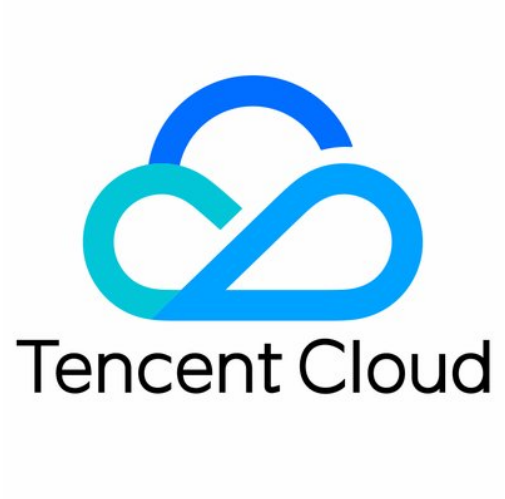
TencentDB for MySQL

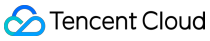
자체개발 커널

제품 문서



저작권 고지

상표 고지



텐센트 클라우드(Tencent Cloud) 및 관련 서비스의 모든 상표는 텐센트(Tencent) 그룹 산하 법인들(모회사, 자회사 및 계열사 포함)이 소유합니다. 본 문서에 언급된 제3자 상표는 해당 법적 권리자가 소유합니다.

서비스 고지

본 문서는 고객에게 텐센트 클라우드(Tencent Cloud) 제품 및 서비스 전부 또는 일부에 대한 현재적 개괄적 정보를 제공 하기 위한 것으로, 특정 제품·서비스의 내용은 수시로 변경될 수 있습니다. 고객이 실제 구매한 제품·서비스의 적용 기준은 고객과 텐센트 클라우드간 체결된 상업계약에 명시된 내용이 우선하며, 별도 서면 합의가 없는 한 텐센트 클라우드는 본 문서 내용에 대해 법적 효력이 있는 명시적·묵시적 진술이나 보증을 일체 하지 않습니다.

목록:

자체개발 커널
TXSQL 커널 개요

최종 업데이트 날짜:: 2024-07-25 16:18:30

TXSQL은 TencentDB 팀에서 유지 관리하는 MySQL 커널 브랜치이며, 네이티브 MySQL과 완벽하게 호환됩니다. 엔터프라이즈급 TDE(Transparent Data Encryption), 감사, 동적 스레드 풀, 암호화 함수, 백업 및 복원, 병렬 쿼리 등 MySQL 엔터프라이즈 에디션과 유사한 다양한 기능을 제공합니다.
TXSQL은 InnoDB 스토리지 엔진, 쿼리 성능 및 복제 성능을 최적화할 뿐만 아니라 TencentDB for MySQL의 사용 용이성과 유지 관리성을 향상시킵니다. MySQL의 모든 이점을 제공하는 동시에 재해 복구, 복원, 모니터링, 성능 최적화, 읽기/쓰기 분리, 투명한 데이터 암호화 및 감사와 같은 엔터프라이즈급 고급 기능을 제공합니다.
다음은 TXSQL에 대한 상세 정보입니다.

- TencentDB for MySQL 커널 버전 업데이트에 대한 자세한 내용은 커널 버전 업데이트 동향을 참고하십시오.
- TencentDB for MySQL의 커널 마이너 버전은 자동 또는 수동으로 업그레이드할 수 있습니다. 자세한 내용은 커널 마이너 버전 업그레이드를 참고하십시오.
- CVM 인스턴스를 사용하여 TencentDB for MySQL 인스턴스에 로그인하고 커널 마이너 버전을 확인할 수 있습니다. 자세한 내용은 커널 마이너 버전 확인을 참고하십시오.

기능적 특성
유휴 트랜잭션 자동 kill

최종 업데이트 날짜:: 2025-02-19 16:10:33

기능 소개

일정 시간을 초과하는 유휴 트랜잭션을 Kill하고 적시에 리소스를 릴리스합니다.

지원 버전

- 커널 버전 MySQL 5.6 20180915 이상
- 커널 버전 MySQL 5.7 20180918 이상
- 커널 버전 MySQL 8.0 20200630 이상

적용 시나리오

트랜잭션 활성화 상태의 연결(begin, start transaction 또는 암시적 활성화 트랜잭션 사용 표시)의 경우 제한 시간 내에 다음 명령이 실행되지 않으면 연결을 kill합니다.

사용 설명

cdb_kill_idle_trans_timeout 매개변수로 이 기능의 활성화 여부를 제어하며, 0은 비활성화, 0 이외의 값은 활성화를 의미하며, session의 wait_timeout 값 보다 작은 값을 사용합니다.

매개변수 이름	동적	유형	기본값	매개변수 값 범위	설명
cdb_kill_idle_trans_timeout	YES	ulong	0	[0, 31536000]	값이 0이면 비활성화를 의미하며, 그렇지 않으면 cdb_kill_idle_trans_timeout 초의 유휴 트랜잭션 kill을 의미합니다.

병렬 복제

최종 업데이트 날짜:: 2024-07-25 16:18:30

기능 소개

MySQL 5.6 이전에는 단일 스레드 모드에서 원본(master) 노드가 binlog를 동기화하고 복제본(slave) 노드가 binlog를 재생합니다. MySQL 5.6 이상 버전은 database 및 logical clock 병렬 복제 체계를 지원하지만 세분성이 너무 커서 대부분의 경우 예상되는 병렬 복제를 수행할 수 없습니다.
Tencent Cloud의 TXSQL 커널 팀은 병렬 복제 체계를 최적화했습니다. 이제 table 병렬 복제가 지원되어 병렬 처리가 향상되고 원본-복제 지연을 줄입니다.

지원 버전

- 커널 버전 MySQL 8.0 20201230 이상
- 커널 버전 MySQL 5.7 20180530 이상
- 커널 버전 MySQL 5.6 20170830 이상

적용 시나리오

이 기능은 일부 로드의 병렬 처리를 최적화하여 복제본(slave) 노드에서 binlog 재생 속도를 높여 원본-복제본 delay를 줄일 수 있는 사용 사례에 적합합니다.

사용 설명

MySQL 5.6 및 5.7의 경우 slave_parallel_type 매개변수를 새로 추가된 TABLE 값으로 설정하여 이 기능을 활성화할 수 있습니다. MySQL 8.0은 TABLE 모드를 지원하지 않습니다.
또한 information_schema 데이터베이스에 cdb_slave_thread_status 테이블이 추가되어 복제본 노드의 스레드 상태를 표시합니다.

MySQL 5.6 매개변수 설명

매개변수 이름	동적	유형	기본값	매개변수 값 범위	설명
slave_parallel_type	YES	char*	SCHEMA	SCHEMA/TABLE	복제본 노드의 병렬 복제 레벨: • SCHEMA: 서로 다른 스키마의 복제 이벤트를 병렬로 실행할 수 있습니다. • TABLE: 서로 다른 테이블의 복제 이벤트를 병렬로 실행할 수 있습니다.

MySQL 5.7 매개변수 설명

매개변수 이름	동적	유형	기본값	매개변수 값 범위	설명
slave_parallel_type	YES	char*	LOGICAL_CLOCK	DATABASE/TABLE/LOGICAL_CLOCK	• 복제본 노드의 병렬 복제 레벨: DATABASE: 서로 다른 데이터베이스의 복제 이벤트를 병렬로 실행할 수 있습니다. • TABLE: 서로 다른 테이블의 복제 이벤트를 병렬로 실행할 수 있습니다. • LOGICAL_CLOCK: 호스트에서 동일한 논리적 클럭의 복제 이벤트를 병렬로 실행할 수 있습니다.

MySQL 8.0 매개변수 설명

매개변수 이름	동적	유형	기본값	매개변수 값 범위	설명
slave_parallel_type	YES	char*	LOGICAL_CLOCK	DATABASE/LOGICAL_CLOCK	복제본 노드의 병렬 복제 레벨: • DATABASE: 서로 다른 데이터베이스의 복제 이벤트를 병렬로 실행할 수 있습니다. • LOGICAL_CLOCK: 호스트에서 동일한 논리적 클럭의 복제 이벤트를 병렬로 실행할 수 있습니다.

returning 지원

최종 업데이트 날짜:: 2024-07-25 16:18:31

기능 소개

일부 사용 시나리오에서는 DML 작업 후 방금 작업된 데이터 행을 반환해야 합니다. 일반적으로 이를 구현하는 방법은 두 가지가 있습니다.

- 첫 번째는 트랜잭션 활성화 후 DML 명령 뒤에 SELECT 명령을 붙이는 것입니다.
- 두 번째는 트리거 등 더 복잡한 작업을 사용하여 달성하는 것입니다.

전자는 주로 SELECT 명령의 오버헤드를 증가시키는 반면 후자는 SQL 구현이 더 복잡하고 유연성은 떨어지도록 만듭니다(트리거 생성 필요).
따라서 RETURNING 구문의 설계는 주로 이 시나리오의 최적화를 목표로 하며, DML 명령 뒤에 RETURNING 키워드를 추가함으로써 상기 요구 사항을 유연하고 효율적으로 구현할 수 있습니다.

