[BAT516] Database Management

# 7강
# Database application

UNIST 융합경영대학원 2023학년도 2학기

UNIST 융합경영대학원
이규민(Gyumin Lee)
glee.optimizt@gmail.com

ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

# 목차

# 1. 빅데이터

## 개요

### ❖ 데이터 과학(Data science)

- 필요성
  - 머신러닝, 인공지능 등 데이터 분석 기술의 발전에 따라, 데이터는 산업 발전의 핵심 동력이 됨
  - 전세계 정보량 증가에 따라 방대한 규모와 다양한 형태의 데이터가 산재
    - 전통적인 방식으로 수집 및 저장, 활용하는 데 한계가 있음
  - 데이터 활용 영역의 다양화
    - 기본적인 검색이나 분류뿐 아니라 데이터에 내재된 패턴을 발견함으로써 숨겨진 가치를 이끌어내고, 미래를 예측하는데 사용됨
- 정의
  - 데이터를 수집하고, 적절한 방법으로 분석하여 데이터가 내재한 정보를 파악하고, 이를 바탕으로 새로운 지식을 발견하여 특정한 문제를 해결하는 모든 과정의 활동

# 1. 빅데이터

## 개요

### ❖ 데이터 과학(Data science)

- 주요 활용 사례
  - 비즈니스 의사결정 지원
    - 시장 동향, 소비자 행동, 경쟁사 분석 등 데이터 분석을 통해 기업의 비즈니스 전략 수립을 위한 의사결정을 지원함
  - 소셜 미디어 분석
    - 데이터를 활용하여 사용자 행동, 트렌드 등을 분석하여 맞춤형 광고, 추천, 네트워크 분석 등을 수행함
  - 헬스케어
    - 환자 및 질병 데이터를 분석하여 개인화된 치료, 질병 조기 진단 및 예방 등 의료 서비스 개선에 활용됨

# 1. 빅데이터

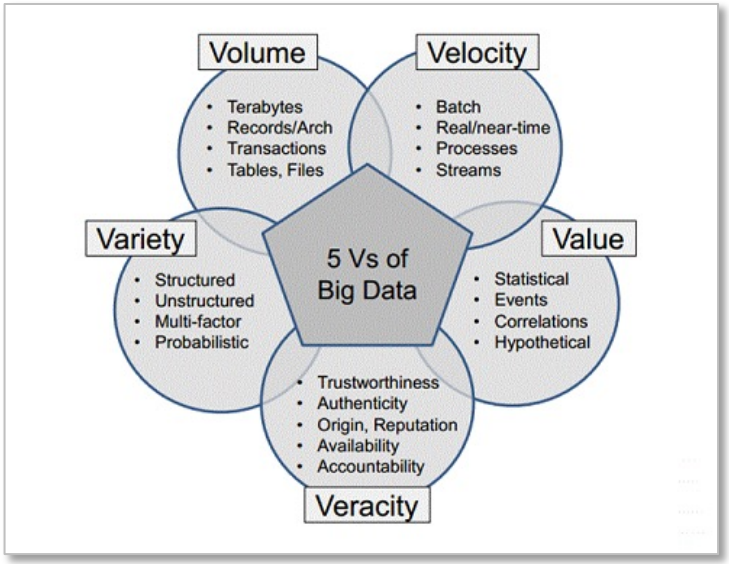
## 개요

### ❖ 빅데이터(Big data)의 정의

- 기존 데이터베이스가 저장하고 처리할 수 있는 범위를 넘어서는 대규모의 데이터
- 대규모 데이터를 저장하고 관리하는 기술과 데이터로부터 가치있는 정보를 추출하기 위한 분석 기술까지 포함

### ❖ 특징(5V)

- 볼륨(Volume):
  - 저장 및 관리되는 데이터의 양
- 속도(Velocity)
  - 시스템 내에서 데이터가 처리되는 속도
- 다양성(Variety)
  - 저장되는 데이터 유형의 다양성(정형, 비정형, 반정형 등)
- 가치(Value)
  - 데이터 분석을 통해 가치 있는 정보를 제공할 수 있는 정도
- 정확성(Veracity)
  - 데이터 및 정보 자산의 정확성 또는 신뢰성



