**제 2절 조인의 원리와 활용**

1. **Nested Loop 조인**

- Random Access 위주의 조인방식 -> Index 필수!

- 조인을 한 레코드씩 순차적으로 진행

- 대용량 처리시 치명적 한계점 발생 -> 좁은 범위에서만 쓰임 -> OLTP 환경에 적합

- 인덱스 구성 전략이 매우 중요!

**2. Sort merge 조인**

- 소트 단계 : 양쪽 집합을 조인 컬럼 기준으로 정렬

- 머지 단계 : 정렬된 양쪽 집합을 merge

- 조인을 위해 실시간으로 인덱스를 생성하는 것과 같은 효과

- 양쪽 집합을 정렬 후 NL조인과 같은 방식으로 진행하지만, PGA 영역에서 처리 -> 빠름!

- 소트 부하만 감수한다면 버퍼 캐시에서 조인하는 NL조인 보다 유리

- 조인 컬럼에 인덱스 유무와 상관없이 조인

- 사용하는 경우 : Firtst 테이블에 소트 연산을 대체할 인덱스가 있을 때, 조인할 First 집합이 이미 정렬되어 있을 때, 조인 조건식이 등치 조건이 아닐 때

**3. Hash Join**

- Build Input Table -(Hash Fuction)-> Hash Area <-(Hash Fuction)- Probe Input

- S G A P G A

- 두 개의 테이블 중 작은 집합을 읽어 Hash Area에 적재 (Build Input)

- 반대쪽 큰 집합을 읽어 해시 테이블을 탐색 (Probe Input)

- 해시테이블을 탐색할 때 해시 함수를 사용

**- 등치조건만** 사용 가능!

-> NL 조인처럼 Random 액세스 부하 없으며 소트머지처럼 소트 부하도 없음.

- 단, Build Input이 Hash Area 크기 안에 들어가는 것이 성능의 성패를 좌우

- 해시 조인은 래치 획득과정이 없는 PGA에서 처리해서 빠른 탐색과정

- 사용 기준

1. 한쪽 테이블이 Hash Area에 담길 정도로 **충분히 작아야 함**
2. Build Input 해시 키 컬럼에 중복 값이 거의 없어야 함 -> Hash Chain(경로)이 길어짐
3. 조인 컬럼에 적당한 인덱스가 없어 NL조인이 비효율적일 때
4. NL 조인시 액세스 량이 많아 **Random 액세스 부하**가 심할 때
5. 소트머지 조인을 하기엔 두 테이블이 너무 커서 소트 부하가 심할 때
6. 수행빈도가 **낮고** 수행시간이 오래 걸리는 대용량 테이블을 조회할 때

-> 수행빈도가 높고 OLTP 에선 사용 불가. **CPU와 메모리 사용률이 크게 증가!**

* Hash 조인 관련 힌트

- SWAP\_JOIN\_INPUTS(Table Name) -> Build Input으로 사용

- NO\_SWAP\_JOIN\_INPUTS(Table Name) -> Probe Input으로 사용

**4. 조인 순서의 중요성**

- NL 조인시에는 Driving 테이블의 인덱스 결과 값이 작아야하거나 쓰이는 컬럼이 인덱스에 모두 존재하면 됨

- Sort Merge 조인시 Disk sort가 필요한 경우 큰 테이블 Driving이 유리 (Disk I/O 감소)

PGA Sort Area안에 담길 경우 적은 테이블 Driving이 유리 (Join 회수 감소)

- Hash 조인은 Hash Area에 충분히 담길 정도로 적은 테이블이 Build Input으로 쓰여야 하며 Hash Chain이 작아야함

**5. Outer Join**

1) Outer NL Join

- (+)의 반대쪽이 드라이빙 테이블로 선택

- Leading 힌트로도 순서 변경 불가능 (논리적 모순)

2) Outer Sort Merge Join

- (+)의 반대 쪽이 드라이빙 테이블로 선택

- Leading 힌트로도 순서 변경 불가능 (논리적 모순)

