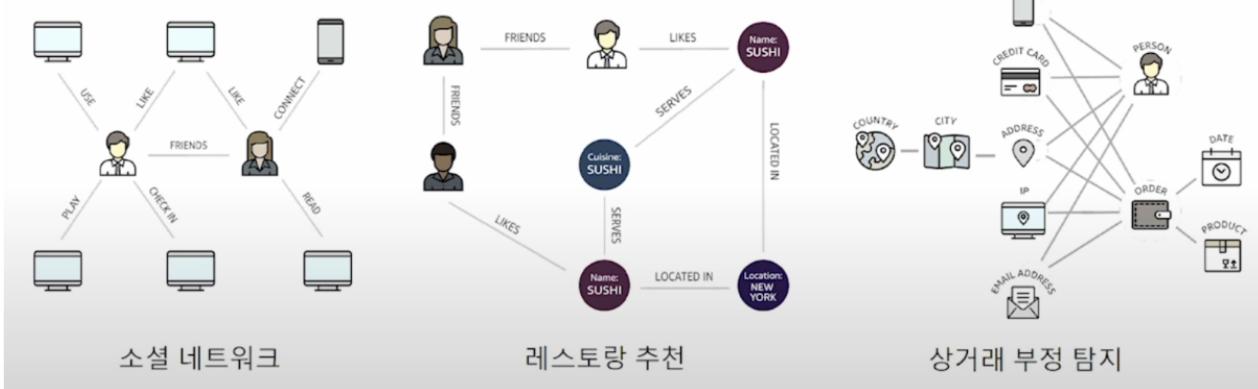


Graph Database

1. GDB란 무엇인가?

■ 설명

- 데이터를 표현하고 저장하기 위해 Node, Edge, Property이 있는 SemanticQuery(시맨틱쿼리)에 그래프 구조를 사용하는 데이터베이스
 - 시맨틱쿼리란?
 - 데이터에 포함된 구문, 의미 및 구조 정보를 기반으로 명시 또는 암시적으로 파생된 정보 모두를 검색할 수 있는 쿼리.
- NoSQL의 일종으로 관계형 데이터베이스의 한계를 극복하기 위해 만들어진 데이터베이스
- 사용 연결성이 높은 데이터 활용 **Highly connected**



- 데이터 끼리의 관계가 있음은 물론(RDB 또한 관계는 존재한다), 하지만 각 관계별로 패턴을 발견하고자 할때 그래프 데이터 베이스를 활용한다.
 - 빠르고 효과적으로 구현이 가능.

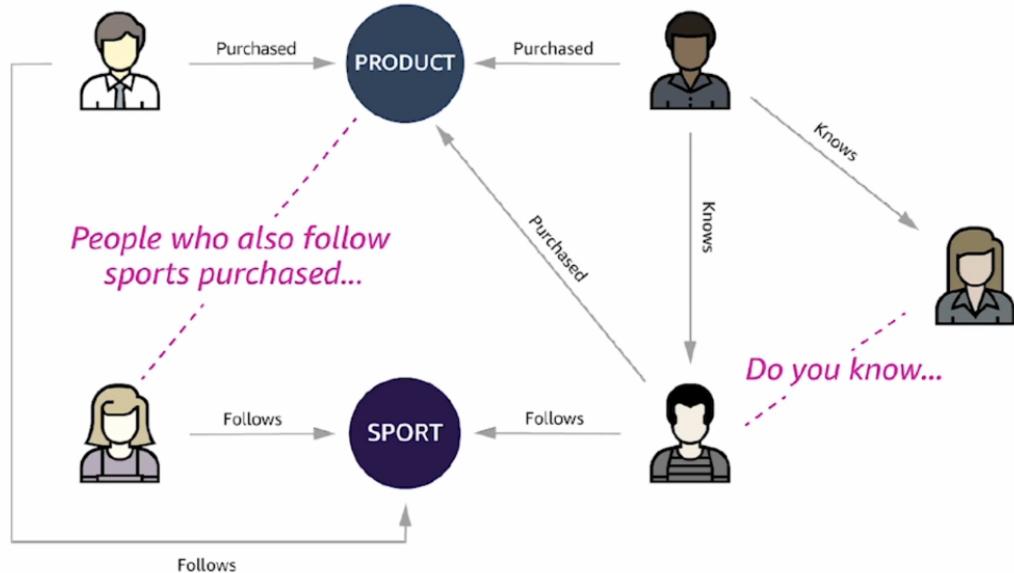
■ 표현



■ UseCases

- 소셜네트워크
- 추천 시스템

- 친구관계가 있는 주변인물의 구매 내역을 바탕으로 구매할 가능성이 높은 제품 추천



• 지식 그래프

- 각 데이터 간의 상호 연결된 정보의 모음
- 추가 정보**
- Google / WikiPedia

Google search results for "amazon aws":

