Chapter 13. 데이터베이스 객체

이번 장에서는 테이블과 뷰를 제외한 나머지 데이터베이스 객체인 시퀀스(Sequence), 인덱스(Index), 동의어(Synonym)에 대하여 설명한다.

시퀀스

시퀀스는 여러 사용자들이 공유하는 데이터베이스 객체로서 호출 될 때마다 중복되지 않은 고유한 숫자를 리턴하는 객체이다. 그러므로, 시퀀스는 기본키 컬럼에 사용할 값을 발생시키는데 주로 사용된다. 이와 같이, 번호를 중복되지 않게 취득하는 것을 보통 채번이라고 부르는데 반드시 시퀀스를 사용해야 하는 것은 아니며 애플리케이션 코드에 의해서도 구현될 수 있다. 그러나 시퀀스를 이용하면 한 번 호출시 여러 개의 시퀀스 번호를 메모리에 미리 캐시에 저장해두고 사용할 수 있으므로 작업 속도를 향상 시킬 수 있다.

■ 시퀀스 생성

시퀀스 번호를 자동으로 생성해주는 시퀀스를 작성하는 명령은 다음과 같다.

```
CREATE SEQUENCE sequence
       [INCREMENT BY n]
       [START WITH n]
       [{MAXVALUE n | NOMAXVALUE}]
       [{MINVALUE n | NOMINVALUE}]
       [{CYCLE | NOCYCLE)]
       [{CACHE n | NOCACHE}];
```

위에서 START WITH에 시퀀스 번호의 초기값을 입력하고, INCREMENT BY에 의해 시퀀스 번호의 증분을 기술한다. MAXVALUE와 MINVALUE에는 시퀀스 번호의 최대값 및 최소값을 입력한다. 만약, NOMAXVALUE로 지정된 경우 증분이 (+)이면 10^{27}, (-)이면 -1이 되며 NOMINVALUE로 지정된 경우 증분이 (+)이면 1, (-)이면 10^{26}이 된다.

시퀀스 작성시 CYCLE을 지정하면 시퀀스 번호가 최대값에 도달한 후 순환되므로. 시퀀스 번호를 기본키에 사용해서는 안된다.

다음은 EMP 테이블의 기본키 할당을 위한 시퀀스를 작성한 것이다.

```
SQL> CREATE SEQUENCE EMP_EMPNO_SEQ
  2  INCREMENT BY 1
  3  START WITH 8000
  4  MAXVALUE 9999
  5  NOCACHE
  6  NOCYCLE;

주문번호가 생성되었습니다.
```

작성된 시퀀스는 USER_SEQUENCES 데이터 딕셔너리에서 확인 할 수 있다.

```
SQL> SELECT SEQUENCE_NAME, MIN_VALUE, MAX_VALUE,
  2  INCREMENT_BY, LAST_NUMBER
  3  FROM USER_SEQUENCES;

SEQUENCE_NAME                  MIN_VALUE  MAX_VALUE INCREMENT_BY LAST_NUMBER
------------------------------ ---------- ---------- ------------ -----------
EMP_EMPNO_SEQ                           1       9999            1        8000
```

시퀀스 작성시 NOCACHE를 지정하면 LAST_NUMBER컬럼에 다음 발생할 시퀀스 번호가 표시된다.

■ NEXTVAL, CURRVAL 가상 컬럼

시퀀스를 작성한 후에는 NEXTVAL, CURRVAL 가상 컬럼을 이용하여 시퀀스 번호를 검색할 수 있다. NEXTVAL 가상 컬럼은 지정된 시퀀스에서 순차적인 시퀀스 번호를 추출할 때 사용하는데, 시퀀스명.NEXTVAL에 의해서 시퀀스 번호가 추출되면 해당 시퀀스의 CURRVAL에 추출된 시퀀스 번호가 저장된다. CURRVAL 가상 컬럼은 사용자가 방금 추출한 시퀀스 번호를 참조하는데 사용되므로, 반드시 NEXTVAL에 의해서 시퀀스 번호를 추출한 후 사용해야한다.

