Elaborado por:	Jaime López Patiño		
Cliente	ITS Solutions S.A.S.		
Objetivo:	Capacitar a los funcionarios de ITS en las buenas prácticas de Store Procedure		
Sistema de información:	Plataforma Iseries	**Plataforma:**	Iseries
Unidad de Negocio:	Área de tecnología	**Centro de Costo:**	
Usuario Contacto:	Jaime López Patiño jlopez@itstrategys.com	**Teléfono - Ext.:**	

1 Gestión de versiones

Versión	Especificado por	Revisado por	Descripción	Fecha
1	Jaime López		Elaborar documento	2020/08/27
2	Jaime López		Actualizar documento	2020/08/31

2 Definiciones, acrónimos y abreviaturas

Definiciones, acrónimos y abreviaturas	
Plataforma Iseries	Corresponde al sistema AS/400 que es un equipo de IBM de gama media y alta, para todo tipo de empresas y grandes departamentos. Se trata de un sistema multiusuario, con una interfaz controlada mediante menús y comandos CL intuitivos que utiliza terminales y un sistema operativo basado en objetos y bibliotecas, denominado OS/400.
Stored Procedures (SP)	Procedimiento almacenado es un programa el cual es almacenado físicamente en una base de datos. Generalmente son escritos en un lenguaje de bases de datos.

3 Alumnos

El equipo que conforma la capacitación son los funcionarios asignados al cliente Banco Finandina:

1. Álvaro Santamaría
2. Cesar Ricardo López
3. Flor Aydeé Avellaneda
4. Jorge Martínez

Cada persona tiene un conocimiento del uso de las sentencias SQL.

4 Temas explicados

El equipo informó que el área de tecnología de Banco Finandina realizó la inducción a los funcionarios de ITS sobre los SP para ser ejecutados desde el software de Navigator e invocar programas de RPG.

Por solicitud del equipo, se realiza la capacitación para realizar SP con sentencias estándares de SQL.

4.1 Definición

Un procedimiento almacenado (SP) es un programa o procedimiento el cual es almacenado físicamente en una base de datos. Generalmente son escritos en un lenguaje de bases de datos. La ventaja de un procedimiento almacenado es que, al ser ejecutado, en respuesta a una petición de usuario, es ejecutado directamente en el motor de bases de datos, el cual usualmente se ejecuta en un servidor separado. Como tal posee acceso directo a los datos que necesita

manipular y sólo necesita enviar sus resultados de regreso al usuario, deshaciéndose de la sobre carga resultante de comunicar grandes cantidades de datos salientes y entrantes.

4.2 Usos

Los SP son generalmente utilizados para lo siguiente:

- Validación de datos integrados a la base de datos, este tipo de uso es ejecutado desde desencadenantes o Triggers.

- Encapsular un proceso grande y complejo en lugar de un programa ejecutándose de forma permanente.

- Control de gestión de operaciones para ejecutar SP dentro de una transacción de tal manera que las transacciones sean efectivamente transparentes para ellos.

4.3 Ventajas

- Los SP están bajo el control del motor del manejador de la base de datos, el cual se ejecuta en un servidor separado.

- Los SP permiten que la lógica del negocio se encuentre como un API en la base de datos, que pueden simplificar la gestión de los datos y reducir la necesidad de codificar la lógica en el resto de los programas.

- Reduce la posibilidad de que los datos sean alterados por el uso de programas externos.

- El motor de la base de datos puede asegurar la integridad de los datos y la consistencia, con la ayuda de SP.

4.4 Tipos

Para la plataforma Iseries hay dos tipos de SP:

4.4.1 Externo

Es un SP para ejecutar programas de alto nivel como RPG, Java, C, etc.

4.4.2 Interno

- Es un SP para ejecutar sentencias estándares de SQL.
- Permite crear, modificar o eliminar tablas.
- Permite crear, modificar o eliminar registros de tablas.

4.5 Ejecución

- Los SP se invocan desde SQL, por medio de instrucciones CALL.

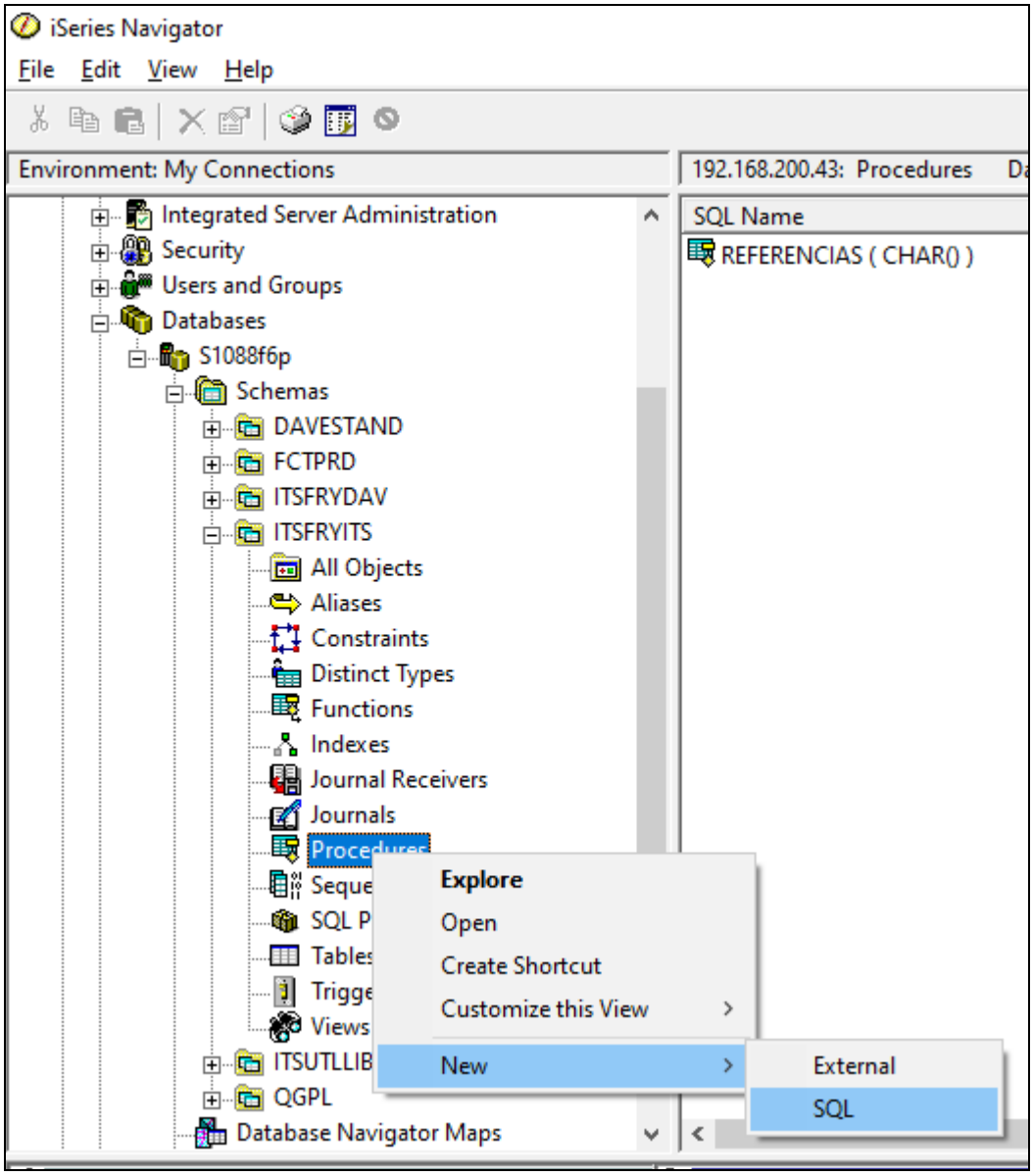
4.6 Implementar SP

La sentencia CREATE PROCEDURE permite al usuario crear un procedimiento, especificando los parámetros que se han de pasar al procedimiento, sus atributos, el nombre del programa que corresponde al procedimiento y el lenguaje en que esta escrito el procedimiento. Se puede hacer referencia al procedimiento en una sentencia CALL para invocar el SP.

4.7 Crear utilizando navigator Iseries

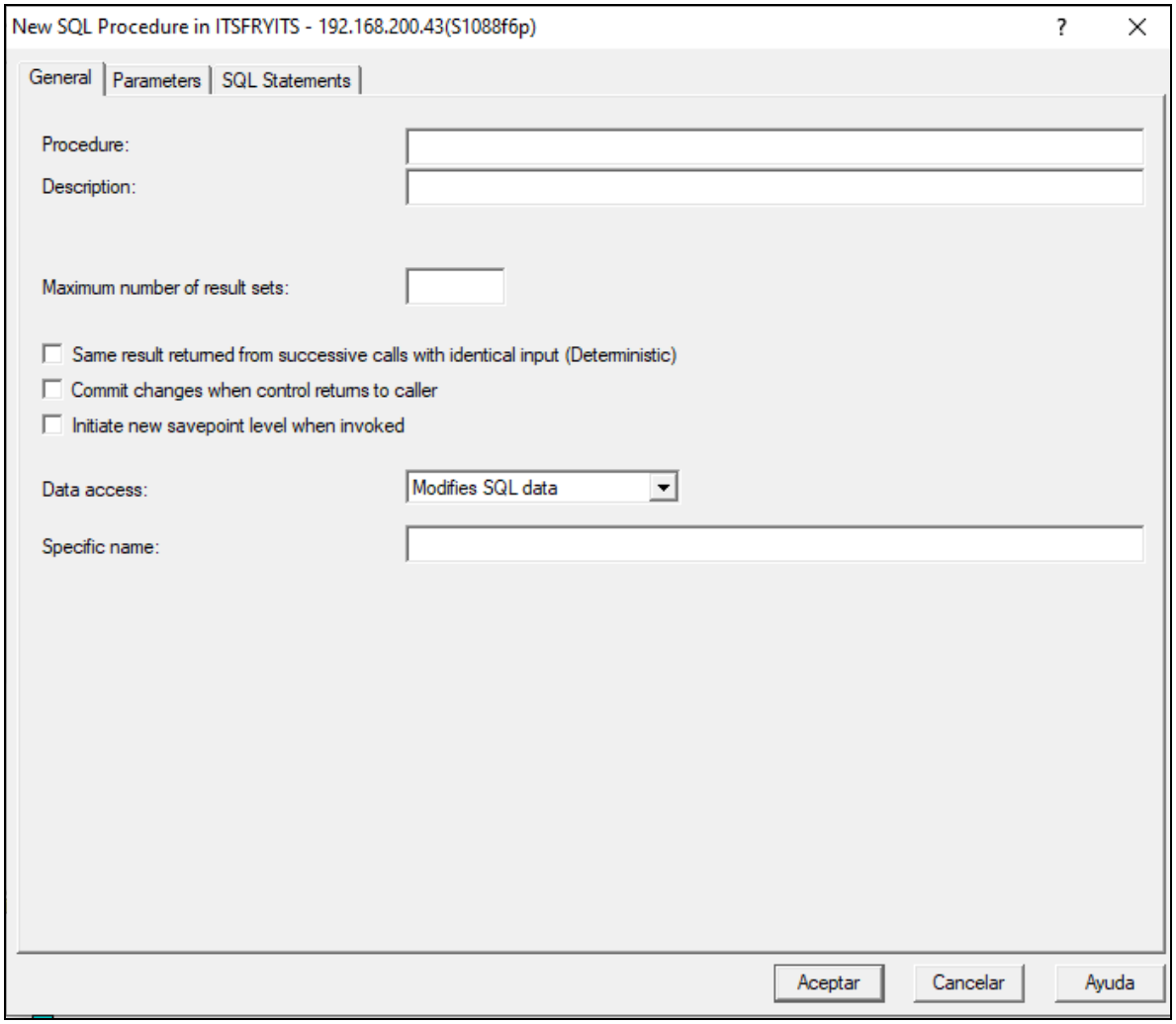
Las siguientes figuras muestra como se crean procedimientos desde el Navigator Iseries:

Se ubica la biblioteca de datos, seleccionamos el servicio Procedimientos:

