하세요.

트랜잭션 오버헤드 감소

daemon_memcached_r_batch_size 및 daemon_memcached_w_batch_size의 기본값 1은 결과의 안정성과 저장 또는 업데이트된 데이터의 안전성을 극대화하기 위한 것입니다.

애플리케이션 유형에 따라 이 설정 중 하나 또는 둘 모두를 증가시켜 잦은 커밋 작업으로 인한 오버헤드를 줄일 수 있습니다. 사용량이 많은 시스템에서는 SQL을 통해 수행된 데이터 변경 사항이 즉시(즉, *N개의* get 작업이 더 처리될 때까지) memcached에 표시되지 않을 수 있다는 것을 알고 daemon_memcached_r_batch_size를 늘릴 수 있습니다. 모든 쓰기 작업이 안정적으로 저장되어야 하는 데이터를 처리할 때는 DAEMON_MCACHED_W_BATCH_SIZE를 1로 설정합니다. 통계 분석만을 목적으로 하는 대량의 업데이트를 처리하는 경우, 예기치 않은 종료로 마지막 *N개의* 업데이트가 손실되는 것이 허용 가능한 위험이라면 이 설정을 높입니다.

예를 들어, 매일 약 100,000대의 차량에 대한 데이터를 기록하면서 혼잡한 다리를 건너는 교통량을 모니터링하는 시스템을 상상해 보세요. 애플리케이션이 교통 패턴을 분석하기 위해 다양한 유형의 차량을 카운트하는 경우, daemon_memcached_w_batch_size를 1에서 100으로 변경하면 커밋 작업에 대한 I/O 오버헤드가 99% 감소합니다. 중단이 발생하는 경우 최대 100개의 레코드가 손실되며, 이는 허용 가능한 오차 범위일 수 있습니다. 대신 애플리케이션이 자동화된 톨링 작업을 수행하는 경우 컬렉션의 경우, 각 통행료 기록이 즉시 디스크에 저장되도록 daemon_memcached_w_batch_size를 1로 설정할 수 있습니다.

InnoDB가 디스크에서 멤캐시된 키 값을 구성하는 방식 때문에 생성할 키의 수가 많은 경우 임의의 순서로 키를 생성하는 것보다 애플리케이션에서 키 값별로 데이터 항목을 정렬하고 정렬된 순서대로 추가하는 것이 더 빠를 수 있습니다.

일반 `memcached` 배포판의 일부이지만 `daemon_memcached` 플러그인에는 포함되지 않은 `memslap` 명령은 다양한 구성을 벤치마킹하는 데 유용할 수 있습니다. 또한 자체 벤치마크에 사용할 샘플 키-값 쌍을 생성하는 데에도 사용할 수 있습니다.

15.20.6.4 InnoDB 멤캐시드 플러그인의 트랜잭션 동작 제어하기

기존 memcached와 달리 `daemon_memcached` 플러그인을 사용하면 추가, 설정, `incr` 등의 호출을 통해 생성된 데이터 값의 지속성을 제어할 수 있습니다. 기본적으로 멤캐시드 인터페이스를 통해 쓰여진 데이터는 디스크에 저장되며, 가져오기 호출은 디스크에서 가장 최근 값을 반환합니다.
기본 동작이 최상의 원시 성능을 제공하지는 않지만 `InnoDB` 테이블의 SQL 인터페이스와 비교하면 여전히 빠릅니다.

`daemon_memcached` 플러그인 사용 경험이 쌓이면 가동 중단 시 일부 업데이트된 값이 손실되거나 약간 오래된 데이터가 반환될 수 있으므로 중요하지 않은 데이터 클래스에 대한 내구성 설정을 완화하는 것을 고려할 수 있습니다.

커밋 빈도

내구성과 원시 성능 사이의 한 가지 절충점은 새 데이터와 변경된 데이터가 얼마나 자주 커밋되는지입니다. 데이터가 중요한 경우, 즉시 커밋하여 장애 발생 시에도 안전하도록 해야 합니다.
예기치 않은 종료 또는 중단. 예기치 않은 종료 후 재설정되는 카운터나 손실될 수 있는 로깅 데이터와 같이 데이터가 덜 중요한 경우, 커밋 빈도가 낮은 원시 처리량을 더 선호할 수 있습니다.

멤캐시드 작업이 기본 InnoDB 테이블에 데이터를 삽입, 업데이트 또는 삭제할 때, 변경 사항은 즉시 (daemon_memcached_w_batch_size=1인 경우) 또는 일정 시간이 지난 후 (daemon_memcached_w_batch_size 값이 더 큰 경우) InnoDB 테이블에 커밋될 수 있습니다. 보다 큽니다). 두 경우 모두 변경 사항을 롤백할 수 없습니다. 바쁜 시간 동안 높은 I/O 오버헤드를 피하기 위해 daemon_memcached_w_batch_size 값을 늘리면 워크로드가 감소할 때 커밋이 자주 발생하지 않을 수 있습니다. 안전 조치로 백그라운드 스레드는 멤캐시드 API를 통해 변경된 내용을 일정한 간격으로 자동으로 커밋합니다. 이 간격은 기본 설정이 5초인 innodb_api_bk_commit_interval 구성 옵션으로 제어됩니다.

멤캐시드 작업이 기본 InnoDB 테이블에 데이터를 삽입하거나 업데이트할 때, 변경된 데이터는 MySQL 측에서 아직 커밋되지 않았더라도 메모리 캐시에 새 값이 남아 있기 때문에 다른 멤캐시드 요청에서 즉시 볼 수 있습니다.

트랜잭션 격리

get 또는 incr과 같은 멤캐시드 작업으로 인해 기본 InnoDB 테이블에 대한 쿼리 또는 DML 작업이 발생하는 경우, 작업에서 테이블에 기록된 최신 데이터를 볼지, 커밋된 데이터만 볼지 또는 다른 다양한 트랜잭션 격리 수준을 제어할 수 있습니다. 이 기능을 제어하려면 innodb_api_trx_level 구성 옵션을 사용합니다. 이 옵션에 지정된 숫자 값은 반복 읽기 등의 격리 수준에 해당합니다. 다른 설정에 대한 자세한 내용은 innodb_api_trx_level 옵션 설명을 참조하세요.

엄격한 격리 수준은 검색한 데이터가 갑자기 롤백되거나 변경되어 후속 쿼리에서 다른 값이 반환되는 것을 방지합니다. 그러나 엄격한 격리 수준은 더 많은 잠금 오버헤드가 필요하므로 대기 시간이 발생할 수 있습니다. 장기 실행 트랜잭션을 사용하지 않는 NoSQL 스타일 애플리케이션의 경우 일반적으로 기본 격리 수준을 사용하거나 덜 엄격한 격리 수준으로 전환할 수 있습니다.

멤캐시드 DML 연산에 행 잠금 비활성화하기

다음과 같은 경우 innodb_api_disable_rowlock 옵션을 사용하여 행 잠금을 비활성화할 수 있습니다. 멤캐시드 요청은 daemon_memcached 플러그인을 통해 DML 작업을 유발합니다. 기본적으로 innodb_api_disable_rowlock은 OFF로 설정되어 있으며, 이는 멤캐시가 get 및 set 연산에 대해 행 잠금을 요청한다는 의미입니다. innodb_api_disable_rowlock을 ON으로 설정하면 memcached는 행 잠금 대신 테이블 잠금을 요청합니다.

innodb_api_disable_rowlock 옵션은 동적이지 않습니다. 이 옵션은 시작 시 mysqld 명령줄에 입력하거나 MySQL 구성 파일에 입력합니다.

DDL 허용 또는 허용 안 함

기본적으로 daemon_memcached 플러그인에서 사용하는 테이블에 대해 ALTER TABLE과 같은 DDL 작업을 수행할 수 있습니다. 이러한 테이블이 다음과 같은 경우 잠재적인 속도 저하를 방지하려면
처리량이 많은 애플리케이션에 사용되는 경우, 시작할 때 innodb_api_enable_mdl을 활성화하여 이러한 테이블에 대한 DDL 작업을 비활성화합니다. 이 옵션은 보고 쿼리를 실행하는 데 중요할 수 있는 테이블의 CREATE INDEX 문을 차단하기 때문에 memcached와 SQL을 통해 동일한 테이블에 액세스하는 경우에는 적합하지 않습니다.

디스크, 메모리 또는 둘 다에 데이터 저장

innodb_memcache.cache_정책 테이블은 멤캐시드 인터페이스를 통해 쓰여진 데이터를 디스크에 저장할지(기본값인 innodb_only), 기존 멤캐시드와 같이 메모리에만 저장할지(cache_only), 아니면 둘 다 저장할지(캐싱)를 지정합니다.

캐싱 설정을 사용하면 memcached가 메모리에서 키를 찾을 수 없는 경우 InnoDB 테이블에서 값을 검색합니다. 캐싱 설정에 따라 get 호출에서 반환된 값은 InnoDB 테이블의 디스크에서 업데이트되었지만 메모리 캐시에서 아직 만료되지 않은 경우 오래된 값일 수 있습니다.

캐싱 정책은 가져오기, 설정(incr 및 decr 포함), 삭제, 삭제에 대해 독립적으로 설정할 수 있습니다. 플러시 작업.

