



실무에 활용하는 Elasticsearch 검색엔진 구축

7일차 : ElasticSearch 3

Aggregation

기타 고급검색기법

오늘의 아젠다



1. Aggregation

2. 기타 고급검색 기법

- Suggesters (스펠체커)
- 연관검색어 추출 기법
- 플러그인 만들기

1. Aggregation

Aggregation의 개요

Facet

- Term Facet
- Range Facet
- Date Facet
- Statistical Facet
- geo distance Facet

Aggregations

버킷 단위의 도큐먼트 셋에 대해서 지속적인 추적과 연산 수행
-최소(min),최대(max),합(sum),평균(avg),개수(value count)

- Term aggregations
- Range aggregations
- Data aggregations

aggregation은 “facet 의 재 탄생” 입니다.

TV Display Size

- ☐ 32 Inches & Under (987)
- ☐ 33 to 43 Inches (548)
- ☐ 44 to 49 Inches (350)
- ☐ 50 to 59 Inches (559)
- ☐ 60 to 69 Inches (332)
- ☐ 70 Inches & Up (167)

Television Resolution

- ☐ 4K Ultra HD (353)
- ☐ 1080p (1,463)
- ☐ 1080i (16)
- ☐ 760p (6)
- ☐ 760i
- ☐ 720p (617)
- ☐ 720i (2)
- ☐ 480p (15)
- + See more

Television Feature

- ☐ Smart TV (983)
- ☐ 3D (317)

Sponsored ⓘ

HDTV Antenna In Double Blister Clamps and Analog TV Broadcast

by TV Antenna

\$12.99 ~~\$39.99~~ Prime

Get it by **Tuesday, Aug 2**

FREE Shipping on eligible orders

★★★★☆ 50

Aggregation의 구성요소

Bucketing

기준을 만족하는 문서 집합

조건에 해당하는 문서를 버킷이라는 저장소에 저장하는 단위

Metric

우리가 알고 싶은 정보

Bucket에 포함된 문서에 대한 통계정보 (min,max,avg, etc ..)

Matrix

여러 필드에서 작동하고 요청된 문서 필드에서 추출한 값을 기반으로

행렬 결과를 생성하는 집합

Pipeline

집계의 결과를 다른 집계에 활용한다.

레벨에 따라 부모/자식 집계로 나뉜다.

buckets_path 지정 필수

ex) metric
SELECT count(hit) FROM table bucket GROUP BY hit

Aggregation의 기본 문법

```
"aggregations" : {  
  "<aggregation_name>" : {  
    "<aggregation_type>" : {  
      <aggregation_body>  
    }  
    [, "meta" : { [<meta_data_body> ] } ]?  
    [, "aggregations" : { [<sub_aggregation>]+ } ]?  
  }  
  [, "<aggregation_name_2>" : { ... } ]*  
}
```

```
ex)  
{  
  "aggs" : {  
    "terms" : {  
      "field" : "color"  
    }  
  }  
}  
select color, count(*) as colors  
from cars  
group by color
```

Aggregation의 Scope

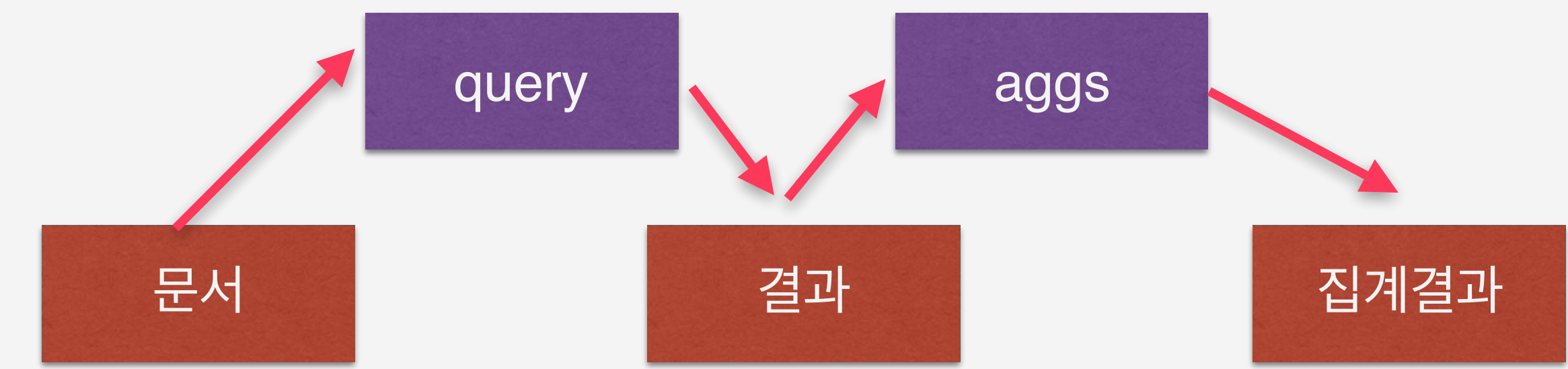
1. 어그리게이션을 검색 요청과 함께 실행하면 질의 결과내(Scope) 안에서 수행
2. 질의 가 생략된 어그리게이션은 모든 문서에 대한 질의로 간주(MatchAllQuery)
3. 글로벌 버킷을 통해 질의가 포함된 어그리게이션도 모든 문서에 대한 어그리게이션 계산 가능
예)

```
"aggs" : {  
  "avg_price" : {  
    "avg" : { "field" : "price"}  
  },  
  "all_price" : {  
    "global" : {},  
    "aggs" : {  
      "avg_price" : {  
        "avg" : { "field" : "price"}  
      }  
    }  
  }  
}
```

