

MySQL 관리용 SQL

- [Metadata Lock 모니터링](#)
- [쿼리 프로파일링](#)
- [데이터 파일 사이즈 계산](#)
- [RDS MySQL 함수 확인](#)

Metadata Lock 모니터링

Metadata Lock 모니터링

```
SELECT b.OBJECT_TYPE, b.OBJECT_SCHEMA, b.OBJECT_NAME,
b.LOCK_TYPE, b.LOCK_STATUS, c.THREAD_ID, c.PROCESSLIST_ID,c.PROCESSLIST_INFO
FROM performance_schema.metadata_locks a JOIN performance_schema.metadata_locks b
ON a.OWNER_THREAD_ID<>b.OWNER_THREAD_ID AND a.OBJECT_NAME=b.OBJECT_NAME
AND a.LOCK_STATUS = 'PENDING'
JOIN performance_schema.threads c
ON b.OWNER_THREAD_ID = c.THREAD_ID;
```

쿼리 프로파일링

- 1. 프로파일링 활성화:
SET profiling = 1;
- 2. 원하는 쿼리 실행: 예를 들면:
SELECT * FROM your_table WHERE some_column = 'some_value';
- 3. 프로파일 결과 확인:
SHOW PROFILES;
이 명령은 실행한 쿼리의 목록을 보여줍니다. 각 쿼리는 고유한 쿼리 ID를 가지며, 이 ID를 사용하여 세부 프로파일 정보를 볼 수 있습니다.
- 4. 특정 쿼리의 프로파일 세부 정보 확인: 예를 들어 쿼리 ID가 1인 경우:
SHOW PROFILE FOR QUERY 1;
이 명령은 쿼리의 실행 과정에서 발생하는 각 단계별로 소요된 시간을 보여줍니다.
- 5. 프로파일링 비활성화: 프로파일링을 완료한 후에는 프로파일링을 비활성화해야 합니다:
SET profiling = 0;

예시)

```
mysql> set profiling=1;
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> select * from a order by a desc limit 20;
+-----+-----+-----+-----+-----+
| a      | b      | c      | d      | e      | f      |
+-----+-----+-----+-----+-----+
| 2424796 | aasdwt | niowqt | niowqt | niowqt | niowqt |
| 2424795 | aasdwt | niowqt | niowqt | niowqt | niowqt |
| 2424794 | aasdwt | niowqt | niowqt | niowqt | niowqt |
| 2424793 | aasdwt | niowqt | niowqt | niowqt | niowqt |
| 2424792 | aasdwt | niowqt | niowqt | niowqt | niowqt |
| 2424791 | aasdwt | niowqt | niowqt | niowqt | niowqt |
| 2424790 | aasdwt | niowqt | niowqt | niowqt | niowqt |
| 2424789 | aasdwt | niowqt | niowqt | niowqt | niowqt |
| 2424788 | aasdwt | niowqt | niowqt | niowqt | niowqt |
| 2424787 | aasdwt | niowqt | niowqt | niowqt | niowqt |
| 2424786 | aasdwt | niowqt | niowqt | niowqt | niowqt |
| 2424785 | aasdwt | niowqt | niowqt | niowqt | niowqt |
| 2424784 | aasdwt | niowqt | niowqt | niowqt | niowqt |
| 2424783 | aasdwt | niowqt | niowqt | niowqt | niowqt |
| 2424782 | aasdwt | niowqt | niowqt | niowqt | niowqt |
| 2424781 | aasdwt | niowqt | niowqt | niowqt | niowqt |
| 2424780 | aasdwt | niowqt | niowqt | niowqt | niowqt |
| 2424779 | aasdwt | niowqt | niowqt | niowqt | niowqt |
| 2424778 | aasdwt | niowqt | niowqt | niowqt | niowqt |
| 2424777 | aasdwt | niowqt | niowqt | niowqt | niowqt |
+-----+-----+-----+-----+-----+
20 rows in set (0.02 sec)

mysql> show profiles;
+-----+-----+-----+
| Query_ID | Duration | Query |
+-----+-----+-----+
| 1 | 0.01157325 | select * from a order by a desc limit 20 |
+-----+-----+-----+
1 row in set, 1 warning (0.00 sec)

mysql> show profile for query 1;
+-----+-----+
| Status | Duration |
+-----+-----+
| starting | 0.002565 |
| checking permissions | 0.000406 |
| Opening tables | 0.000430 |
| init | 0.000609 |
| System lock | 0.000282 |
| optimizing | 0.000098 |
| statistics | 0.000302 |
| preparing | 0.002321 |
```

```
| Sorting result      | 0.000119 |
| executing           | 0.000033 |
| Sending data        | 0.003081 |
| end                 | 0.000120 |
| query end           | 0.000189 |
| closing tables      | 0.000128 |
| freeing items       | 0.000433 |
| cleaning up         | 0.000458 |
+-----+-----+
16 rows in set, 1 warning (0.03 sec)
```

