일괄처리

시스템의 3가지 유형

- 서비스 (온라인 시스템)
  - 클라이언트로 부터 요청이나 지시가 올때까지 기다린다. 요청이 들어오면 요청을 처리해서 응답을 되돌려 보낸다.
- 일괄 처리 시스템 (오프라인 시스템)
  - 매우 큰 입력 데이터를 받아 처리하는 작업을 수행하고 결과 데이터를 생산한다. 처리하는 동안 사용자가 대기하지 않는다.
- 스트림 처리 시스템 (준실시간 시스템)
  - 온라인과 오프라인/일괄 처리 사이의 어딘가에 있는 시스템. 일괄처리는 정해진 데이터를 대상으로 작동이 되지만, 스트림 처리는 입력 이벤트가 발생한 직후 바로 동작을 한다.

유닉스 도구로 일괄 처리 하기

- 단순 로그 분석

```
cat /var/log/nginx/access.log |
 awk '{print $7}' |
 sort  |
 uniq –c  |
 sort –r –n  |
 head –n 5
```

awk : 공백으로 분리해서 7번 째 필드만 출력

sort : 알파벳 순으로 정렬

uniq : 인접한 두 줄의 중복제거 (-c 중복 횟수 함께 출력)

-n : 매 줄 첫번째 나타나는 수를 기준으로 다시 정렬

-r : 내림차순

head: 앞에서 부터 출력 (-n 5 앞에서 5번째)

유닉스의 철학

- 각 프로그램이 한 가지 일만 하도록 작성하라. 새 작업을 하려면 기존 프로그램을 고쳐 새로운 기능을 추가해 프로그램을 복잡하게 만들기보다는 새로운 프로그램을 작성하라.
- 모든 프로그램의 출력은 아직 알려지지 않은 다른 프로그램의 입력으로 쓰일 수 있다고 생각하라. ... 대화형 입력을 고집하지 마라.

- 소프트웨어를 빠르게 써볼 수 있게 설계하고 구축하라. 심지어 운영체제도 마찬가지다. 수 주 안에 끝내는 것이 이상적이다. 거슬리는 부분은 과감히 새로 구축하라.
- 프로그래밍 작업을 줄이려면 미숙한 도움보단 도구를 사용하라. 도구를 빌드하기 위해 한참 둘러가야 하고 게다가 사용 후 바로 버린다고 할지라도 도구를 써라.

