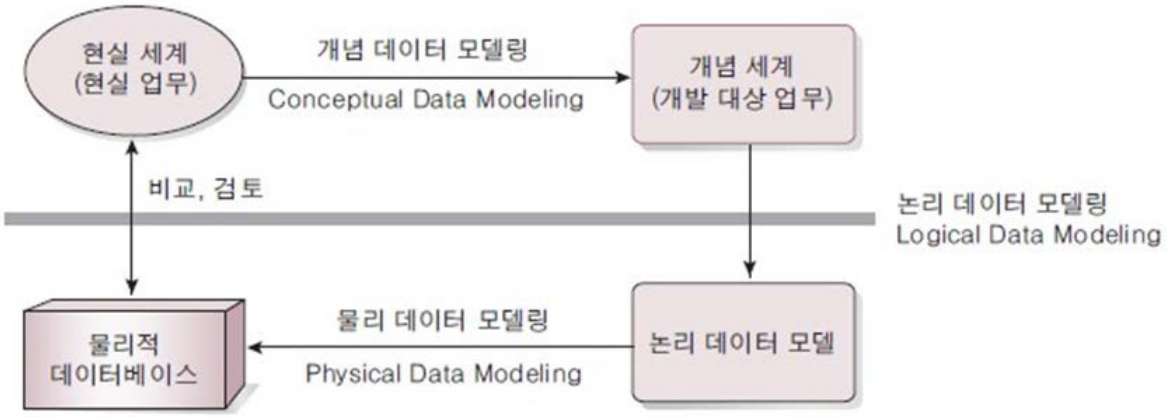
23장 데이터 모델링

23.1 데이터 모델링의 개요

23.1.1 데이터 모델링의 정의

■ 데이터 모델(Data Model)은 데이터베이스 내에 존재하는 데이터의 타입을 정의하고 데이터들 사이의 관계를 규정하며 데이터의 의미와 데이터에 가해진 제약 조건을 명시하기 위해 사용하는 개념적인 도구이다.



23.1.2 데이터 모델링의 수행절차

절차	주 요 내 용	산출물
요구사항분석	- 기업 비즈니스를 이해하고 구조화 하기 위한 정보를 정의 - 인터뷰 및 장표 분석을 통하여 요구사항을 도출, 정의, 명세, 검증 수행함	요구사항 분석서
개념적 모델링	- 실세계의 정보 구조의 모형을 변환하여 일반화 시키는 단계 - 핵심 엔터티와 그들 간의 관계를 표현하기 위해 ERD를 작성 - 사용자와 시스템 개발자가 데이터 요구 사항을 발견하는 것을 지원하며 의사소통의 기반을 마련함 - 현 시스템이 어떻게 변경되어야 하는가를 이해하는데 유용함	개념적 ERD
논리적 모델링	- 개념적 설계에서 추출된 실체와 속성들의 관계를 구조적으로 설계 하는 단계 - 정확한 업무 분석을 통한 자료의 흐름을 분석하여 실체와 속성들의 관계를 구조적으로 설계 - 논리적 데이터베이스 모델링 단계에서 완벽한 **정규화** 과정을 수행 - **식별자 확정, 정규화, M:M 관계 해소, 참조 무결성 규칙 정의** 등을 수행	논리 데이터 모델
물리적 모델링	- 정규화된 논리적 데이터모델을 개발 DBMS의 특성에 적합하도록 효율적인 데이터베이스 스키마를 구축하는 단계 - Data의 DISK상의 위치, 인덱스, 파티션테이블, 분산 설계 등을 수행 - 성능을 고려한 반정규화 실시	물리 데이터 모델

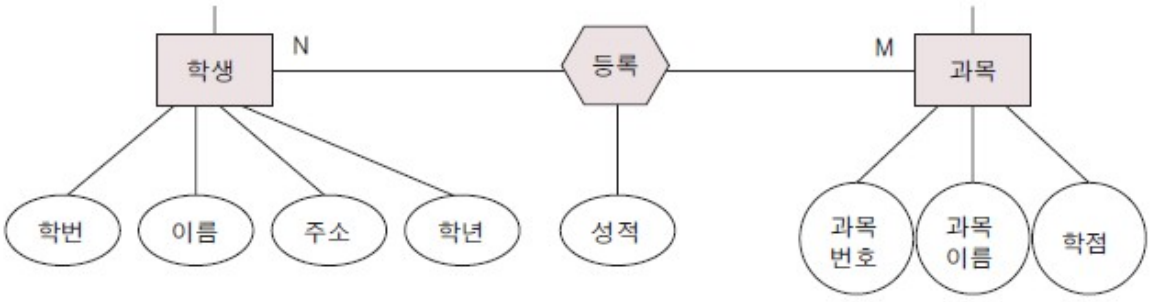
23.1.3 좋은 데이터 모델링이란?

- 완전성: 업무에 필요한 모든 데이터가 데이터 모델에 정의되어 있어야 한다.
- 중복배제: 하나의 데이터베이스에 동일한 사실은 한번만 기록하여 저장공간의 낭비나 중복관리 데이터의 일관성을 유지하기 위한 비효율적인 업무 지양한다.
- 업무규칙: 데이터 모델링 과정에서 도출되는 업무규칙들을 데이터 모델로 표현하고 공유하고 활용하여 효율성을 증대한다.
- 의사소통: 데이터 모델이 각각의 이해관계 사이의 의사소통의 역할을 담당한다. 즉, 데이터 모델은 업무를 데이터 관점에서 분석하고 이를 설계하여 나오는 최종 산출물이므로 효과적인 의사소통 수단이 될 수 있다.

23.2 개체-관계 모델(E-R : Entity-Relationship Model)

23.2.1 개체-관계 모델의 정의

- 1976년 피터 첸(Peter Chen)이 제안한 개체-관계 모델은 데이터베이스 구조의 논리적/물리적 구조를 표현 하기 위해 제안된 모델로 데이터 구조와 데이터 간의 관계를 엔티티 집합(Entity Set)과 관계 집합 (Relationship Set)을 이용해 도형화하여 개념적으로 표현하는 모델이다.
- 데이터베이스의 개념적 설계를 위해 가장 널리 사용되는 대표적인 개념적 데이터 모델로 개체-관계 다이어그램으로 표현된다.



23.2.2 개체-관계 모델의 구성요소

(1) 엔티티 타입(Entity Type)

- 데이터 모델링을 진행하는 첫번째 단계는 엔티티 집합을 정의하는 것으로 엔티티는 현실세계 안의 유.무형의 다른 것들과 구분되는 사물들의 속성을 의미하며 속성들의 집합을 가진다.
- 엔티티는 인스턴스(instance)라고 불리는 개별적인 객체들의 집합이다.

