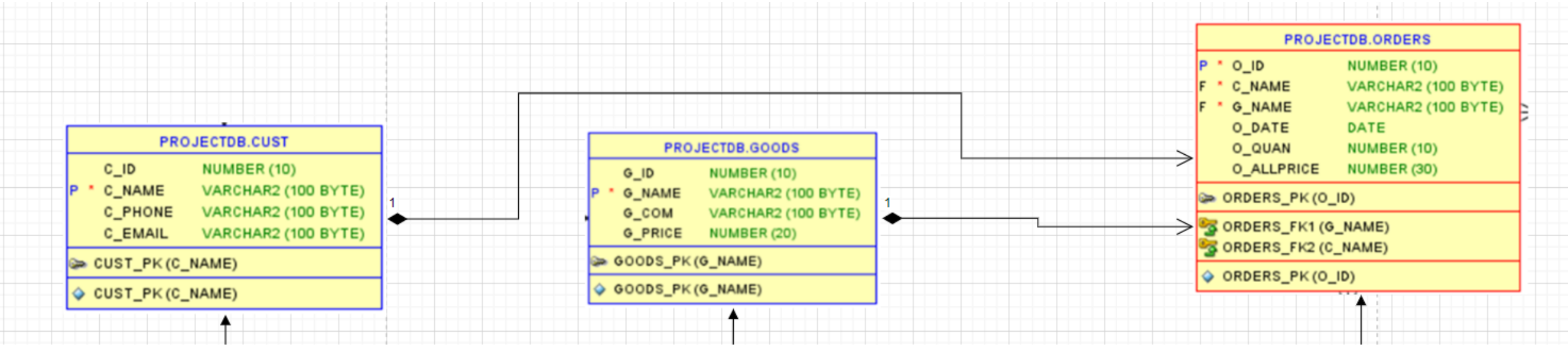
3차 미니 프로젝트

오스템 임플란트 교육생
정재현

목차

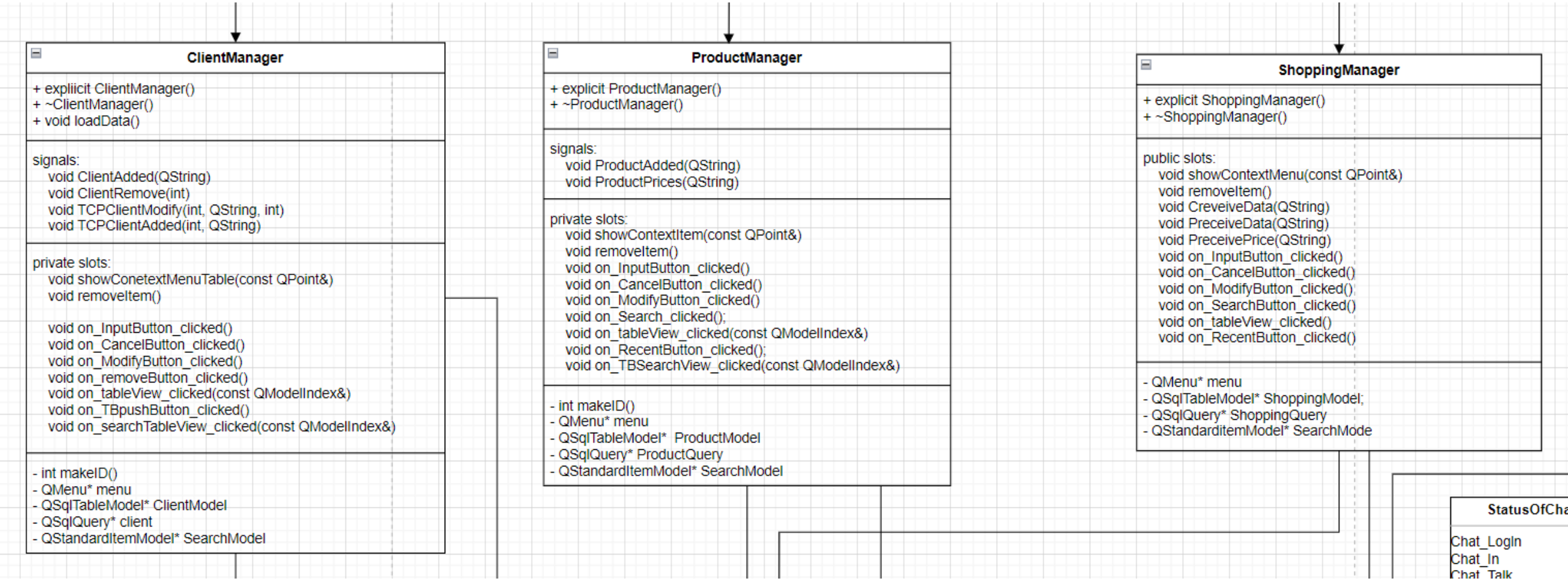
수정된 2차 미니프로젝트 UML다이어그램

데이터 베이스 테이블



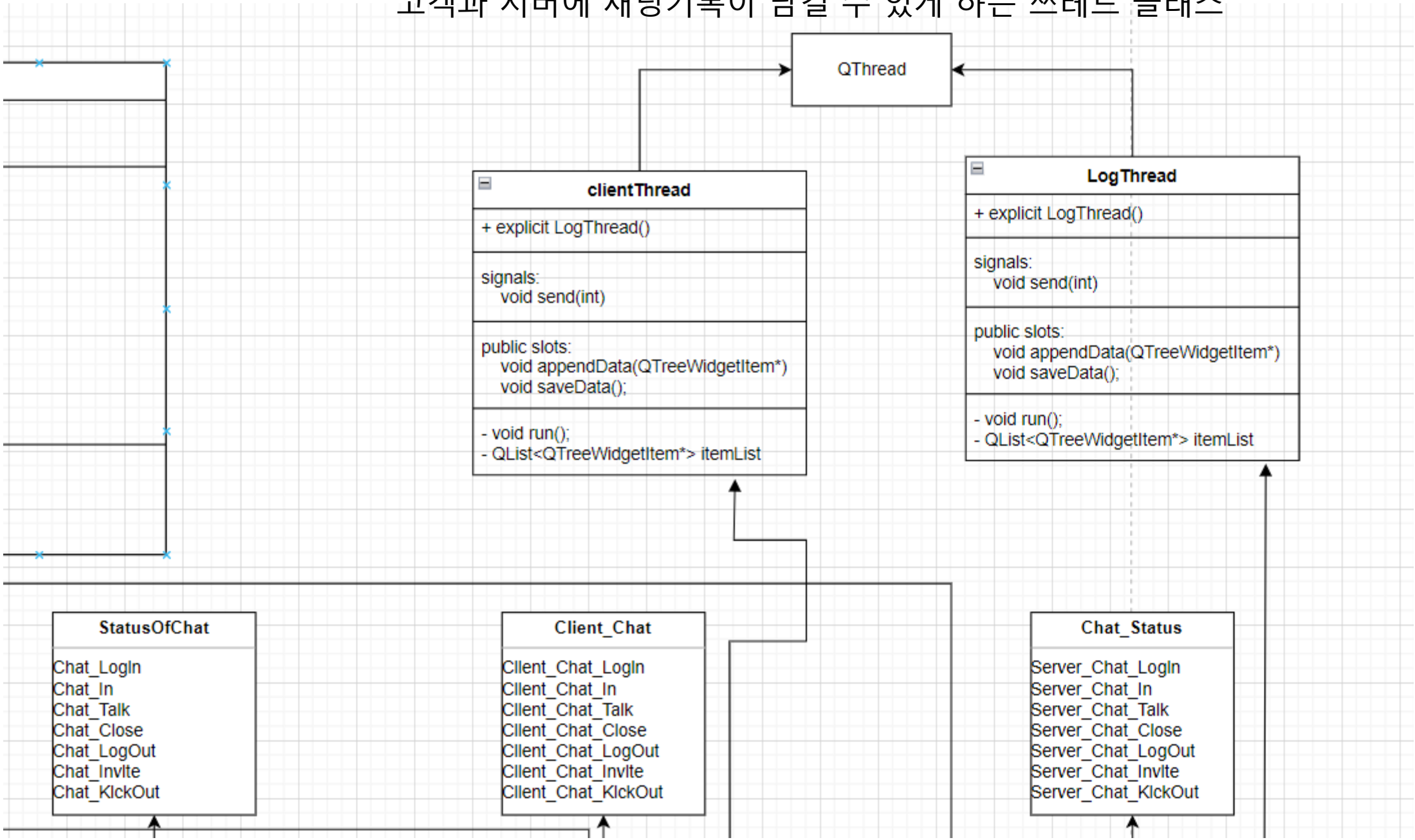
고객과 상품의 이름이 유일성을 가지고 한 개 의 이름이 여러 주문 정보로 1 대 N 관계를 형성