NEXTVAL과 CURRVAL 가상 컬럼을 사용 할 수 있는 SQL 문장은 다음과 같다.
· 서브쿼리가 아닌 SELECT 문장의 컬럼
· INSERT 문장내 SELECT 문장의 컬럼
· INSERT 문장의 VALUES 절
· UPDATE 문장의 SET 절

다음은 NEXTVAL과 CURRVAL 가상 컬럼을 사용 할 수 없는 SQL 문장이다.
· 뷰 안에 포함된 SELECT 문장의 컬럼
· DISTINCT가 포함된 SELECT 문장
· GROUP BY, HAVING, ORDER BY 절이 포함된 SELECT 문장
· SELECT, DELETE, UPDATE 문장내 서브쿼리
· CREATE TABLE 또는 ALTER TABLE 문장에서 DEFAULT 값

■ 시퀀스의 사용

앞 절에서 생성한 EMP_EMPNO_SEQ를 이용하여 사원 테이블에 신규 데이터를 입력하면 다음과 같다.

```
SQL> INSERT INTO EMP
  2  VALUES (EMP_EMPNO_SEQ.NEXTVAL, '길동', '홍보', NULL,
  3  SYSDATE, 3000, 300, 40);

1 개의 행이 만들어졌습니다.
```

EMP_EMPNO_SEQ 시퀀스에서 현재 시퀀스 번호를 확인하면 다음과 같다.

```
SQL> SELECT EMP_EMPNO_SEQ.CURRVAL FROM DUAL;

   CURRVAL
----------
      8000
```

시퀀스 생성시 CACHE 옵션을 사용하면 시퀀스 번호의 추출을 좀 더 빠르게 진행 할 수 있다. NEXTVAL에 의해 최초로 시퀀스 번호를 추출하면 CACHE 옵션에 기술된 숫자만큼의 시퀀스 번호가 추출되어 메모리에 캐시되며 이후, 다시 NEXTVAL에 의해 시퀀스 번호를 추출하면 캐시된 시퀀스 번호를 하나씩 리턴해준다. 이 후, 캐시된 시퀀스 번호가 모두 소비되면 다음번 NEXTVAL에 의해 다시 시퀀스 번호들이 메모리에 캐시된다.

시퀀스에 의해 추출되는 번호는 순차적이지만, 시퀀스 번호는 COMMIT 또는 ROLLBACK 명령의 영향을 받지 않기 때문에 시퀀스 번호에는 빈 번호들이 발생 할 수 있다. 즉, INSERT 문장에 의해 시퀀스 번호를 테이블에 입력한 다음, 해당 문장이 롤백되었다면 해당 시퀀스 번호를 영원히 잃어버리게 된다. 또한, 시퀀스 번호를 캐시하여 사용하는 경우, 시스템 오류에 의해 데이터베이스가 종료 되면 미리 캐시해둔 시퀀스 번호들도 모두 잃게 된다.

■ 시퀀스 변경

시퀀스의 증분, 최대값, 최소값, 순환여부, 캐시여부를 변경하는 방법은 다음과 같다.

```
SQL> ALTER SEQUENCE EMP_EMPNO_SEQ
  2  INCREMENT BY 2
  3  MAXVALUE 9000
  4  NOCACHE
  5  NOCYCLE;

주문번호가 변경되었습니다.
```

시퀀스 변경시 주의사항은 다음과 같다.

· 시퀀스를 변경하려면 해당 시퀀스의 소유자이거나 ALTER SEQUENCE 권한을 부여받아야 한다.
· 시퀀스가 변경되면 다음번 시퀀스 번호 추출부터 변경사항이 적용된다.
· START WITH 옵션은 변경 할 수 없으며, 시퀀스를 삭제하고 재생성해야만 한다.
· MAXVALUE 값은 현재 시퀀스 번호보다 큰 번호로 지정해야 한다.

■ 시퀀스 삭제

데이터 딕셔너리에서 시퀀스를 삭제하는 방법은 다음과 같으며, 해당 시퀀스의 소유자이거나 DROP ANY SEQUENCE 권한을 부여받아야 한다.

```
SQL> DROP SEQUENCE EMP_EMPNO_SEQ;

주문번호가 삭제되었습니다.
```

인덱스

인덱스는 테이블에서 행을 검색할 때 검색 속도를 높이기 위해 Oracle 서버가 사용하는 스키마 객체이다. 테이블에 인덱스를 생성하지 않으면 데이터를 검색 할 때 테이블의 모든 데이터를 읽어 검색 조건에 의해 데이터가 선별되지만, 인덱스를 사용하면 조건에 맞는 데이터를 해당 테이블에 직접 접근하여 읽어오기 때문에 디스크의 I/O를 급격히 감소시킬 수 있다. 또한, 해당 테이블과 논리적으로 독립적이며 Oracle 서버에 의해 자동으로 사용 및 관리된다. 반면, 테이블을 삭제하면 관련 인덱스도 삭제된다.

