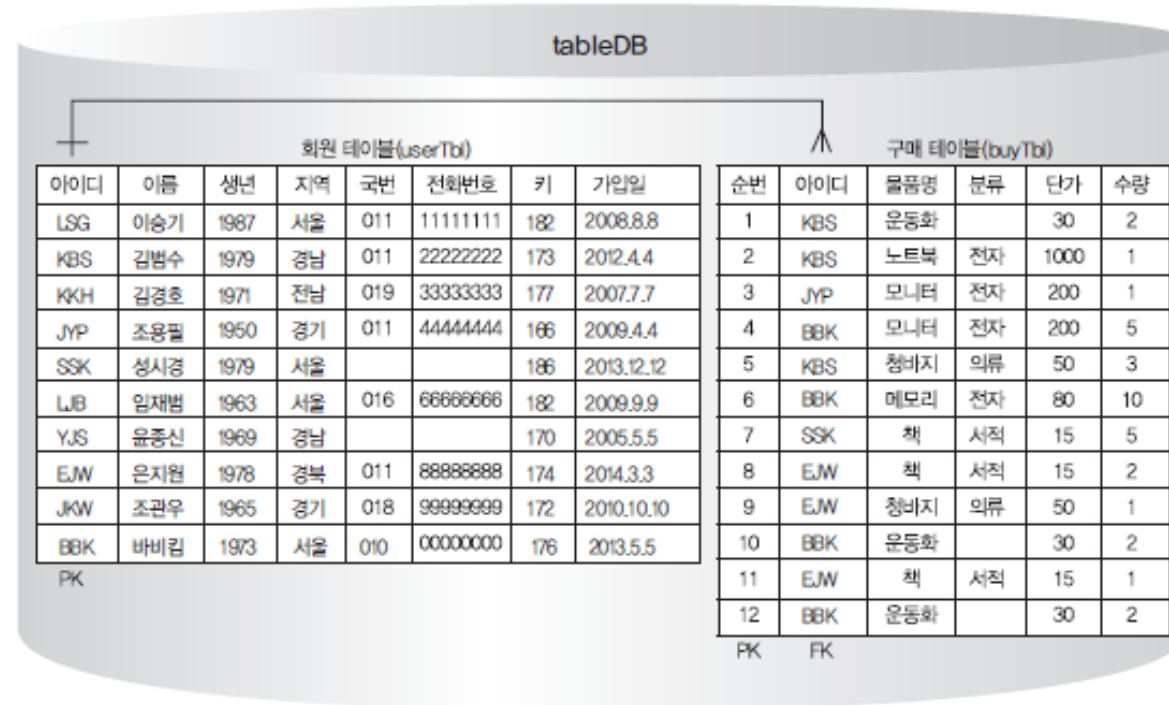


# SECTION 01 테이블

## 1.1 테이블 만들기

- SQL Developer에서 테이블 생성

- 테이블은 만드는 방법이 중요한 것이 아니라 테이블을 어떻게 모델링(설계)했느냐가 훨씬 중요함.



[그림 8-1] 샘플로 사용할 tableDB

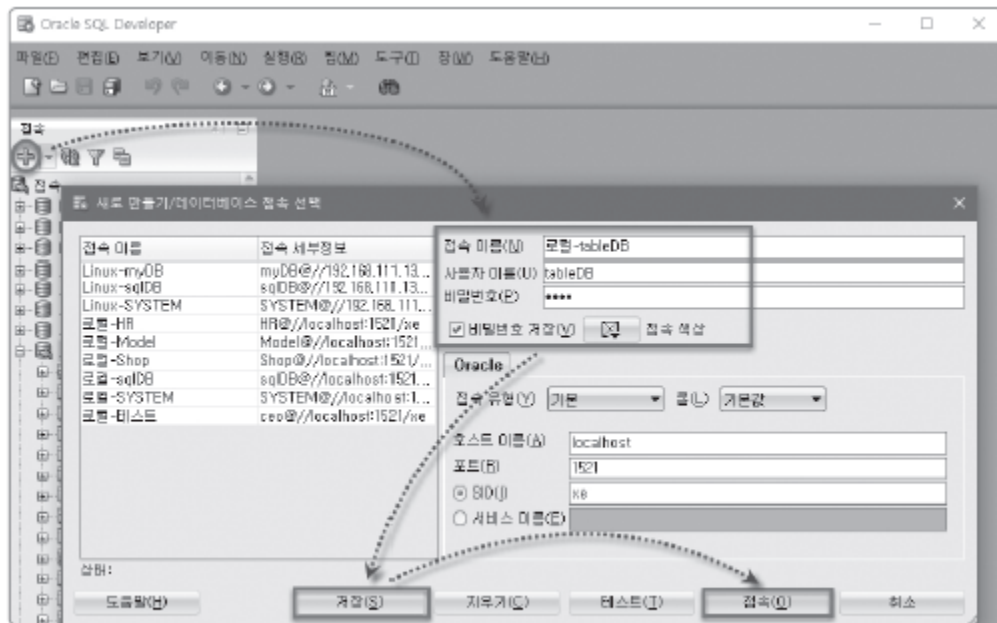
# SECTION 01 테이블

## 실습

SQL Developer에서 테이블을 생성하고, 데이터를 입력하자.

### step 0

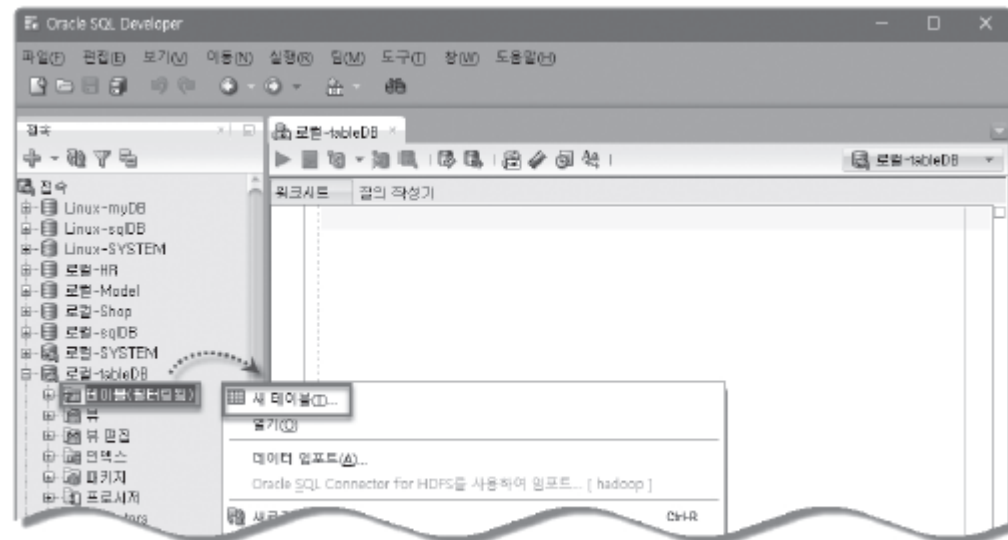
먼저 tableDB 스키마(=사용자)를 생성하고 [로컬-tableDB] 접속도 생성해 보자.



[그림 8-2] [로컬-tableDB] 접속의 생성

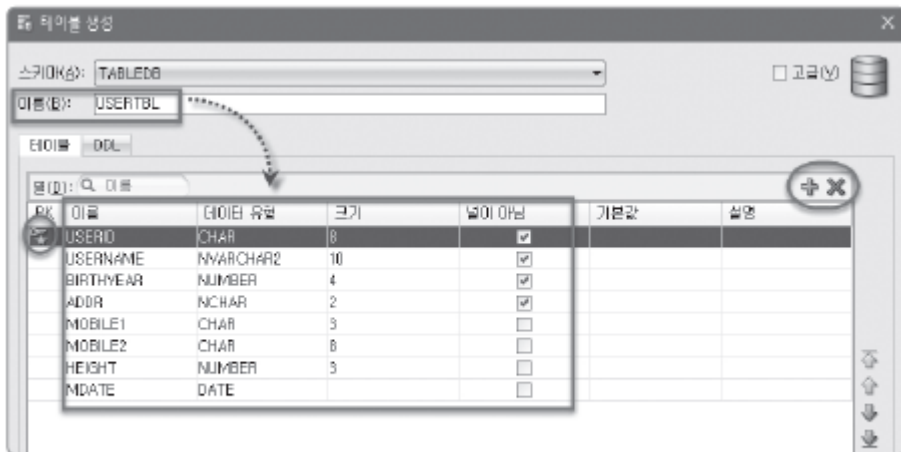
### step 1

SQL Developer의 그래픽 환경에서 테이블을 생성해 보자.