선정절차	설명
개체(Entity) 수집	개체가 될 가능성이 있는 모든 대상을 수집 개념이 모호한 대상은 업무 전문가와 인터뷰를 통해 개념 파악 개체에 해대한 핵심적인 특징을 파악 구현할 시스템의 업의 본질을 항상 염두해 둘 것 * 수집대상 : 현업장표, 인터뷰, DFD, 구시스템, 향후전략, 각 종 문서
개체(Entity) 분류	수집된 개체(Entity)를 Key Entity, Main Entity, Action Entity로 분류

	Key Entity	데이터를 발생시키는 주체로 자신의 부모를 갖지 않는 엔티티 (예: 사원, 부서, 거래처, 자재 등)
	Main Entity	부모로부터 태어난 실체지만 업무의 중심이 되는 실체로 많은 자손을 가짐, 데이터를 발생시키는 주체 (예: 카드, 계약, 단가, 공사..)
	Action Entity	실제 발생하는 업무로 자주 변경되고 데이터가 지속적으로 증가되고 반드시 부모를 가짐 (예: 카드이용, 계약변경, 공사내역..)
개체(Entity) 선정	분류된 개체를 중심으로 Entity의 선정 반드시 어떤 요소들이 포함되는지 명확하게 정의 및 명칭 부여 Entity 명칭은 그 집합을 정확히 정의하는 가장 중요한 역할을 담당	
개체(Entity) 검증	선정한 Entity로서 적합한 자격이 있는지 검증 관리하고자 하는 가로축(속성, 관리항목), 세로축(개체, Instance) 확인 개체와 관계를 명확하게 구분(예: 납입자와 가입자는 Entity가 아니라 Relation)	

(2) 관계(Relationship)

- 관계(Relationship)란 여러 엔티티 간의 업무적인 연관성(association)으로 엔티티 타입의 모든 인스턴스들 즉, 엔티티 집합들 사이의 대응성(Correspondence) 또는 사상(Mapping)을 의미한다.

- 관계를 분류하는 기준으로 대응수(Mapping cardinality)가 있는데 이는 관계를 통해 다른 엔티티와 연결될 수 있는 엔티티의 수를 의미하는 것으로 두 엔티티 집합이 어떤 관계를 맺고 있다고 할 때 반드시 다음 유형 중 하나의 대응수를 가진다.

대응수	설명	사례
일대일(1:1)	관계를 맺고 있는 A 엔티티 집합의 각 원소가 B 엔티티집합의 각 원소와 서로 하나씩 대응되는 관계	
일대다(1:N)	관계를 맺고 있는 A 엔티티 집합의 각 원소가 B 엔티티 집합의 임의의 수의 원소에 대응되는 관계 로 이때 B 엔티티 집합의 하나의 원소 A는 엔티티 집합 의 하나의 원소에만 대응되어야 함	
다대일(N:1)	관계를 맺고 있는 A 엔티티 집합의 각 원소가 최대 한 개의 B 엔티티 집합의 원소에 대응되는 관계. 이때 B 엔티티 집합의 하나의 원소는 A 엔티 티 집합 의 임의의 수의 원소에 대응되는 관계	
다대다(N:M)	관계를 맺고 있는 A 엔티티 집합의 각 원소가 임의의 개수가 있는 B 엔티티 집합의 원소에 대응될 수 있고 B 엔티티 집합의 각 원소 역시 A 엔티티 집합의 임의의 원소에 대응되는 관계	

(3) 속성(Attribute)

- 속성(Attribute)이란 저장할 필요가 있는 엔티티에 관한 정보를 의미하는 것으로서 엔티티의 성질, 분류, 수량, 상태, 특성, 특징 등을 나타내는 세부 사항이다.

- 속성은 최종 데이터베이스 모델링 단계를 통해 테이블의 컬럼(column)으로 활용한다.

유형	설명
기초속성	원래 갖고 있는 속성으로 현업에서 기본적으로 사용되는 속성
추출속성	기초 속성을 가공처리(계산에 의해)하여 얻을 수 있는 속성으로 자료의 중복성 및 무결성 확보를 위해 최소화 하는 것이 바람직함
설계속성	실제로 존재하지는 않으나 시스템의 효율성을 도모하기 위해 설계자가 임의로 부여하는 속성

(4) 식별자(UID : Unique Identifier)

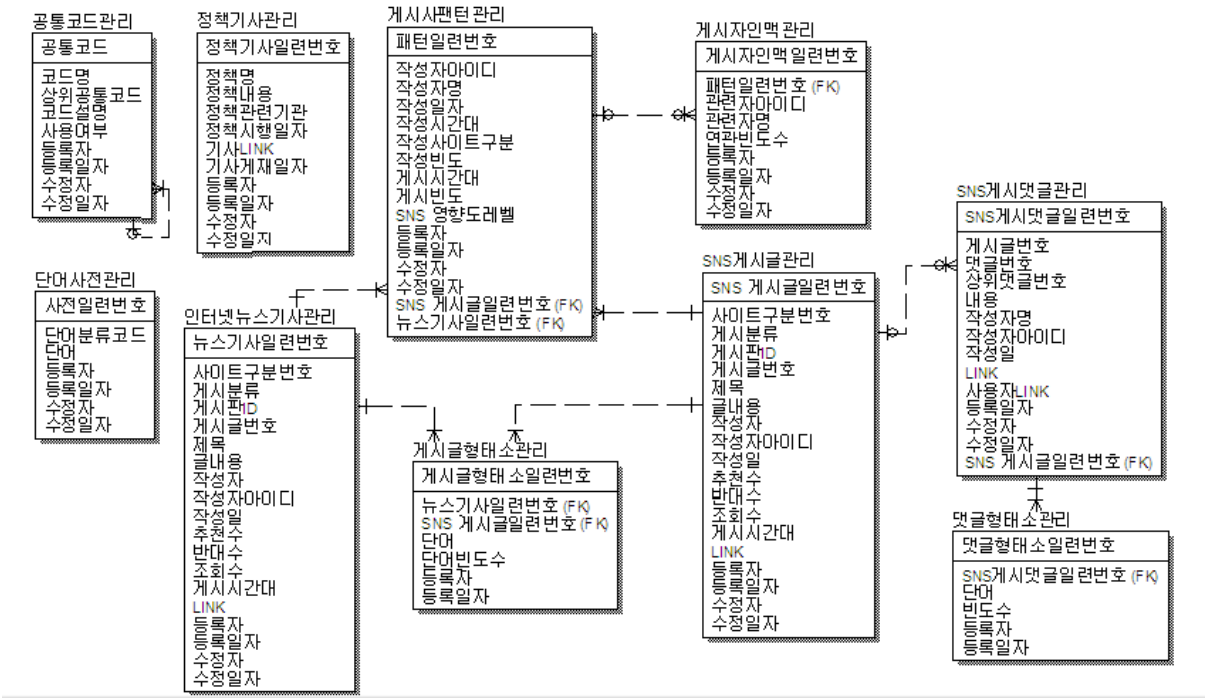
- 식별자(Unique Identifier)는 하나의 엔티티에 구성되어 있는 여러 개의 속성 중에 엔티티를 대표할 수 있는 속성을 의미하며 하나의 엔티티는 반드시 하나의 유일한 식별자가 존재한다.