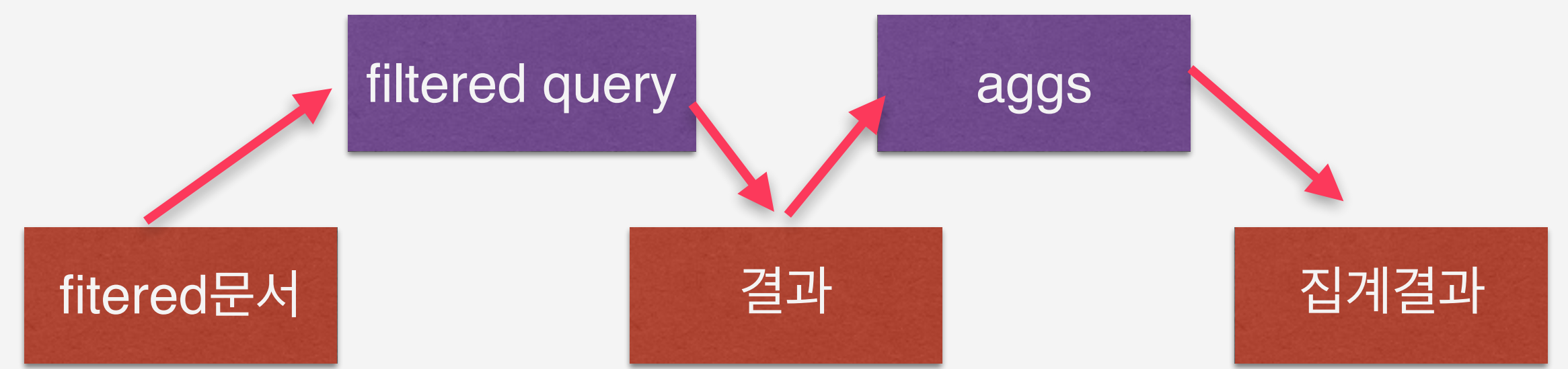

Aggregation의 필터

검색 질의가 아니라 에그리게이션 내부에서 직접 필터링 하여 적용 할수 있음
예)

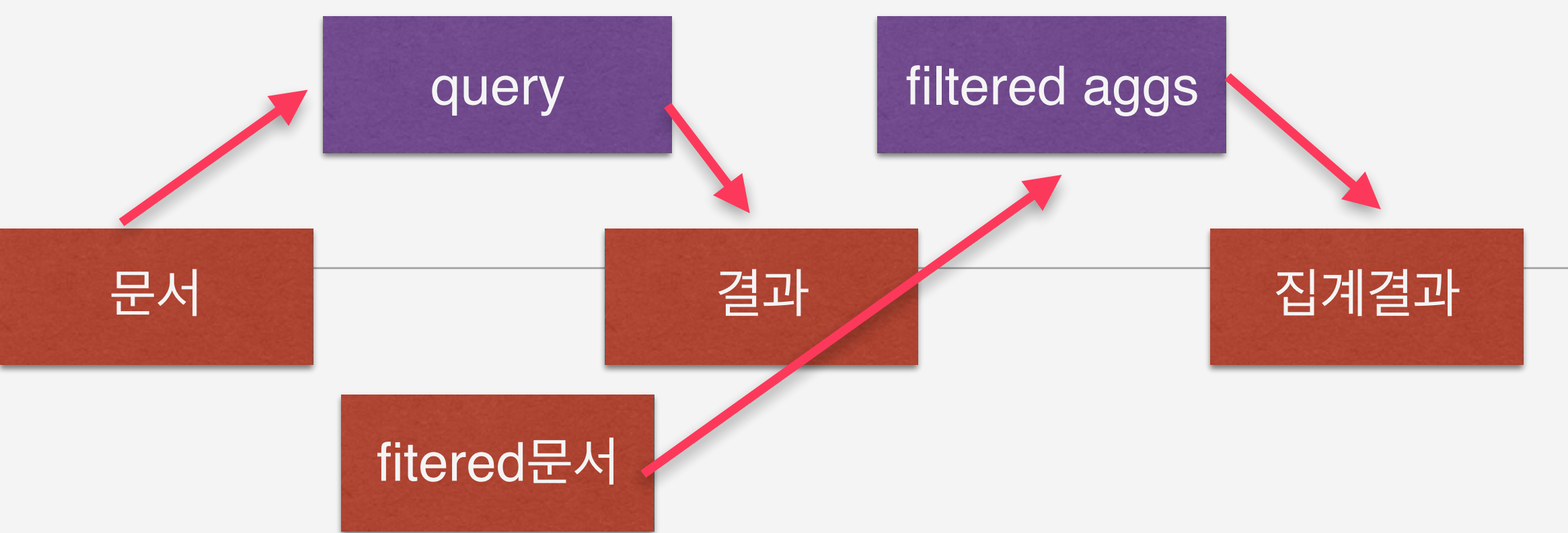
```
"aggs" : {  
  "recent_sales" : {  
    "filter" : {  
      "range" : "{  
        "sold" : {  
          "from" : "now-1M"  
        }  
      }  
    }  
  },  
  "aggs" : {  
    "avg_price" : {  
      "avg" : { "field" : "price"}  
    }  
  }  
}
```



