Anexo VII. Creación de las tablas del DW

Dimensión Address: W_ADDRESS_D

Creación de la tabla de dimensiones de Addres, sus secuencias y disparadores

```
/*****************************************************************************/
/* ADDRESS DIMENSION */
CREATE TABLE W_ADDRESS_D(
        address_WID INTEGER CONSTRAINT PK_W_Address_D PRIMARY KEY
  , addressCode INTEGER CONSTRAINT nn_addressCode NOT NULL
  , city_WID INTEGER CONSTRAINT nn_city_WID NOT NULL
  , cityCode INTEGER CONSTRAINT nn_cityCode NOT NULL
        , cityName VARCHAR2 (100) CONSTRAINT nn_cityMeterAddress NOT NULL
  , country_WID INTEGER CONSTRAINT nn_country_WID NOT NULL
  , countryCode INTEGER CONSTRAINT nn_countryCode NOT NULL
  , countryName VARCHAR2 (100 CHAR) CONSTRAINT NN_countryName NOT NULL) TABLESPACE
DIMENSIONS;

/* ADDRESS PK SEQUENCE*/
CREATE SEQUENCE seq_address_WID INCREMENT BY 1 START WITH 1;
/* CITY PK SEQUENCE*/
CREATE SEQUENCE seq_city_WID INCREMENT BY 1 START WITH 1;
/* COUNTRY PK SEQUENCE*/
CREATE SEQUENCE seq_country_WID INCREMENT BY 1 START WITH 1;

/*ADDRESS PK TRIGGER */
CREATE OR REPLACE TRIGGER Insert_address_WID
  BEFORE INSERT ON W_ADDRESS_D
  FOR EACH ROW
BEGIN
  SELECT seq_address_WID.NEXTVAL INTO :NEW.address_WID FROM DUAL;
  SELECT seq_city_WID.NEXTVAL INTO :NEW.city_WID FROM DUAL;
  SELECT seq_country_WID.NEXTVAL INTO :NEW.country_WID FROM DUAL;
END Insert_address_WID;
/*****************************************************************************
*/
```

Dimensión Company: W_COMPANY_D

Creación de la tabla de dimensiones de Company, sus secuencias y disparadores

```
/*****************************************************************************/
/* COMPANY DIMENSION */
CREATE TABLE W_COMPANY_D(
        company_WID INTEGER CONSTRAINT PK_W_COMPANY_D PRIMARY KEY
        , COMPANYCODE VARCHAR2(15 CHAR) CONSTRAINT NN_COMPANYCODE NOT NULL
, companyName VARCHAR2 (200 CHAR) CONSTRAINT NN_W_companyName NOT NULL
) TABLESPACE DIMENSIONS;

/* COMPANY PK SEQUENCE */
CREATE SEQUENCE seq_company_WID INCREMENT BY 1 START WITH 1;

/* COMPANY PK TRIGGER  */
CREATE OR REPLACE TRIGGER Insert_company_WID
  BEFORE INSERT ON W_COMPANY_D
  FOR EACH ROW
BEGIN
  SELECT seq_company_WID.NEXTVAL INTO :NEW.company_WID FROM DUAL;
END Insert_company_WID;
```

Dimensión Consumer: W_CONSUMER_D

Creación de la tabla de dimensiones de Consumer, sus secuencias y disparadores

```
/*****************************************************************************/
/* CONSUMER DIMENSION */
```

```
CREATE TABLE W_CONSUMER_D(
        consumer_WID INTEGER CONSTRAINT PK_W_CONSUMER_D PRIMARY KEY
, CONSUMERCODE INTEGER CONSTRAINT NN_CONSUMERCODE NOT NULL
        , consumerName VARCHAR2 (100 CHAR) CONSTRAINT NN_consumer_d_Name NOT NULL
) TABLESPACE DIMENSIONS;

/* CONSUMER PK SEQUENCE */
CREATE SEQUENCE seq_consumer_WID INCREMENT BY 1 START WITH 1;

/* CONSUMER PK TRIGGER  */
CREATE OR REPLACE TRIGGER Insert_consumer_WID
  BEFORE INSERT ON W_CONSUMER_D
  FOR EACH ROW
BEGIN
  SELECT seq_consumer_WID.NEXTVAL INTO :NEW.consumer_WID FROM DUAL;
END Insert_consumer_WID;
```

Dimensión Date: W_DATE_D

Creación de la tabla de dimensiones de fecha (date), sus secuencias y disparadores. La tabla date se encuentra particionada en el *tablespace* de *facts* (hechos). Esto es así porque la partición de la tabla de hechos se realiza por referencia a la clave foránea de esta tabla.

```
/* DATE DIMENSION */
CREATE TABLE W_DATE_D(
        date_WID DATE CONSTRAINT PK_W_Date_D PRIMARY KEY
        , dayOfWeek INTEGER CONSTRAINT NN_DayOfWeek NOT NULL
        , dayNumberMonth INTEGER CONSTRAINT NN_dayNumberMonth NOT NULL
        , monthNumber INTEGER CONSTRAINT NN_monthNumber NOT NULL
        , monthName VARCHAR2 (3 CHAR) CONSTRAINT NN_monthName NOT NULL
        , yearNumber INTEGER CONSTRAINT NN_yearNumber NOT NULL
)
PARTITION BY LIST(monthNumber) (
  PARTITION PART1 VALUES (1) TABLESPACE FACTS_01
  , PARTITION PART2 VALUES (2) TABLESPACE FACTS_02
  , PARTITION PART3 VALUES (3) TABLESPACE FACTS_03
  , PARTITION PART4 VALUES (4) TABLESPACE FACTS_04
  , PARTITION PART5 VALUES (5) TABLESPACE FACTS_05
  , PARTITION PART6 VALUES (6) TABLESPACE FACTS_06
  , PARTITION PART7 VALUES (7) TABLESPACE FACTS_07
  , PARTITION PART8 VALUES (8) TABLESPACE FACTS_08
  , PARTITION PART9 VALUES (9) TABLESPACE FACTS_09
  , PARTITION PART10 VALUES (10) TABLESPACE FACTS_10
  , PARTITION PART11 VALUES (11) TABLESPACE FACTS_11
  , PARTITION PART12 VALUES (12) TABLESPACE FACTS_12
);
```

