ch13. 객체 종류

13-1. 데이터베이스를 위한 데이터를 저장한 데이터 사전

데이터 사전이란?

오라클 데이터베이스 테이블은 사용자 테이블 *user table* 과 데이터 사전 *data dictionary* 으로 나뉜다.

- 사용자 테이블 : 테이터베이스를 통해 관리할 데이터를 저장하는 테이블
- 데이터 사전 : 데이터베이스를 구성하고 운영하는 데 필요한 모든 정보를 저장하는 특수한 테이블로 데이터베이스가 생성되는 시점에 자동으로 만들어진다.
  - 데이터 사전에는 데이터베이스 메모리, 성능, 사용자, 권한, 객체 등 오라클 데이터베이스 운영에 중요한 데이터가 보관되어있다.

  ⇒ 오라클 데이터 베이스는 사용자가 데이터 사전 정보에 직접 접군하거나 작업하는 것을 허용하지 않는다. 그 대신 데이터 사전 뷰 *data dictionary view* 를 제공하여 SELECT문으로 정보를 열람할 수 있게 했다.

데이터 사전 뷰

: 용도에 따라 이름 앞에 접두어를 지정하여 분류

접두어	설명
USER_XXXX	현재 데이터베이스에 접속한 사용자가 소유한 객체 정보
ALL_XXXX	현재 데이터베이스에 접속한 사용자가 소유한 객체 또는 다른 사용자가 소유한 객체 중 사용 허가를 받은 객체, 즉 사용 가능한 모든 객체 정보
DBA_XXXX	데이터베이스 관리를 위한 정보 (데이터베이스 관리 권한을 가진 SYSTEM, SYS 사용자만 열람 가능)
V$_XXXX	데이터베이스 성능 관련 정보 (X$_XXXX 테이블의 뷰)

USER_ 접두어를 가진 사전

→ 현재 오라클에 접속해 있는 사용자가 소유한 객체 정보가 보관되어 있다.

⇒ 접두어 뒤엔 복수형이 온다.

- SCOTT계정이 가지고 있는 객체 정보 살펴보기

```
SELECT TABLE_NAME
  FROM USER_TABLES;
```

	TABLE_NAME
1	DEPT
2	EMP
3	BONUS
4	SALGRADE
5	DEPT_TEMP
6	EMP_TEMP
7	DEPT_TEMP2
8	EMP_TEMP2
9	CHAP10HW_EMP
10	CHAR10HW_DEPT
11	CHAR10HW_SALGRADE
12	CHAR10HW_DEPTT
13	DEPT_TCL
14	EMP_DDL

ALL_ 접두어를 가진 데이터 사전

→ 오라클 데이터베이스에 접속해 있는 사용자가 소유한 객체 및 다른 사용자가 소유한 객체 중 사용이 허락되어 있는 객체 정보를 가지고 있다..

⇒ 접두어 뒤에 객체를 명시할 때 복수형 단어를 사용한다.

```
SELECT OWNER, TABLE_NAME
  FROM ALL_TABLES;
```

DBA_ 접두어를 가진 데이터 사전

→ 데이터베이스 관리 권한을 가진 사용자만 조회할 수 있는 테이블로서
SCOTT 계정으로는 조회가 불가능하다.

```
SELECT * FROM DBA_TABLES;
```

⇒ SCOTT계정으론 조회할 수 없지만, 데이터베이스 권한이 있는
SYSTEM 사용자로 접속하면 조회가 가능하다.

13-2. 더 빠른 검색을 위한 인덱스

> 인덱스란?

인덱스 *index* : 오라클 데이터베이스에서 데이터 검색 성능의 향상을 위해
테이블 열에 사용하는 객체

- 테이블에 보관된 특정 행 데이터의 주소, 즉 위치 정보를 책 페이지처럼 목록으로 만들어 놓은 것.
- 인덱스 테이블 열을 여러 가지 분석을 통해 선정하여 설정할 수 있다.

- Table Full Scan
  - 테이블 데이터를 처음부터 끝까지 검색하여 원하는 데이터를 찾는 방식
- Index Scan
  - 인덱스를 통해 데이터를 찾는 방식

- SCOTT계정이 소유한 인덱스 정보 알아보기

```
SELECT *
  FROM USER_INDEXES;
```

- SCOTT 계정이 소유한 인덱스 컬럼 정보 알아보기

```
SELECT *
  FROM USER_IND_COLUMNS;
```

⇒ 인덱스는 사용자가 직접 특정 테이블의 열에 지정할 수도 있지만 열이 기본키, 고유키일 경우에 자동으로 생성된다.

