**### 인덱스**

테이블에 데이터가 계속 쌓이다 보면 테이블에서 특정 데이터 조회 작업이 느려져서

좀 더 빠르게 데이터를 조회,검색 하기 위해 도입된 개념이 인덱스

정의 : 테이블에 있는 특정 데이터를 빠르게 조회하는데 사용되는 데이터베이스 객체

테이블 -> **책**, 인덱스 -> **찾아보기** (테이블 로우의 주소 정보(rowid) / 빠름)

**## 방식**

테이블에 있는 데이터를 찾는 작업 -> 스캔(Scan)

테이블 전체에서 데이터를 찾는 작업 -> Full (테이블) 스캔

인덱스를 통해 데이터를 찾는 작업 -> 인덱스 스캔 (Full 스캔보다 빠름)

자주 사용되는 쿼리 분석 -> 가장 많이, 자주 사용되는 where 조건 컬럼을 인덱스로 생성

한 개 혹은 두 개 이상의 컬럼(결합 인덱스)으로 인덱스 생성 가능 -> 1개 인덱스에 최대 32개 컬럼

일반적으로 테이블 전체 ROW 수의 15% 이하 데이터 조회 시 인덱스 사용

인덱스 종류 : B-Tree 인덱스 (대표)

**## 일반 (B-Tree) 인덱스 종류**

컬럼 수에 따른 분류 -> 단일 인덱스 / 결합 인덱스

컬럼 값에 따른 분류 -> 유일(Unique) 인덱스 / 비유일(Non-Unique) 인덱스

* 기본키를 만들면 자동으로 유일 인덱스가 만들어진다. 유일 인덱스 때문에 중복값이 입력이 안되는 것이다.

INSERT 시에는 불리 -> 대량 데이터 INSERT 시, 인덱스 생성도 오래 걸림

* Table, insert 각각에 다 들어가야하기 때문에 시간 소요

**문장형식**

**CREATE [UNIQUE] INDEX** 인덱스명 **ON** 테이블명 (컬럼1, 컬럼2, …);

**인덱스 삭제**

**DROP INDEX** 인덱스명;

**인덱스 사용**

WHERE 절에서 인덱스 컬럼을 조회조건으로 사용

인덱스가 있어도 인덱스를 사용하지 못하는 경우

-> WHERE 절에서 인덱스 컬럼을 가공

WHERE SUBSTR(이름,1,1) = '홍' -> WHERE 이름 LIKE '홍%‘

풀스캔으로 가공된다.

WHERE TO\_CHAR(**입사일**, 'yyyymmdd') = ‘20200120'

-> WHERE **입사일** >= TO\_DATE(:입력일, 'yyyymmdd‘)

AND **입사일** < TO\_DATE(:입력일 , 'yyyymmdd‘) + 1

WHERE **이름** || 직급 = '홍길동부장'

WHERE **이름** = '홍길동'

AND 직급 = '부장

* 인덱스가 Where절을 사용할 때 가공하지 마라!

**## 인덱스 테스트용 테이블 생성 – orauser 사용자로 생성**

**CREATE TABLE** **INDEX\_TEST** **AS**

SELECT \*

ALL\_OBJECTS , ## 모든 객체 정보

( SELECT \* ## Cartesian Product (크로스 조인)

FROM DUAL ## 서브쿼리 & where가 생략된 join

CONNECT BY LEVEL <= 100); ## 오라클에만 있고 어려운 개념이라 생략

**## INDEX\_TEST 테이블 통계정보 생성**

EXEC **DBMS\_STATS.GATHER\_TABLE\_STATS** ('ORAUSER', 'INDEX\_TEST');

이런게 있다라는 것만 알아두면 됨.

## INDEX\_TEST 테이블에서 EMPLOYEES 조회

SELECT COUNT(\*)

FROM **INDEX\_TEST**

WHERE OBJECT\_NAME = **'EMPLOYEES';**

SELECT COUNT(\*)

FROM **INDEX\_TEST**

WHERE OBJECT\_NAME = **'LOCATIONS';**

## INDEX\_TEST 테이블 인덱스 생성

**CREATE INDEX** INDEX\_TEST\_IDX1 **ON** INDEX\_TEST ( OBJECT\_NAME)

INDEX\_TEST테이블에 OBJECT\_NAME 컬럼으로 INDEX\_TEST\_IDX1인덱스를 만들어라

위에 **EMPLOYEES LOCATIONS을** 인덱스 적용한 후에 사용하면 검색된 속도가 굉장히 빠르다.

