## 핵심정리

### ◦ 성능 데이터 모델링

DB 성능향상을 목적으로 설계단계의 데이터 모델링 때부터 정규화, 반정규화, 테이블 통합, 테이블 분할, 조인구조, PK,FK 등 여러 가지 성능과 관련된 사항이 데이터 모델링에 반영될 수 있도록 하는 것

분석/설계 단계에서 데이터 모델에 성능을 고려한 데이터 모델링을 수행할 경우 성능 저하에 따른 재업무 비용을 최소화할 수 있음. 데이터의 증가가 빠를수록 성능저하에 따른 성능개선비용은 기하급수적으로 증가하게 된다.

### - 성능 데이터 모델링 고려사항 순서

1. 데이터 모델링을 할 때 정규화를 정확하게 수행
2. DB 용량 산정을 수행
3. DB에 발생되는 트랜잭션의 유형을 파악
4. 용량과 트랜잭션의 유형에 따라 반정규화 수행
5. 이력모델의 조정, PK/FK 조정, 슈퍼/서브 타입 조정
6. 성능관점에서 데이터 모델을 검증

### ◦ 함수적 종속성

데이터들이 어떤 기준 값에 의해 종속되는 현상

### ◦ (-) 정규화

반복적인 데이터를 분리하고 각 데이터가 종속된 테이블에 적절하게 배치되도록 하는 것

- 1차 정규화: 같은 성격, 내용 컬럼이 연속될 때 컬럼 제거, 테이블 생성
- 2차 정규화: PK 복합키 구성일 때 부분적 함수 종속 관계 테이블 분리
- 3차 정규화: PK가 아닌 일반 컬럼에 의존하는 컬럼분리

### ◦ (+) 반정규화

정규화된 엔터티, 속성, 관계에 대해 시스템의 성능향상과 개발과 운영의 단순화를 위해 중복, 통합, 분리 등을 수행하는 데이터 모델링의 기법. 조회시 디스크 I/O가 많거나 경로가 먼 조인에 의한 성능 저하를 막기위해 수행

일반적으로 정규화 H 입력/수정/삭제 성능이 향상되며 반정규화시 조인 성능이 향상된다.

### ◦ 반정규화 절차

1. 반정규화 대상조사 (범위처리 빈도수, 범위, 통계성)
   1. 자주 사용되는 테이블에 접근하는 프로세스의 수가 많고 항상 일정한 범위만을 조회하는 경우
   2. 테이블에 대량의 데이터가 있고 대량의 데이터 범위를 자주 처리하는 경우에 처리 범위를 일정하게 줄이지 않으면 성능을 보장할 수 없는 경우
   3. 통계성 프로세스에 의해 통계 정보로 필요로 할 때 별도의 통계 테이블을 생성한다.
   4. 테이블에 지나치게 많은 조인이 걸려 데이터를 조회하는 작업이 기술적으로 어려울 경우.
2. 다른 방법 유도 검토 (뷰, 클러스터링, 인덱스 조정)
   1. VIEW 사용 : 지나치게 많은 조인이 걸려 데이터 조회하는 작업이 기술적으로 어려울 경우 VIEW를 사용 (뷰가 성능향상시키는건 X)
   2. 클러스터링 : 대량의 데이터 처리나 범위처리에 의해 성능이 저하되는 경우 클러스터링을 적용하거나 인덱스를 조정함 (조회가 대부분인 때 클러스터링 적용)
   3. 파티셔닝 : 대량의 데이터는 PK의 성격에 따라 부분적인 테이블로 분리할 수 있다. 파티셔닝 키에 의해 물리적 저장공간 분리
   4. 캐시 : 응용 애플리케이션에서 로직은 구사하는 방법은 변경함으로써 성능을 향상 시킬 수 있다.

