-2022. 02

㈜네오클로바

목차

1 Performance_schema

- ☑ 개요
- ☑ 사용법
- ☑ 구성요소
- ☑ 활용

개요

의미

- MySQL 서버가 내부 실행 중에 발생하는 이벤트를 검사하여 모니터링
- DBMS의 곳곳에 숨어있는 여러 수집 도구용 코드들을 통해서 데이터를 수집하여 메모리에 저장, 이를 SQL을 통해서 쉽게 조회, 집계해 볼 수 있는 도구
- 서버의 수행관점에서 수집된 데이터를 제공하는 Storage Engine
- 활성화 할 경우 일부 성능저하 발생.

개요

특징

- 런타임에 서버 내부 실행을 검사하는 방법 제공
- Binary / Replication Logging 안함.
- Server Variables 통해서 제어(기본 활성화)
- SQL을 통해서 수정 가능, 소문자로 작성, 구성변경은 데이터 수집에 즉시 반영
- Memory에 저장되고 재기동시 초기화 됨. 따라서 *.frm만 존재
- 소스코드 내부에 구현 되어 있어, 별도의 스레드가 없음
- 성능스키마의 실패 여부와 상관없이 서버의 동작은 변경되지 않음
- Server Variables와 Setup Table을 통해서 제어

개요

용어

- EVENT(이벤트)
  시간이 걸리고 타이밍 정보를 수집할 수 있도록 계측된 서버가 수행하는 모든 작업
  함수호출, OS대기, 구문분석, SQL실행단계 등을 의미
  cf) Binary Log Event, mysql.event 아님.

- Instrument(계측기)
  성능스키마를 통해서 모니터링 하고자 하는 코드 유형

- Consumer(소비자?)
  성능스키마를 통해서 모니터링 결과를 저장하는 테이블 유형

- 다이제스트(digest)
  SQL문에 대하여 유사한 명령문을 그룹화, 정규화된 쿼리 형태(prepared)?

-

구성요소

주요 구성요소

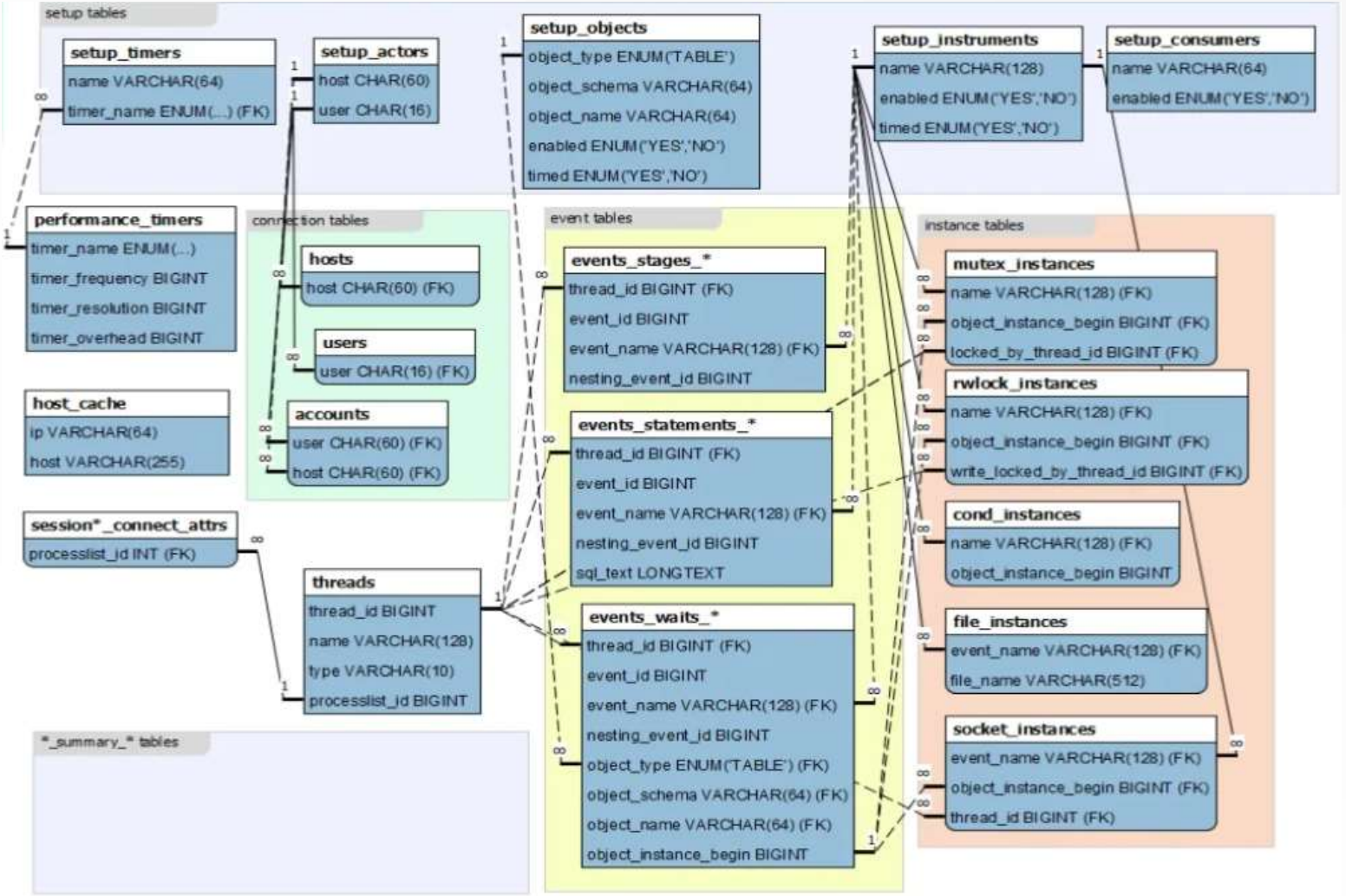
- Server Variables : 45개

- Status Variables : 31개

- Tables : 총 110개 , 6개 그룹
  - Setup Tables : 모니터링 구성을 표시하는데 사용
    setup_*
  - Events Tables : 스레드 이벤트, 스테이지 이벤트, 명령문 이벤트 정보
    events_*
  - History Tables : 가장 최근의 이벤트 기록 정보.
    events_*_history*
  - Summary Tables : 종료된 이벤트에 대한 집계 정보.
    events_*_summary*
  - Instance Tables : 측정된 오브젝트 유형정보
    *_instances
  - Etc Tables : 기타정보
    host_cache, performance_timers, threads, …

