

객체 종류

오라클 데이터베이스는 데이터 보관 및 관리를 위한 여러 기능과 저장 공간을 객체를 통해 제공합니다. 테이블은 SQL문과 더불어 오라클에서 가장 많이 사용하는 객체 중 하나입니다. 이 장에서는 앞에서 다른 테이블 외에 데이터 사전(data dictionary), 인덱스(index), 뷰(view), 시퀀스(sequence), 동의어(synonym) 등 사용빈도가 높은 객체의 사용법을 간단히 소개하겠습니다.

13-1 데이터베이스를 위한 데이터를 저장한 데이터 사전

13-2 더 빠른 검색을 위한 인덱스

13-3 테이블처럼 사용하는 뷰

13-4 규칙에 따라 순번을 생성하는 시퀀스

13-5 공식 별칭을 지정하는 동의어

이 장에서 꼭 익혀야 할 것

- USER_, ALL_ 접두어를 가진 데이터 사전 뷰의 의미와 사용 방법
- 인덱스 의미와 생성 방법
- 뷰 생성 방법과 사용 이유
- 시퀀스 사용 방법

13-1 데이터베이스를 위한 데이터를 저장한 데이터 사전

데이터 사전이란?

오라클 데이터베이스 테이블은 사용자 테이블(user table)과 데이터 사전(data dictionary)으로 나뉩니다. 사용자 테이블은 데이터베이스를 통해 관리할 데이터를 저장하는 테이블을 뜻합니다. 앞에서 여러 SQL문에서 활용한 EMP, DEPT, SALGRADE 테이블이 바로 사용자 테이블이죠. 데이터 사전은 데이터베이스를 구성하고 운영하는 데 필요한 모든 정보를 저장하는 특수한 테이블로 데이터베이스가 생성되는 시점에 자동으로 만들어집니다.

☞ 사용자 테이블은 Normal Table, 데이터 사전은 Base Table이라고 부르기도 합니다.

데이터 사전에는 데이터베이스 메모리·성능·사용자·권한·객체 등 오라클 데이터베이스 운영에 중요한 데이터가 보관되어 있습니다. 만약 이 데이터에 문제가 발생한다면 오라클 데이터베이스 사용이 불가능해질 수도 있습니다.

☞ 우리가 자주 사용하는 컴퓨터 윈도우로 비유하자면 사용자 테이블은 윈도우에서 사용할 응용 프로그램, 즉 윈도우 동작 자체와는 무관한 프로그램이 저장된 Program Files 폴더, 데이터 사전은 윈도우 구동을 위한 파일들로 이루어진 Windows 폴더인 것이죠.

따라서 오라클 데이터베이스는 사용자가 데이터 사전 정보에 직접 접근하거나 작업하는 것을 허용하지 않습니다. 그 대신 데이터 사전 뷰(data dictionary view)를 제공하여 SELECT 문으로 정보 열람을 할 수 있게 해 두었습니다.

☞ 뷰(view)는 어떤 목적을 위해 테이블 일부 또는 전체 데이터 열람을 주 목적으로 사용하는 객체를 뜻합니다.

데이터 사전 뷰는 용도에 따라 이름 앞에 다음과 같은 접두어를 지정하여 분류합니다.

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)

사용 가능한 데이터 사전을 알고 싶다면 다음과 같이 DICTIONARY 또는 DICT를 조회합니다. 종류가 꽤 많으므로 자주 사용하는 몇 개 정도만 알아 두고 그 밖의 것들은 필요할 때 찾아서 사용하면 됩니다.

실습 13-1 SCOTT 계정에서 사용 가능한 데이터 사전 살펴보기(DICT 사용)

```
01 SELECT * FROM DICT;
```

실습 13-2 SCOTT 계정에서 사용 가능한 데이터 사전 살펴보기(DICTIONARY 사용)

```
01 SELECT * FROM DICTIONARY;
```

:: 결과 화면(실습 13-1, 13-2의 실행 결과가 같음)

TABLE_NAME	COMMENTS
USER_AW_PROP	Object properties in Analytic Workspaces owned by the user
USER_AW_PS	Pagespaces in Analytic Workspaces owned by the user
USER_BASE_TABLE_MVIEWS	All materialized views with log(s) owned by the user in the database
USER_CATALOG	Tables, Views, Synonyms and Sequences owned by the user
USER_CHANGE_NOTIFICATION_REGS	change notification registrations for current user
USER_CLUSTERS	Descriptions of user's own clusters
USER_CLUSTER_HASH_EXPRESSIONS	Hash functions for the user's hash clusters
USER_CLU_COLUMNS	Mapping of table columns to cluster columns
USER_COLL_TYPES	Description of the user's own named collection types
USER_COL_COMMENTS	Comments on columns of user's tables and views
USER_COL_PENDING_STATS	Pending statistics of tables, partitions, and subpartitions
USER_COL_PRIVS	Grants on columns for which the user is the owner, grantor or grantee
USER_COL_PRIVS_MADE	All grants on columns of objects owned by the user
▶ USER_COL_PRIVS_REC0	Grants on columns for which the user is the grantee

데이터 사전 일부만 표시했습니다.

USER_ 접두어를 가진 데이터 사전

USER_ 접두어로 시작하는 이름의 데이터 사전에는 현재 오라클에 접속해 있는 사용자가 소유한 객체 정보가 보관되어 있습니다. 예를 들어 현재 오라클 데이터베이스에 접속해 있는 SCOTT 계정이 소유하는 테이블 정보는 USER_TABLES를 사용합니다.

실습 13-3 SCOTT 계정이 가지고 있는 객체 정보 살펴보기(USER_ 접두어 사용)

```
01 SELECT TABLE_NAME  
02     FROM USER_TABLES;
```

:: 결과 화면(일부 데이터만 표시함)

TABLE_NAME
▶ DEPT
EMP
BONUS
SALGRADE
TEST
DEPT_TEMP
EMP_TEMP
DEPT_TEMP2
EMP_TEMP2
EMP_TCL
DEPT_TCL
EMP_DDL
DEPT_DDL
EMP_DDL_30
EMPDEPT_DDL



SCOTT 계정이 가지고 있는 테이블
이름을 알고 싶을 때 유용합니다.

SCOTT 계정이 처음부터 소유한 테이블과 데이터 정의어를 사용하면서 생성한 테이블 정보를 조회합니다.

접두어 뒤에 복수형 단어로 이름을 구성하고 있음을 눈여겨보세요. 앞으로 살펴볼 USER_, ALL_, DBA_ 접두어 다음에 객체가 오면 복수형 단어가 오게 됩니다. 실습 13-3에서 확인할 수 있듯이 접두어 USER_ 다음에 TABLE이 아닌 TABLES가 나오고 있습니다.

ALL_ 접두어를 가진 데이터 사전

ALL_ 접두어를 가진 데이터 사전은 오라클 데이터베이스에 접속해 있는 사용자가 소유한 객체 및 다른 사용자가 소유한 객체 중 사용이 허락되어 있는 객체 정보를 가지고 있습니다. 즉 SCOTT 계정으로 접속하여 ALL_TABLES를 조회하면 SCOTT 계정이 사용할 수 있는 테이블 정보를 모두 보여 줍니다. ALL_ 접두어의 데이터 사전 역시 뒤에 객체를 명시할 때 복수형 단어를 사용합니다.

실습 13-4 SCOTT 계정이 사용할 수 있는 객체 정보 살펴보기(ALL_ 접두어 사용)

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

:: 결과 화면(일부 데이터만 표시함)

OWNER	TABLE_NAME
SYS	DUAL
SYS	SYSTEM_PRIVILEGE_MAP
SYS	TABLE_PRIVILEGE_MAP
SYS	STMT_AUDIT_OPTION_MAP
SYS	AUDIT_ACTIONS
SYS	WRR\$REPLAY_CALL_FILTER
SYS	HS_BULKLOAD_VIEW_OBJ
SYS	HS\$PARALLEL_METADATA
SYS	HS_PARTITION_COL_NAME
SYS	HS_PARTITION_COL_TYPE
SYSTEM	HELP
CTXSYS	DR\$OBJECT_ATTRIBUTE
CTXSYS	DR\$POLICY_TAB
CTXSYS	DR\$THS
CTXSYS	DR\$THS_PHRASE
CTXSYS	DR\$NUMBER_SEQUENCE
XDB	XDB\$XIDX_IMP_T



ALL_ 접두어를 사용하면 사용 가능
한 테이블이 모두 출력됩니다.

