16장 인덱스

16.1 인덱스의 개요

16.1.1 인덱스란?

- 인덱스란 SQL 명령문의 처리 속도를 향상시키기 위해서 컬럼에 대해서 생성하는 오라클 객체이다.
- 인덱스의 장점
  - 검색 속도가 빨라진다.
  - 시스템에 부하를 줄여서 시스템의 전체 성능을 향상시킨다.
- 인덱스의 단점
  - 인덱스를 위한 추가 공간이 필요하다.
  - 인덱스를 생성하는 데 시간이 걸린다.
  - 데이터의 변경 작업(INSERT/UPDATE/DELETE)이 자주 일어날 때는 오히려 성능이 저하된다.

16.1.2 인덱스 정보 조회

- USER_COLUMNS와 USER_IND_COLUMNS 데이터 딕셔너리 뷰를 살펴보아야 한다.
- 인덱스는 기본 키나 유일 키와 같은 제약 조건을 지정하면 따로 생성하지 않더라도 자동으로 생성된다.

```
COLUMN TABLE_NAME FORMAT A15
COLUMN COLUMN_NAME FORMAT A15  -- 컬럼의 출력 형식 조정하기

SELECT INDEX_NAME, TABLE_NAME , COLUMN_NAME
FROM USER_IND_COLUMNS
WHERE TABLE_NAME IN('EMP', 'DEPT');
```

16.1.3 조회 속도 비교하기

- 인덱스가 조회 속도를 빠르게 해 준다는 것을 증명하기 위해서 기본 키나 유일 키로 지정하지 않는 컬럼인 사원 이름으로 검색해 본다. 아마도 시간이 어느 정도 소요될 것이다.
- 검색을 위해서 WHERE 절에 사용되는 컬럼인 사원 이름 컬럼을 인덱스로 생성한 후에 다시 한 번 사원 이름으로 검색해보면 검색시간이 현저하게 줄어드는 것을 확인할 수 있다.

[실습] 인덱스가 아닌 컬럼으로 검색하기

```
-- 사원 테이블 복사하기
DROP TABLE EMP01;
CREATE TABLE EMP01
AS
SELECT * FROM EMP;

SELECT TABLE_NAME, INDEX_NAME, COLUMN_NAME
FROM USER_IND_COLUMNS
WHERE TABLE_NAME IN('EMP', 'EMP01');
```

```
-- 인덱스가 아닌 컬럼으로 검색하기
INSERT INTO EMP01 SELECT * FROM EMP01;
--... 테이블 자체 복사를 여러 번 반복해서 상당히 많은 양의 행을 생성한다.
INSERT INTO EMP01 SELECT * FROM EMP01;
INSERT INTO EMP01(EMPNO, ENAME) VALUES(1111, 'SYJ');

SET TIMING ON
SELECT DISTINCT EMPNO, ENAME
FROM EMP01
WHERE ENAME='SYJ';
```

16.1.4 인덱스 생성 및 제거하기

■ 제약 조건에 의해 자동으로 생성되는 인덱스 외에 CREATE INDEX 명령어로 직접 인덱스를 생성할 수도 있다.

```
-- 형식
CREATE INDEX index_name
ON table_name (column_name);

DROP INDEX index_name;
```

[실습] 인덱스 설정하여 조회속도 비교하기

```
-- 인덱스 설정하기
CREATE INDEX IDX_EMP01_ENAME
ON EMP01(ENAME);

SELECT DISTINCT EMPNO, ENAME
FROM EMP01
WHERE ENAME='SYJ';

-- 인덱스 제거하기
SELECT INDEX_NAME, TABLE_NAME , COLUMN_NAME
FROM USER_IND_COLUMNS
WHERE TABLE_NAME IN('EMP01');

DROP INDEX IDX_EMP01_ENAME;

SELECT DISTINCT EMPNO, ENAME
FROM EMP01
WHERE ENAME='SYJ';
```