Tables - Relation

https://www.slideshare.net/ValeriyKravchuk/mysql-performance-schema-missingmanualflossuk



1. Performance schema 구성요소

Server Variables

Server Variables	기본값	Dynamic	설명
performance_schema	ON	No	성능스키마 활성화여부
performance_schema_accounts_size	-1	No	performance_schema.accounts 테이블 최대 행수. 0(저장안함), -1(자동크기조정)
performance_schema_digests_size	10000	No	events_statements_summary_by_digest 최대항수. 0(저장안함), -1(자동크기조정)
performance_schema_error_size	5018	No	계측된 서버오류 코드 수. 0(오류 계측없음)
performance_schema_events_stages_history_long_size	10000	No	events_stages_history_long 테이블 최대 행수. -1(자동크기조정)
performance_schema_events_stages_history_size	10	No	events_stages_history 테이블의 스레드당 행수. -1(자동크기조정)
performance_schema_events_statements_history_long_size	10000	No	events_statements_history_long 테이블의 최대행수. -1(자동크기조정)
performance_schema_events_statements_history_size	10	No	events_statements_history 테이블의 스레드당 행수. -1(자동크기조정)
performance_schema_events_transactions_history_long_size	10000	No	events_transactions_history_long 테이블 최대행수. -1(자동크기조정)
performance_schema_events_transactions_history_size	10	No	events_transactions_history 테이블 의 스레드당 행수. -1(자동크기조정)
performance_schema_events_waits_history_long_size	10000	No	events_waits_history_long 테이블 최대행수. -1(자동크기조정)
performance_schema_events_waits_history_size	10	No	events_waits_history 테이블 스레드당 행수. -1(자동크기조정)
performance_schema_hosts_size	-1	No	hosts 테이블 의 행 수. 0(저장안함), -1(자동크기조정)
performance_schema_max_cond_classes	150	No	계측조건 최대수
performance_schema_max_cond_instances	-1	No	계측조건 객체의 최대수. -1(자동크기조정)

Server Variables

Server Variables	기본값	Dynamic	설명
performance_schema_max_digest_length	1024	No	다이제스트 값을 계산하기 위해 명령문당 예약된 메모리의 최대 바이트 수
performance_schema_max_digest_sample_age	60	No	새 명령문을 수집하는 샘플링 최대대기시간
performance_schema_max_file_classes	80	No	파일 계측기의 최대 수.
performance_schema_max_file_handles	32768	No	열린 파일 개체의 최대 수. 항상 open_file_limit보다 커야함.
performance_schema_max_file_instances	-1	No	계측된 파일개체의 최대 수. -1(자동크기조정)
performance_schema_max_index_stat	-1	No	통계를 유지 관리하는 최대 인덱스 수. -1(자동크기조정)
performance_schema_max_memory_classes	450	No	메모리 기기의 최대 수.
performance_schema_max_metadata_locks	-1	No	메타데이터 잠금 도구의 최대 수. -1(자동크기조정)
performance_schema_max_mutex_classes	350	No	뮤텍스 기기의 최대 수
performance_schema_max_mutex_instances	-1	No	계측된 뮤텍스 개체의 최대 수. -1(자동크기조정)
performance_schema_max_prepared_statements_instances	-1	No	prepared_statements_instances 테이블의 최대행수. -1(자동크기조정)
performance_schema_max_program_instances	-1	No	통계를 유지 관리하는 최대 저장 프로그램 수. -1(자동크기조정)
performance_schema_max_rwlock_classes	60	No	rwlock 기기의 최대 수
performance_schema_max_rwlock_instances	-1	No	계측된 rwlock 개체의 최대 수
performance_schema_max_socket_classes	10	No	소켓 기기의 최대 수
performance_schema_max_socket_instances	-1		계측된 소켓 개체의 최대 수

Server Variables

Server Variables	기본값	Dynamic	설명
performance_schema_max_sql_text_length	1024	No	SQL 문을 저장하는 데 사용되는 최대 바이트 수
performance_schema_max_stage_classes	175	No	stage 계측의 최대수
performance_schema_max_statement_classes	219	No	SQL문 계측의 최대수
performance_schema_max_statement_stack	10	No	통계를 유지 관리하는 중첩된 저장 프로그램 호출의 최대 깊이
performance_schema_max_table_handles	-1	No	열린 테이블 개체의 최대 수
performance_schema_max_table_instances	-1	No	계측된 테이블 개체의 최대 수
performance_schema_max_table_lock_stat	-1	No	잠금 통계를 유지 관리하는 최대 테이블 수
performance_schema_max_thread_classes	100	No	스레드 도구의 최대 수
performance_schema_max_thread_instances	-1	No	계측된 스레드 개체의 최대 수
performance_schema_session_connect_attrs_size	512	No	연결 속성 문자열을 보유하기 위해 스레드별로 사전 할당된 메모리
performance_schema_setup_actors_size	-1	No	setup_actors테이블 의 행 수
performance_schema_setup_objects_size	-1	No	setup_objects테이블 의 행 수
performance_schema_show_processlist	OFF	Yes	performance_schema.processlist기반으로 show processlist 구현여부
performance_schema_users_size	-1	No	users 테이블 의 행 수