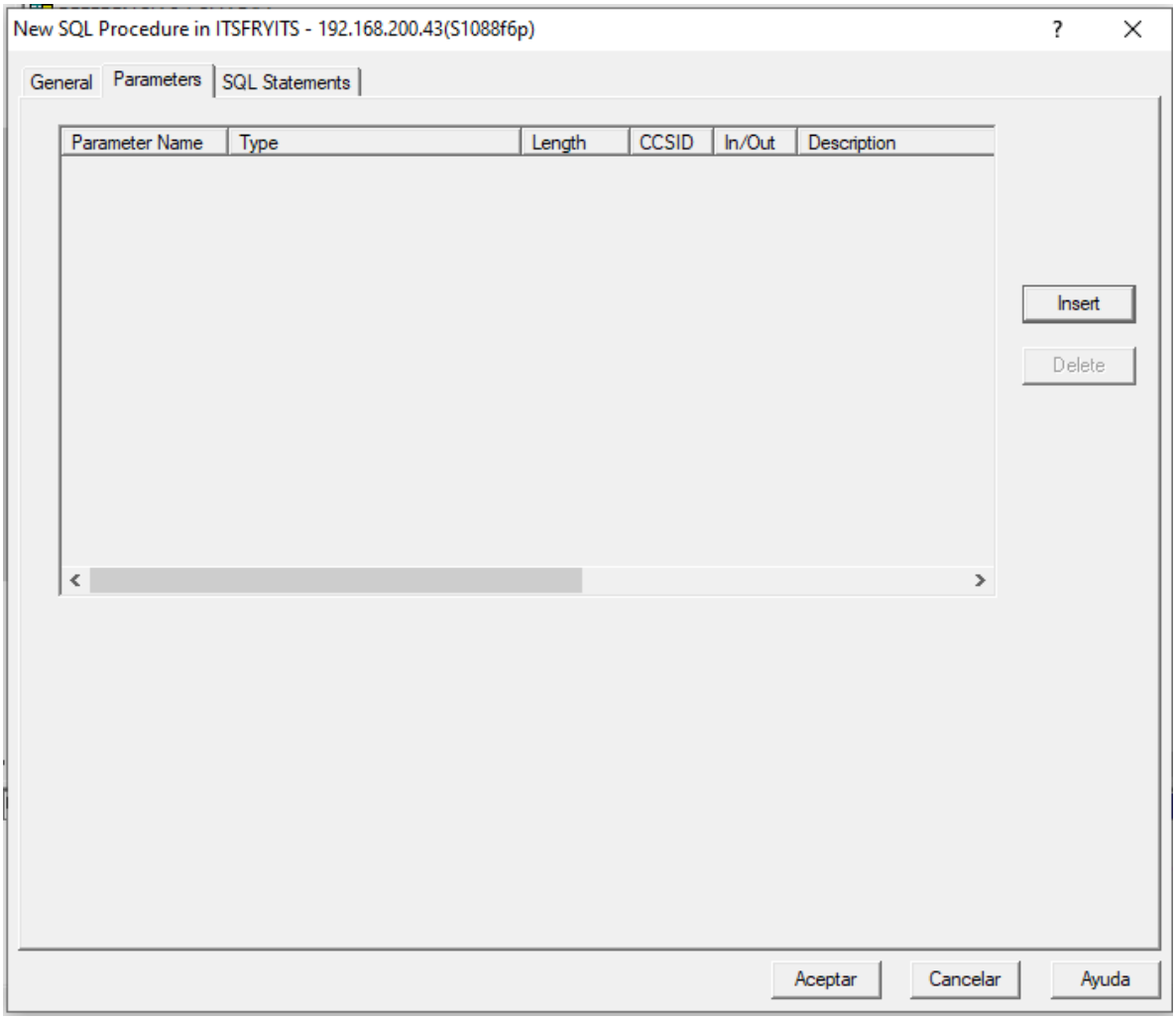


Se despliega la siguiente pantalla:

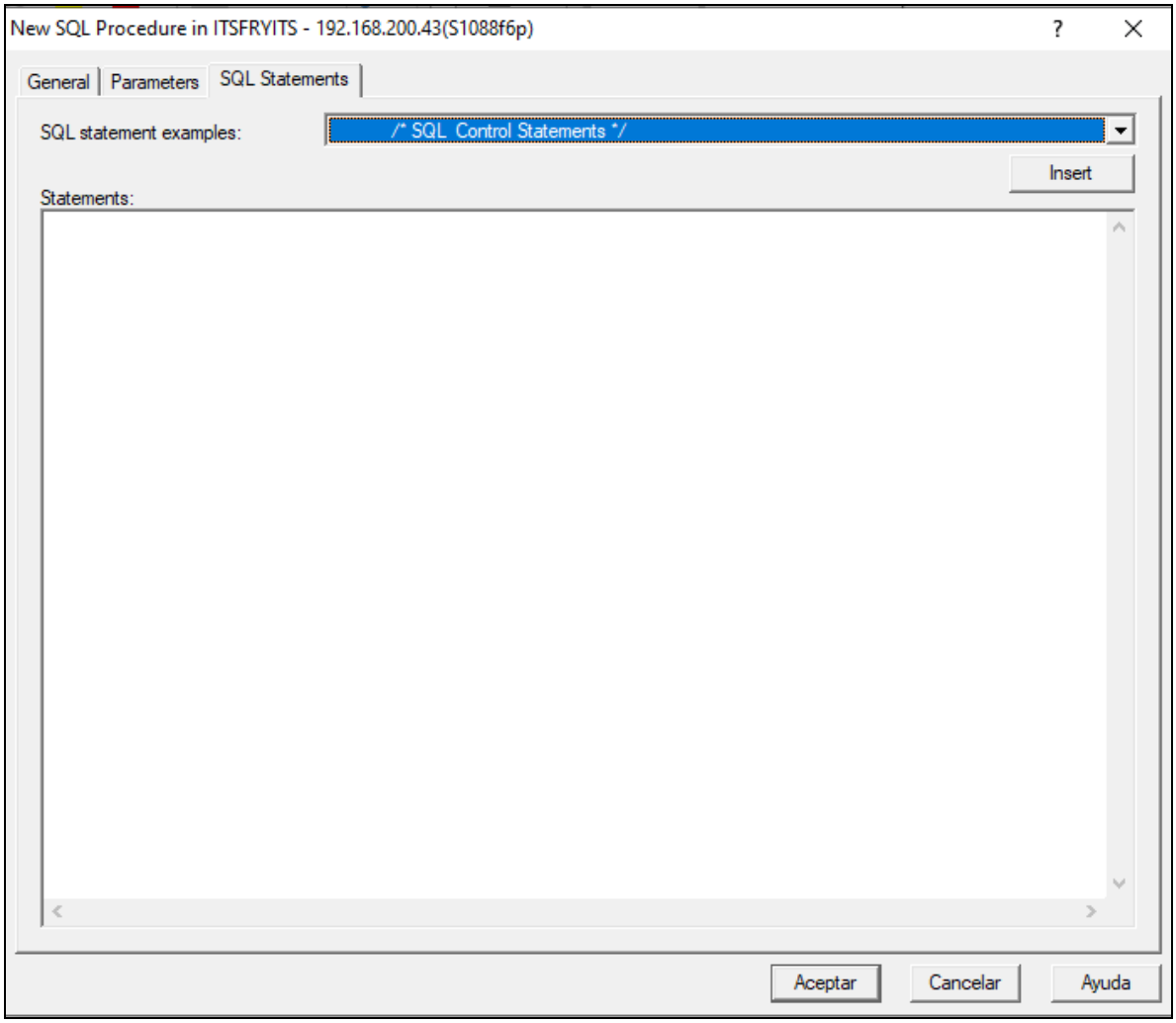
Pestaña General:



Pestaña Parámetros



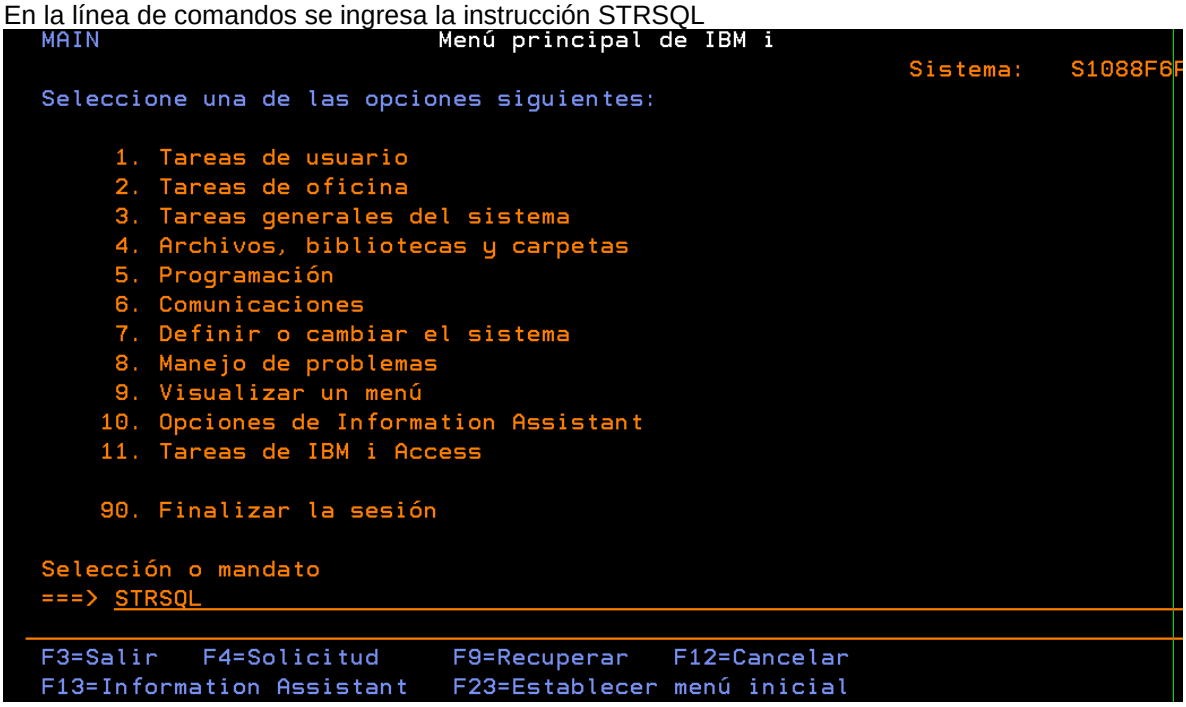
Pestaña Sentencia



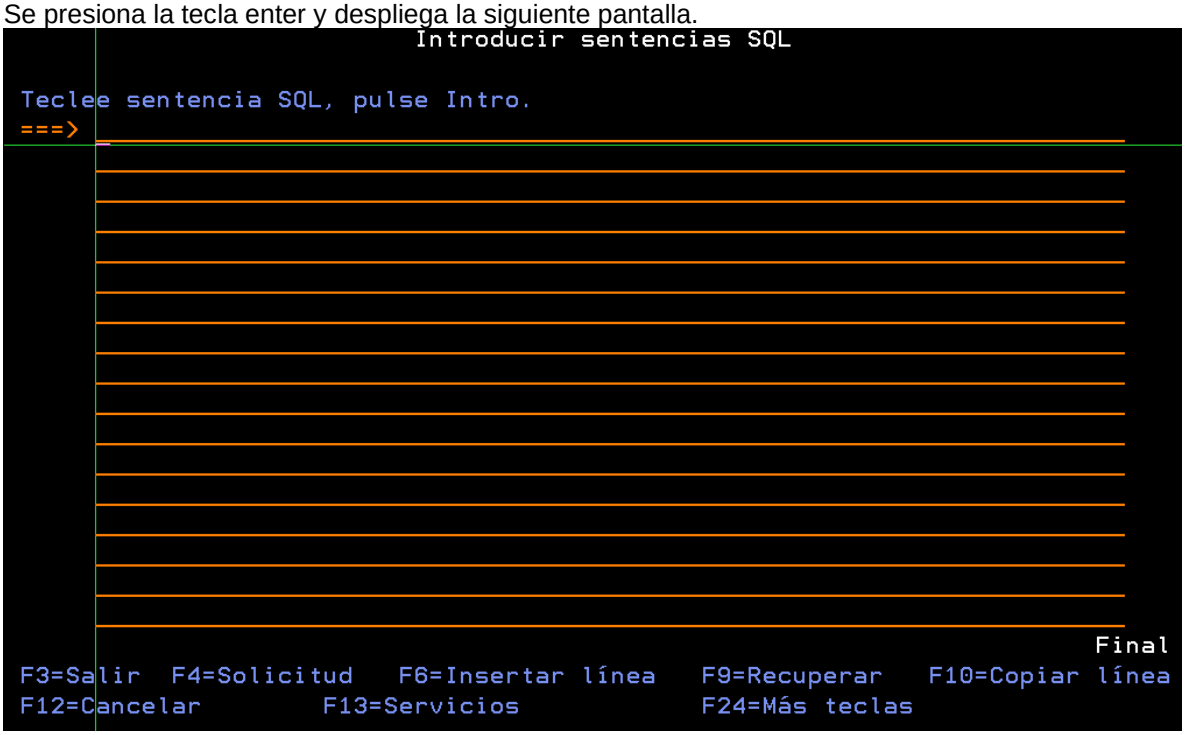
4.8 Crear utilizando emulador del Iseries