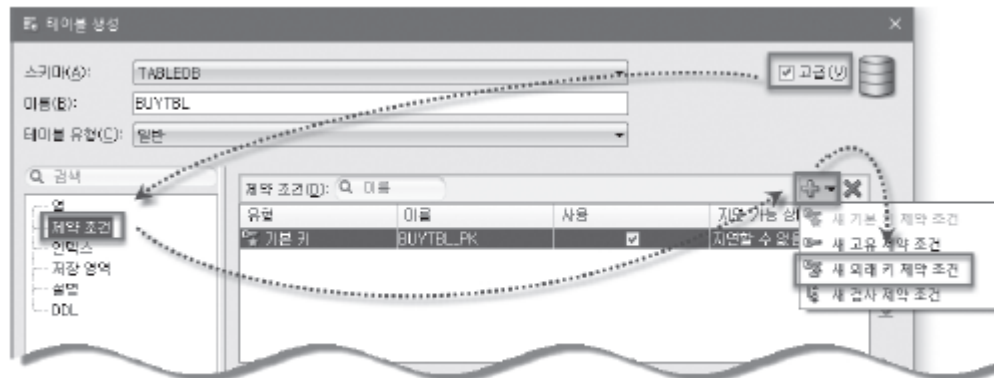


[그림 8-3] 회원 테이블(userTBL) 생성 1

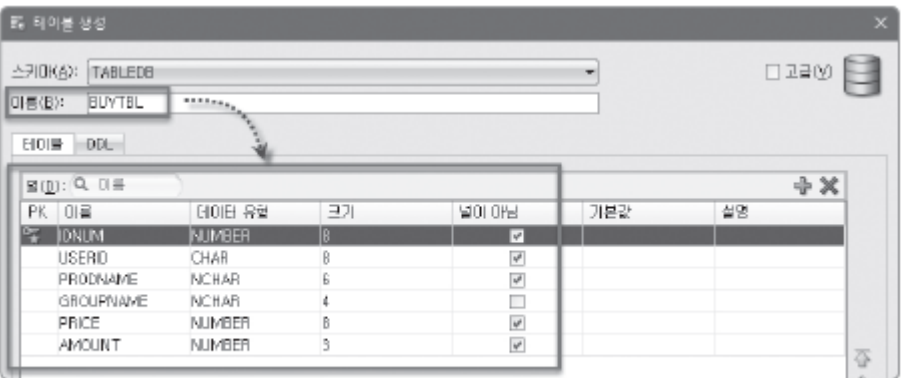
# SECTION 01 테이블



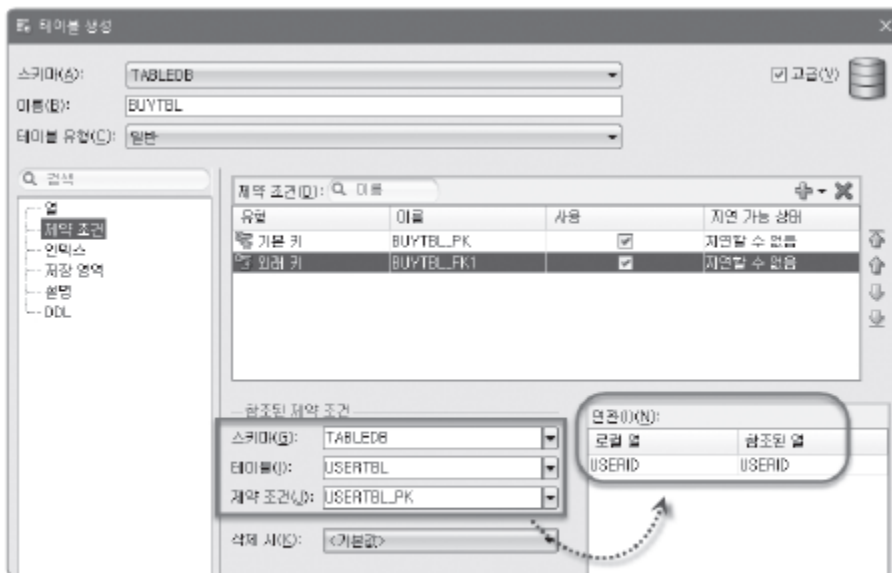
[그림 8-4] 회원 테이블(userTBL) 생성 2



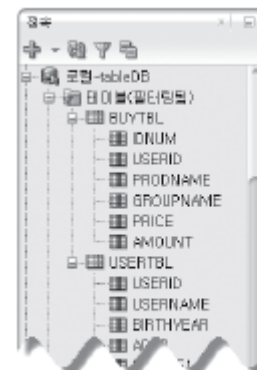
[그림 8-6] 구매 테이블(buyTBL) 생성 2



[그림 8-5] 구매 테이블(buyTBL) 생성 1



[그림 8-7] 구매 테이블(buyTBL) 생성 3

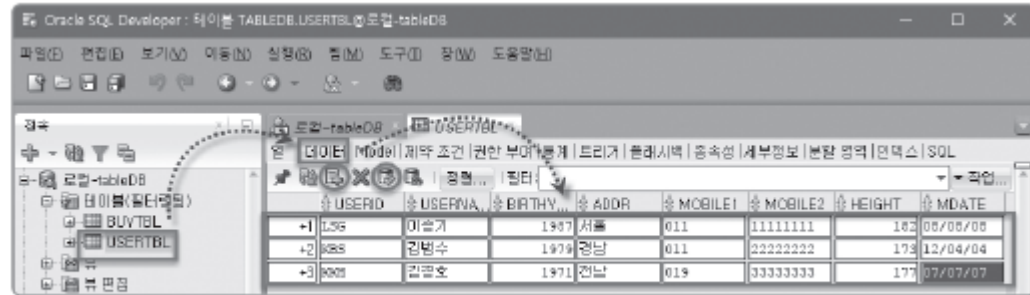


[그림 8-8] 테이블 확인

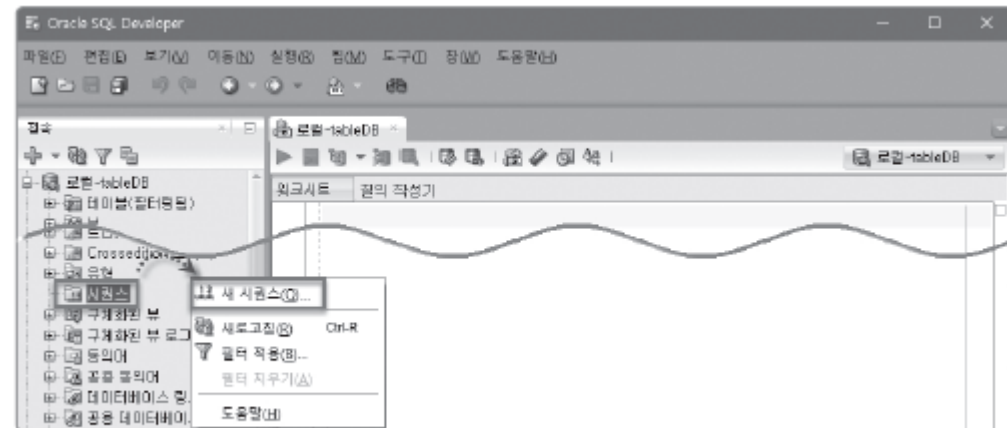
# SECTION 01 테이블

step2

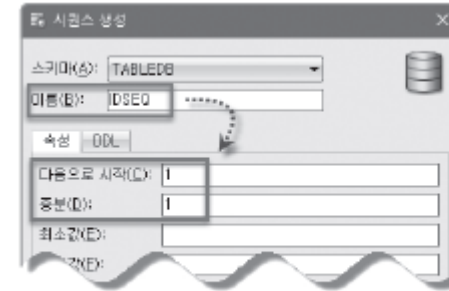
이번에는 SQL Developer에서 데이터를 입력하자.



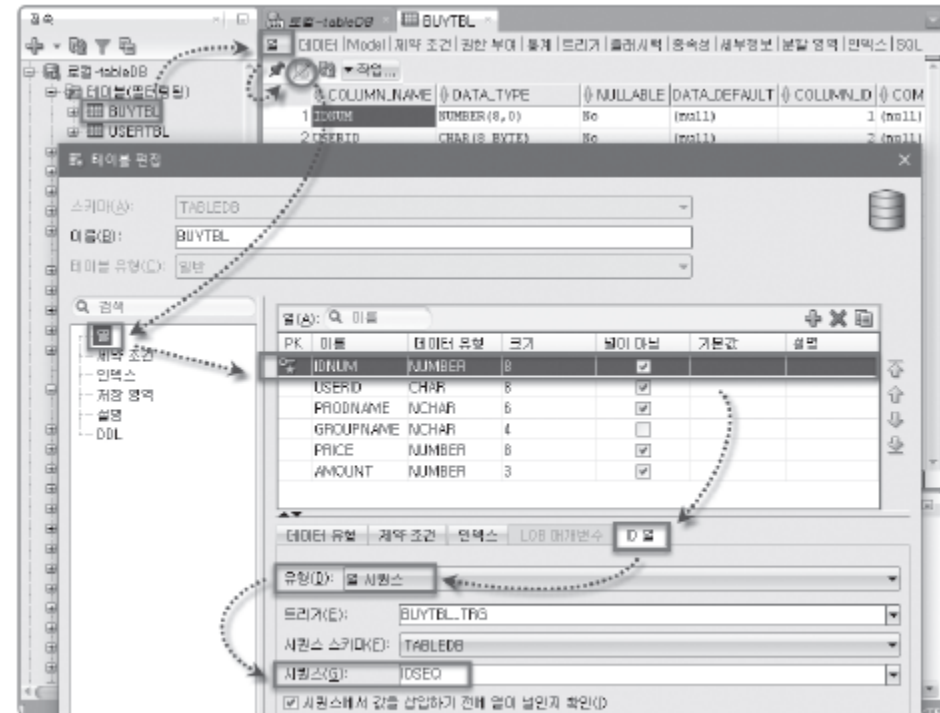
[그림 8-9] 회원 테이블의 일부 데이터 입력



[그림 8-10] 시퀀스 생성 1

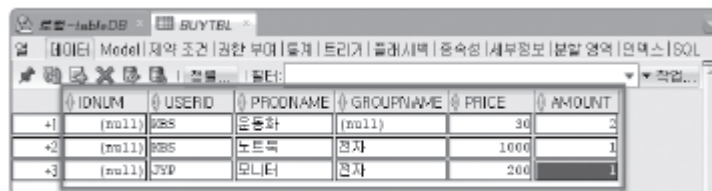


[그림 8-11] 시퀀스 생성 2



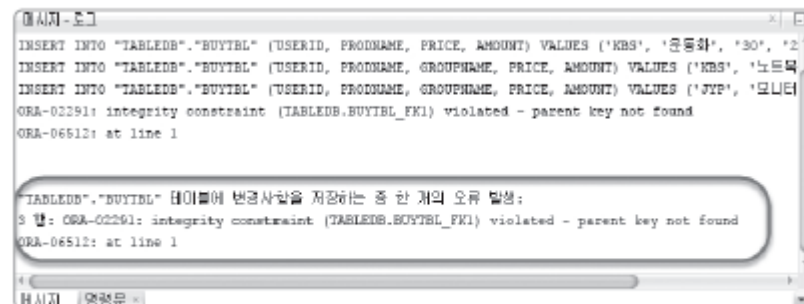
[그림 8-12] 구매 테이블의 IDNum 열에 IDSEQ 시퀀스 연결

# SECTION 01 테이블



	IDNUM	USERID	PRONAME	GROUPNAME	PRICE	AMOUNT
+1	(null)	BBS	운동화	(null)	30	2
+2	(null)	BBS	노트북	전자	1000	1
+3	(null)	JYP	모니터	전자	200	1

[그림 8-13] 구매 테이블의 일부 데이터 입력



```
INSERT INTO "TABLEDB"."BUYTEL" (USERID, PRONAME, PRICE, AMOUNT) VALUES ('BBS', '운동화', '30', '2')
INSERT INTO "TABLEDB"."BUYTEL" (USERID, PRONAME, GROUPNAME, PRICE, AMOUNT) VALUES ('BBS', '노트북', '전자', '1000', '1')
INSERT INTO "TABLEDB"."BUYTEL" (USERID, PRONAME, GROUPNAME, PRICE, AMOUNT) VALUES ('JYP', '모니터', '전자', '200', '1')
ORA-02291: integrity constraint (TABLEDB.BUYTEL_FK1) violated - parent key not found
ORA-06512: at line 1
```

"TABLEDB"."BUYTEL" 테이블에 변경사항을 저장하는 중 한 개의 오류 발생:  
3 행: ORA-02291: integrity constraint (TABLEDB.BUYTEL\_FK1) violated - parent key not found  
ORA-06512: at line 1

[그림 8-14] 오류 메시지



	IDNUM	USERID	PRON...	GROUP...	PRICE	AMOUNT
+1	(null)	BBS	운동화	(null)	30	2
+2	(null)	BBS	노트북	전자	1000	1
+3	(null)	JYP	모니터	전자	200	1

[그림 8-15] 행 삭제

# SECTION 01 테이블

## ◦ SQL로 테이블 생성

- Oracle 도움말에 나오는 테이블을 생성하는 기본적인 형식은 다음과 같음.