Status Variables

Status Variables	기본값	설명
Performance_schema_accounts_lost	0	테이블이 가득 차서 행을 테이블에 추가 할 수 없었던 횟수
Performance_schema_cond_classes_lost	0	로드할수 없었던 지표 수
Performance_schema_cond_instances_lost	0	생성할 수 없었던 조건 도구수
Performance_schema_digest_lost	0	events_statements_summary_by_digest 에 계측할수 없었던 도구 인스턴스 수
Performance_schema_file_classes_lost	0	로드할 수 없었던 파일 조건 지표수
Performance_schema_file_handles_lost	0	로드할 수 없었던 파일 핸들러 수
Performance_schema_file_instances_lost	0	로드할 수 없었던 파일 인스턴스 수
Performance_schema_hosts_lost	0	hosts테이블이 가득 차서 행을 테이블에 추가 할 수 없었던 횟수
Performance_schema_index_stat_lost	0	
Performance_schema_locker_lost	0	
Performance_schema_memory_classes_lost	0	로드할 수 없었던 메모리 도구 횟수
Performance_schema_metadata_lock_lost	0	metadata_locks테이블에 계측될 수 없는 메타 데이터 잠금의 수
Performance_schema_mutex_classes_lost	0	로드할 수 없었던 뮤텍스 객체수
Performance_schema_mutex_instances_lost	0	로드할 수 없었던 뮤텍스 인스턴스 수
Performance_schema_nested_statement_lost	0	통계가 손실 된 저장된 프로그램 문의 수
Performance_schema_prepared_statements_lost	0	prepared_statements_instances 테이블에 계측 될 수 없는 준비된 명령문의 수

Status Variables

Status Variables	기본값	설명
Performance_schema_program_lost	0	통계가 손실 된 저장된 프로그램의 수
Performance_schema_rwlock_classes_lost	0	로드할 수 없었던 rwlock 객체수
Performance_schema_rwlock_instances_lost	0	로드할 수 없었던 rwlock 인스턴스수
Performance_schema_session_connect_attrs_longest_seen	115	연결 속성 거부 전의 최대로 긴 속성 버퍼 크기
Performance_schema_session_connect_attrs_lost	0	연결 속성 절단이 발생한 연결 수
Performance_schema_socket_classes_lost	0	로드할 수 없었던 소켓 객체수
Performance_schema_socket_instances_lost	0	로드할 수 없었던 소켓 인스턴스수
Performance_schema_stage_classes_lost	0	로드할 수 없었던 스테이지 객체수
Performance_schema_statement_classes_lost	0	로드할 수 없었던 명령문 객체수
Performance_schema_table_handles_lost	0	로드할 수 없었던 테이블 핸들러수
Performance_schema_table_instances_lost	0	로드할 수 없었던 테이블 인스턴스수
Performance_schema_table_lock_stat_lost	0	잠금 통계가 유실 된 테이블의 수
Performance_schema_thread_classes_lost	0	로드할 수 없었던 스레드 객체수
Performance_schema_thread_instances_lost	0	로드할 수 없었던 스레드 인스턴스수
Performance_schema_users_lost	0	users테이블이 가득 차서 행을 테이블에 추가 할 수 없었던 횟수

In MySQL 8.0.28

Tables - Setup

설정테이블들은 성능스키마 세부설정을 위한 중요한 테이블들 입니다.

- setup_actors : 수집 대상의 연결 세션 설정

```
mysql> select * from setup_actors;
+------+------+------+---------+---------+
| HOST | USER | ROLE | ENABLED | HISTORY |
+------+------+------+---------+---------+
| %    | %    | %    | YES     | YES     |
+------+------+------+---------+---------+
```

- setup_objects : 수집 객체를 설정

```
mysql> select * from setup_objects;
+-------------+--------------------+-------------+---------+-------+
| OBJECT_TYPE | OBJECT_SCHEMA      | OBJECT_NAME | ENABLED | TIMED |
+-------------+--------------------+-------------+---------+-------+
| EVENT       | mysql              | %           | NO      | NO    |
…(20개)…
+-------------+--------------------+-------------+---------+-------+
```

- setup_instruments : 수집할 대상(계측기) 설정 (다음 장~)

- setup_timers : deprecated !!

- setup_consumers: 수집한 이벤트정보의 저장할 테이블 설정

- setup_threads : 스레드 클래스의 활성화 설정

```
mysql> select name, enabled, history from setup_threads;
+--------------------------------+---------+---------+
| name                           | enabled | history |
+--------------------------------+---------+---------+
| thread/performance_schema/setup | YES     | YES     |
…(54개)…
+--------------------------------+---------+---------+
```

Tables – setup_instruments

- 성능스키마를 통해서 모니터링 하고자 하는 계측기(코드) 유형을 나타냅니다.
- 계측기는 Tree구조로 표현됩니다.
- MySQL 8.0.28에는 1237개 계측기가 있습니다.

- idle
- stage
- statement
  statement/abstract/*
  statement/com/*
  statement/sql/*
  …
- wait
  wait/io/file/*
  wait/io/socket/*
  wait/io/table/*
  wait/lock/*
  wait/synch/*

```
select * from setup_instruments ~;
+-------------+----------+
| name        | count(*) |
+-------------+----------+
| error       |        1 |
| idle        |        1 |
| memory      |      497 |
| stage       |      131 |
| statement   |      213 |
| transaction |        1 |
| wait        |      393 |
+-------------+----------+
```

```
mysql> select * from setup_instruments where name like 'memory%' limit5;
+----------------------------------------+---------+-------+-----------------+-----------+-----------------------------------------------------------+
| NAME                                   | ENABLED | TIMED | PROPERTIES      | VOLATILITY | DOCUMENTATION                                             |
+----------------------------------------+---------+-------+-----------------+-----------+-----------------------------------------------------------+
| memory/performance_schema/mutex_instances  | YES     | NULL  | global_statistics |          1 | Memory used for table performance_schema.mutex_instances  |
| memory/performance_schema/rwlock_instances | YES     | NULL  | global_statistics |          1 | Memory used for table performance_schema.rwlock_instances |
| memory/performance_schema/cond_instances   | YES     | NULL  | global_statistics |          1 | Memory used for table performance_schema.cond_instances   |
| memory/performance_schema/file_instances   | YES     | NULL  | global_statistics |          1 | Memory used for table performance_schema.file_instances   |
| memory/performance_schema/socket_instances | YES     | NULL  | global_statistics |          1 | Memory used for table performance_schema.socket_instances |
+----------------------------------------+---------+-------+-----------------+-----------+-----------------------------------------------------------+
```

수집환경 설정

1. performance_schema 기능 활성화

```
[mysqld]
performance_schema=ON
```

```
mysql> show global variables like 'performance_schema';
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| performance_schema | ON    |
+--------------------+-------+
```

수집환경 설정

2. 모니터링할 사용자 세션에 대하여 설정한다.

