

인덱스

자바 강의실

목차

I. 인덱스 개념

II. 인덱스 관리

1. 인덱스 개념

- 인덱스를 왜 사용하는 것일까요?
 - 이에 대한 답은 “빠른 검색을 위해서 인덱스를 사용합니다.” 입니다.
 - 여러분이 테이블 생성 방법을 책에서 찾으려고 할 때 어떻게 합니까? 책 첫 페이지부터 한 장씩 넘겨가면서 테이블 생성 방법이 기술되어 있는지 일일이 살펴보는 사람은 드물 것입니다.
 - 일반적으로 책 맨 뒤에 있는 색인(인덱스, 찾아보기)에서 해당 단어(테이블)를 찾아 그 페이지로 이동합니다.
 - 이렇게 원하는 단어를 쉽게 찾는 방법으로 색인, 인덱스가 사용되는 것처럼 오라클의 인덱스 역시 원하는 데이터를 빨리 찾기 위해서 사용됩니다.
- 인덱스란 SQL 명령문의 처리 속도를 향상시키기 위해서 컬럼에 대해서 생성하는 오라클 객체입니다.

1. 인덱스 개념

- 하지만 인덱스는 장점만 있는 것이 아닙니다.
 - 오라클에서의 인덱스의 내부 구조는 B* 트리 형식으로 구성되어 있습니다.
 - 트리란 나무의 뿌리 모양을 생각해 보시면 쉽게 이해할 수 있습니다.
 - 뿌리(루트)를 근거로 아래로 나무뿌리 들이 뻗어 있는 모양을 하고 있습니다.
 - 컬럼에 인덱스를 설정하면 이를 위한 B* 트리도 생성되어야 하기 때문에 인덱스를 생성하기 위한 시간도 필요하고 인덱스를 위한 추가적인 공간이 필요하게 됩니다.
 - 인덱스가 생성된 후에 새로운 행을 추가하거나 삭제할 경우 인덱스로 사용된 컬럼 값도 함께 변경되는 경우가 발생합니다.
 - 인덱스로 사용된 컬럼 값이 변경되는 이를 위한 내부 구조(B* 트리) 역시 함께 수정 돼야 합니다.
 - 이 작업은 오라클 서버에 의해 자동으로 일어나는데 그렇기 때문에 인덱스가 없는 경우 보다 인덱스가 있는 경우에 DML 작업이 훨씬 무거워지게 됩니다.

1. 인덱스 개념

- 인덱스의 장점
 - 검색 속도가 빨라진다.
 - 시스템에 걸리는 부하를 줄여서 시스템 전체 성능을 향상시킨다.
- 인덱스의 단점
 - 인덱스를 위한 추가적인 공간이 필요하다.
 - 인덱스를 생성하는데 시간이 걸린다.
 - 데이터의 변경 작업(INSERT/UPDATE/DELETE)이 자주 일어날 경우에는 오히려 성능이 저하된다.

1. 인덱스 개념

- 책의 색인 란과 동일한 역할인 쿼리를 빠르게 수행하기 위한 용도로 사용되는 인덱스는 기본 키나 유일 키와 같은 제약 조건을 지정하면 따로 생성하지 않더라도 자동으로 생성해줍니다.
- 기본 키나 유일 키는 데이터 무결성을 확인하기 위해하기 위해서 수시로 데이터를 검색하기 때문에 빠른 조회를 목적으로 오라클에서 내부적으로 해당 컬럼에 인덱스를 자동으로 생성하는 것입니다.

