

IDG Tech Report

빅데이터 시대의 침병, NoSQL의 분류와 선택 기준

NoSQL은 Not only SQL의 줄인 말이다. 그동안 계층형, 네트워크형, 관계형으로 발전해 온 데이터베이스에서 대용량 데이터 처리가 이슈가 되면서 관계형 데이터베이스(RDBMS)로 해결하지 못하는 난제를 보완하고자 나온 것이 바로 NoSQL이다. 데이터를 여러 서버에 분산해 작업하는 방식인 NoSQL은 저가의 서버를 클러스터링 방식으로 나누어 처리할 수 있다. 그런데 NoSQL은 말 그대로 SQL이 아닐뿐, 종류가 너무나 많고, 단점 또한 하나둘씩 나타나고 있다. 특히 빅데이터 시대에는 과거의 RDBMS가 더 이상 적합하지 않는 데이터베이스 작업이 많이 있다. 빅데이터 시대를 맞이하는 침병 역할을 맡은 NoSQL에 대해 살펴보고 대안이 될 수 있는 NoSQL 가운데 적합한 제품을 선택해보자.

NoSQL 표준의 시대가 왔다

빅데이터와 NoSQL에 대한 숨겨진 진실

어떤 데이터베이스를 사용해야 할까?

- 키-값 쌍 데이터베이스
- 컬럼 패밀리/ 빅테이블 데이터베이스
- 문서 데이터베이스
- 그래프 데이터베이스

무단 전재 재배포 금지

본 PDF 문서는 IDG Korea의 프리미엄 회원에게 제공하는 문서로, 저작권법의 보호를 받습니다.
IDG Korea의 허락 없이 PDF 문서를 온라인 사이트 등에 무단 게재, 전재하거나 유포할 수 없습니다.

NoSQL 표준의 시대가 왔다

Andrew Oliver | InfoWorld

향후 15년동안 오라클의 하향세는 피할 수 없는 일이다. NoSQL과 빅 데이터 아래에 하나로 뭉친 새로운 데이터베이스의 물결이 다가오고 있는 상황에서 RDBMS 유일의 패러다임을 유지하기는 불가능하다.

오라클은 파트너십을 통해 대처하고 있고 이미 NoSQL 데이터베이스도 보유하고 있지만 오라클이 현재의 수익 흐름을 그대로 유지한 채 전환할 수 있을 것이라는 기대를 하긴 힘들다. 이는 1996년의 노벨과 비슷한 상황이다.

그러나 RDBMS(Relational Data Base Management System)가 쇠퇴하는 데에는 시간이 걸릴 것이다. 워낙 확립된 기술이기도 하지만 애초에 RDBMS의 광범위한 보급을 이끈 장점들이 당분간 계속 유지될 것이기 때문이다.

다만 트랜잭션(Transaction)은 이런 장점에 포함되지 않는다. 트랜잭션은 그동안 과대평가됐다. 너무 세밀한 탓에 복수의 요청/응답 사이클에서 그 유용성이 떨어지는 트랜잭션을 대부분의 애플리케이션에서 필수적인 도구라고 지칭하는 것은 합당하지 않다. 게다가 적절한 일관성을 보장할 수 있는 다른 방법들도 있다.

RDBMS의 가장 큰 장점은 트랜잭션이 아니라 표준화에 있다. RDBMS의 역사를 보면 미약한 표준화라 해도 아예 표준화가 없는 경우보다는 더 큰 시장을 창출했음을 알 수 있다. 사실 표준화는 새로운 종의 데이터베이스가 시장을 확대하는 데 있어 핵심적인 장벽이다.



새로운 데이터 시대

NoSQL의 전환은 필연적인 현상이다. 필자는 '뉴DB(NewDB)'라고 부르기를 좋아한다. RDBMS는 느린 10MB 하드 드라이브가 사용되고 기대 수준도 낮았던 시대에 만들어졌다.

반면 NoSQL은 인터넷 시대의 기술이다.

NoSQL은 스토리지가 저렴하고 성능과 확장성에 대한 기대치가 높은 시대에 맞게 만들어졌으며, 방대한 디지털 자료를 닥치는 대로 보관하는 사람들을 위해 설계됐다.

