Anexo IX. Generación de datos de prueba en la BBDD transaccional para las pruebas del proceso ETL y consultas del DW

Generación de valores para la tabla CITY

Se generan 1000 valores de ciudad con nombres aleatorios

```
/*************************************************/
CREATE OR REPLACE PROCEDURE ETL_LOAD_CITY_DATA IS
/************************************************
 INSERT TEST DATA IN CITY TABLE FOR ETL PROCESS
*************************************************/
  Resultat VARCHAR2 (200);
  citycode NUMBER;
  cityname varchar2(40);
  zipcode varchar2(5);
  provincecode number;
BEGIN

FOR I In 1..1000 LOOP
  SELECT TRUNC(DBMS_RANDOM.VALUE(780000, 789999)) INTO CITYCODE FROM DUAL;
  SELECT DBMS_RANDOM.STRING('U',TRUNC(DBMS_RANDOM.VALUE(20,40))) INTO CITYNAME FROM DUAL;
  SELECT TO_CHAR(TRUNC(DBMS_RANDOM.VALUE(78100, 78999))) INTO ZIPCODE FROM DUAL;
  SELECT PROVINCEID INTO PROVINCECODE FROM (SELECT PROVINCEID FROM PROVINCE ORDER BY
dbms_random.VALUE) WHERE ROWNUM=1;

  INS_CITY(CITYCODE, CITYNAME, ZIPCODE , PROVINCECODE, Resultat);

END LOOP;
END;

/* EXECUTE DATA LOAD PROCEDURE IN CITY */
EXECUTE ETL_LOAD_CITY_DATA;
```

Generación de valores para la tabla ADDRESS (direcciones)

Se generan 1100 valores de dirección con datos aleatorios

```
/*************************************************/
CREATE OR REPLACE PROCEDURE ETL_LOAD_ADDRESS_DATA IS



BEGIN
/************************************************
 INSERT TEST DATA IN ADDRESS TABLE FOR ETL PROCESS
*************************************************/
FOR I IN 1..1100 LOOP
  INSERT INTO ADDRESS SELECT
      seq_Address.NEXTVAL
      , TRUNC(DBMS_RANDOM.VALUE(1,6)) AS ADDRESSCODE
      , DBMS_RANDOM.STRING('U',TRUNC(DBMS_RANDOM.VALUE(30,150))) AS STREETCODE
      , DBMS_RANDOM.STRING('U',5) AS STREETNAME
      , DBMS_RANDOM.STRING('U',10) AS ADDRESSNUMBER
      , DBMS_RANDOM.STRING('U',4) AS FLOOR
      , TO_CHAR(TRUNC(DBMS_RANDOM.VALUE(620000000, 690999999))) AS PHONENUMBER
      , (SELECT CITYCODE FROM (SELECT CITYCODE FROM CITY ORDER BY dbms_random.VALUE) WHERE
ROWNUM=1) AS CITYCODE
    FROM DUAL;
END LOOP;

END;

/* EXECUTE DATA LOAD PROCEDURE IN CITY */
EXECUTE ETL_LOAD_ADDRESS_DATA;
```

Generación de valores para la tabla CONSUMER (clientes)