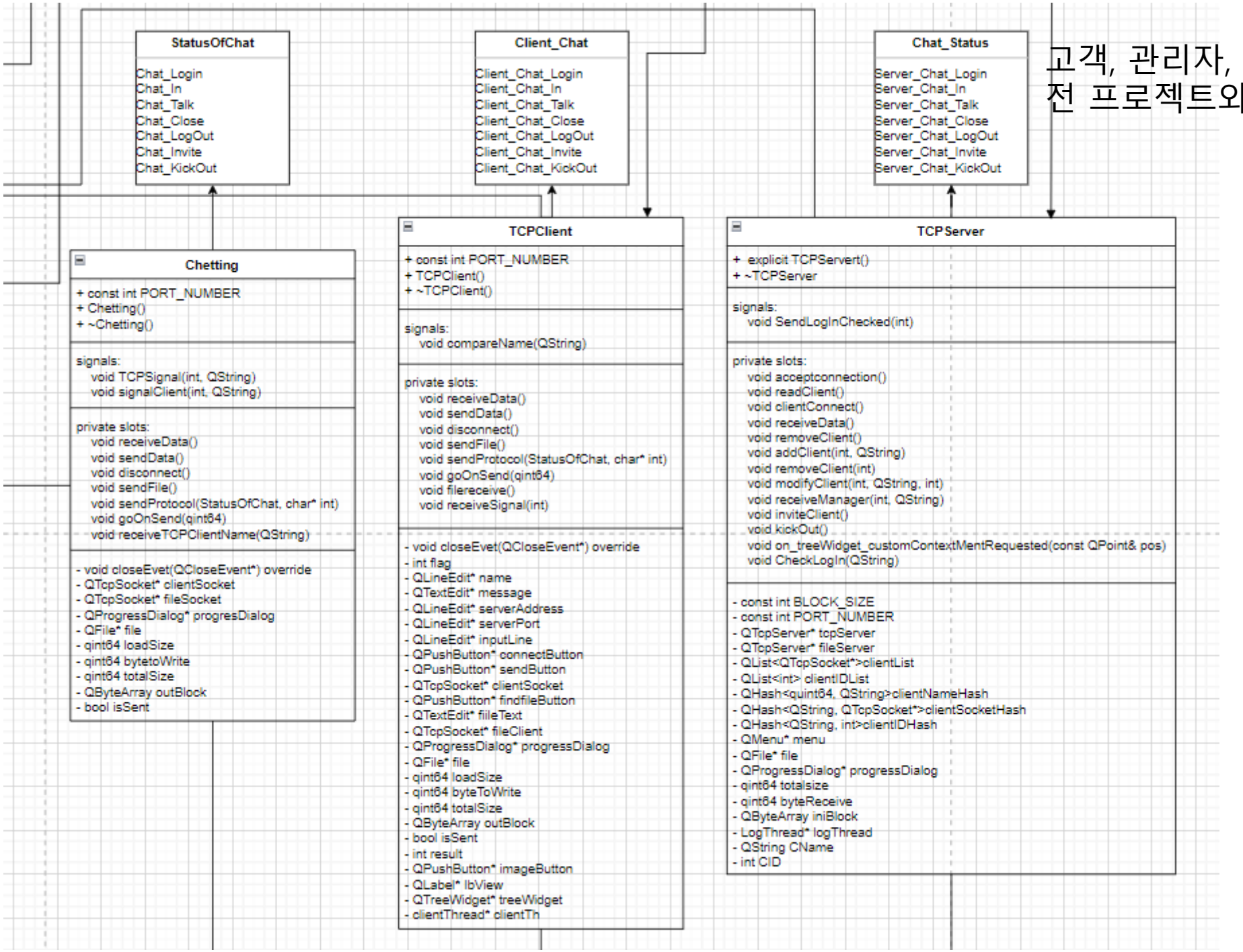
고객, 상품 주문정보 매니저 클래스



2차 미니프로젝트와는 다르게 데이테베이스의 데이터를 받고 기존에 파일로 보관하던 데이터는 사용하지 않음

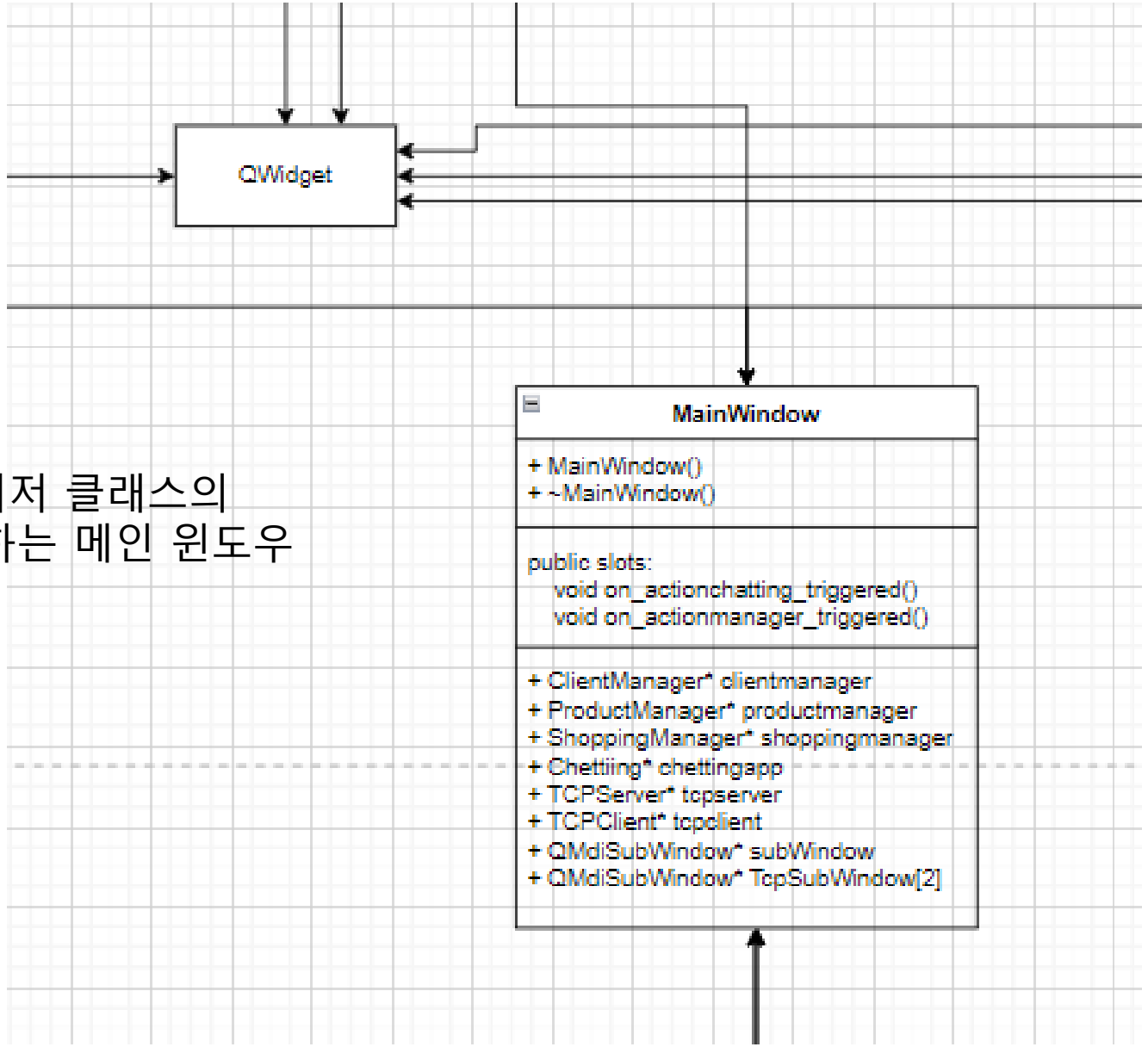
고객과 서버에 채팅기록이 남길 수 있게 하는 쓰레드 클래스



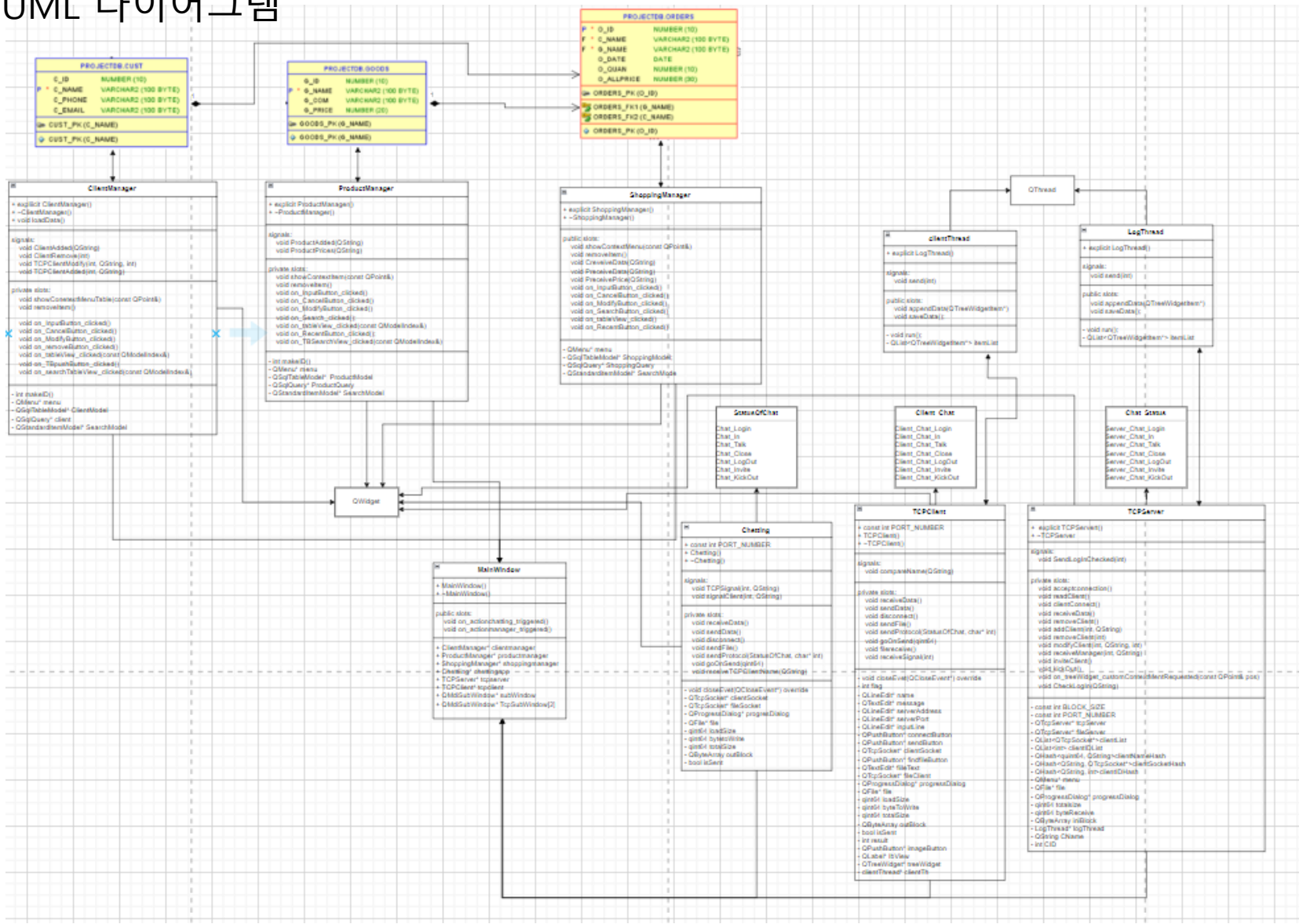


고객, 관리자, 서버 클래스는 전 프로젝트와 동일

채팅, 서버, 매니저 클래스의
컨트롤을 담당하는 메인 윈도우



전체적인 UML 다이어그램



2차 미니프로젝트와 다른점

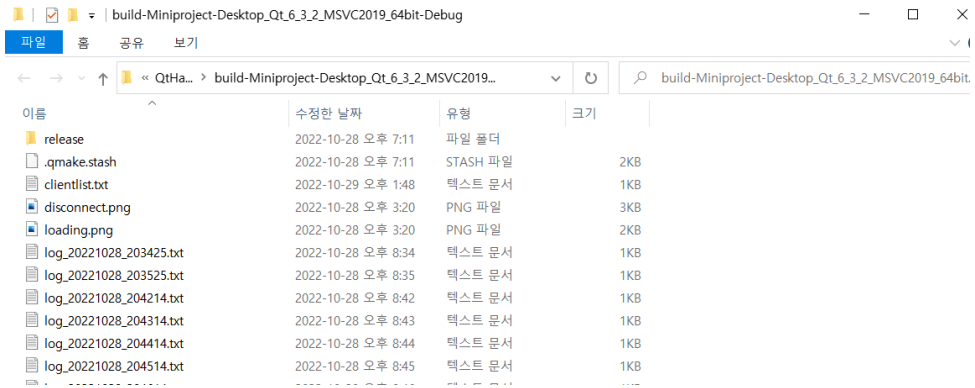
데이터베이스를 추가, UI에서 테이블을 호출하여
입력, 수정, 삭제가 가능하게끔 설정

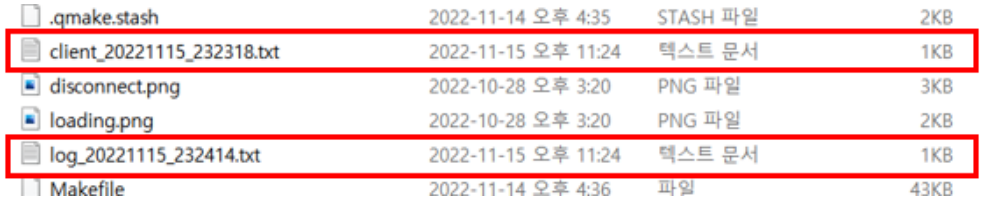
ClientID	ClientName	ClientPhone	ClientEmail
1	정재현	010-2464-2739	0306jh@naver.c...
2	정다현	010-9156-2739	not-found
3	정해영	010-5637-2739	not-found
4	정민희	010-3421-2739	not-found

ClientDB

	c_id	c_name	c_phone	c_email
1	1	정재현	010-2464-2739	0306jh@naver.com
2	2	정다현	010-9156-2739	not-found
3	3	정해영	010-5637-2739	not-found
4	4	정민희	010-3421-2739	not-found

2차 미니프로젝트의 트리 위젯을 데이터베이스로 변경





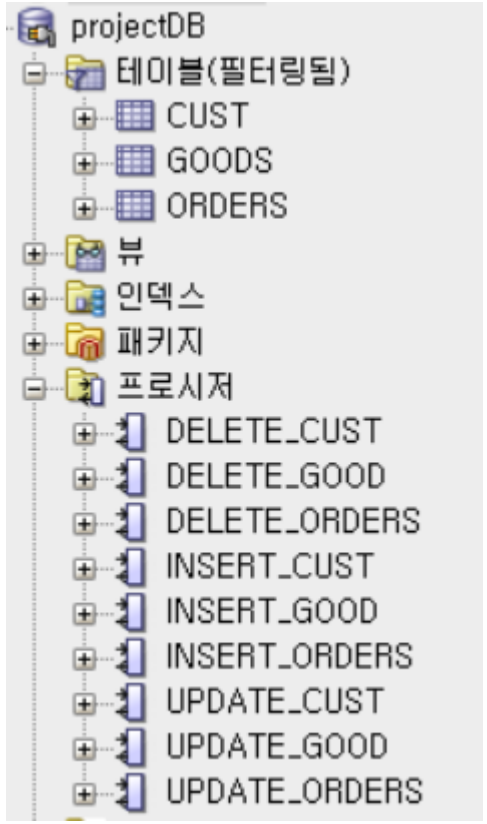
저장된 텍스트 파일들을 데이터 베이스로 전환, 클라이언트와 서버의 로그 저장

오라클 데이터 베이스 연동

- 데이터 베이스 관리 시스템 (DataBase Management System) : DBMS의 한 종류
- 방대한 양의 데이터를 편리하게 저장하고 효율적으로 관리 하고 검색 할 수 있는 환경을 제공하는 소프트웨어 시스템이라고 할 수 있다.
- 장점 : 작성과 이용이 비교적 쉽고 확장이 용이하다.