일반 질의



필터 질의



필터 질의

Metric의 종류

Avg Aggregation

Cardinality Aggregation

Extended Stats Aggregation

Geo Bounds Aggregation

Geo Centroid Aggregation

Max Aggregation

Min Aggregation

Percentiles Aggregation

Percentile Ranks Aggregation

Scripted Metric Aggregation

Stats Aggregation

Sum Aggregation

Top hits Aggregation

Value Count Aggregation

Metric 문법 구조

```
GET /{index_name}/_search?pretty
{
  "size": 0,
  "aggs": { // 집계
    "my_aggs": { // 집계 명 (사용자 정의)
      "{metric_args_type}": { // 집계 타입
        "field": "{file_name}" // 집계 대상 필드
        "order" : { "{field_name}" : "desc" } // 정렬
      }
    }
  }
}
```

Bucket의 종류

Adjacency Matrix Aggregation

Children Aggregation

Date Histogram Aggregation

Date Range Aggregation

Diversified Sampler Aggregation

Filter Aggregation

Filters Aggregation

Geo Distance Aggregation

GeoHash grid Aggregation

Global Aggregation

Histogram Aggregation

IP Range Aggregation

Missing Aggregation

Nested Aggregation

Range Aggregation

Reverse nested Aggregation

Sampler Aggregation

Significant Terms Aggregation

Terms Aggregation

Bucket 문법 구조

GET /{index_name}/_search?pretty

```
{
  "size": 0,
  "aggs": { // 집계
    "my_aggs": { // 집계 명 (사용자 정의)
      "{bucket_args_type}": { // 집계 타입
        "field": "{file_name}" // 집계 대상 필드
        "order" : { "{field_name}" : "desc" } // 정렬
      }
    }
  }
}
```

Significant Terms Aggregation

통계적으로 의미 있는 데이터를 찾아냄

데이터 집합에서 Uncommonly common 텀즈를 찾는것이 목적

Uncommonly common 데이터

- 비정상적으로 자주 발생하는 데이터
- 일반적인 통계적 배경 오류보다 더 빈번한 데이터
- 통계적으로 예외적인 데이터
- 관측자가 주목해야 하는 관심데이터

주) 내부적 통계치 계산을 수반하기 때문에 큰 색인에서는 매우 무거울수 있음

Significant Terms Aggregation (2)

결과 예)

```
"aggregations" : {  
  "significantCrimeTypes" : {  
    "doc_count": 47347,  
    "buckets" : [  
      {  
        "key": "Bicycle theft",  
        "doc_count": 3640,  
        "score": 0.371235374214817,  
        "bg_count": 66799  
      },  
      ...  
    ]  
  }  
}
```

background rate = $66799 \text{ (bg_count)} / 100000 \text{ (total)}$

foreground rate = $3640 \text{ (doc_count)} / 47347 \text{ (그룹의 갯수)}$

score 계산

$= (\text{absolute change}) * (\text{relative change})$

$= |\text{foreground rate} - \text{background rate}| * (\text{foreground rate} / \text{background rate})$

foreground 그룹 : 특정한 조건에 대한 사용자 그룹에 대해서 높은 빈도 분석

background 그룹 : 모든 그룹에 대해서 높은 빈도 분석

Aggregation의 정렬

1) 버킷의 경우

_count : 결과 문서의 갯수 기반 정렬 (기본값)

terms.histogram, data_histogram 버킷 적용

_term : 텀의 알파벳 값으로 정렬

_key : 버킷의 키값으로 정렬

2) 메트릭의 경우

계산된 계산값에 의해 정렬

예)

```
"aggs" : {  
  "color" : {  
    "terms" : {  
      "field" : "color",  
      "order" : "{  
        "avg_price" : "asc"  
      }  
    }  
  },  
  "aggs" : {  
    "avg_price" : {  
      "avg" : { "field" : "price"}  
    }  
  }  
}
```

Pipeline의 종류

Avg Bucket Aggregation

Derivative Aggregation

Max Bucket Aggregation

Min Bucket Aggregation

Sum Bucket Aggregation

Stats Bucket Aggregation

Extended Stats Bucket Aggregation

Percentiles Bucket Aggregation

Moving Average Aggregation

Cumulative Sum Aggregation

Bucket Script Aggregation

Bucket Selector Aggregation

Serial Differencing Aggregation

파이프 라인 집계 구조

```
{
  "size": 0,
  "aggs": {
    "{parent_aggs_name}": { // 부모 집계 이름 (사용자 정의)
      "{bucket_type_name}": { // 버킷 집계 타입 이름
        "field": "{field_name}" // 버킷 집계 대상 필드
      },
      "aggs": {
        "{aggs_name}": { // 집계 이름 (사용자 정의)
          "{metric_aggs_type}": { // 매트릭 집계 타입
            "field": "{field_name}" // 매트릭 집계 대상 필드
          }
        }
      },
    },
    "{pipe_arrgs_name}": { // 파이프라인 집계 이름(사용자정의)
      "{pipe_arrgs_type}": { // 파이프라인 집계 타입
        "buckets_path": "{parent_aggs_name} > {aggs_name}" // 경로
      }
    }
  }
}
```


