제 6장
DDL(데이터정의어)

목차

6.1 개체(Object)

●데이터 정의어 (DDL : Data Definition Language)

- ➢데이터베이스의 3층 스키마를 정의
- ➢데이터베이스 여러 개체 기술

●개체(Object)

Object	설 명
Table	행과 열로 구성된 2차원 테이블로 데이터 저장하는 개체
View	하나 이상의 테이블로부터 유도된 데이터의 부분집합 개체
Sequence	순차적인 숫자 값을 생성하는 개체
Index	빠른 검색 위해 사용하는 개체

6.2 테이블(Table)

●테이블(Table)

- 테이블은 데이터를 저장할 수 있는 개체
- 테이블의 생성을 위해서는 CREATE TABLE
- 변경을 위해서는 ALTER TABLE
- 삭제를 위해서는 DROP TABLE

●테이블 생성 SQL

```
SQL> create table testk
  2    (u_id   varchar2(10),
  3     u_date  date);
```

6.2.1 데이터 속성(type)

●데이터 속성(type)

타 입	설 명
NUMBER(n,m)	숫자 데이터에 대한 정의에 사용
CHAR(n)	문자 데이터에 대한 정의에 사용
VARCHAR2(n)	가변길이 문자데이터에 대한 정의에 사용
DATE	날짜 데이터에 대한 정의에 사용
LONG	2GB의 가변길이 문자 데이터에 대한 정의에 사용
TIMESTAMP	년,월,일,시,분,초, 6자리 소수부 초 형태로 시간정보를 정의에 사용
BLOB	4GB의 텍스트
CLOB	4GB의 동영상, 이미지, 사운드 등에 대한 정의에 사용
ROWID	각행에 대한 논리적인 위치(주소), 의사열

6.2.2 테이블 생성

●테이블 생성

```
CREATE TABLE  table-name
        ( column-name1     data-type  default-value,
          column-name2     data-type  default-value,
                  ……
         column-name     data-type  default-value );
```

또는

```
CREATE TABLE table-name
        AS  sub-query;
```

6.2.2 테이블 생성

●STUDENT 테이블 생성

```
SQL> create table student(
  2    stu_no  char(9),
  3    stu_name  varchar2(12),
  4    stu_dept  varchar2(20),
  5    stu_grade  number(1),
  6    stu_class  char(1),
  7    stu_gender  char(1),
  8    stu_height  number(5,2),
  9    stu_weight  number(5,2));

테이블이 생성되었습니다.
```

6.2.2 테이블 생성

●기존의 테이블을 이용하여 새로운 테이블 생성

```
SQL> create table t_student
  2    as select * from student where stu_dept = '기계';

테이블이 생성되었습니다.
```

6.2.2 테이블 생성

●생성된 새로운 테이블 확인

SQL> desc t_student;

이름	널?	유형
STU_NO		CHAR(9)
STU_NAME		VARCHAR2(12)
STU_DEPT		VARCHAR2(20)
STU_GRADE		NUMBER(1)
STU_CLASS		CHAR(1)
STU_GENDER		CHAR(1)
STU_HEIGHT		NUMBER(5,2)
STU_WEIGHT		NUMBER(5,2)

6.2.2 테이블 생성

●새로 생성한 t_student의 데이터를 확인

SQL> select * from t_student;

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20153075	옥한빛	기계	1	C	M	177	80
20153088	이태연	기계	1	C	F	162	50
20143054	유가인	기계	2	C	F	154	47

6.2.3 테이블 변경

●테이블 변경

- ALTER TABLE 명령어 사용
- 새로운 열을 삽입, 기존의 열을 삭제, 기존 열을 변경한다.

(1) 새로운 열 추가

```
ALTER TABLE table-name
        ADD ( column-name1     data-type,
                        column-name2     data-type,
                                                ..... );
```

6.2.3 테이블 변경

- t_student 테이블에 army 열을 삽입

```
SQL> alter table t_student
  2    add ( army    char(1));
```

테이블이 변경되었습니다.

6.2.3 테이블 변경

- 새로운 열 삽입 확인.