16.1.5 인덱스를 사용해야 하는 경우 판단하기

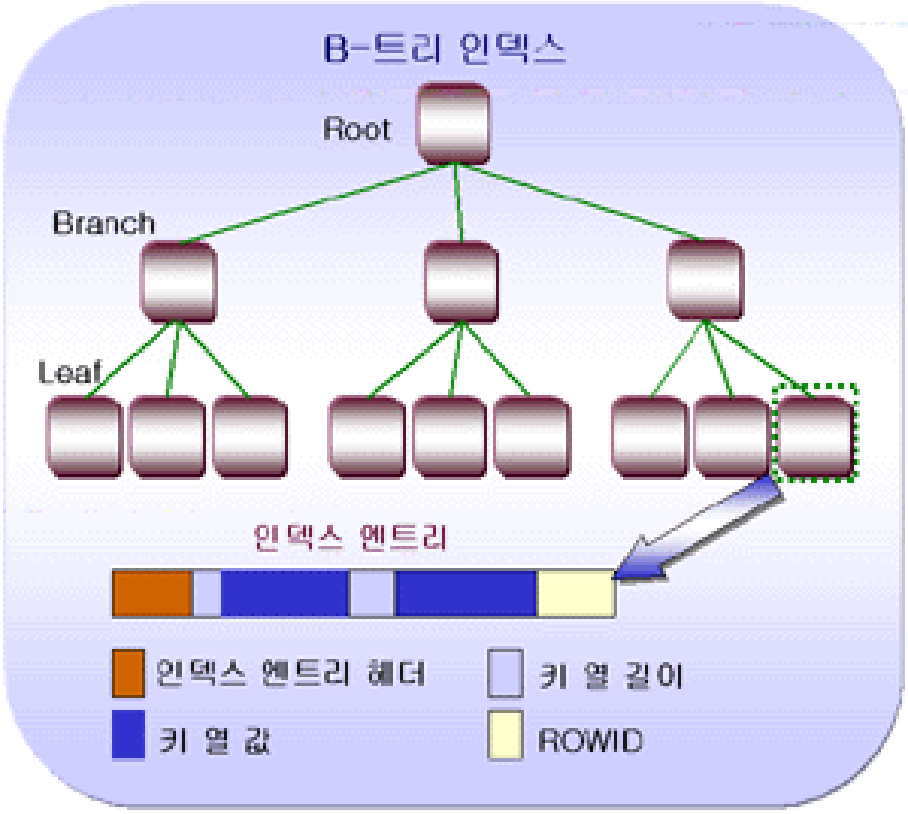
인덱스를 사용해야 하는 경우	인덱스를 사용하지 말아야 하는 경우
테이블에 행의 수가 많을 때	테이블에 행의 수가 적을 때
WHERE 문에 해당 컬럼이 많이 사용될 때	WHERE 문에 해당 컬럼이 자주 사용되지 않을 때
검색 결과가 전체 데이터의 2%~4% 정도 일 때	검색 결과가 전체 데이터의 10%~15% 이상 일 때
JOIN에 자주 사용되는 컬럼이나 NULL을 포함하는 컬럼이 많은 경우	테이블에 DML 작업이 많은 경우 즉, 입력 수정 삭제 등이 자주 일어 날 때

- 다음과 같은 조건에서 사원 테이블의 부서번호(DEPTNO)에 인덱스를 거는 것이 좋을까요?
  - 테이블에 전체 행의 수는 10000 건이다.
  - 위의 쿼리문을 전체 쿼리문들 중에서 95% 사용된다.
  - 쿼리문의 결과로 구해지는 행은 10건 정도이다.
- 조건을 위 표를 비추어보고 판단해 보면 부서번호(DEPTNO) 컬럼을 인덱스로 사용하기에 알맞다는 결론이 난다.

16.2 인덱스의 물리적인 구조와 재생성

16.2.1 B-트리 인덱스 구조

- 오라클에서의 인덱스의 내부 구조는 B-트리 형식으로 구성되어 있다.
- 뿌리(루트)를 근거로 아래로 나무뿌리 들이 뻗어 있는 모양을 하고 있다.



