IRIS SQL Reference

Mobigen, November 20, 2015

목차

1 지원 정보

1.1 문서 버전

0.1

1.2 IRIS 버전

1.5.1

1.3 사용 문의 및 기술 정보

(주) 모비젠

본사

- 135-280 서울 강남구 대치동 967-3번지 KM빌딩 2층, (주) 모비젠
- T : 02 - 538 - 9360
- F : 02 - 538 - 9369

기술 연구소

- 135-280 서울 강남구 대치동 967-3번지 KM빌딩 5층, (주) 모비젠
- T : 02 - 538 - 9364
- F : 02 - 538 - 9368

모비젠 IRIS 기술 지원

- T : 02 - 538 - 9364
- M : iris@mobigen.com

2 SQL Reference

IRIS 에서 사용가능한 CLI 명령어 및 SQL 문법을 설명합니다.

2.1 CLI 명령어

IRIS 에서는 시스템 정보 조회, 또는 IRIS 에서 지원하는 기능을 사용할 수 있는 CLI 명령어를 제공합니다. API 또는 IPLUS 를 사용해서 쿼리와 동일하게 CLI 명령어를 실행, 결과를 리턴받습니다. IRIS 가 설치되지 않은 원격 클라이언트에서 사용 가능하며, 일부 기능은 계정권한에 영향을 받습니다.

2.1.1 IPLUS CMD 명령어

IRIS 에서는 IRIS 에 편하게 접속이 가능한 IPLUS CLI 를 제공합니다.

접속방법 : iplus 계정명[@호스트[:포트번호]]

계정명만 사용하는 경우, 해당 노드에 설치된 IRIS 에 접속합니다.

```
[iris@ ~]$ iplus test
Password:
Connecting to IRIS(test@192.168.131.13:5050).
Connected to IRIS.
Enter ".help" for instructions
iplus>
```

마스터노드를 호스트로 설정한 경우 포트번호 입력없이 접속가능합니다.

포트번호는 5050 입니다. (기본값이므로 입력안해도 무방합니다.)

```
$ iplus test@192.168.131.13
Password:
```

```
Connecting to IRIS(test@192.168.131.13:5050).
Connected to IRIS.
Enter ".help" for instructions
iplus>
```

특정 슬레이브노드를 호스트로 설정한 경우 포트번호는 5100 입니다.

```
$ iplus test@192.168.131.131:5100
Password:
Connecting to IRIS(test@192.168.131.131:5100).
Connected to IRIS.
Enter ".help" for instructions
iplus> .q
```

시퀀스파일 실행방법 : IPlus < {시퀀스파일경로}

STDIN 으로 시퀀스파일경로를 넣으면 해당 파일속의 명령어/쿼리를 순서대로 실행합니다.

```
[iris@localhost ~]$ cat seq_test
--#test/test#--
.recordSep \n
.fieldSep \t
SELECT * FROM LOCAL_TEST_TABLE;
[iris@localhost ~]$

[iris@localhost ~]$ iplus < seq_test
Connecting to IRIS(test@192.168.131.13:5050).
Connected to IRIS.
Enter ".help" for instructions
SQL syntax error.
After the error occured, session recreated.
SQL syntax error.
After the error occured, session recreated.
Ret : +OK Success

 K            P            A
```

```
=====================================================
 k6             20150616000000 None
 k7             20150616000000
 k4             20150616000000 0
 k5             20150616000000 0.1
 k3             20150616000000 1.2
 k2             20150616000000 1
=====================================================
6 row in set
0.0982 sec

Goodbye.
[iris@localhost ~]$
```

2.1.2 IPLUS CLI 명령어

IPLUS CLI 에서 사용가능한 명령어입니다.

명령어	설명
.history	명령/쿼리 실행 기록을 출력합니다. IRIS 에 저장되지 않고 서버 계정별로 로컬에 저장됩니다.
.sep {Field구분자} {Record구분자}	현재 IPLUS 접속에서 사용할 구분자를 설정합니다.
.field_sep {Field구분자}	현재 IPLUS 접속에서 사용할 Field 구분자만 설정합니다.
.record_sep {Record구분자}	현재 IPLUS 접속에서 사용할 Record 구분자만 설정합니다.
.width {길이} {길이} …	각 컬럼별 길이를 순서대로 설정합니다. 설정되지 않은 컬럼은 13 으로 설정됩니다.
.load {테이블명} {키} {파티션} {컨트롤파일경로} {데이터파일경로}	로컬에 있는 파일을 IRIS 로 로딩합니다. 파일경로는 절대경로를 사용하세요.
.spool {파일경로}	결과데이터를 파일로 출력합니다. 결과메세지는 출력하지 않습니다.
.spool	파일경로를 넣지 않으면 현재 spool 기능을 끄고 stdout 으로 출력합니다.
!{Shell명령어}	Shell 명령어를 실행합니다.
@{시퀀스파일}	시퀀스파일 속의 명령어/쿼리를 순서대로 실행합니다.

2.1.3 IRIS 정보조회/수정 관련 명령어

2.1.3.1 테이블 명령어 테이블목록을 조회하고, 테이블 설정값을 수정합니다. 테이블 인덱스도 조회/수정합니다.