ALL_TABLES에는 USER_TABLES와 달리 OWNER 열이 하나 더 있습니다. 이 열은 테이블을 소유한 사용자를 명시합니다. 앞장에서 사용한 DUAL 테이블은 오라클 관리 계정 SYS 소유이고 SCOTT 계정은 이 테이블의 사용을 허가받은 것입니다.

OWNER 열 이외의 열은 USER_TABLES, ALL_TABLES 모두 동일한 열 구조를 가집니다. 다음은 USER_TABLES와 ALL_TABLES의 열 일부입니다.

열 이름	자료형	NULL 여부	설명
OWNER	VARCHAR2(30)	NOT NULL	테이블을 소유한 사용자 (ALL_TABLES에만 존재)
TABLE_NAME	VARCHAR(30)	NOT NULL	테이블 이름
TABLESPACE_NAME	VARCHAR(30)		테이블 스페이스 이름
NUM_ROWS	NUMBER		테이블에 저장된 행 수

데이터 사전에 저장된 데이터베이스 관리와 관련된 정보 소개 또는 모든 열 정보가 필요하다면 오라클 공식 문서(docs.oracle.com/cd/B28359_01/server.111/b28320/statviews_2105.htm#REFRN20286)를 참고하세요.

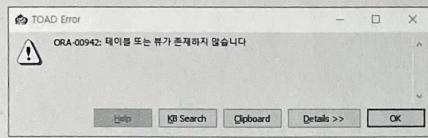
DBA_ 접두어를 가진 데이터 사전

DBA_ 접두어를 가진 데이터 사전은 데이터베이스 관리 권한을 가진 사용자만 조회할 수 있는 테이블로서 SCOTT 계정으로는 조회가 불가능합니다.

실습 13-5 SCOTT 계정으로 DBA_ 접두어 사용하기

01 SELECT * FROM DBA_TABLES;

:: 결과 화면



◀ 한 발 더 나가기!! 왜 데이터베이스에 존재하는데 '존재하지 않습니다' 라고 출력되나요?

실습 13-5의 결과 화면을 조금 더 자세히 살펴볼까요? DBA_TABLES는 분명 오라클 데이터베이스에 존재하지만 사용 권한이 없는 SCOTT 계정으로 조회하면 테이블이 존재하지 않는다는 오류 메시지가 나옵니다. 이는 사용 권한이 없는 사용자는 해당 개체의 존재 여부조차 확인할 수 없음을 의미합니다. '존재는 하지만 너에겐 권한이 없어'라는 식의 문구는 보안 문제를 일으킬 수 있습니다.

예를 들어 우리가 자주 사용하는 여러 웹 서비스의 로그인을 생각해 봅시다. 만약 어떤 사람이 악의적인 의도로 여러분이 사용하는 다른 웹 서비스의 아이디를 알아냈다고 가정해 보죠. 그리고 그 웹 서비스에서 해당 아이디로 로그인을 시도했을 때 '비밀번호가 틀렸다'라는 메시지가 나오면 이 웹 서비스에 '해당 아이디의 존재'를 인정하는 것이 되므로 추가 위험을 초래할 수 있습니다. 따라서 웹 서비스의 로그인 실패 메시지는 아이디, 비밀번호 둘 중 하나의 오류임을 알려 주지 않고 '아이디나 비밀번호가 틀렸다'라는 식으로 둘 중 하나가 틀린 것 같다고 제시하는 것이 일반적입니다. 즉 아이디 존재 여부를 알려 주지 않는 것이죠.

이와 마찬가지로 오라클 데이터베이스에서 어떤 사용자가 사용 권한이 없는 정보 열람을 시도할 경우에 오라클 데이터베이스는 해당 개체가 존재하지 않는다고 알려줍니다.



데이터베이스 권한이 있는 SYSTEM 사용자(비밀번호 oracle로 지정)로 접속하면 DBA_TABLES 조회가 가능합니다. 데이터베이스에 존재하는 모든 테이블이 출력되며 열 구성은 ALL_TABLES와 같습니다.

☞ 이 책에서 사용하는 토드 버전은 한 번에 접속할 수 있는 세션 수를 하나로 제한하고 있습니다. 따라서 SCOTT 계정을 사용 중이라면 종료 후 SYSTEM 계정으로 다시 접속하여 실습 13-6을 실행해 보세요.

실습 13-6 SYSTEM 계정으로 DBA_ 접두어 사용하기(SYSTEM 계정으로 접속했을 때)

```
01 SELECT * FROM DBA_TABLES;
```

:: 결과 화면

OWNER	TABLE_NAME	TABLESPACE_NAME	CLUSTER_NAME	IOT_NAME	STATUS
▶ SYS	ICOL\$	SYSTEM	C_OBJ#		VALID
SYS	CONS	SYSTEM			VALID
SYS	UNDO\$	SYSTEM			VALID
SYS	PROXY_ROLE_DATA\$	SYSTEM			VALID
SYS	FILE\$	SYSTEM			VALID
SYS	UET\$	SYSTEM	C_FILE#_BLOCK#		VALID
SYS	IND\$	SYSTEM	C_OBJ#		VALID
SYS	SEG\$	SYSTEM	C_FILE#_BLOCK#		VALID
SYS	COL\$	SYSTEM	C_OBJ#		VALID
SYS	CLU\$	SYSTEM	C_OBJ#		VALID
SYS	PROXY_DATA\$	SYSTEM			VALID
SYS	TS\$	SYSTEM	C_TS#		VALID

일부 데이터 및 열만 표시했습니다.

DBA_USERS로 사용자 정보 살펴보기

오라클 데이터베이스에 등록된 사용자 정보는 DBA_USERS에 있습니다. SCOTT 사용자 정보를 보려면 USERNAME 열을 WHERE 조건으로 지정하여 사용하면 됩니다.

실습 13-7 DBA_USERS를 사용하여 사용자 정보를 알아보기(SYSTEM 계정으로 접속했을 때)

```
01 SELECT *
02   FROM DBA_USERS
03  WHERE USERNAME = 'SCOTT';
```

:: 결과 화면

USERNAME	USER_ID	PASSWORD	ACCOUNT_STATUS	LOCK_DATE	EXPIRY_DATE	DEFAULT_TABLESPACE
▶ SCOTT	84		OPEN		2018-09-02 오후 7:41:14	USERS

일부 데이터 및 열만 표시했습니다.

이처럼 DBA_ 접두어가 붙은 데이터 사전은 오라클 데이터베이스 운영과 관련된 여러 정보를 보관합니다. 데이터베이스 자체를 관리하는 목적 외에 오라클 데이터베이스를 사용하여 데이터를 보관하고 관리하는 업무를 진행할 때는 그리 자주 사용하지 않습니다.

☞ 이 책에서는 이후 데이터 사전 뷰를 간단히 데이터 사전으로 명시합니다.

1분
복습

다음 빈칸을 채우며 복습해 보세요.

데이터 사전은 오라클 데이터베이스를 구성하고 운영하는 데이터를 저장하는 특수한 테이블로서 오라클 사용자가 직접 접근할 수 없습니다. 하지만 SELECT문으로 데이터를 열람할 수 있도록

¹ 데 _____ 를 제공합니다.

대표적인 ¹ 데 _____ 중 현재 접속한 사용자가 소유하는 테이블 목록을 보기 위해서

는 ² U _____ _TABLES를 사용합니다. 또한 사용자가 소유하는 테이블을 포함해 다른 사용자가 소유한 테이블 중 현재 사용자에게 사용 허가가 되어 있는 테이블을 보기 위해서는

³ A _____ _TABLES를 사용합니다.

답案 1. DBOBJECT 디렉터리 2. USER 3. ALL

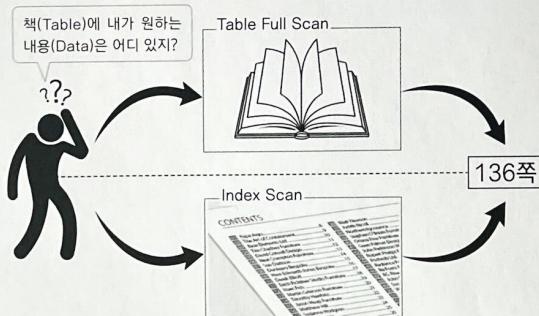
13-2 더 빠른 검색을 위한 인덱스

인덱스란?

책에서 특정 단어 또는 내용이 있는 페이지를 찾으려면 두 가지 방법을 생각할 수 있습니다. 책을 처음부터 끝까지 읽어 보며 찾을 수도 있고, 책의 목차나 색인을 통해 찾으려는 단어가 있는 페이지를 바로 찾아볼 수도 있습니다. 찾는 속도를 생각한다면 책의 모든 페이지를 읽어서 찾는 것보다 목록이나 색인을 활용하여 찾는 것이 빠릅니다. 그리고 책 내용이 많아질수록 속도 차이는 더욱 커집니다.

