Anexo I. Creación de las tablas de la BBDD del sistema operacional

Script para la creación de la tabla Country

La tabla COUNTRY almacena los datos de los países: El código de país y el nombre.

```
CREATE TABLE Country (
       CountryCode INTEGER CONSTRAINT PK_countryCode_Country PRIMARY KEY
       , countryName VARCHAR2 (100 CHAR) CONSTRAINT NN_countryName NOT NULL
       , CONSTRAINT UN_CountryName UNIQUE (countryName)
);
```

Script para la creación de la tabla Province (secuencia, trigger para asignación de la clave sintética y definición de la tabla)

La tabla PROVINCE almacena las provincias o regiones. Está formada por un código de provincial, una descripción o nombre de la provincial, y el código del país al que pertenece.

```
CREATE TABLE Province (
       provinceId INTEGER CONSTRAINT PK_Province PRIMARY KEY
       , provinceCode VARCHAR2(10) CONSTRAINT NN_province_provinceCode NOT NULL
       , provinceName VARCHAR2 (100 CHAR) CONSTRAINT NN_provinceName NOT NULL
       , countryCode INTEGER CONSTRAINT NN_CountryCode_Province NOT NULL
              CONSTRAINT FK_Country_Province REFERENCES Country (CountryCode)
       , CONSTRAINT UN_provinceName UNIQUE (provinceName)
       , CONSTRAINT UN_ProvinceCode_CountryCode UNIQUE (provinceCode, countryCode)
);

CREATE SEQUENCE seq_Province INCREMENT BY 1 START WITH 1;

CREATE OR REPLACE TRIGGER Insert_provinceId_Province
  BEFORE INSERT ON Province
  FOR EACH ROW
BEGIN
  SELECT seq_Province.NEXTVAL INTO :NEW.provinceId FROM DUAL;
END Insert_provinceId_Province;
```

Script para la creación de la tabla City

La tabla CITY almacena las ciudades. Los atributos son un código, el nombre de la ciudad, un código postal o ZIPCODE, y el código de la provincia o región a la que pertenece.

```
CREATE TABLE City (
       cityCode INTEGER CONSTRAINT pk_cityCode_City PRIMARY KEY
       , cityName VARCHAR2 (100) CONSTRAINT nn_cityName NOT NULL
       , zipCode VARCHAR2 (15) CONSTRAINT nn_zipcode NOT NULL
       , CONSTRAINT UN_City UNIQUE (cityCode, cityName)
       , provinceCode INTEGER
              CONSTRAINT FK_City_Province REFERENCES Province (provinceCode)
);
```

Script para la creación de la tabla StreetType

La tabla STREETYPE define el tipo de vía. Es un descriptor al estilo de "CL" para definir calle, "AV" para avenida… Está compuesto por un código y un descriptor de texto.

```
CREATE TABLE StreetType (
       streetTypeCode INTEGER CONSTRAINT PK_streetTypeCode_StreetType PRIMARY KEY
       , streetTypeName VARCHAR2 (2 CHAR) CONSTRAINT NN_streetTypeName NOT NULL
       , CONSTRAINT UN_streetTypeName UNIQUE (streetTypeName)
);
```

Script para la creación de la tabla Address (secuencia, trigger para asignación de la clave sintética y definición de la tabla)