- 관계형 데이터 베이스 정보를 테이블 형태로 저장한다.
- SQL : Structured Query Language
  - 데이터 베이스에 저장된 데이터를 조회, 입력, 수정, 삭제하는 조작이나, 테이블을 비롯한 다양한 객체(시퀀스, 프로시져 등)를 생성 및 제어하는 역할

오라클 데이터 베이스 연동

SQL Developer를 활용



	C_ID	C_NAME	C_PHONE	C_EMAIL
1	1	정재현	010-2464-2739	0306jh@naver.com
2	2	정다현	010-9156-2739	not-found
3	3	정해영	010-5637-2739	not-found
4	4	정민희	010-3421-2739	not-found

	G_ID	G_NAME	G_COM	G_PRICE
1	1	오스템 임플란트	오스템 컴퍼니	10000
2	2	린치 임플란트	린치 컴퍼니	12000
3	3	스마트 임플란트	스마트 컴퍼니	9500

고객, 상품의 이름이 주문정보 테이블에 들어가는 관계를 형성

	O_ID	C_NAME	G_NAME	O_DATE	O_QUAN	O_ALLPRICE
1	1	정재현	오스템 임플란트	22/11/12	2	20000
2	2	정다현	린치 임플란트	22/11/12	1	12000
3	3	정해영	스마트 임플란트	22/11/14	1	9500

SQL Developer를 활용한 테이블과 프로시져 생성

데이터 베이스 테이블 설정

고객 데이터 테이블

```
CREATE TABLE "PROJECTDB"."CUST"
 (    "C_ID" NUMBER(10,0),
  "C_NAME" VARCHAR2(100 BYTE) NOT NULL ENABLE,
  "C_PHONE" VARCHAR2(100 BYTE),
  "C_EMAIL" VARCHAR2(100 BYTE),
   CONSTRAINT "CUST_PK" PRIMARY KEY ("C_NAME")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "SYSTEM"  ENABLE
 ) SEGMENT CREATION IMMEDIATE
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "SYSTEM" ;
```

