Anexo III. Creación procedimientos ABM (Alta, Baja, Modificación)

Procedimientos almacenados de inserción de datos

```
/*creacion de las store procedure para insertar datos */

/* creacion de un usuario para realizar las operaciones CRUD de las compañías */
--CREATE USER soluser IDENTIFIED BY ajcc
--DEFAULT TABLESPACE users
--QUOTA UNLIMITED ON users
--TEMPORARY TABLESPACE temp;


--GRANT CREATE SESSION,RESOURCE TO soluser;
```

El procedimiento INS_LUZLOG registra la ejecución de los procedimientos almacenados en la tabla LOG denominada LUZLOG.

```
/*********************************************************************************************
*******************************/
CREATE OR REPLACE PROCEDURE Ins_LuzLog(
/*********************************************
| Insert data about store procedure executed
|
| In Params:
|        - Store Procedure Name
|        - In Params
|        - Out Params
| Out Params:  --
| Date:
| Author: AJCC
|*********************************************
*/
    pSPName     IN  LuzLog.SPName%TYPE
  , pInParams   IN  LuzLog.inParams%TYPE
  , poutParams  IN  LuzLog.outParams%TYPE
  , RSP         OUT VARCHAR2
)
IS
  lInParams VARCHAR2(2000);
  lOutParams VARCHAR2(2000);
BEGIN
    select q'['|| pInParams  ||']' into lInParams from dual;
    select q'['|| poutParams ||']' into loutParams from dual;

    INSERT INTO LuzLog(
      execDate, SPName
      , inParams , outParams
      , userExec
    )
    VALUES (
      current_timestamp, pSpName
      , pInParams, poutParams
      , (SELECT SYS_CONTEXT ('USERENV', 'SESSION_USER') FROM DUAL)
    );
    RSP:='OK';

    DBMS_OUTPUT.PUT_LINE ($$PLSQL_UNIT ||':'|| RSP);
EXCEPTION

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    DBMS_OUTPUT.PUT_LINE ($$PLSQL_UNIT ||':'|| RSP);

END Ins_LuzLog;
```

El procedimiento INS_COUNTRY introduce valores en la tabla COUNTRY.

```
/*******************************************************************************************
*****************************/
create or replace PROCEDURE Ins_Country (
/*********************************************
|  Insert Country data in Country table
|
| In Params:   CountryCode, CountryName
| Out Params:  RSP
| Date:
| Programmer: AJCC
|*********************************************
*/
  pCountryCode   Country.CountryCode%TYPE
  , pCountryName Country.CountryName%TYPE
  , RSP          OUT VARCHAR2
)
AS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
BEGIN
  /* Validation Controls */

  --Check null values
  IF pCountryCode IS NULL THEN
    lErrorDescrip:=q'[CountryCode can't be null]';
    RAISE lException;
  END IF;

  IF pCountryName IS NULL THEN
    lErrorDescrip:=q'[CountryName can't be null]';
    RAISE lException;
  END IF;


  --Check datatypes: Number values, quotes in strings, lenght...
  IF LENGTH(pCountryName)>100 THEN
    lErrorDescrip:='CountryName too long, >100';
    RAISE lException;
  END IF;

  --Insert the Country
  INSERT INTO COUNTRY (COUNTRYCODE, COUNTRYNAME) VALUES (pCountryCode, pCountryName);
  RSP:='OK';

  --Track Store Procedure Result
  Ins_LuzLog($$PLSQL_UNIT, to_char(pCountryCode) || q'[, ']' || pCountryName ||  q']',
RSP, lRSP);


  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
    Ins_LuzLog($$PLSQL_UNIT, to_char(pCountryCode) || q'[, ']' || pCountryName ||  q']',
RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    Ins_LuzLog($$PLSQL_UNIT, to_char(pCountryCode) || q'[, ']' || pCountryName ||  q']',
RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento INS_PROVINCE introduce valores en la tabla PROVINCE.

```
/*****************************************************************************************
****************************/
CREATE OR REPLACE PROCEDURE Ins_Province (
/*********************************************
|  Insert Country data in Province table
|
| In Params:   provinceCode, provinceName, CountryCode
| Out Params:  RSP
| Date:
| Programmer: AJCC
|*********************************************
*/
    pProvinceCode IN  Province.provinceCode%TYPE
  , pProvinceName IN  Province.provinceName%TYPE
  , pCountryCode  IN  Province.CountryCode%TYPE
  , RSP           OUT VARCHAR2
  , pProvinceId   OUT Province.provinceId%TYPE
)
AS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
BEGIN
   /* Validation Controls */


  --Check null values
  IF pProvinceCode IS NULL THEN
    lErrorDescrip:=q'[ProvinceCode can't be null]';
    RAISE lException;
  END IF;

  IF pCountryCode IS NULL THEN
    lErrorDescrip:=q'[CountryCode can't be null]';
    RAISE lException;
  END IF;

  IF pProvinceName IS NULL THEN
    lErrorDescrip:=q'[ProvinceName can't be null]';
    RAISE lException;
  END IF;

  --Check datatypes: Number values, quotes in strings, lenght..
  IF LENGTH(pProvinceCode)>10 THEN
    lErrorDescrip:='ProvinceCode too long, >10';
    RAISE lException;
  END IF;

  IF LENGTH(pProvinceName)>100 THEN
    lErrorDescrip:='ProvinceName too long, >100';
    RAISE lException;
  END IF;

  -- Foreign Keys validation
  -- Then system can do itself

  --Insert the Province
  INSERT INTO PROVINCE (ProvinceCode, ProvinceName, CountryCode) VALUES (pProvinceCode,
pProvinceName, pCountryCode) RETURNING provinceId INTO pProvinceId;
  RSP:='OK' || ',' || to_char(pProvinceId);

  --Track Store Procedure Result
  Ins_LuzLog($$PLSQL_UNIT,(q'[']' || (pProvinceCode) || q'[', ']' || pProvinceName ||
q'[', ']' || to_char(pCountryCode)), RSP, lRSP);


  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
```

```
    Ins_LuzLog($$PLSQL_UNIT,(q'[']' || (pProvinceCode) || q'[', ']' || pProvinceName ||
q'[', ']' || to_char(pCountryCode)), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    Ins_LuzLog($$PLSQL_UNIT,(q'[']' || (pProvinceCode) || q'[', ']' || pProvinceName ||
q'[', ']' || to_char(pCountryCode)), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento INS_CITY introduce valores en la tabla CITY.

```
/*****************************************************************************************
************************************/
CREATE OR REPLACE PROCEDURE Ins_City (
/*********************************************
|  Insert City data in City table
|
| In Params:   cityCode, cityName, zipCode, provinceCode
| Out Params:  RSP
| Date:
| Programmer: AJCC
|*********************************************
*/
    pCityCode     IN  City.cityCode%TYPE
  , pCityName     IN  City.cityName%TYPE
  , pZipCode      IN  City.zipCode%TYPE
  , pProvinceCode IN  City.provinceCode%TYPE
  , RSP           OUT VARCHAR2
)
AS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
BEGIN

  /* Validation Controls */


  --Check null values
  IF pCityCode IS NULL THEN
    lErrorDescrip:=q'[CityCode can't be null]';
    RAISE lException;
  END IF;

  IF pCityName IS NULL THEN
    lErrorDescrip:=q'[CityName can't be null]';
    RAISE lException;
  END IF;

  IF pZipCode IS NULL THEN
    lErrorDescrip:=q'[ZipCode can't be null]';
    RAISE lException;
  END IF;

  IF pProvinceCode IS NULL THEN
    lErrorDescrip:=q'[ProvinceCode can't be null]';
    RAISE lException;
  END IF;


  --Check datatypes: Number values, quotes in strings, lenght...
  IF LENGTH(pCityName)>100 THEN
    lErrorDescrip:='CityName too long, >100';
    RAISE lException;
  END IF;

  IF LENGTH(pZipCode)>15 THEN
    lErrorDescrip:='ZipCode too long, >15';
    RAISE lException;
  END IF;
```

```
  -- Foreign Keys validation
  -- Then system can do itself

  --Insert the City
  INSERT INTO CITY (CITYCODE, CITYNAME, ZIPCODE, PROVINCECODE) VALUES (pCityCode,
pCityName, pZipCode, pProvinceCode);
  RSP:='OK';

  --Track Store Procedure Result
  Ins_LuzLog($$PLSQL_UNIT,(to_char(pCityCode) || q'[, ']' || pCityName || q'[', ']' ||
pZipCode || q'[', ]' || to_char(pProvinceCode)), RSP, lRSP);


  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
    Ins_LuzLog($$PLSQL_UNIT,(to_char(pCityCode) || q'[, ']' || pCityName || q'[', ']' ||
pZipCode || q'[', ]' || to_char(pProvinceCode)), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    Ins_LuzLog($$PLSQL_UNIT,(to_char(pCityCode) || q'[, ']' || pCityName || q'[', ']' ||
pZipCode || q'[', ]' || to_char(pProvinceCode)), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento INS_StreetType introduce valores en la tabla StreetType.

```
/*****************************************************************************************
**************************************/
CREATE OR REPLACE PROCEDURE Ins_StreetType (
/*********************************************
|  Insert Street types data in StreetType table
|
| In Params:   streetTypeCode, streetTypeName
| Out Params:  RSP
| Date:
| Programmer: AJCC
|*********************************************
*/
    pStreetTypeCode IN  StreetType.streetTypeCode%TYPE
  , pStreetTypeName IN  StreetType.streetTypeName%TYPE
  , RSP             OUT VARCHAR2
)
AS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
BEGIN

  /* Validation Controls */

  --Check null values
  IF pStreetTypeCode IS NULL THEN
    lErrorDescrip:=q'[StreetTypeCode can't be null]';
    RAISE lException;
  END IF;

  IF pStreetTypeName IS NULL THEN
    lErrorDescrip:=q'[StreetTypeName can't be null]';
    RAISE lException;
  END IF;

  --Check datatypes: Number values, quotes in strings, lenght...
  IF LENGTH(pStreetTypeName)>2 THEN
```

```
    lErrorDescrip:='StreetTypeName too long, >2';
    RAISE lException;
  END IF;


  --Insert the StreetType
  INSERT INTO StreetType (STREETTYPECODE, STREETTYPENAME) VALUES (pStreetTypeCode,
pStreetTypeName);
  RSP:='OK';

  --Track Store Procedure Result
  Ins_LuzLog($$PLSQL_UNIT,(to_char(pStreetTypeCode) || q'[, ']' || pStreetTypeName ||
q'[']'), RSP, lRSP);


  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
    Ins_LuzLog($$PLSQL_UNIT,(to_char(pStreetTypeCode) || q'[, ']' || pStreetTypeName ||
q'[']'), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    Ins_LuzLog($$PLSQL_UNIT,(to_char(pStreetTypeCode) || q'[, ']' || pStreetTypeName ||
q'[']'), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento INS_ADDRESS introduce valores en la tabla ADDRESS.

```
/*****************************************************************************************
************************************/
CREATE OR REPLACE PROCEDURE Ins_Address (
/*********************************************
|  Insert City data in City table
|
| In Params:    streetCode, streetName, adddressNumber, floor, doorNumber, phoneNumber,
cityCode
| Out Params:  RSP, addressCode
| Date:
| Programmer: AJCC
|*********************************************
*/
    pStreetCode     IN  Address.streetCode%TYPE
  , pStreetName     IN  Address.streetName%TYPE
  , pAdddressNumber IN  Address.adddressNumber%TYPE
  , pFloor          IN  Address.floor%TYPE
  , pDoorNumber     IN  Address.doorNumber%TYPE
  , pPhoneNumber    IN  Address.phoneNumber%TYPE
  , pCityCode       IN  Address.cityCode%TYPE
  , RSP             OUT VARCHAR2
  , pAddressCode    OUT Address.addressCode%TYPE
)
AS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
BEGIN

  /* Validation Controls */


  --Check null values
  IF pStreetCode IS NULL THEN
    lErrorDescrip:=q'[StreetCode can't be null]';
    RAISE lException;
  END IF;
```

```
  IF pStreetName IS NULL THEN
    lErrorDescrip:=q'[StreetName can't be null]';
    RAISE lException;
  END IF;

  IF pAdddressNumber IS NULL THEN
    lErrorDescrip:=q'[AdddressNumber can't be null]';
    RAISE lException;
  END IF;

  IF pCityCode IS NULL THEN
    lErrorDescrip:=q'[CityCode can't be null]';
    RAISE lException;
  END IF;


  --Check datatypes: Number values, quotes in strings, lenght...
  IF LENGTH(pStreetName)>200 THEN
    lErrorDescrip:='StreetName too long, >200';
    RAISE lException;
  END IF;

  IF LENGTH(pAdddressNumber)>10 THEN
    lErrorDescrip:='AdddressNumber too long, >10';
    RAISE lException;
  END IF;

  IF LENGTH(pFloor)>10 THEN
    lErrorDescrip:='Floor too long, >10';
    RAISE lException;
  END IF;

  IF LENGTH(pDoorNumber)>10 THEN
    lErrorDescrip:='DoorNumber too long, >10';
    RAISE lException;
  END IF;

  IF LENGTH(pPhoneNumber)>12 THEN
    lErrorDescrip:='PhoneNumber too long, >12';
    RAISE lException;
  END IF;


  -- Foreign Keys validation -->The system can do itself


  --Insert the Address
  INSERT INTO Address (
    STREETCODE, STREETNAME, adddressNumber, floor, DOORNUMBER, PHONENUMBER, CITYCODE)
  VALUES (
    pStreetCode,  pStreetName,  pAdddressNumber,  pFloor,  pDoorNumber,  pPhoneNumber,
pCityCode) RETURNING ADDRESSCODE INTO pAddressCode;

  RSP:='OK' || ',' || to_char(pAddressCode);

  --Track Store Procedure Result
  Ins_LuzLog(
      $$PLSQL_UNIT
      ,( to_char(pStreetCode) || q'[, ']' || pStreetName || q'[', ']' || pAdddressNumber
|| q'[', ']' || pFloor || q'[', ']' || pDoorNumber || q'[', ']' || pPhoneNumber || q'[',
]' || TO_CHAR(pCityCode))
      , RSP, lRSP);

  DBMS_OUTPUT.PUT_LINE ('AddressCode:' || to_char(pAddressCode));
  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
   Ins_LuzLog(
      $$PLSQL_UNIT
      ,( to_char(pStreetCode) || q'[, ']' || pStreetName || q'[', ']' || pAdddressNumber
|| q'[', ']' || pFloor || q'[', ']' || pDoorNumber || q'[', ']' || pPhoneNumber || q'[',
]' || TO_CHAR(pCityCode))
      , RSP, lRSP);
```

```
    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
   Ins_LuzLog(
      $$PLSQL_UNIT
      ,( to_char(pStreetCode) || q'[, ']' || pStreetName || q'[', ']' || pAdddressNumber
|| q'[', ']' || pFloor || q'[', ']' || pDoorNumber || q'[', ']' || pPhoneNumber || q'[',
]' || TO_CHAR(pCityCode))
      , RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento INS_BANK introduce valores en la tabla BANK.