16.2.2 B-트리 인덱스의 추가, 삭제

- DML 작업 특히 DELETE 명령을 수행한 경우에는 해당 인덱스 엔트리가 논리적으로만 제거 되고 실제 인덱스 엔트리는 그냥 남아 있게 된다.
- 인덱스에 제거된 엔트리가 많아질 경우에는 제거된 인덱스들이 필요 없는 공간을 차지하고 있기 때문에 종종 인덱스를 재생성시켜야 한다.

```
-- 형식
ALTER INDEX index_name REBUILD;

-- 예
```

```
ALTER INDEX IDX_EMP01_ENAME REBUILD;
```

16.3 인덱스의 종류 살피기

16.3.1 고유와 비고유 인덱스

- 고유 인덱스(유일 인덱스라고도 부름)는 기본키나 유일키처럼 유일한 값을 갖는 컬럼에 대해서 생성하는 인덱스이다.
- 반면 비고유 인덱스는 중복된 데이터를 갖는 컬럼에 대해서 인덱스를 생성하는 경우를 말한다.
- 고유 인덱스를 설정하려면 UNIQUE 옵션을 추가해서 인덱스를 생성해야 한다.

```
-- 형식
CREATE UNIQUE INDEX index_name
ON table_name (column_name);
```

[실습] 고유 인덱스와 비고유 인덱스 정의하기

- 고유 인덱스와 고유 인덱스를 비교하기 위해서 중복된 데이터가 없는 컬럼(DEPTNO)과 있는 컬럼(LOC)으로 구성된 부서 테이블을 만들어 보겠다.

```
DROP TABLE DEPT01;
CREATE TABLE DEPT01
AS
SELECT * FROM DEPT WHERE 1=0;

INSERT INTO DEPT01 VALUES(10, '인사과', '서울');
INSERT INTO DEPT01 VALUES(20, '총무과', '대전');
INSERT INTO DEPT01 VALUES(30, '교육팀', '대전');
SELECT * FROM DEPT01;

CREATE UNIQUE INDEX IDX_DEPT01_DEPTNO
ON DEPT01(DEPTNO);

CREATE UNIQUE INDEX IDX_DEPT01_LOC
ON DEPT01(LOC);
-- 중복된 데이터를 갖는 컬럼을 인텍스로 지정하면 오류가 발생한다.

CREATE INDEX IDX_DEPT01_LOC
ON DEPT01(LOC);
```

16.3.2 결합 인덱스 정의하기

- 두 개 이상의 컬럼으로 인덱스를 구성하는 것을 결합 인덱스라고 한다.

```
CREATE INDEX IDX_DEPT01_COM
ON DEPT01(DEPTNO, DNAME);

SELECT INDEX_NAME, COLUMN_NAME
```

```
FROM USER_IND_COLUMNS
WHERE TABLE_NAME = 'DEPT01';
```

16.3.3 함수 기반 인덱스 정의하기

- 검색조건으로 WHERE SAL = 300이 아니라 WHERE SAL*12 = 3600와 같이 SELECT 문 WHERE 절에 산술 표현 또는 함수를 사용하는 경우가 있다.
- 이 경우 만약 SAL 컬럼에 인덱스가 걸려 있다면 인덱스를 타서 빠르리라 생각 할 수도 있지만 실상은 SAL 컬럼에 인덱스가 있어도 SAL*12는 인덱스를 타지 못한다.
- 인덱스 걸린 컬럼이 수식으로 정의 되어 있거나 SUBSTR 등의 함수를 사용해서 변형이 일어난 경우는 인덱스를 타지 못하기 때문이다.
- 이러한 수식으로 검색하는 경우가 많다면 아예 수식이나 함수를 적용하여 인덱스를 만들 수 있다. SAL*12로 인덱스를 만들어 놓으면 SAL*12가 검색 조건으로 사용될 시 해당 인덱스를 타게 된다.

```
SET TIMING ON
-- 인덱스 이전 검색하기
SELECT * FROM EMP01
WHERE SAL*12 = 3600;

CREATE INDEX IDX_EMP01_ANNSAL
ON EMP01(SAL*12);

SELECT INDEX_NAME, COLUMN_NAME
FROM USER_IND_COLUMNS
WHERE TABLE_NAME = 'EMP01';

-- 인덱스 이후 검색하기
SELECT * FROM EMP01
WHERE SAL*12 = 3600;
```