

13. 시퀀스 • 인덱스

PRIMARY KEY로 지정한 칼럼에 일련번호를 자동으로 부여받기 위해 시퀀스의 생성과 사용 방법을 학습한다.

시퀀스를 수정, 삭제하는 사용 방법을 학습한다.

조회 성능을 향상시키기 위해서 인덱스 객체의 사용 방법을 학습한다.

(1) 시퀀스 생성

시퀀스는 유일(UNIQUE)한 값을 생성해주는 오라클 객체이다.

시퀀스를 생성하면 기본 키와 같이 순차적으로 증가하는 칼럼을 자동적으로 생성할 수 있게 된다.

다음은 시퀀스 생성을 위한 형식이다.

CREATE SEQUENCE *sequence_name*

[START WITH n] ①

[INCREMENT BY n] ②

[{MAXVALUE n | NOMAXVALUE}] ③

[{MINVALUE n | NOMINVALUE}] ④

[{CYCLE | NOCYCLE}] ⑤

[{CACHE n | NOCACHE}] ⑥

① START WITH

시퀀스의 시작 값을 지정한다. n을 1로 지정하면 1부터 순차적으로 시퀀스 번호가 증가 한다.

② INCREMENT BY

시퀀스의 증가 값을 말한다. n을 2로 하면 2씩 증가한다. START WITH를 1로 하고 INCREMENT BY를 2로 하면 1, 3, 5, 7로 시퀀스 번호가 증가하게 된다.

③ MAXVALUE n | NOMAXVALUE

MAXVALUE는 시퀀스가 증가할 수 있는 최댓값을 말한다. NOMAXVALUE는 시퀀스의 값을 무한대로 지정한다.

④ MINVALUE n | NOMINVALUE

MINVALUE는 시퀀스의 최솟값을 지정한다. 기본 값은 1이며, NOMINVALUE를 지정할 경우 최솟값은 무한소가 된다.

⑤ CYCLE(사이클) | NOCYCLE

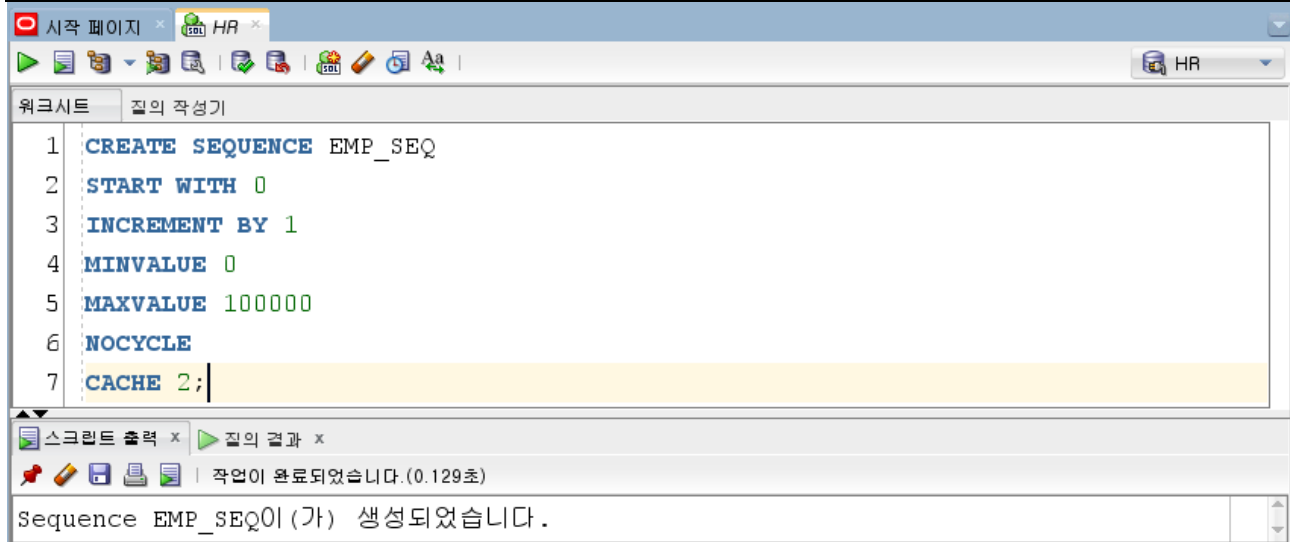
지정된 시퀀스 값이 최댓값까지 증가가 완료되면 다시 최솟값에서부터 시퀀스를 시작하도록 하려면 CYCLE로 지정한다. NOCYCLE은 최댓값을 넘어서면 오류가 발생한다. (디폴트 값 : NOCYCLE)

⑥ CACHE(캐쉬) n | NOCACHE

오라클 서버가 미리 지정하고 메모리에 유지할 값의 수로 디폴트 값은 2이다.