색인이라는 뜻의 인덱스(index)는 책 내용을 찾는 것과 마찬가지로 오라클 데이터베이스에서 데이터 검색 성능의 향상을 위해 테이블 옆에 사용하는 객체를 뜻합니다. 테이블에 보관된 특정 행 데이터의 주소, 즉 위치 정보를 책 페이지처럼 목록으로 만들어 놓은 것입니다. 인덱스는 테이블 옆을 여러 가지 분석을 통해 선정하여 설정할 수 있습니다.

인덱스 사용 여부에 따라 데이터 검색 방식을 Table Full Scan, Index Scan으로 구분합니다. 테이블 데이터를 처음부터 끝까지 검색하여 원하는 데이터를 찾는 방식은 Table Full Scan, 인덱스를 통해 데이터를 찾는 방식은 Index Scan이라고 합니다. 책에서 자신이 원하는 내용을 찾는 두 방법과 비슷합니다.



인덱스도 오라클 데이터베이스 객체이므로 소유 사용자와 사용 권한이 존재합니다. SCOTT 계정으로 접속하여 현재 SCOTT 소유의 인덱스 정보를 열람할 때 USER_INDEXES, USER_IND_COLUMNS와 같은 데이터 사전을 사용합니다. 실습 13-8과 실습 13-9의 결과를 살펴

보면 EMP 테이블의 EMPNO 열, DEPT 테이블의 DEPTNO 열에 인덱스가 이미 생성되어 있습니다. TABLE_NAME 열에서 인덱스가 속한 테이블을 확인할 수 있고 INDEX_NAME 열에서 인덱스를 지정한 열을 알 수 있습니다.

실습 13-8 SCOTT 계정이 소유한 인덱스 정보 알아보기(SCOTT 계정일 때)

```
01 SELECT *
02   FROM USER_INDEXES;
```

:: 결과 화면

INDEX_NAME	INDEX_TYPE	TABLE_OWNER	TABLE_NAME	TABLE_TYPE	UNIQUENESS	COMPRESSION
PK_DEPT	NORMAL	SCOTT	DEPT	TABLE	UNIQUE	DISABLED
PK_EMP	NORMAL	SCOTT	EMP	TABLE	UNIQUE	DISABLED

데이터의 일부 열만 표시했습니다.

실습 13-9 SCOTT 계정이 소유한 인덱스 컬럼 정보 알아보기(SCOTT 계정일 때)

```
01 SELECT *
02   FROM USER_IND_COLUMNS;
```

:: 결과 화면

INDEX_NAME	TABLE_NAME	COLUMN_NAME	COLUMN_POSITION	COLUMN_LENGTH	CHAR_LENGTH	DESCEND
PK_DEPT	DEPT	DEPTNO	1	22	0	ASC
PK_EMP	EMP	EMPNO	1	22	0	ASC

인덱스는 사용자가 직접 특정 테이블의 열에 지정할 수도 있지만 열이 기본키(primary key) 또는 고유키(unique key)일 경우에 자동으로 생성됩니다.

☞ 고유키는 열 데이터의 중복을 허용하지 않는 제약 조건(constraint)입니다. 이 내용은 14장에서 자세히 알아보겠습니다.

인덱스 생성

오라클 데이터베이스에서 자동으로 생성해 주는 인덱스 외에 사용자가 직접 인덱스를 만들 때는 CREATE문을 사용합니다. CREATE문에서는 인덱스를 생성할 테이블 및 열을 지정하며 열은 하나 또는 여러 개 지정할 수 있습니다. 지정한 각 열별로 인덱스 정렬 순서(오름차순 또는 내림차순)를 정할 수도 있습니다.

CREATE INDEX 인덱스 이름

기본 형식

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

EMP 테이블의 SAL 열에 인덱스를 생성하려면 다음과 같이 CREATE문을 실행합니다. 다음 CREATE문을 실행한 후 인덱스 생성을 확인하기 위해 USER_IND_COLUMNS 데이터 사전도 조회해 봅시다.

실습 13-10 EMP 테이블의 SAL 열에 인덱스를 생성하기

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

실습 13-11 생성된 인덱스 살펴보기(USER_IND_COLUMNS 사용)

```
01 SELECT * FROM USER_IND_COLUMNS;
```

:: 결과 화면(실습 13-11)

INDEX_NAME	TABLE_NAME	COLUMN_NAME	COLUMN_POSITION	COLUMN_LENGTH	CHAR_LENGTH	DESCEND
IDX_EMP_SAL	EMP	SAL	1	22	0	ASC
PK_DEPT	DEPT	DEPTNO	1	22	0	ASC
PK_EMP	EMP	EMPNO	1	22	0	ASC

실습 13-11의 결과로 IDX_EMP_SAL 인덱스가 생성되고, 인덱스의 정렬 옵션을 지정하지 않으면 기본값은 오름차순(ASC)으로 지정됩니다. 인덱스가 걸린 SAL열을 WHERE의 검색 조건으로 하여 EMP 테이블을 조회하면 출력 속도가 빨라질 것이라 예상할 수 있습니다. 하지만 인덱스를 지정할 열의 선정은 데이터의 구조 및 데이터의 분포도 등 여러 조건을 고려해서 이루어져야 합니다. 인덱스를 지정하면 데이터 조회를 반드시 빠르게 한다고 보장하기는 어렵습니다. 다음은 이 책에서 다루지 않지만 목적에 따라 여러 방식으로 생성할 수 있는 인덱스의 종류입니다.

방식	사용
단일 인덱스(single index)	CREATE INDEX IDX_NAME ON EMP(SAL);
복합 인덱스(concatenated index) 결합 인덱스(composite index) - 두 개 이상 열로 만들어지는 인덱스 - WHERE절의 두 열이 AND 연산으로 묶이는 경우	CREATE INDEX IDX_NAME ON EMP(SAL, ENAME, …);
고유 인덱스(unique index) - 열에 중복 데이터가 없을 때 사용 - UNIQUE 키워드를 지정하지 않으면 비고유 인덱스(non unique index)가 기본값	CREATE UNIQUE INDEX IDX_NAME ON EMP(EMPNO);
함수 기반 인덱스(function based index) - 열에 산술식 같은 데이터 가공이 진행된 결과로 인덱스 생성	CREATE INDEX IDX_NAME ON EMP(SAL*12 + COMM);
비트맵 인덱스(bitmap index) - 데이터 종류가 적고 같은 데이터가 많이 존재할 때 주로 사용	CREATE BITMAP INDEX IDX_NAME ON EMP(JOB);

인덱스 삭제

인덱스 삭제는 DROP 명령어를 사용합니다.

```
DROP INDEX 인덱스 이름;
```

기본 형식

다음 실습에서 EMP 테이블의 SAL 열에 생성한 IDX_EMP_SAL 인덱스를 삭제합니다.

실습 13-12 인덱스 삭제하기

```
01 DROP INDEX IDX_EMP_SAL;
```

실습 13-13 생성된 인덱스 살펴보기(USER_IND_COLUMNS 사용)

```
01 SELECT * FROM USER_IND_COLUMNS;
```

:: 결과 화면

INDEX_NAME	TABLE_NAME	COLUMN_NAME	COLUMN_POSITION	COLUMN_LENGTH	CHAR_LENGTH	DESCEND
PK_DEPT	DEPT	DEPTNO	1	22	0	ASC
PK_EMP	EMP	EMPNO	1	22	0	ASC

지금까지 인덱스에 대해 알아보았습니다. 인덱스는 데이터 접근 및 검색 속도 향상을 위해 사용하는 객체이지만 인덱스 생성이 항상 좋은 결과로 이어지지는 않습니다. 정확한 데이터 분석에 기반을 두지 않은 인덱스의 무분별한 생성은 오히려 성능을 떨어뜨리는 원인이 되기도 합니다. 인덱스는 데이터 종류, 분포도, 조회하는 SQL의 구성, 데이터 조작 관련 SQL문의 작업 빈도, 검색 결과가 전체 데이터에서 차지하는 비중 등 많은 요소를 고려하여 생성합니다. 인덱스 사용 및 생성과 관련하여 더 자세한 내용은 SQL 튜닝(tunning) 관련 서적 또는 인터넷 문서를 참고하세요.

1분
복습

다음은 인덱스를 생성하는 기본 형식입니다. 빈칸에 들어갈 키워드를 채워 보세요.

```
1 인덱스 이름
2 테이블 이름(열 이름1 ASC or DESC,
    열 이름2 ASC or DESC,
    ...
    );
```

문장 1. CREATE INDEX 2. ON

13-3 테이블처럼 사용하는 뷰

뷰란?