형식 :

```
CREATE [ GLOBAL TEMPORARY ] TABLE [ schema. ]table  
  [ (relational_properties) ]  
  [ ON COMMIT { DELETE | PRESERVE } ROWS ]  
  [ physical_properties ]  
  [ table_properties ] ;
```

(relational\_properties) :

```
{ column_definition  
  | { out_of_line_constraint  
    | out_of_line_ref_constraint  
    | supplemental_logging_props  
  }  
}  
[, { column_definition  
  | { out_of_line_constraint  
    | out_of_line_ref_constraint  
    | supplemental_logging_props  
  }  
}  
]...
```

physical\_properties :

```
{ segment_attributes_clause
```

```
  [ table_compression ]  
  | ORGANIZATION  
    { HEAP  
      [ segment_attributes_clause ]  
      [ table_compression ]  
    | INDEX  
      [ segment_attributes_clause ]  
      index_org_table_clause  
    | EXTERNAL  
      external_table_clause  
    }  
  | CLUSTER cluster (column [, column ]...)  
}
```

table\_properties :

```
[ column_properties ]  
[ table_partitioning_clauses ]  
[ CACHE | NOCACHE ]  
[ parallel_clause ]  
[ ROWDEPENDENCIES | NOROWDEPENDENCIES ]  
[ enable_disable_clause ]  
  [ enable_disable_clause ]...  
[ row_movement_clause ]  
[ AS subquery ]
```

# SECTION 01 테이블

## 실습2

SQL을 이용해서 테이블을 생성하자.

### step 0

열린 워크시트를 모두 닫는다. [로컬-tableDB]의 워크시트를 하나 열고 <실습 1>에서 사용한 테이블, 시퀀스, 트리거를 삭제하자.

```
DROP TABLE buyTBL;
DROP TABLE userTBL;
DROP SEQUENCE idSEQ;
DROP TRIGGER buyTBL_trg;
```

### step 1

[그림 8-1]을 보면서 하나씩 생성하자. 우선은 기본 키, 외래 키, NULL 값 등을 고려하지 말고 테이블의 기본적인 틀만 구성하자. 열 이름은 [그림 8-4]와 [그림 8-5]를 참조하자.

```
CREATE TABLE userTBL -- 회원 테이블
( userID      CHAR(8), -- 사용자 아이디(PK)
  userName    NVARCHAR2(10), -- 이름
  birthYear   NUMBER(4), -- 출생년도
  addr        NCHAR(2), -- 지역(경기, 서울, 경남 식으로 2글자만 입력)
  mobile1     CHAR(3), -- 휴대폰의 국번(010, 011, 016, 017, 018, 019 등)
  mobile2     CHAR(8), -- 휴대폰의 나머지 전화번호(하이픈 제외)
  height      NUMBER(3), -- 키
  mDate       DATE -- 회원 가입일
);

CREATE TABLE buyTBL -- 회원 구매 테이블
( idNum       NUMBER(8), -- 순번(PK)
  userID      CHAR(8), -- 아이디(FK)
  prodName    NCHAR(6), -- 물품명
  groupName   NCHAR(4), -- 분류
  price       NUMBER(8), -- 단가
  amount      NUMBER(3) -- 수량
);
```

>> 이것이 오라클이다

### step 2

추가적인 옵션을 줘서 테이블을 다시 생성하자.

```
DROP TABLE buyTBL;
DROP TABLE userTBL;
CREATE TABLE userTBL
( userID      CHAR(8) NOT NULL,
  userName    NVARCHAR2(10) NOT NULL,

  birthYear   NUMBER(4) NOT NULL,
  addr        NCHAR(2) NOT NULL,
  mobile1     CHAR(3) NULL,
  mobile2     CHAR(8) NULL,
  height      NUMBER(3) NULL,
  mDate       DATE NULL
);

CREATE TABLE buyTBL
( idNum       NUMBER(8) NOT NULL,
  userID      CHAR(8) NOT NULL,
  prodName    NCHAR(6) NOT NULL,
  groupName   NCHAR(4) NULL,
  price       NUMBER(8) NULL,
  amount      NUMBER(3) NOT NULL
);
```



# SECTION 01 테이블

## [‘PRIMARY KEY’ 문]

```
DROP TABLE buyTBL;
DROP TABLE userTBL;
CREATE TABLE userTBL
( userID CHAR(8) NOT NULL PRIMARY KEY,
  -- 중간 생략 --
);
CREATE TABLE buyTBL
( idNum NUMBER(8) NOT NULL PRIMARY KEY,
  -- 중간 생략 --
);
```

## [외래 키로 설정]

```
DROP TABLE buyTBL;
CREATE TABLE buyTBL
( idNum NUMBER(8) NOT NULL PRIMARY KEY,
  userID CHAR(8) NOT NULL,
  -- 중간 생략 --
  , FOREIGN KEY(userID) REFERENCES userTBL(userID)
);
```

step 3

이제는 데이터를 몇 건씩 입력하자.

먼저 회원 테이블에 3건만 입력하자.

```
INSERT INTO userTBL VALUES('LSG', '이승기', 1987, '서울', '011', '11111111', 182, '2008-8-8');
INSERT INTO userTBL VALUES('KBS', '김범수', 1979, '경남', '011', '2222222', 173, '2012-4-4');
INSERT INTO userTBL VALUES('KKH', '김경호', 1971, '전남', '019', '3333333', 177, '2007-7-7');
```

구매 테이블에 3건을 입력하자.

```
INSERT INTO buyTBL VALUES(idSEQ.NEXTVAL, 'KBS', '운동화', NULL, 30, 2);
INSERT INTO buyTBL VALUES(idSEQ.NEXTVAL, 'KBS', '노트북', '전자', 1000, 1);
INSERT INTO buyTBL VALUES(idSEQ.NEXTVAL, 'JYP', '모니터', '전자', 200, 1);
```

오류 메시지:

명령의 9 행에서 시작하는 중 오류 발생 -

```
INSERT INTO buyTBL VALUES(idSEQ.NEXTVAL, 'JYP', '모니터', '전자', 200, 1)
```

오류 보고 -

ORA-02291: integrity constraint (TABLEDB.SYS\_C007512) violated - parent key not found



# SECTION 01 테이블

## 1.2 제약 조건

- 제약 조건이란 데이터의 무결성을 지키기 위한 제한된 조건을 의미함.
- 기본 키 제약 조건
  - 테이블에 존재하는 많은 행의 데이터를 구분할 수 있는 식별자를 '기본 키'라고 부름.

```
CREATE TABLE userTBL
( userID      CHAR(8) NOT NULL PRIMARY KEY ,
  userName    NVARCHAR2(10) NOT NULL ,
  --- 중간 생략 ---
```

[기본키 생성]

```
SELECT * FROM USER_CONSTRAINTS -- 키 정보가 등록된 테이블
WHERE OWNER='TABLEDB' AND
      TABLE_NAME='USERTBL' AND
      CONSTRAINT_TYPE='P'; -- P는 기본 키, R은 외래 키, C는 NOT NULL 또는 CHECK
```

OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME	SEARCH_CONDITION	ROW_NUMBER
TABLEDB	SYS_C007523	P	USERTBL	(null)	(null)

[그림 8-16] 제약 조건 확인 1

```
DROP TABLE userTBL CASCADE CONSTRAINTS; -- 외래 키 제약 조건이 있어도 삭제
CREATE TABLE userTBL
( userID      CHAR(8) NOT NULL CONSTRAINT PK_userTBL_userID PRIMARY KEY ,
  userName    NVARCHAR2(10) NOT NULL ,
  -- 중간 생략 --
);
```

[PRIMARY KEY를 지정, 이름짓기]

OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME	SEARCH_CONDITION	ROW_NUMBER
TABLEDB	PK_USERTBL_USERID	P	USERTBL	(null)	(null)

[그림 8-17] 제약 조건 확인 2

# SECTION 01 테이블

```
DROP TABLE userTBL CASCADE CONSTRAINTS;  
CREATE TABLE userTBL  
( userID      CHAR(8) NOT NULL ,  
  --- 중간 생략 ---  
  mDate       DATE  NULL  
  , CONSTRAINT PK_userTBL_userID PRIMARY KEY (userID)  
);
```

[제약 조건의 이름을 지정]

```
DROP TABLE userTBL CASCADE CONSTRAINTS;  
CREATE TABLE userTBL  
( userID      CHAR(8) NOT NULL ,  
  --- 중간 생략 ---  
  mDate       DATE  NULL  
);  
ALTER TABLE userTBL  
  ADD CONSTRAINT PK_userTBL_userID  
  PRIMARY KEY (userID);
```

[ALTER TABLE 구문 사용]

- ALTER TABLE userTBL

: userTBL을 변경하자.

- ADD CONSTRAINT PK\_userTBL\_userID

: 제약 조건을 추가하자. 추가할 제약 조건 이름은 'PK\_userTBL\_userID' 이다.

- PRIMARY KEY(userID)

: 추가할 제약 조건은 기본 키 제약 조건이다. 그리고, 제약 조건을 설정할 열은 userID 열이다.

제품 코드	제품 일련 번호	제조일자	현 상태
AAA	0001	2019.10.10	판매완료
AAA	0002	2019.10.11	매장진열
BBB	0001	2019.10.12	재고창고
CCC	0001	2019.10.13	판매완료
CCC	0002	2019.10.14	매장진열

[표 8-1] 제품 테이블 샘플

# SECTION 01 테이블

```
CREATE TABLE prodTbl
( prodCode CHAR(3) NOT NULL,
  prodID   CHAR(4) NOT NULL,
  prodDate DATE  NOT NULL,
  prodCur CHAR(10) NULL
);
ALTER TABLE prodTbl
  ADD CONSTRAINT PK_prodTbl_proCode_prodID
    PRIMARY KEY (prodCode, prodID) ;
```

[‘제품 코드 + 제품일련번호 합침]

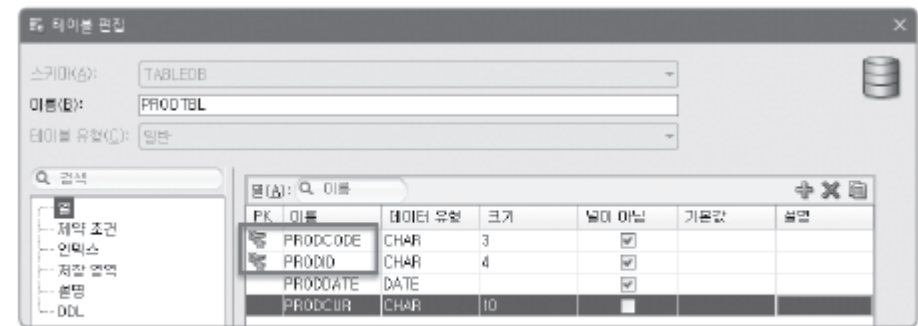
```
DROP TABLE prodTbl;
CREATE TABLE prodTbl
( prodCode CHAR(3) NOT NULL,
  prodID   CHAR(4) NOT NULL,
  prodDate DATE NOT NULL,
  prodCur  CHAR(10) NULL
, CONSTRAINT PK_prodTbl_proCode_prodID PRIMARY KEY (prodCode, prodID)
);
```

[CREATE TABLE 구문 안에 직접 사용]



INDEX_OWNER	INDEX_NAME	TABLE_OWNER	TABLE_NAME	COLUMN_NAME	COLUMN_POSITION
TABLEDB	PK_PROD_TBL_PROD_CODE_PROD_ID	TABLEDB	PROD_TBL	PROD_CODE	1
TABLEDB	PK_PROD_TBL_PROD_CODE_PROD_ID	TABLEDB	PROD_TBL	PROD_ID	2