Search term: amazon aws

Results:

- Amazon Web Services - Sign Up For Free & Access AWS**
Build, Deploy, and Manage Websites, Apps or Processes On AWS' Secure, Reliable Network. Sign Up for a Free Account & Experience AWS' Secure, Reliable, Scalable...
- AWS Cloud Compute Service**
AWS Lightsail: Easy-To-Use Platform For Simple Computing. 1 Year Free.
- AWS Compute Solutions**
Amazon EC2 is Secure & Free to Try. 750 hours/month for 1 Year.
- What is Cloud Computing?**
Benefits & Types of Cloud Computing Learn How AWS Can Help your Success
- Why Choose AWS?**
From Startups to Enterprises. Learn Why Millions of Customers Chose AWS

Right sidebar details for Amazon Web Services:

- 아마존 웹 서비스 (Amazon Web Services)**
- 웹사이트
- CEO: 아담 셀립스키 (2021년 5월 17일-)
- 본사: 미국 워싱턴 시애틀
- 창립: 2006년 3월 3일
- 창립자: 아마존
- 사이트 종류: 웹 서비스, 클라우드 컴퓨팅
- 자회사: 클라우드인더어, AWS 엘리멘탈, 더보기
- 모회사: 아마존
- 면허조회

Related searches:

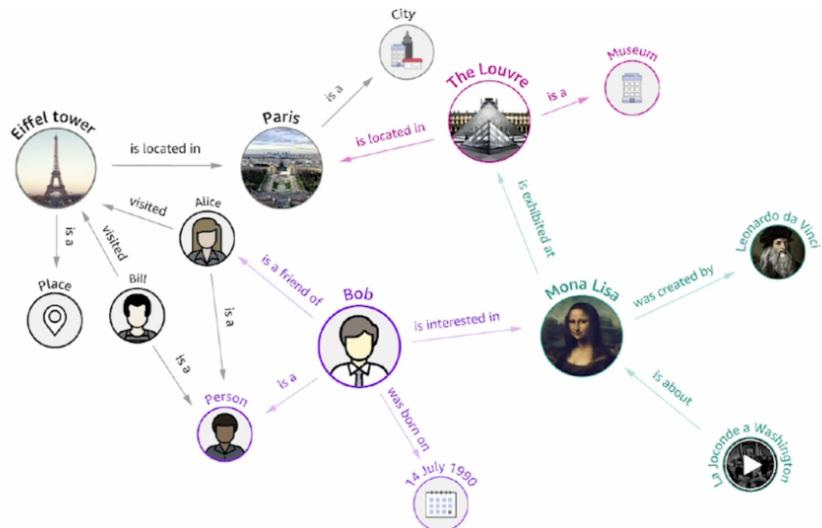
- Azure
- 구글 클라우드 플랫폼
- 깃허브
- 아마존

- Question & Answer

모나리자를 그린 사람은?