지원 버전

커널 버전 MySQL 5.7 20210330 이상

적용 시나리오

현재 MySQL 5.7 20210330 이상의 커널 버전은 INSERT ... RETURNING, REPLACE ... RETURNING, DELETE ... RETURNING을 지원합니다. 이 구문을 사용하면 INSERT/REPLACE/DELETE 명령(단위 statment)으로 작업된 모든 행을 반환할 수 있습니다. 또한, RETURNING은 prepared statements과 저장 과정 중의 사용도 지원합니다.
이 기능을 사용할 때 다음 사항에 주의해야 합니다.

1. RETURNING 사용 시, DELETE...RETURNING 명령은 이전 미러링 이미지 데이터를 반환하고, INSERT/REPLACE...RETURNING 명령은 이후 데이터를 미러링 이미지 데이터를 반환합니다.
2. UPDATE...RETURNING 명령은 현재 지원되지 않습니다.
3. INSERT/REPLACE 시나리오에서 외부 계층 테이블 열은 returning의 서브 쿼리 명령에서 가시적으로 나타나지 않습니다.
4. INSERT/REPLACE의 RETURNING 명령이 last_insert_id()를 반환해야 하는 경우, last_insert_id()의 값은 명령이 성공적으로 실행되기 전의 값입니다. 정확한 last_insert_id() 값을 얻으려면 RETURNING을 사용하여 테이블의 자동 증가 열 ID를 직접 반환하는 것이 좋습니다.

사용 설명

INSERT... RETURNING

```
MySQL [test]> CREATE TABLE `t1` (id1 INT);
Query OK, 0 rows affected (0.04 sec)
```

```
MySQL [test]> CREATE TABLE `t2` (id2 INT);
Query OK, 0 rows affected (0.03 sec)

MySQL [test]> INSERT INTO  t2 (id2) values (1);
Query OK, 1 row affected (0.00 sec)

MySQL [test]> INSERT INTO t1 (id1) values (1) returning *, id1 * 2, id1
+ 1, id1 * id1 as alias, (select * from t2);
+------+---------+---------+-------+--------------------+
| id1  | id1 * 2 | id1 + 1 | alias | (select * from t2) |
+------+---------+---------+-------+--------------------+
|    1 |       2 |       2 |     1 |                  1 |
+------+---------+---------+-------+--------------------+
1 row in set (0.01 sec)

MySQL [test]> INSERT INTO t1 (id1) SELECT id2 from t2 returning id1;
+------+
| id1  |
+------+
|    1 |
+------+
1 row in set (0.01 sec)
```

REPLACE ... RETURNING

```
MySQL [test]> CREATE TABLE t1(id1 INT PRIMARY KEY, val1 VARCHAR(1));
Query OK, 0 rows affected (0.04 sec)

MySQL [test]> CREATE TABLE t2(id2 INT PRIMARY KEY, val2 VARCHAR(1));
Query OK, 0 rows affected (0.03 sec)

MySQL [test]> INSERT INTO t2 VALUES (1,'a'),(2,'b'),(3,'c');
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

MySQL [test]> REPLACE INTO t1 (id1, val1) VALUES (1, 'a');
Query OK, 1 row affected (0.00 sec)

MySQL [test]> REPLACE INTO t1 (id1, val1) VALUES (1, 'b') RETURNING *;
```

```
+------+------+
| id1  | val1 |
+------+------+
|    1 | b    |
+------+------+
1 row in set (0.01 sec)
```

DELETE ... RETURNING

```
MySQL [test]> CREATE TABLE t1 (a int, b varchar(32));
Query OK, 0 rows affected (0.04 sec)

MySQL [test]> INSERT INTO t1 VALUES
    ->    (7,'ggggggg'), (1,'a'), (3,'ccc'),
    ->    (4,'dddd'), (1,'A'), (2,'BB'), (4,'DDDD'),
    ->    (5,'EEEEE'), (7,'GGGGGGG'), (2,'bb');
Query OK, 10 rows affected (0.03 sec)
Records: 10  Duplicates: 0  Warnings: 0

MySQL [test]> DELETE FROM t1 WHERE a=2 RETURNING *;
+------+------+
| a    | b    |
+------+------+
|    2 | BB   |
|    2 | bb   |
+------+------+
2 rows in set (0.01 sec)

MySQL [test]> DELETE FROM t1 RETURNING *;
+------+---------+
| a    | b       |
+------+---------+
|    7 | ggggggg |
|    1 | a       |
|    3 | ccc     |
|    4 | dddd    |
|    1 | A       |
|    4 | DDDD    |
|    5 | EEEEE   |
```

```
|    7 | GGGGGGG |
+------+---------+
8 rows in set (0.01 sec)
```

스토리지 과정

```
MySQL [test]> CREATE TABLE `t` (id INT);
Query OK, 0 rows affected (0.03 sec)

MySQL [test]> delimiter $$
MySQL [test]> CREATE PROCEDURE test(in param INT)
    -> BEGIN
    ->     INSERT INTO t (id) values (param) returning *;
    -> END$$
Query OK, 0 rows affected (0.00 sec)
MySQL [test]> delimiter ;

MySQL [test]> CALL test(100);
+------+
| id   |
+------+
|  100 |
+------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)
```

칼럼 압축

최종 업데이트 날짜:: 2024-07-25 16:18:31

기능 소개

현재 행 형식에 대한 압축과 로그 페이지에 대한 압축이 있지만, 이 두 가지 압축 방법은 테이블의 일부 큰 필드와 기타 많은 작은 필드를 처리하기 때문에 압축과 동시에 작은 필드는 자주 읽기/쓰기 됩니다. 큰 필드의 액세스 빈도가 낮은 경우, 읽기 및 쓰기 액세스 시 컴퓨팅 리소스의 불필요한 낭비가 많이 발생합니다.
칼럼 압축 기능은 액세스 빈도가 높은 큰 필드를 압축하면서 액세스 빈도가 높지 않은 작은 필드는 압축하지 않을 수 있습니다. 이로써 전체 행 필드의 스토리지 공간을 줄일 수 있을 뿐만 아니라 읽기 및 쓰기 액세스 효율을 높일 수 있습니다.
예를 들어, 직원 테이블:

```
create table employee(id int, age int, gender boolean, other varchar(1000) primary key (id))
```

, `id, age, gender` 의 작은 필드에 자주 액세스하는 반면, 'other' 대형 필드에 대한 액세스 빈도가 상대적으로 낮은 경우, 'other' 칼럼은 압축된 칼럼으로 생성될 수 있습니다. 일반적인 상황에서는 `other` 에 대한 읽기 및 쓰기만 칼럼의 압축 및 압축 해제를 트리거하고 다른 칼럼에 대한 액세스는 칼럼의 압축 및 압축 해제를 트리거하지 않습니다. 이를 통해 행 데이터 스토리지 크기를 더욱 줄여 액세스 빈도가 높은 작은 필드 속도를 높이고, 스토리지 용량이 비교적 크고 액세스 빈도가 낮은 필드의 스토리지 용량을 축소할 수 있게 됩니다.

지원 버전

커널 버전 MySQL 5.7 20210330 이상

> **설명:**
> 열 압축 기능은 기본적으로 닫힌 상태입니다. 사용하려면 작업 명세서 제출를 엽니다.

적용 시나리오

테이블에 소량의 큰 필드와 다른 많은 작은 필드가 있는데 작은 필드의 액세스 빈도는 높은 반면 큰 필드 액세스 빈도는 낮은 경우 큰 필드를 압축 칼럼으로 설정할 수 있습니다.

사용 설명

지원되는 데이터 유형

1. `BLOB` (`TINYBLOB` , `MEDIUMBLOB` , `LONGBLOB` 포함)
2. `TEXT` (`TINYTEXT` , `MEDIUMTEXT` , `LONGTEXT` 포함)
3. `VARCHAR`
4. `VARBINARY`

> ⚠ 주의:
> `LONGBLOB` 과 `LONGTEXT` 의 최대 길이는 $2^{32}-2$까지 지원되며, 이는 공식 String Type Storage Requirements가 지원하는 $2^{32}-1$ 보다 1바이트 적습니다.

지원되는 DDL 구문 유형

공식 테이블 구문와 비교했을 때, `column_definition` 의 `COLUMN_FORMAT` 정의가 변경되었습니다. 동시에 칼럼 압축은 Innodb 스토리지 엔진 유형 테이블만 지원됩니다.

```
      column_definition:
        data_type [NOT NULL | NULL] [DEFAULT default_value]
          [AUTO_INCREMENT] [UNIQUE [KEY]] [[PRIMARY] KEY]
          [COMMENT 'string']
          [COLLATE collation_name]
          [COLUMN_FORMAT {FIXED|DYNAMIC|DEFAULT}|COMPRESSED=[zlib]]  #
COMPRESSED 압축된 칼럼 키
          [STORAGE {DISK|MEMORY}]
          [reference_definition]
```

간단한 예시는 다음과 같습니다.

```
CREATE TABLE t1(
  id INT PRIMARY KEY,
  b BLOB COMPRESSED
);
```

이 때, 압축 알고리즘은 생략합니다. 기본적으로 'zlib' 압축 알고리즘이 선택되어 있습니다. 또한 지정된 압축 알고리즘 키워드를 표시할 수도 있습니다. 현재는 'zlib' 압축 알고리즘만 지원됩니다.

```
CREATE TABLE t1(
  id INT PRIMARY KEY,
  b BLOB COMPRESSED=zlib
);
```

지원되는 DDL 구문은 다음과 같이 요약됩니다.
create table 방면:

DDL	압축 속성 상속 여부

`CREATE TABLE t2 LIKE t1;`	Y
`CREATE TABLE t2 SELECT * FROM t1;`	Y
`CREATE TABLE t2(a BLOB) SELECT * FROM t1;`	N

alter table 방면:

DDL	설명
`ALTER TABLE t1 MODIFY COLUMN a BLOB;`	압축 칼럼을 비압축으로 변경
`ALTER TABLE t1 MODIFY COLUMN a BLOB COMPRESSED;`	비압축 칼럼을 압축으로 변경

새 변수 설명

매개변수 이름	동적	유형	기본값	매개변수 값 범위	설명
innodb_column_compression_zlib_wrap	Yes	bool	TRUE	true/false	활성화 시 데이터의 zlib 헤더와 zlib 테일이 생성되고 adler32 인증을 진행합니다.
innodb_column_compression_zlib_strategy	Yes	Integer	0	[0,4]	칼럼 압축에 사용되는 압축 정책. 최소값: 0, 최대값: 4. 0 – 4는 각각 zlib의 압축 정책 Z_DEFAULT_STRATEGY, Z_FILTERED, Z_HUFFMAN_ONLY, Z_RLE 및 Z_FIXED에 해당합니다 . 일반적으로 Z_DEFAULT_STRATEGY는 텍스트 데이터에 가장 적합하고 Z_RLE는 이미지 데이터에 가장 적합합니다.
innodb_column_compression_zlib_level	Yes	Integer	6	[0,9]	칼럼 압축에 사용되는 압축 수준. 최소값: 0, 최대값: 9. 0은 압축하지 않음을 의미하며, 값이 클수록 압축된 더 작게 압축되지만 압축 시간은 길어집니다.
innodb_column_compression_threshold	Yes	Integer	256	[0, 0xffffffff]	칼럼 압축에 사용되는 압축 임계값. 최소값: 1, 최대값: 0xffffffff, 단위: 바이트. 길이가 이 값 이상이면 데이터가 압축되며, 그렇지 않으면 원본 데이터가 변경되지 않고 그대로 유지되고 압축 헤더만 추가됩니다.

innodb_column_compression_pct	Yes	Integer	100	[1, 100]	칼럼 압축에 사용되는 압축률. 최소값: 1, 최대값: 100. 단위: 백분율. **압축 후 데이터 크기 / 압축 전 데이터 크기**가 이 값보다 작으면 데이터가 압축되며, 그렇지 않으면 원본 데이터가 변경되지 않고 그대로 유지되고 압축 헤더만 추가됩니다.

설명:
사용자는 상기 매개변수의 값을 직접 수정할 수 없습니다. 필요 시 티켓 제출을 통해 수정이 가능합니다.

상태 설명 추가

이름	유형	설명
`Innodb_column_compressed`	Integer	칼럼 압축의 압축 횟수. 비압축 형식과 압축 형식의 두 가지 상태 포함
`Innodb_column_decompressed`	Integer	칼럼 압축의 압축 횟수. 비압축 형식과 압축 형식의 두 가지 상태 포함

오류 설명 추가

이름	범위	설명
`Compressed column '%-.192s' can't be used in key specification`	압축 칼럼 이름 지정	인덱싱된 칼럼에 대한 압축 속성 지정 불가
`Unknown compression method: %s"`	DDL 명령에 지정된 압축 알고리즘 이름	`create table` 또는 `alter table` 시 `zlib` 이외의 잘못된 압축 알고리즘 지정
`Compressed column '%-.192s' can't be used in column format specification`	압축 칼럼 이름 지정	동일 칼럼에 `COLUMN_FORMAT` 속성이 지정되어 있으면 압축 속성을 지정할 수 없으며, `COLUMN_FORMAT` 은 NDB에서만 사용됨
`Alter table ... discard/import tablespace not support column compression`	\	칼럼 압축이 있는 테이블은 `Alter table ... discard/import tablespace` 명령 실행 불가

성능

전체 성능은 DDL과 DML의 두 가지 측면으로 나뉩니다.
DDL의 경우 sysbench를 사용하여 테스트합니다:

- 칼럼 압축은 COPY 알고리즘의 DDL 성능에 비교적 큰 영향을 미치며 압축 후 성능은 이전보다 7 – 8배 느려집니다.
- inplace에 미치는 영향은 압축된 데이터의 크기에 따라 달라지며, 압축 후 전체 데이터 크기를 줄이면 DDL의 성능이 향상되고 그렇지 않으면 성능이 일정 수준 저하됩니다.
- instant 에서는, 칼럼 압축은 이러한 유형의 DDL에 거의 영향을 미치지 않습니다.

DML 측면: 가장 일반적인 압축 시나리오(압축 비율 1:1.8)를 고려합니다. 8개의 컬럼을 갖는 테이블이 존재하며, 테이블에 큰 varchar 컬럼이 존재합니다. 삽입된 데이터 길이는 1에서 6000 사이로 균일하게 임의적이며, 삽입된 랜덤 문자는 0–9 및 a – b 사이입니다. 기타 칼럼의 데이터 유형은 char(60) 또는 int입니다. 비압축 칼럼 삽입, 삭제, 쿼리에 대해서는 10% 미만의 향상 효과가 있고, 비압축 칼럼 업데이트의 경우에는 10% 미만의 감소 효과가 있습니다. 압축 칼럼 업데이트의 경우에는 15% 미만의 성능 저하가 있습니다. 이것은 업데이트 프로세스 동안 MySQL이 먼저 행의 값을 읽은 다음 행의 업데이트된 값을 쓰기 때문입니다. 전체 업데이트 프로세스는 압축 해제 및 압축을 트리거하는 반면 삽입 및 쿼리는 한 번만 압축 또는 압축 해제됩니다.

주의 사항

1. 로직 내보내기의 경우, 로직 내보내기 시 create table에는 여전히 Compressed 관련 키워드가 수반됩니다. 따라서 가져올 때 TencentDB for MySQL 내에서 지원됩니다. 기타 MySQL 브랜치 및 공식 버전:
   - 공식 버전 번호 5.7.18 미만이며 바로 가져올 수 있습니다.
   - 공식 버전 번호 5.7.18 이상이며 로직 내보내기 후 압축 키워드를 제거해야 합니다.
2. DTS를 다른 클라우드나 사용자에게 내보낼 때 binlog 동기화 과정 중 비호환성 문제가 발생할 수 있으므로 압축된 키워드를 수반한 DDL 명령을 건너뛸 수 있습니다.

플래시백 쿼리

최종 업데이트 날짜:: 2024-07-25 16:18:30

기능 소개

데이터베이스 운영 과정에서 오작동이 발생할 수 있으며 비즈니스에 심각한 영향을 미칠 수 있습니다. 롤백 및 클론은 오작동에 대한 일반적인 복구 방법이지만 사소한 데이터 변경 및 긴급한 문제 해결 시 오류가 발생하기 쉽고 시간이 많이 소요되며 주요 데이터 변경 사항을 처리할 때 복구 시간을 제어할 수 없습니다.
TXSQL 팀은 Innodb 엔진을 위한 플래시백 쿼리 기능을 개발하고 구현했습니다. 간단한 SQL 문으로 오작동 이전의 과거 데이터를 쿼리하고 특정 SQL 구문을 통해 특정 시점의 데이터를 쿼리할 수 있습니다. 이를 통해 데이터 쿼리 및 복구 시간을 크게 절약하고 신속한 데이터 복구를 통해 비즈니스 연속성을 개선할 수 있습니다.

지원 버전

- 커널 버전: MySQL 5.7 20220715 이상.
- 커널 버전: MySQL 8.0 20220331 이상.

커널 마이너 버전 확인 및 업그레이드 방법에 대한 자세한 내용은 커널 마이너 버전 업그레이드를 참고하십시오.

적용 시나리오

플래시백 쿼리 기능은 데이터베이스 운영 중 오작동이 발생한 후 과거 데이터를 빠르게 쿼리하는 데 사용됩니다. 이 기능을 사용할 때 다음 사항에 주의해야 합니다.

- 플래시백 쿼리는 Innodb 물리적 테이블에 대해서만 지원되며 last_insert_id()와 같은 실제 열이 없는 view, 기타 엔진 또는 함수는 지원되지 않습니다.
- 초(s) 레벨의 플래시백 쿼리만 지원하며 정확도를 완벽하게 보장할 수 없습니다. 1초 내에 여러 변경 사항이 있는 경우 그 중 하나를 반환할 수 있습니다.
- 플래시백 쿼리는 기본 키(또는 GEN_CLUST_INDEX)에 대해서만 지원됩니다.
- 플래시백 쿼리는 prepared statement 또는 stored procedure에서 사용할 수 없습니다.
- 플래시백 쿼리는 DDL을 지원하지 않습니다. 휴지통을 통해 복구해야 하는 truncate table과 같은 테이블에서 DDL을 수행하는 경우 플래시백 쿼리에서 얻은 결과가 예상과 다를 수 있습니다.
- 동일한 구문에서 동일한 테이블에 대해 여러 개의 플래시백 쿼리 시간이 지정된 경우 가장 빠른 시간이 선택됩니다.
- 원본 인스턴스와 복제본 인스턴스의 시간 차이로 인해 Flashback 쿼리에 동일한 시간을 지정하면 인스턴스에 대해 얻은 결과가 다를 수 있습니다.
- 플래시백 쿼리 기능을 활성화하면 실행 취소 로그 정리가 지연되고 메모리 사용량이 증가합니다. 특히 비즈니스 액세스 요청이 빈번한 인스턴스의 경우 Innodb_backquery_window를 큰 값(900에서 1800 사이 권장)으로 설정하지 않는 것이 좋습니다.
- 데이터베이스 인스턴스가 재시작되거나 충돌하는 경우 재시작 또는 충돌 이전의 과거 정보를 쿼리할 수 없습니다. 지정된 시간은 지원 범위(`show status like '%backquery%'` 를 실행하여 상태 변수

Innodb_backquery_up_time 및 Innodb_backquery_low_time을 통해 볼 수 있음)내에 있어야 합니다.