[그림 8-18] 두 열을 하나의 기본 키로 설정한 상태 확인 1



PK	이름	데이터 유형	크기	널이 아님	기본값	설명
	PROD_CODE	CHAR	3	<input checked="" type="checkbox"/>		
	PROD_ID	CHAR	4	<input checked="" type="checkbox"/>		
	PROD_DATE	DATE		<input checked="" type="checkbox"/>		
	PROD_CUR	CHAR	10	<input type="checkbox"/>		

[그림 8-19] 두 열을 하나의 기본 키로 설정한 상태 확인 2

# SECTION 01 테이블

## ◦ 외래 키 제약 조건

- 외래 키 제약 조건은 두 테이블 사이의 관계를 선언함으로써, 데이터의 무결성을 보장해 주는 역할을 함.

```
DROP TABLE buyTBL;
DROP TABLE userTBL;
CREATE TABLE userTBL
( userID CHAR(8) NOT NULL PRIMARY KEY ,
  --- 중간 생략 ---
);
CREATE TABLE buyTBL
( idNum NUMBER(8) NOT NULL PRIMARY KEY,
  userID CHAR(8) NOT NULL REFERENCES userTBL(userID),
  prodName NCHAR(6) NOT NULL,
  --- 중간 생략 ---
  amount NUMBER(3) NOT NULL
);
```

[외래키 생성]

```
DROP TABLE buyTBL;
CREATE TABLE buyTBL
( idNum NUMBER(8) NOT NULL PRIMARY KEY,
  userID CHAR(8) NOT NULL
  CONSTRAINT FK_userTBL_buyTBL REFERENCES userTBL(userID),
  --- 중간 생략 ---
  amount NUMBER(3) NOT NULL
);
```

[직접 외래 키의 이름을 지정]

```
DROP TABLE buyTBL;
CREATE TABLE buyTBL
( idNum NUMBER(8) NOT NULL PRIMARY KEY,
  userID CHAR(8) NOT NULL,
  --- 중간 생략 ---
  amount NUMBER(3) NOT NULL
  , CONSTRAINT FK_userTBL_buyTBL FOREIGN KEY(userID) REFERENCES userTBL(userID)
);
```

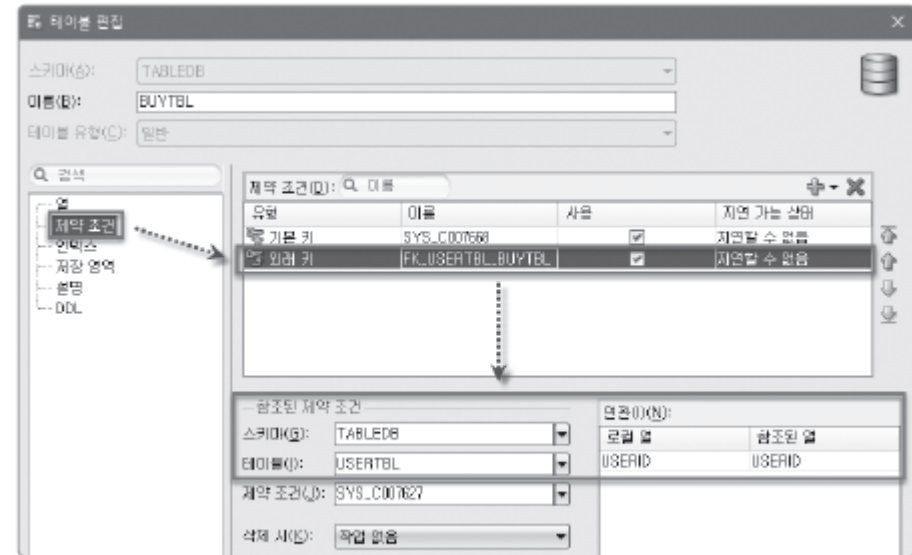
[외래 키의 이름을 지정할 필요가 없을 경우]

# SECTION 01 테이블

```
DROP TABLE buyTBL;
CREATE TABLE buyTBL
( idNum      NUMBER(8) NOT NULL PRIMARY KEY,
  userID     CHAR(8) NOT NULL,
  --- 중간 생략 ---
  amount     NUMBER(3) NOT NULL
);
ALTER TABLE buyTBL
  ADD CONSTRAINT FK_userTBL_buyTBL
  FOREIGN KEY (userID)
  REFERENCES userTBL(userID) ;
```

[ALTER TABLE 구문 사용]

- **ALTER TABLE buyTBL**  
: buyTBL을 수정한다.
- **ADD CONSTRAINT FK\_userTBL\_buyTBL**  
: 제약 조건을 더한다. 제약 조건 이름은 'FK\_userTBL\_buyTBL'로 명명한다.
- **FOREIGN KEY (userID)**  
: 외래 키 제약 조건을 buyTBL의 userID에 설정한다.
- **REFERENCES userTBL (userID)**  
: 참조할 기준 테이블은 userTBL 테이블의 userID 열이다.



[그림 8-20] 외래 키 제약 조건 확인

```
ALTER TABLE buyTBL
  DROP CONSTRAINT FK_userTBL_buyTBL; -- 외래 키 제거
ALTER TABLE buyTBL
  ADD CONSTRAINT FK_userTBL_buyTBL
  FOREIGN KEY (userID)
  REFERENCES userTBL (userID)
  ON DELETE CASCADE ;
```

[외래 키의 옵션 중에 ON DELETE CASCADE 옵션]

# SECTION 01 테이블

- UNIQUE 제약 조건

- UNIQUE 제약 조건은 '중복되지 않는 유일한 값'을 입력해야 하는 조건임.

```
DROP TABLE userTbl CASCADE CONSTRAINTS;
CREATE TABLE userTBL
(  userID    CHAR(8) NOT NULL ,
   --- 중간 생략 ---
   mDate     DATE  NULL,
   email     CHAR(30) NULL UNIQUE
);

DROP TABLE userTbl CASCADE CONSTRAINTS;
CREATE TABLE userTBL
(  userID    CHAR(8) NOT NULL ,
   --- 중간 생략 ---
   mDate     DATE  NULL,
   email     CHAR(30) NULL
, CONSTRAINT AK_email UNIQUE (email)
);

DROP TABLE userTbl CASCADE CONSTRAINTS;
CREATE TABLE userTBL
(  userID    CHAR(8) NOT NULL ,
   --- 중간 생략 ---
   mDate     DATE  NULL,
   email     CHAR(30) NULL
);
ALTER TABLE USERTBL
  ADD CONSTRAINT AK_EMAIL UNIQUE (EMAIL);
```

[Email 주소를 Unique로 설정]

# SECTION 01 테이블

- CHECK 제약 조건

- CHECK 제약 조건은 입력되는 데이터를 점검하는 기능을 함.

```
-- 키는 0 이상이어야 함.  
ALTER TABLE userTbl  
  ADD CONSTRAINT CK_height  
    CHECK (height >= 0) ;
```

```
-- 휴대폰 국번 체크  
ALTER TABLE userTbl  
  ADD CONSTRAINT CK_mobile1  
    CHECK (mobile1 IN ('010','011','016','017','018','019')) ;
```

[예시 '마이너스 값이 들어올 수 없다' 등의 조건을 지정]

```
-- 휴대폰 국번 체크 (기존 무시)  
ALTER TABLE userTbl  
  ADD CONSTRAINT CK_mobile1_new  
    CHECK (mobile1 IN ('010','011','016','017','018','019'))  
  ENABLE NOVALIDATE ;
```



# SECTION 01 테이블

## ◦ DEFAULT 정의

- DEFAULT는 값을 입력하지 않았을 때, 자동으로 입력되는 기본 값을 정의하는 방법임.

```
DROP TABLE userTbl CASCADE CONSTRAINTS;
CREATE TABLE userTBL
( userID      CHAR(8) NOT NULL PRIMARY KEY ,
  userName    NVARCHAR2(10) NOT NULL ,
  birthYear   NUMBER(4) DEFAULT -1 NOT NULL ,
  addr        NCHAR(2) DEFAULT '서울' NOT NULL ,
  mobile1     CHAR(3) NULL,
  mobile2     CHAR(8) NULL,
  height      NUMBER(3) DEFAULT 170 NULL,
  mDate       DATE NULL
);
```

[예시]

```
-- default문은 DEFAULT로 설정된 값을 자동 입력한다.
INSERT INTO userTBL VALUES ('LHL', '이혜리', DEFAULT, DEFAULT, '011', '1234567',
DEFAULT, '2019.12.12');
-- 열 이름이 명시되지 않으면 DEFAULT로 설정된 값을 자동 입력한다.
INSERT INTO userTBL(userID, userName) VALUES('KAY', '김아영');
-- 값이 직접 명기되면 DEFAULT로 설정된 값은 무시된다.
INSERT INTO userTBL VALUES ('WB', '원빈', 1982, '대전', '019', '9876543', 176,
'2020.5.5');
SELECT * FROM userTBL;
```

ID	USERID	USERNAME	BIRTHYEAR	ADDR	MOBILE1	MOBILE2	HEIGHT	MDATE
1	LHL	이혜리	-1	서울	011	1234567	170	19/12/12
2	KAY	김아영	-1	서울	(null)	(null)	170	(null)
3	WB	원빈	1982	대전	019	9876543	176	20/05/05

[그림 8-21] DEFAULT 확인

>> 이것이 오라클이다

```
DROP TABLE userTbl CASCADE CONSTRAINTS;
CREATE TABLE userTBL
( userID      CHAR(8) NOT NULL PRIMARY KEY ,
  userName    NVARCHAR2(10) NOT NULL ,
  birthYear   NUMBER(4) NOT NULL ,
  addr        NCHAR(2) NOT NULL ,
  mobile1     CHAR(3) NULL,
  mobile2     CHAR(8) NULL,
  height      NUMBER(3) NULL,
  mDate       DATE NULL
);
```

```
ALTER TABLE userTBL
  MODIFY birthYear DEFAULT -1;
ALTER TABLE userTBL
  MODIFY addr DEFAULT '서울';
ALTER TABLE userTBL
  MODIFY height DEFAULT 170;
```

[MODIFY문을 사용]

[디폴트가 설정된 열]

# SECTION 01 테이블

- NULL 값 허용

- NULL 값을 허용하려면 NULL을, 허용하지 않으려면 NOT NULL을 사용하면 됨.

```
INSERT INTO userTBL(userID, userName, mobile1) VALUES('MGG', '마징가', NULL);  
INSERT INTO userTBL(userID, userName, mobile1) VALUES('MKD', '메칸더', ' ');  
INSERT INTO userTBL(userID, userName, mobile1) VALUES('JJK', '짱가', ' ');
```

# SECTION 01 테이블

## 1.3 임시 테이블

- 임시 테이블은 이름처럼 임시로 잠깐 사용되는 테이블임.
  - 임시 테이블의 데이터는 세션 내에서만 존재하며, 세션이 닫히면 자동으로 데이터가 삭제됨.