-- 시퀀스 EMP_SEQ 생성

```
CREATE SEQUENCE EMP_SEQ  
START WITH 0  
INCREMENT BY 1  
MINVALUE 0  
MAXVALUE 100000  
NOCYCLE  
CACHE 2;
```

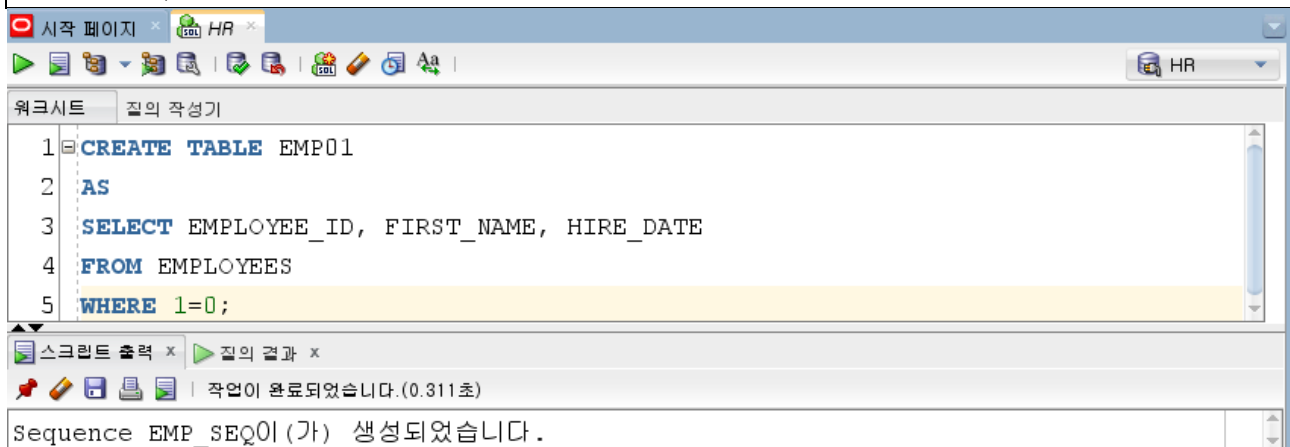


생성된 시퀀스를 사용하기 위해서 EMPLOYEES 테이블에서 일부 칼럼의 구조만 복사하여 비어 있는 테이블을 EMP01란 이름으로 새롭게 생성

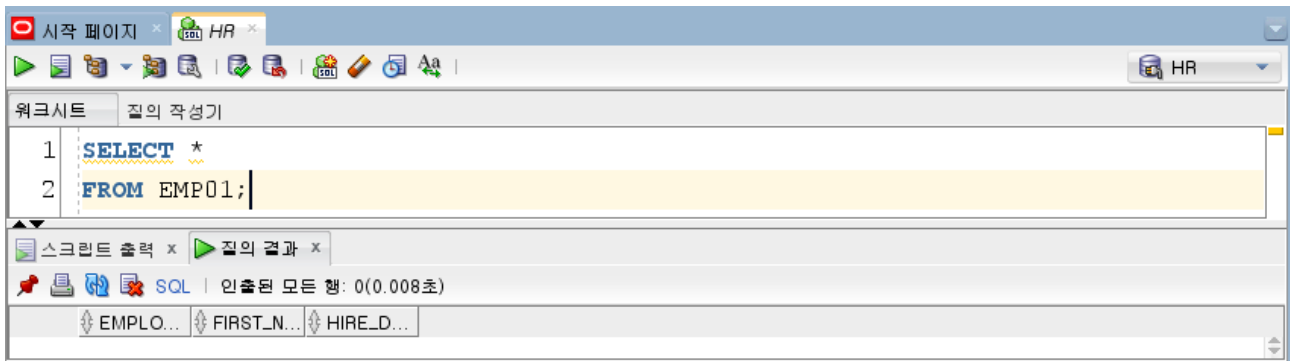
```
DROP TABLE EMP01; -- EMP01 테이블을 제거
```

-- EMPLOYEES 테이블에서 일부 칼럼의 구조만 복사하여 비어 있는 테이블을 생성

```
CREATE TABLE EMP01  
AS  
SELECT EMPLOYEE_ID, FIRST_NAME, HIRE_DATE  
FROM EMPLOYEES  
WHERE 1=0;
```



```
SELECT *
FROM EMP01;
```

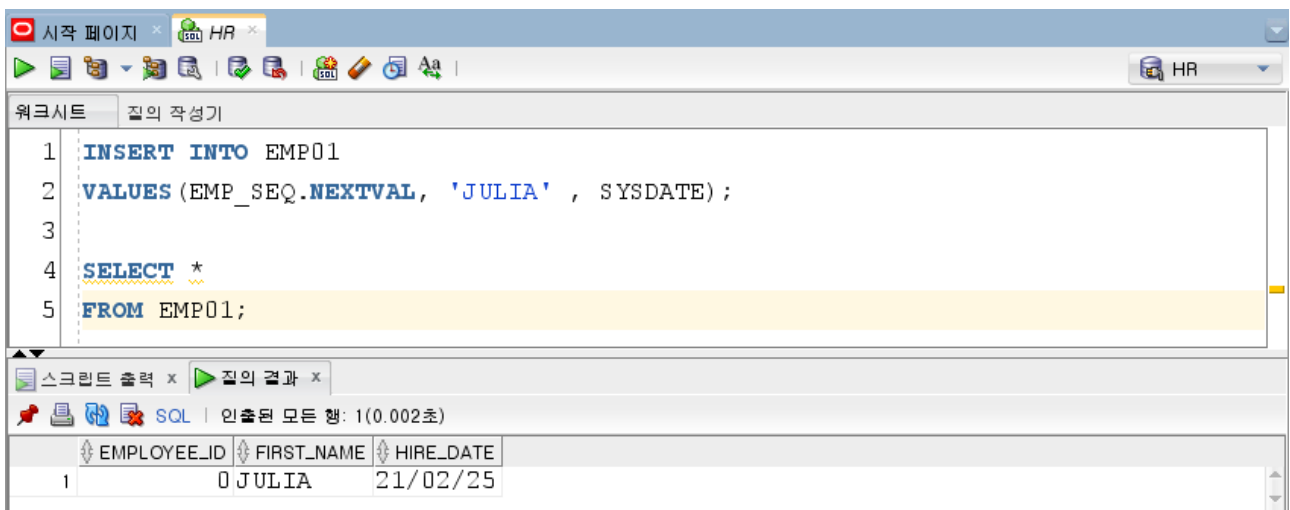


-- EMP_SEQ 시퀀스로부터 직원번호를 자동으로 할당 받아 데이터를 추가

```
INSERT INTO EMP01
VALUES(EMP_SEQ.NEXTVAL, 'JULIA', SYSDATE);
```