사용 설명

플래시백 쿼리는 새 AS OF 구문을 제공합니다. Innodb_backquery_enable 매개변수를 ON으로 설정하여 플래시백 쿼리 기능을 활성화한 다음 다음 구문을 통해 지정된 시간에 데이터를 쿼리할 수 있습니다.

```
SELECT ... FROM <테이블 이름>
AS OF TIMESTAMP <시간>;
```

지정된 시간의 데이터 쿼리 예시

```
MySQL [test]> create table t1(id int,c1 int) engine=innodb;
Query OK, 0 rows affected (0.06 sec)

MySQL [test]> insert into t1 values(1,1),(2,2),(3,3),(4,4);
Query OK, 4 rows affected (0.01 sec)
Records: 4  Duplicates: 0  Warnings: 0

MySQL [test]> select now();
+---------------------+
| now()               |
+---------------------+
| 2022-02-17 16:01:01 |
+---------------------+
1 row in set (0.00 sec)

MySQL [test]> delete from t1 where id=4;
Query OK, 1 row affected (0.00 sec)

MySQL [test]> select * from t1;
+------+------+
| id   | c1   |
+------+------+
|    1 |    1 |
|    2 |    2 |
|    3 |    3 |
+------+------+
3 rows in set (0.00 sec)
```

```
MySQL [test]> select * from t1 as of timestamp '2022-02-17 16:01:01';
+------+------+
| id   | c1   |
+------+------+
|    1 |    1 |
|    2 |    2 |
|    3 |    3 |
|    4 |    4 |
+------+------+
4 rows in set (0.00 sec)
```

과거 데이터에서 테이블 생성 예시

```
create table t3 select * from t1 as of timestamp '2022-02-17 16:01:01';
```

과거 데이터를 테이블에 삽입하는 예시

```
insert into t4 select * from t1 as of timestamp '2022-02-17 16:01:01';
```

매개변수 설명

다음 표에는 플래시백 쿼리 기능의 구성 가능한 매개변수가 나열되어 있습니다.

매개변수	범위	유형	기본값	값 범위/유효한 값	다시 시작 필요	설명
Innodb_backquery_enable	글로벌	Boolean	OFF	ON\OFF	No	플래시백 쿼리 기능의 스위치입니다.
Innodb_backquery_window	글로벌	Integer	900	1 – 86400	No	플래시백 쿼리의 시간 범위(초)입니다. 이 매개변수의 값이 클수록 플래시백 쿼리에 지원되는 과거 데이터 쿼리 시간이 길어지고 실행 취소 테이블스페이스에서 사용하는 저장 공간이 늘어납니다.
Innodb_backquery_h	글로벌	Integer	8000000	1 – 9223372	No	플래시백 쿼리에 대한 undo 연결 목록의 길이입니다. 이 값을 초과하면

istory_limi t				0368544 76000		Innodb_backquery_window 가 무시되고 이전 연결 목록 길이가 이 값보다 작을 때까지 purge가 트리거됩니다.

성능적 특성
대규모 트랜잭션 복사

최종 업데이트 날짜:: 2024-07-25 16:18:30

기능 소개

row 모드에서는 하나의 명령으로 여러 행을 갱신하는 대규모 트랜잭션이 행당 1개의 event를 생성하는데, 한편으로는 많은 수의 binlog가 생성되는 반면, 복제 시 백업 데이터베이스 apply 속도가 비교적 느려, 결과적으로 백업 데이터베이스 복제가 지연됩니다.
Tencent Cloud 커널 팀은 대규모 트랜잭션 복제 시나리오의 분석 및 최적화를 통해 이 기능을 개발했습니다. 대용량 트랜잭션 복제 최적화 기능은 대용량 트랜잭션을 자동으로 식별하고 row 모드 binlog를 statement 형식 binlog로 변환하여 binlog를 줄이고 복제 효율성을 향상시킵니다.

지원 버전

- 커널 버전 MySQL 5.6 20210630 이상
- 커널 버전 MySQL 5.7 20200630 이상
- 커널 버전 MySQL 8.0 20200830 이상

적용 시나리오

- 이 기능은 주로 row 모드에서 기본 키 테이블이 없는 대용량 트랜잭션의 재생 속도를 향상시킵니다. 기본 키가 없어 딜레이가 발생하는 경우 활성화합니다.
- 이 기능은 주로 row 모드에서 대규모 트랜잭션이 있고 복제가 느린 경우에 적합합니다.

성능 데이터

update 복제 시간은 85%, insert는 약 30% 단축됩니다.

사용 설명

대규모 트랜잭션 복제의 최적화 기능은 SQL의 과거 실행 통계를 기반으로 대규모 트랜잭션일 가능성 여부를 판단하는 것입니다. 대용량 트랜잭션으로 인식되어 최적화될 수 있는 경우 격리 수준을 자동으로 RR(반복 읽기) 수준으로 업그레이드하고 binlog를 Statement 형식으로 완료하여 대용량 트랜잭션 백업 데이터베이스의 실행 시간을 단축시킵니다. 그 중:

- cdb_optimize_large_trans_binlog는 이 기능의 스위치입니다.
- cdb_sql_statistics는 SQL 연산에 대한 통계 스위치입니다.
- cdb_optimize_large_trans_binlog_last_affected_rows_threshold 및 cdb_optimize_large_trans_binlog_aver_affected_rows_threshold는 대규모 트랜잭션에 대한 임계값 조건을 공동 구성합니다.

- cdb_sql_statistics_info_threshold는 메모리에 저장된 과거 통계 데이터의 수입니다.

트랜잭션의 실행을 더 잘 모니터링하기 위해, 트랜잭션 현황 통계 쿼리에 사용할 수 있는 information_schema 데이터베이스의 테이블 CDB_SQL_STATISTICS도 추가되었습니다.

신규 매개변수

이름	상태	유형	기본값	설명
cdb_optimize_large_trans_binlog	true	bool	false	binlog 대규모 트랜잭션 최적화 스위치
cdb_optimize_large_trans_binlog_last_affected_rows_threshold	true	ulonglong	10000	대용량 트랜잭션 최적화 조건: 지난 번 행 수에 영향을 미친 임계값
cdb_optimize_large_trans_binlog_aver_affected_rows_threshold	true	ulonglong	10000	대용량 트랜잭션 최적화 조건: 평균 행 수에 영향을 주는 임계값
cdb_sql_statistics	true	bool	false	SQL 연산에 대한 통계 시작 여부
cdb_sql_statistics_info_threshold	true	ulonglong	10000	CDB_SQL_STATISTICS map에 가장 많은 통계가 저장된 SQL의 수

설명:
사용자는 상기 매개변수의 값을 직접 수정할 수 없습니다. 필요 시 티켓 제출을 통해 수정이 가능합니다.

information_schema.CDB_SQL_STATISTICS 테이블 추가

이름	유형	설명
DIGEST_MD5	MYSQL_TYPE_STRING	해당 SQL의 digest에서 변환된 MD5
DIGEST_TEXT	MYSQL_TYPE_STRING	SQL digest 텍스트 형식
SQL_COMMAND	MYSQL_TYPE_STRING	SQL 명령어 유형
FIRST_UPDATE_TIMESTAMP	MYSQL_TYPE_DATETIME	해당 통계 정보 최초 생성 시간
LAST_UPDATE_TIMESTAMP	MYSQL_TYPE_DATETIME	해당 통계 정보의 마지막 업데이트 시간
LAST_ACCESS_TIMESTAMP	MYSQL_TYPE_DATETIME	해당 통계에 마지막으로 액세스

		한 시간
EXECUTE_COUNT	MYSQL_TYPE_LONGLONG	해당 유형의 SQL 실행 횟수
TOTAL_AFFECTED_ROWS	MYSQL_TYPE_LONGLONG	영향을 받는 총 행 수
AVER_AFFECTED_ROWS	MYSQL_TYPE_LONGLONG	영향을 받는 평균 행 수
LAST_AFFECTED_ROWS	MYSQL_TYPE_LONGLONG	마지막으로 영향을 받은 행 수
STMT_BINLOG_FORMAT_IF_POSSIBLE	MYSQL_TYPE_STRING	해당 유형의 SQL을 statement 형식의 binlog로 완성 가능 여부: TRUE 또는 FALSE

캐시 점검 최적화 플랜

최종 업데이트 날짜:: 2024-07-25 16:18:30

기능 소개

MySQL에서 SQL 문 실행은 구문 분석, 준비, 최적화 및 실행의 네 단계로 나뉩니다. 실행 플랜 캐시 기능은 prepare statement에만 사용할 수 있습니다. 이 기능이 활성화되면 prepare statement를 execute할 때 처음 세 단계를 건너뛰어 쿼리 성능이 크게 향상됩니다.
MySQL 8.0 20210830에서 실행 플랜 캐시는 고유 키(UK) 또는 기본 키(PK)를 사용하는 쿼리에만 적용됩니다. 이후 버전에서 더 많은 유형의 쿼리를 다룰 것입니다.

