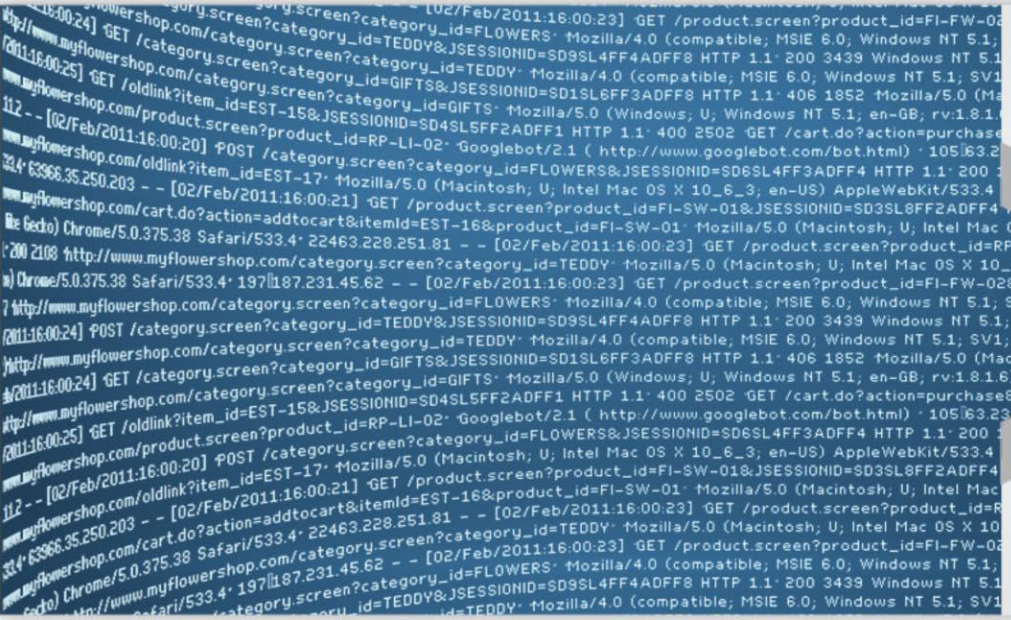
splunk> MOS



데이터 처리 구조

splunk>

Search : 구조

- Splunk 검색 언어 구현의 기초 형식

SEARCH | DATA PROCESS | ANALYSIS

검색 요건 | 데이터 처리 | 분석

100 이상의 명령어

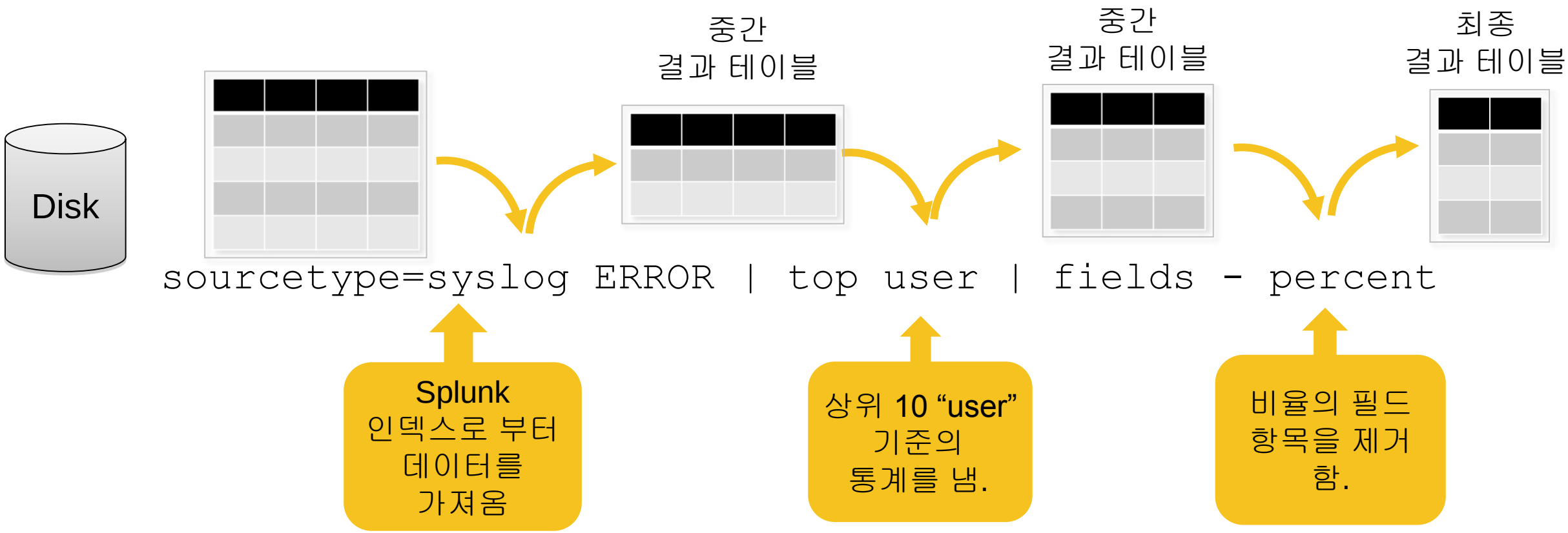
Command	Description
transaction	
eval	
fields	
stats	
head	
sort	
rangemap	
dedup	
multikv	split multi-lined tabular events into single-lined events

