Chapter 11. 제약조건

제약조건은 데이터의 무결성(Integrity)을 지켜주는 중요한 역할을 수행한다. 이번 장에서는 제약조건의 정의와 종류를 살펴보고 이러한 제약조건을 생성 및 관리하는 방법을 알아본다.

제약조건의 정의

Oracle 서버는 제약조건을 이용하여 적절치 않은 데이터가 테이블에 입력되는 것을 방지하며 다음과 같은 장점이 있다.

- 테이블에 행이 입력, 갱신, 삭제 될 때마다 지정된 제약 조건을 반드시 만족해야하므로 데이터의 무결성을 강화시켜준다.
- 다른 테이블에 의해 참조 되고 있는 테이블이 삭제 되는 것을 방지한다.

제약조건을 정의 할 때는 다음 사항을 참고하도록 한다.

- 제약조건에 적절한 이름을 지정하도록 한다. 그렇지 않으면 Oracle 서버가 SYS_C*n* 형식으로 임의 지정하기 때문에 제약조건의 이름으로 어떤 종류의 제약조건인지 식별하기가 어려워진다.
- 제약조건은 테이블의 생성과 동시에 작성되도록 할 수 있으며, 테이블이 생성된 이후에도 추가 할 수 있다.
- 제약조건은 테이블 수준 또는 컬럼 수준에서 정의 할 수 있다.
- 제약조건은 USER_CONSTRAINTS 데이터 딕셔너리를 이용하여 검색 할 수 있다.

제약조건을 정의하는 문법은 다음과 같다.

```
CREATE TABLE [schema.]table
            (column datatype [DEFAULT expr]
            [column_constraint],
            ...
            [table_constraint][, ...]);
```

제약조건을 컬럼 수준에서 정의하려면 *column_constraint*에 제약조건을 기술한다. 컬럼 수준의 제약조건은 한 개의 컬럼에 한 개의 제약조건만 정의 할 수 있으며, 모든 제약조건 타입을 기술 할 수 있다. 작성하는 방법은 다음과 같이 테이블에 대한 컬럼을 정의 한 후, 제약조건을 기술한다.

```
column [CONSTRAINT constraint_name] constraint_type,
```

제약조건을 테이블 수준에서 정의하려면 *table_constraint*에 제약조건을 기술해주며 한 개 이상의 컬럼에 한 개의 제약조건을 정의 할 수 있고, NOT NULL 제약조건을 제외한 모든 제약조건 타입을 기술 할 수 있다. 작성하는 방법은 테이블의 컬럼 정의와 별도로 기술한

다.

```
 column, ...
   [CONSTRAINT constraint_name] constraint_type
   (column, ...),
```

NOT NULL

NOT NULL 제약조건은 해당 컬럼에 NULL 값이 입력되지 않도록 제한하는 것으로, NOT NULL 제약조건이 정의되지 않은 컬럼은 디폴트로 NULL 값의 저장이 허용된다. NOT NULL 제약조건은 컬럼 수준에서만 지정가능하며 사용방법은 다음과 같다.

```
SQL> CREATE TABLE SAWON (
  2  S_NO NUMBER(4),
  3  S_NAME VARCHAR2(10) NOT NULL,
  4  S_HIREDATE DATE CONSTRAINT SAWON_S_HIREDATE_NN NOT NULL);

테이블이 생성되었습니다.

SQL> INSERT INTO SAWON
  2  VALUES(1, '길동', NULL);
INSERT INTO SAWON
*
1행에 오류:
ORA-01400: NULL을 ("SCOTT"."SAWON"."S_HIREDATE") 안에 삽입할 수 없습니다
```

위에서 SAWON 테이블의 S_NAME 컬럼과 S_HIREDATE 컬럼에 NOT NULL 제약조건이 지정되었으며, 해당 컬럼에 NULL 값을 입력하려고 시도하면 제약조건에 위배되어 오류가 발생한다. S_NAME 컬럼과 같이 제약조건 선언시 제약조건 이름을 지정하지 않으면 Oracle 서버가 임의로 제약조건 이름을 정의하게 된다. 제약조건을 검색하는 방법은 다음과 같다.

```
SQL> SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE
  2  FROM USER_CONSTRAINTS
  3  WHERE TABLE_NAME = 'SAWON';

CONSTRAINT_NAME                C
------------------------------ -
SAWON_S_HIREDATE_NN            C
SYS_C003005                    C
```