```
/*****************************************************************************************
**************************************/
CREATE OR REPLACE PROCEDURE Ins_Bank (
/*********************************************
|  Insert Bank data in Bank table
|
| In Params:   bankCode, bankName
| Out Params:  RSP
| Date:
| Programmer: AJCC
|*********************************************
*/
    pBankCode     IN  Bank.bankCode%TYPE
  , pBankName     IN  Bank.bankName%TYPE
  , RSP           OUT VARCHAR2
)
IS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
BEGIN

  /* Validation Controls */

  --Check null values
  IF pBankCode IS NULL THEN
    lErrorDescrip:=q'[BankCode can't be null]';
    RAISE lException;
  END IF;

  IF pBankName IS NULL THEN
    lErrorDescrip:=q'[BankName can't be null]';
    RAISE lException;
  END IF;

  --Check datatypes: Number values, quotes in strings, lenght...
  IF LENGTH(pBankCode)>11 THEN
    lErrorDescrip:='BankCode too long, >11';
    RAISE lException;
  END IF;

  IF LENGTH(pBankName)>100 THEN
    lErrorDescrip:='BankName too long, >100';
    RAISE lException;
  END IF;


  --Insert the Bank
  INSERT INTO BANK (BANKCODE, BANKNAME) VALUES (pBankCode, pBankName);
  RSP:='OK';

  --Track Store Procedure Result
  Ins_LuzLog($$PLSQL_UNIT,( q'[']' || to_char(pBankCode) ||  q'[', ']' || pBankName ||
q'[']'), RSP, lRSP);


  DBMS_OUTPUT.PUT_LINE (RSP);
```

```
EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
     Ins_LuzLog($$PLSQL_UNIT,( q'[']' || to_char(pBankCode) ||  q'[', ']' || pBankName ||
q'[']'), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
     Ins_LuzLog($$PLSQL_UNIT,( q'[']' || to_char(pBankCode) ||  q'[', ']' || pBankName ||
q'[']'), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento INS_IDENTITYTYPE introduce valores en la tabla IDENTITYTYPE.

```
/*****************************************************************************************
************************************/
CREATE OR REPLACE PROCEDURE Ins_IdentityType (
/*********************************************
|  Insert IdentityType data in IdentityType table
|
| In Params:   identityCode, identityType
| Out Params:  RSP
| Date:
| Programmer: AJCC
|*********************************************
*/
    pIdentityCode IN  IdentityType.IdentityType%TYPE
  , pIdentityType IN  IdentityType.IdentityType%TYPE
  , RSP           OUT VARCHAR2
)
AS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
BEGIN

  /* Validation Controls */

  --Check null values
  IF pIdentityCode IS NULL THEN
    lErrorDescrip:=q'[IdentityCode can't be null]';
    RAISE lException;
  END IF;

  IF pIdentityType IS NULL THEN
    lErrorDescrip:=q'[IdentityType can't be null]';
    RAISE lException;
  END IF;

  --Check datatypes: Number values, quotes in strings, lenght...
  IF LENGTH(pIdentityType)>10 THEN
    lErrorDescrip:='IdentityType too long, >10';
    RAISE lException;
  END IF;


  --Insert the IdentityType
  INSERT   INTO   IdentityType   (IdentityCode,   IdentityType)   VALUES   (pIdentityCode,
pIdentityType);
  RSP:='OK';

  --Track Store Procedure Result
  Ins_LuzLog($$PLSQL_UNIT,(to_char(pIdentityCode) || q'[, ']' || pIdentityType || q'[']'),
RSP, lRSP);


  DBMS_OUTPUT.PUT_LINE (RSP);
```

```
EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
    Ins_LuzLog($$PLSQL_UNIT,(to_char(pIdentityCode)  ||  q'[,  ']'  ||  pIdentityType  ||
q'[']'), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    Ins_LuzLog($$PLSQL_UNIT,(to_char(pIdentityCode)  ||  q'[,  ']'  ||  pIdentityType  ||
q'[']'), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento INS_CONSUMER introduce valores en la tabla CONSUMER.

```
/*****************************************************************************************
************************************/
CREATE OR REPLACE PROCEDURE Ins_Consumer (
/***********************************************
|  Insert City data in City table
|
| In Params:   consumerName, consumerSurname, sex, mobilePhone, addressCode, identityCode,
identificationNumber, bankCode, accountCode
| Out Params:  RSP, consumerCode
| Date:
| Programmer: AJCC
|***********************************************
*/
    pConsumerName          IN  Consumer.consumerName%TYPE
  , pConsumerSurname       IN  Consumer.consumerSurname%TYPE
  , pSex                   IN  Consumer.sex%TYPE
  , pMobilePhone           IN  Consumer.mobilePhone%TYPE
  , pAddressCode           IN  Consumer.addressCode%TYPE
  , pIdentityCode          IN  Consumer.identityCode%TYPE
  , pIdentificationNumber IN  Consumer.identificationNumber%TYPE
  , pBankCode              IN  Consumer.bankCode%TYPE
  , pAccountCode           IN  Consumer.accountCode%TYPE
  , RSP                    OUT VARCHAR2
  , pConsumerCode          OUT Consumer.consumerCode%TYPE
)
AS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
  lInParam      VARCHAR2(2000);
BEGIN

  /* Validation Controls */


  --Check null values
  IF pConsumerName IS NULL THEN
    lErrorDescrip:=q'[ConsumerName can't be null]';
    RAISE lException;
  END IF;

  IF pConsumerSurname IS NULL THEN
    lErrorDescrip:=q'[ConsumerSurname can't be null]';
    RAISE lException;
  END IF;

  IF pSex IS NULL THEN
    lErrorDescrip:=q'[Sex can't be null]';
    RAISE lException;
  END IF;

  IF pAddressCode IS NULL THEN
    lErrorDescrip:=q'[AddressCode can't be null]';
```

```
    RAISE lException;
  END IF;

  IF pIdentityCode IS NULL THEN
    lErrorDescrip:=q'[IdentityCode can't be null]';
    RAISE lException;
  END IF;

  IF pIdentificationNumber IS NULL THEN
    lErrorDescrip:=q'[IdentificationNumber can't be null]';
    RAISE lException;
  END IF;

  IF pBankCode IS NULL THEN
    lErrorDescrip:=q'[BankCode can't be null]';
    RAISE lException;
  END IF;

  IF pAccountCode IS NULL THEN
    lErrorDescrip:=q'[AccountCode can't be null]';
    RAISE lException;
  END IF;

  --Check datatypes: Number values, quotes in strings, lenght...
  IF LENGTH(pConsumerName)>100 THEN
    lErrorDescrip:='ConsumerName too long, >100';
    RAISE lException;
  END IF;

  IF LENGTH(pConsumerSurname)>100 THEN
    lErrorDescrip:='ConsumerSurname too long, >100';
    RAISE lException;
  END IF;

  IF pSex NOT IN ('M','F') THEN
    lErrorDescrip:='Sex must be M or F';
    RAISE lException;
  END IF;

  IF LENGTH(pMobilePhone)>9 THEN
    lErrorDescrip:='MobilePhone too long, >9';
    RAISE lException;
  END IF;

  IF LENGTH(pIdentificationNumber)>50 THEN
    lErrorDescrip:='IdentificationNumber too long, >50';
    RAISE lException;
  END IF;

  IF LENGTH(pBankCode)>11 THEN
    lErrorDescrip:='BankCode too long, >9';
    RAISE lException;
  END IF;

  IF LENGTH(pAccountCode)>24 THEN
    lErrorDescrip:='AccountCode too long, >24';
    RAISE lException;
  END IF;

    -- Foreign Keys validation -->The system can do itself


  --Insert the Consumer and return the code assigned
  INSERT INTO Consumer (
    consumerName, consumerSurname, sex, mobilePhone, addressCode, identityCode,
identificationNumber, bankCode, accountCode)
  VALUES (
    pConsumerName, pConsumerSurname, pSex, pMobilePhone, pAddressCode, pIdentityCode,
pIdentificationNumber, pBankCode, pAccountCode) RETURNING ConsumerCode INTO
pConsumerCode;

  RSP:='OK'|| q'[, ]' || to_char(pConsumerCode);

  --Track Store Procedure Result
```

```
  LInParam:=to_char(pConsumerCode)  ||  q'[,  ']'  ||  pConsumerName  ||  q'[',  ']'  ||
pConsumerSurname  ||  q'[',  ']'||  pSex  ||  q'[',  ']'  ||  pMobilePhone  ||  q'[',  ]'  ||
to_char(pAddressCode)
                                  ||  q'[,  ]'  ||  to_char(pIdentityCode)  ||  q'[,  ']'  ||
pIdentificationNumber || q'[', ']' || pBankCode || q'[', ']' || pAccountCode || q']';

  Ins_LuzLog($$PLSQL_UNIT, LInParam , RSP , lRSP);

  DBMS_OUTPUT.PUT_LINE ('ConsumerCode:' || to_char(pConsumerCode));
  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
    LInParam:=to_char(pConsumerCode)  ||  q'[,  ']'  ||  pConsumerName  ||  q'[',  ']'  ||
pConsumerSurname  ||  q'[',  ']'||  pSex  ||  q'[',  ']'  ||  pMobilePhone  ||  q'[',  ]'  ||
to_char(pAddressCode)
                                    ||  q'[,  ]'  ||  to_char(pIdentityCode)  ||  q'[,  ']'  ||
pIdentificationNumber || q'[', ']' || pBankCode || q'[', ']' || pAccountCode || q']';
    Ins_LuzLog($$PLSQL_UNIT, LInParam , RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    RSP:='ERROR: ' || lErrorDescrip;
    LInParam:=to_char(pConsumerCode)  ||  q'[,  ']'  ||  pConsumerName  ||  q'[',  ']'  ||
pConsumerSurname  ||  q'[',  ']'||  pSex  ||  q'[',  ']'  ||  pMobilePhone  ||  q'[',  ]'  ||
to_char(pAddressCode)
                                    ||  q'[,  ]'  ||  to_char(pIdentityCode)  ||  q'[,  ']'  ||
pIdentificationNumber || q'[', ']' || pBankCode || q'[', ']' || pAccountCode || q']';
    Ins_LuzLog($$PLSQL_UNIT, LInParam , RSP, lRSP);

END;
```

El procedimiento INS_COMPANY introduce valores en la tabla COMPANY.

```
/*****************************************************************************************
************************************/
CREATE OR REPLACE PROCEDURE Ins_Company (
/*********************************************
| Insert Company data in Company table
|
| In Params:   companyTaxCode, companyName, addressCode
| Out Params:  RSP
| Date:
| Programmer: AJCC
|*********************************************
*/
    pCompanyTaxCode IN  Company.companyTaxCode%TYPE
  , pCompanyName    IN  Company.companyName%TYPE
  , pAddressCode    IN  Company.addressCode%TYPE
  , RSP             OUT VARCHAR2
)
AS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
BEGIN

  /* Validation Controls */


  --Check null values
  IF pCompanyTaxCode IS NULL THEN
    lErrorDescrip:=q'[CompanyTaxCode can't be null]';
    RAISE lException;
  END IF;

  IF pCompanyName IS NULL THEN
    lErrorDescrip:=q'[CompanyName can't be null]';
    RAISE lException;
```

```
  END IF;

  IF pAddressCode IS NULL THEN
    lErrorDescrip:=q'[AddressCode can't be null]';
    RAISE lException;
  END IF;

  --Check datatypes: Number values, quotes in strings, lenght...
  IF LENGTH(pCompanyTaxCode)>15 THEN
    lErrorDescrip:='CompanyTaxCode too long, >15';
    RAISE lException;
  END IF;

  IF LENGTH(pCompanyName)>200 THEN
    lErrorDescrip:='CompanyName too long, >200';
    RAISE lException;
  END IF;

  -- Foreign Keys validation, the system can do itself

  --Insert the Company
  INSERT INTO Company (companyTaxCode, companyName, addressCode) VALUES (pCompanyTaxCode,
pCompanyName, pAddressCode);
  RSP:='OK';

  --Track Store Procedure Result
  Ins_LuzLog($$PLSQL_UNIT, q'[']' ||pCompanyTaxCode || q'[', ']' || pCompanyName || q'[',
]' || to_char(pAddressCode), RSP, lRSP);