- 식별자(Unique Identifier)는 하나 또는 그 이상의 속성으로 구성되고 식별자는 논리적인 관점에서 사용되고 키 (Key)는 물리적인 관점에서 사용된다.

- 식별자의 종류는 자신의 엔터티 내에서 대표성을 가지는가에 따라 주식별자(Primary Identifier)와 보조식별 자 (Alternate Identifier)로 구분하고 엔터티 내에서 스스로 생성되었는지 여부에 따라 내부식별자와 외부식별자 (Foreign Identifier)로 구분한다.

- 단일 속성으로 식별이 되는가에 따라 단일식별자(Single Identifier)와 복합식별자 (Composite Identifier)로 구분하고 원래 업무적으로 의미가 있던 식별자 속성을 대체하여 일련번호와 같이 새롭게 만든 식별자를 구분한다.

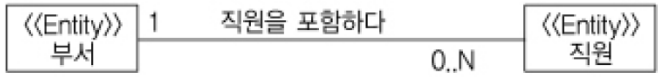
23.2.3 사례



23.3 ERD(Entity Relationship Diagram) 작성 절차

23.3.1 다양한 ERD 표기법

표기법	활용

Chen 	대학교재에서 가장 많이 사용하는 표기법, 실무적으로 많이 사용하지 않음
IDEF1X 	마름모와 원을 이용한 표기법으로 실무현장에서 소수 활용
IE/Crow's Foot 	까마귀 발(Crow's foot) 모양의 표기법으로 가장 많이 사용함 ERWin, ERStudio, eXERD
UML 	스테레오타입을 이용하여 엔티티를 표현 UML로 표현하여 데이터 모델링 할 때 사용 Rational Rose
Case*Method/Barker 	Crow's foot 표기법을 적용하면서 관계 표기법 등 일 부 개선한 모델

23.3.2 ERD 작성 절차

절차	설명
주제영역 설정	• 데이터와 업무 활동 간의 상호관계를 파악하여 정보 수집 대상으로부터 관리될 데이터의 집합을 크게 분류하여 영역을 설정한다. • 설계방식은 개체를 찾고 다음에 속성을 찾는 방식인 하향식(EA : Entity Analysis), 관리가 필요한 데이터 항목이 개체인지 또는 속성으로 분류해야 하는지에 대해 체계적으로 통계분석을 수행하는 상향식(AS : Attribute Analysis), 가장 중요하거나 확실 한 개념 먼저 정의하고 연관된 개념을 확장시켜 나가는 방식인 외향식(inside-out) 등으로 구분한다.
핵심 개체 설정	• 핵심 개체란 단위 주제 영역 내에서 가장 핵심이 되는 개체로 단위 주체 영역과 동일한 이름을 가지는 것으로 업무기술서, 장표, 인터뷰 문서등의 명사에서 도출한다. • 개체 명명의 원칙은 단수 명사를 사용하고 개체명은 유일해야 하며 가능한 현업 업무에서 사용하는 용어를 사용한다.
관계 설정	• 업무의 연관성에 따라 개체 간의 관계를 설정하는 것으로 관련 업무에 따라 발생하는 데이터 간의 양방향 업무 규칙을 표현한다.
핵심 속성 정의	• 정보를 구성하는 최소 단위의 데이터 항목 중 개체와 관계의 특징을 표현하는 요소로 원자단위, 유일성, 독립성을 보유해야 한다. • 식별자로 사용될 속성 및 개체의 주요 속성이 핵심 속성의 대상이다.
식별자 설정	■ 한 개체에서 각각의 인스턴스를 유일하게 구분할 수 있는 속성 또는 속성들의 집합식별자는 후보키, 기본키, 대체키, 외래키로 구분한다.