# 1. 빅데이터

## 개요

### ❖ 데이터 vs. 빅데이터

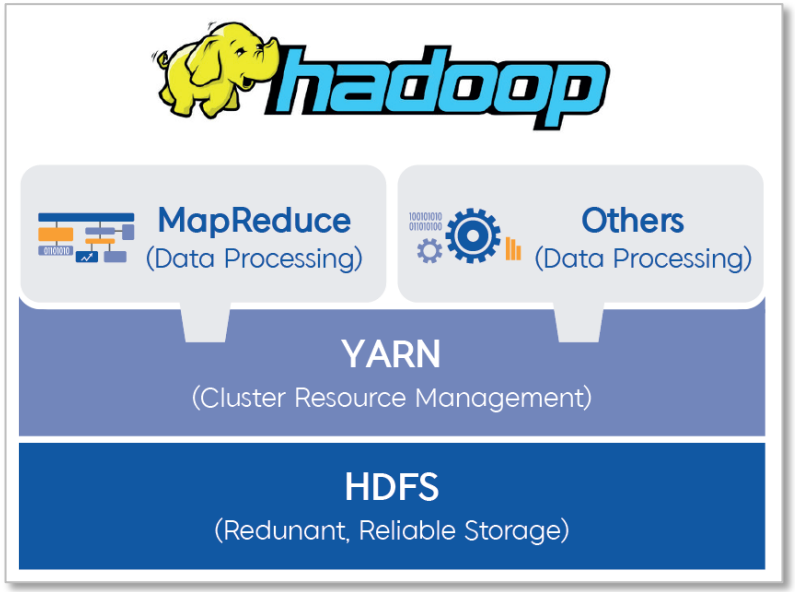
| | 데이터 | 빅데이터 |
|---|---|---|
| 데이터 유형 | 정형화된 문자, 수치형 데이터 중심 | 정형, 반정형, 비정형 데이터 등 |
| 데이터베이스 관리 시스템 | • 관계형 데이터베이스(RDBMS)<br>  • Orcale, MySQL 등 | • 분산 처리 등 대용량 데이터의 고속 처리를 지원하는 데이터베이스<br>  • 하둡, NoSQL, NewSQL 등 |
| 분석 기법 | • 통계 분석 및 데이터마이닝 | • 머신러닝 및 인공지능 기법<br>  • 자연어 처리, 영상 분석 등 |
| 관련 소프트웨어 | • 통계 분석 기반 소프트웨어<br>  • SAS, SPSS 등 | • 대규모 데이터 처리가 용이한 프로그래밍 언어<br>  • Python, R |

# 1. 빅데이터

## 빅데이터 분석을 위한 데이터베이스 시스템

### ❖ 하둡(Hadoop)

- 여러 대의 컴퓨터에 의한 분산 처리를 통해 대용량 데이터를 저장 및 처리하는 Java 기반의 오픈 소스 프레임워크
  - 처리 속도 향상을 위해 여러 대의 컴퓨터를 클러스터로 묶고, 데이터를 분산하여 여러 클러스터에서 병렬로 동시에 분산 처리를 수행함
- 오픈 소스이며 처리 속도가 빠르다는 이점으로 빅데이터 저장 및 처리를 위한 사실상의 표준으로 간주됨