1. 인덱스 개념

예: 인덱스 확인

질의 : USER_IND_COLUMNS 데이터 덱서너리 뷰로 인덱스의 생성 여부를 확인하기

```
SELECT INDEX_NAME, TABLE_NAME , COLUMN_NAME
```

```
FROM USER_IND_COLUMNS
```

```
WHERE TABLE_NAME IN ('EMPLOYEES', 'DEPARTMENTS') ;
```

1. 인덱스 개념

- 위 쿼리문의 결과 화면을 통해서 사용자가 인덱스를 생성하지 않았어도 오라클에서 기본 키나 유일 키에 대해서 자동으로 인덱스를 생성한다는 것을 확인할 수 있습니다.
- 인덱스 역시 테이블이나 뷰나 시퀀스와 같이 오라클 객체의 일종이고 모든 객체들은 이름이 있어야 합니다.
- 기본 키나 유일 키에 대한 인덱스는 오라클이 생성한 것이기에 인덱스의 이름 역시 오라클에서 자동 부여해 줍니다.
- 자동으로 생성되는 인덱스 이름은 제약 조건 (CONSTRAINT)명을 사용함을 확인할 수 있습니다.

1. 인덱스 개념

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

1. 인덱스 개념

예: 인덱스 없는 테이블 생성

질의 : 인덱스로 인해 검색시간이 현저하게 줄어드는 것을 증명하기 위한 실습을 위해서 사원 테이블을 복사해서 새로운 테이블을 생성하기

```
CREATE TABLE EMPLOYEES_COPY  
AS  
SELECT * FROM EMPLOYEES;
```

```
SELECT TABLE_NAME, INDEX_NAME, COLUMN_NAME  
FROM USER_IND_COLUMNS  
WHERE TABLE_NAME IN('EMPLOYEES', 'EMPLOYEES_COPY');
```

1. 인덱스 개념

예: 인덱스 아닌 컬럼으로 검색

질의 : 서브 쿼리문으로 INSERT 문을 여러 번 반복하고 검색을 위한 데이터 입력하기

```
INSERT INTO EMPLOYEES_COPY SELECT * FROM EMPLOYEES_COPY;
```

```
INSERT INTO EMPLOYEES_COPY  
(EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL, SALARY, HIRE_DATE, JOB_ID)  
VALUES (400, '홍이사', '홍이사', 'HONG', 20000, SYSDATE, '이사');
```

1. 인덱스 개념

예: 인덱스 아닌 컬럼으로 검색

질의 : 검색해서 시간확인

SET TIMING ON - 코멘트 모드에서만 가능

```
SELECT * FROM EMPLOYEES_COPY WHERE FIRST_NAME = '홍이사';
```

1. 인덱스 개념

- 제약 조건에 의해 자동으로 생성되는 인덱스 외에 CREATE INDEX 명령어로 직접 인덱스를 생성할 수도 있습니다.
- 다음은 인덱스를 생성하기 위한 기본 형식입니다.

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

- CREATE INDEX 다음에 인덱스 객체 이름을 지정합니다. 어떤 테이블의 어떤 컬럼에 인덱스를 설정할 것인지를 결정하기위해서 ON 절 다음에 테이블 이름과 컬럼 이름을 기술합니다.

1. 인덱스 개념

예: 인덱스 설정하기

질의 : 테이블 EMPLOYEES_COPY 의 컬럼 중에서 이름(FIRST_NAME)에 대해서 인덱스를 생성하기

```
CREATE INDEX IDX_EMPCOPY_FIRST_NAME  
ON EMPLOYEES_COPY(FIRST_NAME);
```

```
SELECT * FROM EMPLOYEES_COPY WHERE FIRST_NAME = '홍이사';
```

1. 인덱스 개념

- 이번에는 인덱스를 삭제해봅시다.
- 이를 위해서 오라클은 DROP INDEX 명령어를 제공합니다.

```
DROP INDEX index_name;
```

2. 인덱스 관리

- 인덱스가 검색을 위한 처리 속도만 향상시킨다고 했습니다.
- 하지만, 무조건 인덱스를 사용한다고 검색 속도가 향상되는 것은 아닙니다.
- 계획성 없이 너무 많은 인덱스를 지정하면 오히려 성능을 저하시킬 수도 있습니다.
- 언제 인덱스를 사용하는 것이 좋을까요?