## 3. 반 정규화 적용

### 3-1. 테이블 반정규화

테이블 병합 ( 1 : 1 관계, 1 : M 관계, 슈퍼 / 서브 타입)

1. 1 : 1 관계를 통하여 성능향상
2. 1 : M 관계를 통하여 성능 향상
3. 슈퍼 / 서브 관계를 통하여 성능 향상

테이블 분할 (수직 분할, 수평분할)

1. 수직분할 : 컬럼 단위 테이블을 디스크 I/O로 분산처리하기 위해 테이블을 1 : 1로 분리하여 성능향상
2. 수평분할 : 로우 단위로 집중 발생되는 트랜잭션을 분석하여 디스크 I/O 및 데이터 접근의 효율성을 높여 성능을 향상하기 위해 로우 단위로 테이블을 쪼갬

테이블 추가

1. 중복 : 다른 업무이거나 서버가 다른 경우 동일한 테이블 구조를 중복하여 원격조인을 제거하여 성능 향상
2. 통계 : SUM, AVG 등을 미리 수행하여 계산해 둠으로써 조회시 성능 향상
3. 이력 : 이력 테이블 중에서 마스터 테이블에 존재하는 Record를 중복하여 이력테이블에 존재시켜 성능 향상
4. 부분 : 하나의 테이블의 전체 컬럼 중 자주 이용하는 집중화된 칼럼들이 있을 때 디스크 I/O를 줄이기 위해 해당 칼럼들을 별도의 반정규화 된 테이블로 생성

### 3-2. 컬럼 반정규화

1. 중복 칼럼 추가 : 조인에 의해 처리할 때 성능 저하를 예방하기 위해 중복된 컬럼을 추가시킴.
2. 파생컬럼 추가 : 트랜잭션이 처리되는 시점에 계산에 의해 발생되는 성능저하를 예방하기 위해 미리 값을 계산하여 컬럼에 보관
3. 이력테이블 컬럼추가 : 대량의 이력데이터를 처리할 때 불특정 날 조회나 최근 값을 조회할 때 나타날 수 있는 성능저하를 예방하기 위해 이력테이블에 기능성 컬럼 (최근 값 여부, 시작과 종료 일자 등)을 추가함
4. 응용시스템 오작동을 위한 컬럼 추가 : 업무적으로는 의미가 없지만 사용자의 실수로 원래 값으로 복구하기 원하는 경우 이전 데이터를 임시적으로 중복하여 보관하는 기법
5. PK에 의한 컬럼 추가 : 단일 PK안에서 특정 값을 별도로 조회하는 경우 성능 저하 발생할 수 있어 의도적으로 추가

### 3-3. 관계 반정규화 : 무결성 유지

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

- 로우 체이닝

로우의 길이가 너무 길어서 데이터 블록 하나에 저장되지 않고 두개 이상의 블록에 걸쳐 하나의 로우가 있는 경우

- 로우 마이그레이션

데이터 블록에서 수정이 발생하면 수정된 데이터를 해당 데이터 블록에서 저장하지 못하고 다른 블록의 빈 공간에 저장하는 방식

· 로우 체이닝과 로우 마이그레이션이 발생하여 많은 블록에 데이터가 저장되면 DB 메모리에서 디스크 I/O가 발생할 때 많은 I/O가 발생하여 성능 저하 발생 가능.

· 트랜잭션을 분석하여 적절하게 1 : 1 관계로 분리함으로써 성능향상이 가능하도록 해야 함.

- PK에 의해 테이블을 분할하는 방법 (파티셔닝)

1. RANGE PARTITION : 대상 테이블이 날짜 또는 숫자 값으로 분리가 가능하고 각 영역별로 트랜잭션이 분리되는 경우
   ex) 요금_04년
2. LIST PARTITION : 지점, 사업소 등 핵심적인 코드값으로 PK가 구성되어 있고, 대량의 데이터가 있는 테이블의 경우
   ex) 고객_서울
3. HASH PARTITION : 지정된 HASH 조건에 따라 해시 알고리즘이 적용되어 테이블이 분리

◦ 테이블에 대한 수평/수직분할의 전자

1. 데이터 모델링을 완성한다.
2. DB 용량 산정을 한다.
3. 대량 데이터가 처리되는 테이블에 대해 트랜잭션 처리 패턴을 분석한다.
4. 컬럼 단위로 집중화된 처리가 발생하는 지, 로우 단위로 집중화된 처리가 발생하는지 분석하여 집중화된 단위로 테이블을 분리하는 것을 검토.
   - 컬럼 많음 → 1:1 분리 - 데이터 용량 → 파티셔닝

◦ 서브/슈퍼 타입 모델

업무를 구성하는 데이터를 공통과 차이점의 특징을 고려하여 효과적으로 표현해는 논리적 모델

-슈퍼타입: 공통부분 / 서브 타입: 공통으로부터 상속받아 다른 엔터티와 차이가 있는 특성

◦ 슈퍼/서브 타입 데이터 모델의 변환 개념

1. 개별로 발생되는 트랜잭션에 대해서는 개별 테이블로 구성 (One To One Type)
2. 슈퍼 + 서브 타입에 대해 발생되는 트랜잭션에 대해서는 슈퍼+서브 타입 테이블로 구성 (Plus Type)
3. 전체를 하나로 묶어 트랜잭션이 발생할 때는 하나의 테이블로 구성 (Single Type, All in One Type)