데이터 파일 사이즈 계산

(EC2에 깔려있다면 OS상에서 데이터 파일 사이즈 보는게 더 정확함)

```
SELECT
    table_name AS `Table`,
    round((((data_length + index_length) / 1024 / 1024), 2) AS `Size (MB)`
FROM
    information_schema.TABLES
WHERE
    table_schema = "your_database_name"
    AND table_name = "your_table_name";
```

메타데이터만 가지고 계산하는거라 100% 정확한 값인 것은 아님

RDS MySQL 함수 확인

RDS에 기본적으로 생성되어있는 MySQL RDS 관리용 함수 이름 확인 정확한 함수 명칭이 헛갈리면 여기서 확인하면 됨

```
SELECT
  ROUTINE_NAME
FROM
  information_Schema.routines
WHERE
  routine_schema = 'mysql'
  AND routine_name LIKE 'rds_%';

+-----+
| ROUTINE_NAME |
+-----+
| rds_assign_gtid_to_anonymous_transactions |
| rds_collect_global_status_history |
| rds_disable_gsh_collector |
| rds_disable_gsh_rotation |
| rds_enable_gsh_collector |
| rds_enable_gsh_rotation |
| rds_external_master |
| rds_external_source |
| rds_import_binlog_ssl_material |
| rds_kill |
| rds_kill_query |
| rds_next_master_log |
| rds_next_source_log |
| rds_remove_binlog_ssl_material |
| rds_reset_external_master |
| rds_reset_external_source |
| rds_rotate_general_log |
| rds_rotate_global_status_history |
| rds_rotate_slow_log |
| rds_set_configuration |
| rds_set_external_master |
| rds_set_external_master_with_auto_position |
| rds_set_external_source |
| rds_set_external_source_with_auto_position |
| rds_set_gsh_collector |
| rds_set_gsh_rotation |
| rds_set_master_auto_position |
| rds_set_source_auto_position |
| rds_show_configuration |
| rds_skip_repl_error |
| rds_skip_transaction_with_gtid |
| rds_start_replication |
| rds_stop_replication |
+-----+
33 rows in set (0.01 sec)

mysql>
```

show create procedure 명령으로 어떻게 만들어진 함수인지도 확인 가능함.

```
mysql> show create procedure mysql.rds_kill\G
***** 1. row *****
      Procedure: rds_kill
      sql_mode:
      Create Procedure: CREATE DEFINER=`rdsadmin`@`localhost` PROCEDURE `rds_kill`(IN thread BIGINT)
      READS SQL DATA
      DETERMINISTIC
      BEGIN
      DECLARE l_user varchar(16);
      DECLARE l_host varchar(64);
```

```

DECLARE foo varchar(255);

SELECT user, host INTO l_user, l_host
FROM information_schema.processlist
WHERE id = thread;

IF l_user = 'rdsadmin' and l_host like 'localhost%' THEN
select `ERROR (RDS): CANNOT KILL RDSADMIN SESSION` into foo;
ELSEIF l_user = 'rdsrepladmin' THEN
select `ERROR (RDS): CANNOT KILL RDSREPLADMIN SESSION` into foo;
ELSE
KILL thread;
END IF;
END

character_set_client: utf8
collation_connection: utf8_general_ci
Database Collation: utf8mb4_0900_ai_ci
1 row in set (0.00 sec)

```

각 함수에 대한 설명은 https://docs.aws.amazon.com/ko_kr/AmazonRDS/latest/UserGuide/Appendix.MySQL.SQLRef.html 페이지에서 확인 가능

구성

바이너리 로그 설정과 관련된 함수. 분명 원래는 더 많은 설정값과 관련된 함수들을 준비할 생각이지 않았을까 싶은데.. 실제로 제대로 동작하는건 바이너리로그 리텐션타임 밖에 없는 듯

