DOCRUZER Ver 3.5

User Manual

(주)코난테크놀로지

서울시 강남구 대치4동 890번지 유니온스틸빌딩 17, 18층

내용 목차

그림 목차

개요 및 특징

이 장에서는 KONAN DOCRUZER의 개요와 주요 특징에 대해 설명한다.

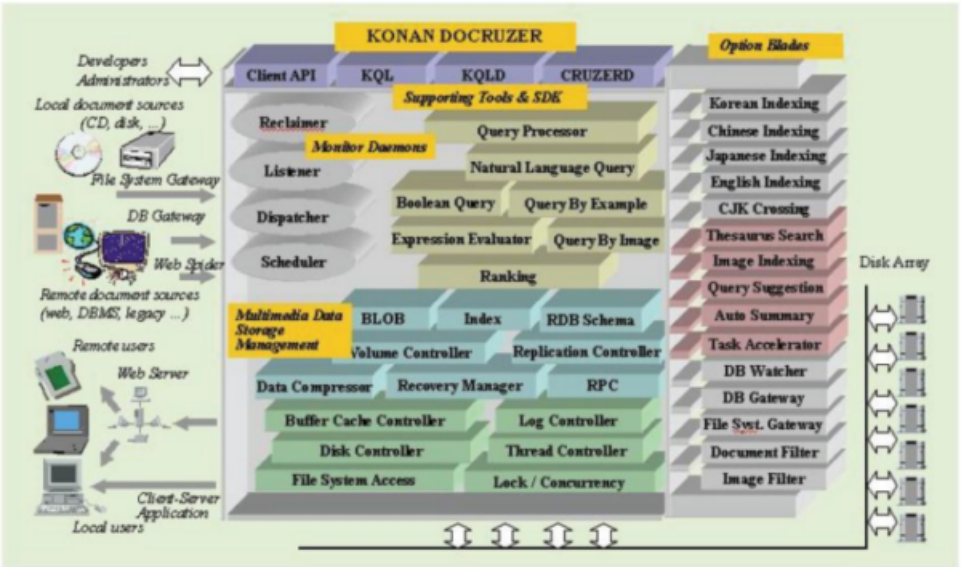
1.1. 개요

DOCRUZER(독크루저)는 다국어를 지원하는 고성능 자연어 분석 기술과 대용량 저장 기술이 결합된 강력한 검색엔진이다.

1.1.1. 시스템 구조

독크루저는 명령어 및 명령어 해석기 KQL, 검색 데몬 DOCRUZERD, SDK(System Development Kit), 각종 데이터 및 옵션 팩으로 이루어져 있다. 자료의 저장은 기능 및 성능 품질의 향상을 위해 상용 DBMS(Database Management System)를 이용하지 않고 전용 저장 관리 시스템을 이용한다. KQL은 저장소를 관리하는 명령어, 명령어 해석기, 클라이언트 API 등으로 이루어져 있다.

다음은 독크루저의 시스템 구조를 간략히 도식화한 것이다.



[그림 1.1] 독크루저 시스템 구조

DOCRUZERD는 KQL의 하위 레벨 데이터 관리 기능의 기반 위에서 검색 서비스 구성을 용이하게 해주는 맞춤형 검색 데몬이다. 각종 옵션 팩은 필요에 따라 독크루저에 추가할 수 있는 구조로 되어 있어서 기능 추가 시 독크루저의 확장이 용이하다.

1.1.2. 디스크 볼륨 구조

독크루저에서는 물리적/논리적으로 분리된 데이터의 저장 단위를 볼륨(volume)이라 하며 하나의 볼륨은 다른 볼륨과 독립된 디스크상의 공간을 배타적으로 점유한다. 볼륨은 파일 시스템에 파일로 저장된다. 이때 저장 단위가 되는 파일을 세그먼트(segment)라고 한다. 세그먼트 파일명은 ts00000.vdf부터 1씩 차례로 증가하는 방식으로 표현된다. 0 번 세그먼트를 제외한 모든 세그먼트 파일의 최대 크기는 일정하며, 운영체제가 지원하는 파일의 최대 크기보다 작거나 같다. 세그먼트 파일의 크기는 사용자 지정 사항이 아니므로 임의로 변경할 수는 없다. 볼륨은 32 비트 운영체제를 기준으로 최대 64 TB까지 지원되며 유효 크기는 12 TB이다.

한번 만들어진 세그먼트 파일은 재사용이 가능하므로 삭제할 필요가 없다. 세그먼트 파일들이 위치한 디렉터리가 볼륨의 데이터 위치가 된다. 세그먼트 파일들을 다른 디렉터리로 이동한 뒤 해당 디렉터리를 볼륨으로 설정하는 것도 가능하다.

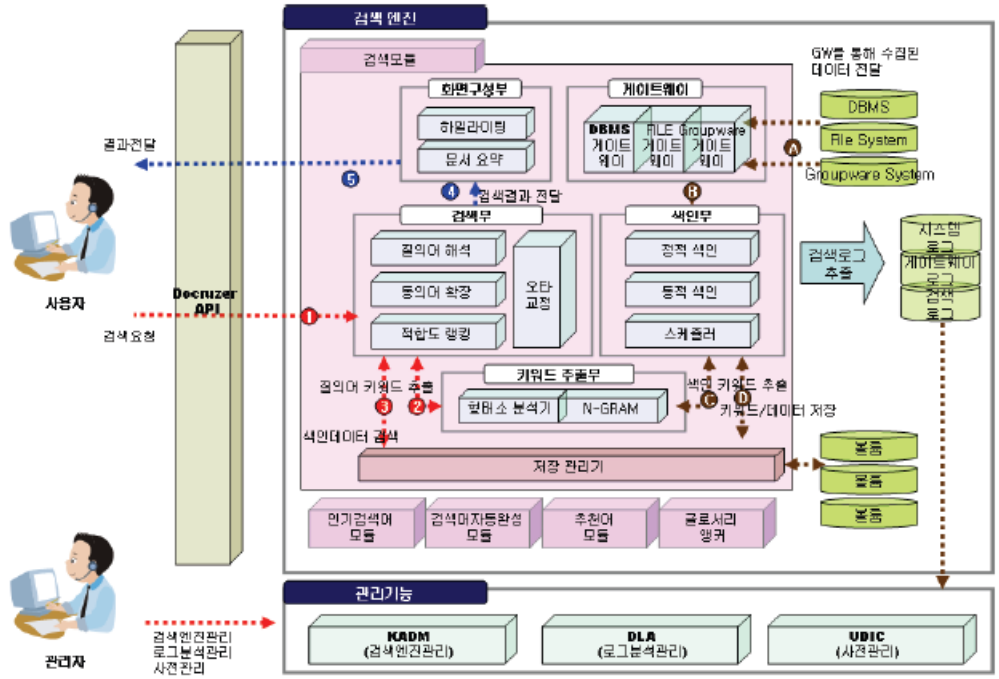
하나의 볼륨에 대해 하나 이상의 데이터 디렉터리를 지정하는 것도 가능하다. 볼륨이 존재하지 않는 상태에서 볼륨 생성 명령어를 수행하면 초기 세그먼트 파일이 자동으로 생성된다. 검색 데이터가 증가하면 세그먼트 파일은 자동적으로 확장된다. 세그먼트 파일이 증가하는 과정에서 현재 이용 중인 디바이스에 가용 공간이 없으면 다음 디바이스에 새로운 세그먼트 파일을 생성하여 가용 공간을 늘려가므로, 여분의 디바이스를 지정해두는 것이 좋다.

1.1.3. 동작 구조

독크루저는 원본 데이터를 검색 데이터로 색인하여 볼륨에 저장한다. 사용자가 요청한 질의는 검색 방법에 적합하게 처리하여 검색한 후 검색 결과를 반환한다. 검색과 시스템 동작 내용은 로그로 생성된다. 부가서비스 모듈을 사용할 경우 서비스 제공 요청을 수용하여 서비스 결과를 반환한다.

관리자는 검색 데몬과 KQL 관리를 커맨드 방식의 인터프리터를 이용한 방식 또는 GUI 독크루저 관리기(KADM)를 이용하여 원격으로 수행할 수 있다. KADM은 'DOCRUZER Administrator Manual'을 참조한다.

다음은 독크루저의 동작 구조를 간략히 도식화한 것이다.



[그림 1.2] 독크루저 동작 구조

1.2. 특징

대용량 저장 기술

대용량 저장 기술은 온라인 검색 환경에서 점증하는 디지털 콘텐츠의 규모에 대처하기 위해 필수적인 기술이다. 문서의 규모가 수천만 또는 수억 건에 달하게 되면 일반적인 상용 RDBMS(Relational Database Management System)로는 검색 요구 조건을 실시간으로 처리하는 것이 어려우므로 고속의 저장 시스템이 탑재, 운용되어야 한다.

실시간 처리가 어렵다는 것은 첫째, 응답 시간과 둘째, 산출량 면에서 살펴볼 수 있다. 응답 시간은 사용자가 질의를 입력하여 결과를 받기까지의 시간을 말하며, 수천만 건의 문서와 인덱스를 하나의 DB에 저장하여 검색할 경우 질의 처리에 수십 초 또는 수 분의 시간이 경과하므로 검색 서비스의 제공이 사실상 어려워진다. 산출량은 하나의 서버가 단위 시간당 얼마나 많은 질의를 처리할 수 있는지를 의미하는 것으로, 검색 포털과 같은 대용량 검색 시스템의 경우 피크 타임 때에는 초당 수십 또는 많게는 수백 개

의 질의가 들어오므로 초당 하나씩 처리하더라도 항상 수백 대의 서버가 대기하고 있어야 한다는 얘기가 된다.

상용 RDBMS를 사용하는 경우 대용량 환경에서 발생하는 문제인 응답 시간과 산출량 저하의 문제를 해결하기 위한 경제성 있는 방법을 찾는 것은 당분간 어려울 것이다. 이러한 점을 고려할 때 RDBMS를 대체할 고속의 대용량 스토리지 시스템을 확보하는 것은 대용량 검색엔진의 구성에 필수적인 부분이라고 할 수 있다.

독크루저의 저장 시스템은 현재 볼륨(=단위 DB)당 12 TB, 20 억 문서를 설계 용량으로 하고 있다. 그러나 멀티볼륨을 지원하므로, 하나 이상의 볼륨을 연결하여 사용하면 얼마든지 저장 용량을 늘려나갈 수 있으므로 사실상 저장 용량의 제한이 없다고 볼 수 있다.

색인 압축

색인 압축은 대용량 콘텐츠의 효율적인 저장을 위한 필수적인 기능이다. 일반적인 검색엔진에서는 색인의 저장 공간이 원문 저장 공간의 150~300 % 정도 되는 것으로 알려져 있다. 만약 원문이 100 GB라면 인덱스 포함 250~400 GB가 된다는 것이다.

자연어 검색을 지원하려면 훨씬 더 많은 색인어가 필요하므로 산술적으로 보면 색인 저장 공간은 원문의 400~500 % 정도로 늘어나게 되며, 그렇게 되면 디스크 공간의 요구량이 원문에 비해 더욱 더 커지게 되고 이는 디스크 구매 비용으로 직결된다.

독크루저는 인덱스를 압축하여 저장함으로써, 일반 검색의 경우에는 원문의 50 %, 자연어 검색을 지원하는 경우에는 원문의 80~90 %의 저장 공간만으로 모든 인덱스를 압축하여 저장하기 때문에 디스크 비용을 크게 절감할 수 있다.

고속 인덱싱

고속 인덱싱은 서비스 콘텐츠의 신규성 확보 등 검색 서비스의 운영 면에서 볼 때 매우 중요한 요소 중의 하나이다. 현재 많은 검색엔진이 문서가 수천만 건 이상이 되면 인덱싱이 되지 않아서 검색 서비스에 이용할 수 없다. 그나마 인덱싱이 가능한 몇 되지 않는 검색엔진의 경우에도 검색 볼륨 구성을 위한 일괄 인덱싱 시간이 짧게는 1 주일에서 길게는 몇 주씩 걸리며, 또는 몇 개월이 걸리는 경우가 있으며 이러한 상황에서는 검색 콘텐츠의 신규성을 유지하는 것이 매우 어려운 일이다. 인덱싱 시간의 지연은 곧바로 검색 서비스의 신규 업데이트 지체 등 운용상의 문제점으로 이어지기 때문이다.

독크루저는 2천만 건의 웹 페이지를 하루에 1 대의 서버(4 x Intel Pentium-III 550 MHz)에서 인덱싱할 수 있는 현재 최고속의 인덱싱 속도를 자랑하는 엔진으로서 (2GB/HOUR),

다른 검색엔진에 비해 수 배~수십 배 향상된 인덱싱 성능을 보이므로 일괄 인덱싱을 통해 검색 콘텐츠를 준비하는 노력을 대폭 절감시켜 준다.

점진적 인덱싱

대량 문서의 고속 일괄 인덱싱 못지 않게 중요한 것이 추가로 발생하는 문서의 삽입/삭제/변경을 지원하는 일이다. 아무리 일괄 인덱싱이 빠르다 하더라도 하나의 문서를 삽입하기 위해 전체를 다시 인덱싱해야 한다면 운영 노력과 비용이 많이 소모될 수밖에 없다.

독크루저는 추가로 발생하는 문서의 삽입/삭제/변경 등 데이터의 점진적 변경을 지원하므로 운용상의 노력을 크게 절감하게 해준다.

온라인 업데이트

서비스 중인 검색 볼륨에서 문서를 삽입/삭제/변경해야 하는 일이 발생할 수 있다. 점진적 인덱싱이 지원되어도 온라인, 즉 동작 중인 상태에서 이러한 변경이 이루어지지 않는다면 역시 운용상의 어려움이 크게 된다. 단지 몇 개의 문서를 추가로 삽입하기 위해 동작 중인 검색 서버를 오프라인으로 만들어 사용자의 질의를 차단하고, 문서를 추가한 다음 다시 온라인으로 만들어야 한다면 운용상의 오버헤드가 클 수밖에 없다.

독크루저는 동작 중인 검색 서버의 검색 콘텐츠를 온라인(비중단) 상태에서 일/분/초 단위로 변경할 수 있기 때문에 운용상의 오버헤드가 크게 절감된다.

고속 검색

고속 검색은 대용량 콘텐츠의 온라인 서비스를 위해 필수적인 요건이다. 응답시간을 최소화함으로써 서비스에 대한 사용자의 만족도를 높일 수 있으며, 검색 요청 한 건당 빠른 처리는 서버당 처리량을 증가시켜서 더 적은 수의 서버로 서비스를 가능하게 한다. 이는 하드웨어 비용의 절감 효과를 가져온다.

독크루저는 소규모 데이터(웹 페이지 10만 문서)에서는 피크 타임 기준 0.001 초의 응답시간과 약 700~800 query/sec에 달하는 처리 성능을 갖는다(4 x Pentium-III 550 MHz 서버 1 대 기준). 이는 중급 성능의 서버 1 대로도 피크 타임 기준 일일 6천만 건의 질의어를 처리할 수 있는 획기적인 성능이며, 결과적으로 하드웨어 비용의 대폭적인 절감을 가능하게 한다.

멀티스레드 API

독크루저는 최적화된 서버 자원 이용을 목적으로 하는 멀티스레드 구조로 설계 및 구현되었고, 멀티스레드 안전성(multi-thread safeness)을 보장하는 API를 지원함으로써 맞춤 작업을 용이하게 한다.

멀티볼륨

독크루저는 콘텐츠를 논리적/물리적으로 분리된 볼륨의 형태로 저장하여 엔진에 자유자재로 붙이거나 떼는 것을 가능하게 한다. 분리하여 관리하고 싶은 데이터를 별개의 물리적 단위로 관리할 수 있기 때문에 서로 다른 시스템에 저장된 문서들의 통합 검색을 구현할 때 유용하다. (예: 검색 ASP)

고성능 언어 분석 기술

고성능 언어 분석 기술은 검색의 정확도 및 성공률을 높이기 위해 필수적인 기술이다. 독크루저는 자연어 분석 기술(Natural Language Analysis Technology) 방식을 채용한 색인어/구의 추출을 통해 문서의 색인 및 검색을 수행함으로써, 가장 정확한 검색이 가능하게 한다.

다국어 검색

다국어(한국어, 영어, 중국어, 일본어) 검색은 국제화된 검색 서비스의 구현과 국내 중심 검색 서비스의 구현에 모두 필수적인 기술이다. 독크루저는 한국어는 물론 영어, 중국어, 일본어에 대한 색인/검색 기술과 사전을 탑재하고 있으므로 각각의 언어에 대한 검색 서비스를 제공할 수 있다.

자연어 검색

독크루저는 고품질 자연어 분석 기술에 기반하여 사용자가 문장 형태의 질의를 입력하고 검색할 수 있도록 지원한다. 이는 사용자가 원하는 것을 검색하기 위해 검색 범위나 검색 키워드 제약, 입력 방식의 제약과 같은 형식의 제약 없이 있는 그대로 '검색 정보'를 표현할 수 있게 함을 의미한다. 자연어 검색은 강력해진 사용자 중심의 검색 서비스를 제공할 수 있다.

로캘 지원

독크루저는 KSC5601, GB2312, SJIS, JIS, Big5, Unicode 등 한국어, 중국어, 일본어에 관한 사실상 모든 표준 문자 세트를 지원한다.

띄어쓰기 오류 수정

웹 문서와 같은 일반 비정형 텍스트를 검색 대상으로 하는 경우, 문서 원문이나 질의어에 띄어쓰기가 잘못된 경우가 많아 검색에 실패하는 일이 많다. 비정형 텍스트를 대상으로 하는 정보검색 시스템의 색인 품질을 높이기 위해서는 띄어쓰기 오류 수정과 같은 고급 기능이 필수적이다.

독크루저의 한국어 분석 컴포넌트는 띄어쓰기 오류 수정 기능이 있으므로 원문이나 질의어의 띄어쓰기가 잘못되었어도 대부분 색인과 검색을 올바르게 수행한다.

오타 교정

키보드의 한/영 키가 잘못 눌려진 채 질의어를 입력하는 경우 자동으로 한/영, 영/한 오타를 교정하여 올바른 질의어로 검색한다.

질의어 추천

모든 사용자가 자신이 원하는 정보를 찾기 위해 언제나 알맞은 질의어를 입력할 수 있는 것은 아니다. 이러한 경우에는 원래 질의어와 연관된 질의 표현을 추천함으로써 사용자로 하여금 정보를 선택하게 할 수 있다.

독크루저는 질의어 추천 기능을 제공하여 입력한 질의어와 의미적 연관성이 있는 용어들을 추천하여 사용자가 원하는 정보를 찾아낼 수 있는 가이드를 제공한다.

복합명사 분해 색인

한국어, 중국어, 일본어는 복합어 표현이 많기 때문에 띄어서 쓴 경우와 붙여서 쓴 경우 키워드가 달라지게 된다. 복합어 분해를 하지 않으면 원하는 정보가 있는데도 제대로 찾아내지 못하게 된다. 독크루저는 강력한 복합명사 분해 모듈에 의해 복합명사 표현을 분해하여 저장하므로 사용자가 원하는 정보를 가진 문서를 최대한 많이 가져온다.

대용량 사전

독크루저는 한국어 280만 어휘, 일본어 73만 어휘, 중국어 57만 어휘 등 대용량 사전을 바탕으로 한 강력한 언어분석 모듈을 탑재하고 있다.

부하 분산 및 멀티 서버 구성

검색 부하가 적을 때는 검색 서버의 수를 적게 쓰고 부하가 많아지면 서버의 수를 늘림으로써 최적의 하드웨어 설비로 서비스를 충족시키는 것이 비용 효과적이다. 이를 위

해 검색엔진은 멀티 서버를 지원해야 한다. 독크루저는 여러 대의 서버에 검색엔진을 설치하고 부하를 분산시킨다. 이를 통해 특정 서버가 병목되지 않고 주어진 하드웨어를 최대한 활용하여 시간적 지연 없이 서비스를 제공할 수 있게 한다.

점진적 확장성

검색 부하가 늘어났을 때 기존에 동작하고 있던 서버는 그대로 두고 새로운 서버를 추가하여 처리 능력을 확장하는 점진적 확장성을 지원한다.

외부 문서 수입 게이트웨이

검색 원문은 검색엔진의 외부에 DBMS나 파일 시스템 등과 같은 여러 가지 다른 저장 매체에 저장되어 있다. 빠른 검색을 수행하기 위해서는 다양한 저장 매체에 분산된 원문을 가져와 검색엔진의 레코드로 변환하는 과정을 거쳐야 한다.

독크루저는 게이트웨이 모듈을 탑재하여 외부 저장 매체로부터 손쉽게 문서를 가져올 수 있다.

SDK 지원

독크루저는 잘 정의된 C API를 제공함으로써 검색엔진을 탑재한 다른 제품군이나 고객의 다양한 서비스 요구 및 응용 환경에 대해 맞춤 서비스를 구현할 수 있다. 개발 환경에 따라 C, Java, ASP, PHP, C# API를 선택하여 사용할 수 있다.

비용의 최적화

상용 서비스 구축 시에 중요한 것은 비용의 최적화를 달성했느냐는 점이다. 검색 솔루션이 어느 정도의 처리량을 달성했느냐 하는 것은 설비 비용으로 직결된다. 처리량이 3 배 증가하였을 경우 서버의 수를 1/3로 줄일 수 있기 때문이다. 이 경우, 필요한 서버의 수가 감소할수록 비용이 절감되므로 설비 비용의 최적화는 검색 솔루션 자체의 비용보다도 사업비 절감에 기여하는 바가 클 수 있다.

독크루저는 응답 시간과 처리량을 획기적으로 개선함으로써 다른 엔진에 비해 훨씬 적은 하드웨어만을 사용하여 비용의 최적화할 수 있다.

제품 설치 및 제거

이 장에서는 독크루저의 제품 등급에 따른 제품 상세 옵션과 구성 요소에 대해 알아보고, 독크루저를 설치하고 제품 키를 등록하는 방법에 대해 설명한다.

2.1. 설치 구성 요소

2.1.1. 패키지 구성

독크루저의 패키지 구성은 다음과 같다.

- 제품 CD
- 제품 키

2.1.2. 제품 등급 구분

독크루저의 제품 등급은 아래 표와 같이 구분한다. 제품은 활용할 CPU의 수와 최대 저장 크기로 등급을 구분한다.

구분	최대 지원 CPU	최대 저장 용량
DOCRUZER 8C UL	8	Unlimited
DOCRUZER 8C 25G	8	25 GB
DOCRUZER 8C 20G	8	20 GB
DOCRUZER 8C 10G	4	10 GB
DOCRUZER 4C UL	4	Unlimited
DOCRUZER 4C 20G	4	20 GB
DOCRUZER 4C 10G	4	10 GB
DOCRUZER 4C 7G	4	7 GB
DOCRUZER 4C 5G	4	5 GB
DOCRUZER 4C 3G	4	3 GB

구분	최대 지원 CPU	최대 저장 용량
DOCRUZER 4C 1G	4	1 GB
DOCRUZER 2C UL	2	Unlimited
DOCRUZER 2C 20G	2	20 GB
DOCRUZER 2C 10G	2	10 GB
DOCRUZER 2C 7G	2	7 GB
DOCRUZER 2C 5G	2	5 GB
DOCRUZER 2C 3G	2	3 GB
DOCRUZER 2C 1G	2	1 GB
DOCRUZER 1C 5G	1	5 GB
DOCRUZER 1C 3G	1	3 GB
DOCRUZER 1C 1G	1	1 GB
DOCRUZER 1C 500M	1	500 MB

2.1.3. 제품 구성

독크루저의 제품 구성 요소는 다음과 같다.

구성 요소	설명
KQL	명령어 인터프리터, DBMS 관리 도구
DOCRUZERD	검색엔진 독크루저 데몬
LIBRARY	KQL 클라이언트 툴킷, 독크루저 클라이언트 툴킷
OPTION PACK	기능 확장을 위한 옵션 모듈 모음
UTILITIES	유용한 유틸리티 모음

2.2. 시스템 요구사항

2.2.1. 지원 플랫폼 및 운영체제

독크루저의 지원 플랫폼 및 운영체제는 다음과 같다.

H/W, S/W 벤더	CPU	OS	Binary Bits
Microsoft	x86, x86_64, IA64, AMD64	2000/XP/2003/2008 등 WinNT 계열	32bit/64bit
GNU/Linux	x86, x86_64, IA64, AMD64	Linux2.4 이상	32bit/64bit
UNIX	Sparc, PPC, PPC64, MIPS, Alpha	Solaris, AIX, HP-UX, Tru64, Irix, FreeBSD	32bit/64bit

2.2.2. H/W 및 S/W 요구사항

독크루저를 설치하기 위해 필요한 H/W 요구사항은 사이트 요구사항에 따라 다르다.

독크루저를 설치하기 위해 필요한 S/W 요구사항은 다음과 같다.

항목	규격
Windows	2000/XP/2003/2008 등 WinNT 계열 추천
GNU/Linux	gcc2.95/gcc3.x 지원
UNIX	특정 사이트 환경에 따라 필요 프로그램 설치

2.3. 설치 절차

2.3.1. 설치 준비

시스템에 독크루저를 설치하려면 다음의 패키지 파일 및 설치 실행 파일, 제품 키(product key)가 필요하다.

- 독크루저 패키지

  'DOCRUZER-버전정보-OS정보-CPU정보-32/64비트정보-날짜_r리비전.kpm'과 같은 형식의 이름으로 되어있으며, 독크루저 구동에 필요한 기본 파일 및 각종 데이터, 사전, 라이브러리, 샘플 등을 모두 포함하고 있다.

  (예: DOCRUZER-3.x.x-linux_rh-i686-32-20100730_r11277.kpm)

- 패키지 매니저

독크루저 패키지 파일을 설치하기 위한 실행 파일

파일명: kpm (Windows는 kpm.exe)

- 제품 키

2.3.2. 설치

Linux에서 독크루저를 설치하는 절차는 다음과 같다.

1. 독크루저 시디롬에서 설치하기 위해서는 설치할 시스템의 대상 디렉터리에 '설치 준비'에 있는 독크루저 패키지, 패키지 매니저 파일을 복사한다.

   다음과 같이 대상 디렉터리에 패키지 파일 및 설치 실행 파일이 있는지 확인한다.

```
$ ls
DOCRUZER-3.x.x-linux_rh-i686-32-20100730_r11277.kpm
kpm
```

2. 패키지 매니저를 이용하여 독크루저 패키지를 설치하면 자동으로 설치가 완료된다. 이때 제품 키가 필요하다.

```
$ ./kpm -isv DOCRUZER-3.x.x-linux_rh-i686-32-20100730_r11277.kpm

 ----------------------------------------
       konan package manager install
 ----------------------------------------

  extract 'bin' ..
  extract 'bin/kql' ..
  extract 'bin/docruzerd' ..
...
 ----------------------------------------
       konan package manager setup
 ----------------------------------------

 product key : XXXXXX-XXXXXX-XXXXXX-XXXXXXX-XXXXXX
  extract 'bin/docruzerd.rc' ..
  extract 'bin/gateway.rc' ..
...
```

```
setup 'DOCRUZER-3.x.x-linux_rh-i686-32-20100730_r11277.kpm'
.. done
```

참고

제품 키를 발급받기 위해서는 시스템에 설치된 네트워크 인터페이스의 MAC Address가 필요하다. MAC Address는 패키지 매니저의 --synfo 옵션으로 알 수 있다.

```
$ ./kpm --synfo
   Host ID. . . : XX-XX-XX-XX-XX-XX

   IP Address . : 127.0.0.1
                  192.168.0.11

   CPU. . . . . : 1988MHz GenuineIntel Intel(R) Xeon(TM) MP
                  CPU 2.00GHz
                  1988MHz GenuineIntel Intel(R) Xeon(TM) MP
                  CPU 2.00GHz
                  1988MHz GenuineIntel Intel(R) Xeon(TM) MP
                  CPU 2.00GHz
                  1988MHz GenuineIntel Intel(R) Xeon(TM) MP
                  CPU 2.00GHz

   Memory . . . : Total(7993 MB) Cached(6778 MB) Free(565 MB)

   OS . . . . . : Linux (release 2.6.9-55.ELsmp)
```

Windows의 경우에도 설치 절차는 동일하다.

2.4. 설치 확인

2.4.1. 디렉터리 구조 확인

독크루저가 설치되면 다음과 같은 디렉터리가 생성된다.

```
docruzer/
     |-- bin
     |-- data
     |-- lib
     |-- sdk
     |   |--ASP
     |   |--C
     |   |--Delphi
     |   |--Java
     |   |--PB
     |   |--PHP
     |   `--VB
     |-- ext
     |-- cxfile
     |-- dictionary
     |-- logs
     |   |--lgm
     |   |--gateway
     |   |--system
     |   |--search
     |   `--kql
     |-- rawdata
     |-- conf
     |-- scenario
     |-- script
     `-- volume
         |--sample
         `--temp
```

다음은 각 디렉터리에 대한 설명이다.

bin

실행 파일 및 docruzerd.rc, kql.rc, gateway.rc 위치

conf

설정 파일 저장 위치

cxfile

형식 파일 저장 위치

data

모듈 및 관련 파일 저장 위치

dictionary

사용자 사전 저장 위치

lib

라이브러리 파일 저장 위치

logs

독크루저가 생성하는 로그 저장 위치

rawdata

검색 샘플 데이터

scenario

시나리오 저장 위치

script

kql 스크립트 저장 위치

sdk

독크루저 API 라이브러리와 샘플 코드

volume

볼륨(검색 데이터) 저장 위치

다음은 각종 로그파일에 대한 설명이다.

kql.log

kql 로그는 색인의 진행을 포함하여 KQL의 거의 모든 동작을 기록하는 검색엔진 기본 로그이다. 에러발생 여부는 이곳에서 확인할 수 있다. kql.rc에서 'sys_log'로 위치 설정 가능하다.

docruzerd.log

독크루저 로그는 검색 서비스 데몬인 docruzerd의 동작을 기록한 로그이다. 단순한 start, stop 관련한 로그로 docruzerd 파일이 있는 디렉터리에 남는다.

yyyymmdd-nn.msg

메시지 파일은 검색 데몬이 남기는 데몬 동작 로그 파일이다. 검색 서비스 장애 시에 어떤 에러가 발생했는가를 확인할 수 있다. docruzerd.rc의 'sys_log_file_location'으로 위치 설정 가능하다. 'nn'은 두 자리 일련번호로 '00'부터 시작한다.

yyyymmdd-nn.log

검색 로그는 검색 질의어를 기록하는 로그이다. 날짜별로 남는 로그이며 설정에 따라 질의어, 처리속도, 패킷 크기, 클라이언트 IP, 캐시 검색 여부 등을 남길 수 있다. 검색 로그는 사용자 질의에 대한 통계 기초자료가 되기도 하며 'replay'라는 유틸리티를 사용하여 재검색해볼 수 있는 데이터가 된다.

로그마이너는 검색 로그를 분석하여 검색 통계를 자동으로 생성한다. docruzerd.rc의 'log_file_location'으로 로그 생성 위치를 설정한다.

2.5. 제품 키

독크루저를 설치한 후, 독크루저를 작동하려면 해당 서버용 제품 키(product key)가 필요하다. 제품 키를 발급받은 후, 시스템 설정 파일(kql.rc)에서 설정한다.

올바른 제품 키가 아닌 경우 서버에서 다음과 같은 오류가 발생하고 기동되지 않는다.

```
$ docruzerd start
Error: 'KQLC_Initialize() unauthorized copy (invalid IP address
        or Host ID).'$
```

2.6. 볼륨 설정 파일(vol.rc)

볼륨 설정 파일은 독크루저가 검색 데이터를 보관, 관리하는 공간인 볼륨의 위치 정보를 설정하는 파일이다. 독크루저 볼륨은 운영체제가 관리하는 파일 시스템을 이용한다. 독크루저는 멀티볼륨을 지원하고 각 볼륨은 논리적, 물리적으로 분리되어 있기 때문에

각 볼륨의 데이터 위치는 서로 다르게 지정할 수 있다. 즉, 멀티볼륨을 사용하는 것은 다수의 디렉터리를 볼륨 설정 파일에 등록하여 검색 데이터를 저장, 관리함을 의미한다. 사용하는 볼륨의 수만큼 볼륨 설정 파일을 작성하여, 'KQL 설정 파일 (kql.rc)'에 등록한다.

- 작성 형식

  볼륨 설정 파일에는 볼륨의 세그먼트 파일들이 위치할 디바이스 경로, 로그 디바이스, 임시 디바이스의 경로를 지정한다. 이 파일은 파라미터와 그 값의 쌍으로 이루어진다.

- 주석

  /* */ 또는 ';'를 사용한다.

- 기본값

  기본값은 없다.

각 파라미터에 대한 설명은 다음과 같다.

파라미터	설명

volume_device	볼륨 디바이스(volume_device)는 볼륨 데이터 파일이 위치한 디바이스를 지정하는 파라미터이며, 파일 시스템 상의 볼륨 디렉터리 경로를 값으로 갖는다. 만약 한 볼륨이 매우 크거나 한 디바이스의 가용 용량을 초과할 것으로 예상되는 경우에는 여러 디바이스를 차례로 등록하면 된다. 필수 파라미터이며 기본값은 없다.
	
	예)
	
	`volume_device = "/home/my_vol_dev0"`
	`volume_device = "/home/my_vol_dev1"`
	`volume_device = "/home/my_vol_dev2"`

log_device	로그 디바이스(log_device)는 시스템이 로깅에 이용할 디바이스를 지정하는 파라미터이며, 파일 시스템 상의 디렉터리 경로를 값으로 갖는다. 만약 여러 디바이스를 이용하고자 한다면 볼륨

파라미터	설명

	디바이스 파라미터와 마찬가지로 각각의 디바이스를 차례로 순차적으로 등록하면 된다. 필수 파라미터이며 기본값은 없다.
	
	예)
	
	`log_device = "/home/my_log_dev0"`
	`log_device = "/home/my_log_dev1"`

temp_device	임시 장치(temp_device)는 실행 중에 사용하는 임시 디스크 공간을 지정하는 파라미터이며, 파일 시스템 상의 디렉터리 경로를 값으로 갖는다. 볼륨 디바이스 파라미터와 마찬가지로 여러 디바이스를 차례로 순차적으로 등록할 수 있다. 기본값은 없다. 예) `temp_device = "/home/my_temp_dev0"`

2.7. KQL 설정 파일 (kql.rc)

'KQL 설정 파일(KQL configuration file)'은 독크루저가 사용하는 자원 한도, 접근 제어, 리스너 주소, 스케줄 등 시스템 자원 접근 및 제어에 관한 세부사항이 기록되어 있는 파일이다. 볼륨 파일이 각 볼륨마다 고유하게 존재하는 것과는 달리, KQL 설정 파일은 독크루저 전체의 자원에 대한 명세이므로 하나만 사용한다.

- 작성 형식
  - KQL 설정은 설정 변수와 그 값의 쌍으로 이루어지며, 한 줄당 '설정 변수명 = 설정값'으로 작성한다.
  - 디렉터리 경로를 설정값으로 가지는 경우 절대 경로, 상대 경로 형태 모두 사용 가능하다. 단, 상대 경로의 기준 위치는 bin 디렉터리이다.
  - 'y' 또는 'n' 설정 값은 대소문자를 구분하지 않으며, 'yes' 또는 'no'로 설정할 수도 있다.

- 주석

  /* */ 또는 ';'를 사용한다.

- 기본값

  일부 설정 변수는 생략가능하며 생략된 설정 변수는 기본값으로 실행된다.

2.7.1. 기본 설정

KQL 기본 설정 파라미터에 대한 설명은 다음과 같다.

파라미터	설명
data_location (데이터 위치)	독크루저에서 이용하는 각종 데이터 파일이 존재하는 디렉터리의 경로를 설정한다. (기본값="."). 예) `data_location = "/home/my_data"`

listener
(리스너)
	원격 클라이언트 요청을 수신하는 스레드의 통신 정보를 설정한다.
	설정 변수값은 서버의 네트워크 주소와 포트의 쌍이다. 서버 주소값은 도메인 이름이나 IP 주소가 모두 될 수 있다. 하나 이상의 리스너를 등록할 수 있다. 네트워크 카드가 2 개 이상인 호스트 서버에서는 서로 다른 IP 주소와 포트를 이용하여 리스너를 등록할 수도 있다. (기본값=로컬 호스트, 6176 포트).
	예)
	`listener = 211.63.24.25, 6177`
	`listener = myhost.mycompany.com, 6178`

max_buffer_cache_size (최대 버퍼 캐시 크기)	독크루저 내에서 디스크 페이지를 빠르게 접근하기 위해 사용할 캐시 버퍼의 최대 크기를 설정한다. 디스크 볼륨의 크기에 따라 다르지만 일반적으로 이 값이 클수록 페이지 접근 성능은 향상된다. (기본값=128 KB).

파라미터	설명
	예) `max_buffer_cache_size = 1024 MB`
max_no_of_connections (최대 연결 수)	리스너를 통하여 독크루저에 동시 연결 가능한 최대 접속 개수를 설정한다. 즉, 동시에 생성되는 최대 세션 수이다. 세션을 연결하는 것은 독크루저에서 동시 실행하는 태스크를 배정하는 것을 의미한다. 하나의 스레드에서 하나 또는 하나 이상의 세션을 설정하는 것은 관계없지만, 2개 이상의 스레드가 하나의 세션을 공유해서는 안된다. (기본값=16). 예) `max_no_of_connections = 20 (정수값)`
max_no_of_open_files (최대 파일 수)	동시에 오픈할 수 있는 파일의 최대 수를 설정한다. 운영체제가 허용하는 최대 파일 수를 초과할 수 없다. (기본값=128). 예) `max_no_of_open_files = 10 (정수값)`
max_no_of_table_handles (최대 테이블 핸들 수)	동시에 오픈할 수 있는 테이블의 최대 수를 설정한다. 테이블 핸들은 테이블을 열 때마다 하나씩 할당된다. (기본값=64). 예) `max_no_of_table_handles = 50 (정수값)`
max_no_of_volume_handles (최대 볼륨 핸들 수)	동시에 오픈할 수 있는 볼륨의 최대 수를 설정한다. 볼륨 핸들은 볼륨을 열 때마다 하나씩 할당된다. (기본값=32). 예) `max_no_of_volume_handles = 64 (정수값)`

파라미터	설명
max_working_memory_size (작업 메모리 크기)	임시 작업용 메모리의 최대 크기를 설정한다. 작업 메모리는 색인이나 검색 작업 중간에 소요되는 임시 메모리 공간을 의미하며 처리 속도에 많은 영향을 주므로 충분한 크기를 지정해 주는 것이 좋다. (기본값=128 KB). 예) `max_working_memory_size = 512MB`
no_of_parallel_task (병렬 태스크 수)	독크루저가 내부적으로 배치 작업을 수행할 때 사용하는 프로세스의 병렬도를 설정한다. 설치할 독크루저의 CPU 수와 같은 값으로 설정한다. 하드웨어가 가진 CPU 수보다 큰 값을 가지면 안된다. 현재는 색인 시 병렬성을 결정하며 검색 스레드와는 무관하다. (기본값=1). 예) `no_of_parallel_task = 4`

product_key
 	
(제품 키)
	제품 키를 등록한다.
	
	설치될 시스템에 유효한 제품 키를 등록해야 독크루저를 실행할 수 있다.
	
	예)
	
	`product_key`
	`= ETJTKCB-6XNT1RB-M1V10R-NM4COFB`

schedule (스케줄)	스케줄(schedule)은 정해진 시간에 실행할 작업을 설정한다. 스케줄 내용은 ';'로 끝나는 KQL 명령어들의 연속으로 표현되며, 큰따옴표(" ")로 둘러싼 스트링형의 값으로 주어져야 한다. 스케줄 내용에 ","가 들어가면 Escape 문자 '\'를 선행시켜야 한다 ('\' 문자는 '\\'로 표현해야 함). 1 개 이상의 스케줄을 등록할 때는 schedule 변수를 추가로 설정한다. 예)

파라미터	설명
	schedule = sch1, "run batch.kql;", every 100 sec but 23:00 – 07:00 ("FAILED" "run coly.kql;")
sys_log	로그 메시지를 표시할 디바이스를 설정한다. 로그 메시지를 콘솔에 표시하려면 console, 파일에 저장하려면 로그 파일 경로를 지정한다. sys_log 값을 지정하지 않으면 로그 메시지가 출력되지 않는다. 예) sys_log = ../logs/kql/kql.log
temp	엔진에서 사용하는 임시 파일(kql#####.tmp)이 생성될 디렉터리를 지정한다. 지정하지 않으면 디폴트로 현재 디렉터리('.')가 사용된다. 디렉터리는 미리 만들어져 있어야 한다. 예) temp = ./tmp
volume (볼륨)	독크루저가 관리할 볼륨을 설정한다. 볼륨의 설정값은 볼륨의 이름과 '볼륨 설정 파일(vol.rc)'에서 설명한 볼륨 설정 파일을 이용한다. 지정한 볼륨명을 이용하여 물리적 디바이스에 존재하는 검색 데이터에 접근한다. 볼륨이 여러 개일 경우, 해당 볼륨의 설정 파일을 이용하여 설정 변수를 추가한다. 예) volume = my_volume1, /home/my_data/my_volume1.rc volume = my_volume2, /home/my_data/my_volume2.rc 볼륨 이름을 변경하려면 설정 파일(kql.rc)에서만 변경해 주면 된다.

파라미터	설명
working_memory_block_size	메모리 할당 시 연속된 메모리 공간의 단위를 설정한다. 성능면에서는 이 값을 'max_working_memory_size / working_memory_block_size >= CPU 수'를 만족하도록 지정해 주는 것이 좋다. (기본값=64 KB). 예) `working_memory_block_size = 64 MB`

볼륨이나 테이블, 필드명은 영문 대/소문자를 구분하며 특수기호는 사용할 수 없다. 단, 언더바(_)는 사용 가능하다.

다음 예와 같은 특수기호들은 볼륨이나 테이블, 필드명에 사용할 수 없다.

```
[ ] ( ) { } < > -
+ / = * ; : " ` '
. , ~ ! @ # $ % ^
& \ | ?
```

2.7.2. 세부 설정

KQL 세부 설정 파라미터에 대한 설명은 다음과 같다.

파라미터	설명
allow_dirty_read	볼륨에 대하여 dirty read를 허용할지 설정한다. (기본값=y). 예) `allow_dirty_read = n`
allow_null_query_key	사용자 질의 분석 단계에서 'null query key'가 추출되었을 때 이를 허용할지 말지 여부를 설정한다. (기본값=N). 예) `allow_null_query_key = Y`

파라미터	설명
asterisk_expansion_limit	(성능상의 이유로) 절단 검색의 키 확장 제한 값을 설정하고자 할 때 사용한다. (기본값=0). 예) `asterisk_expansion_limit = 5`
attach_wordph_document_universal	색인 문서에 대하여 ngram level 값보다 작은 어절만 어절색인어로 추출한다. 어절색인어는 스페셜 마크(0xFFFF)가 부착된 형태로 제공된다. (기본값=y). 예) `attach_wordph_document_universal = n`
attach_wordph_query_universal	쿼리문에 대해 ngram level 값보다 작은 어절만 어절키워드로 추출한다. 어절키워드는 스페셜 마크(0xFFFF)가 부착된 형태로 제공된다. (기본값=y). 예) `attach_wordph_query_universal = n`
charset	EUCKR, EUCCN, BIG5, EUCJP, SJIS, UTF8, USASCII, USER0~USER9 등 시스템의 기본 문자 세트를 설정한다. 생략하면 'EUCKR'로 설정된다. 자세한 값은 '언어와 문자 세트'를 참고한다. (기본값=EUCKR). 예) `charset = big5`
compress_volume_patch	볼륨 패치의 압축 여부를 설정한다. (기본값=n). 예) `compress_volume_patch = y`

파라미터	설명
create_max_sized_segments	세그먼트별로 생성되는 볼륨 파일이 생성될 때 세그먼트 최대 크기의 파일을 생성할지 검색 레코드가 추가될 때마다 세그먼트의 크기를 증가시킬지 여부를 설정한다. ● y: 최대 크기의 세그먼트 파일이 한번에 생성되므로 검색 레코드가 추가되어도 필요 디스크 공간 할당에 소요되는 운영체제 오버헤드가 발생하지 않는다. ● n: 세그먼트 크기는 검색 레코드가 추가될 때마다 증가한다. 단, 세그먼트 허용 크기가 넘으면 새로운 세그먼트가 생성된다. (기본값=n). 예) `create_max_sized_segments = y`
create_volume_by_force	기존 볼륨이 있는 경우 볼륨 생성(create volume) 시 덮어쓰기 여부를 설정한다. (기본값=Y). 볼륨 생성 시스템 명령어로 명령이 성공하면 볼륨에 존재하던 기존 검색 데이터는 모두 삭제된다. ● y: 기존 볼륨이 있어도 'create volume' 시 덮어쓴다. ● n: 기존 볼륨 확인 후 덮어쓴다. 예) `create_volume_by_force = n`
data_validation_level	색인 시, 필드 데이터 적합성 테스트 옵션을 설정한다. 원본 데이터를 검색 데이터로 변환하는 과정에서, 설정된 레코드의 필드 타입과 데이터 값이 맞지 않아 문제가 있는 경우, 색인을 중단할지 무시하고 진행할지 아니면 기본값으로 대체하고 진행할지 설정하는 파라미터이다. (기본값=0).

파라미터	설명
	● 0: 'value not valid at field' 에러 메시지를 표시하고 색인 전체를 중단한다. ● 1: 경고 메시지("Warning: invalid data at field ignored")를 kql.log에 기록하고 색인을 계속 진행한다. ● 2: 문제의 필드값을 기본 필드값으로 변경하여 색인을 계속 진행한다. 이때 기본 필드값은 필드 속성에 따라 빈 문자열(" ")이거나 '0'이 된다. (단, 검색 서버에 설정 시 의도하지 않는 결과가 나올 수 있으므로 검색 서버에는 설정하지 않도록 해야한다.) 예) `data_validation_level = 2`
database_connection_agent	와처 에이전트를 설정한다. 설정값은 데이터베이스명, 포트, 로그 파일의 순서로 지정한다. 데이터베이스명은 oracle, mssql, sybase, db2, mysql, exchange, cubrid 중에서 지정 가능하다. 지정하지 않으면 와처 에이전트 대신 와처를 사용하게 된다. 와처 또는 와처 에이전트가 필요한 경우 가능하면 와처 에이전트를 설정하여 사용한다. 와처 에이전트, 와처에 대한 설명은 'DOCRUZER Gateway Manual'을 참고한다. 예) `database_connection_agent=oracle, 7100, ../logs/oracle.log`
default_retrieval_method	검색 방법을 생략한 경우의 기본 검색 방법(boolean 또는 exact)을 설정한다. (기본값=exact). ● exact: 정확하게 질의문과 일치하는 키 값을 찾음.

파라미터	설명
	예) `default_retrieval_method = boolean`
disk_wait_time	디스크에서 read, seek, write 값이 0일 경우 재시도하며 기다리는 시간(sec)을 설정한다. (기본값=5). 예) `disk_wait_time = 10`
document_compound_level	문서 복합명사 분해 레벨(1~5)을 설정한다. 복합명사 분해 레벨에 대해서는 '키워드 추출 (EXTRACT KEYWORDS)'을 참고한다. (기본값=2). 예) `document_compound_level = 5`
enable_raw_field_sort	독크루저에서는 기본적으로 정렬에 사용하는 필드에 색인을 걸어주어야 한다. 3.5.x 이하 버전에서는 색인이 걸려 있지 않아도 (느리긴 하지만) 정렬이 가능하나, 3.6.0 이상에서는 명시적으로 'C34568' 에러를 내게 바뀌었다. 하위 호환성을 위해서 이 설정 값을 'y'로 하는 경우 3.5.x 처럼 동작하는 기능이 추가됐다. (기본값=n). 예) `enable_raw_field_sort = y`
file_server	리모트 파일 서버를 지정한다. (IP:port 형식). 예) `file_server = 192.168.1.245:6186`
ignore_null_keyword	대용량 색인 단계에서 (시스템 오류든 프로그램 오류든) 비정상적으로 '\0'가 키워드 중간에 끼어들어가 null keyword 오류를 내며 색인이 중단되는 경우 이를 무시하고 계속 색인할지 설정한다. (기본값=n).

파라미터	설명
	예) `ignore_null_keyword = y`
ignore_post_corruption	포스트데이터의 무결성이 깨지는 경우의 무시 여부를 설정한다. 포스트데이터의 무결성은 볼륨 세그먼트의 데이터와 캐시 사이의 불일치 시 깨지는 현상이 발생한다. (기본값=n). ● y: 검색 서비스가 종료되지 않도록 한다. 예) `ignore_post_corruption = y`
ignore_space_insensitive_keyword_english_universal	space insensitive keyword 계열의 옵션이 설정되어 있더라도 영문의 경우에는 무시할지 설정한다. (기본값=y). 예) `ignore_space_insensitive_keyword_english_universal = n`
invalidate_all_buffers_of_copied_volume	증분복사 시 볼륨버퍼 전체 무효화를 할지 설정한다. (기본값=n). 예) `invalidate_all_buffers_of_copied_volume = y`
key_cache_only_group_by	group by 연산 시 key cache만을 사용하여 group by를 처리하도록 한다. 만약 group by 처리하려는 필드의 key cache가 없으면 기존 처리방식으로 동작하게 된다. (기본값=n). 예) `key_cache_only_group_by = y`

파라미터	설명
keyword_case	영단어 키워드의 대소문자 통일 여부를 지정한다. (기본값=lower). ● lower: 모두 소문자로 변환하여 색인/검색한다. (기본값). ● mixed: 대소문자를 변환하지 않고 그대로 색인/검색한다. ● upper: 모두 대문자로 변환하여 색인/검색한다. 예) `keyword_case = mixed`
keyword_charset	키워드를 추출하는 문자 세트를 설정한다. 예) `keyword_charset = utf8`
language	KOREAN, CHINESE, JAPANESE, ENGLISH, UNIVERSAL, USER0~USER9 등의 시스템 기본 언어를 지정한다. (기본값=KOREAN). 자세한 값은 '언어와 문자 세트'를 참고한다. 예) `language = chinese`
language_alias	시스템 색인기(한국어, 중국어, 일본어, 영어, 유럽어)를 사용자정의 색인기(USER0~USER9)로 지정할때 사용한다. 예) `language_alias = korean`

파라미터	설명
location_overflow_check_limit	location overflow 경고 메시지를 내기위한 overflow 최소 횟수를 설정한다. (기본값=2). 예) `location_overflow_check_limit = 3`
longest_synonym_expansion	최장 일치 어절 동의어 확장 방식을 사용할지 설정한다. (기본값=y). 예) `longest_synonym_expansion = n`
matchfield_aggregate_weight	matchfield 정렬 시 점수 합산 방식을 사용할지 설정한다. 이 설정이 n일 경우 큰 점수로 덮어쓰는 방식을 사용한다. (기본값=y). 예) `matchfield_aggregate_weight = n`
max_docsize_universal	색인 대상 문서의 최대 크기를 설정한다. (기본값=10240). 예) `max_docsize_universal = 20480`
max_file_block_count_in_volume_copy	볼륨 복사 시 사용할 파일 블럭의 최대 개수를 설정한다. (기본값=512). 예) `max_file_block_count_in_volume_copy = 1024`
max_memory_block_count_in_volume_copy	볼륨 복사 시 사용할 메모리 블럭의 최대 개수를 설정한다. 클수록 DISK I/O에 따른 오버헤드를 줄일 수 있다. (기본값=256). 예)

파라미터	설명
	`max_memory_block_count_in_volume_copy = 512`
max_no_of_runtime_caches	런타임 캐시의 최대 개수를 설정한다. (기본값=16). 예) `max_no_of_runtime_caches = 18`
max_post_size	최대 포스트 크기를 설정한다. 포스트 크기는 색인 시 추출된 키워드당 몇 건의 레코드 정보를 유지할 것인가를 결정한다. 즉, 설정된 크기가 클수록 해당 키워드와 연관된 레코드 정보를 많이 유지할 수 있다. 따라서, 검색했을 때의 키워드와 관련한 검색 결과 건수가 많아진다. 검색 결과가 매우 정확해야 하는 대용량 시스템의 경우 전체 문서 건수보다 크게 설정하면 된다. (기본값=350000) 예) `max_post_size = 2000000`
mbcs_stop_level	MBCS색인어(한, 중, 일 등의 EUC문자)의 ngram 레벨에 예외를 설정한다. 즉, mbcs_stop_level에 설정된 값 이하 길이의 색인어는 색인어로 추출이 되지 않는다. 설정값은 ngram level보다 작고 0보다 큰 값을 설정해야 한다. (기본값=0). 예) `mbcs_stop_level = 1`
mmf_server_config	멀티미디어 필터 서버 설정 파일의 위치를 설정한다. (기본값="./mmfd.rc"). 예) `mmf_server_config = ../rcfiles/mymmfd.rc`

파라미터	설명
ngram_level	N-GRAM을 사용 시 지정하며 NGRAM 분해 레벨(2 또는 3)을 설정한다. (기본값=2). N-GRAM에 대한 설명은 'N-GRAM 색인'을 참고한다. (단, ngram_level은 UNIVERSAL 언어 섹션 설정에서만 유효하다.) 예) `ngram_level = 3`
put_volume_update_log	볼륨 업데이트 시의 진행상 로그를 kql 시스템 로그에 남길지 여부를 설정한다. (기본값=Y). 예) `put_volume_update_log = Y`
query_compound_level	쿼리 복합명사 분해 레벨(1~5)을 설정한다. 복합명사 분해 레벨에 대해서는 '키워드 추출 (EXTRACT KEYWORDS)'을 참고한다. (기본값=2). 예) `query_compound_level = 2`
reopen_volume_file_on_disk_error	읽기/쓰기 오류 발생 시 볼륨 reopen 여부를 설정한다. (기본값=n). 예) `reopen_volume_file_on_disk_error = y`
retrieval_time_out	검색 처리에 대한 타임아웃 시간(초)을 설정한다. 설정된 시간이 되면 더 이상 검색을 진행하지 않고 설정된 시간까지 서버로부터 응답받은 검색 결과만을 보여준다. (기본값=0: 타임아웃 시간이 설정되지 않음) 예)

파라미터	설명
	`retrieval_time_out = 3`
remove_replaced_file	볼륨 복사 시 임시 백업해둔 이전 버전의 볼륨 파일을 삭제할지 설정한다. (기본값=y). 예) `remove_replaced_file = n`
replace_expression	검색 쿼리의 where절/group by절/order by절의 특정 문자열(A)을 다른 문자열(B)로 변환하도록 설정한다. 예) `replace_expression = /create_dt desc /create_data desc, create_time desc/`
replace_space_at_space _sensitive_keyword_universal	space sensitive하게 키워드를 생성하는 경우에 공백을 '_'로 치환하여 키를 추출할지 여부를 설정한다. (기본값=n). 예) `replace_space_at_space_sensitive_keyword _universal = y`
rlimit_cpu	최대 CPU 사용률을 설정한다. 최대값은 10000(CPU 100 % 사용을 의미)이다. 최대 CPU 사용률을 50 %로 제한하고 싶으면 '5000'으로 설정한다. (기본값=10000). 예) `rlimit_cpu = 5000`
sbcs_stop_level	SBCS 색인어(영문, 숫자 등)의 ngram 레벨에 예외를 설정한다. 즉, SBCS_STOP_LEVEL에 설정된 값 이하 길이의 색인어는 색인어로 추출이 되지 않는다. 설정값은 ngram level보다 작고 0보다 큰 값을 설정해야 한다. (기본값=0).

파라미터	설명
	예) `sbcs_stop_level = 1`
select_query_time_out	select 질의에 대한 time_out 시간을 설정한다. (기본값=0: 사용하지 않음). 예) `select_query_time_out = 600`
set_keyword_case_universal	영문에 대해서 대/소문자 노멀라이즈 여부를 설정한다. (기본값=1). ● 0: as is (mixed). 노멀라이즈를 수행하지 않는다. ● 1: 소문자로 노멀라이즈 한다. ● 2: 대문자로 노멀라이즈 한다. 예) `set_keyword_case_universal = 0`
sort_import_files	색인을 위한 import 시에 파일을 정렬하여 순서대로 import 할지 설정한다. 원하는 효과를 얻기 위해서는 병렬 태스크 수(no_of_parallel_task)는 1로 고정해야 한다. (기본값=n). 예) `sort_import_files = y`
string_list_seperator	string list 필드 색인 시 키 추출 구분자 등록한다. (기본값=" ", "\t", "\n", "\r"). 예) `string_list_separator = {"," , " ", "\}"}`

파라미터	설명
string_para_seperator	string para 필드 색인 시 키 추출 구분자 등록한다. (기본값=" ", "\t", "\n", "\r"). 예) `string_para_separator = {",", "\t", "\n", "\r", " "}`
sync_downloaded_file	볼륨 복사 시 디바이스 파일의 디스크 쓰기를 즉시 동기화할지 설정한다. (기본값=y). 예) `sync_downloaded_file = n`
sync_downloaded_memory_patch	볼륨 업데이트 시 모든 fd를 동기화할지 설정한다. (기본값=n). 예) `sync_downloaded_memory_patch = y`
synonym_compound_level	동의어 확장 시 복합명사 분해레벨을 설정한다. 이 설정은 docruzerd.rc 설정의 morph_synonym_search = 1로 설정된 경우에만 동작하며, 설정되지 않은 경우 query_compound_level과 동일하게 사용된다. (기본값=-1). 예) `synonym_compound_level = 2`

trace_interval_indexing	색인 과정 중 색인 단계에서 로그를 남기는 단위를 설정한다. (기본값=500000)
	
	예)
	
	`TRACE_INTERVAL_INDEXING = 500000`
	`(50만 건당 로그)`

trace_interval_stroing	색인 과정 중 데이터 저장 단계에서 로그를 남기는 단위를 설정한다. (기본값=500000)

파라미터	설명

	예)
	`TRACE_INTERVAL_STORING = 500000`
	`(50만 건당 로그)`

unique_key_duplicate	'unique' 속성이 있는 필드에 중복된 값이 들어올 경우의 처리 방법을 설정한다. (기본값=error). ● error: 색인을 중단하고 오류 처리 ● ignore: 나중에 들어온 키의 데이터를 무시하고 진행 ● overwrite: 나중에 들어온 키의 데이터로 업데이트 예) `unique_key_duplicate = overwrite`
use_default_thread_stack	디폴트 스레드 스택을 사용할지 설정한다. 특수한 환경에서 OS의 디폴트 스택 설정을 그대로 사용하고자 할 때 이용한다. n으로 설정하는 경우 kql.log에 유용한 스레드 정보가 나오지 않을 수도 있으므로 주의한다. (기본값=n). 예) `use_default_thread_stack = y`
use_gate_way_loader	게이트웨이 로더를 사용할지 설정한다. (기본값=n). 예) `use_gate_way_loader = y`
use_stem_keyword	stemming된 단어의 색인어로 저장 여부를 설정한다. (기본값=y). 영어, 유럽어에 대해 지정 가능하다. 예)

파라미터	설명
	`use_stem_keyword = y`
use_stem_keyword_en	영문 색인 시 stemming된 단어의 색인어로 저장 여부를 설정한다. (기본값=y). 한국어, 중국어, 일본어에 대해 지정 가능하다. 예) `use_stem_keyword_en = y`
use_tagged_keyword	품사 기호(A(부사), V(동사,형용사), D(관형사))를 부착한 키워드를 추출한다. (기본값=n). ● y: 각 형태소 분석 결과의 말미에 품사를 붙인다. ● n: 각 형태소 분석 결과의 말미에 품사를 붙이지 않는다. 예) `use_tagged_keyword = y`
use_word_keyword	워드단위 키워드를 추출하며 추출된 키워드에는 '@'가 접두어로 붙는다. (기본값=N). 예) `use_word_keyword = n`
use_word_keyword_en	영문 색인 시 stemming되기 전의 원래 단어의 색인어로 저장 여부를 설정한다. (기본값=n). 한국어에 대해 지정 가능하다. use_stem_keyword_en 과 use_word_keyword_en이 둘 다 'y'이면 stemming 전후 단어 모두 색인어로 저장되고, 둘 다 'n'이면 영문은 색인이 되지 않는다. 예)

파라미터	설명
	`use_word_keyword_en = n`
user_dic_for_coined_word	신조어 사용자정의 사전을 등록한다. 신조어 사용자정의 사전에 대한 자세한 내용은 '사용자 사전 - 신조어 (RELOAD USER DIC FOR COINED WORD)'을 참고한다. 예) `user_dic_for_coined_word =../txt/coined.txt`
user_dic_for_compiled _synonym	컴파일된 동의어 사용자정의 사전을 등록한다. 컴파일된 동의어 사용자정의 사전에 대한 자세한 내용은 '사용자 사전 - 컴파일 된 동의어(RELOAD USER DIC FOR COMPILED SYNONYM)'를 참고한다. 예) `user_dic_for_compiled_synonym = ../txt/compiled_synonym.txt`
user_dic_for_stop_word	불용어 사용자정의 사전을 등록한다. 불용어 사용자정의 사전에 대한 자세한 내용은 '사용자 사전 - 불용어 (RELOAD USER DIC FOR STOP WORD)'를 참고한다. 예) `user_dic_for_stop_word = ../txt/stop.txt`
user_dic_for_synonym	동의어 사용자정의 사전을 등록한다. 동의어 사용자정의 사전에 대한 자세한 내용은 '사용자 사전 - 동의어 (RELOAD USER DIC FOR SYNONYM)'를 참고한다. 예)

파라미터	설명
	`user_dic_for_synonym = ../txt/synonym.txt`
use_half_width_keyword_en	반각 영문 키워드를 추출할지 설정한다. (기본값=n). 예) `use_half_width_keyword_en = y`
use_math_keyword	수식 키워드를 추출할지 설정한다. (기본값=n). 예) `use_math_keyword = y`
use_natural_relevance_for_boolean	boolean 검색 시 랭킹 결과를 natural 검색의 랭킹 결과와 유사하게 설정할지 설정한다. (기본값=n). 예) `use_natural_relevance_for_boolean = y`
use_reversed_sort	order by 구문 처리 시 역정렬 기능을 사용할지 여부를 설정한다. (기본값=y). 예) `use_reversed_sort = n`
use_socket_tcp_nodelay	TCP 소켓을 nodelay 모드로 사용할지를 설정한다. (기본값=n). 예) `use_socket_tcp_nodelay = y`
use_space_insensitive_keyword_document	색인 문서에 대한 space insensitive 키워드 생성여부를 설정한다. (기본값=y). 예) `use_space_insensitive_keyword_document = n`

파라미터	설명
use_space_insensitive _keyword_query	질의문에 대한 space insensitive 키워드의 생성여부를 설정한다. (기본값=y). 예) `use_space_insensitive_keyword_query = n`
use_syllable_based_word_no	음절 단위로 word_no_를 추출할지 여부를 설정한다. (기본값=y). 예) `use_syllable_based_word_no = n`
use_symbol_normalize	`를 '(작은따옴표)로 변경한다. (기본값=y). 예) `use_symbol_normalize = n`
use_symbol_universal	기호가 포함된 색인어 생성 여부를 설정한다. (기본값=y). 예) `use_symbol_universal = n`
use_word_space_correction	1 음절어가 3 회이상 반복될때 띄어쓰기를 보정할지 설정한다. (기본값=y). 예) `use_word_space_correction = n`
use_wordph_query_universal	어절키워드 앞에 부착된 태그(0xFFFF)를 사용할지 여부를 설정한다. (기본값=y). 예) `use_wordph_query_universal = n`

파라미터	설명
use_zero_proximity_weight	모두 붙여 쓴 문장이 띄어 쓴 문장보다 최상위에 나타나는 문제 완화하는 것을 설정한다. (기본값=y). 예) `use_zero_proximity_weight = n`
working_memory_usage_in_bulk_indexing	작업 메모리(max_working_memory_size로 설정되어 있는 메모리) 중에서 벌크 색인 시 캐시로 선점하는 양(0~100%)을 제한한다. (기본값=95(%)). 예) `working_memory_usage_in_bulk_indexing = 60`

2.8. 독크루저 설정 파일 (docruzerd.rc)

'독크루저 설정 파일(docruzer configuration file)'은 검색 서버 주소, 시나리오 등 검색 서비스 제어에 필요한 세부사항을 설정하는 파일이다. 검색 시스템의 제어 설정은 KQL 설정 파일을 이용하며, 검색 서비스와 관련한 설정은 독크루저 설정 파일을 사용한다.

독크루저 검색 서비스는 'docruzerd' 데몬을 이용하여 검색 서비스를 처리하는데, 이때 KQL은 시스템의 커널과 같은 역할을 수행한다. 검색 데몬은 KQL의 리스너를 통한 통신으로 시스템의 하위레벨 자원을 사용한다.

- 작성 형식

  KQL 설정 파일 작성 형식과 같다.

  디렉터리 경로를 설정값으로 가지는 경우 상대 주소, 절대 주소 모두 사용 가능하며 상대 경로의 기준 위치는 bin 디렉터리이다.

- 주석

  /* */, ';'를 사용한다.

- 기본값

독크루저 설정 파일에 명시하지 않았을 경우 자동으로 설정 변수에 할당되는 값이다. 특별한 경우가 아니라면 기본값을 사용한다.

2.8.1. 기본 설정

독크루저 기본 설정 파라미터에 대한 설명은 다음과 같다.

파라미터	설명
include	다른 환경 구성 파일을 포함시키도록 설정한다. 설정값은 별도 작성된 설정 파일의 경로이다. 주로 시나리오를 별도의 파일로 작성하여 등록하기 위해 사용한다. include를 이용하여 한 개 이상의 시나리오를 등록할 수 있다. 예) `include = "../scenario/tbl_scn.scn"`
kql_link (KQL 링크)	검색 시스템이 사용할 KQL 자원을 설정한다. 설정값은 검색 시스템이 사용할 KQL의 핸들명과 KQL 설정 파일이다. 검색 서비스는 KQL 핸들명을 사용하는 시나리오를 이용하여 검색한다. 예) `kql_link = h1, kql.rc`
log_file_location (로그 파일 위치)	독크루저가 처리한 작업을 기록하는 파일이 만들어지는 위치를 설정한다(생략 가능). 예) `log_file_location = ../logs`
scenario (시나리오)	검색 결과로 가져올 볼륨, 테이블, 필드와 가져올 필드의 크기, 하이라이팅 태그를 설정한다. 시나리오에 대한 자세한 설명은 '시나리오 (SCENARIO)'를 참고한다. 예)

파라미터	설명

	`scenario nr1`
	`{`
	`  link = h1`
	`  volume = tv`
	`  table = GISA`
	`  field = TT (,"<b>", "</b>")`
	`  field = BD (300, "<b>", "</b>")`
	`  field = DT`
	`}`

daemon_service_port (서비스 포트 번호)	서비스 포트 번호는 독크루저 서버가 클라이언트 요청을 받아들일 네트워크 포트 번호를 말한다. 예) `daemon_service_port = 7677`
daemon_control_port (컨트롤 포트 번호)	컨트롤 포트 번호는 독크루저 서버가 제어 요청을 받아들일 네트워크 포트 번호를 말한다. 보통 (daemon_service_port + 1)로 설정한다. 예) `daemon_control_port = 7678`

2.8.2. 세부 설정

독크루저 세부 설정 파라미터에 대한 설명은 다음과 같다.