Aggregation의 근사화

성능을 위해 정확도를 100% 보장하지 않음

1) Cardinality Aggregation

유일하고 고유한(unique, distinct) 카운트를 제공함

SELECT COUNT(DISTINCT color) FROM cars

precision_threshold 옵션으로 정확도를 조절

2) Percentiles(백분위) Aggregation

아웃라이어(outlier, 이상치) 를 찾을 때 사용

전체 데이터에 대해 평균(mean) 또는 중앙값(median)만을 보는 경우

```
{
  "aggs" : {
    "load_time_outlier" : {
      "percentiles" : {
        "field" : "load_time"
      }
    }
  }
}
```



```
"aggregations": {
  "load_time_outlier": {
    "values" : {
      "1.0": 15,
      "5.0": 20,
      "25.0": 23,
      "50.0": 25,
      "75.0": 29,
      "95.0": 60,
      "99.0": 150
    }
  }
}
```


Aggregation 사용시 주의점

1. aggregation은 정확한 값을 보장하지 않는다. (size, shard_size 조절)

Calculating Document Count Error 값을 살펴보자

```
{
  ...
  "aggregations" : {
    "products" : {
      "doc_count_error_upper_bound" : 46,
      "buckets" : [
        {
          "key" : "Product A",
          "doc_count" : 100
        },
        {
          "key" : "Product Z",
          "doc_count" : 52
        },
        ...
      ]
    }
  }
}
```

sum_other_doc_count : 상위 갯수에 포함되지 않은 전체 텀

2. text 필드의 fielddata 는 false이다. (성능 고려사항)