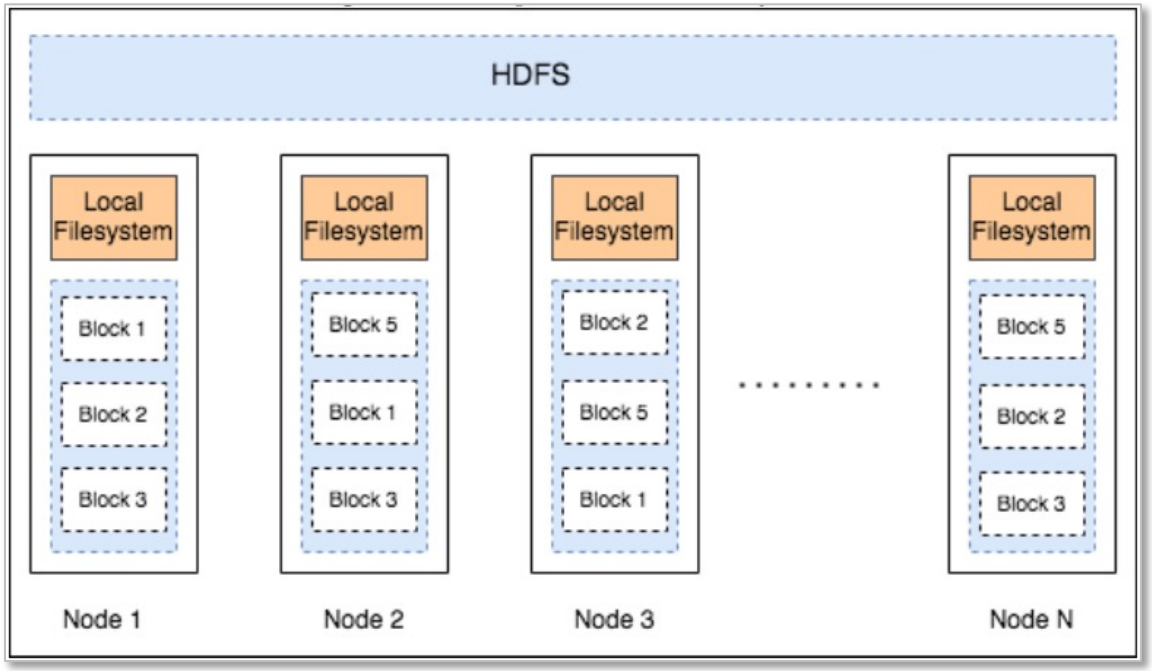


# 1. 빅데이터

## 빅데이터 분석을 위한 데이터베이스 시스템

### ❖ 하둡(Hadoop)

- 분산 파일 시스템 HDFS (Hadoop distributed file system)
  - 대용량 데이터를 수 천대의 분산된 장비의 물리적 블록에 나누어 저장
  - 병행제어 등의 문제가 없어서 고성능 데이터 처리가 가능
  - 동일한 데이터를 3군데 중복하여 분산 저장 → 장비 고장에 유연한 대처 가능

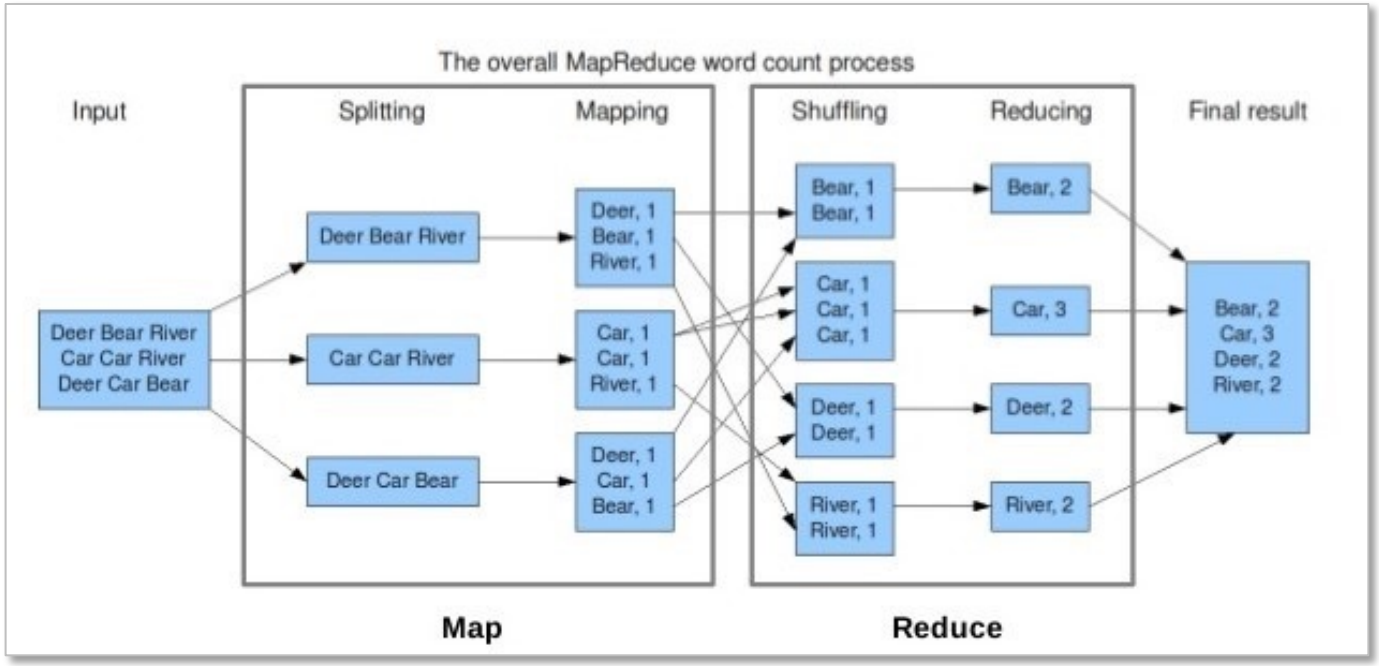


# 1. 빅데이터

## 빅데이터 분석을 위한 데이터베이스 시스템

### ❖ 하둡(Hadoop)

- 분산 처리 시스템 MapReduce
  - 분할 정복(divide and conquer) 전략 사용
  - 맵(map): 흩어져 있는 데이터를 정렬하여 키-밸류 단위로 데이터를 묶어줌
  - 리듀스(reduce): 키-밸류 형태의 데이터를 받아 같은 키 값을 가진 데이터를 하나의 결과로 요약함