## 아래 쿼리 작성 후 F10키 클릭

SELECT COUNT(\*) FROM INDEX\_TEST WHERE OBJECT\_NAME = 'EMPLOYEES';



## 인덱스 컬럼 가공 시, 인덱스 스캔 못함

SELECT COUNT(\*)

FROM INDEX\_TEST

WHERE **SUBSTR**(OBJECT\_NAME,1,9) = 'EMPLOYEES‘

AND OBJECT\_TYPE = 'TABLE';

INDEX를 사용하지 못하고 FULL SCAN을 사용하지 못하게 됨

**## 인덱스 삭제**

**DROP INDEX** INDEX\_TEST\_IDX1;

SELECT COUNT(\*)

FROM INDEX\_TEST

WHERE OBJECT\_NAME = 'EMPLOYEES';

## 효율적인 SQL문 작성을 위한 팁

효율적인 SQL문 이란?

-> 실행 속도가 빠른 쿼리

꼭 필요한 데이터만 처리하자

전체 테이블 데이터 양 중 15% 이하를 조회할 경우는 인덱스 활용

**###시노님**

테이블, 뷰 등 데이터베이스 객체에 대한 대체 이름, 동의어 역할을 하는 데이터베이스 객체

시노님을 사용하는 이유 ?

다른 사용자에게 자신의 객체에 대한 권한 부여 시, 소유자명 없이 사용 가능

(비교대상 : HR)

**## 시노님의 종류**

- PRIVATE SYNONYM : 자신만 사용 가능

- PUBLIC SYNONYM : 모든 사용자가 볼 수 있음

**## 구문형식**

**- 생성/수정**

**CREATE OR REPLACE [PUBLIC] SYNONYM** 시노님명 **FOR** 참조OWNER.참조객체

**- 삭제**

**DROP [PUBLIC] SYNONYM** 시노님명;

**시노님 생성 필요 조건**

- CREATE SYNONYM 권한 : PRIVATE 시노님 생성 권한

- CREATE ANY SYNONYM 권한 : 다른 사용자 스키마 상에서 PRIVATE 시노님 생성 권한

- CREATE PUBLIC SYNONYM 권한 : PUBLIC 시노님 생성 권한

ORAUSER 로 로그인 후 HR2에게 시노님 생성 권한 부여

GRANT **CREATE SYNONYM** TO HR2;

HR2 로 로그인 후 emp\_dept\_v2 시노님 생성

CREATE SYNONYM **emp\_dept\_v2** FOR **HR.emp\_dept\_v2;**

emp\_dept\_v2 시노님 조회 SELECT \* FROM emp\_dept\_v2;

**###시퀀스**

- 유일한 순번(정수형)을 생성하는 데이터베이스 객체 (새로운 사원 들어올 경우)

- 주로 테이블의 기본 키(NUMBER)형 컬럼에 값을 입력 시 사용

employees 테이블의 employee\_id, departments 테이블의 department\_id

- 시퀀스 생성 필요조건

CREATE SEQUENCE 권한 필요

· 시퀀스 생성 구문

**CREATE SEQUENCE** 시퀀스명

**INCREMENT BY** 증감숫자

**START WITH** 시작숫자 ## 생략하면 1부터 시작

NOMINVALUE | **MINVALUE** 최솟값 ## 디폴트 1

NOMAXVALUE | **MAXVALUE** 최댓값 ## 999999999999….

**NOCYCLE** | CYCLE ## : 최대값 도달

시 다시 처음부터 시작, NOCYCLE이 디폴트

**NOCACHE** | CACHE; ## 좀 더 빠른 처리를 위해 메모리

에 미리 할당해 보관하는 시퀀스 값. 디폴트는 20

·시퀀스 수정

ALTER SEQUENCE 시퀀스명 … ·

시퀀스 삭제

DROP SEQUENCE 시퀀스명;

- 시퀀스명.NEXTVAL : 다음 순번을 가져옴

- 시퀀스명.CURRVAL : 현재 순번을 가져옴

- emp\_seq 시퀀스 생성

CREATE SEQUENCE emp\_seq; ## default 값이 적용되서 1에서 시작에서 1

씩 증가하는 시퀀스

SELECT sequence\_name,

min\_value,

max\_value,

increment\_by,

cache\_size,

last\_number,

cycle\_flag

FROM USER\_SEQUENCES;

해당 시퀀스 관련 정보 확인가능

· emp 테이블에 데이터

INSERT INSERT INTO emp

SELECT emp\_seq.NEXTVAL,

first\_name || ' ' || a.last\_name,

salary, hire\_date

FROM employees

WHERE department\_id = 90;

SELECT \*

FROM emp;

employee 테이블의 department\_id=90인 것을 emp 테이블에 넣는데 emp\_seq라는 시퀀스를 순번대로 넣겠다는 의미 (결과값 3개)

INSERT INSERT INTO emp

SELECT emp\_seq.NEXTVAL,

first\_name || ' ' || a.last\_name,

salary, hire\_date

FROM employees

WHERE department\_id = 60;

SELECT \*

FROM emp;

employee 테이블의 department\_id=60인 것을 emp 테이블에 넣는데 emp\_seq라는 시퀀스를 순번대로 넣겠다는 의미 (결과값 5개)

SELECT **emp\_seq.CURRVAL**

FROM dual;

현재순번은 (3+5 =8)이 결과값으로 나온다.