setup_actors - 기본값 사용.

```
mysql> select * from setup_actors;
+------+------+------+---------+---------+
| HOST | USER | ROLE | ENABLED | HISTORY |
+------+------+------+---------+---------+
| %    | %    | %    | YES     | YES     |
+------+------+------+---------+---------+
```

(모든 클라이언트의 연결계정에 대해서 모니터링)

수집환경 설정

3. 모니터링할 객체가 EVENT, FUNCTION, PROCEDURE, TABLE, TRIGGER 인 경우 선택한다.

setup_objects - 기본값 사용.

```
# 현재 기본으로 설정된 값 확인.
mysql> select * from setup_objects where enabled='yes';
+-------------+---------------+-------------+---------+-------+
| OBJECT_TYPE | OBJECT_SCHEMA | OBJECT_NAME | ENABLED | TIMED |
+-------------+---------------+-------------+---------+-------+
| EVENT       | %             | %           | YES     | YES   |
| FUNCTION    | %             | %           | YES     | YES   |
| PROCEDURE   | %             | %           | YES     | YES   |
| TABLE       | %             | %           | YES     | YES   |
| TRIGGER     | %             | %           | YES     | YES   |
+-------------+---------------+-------------+---------+-------+
(모든 event, function, procedure, table, trigger 모니터링)
```

수집환경 설정

4. 수집할 이벤트를 계측할 개체를 선택합니다.

setup_instruments - wait, statement, stage, transactions 등 원하는 개체 설정.(모두 수집)

```
# 기본으로 설정되어 있는 개체 확인
mysql> select * from setup_instruments where enabled='yes' and timed='yes';
...
285 rows in set (0.00 sec)

# 모든 대상을 수집.
mysql> update setup_instruments set enabled='yes', timed='yes';

# 일부 객체 활성화
mysql> update setup_instruments set enabled='YES', timed='YES' where name like '%metadata%';

# 특정 계측 개체를 비활성화
mysql> update setup_instruments set enabled= ‘no’ where name like  ‘wait/io/file/%’ ;
```

수집환경 설정

5. 수집데이터를 보내고 저장할 대상 테이블의 사용여부를 설정한다.

setup_consumers - 각자 목적에 맞게 설정 (모두 활성화)
consumer항목은 계층구조로 하위는 상위의 활성화 여부에 따라 작동한다.
하위측정을 활성화 하려면 상위가 모두 활성화해야 합니다.

- global_instrumentation
  - thread_instrumentation
    - events_stages_current
      - events_stages_history
      - events_stages_history_long
    - events_statements_cpu
    - events_statements_current
      - events_statements_history
      - events_statements_history_long
    - events_transactions_current
      - events_transactions_history
      - events_transactions_history_long
    - events_waits_current
      - events_waits_history
      - events_waits_history_long
  - statements_digest

```
mysql> select * from setup_consumers;
+----------------------------------+---------+
| NAME                             | ENABLED |
+----------------------------------+---------+
| events_stages_current            | NO      |
| events_stages_history            | NO      |
| events_stages_history_long       | NO      |
| events_statements_cpu            | NO      |
| events_statements_current        | YES     |
| events_statements_history        | YES     |
| events_statements_history_long   | NO      |
| events_transactions_current      | YES     |
| events_transactions_history      | YES     |
| events_transactions_history_long | NO      |
| events_waits_current             | NO      |
| events_waits_history             | NO      |
| events_waits_history_long        | NO      |
| global_instrumentation           | YES     |
| thread_instrumentation           | YES     |
| statements_digest                | YES     |
+----------------------------------+---------+

#모든 카테고리 모두 활성화
mysql> update setup_consumers set enabled='yes';
```

수집환경 설정

6. 쓰레드 관련 클래스의 활성화 여부 설정합니다.

setup_threads - 기본 설정(ALL Enabled).

```
# 기본으로 설정되어 있는 개체 확인(모두 enabled)
mysql> select * from setup_threads;

# 모든 쓰레드 관련 클래스의 활성화 설정.
mysql> update setup_threads set enabled='yes', history='yes';

# 특정 계측 개체를 비활성화
mysql> update setup_threads set enabled= 'no' , history= 'no' where name like  'wait/io/file/%' ;
```

수집환경 설정

7. 기존 계측된 데이터들 초기화한다.

#이전에 기록된 요약정보 테이블 데이터를 TRUNCATE한다.

```
SELECT table_name
FROM INFORMATION_SCHEMA.TABLES
WHERE table_schema = 'performance_schema'
AND (table_name LIKE '%summary%' OR table_name LIKE '%history%');
```

결과 테이블들을 모두 TRUNCATE해야 합니다.
번거로움….

그래서 sys schema 를 이용.

```
mysql> CALL sys.ps_truncate_all_tables(FALSE);
+---------------------+
| summary             |
+---------------------+
| Truncated 49 tables |
+---------------------+
```

2 Sys schema

- ☑ 개요
- ☑ 사용법
- ☑ 구성요소
- ☑ 활용

개요

의미

- Performance_schema만 가지곤 데이터 분석에 힘든 부분이 있어요
- Performance_schema를 편하게 볼 수 있도록 만든 Views

개요

특징

- performance_schema 와 information_schema를 이용

- formatted view와 raw view로 구성
  formatted view는 사용자가 식별이 편하도록 구성.
  raw view는 x$~가 테이블명 앞에 붙음

- formatted view는 ms, us로 보이나, raw view는 pico second(1/1조)로 보임

```
mysql> select * from sys.schema_object_overview where db='sys';
+-----+---------------+-------+
| db  | object_type   | count |
+-----+---------------+-------+
| sys | BASE TABLE    |     1 |
| sys | FUNCTION      |    22 |
| sys | INDEX (BTREE) |     1 |
| sys | PROCEDURE     |    26 |
| sys | TRIGGER       |     2 |
| sys | VIEW          |   100 |
+-----+---------------+-------+
```

구성요소

주요 구성요소

- Table(1) : sys_config

- sys Views
  - User/Host Summary views (12/12) : user/host 정보
  - IO Summary views (10) : io정보
  - Schema Analysis views (15) : schema 통계정보
  - Wait Analysis views (10) : wait 요약
  - Statement Analysis views (12) : SQL 분석결과
  - Etc : processlist, version