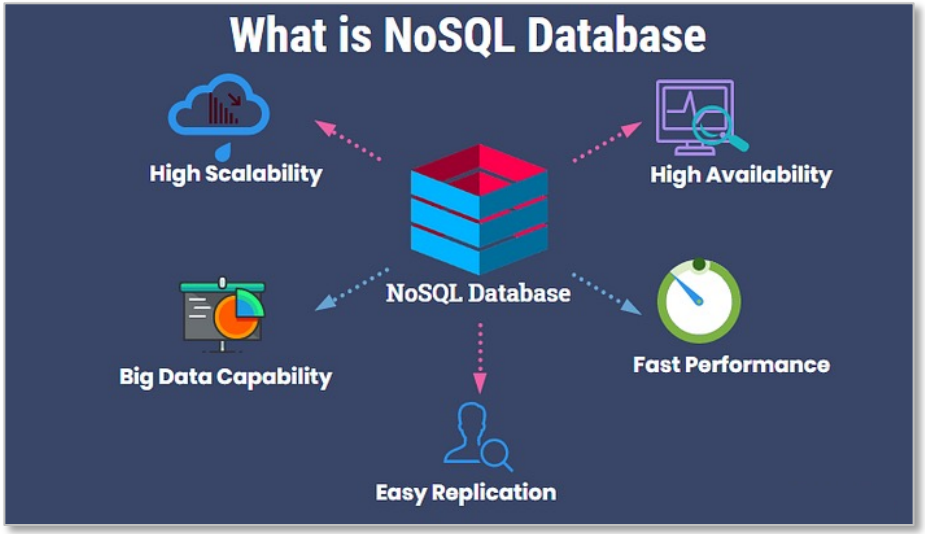


# 1. 빅데이터

## 빅데이터 분석을 위한 데이터베이스 시스템

### ❖ NoSQL

- 기존의 관계형 데이터 모델 및 SQL을 기반으로 하지 않는 데이터베이스 관리 시스템의 총칭
- 데이터의 일관성보다는 가용성 및 확장성에 중점을 둠
- 비정형 데이터 저장 및 처리를 위한 유연한 데이터 모델을 지원
- 분산 및 병렬 처리를 지원하여, 관계형 데이터베이스에 비해 대용량 데이터 저장 및 처리를 더 저렴한 비용으로 수행 가능
- NoSQL 예시: MongoDB, Hbase, GoogleBigTable, Neo4j 등



# 1. 빅데이터

## 빅데이터 분석을 위한 데이터베이스 시스템

### ❖ NewSQL

- 전통적인 RDBMS의 대안으로 떠오른 NoSQL은 대용량 데이터 처리를 위해 데이터의 중복과 분산 처리를 우선시함 → 데이터 무결성을 유지하기 위해 ACID를 만족하는 새로운 형태의 데이터베이스 관리 시스템의 개발이 요구됨
  - ACID: 데이터 무결성을 유지하기 위한 원자성(Atomicity), 일관성(Consistency), 고립성(Isolation), 지속성(Durability) 4가지 성질
- 관계형 데이터베이스의 확장성과 성능의 한계를 극복하고자 관계형 데이터베이스와 NoSQL의 장점을 취하여 개발됨
  - SQL을 주 인터페이스로 채택, ACID 만족, 분산 데이터 처리 가능
  - 둘 모두를 완전히 대체할 수 있는 수준에 도달하지 못하여, 아직까지 시장에서 자리잡은 제품이 없음
- 예시: ClusterixDB, NuoDB

# 2. NoSQL

## 개요

### ❖ 등장배경

- 관계형 데이터베이스를 대체할 새로운 대안의 필요성
  - 관계형 데이터베이스는 정형 데이터를 주로 처리 → 빠른 속도로 대량 생산되는 다양한 유형의 비정형 데이터를 저장 및 관리하는데 적합하지 않음
  - 관계형 데이터베이스는 단일 컴퓨터 환경에서 주로 사용되어 확장성 측면에서 비효율적

### ❖ 정의 및 특징

- ACID를 위한 트랜잭션 기능을 제공하지 않는 대신 저렴한 비용으로 대용량 데이터에 대한 분산 저장 및 병렬 처리가 가능한 데이터베이스
- 관계형 데이터 모델에서 벗어나 다양한 데이터 형태에 적합한 데이터 모델을 사용
- 고정된 스키마를 사용하지 않으므로, 데이터 구조를 미리 정의하지 않아도 되며 필요에 따라 구조를 바꿀 수 있어 비정형 데이터를 저장하기에 적합
- 각 데이터들이 다른 데이터와 관계없이 개별적으로 추가되어 중복 데이터가 발생할 수 있음
  → 각각 필요한 정보를 원하는 구조로 원하는 만큼 삽입할 수 있음