SQL> desc t_student;

이름	널?	유형
STU_NO		CHAR(9)
STU_NAME		VARCHAR2(12)
STU_DEPT		VARCHAR2(20)
STU_GRADE		NUMBER(1)
STU_CLASS		CHAR(1)
STU_GENDER		CHAR(1)
STU_HEIGHT		NUMBER(5,2)
STU_WEIGHT		NUMBER(5,2)
ARMY		CHAR(1)

6.2.3 테이블 변경

(2) 열 구조 변경

```
ALTER TABLE table-name
        MODIFY ( column-name1     data-type,
                 column-name2     data-type,
                                  ..... );
```

6.2.3 테이블 변경

- t_student 테이블에 army 열의 구조를 변경

```
SQL> alter table t_student
  2    modify( army    number);
```

테이블이 변경되었습니다.

6.2.3 테이블 변경

- army 열 변경 확인

SQL> desc t_student;

이름	널?	유형
STU_NO		CHAR(9)
STU_NAME		VARCHAR2(12)
STU_DEPT		VARCHAR2(20)
STU_GRADE		NUMBER(1)
STU_CLASS		CHAR(1)
STU_GENDER		CHAR(1)
STU_HEIGHT		NUMBER(5,2)
STU_WEIGHT		NUMBER(5,2)
ARMY		NUMBER

6.2.3 테이블 변경

(3) 열의 삭제

```
ALTER TABLE table-name
        DROP ( column-name1 ,   column-name2   ····· );
```

6.2.3 테이블 변경

- t_student 테이블에 army 열 삭제

```
SQL> alter table t_student
  2     drop ( army);
```

테이블이 변경되었습니다.

6.2.3 테이블 변경

- army 열 삭제 확인

SQL> desc t_student;

이름	널?	유형
STU_NO		CHAR(9)
STU_NAME		VARCHAR2(12)
STU_DEPT		VARCHAR2(20)
STU_GRADE		NUMBER(1)
STU_CLASS		CHAR(1)
STU_GENDER		CHAR(1)
STU_HEIGHT		NUMBER(5,2)
STU_WEIGHT		NUMBER(5,2)

6.2.4 테이블 이름 변경

●테이블 이름변경

```
RENAME old_table_name TO new_table_name;
```

●t_student 테이블 이름을 test_student로 변경

SQL> rename t_student to test_student;

테이블이 변경되었습니다.

6.2.4 테이블 이름 변경

- 이름변경 확인

SQL> desc t_student;

ERROR : ORA-04043: 객체 t_student가 존재하지 않습니다

SQL> desc test_student;

이름	널?	유형
STU_NO		CHAR(9)
STU_NAME		VARCHAR2(12)
STU_DEPT		VARCHAR2(20)
STU_GRADE		NUMBER(1)
STU_CLASS		CHAR(1)
STU_GENDER		CHAR(1)
STU_HEIGHT		NUMBER(5,2)
STU_WEIGHT		NUMBER(5,2)

6.2.5 테이블내의 데이터 삭제

●테이블 내의 데이터 삭제

```
TRUNCATE  TABLE  table-name
```

●test_student 내의 모든 데이터 삭제

SQL> truncate table test_student;

테이블이 잘렸습니다.

●test_student 테이블 검색

SQL> select * from test_student;

선택된 레코드가 없습니다.

6.2.6 테이블 삭제

●테이블 삭제

```
DROP TABLE table-name;
```

●test_student 테이블 삭제

SQL> drop table test_student;

테이블이 삭제되었습니다.

●test_student 테이블 검색

SQL> desc test_student;

ERROR : ORA-04043: 객체 test_student가 존재하지 않습니다.

6.3 제약조건

- **Constraints(제약조건)**
  - 제약조건은 데이터베이스 상태가 항상 만족해야 할 기본 규칙
  - 제약조건은 데이터베이스 스키마에 명시
  - 데이터가 삽입, 삭제, 수정 등의 연산이 일어나도 지속적으로 만족

(1) 도메인 제약 조건

- 각 열의 값은 반드시 해당 도메인에 속하는 원자

(2) 키 제약 조건

- 테이블에는 테이블의 각 레코드를 유일하게 식별할 수 있는 수단
  즉, 최소한 하나의 기본키를 가지고 있어야 한다.