"CUST" 라는 고객 테이블 이름 설정

고객의 아이디 (10자리 정수로 제한)

고객의 성함 (문자열 100Byte로 제한 NULL로 존재 해서는 안됨)

고객의 전화번호 (문자열 100Byte로 제한)

고객의 이메일 (문자열 100Byte로 제한)

고객 테이블의 primary key를 이름으로 설정

데이터 베이스 테이블 설정

상품 데이터 테이블

```
CREATE TABLE "PROJECTDB"."GOODS"
 (    "G_ID" NUMBER(10,0),
  "G_NAME" VARCHAR2(100 BYTE) NOT NULL ENABLE,
  "G_COM" VARCHAR2(100 BYTE),
  "G_PRICE" NUMBER(20,0),
   CONSTRAINT "GOODS_PK" PRIMARY KEY ("G_NAME")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "SYSTEM"  ENABLE
 ) SEGMENT CREATION IMMEDIATE
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "SYSTEM" ;
```

"GOODS" 라는 상품 테이블 이름 설정

상품의 아이디 (10자리 정수로 제한)

상품의 이름(문자열 100Byte로 제한 NULL로 존재 해서는 안됨)

상품의 회사이름(문자열 100Byte로 제한)

상품의 가격(20자리 정수로 제한)

상품 테이블의 primary key를 이름으로 설정

데이터 베이스 테이블 설정

주문정보 데이터 테이블

```
CREATE TABLE "PROJECTDB"."ORDERS"
 (    "O_ID" NUMBER(10,0) NOT NULL ENABLE,
  "C_NAME" VARCHAR2(100 BYTE) NOT NULL ENABLE,
  "G_NAME" VARCHAR2(100 BYTE) NOT NULL ENABLE,
  "O_DATE" DATE,
  "O_QUAN" NUMBER(10,0),
  "O_ALLPRICE" NUMBER(30,0),
   CONSTRAINT "ORDERS_PK" PRIMARY KEY ("O_ID")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "SYSTEM"  ENABLE,
   CONSTRAINT "ORDERS_FK1" FOREIGN KEY ("G_NAME")
    REFERENCES "PROJECTDB"."GOODS" ("G_NAME") ENABLE,
   CONSTRAINT "ORDERS_FK2" FOREIGN KEY ("C_NAME")
    REFERENCES "PROJECTDB"."CUST" ("C_NAME") ENABLE
 ) SEGMENT CREATION IMMEDIATE
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "SYSTEM" ;
```

"ORDERS" 라는 주문정보 테이블 이름 설정

주문정보의 아이디 (10자리 정수로 제한, NULL로 존재해서는 안됨)

주문정보에 추가될 고객의 성함 (문자열 100Byte로 제한 NULL로 존재 해서는 안됨)

주문정보에 추가될 상품의 이름 (문자열 100Byte로 제한 NULL로 존재 해서는 안됨)

주문정보의 주문날짜(날짜/시간 변수)

주문정보의 수량(10의 자리 정수형으로 제한)

주문한 제품에 수량에 따른 가격(해당 주문 가격 x 수량, 30자리 정수로 제한)

주문정보의 테이블의 primary key를 아이디로 설정

데이터 베이스 프로시져 설정

고객 데이터 입력 프로시져

```
create or replace PROCEDURE Insert_Cust
(v_cusid IN CUST.C_ID%TYPE,
v_cusname IN CUST.C_NAME%TYPE,
v_cusphone IN CUST.C_PHONE%TYPE,
v_cusemail IN CUST.C_EMAIL%TYPE)
IS
BEGIN
    DBMS_OUTPUT.ENABLE;

    INSERT INTO CUST
    (C_ID, C_NAME, C_PHONE, C_EMAIL)
    VALUES(v_cusid, v_cusname, v_cusphone, v_cusemail);
    COMMIT;
    DBMS_OUTPUT.PUT_LINE('CUST NO : '||v_cusid);
    DBMS_OUTPUT.PUT_LINE('CUST NAME : '||v_cusname);
    DBMS_OUTPUT.PUT_LINE('CUST PHONE : '||v_cusphone);
    DBMS_OUTPUT.PUT_LINE('CUST EMAIL : '||v_cusemail);
END;
```

고객 데이터 수정 프로시져

```
create or replace PROCEDURE Update_Cust
(v_cusid IN CUST.C_ID%TYPE,              --고객의 정보를 고객의 아이디
v_cusname IN CUST.C_NAME%TYPE,           --수정할 고객의 이름
v_cusphone IN CUST.C_PHONE%TYPE,         --수정할 고객의 전화번호
v_cusemail IN CUST.C_EMAIL%TYPE)         --수정할 고객의 이메일
IS
    --수정 데이터를 확인하기 위한 변수 선언
    v_cus CUST%ROWTYPE;
BEGIN DBMS_OUTPUT.ENABLE;
    --이름 수정
    UPDATE CUST
    SET c_name = v_cusname,
    c_phone = v_cusphone,
    c_email = v_cusemail
    WHERE c_id = v_cusid;

    COMMIT;

    DBMS_OUTPUT.PUT_LINE('Data Update Success');

    --수정할 데이터 확인하기 위해 검색
    SELECT c_id, c_name, c_phone, c_email
    INTO v_cus.c_id, v_cus.c_name, v_cus.c_phone, v_cus.c_email
    FROM CUST
    WHERE c_id = v_cusid;
    DBMS_OUTPUT.PUT_LINE('**** Confirm Update Data ****');
    DBMS_OUTPUT.PUT_LINE('CUS ID : '||v_cus.c_id);
    DBMS_OUTPUT.PUT_LINE('CUS NAME : '||v_cus.c_name);
    DBMS_OUTPUT.PUT_LINE('CUS PHONE : '||v_cus.c_phone);
    DBMS_OUTPUT.PUT_LINE('CUS EMAIL : '||v_cus.c_email);
END;
```

고객 데이터 삭제 프로시져

```
create or replace PROCEDURE Delete_Cust
(v_cusid IN CUST.c_id%TYPE)
IS
    -- 삭제 데이터를 확인하는 레코드 선언
    TYPE del_record IS RECORD
    (v_cusid CUST.c_id%TYPE,
    v_cusname CUST.c_name%TYPE,
    v_cusphone CUST.c_phone%TYPE,
    v_cusemail CUST.c_email%TYPE);

    v_cus del_record;
BEGIN
    DBMS_OUTPUT.ENABLE;
    --삭제된 데이터 확인용 쿼리
    SELECT c_id, c_name, c_phone, c_email
    INTO v_cus.v_cusid, v_cus.v_cusname,
    v_cus.v_cusphone, v_cus.v_cusemail
    FROM CUST
    WHERE c_id = v_cusid;

    DBMS_OUTPUT.PUT_LINE('CUS NO : '||v_cus.v_cusid);
    DBMS_OUTPUT.PUT_LINE('CUS NAME : '||v_cus.v_cusname);
    DBMS_OUTPUT.PUT_LINE('CUS PHONE : '||v_cus.v_cusphone);
    DBMS_OUTPUT.PUT_LINE('CUS EMAIL : '||v_cus.v_cusemail);

    --삭제 쿼리
    DELETE
    FROM CUST
    where c_id = v_cusid;

    COMMIT;

    DBMS_OUTPUT.PUT_LINE('Data Delete Success');
END;
```

데이터 베이스 프로시져 설정

상품 데이터 입력 프로시져

```
create or replace PROCEDURE Insert_Good
(v_gooid IN GOODS.G_ID%TYPE,          --추가할 상품의 아이디
v_gooname IN GOODS.G_NAME%TYPE,       --추가할 상품의 이름
v_goocom IN GOODS.G_COM%TYPE,     --추가할 상품의 전화번호
v_gooprice IN GOODS.G_PRICE%TYPE)     --추가할 상품의 가격
IS
BEGIN
    DBMS_OUTPUT.ENABLE;

    INSERT INTO GOODS
    (G_ID, G_NAME, G_COM, G_PRICE)
    VALUES(v_gooid, v_gooname, v_goocom, v_gooprice);
    COMMIT;
    DBMS_OUTPUT.PUT_LINE('CUST NO : '||v_gooid);
    DBMS_OUTPUT.PUT_LINE('CUST NAME : '||v_gooname);
    DBMS_OUTPUT.PUT_LINE('CUST PHONE : '||v_goocom);
    DBMS_OUTPUT.PUT_LINE('CUST EMAIL : '||v_gooprice);
END;
```