# 2. NoSQL

## 개요

### ❖ RDBMS vs. NoSQL

| | RDBMS | NoSQL |
|---|---|---|
| 처리 데이터 | • 정형 데이터 | • 정형, 반정형, 비정형 데이터 |
| 대용량 데이터 지원 | • 대용량 데이터 처리가 어려움 | • 분산 저장 방식을 통해 대용량 데이터 처리 가능 |
| 스키마 | • 데이터의 종류와 관계를 나타내는 스키마가 미리 정의되어야 함 | • 스키마가 고정되어 있지 않고, 변경이 자유로움 |
| 트랜잭션 | • 트랜잭션을 통해 데이터 일관성 유지를 보장 | • 트랜잭션을 지원하지 않아 데이터 일관성 유지를 보장하지 못함 |
| 검색 기능 | • 조인 등 복잡한 검색 기능 제공 | • 단순 검색 기능 제공 |
| 확장성 | • 클러스터 환경에 부적합 | • 클러스터 환경에 적합 |
| 활용 영역 | • 정형 데이터를 주로 다루며, 데이터 일관성이 중요하고 복잡한 질의 처리가 필요한 시스템(예: 인사, 회계 자료 등) | • 빠른 속도로 대용량 데이터가 생성되어 수정보다는 삽입 연산이 주로 일어나는 시스템(예: SNS, CCTV, 센싱 데이터 등) |

# 2. NoSQL

## 종류

### ❖ 키-값(Key-value) 데이터베이스

- 키-밸류 쌍의 집합체로 데이터를 저장하는 방식
  - 키(Key): 데이터의 식별자 역할
  - 밸류(Value): 데이터의 실질적인 내용. 문자, 숫자, 이미지, XML 문서 등
- 데이터의 의미 파악은 응용 프로그램에서 이루어지며, 데이터베이스는 특정 키에 해당하는 밸류만 저장함 → 데이터 처리 속도가 빠르고 확장이 용이함
- 검색, 저장, 삭제 3가지의 단순한 데이터베이스 조작 기능만 존재
- 대표 사례: Dynamo, Redis 등

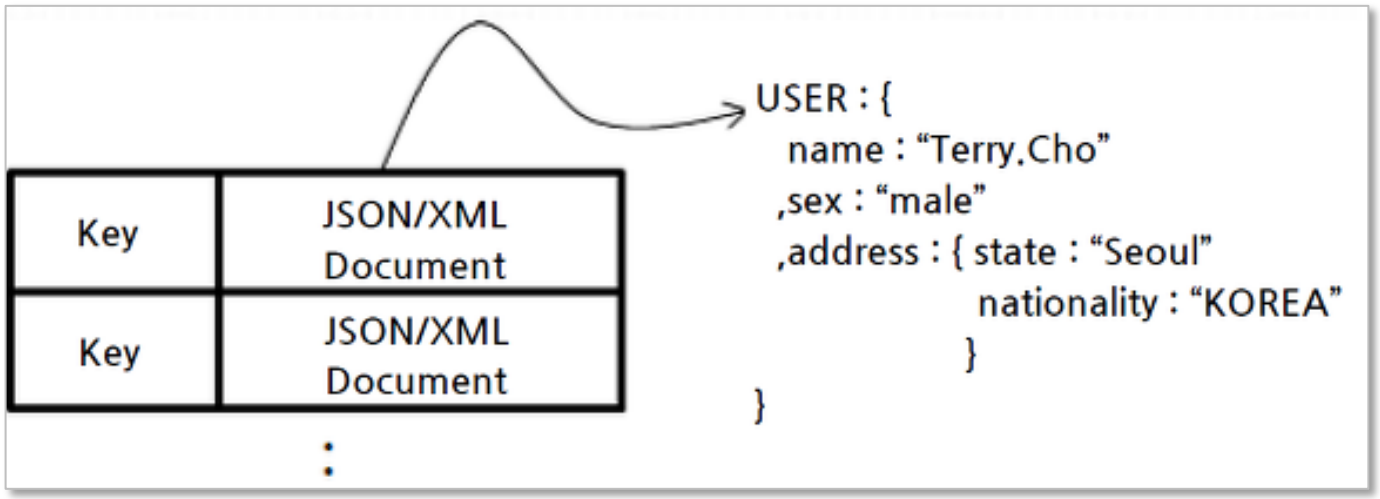
| Key | Value |
|---|---|
| K1 | AAA,BBB,CCC |
| K2 | AAA,BBB |
| K3 | AAA,DDD |
| K4 | AAA,2,01/01/2015 |
| K5 | 3,ZZZ,5623 |