UNIQUE

UNIQUE 제약조건은 컬럼에 중복된 값이 입력되지 않도록 제한하는 것으로, 한 개 이상의 컬럼으로 구성 할 수 있다. UNIQUE 제약조건이 지정된 컬럼에 별도의 NOT NULL 제약조건을 지정하지 않는 한, NULL 값이 중복되어 입력 될 수 있으며, 한 개의 컬럼으로 UNIQUE 제약조건을 구성 할 때는 컬럼 수준 및 테이블 수준에서 제약조건의 정의가 가능하지만, 두 개 이상의 컬럼으로 구성하는 경우는 테이블 수준에서만 정의 할 수 있다. UNIQUE 제약조건이 지정된 컬럼을 고유키라고도 부르며 사용방법은 다음과 같다.

```
SQL> CREATE TABLE SAWON (
  2  S_NO NUMBER(4),
  3  S_NAME VARCHAR2(10),
  4  S_SAL NUMBER(10),
  5  S_EMAIL VARCHAR2(20) CONSTRAINT SAWON_S_EMAIL_UK UNIQUE);

테이블이 생성되었습니다.

SQL> INSERT INTO SAWON VALUES (1, '길동', 1000, 'GDH@XYZ.COM');

1 개의 행이 만들어졌습니다.

SQL> INSERT INTO SAWON VALUES (2, '콩쥐', 2000, 'GDH@XYZ.COM');
INSERT INTO SAWON VALUES (2, '콩쥐', 2000, 'GDH@XYZ.COM')
*
1행에 오류:
ORA-00001: 무결성 제약 조건(SCOTT.SAWON_S_EMAIL_UK)에 위배됩니다

SQL> INSERT INTO SAWON VALUES (3, '팥쥐', 1500, NULL);

1 개의 행이 만들어졌습니다.
```

위의 UNIQUE 제약조건을 테이블 수준에서 정의하면 다음과 같다.

```
CREATE TABLE SAWON (
  S_NO NUMBER(4),
  S_NAME VARCHAR2(10),
  S_SAL NUMBER(10),
  S_EMAIL VARCHAR2(20),
  CONSTRAINT SAWON_S_EMAIL_UK UNIQUE (S_EMAIL));
```

두 개 이상의 컬럼으로 UNIQUE 제약조건을 구성 할 때는 테이블 수준에서 정의하여야만 한다.

```
CREATE TABLE SAWON (
  S_NO NUMBER(4),
  S_NAME VARCHAR2(10),
  S_SAL NUMBER(10),
  S_COMM NUMBER(10),
  CONSTRAINT SAWON_UK UNIQUE (S_NO, S_NAME));
```

또한, UNIQUE 제약조건을 지정하면 해당 컬럼에 UNIQUE INDEX가 생성된다.