형식 :

CREATE GLOBAL TEMPORARY TABLE 테이블이름

( 열 정의 ... )

[ON COMMIT DELETE ROWS 또는 ON COMMIT PRESERVE ROWS]

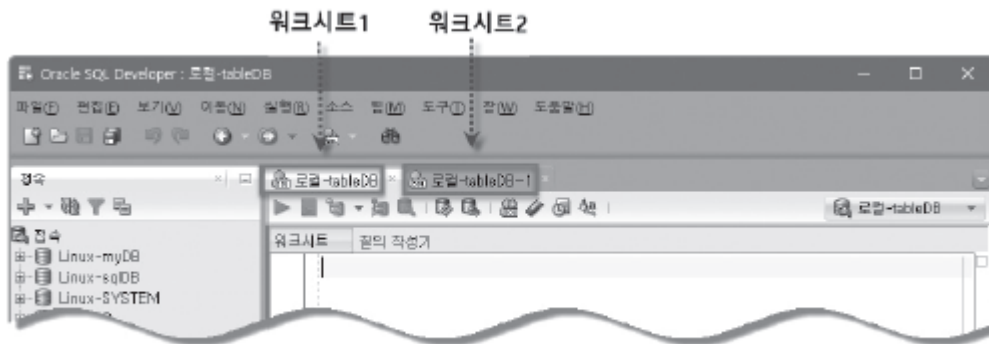
# SECTION 01 테이블

## 실습3

임시 테이블을 사용하자.

### step 0

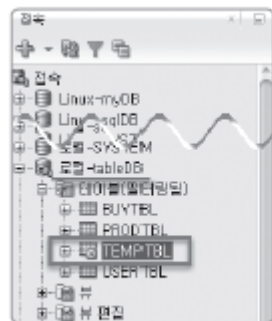
워크시트를 2개 준비하자. SQL Developer를 종료하고 다시 실행하자.



[그림 8-22] 2개의 워크시트

### step 1

(워크시트 1) 임시 테이블을 사용해 보자.



[그림 8-23] 임시 테이블 확인 1

[SELECT TABLE\_NAME,  
TEMPORARY FROM USER\_TABLES; 문으로 확인]

	TABLE_NAME	TEMPORARY
1	PRODTBL	N
2	BUYTBL	N
3	USERTBL	N
4	TEMPTBL	Y

[그림 8-24] 임시 테이블 확인 2

```
INSERT INTO temptBL VALUES('Thomas', '토마스');  
INSERT INTO temptBL VALUES('James', '제임스');  
SELECT * FROM temptBL;
```

ID	UNAME
1 Thomas	토마스
2 James	제임스

[그림 8-25] 쿼리 실행 결과

```
COMMIT;  
SELECT * FROM temptBL;
```

ID	UNAME
----	-------

[그림 8-26] 쿼리 실행 결과

# SECTION 01 테이블

step 2

(워크시트 2) 옵션을 변경해서 임시 테이블을 생성해 보자.

```
CREATE GLOBAL TEMPORARY TABLE tempTBL2 (id CHAR(8), uName NCHAR(10))  
ON COMMIT PRESERVE ROWS;  
INSERT INTO tempTBL2 VALUES('Arthur', '아서');  
INSERT INTO tempTBL2 VALUES('Murdoch', '머독');
```

[ON COMMIT PRESERVE ROWS문으로 생성]

```
COMMIT;  
SELECT * FROM tempTBL2;
```

ID	UName
1 Arthur	아서
2 Murdoch	머독

[그림 8-27] 쿼리 실행 결과

step 3

(워크시트 1) 워크시트 1에서 생성한 테이블에 접근해 보자.

```
SELECT * FROM tempTBL2;
```

step 4

연결을 종료하고 임시 테이블을 확인해 보자.

```
SELECT * FROM tempTBL2;
```

[[로컬-tableDB]에서 생성했던 임시 테이블을 확인]

step 5

임시 테이블을 삭제하자. 일반 테이블과 동일하게 삭제된다.

```
DROP TABLE tempTBL;  
DROP TABLE tempTBL2;
```

# SECTION 01 테이블

## 1.4 테이블 삭제

- 외래 키 제약 조건의 기준 테이블은 원칙적으로 삭제할 수가 없음.
- 먼저, 외래 키가 생성된 외래 키 테이블을 삭제해야 함.

```
DROP TABLE 테이블이름 [CASCADE CONSTRAINTS];
```

# SECTION 01 테이블

## 1.5 테이블 수정

- 테이블의 수정은 ALTER TABLE문을 사용함.

```
ALTER TABLE [ schema. ] table
[ alter_table_properties
| column_clauses
| constraint_clauses
| alter_table_partitioning
| alter_external_table_clauses
| move_table_clause
]
[ enable_disable_clause
| { ENABLE | DISABLE } { TABLE LOCK | ALL TRIGGERS }
] ...
;
```



# SECTION 01 테이블

## ◦ 열의 추가

```
ALTER TABLE userTBL  
  ADD homepage VARCHAR(30) -- 열 추가  
  DEFAULT 'http://www.hanbit.co.kr' -- 디폴트값  
  NULL; -- Null 허용함
```

[회원 테이블(userTBL)에 회원의 홈페이지 주소를 추가]

```
ALTER TABLE userTBL  
  ADD (homeAddr NVARCHAR2(20), postNum VARCHAR(5) );
```

[여러 개의 열을 추가]

## ◦ 열의 삭제

```
ALTER TABLE userTBL  
  DROP COLUMN homeAddr;
```

[열을 삭제]

```
ALTER TABLE userTBL  
  DROP (homepage, postNum);
```

[여러 개의 열을 삭제]

# SECTION 01 테이블

- 열의 이름 변경

```
ALTER TABLE userTBL  
  RENAME COLUMN userName TO uName;
```

[uName으로 변경]

- 열의 데이터 형식 변경

```
ALTER TABLE userTBL  
  MODIFY (addr NVARCHAR2(10) NULL);
```

[NVARCHAR2(10)으로 변경]

- 열의 제약 조건 추가 및 삭제

```
ALTER TABLE userTBL  
  DROP PRIMARY KEY;
```

[기본 키를 삭제 - 오류]

```
ALTER TABLE buyTBL  
  DROP CONSTRAINT SYS_C007801; -- 외래 키 이름을 다를 수 있음
```

[외래 키를 제거한 후에 다시 기본 키를 제거]

# SECTION 01 테이블

## 실습4

지금까지 익힌 테이블의 제약 조건 및 수정 방법을 실습을 통해서 익히자.

### step 0

모든 워크시트를 닫고, [로컬-tableDB]에서 워크시트를 하나 연다.

### step 1

모든 제약 조건을 제외하고 [그림 8-1]의 테이블을 다시 만들자. 단, 구매 테이블(buyTBL)의 idNum 열만 PRIMARY KEY 속성을 준다.

```
DROP TABLE buyTbl;
DROP TABLE userTbl;
CREATE TABLE userTBL
( userID      CHAR(8),
  userName    NVARCHAR2(10),
  birthYear   NUMBER(4),
  addr        NCHAR(2),
  mobile1     CHAR(3),
  mobile2     CHAR(8),
  height      NUMBER(3),
  mDate       DATE
);
CREATE TABLE buyTBL
( idNum       NUMBER(8) PRIMARY KEY,
  userID      CHAR(8),
  prodName    NCHAR(6),
  groupName   NCHAR(4),
  price       NUMBER(8),
  amount      NUMBER(3)
);
DROP SEQUENCE idSEQ;
CREATE SEQUENCE idSEQ;
```

>> 이것이 오라클이다

### step 2

먼저 각각의 테이블에 데이터를 테이블당 4건씩만 입력하자. 입력 시에 김범수의 출생년도는 모르는 것으로 NULL 값을 넣고, 김경호의 출생년도는 1871년으로 잘못 입력해 보자.

```
INSERT INTO userTBL VALUES('LSG', '이승기', 1987, '서울', '011', '1111111', 182, '2008-8-8');
INSERT INTO userTBL VALUES('KBS', '김범수', NULL, '경남', '011', '2222222', 173, '2012-4-4');
INSERT INTO userTBL VALUES('KKH', '김경호', 1871, '전남', '019', '3333333', 177, '2007-7-7');
INSERT INTO userTBL VALUES('JYP', '조용필', 1950, '경기', '011', '4444444', 166, '2009-4-4');
INSERT INTO buyTBL VALUES(idSEQ.NEXTVAL, 'KBS', '운동화', NULL, 30, 2);
INSERT INTO buyTBL VALUES(idSEQ.NEXTVAL, 'KBS', '노트북', '전자', 1000, 1);
INSERT INTO buyTBL VALUES(idSEQ.NEXTVAL, 'JYP', '모니터', '전자', 200, 1);
INSERT INTO buyTBL VALUES(idSEQ.NEXTVAL, 'BBK', '모니터', '전자', 200, 5);
```

### step 3

제약 조건을 생성하자.

[기본 키 제약 조건을 생성]

```
ALTER TABLE userTBL
ADD CONSTRAINT PK_userTBL_userID
PRIMARY KEY (userID);
```

[기본 키를 확인]

```
SELECT * FROM USER_CONSTRAINTS
WHERE OWNER='TABLEDB' AND TABLE_NAME='USERTBL' AND CONSTRAINT_TYPE='P';
DESCRIBE userTBL;
```

OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME	SEARCH_CONDITION
TABLEDB	PK_USERTBL_USERID	P	USERTBL	Null

[그림 8-28] 쿼리 실행 결과

# SECTION 01 테이블

step 4

외래 키를 설정해 보자.

[userID 열에 외래 키를 설정]

```
ALTER TABLE buyTBL
  ADD CONSTRAINT FK_userTBL_buyTBL
  FOREIGN KEY (userID)
  REFERENCES userTBL (userID);
```

오류 메시지:

```
ORA-02298: cannot validate (TABLEDB.FK_USERTBL_BUYTBL) - parent keys not found
02298. 00000 - "cannot validate (%s,%s) - parent keys not found"
```

\*Cause: an alter table validating constraint failed because the table has child records.