Dimensión Meter: W_METER_D

Creación de la tabla de dimensiones de contadores (Meter), sus secuencias y disparadores.

```
/*****************************************************************************/
/* METER DIMENSION */
CREATE TABLE W_METER_D(
        meter_WID INTEGER CONSTRAINT PK_W_Meter_D PRIMARY KEY
        , METERCODE VARCHAR2(20 CHAR) CONSTRAINT NN_METERCODE NOT NULL
        , meterInstallationDate DATE CONSTRAINT NN_meterInstallationDate NOT NULL
        , modelName VARCHAR2 (100 CHAR) CONSTRAINT NN_modelName NOT NULL
) TABLESPACE DIMENSIONS;

/* METER SEQUENCE */
CREATE SEQUENCE seq_meter_WID INCREMENT BY 1 START WITH 1;

/* METER PK TRIGGER */
CREATE OR REPLACE TRIGGER Insert_meter_WID
  BEFORE INSERT ON W_METER_D
  FOR EACH ROW
BEGIN
  SELECT seq_meter_WID.NEXTVAL INTO :NEW.meter_WID FROM DUAL;
END Insert_meter_WID;
```

Dimensión Time: W_TIME_D

Creación de la tabla de dimensiones de tiempo (Time), sus secuencias y disparadores.

```
/*******************************************************************************/
/* TIME DIMENSION */
CREATE TABLE W_TIME_D(
       time_WID INTEGER CONSTRAINT PK_W_Time_D PRIMARY KEY
       , timeHourNumber VARCHAR2(5)
) TABLESPACE DIMENSIONS;
```

Tabla de Hechos: W_CONSUMPTION_F

Creación de la tabla de hechos (facts) con las lecturas de los contadores, sus secuencias y disparadores.

```
/*******************************************************************************/
/* CONSUMPTION FACT TABLE */
CREATE TABLE W_CONSUMPTION_F(
       consumption_Wid INTEGER CONSTRAINT PK_W_CONSUMPTION_F PRIMARY KEY
       , date_WID DATE
      CONSTRAINT NN_date_WID NOT NULL
      CONSTRAINT FK_w_date_D REFERENCES W_DATE_D (date_WID)
       , time_WID INTEGER
      CONSTRAINT NN_time_WID NOT NULL
      CONSTRAINT FK_W_TIME_D REFERENCES W_TIME_D (time_WID)
       , meter_WID INTEGER
      CONSTRAINT NN_meter_WID NOT NULL
      CONSTRAINT FK_W_METER_D REFERENCES W_METER_D (meter_WID)
, address_WID INTEGER
      CONSTRAINT NN_address_WID NOT NULL
      CONSTRAINT FK_W_address_D REFERENCES W_address_D (address_WID)
, company_WID INTEGER
      CONSTRAINT NN_company_WID NOT NULL
      CONSTRAINT FK_W_COMPANY_D REFERENCES W_COMPANY_D (company_WID)
       , consumer_WID INTEGER
      CONSTRAINT NN_consumer_WID NOT NULL
      CONSTRAINT FK_W_CONSUMER_D REFERENCES W_CONSUMER_D (consumer_WID )
, previousReading INTEGER
       , actualReading INTEGER
       , consumption INTEGER CONSTRAINT NN_consumption NOT NULL
       , price NUMBER (7, 6) CONSTRAINT NN_price NOT NULL
       , energyCost NUMBER(10,2) CONSTRAINT NN_cost NOT NULL
)
PARTITION BY REFERENCE (FK_w_date_D);

/* consumption PK SEQUENCE*/
CREATE SEQUENCE seq_consumption_Wid INCREMENT BY 1 START WITH 1;

/* CONSUMPTION PK TRIGGER */
CREATE OR REPLACE TRIGGER Insert_consumption_Wid
  BEFORE INSERT ON W_CONSUMPTION_F
  FOR EACH ROW
BEGIN
  SELECT seq_consumption_Wid.NEXTVAL INTO :NEW.consumption_Wid FROM DUAL;
END Insert_consumption_Wid;
```

Trigger para los cálculos de W_CONSUMPTION_F y llenado de las tablas resumen

El trigger cálcula el consumo como la diferencia entre la lectura anterior y la actual siempre que ninguna de las dos tenga un valor nulo. También calcula el coste de la energía mediante el valor del precio proporcionado.

