

## 8-1. 인덱스, 시노님, 시퀀스

홍형경

[chariehong@gmail.com](mailto:chariehong@gmail.com)

2020.11

# 1. 인덱스 (Index)

- 테이블에 데이터가 계속 쌓이다 보면 테이블에서 특정 데이터 조회 작업이 느려 짐
- 테이블 로우 수 증가에 따라 특정 데이터를 검색, 추출이 오래 걸림
- 좀 더 빠르게 데이터를 조회,검색 하기 위해 도입된 개념이 인덱스

# 1. 인덱스 (Index)

- 테이블에 있는 특정 데이터를 빠르게 조회하는데 사용되는 데이터베이스 객체
- 테이블 → 책, 인덱스 → 찾아보기
- 찾아보기를 통해 특정 키워드를 찾는게 훨씬 빠름
- 인덱스에는 인덱스 컬럼 값과 해당 값이 있는 테이블 로우의 주소 정보(rowid)가 있음  
→ 특정 로우를 빠르게 찾을 수 있음
- 인덱스 값은 정렬되어 들어가 있어 검색이 빠름

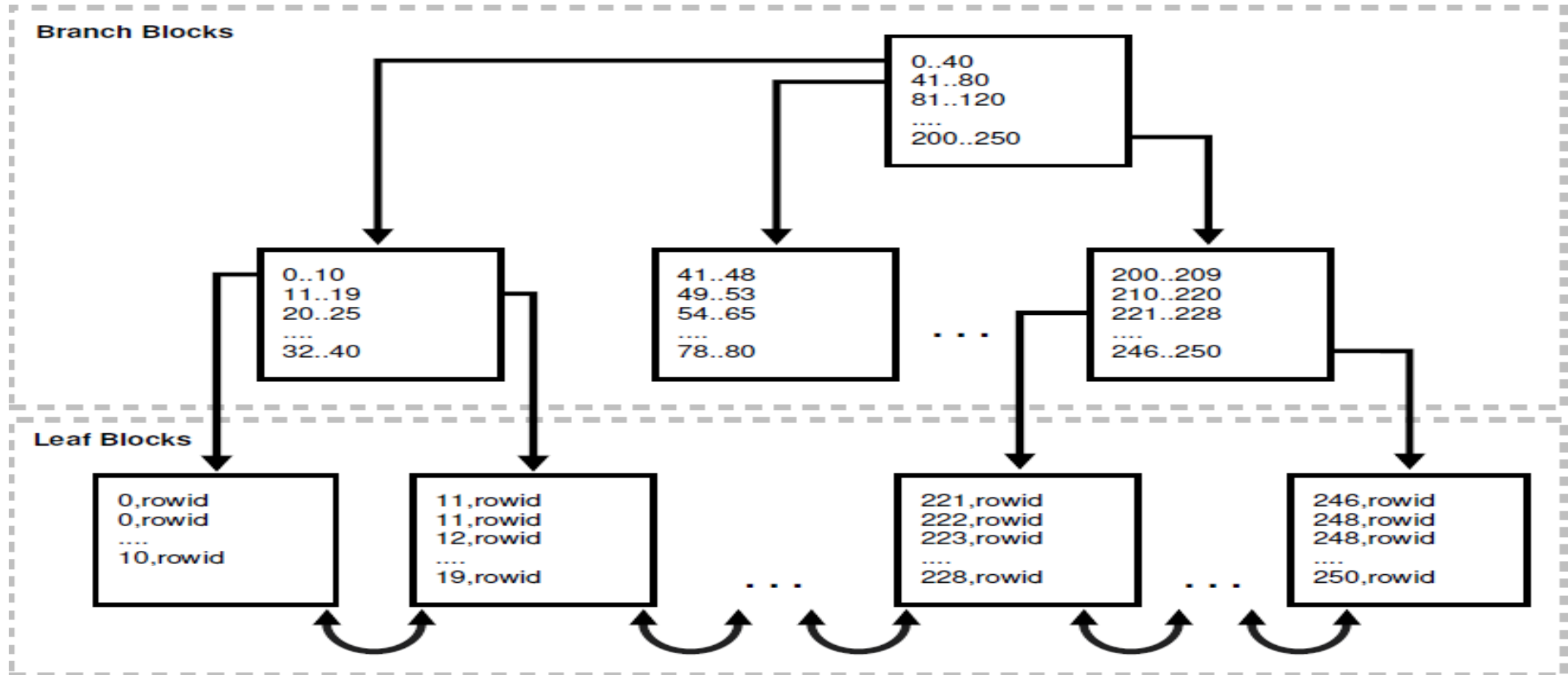
# 1. 인덱스 (Index)