\*Action: Obvious

[BBK행을 삭제하고, 다시 외래 키를 설정]

```
DELETE FROM buyTBL WHERE userID = 'BBK';
ALTER TABLE buyTBL
  ADD CONSTRAINT FK_userTBL_buyTBL
  FOREIGN KEY (userID)
  REFERENCES userTBL (userID);
```

[buyTBL의 네 번째 데이터를 다시 입력]

```
INSERT INTO buyTBL VALUES(idSEQ.NEXTVAL,'BBK', '모니터', '전자', 200, 5);
```

오류 메시지:

```
ORA-02291: integrity constraint (TABLEDB.FK_USERTBL_BUYTBL) violated - parent key
not found
```

[외래 키 제약 조건을 활성화]

```
ALTER TABLE buyTBL
  DISABLE CONSTRAINT FK_userTBL_buyTBL;
INSERT INTO buyTBL VALUES(idSEQ.NEXTVAL, 'BBK', '모니터', '전자', 200, 5);
INSERT INTO buyTBL VALUES(idSEQ.NEXTVAL, 'KBS', '청바지', '의류', 50, 3);
INSERT INTO buyTBL VALUES(idSEQ.NEXTVAL, 'BBK', '메모리', '전자', 80, 10);
INSERT INTO buyTBL VALUES(idSEQ.NEXTVAL, 'SSK', '책', '서적', 15, 5);
INSERT INTO buyTBL VALUES(idSEQ.NEXTVAL, 'EJW', '책', '서적', 15, 2);
INSERT INTO buyTBL VALUES(idSEQ.NEXTVAL, 'EJW', '청바지', '의류', 50, 1);
INSERT INTO buyTBL VALUES(idSEQ.NEXTVAL, 'BBK', '운동화', NULL, 30, 2);
INSERT INTO buyTBL VALUES(idSEQ.NEXTVAL, 'EJW', '책', '서적', 15, 1);
INSERT INTO buyTBL VALUES(idSEQ.NEXTVAL, 'BBK', '운동화', NULL, 30, 2);
ALTER TABLE buyTBL
  ENABLE NOVALIDATE CONSTRAINT FK_userTBL_buyTBL;
```

step 5

이번에는 userTBL의 출생년도를 1900 ~ 2017까지만 설정하도록 CHECK 제약 조건을 설정하자.

```
ALTER TABLE userTBL
  ADD CONSTRAINT CK_birthYear
  CHECK (birthYear >= 1900 AND birthYear <= 2017)
  ENABLE;
```

[CHECK 제약 조건을 설정]

[CHECK 제약 조건을 설정]

```
ALTER TABLE userTBL
  ADD CONSTRAINT CK_birthYear
  CHECK (birthYear >= 1900 AND birthYear <= 2017)
  ENABLE NOVALIDATE ;
```

# SECTION 01 테이블

step 6

나머지 userTBL의 데이터도 입력하자.

```
INSERT INTO userTBL VALUES('SSK', '성시경', 1979, '서울', NULL, NULL, 186, '2013-12-12');
INSERT INTO userTBL VALUES('LJB', '임재범', 1963, '서울', '016', '6666666', 182, '2009-9-9');
INSERT INTO userTBL VALUES('YJS', '윤종신', 1969, '경남', NULL, NULL, 170, '2005-5-5');
INSERT INTO userTBL VALUES('EJW', '은지원', 1972, '경북', '011', '8888888', 174, '2014-3-3');
INSERT INTO userTBL VALUES('JKW', '조관우', 1965, '경기', '018', '9999999', 172, '2010-10-10');
INSERT INTO userTBL VALUES('BBK', '바비킴', 1973, '서울', '010', '0000000', 176, '2013-5-5');
```

step 7

이제부터는 정상적으로 운영하면 된다.

step 8

이번에는 바비킴(BBK)이 회원을 탈퇴하면 (= 회원 테이블에서 삭제되면) 구매한 기록도 삭제되는지 확인하자.

```
DELETE FROM userTBL WHERE userID = 'BBK';
```

오류 메시지:

```
ORA-02292: integrity constraint (TABLEDB.FK_USER_TBL_BUY_TBL) violated - child record found
```

[바비킴(BBK) 회원을 삭제]

```
ALTER TABLE buyTBL
  DROP CONSTRAINT FK_userTBL_buyTBL;
ALTER TABLE buyTBL
  ADD CONSTRAINT FK_userTBL_buyTBL
    FOREIGN KEY (userID)
    REFERENCES userTBL (userID)
    ON DELETE CASCADE;
```

[ON DELETE CASCADE문을 함께 설정]

```
DELETE FROM userTBL WHERE userID = 'BBK';
SELECT * FROM buyTBL;
```

IDNUM	USERID	PRODNAME	GROUPNAME	PRICE	AMOUNT
1	1 KBS	음향화	(null)	30	2
2	2 KBS	노트북	전자	1000	1
3	3 JYP	모니터	전자	200	1
4	7 KBS	정바지	의류	50	3
5	9 SSK	책	서적	15	5
6	10 EJW	책	서적	15	2
7	11 EJW	정바지	의류	50	1
8	13 EJW	책	서적	15	1

[그림 8-29] 쿼리 실행 결과

[buyTBL에도 따라서 삭제되었는지 확인]

step 9

이번에는 userTBL에서 CHECK 제약 조건이 걸린 출생년도 birthYear 열을 삭제해 보자.

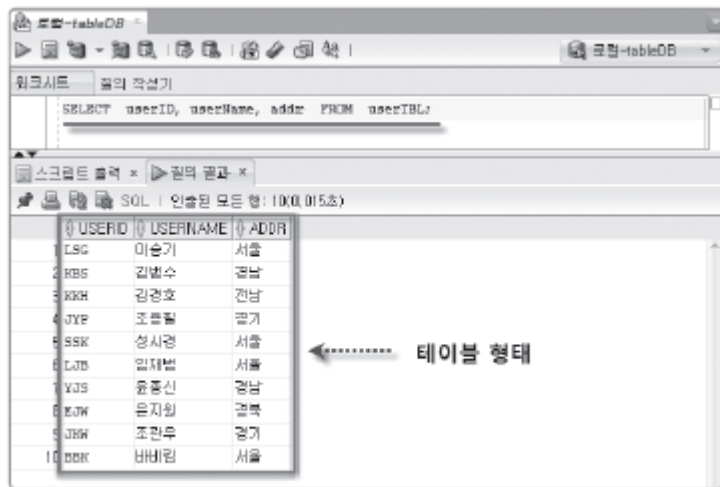
ALTER TABLE로 삭제하자.

```
ALTER TABLE userTBL
  DROP COLUMN birthYear ;
```

# SECTION 02 뷰

## 2.1 뷰의 개념

- 뷰는 일반 사용자 입장에서는 테이블과 동일하게 사용하는 개체임.



The screenshot shows a database client window with a SQL query: `SELECT userID, userName, addr FROM userTBL;` The result is displayed in a table with columns `USERID`, `USERNAME`, and `ADDR`. An arrow points to the table with the text "테이블 형태" (Table format).

USERID	USERNAME	ADDR
1.SG	이승기	서울
2.BBS	김범수	경남
3.BBS	김경호	전남
4.JYP	조용필	경기
5.SSK	성시경	서울
6.LJB	임재범	서울
7.YJS	윤종신	경남
8.EJW	은지원	경북
9.JHW	조관우	경기
10.BBK	버버릴	서울

[그림 8-30] 테이블의 쿼리와 그 결과

[워크시트에서 SELECT문을 수행]



[그림 8-32] 뷰의 작동 방식

```
CREATE VIEW v_userTBL
AS
SELECT userID, userName, addr FROM userTBL;
```

[뷰를 생성하는 구문]

```
SELECT * FROM v_userTBL; -- 뷰를 테이블이라고 생각해도 무방
```



USERID	USERNAME	ADDR
1.SG	이승기	서울
2.BBS	김범수	경남
3.BBS	김경호	전남
4.JYP	조용필	경기
5.SSK	성시경	서울
6.LJB	임재범	서울
7.YJS	윤종신	경남
8.EJW	은지원	경북
9.JHW	조관우	경기
10.BBK	버버릴	서울

[그림 8-31] 뷰의 쿼리 결과

# SECTION 02 뷰

## 2.2 뷰의 장점

- 뷰를 사용해서 얻을 수 있는 장점
  - 보안에 도움이 됨.
  - 복잡한 쿼리를 단순화시켜줄 수 있음.

실습5

뷰를 생성해서 활용하자.

step 0

7장 <실습 1>의 step 0 을 참조해서 sqlDB를 초기화한다. 독자가 스스로 한다.

step 1

[로컬-sqlDB]에서 워크시트를 열고 기본적인 뷰를 생성한다. 뷰의 생성 시에 뷰에서 사용될 열의 이름을 변경할 수도 있다.

```
CREATE OR REPLACE VIEW v_userbuyTBL
AS
SELECT U.userID AS "USER ID", U.userName AS "USER NAME", B.prodName AS "PRODUCT NAME",
       U.addr, CONCAT(U.mobile1, U.mobile2) AS "MOBILE PHONE"
FROM userTBL U
INNER JOIN buyTBL B
ON U.userID = B.userID;

SELECT "USER ID", "USER NAME" FROM v_userbuyTBL;
```

	USER ID	USER NAME
1	1000	김범수
2	1000	김범수
3	1000	조용필
4	1000	김민준
10	1000	박대립
11	1000	윤지원
12	1000	박대립

[그림 8-34] 쿼리 실행 결과



# SECTION 02 뷰

step2

뷰의 수정은 생성 시와 마찬가지로 CREATE OR REPLACE VIEW 구문을 사용하면 된다. 한글의 열 이름도 가능하다. (호환성 문제로 별로 권장하지는 않는다.)

```
CREATE OR REPLACE VIEW v_userbuyTBL
AS
SELECT U.userID AS "사용자 아이디", U.userName AS "이름", B.prodName AS "제품 이름",
       U.addr, CONCAT(U.mobile1, U.mobile2) AS "전화 번호"
FROM userTBL U
     INNER JOIN buyTBL B
       ON U.userID = B.userID ;

SELECT "이름", "전화 번호" FROM v_userbuyTBL;
```

step3

뷰의 삭제는 DROP VIEW를 사용하면 된다.

```
DROP VIEW v_userbuyTBL;
```

step4

뷰에 대한 정보는 시스템 뷰인 USER\_에, 디렉터리 뷰인 USER\_VIEWS에 들어 있다.

```
CREATE OR REPLACE VIEW v_userTBL
AS
SELECT userID, userName, addr FROM userTBL;
```

[뷰 생성]

```
SELECT * FROM USER_VIEWS;
```

[뷰 정보확인]

VIEW_NAME	TEXT_LENGTH	TEXT	TYPE_TEXT_LENGTH
V_USERTBL	42	SELECT userID, userName, addr FROM userTBL	(null)

[그림 8-35] 뷰 소스 확인

>> 이것이 오라클이다

step5

뷰를 통해서 데이터를 변경해 보자.

[데이터 수정]

```
UPDATE v_userTBL SET addr = '부산' WHERE userID='JKW';
```

[데이터 입력]

```
INSERT INTO v_userTBL(userID, userName, addr) VALUES('KBM','김병만','충북');
```

오류 메시지:

```
ORA-01400: cannot insert NULL into ("SQLDB"."USERTBL"."BIRTHYEAR")
```

```
CREATE OR REPLACE VIEW v_userTBL
AS
```

[WITH READ ONLY문 사용]

```
SELECT userID, userName, addr FROM userTBL
WITH READ ONLY;
```

[읽기 전용 뷰라는 오류발생]

```
UPDATE v_userTBL SET addr = '태국' WHERE userID='SSK';
```

오류 메시지:

```
ORA-42399: cannot perform a DML operation on a read-only view
```

# SECTION 02 부

step 6

이번에는 그룹 함수를 포함하는 뷰를 정의해 보자.

6-1 SUM() 함수를 사용하는 뷰를 간단히 정의해 보자. 당연히 결과는 잘 나왔다.

```
CREATE OR REPLACE VIEW v_sum
AS
  SELECT userID , SUM(price*amount) AS "Total"
  FROM buyTBL GROUP BY userID;

SELECT * FROM v_sum;
```

USERID	Total
1 BBR	1820
2 SKK	75
3 KBS	1210
4 ESW	95
5 JYP	200

[그림 8-36] 그룹 함수를 뷰에 포함시킴

step 7

지정한 범위로 뷰를 생성하고 데이터를 입력하자.