예를 들어, 테이블과 멤캐시된 메모리 캐시를 동시에 쿼리하거나 업데이트하는 get 및 set 작업을 허용하고(캐싱 설정 사용), 삭제, 플러시 또는 둘 다 인메모리 복사본에서만 작동하도록 할 수 있습니다(cache_only 설정 사용). 이렇게 하면 항목을 삭제하거나 플러시하면 캐시에서 해당 항목만 만료되고 다음에 해당 항목이 요청될 때 InnoDB 테이블에서 최신 값이 반환됩니다.

```
mysql> SELECT * FROM innodb_memcache.cache_정책;
+--------------+------------+------------+--------------+--------------------+
| 정책_이름 | 정책 가져오기 | 정 책 설정하기 | 정 책 삭제하기 | 정책 플러시하기 | 정책 삭제하기
+--------------+------------+------------+--------------+--------------------+
캐시 정책 | innodb_only | innodb_only | innodb_only | innodb_only|    innodb_only |
+--------------+------------+------------+--------------+--------------------+

mysql> UPDATE innodb_memcache.cache_policies SET set_policy = 'caching'
       WHERE policy_name = 'cache_정책';
```

innodb_memcache.cache_policies 값은 시작할 때만 읽습니다. 이 표의 값을 변경한 후 daemon_memcached 플러그인을 제거했다가 다시 설치하여 변경 사항이 적용되도록 합니다.

```
mysql> UNINSTALL PLUGIN daemon_memcached;

mysql> INSTALL PLUGIN daemon_memcached soname "libmemcached.so";
```

15.20.6.5 멤캐시드 연산에 DML 문 적용하기

벤치마크에 따르면 daemon_memcached 플러그인은 쿼리 속도보다 DML 작업(삽입, 업데이트, 삭제) 속도가 더 빠른 것으로 나타났습니다. 따라서 초기 개발 작업은 쓰기 집약적인 애플리케이션에 집중하는 것이 좋으며, 새로운 쓰기 집약적인 애플리케이션에는 daemon_memcached 플러그인과 함께 MySQL을 사용할 수 있는 기회를 찾아보세요.

단일 행 DML 문은 멤캐시드 연산으로 전환하기 가장 쉬운 유형의 문입니다. INSERT는 add가 되고, UPDATE는 set, incr 또는 decr이 되며, DELETE는 삭제가 됩니다. 이러한 연산은 테이블 내에서 *키가* 고유하기 때문에 멤캐시드 인터페이스를 통해 실행될 때 하나의 행에만 영향을 미치도록 보장됩니다.

다음 SQL 예제에서 t1은 innodb_memcache.containers 테이블의 구성을 기반으로 멤캐시드 작업에 사용되는 테이블을 참조합니다. key는 key_columns 아래에 나열된 열을 참조하고 val은 value_columns 아래에 나열된 열을 참조합니다.

```
INSERT INTO t1 (key,val) VALUES (some_key,some_value);
SELECT val FROM t1 WHERE key = some_key;
UPDATE t1 SET val = new_value WHERE key = some_key;
UPDATE t1 SET val = val + x WHERE key = some_key;
DELETE FROM t1 WHERE key = some_key;
```

테이블에서 모든 행을 제거하는 다음 TRUNCATE TABLE 및 DELETE 문은 이전 예제에서와 같이 t1이 멤캐시드 연산을 위한 테이블로 구성된 flush_all 연산에 해당합니다.

```
테이블 잘라내기 t1;
DELETE FROM t1;
```

15.20.6.6 기본 InnoDB 테이블에 대한 DML 및 DDL 문 수행

표준 SQL 인터페이스를 통해 기본 `InnoDB` 테이블(기본값은 `test.demo_test`)에 액세스할 수 있습니다. 하지만 몇 가지 제한 사항이 있습니다:

- 멤캐시드 인터페이스를 통해서도 액세스되는 테이블을 쿼리할 때는 다음 사항을 기억하세요.
  멤캐시드 작업은 매 쓰기 후가 아닌 주기적으로 커밋되도록 구성할 수 있습니다.

작업을 수행합니다. 이 동작은 daemon_memcached_w_batch_size 옵션에 의해 제어됩니다. 이 옵션이 1보다 큰 값으로 설정된 경우 READ UNCOMMITTED 쿼리를 사용하여 방금 삽입된 행을 찾습니다.

```
mysql> SET SESSSION 트랜잭션 격리 수준 읽기 미완료;

mysql> SELECT * FROM demo_test;
+------+------+------+------+-----------+------+------+------+------+------+ -------------+
| CX| CY| C1| CZ| C2| CA| CB| C3| CU| C4| C5      |
+------+------+------+------+-----------+------+------+------+------+------+ -------------+
| NULL | NULL | a11 | NULL | 123456789 | NULL | NULL | NULL |10           | NULL |3         |
NULL |
+------+------+------+------+-----------+------+------+------+------+------+ -------------+
```

- 멤캐시드 인터페이스를 통해 액세스하는 SQL을 사용하여 테이블을 수정하는 경우, 모든 읽기 작업이 아니라 주기적으로 새 트랜잭션을 시작하도록 멤캐시드 작업을 구성할 수 있습니다. 이 동작은 daemon_memcached_r_batch_size 옵션으로 제어됩니다. 이 옵션을 1보다 큰 값으로 설정하면 SQL을 사용하여 테이블을 변경해도 멤캐시드 작업에 즉시 표시되지 않습니다.

- InnoDB 테이블은 트랜잭션의 모든 작업에 대해 IS(인텐션 공유) 또는 IX(인텐션 독점)로 잠겨 있습니다. daemon_memcached_r_batch_size 및 DAEMON_MEMCACHED_W_BATCH_SIZE를 기본값인 1보다 크게 설정하면 각 작업 사이에 테이블이 잠겨 테이블에 대한 DDL 문을 사용할 수 없을 가능성이 높습니다.

15.20.7 InnoDB 멤캐시드 플러그인 및 복제

daemon_memcached 플러그인은 MySQL 바이너리 로그를 지원하므로, 멤캐시드 인터페이스를 통해 소스 서버를 복제하여 백업, 집중적인 읽기 워크로드 분산, 고가용성 등을 수행할 수 있습니다. 모든 memcached 명령은 바이너리 로깅과 함께 지원됩니다.

복제본 서버에서 daemon_memcached 플러그인을 설정할 필요는 없습니다. 이 구성의 가장 큰 장점은 원본의 쓰기 처리량이 증가한다는 것입니다. 복제 메커니즘의 속도는 영향을 받지 않습니다.

다음 섹션에서는 daemon_memcached를 사용할 때 바이너리 로그 기능을 사용하는 방법을 보여줍니다. 플러그인을 사용해야 합니다. 섹션 15.20.3, "InnoDB 멤캐시드 플러그인 설정하기"에 설명된 설정을 완료한 것으로 가정합니다.

InnoDB 멤캐시드 바이너리 로그 활성화

1. MySQL 바이너리 로그와 함께 daemon_memcached 플러그인을 사용하려면 소스 서버에서 innodb_api_enable_binlog 구성 옵션을 활성화합니다. 이 옵션은 서버를 시작할 때만 설정할 수 있습니다. 또한 소스 서버에서 MySQL 바이너리 로그를 사용하도록 설정하려면 --로그빈 옵션을 추가합니다. 이러한 옵션은 MySQL 구성 파일에 추가하거나 mysqld에 추가할 수 있습니다. 명령줄을 입력합니다.

   ```
   mysqld ... --log-bin --innodb_api_enable_binlog=1
   ```

2. 17.1.2절, "바이너리 로그 파일 위치 기반 복제 설정"에 설명된 대로 소스 및 복제본 서버를 구성합니다.

3. mysqldump를사용하여 소스 데이터 스냅샷을 만들고 복제본 서버에 스냅샷을 동기화합니다.

   ```
   source $> mysqldump --all-databases --lock-all-tables > dbdump.db
   복제본 $> mysql < dbdump.db
   ```

4. 소스 서버에서 `SHOW BINARY LOG STATUS`를 실행하여 소스 바이너리 로그 좌표를 얻습니다.

```
mysql> SHOW BINARY LOG STATUS;
```

5. 복제본 서버에서 `CHANGE REPLICATION SOURCE TO` 문을 사용하여 원본 바이너리 로그 좌표를 사용하여 복제본 서버를 설정합니다.

```
mysql> 복제 원본을 다음으로 변경합니다.
       SOURCE_HOST='localhost',
       SOURCE_USER='root',
       SOURCE_PASSWORD='',
       SOURCE_PORT = 13000,
       SOURCE_LOG_FILE='0.000001,
       SOURCE_LOG_POS=114;
```

6. 복제본을 시작합니다.

```
mysql> 복제 시작;
```

오류 로그에 다음과 유사한 출력이 인쇄되면 복제본이 복제할 준비가 된 것입니다.

```
2013-09-24T13:04:38.639684Z 49 [참고] 복제 I/O 스레드: 소스
'root@localhost:13000'에 연결됨, 로그 '0.000001'에서 복제 시작됨
114번 위치
```

InnoDB 멤캐시드 복제 구성 테스트

이 예제에서는 멤캐시드 및 텔넷을 사용하여 데이터를 삽입, 업데이트 및 삭제하기 위해 InnoDB 멤캐시드 복제 구성을 테스트하는 방법을 보여 줍니다. 소스 및 복제본 서버에서 결과를 확인하기 위해 MySQL 클라이언트가 사용됩니다.

이 예제에서는 demo_test 테이블을 사용하는데, 이 테이블은 daemon_memcached 플러그인을 초기 설정하는 동안 innodb_memcached_config.sql 구성 스크립트에 의해 생성되었습니다. demo_test 테이블에는 단일 예제 레코드가 포함되어 있습니다.

