Chapter 10

SEQUENCE (순차적으로 증가하는 값)

INDEX

- 자동적으로 유일 번호를 생성합니다.
- 공유 가능한 객체입니다.
- 주로 기본 키 값을 생성하기 위해 사용합니다.
- 어플리케이션 코드를 대체합니다.
- 메모리에 캐쉬(Cache)되면 시퀀스 값을 액세스 하는 효율성을 향상시킵니다.

시퀀스는 테이블의 행에 대한 시퀀스 번호를 자동적으로 생성하기 위해 사용될 수 있습니다. 시퀀스는 사용자가 생성한 데이터베이스 객체이며 여러 사용자가 공유할 수 있습니다. 시퀀스의 가장 일반적이고 전형적인 사용은 각 행을 유일하게 구분하는 기본 키 값을 생성하기 위해서 입니다. 시퀀스 번호는 테이블과 관계없이 생성되고 저장됩니다. 그러므로 동일한 시퀀스는 여러 테이블에 대해 사용될 수 있습니다.

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

다음 구문은 DEPTS 테이블의 DEPARTMENT_ID에 대해 사용되는 DEPTS_SEQ라는 시퀀스를 생성합니다. 시퀀스는 91에서 시작하며, 캐쉬를 허용치 않으며, 사이클을 허용치 않습니다.

```
SQL> CREATE SEQUENCE depts_seq
  2      INCREMENT BY 1
  3      START WITH  91
  4      MAXVALUE   100
  5      NOCACHE
  6      NOCYCLE;

Sequence DEPTS_SEQ이(가) 생성되었습니다.
```

CACHE- 시퀀스를 빨리 제공하기 위해서 미리 메모리에 CACHE 갯수 만큼의 시퀀스를 만들어서 준비
CYCLE – 최대값에 도달 했을때 다시 최소값 부터 시작함.

시퀀스 확인

SELECT * FROM user_sequences;

```
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
1	DEPARTMENTS_SEQ	1	9990	10	280
2	DEPTS_SEQ	1	100	1	91
3	EMPLOYEES_SEQ	1	999999999999999999999999999	1	207
4	LOCATIONS_SEQ	1	9900	100	3300

- NEXTVAL은 다음 사용 가능한 시퀀스 값을 리턴합니다.
- 시퀀스가 참조될 때마다, 다른 사용자에게 조차도 유일한 값을 반환합니다.
- CURRVAL은 현재 시퀀스 값을 리턴합니다.
- CURRVAL이 참조되기 전에 NEXTVAL이 먼저 이용되어야 합니다.

NEXTVAL 의사열은 지정된 시퀀스에서 성공적으로 시퀀스 번호를 빼내기 위해 사용됩니다. 시퀀스 명으로써 NEXTVAL을 참조합니다. *sequence*.NEXTVAL을 참조할 때는 새 시퀀스 번호가 생성되고 현재 시퀀스 번호는 CURRVAL에 위치하게 됩니다.

CURRVAL 의사열은 막 생성된 현재 사용자의 시퀀스 번호를 참조하기 위해 사용됩니다. NEXTVAL은 CURRVAL이 참조될 수 있기 전에 현재 사용자의 세션에서 시퀀스 번호를 생성하기 위해 사용되어야 합니다. *sequence*.CURRVAL이 참조되면 사용자의 프로세스에 사용된 최후 시퀀스 값이 디스플레이 됩니다.

다음 구문은 DEPTS 테이블에 새로운 부서 정보를 삽입합니다.

```
SQL> INSERT INTO depts(deptno, dname, loc)
  2  VALUES (depts_seq.NEXTVAL, 'MARKETING', 'SAN DIEGO');

1 행 이(가) 삽입되었습니다.
```

입력된 행을 조회해 봅니다.