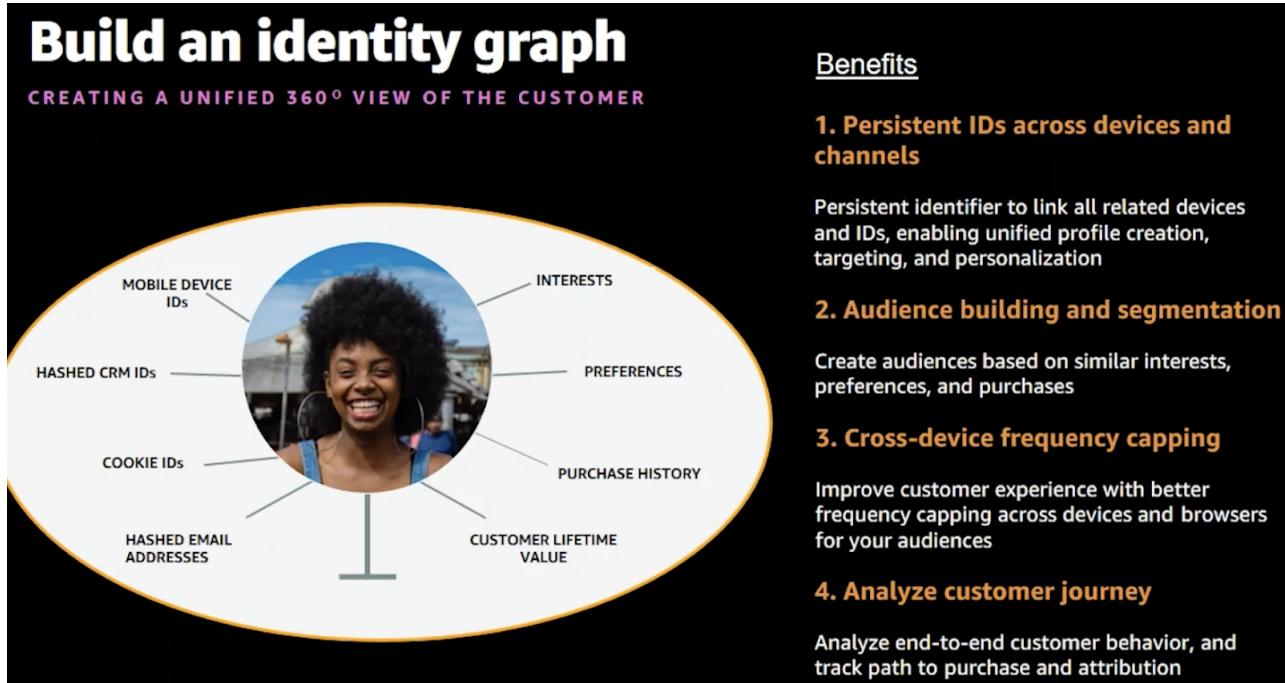
The Louvre에 작품이 있는 작가들은?

Alice가 Paris에 있는 동안 방문할 museums 들은?



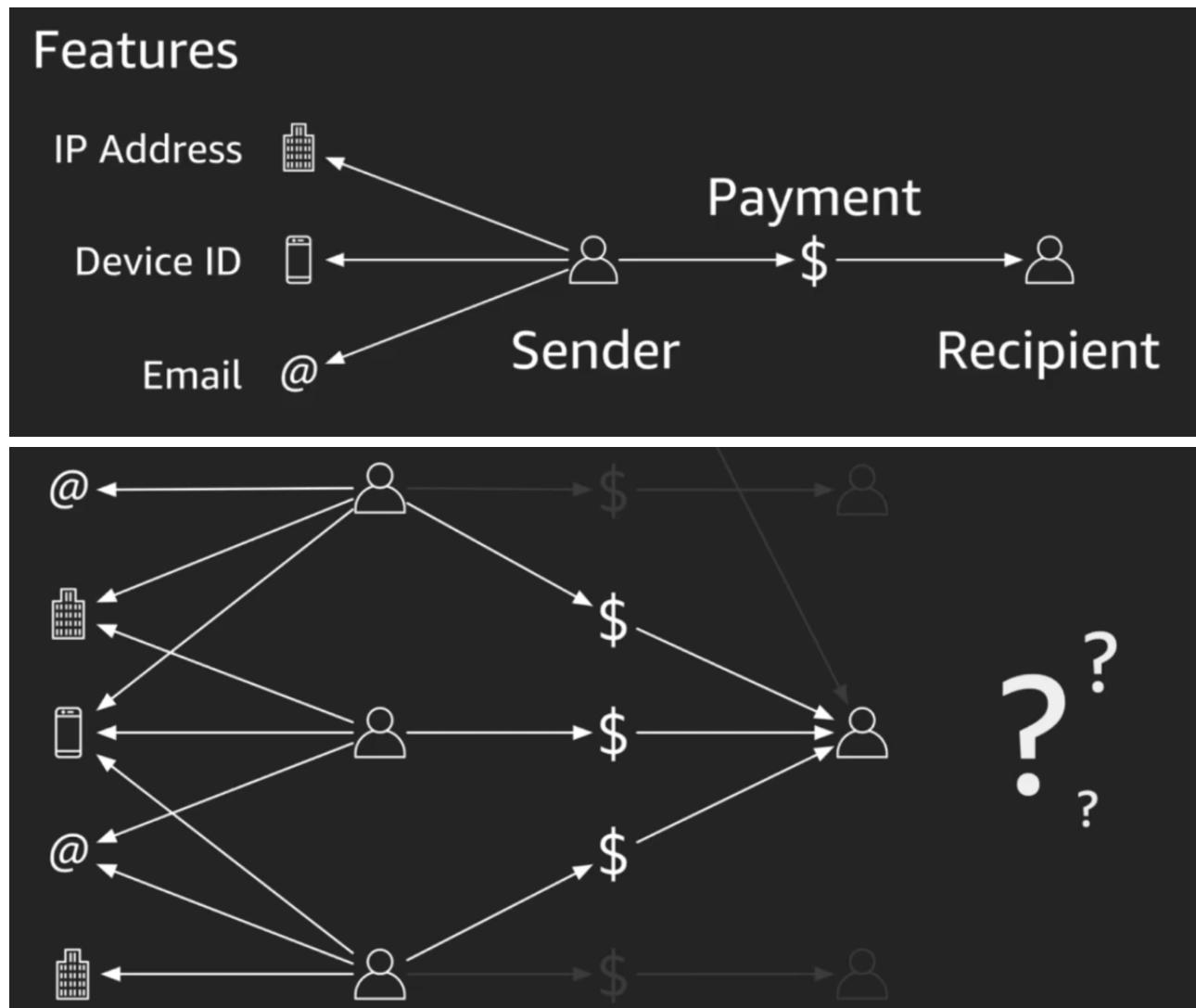
- 질문들에 대한 답변을 유추 할 수 있다.
- 모나리자 -> 다빈치
- 루브르 -> 특정 작품 -> 해당 작가
- Alice가 Paris에서 방문해야 할 박물관은?
 - Alice의 친구
 - 방문했던 박물관 또는 관심이 높은 박물관 선택
- 위와 같은 형태로 서로 관계를 가진 데이터를 바탕으로 정보를 검색