흔히 가상 테이블(virtual table)로 부르는 뷰(view)는 하나 이상의 테이블을 조회하는 SELECT 문을 저장한 객체를 뜻합니다. SELECT문을 저장하기 때문에 물리적 데이터를 따로 저장하지는 않습니다. 따라서 뷰를 SELECT문의 FROM절에 사용하면 특정 테이블을 조회하는 것과 같은 효과를 얻을 수 있습니다. 다음 SELECT문을 VW_EMP20이란 이름의 뷰로 생성한 경우를 예로 살펴보겠습니다.

```
SELECT EMPNO, ENAME, JOB, DEPTNO  
  FROM EMP  
 WHERE DEPTNO = 20;
```

위 SELECT문은 부서 번호가 20인 사원의 사원 번호, 사원 이름, 직책, 부서 번호를 출력합니다. 이 SELECT문을 저장한 VW_EMP20을 생성하면 다음과 같이 SELECT문으로 VW_EMP20 뷰를 테이블처럼 조회할 수 있습니다. 마치 서브쿼리를 사용한 것 같습니다.

뷰	서브쿼리
<pre>SELECT * FROM VW_EMP20;</pre>	<pre>SELECT * FROM(SELECT EMPNO, ENAME, JOB, DEPTNO FROM EMP WHERE DEPTNO = 20);</pre>

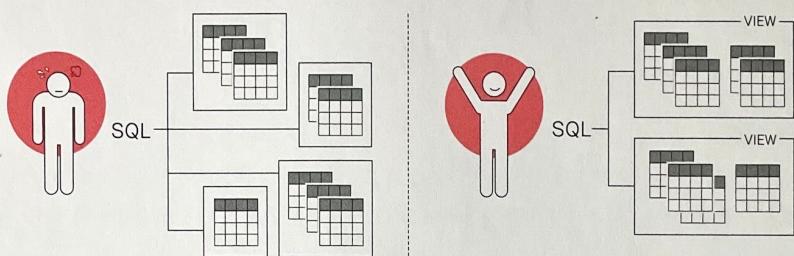
뷰의 사용 목적(편리성)

뷰와 서브쿼리를 비교한 후 ‘그러면 뷰를 굳이 쓸 필요가 없지 않나?’라는 생각이 들 수도 있습니다. 하지만 뷰는 크게 다음 두 가지 목적을 위해 주로 사용합니다.

1. 편리성 : SELECT문의 복잡도를 완화하기 위해
2. 보안성 : 테이블의 특정 열을 노출하고 싶지 않을 경우

실무에서 사용하는 SELECT문은 이 책의 예제와 같이 짧게 몇 줄로 이루어진 것도 있지만 길게는 A4 용지 몇장을 꽉 채울 정도의 분량으로 이루어진 경우도 어렵지 않게 찾아볼 수 있습니다. 이렇듯 많은 분량의 SELECT문 여러 개의 결과 값을 다시 조인하고 서브쿼리로 WHERE 조건식에도 사용한다면 전체 SELECT문은 훨씬 더 커질 것입니다.

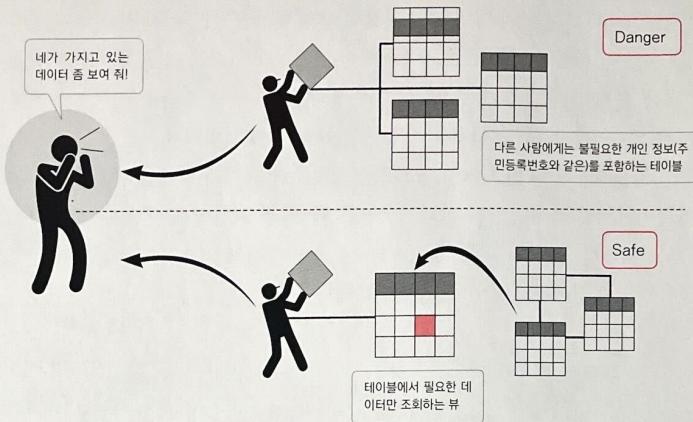
또한 이후 수정이 필요하거나 다른 개발자가 코드를 처음부터 파악해야 하는 경우에 적잖은 시간과 노력이 든다는 것도 생각해야 합니다. 이럴 때 여러 SQL문에서 자주 활용하는 SELECT문을 뷰로 저장해 놓은 후 다른 SQL문에서 활용하면 전체 SQL문의 복잡도를 완화하고 본래 목적의 메인 쿼리에 집중할 수 있어 편리합니다.



뷰의 사용 목적(보안성)

뷰를 사용하는 또 다른 중요한 이유는 데이터 보안과 관련이 있습니다. VW_EMP20 뷰가 저장한 SELECT문은 20번 부서 사원의 사원 번호, 이름, 직책, 부서 번호를 출력합니다. SCOTT 계정이 소유한 EMP 테이블의 데이터를 다른 사용자가 조회하는 일이 생겼을 때를 생각해 보죠.

이 다른 사용자는 EMP 테이블의 사원 번호, 이름, 직책, 부서 번호 데이터만 열람할 수 있으면 원하는 작업을 할 수 있다고 가정해 보겠습니다. EMP 테이블에는 급여(SAL)나 추가 수당(COMM)과 같이 아무에게나 노출하기에는 예민한 데이터가 존재합니다. 때로는 주민등록번호처럼 담당자 이외에 노출이 허용되지 않는 중요한 개인 정보 데이터가 존재할 수도 있습니다. 이럴 때 해당 사용자에게 특정 테이블의 전체 조회 권한을 부여하는 것은 데이터 보안에 위협이 될 수 있으므로 주의해야 합니다. 테이블의 일부 데이터 또는 조인이나 여러 함수 등으로 가공을 거친 데이터만 SELECT하는 뷰 열람 권한을 제공하는 것이 불필요한 데이터 노출을 막을 수 있기 때문에 더 안전한 방법입니다.



뷰 생성

뷰는 CREATE문으로 생성할 수 있습니다. SCOTT 계정은 뷰 생성 권한이 없으므로 SYSTEM 계정으로 접속한 후 다음 명령어를 사용하여 SCOTT 계정에 권한을 부여해 주어야 합니다.

실습 13-14 뷰를 생성하기 위해 계정 변경 접속하기(SQL*PLUS)

01 SQLPLUS SYSTEM/oracle

02 GRANT CREATE VIEW TO SCOTT;

:: 결과 화면

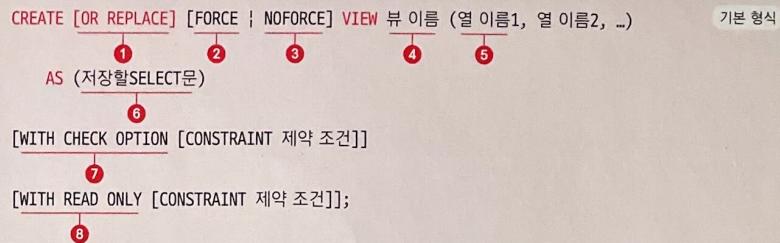
```
명령 프롬프트 - SQLPLUS SYSTEM/oracle
C:\Users\easy\publishing>SQLPLUS SYSTEM/oracle
SQL*Plus: Release 11.2.0.1.0 Production on 토 7월 14 02:06:48 2018
Copyright <c> 1982, 2010, Oracle. All rights reserved.

다음에 접속됨:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> GRANT CREATE VIEW TO SCOTT;
권한이 부여되었습니다.

SQL>
```

권한 부여가 완료되었다면 토드에서 SCOTT 계정으로 접속하여 뷰를 생성해 보겠습니다. 다음은 CREATE문의 기본 형식입니다.



요소	설명
① OR REPLACE	같은 이름의 뷰가 이미 존재할 경우에 현재 생성할 뷰로 대체하여 생성(선택)
② FORCE	뷰가 저장할 SELECT문의 기반 테이블이 존재하지 않아도 강제로 생성(선택)
③ NOFORCE	뷰가 저장할 SELECT문의 기반 테이블이 존재할 경우에만 생성(기본값)(선택)
④ 뷰 이름	생성할 뷰 이름을 지정(필수)
⑤ 열 이름	SELECT문에 명시된 이름 대신 사용할 열 이름 지정(생략 가능)(선택)
⑥ 저장할 SELECT문	생성할 뷰에 저장할 SELECT문 지정(필수)
⑦ WITH CHECK OPTION	지정한 제약 조건을 만족하는 데이터에 한해 DML 작업이 가능하도록 뷰 생성(선택)
⑧ WITH READ ONLY	뷰의 열람, 즉 SELECT만 가능하도록 뷰 생성(선택)

앞에서 뷰를 설명하면서 예로 든 VW_EMP20 뷰를 다음과 같이 만들어 보죠.