mysql.rds_set_configuration

- 용도: 바이너리 로그 보관 시간 설정. 기본적으로 (binlog retention hours, null)로 설정되어 있는데, 이 경우 바이너리 로그가 필요없어지면 최대한 빨리 삭제함. 이 설정을 통해 값을 지정하면 바이너리 로그의 보관 기간이 더 길게 유지할 수 있음
- 용도2: (기본적으로 이 함수는 아래 바이너리 로그 보관시간 확인하는 함수랑 한 세트로, 바이너리 로그 보관 시간 설정하는 용도가 일순위인 것 같은데...) 이 함수를 통해 source delay, target delay라는 설정을 할 수도 있음. 현대 결국은 다른 함수를 추가 호출해서 구현된다고함. 자세한 내용은 [AWS Documentation](#) 참고
- 사용법: call mysql.rds_set_configuration(이름,값=숫자); 이름 - binlog retention hours = 바이너리 로그를 유지할 시간. 3이면 3시간 보관하는 식. source delay = 복제 원복측 (=마스터)에서 복제를 지연시킬 시간. 초단위. target delay = 복제 타겟측 (=슬레이브)에서 복제를 지연시킬 시간. 초단위.

```

mysql> call mysql.rds_set_configuration('binlog retention hours', 24);
Query OK, 0 rows affected (0.00 sec)

```

-> 결과 확인은 하단의 mysql.rds_show_configuration 함수로 확인하면 됨

- 주의: 이름을 문자열로 받는데 예외처리같은게 되어있는것도 아니라, 아무 이름이나 막 넣어도 함수 실행에 성공함

```

mysql> call mysql.rds_set_configuration('hello db.tips', 12345);
Query OK, 0 rows affected (0.00 sec)
-- 이렇게 아무렇게나 막 값을 넣어도 에러가 안난다.

```

- 주의2: Documentation상의 설명과 조금 다른 부분이 있는 것 같음

```

mysql> show create procedure mysql.rds_set_configuration\G
***** 1. row *****
      Procedure: rds_set_configuration
      sql_mode: NO_ENGINE_SUBSTITUTION
      Create Procedure: CREATE DEFINER=`rdsadmin`@`localhost` PROCEDURE `rds_set_configuration`(IN name VARCHAR(30), IN value INT)
      READS SQL DATA
      DETERMINISTIC
      BEGIN
        DECLARE sql_logging BOOLEAN;
        IF name = 'binlog retention hours' AND value < 1 THEN
          SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'For binlog retention hours the value must be between 1 and 2160 inclusive or be NULL';
        END IF;
        IF name = 'binlog retention hours' AND value > 2160 THEN
          SET value = 2160;
        END IF;

```

```

SELECT @@sql_log_bin INTO sql_logging;
SET @@sql_log_bin = OFF;
UPDATE mysql.rds_configuration
SET mysql.rds_configuration.value = value
WHERE BINARY mysql.rds_configuration.name = BINARY name;
SET @@sql_log_bin = sql_logging;
END

```

character_set_client: utf8mb4

collation_connection: utf8mb4_0900_ai_ci

Database Collation: utf8mb4_0900_ai_ci

1 row in set (0.00 sec)

-- 설명과 달리 함수 생성 구문 내에는 binlog retention hours에 대한 내용밖에 없다. 따라서 source delay / target delay가 정확히 동작하는지 알 수 없다. 함수 생성 구문만 보면 왜 오타에 에러가 안나는지도 알 것 같고..

-- 그리고 다크멘테이션엔 168까지 설정 가능하다고 하는데, 함수엔 2160까지 되는 것 처럼 보인다. 각 DB의 사이즈나 버전 같은것에 따라서 설정이 달라지는게 아닐까 싶으니 확인 잘 해보고 써야될 것 같다.

mysql.rds_show_configuration

- 용도: 바이너리 로그 보관 시간 확인
- 사용법: call mysql.rds_show_configuration;

```
mysql> call mysql.rds_show_configuration;
```

```

+-----+-----+-----+-----+
| name          | value | description                                                                 |
+-----+-----+-----+-----+
| binlog retention hours | 24   | binlog retention hours specifies the duration in hours before binary logs are automatically deleted. |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

```

-> 위에서 rds_set_configuration 함수 수행한대로 24로 변경되어 있음

-- 원복함

```
mysql> call mysql.rds_set_configuration('binlog retention hours', null);
```

Query OK, 0 rows affected (0.01 sec)

```
mysql> call mysql.rds_show_configuration;
```

```

+-----+-----+-----+-----+
| name          | value | description                                                                 |
+-----+-----+-----+-----+
| binlog retention hours | NULL  | binlog retention hours specifies the duration in hours before binary logs are automatically deleted. |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

```

Query OK, 0 rows affected (0.00 sec)