상품 데이터 수정 프로시져

```
create or replace PROCEDURE Update_Good
(v_gooid IN GOODS.G_ID%TYPE,              --상품의 정보를 고객의 아이디
v_gooname IN GOODS.G_NAME%TYPE,           --수정할 상품의 이름
v_goocom IN GOODS.G_COM%TYPE,         --수정할 상품의 회사
v_gooprice IN GOODS.G_PRICE%TYPE)         --수정할 상품의 가격
IS
    --수정 데이터를 확인하기 위한 변수 선언
    v_goo GOODS%ROWTYPE;
BEGIN DBMS_OUTPUT.ENABLE;
    --이름, 회사, 가격
    UPDATE GOODS
    SET g_name = v_gooname,
    g_com = v_goocom,
    g_price = v_gooprice
    WHERE g_id = v_gooid;

    COMMIT;

    DBMS_OUTPUT.PUT_LINE('Data Update Success');

    --수정할 데이터 확인하기 위해 검색
    SELECT g_id, g_name, g_com, g_price
    INTO v_goo.g_id, v_goo.g_name, v_goo.g_com, v_goo.g_price
    FROM GOODS
    WHERE g_id = v_gooid;
    DBMS_OUTPUT.PUT_LINE('**** Confirm Update Data ****');
    DBMS_OUTPUT.PUT_LINE('GOO ID : '||v_goo.g_id);
    DBMS_OUTPUT.PUT_LINE('GOO NAME : '||v_goo.g_name);
    DBMS_OUTPUT.PUT_LINE('GOO COMPANY : '||v_goo.g_com);
    DBMS_OUTPUT.PUT_LINE('GOO PRICE : '||v_goo.g_price);
END;
```

상품 데이터 삭제 프로시져

```
create or replace PROCEDURE Delete_Good
(v_gooid IN GOODS.g_id%TYPE)
IS
    -- 삭제 데이터를 확인하는 레코드 선언
    TYPE del_record IS RECORD
    (v_gooid GOODS.g_id%TYPE,
    v_gooname GOODS.g_name%TYPE,
    v_goocom GOODS.g_com%TYPE,
    v_gooprice GOODS.g_price%TYPE);

    v_goo del_record;
BEGIN
    DBMS_OUTPUT.ENABLE;
    --삭제된 데이터 확인용 쿼리
    SELECT g_id, g_name, g_com, g_price
    INTO v_goo.v_gooid, v_goo.v_gooname,
    v_goo.v_goocom, v_goo.v_gooprice
    FROM GOODS
    WHERE g_id = v_gooid;

    DBMS_OUTPUT.PUT_LINE('GOO NO : '||v_goo.v_gooid);
    DBMS_OUTPUT.PUT_LINE('GOO NAME : '||v_goo.v_gooname);
    DBMS_OUTPUT.PUT_LINE('GOO COMPANY : '||v_goo.v_goocom);
    DBMS_OUTPUT.PUT_LINE('GOO PRICE : '||v_goo.v_gooprice);

    --삭제 쿼리
    DELETE
    FROM GOODS
    where g_id = v_gooid;

    COMMIT;

    DBMS_OUTPUT.PUT_LINE('Data Delete Success');
END;
```

데이터 베이스 프로시져 설정

주문정보 입력 프로시져

```
create or replace PROCEDURE Insert_Orders
(v_orderid IN ORDERS.O_ID%TYPE,
v_cusname IN CUST.C_NAME%TYPE,
v_gooname IN GOODS.G_NAME%TYPE,
v_orderquan IN ORDERS.O_QUAN%TYPE
/*, v_allprice IN GOODS.G_PRICE%TYPE*/)
IS

BEGIN
    DBMS_OUTPUT.ENABLE;

    INSERT INTO ORDERS
    (O_ID, C_NAME, G_NAME, O_DATE, O_QUAN/*, O_ALLPRICE*/)
    VALUES(v_orderid, v_cusname, v_gooname, sysdate,
    v_orderquan /*, v_allprice*/);

    UPDATE orders
    SET o_allprice = o_quan * (SELECT G.G_PRICE
                               FROM GOODS G
                               WHERE g.g_name = v_gooname)
    WHERE O_ID = v_orderid;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE('ORDER NO : '||v_orderid);
    DBMS_OUTPUT.PUT_LINE('ORDER CNAME : '||v_cusname);
    DBMS_OUTPUT.PUT_LINE('ORDER GNAME : '||v_gooname);
    DBMS_OUTPUT.PUT_LINE('ORDER QUANTITY : '||v_orderquan);
    --DBMS_OUTPUT.PUT_LINE('ORDER G_Price : '||v_allprice);
END;
```

주문정보 수정 프로시져

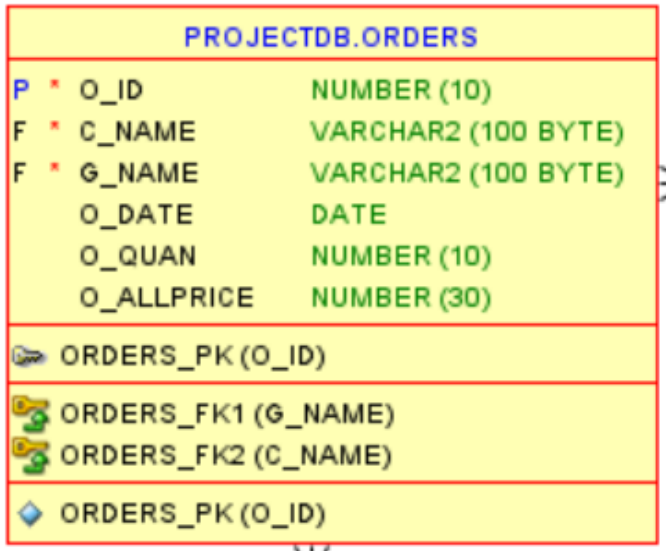
```
create or replace PROCEDURE UPDATE_ORDERS
(V_OID IN ORDERS.O_ID%TYPE,
V_CUSNAME IN ORDERS.C_NAME%TYPE,
V_GOONAME IN ORDERS.G_NAME%TYPE,
V_OQUAN IN ORDERS.O_QUAN%TYPE)
IS
    V_ORDER ORDERS%ROWTYPE;
BEGIN
    DBMS_OUTPUT.ENABLE;
    UPDATE ORDERS
    SET
    C_NAME = V_CUSNAME,
    G_NAME = V_GOONAME,
    O_QUAN = V_OQUAN,
    O_ALLPRICE = V_OQUAN * (SELECT G.G_PRICE
                            FROM GOODS G
                            WHERE G.G_NAME = V_GOONAME)
    WHERE O_ID = V_OID;

    COMMIT;

    DBMS_OUTPUT.PUT_LINE('DATA UPDATE SUCCESS');
END;
```

주문정보 삭제 프로시져

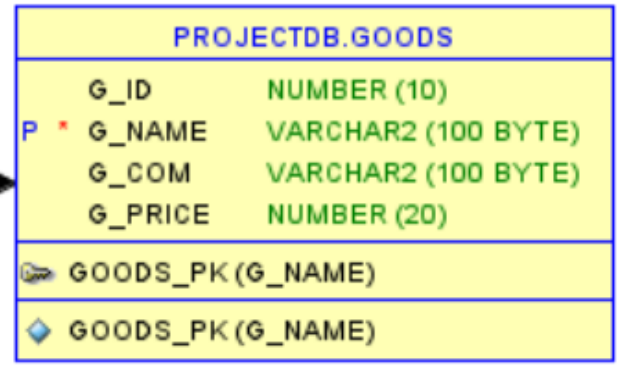
```
create or replace PROCEDURE DELETE_ORDERS
(P_ID IN ORDERS.O_ID%TYPE)
IS
    TYPE DEL_RECORD IS RECORD
    (V_ORDERID ORDERS.O_ID%TYPE,
    V_CUSNAME ORDERS.C_NAME%TYPE,
    V_GOONAME ORDERS.G_NAME%TYPE,
    V_DATE ORDERS.O_DATE%TYPE,
    V_ORDERQUAN ORDERS.O_QUAN%TYPE,
    V_OALLPRICE ORDERS.O_ALLPRICE%TYPE);

    V_ORDER DEL_RECORD;
BEGIN
    DBMS_OUTPUT.ENABLE;
    SELECT O_ID
    INTO V_ORDER.V_ORDERID
    FROM ORDERS
    WHERE O_ID = P_ID;

    DBMS_OUTPUT.PUT_LINE('ORDER ID : '||V_ORDER.V_ORDERID);

    DELETE
    FROM ORDERS
    WHERE O_ID = P_ID;

    COMMIT;

    DBMS_OUTPUT.PUT_LINE('DATA DELETE SUCCESS');
END;
```

DB 설계

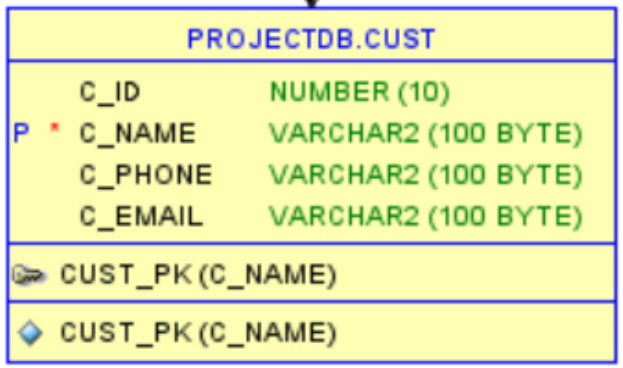
주문정보 데이터 베이스



상품정보 데이터 베이스



주문정보는 상품정보와 다 대 일 관계, 고객정보와 다 대 일 관계를 성립하게 해야 한다.