파라미터	설명	기본값
allow_empty_where_clause	검색 질의어가 입력되지 않은 경우에도 검색을 허용하는 옵션이다. ● 0: 질의어를 무시하여 검색 결과를 '0'으로 처리한다. ● 1: 테이블의 모든 레코드를 가져온다.	0

파라미터	설명	기본값
	예) `allow_empty_where_clause = 1`	
daemon_cache _domain_no_search	검색 요청 캐시 사용 시에 사용할 캐시 모듈의 도메인 번호를 설정한다. 캐시에 대한 상세 설명은 캐시 모듈의 해당 도메인에 직접 설정한다.	
daemon_data_location	독크루저 모듈이 있는 디렉터리 위치를 지정한다. 검색 서버가 인기검색어나 검색어 자동완성과 같은 부가서비스 모듈을 사용하는 경우 반드시 설정한다. 예) `daemon_data_location= ../data`	

daemon_product_key	독크루저 제품 키를 등록한다. 독크루저의 일부 모듈은 사용 권한이 있는 제품 키를 등록해야만 사용할 수 있다.
	예)
	`daemon_product_key`
	`= ETJTKCB-6XNT1RB-M1V10R-NM4COFB`	

kql_socket_connection _timeout	검색 데몬과 kql 간 통신에서의 receive timeout 시간(초)을 설정한다. 예) `kql_socket_connection_timeout = 6`	5(초)
log_broken_request _cancelation	클라이언트가 검색 서버 응답을 받기 전에 미리 연결을 끊어버린 경우 로그를 남길지 결정한다. ● 0: 로그를 남기지 않는다.	0

파라미터	설명	기본값
	● 1: 로그를 남긴다. 예) `log_broken_request_cancelation = 1`	
log_client_ip	검색 로그 생성 시 검색 요청 클라이언트의 IP 정보를 검색 로그에 남길지 여부를 설정한다. ● 0: 로그를 남기지 않는다. ● 1: 로그를 남긴다. 예) `log_client_ip = 1`	0
log_highlight_text	검색 로그 파일에 검색 키워드의 하이라이트와 관련하여 어떤 정보를 남길지 지정한다. ● 0: 로그를 남기지 않는다. ● 1: 주어진 그대로 로그를 남긴다. ● 2: 형태소 분석 후 로그를 남긴다. ● 3: 주어진 형태 그대로의 값과 형태소 분석 결과값 두 가지 모두를 로그로 남긴다. ● 4: '3'으로 설정했을 때 남기는 2가지 값 이외에 동의어 대표값도 로그로 남긴다. 예) `log_highlight_text = 2`	0
log_packet_size	검색 로그에 주고받은 패킷 크기를 로그로 남길지 여부를 설정한다.	0

파라미터	설명	기본값
	● 0: 로그를 남기지 않는다. ● 1: 로그를 남긴다. 예) `log_packet_size = 1`	
log_socket_3way_mode	검색 로그 생성 시 3-way handshaking 방식을 사용하여 검색 서비스를 제공하는 정보를 로그로 남길지 여부를 설정한다. ● 0: 로그를 남기지 않는다. ● 1: 로그를 남긴다. 예) `log_socket_3way_mode = 1`	0
log_system_status	검색 엔진 시스템 상태를 로그(~.msg 로그 파일)에 남길지 여부를 설정한다. ● 0: 로그를 남기지 않는다. ● 1: 검색 처리 상태 정보를 로그를 남긴다(1초 간격). 예) `log_system_status = 1`	0
max_acpt_que_size	검색/시스템 제어 요청을 보관하는 큐의 크기를 지정한다. 최대값은 '1024'까지 가능하다. 예) `max_acpt_que_size = 200`	200

파라미터	설명	기본값
max_acpt_thread_count	검색/시스템 제어 요청 처리 단계 중 소켓 접속을 처리하는 스레드의 개수를 지정한다. 예) `max_acpt_thread_count = 2`	2(개)
max_backlog_queue_size	클라이언트의 접속 요청을 받아들이는 소켓의 backlog 개수를 지정한다. '0'보다 큰 값으로 운영체제가 지원해주는 한도까지 지정 가능하다. 예) `max_backlog_queue_size = 156`	16(개)
max_highlight_mark_count	검색 결과 요약문에 최대로 출현 가능한 키워드 개수를 지정한다. 예) `max_highlight_mark_count = 100`	50(개)
max_linger_timeout	소켓을 close 하기 전에 전송할 데이터가 큐에 남아있는 경우 전송할 때까지 대기할 timeout 시간(초)을 설정한다. 예) `max_linger_timeout = 5`	7(초)
max_recv_que_size	패킷 수신이 끝난 클라이언트 요청을 보관하는 큐의 크기를 지정한다. 예) `max_recv_que_size = 200`	200
max_recv_thread_count	클라이언트 요청 처리 단계 중 패킷 수신 작업을 하는 스레드의 개수를 지정한다.	2(개)

파라미터	설명	기본값
	예) `max_recv_thread_count = 2`	
max_retry_search_period	실시간 색인 때문에 볼륨에 잠금(locking)이 걸려서 검색을 하지 못하는 경우, 검색을 재시도하는 시간을 설정한다. 예) `max_retry_search_period = 10`	0(초)
max_retry_socket_bind	서버의 주소가 이미 사용 중인 경우의 bind 재시도 횟수를 설정한다. 예) `max_retry_socket_bind = 5`	2회
max_send_que_size	검색된 결과를 보관하는 큐의 개수를 지정한다. 예) `max_send_que_size = 200`	200(개)
max_send_thread_count	클라이언트 요청 처리 단계 중 패킷 송신 작업을 하는 스레드의 개수를 지정한다. 예) `max_send_thread_count = 2`	2(개)
max_socket_timeout	소켓의 accept/receive timeout 시간(초)을 지정한다. 예) `max_socket_timeout = 3`	300(초)
max_socket_timeout_msec	소켓의 accept/receive timeout 시간(밀리초)을 지정한다. max_socket_timeout 설정에 더해진다.	0(밀리초)

파라미터	설명	기본값
	예) `max_socket_timeout_msec = 500`	
max_socket_recv_timeout_msec	소켓의 receive timeout 시간(밀리초)을 지정한다. 0보다 큰 값으로 설정된 경우 max_socket_timeout,max_socket_timeout_msec 설정을 무시하고 이 값만 사용한다. 예) `max_socket_timeout_msec = 500`	-1(사용하지 않음)
max_work_thread_count	실제로 검색을 수행하는 스레드의 개수를 지정한다. no_of_parallel_task 와 같은 의미이다. 2가지 항목은 같은 의미이며, 동시에 설정한 경우 나중에 설정한 값으로 적용된다. 예) `max_work_thread_count = 4`	
morph_synonym_search	동의어 확장 검색 시 형태소 분석 결과를 사용할지 여부를 지정한다. 예) `morph_synonym_search = 1`	
no_of_log_file_per_day	하루에 생성할 검색 로그의 파일 개수를 설정한다. '4'인 경우는 6 시간마다 새로운 파일을 생성하게 된다. (24/4=6) 예) `no_of_log_file_per_day=4`	1(개)
no_of_sys_log_file_per_day	하루에 생성할 시스템 로그의 파일 개수를 설정한다. '4'인 경우는 6 시간마다 새로운 파일을 생성하게 된다. (24/4=6)	1(개)

파라미터	설명	기본값
	예) `no_of_sys_log_file_per_day=4`	
process_priority	검색엔진 데몬의 프로세스 우선 순위(priority)를 설정하는 항목이다. ● UNIX: -20~20(-20이 우선 순위가 가장 높음) ● Windows: 0~5(0이 우선 순위가 가장 낮음) 예) `process_priority = 20`	0
synonym_insensitive_highlight	동의어 모두에 대한 하이라이팅 여부를 설정한다. ● 0: 하이라이팅 하지 않는다. ● 1: 하이라이팅 한다. 예) `synonym_insensitive_highlight = 1`	0
sys_log_file_location	시스템 로그(~.msg)의 디렉터리 위치를 지정한다. 예) `sys_log_file_location = ../logs`	
trace_working_thread	스레드가 수행하고 있는 곳의 위치 추적 기능을 사용한다. 예) `trace_working_thread = 1`	0

파라미터	설명	기본값
use_parallel_union_select	분산볼륨 병렬 검색 사용 여부를 설정한다. ● 0: 사용하지 않는다. ● 1: 사용한다. 예) `use_parallel_union_select = 1`	0
use_broken_request _cancelation	검색 요청을 수행하기에 앞서 클라이언트와의 연결상황을 검사하여 끊어진 경우에는 수행 여부를 설정한다. ● 0: 수행하지 않는다. ● 1: 수행한다. 예) `use_broken_request_cancelation = 1`	0
use_extended_search_api	특별히 제작된 내부 검색 함수 루틴 (KQLC_RetrieveXP)으로 검색하고자 할 때 1로 설정한다. 예) `use_extended_search_api = 1`	0
use_highlight_text	검색 결과의 하이라이팅 방법을 지정한다. ● 0: 검색 질의어의 형태소 분석 결과를 사용한다. ● 1: 검색 요청 시 따로 지정한 하이라이트 문자열을 사용한다.	

파라미터	설명	기본값
	• 2: 검색 요청 시 따로 지정한 하이라이트 문자열의 형태소 분석 결과를 사용한다. • 3: 1, 2 항목을 동시에 하이라이팅에 사용한다. • 4: 시맨틱 검색용 하이라이팅에 사용한다. 예) `use_highlight_text = 1`	
use_socket_3way_mode	3-way handshaking 방식의 사용 여부를 설정한다. 3-way handshaking은 클라이언트가 데이터를 다 받은 것을 확인한 후에 소켓 연결을 종료하는 방식이다. • 0: 사용하지 않는다. • 1: 사용한다. 예) `use_socket_3way_mode = 1`	0

2.9. 기동 및 정지

2.9.1. Windows

Windows에서 기동하려면 독크루저가 설치된 bin/ 디렉터리로 이동하여 셸 명령행에 다음과 같이 입력한다.

```
d:\konan\bin> docruzerd.exe start -f docruzerd.rc
>DOCRUZER: READY
d:\konan\bin>
```

-f docruzerd.rc 옵션은 생략 가능하다. 생략 시 docruzerd.rc 파일을 설정 파일로 찾아 처리한다.

정지하려면 셸 명령행에 다음과 같이 입력한다.

```
d:\konan\bin> docruzerd.exe stop -f docruzerd.rc
>DOCRUZER: SHUTDOWN
d:\konan\bin>
```

2.9.2. GNU/Linux

GNU/Linux에서 기동하려면 독크루저가 설치된 bin/ 디렉터리로 이동하여 셸 명령행에 다음과 같이 입력한다.

```
$ ./docruzerd start -f docruzerd.rc
>DOCRUZER: READY
$
```

-f docruzerd.rc 옵션은 생략 가능하다. 생략 시 docruzerd.rc 파일을 설정 파일로 찾아 처리한다.

정지하려면 셸 명령행에 다음과 같이 입력한다.

```
$ ./docruzerd stop -f docruzerd.rc
>DOCRUZER: SHUTDOWN
$
```

2.9.3. UNIX

Linux와 동일한 방법으로 기동 및 정지한다.

2.10. 제거 절차

설치 디렉터리 전체를 삭제하면 독크루저가 삭제된다. 현재는 별도로 시스템에 설치하는 컴포넌트가 없다.

KQL

이 장에서는 독크루저의 데이터 타입과 독크루저 명령어에 대해 설명하고 예를 이용하여 각 명령어의 동작을 확인할 수 있도록 한다.

KQL(Konan Query Language)은 독크루저 엔진의 볼륨을 접근 및 관리하기 위한 데이터베이스 명령어로 인터프리터를 통해 실행된다. 현재 지원되고 있는 명령어와 개략적인 기능이 아래에 설명되어 있다. 모든 키워드는 대소문자 구분이 없지만, 그 밖의 모든 식별자는 대소문자를 구분한다. 또한 모든 명령어는 하나의 단위 트랜잭션으로 간주되어 부분 실행이 방지된다. 성공적으로 종료된 명령어의 실행은 취소할 수 없다.

볼륨 제어

다음은 볼륨 제어 명령어와 기능의 목록이다.

명령어 구문	기능
`SHOW(LIST) VOLUMES`	볼륨 목록 보기
`CREATE VOLUME [IF NOT EXISTS | BY FORCE] volume_name [INITIAL SIZE = number {MB|GB|TB}]`	볼륨 생성
`USE VOLUME volume_name`	볼륨 선택
`COPY VOLUME //host:port/src_volume_name TO dst_volume_name [INCREMENTALLY][USE SNAPSHOT]`	볼륨 복사
`SWAP VOLUME volume_name1 volume_name2`	볼륨 교환
`CREATE SNAPSHOT FOR VOLUME volume_name [RESTART | COMPRESS=[ON|OFF*]]`	볼륨 스냅샷 생성

테이블 제어

다음은 테이블 제어 명령어와 기능의 목록이다.

명령어 구문	기능
`SHOW TABLES`	테이블 목록 보기
`CREATE TABLE table_name ( field_definition )`	테이블 생성
`DROP TABLE [IF EXISTS] table_name`	테이블 삭제

`ALTER TABLE table_name [`
`  CHANGE COLUMN old_field_name new_field_name new_type |`
`  CHANGE TAG new_func_decl |`
`  CHANGE CONSTRAINT const_name const_decl |`
`  DELETE CONSTRAINT const_name]`
	테이블 스키마 변경

`EXPLAIN table_name [MORE]`	테이블 상세 보기
`CREATE INDEX index_name ON table_name ( field_name_list )`	인덱스 생성
`DROP INDEX index_name ON table_name`	인덱스 삭제
`INSERT INTO table_name SET field_name = literal, ...`	레코드 삽입
`UPDATE table_name SET field_name = literal, ...` ` [WHERE where_definition] [LIMIT number]`	레코드 수정
`DELETE FROM table_name [WHERE where_definition]` ` [LIMIT number]`	where 조건에 해당하는 레코드 삭제
`DELETE rowid, ... FROM table_name`	rowid로 레코드 삭제
`UNDELETE { * | rowid, ... } FROM table_name TRASH BOX`	레코드 복원
`TRUNCATE TABLE table_name`	테이블 비우기

명령어 구문	기능
SELECT {* | *field_name_list*} FROM *table_name* [WHERE *where_definition*] [*method*] [ORDER BY { *field_name* | $ROWID | $RELEVANCE | $MATCHFIELD(*field_name*, ...) | $SIZE(*field_name*) | *usersort_field_name* (*key* ...) | $CATEGORYFIELD(*field_name*(*domain*) ..., '*keyword*') | $USERDEF(*libuser.so*)} [ASC|DESC] , ...] [WEIGHT BY *field_name=number*, ...] [EVALUATE BY modrev(*param*) [domain=*domain*][, cmd=*cmd*]] [EXCLUDE BY *field_name*(*domain*)] [EXTRACT BY {MIN | MAX | MINMAX}(*field_name*)] [GROUP BY *field_name* [ORDER BY COUNT(*) [ASC|DESC]]] [DISTINCT BY *field_name*] [PAGELENGTH=*number*] [PAGEFORMAT=LIST | CSV | CLUSTERED BY *field_name* [*count*]] [AS LITERAL]	레코드 검색(select)
RETRIEVE {* | *field_name_list*} FROM *table_name* [WHERE *where_definition*] [*method*] [ORDER BY { *field_name* | $ROWID | $RELEVANCE | $MATCHFIELD(*field_name*, ...) | $SIZE(*field_name*) | *usersort_field_name*(*key* ...) | $CATEGORYFIELD(*field_name*(*domain*) ..., '*keyword*') | $USERDEF(*libuser.so*)} [ASC|DESC] , ...] [WEIGHT BY *field_name=number*, ...] [EVALUATE BY modrev(*param*) [domain=*domain*][, cmd=*cmd*]] [EXCLUDE BY *field_name*(*domain*)] [EXTRACT BY {MIN | MAX | MINMAX}(*field_name*)] [GROUP BY *field_name* [ORDER BY COUNT(*) [ASC|DESC]]] [DISTINCT BY *field_name*] [ABSOLUTE] [PAGELENGTH=*number*] [PAGEFORMAT=LIST | CSV | CLUSTERED BY *field_name* [*count*]] [AS LITERAL]	레코드 검색(retrieve)
DUMP TO *local_filename statement*	실행 결과 덤프

명령어 구문	기능
EXPORT RECORDS FROM *table_name* [WHERE *expr*] TO [*//listener/*]*folder* [DEPTH *number*][DIVIDE BY *number*] [LIMIT *number*][FILE SIZE *number*] [PREFIX *string*][INFIX LENGTH *number*][SUFFIX *string*] FORMAT *format_decl*	레코드 내보내기(export)
BUILD INDEX TO *table_name*[.*field_name*][PARALLEL *number*]	동적 필드 색인
TRUNCATE INDEX TO *table_name*[.*field_name*]	인덱스 비우기
VIEW INDEX TO *table_name.field_name* [PATTERN=*term_spec* | KEYWORD=*term*] [PAGELENGTH=*number*]	색인 결과 확인
EDIT INDEX TO *table_name.field_name* [KEYWORD=*term*] {DELETE [*rowid*, ...] | INSERT *rowid* [*loc*, ...]}	색인 결과 편집
PUT INDEX SUMMARY OF *table_name.field_name* TO *file_path*	색인 결과 요약

레코드 가져오기 및 게이트 웨이

다음은 레코드 가져오기 및 게이트 웨이 명령어와 기능의 목록이다.

명령어 구문	기능
CREATE GATEWAY *gateway_name* ON *table_name* TO FILE SYSTEM FORMAT FIELD [LINE] {STARTS|ENDS} WITH *field_name=value*, ...	파일 시스템 게이트웨이 생성
IMPORT FILES [*//listener/*]*folder file_spec* TO *table_name* [THROUGH *gateway_name*| FORMAT *format_decl*] [PARALLEL *number*] [OPERATION {INSERT|STORE|INSERT_OR_SKIP|UPDATE| UPDATE_OR_INSERT|UPDATE_OR_SKIP|DELETE|DELETE_OR_SKIP|	원격/로컬 파일 레코드 가져오기(import files)

명령어 구문	기능
DELETE_AND_INSERT|DELETE_INSERT|INSERT_IF_DELETE}] [LIMIT *number*][LANGUAGE=*language*][CHARSET=*charset*]	
IMPORT RECORDS FROM FILE SYSTEM {TO *table_name* THROUGH *gateway_name* SELECT *dir_path* [*file_spec*] [OPERATION {INSERT|STORE}] [LIMIT *number*][LANGUAGE=*language*][CHARSET=*charset*] }	파일 시스템 레코드 가져오기 (import records)
CREATE GATEWAY *gateway_name* ON *dst_tn* [[TO ORACLE *src_tn* FORMAT *format_decl*] | [TO SQLSERVER *database.dbo.src_tn* FORMAT *format_decl*] | [TO DB2 *src_tn* FORMAT *format_decl*] | [TO SYBASE *database.dbo.src_tn* FORMAT *format_decl*] | [TO NOTES *nsf_path* FORMAT *format_decl*] | [TO MYSQL *src_tn* FORMAT *format_decl*] | [TO EXCHANGE *src_tn* FORMAT *format_decl*] | [TO CUBRID *src_tn* FORMAT *format_decl*]]	데이터베이스 게이트웨이 생성
IMPORT RECORDS FROM [[ORACLE(*username, password, tnsname*)] | [SQLSERVER (*hostname, username, password*)] | [DB2 (*database, username, password* [,*schema_name*])] | [SYBASE (*hostname, username, password*)] | [MYSQL (hostname, username, password)]] TO dst_table_name THROUGH gateway_name [SELECT expr] [LIMIT number] [LANGUAGE=language] [CHARSET=charset]	데이터베이스 레코드 가져오기
IMPORT RECORDS FROM NOTES (*hostname, profile, password*) TO *table_name* THROUGH *gateway_name* [SELECT {ALL | LATEST}] [LIMIT *number*][LANGUAGE=*language*][CHARSET=*charset*]	NOTES 레코드 가져오기

와처

다음은 와처 명령어와 기능의 목록이다.

명령어 구문	기능
CREATE WATCHER [ON *table_space*] AT *dbms_spec*	와처 생성
DROP WATCHER AT *dbms_spec*	와처 제거
NOTIFY WATCHER AT *dbms_spec* GET KEY FIELD {*db_vw_fn* FROM *db_table_name.fn, ...*} [GET VIEW FIELD {*db_vw_fn, ...* FROM *db_table_name, ...*}] PUT TO gateway_name OF kql_table_name	와처 동작 지정
UNNOTIFY WATCHER AT *dbms_spec db_vw*	와처 동작 삭제
RESTART WATCHER AT *dbms_spec* [*db_vw*]	와처 변경 기록 삭제
UPDATE *kql_table_name* WATCHING *dbms_spec* THROUGH *kql_gateway_name* [LIMIT *number*][TIMELIMIT *number* SEC] [WRITE_METHOD={BUFFERED | UNBUFFERED}] [HANDLING *exception* ON {INSERT|UPDATE|DELETE} BY {INSERT|UPDATE|IGNORE|FAIL}] [LANGUAGE=*language*][CHARSET=*charset*]	와처를 통한 레코드 수입

스케줄

다음은 스케줄 명령어와 기능의 목록이다.

명령어 구문	기능
`LIST (SHOW) SCHEDULES`	스케줄 목록 보기

명령어 구문	기능
`SCHEDULE { number | name | * } ON`	스케줄 시작
`SCHEDULE { number | name | * } OFF`	스케줄 중단
`ATTACH SCHEDULE name, "action", period`	스케줄 동적 등록

언어 데이터 제어

다음은 언어 데이터 제어 명령어와 기능의 목록이다.

명령어 구문	기능
RELOAD USER DIC FOR { COINED WORD | STOP WORD | SYNONYM | WORD WEIGHT } *text_file_path* [LANGUAGE=*language*][CHARSET=*charset*]	사용자 사전 등록
`EXTRACT KEYWORDS FROM { QUERY | DOCUMENT } literal` `[LANGUAGE=language][CHARSET=charset]`	색인어 추출
`EXPAND QUERY string`	동의어 확인

캐시

다음은 캐시 명령어와 기능의 목록이다.

명령어 구문	기능

`ATTACH FIELD KEY CACHE TO table_name.field_name`
`[ROW_SIZE=number]`
`[COLUMN_SIZE=number]`
	캐시 적재 (field order)

명령어 구문	기능
`ATTACH KEY ORDER CACHE TO table_name.field_name`	캐시 적재 (key order)
`ATTACH POST CACHE TO table_name.field_name`	캐시 적재 (post)
`DETACH { FIELD KEY | KEY ORDER | POST } CACHE FROM table_name.field_name`	캐시 해제
`MONITOR { FIELD KEY | KEY ORDER | POST } CACHE ON table_name.field_name`	캐시 모니터

기타

다음은 기타 명령어와 기능의 목록이다.

명령어 구문	기능
RUN *kql_file_path*	명령어 일괄 처리
EXEC *shell_command*	셸명령 실행
ECHO *string*	문자열 출력
SCRIPT *file_name*	결과 화면 파일 출력
AUTOFILL [ON | OFF]	문장 자동완성
TIME [ON | OFF]	처리시각 표시
HELP	도움말

3.1. KQL 인터프리터 (KQL)

KQL 명령문의 실행은 같은 이름의 인터프리터를 통해 이루어진다. KQL 인터프리터는 커맨드 방식으로 명령어를 처리한다. KQL 인터프리터를 이용하여 검색 시스템을 제어하는 상황은 2 가지가 있다. 첫 번째는 실행 중인 시스템에 연결하여 시스템을 제어하는 경우이고, 두 번째는 검색 데몬 실행 없이 KQL만으로 시스템을 제어하는 경우이다. 첫 번째의 경우는 검색 서비스를 제공하는 있는 상황에서 서비스 중단 없이 시스템에

대한 제어를 수행하는 경우이며, 두 번째는 초기 검색 시스템을 구축하거나, 시스템 재구축 등 검색 서비스 없이 검색 환경을 조성하는 단계에서 시스템을 제어하는 경우이다.

검색 데몬이 실행 중일 때 시스템을 제어하기 위해서는 KQL의 리스너(listener, kql.rc) 설정을 이용하여 검색 데몬에 연결하여 시스템 명령어를 실행한다.

연결은 명령행에서 다음 형식과 같이 명령어를 입력한다.

```
kql –f server_ip|host_address :listener_port
```

KQL 명령만으로 시스템으로 제어하기 위해 KQL 인터프리터를 실행할 때는 명령행에서 다음 형식과 같이 명령어를 입력한다. 구문 중 kql.rc는 KQL 설정 파일이 bin 디렉터리에 없다면 파일이 존재하는 디렉터리 경로까지 입력하면 된다.

```
kql –f kql.rc
```

KQL 설정 파일을 이용하여 실행된 경우 해당 실행이 데몬 역할을 하여 KQL 리스너를 이용한 명령어 요청을 수락, 실행할 수 있다.

다음은 KQL의 명령행 구문과 표준 입력에서 KQL 명령문을 입력 받아 실행시켜 주는 대화식 실행의 예이다. KQL 인터프리터 또는 KQL API가 관련된 모든 부분에서 –f 옵션 뒤의 링크값의 규약은 동일하다. 즉, 링크값이 시스템 구성 파일의 경로이면 KQL 인터프리터 또는 응용 자신이 서버 데몬이 되고, 링크값이 원격 데몬의 리스너 주소이면 KQL 인터프리터 또는 응용은 원격 데몬의 클라이언트가 된다. 일단 링크가 설정되고 나면 원격이든 지역 링크이든 작업 절차나 효과는 동일하다.

```
$ kql -h
KQL Interpreter (Version 3.5.2 Linux i686)
Copyright (c) 1999-2011 Konan Technology, Inc.
Usage: ./kql [-f <link>] [-x "<queries>"]
         link = [<host:port> | <sys conf file>]
         ./kql -h,-help
         ./kql -t,-time
         ./kql -v,-version
         ./kql -q,-query
$ kql -f kql.rc
Welcome to KQL Interpreter (Version 3.5.2 Linux i686).
Type 'help' for help. Commands end with ';'.
```

```
Copyright (c) 1999-2011 Konan Technology, Inc.

kql> echo "kql started.";
OK
```

'ECHO'는 해당 문자열을 로그에 출력해 주는 명령이다. 대화식에는 'HELP', 'QUIT', 'SCRIPT'의 3 가지 제어 명령이 있으며 이들은 볼륨 데이터 관리가 아닌 인터프리터의 실행 자체에 관계된 명령어라는 점에서 KQL 명령문과는 다르다.

다음은 각 제어 명령에 대한 설명이다.

명령어	설명
HELP	도움말을 출력
QUIT	인터프리터의 실행을 종료
SCRIPT	표준 출력되는 인터프리터의 실행 과정 및 결과를 파일에 기록

시작과 종료는 각각 SCRIPT [*filename*], SCRIPT OFF를 입력한다.

```
$ kql -f kql.rc
Welcome to KQL Interpreter (Version 3.5.2 Linux i686).
Type 'help' for help. Commands end with ';'.
Copyright (c) 1999-2011 Konan Technology, Inc.

kql> script
Script started, file is 'kql.out'.

kql> script off
Script done, file is 'kql.out'.

kql>
```

KQL 인터프리터는 비대화식 처리도 지원하고 있다. 비대화식 처리에서는 처리하고자 하는 하나 이상의 KQL 명령문들을 –x 옵션의 인자로 전달하면 된다. 각 KQL 명령문은 ';'으로 끝난다.

```
$ kql -f kql.rc -x "use volume myvol; explain tbl;"
```

또는 스크립트 파일을 실행해주는 KQL 명령 RUN을 이용하여 비대화식 실행을 하면 다음과 같이 더욱 간단히 명령행을 구성하여 동일한 결과를 얻을 수 있다. 실제 운영 시에 유용하게 사용될 수 있는 방식이다.

```
--------- ../script/my_script.kql ------------
use volume myvol;
explain tbl;

$ kql -f kql.rc -x "run ../script/my_script.kql;"
```

3.2. 볼륨 제어 명령

3.2.1. 볼륨 목록 보기 (SHOW VOLUMES)

독크루저 내에 등록된 볼륨들의 목록과 상태를 보여준다.

- 구문

```
SHOW(LIST) VOLUMES
```

- 예제

```
--------------- kql.rc --------------
max_no_of_sessions = 10
max_no_of_open_files = 128
max_buffer_cache_size = 128 MB
max_working_memory_size = 128 MB
no_of_parallel_task = 4
data_location = "../data"
volume = myvol, ../conf/myvol.rc

kql> show volumes;
+-------------+--------+---------+
| VOLUME NAME | STATUS | VERSION |
+-------------+--------+---------+
|       myvol |     OK | 3.0.2   |
+-------------+--------+---------+
```

```
Total 1 volume.
OK
```

3.2.2. 볼륨 생성 (CREATE VOLUME)

초기 볼륨 생성을 위한 명령어로써, 빈 볼륨을 생성한다.

- 구문

```
CREATE VOLUME [IF NOT EXISTS | BY FORCE] volume_name
[INITIAL SIZE = number {MB|GB|TB}]
```

구성요소	설명
IF NOT EXISTS	기존 볼륨이 존재하지 않을 경우에 한해서만 명령어가 성공적으로 수행되게 하는 선택 지정어
BY FORCE	기존 볼륨이 존재할 경우에도 볼륨을 덮어쓰도록 하는 선택 지정어
volume_name	새로 생성할 볼륨 이름
INITIAL SIZE = *number* {MB|GB|TB}	초기에 미리 할당해 놓을 볼륨 크기

선택 지정어 없이 CREATE VOLUME volume_name 명령을 하면 기본 동작방식은 KQL 설정 파일의 'create_volume_by_force' 설정에 따라 변경이 가능하다.

다음은 'create_volume_by_force'의 설정에 따른 동작 방식이다.

– create_volume_by_force = y (기본)

  CREATE VOLUME *volume_name* = CREATE VOLUME BY FORCE *volume_name*

– create_volume_by_force = n

  CREATE VOLUME *volume_name* = CREATE VOLUME IF NOT EXISTS *volume_name*

- 예제

```
----------------- kql.rc -----------------
max_no_of_sessions = 10
max_no_of_open_files = 128
max_buffer_cache_size = 256 MB
max_working_memory_size= 256 MB
no_of_parallel_task= 4
data_location = "../data"
volume = my_vol, "../conf/myvol.rc"

------------ ../conf/myvol.rc ------------
volume_device= "../volume/myvol"
log_device= "../volume/myvol"
temp_device= "../volume/myvol"

kql> create volume myvol;
OK
```

3.2.3. 볼륨 선택 (USE VOLUME)

작업 대상이 되는 볼륨을 선택한다. 명령을 실행하면 기존에 선택되었던 볼륨은 닫히고 새로운 볼륨이 열리게 된다. 이것은 CREATE VOLUME 이나 SHOW VOLUMES를 제외한 모든 볼륨 및 테이블 제어 명령어의 실행 이전에 실행되어야 한다.

USE VOLUME 명령을 실행하기 위해서는 해당 볼륨이 이미 생성되어 있어야 한다. USE VOLUME 명령어를 입력하여 선택된 볼륨이 있는 상태에서 다른 볼륨을 사용하고자 할 때는 다시 원하는 볼륨에 대해 USE VOLUME *volume_name* 명령을 실행하면 된다.

- 구문

```
USE VOLUME volume_name
```

구성요소	설명
volume_name	선택, 사용할 볼륨의 이름

- 예제

```
kql> use volume myvol;
OK
```

```
kql/my_vol> use volume myvol2;
OK

kql/my_vol2>
```

3.2.4. 볼륨 복사 (COPY VOLUME)

원시 볼륨의 내용을 목적 볼륨으로 복사한다. 복사한 목적 볼륨은 원시 볼륨과 동일한 데이터 값을 가지게 되며, 볼륨 복사 역시 하나의 트랜잭션이므로 복사 실패 시에는 원래의 값으로 복원된다.

이 명령은 빈번한 또는 장시간이 소요되는 데이터 삽입, 삭제, 수정 시에 볼륨 잠김이 일어나 데이터 접근이 방지되는 상황을 회피하고자 할 때 유용하다. 즉, 데이터 삽입, 삭제, 수정을 모두 임시 볼륨에 대해 수행한 후 모든 절차가 성공적으로 종료된 후에만 정식 볼륨에 복사함으로써 볼륨 잠김 시간을 최소화할 수 있다.

- 구문

```
COPY VOLUME //host:port/src_volume_name TO dst_volume_name
 [INCREMENTALLY][USE SNAPSHOT]
```

구성요소	설명
host:port	KQL의 리스너 주소
src_volume_name	원시 볼륨의 이름 원시 볼륨은 지역 볼륨일 수도 있고 다른 독크루저 인스턴스가 관리하는 원격 볼륨일 수도 있다. 지역 볼륨일 경우는 자신의 리스너 주소를 사용하여 복사하고, 원격 볼륨인 경우에는 원격 독크루저의 리스너 주소를 붙여준다.
INCREMENT	두 볼륨을 비교하여 바뀐 부분만 점진적으로 복사하도록 지정
dst_volume_name	목적 볼륨의 이름
USE SNAPSHOT	스냅샷 방식의 증분 복사를 함

- 예제

```
$ cat kql.rc
volume = source_vol, "../conf/source_vol.rc"
volume = target_vol, "../conf/target_vol.rc"

kql> copy volume //192.168.1.32:3274/remote_source_vol
 to target_vol;
OK
```

3.2.5. 볼륨 교환 (SWAP VOLUME)

두 볼륨을 교환한다. 이 명령이 성공적으로 종료되면 volume_name1, volume_name2의 볼륨 데이터 파일 및 로그 파일은 각각 volume_name2, volume_name1로 접근된다. 그러나 물리적 파일의 위치가 바뀌는 것은 아니므로 파일 시스템 상의 파일은 그대로 유지된다.

이 명령은 볼륨 복사보다도 더 즉각적으로 완료된다. 볼륨 복사는 전체 볼륨을 복사하므로 완료까지의 시간도 길어지게 되며 트랜잭션 롤백을 위해 로깅이 수반되므로 파일의 단순 복사보다도 시간이 더 소요된다.

예를 들어, 만약 볼륨의 크기가 수백 MB인데 1 초 이내에 복사를 해야 하는 응용에서는 파일 복사 방식은 사실상 적용하기 어려운 선택이 된다. 이러한 응용의 경우에는 볼륨 교환이 해결책이 될 수도 있다. 볼륨 교환은 두 볼륨이 참조하는 볼륨 데이터 및 로그 파일을 복사하지 않고 바꿔치기 하기 때문에 볼륨의 내용을 즉각적으로 변경할 수 있다. 그러나 파일 시스템상의 물리적 경로를 바꾸지는 않으므로 데이터 및 로그 파일의 경로는 그대로 유지된다.

- 구문

```
SWAP VOLUME volume_name1 volume_name2
```

구성요소	설명
volume_name1 volume_name2	교환할 볼륨의 이름

- 예제

```
$ cat kql.rc
volume = bg_vol, ../conf/bg_vol.rc
volume = fg_vol, /conf/fg_vol.rc

kql> swap volume bg_vol fg_vol;
OK
```

3.2.6. 스냅샷 생성 (CREATE SNAPSHOT FOR VOLUME)

색인된 볼륨의 스냅샷을 생성하는 명령어이다. 스냅샷은 증분 복사 시 사용되는 원시 볼륨의 복사본이 되는 파일의 형식을 말한다. 스냅샷을 생성하면 볼륨 데이터 디렉터리에 스냅샷 파일(*.snf)이 생성된다.

구문

```
CREATE SNAPSHOT FOR VOLUME volume_name
 [RESTART | COMPRESS=[ON|OFF*]]
```

구성요소	설명
volume_name	스냅샷을 생성할 색인 볼륨의 이름
RESTART	증분복사 시작점 설정. 벌크색인하고 벌크복사 이후 증분 복사 시작직전에 수행.
COMPRESS	'ON' 설정 시 스냅샷 파일 압축 저장. (기본값=OFF)

스냅샷 방식의 증분 복사

기존 방식의 증분 복사는 볼륨 잠금을 수반한다. 증분 색인 또한 볼륨 잠금을 수반하므로 둘이 동시에 진행될 경우 서로 간에 레이스가 발생하고 이것이 증분 색인 및 증분 복사의 성능 저하를 야기할 수 있다.

검색 서버와 색인 서버가 각각 하나인 경우 혹은 증분 복사가 거의 없는 경우 스냅샷 파일을 만드는 오버헤드로 인해 오히려 스냅샷 방식 증분 복사가 느릴 수 있다.

따라서 스냅샷 방식 증분 복사가 효력을 발휘하는 상황은 다음과 같다.

- 색인 서버에서 빈번한 INSERT/UPDATE/DELETE 가 발생하는 경우
- 검색 서버의 수가 많은 경우

- 각 검색 서버가 증분 복사를 빈번히 수행하는 경우

이전 방식은 색인 서버에서 여러 대의 검색 서버로 차례로 증분 복사 명령을 내려야 하지만, 스냅샷 방식의 증분 복사는 벌크 색인/벌크 복사 시간을 피해서 각 검색 서버가 주체적으로 증분 복사를 해가도록 구성이 가능하다(병렬적으로 수행).

스냅샷 방식의 증분 복사를 하는 절차는 다음과 같다.

1. 벌크 색인/벌크 복사 후 증분 복사 시작 전 RESTART 명령을 실행하고 증분 복사 시작점을 설정한다.

```
CREATE SNAPSHOT FOR VOLUME volume_name RESTART
```

2. 증분 색인 후 스냅샷을 생성한다.

```
CREATE SNAPSHOT FOR VOLUME volume_name [COMPRESS=[ON|OFF*]]
```

3. 스냅샷을 이용하여 증분 복사를 한다.

```
COPY VOLUME //host:port/src_volume_name TO dst_volume_name
 INCREMENTALLY USE SNAPSHOT
```

스냅샷 파일은 그 크기가 매우 커질 수 있는데 이 경우 디스크 용량이 부족한 경우라면 COMPRESS 옵션을 사용할 수 있다. 단, COMPRESS 옵션은 증분 복사의 성능 저하를 가져올 수도 있다.

스냅샷 파일에 대한 또다른 처리 방법으로 외부 스크립트로 생성된지 일정 시간이 지난 *.snf 파일들을 주기적으로 지워주는 것이 있다(ts00000.snf는 지우면 안됨). 이때, 검색 서버에서 해당 스냅샷들을 이미 다 가져간 것이 확실하다는 것을 보장할 수 있게 시간 간격을 충분히 길게 잡아야 한다. 참고로, CREATE SNAPSHOT ... RESTART 명령을 수행할 때 예전 스냅샷 파일들은 다 지워진다.

3.3. 테이블 제어 명령

3.3.1. 테이블 생성 (CREATE TABLE)

선택한 볼륨에 새로운 테이블을 정의하는 스키마 정의 명령어이다.

3.3.1.1. 테이블/필드 스키마 정의

구문

```
CREATE TABLE table_name ( field_definition )
```

구성요소	설명
table_name	생성하고자 하는 테이블의 이름
field_definition	테이블의 각 필드에 대한 스키마들의 반복으로써 '필드명', '타입', '속성'으로 구성

*table_name*을 정할 때에는 다음과 같은 규칙을 따른다.

- 공백 문자로 분리되지 말아야 한다.
- 30 문자(영숫자 기준)를 넘지 말아야 한다.
- 한글, 영문, 특수 문자 등이 모두 사용될 수 있다.
- 영문의 경우 대소문자가 구분된다.

*field_definition*을 정할 때에는 다음과 같은 규칙을 따른다.

- '필드명'은 테이블명과 같은 명명 규칙을 따른다
- '타입'은 필드값의 데이터 타입이며 STRING, TEXT, DATE, TIME, INT1, INT8, INT16, INT32, INT64, BLOB, AUDIO, VIDEO, IMAGE 등이 있다.

  다음은 필드값을 작성할 때 사용하는 데이터 타입에 대한 설명이다.

파라미터	설명	데이터값의 예
STRING	공백문자로 분리되지 않은 임의의 문자열 (단, 크기는 32 MB 미만, 추출되는 키워드 최대 길이는 2 KB 미만)	Hello-world

파라미터	설명	데이터값의 예
TEXT	임의의 비정형 텍스트 (단, 크기는 32 MB 미만)	기상청은 10 일 남극 상공의 오존 파괴량이 ...
DATE	날짜 데이터 값. CCYYMMDD 포맷 (단, 1900 <= CCYY <= 2033)	19000101
TIME	시간 데이터 값. CCYYMMDDHHMMSS 포맷 (단, 1900 <= CCYY <= 2033)	20001231133559
INT1	1 비트 크기의 부호없는 정수(0 또는 1)	1
INT8	8 비트 크기의 부호없는 십진 정수	100
INT16	16 비트 크기의 부호없는 십진 정수 (단, 천단위 콤마(',') 또는 소수점 등은 불허)	65535
INT32	32 비트 크기의 부호없는 십진 정수 (단, 천단위 콤마(',') 또는 소수점 등은 불허)	1258490610
INT64	64 비트 크기의 부호없는 십진 정수 (단, 천단위 콤마(',') 또는 소수점 등은 불허)	1329239842389
BLOB	임의의 바이너리 값	

- '속성'은 필드가 가지는 성격을 나타내는 지정어이다.

  다음은 각 지정어에 대한 설명이다.

속성	설명
PRIMARY	2 개 이상의 키워드를 값으로 가질 수 있는 경우(TEXT, STRING LIST)를 제외한 데이터 타입에만 쓰이는 속성을 지정한다. 해당 필드가 프라이머리 키가 됨을 지정하며 자동적으로 UNIQUE 속성이 지정된다. 예)

속성	설명
	`fd INT32 PRIMARY NULL`
UNIQUE	해당 필드의 데이터 값이 유일해야 함을 지정한다. 예) `fd INT32 UNIQUE NULL`
NULL	해당 필드의 값이 생략될 수도 있음을 지정한다.(기본값=NULL을 허용 않음).
UNIVERSAL	TEXT 데이터 타입에만 쓰이는 속성을 지정한다. 색인 방식을 n-gram 으로 지정한다.
LINE	TEXT 데이터 타입에만 쓰이는 속성 지정어이다. 해당 필드의 데이터 값이 개행(new line)문자로 분리된 단위로 구분됨을 지정한다.
LIST	STRING 데이터 타입에만 쓰이는 속성 지정어이다. 해당 필드의 데이터 값이 공백문자로 분리된 임의의 STRING 데이터 값들의 연속됨을 지정한다. 공백문자는 ' ', '\t', '\r', '\n' 이 해당된다.
VERBATIM	STRING 데이터 타입에만 쓰이는 속성 지정어이다. 해당 필드의 데이터 값을 있는 그대로 해석한다.
PARA	VERBATIM과 반대. 필드값에서 구분자(' ','\t','\r','\n') 문자를 없애고 대소문자를 구분하지 않는다.
META	숫자/날짜 필드처럼 고정길이를 가지는 데이터 타입에만 쓰이는 속성 지정어이다. 정렬 속도가 향상되는 효과가 있다.

필드 스키마가 여러 번 반복될 때에는 콤마(',')로 구분해준다. 형태소 분석을 하여 키워드를 추출하려는 필드에는 TEXT 속성을 부여한다.

3.3.1.2. 동적 태그

KQL 3.3.0 이상부터 레코드 수입 시 동적으로 필드 값을 수정해서 삽입할 수 있는 기능–동적 태깅 기능이 추가되었다. 통상적인 태깅은 다음과 같이 오프라인에서 별도로 태그를 추출해야 하지만, 동정 태깅 방식을 이용하면 아래와 같이 import할 때 해당 태깅

결과가 볼륨에 저장된다. 즉, 별도의 오프라인 처리를 하지 않고 태그를 동적으로 추출해서 필드 값으로 지정할 수 있다.

- [기존 태깅 방식]

  원시 MD(메타데이터) 파일 => [오프라인 태그 추출] => (원시+태그) MD 파일 생성 => [검색엔진 import] => (원시+태그) 필드로 구성된 레코드

- [동적 태깅]

  원시 MD 파일 => [검색엔진 import] => (원시+태그) 필드로 구성된 레코드

태그 추출 모듈은 KQL에 내재화되어 있는 것이거나 내재화되어 있지 않은 것일 수도 있다. KQL에 내재화된 태그의 대표적인 것으로는 키워드, 개체명, 오디오 핑거프린트, 이미지 핑거프린트 추출 등 각종 내용기반 정보 추출 모듈들이 해당된다. 내재화되지 않은 경우는 사용자 정의 함수를 이용하여 모듈을 별도로 생성하여 적용할 수 있다.

태그 추출 모듈을 구동하려면 다음과 같이 해야 한다.

먼저, 테이블 생성 DDL에 실제 import되는 필드(다음 예에서는 TT, BD) 외에 태그 필드(다음 예에서는 FD)를 정의하였다. 태그 값이 생성되면 일반 필드와 똑같이 값을 참조할 수 있으므로 태그 필드의 타입은 태그 값에 어떤 색인을 생성할 지에 전적으로 달려 있다. 단순히 리터럴 상태로 참조만 할 것이라면 아무 타입이나 관계 없다. 필드 정의 구문 다음에 tag constraint를 지정한다. 이 tag constraint에는 내재화 태그와 사용자 정의 태그가 올 수 있다.

```
CREATE TABLE tab (
    TT TEXT NULL,
    BD TEXT NULL,
    FD STRING NULL,

TAG
    FD = <내부태깅함수> 또는 (USERDEF <사용자정의 DLL 파일명>)
);
```

다음은 내재화(built-in) 태그들에 대한 설명이다.

일반 태그

FD = DATE()

레코드 저장 시간. DATE 타입과 같은 포맷 (예. 20070618)

FD = TIME()

레코드 저장 시간. TIME 타입과 같은 포맷 (예. 20070618115959)

FD = SIZE()

레코드 저장 시간. TIME 타입과 같은 포맷 (예. 20070618115959)

FD = FILE_SIZE(PT)

특정 경로(PT)의 파일의 크기.

FD = MD5(BD)

MD5 모듈을 통해 특정 필드(BD)에 대해 추출된 요약 값.

FD = KEYWORD(BD)

형태소 분석기에 의해 특정 필드(BD)에서 추출된 키워드 정보.

FD = KEYWORD_LIST(BD)

형태소 분석기에 의해 특정 필드(BD)에서 추출된 키워드 목록.

FD = NAME(BD)

NER 모듈에 의해 특정 필드(BD)에서 추출된 고유명사 정보.

FD = NAME_LIST(BD)

NER 모듈에 의해 특정 필드(BD)에서 추출된 고유명사 목록.

TP = DOWNLOAD_UF(OU)

특정 URL(OU)을 로컬에 다운로드하여 임시 경로(TP)에 해당 파일을 저장. 본 태그를 이용하여 URL 형태로 되어있는 이미지/동영상/음악 파일 등을 다운로드할 수 있다.

FD = TEXT_FILTER_BB(BD, "HTML")

HTML 문서의 필터링 및 < 와 > 사이에 있는 태그 필터링. html-conf.rc라는 설정 파일을 참조로 처리하게 된다. 설정 파일은 'bin/' 디렉터리에 html-conf.rc 파일로 생성하여 설정한다.

```
예)html-conf.rc
-----------------------------------------------------------------
```

```
output_charset = utf8 // 필터링 후 색인 될 데이터가 UTF-8 인 경우
 (기본은 euckr)
(utf8, UTF8, utf-8, UTF-8 도 인식, 그 외에는 모두 euckr 로 인식)
filtering_lt_gt_html_tag = y // &lt; ~~ &gt; 내의 내용을 제거할 경우
 (기본은 내용 추출)
(y, yes, 1 도 인식 그 외에는 &lt; 는 '<'로 &gt; 는 '>' 바꿔서 추출)
exclude_tag = "IMG-CAPTION" // 제외할 태그 이름
exclude_tag = "COMMENT"
exclude_tag = "LIBRARYLIST"
replace_tag = "TH", "|" // "치환할 태그 이름", "치환될 문자열".
replace_tag = "TD", "|"
-----------------------------------------------------------------
```

FD = MASK_PRIVATE_INFO(BD)

특정 필드(BD) 값에 존재하는 개인정보(주민번호, 계좌번호, 이메일, 전화번호)를 노출되지 않게 '*' 기호로 매스킹시킨다.

예제) BD: 나의 전화번호는 011-435-3456 이다.
=>FD: 나의 전화번호는 ********** 이다.

PT = REMOVE(PT)

로컬에 있는 특정 경로명(PT)의 파일을 삭제.

FD = NIL()

특정 필드(FD) 값 초기화. 본 태그를 실행하게 되면 해당 필드 값은 " "이 된다.

시맨틱 태그

FD = CATEGORY(BD)

문단/단락 텍스트 특징(시맨틱 태그) 추출(3.6이후 'PARAGRAPH_FEATURE'로 변경)

FD = THEME_Q(BD)

즉답 특징(시맨틱 태그) 추출(3.6이후 'QUESTION_FEATURE'로 변경)

FD = THEME_Q_LIST(BD)

즉답 특징(시맨틱 태그) 추출(위치 정보 없음)(3.6이후 'QUESTION_FEATURE_LIST'로 변경)

FD = THEME(BD)

문장 특징(시맨틱 태그) 추출(3.6이후 'SENTENCE_FEATURE'로 변경)

FD = THEME_LIST(BD)

문장 특징(시맨틱 태그) 추출(위치 정보 없음)(3.6이후 'SENTENCE_FEATURE_LIST'로 변경)

이미지 태그

MD = IMAGE_METADATA_FB(PT)

특정 경로(PT)에 있는 이미지 파일의 메타데이터(헤더/EXIF 정보 등)를 일괄 추출하여 출력 필드(MD)에 저장. (내부적으로 정해진 구조체 사용)

FD = IMAGE_EXIFTAGGED (MD)

IMAGE_METADATA_FB 출력(MD) 필드에서 EXIF 존재 유무 플래그(존재: 1, 그외: 0 또는 NULL) 추출.

FD = IMAGE_FORMAT_NUM (MD)

IMAGE_METADATA_FB 출력(MD) 필드에서 이미지 포맷 코드 번호 추출. 내부적으로 정해진 이미지 포맷 코드 번호는 다음 표를 참고하면 된다.

이미지 포맷	코드 번호
etc	0
BMP	1
JPG	2
GIF	3
TIFF	4
PCX	5
PNG	6
SWF	7
ICO	10
IFF	11
CUR	12
TGA	13

이미지 포맷	코드 번호
PPM	14
PSD	15

FD = IMAGE_ANIMATED (MD)

IMAGE_METADATA_FB 출력(MD) 필드에서 이미지 애니메이션 플래그 추출. 이미지가 애니메이션인 경우에는 '1', 그 외는 '0' 또는 NULL을 출력함.

FD = IMAGE_WIDTH (MD)

IMAGE_METADATA_FB 출력(MD) 필드에서 이미지의 가로 길이(픽셀 단위) 추출.

FD = IMAGE_HEIGHT (MD)

IMAGE_METADATA_FB 출력(MD) 필드에서 이미지의 세로 길이(픽셀 단위) 추출.

FD = IMAGE_COMMENTS (MD)

IMAGE_METADATA_FB 출력(MD) 필드에서 EXIF 정보 중 주석 정보 추출.

FD = IMAGE_CAMERAMAKER (MD)

IMAGE_METADATA_FB 출력(MD) 필드에서 EXIF 정보 중 카메라 제조사 정보 추출.

FD = IMAGE_CAMERAMODEL (MD)

IMAGE_METADATA_FB 출력(MD) 필드에서 EXIF 정보 중 카메라 모델 정보 추출.

FD = IMAGE_CAMERAMAKER_GROUP (MD)

IMAGE_METADATA_FB 출력(MD) 필드에서 EXIF 정보 중 카메라 제조사의 대표 명칭 또는 그룹핑 시킨 정보 추출. IMAGE_CAMERAMAKER 에서 추출한 정보는 사진 파일의 EXIF 정보 값 그대로 추출하는 반면, 본 태그 함수는 해당 제조사 정보를 일관된 명칭으로 정제시킨 결과 이다. 같은 제조사의 카메라인 경우에도 제조사 명칭이 다양한 경우가 있기 때문에 정제시킬 필요가 있다. 정제된 제조사 그룹명은 다음 태그 함수인 IMAGE_CAMERAMAKER_NUM의 제조사에 따른 코드 번호 표를 참고하면 된다.

FD = IMAGE_CAMERAMAKER_NUM (MD)

IMAGE_METADATA_FB 출력(MD) 필드에서 EXIF 정보 중 카메라 제조사의 대표 명칭 또는 그룹핑 시킨 제조사 코드번호 추출. 즉, 카메라 제조사 정보를 일관된 명칭으로 정제한 후 코드 번호를 할당한 것이다. 정제된 제조사 코드 번호는 다음 표와 같다.

카메라 제조사	코드 번호
etc	0
NIKON	1
MINOLTA	2
SAMSUNG	3
SONY	4
OLYMPUS	5
CASIO	6
CANON	7
KONICA	8
KODAK	9
PANASONIC	10
PENTAX	11
FUJIFILM	12
AGFA	20
AIPTEK	21
ASHAMPOO	22
BENQ	23
CONCORD	24
EPSON	25
FUJITSU	26
HASSELBLAD	27
HITACHI	28
HP	29

카메라 제조사	코드 번호
JENOPTIK	30
JVC	31
KTFT	32
KYOCERA	33
LEICA	34
LG	35
MICROTEK	36
MOTOROLA	37
NOKIA	38
NORITSU	39
NTTDOCOMO	40
PANTECH	41
POLAROID	42
PREMIER	43
RICOH	44
SANYO	45
SHARP	46
SIGMA	47
SIPIX	48
TOSHIBA	49
UMAX	50
VIVITAR	51

FD = IMAGE_SHOOTINGTIME (MD)

IMAGE_METADATA_FB 출력(MD) 필드에서 EXIF 정보 중 촬영시간 정보 (CCYYMMDDHHMMSS) 추출.

FD = IMAGE_FLASHUSED (MD)

IMAGE_METADATA_FB 출력(MD) 필드에서 EXIF 정보 중 플래시 사용 정보 추출.

필드 값에 따른 플래시 사용 의미는 다음의 표를 참고한다.

필드값	의미
5	Strobe light not detected
7	Strobe light detected
9	manual
13	manual, return light not detected
15	manual, return light detected
24	auto
25	auto
29	auto, return light not detected
31	auto, return light detected
65	red eye reduction mode
69	red eye reduction mode return light not detected
71	red eye reduction mode return light detected
73	manual, red eye reduction mode
77	manual, red eye reduction mode, return light not detected
79	red eye reduction mode, return light detected
89	auto, red eye reduction mode
93	auto, red eye reduction mode, return light not detected
95	auto, red eye reduction mode, return light detected
others	flash not used

FD = IMAGE_FOCALLENGTH (MD)

IMAGE_METADATA_FB 출력(MD) 필드에서 EXIF 정보 중 렌즈 초점 길이 정보 추출. 정수형과 호환을 위해 실제 값(단위: mm)의 100 배로 추출한다. (값의 범위: 1~999,999)

FD = IMAGE_FOCALLENGTH35MMEQUIV (MD)

IMAGE_METADATA_FB 출력(MD) 필드에서 EXIF 정보 중 35 mm 기준 초점 길이 정보 추출

FD = IMAGE_DIGITALZOOMRATIO (MD)

IMAGE_METADATA_FB 출력(MD) 필드에서 EXIF 정보 중 줌 비율 정보 추출. (1: not used)

FD = IMAGE_EXPOSURETIME (MD)

IMAGE_METADATA_FB 출력(MD) 필드에서 EXIF 정보 중 노출 시간 정보 추출. 정수형과 호환을 위해 실제 값(단위: sec)의 10,000 배로 추출한다. (값의 범위: 0~36,000,000)

FD = IMAGE_APERTUREFNUMBER (MD)

IMAGE_METADATA_FB 출력(MD) 필드에서 EXIF 정보 중 조리개 정보 추출.

FD = IMAGE_DISTANCE (MD)

IMAGE_METADATA_FB 출력(MD) 필드에서 EXIF 정보 중 초점 거리 정보 추출.

FD = IMAGE_CCDWIDTH (MD)

IMAGE_METADATA_FB 출력(MD) 필드에서 EXIF 정보 중 CCD 크기 정보 추출.

FD = IMAGE_ISOEQUIVALENT (MD)

IMAGE_METADATA_FB 출력(MD) 필드에서 EXIF 정보 중 ISO 정보 추출.

FD = IMAGE_WHITEBALANCE (MD)

IMAGE_METADATA_FB 출력(MD) 필드에서 EXIF 정보 중 화이트 밸런스 정보 추출. (0: auto, 1: manual)

FD = IMAGE_METERINGMODE (MD)

IMAGE_METADATA_FB 출력(MD) 필드에서 EXIF 정보 중 측광모드 정보 추출. (2: center weight, 3: spot, 5: matrix)

FD = IMAGE_EXPOSUREPROGRAM (MD)

IMAGE_METADATA_FB 출력(MD) 필드에서 EXIF 정보 중 노출방식 정보 추출. (1: manual, 2: program (auto), 3: aperture priority (semi-auto), 4: shutter priority (semi-auto), 5: creative program (based towards depth of field), 6: action program (based towards fast shutter speed), 7: portrait mode, 8: landscape mode)

FD = IMAGE_EXPOSUREMODE (MD)

IMAGE_METADATA_FB 출력(MD) 필드에서 EXIF 정보 중 노출모드 정보 추출. (0: auto, 1: manual, 2: auto bracketing)

FD = IMAGE_LIGHTSOURCE (MD)

IMAGE_METADATA_FB 출력(MD) 필드에서 EXIF 정보 중 광원 정보 추출. (1: daylight, 2: Fluorescent, 3: Incandescent, 4: flash, 9: fine weather, 11: shade)

FD = IMAGE_GPSLATITUDEREF (MD)

IMAGE_METADATA_FB 출력(MD) 필드에서 EXIF 정보 중 GPS 위도 방향 정보 추출. (N: north, S: south, Other: reserved)

FD = IMAGE_GPSLATITUDE (MD)

IMAGE_METADATA_FB 출력(MD) 필드에서 EXIF 정보 중 GPS 위도 정보 추출. 실제 값의 100 만 배로 추출. (0~90,000,000)

FD = IMAGE_GPSLONGITUDEREF (MD)

IMAGE_METADATA_FB 출력(MD) 필드에서 EXIF 정보 중 GPS 경도 방향 정보 추출. (E: east, W: west, Other: reserved)

FD = IMAGE_GPSLONGITUDE (MD)

IMAGE_METADATA_FB 출력(MD) 필드에서 EXIF 정보 중 GPS 경도 정보 추출. 실제 값의 100 만 배로 추출. (0~180,000,000)

FD = IMAGE_GPSALTITUDEREF (MD)

IMAGE_METADATA_FB 출력(MD) 필드에서 EXIF 정보 중 GPS 고도 방향 정보 추출. (+: above sea level, -: below sea level, Other: reserved)

FD = IMAGE_GPSALTITUDE (MD)

IMAGE_METADATA_FB 출력(MD) 필드에서 EXIF 정보 중 GPS 고도 정보 추출. 실제 값(단위: meter)의 100 배로 추출. (단위: centimeter)

DF = IMAGE_FILTER_FF(PT)

특정 경로(PT)에 있는 이미지 파일을 디코딩하여 임시 파일 경로(DF)에 저장. 본 태그 함수를 실행하기 위해서는 멀티미디어 필터 데몬이 필요하다.

FD = IMAGE_NOBODY_FB(DF)

IMAGE_FILTER_FF에서 디코딩한 파일(DF)로부터 비인물 플래그 값 추출. (1: 비인물, 그외: 0/NULL)

MD = IMAGE_FACEDETECT_FB(DF)

IMAGE_FILTER_FF에서 디코딩한 파일(DF)로부터 인물 정보 일괄 추출.

FD = IMAGE_FACECOUNT(MD)

IMAGE_FACEDETECT_FB에서 추출한 인물 정보(MD)로부터 검출된 얼굴 개수 추출.

FD = IMAGE_FACEPROPORTIONONE(MD)

IMAGE_FACEDETECT_FB에서 추출한 인물 정보(MD)로부터 검출된 얼굴 중 가장 큰 얼굴의 비중 값 추출. (비중: 얼굴 영역/이미지 전체 영역 X 100)

FD = IMAGE_FACEPROPORTIONALL(MD)

IMAGE_FACEDETECT_FB에서 추출한 인물 정보(MD)로부터 검출된 얼굴의 모든 비중 값을 리스트 형식으로 추출.

FD = IMAGE_FACELOCATIONONE(MD)

IMAGE_FACEDETECT_FB에서 추출한 인물 정보(MD)로부터 검출된 얼굴 중 가장 큰 얼굴의 위치 정보 추출. (위치: X1 Y1 X2 Y2 (좌측 상단 꼭지점, 우측 하단 꼭지점, 얼굴 영역을 사각형으로 가정))

FD = IMAGE_FACELOCATIONALL(MD)

IMAGE_FACEDETECT_FB에서 추출한 인물 정보(MD)로부터 검출된 얼굴 모두의 위치 정보 추출.

BF = IMAGE_CONTOURFILTER_FB(DF)

IMAGE_ FILTER_FB의 디코딩 파일(DF)에서 형상 정보를 일괄 추출.

BF = IMAGE_CONTOURFILTER_COMBI_BB(MD)

IMAGE_FACEDETECT_FB에서 추출한 인물 정보(MD)와 MD 필드 내부의 저장된 디코딩된 파일 경로를 참조로 이미지의 형상 및 유사이미지 정보를 일괄 추출

FD = IMAGE_CONTOURID(BF)

IMAGE_CONTOURFILTER_FB 또는 IMAGE_CONTOURFILTER_COMBI_BB에서 일괄 추출된 형상 정보(BF)에서 모양의 ID 번호를 추출.

모양에 따른 ID 번호 값은 다음의 표를 참고한다.

모양	모양 ID
1	ARROW
2	CIRCLE
3	CYLINDER
4	LADDER
5	ELLIPSE
6	PARALLELOGRAM
7	SQUARE
8	STAR
9	RECTANGLE
10	TRIANGLE

FD = IMAGE_CONTOURPROPORTION(BF)

IMAGE_CONTOURFILTER_FB 또는 IMAGE_CONTOURFILTER_COMBI_BB에서 일괄 추출된 형상 정보(BF)에서 검출된 모양의 비중 값 추출. (비중: 검출 영역/이미지 전체 영역 X 100)

FD = IMAGE_CONTOURREGION(BF)

IMAGE_CONTOURFILTER_FB 또는 IMAGE_CONTOURFILTER_COMBI_BB에서 일괄 추출된 형상 정보(BF)에서 검출된 모양의 위치 정보 추출. (위치: X Y DX DY (좌측 상단 시작점, 가로 넓이, 세로 높이))

MD = IMAGE_COLORFILTER_FB(DF)

IMAGE_ FILTER_FB의 디코딩 파일(DF)에서 컬러 정보를 일괄 추출.

FD = IMAGE_COLORCOUNT(MD)

IMAGE_COLORFILTER_FB에서 일괄 추출된 컬러 정보(MD)로부터 컬러 개수 추출.

FD = IMAGE_COLORID_GLOBAL(MD)

IMAGE_COLORFILTER_FB에서 일괄 추출된 컬러 정보(MD)로부터 대표 컬러 ID 추출. (대표 컬러: 비중이 가장 큰 컬러).

사용되는 컬러명과 ID 정보는 다음의 표와 같다.

컬러ID	헥사코드	컬러이름
1	#1f1f1f	검정
2	#5e5e5e	어두운회색
3	#9e9e9e	회색
4	#dcdcdc	밝은회색
5	#672525	갈색
6	#961d14	벽돌
7	#cd3333	빨강
8	#dc9c9c	밝은빨강
9	#754f14	어두운황토
10	#b5814a	황토
11	#d88a21	주황
12	#dcbb97	밝은주황
13	#636320	어두운올리브
14	#afaf4a	올리브

컬러ID	헥사코드	컬러이름
15	#d9d03f	노랑
16	#dfd99b	밝은노랑
17,21	#446720	어두운녹색
18,22	#549217	바다녹색
19,23	#7dd229	연두
20,24	#bbdc97	밝은연두
25	#104225	어두운초록
26	#319366	숲녹색
27	#1fcf7a	라임
28	#97dcb9	밝은라임
29	#115a59	어두운청록
30	#2b9495	청록
31	#5dbcbd	하늘
32	#97d8d7	밝은하늘
33	#324a8e	어두운파랑
34	#4a81b5	다져블루
35	#78a1c6	스틸블루
36	#97bbdc	밝은스틸블루
37	#1c155c	미드나잇블루
38	#241390	네이비
39	#3453c5	파랑
40	#9b9cdc	밝은파랑
41	#3a137a	어두운보라
42	#671ab7	진보라
43	#8968cd	보라
44	#b997dc	밝은연보라
45	#731a6a	어두운제비꽃
46	#ae4aaf	제비꽃

컬러ID	헥사코드	컬러이름
47	#d15fee	분홍
48	#d797d8	밝은분홍
49	#5e043b	어두운자주
50	#b54a81	자주
51	#ee30a7	진분홍
52	#dc97bb	밝은진분홍

FD = IMAGE_COLORID_LEFT(MD)

IMAGE_COLORFILTER_FB에서 일괄 추출된 컬러 정보(MD)로부터 좌측 영역의 대표 컬러 ID 추출. (대표 컬러: 해당 영역에서 비중이 가장 큰 컬러)

FD = IMAGE_COLORID_RIGHT(MD)

IMAGE_COLORFILTER_FB에서 일괄 추출된 컬러 정보(MD)로부터 우측 영역의 대표 컬러 ID 추출. (대표 컬러: 해당 영역에서 비중이 가장 큰 컬러)

FD = IMAGE_COLORID_TOP(MD)

IMAGE_COLORFILTER_FB에서 일괄 추출된 컬러 정보(MD)로부터 위 영역의 대표 컬러 ID 추출. (대표 컬러: 해당 영역에서 비중이 가장 큰 컬러)

FD = IMAGE_COLORID_BOTTOM(MD)

IMAGE_COLORFILTER_FB에서 일괄 추출된 컬러 정보(MD)로부터 다음 영역의 대표 컬러 ID 추출. (대표 컬러: 해당 영역에서 비중이 가장 큰 컬러)

FD = IMAGE_COLORID_CENTER(MD)

IMAGE_COLORFILTER_FB에서 일괄 추출된 컬러 정보(MD)로부터 중심 영역의 대표 컬러 ID 추출. (대표 컬러: 해당 영역에서 비중이 가장 큰 컬러)

FD = IMAGE_COLORID_GLOBALS(MD)

IMAGE_COLORFILTER_FB에서 일괄 추출된 컬러 정보(MD)로부터 컬러 ID 리스트 추출. (최소 비중이 5% 이상인 컬러 중 최대 12 개 까지 추출)

FD = IMAGE_COLORPROPORTION_GLOBAL(MD)

IMAGE_COLORFILTER_FB에서 일괄 추출된 컬러 정보(MD)로부터 대표 컬러 비중 추출.

FD = IMAGE_COLORPROPORTION_LEFT(MD)

IMAGE_COLORFILTER_FB에서 일괄 추출된 컬러 정보(MD)로부터 좌측 영역의 대표 컬러 비중 추출.

FD = IMAGE_COLORPROPORTION_RIGHT(MD)

IMAGE_COLORFILTER_FB에서 일괄 추출된 컬러 정보(MD)로부터 우측 영역의 대표 컬러 비중 추출.

FD = IMAGE_COLORPROPORTION_TOP(MD)

IMAGE_COLORFILTER_FB에서 일괄 추출된 컬러 정보(MD)로부터 위 영역의 대표 컬러 비중 추출.

FD = IMAGE_COLORPROPORTION_BOTTOM(MD)

IMAGE_COLORFILTER_FB에서 일괄 추출된 컬러 정보(MD)로부터 다음 영역의 대표 컬러 비중 추출.

FD = IMAGE_COLORPROPORTION_CENTER(MD)

IMAGE_COLORFILTER_FB에서 일괄 추출된 컬러 정보(MD)로부터 중심 영역의 대표 컬러 비중 추출.

FD = IMAGE_COLORPROPORTION_GLOBALS(MD)

IMAGE_COLORFILTER_FB에서 일괄 추출된 컬러 정보(MD)로부터 컬러 비중 리스트 추출. (최소 비중이 5% 이상인 컬러 중 최대 12 개까지 추출)

FD = IMAGE_COLORBLACKANDWHITE(MD)

IMAGE_COLORFILTER_FB에서 일괄 추출된 컬러 정보(MD)로부터 흑백 이미지 플래그 값 추출. (흑백: 1, 그외: 0/NULL)

동영상/오디오 태그

MD = VIDEO_METADATA_FB(PT)

특정 경로(PT)에 있는 동영상 파일의 메타데이터(헤더 정보 등)를 일괄 추출하여 출력 필드(MD)에 저장. (내부적으로 정해진 구조체 사용)

FD = VIDEO_WIDTH(MD)

VIDEO_METADATA_FB에서 추출된 동영상의 메타 정보(MD)로부터 동영상 프레임의 가로 길이 추출.

FD = VIDEO_HEIGHT(MD)

VIDEO_METADATA_FB에서 추출된 동영상의 메타 정보(MD)로부터 동영상 프레임의 세로 길이 추출.

FD = VIDEO_DURATION(MD)

VIDEO_METADATA_FB에서 추출된 동영상의 메타 정보(MD)로부터 동영상 재생 길이 추출. (단위: SECOND)

FD = VIDEO_DURATIONV(MD)

VIDEO_METADATA_FB에서 추출된 동영상의 메타 정보(MD)로부터 동영상의 비디오 스트림의 재생 길이 추출. (단위: SECOND)

FD = VIDEO_DURATIONA(MD)

VIDEO_METADATA_FB에서 추출된 동영상의 메타 정보(MD)로부터 동영상의 오디오 스트림의 재생 길이 추출. (단위: SECOND)

FD = VIDEO_BITRATE(MD)

VIDEO_METADATA_FB에서 추출된 동영상의 메타 정보(MD)로부터 동영상의 비트율 추출.

FD = VIDEO_FRAMERATE(MD)

VIDEO_METADATA_FB에서 추출된 동영상의 메타 정보(MD)로부터 동영상의 프레임 비율 추출.

DF = VIDEO_FILTER_FF(PATH)

특정 경로(PT)에 있는 동영상 파일을 디코딩하여 임시 파일 경로(DF)에 저장. 본 태그 함수를 실행하기 위해서는 멀티미디어 필터 데몬이 필요하다.

FD = VIDEO_FINGERPRINT(DF)

VIDEO_FILTER_FF에서 디코딩한 파일(DF)에서 비디오 핑거프린트 정보 추출.

FD = VIDEO_DURATION_EX(DF)

VIDEO_FILTER_FF에서 디코딩한 파일(DF)에서 비디오 재생 길이 추출. (단위: SECOND)

DF = AUDIO_FILTER_FF(PT)

특정 경로(PT)에 있는 동영상/오디오(음악) 파일을 디코딩하여 임시 파일 경로(DF)에 저장. 본 태그 함수를 실행하기 위해서는 멀티미디어 필터 데몬이 필요하다.

FD = AUDIO_FINGERPRINT(DF)

AUDIO_FILTER_FF에서 디코딩한 파일(DF)로부터 오디오 핑거프린트 정보 추출.

