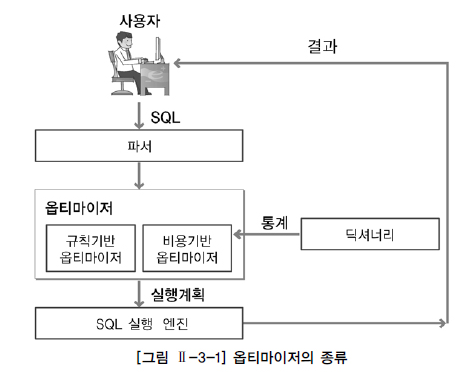
**[SQL 최적화의 원리]**

**1.** **옵티마이저(Optimizer)와 실행계획**

Optimizer란? SQL을 처리하는 최저비용의 경로를 생성해주는 DBMS 내부엔진. 옵티마이저가 생성한 처리경로를 **실행 계획**이라 한다. 실행계획이 SQL 성능을 좌우함.

옵티마이저는 데이터 딕셔너리에 있는 통계 정보를 사용하여 예상되는 비용을 산정하고, 최저비용의 계획을 선택해서 SQL문을 실행한다.

**2.** **옵티마이저 종류**

****

- 파서: 구문분석, SQL 문법 검사

- 현재 대부분의 DBMS는 비용기반 옵티마이저만을 제공. 규칙기반 옵티마이저를 제공하더라도 비용기반 옵티마이저가 디폴트. 비용기반 옵티마이저는 딕셔너리로부터 오는 통계 정보가 부족할 경우, 성능 저하가 오는 한계가 있음.

**3.** **인덱스**

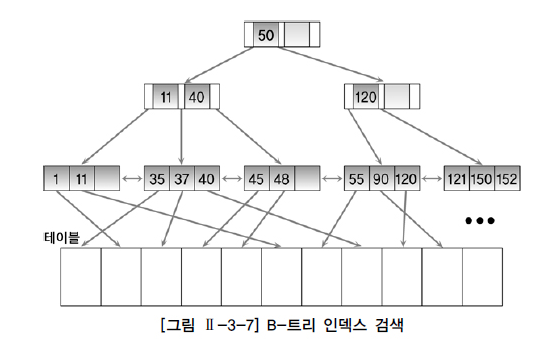
인덱스는 원하는 데이터를 쉽고 빠르게 찾을 수 있는 방법을 제공한다. 데이터와 매핑된 또 다른 테이블.

인덱스 키로 정렬되어 있어 원하는 데이터를 빠르게 조회하며, 오름차순/내림차순 탐색이 가능하다.

**비용 기반 옵티마이저**의 경우 적절한 인덱스가 존재하더라도 비용이 더 적으면 인덱스를 사용하기보다 전체 테이블 스캔이 유리다고 판단해 채택할 수 있다.

**규칙 기반 옵티마이저**의 규칙에 따라 적절한 인덱스가 존재하면 전체 테이블 스캔보다는 항상 인덱스를 사용하려고 한다.

하나의 테이블에 여러 인덱스를 생성할 수 있고 하나의 인덱스는 여러 개의 컬럼으로 구성될 수 있다.



가장 일반적인 인덱스: **B-트리 인덱스**(SQLD에서는 B-트리 인덱스만 다루는 듯)

**Root Block - Branch Block - Leaf Block**으로 구성되며 Branch Block은 다음 단계의 주소를 가지고 있는 포인터로 구성된다.

Leaf Block은 인덱스 키, ROWID로 구성되며, 인덱스 키는 정렬되어 저장되어 있고, Double Linked List 형태로 되어 있어 양방향 탐색이 가능하다. 인덱스 키를 읽으면 ROWID를 사용하여 행을 읽어 온다.

인덱스 생성: CREATE INDEX 문을 사용하여 생성한다. 인덱스는 한 개 이상의 컬럼을 사용해서 생성하며, 오름차순 정렬이 기본 설정이지만 DESC로 내림차순 가능.

**Q. 옵티마이저에 대한 설명으로 적절하지 않은 것은?**

1. 옵티마이저는 질의(query)에 대해 실행계획을 생성한다.
2. 비용기반 옵티마이저는 적절한 인덱스가 존재하면 반드시 인덱스를 사용한다.
3. 규칙기반 옵티마이저에서 제일 낮은 우선순위는 전체 테이블 스캔이다.
4. 비용 기반 옵티마이저는 비용 계산을 위해 다양한 통계정보를 사용한다.

**[인덱스 스캔]**

(1) 인덱스 유일 스캔(Index Unique SCAN): 인덱스 키의 값이 중복되지 않을 때, 해당 인덱스를 사용하는 경우. 검색 속도가 가장 빠르다. 기본키 또는 유니크 인덱스로 액세스해야 한다. 예) WHERE EMPNO = 1000

(2) 인덱스 범위 스캔(Index Range SCAN): SELECT 문에서 특정 범위를 조회하는 WHERE 문을 사용할 경우. Leaf Block의 특정 범위를 스캔하면 된다. LIKE나 BETWEEN이 그 예이다. WHERE EMPNO LIKE '1%'

(3) 인덱스 전체 스캔(Index Full SCAN): 인덱스에서 검색되는 인덱스 키가 많은 경우, Leaf Block의 처음부터 끝까지 전체를 읽는다. WHERE EMPNO LIKE '%' AND SAL > 0

(4) 테이블 전체 스캔(Table Full SCAN): 테이블의 데이터를 모두 읽는 것을 의미한다. 테이블을 읽을 때 High Watermark 이하까지만 Table Full SCAN한다.