- 테이블에 있는 데이터를 찾는 작업 ➔ 스캔(Scan)
- 테이블 전체에서 데이터를 찾는 작업 ➔ Full (테이블) 스캔
- 인덱스를 통해 데이터를 찾는 작업 ➔ 인덱스 스캔
- 대부분의 경우, 인덱스 스캔이 full 스캔보다 빠름

# 1. 인덱스 (Index)

- 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 ➔ 한 인덱스에 최대 32개 컬럼
- 일반적으로 테이블 전체 ROW 수의 15% 이하 데이터 조회 시 인덱스 사용
- 인덱스 종류 : B-Tree 인덱스 / 비트맵 인덱스 / 파티션 인덱스 / 함수기반 인덱스 / 도메인 인덱스
- 일반적으로 인덱스라고 하면 B(balanced)-Tree 인덱스를 뜻함

# 1. 인덱스 (Index) – B tree Index 구조



# 1. 인덱스 (Index)

- 일반 (B-Tree) 인덱스 종류
  - 컬럼 수에 따른 분류 → 단일 인덱스 / 결합 인덱스
  - 컬럼 값에 따른 분류 → 유일(Unique) 인덱스 / 비유일(Non-Unique) 인덱스
- 인덱스는 테이블의 특정 ROW를 빨리 찾을 때 유리
- 인덱스는 인덱스 컬럼 데이터, 인덱스가 가리키는 테이블 ROW의 주소 저장
  - 테이블에 INSERT 시, 인덱스 데이터도 자동 입력됨
  - 테이블 ROW 수 증가 → 인덱스 용량 증가
  - INSERT 시에는 불리 → 대량 데이터 INSERT 시, 인덱스 생성도 오래 걸림

# 1. 인덱스 (Index)

- 인덱스 생성

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

- 테이블에 주요 키(PK)를 생성하면 자동으로 유일 인덱스까지 생성됨

- 인덱스 삭제

**DROP INDEX** 인덱스명;

- 인덱스 사용

**WHERE** 절에서 인덱스 컬럼을 조회조건으로 사용



# 1. 인덱스 (Index)

- 인덱스가 있어도 인덱스를 사용하지 못하는 경우  
→ WHERE 절에서 인덱스 컬럼을 가공
- WHERE SUBSTR(이름,1,1) = '홍' → WHERE 이름 LIKE '홍%'
- 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'

# 1. 인덱스 실습

- 인덱스 테스트용 테이블 생성 – orauser 사용자로 생성

```
CREATE TABLE INDEX_TEST AS
```

```
SELECT *
```

```
FROM ALL_OBJECTS ,
```

```
( SELECT *
```

```
FROM DUAL
```

```
CONNECT BY LEVEL <= 100);
```

- ALL\_OBJECTS : 모든 객체 정보, 약 56,000건
- CONNECT BY : 계층형 쿼리,  $56,000\text{건} * 100 = 560\text{만건}$  데이터 생성

# 1. 인덱스 실습

- INDEX\_TEST 테이블 통계정보 생성

```
EXEC DBMS_STATS.GATHER_TABLE_STATS ('ORAUSER', 'INDEX_TEST');
```

- INDEX\_TEST 테이블 조회

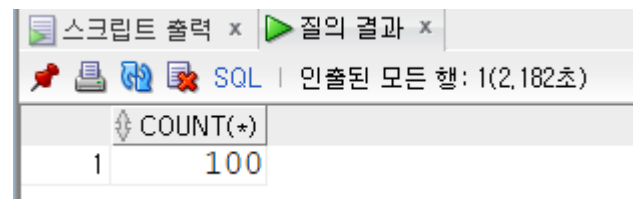
```
SELECT COUNT(*)  
FROM INDEX_TEST;
```