  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
    Ins_LuzLog($$PLSQL_UNIT, q'[']' ||pCompanyTaxCode || q'[', ']' || pCompanyName ||
q'[', ]' || to_char(pAddressCode), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    Ins_LuzLog($$PLSQL_UNIT, q'[']' ||pCompanyTaxCode || q'[', ']' || pCompanyName ||
q'[', ]' || to_char(pAddressCode), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento INS_METER introduce valores en la tabla METER.

```
/*****************************************************************************************
************************************/
CREATE OR REPLACE PROCEDURE Ins_Meter (
/*********************************************
| Insert Meter data in Meter table
|
|   In   Params:         serialNumber,   meterModel,   contractCode,   contractedPower,
lastTechnicalInspection, installationDate, companyCode, consumerCode, addressCode
| Out Params:  RSP
| Date:
| Programmer: AJCC
|*********************************************
*/
    pSerialNumber             IN  Meter.serialNumber%TYPE
  , pMeterModel               IN  Meter.meterModel%TYPE
  , pContractCode             IN  Meter.contractCode%TYPE
  , pContractedPower          IN  Meter.contractedPower%TYPE
  , pLastTechnicalInspection  IN  Meter.lastTechnicalInspection%TYPE
  , pInstallationDate         IN  Meter.installationDate%TYPE
  , pCompanyCode              IN  Meter.companyCode%TYPE
  , pConsumerCode             IN  Meter.consumerCode%TYPE
  , pAddressCode              IN  Meter.addressCode%TYPE
  , RSP                       OUT VARCHAR2
```

```
)
IS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
  lInparam      VARCHAR2(2000);
BEGIN

  /* Validation Controls */

    --Check null values
  IF pSerialNumber IS NULL THEN
    lErrorDescrip:=q'[SerialNumber can't be null]';
    RAISE lException;
  END IF;

  IF pMeterModel IS NULL THEN
    lErrorDescrip:=q'[MeterMode can't be null]';
    RAISE lException;
  END IF;

  IF pContractedPower IS NULL THEN
    lErrorDescrip:=q'[contractedPower can't be null]';
    RAISE lException;
  END IF;

  IF pInstallationDate IS NULL THEN
    lErrorDescrip:=q'[InstallationDate can't be null]';
    RAISE lException;
  END IF;

  IF pCompanyCode IS NULL THEN
    lErrorDescrip:=q'[companyCode can't be null]';
    RAISE lException;
  END IF;

  IF pAddressCode IS NULL THEN
    lErrorDescrip:=q'[AddressCode can't be null]';
    RAISE lException;
  END IF;

  --Check datatypes: Number values, quotes in strings, lenght...
  IF LENGTH(pSerialNumber)>20 THEN
    lErrorDescrip:='SerialNumber too long, >20';
    RAISE lException;
  END IF;

  IF LENGTH(pMeterModel)>100 THEN
    lErrorDescrip:='MeterModel too long, >100';
    RAISE lException;
  END IF;

  IF LENGTH(pContractCode)>100 THEN
    lErrorDescrip:='ContractCode too long, >10';
    RAISE lException;
  END IF;

  IF pInstallationDate>pLastTechnicalInspection THEN
    lErrorDescrip:='InstallationDate before LastTechnicalInspection?';
    RAISE lException;
  END IF;

  IF LENGTH(pCompanyCode)>15 THEN
    lErrorDescrip:='CompanyCode too long, >15';
    RAISE lException;
  END IF;

  IF LENGTH(pCompanyCode)>15 THEN
    lErrorDescrip:='CompanyCode too long, >15';
    RAISE lException;
  END IF;

  -- Foreign Keys validation -->The system can do itself

  --Insert the Address
  INSERT INTO METER(
```

```
    SERIALNUMBER
    , METERMODEL
    , CONTRACTCODE
    , CONTRACTEDPOWER
    , LASTTECHNICALINSPECTION
    , INSTALLATIONDATE
    , COMPANYCODE
    , CONSUMERCODE
    , ADDRESSCODE)
  VALUES (
    pSERIALNUMBER
    , pMETERMODEL
    , pCONTRACTCODE
    , pCONTRACTEDPOWER
    , pLASTTECHNICALINSPECTION
    , pINSTALLATIONDATE
    , pCOMPANYCODE
    , pCONSUMERCODE
    , pADDRESSCODE);

  RSP:='OK';

  --Track Store Procedure Result
  lInparam:= q'[']' || pSERIALNUMBER || q'[', ']' || pMETERMODEL || q'[', ']' ||
pCONTRACTCODE || q'[', ]' || to_char(pCONTRACTEDPOWER) || q'[, ']'
                    || to_char(pLASTTECHNICALINSPECTION, 'yyyy/mm/dd')   || q'[', ']' ||
to_char(pINSTALLATIONDATE, 'yyyy/mm/dd') || q'[', ']'
                    || pCOMPANYCODE || q'[', ]' || to_char(pCONSUMERCODE) || q'[, ]'  ||
to_char(pADDRESSCODE);

  Ins_LuzLog($$PLSQL_UNIT, lInparam, RSP, lRSP);

  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
    lInparam:= q'[']' || pSERIALNUMBER || q'[', ']' || pMETERMODEL || q'[', ']' ||
pCONTRACTCODE || q'[', ]' || to_char(pCONTRACTEDPOWER) || q'[, ']'
                    || to_char(pLASTTECHNICALINSPECTION, 'yyyy/mm/dd')    || q'[', ']'
|| to_char(pINSTALLATIONDATE, 'yyyy/mm/dd') || q'[', ']'
                    || pCOMPANYCODE || q'[', ]' || to_char(pCONSUMERCODE) || q'[, ]'  ||
to_char(pADDRESSCODE);

    Ins_LuzLog($$PLSQL_UNIT, lInparam, RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    lInparam:= q'[']' || pSERIALNUMBER || q'[', ']' || pMETERMODEL || q'[', ']' ||
pCONTRACTCODE || q'[', ]' || to_char(pCONTRACTEDPOWER) || q'[, ']'
                    || to_char(pLASTTECHNICALINSPECTION, 'yyyy/mm/dd')    || q'[', ']'
|| to_char(pINSTALLATIONDATE, 'yyyy/mm/dd') || q'[', ']'
                    || pCOMPANYCODE || q'[', ]' || to_char(pCONSUMERCODE) || q'[, ]'  ||
to_char(pADDRESSCODE);

    Ins_LuzLog($$PLSQL_UNIT, lInparam, RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento INS_CONNECTION introduce valores en la tabla CONNECTION.

```
/*******************************************************************************
************************************/
CREATE OR REPLACE PROCEDURE Ins_Connection (
/*********************************************
| Insert Connection data in Connection table
|
| In Params:   readingDate, meterSerialNumber, instantConsumption, isSuccess
| Out Params:  RSP
```

```
| Date:
| Programmer: AJCC
|*********************************************
*/
    pReadingDate        IN  Connection.readingDate%TYPE
  , pMeterSerialNumber  IN  Connection.meterSerialNumber%TYPE
  , pInstantConsumption IN  Connection.instantConsumption%TYPE
  , pisSuccess          IN  Connection.isSuccess%TYPE
  , RSP                 OUT VARCHAR2
)
IS
  lException          EXCEPTION;
  lErrorDescrip       VARCHAR2(100);
  lRSP                VARCHAR2(2000);
  lInparam            VARCHAR2(2000);

BEGIN

  /* Validation Controls */


  --Check null values
  IF pReadingDate IS NULL THEN
    lErrorDescrip:=q'[ReadingDate can't be null]';
    RAISE lException;
  END IF;

  IF pMeterSerialNumber IS NULL THEN
    lErrorDescrip:=q'[MeterSerialNumber can't be null]';
    RAISE lException;
  END IF;

  IF pInstantConsumption IS NULL THEN
    lErrorDescrip:=q'[InstantConsumption can't be null]';
    RAISE lException;
  END IF;

  IF pIsSuccess IS NULL THEN
    lErrorDescrip:=q'[IsSuccess can't be null]';
    RAISE lException;
  END IF;

  --Check datatypes: Number values, quotes in strings, lenght...
  IF LENGTH(pMeterSerialNumber)>20 THEN
    lErrorDescrip:='meterSerialNumber too long, >20';
    RAISE lException;
  END IF;

  IF pIsSuccess NOT IN ('Y','N') THEN
    lErrorDescrip:='isSuccess must be Y or N';
    RAISE lException;
  END IF;

  -- Foreign Keys validation -->The system can do itself

  --Insert the Connection
  INSERT INTO Connection (readingDate, meterSerialNumber, instantConsumption, isSuccess)
VALUES (pReadingDate, pMeterSerialNumber, pInstantConsumption, pIsSuccess);
  RSP:='OK';

  --Track Store Procedure Result
  lInparam:= q'[']' || to_char(preadingDate, 'DD/MM/YYYY HH24:MI:SS')  || q'[', ']' ||
pMeterSerialNumber || q'[', ]' || 'XXXX' || q'[, ']' || 'X' || q'[']';

  Ins_LuzLog($$PLSQL_UNIT,lInparam, RSP, lRSP);

  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
    lInparam:= q'[']' || to_char(preadingDate, 'DD/MM/YYYY HH24:MI:SS')  || q'[', ']' ||
pMeterSerialNumber || q'[', ]' || 'XXXX' || q'[, ']' || 'X' || q'[']';
    Ins_LuzLog($$PLSQL_UNIT,lInparam, RSP, lRSP);
```

```
    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    lInparam:= q'[']' || to_char(preadingDate, 'DD/MM/YYYY HH24:MI:SS')  || q'[', ']' ||
pMeterSerialNumber || q'[', ]' || 'XXXX' || q'[, ']' || 'X' || q'[']';
    Ins_LuzLog($$PLSQL_UNIT,lInparam, RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento INS_PRICE introduce valores en la tabla PRICE.

```
/*****************************************************************************************
***************************************/
CREATE OR REPLACE PROCEDURE Ins_Price (
/*********************************************
| Insert Price data in Price table
|
| In Params:   changeData, countryCode, companyCode, newPrice
| Out Params:  RSP
| Date:
| Programmer: AJCC
|*********************************************
*/
    pchangeData  IN  Price.changeData%TYPE
  , pcountryCode IN  Price.countryCode%TYPE
  , pcompanyCode IN  Price.companyCode%TYPE
  , pnewPrice    IN  Price.newPrice%TYPE
  , RSP          OUT VARCHAR2

)
IS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
  lInparam      VARCHAR2(2000);
BEGIN

  /* Validation Controls */

  --Check null values
  IF pChangeData IS NULL THEN
    lErrorDescrip:=q'[ChangeData can't be null]';
    RAISE lException;
  END IF;

  IF pCountryCode IS NULL THEN
    lErrorDescrip:=q'[CountryCode can't be null]';
    RAISE lException;
  END IF;

  IF pCompanyCode IS NULL THEN
    lErrorDescrip:=q'[CompanyCode can't be null]';
    RAISE lException;
  END IF;

  IF pNewPrice IS NULL THEN
    lErrorDescrip:=q'[NewPrice can't be null]';
    RAISE lException;
  END IF;


  --Check datatypes: Number values, quotes in strings, lenght...
  IF pchangeData<SYSDATE THEN
    lErrorDescrip:='ChangeData before Now???';
    RAISE lException;
  END IF;

  IF LENGTH(pCompanyCode)>15 THEN
    lErrorDescrip:='CompanyCode too long, >15';
    RAISE lException;
  END IF;

  IF pNewPrice<0 THEN
```

```
    lErrorDescrip:=q'[NewPrice must be greather than 0]';
    RAISE lException;
  END IF;

  -- Foreign Keys validation -->The system can do itself

  --Insert the Address
  INSERT INTO Price(
    changeData
    , countryCode
    , companyCode
    , newPrice
    )
  VALUES (
    pChangeData
    , pCountryCode
    , pCompanyCode
    , pNewPrice
    );

  RSP:='OK';

  --Track Store Procedure Result
  Ins_LuzLog($$PLSQL_UNIT, q'[']' || to_char(pChangeData, 'yyyy/mm/dd')  || q'[', ]' ||
to_char(pCountryCode)    ||    q'[,    ']'    ||    pCompanyCode    ||    q'[',    ]'||
to_char(pNewPrice,'9.999999'), RSP, lRSP);

  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
    Ins_LuzLog($$PLSQL_UNIT, q'[']' || to_char(pChangeData, 'yyyy/mm/dd')  || q'[', ]' ||
to_char(pCountryCode)    ||    q'[,    ']'    ||    pCompanyCode    ||    q'[',    ]'||
to_char(pNewPrice,'9.999999'), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    Ins_LuzLog($$PLSQL_UNIT, q'[']' || to_char(pChangeData, 'yyyy/mm/dd')  || q'[', ]' ||
to_char(pCountryCode)    ||    q'[,    ']'    ||    pCompanyCode    ||    q'[',    ]'||
to_char(pNewPrice,'9.999999'), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento INS_CLIENTS introduce valores en la tabla CLIENTS.

