**NoSQL의 특징**

**1. Denormalization**

**“비정규화” = 데이터중복 허용**

쿼리 프로세싱을 간단하게 하거나 최적화하기 위해서, 혹은 사용자의 데이터를 특정한 데이터 모델에 맞추기 위해 같은 데이터를 여러 Document나 Table에 복제하여 중복하는 것을 허용한다.

**비정규화로 인한 trade-off**

Denormalization으로 인해 쿼리당 I/O(IO per query) 또는 쿼리 데이터사이즈(Query Data Volume)은 감소하지만 전체 데이터 사이즈(Total Data Volume)은 증가하는 단점이 있다

추가로, 쿼리 수행의 복잡도(Processing complexity)는 낮아지는 장점과 total data volume이 증가하는 단점이 있다

빅데이터 solution의 경우, data volume에 크게 연연하지 않기에, 확장성에 초점을 둔다.

**2. Aggregation**

“유연한 스키마” 속성은 복잡하고 다양한 구조의 내부 요소(Nested Entities)를 가지고 있는 데이터 클래스 구성을 가능하게 한다.

즉, 1:n 관계를 최소화하여 결과적으로 JOIN연산을 줄인다 – 수행시간 단축, 저렴한 비용의 대용량 데이터 지원이 가능

복잡하고 다양한 비즈니스 요소를 담을 수 있음 – 추후 확장성 및 변동성에 대한 유연한 대응 가능

3. Application Side Join

대용량 데이터에 대한 빠른 응답과 확장성, 가용성을 최우선 목적으로 하기에, 쿼리타임 조인을 최대한 피하여 데이터 모델을 구성하는 것을 가정함.

Join을 불가피하게 수행해야하는 경우 – 대상데이터가 다대다(n:m) 관계로, 복잡하게 얽혀있는 경우, Join 대상 데이터의 수시변동이 있을 경우

대상 데이터가 수시로 변동되는 경우, 비정규화와 Aggregation을 통해 많은 entity에서 해당 데이터의 중복을 허용했기 때문에, 이를 업데이트하기엔 많은 비용이 발생함.

이런 경우, 변경이 잦은 데이터만을 추려내어 쿼리 타임 조인을 수행하는 것이 대안

**파란글에 대한 Solution 예시**

변경이 잦은 데이터는 RDBMS에 저장을 하는 방식

RDBMS를 사용하지 않는 경우, NoSQL에 존재하는 두 Table은 내부에서는 Join이 안되기 때문에, 외부 Application으로 불러와 외부에서 Join하는 방식으로 처리한다.

|  |
| --- |
| V1 = Get(Key) from TABLE1  V2 = Get(TABLE.FK) from TABLE2  Return V1, V2 |

아직 이부분을 이해하지 못했음. 개인적인 의견이지만, 이러한 특성 때문에 데이터분석에서 적합하다고 여겨짐.

**주요 NoSQL 데이터 모델링 기법**

|  |
| --- |
| **General Modeling Techniques** |
| Atomic Aggregates |
| Enumerable Keys |
| Dimensionality Reduction |
| Index Table |
| **Composite Key Index** |
| Aggregation with Composite Keys |
| **Inverted Search – Direct Aggregation** |

**Composite Key**

* 하나 이상의 필드를 deliminator를 이용하여 구분지어 사용하는 방법
* Ordered KV Store의 경우에는 이를 이용하여 order by와 같은 sorting기능이나 grouping을 구현 가능
* NoSQL N개의 서버로 구성된 클러스로 동작하고, 데이터는 Key를 기준으로 N개의 서버에 나눠서 저장이 된다.
* Key를 선정할 때는 전체 서버에 걸쳐서 부하가 골고루 분산될 수 있는 Key를 선정함

|  |  |
| --- | --- |
| **Key** | **Value** |
| Windows:etc | … |
| Windows:programfile | … |
| Windows:system32 | … |
| Windows:temp | … |

**Inverted Search Index** – 검색엔진에서 많이 사용하는 방법

* Value의 내용을 key로 하고, key의 내용을 반대로 value로 하는 패턴
* 검색엔진은 사이트의 모든 페이지를 검색로봇이 검색해서 문서내의 단어들을 색인하여 URL에 맵핑하여 저장함
* 검색은 단어를 Key로 검색하기 때문에, value에 검색 키워드들이 들어가 있을 경우에는 효과적인 검색이 불가함
* 검색 키워드를 Key로 설정하여 URL을 value로 하는 테이블을 다시 만든 다음, 검색 키워드를 Key로 하여 검색을 하면 신속하게 검색 키워드를 가지고 있는 URL을 반환

**계층 데이터 구조 모델링 패턴**

* NoSQL은 다양한 데이터 모델이 있지만, 기본적으로 row, column을 가지고 있는 테이블 구조의 저장 구조를 가짐
* App 개발 중에는 이러한 테이블 구조 뿐만 아니라 Tree와 같은 계층형 데이터 구조를 저장해야 할 경우가 있는데, NoSQL은 이러한 계층형 구조를 저장하는 것이 쉽지않음
* RDBMS의 경우에도 이런 계층형구조를 저장하기 위해서 솔루션에서 기능적으로 자체 지원하기도 하며, 데이터 모델링을 통해서 계층형 구조를 저장할 수 있음
* NoSQL에서 계층형 구조를 저장하는 기법은 RDBMS에서 사용하는 기법들을 참고하여 구현함

**Tree Aggregation**

* Tree 구조 자체를 하나의 Value에 저장하는 방식
* JSON이나 XML 등을 이용하여 트리 구조를 정의하고, Value에 저장함
* Tree 자체가 크지 않고, 변경이 많이 없는 경우에 적합함

|  |
| --- |
| {  Comments : [  {by : “Mike”, message : “….”, replies : [  {by : “Sam”, message : “…”, replies : []}  ]},  {by : “Eliot”, message : “…”, replies : []}  ]  } |

Materialized Path