고객정보 데이터 베이스

데이터 베이스 설명

- 각 데이터 베이스 마다 primary key 보유
- 고객, 상품의 데이터 베이스에는 이름을 키 값으로 설정
- 구매정보 데이터 베이스는 아이디를 키, 고객과 상품의 이름을 외래키로 설정(이름을 외래키로 설정한 이유는 구매정보 UI의 입력 시 기준이 되는 데이터가 고객과 상품의 이름이다.)

외래키 설정을 구매정보에서 설정

데이터 베이스 내 변수 설명
NUMBER : 데이터 베이스 내의 정수형 변수를 설정
VARCHAR2 : 데이터 베이스 내의 문자형 변수를 설정
DATE : 데이터 베이스 내의 날짜형 변수 지정(yy-MM-dd 형태)

DB의 UI연계 방법

```
SELECT
    (SELECT VALUE FROM NLS_DATABASE_PARAMETERS WHERE PARAMETER='NLS_LANGUAGE')
     || '_' ||
     (SELECT VALUE FROM NLS_DATABASE_PARAMETERS WHERE PARAMETER='NLS_TERRITORY')
     || '.' ||
     (SELECT VALUE FROM NLS_DATABASE_PARAMETERS WHERE PARAMETER='NLS_CHARACTERSET')
FROM DUAL
```

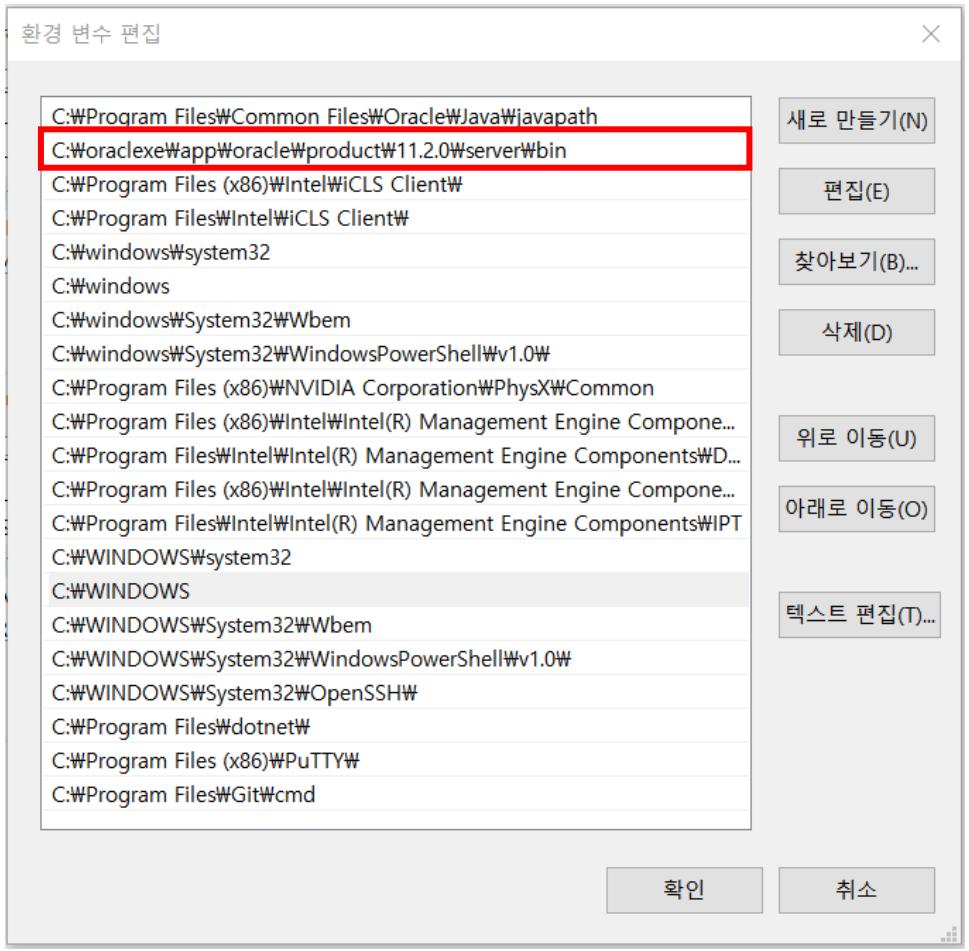
질의... ×

SQL | 인출된 모든 행: 1(0.013초)

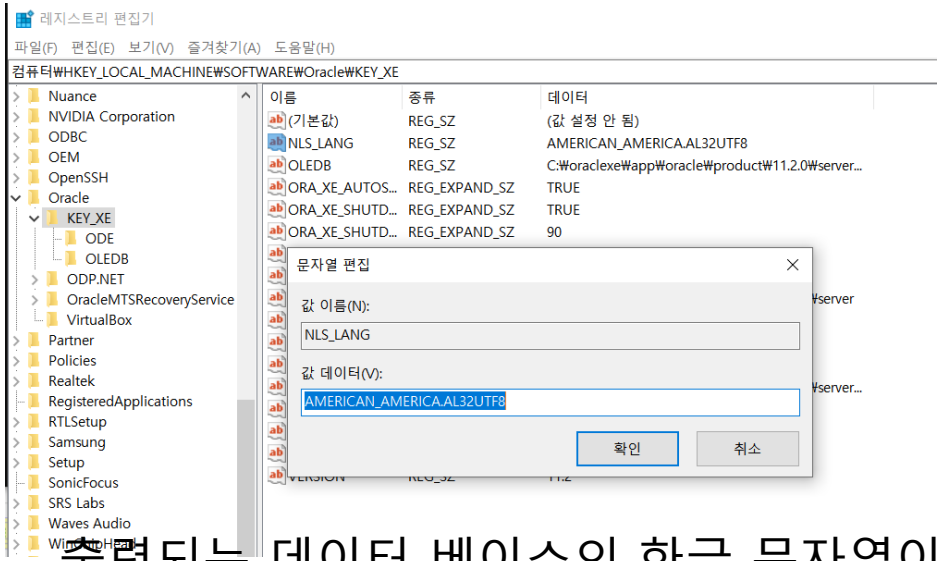
(SELECTVALUEFROMNLS_DATABASE_PARAMETERSWHEREPARAMETER='NLS_LANGUAGE')||'_'||(SEL