-- 함수 실행 잘 되고, 다시 null로 바뀐 것도 확인

세션 또는 쿼리 종료

개인적으로, 실질적으로 이 중에 제일 많이 쓰는 함수. kill id, kill query id의 함수판

mysql.rds_kill

- 용도: 연결된 세션을 끊음.
- 사용법: call mysql.rds_kill(프로세스ID);

```
mysql> SHOW FULL PROCESSLIST;
```

```

+---+-----+-----+-----+-----+-----+-----+-----+
| Id | User   | Host      | db  | Command | Time | State      | Info                                |
+---+-----+-----+-----+-----+-----+-----+-----+
| 5  | rdsadmin | localhost | NULL | Sleep   | 11   |           | NULL                                |
| 6  | my_user  | localhost | mydb | Query   | 10   | Sending data | SELECT * FROM big_tbl |
+---+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

```

```
mysql> CALL mysql.rds_kill(6);
```



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

```
mysql> SHOW FULL PROCESSLIST;
```

```
+---+-----+-----+-----+-----+-----+-----+
| Id | User   | Host   | db   | Command | Time | State | Info |
+---+-----+-----+-----+-----+-----+-----+
| 5 | rdsadmin | localhost | NULL | Sleep   | 16 |      | NULL |
+---+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

-> 6번 세션이 죽었음

mysql.rds_kill_query

- 용도: 쿼리를 수행 중인 세션의 쿼리만 죽임. 세션 연결은 그냥 남아 있음
- 사용법: `call mysql.rds_kill_query(프로세스ID);`

```
mysql> SHOW FULL PROCESSLIST;
```

```
+---+-----+-----+-----+-----+-----+-----+
| Id | User   | Host   | db   | Command | Time | State      | Info |
+---+-----+-----+-----+-----+-----+-----+
| 5 | rdsadmin | localhost | NULL | Sleep   | 11 |          | NULL |
| 6 | my_user  | localhost | mydb | Query   | 10 | Sending data | SELECT * FROM big_tbl |
+---+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

```
mysql> CALL mysql.rds_kill_query(6);
```

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

```
mysql> SHOW FULL PROCESSLIST;
```

```
+---+-----+-----+-----+-----+-----+-----+
| Id | User   | Host   | db   | Command | Time | State | Info |
+---+-----+-----+-----+-----+-----+-----+
| 5 | rdsadmin | localhost | NULL | Sleep   | 16 |      | NULL |
| 6 | my_user  | localhost | mydb | Sleep   | 0 |      | NULL |
+---+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

-> 6번 세션의 연결은 그냥 남아있고, 수행중이던 쿼리만 끝났음

로깅

로깅 테이블 스왑

mysql.rds_rotate_general_log

- 용도: 제너럴 로그 로테이션. `mysql.general_log` <-> `mysql.general_log_backup` 두 테이블간 전환
- 사용법: `CALL mysql.rds_rotate_general_log;`

```
mysql> CALL mysql.rds_rotate_general_log;
```

```
Query OK, 0 rows affected (0.06 sec)
```

mysql.rds_rotate_slow_log

- 용도: 슬로우 로그 로테이션. `slow_log` <-> `slow_log_backup` 두 테이블간 전환
- 사용법: `CALL mysql.rds_rotate_slow_log;`

```
mysql> CALL mysql.rds_rotate_slow_log;
```

```
Query OK, 0 rows affected (0.06 sec)
```

-> 둘다 마찬가지로, 로그를 파일로 쌓고 있는 경우엔 딱히 의미있는 함수는 아님. `log_output=TABLE`로 되어 있을 경우엔 영향이 있을 것임. 스왑 해놓고 실제로 데이터를 백업하거나, 가져다 쓰거나 그런 작업은 직접 해야됨. 이 함수는 테이블 rename만 해줌

전역적 상태 이력 관리

GSH(=Global status history)에 대한 함수들. show global status 명령의 결과에 대한 히스토리를 수집해주는 기능. 이벤트 스케줄러를 이용하는 기능이기 때문에, event_scheduler=on으로 설정되어 있어야 동작함

- GSH collect와 rotate에 대한 함수들로 이루어져 있고 =2
- 각각을 당장 수행하는 것 / 자동화 구성 하는 것 / 그걸 켜고, 끄는 것 = 4
- 2 x 4 = 8개의 함수로 되어있음 Aurora v3 (= MySQL 8.0)에서는 information_schema.GLOBAL_STATUS, information_schema.GLOBAL_VARIABLES 테이블들이 사라져서 아래 함수들이 더이상 동작하지 않음. 최신 버전에 맞춰서 함수가 수정되거나 하지는 않은 듯. 지금으로선 사실상 없어진 기능인듯...

mysql.rds_set_gsh_collector

- 용도: GSH 스냅샷을 남기는 주기 설정.
- 사용법: call mysql.rds_set_gsh_collector(시간=분단위);