3) Outer Hash Join

- (+)의 반대쪽이 드라이빙 테이블로 선택 -> 10g 부터는 Right Outer 해시 조인 가능

- SWAP\_JOIN\_INPUTS(Table Name)으로 순서 조정 가능

**6. 스칼라 서브쿼리를 이용한 조인**

- Scalar Sub Query 란 Select 절에 쓰이는 Sub Query

- 수행 회수 최소화를 위한 캐싱 기능을 제공한다

**7. 조인을 내포한 DML 튜닝**

- bypass\_ujvc 힌트를 이용 -> Updatable Join View로, 1:M 관계에서 M쪽만 변경

-> 수정 가능 뷰 체크를 생략

- DBMS는 확인 불가능 해도 사람에게 Check를 넘기는 힌트.

- 반드시 Update를 위해 참조하는 집합에 중복 레코드가 없을 때만 사용 필요

\* 11g 에서만 사용이 불가능하며 Merge Into Table1 Using Table2 형식으로 사용

**8. 고급 조인 테크닉**

**- 윈도우 함수**

|  |  |  |  |
| --- | --- | --- | --- |
| 함수 구분 | 함수 | 설명 | 비고 |
| 순위 | RANK | 순위 | 1,2,2,4,5 |
| DENSE\_RANK | 동일 순위를 1건으로 | 1,2,2,3,4 |
| ROW\_NUMBER | 동일 순위 미 인정 | 1,2,3,4,5 |
| 집계 | SUM, MAX, MIN, AVG, COUNT | 합계, 최대, 최소, 평균, 건수 |  |
| 행 순서 | FIRST\_VALUE, LAST\_VALUE | 파티션 내 첫 번쨰, 최종 값 |  |
| LAG, LEAD | 이전 레코드, 이후 레코드 값 |  |
| 비율 | CUME\_DIST | 파티션 내 데이터 값에 대한 누적 비율 | 결과 값 0~1 사이 |
| PERCENT\_RANK | 파티션 내의 row 개수에 대한 누적 비율 |
| RATIO\_TO\_REPORT | 파티션 내의 비율 |
| NTILE | 인자 값으로 전체 로우 개수를 나누고, 인자 값의 순서를 결정 | ntile(4) over(order by sal) sal 컬럼 순으로 정렬후 전체 로우를 4등분 후 1~4 값 설정 |

- 윈도우 함수 사용법

1) BETWEEN 사용 시

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ROWS | BETWEEN | UNBOUNDED PRECEDING (맨위) | AND | CURRENT ROW |
| 10 PRECEDING (10개전) | UNBOUNDED FOLLOWING |
| RANGE |
| CURRENT ROW (지금) | 10 FOLLOWING |

2)BETWEEN 미 사용 -> 현재 행 기준으로!

|  |  |
| --- | --- |
| ROWS | UNBOUNDED PRECEDING |
| 10 PRECEDING |
| CURRENT ROW |
| RANGE |
| UNBOUNDED FOLLOWING |
| 10 FOLLOWING |

ex) Select 일련번호, 측정값,

last\_value(상태코드) over (partition by 부서 order by 상태코드 rows between

unbounded preceding and current row) 상태코드

from 장비측정 order by 부서, 일련번호;

**계층 쿼리**

- START WITH : 시작 점 지정

- CONNECT BY ~ PRIOR ~ : 진행 경로 설정

- ORDER SIBLINGS BY : 같은 레벨별 정렬 기준

- LEVEL : 루트부터 각 레벨 별 1씩 증가

- CONNECT\_BY\_ISLEAF : LEAF = 1, NON LEAF = 0

- CONNECT\_BY\_ROOT : 루트 데이터 출력

- CONNECT\_BY\_PATH : 전체 경로 표시

\*CONNECT BY 절에서 Column <> 나온다면 연결된(하위혹은 상위) 데이터 비출력

\*WHERE 절에서 Column <> 나온다면 그 관련 데이터만 비출력

(2) 선분이력 끊기

- 월말 기준으로 선분을 끊는 경우

Select **greatest** (a.시작일자, b.시작일자) 시작일자, **least** (a.종료일자, b.종료일자) 종료일자

From 월도 a, 선분이력 b

Where b.시작일자 <= a.종료일자

and b.종료일자 >= a.시작일자; => greatest와 least 활용!