splunk.com > Documentation > Search Reference

abstract accum addcoltotals addinfo addtotals af analyzefields anomalies anomalousvalue append appendcols ar associate audit autoregress bin bucket chart cluster collect common contingency convert correlate counttable crawl ctable dbinspect dedup delete delta diff discretize erex eval eventcount eventstats excerpt extract file fillnull folderize format gentimes head highlight iconify input inputcsv inputlookup iplocation join kmeans kv kvform loadjob localize localop lookup macro makecontinuous makemv maketable map metadata multikv mvcombine mvexpand nomv outlier outlierfilter outputcsv outputlookup outputtext overlap rangemap rare regex relevancy rename replace reverse run savedsearch savedsplunk script scrub selfjoin sendemail set sichart sirare sistats sitimechart sitop slc stash strcat streamstats sumindex summaryindex tail test timechart top transaction transam trendline typeahead typelearner typer uniq untable xmlkv xmlunescape xpath xyseries

검색 파이프 라인

- ✓ 검색은 데이터를 불러 오는 명령어 입니다.
- ✓ 불러온 데이터를 “pipeline” 을 통하여 다음 커맨드로 전달 합니다.



Data의 처리 및 정리

- ✓ “|” 심볼을 사용한 이후 그 결과를 다음 명령의 Input으로 주입합니다.
  (UNIX Shell 과 비슷한 구조)

```
eventtype=FW_deny tag=publicID user!=araitz | sort -user
```

- ✓ 예를 들어 위의 경우 불려진 데이터의 결과는 “user” 란 필드 기준으로 정렬을 하였습니다.

검색 : 명령어 유형

- 필드 정의 / 데이터 처리
  - multikv
  - eval
  - strcat
  - rex
- 분석
  - stats
  - chart
  - timechart
  - top
- 데이터 필터
  - dedup
  - regex
  - search
- 필드 변환
  - rename
  - replace
- 데이터 정렬
  - sort
  - head
  - tail

명령어 : 필드 처리

1. 필드 처리 주요 명령어

- **multikv** : Table 형태의 raw data 를 상위 필드명을 기준으로 Field 등록을 자동으로 함

```
sourcetype=ps | multikv | stats max(pctCPU) by COMMAND
```

- **rex** : 수집된 데이터에서 정규식으로 필드 추출. 예로 "from“ 과 "to" 란 필드를 "From: Susan To: Bob“ 이란 raw데이터에서 추출 함

```
* | rex field=_raw "From: (?<from>.*) To: (?<to>.*)"
```

- **strcat** : 신규로 복합 필드의 값이 들어간 필드 생성. user_name_add=“user_name“/“user_add".

```
* | strcat user_name “ / " user_add user_name_add
```

- **eval** : velocity 라는 신규 벨류를 velocity = distance field value / time field value 로 연산하여 생성

```
* | eval velocity=distance/time
```

명령어 : 필드 변환 / 필터

2. 필드 변환 주요 명령어

- **rename** : user_name 이라는 필드명을 "고객이름" 이라는 필드 명으로 변경 .

```
* | rename user_name as "고객 이름"
```

- **replace** : user_name 이란 필드의 밸류 값 중 "laura" 라고 있는 데이터를 "로라" 로 변경

```
* | replace laura with "로라" in user_name
```

3. 데이터 필터 주요 명령어

- **search** : product_name 에 "oil" 을 포함한 데이터와 user_add 에 "Kang" 이 있는 데이터만 추려 냄.

```
* | search product_name="*oil*" OR user_add="Kang*"
```

- **regex** : 정규식으로 필터 하여 10.xx.xx.xx.xx 대의 IP가 있는 데이터 만 추려 냄.

```
* | regex _raw=(?<!\d)10.\d{1,3}\.\d{1,3}\.\d{1,3}(?!\d
```

- **dedup** : user_name 이란 필드의 수치가 유일한 하나의 데이터인 필터 :

```
* | dedup user_name
```

명령어 : 데이터 정렬 / 결과 묶음

4. 데이터 정렬 명령어

- **sort** : 결과를 product_name 의 오름차순과 user_name 의 내림차순으로 정렬.

```
* | sort product_name , - user_name
```

- **reverse** : 결과를 순서를 뒤로 돌려 정렬.

```
* | reverse
```

- **head** : 처음 20 개의 결과를 보여주며 정렬 함.

```
* | head 20
```

5. 결과 묶음 명령어

- **transaction** : 결과가 product_name 과 user_name 이 동일하게 들어가 데이터를 묶어 해당 이벤트들의 시간적 범위가 30초 내에 있고 각각의 이벤트 사이가 5초 내에 떨어져 있는 것들을 하나의 이벤트로 묶습니다.

```
* | transaction fields="product_name, user_name" maxspan=30s maxpause=5s
```