1. set 명령을 사용하여 키가 test1, 플래그 값이 10, 만료 값이 0, cas 값이 1, 값이 t1인 레코드를 삽입합니다.

```
텔넷 127.0.0.1 11211
127.0.0.1 시도 중...
127.0.0.1에 연결되었습니다. 이
스케이프 문자는 '^]'입니다. set
test1 10 0 1
t1
저장됨
```

2. 소스 서버에서 레코드가 demo_test 테이블에 삽입되었는지 확인합니다. demo_test 테이블이 이전에 수정되지 않았다고 가정하면 두 개의 레코드가 있어야 합니다. 키가 AA인 예제 레코드와 키가 test1인 방금 삽입한 레코드가 있습니다. c1 열은 키에, c2 열은 값에, c3 열은 플래그 값에, c4 열은cas 값에, c5 열은만료 시간에 매핑됩니다. 만료 시간은 사용되지 않으므로 0으로 설정되었습니다.

```
mysql> SELECT * FROM test.demo_test;
+-------+--------------+------+------+ ----------+
| C1| C2| C3| C4| C5    |
+-------+--------------+------+------+ ----------+
| aa|   여보세요, 여보세요 |    8 |    0 |    0 |
| test1 | t1              |10 |1 |0 |
+-------+--------------+------+------+ ----------+
```

3. 동일한 레코드가 복제본 서버에 복제되었는지 확인합니다.

```
mysql> SELECT * FROM test.demo_test;
+-------+--------------+------+------+ ----------+
| C1| C2| C3| C4| C5    |
+-------+--------------+------+------+ ----------+
| aa|    여보세요, 여보세요 |    8 |    0 |    0 |
| test1 | t1                |10 |1 |0 |
+-------+--------------+------+------+ ----------+
```

4. `set` 명령을 사용하여 키를 `new` 값으로 업데이트합니다.

```
텔넷 127.0.0.1 11211
127.0.0.1 시도 중...
```

```
127.0.0.1에 연결되었습니다. 이
스케이프 문자는 '^]'입니다. set
test1 10 0 2
new
저장됨
```

업데이트가 복제본 서버에 복제됩니다(`cas` 값도 업데이트됩니다).

```
mysql> SELECT * FROM test.demo_test;
+-------+--------------+------+------+ ----------+
| C1| C2| C3| C4| C5     |
+-------+--------------+------+------+ ----------+
| aa|    여보세요, 여보세요 |    8 |    0 |    0 |
| test1 | new              |10 |2 |0 |
+-------+--------------+------+------+ ----------+
```

5. `삭제` 명령을 사용하여 `test1` 레코드를 삭제합니다.

```
텔넷 127.0.0.1 11211
127.0.0.1 시도 중...
127.0.0.1에 연결되었습니다. 이
스케이프 문자는 '^]'입니다.
test1 삭제
삭제됨
```

`삭제` 작업이 복제본에 복제되면 복제본의 `test1` 레코드도 삭제됩니다.

```
mysql> SELECT * FROM test.demo_test;
+----+--------------+------+------+ ----------+
| C1 | C2| C3| C4| C5       |
+----+--------------+------+------+ ----------+
| AA | 헬로, 헬로 |8 |0 |0 |0 |
+----+--------------+------+------+ ----------+
```

6. `flush_all` 명령을 사용하여 테이블에서 모든 행을 제거합니다.

```
텔넷 127.0.0.1 11211
127.0.0.1 시도 중...
127.0.0.1에 연결되었습니다. 이
스케이프 문자는 '^]'입니다.
mysql> SELECT * FROM test.demo_test;
빈 세트 (0.00초)
```

7. 소스 서버에 텔넷으로 접속하여 두 개의 새 레코드를 입력합니다.

```
텔넷 127.0.0.1 11211
127.0.0.1 시도 중...
127.0.0.1에 연결되었습니다.
이스케이프 문자는 '^]'입니다.
test2 10 0 4 설정
다시
저장됨
SET TEST3 10 0 5
AGAIN1
저장됨
```

8. 두 레코드가 복제본 서버에 복제되었는지 확인합니다.

```
mysql> SELECT * FROM test.demo_test;
+-------+--------------+------+------+ ----------+
| C1| C2| C3| C4| C5    |
+-------+--------------+------+------+ ----------+
| test2 | 다시          |   10 |    4 |    0 |
| test3 | 다시1             |10 |5 |0 |
+-------+--------------+------+------+ ----------+
```

9. `flush_all` 명령을 사용하여 테이블에서 모든 행을 제거합니다.

```
telnet 127.0.0.1 11211
127.0.0.1 시도 중...
127.0.0.1에 연결되었습니다. 이
스케이프 문자는 '^]'입니다.
flush_all
확인
```

10. flush_all 작업이 복제본 서버에 복제되었는지 확인합니다.

```
mysql> SELECT * FROM test.demo_test;
빈 세트 (0.00초)
```

InnoDB 멤캐시드 바이너리 로그 노트

바이너리 로그 형식:

- 대부분의 멤캐시드 작업은 DML 문(삽입, 삭제, 업데이트와 유사)에 매핑됩니다. MySQL 서버에서 처리되는 실제 SQL 문이 없으므로 모든 memcached 명령(flush_all 제외)은 서버 binlog_format 설정과 무관한 행 기반 복제(RBR) 로깅을 사용합니다.
- DDL 명령은 문 기반 로깅만 사용할 수 있으므로 flush_all 명령은 TRUNCATE TABLE 문을 전송하여 복제됩니다. MySQL 8.2에서 flush_all은 DELETE에 매핑되지만 여전히 TRUNCATE TABLE 문을 전송하여 복제됩니다.

거래:

- 트랜잭션의 개념은 일반적으로 멤캐시드 애플리케이션에 포함되지 않았습니다. 성능 고려를 위해 daemon_memcached_r_batch_size 및 daemon_memcached_w_batch_size는 읽기 및 쓰기 트랜잭션의 배치 크기를 제어하는 데 사용됩니다. 이러한 설정은 복제에 영향을 미치지 않습니다. 기본 InnoDB 테이블에 대한 각 SQL 작업은 성공적으로 완료된 후 복제됩니다.
- daemon_memcached_w_batch_size의 기본값은 1이며, 이는 각 멤캐시드 쓰기 작업이 즉시 커밋됨을 의미합니다. 이 기본 설정은 원본 서버와 복제본 서버에 표시되는 데이터의 불일치를 방지하기 위해 일정량의 성능 오버헤드를 발생시킵니다. 복제된 레코드는 복제본 서버에서 항상 즉시 사용할 수 있습니다. daemon_memcached_w_batch_size를 1보다 큰 값으로 설정하면 멤캐시를 통해 삽입되거나 업데이트된 레코드는 원본 서버에 즉시 표시되지 않으며, 커밋되기 전에 원본 서버에서 레코드를 보려면 트랜잭션 격리 수준 읽기 미커밋 설정을 실행합니다.

15.20.8 InnoDB 멤캐시드 플러그인 내부

InnoDB 멤캐시드 플러그인을 위한 InnoDB API

InnoDB 멤캐시드 엔진은 대부분 임베디드 InnoDB에서 직접 채택한 InnoDB API를 통해 InnoDB에 액세스합니다. InnoDB API 함수는 콜백 함수로 InnoDB 멤캐시드 엔진에 전달됩니다. InnoDB API 함수는 InnoDB 테이블에 직접 액세스하며, TRUNCATE TABLE을 제외한 대부분의 함수는 DML 연산입니다.

`memcached` 명령은 `InnoDB memcached` API를 통해 구현됩니다. 다음 표는 `memcached` 명령이 DML 또는 DDL 작업에 매핑되는 방법을 간략하게 설명합니다.

표 15.27 memcached 명령 및 관련 DML 또는 DDL 작업

멤캐시드 명령	DML 또는 DDL 작업
`get`	읽기/가져오기 명령
`set`	검색 후 삽입 또는 업데이트 (키의 존재 여부에 따라 다름)
추가	검색 후 삽입 또는 업데이트

멤캐시드 명령	DML 또는 DDL 작업
교체	검색 후 업데이트
추가	검색 후 업데이트(업데이트 전에 결과에 데이터 추가)
prepend	검색 후 업데이트(업데이트 전에 결과에 데이터를 추가)
incr	검색 후 업데이트
decr	검색 후 업데이트
삭제	검색 후 삭제
flush_all	테이블 잘라내기(DDL)

InnoDB 멤캐시드 플러그인 구성 테이블

이 섹션에서는 daemon_memcached 플러그인에서 사용하는 구성 테이블에 대해 설명합니다. 캐시 정책 테이블, config_옵션 테이블, 컨테이너 테이블은 innodb_memcache 데이터베이스의 innodb_memcached_config.sql 구성 스크립트에 의해 생성됩니다.

```
mysql> USE innodb_memcache;
데이터베이스가 변경되었습
니다. mysql> 테이블 표
시;
+----------------------------+
| Tables_in_innodb_memcache |
+----------------------------+
| 캐시 정책                    |
| config_options             |
| 컨테이너                     |
+----------------------------+
```

캐시 정책 테이블

cache_policies 테이블은 InnoDB 멤캐시드 설치에 대한 캐시 정책을 정의합니다. 단일 캐시 정책 내에서 가져오기, 설정, 삭제, 플러시 작업에 대한 개별 정책을 지정할 수 있습니다. 모든 작업에 대한 기본 설정은 innodb_only입니다.