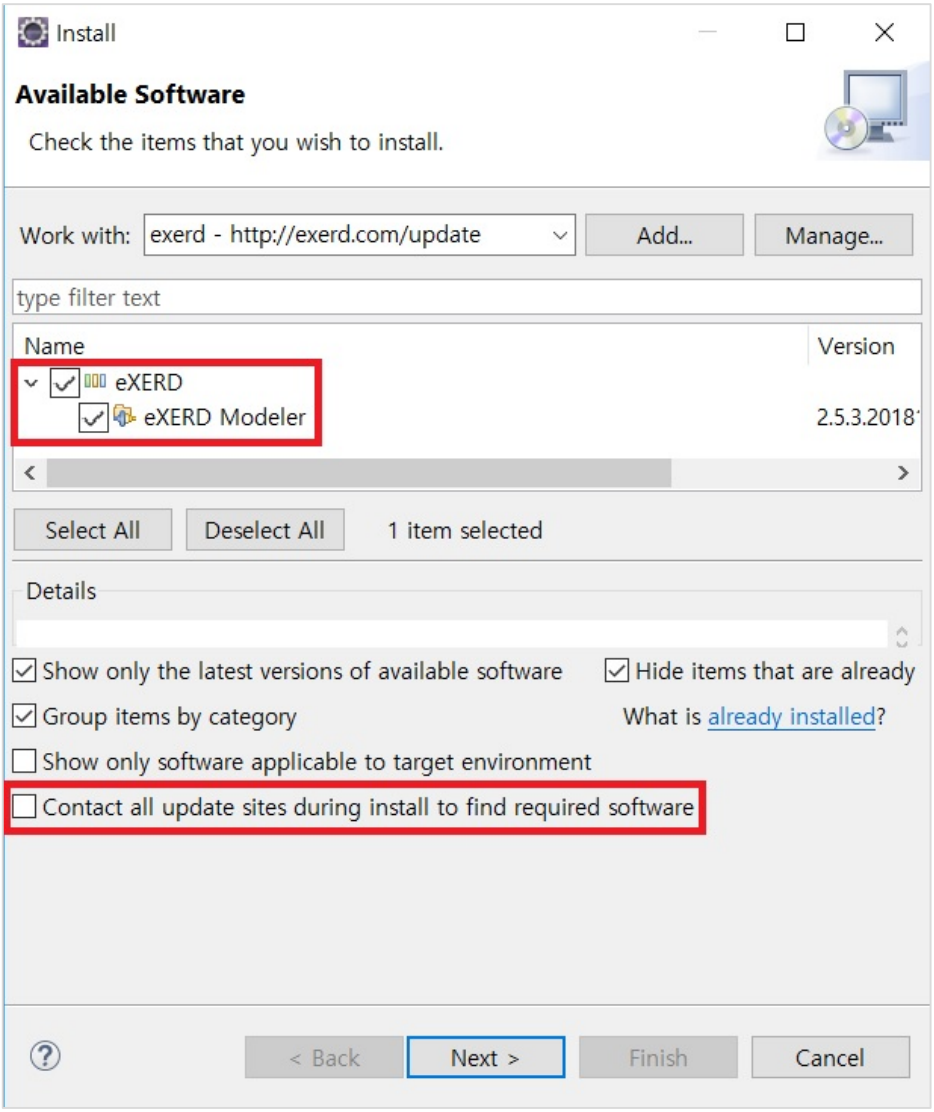
23.4 ERD 도구 - eXERD

23.4.1 Eclipse 플러그인으로 설치

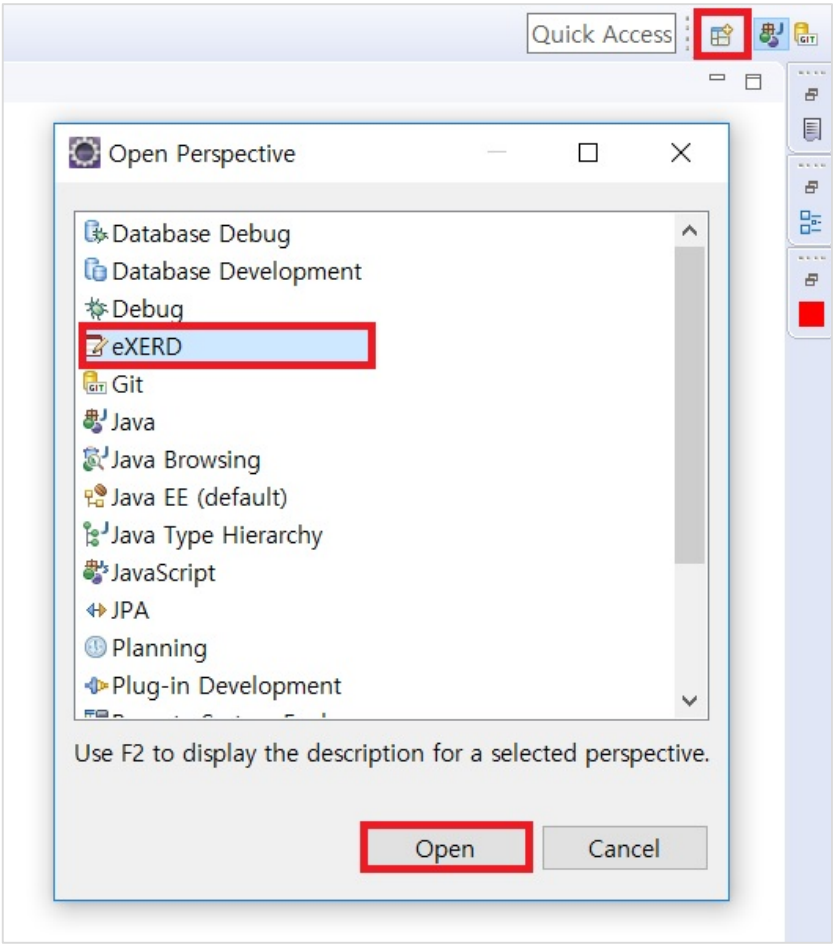
■ Eclipse 메뉴 Help > Install New Software > Add 버튼을 클릭

■ Add Repository 창에서 아래와 같이 입력하고 Add 버튼을 클릭
- Name: exerd
- Location: http://exerd.com/update

■ Available Software 창에서 아래와 같이 선택한다.
- eXERD 선택
- eXERD Modeler 선택
- Details 마지막 항목 체크 해제



■ 설치 후, Eclipse 화면에서 Open Perspective 버튼 클릭 > eXERD 클릭 > Open 버튼 클릭



[꿀팁] eXERD 평가판으로 설치

- 다운로드: http://ko.exerd.com/#download-section

23.4.2 관계 설정

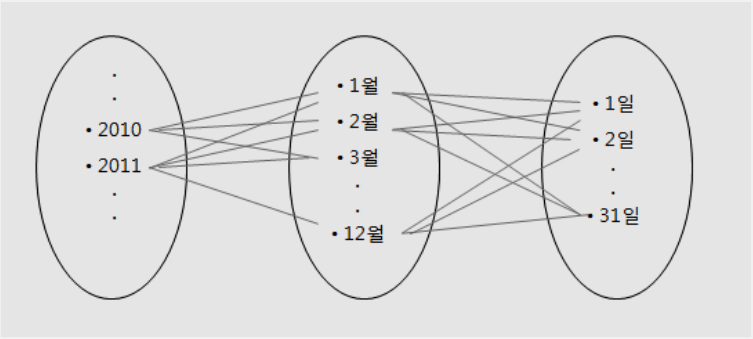
- 부모테이블의 유니크 키나 기본키로 지정된 컬럼이 자식테이블의 기본키 컬럼과 연결된 경우, 실선은 식별 관계라고 하며 점선은 비식별 관계라고 한다.

- 쉽게 이야기하면 부모 컬럼을 참조하는 자식 컬럼이 식별 (PK) 가능하면 식별 관계라고 하고 식별이 가능하지 않으면 비식별 관계라고 한다.

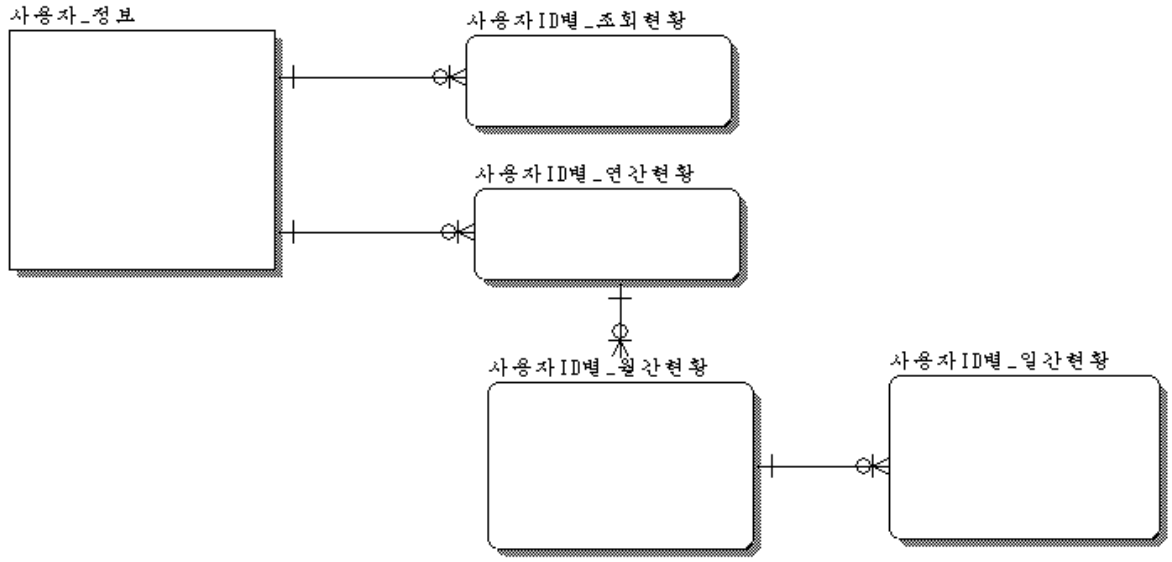
[실습] 사용자ID별 조회수 통계 모델링

■ 목표 시스템
- 사용자ID별 조회수 통계 집계

■ 요구사항
- 사용자ID별 조회수 통계를 작성할 수 있어야 한다.
- 사용자ID 기준으로 연, 월, 일자별 조회수 통계를 작성할 수 있어야 한다.