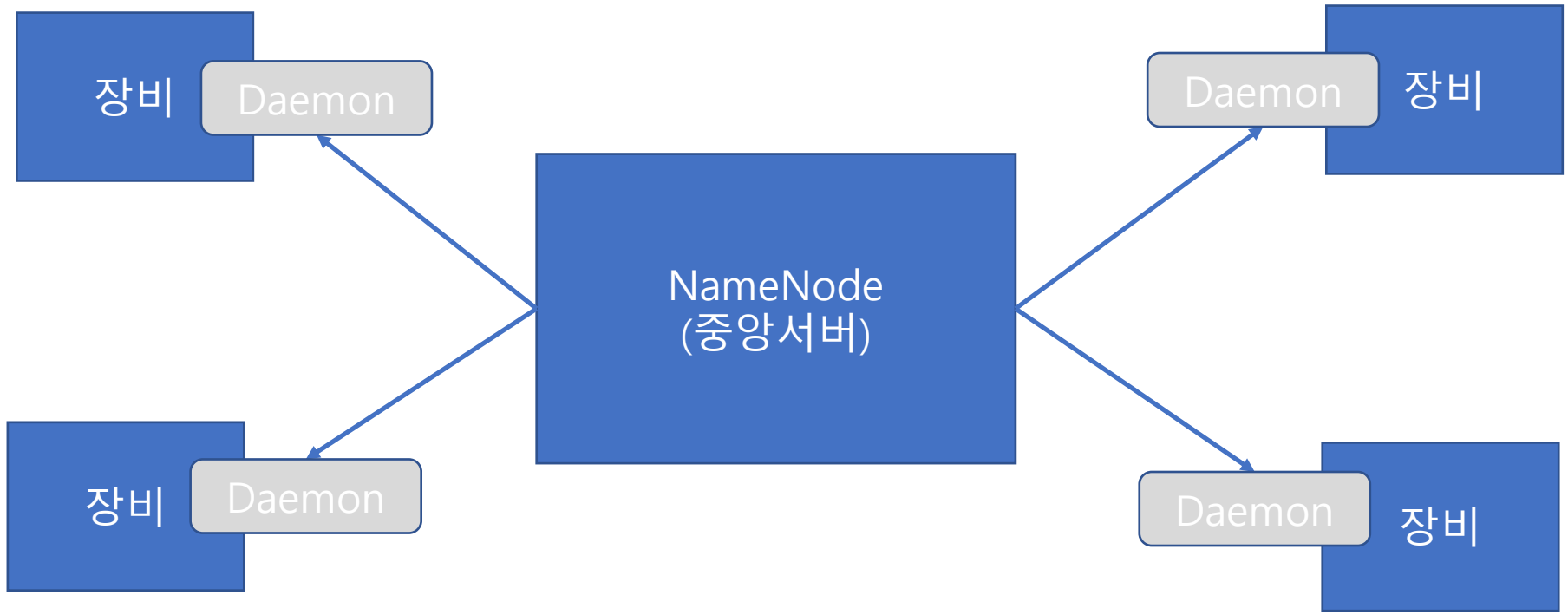
투명성과 실험

- 유닉스 명령에 들어가는 입력 파일은 일반적으로 불변으로 처리 된다. 이것은 다양한 명령행 옵션을 사용해가며 원하는 만큼 명령을 수행하더라도 입력 파일에는 손상을 주지 않는다는 것이다.
- 어느 시점이든 파이프라인을 중단하고 출력을 파이프를 통해 less 로 보내 원하는 형태로 출력이 나오는지 확일 할 수 있다.

- 특정 파이프라인 단계의 출력을 파일에 쓰고 그 파일을 다음 단계의 입력으로 사용할 수 있다. 이렇게 하면 전체 파이프라인을 다시 시작하지 않고 다음 단계부터 재시작할 수 있다.
- 단점) 가장 큰 제약은 단일 장비에서만 실행된다는 점이다. 이점이 하둡같은 도구가 필요한 이유이다.

맵리듀스와 분산 파일 시스템

- 유닉스 도구와 비슷하지만, 수천 대의 장비에서 실행
- 유닉스 도구가 stdin, stdout 을 사용하는데 반해, 맵리듀스는 HDFS(Hadoop Distributed File System) 라고 하는 분산파일시스템을 사용함
  - HDFS 는 비공유방식 기반으로 작동함. 특별한 하드웨이없이 일반적인 컴퓨터에서 실행되는 것을 비공유 방식이라고 함

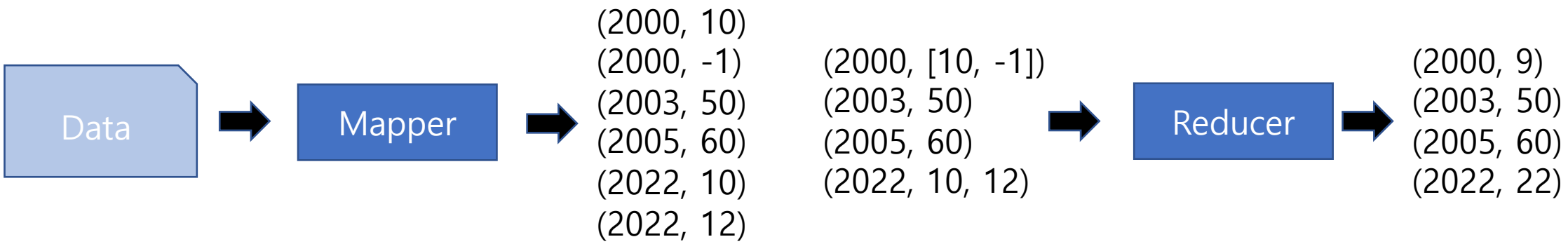


데몬 프로세스에 의해 각 장비의 파일 시스템에 접근한다.

=> 개념적으로 하나의 큰 파일 시스템이라고 간주함.

맵리듀스 작업 실행하기

- 맵리듀스란 HDFS 같은 분산 파일 시스템을 이용해 대용량 데이터셋을 처리하는 프로그래밍 프레임워크. 2004년 구글에서 발표함.
- 매퍼 (Mapper)
  - 입력레코드로부터 키와 값을 추출하는 작업. 모든 입력 레코드마다 한번씩만 실행된다. 입력레코드로 부터 다음 레코드까지 상태를 유지하지 않기 때문에 각 레코드는 독립적으로 처리된다.
- 리듀서 (Reducer)
  - 매퍼가 생성한 키-값 쌍을 받아 같은 키를 가진 레코드를 모으고, 해당 값의 집합을 반복해 리듀서 함수를 호출해 출력 데이터를 생산한다.