```
/*******************************************************************************************
************************************/
CREATE OR REPLACE PROCEDURE Ins_Clients (
/*********************************************
| Insert Client data in Clients table
|
| In Params:   companyCode, consumerCode
| Out Params:  RSP
| Date:
| Programmer: AJCC
|*********************************************
*/
    pHireDate     IN  Clients.hireDate%TYPE
  , pCompanyCode  IN  Clients.companyCode%TYPE
  , pConsumerCode IN  Clients.consumerCode%TYPE
  , RSP           OUT VARCHAR2
)
IS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
  lInparam      VARCHAR2(2000);
BEGIN
```

```
  /* Validation Controls */

  --Check null values
  IF pHireDate IS NULL THEN
    lErrorDescrip:=q'[HireDate can't be null]';
    RAISE lException;
  END IF;


  IF pCompanyCode IS NULL THEN
    lErrorDescrip:=q'[CompanyCode can't be null]';
    RAISE lException;
  END IF;

  IF pConsumerCode IS NULL THEN
    lErrorDescrip:=q'[ConsumerCode can't be null]';
    RAISE lException;
  END IF;

  --Check datatypes: Number values, quotes in strings, lenght...
  IF pHireDate< SYSDATE THEN
    lErrorDescrip:='HireDate before NOW???';
    RAISE lException;
  END IF;

  IF LENGTH(pCompanyCode)>15 THEN
    lErrorDescrip:='CompanyCode too long, >15';
    RAISE lException;
  END IF;

  -- Foreign Keys validation -->The system can do itself

  --Insert the Clients
  INSERT INTO CLIENTS
    (CONSUMERCODE ,COMPANYCODE ,HIREDATE)
  VALUES
    ( pCONSUMERCODE ,pCOMPANYCODE ,pHIREDATE);

  RSP:='OK';

  --Track Store Procedure Result
  lInparam:=   q'[']'   ||   to_char(pHireDate,   'DD/MM/YYYY')      ||   q'[',   ]'   ||
to_char(pConsumerCode) || q'[, ']' || pCompanyCode || q'[']' ;
  Ins_LuzLog($$PLSQL_UNIT, lInparam, RSP, lRSP);

  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
    lInparam:=   q'[']'   ||   to_char(pHireDate,   'DD/MM/YYYY')      ||   q'[',   ]'   ||
to_char(pConsumerCode) || q'[, ']' || pCompanyCode || q'[']' ;
    Ins_LuzLog($$PLSQL_UNIT, lInparam, RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    lInparam:=   q'[']'   ||   to_char(pHireDate,   'DD/MM/YYYY')      ||   q'[',   ]'   ||
to_char(pConsumerCode) || q'[, ']' || pCompanyCode || q'[']' ;
    Ins_LuzLog($$PLSQL_UNIT, lInparam, RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento INS_OEPRATORS introduce valores en la tabla OPERATORS.

```
/*****************************************************************************************
************************************/
CREATE OR REPLACE PROCEDURE Ins_Operators (
/*******************************************
```

```
| Insert Operators data in Operators table
|
| In Params:   countryCode, companyCode
| Out Params:  RSP
| Date:
| Programmer: AJCC
|*********************************************
*/
    pCountryCode  IN  Operators.CountryCode%TYPE
  , pCompanyCode  IN  Clients.companyCode%TYPE
  , RSP           OUT VARCHAR2
)
IS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
BEGIN

  /* Validation Controls */

  --Check null values
  IF pCountryCode IS NULL THEN
    lErrorDescrip:=q'[CountryCode can't be null]';
    RAISE lException;
  END IF;


  IF pCompanyCode IS NULL THEN
    lErrorDescrip:=q'[CompanyCode can't be null]';
    RAISE lException;
  END IF;

  --Check datatypes: Number values, quotes in strings, lenght...
  IF LENGTH(pCompanyCode)>15 THEN
    lErrorDescrip:='CompanyCode too long, >15';
    RAISE lException;
  END IF;

  -- Foreign Keys validation -->The system can do itself

  --Insert the Clients
  INSERT INTO OPERATORS
    (CountryCode ,COMPANYCODE)
  VALUES
    ( pCountryCode ,pCOMPANYCODE);

  RSP:='OK';

  --Track Store Procedure Result
  Ins_LuzLog($$PLSQL_UNIT,  to_char(pCountryCode) || q'[, ']' || pCompanyCode || q']',
RSP, lRSP);

  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
    Ins_LuzLog($$PLSQL_UNIT,  to_char(pCountryCode) || q'[, ']' || pCompanyCode || q']',
RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    Ins_LuzLog($$PLSQL_UNIT,  to_char(pCountryCode) || q'[, ']' || pCompanyCode || q']',
RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

Procedimientos almacenados de borrado de datos

```
/*creacion de las store procedure para borrar datos */
```

El procedimiento DEL_LUZLOG borra el registro de la tabla LUZLOG cuyo ID se pasa como parámetro.

```
/*****************************************************************************************
*******************************/
CREATE OR REPLACE PROCEDURE Del_LuzLog(
/*********************************************
| Delete LuzLog data from LuzLog
|
| In Params:
|        - Store Procedure Name
|        - In Params
|        - Out Params
| Out Params:  --
| Date:
| Author: AJCC
|*********************************************
*/
    pIdLog  IN  LuzLog.IdLog%TYPE
  , RSP     OUT VARCHAR2
)
IS
  lException EXCEPTION;
  lIdLog INTEGER;
  lErrorDescrip VARCHAR2(2000);

BEGIN
  /* Validation Controls */

  --Check null values
  IF pIdLog IS NULL THEN
    lErrorDescrip:=q'[IdLog can't be null]';
    RAISE lException;
  END IF;

  DELETE FROM LuzLog WHERE IdLog=pIdLog;
  RSP:='OK';

  DBMS_OUTPUT.PUT_LINE ($$PLSQL_UNIT ||':' || RSP);
EXCEPTION
  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
    DBMS_OUTPUT.PUT_LINE ($$PLSQL_UNIT ||':'|| RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    DBMS_OUTPUT.PUT_LINE ($$PLSQL_UNIT ||':'|| RSP);

END Del_LuzLog;
```

El procedimiento DEL_COUNTRY borra el registro de la tabla COUNTRY cuyo ID se pasa como parámetro.

```
/*****************************************************************************************
*****************************/
create or replace PROCEDURE Del_Country (
/*********************************************
|  Delete Country data from Country table
|
| In Params:   CountryCode
| Out Params:  RSP
| Date:
```

```
| Programmer:  AJCC
|*********************************************
*/
  pCountryCode   Country.CountryCode%TYPE
  , RSP          OUT VARCHAR2
)
AS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
BEGIN
  /* Validation Controls */

  --Check null values
  IF pCountryCode IS NULL THEN
    lErrorDescrip:=q'[CountryCode can't be null]';
    RAISE lException;
  END IF;

  --Delete the Country
  DELETE FROM COUNTRY WHERE countryCode=pCountryCode;
  RSP:='OK';

  --Track Store Procedure Result
  Ins_LuzLog($$PLSQL_UNIT,to_char(pCountryCode), RSP, lRSP);

  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
    Ins_LuzLog($$PLSQL_UNIT,to_char(pCountryCode), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    Ins_LuzLog($$PLSQL_UNIT,to_char(pCountryCode), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento DEL_PROVINCE borra el registro de la tabla PROVINCE cuyo ID se pasa como parámetro.

```
/*****************************************************************************************
****************************/
CREATE OR REPLACE PROCEDURE Del_Province (
/*********************************************
|  Delete Province data from Province
|
| In Params:   provinceId
| Out Params:  RSP
| Date:
| Programmer: AJCC
|*********************************************
*/
    pProvinceId   IN  Province.provinceId%TYPE
  , RSP           OUT VARCHAR2
)
AS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
BEGIN
   /* Validation Controls */


  --Check null values
  IF pProvinceId IS NULL THEN
    lErrorDescrip:=q'[provinceId can't be null]';
```

```
    RAISE lException;
  END IF;

  -- Foreign Keys validation ->The system can do itself

  --Delete the Province
  DELETE FROM PROVINCE WHERE provinceId=pprovinceId;
  RSP:='OK';

  --Track Store Procedure Result
  Ins_LuzLog($$PLSQL_UNIT,to_char(pProvinceId), RSP, lRSP);

  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
   Ins_LuzLog($$PLSQL_UNIT,to_char(pProvinceId), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    Ins_LuzLog($$PLSQL_UNIT,to_char(pProvinceId), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento DEL_CITY borra el registro de la tabla CITY cuyo ID se pasa como parámetro.

```
/*****************************************************************************************
****************************************/
CREATE OR REPLACE PROCEDURE Del_City (
/*********************************************
|  Delete City data from City table
|
| In Params:   cityCode
| Out Params:  RSP
| Date:
| Programmer: AJCC
|*********************************************
*/
    pCityCode     IN  City.cityCode%TYPE
  , RSP           OUT VARCHAR2
)
AS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
BEGIN

  /* Validation Controls */


  --Check null values
  IF pCityCode IS NULL THEN
    lErrorDescrip:=q'[CityCode can't be null]';
    RAISE lException;
  END IF;

  -- Foreign Keys validation -->The system can do itself

  --Delete the City
  DELETE FROM CITY WHERE CITYCODE=pCityCode;
  RSP:='OK';

  --Track Store Procedure Result
  Ins_LuzLog($$PLSQL_UNIT,to_char(pCityCode) , RSP, lRSP);

  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION
```

```
  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
    Ins_LuzLog($$PLSQL_UNIT,to_char(pCityCode) , RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    Ins_LuzLog($$PLSQL_UNIT,to_char(pCityCode) , RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento DEL_STREETTYPE borra el registro de la tabla STREETTYPE cuyo ID se pasa como parámetro.

```
/*****************************************************************************************
************************************/
CREATE OR REPLACE PROCEDURE Del_StreetType (
/*******************************************
|  Delete Street types data from StreetType table
|
| In Params:   streetTypeCode
| Out Params:  RSP
| Date:
| Programmer: AJCC
|*******************************************
*/
    pStreetTypeCode IN  StreetType.streetTypeCode%TYPE
  , RSP             OUT VARCHAR2
)
AS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
BEGIN

  /* Validation Controls */

  --Check null values
  IF pStreetTypeCode IS NULL THEN
    lErrorDescrip:=q'[StreetTypeCode can't be null]';
    RAISE lException;
  END IF;

  --Delete the StreetType
  DELETE FROM  StreetType WHERE STREETTYPECODE=pStreetTypeCode;
  RSP:='OK';

  --Track Store Procedure Result
  Ins_LuzLog($$PLSQL_UNIT,to_char(pStreetTypeCode), RSP, lRSP);

  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
    Ins_LuzLog($$PLSQL_UNIT,to_char(pStreetTypeCode), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    Ins_LuzLog($$PLSQL_UNIT,to_char(pStreetTypeCode), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento DEL_ADDRESS borra el registro de la tabla ADDRESS cuyo ID se pasa como parámetro.

```
/*************************************************************************************************
**************************************/
CREATE OR REPLACE PROCEDURE Del_Address (
/*********************************************
|  Delete Address data from Address table
|
| In Params:   addressCode
| Out Params:  RSP, addressCode
| Date:
| Programmer: AJCC
|*********************************************
*/
  pAddressCode    IN Address.addressCode%TYPE
  , RSP             OUT VARCHAR2

)
AS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
BEGIN

  /* Validation Controls */

  --Check null values
  IF pAddressCode IS NULL THEN
    lErrorDescrip:=q'[AddressCode can't be null]';
    RAISE lException;
  END IF;

  -- Foreign Keys validation -->The system can do itself


  --Delete the Address
  DELETE FROM Address WHERE ADDRESSCODE =pAddressCode;

  RSP:='OK';

  --Track Store Procedure Result
  Ins_LuzLog($$PLSQL_UNIT, to_char(pAddressCode), RSP, lRSP);

  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
    Ins_LuzLog($$PLSQL_UNIT, to_char(pAddressCode), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    Ins_LuzLog($$PLSQL_UNIT, to_char(pAddressCode), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento DEL_BANK borra el registro de la tabla BANK cuyo ID se pasa como parámetro.

```
/*************************************************************************************************
**************************************/
CREATE OR REPLACE PROCEDURE Del_Bank (
/*********************************************
| Delete Bank data from Bank table
|
| In Params:   bankCode
| Out Params:  RSP
```

```
| Date:
| Programmer: AJCC
|*********************************************
*/
    pBankCode     IN  Bank.bankCode%TYPE
  , RSP           OUT VARCHAR2
)
IS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
BEGIN

  /* Validation Controls */

  --Check null values
  IF pBankCode IS NULL THEN
    lErrorDescrip:=q'[BankCode can't be null]';
    RAISE lException;
  END IF;

  --Check datatypes: Number values, quotes in strings, lenght...
  IF LENGTH(pBankCode)>11 THEN
    lErrorDescrip:='BankCode too long, >11';
    RAISE lException;
  END IF;


  --Delete the Bank
  DELETE FROM BANK WHERE BANKCODE=pBankCode;
  RSP:='OK';

  --Track Store Procedure Result
  Ins_LuzLog($$PLSQL_UNIT,q'[']'|| to_char(pBankCode)|| q'[']', RSP, lRSP);

  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
    Ins_LuzLog($$PLSQL_UNIT,q'[']'|| to_char(pBankCode)|| q'[']', RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    Ins_LuzLog($$PLSQL_UNIT,q'[']'|| to_char(pBankCode)|| q'[']', RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento DEL_IDENTITYTYPE borra el registro de la tabla IDENTITYTYPE cuyo ID se pasa como parámetro.