인덱스 생성

오라클에서 자동으로 생성해 주는 인덱스 외에 사용자가 직접 인덱스를 만들 때 CREATE문 사용.

```
CREATE INDEX 인덱스 이름
  ON 테이블 이름(열 이름1 ASC or DESC,
                 열 이름2 ASC or DESC,
                 ...                  );
```

- EMP 테이블의 SAL 열에 인덱스 생성

```
CREATE INDEX IDX_EMP_SAL
  ON EMP(SAL);
```

- 생성된 인덱스 살펴보기 (USER_IND_COLUMNS 사용)

```
SELECT * FROM USER_IND_COLUMNS;
```

인덱스 삭제

인덱스 삭제는 DROP 명령어 사용.

```
DROP INDEX 인덱스 이름;
```

- EMP테이블의 SAL열에 생성한 IDX_EMP_SAL인덱스 삭제하기

```
DROP INDEX IDX_EMP_SAL;
```

📎 * 인덱스는 데이터 접근 및 검색 속도 향상을 위해 사용하는 객체이지만 인덱스 생성이 항상
좋은 결과로 이어지진 않는다.
* 정확한 데이터 분석에 기반을 두지 않은 인덱스의 무분별한 생성은 오히려 성능을
떨어뜨리는 원인이 되기도 한다.
* 인덱스는 데이터 종류 분포도, 조회하는 SQL의 구성, 데이터 조작 관련 SQL문의 작업 빈도,
검색 결과가 전체 데이터에서 차지하는 비중 등 많은 요소를 고려하여 생성해야 한다.

13-3. 테이블처럼 사용하는 뷰

뷰란?

가상 테이블 *virtual table* 로 부르는 뷰 *view*는 하나 이상의 테이블을 조회하는 SELECT문을 저장한 객체를 뜻한다.

⇒ 뷰를 SELECT문의 FROM절에 사용하면 특정 테이블을 조회하는 것과 같은 효과를 얻을 수 있다.

뷰의 사용목적

- 편리성 : SELECT문의 복잡도를 완화하기 위해
- 보안성 : 테이블의 특정 열을 노출하고 싶지 않을 경우

뷰 생성

CREATE 문으로 생성.

- SCOTT 계정은 뷰 권한이 없으므로 SYSTEM계정으로 접속한 후 다음 명령어를 사용하여 SCOTT계정에 권한을 부여해줘야 한다.

- 명령창에서 권한 부여

`$SQLPLUS SYSTEM/oracle`

`$GRANT CREATE VIEW TO SCOTT;`

- 권한을 부여한 후 뷰를 생성한다.

```
CREATE [OR REPLACE] [FORCE | NOFORCE] VIEW 뷰 이름 (열 이름1, 열 이름2, ...)
  AS (저장할 SELECT문)
[WITH CHECK OPTION [CONSTRAINT 제약 조건]]
[WITH READ ONLY [CONSTRAINT 제약 조건]];
```

OR REPLACE	같은 이름의 뷰가 이미 존재할 경우에 현재 생성할 뷰로 대체하여 생성 (선택)
FORCE	뷰가 저장할 SELECT문의 기반 테이블이 존재하지 않아도 강제로 생성 (선택)
NOFORCE	뷰가 저장할 SELECT문의 기반 테이블이 존재할 경우에만 생성 (기본값) (선택)
뷰 이름	뷰 이름을 지정 (필수)
열 이름	SELECT문에 명시된 이름 대신 사용할 열 이름 지정 (생성 가능) (선택)
저장할 SELECT문	생성할 뷰에 저장할 SELECT문 지정 (필수)
WITH CHECK OPTION	지정한 제약 조건을 만족하는 데이터에 한에 DML작업이 가능하도록 뷰 생성 (선택)
WITH READ ONLY	뷰의 열람, 즉 SELECT만 가능하도록 뷰 생성 (선택)

- VW_EMP20 뷰 생성

```
CREATE VIEW VW_EMP20
  AS (SELECT EMPNO,ENAME,JOB,DEPTNO
        FROM EMP
       WHERE DEPTNO = 20);
```

- 생성한 뷰 내용 확인하기

```
$SELECT VIEW_NAME, TEXT_LENGTH, TEXT
 FROM USER_VIEWS;
```

- 생성한 뷰 조회하기

```
SELECT * FROM VW_EMP20;
```

뷰 삭제

```
DROP VIEW VW_EMP20;
```

생성/삭제한 뷰 확인하기

```
SELECT *
  FROM USER_VIEWS;
```

→ 뷰는 실제 데이터가 아닌 SELECT문만 저장하므로 뷰를 삭제해도 테이블이나 데이터가 삭제되는 것은 아니다.