◦ 인덱스 특성을 고려한 PK/FK DB 성능 향상

인덱스의 특징은 여러개의 속성이 하나의 인덱스로 구성되어 있을 때 앞쪽에 위치한 속성의 값이 비교값 있어야 좋은 효율을 낼 수 있다.

(앞쪽에 위치한 속성값이 상수적 '=' 또는 최우선 Between <> 비교되어야 함)

◦ 분산 DB

1. 여러 곳으로 분산되어 있는 DB를 하나의 가상 시스템으로 사용할 수 있도록한 DB
2. 논리적으로 통일한 시스템에 속하지만, 컴퓨터 네트워크를 통해 물리적으로 분산되어 있는 데이터 집합

◦ 분산 DB를 만족하기 위한 6가지 투명성

1. 분할 투명성 (단편화): 하나의 논리적 Relation이 여러 단편으로 분할되어 각 사본이 여러 Site에 저장
2. 위치 투명성: 사용하는 데이터의 저장 장소 명시 불필요. 위치 정보가 시스템 카탈로그에 유지
3. 지역 사상 투명성: 지역 DBMS와 물리적 DB 서버의 Mapping 보장
4. 중복 투명성: DB 객체가 여러 Site에 중복되어 있는지 알 필요가 없는 성질
5. 장애 투명성: 구성 요소의 장애에 무관한 트랜잭션의 원자성 유지
6. 병행 투명성: 다수 트랜잭션 동시 수행시 결과의 일관성 유지. Timestamp나 분산 2단계 Locking 사용

◦ 분산 DB 장단점.

- 장점: 지역 자치성, 신뢰 가용성, 효용성, 융통성, 빠른 응답속도, 비용 절감, 각 지역 사용자 요구 수용.
- 단점: 비용 증가, 오류의 잠재성 증대, 설계 관리의 복잡성, 불균일한 응답 속도, 통제의 어려움, 데이터 무결성 위협

◦ 분산 DB 적용 기법

1. 테이블 위치 분산: 선계된 테이블을 부서나 지사 단위로 분산. 위치별 DB 분서 필요.
2. 테이블 분할 분산: 각 테이블은 쪼개어 분산
   - 수평분할: 로우 단위로 분리, 지사별로 다른 때, 중복 X
   - 수직 분할: 칼럼 단위로 분리, 각 테이블에 동일 PK 있어야 함.
3. 테이블 복제 분산: 동일한 테이블은 다른 지역이나 서버에서 동시에 생성하여 관리하는 유형
   - 부분 복제: 마스터 DB에서 테이블의 일부의 내용만 다른 지역이나 서버에 위치
   - 광역 복제: 마스터 DB 테이블의 내용을 각 지역이나 서버에 존재
4. 테이블 요약 분산: 지역 간에, 서버 간에 데이터가 비슷하지만 서로 다른 유형으로 존재하는 경우
   - 분석 요약: 동일한 테이블 구조를 가지고 있으면서 분산되어 있는 동일한 내용의 데이터를 이용하여 통합된 데이터를 산출하는 방식. EX) 판매실적 지사 A, 지사 B
   - 통합 요약: 분산되어 있는 다른 내용의 데이터를 이용하여 통합된 데이터를 산출하는 방식 EX) 판매실적 지사 A: C제품, 지사 B: D제품

◦ 분산 DB 설계를 고려해야 하는 경우

1. 성능이 중요한 서브
2. 공통코드, 기준 정보, 마스터 데이터의 성능향상
3. 실시간 동기화가 요구되지 않는 경우, Near Real time 특징을 가지고 있는 경우
4. 특정 서버에 부하가 집중되어 부하를 분산
5. 백업 사이트를 구성하는 경우.