-- EMP01 테이블의 내용 확인

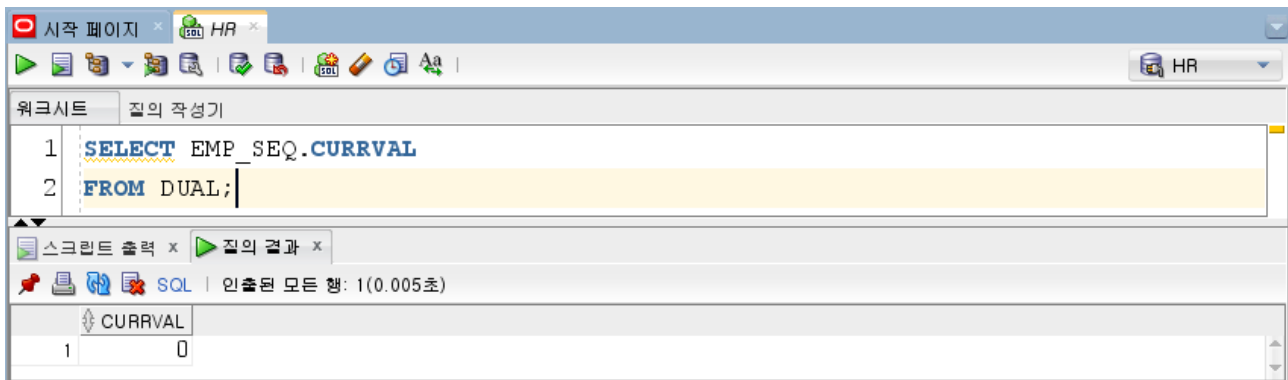
```
SELECT *
FROM EMP01;
```



EMP_SEQ 시퀀스의 현재 값을 알고 싶을 때에는 CURRVAL를 사용한다.

-- EMP_SEQ 시퀀스의 현재 값을 알아봄

```
SELECT EMP_SEQ.CURRVAL
FROM DUAL;
```



CURRVAL : 현재 값을 반환

NEXTVAL : 현재 시퀀스 값의 다음 값을 반환

NEXTVAL, CURRVAL을 사용할 수 있는 경우

- | | |
|----------------------|---------------------|
| - 서브 쿼리가 아닌 SELECT 문 | - INSERT 문의 VALUES절 |
| - INSERT 문의 SELECT 절 | - UPDATE 문의 SET 절 |

NEXTVAL, CURRVAL을 사용할 수 없는 경우

- | | |
|--------------------------------------------|---------------------------------|
| - VIEW의 SELECT 절 | - SELECT, DELETE, UPDATE의 서브 쿼리 |
| - DISTINCT 키워드가 있는 SELECT 문 | - CREATE TABLE, ALTER TABLE 명령의 |
| - GROUP BY, HAVING, ORDER BY절이 있는 SELECT 문 | DEFAULT 값 |

(2) 시퀀스 수정 및 삭제

시퀀스를 사용하기 위해서 DEPT테이블의 구조만 복사하고 내용은 비어 있는 테이블을 DEPT01이라는 이름으로 새롭게 생성한다.

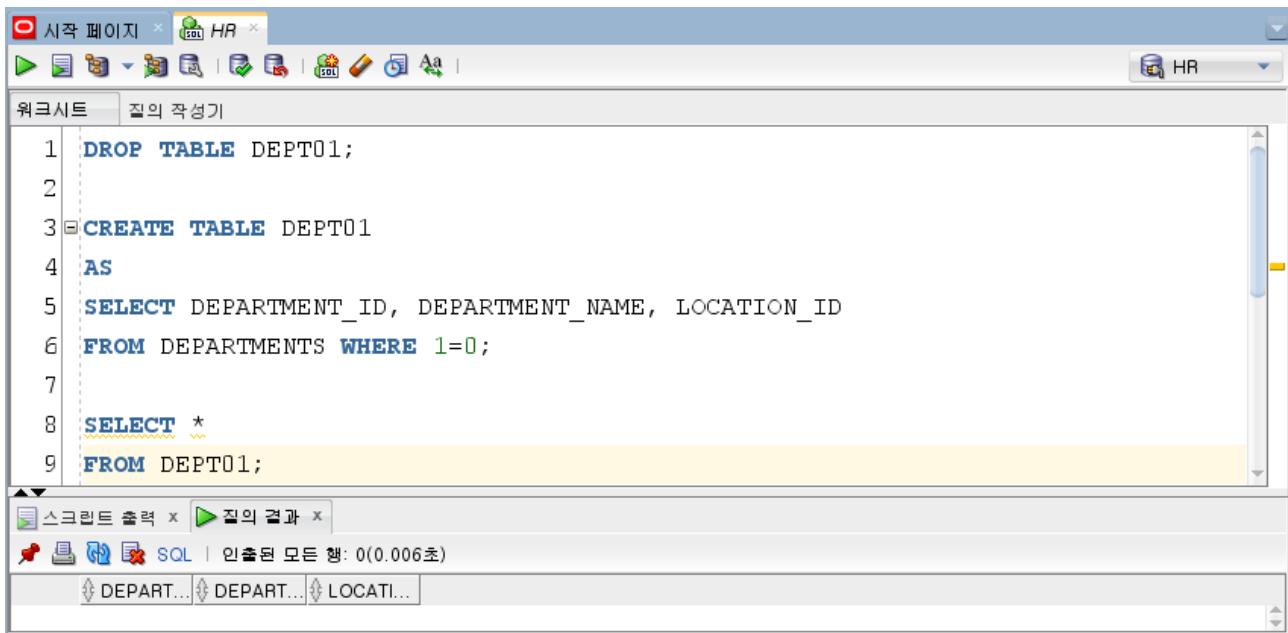
```
DROP TABLE DEPT01; -- DEPT01 테이블을 제거
```