Se generan 500 clientes con nombres aleatorios

```
/*****************************************************/
create or replace PROCEDURE ETL_LOAD_CONSUMER_DATA IS
    Resultat              VARCHAR2 (200);
    CONSUMERCODE          VARCHAR2 (200);

    CONSUMERNAME          VARCHAR2(100);
    CONSUMERSURNAME       VARCHAR2(100);
    SEX                   VARCHAR2(1);
    MOBILEPHONE           VARCHAR2(9);
    ADDRESSCODE1           NUMBER;
    IDENTITYCODE          NUMBER;
    IDENTIFICATIONNUMBER  VARCHAR2(50);
    BANKCODE1              VARCHAR2(11);
    ACCOUNTCODE           VARCHAR2(24);
BEGIN
/*************************************************
 INSERT TEST DATA IN CONSUMER TABLE FOR ETL PROCESS
*************************************************/
FOR I IN 1..500 LOOP
   SELECT DBMS_RANDOM.STRING('U',TRUNC(DBMS_RANDOM.VALUE(2,15))) INTO CONSUMERNAME FROM
DUAL;
   SELECT          (DBMS_RANDOM.STRING('U',TRUNC(DBMS_RANDOM.VALUE(5,15)))||'         '||
DBMS_RANDOM.STRING('U',TRUNC(DBMS_RANDOM.VALUE(5,15)))) INTO CONSUMERSURNAME FROM DUAL;
   SELECT (CASE WHEN TRUNC(DBMS_RANDOM.VALUE)=0 THEN 'M' ELSE 'F' END) INTO SEX FROM DUAL;
   SELECT TO_CHAR(TRUNC(DBMS_RANDOM.VALUE(620000000, 690999999))) INTO MOBILEPHONE FROM
DUAL;
   SELECT ADDRESSCODE INTO ADDRESSCODE1 FROM (SELECT ADDRESSCODE FROM ADDRESS ORDER BY
dbms_random.value) WHERE rownum = 1;
   SELECT DBMS_RANDOM.VALUE(1,3) INTO IDENTITYCODE  FROM DUAL;
   SELECT TO_CHAR(TRUNC(DBMS_RANDOM.VALUE(39690000, 39699999))) INTO IDENTIFICATIONNUMBER
FROM DUAL;
   SELECT BANKCODE INTO BANKCODE1 FROM (SELECT BANKCODE FROM BANK ORDER BY
dbms_random.value) WHERE rownum = 1;
   SELECT TO_CHAR(TRUNC(DBMS_RANDOM.VALUE(62000000000, 690999999999))) INTO ACCOUNTCODE
FROM DUAL;

   INS_CONSUMER(CONSUMERNAME,   CONSUMERSURNAME,   SEX,   MOBILEPHONE,   ADDRESSCODE1,
IDENTITYCODE, IDENTIFICATIONNUMBER, BANKCODE1, ACCOUNTCODE, RESULTAT, CONSUMERCODE);
END LOOP;
END;


/* EXECUTE DATA LOAD PROCEDURE IN consumer */
EXECUTE ETL_LOAD_CONSUMER_DATA;
```

Generación de valores para la tabla Company

Se generan 50 compañías con nombres aleatorios.

```
/*****************************************************/
create or replace PROCEDURE ETL_LOAD_COMPANY_DATA IS
    Resultat       VARCHAR2 (200);

    COMPANYTAXCODE VARCHAR2(15);
    COMPANYNAME    VARCHAR2(200);
    ADDRESSCODE1    NUMBER;

BEGIN
/*************************************************
 INSERT TEST DATA IN COMPANY TABLE FOR ETL PROCESS
*************************************************/
FOR I IN 1..50 LOOP
   SELECT  ('ESA'  ||  TO_CHAR(TRUNC(DBMS_RANDOM.VALUE(1000000,1500000)))  ||  'X')  INTO
COMPANYTAXCODE FROM DUAL;
   SELECT DBMS_RANDOM.STRING('U',TRUNC(DBMS_RANDOM.VALUE(10,15))) INTO COMPANYNAME FROM
DUAL;
```

```
   SELECT ADDRESSCODE INTO ADDRESSCODE1 FROM (SELECT ADDRESSCODE FROM ADDRESS ORDER BY
dbms_random.value) WHERE rownum = 1 AND ADDRESSCODE NOT IN (SELECT ADDRESSCODE FROM
CONSUMER) ;

   INS_COMPANY(COMPANYTAXCODE, COMPANYNAME, ADDRESSCODE1, RESULTAT);
END LOOP;
END;


/* EXECUTE DATA LOAD PROCEDURE IN CITY */
EXECUTE ETL_LOAD_COMPANY_DATA;
```