명령어	설명
.table list	현재 접속한 계정 소유의 테이블 목록을 출력합니다. -a 옵션을 뒤에 붙이면 전체테이블이 조회됩니다. -s 옵션을 뒤에 붙이면 시스템테이블이 조회됩니다.
.table info {테이블이름}	인자로 입력한 테이블의 정보를 출력합니다.
.table schema {테이블이름}	인자로 입력한 테이블의 스키마를 출력합니다. IRIS 옵션은 출력하지 않습니다.
.table index	현재 유저가 사용가능한 테이블들의 인덱스 목록을 출력합니다.
.table index –all	전체 인덱스 목록을 출력합니다.
.table index –add {인덱스명} {테이블명} (컬럼, ⋯)	LOCAL 테이블의 인덱스를 추가합니다.
.table index –del {인덱스명}	해당 LOCAL 테이블의 인덱스를 제거합니다.
.table expire {테이블명} –ram {램보관주기(분)}	해당 LOCAL 테이블의 램디스크 보관주기를 변경합니다.
.table expire {테이블명} –disk {디스크보관주기(분)}	해당 LOCAL 테이블의 디스크 보관주기를 변경합니다.

2.1.3.2 세션 관리 명령어 세션목록을 조회하거나 특정 세션을 종료합니다. 세션 종료는 사용자가 직접 1개씩 SID 를 입력해야 합니다.

명령어	설명
.session list	세션 목록을 출력합니다. 옵션을 사용해서 여러 조건에 맞는 세션목록만 출력할 수 있습니다. 기본값 현재시간부터 1시간 전까지 모든 타입에 대해서 가장 최근에 끝난 세션부터 10개를 출력합니다.
.session info {SID}	입력한 SID 에 해당하는 세션의 정보를 출력합니다. 전체 쿼리 문자열을 확인할 수 있습니다.
.session term {SID}	입력한 SID 에 해당하는 세션이 살아있는경우, 해당 세션을 강제종료합니다. 이미 종료된 상태이면 오류메세지를 출력합니다.

2.1.3.3 휴지통기능 명령어 휴지통기능을 사용하는 테이블 목록을 조회/관리 합니다. 비우기, 복원하기 등의 휴지통기능은 QUERY 로 지원합니다.

DDL, DML 항목을 참조하세요.

명령어	설명
.recyclebin list	현재 휴지통기능을 사용하는 테이블 목록을 출력합니다.
.recyclebin add {테이블이름}	인자로 입력한 테이블을 휴지통기능을 사용하도록 설정합니다.
.recyclebin del {테이블이름}	인자로 입력한 테이블을 휴지통기능을 사용하지 않도록 설정합니다.

2.1.3.4 파일 관리 명령어 테이블의 보관기관과 관련된 옵션을 조회/수정합니다. 위의 테이블명령어에서 설명한 expire 옵션을 반영하는 기준을 설정하는 옵션입니다.

명령어	설명
.pm list	현재 각 테이블별 램옵션, 디스크옵션을 출력합니다. 둘 다 $DEFAULT 인 경우는 출력되지 않습니다.
.pm {테이블이름} {램옵션} {디스크옵션}	이 명령어는 테이블의 램옵션, 디스크옵션을 설정하는 명령어입니다. 옵션은 테이블 생성시에 설정한 expire 값과 관련있는 옵션입니다. OFF : 데이터의 저장소변경/자동삭제 를 하지 않습니다. LOCAL_TIME_BASE : 서버시간 기준으로 expire 시간을 계산합니다. PARTITION_BASE : 현재 저장된 데이터 기준으로 expire 시간을 계산합니다. $DEFAULT : 기본옵션이며, .pm list 로 확인할 수 있습니다. 램 옵션 : 램디스크에 저장된 데이터가 하드디스크로 옮기는 시간의 옵션입니다. 디스크 옵션 : 디스크에 저장된 데이터가 자동삭제되는 시간의 옵션입니다.
.pm default {램옵션} {디스크옵션}	$DEFAULT 의 값을 변경합니다. 옵션설명은 위와 같습니다.
.pm ram {테이블이름} {램옵션}	테이블의 램옵션만을 수정합니다. 옵션설명은 위와 같습니다.
.pm disk {테이블이름} {디스크옵션}	테이블의 디스크옵션만을 수정합니다. 옵션설명은 위와 같습니다.
.pm del {테이블이름}	테이블의 옵션을 삭제합니다. 삭제되면 램옵션, 디스크옵션 둘 다 $DEFAULT 로 적용됩니다.

2.1.3.5 계정 명령어 계정 명령어는 password 를 제외하고 전부 root 계정으로 접속해야 사용가능합니다. root 계정의 비밀번호는 m6administrator 입니다.

명령어	설명
.user list	사용자목록을 출력합니다.
.user add {계정명:암호@호스트규칙}	계정을 추가합니다. 계정명, 암호, 접속가능한 호스트규칙을 함께 넣습니다. 모든 호스트에서 접속하게 하려면 * 을 사용합니다.
.user password {계정명:암호@호스트규칙}	계정의 암호를 수정합니다. 모든 계정은 자신의 암호를 수정할 수 있습니다. root 계정은 모든 계정의 암호를 수정할 수 있습니다.
.user del {계정명@호스트규칙}	계정을 삭제합니다. 소유 테이블은 삭제하지 않습니다.

2.1.4 System 정보조회 명령어

시스템정보를 조회합니다. 옵션을 사용해서 최신정보, 또는 기간별 정보를 출력합니다.

명령어	설명
.node list	현재 전체 노드정보를 출력합니다. IRIS 명령어의 ntop 과 동일합니다.
.statistics system	시스템정보 통계를 출력합니다. 다양한 옵션을 사용할 수 있으며, 기본으로 각 노드별 현재 시스템정보를 출력합니다.
.statistics table	테이블정보 통계를 출력합니다. 다양한 옵션을 사용할 수 있으며, 기본으로 전체 LOCAL 테이블의 최신 크기/파일갯수를 출력합니다.