Data
Mapper
(2000, 10)
(2000, -1)
(2003, 50)
(2005, 60)
(2022, 10)
(2022, 12)
(2000, [10, -1])
(2003, 50)
(2005, 60)
(2022, 10, 12)
Reducer
(2000, 9)
(2003, 50)
(2005, 60)
(2022, 22)

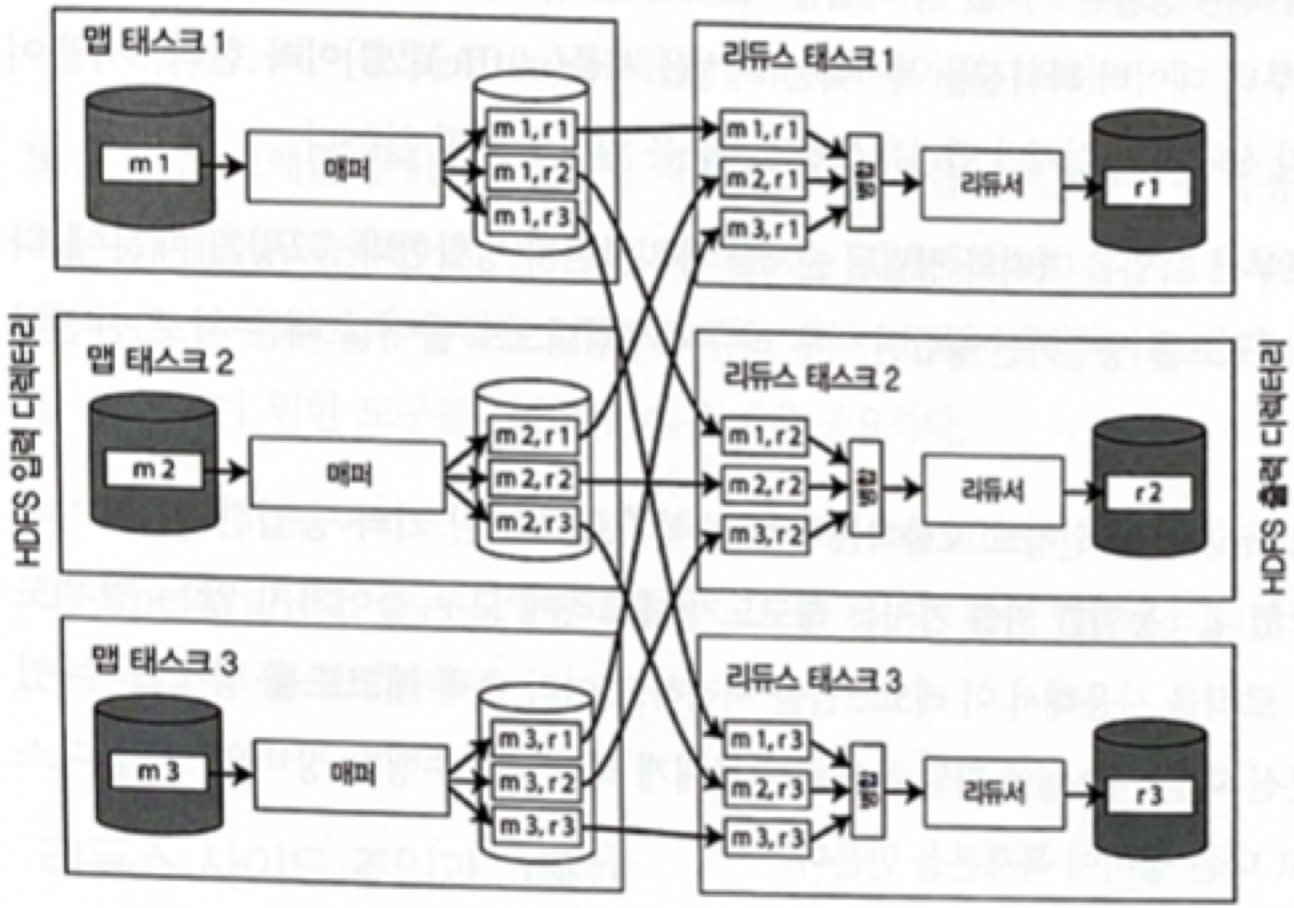


그림 10-1. 매퍼 3개와 리듀서 3개로 구성된 맵리듀스 작업

맵리듀스 워크플로

- 맵리듀스 하나로 해결할 수 있는 작업 범위는 제한적
- 여러개의 맵리듀스 작업을 연결해서 사용하는 것이 워크플로(workflow)이다.
- 하나의 출력을 다른 맵리듀스의 입력으로 이용하는 방법.