Valores de prueba para la tabla METER

Se generan 500 contadores para las pruebas con valores aleatoriamente generados de nombre, fecha de instalación, fecha de inspección, etc…

```
/*****************************************************/
create or replace PROCEDURE ETL_LOAD_METER_DATA IS
    Resultat       VARCHAR2 (200);

    SERIALNUMBER  VARCHAR2(20);
    METERMODEL    VARCHAR2(100);
    CONTRACTCODE  VARCHAR2(100);
    CONTRACTEDPOWER NUMBER(5,2);
    LASTTECHNICALINSPECTION DATE;
    INSTALLATIONDATE DATE;
    COMPANYCODE1 VARCHAR2(15);
    CONSUMERCODE1 NUMBER;
    ADDRESSCODE1 NUMBER;


BEGIN
/*****************************************************
 INSERT TEST DATA IN COMPANY TABLE FOR ETL PROCESS
*****************************************************/
FOR I IN 1..500 LOOP
   SELECT TO_CHAR(TRUNC(DBMS_RANDOM.VALUE(600000000,700000000))) INTO SERIALNUMBER FROM
DUAL;
   SELECT DBMS_RANDOM.STRING('U',TRUNC(DBMS_RANDOM.VALUE(20,70))) INTO METERMODEL FROM
DUAL;
   SELECT TO_CHAR(TRUNC(DBMS_RANDOM.VALUE(600000000,700000000))) INTO CONTRACTCODE FROM
DUAL;
   SELECT DBMS_RANDOM.VALUE(4,5) INTO CONTRACTEDPOWER FROM DUAL;
   SELECT   TO_DATE(TRUNC(DBMS_RANDOM.VALUE(TO_CHAR(DATE   '2013-01-01','J'),TO_CHAR(DATE
'9999-12-31','J'))),'J') INTO LASTTECHNICALINSPECTION FROM DUAL;
   SELECT   TO_DATE(TRUNC(DBMS_RANDOM.VALUE(TO_CHAR(DATE   '2013-01-01','J'),TO_CHAR(DATE
'9999-12-31','J'))),'J') INTO INSTALLATIONDATE FROM DUAL;
   SELECT COMPANYTAXCODE INTO COMPANYCODE1 FROM (SELECT COMPANYTAXCODE FROM COMPANY ORDER
BY dbms_random.value) WHERE rownum = 1;
   SELECT CONSUMERCODE INTO CONSUMERCODE1 FROM (SELECT CONSUMERCODE FROM CONSUMER ORDER
BY dbms_random.value) WHERE rownum = 1;
   SELECT ADDRESSCODE INTO ADDRESSCODE1 FROM (SELECT ADDRESSCODE FROM ADDRESS ORDER BY
dbms_random.value) WHERE rownum = 1 AND ADDRESSCODE NOT IN (SELECT ADDRESSCODE FROM
CONSUMER) AND ADDRESSCODE NOT IN (SELECT ADDRESSCODE FROM COMPANY) ;

   INS_METER(SERIALNUMBER,       METERMODEL,       CONTRACTCODE,       CONTRACTEDPOWER,
LASTTECHNICALINSPECTION,  INSTALLATIONDATE,  COMPANYCODE1,CONSUMERCODE1,  ADDRESSCODE1,
RESULTAT);
END LOOP;
END;


/* EXECUTE DATA LOAD PROCEDURE IN CITY */
EXECUTE ETL_LOAD_METER_DATA;
```

Valores de prueba para la tabla PRICE

Se generan 3000 cambios de precio

Ahora es necesario modificar el SP INS_PRICE para que nos permita grabar precios pasados (la SP tiene un control que solo permite introducir precios con fechas del futuro, para evitar modificaciones engañosas)

En el siguiente código mostraremos el nuevo procedimiento para introducir precios con el control de fecha desactivado (INS_PRICE_PAST)

```
create or replace PROCEDURE Ins_Price_Past (
/*********************************************
| 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...
  /* Modified
  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;
```

En primer lugar es necesario crear una “baseline”, es decir un periodo de fechas en el que todas las compañías tienen precio.

```
/*******************************************************/
Create or replace PROCEDURE   ETL_LOAD_PRICE_DATA_BASELINE(
    DATEINI IN DATE
 ) IS
Resultat       VARCHAR2 (200);

    CHANGEDATA  DATE;
    NEWPRICE    NUMBER;

BEGIN
/*************************************************
 INSERT TEST DATA IN PRICE TABLE FOR ETL PROCESS
*************************************************/
  FOR comp IN (SELECT COMPANYTAXCODE FROM COMPANY)
  LOOP
    FOR countr IN (SELECT COUNTRYCODE FROM COUNTRY)
    LOOP
      SELECT DATEINI INTO CHANGEDATA FROM DUAL;
      SELECT DBMS_RANDOM.VALUE INTO NEWPRICE FROM DUAL;
      NEWPRICE:=NEWPRICE+1;
```

```
      INS_PRICE_past(CHANGEDATA,   COUNTR.COUNTRYCODE,   COMP.COMPANYTAXCODE,   NEWPRICE,
RESULTAT);

    END LOOP;
  END LOOP;
END;

EXECUTE ETL_LOAD_PRICE_DATA_BASELINE(to_date('01/01/2013', 'DD/MM/YYYY'));
```