-- DEPT 테이블의 구조만 복사하고 내용은 비어 있는 테이블을 생성

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

-- 새롭게 생성된 DEPT01 테이블의 내용 확인

```
SELECT *
FROM DEPT01;
```

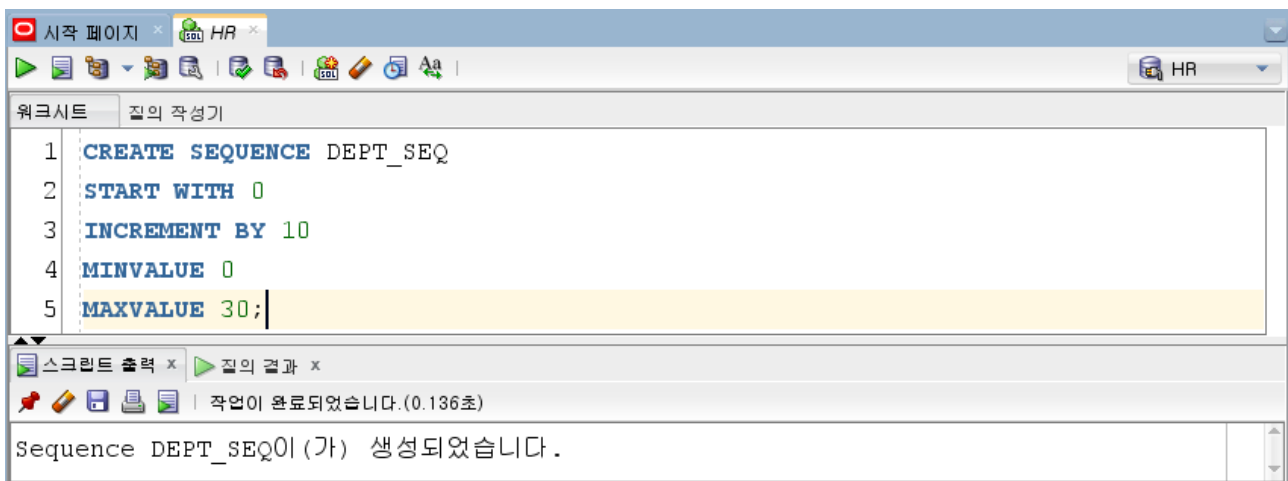


-- 10부터 10씩 증가하면서 최대 30까지의 값을 갖는 시퀀스를 생성

```

CREATE SEQUENCE DEPT_SEQ
START WITH 0
INCREMENT BY 10
MINVALUE 0
MAXVALUE 30;

```



-- DEPT_SEQ 시퀀스로부터 부서번호를 자동으로 할당받아 데이터를 추가

```

INSERT INTO DEPT01
VALUES(DEPT_SEQ.NEXTVAL, '인사과', 2);

INSERT INTO DEPT01
VALUES(DEPT_SEQ.NEXTVAL, '총무과', 42);

INSERT INTO DEPT01

```

```
VALUES(DEPT_SEQ.NEXTVAL, '교육팀', 32);
```

```
SELECT *  
FROM DEPT01;
```

The screenshot shows the SQL Developer interface with a script editor and a results window. The script editor contains the following SQL statements:

```
1 INSERT INTO DEPT01  
2 VALUES (DEPT_SEQ.NEXTVAL, '인사과', 2);  
3  
4 INSERT INTO DEPT01  
5 VALUES (DEPT_SEQ.NEXTVAL, '총무과', 42);  
6  
7 INSERT INTO DEPT01  
8 VALUES (DEPT_SEQ.NEXTVAL, '교육팀', 32);  
9  
10 SELECT *  
11 FROM DEPT01;
```

The results window shows the output of the SELECT statement:

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
1	10 인사과	2
2	20 총무과	42
3	30 교육팀	32

-- DEPT_SEQ 시퀀스가 NOCYCLE 상태이므로 MAXVALUE 값인 30을 초과하게 되면 오류

```
INSERT INTO DEPT01  
VALUES(DEPT_SEQ.NEXTVAL, '기술팀', 44);
```