Por otra parte ejecuta procesos MERGE en las tablas resumen acumulando los valores de consumo energético y coste de la energía.

```
create or replace TRIGGER consumption_Calcs
  BEFORE INSERT OR UPDATE ON W_CONSUMPTION_F
  FOR EACH ROW
DECLARE
  CONSU  INTEGER;
  ECOST  NUMBER(12,2);
BEGIN
  BEGIN
      /*CALCULATE CONSUMPTION */
      IF NVL(:NEW.previousReading,0) >0 AND NVL(:NEW.actualReading,0)>0 THEN
          SELECT (:NEW.actualReading-:NEW.previousReading) INTO :NEW.consumption FROM
DUAL;
          SELECT (:NEW.actualReading-:NEW.previousReading) INTO CONSU FROM DUAL;
          /* CALCULATE ENERGY COST */
          IF NVL(:NEW.PRICE,0)>0 THEN
            SELECT (:NEW.actualReading-:NEW.previousReading)*:NEW.Price INTO
:NEW.energyCost FROM DUAL;
            SELECT (:NEW.actualReading-:NEW.previousReading)*:NEW.Price INTO ECOST FROM
DUAL;
          ELSE
            SELECT 0 INTO :NEW.energyCost FROM DUAL;
            SELECT 0 INTO ECOST FROM DUAL;
          END IF;
      ELSE
        SELECT 0 INTO :NEW.cONSUMPTION FROM DUAL;
        SELECT 0 INTO :NEW.energyCost FROM DUAL;
        SELECT 0 INTO CONSU FROM DUAL;
        SELECT 0 INTO ECOST FROM DUAL;
      END IF;
  EXCEPTION
    WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('ERROR in CALCULATE CONSUMPTION:' || SQLERRM);
  END;


  BEGIN
     /************************ SUMMARY TABLES: CITY_METER
MONTH*************************/
     MERGE INTO W_CITY_METER_SUMMARY_M S
     USING (SELECT W_ADDRESS_D.CITY_WID, :NEW.METER_WID AS MET_WID,
TO_NUMBER(TO_CHAR(:NEW.DATE_WID, 'MM')) AS MO, TO_NUMBER(TO_CHAR(:NEW.DATE_WID, 'YYYY'))
AS YE
            ,(SELECT CONSU FROM DUAL) AS CONSUM, (SELECT ECOST FROM DUAL) AS COSTE
            FROM DUAL INNER JOIN W_ADDRESS_D ON
:NEW.ADDRESS_WID=W_ADDRESS_D.ADDRESS_WID) D
     ON (S.CITY_WID=D.CITY_WID AND S.METER_WID=D.MET_WID AND S.MONTHNUMBER=D.MO AND
S.YEARNUMBER=D.YE)
    WHEN MATCHED THEN
      UPDATE SET
        S.CONSUMPTION_M=S.CONSUMPTION_M+D.CONSUM
        ,S.ENERGYCOST=S.ENERGYCOST+D.COSTE
    WHEN NOT MATCHED THEN
      INSERT(S.CITY_WID, S.METER_WID, S.MONTHNUMBER, S.YEARNUMBER, S.CONSUMPTION_M,
S.ENERGYCOST)
      VALUES(D.CITY_WID, D.MET_WID, D.MO, D.YE, D.CONSUM, D.COSTE);
  EXCEPTION
    WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('ERROR in  CITY_METER MONTH:' || SQLERRM);
  END;


  BEGIN
    /************************ SUMMARY TABLES: CITY_CONSUMER
MONTH*************************/
     MERGE INTO W_CITY_CONS_SUMMARY_M S
     USING (SELECT W_ADDRESS_D.CITY_WID, :NEW.CONSUMER_WID AS CONS_WID,
TO_NUMBER(TO_CHAR(:NEW.DATE_WID, 'MM')) AS MO, TO_NUMBER(TO_CHAR(:NEW.DATE_WID, 'YYYY'))
AS YE
            ,(SELECT CONSU FROM DUAL) AS CONSUM, (SELECT ECOST FROM DUAL) AS COSTE
            FROM DUAL INNER JOIN W_ADDRESS_D ON
:NEW.ADDRESS_WID=W_ADDRESS_D.ADDRESS_WID) D
     ON (S.CITY_WID=D.CITY_WID AND S.CONSUMER_WID=D.CONS_WID AND S.MONTHNUMBER=D.MO AND
S.YEARNUMBER=D.YE)
    WHEN MATCHED THEN
      UPDATE SET
        S.CONSUMPTION_M=S.CONSUMPTION_M+D.CONSUM
        ,S.ENERGYCOST=S.ENERGYCOST+D.COSTE
```

```
    WHEN NOT MATCHED THEN
      INSERT(S.CITY_WID, S.CONSUMER_WID, S.MONTHNUMBER, S.YEARNUMBER, S.CONSUMPTION_M,
S.ENERGYCOST)
      VALUES(D.CITY_WID, D.CONS_WID, D.MO, D.YE, D.CONSUM, D.COSTE);
  EXCEPTION
    WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('ERROR in CITY_CONSUMER MONTH:' || SQLERRM);
  END;


  BEGIN
      /************************ SUMMARY TABLES: COUNTRY_CITY
MONTH*************************/
       MERGE INTO W_COUNTRY_CITY_SUMMARY_M S
       USING (
          SELECT
                W_ADDRESS_D.COUNTRY_WID, W_ADDRESS_D.CITY_WID
              , TO_NUMBER(TO_CHAR(:NEW.DATE_WID, 'MM')) AS MO,
TO_NUMBER(TO_CHAR(:NEW.DATE_WID, 'YYYY')) AS YE
              , (SELECT CONSU FROM DUAL) AS CONSUM, (SELECT ECOST FROM DUAL) AS COSTE