Ahora ya podemos empezar a generar valores de cambio de precio de forma aleatoria entre los periodos 02/01/2013 y 31/12/2015.

```
/*****************************************************/
create or replace PROCEDURE ETL_LOAD_PRICE_DATA(
      DATEINI IN DATE
    , DATEFIN IN DATE
)IS
    Resultat       VARCHAR2 (200);

    CHANGEDATA  DATE;
    COUNTRYCODE1    NUMBER;
    COMPANYCODE1 VARCHAR2(15);
    NEWPRICE    NUMBER;

BEGIN
/*************************************************
 INSERT TEST DATA IN PRICE TABLE FOR ETL PROCESS
*************************************************/
FOR I IN 1..3000 LOOP
   SELECT
TO_DATE(TRUNC(DBMS_RANDOM.VALUE(TO_CHAR(DATEINI,'J'),TO_CHAR(DATEFIN,'J'))),'J')     INTO
CHANGEDATA FROM DUAL;
   SELECT COUNTRYCODE INTO COUNTRYCODE1 FROM (SELECT COUNTRYCODE FROM COUNTRY ORDER BY
dbms_random.value) WHERE rownum = 1;
   SELECT COMPANYTAXCODE INTO COMPANYCODE1 FROM (SELECT COMPANYTAXCODE FROM COMPANY ORDER
BY dbms_random.value) WHERE rownum = 1;
   SELECT DBMS_RANDOM.VALUE INTO NEWPRICE FROM DUAL;
   NEWPRICE:=NEWPRICE+1;
   DBMS_OUTPUT.PUT_LINE(NEWPRICE);

   INS_PRICE_PAST(CHANGEDATA, COUNTRYCODE1, COMPANYCODE1, NEWPRICE, RESULTAT);
END LOOP;
END;



/* EXECUTE DATA LOAD PROCEDURE IN PRICE */
EXECUTE  ETL_LOAD_PRICE_DATA(to_date('02/01/2013',  'DD/MM/YYYY'),  to_date('31/12/2015',
'DD/MM/YYYY'));
```

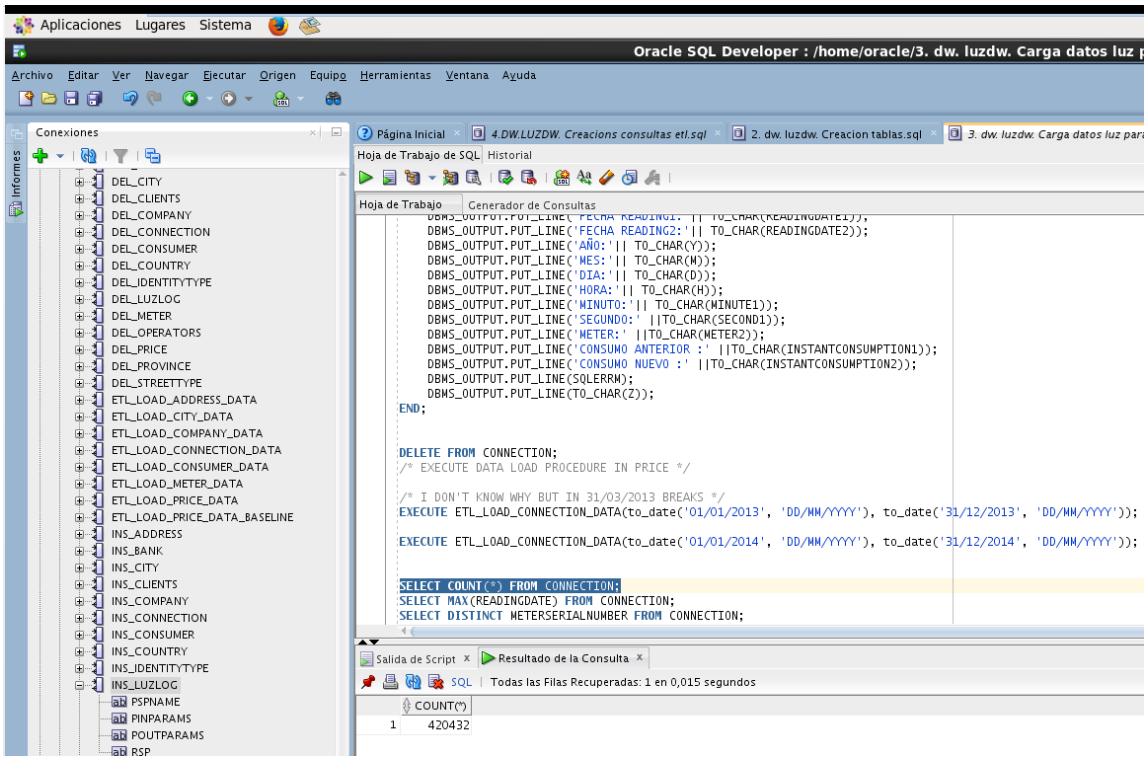
Valores de prueba para la tabla CONNECTION (consumos)

Se generan los datos de lecturas de un año para los 30 contadores (30 lecturas de contador diarias, para generar valores prohibidos sin consumo).