지금 비즈니스, 마케팅, 그리고 정보 기술의 흐름은 예견대 킴 카다시안이 정확히 어떤 사람이며, 사람들이 얼마나 그를 좋아하는지 필자가 알 수 있도록 해준다. 이 정보가 왜 필요한지는 모르겠지만 어쨌든 정보는 있고, 정보가 있으니 분석해야 한다.

또한 전통적인 기업의 내부 IT 부서는 아웃소싱 여부의 심사 대상이 되는 시대가 됐다. 오라클 데이터베이스를 관리하고 유지하는 인력의 전문 기술에 대한 수요는 앞으로 급격히 줄어들 가능성이 높다. 대부분의 기업에서 DBA는 그저 숙련된 시스템 관리자일 뿐이다.

이런 모든 현상은 데이터를 평준화해서 하나의 구조로 밀어 넣지 않아도 되는, 이후 애플리케이션에서 다시 변형을 거쳐야만 데이터를 사용할 수 있는 데이터베이스가 필요하다는 것을 의미한다.

지금 필요한 데이터베이스는 사용자에게 즉각적인 만족감을 주는 데 필요한 만큼의 여러 디스크에 걸쳐 있는 방대한 데이터 스토리지를 다룰 수 있는 데이터베이스다. 지연은 용납되지 않는다.

관건은 표준

그러나 전환을 가로막는 장애물이 있다. 첫째, NoSQL에는 주도적인 세력이 없다. RDBMS의 경우 어떤 제품을 선택하든 최소한 ANSI(American National Standards Institute) 표준 SQL의 부분 집합이라도 있기 때문에 거기에 의존할 수 있다.

새로운 데이터베이스에는 피그(Pig), 하이브(Hive), 스파클(SPARK), 몽고(Mongo), 사이퍼(Cypher) 등이 사용된다. 이런 언어 간에는 공통점이 거의 없다. RDBMS의 경우 적어도 유서깊은 ODBC(Open DataBase Connectivity)에 몇 가지 커넥터 표준이라도 있다. NoSQL의 경우 데이터베이스별 커넥터에 의존해야 한다.

대부분의 사람들은 RDBMS가 모든 사람, 모든 용도에 맞는 만능 기술이 아니라는 사실을 알고 있었지만 표준이 있었기에 시장이 형성됐다. 시장은 장기적인 사회적, 개인적 단계적 투자를 유도한다.

제품, 개발업체, 그리고 벤처 투자자들의 근시안적인 관심이 나타났다가 사라지는 동안에도 표준은 존속한다. RDBMS 업계에는 무수히 많은 틈새 시장이 존재하지만 장기적으로 가장 큰 수익을 거둔 기술은 표준화된 영역에서 사용된 것이었다. 벤처 캐피탈의 자금을 받은 신생 업체보다 오래 존속한 기술을 중심으로 시장이 형성됐다.

NoSQL이 오라클을 물리치려면 무엇이 필요할까? 무엇보다 데이터베이스 드라이버를 위한 SPI(Service Programming Interface)와 주요 언어 및 플랫폼을 위한 API, 그리고 표준 쿼리 언어다.

특수한 경우를 제외하고는 많은 노력이 들어가도록 하면 안 된다. 달리 말하자면, 'ID별로 불러오기' 또는 '속성 또는 하위 속성별로 찾기'와 같은 일반적인 쿼리를 위한 필수적인 쿼리 언어 기능을 사용할 수 있어야 한다는 의미다.

기본적인 드라이버와 표준화된 API라면 CRUD(Create, Read, Update, Delete) 작업과 기초적인 검색 쿼리용으로 유용할 것이다. 선택적인 기능과 특정 API는 특정 데이터베이스 또는 데이터베이스 유형에 국한된 기능에 대해서

만 필요해야 한다(예: 그래프에서 두 레코드 간의 거리).

완벽할 필요는 없다. RDBMS 개발업체를 보더라도 업체 개별적인 비호환 확장 기능과 변형(결과적으로 고객을 묶어두는 역할을 함)없이 목적을 달성한 경우는 없다. 표준은 사람들이 부담없이 데이터베이스를 바꿀 수 있을 정도면 된다.