인라인 뷰를 사용한 TOP-N SQL문

- 인라인 뷰 *inline view* 는 CREATE문을 통해 객체로 만들어지는 뷰 외에 SQL문에서 일회성으로 만들어서 사용하는 뷰.

- 인라인 뷰와 ROWNUM을 사용하면 ORDER BY절을 통해 정렬된 결과 중 최상위 몇 개 데이터만을 출력하는 것이 가능하다.

```
SELECT ROWNUM, E.*
  FROM EMP E;
```

ROWNUM : 의사 열 *pseudo column* 이라고 하는 특수 열.

→ 의사 열은 데이터가 저장되는 실제 테이블에 존재하지는 않지만 특정 목적을 위해 테이블에 저장되어 있는 열처럼 사용 가능한 열을 뜻한다.

→ ROWNUM 열 데이터 번호는 테이블에 저장된 행이 조회된 순서대로 매겨진 일련번호.

- EMP 테이블을 SAL 열 기준으로 정렬하기

```
SELECT ROWNUM, E.*
  FROM EMP E
ORDER BY SAL DESC;
```

	ROWNUM	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	9	7839	KING	PRESIDENT	(null)	81/11/17	5000	(null)	10
2	13	7902	FORD	ANALYST	7566	81/12/03	3000	(null)	20
3	8	7788	SCOTT	ANALYST	7566	87/04/19	3000	(null)	20
4	4	7566	JONES	MANAGER	7839	81/04/02	2975	(null)	20
5	6	7698	BLAKE	MANAGER	7839	81/05/01	2850	(null)	30
6	7	7782	CLARK	MANAGER	7839	81/06/09	2450	(null)	10
7	2	7499	ALLEN	SALESMAN	7698	81/02/20	1600	300	30
8	10	7844	TURNER	SALESMAN	7698	81/09/08	1500	0	30
9	14	7934	MILLER	CLERK	7782	82/01/23	1300	(null)	10
10	3	7521	WARD	SALESMAN	7698	81/02/22	1250	500	30
11	5	7654	MARTIN	SALESMAN	7698	81/09/28	1250	1400	30
12	11	7876	ADAMS	CLERK	7788	87/05/23	1100	(null)	20
13	12	7900	JAMES	CLERK	7698	81/12/03	950	(null)	30
14	1	7369	SMITH	CLERK	7902	80/12/17	800	(null)	20

→ 급여 기준으로 정렬을 했지만, 앞에서 사용한 SELECT문의 행 번호와 같은 번호로 매겨져 있다.

→ ROWNUM은 데이터를 하나씩 추가할 때 매겨지는 번호!!

- 이러한 특성을 인라인 뷰에서 적용하면 정렬된 SELECT문의 결과 순번을 매겨서 출력이 가능.

```
SELECT ROWNUM, E.*
  FROM (SELECT *
          FROM EMP E
        ORDER BY SAL DESC) E;
```

```
WITH E AS (SELECT * FROM EMP ORDER BY SAL DESC)
SELECT ROWNUM, E.*
  FROM E;
```

	ROWNUM	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	1	7839	KING	PRESIDENT	(null)	81/11/17	5000	(null)	10
2	2	7902	FORD	ANALYST	7566	81/12/03	3000	(null)	20
3	3	7788	SCOTT	ANALYST	7566	87/04/19	3000	(null)	20
4	4	7566	JONES	MANAGER	7839	81/04/02	2975	(null)	20
5	5	7698	BLAKE	MANAGER	7839	81/05/01	2850	(null)	30
6	6	7782	CLARK	MANAGER	7839	81/06/09	2450	(null)	10
7	7	7499	ALLEN	SALESMAN	7698	81/02/20	1600	300	30
8	8	7844	TURNER	SALESMAN	7698	81/09/08	1500	0	30
9	9	7934	MILLER	CLERK	7782	82/01/23	1300	(null)	10
10	10	7521	WARD	SALESMAN	7698	81/02/22	1250	500	30
11	11	7654	MARTIN	SALESMAN	7698	81/09/28	1250	1400	30
12	12	7876	ADAMS	CLERK	7788	87/05/23	1100	(null)	20
13	13	7900	JAMES	CLERK	7698	81/12/03	950	(null)	30
14	14	7369	SMITH	CLERK	7902	80/12/17	800	(null)	20

- 급여가 높은 상위 세 명의 데이터만 출력하기

```
SELECT ROWNUM, E.*
  FROM (SELECT *
          FROM EMP E
         ORDER BY SAL DESC) E
WHERE ROWNUM <= 3;
```

```
WITH E AS (SELECT * FROM EMP ORDER BY SAL DESC)
SELECT ROWNUM, E.*
  FROM E
WHERE ROWNUM <= 3;
```

	ROWNUM	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	1	7839	KING	PRESIDENT	(null)	81/11/17	5000	(null)	10
2	2	7788	SCOTT	ANALYST	7566	87/04/19	3000	(null)	20
3	3	7902	FORD	ANALYST	7566	81/12/03	3000	(null)	20

13-4. 규칙에 따라 순번을 생성하는 시퀀스

시퀀스란?