```
mysql> call mysql.rds_set_gsh_collector(2);
```

```
Query OK, 0 rows affected (0.02 sec)
```

mysql.rds_collect_global_status_history

- 용도: GSH를 수집
- 사용법: call mysql.rds_collect_global_status_history;

```
mysql> call mysql.rds_collect_global_status_history;
```

```
Query OK, 0 rows affected (0.02 sec)
```

mysql.rds_enable_gsh_collector

- 용도: GSH 자동 수집 기능 활성화. 주기 지정 안하면 5분 간격으로 설정됨
- 사용법: call mysql.rds_enable_gsh_collector;

```
mysql> call mysql.rds_enable_gsh_collector;
```

```
Query OK, 0 rows affected (0.02 sec)
```

mysql.rds_disable_gsh_collector

- 용도: GSH 자동 수집 기능 비활성화.
- 사용법: call mysql.rds_disable_gsh_collector;

```
mysql> call mysql.rds_disable_gsh_collector;
```

```
Query OK, 0 rows affected (0.02 sec)
```

mysql.rds_set_gsh_rotation

- 용도: GSH 테이블 로테이션 주기 설정. 로테이션이 수행되면 mysql.rds_global_status_history -> mysql.rds_global_status_history_old로 변경하고, mysql.rds_global_status_history테이블 새로 생성
- 사용법: call mysql.rds_set_gsh_rotation(시간=일단위);

```
mysql> call mysql.rds_set_gsh_rotation(14);
```

```
Query OK, 0 rows affected (0.02 sec)
```

mysql.rds_rotate_global_status_history

- 용도: GSH 테이블 로테이션 수행
- 사용법: call mysql.rds_rotate_global_status_history;

```
mysql> call mysql.rds_rotate_global_status_history;
```

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

mysql.rds_enable_gsh_rotation

- 용도: GSH 테이블 자동 로테이션 기능 활성화. 주기 지정 안하면 7일 간격으로 설정됨.
- 사용법: call mysql.rds_enable_gsh_rotation;

```
mysql> call mysql.rds_enable_gsh_rotation;
```

```
Query OK, 0 rows affected (0.02 sec)
```

mysql.rds_disable_gsh_rotation

- 용도: GSH 테이블 자동 로테이션 기능 비활성화.
- 사용법: call mysql.rds_disable_gsh_rotation;

```
mysql> call mysql.rds_disable_gsh_rotation;
```

```
Query OK, 0 rows affected (0.02 sec)
```

복제

리플리케이션과 관련된 함수들. RDS간에 작업은 매니지먼트 콘솔이나. aws cli로 제어가 가능한 부분이고. 이 함수들의 존재 의미는, RDS가 아닌 외부의 MySQL과의 리플리케이션 연결에 있음. 리플리케이션 관련 용어인 **마스터-슬레이브**가 공식적으로 폐기되고 **소스-레플리카**로 변경될 예정이라, 실제로 아래 복제관련 함수들을 실행해보면 deprecated되었고 source/replica라는 이름 쓰는 함수로 대체되었다는 메세지가 뜨는 것을 볼 수 있음

mysql.rds_import_binlog_ssl_material

- 용도: 복제 연결에 SSL이 사용될 경우, 그 때 사용할 SSL 정보 입력
- 사용법: call mysql.rds_import_binlog_ssl_material (JSON형식의 SSL 정보 - 하단참고);

```
{'ssl_ca':"-----BEGIN CERTIFICATE----- `ssl_ca_pem_body_code` -----END CERTIFICATE-----\n',"ssl_cert':"-----BEGIN CERTIFICATE----- `ssl_cert_pem_body_code` -----END CERTIFICATE-----\n',"ssl_key':"-----BEGIN RSA PRIVATE KEY----- `ssl_key_pem_body_code` -----END RSA PRIVATE KEY-----\n'}
```

