성능 데이터 모델링

① 아키텍처 모델링 … 데이터의 구조

테이블, 파티션

성능에 미치는 영향 ↑

② SQL 명령문

조인 수행원리

optimizer

실행 계획

정규화

- 방법

1차 … 원자성 확보 속성이 그게인 경우 1개씩 분할

2차 부분함수 종속성 제거

3차 이행 함수 종속 제거

BCNF 후보키가 삼목화는 것 제거 2개이상 복합키에서 후보키 들어나

- 이상현상

삽입 이상 / 삭제 이상

- 성능

select는 join 때문에 ↓ insert / update는 ↑ … 테이블이 작아지기 때문

반정규화

데이터 무결성을 해칠수 있음

대상 조사를 한다 ← 대량범위 / 범위처리 / 통계처리 → 응용 시스템변경 / 클러스터링/인덱스 / 뷰 → 이런 처리후 반정규화

- 테이블
  - 별합 : 슈퍼/서브
  - 분할 : 원 테이블 / 통계T / 중복T
- 속성
  1. 파생 컬럼 추가
  2. 응용 시스템에의한 오류시 원시컬럼
  3. 이력 컬럼 추가
  4. PK를 일반 속성으로 전환
  5. 중복 속성
- 관계 : 중복관계추가

대규모 데이터

row migration
row Chaining) 언제 일어나는지? 특징 → 해결방안: list / range / hash — partitioning 파티셔닝

특징: hash 관리 어렵고, range는 쉽다
range가 가장 많이 쓰임
관리용이성

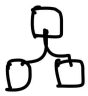
슈퍼/서브 타입

용량별로 처리 가능 < 적은 경우 one to one

트랜잭션이 개별로 들어가는 경우

큰경우 → 트랜잭션 유형에 따라서

① 공통/차이



plus 타입

② 전체 통합

상위 타입 하나로 통합된 테이블

분산 데이터 베이스

· 투명성

· 여러개의 서버로 뜯어 놓는 다는 점에서 반정규화와 유사
~ 데이터 무결성을 해침

조인 수행 원리

NL — 랜덤 엑세스. 대용량 소트 작업시 유리

Sort merge — 조인키 기준 정렬. 동가 / 비동가 조인

Hash ·· 동가 조인만. 선행 테이블이 작음. hash 처리를 위한 별도 저장공간 필요.

옵티마이저

CBO cost base optimizer — 경로에서 가장 싼것

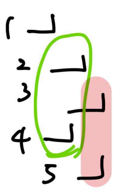
RBO role 〃 — 규칙에 따라

인덱스

언제 사용? (부정형, like, 형변환 때 사용 X)

인덱스 사용시 성능↓(느려짐): insert, update, delet ... DML의 성능 (계속 인덱스를 새로 생성해야 함)

선행 계획



(3-2)-(5-4)-1

같은 레벨은 동등이고
제일 앞에껀 뒤로