(3) 데이터 복제를 통한 소계 구하기

- dummy table (dual)를 통해서 Cartesian Product로 만듬

- Select b.no a.depno

From emp a, (Select rownum no from dual connect by level <= 2) b;

-> a테이블의 2배 만큼 출력 됨

=> Group By를 통해 소계를 만들 수 있음 -> ROLLUP 함수로 대체 가능!

**APPENDIX**

1. **집계 함수 (Aggregation Fuction)**

- NULL값은 집계 함수 연산 참여 X

- NULL + 숫자 = NULL

1. **Sub-Query**

- 연관 서브쿼리 : 서브쿼리에 메인쿼리 칼럼 보유. 메인쿼리가 먼저 수행 후 조건 확인

- 비연관 서브쿼리 : 서브쿼리에 메인쿼리 칼럼 미 보유. 서브쿼리 먼저 수행 후 메인에 결과 값을 제공.

- 단일 행 서브쿼리 : 서브쿼리 결과 값이 항상 1건 이하. 단일 행 비교 연산자와 함께 사용

- 다중 행 서브쿼리 : 서브쿼리 결과 값이 여러 건. 다중 행 비교 연산자와 함께 사용 ex) IN

- 다중 칼럼 서브쿼리 : 반환하는 칼럼이 2개 이상인 경우 -> SQL SERVER 불가!

1. **집합 연산자**

- UNION ALL : 합집합, 중복 허용

- UNION (합집합), INTERSECT(교집합), EXCEPT(=MINUS-오라클) 차집합 => 중복제거

1. **Join – Using절, On절**

- Natural Join : 두 테이블 중 칼럼 명 동일한 것을 활용. 같은 값만 출력. 테이블 접두사 X

- Using : 두 테이블 중 동일한 칼럼 명 기술, 테이블 접두사 X

- On : ANSI-SQL. 명칭이 상이한 칼럼 간 조인 연결고리 사용 가능. 테이블 접두사 O

1. **Outer Join – On절, Where절 & 전통적인 Join에서 흔한 실수**

- On절에서 조건을 기술 할 경우 조건에 맞지 않는 것도 출력. where 절에서는 출력 X

- Outer Join이며, 전통적인 방식의 쿼리일 때 Inner 테이블에 조건을 주고 싶을 경우 반드시 Inner 조건에 Outer 조건(+) 표기 필요.

**8. 고급 조인 테크닉**

(4) 상호배타적 관계적 조인

- 아크(한 테이블이 조인하는 테이블이 경우에 따라 다를 때)일 때

1) Outer Join를 이용한 외래키 분리방법

2) 외래키를 통합하는 방법 -> 이 경우도 Outer Join을 사용할 수 있는데, 이 때 DECODE함수를 통하여 하면 불필요한 조인 횟수가 감소한다.

(5) 최종 출력 건에 대해서만 조인하기

- 조인을 적은 출력 건수 (최종 출력 건수)로 마지막에 조인할 것

(6) 징검다리 테이블 조인을 이용한 튜닝

- 3개의 테이블이 존재하고 첫 번째 테이블의 출력 건은 적고 두 번째 테이블 출력 건은 급격히 많아지고 마지막 테이블의 출력 건은 적어질 때에 이용할 수 있는 방법으로 두 번째와 세 번째 테이블의 BRIDGE 테이블을 만들고 인덱스를 더해주어 인덱스를 통해서만 두 테이블이 조인을 하고 이를 통해 나온 값들의 ROWID를 통해서 각각의 테이블에 접근하여 원하는 값을 참조하는 방법

(7) 점이력 조회

- MIN / MAX 쿼리 활용하며 낮은 성능으로 어쩔 수 없는 경우 사용

- 인덱스 변경 시 대처 가능한 프로세스 확립 필요

- 스칼라 서브쿼리로 변경할 경우 테이블 1회 ACCESS 가능하지만 INDEX DESC방식으로 하며 인덱스 변경 시 잘못될 확률이 많으므로 추천 X

-> 가능한 분석함수 (ROW\_NUMBER)를 통하여 사용할 경우 훨씬 효율적이다.