              , (SELECT COUNT(METER_WID) FROM W_CITY_METER_SUMMARY_M WHERE
CITY_WID=W_ADDRESS_D.CITY_WID AND MONTHNUMBER=TO_NUMBER(TO_CHAR(:NEW.DATE_WID, 'MM'))
AND YEARNUMBER=TO_NUMBER(TO_CHAR(:NEW.DATE_WID, 'YYYY'))) AS METERNUMBER
              , (SELECT COUNT(CONSUMER_WID) FROM W_CITY_CONS_SUMMARY_M WHERE
CITY_WID=W_ADDRESS_D.CITY_WID AND MONTHNUMBER=TO_NUMBER(TO_CHAR(:NEW.DATE_WID, 'MM'))
AND YEARNUMBER=TO_NUMBER(TO_CHAR(:NEW.DATE_WID, 'YYYY'))) AS CONSUMERNUMBER
          FROM DUAL INNER JOIN W_ADDRESS_D ON :NEW.ADDRESS_WID=W_ADDRESS_D.ADDRESS_WID)
D
          ON (S.COUNTRY_WID=D.COUNTRY_WID AND S.CITY_WID=D.CITY_WID AND
S.MONTHNUMBER=D.MO AND S.YEARNUMBER=D.YE)
      WHEN MATCHED THEN
        UPDATE SET
           S.CONSUMPTION_M=S.CONSUMPTION_M+D.CONSUM
         , S.ENERGYCOST=S.ENERGYCOST+D.COSTE
         , S.METERNUMBER=D.METERNUMBER
         , S.CONSUMERNUMBER=D.CONSUMERNUMBER
      WHEN NOT MATCHED THEN
        INSERT(S.COUNTRY_WID,S.CITY_WID, S.MONTHNUMBER, S.YEARNUMBER, S.CONSUMPTION_M,
S.ENERGYCOST, S.METERNUMBER, S.CONSUMERNUMBER)
        VALUES(D.COUNTRY_WID,D.CITY_WID, D.MO, D.YE, D.CONSUM, D.COSTE, D.METERNUMBER,
D.CONSUMERNUMBER);
  EXCEPTION
    WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('ERROR in COUNTRY_CITY MONTH:' || SQLERRM);
  END;


  BEGIN
    /************************ SUMMARY TABLES: CITY MONTH*************************/
     MERGE INTO W_CITY_SUMMARY_M S
     USING (
        SELECT W_ADDRESS_D.CITY_WID
            , TO_NUMBER(TO_CHAR(:NEW.DATE_WID, 'MM')) AS MO,
TO_NUMBER(TO_CHAR(:NEW.DATE_WID, 'YYYY')) AS YE
            , (SELECT CONSU FROM DUAL) AS CONSUM, (SELECT ECOST FROM DUAL) AS COSTE
            , (SELECT COUNT(METER_WID) FROM W_CITY_METER_SUMMARY_M WHERE
CITY_WID=W_ADDRESS_D.CITY_WID AND MONTHNUMBER=TO_NUMBER(TO_CHAR(:NEW.DATE_WID, 'MM'))
AND YEARNUMBER=TO_NUMBER(TO_CHAR(:NEW.DATE_WID, 'YYYY'))) AS METERNUMBER
            , (SELECT COUNT(CONSUMER_WID) FROM W_CITY_CONS_SUMMARY_M WHERE
CITY_WID=W_ADDRESS_D.CITY_WID AND MONTHNUMBER=TO_NUMBER(TO_CHAR(:NEW.DATE_WID, 'MM'))
AND YEARNUMBER=TO_NUMBER(TO_CHAR(:NEW.DATE_WID, 'YYYY'))) AS CONSUMERNUMBER
        FROM DUAL INNER JOIN W_ADDRESS_D ON :NEW.ADDRESS_WID=W_ADDRESS_D.ADDRESS_WID) D
     ON (S.CITY_WID=D.CITY_WID AND S.MONTHNUMBER=D.MO AND S.YEARNUMBER=D.YE)
    WHEN MATCHED THEN
      UPDATE SET
         S.CONSUMPTION_M=S.CONSUMPTION_M+D.CONSUM
       , S.ENERGYCOST=S.ENERGYCOST+D.COSTE
       , S.METERNUMBER=D.METERNUMBER
       , S.CONSUMERNUMBER=D.CONSUMERNUMBER
    WHEN NOT MATCHED THEN
      INSERT(S.CITY_WID, S.MONTHNUMBER, S.YEARNUMBER, S.CONSUMPTION_M, S.ENERGYCOST,
S.METERNUMBER, S.CONSUMERNUMBER)
      VALUES(D.CITY_WID, D.MO, D.YE, D.CONSUM, D.COSTE, D.METERNUMBER,
D.CONSUMERNUMBER);
  EXCEPTION
    WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('ERROR in CITY MONTH:' || SQLERRM);
```

```
  END;

  BEGIN
    /*********************** SUMMARY TABLES: CITY_METER*************************/
     MERGE INTO W_CITY_METER_SUM_T S
     USING (
        SELECT W_ADDRESS_D.CITY_WID, :NEW.METER_WID AS MET_WID
            , (SELECT CONSU FROM DUAL) AS CONSUM, (SELECT ECOST FROM DUAL) AS COSTE
        FROM DUAL INNER JOIN W_ADDRESS_D ON :NEW.ADDRESS_WID=W_ADDRESS_D.ADDRESS_WID) D
      ON (S.CITY_WID=D.CITY_WID AND S.METER_WID=D.MET_WID)
    WHEN MATCHED THEN
      UPDATE SET
          S.CONSUMPTION_T=S.CONSUMPTION_T+D.CONSUM
        , S.ENERGYCOST=S.ENERGYCOST+D.COSTE
    WHEN NOT MATCHED THEN
      INSERT(S.CITY_WID, S.METER_WID, S.CONSUMPTION_T, S.ENERGYCOST)
      VALUES(D.CITY_WID, D.MET_WID, D.CONSUM, D.COSTE);
  EXCEPTION
    WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('ERROR in CITY METER:' || SQLERRM);
  END;


  BEGIN
    /*********************** SUMMARY TABLES: COMPANY MONTH*************************/