- innodb_only: 데이터 저장소로 InnoDB를 사용합니다.
- 캐시 전용: 멤캐시드 엔진을 데이터 저장소로 사용합니다.
- 캐싱: InnoDB와 멤캐시드 엔진을 모두 데이터 저장소로 사용합니다. 이 경우, 멤캐시된 는 메모리에서 키를 찾을 수 없으면 InnoDB 테이블에서 값을 검색합니다.
- 비활성화합니다: 캐싱을 비활성화합니다.

표 15.28 cache_정책 열

열	설명

정책_이름	캐시 정책의 이름입니다. 기본 캐시 정책 이름은 cache_policy입니다.
get_policy	get 작업에 대한 캐시 정책입니다. 유효한 값은 innodb_only, cache_only, caching 또는 disabled입니다. 기본 설정은 innodb_only입니다.
set_policy	집합 연산에 대한 캐시 정책입니다. 유효한 값은 innodb_only, cache_only, caching 또는 disabled입니다. 기본 설정은 innodb_only입니다.
삭제_정책	삭제 작업에 대한 캐시 정책입니다. 유효한 값은 innodb_only, cache_only, caching 또는 disabled입니다. 기본 설정은 innodb_only입니다.

열	설명
플러시_정책	플러시 작업에 대한 캐시 정책입니다. 유효한 값은 innodb_only, cache_only, caching 또는 disabled입니다. 기본 설정은 innodb_only입니다.

config_옵션 테이블

config_options 테이블에는 SQL을 사용하여 런타임에 변경할 수 있는 멤캐시드 관련 설정이 저장됩니다. 지원되는 구성 옵션은 구분 기호 및 table_map_delimiter입니다.

표 15.29 config_options 열

열	설명

이름	멤캐시드 관련 구성 옵션의 이름입니다. config_options 테이블에서 지원하는 구성 옵션은 다음과 같습니다:
	• 구분 기호: 여러 개의 value_column이 정의된 경우 긴 문자열의 값을 별도의 값으로 구분하는 데 사용됩니다. 기본적으로 사용됩니다, 구분 기호는 | 문자입니다. 예를 들어, col1, col2를 값 열로 정의하고 |를 구분 기호로 정의한 경우 다음 memcached 명령을 실행하여 각각 col1과 col2에 값을 삽입할 수 있습니다:
	`SET KEYX 10 0 19`
	`VALUECOLX|VALUECOLY`
	VALUECOL1X는 COL1에 저장되고 VALUE_COLY는 COL2에 저장됩니다.
	• 테이블_맵_구분기호: 키 이름에 @@ 표기법을 사용하여 특정 테이블의 키에 액세스할 때 스키마 이름과 테이블 이름을 구분하는 문자입니다. 예를 들어, @@t1.some_key 및 @@t2.일부_키의 키 값은 동일하지만 다른 테이블에 저장됩니다.

가치	멤캐시드 관련 구성 옵션에 할당된 값입니다.

컨테이너 테이블

컨테이너 테이블은 세 가지 구성 테이블 중 가장 중요한 테이블입니다. 멤캐시된 값을 저장하는 데 사용되는 각 InnoDB 테이블에는 containers 테이블에 항목이 있어야 합니다. 항목

는 InnoDB 테이블 열과 컨테이너 테이블 열 간의 매핑을 제공하며, 이는 다음과 같은 경우에 필요합니다. 멤캐시를 사용하여 InnoDB 테이블과 함께 작동합니다.

컨테이너 테이블에는 innodb_memcached_config.sql 구성 스크립트에 의해 생성되는 test.demo_test 테이블에 대한 기본 항목이 포함되어 있습니다. 자체 InnoDB 테이블과 함께 daemon_memcached 플러그인을 사용하려면 컨테이너 테이블에 항목을 만들어야 합니다.

표 15.30 컨테이너 열

열	설명
이름	컨테이너에 지정된 이름입니다. InnoDB 테이블이 @@ 표기법을 사용하여 이름으로 요청되지 않은 경우, daemon_memcached 플러그인은 containers.name 값이

열	설명
	기본값입니다. 이러한 항목이 없는 경우 컨테이너 테이블의 첫 번째 항목이 이름별로 알파벳순(오름차순)으로 정렬되어 기본 InnoDB 테이블이 결정됩니다.
db_schema	InnoDB가 저장된 데이터베이스의 이름입니다. 테이블이 있는 위치입니다. 필수 값입니다.
db_table	저장하는 InnoDB 테이블의 이름입니다. 멤캐시된 값입니다. 필수 값입니다.
키_열	멤캐시된 작업에 대한 조회 키 값이 포함된 InnoDB 테이블의 열입니다. 필수 값입니다.
value_columns	멤캐시된 데이터를 저장하는 InnoDB 테이블 열(하나 이상)입니다. innodb_memcached.config_options 테이블에 지정된 구분 문자를 사용하여 여러 열을 지정할 수 있습니다. 기본적으로 구분 문자는 파이프 문자("|")입니다. 여러 열을 지정하려면 정의된 구분 문자로 열을 구분합니다. 예: col1|col2|col3. 이 값은 필수입니다.
플래그	멤캐시에 대한 플래그(기본 값과 함께 저장 및 검색되는 사용자 정의 숫자 값)로 사용되는 InnoDB 테이블 열입니다. 플래그 값은 다음과 같이 사용할 수 있습니다. 일부 연산에 대한 열 지정자(예로 추가)를 사용하여 멤캐시된 값이 여러 열에 매핑되어 있는 경우 지정된 열에서 작업이 수행되도록 합니다. 예를 들어, value_columns를 세 개의 InnoDB 테이블 열이 있고 한 열에서만 증분 연산을 수행하려고 합니다, 플래그 열을 사용하여 열을 지정합니다. 플래그 열을 사용하지 않는 경우 사용하지 않음을 나타내는 값을 0으로 설정합니다.
cas_column	비교 및 스왑(cas) 값을 저장하는 InnoDB 테이블 열입니다. cas_column 값은 멤캐시가 여러 서버에 요청을 해시하고 메모리에 데이터를 캐시하는 방식과 관련이 있습니다. InnoDB 멤캐시드 플러그인은 단일 멤캐시드 데몬과 긴밀하게 통합되어 있고 인메모리 캐싱 메커니즘은 MySQL과 InnoDB 버퍼 풀에서 처리하므로 이 열은 거

		의 필요하지 않습니다. 이 열을 사용하지 않는 경우 값을 0으로 설정하여 사용하지 않음을 표시하세요.
	만료_시간_열	만료 값을 저장하는 InnoDB 테이블 열입니다. 만료_시간_열 값은 멤캐시가 여러 서버에 요청을 해시하고 데이터를 메모리에 캐싱하는 방식과 관련이 있습니다. InnoDB 멤캐시드 플러그인은 단일 멤캐시드 데몬과 긴밀하게 통합되어 있고 인메모리 캐싱 메커니즘이 처리되기 때문입니다. 을 사용하는 경우 이 열은 거의 필요하지 않습니다. 이 열을 사용하지 않는 경우 값을 0으로 설정하여 열이 사용되지 않음을 나타냅니다. 최대 만료 시간은 다음과 같습니다.

열	설명
	INT_MAX32 또는 2147483647 초(약 68년)로 정의됩니다.
고유_IDX_이름_온_키	키 열에 있는 인덱스의 이름입니다. 고유 인덱스여야 합니다. 기본 키이거나 보조 인덱스입니다. 가급적이면 InnoDB 테이블의 기본 키를 사용하는 것이 좋습니다. 기본 키를 사용하면 보조 인덱스를 사용할 때 수행되는 조회를 피할 수 있습니다. 멤캐시 조회를 위한 커버링 인덱스를 만들 수 없으며, 키 및 값 열 모두에 대해 복합 보조 인덱스를 정의하려고 하면 InnoDB에서 오류를 반환합니다.

컨테이너 테이블 열 제약 조건

- db_schema, db_name, key_columns, value_columns 및 unique_idx_name_on_key에 대한 값을 제공해야 합니다. 플래그, cas_column, expire_time_column은 사용하지 않는 경우 0을 지정합니다. 그렇지 않으면 설정이 실패할 수 있습니다.
- 키_열: 멤캐시된 키의 최대 제한은 250자로, 다음과 같이 강제됩니다.
  멤캐시드. 매핑된 키는 Null이 아닌 CHAR 또는 VARCHAR 유형이어야 합니다.
- value_columns: CHAR, VARCHAR 또는 BLOB 열에 매핑되어야 합니다. 길이 제한은 없으며 값은 NULL일 수 있습니다.
- cas_column: cas 값은 64비트 정수입니다. 최소 8바이트의 BIGINT에 매핑되어야 합니다. 이 열을 사용하지 않는 경우 사용하지 않음을 나타내는 값을 0으로 설정합니다.
- 만료_시간_열: 4바이트 이상의 INTEGER에 매핑되어야 합니다. 만료 시간은 유닉스 시간의 경우 32비트 정수(1970년 1월 1일 이후 32비트 값으로 초 수) 또는 현재 시간부터 시작되는 초 수로 정의됩니다. 후자의 경우 초 수는 60*60*24*30(30일의 초 수)을 초과할 수 없습니다. 클라이언트가 보낸 숫자가 더 큰 경우 서버는 이를 현재 시간에서 오프셋된 값이 아닌 실제 유닉스 시간 값으로 간주합니다. 이 열을 사용하지 않는 경우 0으로 설정하여 사용하지 않음을 나타냅니다.
- 플래그: 최소 32비트 이상의 INTEGER에 매핑되어야 하며 NULL일 수 있습니다. 이 열을 사용하지 않는 경우 0으로 설정하여 사용하지 않음을 나타냅니다.