2.1.5 Error 로그 조회 명령어

오류로그를 조회합니다.

명령어	설명
.error list	오류목록을 시간순서대로 출력합니다. 옵션을 사용해서 여러 조건에 맞는 오류목록만 출력할 수 있습니다. 기본값 : 현재시간부터 1시간 전까지 출력합니다.
.error count	오류 갯수를 출력합니다. 기본값은 list 명령과 동일합니다.

명령어	설명

2.1.6 노드 명령어

IRIS 노드의 정보를 조회하거나 상태를 변경합니다.

명령어	설명
.node list	현재 전체 노드의 상태정보를 출력합니다.
.node enable {노드ID}	해당 노드를 활성화 합니다. 활성화하기전에 해당 노드 복구작업이 성공적으로 종료되었는지 확인하세요.
.node disable {노드ID}	해당 노드를 비활성화 합니다. 모든 노드가 비활성화되면 기존 접속은 유지되나 신규접속은 불가능합니다.

2.1.7 Daemon 프로세스 명령어

IRIS Daemon 프로세스의 상태를 조회하거나 시작/종료합니다.

명령어	설명
.daemon list	Daemon 프로세스 목록을 출력합니다. 옵션을 사용해서 해당 조건에 맞는 항목만 확인할 수 있습니다. 지원 옵션 : –node {노드ID}, –abn, –name {프로세스명}, –cmd
.daemon act {노드ID} {MID}	해당 노드의 MID 에 해당하는 Daemon 프로세스를 시작합니다.
.daemon term {노드ID} {MID}	해당 노드의 MID 에 해당하는 Daemon 프로세스를 종료합니다.

2.1.8 부가기능 명령어

기타 유용한 부가기능 명령어 목록입니다.

명령어	설명
.whoami	현재 접속중인 IRIS 계정이름을 출력합니다.

2.2 DDL

2.2.1 CREATE TABLE

2.2.1.1 Syntax

```
create_table_stmt : CREATE TABLE STRING LPAREN table_schema RPAREN table_value

table_schema : column_define_list table_constraint_list

table_value : DATASCOPE STRING RAMEXPIRE value DISKEXPIRE value
              PARTITIONKEY value PARTITIONDATE value PARTITIONRANGE value

column_define_list : column_define
                   | column_define_list COMMA column_define

table_constraint_list : empty
                      | COMMA table_constraint_stmt
                      | COMMA table_constraint_stmt table_constraint_list

table_constraint_stmt : PRIMARY KEY LPAREN indexed_column_list RPAREN
                      | UNIQUE LPAREN indexed_column_list RPAREN

indexed_column_list : indexed_column
                    | indexed_column COMMA indexed_column_list

indexed_column : column_name
               | column_name ASC
               | column_name DESC

column_define : column_name column_type column_constraint_list

column_type : TEXT
            | REAL
            | NUMBER
            | BFILE
            | INTEGER

column_constraint_list : column_constraint
                       | column_constraint_list column_constraint
```

```
                    | empty

order_type : empty
           | ASC
           | DESC

column_constraint : NOT NULL
                  | DEFAULT value
                  | UNIQUE
                  | PRIMARY KEY autoincrement

autoincrement : AUTOINCREMENT
              | empty
```

2.2.1.2 기본 사용방법 일반적인 CREATE SQL 아래에 IRIS 테이블 옵션이 추가됩니다. 테이블 옵션은 모두 설정해야 합니다. 테이블 옵션의 설명은 아래에 소개되어 있습니다.

```
예시)
CREATE TABLE LOCAL_TEST_TABLE (
   k         TEXT,
   p         TEXT,
   a         TEXT
)
datascope       LOCAL
ramexpire       30
diskexpire      34200
partitionkey    k
partitiondate   p
partitionrange  10
;
```

2.2.1.3 IRIS 테이블 옵션

- datascope : 데이터 저장 방식 설정 [LOCAL OR GLOBAL]

- ramexpire : 램에 저장하는 시간 [GLOBAL 테이블일 경우 0]
- diskexpire : 디스크에 저장하는 시간 [GLOBAL 테이블 경우 0]
- partitionkey : KEY 로 사용할 값이 저장되는 컬럼 [GLOBAL 테이블 경우 None]
- partitiondate : PARTITION으로 사용할 값이 저장되는 컬럼 [GLOBAL 테이블 경우 None]
- partitionragne : 하나의 PARTITION 이 가지는 범위 [GLOBAL 테이블 경우 0]

2.2.1.4 지원하는 column-constraint

2.2.1.4.1 PRIMARY KEY [|ASC|DESC] [AUTOINCREMENT]

GLOBAL TABLE 에서만 정상동작합니다.

PRIMARY KEY 는 NULL 을 허용합니다. DESC 옵션은 현재 정상지원하지 않으며 ASC와 동일하게 동작합니다. 모든 컬럼타입에서 동작합니다.

2.2.1.4.2 INTEGER PRIMARY KEY 컬럼 INTEGER PRIMARY KEY 컬럼은 기본적으로 값증가기능(autoincrement)이 적용되어 있습니다.

정수형 데이터만 insert, update가 가능합니다. (예외로, NULL 값을 insert 하는 경우 키값을 생성해서 (현재 최대값+1) 넣습니다. 이외의 경우에는 datatype mismatch 오류를 출력합니다.)