- 이런 워크플로우를 맵리듀스에서는 지원을 하지 않기 때문에 디렉토리 이름을 통해 암묵적으로 연결을 해서 사용함.
- 이 작업은 맵리듀스 관점에서는 완전히 독립적인 작업이 됨.

- 워크플로 작업은 일괄처리 작업이 성공적으로 끝나야만 유효한 작업임.
- 선행작업이 완전히 정상적으로 끝나야지 다음작업이 진행이 되는데, 이런 작업간의 의존성을 관리하기 위한 다양한 스케쥴러가 존재함.
  - 우지(Oozie), 아즈카반(Azkaban), 루이지(Luigi), 에어플로(Airflow), 핀볼(Pinball) 등...

정렬 병합 조인

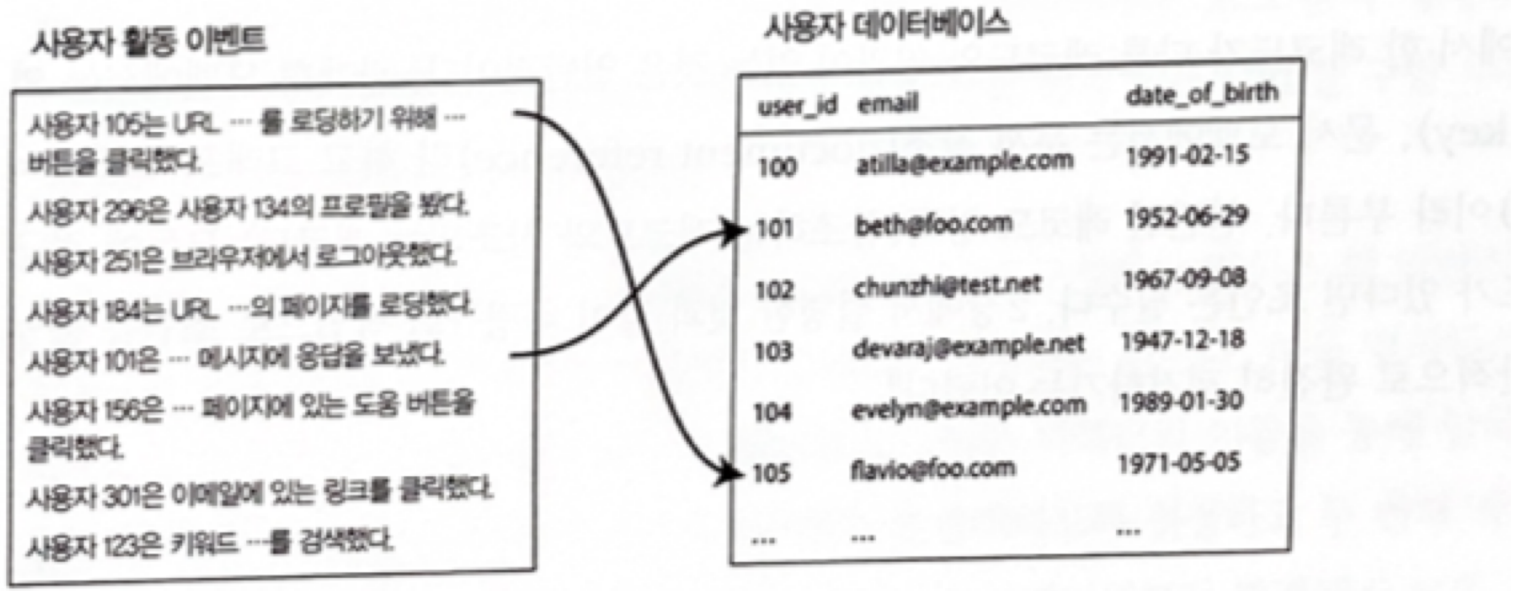
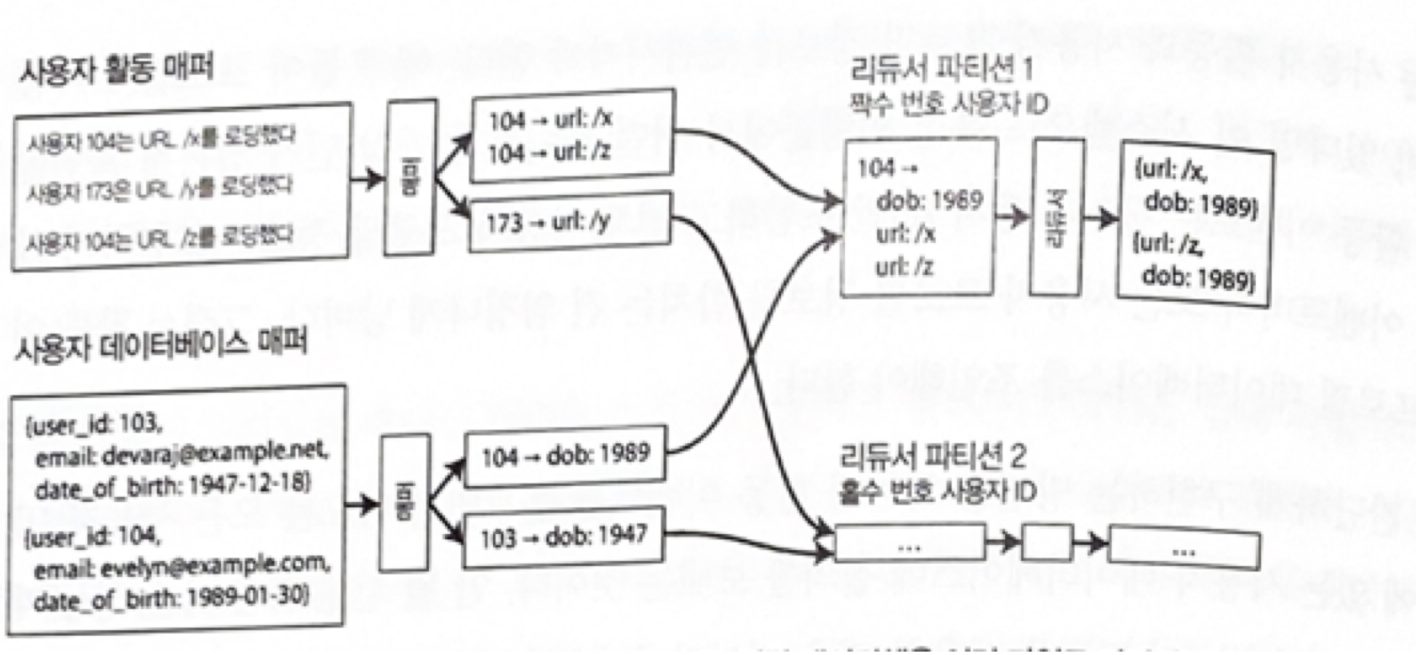


