**3과목**

* **데이터 저장 구조**

1. **블록**

* **데이터를 읽고 쓸 때의 논리적인 단위**

1. **익스텐트**

* **테이블스페이스로부터 공간을 할당하는 단위**
* **익스텐트 내 블록은 논리적으로 인접하다**
* **각 인스텐트는 논리적으로 인접하지 않다**

1. **세그먼트**

* **세그먼트는 익스텐트 단위로 여러 데이터 파일에 걸쳐 저장**

1. **테이블스페이스**

* **세그먼트를 담는 콘테이너**
* **Fast Commit**
* **변경된 내용이 메모리상의 버퍼 블록에만 기록되고 디스크에 기록되지 않았더라도 Redo로그를 믿고 빠르게 커밋을 완료하는 것**
* **Write Ahead Logging**
* **Dirty 버퍼를 디스크에 기록하기 전에 로그 파일에 기록하는 것**
* **Log Force at commit**
* **로그 버퍼를 늦어도 커밋 시점에는 로그 파일에 기록해야 한다**
* **SQL은 구조적(structured), 집합적(set-based), 선언적(declarative) 질의 언어**
* **절차적X**
* **버퍼 캐시 히트율(BCHR, Buffer Cache Hit Ratio)**
* **계산식 = (query + current - disk)/( query + current)**
* **SQL Server에서 힌트를 기술하는 방법**

1. **테이블 힌트**
2. **조인 힌트**
3. **쿼리 힌트**

* **SQL 트레이스 분석(523p 참고)**

1. **disk, query, current: ~에서 읽은 블록 수**
2. **cr: Consistent 모드 블록 일기**
3. **pr: 디스크 블록 읽기**
4. **pw: 디스크 블록 쓰기**
5. **time: 소요시간**

* **cr, pr, pw, time은 누적된 값으로 표시된다**
* **응답 시간 분석**

1. **Response Time = 서비스 시간(service time) + 대기 시간(wait time)**
2. **Response Time = CPU Time + Queue Time**

* **Index Range Scan**
* **인덱스 선두 칼럼이 조건절에 존재할 때 사용**
* **Index Full Scan**
* **검색을 위한 최적의 인덱스가 없을 때 사용**
* **인덱스 스캔 단계에서 대부분을 필터링하고 일부에 대해서만 테이블 엑세스할 경우 Table Full Scan보다 좋다**
* **Index Unique Scan**
* **Unique 인덱스를 = 조건으로 탐색하는 경우 사용**
* **Index Skip Scan**
* **조건절에 빠진 인덱스 선두 컬럼의 Distinct Value 개수가 적고 후행 칼럼의 Distinct Value 개수가 많을 때 사용**
* **Index Fast Full Scan**
* **인덱스에 포함된 컬럼만 조회할 때 사용**
* **비트맵 인덱스**
* **컬럼의 Distinct Value 개수가 적을 때 저장효율이 좋다**
* **B tree인덱스보다 적은 용량을 차지하여 대용량 테이블에 유용**
* **여러 비트맵 인덱스를 동시에 사용할 수 있다**
* **Null도 저장한다**
* **다양한 조건절이 사용되는 정형화되지 않은 임의 질의가 많은 환경에 적합**
* **DML부하가 심해 OLTP성 환경보다는 OLAP환경에 적합**
* **리버스 키 인덱스**
* **순차적으로 증가하는 값이 오른쪽 리프 블록에만 쌓이는 것을 방지**
* **범위 스캔 혹은 인덱스 사용이 불가능한 경우**

**1. 인덱스 선두 컬럼을 조건절에서 가공한 경우**

**2. 부정형 비교를 사용한 경우**

**Ex) where 직업 <> ‘학생’**

**3. is not null**

**4. is null => 다른 인덱스 컬럼에 is null이 아닌 조건식 혹은 not null 제약이 있으면 Index Range Scan 가능**

* **선택도**
* **출력될 것으로 예상되는 레코드 비율**
* **선택도 = 1/Distinct Value 개수**
* **카디널리티**
* **출력될 것으로 예상되는 결과 건수**
* **카디널리티 = 총 로우 수 \* 선택도 = 총 로우 수/ Distinct Value 개수**
* **서브쿼리 Unnesting**
* **중첩된 서브쿼리를 풀어내는 것**
* **서브쿼리가 1쪽 집합이거나 Unique 인덱스여야 한다**
* **NL세미조인**
* **NL조인과 동일하지만 Outer 테이블의 한 로우가 Inner 테이블의 한 로우와 조인에 성공하는 순간 진행을 멈추고 Outer 테이블의 다음 로우를 처리하는 방식**
* **isExists 조건일 때 자주 사용**
* **뷰 Merging이 불가능한 경우**
* **집합 연산자: union, union all, intersect, minus**
* **Connect by 절**
* **ROWNUM pseudo 칼럼**
* **Select-list에 집계 함수(avg, count, max, min, sum) 사용**
* **Group by가 쓰이면 뷰 Merging 자주 발생…**
* **분석 함수(WINDOW 함수)**
* **조건절 Pushing**
* **뷰를 참조하는 쿼리 블록의 조건절을 뷰 쿼리 블록 안으로 밀어 넣는 것**
* **Oracle에서 Direct Path Insert를 사용하는 방법**
* **Insert select 문장에 /\*+ append \*/ 힌트 사용**
* **병렬 모드로 insert**
* **Direct 옵션을 지정하고 SQL Loader로 데이터를 로드**
* **CTAS(Create Table AS SELECT)로 테이블 생성**
* **조인을 위한 데이터 분배 방식 힌트 사용법(pq\_distribute)**

**ex) pq\_distribute(inner, none, none): 양쪽 테이블이 이미 파티셔닝돼 있는 경우**

**ex) pq\_distribute(inner, partion, none): Inner가 이미 파티셔닝돼 있는 경우**

**ex) pq\_distribute(inner, none, partion): Outer가 이미 파티셔닝돼 있는 경우**

* **Snapshot too old 방지법**

**1. Undo 영역 크기 증가**

**2. 불필요한 커밋 수행 X**

**3. Fetch across commit 형태의 프로그램 작성 X**

**4. 트랜잭션이 몰리는 시간대에 오래 걸리는 쿼리 실행 X**

**5. 큰 테이블을 일정 범위로 나누고 단계적으로 실행할 수 있도록 코딩**

**6. 오래 걸리는 NL조인문을 변경**

**7. 소트 연산을 발생시킨다**

**8. 대량 업데이트 후 해당 테이블 혹은 인덱스를 Full Scan하도록 코딩**

* **계층형 질의 vs 집계 함수(rollup, cube, grouping sets)**
* **계층형 질의 루트 노드의 level은 1**
* **집계 함수 GROUPING 값은 1**
* **핵심문제 2과목 88, 106번 참고**