```
mysql> call mysql.rds_import_binlog_ssl_material( {'ssl_ca':"-----BEGIN CERTIFICATE-----
AAAAB3NzaC1yc2EAAAADAQABAAQCIKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V
hz2ltxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4xyyb/wB96xbiFveSFJuOp/d6RJhJOi0iBXr
lsLnBltnctkiJ7FbtXJMXLvwwJryDUiIBMTjYtwB+QhYXUMOZce5Pjz5/i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OziPnWOyN0qFU0XA246RA8QFYiCNYwl3f05p6KLxEXAMPLE -----END CERTIFICATE-----\n',"ssl_cert':"-----BEGIN CERTIFICATE-----
AAAAB3NzaC1yc2EAAAADAQABAAQCIKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V
hz2ltxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4xyyb/wB96xbiFveSFJuOp/d6RJhJOi0iBXr
lsLnBltnctkiJ7FbtXJMXLvwwJryDUiIBMTjYtwB+QhYXUMOZce5Pjz5/i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OziPnWOyN0qFU0XA246RA8QFYiCNYwl3f05p6KLxEXAMPLE -----END CERTIFICATE-----\n',"ssl_key':"-----BEGIN RSA PRIVATE
KEY----- AAAAB3NzaC1yc2EAAAADAQABAAQCIKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V
hz2ltxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4xyyb/wB96xbiFveSFJuOp/d6RJhJOi0iBXr
lsLnBltnctkiJ7FbtXJMXLvwwJryDUiIBMTjYtwB+QhYXUMOZce5Pjz5/i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OziPnWOyN0qFU0XA246RA8QFYiCNYwl3f05p6KLxEXAMPLE -----END RSA PRIVATE KEY-----\n"});
+-----+
| Message          |
+-----+
| SSL material import complete. |
+-----+
1 row in set (0.04 sec)

Query OK, 0 rows affected (0.04 sec)
```

mysql.rds_remove_binlog_ssl_material

- 용도: 앞의 명령어로 설정한 SSL관련 정보를 제거하는 명령어
- 사용법: call mysql.rds_remove_binlog_ssl_material;

```
mysql> call mysql.rds_remove_binlog_ssl_material;
Query OK, 0 rows affected (0.01 sec)
```

mysql.rds_set_external_master

- 용도: 복제 마스터 서버의 정보를 입력. change master to 명령에 해당
- 사용법:

```
mysql> call mysql.rds_set_external_master (
'master-host.example.com', -- 마스터 서버의 호스트 이름
3306,                      -- 마스터 서버의 포트
'repl_user',               -- 복제에 사용될 사용자 이름
'repl_password',           -- 복제에 사용될 사용자의 비밀번호
'mysql-bin-changelog.000123', -- 마스터 서버의 바이너리 로그 파일 이름
4567890,                  -- 바이너리 로그 파일 내의 시작 위치
0                          -- SSL 사용 여부. 1=사용, 0=미사용
);
+-----+
| Message |
+-----+
| rds_set_external_master is deprecated and will be removed in a future release. Use rds_set_external_source instead. Refer to the documentation for more information on deprecated statements |
+-----+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.05 sec)
mysql>
mysql> show slave status\G
***** 1. row *****
      Slave_IO_State:
      Master_Host: master-host.example.com
      Master_User: repl_user
      Master_Port: 3306
      Connect_Retry: 60
      Master_Log_File: mysql-bin-changelog.000123
      Read_Master_Log_Pos: 4567890
      Relay_Log_File: MySQL8023-relay-bin.000001
      Relay_Log_Pos: 4
      Relay_Master_Log_File: mysql-bin-changelog.000123
      Slave_IO_Running: No
      Slave_SQL_Running: No
      Replicate_Do_DB:
      ...
```

mysql.rds_set_external_master_with_auto_position

- 용도: 복제 마스터 서버의 정보를 입력. change master to 명령에 해당. 근데 이걸 GTID 방식
- 사용법:

```
mysql> call mysql.rds_set_external_master_with_auto_position (
'master-host.example.com', -- 마스터 서버의 호스트 이름
3306,                      -- 마스터 서버의 포트
'repl_user',               -- 복제에 사용될 사용자 이름
'repl_password',           -- 복제에 사용될 사용자의 비밀번호
0,                          -- SSL 사용 여부. 1=사용, 0=미사용
0                          -- 복제 딜레이 시간. 초단위
);
+-----+
| Message |
+-----+
| rds_set_external_master_with_auto_position is deprecated and will be removed in a future release. Use rds_set_external_source_with_auto_position instead. Refer to the documentation for more information on deprecated statements |
+-----+
```

```

1 row in set (0.03 sec)
mysql>
mysql> show slave status\G
***** 1. row *****
Slave_IO_State:
Master_Host: master-host.example.com
Master_User: repl_user
Master_Port: 3306
Connect_Retry: 60
...
Auto_Position: 1
...

```

mysql.rds_set_master_auto_position

- 용도: 오토포지션 모드를 켜고/끄고, 키면 GTID로 복제하는 것. 끄면 바이너리 로그 포지션 방식으로 복제함
- 사용법: `mysql.rds_set_master_auto_position (0 / 1);`

```

mysql> CALL mysql.rds_set_master_auto_position (1);
+-----+
| Message |
+-----+
| rds_set_master_auto_position is deprecated and will be removed in a future release. Use rds_set_source_auto_position instead. Refer to the documentation for more information on deprecated statements |
+-----+
1 row in set (0.06 sec)