	COUNT(*)
1	68337

# 1. 인덱스 실습

- INDEX\_TEST 테이블에서 EMPLOYEES 조회

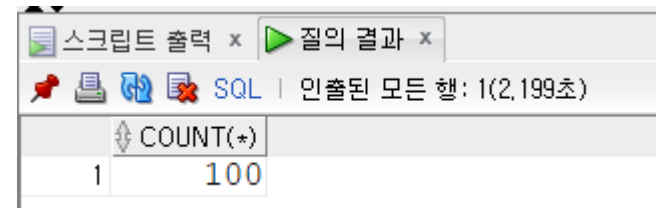
```
SELECT COUNT(*)  
FROM INDEX_TEST  
WHERE OBJECT_NAME = 'EMPLOYEES';
```



A screenshot of a SQL query execution window. The window has two tabs: '스크립트 출력 x' and '질의 결과 x'. The '질의 결과' tab is active, showing a table with one row and one column. The column is labeled 'COUNT(\*)' and the row contains the value '100'. The status bar at the bottom indicates '인출된 모든 행: 1(2,182초)'.

COUNT(*)
100

```
SELECT COUNT(*)  
FROM INDEX_TEST  
WHERE OBJECT_NAME = 'LOCATIONS';
```



A screenshot of a SQL query execution window. The window has two tabs: '스크립트 출력 x' and '질의 결과 x'. The '질의 결과' tab is active, showing a table with one row and one column. The column is labeled 'COUNT(\*)' and the row contains the value '100'. The status bar at the bottom indicates '인출된 모든 행: 1(2,199초)'.

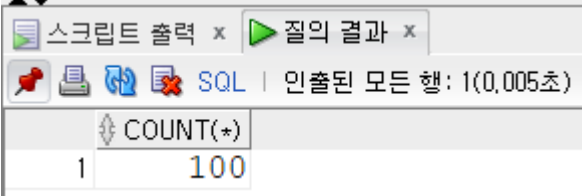
COUNT(*)
100

# 1. 인덱스 실습

- INDEX\_TEST 테이블 인덱스 생성

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

```
SELECT COUNT(*)  
FROM INDEX_TEST  
WHERE OBJECT_NAME = 'EMPLOYEES';
```

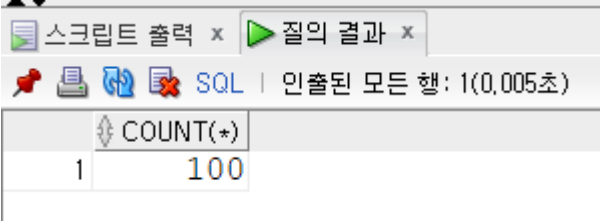


스크립트 출력 x | 질의 결과 x

SQL | 인출된 모든 행: 1(0,005초)

	COUNT(+)
1	100

```
SELECT COUNT(*)  
FROM INDEX_TEST  
WHERE OBJECT_NAME = 'LOCATIONS';
```



스크립트 출력 x | 질의 결과 x

SQL | 인출된 모든 행: 1(0,005초)

	COUNT(+)
1	100

# 1. 인덱스 실습

. 아래 쿼리 작성 후 F10키 클릭

```
SELECT COUNT(*)  
FROM INDEX_TEST  
WHERE OBJECT_NAME = 'EMPLOYEES';
```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	4
SORT		AGGREGATE	1	
INDEX	INDEX_TEST_IDX1	RANGE SCAN	120	4
Access Predicates	OBJECT_NAME='EMPLOYEES'			
Other XML	{info}			
info type="db_version"	18.0.0.0			
info type="parse_schema"	"ORAUSER"			
info type="plan_hash_full"	1629682494			
info type="plan_hash"	1694202047			
info type="plan_hash_2"	1629682494			
{hint}	INDEX(@"SEL\$1" "INDEX_TEST"@"SEL\$1" ("INDEX_TEST","OBJECT_NAME")) OUTLINE_LEAF(@"SEL\$1") ALL_ROWS DB_VERSION('18.1.0') OPTIMIZER_FEATURES_ENABLE('18.1.0') IGNORE_OPTIM_EMBEDDED_HINTS			