     MERGE INTO W_COMPANY_SUMMARY_M S
     USING (SELECT :NEW.COMPANY_WID AS COM_WID, TO_NUMBER(TO_CHAR(:NEW.DATE_WID, 'MM'))
AS MO, TO_NUMBER(TO_CHAR(:NEW.DATE_WID, 'YYYY')) AS YE
              ,(SELECT CONSU FROM DUAL) AS CONSUM, (SELECT ECOST FROM DUAL) AS COSTE
            , (CASE WHEN CONSU=0 THEN 0 ELSE 1 END) AS CORRECT, 1 AS TOTALR FROM DUAL) D
      ON (S.COMPANY_WID=D.COM_WID AND S.MONTHNUMBER=D.MO AND S.YEARNUMBER=D.YE)
    WHEN MATCHED THEN
      UPDATE SET
        S.CONSUMPTION_M=S.CONSUMPTION_M+D.CONSUM
        , S.ENERGYCOST=S.ENERGYCOST+D.COSTE
        , S.CORRECTREADINGS=S.CORRECTREADINGS+D.CORRECT
        , S.TOTALREADINGS=S.TOTALREADINGS+D.TOTALR
    WHEN NOT MATCHED THEN
      INSERT(S.COMPANY_WID, S.MONTHNUMBER, S.YEARNUMBER, S.CONSUMPTION_M, S.ENERGYCOST,
S.CORRECTREADINGS, S.TOTALREADINGS)
      VALUES(D.COM_WID, D.MO, D.YE, D.CONSUM, D.COSTE, D.CORRECT, D.TOTALR);
  EXCEPTION
    WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('ERROR in COMPANY MONTH:' || SQLERRM);
  END;


  BEGIN
     /*********************** SUMMARY TABLES: CONSUMER MONTH*************************/
     MERGE INTO W_CONSUMER_SUMMARY_M S
     USING (SELECT :NEW.CONSUMER_WID AS CON_WID, TO_NUMBER(TO_CHAR(:NEW.DATE_WID, 'MM'))
AS MO, TO_NUMBER(TO_CHAR(:NEW.DATE_WID, 'YYYY')) AS YE
            ,(SELECT CONSU FROM DUAL) AS CONSUM, (SELECT ECOST FROM DUAL) AS COSTE
             FROM DUAL) D
     ON (S.CONSUMER_WID=D.CON_WID AND S.MONTHNUMBER=D.MO AND S.YEARNUMBER=D.YE)
    WHEN MATCHED THEN
      UPDATE SET
        S.CONSUMPTION_M=S.CONSUMPTION_M+D.CONSUM
        ,S.ENERGYCOST=S.ENERGYCOST+D.COSTE
    WHEN NOT MATCHED THEN
      INSERT(S.CONSUMER_WID, S.MONTHNUMBER, S.YEARNUMBER, S.CONSUMPTION_M, S.ENERGYCOST)
      VALUES(D.CON_WID, D.MO, D.YE, D.CONSUM, D.COSTE);
  EXCEPTION
    WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('ERROR in CONSUMER MONTH:' || SQLERRM);
  END;


  BEGIN
    /*********************** SUMMARY TABLES: COUNTRY MONTH*************************/
     MERGE INTO W_COUNTRY_SUMMARY_M S
     USING (SELECT W_ADDRESS_D.COUNTRY_WID, TO_NUMBER(TO_CHAR(:NEW.DATE_WID, 'MM')) AS
MO, TO_NUMBER(TO_CHAR(:NEW.DATE_WID, 'YYYY')) AS YE
            ,(SELECT CONSU FROM DUAL) AS CONSUM, (SELECT ECOST FROM DUAL) AS COSTE
            FROM DUAL INNER JOIN W_ADDRESS_D ON
:NEW.ADDRESS_WID=W_ADDRESS_D.ADDRESS_WID) D
     ON (S.COUNTRY_WID=D.COUNTRY_WID AND S.MONTHNUMBER=D.MO AND S.YEARNUMBER=D.YE)
```

```
    WHEN MATCHED THEN
      UPDATE SET
        S.CONSUMPTION_M=S.CONSUMPTION_M+D.CONSUM
        ,S.ENERGYCOST=S.ENERGYCOST+D.COSTE
    WHEN NOT MATCHED THEN
      INSERT(S.COUNTRY_WID, S.MONTHNUMBER, S.YEARNUMBER, S.CONSUMPTION_M, S.ENERGYCOST)
      VALUES(D.COUNTRY_WID, D.MO, D.YE, D.CONSUM, D.COSTE);
  EXCEPTION
    WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('ERROR in COUNTRY MONTH:' || SQLERRM);
  END;


  BEGIN
    /********************** SUMMARY TABLES: METER MONTH*************************/
    MERGE INTO W_METER_SUMMARY_M S
      USING (SELECT :NEW.METER_WID AS MET_WID, TO_NUMBER(TO_CHAR(:NEW.DATE_WID, 'MM'))
AS MO, TO_NUMBER(TO_CHAR(:NEW.DATE_WID, 'YYYY')) AS YE
             ,(SELECT CONSU FROM DUAL) AS CONSUM, (SELECT ECOST FROM DUAL) AS COSTE
             FROM DUAL) D
      ON (S.METER_WID=D.MET_WID AND S.MONTHNUMBER=D.MO AND S.YEARNUMBER=D.YE)
    WHEN MATCHED THEN
      UPDATE SET
        S.CONSUMPTION_M=S.CONSUMPTION_M+D.CONSUM
        ,S.ENERGYCOST=S.ENERGYCOST+D.COSTE
    WHEN NOT MATCHED THEN
      INSERT(S.METER_WID, S.MONTHNUMBER, S.YEARNUMBER, S.CONSUMPTION_M, S.ENERGYCOST)
      VALUES(D.MET_WID, D.MO, D.YE, D.CONSUM, D.COSTE);
  EXCEPTION
    WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('ERROR in METER MONTH:' || SQLERRM);
  END;
END consumption_Calcs;
```