PRIMARY KEY

PRIMARY KEY 제약조건은 테이블에 기본키를 작성하는 것으로 한 개의 테이블에는 오직 한 개의 기본키를 만들 수 있다. PRIMARY KEY 제약조건은 해당 컬럼에 중복된 값과 NULL 값이 입력되지 않도록 제한하며, 한 개 이상의 컬럼으로 PRIMARY KEY 제약조건을 구성 할 수 있다. 한 개의 컬럼으로 PRIMARY KEY 제약조건을 구성하는 경우에는 컬럼 수준 및 테이블 수준에서 제약조건의 정의가 가능하지만, 두 개 이상의 컬럼으로 PRIMARY KEY 제약조건을 구성하는 경우 반드시 테이블 수준에서 제약조건을 정의해야 한다.

```
SQL> CREATE TABLE SAWON (
  2  S_NO NUMBER(4) CONSTRAINT SAWON_S_NO_PK PRIMARY KEY,
  3  S_NAME VARCHAR2(10),
  4  S_SAL NUMBER(10));

테이블이 생성되었습니다.

SQL> INSERT INTO SAWON VALUES (1, '길동', 1000);

1 개의 행이 만들어졌습니다.

SQL> INSERT INTO SAWON VALUES (1, '콩쥐', 2000);
INSERT INTO SAWON VALUES (1, '콩쥐', 2000)
*
1행에 오류:
ORA-00001: 무결성 제약 조건(SCOTT.SAWON_S_NO_PK)에 위배됩니다

SQL> INSERT INTO SAWON VALUES (NULL, '팥쥐', 3000);
INSERT INTO SAWON VALUES (NULL, '팥쥐', 3000)
*
1행에 오류:
ORA-01400: NULL을 ("SCOTT"."SAWON"."S_NO") 안에 삽입할 수 없습니다
```

위의 PRIMARY KEY 제약조건을 테이블 수준에서 정의하면 다음과 같다.

```
CREATE TABLE SAWON (
  S_NO NUMBER(4),
  S_NAME VARCHAR2(10),
  S_SAL NUMBER(10),
  CONSTRAINT SAWON_S_NO_PK PRIMARY KEY (S_NO));
```

두 개 이상의 컬럼으로 PRIMARY KEY 제약조건을 구성하려면 테이블 수준에서 정의하여야만 한다.

```
CREATE TABLE SAWON (
  S_NO NUMBER(4),
  S_NAME VARCHAR2(10),
  S_SAL NUMBER(10),
  CONSTRAINT SAWON_PK PRIMARY KEY (S_NO, S_NAME));
```

또한, PRIMARY KEY 제약조건을 지정하면 UNIQUE 제약조건과 마찬가지로 해당 컬럼에 UNIQUE INDEX가 생성된다.

FOREIGN KEY

FOREIGN KEY 제약조건은 참조 무결성 제약조건이라고 부르며 외래키 컬럼에 지정되어 반드시 다른 테이블의 기본키나 고유키 컬럼의 값을 참조하도록 제한하는 것이다. FOREIGN KEY 제약조건이 지정된 컬럼에 별도의 NOT NULL 제약조건이 지정되지 않으면 NULL 값도 입력 가능하다. 한 개의 컬럼으로 FOREIGN KEY 제약조건을 구성하는 경우에는 컬럼 수준 및 테이블 수준에서 제약조건의 정의가 가능하지만, 두 개 이상의 컬럼으로 FOREIGN KEY 제약조건을 구성하는 경우 반드시 테이블 수준에서 제약조건을 정의해

야 한다.

FOREIGN KEY 제약조건을 설정하기에 앞서 외래키가 참조할 기본키가 있는 테이블을 먼저 작성해보자

```
SQL> CREATE TABLE BUSE0(
  2  B_NO NUMBER(4) CONSTRAINT BUSE0_B_NO_PK PRIMARY KEY,
  3  B_NAME VARCHAR2(10),
  4  B_LOC VARCHAR2(10));

테이블이 생성되었습니다.

SQL> INSERT INTO BUSE0 VALUES (100, '인사과', '서울');

1 개의 행이 만들어졌습니다.

SQL> INSERT INTO BUSE0 VALUES (200, '총무과', '대전');

1 개의 행이 만들어졌습니다.

SQL> INSERT INTO BUSE0 VALUES (300, '경리과', '부산');

1 개의 행이 만들어졌습니다.
```