```
/*****************************************************************************************
***************************************/
CREATE OR REPLACE PROCEDURE Del_IdentityType (
/*********************************************
| Delete IdentityType data from IdentityType table
|
| In Params:   identityCode
| Out Params:  RSP
| Date:
| Programmer: AJCC
|*********************************************
*/
    pIdentityCode IN  IdentityType.IdentityType%TYPE
  , RSP           OUT VARCHAR2
)
AS
  lException    EXCEPTION;
```

```
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
BEGIN

  /* Validation Controls */

  --Check null values
  IF pIdentityCode IS NULL THEN
    lErrorDescrip:=q'[IdentityCode can't be null]';
    RAISE lException;
  END IF;

  --Delete the IdentityType
  DELETE FROM IdentityType WHERE IdentityCode=pIdentityCode;
  RSP:='OK';

  --Track Store Procedure Result
  Ins_LuzLog($$PLSQL_UNIT,to_char(pIdentityCode), RSP, lRSP);

  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
    Ins_LuzLog($$PLSQL_UNIT,to_char(pIdentityCode), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    Ins_LuzLog($$PLSQL_UNIT,to_char(pIdentityCode), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento DEL_CONSUMER borra el registro de la tabla CONSUMER cuyo ID se pasa como parámetro.

```
/*****************************************************************************************
***************************************/
CREATE OR REPLACE PROCEDURE Del_Consumer (
/*********************************************
|  Delete City data from City table
|
| In Params:   consumerCode
| Out Params:  RSP
| Date:
| Programmer: AJCC
|*********************************************
*/
    pConsumerCode         IN Consumer.consumerCode%TYPE
  , RSP                   OUT VARCHAR2

)
AS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
BEGIN

  /* Validation Controls */

  --Check null values
  IF pConsumerCode IS NULL THEN
    lErrorDescrip:=q'[ConsumerCode can't be null]';
    RAISE lException;
  END IF;

  --Check datatypes: Number values, quotes in strings, lenght...

  -- Foreign Keys validation -->The system can do itself
```

```
  --Delete the Consumer
  DELETE FROM CONSUMER WHERE CONSUMERCODE=pConsumerCode;
  RSP:='OK';

  --Track Store Procedure Result
  Ins_LuzLog($$PLSQL_UNIT, to_char(pConsumerCode) , RSP, lRSP);

  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
    Ins_LuzLog($$PLSQL_UNIT, to_char(pConsumerCode) , RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    Ins_LuzLog($$PLSQL_UNIT, to_char(pConsumerCode) , RSP, lRSP);

END;
```

El procedimiento DEL_COUNTRY borra el registro de la tabla COUNTRY cuyo ID se pasa como parámetro.

```
/*****************************************************************************************
***************************************/
CREATE OR REPLACE PROCEDURE Del_Company (
/*********************************************
| Delete Company data from Company table
|
| In Params:   companyTaxCode
| Out Params:  RSP
| Date:
| Programmer: AJCC
|*********************************************
*/
    pCompanyTaxCode IN  Company.companyTaxCode%TYPE
  , RSP             OUT VARCHAR2
)
AS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
BEGIN

  /* Validation Controls */


  --Check null values
  IF pCompanyTaxCode IS NULL THEN
    lErrorDescrip:=q'[CompanyTaxCode can't be null]';
    RAISE lException;
  END IF;

  --Check datatypes: Number values, quotes in strings, lenght...
  IF LENGTH(pCompanyTaxCode)>15 THEN
    lErrorDescrip:='CompanyTaxCode too long, >15';
    RAISE lException;
  END IF;

  -- Foreign Keys validation, the system can do itself

  --Delete the Company
  DELETE FROM COMPANY WHERE companyTaxCode=pCompanyTaxCode;
  RSP:='OK';

  --Track Store Procedure Result
  Ins_LuzLog($$PLSQL_UNIT, q'[']'|| pCompanyTaxCode || q'[']', RSP, lRSP);
```

```
  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
    Ins_LuzLog($$PLSQL_UNIT, q'[']'|| pCompanyTaxCode || q'[']', RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    Ins_LuzLog($$PLSQL_UNIT, q'[']'|| pCompanyTaxCode || q'[']', RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento DEL_METER borra el registro de la tabla METER cuyo ID se pasa como parámetro.

```
/*****************************************************************************************
************************************/
CREATE OR REPLACE PROCEDURE Del_Meter (
/*********************************************
| Delete Meter data from Meter table
|
| In Params:   serialNumber
| Out Params:  RSP
| Date:
| Programmer: AJCC
|*********************************************
*/
    pSerialNumber             IN  Meter.serialNumber%TYPE
  , RSP                       OUT VARCHAR2

)
IS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
BEGIN

  /* Validation Controls */

  --Check null values
  IF pSerialNumber IS NULL THEN
    lErrorDescrip:=q'[SerialNumber can't be null]';
    RAISE lException;
  END IF;

  --Check datatypes: Number values, quotes in strings, lenght...
  IF LENGTH(pSerialNumber)>20 THEN
    lErrorDescrip:='SerialNumber too long, >20';
    RAISE lException;
  END IF;

  -- Foreign Keys validation -->The system can do itself

  --Delete the Address
  DELETE FROM METER WHERE SERIALNUMBER=pSERIALNUMBER;
  RSP:='OK';

  --Track Store Procedure Result
  Ins_LuzLog($$PLSQL_UNIT, q'[']' || pSerialNumber || q'[']', RSP, lRSP);

  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
```

```
    Ins_LuzLog($$PLSQL_UNIT, q'[']' || pSerialNumber || q'[']', RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    Ins_LuzLog($$PLSQL_UNIT, q'[']' || pSerialNumber || q'[']', RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento DEL_CONNECTION borra el registro de la tabla CONNECTION cuyo ID se pasa como parámetro.

```
/*****************************************************************************************
****************************************/
CREATE OR REPLACE PROCEDURE Del_Connection (
/*********************************************
| Delete Connection data from Connection table
|
| In Params:   readingDate, meterSerialNumber
| Out Params:  RSP
| Date:
| Programmer: AJCC
|*********************************************
*/
    pReadingDate        IN  Connection.readingDate%TYPE
  , pMeterSerialNumber  IN  Connection.meterSerialNumber%TYPE
  , RSP                 OUT VARCHAR2
)
IS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
BEGIN

  /* Validation Controls */

  --Check null values
  IF pReadingDate IS NULL THEN
    lErrorDescrip:=q'[ReadingDate can't be null]';
    RAISE lException;
  END IF;

  IF pMeterSerialNumber IS NULL THEN
    lErrorDescrip:=q'[MeterSerialNumber can't be null]';
    RAISE lException;
  END IF;

  --Check datatypes: Number values, quotes in strings, lenght...
  IF LENGTH(pMeterSerialNumber)>20 THEN
    lErrorDescrip:='meterSerialNumber too long, >20';
    RAISE lException;
  END IF;

  -- Foreign Keys validation -->The system can do itself

  --Insert the Connection
  DELETE       FROM       Connection       WHERE       readingDate=pReadingDate       AND
meterSerialNumber=pMeterSerialNumber;
  RSP:='OK';

  --Track Store Procedure Result
  Ins_LuzLog($$PLSQL_UNIT,q'[']' || to_char(preadingDate, 'DD/MM/YYYY HH24:MI:SS')   ||
q'[', ']' || pMeterSerialNumber || q'[']', RSP, lRSP);

  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
```

```
    Ins_LuzLog($$PLSQL_UNIT,q'[']' || to_char(preadingDate, 'DD/MM/YYYY HH24:MI:SS')  ||
q'[', ']' || pMeterSerialNumber || q'[']', RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    Ins_LuzLog($$PLSQL_UNIT,q'[']' || to_char(preadingDate, 'DD/MM/YYYY HH24:MI:SS')  ||
q'[', ']' || pMeterSerialNumber || q'[']', RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento DEL_PRICE borra el registro de la tabla PRICE cuyo ID se pasa como parámetro.

```
/*****************************************************************************************
*************************************/
CREATE OR REPLACE PROCEDURE Del_Price (
/*********************************************
| Delete Price data from Price table
|
| In Params:   changeData, countryCode, companyCode
| Out Params:  RSP
| Date:
| Programmer:  AJCC
|*********************************************
*/
    pchangeData  IN  Price.changeData%TYPE
  , pcountryCode IN  Price.countryCode%TYPE
  , pcompanyCode IN  Price.companyCode%TYPE
  , RSP          OUT VARCHAR2

)
IS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
BEGIN

  /* Validation Controls */

  --Check null values
  IF pChangeData IS NULL THEN
    lErrorDescrip:=q'[ChangeData can't be null]';
    RAISE lException;
  END IF;

  IF pCountryCode IS NULL THEN
    lErrorDescrip:=q'[CountryCode can't be null]';
    RAISE lException;
  END IF;

  IF pCompanyCode IS NULL THEN
    lErrorDescrip:=q'[CompanyCode can't be null]';
    RAISE lException;
  END IF;

  --Check datatypes: Number values, quotes in strings, lenght...
  IF LENGTH(pCompanyCode)>15 THEN
    lErrorDescrip:='CompanyCode too long, >15';
    RAISE lException;
  END IF;


  -- Foreign Keys validation -->The system can do itself

  --Delete the Price
  DELETE FROM Price WHERE changeData=pChangeData AND countryCode=pCountryCode   AND
companyCode=pCompanyCode;
  RSP:='OK';

  --Track Store Procedure Result
  Ins_LuzLog($$PLSQL_UNIT, q'[']' || to_char(pChangeData, 'yyyy/mm/dd')  || q'[, ]' ||
to_char(pCountryCode) || q'[, ']' || pCompanyCode || q'[']', RSP, lRSP);
```

```
  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
    Ins_LuzLog($$PLSQL_UNIT, q'[']' || to_char(pChangeData, 'yyyy/mm/dd')  || q'[', ]' ||
to_char(pCountryCode) || q'[, ']' || pCompanyCode || q'[']', RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    Ins_LuzLog($$PLSQL_UNIT, q'[']' || to_char(pChangeData, 'yyyy/mm/dd')  || q'[', ]' ||
to_char(pCountryCode) || q'[, ']' || pCompanyCode || q'[']', RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento DEL_CLIENTS borra el registro de la tabla CLIENTS cuyos ID's se pasan como parámetro.

```
/*****************************************************************************************
***************************************/
CREATE OR REPLACE PROCEDURE Del_Clients (
/*********************************************
| Delete Client data from Clients table
|
| In Params:   companyCode, consumerCode
| Out Params:  RSP
| Date:
| Programmer: AJCC
|*********************************************
*/
    pCompanyCode  IN  Clients.companyCode%TYPE
  , pConsumerCode IN  Clients.consumerCode%TYPE
  , RSP           OUT VARCHAR2
)
IS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
BEGIN

  /* Validation Controls */

  --Check null values
  IF pCompanyCode IS NULL THEN
    lErrorDescrip:=q'[CompanyCode can't be null]';
    RAISE lException;
  END IF;

  IF pConsumerCode IS NULL THEN
    lErrorDescrip:=q'[ConsumerCode can't be null]';
    RAISE lException;
  END IF;

  --Check datatypes: Number values, quotes in strings, lenght...
  IF LENGTH(pCompanyCode)>15 THEN
    lErrorDescrip:='CompanyCode too long, >15';
    RAISE lException;
  END IF;

  -- Foreign Keys validation -->The system can do itself

  --Delete the Clients
  DELETE FROM CLIENTS WHERE CONSUMERCODE=pCONSUMERCODE AND COMPANYCODE=pCOMPANYCODE;
  RSP:='OK';

  --Track Store Procedure Result
```

```
  Ins_LuzLog($$PLSQL_UNIT, to_char(pCONSUMERCODE) || q'[, ']' || pCompanyCode || q'[']',
RSP, lRSP);

  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
    Ins_LuzLog($$PLSQL_UNIT, to_char(pCONSUMERCODE) || q'[, ']' || pCompanyCode || q'[']',
RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    Ins_LuzLog($$PLSQL_UNIT, to_char(pCONSUMERCODE) || q'[, ']' || pCompanyCode || q'[']',
RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento DEL_OPERATORS borra el registro de la tabla OPERATORS cuyos ID's se pasan como parámetro.

```
/*****************************************************************************************
***************************************/
CREATE OR REPLACE PROCEDURE Del_OPerators (
/*********************************************
| Delete Operator data from Operators table
|
| In Params:   countryCode, companyCode
| Out Params:  RSP
| Date:
| Programmer: AJCC
|*********************************************
*/
    pCountryCode  IN  Operators.countryCode%TYPE
  , pCompanyCode  IN  Operators.companyCode%TYPE
  , RSP           OUT VARCHAR2
)
IS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
BEGIN

  /* Validation Controls */

  --Check null values
  IF pCountryCode IS NULL THEN
    lErrorDescrip:=q'[CountryCode can't be null]';
    RAISE lException;
  END IF;

  IF pCompanyCode IS NULL THEN
    lErrorDescrip:=q'[CompanyCode can't be null]';
    RAISE lException;
  END IF;

  --Check datatypes: Number values, quotes in strings, lenght...
  IF LENGTH(pCompanyCode)>15 THEN
    lErrorDescrip:='CompanyCode too long, >15';
    RAISE lException;
  END IF;

  -- Foreign Keys validation -->The system can do itself

  --Delete the Operators
  DELETE FROM OPERATORS WHERE COUNTRYCODE=pCOUNTRYCODE AND COMPANYCODE=pCOMPANYCODE;
  RSP:='OK';
```

```
  --Track Store Procedure Result
  Ins_LuzLog($$PLSQL_UNIT,  to_char(pCountryCode) || q'[, ']' || pCompanyCode || q'[']',
RSP, lRSP);

  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
    Ins_LuzLog($$PLSQL_UNIT,  to_char(pCountryCode) || q'[, ']' || pCompanyCode || q'[']',
RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    Ins_LuzLog($$PLSQL_UNIT,  to_char(pCountryCode) || q'[, ']' || pCompanyCode || q'[']',
RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

Procedimientos almacenados de Modificación de datos

```
/*creacion de las store procedure para actualizar datos */
```

El procedimiento UPD_COUNTRY actualiza los datos de la tabla COUNTRY cuyo ID y descripción se pasan como parámetros.

```
/*****************************************************************************************
*******************************/
create or replace PROCEDURE Upd_Country (
/*********************************************
|  Update Country data in Country table
|
| In Params:   CountryCode, CountryName
| Out Params:  RSP
| Date:
| Programmer: AJCC
|*********************************************
*/
    pCountryCode   Country.CountryCode%TYPE
  , pCountryName Country.CountryName%TYPE
  , RSP          OUT VARCHAR2
)
AS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
BEGIN
  /* Validation Controls */

    --Check null values
  IF pCountryCode IS NULL THEN
    lErrorDescrip:=q'[CountryCode can't be null]';
    RAISE lException;
  END IF;

  IF pCountryName IS NULL THEN
    lErrorDescrip:=q'[CountryName can't be null]';
    RAISE lException;
  END IF;

  --Check datatypes: Number values, quotes in strings, lenght...
  IF LENGTH(pCountryName)>100 THEN
    lErrorDescrip:='CountryName too long, >100';
    RAISE lException;
  END IF;

  --Update the Country
  UPDATE COUNTRY SET COUNTRYNAME=pCountryName WHERE COUNTRYCODE= pCountryCode;
  RSP:='OK';

  --Track Store Procedure Result
  Ins_LuzLog($$PLSQL_UNIT, to_char(pCountryCode) || q'[, ']' || pCountryName ||  q']',
RSP, lRSP);

  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
    Ins_LuzLog($$PLSQL_UNIT, to_char(pCountryCode) || q'[, ']' || pCountryName ||  q']',
RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    Ins_LuzLog($$PLSQL_UNIT, to_char(pCountryCode) || q'[, ']' || pCountryName ||  q']',
RSP, lRSP);
```

```
    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento UPD_PROVINCE actualiza los datos de la tabla PROVINCE cuyo ID y descripción se pasan como parámetros.