실습 13-15 뷰 생성하기(토드)

```

01  CREATE VIEW VW_EMP20
02    AS (SELECT EMPNO, ENAME, JOB, DEPTNO
03      FROM EMP
04     WHERE DEPTNO = 20);

```

VW_EMP20 뷰가 잘 생성되었는지 알아보려면 USER_VIEWS 데이터 사전을 조회해 보면 됩니다.

실습 13-16 생성한 뷰 확인하기(토드)

```

01  SELECT *
02    FROM USER_VIEWS;

```

실습 13-16의 SELECT문을 토드에서 실행한 결과를 보면 TEXT 열의 데이터가 제대로 나오지 않습니다.

:: 결과 화면

VIEW_NAME	TEXT_LENGTH	TEXT	TYPE_TEXT_LENGTH	TYPE_TEXT
VW_EMP20	80	(WIDEMEMO)		

데이터의 일부 열만 표시했습니다.

☞ TEXT 열의 자료형이 LONG으로 지정되어 나오지 않으므로 토드 위쪽 메뉴의 [View → Toad Options → Data Grids → Data]의 Preview CLOB and LONG data 옵션을 체크해 주면 됩니다.

하지만 SQL*PLUS에서 USER_VIEW로 조회하면 VW_EMP20 뷰에 저장된 SELECT문을 다음과 같이 확인할 수 있습니다.

실습 13-17 생성한 뷰 내용 확인하기(SCOTT 계정으로 접속했을 때)

```
01  SELECT VIEW_NAME, TEXT_LENGTH, TEXT
02    FROM USER_VIEWS;
```

:: 결과 화면

```
C:\W>SQLPLUS SCOTT/tiger
SQL*Plus: Release 11.2.0.1.0 Production on 수 4월 5 00:44:31 2017
Copyright (c) 1982, 2010, Oracle. All rights reserved.

다음에 접속됨:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> SELECT VIEW_NAME, TEXT_LENGTH, TEXT
  2    FROM USER_VIEWS;
  VIEW_NAME          TEXT_LENGTH
  -----
  TEXT
  -----
VW_EMP20                      80
<SELECT EMPNO, ENAME, JOB, DEPTNO
   FROM EMP
 WHERE DEPTNO = 20>

SQL>
```

이렇게 생성한 뷰는 앞에서 언급한 대로 SELECT문의 FROM절에서 테이블처럼 지정하여 사용할 수 있습니다. VW_EMP20 뷰는 EMP 테이블 하나만 조회하는 SELECT문을 저장하지만 여러 테이블을 조인하거나 서브쿼리를 사용한 좀 더 복합적인 SELECT문도 뷰에 저장할 수 있다는 것을 기억하세요.

실습 13-18 생성한 뷰 조회하기

```
01  SELECT *
02    FROM VW_EMP20;
```

:: 결과 화면

	EMPNO	ENAME	JOB	DEPTNO
▶	7369	SMITH	CLERK	20
	7566	JONES	MANAGER	20
	7788	SCOTT	ANALYST	20
	7876	ADAMS	CLERK	20
	7902	FORD	ANALYST	20

1분 복습

부서 번호가 30인 사원 정보의 모든 열을 출력하는 VM_EMP30ALL 뷰를 작성하는 다음 SQL문의 빈칸을 채워 보세요.

```
1          VW_EMP30ALL
AS (SELECT *
     FROM EMP
     WHERE 2
   );
```

답지 1. CREATE VIEW 2. DEPTNO = 30

뷰 삭제

뷰를 삭제할 때 DROP문을 사용합니다. 생성한 VW_EMP20 뷰를 삭제해 볼까요?

실습 13-19 뷰 삭제하기

```
01  DROP VIEW VW_EMP20;
```

VW_EMP20 뷰를 삭제한 후 USER_VIEWS 데이터 사전을 조회해 보면 뷰가 삭제되었음을 알 수 있습니다. 뷰는 실제 데이터가 아닌 SELECT문만 저장하므로 뷰를 삭제해도 테이블이나 데이터가 삭제되는 것은 아닙니다.

:: 결과 화면

VIEW_NAME	TEXT_LENGTH	TEXT	TYPE_TEXT_LENGTH	TYPE_TEXT	OID_TEXT_LENGTH

● 한 발 더 나가기!! 뷰와 데이터 조작어

뷰는 SELECT문만 저장하는 객체이기 때문에 데이터 삽입·수정·삭제 같은 데이터 조작어 사용이 불가능할 것이라 생각할 수 있지만, 의외로 뷰에도 데이터 조작어를 직접 사용할 수 있는 경우가 있습니다. 하지만 뷰를 통한 테이블 데이터 조작이 가능하려면 여러 가지 조건을 만족해야 하고 테이블을 설계할 때 누락된 내용이 있으면 뷰를 통한 데이터 조작으로 인해 적합하지 않은 데이터가 생길 수도 있으므로 자주 사용하는 편은 아닙니다.

이는 뷰의 주 목적이 물리적 데이터를 저장하지 않고 SELECT문만 저장함으로써 테이블의 데이터를 열람하는 것이기 때문입니다. 이 책에서 다루지는 않지만 데이터를 따로 저장하는 것이 협용되는 구체화 뷰(materialized view)도 존재합니다.

인라인 뷰를 사용한 TOP-N SQL문

CREATE문을 통해 객체로 만들어지는 뷰 외에 SQL문에서 일회성으로 만들어서 사용하는 뷰를 인라인 뷰(inline view)라고 합니다. SELECT문에서 사용되는 서브쿼리, WITH절에서 미리 이름을 정의해 두고 사용하는 SELECT문 등이 이에 해당합니다.

☞ 인라인 뷰를 사용한 서브쿼리와 WITH절은 09장에서 다루었습니다.

이 인라인 뷰와 ROWNUM을 사용하면 ORDER BY절을 통해 정렬된 결과 중 최상위 몇 개 데이터만을 출력하는 것이 가능합니다. ROWNUM을 알아보기 위해 다음 SELECT문을 실행하여 결과를 확인해 보세요.

실습 13-20 ROWNUM을 추가로 조회하기

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

다음 결과 화면을 살펴보면 ROWNUM 열은 EMP 테이블에 존재하지는 않지만 ROWNUM 열의 데이터가 숫자로 출력되고 있음을 확인할 수 있습니다.

:: 결과 화면

ROWNUM	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7369	SMITH	CLERK	7902	1980/12/17	800		20
2	7499	ALLEN	SALESMAN	7698	1981/02/20	1600	300	30
3	7521	WARD	SALESMAN	7698	1981/02/22	1250	500	30
4	7566	JONES	MANAGER	7839	1981/04/02	2975		20
5	7654	MARTIN	SALESMAN	7698	1981/09/28	1250	1400	30
6	7698	BLAKE	MANAGER	7839	1981/05/01	2850		30
7	7782	CLARK	MANAGER	7839	1981/06/09	2450		10
8	7788	SCOTT	ANALYST	7566	1987/04/19	3000		20
9	7839	KING	PRESIDENT		1981/11/17	5000		10
10	7844	TURNER	SALESMAN	7698	1981/09/08	1500	0	30
11	7876	ADAMS	CLERK	7788	1987/05/23	1100		20
12	7900	JAMES	CLERK	7698	1981/12/03	950		30
13	7902	FORD	ANALYST	7566	1981/12/03	3000		20
14	7934	MILLER	CLERK	7782	1982/01/23	1300		10

ROWNUM은 의사 열(pseudo column)이라고 하는 특수 열입니다. 의사 열은 데이터가 저장되는 실제 테이블에 존재하지는 않지만 특정 목적을 위해 테이블에 저장되어 있는 열처럼 사용 가능한 열을 뜻합니다.

② ROWNUM 외에 인덱스와 밀접하게 연관된 ROWID도 대표적인 의사 열입니다. 자세한 내용은 오라클 공식 문서(docs.oracle.com/cd/B28359_01/server.111/b28286/pseudocolumns.htm#SQLRF0025)를 참고하세요.

ROWNUM 열 데이터 번호는 테이블에 저장된 행이 조회된 순서대로 매겨진 일련번호입니다. 여기에서 ORDER BY 절을 사용하여 내림차순 급여로 EMP 테이블을 다시 조회해 보겠습니다.