지원 버전

커널 버전 MySQL 8.0 20210830 이상

적용 시나리오

이 기능은 주로 TencentDB 인스턴스에서 UK 또는 PK로 짧은 prepare statement를 실행할 때 쿼리 성능을 향상시키는 데 사용됩니다. 그러나 성능이 향상될 수 있는 정도는 비즈니스에 따라 다릅니다.

성능 영향

- UK 및 PK SQL 문의 경우 실행 플랜 캐시가 활성화된 후 지연이 20% – 30% 감소하고 처리량이 20% – 30% 향상됩니다(point_select.lua 스크립트를 사용한 sysbench 테스트에 따름).
- 실행 플랜 캐시를 활성화하면 메모리 사용량이 증가합니다.

사용 설명

cdb_plan_cache 매개변수를 사용하여 실행 플랜 캐시를 활성화 또는 비활성화하고 cdb_plan_cache_stats 매개변수를 사용하여 캐시 히트에 대한 정보를 쿼리할 수 있습니다. tencentroot 권한이 있는 계정만 두 매개변수를 사용할 수 있습니다.

매개변수 이름	즉시 효력 발생	유형	기본값	유효한 값/값 범위	설명
cdb_plan_cache	yes	bool	false	true/false	기능을 활성화할지 여부입니다. 기능 권한이 있는 계정만 매개변수를 사용할 수 있습니다.

설명:
현재 위 매개변수의 값을 직접 수정할 수 없습니다. 필요한 경우 티켓 제출하여 도움을 받으십시오.

`show cdb_plan_cache` 명령을 실행하여 실행 플랜 캐시 히트에 대한 정보를 쿼리할 수 있습니다. 이 명령은 다음 필드를 반환합니다.

필드 이름	설명
sql	이 문의 실행 계획이 캐시되었음을 나타내는 물음표(?)가 있는 SQL 문입니다.
mode	SQL 캐시 모드. 현재 prepare 모드만 지원됩니다.
hit	이 세션의 히트 수

cdb_plan_cache_stats가 활성화되면 캐시 히트 정보가 기록되어 데이터베이스 성능에 영향을 미칩니다.

관련 상태 설명

show profile을 실행하여 SQL 문 실행의 각 단계에서 상태를 표시할 수 있습니다. 그러나 실행 계획 캐시에 히트되면 optimizing, statistics, preparing 상태는 생략됩니다.

fdatasync 사용 지원

최종 업데이트 날짜:: 2024-07-25 16:18:31

기능 소개

redo 로그 파일은 fsync 시스템 호출을 사용하여 디스크에 기록되며, 이는 파일 메타데이터 저장 및 파일 데이터 저장을 포함합니다. 파일 메타데이터 정보에는 마지막 수정 시간과 같은 중요도가 높지 않은 정보가 포함되어 있습니다. 일부 redo 저장 시나리오에서 항상 파일 메타데이터를 스토리지 장치에 플러시(fdatasync 시스템 호출 사용)하는 것을 방지해 비용을 줄일 수 있습니다.

지원 버전

- 커널 버전 MySQL 5.7 20201230 이상
- 커널 버전 MySQL 8.0 20201230 이상

적용 시나리오

쓰기 부하가 상대적으로 높은 시나리오에 적합합니다.

성능 데이터

sysbench-write-only의 높은 동시성 연속 쓰기 시나리오에서 TPS 성능이 약 10% 향상됩니다.

사용 설명

innodb_flush_redo_using_fdatasync=true/false 매개변수를 설정하여 redo 로그 파일 메타데이터의 실시간 저장을 피하기 위한 fdatasync 사용 여부를 제어합니다. 기본값은 false입니다.

매개변수 이름	동적	유형	기본값	매개변수 값 범위	설명
innodb_flush_redo_using_fdatasync	yes	bool	false	true/false	fdatasync를 사용한 redo 플러시 여부

자동 증가 칼럼 영속화 지원

최종 업데이트 날짜:: 2024-07-25 16:18:30

기능 소개

자동 증가 값 중복 문제를 피하기 위해 데이터 페이지에 자동 증가 칼럼의 영속화를 구현합니다.

지원 버전

커널 버전 MySQL 5.7 20190830 이상

적용 시나리오

과거 데이터 아카이브와 같은 자동 증가 값 중복 발생을 원하지 않는 시나리오.

사용 설명

커널은 기본적으로 활성화되어 있습니다.

bufferpool 초기화

최종 업데이트 날짜:: 2024-07-25 16:18:30

기능 소개

buffer pool 초기화 속도를 높이고 데이터베이스 인스턴스 실행 소요시간을 줄입니다.

지원 버전

- 커널 버전 MySQL 5.6 20200915 이상
- 커널 버전 MySQL 5.7 20200630 이상

적용 시나리오

데이터베이스 실행 속도 향상에 사용됩니다.

성능 테스트 데이터

8개의 instance가 주어진 경우, 성능 테스트 데이터는 다음과 같습니다:

buffer_pool_size	buffer pool 초기화 시간(최적화 전)	buffer pool 초기화 시간(최적화 후)	속도 향상률
50GB	2.55s	0.13s	1962%
200GB	10.28s	0.52s	1977%
500GB	25.72s	1.32s	1948%

사용 설명

커널은 기본적으로 구현됩니다.

FAST DDL

최종 업데이트 날짜:: 2024-07-25 16:18:30

기능 소개

이 기능은 시간이 많이 소요되는 2단계 인덱스 생성 프로세스를 최적화합니다. 이 기능을 활성화하면 멀티 스레드 동시성을 사용하여 2단계 인덱스 데이터를 외부적으로 정렬하고 flush bulk loading 단계에서 flush list의 잠금 작업을 최적화하여 CREATE INDEX 소요시간과 동시 DML에 대한 영향을 효과적으로 줄입니다.

지원 버전

- 커널 버전 MySQL 8.0 20210330 이상
- 커널 버전 MySQL 5.7 20210331 이상

적용 시나리오

데이터베이스는 DDL 작업을 자주 수행하며, 다음 예시와 같은 DDL 관련 문제가 자주 발생합니다.

- 인덱싱이 인스턴스 지터를 유발하고 정상적인 비즈니스 읽기 및 쓰기에 영향을 미치는 이유는 무엇입니까?
- 1GB 미만의 테이블에 대해 DDL을 실행하는 데 때때로 10분 이상 걸리는 이유는 무엇입니까?
- 임시 테이블을 사용하는 연결이 종료될 때 인스턴스 지터가 발생하는 이유는 무엇입니까?

TXSQL 커널 팀은 위와 같은 공통적인 문제에 대응하여 flush bulk loading 단계에서 flush list의 잠금 작업을 최적화하였으며, CREATE INDEX의 소요시간, 동시 DML에 대한 영향 및 DDL 작업의 영향을 줄였습니다.

성능 데이터

sysbench 테스트는 20억 행의 데이터를 가져왔고 데이터량은 약 453GB이며 FAST DDL 기능이 활성화되어 있습니다.

```
mysql> set global innodb_fast_ddl=ON;
Query OK, 0 rows affected (0.00 sec)
```

전원이 켜질 때까지 4395초가 소요되었으며, 전원이 켜진 후 2455초가 소요되었습니다.

사용 설명

이 기능은 innodb_fast_ddl 매개변수로 활성화/비활성화할 수 있습니다.
innodb_parallel_merge_threads 매개변수는 외부 동시 정렬 프로세스에 사용되는 동시 스레드 수 제어에 사용되며, 기본값은 8, 최댓값은 32입니다.

매개변수 이름	동적	유형	기본값	매개변수 값 범위	설명

innodb_fast_ddl	Yes	bool	OFF	{ON,OFF}	FAST DDL 활성화/비활성화
innodb_parallel_merge_threads	Yes	Integer	8	1 – 32	merge sort 시 사용되는 동시 스레드 수

설명:
사용자는 현재 상기 매개 변수의 매개 변수 값을 직접 수정할 수 없습니다. 만약 수정하려면 작업 명세서 제출 에서 수정할 수 있습니다.

invisible index

최종 업데이트 날짜:: 2024-07-25 16:18:30

기능 소개

많은 사용자는 인덱스를 삭제 가능 여부를 확인하기 위해 인덱스 숨기기 기능이 필요합니다. Invisible Index를 통해 사용자는 인덱스 삭제를 최종 결정하기 전에 응용 프로그램 또는 데이터베이스 사용자가 실제로 사용하는지(오류 발생/보고 여부) 확인할 수 있으며 이 기능은 8.0에서 5.7 버전으로 이식되었습니다.

지원 버전

커널 버전 MySQL 5.7 20180918 이상

적용 시나리오

인덱스 삭제 전, 인덱스를 invisible로 설정하여 인덱스가 유용한지 확인하고 인덱스를 안전하게 삭제할 수 있습니다.