플러그인 로드 시 사전 검사를 수행하여 열 제약 조건을 적용합니다. 불일치가 발견되면 플러그인이 로드되지 않습니다.

다중 값 열 매핑

- 플러그인 초기화 중에 컨테이너 테이블에 정의된 정보로 InnoDB memcached가 구성되면,

containers.value_columns에 정의된 각 매핑된 열이 매핑된 InnoDB 테이블과 비교하여 확인됩니다. 여러 InnoDB 테이블 열이 매핑된 경우, 각 열이 존재하고 올바른 유형인지 확인합니다.

- 런타임에 멤캐시된 삽입 작업의 경우 매핑된 열 수보다 구분된 값이 더 많은 경우 매핑된 값의 수만 사용됩니다. 예를 들어 매핑된 열이 6개이고 구분된 값이 7개 제공된 경우 처음 6개 구분된 값만 가져옵니다. 일곱 번째 구분 값은 무시됩니다.
- 매핑된 열보다 구분된 값의 수가 적은 경우 채워지지 않은 열은 NULL로 설정됩니다. 채워지지 않은 열을 NULL로 설정할 수 없는 경우 삽입 작업이 실패합니다.
- 테이블에 매핑된 값보다 많은 열이 있는 경우 여분의 열은 결과에 영향을 주지 않습니다.

데모 테스트 예제 표

innodb_memcached_config.sql 구성 스크립트는 테스트에서 demo_test 테이블을 생성합니다. 데이터베이스에 액세스하여 설정 직후 InnoDB 멤캐시드 플러그인 설치를 확인하는 데 사용할 수 있습니다.

innodb_memcached_config.sql 구성 스크립트는 demo_test에 대한 항목도 생성합니다. 테이블을추가합니다.

```
mysql> SELECT * FROM innodb_memcache.containers\G
*************************** 1. 행 *************************** 이
                  름: AAA
            DB_SCHEMA: 테스트
             DB_TABLE: 데모 테스트
          KEY_COLUMNS: C1
        VALUE_COLUMNS: C2
                  플래그: C3
           CAS_COLUMN: C4
   EXPIRE_TIME_COLUMN: C5
고유_IDX_NAME_ON_KEY: PRIMARY

mysql> SELECT * FROM test.demo_test;
+----+-----------------+------+------+----------+
| C1 |                 C2| C3| C4| C5 |
+----+-----------------+------+------+----------+
| aa | 안녕하세요, 안녕하세요     |8 |0 |0 |
+----+-----------------+------+------+----------+
```

15.20.9 InnoDB 멤캐시드 플러그인 문제 해결

이 섹션에서는 InnoDB 멤캐시드 플러그인을 사용할 때 발생할 수 있는 문제에 대해 설명합니다.

- MySQL 오류 로그에서 다음과 같은 오류가 발생하면 서버가 시작되지 않을 수 있습니다:

  열린 파일에 대한 제한을 설정하지 못했습니다. 루트로 실행하거나 더 작은 maxconns 값을 요청해 보세요.

  이 오류 메시지는 멤캐시드 데몬에서 발생한 것입니다. 한 가지 해결책은 열려 있는 파일 수에 대한 OS 제한을 높이는 것입니다. 열려 있는 파일 제한을 확인하고 늘리는 명령은 운영 체제에 따라 다릅니다. 이 예는 Linux 및 macOS용 명령을 보여줍니다:

  ```
  # Linux
  $> ulimit -n
  1024
  $> ulimit -n 4096
  $> ulimit -n
  4096

  # macOS
  $> ulimit -n
  256
  $> ulimit -n 4096
  $> ulimit -n
  4096
  ```

  다른 해결책은 멤캐시드 데몬에 허용되는 동시 연결 수를 줄이는 것입니다. 이렇게 하려면 MySQL 구성 파일의 daemon_memcached_option 구성 매개변수에서 -c memcached 옵션을 인코딩하면 됩니다.

`c` 옵션의 기본값은 1024입니다.

```
[mysqld]
...
loose-daemon_memcached_option='-c 64'
```

- 멤캐시드 데몬이 `InnoDB` 테이블 데이터를 저장하거나 검색할 수 없는 문제를 해결하려면 MySQL 구성 파일의 `daemon_memcached_option` 구성 매개변수에서 `-vvv memcached` 옵션을 인코딩하세요. 멤캐시드 작업과 관련된 디버그 출력은 MySQL 오류 로그에서 확인합니다.

```
[mysqld]
...
loose-daemon_memcached_option='-vvv'
```

- 멤캐시된 값을 저장하도록 지정된 열이 문자열 유형이 아닌 숫자 유형과 같이 잘못된 데이터 유형인 경우 키-값 쌍을 저장하려는 시도가 특정 오류 코드나 메시지 없이 실패합니다.

- daemon_memcached 플러그인으로 인해 MySQL 서버 시작 문제가 발생하는 경우, 문제를 해결하는 동안 MySQL 구성 파일의 [mysqld] 그룹 아래에 이 줄을 추가하여 daemon_memcached 플러그인을 일시적으로 비활성화할 수 있습니다:

  ```
  데몬_메모리캐시드=OFF
  ```

  예를 들어, 필요한 데이터베이스 및 테이블을 설정하기 위해 innodb_memcached_config.sql 구성 스크립트를 실행하기 전에 INSTALL PLUGIN 문을 실행하면 서버가 예기치 않게 종료되고 시작에 실패할 수 있습니다. 또한 innodb_memcache.containers 테이블의 항목을 잘못 구성하면 서버가 시작되지 않을 수 있습니다.

  MySQL 인스턴스에 대한 멤캐시드 플러그인을 제거하려면 다음 문을 실행합니다:

  ```
  mysql> UNINSTALL PLUGIN daemon_memcached;
  ```

- 동일한 시스템에서 두 개 이상의 MySQL 인스턴스를 실행하고 각 인스턴스에서 daemon_memcached 플러그인을 활성화한 경우, daemon_memcached_option 구성 매개변수를 사용하여 각 daemon_memcached 플러그인에 대해 고유한 memcached 포트를 지정합니다.

- SQL 문이 InnoDB 테이블을 찾을 수 없거나 테이블에서 데이터를 찾지 못하지만 memcached API 호출이 예상 데이터를 검색하는 경우, innodb_memcache.containers 테이블에서 InnoDB 테이블에대한 항목이 누락되었거나 *@@table_id* 표기법을 사용하여 get 또는 set 요청을 실행하여 올바른 InnoDB 테이블로 전환하지 않았을 수 있습니다. 이 문제는 MySQL 서버를 다시 시작하지 않고 innodb_memcache.containers 테이블의 기존 항목을 변경하는 경우에도 발생할 수 있습니다. 자유 형식 저장 메커니즘은 충분히 유연하여, 데몬이 단일 열에 값을 저장하는 test.demo_test 테이블을 사용하는 경우에도 col1|col2|col3과 같은 다중 열 값을 저장하거나 검색하는 요청이 계속 작동할 수 있습니다.

- daemon_memcached 플러그인과 함께 사용할 자체 InnoDB 테이블을 정의하고 테이블의 열이 NOT NULL로 정의된 경우 NOT NULL 열에 대한 값을 제공해야 합니다.
  테이블에 대한 레코드를 innodb_memcache.containers 테이블에 삽입할 때. innodb_memcache.containers 레코드에 대한 INSERT 문에 매핑된 열의 수보다 구분된 값이 적게 포함된 경우, 채워지지 않은 열은 NULL로 설정됩니다. NOT NULL 열에 NULL 값을 삽입하려고 하면 INSERT가 실패하게 되며, 이는 daemon_memcached 플러그인을 다시 초기화하여 innodb_memcache.containers 테이블에 변경 사항을 적용한 후에야 분명해질 수 있습니다.

- innodb_memcached.containers 테이블의 cas_column 및 expire_time_column 필드가 NULL로 설정되어 있으면 memcached 플러그인 로드를 시도할 때 다음과 같은 오류가 반환됩니다:

  ```
  InnoDB_Memcached: 데이터베이스 'innodb_memcache'의 구성 테이블 '컨테이너' 항목에 있는 열
  6에 잘못된 NULL 값이 있습니다.
  ```

`memcached` 플러그인은 `cas_column` 및 `expire_time_column`에서 `NULL`의 사용을 거부합니다. 열을 설정합니다. 열을 사용하지 않을 경우 이 열의 값을 `0`으로 설정합니다.

- 멤캐시된 키와 값의 길이가 길어지면 크기와 길이 제한이 발생할 수 있습니다.
  - 키가 250바이트를 초과하면 멤캐시드 연산은 오류를 반환합니다. 이는 현재 `memcached` 내에서 고정된 제한입니다.
  - 값이 크기가 768바이트, 3072바이트 또는 `innodb_page_size` 값의 절반을 초과하는 경우 `InnoDB` 테이블 제한이 발생할 수 있습니다. 이러한 제한은 주로 SQL을 사용하여 해당 열에 대한 보고서 생성 쿼리를 실행하기 위해 값 열에 인덱스를 만들려는 경우에 적용됩니다. 자세한 내용은 섹션 15.22, "InnoDB 제한"을 참조하세요.