AUTOINCREMENT 옵션은 INTEGER PRIMARY KEY 컬럼 만 가능합니다.

INTEGER PRIMARY KEY 컬럼에 AUTOINCREMENT 옵션을 넣으면 키값 생성 알고리즘이 변경되어 키값의 “monotonically increasing” 이 보장됨

니다.

AUTOINCREMENT 옵션 없을때 : 다음 키값이 키값의 최대값 (9223372036854775807) 을 넘으면 임의 (랜덤) 의 수를 현재 최 대키값으로 잡고 다음 키값을 생성합니다. (unique 속성 보장)

AUTOINCREMENT 옵션 있을때 : 다음 키값이 키값의 최대값 (9223372036854775807) 을 넘으면 "database or disk is full" 오 류메세지를 출력합니다.

AUTOINCREMENT 옵션은 ASC 또는 공백만 허용합니다. (공백이면 ASC로 설정합니다.)

2.2.1.4.3 DEFAULT {default-value} 컬럼 기본값을 설정합니다.

2.2.1.4.4 NOT NULL NULL 값이 들어오면 오류를 발생합니다. LOAD 작업시, 해당되는 데이터는 무시되고 나머지 데이터만 저장됩니다.

2.2.1.4.5 UNIQUE 중복값이 들어오면 오류를 발생합니다. GLOBAL TABLE 에서만 정상동작합니다.

2.2.1.5 지원하는 table-constraint PRIMARY KEY (ColumnName [, ColumnName]) : GLOBAL TABLE 에서만 정상동작합니다.

- 각 테이블별로 PRIMARY KEY 설정은 한 번만 할 수 있습니다. (위의 column-constraint 에서 이미 PRIMARY KEY 를 설정했다면 table-constraint 에서 설정할 수 없습니다.)
- PRIMARY KEY 는 NULL 을 허용합니다.

UNIQUE (ColumnName [, ColumnName]) : GLOBAL TABLE 에서만 정상동작합니다.

```
예시)
 CREATE TABLE LOCAL_TEST_TABLE (
    k         TEXT,
    p         TEXT,
    a         TEXT,
    PRIMARY KEY (k,p,a),
    UNIQUE (k,p,a)
 ) ...
```

2.2.2 DROP TABLE

2.2.2.1 Syntax

```
drop_table_stmt : DROP TABLE if_exists STRING PURGE
                | DROP TABLE if_exists STRING
```

테이블을 삭제합니다.

PURGE 는 휴지통기능과 연결된 키워드입니다. 해당 테이블이 휴지통기능을 사용하고 있으면, PURGE 를 사용하지 않은 경우 데이터를 휴지통 소속으로 변경합니다. PURGE 를 사용하면 메타데이터를 삭제합니다.

LOCAL 테이블을 삭제하면, 쿼리 실행시에는 메타데이터만 삭제하며 실제 파일은 백그라운드에서 삭제합니다.

```
예시)
DROP TABLE LOCAL_TEST_TABLE;
```

2.2.3 CREATE INDEX

2.2.3.1 Syntax

```
create_index_stmt : CREATE INDEX STRING ON STRING LPAREN indexed_column_list RPAREN
                  | CREATE UNIQUE INDEX STRING ON STRING LPAREN indexed_column_list RPAREN

indexed_column_list : indexed_column
                    | indexed_column COMMA indexed_column_list

indexed_column : column_name
               | column_name ASC
               | column_name DESC
```

인덱스를 생성합니다. LOCAL 테이블은 해당 DDL 대신 CLI 의 .table index add 를 사용하세요.

```
예시)
CREATE INDEX LOCAL_TEST_TABLE_IDX ON LOCAL_TEST_TABLE (a desc);
```

2.2.4 DROP INDEX

2.2.4.1 Syntax

```
drop_index_stmt : DROP INDEX if_exists STRING
```

인덱스를 삭제합니다. LOCAL 테이블은 해당 DDL 대신 CLI 의 .table index del 을 사용하세요.

```
예시)
DROP INDEX LOCAL_TEST_TABLE_IDX;
```

2.2.5 ALTER TABLE

2.2.5.1 Syntax

```
alter_table_stmt : ALTER TABLE table_name ADD COLUMN column_define
```

테이블 스키마를 변경합니다. 해당 DDL 로는 컬럼 추가만 지원합니다. LOCAL 테이블은 해당 DDL 대신 IRIS 마스터노드에서 Consol 명령어 TableSchemaEdit 를 사용하세요. ($M6_HOME/bin/Admin 디렉토리 안에 존재합니다.)

GLOBAL 테이블은 이 방법으로만 스키마 수정이 가능합니다.

```
예시)
ALTER TABLE LOCAL_TEST_TABLE ADD COLUMN t6 NUMBER UNIQUE;
```

2.3 HINT

IRIS 는 쿼리실행시 IRIS DB 에서 제공하는 기능을 활용하기 위해 HINT 라 불리는 문법을 사용할 수 있습니다.

실행할 쿼리 앞부부분에 붙여서 사용합니다. 어려개의 HINT 를 사용할 경우, 콤마(,) 로 구분해서 여러개를 함께 사용할 수 있습니다.

주의 : LOCAL 테이블 사용시에는 반드시 LOCATION HINT 와 함께 사용하세요.

```
/*+ HINT-String */ Query-String

ex)
/*+ BYPASS, LOCATION (PARTITION >= '20160101000000'
AND PARTITION < '20160201000000')*/
Select * from Test_Table_1;
```

2.3.1 HINT 종류

HINT 는 UPDATE, DELETE, SELECT 쿼리 에서만 사용할 수 있습니다.