Tabla Resumen de Consumidores por ciudad: W_CITY_CONS_SUMMARY_M

Es una tabla resumen que indica el consumo mensual y el coste de cada consumidor por mes y a que ciudad pertenece.

```
/*****************************************************************************/
/* CITY CONSUMER MONTH SUMMARY
*/
/*****************************************************************************/
CREATE TABLE W_CITY_CONS_SUMMARY_M(
  wid INTEGER CONSTRAINT PK_W_CITY_CONS_SUMMARY_M PRIMARY KEY
  , city_Wid INTEGER
  , consumer_Wid INTEGER
  , monthNumber INTEGER
  , yearNumber INTEGER
  , consumption_M INTEGER
  , energyCost NUMBER(12,2)
);

/* W_CITY_CONS_SUMMARY_M PK SEQUENCE*/
CREATE SEQUENCE seq_W_CITY_CONS_SUM_M INCREMENT BY 1 START WITH 1;

/* W_CITY_CONS_SUMMARY_M PK TRIGGER */
CREATE OR REPLACE TRIGGER Insert_W_CITY_CONS_SUM_M
  BEFORE INSERT ON W_CITY_CONS_SUMMARY_M
  FOR EACH ROW
BEGIN
  SELECT seq_W_CITY_CONS_SUM_M.NEXTVAL INTO :NEW.wid FROM DUAL;
END Insert_W_CITY_CONS_SUM_M;
```

Tabla Resumen de Contadores por ciudad: W_CITY_METER_SUMMARY_M

Es una tabla resumen que indica el consumo mensual y el coste de cada contador por mes y a que ciudad pertenece.

```
/*****************************************************************************/
```

```
/* CITY METER MONTH SUMMARY                                                    */
/*****************************************************************************/
CREATE TABLE W_CITY_METER_SUMMARY_M(
  wid INTEGER CONSTRAINT PK_W_CITY_METER_SUMMARY_M PRIMARY KEY
  , city_Wid INTEGER
  , meter_Wid INTEGER
  , monthNumber INTEGER
  , yearNumber INTEGER
  , consumption_M INTEGER
  , energyCost NUMBER(12,2)
);

/* W_CITY_METER_SUMMARY_M PK SEQUENCE*/
CREATE SEQUENCE seq_W_CITY_METER_SUM_M INCREMENT BY 1 START WITH 1;

/* W_CITY_METER_SUMMARY_M PK TRIGGER */
CREATE OR REPLACE TRIGGER Insert_W_CITY_METER_SUM_M
  BEFORE INSERT ON W_CITY_METER_SUMMARY_M
  FOR EACH ROW
BEGIN
  SELECT seq_W_CITY_METER_SUM_M.NEXTVAL INTO :NEW.wid FROM DUAL;
END Insert_W_CITY_METER_SUM_M;
```

Tabla Resumen de Ciudades: W_CITY_SUMMARY_M

Es una tabla resumen que indica el consumo mensual de cada ciudad y su coste.

```
/*****************************************************************************/
/* CITY MONTH SUMMARY                                                        */
/*****************************************************************************/
CREATE TABLE W_CITY_SUMMARY_M(
  wid INTEGER CONSTRAINT PK_W_CITY_SUMMARY_M PRIMARY KEY
  , city_Wid INTEGER
  , monthNumber INTEGER
  , yearNumber INTEGER
  , consumption_M INTEGER
  , energyCost NUMBER(12,2)
  , meterNumber INTEGER
  , consumerNumber INTEGER
);

--ALTER TABLE  W_CITY_SUMMARY_M ADD (meterNumber INTEGER);
--ALTER TABLE  W_CITY_SUMMARY_M ADD (consumerNumber INTEGER);

/* W_CITY_SUMMARY_M PK SEQUENCE*/
CREATE SEQUENCE seq_W_CITY_SUM_M INCREMENT BY 1 START WITH 1;

/* W_CITY_SUMMARY_M PK TRIGGER */
CREATE OR REPLACE TRIGGER Insert_W_CITY_SUM_M
  BEFORE INSERT ON W_CITY_SUMMARY_M
  FOR EACH ROW
BEGIN
  SELECT seq_W_CITY_SUM_M.NEXTVAL INTO :NEW.wid FROM DUAL;
END Insert_W_CITY_SUM_M;
```