La rutina tiene como entrada una fecha inicial y una fecha final, para generar lecturas en ese intervalo. De los parámetros de entrada solo recoge el mes y el año, porque siempre se generan valores entre el día 1 y el 31 (o 28 o 30 según el mes).

Se generan 30 lecturas al día en intervalos de tiempo aleatorios, con la intención de sobrepasar el máximo de una lectura por hora y generar lecturas incorrectas (con valor de consumo a 0). Se Recoge el valor de la ultima lectura del contador para generar un valor, aleatorio también, de consumo (superior a la anterior lectura)

En total se han generado unos 420.432 registros (tiempo aproximado de 98 minutos de carga)



Posteriormente se ha reducido el juego de pruebas con el siguiente procedimiento:

```
/* EXECUTE DATA LOAD PROCEDURE IN PRICE */
EXECUTE  ETL_LOAD_PRICE_DATA(to_date('02/01/2013',  'DD/MM/YYYY'),  to_date('31/12/2015',
'DD/MM/YYYY'));


Create or replace PROCEDURE ETL_LOAD_CONNECTION_DATA (
      DATEINI IN DATE
    , DATEFIN IN DATE
)IS
    Resultat            VARCHAR2 (200);

    READINGDATE1        VARCHAR2(100);
    READINGDATE2        TIMESTAMP;
    MINUTE1             VARCHAR2(3);
    SECOND1             VARCHAR2(3);
    METER2              VARCHAR2(20);
    INSTANTCONSUMPTION1 NUMBER(38,0);
    INSTANTCONSUMPTION2 NUMBER(5,3);
```

```
    Y                   NUMBER;
    Y1                  NUMBER;
    Y2                  NUMBER;
    M                   NUMBER;
    M1                  NUMBER;
    M2                  NUMBER;
    D1                  NUMBER;
    DMAX                NUMBER;
    H                   NUMBER;
    h1                  number;
    D                   NUMBER;
    Z                   NUMBER;
BEGIN
/************************************************
 INSERT TEST DATA IN PRICE TABLE FOR ETL PROCESS
************************************************/
      Z:=1;
      FOR METER1 IN (SELECT SERIALNUMBER FROM METER WHERE ROWNUM<50) LOOP
        METER2:=METER1.SERIALNUMBER;
        --YEAR LOOP
        Y1:=TO_NUMBER(TO_CHAR(DATEINI, 'YYYY'));
        Y2:=TO_NUMBER(TO_CHAR(DATEFIN, 'YYYY'));

        FOR Y IN Y1..Y2 LOOP
          --MONTH LOOP
          M1:=TO_NUMBER(TO_CHAR(DATEINI, 'MM'));
          M2:=TO_NUMBER(TO_CHAR(DATEFIN, 'MM'));

          FOR M IN M1..M2 LOOP
            --DAY LOOP
            SELECT (CASE
                WHEN M IN (4,6,9,11) THEN 30
                WHEN M IN (1,3,5,7,8,10,12) THEN 31
                WHEN M=2 THEN 28
              END)
              INTO DMAX
            FROM DUAL;

            D:=1;
            while  (D<DMAX) LOOP
              /*factor aleatorio*/
                if d=3 then
                  d:=d+d1;
                end if;


              -- HOUR LOOP
              H:=0;
              WHILE H<24 LOOP
                /*factor aleatorio*/
                if h=3 then
                  h:=h+h1;
                end if;

                -- SECONDS CALC
                SELECT TO_CHAR(TRUNC(DBMS_RANDOM.VALUE(0,60))) INTO SECOND1 FROM DUAL;

                -- MINUTE CALC
                SELECT TO_CHAR(TRUNC(DBMS_RANDOM.VALUE(0,60))) INTO MINUTE1 FROM DUAL;

                --CONCAT DATE