1 AMERICAN_AMERICA.AL32UTF8

오라클에서 사용하는 문자열 형식 추출

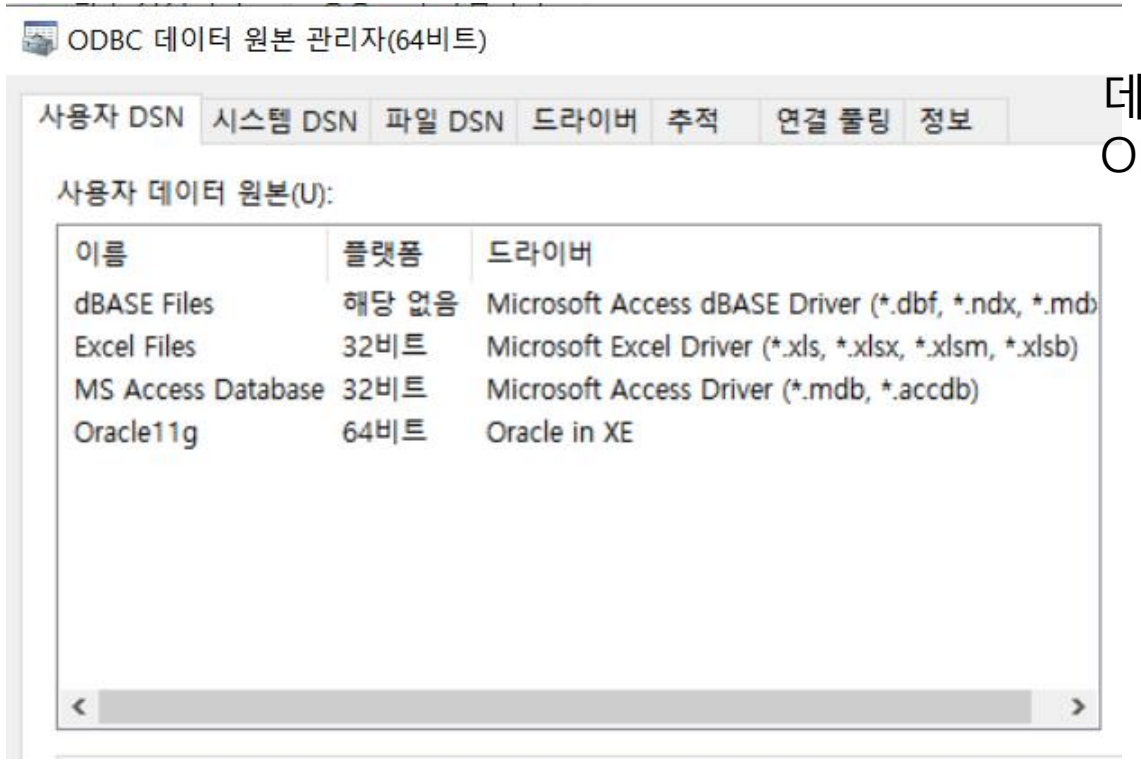


설치된 오라클 데이터베이스의 서버를 환경 변수 PATH에 경로 설정

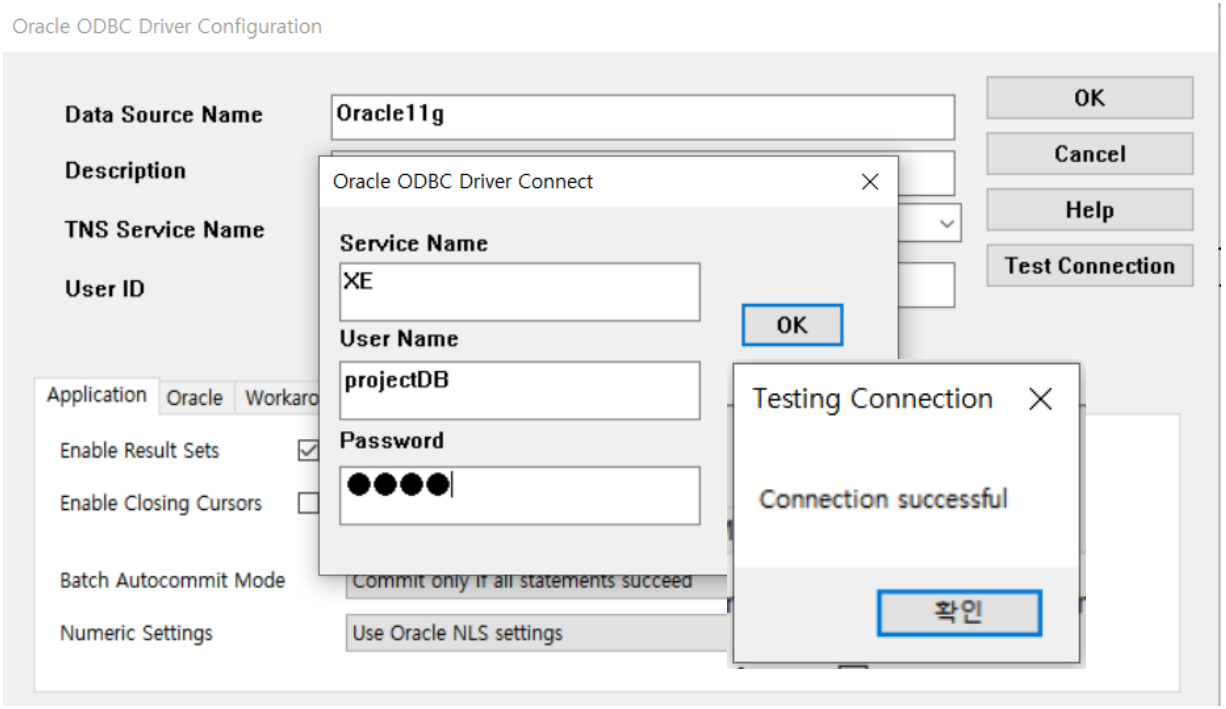


출력되는 데이터 베이스의 한글 문자열이 깨지지 않도록 레지스트리 편집기에서 문자열 편집

DB의 UI연계 방법



데이터 베이스를 연동하기 위한 관리도구 중
ODBC 데이터 원본 관리자를 열어서 사용자 계정을 생성



데이터원본 이름 생성, 데이터베이스 계정, 비밀번호 입력 후 테스트하여 연결 확인

DB의 UI연계 방법

```
/*데이터 베이스의 데이터들을 불러오기*/                    "shoppingConnection"
                                                        "productConnection"
QSqlDatabase db = QSqlDatabase::addDatabase("QODBC", "clientConnection");
/*추가하려는 데이터베이스 종류는 QODBC(Qt Oracle DataBase)*/
db.setDatabaseName("Oracle11g");                /*데이터베이스 이름*/
db.setUserName("projectDB");                    /*데이터 베이스 계정 명*/
db.setPassword("1234");                         /*데이터 베이스 비밀번호*/
```

```
ClientModel->setTable("CUST");
ProductModel->setTable("GOODS");
ShoppingModel->setTable("ORDERS");
```

각 매니저 클라이언트마다 데이터 베이스 추가 및 테이블 호출

디자인 패턴을 적용하여 메모리 관리부분은 DB로 유연하게 스위칭 할 수 있게 구조 변경

고객 관리 매니저

```
ClientModel->setTable("CUST");

clientquery->exec("SELECT * FROM CUST ORDER BY C_ID ASC");
ClientModel->select();

//SearchModel->setTable("SEARCH_CUST");

/*고객 데이터베이스 출력 쿼리문*/
ClientModel->setHeaderData(0, Qt::Horizontal, QObject::tr("c_id"));
ClientModel->setHeaderData(1, Qt::Horizontal, QObject::tr("c_name"));
ClientModel->setHeaderData(2, Qt::Horizontal, QObject::tr("c_phone"));
ClientModel->setHeaderData(3, Qt::Horizontal, QObject::tr("c_email"));


ui->tableView->setModel(ClientModel);
ui->tableView->resizeColumnsToContents();
```

INSERT_CUST

```
clientquery->exec(QString("CALL INSERT_CUST(%1, '%2', '%3', '%4')")
                  .arg(id).arg(name).arg(number).arg(address));
ClientModel->select();                          /*릴레이션 테이블 호출*/
```

UPDATE_CUST

```
clientquery->exec(QString("CALL UPDATE_CUST(%1, '%2', '%3', '%4')")
                  .arg(ID).arg(name).arg(phone).arg(address));

clientquery->exec(QString("select * from CUST ORDER BY C_ID"));
ClientModel->select();
```

DELETE_CUST

```
clientquery->exec(QString("CALL DELETE_CUST(%1)").arg(idx));    /*해당하는 인덱스 데이터를 지우는 프로시져 호출*/
ClientModel->select();      /*해당 테이블 호출*/
```

초기 테이블 호출 및 컬럼 헤드 구성
입력, 수정, 삭제 함수 별로 해당되는 프로시져 실행

디자인 패턴을 적용하여 메모리 관리부분은 DB로 유연하게 스위칭 할 수 있게 구조 변경

상품 관리 매니저

```
ProductModel->setTable("GOODS");                /*GOODS 테이블 탐색*/
ProductModel->select();

/*고객 데이터베이스 출력 쿼리문*/
ProductModel->setHeaderData(0, Qt::Horizontal, QObject::tr("g_id"));
ProductModel->setHeaderData(1, Qt::Horizontal, QObject::tr("g_name"));
ProductModel->setHeaderData(2, Qt::Horizontal, QObject::tr("g_com"));
ProductModel->setHeaderData(3, Qt::Horizontal, QObject::tr("g_price"));

ui->tableView->setModel(ProductModel);
ui->tableView->resizeColumnsToContents();       /*데이터 사이즈에 맞게 열을 정렬*/
```

```
/*오라클 내의 프로시져를 사용*/        INSERT_GOOD
ProductQuery->exec(QString("CALL INSERT_GOOD(%1, '%2', '%3', %4)")
                   .arg(id).arg(name).arg(company).arg(price));
ProductModel->select();                          /*릴레이션 테이블 호출*/

                                        UPDATE_GOOD
ProductQuery->exec(QString("CALL UPDATE_GOOD(%1, '%2', '%3', %4)")
                   .arg(ID).arg(name).arg(company).arg(price));

ProductModel->select();
```

```
                                    DELETE_GOOD
ProductQuery->exec(QString("CALL DELETE_GOOD(%1)").arg(index));    /*해당하는 인덱스 데이터를 지우는 프로시져 호출*/
ProductModel->select();      /*해당 테이블 호출*/
```

디자인 패턴을 적용하여 메모리 관리부분은 DB로 유연하게 스위칭 할 수 있게 구조 변경

구매정보 관리 매니저

```
ShoppingModel->setTable("ORDERS");                  /*테이블 중 ORDERS 테이블 사용*/
ShoppingModel->select();                            /*테이블 호출*/

ShoppingModel->setHeaderData(0, Qt::Horizontal, QObject::tr("o_id"));
ShoppingModel->setHeaderData(1, Qt::Horizontal, QObject::tr("c_name"));
ShoppingModel->setHeaderData(2, Qt::Horizontal, QObject::tr("g_name"));
ShoppingModel->setHeaderData(3, Qt::Horizontal, QObject::tr("o_date"));
ShoppingModel->setHeaderData(4, Qt::Horizontal, QObject::tr("o_quan"));
ShoppingModel->setHeaderData(5, Qt::Horizontal, QObject::tr("o_allprice"));

ui->tableView->setModel(ShoppingModel);
ui->tableView->resizeColumnsToContents();           /*데이터 사이즈에 맞게 열을 정렬*/
```