Tabla Resumen de Compañías: W_COMPANY_SUMMARY_M

Creación de la tabla de resumen (summary table) de la jerarquía país por mes y año.

```
*****************************************************************************/
/* COMPANY MONTH SUMMARY                                                     */
/*****************************************************************************/
CREATE TABLE W_COMPANY_SUMMARY_M(
  wid INTEGER CONSTRAINT PK_W_COMPANY_SUMMARY_M PRIMARY KEY
  , company_Wid INTEGER
  , monthNumber INTEGER
  , yearNumber INTEGER
  , consumption_M INTEGER
  , energyCost NUMBER(12,2)
  , correctReadings INTEGER
  , totalReadings INTEGER
);
```

```
/* W_COMPANY_SUMMARY_M PK SEQUENCE*/
CREATE SEQUENCE seq_W_COMPANY_SUM_M INCREMENT BY 1 START WITH 1;

/* W_COMPANY_SUMMARY_M PK TRIGGER */
CREATE OR REPLACE TRIGGER Insert_W_COMPANY_SUM_M
  BEFORE INSERT ON W_COMPANY_SUMMARY_M
  FOR EACH ROW
BEGIN
  SELECT seq_W_COMPANY_SUM_M.NEXTVAL INTO :NEW.wid FROM DUAL;
END Insert_W_COMPANY_SUM_M;
```

Tabla Resumen de Consumidores: W_CONSUMER_SUMMARY_M

Creación de la tabla de resumen (summary table) de la jerarquía consumidores país por mes y año.

```
/*****************************************************************************/
/* CITY CONSUMER MONTH SUMMARY
*/
/*****************************************************************************/
CREATE TABLE W_CITY_CONS_SUMMARY_M(
  wid INTEGER CONSTRAINT PK_W_CITY_CONS_SUMMARY_M PRIMARY KEY
  , city_Wid INTEGER
  , consumer_Wid INTEGER
  , monthNumber INTEGER
  , yearNumber INTEGER
  , consumption_M INTEGER
  , energyCost NUMBER(12,2)
);

/* W_CITY_CONS_SUMMARY_M PK SEQUENCE*/
CREATE SEQUENCE seq_W_CITY_CONS_SUM_M INCREMENT BY 1 START WITH 1;

/* W_CITY_CONS_SUMMARY_M PK TRIGGER */
CREATE OR REPLACE TRIGGER Insert_W_CITY_CONS_SUM_M
  BEFORE INSERT ON W_CITY_CONS_SUMMARY_M
  FOR EACH ROW
BEGIN
  SELECT seq_W_CITY_CONS_SUM_M.NEXTVAL INTO :NEW.wid FROM DUAL;
END Insert_W_CITY_CONS_SUM_M;
```

Tabla Resumen de ciudades por país: W_COUNTRY_CITY_SUMMARY_M

Es el consumo y coste de cada ciudad por mes y por país. Adicionalmente contiene también el número de contadores por ciudad y el número de consumidores por ciudad para poder obtener medias de consumo.

```
/*****************************************************************************/
/* COUNTRY CITY MONTH SUMMARY
*/
/*****************************************************************************/
CREATE TABLE W_COUNTRY_CITY_SUMMARY_M(
  wid INTEGER CONSTRAINT PK_W_COUNTRY_CITY_SUMMARY_M PRIMARY KEY
  , country_wid INTEGER
  , city_Wid INTEGER
  , monthNumber INTEGER
  , yearNumber INTEGER
  , consumption_M INTEGER
  , energyCost NUMBER(12,2)
  , MeterNumber INTEGER
  , ConsumerNumber INTEGER
);

/* W_COUNTRY_CITY_SUMMARY_M PK SEQUENCE*/
CREATE SEQUENCE seq_W_COUNTRY_CITY_SUM_M INCREMENT BY 1 START WITH 1;

/* W_COUNTRY_CITY_SUMMARY_M PK TRIGGER */
CREATE OR REPLACE TRIGGER Insert_W_COUNTRY_CITY_SUM_M
  BEFORE INSERT ON  W_COUNTRY_CITY_SUMMARY_M
  FOR EACH ROW
```

```
BEGIN
  SELECT seq_W_COUNTRY_CITY_SUM_M.NEXTVAL INTO :NEW.wid FROM DUAL;
END Insert_W_COUNTRY_CITY_SUM_M;
```