Las siguientes figuras muestra cómo se crean procedimientos desde el emulador Iseries:

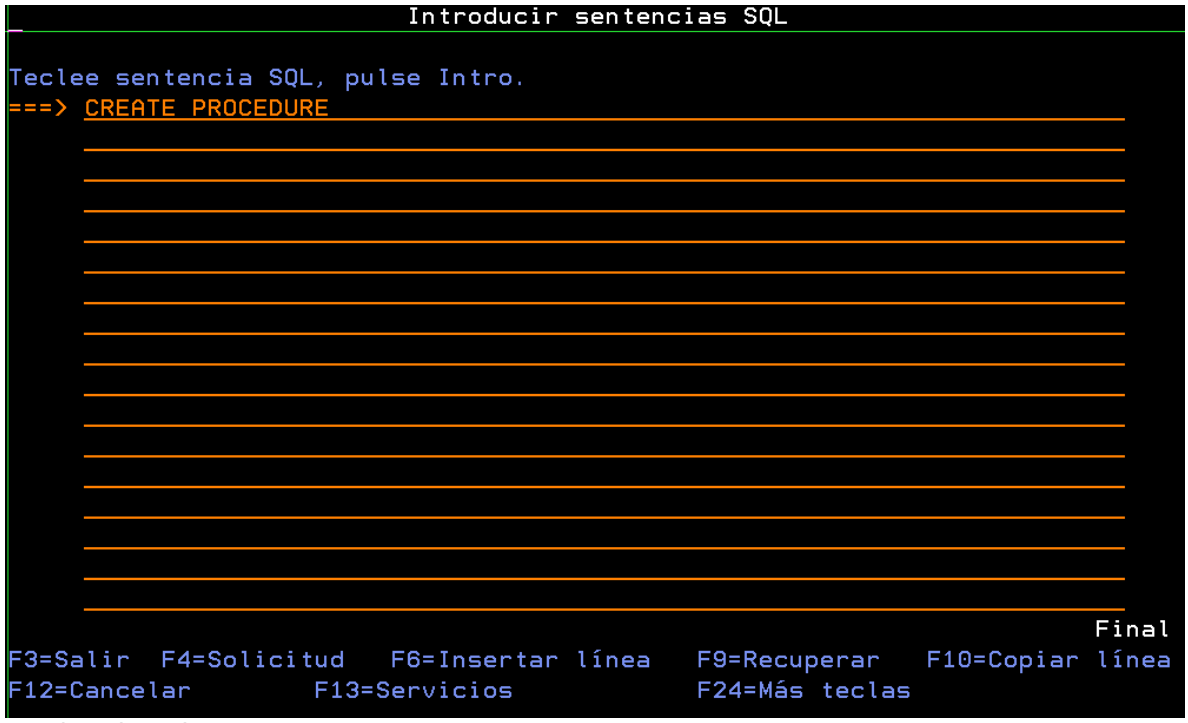
En la línea de comandos se ingresa la instrucción STRSQL



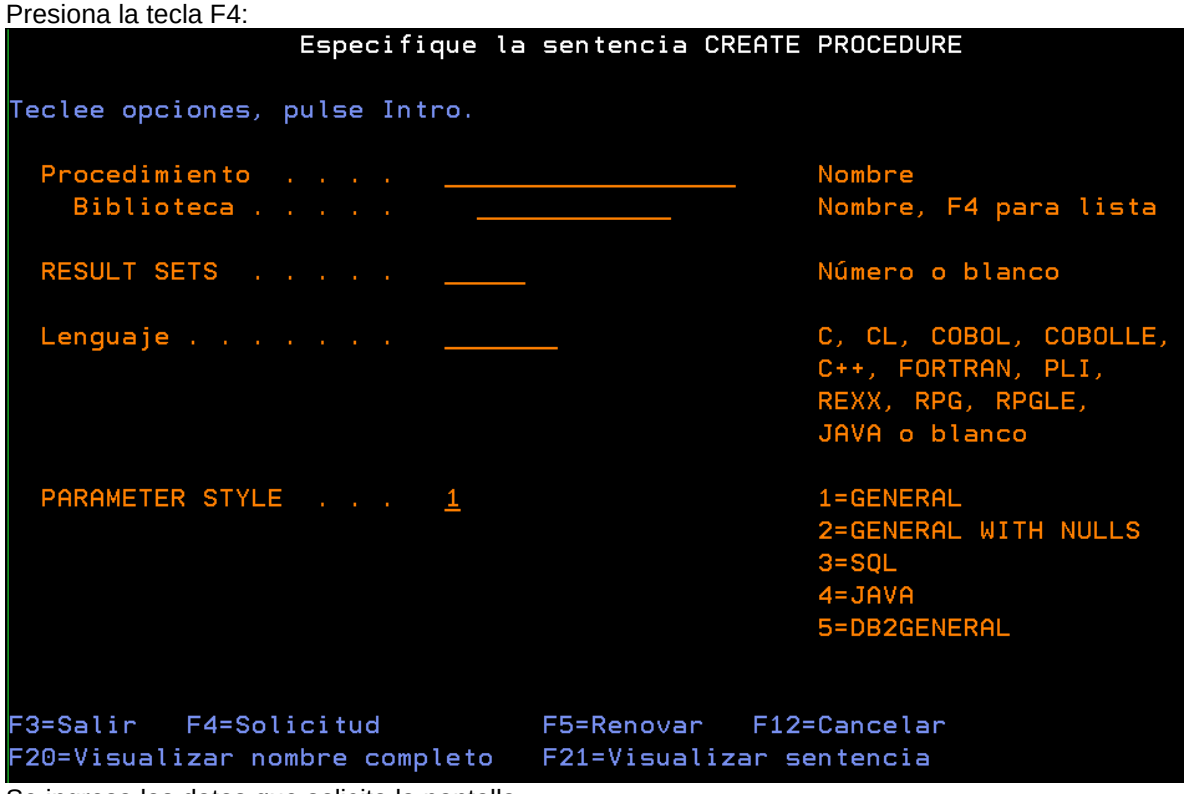
Se presiona la tecla enter y despliega la siguiente pantalla.



Se ingresa la instrucción CREATE PROCEDURE:



Presiona la tecla F4:



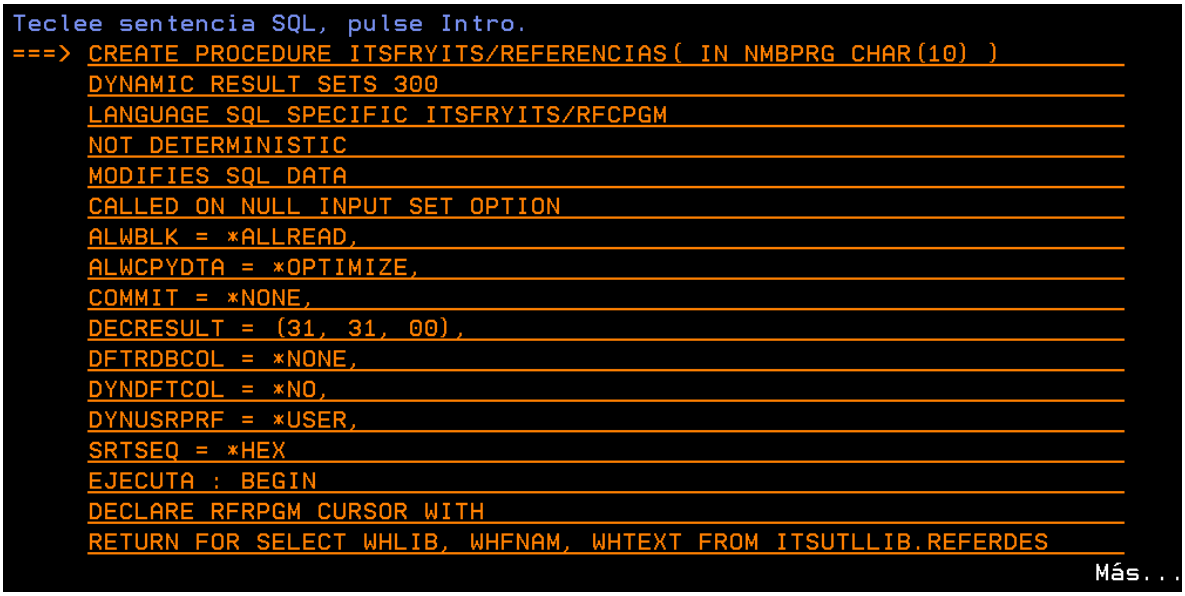
Se ingresa los datos que solicita la pantalla.

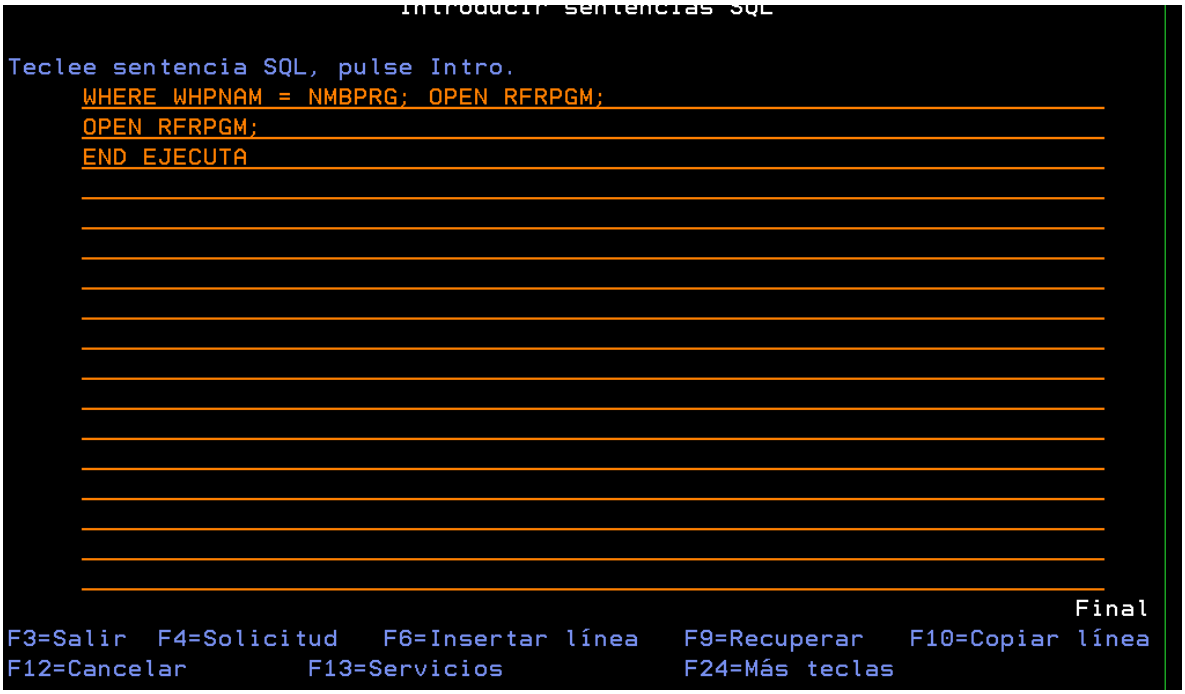
4.9 Ejemplo de SP con el emulador del Iseries

En un archivo en formato y extensión texto se diligencia la sentencia SQL, por ejemplo:

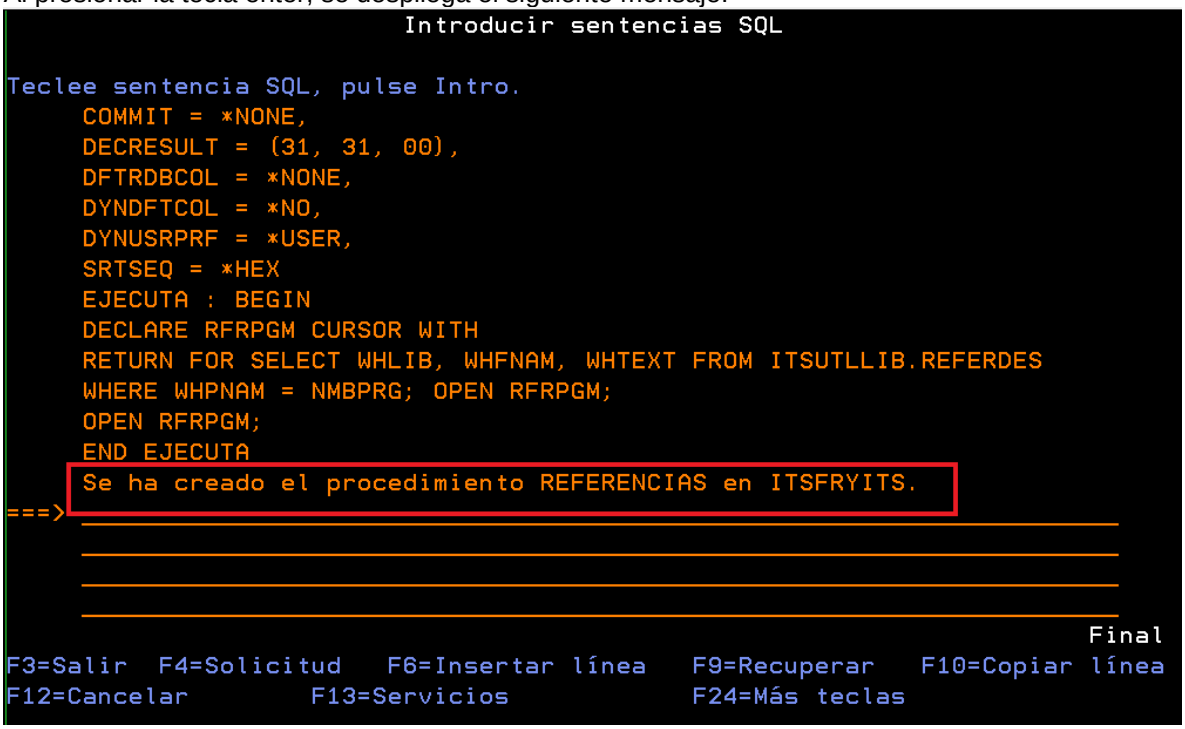
```
CREATE PROCEDURE ITSFRYITS/REFERENCIAS( IN NMBPRG CHAR(10) )
DYNAMIC RESULT SETS 300
LANGUAGE SQL SPECIFIC ITSFRYITS/RFCPGM
NOT DETERMINISTIC
MODIFIES SQL DATA
CALLED ON NULL INPUT SET OPTION
ALWBLK = *ALLREAD,
ALWCPYDTA = *OPTIMIZE,
COMMIT = *NONE,
DECRESULT = (31, 31, 00),
DFTRDBCOL = *NONE,
DYNDFTCOL = *NO,
DYNUSRPRF = *USER,
SRTSEQ = *HEX
EJECUTA: BEGIN
DECLARE RFRPGM CURSOR WITH
RETURN FOR SELECT WHLIB, WHFNAM, WHTEXT FROM ITSUTLLIB/REFERDES
WHERE WHPNAM = NMBPRG;
OPEN RFRPGM;
END EJECUTA
```

Se ingresa al Iseries, se digita el comando STRSQL, se presiona la tecla enter, se digita las anteriores sentencias:





Al presionar la tecla enter, se despliega el siguiente mensaje:

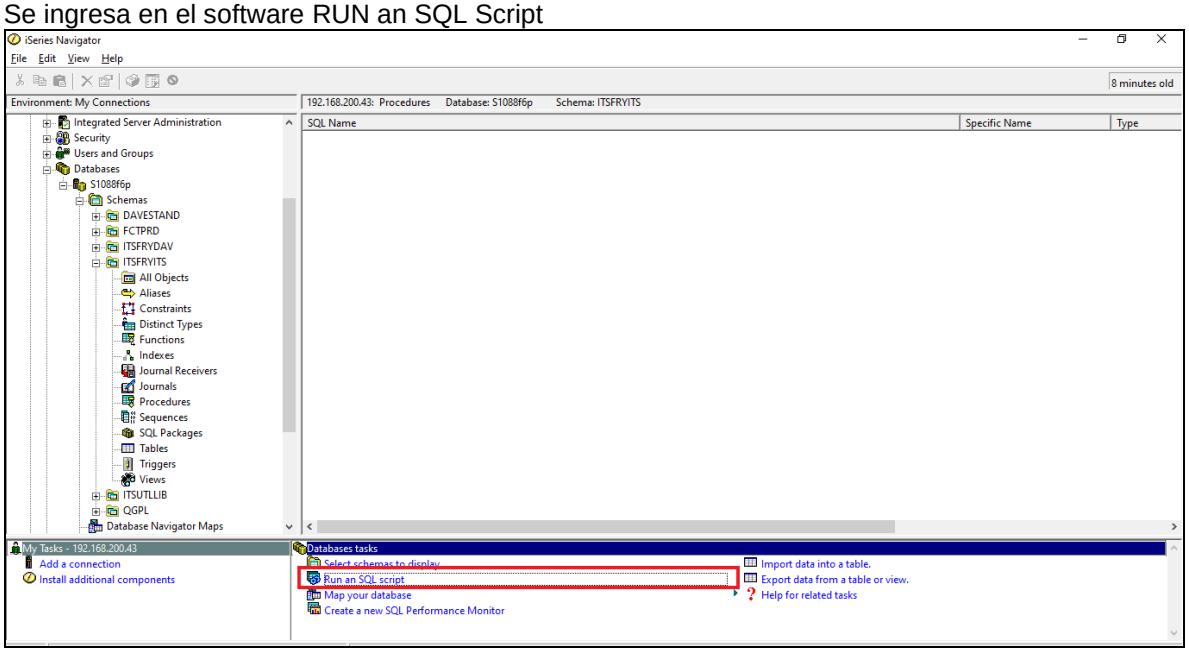


4.10 Crear utilizando Navigator del Iseries

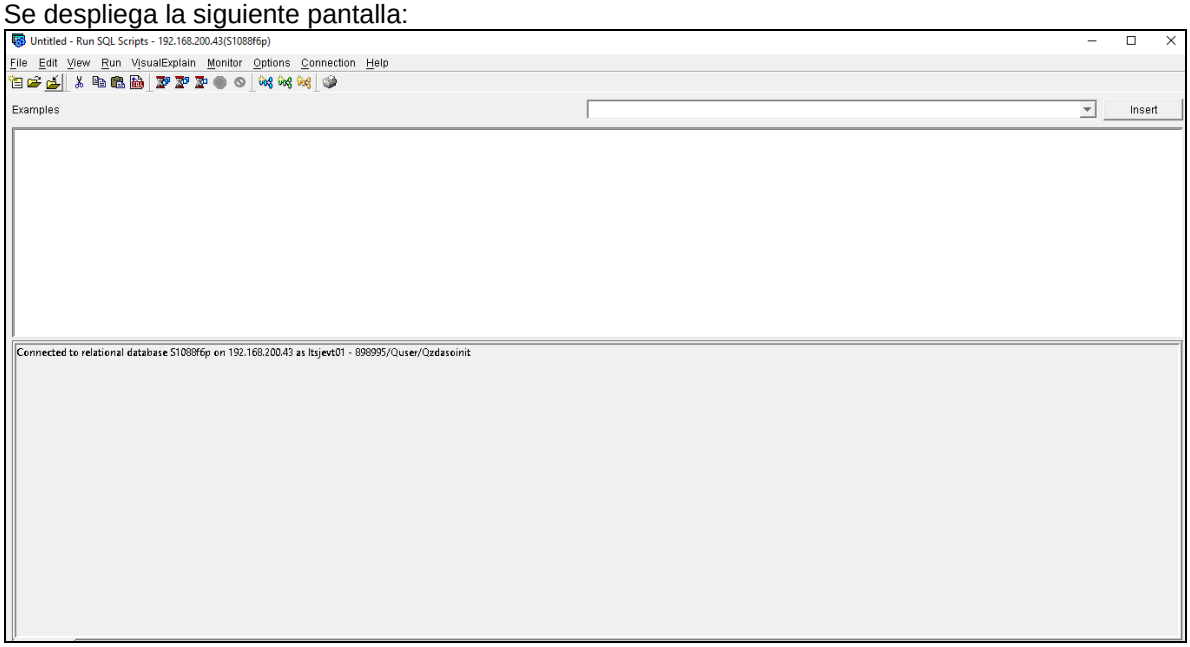
En un archivo en formato y extensión texto se diligencia la sentencia SQL, por ejemplo:

```
CREATE PROCEDURE ITSFRYITS.REFERENCIAS(
   IN NMBPRG CHAR(10) )
DYNAMIC RESULT SETS 300
LANGUAGE SQL-
SPECIFIC ITSFRYITS.RFCPGM
NOT DETERMINISTIC
MODIFIES SQL DATA
CALLED ON NULL INPUT
SET OPTION  ALWBLK = *ALLREAD,
ALWCPYDTA = *OPTIMIZE,
COMMIT = *NONE,
DECRESULT = (31, 31, 00),
DFTRDBCOL = *NONE,
DYNDFTCOL = *NO,
DYNUSRPRF = *USER,
SRTSEQ = *HEX
EJECUTA: BEGIN
-- Cursor para mostrar la referencia del programa
DECLARE RFRPGM CURSOR WITH RETURN FOR
SELECT WHLIB, WHFNAM, WHTEXT
FROM ITSUTLLIB.REFERDES
WHERE WHPNAM = NMBPRG;

OPEN RFRPGM;
END EJECUTA;

COMMENT ON SPECIFIC PROCEDURE ITSFRYITS.RFCPGM
IS 'Prueba de SP';
```

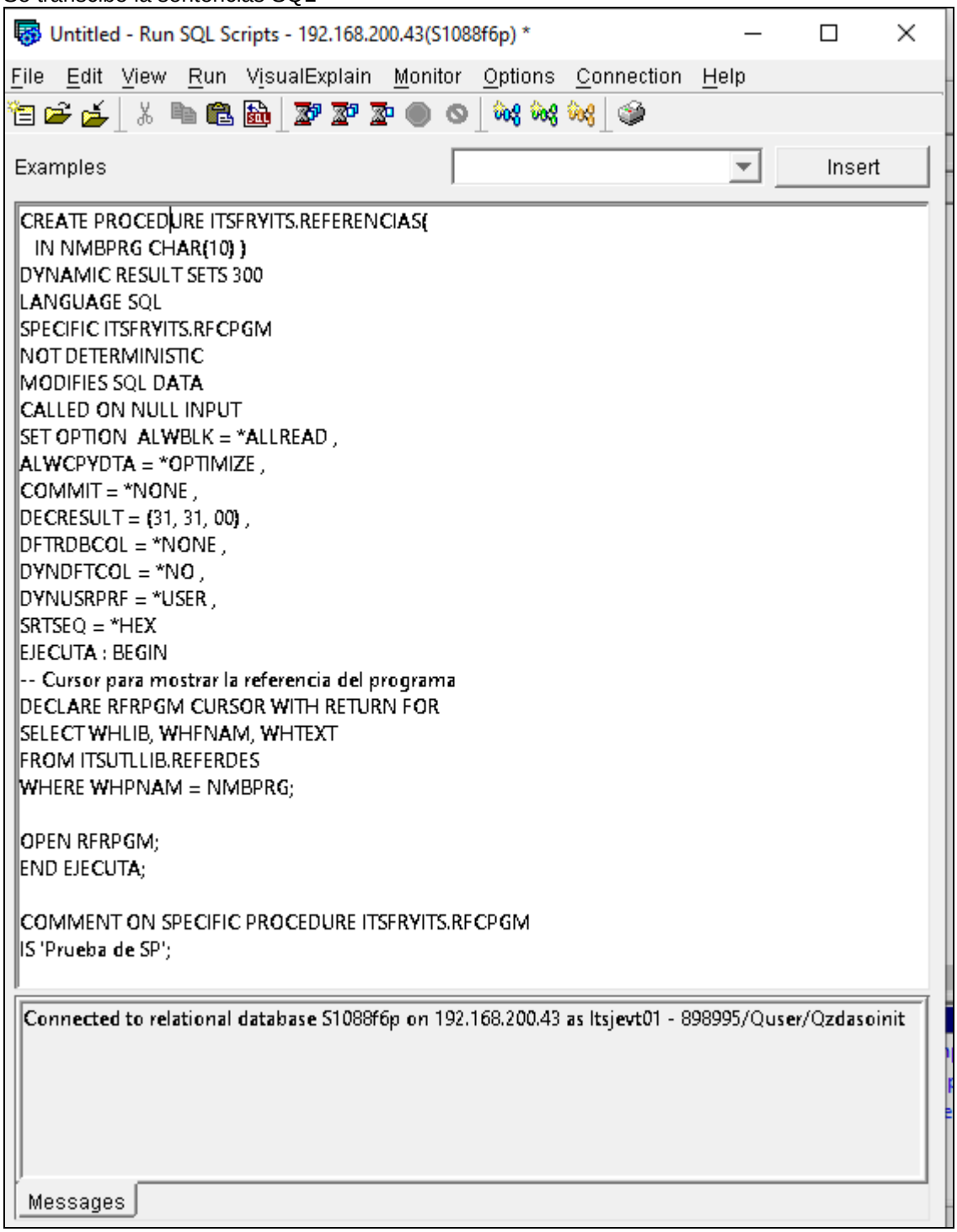
Se ingresa en el software RUN an SQL Script