인덱스는 Oracle 서버에 의해 자동으로 생성되기도 하며, 사용자의 요구에 의해 직접 생성할 수도 있다. 테이블에 PRIMARY KEY 제약조건 또는 UNIQUE 제약조건을 지정하면 해당 컬럼에는 고유 인덱스가 생성되며, 사용자가 필요에 따라 특정 컬럼에 별도의 인덱스를 생성하여 검색 속도를 높일 수 있다. 제약조건 생성시에 자동으로 생성되는 인덱스의 이름은 제약조건의 이름과 동일하게 부여된다.

참고로, FOREIGN KEY 제약조건이 지정된 컬럼에 인덱스를 생성해두면 조인시 검색속도가 향상되므로, 외래키 컬럼에는 반드시 인덱스를 생성하도록 한다.

■ 인덱스 생성

인덱스는 한 개 컬럼 또는 두 개 이상의 컬럼을 결합하여 생성 할 수 있다. 컬럼이 두 개 이상 결합되어 생성된 인덱스를 결합 인덱스라고 부른다.

```
CREATE INDEX index
ON table (column[, column] ...);
```

사원 테이블에서 사원명을 조건으로 검색하는 경우가 많다면 해당 컬럼에 인덱스를 생성하여 검색 속도를 향상 시킬 수 있다.

```
SQL> CREATE INDEX EMP_ENAME_IDX
  2  ON EMP(ENAME);

인덱스가 생성되었습니다.
```

■ 인덱스를 생성할 컬럼의 선정

인덱스가 많을수록 검색속도가 반드시 향상되는 것은 아니다. 또한, 인덱스가 생성된 테이블에 DML 작업을 수행하면 관련 인덱스도 변경되어야 하므로 오히려 속도가 저하된다. 그러므로 인덱스의 수는 최소한으로 유지하면서 최대한 활용될 수 있도록 인덱스를 적절한 컬럼에 생성하도록 하여야 한다.

일반적으로 인덱스를 작성해야 경우는 다음과 같다.

· 값의 범위가 넓은 컬럼. 즉, 컬럼내의 값이 다양할수록 좋다.
· NULL 값이 많은 컬럼. NULL 값은 인덱스에 포함되지 않기 때문에 인덱스의 크기를 줄일 수 있다.

· WHERE 절이나 조인 조건에 사용되는 컬럼
· 테이블이 크고 대부분의 쿼리 문장이 테이블내 전체 데이터의 약 2~4% 이내를 검색하는 경우

다음은 인덱스를 작성할 필요가 없는 경우로서 인덱스를 생성하면 오히려 검색 속도가 늦어질 수도 있다.
· 테이블이 작은 경우
· 쿼리 문장의 조건에 자주 사용되지 않는 컬럼
· 대부분의 쿼리 문장이 테이블내 전체 데이터의 약 2~4% 이상을 검색하는 경우
· 테이블이 자주 변경되는 경우
· 인덱스가 작성된 컬럼이 쿼리 문장의 조건에서 표현식에 포함된 경우

■ 인덱스 확인

USER_INDEXES 데이터 딕셔너리와 USER_IND_COLUMNS 데이터 딕셔너리를 이용하면 인덱스와 관련된 정보를 확인 할 수 있다.

```
SQL> SELECT IC.INDEX_NAME, IC.COLUMN_NAME,
  2  IC.COLUMN_POSITION, IX.UNIQUENESS
  3  FROM USER_INDEXES IX, USER_IND_COLUMNS IC
  4  WHERE IC.INDEX_NAME = IX.INDEX_NAME
  5  AND IC.TABLE_NAME = 'EMP';

INDEX_NAME           COLUMN_NAME                    COLUMN_POSITION UNIQUENES
-------------------- ------------------------------ --------------- ---------
PK_EMP               EMPNO                                        1 UNIQUE
EMP_ENAME_IDX        ENAME                                        1 NONUNIQUE
```