- Functions (22)

- Procedures (26) : 간편한 설정지원 (ps_setup_%), traceing

Table

- sys_config

```
CREATE TABLE `sys_config` (
  `variable` varchar(128) NOT NULL,
  `value` varchar(128) DEFAULT NULL,
  `set_time` timestamp NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  `set_by` varchar(128) DEFAULT NULL,
  PRIMARY KEY (`variable`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

구성요소

Views

- formatted view / raw views(x$~)
  formatted view만 봐도 되겠어요~

  - user_summary_%
  - host_summary_%
  - io_%, latest_file_io
  - schema_%
  - wait_%
  - waits_%
  - statement_%
  - statements_%
  - memory_%
  - processlist, version, session, metrics
  - innodb_buffer_stats_by_schema, innodb_buffer_stats_by_table, innodb_lock_waits

구성요소

Functions

```
mysql> select routine_name from information_schema.routines where ROUTINE_TYPE='FUNCTION';
+----------------------------------+
| ROUTINE_NAME                     |
+----------------------------------+
| extract_schema_from_file_name    | 파일이름을 schema이름을 반환
| extract_table_from_file_name     | 파일이름을 테이블이름을 반환
| format_bytes                     | 바이트정보 가독성 지원. deprecated
| format_path                      | 경로명을 시스템변수 문자열 변환
| format_statement                 | SQL문자열 줄임처리.
| format_time                      | 시간정보 가독성 지원. deprecated
| list_add                         | 쉼표 구분목록에 문자열 추가함수
| list_drop                        | 쉼표 구분목록에 문자열 삭제함수
| ps_is_account_enabled            | 사용자 활성화 확인
| ps_is_consumer_enabled           | consumer 활성화 확인
| ps_is_instrument_default_enabled | 계측기 기본활성화 확인
| ps_is_instrument_default_timed   | 계측기 기본시간 설정확인
| ps_is_thread_instrumented        | 스레드 계측기 활성화 확인
| ps_thread_id                     | connection_id에 대한 계정정보 반환
| ps_thread_account                | connection_id애 대한 스레드 id 반환. deprecated
| ps_thread_stack                  | 스레드id에 대한 JSON 스택정보 반환
| ps_thread_trx_info               | 스레드id에 대한 정보를 JSON으로 반환
| quote_identifier                 | 문자열을 백틱(`)문자가 포함된 문자열로 변환
| sys_get_config                   | sys_config 값을 추출
| version_major                    | 주요버전 반환
| version_minor                    | 마이너버전 반환
| version_patch                    | 릴리즈버전 반환
+----------------------------------+
```

구성요소

Procedures

sys.create_synonym_db	schem의 동의어 생성
sys.execute_prepared_stmt	prepared SQL 실행
sys.diagnostics	백그라운드 스레드 계측 비활성화
sys.ps_statement_avg_latency_histogram	SQL에 대한 히스토그램 표시
sys.ps_trace_statement_digest	SQL 상세정보
sys.ps_trace_thread	스레드 트레이싱
sys.ps_setup_disable_background_threads	백그라운드 스레드 계측 비활성화
sys.ps_setup_disable_consumer	선택 consumer 비활성화
sys.ps_setup_disable_instrument	선택 계측기 비활성화
sys.ps_setup_disable_thread	선택 스레드 비활성화
sys.ps_setup_enable_background_threads	백그라운드 스레드 계측 활성화
sys.ps_setup_enable_consumer	선택 consumer 활성화
sys.ps_setup_enable_instrument	**선택 계측기 활성화**
sys.ps_setup_enable_thread	**선택 스레드 활성화**
sys.ps_setup_reload_saved	동일세션의 구성정보 리로딩
sys.ps_setup_reset_to_default	**기본설정으로 재설정**
sys.ps_setup_save	구성저장
sys.ps_setup_show_disabled	비활성화된 구성정보 표시
sys.ps_setup_show_disabled_consumers	비활성화된 consumer 표시
sys.ps_setup_show_disabled_instruments	비활성화된 계측기 표시
sys.ps_setup_show_enabled	활성화된 구성정보 표시
sys.ps_setup_show_enabled_consumers	활성화된 consumer 표시
sys.ps_setup_show_enabled_instruments	활성화된 계측기 표시
sys.ps_truncate_all_tables	**집계정보테이블 초기화**
sys.statement_performance_analyzer	명령문에 대한 보고서작성
sys.table_exists	테이블/뷰 존재확인

환경 설정

sys schema의 간편한 활용.

```
#초기설정으로 되돌리기
mysql> Call sys.ps_setup_reset_to_default(TRUE);


#일부 계측기 활성화
Call sys.ps_setup_enable_instrument('wait');
Call sys.ps_setup_enable_instrument('stage');
Call sys.ps_setup_enable_instrument('statement');

#일부 consumer활성화
Call sys.ps_setup_enable_consumer('current');
Call sys.ps_setup_enable_consumer('history_long');
```

모니터링

SQL 내부 이벤트에 대한 집계 정보