The screenshot shows the SQL Developer interface with a script editor and a results window. The script editor contains the following SQL statements:

```
1 INSERT INTO DEPT01  
2 VALUES (DEPT_SEQ.NEXTVAL, '기술팀', 44);
```

The results window shows the output of the INSERT statement, which is an error message:

```
명령의 1 행에서 시작하는 중 오류 발생 -  
INSERT INTO DEPT01  
VALUES (DEPT_SEQ.NEXTVAL, '기술팀', 44)  
오류 보고 -  
ORA-08004: 시퀀스 DEPT_SEQ.NEXTVAL exceeds MAXVALUE은 사례로 될 수 없습니다
```

```
SELECT *
FROM DEPT01;
```

① 시퀀스에 관한 데이터 디렉터리 USER_SEQUENCES

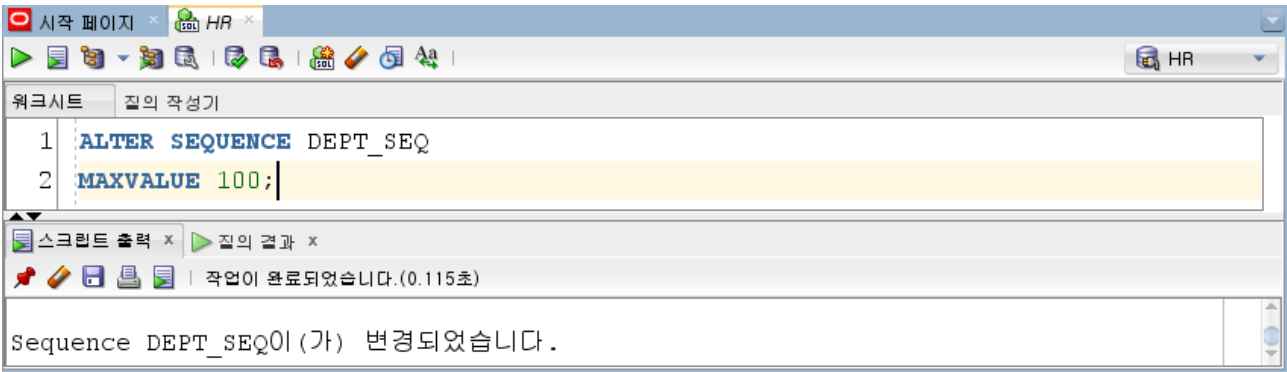
```
-- USER_SEQUENCES 데이터 딕셔너리로 시퀀스의 정보를 확인
```

```
SELECT SEQUENCE_NAME, MIN_VALUE, MAX_VALUE, INCREMENT_BY, CYCLE_FLAG
FROM USER_SEQUENCES;
```

```
ALTER SEQUENCE sequence_name
    [INCREMENT BY n]
    [{MAXVALUE n | NOMAXVALUE}]
    [{MINVALUE n | NOMINVALUE}]
    [{CYCLE | NOCYCLE}]
    [{CACHE n | NOCACHE}]
```

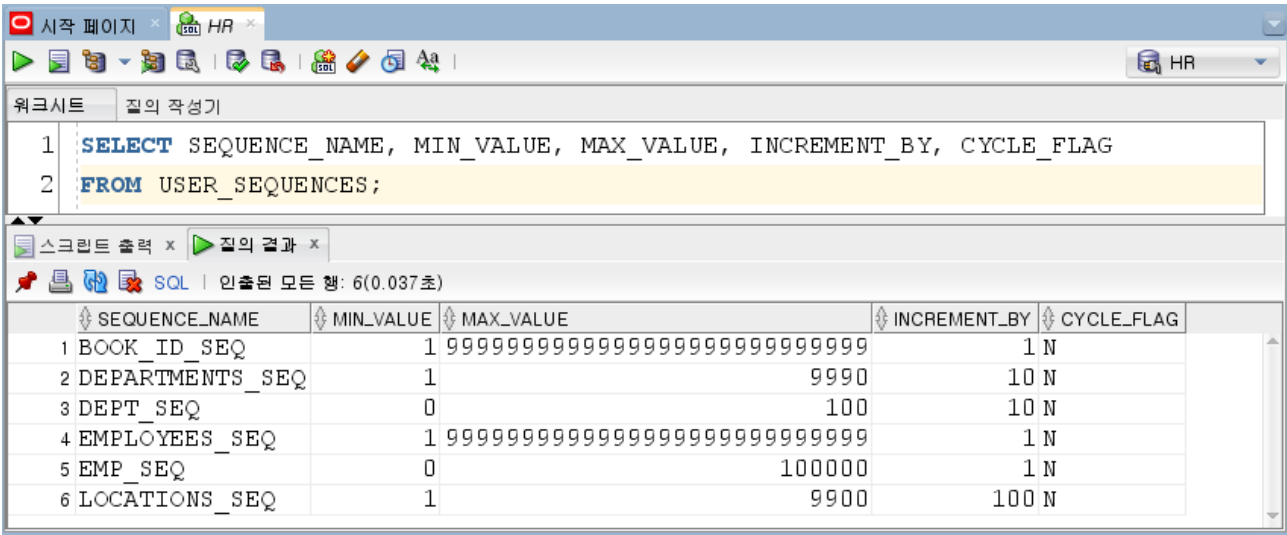
```
ALTER SEQUENCE DEPT_SEQ
```

```
MAXVALUE 100;
```



```
-- USER_SEQUENCES 데이터 딕셔너리로 시퀀스의 최대값이 변경되었는지 확인
```

```
SELECT SEQUENCE_NAME, MIN_VALUE, MAX_VALUE, INCREMENT_BY, CYCLE_FLAG
FROM USER_SEQUENCES;
```