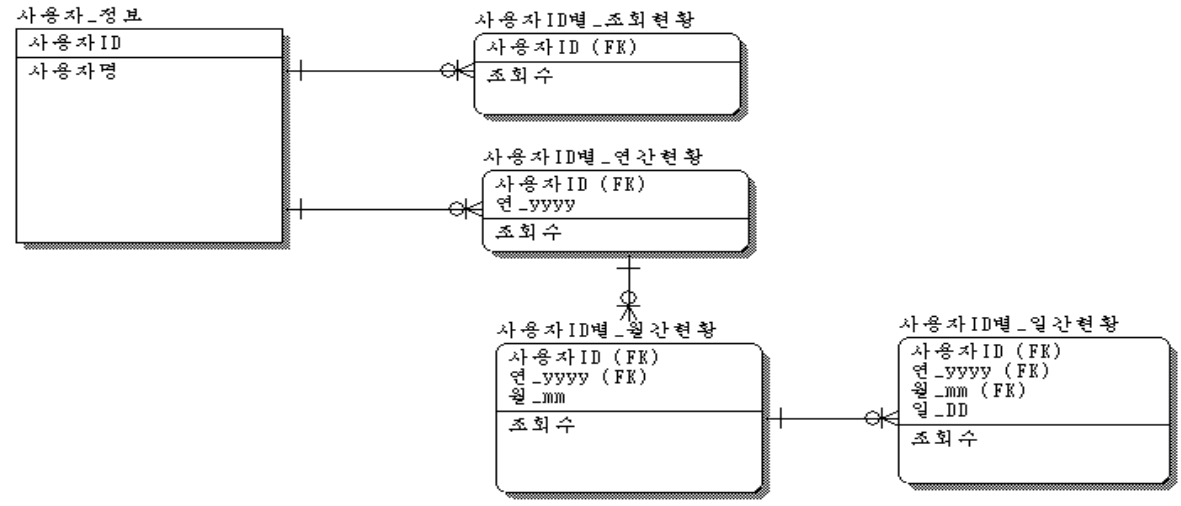


(1) 핵심 개체(Entity) 식별 및 관계 정의
- 현재 제시된 요구사항을 중심으로 개체 식별하여 정의한다.
- Entity 식별
  - 기본 사용자 정보
  - 사용자ID별 전체 조회수 검색을 위한 사용자ID별 조회현황
  - 연간 요약(Summary) 개체인 사용자ID별 연간현황
  - 중간 요약(Summary) 개체인 사용자ID별 월간현황
  - 일자별 조회수 통계인 사용자ID별 일간현황
- Relationship 식별
  - 일반적으로 통계형 개체인 경우 관계를 정의하지 않는 것이 통상적인 관례이나 위의 사례에서는 관계성을 정의하기 위해 날짜를 중심으로 관계 형성
  - 날짜 중심으로 데이터의 상관성을 보았을 때 다음과 같이 정의 가능



(2) 속성(Attribute) 및 식별자(UID) 정의

- 각 각의 엔티티를 유일하게 식별하기 위한 속성 및 식별자 정의
  - 사용자_정보(사용자ID, 사용자명)
  - 사용자ID별_연간현황(사용자ID, 연_yyyy, 조회수)
  - 사용자ID별_월간현황(사용자ID, 연_yyyy, 월_mm, 조회수)
  - 사용자ID별_일간현황(사용자ID, 연_yyyy, 월_mm, 일_dd, 조회수)
  - 사용자ID별_조회현황(사용자ID, ,조회수)
- 통계형 Entity의 경우 개념/논리 모델링에서는 관계를 식별하여 표기 하나 실제 물리 모델링에서는 배치작업 등으로 데이터를 입력하는 경우가 대부분이라 관계를 Foreign Key 로 정의하지 않음



(3) DDL(Data Definition Language) 명령문

실습을 위해 아래와 같이 사전 환경 설정(사용자 계정 생성)을 하도록 한다.

```
-- 1. 사용자 생성하기
DROP USER USER10 CASCADE;
CREATE USER USER10 IDENTIFIED BY USER10;
GRANT CONNECT, RESOURCE TO USER10;
GRANT CREATE SESSION TO USER10;
GRANT CREATE TABLE TO USER10;
```

```
ALTER USER USER10 DEFAULT TABLESPACE USERS QUOTA UNLIMITED ON USERS;

-- 2. ERD를 통한 포워드 엔지리어링 혹은 ERD로 생성한 DDL문을 실행한다.
```

다음은 위 ERD에 대한 객체 생성을 위한 DDL문이다.