시퀀스 *sequence* 는 오라클 데이터베이스에서 특정 규칙에 맞는 연속 숫자를 생성하는 객체.

시퀀스 생성

CREATE문으로 생성한다.

```
CREATE SEQUENCE 시퀀스 이름   (1)
[INCREMENT BY n]             (2)
[START WITH n]               (3)
[MAXVALUE n | NOMAXVALUE]    (4)
[MINVALUE n | NOMINVALUE]    (5)
[CYCLE | NOCYCLE]            (6)
[CACHE n | NOCACHE]          (7)
```

1	생성할 시퀀스 이름 지정, 아래 절들을 지정않는 경우 1부터 시작하여 1만큼 계속 증가하는 시퀀스 생성 (필수)
2	시퀀스에서 생성하 번호의 증가 값 (기본값은 1) (선택)
3	시퀀스에서 생성할 번호의 시작 값 (기본값은 1) (선택)
4	시퀀스에서 생성할 번호의 최댓값 지정, 최댓값은 시작 값(START WITH) 이상, 최소값(MINVALUE)을 초괏값으로 지정. NOMAXVALUE로 지정하였을 경우 오름차순이면 10^27, 내림차순이면 -1으로 설정 (선택)
5	시퀀스에서 생성할 번호의 최솟값 지정, 최솟값은 시작 값(START WITH) 이하, 최댓값(MAXVALUE)미만 값으로 지정. NOMINVALUE로 지정하였을 경우 오름차순이면 1, 내림차순이면 -10^26으로 설정 (선택)
6	시퀀스에서 생성한 번호가 최댓값(MAXVALUE)에 도달했을 경우 CYCLE이면 시작 값 (START WITH)에서 다시 시작, NOCYCLE이면 번호생성이 중단되고, 추가 번호 생성을 요청하면 오류 발생 (선택)
7	시퀀스가 생성할 번호를 메모리에 미리 할당해 놓은 수를 지정, NOCACHE는 미리 생성하지 않도록 설정옵션을 모두 생략하면 기본값은 20 (선택)

- DEPT 테이블을 사용하여 DEPT_SEQUENCE 테이블 생성하기

```
CREATE TABLE DEPT_SEQUENCE
    AS SELECT *
```

```
          FROM DEPT
         WHERE 1 <> 1;
```

- 시퀀스 생성하기

```
CREATE SEQUENCE SEQ_DEPT_SEQUENCE
   INCREMENT BY 10
   START WITH 10
   MAXVALUE 90
   MINVALUE 0
   NOCYCLE
   CACHE 2;
```

시퀀스 사용

생성된 시퀀스를 사용할 때는 [시퀀스 이름. CURRVAL] 과 [시퀀스 이름. NEXTVAL]을 사용.

CURRVAL 은 시퀀스에서 마지막으로 생성한 번호를 반환

→ 시퀀스는 생성하고 바로 사용하면 번호가 만들어진 적이 없으므로 오류가 난다.

NEXTVAL 는 다음 번호를 생성

- 시퀀스에서 생성한 순번을 사용한 INSERT 문 실행

```
INSERT INTO DEPT_SEQUENCE (DEPTNO, DNAME, LOC)
VALUES (SEQ_DEPT_SEQUENCE.NEXTVAL, 'DATABASE', 'SEOUL');

SELECT * FROM DEPT_SEQUENCE ORDER BY DEPTNO;
```

- 가장 마지막으로 생성된 시퀀스 확인하기

```
SELECT SEQ_DEPT_SEQUENCE.CURRVAL
  FROM DUAL;
```

INSERT문을 9번 실행하면 부서번호가 90번까지 이르는데 그 후 다시 실행하면 최댓값이 이미 생성되었고 NOCYCLE옵션으로 순환되지 않도록 설정하였으므로 오류가 난다.