- ID(Identity) Graph



- 여러 채널에서 접근하는 하나의 고객의 정보를 고유한 식별자로 연결하여 통합된 고객 ID를 그리는 과정
- 고객이 온라인, 오프라인, 어떠한 디바이스, 어떠한 채널에서도 일관된 정보와 경험을 제공 받아야하는데 그것으로 해결하는데 사용하는 방법.

- 부정탐지



- 카드 또는 계좌 도용(기존 사용자의 정보와 다른 Device 또는 장소에서 카드를 사용하거나 계좌를 사용하게 되면 발견할 수 있다.)
- 별점 테러를 시행하는 사용자의 IP대역대가 어떤것인지?

- Machine Learning
 - 이 영화의 장르는?
 - 이 리뷰에는 몇점이 주어질까요?
 - 이 유저가 이영화를 리뷰할까요? 볼까요? 등의 정보를 머신러닝으로 예측이 가능하다
 - 자세한 사항의 위의 참고자료를 확인([AWS 자료](#))
- 생명과학
- IT 운용

■ 기존 관계형 데이터베이스와의 차이

▶ 기존 RDB에서의 어려움

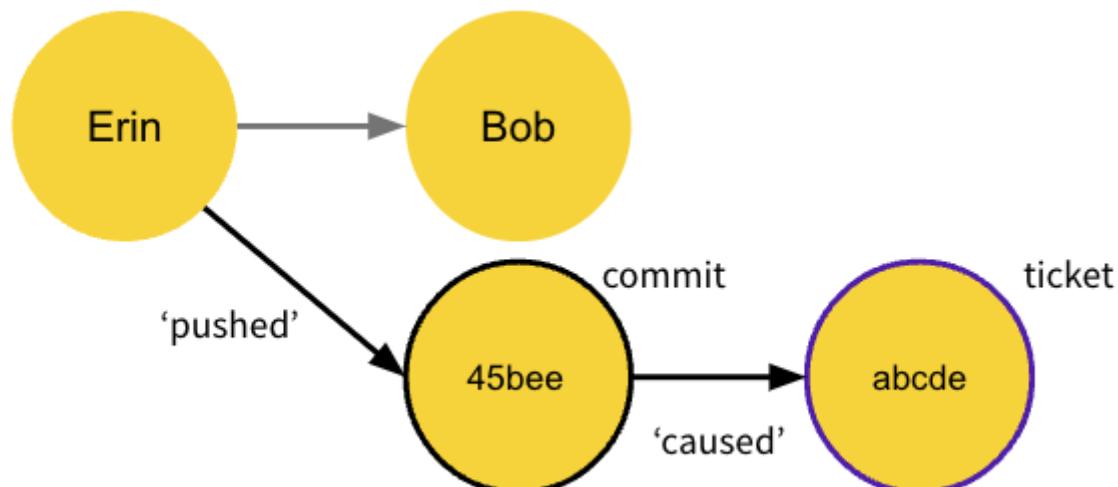


- 새로운 컬럼의 추가하기도 새로운 정보를 넣기 위해서는 새로운 테이블을 모델링하여 테이블 사이의 관계를 고민하여 만들어야함
 - GDB에서는 단순히 Node를 붙이고 Edge(Relation)만 설정하면 쉽게 추가가 가능하다.
 - Schema 자체가 Flexible하므로 다른 Node 자체 Schema를 고려하지 않아도 된다.

▶ 상호 연결성이 높은 데이터의 RDB에서 처리가 어려움



- RDB에서의 관계처리는 Join을 통해 수행, 다수의 관계를 가지게 된다면 다수의 Join을 통해 구현의 복잡함은 물론 시스템적인 부하가 발생한다.
 - SQL은 그래프 혁으로 Traversal 하게 구조가 된 언어가 아님.



- 예제

- How many tickets were created by Bob's direct reports?
 - sql

```

SELECT
  COUNT(DISTINCT(Ticket.ticketId))
  
```

```
FROM EmployeeHierarchy
INNER JOIN GitCommit ON ...
INNER JOIN Ticket ON ...
WHERE ...
```

- Gremlin(query for gdb)

```
g.V().has('name', 'Bob'). -- Bob에게
in('reportsTo'). -- 보고하는 사람중
out('pushed'). -- pushed edge를 가지고
out('caused').dedup().count() -- caused edge가진 unique한 숫자는
```

■ 장점

- 다양한 데이터 참조 질의에 유리.
 - RDB는 무수한 Join이 필요하고 데이터가 늘어날때마다 더욱 복잡해짐
 - GDB는 Edge를 통해 관계가 명확하고 질의에 노드와 엣지를 선언하면 간편하고 효율적으로 질의가 가능.
- 패턴에 매칭되는 노드만 가져오면 시작노드와 관계없는 노드는 고려하지 않으므로, 성능이 데이터의 크기와 독립적이다.
- 스키마 설정에 확장성이 높고 순환 조회 (Traversal)에 특화되어있다.
- 예)
 - Family Tree(족보) 조회시
 - Mysql : 5~10초
 - GDB (Neptune) : 300~500ms
- 쿼리가 직관적이다.

■ 선두 회사 / 프로그램

- 아직 뚜렷한 선두 회사 또는 플랫폼이 존재하지 않는다.
- 그럼에도 Neo4j, AWS Neptune 순으로 가장 많이 사용하고 있다.
 - Nasa, Airbnb, 포춘지 선정 500대 기업은 많은 수로 Neo4j를 선택.

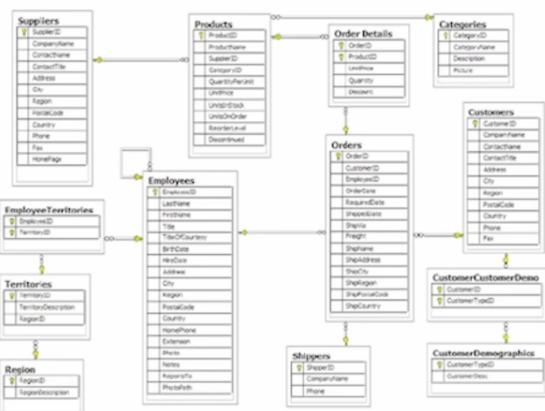
■ 선정 시 중요하게 생각해야 될 부분

- 데이터에 수많은 관계와 속성 값을 가져야한다.
- 각 데이터마다 다른 Property를 가지고 있다.
- 각각의 데이터의 여러 방식으로 관계를 가지고 있다.
- 순회(Traversal) 구조를 가지고 있다.