[USER10.exerd]

```
01    /* 사용자_정보 */
02    CREATE TABLE USR_I (
03              USERID VARCHAR2(20) NOT NULL, /* 사용자ID */
04              USERNAME VARCHAR2(20) /* 사용자명 */
05    );
06
07    CREATE UNIQUE INDEX PK_USR_I
08              ON USR_I (
09                        USERID ASC
10              );
11
12    ALTER TABLE USR_I
13              ADD
14                        CONSTRAINT PK_USR_I
15                        PRIMARY KEY (
16                                  USERID
17                        );
18
19    /* 사용자ID별_조회현황 */
20    CREATE TABLE USR_S (
21              USERID VARCHAR2(20) NOT NULL, /* 사용자ID */
22              COUNT NUMBER NOT NULL /* 조회수 */
23    );
24
25    /* 사용자ID별_연간현황 */
26    CREATE TABLE USR_Y (
27              USERID VARCHAR2(20) NOT NULL, /* 사용자ID */
28              YEAR NUMBER NOT NULL, /* 연_yyyy */
29              COUNT NUMBER /* 조회수 */
30    );
31
32    CREATE UNIQUE INDEX PK_USR_Y
33              ON USR_Y (
34                        USERID ASC,
35                        YEAR ASC
36              );
37
38    ALTER TABLE USR_Y
39              ADD
40                        CONSTRAINT PK_USR_Y
41                        PRIMARY KEY (
42                                  USERID,
43                                  YEAR
44                        );
45
46    /* 사용자ID별_월간현황 */
47    CREATE TABLE USR_M (
48              USERID VARCHAR2(20) NOT NULL, /* 사용자ID */
49              YEAR NUMBER NOT NULL, /* 연_yyyy */
50              MONTH NUMBER NOT NULL, /* 월_mm */
51              COUNT NUMBER /* 조회수 */
52    );
53
54    CREATE UNIQUE INDEX PK_USR_M
55              ON USR_M (
56                        USERID ASC,
57                        YEAR ASC,
```

```
58                      MONTH ASC
59              );
60
61      ALTER TABLE USR_M
62              ADD
63                      CONSTRAINT PK_USR_M
64                      PRIMARY KEY (
65                              USERID,
66                              YEAR,
67                              MONTH
68                      );
69
70      /* 사용자ID별_일간현황 */
71      CREATE TABLE USR_D (
72              YEAR NUMBER NOT NULL, /* 연_yyyy */
73              USERID VARCHAR2(20) NOT NULL, /* 사용자ID */
74              MONTH NUMBER NOT NULL, /* 월_mm */
75              DAY NUMBER NOT NULL, /* 일_DD */
76              COUNT NUMBER /* 조회수 */
77      );
78
79      CREATE UNIQUE INDEX PK_USR_D
80              ON USR_D (
81                      YEAR ASC,
82                      USERID ASC,
83                      MONTH ASC,
84                      DAY ASC
85              );
86
87      ALTER TABLE USR_D
88              ADD
89                      CONSTRAINT PK_USR_D
90                      PRIMARY KEY (
91                              YEAR,
92                              USERID,
93                              MONTH,
94                              DAY
95                      );
96
97      ALTER TABLE USR_S
98              ADD
99                      CONSTRAINT FK_USR_I_TO_USR_S
100                     FOREIGN KEY (
101                             USERID
102                     )
103                     REFERENCES USR_I (
104                             USERID
105                     );
106
107     ALTER TABLE USR_Y
108             ADD
109                     CONSTRAINT FK_USR_I_TO_USR_Y
110                     FOREIGN KEY (
111                             USERID
112                     )
113                     REFERENCES USR_I (
114                             USERID
115                     );
116
117     ALTER TABLE USR_M
118             ADD
119                     CONSTRAINT FK_USR_Y_TO_USR_M
120                     FOREIGN KEY (
121                             USERID,
122                             YEAR
```

```
123                     )
124                     REFERENCES USR_Y (
125                               USERID,
126                               YEAR
127                     );
128
129   ALTER TABLE USR_D
130             ADD
131                     CONSTRAINT FK_USR_M_TO_USR_D
132                     FOREIGN KEY (
133                               USERID,
134                               YEAR,
135                               MONTH
136                     )
137                     REFERENCES USR_M (
138                               USERID,
139                               YEAR,
140                               MONTH
141                     );
```

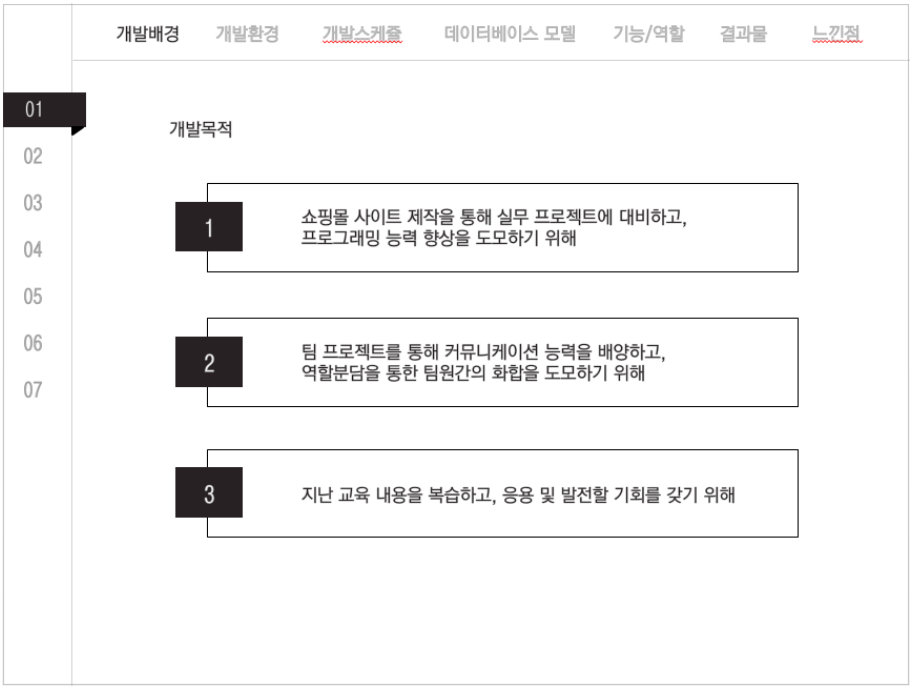
(4) DML을 통한 DB모델링 검증

```
-- 1. 더미(dummy) 데이터 입력한다.
INSERT INTO "USER10"."USR_I" (USERID, USERNAME) VALUES ('TEST1', 'TEST1');
INSERT INTO "USER10"."USR_I" (USERID, USERNAME) VALUES ('TEST2', 'TEST2');
SELECT * FROM USR_I;

-- 2. 요구사항에 맞는 검증 SQL문을 작성한다.
INSERT INTO USR_Y VALUES ('TEST1',2017,132);
INSERT INTO USR_Y VALUES ('TEST2',2017,396);
INSERT INTO USR_Y VALUES ('TEST1',2018,1);
INSERT INTO USR_Y VALUES ('TEST2',2018,5);
INSERT INTO USR_Y VALUES ('TEST3',2018,32);
SELECT * FROM USR_Y;
INSERT INTO USR_M VALUES ('TEST1',2018,1,0);
INSERT INTO USR_M VALUES ('TEST1',2018,2,1);
INSERT INTO USR_M VALUES ('TEST1',2018,2,3);
SELECT * FROM USR_M;

-- 참고: 만약, 해당 ACCOUNT가 접속되어 있어 다음과 같이 삭제할 수 없는 경우
-- ORA-01940: 현재 접속되어 있는 사용자는 삭제할 수 없습니다
-- 01940. 00000 -  "cannot drop a user that is currently connected"
SELECT SID,SERIAL# FROM V$SESSION WHERE USERNAME = 'USER10';
ALTER SYSTEM KILL SESSION '<SID>, <SERIAL>';
DROP USER USER10 CASCADE;
```

[실습] 고객관리 모델링

1 프로젝트 계획 수립

■ 프로젝트의 주제를 정하고 그와 관련된 배경, 목표, 기능, 역할 등을 기술한다.