FD = AUDIO_DURATION_EX(DF)

AUDIO_FILTER_FF에서 디코딩한 파일(DF)로부터 오디오 재생 길이 추출. (단위: SECOND)

MD = AUDIO_METADATA_FB(DF)

특정 경로(PT)에 있는 오디오(음악) 파일의 메타데이터(헤더/TAG 정보 등)를 일괄 추출하여 출력 필드(MD)에 저장. (내부적으로 정해진 구조체 사용)

FD = AUDIO_DURATION(MD)

AUDIO_METADATA_FB에서 일괄 추출한 오디오(음악) 파일의 메타데이터(MD)로부터 오디오 재생 길이 추출. (단위: SECOND)

FD = AUDIO_BITRATE(MD)

AUDIO_METADATA_FB에서 일괄 추출한 오디오(음악) 파일의 메타데이터(MD)로부터 오디오 비트율 추출.

FD = AUDIO_SAMPLINGRATE(MD)

AUDIO_METADATA_FB에서 일괄 추출한 오디오(음악) 파일의 메타데이터(MD)로부터 오디오 샘플링 비율 추출.

FD = AUDIO_CHANNEL(MD)

AUDIO_METADATA_FB에서 일괄 추출한 오디오(음악) 파일의 메타데이터(MD)로부터 오디오 채널 수 추출.

사용자 정의 태그

USERDEF 태그

USERDEF 태그는 시스템에서 기본적으로 제공하는 내재화(built-in) 태그 이외에 특정 응용에 특화된 태깅 동작을 수행할 경우에 사용한다. 사용자가 직접 태그 함수의 입력/출력을 정의하고, 정의된 동작을 수행하는 동적 공유 라이브러리(shared objec module, 이하 'so 모듈')를 작성해서 스키마를 구성할 수 있다.

일반적인 태그 스키마 구문은 다음과 같다.

```
fieldname = USERDEF user-mod.so[(rc-file)]
```

*fieldname*은 USERDEF 태그의 결과값을 저장할 필드 이름이고, *user-mod.so*는 so 모듈의 이름이다. *rc-file*은 so 모듈에서 사용하는 환경설정 파일 경로이다(생략 가능).

*user-mod.so*와 *rc-file* 파일 경로는 시스템의 data 디렉터리 설정 경로에 대한 상대 경로값이 된다.

USERDEF 태그는 한 번 로드가 되면 엔진이 재기동될 때까지는 계속 메모리에 떠 있게 된다. so 모듈을 교체하고 싶으면 엔진을 내렸다 올리거나 아래의 KQL 명령 구문을 이용해서 기존 메모리에 떠 있는 so 모듈을 내리고 교체하면 된다(3.5.3 r12577이상부터 사용 가능).

```
UNLOAD USERDEF TAG TO table_name user-mod.so
```

반드시 user-mod.so 파일을 교체하기 전에 unload를 시켜야한다.

사용자 정의 동적 라이브러리 구성

아래에서 제공해야 하는 함수들은 이름이 예약되어 있으며 임의로 바꿀 수 없다.

- ```
  int MyConstructor(char msg[], char arg[])
  ```

  성공하면 0, 실패하면 -1을 리턴한다. 초기화 함수로서 최초 로딩때 엔진에서 한 번 기동해 주는 '부트스트랩 함수'이다. 이때 *arg* 인자는 USERDEF 태그 스키마에 명시된 *rc-file* 경로이다.

- ```
  void Destructor()
  ```

  USERDEF 모듈 자원을 해제한다. 엔진 종료 시 최후에 한번 기동된다.

- ```
  void MyAction(char* field_name[], char* field_data[],
  int field_size[], int field_count,
  int curr_working_field_no,
  char buf[], int buf_max, int* p_buf_len);
  ```

  필드 데이터 값을 받아서 결과 태그 값을 리턴해 주는 함수이다. 저장/색인 단계 중 저장 단계에서 각각의 문서 저장 시 호출되므로 될 수 있으면 경량으로 작성하는 것이 좋다.

  인자들이 갖는 의미는 다음의 표와 같다.

인자	설명
field_name	테이블 스키마에 명시된 전체 필드 이름 목록이다. 이때 *field_name*은 ""이 될 수 있는데, 이것은 이 USERDEF 태깅을 수행할 때 해당 필드에 대한 값이 바인딩되어 있지 않음을 의미한다. 즉, 제대로 값이 설정된 필드들의 이름만 ""이 아닌 값으로 넘어온다.
field_data	전체 필드 값 목록이다. *field_name* 인자와 마찬가지로 값이 바인딩되어 있지 않은 원소는 ""로 세팅된다.
field_size	바인딩되어 넘어온 각각의 필드 값 길이이다.
field_count	필드 리스트의 개수이다.
curr_working_field_no	스키마 상에서 USERDEF 태그 수행의 결과 값이 저장될 필드의 번호이다.
buf	USERDEF 태그 수행 결과를 저장할 버퍼이다.
buf_max	출력 버퍼의 크기이다.
p_buf_len	USERDEF 태그 수행 결과 값의 길이로서 값에 따라 다음과 같은 특수한 의미를 갖는다. – buf_max: 결과 저장 버퍼가 충분히 크지 않으므로 최소한 *p_buf_len만큼 시스템에서 다시 할당해 주도록 한다. – 0: 출력 필드를 ""로 업데이트한다. – -1: USERDEF를 수행하더라도 출력 필드를 업데이트시키지 않는다.
  ```

3.3.1.3. 일반 테이블 제약

테이블 제약(constraint)을 통해 프라이머리 키, 필드값 포맷, 필드값 참조 규약 등을 지정할 수 있다. 테이블 제약은 테이블에 필요한 모든 필드의 이름과 타입, 속성을 정의한 후에 CONATRAINT 예약어를 이용하여 추가한다. 테이블 제약은 테이블이 가진 필드에 대해 한 개 이상 설정할 수 있다. 동일한 필드가 두 개 이상의 제약을 가질 수 있다.

다음은 필드 제약에 대한 설명이다.

제약 유형	설명
PRIMARY KEY	하나 이상의 필드로 이루어진 프라이머리키 선언 CONSTRAINT *const_name* PRIMARY KEY (*field_name*, ...)
REFERENCE TYPE	필드값이 참조값임을 명시 CONSTRAINT *const_name* REFERENCE TYPE (*field_name*) = { [LOCAL [(*path_formation_rule*)]] | [INTERNAL [*src_field_name*]]}
VALUE FORMAT	필드값이 특정 포맷으로 인코딩되어 있음 CONSTRAINT *const_name* VALUE FORMAT (*field_name*) = [AUTO | HTML | HWP | DOC | XLS | PPT | PDF | HWD | MP3]

3.3.1.4. 테이블 생성 예제

- 일반 테이블 정의

  다음은 일반적인 테이블 정의의 예이다. 예제는 명령행의 인터프리터 방식으로 테이블을 정의하고 있으나 실제 테이블 정의는 전체 과정을 스크립트로 작성한 뒤, 'RUN' 명령어를 이용하거나 비대화형 KQL 인터프리터 명령 실행인 –x 옵션을 이용하여 실행하도록 한다.

```
kql/myvol> create table tbl(str1 STRING PRIMARY, tt TEXT, bd TEXT,
tm DATE, cl INT16);
OK
```

- 복합키를 가지는 테이블 정의

테이블 제약을 이용하면 좀더 다양한 표현력을 가지도록 스키마를 정의할 수 있다.

다음은 하나 이상의 필드의 조합으로 구성되는 복합 키를 가지는 테이블을 PRIMARY KEY 제약을 이용하여 정의한 예이다.

```
kql/myvol> create table tbl(
    -> f1 STRING, f2 INT32, f3 TEXT, f4 TEXT,
    -> CONSTRAINT c1 PRIMARY KEY (f1, f2) );
OK
```

단, 볼륨 레코드 개수가 아주 많으면 복합 PRIMARY KEY 제약은 성능을 떨어뜨릴 수 있으며, 필드에 UNIQUE 속성이 지정이 안 된 경우는 중복 체크에도 한계가 있으므로 사용에 주의해야 한다. 대용량 환경에서는 복합키로 구성하려는 두 개의 키를 이어 붙여 별도의 필드로 구성하고 해당 키에 PRIMARY 속성을 지정하는 것이 좋다.

- 필드값이 경로인 경우의 테이블 정의

  원본 레코드의 필드값이 데이터가 아니고 실제 데이터 값의 위치를 나타내는 경로(예: 파일 경로)일 때 REFERENCE TYPE 제약을 이용하여 테이블을 정의한다.

  필터링 가능한 포맷은 HTML, HWP, DOC, XLS, PPT, PDF, HWD, MP3, TXT, RTF로 지정할 수 있으며 포맷을 미리 알 수 없는 경우는 'AUTO'로 설정 가능하다.

  다음 예제에서 f3 필드는 파일 내용에 해당하는 부분이다. f3에 해당하는 원본 데이터가 파일이 아닌 파일 경로 정보(절대경로)를 가지고 있는 경우 테이블에 대한 정의이다 f3 필드가 파일 경로를 나타내는 경우 검색엔진은 파일을 색인하여 해당 필드의 데이터로 파일 내용을 저장한다.

  ```
  kql/myvol> create table tbl(
      -> f1 STRING, f2 INT32, f3 TEXT, f4 TEXT,
      -> CONSTRAINT c1 TAG(f3) = TEXT_FILTER_FB(f3, "AUTO") );
  OK
  ```

- 다른 필드 값을 참조하는 경우 테이블 정의

  REFERENCE TYPE 제약 활용의 또다른 예로, 동일한 필드 데이타에 대해서 필드 값 자체는 저장을 한 곳에서만 하고 다른 타입의 색인을 생성하고 싶을 때가 있다. 이때는 INTERNAL 키워드를 사용하여 해당 그런 조건을 만들 수 있다.

  ```
  kql/myvol> create table tab (
      -> f1 string, f2 int32, f3 text, f4 string list,
  ```

```
    -> constraint my_const1 reference type(f4) = internal f3 );
OK

kql/myvol> create index i0 on tab(f3); create index i1 on tab(f4);
OK

kql/myvol>
```

예제에서 f4 필드는 실제 값을 저장하지 않고, f3 필드 값을 참조한다. 필드 값에 대한 변경은 f3 필드를 통해서 이루어져야 한다.

- 상대 경로의 참조 해결을 위한 테이블 정의

  독크루저가 참조하는 파일 경로는 절대 경로이다. 따라서, 파일 경로를 가지고 있는 필드값이 상대 경로 형식으로 표현되어 있다면 'prefix' 옵션을 이용하여 절대 경로로 참조할 수 있게 해야 한다.

  아래 예제를 보면, CONSTRAINT 구문에 f3의 값이 파일의 상대 경로명일 경우, 'prefix' 옵션과 함께 상대 경로의 기준 디렉터리 경로를 입력하였다. 이 예에서는 /home/doc 이하에서 주어진 상대 경로를 이용하여 파일을 찾게 된다.

```
kql/my_vol> create table tab_with_file_ref (
    -> f1 string, f2 int32, f3 text, f4 text,
    -> constraint my_const1 primary key (f1, f2),
    -> constraint my_const2 reference type(f3)
       = local (prefix /home/doc) );
OK

kql/my_vol>
```

- 사용자정의 경로의 참조 해결을 위한 테이블 정의

  필드의 값이 파일 경로로부터 참조되어 색인되어야 하는데, 해당 경로를 원본 필드의 값을 이용하여 계산하여야 하는 경우 사용자정의 경로 구성 방법을 쓴다. 사용자정의 경로이라 함은 파일 경로를 반환하는 함수를 사용하여 파일을 색인하는 방법이다. 이 경우 테이블 정의는 파일 경로값을 구하는 사용자정의 함수명을 이용한다. 함수가 정의된 동적 라이브러리는 독크루저의 KQL 설정 파일에서 지정된 데이터 위치(data_location) 디렉터리에 등록해야 한다.

아래의 예에서는 동적 라이브러리 libuser.so에 정의된 my_func라는 함수를 통해 경로를 구하고 있다. 동적 라이브러리 libuser.so의 원시 파일은 아래 user.c와 같다(user.c를 libuser.so로 컴파일하는 것은 사용할 컴파일러의 이용 방법을 참고한다). 예시된 my_func에서는 단순히 필드 0(field_data[0])과 필드 1(field_data[1])의 값을 붙이는 것으로 파일의 경로를 구하고 있지만, 실제 상황에서는 경로 구성 연산이 훨씬 더 복잡할 수 있다. 필드 번호는 0 부터 시작하며 테이블 정의 시에 나열된 순서를 그대로 유지한다.

```
kql/my_vol> create table tab_with_file_ref (
    -> f1 string, f2 int32, f3 text, f4 text,
    -> constraint my_const1 primary key (f1, f2),
    -> constraint my_const2 reference type(f3) = local
       (userdef libuser.so my_func) );
OK

kql/my_vol>
```

```
// libuser.so 소스 코드 내용
extern "C" int my_func(char msg[], char* field_data[],
int field_size[], int n_field, char out_file_path[])
{
  sprintf(out_file_path, "%s%s", field_data[0], field_data[1]);
  return 0;
}
```

- 포맷 해독을 위한 테이블 정의

  필드값이 순수 텍스트만이 아닌 E-mail 문서나 웹 문서와 같이 서식이 있는 파일의 경우 독크루저는 서식 정보를 제거한 순수 데이터만을 사용한다. 원본 데이터의 서식을 제거하기 위해서는 VALUE FORMAT 제약을 이용하여 테이블을 정의한다.

  다음의 예는 f3이 파일 경로 정보이고, 해당 파일이 특정 포맷을 가지는 경우(예를 들어, Microsoft Office, HWP, PDF 등의 서식)에 대한 CONSTRAINT 정의 예이다. 입력 포맷이 항상 같다면 특정 포맷명을 지정함으로써 효율을 높일 수 있다. 현재 지원되는 포맷과 그 포맷명은 HTML(HTML), 한글과 컴퓨터 한글(HWP), Microsoft Word(DOC), Microsoft Excel(XLS), Microsoft PowerPoint(PPT), Adobe PDF(PDF), 핸디아리랑(HWD), MP3(MP3)이다. 만약 입력 포맷이 항상 동일하지 않거나 미리 알 수 없는 경우에는 'AUTO'로 지정한다.

```
kql/my_vol> create table tab_with_file_ref (
    -> f1 string, f2 int32, f3 text, f4 text,
    -> constraint my_const1 primary key (f1, f2),
    -> constraint my_const2 reference type(f3)
       = local (userdef libuser.so my_func)
    -> constraint my_const3 value format(f3) = auto );
OK

kql/my_vol>
```

3.3.2. 인덱스 생성 (CREATE INDEX)

테이블에 정의된 필드 중 검색이나 정렬 대상이 되는 필드는 인덱스로 생성되어야 한다. 이 명령은 인덱스를 정의한다.

- 구문

```
CREATE INDEX index_name ON table_name ( field_name_list )
```

구성요소	설명
index_name	정의할 인덱스의 이름
table_name	해당 테이블의 이름
field_name_list	해당 인덱스에 포함시키고자 하는 필드명의 반복으로써 콤마(',')로 구분된다.

인덱스의 이름은 테이블의 명명 규칙을 따른다.

인덱스를 생성하는 대상 필드가 하나일 경우는 '단일인덱스', 두 개 이상일 경우에는 '복합인덱스'라 부른다. 복합인덱스를 생성하는 경우 내부적으로 각 필드의 인덱스 정보도 생성된다. 복합인덱스를 생성할 경우 인덱스 대상 필드는 모두 같은 속성으로 정의되어야 한다. 인덱스를 정의하기 위해서는 먼저 테이블을 정의하여야 한다.

- 예제

```
kql/myvol> create index TTBB_idx on tbl (TT, BD);
OK
```

TTBD_idx는 인덱스 명으로 TT 필드와 BD 필드의 복합인덱스이다. 이때 TT, BD는 같은 속성으로 테이블에 정의되어있다. 인덱스를 이용하여 검색이나 정렬을 수행하게 되는 데 위 예제의 경우 TTBD_idx의 인덱스를 이용하여 검색하거나(TT, BD 두 필드를 동시 검색) TT, BD의 개별 필드를 대상으로 검색할 수 있다.

3.3.3. 인덱스 삭제(DROP INDEX)

인덱스를 삭제한다.

- 구문

```
DROP INDEX index_name ON table_name
```

구성요소	설명
index_name	삭제할 인덱스의 이름
table_name	테이블의 이름

3.3.4. 테이블 삭제 (DROP TABLE)

테이블을 삭제한다. 이 명령을 실행하면 볼륨 내에서 테이블이 완전히 삭제되고 테이블이 점유하던 자원이 모두 반납된다. 테이블이 존재하지 않으면 명령 수행이 실패한다.

- 구문

```
DROP TABLE [IF EXISTS] table_name
```

구성요소	설명
IF EXISTS	테이블이 존재하는 경우에 한해 명령을 실행
table_name	삭제할 테이블의 이름

- 예제

```
kql/myvol> drop table tbl;
OK
```

```
kql/myvol> drop table tbl;
Error: table 'tbl' not exists.
Operation cancelled.

kql/myvol> drop table if exists tbl;
OK
```

3.3.5. 테이블 스키마 동적 변경 (ALTER TABLE)

데이터는 변경하지 않고 테이블 스키마만 동적으로 변경한다. 변경된 필드에 대해서는 BUILD INDEX TO 명령을 사용하여 재색인을 해주어야 한다. 현재는 동적으로 새로운 컬럼을 추가하는 것은 안되고 변경만 할 수 있다. 향후 많은 동적 변경이 예상된다면 여벌 필드를 미리 만들어 놓는 것이 좋다.

- 구문

```
ALTER TABLE table_name [
    CHANGE COLUMN old_field_name new_field_name new_type |
    CHANGE TAG new_func_decl |
    CHANGE CONSTRAINT const_name const_decl |
    DELETE CONSTRAINT const_name]
```

구성요소	설명
table_name	테이블 이름
CHANGE COLUMN	컬럼 변경 명령
old_field_name new_field_name new_type	현재 이름(old_field_name)에 해당하는 필드를 새로운 타입(new_type)의 새로운 이름(new_field_name)으로 변경
CHANGE TAG	태그 함수 변경 명령
new_func_decl	새로운 태그 함수 선언
CHANGE CONSTRAINT	테이블 제약 변경 명령
const_name const_decl	const_name에 해당하는 테이블 제약은 새로운 제약 선언(const_decl)으로 변경
DELETE CONSTRAINT	테이블 제약 삭제 명령

구성요소	설명
const_name	삭제할 테이블 제약 이름

- 예제

  – 필드 동적 변경

  필드 추가 기능은 지원하지 않기 때문에 스키마상에서 사용하지 않는 필드를 선택하여 변경해야 한다. 다음은 'tbl' 테이블의 'tt'라는 필드 이름을 'title'로, 필드 타입은 TEXT LIST NULL로 설정하는 예시이다.

```
kql/myvol> ALTER TABLE tbl CHANGE COLUMN tt title TEXT LIST
NULL;
OK
```

  – CONSTRAINT 추가/변경

  기존에 있는 CONSTRAINT 이름을 사용하면 변경이 되고, 없던 CONSTRAINT를 사용하면 추가된다.

```
kql/myvol> ALTER TABLE tbl CHANGE CONSTRAINT c1 tag(tt) = NIL();
OK
```

  – CONSTRAINT 삭제

```
kql/myvol> ALTER TABLE tbl DELETE CONSTRAINT c1;
OK
```

3.3.6. 테이블 비우기 (TRUNCATE TABLE)

주어진 테이블에서 모든 레코드를 삭제한다.

- 구문

```
TRUNCATE TABLE table_name
```

구성요소	설명
table_name	테이블의 이름

3.3.7. 테이블 목록 보기 (SHOW TABLES)

볼륨 내에 존재하는 모든 테이블의 목록을 보여준다.

- 구문

```
SHOW TABLES
```

- 예제

```
kql/myvol> show tables;
+------------+
| TABLE NAME |
+------------+
|        tbl |
+------------+
Total 1 table.
OK
```

3.3.8. 테이블 상세 보기 (EXPLAIN)

주어진 테이블의 스키마 및 기타 상세 정보를 보여준다.

- 구문

```
EXPLAIN table_name [MORE]
```

구성요소	설명
table_name	테이블의 이름
MORE	테이블에 정의된 게이트웨이 정보까지 보여준다.

- 예제

```
kql/myvol> explain tbl;
Total 20 fields defined.
Total 4 indices defined.
Total 1000 records stored.
+------------+-----------------+------+-----+---------+
| FIELD NAME | TYPE            | MAX  | MIN | MEAN    |
```

```
+------------+-----------------+------+-----+---------+
| SN         | INT32           | 6    | 2   | 5.66    |
| STR1       | STRING VERBATIM | 135  | 11  | 43.59   |
| UB         | STRING LIST     | 117  | 1   | 26.67   |
| HS         | STRING          | 44   | 8   | 16.92   |
| VB         | TEXT            | 78   | 1   | 1.18    |
| DT         | TEXT            | 2754 | 1   | 4.66    |
| DB         | TEXT            | 5585 | 1   | 11.57   |
| CT         | TEXT            | 483  | 1   | 1.82    |
| CB         | TEXT            | 1174 | 1   | 2.74    |
+------------+-----------------+------+-----+---------+
+------------+------------+
| INDEX NAME | FIELD NAME |
+------------+------------+
|   STR1_idx | STR1       |
|   TTBD_idx | TT, BD     |
|     HS_idx | HS         |
|     UD_idx | UD         |
+------------+------------+

OK
kql/my_vol>
```

3.3.9. 레코드 삽입 (INSERT)

주어진 테이블에 레코드를 삽입한다. 이 명령어는 통상 점진 색인을 위해 이용되고 부분 실행 방지가 보장된다. 색인/검색 서버를 분리 운영하는 환경에서는 색인 서버에만 적용해야 한다.

- 구문

```
INSERT INTO table_name SET field_name = literal, ...
```

구성요소	설명
table_name	테이블의 이름
field_name	필드의 이름

구성요소	설명
literal	필드값 상수로써 공백 문자가 중간에 들어가면 큰 따옴표(" ")로 둘러싸야 한다.

모든 필드의 명칭과 값을 명시할 필요는 없으나, PRIMARY KEY 필드나 NON-NULL 속성의 필드가 있다면 해당 필드는 반드시 명시되어야 한다.

- 예제

```
kql/myvol> insert into WebData tbl set UR str1 =
'http://www.munhwa.co.kr/content/200308070101132916',
    -> TT = '특허심판 청구 상반기 4535건',
    -> BD = '올 상반기 청구된 심판 건수는 4735 건으로 작년 같은 기간
       4118건에 비해 10.1% 증가한 것으로 집계됐다', TM = '20050807',
       CL=0;
OK

kql/myvol>
```

3.3.10. 레코드 수정 (UPDATE)

주어진 테이블에서 기존의 레코드의 값을 수정한다.

명령이 성공적으로 수행되면 선택된 레코드에서 해당 필드 값이 명령어에 주어진 값으로 대치된다. 바뀐 필드 값에 맞도록 인덱스의 갱신도 동시에 이루어지므로 인덱스를 갱신하기 위한 별도의 조치는 필요없다. 이 명령어는 통상 점진 색인을 위해 이용되고, 부분 실행 방지가 보장된다. 색인/검색 서버를 분리 운영하는 환경에서는 색인 서버에만 적용해야 한다.

- 구문

```
UPDATE table_name SET field_name = literal, ...
 [WHERE where_definition] [LIMIT number]
```

대상이 되는 레코드의 선택 조건은 WHERE 절을 통해 명시된다. 만약 WHERE 절이 없으면 항상 조건을 만족하는 것으로 간주되어 모든 레코드가 선택된다.

LIMIT 절은 선택된 레코드의 최대 개수를 제한하기 위해 이용된다.

구성요소	설명
table_name	테이블의 이름
field_name	필드의 이름
literal	필드값 상수로써 공백 문자가 중간에 들어가면 큰따옴표(" ")로 둘러싸야 한다.
where_definition	선택 조건 수식으로써 SELECT 명령어에서와 같은 구문 및 의미를 가진다.
number	선택된 레코드의 최대 개수를 나타내는 십진 정수

3.3.11. 레코드 삭제 (DELETE)

주어진 테이블에서 레코드를 삭제한다. 이 명령어는 통상 점진 색인을 위해 이용되고, 부분 실행 방지가 보장된다. 색인/검색 서버를 분리 운영하는 환경에서는 색인 서버에만 적용해야 한다.

- 구문

```
DELETE FROM table_name [WHERE where_definition]
 [LIMIT number]
```

대상이 되는 레코드의 선택 조건은 WHERE 절을 통해 명시된다. 만약 WHERE 절이 없으면 항상 조건을 만족하는 것으로 간주되어 모든 레코드가 삭제되므로 주의해야 한다.

LIMIT 절은 선택될 레코드의 최대 개수를 제한하기 위해 이용된다. 레코드의 삭제와 함께 인덱스의 갱신도 동시에 이루어지므로 인덱스를 갱신하기 위한 별도의 조치는 필요없다.

구성요소	설명
table_name	테이블의 이름
where_definition	선택 조건 수식으로써 SELECT 명령어에서와 같은 구문 및 의미를 가진다.
number	선택된 레코드의 최대 개수를 나타내는 십진 정수

3.3.12. 레코드 검색 (SELECT, RETRIEVE)

테이블에 저장된 레코드의 값을 탐색한다. RETRIEVE로 검색할 경우 검색 결과의 적합도 점수를 확인할 수 있다.

- 구문

```
SELECT {* | field_name_list} FROM table_name
 [WHERE where_definition] [method]
 [ORDER BY { field_name | $ROWID | $RELEVANCE |
 $MATCHFIELD(field_name, ...) |
 $SIZE(field_name) | usersort_field_name (key ...) |
 $CATEGORYFIELD(field_name(domain) ..., 'keyword') |
 $USERDEF(libuser.so)} [ASC|DESC] , ...]
 [WEIGHT BY field_name=number, ...]
 [EVALUATE BY modrev(param) [domain=domain][, cmd=cmd]]
 [EXCLUDE BY field_name(domain)]
 [EXTRACT BY {MIN | MAX | MINMAX}(field_name)]
 [GROUP BY field_name [ORDER BY COUNT(*) [ASC|DESC]] ]
 [DISTINCT BY field_name ]
 [PAGELENGTH=number]
 [PAGEFORMAT=LIST | CSV | CLUSTERED BY field_name [count]]
 [AS LITERAL]
```

```
RETRIEVE {* | field_name_list} FROM table_name
 [WHERE where_definition] [method]
 [ORDER BY { field_name | $ROWID | $RELEVANCE |
 $MATCHFIELD(field_name, ...) |
 $SIZE(field_name) | usersort_field_name(key ...) |
 $CATEGORYFIELD(field_name(domain) ..., 'keyword') |
 $USERDEF(libuser.so)} [ASC|DESC] , ...]
 [WEIGHT BY field_name=number, ... ]
 [EVALUATE BY modrev(param) [domain=domain][, cmd=cmd]]
 [EXCLUDE BY field_name(domain)]
 [EXTRACT BY {MIN | MAX | MINMAX}(field_name)]
 [GROUP BY field_name [ORDER BY COUNT(*) [ASC|DESC]] ]
 [DISTINCT BY field_name ]
 [ABSOLUTE]
 [PAGELENGTH=number]
 [PAGEFORMAT=LIST | CSV | CLUSTERED BY field_name [count]]
 [AS LITERAL]
```

구성요소	설명
field_name_list	콤마(',')로 분리된 필드 명칭의 연속으로써 검색된 레코드에서 값을 보여주고자 하는 필드. 별표('*')를 명시하면 모든 필드를 보여준다.
table_name	테이블의 이름
where_definition	선택 조건 수식으로써 필드 명칭 변수와 리터럴로 구성되는 관계 및 논리 연산식
PAGELENGTH=*number*	명령어 수행 결과 한 페이지에 보여주는 결과의 개수를 변경한다. (기본값=20(개))
PAGEFORMAT	코드 검색 결과를 보여줄 때 현재의 기본값 대신 다른 형식으로 바꾼다. – LIST: 목록 형태로 보여줌(기본값) – CSV(comma separate value): 콤마(',')로 분리된 데이터 내용만 보여줌 – CLUSTERED BY: 지정한 필드의 내용을 비슷한 내용끼리 적절하게 분류하여 보여줌
count	최대 카테고리 수
AS LITERAL	레퍼런스 제약 등 필드 값에 대한 해석을 하지 않고 필드 값 그대로(raw) 출력함
ABSOLUTE	적합도 절대값을 결과로 같이 출력
ORDER BY	선택된 레코드를 임의의 조건으로 정렬, 여러 개의 필드를 중복하여 나열하는 것이 가능하다. 필드값 이외에도 $ROWID, $MATCHFIELD 등 메타값에 의한 정렬이 가능하다.

ORDER BY와 WEIGHT BY, EVALUATE BY, EXCLUDE BY, EXTRACT BY, GROUP BY, DISTINCT BY에 대한 자세한 내용은 '정렬'을 참고한다.

- 예제

```
kql/tv>> select TT from GISA where TT_INDEX='축구' allword
pagelength=10;
---------- 0 of total 18 (ROWID 19) ---------
TT (44): [프로축구-종합] 성남 골득실차 마지막 4강티켓
---------- 1 of total 18 (ROWID 21) ---------
TT (22): [ST축구기록실] 5월 1일
---------- 2 of total 18 (ROWID 497) ---------
TT (44): <축구소식> 정몽준 회장, FIFA 임시집행위 참석
---------- 3 of total 18 (ROWID 534) ---------
TT (42): [축구] 왕년의 브라질 스타 카레카 서울 온다
---------- 4 of total 18 (ROWID 858) ---------
TT (23): [ST축구기록실] 3월 30일
---------- 5 of total 18 (ROWID 2274) ---------
TT (37): 스페인축구 멘디에타, 스페인 무대 복귀
---------- 6 of total 18 (ROWID 861) ---------
TT (21): [오늘의 축구] 4월 1일
---------- 7 of total 18 (ROWID 1608) ---------
TT (37): 21세미만 유럽청소년축구대회 대진 확정
---------- 8 of total 18 (ROWID 2404) ---------
TT (37): 포르투갈 축구 고메스, 계약무효화 요구
---------- 9 of total 18 (ROWID 911) ---------
TT (40): 아프리카 11개 축구협회, 하야투 지지 표명
10 of 18 records displayed. Continue(Y/n)? n
OK
kql/tv>> select TT from GISA where $ROWID = 19;
---------- 0 of total 1 (ROWID 19) ---------
TT (44): [프로축구-종합] 성남 골득실차 마지막 4강티켓
Total 1 record.
OK
kql/tv>> retrieve TT from tv where TT='축구';
---------- 0 of total 18 (ROWID 8, REL 9992) ---------
TT (54): [개최도시 화려한 축하쇼] 축구 축제! 장엄한 카운트다운
---------- 1 of total 18 (ROWID 19, REL 9989) ---------
TT (44): [프로축구-종합] 성남 골득실차 마지막 4강티켓
---------- 2 of total 18 (ROWID 21, REL 9984) ---------
TT (22): [ST축구기록실] 5월 1일
...
Total 18 records.
OK
```

```
kql/tv>> quit
Bye
```

3.3.13. 삭제된 레코드 복원 (TRASHBOX)

UNDELETE 명령을 이용하면 삭제한 레코드를 복원할 수 있다. 기본적으로 삭제한 레코드는 TRASH BOX에 들어가게 되는데, SELECT 명령으로 확인가능하다.

- 구문
  - 특정 레코드 번호 삭제

```
DELETE rowid, ... FROM table_name
```

  - 레코드 복원

```
UNDELETE { * | rowid, ... } FROM table_name TRASH BOX
```

  - TRASH BOX의 레코드 확인

```
SELECT { * | field_name_list } FROM table_name TRASH BOX
   [WHERE where_definition] [method]
   [ORDER BY { field_name | $ROWID | $RELEVANCE |
    $MATCHFIELD(field_name, ...)]
   ...
```

구성요소	설명
table_name	테이블의 이름
rowid	삭제할 레코드 rowid
field_name_list	보고자 하는 필드 목록

- 예제

```
kql/tv> delete from GISA where SO = 'Inews24';
Total 37 records.
OK
```

```
kql/tv> select SO from GISA trash box;
---------- 0 of total 37 (ROWID 38) ---------
SO (7): Inews24
---------- 1 of total 37 (ROWID 77) ---------
SO (7): Inews24
---------- 2 of total 37 (ROWID 82) ---------
SO (7): Inews24
---------- 3 of total 37 (ROWID 100) ---------
SO (7): Inews24
...

kql/tv> undelete 38, 82 from GISA trash box;
Total 2 records.
OK

kql/tv> select SO from GISA where SO = 'Inews24';
---------- 0 of total 2 (ROWID 38) ---------
SO (7): Inews24
---------- 1 of total 2 (ROWID 82) ---------
SO (7): Inews24
Total 2 records.
OK
```

3.3.14. 레코드 내보내기 (EXPORT RECORD)

검색 레코드 데이터를 텍스트 파일로 내려 받는다.

원격으로 데이터를 내려받을 때는 원격지에 docruzerd 또는 kql이 구동되어 있어야 하며 덤프파일은 지정한 *folder/table_name* 디렉터리 아래에 생성된다. FORMAT 구문에 내려 받을 필드 이름과 필드 구분자를 지정한다. 필드 구분자는 데이터에 나오지 않는 것으로 정의한다. 텍스트 데이터는 지정한 디렉터리 '*folder*' 아래에 '*table_name*/0000/+일련번호.txt'와 같은 형식의 이름으로 생성된다

- 구문

```
EXPORT RECORDS FROM table_name
    [WHERE expr] TO [//listener/]folder
    [DEPTH number][DIVIDE BY number]
    [LIMIT number][FILE SIZE number]
```

```
    [PREFIX string][INFIX LENGTH number][SUFFIX string]
    FORMAT format_decl
```

구성요소	설명
table_name	레코드를 내보낼 테이블의 이름
expr	선택적으로 레코드를 내보낼 수 있게 where 조건을 지정할 수 있음
listener	원격지 독크루저 서비스 주소. 원격지 폴더로 보내고자 하는 경우 명시
folder	내보내기를 할 최상위 디렉터리 이름
DEPTH	서브 디렉터리 깊이 지정. (기본값=1)
DIVIDE BY	디렉터리 내 파일 개수 지정(기본값=100개)
LIMIT	덤프 받을 레코드의 최대 개수 지정(기본값=0, 조건 내의 모든 레코드)
FILE SIZE	생성할 텍스트 파일의 크기(단위 byte) (기본값=4MB)
PREFIX	파일명 시작 스트링(기본값="")
INFIX LENGTH	파일번호 표시 자릿수(기본값=4. 즉 0000, 0001…로 표시)
SUFFIX	파일명 끝 스트링(기본값=".txt")
FORMAT	덤프 받을 파일의 필드 및 태그 지정. (기본값=필드 전체. 태그는 <필드명>)

- 예제

```
kql/myvol> export records from GISA to /home/doc format
TT="<__TT__>", SO="<__SO__>";
Total 2465 records.
OK

$ cat /home/doc/GISA0000.txt
<__TT__>유가하락, 미국석유재고 증가
<__SO__>머니투데이
<__TT__>WP 故 그레이엄 여사 美 신문협회 생애성취상
<__SO__>중앙일보
```

```
<__TT__>[권노갑씨 일문일답]
<__SO__>중앙일보
<__TT__>수원.울산 '4강 격돌' 안양.성남
<__SO__>중앙일보
<__TT__>여경기지사 후보경선 임창열씨 포기할 듯
<__SO__>중앙일보
...
```

3.3.15. 결과 화면 파일출력 (DUMP)

실행 결과를 화면이 아니라 파일로 덤프하도록 지정한다.

- 구문

```
DUMP TO local_filename statement
```

구성요소	설명
local_filename	덤프 결과 출력 파일 경로
statement	덤프할 KQL 구문

- 예제

```
kql/myvol> dump to tbl.out select * from tbl;
OK
```

```
 ============ tbl.out 파일 내용 ============
---------- 0 of total 2465 (ROWID 0) ---------
TT (27): 유가하락, 미국석유재고 증가
BD (507): [머니투데이] 지난주 미국의 석유재고가 예상을 깨고 증가세를
나타내면서...
```

3.3.16. 인덱스 비우기(TRUNCATE INDEX TO)

주어진 필드의 모든 색인 데이터를 삭제한다.

- 구문

```
TRUNCATE INDEX TO table_name[.field_name]
```

구성요소	설명
table_name	색인 필드가 속하는 테이블의 이름
field_name	색인 데이터를 삭제할 필드의 이름

3.3.17. 동적 필드 색인(BUILD INDEX TO)

전체 테이블 및 특정 필드의 색인 키워드를 추출한다. 이 명령을 이용하면 색인 설정이나 사전 변경 시 해당 필드나 테이블을 재색인해 줄 수 있다.

- 구문

```
BUILD INDEX TO table_name[.field_name][PARALLEL number]
```

구성요소	설명
table_name	색인을 생성할 테이블의 이름
field_name	색인을 생성할 필드의 이름
PARALLEL *number*	병렬 처리를 위한 옵션으로 색인 시 사용되는 스레드 수를 지정한다. 미지정 시에는 kql.rc의 no_of_parallel_task를 따른다.

3.3.18. 색인 결과 확인 (VIEW INDEX)

각 필드의 색인어 추출 결과를 확인하한다.

- 구문

```
VIEW INDEX TO table_name.field_name [PATTERN=term_spec |
 KEYWORD=term]
[PAGELENGTH=number]
```

구성요소	설명
table_name	테이블의 이름
field_name	필드의 이름

구성요소	설명
term_spec	키워드 텀(term)의 부분 스트링 패턴
term	색인 키워드
number	한 페이지에 보여줄 레코드 개수

- 예제

```
kql/tv> view index to GISA.TT;
0th: key 0 =  166 ( 166) [1]
1th: key 1 =   17 (  17) [1분기]
2th: key 2 =    6 (   6) [pda]
3th: key 3 =   24 (  24) [분기]
4th: key 4 =   44 (  44) [시장]
5th: key 5 =   19 (  19) [올]
6th: key 6 =    2 (   2) [위축]
7th: key 7 =    8 (   8) [유럽]
8th: key 8 =    6 (   6) [크게]
...

kql/tv> view index to GISA.TT keyword='유럽';
0th: rowid    0 = [0]  1 (2)
1th: rowid 1067 = [0]  1 (0)
2th: rowid 1113 = [0]  1 (0)
3th: rowid 1132 = [0]  1 (4)
4th: rowid 1561 = [0]  1 (1)
5th: rowid 1608 = [0]  1 (1)
6th: rowid 1784 = [0]  1 (4)
7th: rowid 1827 = [0]  1 (7)
[유럽] Total 8 posts.
OK

kql/tv>
```

예제에서 view index to GISA.TT의 결과로 출력된 7th: key 7 = 8 (8) [유럽]에서 각 컬럼의 의미는 다음과 같다:

– 7th: 표시번호

– key 7: 색인키 번호

– 8: 색인어가 출현한 레코드 개수

– (8): 삭제되지 않은 레코드 개수

– [유럽]: 색인어

view index to tbl.TT keyword='유럽';의 결과로 출력된 3th: rowid 1132 = [0] 1 (4)에서 각 컬럼의 의미는 다음과 같다:

– 3th: 표시번호

– rowid 1132: 레코드 번호

– [0]: 문서 내 키워드 가중치

– 1: 출현 개수

– (4): 출현 위치 (4 번째 어절)

delete 명령으로 한 개의 레코드를 삭제하면 다음과 같다. 삭제된 레코드는 '(trashed)'라고 표시된다.

```
kql/tv> delete 1784 from GISA;
Total 1 record.
OK

kql/tv> view index to GISA.TT keyword='유럽';
0th: rowid    0 = [0]  1 (2)
1th: rowid 1067 = [0]  1 (0)
2th: rowid 1113 = [0]  1 (0)
3th: rowid 1132 = [0]  1 (4)
4th: rowid 1561 = [0]  1 (1)
5th: rowid 1608 = [0]  1 (1)
6th: rowid 1784 = [0]  1 (4) (trashed)
7th: rowid 1827 = [0]  1 (7)
[유럽] Total 8 posts (total 1 trashed).
OK
```

3.3.19. 색인 결과 편집 (EDIT INDEX)

색인 결과 중 원하는 레코드를 삭제하거나 추가한다. 검색 결과에 직접적으로 영향을 주게 되므로 사용 시에는 주의해야 한다.

- 구문

```
EDIT INDEX TO table_name.field_name [KEYWORD=term]
 {DELETE [rowid, ...] | INSERT rowid [loc, ...]}
```

구성요소	설명
table_name	테이블의 이름
field_name	필드의 이름
term	색인 키워드
rowid	레코드 ID
loc	위치(어절) 정보

- 예제

```
kql/tv> view index to GISA.TT keyword='유가';
0th: rowid 0 = [0] 1 (0)
1th: rowid 1311 = [0] 1 (2)
2th: rowid 1441 = [0] 1 (0)
3th: rowid 1508 = [0] 1 (0)
4th: rowid 1560 = [0] 1 (1)
5th: rowid 1609 = [0] 1 (0)
6th: rowid 2444 = [0] 1 (0)
[유가] Total 7 posts.
OK

kql/tv> edit index to GISA.TT keyword='유가' delete 1311;
OK

kql/tv> view index to GISA.TT keyword='유가';
0th: rowid 0 = [0] 1 (0)
1th: rowid 1441 = [0] 1 (0)
2th: rowid 1508 = [0] 1 (0)
```

```
3th: rowid 1560 = [0] 1 (1)
4th: rowid 1609 = [0] 1 (0)
5th: rowid 2444 = [0] 1 (0)
[유가] Total 6 posts.
OK

kql/tv>
```

3.3.20. 색인 결과 요약 (PUT INDEX SUMMARY OF)

필드의 색인 결과를 요약하여 파일로 내려 받는다.

- 구문

```
PUT INDEX SUMMARY OF table_name.field_name TO file_path
```

구성요소	설명
table_name	테이블의 이름
field_name	필드의 이름
file_path	요약 결과를 출력할 파일 경로

- 예제

```
kql/tv> put index summary of GISA.TT TO /home/doc/TT.txt ;
OK

------------ /home/doc/TT.txt ----------------
일 117
억 114
월 95
원 91
월드컵 77
```

3.4. 외부 레코드 가져오기 명령 – 파일 시스템

외부에 저장된 레코드 원본은 독크루저가 바로 읽어 들일 수 있는 형태로 존재하지 않고 고유의 저장 매체나 저장 포맷을 가지고 있는 것이 보통이다. 외부 원본의 저장 포맷을 해독하여 데이터 값을 추출해낸 다음 레코드로 저장하는 과정을 '가져오기(import)'이라 한다. 게이트웨이는 이러한 외부 원본으로부터 레코드를 가져오기 위한 '관문'이 된다.

파일 시스템 게이트웨이(File System Gateway)는 파일 시스템에 존재하는 파일을 독크루저의 레코드로 읽어 들일 때 이용하는 게이트웨이이며, 주로 레코드의 필드값이 파일에 어떻게 표현되어 있는지에 관한 정보(예: 필드별 포맷 문자)로 구성된다.

3.4.1. 형식 파일을 가져오기 위한 게이트웨이 정의

형식 파일은 수입할 레코드들이 저장되어 있는 텍스트 파일로 레코드를 구성하는 필드들이 구분자에 의해 구분되어 있다. 형식 파일에 있는 레코드들은 구분자에 의해 수입될 레코드의 필드별 구획을 지정할 수 있다. 이러한 예로는, DBMS 에서 필드간 구분 문자를 이용하여 파일로 덤프 시킨 레코드의 집합을 들 수 있다.

파일 시스템 게이트웨이는 이러한 구분자 정보를 정의하여 검색시스템으로 파일에 저장되어 있는 레코드를 수입하도록 한다. 구분 문자는 각 필드별로 다르게 지정할 수도 있고, 이 구분 문자가 필드의 앞 또는 뒤에 위치하는지도 지정할 수 있으며 행의 시작 또는 끝에 위치하는지를 지정함으로써 더 빨리 포맷을 해독하게 할 수도 있다.

- 구문

```
CREATE GATEWAY gateway_name ON table_name TO FILE SYSTEM
FORMAT FIELD [LINE] {STARTS|ENDS} WITH field_name=value,
...
```

구성요소	설명
gateway_name	사용자가 정의하는 게이트웨이 식별자
table_name	게이트웨이를 통해 가져온 데이터가 저장될 테이블의 이름
LINE	레코드의 각 필드가 라인단위로 구성이 됐을 때 쓸 수 있는 옵션이다. 라인 단위 I/O를 함으로써 데이터 파싱 성능을 높일 수 있다. 기본적으로는 문자열 스트림으로 파싱을 한다.

구성요소	설명
	– STARTS: 필드에 대응하는 태그가 라인의 시작 위치에 온다. – ENDS: 필드에 대응하는 태그가 라인의 끝 위치에 온다.
field_name	KQL 테이블의 필드명
value	해당 필드에 해당하는 파일의 필드 구분자

- 예제

```
-----------  my_collection.txt ----------------
<title> 오페라의 유령
<author> 가스통 르루
<price> 9200
<title> 로봇
<author> 아이작 아시모프
<price> 8000
...

kql/myvol> create table tbl (
    -> f_title TEXT,
    -> f_author TEXT,
    -> f_price INT32);
OK

kql/myvol> create gateway gw on tbl to file system
    -> format field starts with
    -> f_title="<title>",
    -> f_author="<author>",
    -> f_price="<price>";
OK
```

예제에서 'my_collection.txt'는 각각의 필드들이 구분자에 의해서 구분되어 있는 형식 파일이다. 이를 가져오기 파일게이트웨이 'gw'를 정의하였다. 이 게이트웨이는 각 구분자로 시작하는 필드로부터 다음 구분자까지의 텍스트를 읽어 KQL의 레코드 필드로 가져온다.

다음은 필드의 뒷부분에 구분 문자 '|'가 있는 파일의 포맷을 해독할 수 있도록 하는 게이트웨이의 생성 예이다.

```
----------- my_collection.txt ----------------
오페라의 유령 | 가스통 르루 | 9200 |
로봇 | 아이작 아시모프 | 8000 |
묵향 | 전동조 | 7500 |
...

kql/myvol> create table tbl (
    -> f_title TEXT,
    -> f_author TEXT,
    -> f_price INT32);
OK

kql/myvol> create gateway gw on tbl to file system
    -> format field ends with
    -> f_title="|",
    -> f_author="|",
    -> f_price="|";
OK
```

3.4.2. 무형식 파일을 가져오기 위한 게이트웨이 정의

원본 파일이 형식 파일과 같이 레코드, 필드 구분이 없는 일반 파일이라면 레코드를 가져올 때 파일의 이름이나 작성일 등의 속성을 검색 필드의 값으로 지정하는 것이 좋다.

독크루저에서는 다음과 같은 종류의 파일 속성 변수가 제공되어, 포맷 없이 파일을 수입하고자 할 때에 필드의 값으로 파일 속성을 지정하는 것이 가능하다. 이 방법은 파일 시스템에 존재하는 파일을 검색해 주는 시스템을 구현하고자 할 때에 유용하다.

변수	값
$FILE.PATH	파일의 절대 경로명
$FILE.NAME	파일명
$FILE.BODY	전체 파일 내용
$FILE.SIZE	파일의 크기(byte)
$FILE.DATE	파일의 마지막 변경 시간

- 구문

```
CREATE GATEWAY gateway_name ON table_name TO FILE SYSTEM FORMAT
FIELD VALUE TAKES field_name=variable, ...
```

구성요소	설명
gateway_name	생성하려는 게이트웨이 이름
table_name	게이트웨이가 동작할 KQL 테이블 이름
field_name	테이블의 각 필드 이름
variable	$FILE.PATH, $FILE.SIZE 등 무형식 게이트웨이에서 지원하는 파일 속성

- 예제

다음은 파일 시스템을 순회하면서 파일 경로, 파일명, 파일 내용, 파일 크기, 파일 변경 시간 등을 읽어내기 위한 게이트웨이를 생성하는 것이다.

```
kql/myvol> create table tbl (
    -> f_path STRING,
    -> f_name STRING,
    -> f_body TEXT,
    -> f_size INT32,
    -> f_upd_time DATE);
OK

kql/myvol> create gateway gw on tbl to file system format field
value takes
    -> f_path=$FILE.PATH, f_name=$FILE.NAME,
    -> f_body=$FILE.BODY, f_size=$FILE.SIZE,
       f_upd_time=$FILE.DATE;
OK
```

3.4.3. 원격/로컬 파일 가져오기(IMPORT FILES)

파일 게이트웨이는 파일 시스템으로부터 레코드를 수입할 때의 KQL 테이블의 필드와 원본 자료의 필드 항목에 대한 매핑을 정의하는 것이다.

IMPORT FILES 명령어는 이런 파일 게이트웨이에 매핑된 정보를 이용하여 실제로 레코드를 가져오는 명령어이다. 지정된 파일 시스템 상의 디렉터리(*folder*)에서 *file_spec*에 해당하는 파일을 *format_decl*에 맞게 해석하여 테이블로 적재한다.

3.3.0 이상부터 원격지에 구동된 KQL listener를 통해 원격지 파일을 가져와 색인하는 것도 가능하다. 원격파일색인을 할 때는 원격지에 docruzerd 또는 kql이 구동되어 있어야 한다.

INSERT/STORE 연산을 제외한 나머지 연산들은 테이블에 반드시 PRIMARY KEY가 정의되어 있어야 한다.

- 구문

```
IMPORT FILES [//listener/]folder file_spec TO table_name
  [THROUGH gateway_name| FORMAT format_decl]
  [PARALLEL number]
  [OPERATION {INSERT|STORE|INSERT_OR_SKIP|UPDATE|
   UPDATE_OR_INSERT|UPDATE_OR_SKIP|DELETE|DELETE_OR_SKIP|
   DELETE_AND_INSERT|DELETE_INSERT|INSERT_IF_DELETE}]
  [LIMIT number][LANGUAGE=language][CHARSET=charset]
```

구성요소	설명
listener	원격 독크루저 서비스 주소
folder	가져올 원격/로컬 파일이 들어있는 디렉터리
file_spec	파일명 또는 <*.확장자>
table_name	레코드를 import시킬 테이블 이름
gateway_name	레코드 import시 파싱 포맷이 명시된 게이트웨이 이름
format_decl	게이트웨이 대신 필드 매핑 관계를 직접 주게 할 수도 있음
PARALLEL *number*	가져오기 작업을 동시에 할 쓰레드 수
OPERATION {INSERT, ... }	가져온 파일 레코드 처리 방법. OPERATION을 지정하지 않으면 STORE 연산 후에 색인을 생성한다.
LIMIT *number*	가져올 레코드 개수의 최대값(기본값: 제한 없음)
language	가져오는 레코드의 언어(기본값: KOREAN)
charset	가져오는 레코드의 문자 세트(기본값: EUCKR)

- 각 연산(OPERATION)의 의미는 다음과 같다.

연산(OPERATION)	설명
INSERT	레코드를 테이블로 삽입하고 관련 인덱스도 갱신시켜 준다.
STORE	레코드를 테이블로 삽입만하고, 인덱스 갱신은 시키지 않음. 인덱스 갱신은 수동으로 시켜줘야 한다(BUILD INDEX).
INSERT_OR_SKIP	PRIMARY KEY에 해당하는 레코드를 찾아보고 같은 키가 없으면 INSERT시킨다.
UPDATE	PRIMARY KEY에 해당하는 레코드를 찾아보고 같은 키가 있으면 UPDATE시키고 없으면 에러를 낸다.
UPDATE_OR_INSERT	PRIMARY KEY에 해당하는 레코드를 찾아보고 같은 키가 있으면 UPDATE, 없으면 INSERT시킨다.
UPDATE_OR_SKIP	PRIMARY KEY에 해당하는 레코드를 찾아보고 같은 키가 있으면 UPDATE시키고 없으면 그냥 넘어간다.
DELETE	PRIMARY KEY에 해당하는 레코드를 찾아보고 같은 키가 있으면 DELETE시키고 없으면 에러를 낸다.
DELETE_OR_SKIP	PRIMARY KEY에 해당하는 레코드를 찾아보고 같은 키가 있으면 DELETE시키고 없으면 그냥 넘어간다.
DELETE_AND_INSERT	PRIMARY KEY에 해당하는 레코드를 찾아보고 같은 키가 있으면 DELETE시키고 INSERT를 시도하는데, DELETE나 INSERT중 하나라도 실패하면 에러를 낸다. 의미상으로는 UPDATE 연산과 같음.
DELETE_INSERT	PRIMARY KEY에 해당하는 레코드를 찾아보고 같은 키가 있으면 DELETE하고 INSERT, 같은 키가 없으면 INSERT만 수행한다. INSERT가 성공해야 정상 종료이다. 의미상으로는 UPDATE_OR_INSERT 연산과 동일.
INSERT_IF_DELETE	PRIMARY KEY에 해당하는 레코드를 찾아보고 같은 키가 있으면 DELETE하고 INSERT, 같은 키가 없으면 그냥 종료한다. 의미상으로는 UPDATE_OR_SKIP 연산과 동일.

- 예제

```
kql/myvol> IMPORT FILES ../rawdata *.dat TO tbl THROUGH gw
OPERATION STORE;
OK

kql/my_vol> BUILD INDEX TO tbl;
OK
```

3.4.4. 파일시스템 레코드 가져오기(IMPORT RECORDS)

IMPORT FILES 명령의 옛날 버전 형식이다. 하위호환성 목적으로 남아있다.

dir_path 및 하위 디렉터리 밑에서 파일명이 *file_spec*에 맞는 파일을 읽어서 테이블 *table_name*의 스키마에 맞는 레코드로 변환하여 저장 및 인덱싱을 수행한다.

- 구문

```
IMPORT RECORDS FROM FILE SYSTEM
  {TO table_name
  THROUGH gateway_name
  SELECT dir_path [file_spec] [OPERATION {INSERT|STORE}]
  [LIMIT number][LANGUAGE=language][CHARSET=charset] }
```

구성요소	설명
table_name	레코드를 import할 테이블 이름
gateway_name	레코드 import 시 파싱 포맷이 명시된 게이트웨이 이름
dir_path	지정된 파일 시스템 상의 디렉터리 및 하위 디렉터리
file_spec	파일명 또는 <*.확장자>
number	가져올 레코드 개수의 최대값(기본값: 제한 없음)
language	가져오는 레코드의 언어(기본값: KOREAN)
charset	가져오는 레코드의 문자 세트(기본값: EUCKR)

- 예제

```
kql/myvol> import records from file system to tbl through gw
select /home/data/doc *.txt;
OK
```

3.5. 외부 레코드 가져오기 명령 – 데이터베이스

DBMS에 저장되어 있는 레코드를 가져오는 가장 간단한 방법은 레코드 값을 필드별 구분 문자로 구분 지어 임시 파일에 덤프한 다음, 파일 시스템 게이트웨이를 통해 가져오는 방법이다. 그러나 이 방법은 효율이 좋지 않을 뿐 아니라 시시각각 발생하는 DB 상의 업데이트를 실시간 온라인으로 반영하기에 부적합하다. 따라서 DB에서 초기 레코드를 수입한 후 내용이 거의 바뀌지 않는 경우가 아니라면 DB 게이트웨이를 통해 레코드를 가져오는 것이 바람직하다.

게이트웨이를 통해 레코드를 가져올 때 처음에는 전체 범위를 가져오면 되므로 비교적 간편하게 이루어질 수 있다. 일단 레코드를 한번 가져온 후에는 추가적인 변경이 거의 발생하지 않는 응용에서는 이러한 운용 방식이 효과적이다. 그러나 최초로 가져온 후에 레코드의 삽입/변경/삭제가 자주 일어난다면 매번 다른 선택 범위와 연산을 지정해야 하는 IMPORT 명령을 매 변경 시마다 실행해줘야 하므로 운용 노력이 많이 필요하게 된다. 이러한 경우에는 DB 와처를 이용하여 볼륨을 변경시키는 것이 더 효과적이다.

와처는 데이터베이스 내에 상주하면서 DB의 변경 내용을 기록하고 있다가 DB를 가져올 시점이 되면 자동으로 변경 범위와 변경 연산을 지정해 준다. 따라서 와처를 이용하면 IMPORT 명령에 변경 범위와 연산을 지정해야 하는 번거로움을 없앨 수 있다. 또한 독크루저의 스케줄 기능을 이용하여 IMPORT 명령을 데이터의 신규성이 확보될 수 있을만한 시간 간격으로 실행시켜준다면 시시각각 발생하는 데이터의 변경을 자동으로 독크루저의 볼륨에 실시간으로 반영할 수 있게 된다.

지원하는 데이터베이스는 Oracle, SQL 서버, Sybase, DB2, MySQL, Notes이며, 데이터베이스 게이트웨이와 와처에 관한 자세한 사항은 'DOCRUZER Gateway Manual'을 참고한다.

3.6. 캐시 관리 명령

정렬 및 그룹핑 속도를 향상시키기 위해 필드를 캐시 메모리로 적재하여 관리할 수 있다. ATTACH, DETACH CACHE를 사용하면 명령어에 따른 지정한 필드의 키 정보를 캐시 메모리에 적재한다. 키 정보는 max_working_memory_size 내에서 할당된다. 따라서 통상적인 검색이나 증분 복사를 수행하기에 부족하지 않는 수준에서 캐시에 로딩해야 한다. 전체 복사 또는 증분 복사 수행 후에는 DETACH CACHE & ATTACH CACHE를 통하여 캐시를 갱신해 주어야 한다.

3.6.1. 캐시 적재 (ATTACH CACHE)

필드의 키 번호 정보 (ATTACH FIELD KEY CACHE TO)

각 레코드의 원하는 필드 키 번호를 캐시 메모리로 적재한다. 해당 필드로 정렬하거나 그룹핑하는 경우 메모리상에서 정렬/그룹핑이 가능하다. 그룹핑만 하는 경우면 필드키 캐시(FIELD KEY CACHE)만으로 충분하나, 정렬을 인메모리(in-memory)로 하려면 다음 절에 나오는 키오더 캐시(KEY ORDER)도 같이 적재되어야 한다.

- 구문

```
ATTACH FIELD KEY CACHE TO table_name.field_name
[ROW_SIZE=number]
[COLUMN_SIZE=number]
```

구성요소	설명
table_name	테이블의 이름
field_name	필드의 이름
[ROW_SIZE=*number*]	최대 레코드 개수
[COLUMN_SIZE=*number*]	필드 당 평균 키 개수보다 약간 넉넉하게 설정

필드 키값의 순서 정보 (ATTACH KEY ORDER CACHE TO)

정렬을 원하는 필드의 키값 순서정보를 캐시 메모리로 적재한다. 해당 필드로 정렬하는 경우 이미 메모리 적재된 정렬된 순서정보를 사용하기 때문에 정렬 속도가 향상된다.

- 구문

```
ATTACH KEY ORDER CACHE TO table_name.field_name
```

구성요소	설명
table_name	테이블의 이름
field_name	필드의 이름

필드의 키값별 POST 정보 (ATTACH POST CACHE TO)

원하는 필드의 각 키값별 post 정보($ROWID 집합 정보)를 캐시 메모리로 적재한다. 위치 연산을 하지 않는 검색의 경우에 검색 키의 post 정보를 디스크에서 읽지 않고 메모리상에서 검색하여 검색속도가 향상된다.

- 구문

```
ATTACH POST CACHE TO table_name.field_name
```

구성요소	설명
table_name	테이블의 이름
field_name	필드의 이름

3.6.2. 캐시 해제 (DETACH CACHE)

적재된 각 캐시 데이터를 메모리에서 해제한다.

- 구문

```
DETACH { FIELD KEY | KEY ORDER | POST } CACHE FROM
table_name.field_name
```

구성요소	설명
table_name	테이블의 이름
field_name	필드의 이름

3.6.3. 캐시 재적재 (RELOAD CACHE)

적재된 각 캐시 데이터를 메모리에서 해제하고 다시 적재한다. 의미상으로는 DETACH CACHE 후 ATTACH CACHE를 수행하는 것과 똑같은데, 검색 부하가 심한 상황에서 DETACH/ATTACH에 따르는 오버헤드를 약간 더 줄여주는 효과가 있다.

- 구문

```
RELOAD { FIELD KEY | KEY ORDER | POST } CACHE TO
table_name.field_name
```

구성요소	설명
table_name	테이블의 이름
field_name	필드의 이름

3.6.4. 캐시 모니터 (MONITOR CACHE)

적재된 각 캐시 데이터 메모리 사용 현황을 모니터한다.

- 구문

```
MONITOR { FIELD KEY | KEY ORDER | POST } CACHE ON
table_name.field_name
```

구성요소	설명
table_name	테이블의 이름
field_name	필드의 이름

3.7. 스케줄 관리 명령

스케줄은 KQL 설정 파일에서 설정한다. 설정 파일에 다음과 같은 구문 형식으로 등록한 경우 KQL 스케줄 관리 명령을 이용하여 등록한 스케줄을 제어할 수 있다.

- 구문

```
name = "action", period [(exception)]
```

구성요소	설명
name	스케줄 이름
action	해당 스케줄 동작 내용
period	period는 다음과 같은 형식으로 지정할 수 있다. `EVERY { [ {SUN|MON|TUE|WED|THU|FRI|SAT} | day ] hour:minute | number {SEC|MIN|HOUR|DAY [BUT hour:minute – hour:minute]} } | ON STARTUP` 시간 간격은 다음과 같이 지정할 수 있다. – 주기적: every number {sec|min|hour}. – 일간: every hh:mm – 주간: every {sun|mon|tue|wed|thu|fri|sat} hh:mm – 월간: every dd hh:mm – 'ON STARTUP' 구문은 엔진이 시작할 때 한 번만 실행되도록 지정하는 구문이다. – 'BUT hour:minute – hour:minute'으로 실행 제외시각을 지정할 수 있는 데 BUT 설정은 KQL 설정 파일에 등록하는 경우에만 사용 가능하다.
exception	'exception 조건' '실행할 구문', ... 의 형식으로 지정할 수 있다. 조건으로 지정할 수 있는 것은 'OK', 'FAILED'이다.

- 예제

```
schedule=b1, "run batch.kql;", every 100 sec;매100초
schedule=b2, "run batch.kql;", every 100 min but
```

```
 23:00 – 07:00;매100분(23:00-07:00은 제외)
schedule=b4, "run batch.kql;", every 6:30 ("FAILED"
 "run copy.kql;");매일 6:30(실패했을 경우 run 구문 실행)
schedule=b5, "run batch.kql;", every Tue 6:30;매주 16:30
schedule=b6, "run batch.kql;", every 5 6:30;매월5일16:30
```

3.7.1. 스케줄 목록 보기 (SHOW SCHEDULES)

현재 등록되어 있는 스케줄 목록을 표시해준다.

- 구문

```
LIST (SHOW) SCHEDULES
```

- 예제

```
----------------- kql.rc -----------------
schedule = batch, "run batch1.kql;", every 4:00 ("FAILED" "exec
sh alarm.sh;")
schedule = insert, "run insert.kql;", every 100 SEC

kql> list schedules;
+----+--------+---------+--------+------+------------+
-----------------+----------------------------------------+
| NO |   NAME |   SCHED | EXCEPT | STAT | COUNT(F,S) |
 ACTION          | POST ACTION                            |
+----+--------+---------+--------+------+------------+
-----------------+----------------------------------------+
|  0 |  batch |    4:00 | -      |   ON |    0 (0,0) |
 run batch1.kql; | "FAILED" "exec sh alarm.sh;", "OK" ""  |
|  1 | insert | 100 SEC | -      |   ON |    0 (0,0) |
 run insert.kql; |                                        |
+----+--------+---------+--------+------+------------+
-----------------+----------------------------------------+
Total 2 schedules.
OK
```

3.7.2. 스케줄 일시 중단 (SCHEDULE OFF)

번호나 이름이 주어진 또는 전체 스케줄의 동작을 일시 중단시킨다.

- 구문

```
SCHEDULE { number | name | * } OFF
```

구성요소	설명
number	스케줄 보기 명령 결과로 나오는 'NO' 컬럼 번호
name	등록된 스케줄 이름
*	모든 스케줄에 대해 적용

- 예제

```
kql> schedule batch off;
OK

kql> list schedules;
+----+--------+---------+------+-----------+-----------------+
| NO | NAME   | SCHED   | STAT | COUNT(F,S) | ACTION          |
+----+--------+---------+------+-----------+-----------------+
| 0  | batch  | 100 SEC | OFF  | 0 (0,0)    | run batch1.kql; |
| 1  | insert | 4:00    | ON   | 0 (0,0)    | run insert.kql; |
+----+--------+---------+------+-----------+-----------------+
Total 2 schedules.
OK

kql> schedule * off;
OK

kql> list schedules;
+----+--------+---------+--------+------+----------------+
-----------------+---------------------------------------+
| NO |   NAME |   SCHED | EXCEPT | STAT |     COUNT(F,S) |
 ACTION          | POST ACTION                           |
+----+--------+---------+--------+------+----------------+
-----------------+---------------------------------------+
|  0 |  batch |    4:00 | -      |  OFF |        0 (0,0) |
 run batch1.kql; | "FAILED" "exec sh alarm.sh;", "OK" "" |
|  1 | insert | 100 SEC | -      |  OFF |        0 (0,0) |
 run insert.kql; |                                       |
```

```
+----+--------+---------+--------+------+----------------+
-----------------+---------------------------------------+
Total 2 schedules.
OK
```

3.7.3. 스케줄 재시작 (SCHEDULE ON)

번호나 이름이 주어진 또는 전체 스케줄을 동작하게 한다.

- 구문

```
SCHEDULE { number | name | * } ON
```

구성요소	설명
number	스케줄 번호(NO 컬럼 값)
name	스케줄 이름
*	모든 스케줄에 적용

- 예제

```
kql> schedule 0 on;
OK

kql> list schedules;
+----+--------+---------+--------+------+----------------+
-----------------+---------------------------------------+
| NO |   NAME |   SCHED | EXCEPT | STAT |     COUNT(F,S) |
 ACTION          | POST ACTION                           |
+----+--------+---------+--------+------+----------------+
-----------------+---------------------------------------+
|  0 |  batch |    4:00 | -      |   ON |        0 (0,0) |
 run batch1.kql; | "FAILED" "exec sh alarm.sh;", "OK" "" |
|  1 | insert | 100 SEC | -      |  OFF |        0 (0,0) |
 run insert.kql; |                                       |
+----+--------+---------+--------+------+----------------+
-----------------+---------------------------------------+
Total 2 schedules.
```

```
OK

kql>
```

3.7.4. 스케줄 동적 등록 (ATTACH SCHEDULE)

엔진을 내리지 않은 상태에서 새로운 스케줄을 등록하려고 할 때 사용하는 명령어이다. 구문 구성 요소에 대한 설명은 앞 섹션의 스케줄 등록 부분을 참고한다. 이렇게 등록한 스케줄은 엔진을 내리면 사라지므로 영속적으로 등록하려면 KQL 설정 파일(kql.rc)에 스케줄을 추가한다.

- 구문

```
ATTACH SCHEDULE name, "action", period
```

- 예제

다음은 KQL 설정 파일에 등록하는 경우에 사용 가능한 구문 형식이다.

```
kql> list schedules;
+----+--------+---------+--------+------+----------------+
-----------------+---------------------------------------+
| NO |   NAME |   SCHED | EXCEPT | STAT |     COUNT(F,S) |
 ACTION          | POST ACTION                           |
+----+--------+---------+--------+------+----------------+
-----------------+---------------------------------------+
|  0 |  batch |    4:00 | -      |   ON |        0 (0,0) |
 run batch1.kql; | "FAILED" "exec sh alarm.sh;", "OK" "" |
|  1 | insert | 100 SEC | -      |   ON |        0 (0,0) |
 run insert.kql; |                                       |
+----+--------+---------+--------+------+----------------+
-----------------+---------------------------------------+
Total 2 schedules.
OK

kql> attach schedule update, "run update.kql;", every 6:00;
OK

kql> list schedules;
```

```
+----+--------+---------+--------+------+----------------+
-----------------+---------------------------------------+
| NO |   NAME |   SCHED | EXCEPT | STAT |     COUNT(F,S) |
 ACTION          | POST ACTION                           |
+----+--------+---------+--------+------+----------------+
-----------------+---------------------------------------+
|  0 |  batch |    4:00 | -      |   ON |        0 (0,0) |
 run batch1.kql; | "FAILED" "exec sh alarm.sh;", "OK" "" |
|  1 | insert | 100 SEC | -      |   ON |        0 (0,0) |
 run insert.kql; |                                       |
|  2 | update |    6:00 | -      |   ON |        0 (0,0) |
 run update.kql; |                                       |
+----+--------+---------+--------+------+----------------+
-----------------+---------------------------------------+
Total 3 schedules.
OK

kql> attach schedule update_wed, "run update.kql;",
every wed 5:00;
OK

kql> attach schedule update_3, "run update.kql;", every 3 5:00;
OK

kql>
```

3.7.5. 스케줄 동적 삭제 (DETACH SCHEDULE)

동적으로 등록된 스케줄을 삭제한다. 영속적으로 삭제하려면 KQL 설정 파일(kql.rc)에서 등록된 스케줄을 삭제해야 한다.

- 구문

```
DETACH SCHEDULE number
```

구성요소	설명
number	스케줄 보기 명령 결과로 나타나는 'NO' 컬럼의 번호

- 예제

  다음은 detach schedule 명령으로 스케줄을 삭제하는 예제이다.