# 2. NoSQL

## 종류

### ❖ 문서 기반(Document-based) 데이터베이스

- 키-밸류와 비슷한 구조이며, 이를 확장하여 키와 문서(document)의 쌍으로 데이터를 저장하는 방식
  - 문서(document): 계층적 구조가 존재하는 JSON, XML 등의 반정형 형태로 저장된 문서 데이터. 객체 지향 모델에서 객체(object)와 유사한 개념
- 키-밸류 데이터베이스와 유사하게 특정 키에 해당하는 문서 전체를 검색하는 것도 가능하지만, 문서 대상 쿼리(XQuery)를 활용하여 문서 내의 일부에 대한 검색도 가능
- 대표 사례: MongoDB, CouchDB 등



# 2. NoSQL

## 종류

❖ 컬럼 기반(Column-based) 데이터베이스

- 행(row)을 중심으로 데이터를 저장하는 관계형 데이터베이스와 달리 열(column)을 중심으로 데이터를 저장하는 방식
  - 행 중심: 하나의 데이터 블록이 하나의 행에 해당 → 특정 레코드에 대한 모든 속성 포함
  - 열 중심: 하나의 데이터 블록이 하나의 열에 해당 → 특정 속성에 대한 모든 레코드 포함
- 대용량 데이터의 압축, 분산 처리, 집계 함수(SUM, COUNT, AVG 등) 처리, 확장성에 이점을 가짐
- 대표 사례: Hbase, GoogleBigTable 등

Row-oriented

| ID | Name | Grade | GPA |
|---|---|---|---|
| 001 | John | Senior | 4.00 |
| 002 | Karen | Freshman | 3.67 |
| 003 | Bill | Junior | 3.33 |

Column-oriented

| Name | ID |
|---|---|
| John | 001 |
| Karen | 002 |
| Bill | 003 |

| Grade | ID |
|---|---|
| Senior | 001 |
| Freshman | 002 |
| Junior | 003 |

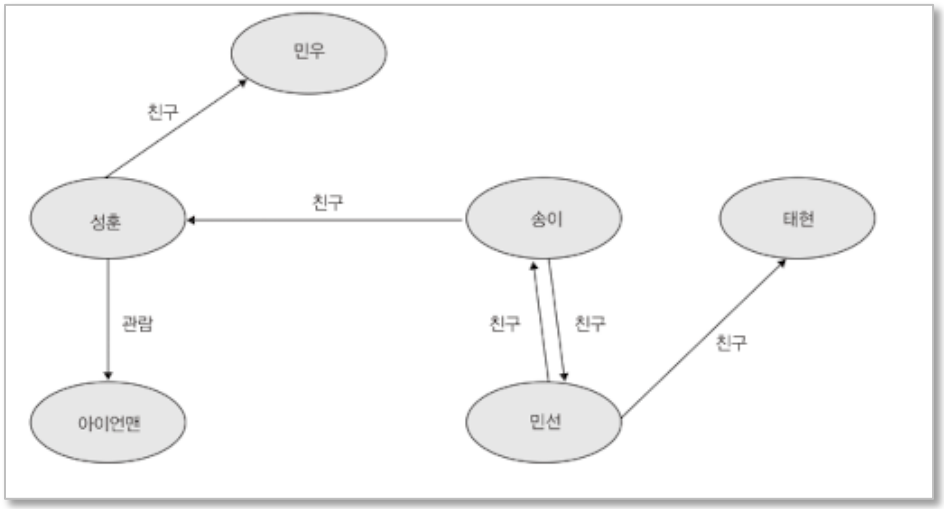
| GPA | ID |
|---|---|
| 4.00 | 001 |
| 3.67 | 002 |
| 3.33 | 003 |

# 2. NoSQL

## 종류

### ❖ 그래프 기반(Graph-based) 데이터베이스

- 그래프 이론(Graph theory)에 기초하여, 특정 개체에 대한 데이터를 노드(node)에 저장하고 개체 간 관계를 엣지(edge; 간선)을 통해 표현하는 방식
  - 엣지는 시작 노드, 끝 노드, 유형, 방향(일방향 또는 양방향)을 가지며, 하나의 노드가 가질 수 있는 관계의 수와 종류에는 제한이 없음
- 쿼리는 그래프 순회(Graph traversal)를 통해 처리됨
- 특히 개체 사이 관계가 중요한 경우에 주로 사용됨
  - 예시: SNS 분석, 질병 관련 접촉자 추적, 추천 시스템 등
- 대표 사례: Neo4j, OrientDB 등



# 2. NoSQL

## 실제 활용 예시

### ❖ USPTO 특허 데이터베이스 – MongoDB

- MongoDB 연결

In [1]:
```
import pymongo
```
executed in 201ms, finished 17:27:23 2023-12-05

In [2]:
```
# 데이터베이스 연결
connection = pymongo.MongoClient('localhost', 27017)
client = pymongo.MongoClient()
print(client)
```
executed in 11ms, finished 17:27:23 2023-12-05

```
MongoClient(host=['localhost:27017'], document_class=dict, tz_aware=False, connect=True)
```