2 요구사항 분석

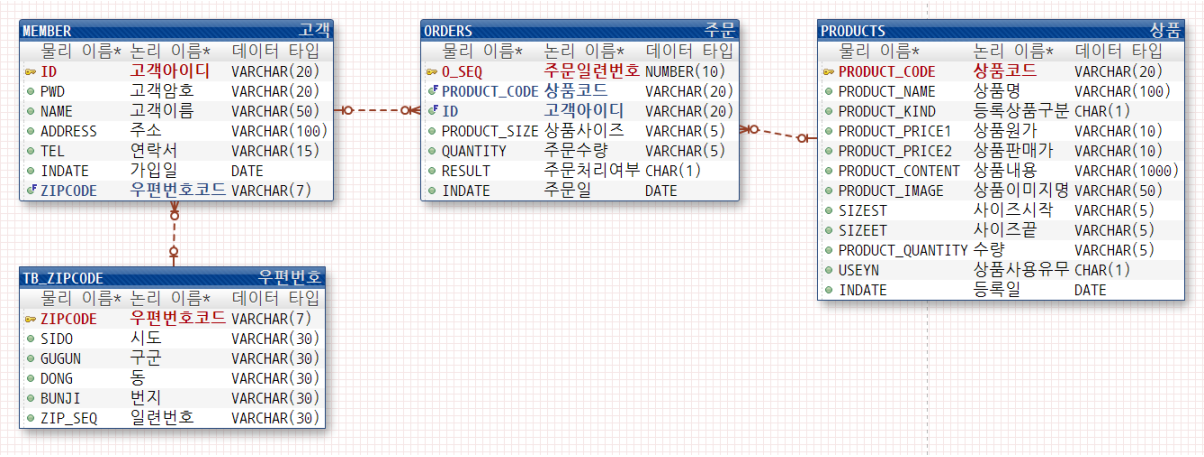
■ 구축하려는 목표시스템과 관련된 요구사항을 정리하여 아래와 같이 "요구사항명세서"를 작성한다.

일련번호	기능	요구사항	담당자	우선순위	비고
1	운영자 기능	운영자는 상품을 등록할 수 있어야 한다.			
2	운영자 기능	운영자는 주문 내역을 조회할 수 있어야 한다.			
3	계정하기	사용자가 직접 회원가입을 할 수 있어야 한다.	홍길동	1	
4	로그인하기	이미 가입된 회원의 아이디로 검색할 수 있어야 한다.			
5	로그인하기	이미 가입된 회원의 아이디로 검색할 수 있어야 한다.			
6	상품 상세보기				
7	장바구니에 상품 담기				
8	주문하기	사용자가 주문 정보를 추가할 수 있어야 한다.			
9	주문 내역보기	운영자는 사용자가 주문한 내용을 조회할 수 있어야 한다.			

3 DB 모델링

3.1 논리적 모델링

■ 요구사항명세서를 바탕으로 테이블, 컬럼, 데이터타입, 제약조건 및 관계도를 표기한 "ERD"를 작성한다.



3.2 물리적 모델링

■ ERD를 바탕으로 "테이블정의서, DDL, DML"를 작성하여 물리적인 DB모델링을 구축한다.

(1) 테이블정의서

엔티티 타입명	우편번호	작성일	20xx xx xx
테이블명	TB_ZIPCODE	작성자	홍길동
테이블 설명			

번호	컬럼명	속성명	도메인	데이터타입	NULL여부	기본값	KEY
1	ZIPCODE	우편번호코드	N/A	VARCHAR(7)	Not Null		PK
2	SIDO	시도	N/A	VARCHAR(30)			
3	GUGUN	구군	N/A	VARCHAR(30)			
4	DONG	동	N/A	VARCHAR(30)			
5	BUNJI	번지	N/A	VARCHAR(30)			
6	ZIP_SEQ	일련번호	N/A	VARCHAR(30)	Not Null		

엔티티 타입명	고객	작성일	20xx xx xx
테이블명	MEMBER	작성자	홍길동
테이블 설명			

번호	컬럼명	속성명	도메인	데이터타입	NULL여부	기본값	KEY
1	ID	고객아이디	N/A	VARCHAR(20)	Not Null		PK
2	PWD	고객암호	N/A	VARCHAR(20)			
3	NAME	고객이름	N/A	VARCHAR(50)			
4	ADDRESS	주소	N/A	VARCHAR(100)			
5	TEL	연락서	N/A	VARCHAR(15)			
6	INDATE	가입일	N/A	DATE			
7	ZIPCODE	우편번호코드	N/A	VARCHAR(7)			FK