Tabla Resumen de Países: W_COUNTRY_SUMMARY_M

Creación de la tabla de resumen de la jerarquía país por mes y año. El trigger de la tabla puebla, a su vez la tabla de resumen de consumo por año (W_COUNTRY_SUMMARY_Y).

```
/*****************************************************************************/
/* COUNTRY MONTH SUMMARY                                                     */
/*****************************************************************************/
CREATE TABLE W_COUNTRY_SUMMARY_M(
  wid INTEGER CONSTRAINT PK_W_COUNTRY_SUMMARY_M PRIMARY KEY
  , country_Wid INTEGER
  , monthNumber INTEGER
  , yearNumber INTEGER
  , consumption_M INTEGER
  , energyCost NUMBER(12,2)
);

/* W_COUNTRY_SUMMARY_M PK SEQUENCE*/
CREATE SEQUENCE seq_W_COUNTRY_SUM_M INCREMENT BY 1 START WITH 1;

/* W_COUNTRY_SUMMARY_M PK TRIGGER */
create or replace TRIGGER Insert_W_COUNTRY_SUM_M
   BEFORE INSERT OR UPDATE ON W_COUNTRY_SUMMARY_M
  FOR EACH ROW
BEGIN
  /***************** PRIMARY KEY *******************************************/
  SELECT seq_W_COUNTRY_SUM_M.NEXTVAL INTO :NEW.wid FROM DUAL;

   /*********************** SUMMARY TABLES: COUNTRY YEAR***********************/
   MERGE INTO W_COUNTRY_SUMMARY_Y S
   USING (SELECT :NEW.COUNTRY_WID AS CO, :NEW.YEARNUMBER AS YE, :NEW.CONSUMPTION_M AS
CONS, :NEW.ENERGYCOST AS COSTE FROM DUAL) D
   ON (S.COUNTRY_WID=D.CO AND S.YEARNUMBER=D.YE)
  WHEN MATCHED THEN
    UPDATE SET
      S.CONSUMPTION=S.CONSUMPTION+D.CONS
      ,S.ENERGYCOST=S.ENERGYCOST+D.COSTE
  WHEN NOT MATCHED THEN
    INSERT(S.COUNTRY_WID, S.YEARNUMBER, S.CONSUMPTION, S.ENERGYCOST)
    VALUES(D.CO, D.YE, D.CONS, D.COSTE);

END Insert_W_COUNTRY_SUM_M;
```

Tabla Resumen de Países: W_COUNTRY_SUMMARY_Y

Creación de la tabla de resumen de la jerarquía país por año. El trigger de la tabla W_COUNTRY_SUMMARY_M puebla esta tabla de resumen.

```
/*****************************************************************************/
/* COUNTRY YEAR SUMMARY                                                      */
/*****************************************************************************/
CREATE TABLE W_COUNTRY_SUMMARY_Y(
  wid INTEGER CONSTRAINT PK_W_COUNTRY_SUMMARY_Y PRIMARY KEY
  , country_Wid INTEGER
  , yearNumber INTEGER
  , consumption INTEGER
  , energyCost NUMBER(12,2)
);

/* W_COUNTRY_SUMMARY_Y PK SEQUENCE*/
```

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

/* W_COUNTRY_SUMMARY_M PK TRIGGER */
CREATE OR REPLACE TRIGGER Insert_W_COUNTRY_SUM_Y
  BEFORE INSERT ON W_COUNTRY_SUMMARY_Y
  FOR EACH ROW
BEGIN
  SELECT seq_W_COUNTRY_SUM_Y.NEXTVAL INTO :NEW.wid FROM DUAL;
END Insert_W_COUNTRY_SUM_Y;
```

Tabla Resumen de Contadores: W_METER_SUMMARY_M

Creación de la tabla de resumen (summary table) de la jerarquía Contadores país por mes y año.

```
/*****************************************************************************/
/* METER MONTH SUMMARY                                                       */
/*****************************************************************************/
CREATE TABLE W_METER_SUMMARY_M(
  wid INTEGER CONSTRAINT PK_W_METER_SUMMARY_M PRIMARY KEY
  , meter_Wid INTEGER
  , monthNumber INTEGER
  , yearNumber INTEGER
  , consumption_M INTEGER
  , energyCost NUMBER(12,2)
);

/* W_METER_SUMMARY_M PK SEQUENCE*/
CREATE SEQUENCE seq_W_METER_SUM_M INCREMENT BY 1 START WITH 1;

/* W_METER_SUMMARY_M PK TRIGGER */
CREATE OR REPLACE TRIGGER Insert_W_METER_SUM_M
  BEFORE INSERT ON W_METER_SUMMARY_M
  FOR EACH ROW
BEGIN
  SELECT seq_W_METER_SUM_M.NEXTVAL INTO :NEW.wid FROM DUAL;
END Insert_W_METER_SUM_M;
```

Tabla Resumen de Contadores: W_CITY_METER_SUM_T

Creación de la tabla de resumen (summary table) de la jerarquía Contadores por ciudad con todo el consumo acumulado.

```
/*****************************************************************************/
/* CITY METER SUMMARY TOTAL                                                  */
/*****************************************************************************/
CREATE TABLE W_CITY_METER_SUM_T(
  wid INTEGER CONSTRAINT PK_W_CITY_METER_SUM_T PRIMARY KEY
  , city_Wid INTEGER
  , meter_Wid INTEGER
  , consumption_T INTEGER
  , energyCost NUMBER(12,2)
);

/* W_CITY_METER_SUM_T PK SEQUENCE*/
CREATE SEQUENCE seq_W_CITY_METER_SUM_T INCREMENT BY 1 START WITH 1;

/* W_CITY_METER_SUM_T PK TRIGGER */
CREATE OR REPLACE TRIGGER Insert_W_CITY_METER_SUM_T
  BEFORE INSERT ON  W_CITY_METER_SUM_T
  FOR EACH ROW
BEGIN
  SELECT seq_W_CITY_METER_SUM_T.NEXTVAL INTO :NEW.wid FROM DUAL;
END Insert_W_CITY_METER_SUM_T;
```