In [3]:
```
# DB 목록
client.list_database_names()
```
executed in 120ms, finished 17:27:23 2023-12-05

Out[3]:
```
['admin', 'config', 'local', 'uspto']
```

In [4]:
```
# Collection (테이블) 목록
client.uspto.list_collection_names()
```
executed in 72ms, finished 17:27:24 2023-12-05

Out[4]:
```
['citations', 'citations_mini', 'USPAT_mini', 'USPAT']
```

# 2. NoSQL

## 실제 활용 예시

### ❖ USPTO 특허 데이터베이스 – MongoDB

- Collection (테이블)에 저장된 Document (레코드) 출력

```
In [5]: # USPAT collection에 저장된 document 출력
        db = client["uspto"]
        collection = db["USPAT"]
        collection.find_one()

executed in 112ms, finished 17:27:24 2023-12-05

Out[5]: {'_id': ObjectId('640448527058e9b008e7c21b'),
         'guid': 'US-11565042-B1',
         'publicationReferenceDocumentNumber': '11565042',
         'compositeId': '1000000383486!US-US-11565042',
         'publicationReferenceDocumentNumber1': '11565042',
         'datePublishedKwicHits': None,
         'datePublished': '2023-01-31T00:00:00Z',
         'inventionTitle': 'Anesthesia and/or sedation system and method',
         'type': 'USPAT',
         'mainClassificationCode': '1/1',
         'applicantName': ['Bibian; Stéphane', 'Zikov; Tatjana', 'Barua; Sankar'],
         'assigneeName': ['NeuroWave Systems Inc.'],
         'uspcFullClassificationFlattened': None,
         'ipcCodeFlattened': 'A61B5/369;A61M5/172;A61M5/142',
         'cpcInventiveFlattened': 'A61M5/1723;A61B5/369',
         'cpcAdditionalFlattened': 'A61M2005/14296;A61M2005/14208',
         'applicationFilingDate': ['2013-08-08T00:00:00Z'],
         'applicationFilingDateKwicHits': None,
         'relatedApplFilingDate': ['2012-08-08T00:00:00Z'],
```

```
In [6]: # citation collection에 저장된 document 출력
        collection = db["citations"]
        collection.find_one()

executed in 92ms, finished 17:27:24 2023-12-05

Out[6]: {'_id': ObjectId('64097a1b01e2bf0d26e37fd8'),
         'cited': '5741211',
         'citing': '11565042'}
```

# 2. NoSQL

## 실제 활용 예시

### ❖ USPTO 특허 데이터베이스 – MongoDB

- 'country' 컬럼이 'US'인 특허 추출

In [7]:
```
# 'country' 컬럼이 'US'인 특허 추출
for i, doc in enumerate(db.USPAT.find({'country': 'US'}, {'guid': 1, 'country': 1, '_id': 0})):
    print(doc)
    if i > 10: break

# [SQL]
# SELECT guid, country
# FROM USPAT
# WHERE country = 'US';
```

executed in 142ms, finished 17:27:24 2023-12-05

```
{'guid': 'US-11565042-B1', 'country': 'US'}
{'guid': 'US-11565025-B1', 'country': 'US'}
{'guid': 'US-11565470-B1', 'country': 'US'}
{'guid': 'US-11568442-B1', 'country': 'US'}
{'guid': 'US-11568982-B1', 'country': 'US'}
{'guid': 'US-11568388-B1', 'country': 'US'}
{'guid': 'US-11568389-B1', 'country': 'US'}
{'guid': 'US-11568038-B1', 'country': 'US'}
{'guid': 'US-11568446-B1', 'country': 'US'}
{'guid': 'US-11568377-B1', 'country': 'US'}
{'guid': 'US-11568301-B1', 'country': 'US'}
{'guid': 'US-11568863-B1', 'country': 'US'}
```

# 2. NoSQL

## 실제 활용 예시

### ❖ USPTO 특허 데이터베이스 – MongoDB

- IPC에 G06F가 포함된 특허 추출

In [8]:
```
# IPC에 G06F가 포함된 특허 추출
for i, doc in enumerate(db.USPAT.find({'ipcCodeFlattened': {"$regex": 'G06F'}},
                                      {'ipcCodeFlattened': 1, 'inventionTitle': 1, '_id': 0}).limit(10)):
    print(doc,"\n")

# [SQL]
# SELECT inventionTitle, ipcCodeFlattened
# FROM USPAT
# WHERE ipcCodeFlattened LIKE '%G06F%';
```

