1과목 2장 데이터 모델과 성능

● 성능 데이터 모델링

DB 성능 향상을 목적으로 설계단계의 데이터 모델링 때부터 정규화, 반정규화, 테이블통합, 테이블분할, 조인구조, PK, FK, 등 여러 가지 성능과 관련된 사항이 데이터 모델링에 반영될 수 있도록 하는 것

성능 저하에 따른 유지보수 비용을 최소화 할 수 있다

● 성능 데이터 모델링 고려사항 순서

(1) 데이터 모델링을 할 때 정규화를 정확하게 수행
(2) DB 용량산정 수행
(3) DB에 발생되는 트랜잭션의 유형 파악
(4) 용량과 트랜잭션의 유형에 따라 반정규화 수행
(5) 이력모델의 조정, PK/FK조정, 슈퍼/서브타입 조정
(6) 성능관점에서 데이터 모델을 검증

● 정규화

- 데이터 일관성, 최소한의 데이터 중복, 최대한의 데이터 유연성을 위한 방법, 데이터를 분해하는 과정
- 데이터 중복을 제거, 데이터 모델의 독립성을 확보

● 정규화 절차 (1~3은 중요하므로 확실히 이해하기)

(1) 1정규화 : 속성의 원자성 확보/ 기본키를 설정
(2) 2정규화 : 기본키가 2개 이상의 속성으로 이루어진 경우, 부분 함수 종속성을 제거(분해)
(3) 3정규화 : 기본키를 제외한 칼럼 간에 종속성 제거 (이행함수 종속성 제거)
(4) BCNF : 기본키를 제외하고 후보키가 있는 경우, 후보키가 기본키를 종속시키면 분해
(5) 4정규화 : 여러 칼럼들이 하나의 칼럼을 종속시키는 경우 분해하여 다중 값 종속성 제거
(6) 5정규화 : 조인에 의해서 종속성이 발생되는 경우 분해

● 정규화의 문제점

- 데이터 조회(SELECT) 시에 조인을 유발하기 때문에 CPU와 메모리를 많이 사용한다.

● 반정규화

- DB의 성능 향상을 위하여 데이터 중복을 허용하고 조인을 줄이는 방법
- 조회속도를 향상시키지만 데이터 모델의 유연성은 낮아짐

● 반정규화 절차

(1) 대상 조사 및 검토(데이터 처리 범위 및 빈도수, 통계성)
(2) 다른 방법 검토(뷰, 클러스터링, 인덱스 조정, 응용 프로그램, 파티션)
(3) 반정규화 수행(테이블, 속성, 관계 등을 반정규화)

√ 클러스터링 : 인덱스 정보를 저장할 때 물리적으로 정렬해서 저장하는 방법, 조회 시에 인접 블록을 연속적으로 읽기 때문에 성능이 향상된다

● 반정규화를 수행하는 경우(반정규화 대상 조사)

- 정규화에 충실(조인이 많음)하여 조회가 느려지는 경우
- 다량의 범위를 자주 처리해야 하는 경우
- 특정 범위의 데이터만 자주 처리하는 경우
- 통계/집계 정보가 자주 요구되는 경우
- SORTING, ORDER BY는 반정규화 대상이 아님

● 다른 방법 검토

(1) 지나치게 많은 조인이 걸려 데이터를 조회하는 작업이 기술적으로 어려울 경우 VIEW를 사용한다.
(2) 대량의 데이터 처리나 부분 처리에 의해 성능이 저하되는 경우 클러스터링을 적용하거나 인덱스를 조정한다.
(3) 대량의 데이터는 PK의 성격에 따라 부분적인 테이블로 분리할 수 있다. (파티셔닝 기법)
(4) 응용 애플리케이션에서 로직을 구사하는 방법을 변경함으로써 성능을 향상시킬 수 있다.

● 반정규화 기법(테이블 반정규화 – 테이블 병합)

(1) 1:1 관계를 하나의 테이블로 병합하여 성능 향상
(2) 1:M 관계를 하나의 테이블로 병합하여 성능 향상
(3) 슈퍼/서브 관계를 통합하여 성능 향상

● 반정규화 기법(테이블 반정규화 – 테이블 분할)

(1) 수직분할 : 하나의 테이블을 두 개 이상의 테이블로 분할한다. (칼럼을 기준으로 1:1로 분리)
(2) 수평분할 : 하나의 테이블에 있는 값을 기준으로 테이블을 분할한다. (레코드를 기준으로 분리)

● 반정규화 기법(테이블 반정규화 – 테이블 추가)