HINT 이름	사용가능한 쿼리	설명/사용법
LOCATION	전체	LOCAL 테이블 대상으로 쿼리실행시 참조할 데이터범위를 설정합니다. 사용법 : 아래 참조
FORMAT	SELECT	쿼리 실행결과의 출력 형태를 변경합니다. 사용법 : 아래 참조
THREAD_COUNT	SELECT	해당 쿼리의 동시접근파일수를 설정해서 (기본값 50) 작업 부하 및 속도를 조절합니다. 사용법 : THREAD_COUNT = {자연수}
BYPASS	SELECT	쿼리 실행결과를 summary 작업 없이 바로 출력합니다. 사용법 : BYPASS
FORCE	전체	일부 데이터에서 쿼리오류가 발생해도 무시하고 나머지 정상데이터를 출력합니다. 사용법 : FORCE

LOCATION HINT

이 HINT 는 LOCAL 테이블에서만 사용합니다.

IRIS 의 LOCAL 테이블은 KEY, PARTITION 값을 사용해서 데이터를 분산저장합니다. 쿼리 실행할 때, LOCATION HINT 에서 참조할 KEY, PARTITION 값의 범위를 설정해서 쿼리의 실행시 부하를 줄이고 속도를 높일 수 있습니다. 괄호 안에 SQL WHERE 절과 동일한 문법으로 조건을 설정할 수 있습니다. 사용가능한 컬럼은 KEY, PARTITION 두 개 입니다.

예시 : KEY 는 key3 이고, PARTITION 은 2015년 1월 1일 부터 2월 직전까지의 범위로 참조하는 경우

```
/*+ LOCATION (
    KEY = 'key3'
    AND PARTITION >= '20160101000000'
    AND PARTITION < '20160201000000'
) */
```

FORMAT HINT

출력결과의 데이터 포맷을 설정합니다. 중간 SubQuery 에는 사용할 수 없으며, 최종결과에만 사용할 수 있습니다. 각 Column 별로 데이터 출력포맷을 설정하며 '인덱스번호 = 포맷' 형태로 되어있습니다. 인덱스는 0 부터 시작합니다. 문자열 포맷 규칙은 Python 언어에서 사용하는 문자열 포맷과 같습니다.

데이터 타입	포맷 문자열	사용법
문자열	%s	%{최소길이}.{최대길이}s
정수	%d	%{최소길이}.{0으로 채우는 최소길이}d
실수	%f	%{최소길이(소숫점영역 포함)}.{소숫점자리 고정길이}f

예시 : 첫번째 컬럼은 3자리 숫자이며 앞은 0으로 채우고, 세번째 컬럼은 소수점 2번째까지 표시

```
/*+FORMAT(0='%03d',2='%.2f')*/ select id,name, point from ACCOUNT;
```

- 기본은 우측정렬이며, %음수s 처럼 가운데숫자를 음수로 설정하면 좌측정렬이 됩니다.
- 정수, 실수의 경우 가운데숫자가 0으로 시작하면 앞부분 빈공간을 0으로 채웁니다.
- 하나의 값이라도 형변환이 실패하면 쿼리작업 전체가 실패합니다.
- HINT 만으로는 문자열 컬럼을 정수 또는 실수로 형변환하지 못합니다.
- 정수, 실수는 문자열로 형변환할 수 있습니다.
- 실수 출력시 표현할 소숫점자리보다 실제 값의 소숫점자리가 길 경우, 자동으로 반올림됩니다.

2.4 DML

2.4.1 INSERT

2.4.1.1 Syntax

```
insert_statement : INSERT INTO STRING LPAREN column_list RPAREN VALUES
                   LPAREN param_list RPAREN

column_list : column_name
            | column_name COMMA column_list

param_list : param COMMA param_list
           | param

param : expr
      | value
      | STAR
```

데이터를 1줄 추가합니다. HINT 를 사용할 수 없으며, BIGIN-END 를 사용할 수 없습니다.

LOCAL 테이블에서 사용한 경우 key,partition 값은 테이블생성시 설정한 컬럼의 값을 사용합니다.

```
예시) k : keypartition, p : datepartition
INSERT INTO LOCAL_TEST_TABLE (k, p, a) VALUES ('k2', '20110616000000', '1');
```

2.4.2 UPDATE

2.4.2.1 Syntax

```
update_statement : UPDATE table_name SET update_column_list where_expression

update_column_list : update_column
                   | update_column COMMA update_column_list

update_column : column_name EQ expr
```

LOCAL 테이블은 LOCATION HINT 와 함께 사용하세요. LOCATION HINT 에 해당하는 데이터파일 속에서 WHERE 절 조건에 맞는 데이터를 수정합니다.

```
예시)
UPDATE LOCAL_TEST_TABLE SET a = 'update';
```

2.4.3 DELETE

2.4.3.1 Syntax

```
delete_statement : DELETE FROM table_name where_expression
```

LOCAL 테이블은 LOCATION HINT 와 함께 사용하세요. LOCATION HINT 에 해당하는 데이터파일 속에서 WHERE 절 조건에 맞는 데이터를 삭제합니다.

```
예시)
DELETE FROM LOCAL_TEST_TABLE WHERE a = 'update';
```

2.4.4 PURGE

2.4.4.1 Syntax

```
purge_statement : PURGE TABLE job_id
                | PURGE BACKEND job_id
                | PURGE RECYCLEBIN
                | PURGE BACKEND job_id LPAREN expr RPAREN
```