  - 키-값 조합의 최대 크기는 1MB입니다.

- 서로 다른 버전의 MySQL 서버에서 구성 파일을 공유하는 경우, `daemon_memcached` 플러그인에 최신 구성 옵션을 사용하면 이전 MySQL 버전에서 시작 오류가 발생할 수 있습니다. 호환성 문제를 방지하려면 옵션 이름에 느슨한 접두사를 사용하세요. 예를 들어, `daemon_memcached_option='-c 64'` 대신 `loose-daemon_memcached_option='-c 64'`를 사용하세요.

- 멤캐시는 키와 값을 바이트 단위로 저장하고 검색하므로 문자 집합에 민감하지 않습니다. 하지만 멤캐시드 클라이언트와 MySQL 테이블이 동일한 문자 집합을 사용하는지 확인해야 합니다.

- 멤캐시드 연결은 인덱싱된 가상 열이 포함된 테이블에 액세스하지 못하도록 차단됩니다. 인덱싱된 가상 열에 액세스하려면 서버에 대한 콜백이 필요하지만 멤캐시드 연결에는 서버 코드에 대한 액세스 권한이 없습니다.

15.21 InnoDB 문제 해결

다음 일반 지침은 `InnoDB` 문제 해결에 적용됩니다:

- 작업이 실패하거나 버그가 의심되는 경우 MySQL 서버 오류 로그를 확인하세요(5.4.2절. '오류 로그' 참조). 서버 오류 메시지 참조는 발생할 수 있는 몇 가지 일반적인 `InnoDB` 관련 오류에 대한 문제 해결 정보를 제공합니다.

- 장애가 교착 상태와 관련된 경우, 각 교착 상태에 대한 세부 정보가 MySQL 서버 오류 로그에 인쇄되도록 `innodb_print_all_deadlocks` 옵션을 활성화한 상태로 실행하세요. 교착 상태에 대한 자세한 내용은 섹션 15.7.5, "InnoDB의 교착 상태"를 참조하세요.

- `InnoDB` 데이터 사전과 관련된 문제인 경우 15.21.4절. 'InnoDB 데이터 사전 작업 문제 해결'을 참조하세요.

- 문제를 해결할 때는 일반적으로 `mysqld_safe` 또는 Windows 서비스가 아닌 명령 프롬프트에서 MySQL 서버를 실행하는 것이 가장 좋습니다. 그러면 `mysqld`가 콘솔에 출력하는 내용을 볼 수 있으므로 무슨 일이 일어나고 있는지 더 잘 파악할 수 있습니다. Windows에서는 `--` 콘솔 옵션을 사용하여 `mysqld`를 시작하면 출력이 콘솔 창으로 출력됩니다.

- `InnoDB` 모니터를 활성화하여 문제에 대한 정보를 얻습니다(섹션 15.17, "InnoDB 모니터" 참조). 문제가 성능과 관련이 있거나 서버가 중단된 것처럼 보이는 경우, 표준 모니터를 활성화하여 `InnoDB`의 내부 상태에 대한 정보를 인쇄해야 합니다. 잠금과 관련된 문제인 경우 잠금 모니터를 활성화합니다. 테이블 생성, 테이블 공간 또는 데이터에 문제가 있는 경우
  사전 작업을 수행하려면 InnoDB 정보 스키마 시스템 테이블을 참조하여 `InnoDB` 내부 데이터 사전의 내용을 검토하세요.

  InnoDB는 다음 조건에서 일시적으로 표준 `InnoDB` 모니터 출력을 활성화합니다:

  - 긴 세마포어 대기
  - InnoDB가 버퍼 풀에서 여유 블록을 찾을 수 없습니다.
  - 버퍼 풀의 67% 이상이 잠금 힙 또는 적응형 해시 인덱스에 의해 점유됩니다.
- 테이블이 손상된 것으로 의심되는 경우 해당 테이블에서 테이블 확인을 실행합니다.

15.21.1 InnoDB I/O 문제 해결

InnoDB I/O 문제에 대한 문제 해결 단계는 문제가 발생하는 시점에 따라 달라집니다. MySQL 서버를 시작하는 동안 또는 파일 시스템 수준의 문제로 인해 DML 또는 DDL 문이 실패하는 정상 작동 중에 발생합니다.

초기화 문제

InnoDB가 테이블 공간 또는 로그 파일을 초기화하려고 시도할 때 문제가 발생하면 InnoDB에서 생성한 모든 파일(모든 ibdata 파일 및 모든 재실행 로그 파일(*#ib_redoN* 파일)을 삭제하세요. InnoDB 테이블을 생성한 경우 MySQL 데이터베이스 디렉터리에서 .ibd 파일도 삭제하세요. 그런 다음 InnoDB를 다시 초기화해 보세요. 가장 쉽게 문제를 해결하려면 명령 프롬프트에서 MySQL 서버를 시작하여 어떤 일이 발생하는지 확인하세요.

런타임 문제

파일 작업 중에 InnoDB에서 운영 체제 오류를 출력하는 경우 일반적으로 다음 해결 방법 중 하나가 있습니다:

- InnoDB 데이터 파일 디렉터리와 InnoDB 로그 디렉터리가 존재하는지 확인합니다.
- mysqld에 해당 디렉터리에 파일을 만들 수 있는 액세스 권한이 있는지 확인합니다.
- mysqld가 지정한 옵션으로 시작할 수 있도록 적절한 my.cnf 또는 my.ini 옵션 파일을 읽을 수 있는지 확인합니다.
- 디스크가 꽉 차 있지 않고 디스크 할당량을 초과하지 않았는지 확인합니다.
- 하위 디렉터리와 데이터 파일에 지정한 이름이 충돌하지 않는지 확인하세요.
- innodb_data_home_dir 및 innodb_data_file_path 값의 구문을 다시 확인합니다. 특히 innodb_data_file_path 옵션의 MAX 값은 하드 리밋이며, 이 값을 초과하면 치명적인 오류가 발생합니다.

15.21.2 복구 실패 문제 해결

재실행 로그 복구가 완료되고 데이터 사전 동적 메타데이터(srv_dict_metadata)가 데이터 사전 테이블(dict_table_t) 개체로 전송될 때까지 체크포인트 및 체크포인트 LSN 전진은 허용되지 않습니다. 복구 중 또는 복구 후(데이터 사전 동적 메타데이터가 데이터 사전 테이블 개체로 전송되기 전) 재실행 로그의 공간이 부족해지면 다음과 같은 결과가 발생합니다.
이 변경 사항을 적용하려면 최소한 SRV_FORCE_NO_IBUF_MERGE 설정으로 시작하거나, 실패할 경우 SRV_FORCE_NO_LOG_REDO 설정으로 시작하여 innodb_force_recovery 재시작이 필요할 수 있습니다. 이 시나리오에서 innodb_force_recovery 재시작이 실패하면 백업에서 복구해야 할 수 있습니다.

15.21.3 InnoDB 복구 강제 적용

데이터베이스 페이지 손상을 조사하려면 다음을 사용하여 데이터베이스에서 테이블을 덤프할 수 있습니다. 선택 ...을 아웃파일에 넣습니다. 일반적으로 이러한 방식으로 얻은 대부분의 데이터는 손상되지 않습니다. 심각한 손상으로 인해 SELECT * FROM *tbl_name* 문 또는 InnoDB 백그라운드 작업이 예기치 않게 종료되거나 어설션이 발생하거나 심지어 InnoDB 롤포워드 복구가 중단될 수도 있습니다. 이러한 경우

innodb_force_recovery 옵션을 사용하여 백그라운드 작업이 실행되지 않도록 하면서 InnoDB 스토리지 엔진을 강제로 시작하여 테이블을 덤프할 수 있습니다. 예를 들어, 서버를 다시 시작하기 전에 옵션 파일의 [mysqld] 섹션에 다음 줄을 추가할 수 있습니다:

```
[mysqld]
innodb_force_recovery = 1
```

옵션 파일 사용에 대한 자세한 내용은 4.2.2.2절. '옵션 파일 사용하기'를 참조하세요.

경고

긴급 상황에서만 innodb_force_recovery를 0보다 큰 값으로 설정하여 InnoDB를 시작하고 테이블을 덤프할 수 있도록 하세요. 수행하기 전에
따라서 데이터베이스를 다시 만들어야 할 경우를 대비하여 데이터베이스의 백업 복사본이 있는지 확인하세요. 4보다 큰 값은 데이터 파일을 영구적으로 손상시킬 수 있습니다. 별도의 물리적 서버 인스턴스에서 설정을 성공적으로 테스트한 후에만 프로덕션 서버 인스턴스에서 4 이상의 innodb_force_recovery 설정을 사용하세요.

> 데이터베이스의 복사본을 만듭니다. InnoDB 복구를 강제로 수행할 때는 항상 innodb_force_recovery=1로 시작하고 필요에 따라 값을 점진적으로 늘려야 합니다.

기본값은 0입니다(강제 복구 없이 정상 시작). innodb_force_recovery에 허용되는 0이 아닌 값은 1~6입니다. 값이 클수록 더 작은 값의 기능도 포함됩니다. 예를 들어, 값 3은 값 1과 2의 모든 기능을 포함합니다.

innodb_force_recovery 값이 3 이하로 테이블을 덤프할 수 있는 경우 손상된 개별 페이지의 일부 데이터만 손실되므로 비교적 안전합니다. 4 이상의 값은 데이터 파일이 영구적으로 손상될 수 있으므로 위험한 것으로 간주됩니다. 값이 6이면 데이터베이스 페이지가 더 이상 사용되지 않는 상태로 남게 되어 B-트리 및 기타 데이터베이스 구조에 더 많은 손상이 발생할 수 있으므로 심각한 수준으로 간주됩니다.