사용 설명

다음 명령을 사용하여 invisible 인덱스를 생성하거나, 기존 인덱스를 invisible로 변경할 수 있습니다.

```
CREATE TABLE t1 (
  i INT,
  j INT,
  k INT,
  INDEX i_idx (i) INVISIBLE
) ENGINE = InnoDB;
CREATE INDEX j_idx ON t1 (j) INVISIBLE;
ALTER TABLE t1 ADD INDEX k_idx (k) INVISIBLE;
```

다음 명령을 사용하여 visible 인덱스로 변경할 수 있습니다.

```
ALTER TABLE t1 ALTER INDEX i_idx INVISIBLE;
ALTER TABLE t1 ALTER INDEX i_idx VISIBLE
```

푸시 다운 계산

최종 업데이트 날짜:: 2024-07-25 16:18:30

기능 소개

이 함수는 단일 테이블 쿼리의 LIMIT/OFFSET 또는 SUM 작업을 InnoDB로 푸시하여 쿼리 딜레이 시간을 효과적으로 줄입니다.

- LIMIT/OFFSET이 2단계 인덱스로 푸시 다운되면, 이 기능은 '테이블로 돌아가기' 작업을 피하고 스캐닝 비용을 효과적으로 절감합니다.
- SUM 작업이 InnoDB로 푸시 다운되면 InnoDB 계층에서 계산이 수행되어 '최종' 결과를 반환하므로 Server 계층과 InnoDB 엔진 계층에서 '각 행' 레코드를 여러 번 스프린트하는 비용을 절약할 수 있습니다.

지원 버전

- LIMIT/OFFSET 최적화에 해당하는 커널 버전 MySQL 5.7 20180530
- SUM 작업 푸시 다운에 해당하는 커널 버전 MySQL 5.7 20180918

적용 시나리오

- 이 기능은 `Select *from tbl Limit 10", "Select* from tbl Limit 10,2` , `Select sum(c1) from tbl` 등의 명령과 같이 단일 테이블 쿼리에 LIMIT/OFFSET 또는 SUM이 있는 시나리오에 주로 사용됩니다.
- 최적화 불가 시나리오:
  - 쿼리 명령에 distinct, group by, having이 있는 경우.
  - 중첩 서브 쿼리가 있는 경우.
  - FULLTEXT 인덱스를 사용하는 경우.
  - order by가 있고 옵티마이저는 index를 사용하여 order by를 구현할 수 없는 경우.
  - 다중 범위 MRR을 사용하는 경우.
  - SQL_CALC_FOUND_ROWS가 있는 경우.

성능 데이터

sysbench가 백만 개의 데이터 행을 가져오기한 후:

- `select * from sbtest1 limit 1000000,1;` 실행 시간이 6.3초에서 2.8초로 감소했습니다.
- `select sum(k) from sbtest1;` 실행 시간이 5.4초에서 1.5초로 감소했습니다.

사용 설명

SQL 실행 중 해당 기능 제어 매개변수의 활성화/비활성화 상황에 따라, 쿼리 옵티마이저는 자동으로 쿼리 플랜을 다시 작성하여 계산 푸시다운의 최적화를 완료합니다.

매개변수는 다음과 같습니다.

매개변수 이름	동적	유형	기본값	매개변수 값 범위	설명
cdb_enable_offset_pushdown	Yes	bool	ON	{ON,OFF}	LIMIT/OFFSET 푸시 다운 제어, 기본 활성화
cdb_enable_sumagg_pushdown	Yes	bool	OFF	{ON,OFF}	SUM 푸시 다운 제어, 기본 비활성화

설명:
사용자는 상기 매개변수의 값을 직접 수정할 수 없습니다. 필요 시 티켓 제출을 통해 수정이 가능합니다.

보안적 특성
투명 데이터 암호화

최종 업데이트 날짜:: 2024-07-25 16:18:30

기능 소개

TXSQL에서는 MySQL의 투명한 암호화 시스템을 계속 사용하여 KEYRING 플러그인의 또 다른 구현을 제공합니다. KEYRING_KMS는 KEYRING을 Tencent Cloud의 엔터프라이즈급 Key Management Service와 통합합니다. Key Management Service(KMS)는 데이터와 키 보안을 보호하는 Tencent Cloud의 핵심 서비스입니다. 서비스 관련 모든 프로세스는 높은 보안성 프로토콜 통신을 채택하여 높은 서비스 보안을 보장하고, 분산형 클러스터 관리 및 핫 백업을 제공하며, 높은 서비스 안정성과 높은 가용성을 보장합니다.
KMS는 두 가지 유형의 키, 즉 사용자 마스터 키(CMK)와 데이터 키(Datakey)를 포함하는 2 계층 키 시스템을 사용합니다. 사용자 마스터 키는 데이터 키 또는 비밀번호, 인증서 및 구성 파일과 같은 작은 데이터 패킷(최대 4KB)을 암호화하는 데 사용됩니다. 데이터 키는 비즈니스 데이터를 암호화하는 데 사용됩니다. 대용량 비즈니스 데이터는 스토리지 또는 통신 과정 중 데이터 키를 사용하여 대칭 암호화 방식으로 암호화되며, 데이터 키는 비대칭 암호화를 사용하여 사용자 마스터 키로 암호화되어 보호됩니다. 2 계층 키 시스템은 데이터가 메모리와 파일에서 암호화되도록 합니다.

지원 버전

- 커널 버전 MySQL 5.7 20171130 이상
- 커널 버전 MySQL 8.0 20200630 이상

적용 시나리오

Transparent Data Encryption(TDE)이란 데이터의 암호화, 복호화 작업을 사용자에게 투명하게 보여주는 것으로, 데이터 파일에 대한 I/O 암호화 및 복호화를 실시간으로 진행합니다. 데이터를 디스크에 입력하기 전에 암호화하고 디스크에서 메모리를 읽을 때 복호화함으로써 정적 데이터 암호화의 컴플라이언스 요구를 충족합니다.

사용 설명

TDE 기능을 활성화하려면 TDE를 참고하십시오.

감사

최종 업데이트 날짜:: 2025-06-05 19:03:07

기능 소개

Tencent Cloud는 TencentDB for MySQL 인스턴스에 대한 데이터베이스 감사 기능을 제공합니다. 데이터베이스에 대한 액세스 및 SQL 명령 실행 상황(명령 활성화 시간, 스캔 행 수, 잠금 대기 시간, CPU 사용 시간, 클라이언트 IP, 사용자 이름, SQL 콘텐츠 등)을 기록하여 기업이 리스크 관리를 수행하고 데이터 보안 수준을 개선할 수 있도록 지원합니다.

사용 사례

이 기능은 데이터베이스가 직면한 리스크를 경고하고, 데이터베이스 SQL 인젝션 및 이상 작업과 같은 데이터베이스 위험 동작을 기록 및 경고해야 하는 시나리오에 적합합니다.

성능 영향

현재 데이터베이스 감사는 동기 감사 모드를 지원합니다. 동기 감사는 모든 감사 로그를 실시간으로 기록하며, 평균 성능 영향은 6% 미만입니다.

사용 설명

MySQL 데이터베이스 감사를 활성화하려면 데이터베이스 감사 활성화를 참조하십시오.

안정적 특성
빠른 칼럼 추가

최종 업데이트 날짜:: 2024-07-25 16:18:31

기능 소개

빠른 칼럼 추가 기능은 이전 칼럼 추가 작업에서 수행해야 하는 데이터 복사를 피하고 데이터 사전을 수정하기만 하면 큰 테이블의 칼럼을 빠르게 추가할 수 있으므로 칼럼을 추가하는 데 필요한 시간을 크게 줄일 수 있습니다. 또한, 큰 테이블과 시스템에 대한 영향을 축소합니다.

지원 버전

- 커널 버전 MySQL 5.7 20190830 이상
- 커널 버전 MySQL 8.0 20200630 이상

적용 시나리오

데이터가 많은 테이블에 칼럼을 추가해야 하는 시나리오에 적합합니다.

성능 데이터

데이터 볼륨이 5GB인 테이블 테스트 결과, 칼럼 추가 작업이 40초에서 1초 미만으로 감소되었습니다.

사용 설명

- Instant Add Column 구문
  Alter Table에 algorithm=instant 절이 추가되었으며, 다음 명령으로 칼럼 추가 연산을 수행할 수 있습니다.

```
ALTER TABLE t1 ADD COLUMN c INT, ADD COLUMN d INT DEFAULT 1000,
ALGORITHM=INSTANT;
```

- inplace 또는 Instant로 설정할 수 있는 매개변수 innodb_alter_table_default_algorithm이 새로 추가되었습니다.
  이 매개변수는 기본적으로 inplace이며 다음과 같이 이 매개변수를 설정하여 Alter Table의 기본 알고리즘을 조정할 수 있습니다. 예시:

```
SET @@global.innodb_alter_table_default_algorithm=instant;
```

  이 매개변수로 기본 알고리즘을 지정하면 알고리즘 미지정 시 테이블 변경 작업에 기본 알고리즘이 사용됩니다.

Instant Add Column 제한

- 문에는 열 추가 작업만 포함될 수 있습니다.
- 마지막에 새로운 열이 추가되며, 열 순서는 변경할 수 없습니다.
- INSTANT ADD COLUMN은 COMPRESSED 행 형식의 테이블에서 지원되지 않습니다.
- INSTANT ADD COLUMN은 전체 텍스트 인덱스가 있는 테이블에서 지원되지 않습니다.
- 임시 테이블에는 INSTANT ADD COLUMN이 지원되지 않습니다.