그림 10-2. 사용자 활동 이벤트 로그와 사용자 프로필 데이터베이스 간 조인



보조 정렬 (secondary sort) : 키 값으로 그룹핑하고 그룹핑된 레코드에 순서를 부여해서 정렬 하는 것. 리듀서가 항상 사용자 데이터 베이스를 먼저 보고 활동 이벤트를 시간 순으로 작업 레코드를 재배열함.

- 리듀서는 모든 사용자 ID 당 한번만 호출되고, 보조 정렬로 인해 첫 번째 값은 사용자 데이터베이스의 생년월일 레코드라고 예상
- 지역 변수에 생년월일을 저장하고, 그 다음부터 같은 사용자 ID 가 들어올 경우 활동 이벤트를 URL 과 사용자 연령의 쌍을 출력한다.
- 이 과정을 정렬병합 조인이라고 함. 매퍼 출력이 키로 정렬된 후에 리듀서가 양측의 정렬된 레코드 목록을 병합.

그룹화

- 같은 곳으로 관련 데이터를 모으는 것과 같은 작업을 수행할 수 있음

  (ex) group by

- 그룹화 연산의 예시
  - 각 그룹의 레코드 수를 카운트
  - 특정 필드 내의 모든 값을 더하기
  - 상위 k 개의 레코드 고르기
- 맵리듀스 상에서는 그룹화할 대상을 키로 함으로써 구현할 수 있음

쏠림다루기

- 키 하나에 너무 많은 데이터가 존재해서 쏠림 현상이 발생할 수 있음
- 핫스팟
  - 리듀서 하나에 너무 많은 데이터가 존재해, 리듀서가 다른 리듀서 보다 더 많은 데이터를 처리해야 하는 상황
  - 쏠린 조인(skewed join) 메서드 : 샘플링 작업을 통해 핫키(불균형한 활성 데이터 베이스 레코드) 를 찾아낸 다음 여러 리듀서에 퍼뜨려 부하 분산