어떤 면에서 이것은 이른바 ‘뉴DB’의 다언어코드 지속성(polyglot persistence)보다 더 중요하다. 가령 몽고DB로 해결하기에 적합해 보이는 문서 데이터베이스 문제가 있는데, 새로운 요구 사항을 반영할 경우 Neo4j와 같은 그래프 데이터베이스로 해결하는 것이 훨씬 더 잘 들어맞는 경우가 있다.

쿼리의 공통 분모

문제는 이 다언어코드(polyglot) 특성에 있다. 그래프 데이터베이스용으로 설계된 쿼리 언어는 문서 데이터베이스 또는 키-값 쌍 구조(a key-value pair structure)에 적합하지 않을 수 있다. 대다수의 쿼리는 간단한 항목에 대한 것이므로 대부분의 경우 이것은 별 문제가 되지 않는다.


이런 데이터베이스의 대부분은 일종의 계층형 쿼리를 지원한다(SQL은 지원하지 않음). 예를 들면 빨간 머리 아이의 모든 부모, 조부모 또는 증조부모를 원하는 경우, 또는 특정 트랜지스터 모델을 포함한 제품을 주문한 모든 고객을 원하는 경우 등이다. 당연히 ‘뉴DB’ 표준 쿼리 언어는 이것을 지원해야 한다. 표준의 조건은 ‘반드시’라기보다는 ‘가능하면’이다.

2011년 희망의 빛이 보였다. 마이크로소프트 연구원 두 명이 NoSQL에 표준이 필요하다고 언급했다. 이는 플랫폼 지원의 점대점 속성과 기본적인 통합 쿼리 언어를 만들고자 하는 시도를 동시에 아우르는, 몽고DB를 위한 UnQL과 LINQ 지원을 구현하기 위한 움직임에 새로운 활기를 불어넣었다.

스프링 데이터(Spring Data)는 CRUD 인터페이스를 중심으로 자바 환경을 통합 중이지만 상황은 여전히 진흙탕 속이다. UnQL은 진흙 속에 다리가 모두 빠져서 보이지 않을 지경이다. Neo4j와 LINQ는 어떠한가?

이들에게 지금 필요한 것은 NoSQL 개발업체, 이해 당사자(스프링소스, 레드햇, 마이크로소프트, IBM), 그리고 다양한 프로젝트가 힘을 합쳐 분산된 작업을 끌어모아 표준을 제안하는 것이다. 먼저 쿼리 레벨을 정하고, 그 다음 커넥터 표준을 정의해야 한다.

표준화의 시장 형성 효과와는 별개로, 이런 형태의 성숙한 시도는 구매 담당자들에게 또 다른 헛된 ‘OODBMS(Object-Oriented DataBase Management System)’ 환상에 투자하지 않았다는 안도감을 준다. 땅 속에 돈이 묻혀 있고, 그 돈을 캐낼 장비가 바로 표준화다. 이 과정의 시작점은 기술 소비자의 최선의 이익만이 아니라 NoSQL 개발업체와 프로젝트 자체의 최선의 이익이기도 하다.

아파치 하둡, 10gen, 네오 테크놀러지, 클라우드베이스, 모두 경쟁은 하되, 모든 배를 움직일 물결도 일으켜야 한다. 물론 오라클의 엘리슨은 여기 동참하지 않을 것이다. 

빅데이터와 NoSQL에 대한 숨겨진 진실

Andrew Oliver | InfoWorld



빅 데이터 고객의 특징을 물으면 대부분 막대한 양의 데이터를 가진 고객이라고 말할 것이다. NoSQL의 고객 특징을 물으면 대부분 높은 수준의 동시 실행(concurrency)이 필요한 고객이라고 말할 것이다.

이것이 NoSQL과 빅데이터 시장의 전부라면 몽고DB를 포함한 하둡을 지원하는 여러 업체들은 지금 문을 닫고 사업을 접어야 할 것이다.