```
mysql> select * from sys.user_summary_by_statement_type;
```

user	statement	total	total_latency	max_latency	lock_latency	cpu_latency	rows_sent	rows_examined	rows_affected	full_scans
root	select	26	200.00 ms	68.01 ms	134.00 us	33.77 ms	1457	29265	0	24
root	call_procedure	1	65.54 ms	65.54 ms	35.00 us	65.54 ms	0	0	0	0
root	Field List	110	51.77 ms	3.21 ms	2.52 ms	51.62 ms	0	0	0	0
root	stmt	115	32.61 ms	3.24 ms	50.18 ms	32.47 ms	1	162	0	0
root	show_tables	1	2.11 ms	2.11 ms	5.00 us	2.11 ms	110	332	0	0
root	show_databases	1	851.22 us	851.22 us	6.00 us	849.55 us	5	21	0	1
root	jump_if_not	57	229.50 us	6.21 us	0 ps	171.59 us	0	0	0	0
root	set	30	213.49 us	37.21 us	0 ps	180.60 us	0	0	0	0
root	cfetch	29	136.88 us	34.79 us	0 ps	106.70 us	0	0	0	0
root	Init DB	1	52.32 us	52.32 us	0 ps	50.90 us	0	0	0	0
root	Quit	1	51.74 us	51.74 us	0 ps	47.88 us	0	0	0	0
root	jump	30	49.83 us	2.06 us	0 ps	20.02 us	0	0	0	0
root	error	1	47.27 us	47.27 us	0 ps	45.85 us	0	0	0	0
root	hreturn	1	2.71 us	2.71 us	0 ps	1.70 us	0	0	0	0
root	hpop	1	1.87 us	1.87 us	0 ps	882.00 ns	0	0	0	0
root	cclose	1	1.83 us	1.83 us	0 ps	872.00 ns	0	0	0	0
root	cpop	1	1.69 us	1.69 us	0 ps	732.00 ns	0	0	0	0
sysbench	commit	102827	56.38 min	144.66 ms	0 ps	17.87 s	0	0	0	0
sysbench	select	1439578	13.98 min	374.91 ms	7.07 s	4.62 min	31979197	62724470	0	0
sysbench	update	205654	1.78 min	314.69 ms	6.43 s	39.58 s	0	205654	205654	0
sysbench	delete	102827	45.01 s	255.12 ms	3.26 s	16.17 s	0	102827	102827	0
sysbench	insert	102827	37.12 s	225.44 ms	534.42 ms	15.63 s	0	0	102827	0
sysbench	begin	102827	4.03 s	27.06 ms	0 ps	3.42 s	0	0	0	0
sysbench	Quit	100	1.42 ms	47.59 us	0 ps	1.22 ms	0	0	0	0

모니터링

현재 실행중인 쿼리

```
MySQL> show full processlist;

MySQL> select * from information_schema.processlist;

MySQL> select * from performance_schema.events_statements_current; #세션들의 마지막 SQL확인가능.

MySQL> select processlist_id id, processlist_user user, processlist_host host, processlist_db db,processlist_command cmd,
processlist_time time, processlist_state state,left(processlist_info, 20) as qry
from performance_schema.threads s
where processlist_id is not null
and processlist_user is not null
and processlist_command not in ('Sleep','Binlog Dump')
order by processlist_time desc;

+----+-----------------+-----------+---------+---------+--------+-----------------------+----------------------+
| Id | User            | Host      | db      | Command | Time   | State                 | Info                 |
+----+-----------------+-----------+---------+---------+--------+-----------------------+----------------------+
|  5 | event_scheduler | localhost | NULL    | Daemon  | 589335 | Waiting on empty queue | NULL                 |
| 37 | root            | localhost | sysbench | Query  |      0 | init                  | show full processlist |
+----+-----------------+-----------+---------+---------+--------+-----------------------+----------------------+

MySQL> select * from sys.processlist;
```

모니터링

수행횟수가 가장 많은 상위SQL 10개

```
MySQL> select digest_text, count_star,(avg_timer_wait/1000000000000)
exec_t_s,sum_cpu_time/count_star,sum_rows_sent/count_star,sum_rows_examined/count_star
from performance_schema.events_statements_summary_by_digest
order by count_star desc
limit 10;
```

수행횟수가 가장 많고 2초 이상 소요된 상위SQL 10개.

```
MySQL> select digest_text,  count_star, (avg_timer_wait/1000000000000) exec_t_s, sum_cpu_time/count_star,
sum_rows_sent/count_star, sum_rows_examined/count_star
from performance_schema.events_statements_summary_by_digest
where (avg_timer_wait/1000000000000) >= 2
order by count_star desc
limit 10;
```

```
MySQL> select query, exec_count, avg_latency, cpu_latency, rows_sent_avg, rows_examined_avg
from sys.statement_analysis
order by exec_count desc
limit 10;
```

```
cf) # mysqldumpslow -t 10 -s c mysql-slow.log
```

모니터링

평균수행시간이 가장 긴 상위SQL 10개

```
MySQL>  select digest_text,  count_star, (avg_timer_wait/1000000000000) exec_t_s, sum_cpu_time/count_star,
sum_rows_sent/count_star, sum_rows_examined/count_star
from performance_schema.events_statements_summary_by_digest
order by avg_timer_wait desc
limit 10;
```

```
MySQL> select query, exec_count, avg_latency, cpu_latency, rows_sent_avg, rows_examined_avg
from sys.statement_analysis
order by avg_latency desc
limit 10;
```

```
cf) # mysqldumpslow -t 10 -s at mysql-slow.log
```

모니터링

평균잠금시간이 가장 긴 상위SQL 10개

```
MySQL>   select digest_text,  count_star, (avg_timer_wait/1000000000000) exec_t_s, sum_cpu_time/count_star,
sum_rows_sent/count_star, sum_rows_examined/count_star, (SUM_LOCK_TIME/count_star)/1000000000000 as avg_lock_time
from performance_schema.events_statements_summary_by_digest
order by (SUM_LOCK_TIME/count_star) desc
limit 10;
```

평균잠금시간이 가장 긴 SQL중 2초 이상 상위SQL 10개

```
MySQL>   select digest_text,  count_star, (avg_timer_wait/1000000000000) exec_t_s, sum_cpu_time/count_star,
sum_rows_sent/count_star, sum_rows_examined/count_star, (SUM_LOCK_TIME/count_star)/1000000000000 as avg_lock_time
from performance_schema.events_statements_summary_by_digest
where (avg_timer_wait/1000000000000) >= 2
order by (SUM_LOCK_TIME/count_star) desc
limit 10;
```

```
MySQL> select query, exec_count, avg_latency, cpu_latency, rows_sent_avg, rows_examined_avg
from sys.statement_analysis
order by (lock_latency/ exec_count) desc
limit 10;
```

```
cf) # mysqldumpslow -t 10 -s al mysql-slow.log
```

모니터링

평균반환행수가 가장 많은 상위SQL 10개

```
MySQL> select digest_text,  count_star, (avg_timer_wait/1000000000000) exec_t_s, sum_cpu_time/count_star as avg_cpu_time,
sum_rows_sent/count_star as avg_sent_rows, sum_rows_examined/count_star as avg_examined_rows,
(SUM_LOCK_TIME/count_star)/1000000000000 as avg_lock_time
from performance_schema.events_statements_summary_by_digest
order by (SUM_ROWS_SENT/count_star) desc
limit 10;
```

평균반환행수가 가장 많은 SQL중 2초 이상 상위SQL 10개