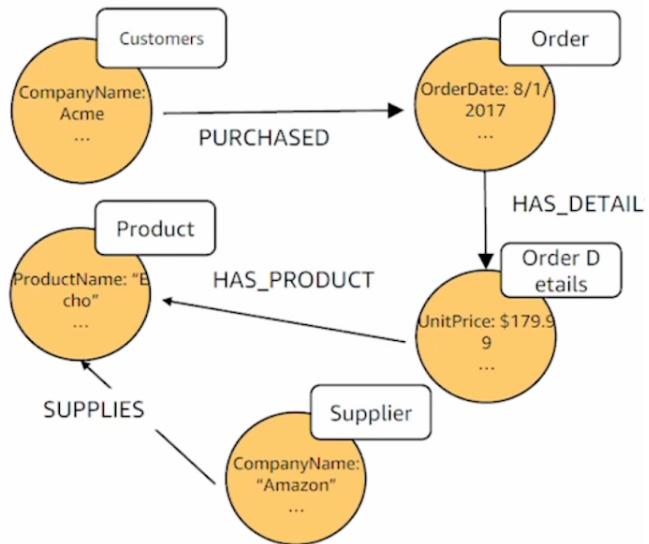
2. 모델링 & Query, Framework

■ 모델링

관계형 모델



그래프 모델



- RDB : 비즈니스 프로세스에 적합한 구조
- GDB : 관계 파악에 적합한 구조
 - 즉, GDB의 핵심은 Relation에 있음
- 하지만 GDB를 단독으로만 사용하는 산업은 없다.

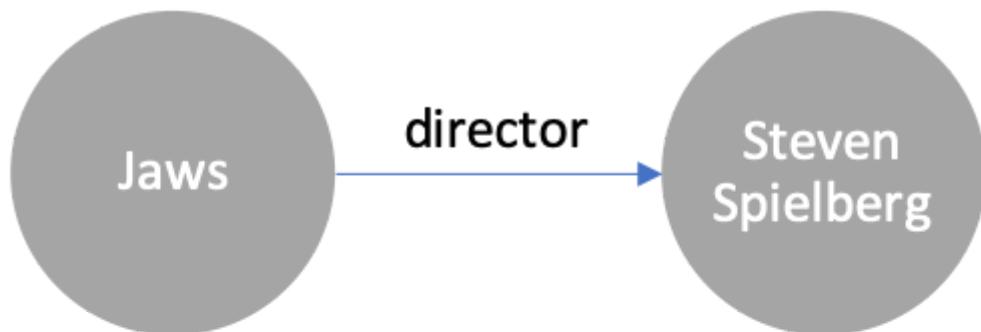
모델링 방법

- Vertices
 - Domain Entity
 - 사물, 이벤트, 컨셉 등
 - Label
 - 타입(Type) 또는 역할(Role)
 - 예) Person, Product)
 - Property
 - Entity의 속성 또는 Meta정보
 - 예) name, price 등
- Edge
 - Label
 - 관계 타입
 - Property
 - 관계의 Strength, Weight, Quality, Meta정보
 - 예) 거리, 생성일시 등

■ Property Graph



- Properties를 가진 Vertices와 Edges들의 묶음
 - Properties
 - Key - Value 구조의 정보
- Vertex
 - Entity와 Domain을 표현
 - Node와 같은 개념이며 Properties를 가진다.
 - 각 Vertex는 **Label**이라는 정보를 가진다.
 - 해당 Vertex를 Identify하는 정보. (Unique한 Key라고 보면 됨)
- Edge
 - 두개의 Vertices간의 직접적인 관계를 표현하는 정보
 - Properties를 가진다.
- Vertex와 Edge는 각각 unique한 ID를 가진다.
- Properties는 Edge와 Vertex간의 관계가 없는 정보를 가진다.
 - 해당 표현의 상세 정보라고 생각하면 됨.



| | |
|-----------------|-----------|
| id | tt0073195 |
| year | 1975 |
| rating | 8.0 |
| numvotes | 493275 |

| | |
|------------------|-----------|
| id | nm0000229 |
| birthyear | 1946 |

- Node : Jaws
- Property
 - id : tt0073195
 - year : 1975

- rating : 8.0
- numvotes : 493275
- Node :Steven Spielberg
- Property
 - id : nm0000229
 - birthyear : 1946
- Edge
- Director