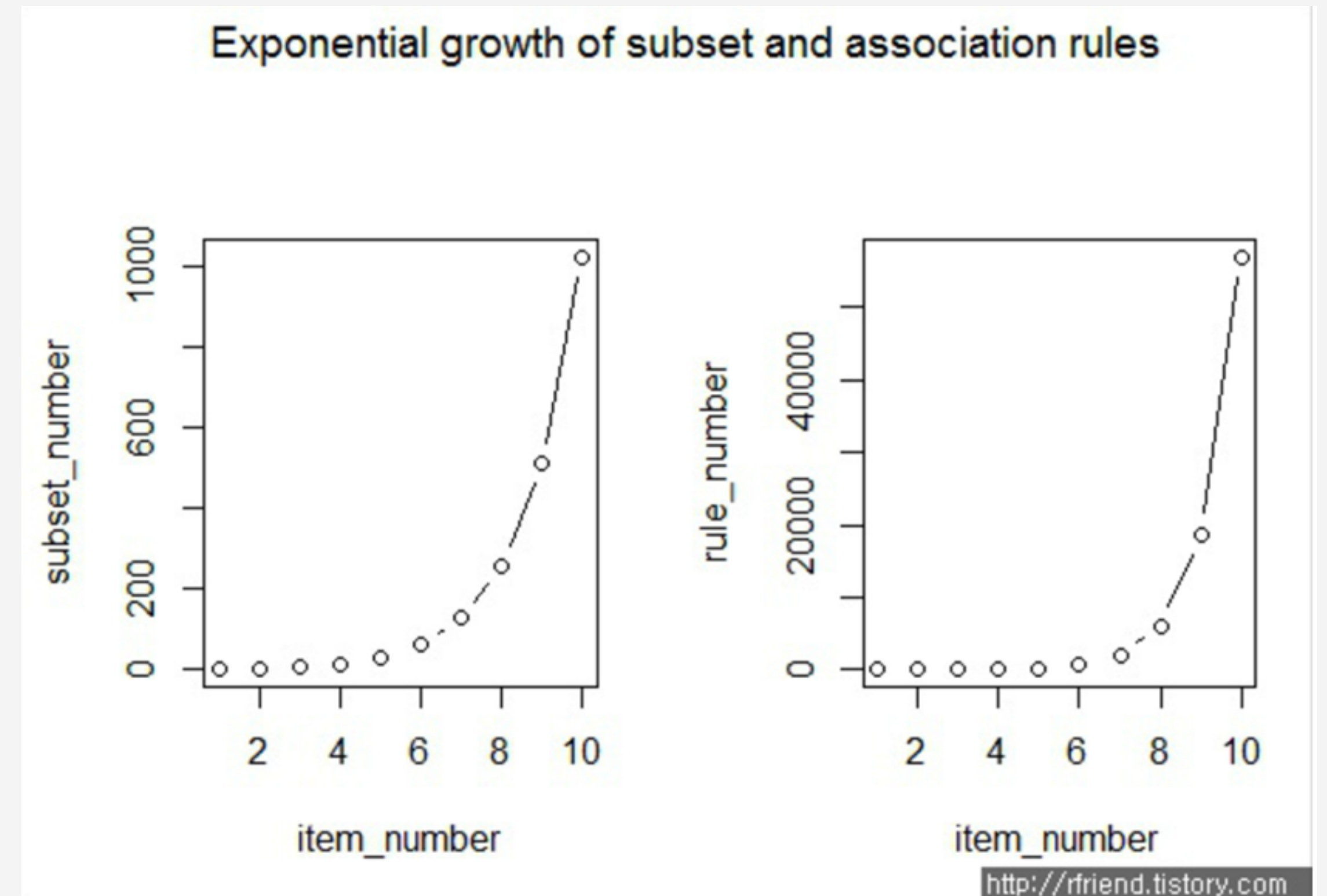
2. 기타 고급 검색기법

연관규칙 탐색

키워드 혹은 거래(Transaction)에서 나타나는 데이터에서 빈발 집합 셋을 찾아 내는 것

모든 항목의 집합은 $number\ of\ rules = 3^k - 2^{k+1} + 1$

효율적인 연관규칙 탐색 알고리즘 필요



연관규칙 탐색

5개의 원소 항목 집합 $I = \{A, B, C, D, E\}$ 일 때,

모든 가능한 항목집합 = $2^k - 1 = 2^5 - 1 = 32 - 1 = 31$

1-항목집합: 5개

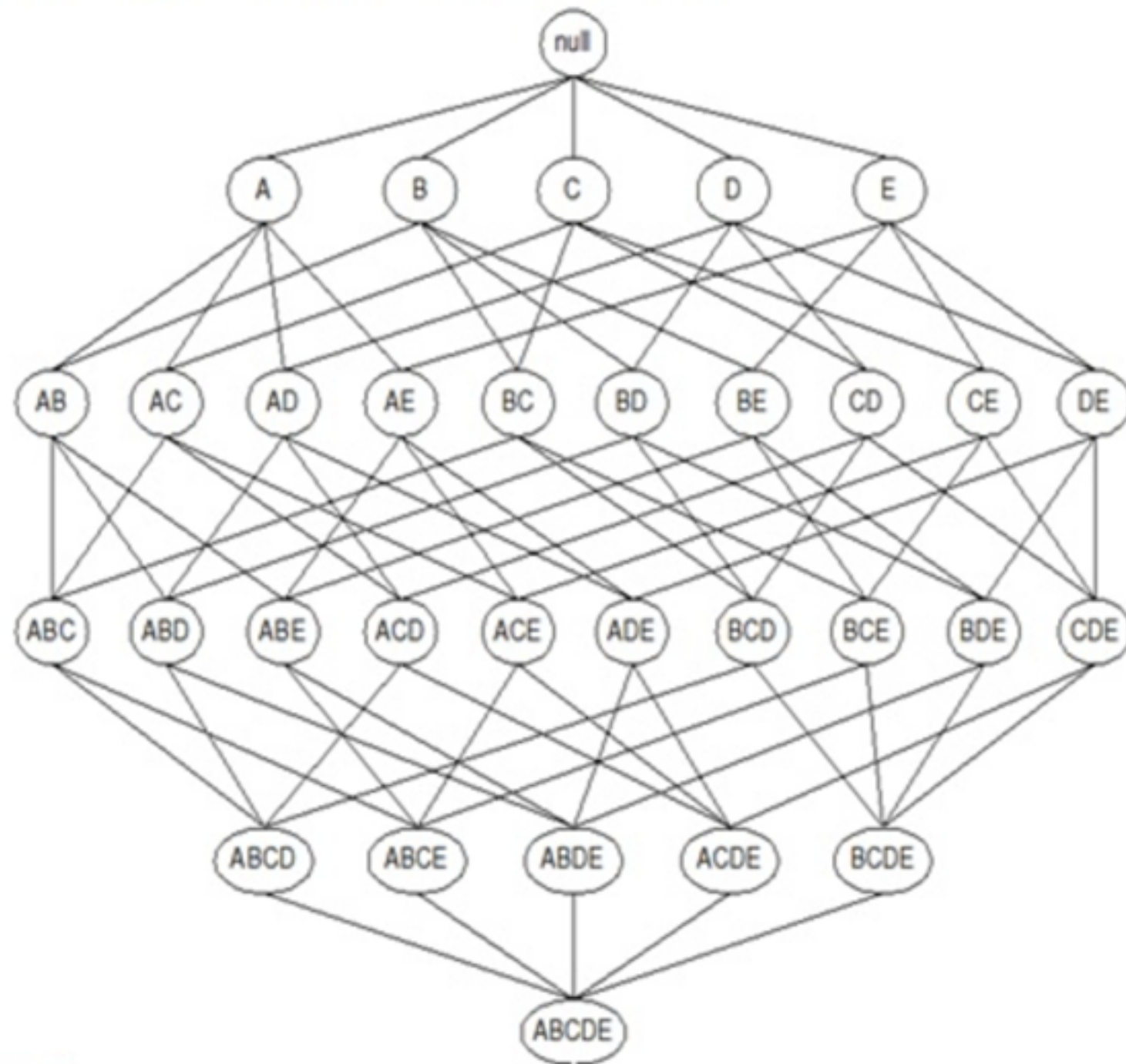