# 1. 인덱스 실습

- 인덱스 컬럼 가공 시, 인덱스 스캔 못함

```
SELECT COUNT(*)  
FROM INDEX_TEST  
WHERE SUBSTR(OBJECT_NAME,1,9) = 'EMPLOYEES'  
AND OBJECT_TYPE = 'TABLE';
```

스크립트 출력 x   계획 설명 x   질의 결과 x	
SQL   인출된 모든 행: 1(2.19초)	
COUNT(*)	
1	100

SQL   0.012초				
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	38335
SORT		AGGREGATE	1	
TABLE ACCESS	INDEX_TEST	FULL	2010	38335
Filter Predicates				
AND				
OBJECT_TYPE='TABLE'				
SUBSTR(OBJECT_NAME,1,9)='EMPLOYEES'				
Other XML				
{info}				
info type="db_version"				
18.0.0.0				
info type="parse_schema"				
"ORAUSER"				
info type="plan_hash_full"				
3731203167				
info type="plan_hash"				
4146568723				
info type="plan_hash_2"				
3731203167				
{hint}				
FULL(@"SEL\$1" "INDEX_TEST"@"SEL\$1")				
OUTLINE_LEAF(@"SEL\$1")				
all rows				

# 1. 인덱스 실습

## . 인덱스 삭제

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

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

스크립트 출력 x

질의 결과 x

📌

📄

🔗

❌

SQL | 인출된 모든 행: 1(2,181초)

	COUNT(*)
1	100

SQL   0.01초				
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	38292
SORT		AGGREGATE	1	
TABLE ACCESS	INDEX_TEST	FULL	120	38292
Filter Predicates	OBJECT_NAME='EMPLOYEES'			
Other XML	{info}			
info type="db_version"	18.0.0.0			
info type="parse_schema"	"ORAUSER"			
info type="plan_hash_full"	3731203167			
info type="plan_hash"	4146568723			
info type="plan_hash_2"	3731203167			
{hint}	FULL(@"SEL\$1" "INDEX_TEST"@"SEL\$1") OUTLINE_LEAF(@"SEL\$1") ALL_ROWS DB_VERSION('18,1,0') OPTIMIZER_FEATURES_ENABLE('18,1,0') IGNORE_OPTIM_EMBEDDED_HINTS			



## 2. 효율적인 SQL문 작성을 위한 팁

- 효율적인 SQL문이란?  
→ 실행 속도가 빠른 쿼리
- 시간이 지날 수록 데이터 양이 증가되어 쿼리가 느려짐
- 데이터가 많은 테이블과 조인 시에도 쿼리는 느려짐
- 느린 쿼리를 빠르게 만드는 작업 → SQL 튜닝(tuning)
- SQL 튜닝은 전문적인 튜너가 수행  
→ 전문적 튜닝 지식과 경험이 없이는 수행하기가 매우 어려움

## 2. 효율적인 SQL문 작성을 위한 팁

- 꼭 필요한 데이터만 처리하자
  - `SELECT *` 보다는 사용할 컬럼을 일일이 명시
  - 불필요한 조인 제거
- 전체 테이블 데이터 양 중 15% 이하를 조회할 경우는 인덱스 활용
- 테이블 통계 정보 갱신
  - 통계 정보란 해당 테이블에 있는 데이터의 메타 정보를 말함  
( 전체 로우 수, 컬럼 당 `distinct` 값 등 )
  - 통계 정보가 최신 상태여야 오라클 엔진이 정확한 쿼리 작성 계획을 세움
  - 통계 정보는 데이터베이스 관리자에게 갱신 요청

### 3. 시노님(Synonym)

- 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
- 시노님의 종류
  - PRIVATE SYNONYM : 자신만 사용 가능
  - PUBLIC SYNONYM : 모든 사용자가 볼 수 있음

### 3. 시노님(Synonym)

- 시노님 생성/수정 구문

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

- 시노님 삭제 구문

**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

### 3. 시노님 실습

- ORAUSER 로 로그인 후 HR2에게 시노님 생성 권한 부여

```
GRANT CREATE SYNONYM TO HR2;
```

Grant을 (를) 성공했습니다.