안전 조치로 InnoDB는 innodb_force_recovery가 0보다 큰 경우 INSERT, UPDATE 또는 DELETE 작업을 수행하지 못하도록 합니다. innodb_force_recovery를 4 이상으로 설정하면 InnoDB가 읽기 전용 모드로 전환됩니다.

- 1 (SRV_FORCE_IGNORE_CORRPT)

  손상된 페이지가 감지되더라도 서버를 실행합니다. SELECT * FROM *tbl_name*을 시도합니다. 손상된 인덱스 레코드와 페이지를 건너뛰면 테이블 덤핑에 도움이 됩니다.

- 2 (SRV_FORCE_NO_BACKGROUND)

  마스터 스레드 및 모든 퍼지 스레드가 실행되지 않도록 합니다. 퍼지 작업 중에 예기치 않은 종료가 발생하는 경우 이 복구 값으로 이를 방지합니다.

- 3 (SRV_FORCE_NO_TRX_UNDO)

  충돌 복구 후 트랜잭션 롤백을 실행하지 않습니다.

- 4 (SRV_FORCE_NO_IBUF_MERGE)

  삽입 버퍼 병합 작업을 방지합니다. 충돌을 일으킬 수 있는 경우 수행하지 않습니다. 테이블 통계를 계산하지 않습니다. 이 값은 데이터 파일을 영구적으로 손상시킬 수 있습니다. 이 값을 사용한 후에는 모든 보조 인덱스를 삭제하고 다시 생성할 준비를 하세요. InnoDB를 읽기 전용으로 설정합니다.

- 5 (SRV_FORCE_NO_UN_LOG_SCAN)

  데이터베이스를 시작할 때 실행 취소 로그를 확인하지 않습니다: InnoDB는 불완전한 트랜잭션도 커밋된 것으로 처리합니다. 이 값은 데이터 파일을 영구적으로 손상시킬 수 있습니다. InnoDB를 읽기 전용으로 설정합니다.

- 6 (SRV_FORCE_NO_LOG_REDO)

  복구와 관련하여 재실행 로그 롤포워드를 수행하지 않습니다. 이 값은 데이터 파일을 영구적으로 손상시킬 수 있습니다. 데이터베이스 페이지를 더 이상 사용되지 않는 상태로 남겨두어 B-트리 및 기타 데이터베이스 구조에 더 많은 손상을 일으킬 수 있습니다. InnoDB를 읽기 전용으로 설정합니다.

테이블에서 SELECT를 수행하여 테이블을 덤프할 수 있습니다. innodb_force_recovery 값이 3 이하인 경우 테이블을 DROP 또는 CREATE 할 수 있습니다. innodb_force_recovery 값이 3보다 큰 경우에도 DROP TABLE이 지원됩니다. innodb_force_recovery 값이 4보다 큰 경우 DROP TABLE은 허용되지 않습니다.

특정 테이블이 롤백 시 예기치 않은 종료를 유발하는 것을 알고 있다면 해당 테이블을 삭제할 수 있습니다. 대량 가져오기 또는 테이블 변경 실패로 인해 롤백이 발생하는 경우, mysqld 프로세스를 종료하고 innodb_force_recovery를 3으로 설정하여 롤백 없이 데이터베이스를 복구한 다음 롤백의 원인이 되는 테이블을 삭제할 수 있습니다.

테이블 데이터 내의 손상으로 인해 전체 테이블 내용을 덤프할 수 없는 경우, ORDER BY *primary_key* DESC 절이 포함된 쿼리를 사용하면 손상된 부분 이후의 테이블 부분을 덤프할 수 있습니다.

InnoDB를 시작하는 데 높은 innodb_force_recovery 값이 필요한 경우, 데이터 구조가 손상되어 복잡한 쿼리(WHERE, ORDER BY 또는 기타 절을 포함하는 쿼리)가 실패할 수 있습니다. 이 경우 기본적인 SELECT * FROM t 쿼리만 실행할 수 있습니다.

15.21.4 InnoDB 데이터 사전 작업 문제 해결

테이블 정의에 대한 정보는 InnoDB 데이터 사전에 저장됩니다. 데이터 파일을 이동하면 사전 데이터가 일관되지 않을 수 있습니다.

데이터 사전 손상 또는 일관성 문제로 인해 InnoDB를 시작할 수 없는 경우 수동 복구에 대한 자세한 내용은 15.21.3절. 'InnoDB 복구 강제 수행'을 참조하세요.

데이터 파일을 열 수 없습니다.

innodb_file_per_table을 사용하도록 설정한 경우(기본값), 테이블별 파일 테이블 스페이스 파일(.ibd 파일)이 누락된 경우 시작 시 다음 메시지가 표시될 수 있습니다:

```
[오류] InnoDB: 파일 작업에서 운영 체제 오류 번호 2가 발생했습니다. [오류] InnoDB: 이 오
류는 시스템이 지정된 경로를 찾을 수 없음을 의미합니다.
[오류] InnoDB: 읽기 전용 데이터 파일을 열 수 없습니다: './test/t1.ibd' OS 오류입니다: 71 [경고]
InnoDB: `test/t1` 테이블스페이스를 열 수 없으므로 무시합니다.
```

이러한 메시지를 해결하려면 DROP TABLE 문을 실행하여 데이터 사전에서 누락된 테이블에 대한 데이터를 제거합니다.

고아 파일-테이블별 ibd 파일 복원하기

이 절차에서는 테이블별 고아 파일 .ibd 파일을 다른 MySQL 인스턴스로 복원하는 방법을 설명합니다. 시스템 테이블 스페이스가 손실되었거나 복구할 수 없는 경우 이 절차를 사용하여 복원할 수 있습니다. .ibd 파일 백업을 새 MySQL 인스턴스에서 수행합니다.

이 절차는 일반 테이블 스페이스 .ibd 파일에는 지원되지 않습니다.

이 절차에서는 .ibd 파일 백업만 있고, 처음에 고아 .ibd 파일을 생성한 것과 동일한 버전의 MySQL로 복구하고 있으며, .ibd 파일 백업이 깨끗한 상태라고 가정합니다. 클린 백업 생성에 대한 자세한 내용은 15.6.1.4절. "InnoDB 테이블 이동 또는 복사"를 참조하세요.

이 절차에는 15.6.1.3절. 'InnoDB 테이블 가져오기'에 설명된 테이블 가져오기 제한 사항이 적용됩니다.

1. 새 MySQL 인스턴스에서 같은 이름의 데이터베이스에 테이블을 다시 만듭니다.

```
mysql> CREATE DATABASE sakila;

mysql> USE sakila;

mysql> CREATE TABLE actor (
         actor_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
         first_name VARCHAR(45) NOT NULL,
         last_name VARCHAR(45) NOT NULL,
         마지막 업데이트 타임스탬프가 널이 아닌 기본 현재_타임스탬프 업데이트시 현재_타임스탬프,
         기본 키(배우_id),
         KEY idx_액터_성명 (성)
       ) 엔진=InnoDB 기본 문자 집합=utf8mb4;
```

2. 새로 생성된 테이블의 테이블 스페이스를 삭제합니다.

```
mysql> ALTER TABLE sakila.actor DISCARD TABLESPACE;
```

3. 백업 디렉터리에서 새 데이터베이스 디렉터리로 고아 `.ibd` 파일을 복사합니다.

```
cp /backup_directory/actor.ibd 경로/to/mysql-5.7/data/sakila/
```

4. `.ibd` 파일에 필요한 파일 권한이 있는지 확인합니다.

5. 고아 .ibd 파일을 가져옵니다. InnoDB가 스키마 확인 없이 파일을 가져오려고 시도하고 있음을 나타내는 경고가 표시됩니다.

```
mysql> ALTER TABLE sakila.actor IMPORT TABLESPACE; SHOW WARNINGS;
쿼리 확인, 영향을 받는 행 0개, 경고 1개(0.15초)

경고 | 1810 | InnoDB: IO 읽기 오류: (2, 해당 파일 또는 디렉터리 없음)
'./sakila/actor.cfg'를 여는 동안 오류가 발생하여 스키마 확인 없이 가져오기
를 시도합니다.
```

6. 테이블을 쿼리하여 .ibd 파일이 성공적으로 복원되었는지 확인합니다.

```
mysql> SELECT COUNT(*) FROM sakila.actor;
+----------+
| count(*) |
+----------+
       |200 |
+----------+
```

15.21.5 InnoDB 오류 처리

다음 항목은 InnoDB가 오류 처리를 수행하는 방법을 설명합니다. InnoDB는 실패한 문만 롤백하는 경우도 있고, 전체 트랜잭션을 롤백하는 경우도 있습니다.

- 테이블 스페이스의 파일 공간이 부족하면 MySQL 테이블이 가득 찼습니다 오류가 발생하고 InnoDB은 SQL 문을 롤백합니다.
- 트랜잭션 교착 상태가 발생하면 InnoDB가 전체 트랜잭션을 롤백합니다. 이 경우 전체 트랜잭션을 다시 시도합니다.