실습 13-21 EMP 테이블을 SAL 열 기준으로 정렬하기

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

실습 13-21의 결과 화면에서 데이터를 급여 기준으로 정렬(내림차순)했지만 ROWNUM은 앞에서 사용한 SELECT문의 행 번호와 같은 번호로 매겨져 있다는 것을 눈여겨봅시다. ROWNUM은 데이터를 하나씩 추가할 때 매겨지는 번호이므로 ORDER BY 절을 통해 정렬해도 유지되는 특성이 있습니다.

:: 결과 화면

ROWNUM	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
▶ 9	7839	KING	PRESIDENT		1981/11/17	5000		10
13	7902	FORD	ANALYST	7566	1981/12/03	3000		20
8	7788	SCOTT	ANALYST	7566	1987/04/19	3000		20
4	7566	JONES	MANAGER	7839	1981/04/02	2975		20
6	7698	BLAKE	MANAGER	7839	1981/05/01	2850		30
7	7782	CLARK	MANAGER	7839	1981/06/09	2450		10
2	7499	ALLEN	SALESMAN	7698	1981/02/20	1600	300	30
10	7844	TURNER	SALESMAN	7698	1981/09/08	1500	0	30
14	7934	MILLER	CLERK	7782	1982/01/23	1300		10
3	7521	WARD	SALESMAN	7698	1981/02/22	1250	500	30
5	7654	MARTIN	SALESMAN	7698	1981/09/28	1250	1400	30
11	7876	ADAMS	CLERK	7788	1987/05/23	1100		20
12	7900	JAMES	CLERK	7698	1981/12/03	950		30
1	7369	SMITH	CLERK	7902	1980/12/17	800		20

이 특성을 인라인 뷰에서 적용하면 정렬된 SELECT문의 결과 순번을 매겨서 출력할 수 있습니다. 서브쿼리를 인라인 뷰로 사용한 SELECT문과 WITH절의 인라인 뷰를 사용한 SELECT문의 결과는 같습니다.

실습 13-22 인라인 뷰(서브쿼리 사용)

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

실습 13-23 인라인 뷰(WITH절 사용)

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

다음 결과에서 ORDER BY절로 정렬(SAL 열의 내림차순)된 SELECT문의 데이터가 메인쿼리의 SELECT문에서 한 행씩 순서대로 ROWNUM이 매겨져 정렬된 순서 그대로 번호가 매겨진 것을 확인할 수 있습니다.

:: 결과 화면(실습 13-22, 13-23의 실행 결과가 동일)

■	ROWNUM	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
▶	1	7839	KING	PRESIDENT		1981/11/17	5000		10
	2	7902	FORD	ANALYST	7566	1981/12/03	3000		20
	3	7788	SCOTT	ANALYST	7566	1987/04/19	3000		20
	4	7566	JONES	MANAGER	7839	1981/04/02	2975		20
	5	7698	BLAKE	MANAGER	7839	1981/05/01	2850		30
	6	7782	CLARK	MANAGER	7839	1981/06/09	2450		10
	7	7499	ALLEN	SALESMAN	7698	1981/02/20	1600	300	30
	8	7844	TURNER	SALESMAN	7698	1981/09/08	1500	0	30
	9	7934	MILLER	CLERK	7782	1982/01/23	1300		10
	10	7521	WARD	SALESMAN	7698	1981/02/22	1250	500	30
	11	7654	MARTIN	SALESMAN	7698	1981/09/28	1250	1400	30
	12	7876	ADAMS	CLERK	7788	1987/05/23	1100		20
	13	7900	JAMES	CLERK	7698	1981/12/03	950		30
	14	7369	SMITH	CLERK	7902	1980/12/17	800		20

마지막으로 굽여가 높은 상위 세 명의 데이터만 출력하려면 ROWNUM을 WHERE절 조건으로 지정하면 됩니다. 인라인 뷰를 사용한 TOP-N 추출은 활용 빈도가 높으니 꼭 기억해 두세요.

실습 13-24 인라인 뷰로 TOP-N 추출하기(서브쿼리 사용)

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

실습 13-25 인라인 뷰로 TOP-N 추출하기(WITH절 사용)

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

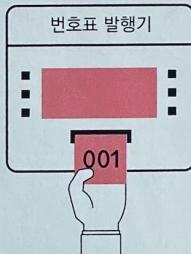
:: 결과 화면(실습 13-24, 13-25의 실행 결과가 동일)

■	ROWNUM	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
▶	1	7839	KING	PRESIDENT		1981/11/17	5000		10
	2	7788	SCOTT	ANALYST	7566	1987/04/19	3000		20
	3	7902	FORD	ANALYST	7566	1981/12/03	3000		20

13-4 규칙에 따라 순번을 생성하는 시퀀스

시퀀스란?

시퀀스(sequence)는 오라클 데이터베이스에서 특정 규칙에 맞는 연속 숫자를 생성하는 객체입니다. 은행이나 병원의 대기 순번표와 마찬가지로 번호를 사용해야 하는 사용자에게 계속 다음 번호를 만들어 주는 역할을 합니다.



단지 연속하는 새로운 번호를 만드는 일이라면 다음과 같이 MAX 함수에 1을 더한 값을 사용해도 상관없을 것입니다. 이 방식은 실제로 연속하는 숫자로 이루어진 웹 서비스의 새로운 게시판 번호나 상품 주문 번호 등을 생성할 때 종종 생성하는 방식입니다.

```
SELECT MAX(글 번호) + 1  
FROM 게시판 테이블
```

하지만 이 방식은 테이블 데이터가 많아질수록 가장 큰 데이터를 찾고 새로운 번호를 계산하는 시간이 함께 늘어나므로 아쉬운 부분이 있습니다. 또한 동시에 여러 곳에서 새로운 번호를 요구했을 경우에 SELECT문의 결과 값이 같이 나와 번호가 중복될 수도 있습니다. 이와 비교하여 시퀀스는 단순히 번호 생성을 위한 객체이지만 지속적이고 효율적인 번호 생성이 가능하므로 여러모로 자주 사용하는 객체입니다.

시퀀스 생성

시퀀스 역시 CREATE문으로 생성하며 다음과 같이 다양한 옵션을 지정할 수 있습니다.

CREATE SEQUENCE 시퀀스 이름 -①

기본 형식

- [INCREMENT BY n] -②
- [START WITH n] -③
- [MAXVALUE n | NOMAXVALUE] -④
- [MINVALUE n | NOMINVALUE] -⑤
- [CYCLE | NOCYCLE] -⑥
- [CACHE n | NOCACHE] -⑦

번호	설명
①	생성할 시퀀스 이름 지정. 아래 절(② ~ ⑦)들을 지정하지 않았을 경우 1부터 시작하여 1만큼 계속 증가하는 시퀀스가 생성(필수)
②	시퀀스에서 생성할 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, 최솟값(MINVALUE)을 초과값으로 지정. NOMAXVALUE로 지정하였을 경우 오름차순이면 10^{37} , 내림차순일 경우 -1로 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^{36} 으로 설정(선택)
⑥	시퀀스에서 생성한 번호가 최댓값(MAXVALUE)에 도달했을 경우 CYCLE이면 시작 값(START WITH)에서 다시 시작, NOCYCLE이면 번호 생성이 중단되고, 추가 번호 생성을 요청하면 오류 발생(선택)
⑦	시퀀스가 생성할 번호를 메모리에 미리 할당해 놓은 수를 지정, NOCACHE는 미리 생성하지 않도록 설정. 옵션을 모두 생략하면 기본값은 20(선택)

시퀀스를 사용하기 위해 DEPT 테이블과 열 구성은 같고 데이터는 없는 DEPT_SEQUENCE 테이블을 생성해 보죠.

실습 13-26 DEPT 테이블을 사용하여 DEPT_SEQUENCE 테이블 생성하기

```

01 CREATE TABLE DEPT_SEQUENCE
02     AS SELECT *
03         FROM DEPT
04         WHERE 1 <> 1;

05 SELECT * FROM DEPT_SEQUENCE;

```

:: 결과 화면

	DEPTNO	DNAME	LOC

기존의 DEPT 테이블에서 부서 번호(DEPTNO)는 10으로 시작해서 10씩 증가했습니다. 이와 같이 번호가 매겨질 수 있도록 오른쪽과 같이 시퀀스를 생성해 봅시다. 시퀀스 생성을 확인하기 위해 다음과 같이 USER_SEQUENCES 데이터 사전을 조회하여 확인해 보세요.