- HR2 로 로그인 후 emp\_dept\_v2 시노님 생성

```
CREATE SYNONYM emp_dept_v2 FOR HR.emp_dept_v2;
```

Synonym EMP\_DEPT\_V2이 (가) 생성되었습니다.

- emp\_dept\_v2 시노님 조회

```
SELECT *  
FROM emp_dept_v2;
```

EMPLOYEE_ID	EMP_NAMES	DEPARTMENT_ID	DEPARTMENT_NAME
1	100 Steven King	90	Executive
2	101 Neena Kochhar	90	Executive
3	102 Lex De Haan	90	Executive
4	103 Alexander Hunold	60	IT
5	104 Bruce Ernst	60	IT
6	105 David Austin	60	IT
7	106 Valli Pataballa	60	IT
8	107 Diana Lorentz	60	IT
9	108 Nancy Greenberg	100	Finance
10	109 Daniel Faviet	100	Finance
11	110 John Chen	100	Finance
12	111 Ismael Sciarra	100	Finance
13	112 Jose Manuel Urman	100	Finance
14	113 Luis Popp	100	Finance
15	114 Den Raphaely	30	Purchasing
16	115 Alexander Khoo	30	Purchasing
17	116 Shelli Baida	30	Purchasing
18	117 Sigal Tobias	30	Purchasing
19	118 Guy Himuro	30	Purchasing
20	119 Karen Colmenares	30	Purchasing

### 3. 시노님 실습

```
SELECT OWNER, OBJECT_NAME, OBJECT_ID, OBJECT_TYPE
FROM ALL_OBJECTS
WHERE OBJECT_NAME IN ( 'ALL_OBJECTS', 'ALL_TABLES', 'USER_TABLES');
```

- ALL\_, USER\_ 데이터 디렉터리를 HR 유저로 조회할 수 있는 것은  
PUBLIC 시노님이기 때문

	OWNER	OBJECT_NAME	OBJECT_ID	OBJECT_TYPE
1	SYS	USER_TABLES	4536	VIEW
2	SYS	ALL_TABLES	4543	VIEW
3	SYS	ALL_OBJECTS	4581	VIEW
4	PUBLIC	USER_TABLES	4537	SYNONYM
5	PUBLIC	ALL_TABLES	4544	SYNONYM
6	PUBLIC	ALL_OBJECTS	4583	SYNONYM

## 4. 시퀀스(Sequence)

- 유일한 순번(정수형)을 생성하는 데이터베이스 객체
- 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

## 4. 시퀀스(Sequence)

## · 시퀀스 생성 구문

## CREATE SEQUENCE 시퀀스명

## INCREMENT BY 증감숫자

## START WITH 시작숫자

NOMINVALUE | MINVALUE 최소값

NOMAXVALUE | MAXVALUE 최댓값

**NOCYCLE** | **CYCLE**

# NOCACHE | CACHE;

- INCREMENTY BY : 양수, 음수도 가능(0은 안됨), 디폴트 1
  - START WITH : 시작숫자, 디폴트는 ...  
증가시 MINVALUE, 감소시 MAXVALUE
  - MINVALUE : 최솟값, 디폴트 1
  - MAXVALUE : 최댓값, 디폴트 9999999999999999999999999999999
  - CYCLE : 최대값 도달 시 다시 처음부터 시작,  
NOCYCLE이 디폴트
  - CACHE : 좀 더 빠른 처리를 위해 메모리에 미리 할당해  
보관하는 시퀀스 값. 디폴트는 20
- ➔ 메모리 상주 값이므로 메모리 오류 발생 시, 시퀀스 값이  
연속해 이어지지 않는 경우가 많음



## 4. 시퀀스(Sequence)

- 시퀀스 수정

**ALTER SEQUENCE** 시퀀스명

...