NoSQL과 빅데이터 붐의 근본 이유, 경제적 비용 절감

사실 하둡 도입은 경제적인 측면에서 결정된 경향이 있다. 충분한 자금과 엄청난 양의 데이터, 두 가지 모두 가진 기업이라면 IBM, SAP 또는 테라데이터의 최고급 MPP(Massively Parallel Processing) 솔루션에 돈을 투자할 가능성이 높다.

대부분의 대기업들은 이미 그렇게 해왔다. 그러나 우리 모두가 100달러짜리 지폐로 담배불을 붙이는 상위 1% 무리와 어울려 지내진 않는다. 설령 그런 사람이라 해도 데이터를 보관하고 이 데이터로 무엇을 할지는 나중에 결정하는 막대한 비용에 대한 결정을 ‘먼저’ 내려야 한다.

이 외에 우리 같은 보통 사람에게 하둡은 이전에는 이용할 수 없었던 분석 기능을 제공한다. 상업적으로 지원되는 ‘엔터프라이즈’ 하둡 배포판의 비용이라 해 봤자 예를 들어 IBM 네티자(Netezza)와 비교하면 극히 미미한 수준이다.

또한 몽고DB 또는 Neo4j와 같은 NoSQL 기술을 도입하는 것은 사실상 경제적인 의사 결정이다. 막강한 고성능 서버를 구입하고 개발자 인건비에 충분히 투자한다면 원하는 RDBMS에서 거의 모든 문서 또는 그래프 데이터베이스 작업을 실행할 수 있다.

그러나 개발자 인건비는 결코 싸지 않고 서버 라이선스 비용도 높다. 게다가 고가용성과 재해복구를 지원하도록 RDBMS를 확장하기 위한 인프라스트럭처에도 큰 비용이 든다. 따라서 현명한 운영자라면 NoSQL 대안을 반기는 것이 당연하다. 즉, 보편적인 하드웨어를 사용해 비용을 절약하고 필요에 따라 서버를 추가로 가동하면 된다.

아주 작은 중소기업을 제외하면 데이터가 ‘작고’, ‘동시 실행’을 요구하지 않는 기업이란 없다. 하둡과 몽고DB가 이미 막대한 양의 데이터를 보관하고 있고 수

백만 명의 사용자를 가진 기업들만 상대했다면, 시장은 몽고DB 하나의 가치보다도 훨씬 더 작은 규모가 되었을 것이다.

그렇다. 빅데이터와 NoSQL에 숨겨진 비밀은 관련 개발업체들이 페이스북, 구글과 같이 소비자를 상대로 하는 B2C 대기업만을 고객으로 하지 않는다는 점이다. 이 기술은 훨씬 더 광범위하게 적용된다.

높은 동시 실행, 낮은 비용, 유연한 데이터 스토리지의 공급이 증가하면 이에 따라 수요도 증가하게 된다. 그 모든 데이터를 저렴하게 모아둘 수 있다면 마이닝도 저렴하게 해서 큰 기업들과 경쟁하면 되지 않겠는가?

게다가 조직이 성숙해짐에 따라 RDBMS 스키마 디자인(그리고 더 중요한 스키마 업데이트)에는 막대한 조정 작업이 필요하게 된다. 막대한 조정이 필요한 일이란 드물수록 좋다.

반면 NoSQL과 같이 유연한 기술은 변덕스러운 고객 수요와 확장되고 변화하는 시장에 적용하기 위한 방법을 모색하는, 치열하게 경쟁하는 기업들의 요구에도 잘 부합한다.

오해는 하지 마라. 엄청난 양의 데이터를 갖고 광범위한 확장성을 지닌 기업은 NoSQL 데이터베이스와 빅데이터 도구를 배포할 수 있고 실제로 그렇게 한다.

그러나 지금까지 IT 부서에서는 당면한 관련성이 없는 데이터는 버리도록 스스로를 교육시켜왔을 것이다. 빅데이터와 NoSQL 기술은 모두에게 유용하며 데이터에 대한 사고 방식을 바꾸어 놓는다. 즉, 일단 보관하고 분석은 나중에 한다는 것이다. 이 시장은 생각보다 훨씬 더 큰 시장이다. **ITWORLD**



테크놀로지 및 비즈니스 의사 결정을 위한 최적의 미디어 파트너



기업 IT 책임자를 위한 글로벌 IT 트렌드와 깊이 있는 정보

ITWorld의 주 독자층인 기업 IT 책임자들이 원하는 정보는 보다 효과적으로 IT 환경을 구축하고 IT 서비스를 제공하여 기업의 비즈니스 경쟁력을 높일 수 있는 실질적인 정보입니다.