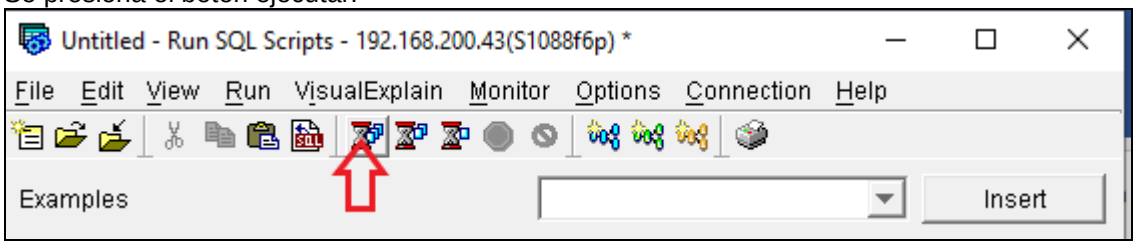
Se despliega la siguiente pantalla:



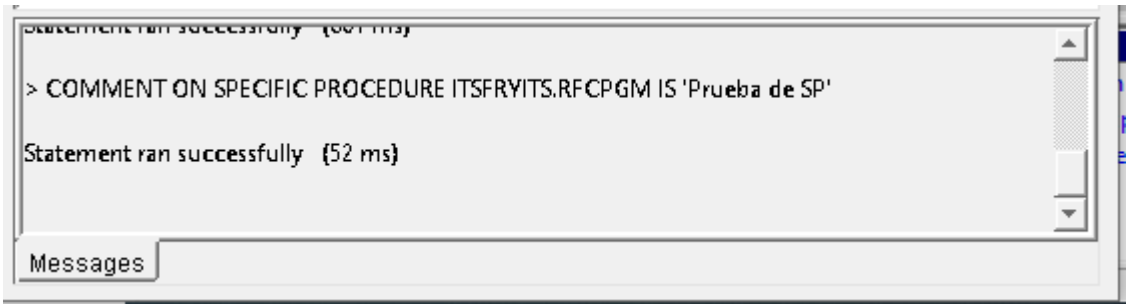
Se transcibe la sentencias SQL



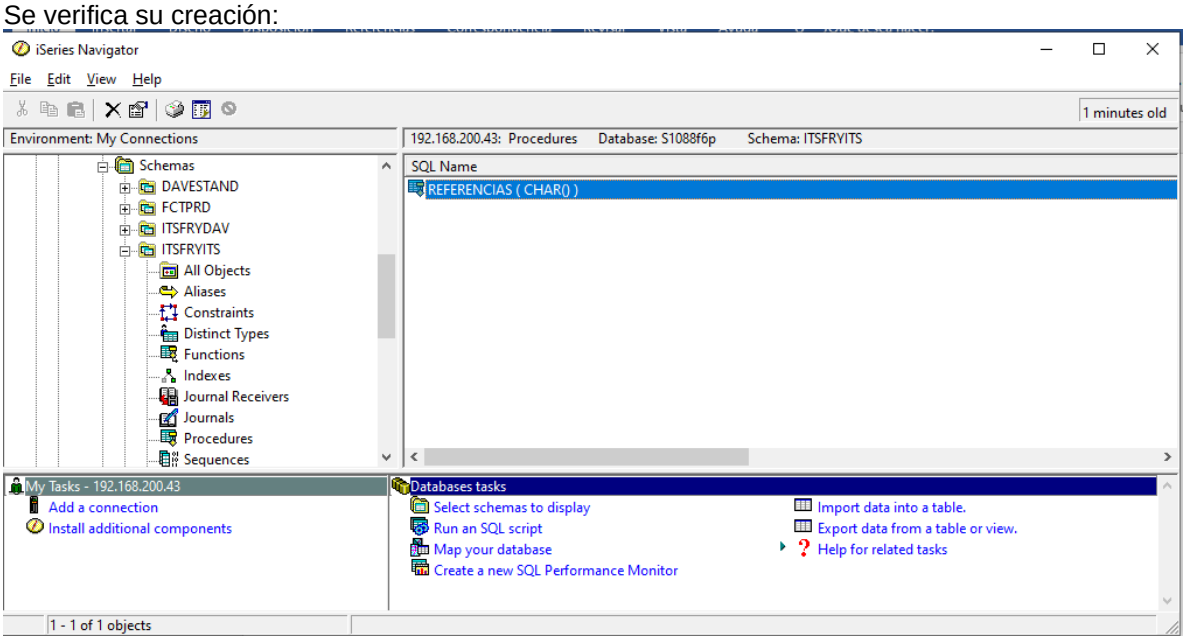
Se presiona el boton ejecutar:



Se despliega el siguiente mensaje:

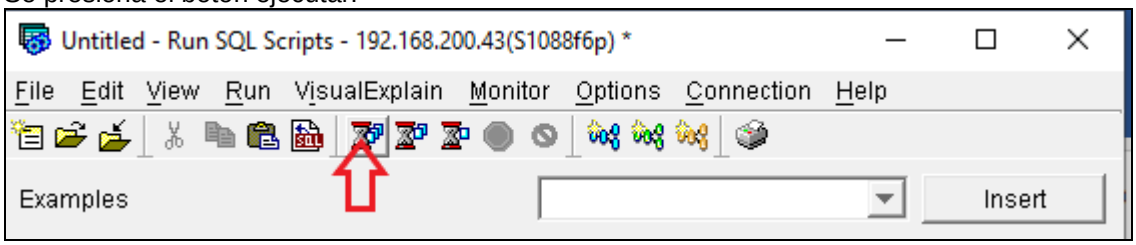


Se verifica su creación:

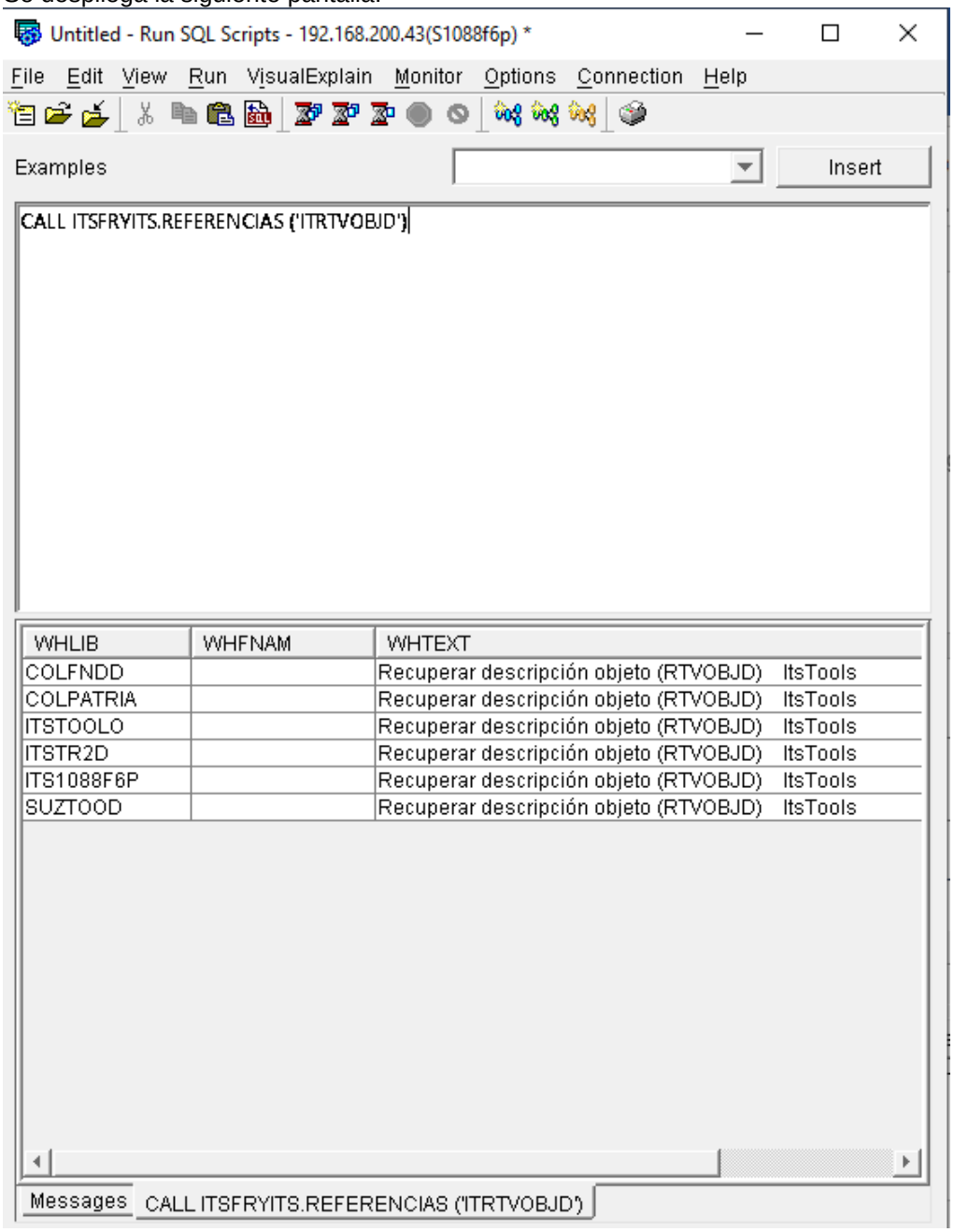


Se ejecuta el procedimiento:

Se presiona el boton ejecutar:



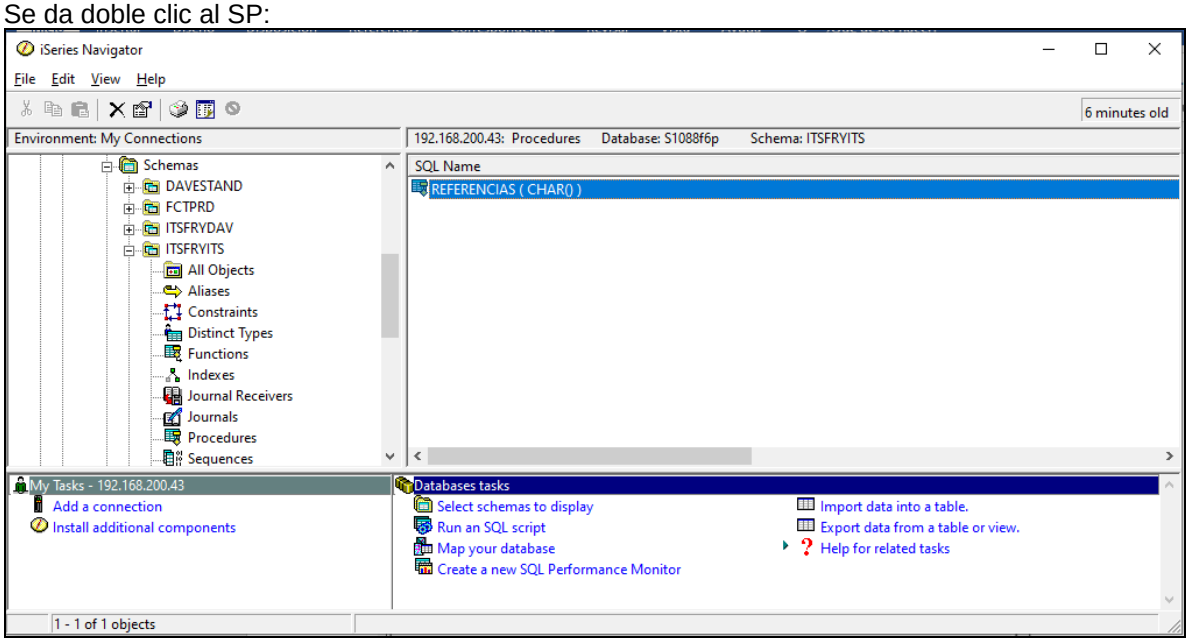
Se despliega la siguiente pantalla:



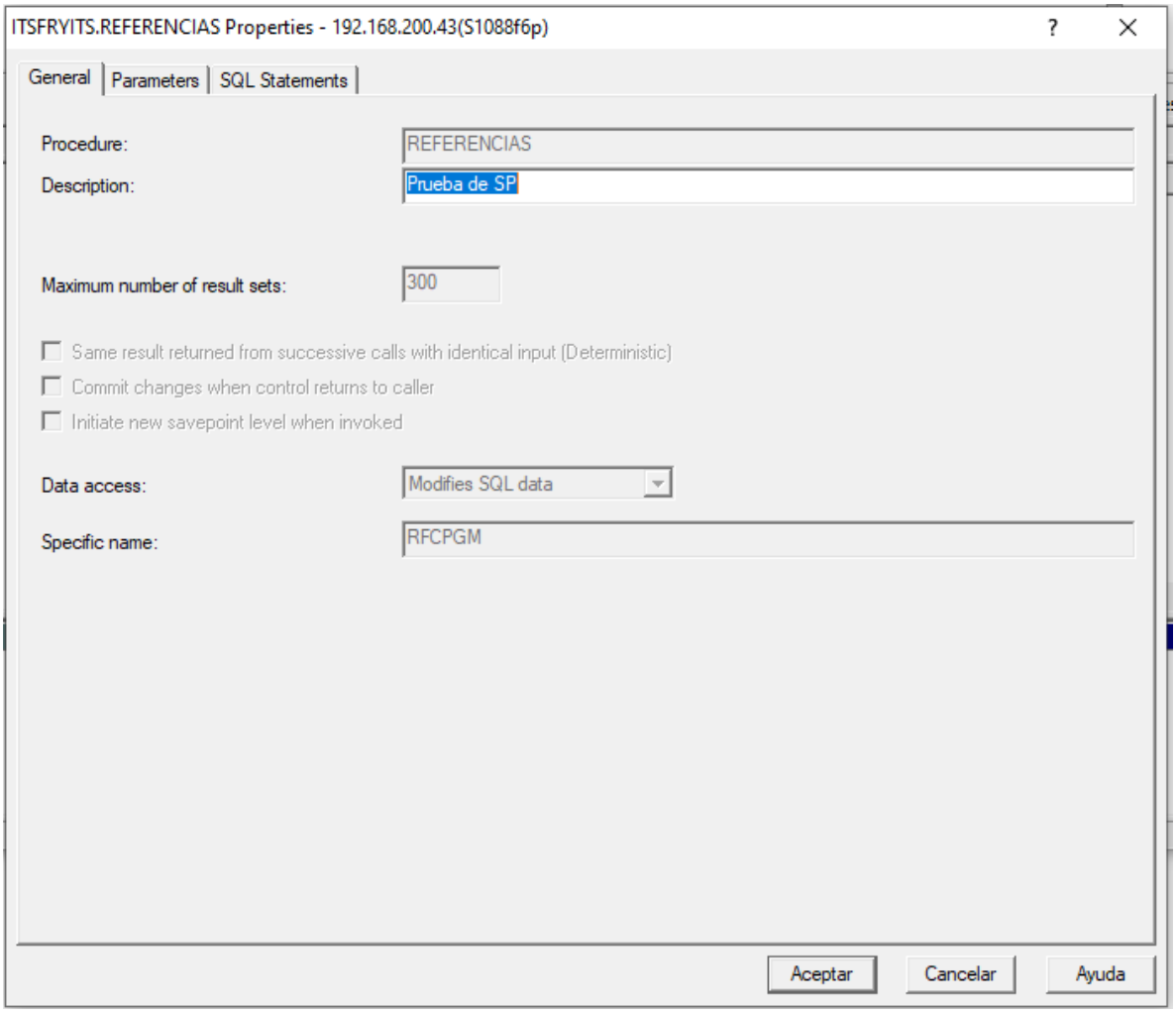
4.11 Propiedades del SP REFERENCIAS

Propiedades del SP REFERENCIAS

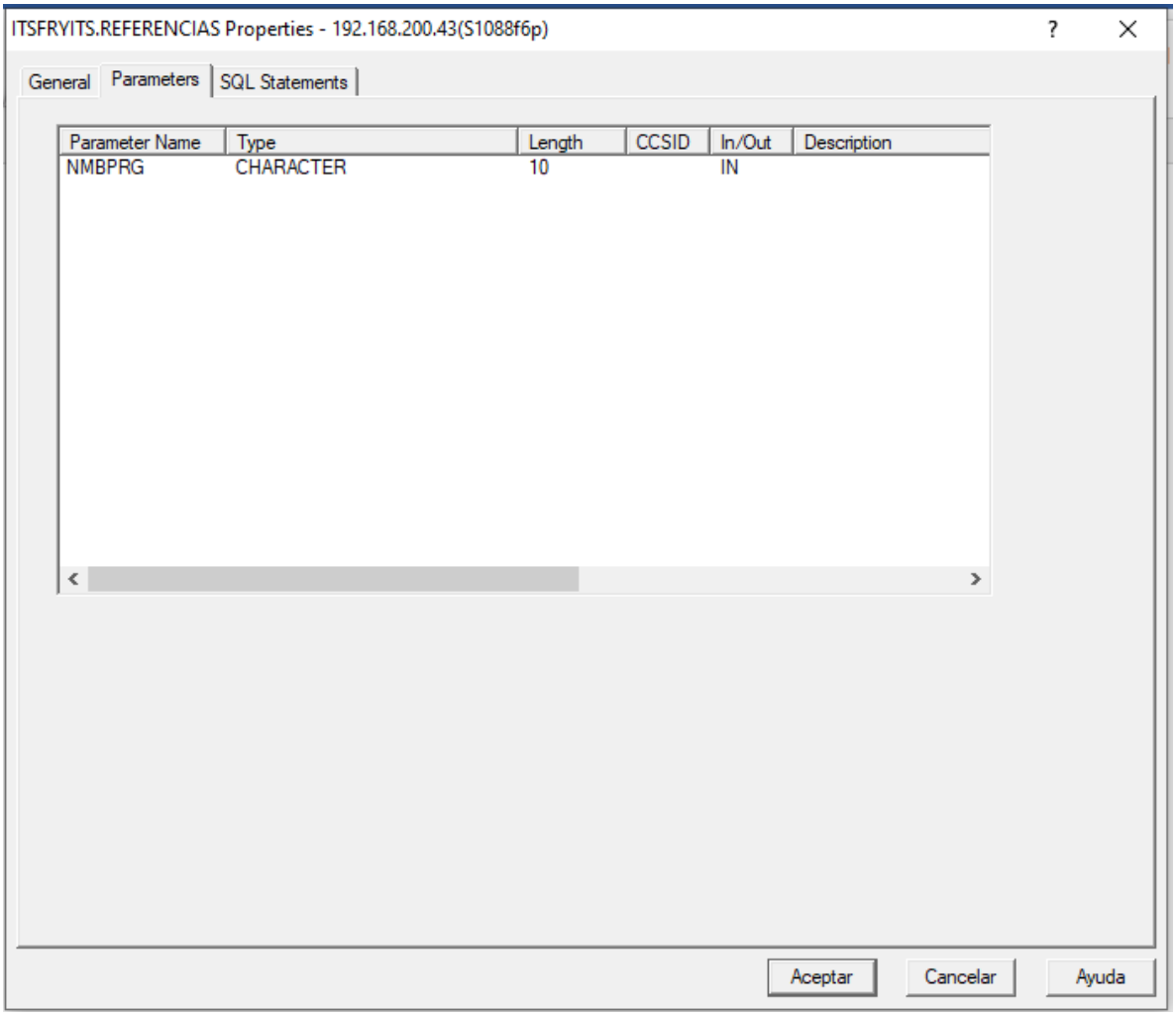
Se da doble clic al SP:



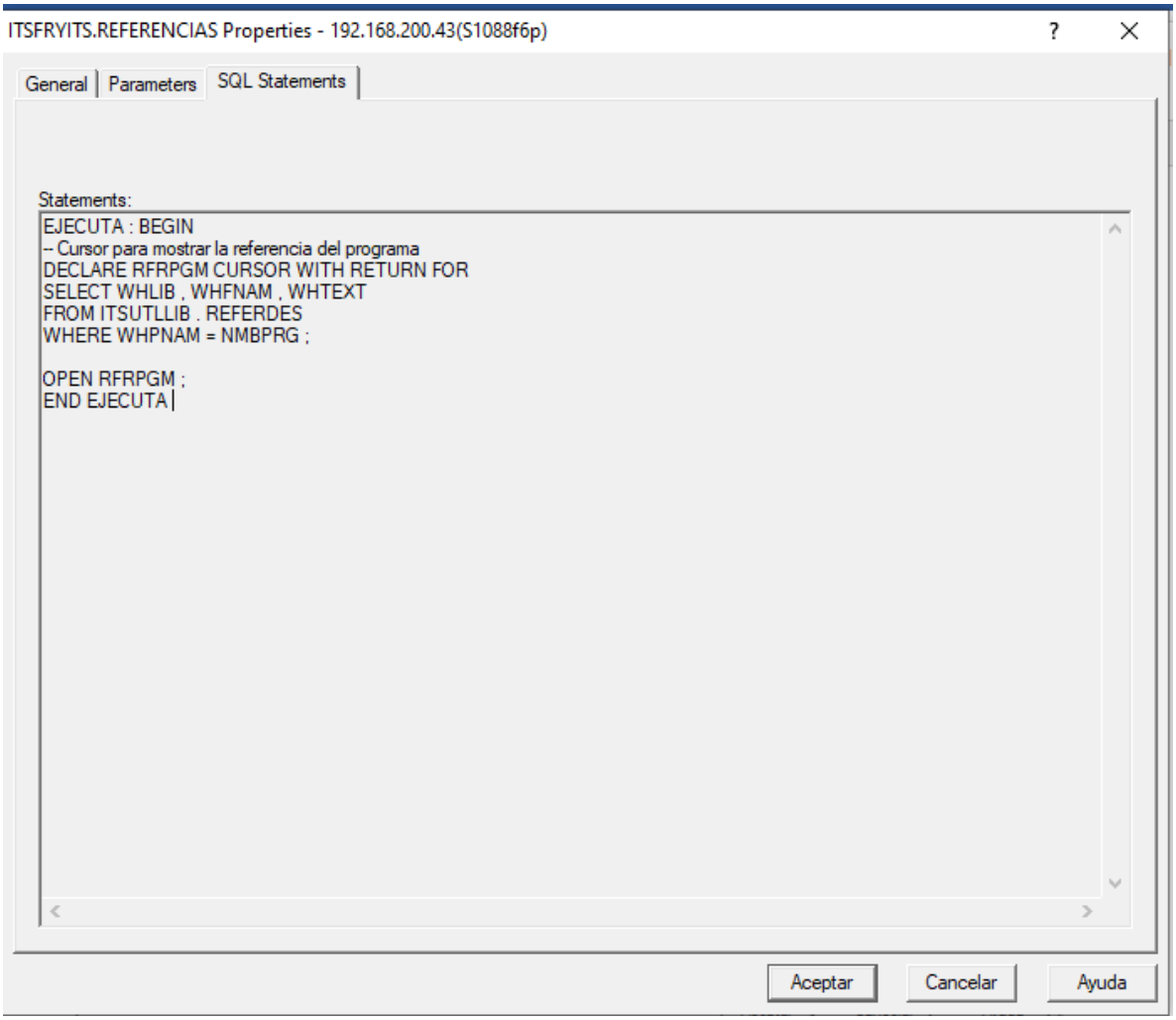
Pestaña General



Pestaña Parámetros



Pestaña Sentencias



4.12 Plantilla para las practicas

Para realizar practicas se recomienda utilizar el Navigator y utilizar las siguientes sentencias:

```
CREATE PROCEDURE biblioteca.nombre SP(
   IN nombre parámetros CHAR(##) )
DYNAMIC RESULT SETS 300
LANGUAGE SQL-
SPECIFIC biblioteca.nombre objeto
NOT DETERMINISTIC
MODIFIES SQL DATA
CALLED ON NULL INPUT
SET OPTION  ALWBLK = *ALLREAD,
ALWCPYDTA = *OPTIMIZE,
COMMIT = *NONE,
DECRESULT = (31, 31, 00),
DFTRDBCOL = *NONE,
DYNDFTCOL = *NO,
DYNUSRPRF = *USER,
SRTSEQ = *HEX
EJECUTA: BEGIN
-- Comentarios
--de aquí en adelante Se digita las Sentencias SQL
END EJECUTA;

COMMENT ON SPECIFIC PROCEDURE biblioteca.nombre objeto
IS 'descripción';
```

En internet hay disponible bastantes ejemplos de SP.

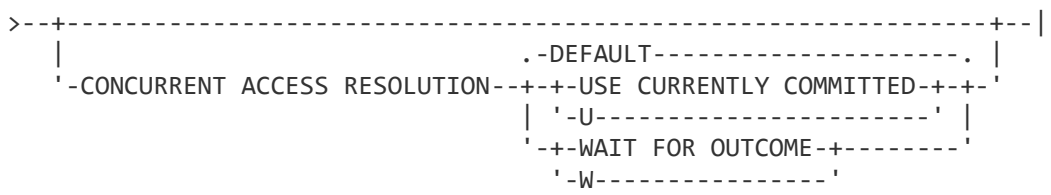
4.13 Sintaxis

De acuerdo al sitio Web de IBM, la sintaxis para SP son las siguientes:

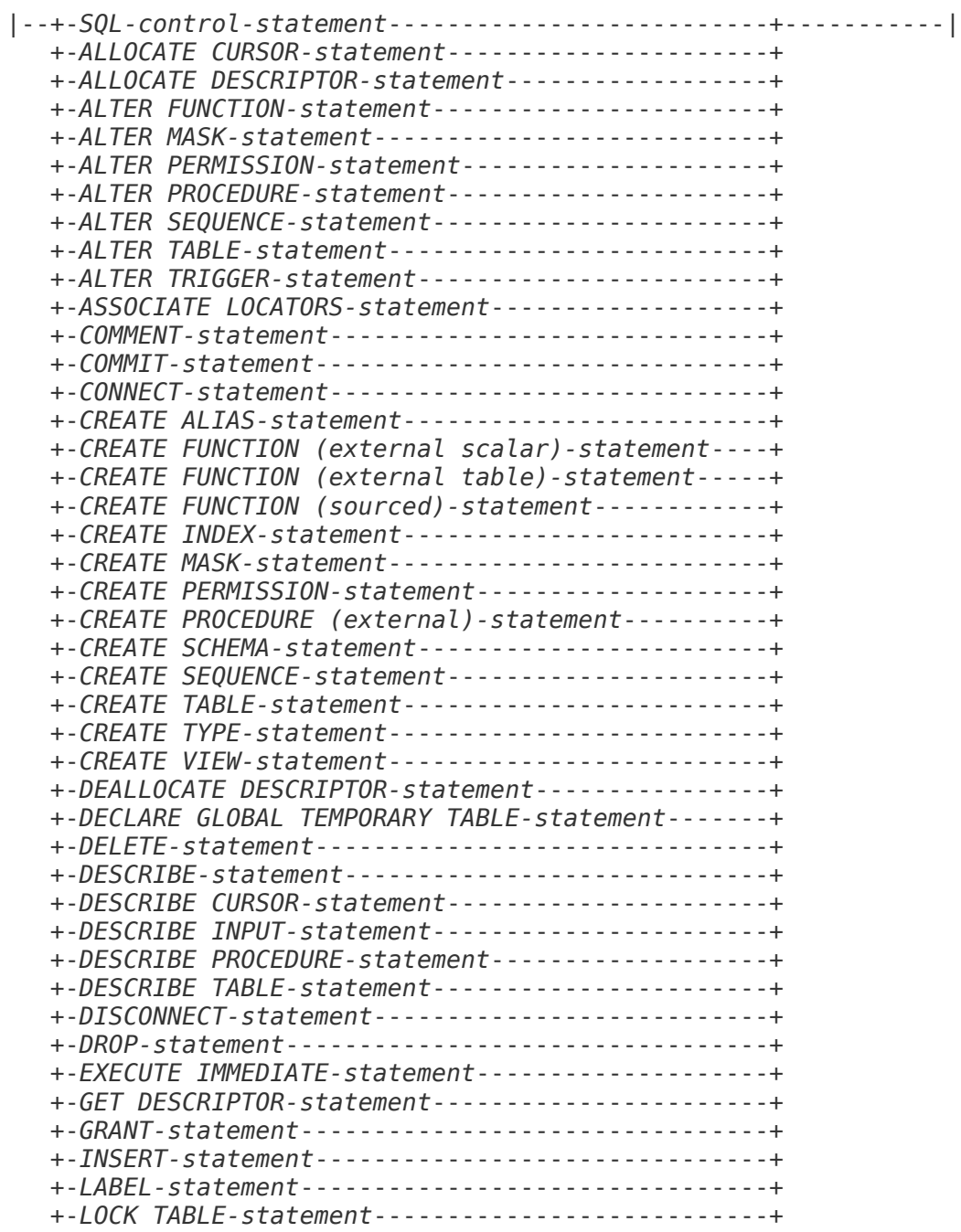
4.13.1 Interno

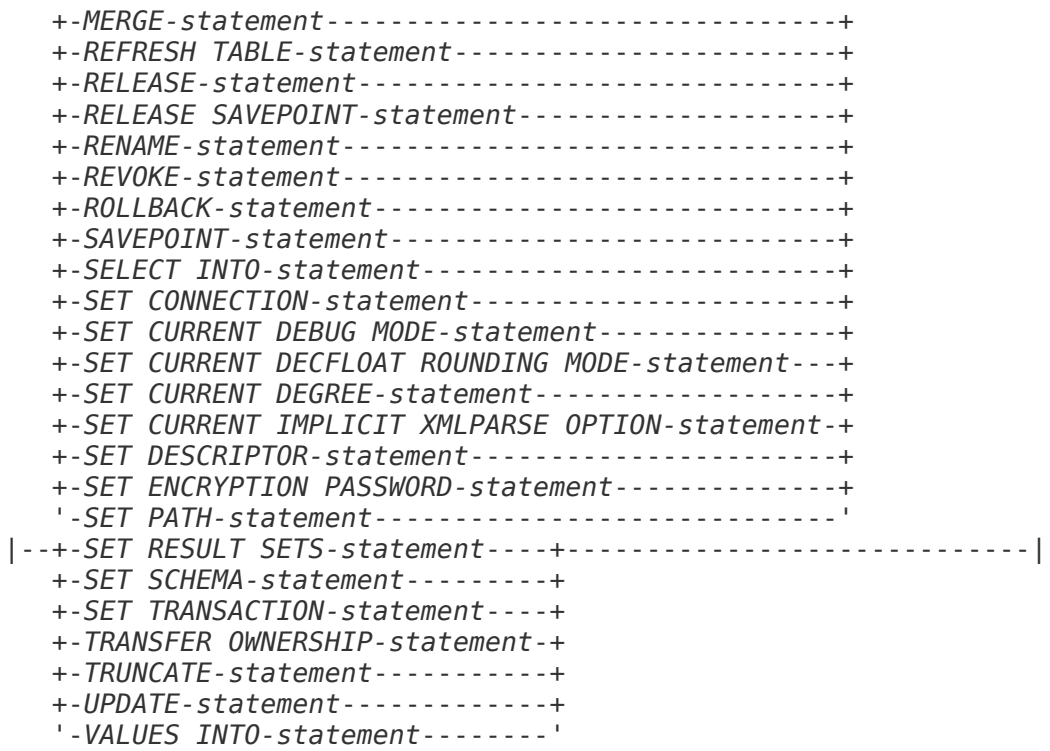
```
>>-CREATE--+------------+--PROCEDURE--procedure-name------------>
           '-OR REPLACE-'

>--+-----------------------------------------+------------------>
   '-(--+-------------------------------+--)-'
        | .-,-------------------------. |
        | V                           | |
        '-----parameter-declaration---+-'

>--+-procedure-definition--------------+---------------------><
   '-WRAPPED--obfuscated-statement-text-'

procedure-definition

|--option-list--+---------------------+--SQL-routine-body------|
                '-SET OPTION-statement-'
```

```
parameter-declaration

   .-IN----.
|--+-------+--parameter-name--data-type--+----------------+-----|
   +-OUT---+                              '-default-clause-'
   '-INOUT-'

data-type

|--+-built-in-type------+-------------------------------------|
   +-distinct-type-name-+
   '-array-type-name----'

default-clause

|--DEFAULT--+-NULL-------------+-------------------------------|
            +-constant---------+
            +-special-register-+
            +-global-variable--+
            '-(--expression--)-'


option-list

   .-LANGUAGE SQL-. (1)  .-NOT DETERMINISTIC-.
|--+--------------+------+-------------------+----------------->
                         '-DETERMINISTIC-----'

   .-MODIFIES SQL DATA-.  .-CALLED ON NULL INPUT .
>--+-------------------+--+----------------------+-------------->
   +-READS SQL DATA----+
   '-CONTAINS SQL------'

   .-INHERIT SPECIAL REGISTERS-.
>--+---------------------------+------------------------------->

   .-DYNAMIC RESULT SETS--0-------.
>--+------------------------------+----------------------------->
   '-DYNAMIC RESULT SETS--integer-'

>--+-------------------------+--+--------------------+--------->
   '-SPECIFIC--specific-name-'  +-DISALLOW DEBUG MODE-+
                                +-ALLOW DEBUG MODE----+
                                '-DISABLE DEBUG MODE--'

   .-FENCED-----.  .-PROGRAM TYPE MAIN-.
>--+------------+--+-------------------+----------------------->
   '-NOT FENCED-'  '-PROGRAM TYPE SUB--'

   .-OLD SAVEPOINT LEVEL-.    .-COMMIT ON RETURN NO--.
>--+---------------------+--+-+----------------------+-+-------->
   '-NEW SAVEPOINT LEVEL-'  | '-COMMIT ON RETURN YES-' |
                            '-AUTONOMOUS--------------'
```