2-항목집합: 10개

3-항목집합: 10개

4-항목집합: 5개

5-항목집합: 1개

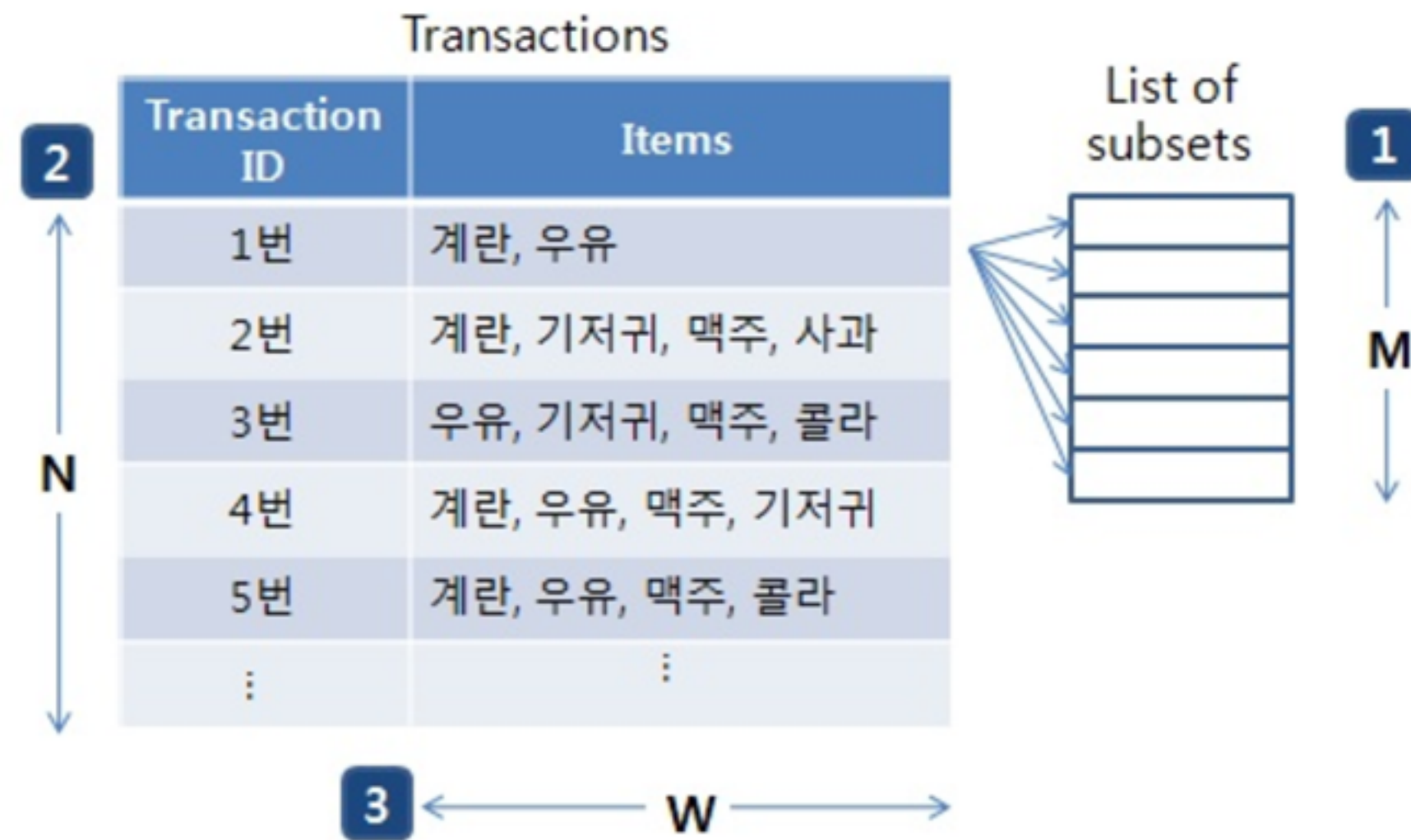
총 계 31개



[R 분석과 프로그래밍] <http://rfriend.tistory.com>

연관규칙 탐색 알고리즘의 종류

Association Rule : strategy and algorithm



- 1 모든 가능한 항목집합 개수(M)를 줄이는 방식 ➡ Apriori algorithm
- 2 Transaction 개수(N)를 줄이는 방식 ➡ DHP Algorithm
- 3 비교하는 수(W)를 줄이는 방식 ➡ FP-growth Algorithm