ITWorld는 단편적인 뉴스를 전달하는 데 그치지 않고 업계 전문가들의 분석과 실제 사용자들의 평가를 기반으로 한 깊이 있는 정보를 전달하는 데 주력하고 있습니다. 이를 위해 다양한 설문조사와 사례 분석을 진행하고 있으며, 실무에 활용할 수 있고 자료로서의 가치가 있는 내용과 형식을 지향하고 있습니다.

특히 IDG의 글로벌 네트워크를 통해 확보된 방대한 정보와 전세계 IT 리더들의 경험 및 의견을 통해 글로벌 IT의 표준 패러다임을 제시하고자 합니다.

어떤 데이터베이스를 사용해야 할까?

Andrew Oliver | InfoWorld

필자는 한때 시카고에 머무르면서 자사의 첫 위성 사무소를 설립하는 업무를 본 적이 있었다. 빅데이터 개발업체들의 본거지는 실리콘 벨리지만, 빅데이터 사용자와 관련 전문가들은 시카고에 많이 위치해있다. 정말 많은 사람들이 있었고 당시 시카고에는 빅데이터 관련 행사와 미팅이 매일 개최될 정도였다. 이런 빅데이터 행사에는 NoSQL에 대한 소개와 RDBMS가 더 이상 ‘만병 통치약’이 되지 않는 이유에 대한 설명이 빠지지 않았다.

이렇게 고객 가운데 상당수가 친숙하지 않은 지역에 위치해있다. 몇 가지 종류의 NoSQL 데이터베이스가 있다. 또 상황에 따라, 데이터세트에 따라 데이터베이스를 선택해야 할 이유가 있다. IT 산업의 마케팅 종사자들은 ‘NoSQL은 확장’이라는 명제로 홍보해왔다. 그러나 이보다는 훨씬 복잡한 원칙이 적용된다.

브루어 이론, 일관성-가용성-파티션 수용성 세 가지 다 되는 건 없다

NoSQL 데이터베이스 종류가 다양하게 된 부분적인 이유는 이른바, 브루어 이론(Brewer's Theorem)라 불리는 CAP 이론과 관련이 있다.

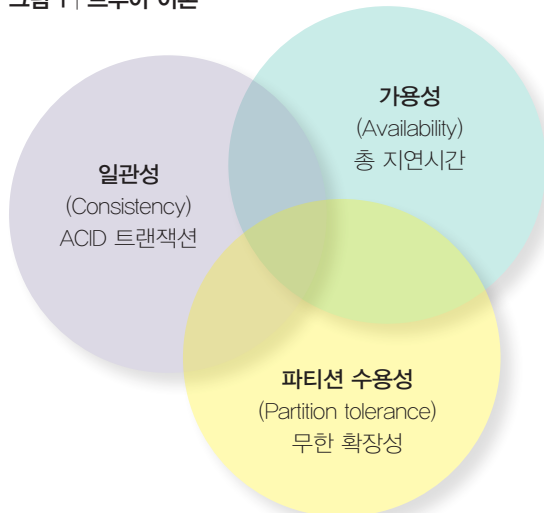
CAP 이론에 따르면, 일관성(C: Consistency), 가용성(A: Availability), 파티션 수용성(P: Partition tolerance)이라는 세 가지 요소 가운데 두 가지만 구현할 수 있다. 예를 들어 만약 일관성과 가용성에 초점을 맞췄다면 파티션 수용성은 잃게 된다. 데이터세트와 런타임 규칙이 달라 균형점도 다르며, 각 데이터 기술 또한 초점을 맞춘 균형점이 제각기 다르다. 여기에 더해 데이터의 복잡성과 시스템의 확장성을 감안해야 한다.