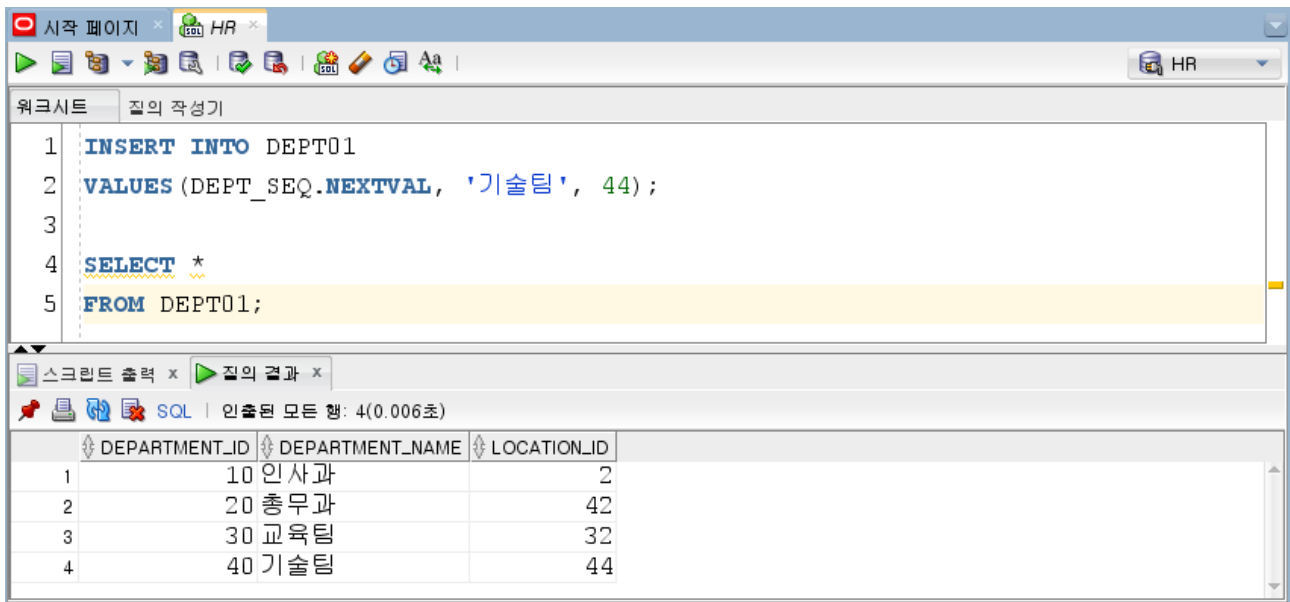


-- 새로운 부서 추가

```
INSERT INTO DEPT01
VALUES(DEPT_SEQ.NEXTVAL, '기술팀', 44);
```

-- 40번 부서가 추가된 것을 확인

```
SELECT *
FROM DEPT01;
```

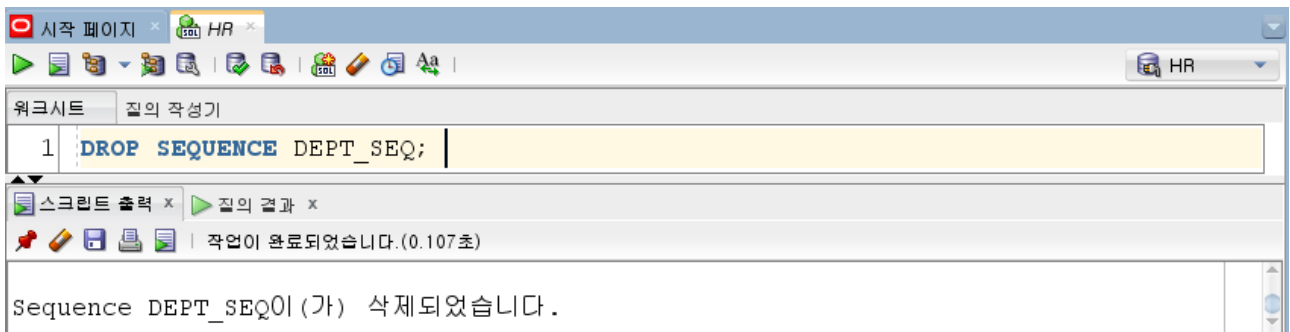
② 시퀀스의 삭제

시퀀스를 삭제하려면 DROP SEQUENCE를 사용한다.

DROP SEQUENCE sequence_name;

-- 더 이상 사용되지 않는 시퀀스를 삭제

DROP SEQUENCE DEPT_SEQ;



(3) 조회 시 성능 향상을 위한 인덱스

인덱스(INDEX)는 테이블에 있는 데이터를 빨리 찾기 위한 용도의 데이터베이스 객체다.

인덱스의 장점
<ul style="list-style-type: none">- 검색 속도가 빨라진다.- 시스템에 걸리는 부하를 줄여서 시스템 전체 성능을 향상시킨다.