```
kql> show schedules;
+----+--------+---------+--------+------+----------------+
-----------------+--------------------------------------+
| NO |   NAME |   SCHED | EXCEPT | STAT |     COUNT(F,S) |
 ACTION          | POST ACTION                          |
+----+--------+---------+--------+------+----------------+
-----------------+--------------------------------------+
|  0 |  batch |    4:00 | -      |   ON |        0 (0,0) |
 run batch1.kql; | "FAILED" "exec sh alarm.sh;", "OK" "" |
|  1 | insert | 100 SEC | -      |   ON |        0 (0,0) |
 run insert.kql; |                                      |
+----+--------+---------+--------+------+----------------+
-----------------+--------------------------------------+
Total 2 schedules.
OK

kql> detach schedule 1;
OK

kql> show schedules;
+----+-------+-------+--------+------+------------------+
-----------------+--------------------------------------+
| NO |  NAME | SCHED | EXCEPT | STAT |        COUNT(F,S) |
 ACTION          | POST ACTION                          |
+----+-------+-------+--------+------+------------------+
-----------------+--------------------------------------+
|  0 | batch |  4:00 | -      |   ON |           0 (0,0) |
 run batch1.kql; | "FAILED" "exec sh alarm.sh;", "OK" "" |
+----+-------+-------+--------+------+------------------+
-----------------+--------------------------------------+
Total 1 schedules.
OK

kql>
```

3.8. 언어 데이터 제어 명령

RELOAD USER DIC 명령을 이용하여 동적으로 사용자 사전을 등록할 수 있다.

신조어, 불용어, 동의어, 컴파일 된 동의어 사전의 위치는 kql.rc에 각각 다음과 같이 설정할 수 있다.

- 신조어: user_dic_for_coined_word
- 불용어: user_dic_for_stop_word
- 동의어: user_dic_for_synonym
- 컴파일된 동의어: user_dic_for_compiled_synonym

사전이 컴파일되어 있을 경우, 독크루저를 재시작 할 때 해당 사전은 자동으로 적재되므로 RELOAD MODULE 명령어를 실행하지 않아도 된다.

3.8.1. 사용자 사전 - 신조어 (RELOAD USER DIC FOR COINED WORD)

신조어 사전을 등록한다.

신조어란 새로 등장한 어휘를 말한다. 신조어는 검색 서비스가 이루어지고 있는 과정에서 계속 생성되는 특징이 있기 때문에 관리자는 신조어를 필요에 따라 사전 형태로 등록하여 검색 서비스에 적용할 수 있다.

신조어 사용자 사전의 형식은 개행 문자로 구분된 단어 목록이며 복합명사는 띄어쓰기 단위로 입력한다. 품사 태깅도 허용한다. 주석 표시 문자로는 세미콜론(;)을 사용한다.

신조어 파일을 생성, 수정 후 RELOAD 명령을 실행하면 변경된 사전 내용이 반영된다.

- 구문

```
RELOAD USER DIC FOR COINED WORD file_path
[LANGUAGE=language][CHARSET=charset]
```

구성요소	설명
file_path	사전 파일 경로

구성요소	설명
language	사전 파일 언어 (기본값=KOREAN)
charset	사전 파일 문자 세트 (기본값=EUCKR)

- 예제

다음 예제에서 '테스토니'는 '테스'와 '토니'로 추출되었으나 신조어 적용 후에는 '테스토니'로 추출되었다.

```
----------- coined.txt ----------------
테스토니

kql> extract keywords from query '이태리명품 테스토니';
0: '이태리' 0
1: '명품' 0
2: '테스' 1
3: '토니' 1
Total 4 keywords.
OK

kql> reload user dic for coined word 'coined.txt';
OK

kql> extract keywords from query '이태리명품 테스토니';
0: '이태리' 0
1: '명품' 0
2: '테스토니' 1
Total 3 keywords.
OK

kql>
```

다음은 신조어 파일 형식의 예이다.

```
;다마스쿽
나라야
아이템 콜
제주 꽃배달 서비스
반지의제왕 [반지/NN 의/PN 제왕/NN]
```

일본어와 중국어의 경우는 여러 가지 문자 세트 중 어느 것으로 만든 사용자 사전인지 첫줄에 표시해주어야 한다. 불용어, 동의어, 가중어 등 모든 사용자 사전에 해당하는 항목이다.

```
CHARSET = SJIS
東アジア　協議　[東/PF アジア/NN　/NN 協議/NN　/SF]
ミサイル
;スクリ ンショット
```

3.8.2. 사용자 사전 - 불용어 (RELOAD USER DIC FOR STOP WORD)

불용어 사전을 등록한다.

불용어란 색인어 추출에서 제외되거나 검색에서 무시해도 되는 어휘를 말한다. 불용어를 개행 문자로 구분한 사전을 사용하여 검색 시스템에 반영한다.

불용어 사전을 편집한 후 RELOAD 명령어를 실행하면 변경된 사전 내용이 반영된다.

- 구문

```
RELOAD USER DIC FOR STOP WORD file_path
[LANGUAGE=language][CHARSET=charset]
```

RELOAD USER DIC FOR STOP WORD 구문의 구성요소에 대한 설명은 '사용자 사전 - 신조어 (RELOAD USER DIC FOR COINED WORD)'를 참고한다.

- 예제

다음 예제에서는 불용어 사전에 '나', '너'를 추가하고 독크루저에 반영한 후 키워드를 추출한 결과 불용어가 키워드에서 제외되어 있다.

```
---------- stop.txt ----------------
나
너

kql> extract keywords from query '나 너 그리고 우리';
0: '나' 0
1: '너' 1
2: '그리고' 2
3: '우리' 3
Total 4 keywords.
```

```
OK

kql> reload user dic for stop word 'stop.txt';
OK

kql> extract keywords from query '나 너 그리고 우리';
0: '그리고' 2
1: '우리' 3
Total 2 keywords.
OK
```

3.8.3. 사용자 사전 - 동의어 (RELOAD USER DIC FOR SYNONYM)

동의어 사전을 등록한다.

동의어(유의어)란 같은 의미를 가지는 어휘들로서 의미적인 동의어뿐 아니라 약어, 외래어, 한자어, 별명 등 유사한 표현의 검색에 적용 가능하다. 사전 편집 시 동의어들은 콤마(,)로 구분하여 한 줄에 모두 작성한다.

동의어 사전을 편집한 후 RELOAD 명령어를 실행하면 변경된 사전 내용이 반영된다.

- 구문

```
RELOAD USER DIC FOR SYNONYM file_path
[LANGUAGE=language][CHARSET=charset]
```

RELOAD USER DIC FOR SYNONYM 구문의 구성요소에 대한 설명은 '사용자 사전 - 신조어 (RELOAD USER DIC FOR COINED WORD)'를 참고한다.

- 예제

```
------------ synonym.txt ---------------
광개토대왕, 호태왕
산업자원부, 산자부
모차르트, 모짜르트, 모짤트
달걀, 계란
비무장지대, DMZ
나이키, Nike
...

kql> reload user dic for synonym synonym.txt;
```

```
OK

Kql> expand query '달걀';
0 : '달걀'
1 : '계란'
Total 2 synonyms.
OK
```

3.8.4. 사용자 사전 - 컴파일 된 동의어(RELOAD USER DIC FOR COMPILED SYNONYM)

'CONTROL SYNONYM DICTIONARY' 명령으로 컴파일된 동의어 사전을 등록하는 명령어이다.

'RELOAD USER DIC FOR SYNONYM' 명령은 텍스트 형태의 동의어 파일을 컴파일과 동시에 로드하지만, 이 명령은 컴파일되어 있는 바이너리 형태의 동의어 사전을 등록한다는 점에서 다르다. 동의어 데이터 파일이 클 경우 미리 컴파일된 바이너리를 생성해서 로드함으로써, 온라인 서비스시 컴파일링 부하로 인한 시스템 성능 저하를 막을 수 있다.

- 구문

```
RELOAD USER DIC FOR COMPILED SYNONYM file_path [LANGUAGE=language]
[CHARSET=charset]
```

구성요소	설명
file_path	컴파일된 동의어 사전 경로
[LANGUAGE=*language*]	사전 언어 (기본값=KOREAN)
[CHARSET=*charset*]	사전 문자세트 (기본값=EUCKR)

- 예제

```
--------------- compiled_synonym.txt ---------------
광개토대왕, 호태왕
산업자원부, 산자부
모차르트, 모짜르트, 모짤트
달걀, 계란
```

```
비무장지대, DMZ
나이키, Nike
...

kql> control synonym dictionary operation compile
compiled_synonym.txt to
compiled_synonym.dat;
OK

kql> reload user dic for compiled synonym compiled_synonym.dat;
OK

Kql> expand query '달걀';
0 : '달걀'
1 : '계란'
Total 2 synonyms.
OK
```

3.8.5. 동의어 컴파일 (CONTROL SYNONYM DICTIONARY)

텍스트 형태의 동의어 사전을 바이너리 형태로 컴파일한다. 컴파일 후 RELOAD USE DIC FOR COMPILED SYNONYM 명령을 통해 컴파일된 동의어 사전을 등록할 수 있다.

- 구문

```
CONTROL SYNONYM DICTIONARY OPERATION COMPILE text_file_path, ...
TO binary_file_path [LANGUAGE=language] [CHARSET=charset]
```

구성요소	설명
text_file_path	바이너리 포맷으로 변환(컴파일)할 텍스트 사전 경로
binary_file_path	최종 만들어질 바이너리 동의어 사전 파일 경로
[LANGUAGE=*language*]	사용할 언어 (기본값=KOREAN)
[CHARSET=*charset*]	사용할 문자 세트 (기본값=EUCKR)

- 예제

```
kql/tv> control synonym dictionary operation compile
./synonym.txt to synonym.dat;
OK

kql/tv>
```

3.8.6. 형태소 분석 사전(RELOAD SYSTEM DIC)

형태소 분석 사전 업데이트 시 변경된 내용을 엔진에 반영시킨다.

- 구문

```
RELOAD SYSTEM DIC [LANGUAGE=language] [CHARSET=charset]
```

구성요소	설명
[LANGUAGE=*language*]	사용할 언어 (기본값=KOREAN)
[CHARSET=*charset*]	사용할 문자 세트 (기본값=EUCKR)

- 예제

```
kql/tv> reload system dic;
OK

kql/tv>
```

3.8.7. 동의어 확인 (EXPAND QUERY)

질의어가 사용자 동의어 사전을 참고하여 동작하는 '동의어 확장' 결과를 확인한다.

- 구문

```
EXPAND QUERY string
```

3.8.8. 키워드 추출 (EXTRACT KEYWORDS)

주어진 문자열의 형태소를 분석하여 키워드를 추출한다.

- 구문

```
EXTRACT KEYWORDS FROM { QUERY | DOCUMENT } literal
[LANGUAGE=language][CHARSET=charset]
```

구성요소	설명
QUERY	KQL 설정 파일 중 'query_compound_level'에 지정된 색인 레벨에 따라 질의에 대한 색인어를 추출한다.
DOCUMENT	KQL 설정 파일 중 'document_compound_level'에 지정된 색인 레벨에 따라 문서에 대한 색인어를 추출한다.
literal	형태소 분석 대상인 텍스트 문자열, 작은 따옴표(' ')로 감싼다.
language	사용할 언어 (기본값=KOREAN)
charset	사용할 문자 세트 (기본값=EUCKR)

- 예제
  - 다음 예제는 색인 레벨이 5일때 '한국대학교'에 대한 색인어 추출 결과이다. 단, 이 결과는 '한국대학교'의 형태소 분석값이 '한국/명사 대학/명사 교/접미사'라고 가정한 결과이며, 형태소 분석값이 다를 경우 키워드 추출 결과는 달라질 수 있다.

```
kql> extract keywords from document '한국대학교';
0: '대학' (1)
1: '대학교' (1)
2: '한국' (1)
3: '한국대학' (1)
4: '한국대학교' (1)
Total 5 keywords.
OK

kql>
```

    신조어 사전을 편집하면 형태소 분석값을 조정할 수 있다. 예를 들어, '한국대학교'의 형태소 분석 결과가 잘못되었을 경우 '한국대학교 [한국/NN 대학/NN 교/SN]' 엔트리를 신조어 사전에 추가하면 정확한 분석값을 얻을 수 있다.

    다음은 색인 레벨이 2일때의 색인어 추출 결과로 '한국대학', '한국대학교'는 색인어에서 제외되었음을 알 수 있다.

```
kql> extract keywords from document '한국대학교';
0: '대학' (1)
1: '대학교' (1)
2: '한국' (1)
Total 3 keywords.
OK

kql>
```

다음은 문장 전체에서 색인어를 추출한 결과이다.

```
kql> extract keywords from document '그는 평생 코르시카인의 거칠음,
솔직함을 잃지않았다';
0: '거칠음' (1)
1: '그' (1)
2: '솔직' (1)
3: '않았다' (1)
4: '잃지' (1)
5: '코르시카' (1)
6: '코르시카인' (1)
7: '평생' (1)
Total 8 keywords.
OK
```

– EXTRACT KEYWORDS FROM QUERY *literal*

다음은 색인 레벨이 5일때 '한국대학교'에 대한 색인어 추출 결과이다. 질의에 대한 색인어 추출에서는 최장 일치되는 키워드만을 선별하기 때문에 문서로부터 색인어 추출에 포함되었던 '한국', '대학', '대학교' 등은 제외되었다.

```
kql> extract keywords from query '한국대학교';
0: '한국대학교' 0
Total 1 keyword.
OK
```

다음은 색인 레벨이 2일때의 색인어 추출 결과로 역시 최장 일치 원칙에 따라 '대학'은 제외되었다.

```
kql> extract keywords from query '한국대학교';
0: '한국' 0
1: '대학교' 0
```

```
Total 2 keywords.
OK

kql>
```

3.9. 기타 명령

3.9.1. 명령어 일괄 처리 (RUN)

스키마 정의나 데이터 조작 명령어가 아닌 실행하고자 하는 하나 이상의 명령어들이 순차적으로 기록되어 있는 스크립트 파일을 실행한다. RUN 명령은 내포 가능하다. 즉, 스크립트 파일 내에 다시 RUN 명령어가 존재해도 된다.

- 구문

```
RUN kql_file_path
```

구성요소	설명
kql_file_path	스크립트 파일의 절대 경로

- 예제

다음은 초기 볼륨 생성, 테이블 스키마 정의, 인덱스 스키마 정의를 차례로 수행하는 예제이다.

```
---------- /home/my_script.kql -----------
create volume myvol;
use volume myvol;
create table tbl (STR1 STRING PRIMARY, TT TEXT, BD TEXT, TM DATE,
CL INT16);
create index TTBB_idx on tbl (TT, BD);

kql> run /home/konan/my_script.kql;
OK
```

3.9.2. 셸 명령 실행 (EXEC)

KQL 커맨드상에서 모든 셸 명령어를 실행한다.

- 구문

```
EXEC shell_command
```

구성요소	설명
shell_command	실행하려는 셸 명령어

- 예제

```
kql> exec "ls";
crmcli
crzcli
docruzerd
docruzerd.rc
gateway.rc
kql
kql.hst
kql.log
replay
run.kql
kql.rc
vol.rc
OK
```

3.9.3. 문자열 출력 (ECHO)

지정한 문자열을 로그에 출력한다. KQL 스크립트 작성 시 매 단계마다 ECHO 명령어를 사용하면 작업을 추적할 때 편리하다.

- 구문

```
ECHO string
```

구성요소	설명
string	로그에 출력하고자 하는 문자열

- 예제

```
kql> echo "batch job started.";
OK

kql> run run.kql;
OK

kql/myvol> echo "batch job completed.";
OK

------------------------ kql.log -------------------------
[Tue Jul 19 13:55:06 2005] batch job started.
[Tue Jul 19 13:55:49 2005] table is created.
…
Processed        8/9 Tue Jul 19 13:56:14 2005
================ END OF POST INDEXING (tbl) ====================
(2 Segments, 15.11 MB Used)
[Tue Jul 19 13:56:14 2005] TR95 ended [disk=(2,64) age=96
 committed=(96,1.50MB)].
[Tue Jul 19 13:56:14 2005] records are imported through gateway.
[Tue Jul 19 13:56:26 2005] batch job completed.
```

3.9.4. 시간 출력 (TIME)

각 KQL 구문 수행 처리에 걸린 시간을 표시할지 여부를 지정한다.

- 구문

```
TIME [ON | OFF]
```

구성요소	설명
ON	구문의 처리시간을 표시함

구성요소	설명
OFF	구문의 처리시간을 표시하지 않음

- 예제

```
kql> time on
Time on.

kql> run run.kql;
23.390 sec
OK
```

3.9.5. 자동 완성 (AUTOFILL)

문장 자동완성 기능의 사용 여부를 지정한다. UNIX 계열에서는 기본으로 KQL 명령어의 일부만 입력해도 자동완성 시켜주는 기능이 있다.

- 구문

```
AUTOFILL [ON | OFF]
```

구성요소	설명
ON	문장 자동완성 기능을 사용
OFF	문장 자동완성 기능을 사용하지 않음

- 예제

```
kql > autofill off
Auto-filling disabled.

kql> use volume myvol;
OK

kql/myvol> autofill on
Auto-filling enabled.
```

```
kql/myvol> use volume myvol;  // 'us'까지만 입력해도 'use volume'까지
OK                            // 자동완성된다.
```

3.9.6. 동적 설정 변경 (RECONFIGURE)

엔진 실행 중에 재가동 없이 kql.rc의 설정을 변경한다.

- 구문

```
RECONFIGURE <kql_conf>
```

- 예제

```
kql> reconfigure send_volume_bit_rate = 4;
OK

kql> reconfigure max_buffer_cache_size = 128 MB;
Error: 'max_buffer_cache_size' not reconfigurable.
Operation cancelled.
```

모든 설정 변수가 동적으로 변경 가능한 것은 아니고, 동적으로 변경해도 문제가 없는 몇몇 설정 변수만 RECONFIGURE 명령어로 변경할 수 있다. 이외의 설정변수를 변경하려면 kql.rc를 직접 수정한 후 엔진을 재시작해야 한다.

RECONFIGIGURE로 변경 가능한 명령어 목록 예시는 다음과 같다.

```
CONNECTION_TIME_OUT
LONG_QUERY_MIN_LENGTH
RECV_VOLUME_BIT_RATE
RETRIEVAL_TIME_OUT
RLIMIT_CPU
SEND_VOLUME_BIT_RATE
SYS_LOG
VOLUME_WAIT_TIME
...
```

3.9.7. 명령 대기 모드(PROMPT)

명령 실행 후 결과 페이지가 1페이지가 넘어가는 경우 더 보기를 할지 말지에 대해 사용자 입력을 대기하지 않고 명령을 끝낸다.

- 구문

```
prompt
```

- 예제

```
kql> prompt
Interactive mode off.

kql> prompt
Interactive mode on.
```

검색과 응용

이 장에서는 독크루저를 이용한 다양한 검색 방법과 정렬 방법에 대해 소개한다.

4.1. 검색 방법

독크루저는 초보적인 검색엔진 사용자부터 전문적인 정보 검색사에 이르기까지 다양한 유형의 사용자들이 쉽고도 정확하게 원하는 정보를 검색할 수 있도록 많은 검색 방법을 제공한다. 독크루저는 다른 시스템에 비해 월등히 많은 검색 방법을 지원하는데, 시스템 환경과 사용자의 필요에 따라 적절한 검색 방법을 선택하면 최상의 결과를 얻을 수 있다.

4.1.1. 자연어 검색 (NATURAL)

독크루저의 기본 검색 방법으로, 정보 검색에 익숙하지 않은 사용자들을 위해 일상 생활에서 사용하는 자연어 문장을 그대로 받아들인 후 시스템 내부에서 자동으로 여러 조건들을 고려하여 우선순위를 계산한 후 사용자에게 결과를 제시한다. 문장으로 찾는다는 말로 유명해진 이 방식은 사용의 편리성과 정확도 면에서 탁월함이 입증되었다. 대부분의 경우 단순검색은 '자연어 검색'으로 하는 것이 좋다. 색인 필드가 TEXT 타입인 경우에 적절하다.

- 구문

```
SELECT * FROM table WHERE field = 'query_word' NATURAL
```

자연어 검색은 다음과 같이 일상언어 문장으로 질의어를 주고 결과를 얻는다.

```
"세계의 문화 유산으로 지정된 한국 문화재는 ?"
"자동차 배출 가스가 환경에 미치는 영향은 ?"
"이승엽의 연간 수입은 ?"
"부시 정권의 대북 정책에 대하여"
"날씨와 주가의 관계"
"청소년 보호 프로그램"
"세계의 문화 유산으로 지정된 한국 문화재"
"유승준 뮤직 비디오"
"2004년 올림픽 개최지"
```

```
"노근리 사건에 대한 정부의 공식 입장"
"국내에서 번지 점프할 수 있는 장소"
```

검색엔진은 이 질의어에 대해 형태소 분석을 하여 내부적으로 최적의 연산식을 생성한 다음 검색한다. 따라서 대부분의 경우는 자연어 검색만으로도 원하는 결과를 얻을 수 있다.

4.1.2. 고급 검색 (ALLWORD, SOMEWORD, ANYWORD, ...)

기본적인 자연어 검색 외에 약간의 검색 조건들을 추가한 고급 검색이 있다. 고급 검색의 조건들은 자연어 검색의 결과를 여과(filtering)해 주는 기능이 있다. 자연어 검색의 정밀도를 유지한 채 파일 이름 검색 등 특수한 목적으로 사용할 수 있다. 고급 검색 방법은 자연어 검색 방법을 개량하여 특정 상황에 적합하도록 만든 것으로 볼 수 있다. 따라서 검색 상황이 좀 특수한 편이라면 고급 검색을 고려해 보는 것이 좋다. 고급 검색 방법을 사용하여 검색하는 경우는 검색 대상 필드가 TEXT 타입인 경우에 적절하다.

- 구문

```
SELECT {*, field_name,…} FROM table_name WHERE {field_name |
 index_name} = 'query_word' {ALLWORD | SOMEWORD | ANYWORD | …}
```

각 검색 방법들을 비교하기 위하여 다음과 같이 테이블과 인덱스를 만들고 데이터를 입력한다.

```
kql> create volume tv4;
kql> use volume tv4;
kql/tv4> create table tb (rowid string, fd text, fd2 text);
kql/tv4> create index idx on tb(fd, fd2);
kql/tv4> insert into tb set rowid= '1', fd='1779년 아버지를 따라
프랑스에 건너가, 10세 때 브리엔 유년학교에 입학하여 5년간 기숙사
생활을 하였다. 코르시카 방언으로 프랑스어 회화에 고민하며 혼자
도서실에서 역사책을 읽는 재미로 지냈으나, 수학만은 뛰어난 성적을
보였다.', fd2='백과사전';
kql/tv4> insert into tb set rowid='2', fd='1784년 파리육군사관학교에
입학, 임관 뒤 포병소위로 지방연대에 부임하였다. 1789년 프랑스혁명 때
코르시카로 귀향하여, 파올리 아래서 코르시카국민군 부사령에 취임하였다.
프랑스 육군은 3회에 걸친 군대이탈과 2중군적에 대해 휴직을 명하였다.',
```

```
fd2='백과사전';
.........................................
```

생성된 입력 테이블과 인덱스는 다음과 같다.

rowid	fd	fd2
1	1779 년 아버지를 따라 프랑스에 건너가, 10 세 때 브리엔 유년학교에 입학하여 5 년간 기숙사 생활을 하였다. 코르시카 방언으로 프랑스어 회화에 고민하며 혼자 도서실에서 역사책을 읽는 재미로 지냈으나, 수학만은 뛰어난 성적을 보였다.	백과사전
2	1784 년 파리육군사관학교에 입학, 임관 뒤 포병소위로 지방연대에 부임하였다. 1789 년 프랑스혁명 때 코르시카로 귀향하여, 파올리 아래서 코르시카국민군 부사령에 취임하였다. 프랑스 육군은 3 회에 걸친 군대이탈과 2 중군적에 대해 휴직을 명하였다.	백과사전
3	1792 년 파올리와 결별하고, 일가와 함께 프랑스로 이주하였다. 1793 년 가을 툴롱항구 왕당파반란을 토벌하는 여단 부관으로 복귀하여, 최초의 무훈을 세웠다.	백과사전
4	F.로베스피에르의 아우와 지우(知遇)를 갖게 되어 이탈리아 국경군의 지휘를 맡았다. 테르미도르(Thermidor)의 반동 쿠데타로 로베스피에르파(派)로 몰려 체포되어 다시 실각, 1 년간 허송세월을 보냈다. 1795 년 10 월 5 일(방데미에르 13 일), 파리에 반란이 일어나 국민공회(國民公會)가 위기에 직면하자, 바라스로부터 구원을 요청받고, 포격으로 폭도들을 물리쳤다.	백과사전
5	이 기민한 조치로 재기의 기회를 포착, 1796 년 3 월 바라스의 정부(情婦)이자 사교계의 꽃이던 조제핀과 결혼, 총재정부로부터 이탈리아 원정군사령관으로 임명되었다. 이탈리아에서 오스트리아군을 격파하여 5 월에 밀라노에 입성, 1797 년 2 월에는 만토바를 점령하는 전과를 올렸다.	백과사전
6	10 월 오스트리아와 캄포포르미오(Campoformio)조약을 체결하여, 이탈리아 각지에 프랑스혁명의 이상을 도입한 인민공화국을 건설하였다. 그의 명성은 프랑스에서도 한층 높아졌다. 하루 3 시간만 잔다는 소문도 있었으나, 비서 브리센에 의하면 건강에 항상 신경을 써서 하루 8 시간은 잤다고 한다.	백과사전

rowid	fd	fd2
7	1798 년 5 월 5 만여 명의 병력을 이끌고 이집트를 원정하여 결국 카이로에 입성하였다. 7 월 해군이 아부키르만(灣)에서 영국함대에 패하여 본국과의 연락이 끊기자 혼자서 이집트를 탈출, 10 월에 프랑스로 귀국하였다. 곧 그를 통해 총재정부를 타도하려는 세이에스·탈레랑 등의 음모에 말려들었다. 1799 년 11 월 9 일(브뤼메르 18 일) 군을 동원, 500 인회를 해산시켜 원로원으로부터 제 1 통령으로 임명되고, 군사독재가 시작되었다.	백과사전
8	그는 평생 코르시카인의 거칠음·솔직함을 잃지 않아, 농민출신 사병들로부터 신뢰를 받고 있었으나, 역사적 영웅으로 보면 인간성을 무시하고 도덕성이 결여된 행동의 주인공이었다. 광대한 구상력, 끝없는 현실파악의 지적 능력, 감상성 없는 행동력은 마치 마력적이라고 할 정도였다. 이처럼 사상 유례 없는 개성이 혁명 후의 안정을 지향하는 과도기의 사회상황에서 보나파르티즘이라는 나폴레옹의 정치방식이 확립되었다. 제 1 통령으로서 국정정비·법전편찬에 임하고, 대(對)오스트리아와의 결전을 서둘러 1800 년 알프스를 넘어 마렝고에서 전승을 이룩하였다.	백과사전

아무 단어나 포함 - ANYWORD

```
kql/tv4> select fd from tb where fd='이집트 공화국' anyword;
-------------------- 1 of total 2 (ROWID 6) --------------------
fd (398): 1798년 5월 5만여 명의 병력을 이끌고 이집트를 원정하여 결국
카이로에 입성하였다. 7월 해군이 아부키르만(灣)에서 영국함대에 패하여
본국과의 연락이 끊기자 혼자서 이집트를 탈출, 10월에 프랑스로 귀국하였다.
곧 그를 통해 총재정부를 타도하려는 세이에스•탈레랑 등의 음모에 말려들었다.
1799년 11월 9일(브뤼메르 18일) 군을 동원, 500인회를 해산시켜
원로원으로부터 제1통령으로 임명되고, 군사독재가 시작되었다.
-------------------- 2 of total 2 (ROWID 5) --------------------
fd (275): 10월 오스트리아와 캄포포르미오(Campoformio)조약을 체결하여,
이탈리아 각지에 프랑스혁명의 이상을 도입한 인민공화국을 건설하였다.
그의 명성은 프랑스에서도 한층 높아졌다. 하루 3시간만 잔다는 소문도
있었으나, 비서 브리센에 의하면 건강에 항상 신경을 써서 하루 8시간은
잤다고 한다.
Total 2 records.
OK
```

모든 단어 포함(순서 없이) - ALLWORD

사용자가 입력한 질의어에 포함된 검색어가 모두 포함된 문서를 찾는다. 불리언 검색의 AND와 같은 조건이다. 아래 예는 테이블 tb의 각 레코드들 중에서 필드 fd에 '로베스피에르'와 '아우'라는 단어가 모두 존재하는 것들을 검색한 결과를 보여준다. 순서가 바뀌어도 검색된다.

```
kql/tv4> select fd from tb where fd='로베스피에르의 아우' allword;
-------------------- 1 of total 1 (ROWID 3) --------------------
fd (339): F.로베스피에르의 아우와 지우(知遇)를 갖게 되어 이탈리아 국경군의
지휘를 맡았다. 테르미도르(Thermidor)의 반동 쿠데타로 로베스피에르파(派)로
몰려 체포되어 다시 실각, 1년간 허송세월을 보냈다. 1795년 10월 5일(방데미에르
13일), 파리에 반란이 일어나 국민공회(國民公會)가 위기에 직면하자, 바라스로부터
구원을 요청받고, 포격으로 폭도들을 물리쳤다.
Total 1 record.
OK

kql/tv4> select fd from tb where fd='아우 로베스피에르의' allword;
-------------------- 1 of total 1 (ROWID 3) --------------------
fd (339): F.로베스피에르의 아우와 지우(知遇)를 갖게 되어 이탈리아 국경군의
지휘를 맡았다. 테르미도르(Thermidor)의 반동 쿠데타로 로베스피에르파(派)로
몰려 체포되어 다시 실각, 1년간 허송세월을 보냈다. 1795년 10월 5일(방데미에르
13일), 파리에 반란이 일어나 국민공회(國民公會)가 위기에 직면하자, 바라스로부터
구원을 요청받고, 포격으로 폭도들을 물리쳤다.
Total 1 record.
OK
```

모든 단어 포함(순서 없이, 25 단어 안에) - ALLIN25

사용자가 입력한 질의어에 포함된 검색어가 순서에 상관없이 25 단어 안에 포함된 문서를 찾는다. 아래 예는 테이블 tb의 각 레코드들 중에서 필드 fd에 '마렝고'와 '혁명'이라는 단어가 25 단어 이내에 존재하는 문서들을 검색한 결과를 보여준다. '평생'과 '마렝고'로 검색을 하면 25 단어를 넘기 때문에 검색되지 않는다.

```
kql/tv4> select fd from tb where fd='마렝고 혁명' allin25;
-------------------- 1 of total 1 (ROWID 7) --------------------
fd (526): 그는 평생 코르시카인의 거칠음•솔직함을 잃지 않아, 농민출신
사병들로부터 신뢰를 받고 있었으나, 역사적 영웅으로 보면 인간성을 무시하고
도덕성이 결여된 행동의 주인공이었다. 광대한 구상력, 끝없는 현실파악의
지적 능력, 감상성 없는 행동력은 마치 마력적이라고 할 정도였다. 이처럼 사상
유례 없는 개성이 혁명 후의 안정을 지향하는 과도기의 사회상황에서
```

```
보나파르티즘이라는 나폴레옹의 정치방식이 확립되었다.
제1통령으로서 국정정비•법전편찬에 임하고, 대(對)오스트리아와의 결전을 서둘러
1800년 알프스를 넘어 마렝고에서 전승을 이룩하였다.
Total 1 record.
OK

kql/tv4> select fd from tb where fd='평생 마렝고' allin25;
Total 0 record.
OK
```

모든 단어 포함(순서없이, 인접해서) - ALLADJACENT

사용자가 입력한 질의어에 포함된 검색어가 순서에 상관없이 인접해서 나타난 문서를 찾는다. 아래 예는 테이블 tb의 각 레코드들 중에서 필드 fd 에 '코르시카'와 '방언'이라는 단어가 인접해서 존재하는 문서들을 검색한 결과를 보여준다.'코르시카'와 '프랑스어'는 인접해 있지 않기 때문에 검색되지 않는다. 노래제목이나 파일명을 찾는 검색에서 유용하게 쓰인다.

```
kql/tv4> select fd from tb where fd='코르시카 방언' alladjacent;
-------------------- 1 of total 1 (ROWID 0) --------------------
fd (221): 1779년 아버지를 따라 프랑스에 건너가, 10세 때 브리엔 유년학교에
입학하여 5년간 기숙사 생활을 하였다. 코르시카 방언으로 프랑스어 회화에 고민하며
혼자 도서실에서 역사책을 읽는 재미로 지냈으나, 수학만은 뛰어난 성적을 보였다.
Total 1 record.
OK

kql/tv4> select fd from tb where fd='코르시카 프랑스어' alladjacent;
Total 0 record.
OK
```

모든 단어 포함(순서대로, 인접해서) - ALLORDERADJACENT

사용자가 입력한 질의어에 포함된 검색어가 순서대로 인접해서 나타난 문서를 찾는다. 아래 예는 테이블 tb의 각 레코드들 중에서 필드 fd에 '도서실'과 '역사책'이라는 단어가 순서대로 인접해서 존재하는 문서들을 검색한 결과를 보여준다. 순서를 바꾸어 '역사책을 도서실에서'로 검색하면 검색되지 않는다.

```
kql/tv4> select fd from tb where fd='도서실에서 역사책을'
allorderadjacent;
-------------------- 1 of total 1 (ROWID 0) --------------------
fd (221): 1779년 아버지를 따라 프랑스에 건너가, 10세 때 브리엔 유년학교에
```

```
입학하여 5년간 기숙사 생활을 하였다. 코르시카 방언으로 프랑스어 회화에 고민하며
혼자 도서실에서 역사책을 읽는 재미로 지냈으나, 수학만은 뛰어난 성적을 보였다.
Total 1 record.
OK

kql/tv4> select fd from tb where fd='역사책을 도서실에서'
allorderadjacent;
Total 0 record.
OK
```

모든 단어 포함(인덱스 전체에서) - ALLWORDTHRUINDEX

사용자가 입력한 질의어에 포함된 검색어가 인덱스 전체에서 모두 포함된 문서를 찾는다. 이 검색 방법은 복합인덱스를 검색할 때 유용하다. 아래 예는 테이블 tb의 각 레코드들 중에서 2개의 필드 fd, fd2를 묶은 복합인덱스 idx에서 '나폴레옹', '백과사전' 모두 포함된 문서를 검색한 결과를 보여준다. '모든 단어포함'인 allword로는 검색이 되지 않는 점에 유의해야 한다.

```
kql/tv4> select fd, fd2 from tb where idx='나폴레옹 백과사전' allword;
Total 0 record.
OK

kql/tv4> select fd, fd2 from tb where idx='나폴레옹 백과사전'
allwordthruindex;
-------------------- 1 of total 1 (ROWID 7) --------------------
fd (526): 그는 평생 코르시카인의 거칠음•솔직함을 잃지 않아, 농민출신
사병들로부터 신뢰를 받고 있었으나, 역사적 영웅으로 보면 인간성을 무시하고
도덕성이 결여된 행동의 주인공이었다. 광대한 구상력, 끝없는 현실파악의
지적 능력, 감상성 없는 행동력은 마치 마력적이라고 할 정도였다. 이처럼 사상
유례 없는 개성이 혁명 후의 안정을 지향하는 과도기의 사회상황에서
보나파르티즘이라는 나폴레옹의 정치방식이 확립되었다.
제1통령으로서 국정정비•법전편찬에 임하고, 대(對)오스트리아와의 결전을 서둘러
1800년 알프스를 넘어 마렝고에서 전승을 이룩하였다.
fd2 (8): 백과사전
Total 1 record.
OK
```

이 검색 방법은 각 필드를 합하여 하나의 필드처럼 보고 여기에서 검색하는 것과 같은 효과를 준다. 예를 들어 저자와 책 제목을 같은 인덱스로 묶고 이 검색 방법을 사용하면, 사용자로부터 저자 이름과 책 제목을 한 질의어로 받아서 직접적으로 검색 결과를 얻는다.

최소 몇 단어 이상 포함 - SOMEWORD

사용자가 입력한 질의어에 포함된 검색어 개수가 최소 몇 개 이상 포함된 문서를 찾아준다. 아래 기준표를 보면 검색어가 3 개일 경우는 최소 2개 포함된 문서까지 검색하고 검색어가 5 개일 경우는 최소 3 개 포함된 문서까지 검색한다. 질의어에 포함된 검색어 수는 질의어에 대한 형태소 분석 결과에 따른다.

검색어 수	1	2	3	4	5	6	7	8	9	10
포함 키워드 수	1	2	2	2	3	3	4	4	4	5

아래 예는 테이블 tb의 각 레코드들 중 필드 fd에서 '이탈리아 이집트 오스트리아'라는 질의어에 포함된 '이탈리아', '이집트', '오스트리아' 3 개의 검색어에 대해 최소 2 개 이상의 검색어가 포함된 문서들을 검색한 결과를 보여준다.

```
kql/tv4> select fd from tb where fd='이탈리아 이집트 오스트리아'
someword;
-------------------- 0 of total 2 (ROWID 4) --------------------
fd (261): 이 기민한 조치로 재기의 기회를 포착, 1796년 3월 바라스의
정부(情婦)이자 사교계의 꽃이던 조제핀과 결혼, 총재정부로부터 이탈리아
원정군사령관으로 임명되었다.
이탈리아에서 오스트리아군을 격파하여 5월에 밀라노에 입성, 1797년 2월에는
만토바를 점령하는 전과를 올렸다.
-------------------- 1 of total 2 (ROWID 5) --------------------
fd (275): 10월 오스트리아와 캄포포르미오(Campoformio)조약을 체결하여,
이탈리아 각지에 프랑스혁명의 이상을 도입한 인민공화국을 건설하였다. 그의 명성은
프랑스에서도 한층 높아졌다. 하루 3시간만 잔다는 소문도 있었으나, 비서 브리센에
의하면 건강에 항상 신경을 써서 하루 8시간은 잤다고 한다.
Total 2 records.
OK

kql/tv4> select fd from tb where fd='이탈리아 이집트 오스트리아'
anyword;
-------------------- 0 of total 5 (ROWID 4) --------------------
fd (261): 이 기민한 조치로 재기의 기회를 포착, 1796년 3월 바라스의
정부(情婦)이자 사교계의 꽃이던 조제핀과 결혼, 총재정부로부터 이탈리아
원정군사령관으로 임명되었다.
이탈리아에서 오스트리아군을 격파하여 5월에 밀라노에 입성, 1797년 2월에는
만토바를 점령하는 전과를 올렸다.
-------------------- 1 of total 5 (ROWID 5) --------------------
fd (275): 10월 오스트리아와 캄포포르미오(Campoformio)조약을 체결하여,
```

```
이탈리아 각지에 프랑스혁명의 이상을 도입한 인민공화국을 건설하였다. 그의 명성은
프랑스에서도 한층 높아졌다. 하루 3시간만 잔다는 소문도 있었으나, 비서 브리센에
의하면 건강에 항상 신경을 써서 하루 8시간은 잤다고 한다.
-------------------- 2 of total 5 (ROWID 6) --------------------
fd (398): 1798년 5월 5만여 명의 병력을 이끌고 이집트를 원정하여 결국
카이로에 입성하였다. 7월 해군이 아부키르만(灣)에서 영국함대에 패하여 본국과의
연락이 끊기자 혼자서 이집트를 탈출, 10월에 프랑스로 귀국하였다. 곧 그를 통해
총재정부를 타도하려는 셰이에스 •탈레랑 등의 음모에 말려들었다. 1799년 11월 9일
(브뤼메르 18일) 군을 동원, 500인회를 해산시켜 원로원으로부터 제1통령으로
임명되고, 군사독재가 시작되었다.
-------------------- 3 of total 5 (ROWID 3) --------------------
fd (339): F.로베스피에르의 아우와 지우(知遇)를 갖게 되어 이탈리아 국경군의
지휘를 맡았다. 테르미도르(Thermidor)의 반동 쿠데타로 로베스피에르파(派)로
몰려 체포되어 다시 실각, 1년간 허송세월을 보냈다. 1795년 10월 5일
(방데미에르 13일), 파리에 반란이 일어나 국민공회(國民公會)가 위기에 직면하자,
바라스로부터 구원을 요청받고, 포격으로 폭도들을 물리쳤다.
-------------------- 4 of total 5 (ROWID 7) --------------------
fd (526): 그는 평생 코르시카인의 거칠음•솔직함을 잃지 않아, 농민출신
사병들로부터 신뢰를 받고 있었으나, 역사적 영웅으로 보면 인간성을 무시하고
도덕성이 결여된 행동의 주인공이었다. 광대한 구상력, 끝없는 현실파악의
지적 능력, 감상성 없는 행동력은 마치 마력적이라고 할 정도였다. 이처럼 사상
유례 없는 개성이 혁명 후의 안정을 지향하는 과도기의 사회상황에서
보나파르티즘이라는 나폴레옹의 정치방식이 확립되었다.
제1통령으로서 국정정비•법전편찬에 임하고, 대(對)오스트리아와의 결전을 서둘러
1800년 알프스를 넘어 마렝고에서 전승을 이룩하였다.
Total 5 records.
OK

kql/tv4> select fd from tb where fd='이탈리아 이집트 오스트리아'
allword;
Total 0 record.
OK
```

anyword로 검색하면 각 검색어 중 어느 것이라도 포함된 문서를 모두 가져오고, allword로 검색하면 모든 검색어를 포함한 문서가 없기 때문에 검색 결과가 없는 것에 유의한다. someword는 사용자가 검색한 결과를 적절한 개수로 유지해주는 효과가 있다.

최소 몇 단어 이상 포함(인덱스 전체에서) - SOMEWORDTHRUINDEX

사용자가 입력한 질의어에 포함된 검색어 개수가 인덱스 전체에서 최소 몇 개 이상 포함된 문서를 찾아준다. 아래 기준표를 보면 검색어가 3개일 경우는 최소 2개 포함된 문서까지 검색하고 검색어가 5개일 경우는 최소 3개 포함된 문서까지 검색한다.

검색어 수	1	2	3	4	5	6	7	8	9	10
포함 키워드 수	1	2	2	2	3	3	4	4	4	5

다음은 테이블 tb의 각 레코드들 중에서 2 개의 필드 fd, fd2를 묶은 인덱스 idx 에서 '나폴레옹', '백과사전', '조제핀' 중 2 개 이상을 포함한 문서를 검색한 결과를 보여준다. 한 필드에서 '최소 몇 단어이상 포함'인 someword로는 검색이 되지 않는 점에 유의한다.

```
kql/tv4> select fd,fd2 from tb where idx='나폴레옹 백과사전 조제핀'
someword;
Total 0 record.
OK

kql/tv4> select fd,fd2 from tb where idx='나폴레옹 백과사전 조제핀'
somewordthruindex;
-------------------- 0 of total 2 (ROWID 4) --------------------
fd (261): 이 기민한 조치로 재기의 기회를 포착, 1796년 3월 바라스의
정부(情婦)이자 사교계의 꽃이던 조제핀과 결혼, 총재정부로부터 이탈리아
원정군사령관으로 임명되었다.
이탈리아에서 오스트리아군을 격파하여 5월에 밀라노에 입성, 1797년 2월에는
만토바를 점령하는 전과를 올렸다.
fd2 (8): 백과사전
-------------------- 1 of total 2 (ROWID 7) --------------------
fd (526): 그는 평생 코르시카인의 거칠음•솔직함을 잃지 않아, 농민출신
사병들로부터 신뢰를 받고 있었으나, 역사적 영웅으로 보면 인간성을 무시하고
도덕성이 결여된 행동의 주인공이었다. 광대한 구상력, 끝없는 현실파악의
지적 능력, 감상성 없는 행동력은 마치 마력적이라고 할 정도였다. 이처럼 사상
유례 없는 개성이 혁명 후의 안정을 지향하는 과도기의 사회상황에서
보나파르티즘이라는 나폴레옹의 정치방식이 확립되었다.
제1통령으로서 국정정비•법전편찬에 임하고, 대(對)오스트리아와의 결전을 서둘러
1800년 알프스를 넘어 마렝고에서 전승을 이룩하였다.
fd2 (8): 백과사전
Total 2 records.
OK
```

4.1.3. 불리언 검색 (BOOLEAN)

자연어 검색은 검색엔진이 스스로 사용자 질의어를 분석하여 찾는 방식이고 고급 검색은 여기에 약간의 제약조건들을 덧붙이는 방식이다. 그러나 전문적인 정보 검색사이거나 특수한 상황이라면 검색엔진의 질의어 분석기를 통과하지 않고 직접 형식질의어를 만들어 그대로 찾기를 원할 것이다. 불리언 검색을 사용하기 위해서는 일단 불리언식에 익숙해야 하며 연산식을 만들 때 주의해야 한다. 조금만 오류가 있어도 검색엔진에서 결과가 나오지 않는다.

- 구문

```
SELECT * FROM table WHERE index
 = 'query_word & query_word' BOOLEAN
```

독크루저는 다음과 같은 각종 연산자들을 지원한다. 연산자들을 잘 조합하여 사용하면 정교한 검색을 할 수 있으나 잘 사용하지 못할 경우는 검색 결과가 엉망이 될 수 있다. 연산자뿐만 아니라 키워드 선정에도 주의를 기울여야 한다.

불리언 연산자는 다음과 같은 것들이 있다.

이름	표기법	의미	질의어의 예
AND	str1 & str2	str1과 str2가 모두 포함된 문서를 검색	꽃 & 나비
OR	str1 | str2	str1이나 str2가 포함된 문서를 검색	꽃 | 나비
ANDNOT	str1 ! str2	str1은 포함되어 있으나 str2는 포함되어 있지 않은 문서를 검색	꽃 ! 나비
WITHIN	str1 ~n str2	str1과 str2가 순서대로 n 이내에 위치한 문서를 검색	꽃 ~7 나비
NEAR			꽃 ^7 나비
좌측절단	*str	str로 끝나는 키워드를 포함하는 문서를 검색	*대학교
우측절단	str*	str로 시작하는 키워드를 포함하는 문서를 검색	서울*
양측절단	*str*	str을 포함하는 키워드를 가지고 있는 문서를 검색	*정보*
중간절단	str1*str2	str1으로 시작하고 str2로 끝나는 키워드를 포함하는 문서 검색	연*소

불리언 검색은 TEXT 타입의 필드에 대해서만 사용 가능하며 쿼리를 형태소분석 하지 않는다. WHERE절에 명시된 쿼리에서 공백은 '&' 연산자로 처리한다. 불리언 검색의 예는 다음과 같다.

필드명	필드 타입	인덱스명	필드명
Key	INT32 NULL	total_idx	fd1 fd2
fd1	TEXT NULL		
fd2	TEXT NULL		
dt	DATE NULL		

key	fd1	fd2	dt
1	대한 민국		20090920
2	나라 한국	대한	20090923
3	미국 대한	북한	20091002
4	북한 대한	나라	20091003
5	나라 민국	미국	20091010
6	대한민국		20091011
7	대한 민국	미국북한	20091018
8	한국	민국 대한	20091109

다음 예는 날짜로 정렬하면서 fd1, fd2 필드의 복합 인덱스인 total_idx에 '나라', '민국', '미국' 키워드가 하나 이상 있으면서 '대한' 키워드가 존재하고 '북한' 키워드가 없는 레코드를 검색한다. 불리언 검색은 다른 정렬 옵션들과도 함께 사용될 수 있다.

```
kql/myvol>> select * from tbl where total_idx='(나라|민국|미국)&대한!
북한' boolean order by dt;
---------- 0 of total 2 (ROWID 0) ---------
key (1): 1
fd1 (9): 대한 민국
fd2 (0):
dt (8): 20090920
code (1): 1
---------- 1 of total 2 (ROWID 7) ---------
key (1): 8
fd1 (4): 한국
```

```
fd2 (9): 민국 대한
dt (8): 20091109
code (1): 2
Total 2 records.
OK
```

4.1.4. 유사문서 검색 (SIMILAR)

사용자는 검색 결과 출력된 문서들을 훑어 보다가 마음에 드는 문서를 발견하고 이와 유사한 문서들을 더 얻기를 원하는 경우가 있다. 일반 검색이 사용자의 질의어에 합당한 문서들을 검색해 내는 데 반해 유사문서 검색은 주어진 문서와 유사한 문서들을 검색한다. 즉, 문서에 의한 질의(query by document)이다.

- 구문

```
SELECT {*|field_name, …} FROM table_name WHERE field_name = ROWID
{SIMILAR | SIMILAR2}
```

유사문서 검색은 질의어 검색에 비해 몇 가지 특징을 가지고 있다. 질의어 검색에서는 비교적 적은 수의 단어들과 그들간의 거리 및 순서 정보를 바탕으로 검색을 한다. 그러나 유사문서 검색은 통계적인 방법을 사용하여 비교적 많은 수의 질의어들을 추출하고 그들간의 거리 및 위치정보는 무시한다. 사용자가 고르는 질의어는 개인의 개념 스키마가 반영되어 있는 반면에 유사문서 검색은 대상 문서 집합에서 단어의 통계적 출현 빈도수가 중요한 역할을 한다.

다음은 TT 필드에서 '물리학'과 관련된 문서를 찾아본 결과이다.

```
kql/news_vol> select TT from GISA where TT='물리학';
---------------- 1 of total 35 (ROWID 449848) --------------------
TT (56): 재미있는 물리학 대중화에 앞장선 한국물리학회 송희성 회장
---------------- 2 of total 35 (ROWID 19334) --------------------
TT (40): 노벨 물리학상 네덜란드 두 과학자에 (2보)
---------------- 3 of total 35 (ROWID 19344) --------------------
TT (66): <긴급>올 노벨 물리학상 게라르두스 후프트와 마르티누스 벨트만 수상
---------------- 4 of total 35 (ROWID 19351) --------------------
TT (40): 노벨 물리학상 네덜란드 두 과학자에 (3보)
---------------- 5 of total 35 (ROWID 19366) --------------------
TT (31): <올해 노벨물리학상 수상자 업적>
---------------- 6 of total 35 (ROWID 19371) --------------------
TT (40): 노벨 물리학상 네덜란드 두 과학자에 (4보)
```

```
---------------- 7 of total 35 (ROWID 19495) --------------------
TT (55): 분자물리이론 수학적 토대 닦아...노벨 물리학 수상자 공적
---------------- 8 of total 35 (ROWID 19501) --------------------
TT (55): 분자물리이론 수학적 토대 닦아...노벨 물리학 수상자 공적
---------------- 9 of total 35 (ROWID 19547) --------------------
TT (33): <아쉬움 더해주는 올 노벨물리학상>
```

만약 이 중에서 ROWID가 19334인 두 번째 문서와 유사한 문서를 찾고 싶다면 다음과 같이 찾을 수 있다.

```
kql/news_vol> select TT from GISA where TT=19334 similar;
---------------- 1 of total 4508 (ROWID 19334) --------------------
TT (40): 노벨 물리학상 네덜란드 두 과학자에 (2보)
---------------- 2 of total 4508 (ROWID 19351) --------------------
TT (40): 노벨 물리학상 네덜란드 두 과학자에 (3보)
---------------- 3 of total 4508 (ROWID 19371) --------------------
TT (40): 노벨 물리학상 네덜란드 두 과학자에 (4보)
---------------- 4 of total 4508 (ROWID 60874) --------------------
TT (45): 과학자들, 2025년까지 노벨상후보 2명 배출 전망
---------------- 5 of total 4508 (ROWID 19344) --------------------
TT (66): <긴급>올 노벨 물리학상 게라르두스 후프트와 마르티누스 벨트만 수상
---------------- 6 of total 4508 (ROWID 19366) --------------------
TT (31): <올해 노벨물리학상 수상자 업적>
---------------- 7 of total 4508 (ROWID 19547) --------------------
TT (33): <아쉬움 더해주는 올 노벨물리학상>
---------------- 8 of total 4508 (ROWID 20003) --------------------
TT (46): [사이언스] 노벨물리학상에 토프트/벨트만씨 선정
```

검색 결과 '노벨 물리학상 네덜란드 두 과학자에'와 유사한 문서들이 함께 검색된다.

유사문서 검색 모듈(similar2)

독크루저가 기본적으로 제공하는 검색 방법의 similar 이외에 유사문서 검색 모듈(mod-sim, kql240.mod)을 이용한 검색이 있다. 유사문서 검색 모듈의 검색 메소드는 'similar2'를 이용한다. 유사문서 검색(similar2)은 대상 문서의 키워드를 추출하여 통계적 출현 빈도수를 고려하여 검색하는 기본 개념은 similar 검색 방법과 비슷하지만 두 문서간의 유사도를 직접 계산하여 미리 설정된 유사도 기준값 이상의 문서들을 유사도 순으로 제공하고, 여러 필드에 걸쳐 유사도 산출을 할 수 있으며, 세부 기준을 직접 고객사 환경에 맞게 설정하여 사용할 수 있도록 한다.

자세한 설명은 '유사문서 검색 (SIMILAR DOCUMENT SEARCH)'을 참고한다.

4.1.5. 복제문서 검색 (REPLICATE)

복제문서란 다른 문서의 내용을 짜깁기하여 생성된 문서를 말한다. 원본 문서의 내용에 약간의 변형을 가하여 의도를 가진 기만행위가 들어가 있다. 복제문서 검색 모듈의 검색 메소드는 'replicate'를 이용한다.

- 구문

```
SELECT {*|field_name, …} FROM table_name WHERE field_name =
{ROWID | 'body_text'} REPLICATE
```

자세한 설명은 '복제문서 검색 (REPLICATED DOCUMENT SEARCH)'을 참고한다.

4.1.6. 근접어 검색 (PROXKEYMATCH)

사전과 같은 데이터를 검색하는 경우 질의어에 해당하는 키와 근접해 있는 키들을 더 얻기를 원하는 경우가 있다. 이 경우 근접어 검색 기능을 사용하여 주어진 키와 근접한 레코드들을 검색할 수 있다.

- 구문

```
SELECT {*|field_name, …} FROM table_name WHERE {field_name |
index_name} = 'query_word' PROXKEYMATCH
```

아래의 예는 'fd' 필드에서 질의어 'returq3'의 근접어를 검색한 결과이다.

```
kql/tv> select * from tbl where fd = 'returq3' PROXKEYMATCH;
---------------- 1 of total 10 (ROWID 12) --------------------
fd (6): returm
---------------- 2 of total 10 (ROWID 13) --------------------
fd (6): return
---------------- 3 of total 10 (ROWID 14) --------------------
fd (6): returo
---------------- 4 of total 10 (ROWID 15) --------------------
fd (6): returp
---------------- 5 of total 10 (ROWID 16) --------------------
fd (6): returq
---------------- 6 of total 10 (ROWID 17) --------------------
fd (6): returr
---------------- 7 of total 10 (ROWID 18) --------------------
fd (6): returs
```

```
--------------- 8 of total 10 (ROWID 19) --------------------
fd (6): returt
--------------- 9 of total 10 (ROWID 20) --------------------
fd (6): returu
--------------- 10 of total 10 (ROWID 21) --------------------
fd (6): returv
Total 10 records.
OK

kql/tv>
```

데이터 상으로 'returq3'이 정확히 일치되는 엔트리는 존재하지 않지만 그 부근의 키로 검색한 레코드들을 찾을 수 있다.

4.1.7. 동의어 검색 (SYNONYM)

사용자의 질의어를 같은 의미의 다른 단어로도 확장하여 검색하기 원하는 경우 동의어 검색을 사용한다. 동의어 사전에 등록한 단어를 대상으로 하며 구문은 검색 방법 뒤에 SYNONYM을 덧붙여주면 된다. SYNONYM을 사용하여 검색 하는 경우는 kql.rc에 등록한 동의어 사전을 사용한다.

- 구문

```
SELECT {*|field_name, …} FROM table_name WHERE {field_name |
index_name} = 'query_word' SYNONYM
```

동의어 확장을 하려면 동의어 사전이 필요한데, 이 사전을 통한 동의어 확장 방식은 일반 동의어 검색과 데몬 동의어 검색 두가지 경로로 제공된다. 검색쿼리를 구성하는 것은 검색 메소드에 synonym을 붙여 주는 것으로 동일하지만, 몇가지 설정과 사용방법에 차이점이 있다. 여기서는 일반 동의어 검색에 대한 설명만 하고 데몬 동의어 검색에 대한 자세한 설명은 '데몬 동의어 (SYNONYM KEYWORD)'를 참조 한다. 데몬 동의어를 사용하고자 할 경우 검색식에 SYNONYM을 사용하지 않도록 한다.

동의어 사전파일 예는 다음과 같다.

```
달걀, 계란
텔레비전, 텔레비전, TV, television
```

다음 예에서 'natural'로만 검색할 때는 '달걀'에 대한 검색 결과만 나오나 'natural synonym'으로 검색하는 경우 '달걀'이 '계란'으로 동의어 확장되어 모두에 대한 검색 결과가 나오는 것을 볼 수 있다.

```
kql/tv> select ingredient from recipe where ingredient = '달걀'
natural;
---------- 0 of total 1 (ROWID 5) ---------
ingredient (38): 밀가루 1컵, 돼지고기 600그램, 달걀 2개
Total 1 record.
OK

kql/tv> select ingredient from recipe where ingredient = '달걀'
natural synonym;
---------- 0 of total 2 (ROWID 4) ---------
ingredient (32): 간장 1큰술, 계란 3개, 버터 3큰술
---------- 1 of total 2 (ROWID 5) ---------
ingredient (38): 밀가루 1컵, 돼지고기 600그램, 달걀 2개
Total 2 records.
OK
```

일반 동의어 검색의 경우 사전이 변경되었을 때 아래와 같은 방법으로 엔진 재시작 없이 사전을 반영시킬 수 있다.

```
kql> reload user dic for synonym ../dictionary/synonym.txt;
OK
```

위 명령어는 동의어 사전을 컴파일 하고 로딩시키는 명령으로, 자세한 설명은 '사용자 사전 - 동의어 (RELOAD USER DIC FOR SYNONYM)'를 참고한다. 일반 동의어 사전은 텍스트 그대로 로딩 되므로 사전이 클 경우 초기에 사전 로딩 시간이 오래 걸릴 수 있다. 이럴 경우 다음과 같이 텍스트형식의 사전을 컴파일 후 컴파일 된 데이터 파일을 등록할 수 있다.

```
kql> control synonym dictionary operation compile ../dictionary/
synonym.txt to ../data/synonym.dat;
OK

kql> reload user dic for compiled synonym ../data/synonym.dat;
OK
```

이에 대한 자세한 설명은 '사용자 사전 - 컴파일 된 동의어(RELOAD USER DIC FOR COMPILED SYNONYM)'를 참고한다.

4.1.8. 선택 검색 (<, >, LIKE, ANDNOT, !=, in)

키워드에 의한 검색은 언어로 이루어진 텍스트에 적합한 검색 방법이다. 그러나 하나의 문서는 이 밖에도 작성일, URL, HOST 등 많은 메타정보들을 가질 수 있다. 예를 들어 '청와대'가 포함된 문서를 원할 수도 있지만 2000 년도에 작성된 문서를 원할 수도 있고 HOST가 www.cwd.go.kr인 사이트에서 가져온 문서를 원할 수도 있다. 선택검색은 TEXT 타입이 아닌 검색 필드에 대한 검색 용도로 사용한다.

- 구문

```
SELECT {*|field_name} FROM table_name WHERE {field_name|
index_name} {LIKE | NOT LIKE} 'query_word*'
SELECT {*|field_name} FROM table_name WHERE {field_name1|
index_name1}='query_word' {AND | ANDNOT | OR} {field_name2|
index_name}='query_word'
SELECT {*|field_name} FROM table_name WHERE {field_name1|
index_name2} {> | >= | < | <= | !=} 'query_word'
SELECT {*|field_name} FROM table_name WHERE {field_name1|
index_name2} {IN} {{'query_word1', 'query_word2', ...}}
```

구간 검색(<, >)

데이터 타입이 날짜(DATE)나 정수(INT16, INT32)일 때 유용하다. 다음 예제는 날짜가 2001 년 7 월 3 일 이후이고, 2002 년 10 월 3 일 이전인 문서만 찾은 결과를 보여준다. 사용 방법은 날짜나 정수 필드에 대해 '<', '>', '<=', '>=' 연산자와 숫자값으로 표현하면 된다.

```
kql/tv2> select UD,TT from WebData where UD<20021003 AND UD >
20010703;
--------------- 1 of total 6 (ROWID 160331) ---------------
UD (8): 20020305
TT (16): kwangbul's world
--------------- 2 of total 6 (ROWID 384881) ---------------
UD (8): 20020310
TT (10): 우리반??!!
--------------- 3 of total 6 (ROWID 5366495) --------------
UD (8): 20020213
TT (42): KIDS3 : KITSAT-3 Image Distribution System
--------------- 4 of total 6 (ROWID 5686374) --------------
UD (8): 20020223
```

```
TT (8): Untitled
--------------- 5 of total 6 (ROWID 7655589) --------------
UD (8): 20020228
TT (8): 학교연혁
--------------- 6 of total 6 (ROWID 13064723) -------------
UD (8): 20020213
TT (42): KIDS3 : KITSAT-3 Image Distribution System
Total 6 records.
OK

kql/tv2>
```

LIKE 검색

데이터 타입이 스트링(STRING)인 경우 유용하다. 호스트이름이나 URL 을 스트링으로 저장하면 [호스트 내 검색]이나 [개인 홈페이지 검색]을 할 수 있다. 절단연산자(*)를 사용할 수 있다. 단 'NOT LIKE' 연산은 다른 긍정 질의어와 함께 사용해야 한다.

다음은 LIKE 검색의 예이다.

```
kql/tv2> select UR from WebData where UR like 'www.kbs.*';
---------------- 1 of total 6587 (ROWID 20395) --------------------
UR (37): www.kbs.co.kr/1tv/dol/main-center.htm
---------------- 2 of total 6587 (ROWID 20397) --------------------
UR (38): www.kbs.co.kr/1tv/nreport/main-top.htm
---------------- 3 of total 6587 (ROWID 20403) --------------------
UR (51): www.kbs.co.kr/drama/humanhouse/Vod/Scenario/019.htm
---------------- 4 of total 6587 (ROWID 20408) --------------------
```

NOT LIKE 검색

데이터 타입이 스트링(STRING)인 경우 유용하다. 호스트이름이나 URL을 스트링으로 저장하면 [호스트 내 검색]이나 [개인 홈페이지 검색]을 할 수 있다. 절단연산자(*)를 사용할 수 있다.

다음은 NOT LIKE 검색의 예이다.

```
kql/myvol>> select tt from tbl;
---------- 0 of total 10 (ROWID 0) ---------
tt (1): 1
---------- 1 of total 10 (ROWID 1) ---------
tt (1): 1
```

```
---------- 2 of total 10 (ROWID 2) ---------
tt (1): 1
---------- 3 of total 10 (ROWID 3) ---------
tt (1): 1
---------- 4 of total 10 (ROWID 4) ---------
tt (1): 1
---------- 5 of total 10 (ROWID 5) ---------
tt (1): 1
---------- 6 of total 10 (ROWID 9) ---------
tt (1): 2
Total 6 records.
OK

kql/myvol>> delete from tbl where tt not like '2';
Total 5 records.
OK

kql/myvol>> select tt from tbl;
---------- 0 of total 1 (ROWID 9) ---------
tt (1): 2
Total 1 record.
OK
```

선택 검색의 복합 응용

선택 검색은 일반 검색과 함께 사용하면 검색 문서를 조건에 맞게 필터링하는 효과가 있다.

다음은 'korealink.co.kr' 사이트에서 제목란에 '경제'가 포함된 문서들 중 2002 년 2 월 1 일 이후의 문서들을 찾으라는 질의어를 사용한 예이다. 절단 연산자는 가급적 우절단(str*)을 사용하도록 한다. 좌절단(*str)을 사용하면 검색 속도가 현저히 저하될 수 있다.

```
TT='경제' AND UR like 'korealink.co.kr*' AND UD > 20020201
```

```
kql/tv2> select TT, UD, UR from WebData where TT='경제' AND UR like
'korealink.co.kr*' AND UD > 20020201
---------------- 1 of total 3 (ROWID 7714) --------------------
TT (42): 서울경제 : 산업용LNG 특소세 놓고 업계 대립
UD (8): 20020205
UR (60): korealink.co.kr/sed_industry/200202/e20020205170947345l0
.htm
```

```
---------------- 2 of total 3 (ROWID 32686) --------------------
TT (45): 서울경제 : 울산~日키타큐슈 빠르면 4월 첫 운항
UD (8): 20020203
UR (56): korealink.co.kr/sed_news/200202/e200202031704483G510.htm
---------------- 3 of total 3 (ROWID 83410) --------------------
TT (35): 서울경제 : "금융민영화 조속히 매듭"
UD (8): 20020205
UR (56): korealink.co.kr/sed_news/200202/e200202051911133G510.htm
Total 3 records.
OK

kql/tv2>
```

텍스트 라인 검색

텍스트 문서 중 한 라인이 특별한 의미를 가질 수 있는 경우가 있다. 예를 들어 mp3 파일의 경우 파일 이름은 대개 "박지윤의 성인식.mp3"이고 이 안에 개행 문자가 들어가는 경우는 거의 없다. 텍스트 라인 검색 기능을 사용하면 이러한 문서들만을 골라서 찾아줄 수 있다. 이외에도 중요한 정보들이 라인별로 나열된 문서를 검색할 때 유용하다.

다음은 필드 속성이 'text'일 때와 'text line'일 때 검색 결과의 예이다. fd1과 fd3는 'text line'이고 fd2는 'text'이다. 입력된 값은 fd1과 fd2가 "박지윤의 \n 성인식"이고 fd3는 "박지윤의 성인식"이다.

```
kql/tv3> create table t4 ( fd1 text line, fd2 text, fd3 text line );
kql/tv3> create index idx on t4(fd1, fd2, fd3);
.................................................
kql/tv3> select * from t4;
---------------- 1 of total 1 (ROWID 0) --------------------
fd1 (15): 박지윤의
성인식
fd2 (15): 박지윤의
성인식
fd3 (15): 박지윤의 성인식
Total 1 record.
OK
```

이제 각 필드에 대해 동일하게 '박지윤의 성인식' allword 검색을 해 본다.

```
kql/tv3> select * from t4 where fd1='박지윤의 성인식' allword;
Total 0 record.
```

```
OK

kql/tv3> select * from t4 where fd2='박지윤의 성인식' allword;
---------------- 1 of total 1 (ROWID 0) --------------------
fd1 (15): 박지윤의
성인식
fd2 (15): 박지윤의
성인식
fd3 (15): 박지윤의 성인식
Total 1 record.
OK

kql/tv3> select * from t4 where fd3='박지윤의 성인식' allword;
---------------- 1 of total 1 (ROWID 0) --------------------
fd1 (15): 박지윤의
성인식
fd2 (15): 박지윤의
성인식
fd3 (15): 박지윤의 성인식
Total 1 record.
OK

kql/tv3>
```

fd1에 대한 검색에서 검색이 안된 이유는 '박지윤'과 '성인식' 사이에 개행 문자가 있기 때문이다. fd2에서 검색된 이유는 'text' 속성이기 때문에 조건에 맞는다. fd3에서 검색된 이유는 '박지윤'과 '성인식' 사이에 개행 문자가 없이 같은 라인에 존재하기 때문이다.

엔트리 검색

절단 검색 시에는 절단조건에 맞는 키워드들을 포함하는 모든 문서들이 합해져서 검색된다. 그러나 사전 검색의 경우 주어진 단어와 근접한 다른 엔트리에 대한 정보를 개별적으로 얻어야 하는 경우가 있다. 엔트리 검색은 절단 검색 시 각 엔트리들을 합하지 않고 개별적으로 검색할 수 있도록 한다.

엔트리 검색은 다음과 같은 질의 형태의 검색을 일컫는다.

```
IX > 'aaa' AND IX < 'bbb' AND IX like 'aa*'
```

엔트리 검색 방법을 이용한 사전 검색의 예는 다음과 같다.

```
fd1 <='aristocratically'AND fd1 >='aristides'AND fd1 like'arist*'
```

다음 결과를 보면 조건에 맞는 각 엔트리들이 합해지지 않고 개별적으로 검색되었다.

```
Aristides [2] 46, 132
aristo [3] 46, 85, 98
aristo- [1] 5
aristocracy [6] 1, 46, 47, 51, 64, 117
aristocrat [2] 3, 32
aristocratic [1] 3
aristocratically [1] 3
```

제외어 검색 (ANDNOT)

검색 시 특정 키워드들을 뺀 검색을 하고 싶은 경우가 있다. 다음의 예는 '한석규'는 나오지만 '심은하'가 나오는 문서를 제외시키는 방법을 보여준다. 제외어 검색은 다른 긍정 질의어와 함께 사용해야 한다. 'AND NOT'은 동작하나 'NOT' 단독으로는 동작하지 않음에 유의한다.