■ Framework & Query Language

▶ Framework

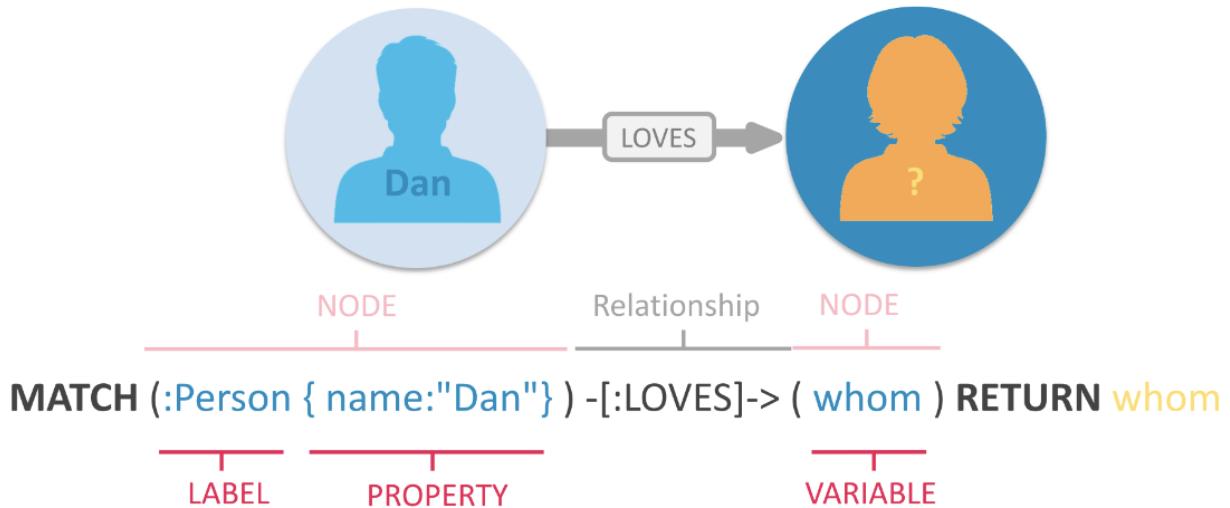
- Apache Thinkpop
 - Property graph를 위한 Opensource graph computing framework.
 - OLTP, OLAP에 모두 사용할 수 있다.
- Neo4j
 - Neo4j사가 개발한 그래프 데이터베이스 관리 시스템
 - 네이티브 그래프 저장 및 처리 기능을 갖춘 ACID를 준수하는 트랜잭션 데이터베이스로 기술
 - GDB 중에서 가장 대중적인 GDB이다

▶ Query Language

- Gremlin
 - 아파치 소프트웨어 재단의 아파치 텁커팝(Apache TinkerPop)이 개발한 그래프 순회 언어이자 가상 머신이다

```
--그램린의 친구의 친구의 이름들은?  
g.V().has("name","gremlin").  
out("knows").out("knows").values("name")  
==>[jack, peter, john .....]  
  
-- 그래프의 Vertices의 Label 별로 Grouping Count는  
gremlin> g.V().label().groupCount()  
==>[occupation:21, movie:3883, category:18, user:6040]  
  
-- 만들어진 가장 오래된 영화의 연도는  
gremlin> g.V().hasLabel('movie').values('year').min()  
==>1919
```

- Cypher



- 식으로 나타내는 프로퍼티 그래프의 효율적인 질의 및 업데이트를 허용하는 선언형 그래프 질의어
- 상대적으로 단순
- 처음에는 그래프 데이터베이스 Neo4j를 대상으로 Neo4j사가 만들었으나 2015년 10월 오픈사이트(openCypher) 프로젝트를 통해 개방

```

MATCH (nicole:Actor {name: 'Nicole Kidman'})-[:ACTED_IN]->
(movie:Movie)
WHERE movie.year < $yearParameter
RETURN movie

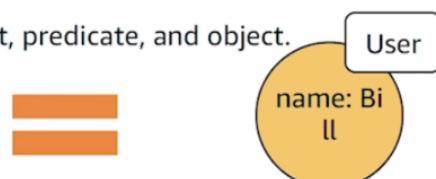
MATCH (start:Content)-[:RELATED_CONTENT]->(content:Content)
WHERE content.source = 'user'
OPTIONAL MATCH (content)-[r]-()
DELETE r, content

```

■ RDF Graph

- **RDF Graphs** are described as a collection of triples: subject, predicate, and object.

subject → <<http://www.socialnetwork.com/person#1>>
 predicate → rdf:type contacts:User;
 Object (literal) → contact:name: "Bill" .