  잠금 대기 시간 초과가 발생하면 InnoDB는 현재 문(잠금을 기다리다가 시간 초과가 발생한 문)을 롤백합니다. 전체 트랜잭션을 롤백하려면 --innodb-rollback-on-timeout을 활성화하여 서버를 시작합니다. 기본 동작을 사용하는 경우 문을 다시 시도하고, --innodb-rollback-on-timeout을 사용하는 경우 전체 트랜잭션을 다시 시도합니다.

  교착 상태와 잠금 대기 시간 초과 모두 바쁜 서버에서 정상적으로 발생하며, 애플리케이션은 교착 상태가 발생할 수 있음을 인지하고 재시도를 통해 처리해야 합니다. 트랜잭션 중 데이터에 대한 첫 번째 변경과 커밋 사이에 가능한 한 적은 작업을 수행하여 잠금이 가능한 한 가장 짧은 시간 동안, 가능한 한 적은 수의 행에 대해 유지되도록 하면 이러한 가능성을 줄일 수 있습니다. 때로는 서로 다른 트랜잭션 간에 작업을 분할하는 것이 실용적이고 도움이 될 수 있습니다.

- 문에 IGNORE 옵션을 지정하지 않은 경우 중복 키 오류가 발생하면 SQL 문이 롤백됩니다.
- 행이 너무 긴 오류는 SQL 문을 롤백합니다.
- 기타 오류는 대부분 MySQL 코드 계층(InnoDB 스토리지 엔진 수준 위)에서 감지되며, 해당 SQL 문을 롤백합니다. 단일 SQL 문을 롤백할 때는 잠금이 해제되지 않습니다.

암시적 롤백 중은 물론 명시적 `ROLLBACK` SQL 문을 실행하는 동안에도 `SHOW PROCESSLIST`는 관련 연결의 상태 열에 롤백을 표시합니다.

15.22 InnoDB 제한

이 섹션에서는 `InnoDB` 테이블, 인덱스, 테이블 스페이스 및 기타 측면에 대한 제한에 대해 설명합니다. `InnoDB` 스토리지 엔진.

- 테이블에는 최대 1017개의 열을 포함할 수 있습니다. 가상으로 생성된 열도 이 제한에 포함됩니다.
- 테이블에는 최대 64개의 보조 인덱스가 포함될 수 있습니다.

- 인덱스 키 접두사 길이 제한은 DYNAMIC 또는 COMPRESSED를 사용하는 InnoDB 테이블의 경우 3072바이트입니다. 행 형식입니다.

  인덱스 키 접두사 길이 제한은 중복 또는 COMPACT 행 형식을 사용하는 InnoDB 테이블의 경우 767바이트입니다. 예를 들어, utf8mb4 문자 집합과 각 문자에 대한 최대 4바이트가 있다고 가정할 때 TEXT 또는 VARCHAR 열의 열 접두사 인덱스가 191자를 초과하는 경우 이 제한에 도달할 수 있습니다.

  제한을 초과하는 인덱스 키 접두사 길이를 사용하려고 하면 오류가 반환됩니다.

  MySQL 인스턴스 생성 시 innodb_page_size 옵션을 지정하여 InnoDB 페이지 크기를 8KB 또는 4KB로 줄이면 16KB 페이지 크기의 경우 3072바이트 제한을 기준으로 인덱스 키의 최대 길이가 비례적으로 줄어듭니다. 즉, 페이지 크기가 8KB인 경우 최대 인덱스 키 길이는 1536바이트, 페이지 크기가 4KB인 경우 768바이트가 됩니다.

  인덱스 키 접두사에 적용되는 제한은 전체 열 인덱스 키에도 적용됩니다.

- 다중 열 인덱스에는 최대 16개의 열이 허용됩니다. 제한을 초과하면 오류가 반환됩니다.

  ```
  오류 1070 (42000): 키 부품이 너무 많이 지정되었습니다. 최대 16개의 부품이 허용됩니다.
  ```

- 페이지 외부에 저장되는 가변 길이 열을 제외한 최대 행 크기는 4KB, 8KB, 16KB 및 32KB 페이지 크기의 경우 페이지의 절반에 약간 못 미칩니다. 예를 들어, 기본 innodb_page_size 16KB의 최대 행 크기는 약 8000바이트입니다. 그러나 InnoDB 페이지 크기가 64KB인 경우 최대 행 크기는 약 16000바이트입니다. LONGBLOB 및 LONGTEXT 열은 4GB 미만이어야 하며, BLOB 및 TEXT 열을 포함한 총 행 크기는 4GB 미만이어야 합니다.

  행의 길이가 반 페이지 미만인 경우 모든 행이 페이지 내에 로컬로 저장됩니다. 페이지의 절반을 초과하는 경우 15.11.2절. '파일 공간 관리'에 설명된 대로 행이 페이지의 절반에 맞을 때까지 가변 길이 열이 외부 페이지 외부 저장소로 선택됩니다.

- InnoDB는 내부적으로 65,535바이트보다 큰 행 크기를 지원하지만, MySQL 자체는 모든 열의 합산 크기에 대해 65,535라는 행 크기 제한을 부과합니다. 섹션 8.4.7, "테이블 열 수 및 행 크기 제한"을 참조하세요.

- 일부 구형 운영 체제에서는 파일이 2GB 미만이어야 합니다. 이는 InnoDB 제한 사항이 아닙니다. 큰 시스템 테이블 스페이스가 필요한 경우 하나의 큰 데이터 파일 대신 여러 개의 작은 데이터 파일을 사용하여 구성하거나 테이블별 파일 및 일반 테이블 스페이스 데이터 파일에 테이블 데이터를 분산합니다.

- InnoDB 로그 파일의 총 최대 크기는 512GB입니다.

- 최소 테이블 스페이스 크기는 10MB보다 약간 큽니다. 최대 테이블 스페이스 크기는 InnoDB 페이지 크기에 따라 다릅니다.

표 15.31 InnoDB 최대 테이블 공간 크기

InnoDB 페이지 크기	최대 테이블 공간 크기

4KB	16TB
8KB	32TB
16KB	64TB
32KB	128TB
64KB	256TB

최대 테이블 스페이스 크기는 테이블의 최대 크기이기도 합니다.

- `InnoDB` 인스턴스는 최대 2^32(4294967296) 테이블 스페이스를 지원하며, 이 중 소수의 테이블 스페이스는 실행 취소 및 임시 테이블용으로 예약되어 있습니다.
- 공유 테이블 공간은 최대 2^32(4294967296) 테이블을 지원합니다.

- 파일 이름을 포함한 테이블스페이스 파일의 경로는 Windows에서 `MAX_PATH` 제한을 초과할 수 없습니다. Windows 10 이전 버전에서는 `MAX_PATH` 제한이 260자였습니다. Windows 10 버전 1607부터 일반적인 Win32 파일 및 디렉터리 함수에서 `MAX_PATH` 제한이 제거되었지만 새로운 동작을 사용하도록 설정해야 합니다.

- 동시 읽기-쓰기 트랜잭션과 관련된 제한에 대해서는 15.6.6절. '로그 실행 취소'를 참조하세요.

15.23 InnoDB 제한 및 제한 사항

이 섹션에서는 `InnoDB` 스토리지 엔진의 제한 사항과 한계를 설명합니다.

- 내부 `InnoDB` 열 이름과 일치하는 열 이름으로 테이블을 생성할 수 없습니다(`DB_ROW_ID`, `DB_TRX_ID` 및 `DB_ROLL_PTR` 포함). 이 제한은 대소문자를 구분하지 않는 이름 사용에 적용됩니다.

```
mysql> CREATE TABLE t1 (c1 INT, db_row_id INT) ENGINE=INNODB;
오류 1166 (42000): 잘못된 열 이름 'db_row_id'
```

- 테이블 상태 표시에서는 테이블이 예약한 물리적 크기를 제외하고 `InnoDB` 테이블에 대한 정확한 통계를 제공하지 않습니다. 행 수는 SQL 최적화에 사용되는 대략적인 추정치일 뿐입니다.

- 동시 트랜잭션이 동시에 다른 수의 행을 '볼' 수 있기 때문에 InnoDB는 테이블의 행 내부 개수를 유지하지 않습니다. 따라서 `SELECT COUNT(*)` 문은 현재 트랜잭션에 표시되는 행의 개수만 계산합니다.

  `InnoDB`가 `SELECT COUNT(*)` 문을 처리하는 방법에 대한 자세한 내용은 `COUNT()` 섹션 12.19.1, '집계 함수 설명'에 설명되어 있습니다.

- 16KB보다 큰 페이지 크기에는 ROW_FORMAT=COMPRESSED가 지원되지 않습니다.

- 특정 `InnoDB` 페이지 크기(`innodb_page_size`)를 사용하는 MySQL 인스턴스는 다른 페이지 크기를 사용하는 인스턴스의 데이터 파일 또는 로그 파일을 사용할 수 없습니다.

- *이동 가능한 테이블 스페이스* 기능을 사용하여 테이블을 가져오는 것과 관련된 제한 사항은 테이블 가져오기 제한 사항을 참조하십시오.

- 온라인 DDL과 관련된 제한 사항은 15.12.8절. "온라인 DDL 제한 사항"을 참조하세요.

- 일반 테이블 스페이스와 관련된 제한 사항은 일반 테이블 스페이스 제한 사항을 참조하십시오.

- 저장 데이터 암호화와 관련된 제한 사항은 암호화 제한 사항을 참조하세요.