```
kql/myvol> select * from tbl;
---------------- 1 of total 2 (ROWID 0) --------------------
fd (15): 한석규와 심은하
---------------- 2 of total 2 (ROWID 1) --------------------
fd (20): 한석규가 출연한 영화
Total 2 records.
OK

kql/myvol> select * from tbl where fd='한석규';
---------------- 1 of total 2 (ROWID 0) --------------------
fd (15): 한석규와 심은하
---------------- 2 of total 2 (ROWID 1) --------------------
fd (20): 한석규가 출연한 영화
Total 2 records.
OK

kql/myvol> select * from tbl where fd='한석규' ANDNOT fd='심은하';
---------------- 1 of total 1 (ROWID 1) --------------------
fd (20): 한석규가 출연한 영화
Total 1 record.
OK
```

부정 연산자 (!=)

ANDNOT과 동일한 효과를 줄 수 있는 연산자이다.

```
fd1 = '한석규' ANDNOT fd1 = '>심은하'
```

구문은

```
fd1 = '한석규' AND fd1 != '심은하'
```

구문과 동치이다.

논리합 연산자(IN)

OR 연산자 대신에 사용할 수 있다.

```
pk='1' or pk='2' or pk='3'
```

구문은

```
pk in {'1', '2', '3'}
```

구문과 동치이다.

다양한 텍스트 검색 방법과 STRING, 숫자형 검색 방법을 조합하여 사용할 경우 검색 결과의 조합에 따라 검색 결과가 달라진다. 검색 조건이 조합되었을 경우 괄호를 이용하여 검색 순서를 명시적으로 결정해주는 것도 검색 결과를 예측하는 데 좋은 방법이 될 것이다.

4.2. 정렬

4.2.1. 검색 결과 정렬 (ORDER BY)

검색 결과 정렬 (ORDER BY)

검색에 따라서 특정 필드별로 검색 결과를 정렬할 필요가 있는 경우 ORDER BY 구문을 사용한다. 정렬 조건에 포함되는 필드는 인덱싱되어 있어야 한다.

- 구문

```
ORDER BY { field_name [ASC|DESC], ...}
```

- 예제

```
kql/tv3> select * from t5 where fd1='한석규' order by fd1;
---------------- 1 of total 2 (ROWID 1) --------------------
fd1 (20): 한석규가 출연한 영화
---------------- 2 of total 2 (ROWID 0) --------------------
fd1 (15): 한석규와 심은하
Total 2 records.
OK

kql/tv3> select * from t5 where fd1='한석규' order by fd1 desc;
---------------- 1 of total 2 (ROWID 0) --------------------
fd1 (15): 한석규와 심은하
---------------- 2 of total 2 (ROWID 1) --------------------
fd1 (20): 한석규가 출연한 영화
Total 2 records.
OK

kql/tv3>
```

위 결과를 보면 'order by'에 의해 정렬 순서가 바뀌었음을 알 수 있다.

정렬 가능한 필드 타입에 대한 제약이 있다. 다음 필드 타입들에 대한 정렬은 지원하지 않는다.

만일 해당 타입의 데이터 값으로 꼭 정렬을 해야 한다면 정렬 가능한 다른 필드 타입으로 변경후 정렬하는 방법(ex.텍스트->스트링)이 있다.

타입 구문	설명
TEXT	텍스트 필드
STRING LIST	스트링 리스트 필드(나머지 스트링 타입(STRING, STRING VERBATIM, STRING PARA 등)은 가능)
BLOB	바이너리 데이터 필드
IMAGE	이미지 데이터 필드
AUDIO	오디오 데이터 필드
VIDEO	비디오 데이터 필드

레코드 고유번호순 정렬 ($ROWID)

검색한 결과를 레코드 고유번호($ROWID) 순으로 정렬한다. 먼저 삽입된 레코드가 가장 먼저 나오는 효과가 있다.

- 구문

```
ORDER BY $ROWID [ASC|DESC]
```

적합도순 정렬 ($RELEVANCE)

검색 결과에 대해 별도의 정렬 조건이 명시되지 않으면 적합도 순으로 정렬한다. 적합도는 사용자가 입력한 질의와 문서가 얼마나 관련이 높은가를 나타낸다. ORDER BY를 명시하여 다른 정렬조건을 주는 경우, 적합도 정렬을 함께 사용하고자 할 때 $RELEVANCE 정렬을 함께 사용할 수 있다.

예를 들어, 날짜 필드의 역순으로 정렬하였을 경우 문서는 적합도와 관계없이 최근 날짜 순으로 정렬된다. 내부적으로 날짜가 같은 레코드들은 ROWID가 낮은 순으로 정렬된다. 만약 같은 날짜의 레코드들이 여러 개 있을 때 이들을 다시 적합도가 높은 것 순으로 나열하고 싶은 경우에 $RELEVANCE 정렬 옵션을 같이 사용한다.

- 구문

```
ORDER BY $RELEVANCE [ASC|DESC]
```

- 사용 예

```
SELECT * FROM table WHERE field = 'query_word'
ORDER BY date DESC, $RELEVANCE DESC
```

- 예제

다음은 적합도에 의해 정렬된 결과이다. $RELEVANCE로만 정렬했으므로 ORDER BY를 명시하지 않는 경우와 동일한 결과이다.

```
kql/tv> select * from tb1 where all_idx='미국 중국 한국 독일'
natural order by
$RELEVANCE DESC;
---------- 0 of total 5 (ROWID 4) ---------
f1 (61): 미국 중국 한국 독일 미국 중국 한국 부시 부시 부시 프랑스 독일
---------- 1 of total 5 (ROWID 3) ---------
f1 (61): 미국 부시 중국 한국 독일 미국 부시 부시 중국 한국 프랑스 독일
```

```
---------- 2 of total 5 (ROWID 0) ---------
f1 (14): 한국 미국 중국
---------- 3 of total 5 (ROWID 1) ---------
f1 (14): 미국 한국 중국
---------- 4 of total 5 (ROWID 2) ---------
f1 (19): 한국 한국 한국 한국
Total 5 records.
```

매치 필드별 정렬($MATCHFIELD)

검색된 필드별 정렬이다.

예를 들어 $MATCHFIELD(TT, BD) DESC는 검색된 문서들 중 TT에서 질의어를 포함한 문서를 먼저 나열하고 그 뒤에 BD에서 검색된 문서들을 나열하라는 의미이다. 검색 인덱스에 포함되지 않은 필드가 괄호 안에 포함된 경우 그 필드는 정렬에 사용되지 않는다.

다른 정렬방식과 함께 있을 때 의미를 가지게 되는 데 $MATCHFIELD(TT, BD) DESC만 지정했을 때는 ORDER BY $MATCHFIELD(TT,BD) DESC, $ROWID ASC와 같은 의미이다. TT, BD에서 모두 검색된 문서들을 $ROWID 순으로 모두 나열한 후 그 다음으로 TT 필드로 검색된 문서를, 마지막으로 BD 필드로 검색된 문서를 나열한다.

ORDER BY $MATCHFIELD(TT, BD) DESC, $RELEVANCE DESC인 경우는 검색된 레코드 중에서 TT, BD에서 검색된 문서들을 적합도 순으로 재정렬하여 나열한 이후 TT 필드, BD 필드로 검색된 문서들을 적합도 순으로 재정렬하여 나열하게 된다.

- 구문

```
ORDER BY $MATCHFIELD( field_name, ... ) [ASC|DESC]
```

- 예제

```
kql/tv> select * from GISA;
---------- 0 of total 4 (ROWID 0) ---------
TT (11): 마라톤 영웅
BD (14): 한강 춘천 경기
---------- 1 of total 4 (ROWID 1) ---------
TT (11): 마라톤 영웅
BD (26): 마라톤 영웅 한강 춘천 경기
---------- 2 of total 4 (ROWID 2) ---------
TT (13): 한강에서 열림
```

```
BD (39): 한강에서 열린 마라톤 경기에서 영웅 탄생
---------- 3 of total 4 (ROWID 3) ---------
TT (4): 열풍
BD (23): 마라톤 영웅 열풍이 불고 있다
Total 4 records.
OK

kql/tv> select * from GISA where TTBD_IDX='마라톤 영웅' natural;
---------- 0 of total 4 (ROWID 1) ---------
TT (11): 마라톤 영웅
BD (26): 마라톤 영웅 한강 춘천 경기
---------- 1 of total 4 (ROWID 0) ---------
TT (11): 마라톤 영웅
BD (14): 한강 춘천 경기
---------- 2 of total 4 (ROWID 3) ---------
TT (4): 열풍
BD (28): 마라톤 영웅 열풍이 불고 있다
---------- 3 of total 4 (ROWID 2) ---------
TT (13): 한강에서 열림
BD (39): 한강에서 열린 마라톤 경기에서 영웅 탄생
Total 4 records.
OK
-> 정렬조건이 없으므로 기본 정렬방식인 적합도 역순으로 정렬되었다.

kql/tv> select * from GISA where TTBD_IDX='마라톤 영웅'
natural order by $RELEVANCE
DESC;
---------- 0 of total 4 (ROWID 1) ---------
TT (11): 마라톤 영웅
BD (26): 마라톤 영웅 한강 춘천 경기
---------- 1 of total 4 (ROWID 0) ---------
TT (11): 마라톤 영웅
BD (14): 한강 춘천 경기
---------- 2 of total 4 (ROWID 3) ---------
TT (4): 열풍
BD (28): 마라톤 영웅 열풍이 불고 있다
---------- 3 of total 4 (ROWID 2) ---------
TT (13): 한강에서 열림
BD (39): 한강에서 열린 마라톤 경기에서 영웅 탄생
Total 4 records.
```

```
OK
-> 적합도 역순으로 정렬되었다.

kql/tv> select * from GISA where TTBD_IDX='마라톤 영웅'
natural order by $ROWID
DESC;
---------------- 1 of total 4 (ROWID 3) --------------------
TT (4): 열풍
BD (23): 마라톤 영웅 열풍이 불고 있다
---------------- 2 of total 4 (ROWID 2) --------------------
TT (13): 한강에서 열림
BD (39): 한강에서 열린 마라톤 경기에서 영웅 탄생
---------------- 3 of total 4 (ROWID 1) --------------------
TT (11): 마라톤 영웅
BD (26): 마라톤 영웅 한강 춘천 경기
---------------- 4 of total 4 (ROWID 0) --------------------
TT (11): 마라톤 영웅
BD (14): 한강 춘천 경기
Total 4 records.
OK
-> ROWID 역순으로 정렬되었다.

kql/tv> select * from GISA where TTBD_IDX='마라톤 영웅'
natural order by
$MATCHFIELD(TT, BD) DESC;
---------- 0 of total 4 (ROWID 1) ---------
TT (11): 마라톤 영웅
BD (26): 마라톤 영웅 한강 춘천 경기
---------- 1 of total 4 (ROWID 0) ---------
TT (11): 마라톤 영웅
BD (14): 한강 춘천 경기
---------- 2 of total 4 (ROWID 2) ---------
TT (13): 한강에서 열림
BD (39): 한강에서 열린 마라톤 경기에서 영웅 탄생
---------- 3 of total 4 (ROWID 3) ---------
TT (4): 열풍
BD (23): 마라톤 영웅 열풍이 불고 있다
Total 4 records.
OK
-> TT, BD에 의해 검색된 레코드가 제일 위로 올라왔다. 그 다음으로는 TT에
```

```
의해 검색된 레코드 ROWID 0이 위치한다.
ROWID 2와 3은 동일하게 BD에서 검색이 되었기때문에 MATCHFIELD에 의한
조건은 동일하고 MATCHFIELD외의 다른 정렬조건이 없으므로 ROWID의
오름차순으로 정렬되었다.

kql/tv> select * from GISA where TTBD_IDX='마라톤 영웅'
natural order by
$MATCHFIELD(TT, BD) DESC, $ROWID DESC;
---------------- 1 of total 4 (ROWID 1) --------------------
TT (11): 마라톤 영웅
BD (26): 마라톤 영웅 한강 춘천 경기
---------------- 2 of total 4 (ROWID 0) --------------------
TT (11): 마라톤 영웅
BD (14): 한강 춘천 경기
---------------- 3 of total 4 (ROWID 3) --------------------
TT (4): 열풍
BD (23): 마라톤 영웅 열풍이 불고 있다
---------------- 4 of total 4 (ROWID 2) --------------------
TT (13): 한강에서 열림
BD (39): 한강에서 열린 마라톤 경기에서 영웅 탄생
Total 4 records.
OK
-> TT, BD에 의해 검색된 레코드가 제일 위로 올라왔다. 그 다음으로는
TT에 의해 검색된 레코드 ROWID 0이 위치한다.
ROWID 2와 3은 동일하게 BD에서 검색이 되었기 때문에 MATCHFIELD에
의한 조건은 동일한데 그 다음 정렬조건이 ROWID DESC이므로
ROWID 3, 2의 순으로 정렬되었다.

kql/tv> select * from GISA where TTBD_IDX='마라톤 영웅'
natural order by
$MATCHFIELD(TT,BD) DESC, $RELEVANCE DESC;
---------------- 1 of total 4 (ROWID 1) --------------------
TT (11): 마라톤 영웅
BD (26): 마라톤 영웅 한강 춘천 경기
---------------- 2 of total 4 (ROWID 0) --------------------
TT (11): 마라톤 영웅
BD (14): 한강 춘천 경기
---------- 2 of total 4 (ROWID 3) ---------
TT (4): 열풍
BD (28): 마라톤 영웅 열풍이 불고 있다
```

```
---------- 3 of total 4 (ROWID 2) ---------
TT (13): 한강에서 열림
BD (39): 한강에서 열린 마라톤 경기에서 영웅 탄생
Total 4 records.
OK
-> TT, BD에 의해 검색된 레코드가 제일 위로 올라왔다. 그 다음으로는
TT에 의해 검색된 레코드 ROWID 0이 위치한다.
ROWID 2와 3은 동일하게 BD에서 검색이 되었기 때문에 MATCHFIELD에
의한 조건은 동일한데 적합도가 높은 ROWID 3이 위로 올라왔다.
```

검색된 필드별 정렬 사용 시 유의할 점이 한가지 있는데, 필드에서 검색되었다는 기준이 검색 메서드를 따른다는 점이다. 예를 들어 idx='A B' anyword order by $MATCHFIELD(fd1, fd2) DESC로 검색을 할 때 fd1에서 키워드 'A' 또는 'B' 둘중의 하나만 검색되어도 검색된 필드별 정렬의 입장에서 '검색되었다'에 해당된다는 점이다.

예를 들면 다음과 같다.

```
kql/tv> select * from tb3 where idx='한국 미국' anyword order by
$MATCHFIELD(fd1,fd2) desc;
---------- 0 of total 2 (ROWID 1) ---------
fd1 (4): 한국
fd2 (4): 미국
---------- 1 of total 2 (ROWID 2) ---------
fd1 (9): 한국 미국
fd2 (4): 미국
Total 2 records.
OK

kql/tv> select * from tb3 where idx='한국 미국' anyword order by
$MATCHFIELD(fd1, fd2) desc, $RELEVANCE DESC;
---------- 0 of total 2 (ROWID 2) ---------
fd1 (9): 한국 미국
fd2 (4): 미국
---------- 1 of total 2 (ROWID 1) ---------
fd1 (4): 한국
fd2 (4): 미국
Total 2 records.
OK
```

위의 경우 검색어는 '한국 미국'이지만 검색 메서드가 ANYWORD였으므로 두 개의 문서는 검색된 필드별 정렬의 입장에서는 동등한 문서이다. 그러므로

$MATCHFIELD(fd1, fd2) DESC만 명시한 경우에는 ROWID의 오름차순으로 정렬된다. 이 경우 '한국'과 '미국'이 모두 들어간 문서를 더 상위로 오게 하려면 $MATCHFIELD와 함께 다른 정렬방식을 명시해야 한다. 위의 경우는 $RELEVANCE DESC를 함께 사용하여 검색된 필드가 동등한 경우에는 적합도로 정렬하도록 하였다. ORDER BY로 정렬 구문을 주는 경우 몇 가지의 정렬조건을 함께 사용할 수 있는 데 이러한 경우 앞쪽의 정렬조건이 우선 적용된다.

ALIASING 구문

어떤 필드에 대해 여러 가지 검색 방식을 적용하고 난 후 각 검색 방식에 따른 정렬이 필요한데, 이것을 되도록이면 하나의 질의문 형태로 처리하고 싶어하는 경우가 있다.

```
tt='사과 나무' alladjacent or tt='사과 나무' allword or tt='사과 나무'
anyword
```

예를들면, 위와 같은 질의의 결과로 나오는 문서들을 적합도 정렬을 시키면 대략 alladjacent > allword > someword 순으로 레코드가 나온다. 그런데 모두 동일한 필드에 대해 동일한 키워드로 검색한 것이기 때문에, 여기에 추가로 'order by $ROWID desc'와 같이 정렬하면 앞서의 적합도 정렬이 무의미해진다.

"aliasing" 구문은 어떤 필드에 대한 'symbolic link' 개념의 필드들을 지정할 수 있게 하려는 것이다. 물리적인 필드를 생성하는 것이 아니라 질의시에만 임시로 만들어지는 필드에 대한 레퍼런스라고 보면 된다.

"aliasing" 구문을 이용하여 동일한 필드를 여러 개의 다른 필드 이름으로 구분해서 검색한 후 $MATCHFIELD 정렬과 결합하여 위의 '사과나무' 질의에서 얻고자 하는 소기의 목적을 달성할 수 있다.

- 구문

```
... WHERE ALIASING fd AS fd1, fd AS fd2 ...
```

- 예제

```
kql/tv> retrieve * from tab458 where
       -> aliasing tt as tt1, tt as tt2, ss as sx
       -> tt='사과 나무' alladjacent or
       -> tt1='사과 나무' allword or
       -> tt2='사과 나무' anyword or
       -> sx='2'
       -> order by $MATCHFIELD(tt,tt1,tt2,sx) desc, $ROWID asc;
```

```
---------- 0 of total 6 (ROWID 5, REL 9820) ---------
sn (0):
ss (3): 2 4
tt (32): 그가 심은 <<사과나무>>는 10년 후에야
bd (0):
---------- 1 of total 6 (ROWID 2, REL 9826) ---------
sn (0):
ss (3): 1 1
tt (30): <<사과나무>>학교에 아이를 보내면서
bd (0):
---------- 2 of total 6 (ROWID 1, REL 9817) ---------
sn (0):
ss (3): 2 1
tt (38): 밤<<나무>>새순 절취 작업과 <<사과>>봉지 씌우기
bd (0):
---------- 3 of total 6 (ROWID 0, REL 9819) ---------
sn (0):
ss (3): 1 3
tt (18): <<사과>>와 오렌지 <<나무>>
bd (0):
---------- 4 of total 6 (ROWID 3, REL 9813) ---------
sn (0):
ss (3): 2 2
tt (10): <<사과>>가 쿵!
bd (0):
---------- 5 of total 6 (ROWID 4, REL 9808) ---------
sn (0):
ss (3): 1 3
tt (21): 나의 라임 오렌지 <<나무>>
bd (0):
Total 6 records.
OK
```

카테고리 랭킹 정렬($CATEGORYFIELD)

검색 키워드가 속한 카테고리에 들어 있는 문서를 상위로 정렬한다. 키워드 별 상위 랭킹 될 카테고리와 제외할 카테고리명을 정의할 수 있고 동의어 확장이 가능하다.

- 구문

```
ORDER BY $CATEGORYFIELD(field_name(domain), keyword)
```

카테고리 랭킹 정렬을 사용하기 위해서는 카테고리 랭킹 사전을 정의해야 하는데 사전 형식은 다음과 같다.

```
키워드 : 카테고리명, …, : 제외 할 카테고리명, …
```

카테고리 랭킹 정렬에 대한 자세한 내용은 '카테고리 랭킹 (CATEGORY RANKING)'을 참고한다.

필드 크기순 정렬($SIZE)

특정 필드의 필드값의 크기 순서로 정렬한다.

- 구문

```
BY $SIZE(field_name) [ASC|DESC]
```

- 예제

```
================================================================
order by $relevance 적용 예
================================================================
kql/myvol> retrieve fd from tbl where fd='삼성전자' allword
order by $relevance desc pagelength=20;
---------- 0 of total 15 (ROWID 0, REL 10000) ---------
fd (27): LG와삼성--전자업계의 라이벌
---------- 1 of total 15 (ROWID 1, REL 10000) ---------
fd (16): 현대전자삼성전기
---------- 2 of total 15 (ROWID 2, REL 10000) ---------
fd (16): (주)삼성전기전자
---------- 3 of total 15 (ROWID 3, REL 10000) ---------
fd (10): [삼성전자]
---------- 4 of total 15 (ROWID 4, REL 10000) ---------
fd (38): 수원시_영통구_매탄동_삼성전자_수원공장
---------- 5 of total 15 (ROWID 5, REL 10000) ---------
fd (18): 삼성♡♡♡♡♡전자
---------- 6 of total 15 (ROWID 6, REL 10000) ---------
fd (9): 삼성전자.
---------- 7 of total 15 (ROWID 7, REL 10000) ---------
fd (32): 고객제일주의_삼성전자_서초대리점
---------- 8 of total 15 (ROWID 8, REL 10000) ---------
fd (12): <삼성><전자>
---------- 9 of total 15 (ROWID 9, REL 10000) ---------
```

```
fd (32): 기쁨주는_삼성전자_남부서비스센터
---------- 10 of total 15 (ROWID 10, REL 10000) ---------
fd (8): 삼성전자
---------- 11 of total 15 (ROWID 11, REL 10000) ---------
fd (26): 삼성-전자업계의 선두주자!!
---------- 12 of total 15 (ROWID 12, REL 10000) ---------
fd (12): ~~삼성전자~~
---------- 13 of total 15 (ROWID 13, REL 10000) ---------
fd (17): e삼성--IT전자계열
---------- 14 of total 15 (ROWID 14, REL 10000) ---------
fd (11): [삼성/전자]
Total 15 records.
OK
================================================================
order by $relevance, $size(fd) 적용 예
================================================================
kql/myvol> retrieve fd from tbl where fd='삼성전자' allword
order by $relevance desc,$size(fd) pagelength=20;
---------- 0 of total 15 (ROWID 10, REL 10000) ---------
fd (8): 삼성전자
---------- 1 of total 15 (ROWID 6, REL 10000) ---------
fd (9): 삼성전자.
---------- 2 of total 15 (ROWID 3, REL 10000) ---------
fd (10): [삼성전자]
---------- 3 of total 15 (ROWID 14, REL 10000) ---------
fd (11): [삼성/전자]
---------- 4 of total 15 (ROWID 8, REL 10000) ---------
fd (12): <삼성><전자>
---------- 5 of total 15 (ROWID 12, REL 10000) ---------
fd (12): ~~삼성전자~~
---------- 6 of total 15 (ROWID 1, REL 10000) ---------
fd (16): 현대전자삼성전기
---------- 7 of total 15 (ROWID 2, REL 10000) ---------
fd (16): (주)삼성전기전자
---------- 8 of total 15 (ROWID 13, REL 10000) ---------
fd (17): e삼성--IT전자계열
---------- 9 of total 15 (ROWID 5, REL 10000) ---------
fd (18): 삼성♡♡♡♡♡전자
---------- 10 of total 15 (ROWID 11, REL 10000) ---------
fd (26): 삼성-전자업계의 선두주자!!
```

```
---------- 11 of total 15 (ROWID 0, REL 10000) ---------
fd (27): LG와삼성--전자업계의 라이벌
---------- 12 of total 15 (ROWID 7, REL 10000) ---------
fd (32): 고객제일주의_삼성전자_서초대리점
---------- 13 of total 15 (ROWID 9, REL 10000) ---------
fd (32): 기쁨주는_삼성전자_남부서비스센터
---------- 14 of total 15 (ROWID 4, REL 10000) ---------
fd (38): 수원시_영통구_매탄동_삼성전자_수원공장
Total 15 records.
```

사용자 정의 순서 정렬

특정 필드에서 사용자가 지정한 키 값 순서대로 정렬한다. 순서를 지정하지 않은 키 값은 순서 무관으로 가정한다. 같은 키 값을 2 번 이상 지정하지 않도록 주의하여야 한다.

- 구문

```
ORDER BY field_name (key0 key1 key2, …)
```

- 예제

```
================================================================
기본 예 (asc 정렬 예)
================================================================
kql/myvol> select fd from tbl where fd like '*' order by fd asc
pagelength=5;
---------- 0 of total 10 (ROWID 0) ---------
fd (1): a
---------- 1 of total 10 (ROWID 1) ---------
fd (1): a
---------- 2 of total 10 (ROWID 5) ---------
fd (1): b
---------- 3 of total 10 (ROWID 3) ---------
fd (1): c
---------- 4 of total 10 (ROWID 2) ---------
fd (1): d
5 of 10 records displayed. Continue(Y/n)?
OK
================================================================
순서 지정 예 1
================================================================
```

```
kql/myvol> select fd from tbl where fd like '*' order by fd
(b c a g) pagelength=5;
---------- 0 of total 10 (ROWID 5) ---------
fd (1): b
---------- 1 of total 10 (ROWID 3) ---------
fd (1): c
---------- 2 of total 10 (ROWID 0) ---------
fd (1): a
---------- 3 of total 10 (ROWID 1) ---------
fd (1): a
---------- 4 of total 10 (ROWID 16) ---------
fd (1): g
5 of 10 records displayed. Continue(Y/n)?
OK
================================================================
순서 지정 예 2
================================================================
kql/myvol> select fd from tbl where fd like '*' order by fd
(i a d f) pagelength=5;
---------- 0 of total 10 (ROWID 10) ---------
fd (1): i
---------- 1 of total 10 (ROWID 0) ---------
fd (1): a
---------- 2 of total 10 (ROWID 1) ---------
fd (1): a
---------- 3 of total 10 (ROWID 2) ---------
fd (1): d
---------- 4 of total 10 (ROWID 15) ---------
fd (1): f
5 of 10 records displayed. Continue(Y/n)?
OK
```

4.2.2. 그룹별 건수 (GROUP BY)

검색 결과를 특정 필드값으로 그룹핑하여 그룹별 건수를 가져온다. 카테고리별 검색건수 등을 표시하는 경우에 해당된다. 이러한 경우 GROUP BY 구문을 사용하나 KQL 상에서는 가능하지 않고 독크루저 API, KQL API 등의 검색 API를 통해 구문을 사용하면 score가 표시되는 위치에 그룹별 건수가 표시된다.

- 구문

```
GROUP BY field_name [ORDER BY COUNT(*) [ASC | DESC]]
```

4.2.3. 중복 필드 제거 (DISTINCT BY)

field_name 필드에서 내용이 동일한 값을 갖는 필드는 적합도가 가장 높은 레코드 하나만 보여준다. 독크루저의 API를 이용하여 적용 할 때는 SubmitQuery()나 Search()의 sorting 파라미터로 전달한다.

- 구문

```
DISTINCT BY field_name
```

자세한 사용법은 'DOCRUZER API Reference Manual'을 참고한다.

4.2.4. 적합도 재조정 (EVALUATE BY)

독크루저의 적합도 산정은 내부적으로 정의 된 여러가지 세부 기준들로 이루어 지기 때문에 외부에서 해석하기 어렵고 API 내부에 가려져 있다. 그런데 점차로 내부 알고리즘인 적합도 산정 방식에 직접 관여하여 랭킹을 서비스에 적합하도록 커스터마이징 하고자 하는 요구가 증가하여 evaluate by 구문이 만들어 졌다. 적합도 값은 고객 서비스별로 커스터마이징 가능하도록 설계 된 데몬 모듈(mod-rev)과 이 모듈에 명령을 내리는 검색 구문으로 재조정할 수 있다. 적합도 재조정 기능은 독크루저 데몬 모듈을 사용하므로 KQL 단독으로는 수행할 수 없다. mod-rev 데몬 모듈은 미리 정의된 기본 검색 구문과 USERDEF 검색 구문을 지원한다.

- 기본 검색 구문

  'domain'이 생략되면 0번 도메인이 지정되고 *param*에는 미리 정의된 기본명령 구문을 입력한다.

```
BY modrev(param) [domain=domain] [, cmd=cmd]
```

- 적압도 재조정 기본 검색 예

```
select * from tbl where fd = '미림 여자 고등학교' natural evaluate
by modrev(GBRS 5000 4000 1000) domain=3 order by dt desc;
```

다음은 데몬 모듈(mod-rev)의 기본 기능들이다.

기본 명령	사용 예	설명
GBAS	evaluate by modrev (GBAS 0x40000000 0x1FFFFFFF)	검색된 레코드들을 절대점수 기준으로 그룹핑하여 적합도 값을 재부여한다. ● 절대점수 < 0x1FFFFFFF : 0 ● 0x1FFFFFFF <= 절대점수 < 0x40000000 : 1 ● 0x40000000 < 절대점수 : 2
GBRS	evaluate by modrev (GBRS 8000 5000 1000)	검색된 레코드들을 상대점수 기준으로 그룹핑하여 적합도 값을 재부여한다. ● 상대점수 < 1000 : 0 ● 1000 <= 상대점수 < 5000 : 1 ● 5000 <= 상대점수 < 8000 : 2 ● 8000 < 상대점수 : 3
GBRK	evaluate by modrev (GBRK 10 20 30 40)	검색된 레코드들을 등수 기준으로 그룹칭하여 적합도 값을 재부여한다. ● 10등 이내 : 4 ● 20등 이내 : 3 ● 30등 이내 : 2 ● 40등 이내 : 1 ● 40등 초과 : 0

기본 명령	사용 예	설명
GBRP	evaluate by modrev (GBRP 1 5 10)	검색된 레코드들을 등수 퍼센티지 기준으로 그룹핑하여 적합도 값을 재부여한다. ● 상위 1% 이내 : 3 ● 상위 5% 이내 : 2 ● 상위 10% 이내 : 1 ● 상위 10% 초과 : 0

- USERDEF 검색 구문

'domain'이 생략되면 0번 도메인이 지정되고 param에 명시된 임의의 스트링 값은 USERDEF 함수에 전달된다. field_name에 명시된 필드는 USERDEF 함수에 전달된다. META 속성의 필드 사용을 권장한다.

```
EVALUATE BY modrev(param) [domain=domain] [, cmd=cmd field=
{field_name field_name …}]
```

- 적합도 재조정 USERDEF 검색 예

```
select * from tbl where fd = '미림 여자 고등학교' natural evaluate
by modrev(A 5000 4000 1000) domain=3, cmd=3, field={dt} order by
$relevance desc;
```

mod-rev의 USERDEF 기능을 사용하기 위해서는 rev-conf.domain.rc에 다음과 같은 설정을 한다.

파라미터	의미	사용 예
userdef_module_name	사용할 USERDEF 라이브러리명	Ex) module_name = multibits.so
userdef_function_name	사용할 USERDEF 함수명	Ex) userdef_function_name = evaluate_relevance_for_multibits

rev-conf.domain.rc에서 설정한 .so 파일의 경로는 docruzerd.rc에서 'daemon_data_location'으로 설정한 경로이다.

- USERDEF 라이브러리 구현

  rev-conf.domain.rc에 설정된 라이브러리명과 함수명으로 .so를 구현한다. 이때 함수로 전달되는 파라미터 정보는 다음과 같다.

파라미터명	설명
char msg[]	이 함수에서 외부로 전달할 메시지 문자열
int para_rowid_count	전달받은 ROWID배열의 크기이다.
Int para_rowid_buf	정수 배열로써 ROWID의 배열이다.
unsigned int para_abs_core_buf[]	정수 배열로써 적합도 절대값의 배열이다.
unsigned int para_rel_score_buf[]	정수 배열로써 적합도 상대값의 배열이다.
int para_field_count	검색 쿼리의 field={<field> <field> ..} 절에 입력된 <field>의 개수이다.
char* para_pp_field_name[]	문자열로써 검색 쿼리의 field={<field> <field> ..} 절에 입력된 <field>들의 배열이다.
char* para_pp_field_type[]	문자열로써 검색 쿼리의 field={<field> <field> ..} 절에 입력된 <field>들의 자료형 배열이다.
void* para_pp_field_value[]	검색 쿼리 field={<field> <field> ..} 절에 입력된 <field>들의 값에 대한 포인터 배열이다.
char para_cond[]	문자열로써 검색 쿼리modrev(<param>)절에 입력된 <param> 값이다.

4.2.5. 제외 검색 (EXCLUDE BY)

제외 검색어란 입력된 사용자 검색어로 검색된 결과에서 특정 키워드가 포함된 문서를 제외시키는 기능이다. 제외 검색어는 소팅절의 'exclude by' 구문으로 작동된다.

- 구문

```
EXCLUDE BY field_name(domain)
```

- 예제

```
select * from tbl where fd = '핸드폰' natural exclude by fd(0)
```

제외 검색어에 대한 자세한 내용은 '제외검색어 (EXCLUSIVE KEYWORD)'를 참고한다.

4.2.6. 최대/최소값 검색 (EXTRACT BY)

결과집합에서 최소값, 최대값, 최소값과 최대값을 얻을 수 있다. min(), max(), minmax() 값은 select에 의해 직접 알 수 있지 않고 'extract by'라는 결과 필터링 기능에 의해 그 조건을 만족하는 레코드 집합만을 알 수 있을 뿐이다.

- 구문

```
EXTRACT BY {MIN | MAX | MINMAX}(field_name)
```

- 예제

```
kql/myvol> select * from tbl where  fd = '1';
---------- 0 of total 6 (ROWID 0) ---------
fd (1): 1
price (2): 10
age (2): 15
---------- 1 of total 6 (ROWID 1) ---------
fd (1): 1
price (2): 20
age (2): 25
---------- 2 of total 6 (ROWID 2) ---------
fd (1): 1
price (2): 10
age (2): 13
---------- 3 of total 6 (ROWID 3) ---------
fd (1): 1
price (1): 3
age (2): 13
---------- 4 of total 6 (ROWID 4) ---------
fd (1): 1
price (2): 25
age (3): 135
```

```
---------- 5 of total 6 (ROWID 5) ---------
fd (1): 1
price (2): 25
age (2): 35
Total 6 records.
OK
============================================================
결과 집합에서 'price' 최소값을 알고자 하는 경우.
============================================================
kql/myvol> select * from tbl where  fd = '1' extract by
min(price);
---------- 0 of total 1 (ROWID 3) ---------
fd (1): 1
price (1): 3
age (2): 13
Total 1 record.
OK
============================================================
결과 집합에서 'price' 최대값 또는 'age'최대값을 얻고자 하는 경우.
============================================================
kql/myvol> select * from tbl where  fd = '1' extract by
max(price age);
---------- 0 of total 1 (ROWID 4) ---------
fd (1): 1
price (2): 25
age (3): 135
Total 1 record.
OK
============================================================
결과 집합에서 'price'의 최대값과 최소값, 'age'의 최소값과 최대값을
한번에 얻고자 하는 경우.
============================================================
kql/myvol> select * from tbl where  fd = '1' extract by
minmax(price age);
---------- 0 of total 3 (ROWID 3) ---------
fd (1): 1
price (1): 3
age (2): 13
---------- 1 of total 3 (ROWID 4) ---------
fd (1): 1
```

```
price (2): 25
age (3): 135
---------- 2 of total 3 (ROWID 2) ---------
fd (1): 1
price (2): 10
age (2): 13
Total 3 records.
OK
```

'extract by minmax(price age)' 구문은 단지 minmax(price, age) 조건을 만족하는 레코드 rowid 집합을 구할 뿐이고, 실제 min(price), max(price), min(age), max(age) 값은 시나리오로 가져온 필드값을 조사해야 한다. 한 레코드가 min(price) 이면서 max(age) 일 수도 있으므로 minmax(price age)로 리턴되는 레코드 수는 4와 같거나 작을 수 있다.

4.3. 기타 응용

4.3.1. N-GRAM 색인

독크루저는 필드 속성을 Universal로 지정한 경우 N-GRAM 색인 방식으로 키워드를 추출한다. n-gram이란 인접한 n 개의 글자를 말하는 데 예를 들어 '프로그래밍'이란 단어에 대해 bi-gram은 '프로', '로그', '그래', '래밍'이며 tri-gram은 '프로그', '로그래', '그래밍'이다. 분해된 음절을 키워드로 색인 검색하는 방식이며 형태소 오분석으로 인한 검색 결과 누락을 방지하기 위한 방법으로 사용할 수 있다.

이 색인 방법을 사용하려면 다음과 같은 설정이 필요하다.

```
-------------- kql.rc ----------------
[universal]
keyword_case = lower
keyword_charset = euckr
ngram_level = 3

------------- ddl.kql -------------------
create table tbl ( title TEXT UNIVERSAL, body TEXT UNIVERSAL );
create index tt_idx on tbl ( title );
create index bd_idx on tbl ( body );
```

N-GRAM에 해당하는 필드를 검색할 때는 boolean 검색 방법을 사용한다.

사용 가능한 연산자는 다음과 같다.

이름	표기법	의미	질의어의 예
양측절단	*str*	str을 포함하는 키워드를 가지고 있는 문서를 검색	*정보*
좌측절단	*str	str로 끝나는 키워드를 포함하는 문서를 검색	*대학교
우측절단	str*	str로 시작하는 키워드를 포함하는 문서를 검색	서울*
한글자대치	str?str2	str1과 str2 사이에 한 문자가 들어간 키워드를 포함하는 문서 검색	충청?도
구검색	str	phrase exact matching으로써 정확히 일치하는 구문을 가진 문서를 검색	청룡영화대상이 가지는 의미

다음은 boolean 검색 방법을 사용한 예이다.

```
kql/myvol> create table tbl (fd1 TEXT NULL, fd2 TEXT UNIVERSAL);
OK

kql/myvol> create index idx1 on tbl(fd1);
OK

kql/myvol> create index idx2 on tbl(fd2);
OK

kql/myvol> insert into tbl set fd1='인천국제공항', fd2='인천국제공항';
OK

kql/myvol> select * from tbl where fd1='천국' natural;
Total 0 record.
OK

kql/myvol> select * from tbl where fd2='천국' boolean;
---------- 0 of total 1 (ROWID 0) ---------
fd1 (12): 인천국제공항
```

```
fd2 (12): 인천국제공항
Total 1 record.
OK
```

4.3.2. 언어와 문자 세트

독크루저는 한국어 이외에도 일본어, 중국어, 영어, 기타 유럽어를 지원한다. 기본으로 지정된 한국어가 아니라 다른 언어에 대해 색인이나 검색을 하고자 하는 경우에는 LANGUAGE, CHARSET을 설정한다. 언어와 문자 세트는 필드 타입이 TEXT 인 데이터에만 영향을 준다.

다음은 지원되는 언어와 해당 문자 세트를 나열한 표이다.

언어	문자 세트
한국어(korean)	EUCKR(완성형한글), UTF8(유니코드), USASCII(아스키)
일본어(japanese)	EUCJP(일본어 euc 코드), SJIS(일본어 shift-jis), UTF8, USASCII
중국어(chinese)	EUCCN(중국어 간체), BIG5(중국어 번체), UTF8, USASCII
n-gram(universal)	EUCKR, EUCJP, SJIS, EUCCN, BIG5, UTF8, USASCII
영어(english)	ENGLISH, USASCII, UTF8
덴마크어(danish)	DANISH, LATIN1, UTF8
네덜란드어(dutch)	DUTCH, LATIN1, UTF8
핀란드어(finnish)	FINNISH, LATIN1, UTF8
프랑스어(french)	FRENCH, LATIN1, UTF8
독일어(german)	GERMAN, LATIN1, UTF8
이탈리아어(italian)	ITALIAN, LATIN1, UTF8
노르웨이어(norwegian)	NORWEGIAN, LATIN1, UTF8
포르투갈어(portuguese)	PORTUGUESE, LATIN1, UTF8
스페인어(spanish)	SPANISH, LATIN1, UTF8
스웨덴어(swedish)	SWEDISH, LATIN1, UTF8
러시아어(russian)	RUSSIAN, CYRILLIC, UTF8
사용자언어(user0 ~ user9)	임의의 문자 세트, USER0 ~ USER9

각 언어에 대해 해당 언어 모듈이 형태소 분석 등을 전담하게 되는데, 하나의 언어에 대해 여러 가지 문자 세트가 필요한 경우나 형태소 분석과 관련해 커스터마이징이 필요한 경우 등을 지원하기 위해 사용자 언어 기능을 제공하고 있다. user0 ~ user9가 이에 해당한다(사용자정의 색인기). 형태소 분석 색인이 아니라 N-GRAM 색인 기법으로 색인하는 경우에 대한 필드 설정은 TEXT UNIVERSAL로 한다.

각종 KQL 명령어의 [LANGUAGE=*language*][CHARSET=*charset*] 구문에 위 언어와 문자 세트를 맞춰주도록 한다.

다음은 KQL 설정 파일(KQL configuration file)의 언어 섹션 부분이다. 다음과 같은 형식으로 언어 섹션을 설정하면 된다.

```
[language]
name=value
```

다음은 언어 섹션에서 지정할 수 있는 항목들이다.

[KOREAN]

항목	설명
keyword_case	영단어 키워드의 대소문자 통일 여부를 지정한다. (기본값=lower). ● mixed: 대소문자 그대로 사용 ● lower: 모두 소문자로 통일 ● upper: 모두 대문자로 통일 예) `keyword_case = lower`
query_compound_level	쿼리 복합명사 분해 레벨(1~5)을 설정한다. (기본값=2). 예) `query_compound_level = 5`
document_compound_level	문서 복합명사 분해 레벨(1~5)을 설정한다. (기본값=2).

항목	설명
	예) `document_compound_level = 5`
max_keyword_occurrence	키워드 위치 정보 최대값(1~127)을 설정한다. (기본값=12). 예) `max_keyword_occurence = 10`
use_word_keyword	워드단위 키워드를 추출하며 추출된 키워드에는 '@'가 접두어로 붙는다. (기본값=N). 예) `use_word_keyword = Y`
use_word_keyword_en	영단어의 경우 단어 그대로 키워드를 추출한다. (기본값=N). 예) `use_word_keyword = Y`
use_stem_keyword_en	영단어의 경우 스테밍 결과로 키워드를 추출한다. (기본값=Y). 예) `use_stem_keyword_en = N`
use_half_width_keyword_en	전각문자중 영문 알파벳을 전각과 1바이트 아스키 문자로 색인한다. (기본값=N). 예) `use_half_width_keyword_en = Y`
use_han_sound_keyword	한자독음의 한글 키워드 추출 여부를 설정한다. (기본값=N).

항목	설명
	예) `use_han_sound_keyword = Y`
use_symbol_keyword	특수문자의 키워드 추출 여부를 설정한다. (기본값=N). 예) `use_symbol_keyword = Y`
use_math_keyword	수식의 키워드 추출 여부를 설정한다. (기본값=N). 예) `use_math_keyword = Y`
use_tagged_keyword	품사(A/V/D) 태그를 부착하여 키워드 추출 여부를 설정한다. (기본값=N). 예) `use_tagged_keyword = Y`
prefer_noun_query	질의에서 명사 키워드 우선 추출 여부를 설정한다. (기본값=N). 예) `prefer_noun_query = Y`
allow_query_escaping	질의에서 첫글자가 '#'인 경우 형태소 분석을 하지 않고 그대로 키워드 추출 여부를 설정한다. (기본값=N). 예) `allow_query_escaping = Y`
stop_knx	키워드 추출 시 의존명사(것, 뿐, …)의 제외 여부를 설정한다. (기본값=N). 예)

항목	설명
	`stop_knx = Y`
stop_kad	키워드 추출 시 부사(매우, 빨리, …)의 제외 여부를 설정한다. (기본값=N). 예) `stop_kad = Y`
stop_kvv	키워드 추출 시 동사(먹다, 배우다, …)의 제외 여부를 설정한다. (기본값=N). 예) `stop_kvv = Y`
stop_enn	키워드 추출 시 영문(star, keyword, …)의 제외 여부를 설정한다. (기본값=N). 예) `stop_enn = Y`
stop_nud	키워드 추출 시 숫자(1, 2, …)의 제외 여부를 설정한다. (기본값=N). 예) `stop_nud = Y`

[UNIVERSAL]

항목	설명
keyword_case	영문 색인 시 대소문자 통일 여부를 지정한다. (기본값=lower). ● mixed: 대소문자 그대로 사용 ● lower: 모두 소문자로 통일

항목	설명
	● upper: 모두 대문자로 통일 예) `keyword_case = upper`
use_space_insensitive_keyword_query = use_space_insensitive_keyword_query_universal	질의문에 대한 Space insensitive 키워드의 생성여부를 결정한다. (기본값=1). ● 0: 사용하지 않음 ● 1: 사용함 영문에 대해 space insensitive하게 동작하려고 하면 ignore_space_insensitive_keyword_english를 0으로 하고 use_space_insensitive_keyword_query를 1로 설정해야 한다. 키워드 추출 예제) 입력 : "가졌다 교섭" (기본값=Y). ● Y: {가졌, 졌다, 다교, 교섭} ● N: {가졌, 졌다, 다_ _교,H 교섭} 예) `use_space_insensitive_keyword_query = N`
use_space_insensitive_keyword_document = use_space_insensitive_keyword_document_universal	색인문서에 대한 space insensitive 키워드 생성 여부를 결정한다. (기본값=Y). 예) `use_space_insensitive_keyword_document = N`
ignore_space_insensitive_keyword_english_universal	Space insensitive keyword 계열의 옵션의 설정이 되어 있더라도 영문의 경우에는 무시한다. 즉, 1로 설정하면 무조건 space sensitive하게 동작하도록 한다. 단, 영문만으로 이루어진 문장에 대해서만 적용된다.

항목	설명
	예) `ignore_space_insensitive_keyword_english_universal`
ngram_level=ngram_level_universal	n-gram 분해 레벨 (bigram,trigram)을 설정한다. (기본값=2). 2보다 작은 값은 2로 5보다 큰값은 5로 보정. 키워드 추출 예제) 입력 : "가졌다 교섭" 2 : {가졌,졌다,다교,교섭} 3 : {가졌다,졌다교,다교섭} 예) `ngram_level = 3`
use_symbol_normalize = use_symbol_normalize_universal	`를 '(작은따옴표)로 변경. (기본값=Y). 예) `use_symbol_normalize = N`
use_syllable_based_word_no = use_syllable_based_word_no_universal	음절 단위로 word_no 추출 여부를 설정한다. (기본값=Y). 예) `use_syllable_based_word_no = N`
sbcs_stop_level = sbcs_stop_level_universal	SBCS 색인어(영문,숫자 등)의 ngram 레벨에 예외를 설정한다. 즉, SBCS_STOP_LEVEL에 설정된 값 이하 길이의 색인어는 색인어로 추출이 되지 않는다. (기본값=0). ● SBCS_STOP_LEVEL = 1: 영문에 대해서 1음절 초과(바이트) 키워드만 생성 ● 값: nGramLevel 보다 작고 0보다 큰 값(기본값=0). 예) `sbcs_stop_level = 1`

항목	설명
mbcs_stop_level = mbcs_stop_level_universal	MBCS색인어(한,중,일등의 EUC문자)의 ngram 레벨에 예외를 설정한다. 즉, SBCS_STOP_LEVEL에 설정된 값 이하 길이의 색인어는 색인어로 추출이 되지 않는다. ● SMBCS_STOP_LEVEL = 1: 2바이트 문자에 대해서 1음절 초과(2바이트) 키워드만 생성 ● 값: nGramLevel 보다 작고 0보다 큰 값 예) `mbcs_stop_level = 1`
set_keyword_case_universal	기호가 포함된 색인어 생성 여부를 설정한다. (기본값=Y). 키워드 추출 예제 입력 : "가졌다-교섭" N : {가졌,졌다,교섭} Y : {가졌,졌다,다-,-교,교섭} 예) `use_symbol_universal = N`
max_docsize_universal	색인대상문서의 최대크기 설정. (기본값=1024) 예) `max_docsize_universal = 2048`
attach_wordph_document _universal	색인 문서에 대해 N-gram level 값보다 작은 어절만 어절 색인어로 추출 여부를 설정한다. (기본값=Y). 어절 색인어는 스페셜 마크가 부착된 형태로 제공: (0xFFFF) 일반키워드와 동일한 어절키워드도 생성. 예) `attach_wordph_document_universal = N`
attach_wordph_query _universal	쿼리문에 대해 N-gram level 값보다 작은 어절만 어절키워드로 추출 여부를 설정한다. (기본값=Y).

항목	설명
	어절키워드는 스페셜 마크가 부착된 형태로 제공: (0xFFFF) 일반키워드와 동일한 어절키워드는 생성하지 않음. 예) `attach_wordph_query_universal = N`
use_wordph_header _query_universal	어절키워드 앞에 부착된 태그(0xFFFF)를 사용할지 여부를 설정한다. (기본값=Y). ● Y: 어절 태그를 사용함 ● N: 어절 태그를 삭제함 예) `use_wordph_header_query_universal = N`

[CHINESE], [JAPAN]

항목	설명
keyword_case	영문 색인 시 대소문자 통일 여부를 지정한다. (기본값=lower). ● mixed: 대소문자 그대로 사용 ● lower: 모두 소문자로 통일 ● upper: 모두 대문자로 통일 예) `keyword_case = lower`
query_compound_level	쿼리 복합명사 분해 레벨(1~5)을 설정한다. (기본값=2). 예)

항목	설명
	`query_compound_level = 5`
use_word_keyword	영단어의 경우 단어 그대로 키워드 추출(영문). (기본값=Y). 예) `use_word_keyword = Y`
use_stem_keyword_en	영단어의 경우 스테밍 결과로 키워드를 추출한다. (기본값=Y). 예) `use_stem_keyword_en = N`
prefer_noun_query	질의에서 명사 키워드 우선 추출 여부를 설정한다. (기본값=N). 예) `prefer_noun_query = Y`
allow_query_escaping	질의에서 첫글자가 '#'인 경우 형태소 분석을 하지 않고 그대로 키워드 추출 여부를 설정한다. (기본값=N). 예) `allow_query_escaping = Y`
use_half_width_keyword_en	전각문자중 영문 알파벳을 전각과 1바이트 아스키 문자로 색인한다. (기본값=N). 예) `use_half_width_keyword_en = Y`
use_tagged_keyword	품사(A/V/D) 태그를 부착하여 키워드 추출 여부를 설정한다. (기본값=N). 예)

항목	설명
	`use_tagged_keyword = Y`

[ENGLISH], [DANISH], [DUTCH], [FINISH], [FRENCH], [GERMAN], [ITALIAN], [NORWEGIAN], [PORTUGUESE], [SPANISH], [SWEDISH], [RUSSIAN]

항목	설명
keyword_case	영문 색인 시 대소문자 통일 여부를 지정한다. (기본값=lower). ● mixed: 대소문자 그대로 사용 ● lower: 모두 소문자로 통일 ● upper: 모두 대문자로 통일 예) `keyword_case = upper`
use_tagged_keyword	품사(A/V/D) 태그를 부착하여 키워드 추출 여부를 설정한다. (기본값=N). 예) `use_tagged_keyword = Y`
max_keyword_occurrence	키워드 위치 정보 최대값(1~127)을 설정한다. (기본값=12). 예) `max_keyword_occurence = 10`
use_word_keyword	stemming되기 전 단어의 색인어로 저장 여부를 설정한다. (기본값=Y). 예)

항목	설명
	`use_word_keyword = N`
use_stem_keyword	stemming된 단어의 색인어로 저장 여부를 설정한다. (기본값=Y). 예) `use_stem_keyword = N`
use_morph_keyword	형태소 키워드의 추출 여부를 설정한다. (기본값=Y). 예) `use_morph_keyword = N`

다음은 각 언어별 태킹 설정 예이다.

```
…
[Korean] ; korean 언어섹션
keyword_case = lower
use_stem_keyword_en = y
use_word_keyword_en = y
use_han_sound_keyword = y
use_half_width_keyword_en = y
max_keyword_occurrence = 120

[Universal] ; Universal 언어섹션
keyword_case = lower
ngram_level = 2
…
```

4.3.3. 구조화 된 문서 (KTML)

문서 본문에 나타나는 키워드의 가중치는 문서나 문서 집합 내에서 자동으로 결정되는 값이기 때문에 사용자가 임의로 값을 조절할 수 없었다. 검색용 데이터를 생성할 때 특정 키워드를 특정 문서에서 중요하게 사용하고 싶은 경우 Konan Text Markup Language(KTML)를 사용해 키워드 가중치를 부여할 수 있다. KTML은 가중치 정보를 검색용 데이터와 같이 입력함으로써, 색인 시 가중치 정보까지 색인될 수 있도록 하는 수단이다. KTML을 이용한 가중치 정보 생성은 텍스트 필드에 대해서만 적용된다.

다음은 KTML 태그에 대한 설명이다.

태그	설명
"<tm>" 또는 "<ktml>"	색인 시 문서의 시작을 알려준다.
<em[0-7]	이 사이에 나오는 키워드들이 em 값에 의해 가중치가 부여된다. 기본은 0이며 숫자가 높을수록 가중치도 높아진다.
<vt> </vt>	있는 그대로 키를 추출하기 위해 사용한다. 특정 문장 전체를 키워드로 잡아야 하는 경우 유용하다.

다음 예를 보면 첫 번째 검색에서는 "검색"이라는 키워드의 빈도수가 많은 0 번 문서가 상위에 랭크되었으나 두 번째 검색에서는 "검색엔진"이 키워드 가중치를 받아서 키워드 빈도수를 누르고 1 번 문서가 상위에 랭크되었다. "<vt>"의 예제를 보면 "h a"가 공백 기호에도 불구하고 키워드로 잡혔으며 특수기호인 "!*)" 조차도 있는 그대로 키로 잡혔음을 알 수 있다.

```
kql/tv2>
kql/tv2> select * from tab where f1 = '검색' natural;
---------------- 1 of total 2 (ROWID 0) --------------------
f1 (91): 통합검색 서비스를 통해 그동안 운행노선 중심으로 이루어진 검색을
여행상품 및 게시판 검색까지
---------------- 2 of total 2 (ROWID 1) --------------------
f1 (27): 전문 검색엔진 업체 - 엠파스
Total 2 records.
OK

kql/tv2> update tab set f1 = '<tm>전문 <em3>검색엔진</em> 업체 –
엠파스' where
$ROWID = 1;
OK

kql/tv2> select * from tab where f1 = '검색' natural;
---------------- 1 of total 2 (ROWID 1) --------------------
f1 (41): <tm>전문 <em3>검색엔진</em> 업체 - 엠파스
---------------- 2 of total 2 (ROWID 0) --------------------
f1 (91): 통합검색 서비스를 통해 그 동안 운행노선 중심으로
```

```
이루어진 검색을 여행상품 및 게시판 검색까지
Total 2 records.
OK

kql/tv2> select * from tab;
---------------- 1 of total 2 (ROWID 0) --------------------
f1 (18): <tm><vt> h a </vt>
---------------- 2 of total 2 (ROWID 1) --------------------
f1 (16): <tm><vt>!*)</vt>
Total 2 records.
OK

kql/tv2> select * from tab where f1 = 'h';
Total 0 record.
OK

kql/tv2> select * from tab where f1 = ' h a ';
---------------- 1 of total 1 (ROWID 0) --------------------
f1 (18): <tm><vt> h a </vt>
Total 1 record.
OK

kql/tv2> select * from tab where f1 = '!*) ';
---------------- 1 of total 1 (ROWID 1) --------------------
f1 (16): <tm><vt>!*)</vt>
Total 1 record.
OK

kql/tv2>
```

독크루저 데몬

이 장에서는 독크루저의 데몬이 동작하는 동안 처리할 수 있는 다양한 부가서비스 제어 명령어와 기타 데몬 제어 명령어들의 사용에 대해 설명한다.

5.1. DOCRUZERD

검색 서비스를 위한 데몬 실행 파일이다. 다음은 DOCRUZERD의 사용법이다.

- 구문

```
docruzerd {start | stop| reload} [ –f config_file ]
```

구성요소	설명
start	검색 서비스를 시작
stop	서비스 중인 데몬을 중지
reload	검색 서비스는 중단하지 않으면서 독크루저 설정 파일의 변경내용을 새로 적용
–f *config_file*	해당 독크루저 설정 파일을 지정하는 옵션 생략하면 해당 디렉터리의 docruzerd.rc를 참조

- 예제

```
$ ./docruzerd start
----------------- docruzerd.log -----------------
>DOCRUZER: READY [Thu Jul 14 10:49:27 2005] >DOCRUZER: Start up
search engine daemon.
[Thu Jul 14 10:49:27 2005] >DOCRUZER: Reading configuration file
'docruzerd.rc'.
[Thu Jul 14 10:49:27 2005] >DOCRUZER: Reading included
configuration file 'scn.rc'.
[Thu Jul 14 10:49:32 2005] >DOCRUZER: READY
```

```
>DOCRUZER: READY
-------------------------------------------------

$ ./docruzerd stop
>DOCRUZER: SHUTDOWN
$
```

여러 가지 이유로 인하여 데몬 기동이 실패하는 경우 터미널에는 기동이 실패했다는 에러 메시지가 출력된다. 기동 실패 이유에 대한 자세한 정보는 시스템 로그 파일(~.msg)에 기록된다.

```
$docruzerd start -f docruzerd.rc
"Error: 'daemon startup failed.(thrd:698)'
```

이 경우에는 로그 디렉터리의 오늘 날짜에 해당하는 .msg 파일을 통해 데몬 start가 실패한 이유를 분석할 수 있다.

독크루저 UI: Windows용 독크루저 데몬 실행

Windows 운영체제용 독크루저는 검색 데몬 실행을 위해 별도의 실행 프로그램을 제공한다. 도스 명령창을 이용하여 데몬을 실행한 후 도스 명령창을 종료시키면 데몬도 같이 종료되는 현상을 방지하며, 시스템이 실행될 때 데몬을 자동으로 실행할 수 있도록 설정할 수 있다. 설치된 패키지의 bin 디렉터리에 있는 docruzer.exe가 Windows 용 데몬 제어 프로그램인 독크루저 UI다.

Windows 운영체제는 또한 네트워크 상에 존재하는 다른 컴퓨터의 공유 디렉터리를 네트워크 디렉터리로 연결하여 로컬 디스크처럼 제어할 수 있다. 독크루저 UI는 실행된 데몬이 네트워크 디스크 자원을 이용할 수 있도록 부가 설정 파일을 필요로 한다.

독크루저 UI는 Windows 트레이에서 실행하는 것과 Windows 서비스를 이용하는 두 가지 형태로 실행할 수 있다. 설치된 독크루저 디렉터리 중 bin 디렉터리에 트레이용은 docruzer.exe, Windows 서비스용은 docruzer_svc.exe로 존재한다. 둘 중 실행하고 싶은 UI 실행 파일을 docruzer.exe로 명명하여 사용한다.

독크루저 UI는 다음의 기능을 제공한다:

- 독크루저 데몬이 예기치 않게 종료되었을 경우 재기동시킴
- 독크루저 데몬을 일반 계정으로 실행시킬 수 있음

- 다수의 네트워크 드라이브 사용 가능

- 서비스에 등록하여 Windows 시작 시 독크루저 데몬의 실행

네트워크 드라이브나 일반 계정 실행 기능을 사용하고자 할 경우는 독크루저 UI용 설정 파일을 정의해야 한다. 다음은 설정 파일(networkdrive.rc)의 예이다.

```
LocalUserName = administrator       ;;독크루저 데몬을 구동할 계정명
LocalPassword = 1234                ;;독크루저 데몬을 구동할 계정 암호
no_of_networkdrive = 2              ;;사용할 네트워크 드라이브 개수

;;네트워크 드라이드 명(=X) 및 접속 환경
<networkdrive = X>
RemoteUserName = administrator
RemotePassword = kql
RemotePath = \\192.168.7.17\logs
</networkdrive>

<networkdrive = Z>
RemoteUserName = administrator
RemotePassword = kql
RemotePath = \\192.168.7.17\fileTest
</networkdrive>
```

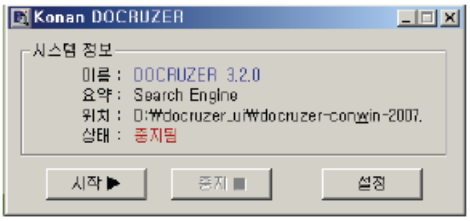
Windows 트레이의 UI로 실행하기

독크루저 UI Windows 트레이 버전을 구동시켜 독크루저를 실행하는 절차는 다음과 같다.

1. 독크루저 UI Windows 트레이 버전을 구동시키면 다음과 같은 화면이 나타난다. Windows 트레이에서 독크루저 UI 아이콘을 클릭한다.

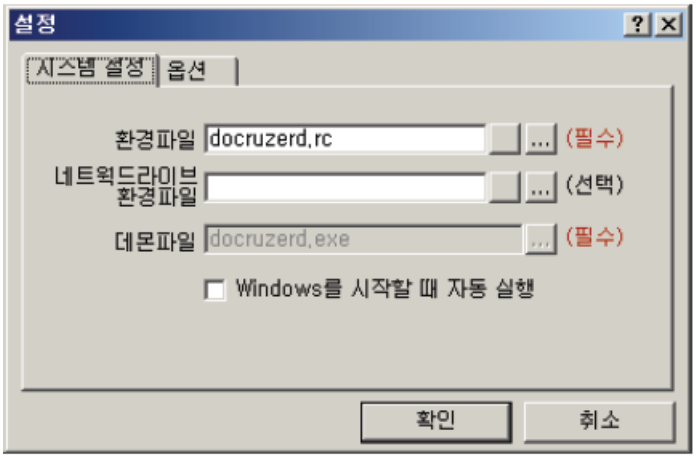
[그림 5.1] Windows 트레이의 독크루저 UI 아이콘

2. 독크루저 UI 화면에서 **[설정]** 버튼을 클릭한다.



[그림 5.2] 독크루저 UI 화면

3. 설정 화면에서 항목에 맞게 값을 입력하고 **[확인]**을 클릭한다. UI 프로그램은 데몬 파일과 같은 경로에 위치하여야 한다.



[그림 5.3] 독크루저 설정 화면

항목	설명
환경파일	독크루저 설정 파일(docruzerd.rc)을 입력한다.
네트웍 드라이브 환경파일	UI 환경설정 파일을 입력한다.
데몬 파일	docruzerd.exe를 입력한다.

4. 독크루저 UI 화면에서 **[시작]** 버튼을 클릭하면 검색 서비스를 시작하게 된다.

Windows 서비스로 UI 시작하기

독크루저 UI Windows 서비스 버전의 경우, 서비스의 시작 유형이 '자동'일 경우 Windows 시작 시 독크루저 UI와 독크루저 데몬이 실행된다. 이때 Windows 트레이 버전과는 달리 Windows 사용자 로그인을 하지 않아도 실행된다.

- 서비스 등록

  Windows 서비스로 UI 를 실행하기 위해서는 서비스 등록을 해야 한다. docruzer_svc.exe 파일명을 docruzer.exe로 변경한 후 실행한다. 도스 명령 창에서 다음과 같이 입력하면 서비스로 등록된다. 단, 이미 서비스가 등록되어 있는 경우는 에러 메시지를 반환한다.

```
docruzer -c '서비스명' -f '환경파일' -n '네트워크 드라이브 환경파일'
```

  다음의 예제와 같이 입력한다.

```
bin> docruzer -c DOCRUZER -f docruzerd.rc -n networkdrive.rc
```

- 서비스 시작, 종료, 삭제, 상태 보기

  서비스를 등록한 후 서비스 시작과 종료, 삭제, 상태 보기는 다음 구문으로 실행한다.

```
docruzer -s|-t |-d|-u '서비스명' (-s: 시작, -t: 종료, -d: 삭제, -u: 상태보기)
```

  다음의 예제와 같이 입력한다.

```
bin>docruzer -s DOCRUZER
```

5.2. 시나리오 (SCENARIO)

일반 용법

독크루저로 검색을 하기 위해서는 검색 시나리오가 필요하다. 검색 시나리오는 복잡한 검색 과정을 미리 정형화시켜서 검색의 효율을 높인다.

시나리오는 정의 파일은 scenario 디렉터리에 .scn 확장자로 저장하여 관리한다. 시나리오 정의 파일에는 하나의 시나리오만 정의하는 것을 원칙으로 하며 시나리오명과 파일명을 동일하게 한다.

- 일반 용법

```
scenario scenario_name
{
volume = volume_name
table = table_name
field = field_name [( [size], start_tag, end_tag)]
complex_field = field_name([size], start_tag, end_tag, delemeter,
KEYWORDONLY | MATCHEDONLY | MULTIPOINT | FROMSTART)
...
}
```

KQL 검색 명령어 구문의 "SELECT column_name_list FROM table_name" 부분을 지정해 주는 것이 시나리오의 역할이다. 검색 대상이 되는 테이블은 *link_name*, *volume_name*, *table_name*의 3 단계로 지정한다.

구성요소	설명
volume_name	KQL 설정 파일(kql.rc)에서 volume 으로 지정한 볼륨명
table_name	테이블의 이름

검색 결과로 보여줄 필드명을 field 나 complex_field 로 나열한다.

field 를 이용하여 검색 결과 필드를 정의할 때는 필드명만 적어주거나 크기, 하이라이트 옵션을 함께 지정하면 된다. 요약문은 필드가 키워드를 포함하고 있을 경우 키워드를 중심으로 크기만큼 추출하며 포함하지 않은 경우는 데이터의 처음부터 크기만큼 추출한다. 크기(size)를 지정하면 지정한 크기만큼을 해당 필드에 요약문으로 추출한다. 요약 기능은 검색 결과 목록에서 너무 많은 내용을 보여주기 힘들 때 유용하다. 모든 내용을 검색할 때는 크기를 지정하지 않으면 된다.

구성요소	설명
field_name	필드명
size	크기
start_tag, end_tag	하이라이팅 태그로 검색 결과를 반환할 때 키워드의 앞, 뒤에 덧붙인다. 웹으로 검색 결과를 제공할 경우 검색 결과 화면에서 키워드를 강조하는 효과를 낼 수 있는 HTML 태그 문자열을 입력한다.

complex_field를 이용하여 검색 결과 필드를 정의하는 것은 3.2.10 이상에서 지원하는 기능으로 부가적인 요약 기능을 제공한다. 하나의 필드에 여러 개의 첨부 문서가 색인되어 있는 경우와 같이 구분자로 여러 개의 데이터가 색인되어 있는 경우에 데이터별로 요약/하이라이팅된 검색 결과를 제공한다.

구성요소	설명
field_name	필드명
size	크기
start_tag, end_tag	하이라이팅 태그로 검색 결과를 반환할 때 키워드의 앞, 뒤에 덧붙인다. 웹으로 검색 결과를 제공할 경우 검색 결과 화면에서 키워드를 강조하는 효과를 낼 수 있는 HTML 태그 문자열을 입력한다.
delemeter	색인 시 구분자로 등록한 것과 같은 값으로 설정한다. 반드시 큰따옴표(" ")를 사용하여 등록하고, 별도의 설정 값이 없는 경우 빈문자열(" ")을 입력한다. 필드의 내용이 *delimiter* 에 의해 다수의 데이터가 색인되지 않았지만 complex_field 기능을 사용하고 싶은 경우는 빈 문자열(" ")로 설정한다.
KEYWORDONLY	요약 옵션, 본문에서 일치한 키워드만 출력
MATCHEDONLY	요약 옵션, 키워드가 일치하지 않는 경우 요약문을 추출하지 않음
MULTIPOINT	요약 옵션, 키워드가 출현한 여러 지점 요약 ('요약/하이라이팅 (SUMMARIZATION/HIGHLIGHT)' 참조)
FROMSTART	요약 옵션, 무조건 본문 처음부터 일정 길이를 요약

```
$ cat docruzerd.rc
...
kql_link = h1, kql.rc
include scn.rc

$ cat kql.rc
```

```
...
volume = tv, "vol.rc"

$ cat scn.rc

scenario nr1
{
volume = tv
table = GISA
field = TT ( , "<b>","</b>")
field = BD ( 200, "<b>", "</b>")
field = DT
field = AU
field = SC
field = SO
}

$
```

다른 테이블과 조인

표시해야 할 필드가 지정한 검색 테이블에 없는 경우는 외래키를 이용하여 가져오도록 한다.

```
field = table_name.field_name ( primary_key_field )
```

조인 조건 필드가 다른 테이블의 프라이머리 키이거나 복합키인 경우는 다음 구문을 사용할 수 있다.

```
field = table_name.field_name ( composite_primary_key_field_list )
```

```
create table movie
(
movie_id int32 primary,
movie_name text
);

create table actor
(
actor_id int32 primary,
actor_name text,
```

```
movie_id int32
);

create index idx1 on movie (movie_id);
create index idx2 on movie (movie_name);
create index idx1 on actor (actor_id);
create index idx2 on actor (movie_id);

$ cat scn.rc
scenario actor
{
link = h1
volume = tv
table = actor
field = actor_id
field = actor_name
field = movie.movie_name(movie_id)
}
$
```

'table_name.field_name'은 디스플레이 할 필드이다. {< 다음의 'field_name'은 디스플레이 대상인 테이블의 조건 필드이며 '[field_name]'은 지정한 검색 테이블에서의 조건 필드이다. 따라서 두 테이블 간의 조건 필드값이 맞는 경우에 한해 디스플레이 할 필드를 가져오게 된다. 주로 해당 테이블에는 코드만 있고, 다른 테이블에 코드명이 있는 경우에 많이 사용하게 된다.