휴지통기능을 사용하는 테이블을 대상으로 사용할 수 있습니다. 휴지통에 있는 테이블/백엔드파일을 완전히 삭제합니다. 삭제시 메타데이터만 삭제하며 실제 파일은 백그라운드로 삭제됩니다. HINT 를 사용하지 않습니다.

```
예시)
purge table LOCAL_TEST_TABLE;
```

2.4.5 FLASHBACK

2.4.5.1 Syntax

```
flashback_statement : FLASHBACK TABLE job_id flashback_table_mode_stmt
                    | FLASHBACK BACKEND job_id flashback_backend_mode_stmt
                      flashback_backend_option_stmt

flashback_table_mode_stmt : OVERWRITE
                          | empty

flashback_backend_mode_stmt : RECYCLEBINBASE
                            | TABLEBASE
                            | empty

flashback_backend_option_stmt : STRING TABLE
                              | STRING STRING
                              | empty
```

휴지통기능을 사용하는 테이블을 대상으로 사용할 수 있습니다. 휴지통에 있는 테이블/백엔드파일을 복원합니다. HINT 를 사용하지 않습니다.

```
예시)
FLASHBACK TABLE LOCAL_TEST_TABLE;
```

2.5 SELECT 쿼리

LOCAL 테이블에 대한 쿼리를 실행할 때에는 LOCATION HINT 를 반드시 사용하세요.

2.5.1 Syntax

```
select_statement : SELECT result_expression FROM table_expression where_expression
                   group_by_expression compound_op_select order_by_expression
                   limit_expression

result_expression : result_column
                  | result_column COMMA result_expression

result_column : STAR
              | column_name
              | alias_expression
              | term
              | expr

alias_expression : expr STRING
                 | expr AS STRING
                 | LPAREN select_statement RPAREN STRING
                 | LPAREN select_statement RPAREN AS STRING

table_expression : table_list

table_list : table_name
           | table_name join_expr_list

join_expr_list : join_expr join_expr_list
               | join_expr

join_expr : empty
          | join_op table_name join_args

join_op : COMMA
        | INNER JOIN
```

```
        | LEFT OUTER JOIN

join_args : ON expr
          | empty

where_expression : [where] expr
                 | empty

group_by_expression : GROUP BY result_expression
                    | empty

compound_op_select : compound_op select_core
                   | empty

compound_op : UNION
            | UNION ALL
            | INTERSECT
            | EXCEPT

order_by_expression : ORDER BY order_by_result_expression
                    | empty

order_by_result_expression : result_column
                           | result_column ASC
                           | result_column DESC
                           | result_column COMMA order_by_result_expression
                           | result_column ASC COMMA order_by_result_expression
                           | result_column DESC COMMA order_by_result_expression

limit_expression : ideallimit_expression
                 | mlimit_expression
                 | dlimit_expression

ideallimit_expression : LIMIT NUMBER
                      | LIMIT NUMBER COMMA NUMBER
                      | empty

mlimit_expression : MLIMIT NUMBER
                  | MLIMIT NUMBER COMMA NUMBER
                  | empty
```

```
dlimit_expression : DLIMIT NUMBER
                  | DLIMIT NUMBER COMMA NUMBER
                  | empty
```

2.5.2 기본 형태

```
예시)
/*+ BYPASS, LOCATION (KEY = 'k1' AND PARTITION >= '20150101000000'
AND PARTITION < '20150102000000') */
SELECT * FROM LOCAL_TEST_TABLE;

/*+ LOCATION (PARTITION >= '20150101000000' AND PARTITION < '20150102000000') */
SELECT sum(a) / count(a) FROM LOCAL_TEST_TABLE WHERE a > 10;
```

2.5.3 유의사항

[*] 컬럼 사용에 제약사항이 있습니다.

- Table Alias 를 지원하지 않습니다.

```
FAIL : select a.* from (select * from X) a;
```

- 단일 테이블을 사용한 에서만 사용할 수 있습니다.

```
FAIL : select * from X, Y;
```

2.6 JOIN

IRIS 는 JOIN 사용시 제약사항이 있습니다.

모든 SubQuery 를 포함해서, LOCAL 타입 테이블은 0개, 혹은 1개만 존재할 수 있으며, 테이블이름이 같더라도 여러번 사용했으면 여려개가 존재하는 것으로 간주합니다.

사용가능한 조인은 INNER JOIN, LEFT OUTER JOIN 두 가지 입니다.

2.6.1 INNER JOIN

```
FROM <TableName> INNER JOIN <TableName> [ON <expr>] [WHERE <expr>] ...
FROM <TableName>, <TableName> ... [WHERE <expr>] ...
```

일반적인 n * n JOIN 입니다.

데이터량이 많아질 수 있으므로 LOCATION HINT 와 WHERE 절을 잘 활용하세요. 중간데이터량이 매우 많은 경우 IRIS 에서 해당 작업을 강제종료할 수 있습니다.

2.6.2 LEFT OUTER JOIN

```
FROM <A> LEFT OUTER JOIN <B> [ON <expr>] [WHERE <expr>] ...
```

LOCAL 테이블은 B 의 위치에 사용합니다.

JOIN 연산은 각 분산저장된 파일 단위로 수행됩니다. 각 분산파일단위로 JOIN 수행시 일부 데이터가 누락될 수 있으므로 값이 없을 때 null 이 되는 B 의 위치에 LOCAL 테이블을 사용합니다.