실습 13-27 시퀀스 생성하기

```
01 CREATE SEQUENCE SEQ_DEPT_SEQUENCE  
02 INCREMENT BY 10  
03 START WITH 10  
04 MAXVALUE 90  
05 MINVALUE 0  
06 NOCYCLE  
07 CACHE 2;
```

실습 13-28 생성한 시퀀스 확인하기

```
01 SELECT *  
02 FROM USER_SEQUENCES;
```

:: 결과 화면(실습 13-28 결과)

SEQUENCE_NAME	MIN_VALUE	MAX_VALUE	INCREMENT_BY	CYCLE_FLAG	ORDER_FLAG	CACHE_SIZE	LAST_NUMBER
SEQ_DEPT_SEQUENCE	10	90	10	N	N	2	10

시퀀스 사용

생성된 시퀀스를 사용할 때는 [시퀀스 이름.CURRVAL]과 [시퀀스 이름.NEXTVAL]을 사용할 수 있습니다. CURRVAL은 시퀀스에서 마지막으로 생성한 번호를 반환하며 NEXTVAL은 다음 번호를 생성합니다. 그리고 CURRVAL은 시퀀스를 생성하고 바로 사용하면 번호가 만들어진 적이 없으므로 오류가 납니다.

먼저 SEQ_DEPT_SEQUENCE 시퀀스를 사용하여 DEPT_SEQUENCE 테이블에 새로운 부서를 추가하려면 다음과 같이 INSERT문에 NEXTVAL을 사용합니다. 시작 값(STRAT WITH)이 10이므로 부서 번호가 10으로 삽입되었음을 알 수 있습니다.

실습 13-29 시퀀스에서 생성한 순번을 사용한 INSERT문 실행하기

```
01 INSERT INTO DEPT_SEQUENCE (DEPTNO, DNAME, LOC)  
02 VALUES (SEQ_DEPT_SEQUENCE.NEXTVAL, 'DATABASE', 'SEOUL');  
  
03 SELECT * FROM DEPT_SEQUENCE ORDER BY DEPTNO;
```

:: 결과 화면

DEPTNO	DNAME	LOC
10	DATABASE	SEOUL

CURRVAL을 사용하면 가장 마지막으로 생성된 순번, 즉 10이 반환되는 것도 확인해 보세요.

실습 13-30 가장 마지막으로 생성된 시퀀스 확인하기

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

:: 결과 화면

☰	CURRVAL
▶	10

실습 13-29의 INSERT문을 시퀀스의 NEXTVAL을 사용하여 부서 번호가 90번에 이를 때 까지 실행해 보죠(실습 13-29의 INSERT문을 9번 실행하기). 그 후 다시 실행하면 최댓값(MAXVALUE)이 이미 생성되었고 NOCYCLE 옵션으로 순환되지 않도록 설정하였으므로 오류가 납니다.

실습 13-31 시퀀스에서 생성한 순번을 반복 사용하여 INSERT문 실행하기

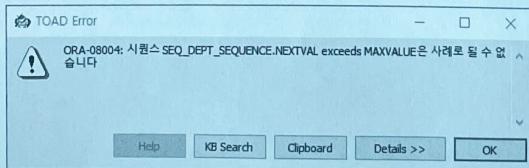
```
01  INSERT INTO DEPT_SEQUENCE (DEPTNO, DNAME, LOC)  
02    VALUES (SEQ_DEPT_SEQUENCE.NEXTVAL, 'DATABASE', 'SEOUL');  
  
03  SELECT * FROM DEPT_SEQUENCE ORDER BY DEPTNO;
```

:: 결과 화면

☰	DEPTNO	DNAME	LOC
▶	10	DATABASE	SEOUL
	20	DATABASE	SEOUL
	30	DATABASE	SEOUL
	40	DATABASE	SEOUL
	50	DATABASE	SEOUL
	60	DATABASE	SEOUL
	70	DATABASE	SEOUL
	80	DATABASE	SEOUL
	90	DATABASE	SEOUL



INSERT문을 9번 반복하여 실행하면
DEPT_SEQUENCE 테이블에 9개의
열이 생성됩니다.



시퀀스 최댓값 90에 도달 후 다시 INSERT문을 실행하면 시퀀스는 번호를 더 생성하지 못합니다.

시퀀스 수정

ALTER 명령어로 시퀀스를 수정하고 DROP 명령어로 시퀀스를 삭제합니다. 시퀀스 수정은 오른쪽과 같이 옵션을 재설정하는 데 사용합니다. 그리고 START WITH 값은 변경할 수 없습니다.

ALTER SEQUENCE 시퀀스 이름

기본 형식

[INCREMENT BY n]
[MAXVALUE n | NOMAXVALUE]
[MINVALUE n | NOMINVALUE]
[CYCLE | NOCYCLE]
[CACHE n | NOCACHE]

앞에서 생성한 시퀀스인 SEQ_DEPT_SEQUENCE의 최댓값(MAXVALUE)을 99, 증가 값(INCREMENT BY)을 3, 그리고 NOCYCLE 대신 CYCLE 옵션을 주어 오른쪽과 같이 수정해 볼까요?

실습 13-32 시퀀스 옵션 수정하기

```
01 ALTER SEQUENCE SEQ_DEPT_SEQUENCE  
02 INCREMENT BY 3  
03 MAXVALUE 99  
04 CYCLE;
```

실습 13-33 옵션을 수정한 시퀀스 조회하기

```
01 SELECT *  
02 FROM USER_SEQUENCES;
```

:: 결과 화면(실습 13-33 결과)

SEQUENCE_NAME	MIN_VALUE	MAX_VALUE	INCREMENT_BY	CYCLE_FLAG	ORDER_FLAG	CACHE_SIZE	LAST_NUMBER
SEQ_DEPT_SEQUENCE	0	99	3	Y	N	2	93

시퀀스를 수정한 후 실습 13-34와 같이 다시 INSERT문을 실행해 결과를 확인해 보세요.

실습 13-34 수정한 시퀀스를 사용하여 INSERT문 실행하기

```
01 INSERT INTO DEPT_SEQUENCE (DEPTNO, DNAME, LOC)  
02 VALUES (SEQ_DEPT_SEQUENCE.NEXTVAL, 'DATABASE', 'SEOUL');  
  
03 SELECT * FROM DEPT_SEQUENCE ORDER BY DEPTNO;
```

:: 결과 화면

DEPTNO	DNAME	LOC
10	DATABASE	SEOUL
20	DATABASE	SEOUL
30	DATABASE	SEOUL
40	DATABASE	SEOUL
50	DATABASE	SEOUL
60	DATABASE	SEOUL
70	DATABASE	SEOUL
80	DATABASE	SEOUL
90	DATABASE	SEOUL
93	DATABASE	SEOUL



수정된 시퀀스의 증가 값 때문에 새로 추가된 마지막 열의 DEPTNO 값은 100이 아닌 93이 되었습니다.

실습 13-35의 INSERT문을 몇 번 더 반복 실행하면 96, 99(MAXVALUE) 번호가 생성된 후 CYCLE 옵션으로 인해 최솟값(MINVALUE)이 0에서 다시 3씩 증가되는 것을 확인할 수 있습니다.

실습 13-35 CYCLE 옵션을 사용한 시퀀스의 최댓값 도달 후 수행 결과 확인하기

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

03  SELECT * FROM DEPT_SEQUENCE ORDER BY DEPTNO;
```

:: 결과 화면

DEPTNO	DNAME	LOC
0	DATABASE	SEOUL
3	DATABASE	SEOUL
10	DATABASE	SEOUL
20	DATABASE	SEOUL
30	DATABASE	SEOUL
40	DATABASE	SEOUL
50	DATABASE	SEOUL
60	DATABASE	SEOUL
70	DATABASE	SEOUL
80	DATABASE	SEOUL
90	DATABASE	SEOUL
93	DATABASE	SEOUL
96	DATABASE	SEOUL
99	DATABASE	SEOUL

새로 추가된 열의 DEPTNO 값이 99(시퀀스의 최댓값)가 되면 다시 0(시퀀스의 최솟값)부터 3씩 증가되어 열이 추가됩니다.

시퀀스 삭제

DROP SEQUENCE를 사용하면 시퀀스를 삭제할 수 있습니다. 생성한 SEQ_DEPT_SEQUENCE 시퀀스를 삭제해 보죠. 시퀀스를 삭제해도 시퀀스를 사용(SEQ_DEPT_SEQUENCE.NEXTVAL)하여 추가된 데이터는 삭제되지 않는다는 것도 눈여겨보세요.

실습 13-36 시퀀스 삭제 후 확인하기

```
01  DROP SEQUENCE SEQ_DEPT_SEQUENCE;