```
field = table_name.field_name {<field_name = [field_name]>}
```

구성요소	설명
table_name.field_name	디스플레이 할 필드
field_name	디스플레이 대상인 테이블의 조건 필드
[field_name]	지정한 검색 테이블에서의 조건 필드

```
create table play
(
id int32 primary,
title text,
genre_id string,
place_id string,
```

```
program text
);

create table genre
(
id string,
name text
);

create table place
(
id string,
name text
);

create index id_idx on play (id);
create index title_idx on play (title);
create index program_idx on play (program);
create index id_idx on genre (id);
create index id_idx on place (id);
insert into genre set id = 'opera', name = '오페라';
insert into place set id = '13', name = '토월극장';
insert into play set id = 1, title = '헨젤과 그레텔',
genre_id = 'opera', place_id = '13', program = '엥겔베르트 훔퍼딩크는
1854년 독일 지크부르크에서 태어나 1921년 노이스트렐리쯔에서 67세를
일기로 세상을 떠날 때까지 불과 세 개의 오페라를 작곡했지만 이
"헨젤과 그레텔" 때문에 지금까지 세계적인 오페라 작곡가로서 명성을
누리고 있다. 당시 이탈리아에서는 베르디의 뒤를 이어 사실주의 오페라가
계속 발전하고 있었으나 독일에서는 바그너를 이을 마땅한 오페라가
없었을 때 마침 이 작품 하나로 독일 오페라의 돌파구를 마련한 운 좋은
작곡가인 셈이다.
작곡가가 이 오페라를 쓰게 된 동기는 그의 누이동생 베테(A.Wette)가
그림 형제가 쓴 "어린이와 가정을 위한 동생"을 정리해서 오빠인
훔퍼딩크에게 가족을 위한 오페라를 써달라고 부탁해서 작곡된 것이
바로 이 작품이다.';
```

다음은 지정한 테이블의 필드만 가져온 경우로 코드 ID 를 그대로 보여주고 있다.

```
$ cat scn.rc

scenario play_raw
```

```
{
link = h1
volume = tv
table = play
field = id
field = title
field = genre_id
field = place_id
}

$ ./crzcli 192.168.1.233 7577 play_raw
>title_idx = '헨젤' allword
질의어: 'title_idx = '헨젤' allword', 검색문서수: 1, 검색시간: 38 msec
+---------- 1/1 번째 문서 (0) SCORE : 10000 ----------+
[0](1) 1
[1](13) 헨젤과 그레텔
[2](5) opera
[3](2) 13
[KQS] [0]
[CLS] 0 개의 클러스터가 검색되었습니다.
>
$
```

다음은 조건에 맞는 코드명을 가져와서 보여주는 경우이다.

```
$ cat scn.rc
scenario play_chained
{
link = h1
volume = tv
table = play
field = id
field = title
field = genre.name{<id=[genre_id]>}
field = place.name{<id=[place_id]>}
}

$ ./crzcli 192.168.1.233 7577 play_chained
>title_idx='헨젤' allword
질의어: 'title_idx='헨젤' allword', 검색문서수: 1, 검색시간: 0 msec
+---------- 1/1 번째 문서 (0) SCORE : 10000 ----------+
```

```
[0](1) 1
[1](13) 헨젤과 그레텔
[2](6) 오페라
[3](8) 토월극장
[KQS] [0]
[CLS] 0 개의 클러스터가 검색되었습니다.
>
$
```

최소 검색어 수 지정

검색 방법 중 자연어 검색(NATURAL)을 사용하는 경우 시나리오에서 검색어 개수별로 일치되어야 할 최소 검색어 개수를 지정할 수 있다.

- 구문

```
control_relevance = { number, ... }
```

구성요소	설명
number	검색어 개수별로 일치되어야 할 검색어 개수이다. 순서는 각각 검색어 1 개일 때, 2 개일 때, 3 개일 때 일치 되어야 할 검색어 개수이며 최대 10 개까지 지정할 수 있다.

- 예제

```
b------------movie.scn---------------
scenario     movie
{
    link                = h1
    volume              = myvol
    table               = movie
    control_relevance   = {1,1,2,2,3,3}

    field               = movie_id
    field               = movie_name
}
```

```
$
```

분산 볼륨

분산볼륨 기능은 하나의 엔진 내에 있는 복수의 볼륨을 동시 검색하는 기능이다. 각 볼륨 테이블은 동일한 스키마를 가져야 하며 시나리오 파일에 각 복수의 볼륨명, 테이블명을 등록하고 독크루저 API인 Search()를 통해 검색하여야 한다. 분산볼륨의 목적은 하나의 볼륨이 너무 클 경우 실시간 색인 및 검색 속도가 느려질 수 있으므로 볼륨을 나누어 쓰면서 복수개 볼륨의 동시 검색이 용이하게 하기 위함이다. 카테고리별 검색과 통합 검색을 같이 사용 할 경우, 예를 들어 쇼핑몰에서 카테고리별로 볼륨을 구성한 뒤 카테고리별 검색은 해당 볼륨에서, 통합 검색은 분산볼륨 기능을 통해 검색하는 것이 가능하다. 이 경우, 볼륨 크기가 작아져 실시간 색인 및 검색 속도 향상을 기대할 수 있다.

다음과 같이 시나리오 파일에 각 복수의 볼륨명.테이블명을 등록할 수 있다. 각 테이블들은 동일한 스키마를 가지고 있어야 한다.

```
-----------sample.scn---------------
{
    link      = h1
    volume  = myvol
    table    = tbl, myvol2.tbl, myvol3.tbl

    field    = tt
    field    = bd (200, "<b>", "</b>")
    field    = dt
    field    = au
    field    = sc
    field    = so
}
```

시나리오 볼륨 크기 제한

독크루저는 라이선스에 대해 볼륨 크기를 체크하는데, 단일 볼륨을 사용하는 경우에는 색인에서 라이선스에 대한 볼륨크기를 체크하고, 분산볼륨을 사용하는 경우에는 검색에서 라이선스에 대한 볼륨크기를 체크하는데, 이것이 시나리오 볼륨 크기 제한이다.

독크루저 API인 Search()를 사용하는 경우에, 해당 시나리오에 등록된 볼륨의 크기가 라이선스의 볼륨크기를 초과하는 경우에는 다음과 같은 메시지와 함께 검색이 제한된다.

```
scenario scenario_name exceed licensed volume size limit
(xxx MB > yyy MB) (C483861)
```

그런데 볼륨크기가 계속 증가하다가 갑자기 검색이 되지 않으면 안되기 때문에 확인 차원에서 데몬은 주기적으로 시나리오에 등록된 볼륨의 크기를 체크하여 볼륨의 크기에 변동이 생기면 아래와 같은 로그를 시스템 로그 파일에 출력한다.

```
scenario volume size updated 450 MB -> 451 MB (licensed 10240 MB,
4.40% used) [scenario_name]
```

관리자는 이 로그를 확인하여 시나리오에 등록된 총 볼륨의 크기가 라이센스의 볼륨크기에 근접하는 경우 미리 조치를 취하여 검색이 제한되는 것을 방지하도록 해야한다.

5.3. 독크루저 인터프리터 (DOCRUZERD)

독크루저 인터프리터는 서비스 중인 검색 데몬의 기능을 제어하고 관리하기 위한 명령어를 처리한다. 인터프리터를 이용한 명령어 적용은 검색 데몬의 중지, 즉 검색 서비스의 중지 없이 데몬 기능을 제어하고자 할 때 유용하다.

```
$bin>docruzerd
Welcome to DOCRUZER Interpreter (3.5.2 2010-02-08 15:11:55).
Type 'help' for help. Commands end with ';'.
Copyright (c) 1999-2011 Konan Technology, Inc.
DOCRUZER>
```

서비스 중인 검색 데몬을 제어하기 위한 주요 명령어 목록은 다음 표와 같다. 검색 데몬 제어 명령은 반드시 원하는 검색 데몬으로 '*connect*'가 이루어진 후에 실행한다. 명령어는 대소문자 구분이 없으나 사용자정의 식별자는 대소문자를 구분한다.

인터프리터 제어

명령어 구문	기능
CONNECT *server_ip:daemon_service_port*	검색 데몬에 연결
DISCONNECT	연결 해제

명령어 구문	기능
QUIT	인터프리터 종료

환경 변수 제어

명령어 구문	기능
SET [*parameter_name* | *parameter_name* = *parameter_value*]	검색 환경 변수값 확인/변경

모듈 제어

명령어 구문	기능
SHOW(LIST) MODULES	모듈 목록 보기
EXPLAIN MODULE *module_number* [*domain_number*]	모듈 상세 정보
COMPILE MODULE *module_number* [*domain_number*]	모듈 사전 컴파일
RELOAD MODULE *module_number* [*domain_number*]	모듈 사전 재적재

시나리오 제어

명령어 구문	기능
SHOW SCENARIOS	시나리오 목록 보기
EXPLAIN SCENARIO *scenario_name*	시나리오 상세 정보
RELOAD SCENARIO	시나리오 재적재

제품키 정보

명령어 구문	기능
SHOW DAEMON PRODUCT KEYS	모든 제품키 정보
EXPLAIN DAEMON PRODUCT KEY *index*	*index* 번 제품키 정보

5.3.1. 인터프리터 제어 명령

데몬 제어 명령은 독크루저 인터프리터를 사용하여 실행 중인 데몬에 접속, 해제하는 명령어이다.

- CONNECT

서비스 중인 독크루저 데몬에 접속하는 명령어이다.

– 구문

```
CONNECT server_ip:service_port_number;
```

구성요소	설명
server_ip	독크루저 데몬이 실행되고 있는 서버의 IP 주소
service_port_number	독크루저 설정 파일(docruzerd.rc)에서 등록한 'daemon_service_port'의 포트번호

– 예제

```
$bin> docruzerd
Welcome to DOCRUZER Interpreter (3.5.2 2011-01-24 15:11:55).
Type 'help' for help. Commands end with ';'.
Copyright (c) 1999-2011 Konan Technology, Inc.

DOCRUZER> connect 192.168.7.5:8899;
connected
```

- DISCONNET

  연결된 검색 데몬과의 연결 해지명령어이다. 해당 데몬에 접속된 후에 실행 가능하다.

  – 구문

  ```
  DISCONNECT;
  ```

- QUIT

  인터프리터를 종료하는 명령어이다. 해당 데몬에 접속된 후에 실행 가능하다.

  – 구문

  ```
  QUIT
  ```

5.3.2. 환경 변수 제어 명령

독크루저 인터프리터에서는 동적으로 환경 변수를 제어할 수 있다. SET 명령어는 독크루저 설정 파일(docruzerd.rc)에서 설정한 환경 변수값을 인터프리터 상에서 확인하거나 새로운 값으로 할당하는 명령어이다.

- 구문

```
SET [parameter_name | parameter_name = parameter_value];
```

구성요소	설명
parameter_name	환경 변수명이며, 환경 변수명 만으로 명령어를 실행할 경우는 현재 설정된 변수값을 확인할 수 있다.
parameter_name = *parameter_value*	환경 변수와 값을 이용하여 실행할 경우 현재 설정된 환경 변수의 값을 새로운 값으로 설정한다.

SET 명령어를 이용하여 변경할 수 있는 환경 변수는 제한되어 있다. 독크루저 환경 변수 목록은 '독크루저 설정 파일 (docruzerd.rc)'에서 확인할 수 있다. 독크루저 인터프리터를 이용하여 환경 변수를 변경하였을 때 해당 값의 유효 기간은 독크루저 종료 전까지이다. 환경 변수값을 영구적으로 반영하고 싶은 경우는 독크루저 설정 파일에서 값을 변경한 후 설정 파일을 다시 반영해주는 'docruzerd reload' 명령어를 이용하여 적용하거나, 독크루저를 재시작한다.

5.3.3. 모듈 제어 명령

독크루저가 제공하는 일부 기능은 독립된 모듈에 의해 제공된다. 모듈 제어 명령은 각 모듈들을 제어하거나 독크루저 데몬이 사용 중인 모듈 정보를 확인하는 명령들이다.

SHOW MODULES

현재 데몬이 사용하고 있는 모듈 목록을 반환한다. 독크루저 데몬이 서비스 제공을 위해 적재한 모듈 리스트를 보여준다.

- 구문

```
SHOW MODULES;
```

- 예제

```
DOCRUZER> connect 192.168.7.5:8899;
connected

DOCRUZER> show modules;
+----+--------+-------------------------------------------------+
| no | domain | version                                         |
+----+--------+-------------------------------------------------+
| 74 | 0      | Sep 13 2006 17:25:01 (Tue Oct 31 10:52:52 2006) |
+----+--------+-------------------------------------------------+
| 80 | -      | Jul 7 2006 15:56:43                             |
+----+--------+-------------------------------------------------+
| 81 | 0      | Sep 13 2006 17:25:14 (Thu Apr 12 13:12:54 2007) |
+----+--------+-------------------------------------------------+
Total 3 module(s) loaded.
OK.
```

LIST MODULES

이 명령어는 제품 키에 의해 사용할 수 있는 모듈 목록을 제시해주는 명령어이다. 독크루저 제공하는 기능 모듈은 각 각 번호가 부여되어 있어 해당 번호로 모듈의 기능을 제어한다. 일부 모듈은 서비스를 제공하기 위해 사전을 사용하는데, 동일한 모듈이 사용할 수 있는 사전은 한 개 이상일 수 있다. 이들 사전의 구분은 '도메인' 번호로 한다. 사전의 기본 도메인 0 번부터 시작한다.

- 구문

```
LIST MODULES;
```

- 예제

```
DOCRUZER> connect 192.168.7.5:8899;
connected

DOCRUZER> list modules;
+----+-------------------------------------+--------+
| no | description                         | domain |
+----+-------------------------------------+--------+
| 74 | automatic keyword completion module | 0 - 50 |
+----+-------------------------------------+--------+
| 75 | spell checker module                | -      |
```

```
+----+-------------------------------------+--------+
| 76 | keyword recommendation module       | 0 - 50 |
+----+-------------------------------------+--------+
| 80 | daemon authentication module        | -      |
+----+-------------------------------------+--------+
| 81 | popular keyword module              | 0 - 50 |
+----+-------------------------------------+--------+
| 87 | daemon glossary anchoring module    | 0 - 50 |
+----+-------------------------------------+--------+
| 82 | daemon synonym module               | 0 - 50 |
+----+-------------------------------------+--------+
| 86 | daemon category ranking module      | 0 - 50 |
+----+-------------------------------------+--------+
| 88 | unformatted file store module       | 0 - 50 |
+----+-------------------------------------+--------+
| 96 | daemon scenario module              | -      |
+----+-------------------------------------+--------+
| 98 | daemon text summarization module    | -      |
+----+-------------------------------------+--------+
Total 10 module(s).
OK.
```

EXPLAIN MODULE *module_number* [*domain_number*];

독크루저 데몬이 서비스를 제공하기 위해 적재한 모듈에 대한 상세 정보를 출력해주는 명령어이다.

- 구문

```
EXPLAIN MODULE module_number [domain_number];
```

구성요소	설명
module_number	모듈 번호
domain_number	해당 모듈이 사용하는 사전의 도메인 번호

- 예문

```
DOCRUZER> explain module 74 0;
+-------------+-------------------------------------------------+
```

```
| module no   | 74                                                  |
+-------------+-----------------------------------------------------+
| domain no   | 0                                                   |
+-------------+-----------------------------------------------------+
| version     | Sep 13 2006 17:25:01 (Tue Oct 31 10:52:52 2006) |
+-------------+-----------------------------------------------------+
| description | automatic keyword completion module             |
+-------------+-----------------------------------------------------+
| meta data   | -                                                   |
+-------------+-----------------------------------------------------+
OK.
```

COMPILE MODULE *module_number* [*domain_number*];

독크루저가 제공하는 모듈 중 일부는 사용자가 작성한 사전을 사용한다. 사용자 사전은 텍스트 기반 형식으로 작성한다. 독크루저는 해당 사전을 시스템이 사용하는 형식으로 변환하여 사용하는데, 이때 사용하는 명령어가 COMPILE MODULE이다. 사용자 사전 컴파일 명령어가 성공적으로 수행되면 확장자가 'compiled'라는 시스템 형식의 데이터 파일이 생성된다.

각 모듈이 사용하는 사용자 사전과 컴파일에 대한 설명은 본 매뉴얼의 각 기능별로 상세 기술되어 있다. 모듈이 사전을 컴파일하여 사용하기 위해서는 독크루저 설정 파일에 'daemon_data_location' 항목이 모듈 디렉터리의 경로를 값으로 하여 설정되어 있어야 하고, 사용할 사전도 해당 위치에 있어야 한다.

- 구문

```
COMPILE MODULE module_number [domain_number];
```

구성요소	설명
module_number	모듈 번호
domain_number	해당 모듈이 사용하는 사전의 도메인 번호

- 예제

```
DOCRUZER> compile module 74 0;
OK.
```

RELOAD MODULE *module_number* [*domain_number*];

COMPILE MODULE 명령어에 의해 시스템 형식으로 변환된 사전을 서비스 중인 독크루저에 반영하도록 하는 명령어이다. 명령어가 성공적으로 실행되면 컴파일된 사전의 내용이 검색 서비스에 바로 적용된다. 독크루저를 재시작하는 경우는 해당 사전은 자동으로 적재되므로 RELOAD MODULE 명령어를 실행하지 않아도 된다.

- 구문

```
RELOAD MODULE module_number [domain_number];
```

구성요소	설명
module_number	모듈 번호
domain_number	해당 모듈이 사용하는 사전의 도메인 번호

- 예제

```
DOCRUZER> reload module 74 0;
OK.
```

5.3.4. 시나리오 제어 명령

독크루저 인터프리터를 이용하여 독크루저 시스템에 등록된 시나리오 정보를 확인하거나 변경할 수 있다. 시나리오 제어 명령을 실행하기 전에 CONNECT 명령어를 이용하여 서비스 중인 독크루저와 접속을 수행해야 한다.

SHOW SCENARIOS;

등록된 시나리오 목록을 보여주는 명령어이다.

- 구문

```
SHOW SCENARIOS;
```

- 예제

```
DOCRUZER> show scenarios;
+----------+
| scenario |
+----------+
```

```
| news     |
| app      |
+----------+
Total 2 scenario(s).
OK.
```

EXPLAIN SCENARIO

*scenario_name*으로 정의된 시나리오의 상세 정보를 보여준다.

- 구문

```
EXPLAIN SCENARIO scenario_name;
```

- 예제

```
DOCRUZER> explain scenario tbl;
scenario 'tbl'
{
link = h1
volume = myvol
table = tbl
field = title (0, "<b>","</b>")
field = body (200, "<b>","</b>")
field = author (0, "<b>","</b>")
field = section
field = source
}
OK.
```

RELOAD SCENARIO;

독크루저 설정 파일(docruzerd.rc)에서 시나리오가 'include' 환경 변수에 의해 별도의 파일로 등록되어 있을 때 해당 파일을 변경한 후 RELOAD SCENARIO 명령어를 이용하여 변경된 시나리오를 바로 반영한다. 독크루저 설정 파일에 직접 시나리오를 정의한 경우, 시나리오 정보를 변경하려면 'docruzerd reload' 명령어를 사용한다.

- 구문

```
RELOAD SCENARIO;
```

- 예제

```
DOCRUZER> reload scenario;
OK.
```

5.3.5. 제품키 정보 명령

독크루저 인터프리터를 이용하여 확인하는 제품키 정보는 독크루저 설정 파일(docruzerd.rc)에 'daemon_product_key'로 등록된 제품 키 정보이다.

SHOW DAEMON PRODUCT KEYS;

등록된 제품 키의 정보를 확인하는 명령어이다. 독크루저 기능을 사용하는 중에 "Error: 'daemon product key is not allowed to use OOO module ~"과 같은 에러 메시지가 발생한 경우, SHOW DAEMON PRODUCT KEYS 명령어를 이용하여 제품 키가 특정 모듈을 사용할 수 있도록 발급되었는지 확인한다.

- 구문

```
show daemon product keys;
```

- 예제

다음 실행 예에서 'host id'는 제품 키 발급에 사용한 서버 호스트 정보이고, 'expire date'는 사용 기한이다. 사용 기한이 '0000- 00-00'으로 나타나는 제품 키는 사용 기한에 제약이 없다. 'CPU'는 최대 CPU 활용 수이다. 독크루저가 설치된 하드웨어의 CPU 수가 많더라도 제품 키에 의해 제한된 수만큼의 CPU 를 활용한다. 'available modules'는 사용 권한이 있는 모듈의 번호를 나타낸다.

```
DOCRUZER> show daemon product keys;
+--+-------------------------------------+-----------------+
-----------+---+-----------------+
|no| daemon product key                  | host id         |
expire date|cpu|available modules|
+--+-------------------------------------+-----------------+
-----------+---+-----------------+
|0 |3AU13T-SH4R5C-F5TX37B-R179EYB-8SNX52|00-50-8B-F9-4B-34|
0000-00-00 | 8 | 74 76 81 86 87  |
+--+-------------------------------------+-----------------+
-----------+---+-----------------+
```

```
Total 1 key(s).
OK.
```

EXPLAIN DAEMON PRODUCT KEY *index*;

제품 키에 대한 상세 정보를 보여주는 명령어이다.

- 구문

```
EXPLAIN DAEMON PRODUCT KEY index;
```

구성요소	설명
index	제품 키는 최대 2 개까지 등록할 수 있어 *index*의 값은 0 또는 1이다.

- 예제

```
DOCRUZER> explain daemon product key 0;
+--------------------+--------------------------------------+
| daemon product key | 3AU13T-SH4R5C-F5TX37B-R179EYB-8SNX52 |
+--------------------+--------------------------------------+
| host id            | 00-50-8B-F9-4B-34                    |
+--------------------+--------------------------------------+
| issue date         | 2007-04-16                           |
+--------------------+--------------------------------------+
| expire date        | 0000-00-00                           |
+--------------------+--------------------------------------+
| available modules  | 74 76 81 86 87                       |
+--------------------+--------------------------------------+
| model name         | 8CUL                                 |
+--------------------+--------------------------------------+
| cpu                | 8                                    |
+--------------------+--------------------------------------+
| evaluation copy    | no                                   |
+--------------------+--------------------------------------+
| server type        | Search & Indexing                    |
+--------------------+--------------------------------------+
OK.

DOCRUZER>
```

예제에서 'issue date'는 제품 키의 발급일이다. 'model name'은 '설치 준비에 의해 발급된 제품 규격 정보이다. 예를 들어 '8CUL'은 8 개 CPU를 활용하며 볼륨 제한이 없는(UnLimited) 제품이다. 'evaluation copy'는 평가판 여부(yes/no)를 보여주고, 'server type'은 제품 키가 검색용인지 색인용인지, 색인/검색 모두 가능한지를 나타낸다.

5.4. 독크루저 모듈

독크루저 모듈을 사용하기 위해서는 공통적으로 아래와 같이 데몬 모듈(*.mod)이 위치한 디렉토리와 해당 모듈의 사용 권한이 있는 제품키를 docruzerd.rc에 등록해야 한다.

daemon_data_location으로 설정된 디렉터리에는 데몬 모듈(*.mod) 뿐만 아니라 각 모듈의 사전 및 설정파일도 위치하여야 한다. 그리고 DOCRUZERD에서 모듈을 컴파일 하고 데몬을 구동하면 자동으로 해당 모듈이 활성화 된다.

- docruzerd.rc 등록 예)

```
daemon_product_key = LIKH4DB-KGTKO9-42JES4B-0VOD9TB-B83V85
daemon_data_location = ../data
```

5.4.1. 추천검색어 (KEYWORD RECOMMENDATION)

검색어 입력 시 해당 검색어와 연관된 다른 검색어들을 추천해 주는 기능이 필요한 경우 독크루저의 추천검색어 기능을 사용할 수 있다. 추천검색어 기능은 검색어 입력 시 지정된 개수의 추천검색어를 제시해 준다.

추천검색어 사전은 서로 연관성이 있는 검색어의 목록으로, 검색어 입력 전-후의 검색 패턴을 분석하여 추출한다. 로그마이너를 사용할 경우 검색 로그를 이용하여 주기적으로 갱신되는 추천검색어 목록을 사전 데이터로 사용할 수 있으며, 별도의 추천검색어 목록이 있을 경우 이를 사용할 수도 있다.

추천검색어 모듈 구성은 다음과 같다.

구성 부분	설명	비고
kql076.mod	실질적인 요청을 처리하는 데몬 모듈	필수
kre-conf.*domain*.rc	MOD_KRE의 환경설정 파일	필수
kre-data.*domain*.compiled	컴파일 된 데이터 파일	필수

구성 부분	설명	비고
kre-user.*domain*.txt	추천검색어 주 사용자 사전	필수
kre-usr2.*domain*.txt	추천검색어 보조 사용자 사전	
kre-stop.*domain*.txt	추천검색어 금칙어 사전	

추천검색어 모듈 설정파일은 kre-conf.domain.rc이며 설정값은 다음과 같다.

파라미터	설명
use_keyword_case	입력문이 영문 일 경우 대/소문자를 구분할 것인지 설정. (기본값=0). 예) `use_keyword_case = 1`
use_lower_keyword	영문 대/소문자 구분 없이 사용할 것인지 설정. 예) `use_lower_keyword = 1`

- kre-conf.0.rc 설정 예

```
use_keyword_case = 0
use_lower_keyword = 1
```

추천검색어 사전은 주 사용자 사전(kre-user.domain.txt), 보조 사용자 사전(kre-usr2.domain.txt), 금칙어 사전(kre-stop.domain.txt)으로 구성된다. domain은 도메인 번호로 0~49까지 지정할 수 있다. 도메인 번호를 지정하여 여러 도메인의 추천검색어 사전을 선택적으로 사용할 수 있다. 사전이 분리되어 있기 때문에 활용 목적에 따라 별도 관리가 가능하다.

- 사용 예
  - 주 사용자 사전: 관리기를 통한 수동 관리
  - 보조 사용자 사전: 로그마이너를 통한 자동 관리

- 사전 파일의 예

```
--------------data/ kre-user.0.txt -----------------
핸드폰, 이효리, 초코렛폰, 테레비
"동급생, 하급생", 동급생, 하급생, "동급셍, 하급셍", 동급셍, 하급셍
"록맨, 메가맨", 록맨 메가맨, 록멘 메가멘, 록멘 메가맨, 록맨 매가맨,
록맨 매가멘
"록맨, 메가맨 (31)", "Rock Man, Mega Man"
061JOB, 061job
1492마일스, 1492MILES, 일사구이마일스, 1492마일즈, 일사구이마일즈
16진수 , hexadecimal
24h 크림, 24시간 크림
2단계 숙성 , 2단게 숙성 , 2단개 숙성
2단계 커미트 , 2PC, two-phase commit
-------------- data/kre-stop.0.txt -----------------
테레비
```

위 사전 파일을 사용하는 경우 검색어 '핸드폰'에 대해 추천검색어를 검색하면 '이효리'와 '초코렛폰'이 검색된다. 마찬가지로 '2PC'에 대한 추천검색어는 '2 단계 커미트', 'two-phase commit'가 검색된다. 사전 데이터에 등록된 순서대로 추천어 검색 결과에 나온다. '핸드폰'으로 검색한 경우 '테레비'가 검색되지 않는 이유는 '테레비'가 kre-stop.0.txt에 등록되어있기 때문이다.

추천검색어 사전들은 docruzerd.rc에 명시된 daemon_data_location에 위치해야 하며 compile이 완료되면 같은 위치에 kre-data.0.compiled와 같은 파일이 생성된다. 컴파일한 사전을 독크루저에 바로 적용시키기 위해서는 reload 명령을 내려주면 된다.

```
DOCRUZER> compile module 76 0;
OK.

DOCRUZER> reload module 76 0;
OK.
```

추천검색어 기능은 독크루저 API RecommendKeyword()를 사용한다. API에 대한 설명은 'DOCRUZER API Reference Manual'을 참고한다.

5.4.2. 오타 교정 (SPELL CHECK)

사용자가 입력한 검색어는 다양한 요인에 의해 잘못된 형태로 입력되는 경우가 있다. 맞춤법이 틀렸거나, 영문 입력 모드에서 한글의 내용을 입력하는 경우 등에 의해 잘못된 검색어로 원치 않은 검색을 수행하는 경우가 발생한다.

오타 교정 모듈은 사용자가 잘못 입력한 검색어를 교정하여 정확한 검색 결과를 얻을 수 있도록 지원하는 모듈이다. 오타교정 모듈은 오타교정 사전을 이용하여 검색어의 오타 여부를 결정한다. 입력된 검색어가 오타교정 사전의 오타 정보에 등록되어 있을 경우 입력된 검색어 대신 사전에 등록된 정답어를 검색어로 전달한다.

오타 교정 모듈 구성은 다음과 같다.

구성 부분	설명	비고
kql075.mod	실질적인 요청을 처리하는 데몬 모듈	필수
spc-conf.rc	MOD_SPC의 환경설정 파일	
spc-data.compiled	컴파일 된 데이터 파일	필수
spc-syst.dat	오타 교정 모듈 시스템 사전	필수
spc-user.txt	오타 교정 모듈 사용자 사전	
spc-stop.txt	오타 교정 모듈 금칙어 사전	

오타 교정 모듈은 오타와 정답 정보 사전을 사용한다. 사전 종류는 사용자 사전(spc-user.txt), 금칙어 사전(spc-stop. txt), 시스템 사전(spc-syst.dat)의 세 가지가 있다. 시스템 사전은 독크루저에 기본적으로 포함되어 있으며 사용자에 의한 임의의 수정은 불가능하다. 사용자 사전과 금칙어 사전은 필요에 의해 작성하여 사용한다.

오타 교정 모듈 설정파일은 spc-conf.rc이며 설정값은 다음과 같다.

파라미터	설명
spc_opt_term_validation_level	영->한 키스트로크 변환 시 교정 방법을 설정한다. (기본값=0). ● 0: 한글 자모 구성 규칙에 맞으면 교정 ● 1: 사용자사전이나 시스템사전에 올바른 철자로 등록되어 있는 경우 교정

파라미터	설명
	예) `spc_opt_term_validation_level = 1`
spc_opt_min_term_length	알파벳으로만 입력된 경우 유사 단어를 제시하기 위한 최소 문자수를 설정한다. (기본값=4). 설정된 값보다 적은 문자수를 가진 입력의 경우 유사단어 제시 알고리즘은 수행하지 않음. 예) `spc_opt_min_term_length = 4`
spc_opt_use_user_dic	사용자사전의 사용 여부를 설정한다. (기본값=1). 예) `spc_opt_use_user_dic = 0`
spc_opt_use_system_dic	시스템사전의 사용 여부를 설정한다. (기본값=1). 예) `spc_opt_use_system_dic = 0`
spc_opt_use_kscv_e2k	영->한 키스트로크 변환의 사용 여부를 설정한다. (기본값=1). 예) `spc_opt_use_kscv_e2k = 0`
spc_opt_use_kscv_k2e	한->영 키스트로크 변환의 사용 여부를 설정한다. (기본값=1). 예) `spc_opt_use_kscv_k2e = 0`

파라미터	설명
spc_opt_use_typo_e2e	입력이 알파벳으로만 구성된 경우 유사단어 제시 기능의 사용 여부를 설정한다. (기본값=1). 예) `spc_opt_use_typo_e2e = 0`
spc_opt_use_stopword	금칙어 사전의 적용 여부를 설정한다. (기본값=1). 예) `spc_opt_use_typo_e2e = 0`

spc-user.txt는 사용자가 편집할 수 있는 사전으로 기존 시스템 사전의 오타 교정 정보에 없는 내용을 추가하거나 변경하여 적용하고 싶은 경우 등록하여 사용한다. 사용자 사전에 등록된 오타 변환 규칙은 최고의 우선순위를 가지고 처리된다. 시스템 사전(spc-syst.dat)과 충돌이 발생할 경우 우선 적용된다.

파일 형식은 다음과 같다.

```
correct_word [ : incorrect_word [, incorrect_word,…]]
```

*correct_word*는 입력한 오타에 대한 정답이고, *incorrect_word*는 콤마(,)로 나열되며 정답으로 수정될 오타 항목이다.

사전 예제는 다음과 같다.

```
--------- spc-user.0.txt ---------
daemon: deamon
simple: simpel
김치찌개: 김치치개, 김치찌게
```

spc-stop. txt는 오타 교정된 검색어 중 검색에서 제외될 금칙어들을 등록한 사전이다. 금칙어 사전에는 한 줄에 하나의 금칙어를 등록한다.

spc-stop. txt의 예는 다음과 같다.

```
--------- spc-stop.txt ---------
섹스
야설
```

```
야동
…
```

spc-user.txt나 spc-stop. txt를 사용자가 새로이 작성하거나, 수정하여 사용할 경우에는 데몬에서 사전을 컴파일하여야 spc-data.compiled에 수정된 내용이 반영된다.

위의 사전 파일을 데몬에서 컴파일하는 방법은 아래와 같다.

```
DOCRUZER> compile module 75;
OK.

DOCRUZER> reload module 75;
OK.
```

컴파일이 성공적으로 완료되면 daemon_data_location에 spc-data.compiled가 생성된다.

실제 오타 교정 기능은 독크루저 API SpellCheck()를 사용한다.

API에 대한 설명은 'DOCRUZER API Reference Manual'을 참고한다.

5.4.3. 카테고리 랭킹 (CATEGORY RANKING)

컴파일이 성공적으로 완료되면 daemon_data_location에 spc-data.compiled가 생성된다.

카테고리 랭킹을 사용하기 위해서는 레코드 중 카테고리 필드의 타입은 STRING PARA 로 선언하고 인덱스 생성, 검색 시 정렬 옵션으로 $CATEGORYFIELD 사용해야 한다. 카테고리 필드는 1 개 이상을 사용할 수 있고 $CATEGORYFIELD에는 총 10 개의 도메인 혹은 10 개의 카테고리 필드를 나열할 수 있다.

카테고리 랭킹 모듈 구성은 다음과 같다.

구성 부분	설명	비고
kql086.mod	실질적인 요청을 처리하는 데몬 모듈	필수
ctr-conf.domain.rc	MOD_CTR의 환경설정 파일	필수
ctr-data.domain.compiled	컴파일된 데이터 파일	필수
ctr-user.domain.txt	사전 파일	필수

카테고리 랭킹 모듈 설정파일은 ctr-conf.domain.rc 이며 설정값은 다음과 같다.

파라미터	설명
synonym_domain_no	동의어 확장을 위한 데몬 동의어 모듈 번호. 예) `synonym_domain_no = 0`

카테고리 랭킹을 사용하기 위해서는 검색어별로 카테고리를 등록해 놓은 카테고리 사전이 필요하다. 카테고리 사전은 독크루저의 data 디렉터리에 위치하며, ctr-user.domain.txt 형식의 이름을 가진다.

사전 형식은 다음과 같다.

```
키워드 : 카테고리명, … : 제외카테고리명, …
```

다음 두가지 예는 동일한 효과를 가진다.

```
핸드폰: SKT, KTF

핸드폰: SKT
핸드폰: KTF
```

':' 이후 다음과 같이 제외카테고리를 등록할 수 있다.

```
아이폰 : KTF :  케이스, 줄
```

'아이폰'이라는 검색어에 대해 'KTF'는 상위로 랭크시키고 '케이스', '줄' 카테고리에 속한 레코드는 결과에서 제거한다.

하나의 검색어는 콤마(,)로 구분된 다수의 카테고리에 등록될 수 있다. 위 예에서 '아이폰' 검색어에는 'KTF' 카테고리와 '케이스 제외카테고리가 등록되었고, '모토로이' 검색어는 'SKT', '케이스' 카테고리가 등록되었고 제외카테고리는 없다.

검색 서비스 중에 변경된 카테고리 랭킹 사전을 반영하는 과정은 다음과 같다.

```
DOCRUZER> compile module 86 0;
OK.

DOCRUZER> reload module 86 0;
OK.
```

컴파일을 성공적으로 마치면 ctr-data.0.compiled가 생성된다.

카테고리 랭킹을 이용한 검색 사용 예제는 다음과 같다.

필드명	타입	인덱스명	필드명
fd1	text null	idx1	fd1
fd2	text null	idx2	fd2
fd3	string para null	idx3	fd3

다음 예제에서는 선언된 필드 중 fd3 필드가 정렬에 사용할 카테고리 값을 가지고 있다. fd3은 카테고리 랭킹에 의한 정렬 기준이 되는 필드로 STRING PARA로 선언되어있다.

```
---------data/ctr-user.0.txt----------
핸드폰:SK텔레콤
핸드폰:KTF
---------data/ctr-user.1.txt----------
핸드폰:KTF
핸드폰:KTF기변
핸드폰:악세사리, 엑세서리

kql/myvol>> retrieve * from tbl where idx1='핸드폰' allword absolute;
---------- 0 of total 7 (ROWID 0, REL 268439039) ---------
fd1 (13): 핸드폰 핸드폰
fd2 (11): 휴대폰/통신
fd3 (15): 휴대폰 주변기기
---------- 1 of total 7 (ROWID 1, REL 268438783) ---------
fd1 (13): 핸드폰 목걸이
fd2 (11): 휴대폰/통신
fd3 (15): 휴대폰 주변기기
---------- 2 of total 7 (ROWID 2, REL 268438783) ---------
fd1 (9): 핸드폰 줄
fd2 (11): 시계/쥬얼리
fd3 (8): 악세사리
---------- 3 of total 7 (ROWID 4, REL 268438783) ---------
fd1 (13): 핸드폰 거치대
fd2 (15): 자동차/악세사리
fd3 (13): 핸드폰 거치대
---------- 4 of total 7 (ROWID 5, REL 268438782) ---------
fd1 (14): SKY-YYY 핸드폰
fd2 (11): 휴대폰/통신
fd3 (8): SK텔레콤
```

```
---------- 5 of total 7 (ROWID 3, REL 268438781) ---------
fd1 (21): 애니콜 SPH-XXX 핸드폰
fd2 (11): 휴대폰/통신
fd3 (3): KTF
---------- 6 of total 7 (ROWID 7, REL 268438781) ---------
fd1 (19): SCH-333 삼성 핸드폰
fd2 (11): 휴대폰/기변
fd3 (8): SK텔레콤
Total 7 records.
OK
================================================================
카테고리 필드를 fd3이라 지정하고, 카테고리 랭킹 도메인은 0으로 지정한 후,
'핸드폰'으로 검색된 문서들을 카테고리 랭킹으로 정렬한다. 즉, 카테고리 필드
fd3의 값이 0번 도메인에서 카테고리 검색 키워드 '핸드폰'이 속한 카테고리인
'SK텔레콩', 'KTF'인 문서들이 상위로 올린다.
검색어와 카테고리 검색 키워드는 서로 달라도 된다.
================================================================
kql/myvol>> retrieve * from tbl where idx1='핸드폰' allword
order by $CATEGORYFIELD(fd3(0), '핸드폰') absolute;
---------- 0 of total 7 (ROWID 3, REL 268438781) ---------
fd1 (21): 애니콜 SPH-XXX 핸드폰
fd2 (11): 휴대폰/통신
fd3 (3): KTF
---------- 1 of total 7 (ROWID 5, REL 268438782) ---------
fd1 (14): SKY-YYY 핸드폰
fd2 (11): 휴대폰/통신
fd3 (8): SK텔레콤
---------- 2 of total 7 (ROWID 7, REL 268438781) ---------
fd1 (19): SCH-333 삼성 핸드폰
fd2 (11): 휴대폰/기변
fd3 (8): SK텔레콤
---------- 3 of total 7 (ROWID 0, REL 268439039) ---------
fd1 (13): 핸드폰 핸드폰
fd2 (11): 휴대폰/통신
fd3 (15): 휴대폰 주변기기
---------- 4 of total 7 (ROWID 1, REL 268438783) ---------
fd1 (13): 핸드폰 목걸이
fd2 (11): 휴대폰/통신
fd3 (15): 휴대폰 주변기기
---------- 5 of total 7 (ROWID 2, REL 268438783) ---------
```

```
fd1 (9): 핸드폰 줄
fd2 (11): 시계/쥬얼
fd3 (8): 악세사리
---------- 6 of total 7 (ROWID 4, REL 268438783) ---------
fd1 (13): 핸드폰 거치대
fd2 (15): 자동차/악세사리
fd3 (13): 핸드폰 거치대
Total 7 records.
OK

kql/myvol>> retrieve * from tbl where idx1='핸드폰' allword
order by $CATEGORYFIELD(fd3(1), '핸드폰') absolute;
---------- 0 of total 7 (ROWID 2, REL 268438783) ---------
fd1 (9): 핸드폰 줄
fd2 (11): 시계/쥬얼리
fd3 (8): 악세사리
---------- 1 of total 7 (ROWID 3, REL 268438781) ---------
fd1 (21): 애니콜 SPH-XXX 핸드폰
fd2 (11): 휴대폰/통신
fd3 (3): KTF
---------- 2 of total 7 (ROWID 0, REL 268439039) ---------
fd1 (13): 핸드폰 핸드폰
fd2 (11): 휴대폰/통신
fd3 (15): 휴대폰 주변기기
---------- 3 of total 7 (ROWID 1, REL 268438783) ---------
fd1 (13): 핸드폰 목걸이
fd2 (11): 휴대폰/통신
fd3 (15): 휴대폰 주변기기
---------- 4 of total 7 (ROWID 4, REL 268438783) ---------
fd1 (13): 핸드폰 거치대
fd2 (15): 자동차/악세사리
fd3 (13): 핸드폰 거치대
---------- 5 of total 7 (ROWID 5, REL 268438782) ---------
fd1 (14): SKY-YYY 핸드폰
fd2 (11): 휴대폰/통신
fd3 (8): SK텔레콤
---------- 6 of total 7 (ROWID 7, REL 268438781) ---------
fd1 (19): SCH-333 삼성 핸드폰
fd2 (11): 휴대폰/기변
fd3 (8): SK텔레콤
```

```
Total 7 records.
OK
================================================================
fd3의 값이 도메인 1에서 '핸드폰'이 속한 카테고리인 'KTF', 'KTF기변',
'액세서리', '엑세서리'인 문서를 가장 상위로, 그 다음에는 fd3의 값이
도메인 0에서 '핸드폰'이 속한 카테고리인 'SK텔레콤', 'KTF'인 문서를
위치시킨다.
================================================================
kql/myvol>> retrieve * from tbl where idx1='핸드폰' allword
order by $CATEGORYFIELD(fd3(1) fd3(0), '핸드폰') absolute;
---------- 0 of total 7 (ROWID 3, REL 268438781) ---------
fd1 (21): 애니콜 SPH-XXX 핸드폰
fd2 (11): 휴대폰/통신
fd3 (3): KTF
---------- 1 of total 7 (ROWID 2, REL 268438783) ---------
fd1 (9): 핸드폰 줄
fd2 (11): 시계/쥬얼리
fd3 (8): 악세사리
---------- 2 of total 7 (ROWID 5, REL 268438782) ---------
fd1 (14): SKY-YYY 핸드폰
fd2 (11): 휴대폰/통신
fd3 (8): SK텔레콤
---------- 3 of total 7 (ROWID 7, REL 268438781) ---------
fd1 (19): SCH-333 삼성 핸드폰
fd2 (11): 휴대폰/기변
fd3 (8): SK텔레콤
---------- 4 of total 7 (ROWID 0, REL 268439039) ---------
fd1 (13): 핸드폰 핸드폰
fd2 (11): 휴대폰/통신
fd3 (15): 휴대폰 주변기기
---------- 5 of total 7 (ROWID 1, REL 268438783) ---------
fd1 (13): 핸드폰 목걸이
fd2 (11): 휴대폰/통신
fd3 (15): 휴대폰 주변기기
---------- 6 of total 7 (ROWID 4, REL 268438783) ---------
fd1 (13): 핸드폰 거치대
fd2 (15): 자동차/악세사리
fd3 (13): 핸드폰 거치대
Total 7 records.
OK
```

제외카테고리로 등록하면 해당 카테고리의 데이터를 검색 결과에서 아예 제외한다.

다음은 제외카테고리의 검색 사용 예이다.

```
---------data/ctr-user.0.txt----------
아이폰 : KTF : 케이스
모토로이 : SKT, 케이스

kql/myvol>> select * from tbl;
---------- 0 of total 8 (ROWID 0) ---------
fd (15): 모토로이 핸드폰
ct1 (3): LGT
---------- 1 of total 8 (ROWID 1) ---------
fd (8): 모토로이
ct1 (3): SKT
---------- 2 of total 8 (ROWID 2) ---------
fd (13): 모토로이 블랙
ct1 (3): SKT
---------- 3 of total 8 (ROWID 3) ---------
fd (20): 모토로이 투명 케이스
ct1 (6): 케이스
---------- 4 of total 8 (ROWID 4) ---------
fd (18): 아이폰보다 나은 폰
ct1 (3): LGT
---------- 5 of total 8 (ROWID 5) ---------
fd (16): 아이폰 즉시 개통
ct1 (3): KTF
---------- 6 of total 8 (ROWID 6) ---------
fd (18): 아이폰 투명 케이스
ct1 (6): 케이스
---------- 7 of total 8 (ROWID 7) ---------
fd (11): 아이폰 블랙
ct1 (3): KTF
Total 8 records.
OK
================================================================
네가티브 카테고리를 설정하지 않은 모토로이는 SKT, 케이스 카테고리를
함께 상위로 랭크한다.
================================================================
kql/myvol>> retrieve * from tbl where f='모토로이' allword
order by $CATEGORYFIELD(ct
```

```
1(0), '모토로이') absolute;
---------- 0 of total 4 (ROWID 1, REL 553653758) ---------
fd (8): <<모토로이>>
ct1 (3): SKT
---------- 1 of total 4 (ROWID 2, REL 553653758) ---------
fd (13): <<모토로이>> 블랙
ct1 (3): SKT
---------- 2 of total 4 (ROWID 3, REL 553653758) ---------
fd (20): <<모토로이>> 투명 케이스
ct1 (6): 케이스
---------- 3 of total 4 (ROWID 0, REL 553653758) ---------
fd (15): <<모토로이>> 핸드폰
ct1 (3): LGT
Total 4 records.
OK
================================================================
케이스를 네가티브 카테고리로 설정하여 둔 아이폰은 KTF 카테고리를 상위로
랭크하며 케이스 카테고리를 검색 결과에서 제외한다.
================================================================
kql/myvol>> retrieve * from tbl where fd='아이폰' allword
order by $CATEGORYFIELD(ct1(
0), '아이폰') absolute;
---------- 0 of total 3 (ROWID 5, REL 553653758) ---------
fd (16): <<아이폰>> 즉시 개통
ct1 (3): KTF
---------- 1 of total 3 (ROWID 7, REL 553653758) ---------
fd (11): <<아이폰>> 블랙
ct1 (3): KTF
---------- 2 of total 3 (ROWID 4, REL 553654014) ---------
fd (18): <<아이폰>>보다 나은 <<폰>>
ct1 (3): LGT
Total 3 records.
OK
```

카테고리 랭킹은 다음과 같이 다른 정렬 옵션과 함께 사용할 수 있으며 여러 개의 정렬 옵션이 조합되어 있을 때 조합의 적용 순서에 영향을 주지 않는다.

- 예제

```
order by $CATEGORY(fd3(0), '핸드폰'), $MATCHFIELD(fd1, fd2)
DESC, $RELEVANCE DESC
order by $CATEGORYFIELD(fd3(0), '핸드폰') group by fd3;
```

5.4.4. 글로서리 앵커링 (GLOSSARY ANCHORING)

글로서리는 어려운 말에 대한 용어집, 어휘사전을 말한다. 글로서리 앵커링 모듈은 검색결과 페이지에 인물명, 기업명, 용어 등 특정 단어에 링크 표시를 자동생성 하여 페이지를 읽는 중 용어 관련 정보에 대한 별도 검색 없이 곧바로 확장 정보에 대한 조회를 가능하게 한다.

글로서리 앵커링 모듈 구성은 다음과 같다.

구성 부분	설명	비고
kql087.mod	실질적인 요청을 처리하는 데몬 모듈	필수
act-conf.*domain*.rc	MOD_CTR의 환경설정 파일	필수
act-data.*domain*.compiled	컴파일 된 데이터 파일	필수
act-user.*domain*.txt	글로서리 앵커링 사전 파일	필수

글로서리 앵커리 모듈 설정 파일은 act-conf.*domain*.rc이며 설정값은 다음과 같다.

파라미터	설명
MinClauseLetterCount	사전에서 찾을 수 있는 최소 단어 길이 설정. (기본값=3). 예) `MinClauseLetterCount = 2`
NormalCharactor	사전과 본문에 허용할 수 있는 특수 문자 설정. 예) `NormalCharactor = &`
GlossaryData	사전정보에 대한 제반 정보를 지정하는 파라메터 값. 하나 이상의 카테고리의 사전을 제공하는 경우 이 파라미터를 계속 추가하면 됨. 파라미터 옆의 값은 해당 사전을 표현하는 사전명으로, 중복될 수 없다.

파라미터	설명
	예) `GlossaryData word { … }`
skip_html_tag	1로 설정하면 입력 텍스트가 HTML문서인 경우에 html tag를 앵커링 대상에서 제외. (기본값=0). 예) `skip_html_tag = 1`

GlossaryData 설정값은 다음과 같다.

파라미터	설명
record_start_tag	하나의 글로서리 설명이 시작되는 태그를 지정. 예) `record_start_tag = <CATEGORY>`
data_location	사전이 위치한 경로와 확장자를 지정. 지정된 디렉터리의 하위 디렉터리도 모두 포함. 하나 이상 지정 가능. 예) `data_location = /usr2/docruzer/data *.data *.data1`
key_start_tag = term_start_tag	키워드가 시작되는 태그를 지정. 예) `key_start_tag = <WORD_SEAR>`
whitespace_sensitive	글로서리를 검색할 때 공백을 포함하여 문자열 비교를 할 것인가를 설정. 예)

파라미터	설명
	`whitespace_sensitive = Y`
case_sensitive	영문 글로서리의 경우 대소문자를 구분하여 문자열 비교를 할 것인가를 설정. 예) `case_sensitive = Y`

설정 파일의 작성 예는 다음과 같다.

- act-conf.0.rc 설정 예

```
MinClauseLetterCount = 2
;사전을 찾을 최소 글자 개수 ;default:3 (한글 3글자, 영어 3character
등...)

NormalCharacter = &
NormalCharacter = '
NormalCharacter = "
NormalCharacter = -
NormalCharacter = (
NormalCharacter = )
;여기에 설정된 문자는 특수문자로 취급되지 않고 일반 문자로 취급되어
;사전을 찾을때 사용된다.
;생략되면 아스키 문자 중 숫자, 알파벳 대소문자를 제외한 다른 특수문자는
;모두 특수문자로 취급하여 단어를 이 특수문자를 기준으로 자른다

GlossaryData word
{
  record_start_tag = <CATEGORY>
  data_location = /tmp/bb *.txt
  key_start_tag = <WORD_SEAR>
  whitespace_sensitive = N
  case_sensitive = Y
}
```

글로서리 앵커링 관련 docruzerd.rc 설정값은 다음과 같다.

파라미터	설명
max_module_buffer_size_act	MOD_ACT의 출력버퍼의 크기를 지정.
daemon_module_domain _count_act	앵커링 모듈의 최대 도메인 수를 지정. (기본값=1).
daemon_module_session _count_act	앵커링 모듈의 세션 개수를 지정. (기본값=4).

글로서리 앵커링 사전파일 예는 다음과 같다.

- act-user.0.txt 사전 파일 예

```
<CATEGORY>WORD
<IDEN_CODE>863
<WORD_SEAR>자동차
<WORD_BODY>원동기를 장치하여 그 동력으로 바퀴를 굴려서 철길이나 가설된
선에 의하지 아니하고 땅 위를 움직이도록 만든 차. 승용 자동차, 승합자동차,
화물 자동차, 특수 자동차 및 이륜자동차가 있다.
```

검색 서비스 중에 변경된 글로서리 앵커링 사전을 반영하는 과정은 다음과 같다.

```
DOCRUZER> compile module 87 0;
OK.

DOCRUZER> reload module 87 0;
OK.
```

컴파일을 성공적으로 마치면 act-data.0.copiled 파일이 생성된다.

글로서리 앵커링을 사용하기 위해서 독크루저 API AnchorText()를 이용한다. API에 대한 설명은 'DOCRUZER API Reference Manual'을 참고한다.

5.4.5. 인기검색어 (POPULAR KEYWORD)

인기검색어는 독크루저를 이용한 검색 서비스 제공 시 검색어 순위 목록을 제시하는 기능을 제공한다. 인기검색어는 로그마이너를 이용하여 자동으로 생성하여 적용하거나 검색 서비스 관리자가 서비스에 적합한 인기검색어를 생성하여 적용한다. 자동, 수동으로 생성되는 인기검색어는 인기검색어 목록 생성 방법만 다르며 동일한 방법에 의해 서비스에 반영된다.

인기검색어 모듈 구성은 다음과 같다.

구성 부분	설명	비고
kql081.mod	실질적인 요청을 처리하는 데몬 모듈	필수
ppk-conf.domain.rc	MOD-PPK의 환경설정 파일	
ppk-data.domain.compiled	컴파일 된 데이터 파일	필수
ppk-user.domain.txt	인기검색어 주 사용자 사전	
ppk-usr2.domain.txt	인기검색어 보조 사용자 사전	
ppk-stop.domain.txt	인기검색어 금칙어 사전	

인기검색어 모듈 설정파일은 ppk-conf.domain.rc이며 설정값은 다음과 같다.

파라미터	설명
max_no_of_user_data	인기검색어 최대 출력 개수 설정. 기본값은 ppk-user.domain.txt에 저장된 인기검색어 목록을 모두 사용. 예) max_no_of_user_data = 5

max_no_of_user_data 에 설정된 개수만큼 ppk-user.domain.txt에서 인기검색어가 반환되며, 결과물의 개수가 부족한 경우 ppk-usr2.domain.txt에 저장된 인기검색어 목록을 추가적으로 사용한다.

인기검색어 사전파일 예는 다음과 같다.

```
-------------- ppk-user.0.txt -----------------
DOCRUZER
코난테크놀로지
JSP
대한민국
서울
-------------- ppk-stop.0.txt -----------------
음란물
```

검색 서비스 중에 변경된 인기검색어 사전을 반영하는 과정은 다음과 같다.

```
DOCRUZER> compile module 81 0;
OK.
```

```
DOCRUZER> reload module 81 0;
OK
```

위 예에서는 colpile 명령어가 성공적으로 종료하면 ppk-data.0.compiled 파일이 생성된다. reload 명령어에 의해 해당 사전이 시스템에 적용된다. 인기검색어의 반영 주기를 짧게 하고 싶은 경우 위 과정의 수행 주기를 짧게 하면 된다.

검색 시스템으로 인기검색어를 구축할 때는 독크루저 API GetPopularKeyword2()를 사용한다. API에 대한 설명은 'DOCRUZER API Reference Manual'을 참고한다.

5.4.6. 실시간 인기검색어 (REAL TIME POPULAR KEYWORD)

기존의 인기검색어 모듈(mod-ppk)은 검색 로그 파일로부터 DLA 분석을 거쳐 빈도수가 높은 키워드를 추출하고 관리기를 통해 키워드를 정제한 후 독크루저 클라이언트 API를 통해 외부에 출력한다.

이에 비해 실시간 인기검색어 모듈(mod-rpk)은 인기 검색어 모듈(mod-ppk)의 기능에 실시간으로 검색 요청의 검색어들을 저장하고 있다가 클라이언트 API 를 통해 곧바로 외부에 출력하는 기능이 추가되었다.

실시간 인기검색어 모듈의 구성 요소는 다음과 같다. rpk-conf.<domain_no>.rc 파일은 필수이며, 'entry_size' 항목이 반드시 기입되어 있어야 한다.

구성요소	설명
kql202.mod	실질적인 요청을 처리하는 데몬 모듈. 필수 요소
rpk-conf.<domain_no>.rc	mod-rpk의 환경설정 파일. 필수 요소
rpk-data.<domain_no>.compiled	컴파일된 메모리 이미지 데이터 파일. 필수 요소
rpk-user.<domain_no>.txt	사용자 사전 텍스트 파일. (사용하지 않을 경우 빈파일 형태로라도 있어야 한다.) 필수 요소
rpk-stop.<domain_no>.txt	불용어 사전 텍스트 파일. 선택 요소

환경설정 파일의 작성 예는 다음과 같다.

rpk-conf.0.rc

```
entry_size = 10000
use_refresh_interval = 1
refresh_interval = 60
```

다음은 환경설정 파일의 설정값에 대한 설명이다.

파라미터	설명
entry_size	반드시 있어야 하는 값으로 인기검색어를 추출하기 위한 검색어 저장 버퍼의 크기이다.
use_refresh_interval	● 0: 인기검색어 목록을 한번 가져가면 이전에 쌓은 검색어 목록 데이터가 삭제된다. (기본값) ● 1: 일정 주기로 인기검색어 목록을 가져온다.
refresh_interval	실시간 인기검색어 목록을 재작성 하는 주기이며 단위는 second(초)이다. 이 주기마다 인기검색어 목록을 새로 만들고 이전 순위와 비교하여 변동된 순위를 알 수 있다.

실시간 인기검색어의 사전파일 예는 다음과 같다. 첫줄부터 인기순으로 나열한다.

```
-------- rpk-user.0.txt--------------
코난
독크루저
실시간 인기 검색어
…
```

실시간 검색어를 사용하기 위해서는 시나리오에 사용할 실시간검색 도메인을 등록한다.

```
real_time_popular_keyword_domain_no = domain_number
```

시나리오 설정 예는 다음과 같다.

```
scenario tbl
{
  link = h1
  volume = myvol
  table = tbl

  real_time_popular_keyword_domain_no = 0
```

```
  field = fd
  ....
}
```

검색 서비스 중에 변경된 실시간 검색어 사전을 반영하는 과정은 다음과 같다. compile 한 사전을 독크루저에 바로 적용 시키기 위해서는 reload 명령을 내려주면 된다.

```
DOCRUZER> compile module 202 0;
OK.

DOCRUZER> reload module 202 0;
OK.
```

서비스 제공 시 독크루저 API GetRealTimePopularKeyword()를 이용하여 실시간 인기검색어 목록을 가져온다.

API에 대한 설명은 'DOCRUZER API Reference Manual'을 참고한다.

5.4.7. 검색어 자동완성 (AUTOMATIC KEYWORD COMPLETION)

검색어 자동완성은 사용자가 검색어를 입력할 때 시스템에 등록된 사전을 이용하여 입력된 검색어를 포함한 완성형 검색어를 제시해주는 기능이다.

검색어 자동완성 모듈 구성은 다음과 같다.

구성 부분	설명	비고
kql074.mod	실질적인 요청을 처리하는 데몬 모듈	필수
akc-conf.domain.rc	MOD_AKC의 환경설정 파일	
akc-data.domain.compiled	컴파일 된 데이터 파일	필수
akc-user.domain.txt	검색어 자동완성 주 사용자 사전	필수
akc-usr2.domain.txt	검색어 자동완성 보조 사용자 사전	
akc-stop.domain.txt	검색어 자동완성 금칙어 사전	

검색어 자동완성 모듈 설정파일은 akc-conf.domain.rc이며 설정값은 다음과 같다.

파라미터	설명
use_character_series	한글 풀어쓰기를 사용 할 것인지 설정. (기본값=1). 예) use_character_series = 0

파라미터	설명
use_keyword_case	입력문이 영문일 경우 대소문자를 구분 할 것인지 설정. (기본값=0). 예) use_keyword_case = 1

검색어 자동완성 사전은 주 사용자 사전(akc-user.domain.txt)과 보조 사용자 사전(akc-usr2.domain.txt), 금칙어 사전(akc-stop.domain.txt) 등으로 구성된다. 사용자 사전이 분리되어 있기 때문에 활용 목적에 따라 별도 관리가 가능하다.

- 사용 예
  - 주 사용자 사전: 관리기를 통한 수동 관리
  - 보조 사용자 사전: 로그마이너를 통한 자동 관리

- 사전 형식

```
keyword:[tag1]:[tag2]
```

keyword는 완성형 검색어로 제시될 검색어이다. 콜론(':')으로 구분되어 입력하는 tag1과 tag2는 검색어 자동완성 사용 서비스 시 부가적인 정보를 생성하여 검색 서비스에 추가 기능을 제시할 수 있다. 추가 기능의 예를 들면 검색어 자동완성 후보가 기관명을 가리킬 때 기관 홈페이지 주소를 tag1에 추가하여 해당 검색어와 함께 제시하여, 해당 검색어를 선택하였을 경우 기관 홈페이지로 이동하도록 서비스를 구축하는 것이다.

검색어 자동완성 사전 파일 예는 다음과 같다.

```
-------------- akc-user.0.txt -----------------
DOCRUZER::1000
코난테크놀로지:www.konantechnology.co.kr:
JSP::
대한민국:www.bluehouse.go.kr:
서울:www.seoul.go.kr:
...
-------------- akc-stop.0.txt -----------------
음란물
```

검색 서비스 중에 변경된 검색어 자동완성 사전을 반영하는 과정은 다음과 같다. 74는 인기검색어 모듈 번호이고, 0은 사전의 도메인 번호이다. compile 명령어가 정상적으로 완료되면 ack-data.domain.compiled 파일이 생성된다.

```
DOCRUZER> compile module 74 0;
OK.

DOCRUZER> reload module 74 0;
OK
```

검색어 자동완성은 독크루저 API인 CompleteKeyword2()를 사용하여 서비스 제공 시 가져온다. 해당 API의 활용 방법은 'DOCRUZER API Reference Manual'을 참고한다.

5.4.8. 제외검색어 (EXCLUSIVE KEYWORD)

제외검색어란 입력된 사용자 검색어로 검색된 결과에서 특정 키워드가 포함된 문서를 제외시키는 기능이다.

제외검색어 기능은 사용자가 '핸드폰'이란 검색어로 검색을 하였는데, 결과 집합에 '핸드폰 줄'이 포함된 문서가 많이 검색되어 의도했던 '핸드폰'을 찾기 어려울 때, '핸드폰'이란 검색어에 대해 '줄'을 제외검색어로 등록 해 놓으면 검색 결과에서 '핸드폰 줄'이 들어간 문서를 제외시켜 준다.

'핸드폰 검색'	'핸드폰' 검색, 제외어 '줄'
1. '핸드폰 줄'	1. '삼성 핸드폰'
2. '핸드폰 악세서리 줄'	2. 'LG 핸드폰'
3. '삼성 핸드폰'	
4. 'LG 핸드폰'	

제외 검색어는 소팅절의 'exclude by' 구문으로 작동하며 다음과 같이 사용될 수 있다.

```
select * from t where f = '핸드폰' natural exclude by f(0)
```

검색 결과 집합에서 'fd1' 필드에 '핸드폰'이 있으며, fd1의 색인어들이 0번 도메인 제외검색어 사전에서 '핸드폰'에 대해 등록된 제외검색어를 포함하고 있는 경우, 해당 문서는 검색 결과로 재현되지 않는다. 즉, 문서 중 fd1 필드가 검색어와 제외검색어를 모두 포함하고 있는 문서만 검색 결과로 재현되지 않는다.

제외검색어 모듈 구성은 다음과 같다.

구성 부분	설명	비고
kql095.mod	실질적인 요청을 처리하는 데몬 모듈	필수
xwd-conf.domain.rc	MOD_XWD의 환경설정 파일	필수
xwd-data.domain.compiled	컴파일 된 데이터 파일	필수
xwd-user.domain.txt	검색어 자동완성 주 사용자 사전	필수

제외검색어 사전은 xwd-user.domain.txt의 이름을 가진다. 사전의 입력 형식은 아래와 같다.

```
keyword: exclusive_keyword[, exclusive keyword]*
```

keyword는 검색어이며, exclusive_keyword는 해당 검색어로 검색한 경우 검색 결과로 제시되지 않을 제외 검색어 목록이다. 하나의 검색어에 대해 exclusive_keyword는 한 개 이상 등록할 수 있으며, 한 개 이상일 때 제외어의 구분은 콤마(',')로 한다.

사전 편집 시 다음 사항을 고려하여 서비스에 적합한 사전을 구성하도록 한다.

- 제외어 검색 사전에서 검색어(keyword)는 대소문자 구분, 공백 구분을 하지 않는다.
- 제외어 검색 사전에서 제외검색어(exclustive_keyword)는 대소문자는 구분되지 않으며, 여러 개의 공백은 하나의 공백으로 처리된다. 이 공백을 기준으로 제외어는 'AND' 검색을 하게 된다.

예를 들면, 한글 형태소 분석의 경우 제외검색어 부분이 복합명사이고 제외검색어를 적용할 필드가 텍스트 필드라면 제외검색어 부분은 문서가 어떻게 형태소 분석되어 키워드가 저장될지를 고려하여 편집해야한다. '핸드폰'으로 검색된 문서에서 '액정보호필름'이 포함된 문서를 제외하고자 할 때, 해당 필드에서 '액정보호필름'이 '액정' + '보호' + '필름'으로 형태소 분석이 되어 키워드가 저장된다면, 제외검색어 등록 시에는 다음과 같이 공백으로 구분하여 사전을 편집해야한다.

```
핸드폰: 액정 보호 필름
```

영문의 경우에 만일 텍스트 필드에서 대소문자를 구분하여 키워드가 저장되도록 설정(kql.rc에서 keyword_case = mixed)되어 있다면 제외검색어에서 필드값이 원래 소문자였던 필드값만 제외검색어가 적용된다. 또한 영문을 stemming하여 키워드를 추출하도

록 설정(kql.rc에서 USE_STEM_KEYWORD_EN = y)되어 있다면 제외검색어도 stemming 된 결과로 편집해야한다.

제외검색어 사전 파일 예는 다음과 같다.

```
---------xwd-user.0.txt----------
핸드폰: 악세사리,줄
핸드폰: 엑세서리
핸드폰: 액정 보호 필름
자동차 키:악세사리
car: accessory
car: Navigation
cellphone: PDA
```

검색 서비스 중에 변경된 제외검색어 사전을 반영하는 과정은 다음과 같다.

```
DOCRUZER> compile module 95 0;
OK.

DOCRUZER> reload module 95 0;
OK
```

compile 명령이 성공적으로 종료되면 xwd-user.domain.compiled 파일이 생성된다. 제외어 검색의 적용은 검색 메서드에서 제외어 검색 옵션을 추가해주면 된다.

필드명	타입	인덱스명	필드명
fd1	text null	idx1	fd1
fd2	text null	idx2	fd2
fd3	string para null	idx3	fd3

독크루저 API SubmitQuery()를 이용하여 제외어 검색을 수행할 경우에는 sorting 절에 제외어 옵션을 전달한다.

다음은 제외검색어를 사용한 검색 예이다.

필드명	타입	인덱스명	필드명
fd1	text null	idx1	fd1 fdf2
fd2	text null	idx2	fd3
fd3	text null		

```
kql/myvol>> select * from tbl;
---------- 0 of total 5 (ROWID 0) ---------
fd1 (14): car navigation
fd2 (0):
fd3 (0):
---------- 1 of total 5 (ROWID 1) ---------
fd1 (15): 핸드폰 악세사리
fd2 (8): 악세사리
fd3 (6): 핸드폰
---------- 2 of total 5 (ROWID 2) ---------
fd1 (20): 핸드폰 액정보호 필름
fd2 (12): 액정보호필름
fd3 (0):
---------- 3 of total 5 (ROWID 3) ---------
fd1 (15): 핸드폰 액정보호
fd2 (12): 액정보호필름
fd3 (0):
---------- 4 of total 5 (ROWID 4) ---------
fd1 (13): cellphone PDA
fd2 (0):
fd3 (0):
Total 5 records.
OK

kql/myvol>> select * from tbl where idx1='핸드폰' natural;
---------- 0 of total 3 (ROWID 1) ---------
fd1 (15): 핸드폰 악세사리
fd2 (8): 악세사리
fd3 (6): 핸드폰
---------- 1 of total 3 (ROWID 2) ---------
fd1 (20): 핸드폰 액정보호 필름
fd2 (12): 액정보호필름
fd3 (0):
---------- 2 of total 3 (ROWID 3) ---------
fd1 (15): 핸드폰 액정보호
fd2 (12): 액정보호필름
fd3 (0):
Total 3 records.
OK
```

```
kql/myvol>> select * from tbl where idx1='핸드폰' natural exclude
by fd1(0);
---------- 0 of total 1 (ROWID 3) ---------
fd1 (15): 핸드폰 액정보호
fd2 (12): 액정보호필름
fd3 (0):
Total 1 record.
OK
=======================================================
ROWID 1, 2는 fd1에서 '핸드폰'으로 검색이 되었고 fd1에 해당 제외검색어인
'액세서리', '액정보호필름'도 있으므로 검색 결과에서 제외된다.
=======================================================
kql/myvol>> select * from tbl where idx1='핸드폰' natural exclude
by fd2(0);
---------- 0 of total 3 (ROWID 1) ---------
fd1 (15): 핸드폰 악세사리
fd2 (8): 악세사리
fd3 (6): 핸드폰
---------- 1 of total 3 (ROWID 2) ---------
fd1 (20): 핸드폰 액정보호 필름
fd2 (12): 액정보호필름
fd3 (0):
---------- 2 of total 3 (ROWID 3) ---------
fd1 (15): 핸드폰 액정보호
fd2 (12): 액정보호필름
fd3 (0):
Total 3 records.
OK

ROWID 2의 경우는 비록 fd2에 제외검색어인 '액정보호필름'이 있지만, 사용자
검색어인 '핸드폰'으로 fd2에서 검색된 것이 아니므로 검색 결과에 포함된다.
=======================================================

kql/myvol>> select * from tbl where idx1='car' natural;
---------- 0 of total 1 (ROWID 0) ---------
fd1 (14): car navigation
fd2 (0):
fd3 (0):
Total 1 record.
OK
```

```
=======================================================
ROWID 0은 'car'의 제외검색어 'navigation'이 fd1에서 검색되므로 검색
결과에서 제외된다.
=======================================================
kql/myvol>> select * from tbl where idx1='car' natural exclude
by fd1(0);
Total 0 record.
OK
```

제외어 검색 기능은 검색 메서드의 정렬 옵션에 해당하며 다른 정렬 옵션과 같이 사용할 수 있다. 예를 들면, 검색 옵션에 ~ EXCLUDE BY fd(0) ORDER BY fd ASC WEIGHT BY fd=100;와 같이 쓸 수 있다.

5.4.9. 데몬 동의어 (SYNONYM KEYWORD)

데몬 동의어 기능은 '사용자 사전 - 동의어 (RELOAD USER DIC FOR SYNONYM)'를 사용하는 동의어 검색과는 다른 형태로 동의어 검색을 지원하는 방법이다. 기본적으로는 '동의어 검색 (SYNONYM)'과 같은 검색 결과를 반환하지만, 도메인지원과 기본 동의어 사전 제공, 동의어 검색의 효율성 증대의 기능 향상을 제공한다.

구성 부분	설명	비고
kql082.mod	실질적인 요청을 처리하는 데몬 모듈	필수
syn-conf.domain.rc	MOD_SYN3의 환경설정 파일	필수
syn-data.domain.compiled	컴파일 된 데이터 파일	필수
syn-user.domain.txt	데몬동의어 사용자 사전	
syn-stop.domain.txt	데몬동의어 금칙어 사전	

데몬 동의어 사전은 사용자 사전(syn-user.domain.txt)과 금칙어 사전(syn-stop.domain.txt)이 있고 모든 사전은 생략 가능하다.

데몬 동의어 사용자 사전의 예는 다음과 같다.