6.3 제약조건

(3) 무결성 제약 조건(integrity constraint)

- **엔티티 무결성(entity integrity)**
  - 기본키 속성들의 값은 어떠한 경우에도 널 값을 가질 수 없다.

- **참조 무결성(referential integrity)**
  - 한 테이블에 있는 레코드가 다른 테이블에 있는 레코드를 참조하려면 반드시 참조되는 레코드가 그 테이블 내에 존재해야 한다.(외래키)

6.3 제약조건

- 오라클 제약조건 5가지 유형

제약 조건	설 명
NOT NULL	열에 NULL값을 허용하지 않음
UNIQUE KEY	열 또는 열의 조합이 유일성(유일한 값)을 가져야 함
PRIMARY KEY	열 또는 열의 조합이 NULL값이 아니며, 유일성을 가져야 함
FOREIGN KEY	다른 테이블의 열을 참조되는 테이블에 값이 존재하여야 함
CHECK	열에 들어갈 값에 대한 조건을 명시함

6.3.1 NOT NULL

- NOT NULL
  - 해당 열에 NULL 값이 저장되어서는 안 되는 경우 사용
  - Primary Key 제약조건은 NOT NULL 조건 만족

```
SQL> create table t_student(
  2   stu_no  char(9),
  3   stu_name  varchar2(12),
  4   stu_dept varchar2(20)
  5     constraint n_stu_dept not null,
  6   stu_grade   number(1),
  7   stu_class    char(1),
  8   stu_gender  char(1),
  9   stu_height   number(5,2),
 10   stu_weight   number(5,2));
```

6.3.2 UNIQUE KEY

- **UNIQUE KEY**
  - 기본 키가 아닌 열의 값이 유일한 값을 유지하여야 할 때 사용
  - 자동으로 INDEX가 만들어진다.

```
SQL> create table t_student(
  2   stu_no  char(9),
  3   stu_name  varchar2(12)
  4     constraint u_stu_name unique,
  5   stu_dept varchar2(20)
  6     constraint n_stu_dept not null,
  7   stu_grade   number(1),
  8   stu_class    char(1),
  9   stu_gender  char(1),
 10   stu_height   number(5,2),
 11   stu_weight   number(5,2));
```

6.3.3 PRIMARY KEY

- PRIMARY KEY
  - 기본 키는 한 테이블에 단 한 개만 존재
  - 자동으로 INDEX가 만들어진다.

```
SQL> create table t_student(
  2   stu_no  char(9),
  3   stu_name  varchar2(12)
  4     constraint u_stu_name unique,
  5   stu_dept varchar2(20)
  6     constraint n_stu_dept not null,
  7   stu_grade   number(1),
  8   stu_class    char(1),
  9   stu_gender  char(1),
 10   stu_height   number(5,2),
 11   stu_weight   number(5,2));
 12     constraint p_stu_no primary key(stu_no)
```

6.3.3 PRIMARY KEY

```
SQL> create table t_enrol(
  2   sub_no  char(3),
  3   stu_no   char(9),
  4   enr_grade  number(3),
  5     constraint p_enol primary key(sub_no,stu_no));
```

6.3.4 FOREIGN KEY

- FOREIGN KEY
  - 참조 무결성을 유지하기 위해 자식 테이블의 열을 외래 키로 선언
  - Enrol 테이블의 sub_no, stu_no는 외래 키이며, 부모 테이블은 각각 subject, student 이다.

```
SQL> create table t_enrol(
  2   sub_no  char(3),
  3   stu_no   char(9),
  4   enr_grade  number(3)
  5   constraint enr_stu_no foreign key(stu_no)
  8                                  references  t_student(stu_no),
  9    constraint p_enol primary key(sub_no,stu_no));
```