- 시퀀스에서 생성한 순번을 반복 사용하여 INSERT문 실행

```
INSERT INTO DEPT_SEQUENCE (DEPTNO, DNAME, LOC)
VALUES (SEQ_DEPT_SEQUENCE.NEXTVAL, 'DATABASE', 'SEOUL');

SELECT * FROM DEPT_SEQUENCE ORDER BY DEPTNO;
```

시퀀스 수정

ALTER 명령어로 시퀀스를 수정하고 DROP명령어로 시퀀스를 삭제한다.

START WITH 값은 변경할 수 없다.

```
ARTER SEQUENCE 시퀀스 이름
[INCREMENT BY n]
[MAXVALUE n | NOMAXVALUE]
[MINVALUE n | NOMINVALUE]
[CYCLE | NOCYCLE]
[CACHE n | NOCACHE]
```

- 시퀀스 옵션 수정하기

```
ALTER SEQUENCE SEQ_DEPT_SEQUENCE
   INCREMENT BY 3
   MAXVALUE 99
   CYCLE;
```

- 수정한 시퀀스를 사용하여 INSERT문 조회하기

```
INSERT INTO DEPT_SEQUENCE (DEPTNO, DNAME, LOC)
VALUES (SEQ_DEPT_SEQUENCE.NEXTVAL, 'DATABASE', 'SEOUL');

SELECT * FROM DEPT_SEQUENCE ORDER BY DEPTNO;
```

- CYCLE 옵션을 사용한 시퀀스의 최댓값 도달 후 수행 결과 확인하기

```
INSERT INTO DEPT_SEQUENCE (DEPTNO, DNAME, LOC)
VALUES (SEQ_DEPT_SEQUENCE.NEXTVAL, 'DATABASE', 'SEOUL');

SELECT * FROM DEPT_SEQUENCE ORDER BY DEPTNO;
```

시퀀스 삭제

```
DROP SEQUENCE SEQ_DEPT_SEQUENCE;

SELECT * FROM USER_SEQUENCES;
```

13-5. 공식 별칭을 지정하는 동의어

동의어란?

동의어 synonym 는 테이블, 뷰, 시퀀스 등 객체 이름 대신 사용할 수 있는 다른 이름을 부여하는 객체

```
CREATE [PUBLIC] SYNONYM 동의어 이름
FOR [사용자.][객체이름];
```

PUBLIC	동의어를 데이터베이스 내 모든 사용자가 사용할 수 있도록 설정. 생략할 경우 동의어를 생성한 사용자만 사용 가능 (선택)
동의어 이름	동의어 이름 (필수)
사용자.	생성할 동의어의 본래 객체 소유 사용자를 지정. 생략할 경우 현재 접속한 사용자로 지정 (선택)
객체이름	생성할 대상 객체 이름 (필수)

- 권한 부여하기

`$SQLPLUS SYSTEM/oracle`

`$GRANT CREATE SYNONYM TO SCOTT;`

`$GRANT CREATE PUBLIC SYNONYM TO SCOTT;`

동의어 생성

- EMP 테이블의 동의어 생성

```
CREATE SYNONYM E
   FOR EMP;
```

- E 테이블 전체 내용 조회하기

```
SELECT * FROM E;
```

동의어 삭제

```
DROP SYNONYM E;
```

Q

1.

```
CREATE TABLE EMPIDX
AS SELECT *
FROM EMP;
```

```
CREATE INDEX IDX_EMPIDX_EMPNO
    ON EMPIDX (EMPNO);
```

```
SELECT *
  FROM USER_INDEXES
 WHERE INDEX_NAME = 'IDX_EMPIDX_EMPNO';
```

2.

```
CREATE OR REPLACE VIEW EMPIDX_OVER15K
    AS (SELECT EMPNO, ENAME, JOB, DEPTNO,
               SAL, NVL2(COMM, 'O', 'X') AS COMM
          FROM EMPIDX
         WHERE SAL > 1500);
```

3.

```
CREATE TABLE DEPTSEQ
    AS SELECT *
         FROM DEPT;
```

```
CREATE SEQUENCE SEQ_DEPTSEQ
   INCREMENT BY 1
   START WITH 1
   MAXVALUE 99
   MINVALUE 1
   NOCYCLE NOCACHE;
```

```
INSERT INTO DEPTSEQ (DEPTNO, DNAME, LOC)
VALUES (SEQ_DEPTSEQ.NEXTVAL, 'DATABASE', 'SEOUL');

INSERT INTO DEPTSEQ (DEPTNO, DNAME, LOC)
VALUES (SEQ_DEPTSEQ.NEXTVAL, 'WEB', 'BUSAN');

INSERT INTO DEPTSEQ (DEPTNO, DNAME, LOC)
VALUES (SEQ_DEPTSEQ.NEXTVAL, 'MOBILE', 'ILSAN');
```