부서 테이블을 참조하는 사원 테이블의 외래키에 FOREIGN KEY 제약조건을 지정하여 데이터를 입력하면 다음과 같다.

```
SQL> CREATE TABLE SAWON(
  2  S_NO NUMBER(4) CONSTRAINT SAWON_S_NO_PK PRIMARY KEY,
  3  S_NAME VARCHAR2(10),
  4  S_SAL NUMBER(5),
  5  B_NO NUMBER(4) CONSTRAINT SAWON_B_NO_FK REFERENCES BUSE0(B_NO));

테이블이 생성되었습니다.

SQL> INSERT INTO SAWON VALUES (1, '길동', 1000, 100);

1 개의 행이 만들어졌습니다.

SQL> INSERT INTO SAWON VALUES (2, '콩쥐', 2000, 150);
INSERT INTO SAWON VALUES (2, '콩쥐', 2000, 150)
*
1행에 오류:
ORA-02291: 무결성 제약조건(SCOTT.SAWON_B_NO_FK)이 위배되었습니다- 부모 키가
없습니다

SQL> INSERT INTO SAWON VALUES (3, '팥쥐', 3000, NULL);

1 개의 행이 만들어졌습니다.
```

위의 제약조건을 테이블 수준에서 정의하면 다음과 같다.

```
CREATE TABLE SAWON(
  S_NO NUMBER(4) CONSTRAINT SAWON_S_NO_PK PRIMARY KEY,
  S_NAME VARCHAR(10),
  S_SAL NUMBER(5),
  B_NO NUMBER(4),
  CONSTRAINT SAWON_B_NO_FK FOREIGN KEY (B_NO) REFERENCES BUSEO(B_NO));
```

위에서 보는 것과 같이 FOREIGN KEY 구문은 제약조건을 컬럼 수준에서 정의하는 경우에는 기술 할 필요가 없으며, 테이블 수준에서 정의하는 경우에만 사용한다. REFERENCES 구문 뒤에는 참조할 테이블과 컬럼을 기술해주며 다음과 같은 옵션을 추가 할 수 있다.

- ON DELETE CASCADE : FOREIGN KEY 제약조건에 의해 참조되는 테이블(부모 테이블)의 행이 삭제되면, 해당 행을 참조하는 테이블(자식 테이블)의 행도 삭제되도록 한다.
- ON DELETE SET NULL : FOREIGN KEY 제약조건에 의해 참조되는 테이블(부모 테이블)의 행이 삭제되면, 해당 행을 참조하는 테이블(자식 테이블)의 외래키 컬럼을 NULL로 변경한다.

CHECK

CHECK 제약조건은 해당 컬럼의 값이 반드시 만족해야 될 조건을 지정하는 것으로 제약조건에 CURRVAL, NEXTVAL, LEVEL, ROWNUM과 같은 가상 컬럼(Pseudocolumn)과 SYSDATE, UID, USER, USERENV 함수를 호출 할 수 없으며 다른 행의 값도 참조 할 수 없다. CHECK 제약조건은 컬럼 수준 및 테이블 수준에서 정의 할 수 있다.

```
SQL> CREATE TABLE SAWON (
  2  S_NO NUMBER(4),
  3  S_NAME VARCHAR2(10),
  4  S_SAL NUMBER(10) CONSTRAINT SAWON_S_SAL_CK CHECK (S_SAL > 0));

테이블이 생성되었습니다.

SQL> INSERT INTO SAWON VALUES (1, '길동', 1000);

1 개의 행이 만들어졌습니다.

SQL> INSERT INTO SAWON VALUES (2, '콩쥐', -1000);
INSERT INTO SAWON VALUES (2, '콩쥐', -1000)
*
1행에 오류:
ORA-02290: 체크 제약조건(SCOTT.SAWON_S_SAL_CK)이 위배되었습니다
```

위 제약조건을 테이블 수준에서 정의하면 다음과 같다.

```
CREATE TABLE SAWON (
  S_NO NUMBER(4),
  S_NAME VARCHAR2(10),
  S_SAL NUMBER(10),
  CONSTRAINT SAWON_S_SAL_CK CHECK (S_SAL > 0));
```

제약조건의 관리

ALTER TABLE 명령을 이용하면 기존의 테이블에 제약조건을 추가 할 수 있으며, 기존 제약조건을 활성화하거나 비활성화 할 수 있다.