- **Internationalized Resource Identifiers (IRIs)** uniquely identify subjects.

IRI → <<http://www.socialnetwork.com/person#1>>

- The Object can be an IRI or Literal.

- A Literal in RDF is like a property and RDF supports the XML data types.
- When the Object is an IRI, it forms an “Edge” in the graph.

subject → <<http://www.socialnetwork.com/person#1>>
 predicate → contacts:friend
 Object (IRI) → <<http://www.socialnetwork.com/person#2>> .



- [w3.org](#)
- Resource Description Framework
- Subject - Predicte - Object 3가지를 기반으로 함
- URI 형태로 Link 표현을 원어로 표현하여 더욱 정확하게 표현함
- RDF Triplestore
 - RDF Model을 지원하는 Database를 칭함.

information

Bob  Knows Alice 

- 모델
- 위 정보를 아래로 표현

```

<http://example.com/person/1> :hasName "Bob"
<http://example.com/person/2> :hasName "Alice"
<http://example.com/person/1> foaf:knows <http://example.com/person/2>
```

▶ Framework(RDF)

- [4Store](#)
- [ARC](#)
- [AllegroGraph](#)

▶ Query Language(RDF)

- SPARQL(for RDF)
 - 시맨틱 질의어로서 자원 기술 프레임워크(RDF) 형식으로 저장된 데이터를 검색, 조작
 - [참조](#)

```

-- "아프리카의 모든 국가 수도는?"이라는 질문의 SPARQL 쿼리 예제
PREFIX ex: <http://example.com/exampleOntology#>
SELECT ?capital
      ?country
WHERE
{
  ?x  ex:cityname      ?capital   ;
      ex:isCapitalOf    ?y        .
  ?y  ex:countryname   ?country   ;
      ex:isInContinent  ex:Africa .
}
```

3. AWS Service

Neptune

- AWS Neptune은 Thinker Pop 3.3.0(lastest version released 2017/10)을 완전히 호환한다.
- 지원 언어
 - Gremlin
 - SPARQL
 - 두가지다 사용이 가능

완전 관리형 서비스

- 콘솔에서도 쉽게 구성 가능
- Multi-AZ 고가용성
- 최대 15개 읽기 Cluster 사용 가능.(각 AZ마다)
- 저장 / 전송시 암호화
- 백업, 복구, 특정 시점으로 복원 (S3 Backup)

클라우드 네이티브 스토리지 엔진

- 3개 AZ에 6개의 Replica가 존재함
 - S3에 연속 백업
- 10GB단위로 세그먼트를 저장
- 스토리지 최대 64TB까지 지원

고가용성

- 4/6 쓰기 (6개 복제 본 중에 4개가 살아있을 경우)
- 3/6 읽기 (6개 복제 본 중에 3개가 살아있을 경우)
- 복구를 위한 Peer to Peer 복제
- DB 노드 장애 자동 감지 및 복구
- DB 프로세스 장애 자동 감지 및 재시작
- 사용자 지정된 순으로 Fail-Over

지속적 백업

- 주기적으로 스냅샷을 병렬로 수행 및 로그를 S3 스트리밍 전송
- 최대 64TB까지 사용하므로 큰 용량 백업이 필요
 - 지속적으로 S3로 전송하여 백업수행
 - 변경분에 대해 백업
- 일괄 백업의 경우 S3에 보관된 백업본을 합쳐서 수행하게 되므로 현재 사용하는 서비스와는 상관없이 백업

4. AWS Neptune 체크 포인트

- 완전 관리형 그래프 데이터베이스
- Apache TinkerPop 및 W3C RDF 그래프 모델 지원
- Gremlin, SPARQL 쿼리 언어 지원
- 손쉽게 콘솔에서 구성 가능
- Multi-AZ 고가용성
- 최대 15개 읽기 복제
- 저장 / 전송 시 암호화

- 백업 및 복구, 특정 시점으로 복원(point-in-time recovery)

5. Neptune을 활용한 인맥 추천 서비스

- 리멤버라는 회사에서 명함 정보를 입력 받아서 해당 Text를 추출하고 해당 내용을 Kinesis Stream을 이용하여 Lambda를 호출 후 Neptune으로 저장하여 추후 인맥 추천을 위해 Neptune를 사용하는 아키텍처이다.