```
SQL> SELECT * FROM depts;
```

	DEPTNO	DNAME	LOC
1	91	MARKETING	SAN DIEGO

다음 구문은 DEPTS_SEQ 시퀀스에 대한 현재 값을 봅니다.

```
SQL> SELECT depts_seq.CURRVAL
  2  FROM   dual;
```

	CURRVAL
1	91

시퀀스에 대해 MAXVALUE 한계에 도달한다면, 더 이상 시퀀스 값을 할당 받을 수 없을 것이고, MAXVALUE를 초과하는 시퀀스를 나타내는 에러를 받을 것입니다. 시퀀스를 계속 사용하기 위해, ALTER SEQUENCE 문장을 사용하여 수정할 수 있습니다.

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

다음 구문의 예는 DEPTS_SEQ 시퀀스의 최댓값을 99999로 변경합니다.

```
SQL> ALTER SEQUENCE depts_seq
  2        MAXVALUE 99999;

Sequence DEPTS_SEQ이(가) 변경되었습니다.
```

Chapter 10

SEQUENCE (순차적으로 증가하는 값)
INDEX

- 테이블이나 클러스터에서 쓰이는 선택적인 객체입니다.
- 오라클 데이터베이스 테이블내의 원하는 레코드를 빠르게 찾아갈 수 있도록 만들어진 데이터 구조입니다.
- 포인터를 사용하여 행의 검색을 촉진하기 위해 오라클 서버가 사용합니다.
- 빠르게 데이터를 찾기 위해 빠른 경로 액세스 방법을 사용하여 디스크 I/O를 경감시킵니다.
- 자동 인덱스는 Primary Key 또는 Unique 제한 규칙에 의해 자동적으로 생성되는 인덱스들입니다.
- 수동 인덱스는 CREATE INDEX 명령을 실행해서 만드는 인덱스들입니다.

- 자동으로 : 유일 인덱스는 테이블 정의의 PRIMARY KEY 또는 UNIQUE 키 제약조건을 정의할 때 자동으로 생성됩니다.
- 수동으로 : 사용자는 행에 대한 액세스 시간을 향상시키기 위해 열에서 유일하지 않은 인덱스를 생성할 수 있습니다.

```
CREATE [UNIQUE ¦ BITMAP] INDEX index_name
ON table_name (column1[, column2]...);
```

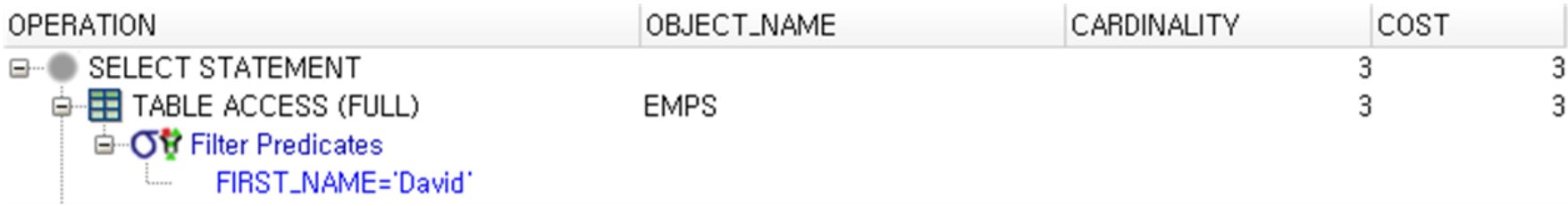
구문 형식에서...

- *index_name* : 인덱스의 이름입니다.
- *table_name* : 테이블의 이름입니다.
- *column* : 인덱스 되기 위한 테이블의 열 이름입니다.

다음 구문은 EMPS 테이블에서 FIRST_NAME 열을 이용해 데이터를 조회합니다.