인덱스의 단점
<ul style="list-style-type: none">- 인덱스를 위한 추가적인 공간이 필요하다.- 인덱스를 생성하는데 시간이 걸린다.- 데이터의 변경 작업(INSERT/UPDATE/DELETE)이 자주 일어날 경우에는 오히려 성능이 저하된다.

```
DROP TABLE EMP01; -- EMP01 테이블 삭제
```

-- EMPLOYEES 테이블을 복사해서 EMP01 생성

```
CREATE TABLE EMP01
AS
SELECT *
FROM EMPLOYEES;
```

-- EMPLOYEES 와 EMP01 테이블에 설정된 인덱스를 확인

-- USER_IND_COLUMNS 데이터 디셔너리에서 인덱스의 존재를 확인.

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

The screenshot shows the Oracle SQL Developer interface. The top toolbar includes icons for running queries, saving, and other database operations. The main window displays a SQL script with the following query:

```
1 SELECT TABLE_NAME, INDEX_NAME, COLUMN_NAME
2 FROM USER_IND_COLUMNS
3 WHERE TABLE_NAME IN ('EMPLOYEES', 'EMP01');
```

Below the script, the 'Results' tab is active, showing the output of the query. The results are displayed in a table with three columns: TABLE_NAME, INDEX_NAME, and COLUMN_NAME. The data is as follows:

TABLE_NAME	INDEX_NAME	COLUMN_NAME
EMPLOYEES	EMP_EMAIL_UK	EMAIL
EMPLOYEES	EMP_EMP_ID_PK	EMPLOYEE_ID
EMPLOYEES	EMP_DEPARTMENT_IX	DEPARTMENT_ID
EMPLOYEES	EMP_JOB_IX	JOB_ID
EMPLOYEES	EMP_MANAGER_IX	MANAGER_ID
EMPLOYEES	EMP_NAME_IX	LAST_NAME
EMPLOYEES	EMP_NAME_IX	FIRST_NAME

EMPLOYEES를 서브 쿼리로 복사한 EMP01 테이블은 구조와 내용만 복사될 뿐 제약 조건은 복사되지 않는다.

```
SELECT *
FROM EMP01
WHERE EMPLOYEE_ID= 188;
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
188	Kelly	Chung	KCHUNG	650.505.1876	05/06/14	SH_CLERK	3800	(null)	122	50

위 쿼리문을 F10를 눌러 계획 설명을 확인하면 TABLE ACCESS로 데이터를 조회한 것을 확인할 수 있다.

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				3
TABLE ACCESS	EMP01	FULL	1	3

```
SELECT *
FROM EMPLOYEES
WHERE EMPLOYEE_ID= 188;
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
188	Kelly	Chung	KCHUNG	650.505.1876	05/06/14	SH_CLERK	3800	(null)	122	50

위 쿼리문을 F10를 눌러 계획 설명을 확인하면 INDEX로 데이터를 조회한 것을 확인할 수 있다.

SQL Developer window showing the execution plan for the query:

```

1 SELECT *
2 FROM EMPLOYEES
3 WHERE EMPLOYEE_ID= 188;

```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				1
TABLE ACCESS	EMPLOYEES	BY INDEX ROWID	1	1
INDEX	EMP_EMP_ID_PK	UNIQUE SCAN	1	0

Additional details in the plan:

- Access Predicates: EMPLOYEE_ID=188
- Other XML:
 - info type='db_version': 18.0.0.0
 - info type='parse_schema': 'HR'
 - info type='plan_hash_full': 3236529094
 - info type='plan_hash': 1833546154
 - info type='plan_hash_2': 3236529094
 - hint:
 - INDEX_RS_ASC(@SEL\$1 'EMPLOYEES'@SEL\$1 ('EMPLOYEES','EMPLOYEE_ID'))
 - OUTLINE_LEAF(@SEL\$1)
 - ALL_ROWS
 - DB_VERSION('18.1.0')
 - OPTIMIZER_FEATURES_ENABLE('18.1.0')
 - IGNORE_OPTIM_EMBEDDED_HINTS

제약조건에 의해 자동으로 생성되는 인덱스 외에 CREATE UNIQUE INDEX명령어로 직접 인덱스를 생성할 수 있다.

```

CREATE UNIQUE INDEX index_name
ON table_name (column_name);

```

-- EMP01 테이블의 EMPNO 칼럼에 인덱스를 생성

```

CREATE UNIQUE INDEX INDEX_EMPNO_EMP
ON EMP01 (EMPLOYEE_ID);

```

SQL Developer window showing the execution of the command:

```

1 CREATE UNIQUE INDEX INDEX_EMPNO_EMP
2 ON EMP01 (EMPLOYEE_ID);