맵사이드 조인

- 이전까지의 조인은 리듀스 사이드 조인. 매퍼는 키와 값을 추출하는 역할만 담당. 입력데이터 준비하는 역할.
  - 장점: 입력데이터에 대한 특정 가정이 필요 없음. 입력데이터의 속성과 구조가 무엇이든 매퍼는 데이터를 조인할 준비가 되어 있음
  - 단점: 정렬 후 리듀서로 복사한 뒤 리듀서 입력을 병합하는 과정의 비용이 비쌈.
- 입력데이터의 특정이 가능하다면 맵 사이드 조인 (map-side join) 기법을 사용할 수 있음
  - 브로드캐스트 해시 조인
  - 파티션 해시 조인
  - 맵 사이드 병합 조인
  - 워크플로

일괄처리 워크플로의 출력

- 일괄처리의 출력은 일반적인 보고서 종류의 출력이 아님
  - 일반적인 보고서의 역할
    - SQL 질의는 사용자에게 보여줄 소량의 레코드만 조회해서 출력함.
    - 이런 데이터는 분석가나 사업상 결정을 내리는데 이용할 수가 있음

검색 색인 구축

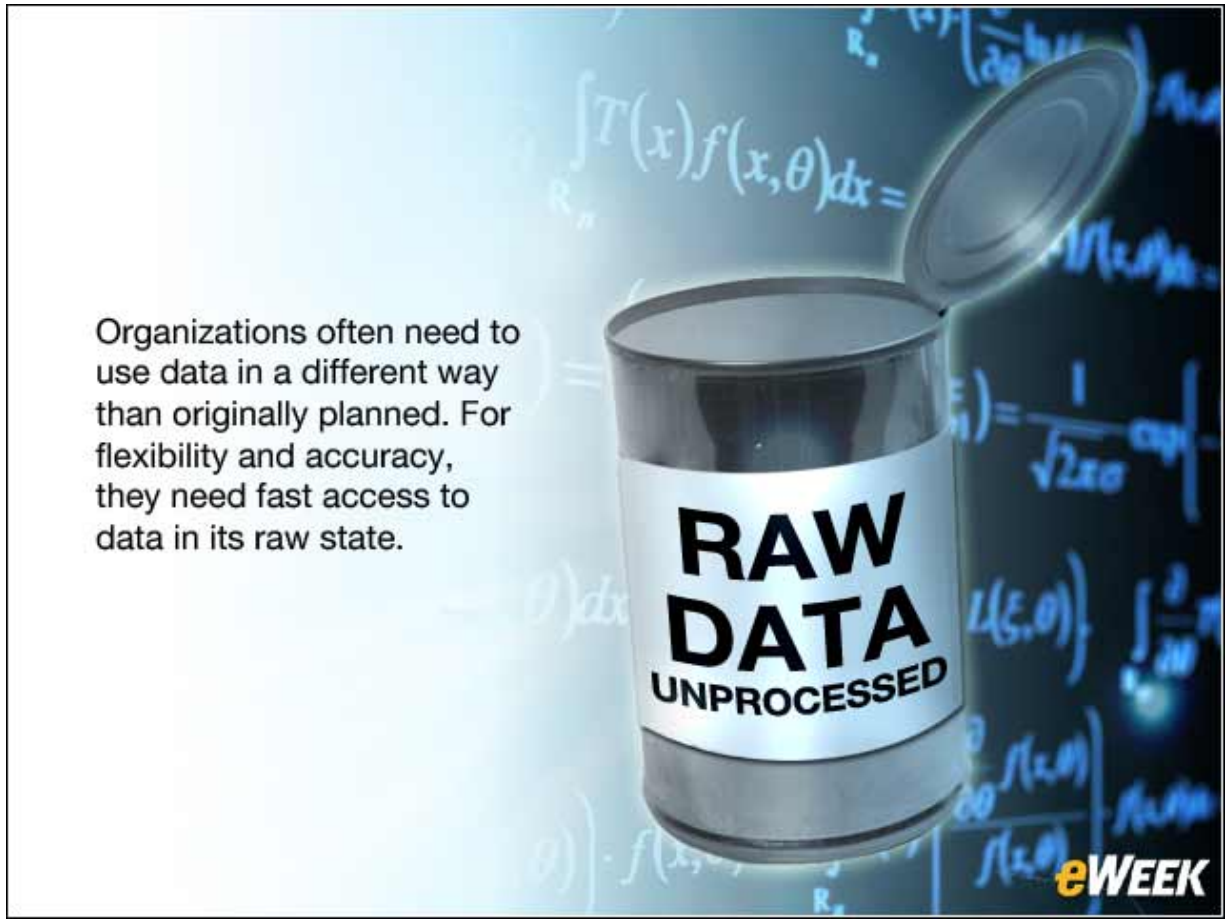
- 검색 색인 구축
  - 맵리듀스는 구글에서 검색엔진에 사용할 색인을 구축하기 위해 사용됨
  - 매퍼는 필요에 따라 문서 집합을 파티셔닝 함
  - 리듀서가 해당 파티션에 대한 색인을 구축
  - 색인된 문서 집합이 변하면 전체 문서 대상으로 주기적으로 색인
- 키-값을 저장
  - 맵리듀스를 활용해 데이터베이스 파일을 생성

