### 쿼리 문법

#### 명령문 형식

쿼리는 1개 이상의 명령문으로 구성됩니다. 명령문을 구성하는 기본 단위는 명령어와 옵션, 대상 개체(object)입니다.

대상 개체가 있는 명령문

개체는 로그 수집기나 데이터 스트림, 로그 파서, 데이터가 저장된 테이블, 테이블 인덱스일 수 있습니다. 서브쿼리문, 함수, 프로시저 등으로 구성된 표현식을 호출해 결과 값을 대상으로 실행할 수도 있습니다. 이와 같은 개체를 대상으로 실행하는 명령문의 구성 형식은 다음과 같습니다.

command-name [opt\_1=VALUE] [opt\_2=VALUE] ... OBJECT[, ...]

가장 단순한 명령문의 예로, 기본 시스템 테이블인 araqne\_query\_logs에서 데이터를 조회하는 명령문은 다음과 같습니다.

table araqne\_query\_logs

대상 개체가 없는 명령문

개체가 없는 명령문은 다른 명령문으로부터 데이터를 전달받아 처리하는 명령어에서 주로 사용합니다. 예를 들어, [decodedns](https://docs.logpresso.com/ko/query/decodedns-command) 명령어가 그렇습니다. 구성 형식은 다음과 같습니다.

FORWADING\_STATEMENT | command-name [opt\_1=VALUE] [opt\_2=VALUE] [opt\_N=VALUE] ...

이와 같은 명령어들은 개체를 전달하는 선행 명령문(FORWARDING\_STATEMENT)이 반환하는 출력을 파이프(|)를 통해 입력으로 전달받아 처리합니다.

#### 파이프를 이용한 입력 처리

로그프레소에서 명령문은 파이프를(|)를 이용해 명령문의 출력을 다른 명령문에 입력으로 전달합니다. 예를 들어, **araqne\_query\_log** 테이블에서 **login\_name** 필드가 "root"인 로그만 조회하려면 다음과 같이 쿼리합니다.

table araqne\_query\_logs | search login\_name == "root"

이 쿼리문은 root 계정으로 실행된 쿼리문들을 보여줍니다. root 문자열을 포함하는 행이 10분 단위로 몇 건씩 발생하는지 10분 단위로 통계를 계산하려면 다음과 같이 쿼리합니다:

table araqne\_query\_logs| search login\_name == "root"| timechart span=10m count

이와 같이 첫번째 명령문의 출력을 두번째 명령문의 입력으로, 두번째 명령문의 출력을 세번째 명령문의 입력으로 전달하는 과정을 거쳐, 세번째 명령문의 출력이 전체 쿼리문의 결과로 전달됩니다. 쿼리 결과는 클라이언트의 요청 형태에 따라 디스크에 임시로 기록되거나, 네트워크를 경유하여 즉시 스트리밍할 수 있습니다.

#### 서브쿼리

어떤 쿼리 명령어는 명령문 안에 중첩된 명령문을 실행하고, 실행된 결과를 받아서 실행합니다. 중첩된 명령문을 서브쿼리(subquery)라 합니다.

서브쿼리문은 입력할 때 대괄호 쌍([ ])으로 감싸서 표현합니다. 서브쿼리는 전체 쿼리문 실행 시점에 상위 쿼리문보다 앞서 실행됩니다. 서브쿼리가 반환하는 레코드를 쿼리 명령어가 받아서 실행합니다.

서브쿼리가 있을 경우, 명령문 구조는 다음과 같습니다.

command [ SUBCOMMAND\_STATEMENT ]

#### 명령문의 주석 처리

주석 처리 명령어 '#'을 이용해 명령행에 설명을 삽입하거나, 단일 명령행 또는 연속된 명령행을 주석 처리할 수 있습니다. 쿼리 입력 상자에서 주석 처리된 명령행은 회색으로 표시됩니다.

단일 명령행의 주석 처리

명령문 첫머리에 '#'을 삽입해 주석 처리할 수 있습니다.

'#' 뒤에 공백문자가 필요합니다. 공백문자가 없으면 주석 처리가 되지 않습니다.

# 최근 1시간 동안 sys\_cpu\_log에 기록된 CPU 사용률 조회하기 | table duration=1h sys\_cpu\_logs | # eval total = kernel + user

위 예시에서는 최근 1시간 동안 sys\_cpu\_log에 기록된 CPU 사용률 조회하기, eval total = kernel + user가 주석 처리되어 table duration=1h sys\_cpu\_logs만 실행합니다.

여러줄의 명령행을 주석 처리

# [과 ]으로 주석 처리할 명령행들을 감싸서 주석 처리합니다.

table duration=1h sys\_cpu\_logs | # [ eval total = kernel + user | search total > 10 ] | sort \_time

위 예시에서는 대괄호 안의 명령문이 주석 처리되어 실제 실행 쿼리는 table duration=1h sys\_cpu\_logs | sort \_time 입니다.

주석 처리 명령어 #은 뒤에 오는 문자열 뿐만 아니라, 서브쿼리를 감싸는 대괄호 쌍([ ]) 안에 있는 파이프(|)를 무시합니다. 서브쿼리에 줄바꿈이 있어도 무시합니다. 다시 말하면, 서브쿼리문 전체를 무시하고 대괄호 쌍 밖에 파이프가 나타날 때까지 주석 처리합니다.

table sys\_cpu\_logs | # union [ table sys\_cpu\_logs | limit 30 ] | eval total = kernel + user

위 예시에서 서브쿼리인 union [ table sys\_cpu\_logs | limit 30 ]는 모두 주석 처리되어 실제 실행 쿼리는 table sys\_cpu\_logs | eval total = kernel + user 입니다.