```
/*****************************************************************************************
****************************/
CREATE OR REPLACE PROCEDURE Upd_Province (
/*********************************************
| Update Province data in Province table
|
| In Params:   provinceId, provinceCode, provinceName, CountryCode
| Out Params:  RSP
| Date:
| Programmer: AJCC
|*********************************************
*/
    pProvinceId   IN  Province.provinceId%TYPE
  , pProvinceCode IN  Province.provinceCode%TYPE
  , pProvinceName IN  Province.provinceName%TYPE
  , pCountryCode  IN  Province.CountryCode%TYPE
  , RSP           OUT VARCHAR2
)
AS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
BEGIN
   /* Validation Controls */


  --Check null values
  IF pProvinceId IS NULL THEN
    lErrorDescrip:=q'[ProvinceId can't be null]';
    RAISE lException;
  END IF;

  IF pProvinceCode IS NULL THEN
    lErrorDescrip:=q'[ProvinceCode can't be null]';
    RAISE lException;
  END IF;

  IF pProvinceName IS NULL THEN
    lErrorDescrip:=q'[ProvinceName can't be null]';
    RAISE lException;
  END IF;

  IF pCountryCode IS NULL THEN
    lErrorDescrip:=q'[CountryCode can't be null]';
    RAISE lException;
  END IF;

  --Check datatypes: Number values, quotes in strings, lenght...
  IF LENGTH(pProvinceCode)>10 THEN
    lErrorDescrip:='ProvinceCode too long, >10';
    RAISE lException;
  END IF;

  IF LENGTH(pProvinceName)>100 THEN
    lErrorDescrip:='ProvinceName too long, >100';
    RAISE lException;
  END IF;

  -- Foreign Keys validation
  -- Then system can do itself

  --Update the Province
  UPDATE     PROVINCE     SET     ProvinceCode=pProvinceCode,ProvinceName=pProvinceName,
CountryCode=pCountryCode WHERE ProvinceId=pProvinceId;
  RSP:='OK';

  --Track Store Procedure Result
```

```
  Ins_LuzLog($$PLSQL_UNIT,(to_char(pProvinceId) || q'[, ']' || pProvinceCode || q'[', ']'
|| pProvinceName || q'[', ]' || to_char(pCountryCode)), RSP, lRSP);

  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
    Ins_LuzLog($$PLSQL_UNIT,(to_char(pProvinceId) || q'[, ']' || pProvinceCode || q'[',
']' || pProvinceName || q'[', ]' || to_char(pCountryCode)), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    Ins_LuzLog($$PLSQL_UNIT,(to_char(pProvinceId) || q'[, ']' || pProvinceCode || q'[',
']' || pProvinceName || q'[', ]' || to_char(pCountryCode)), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento UPD_CITY actualiza los datos de la tabla CITY cuyo ID y descripción se pasan como parámetros.

```
/*****************************************************************************************
************************************/
CREATE OR REPLACE PROCEDURE Upd_City (
/*********************************************
|  Update City data in City table
|
| In Params:   cityCode, cityName, zipCode, provinceCode
| Out Params:  RSP
| Date:
| Programmer: AJCC
|*********************************************
*/
    pCityCode     IN  City.cityCode%TYPE
  , pCityName     IN  City.cityName%TYPE
  , pZipCode      IN  City.zipCode%TYPE
  , pProvinceCode IN  City.provinceCode%TYPE
  , RSP           OUT VARCHAR2
)
AS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
BEGIN

  /* Validation Controls */

  --Check null values
  IF pCityCode IS NULL THEN
    lErrorDescrip:=q'[CityCode can't be null]';
    RAISE lException;
  END IF;

  IF pCityName IS NULL THEN
    lErrorDescrip:=q'[CityName can't be null]';
    RAISE lException;
  END IF;

  IF pZipCode IS NULL THEN
    lErrorDescrip:=q'[ZipCode can't be null]';
    RAISE lException;
  END IF;

  IF pProvinceCode IS NULL THEN
    lErrorDescrip:=q'[ProvinceCode can't be null]';
    RAISE lException;
  END IF;
```

```
  --Check datatypes: Number values, quotes in strings, lenght...
  IF LENGTH(pCityName)>100 THEN
    lErrorDescrip:='CityName too long, >100';
    RAISE lException;
  END IF;

  IF LENGTH(pZipCode)>15 THEN
    lErrorDescrip:='ZipCode too long, >15';
    RAISE lException;
  END IF;


  -- Foreign Keys validation
  -- Then system can do itself

  --Update the City
  UPDATE CITY SET CITYNAME=pCityName, ZIPCODE=pZipCode, PROVINCECODE=pProvinceCode WHERE
CITYCODE=pCityCode;
  RSP:='OK';

  --Track Store Procedure Result
  Ins_LuzLog($$PLSQL_UNIT,(to_char(pCityCode) || q'[, ']' || pCityName || q'[', ']' ||
pZipCode || q'[', ]' || to_char(pProvinceCode)), RSP, lRSP);

  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
    Ins_LuzLog($$PLSQL_UNIT,(to_char(pCityCode) || q'[, ']' || pCityName || q'[', ']' ||
pZipCode || q'[', ]' || to_char(pProvinceCode)), RSP, lRSP);
    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    Ins_LuzLog($$PLSQL_UNIT,(to_char(pCityCode) || q'[, ']' || pCityName || q'[', ']' ||
pZipCode || q'[', ]' || to_char(pProvinceCode)), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento UPD_STREETTYPE actualiza los datos de la tabla STREETTYPE cuyo ID y descripción se pasan como parámetros.

```
/*****************************************************************************************
**************************************/
CREATE OR REPLACE PROCEDURE Upd_StreetType (
/*********************************************
|  Update Street types data in StreetType table
|
| In Params:   streetTypeCode, streetTypeName
| Out Params:  RSP
| Date:
| Programmer: AJCC
|*********************************************
*/
    pStreetTypeCode IN  StreetType.streetTypeCode%TYPE
  , pStreetTypeName IN  StreetType.streetTypeName%TYPE
  , RSP             OUT VARCHAR2
)
AS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
BEGIN

  /* Validation Controls */

  --Check null values
  IF pStreetTypeCode IS NULL THEN
```

```
    lErrorDescrip:=q'[StreetTypeCode can't be null]';
    RAISE lException;
  END IF;

  IF pStreetTypeName IS NULL THEN
    lErrorDescrip:=q'[StreetTypeName can't be null]';
    RAISE lException;
  END IF;

  --Check datatypes: Number values, quotes in strings, lenght...
  IF LENGTH(pStreetTypeName)>2 THEN
    lErrorDescrip:='StreetTypeName too long, >2';
    RAISE lException;
  END IF;


  --Update the StreetType
  UPDATE        StreetType        SET        STREETTYPENAME=pStreetTypeName        WHERE
STREETTYPECODE=pStreetTypeCode;
  RSP:='OK';

  --Track Store Procedure Result
  Ins_LuzLog($$PLSQL_UNIT,(to_char(pStreetTypeCode)  ||  q'[,  ']'  ||  pStreetTypeName  ||
q']'), RSP, lRSP);

  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
   Ins_LuzLog($$PLSQL_UNIT,(to_char(pStreetTypeCode)  ||  q'[,  ']'  ||  pStreetTypeName  ||
q']'), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
   Ins_LuzLog($$PLSQL_UNIT,(to_char(pStreetTypeCode)  ||  q'[,  ']'  ||  pStreetTypeName  ||
q']'), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento UPD_ADDRESS actualiza los datos de la tabla ADDRESS cuyo ID y datos se pasan como parámetros.

```
/*****************************************************************************************
***************************************/
CREATE OR REPLACE PROCEDURE Upd_Address (
/*********************************************
|  Update City data in City table
|
| In Params:    streetCode, streetName, adddressNumber, floor, doorNumber, phoneNumber,
cityCode
| Out Params:  RSP, addressCode
| Date:
| Programmer: AJCC
|*********************************************
*/
    pAddressCode    IN  Address.addressCode%TYPE
  , pStreetCode     IN  Address.streetCode%TYPE
  , pStreetName     IN  Address.streetName%TYPE
  , pAdddressNumber IN  Address.adddressNumber%TYPE
  , pFloor          IN  Address.floor%TYPE
  , pDoorNumber     IN  Address.doorNumber%TYPE
  , pPhoneNumber    IN  Address.phoneNumber%TYPE
  , pCityCode       IN  Address.cityCode%TYPE
  , RSP             OUT VARCHAR2

)
AS
```

```
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
BEGIN

  /* Validation Controls */

  --Check null values
  IF pAddressCode IS NULL THEN
    lErrorDescrip:=q'[AddressCode can't be null]';
    RAISE lException;
  END IF;

  IF pStreetCode IS NULL THEN
    lErrorDescrip:=q'[StreetCode can't be null]';
    RAISE lException;
  END IF;

  IF pStreetName IS NULL THEN
    lErrorDescrip:=q'[StreetName can't be null]';
    RAISE lException;
  END IF;

  IF pAdddressNumber IS NULL THEN
    lErrorDescrip:=q'[AdddressNumber can't be null]';
    RAISE lException;
  END IF;

  IF pCityCode IS NULL THEN
    lErrorDescrip:=q'[CityCode can't be null]';
    RAISE lException;
  END IF;


  --Check datatypes: Number values, quotes in strings, lenght...
  IF LENGTH(pStreetName)>200 THEN
    lErrorDescrip:='StreetName too long, >200';
    RAISE lException;
  END IF;

  IF LENGTH(pAdddressNumber)>10 THEN
    lErrorDescrip:='AdddressNumber too long, >10';
    RAISE lException;
  END IF;

  IF LENGTH(pFloor)>10 THEN
    lErrorDescrip:='Floor too long, >10';
    RAISE lException;
  END IF;

  IF LENGTH(pDoorNumber)>10 THEN
    lErrorDescrip:='DoorNumber too long, >10';
    RAISE lException;
  END IF;

  IF LENGTH(pPhoneNumber)>9 THEN
    lErrorDescrip:='PhoneNumber too long, >9';
    RAISE lException;
  END IF;

  -- Foreign Keys validation -->The system can do itself

  --Update the Address
  UPDATE Address SET
    STREETCODE=pStreetCode
    , STREETNAME=pStreetName
    , adddressNumber=pAdddressNumber
    , floor=pFloor
    , DOORNUMBER=pDoorNumber
    , PHONENUMBER=pPhoneNumber
    , CITYCODE=pCityCode
  WHERE ADDRESSCODE=pAddressCode;

  RSP:='OK';

  --Track Store Procedure Result
  Ins_LuzLog(
```

```
      $$PLSQL_UNIT
      ,(  to_char(pAddressCode)  ||  q'[,  ]'  ||  to_char(pStreetCode)  ||  q'[,  ']'  ||
pStreetName  ||  q'[', ']'  ||  pAdddressNumber  ||  q'[', ']'  ||  pFloor  ||  q'[', ']'  ||
pDoorNumber || q'[', ']' || pPhoneNumber || q'[', ]' || TO_CHAR(pCityCode))
      , RSP, lRSP);


  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
   Ins_LuzLog(
      $$PLSQL_UNIT
      ,(  to_char(pAddressCode)  ||  q'[,  ]'  ||  to_char(pStreetCode)  ||  q'[,  ']'  ||
pStreetName  ||  q'[', ']'  ||  pAdddressNumber  ||  q'[', ']'  ||  pFloor  ||  q'[', ']'  ||
pDoorNumber || q'[', ']' || pPhoneNumber || q'[', ]' || TO_CHAR(pCityCode))
      , RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
   Ins_LuzLog(
      $$PLSQL_UNIT
      ,(  to_char(pAddressCode)  ||  q'[,  ]'  ||  to_char(pStreetCode)  ||  q'[,  ']'  ||
pStreetName  ||  q'[', ']'  ||  pAdddressNumber  ||  q'[', ']'  ||  pFloor  ||  q'[', ']'  ||
pDoorNumber || q'[', ']' || pPhoneNumber || q'[', ]' || TO_CHAR(pCityCode))
      , RSP, lRSP);


    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento UPD_BANK actualiza los datos de la tabla BANK cuyo ID y descripción se pasa como parámetro.