- 시퀀스 삭제

**DROP SEQUENCE** 시퀀스명;

## 4. 시퀀스(Sequence)

- 시퀀스 참조 (의사컬럼)

- 시퀀스명.NEXTVAL : 다음 순번을 가져옴
- 시퀀스명.CURRVAL : 현재 순번을 가져옴
- 반드시 NEXTVAL을 사용한 다음에야 CURRVAL을 조회가 가능함

## 4. 시퀀스 실습

- emp 테이블 데이터 삭제

```
TRUNCATE TABLE emp;
```

- emp\_seq 시퀀스 생성

```
CREATE SEQUENCE emp_seq;
```

---

Sequence EMP\_SEQ이 (가) 생성되었습니다.

## 4. 시퀀스 실습

## · emp\_seq 조회

[illegible]

## 4. 시퀀스 실습

- emp\_seq 조회 (user\_sequences)

```
SELECT sequence_name,
```

**min\_value,**

**max\_value,**

**increment\_by,**

**cache\_size,**

**last\_number,**

## cycle\_flag

FROM USER\_SEQUENCES;

[illegible]

## 4. 시퀀스 실습

- 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;

EMP_NO	EMP_NAME	SALARY	HIRE_DATE
11	Steven King	24000	2003-06-17 00:00:00
22	Neena Kochhar	17000	2005-09-21 00:00:00
33	Lex De Haan	17000	2001-01-13 00:00:00

## 4. 시퀀스 실습

```
SELECT emp_seq.CURRVAL
FROM dual;
```

	CURVAL	
1	3	

```
SELECT sequence_name,  
       min_value,  
       max_value,  
       increment_by,  
       cache_size,  
       last_number,  
       cycle_flag
```

[illegible]

**FROM user\_sequences**  
**WHERE sequence\_name = 'EMP\_SEQ';**

## 4. 시퀀스 실습

- emp 테이블에 데이터 INSERT

```
INSERT INTO emp
SELECT emp_seq.NEXTVAL,
       first_name || ' ' || a.last_name,
       salary, hire_date
FROM employees
WHERE department_id = 60;

COMMIT;


SELECT *
FROM emp;
```

	EMP_NO	EMP_NAME	SALARY	HIRE_DATE
1	1	Steven King	24000	2003-06-17 00:00:00
2	2	Neena Kochhar	17000	2005-09-21 00:00:00
3	3	Lex De Haan	17000	2001-01-13 00:00:00
4	4	Alexander Hunold	9000	2006-01-03 00:00:00
5	5	Bruce Ernst	6000	2007-05-21 00:00:00
6	6	David Austin	4800	2005-06-25 00:00:00
7	7	Valli Pataballa	4800	2006-02-05 00:00:00
8	8	Diana Lorentz	4200	2007-02-07 00:00:00



## 4. 시퀀스 실습

```
SELECT emp_seq.CURRVAL
FROM dual;
```

	 CURRVAL
1	8

```
SELECT sequence_name,  
       min_value,  
       max_value,  
       increment_by,  
       cache_size,  
       last_number,  
       cycle_flag
```

[illegible]

**FROM user\_sequences**  
**WHERE sequence\_name = 'EMP\_SEQ';**

# 학습정리

- 조건절에서 자주 사용되는 컬럼을 인덱스로 만들면 조회 성능을 향상시킬 수 있다.
- 인덱스로 생성된 컬럼을 가공해 조회하면 인덱스를 이용할 수 없다.
- 시노ним은 다른 데이터베이스 객체에 대해 새로운 이름을 부여해 이 이름으로 해당 객체를 참조할 수 있는 데이터베이스 객체이다.
- 시퀀스는 일련번호를 생성하는 데이터베이스 객체이며, 주로 테이블의 숫자형 기본 키 컬럼에 데이터를 입력 시 사용된다.

## Quiz

1. emp 테이블은 emp\_no 컬럼에 기본 키가 잡혀 있다. 실습시간에 실행했던 아래 쿼리를 또 다시 실행하면 어떻게 될까? 그리고 그 이유는 무엇일까?

```
INSERT INTO emp
SELECT emp_seq.NEXTVAL,
       first_name || ' ' || last_name,
       salary, hire_date
FROM employees
WHERE department_id = 90;
```