[R 분석과 프로그래밍] <http://rfriend.tistory.com>

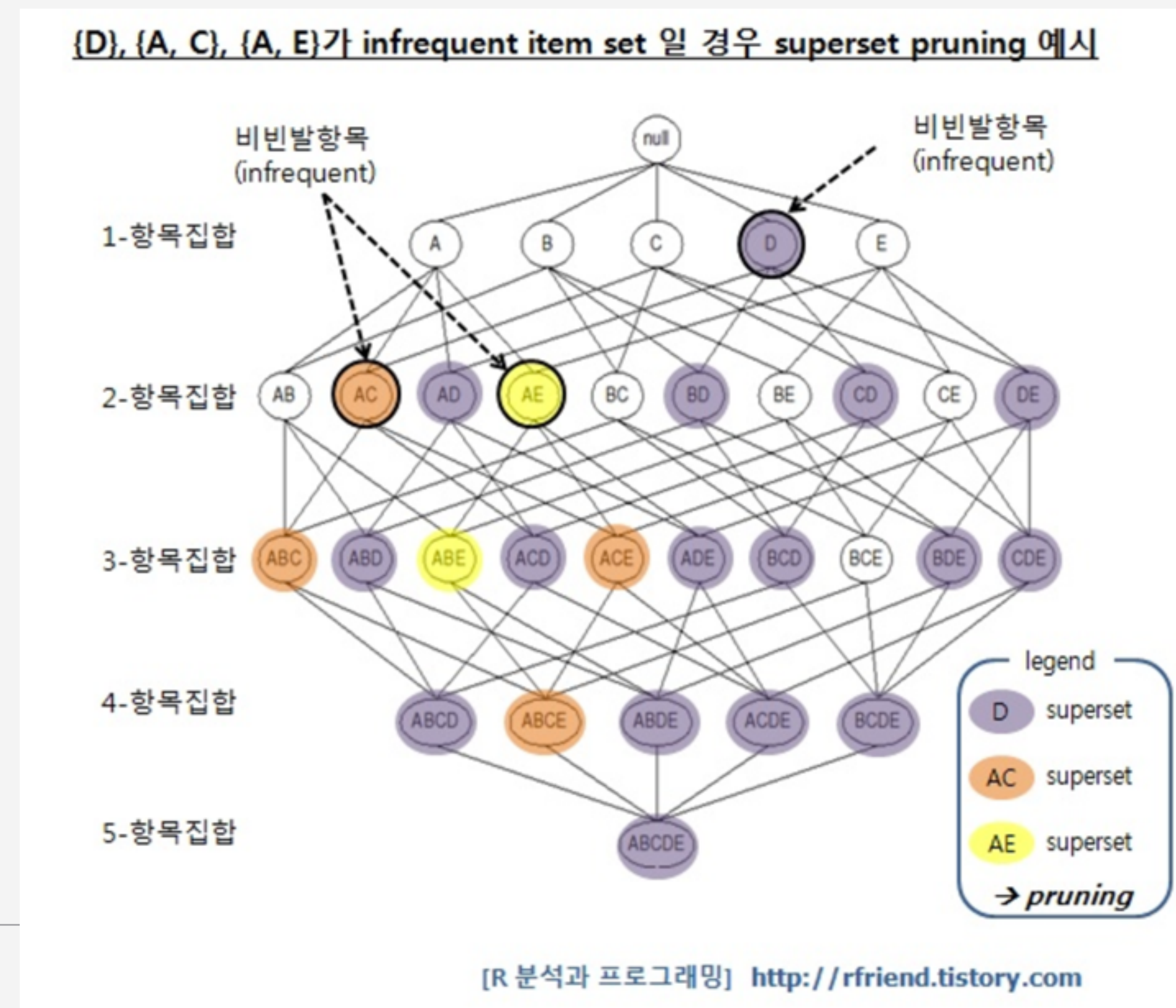
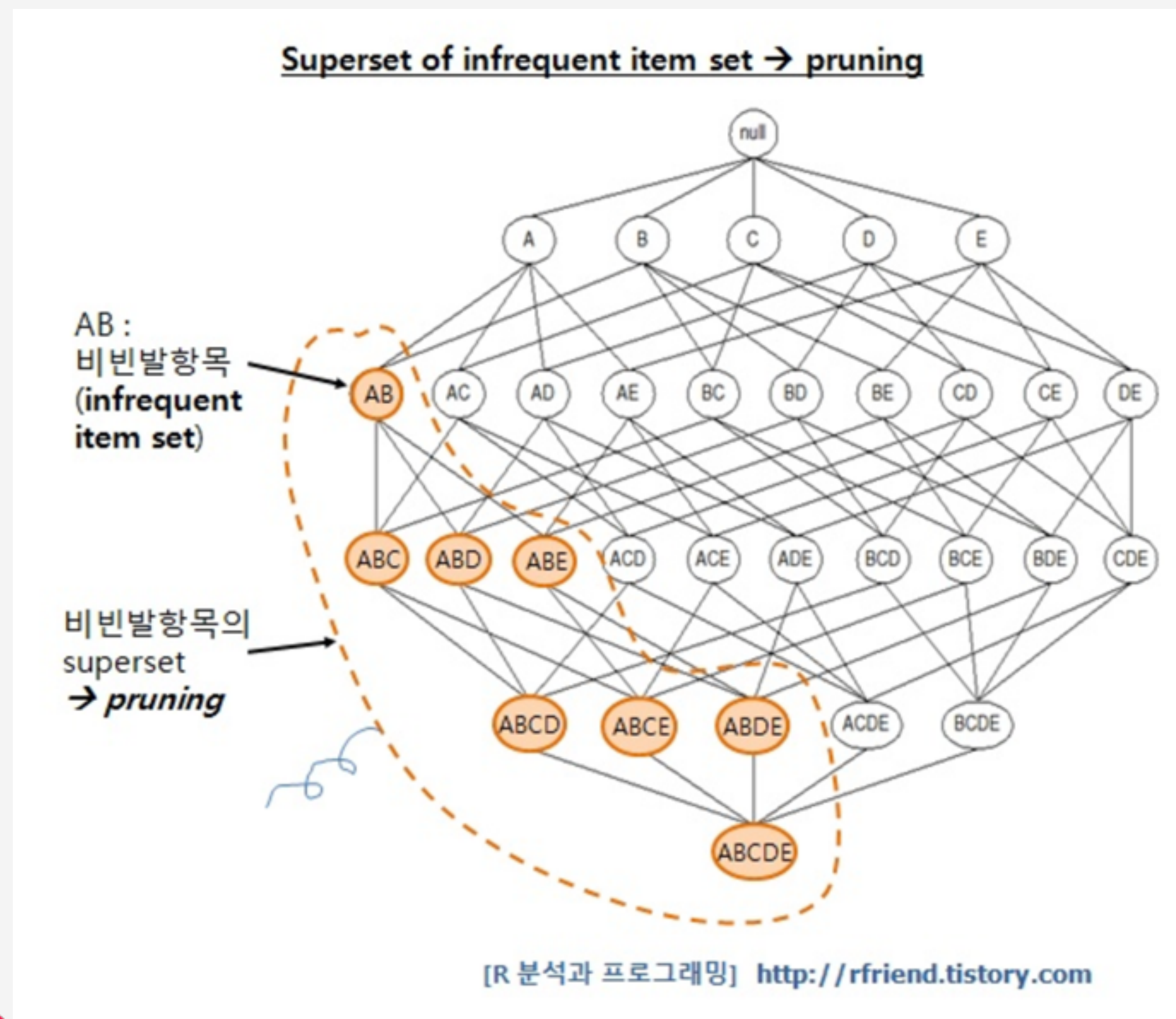
FP-Growth