```
MySQL>  select digest_text,  count_star, (avg_timer_wait/1000000000000) exec_t_s, sum_cpu_time/count_star as
avg_cpu_time, sum_rows_sent/count_star as avg_sent_rows, sum_rows_examined/count_star as avg_examined_rows,
(SUM_LOCK_TIME/count_star)/1000000000000 as avg_lock_time
from performance_schema.events_statements_summary_by_digest
where (avg_timer_wait/1000000000000) >= 2
order by (SUM_ROWS_SENT /count_star) desc
limit 10;
```

```
MySQL> select query, exec_count, avg_latency, cpu_latency, rows_sent_avg, rows_examined_avg
from sys.statement_analysis
order by rows_sent_avg desc
limit 10;
```

```
cf) # mysqldumpslow -t 10 -s ar mysql-slow.log
```

모니터링

비효율 SQL 쿼리확인

```
MySQL> select schema_name, left(digest_text, 40) qry, count_star, sum_cpu_time,
sum_created_tmp_disk_tables sctdt, sum_select_full_join ssfj, sum_select_range_check ssrc, sum_sort_merge_passes ssmp
from performance_schema.events_statements_summary_by_digest
where sum_created_tmp_disk_tables > 0
OR sum_select_full_join > 0
OR sum_select_range_check > 0
OR sum_sort_merge_passes > 0
ORDER BY sum_sort_merge_passes DESC
LIMIT 10;

MySQL> select * from sys.schema_tables_with_full_table_scans;

MySQL> select query, exec_count, avg_latency, cpu_latency, rows_sent_avg, rows_examined_avg
from sys.statement_analysis
where db not in ('mysql','sys','performance_schema','information_schema')
and ( tmp_disk_tables >0 or sort_merge_passes >0 or full_scan='*' )
order by sort_merge_passes desc
limit 10;
```

모니터링

특정 SQL의 프로파일링

```
MySQL> SELECT s.event_id AS Query_ID, TRUNCATE(s.timer_wait/1000000000000, 6) as Duration,LEFT(s.sql_text, 120) AS Query
FROM performance_schema.events_statements_history_long s
INNER JOIN performance_schema.threads t on t.thread_id=s.thread_id
where 1=1
#and t.processlist_id = CONNECTION_ID() #현재 My session
order by Duration desc
limit 10;

+----------+----------+-----------------------------------------+
| Query_ID | Duration | Query                                   |
+----------+----------+-----------------------------------------+
|      874 |   0.0064 | select * from sbtest1 limit 10000       |
|     1403 |   0.0058 | select * from events_stages_history_long |
|      443 |   0.0058 | desc events_statements_history_long     |
|     1158 |   0.0037 | desc events_stages_history_long         |
|     1403 |   0.0013 | select * from sbtest2 limit 1500        |
|     1524 |   0.0013 | select * from sbtest2 limit 2000        |

MySQL> select event_name, truncate(timer_wait/1000000000000,6) as duration
from performance_schema.events_stages_history_long
where nesting_event_id=874
order by timer_start asc;
```

모니터링

핫 블럭 확인… 미사용 테이블 확인(after long time)

```
MySQL> select  object_schema, object_name,
count_star, count_read, count_write,
count_insert as inserts, count_update as updates, count_delete as deletes, count_fetch as fetchs
from performance_schema.table_io_waits_summary_by_table
where count_star > 0
order by count_star desc;
+---------------+-------------+------------+------------+-------------+---------+---------+---------+--------+
| object_schema | object_name | count_star | count_read | count_write | inserts | updates | deletes | fetchs |
+---------------+-------------+------------+------------+-------------+---------+---------+---------+--------+
| sysbench      | sbtest2     |     503734 |     498949 |        4785 |    1207 |    2371 |    1207 | 498949 |
| sysbench      | sbtest1     |     497531 |     492856 |        4675 |    1158 |    2359 |    1158 | 492856 |
+---------------+-------------+------------+------------+-------------+---------+---------+---------+--------+
```

모니터링

풀스캔이 많은 테이블 확인

```
MySQL> select * from (
select
digest_text,
if(sum_no_good_index_used>0 or sum_no_index_used>0, 'FULLSCAN','') as fullscan,
sum_timer_wait
from performance_schema.events_statements_summary_by_digest
where schema_name not in ('mysql','sys','performance_schema','information_schema')
) a
where fullscan<>''
order by sum_timer_wait desc
limit 10;

mysql> select * from sys.schema_tables_with_full_table_scans;
+---------------+-------------+-------------------+---------+
| object_schema | object_name | rows_full_scanned | latency |
+---------------+-------------+-------------------+---------+
| sysbench      | sbtest1     |              1000 | 25.72 s |
+---------------+-------------+-------------------+---------+
```

모니터링

미사용 인덱스 확인

```
MySQL>
SELECT DISTINCT s.table_schema, s.table_name, s.index_name, i.count_star
FROM information_schema.statistics AS s
LEFT JOIN performance_schema.table_io_waits_summary_by_index_usage AS i
ON (s.table_schema = i.object_schema AND s.table_name = i.object_name AND s.index_name = i.index_Name)
WHERE s.table_schema NOT IN ('mysql', 'performance_schema', 'sys', 'information_schema')
AND s.index_name != 'PRIMARY'
AND i.count_star = 0
ORDER BY s.table_schema, s.table_name, s.index_name;
+--------------+------------+------------+------------+
| TABLE_SCHEMA | TABLE_NAME | INDEX_NAME | count_star |
+--------------+------------+------------+------------+
| sysbench     | sbtest1    | k          |          0 |
| sysbench     | sbtest1    | k_2        |          0 |
+--------------+------------+------------+------------+

MySQL> select * from sys.schema_unused_indexes where object_schema not in ('performance_schema') ;
+---------------+-------------+------------+
| object_schema | object_name | index_name |
+---------------+-------------+------------+
| sysbench      | sbtest1     | k          |
| sysbench      | sbtest1     | k_2        |
+---------------+-------------+------------+
```

모니터링

중복 인덱스 확인