하둡과 분산 데이터베이스의 비교

- 저장소의 다양성
  - 대규모 병렬 처리 (Massively parallel processing, MPP) 데이터베이스는 특정 모델을 따라 데이터를 구조화 해야함
  - 반면 분산파일시스템의 파일은 어떤 데이터 모델과 인코딩을 사용해도 기록할 수가 있음. 어떤 데이터 형태라도 HDFS 로 덤프를 할 수가 있음.
  - 커다란 조직의 다양한 부분에서 나온 데이터를 한곳에 모으는 작업만으로 큰 가치가 있다. 원시 데이터를 수집하고 스키마 설계는 나중에 고민하면 데이터 수집의 속도가 올라간다 – 데이터 호수 (data lake) 또는 엔터프라이즈 데이터 허브 (enterprise data hub)

초밥원리 (sushi principle)

- 생산자와 소비자가 서로 다른 우선순위를 가진 다른 팀에 있다면 이것은 이점이 있다. 하나의 이상적인 데이터 모델은 존재하지 않을지 몰라도 데이터에는 여러 목적에 적합한 다양한 관점이 존재한다. 원시 상태로 데이터를 덤프하는 것만으로도 이런 여러 변환이 가능하다. 이런 접근법을 초밥원리라 부른다. 원시 데이터가 좋다.



조직은 종종 원래 계획과 다른 방식의 데이터를 사용해야 한다.

유연성과 정확성을 위해 원시 상태의 데이터에 빠르게 액세스 해야 한다.

처리모델의 다양성

- MPP 데이터베이스는 디스크 저장소, 레이아웃, 질의 계획, 스케쥴링 같은 것들이 일체식으로 고정되어 있음. 이런 요소들은 데이터베이스의 특정한 필요에 따라 튜닝되거나 움직이기 때문에 매우 좋은 성능을 발휘함.
- 반면 머신러닝, 추천 시스템, 이미지 분석 시스템 같은 범용적인 데이터 모델을 처리하지는 못함. 맵리듀스를 이용하면 자신이 작성한 코드를 대용량 데이터셋 상에서 쉽게 실행을 할 수가 있음. – 하이브 프로젝트 (HDFS 와 맵리듀스 위에 SQL 질의 실행엔진을 구축)

결함을 줄이는 설계

	MPP	맵리듀스
결함을 다루는 방식	장비 한대에 문제가 생기면 전체 질의가 중단	개별 수준에서 작업을 진행하기 때문에 전체 작업에 영향을 받지 않음
데이터 기록	메모리	디스크
민감도	사용자에게 바로 결과를 줘야 하기 때문에 민감함	실패시 다시 처리하기 때문에 덜 민감함

- 일괄처리 작업은 "식탁 아래에서 조각들을 모은다." 우선 순위가 높은 프로세스가 필요한 작업을 수행한 뒤에 남은 자원을 사용해 연산을 수행함. 태스크 선점의 이슈가 있기 때문에 맵리듀스는 이런 예상치 못한 종료에 견디게 설계가 되었음.