33. 성능 데이터 모델링 순서

정규화 수행 후 용량산정과 트랜잭션의 유형을 파악하여 반정규화 수행. PK/FK 등을 조정하여 인덱스의 특징을 반영한 데이터 모델로 만들고, 이후에 데이터 모델을 검증

35. 함수 종속성 규칙 : 아래와 같은 보관금품장 엔터티에서 관서에 대한 정보가 반정규화 되어 있기 때문에 관서 정보를 조회할 때 성능 저하가 발생하고 있다. 이 엔터티에 대해 몇 차 정규화가 필요한 지와 분리된 스키마 구조를 가장 바르게 적용한 것은?

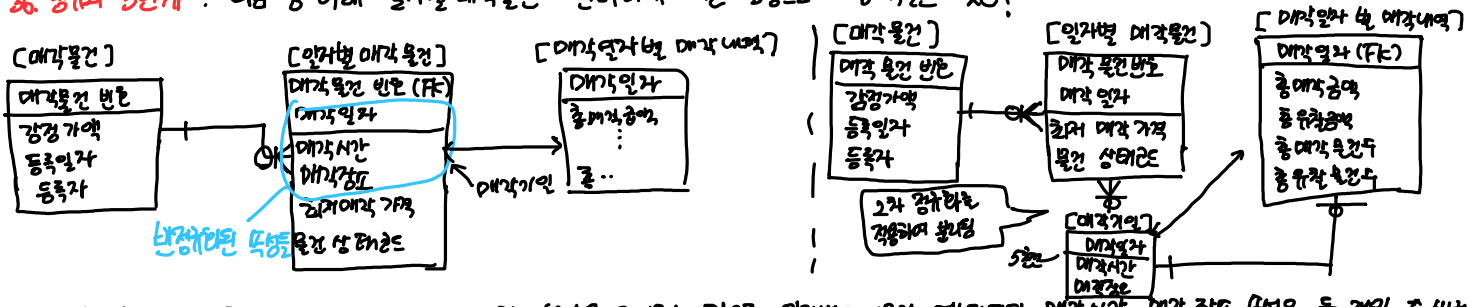
[보관금품장]

| 관서번호 |
| --- |
| 납부자번호 |
| 관할청번호 관서명 |
| 상태 관서 등록일자 |
| 걱금명 통신번호 |

함수종속성(FD): [관서번호, 납부자번호] → [걱금명, 통장번호], [관서번호] → [관할청 번호, 관서명, 상태, 관서등록일자]

함수종속성의 규칙에 따라 [관서번호] → [관할청 번호, 관서명, 상태, 관서등록일자] 인 관서번호가 PK인 엔터티가 2차 정규화로 분리.

36. 정규화 3단계 : 다음 중 아래 '일자별 매각물건' 엔터티에 대한 설명으로 가장 적절한 것은?




매각기일은 일자별로 매각이 시행되는 장소와 시간을 의미하는 것으로, 일자별 매각물건 엔터티의 매각시간, 매각 장소 속성은 두 개의 주식별자 속성 중 매각일자에만 종속되기 때문에 2차 정규화 대상이 된다.

그러므로 매각일자를 주식별자로하고 매각 시간과 매각장소 속성을 포함하는 매각기일 엔터티를 독립시킨다. 이때 매각기일 엔터티는 일자별매각물건의 주식별자 중 일부로부터 종속됐기 때문에 매각기일과 일자별 매각물건은 1:M 관계로 연결된다.

이와 같은 2차 정규화를 통해 특정 장소에서 어느기간 매각내역을 조회하고자 할 때 100만 건의 일자별 매각내역 데이터를 모두 읽어 해당하는 인스턴스들을 찾아 매각일자 별로 그룹핑한 후 매각일자별 매각내역과 조인할 필요가 없이 매우 적은 수의 매각기일 엔터티에서 특정 장소에 해당하는 매각일자들을 찾아 매각일자별 매각내역과 1:1로 바로 조인하면 되기 때문에 I/O를 현저하게 감소시킬 수 있어 성능 향상 효과를 얻을 수 있다.

37. 정규화

칼럼에 의한 반복적인 속성값을 갖는 형태는 속성의 원자성을 위배한 것 1차 정규화의 대상이 된다. 이와 같은 반복적인 속성 나열 형태에서는 각 속성에 대해 'or' 연산자로 연결된 조건들이 사용되는데, 이 때 어느 하나의 속성이라도 인덱스가 정의되어 있지 않게 되면 'or'로 연결된 모든 조건절들이 인덱스를 사용하지 않고 한 번의 전체 데이터 스캔으로 처리되게 되어 성능 저하가 나타날 수 있게 되며, 또한 모든 반복 속성에 인덱스를 생성하게 되면 검색 속도는 좋아지겠지만 반대급부적으로 너무 많은 인덱스로 인해 입력, 수정, 삭제의 성능이 저하되므로, 1차 정규화를 통해서 자연스럽게 문제가 해결될 수 있도록 해야 한다.