https://www.youtube.com/watch?v=vPcJEFFWN_k

Apriori 알고리즘

1) 한 항목집합이 빈발(frequent)하다면 이 항목집합의 모든 부분집합은 역시 빈발항목집합이다.
(frequent item sets -> next step)

2) 한 항목집합이 비빈발(infrequent)하다면 이 항목집합을 포함하는 모든 집합은 비빈발항목집합이다. (superset -> pruning)



검색엔진을 이용한 손쉬운 계산

필요 데이터

: 하나의 검색 세션을 구분 가능한 검색 쿼리 로그

하나의 검색 세션은 아이디 혹은 세션아이디와 시간을 기준으로 정의

세션	키워드
1	삼성
1	애플
1	엘지
2	삼성
2	애플
2	구글
3	김연아
3	피겨
3	동계올림픽
4	김태희

삼성과 연관있는 빈발 집합셋을 찾고자 한다면?

- 1. 삼성을 검색하여 나온 세션을 찾는다.
- 2. 세션값을 기준으로 검색하여 모든 키워드 셋을 찾는다.
- 3. Facet혹은 Aggregation을 이용하여 가장 많이 언급된 빈발셋을 순서대로 찾는다.
- 4. 최소 지지도를 설정하여 최소 지지도 미만의 데이터는 버린다.

검색엔진을 이용한 손쉬운 계산

Solr Example (Self JOIN 연산 기능 이용)

`http://{검색엔진호스트}:{검색엔진포트}/solr/querylog/select?q={!join+from=signature%20to=signature}keyword_s:"삼성"&facet=true&facet.field=keyword_s&facet.mincount=1&facet.count=5&rows=0`

ElasticSearch 의 경우

term aggregation 이후의 session값으로 재 aggregation 수행

엘라스틱서치에서 연관검색어 추출 기법

연관규칙추출실습.txt

Ingest Node 사용법

엘라스틱 서치 5.0에 새로 추가된 노드
기본 값은 node.ingest : true

기본 사용 방법

```
{  
  "description" : "describe pipeline",  
  "processors" : [  
    {  
      "set" : {  
        "field": "foo",  
        "value": "bar"  
      }  
    }  
  ]  
}
```

ingest node 예제.txt

기타 사용 가능한 프로세스들

Append Processor
Convert Processor
Date Processor
Date Index Name Processor
Fail Processor
Foreach Processor
Grok Processor
Gsub Processor
Join Processor
JSON Processor
KV Processor
Lowercase Processor
Remove Processor
Rename Processor
Script Processor
Set Processor
Split Processor
Sort Processor
Trim Processor
Uppercase Processor
Dot Expander Processor
URL Decode Processor

Term Suggesters (스펠체커)

기본 문법

```
{
  "query" : {
    "match": {
      "message": "tring out Elasticsearch"
    }
  },
  "suggest" : {
    "my-suggestion" : {
      "text" : "trying out Elasticsearch",
      "term" : {
        "field" : "message"
      }
    }
  }
}
```

주요 옵션

text : 스펠체킹을 하기 위한 텍스트

field : 스펠체킹을 하기 위한 필드

analyzer : 스펠체킹 검색을 질의하기 위한 분석기 (기본값은 매핑값)

size : 출력되는 후보 갯수

sort : 정렬 기준 (기본 score, frequency)

string_distance : 문자열 비교를 위한 알고리즘 (damerau_levenshtein 기본)

“스펠체커만들기에제.txt” 실습

Term Suggesters (스펠체커)

한글 처리를 위한 색인 방법

1. 색인시 icu_normalizer 를 NFC로 Decompose 한다.

(노드에 icu 플러그인 설치 #sudo node1/bin/elasticsearch-plugin install analysis-icu)

2. 실제 서비스 사용시 Denormalize 한다.

Normalizer.normalize({결과스트링}, Normalizer.Form.NFC)

Completion Suggester (자동완성)

영문 위주의 자동완성 모듈이므로
커스텀 플러그인을 만들어 사용

실습!!

ElasticSearch 플러그인 만들기

모든 플러그인은 `org.elasticsearch.plugins.Plugin`를 상속

색인 분석이 필요한 경우 `org.elasticsearch.plugins.AnalysisPlugin` 인터페이스를 구현

REST API 가 필요한 경우 `org.elasticsearch.plugins.ActionPlugin` 인터페이스를 구현

`AnalysisPlugin` 의 경우 `AnalysisPlugin::getTokenFilters` 오버라이드

`ActionPlugin` 의 경우 `ActionPlugin::getRestHandlers` 오버라이드

https://github.com/blaze94/fastcampus_plugin-example.git 실습

감사합니다.

다음 주제 : 8일차 Elasticsearch Chapter 4
ELK 로 구축하는 데이터 시각화 및 분석
기타 클러스터 관리 기법