■ 제약조건의 추가

기존 테이블에 제약조건을 추가하는 명령은 다음과 같다. 단, NOT NULL 제약조건의 경우는 컬럼 변경 명령인 ALTER TABLE ... MODIFY 명령을 사용해야만 한다.

```
ALTER TABLE table
ADD [CONSTRAINT constraint] type (column);
```

기존 테이블에 제약조건을 설정하는 방법은 다음과 같다. SAWON 테이블의 S_MGR 컬럼 값이 S_NO 컬럼에 존재하는 값을 참조하도록 FOREIGN KEY 제약조건을 추가하였다.

```
SQL> CREATE TABLE SAWON (
  2  S_NO NUMBER(4) CONSTRAINT SAWON_S_NO_PK PRIMARY KEY,
  3  S_NAME VARCHAR2(10),
  4  S_MGR NUMBER(4));

테이블이 생성되었습니다.

SQL> ALTER TABLE SAWON
  2  ADD CONSTRAINT SAWON_S_MGR_FK FOREIGN KEY (S_MGR)
  3  REFERENCES SAWON(S_NO);

테이블이 변경되었습니다.
```

■ 제약조건의 삭제

기존 테이블에서 제약조건을 삭제하는 방법은 다음과 같다.

```
SQL> ALTER TABLE SAWON
  2  DROP CONSTRAINT SAWON_S_MGR_FK;

테이블이 변경되었습니다.
```

PRIMARY KEY 제약조건을 삭제했을 때, 해당 기본키를 참조하는 FOREIGN KEY 제약조건을 동시에 삭제하는 명령은 다음과 같다.

```
SQL> ALTER TABLE SAWON
  2  DROP PRIMARY KEY CASCADE;

테이블이 변경되었습니다.
```

■ 제약조건 비활성화

ALTER TABLE 명령의 DISABLE 구문을 이용하면 기존 제약조건을 잠시 비활성화 할 수 있다. 비활성화 할 제약조건에 CASCADE 옵션을 추가하면 해당 제약조건과 관련된 제약조건을 모두 비활성화 할 수 있다.

```
SQL> CREATE TABLE SAWON (
  2  S_NO NUMBER(4) CONSTRAINT SAWON_S_NO_PK PRIMARY KEY,
  3  S_NAME VARCHAR2(10),
  4  S_MGR NUMBER(4));

테이블이 생성되었습니다.

SQL> ALTER TABLE SAWON
  2  DISABLE CONSTRAINT SAWON_S_NO_PK;

테이블이 변경되었습니다.
```

PRIMARY KEY 제약조건과 UNIQUE 제약조건을 비활성화하면 해당 제약조건에 생성된 인덱스는 자동으로 삭제된다.

■ 제약조건 활성화

비활성화 된 제약조건은 ALTER TABLE 명령의 ENABLE 구문으로 다시 활성화 할 수 있다. 제약조건이 활성화 되면 해당 테이블에 입력된 모든 행에 적용되며 반드시 모든 데이터는 제약조건에 만족해야한다.

```
SQL> ALTER TABLE SAWON
  2  ENABLE CONSTRAINT SAWON_S_NO_PK;

테이블이 변경되었습니다.
```

만약, CASCADE 옵션으로 비활성화된 관련 제약조건은 ENABLE 명령으로도 활성화되지 않음을 주의해야 한다. 즉, PRIMARY KEY 제약조건을 CASCADE 옵션으로 비활성화 하면 해당 기본키를 참조하는 FOREIGN KEY 제약조건도 비활성화 되지만 이후, PRIMARY KEY 제약조건을 활성화하면 해당 FOREIGN KEY 제약조건은 활성화되지 않는다. 그리고, PRIAMRY KEY 제약조건과 UNIQUE 제약조건이 활성화되면 삭제된 인덱스가 다시 생성된다.