[뷰 생성]

```
CREATE OR REPLACE VIEW v_height177
AS
  SELECT * FROM userTBL WHERE height >= 177 ;

SELECT * FROM v_height177 ;
```

USERID	USERNAME	BIRTHYEAR	ADDR	MOBILE1	MOBILE2	HEIGHT	MDATE
1 LSG	이승기	1987	서울	011	11111111	182	08/08/08
2 YKH	김경호	1971	전남	019	33333333	177	07/07/07
3 SKK	성시경	1979	서울	(null)	(null)	186	13/13/13
4 LJB	임재범	1963	서울	016	66666666	183	09/09/09

[그림 8-37] 범위를 지정한 뷰

[데이터 삭제]

```
DELETE FROM v_height177 WHERE height < 177 ;
```

결과 메시지:

0개 행 이(가) 삭제되었습니다.

[데이터 입력]

```
INSERT INTO v_height177 VALUES('KBM', '김병만', 1977, '경기', '010', '5555555', 158,
                                '2019-01-01') ;
```

결과 메시지:

1 행 이(가) 삽입되었습니다.

## SECTION 02 뷰

step 8

두 개 이상의 테이블이 관련되는 복합 뷰를 생성하고 데이터를 입력하자.

```
CREATE OR REPLACE VIEW v_userbuyTBL
AS
SELECT U.userID, U.userName, B.prodName, U.addr, U.mobile1 || U.mobile2 AS mobile
FROM userTBL U
     INNER JOIN buyTBL B
       ON U.userID = B.userID ;
```

```
INSERT INTO v_userbuyTBL VALUES('PKL','박경리','운동화','경기','000000000000');
```

오류 메시지:

```
ORA-01779: cannot modify a column which maps to a non key-preserved table
01779. 00000 - "cannot modify a column which maps to a non key-preserved table"
*Cause:      An attempt was made to insert or update columns of a join view which
              map to a non-key-preserved table.
*Action:     Modify the underlying base tables directly.
```

step 9

뷰가 참조하는 테이블을 삭제해보자.

[데이터 삭제]

```
DROP TABLE userTbl CASCADE CONSTRAINTS;
```

```
SELECT * FROM v_userbuyTBL;
```

[뷰 조회]

오류 메시지:

```
ORA-04063: view "SQLDB.V_USERBUYTBL" has errors
```

```
04063. 00000 - "%s has errors"
```

```
*Cause:      Attempt to execute a stored procedure or use a view that has
              errors. For stored procedures, the problem could be syntax errors
              or references to other, non-existent procedures. For views,
              the problem could be a reference in the view's defining query to
              a non-existent table.
              Can also be a table which has references to non-existent or
              inaccessible types.
```

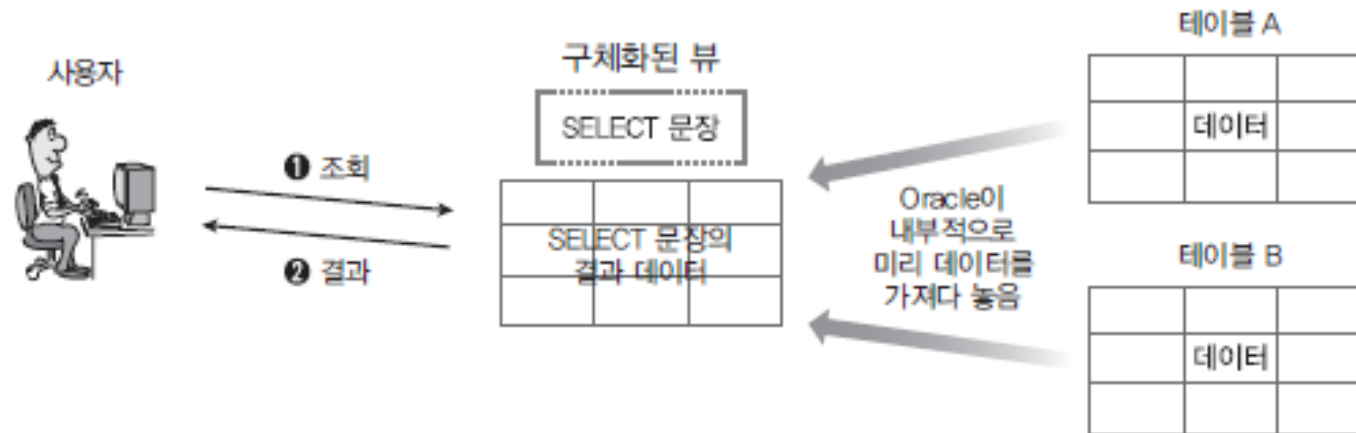
```
*Action:     Fix the errors and/or create referenced objects as necessary.
```

```
82행, 15열에서 오류 발생
```

## SECTION 02 뷰

### 2.3 구체화된 뷰

- 뷰의 실체는 SELECT문뿐이며 그 안에 데이터는 없음.
- 구체화된 뷰 : '실제 데이터가 존재하는 뷰'라고 말할 수 있음.



[그림 8-38] 구체화된 뷰의 작동 방식

## SECTION 02 뷰

- 뷰의 옵션 중 BUILD IMMEDIATE는 구체화된 뷰를 생성한 후 동시에 구체화된 내부에 데이터가 채워지고, BUILD DEFERRED는 나중에 채워짐.
- 관심있게 볼 옵션은 REFRESH ON COMMIT과 REFRESH ON DEMAND인데 ON COMMIT으로 설정되면 원본 테이블에 커밋이 발생할 때마다 구체화된 뷰의 내용이 변경되고, ON DEMAND는 직접 DBMS\_MVIEW 패키지를 실행해서 구체화된 뷰의 내용을 변경함.
- FAST | COMPLETE | FORCE | NEVER 옵션은 구체화된 뷰의 업데이트 방식을 지정함.
- FAST와 FORCE는 원본 테이블에 변경된 데이터만 구체화된 뷰에 적용시키고, COMPLETE는 원본 테이블이 변경되면 전체를 구체화된 뷰에 적용시킴. NEVER는 원본 테이블이 변경되어도 구체화된 뷰에는 적용시키지 않는다는 의미임.

### [구체화된 뷰의 형식과 주요 옵션]

```
형식 :  
CREATE MATERIALIZED VIEW 뷰이름  
[ BUILD { IMMEDIATE | DEFERRED }  
  REFRESH {ON COMMIT | ON DEMAND} { FAST | COMPLETE | FORCE | NEVER }  
  ENABLE QUERY REWRITE ]  
AS  
  SELECT 문장;
```

# SECTION 02 부

## 실습6

구체화된 뷰를 활용해 보자.

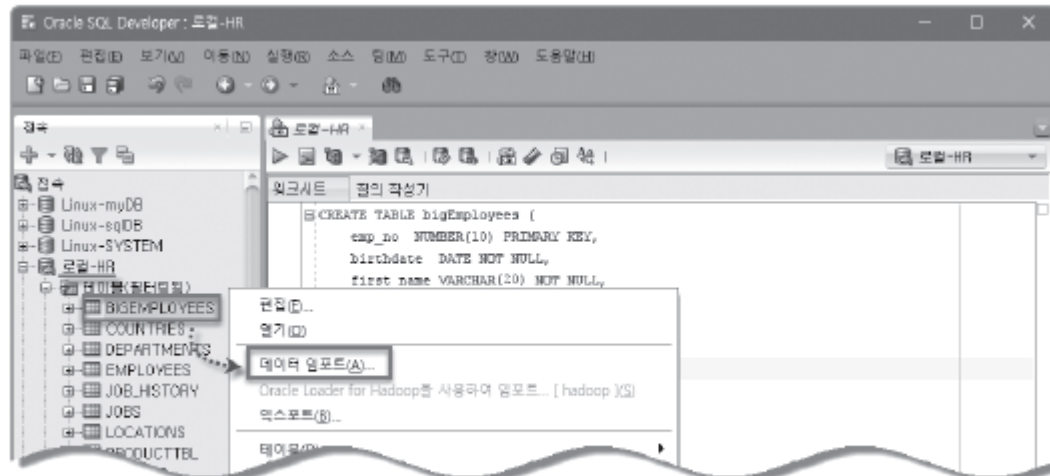
### step 0

실습을 위해서 먼저 대용량 테이블을 준비하자. 약 30만 건의 데이터가 있는 bigEmployees 테이블을 HR 스키마에 생성하자.

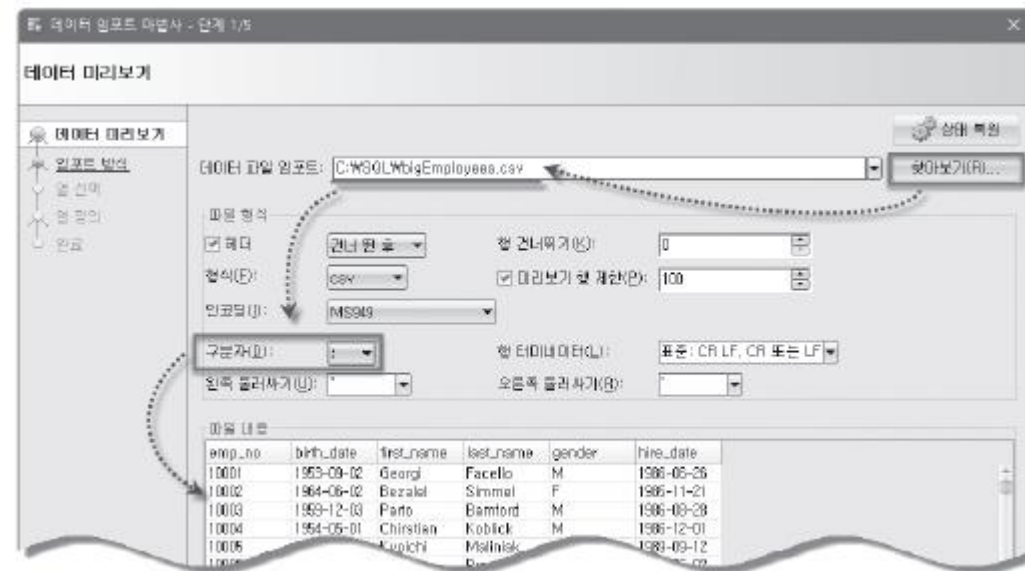
[책의 자료실(<http://cafe.naver.com/thisisOracle>)에서 bigEmployees.csv 파일을 다운로드해서 C:\SQL\ 폴더에 저장함]

[bigEmployees 테이블을 만들]

```
CREATE TABLE bigEmployees (  
    emp_no NUMBER(10) PRIMARY KEY,  
    birth_date DATE NOT NULL,  
    first_name VARCHAR(20) NOT NULL,  
    last_name VARCHAR(20) NOT NULL,  
    gender CHAR(1) NOT NULL,  
    hire_date DATE NOT NULL  
);
```

