**제 3장 옵티마이저 원리**

1) 옵티마이저 소개

(가) 옵티마이저란?

- 쿼리 실행을 위해 후보군이 될 실행 계획을 찾는다

- 오브젝트 통계 및 시스템 통계 정보를 이용해 실행 계획의 예상비용 산정

- 각 실행 계획을 비교하여 최저 비용의 실행 계획 선택

(나) 옵티마이저 종류

- 옵티마이저 종류로는 규칙 / 비용 기반 옵티마이저가 있다

- 규칙기반 옵티마이저 : 미리 정해 높은 규칙에 따라 액세스 경로 평가 및 실행계획 선택

- 비용기반 옵티마이저 : 쿼리를 수행하는데 소요되는 일량 또는 시간을 비용으로 산정

-> 비용은 테이블, 인덱스에 대한 통계 정보를 기초하여 산정

(다) SQL 최적화 과정

PARSER(문법, 의미 오류 체크) -> OPTIMIZER (QUERY TRANSFORMER-표준 형태로, ESTIMATOR –통계 정보를 이용해 총 비용 계산, PLAN GENERATOR) -> ROW-SOURCE GENERATOR(엔진이 실제 실행할 수 있는 코드 형태) -> SQL ENGINE (실행)

(라) 최적화 목표

- 전체 처리속도 최적화 : ALL\_ROWS(전체 다), FIRST\_ROWS(일부만 읽고 끝), CHOOSE (옵티마이저가 선택) 등의 방법을 통해 힌트를 줌. ALL\_ROWS가 기본 전제로 실행계획을 선택함

- 최초 응답속도 최적화 : ALL\_ROWS가 아닌 FIRST\_ROWS를 통해 일부만 읽다 멈추는 것을 전제로 실행계획 수립

2) 옵티마이저 행동에 영향을 미치는 요소

가) SQL 연산자 형태 : 연산자나 형태를 어떤걸 사용했는지에 따라 실행계획이 달라짐

나) 옵티마이징 팩터 : 인덱스, IOT, C.F, 파티셔닝, MV 구성에 따라 변경

다) DBMS 제약 설정 : PK, FK, NOT NULL, CHECK 등은 성능 최적화에 매우 중요한 요소

라) 옵티마이저 힌트 마) 통계정보 바) 옵티마이저 관련 파라미터 사) DBMS 버전과 종류

**3. 옵티마이저의 한계**

가) 옵티마이징 팩터의 부족 : 효과적 인덱스, IOT, 클러스터링, 파티셔닝 등의 팩터제공 필요

나) 통계정보의 부정확성

다) 바인드 변수 사용 시 균등분포 가정 -> OPTIC\_BIND\_PEAK 변수 사용

라) 비현실적인 가정

마) 규칙에 의존하는 CBO : 부분적으로 규칙에 의존

바) 하드웨어 성능 특성 -> 실제 운영하는 환경이랑 다름!. 시스템 통계정보 필요!

**4. 통계정보를 이용한 비용 계산 원리**

가) 선택도 : 1/DISTINCT

나) 카디널리티 : 총 로우수 x 선택도 = 총 로우수 / DISTINCT VALUE 수

다) 히스토그램 : DISTINCT가 적을 땐 도수분포, 많을 땐 높이균형 히스토그램

라) 비용 : CBO는 비용기반으로 최적화 수행. 비용은 쿼리를 수행하는 일량 또는 시간으로 예상치. I/O(블락을 몇 개 읽는지), CPU, CHOOSE(시스템통계 있을 시 cpu모델. 없을시 I/O) 비용 모델 등이 있다.

**5. 옵티마이저 힌트**

가) 힌트가 무시되는 경우

- 문법/의미적으로 안 맞게 기술한 힌트

\*의미적으로 안 맞는 다는 건 서브쿼리에 unnest와 subq를 같이 기술한 경우 등을 말함

- 잘못된 참조 사용 : 없는 테이블이나 Alias 사용. 없는 인덱스명을 지정

- 논리적으로 불가능한 액세스 경로 : 등치 조건이 없는데 해쉬 조인 유도, NULL이 허용된 칼럼에 대해 인덱스를 이용 전체 건수 count 등

- 버그 : 위 5가지 조건에 해당하지 않으면 무조건 수행되며 사용자 명령어로 인식. SQL 서버의 경우 힌트에서 사용한 인덱스를 삭제하였을 경우 에러(안정성은 낮으나 DBA 조정가능), Oracle은 그냥 수행(안정성은 높으나 성능의 문제)