■ 함수 기반 인덱스

인덱스가 생성된 컬럼이 함수에 입력되어 사용되거나 연산에 사용되면 절대 인덱스를 사용할 수 없다. 그래서 함수 기반 인덱스는 WHERE 조건의 함수 또는 연산식 자체를 인덱스로 생성한 것이다. 함수 기반 인덱스를 생성하려면 QUERY REWRITE 권한을 부여받아야 한다.

```
SQL> CONNECT / AS SYSDBA
연결되었습니다.
SQL> GRANT QUERY REWRITE TO SCOTT;

권한이 부여되었습니다.

SQL> CONNECT SCOTT/TIGER
연결되었습니다.
SQL> CREATE INDEX LOWER_JOB_IDX
  2  ON EMP(LOWER(JOB));

인덱스가 생성되었습니다.

SQL> SELECT * FROM EMP WHERE LOWER(JOB) = 'manager';
```

■ 인덱스 삭제

인덱스를 삭제하는 방법은 다음과 같다. 인덱스를 삭제하려면 인덱스의 소유자이거나 DROP ANY INDEX 권한을 부여받아야 한다.

```
SQL> DROP INDEX LOWER_JOB_IDX;

인덱스가 삭제되었습니다.
```

참고로, 테이블을 삭제하면 관련 인덱스와 제약조건은 자동으로 삭제되지만 뷰와 시퀀스는 삭제되지 않는다.

동의어

동의어는 객체에 대한 별칭이다. 예를 들어, 다른 사용자가 소유한 테이블을 검색하기 위해서는 검색 할 테이블의 앞에 소유자의 이름을 붙여주어야 하지만 동의어를 사용하면 좀 더 간단하게 해당 테이블을 검색 할 수 있다.

■ 동의어 생성

동의어를 생성하는 명령어는 다음과 같다.

```
CREATE [PUBLIC] SYNONYM synonym
FOR object;
```

EMP 테이블에 대한 동의어를 생성하면, 동의어를 통하여 검색이 가능하다.

```
SQL> CREATE SYNONYM E FOR EMP;

동의어가 생성되었습니다.

SQL> SELECT EMPNO, ENAME FROM E
  2  WHERE ENAME = 'SMITH';

     EMPNO ENAME
---------- ----------
      7369 SMITH
```

동의어 작성시 PUBLIC 옵션을 추가하면 다른 사용자들도 같이 사용 할 수 있는 공용 동의어가 생성된다. 그러나, DBA 권한을 갖는 관리자만이 공용 동의어를 만들 수 있으며 다른 사용자들이 사용하기 위해서는 공용 동의어에 의해 참조되는 테이블에 접근 할 수 있는 권한을 부여 받아야 한다.

```
SQL> CONNECT / AS SYSDBA;
연결되었습니다.
SQL> CREATE PUBLIC SYNONYM HR_EMP FOR HR.EMPLOYEES;

동의어가 생성되었습니다.

SQL> GRANT SELECT ON HR.EMPLOYEES TO SCOTT;

권한이 부여되었습니다.

SQL> CONNECT SCOTT/TIGER
연결되었습니다.

SQL> SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME FROM HR_EMP
  2  WHERE FIRST_NAME = 'Steven';

EMPLOYEE_ID FIRST_NAME           LAST_NAME
----------- -------------------- -------------------------
        100 Steven               King
        128 Steven               Markle
```

■ 동의어 삭제

동의어를 삭제하는 방법은 다음과 같다. 공용 동의어는 관리자만이 삭제할 수 있다.

```
SQL> DROP SYNONYM E;

동의어가 삭제되었습니다.

SQL> CONNECT / AS SYSDBA
연결되었습니다.
SQL> DROP PUBLIC SYNONYM HR_EMP;

동의어가 삭제되었습니다.
```

복습

1. 초기값 1, 증분 1, 최소값 1, 최대값 9999인 TEST_SEQ를 생성하시오. 단, 시퀀스 번호는 순환되지 않으며, 시퀀스 번호 추출시 한번에 10개씩 메모리에 캐시하도록 한다.

2. 사원 테이블의 외래키인 부서코드 컬럼에 인덱스를 작성하시오.

3. 사원 테이블의 사원명 컬럼과 업무 컬럼에 결합 인덱스를 작성하시오.

4. 부서 테이블에 대한 동의어 D를 생성하시오.

5. 사원 테이블에 대한 공용 동의어 E를 생성하시오.