또한, NoSQL 기술이 다양한 이유는 기초 컴퓨터 공학이나 심지어 기초 수학에서도 근거를 발견할 수 있다.

일부 데이터세트는 키-값 쌍(Key-value pair)으로 쉽게 매핑할 수 있다. 데이터를 병합(Flatten)한다고 해서 의미가 퇴색되지 않으며, 관계를 재구축할 필요가 없다. 반면 다른 데이터 항목과의 관계가 데이터 항목 그 자체만큼이나 중요한 데이터세트도 있다.

RDBMS는 어느 정도는 집합론(set theory)의 부산물인 관계 대수(Relational Algebra)에 바탕을 두고 있다. 집합론에 기반을 둔 관계(Relationships)는 많은 데이터세트에 효과적이다. 그러나 페어런트-차일드 관계

그림 1 | 브루어 이론



(parent-child relationship)나 관계에 거리가 필요한 경우, 집합론은 아주 효과적이지 못하다. 효율적으로 데이터 솔루션을 설계하기 위해서는 그래프 이론(graph theory)이 필요할 수 있다.

달리 설명하면, RDBMS는 키-값 쌍(Key value pair)을 효과적으로 사용할 수 있는 데이터에는 ‘넘치고’, 더 많은 맥락(Context)이 필요한 데이터에는 ‘부족하다’. 넘치면 확장성에서, 부족하면 성능에서 손해를 본다.

키-값 쌍 데이터베이스(Key value pair databases)

키-값 쌍 데이터베이스로는 아파치 카산드라(Apache Cassandra)와 카우치베이스(Couchbase) 1.8 에디션 등이 있다.

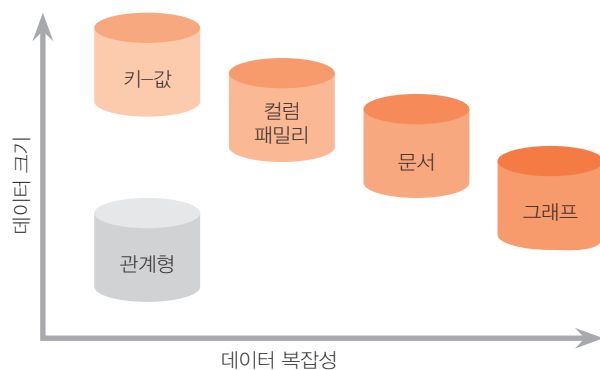
확장성이 높지만, 개발자에게 복잡한 데이터셋을 지원하지 않는다. 디스크 기반의 분산된 해시 테이블이 필요하고, ID(identity)를 기준으로 모든 부분을 조사할 수 있다면, 쉽게 확장이 되고, 빠르게 경량화(lightning)가 된다. 그러나

여러 다른 키를 조사해야 값을 얻을 수 있다면, 더 복잡한 형태를 갖고 있을 것이다.

키-값 쌍 데이터베이스에는 여러 다양한 순열이 있다. CAP 이론에서는 기본적으로 다른 트레이드 오프(trade off)이며, 스토리 및 메모리 사용에 관한 구성도 다르다. 궁극적으로, 기본적으로는 해시(hash) 테이블인 형태를 갖게 된다.

혼합이 되지 않는다면 플랫(flat) 파트 리스트에는 좋다. 또 키가 의미를 갖고 있고, 값을 조사하는 주된 경로인 경우에 해당하는 다른 형태의 리스트나, 당장의 주식 시세에도 좋다. 이는 일반적으로 인덱스와 결합이 된다. 그리고 값에 대한 쿼리와 키 리스트를 생성할 방법이 있다. 그러나 이런 부분들이 많이 필요하다면, 다른 기술을 찾아야 할지 모른다.

그림 2 | 키-값 쌍 데이터베이스



컬럼 패밀리/빅 테이블 데이터베이스

카산드라 등 대다수의 키-값 저장소는 컬럼을 그룹화 한 형태를 제공한다. 이를 ‘컬럼 패밀리(column family)’ 또는 ‘빅 테이블(big table)’로 간주할 수 있다. HBase 등 처음부터 컬럼 패밀리 저장소로 설계된 데이터베이스도 있다. 이는 더욱 발전된 형태의 키-값 쌍 데이터베이스다. 본질적으로 키와 값은 혼합이 된다. 다차원으로 교차 배열된 해시 맵을 생각하면 된다. 본질적으로 각 컬럼에는 하나의 데이터 열이 포함된다.