(참고: High Watermark? – 데이터가 저장된 블록의 최상위 위치를 의미)

💡 힌트(Oracle Hint)란 무엇일까?

힌트란 SQL 튜닝의 핵심 부분으로 일종의 **지시 구문이다.** 즉, 오라클 옵티마이저(Optimizer)에게 SQL문 실행을 위한 데이터를 스캐닝하는 경로, 조인하는 방법 등을 알려주기 위해 SQL사용자가 SQL 구문에 작성하는 것을 뜻한다. 오라클이 항상 최적의 실행 경로를 만들어 내기는 불가능하기 때문에 **직접 최적의 실행 경로를 작성해 주는 것**이다. 사용자가 특정 SQL 문장에서 어떤 인덱스가 선택도가 높은지 알고 있는 경우 Optimizer에 의존한 실행 계획보다 훨씬 효율적인 실행 계획을 구사할 수 있다.

단, 힌트, 인덱스, 조인의 개념을 정확히 알고 사용하지 않은 무분별한 힌트의 사용은 성능의 저하를 초래하기 때문에 잘 알고 최적의 실행 경로를 알고 있을 경우 적절하게 사용하여야 한다. /\* \*/ 주석 달듯 사용

|  |
| --- |
| SELECT /\*+ RULE \*/ \* FROM EMP WHERE ROWID = 'AAAHYhASABAAALNJAAN'; |

**[옵티마이저 조인]**

1. Nested Loop Join (힌트: use\_nl)

- 하나의 테이블에서 데이터를 먼저 찾고, 그 다음 테이블을 조인하는 방식. 중첩 for문과 유사함.

- 행의 수가 더 적은 테이블을 외부 테이블(**outer table/driving table**)로 두고 큰 테이블을 내부 테이블(inner table)로 두는 것이 좋다

- RANDOM ACCESS가 발생하고 이로 인해 성능 지연 발생.

- 결과를 가능한 빨리 화면에 보여줘야 하는 인터넷 서비스에 유리한 조인 방식

2. Sort Merge Join

- 두 테이블을 SORT\_AREA 라는 메모리공간에 모두 로딩하고 SORT 실행

- 두 테이블을 모두 조인 컬럼을 기준으로 SORT한 다음에 병합한다.

- Merge가 완료되면 한번의 Full Scan으로 데이터를 검색한다.

- Sort로 인해 데이터 양이 많을 경우 성능이 떨어진다. 너무 큰 테이블의 경우 임시 영역에서 sort

- 힌트(use\_merge)를 사용해서 Sort Merge Join을 할 수 있다. ‘ordered’ 힌트와 같이 사용한다.

- NL join은 주로 랜덤 액세스 방식으로 데이터를 읽는 반면, SM Join은 주로 스캔 방식으로 데이터를 읽는다.

- NL Join에서 부담이 되던 넓은 범위의 데이터를 처리할 때 이용되던 조인 기법이다.

- 배치 작업에 많이 사용되는 조인 방식

3. Hash Join (힌트: use\_hash)

- 두 테이블 중 작은 테이블을 해시 메모리에 로딩하고 두 테이블의 조인 키를 사용해 해시테이블을 생성한다. 해시 함수를 사용하며 계산하기 때문에 CPU 연산을 많이 한다.

- 선행 테이블이 메모리에 충분히 로딩되는 크기여야 한다.

- Equal Join만 사용이 가능하다.

- 조인 칼럼의 인덱스가 존재하지 않을 경우에도 사용할 수 있는 기법이다.

- 선행 테이블을 Build Input이라 하며 후행 테이블을 Prove Input이라 한다.

**Q. 다음 중 Sort Merge Join 에 대한 설명으로 가장 부적절한 것은?**

1. 조인 칼럼에 적당한 인덱스가 없어서 Nested Loop 조인이 비효율적일 때 사용할 수 있다.
2. Driving Table의 개념이 중요하지 않은 조인 방식이다.
3. 조인 조건의 인덱스의 유무에 영향 받지 않는다.
4. EQUI 조인 조건에서만 동작한다.

**Q. 다음 중 Nested Loop Join 에 대한 설명으로 가장 부적절한 것은?**

1. Join 컬럼에 적절한 인덱스가 있어서 Natural Join이 효율적일 때 유용하다
2. Driving Table의 조인 데이터 양이 큰 영향을 주는 조인 방식이다.
3. Sort Merge Join 하기에 두 테이블이 너무 커서 Sort 부하가 심할 때 유용하다
4. 유니크 인덱스를 활용하여 수행시간이 적게 걸리는 소량 테이블을 온라인 조회하는 경우 유용하다.

**파티션 인덱스**

- Global/Local

- Global: 여러 개의 파티션에서 하나의 인덱스를 사용한다

- Local: 해당 파티션 별로 각자의 인덱스를 사용한다.

- Prefixed/Non Prefixed

- Prefixed: 파티션 키와 인덱스 키가 같다.

- Non-Prefixed: 파티션 키와 인덱스 키가 다르다