(1) 중복 테이블 추가
다른 업무이거나 서버가 다른 경우 동일한 테이블 구조를 중복하여 원격조인을 제거하여 성능 향상
(2) 통계 테이블 추가
SUM, AVG 등을 미리 수행하여 계산해 둠으로써 조회 시 성능 향상
(3) 이력 테이블 추가
이력테이블 중에서 마스터 테이블에 존재하는 레코드를 중복하여 이력테이블에 존재시켜 성능 향상
(4) 부분 테이블 추가
하나의 테이블의 전체 칼럼 중 자주 이용하는 집중화된 칼럼들이 있을 때 디스크 I/O를 줄이기 위해 해당 칼럼들을 모아놓은 별도의 반정규화된 테이블을 생성

● 반정규화 기법(칼럼 반정규화)

(1) 중복칼럼
조인에 의한 처리를 할 때 성능저하를 예방하기 위해 중복된 칼럼을 위치시킴

(2) 파생칼럼
트랜잭션이 처리되는 시점에 계산에 의해 발생되는 성능저하를 예방 / 미리 값을 계산하여 칼럼에 보관

(3) 이력 테이블
대량의 이력데이터를 처리할 때 불특정 일자 조회나 최근 값을 조회할 때 나타날 수 있는 성능저하를 예방하기 위해 이력테이블에 기능성 칼럼(최근값 여부, 시작과 종료일자)을 추가함

(4) PK에 의한 칼럼 추가
이미 PK안에 데이터가 존재하지만 성능향상을 위해 일반속성으로 포함하는 경우

(5) 응용시스템 오작동을 위한 칼럼 추가
업무적으로는 의미가 없지만 사용자의 실수로 원래 값으로 복구하기 원하는 경우 이전 데이터를 임시적으로 중복하여 보관하는 기법

● 반정규화 기법(관계 반정규화)

중복관계 추가 : 데이터를 처리하기 위한 여러 경로를 거쳐 조인이 가능하지만 이 때 발생할 수 있는 성능저하를 예방하기 위해 추가적인 관계를 맺는 방법

√ **로우 체이닝** : 행의 길이가 너무 길어서 데이터 블록 하나에 데이터가 모두 저장되지 않고 두 개 이상의 블록에 걸쳐 하나의 행이 저장되어 있는 형태

√ **로우 마이그레이션** : 데이터블록에서 수정이 발생하면 수정된 데이터를 해당 데이터 블록에서 저장하지 못하고 다른 블록의 빈 공간을 찾아 저장하는 방식

√ 로우 체이닝과 로우 마이그레이션이 발생하면 많은 디스크 I/O가 발생하여 성능저하 발생, 트랜잭션을 분석하여 적절하게 1:1 관계로 분리해야 한다

● 파티셔닝 기법

- 파티션을 사용하여 테이블을 분할할 수 있다.
- 파티션을 사용하면 논리적으로는 하나의 테이블이지만 여러 개의 데이터 파일에 분산되어 저장된다.

(1) RANGE : 대상 테이블이 날짜 또는 숫자값으로 분리가 가능하고 각 영역별로 트랜잭션이 분리되는 경우

(2) LIST : 지점, 사업소 등 핵심적인 코드값으로 PK가 구성되어 있고 대량의 데이터가 있는 테이블의 경우

(3) HASH : 지정된 HASH 조건에 따라 해쉬 알고리즘이 적용되어 테이블이 분리

● 파티션 테이블의 장점

- 데이터 조회 시 접근범위가 줄어들기 때문에 성능이 향상
- 데이터가 분할되어 있기 때문에 I/O성능 향상
- 각 파티션을 독립적으로 백업/복구 가능

● 테이블에 대한 수평/수직분할 절차

(1) 데이터 모델링 완성

(2) DB 용량산정 수행

(3) 대량 데이터가 처리되는 테이블에 대해 트랜잭션 처리 패턴을 분석

(4) 칼럼 단위로 집중화된 처리가 발생하는지, 로우 단위로 집중화된 처리가 발생하는지 분석하여 집중화된 단위로 테이블을 분리하는 것을 검토한다.

● 슈퍼타입과 서브타입

- 엔터티들의 공통의 부분은 슈퍼타입
- 공통으로부터 상속받아 다른 엔터티와 차이가 있는 속성에 대해서는 별도의 서브엔터티로 구분

ex)고객 – (개인고객/법인고객)
- 고객은 슈퍼타입, 개인/법인고객은 서브타입
- 즉, 부모와 자식 간의 관계가 나타난다.

● 슈퍼/서브타입 데이터 모델의 변환

슈퍼/서브타입에 대한 변환을 잘못하면 성능이 저하된다(트랜잭션의 특성을 고려하지 않고 테이블을 설계했기 때문에)

(1) 트랜잭션은 항상 일괄로 처리하는데 테이블은 개별로 유지되어 UNION 연산에 의해 성능이 저하될 수 있다.

(2) 트랜잭션은 항상 서브타입 개별로 처리하는데 테이블은 하나로 통합되어 있어 불필요하게 많은 양의 데이터가 집약되어 있어 성능이 저하되는 경우가 있다.