#### 쿼리 매개변수

쿼리 매개변수에 값을 할당하고, 필요할 때 호출해 사용할 수 있습니다. 상수 대신에 함수 등을 이용한 표현식을 이용함으로써 동적으로 값을 할당해 쿼리를 실행할 때 유용합니다. 예를 들어, 예약된 쿼리를 실행할 때 현재 날짜를 기준으로 일주일 범위의 데이터를 조회해서 처리하거나, 프로시저 실행 시 사용자가 입력한 매개변수 값을 이용하여 쿼리를 실행하려고 할 때 쿼리 매개변수를 사용합니다.

매개변수의 선언

매개변수는 [set](https://docs.logpresso.com/ko/query/set-command) 혹은 [setq](https://docs.logpresso.com/ko/query/setq-command) 명령을 이용해 선언합니다.

매개변수의 참조

매개변수에 할당된 값은 매개변수 참조 함수인 [$()](https://docs.logpresso.com/ko/query/$-function)으로 참조합니다.

#### 함수

쿼리문에 함수를 사용할 수 있습니다. 함수는 표현식을 사용할 수 있는 곳이면 어디든 사용 가능합니다.

#### 프로시저

로그프레소는 사전 정의된 쿼리문을 함수처럼 호출할 수 있는 프로시저 기능을 제공합니다. DBMS의 프로시저와 유사한 기능으로, 다음과 같은 장점을 제공합니다.

**재사용성 및 유지보수 향상**

프로시저를 통해 특정한 기능을 제공하는 쿼리를 모듈화함으로써 재사용성을 높일 수 있습니다. 사용자에게는 프로시저 이름과 사용할 매개변수만 알려주면 됩니다. 사용자는 반복적인 쿼리를 쓰지 않아도 되고, 쿼리문이 간결해져 유지보수하기 쉬워집니다.

**보안성 향상**

[dbquery](https://docs.logpresso.com/ko/query/dbquery-command), [ftp](https://docs.logpresso.com/ko/query/ftp-command), [sftp](https://docs.logpresso.com/ko/query/sftp-command) 등 외부의 시스템에 접속하는 명령어는 프로파일 사용 권한이 필요합니다. 프로파일 권한을 사용자에게 직접 부여하면 외부 시스템에서 임의의 작업을 수행할 수 있으므로 안전하지 않습니다. 사용자가 특정한 관리자 권한이 필요한 명령을 실행하거나 로컬/원격 호스트에 임의의 작업을 수행할 수 있는 명령을 프로시저로 구성한 다음 사용자 권한을 관리함으로써 시스템 전체의 관리자 권한을 부여하지 않고도 원하는 작업을 사용자 계정으로 사용할 수 있게 해줍니다. 원본데이터의 일부만 조회하도록 제한하거나, 원본 데이터를 마스킹해 보여주는 방식으로 응용할 수 있습니다.

**로그프레소의 설정 정보 접근**

로그프레소의 시스템 테이블은 관리자 권한이 있어야 접근할 수 있습니다. 로그프레소의 시스템 설정 정보를 사용자가 접근해야 할 때 프로시저를 통해 접근할 수 있도록 해줍니다.

프로시저 정의하기

웹 콘솔에서 프로시저를 정의하고 관리할 수 있습니다. 프로시저를 관리하는 기능은 다음 경로에서 사용할 수 있습니다.

* (ENT, STD) **쿼리 > 프로시저**
* (SNR) **분석 > 프로시저**

프로시저로 사용할 쿼리문은 프로시저를 호출할 때 사용할 매개변수 또는 사용자 정의 필드를 포함할 수 있습니다.

프로시저 화면에서 정의하는 쿼리는 [$()](https://docs.logpresso.com/ko/query/$-function) 함수를 사용하여 사용자가 프로시저를 호출할 때 넘긴 매개변수를 참조할 수 있습니다. 예를 들면 아래와 같습니다:

table duration=1d sys\_cpu\_logs | search kernel + user >= $("threshold")

예시된 쿼리문에서 threshhold가 매개변수입니다.

프로시저 작성 시 가장 흔한 실수는 [$()](https://docs.logpresso.com/ko/query/$-function) 함수 참조가 매크로처럼 치환된다고 생각하여 쿼리를 작성하는 것입니다. [$()](https://docs.logpresso.com/ko/query/$-function) 함수는 쿼리 명령어에서 표현식을 입력할 수 있는 위치에만 지정할 수 있습니다. 예를 들어, 아래의 프로시저는 [dbquery](https://docs.logpresso.com/ko/query/dbquery-command)가 임의의 SQL 문장 입력을 표현식으로 지원하지 않으므로 올바른 쿼리가 아닙니다:

dbquery USERDB $("sql")

프로시저의 호출

프로시저를 호출해 실행하는 명령어로는 [proc](https://docs.logpresso.com/ko/query/proc-command)이 있습니다. 호출 방법은 명령어의 설명을 참조하십시오.