Query OK, 0 rows affected (0.07 sec)
mysql>

```

mysql.rds_set_external_master_with_delay

- 용도: `rds_set_external_master`와 기본적으로 같은 용도인데, `delay` 옵션이 한줄 추가됨
- 사용법:

```

mysql> call mysql.rds_set_external_master (
'master-host.example.com', -- 마스터 서버의 호스트 이름
3306,                      -- 마스터 서버의 포트
'repl_user',               -- 복제에 사용될 사용자 이름
'repl_password',           -- 복제에 사용될 사용자의 비밀번호
'mysql-bin-changelog.000123', -- 마스터 서버의 바이너리 로그 파일 이름
4567890,                  -- 바이너리 로그 파일 내의 시작 위치
0,                        -- SSL 사용 여부. 1=사용, 0=미사용
0                          -- 복제 딜레이 시간: 초단위
);

```

mysql.rds_set_source_delay

- 용도: 다른 복제 옵션 수정 없이, 복제 딜레이 시간만 변경할 때 사용하는 함수
- 사용법: `call mysql.rds_set_source_delay(시간 초단위)`

```

mysql> call mysql.rds_set_source_delay(10);
Query OK, 0 rows affected (0.01 sec)

```

mysql.rds_reset_external_master

- 용도: 현재의 리플리케이션 구성 정보를 모두 리셋함. `reset slave` 명령에 해당.
- 사용법: `call mysql.rds_reset_external_master;`

```

mysql> call mysql.rds_reset_external_master;
Query OK, 0 rows affected (0.01 sec)

```

mysql.rds_next_master_log

- 용도: 다음 순번의 마스터 로그를 사용하여 복제를 계속하도록 함. 현 시점의 마스터로그가 다양한 에러 등으로 인해 읽을수가 없는 상황이면, 현재 파일을 아예 건너뛰고 다음 파일부터 계속 진행하기 위해 이 함수를 사용 가능. 하지만 스킵되는 부분이 많을 수 있어서 완전한 해결책이 되지는 못할 것임. 정합성이 깨진채로 운영될 수도 있기에, 일반적으로 그냥 복제 재구성을 하는게 나을 가능성이 높음
- 사용법: call mysql.rds_next_master_log(현재시점마스터로그번호);

```
mysql> SHOW SLAVE STATUS\G;
...
      Master_Log_File: mysql-bin-changelog.000123
      Read_Master_Log_Pos: 4567890
...
```

-- 현재 적용 중인 마스터로그 파일명 확인

```
mysql> call mysql.rds_next_master_log(123);
Query OK, 0 rows affected (0.02 sec)
-- 파일명을 전부 넣는 것이 아니고 숫자 부분만 넣는 것임
```

```
mysql> SHOW SLAVE STATUS\G;
...
      Master_Log_File: mysql-bin-changelog.000124
      Read_Master_Log_Pos: 156
...
```

- 다큐멘테이션 상에는 1236 에러가 발생한 경우이나 사용하라 되어있음

```
[mysql@MySQL8023 scripts]$ perror 1236
MySQL error code MY-001236 (ER_MASTER_FATAL_ERROR_READING_BINLOG): Got fatal error %d from master when reading data from binary log: '%-.512s'
```

-> 1236에러는 심각한 에러. 마스터에서 바이너리로그를 읽을 수 없음. 상태임. 디스크가 깨졌다거나, 실수로 로그파일을 지워먹었다거나 그럴때 발생하는 에러임. 어차피 파일 망가졌거나 가져올 수 없으니, 파일 통째로 스킵하는 것

mysql.rds_skip_transaction_with_gtid

- 용도: SQL 수행 관련 문제로 복제가 중단 되었을 때에, 이 함수를 통해 지정된 GTID 포지션까지 쿼리 수행을 건너뛸 수 있음
- 사용법: call mysql.rds_skip_transaction_with_gtid(목표GTID);

```
mysql> SHOW SLAVE STATUS\G;
***** 1. row *****
      Slave_IO_State: Waiting for master to send event
      Master_Host: 192.168.100.100
      Master_User: repl
      ...
      Slave_SQL_Running: No
      ...

mysql>
mysql> call mysql.rds_skip_transaction_with_gtid('30d89c39-4efc-11eb-9f39-eec956d5f988:255');
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> SHOW SLAVE STATUS\G;
***** 1. row *****
      Slave_IO_State: Waiting for master to send event
      Master_Host: 192.168.100.100
      Master_User: repl
      ...
      Slave_SQL_Running: Yes
      ...
```

mysql.rds_skip_repl_error

- 용도: SQL 수행 관련 문제로 복제가 중단 되었을 때에, 이를 해소하기 수행하지 못하고 있는 쿼리 1건을 스킵하고 넘어가게 해주는 함수
- 사용법: call mysql.rds_skip_repl_error;