INSERT_ORDERS

```
ShoppingQuery->exec(QString("CALL INSERT_ORDERS(%1, '%2', '%3', %4)")
                    .arg(id).arg(client).arg(product).arg(quan));
ShoppingModel->select();                            /*릴레이션 테이블 호출*/
```

DELETE_ORDERS

```
ShoppingQuery->exec(QString("CALL DELETE_ORDERS(%1)").arg(index));
/*해당하는 인덱스 데이터를 지우는 프로시져*/
ShoppingModel->select();            /*해당 테이블 select 호출*/
```

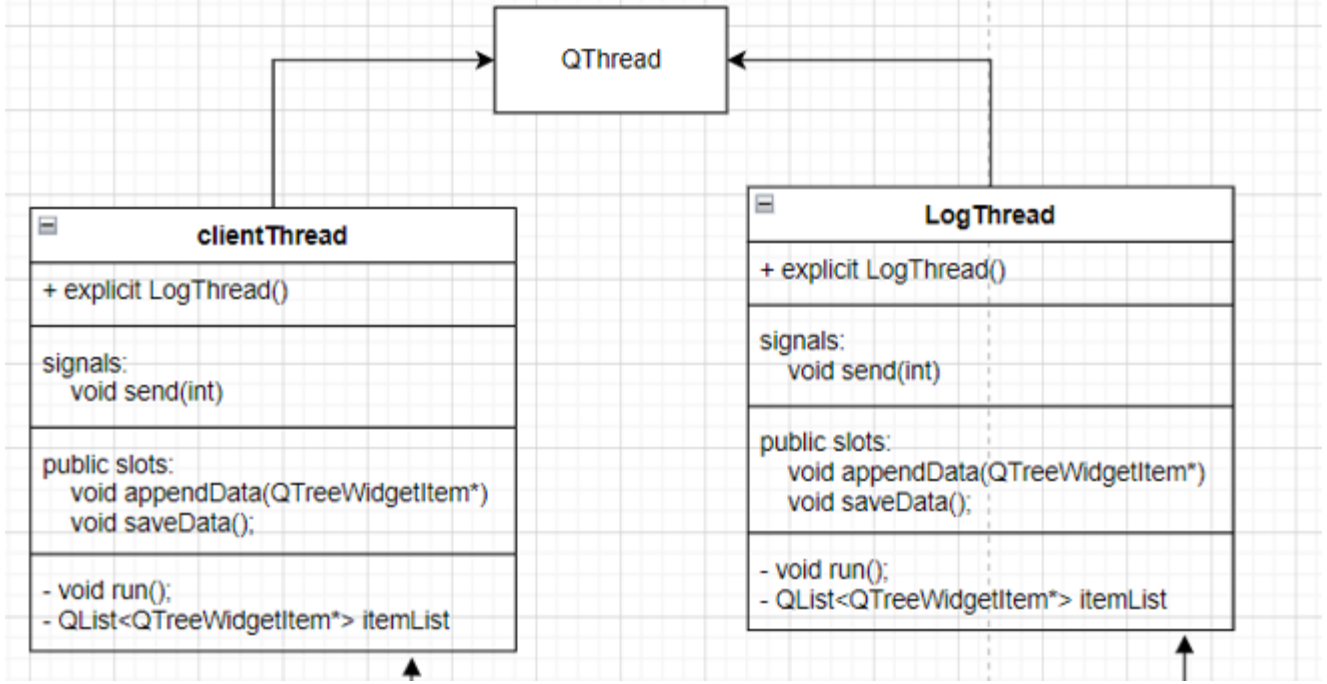
UPDATE_ORDERS

```
ShoppingQuery->exec(QString("CALL UPDATE_ORDERS(%1, '%2', '%3', %4)")
                    .arg(ID).arg(clientname).arg(productname).arg(quan));
/*구매 정보 업데이트 프로시져 호출*/
ShoppingModel->select();        /*테이블 호출*/
ui->tableView->update();        /*테이블 업데이트*/
ui->tableView->resizeColumnsToContents();           /*데이터 사이즈에 맞게 열을 정렬*/
```

생성자 소비자 패턴을 적용,
로그를 매체 별 저장할 수 있게 구조 변경

생성자 소비자 패턴 이란?
생산자(Producer) 소비자(Consumer) 패턴은 작업 목록을 가운데에 두고 '작업을 생산해내는 주체' '작업을 처리하는 주체' 를 분리시키는 설계 방법 '작업을 생성하는 부분(Producer)'과 '처리하는 부분(Consumer)'이 각각 감당 할 수 있는 부하를 조절 할 수 있다는 장점이 있다.

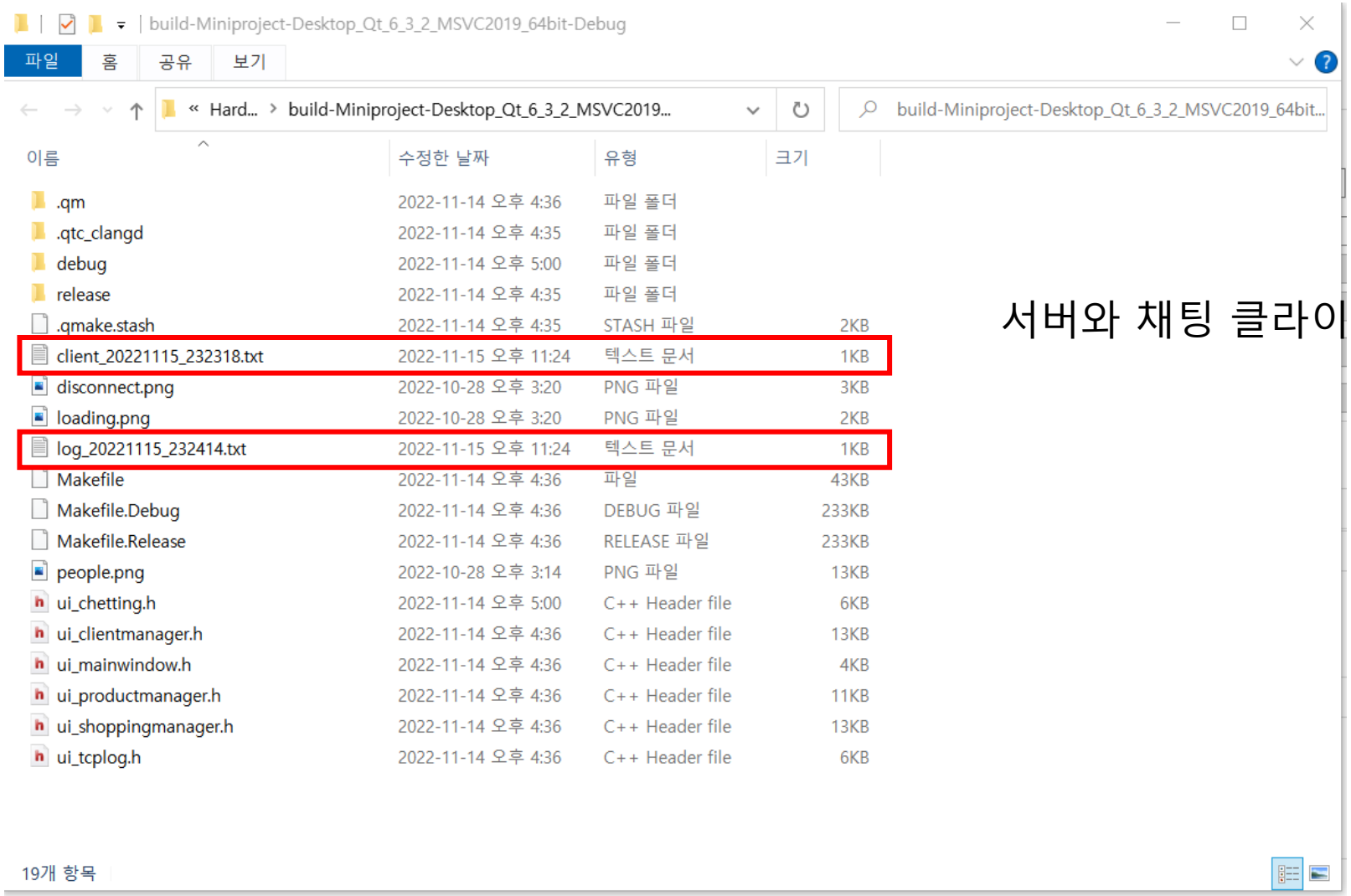
따라서 위의 주제대로 로그를 매체 별로 저장하는 것이 생성자, 소비자 패턴을 적용하는 것 이 된다.



멀티 스레드를 이용한 디자인 패턴

고객 채팅로그와 서버채팅 로그는 모두 별개로 저장

생성자 소비자 패턴을 적용,
로그를 매체 별 저장할 수 있게 구조 변경



서버와 채팅 클라이언트의 로그를 시간별로 저장