02  SELECT * FROM USER_SEQUENCES;
```

:: 결과 화면

SEQUENCE_NAME	MIN_VALUE	MAX_VALUE	INCREMENT_BY	CYCLE_FLAG	ORDER_FLAG	CACHE_SIZE	LAST_NUMBER

13-5 공식 별칭을 지정하는 동의어

동의어란?

동의어(synonym)는 테이블·뷰·시퀀스 등 객체 이름 대신 사용할 수 있는 다른 이름을 부여하는 객체입니다. 주로 테이블 이름이 너무 길어 사용이 불편할 때 좀 더 간단하고 짧은 이름을 하나 더 만들어 주기 위해 사용합니다. 동의어를 만들기 위해서는 CREATE문을 사용하며 다음과 같이 작성합니다.



요소	설명
① PUBLIC	동의어를 데이터베이스 내 모든 사용자가 사용할 수 있도록 설정. 생략할 경우 동의어를 생성한 사용자만 사용 가능(PUBLIC으로 생성되어도 본래 객체의 사용 권한이 있어야 사용 가능)(선택)
② 동의어 이름	생성할 동의어 이름(필수)
③ 사용자.	생성할 동의어의 본래 객체 소유 사용자를 지정. 생략할 경우 현재 접속한 사용자로 지정(선택)
④ 객체이름	동의어를 생성할 대상 객체 이름(필수)

생성한 동의어는 SELECT, INSERT, UPDATE, DELETE 등 다양한 SQL문에서 사용할 수 있습니다.

☞ 동의어는 여러 종류의 객체와 SQL문에서 사용할 수 있지만 본문에서는 이 책에서 소개한 객체 및 SQL문만 명시하겠습니다. 더 자세한 동의어 사용은 오라클 공식 문서(docs.oracle.com/cd/B28359_01/server.111/b28286/statements_7001.htm#SQLRF01401)를 참고하세요.

동의어는 SELECT문의 SELECT절, FROM절에서 사용한 열 또는 테이블 별칭과 유사하지만, 오라클 데이터베이스에 저장되는 객체이기 때문에 일회성이 아니라는 점에서 차이가 납니다. 동의어 생성 역시 권한을 따로 부여해야 하기 때문에 다음과 같이 SQL*PLUS에서 SYSTEM에 접속하여 SCOTT 계정에 동의어 생성 권한을 부여해 보세요. PUBLIC SYNONYM 권한도 따로 부여해 주어야 합니다.

실습 13-37 권한 부여하기(SQL*PLUS)

```
01  SQLPLUS SYSTEM/oracle  
  
02  GRANT CREATE SYNONYM TO SCOTT;  
  
03  GRANT CREATE PUBLIC SYNONYM TO SCOTT;
```

:: 결과 화면

```
명령 프롬프트 - SQLPLUS 'SYSTEM/oracle'  
  
C:\Users\easy\publishing>SQLPLUS SYSTEM/oracle  
  
SQL*Plus: Release 11.2.0.1.0 Production on 토 7월 14 02:08:39 2018  
  
Copyright (c) 1982, 2010, Oracle. All rights reserved.  
  
다음에 접속됨:  
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production  
With the Partitioning, OLAP, Data Mining and Real Application Testing options  
  
SQL> GRANT CREATE SYNONYM TO SCOTT;  
권한이 부여되었습니다.  
  
SQL> GRANT CREATE PUBLIC SYNONYM TO SCOTT;  
권한이 부여되었습니다.  
  
SQL>
```

SYSTEM 계정으로 접속하여 SCOTT 계정에 권한을 부여해야 합니다.

동의어 생성

동의어 생성 권한을 부여했다면 토드에서 SCOTT 계정으로 접속하여 다음과 같이 EMP 테이블에 동의어 E를 만들어 보죠.

실습 13-38 EMP 테이블의 동의어 생성하기(토드)

```
01  CREATE SYNONYM E  
02    FOR EMP;
```

E 동의어로 SELECT문을 실행하면 EMP 테이블의 데이터가 조회됩니다.

실습 13-39 E 테이블 전체 내용 조회하기(토드)

```
01  SELECT * FROM E;
```

:: 결과 화면

EMPNO	EN...	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	1980/12/17	800		20
7499	ALLEN	SALESMAN	7698	1981/02/20	1600	300	30
7521	WARD	SALESMAN	7698	1981/02/22	1250	500	30
7566	JONES	MANAGER	7839	1981/04/02	2975		20
7654	MARTIN	SALESMAN	7698	1981/09/28	1250	1400	30
7698	BLAKE	MANAGER	7839	1981/05/01	2850		30
7782	CLARK	MANAGER	7839	1981/06/09	2450		10
7788	SCOTT	ANALYST	7566	1987/04/19	3000		20
7839	KING	PRESIDENT		1981/11/17	5000		10
7844	TURNER	SALESMAN	7698	1981/09/08	1500	0	30
7876	ADAMS	CLERK	7788	1987/05/23	1100		20
7900	JAMES	CLERK	7698	1981/12/03	950		30
7902	FORD	ANALYST	7566	1981/12/03	3000		20
7934	MILLER	CLERK	7782	1982/01/23	1300		10

동의어 삭제

DROP SYNONYM을 사용하여 동의어를 삭제합니다. 생성한 E 동의어를 삭제해 볼까요?

실습 13-40 | 동의어 삭제하기(토드)

01 **DROP SYNONYM E;**

동의어를 삭제하면 E 동의어로 SELECT를 할 수 없지만 EMP 테이블 이름과 데이터에는 아무 영향을 주지 않습니다.

지금까지 오라클 데이터베이스 객체에 대해 알아보았습니다. 이 책에서는 오라클 데이터베이스에서 제공하는 많은 객체들 중에서 사용 빈도가 높은 객체들과 필수적인 사용법을 소개하고 있습니다. 여기에서 소개한 내용만으로도 기본 업무를 수행하기에 충분하지만, 이후 주어진 업무에 따라 각 객체의 더 자세한 사용법이나 다른 객체의 사용법이 필요해질 수 있으므로 이 장의 내용을 꼭 익히고 넘어가세요.

Q1 다음 SQL문을 작성해 보세요.

- ① EMP 테이블과 같은 구조의 데이터를 저장하는 EMPIDX 테이블을 만들어 보세요.
- ② 생성한 EMPIDX 테이블의 EMPNO 열에 IDX_EMPIDX_EMPNO 인덱스를 만들어 보세요.
- ③ 마지막으로 인덱스가 잘 생성되었는지 적절한 데이터 사전 뷰를 통해 확인해 보세요.

Q2 문제 1번에서 생성한 EMPIDX 테이블의 데이터 중 급여(SAL)가 1500 초과인 사원들만 출력하는 EMPIDX_OVER15K 뷰를 생성해 보세요. 이 이름을 가진 뷰가 이미 존재할 경우에 새로운 내용으로 대체 가능해야 합니다. EMPIDX_OVER15K 뷰는 사원 번호, 사원 이름, 직책, 부서 번호, 급여, 추가 수당 열을 가지고 있습니다. 추가 수당 열의 경우에 추가 수당이 존재하면 O, 존재하지 않으면 X로 출력합니다.

:: 결과 화면

	EMPNO	ENAME	JOB	DEPTNO	SAL	COMM
▶	7499	ALLEN	SALESMAN	30	1600	O
	7566	JONES	MANAGER	20	2975	X
	7698	BLAKE	MANAGER	30	2850	X
	7782	CLARK	MANAGER	10	2450	X
	7788	SCOTT	ANALYST	20	3000	X
	7839	KING	PRESIDENT	10	5000	X
	7902	FORD	ANALYST	20	3000	X

Q3 다음 세 가지 SQL문을 작성해 보세요.

① DEPT 테이블과 같은 열과 행 구성을 가지는 DEPTSEQ 테이블을 작성해 보세요.

② 생성한 DEPTSEQ 테이블의 DEPTNO 열에 사용할

시퀀스를 오른쪽 특성에 맞게 생성해 보세요.

부서 번호의 시작 값 : 1

부서 번호의 증가 : 1

부서 번호의 최댓값 : 99

부서 번호의 최솟값 : 1

부서 번호 최댓값에서 생성 중단

캐시 없음

③ 마지막으로 생성한 DEPTSEQ를 사용하여 오른쪽과
같이 세 개 부서를 차례대로 추가해 보세요.

부서 이름(DNAME)	부서 위치(LOC)
DATABASE	SEOUL
WEB	BUSAN
MOBILE	ILSAN

:: 결과 화면

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
1	DATABASE	SEOUL
2	WEB	BUSAN
3	MOBILE	ILSAN

정답 이지스퍼블리싱 홈페이지에서 확인하세요.