```
mysql> SHOW SLAVE STATUS\G;
***** 1. row *****
```

```

Slave_IO_State: Waiting for master to send event
Master_Host: 192.168.100.100
Master_User: repl
...
Slave_SQL_Running: No
...
mysql>
mysql> call mysql.rds_skip_repl_error;
Query OK, 0 rows affected (0.02 sec)
mysql>
mysql> SHOW SLAVE STATUS\G;
***** 1. row *****
Slave_IO_State: Waiting for master to send event
Master_Host: 192.168.100.100
Master_User: repl
...
Slave_SQL_Running: Yes
...

```

-> 1건 스킵했는데도 에러가 계속 발생한다면, 문제가 없을 때까지 반복 수행 함으로써 계속해서 쿼리를 스킵해나갈 수 있음

mysql.rds_start_replication

- 용도: 리플리케이션 시작. start slave에 해당. RDS가 slave일 때 적용되는 함수
- 사용법: call mysql.rds_start_replication;

```

mysql> CALL mysql.rds_start_replication;
Query OK, 0 rows affected (0.02 sec)
mysql> SHOW SLAVE STATUS\G;
***** 1. row *****
Slave_IO_State: Waiting for master to send event
Master_Host: 192.168.100.100
Master_User: repl
...
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
...

```

mysql.rds_start_replication_until

- Documentation상에는 존재하는데, 현재 확인 가능한 Aurora V2, Aurora V3 둘 모두 이 함수가 존재하지 않는 상태임. 원래는 지정된 바이너리로그 파일까지만 복제하는 목적이었던 듯 함

mysql.rds_start_replication_until_gtid

- Documentation상에는 존재하는데, 현재 확인 가능한 Aurora V2, Aurora V3 둘 모두 이 함수가 존재하지 않는 상태임. 원래는 지정된 GTID 까지만 복제하는 목적이었던 듯 함

mysql.rds_stop_replication

- 용도: 리플리케이션 중단. stop slave에 해당. RDS가 slave일 때 적용되는 함수
- 사용법: call mysql.rds_stop_replication;

```

mysql> CALL mysql.rds_stop_replication;
Query OK, 0 rows affected (0.01 sec)
mysql> SHOW SLAVE STATUS\G;
***** 1. row *****
Slave_IO_State:
Master_Host: 192.168.100.100
Master_User: repl
...
Slave_IO_Running: No
Slave_SQL_Running: No
...

```

InnoDB 캐시 워밍

[InnoDB Buffer Pool Dump](#) 관련 작업을 수행해주는 함수들. 이 함수들은 Aurora V3에는 들어있지 않음 MySQL 8.0에서 버퍼풀 덤프 기능이 없어졌다거나 그런 것은 아니니까, 굳이 함수가 없어도 문제는 없음. 어차피 그냥 각각의 파라미터를 1로 세팅하는 것 정도밖에 하는게 없음

mysql.rds_innodb_buffer_pool_dump_now

- 용도: INNODB_BUFFER_POOL_DUMP_NOW 파라미터를 1로 설정해줌. 즉, InnoDB 버퍼풀 덤프를 실행함
- 사용법: call mysql.rds_innodb_buffer_pool_dump_now;

```
mysql> call mysql.rds_innodb_buffer_pool_dump_now;;
```

Query OK, 0 rows affected (0.03 sec)

mysql.rds_innodb_buffer_pool_load_now

- 용도: INNODB_BUFFER_POOL_LOAD_NOW 파라미터를 1로 설정해줌. 즉, InnoDB 버퍼풀 덤프 백업 기반으로 메모리 로딩을 실행함
- 사용법: call mysql.rds_innodb_buffer_pool_load_now;

```
mysql> call mysql.rds_innodb_buffer_pool_load_now;
```

Query OK, 0 rows affected (0.02 sec)

mysql.rds_innodb_buffer_pool_load_abort

- 용도: INNODB_BUFFER_POOL_LOAD_ABORT 파라미터를 1로 설정해줌. 즉, 로딩 중이던 InnoDB 버퍼풀 덤프 작업을 중지함
- 사용법: call mysql.rds_innodb_buffer_pool_load_abort;

```
mysql> call mysql.rds_innodb_buffer_pool_load_abort;
```

Query OK, 0 rows affected (0.02 sec)