■ CASCADE CONSTRAINTS

기존 테이블에서 컬럼을 삭제하는 ALTER TABLE ... DROP 명령에 CASCADE CONSTRAINTS 구문을 추가하면 PRIMARY KEY 또는 UNIQUE 제약조건이 지정된 컬럼을 삭제했을 때, 해당 컬럼을 참조하는 모든 제약조건을 삭제 할 수 있다. 또한, 삭제된 컬럼이 포함되어 있는 모든 제약조건도 삭제된다.

```
SQL> CREATE TABLE TEST1 (
  2  PK NUMBER PRIMARY KEY,
  3  FK NUMBER,
  4  C1 NUMBER,
  5  C2 NUMBER,
  6  CONSTRAINT FK_CONSTRAINT FOREIGN KEY (FK) REFERENCES TEST1,
  7  CONSTRAINT CK1 CHECK (PK > 0 AND C1 > 0),
  8  CONSTRAINT CK2 CHECK (C2 > 0));

테이블이 생성되었습니다.

SQL> ALTER TABLE TEST1 DROP (PK);
ALTER TABLE TEST1 DROP (PK)
                        *
1행에 오류:
ORA-12992: 부모 키 열을 삭제할 수 없습니다


SQL> ALTER TABLE TEST1 DROP (C1);
ALTER TABLE TEST1 DROP (C1)
                        *
1행에 오류:
ORA-12991: 열이 다중-열 제약 조건에 참조되었습니다
```

위에서 PK 컬럼을 CASCADE CONSTRAINTS 구문을 이용하여 삭제하면, PRIMARY KEY 제약조건과 해당 컬럼을 참조하는 FOREIGN KEY 제약조건 및 CK1 CHECK 제약조건이 동시에 삭제된다.

```
SQL> ALTER TABLE TEST1 DROP (PK) CASCADE CONSTRAINTS;

테이블이 변경되었습니다.
```

만약, CASCADE CONSTRAINTS 구문을 사용하지 않으려면 관련 제약조건들이 지정된 모든 컬럼을 동시에 삭제하면 된다.

```
SQL> ALTER TABLE TEST1 DROP (PK, FK, C1);

테이블이 변경되었습니다.
```

제약조건의 확인

USER_CONSTRAINTS 뷰를 이용하면 제약조건의 정의와 이름을 아래와 같이 확인 할 수 있다.

```
SQL> SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, SEARCH_CONDITION
  2  FROM USER_CONSTRAINTS
  3  WHERE TABLE_NAME = 'EMP';

CONSTRAINT_NAME                C SEARCH_CON
------------------------------ - ----------
PK_EMP                         P
FK_DEPTNO                      R
```

위 결과에서 CONSTRAINT_TYPE 컬럼의 값이 C인 경우는 CHECK 또는 NOT NULL 제약조건, P인 경우는 PRIMARY KEY, R인 경우는 FOREIGN KEY, U인 경우는 UNIQUE 제약조건을 의미한다.

USER_CONS_COLUMNS 뷰를 이용하면 제약조건이 지정된 컬럼명도 확인 할 수 있다.

```
SQL> SELECT CONSTRAINT_NAME, COLUMN_NAME
  2  FROM USER_CONS_COLUMNS
  3  WHERE TABLE_NAME = 'EMP';

CONSTRAINT_NAME                COLUMN_NAME
------------------------------ --------------------
FK_DEPTNO                      DEPTNO
PK_EMP                         EMPNO
```

저자가 제약조건에 대하여 강의를 하다보면 외래키와 FOREIGN KEY 제약조건을 혼동하는 수강생이 다소 많은 편이다. 즉, 외래키와 FOREIGN KEY 제약조건을 동일한 것으로 생각하고 외래키를 만들기 위해서는 반드시 FOREIGN KEY 제약조건을 지정해야 하는 것으로 알고 있는데, FOREIGN KEY 제약조건은 외래키가 자신의 역할을 충실히 수행 할 수 있도록 지켜주는 역할을 하는 것으로 외래키가 사용자에 의해서 적절히 관리 될 수 있다면 반드시 FOREIGN KEY 제약조건을 지정 할 필요는 없다. 테이블에 제약조건이 많으면 많을수록 행의 입력, 변경, 삭제시 모든 제약조건을 확인하게 되므로 데이터베이스의 성능은 상대적으로 저하되게 된다.

복습

다음 ERD를 보고 테이블을 작성하시오. 단, 제약조건은 PRIMARY KEY, FOREIGN KEY만 사용하시오.