executed in 59ms, finished 17:27:24 2023-12-05

```
{'inventionTitle': 'Machine learning in resource-constrained environments', 'ipcCodeFlattened': 'G06N20/00;G06F8/
65'}

{'inventionTitle': 'Applied artificial intelligence technology for narrative generation based on explanation comm
unication goals', 'ipcCodeFlattened': 'G06F40/30;G06F40/295'}

{'inventionTitle': 'Semi-structured data machine learning', 'ipcCodeFlattened': 'G06N20/00;G06F16/835'}

{'inventionTitle': 'Tree-based format for data storage', 'ipcCodeFlattened': 'G06F16/22;G06F16/28'}

{'inventionTitle': 'Virtual storage interface', 'ipcCodeFlattened': 'G06F16/188;G06Q40/00'}

{'inventionTitle': 'Copying buckets from a remote shared storage system to memory associated with a search node f
or query execution', 'ipcCodeFlattened': 'G06F16/901;G06F16/2458'}

{'inventionTitle': 'Event-based debug, trace, and profile in device with data processing engine array', 'ipcCodeF
lattened': 'G06F13/10;G06F13/16'}

{'inventionTitle': 'Blockchain network based on machine learning-based proof of work', 'ipcCodeFlattened': 'G06N2
```

# 2. NoSQL

## 실제 활용 예시

### ❖ USPTO 특허 데이터베이스 – MongoDB

- 특허 문서 수가 100개 이상인 특허 클래스 조합별 특허 문서 수와 평균 청구항 수 추출

In [11]:

```
# 특허 문서 수가 100개 이상인 특허 클래스 조합별 특허 문서 수와 평균 청구항 수 추출
pipeline = list()
pipeline.append({'$group': {'_id': '$ipcAllMainClassification', 'patent_Count': {'$sum': 1},
                            'average_Claims': {'$avg': {'$toDecimal': '$numberOfClaims'}}}})
pipeline.append({'$addFields': {'average_Claims': {'$round': ['$average_Claims', 2]}}})
pipeline.append({'$match': {'patent_Count': {'$gte': 100}}})
print(pipeline, "\n")

results = db.USPAT_mini.aggregate(pipeline)
for i, doc in enumerate(results):
    print(doc)
    if i > 10: break

# [SQL]
# SELECT ipcAllMainClassification, patent_Count, average_Claims
# FROM
# (
#    SELECT ipcAllMainClassification, COUNT(*) AS patent_Count, AVG(numberofClaims) AS average_Claims
#    FROM USPAT
#    GROUP BY ipcAllMainClassification
# )
# WHERE patent_Count >= 100;
```

executed in 4.00s, finished 17:29:03 2023-12-05

# 2. NoSQL

## 실제 활용 예시

### ❖ USPTO 특허 데이터베이스 – MongoDB

- 특허 문서 수가 100개 이상인 특허 클래스 조합별 특허 문서 수와 평균 청구항 수 추출

```
[{'$group': {'_id': '$ipcAllMainClassification', 'patent_Count': {'$sum': 1}, 'average_Claims': {'$avg': {'$toDecim
al': '$numberOfClaims'}}}}, {'$addFields': {'average_Claims': {'$round': ['$average_Claims', 2]}}}, {'$match': {'pa
tent_Count': {'$gte': 100}}}]

{'_id': ['G06Q', 'H04L', 'G06F'], 'patent_Count': 178, 'average_Claims': Decimal128('19.25')}
{'_id': ['H04L', 'H04W'], 'patent_Count': 983, 'average_Claims': Decimal128('18.55')}
{'_id': ['H04L', 'H04B'], 'patent_Count': 138, 'average_Claims': Decimal128('19.51')}
{'_id': ['H04L', 'G06F'], 'patent_Count': 1153, 'average_Claims': Decimal128('18.66')}
{'_id': ['G11C', 'H01L'], 'patent_Count': 176, 'average_Claims': Decimal128('17.53')}
{'_id': ['H04N', 'G06F'], 'patent_Count': 157, 'average_Claims': Decimal128('16.24')}
{'_id': ['H01M'], 'patent_Count': 586, 'average_Claims': Decimal128('13.31')}
{'_id': ['G03G'], 'patent_Count': 129, 'average_Claims': Decimal128('15.62')}
{'_id': ['G02B'], 'patent_Count': 176, 'average_Claims': Decimal128('16.59')}
{'_id': ['G06N', 'G06F'], 'patent_Count': 410, 'average_Claims': Decimal128('18.79')}
{'_id': ['G06N'], 'patent_Count': 122, 'average_Claims': Decimal128('18.34')}
{'_id': ['E21B'], 'patent_Count': 259, 'average_Claims': Decimal128('17.15')}
```

# Thank you

UNIST 융합경영대학원
이규민(Gyumin Lee)
glee.optimizt@gmail.com

ULSAN NATIONAL INSTITUTE OF
SCIENCE AND TECHNOLOGY