6.3.5 CHECK

- CHECK
  - 어떤 열의 값에 조건을 제시

```
SQL> create table t_student(
  2   stu_no  char(9),
  3   stu_name  varchar2(12)
  4    constraint u_stu_name unique,
  5   stu_dept varchar2(20)
  6    constraint n_stu_dept not null,
  7   stu_grade   number(1),
  8   stu_class    char(1),
  9   stu_gender  char(1),
 10    constraint c_stu_gender check (stu_gender in('M','F'))
 11   stu_height   number(5,2),
 12   stu_weight   number(5,2));
 13    constraint p_stu_no primary key(stu_no) );
```

6.3.6 제약 조건의 확인

- 제약 조건의 확인
  - 데이터 사전의 테이블은 USER_CONSTRAINTS

```
SQL> select *
  2  from user_constraints
  3  where table_name = 'T_STUDENT';
```

6.3.7 제약 조건의 삭제

●제약 조건의 삭제

➢제약조건을 삭제할 경우, ALTER TABLE 명령 사용

```
ALTER TABLE table-name
        DROP CONSTRAINT constraint_name [CASCADE];
```

```
SQL> alter table t_enrol
  2    drop constraint enr_stu_no cascade;
```

6.3.8 제약조건의 활성화, 비활성화

●제약 조건의 활성화, 비활성화

➢제약조건을 비활성화 할 경우, ALTER TABLE 명령을 사용

```
ALTER TABLE table-name
    DISABLE | ENABLE CONSTRAINT constraint_name [CASCADE];
```

```
SQL> alter table t_student
  2  disable constraint n_stu_dept;
```

```
SQL> alter table t_student
  2  enable constraint n_stu_dept;
```

6.4 뷰(View)

- **뷰(VIEW)**
  - 가상의 테이블 (virtual table)
  - 뷰의 장점
    ① 뷰는 데이터베이스를 재구성하여 논리적 데이터 독립성 제공
    ② 원하는 데이터만을 조작함으로 데이터의 보완기능 강화

```
CREATE  VIEW  view-name
          [ col-name [, col-name  ……. ]]
AS
      subquery;
```

```
DROP VIEW view-name;
```

6.4 뷰(View)

- 단순 뷰(VIEW)
  - 단순 뷰는 단일 베이스 테이블로 부터 유도된 뷰이다.

```
SQL> create or replace view v_student1
  2    as select * from student where stu_dept = '컴퓨터정보';

뷰가 생성되었습니다.
```

6.4 뷰(View)

●뷰의 생성을 확인

```
SQL> select *
  2  from v_student1;
```

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20151062	김인중	컴퓨터정보	1	B	M	166	67
20141007	진현무	컴퓨터정보	2	A	M	174	64
20131001	김종현	컴퓨터정보	3	C	M		72
20131025	옥성우	컴퓨터정보	3	A	F	172	63

6.4 뷰(View)

●조인 뷰(VIEW)

➢조인뷰는 2개 이상의 베이스 테이블로부터 유도된 뷰

```
SQL> create or replace view v_enrol1
  2  as select sub_name, a.sub_no, stu_no, enr_grade
  3  from enrol a, subject b
  4  where a.sub_no = b.sub_no;
```

뷰가 생성되었습니다.

6.4 뷰(View)

●뷰의 생성을 확인

```
SQL> select *
  2  from v_enrol1;
```

SUB_NAME	SUB_NO	STU_NO	ENR_GRADE
컴퓨터개론	101	20131001	80
컴퓨터개론	101	20131025	65
기계공작법	102	20153075	66
기계공작법	102	20153088	61
기초전자실험	103	20152088	45
시스템분석설계	104	20131001	56
시스템분석설계	104	20131025	65
기계요소설계	105	20153088	78
기계요소설계	105	20153075	56
전자회로실험	106	20132003	72
CAD응용실습	107	20143054	41
소프트웨어공학	108	20151062	81

6.4 뷰(View)