대형 테이블 비동기 삭제

최종 업데이트 날짜:: 2025-09-25 09:48:29

기능 소개

이 기능은 주로 IO 지터를 피하기 위해 대용량 데이터 파일이 있는 테이블을 삭제하는 데 사용됩니다.
DROP TABLE 명령 실행 시 시스템은 원본 데이터베이스 파일(.ibd)을 새로운 임시 파일로 이름을 변경하고 즉시 작업 성공 메시지를 반환합니다. 이 임시 파일은 매개변수 innodb_async_drop_tmp_dir에 지정된 디렉토리에 저장되며, 백엔드에서 배치로 truncate 처리가 수행됩니다. 각 truncate된 파일 크기는 매개변수 innodb_async_truncate_size(MySQL 5.6 미지원, MySQL 5.7 및 8.0 지원)에 의해 제어됩니다.
해당 기능은 사용자의 별도 조작 없이 커널이 자동으로 수행합니다. 삭제 원리는, 테이블 삭제 시 다른 디렉터리에 테이블 데이터 파일의 하드 링크를 생성하고, DROP TABLE 명령을 실행할 때 삭제되는 것은 이 하드 링크입니다. 이후 백엔드 스레드는 하드 링크 디렉터리를 스캔하여 삭제할 파일을 발견하면, 백엔드에서 drop된 테이블의 데이터 파일을 자동으로 truncate합니다.

지원 버전

- 커널 버전 MySQL 5.6 20220303 이상
- 커널 버전 MySQL 5.7 20230601 이상
- 커널 버전 MySQL 8.0 20200630 이상

적용 시나리오

이 기능은 삭제할 테이블 데이터 파일이 매우 큰 시나리오에 적합합니다.

사용 설명

MySQL 5.6 버전 사용 설명

기능 활성화 방법

1. 비동기적 대형 테이블 삭제 기능을 사용하기 전에, 매개변수 innodb_adaptive_hash_index를 OFF로 설정합니다.
2. 매개변수 innodb_async_truncate_work_enabled를 ON으로 설정하여 비동기적 대형 테이블 삭제 기능을 활성화합니다. 매개변수 설정에 대한 자세한 내용은 인스턴스 매개변수 설정을 참조하십시오.

관련 매개변수 설명

매개변수 이름	동적	유형	기본값	매개변수 값 범위	설명

innodb_adaptive_hash_index	yes	string	ON	ON/OFF	InnoDB 적응형 해시 인덱스 활성화 여부– ON: 활성화– OFF: 비활성화
innodb_async_truncate_work_enabled	yes	string	OFF	ON/OFF	비동기적 대형 테이블 삭제 기능 활성화 여부– ON:활성화– OFF:비활성화

MySQL 5.7 및 MySQL 8.0 버전 사용 설명

기능 활성화 방법

1. 비동기적 대형 테이블 삭제 기능을 사용하기 전에, 매개변수 innodb_fast_ahi_cleanup_for_drop_table을 ON으로 설정합니다.
2. 매개변수 innodb_adaptive_hash_index를 OFF로 설정합니다.
3. 매개변수 innodb_table_drop_mode를 ASYNC_DROP으로 설정하여 비동기적 대형 테이블 삭제 기능을 활성화합니다. 매개변수 설정에 대한 자세한 내용은 인스턴스 매개변수 설정 을 참조하십시오.
4. (선택) 매개변수 innodb_fast_ddl을 ON으로 설정하여 비동기적 대형 테이블 삭제 기능의 효율이 향상됩니다. FAST DDL 기능 및 관련 매개변수에 대한 자세한 내용은 FAST DDL 을 참조하십시오.

관련 매개변수 설명

매개변수 이름	동적	유형	기본 값	매개변수 값 범위	설명
innodb_fast_ahi_cleanup_for_drop_table	yes	string	ON	ON/OFF	적응형 해시 인덱스의 빠른 정리 최적화 활성화 여부– ON: 활성화 (hash 정리 지연으로 인한 대형 테이블 삭제 지연 방지)– OFF: 비활성화
innodb_adaptive_hash_index	yes	string	OFF	ON/OFF	InnoDB 적응형 해시 인덱스 활성화 여부– ON: 활성화– OFF: 비활성화
innodb_table_drop_mode	yes	string	ASYNC_DROP	SYNC_DROP/ASYNC_DROP	비동기적 대형 테이블 삭제 기능 활성화 여부– ASYNC_DROP: 비동기 모드(활성화)– SYNC_DROP: 동기 모드(비활성화)

innodb_async_truncate_size	yes	int	128	128 – 168	비동기적 대형 테이블 삭제 기능에서 백엔드로 truncate 할 때마다 처리하는 파일 크기(단위: MB)

핫스팟 업데이트

최종 업데이트 날짜:: 2024-07-25 16:18:30

기능 소개

빈번한 업데이트 또는 타임 세일 같은 비즈니스 시나리오 적용 시 핫스팟 데이터 update 작업의 성능을 대폭 최적화할 수 있습니다. 핫스팟 업데이트 자동 감지가 활성화되면 시스템은 단일 행 핫스팟 업데이트 여부를 자동으로 감지합니다. 대량의 행 잠금으로 인한 동시 접속 성능 저하를 줄이기 위해 대량의 동시 접속 update가 실행 대기합니다.

지원 버전

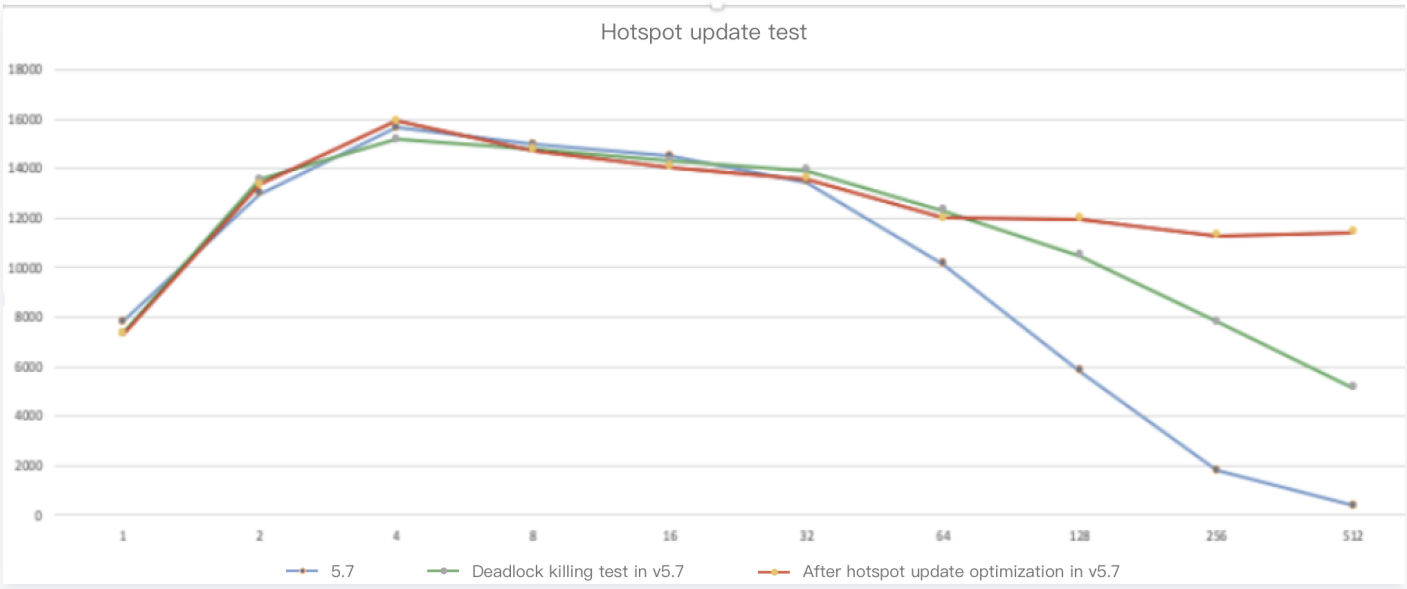
- 커널 버전 MySQL 5.7 20200630 이상
- 커널 버전 MySQL 8.0 20200830 이상

적용 시나리오

싱글포인트 또는 멀티포인트 기본 키의 업데이트 스트레스가 매우 높은 시나리오(타임 세일)에 적합합니다.

성능 데이터

동시성이 높은 환경에서 기본 키 조건의 싱글포인트 동시 update 성능이 10배 이상 향상되었습니다.



사용 설명

핫스팟 업데이트 보호

SQL 스로틀링

최종 업데이트 날짜:: 2024-07-25 16:18:30

기능 소개

키워드 설정을 통해 특정 SQL이 동시 실행할 수 있는 동시성 정도를 제한합니다.

지원 버전

- 커널 버전 MySQL 5.7 20200330 이상
- 커널 버전 MySQL 5.6 20200915 이상

적용 시나리오

동시성이 높고 리소스 점유량이 커서 시스템 성능 저하를 일으키는 일부 SQL.

사용 예시

SQL 스로틀링

statement outline

최종 업데이트 날짜:: 2024-07-25 16:18:30

기능 소개

SQL 최적화는 데이터베이스 성능 최적화의 매우 중요한 부분입니다. 옵티마이저가 적절한 실행 플랜을 선택하지 못해 가져오는 영향을 피하기 위해, TXSQL은 사용자가 실행 플랜을 바인딩할 수 있는 OUTLINE 기능을 제공합니다. MySQL 데이터베이스에는 HINT를 통해 실행 플랜을 수동으로 바인딩하는 기능이 있으며, HINT 정보에는 SQL이 사용하는 최적화 규칙, 알고리즘 종류 및 데이터 스캔에 사용되는 인덱스가 종류가 포함됩니다. OUTLINE은 주로 HINT에 의존하여 쿼리 플랜을 지정합니다. 사용자가 플랜 바인딩 규칙을 추가할 수 있도록 시스템 테이블 mysql.outline을 제공하고, 스위치(cdb_opt_outline_enabled)를 사용하여 이 기능의 활성화 여부를 제어할 수 있도록 합니다.