38. 정규화

컬럼 단위에서 중복된 경우도 1차 정규화의 대상이 된다. 이에 대한 분리는 1:M의 관계로 두 개의 엔터티로 분리된다.

39. 정규화

PK에 대해 반복이 되는 그룹(Repeating)이 존재하지 않으므로 1차 정규형이라고 할 수 X. 부분함수종속의 규칙을 가지고 있으므로 2차 정규형이라고 할 수 X. 2차 정규화의 대상이 되는 엔터티임.

40. 반정규화시 고려요소

- 다량 데이터 탐색의 경우 인덱스가 아닌 파티션 및 데이터 클러스터링 등의 다양한 물리저장 기법을 활용하여 성능 개선을 유도 O
  다만, 하나의 결과 셋을 추출하기 위해 다량의 데이터를 탐색하는 처리가 반복적으로 빈번하게 발생한다면 이때는 반정규화를 고려
- 이전 또한 이후 위치의 레코드에 대한 탐색은 window function으로 접근 가능하다.
- 참고로 테이블 이외에도 다양한 유형(대수 테이블의 N:M 연결 테이블 등)에 대하여 반정규화 테이블 적용이 필요할 수 있다.

44.

과도한 조인 → 성능 저하

45. 정규화 / 반정규화

한 테이블에 많은 컬럼들이 존재할 경우 데이터가 물리적으로 저장되는 디스크상에 넓게 분포할 가능성이 커지게 되어 디스크 I/O가 대량으로 발생할 수 있고, 이로 인해 성능이 저하될 수 있음. 따라서 트랜잭션이 접근하는 컬럼위주로 분석해서 자주 접근하는 컬럼들과 상대적으로 접근빈도가 낮은 컬럼들을 구분하여 1:1로 테이블을 분리하면 디스크 I/O가 줄어들어 성능을 향상 시킬 수 있다.
테이블 내에서 컬럼의 위치를 고정하는 것은 데이터 주로 채워지는 컬럼을 앞 쪽에 위치시키고, 데이터가 채워지지 않고 주로 NULL상태로 존재하는 컬럼들은 뒤쪽에 모아둠으로써 로우의 길이를 어느 정도 감소시킬 수 있으나, NULL 상태이면 컬럼에 대응해 데이터가 채워지게 될 경우 더 많은 로우 체인이 발생할 수도 있기 때문에 바람직한 해결책이라고 보기에 부족하며, 무엇보다도 데이터가 채워지지 않고 NULL 상태로 존재하게 되는 컬럼들이 많이 나타내는 경우, 너무 많은 엔터티들에 무리하게 통합성을 부여하여 통합을 수행했거나, 예측하기 어려운 미래 시점을 지향하여 고객하게 의욕적으로 속성을 확장한 경우 등에서 주로 나타나기 때문에 자주 사용되는 컬럼들이나 현 시점에서 주로 사용되는 컬럼들은 한데 모으고, 사용빈도가 낮은 컬럼들이나 DBMS 시점에서 사용될 것으로 예상되는 나머지 컬럼들을 한데 모아 별도의 1:1관계 엔터티로 분리하는 등의 데이터모델 설계 수정을 고려해 보는 것이 좋다.

48.

트랜잭션은 항상 전체를 통합하여 분석처리하는데 슈퍼-서브타입이 하나의 테이블로 통합되어 있으면 하나의 테이블에 집적된 데이터만 있어 처리할 수 있기 때문에 다른 형식에 비해 더 성능이 우수하다. (조인감소)

52.

Global Single Instance(GSI)는 통합된 한개의 인스턴스 즉, 통합 DB 구조를 의미함으로, 분산 DB와는 대치되는 개념이다.
공통코드, 기준정보 등과 같은 마스터 데이터를 한 곳에 두고 운영하는 경우 원격지에서의 접근이 빈번한 수록 실시간 입무처리에 대해 좋은 성능을 얻기가 어려울 수 있기 때문에 분산 환경에 복제분산을 하는 방법으로 분산DB를 구성할 수 있다. 또한 백업 사이트 구성에 대해서도 분산 환경으로 구성하여 적용할 수있다.