La tabla ADDRESS almacena direcciones. Cada dirección está compuesta por un código, un código de tipo de vía (STREETTYPE), un nombre de la vía, un número de vía, un número de piso o planta, un número de puerta, un teléfono, y un código de ciudad.

```
CREATE TABLE Address (
       addressCode INTEGER CONSTRAINT PK_addressCode_Address PRIMARY KEY
       , streetCode INTEGER CONSTRAINT NN_streetCode NOT NULL CONSTRAINT FK_streetType
       REFERENCES streetType(StreetTypeCode)
       , streetName VARCHAR2 (200 CHAR) CONSTRAINT NN_streetName NOT NULL
       , addressNumber VARCHAR2 (10 CHAR) CONSTRAINT NN_number NOT NULL
       , floor VARCHAR2 (10 CHAR)
       , doorNumber VARCHAR2 (10 CHAR)
       , phoneNumber VARCHAR2 (12 CHAR)
       , cityCode INTEGER CONSTRAINT NN_cityCode NOT NULL CONSTRAINT FK_City REFERENCES
       City (cityCode)
);

CREATE SEQUENCE seq_Address INCREMENT BY 1 START WITH 1;

CREATE OR REPLACE
TRIGGER Insert_AddressCode_Address
       BEFORE INSERT ON Address
       FOR EACH ROW
BEGIN
       SELECT seq_Address.NEXTVAL INTO :NEW.AddressCode FROM DUAL;
END Insert_AddressCode_Address;
```

Script para la creación de la tabla Bank

El único objetivo de la tabla BANK es almacenar los códigos de banco y el nombre del banco con el que los consumidores pagan el suministro.

```
CREATE TABLE Bank (
       bankCode_INTEGER CONSTRAINT PK_bankCode_Bank PRIMARY KEY
       , bankName VARCHAR2 (100 CHAR) CONSTRAINT NN_bankName NOT NULL
       , CONSTRAINT UN_bankName UNIQUE (bankName)
);
```

Script para la creación de la tabla IdentityType

La tabla IDENTITYTYPE almacena los tipos de identificación que se pueden dar en el sistema. Un tipo de identificación se define como un código numérico y un descriptor de texto.

```
CREATE TABLE IdentityType (
       IdentityCode INTEGER CONSTRAINT PK_identityCode_IdentityType PRIMARY KEY
```

```
        , identityType VARCHAR2 (10 CHAR) CONSTRAINT NN_identityType NOT NULL
        , CONSTRAINT UN_identityType UNIQUE (identityType)
);
```

Script para la creación de la tabla Consumer (secuencia, trigger para asignación de la clave sintética y definición de la tabla)

La tabla CONSUMER almacena los datos de los consumidores del suministro energético. Almacena un código de consumidor , el nombre y apellidos del consumidor, un teléfono móvil (nótese que el teléfono móvil se asocia al consumidor y el teléfono fijo a una dirección en la tabla ADDRESS), un código de dirección, un tipo de identificación (DNI, NNIF, carnet de conducir, pasaporte…), el número (o cadena) de identificación, un código de banco que define una entidad bancaria, y un número de cuenta bancario.

```
CREATE SEQUENCE seq_Consumer INCREMENT BY 1 START WITH 1;

CREATE OR REPLACE
TRIGGER Insert_consumerCode_Consumer
        BEFORE INSERT ON Consumer
        FOR EACH ROW
BEGIN
        SELECT seq_Consumer.NEXTVAL INTO :NEW.ConsumerCode FROM DUAL;
END Insert_consumerCode_Consumer;


CREATE TABLE Consumer (
        consumerCode INTEGER CONSTRAINT PK_consumerCode_Consumer PRIMARY KEY
        ,consumerName VARCHAR2 (100 CHAR) CONSTRAINT NN_consumerName NOT NULL
        , consumerSurname VARCHAR2 (100 CHAR) CONSTRAINT NN_consumerSurname NOT NULL
        , sex VARCHAR2 (1 CHAR) CONSTRAINT NN_sex NOT NULL
                CONSTRAINT CH_sex CHECK (sex in ('M','F'))
        , mobilePhone VARCHAR2 (9 CHAR)
        , addressCode INTEGER CONSTRAINT NN_addressCode NOT NULL
                CONSTRAINT FK_Address_Consumer REFERENCES Address (addressCode)
        , identityCode INTEGER CONSTRAINT NN_identityCode NOT NULL
                CONSTRAINT FK_IdentityType REFERENCES IdentityType (identityCode)
        , identificationNumber VARCHAR2 (50 CHAR)
                CONSTRAINT NN_ identificationNumber NOT NULL
        , bankCode VARCHAR2(11 CHAR) CONSTRAINT NN_bankCode NOT NULL
                CONSTRAINT FK_ Bank REFERENCES Bank (bankCode)
        , accountCode VARCHAR2 (100 CHAR) CONSTRAINT NN_accountCode NOT NULL
        , CONSTRAINT UN_Consumer UNIQUE (consumerName, consumerSurname, sex, IdentityCode,
identificationNumber)

);
```