A 는 GLOBAL 테이블을, B 에는 LOCAL 테이블또는 GLOBAL 테이블을 위치시켜서 사용합니다.

B(LOCAL TABLE) 만 대상으로 하는 WHERE 조건은 on 절에 넣습니다.

나머지 WHERE 조건은 WHERE 절에 넣습니다.

2.6.3 기타 : Oracle JOIN 과 비교

ORACLE 은 릴리즈 6부터 Oracle-specific한 문법으로 사용되는 left outer-join의 특별한 형식을 지원합니다.

ORACLE 의 left outer join 문법과 IRIS 의 문법은 동일하지 않습니다. 아래는 동일한 작업을 각각의 SQL 문법으로 표현한 것입니다.

```
오라클의 경우

SELECT T1.d, T2.c
FROM T1, T2
WHERE T1.x = T2.x (+);


IRIS 의 경우

SELECT T1.d, T2.c
FROM T1 LEFT OUTER JOIN T2
ON (T1.x = T2.x);
```

2.7 지원 함수 목록

IRIS SQL 에서 지원하는 함수 목록입니다.

Group	Attribute	Global Table	Local Table
Aggregate Func	count()	O	O 별첨
	max()	O	O 별첨
	min()	O	O 별첨
	sum()	O	O 별첨
Func	concat()	O 별첨	O 별첨
	abs()	O	O
	char()	O	O
	coalesce()	O	O
	ifnull()	O	O
	hex()	O	O

Group	Attribute	Global Table	Local Table
	length()	O	O
	lower()	O	O
	ltrim()	O	O
	nullif()	O	O
	quote()	O	O
	randomblob()	O	O
	replace()	O	O
	round()	O	O
	rtrim()	O	O
	sqlite_compileoption_get()	O	O
	sqlite_compileoption_used()	O	O
	substr()	O	O
	trim()	O	O
	typeof()	O	O
	unicode()	O	O
	upper()	O	O
	zeroblob()	O	O
	date()	O	O
	time()	O	O
	datetime()	O	O
	julianday()	O	O
	strftime()	O	O
	to_number()	X 별첨	X 별첨
	to_date()	O 별첨	O 별첨
	to_char()	O	O
	power()	O	O
	instr()	O	O
	lpad()	O	O
	rpad()	O	O
	ifnull2()	O	O
	ceil()	O	O
	floor()	O	O
	sqrt()	O	O
	b64d()	O	O
	fstr()	O	O
	rmhint()	O	O
	rmcar()	O	O
	encrypt()	O	O
	decrypt()	O	O

Group	Attribute	Global Table	Local Table
	sha256()	O	O
	passwd()	O	O
Paragraph	group by	O	O 별첨
	order by[asc, desc]	O	O 별첨
	case when then end	O	O
	having	X	X
	limit	O 별첨	O 별첨
Operator	[=, >, <, >=, <=, <>]	O	O
	between and	O	O
	In	O	O
	like	O	O
	Is null, is not null	O	O
	and, or, not	O	O
	escape	O 별첨	O 별첨
	distinct	O	O

2.7.1 별첨항목

Aggregation 함수(sum, min, max, count), Group by, Order by 사용법

IRIS 는 LOCAL 테이블 사용시 최상위쿼리에서만 Aggregation 함수나나 Group by, Order by 연산이 정상적으로 동작합니다. 하위쿼리(SubQuery)에서도 사용은 가능하지만, 해당 작업은 개별 파일별로 실행되기 때문에 일부 데이터만으로 연산을 수행하며, 의미상 원하는 값이 나오지 않습니다. 사용법은 일반 함수와 동일하며, IRIS 내부에서 해당 함수를 Map/Reduce 작업에 맞게 쿼리를 변형해서 사용합니다.

ex) 정상실행되지 않는 경우

```
SELECT a_sum FROM (SELECT sum(a) AS a_sum FROM ONE_LOCAL_TABLE Group by a)
Order by a_sum;
```

ex) 정상실행되는 경우

```
SELECT sum(val), sum(val2), count(val) FROM ( SELECT val, val * val AS val2
FROM ONE_LOCAL_TABLE);
```

concat

- 사용법 : A||B

문자열을 더합니다. 한쪽이 NULL 일 경우 NULL 을 리턴합니다.

to_number

타입이 연산과정에서 자동으로 변환됩니다. 숫자로 변환가능한 문자열은 그 숫자로 변환합니다. 숫자로 변환불가능한 문자열은 0 으로 취급합니다.

limit

IRIS 에서는 limit 문법이 총 3가지 있습니다.

- limit
  - 일반 limit 문법과 동일한 결과가 나옵니다.
  - 전체 데이터를 가져와서 최종 summary 할 때 limit 를 적용합니다.
- dlimit
  - 분할된 데이터에 limit 를 적용해서 가져옵니다.
  - 최종 summary 할 때는 limit 가 적용되지 않습니다.
- mlimit
  - limit 와 dlimit 을 합친 것처럼 동작합니다.

– 분할된 데이터에 limit 를 적용해서 가져온 후 최종 summary 할 때도 limit 를 적용합니다.

escape

like 와 함께 사용합니다. like 뒤에 위치하며, like 조건의 문자열에서 사용할 escape 문자를 설정합니다.

```
LIKE '<condition>' escape '<char>'

ex) % 를 와일드카드가 아니라 문자열로 사용하기 위해 \ 를 escape 문자열로 설정하고 \% 를 사용함.
    select * from TABLE_1 where a like '123\%' escape '\';
```

2.7.2 지원함수 명세

upper

- 사용법 : upper(A)

영문 소문자를 대문자로 변환합니다.

substr

- 사용법 : substr(X,Y,Z) , substr(X,Y)

X 문자열을 Y위치부터 Z길이만큼 부분문자열을 만듭니다. substr(X,Y) 으로 사용하면 X 문자열을 Y위치부터 X문자열 끝까지 부분문자열을 만듭니다.