```
SQL> SELECT * FROM emps WHERE first_name='David';
```

다음 그림은 위 구문을 실행했을 때 실행계획입니다. SQL Developer에서 실행결과를 보기 위한 단축키는 F10입니다. 인덱스가 없는 열을 기준으로 데이터를 검색할 때 FULL 스캔되는 것을 볼 수 있습니다.



다음 구문은 FIRST_NAME 열에 인덱스를 생성합니다.

```
SQL> CREATE INDEX emps_first_name_idx
  2  ON           emps(first_name);

Index EMPS_FIRST_NAME_IDX이(가) 생성되었습니다.
```

다음 그림은 인덱스를 생성한 후 실행계획입니다. 테이블 액세스가 FULL 스캔이 아니고 INDEX ROWID에 의한 것임을 볼 수 있습니다.

OPERATION	OBJECT_NAME	CARDINALITY	COST
SELECT STATEMENT		3	2
TABLE ACCESS (BY INDEX ROWID)	EMPS	3	2
INDEX (RANGE SCAN)	EMPS_FIRST_NAME_IDX	1	1
Access Predicates			
FIRST_NAME='David'			

인덱스의 구조는 테이블과 독립적이므로 인덱스의 삭제는 테이블의 데이터에는 아무런 영향도 미치지 않습니다. 인덱스를 삭제하려면 INDEX의 소유자이거나 [DROP ANY INDEX] 권한을 가지고 있어야 합니다.

INDEX는 ALTER 할 수 없습니다.

```
SQL> DROP INDEX emps_first_name_idx;
```

테이블의 많은 인덱스가 반드시 질의의 응답속도 향상을 의미하는 것은 아닙니다. 인덱스를 가지고 있는 테이블에 대한 각 DML 작업은 인덱스도 갱신되어야 함을 의미합니다. 많은 인덱스가 테이블과 관련되어 있으면, 오라클 서버는 DML 후에 모든 인덱스를 갱신시키기 위해 더 많은 노력이 필요하게 됩니다.

인덱스는 다음 경우에 생성하는 것을 권장합니다.

- 열이 WHERE 절 또는 조인조건에서 자주 사용될 경우
- 열이 광범위한 값을 포함할 경우
- 열이 많은 수의 Null 값을 포함할 경우
- 둘 또는 이상의 열이 WHERE 절 또는 조인조건에서 자주 함께 사용될 경우
- 테이블은 대형이고 대부분 질의가 행의 2~4%보다 적게 읽을 것으로 예상할 경우

다음 경우에는 언제 인덱스를 생성하지 않기를 권장합니다.

- 테이블이 작을 경우
- 열이 질의의 조건으로 자주 사용되지 않을 경우
- 대부분 질의가 행의 2~4% 이상 읽을 것으로 예상할 경우
- 테이블은 자주 갱신될 경우 테이블에 하나 이상 인덱스를 가지고 있다면 테이블을 액세스하는 DML 문장은 인덱스의 유지 때문에 상대적으로 더 많은 시간이 걸리게 됩니다.

- **Unique 인덱스**
  - **Primary Key와 Unique 제약 조건 시 자동 생성됩니다.**
  - **중복된 값이 입력될 수 없습니다.**

다음 구문은 Unique 인덱스를 생성합니다.

```
SQL> CREATE UNIQUE INDEX emps_email_idx ON emps(email);

Unique index EMPS_EMAIL_IDX이(가) 생성되었습니다.
```

Unique 인덱스 생성 후 EMPS 테이블에서 EMAIL 열을 이용해 데이터를 조회한 다음 실행계획을 확인합니다.

```
SQL> SELECT * FROM emps WHERE email='DAUSTIN';
```

위 구문은 UNIQUE 스캔이 일어납니다.

OPERATION	OBJECT_NAME	CARDINALITY	COST
SELECT STATEMENT		1	1
TABLE ACCESS (BY INDEX ROWID)	EMPS	1	1
INDEX (UNIQUE SCAN)	EMPS_EMAIL_IDX	1	0
Access Predicates			
EMAIL='DAUSTIN'			