```
/*****************************************************************************************
***************************************/
CREATE OR REPLACE PROCEDURE Upd_Bank (
/*********************************************
|  Update Bank data in Bank table
|
| In Params:   bankCode, bankName
| Out Params:  RSP
| Date:
| Programmer: AJCC
|*********************************************
*/
    pBankCode     IN  Bank.bankCode%TYPE
  , pBankName     IN  Bank.bankName%TYPE
  , RSP           OUT VARCHAR2
)
IS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
BEGIN

  /* Validation Controls */

  --Check null values
  IF pBankCode IS NULL THEN
    lErrorDescrip:=q'[BankCode can't be null]';
    RAISE lException;
  END IF;

  IF pBankName IS NULL THEN
    lErrorDescrip:=q'[BankName can't be null]';
    RAISE lException;
  END IF;
```

```
  --Check datatypes: Number values, quotes in strings, lenght...
  IF LENGTH(pBankCode)>11 THEN
    lErrorDescrip:='BankCode too long, >11';
    RAISE lException;
  END IF;

  IF LENGTH(pBankName)>100 THEN
    lErrorDescrip:='BankName too long, >100';
    RAISE lException;
  END IF;


  --Update the Bank
  UPDATE BANK SET BANKNAME=pBankName WHERE BANKCODE=pBankCode;
  RSP:='OK';

  --Track Store Procedure Result
  Ins_LuzLog($$PLSQL_UNIT,( q'[']' || to_char(pBankCode) ||  q'[', ']' || pBankName ||
q'[']'), RSP, lRSP);

  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
    Ins_LuzLog($$PLSQL_UNIT,( q'[']' || to_char(pBankCode) ||  q'[', ']' || pBankName ||
q'[']'), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    Ins_LuzLog($$PLSQL_UNIT,( q'[']' || to_char(pBankCode) ||  q'[', ']' || pBankName ||
q'[']'), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento UPD_IDENTITYTYPE actualiza los datos de la tabla IDENTITYTYPE cuyo ID y descripción se pasa como parámetro.

```
/*****************************************************************************************
****************************************/
CREATE OR REPLACE PROCEDURE Upd_IdentityType (
/*********************************************
|  Update IdentityType data in IdentityType table
|
| In Params:   identityCode, identityType
| Out Params:  RSP
| Date:
| Programmer: AJCC
|*********************************************
*/
    pIdentityCode IN  IdentityType.IdentityType%TYPE
  , pIdentityType IN  IdentityType.IdentityType%TYPE
  , RSP           OUT VARCHAR2
)
AS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
BEGIN

  /* Validation Controls */

  --Check null values
  IF pIdentityCode IS NULL THEN
    lErrorDescrip:=q'[IdentityCode can't be null]';
    RAISE lException;
  END IF;
```

```
  IF pIdentityType IS NULL THEN
    lErrorDescrip:=q'[IdentityType can't be null]';
    RAISE lException;
  END IF;

  --Check datatypes: Number values, quotes in strings, lenght...
  IF LENGTH(pIdentityType)>10 THEN
    lErrorDescrip:='IdentityType too long, >10';
    RAISE lException;
  END IF;

  --Update the IdentityType
  UPDATE IdentityType SET IdentityType=pIdentityType WHERE IdentityCode=pIdentityCode;
  RSP:='OK';

  --Track Store Procedure Result
  Ins_LuzLog($$PLSQL_UNIT,(to_char(pIdentityCode) || q'[, ']' || pIdentityType || q'[']'),
RSP, lRSP);

  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
   Ins_LuzLog($$PLSQL_UNIT,(to_char(pIdentityCode)  ||  q'[,  ']'  ||  pIdentityType  ||
q'[']'), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    Ins_LuzLog($$PLSQL_UNIT,(to_char(pIdentityCode)  ||  q'[,  ']'  ||  pIdentityType  ||
q'[']'), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento UPD_CONSUMER actualiza los datos de la tabla CONSUMER cuyo ID y datos se pasan como parámetros.

```
/*****************************************************************************************
***************************************/
CREATE OR REPLACE PROCEDURE Upd_Consumer (
/*********************************************
|  Update Consumer data in Consumer table
|
| In Params:   consumerCode, consumerName, consumerSurname, sex, mobilePhone, addressCode,
identityCode, identificationNumber, bankCode, accountCode
| Out Params:  RSP
| Date:
| Programmer: AJCC
|*********************************************
*/
    pConsumerCode         IN  Consumer.consumerCode%TYPE
  , pConsumerName         IN  Consumer.consumerName%TYPE
  , pConsumerSurname      IN  Consumer.consumerSurname%TYPE
  , pSex                  IN  Consumer.sex%TYPE
  , pMobilePhone          IN  Consumer.mobilePhone%TYPE
  , pAddressCode          IN  Consumer.addressCode%TYPE
  , pIdentityCode         IN  Consumer.identityCode%TYPE
  , pIdentificationNumber IN  Consumer.identificationNumber%TYPE
  , pBankCode             IN  Consumer.bankCode%TYPE
  , pAccountCode          IN  Consumer.accountCode%TYPE
  , RSP                   OUT VARCHAR2
)
IS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
  lInParam      VARCHAR2(2000);
```

```
BEGIN

  /* Validation Controls */

  --Check null values
  IF pConsumerCode IS NULL THEN
    lErrorDescrip:=q'[ConsumerCode can't be null]';
    RAISE lException;
  END IF;

  IF pConsumerName IS NULL THEN
    lErrorDescrip:=q'[ConsumerName can't be null]';
    RAISE lException;
  END IF;

  IF pConsumerSurname IS NULL THEN
    lErrorDescrip:=q'[ConsumerSurname can't be null]';
    RAISE lException;
  END IF;

  IF pSex IS NULL THEN
    lErrorDescrip:=q'[Sex can't be null]';
    RAISE lException;
  END IF;

  IF pAddressCode IS NULL THEN
    lErrorDescrip:=q'[AddressCode can't be null]';
    RAISE lException;
  END IF;

  IF pIdentityCode IS NULL THEN
    lErrorDescrip:=q'[IdentityCode can't be null]';
    RAISE lException;
  END IF;

  IF pIdentificationNumber IS NULL THEN
    lErrorDescrip:=q'[IdentificationNumber can't be null]';
    RAISE lException;
  END IF;

  IF pBankCode IS NULL THEN
    lErrorDescrip:=q'[BankCode can't be null]';
    RAISE lException;
  END IF;

  IF pAccountCode IS NULL THEN
    lErrorDescrip:=q'[AccountCode can't be null]';
    RAISE lException;
  END IF;

  --Check datatypes: Number values, quotes in strings, lenght...
  IF LENGTH(pConsumerName)>100 THEN
    lErrorDescrip:='ConsumerName too long, >100';
    RAISE lException;
  END IF;

  IF LENGTH(pConsumerSurname)>100 THEN
    lErrorDescrip:='ConsumerSurname too long, >100';
    RAISE lException;
  END IF;

  IF pSex NOT IN ('M','F') THEN
    lErrorDescrip:='Sex must be M or F';
    RAISE lException;
  END IF;

  IF LENGTH(pMobilePhone)>9 THEN
    lErrorDescrip:='MobilePhone too long, >9';
    RAISE lException;
  END IF;

  IF LENGTH(pIdentificationNumber)>50 THEN
    lErrorDescrip:='IdentificationNumber too long, >50';
    RAISE lException;
  END IF;

  IF LENGTH(pBankCode)>11 THEN
```

```
    lErrorDescrip:='BankCode too long, >9';
    RAISE lException;
  END IF;

  IF LENGTH(pAccountCode)>24 THEN
    lErrorDescrip:='AccountCode too long, >24';
    RAISE lException;
  END IF;

  -- Foreign Keys validation -->The system can do itself


  --Update Consumer
  UPDATE Consumer SET
    consumerName=pConsumerName
    , consumerSurname=pConsumerSurname
    , sex=pSex
    , mobilePhone=pMobilePhone
    , addressCode=pAddressCode
    , identityCode=pIdentityCode
    , identificationNumber=pIdentificationNumber
    , bankCode=pBankCode
    , accountCode= pAccountCode
  WHERE ConsumerCode=pConsumerCode;

  RSP:='OK';

  --Track Store Procedure Result
  LInParam:=to_char(pConsumerCode)  ||  q'[,  ']'  ||  pConsumerName  ||  q'[',  ']'  ||
pConsumerSurname  ||  q'[',  ']'||  pSex  ||  q'[',  ']'  ||  pMobilePhone  ||  q'[',  ]'  ||
to_char(pAddressCode)
                                  || q'[, ]' || to_char(pIdentityCode) || q'[, ']' ||
pIdentificationNumber || q'[', ']' || pBankCode || q'[', ']' || pAccountCode || q'[']';

  Ins_LuzLog($$PLSQL_UNIT, LInParam , RSP, lRSP);

  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
    LInParam:=to_char(pConsumerCode)  ||  q'[,  ']'  ||  pConsumerName  ||  q'[',  ']'  ||
pConsumerSurname  ||  q'[',  ']'||  pSex  ||  q'[',  ']'  ||  pMobilePhone  ||  q'[',  ]'  ||
to_char(pAddressCode)
                                  || q'[, ]' || to_char(pIdentityCode) || q'[, ']' ||
pIdentificationNumber || q'[', ']' || pBankCode || q'[', ']' || pAccountCode || q'[']';

    Ins_LuzLog($$PLSQL_UNIT, LInParam , RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    LInParam:=to_char(pConsumerCode)  ||  q'[,  ']'  ||  pConsumerName  ||  q'[',  ']'  ||
pConsumerSurname  ||  q'[',  ']'||  pSex  ||  q'[',  ']'  ||  pMobilePhone  ||  q'[',  ]'  ||
to_char(pAddressCode)
                                  || q'[, ]' || to_char(pIdentityCode) || q'[, ']' ||
pIdentificationNumber || q'[', ']' || pBankCode || q'[', ']' || pAccountCode || q'[']';

    Ins_LuzLog($$PLSQL_UNIT, LInParam , RSP, lRSP);
    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento UPD_COMPANY actualiza los datos de la tabla COMPANY cuyo ID y datos se pasan como parámetros.

```
/*****************************************************************************************
************************************/
CREATE OR REPLACE PROCEDURE Upd_Company (
/*******************************************
| Update Company data in Company table
```

```
|
| In Params:   companyTaxCode, companyName, addressCode
| Out Params:  RSP
| Date:
| Programmer: AJCC
|*********************************************
*/
    pCompanyTaxCode IN  Company.companyTaxCode%TYPE
  , pCompanyName    IN  Company.companyName%TYPE
  , pAddressCode    IN  Company.addressCode%TYPE
  , RSP             OUT VARCHAR2
)
AS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
BEGIN

  /* Validation Controls */

  --Check null values
  IF pCompanyTaxCode IS NULL THEN
    lErrorDescrip:=q'[CompanyTaxCode can't be null]';
    RAISE lException;
  END IF;

  IF pCompanyName IS NULL THEN
    lErrorDescrip:=q'[CompanyName can't be null]';
    RAISE lException;
  END IF;

  IF pAddressCode IS NULL THEN
    lErrorDescrip:=q'[AddressCode can't be null]';
    RAISE lException;
  END IF;

  --Check datatypes: Number values, quotes in strings, lenght...
  IF LENGTH(pCompanyTaxCode)>15 THEN
    lErrorDescrip:='CompanyTaxCode too long, >15';
    RAISE lException;
  END IF;

  IF LENGTH(pCompanyName)>200 THEN
    lErrorDescrip:='CompanyName too long, >200';
    RAISE lException;
  END IF;

  -- Foreign Keys validation, the system can do itself

  --Update the Company
  UPDATE Company SET
    companyName=pCompanyName
    , addressCode=pAddressCode
  WHERE companyTaxCode=pCompanyTaxCode;
  RSP:='OK';

  --Track Store Procedure Result
  Ins_LuzLog($$PLSQL_UNIT, q'[']' ||pCompanyTaxCode || q'[', ']' || pCompanyName || q'[',
]' || to_char(pAddressCode), RSP, lRSP);

  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
    Ins_LuzLog($$PLSQL_UNIT, q'[']' ||pCompanyTaxCode || q'[', ']' || pCompanyName ||
q'[', ]' || to_char(pAddressCode), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    Ins_LuzLog($$PLSQL_UNIT, q'[']' ||pCompanyTaxCode || q'[', ']' || pCompanyName ||
q'[', ]' || to_char(pAddressCode), RSP, lRSP);
```

```
    DBMS_OUTPUT.PUT_LINE (RSP);
END;

/*****************************************************************************************
**************************************/
CREATE OR REPLACE PROCEDURE Upd_Meter (
/*********************************************
| Update Meter data in Meter table
|
|   In   Params:          serialNumber,   meterModel,   contractCode,   contractedPower,
lastTechnicalInspection, installationDate, companyCode, consumerCode, addressCode
| Out Params:  RSP
| Date:
| Programmer: AJCC
|*********************************************
*/
    pSerialNumber             IN  Meter.serialNumber%TYPE
  , pMeterModel               IN  Meter.meterModel%TYPE
  , pContractCode             IN  Meter.contractCode%TYPE
  , pContractedPower          IN  Meter.contractedPower%TYPE
  , pLastTechnicalInspection  IN  Meter.lastTechnicalInspection%TYPE
  , pInstallationDate         IN  Meter.installationDate%TYPE
  , pCompanyCode              IN  Meter.companyCode%TYPE
  , pConsumerCode             IN  Meter.consumerCode%TYPE
  , pAddressCode              IN  Meter.addressCode%TYPE
  , RSP                       OUT VARCHAR2

)
IS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
  lInparam      VARCHAR2(2000);
BEGIN

  /* Validation Controls */

    --Check null values
  IF pSerialNumber IS NULL THEN
    lErrorDescrip:=q'[SerialNumber can't be null]';
    RAISE lException;
  END IF;

  IF pMeterModel IS NULL THEN
    lErrorDescrip:=q'[MeterMode can't be null]';
    RAISE lException;
  END IF;

  IF pContractedPower IS NULL THEN
    lErrorDescrip:=q'[contractedPower can't be null]';
    RAISE lException;
  END IF;

  IF pInstallationDate IS NULL THEN
    lErrorDescrip:=q'[InstallationDate can't be null]';
    RAISE lException;
  END IF;

  IF pCompanyCode IS NULL THEN
    lErrorDescrip:=q'[companyCode can't be null]';
    RAISE lException;
  END IF;

  IF pAddressCode IS NULL THEN
    lErrorDescrip:=q'[AddressCode can't be null]';
    RAISE lException;
  END IF;

  --Check datatypes: Number values, quotes in strings, lenght...
  IF LENGTH(pSerialNumber)>20 THEN
    lErrorDescrip:='SerialNumber too long, >20';
    RAISE lException;
  END IF;

  IF LENGTH(pMeterModel)>100 THEN
    lErrorDescrip:='MeterModel too long, >100';
```

```
    RAISE lException;
  END IF;

  IF LENGTH(pContractCode)>100 THEN
    lErrorDescrip:='ContractCode too long, >10';
    RAISE lException;
  END IF;

  IF pInstallationDate>pLastTechnicalInspection THEN
    lErrorDescrip:='InstallationDate before LastTechnicalInspection?';
    RAISE lException;
  END IF;

  IF LENGTH(pCompanyCode)>15 THEN
    lErrorDescrip:='CompanyCode too long, >15';
    RAISE lException;
  END IF;

  IF LENGTH(pCompanyCode)>15 THEN
    lErrorDescrip:='CompanyCode too long, >15';
    RAISE lException;
  END IF;

  -- Foreign Keys validation -->The system can do itself

  --Update the Address
  UPDATE METER SET
    METERMODEL= pMETERMODEL
    , CONTRACTCODE= pCONTRACTCODE
    , CONTRACTEDPOWER= pCONTRACTEDPOWER
    , LASTTECHNICALINSPECTION= pLASTTECHNICALINSPECTION
    , INSTALLATIONDATE= pINSTALLATIONDATE
    , COMPANYCODE= pCOMPANYCODE
    , CONSUMERCODE= pCONSUMERCODE
    , ADDRESSCODE= pADDRESSCODE
  WHERE
    SERIALNUMBER= pSERIALNUMBER;

  RSP:='OK';

  --Track Store Procedure Result
  lInparam:= q'[']' || pSERIALNUMBER || q'[', ']' || pMETERMODEL || q'[', ']' ||
pCONTRACTCODE || q'[', ]' || to_char(pCONTRACTEDPOWER) || q'[, ']'
                    || to_char(pLASTTECHNICALINSPECTION, 'yyyy/mm/dd')   || q'[', ']' ||
to_char(pINSTALLATIONDATE, 'yyyy/mm/dd') || q'[', ']'
                    || pCOMPANYCODE || q'[', ]' || to_char(pCONSUMERCODE) || q'[, ]'  ||
to_char(pADDRESSCODE);

  Ins_LuzLog($$PLSQL_UNIT, lInparam, RSP, lRSP);

  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
    lInparam:= q'[']' || pSERIALNUMBER || q'[', ']' || pMETERMODEL || q'[', ']' ||
pCONTRACTCODE || q'[', ]' || to_char(pCONTRACTEDPOWER) || q'[, ']'
                    || to_char(pLASTTECHNICALINSPECTION, 'yyyy/mm/dd')   || q'[', ']'
|| to_char(pINSTALLATIONDATE, 'yyyy/mm/dd') || q'[', ']'
                    || pCOMPANYCODE || q'[', ]' || to_char(pCONSUMERCODE) || q'[, ]'  ||
to_char(pADDRESSCODE);
    Ins_LuzLog($$PLSQL_UNIT, lInparam, RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    lInparam:= q'[']' || pSERIALNUMBER || q'[', ']' || pMETERMODEL || q'[', ']' ||
pCONTRACTCODE || q'[', ]' || to_char(pCONTRACTEDPOWER) || q'[, ']'
                    || to_char(pLASTTECHNICALINSPECTION, 'yyyy/mm/dd')   || q'[', ']'
|| to_char(pINSTALLATIONDATE, 'yyyy/mm/dd') || q'[', ']'
                    || pCOMPANYCODE || q'[', ]' || to_char(pCONSUMERCODE) || q'[, ]'  ||
to_char(pADDRESSCODE);
    Ins_LuzLog($$PLSQL_UNIT, lInparam, RSP, lRSP);
```

```
    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento UPD_CONNECTION actualiza los datos de la tabla CONNECTION cuyo ID y datos se pasan como parámetros.