Script para la creación de la tabla Company

La tabla COMPANY almacena los datos de las compañías suministradoras de la energía. La tabla está formada por un código que consiste en la identificación administrativa de la empresa, el nombre de la empresa, y un código de dirección que define la sede de la razón social de la empresa (dirección, ciudad, provincia y país).

```
CREATE TABLE Company (
```

```
        CompanyTaxCode VARCHAR2 (20 CHAR)
               CONSTRAINT PK_CompanyCode_Company PRIMARY KEY
        , companyName VARCHAR2 (200 CHAR) CONSTRAINT NN_companyName NOT NULL
        , addressCode INTEGER CONSTRAINT NN_addressCode_Company NOT NULL
         CONSTRAINT FK_Address_Company REFERENCES Address (addressCode)

);
```

Script para la creación de la tabla Meter

La tabla METER almacena los datos de los contadores. Los contadores se definen como: Un código de contador que coincide con el número de serie del mismo, el modelo del contador, el código de contrato con el que se asocia el contador al ser puesto en marcha, la potencia contratada, la fecha de la última inspección técnica, la fecha de instalación, la compañía a la que pertenece el contador, y un código de dirección que define la ubicación física del contador (dirección, ciudad, provincia y país).

```
CREATE TABLE Meter (
        serialNumber VARCHAR2 (20 CHAR) CONSTRAINT PK_serialNumber_Meter PRIMARY KEY
        , meterModel VARCHAR2 (100 CHAR) CONSTRAINT NN_model NOT NULL
        , contractCode VARCHAR2 (100 CHAR)
        , contractedPower NUMBER (5, 2) CONSTRAINT NN_contractedPower NOT NULL
        , lastTechnicalInspection DATE
        , installationDate DATE CONSTRAINT NN_installationDate NOT NULL
        , companyCode VARCHAR2 (15 CHAR)
               CONSTRAINT NN_MeterCompanyCode NOT NULL
               CONSTRAINT FK_Meter_Company REFERENCES Company (companyTaxCode)
        , consumerCode INTEGER CONSTRAINT FK_Consumer REFERENCES Consumer (consumerCode)
        , addressCode INTEGER
               CONSTRAINT NN_MeterAddressCode NOT NULL
               CONSTRAINT FK_Address_Meter REFERENCES Address (addressCode)
);
```

Script para la creación de la tabla Connection

La tabla CONNECTION es la más importante de todas. Define las lecturas de los contadores y su estructura almacena: la fecha y hora de la lectura del contador, el número de serie del contador, el consumo instantáneo, y si la lectura ha sido correcta o no. La información de esta tabla la produce el procedimiento almacenado correspondiente que controla la grabación o no de la lectura del contador en función de las restricciones previstas en el enunciado del problema.

```
CREATE TABLE Connection (
  readingDate TIMESTAMP WITH LOCAL TIME ZONE
       CONSTRAINT NN_ReadingDate NOT NULL
  , meterSerialNumber VARCHAR2 (20 CHAR)
       CONSTRAINT NN_meterSerialNumber NOconnT NULL
       CONSTRAINT FK_Connection_Meter REFERENCES Meter (serialNumber)
  , instantConsumption INTEGER
  , isSuccess VARCHAR2 (1 CHAR)
       CONSTRAINT NN_isSuccess NOT NULL
       CONSTRAINT CK_isSuccess CHECK (isSuccess in ('Y','N'))
  , CONSTRAINT PK_Connection PRIMARY KEY (readingDate, meterSerialNumber)
);
```