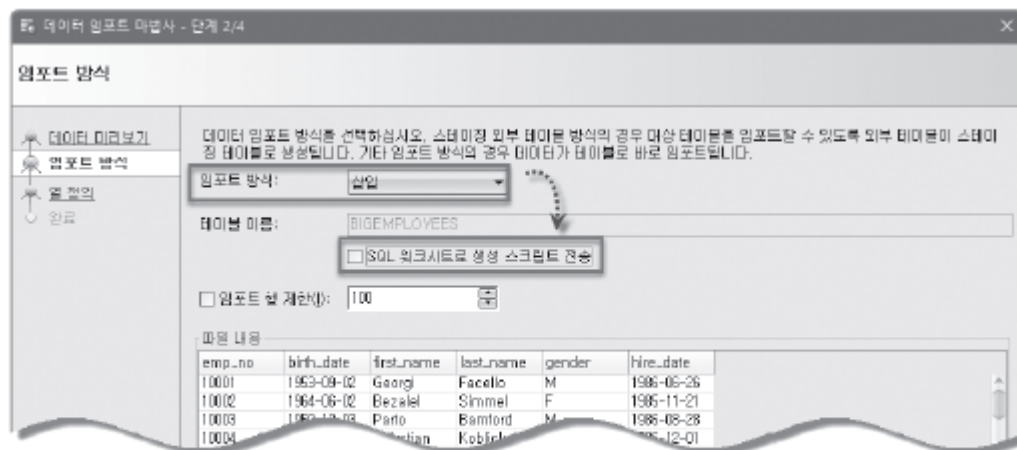


[그림 8-39] CSV 파일 임포트 1

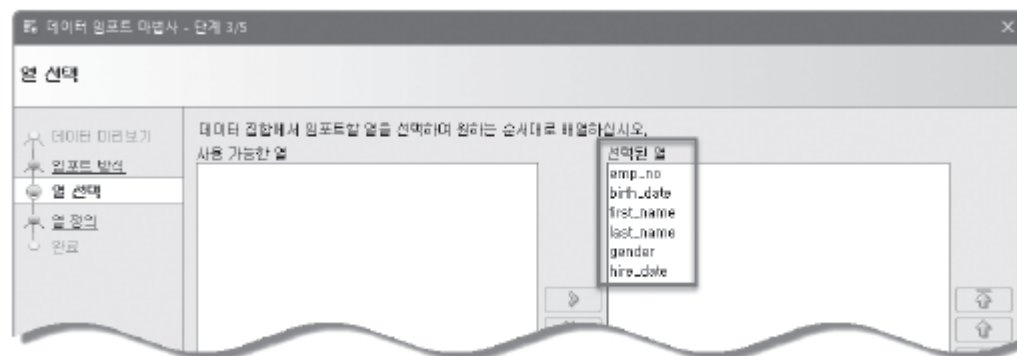


[그림 8-40] CSV 파일 임포트 2

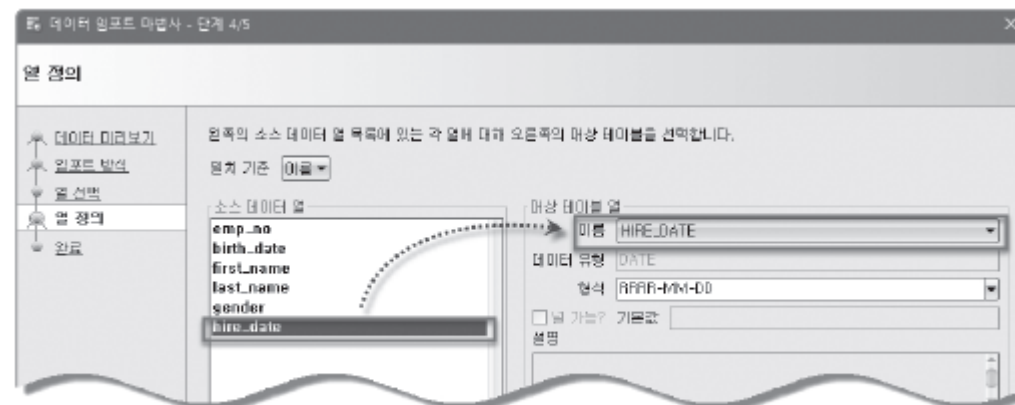
## SECTION 02 뷰



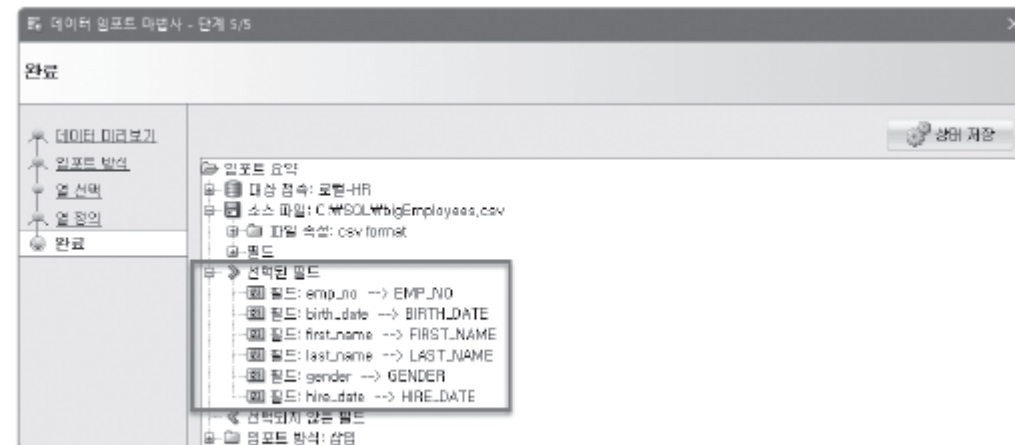
[그림 8-41] CSV 파일 임포트 3



[그림 8-42] CSV 파일 임포트 4



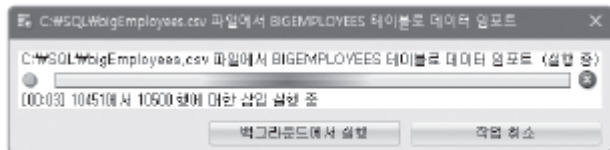
[그림 8-43] CSV 파일 импорт 5



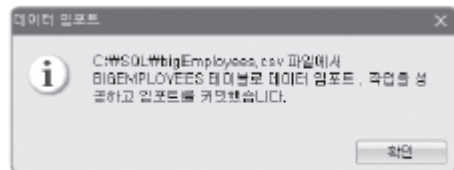
[그림 8-44] CSV 파일 임포트 6



# SECTION 02 부



[그림 8-45] CSV 파일 임포트 7



[그림 8-46] CSV 파일 임포트 8

```
SELECT COUNT(*) FROM bigEmployees; -- 약 30만 건이 나오면 됨
SELECT * FROM bigEmployees WHERE ROWNUM <= 10;
```

EMP_NO	BIRTH_DATE	FIRST_NAME	LAST_NAME	GENDER	HIRE_DATE
1	10666 64/10/03	Zhonghua	Recker	M	89/05/20
2	10667 55/08/05	Hakato	Quesworth	F	90/06/25
3	10668 59/03/20	Maren	Granlund	F	89/09/25
4	10669 56/05/30	Yuriy	Iizaka	F	95/06/09
5	10670 62/04/23	Shunichi	McKffer	M	88/03/23
6	10671 53/06/13	Parvis	Rebaine	M	87/01/08
7	10672 64/03/23	Bluma	Zeilberger	M	86/07/13
8	10673 59/12/26	Hyuckchul	Casperson	M	90/03/01
9	10674 61/03/03	Abdelasis	Bisiani	M	88/05/16
10	10675 60/03/26	Ynte	Waschkowski	F	91/11/05

[그림 8-47] 데이터 확인

step 1

대용량의 테이블을 준비하자.

[테이블을 복사]

```
CREATE TABLE bigTBL AS SELECT * FROM HR.bigEmployees;
CREATE TABLE smallTBL AS SELECT * FROM HR.Employees;
```

step 2

일반적인 집계 함수를 이용한 쿼리의 수행 시간을 확인하자.

```
SELECT ROUND(AVG(EXTRACT(YEAR FROM B.birth_date)), 0) AS "평균 출생년도"
FROM bigTBL B
CROSS JOIN smallTBL S; -- 일부러 시간이 걸리도록 추가한 구문
```

[평균을 내는 쿼리 수행]

평균 출생년도
1958

[그림 8-48] 일반 쿼리의 실행

[뷰를 생성]

```
CREATE MATERIALIZED VIEW mv_AvgYear
AS
SELECT ROUND(AVG(EXTRACT(YEAR FROM B.birth_date)), 0) AS "평균 출생년도"
FROM bigTBL B
CROSS JOIN smallTBL S;
```

```
SELECT * FROM mv_AvgYear;
```

[뷰조회]

평균 출생년도
1958

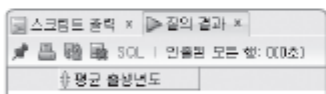
[그림 8-49] 구체화된 뷰의 실행 1

# SECTION 02 뷰

[옵션을 지정해서 다양한 설정]

```
DROP MATERIALIZED VIEW mv_AvgYear;
CREATE MATERIALIZED VIEW mv_AvgYear
  BUILD DEFERRED
AS
  SELECT ROUND(AVG(EXTRACT(YEAR FROM B.birth_date)), 0) AS "평균 출생년도"
  FROM bigTBL B
  CROSS JOIN smallTBL S;

SELECT * FROM mv_AvgYear;
```



평균 출생년도
2000

[그림 8-50] 구체화된 뷰의 실행 2

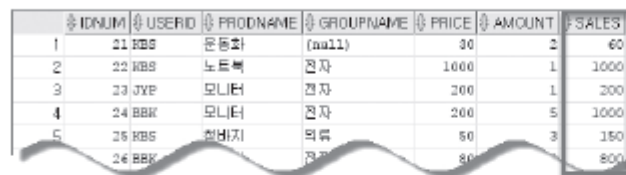
[REFRESH() 프로시저를 실행]

```
EXECUTE DBMS_MVIEW.REFRESH(LIST =>'mv_AvgYear');
SELECT * FROM mv_AvgYear;
```

step 3

이번에는 다른 예를 살펴보자. 원본 테이블의 데이터가 변경되었을 때, 구체화된 뷰의 값을 확인해 보자.

```
ALTER TABLE buyTBL
  ADD sales GENERATED ALWAYS AS (price * amount);
SELECT * FROM buyTBL;
```



IDNUM	USERID	PRODNAME	GROUPNAME	PRICE	AMOUNT	SALES
1	21 KBS	문동화	(null)	30	2	60
2	22 KBS	노트북	전자	1000	1	1000
3	23 JYP	모니터	전자	200	1	200
4	24 BBB	모니터	전자	200	5	1000
5	25 KBS	컴퓨터	의류	50	3	150
6	26 BBB	컴퓨터	의류	80	10	800

[그림 8-51] 가상 열이 추가된 테이블

[구체화된 뷰를 생성]

```
CREATE MATERIALIZED VIEW mv_SumSales
  BUILD IMMEDIATE
  REFRESH COMPLETE -- 전체 뷰가 변경됨
  ON COMMIT -- 원본 테이블이 COMMIT되는 즉시 변경됨
AS
  SELECT SUM(sales)
  FROM buyTBL;

SELECT * FROM mv_SumSales;
```



SUM(SALES)
3500

[구체화된 뷰를 확인]

```
UPDATE buyTBL SET price = price*2;
COMMIT;
SELECT * FROM mv_SumSales;
```



SUM(SALES)
7000