```

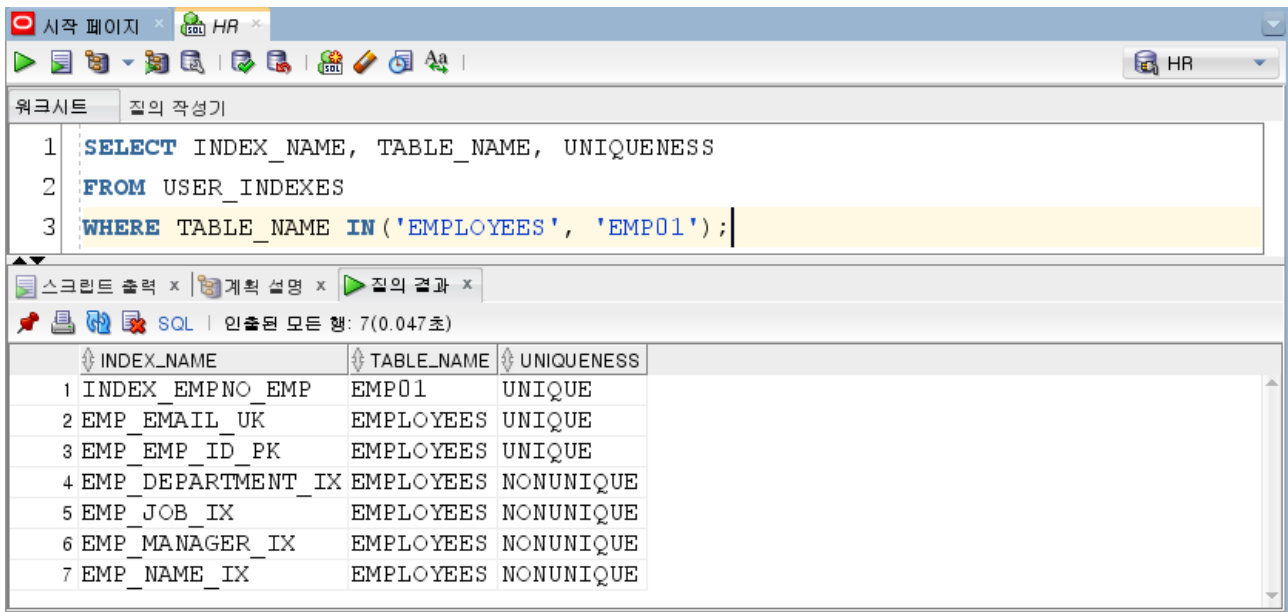
Message: INDEX INDEX_EMPNO_EMP이 (가) 생성되었습니다.

-- EMP와 EMP01 테이블에 설정된 인덱스를 확인

```

SELECT INDEX_NAME, TABLE_NAME, UNIQUENESS
FROM USER_INDEXES
WHERE TABLE_NAME IN('EMPLOYEES', 'EMP01');

```

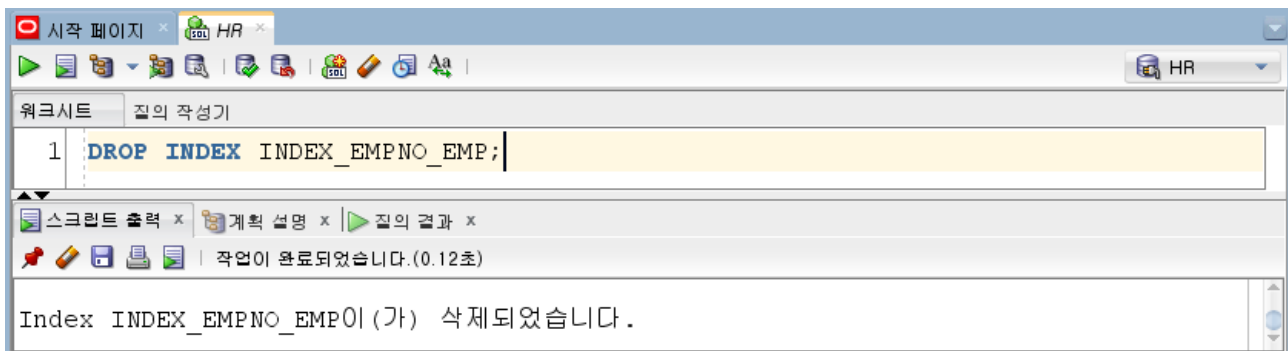


인덱스 삭제는 DROP INDEX 명령어를 사용한다.

DROP INDEX *index_name*;

-- EMP01 테이블의 EMPNO 칼럼에 설정한 인덱스를 제거

DROP INDEX INDEX_EMPNO_EMP;



-- EMPLOYEES와 EMP01 테이블에 설정된 인덱스를 확인

```

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

```

시작 페이지 x HR x

워크시트 | 질의 작성기

```

1 SELECT TABLE_NAME, INDEX_NAME, COLUMN_NAME
2 FROM USER_IND_COLUMNS
3 WHERE TABLE_NAME IN ('EMPLOYEES', 'EMP01');

```

스크립트 출력 x | 계획 설명 x | 질의 결과 x

SQL | 인출된 모든 행: 7(0.058초)

	TABLE_NAME	INDEX_NAME	COLUMN_NAME
1	EMPLOYEES	EMP_EMAIL_UK	EMAIL
2	EMPLOYEES	EMP_EMP_ID_PK	EMPLOYEE_ID
3	EMPLOYEES	EMP_DEPARTMENT_IX	DEPARTMENT_ID
4	EMPLOYEES	EMP_JOB_IX	JOB_ID
5	EMPLOYEES	EMP_MANAGER_IX	MANAGER_ID
6	EMPLOYEES	EMP_NAME_IX	LAST_NAME
7	EMPLOYEES	EMP_NAME_IX	FIRST_NAME

인덱스 생성할 때 고려해야 할 사항을 정리하면 다음과 같다.

- 일반적으로 테이블 전체 로우 수의 15%이하의 데이터를 조회할 때 인덱스를 생성한다.
- 테이블 건수가 적다면 굳이 인덱스를 만들 필요가 없다.
- NULL을 많이 포함된 컬럼은 인덱스 컬럼으로 만들기 적당치 않다.
- 테이블에 만들 수 있는 인덱스 수의 제한은 없으나, 너무 많이 만들면 오히려 성능 저하가 발생한다.