카산드라 공인 판매 업체인 데이터스택스(DataStax) 제품 부사장 로빈 슈마허에 따르면, 카산드라는 기기, 센서, 웹사이트(웹로그 등), 파이낸셜 틱 데이터(financial tick data) 등의 시계열 데이터에 가장 많이 사용한다. 일반적으로 속도가 빠르고, 한 번에 여러 장소에서 전송되며, 신속하게 추가되고, 빠른 쓰기



많은 개발자가
문서 데이터베이스를
'성배'와 같이 신성시
한다. 객체 지향형
프로그래밍에 잘 맞기
때문이다.

능력과 고성능의 읽기 성능이 필요한 데이터들이다.

여기에 맵리듀스(MapReduce)를 사용할 수도 있다. 이 경우, 반구조화(semi-structured) 데이터를 대상으로 하는 우수한 분석 저장소가 될 수 있다. 맵리듀스는 높은 확장성을 지니고 있지만, 일반적으로 트랜잭션에는 그닥 좋지 않다. 거리 또는 경로 계산 등과 같은 데이터간 관계가 데이터 자체만큼 중요하다면, 컬럼 패밀리/빅 테이블 데이터베이스를 사용할 수 없다.

문서 데이터베이스(Document databases)

많은 개발자들이 문서 데이터베이스(Document databases)를 '성배'와 같이 신성시 한다. 객체 지향형 프로그래밍에 잘 맞기 때문이다. 10젠(10gen)의 몽고DB, 카우치베이스(Couchbase), 아파치 카우치DB(CouchDB) 등 유명 개발업체들이 가장 초점을 맞추고 있는 분야다.

카우치베이스의 프랭크 위겔은 기업들이 버전 1.8인 키-값 쌍 데이터베이스에서 2.0인 문서 데이터베이스로 이동하고 있다고 말했다. 위겔은 "문서 데이터베이스는 자연스러운 발전이다. 클러스터링에서 데이터 액세스까지 문서 데이터베이스와 키-값 저장소는 정확히 같다. 단 문서 데이터베이스의 경우 데이터 저장소의 문서를 파악한다는 차이가 있다"고 설명했다. 다른 말로 설명하면, 값이 JSON이다. 따라서 쿼리와 검색 능력을 높이기 위해 JSON 문서 내부 요소를 인덱스로 처리할 수 있다.

여기에서의 장점은 이미 JSON 문서를 생성하고 있을 수 있다는 것이다. 10젠의 대표 맥스 쉬어슨에 의하면, RDBMS에서 모델화 하기에는 너무 복잡한 데이터에는 문서 데이터베이스를 고려해야 한다. 예를 들어, 복잡한 파생 유가증권(derivative security) 데이터는 전통적인 형식으로 저장하기란 아주 어렵다. 전자 의료 기록도 또 다른 좋은 사례다. XML 저장을 활용할 계획을 갖고 있다면, 몽고DB와 JSON/BSON을 고려해야 한다는 신호다.

사용자, 시스템, 소셜 네트워크, 기타 출처에서 수집하는 데이터를 저장하는 운영 저장소가 될 수도 있다. 몽고DB와 같은 데이터베이스도 맵리듀스 기능을 갖고 있는 경우가 많지만, 리포팅용 데이터베이스로는 적합하지 않다. 몽고DB에서도 무엇이든 쿼리를 할 수 있다. 그러나 인덱스가 없으면 일반적으로 수용할만한 결과를 얻지 못한다.

