제 1 절. 성능 데이터 모델링의 개요

1. 성능 데이터 모델링의 정의

- DB 성능향상을 목적으로 설계단계의 데이터 모델링때부터 성능과 관련된 사항이 데이터 모델링에 반영될수 있도록 하는 것.
- 정규화, 반정규화, 인덱싱, PK FK ... 등을 최적의 성능이 나올 수 있게 설계하는 것

2. 성능 데이터 모델링 수행시점

- 일찍할 수록 좋다.

3. 성능 데이터 모델링 고려사항

- 아래 순서대로 수행

    1. 정확한 정규화
    2. DB 용량산정
    3. DB 트랜잭션 유형 파악
    4. 반정규화 고려
    5. 이력모델, PK/FK, Super/Sub Type 조정 수행
    6. 성능관점에서 검증

제 2 절. 정규화와 성능

1. 정규화를 통한 성능 향상 전략

- 정규화 수행한 데이터 모델은
    - 조회 성능 : 처리 조건에 따라 성능이 좋아질 수도 나빠질 수도 있음
    - CUD 성능 : 성능 향상됨
    - (일반적으로 조회 성능과 CUD의 성능은 Trade-Off)

2.3.4.5. 반정규화된 테이블의 성능 저하사례

- 정규화를 통해 조회 및 트랜잭션 시 성능을 높이는 사례

6. 함수적 종속성(FD) 에 근거한 정규화 수행 필요

- 실제 정규화 수행시 매우 중요한 부분
- 주민등록 번호로 이름, 출생지, 호주를 결정할 수 있는 경우
    - 주민등록번호 -> (이름, 출생지, 주소) 로 표현

제 3 절. 반정규화와 성능

1. 반정규화를 통한 성능향상 전략

- 반정규화(De-Normalization)
    - Data 의 중복을 허용하여 데이터 조회시 성능을 향상 시키는 역할
- 적용방법
    1. 반정규화 대상조사

2. 다른 방법유도 검토
3. 반정규화 적용

2. 반정규화 기법

- 테이블 반정규화
    - 테이블 병합
    - 테이블 분할
        - 수직분할 : Col 단위 (Disk I/O 를 분산처리 하기 위해 TB 1:1 로 분리)
        - 수평분할 : ROW 단위 (관계 없음)
    - 테이블 추가
        - 중복테이블 추가 : 업무나 서버 다른경우
        - 통계테이블 추가
        - 이력테이블 추가
        - 부분테이블 추가 : Disk I/O 줄이기 위해 자주 이용하는 Col 만 모아서 만든 중복 TB
- 컬럼 반정규화
    - 중복칼럼 추가
    - 파생칼럼 추가
    - 이력테이블 칼럼추가
    - PK에 의한 칼럼추가
    - 응용시스템 오작동을 위한 칼럼 추가
- 관계 반정규화
    - 중복관계 추가 : 멀리있는 TB 에 조인할 경우 성능저하 막기위해 중복 관계 추가. 데이터 무결성에 영향 안줌.

3.4. 정규화가 잘 정의돈 데이터 모델에서 성능이 저하될 수 있는(저하되는) 경우

- 가 있다.

제 4 절. 대량 데이터에 따른 성능

1. 대량 데이터발생에 따른 테이블 분할 개요

- 대량의 데이터가 존재하는 테이블에 많은 트랜잭션이 발생하여 성능이 저하되는 테이블 구조에 대해 수평/숯ㄱ 분할 설계를 통해 성능저하를 예방한다.
- ROW 길이가 너무 길어지는 경우
    - Row Chaining , Row Migration 으로 I/O 다 많이져 성능저하
        - Row Chaining : ROW 가 너무 길어 한 data block 에 저장 못하여 2개 이상의 block 에 담겨있는 경우
        - Row Migration : 긴 ROW 에서 수정 발생 시 해당 ROW 의 Data Block 을 초과하여 다른 block 의 빈 공간에 저장되는 경우.

2. 한 테이블에 많은 수의 컬럼을 가지고 있는 경우

- Disk I/O 양이 늘어서 성능이 저하됨
- 1:1 관계로 분리하여 개선가능

3. 대량 데이터 저장 및 처리로 인한 성능

- 테이블에 많은 양의 데이터가 예상될 경우 물리적으로 여러 TB로 분리하는 **파티셔닝** 적용 가능
    - 예를들어 1억개의 data 라면 1000만개씩 파티션을 만들어 저장함
    - TB 를 날짜, 숫자 등으로 분리 가능하면 RANGE PARTITION 을 적용하면 좋다.
    - TB 를 "지역 사업소"와 같이 특정 값에 따라 분리 저장하는 경우 LIST PARTITION 을 적용.
    - 해싱 함수를 이용하여 Data 를 랜덤하게 저장하는 HASH PARTITION 도 있다.

4. 테이블에 대한 수평분할/수직분할의 절차

- 절차

    1. 데이터 모델링 완성
    2. DB 용량산정
    3. 대량 데이터가 처리되는 TB에 대해 트랜잭션 처리 패턴 분석
    4. 칼럼 단위로 집중화된 처리가 발생하는지, 로우단위로 집중화된 처리가 발생하는지 분석하여 집중화된 곳을 분리

- Col 이 너무 많은 경우 => 1:1 형태로 TB 분리
- data 자체가 너무 많은경우 => 파티셔닝

제 5 절. 데이터베이스 구조와 성능

1. 슈퍼타입/서브타입 모델의 성능고려 방법

- 데이터의 특징을 고려하여 효과적으로 표현 가능
- 공통부분 : 슈퍼타입 , 엔티티안의 나머지 속성 : 서브타입
- 트랜잭션 성능(데이터 양 & 트랜잭션 유형)을 고려하여 변환해야 성능저하 예방가능.

2. 인덱스 특성을 고려한 PK/FK DB 성능향상

- TB 에 발생되는 트랜잭션의 조회 패턴에 따라 PK/FK 칼럼의 순서 조정해야 함
- 순서에 따라 조회 속도가 다르다. WHERE 절에서 처리하는 패턴 보고 결정

3. 물리적인 테이블에 FK 제약이 걸려있지 않을 경우 인덱스 미생성으로 성능저하

- FK 생성시 조인 조건으로 대부분 활용되므로 CONSTRAINT 잘 걸어야 함

제 6 절. 분산 데이터베이스와 성능

1. 분산 데이터베이스의 개요

- 분산 데이터베이스 : DB 를 연결하는 빠른 네트워크 환경을 이용하여 DB 를 여러 곳으로 분산시켜 논리적으로 동일한 시스템이지만 물리적으로 분산시켜 사용성/성능을 극대화 시킨 DB

2. 분산 데이터베이스의 투명성

- 분할, 위치, 지역사상, 중복, 장애, 병행 등 6가지 투명성 만족해야 함

3. 분산 DB의 적용법 및 장단점

4. 분산설계 방향성

5. DB 분산설계 가치

6. 분산 DB 적용 기법

- TB 위치 분산
- TB 분할(Fregmentation) 분산
    - 수평 / 수직
- TB 복제(Replication) 분산
    - 부분 / 광역
- TB 요약 분산

7. 성능향상 사례

- 효과적인 경우
    - 성능
    - 공통/기준/마스터 데이터 대한 분산환경
    - 실시간 동기화가 요구되지 않을 경우. or 거의 실시간(Near Real Time)
    - 서버 부하 분산
    - 백업