```
코난, 코눈, 크난
…
```

검색 서비스 중에 변경된 데몬 동의어 사용자 사전을 반영하는 과정은 다음과 같다.

```
DOCRUZER> compile module 82 0;
OK.
```

```
DOCRUZER> reload module 82 0;
OK.
```

compile 명령어가 성공적으로 종료하면 syn-data.0.compiled 파일이 생성된다. 데몬 동의어를 이용한 동의어 검색을 수행하기 위해서는 검색 시나리오에 synony_domain_no 옵션을 추가해 주어야 한다.

다음 예제는 동의어 검색을 수행할 경우 동의어 도메인 0 번 사전을 이용한다. 독크루저 API SubmitQuery()나 Search()를 사용할 때 검색 쿼리에 SYNONYM을 사용하지 않도록 한다.

```
scenario tbl
{
    link    = h1
    volume  = myvol
    table   = tbl
    field   = fd

    synonym_domain_no = 0
}
```

동의어 사전이 정상적으로 등록되어 있고, 시나리오에 데몬 동의어를 사용할 수 있도록 설정한 후, '동의어 검색 (SYNONYM)' 방법으로 검색을 수행한다.

데몬 동의어 모듈은 kql 프롬프트에서는 인식이 되지 않는다. 만일 kql.rc에 user_dic_for_synonym도 설정하고, 데몬 동의어 모듈에 대한 시나리오 설정 synonym_domain_no도 설정한 후 동의어 검색을 한다면 kql 프롬프트에서는 ../dictionary/synonym.txt 사전에 따라 동의어 확장이 된 결과가 나온다.

```
----------../dictionary/synonym.txt----------
코난, 코넌
-----------../data/syn-user.0.txt----------
코난, 코눈, 크난

kql/myvol>> select * from tbl where body = '코난' allword synonym;
---------- 0 of total 2 (ROWID 1000) ---------
title (0):
body (6): 코난이
---------- 1 of total 2 (ROWID 1002) ---------
title (0):
```

```
body (6): 코넌
Total 2 records
OK
```

CRZCLI에서 SubmitQuery()를 수행할 경우에는 데몬 동의어 모듈의 사전에 따라서 동의어 확장이 된 결과가 나온다.

```
질의어: W=[body = '코난' allwordthruindex synonym;코난], S=[]
바인딩 필드[2]: title body
검색문서수: 3 (row=3,col=2), (start=0, count=10) 검색시간: 0 msec
+------ 1/3 번째 문서 (1000) SCORE : 9648 MATCHFIELD: -1 ------+
[0](0)
[1](13) <b>코난</b>이
+------ 2/3 번째 문서 (1003) SCORE : 9636 MATCHFIELD: -1 ------+
[0](0)
[1](11) <b>코눈</b>
+------ 3/3 번째 문서 (1005) SCORE : 9636 MATCHFIELD: -1 ------+
[0](0)
[1](11) <b>크난</b>
```

동의어 목록은 독크루저 API GetSynonymList()를 이용하여 시스템 구축 과정에서 활용할 수 있다. API에 대한 설명은 'DOCRUZER API Reference Manual'을 참고한다.

5.4.10. 유사문서 검색 (SIMILAR DOCUMENT SEARCH)

일반적으로 유사문서는 글의 물리적 형식과는 무관하게 의미적 유사성(semantic similarity)을 말한다. 독크루저는 '유사문서'의 개념을 키워드 기반으로 접근한다. 비교 대상인 두 문서가 가지는 키워드들을 분석하여 중요도가 높은 키워드가 두 문서에 공통으로 포함 된 정도를 계산하여 유사성을 판단하는 것이다.

독크루저가 기본적으로 제공하는 검색 방법 similar는 대상 문서에서 정해진 규칙에 따라 중요 키워드를 추출하여 그 키워드로 다시 쿼리를 구성하여 일반 검색과는 달리 단어간 거리나 위치정보는 무시하고 출현 빈도수만을 고려한 고유의 키워드 검색을 수행한다. 그러나, 일부 키워드가 공통되는 연관성이 있는 문서들을 검색할 수 있을 뿐 그 사용성이 제한적이다. 이에 반해, mod-sim(kql240.mod)을 통해 제공되는 '유사문서 검색(similar2)'은 대상 문서의 키워드를 추출하여 통계적 출현 빈도수를 고려하여 검색하는 기본 개념은 similar 검색 방법과 비슷하지만 두 문서간의 유사도를 직접 계산하여 미리 설정된 유사도 기준값 이상의 문서들을 유사도 순으로 제공하고, 여러 필드에 걸쳐 유사도 산출을 할 수 있으며, 세부 기준을 직접 고객사 환경에 맞게 설정하여 사용할 수 있도록 한다.

유사문서 검색 모듈의 구성 요소는 다음과 같다.

구성 부분	설명	비고
kql240.mod	실질적인 요청을 처리하는 데몬 모듈	필수
sim-conf.domain.rc	mod-sim의 환경설정 파일	필수
sim-data.domain.compiled	컴파일된 메모리 이미지 데이터 파일	필수
sim-usr2.domain.txt	사전 파일	필수
sim-stop.domain.txt	금칙어 사전 파일	
kql233.mod	USERDEF 색인을 위한 모듈	필수

유사문서 모듈 설정파일은 sim-conf.domain.rc이며 설정값은 다음과 같다.

파라미터	설명
source_field_name	색인 단계에서 유사문서 검색용 보조 필드에 추출된 키워드를 색인할 원본 필드명 예) `source_field_name = fd`
lnx_domain_no	키워드 추출을 위해 사용할 mod-lnx 모듈의 도메인 번호 예) `lnx_domain_no = 0`
criterion_similarity	유사 여부 판단의 기준값. (0~100). 예) `criterion_similarity = 70`
use_title_similarity	title 보조필드의 사용 여부 예) `use_title_similarity = 1`
title_proportion	title 보조필드의 유사도 기여값. (0~100).

파라미터	설명
	예) `title_proportion = 70`
use_author_identity	author 보조필드의 사용 여부 예) `use_author_identity = 1`
author_proportion	author 보조필드의 유사도 기여값. (0~100). 예) `author_proportion = 70`
use_keyword_weight	레코드별 키워드 가중치 사용 여부 예) `use_keyword_weight = 1`
keyword_weight	레코드별 키워드 가중치 예) `keyword_weight = 100`
max_candidate_count	추출되는 키워드 최대 개수 예) `max_candidate_count = 100`
min_candidate_tf	키워드의 최소 TF 값 예) `min_candidate_tf = 5`
min_candidate_df	키워드의 최소 DF 값 예)

파라미터	설명
	`min_candidate_df = 5`
max_candidate_df	키워드의 최대 DF 값 예) `max_candidate_df =20`
log_level	로그 기록 범위. 0~15. (기본값=0: 로그를 남기지 않음). 예) `log_level = 15`
lgm_domain_no	로그 기록할 mod-lgm의 도메인 번호 예) `lgm_domain_no = 0`
language	사용 언어 예) `language = KOREAN`
charset	사전의 원본 .txt 파일의 charset 예) `charset = EUCKR`

criterion_similarity, title_proportion, author_proportion은 유사 판단에 영향을 미치는 기준 값들이다. 최대 100 % 유사도를 기준으로 볼 때, title의 유사도가 전체 유사도에 기여하는 비율(%)을 title_proportion으로 설정하고, author 동일성 여부가 전체 유사도에 기여하는 비율(%)을 author_proportion으로 설정한다.

```
전체 유사도 = (100 - title_proportion - author_proportion)*(본문유사도)
+ title_proportion*(title유사도) + author_proportion*(author동일성)
```

이렇게 구한 유사도가 criterion_similarity를 넘으면 유사문서로 검색되게 된다.

max_candidate_count는 추출되는 키워드의 최종 개수를 제한하는데, 키워드가 너무 많을 경우 속도가 저하되고, 일정 이상으로 키워드가 많아질 경우 유사도 산출 결과도 나빠질 수 있으므로 사용 환경에 따라 적정한 수로 최대값을 설정할 수 있다.

min_candiate_tf, min_candidate_df, max_candidate_df는 각 키워드의 추출 여부와 가중치를 계산할 때 사용되는 설정값인데, 전체 레코드수의 규모에 따라 더 적정한 유사도 산출을 위해 사용할 수 있으나 일반적으로는 최적화된 기본값을 사용하는 것이 좋다.

log_level, lgm_domain_no는 모듈 자체적인 로그를 기록할 때 사용하는 옵션으로 아무런 설정이 없거나 log_level = 0으로 설정하면 로그를 남기지 않는다.

유사문서 검색을 위한 키워드를 추출하기 위해서는 색인된 전체 문서 집합에서 추출된 모든 색인 키워드와 각각의 DF 값이 필요하다. 이는 색인 후에 kql remote에서 view index to 결과에서 알 수 있다. 따라서, 이 데이터를 파일로 덤프 받아 mod-sim의 sim-usr2.0.txt 사전 데이터로 이용한다. 덤프 받은 파일을 그대로 사용하면 mod-sim에서 자동으로 사전 데이터로 처리된다.

다음의 예는 news.body 필드의 view index to 결과를 ../data/sim-usr2.0.txt로 덤프받는 것이다. 금칙어 등록을 위해서는 sim-stop.0.txt에 기록하면 된다.

```
kql/news>> dump to ../data/sim-usr2.0.txt view index to news.body ;
OK
```

사전파일이 생성되면 다음과 같이 사전 파일을 컴파일하고 엔진에 반영시킨다.

```
DOCRUZER> compile module 240 0;
OK.

DOCRUZER> reload module 240 0;
OK.
```

유사문서 검색을 위해 색인 단계에서 별도의 유사문서 검색용 보조 필드를 추가로 색인하여야 한다. 복제문서 검색과 달리 유사문서 검색에서는 여러 필드에 걸쳐 유사성 판단을 할 수 있으므로 색인용 보조 필드도 그에 따라 여러 개 필요할 수 있다.

유사성 판단의 기준 필드에 대해 '필드명_sim' 필드는 반드시 있어야 하며, 이 필드에 대한 인덱스도 반드시 생성해야 한다. 보조필드들 중에서 '필드_sim'을 제외한 나머지 보조필드는 색인 인덱스를 생성하지 않아도 무방하다.

```
create table tbl
(
```

```
fd              TEXT NULL,
fd_sim          STRING LIST NULL,
fd_sim_title    STRING LIST NULL,
fd_sim_author   STRING NULL,
fd_sim_key      STRING LIST NULL,

constraint c1 TAG(fd_sim) = USERDEF kql233.mod(MOD_SIM.0),
constraint c2 TAG(fd_sim_title) = USERDEF kql233.mod.1(MOD_SIM.1),
constraint c3 TAG(fd_sim_author) = USERDEF copy.so(copy.rc),
constraint c5 TAG(fd_sim_key) = USERDEF kql233.mod.2(MOD_SIM.2),
);

create index body_idx on tbl(fd);
create index body_sim_idx on tbl(fd_sim, fd_sim_title, fd_sim_key);
create index body_sim_author_idx on tbl(fd_sim_author);
```

여기서 string list 필드로 구성되는 fd_sim, fd_sim_title, fd_sim_key의 구분자(delimiter)는 ","이다. 따라서, kql.rc 설정에 string_list_separator = {","}를 반드시 넣어주어야 한다.

위 보조필드 중에서 'fd_sim_title'은 본문 필드와 동일한 방식으로 유사도 산출을 하게 되는 필드이다. 일반적으로 본문(body)과 함께 제목(title)필드를 이용하는 경우가 많아 '필드명_sim_title'이라는 이름으로 만들었을 뿐 어떤 필드를 이 보조 필드의 원본으로 사용하든 상관없다. 보조필드 'fd_sim_author'는 다른 보조필드와 달리 string 필드이다. 문서 작성자의 동일성 여부를 유사도에 반영하고자 할 때 사용하는 필드이다. 보조필드 'fd_sim_key'는 각각의 레코드마다 미리 키워드가 등록되어 있는 경우에 사용하는 것으로, 사용자가 임의로 키워드를 등록하고 이 키워드에 대해 유사도 가중치를 부여하고자 할 경우에 사용하게 된다.

모든 보조필드는 지정된 명명규칙을 따라야한다. 위 예제를 보면, 'fd' 필드를 기준으로 'fd_sim', 'fd_sim_title', 'fd_sim_author', 'fd_sim_key' 필드를 생성하였다. 앞에 공통 된 'fd' 이름은 필드 특성에 따라 변경될 수 있지만 postfix로 붙은 '_sim', '_sim_title', '_sim_author', '_sim_key'는 명명규칙대로 작성해야 한다.

유사문서 검색을 위한 검색 메소드 'similar2'를 이용한다.

```
kql/myvol>>retrieve * from tbl where fd = 1 similar2 absolute ;
```

'retrieve'는 'select'와 같은데 추가로 kql 검색 결과에 점수를 표시 해 준다. 'absolute' 옵션은 mod-sim에서 계산 한 유사도 절대값을 가져온다. 'absolute' 옵션이 없으면, 유사도 점수는 0~10000점 사이로 정규화 된다.

위 검색식은 $ROWID가 1인 문서의 fd 필드값을 기준으로 유사도 높은순으로 유사문서를 검색한다. 유사문서 검색은 인덱스에 대한 검색은 지원되지 않고, 오직 필드에 대한 검색만 지원된다.

자기 자신을 제외한 결과를 보고자 할 때의 검색쿼리는 다음과 같다.

```
kql/myvol>>retrieve * from tbl where fd = 5791 similar2 andnot
$ROWID=5791 absolute ;
```

검색 시 도메인 설정은 시나리오에서 지정 가능하다.

```
scenario tbl
{
    link      = h1
    volume    = myvol
    table     = tvl
    field     = fd (0, "<b>","</b>")
    …
    similar_domain_no = 0
}
```

5.4.11. 복제문서 검색 (REPLICATED DOCUMENT SEARCH)

복제문서란 다른 문서의 내용을 짜깁기하여 생성된 문서를 말한다. 원본 문서의 내용에 약간의 변형을 가해 생성한 문서로 작성자의 기만행위가 들어가 있다. 복제문서 검색은 복제도를 측정하여 기준치 이상의 복제도를 가진 문서를 복제문서로 검색한다.

복제문서 검색 모듈의 구성 요소는 다음과 같다.

구성 부분	설명	비고
kql228.mod	실질적인 요청을 처리하는 데몬 모듈	필수
rpl-conf.domain.rc	mod-rpl의 환경설정 파일	필수
kql233.mod	USERDEF 색인을 위한 모듈	필수

복제문서 모듈 설정파일은 rpl-conf.domain.rc이며 설정값은 다음과 같다. 복제문서 검색에 사용되는 복제문서 검색 모듈(kql228.mod)은 색인 단계와 검색 단계에서 모두 사용된다. 따라서, 색인 단계에 사용될 설정과 검색 단계에 사용될 설정을 모두 rql-conf.0.rc에 포함시켜 두어야 한다.

파라미터	설명
source_field_name	색인 단계에서 복제문서 검색을 위한 문서 내용이 저장된 필드명. 이로부터 정보를 추출하여 '필드명_rpl'에 저장하게 됨. 색인 시 반드시 지정해야 함. 예) `source_field_name = bd`
source_field_language	source_field_name의 언어 (기본값=korean). 예) `ource_field_name=korean`
source_field_charset	source_field_name의 문자 세트 (기본값=euckr). 예) `source_field_name=euckr`
use_pre_refine_flag	복제 판단 시 내용과 무관한 정보를 무시할 지 여부 설정. (기본값=1: 무시함). 예) `use_pre_refine_flag = 0`
feature_delimiter	복제검색용 색인 필드(STRING LIST)의 키 구분자. (기본값= "": 공백문자). 예) `feature_delimiter = |`
feature_level	use_pre_refine_flag = 1로 세팅 한 경우에만 유효한 설정값으로 어떤 종류의 정보를 무시할지를 설정. 0~3단계. (기본값=1). 예) `feature_level = 0`

파라미터	설명
use_multi_expand_evaluate	한 문장의 어절 단위 별로 복제도를 추출하는 옵션. 검색 속도를 고려하여 일정 길이 이하의 본문에서만 작동하는 기능. (기본값=0: 어절단위로 추출하지 않음). 예) `use_multi_expand_evaluate = 2`
criterion_value	복제 여부 판단의 기준 값. (0 ~ 10000). (기본값=5000). 예) `criterion_value = 8000`
log_file_location	로그 디렉터리 생성. 설정 시 해당 경로에 rpl-yyyymmdd.log 파일 생성. 설정하지 않는 경우에는 로그 생성 안함.
expand_line_limit	use_multi_expand_evaluate = 1로 사용 할 경우, 적용 될 문서의 최대 문장 수 제한 설정. (기본값=10). 예) `expand_line_limit = 5`
highlight_prefix	동일 문장의 하이라이팅 prefix 문자열. (기본값=). 예) `highlight_prefix = <i>`
highlight_suffix	동일 문장의 하이라이팅 suffix 문자열. (기본값=). 예) `highlight_suffix = </i>`
highlight_prefix1	유사 문장의 하이라이팅 prefix 문자열. (기본값=). 예) `highlight_prefix1 = <i>`

파라미터	설명
highlight_suffix1	유사 문장의 하이라이팅 suffix 문자열. (기본값=). 예) `highlight_suffix1 = </i>`
use_text_exception	비교대상에서 제외할 문장 등록 기능 설정. (기본값=0).
user_text_for_exception	비교대상에서 제외할 문서 파일 경로 설정
use_highlight_for_eception	비교대상에서 제외된 복제문서의 문장의 하이라이팅 여부 설정. (기본값=0).
highlight_prefix_ex	비교대상에서 제외된 복제문서의 문장의 하이라이팅 prefix 문자열. (기본값= "").
highlight_suffix_ex	비교대상에서 제외된 복제문서의 문장의 하이라이팅 suffix 문자열. (기본값= "").

source_field_name은 색인 단계에서만 필요한 것이다. use_pre_refine_flag를 0으로 설정할 경우에는 예를 들어, 문장 내에서 공백 하나, 쉼표 하나가 달라도 다른 문장으로 취급하게 된다. 일반적인 경우라면 1로 설정하여 사용하는 것이 좋다. feauture_delimiter는 색인 단계에서 문서의 특징을 문장 단위로 추출하여 STRING LIST 형태로 저장 할 때 사용할 키 구분자이다. 기본적으로 공백문자("")를 사용하게 되며, 다른 문자를 사용할 수 있으나, kql.rc의 설정에서 string list 구분자로 설정 되어 있는 것을 사용해야 한다.

featuring_level은 0의 경우 텍스트의 내용과 무관한 특수문자, 중복되는 공백 등을 무시하게 되며, 1의 경우 한글, 영어 만을 인식하고, 2는 한글만 인식하며, 3은 띄어쓰기(공백)도 무시한 한글만을 인식하게 된다. 이 설정의 세부 기능은 모듈의 버전에 따라 다를 수 있다. criterion_value를 기존으로 그 값 이상의 복제도가 있는 문서들만 복제문서 검색에 의해 검색된다.

use_multi_expand_evaluate는 한 문장 안에서 각각의 어절 단위로 비교하여 일정 이상의 어절에 복제 가능성이 있을 경우, 이를 복제도에 반영하는 기능이다. 이 기능은 전체 복제문서 검색의 속도를 고려하여 일정한 길이 이하의 짧은 본문에 대해서만 작동한다.

source_field_language, source_field_charset은 색인 단계에서 사용할 source_field_name 필드의 데이터의 language, charset 정보를 설정한다. 색인 시 kql.rc에 설정되는 language, charset과 동일하게 맞추면 된다.

use_text_exception은 복제 여부 비교 대상에서 제외할 문장을 미리 등록하여, 실제 복제도 비교를 할 때 복제문서에 해당 문장들을 비교 대상에서 제외할지 여부를 설정한다. user_text_for_exception에 설정한 파일에 비교 대상에서 제외할 문장들을 사용자가 입력하여 두면 된다.

use_highlight_for_exception은 위의 use_text_exception 설정에 따라 비교 대상에서 제외된 복제문서상의 문장을 별도로 하이라이팅 하고자 할 경우에 사용한다. highlight_prefix_ex, highlight_suffix_ex를 설정하면 비교 대상에서 제외된 복제문서 상의 문장들을 하이라이팅하게 된다.

설정 파일의 작성 예는 다음과 같다.

```
source_field_name = body

use_pre_refine_flag = 1
featuring_level = 1

use_multi_expand_evaluate = 1
expand_line_limit = 100

highlight_prefix = <b><font color=”red”>
highlight_suffix = </font></b>
highlight_prefix1 = <font color=”blue”>
highlight_suffix1 = </font>

use_text_exception = 1
user_text_for_exception = ./except.txt

use_highlight_for_exception = 1
highlight_prefix_ex = <font color=”grey”>
highlight_suffix_ex = </font>
```

복제문서 검색도 여타의 검색과 마찬가지로 색인과 검색 과정을 거친다. 문서 내용을 저장할 필드 이외에 복제문서 검색을 위한 필드를 추가로 생성해야 한다. 이 필드는 반드시 이름을 '필드명_ rpl'로 해야 하고 속성은 string list로 해야 하며, 두 필드에 대한 인덱스도 반드시 생성해야 한다.

```
create table tbl
(
    tt        TEXT NULL,
    bd        TEXT NULL,
```

```
    bd_rpl    STRING LIST NULL,

    constraint c2 TAG(bd_rpl) = USERDEF kql228.mod (rpl-conf.0.rc)
);
create index bd_idx on tbl(bd);
create index rpl_idx on tbl(bd_rpl);
```

'필드명_rpl' 필드의 string list 구분자(delimiter)는 feature_delimiter에 설정된 값이다. 따라서, 색인을 위한 kql.rc 설정에 해당 구분자가 string list 구분자로 포함되도록 해야 한다.

검색은 $ROWID와 문서 내용을 이용하여 복제문서를 검색할 수 있다.

'retrieve'는 'select'와 같은데 추가로 kql 검색 결과에 점수를 표시 해 준다. 'absolute' 옵션은 mod-rpl에서 계산 한 복제도 절대값을 가져온다. 'absolute' 옵션이 없으면, 복제도 점수는 0~10000점 사이로 정규화 된다. 참고로 일반 검색은 복합 인덱스에 대한 검색이 가능 하지만, 복제문서 검색은 오로지 필드에 대한 검색만 지원한다. 이는 유사문서 검색(similar)과 같다.

다음은 $ROWID를 이용한 복제문서 검색의 예이다. ROWID가 55번인 문서의 bd 필드를 기준으로 복제문서 검색한다.

```
kql/tv>> retrieve tt from tbl where bd=55 replicate absolute;
```

다음은 문서 내용을 이용한 복제 문서 검색의 예이다. 문서 내용으로 찾을 때는 위와 같이 쿼리를 만들면 된다. bd 값으로는 문서 내용 전체가 올 수 있다. 텍스트 값을 그대로 쿼리를 만드는 경우 전체 쿼리 구문이 깨지지 않도록 문자 이스케이핑에 주의해야 한다.

```
kql/tv>>retrieve tt from tbl where bd = '그 책에 따르면 "표절이란
의도적이건 아니건 간에 남의 말이나 생각을 사용하면서 그 사람에게
크레딧을 주지 않는 행위이다. 비록 크레딧을 준다고 할지라도 인용부호를
치거나 묶음인용의 형태를 취하지 않고 다른 사람의 말을 마치 자기 말인
것처럼 전재한 것도 표절에 속한다." 문장을 가져다 쓴 것만 표절이 아니다.
남의 "생각"도 가져다 쓰면 표절이 된다.' replicate absolute;
```

여러 필드에 대한 복제 검색

전체 독크루저 구성에서 여러 필드 데이터에 대해서 서로 다른 설정으로 복제문서 검색을 수행하고자 할 경우에는 복제문서 검색 모듈의 도메인을 여러 개 사용하여야 한다.

1. 색인 단계에서는 USERDEF에 사용 할 모듈 kql228.mod를 여러벌 복사하여 각각을 사용하고, 이때에 사용 할 rpl-conf.domain.rc 파일을 각 필드에 사용 할 복제 검색 도메인과 동일하게 사용하여야 한다.

```
create table rpl_multi
(
    fd0        text null,
    fd0_rpl    string list null,
    fd1        text null,
    fd1_rpl    string list null,
    fd2        text null,
    fd2_rpl    string list null,

constraint c1 TAG(fd0_rpl) = USERDEF kql228.mod.0 (rpl-conf.0.rc)
constraint c2 TAG(fd1_rpl) = USERDEF kql228.mod.1 (rpl-conf.1.rc)
constraint c3 TAG(fd2_rpl) = USERDEF kql228.mod.2 (rpl-conf.2.rc)
);

create index i  on rpl_multi(fd0);
create index i2 on rpl_multi(fd0_rpl);
create index i  on rpl_multi(fd1);
create index i2 on rpl_multi(fd1_rpl);
create index i  on rpl_multi(fd2);
create index i2 on rpl_multi(fd2_rpl);
```

2. 시나리오 설정에서 사용 할 도메인 번호를 지정한다. kql 인터프리터에서는 사용하는 도메인이 0번으로 고정되어 있으므로 1번 이상의 도메인을 사용하는 필드에 대한 복제문서 검색은 kql 인터프리터에서는 결과가 정확하지 않을 수 있음을 주의한다.

```
scenario rpl_multi0
{
    link    = h1
    volume  = rpl_multi
    table   = rpl_multi

    field   = fd0  (200, "<b>","</b>")
    field   = fd1  (200, "<b>","</b>")
    replicate_domain_no = 0
}
```

```
scenario rpl_multi1
{
    link    = h1
    volume  = rpl_multi
    table   = rpl_multi

    field   = fd0  (200, "<b>","</b>")
    field   = fd1  (200, "<b>","</b>")
    replicate_domain_no = 1
}
```

3. 검색 수행을 위해 SubmitQuery(), Search()에 select 구문의 where 이후의 쿼리를 구성할 경우에 복제문서 검색을 할 대상 필드와 시나리오의 복제검색 모듈 도메인이 바르게 매칭되었는지 확인한다. (DOCRUZER API 매뉴얼 참고).

```
$./crzcli 192.168.7.4 6354 rpl_multi0
>fd0 = 3 replicate absolute;

$./crzcli 192.168.7.4 6354 rpl_multi1
>fd1 = 3 replicate absolute;
```

독크루저의 복제검색 모듈(mod-rpl:kql228.mod)은 기본 도메인 개수가 1개로 초기화 되어 있다. 여러 도메인을 사용하기 위해서는 docruzerd.rc 설정에 daemon_module_domain_count_rpl = 3 (필요한 개수 만큼)으로 미리 설정 하여야 한다.

5.4.12. 요약/하이라이팅 (SUMMARIZATION/HIGHLIGHT)

검색 요청이 들어오면 일반적으로 본문 중에서 검색어가 포함된 일부분을 요약문의 형태로 보여주게 된다. 이때 요약문 내의 검색어에 하이라이팅 처리를 해 준다.

요약/하이라이팅 모듈의 구성 요소는 다음과 같다.

구성 부분	설명	비고
kql098.mod	실질적인 요청을 처리하는 데몬 모듈	필수
stx-conf.rc	MOD-STX의 환경설정 파일	필수
kql236.mod	개인정보 마스킹을 처리하는 데몬 모듈	

구성 부분	설명	비고
prv-conf.domain.rc	MOD-PRV(개인정보 추출/차단 모듈)의 환경설정 파일	
kql223.mod	USERDEF 색인을 위한 모듈	

요약/하이라이팅 모듈 설정파일은 stx-conf.rc이며 설정값은 다음과 같다.

설정 항목	설명
stx_multi_summarize	본문이 길 경우 키워드가 나타나는 본문의 여러 부분에서 요약문을 추출하는 기능. (기본값= 0). 예) `stx_multi_summarize= 1`
use_first_match_summarize	키워드가 처음 나타나는 곳에서 요약. (기본값= 0). 예) `use_first_match_summarize = 1`
use_mask_private_information	요약문 결과에서 개인정보를 masking함. (기본값= 0). 예) `use_mask_private_information = 1`
prv_domain_no	사용할 mod-prv의 도메인 번호 설정. 예) `prv_domain_no = 0`
prv_masking_character	포함된 개인정보를 매스킹할 문자 지정. 예) `prv_masking_charactor = *`
log_level	ERROR=1, WARN=2, INFO=4, DEBUG=8의 4단계로 구분되며 각 숫자를 합한 값을 설정.

설정 항목	설명
	예) `log_level = 15`
lgm_domain_no	로그를 실제로 기록 해 줄 mod-lgm의 도메인 번호 설정. 예) `lgm_domain_no = 0`

개인정보 마스킹 기능

use_mask_private_informat은 개인정보 마스킹 기능으로, 검색 결과를 최종적으로 화면에 노출하는 단계에서 주민번호, 전화번호 등의 개인정보가 노출되지 않도록 보호하기 위해 사용한다. 개인정보 추출, 마스킹 기능을 직접 수행하는 모듈은 mod-prv(kql236.mod)이다. 요약/하이라이팅 모듈은 이 mod-prv를 이용하여 개인정보 마스킹을 수행한다. 이 기능을 수행하기 위해서는 kql236.mod와 prv-conf.domain.rc가 필요하다.

다음은 mod-prv에서 추출 가능한 개인정보이다.

```
주민번호(resident_registration_number)
신용카드번호(credit_card_number)
전화번호(phone_number)
이메일주소(email_address)
은행계좌번호(bank_account)
```

위 다섯가지 정보에 대해 추출 여부를 설정할 수 있으며, 기본값은 모두 추출하는 것이다. prv-conf.domain.rc에서 설정한다. 요약/하이라이팅 모듈에서의 개인정보 마스킹 기능에 대한 자세한 정보는 '개인정보 추출/차단 (EXTRACT PERSONAL INFORMATION)'을 참고한다.

complex_field 기능

요약/하이라이팅 기능은 다음과 같이 시나리오에서 해당 필드마다 설정한다.

```
scenario tbl {
    link          = h1
    volume        = myvol
    table         = tbl
    field         = fd
```

```
    field         = fd2(200, "<b>", "</b>")
    complex_field = fd3(300, "<b>", "</b>", "<__delim__>",
                    KEYWORDONLY)
    field         = fd4
}
```

위 시나리오 설정에서 field = fd2(200, "", "")의 설정이 기본적인 요약 하이라이팅 설정이다. 해당 테이블의 fd2 필드의 본문에서 200 바이트 요약문을 만들고, 그 요약문에 포함된 키워드의 앞뒤에 와 를 넣어 하이라이팅 하겠다는 뜻이다.

complex_field는 여러 개의 첨부 파일을 하나의 필드에 색인하여 사용하는 경우에 유용하게 쓸 수 있다. complex_field = fd3(300, "", "", "<__delim__>", KEYWORDONLY)는 기본 설정과 같으나 <__delim__>은 여러 첨부문서를 하나의 필드에 채운 경우 그 delimiter를 나타내며, 이는 색인할 때 사용한 첨부문서 본문 필드내의 delimiter와 같아야 한다.

예를들어, 첨부파일1 = "가가가가가" 첨부파일2 = "나나나나나"라고 할 경우에 delimiter를 "<<aaa>>"로 한다면, 색인되는 첨부파일 필드의 값은 "가가가가가<<aaa>>나나나나나"로 될 것이다. 이런 경우에 complex_field = fd3(300, "", "", "<<aaa>>", KEYWORDONLY)로 할 수 있을 것이다.

결국, complex_field 시나리오 설정은 delimiter로 구분된 여러 문서를 각각 요약/하이라이팅하고 그 결과도 delimiter로 구분하여 출력해 주는 것이다. 만약 하나의 문서에 대해 complex_field 기능을 이용하고 싶다면 delimiter를 ""으로 설정하면 된다.

complex_field의 마지막 설정값은 각각의 첨부파일 본문을 어떻게 요약할 것인가 방법을 설정하는 것인데 다음과 같은 것이 있다.

설정값	설명
KEYWORDONLY	본문에서 매칭된 키워드만 출력한다.
MATCHEDONLY	키워드 매칭 되지 않은 텍스트는 보여주지 않는다.
MULTIPOINT	키워드가 출현한 여러 지점을 요약한다.
FROMSTART	무조건 본문 처음부터 일정 길이를 요약하며 하이라이팅도 한다.
FIRSTMATCH	처음 키워드가 출현한 지점을 요약한다.

5.4.13. 검색 캐시 (SEARCH CACHE)

독크루저 캐시 모듈은 반복된 동일한 검색 요청에 대해 KQL을 거치지 않고 독크루저 데몬의 제어를 받는 캐시 모듈에 미리 저장된 결과를 전송한다. 이로써 검색엔진의 부하를 줄이고 동시에 검색 요청에 대한 응답 속도를 개선할 수 있다.

캐시 전체 엔트리의 개수를 지정할 수 있고, 캐시 life time을 설정할 수 있다. 또한 캐시을 위해 사용하는 메모리 용량에 부담을 느끼는 경우에는 캐시 데이터를 메모리가 아닌 파일로 관리하여 사이즈가 상대적으로 매우 큰 검색 결과에 대해서도 사용이 가능하다. 이 경우에도 캐시 속도에 관계된 부분은 메모리를 사용하므로 캐시 성능의 큰 저하는 없다.

캐시 모듈을 사용하기 위해서는 docruzerd.rc에 캐시 도메인 번호를 설정해 주어야 한다.

설정 예는 다음과 같다.

```
---------- docruzerd.rc -------------
…
daemon_cache_domain_no_search = 0
…
```

참고로, 데몬 캐시 모듈의 작동 여부를 확인하기 위해서는 시스템 로그 파일(YYYYMMDD- nn.msg)을 확인하면 된다. 만약 데몬 캐시 모듈이 작동 중 이라면, .msg 파일에 다음과 같은 로그가 기록될 것이다.

1. [20090108:195030] daemon cache module loaded. domain=0 ((2-0) Jan 7 2009 13:00:03 (Thu Jan 8 19:50:08 2009)): 데몬이 재기동 된 후 캐시 모듈이 정상적으로 로딩된 것을 나타내며, 캐시 모듈의 도메인 번호와 모듈 버전 정보가 기록된다.

2. [20090108:195030] CACHE 0: mem=0, usage=0, collision=0, store=0, lookup=0, hit=0, miss=0: 주기적으로 캐시의 사용률을 표시해 준다.

데몬 캐시 모듈의 구성 요소는 다음과 같다.

구성 부분	설명	비고
kql099.mod	실질적인 요청을 처리하는 데몬 모듈	필수
cch-conf.domain.rc	MOD-RPK의 환경설정 파일	필수
cch-data.domain.compiled	컴파일 된 메모리 이미지 데이터 파일	필수

데몬 캐시 모듈 설정 파일은 다음과 같은 내용이 포함된다.

파라미터	설명
cache_entry_size	캐시 엔트리의 크기
cache_timer_sec	캐시에 저장 된 데이터의 유효기한. 초 단위.
cache_release_resource	하나의 캐시 엔트리가 사용하는 메모리가 이 값을 넘을 경우 다음번 사용시에 메모리를 해제하고 초기화 함. 바이트 단위. (파일 캐시에서는 지원 안됨)
use_file_cache	메모리가 아닌 파일 캐시를 사용할지 여부를 세팅
cache_file_path	파일 캐시를 사용 할 경우, 캐시에 사용 할 파일의 경로를 설정.
max_file_cache_entry_unit_size	파일 캐시에서 한 엔트리의 최대 크기. 이보다 큰 캐시 데이터는 저장 안됨.

cch-conf.0.rc 파일의 설정 예는 다음과 같다.

```
cache_entry_size = 10000
 ; 캐시 엔트리 수. 최대 1만 개의 검색 결과를 저장.
cache_timer_sec = 180
 ; 데이터 유효기한. 하나의 검색 결과가 저장된 후 180 초가 경과되면
   이 데이터는 무효화 됨.
cache_release_resource = 1024
 ; 캐시 되는 검색 결과의 크기가 1024 바이트를 넘을 경우에 해당
   엔트리의 데이터가 무효화된 이후 재사용될 때에는 해당 엔트리의
   메모리는 다시 초기화
use_file_cache = 1                          ; 파일로 관리
cache_file_path = ../cache0                 ; 캐시 파일 디렉터리
max_file_cache_entry_unit_size = 4096
  ; 파일 캐시의 한 엔트리의 최대 크기. 이 크기를 넘는 검색 결과는
    파일 캐시에 저장되지 않는다.
```

priority cache 기능

캐시 할 데이터에 priority를 설정할 수 있다. 사용자 파일인 cch-user.domain.txt와 cch-usr2.domain.txt에 각각 priority 설정할 키 값을 저장하여 둔다. cch-user.domain.txt에 등록된 키워드는 priority=1, cch-usr2.domain.txt에 등록 된 키워드는 priority=2 그리고 아무런 priority 설정이 되지 않은 일반적인 경우는 priority=0으로 작동 된다.

기본적으로 캐시 데이터는 설정된 life time에 따른 우선순위 정책에 따라, life time이 경과 된 데이터를 무효화 하고 신규 데이터를 캐시한다. 이런 경우, 캐시에 기록된 시간 외에는 아무런 우선순위 조작을 할 수 없으므로, 상대적으로 검색결과에 변경 가능성이 낮고 무거운 쿼리로 구성된 데이터를 인위적으로 우선순위를 높여주려고 할 때 priority cache를 사용하면 된다.

데몬 캐시를 독크루저에서 사용하기 위해서는 캐시 데이터를 컴파일 하는 과정이 필요하며, compile 명령 형식은 다음과 같다.

```
DOCRUZER> compile module 99 0;
OK.

DOCRUZER> reload module 99 0;
OK.
```

컴파일이 완료되면 cch-data.0.compiled와 같은 파일이 생성된다.

5.4.14. 문서 필터 (TEXT FILTER)

문서 필터는 다양한 포맷의 파일들에 대해 텍스트를 추출한다. 주로 첨부문서가 있는 데이터를 색인 할 때 문서필터를 사용 한다. 필터링 가능한 포맷은 HTML, HWP, DOC, XLS, PPT, PDF, HWD, MP3, TXT, RTF으로 지정할 수 있으며 포맷을 미리 알 수 없는 경우는 'AUTO'로 설정 가능하다.

문서 필터 모듈 구성은 다음과 같다. HTML을 필터링할 경우 HTML 필터 모듈과 함께 사용한다

구성 부분	설명	비고
bin/txf3d.bin	문서 필터 실행 파일	필수
bin/txf3-conf.rc	MOD_TXF3의 환경설정 파일	필수
data/kql214.mod	실질적인 요청을 처리하는 데몬 모듈	필수
data/kql072.mod	HTML 필터 모듈	
bin/html-conf.rc	HTML 필터 모듈의 설정 파일	
data/kql233.mod	USERDEF 색인을 위한 모듈	

문서 필터 모듈 설정파일은 txf3-conf.rc이며 설정값은 다음과 같다.

파라미터	설명
sys_log	txf3 로그의 위치 예) `sys_log = ../logs/filter`
log_level	로그 레벨 지정. INFO=1, WARN=2, ERROR=4, DEBUG=8, TRACE=16 (기본값=7). 예) `log_level = 31`
separator	멀티 파일명 구분자 예) `separator = "|"`
deny_suffix	필터링하지 않을 확장자 예) `deny_suffix = jpg`
use_utf8_output	UTF8 필터링 결과 (기본값= 0). 예) `use_utf8_output = 1`
use_utf8_filename	UTF8 파일 이름 (기본값= 0). 예) `use_utf8_filename = 1`
html_module_file	kql072.mod의 위치 예) `html_module_file = ../data/kql072.mod`

파라미터	설명
text_separator	멀티 파일 결과값 구분자 예) `text_separator = ##@@##`
max_file_size_kb	필터링 파일 크기 제한. 사이즈보다 큰 파일은 skip한다. 예) `max_file_size_kb = 1024`
max_output_text_size_kb	필터링 결과의 텍스트 크기를 제한하며 지정한 크기로 사이즈를 축소한다. (기본값=1024). 예) `max_output_text_size_kb = 512`
userdef_module_file_[0~4]	userdef 파일 위치 예) `userdef_module_file = ../data/userdef_txf3.so`
txf3d_port	txf3d.bin 네트워크 포트 예) `txf3d_port = 31045`
txf3d_idle_time	txf3d.bin에 인풋이 없을 시에 살아 있는 시간 지정. (기본값=100). 예) `txf3d_idle_time = 600`
txf3d_timeout_sec	txf3d.bin이 지정된 시간내에 응답이 없을 시 종료. (기본값=100).

파라미터	설명
	예) `txf3d_timout_sec = 100`
use_gul_filter	훈민정음 필터링. 훈민정음 필터링 바이너리 파일 필요. (기본값=0). 예) `use_gul_filter = 1`

gul_filter_cmd	훈민정음 필터 사용 커맨드. 훈민정음 필터링 바이너리 파일 필요.
	예)
	`gul_filter_cmd`
	` = brconvjug2txt rnc_sstt091228 –f –a`

discard_begin_str	지정된 문자열로 시작 시 필터링 결과 삭제 예) `discard_begin_str = SCSA`
txf_cache_file_location	필터 캐쉬 파일 위치 예) `txf_cache_file_location = ../txf_cache/`
txf_cache_use_file _modification_check	캐쉬 사용 시 파일 변경 체크. (기본값=1). 예) `txf_cache_use_file_modification_check = 1`
filtering_sleep_time_ms	중간중간 CPU 점유율을 위해 다른 회사 라이브러리 호출 전 후로 CPU 반환. (기본값=0). 예)

파라미터	설명
	`filtering_sleep_time_ms = 1000`
use_drm_bin	DRM 바이너리 복호화 실행 예) `use_drm_bin = 1`
use_drm_java	DRM java 복호화 실행 예) `use_drm_java = 1`
exclude_abnormal_numbers	10 자리 이상으로 나오는 정상적이지 않은 숫자 삭제. (기본값=0). 예) `exclude_abnormal_numbers = 1`
exclude_excel_document_numbers	엑셀 파일에서 10 자리 이상으로 나오는 숫자 삭제. (기본값=0). 예) `exclude_excel_document_numbers = 1`

HTML 필터 설정파일은 html-conf.rc로 설정값은 다음과 같다.

파라미터	설명
output_charset	아웃풋 인코딩 지정. (기본값=euc-kr). 예) `output_charset = utf8`
filtering_lt_gt_html_tag	< &gt 태그를 '<' '>' 문자로 인식. (기본값=0). 예)

파라미터	설명
	`filtering_lt_gt_tag = 1`
exclude_tag	필터링되지 않을 태그 지정. 예) `exclude_tag = "IMG-CAPTION"`
replace_tag	문서 내의 모든 지정된 태그를 지정된 문자열로 교체 예) `replace_tag = "<CLAIM N=1>","[청구항 1]"`
filtering_between_html_tag	<html> ~ </html> 사이의 문자열만 필터링. (기본값=0). 예) `filtering_between_html_tag =1`
filtering_img_alt_sum_tag	img 태그 내의 alt 속성 추출. (기본값=0). 예) `filtering_img_alt_sum_tag = 1`
filtering_php_tag	PHP 문법 제거. (기본값=0). 예) `filtering_php_tag = 1`
filtering_jsp_tag	JSP 문법 제거. (기본값=0). 예) `filtering_jsp_tag = 1`
filtering_asp_tag	ASP 문법 제거. (기본값=0). 예)

파라미터	설명
	`filtering_asp_tag = 1`

문서 필터를 사용하기 위해서는 테이블 정의 ddl 구문에 다음과 같이 constraint를 사용해 주면 된다.

```
=====================================================================
file_body 필드에는 로컬 파일경로가 있고, TEXT_FILTER_FB()를 통해 필터링된
파일 내용이 file_body에 들어간다.
=====================================================================
constraint c1 TAG(file_body) = TEXT_FILTER_FB(file_body, "AUTO")

=====================================================================
위 구문을 아래와 같이 두개의 constraint 구문으로 사용할 수 있다.
=====================================================================
constraint c1 TAG(file_body) = LOCALREF(file_body)
constraint c2 TAG(file_body) = TEXT_FILTER_BB(file_body, "AUTO")

=====================================================================
file_body 필드의 파일 경로 앞에 경로명을 추가할 필요가 있을 경우
LOCALREF_PREFIXED(), TEXT_FILTER_BB()를 사용한다.
=====================================================================
constraint c1 TAG(file_body)
 = LOCALREF_PREFIXED("../dump", file_body)
constraint c2 TAG(file_body) = TEXT_FILTER_BB(file_body, "AUTO")

=====================================================================
file_body 필드의 파일경로가 원격일 경우 REMOTE_TEXT_FILTER_FB()을
이용한다.
=====================================================================
constraint c1 TAG(file_body) = REMOTE_TEXT_FILTER_FB (file_body,
"AUTO")
```

문서 필터 캐시

문서 필터 캐시란 한번 필터링한 문서는 다시 필터링하지 않도록 하는 기능이다. 색인 과정 중에서도 시간이 많이 걸리는 필터링 시간을 단축할 수 있다.

문서 필터 캐시를 사용하려면 txf3-conf.rc에 문서 필터 캐시 설정을 추가하고 txf_cache_file_location으로 설정된 디렉터리를 생성한다.

문서 필터 캐시의 설정(txf3-conf.rc) 예는 다음과 같다.

```
txf_cache_file_location = ../txf_cache
txf_cache_use_file_modification_check = 1
```

5.4.15. 멀티미디어 필터 (MULTIMEDIA FILTER)

멀티미디어 필터는 멀티미디어 데이터로부터 데이터 분석을 위해 필요한 정보들을 추출하는 모듈이다. (멀티미디어 데이터는 동영상, 오디오, 이미지 파일들을 의미한다.) 멀티미디어 필터는 멀티미디어 데이터에 대한 설명을 담고 있는 태그 정보 추출, 동영상의 썸네일 이미지 추출, 그리고 마지막으로 멀티미디어 데이터를 분석이 용이한 포맷으로 디코딩 하는 것으로 분류된다.

디코딩은 멀티미디어 데이터의 종류에 따라 이미지, 오디오, 비디오 디코딩으로 분류된다.

- 이미지 디코딩: JPG, PNG 등 다양한 포맷의 이미지 데이터를 RAW(=RGB) 포맷으로 변환
- 비디오 디코딩: 소리와 영상으로 구성된 동영상 데이터로부터 영상(화면) 데이터를 추출하여 YUV 420P 포맷으로 변환
- 오디오 디코딩: 동영상 데이터로부터 오디오(소리) 데이터를 추출하여 WAV(PCM) 포맷으로 변환

멀티미디어 데이터의 종류에 따른 멀티미디어 필터의 수행 기능은 다음과 같다.

구분	비디오 디코딩	오디오 디코딩	이미지 디코딩	섬네일 이미지	태그 정보
동영상	○	○		○	○
오디오		○			○
이미지			○		○

태그 정보 추출

멀티미디어 데이터로부터 태그 정보를 추출한다. 태그 정보는 멀티미디어 데이터에 대한 간략한 소개를 담고 있는 메타 정보로서 데이터의 종류에 따른 태그 정보 항목은 다음과 같다.

- 비디오 태그 정보

구성 부분	설명
제목	-
비디오 포맷	-
재생 시간	sec
가로	pixel
세로	pixel
전체 비트율	Kbps
비디오 비트율	Kbps
오디오 비트율	Kbps
프레임률	fps
오디오 샘플링 주파수	Hz
오디오 채널 수	-

- 오디오 태그 정보

구성 부분	설명
트랙 번호	-
오디오 포맷	-
제목	-
가수	-
앨범	-
년도	-
장르	-
설명	-
작곡가	-
원 저작자	-
저작권	-
URL	-

구성 부분	설명
인코더	-
재생 시간	sec
샘플링 주파수	Hz
채널 수	-
비트율	kbps

- 이미지 태그 정보

의미	단위	구분
EXIF 정보의 존재유무	-	1: yes, 0: no
카메라 제조사	-	-
카메라 모델	-	-
사진 찍은 날짜	-	-
해상도	pixel	Width x Height
플래쉬 사용 유무	-	– 0x5: (Strobe light not detected) – 0x7: (Strobe light detected) – 0x9: (manual) – 0xd: (manual, return light not detected) – 0xf: (manual, return light detected) – 0x18: (auto) – 0x19: (auto) – 0x1d: (auto, return light not detected) – 0x1f: (auto, return light detected)

의미	단위	구분
		– 0x41: (red eye reduction mode) – 0x45: (red eye reduction mode return light not detected) – 0x47: (red eye reduction mode return light detected) – 0x49: (manual, red eye reduction mode) – 0x4d: (manual, red eye reduction mode, return light not detected) – 0x4f: (red eye reduction mode, return light detected) – 0x59: (auto, red eye reduction mode) – 0x5d: (auto, red eye reduction mode, return light not detected) – 0x5f: (auto, red eye reduction mode, return light detected) – others: flash not used
렌즈 초점 길이	mm	-
35mm 기준 초점 길이	mm	-
디지털 줌 비율	-	– 1 초과: used – 1 이하: not used
노출 시간	sec	-
조리개	f	-
초점 거리	m	– 0 이하: Infinite

의미	단위	구분
CCD 크기	mm	-
ISO 값	-	-
화이트 밸런스	-	– 0: auto – 1: manual
측광 모드	-	– 2: center weight – 3: spot – 5: matrix
노출 방식	-	– 1: Manual – 2: program (auto) – 3: aperture priority (semi-auto) – 4: shutter priority (semi-auto) – 5: Creative Program (based towards depth of field) – 6: Action program (based towards fast shutter speed) – 7: Portrait Mode – 8: LandscapeMode
노출 모드	-	– 0: auto – 1: manual – 2: auto bracketing

의미	단위	구분
광원	-	– 1: Daylight – 2: Fluorescent – 3: Incandescent – 4: Flash – 9: Fine weather – 11: Shade

섬네일 이미지 추출

동영상 데이터로부터 썸네일 이미지를 JPG 포맷으로 추출한다. 동영상의 특정 시점의 화면을 이미지 파일로 추출하는 기능이다.

디코딩

멀티미디어 데이터를 분석이 용이한 포맷으로 디코딩한다.

- 비디오 디코딩
  - 동영상 데이터로부터 영상(화면) 정보를 디코딩
  - YUV 420P 포맷으로 디코딩
- 오디오 디코딩
  - 동영상/오디오 데이터로부터 오디오(소리) 정보를 디코딩
  - WAV(PCM) 포맷으로 디코딩
- 이미지 디코딩
  - 다양한 종류의 이미지 데이터를 RAW 포맷으로 디코딩

– 지원 포맷: BMP, GIF, ICO, JPG, PBM, PCX, PNG, PPM, TIF

리샘플링

원본 이미지를 지정한 가로, 세로 크기로 리샘플링

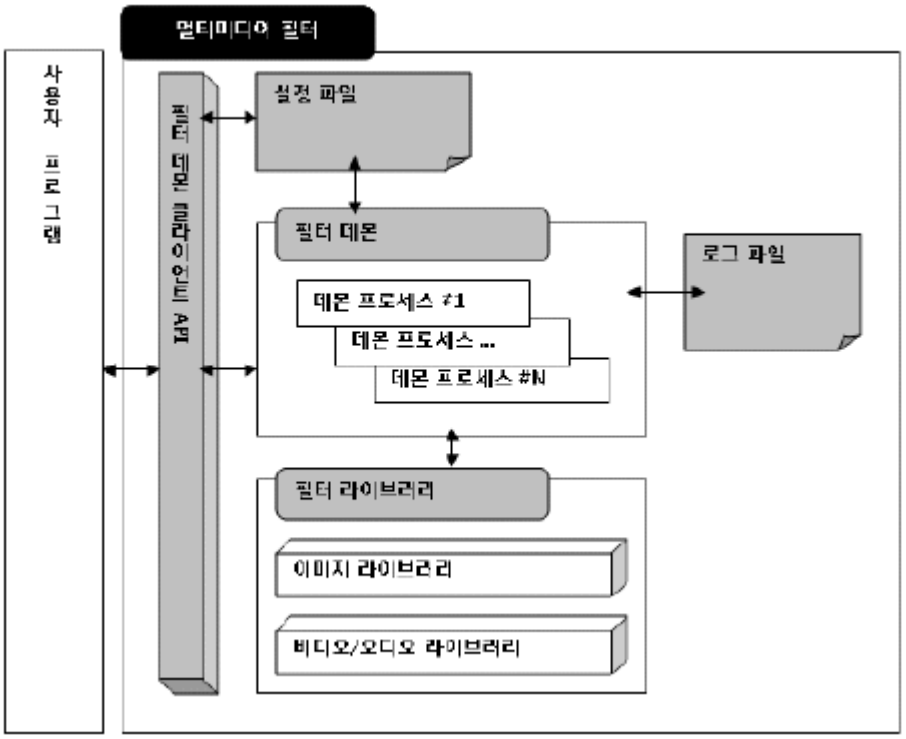
- 이미지 리샘플링
  - 원본 이미지를 jpg 포맷으로 리샘플링
  - 지원 포맷: GIF, JPG, PBM, PCX, PNG, PPM, SWF, TGA

5.4.15.1. 구조

멀티미디어 필터 모듈은 필터 데몬, 필터 데몬 클라이언트 API, 필터 라이브러리, 설정 파일, 로그 파일로 이루어져 있다.

- 필터 데몬: 필터 라이브러리를 호출하여 필터 모듈의 기능(태그 정보, 동영상 썸네일, 디코딩)을 서비스하는 프로그램
- 필터 데몬 클라이언트 API: 필터 데몬을 조작하기 위한 인터페이스로서 사용자에게 공개됨. 사용자는 클라이언트 API를 통해서 필터 데몬에 간접적으로 명령 수행을 요청
- 필터 라이브러리: 태그 정보, 썸네일 추출, 디코딩 등의 멀티미디어 필터의 기능을 실제로 수행하는 라이브러리로서 라이브러리 형태로 제공. 이미지 필터 라이브러리, 비디오/오디오 필터 라이브러리로 구성되며, 각각 이미지, 비디오/오디오에 대한 정보 추출을 담당한다.
- 설정 파일: 필터 모듈의 구동 시 적용할 옵션들을 설정하는 파일
- 로그 파일: 필터 데몬이 구동 중에 발생하는 실패 상황들에 대한 이력이 기록되는 파일

멀티미디어 필터 모듈은 멀티프로세스 모델로서 동시에 병렬적으로 사용자 요청 작업을 처리할 수 있다. 필터 데몬과 데몬 클라이언트 API는 TCP/IP 소켓 통신을 통하여 결과를 주고 받는다. 지금까지 설명한 필터 모듈의 구조는 아래 그림과 같다.



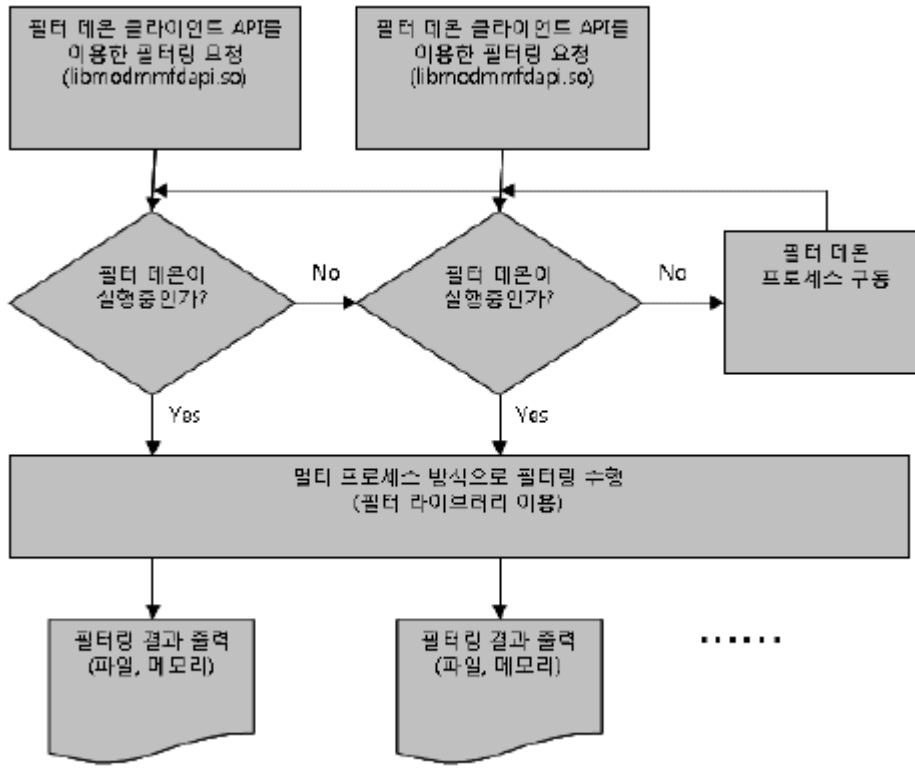
[그림 5.4] 멀티미디어 필터 구조

- 필터 데몬: 필터 데몬 실행 파일, 멀티 프로세스로 동작
  - 파일명: mmfd.bin
- 필터 데몬 클라이언트 API: 동적 라이브러리
  - libmodmmfdapi.so
- 필터 라이브러리(동적 라이브러리)
  - 이미지 필터 라이브러리(libmodimgf.so)
  - 비디오/오디오 필터 라이브러리(libmodmmf.so)
- 설정 파일: 필터 환경 설정 파일
  - mmfd.rc

- 로그 파일: 필터 에러 로그 파일
  - mmfd_err.log

5.4.15.2. 수행 과정

멀티미디어 필터 모듈의 실행은 클라이언트와 데몬의 통신 과정으로 아래와 같은 순서로 이루어 진다. 필터 데몬 클라이언트 API를 통해서 명령을 요청하면 먼저 필터 데몬이 실행 중인지를 확인한다. 실행 중이면 필터 데몬이 해당 명령을 수행한 후 결과를 출력하고, 실행 중이 아니면 필터 데몬을 구동한 후 구동된 필터 데몬이 명령을 수행하여 결과를 출력한다. 필터 데몬은 처음 구동 후 일정시간(설정 파일을 통해 설정)동안 시스템에 상주하고 일정 시간 아무 명령 요청이 없으면 스스로 종료한다.



[그림 5.5] 멀티미디어 필터 수행 과정

5.4.15.3. 설정 파일

멀티미디어 필터 모듈의 설정 파일은 필터 모듈이 사용하는 다양한 환경 설정 값이 기록되어 있는 파일이다. 설정 파일의 값은 필터 데몬 클라이언트 API와 필터 데몬이 참조하며, 필터 모듈의 구동 시에 설정 값이 반영된다. 따라서 새로운 값을 적용하기 위해서는 필터 데몬을 재구동 하여야 한다. 설정은 설정 변수와 그 값의 쌍들로 이루어지며, 작성 형식은 한 줄당 '설정 변수명=설정값'이다. 일부 설정 변수는 생략가능하며 생략된 설정 변수는 기본값으로 실행된다.

각 항목에 대한 설명은 다음과 같다.

구성 부분	설명
time_out_sec	필터링 요청 후 지정한 시간 동안 응답이 없으면 필터링 관련 함수는 -1을 반환하고 "filter time out(%d ms)" 메시지가 기록된다. 명령 수행 시간이 오래 걸린다면 충분한 시간을 설정하도록 한다. 기본값은 10 초 이다. 예) time_out_sec = 10
daemon_name	필터 데몬 실행 파일의 경로를 입력 받아, 이후 이 값을 이용하여 데몬 프로세스를 실행한다. 기본값은 mmfd.bin 이다. 예) daemon_name = mmfd.bin
daemon_port	필터 데몬 프로세스가 사용할 소켓의 포트 번호를 설정한다. 기본값은 31045 이다. 예) daemon_port = 31045
idle_time_sec	지정한 시간 동안 필터 데몬으로 요청이 없으면 필터 데몬 프로세스는 스스로 종료한다. 기본값은 6초 이다. 예) 예) idle_time_sec = 6
sys_log	로그 파일의 경로를 설정한다. 생략하면 기본값은 mmfd_err.log 이다. 예) sys_log = mmfd_err.log
no_of_seek_key_frames_for_video_decode	비디오 디코딩 옵션으로 부분적 디코딩 시 기준 단위를 설정한다. 기본값은 0이며, 기본 디코딩을 수행한다.

구성 부분	설명
	예) no_of_seek_key_frames_for_video_decode = 0
no_of_decoding_frames_per_video_key_frame	비디오 디코딩 옵션으로 부분적 디코딩 시 단위 프레임 수를 설정한다. no_of_seek_key_frames_for_video_decode 옵션과 함께 사용되며, mmf_video_decode_level 값이 0 이상일 때 적용된다. 단위 프레임 수는 각 기준 단위에서부터 출력할 프레임 수를 의미한다. 기본값은 0이며, 기본 FPS 에 따라 프레임을 출력한다. 예) no_of_decoding_frames_per_video_key_frame = 0
mod_mmf_filename	오디오/비디오 필터 라이브러리의 파일명을 설정한다. 예) mod_mmf_filename = libmodmmf.so
mod_imgf_filename	이미지 필터 라이브러리의 파일명을 설정한다. 예) mod_imgf_filename = libmodimgf.so
min_duration_for_video_decode	비디오 디코딩이 가능한 파일의 최소 재생 시간을 설정한다. 설정 값 미만인 재생 시간을 가진 파일은 디코딩 실패한다. 설정 단위는 초이며, 기본값은 5 초이다. 예) min_duration_for_video_decode = 5
min_duration_for_audio_decode	오디오 디코딩이 가능한 파일의 최소 재생 시간을 설정한다. 설정 값 미만인 재생 시간을 가진 파일은 디코딩 실패한다. 설정 단위는 초이며, 기본값은 5초이다. 예) min_duration_for_audio_decode = 5
max_duration_for_video_decode	비디오 디코딩 시간을 제한한다. 설정 값 이하로 디코딩을 수행한다. 설정 단위는 초이며, 생략하면 제한하지 않는다. 예) max_duration_for_video_decode=10
max_duration_for_audio_decode	오디오 디코딩 시간을 제한한다. 설정 값 이하로 디코딩을 수행한다. 설정 단위는 초이며, 생략하면 제한하지 않는다.

구성 부분	설명
	예) max_duration_for_audio_decode= 10
frame_width_for_video_decode	비디오 디코딩 시 디코딩되는 화면 넓이를 설정한다. 설정 단위는 픽셀이며, 생략하면 160 이다. 예) frame_width_for_video_decode=160
frame_height_for_video_decode	비디오 디코딩 시 디코딩되는 화면 높이를 설정한다. 설정 단위는 픽셀이며, 생략하면 120 이다. 예) frame_height_for_video_decode=120

5.4.15.4. 로그 파일

로그 파일은 필터 모듈의 수행 중에 발생한 에러에 대한 이력 정보가 기록되는 파일이다. 로그 파일의 내용은 (1) 로그 기록 시간, (2) 명령 타입, (3) 소스 파일 경로, (4) 에러 메시지로 구성되며, 로그 파일의 출력 예는 아래와 같다.

```
[Tue Dec 25 15:35:15 2007]                                          (1)
COMMAND<261>                                                        (2)
/data1/ucc_video/video/20070925/22/20070819160800226988391001.flv(3)
ERROR: can not find codec parameters [err=-1] (C02036)              (4)

[Tue Dec 25 15:35:28 2007]
COMMAND<261>
/data1/ucc_video/20070925/19/20070719160700201246381001.flv
ERROR: can not find codec parameters [err=-1] (C02036)

..... (계속)
```

5.4.16. 개인정보 추출/차단 (EXTRACT PERSONAL INFORMATION)

개인정보 추출/차단 모듈은 주민번호, 전화번호, 이메일, 카드번호 등 개인의 중요한 정보를 추출해 내거나, 개인정보를 외부에 노출되지 않도록 마스킹하기 위해 사용된다. 검색 단계에서와 색인 단계에서 개인정보를 마스킹할 수 있다.

모듈의 구성 요소는 다음과 같다.

구성 요소	설명	비고
kql236.mod	실질적인 요청을 처리하는 데몬 모듈	필수
prv-conf.domain.rc	MOD-PRV의 환경설정 파일	필수
kql233.mod	USERDEF 색인을 위한 모듈	필수

개인정보 추출/차단 모듈 설정파일은 prv-conf.domain.rc이며 설정값은 다음과 같다.

파라미터	설명
source_field_name	USERDEF 색인 시에만 사용되는 설정으로, 테이블의 전체 필드 중에서 어떤 필드의 데이터를 원본 소스로 이용할 것인지 그 필드 이름을 명시. 예) `source_field_name`
extract_resident_registration_number	주민번호를 추출할 것인지 여부. 0 또는 1의 값을 가짐. (기본값=1). 예) `extract_resident_registration_number = 0`
extract_credit_card_number	신용카드번호를 추출할 것인지 여부. 0또는 1의 값을 가짐. (기본값=1). 예) `extract_credit_card_number = 0`
extract_phone_number	전화번호(유선전화, 휴대전화)를 추출할 것인지 여부. 0 또는 1의 값을 가짐. (기본값=1). 예) `extract_phone_number = 0`
extract_email_address	전자우편주소(E-mail address)를 추출할 것인지 여부. 0 또는 1의 값을 가짐. (기본값=1).

파라미터	설명
	예) `extract_email_address = 0`
extract_bank_account_number	은행계좌번호를 추출할 것인지 여부. 0 또는 1의 값을 가짐. (기본값=1). 예) `extract_bank_account_number = 0`
max_entity_count	추출되는 개인정보 개수의 최대값을 설정함. (기본값=100). 예) `max_entity_count = 1000`
entity_separator	USERDEF 색인 시에만 사용되며, 추출한 개인정보 entity의 구분자(character type)를 설정. (기본값=','). 예) `entity_separator = /`

설정 파일의 작성 예는 다음과 같다.

```
source_field_name = f

extract_resident_registration_number = 1
extract_credit_card_number = 1
extract_phone_number = 1
extract_email_address = 1
extract_bank_account_number = 1

max_entity_count = 500
entity_separator = /
```

검색 결과 화면에서 개인정보 보호

색인 단계에서는 여과 없이 그냥 본문 전체를 색인 한 후, 검색 결과를 화면에 노출 시키는 단계에서만 개인정보가 보이지 않도록 하려면, '요약/하이라이팅(SUMMARIZATION/HIGHLIGHT)'에서 개인정보 마스킹 옵션을 사용하면 된다.

다음과 같이 레코드가 들어 있는 경우,

```
kql/myvol>> select * from tbl;
---------- 0 of total 1 (ROWID 0) ---------
fd (1048): 1 검색그룹팀 홍길동 123-1234 011-999-9999 hong@konantech.com
3 검색서비스팀 이몽룡 111-2222 010-234-2345 aaaa@konantech.com
18 검색개발팀 성춘향 333-4444 011-9999-2345 abcd@konantech.com 미등록
31 경영지원팀 이도령 777-7777 011-9999-9999 tttt@konantech.com
주민번호(O) : 340723-1555555, 000101-4033333
        (X) : 991323-1333333, 220432-2111111, 660325-5222222
```

결과를 하이라이팅 할 때 개인정보를 마스킹하도록 하려면 stx-conf.rc에 다음과 같이 세팅하면 된다.

stx-conf.rc 파일의 설정 예는 다음과 같다.

```
use_mask_private_information = 1
prv_domain_no = 0
prv_masking_character = *
```

결과는 다음과 같이 개인정보가 마스킹 되어 출력된다.