지원 버전

커널 버전 MySQL 8.0 20201230 이상

적용 시나리오

온라인 실행 플랜에 잘못된 인덱스 선택 등 온라인 실행 플랜이 잘못되었지만, SQL 수정 및 새 버전 배포 없이 해결하고 싶은 경우.

성능 영향

- cdb_opt_outline_enabled가 활성화된 경우, outline에 히트되지 않은 SQL의 실행 효율은 영향을 받지 않습니다.
- outline에 히트된 SQL의 실행 효율은 일반 실행만큼 높지 않지만, 일반 outline 바인딩은 기존 플랜보다 성능이 몇 배는 더 향상됩니다.
- 스위치 사용 시 성능 저하를 유발할 수 있는 바인딩 오류를 방지하기 위해 유지보수 또는 커널 담당자와 상의해야 합니다.

사용 설명

OUTLINE 구문 설정은 다음과 같은 새로운 구문 형식을 채택합니다.

- OUTLINE 정보 설정: `outline "sql" set outline_info "outline";`
- OUTLINE 정보 리셋: `outline reset ""; outline reset all;`
- OUTLINE 정보 플러시: `outline flush;`

다음 schema를 예시로 OUTLINE의 주요 사용법을 설명합니다.

```
create table t1(a int, b int, c int, primary key(a));
create table t2(a int, b int, c int, unique key idx2(a));
```

```
create table t3(a int, b int, c int, unique key idx3(a));
```

매개변수 이름	동적	유형	기본값	매개변수 값 범위	설명
cdb_opt_outline_enabled	yes	bool	fasle	true/false	outline 기능 활성화 여부

설명:
사용자는 상기 매개변수의 값을 직접 수정할 수 없습니다. 필요 시 티켓 제출을 통해 수정이 가능합니다.

OUTLINE 바인딩

OUTLINE을 직접 바인딩하는 방법은 하나의 SQL을 다른 SQL로 교체하는 것이나, SQL의 의미는 바뀌지 않았으며, 옵티마이저에게 실행 방법을 알려주기 위해 일부 HINT 정보만 추가되었습니다.
구문 형식: `outline "sql" set outline_info "outline";` 이며, outline_info 뒤의 문자열은 "OUTLINE:"으로 시작해야 하고, "OUTLINE:" 뒤의 문자열은 HINT를 추가한 후의 SQL입니다. 예를 들어 SQL `select *from t1, t2 where t1.a = t2.a` 의 t2 테이블에 a 열의 인덱스를 추가합니다.

```
outline "select* from t1, t2 where t1.a = t2.a" set outline_info
"OUTLINE:select * from t1, t2 use index(idx2) where t1.a = t2.a";
```

optimizer hint 바인딩

기능을 보다 유연하게 하기 위해 TXSQL은 SQL에 optimizer hint를 점진적으로 추가할 수 있으며, outline을 직접 바인딩하여 동일한 기능을 구현할 수 있습니다.
구문 형식은 `outline "sql" set outline_info "outline";` 이며, outline_info 뒤의 문자열은 "OPT:"로 시작해야 하고 "OPT:" 뒤에는 optimizer hint 정보를 추가합니다. 예를 들어, s `elect *from t1 where t1.a in (select b from t2)` 의 SQL에 대해 MATERIALIZATION/DUPSWEEDOUT의 SEMIJOIN을 지정합니다.

```
outline "select* from t1 where t1.a in (select b from t2)" set
outline_info "OPT:2#qb_name(qb2)";
outline "select * from t1 where t1.a in (select b from t2)" set
outline_info "OPT:1#SEMIJOIN(@qb2 MATERIALIZATION, DUPSWEEDOUT)";
```

기존 SQL 명령에 OPTIMIZER HINT를 추가하면 한 번에 하나의 HINT만 추가할 수 있습니다. 구문의 세 가지 사항에 유의해야 합니다.

- OPT 키워드는 " 바로 뒤에 와야 합니다.

- 바인딩할 새로운 명령은 ':'가 앞에 와야 합니다.
- 두 개의 필드(query block 번호 #optimizer hint 문자열)를 추가해야 하며 둘 사이는 #으로 구분해야 합니다(`ie. "OPT:1#max_execution_time(1000)"`).

index hint 바인딩

기능을 보다 유연하게 하기 위해 TXSQL은 SQL에 index hint를 점진적으로 추가할 수 있으며, outline을 직접 바인딩하여 동일한 기능을 구현할 수 있습니다.
구문 형식은 `outline "sql" set outline_info "outline";` 이며, outline_info 뒤의 문자열은 "INDEX:"로 시작해야 하고 "INDEX:" 뒤에는 index hint 정보를 추가합니다.
예시는 다음과 같습니다. SQL

```
select *from t1 where t1.a in (select t1.a from t1.b in (select t1.a from t1 left join t2 on t1.a = t2.a))
```

의 query block 3 데이터베이스 test 아래에 있는 t1 테이블에 USE INDEX의 인덱스 idx1을 추가합니다. 유형은 FOR JOIN입니다.

```
outline "select* from t1 where t1.a in (select t1.a from t1 where t1.b
in (select t1.a from t1 left join t2 on t1.a = t2.a))" set outline_info
" INDEX:3#test#t1#idx1#1#0";
```

INDEX의 HINT를 기존 SQL 명령에 추가합니다. 한 번에 하나의 HINT만 추가할 수 있습니다. 구문의 네 가지 사항에 유의해야 합니다.

- INDEX 키워드는 " 바로 뒤에 와야 합니다.
- 바인딩할 새로운 명령은 ':'가 앞에 와야 합니다.
- 5개의 필드(query block 번호 #db_name#table_name#index_name#index_type#clause)를 추가해야 합니다.
- Index_type에 3개의 값(0은 INDEX_HINT_IGNORE, 1은 INDEX_HINT_USE, 2는 INDEX_HINT_FORCE)이, clause에 3개의 값(1은 FOR JOIN, 2는 FOR ORDER BY, 3은 FOR GROUP BY)이 있어야 합니다. 둘 사이는 #으로 나눕니다(`ie. "INDEX:2#test#t2#idx2#1#0"` 은 두 번째 query block의 test.t2 테이블에서 USE INDEX FOR JOIN 유형의 idx1 인덱스를 바인딩하는 것을 의미합니다).

특정 SQL에 해당하는 OUTLINE 정보 삭제

TXSQL은 사용자가 특정 SQL 명령의 OUTLINE 바인딩 정보를 삭제할 수 있도록 허용합니다.
구문: `outline reset "sql";` . `select *from t1, t2 where t1.a = t2.a` 의 outline 정보 삭제 예시: `outline reset "select* from t1, t2 where t1.a = t2.a";` .

모든 OUTLINE 정보 지우기

TXSQL은 사용자가 커널에서 모든 OUTLINE 바인딩 정보를 삭제할 수 있도록 허용합니다. 구문은 `outline reset all` 이고 실행 명령은 `outline reset all;` 입니다.

때때로 온라인 비즈니스에서 인덱스에 바인딩해야 하는 매우 구체적인 문제가 있습니다. 여기에서 OUTLINE을 바인딩하도록 직접 설정할 수 있습니다.
OUTLINE 설정 후 발생할 수 있는 성능 롤백을 분석하고, 허용 가능한 성능 롤백 범위 내에서 바인딩합니다. 필요에 따라 커널 담당자와 논의해야 합니다.

관련 매개변수 상태 설명

TXSQL은 다양한 방식으로 사용자의 SQL을 볼 수 있는 OUTLINE 바인딩을 제공하며, 우선 mysql.outline 테이블을 통해 사용자 OUTLINE 설정을 확인할 수 있습니다. 그런 다음 show cdb_outline_info와 select * from information_schema.cdb_outline_info 두 인터페이스를 통해 메모리에 있는 OUTLINE 정보를 볼 수 있습니다.
SQL 변경 여부는 메모리 안에 OUTLINE 정보 존재 여부에 달려 있기 때문에 사용자는 상기 두 인터페이스로 디버깅할 수 있습니다.
mysql.outline 시스템 테이블이 새로 추가되었으며 사용자가 설정한 OUTLINE 정보 레코드가 이 테이블에 저장됩니다. 테이블 필드는 다음과 같습니다.

필드 이름	설명
Id	OUTLINE 정보 설정 번호
Digest	원본 SQL 명령 해시 값
Digest_text	원본 SQL 명령의 지문 정보 텍스트
Outline_text	OUTLINE이 바인딩된 SQL 명령의 지문 정보 텍스트

show cdb_outline_info 또는 select * from information_schema.cdb_outline_info를 통해 메모리에 있는 레코드를 볼 수 있으며, SQL이 OUTLINE 레코드 바인딩 플랜을 실행합니다. 매개변수는 다음과 같습니다.

필드 이름	설명
origin	원본 SQL 명령 지문
outline	OUTLINE이 바인딩된 SQL 명령 지문