Script para la creación de la tabla Price

```
CREATE TABLE Price (
```

```
        changeData DATE CONSTRAINT NN_ Changedata NOT NULL
        , countryCode INTEGER
                CONSTRAINT NN_countryCode NOT NULL
                CONSTRAINT FK_Price_Country REFERENCES Country (CountryCode)
        , companyCode VARCHAR2 (15 CHAR)
                CONSTRAINT NN_ companyCode NOT NULL
                CONSTRAINT FK_ Company REFERENCES Company (CompanyTaxCode)
        , newPrice NUMBER (7, 6) CONSTRAINT NN_newPrice NOT NULL
        , CONSTRAINT PK_Price PRIMARY KEY (ChangeData, countryCode, companyCode)
);
```

Script para la creación de la tabla Clients

La tabla CLIENTS almacena el país al que pertenece un consumidor. Es una tabla auxiliar compuesta por: la fecha de alta del contrato del consumidor, la compañía suministradora con la que suscribe el contrato, y el código de consumidor.

```
CREATE TABLE Clients (
        hireDate DATE CONSTRAINT NN_hireDate NOT NULL
        , companyCode VARCHAR2 (15 CHAR)
                CONSTRAINT NN_ClientsCompanyCode NOT NULL
                CONSTRAINT FK_Clients_Company REFERENCES Company (CompanyTaxCode)
        , consumerCode INTEGER
                CONSTRAINT NN_ClientsConsumerCode NOT NULL
                CONSTRAINT FK_Clients_Consumer REFERENCES Address (addressCode)
        , CONSTRAINT PK_Clients PRIMARY KEY (companyCode, consumerCode)
);
```

Script para la creación de la tabla Operators

La tabla OPERATORS define los países en los que presta servicio una compañía suministradora. Se compone de un código de país, y un código de compañía suministradora.

```
CREATE TABLE Operators (
        countryCode INTEGER CONSTRAINT NN_OperatorsCountryCode NOT NULL
                CONSTRAINT FK_Operators_Country REFERENCES Country (CountryCode)
        , companyCode VARCHAR2 (15 CHAR)
                CONSTRAINT NN_OperatorsCompanyCode NOT NULL
                CONSTRAINT FK_Operators_Company REFERENCES Company (CompanyTaxCode)
        , CONSTRAINT PK_Operators PRIMARY KEY (countryCode, companyCode)
);
```

Script para la creación de la tabla Log (secuencia, trigger, y creación de la tabla)

La tabla LOG registra la actividad de los procedimientos ABM. La ejecución de cualquier procedimiento almacenado produce un registro en esta tabla compuesto por: La fecha y hora de ejecución del registro almacenado, el nombre del procedimiento almacenado ejecutado, los parámetros de entrada del procedimiento almacenado, los parámetros de salida y el usuario que ejecuta el proceso.

```
CREATE TABLE LuzLog (
  IdLog INTEGER CONSTRAINT Pk_LuzLog PRIMARY KEY
  , execDate TIMESTAMP
  , SPName VARCHAR(100) CONSTRAINT NN_execProc NOT NULL
```

```
  , inParams VARCHAR(2000)
  , outParams VARCHAR(200)
  , userExec VARCHAR2 (100) CONSTRAINT NN_userExec NOT NULL
);

CREATE SEQUENCE seq_LuzLog INCREMENT BY 1 START WITH 1;

CREATE OR REPLACE TRIGGER Insert_idLog_LuzLog
  BEFORE INSERT ON LuzLog
  FOR EACH ROW
BEGIN
  SELECT seq_LuzLog.NEXTVAL INTO :NEW.idLog FROM DUAL;
END Insert_idLog_LuzLog;
```