```
mysql> select * from sys.schema_redundant_indexes;
*************************** 1. row ***************************
              table_schema: sysbench
                table_name: sbtest1
        redundant_index_name: k
     redundant_index_columns: k
redundant_index_non_unique: 1
         dominant_index_name: k_2
      dominant_index_columns: k,vk
 dominant_index_non_unique: 1
            subpart_exists: 0
            sql_drop_index: ALTER TABLE `sysbench`.`sbtest1` DROP INDEX `k`
```

모니터링

잠금대기(Lock Wait) 확인

```
mysql> select * from information_schema.innodb_locks;
ERROR 1109 (42S02): Unknown table 'INNODB_LOCKS' in information_schema

mysql> select thread_id, event_id, object_schema,object_name, index_name, lock_type, lock_mode, lock_status
from performance_schema.data_locks;
+-----------+----------+---------------+-------------+------------+-----------+---------------+-------------+
| thread_id | event_id | object_schema | object_name | index_name | lock_type | lock_mode     | lock_status |
+-----------+----------+---------------+-------------+------------+-----------+---------------+-------------+
|       127 |    51844 | sysbench      | sbtest1     | NULL       | TABLE     | IX            | GRANTED     |
|       127 |    51853 | sysbench      | sbtest1     | PRIMARY    | RECORD    | X,REC_NOT_GAP | GRANTED     |
+-----------+----------+---------------+-------------+------------+-----------+---------------+-------------+
cf) 5.7 처럼 해당 세션으로 인한 타 세션의 대기가 없으면 안보이는게 아니라 해당 세션에 대한 Lock 정보확인 가능.

mysql> select a.thread_id, a.event_id, a.object_name, a.index_name, a.lock_type, a.lock_mode, b.sql_text
from performance_schema.data_locks a
inner join performance_schema.events_statements_current b on b.thread_id= a.thread_id
order by thread_id, event_id;
+-----------+----------+-------------+------------+-----------+---------------+-----------------------------------+
| thread_id | event_id | object_name | index_name | lock_type | lock_mode     | sql_text                          |
+-----------+----------+-------------+------------+-----------+---------------+-----------------------------------+
|       127 |    52287 | sbtest1     | NULL       | TABLE     | IX            | update sbtest1 set k=k+1 where id=1 |
|       127 |    52296 | sbtest1     | PRIMARY    | RECORD    | X,REC_NOT_GAP | update sbtest1 set k=k+1 where id=1 |
+-----------+----------+-------------+------------+-----------+---------------+-----------------------------------+
```

모니터링

세션간 잠금대기(Lock Wait) 경합 확인

```
mysql> select * from information_schema.innodb_lock_waits;
ERROR 1109 (42S02): Unknown table 'INNODB_LOCK_WAITS' in information_schema

mysql> select * from performance_schema.data_lock_waits₩G;

mysql> select a.BLOCKING_THREAD_ID, c.sql_text,'---is blocking ---', a.REQUESTING_THREAD_ID, b.sql_text
from performance_schema.data_lock_waits a
inner join events_statements_current b on b.thread_id= a.REQUESTING_THREAD_ID
inner join events_statements_current c on c.thread_id= a.BLOCKING_THREAD_ID;
+-------------------+-----------------------------------+-------------------+---------------------+-----------------------------------+
| BLOCKING_THREAD_ID | sql_text                          | ---is blocking --- | REQUESTING_THREAD_ID | sql_text                          |
+-------------------+-----------------------------------+-------------------+---------------------+-----------------------------------+
|               127 | update sbtest1 set k=k+1 where id=1 | ---is blocking --- |                 129 | update sbtest1 set k=k+1 where id<10 |
+-------------------+-----------------------------------+-------------------+---------------------+-----------------------------------+


mysql> select * from sys.innodb_lock_waits;
```

모니터링

메타데이터 잠금 확인

[mysqld]
performance-schema-instrument='wait/lock/metadata/sql/mdl=ON'

```
MySQL> UPDATE performance_schema.setup_instruments SET ENABLED = 'YES', TIMED = 'YES'
WHERE NAME = 'wait/lock/metadata/sql/mdl';
```

```
mysql> select * from performance_schema.metadata_locks;

mysql> select b.thread_id, #a.owner_event_id, a.object_schema, a.object_name,
a.lock_type, a.lock_status,
b.processlist_command, b.processlist_time, b.processlist_state, left(b.processlist_info,40) as running_sql,
c.sql_text as last_sql
from performance_schema.metadata_locks a
inner join performance_schema.threads b on b.thread_id = a.owner_thread_id
inner join performance_schema.events_statements_current c on c.thread_id = b.thread_id
where a.object_schema not in ('sys','mysql','information_schema','performance_schema');
```

```
+-----------+-------------+-------------+--------------------+-----------------+------------------+-------------+-----------------------------------+
| thread_id | lock_type   | lock_status | processlist_command | processlist_time | processlist_state | running_sql | last_sql                          |
+-----------+-------------+-------------+--------------------+-----------------+------------------+-------------+-----------------------------------+
|        51 | SHARED_WRITE | GRANTED     | Sleep              |             204 | NULL             | NULL        | update sbtest1 set k=k+1 where id=1 |
+-----------+-------------+-------------+--------------------+-----------------+------------------+-------------+-----------------------------------+
```

정리

설정 및 운용

- 필요한 instruments만 활성화
  Default + metadata_locks 정도만 enable

```
[mysqld]
performance-schema= ON
performance-schema-instrument='wait/lock/metadata/sql/mdl=ON'
```

- sys schem의 프로시저를 활용

```
CALL sys.ps_truncate_all_tables(FALSE);
Call sys.ps_setup_reset_to_default(TRUE);

Call sys.ps_setup_enable_instrument('wait');
Call sys.ps_setup_enable_consumer('current');
Call sys.ps_setup_enable_consumer('history_long');
```

-

NeoClova

㈜네오클로바의 지향은 '가치창조' 입니다.

좋은 제품과 최고의 서비스로 고객만족을 통해
회사의 가치가 창조된다고 생각합니다.

항상 고객만족을 위해 최선을 다할 것을 약속
드립니다.

www.neoclova.co.kr (02) 539-4880 sales@neoclova.co.kr