```
/*****************************************************************************************
**************************************/
CREATE OR REPLACE PROCEDURE Upd_Connection (
/*********************************************
| Update Connection data in Connection table
|
| In Params:   readingDate, meterSerialNumber, instantConsumption, isSuccess
| Out Params:  RSP
| Date:
| Programmer: AJCC
|*********************************************
*/
    pReadingDate        IN  Connection.readingDate%TYPE
  , pMeterSerialNumber  IN  Connection.meterSerialNumber%TYPE
  , pInstantConsumption IN  Connection.instantConsumption%TYPE
  , pisSuccess          IN  Connection.isSuccess%TYPE
  , RSP                 OUT VARCHAR2
)
IS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
  lInparam      VARCHAR2(2000);
BEGIN

  /* Validation Controls */

  --Check null values
  IF pReadingDate IS NULL THEN
    lErrorDescrip:=q'[ReadingDate can't be null]';
    RAISE lException;
  END IF;

  IF pMeterSerialNumber IS NULL THEN
    lErrorDescrip:=q'[MeterSerialNumber can't be null]';
    RAISE lException;
  END IF;

  IF pInstantConsumption IS NULL THEN
    lErrorDescrip:=q'[InstantConsumption can't be null]';
    RAISE lException;
  END IF;

  IF pIsSuccess IS NULL THEN
    lErrorDescrip:=q'[IsSuccess can't be null]';
    RAISE lException;
  END IF;

  --Check datatypes: Number values, quotes in strings, lenght...
  IF LENGTH(pMeterSerialNumber)>20 THEN
    lErrorDescrip:='meterSerialNumber too long, >20';
    RAISE lException;
  END IF;

  IF pIsSuccess NOT IN ('Y','N') THEN
    lErrorDescrip:='isSuccess must be Y or N';
    RAISE lException;
  END IF;

  -- Foreign Keys validation -->The system can do itself

  --Update  the Connection
  UPDATE Connection SET
    instantConsumption=pInstantConsumption
    , isSuccess=pIsSuccess
  WHERE
    readingDate=pReadingDate
```

```
    AND meterSerialNumber=pMeterSerialNumber;
  RSP:='OK';

  --Track Store Procedure Result
  lInparam:= q'[']' || to_char(preadingDate, 'DD/MM/YYYY HH24:MI:SS')  || q'[', ']' ||
pMeterSerialNumber || q'[', ]' || 'XXXX' || q'[, ]' || 'X';
  Ins_LuzLog($$PLSQL_UNIT,lInparam, RSP, lRSP);

  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
    lInparam:= q'[']' || to_char(preadingDate, 'DD/MM/YYYY HH24:MI:SS')  || q'[', ']' ||
pMeterSerialNumber || q'[', ]' || 'XXXX' || q'[, ]' || 'X';
    Ins_LuzLog($$PLSQL_UNIT,lInparam, RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    lInparam:= q'[']' || to_char(preadingDate, 'DD/MM/YYYY HH24:MI:SS')  || q'[', ']' ||
pMeterSerialNumber || q'[', ]' || 'XXXX' || q'[, ]' || 'X';
    Ins_LuzLog($$PLSQL_UNIT,lInparam, RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento UPD_PRICE actualiza los datos de la tabla PRICE cuyos ID's se pasan como parámetros.

```
/*****************************************************************************************
**************************************/
CREATE OR REPLACE PROCEDURE Upd_Price (
/*********************************************
| Update Price data in Price table
|
| In Params:   changeData, countryCode, companyCode, newPrice
| Out Params:  RSP
| Date:
| Programmer: AJCC
|*********************************************
*/
    pchangeData  IN  Price.changeData%TYPE
  , pcountryCode IN  Price.countryCode%TYPE
  , pcompanyCode IN  Price.companyCode%TYPE
  , pnewPrice    IN  Price.newPrice%TYPE
  , RSP          OUT VARCHAR2

)
IS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
  lInparam      VARCHAR2(2000);
BEGIN

  /* Validation Controls */

  --Check null values
  IF pChangeData IS NULL THEN
    lErrorDescrip:=q'[ChangeData can't be null]';
    RAISE lException;
  END IF;

  IF pCountryCode IS NULL THEN
    lErrorDescrip:=q'[CountryCode can't be null]';
    RAISE lException;
  END IF;

  IF pCompanyCode IS NULL THEN
```

```
    lErrorDescrip:=q'[CompanyCode can't be null]';
    RAISE lException;
  END IF;

  IF pNewPrice IS NULL THEN
    lErrorDescrip:=q'[NewPrice can't be null]';
    RAISE lException;
  END IF;


  --Check datatypes: Number values, quotes in strings, lenght...
  IF pchangeData<SYSDATE THEN
    lErrorDescrip:='ChangeData before Now???';
    RAISE lException;
  END IF;

  IF LENGTH(pCompanyCode)>15 THEN
    lErrorDescrip:='CompanyCode too long, >15';
    RAISE lException;
  END IF;

  IF pNewPrice<0 THEN
    lErrorDescrip:=q'[NewPrice must be greather than 0]';
    RAISE lException;
  END IF;

  -- Foreign Keys validation -->The system can do itself

  --Update the Address
  UPDATE PRICE SET
    NEWPRICE = pNEWPRICE
  WHERE
    CHANGEDATA = pCHANGEDATA
    AND COUNTRYCODE  = pCOUNTRYCODE
    AND COMPANYCODE  = pCOMPANYCODE;

  RSP:='OK';

  --Track Store Procedure Result
  Ins_LuzLog($$PLSQL_UNIT, q'[']' || to_char(pChangeData, 'yyyy/mm/dd')  || q'[', ]' ||
to_char(pCountryCode)    ||    q'[,    ']'    ||    pCompanyCode    ||    q'[',    ]'||
to_char(pNewPrice,'9.999999'), RSP, lRSP);

  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
    Ins_LuzLog($$PLSQL_UNIT, q'[']' || to_char(pChangeData, 'yyyy/mm/dd')  || q'[', ]' ||
to_char(pCountryCode)    ||    q'[,    ']'    ||    pCompanyCode    ||    q'[',    ]'||
to_char(pNewPrice,'9.999999'), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);

  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    Ins_LuzLog($$PLSQL_UNIT, q'[']' || to_char(pChangeData, 'yyyy/mm/dd')  || q'[', ]' ||
to_char(pCountryCode)    ||    q'[,    ']'    ||    pCompanyCode    ||    q'[',    ]'||
to_char(pNewPrice,'9.999999'), RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```

El procedimiento UPD_CLIENTS actualiza los datos de la tabla CLIENTS cuyo ID y datos se pasan como parámetros.

```
/*****************************************************************************************
************************************/
CREATE OR REPLACE PROCEDURE Upd_Clients (
/*******************************************
| Update Client data in Clients table
```

```
|
| In Params:   companyCode, consumerCode
| Out Params:  RSP
| Date:
| Programmer: AJCC
|*********************************************
*/
    pHireDate     IN  Clients.hireDate%TYPE
  , pCompanyCode  IN  Clients.companyCode%TYPE
  , pConsumerCode IN  Clients.consumerCode%TYPE
  , RSP           OUT VARCHAR2
)
IS
  lException    EXCEPTION;
  lErrorDescrip VARCHAR2(100);
  lRSP          VARCHAR2(2000);
  lInparam      VARCHAR2(2000);
BEGIN

  /* Validation Controls */

  --Check null values
  IF pHireDate IS NULL THEN
    lErrorDescrip:=q'[HireDate can't be null]';
    RAISE lException;
  END IF;

  IF pCompanyCode IS NULL THEN
    lErrorDescrip:=q'[CompanyCode can't be null]';
    RAISE lException;
  END IF;

  IF pConsumerCode IS NULL THEN
    lErrorDescrip:=q'[ConsumerCode can't be null]';
    RAISE lException;
  END IF;

  --Check datatypes: Number values, quotes in strings, lenght...
  IF pHireDate< SYSDATE THEN
    lErrorDescrip:='HireDate before NOW???';
    RAISE lException;
  END IF;

  IF LENGTH(pCompanyCode)>15 THEN
    lErrorDescrip:='CompanyCode too long, >15';
    RAISE lException;
  END IF;

  -- Foreign Keys validation -->The system can do itself

  --Update the Clients
  UPDATE CLIENTS SET
    HIREDATE=pHIREDATE
  WHERE
    CONSUMERCODE=pCONSUMERCODE
    AND COMPANYCODE=pCOMPANYCODE;

  RSP:='OK';

  --Track Store Procedure Result
  lInparam:=   q'['   ||   to_char(pHireDate,   'DD/MM/YYYY')      ||   q'[',   ]'   ||
to_char(pConsumerCode) || q'[, ']' || pCompanyCode || q'[']' ;
  Ins_LuzLog($$PLSQL_UNIT, lInparam, RSP, lRSP);

  DBMS_OUTPUT.PUT_LINE (RSP);

EXCEPTION

  WHEN lException THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || lErrorDescrip;
   lInparam:=   q'['   ||   to_char(pHireDate,   'DD/MM/YYYY')      ||   q'[',   ]'   ||
to_char(pConsumerCode) || q'[, ']' || pCompanyCode || q'[']' ;
    Ins_LuzLog($$PLSQL_UNIT, lInparam, RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
```

```
  WHEN OTHERS THEN
    --Track Store Procedure Result
    RSP:='ERROR: ' || sqlerrm;
    lInparam:=   q'['']'   ||   to_char(pHireDate,   'DD/MM/YYYY')      ||   q'[',   ]'   ||
to_char(pConsumerCode) || q'[, ']' || pCompanyCode || q'['']' ;
    Ins_LuzLog($$PLSQL_UNIT, lInparam, RSP, lRSP);

    DBMS_OUTPUT.PUT_LINE (RSP);
END;
```