● 인라인(INLINE)뷰

- 인라인 뷰는 FROM절에 SELECT문으로 정의된 뷰이다.
- 학과별 평균 신장보다 큰 학생들의 학번, 이름, 신장 검색

```
SQL> select stu_no, stu_name, a.stu_dept, stu_height
  2  from student a, (select stu_dept, avg(stu_height)
  3                   as avg_height
  4                   from student
  5                   group by stu_dept) b
  6  where a.stu_dept = b.stu_dept
  7  and a.stu_height > b.avg_height
```

STU_NO	STU_NAME	STU_DEPT	STU_HEIGHT
20153075	옥한빛	기계	177
20152088	조민우	전기전자	188
20141007	진현무	컴퓨터정보	174
20131025	옥성우	컴퓨터정보	172

6.4 뷰(View)

- **TOP-N 질의**
  - "신장이 큰 상위 5명의 학생" 또는 "성적이 높은 상위 5명의 학생"과 같이 최댓값 또는 최솟값을 가진 열에 몇 개의 레코드만 추출할 때 사용되는 기능

```
SELECT item-commalist
        FROM  (SELECT item-commalist
                        FROM table name
                        ORDER BY item)
WHERE ROWNUM <= n;
```

6.4 뷰(View)

- 학생 테이블에서 신장이 큰 상위 5명의 학번, 이름, 신장 검색

```
SQL> select stu_no, stu_name, stu_height
  2  from (select stu_no, stu_name, stu_height
  3        from student
  4        where stu_height is not null
  5        order by stu_height desc)
  6  where rownum <= 5;
```

STU_NO	STU_NAME	STU_HEIGHT
20152088	조민우	188
20153075	옥한빛	177
20141007	진현무	174
20131025	옥성우	172
20142021	심수정	168

6.5 인덱스(Index)

●인덱스

- 빠른 검색을 위하여 정의되며, 한 개 이상의 열로 구성됨
- 학생 테이블의 학번은 기본 키이므로 자동으로 인덱스 정의
- 이름으로 검색하는 질의가 많을 경우 이름으로 인덱스를 정의
- 학과로 검색하는 경우가 극히 적다면 학과를 인덱스로 정의할 경우 활용도가 떨어지며, 기억공간은 낭비
- 데이터사전의 USER_INDEXES, USER_ID_COLUMNS

6.5 인덱스(Index)

●학생의 이름으로 인덱스 생성

SQL> create index i_stu_name on student(stu_name)

● 학생의 학번과 이름을 합쳐서 인덱스를 생성

SQL> create index i_stu_no_name on student(stu_no, stu_name)

● 유일한 값으로 인덱스 생성

SQL> create unique index i_stu_name on student(stu_name)

● 함수나 수식을 이용하여 인덱스 생성

SQL> create index i_stu_weight on student(stu_weight-5);

6.5 인덱스(Index)

●인덱스 생성 유무 및 상태 분석

```
SQL> select * from user_indexes
  2    where table_name = 'STUDENT';
```

INDEX_NAME	INDEX_TYPE	TABLE_OWNER	……
STU_NO_PK	NORMAL	SCOTT	……
I_STU_NAME	NORMAL	SCOTT	……
I_STU_NO_NAME	NORMAL	SCOTT	……
I_STU_WEIGHT	FUNCTION-BASED NORMAL	SCOTT	……

●인덱스 삭제

```
SQL> drop index i_stu_name;
```

6.6 시퀀스(Sequence)

● 시퀀스(Sequence)

- 어떤 연속적인 숫자 값을 자동적으로 증가하여 사용

```
CREATE  SEQUENCE  sequence-name
        INCREMENT BY n
        START WITH n
        MAXVALUE   n   |  NOMAXVALUE
        MINVALUE   n   | NOMINVALUE
        CYCLE   |   NOCYLE
        CACHE  |   NOCACHE
```

```
DROP  SEQUENCE  sequence-name
```

6.6 시퀀스(Sequence)

```
SQL> create sequence seq1
  2  increment by 2
  3  start with 1000
  4  maxvalue 10000;

주문번호가 생성되었습니다.
```

6.6 시퀀스(Sequence)

SQL> select seq1.nextval from dual;

NEXTVAL
1000

SQL> run

NEXTVAL
1002

SQL> select seq1.nextval from dual;

NEXTVAL
1004