                READINGDATE1:=TO_CHAR(Y) || '-' || TO_CHAR(M) || '-' || TO_CHAR(D) || ' '
|| TO_CHAR(H) || ':' || MINUTE1 || ':' || SECOND1;
                READINGDATE2:=TO_TIMESTAMP(READINGDATE1,'YYYY-MM-DD HH24:MI:SS');

                --LAST CONSUMPTION CALC
                BEGIN
                  SELECT INSTANTCONSUMPTION INTO INSTANTCONSUMPTION1 FROM (SELECT
INSTANTCONSUMPTION FROM CONNECTION WHERE READINGDATE<READINGDATE2 AND ROWNUM=1 AND
METERSERIALNUMBER=METER1.SERIALNUMBER AND INSTANTCONSUMPTION IS NOT NULL ORDER BY
READINGDATE DESC) ;
                EXCEPTION
                  WHEN NO_DATA_FOUND THEN
                    INSTANTCONSUMPTION1:=0;
                  WHEN others THEN
                    INSTANTCONSUMPTION1:=0;
                END;
```

```
                SELECT DBMS_RANDOM.VALUE INTO INSTANTCONSUMPTION2 FROM DUAL;
                INSTANTCONSUMPTION1:=NVL(INSTANTCONSUMPTION1,0)+INSTANTCONSUMPTION2*100;
                begin
                  INS_CONNECTION(READINGDATE2, METER1.SERIALNUMBER, INSTANTCONSUMPTION1,
'Y', RESULTAT);
                exception
                  when others then
                    DBMS_OUTPUT.PUT_LINE('es un error de la ins');
                end;

                COMMIT;

                select DBMS_RANDOM.VALUE(1,2) into h1 from dual;

                H:=H+5;
                --DBMS_OUTPUT.PUT_LINE('bucle  hora  '  ||TO_CHAR(READINGDATE1)  ||'  '
||TO_CHAR(READINGDATE2));
              END LOOP;

              select DBMS_RANDOM.VALUE(1,2) into d1 from dual;

              d:=d+5;
              --DBMS_OUTPUT.PUT_LINE('bucle  dia  '  ||TO_CHAR(READINGDATE1)  ||'  '
||TO_CHAR(READINGDATE2));

            END LOOP;

            --DBMS_OUTPUT.PUT_LINE('bucle  mes  '  ||TO_CHAR(READINGDATE1)  ||'  '
||TO_CHAR(READINGDATE2));
          END LOOP;
          --DBMS_OUTPUT.PUT_LINE('bucle  año  '  ||TO_CHAR(READINGDATE1)  ||'  '
||TO_CHAR(READINGDATE2));
        END LOOP;
        --DBMS_OUTPUT.PUT_LINE('bucle  meter  '  ||TO_CHAR(READINGDATE1)  ||'  '
||TO_CHAR(READINGDATE2));
        Z:=Z+1;
      END LOOP;
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('error:');
    DBMS_OUTPUT.PUT_LINE('FECHA READING1:'|| TO_CHAR(READINGDATE1));
    DBMS_OUTPUT.PUT_LINE('FECHA READING2:'|| TO_CHAR(READINGDATE2));
    DBMS_OUTPUT.PUT_LINE('AÑO:'|| TO_CHAR(Y));
    DBMS_OUTPUT.PUT_LINE('MES:'|| TO_CHAR(M));
    DBMS_OUTPUT.PUT_LINE('DIA:'|| TO_CHAR(D));
    DBMS_OUTPUT.PUT_LINE('HORA:'|| TO_CHAR(H));
    DBMS_OUTPUT.PUT_LINE('MINUTO:'|| TO_CHAR(MINUTE1));
    DBMS_OUTPUT.PUT_LINE('SEGUNDO:' ||TO_CHAR(SECOND1));
    DBMS_OUTPUT.PUT_LINE('METER:' ||TO_CHAR(METER2));
    DBMS_OUTPUT.PUT_LINE('CONSUMO ANTERIOR :' ||TO_CHAR(INSTANTCONSUMPTION1));
    DBMS_OUTPUT.PUT_LINE('CONSUMO NUEVO :' ||TO_CHAR(INSTANTCONSUMPTION2));
    DBMS_OUTPUT.PUT_LINE(SQLERRM);
    DBMS_OUTPUT.PUT_LINE(TO_CHAR(Z));
END;
```

Mediante este procedimiento se ha ejecutado la orden:

```
EXECUTE           ETL_LOAD_CONNECTION_DATA(to_date('01/01/2013',           'DD/MM/YYYY'),
to_date('31/12/2014', 'DD/MM/YYYY'));
```

Para la generación de un juego de pruebas de unos 35.280 registros