2. 인덱스 관리

- 인덱스를 사용해야 하는 경우 판단하기

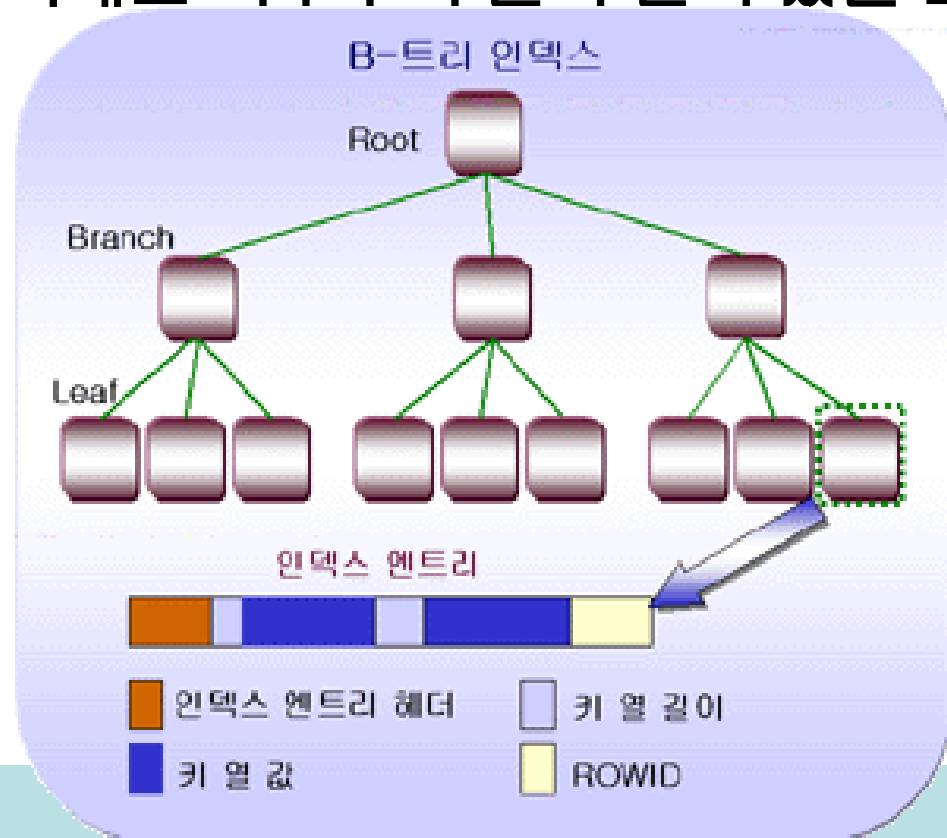
인덱스를 사용해야 하는 경우	인덱스를 사용하지 말아야 하는 경우
테이블에 행의 수가 많을 때	테이블에 행의 수가 적을 때
WHERE 문에 해당 컬럼이 많이 사용될 때	WHERE 문에 해당 컬럼이 자주 사용되지 않을 때
검색 결과가 전체 데이터의 2%~4% 정도 일 때	검색 결과가 전체 데이터의 10%~15% 이상 일 때
JOIN에 자주 사용되는 컬럼이나 NULL을 포함하는 컬럼이 많은 경우	테이블에 DML 작업이 많은 경우 즉, 입력 수정 삭제 등이 자주 일어 날 때

2. 인덱스 관리

- 다음과 같은 조건에서 사원 테이블의 부서 번호에 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
DEPARTMENT_ID 컬럼을 인덱스로 사용하기에 알맞다는 결론이 납니다.

2. 인덱스 관리

- 오라클에서의 인덱스의 내부 구조는 B-트리 형식으로 구성되어 있습니다.
- 트리란 나무의 뿌리 모양을 생각해 보시면 쉽게 이해할 수 있습니다.
- 뿌리(루트)를 근거로 아래로 나무뿌리 들이 뻗어 있는 모양을 하고 있습니다.



