강의 **06**/16:
OSS Database management system

신정규

2017년 4월 6일

Open-source DBMS

DBMS

- 데이터베이스
  - 데이터들의 집합
- Database management system
  - 데이터베이스 안의 데이터를 이용하기 위한 소프트웨어
  - 데이터베이스 접근 + 운영 + 관리 등을 포괄
- 장점
  - 데이터 접근의 일관적 방법 보장
  - 데이터 접근의 권한 설정 가능
  - 데이터의 관계성 정의 및 보장 가능

DBMS

- SQL
  - Standard Query Language
  - 데이터베이스에서 데이터를 읽기 위한 언어
  - 여러분이 배워야 할 것

```
SELECT name from Attendance
WHERE count(present) > 3
```


RDBMS

- RDBMS
  - 관계형 모델이 정의되는 데이터베이스
- 관계형 모델
  - 행 (row) /열 (column) 로 결정되는 하나 이상의 테이블 형태로 데이터를 저장
  - Record – row 에 해당 / attribute – column 에 해당
  - 테이블 – record의 집합
  - 엑셀을 상상해 봅시다.
- 거의 모든 고전적 DBMS의 구조

RDBMS

- Transaction
  - DBMS에 특정 작업 (CRUD) 을 할 때의 단위
  - CRUD: Create, Read, Update, Delete
- Constraint
  - 특정 필드나 속성의 값에 거는 제약 조건
- Index
  - 데이터를 빠르게 검색하고 찾기 위해 미리 계산해 놓는 필드/필드 묶음

요구사항 / 용어

- Transaction / Locking
  - 특정한 '작업' 을 어떻게 다루는가?
  - 하나의 작업이 다른 작업에 영향을 어떻게 덜 미치게 만드는가?
- Fail-over
  - 하드웨어 / 소프트웨어
- Replication
  - Fail-over 를 위한 대비
  - 동시 접속 수 / 성능 개선을 위한 용도

ACID

- DBMS transaction이 가져야 할 조건들
  - Atomicity
    - 모든 데이터베이스 오퍼레이션은 완료가 보장되어야 한다
  - Consistency
    - 데이터의 정합도가 보장되어야 한다. 일관성이 유지되어야 한다.
  - Isolation
    - 하나의 트랜잭션이 수행될 시 다른 작업이 해당 트랜잭션의 사이에 들어올 수 없어야 한다.
  - Durability
    - 트랜잭션이 끝나면 그 결과가 항상 유지되고 보존되어야 한다.

NoSQL

- SQL 인터페이스를 사용하지 않은 DBMS
  - 구조상의 제약이 적어 속도에 최적화 된 솔루션이 많음
  - ACID 를 포기하겠다!
  - Kvstore, Object DB, Document Store, Wide Columnar Store 등
- 분야
  - 데이터가 들어오는 속도가 처리하는 속도보다 빠른 분야: 빅데이터
  - Memcached, ZopeDB (Object), CouchDB, MongoDB (Document Store), Cassandra, BigTable (Wide Columnar Store)

Open-source DBMS

MariaDB

- MySQL의 후신
- 보편성
  - 엄청나게 많은 사용으로 인한 편의성
  - 실서비스에서 다수 사용 – 적정 규모에서의 안정성 보장
  - 다양한 스토리지 엔진 지원
- 문제점
  - 최적화 난이도가 높음
  - 설정에 따라 DBMS의 동작의 일관성이 떨어짐
  - String 처리시 언어의 영향을 (아직도) 받음
  - Memcached 인터페이스가 없음

PostgreSQL

- 대중성
  - 오랫동안 (특히 학계를 중심으로) 널리 쓰였음
  - 데이터 타입의 확장이 자유로움
- 확장성
  - View, procedure등을 사용자가 임의로 만들 수 있음
  - 다양한 검색 알고리즘을 선택가능
- 주기적인 최적화
  - vacuum을 해 줘야 함
- 다양한 주체에 의해 개발
  - 코드간 통일성 문제
  - 성능상의 불이익

MongoDB

- NoSQL의 대표격
- 성능
  - Validation 지원
  - In-memory 스토리지 지원
  - 스토리지 암호화 지원
  - SQL 일부 지원
  - 대용량 데이터 스트리밍 처리에 유리

- 복잡도가 높은 작업시
  - 복잡한 트랜잭션이 있는 서비스에서는 쓸 수 없음
  - 기존 RDBMS를 바로 대체할 수 없음

Firebird

- 역사와 전통
  - 을 자랑하는 RDBMS
  - 다양한 플랫폼에 포팅되어 있음
  - 표준 SQL 호환성
- 장점
  - (역사로 인한) 다양한 개발 도구 및 트레이서
  - 다양한 확장 기능 및 플러그인 지원
  - 프로시저 및 트리거 지원
- 단점
  - 지금와서 굳이...
  - 대규모 확장이 용이하지 않음

Cubrid

- 국산 데이터베이스 플랫폼
- 엔터프라이즈에 적합
  - GUI 도구 및 언어 바인딩 기본 지원
  - 장애시 자동 복구 기능 지원
  - `샤딩` 지원
  - 복제 및 트랜잭션의 consistency 보장
- 문제점
  - 실제 사용예가 많지 않음
  - 플랫폼에 따라 설치가 용이하지 않은 경우들이 있음

SQLite

- 역사
  - 2000년 이후 엄청난 발전
- 간편성
  - 단일 파일로 동작
  - ACID 지원이 됨
  - 속도가 엄청나게 빠름
  - 인증 과정이 없음
- 확장성의 부재
  - 단일 파일에서 오는 한계: 큰 데이터를 다루기 어려움
  - Lock의 단위가 커서 동시 접속수가 많은 경우를 핸들할 수 없음

Learn by run: 데이터 읽고 쓰기

- Backend + DBMS
  - DBMS는 편의상 SQLite만을 씁니다
  - 자신이 고른 backend를 통해 데이터를 쓰고 읽는 간단한 코드를 짜 봅시다.

  - 프론트엔드에서 백엔드로 데이터를 보내서 데이터를 기록해 봅시다.
  - 그 데이터가 제대로 저장되었는지 간단한 연습 코드를 만들어 데이터베이스를 읽어 봅시다.

Assignment #4

- 데이터가 저장되는 노트앱 만들기
  - 이제 실제로 저장을 해 봅시다.
  - 백엔드를 골라 데이터를 받아 저장하는 부분을 구현해 봅시다.
  - 메모가 저장되고, 저장된 메모들이 출력되는 페이지를 구현해 봅시다.

  - 제출할 것: 만든 코드가 있는 github 주소 (4월 12일 24시까지)

Next is...

7/16: Special lecture (tentative)

@inureyes
Questions? inureyes@gmail.com