그래프 데이터베이스(Graph databases)

그래프 데이터베이스(Graph databases)는 데이터 볼륨이나 가용성보다는 데이터 관계 방식과 수행하고자 하는 계산과 더 큰 관련이 있다. Neo4j 개발업체인 네오 테크놀로지스(Neo Technologies)의 제품 엔지니어링 선임 이사인 필립 라틀은 데이터셋이 기본적으로 테이블이 아닌 데이터와 상호 연결이 되어 있을 때 그래프 데이터베이스가 아주 유용하다고 설명했다. 기본적인 데이터 액세스 패턴이 트랜잭션이다. 즉 배치가 아닌 레코드의 OLTP/시스템이다. 그



*RDBMS로 모든
문제를 해결할 수 없듯,
집합론으로 모든 수학
문제를 해결할 수 없다.*

래프 데이터베이스는 트랜잭션 방식으로 연관성을 실행시킬 수 있지만 RDBMS에서는 배치 방식으로 처리해야 한다.

그래프 데이터베이스의 이런 방식은 대다수 NoSQL 마케팅에 도전하는 개념이다. 그래프 데이터베이스가 필요한 곳은 RDBMS가 구현하는 것과 비교해 데이터 구조에 더 정확한 트랜잭션이 요구되는 상황이다.

그래프 데이터베이스는 지리공간(Geospatial) 문제들, 추천 엔진, 네트워크/클라우드 분석, 생물정보학에 많이 사용된다. 기본적으로 데이터 간 관계가 데이터 자체만큼이나 중요한 경우다. 이는 다양한 금융 분석 기능에도 중요한 역할을 하는 기술이다. 예를 들어, 다른 회사의 '나쁜 소식'이 한 회사에 어느 정도의 악영향을 초래하는지 분석하고 싶다면, 관계의 방향이 아주 중요한 계산 요소가 될 수 있다. 여러 SQL 스테이트먼트에서 이를 쿼리하기 위해서는 많은 코드가 필요하고, 속도도 빠르지 않다. 그러나 그래프 데이터베이스를 이를 능가한다.

데이터가 단순하거나 테이블식이라면 그래프 데이터베이스가 필요없다. 또 OLAP나 Length 분석에는 그래프 데이터베이스가 적합하지 않다. 일반적으로 그래프 데이터베이스는 인덱스를 이용해 검색과 조사 능력을 높여준다. 그러나 그래프 부분을 트래버스(Traverse) 해야 하며, 이를 위해서는 초기 노드 일부를 수정할 필요가 있다.

NoSQL, 명칭에 관한 문제

그래프 데이터베이스는 이들 새로운 형태의 데이터베이스에 이름을 붙이기 힘든 이유를 보여준다. 필자가 선호하는 이름은 '뉴DB(NewDB)'다. 다만 일부 데이터베이스는 RDBMS만큼 오랜 역사를 가지고 있다는 문제가 있다. NoSQL은 좋은 이름이 아니다. 일부가 SQL을 지원하지만, SQL은 이들 시스템의 기능에 딱 들어맞는 것은 아니기 때문이다.

'빅데이터(Bigdata)'도 정확하지 않다. 데이터가 관계형 데이터베이스보다 더 자연스럽게 부합하는 데이터베이스를 심분 활용할 수 있다면 대용량의 데이터 세트는 필요하지 않기 때문이다. 비관계형(Nonrelational)도 어울리지 않는다. 그래프 데이터베이스는 관계형 데이터베이스에 가깝기 때문이다. 전통적인 RDBMS와는 다른 형태의 관계를 추적할 뿐이다.

사실 이들 데이터베이스는 나머지 문제들을 해결할 수 있는 나머지 데이터베이스다. 지난 수십 년 동안의 마케팅 오류와 함께 하드웨어 및 대역의 제한, 대기시간과 볼륨에 있어 상대적으로 낮은 기대치 등이 이 오래된 데이터베이스들이 RDBMS만큼 성장하지 못하도록 방해해왔다.

RDBMS로 모든 문제를 해결할 수 없듯, 집합론으로 모든 수학 문제를 해결할 수 없다. 현재 데이터 문제가 점차 복잡해져 가고 있다. 확장성과 성능(낮은 대기시간), 볼륨에 대한 요구가 커지고 있다. 이런 문제를 해결하기 위해서는 하나 이상의 데이터베이스 기술을 필수적이다. 자, 어떤 데이터베이스를 선택할 것인가? 