```
[0](1076) 1 검색그룹팀 홍길동 ******** ************ ***************
3 검색서비스팀 이몽룡 ******** ************ **********************
18 검색개발팀 성춘향 ******** ************ ***************** 미등록
31 경영지원팀 이도령 ********* ************* ******************
주민번호(O) : **************, **************
        (X) : 991323-1333333, 220432-2111111, 660325-5222222
```

색인 단계에서 개인정보 추출 또는 차단

색인 단계에서 미리 개인정보를 차단하여 아예 검색 자체가 되지 않도록 하거나, 본문에서 개인정보만을 따로 추출하여 별도 필드에 저장하는 방법은 MOD-UDT를 이용하여 USERDEF 색인을 하는 것이다.

USERDEF 색인을 하기 위해서는 data 디렉터리에 MOD-UDT(kql233.mod)가 있어야 하고 테이블 정의에 다음과 같이 USERDEF CONSTRAINT를 정의한다.

테이블 정의의 예는 다음과 같다.

```
create volume myvol;

use volume myvol;

create table tbl (
    fd text null,
    fd2 string list null,
    fd3 text null,

    constraint c1 TAG(fd2) = USERDEF kql233.mod(MOD_PRV_EX.0)
    constraint c2 TAG(fd3) = USERDEF kql233.mod.1(MOD_PRV_MK.0)
);
```

위 예에서 MOD_PRV_EX는 개인정보를 추출하여 f2 필드에 넣는 것이고, MOD_PRV_MK는 본문에서 개인정보가 있는 부분만을 마스킹하여 f3에 넣는다.

색인 결과는 다음과 같다.

```
kql/myvol>> select * from tbl;
---------- 0 of total 1 (ROWID 0) ---------
fd (1048): 1 검색그룹팀 홍길동 123-1234 011-999-9999 hong@konantech.com
3 검색서비스팀 이몽룡 111-2222 010-234-2345 aaaa@konantech.com
18 검색개발팀 성춘향 333-4444 011-9999-2345 abcd@konantech.com 미등록
31 경영지원팀 이도령 777-7777 011-9999-9999 tttt@konantech.com

fd2 (539): 123-1234,011-999-9999, hong@konantech.com, 111-2222,
010-234-2345, aaaa@konantech.com, 333-4444, 011-9999-2345,
abcd@konantech.com, 777-7777, 011-9999-9999, tttt@konantech.com

fd3 (1048): 1 검색그룹팀 홍길동 ******** ************ **************
3 검색서비스팀 이몽룡 ******** ************ **********************
18 검색개발팀 성춘향  ******** ************ ***************** 미등록
31 경영지원팀 이도령 ********* ************* ******************
```

위 결과에서 fd2 필드의 값은 추출된 개인정보를 정해진 entity_separator로 구별한 string list 데이터임을 확인할 수 있다. 색인 시 kql.rc에 설정된 string_list_separator와 prv-conf.domain.rc에 설정된 entity_separator가 일치하도록 주의한다.

5.5. 유틸리티

5.5.1. CRZCLI

시나리오를 이용한 검색 결과를 확인하는 기능을 한다. 즉, 검색 서버에 검색질의어를 보낸 후 결과를 받아 보여주는 기능을 한다.

- 사용법

```
crzcli {server port | -f file} scenario [ start count]
```

구성요소	설명
server port	요청할 검색 서버(docruzerd)의 주소와 포트
-f file	서버와 포트를 지정하는 대신 독크루저 설정 파일을 지정하는 경우에 사용
scenario	검색질의어 대상이 되는 시나리오
start	start offset
start count	받아올 최대 검색 결과 수

- 예제

```
$ crzcli 192.168.1.233 7577 actor
>name_idx='우주' allword
질의어: 'name_idx='우주' allword', 검색문서수: 1, 검색시간: 0 msec
+----- 1/1 번째 문서 (2) SCORE : 9998 -----+
[0](1) 2
[1](11) 다코타 패닝
[2](1) 1
[3](8) 우주전쟁
[KQS] [0]
[CLS] 0 개의 클러스터가 검색되었습니다.
>
$
```

5.5.2. REPLAY

검색 로그 파일의 질의어들을 검색 서버에 보내는 기능을 한다.

- 사용법

```
replay server port { logfile | stdin } [ -n thread_count]
[ -t timeout] [ -c record_count] [qps]
```

구성요소	설명
server port	요청할 검색 서버(docruzerd)의 주소와 포트를 지정한다.
logfile	사용자 질의어가 담긴 검색 로그 파일로 logs 디렉터리에 년-월-일-00.log 형식으로 존재한다. 검색부하 테스트를 하는 용도로 사용 가능하다.
stdin	로그파일과 같은 형식으로 표준입력으로 받아들이고자 하는 경우 stdin을 사용하면 된다.
thread_count	스레드 개수이며 여러 스레드에서 동시에 검색 요청을 보내고 싶을때 사용한다.
timeout	타임아웃 설정 옵션으로 검색 시 타임아웃을 설정할 수 있다.
record_count	요청할 레코드 건수로 기본 건수는 10 건으로 지정되어 있다.
qps	query per second의 약자로 1 초에 보내는 질의수를 조절하고 싶을 때 사용한다. 5로 지정하면 초당 최대 5 개의 질의어를 서버에 보내게 된다.

- 예제

```
$ replay 192.168.1.233 7577 20050720-00.log

CTRL+C
$
```

5.5.3. CCHTUNE

검색 캐시 기능을 사용하면 어느 정도의 hit ratio가 발생하는지 시뮬레이션 기능을 한다. 테스트 시에는 검색 캐시 엔트리 사이즈와 캐시 에이지를 변화시켜 가면서 적절한 최적의 값을 찾아야 한다.

- 사용법

```
cachetune -s size -a age -l line -f input_file
```

구성요소	설명
size	검색 캐시 엔트리 사이즈이다.
age	검색 캐시에 보유하고 있는 초 단위 시간을 지정하는 항목이다. 결과 리프레시가 되어야 하는 시간을 감안해 적당한 시간으로 지정하도록 한다. 너무 큰 숫자를 지정하면 검색 결과가 리프레시되는 시간이 많이 늦어질 수 있기 때문이다.
line	통계 결과를 표시할 질의어의 개수 단위이다.
input_file	캐시 효율을 테스트해 볼 검색 로그 파일이다.

- 예제

다음의 모의 테스트 예를 보면 캐시를 사용하게 될 때의 hit ratio는 0.232, 성능 향상은 26.98 %가 예상됨을 알 수 있다.

```
$ cchtune ../data/cch-conf.0.rc –l 1 –f
../logs/search/20100415-00.log

20050223:004014 CACHE: usage=1025, collision=36205, store=37511,
lookup=50000, hit=12489, miss=37511
20050223:012604 CACHE: usage=1025, collision=72046, store=74765,
lookup=100000, hit=25235, miss=74765
20050223:022938 CACHE: usage=1025, collision=98245, store=111852,
lookup=150000, hit=38148, miss=111852
20050223:042923 CACHE: usage=1025, collision=105231, store=151030,
lookup=200000, hit=48970, miss=151030
20050223:084952 CACHE: usage=1025, collision=106981, store=192587,
lookup=250000, hit=57413, miss=192587
20050223:100054 CACHE: usage=1025, collision=131513, store=231811,
lookup=300000, hit=68189, miss=231811
20050223:105202 CACHE: usage=1025, collision=166948, store=270248,
lookup=350000, hit=79752, miss=270248
20050223:112952 CACHE: usage=1025, collision=205016, store=308580,
lookup=400000, hit=91420, miss=308580
```

```
20050223:120633 CACHE: usage=1025, collision=242715, store=346371,
lookup=450000, hit=103629, miss=346371

------------------ STATISTICS ------------------
max cache size = 2048
max cache age = 180

CACHE: usage=2049, collision=355753, store=459704, lookup=598643,
hit=138939, miss=459704
       hit ratio = 0.232

search time per request(no cache) = 40.69 ms
search time per request(with cache) = 29.71 ms
performance advantage = 26.98 %
$
```

Appendix A.

부록A에서는 간단한 FAQ와 독크루저를 이용한 검색 서비스 구현 방법에 대해 설명한다.

A.1. FAQ

A.1.1. 독크루저는 어떤 환경에서 사용하면 좋습니까?

대량의 데이터를 다루는 모든 시스템에서는 이제 검색 기능이 선택이 아닌 필수가 되고 있습니다. 따라서 독크루저는 검색이 필요한 거의 모든 환경에서 편리하고 정확한 검색이 필요할 때 사용하면 좋습니다.

A.1.2. 독크루저는 어떻게 구성되어 있습니까?

독크루저는 하부 저장을 관리하는 저장 시스템, 각종 언어 데이터를 분석하는 색인 시스템, DBMS 등 외부 자료원으로부터 데이터를 끌어오는 게이트웨이 및 와처, 사용자 질의어를 분석하고 정확한 자료를 찾아주는 검색 시스템, 동의어, 유의어, 요약, 분류, 추천을 담당하는 지식관리 시스템으로 구성되어 있습니다.

A.1.3. 독크루저와 타 검색엔진과의 차별화 되는 요소는 무엇입니까?

- 독크루저는 고성능 언어 분석 기술과 대용량 데이터 저장 기술을 이용한 인터넷 검색포털 - 네이트를 통해 입증된 최강의 대용량, 초고속, 고품질 검색 솔루션입니다.

- 전용 DBMS를 이용한 데이터의 자유로운 검색을 지원하는 차세대 검색 솔루션입니다.

- 투명한 설계를 바탕으로 각 모듈들이 최상의 성능을 발휘하도록 작성되었기 때문에 시스템이 필요한 다양한 상황에서 언제든지 최적의 커스터마이징이 가능합니다.

A.1.4. 독크루저 Architecture 구성의 특장점은 무엇입니까?

- 독크루저 엔진의 저장 시스템은 현재 볼륨(=단위 데이터베이스당)12 TB, 20 억 문서를 설계 용량으로 하고 있으며 멀티볼륨을 지원하므로 사실상 무한대의 저장 용량 지원이 가능합니다.

- 독크루저 엔진은 인덱스를 압축하여 저장함으로써, 일반 검색의 경우에는 원문의 50 %, 자연어 검색을 지원하는 경우에는 80~90 %의 저장 공간만으로 인덱스를 압축하여 저장하기 때문에 디스크 요구량이 다른 엔진에 비해 훨씬 적다는 장점이 있습니다.

- 독크루저는 다양한 컴포넌트들이 코어 엔진에 조화롭게 결합되어 점진적으로 엔진의 기능 및 성능을 확장시켜 나갈 수 있는 구조이므로, 다양한 기간 시스템과의 연계 등에 매우 유리한 구조입니다.

- 독크루저는 서버 자원의 이용과 흐름을 최적화할 수 있도록 멀티스레드 구조로 설계되어 동급의 하드웨어에서도 훨씬 더 빠른 응답시간과 높은 처리율을 보입니다.

- 소스 코드 및 지식 자료 등의 원천 기술이 거의 100 % 자체 기술로 만들어졌으므로 향후 지속적인 기술 개량이 가능합니다.

A.1.5. 기존의 데이터베이스와는 어떻게 연결, 작동 됩니까?

독크루저는 기존의 데이터베이스 시스템들을 지원하기 위해 게이트웨이 및 와처를 포함하고 있습니다. 각각의 데이터베이스 시스템들은 나름대로의 특징이 있기 때문에 이것에 맞추어 만들어진 게이트웨이를 통해 검색엔진으로 데이터를 끌어오며, 변경되는 데이터들은 주기적으로 실행되는 와처를 통해 검색엔진에 반영됩니다.

A.1.6. 연동 가능한 데이터베이스에는 어떤 것이 있습니까?

현재 Oracle, MS SQL Server, Sybase, DB2, Notes, MySQL과 연동됩니다. (Informix는 지원할 예정)

A.1.7. Oracle DB에 저장된 테이블을 검색하려면?

독크루저는 Oracle 게이트웨이를 지원하기 때문에 간편하게 Oracle 데이터베이스와 연동할 수 있습니다. 독크루저의 테이블과 Oracle의 테이블 간 필드 맵핑만 정의해주면 됩니다.

A.1.8. 하드디스크에 저장된 업무 문서를 검색하려면?

독크루저의 파일시스템 게이트웨이와 문서 필터링 기능을 이용하시면 됩니다.

A.1.9. 툴킷의 용도는 무엇입니까?

고객들은 다양한 환경에서 검색엔진을 필요로 합니다. 따라서 검색엔진은 새로 구축하거나 이미 구축되어 있는 시스템에서 고객이 원하는 방식으로 작동되도록 지원할 수 있어야 합니다.

툴킷은 크게 KQL 툴킷과 독크루저 툴킷으로 제공되는데, KQL 툴킷은 보다 전문적인 로우 레벨에서 작업이 필요할 때 제공되고, 독크루저 툴킷은 단순한 검색 기능만 필요로 할 때 제공되는 라이브러리입니다.

A.1.10. 자연어 검색은 어떻게 이루어집니까?

자연어 검색은 원하는 정보를 얻기 위해 사용자가 가지게되는 부담을 검색엔진이 대신 해결해 주는 기술입니다. 그러므로 자연어 검색은 일반 문장으로 된 사용자 질의어를 분석하여 내부적으로 최적의 결과를 얻기 위한 조건식으로 변환하여 검색을 합니다.

A.1.11. 자연어 검색의 장점은 무엇입니까?

대부분의 사용자들은 복잡한 검색에 익숙하지 않습니다. 따라서 일상적으로 사용하는 언어로 질의하고 싶어합니다. 독크루저는 뛰어난 언어처리 기술을 바탕으로 최상의 자연어 검색 기능을 보장합니다.

A.1.12. 유사어 검색이란 무엇입니까?

여러 코퍼스로부터 각 단어들간의 연관성을 학습한 후 검색 시 이 정보를 바탕으로 사용자가 미처 생각하지 못했을 법한 질의어를 추천해 줍니다. 사용자는 추천된 질의어 리스트 중 원하는 것을 선택하여 다시 검색합니다.

A.1.13. 오타자 검색도 가능합니까?

오타 발생시 올바른 단어로 추천해주는 기능이 있습니다. 한/영 키가 잘못 눌린 상태에서 검색어를 입력한 경우에도 한/영 오타 교정 기능을 사용해 올바른 단어를 추천해 줄 수 있습니다.

A.1.14. 질의어에 대한 결과물의 순서는 어떻게 결정되는 것입니까?

사용자 질의어를 바탕으로 단어들 간의 출현빈도, 근사도, 정보량등 다양한 기준을 종합적으로 적용하여 순서를 매깁니다. 옵션에 따라서는 문서가 가지는 고유한 중요도 점수를 정렬에 반영할 수도 있고, 각 필드별로 정렬을 할 수도 있습니다.

A.1.15. 독크루저를 응용할 수 있는 분야에는 어떤 것이 있습니까?

- 전자 상거래를 위한 제품 카탈로그 검색
- 온라인 검색 포탈
- 업무용 인트라넷 문서 검색
- 전자도서관을 위한 서지/전문 기반 도서 검색

A.1.16. 지원 가능한 플랫폼에는 어떤 것이 있습니까?

현재 지원 가능한 하드웨어 및 운영 체제는 Windows 2000/2003, Intel Linux, Sun Sparc Solaris, IBM RS6000 AIX, HP HPUX, Silicon Graphics SGI IRIX입니다. 기타 플랫폼은 필요 시에 지원가능합니다.

A.1.17. 검색 서비스 구축을 위한 하드웨어의 적정 규격은 어떻게 됩니까?

중소 규모의 시스템은 2 CPU, 2 GB 메모리, 50 GB RAID를 갖춘 서버(예: Compaq ML570)를 사용하고, 부하가 많이 걸리는 대규모 시스템에서는 4 CPU, 4 GB 이상, 100 GB 이상의 RAID를 갖춘 서버(예: Compaq PL8000)를 추천합니다.

A.1.18. 로드 밸런싱은 어떻게 구성해야 합니까?

사용자는 동시다발적으로 여러 웹서버에 접속하고 경우에 따라 이들 웹서버를 통해 검색엔진에 검색 요구를 보냅니다. 한편, 독크루저는 자신과 동일한 replica 서버를 여러 대 관리합니다. 코난 로드밸런서(KLB)는 replica 서버들의 처리 상황을 모니터링 하면서 부하가 가장 적게 걸린 서버에게 사용자의 요구를 보냅니다.

A.1.19. 독크루저의 언어 처리 기술의 특징들은 무엇입니까?

독크루저는 고품질 자연어 분석 기술에 기반하여 사용자가 이용하기 가장 간편한 형태의 질문의 문장 형태 질의를 지원하며, 세계 최초의 복합명사 처리, 띄어쓰기 오류 자동 수정 기능 및 관련어 추천기능. 동의어 시소러스 검색 지원, 대용량 사전 등의 강력한 언어 처리 기능들을 제공 하고 있습니다.

A.1.20. 독크루저의 다국적 언어 지원의 특장점은 무엇입니까?

한국어, 중국어, 일본어 질의에 대해 해당 언어로 된 문서를 검색해 줍니다.

A.1.21. 기존 데이터베이스나 기간 시스템과의 연동 방법은 어떻게 됩니까?

독크루저는 기존 환경과 쉽게 통합할 수 있도록 해주는 각종 API(Multi-Threaded API, 인덱싱 API, 검색 API)를 지원합니다. 이를 활용하여 기존 환경에 쉽게 검색엔진 서비스를 채용, 적용할 수 있습니다.

A.1.22. 사내 KMS 등에 있는 HWP, DOC, PDF 등의 문서 검색이 가능합니까?

각종 문서작성기로 작성된 문서파일이나 웹 문서 파일은 일정한 형식으로 데이터를 구성하고 있어서 데이터를 바로 사용할 수 없으므로, 색인에 적합한 실제 데이터만을 추출하는 필터링(Filtering) 작업을 수행한 후에 색인을 해야 합니다. 독크루저는 HTML, HWP, DOC, EXCEL, PDF 등 다양한 필터를 지원합니다.

A.1.23. 독크루저의 검색 순위를 사용자 요구에 맞게 조정할 수도 있습니까?

예, 가능합니다. 독크루저는 기본적으로 질의어와의 유사도 순으로 정렬하지만 사용자의 요구에 맞게 검색 방법이나 정렬조건, 기타 옵션들을 설정하여 검색 결과를 조정할 수 있습니다.

A.1.24. 고객의 쇼핑 및 구매 행태에 기반한 개인 맞춤 기능을 구현할 수 있습니까?

예, 가능합니다. 고객이 사이트에 접속하여 하는 모든 행동 (자주 보는 페이지, 구매내력 등)을 모아 개인 프로파일을 만들고 이를 토대로 다음 번 접속했을 때 그 고객이 원하는 정보를 좀 더 쉽게 얻을 수 있도록 물품의 보여주는 우선순위를 조정할 수도 있고, 그 고객과 유사한 행동을 보이는 다른 고객의 정보를 이용하여 좋아할 것으로 예상되는 새로운 물품을 추천할 수도 있습니다.

A.1.25. 검색 서비스를 구축하는 데는 어느 정도의 인력과 시간이 필요합니까?

독크루저는 사용이 간편한 범용 검색 서버뿐 아니라 클라이언트 개발을 위한 라이브러리를 제공하고 있고, 다양한 상황의 샘플 검색 시스템도 유지하고 있습니다. 따라서 간단한 검색 시스템을 구축할 경우는 신속하게 일을 끝마칠 수 있습니다. 그러나 대개 검색만 단독으로 설치되기 보다는 다른 시스템과 연동되기 때문에 프로젝트로 진행되며 상황에 따라 1 주일에서 1 달 정도 소요됩니다.

A.1.26. 업그레이드나 확장에 필요한 하드웨어 및 비용은 어느 정도입니까?

검색 성능 확장 방법은 다음과 같습니다.

- 현 검색 서버의 CPU 및 메모리 증설

  CPU 및 메모리 등 하드웨어의 증설 후 별도의 병렬 CPU 지원 팩을 구입해서 독크루저의 용량을 증설해야 합니다.

- 검색 서버의 증설

서버를 증설한 후, 증설된 서버 수만큼의 독크루저를 구입하여 설치한 후 로드 밸런싱용 스위치와 미들웨어를 구입하여 멀티 서버 구성을 지원받아야 합니다.

A.1.27. 선호하는 특정 제품의 서버를 사용하는 것도 가능합니까?

독크루저는 대부분의 상용 UNIX 서버용 버전과 Windows용 버전을 제공하고 있습니다. 자주 쓰이지 않는 특별한 하드웨어에 대해서는 별도의 포팅 작업을 통해 지원할 수 있습니다.

A.1.28. 코난테크놀로지의 제품을 선택해야 하는 이유는 무엇입니까?

코난테크놀로지의 제품은 다른 제품들에 비해 속도, 성능, 질적인 면 모두에서 앞서 있습니다. 이는 곧 검색 서비스를 구축하는 데 하드웨어 비용을 절감하고 고객들의 만족도를 향상시킬 수 있다는 것입니다. 또한 한, 중, 일, 영어 지원 및 각종 DBMS, 문서 포맷을 지원하기 때문에 앞으로의 확장을 고려할 때에도 코난테크놀로지의 제품을 사용하는 것이 좋습니다.

A.2. 구현 예제

A.2.1. 기본 데이터 사용 예

기본 설치 환경의 구성 파일이다. 다음은 KQL 설정 파일과 볼륨 설정 파일이다. 'listener'와 'product_key' 항목이 제대로 입력되어있는지 확인한다.

```
 ================== kql.rc 설정 ==================
/*************************************************************
**                                                          **
**            DATABASE CONFIGURATION FILE                   **
**                        FOR                               **
**                  KONAN TEXTSEARCH                        **
**                                                          **
*************************************************************/
;;;;;;;;;; GENERAL DATABASE CONTROL PARAMETERS ;;;;;;;;;;;;;;
```

```
max_no_of_sessions = 100
max_no_of_volume_handles = 500
max_no_of_table_handles = 500
max_no_of_open_files = 100

max_buffer_cache_size = 256 MB
max_working_memory_size = 256 MB

no_of_parallel_task = 4
data_location = ../data

;sys_log = kql.log
sys_log = console

;volume = tv, "vol.rc"

listener = 192.168.1.233, 6333
product_key = YIR8CAB-RPVKJS-VBDERR-QTSBLB

volume = tv, "vol.rc"

================== vol.rc 설정 ==================
volume_device = "../volume"
log_device = "../volume"
temp_device = "../volume"
```

다음은 독크루저 설정 파일과 시나리오 파일이다.

```
================== docruzerd.rc 설정 ==================
;
; DOCRUZER SYSTEM CONFIGURATION
;

;;;;;;;;;;;;;;;; LOGGING ;;;;;;;;;;;;;;;;;;;;;;;;
;log_file_location = console
log_file_location = ../logs
;sys_log_file_location = console
sys_log_file_location = ../logs
;no_of_log_file_per_day = 2
;no_of_sys_log_file_per_day = 1
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;;; MONITORING ;;;;;;;;;;;;;;;;;;;;;;
;max_search_cache_size = 32768
;max_search_cache_age = 60
;trace_socket_option = 1
;log_system_status = 10
;auto_restart = 300; 5*60=300
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

daemon_service_port = 7577

kql_link = h1, kql.rc

include scn.rc

$ cat scn.rc

scenario nr1
{
         link = h1
         volume = tv
         table = GISA
         field = TT (256, "<font color=#FF0000>", "</font>")
         field = BD (4096, "<font color=#FF0000>", "</font>")
         field = DT
         field = AU
         field = SC
         field = SO
}
```

다음은 제공되는 뉴스 데이터로 색인을 하는 경우이다. KQL을 실행하여 색인 스크립트인 run.kql을 실행하도록 한다.

```
$ ./kql -f kql.rc
Welcome to KQL Interpreter (Version 3.5.2 Linux i686).
Type 'help' for help. Commands end with ';'.
Copyright (c) 1999-2011 Konan Technology, Inc.

kql> show volumes;
+-------------+--------+---------+
| VOLUME NAME | STATUS | VERSION |
```

```
+-------------+--------+---------+
|          tv |     OK | 3.5.2   |
+-------------+--------+---------+
Total 1 volume.
OK

kql> run run.kql;
[2011-01-25 11:14:40] M0/0  tv Write Exclusive requested.
[2011-01-25 11:14:40] M0/0  tv 25 buffers deactivated (400 KB,
0 % of total 16384).
[2011-01-25 11:14:40] M0/0  tv Write Exclusive requested.
[2011-01-25 11:14:40] M0/0  tv TR1 started [disk=(2,224)
op=Q:CREATE_TABLE].
[2011-01-25 11:14:40] M0/0  108 pages updated (6 logged + 102 extra
= 108)
[2011-01-25 11:14:40] M0/0  tv TR1 ended [2 1.15 1.117 1.3583
6 0.1 MB].
[2011-01-25 11:14:40] M0/0  tv Write Exclusive requested.
[2011-01-25 11:14:40] M0/0  tv TR2 started [disk=(2,224)
op=Q:CREATE_INDEX].
[2011-01-25 11:14:40] M0/0  3 pages updated (3 logged + 0 extra = 3)
[2011-01-25 11:14:40] M0/0  tv TR2 ended [3 1.117 1.117 1.3583
3 0.0 MB].
[2011-01-25 11:14:40] M0/0  tv Write Exclusive requested.
[2011-01-25 11:14:40] M0/0  tv TR3 started [disk=(2,224)
op=Q:CREATE_INDEX].
[2011-01-25 11:14:40] M0/0  3 pages updated (3 logged + 0 extra = 3)
[2011-01-25 11:14:40] M0/0  tv TR3 ended [4 1.117 1.117
1.3583 3 0.0 MB].
[2011-01-25 11:14:40] M0/0  tv Write Exclusive requested.
...
[2011-01-25 11:14:40] M0/0  tv TR9 started [disk=(2,224)
op=Q:CREATE_GATEWAY].
[2011-01-25 11:14:40] M0/0  2 pages updated (2 logged + 0 extra = 2)
[2011-01-25 11:14:40] M0/0  tv TR9 ended [10 1.117 1.117 1.3583
2 0.0 MB].
[2011-01-25 11:14:40] M0/0  tv Read Exclusive requested.
[2011-01-25 11:14:40] M0/0  tv Write Shared requested.
[2011-01-25 11:14:40] M0/0  tv TR10 started [disk=(2,224)
op=Q:IMPORT_FILES].
```

```
Warning: import process underloaded (4 tasks on 8 CPUs).
[2011-01-25 11:14:40] M0/0  ===== START OF STORE (/tv/GISA) ======
[2011-01-25 11:14:40] A3/0             0 [0:00:00]   2.7 MB
[2011-01-25 11:14:40] M0/0          2498 [0:00:00]   5.8 MB
[2011-01-25 11:14:40] M0/0  ===== END OF STORE (/tv/GISA) ========
[2011-01-25 11:14:40] M0/0  Document Collection = ../rawdata.
[2011-01-25 11:14:40] M0/0  Total 5 files (3.0 MB) stored.
[2011-01-25 11:14:40] M0/0  Disk Space Used = 5.8 MB (2 Segments)
[2011-01-25 11:14:40] M0/0  214 pages updated (16 logged
+ 198 extra = 214)
[2011-01-25 11:14:40] M0/0  tv TR10 ended [11 1.117 1.315 1.3583
16 0.2 MB].
[2011-01-25 11:14:40] M0/0  tv Write Shared requested.
[2011-01-25 11:14:40] M0/0  tv TR11 started [disk=(2,224)
op=Q:BUILD_INDEX].
[2011-01-25 11:14:41] M0/0  === START OF INDEXING (/tv/GISA) ===
[2011-01-25 11:14:42] M0/0  Started field  0 (TT) indexing
[2011-01-25 11:14:43] A0/0      0/2499 0:00:00 [0:00:02]  5.9MB  0%
[2011-01-25 11:14:43] A0/0   2498/2499 0:00:00 [0:00:02]  5.9MB  3%
[2011-01-25 11:14:44] M0/0   9681 2499 0:00:01 [0:00:02]  7.9MB
[2011-01-25 11:14:45] M0/0 Started field  1 (BD) indexing
[2011-01-25 11:14:46] A0/0      0/2499 0:00:00 [0:00:04]  7.9MB  3%
[2011-01-25 11:14:47] A2/0   2498/2499 0:00:01 [0:00:05]  8.4MB 97%
[2011-01-25 11:14:48] M0/0  73085 2499 0:00:02 [0:00:07] 14.4MB
[2011-01-25 11:14:49] M0/0 Started field  2 (DT) indexing
[2011-01-25 11:14:50] A1/0      0/2499 0:00:00 [0:00:08] 14.4MB 97%
[2011-01-25 11:14:50] A1/0   2498/2499 0:00:00 [0:00:08] 14.4MB 98%
[2011-01-25 11:14:50] M0/0   1818 2499 0:00:01 [0:00:09] 14.4MB
[2011-01-25 11:14:51] M0/0 Started field  3 (AU) indexing
[2011-01-25 11:14:52] A0/0      0/2499 0:00:00 [0:00:10] 14.4MB 98%
[2011-01-25 11:14:52] A1/0   2498/2499 0:00:00 [0:00:10] 14.4MB 98%
[2011-01-25 11:14:52] M0/0    255 2499 0:00:00 [0:00:11] 14.6MB
[2011-01-25 11:14:53] M0/0 Started field  4 (SC) indexing
[2011-01-25 11:14:54] A0/0      0/2499 0:00:00 [0:00:12] 14.6MB 98%
[2011-01-25 11:14:54] A0/0   2498/2499 0:00:00 [0:00:12] 14.6MB 99%
[2011-01-25 11:14:54] M0/0     11 2499 0:00:01 [0:00:13] 14.6MB
[2011-01-25 11:14:55] M0/0 Started field  5 (SO) indexing
[2011-01-25 11:14:56] A2/0      0/2499 0:00:00 [0:00:15] 14.6MB 99%
[2011-01-25 11:14:56] A2/0   2498/2499 0:00:00 [0:00:15] 14.6MB 99%
[2011-01-25 11:14:57] M0/0      9 2499 0:00:01 [0:00:16] 14.6MB
```

```
[2011-01-25 11:14:59] M0/0 == END OF INDEXING (/tv/GISA) ==
[2011-01-25 11:14:59] M0/0 Disk Space Used = 14.61 MB (2 Segments)
[2011-01-25 11:15:00] M0/0 658 pages updated (93 logged
+ 565 extra = 658)
[2011-01-25 11:15:00] M0/0 tv TR11 ended [12 1.315 1.880 1.3583
93 1.5 MB].
OK


kql/tv> show tables;
+------------+
| TABLE NAME |
+------------+
|       GISA |
+------------+
Total 1 table.
OK

kql/tv> explain GISA more;
Total 6 fields defined.
Total 0 constraint defined.
Total 7 indexes defined.
Total 1 gateway defined.
Total 2,499 records stored.
+------------+------------------+-------+-----+---------+
| FIELD NAME | TYPE             |   MAX | MIN |    MEAN |
+------------+------------------+-------+-----+---------+
|         TT | TEXT NULL        |    81 |   8 |   36.91 |
|         BD | TEXT NULL        | 17184 |   0 | 1148.79 |
|         DT | STRING NULL      |    14 |  14 |   14.00 |
|         AU | STRING NULL      |     6 |   0 |    2.14 |
|         SC | STRING LIST NULL |     8 |   0 |    4.08 |
|         SO | STRING NULL      |    12 |   7 |   10.00 |
+------------+------------------+-------+-----+---------+
+------------+------------+
| INDEX NAME | FIELD NAME |
+------------+------------+
|   TT_INDEX | TT         |
|   BD_INDEX | BD         |
|   DT_INDEX | DT         |
```

```
|   AU_INDEX | AU         |
|   SC_INDEX | SC         |
|   SO_INDEX | SO         |
| TTBD_INDEX | TT BD      |
+------------+------------+
+--------------+--------------------------+
| GATEWAY NAME | DEFINITION               |
+--------------+--------------------------+
|          GW1 | FIELD LINE STARTS WITH   |
|              | TT = "<__tt__>" (LITERAL) |
|              | BD = "<__bd__>" (LITERAL) |
|              | DT = "<__dt__>" (LITERAL) |
|              | AU = "<__au__>" (LITERAL) |
|              | SC = "<__sc__>" (LITERAL) |
|              | SO = "<__so__>" (LITERAL) |
+--------------+--------------------------+
OK

kql/tv> quit
Bye

$
```

독크루저 검색 데몬을 시작한 다음 설정되어 있는 시나리오(nr1)로 검색 요청을 한다. 다양한 검색 질의어를 보내고 결과를 살펴보도록 한다.

```
$ ./docruzerd start
>DOCRUZER: READY

$ ./crzcli 192.168.1.233 7577 nr1
>TT_INDEX = '축구' allword

질의어: W=[TT_INDEX='축구' allword], S=[]
바인딩 필드[6]: TT BD DT AU SC SO
검색문서수: 18 (row=10,col=6), (start=0, count=10) 검색시간: 129 msec
+----- 1/10 번째 문서 (47) SCORE : 9733 MATCHFIELD: -1 -----+
[0](71) [프로<font color=#FF0000>축구</font>-종합] 성남 골득실차
마지막 4강티켓
[1](1837) 안양 LG-성남 일화,수원 삼성-울산 현대가 아디다스컵 결승티켓을 놓고
오는 5일 한판 승부를 벌이게 됐다.
```

성남은 1일 부천종합운동장에서 열린 2002아디다스컵 조별리그 부천 SK와의 A조 최종 8차전에서 전후반과 연장전을 2-2로 비긴 뒤 승부차기에서 3-5로 졌다.

하지만 5승3패 승점 10점을 기록한 성남은 골득실차(+7)에서 부천(승점 10 -2)과 전북(승점 10 -2)에 앞서 간신히 조 2위로 4강대열에 합류했다. 아디다스컵 4년 연속 4강에 오른 성남은 지난 92년 아디다스컵 원년 대회 우승 이후 11년 만에 다시 패권을 거머쥘 수 있는 좋은 기회를 마련했다. 성남은 오는 5일 안양에서 A조 1위를 차지한 안양 LG와 단판제의 4강전을 갖는다.

부천은 이날 최소한 연장전에서 이겨야 승점 2점을 확보,총 11점으로 성남을 밀어내고 조 2위가 될 수 있었지만 아쉽게 승부차기승리에 만족해야 했다. 부천은 3위에 머물렀다.

0-0이던 후반 8분,성남은 올시즌 부천과의 개막전(6-0 승)에서 혼자 5골을 폭발시킨 '부천 킬러' 샤샤가 선제골을 뽑아냈다. 부천의 결정적 실수를 골로 연결시킨 것. 성남은 부천 최거룩이 GK에게 백패스한 볼을 황연석이 골지역 오른쪽에서 가로채 문전으로 센터링,이 볼을 샤샤가 텅빈 골문으로 밀어넣었다. 샤샤는 이 골로 올시즌 7골째를 기록,득점왕 선두자리를 굳게 지켰다. 이어 성남은 후반 19분,올리베의 센터링을 황연석이 다이빙 헤딩슛으로 연결시켜 2-0을 만들며 사실상 승부를 결정짓는 듯했다.

그러나 부천의 후반 반격은 무서웠다. 부천은 경기 종료 7분을 남기고 내리 2골을 뽑아내는 무서운 뒷심을 발휘했다. 후반 38분 이원식이 단독 돌파해 2-1을 만든 데 이어 후반 42분 최문식이 동점골을 만든 것.

이어진 연장전,두 팀은 필사의 골찬스를 만들었으나 승부를 가리지 못한 채 승부차기 끝에 부천이 5-3으로 이겼다.

한편 수원에서는 이미 4강행을 확정지은 수원이 포항에 1-2로 덜미를 잡혔으며, 안양에서는 안양과 울산이 전후반과 연장전에서 득점 없이 비긴 뒤 승부차기에서 울산이 4-3으로 힘겹게 이겼다. 부산 아이콘스는 홈경기에서 최광수 전우근의 연속골로 대전에 2-0으로 승리했다.
[2](14) 20020501224001
[3](0)
[4](6) 스포츠
[5](12) 스포츠투데이
+----- 2/10 번째 문서 (49) SCORE : 9733 MATCHFIELD: -1 -----+
[0](49) [ST축구기록실] 5월 1일
[1](203) ▲제35회 대통령배 전국고교대회

```
서울체 2<4 승부차기 1>2 중대부속

금호 1<5 승부차기 3>1 거제

경희 4-0 구리

광양제철 7-1 알로이시오

안동 7-3 재현

배재 2<4 승부차기 2>2 창원기공
[2](14) 20020501224001
[3](0)
[4](6) 스포츠
[5](12) 스포츠투데이
+----- 3/10 번째 문서 (627) SCORE : 9733 MATCHFIELD: -1 -----+
...
+----- 10/10 번째 문서 (751) SCORE : 9707 MATCHFIELD: -1 ----+
[0](67) 아프리카 11개 <font color=#FF0000>축구</font>협회, 하야투 지지
표명
[1](782) (나이로비=연합뉴스) 소말리아 등 아프리카대륙 11개
<font color=#FF0000>축구</font>협회가 국제<font color=#FF0000>축구
</font>연맹(FIFA) 차기 회장 선거에 출마한 이사 하야투에 대한 지지를 표명했다.
    11개국을 대표하는 동.중부아프리카<font color=#FF0000>축구</font>
협회평의회(CECAFA)의 파라 아도 의장은 31일(한국시간) "하야투는 FIFA 회장을
맡을 자격이 있는 인물"이라며 "지난 98년 FIFA회장 선거 때 제프 블래터를
지지했던 실수를 되풀이 해서는 안된다"라고 밝혔다.
    아도 의장은 "지금이 아프리카에는 절호의 기회"라며 "이 기회를  놓치면
많은 시간을 기다려야 한다"며 하야투를 지지해 줄 것을 회원국에 호소했다.
    한편 CECAFA에는 부룬디, 지부티, 에티오피아, 케냐, 르완다, 소말리아,
수단, 탄자니아, 우간다, 잔지바르, 에리트레아가 회원국으로 가입해 있다.
    cty@yna.co.kr
[2](14) 20020331110700
[3](0)
[4](4) 오늘
[5](8) 연합뉴스

>

$
```

KQL 설정 파일에 지정한 리스너 포트를 지정하여 독크루저 검색 데몬에 원격 접속한 후에 테이블 레코드를 업데이트, 삭제하는 등 매뉴얼 상의 테이블 관리 명령어들을 실습해보도록 한다.

```
$ ./kql -f 192.168.1.233:6333
Welcome to KQL Interpreter (Version 3.5.2 Linux i686).
Type 'help' for help. Commands end with ';'.
Copyright (c) 1999-2011 Konan Technology, Inc.
(Connected to a remote server 192.168.1.233:6333, link 2)

kql>> use volume tv;
OK

kql/tv>> select TT from GISA where TT_INDEX='차' allword;
---------------- 1 of total 13 (ROWID 607) --------------------
TT (42): 대우차 부산 버스공장, 영안모자서 인수할 듯
---------------- 2 of total 13 (ROWID 735) --------------------
TT (36): 대우차매각 본계약, 4월중순 체결될 듯
---------------- 3 of total 13 (ROWID 736) --------------------
TT (33): 12세이하 어린이 차앞좌석 못태운다
---------------- 4 of total 13 (ROWID 1009) --------------------
TT (30): 알덱스 "2차전지용 MH 특허출원"
---------------- 5 of total 13 (ROWID 1477) --------------------
TT (41): 오늘의 주요기사 1차메모(2월26일.화)(종합)
---------------- 6 of total 13 (ROWID 1577) --------------------
TT (32): "대우차 임단협 해결 기미"-채권단
---------------- 7 of total 13 (ROWID 1643) --------------------
TT (35): 오늘의 주요기사 2차메모(1월30일.수)
---------------- 8 of total 13 (ROWID 1688) --------------------
TT (29): 제2차 한.일 금융감독 연례회의
---------------- 9 of total 13 (ROWID 1870) --------------------
TT (36): 중기청, 제3차 벤처전문가 토론회 개최
---------------- 10 of total 13 (ROWID 2113) --------------------
TT (38): 현대•기아차,연료전지차 전략제휴(상보)
---------------- 11 of total 13 (ROWID 2232) --------------------
TT (36): 현대• 기아차,美업체와 전략제휴(1보)
---------------- 12 of total 13 (ROWID 2263) --------------------
TT (35): 르노삼성차,오토카페서 편의시설 제공
---------------- 13 of total 13 (ROWID 2457) --------------------
TT (34): 조폐공사, 월드컵 2차 기념주화 판촉
```

```
Total 13 records.
OK

kql/tv>> select TT from GISA where TT_INDEX='차 현대' allword;
---------------- 1 of total 2 (ROWID 2113) --------------------
TT (38): 현대•기아차,연료전지차 전략제휴(상보)
---------------- 2 of total 2 (ROWID 2232) --------------------
TT (36): 현대• 기아차,美업체와 전략제휴(1보)
Total 2 records.
OK

kql/tv>> update GISA set TT = '자동차 업계 제휴' where TT_INDEX =
'차 현대' allword;
Total 2 records.
OK

kql/tv>> select TT from GISA where $ROWID = 2113 or $ROWID = 2232;
---------------- 1 of total 2 (ROWID 2113) --------------------
TT (16): 자동차 업계 제휴
---------------- 2 of total 2 (ROWID 2232) --------------------
TT (16): 자동차 업계 제휴
Total 2 records.
OK

kql/tv>> delete from GISA where $ROWID = 2113;
Total 1 record.
OK

kql/tv>> select TT from GISA where $ROWID = 2113 or $ROWID = 2232;
---------------- 1 of total 1 (ROWID 2232) --------------------
TT (16): 자동차 업계 제휴
Total 1 record.
OK

kql/tv>> quit
Bye

$
```

'tv'라는 이름의 기존 볼륨을 검색 서비스 중인 볼륨으로 가정하고, 임시 볼륨을 하나 추가한 후 임시 볼륨에 색인을 해보도록 한다. 검색 서비스를 중단하지 않고 색인하기 위한 방법이다.

'temp'라는 이름으로 볼륨을 추가하고 run.kql을 변경하여 임시 볼륨에 색인한 후에 서비스 볼륨에 복사하는 방식으로 진행한다.

```
============== kql.rc 내용 ==============
...
sys_log = kql.log
;sys_log = console

volume = tv, "vol.rc"
volume = temp, "vol_temp.rc"

========== vol_temp.rc 내용 ==============
volume_device = "../volume.temp"
log_device = "../volume.temp"
temp_device = "../volume.temp"

/* ../volume.temp 디렉터리가 만들어져 있어야 함 */


============== run.kql ==============
create volume temp;
echo "volume created";

use volume temp;

create table GISA (
         TT TEXT NULL,
         BD TEXT NULL,DT STRING NULL,
         AU STRING NULL,
         SC STRING NULL,
         SO STRING NULL
);

echo "table is created.";

/* 인덱스 생성 */
```

```
create index TT_INDEX on GISA (TT);
create index BD_INDEX on GISA (BD);
create index DT_INDEX on GISA (DT);
create index AU_INDEX on GISA (AU);
create index SC_INDEX on GISA (SC);
create index SO_INDEX on GISA (SO);
create index TTBD_INDEX on GISA (TT, BD);

create gateway GW1 on GISA
to file system
format
        TT="<__tt__>",
        BD="<__bd__>",
        DT="<__dt__>",
        AU="<__au__>",
        SC="<__sc__>",
        SO="<__so__>";

echo "gateway installed.";

import records from file system to GISA through GW1
     select ../sample/rawdata *.txt;

echo "records are imported through gateway.";

swap volume tv temp;
echo "two volumes swapped.";

copy volume tv to temp;
echo "volume copied.";

swap volume tv temp;
echo "two volumes swapped.";
```

KQL 설정 파일을 새로 적용하기 위해서는 독크루저를 종료했다가 다시 기동해야 한다. 원격 접속하여 변경한 run.kql을 실행한 후 결과를 확인한다.

```
$ ./docruzerd stop
>DOCRUZER: SHUTDOWN

$ ./docruzerd start
```

```
>DOCRUZER: READY

$ ./kql -f 192.168.1.233:6333
Welcome to KQL Interpreter (Version 3.5.2 Linux i686).
Type 'help' for help. Commands end with ';'.
Copyright (c) 1999-2011 Konan Technology, Inc.
(Connected to a remote server 192.168.1.233:6333, link 2)

kql>> show volumes;
+-------------+-------------+---------+
| VOLUME NAME | STATUS      | VERSION |
+-------------+-------------+---------+
| tv          | OK          | 3.5.2   |
| temp        | Not Created |         |
+-------------+-------------+---------+
Total 2 volumes.
OK

kql>> run run.kql;
OK

kql/temp>> show volumes;
+-------------+--------+---------+
| VOLUME NAME | STATUS | VERSION |
+-------------+--------+---------+
| tv          | OK     | 3.5.2   |
| temp        | OK     | 3.5.2   |
+-------------+--------+---------+
Total 2 volumes.
OK

kql/temp>> quit
Bye
$
```

하루에 한 번씩 색인이 이루어지도록 스케줄을 등록한다.

```
$ cat kql.rc
...
schedule=batch, "run run.kql;", every 6:00
```

```
$ ./docruzerd stop
>DOCRUZER: SHUTDOWN

$ ./docruzerd start
>DOCRUZER: READY

$ ./kql -f 192.168.1.233:6333
Welcome to KQL Interpreter (Version 3.5.2 Linux i686).
Type 'help' for help. Commands end with ';'.
Copyright (c) 1999-2011 Konan Technology, Inc.
(Connected to a remote server 192.168.1.233:6333, link 2)

kql>> show schedules;
+----+-------+-------+--------+------+-----------+-------------+
-------------+
| NO |  NAME | SCHED | EXCEPT | STAT | COUNT(F,S) | ACTION       |
 POST ACTION |
+----+-------+-------+--------+------+-----------+-------------+
-------------+
|  0 | batch |  6:00 | -      |   ON |    0 (0,0) | run run.kql; |
             |
+----+-------+-------+--------+------+-----------+-------------+
-------------+
Total 1 schedules.
OK

kql>> quit
Bye
$
```

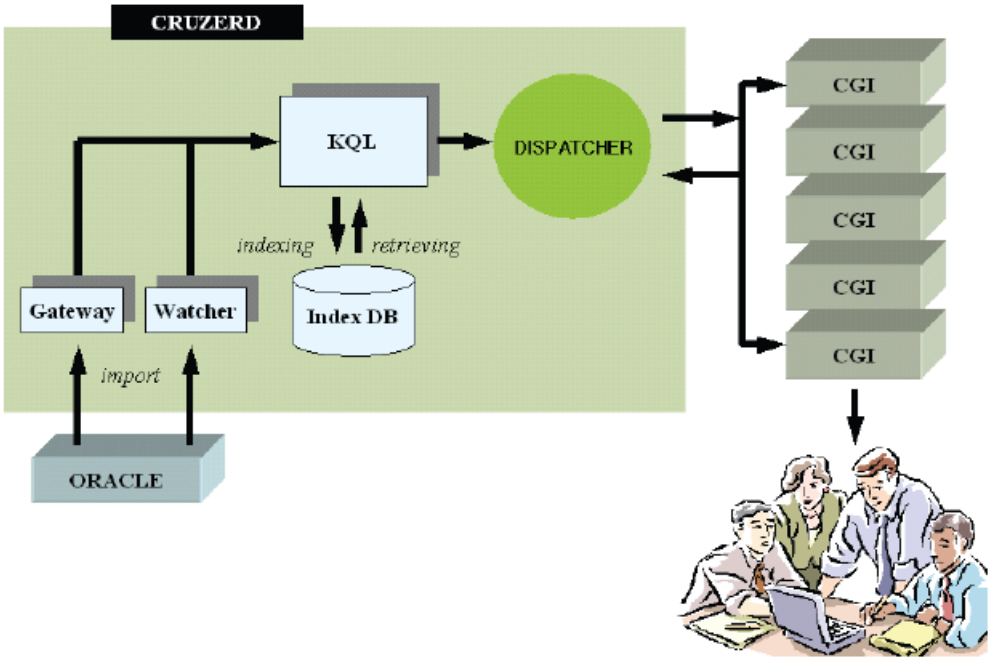
매일 오전 6 시에 색인되도록 설정한 경우이며 오전 6 시가 지난 후 kql.log를 확인하여 색인이 실제로 진행되었는지 확인한다.

A.2.2. 기사 검색 시스템 구현 예

다음 사례는 독크루저를 이용하여 간단하게 구축한 뉴스 검색 시스템 NEWSBOY이다. NEWSBOY는 다음과 같은 특성을 가지고 있다.

- 여러 정보원으로부터 자료 취합
- 정보원으로부터 뉴스가 오는 즉시 검색이 가능

- 시간별, 유사도별 검색 가능
- 뉴스 분야별 검색
- 데이터베이스 시스템과 연동
- 참조 사이트
  - http://www.inews24.com
  - http://news.empas.com



[그림 A.1] 뉴스 검색 시스템

뉴스 검색에서 사용할 테이블을 정의한다.

다음은 news_ddl.kql이다.

```
CREATE TABLE GISA
(
         rowid string,
```

```
        TT text,
        BD text,
        DT string null,
        SO int32 null,
        AU string list null,
        SC int32 null
);
CREATE TABLE SOURCE
(
        rowid string,
        ID int32 primary,
        NM string
);
CREATE TABLE SECTION
(
        rowid string,
        ID int32 primary,
        NM string
);
```

검색할 필드에 대해 인덱스를 정의한다.

```
CREATE INDEX ROWID_INDEX ON GISA(rowid);
CREATE INDEX TTBD_INDEX ON GISA(TT, BD);
CREATE INDEX DT_INDEX ON GISA(DT);
CREATE INDEX SO_INDEX ON GISA(SO);
CREATE INDEX AU_INDEX ON GISA(AU);
CREATE INDEX SC_INDEX ON GISA(SC);

CREATE INDEX ROWID_INDEX ON SOURCE(rowid);
CREATE INDEX ID_INDEX ON SOURCE(ID);
CREATE INDEX NM_INDEX ON SOURCE(NM);

CREATE INDEX ROWID_INDEX ON SECTION(rowid);
CREATE INDEX ID_INDEX ON SECTION(ID);
CREATE INDEX NM_INDEX ON SECTION(NM);
```

Oracle 연동을 위한 게이트웨이를 정의한다.

```
CREATE GATEWAY GISA_GATEWAY ON GISA
TO ORACLE GISA
```

```
FORMAT
        rowid = rowid,
        TT = TT,
        BD = BD,
        DT = DT,
        SO = SO,
        AU = AU,
        SC = SC;
CREATE GATEWAY SOURCE_GATEWAY ON SOURCE
TO ORACLE SOURCE
FORMAT
        rowid = rowid,
        ID = ID,
        NM = NM;
CREATE GATEWAY SECTION_GATEWAY ON SECTION
TO ORACLE SECTION
FORMAT
       rowid = rowid,
       ID = ID,
       NM = NM;
```

IMPORT 명령으로 Oracle 로부터 독크루저로 레코드를 가져올 수 있다.

다음은 news_dml.kql이다.

```
IMPORT RECORDS FROM ORACLE("news", "news", "")
TO GISA THROUGH GISA_GATEWAY;
IMPORT RECORDS FROM ORACLE("news", "news", "")
TO SOURCE THROUGH SOURCE_GATEWAY;
IMPORT RECORDS FROM ORACLE("news", "news", "")
TO SECTION THROUGH SECTION_GATEWAY;
```

실시간 기사 업데이트를 위해서 와처를 구동시킨다.

```
DROP WATCHER AT ORACLE("system", "manager", "");
CREATE WATCHER ON "news" AT ORACLE("system", "manager", "");

NOTIFY WATCHER AT ORACLE("news", "news", "")
 GET KEY FIELD rowid FROM GISA.rowid
 GET VIEW FIELD * FROM GISA
 PUT TO GISA_GATEWAY OF GISA;
```

```
NOTIFY WATCHER AT ORACLE("news", "news", "")
 GET KEY FIELD rowid FROM SOURCE.rowid
 GET VIEW FIELD * FROM SOURCE
 PUT TO SOURCE_GATEWAY OF SOURCE;

NOTIFY WATCHER AT ORACLE("news", "news", "")
 GET KEY FIELD rowid FROM SECTION.rowid
 GET VIEW FIELD * FROM SECTION
 PUT TO SECTION_GATEWAY OF SECTION;
```

실시간 업데이트는 다음과 같은 스크립트를 독크루저 스케줄러에 등록하면 된다.

```
-------------------- kql.rc --------------------
schedule = "run news_daily.kql;", every 4:00
schedule = "run news_minutely.kql;", every 10 sec
------------------------------------------------
```

다음은 news_daily.kql이다.

```
CREATE VOLUME temp_vol;
USE VOLUME temp_vol;

RESTART WATCHER AT ORACLE("system", "manager", "");
RUN news_ddl.kql;
RUN news_dml.kql;

SWAP VOLUME news_vol temp_vol;
COPY VOLUME news_vol TO temp_vol;
SWAP VOLUME news_vol temp_vol;
```

다음은 news_minutely.kql이다.

```
USE VOLUME news_vol;

UPDATE GISA WATCHING ORACLE("news", "news", "") through
GISA_GATEWAY;
UPDATE SOURCE WATCHING ORACLE("news", "news", "") through
SOURCE_GATEWAY;
UPDATE SECTION WATCHING ORACLE("news", "news", "") through
SECTION_GATEWAY;
```

다음과 같이 시스템 환경을 설정한다.

다음은 kql.rc이다.

```
max_no_of_connections = 80
max_no_of_volume_handles = 400
max_no_of_table_handles = 800
max_no_of_open_files = 50

max_buffer_cache_size = 256 MB
max_working_memory_size = 256 MB
no_of_parallel_task = 2
data_location = ../data

listener = 211.63.24.25, 6222

sys_log = kql.log

volume = news_vol, "vol.news.rc"
volume = temp_vol, "vol.tmp.rc"

schedule = "run news_daily.kql;", every 4:00
schedule = "run news_minutely.kql;", every 10 sec
```

두 개의 볼륨을 지정한다.

다음은 vol.news.rc이다.

```
volume_device   = "../vol.news"
log_device      = "../vol.news"
temp_device     = "../vol.news"
```

다음은 vol.tmp.rc이다.

```
volume_device   = "../vol.tmp"
log_device      = "../vol.tmp"
temp_device     = "../vol.tmp"
```

검색을 위한 환경을 설정한다.

```
;
; DOCRUZER SYSTEM CONFIGURATION
;
```

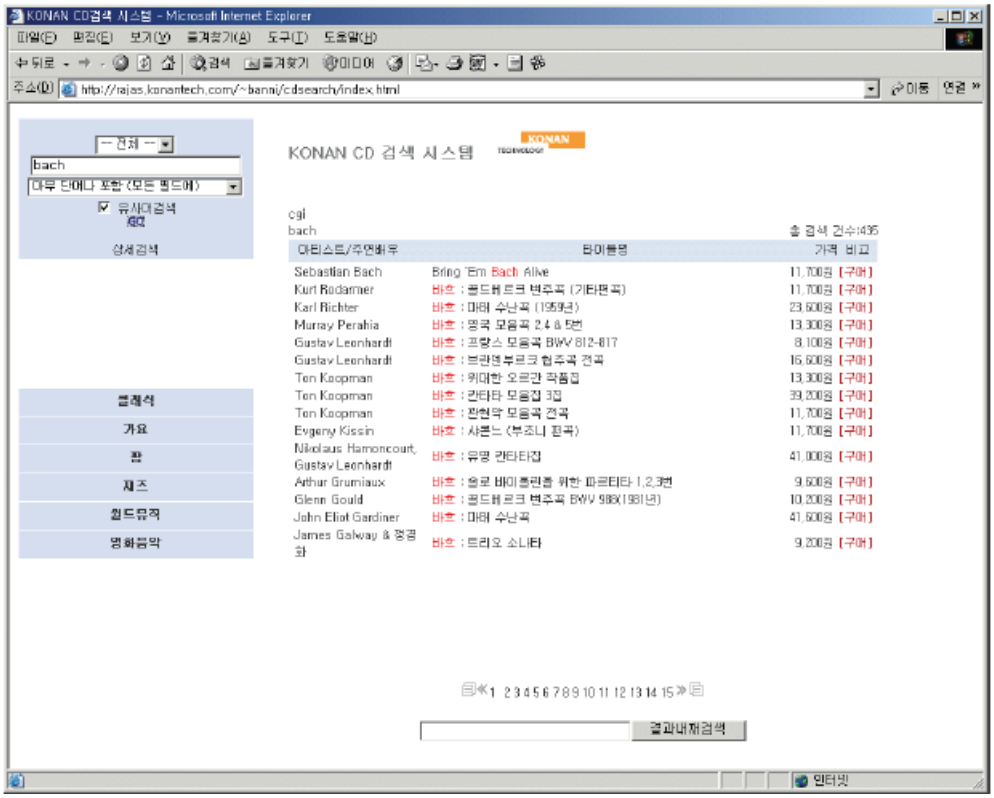
```
;;;;;;;;;;;;;;;; LOGGING ;;;;;;;;;;;;;;;;;;;;;;;
log_file_location = ../logs
sys_log_file_location = ../logs
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

no_of_parallel_task = 4
daemon_service_port = 7889

kql_link = h1, kql.rc

scenario nr1
{
         link = h1
         volume = news_vol
         table = GISA
         field = TT (100, "<font color=#FF0000>", "</font>")
         field = BD (300, "<font color=#FF0000>", "</font>")
}
```

이제 서버 쪽 설정은 끝났다. 클라이언트 쪽은 라이브러리를 사용하여 디자인에 맞게 작성하여야 한다. 다음 예는 CGI로 만들어 검색한 결과이다.



[그림 A.2] 기사 검색 시스템 구현 CGI 예

A.2.3. CD 검색 시스템 구현 예

음반 매장 같이 대규모의 CD를 관리하는 장소에서는 효율적인 CD 검색 시스템이 필요하다. 본 CD 검색 시스템은 MS-SQL에 입력되어 있는 각종 CD 정보를 색인 및 검색한다. 스키마는 'album' 테이블과 'track' 테이블로 나누어져 있으며 하나의 album에는 여러 개의 트랙이 존재한다. 검색의 단위는 album이다.

MS-SQL에 저장되어 있는 테이블은 다음과 같다.

다음은 'album' 테이블이다.

```
cdid            앨범일련번호        varchar(20)
title           음반명               text
price1          가격                 varchar(256)
price2          할인가격            varchar(256)
stock           재고량               int
label           음반사               varchar(64)
```

```
publisher          배급사                 varchar(64)
instrument         악기                    text
releasedate        출시/수입일            varchar(64)
....
catalogue          카탈로그번호           varchar(64)
media              미디어                  varchar(64)
performer          연주자/핵심아티스트    text
composer           작곡가                  text
recordingtype      녹음방식                varchar(64)
discs              디스크개수              int
bigcategory        대분류                  varchar(32)
midcategory        중분류                  varchar(32)
cover              앨범재킷사진            varchar(128)
```

다음은 track 테이블이다.

```
cdid               앨범일련번호            varchar(20)
discid             디스크일련번호          int
trackid            트랙일련번호            int
composer           작곡가                  text
performer          연주자                  text
conductor          지휘자                  text
recordingdate      녹음날짜                varchar(20)
filename           압축파일                varchar(128)
```

MS-SQL 로부터 레코드를 수입할 독크루저의 테이블을 만든다.

```
CREATE TABLE album (
         rowid             string,
         cdid              string primary,
         title             text,
         price1            string NULL,
         price2            string NULL,
         stock             int32 NULL,
         label             string para NULL,
         publisher         string para NULL,
         instrument        text,
         releasedate       string para NULL,
         catalogue         string para NULL,
         media             string para NULL,
         performer         text,
```

```
         composer         text,
         recordingtype    string para NULL,
         discs            int32 NULL,
         bigcategory      string para NULL,
         midcategory      string para NULL,
         cover            string NULL
);

CREATE TABLE track
(
        rowid             string,
        cdid              string,
        discid            int32,
        trackid           int32,
        tracktitle        text,
        composer          text,
        performer         text,
        conductor         text,
        recordingdate     string para NULL,
        filename          string NULL,
CONSTRAINT track_PK PRIMARY KEY(cdid, discid, trackid)
);

CREATE TABLE albuminfo
(
       rowid             string,
       cdid              string primary,
       info              text
);

CREATE TABLE supplyinfo
(
      rowid             string,
      cdid              string primary,
      info              text
);
```

각 테이블에서 검색할 필드들에 대해 색인을 만든다.

```
/* album 인덱스 */
CREATE INDEX album_INDEX ON album(title, performer, composer);
```

```
CREATE INDEX artist_INDEX ON album(performer, composer);
CREATE INDEX cdid_INDEX ON album(cdid);
CREATE INDEX title_INDEX ON album(title);
CREATE INDEX price1_INDEX ON album(price1);
CREATE INDEX price2_INDEX ON album(price2);
CREATE INDEX stock_INDEX ON album(stock);
CREATE INDEX label_INDEX ON album(label);
CREATE INDEX publisher_INDEX ON album(publisher);
CREATE INDEX instrument_INDEX ON album(instrument);
CREATE INDEX releasedate_INDEX ON album(releasedate);
CREATE INDEX catalogue_INDEX ON album(catalogue);
CREATE INDEX media_INDEX ON album(media);
CREATE INDEX performer_INDEX ON album(performer);
CREATE INDEX composer_INDEX ON album(composer);
CREATE INDEX recordingtype_INDEX ON album(recordingtype);
CREATE INDEX discs_INDEX ON album(discs);
CREATE INDEX bigcategory_INDEX ON album(bigcategory);
CREATE INDEX midcategory_INDEX ON album(midcategory);
CREATE INDEX cover_INDEX ON album(cover);

/* track 인덱스 */
CREATE INDEX track_INDEX ON track(composer, performer, conductor);
CREATE INDEX cdid_INDEX ON track(cdid);
CREATE INDEX discid_INDEX ON track(discid);
CREATE INDEX trackid_INDEX ON track(trackid);
CREATE INDEX tracktitle_INDEX ON track(tracktitle);
CREATE INDEX composer_INDEX ON track(composer);
CREATE INDEX performer_INDEX ON track(performer);
CREATE INDEX conductor_INDEX ON track(conductor);
CREATE INDEX filename_INDEX ON track(filename);

/* supplyinfo */
CREATE INDEX cdid_index ON supplyinfo(cdid);
CREATE INDEX info_index ON supplyinfo(info);

/* albuminfo */
CREATE INDEX cdid_index ON albuminfo(cdid);
CREATE INDEX info_index ON albuminfo(info);
```

MS-SQL 로부터 데이터를 가져오기 위한 게이트웨이를 정의한다.

```
/* CREATE GATEWAY */
CREATE GATEWAY gateway1 ON album
TO sqlserver album_DB.dbo.album
FORMAT
                    cdid = cdid,
                    title = title,
                    price1 = price1,
                    price2 = price2,
                    stock = stock,
                    label = label,
                    publisher = publisher,
                    instrument = instrument,
                    releasedate = releasedate,
                    catalogue = catalogue,
                    media = media,
                    performer = performer,
                    composer = composer,
                    recordingtype = recordingtype,
                    discs = discs,
                    bigcategory = bigcategory,
                    midcategory = midcategory,
                    cover = cover;

CREATE GATEWAY gateway2 ON track
TO sqlserver album_DB.dbo.track
FORMAT
                    cdid = cdid,
                    discid = discid,
                    trackid = trackid,
                    tracktitle = tracktitle,
                    composer = composer,
                    performer = performer,
                    conductor = conductor,
                    recordingdate = recordingdate,
                    filename = filename;

CREATE GATEWAY gateway3 ON albuminfo
TO sqlserver album_DB.dbo.albuminfo
FORMAT
                    cdid = cdid,
```

```
                    info = info;

CREATE GATEWAY gateway4 ON supplyinfo
TO sqlserver album_DB.dbo.supplyinfo
FORMAT
                    cdid = cdid,
                    info = info;
```

IMPORT 명령으로 레코드를 가져온다.

```
/* IMPORT */
IMPORT RECORDS FROM sqlserver("192.168.1.79", "album", "album")
TO album THROUGH gateway1;

IMPORT RECORDS FROM sqlserver("192.168.1.79", "album", "album")
TO track THROUGH gateway2;

IMPORT RECORDS FROM sqlserver("192.168.1.79", "album", "album")
TO albuminfo THROUGH gateway3;

IMPORT RECORDS FROM sqlserver("192.168.1.79", "album", "album")
TO supplyinfo THROUGH gateway4;
```

독크루저의 KQL 설정 파일을 작성한다.

다음은 kql.rc이다.

```
max_no_of_connections           = 80
max_no_of_volume_handles        = 400
max_no_of_table_handles         = 800
max_no_of_open_files            = 50
max_buffer_cache_size           = 256 MB
max_working_memory_size         = 256 MB
no_of_parallel_task             = 2
data_location                   = ../data
listener                        = 192.168.1.233, 6333
volume                          = album_vol, "vol.rc"
sys_log                         = console
```

독크루저의 볼륨 설정 파일을 작성한다.

다음은 vol.rc이다.

```
volume_device                    = "../volume/album"
log_device                       = "../volume/album"
temp_device                      = "../volume/album"
```

독크루저 검색 서버인 docruzerd를 위한 환경설정 파일을 작성한다.

다음은 docruzerd.rc이다.

```
daemon_service_port            = 7577
no_of_parallel_task            = 4
log_file_location              = ../logs
link                           = h1, kql.rc
include                        = scn.rc
```

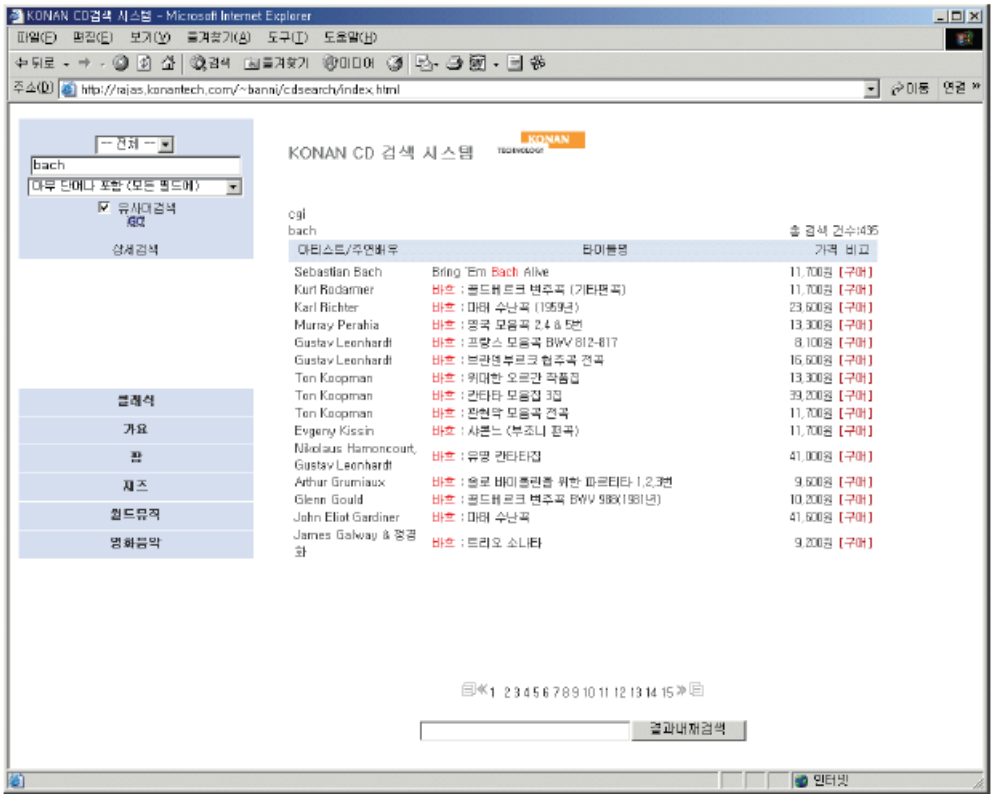
검색 시나리오 파일을 작성한다.

다음은 scn.rc이다.

```
scenario m1
{
         link = h1
         volume = album_vol
         table = album
         field = cdid
         field = title
         field = price1
         field = price2
         field = stock
         field = label
         field = publisher
         field = instrument
         field = releasedate
         field = catalogue
         field = media
         field = performer
         field = recordingtype
         field = bigcategory
         field = midcategory
         field = cover
         field = supplyinfo.info(cdid)
         field = albuminfo.info(cdid)
}
```

```
scenario m2
{
        link = h1
        volume = album_vol
        table = track
        field = cdid
        field = album.title (cdid)
        field = album.price1 (cdid)
        field = album.price2(cdid)
        field = album.stock(cdid)
        field = album.label(cdid)
        field = album.publisher(cdid)
        field = album.instrument(cdid)
        field = album.releasedate(cdid)
        field = album.catalogue(cdid)
        field = album.media(cdid)
        field = album.performer(cdid)
        field = album.recordingtype(cdid)
        field = album.discs(cdid)
        field = album.bigcategory(cdid)
        field = album.midcategory(cdid)
        field = album.cover(cdid)
        field = supplyinfo.info(cdid)
        field = discid
        field = trackid
        field = tracktitle (512, "<font color=red>", "</font>")
        field = composer (256, "<font color=red>", "</font>")
        field = conductor (256, "<font color=red>", "</font>")
        field = performer
        field = recordingdate
        field = filename
}
```

CD 검색 시스템의 사용자 인터페이스는 다음과 같다.



[그림 A.3] CD 검색 시스템 사용자 인터페이스

(주)코난테크놀로지