2. 인덱스 관리

- B-트리 형식의 인덱스를 B-트리 인덱스라고 합니다.
- B-트리 인덱스를 보다 정확하게 말하자면 인덱스 키에 대해 각각의 인덱스 엔트리로 구성되어 있어 있으며 인덱스 엔트리에 로우 아이디를 저장하고 있고 있습니다.
- 인덱스가 생성된 후에 새로운 행이 추가되거나 삭제될 수도 있고 인덱스로 사용된 컬럼 값이 변경될 수도 있습니다.
- 이럴 경우는 본 테이블에서 추가, 삭제, 갱신 작업이 일어날 때 해당 테이블에 걸린 인덱스의 내용도 함께 수정되어야 합니다.
- 이 작업은 오라클 서버에 의해 자동으로 일어나는데 삭제, 갱신 작업이 일어날 경우에 해당 인덱스 엔트리가 바로 인덱스로부터 제거 될까요? DELETE 문이 실행되면 논리적인 삭제 과정만 일어납니다.

2. 인덱스 관리

- DML 작업 특히 DELETE 명령을 수행한 경우에는 해당 인덱스 엔트리가 논리적으로만 제거 되고 실제 인덱스 엔트리는 그냥 남아 있게 됩니다.
- 인덱스에 제거된 엔트리가 많아질 경우에는 제거된 인덱스들이 필요 없는 공간을 차지하고 있기 때문에 종종 인덱스를 재생성시켜야 합니다.
- 다음은 인덱스를 재생성할 때 사용하는 기본 형식입니다.

```
ALTER INDEX index_name REBUILD;
```

- 예를 들어 IDX_EMP01_DEPTNO를 재생성해 봅시다.

```
ALTER INDEX IDX_EMPCOPY_FIRST_NAME REBUILD;
```

2. 인덱스 관리

- B-트리 형식의 인덱스를 B-트리 인덱스라고 합니다.
- B-트리 인덱스를 보다 정확하게 말하자면 인덱스 키에 대해 각각의 인덱스 엔트리로 구성되어 있어 있으며 인덱스 엔트리에 로우 아이디를 저장하고 있고 있습니다.
- 인덱스가 생성된 후에 새로운 행이 추가되거나 삭제될 수도 있고 인덱스로 사용된 컬럼 값이 변경될 수도 있습니다.
- 이럴 경우는 본 테이블에서 추가, 삭제, 갱신 작업이 일어날 때 해당 테이블에 걸린 인덱스의 내용도 함께 수정되어야 합니다.
- 이 작업은 오라클 서버에 의해 자동으로 일어나는데 삭제, 갱신 작업이 일어날 경우에 해당 인덱스 엔트리가 바로 인덱스로부터 제거 될까요? DELETE 문이 실행되면 논리적인 삭제 과정만 일어납니다.

2. 인덱스 관리

- 인덱스는 다음과 같이 구분할 수 있습니다.

1. 고유 인덱스(Unique Index)
2. 비 고유 인덱스(NonUnique Index)
3. 단일 인덱스(Single Index)
4. 결합 인덱스(Composite Index)
5. 함수 기반 인덱스(Function Based Index)

2. 인덱스 관리

- 비 고유 인덱스를 이해하려면 고유 인덱스와 비교해 보아야 합니다.
- 고유 인덱스(유일 인덱스라고도 부름)는 기본키나 유일키처럼 유일한 값을 갖는 컬럼에 대해서 생성하는 인덱스입니다.
- 반면 비고유 인덱스는 중복된 데이터를 갖는 컬럼에 대해서 인덱스를 생성하는 경우를 말합니다.
- 우리가 지금까지 사용한 인덱스는 비고유 인덱스입니다.
- 고유 인덱스를 설정하려면 UNIQUE 옵션을 추가해서 인덱스를 생성해야 합니다.

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


2. 인덱스 관리

- 지금까지 생성한 인덱스들처럼 한 개의 컬럼으로 구성된 인덱스는 단일 인덱스입니다.
- 두 개 이상의 컬럼으로 인덱스를 구성하는 것을 결합 인덱스라고 합니다.

2. 인덱스 관리

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