(3) 트랜잭션은 항상 슈퍼+서브타입을 공통으로 처리하는데 개별로 유지되어 있거나 하나의 테이블로 집약되어 있어 성능이 저하되는 경우가 있다.

● 슈퍼타입 및 서브타입 변환 방법

(1) OneToOne Type
슈퍼타입과 서브타입을 개별 테이블로 도출
테이블 수가 많아서 조인이 많이 발생, 관리가 어려움

(2) Plus Type
슈퍼타입+서브타입 테이블로 도출
조인이 발생하고 관리가 어렵다

(3) Single Type / All in One Type
슈퍼타입과 서브타입을 하나의 테이블로 도출
조인 성능이 좋고 관리가 편하지만 I/O성능이 나쁨

● 인덱스 특성을 고려한 PK/FK DB 성능 향상

- 인덱스 구성 시 앞쪽에 위치한 속성의 값이 '=' 이거나 최소한의 범위 'BETWEEN' '!=' 가 들어와야 효율적이다.
- 물리적인 테이블에 FK 제약을 걸었을 때는 반드시 FK인덱스를 생성하도록 한다.
- FK제약이 걸리지 않았을 경우에는 FK인덱스를 생성하는 것을 기본정책으로 하되 발생되는 트랜잭션에 의해 거의 활동되지 않았을 때에만 FK인덱스를 지운다

√ PK는 인덱스가 자동으로 생성된다.

√ FK에도 인덱스를 생성할 필요가 있다.

● 분산 데이터베이스

(1) 여러 곳으로 분산되어 있는 DB를 하나의 가상 시스템으로 사용할 수 있도록 한 DB

(2) 논리적으로 동일한 시스템에 속하지만, 컴퓨터 네트워크를 통해 물리적으로 분산되어 있는 데이터집합

● 분산 데이터베이스의 투명성 종류

(1) 분할 투명성(단편화) : 하나의 논리적 Relation이 여러 단편으로 분할되어 각 사본이 여러 site에 저장

(2) 위치 투명성 : 사용하려는 데이터의 저장 장소 명시 불필요, 위치정보를 시스템 카탈로그에 유지

(3) 지역사상 투명성 : 지역 DMBS와 물리적 DB 사이의 사상이 보장되어야 함

(4) 중복 투명성 : DB 객체가 여러 시스템에 중복 되어도 데이터의 일관성 유지

(5) 장애 투명성 : 각 지역의 시스템이나 통신망에 이상이 발생해도 데이터의 무결성은 보장되어야 함

(6) 병행 투명성 : 여러 사용자가 동시에 분산 DB에 대한 트랜잭션을 수행해도 결과에 이상이 없어야 함

● 분산 데이터베이스 설계 방식

상향식 설계 : 지역 스키마 작성 → 전역 스키마 작성

하향식 설계 : 전역 스키마 작성 → 지역 스키마 작성

● 분산 데이터베이스 장단점

(1) 장점 : 지역 자치성, 신뢰성, 가용성, 융통성, 빠른 응답속도, 각 지역 사용자 요구 수용, 시스템 용량 확장이 쉬움

(2) 단점 : 관리와 통제가 어려움, 오류의 잠재성 증대, 설계가 복잡함, 데이터 무결성 관리가 어려움, 보안 관리가 어려움

● 분산 DB 적용 기법

(1) 테이블 위치 분산 : 설계된 테이블을 본사/지사단위로 분산

(2) 테이블 분할 분산 : 각각의 테이블을 쪼개어 분산

- 수평분할 : 로우(행) 단위로 분리
- 수직분할 : 칼럼 단위로 분리

(3) 테이블 복제 분산 : 동일한 테이블을 다른 지역이나 서버에서 동시에 생성하여 관리하는 유형

- 부분복제 : 마스터 DB에서 테이블의 일부의 내용만 다른 지역이나 서버에 위치
- 광역복제 : 마스터 DB 테이블의 내용을 각 지역이나 서버에 존재

(4) 테이블 요약 분산 : 지역 간 / 서버 간에 데이터가 비슷하지만 서로 다른 유형으로 존재하는 경우

- 분석요약 : 동일한 테이블 구조를 가지고 있으면서 분산되어 있는 동일한 내용의 데이터를 이용하여 통합된 데이터를 산출
- 통합요약 : 분산되어 있는 다른 내용의 데이터를 이용하여 통합된 데이터를 산출

√ GIS(Global Single Instance) : 통합데이터 구조

● 분산 데이터베이스 설계를 고려해야 하는 이유

(1) 성능이 중요한 사이트

(2) 공통코드, 기준정보, 마스터 데이터의 성능 향상

(3) 거의 실시간의 업무적인 특징을 가지고 있는 경우

(4) 특정 서버에 부하가 집중되어 부하를 분산해야 하는 경우

(5) 백업 사이트를 구성하는 경우