(2) DDL

■ 실질적으로 테이블, 컬럼, 제약조건을 데이터베이스에 생성한다.

```
/* 우편번호 */
CREATE TABLE TB_ZIPCODE (
          ZIPCODE VARCHAR(7) NOT NULL, /* 우편번호코드 */
          SIDO VARCHAR(30), /* 시도 */
          GUGUN VARCHAR(30), /* 구군 */
          DONG VARCHAR(30), /* 동 */
          BUNJI VARCHAR(30), /* 번지 */
          ZIP_SEQ VARCHAR(30) NOT NULL /* 일련번호 */
);
```

```
COMMENT ON TABLE TB_ZIPCODE IS '우편번호';

COMMENT ON COLUMN TB_ZIPCODE.ZIPCODE IS '우편번호코드';

COMMENT ON COLUMN TB_ZIPCODE.SIDO IS '시도';

COMMENT ON COLUMN TB_ZIPCODE.GUGUN IS '구군';

COMMENT ON COLUMN TB_ZIPCODE.DONG IS '동';

COMMENT ON COLUMN TB_ZIPCODE.BUNJI IS '번지';

COMMENT ON COLUMN TB_ZIPCODE.ZIP_SEQ IS '일련번호';

CREATE UNIQUE INDEX PK_TB_ZIPCODE
          ON TB_ZIPCODE (
                    ZIPCODE ASC
          );

ALTER TABLE TB_ZIPCODE
          ADD
                    CONSTRAINT PK_TB_ZIPCODE
                    PRIMARY KEY (
                              ZIPCODE
                    );

/* 고객 */
CREATE TABLE MEMBER (
          ID VARCHAR(20) NOT NULL, /* 고객아이디 */
          PWD VARCHAR(20), /* 고객암호 */
          NAME VARCHAR(50), /* 고객이름 */
          ADDRESS VARCHAR(100), /* 주소 */
          TEL VARCHAR(15), /* 연락서 */
          INDATE DATE, /* 가입일 */
          ZIPCODE VARCHAR(7) /* 우편번호코드 */
);

COMMENT ON TABLE MEMBER IS '고객';

COMMENT ON COLUMN MEMBER.ID IS '고객아이디';

COMMENT ON COLUMN MEMBER.PWD IS '고객암호';

COMMENT ON COLUMN MEMBER.NAME IS '고객이름';

COMMENT ON COLUMN MEMBER.ADDRESS IS '주소';

COMMENT ON COLUMN MEMBER.TEL IS '연락서';

COMMENT ON COLUMN MEMBER.INDATE IS '가입일';

COMMENT ON COLUMN MEMBER.ZIPCODE IS '우편번호코드';

CREATE UNIQUE INDEX PK_MEMBER
          ON MEMBER (
                    ID ASC
          );

ALTER TABLE MEMBER
          ADD
                    CONSTRAINT PK_MEMBER
                    PRIMARY KEY (
                              ID
                    );
```

```
/* 상품 */
CREATE TABLE PRODUCTS (
          PRODUCT_CODE VARCHAR(20) NOT NULL, /* 상품코드 */
          PRODUCT_NAME VARCHAR(100), /* 상품명 */
          PRODUCT_KIND CHAR(1), /* 등록상품구분 */
          PRODUCT_PRICE1 VARCHAR(10), /* 상품원가 */
          PRODUCT_PRICE2 VARCHAR(10), /* 상품판매가 */
          PRODUCT_CONTENT VARCHAR(1000), /* 상품내용 */
          PRODUCT_IMAGE VARCHAR(50), /* 상품이미지명 */
          SIZEST VARCHAR(5), /* 사이즈시작 */
          SIZEET VARCHAR(5), /* 사이즈끝 */
          PRODUCT_QUANTITY VARCHAR(5), /* 수량 */
          USEYN CHAR(1), /* 상품사용유무 */
          INDATE DATE /* 등록일 */
);

COMMENT ON TABLE PRODUCTS IS '상품';

COMMENT ON COLUMN PRODUCTS.PRODUCT_CODE IS '상품코드';

COMMENT ON COLUMN PRODUCTS.PRODUCT_NAME IS '상품명';

COMMENT ON COLUMN PRODUCTS.PRODUCT_KIND IS '등록상품구분';

COMMENT ON COLUMN PRODUCTS.PRODUCT_PRICE1 IS '상품원가';

COMMENT ON COLUMN PRODUCTS.PRODUCT_PRICE2 IS '상품판매가';

COMMENT ON COLUMN PRODUCTS.PRODUCT_CONTENT IS '상품내용';

COMMENT ON COLUMN PRODUCTS.PRODUCT_IMAGE IS '상품이미지명';

COMMENT ON COLUMN PRODUCTS.SIZEST IS '사이즈시작';

COMMENT ON COLUMN PRODUCTS.SIZEET IS '사이즈끝';

COMMENT ON COLUMN PRODUCTS.PRODUCT_QUANTITY IS '수량';

COMMENT ON COLUMN PRODUCTS.USEYN IS '상품사용유무';

COMMENT ON COLUMN PRODUCTS.INDATE IS '등록일';

CREATE UNIQUE INDEX PK_PRODUCTS
          ON PRODUCTS (
                    PRODUCT_CODE ASC
          );

ALTER TABLE PRODUCTS
          ADD
                    CONSTRAINT PK_PRODUCTS
                    PRIMARY KEY (
                              PRODUCT_CODE
                    );

/* 주문 */
CREATE TABLE ORDERS (
          O_SEQ NUMBER(10) NOT NULL, /* 주문일련번호 */
          PRODUCT_CODE VARCHAR(20), /* 상품코드 */
          ID VARCHAR(20), /* 고객아이디 */
          PRODUCT_SIZE VARCHAR(5), /* 상품사이즈 */
          QUANTITY VARCHAR(5), /* 주문수량 */
          RESULT CHAR(1), /* 주문처리여부 */
          INDATE DATE /* 주문일 */
);
```

```
COMMENT ON TABLE ORDERS IS '주문';

COMMENT ON COLUMN ORDERS.O_SEQ IS '주문일련번호';

COMMENT ON COLUMN ORDERS.PRODUCT_CODE IS '상품코드';

COMMENT ON COLUMN ORDERS.ID IS '고객아이디';

COMMENT ON COLUMN ORDERS.PRODUCT_SIZE IS '상품사이즈';

COMMENT ON COLUMN ORDERS.QUANTITY IS '주문수량';

COMMENT ON COLUMN ORDERS.RESULT IS '주문처리여부';

COMMENT ON COLUMN ORDERS.INDATE IS '주문일';

CREATE UNIQUE INDEX PK_ORDERS
          ON ORDERS (
                    O_SEQ ASC
          );

ALTER TABLE ORDERS
          ADD
                    CONSTRAINT PK_ORDERS
                    PRIMARY KEY (
                              O_SEQ
                    );

ALTER TABLE MEMBER
          ADD
                    CONSTRAINT FK_TB_ZIPCODE_TO_MEMBER
                    FOREIGN KEY (
                              ZIPCODE
                    )
                    REFERENCES TB_ZIPCODE (
                              ZIPCODE
                    );

ALTER TABLE ORDERS
          ADD
                    CONSTRAINT FK_MEMBER_TO_ORDERS
                    FOREIGN KEY (
                              ID
                    )
                    REFERENCES MEMBER (
                              ID
                    );

ALTER TABLE ORDERS
          ADD
                    CONSTRAINT FK_PRODUCTS_TO_ORDERS
                    FOREIGN KEY (
                              PRODUCT_CODE
                    )
                    REFERENCES PRODUCTS (
                              PRODUCT_CODE
                    );
```

4 DB 검증

■ 요구사항 항목을 바탕으로 DB모델링를 DML로 검증한다.

(1) DML

일련번호	기능	DML
1	운영자 기능	INSERT INTO PRODUCTS(PRODUCT_CODE, PRODUCT_NAME, PRODUCT_KIND, PRODUCT_PRICE1, PRODUCT_PRICE2, PRODUCT_CONTENT, PRODUCT_IMAGE, SIZEST,SIZEET, PRODUCT_QUANTITY) VALUES('HI0001','회색힐','1','10000','12000','여성용전용 힐', 'W9.JPG', '230', '255', '100');
2	운영자 기능	
3	계정하기	INSERT INTO MEMBER (ID,PWD,NAME,ZIPCODE,ADDRESS,TEL) VALUES('ONE','1111','김일동','152-761','서울시 구로구 구로1동 주공아파트 104-1004호','010-111-1111');
4	로그인하기	SELECT COUNT(*) FROM MEMBER WHERE ID='ONE';
5	로그인하기	이미 가입된 회원의 아이디로 패스워드를 검색할 수 있어야 한다.
6	상품 상세보기	
7	장바구니에 상품 담기	

8	주문하기	-- 시퀀스 생성하기
		CREATE SEQUENCE ORDERS_SEQ
		START WITH 1
		INCREMENT BY 1
		MAXVALUE 100000;
		-- 트리거 생성하기
		CREATE OR REPLACE TRIGGER ORDERS_TR01
		BEFORE INSERT
		ON ORDERS
		FOR EACH ROW
		BEGIN
		SELECT ORDERS_SEQ.NEXTVAL INTO :NEW.O_SEQ FROM DUAL;
		END;
		/
		-- 주문정보 추가하기
		INSERT INTO ORDERS (ID, PRODUCT_CODE, PRODUCT_SIZE, QUANTITY)
		VALUES ('ONE', 'HI0001', '235', '5');

9	주문 내역보기	-- 조인하기 SELECT * FROM ORDERS O, MEMBER M, PRODUCTS P WHERE O.ID=M.ID AND O.PRODUCT_CODE=P.PRODUCT_CODE; -- 뷰 생성하기