Y위치는 양수이면 1이첫번째글자인덱스입니다. 커질수록끝으로이 동합니다. 0 역시 첫번째 글자를 가리키지만 인덱스연산시 혼동할 수 있습니다. 음수이면 -1이 마지막 글자 인덱스입니다. 작아질수록 처음 으로 이동합니다.

Z값은 항상 자연수여야하고, 음수일 경우 해당 값을 사용하지 않고 substr(X,Y) 형태로 실행됩니다.

Y, Z 값으로 만들어진 실제 인덱스범위가 첫번째 글자를 1로 기준으 로 해서 (1 뒤는 -1로 가정합니다.)

(음수, 음수) 인 경우(substr('123',-7,2) => -4,-2) , 실패로 간주 하고 전체 문자열을 반환합니다.

(음수, 양수) 인 경우(substr('123',-7,6) => -4,2) , 범위에 허용 되는 문자열을 생성합니다. ('12')

(양수, 양수) 인 경우(substr('123',2,6) => 2,8) , 범위에 허용되 는 문자열을 생성합니다. ('23')

시작위치가 글자길이보다 크면 빈문자열을 생성합니다. substr('123',5) : ''

length

- 사용법 : length(X)

문자열 X 의 길이를 출력합니다.

round

- 사용법 : round(X,Y), round(X)

숫자 X 를 소수점 Y+1번째 수에서 반올림합니다.

Y값은 0을 포함한 자연수만 정상작동하며, 나머지 값의 경우는 0으로 처리합니다. 소수점 이하에서만 반올림이 가능합니다.

round(X) 으로 사용하면 소수점 첫번째 수에서 반올림합니다. (Y값 이 0인 경우와 동일합니다.)

to_date

- 사용법 : to date(X, FORMAT)

X에 해당하는 값을 FORMAT 형태로 인식해서 Timestamp 값으로 변 환합니다. 1970/01/01 00:00:00 GMT 를 시작으로, 현재까지 경과한 초를 계산합니다.

SYSDATE 키워드를 사용하면 해당 값이 현재시간 문자열로 사용됩니다.

```
현재 시간이 2015년 1월 1일 13시 30분 30초 인 경우
insert into GLOBAL TEST TABLE (k,p,a) values ('a', SYSDATE, 1);
-> insert into GLOBAL TEST TABLE (k,p,a) values ('a', '20150101133030', 1);
```

FORMAT 키워드

Format	Output
%d	day of month: 00
%f	fractional seconds: SS.SSS
%H	hour: 00-24
%j	day of year: 001-366
%J	Julian day number
%m	month: 01-12
%M	minute: 00-59
%s	seconds since 1970-01-01
%S	seconds: 00-59
%w	day of week 0-6 with Sunday==0
%W	week of year: 00-53
%Y	year: 0000-9999
%%	%

```
ex)
SELECT to_date( p, '%Y%m%d%H%M%S' ) FROM LOCAL_TEST_TABLE;
TO_DATE(P, '%Y%m%d%H%M%S')
============================================================
1355106008.0
1355106008.0
1355106008.0
```

to_char

- 사용법 : to char(X, FORMAT)

X의 날짜문자열(YYYYmmddHHMMSS 형태)을 FORMAT 형태로 출력합니다. (위의 to date 에서 사용하는 FORMAT 과 문법이 다릅니다.) X 가 날짜문자열이 아니면 오류를 출력합니다.

FORMAT 에서 아래의 키워드를 X 로부터 생성한 날짜정보로 변환합니다.

Format	Output
YYYY	년
MM	월
DD	일
HH24	시간
MI	분
SS	초
D	요일 ('1' : '일요일', '2' : '월요일', ··· , '7' : '토요일')

```
ex)
SELECT to_char( p, 'YYYY year MM month DD day' ) FROM LOCAL_TEST_TABLE;
TO_CHAR(P, 'YYYY year MM month DD day')
=======================================================
2011 year 06 month 16 day
2011 year 06 month 16 day
```

ifnull

- 사용법 : ifnull(A, B)

A 컬럼값이 NULL 인지 확인해서 NULL 이면 B 를 리턴하고 아니면 원래 값을 리턴합니다.

sum

- 사용법 : sum(X)

해당 컬럼의 값을 모두 더합니다. 정수나 실수로 변환되지 않는 값은 0 으로 인식합니다. 모든 값이 0을 제외한 정수인 경우에만 정수로 출력됩니다.

max

- 사용법 : max(X)

해당 컬럼에서 가장 큰 값을 반환합니다. 기본으로 문자열비교로 크고 작음을 판단하므로, FORMAT HINT 를 사용해서 형변환을 해야합니다.

min

- 사용법 : min(X)

해당 컬럼에서 가장 작은 값을 반환합니다. 기본으로 문자열비교로 크고 작음을 판단하므로, FORMAT HINT 를 사용해서 형변환을 해야합니다.

count

- 사용법 : count(X) , count(*)

테이블의 해당 컬럼의 NULL 이 아닌 레코드갯수를 반환합니다. count(*)은 테이블의 전체 레코드갯수를 반환합니다.