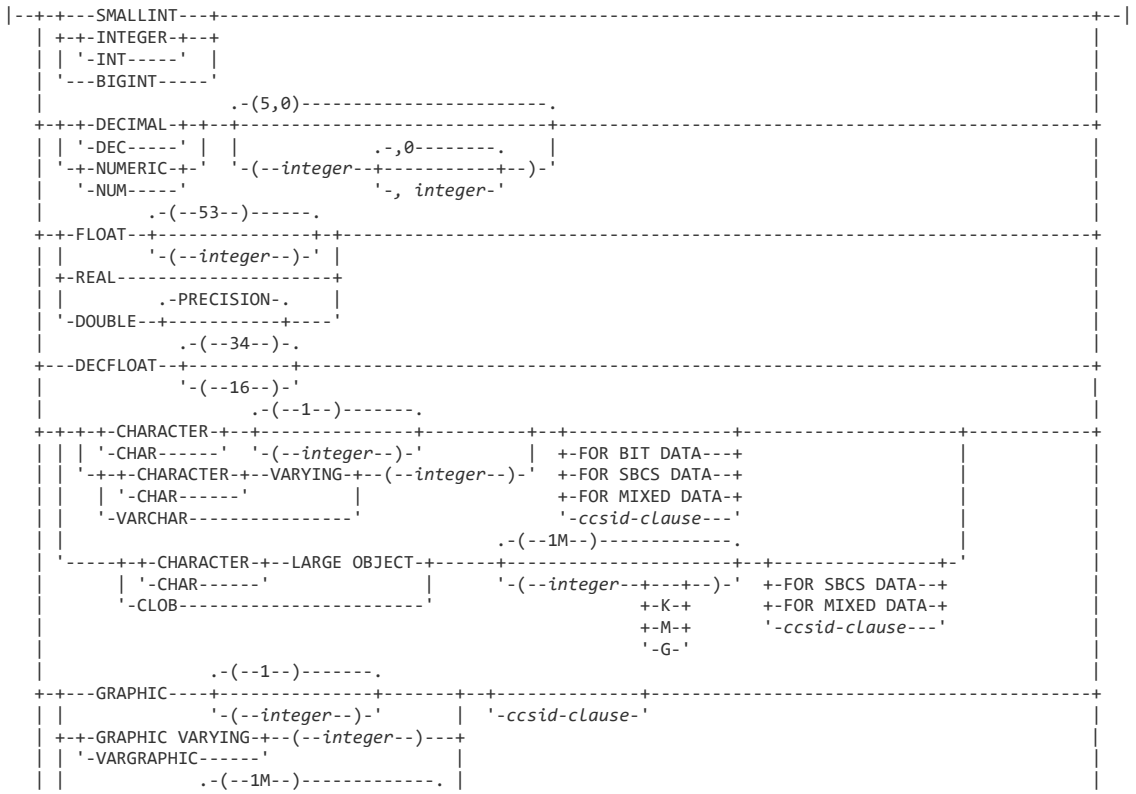


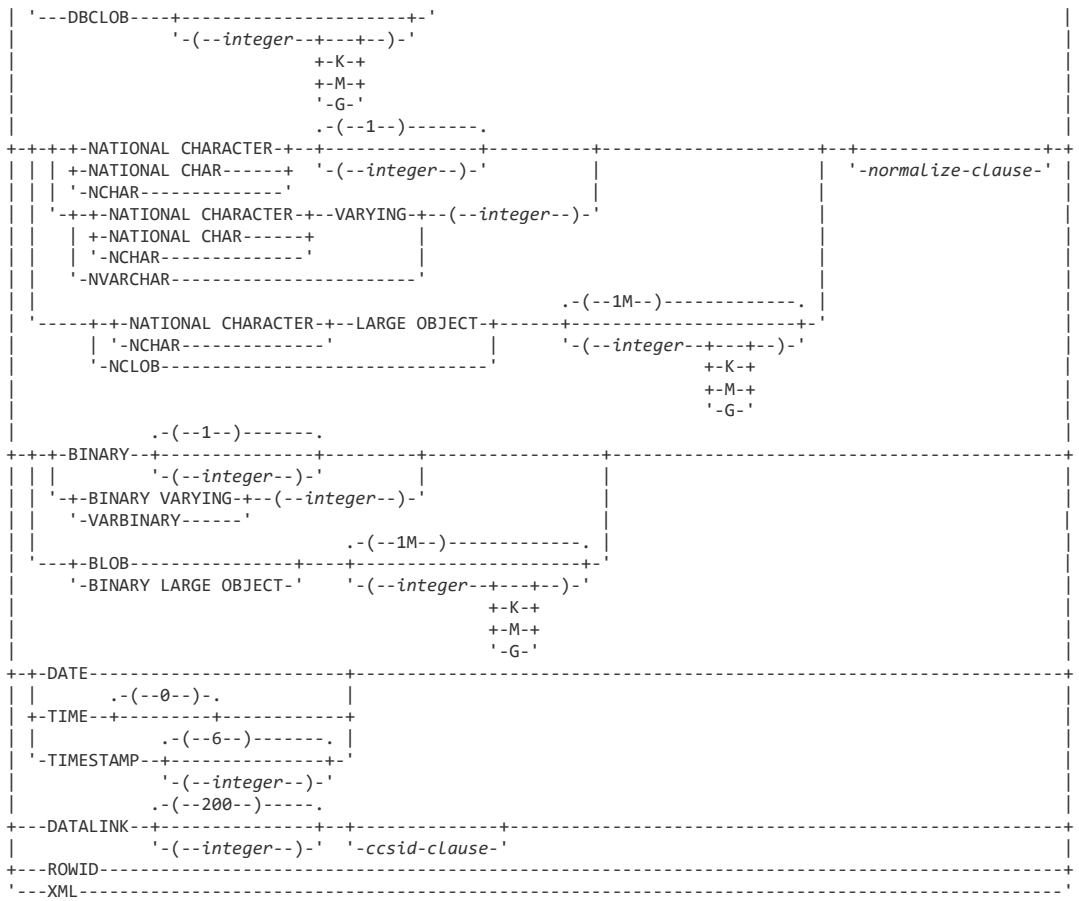
SQL-routine-body



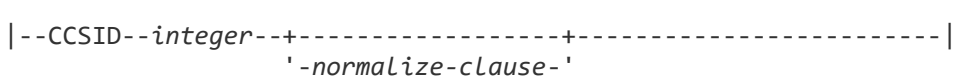


built-in-type

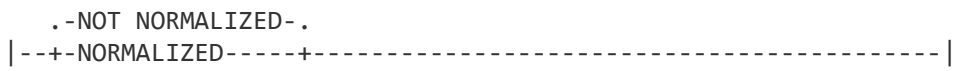




ccsid-clause



normalize-clause



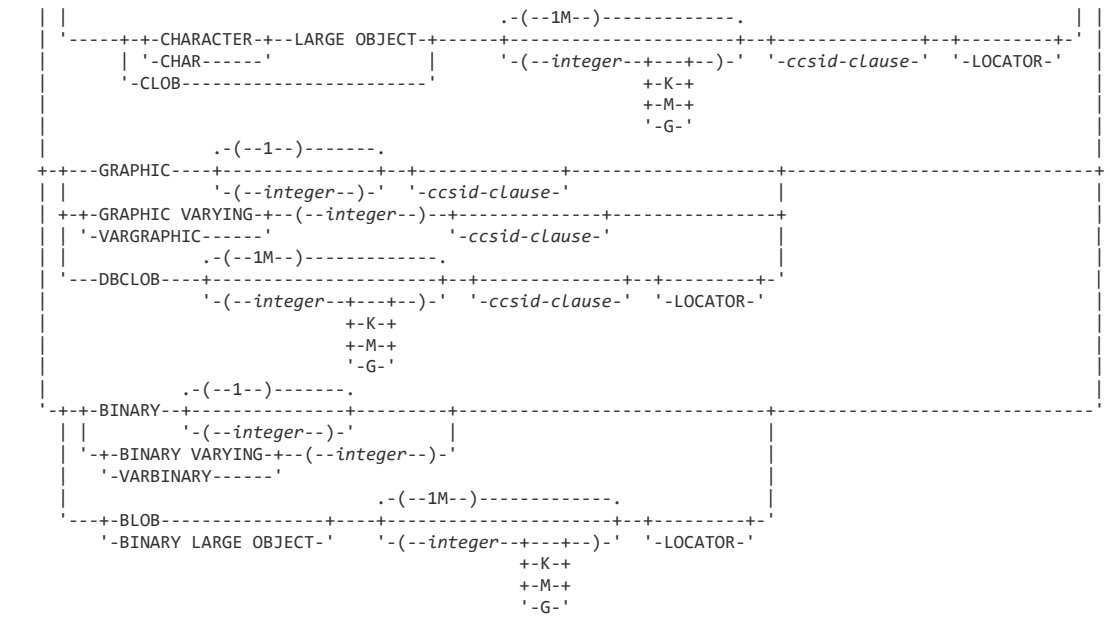
Ejemplos:

```
CREATE PROCEDURE MEDIAN_RESULT_SET (OUT medianSalary DECIMAL(7,2))
     LANGUAGE SQL
     DYNAMIC RESULT SETS 1
BEGIN
     DECLARE v_numRecords INTEGER DEFAULT 1;
     DECLARE v_counter INTEGER DEFAULT 0;
     DECLARE c1 CURSOR FOR
        SELECT salary
           FROM staff
           ORDER BY salary;
     DECLARE c2 CURSOR WITH RETURN FOR
```

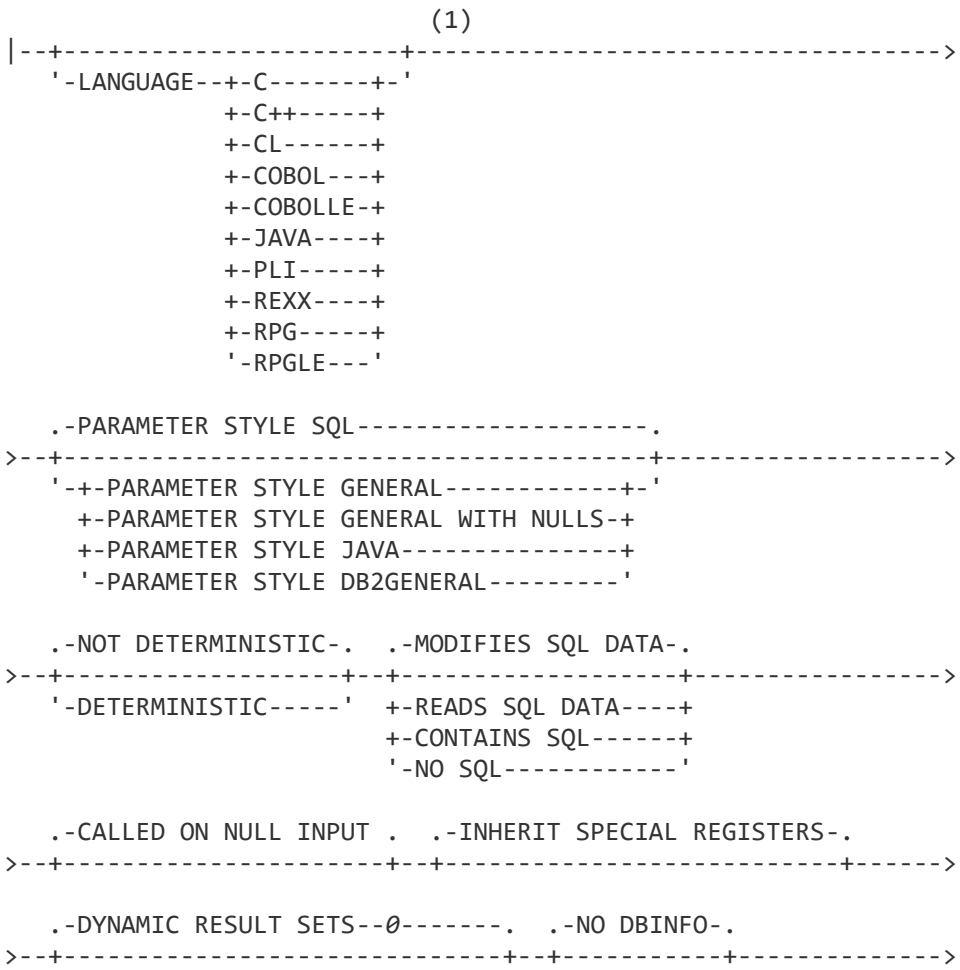
```
            SELECT name, job, salary
               FROM staff
               WHERE salary > medianSalary
               ORDER BY salary;
         DECLARE EXIT HANDLER FOR NOT FOUND
            SET medianSalary = 6666;
         SET medianSalary = 0;
         SELECT COUNT(*) INTO v_numRecords FROM STAFF;
         OPEN c1;
         WHILE v_counter < (v_numRecords / 2 + 1)
            DO FETCH c1 INTO medianSalary;
            SET v_counter = v_counter + 1;
         END WHILE;
         CLOSE c1;
         OPEN c2;
   END
```

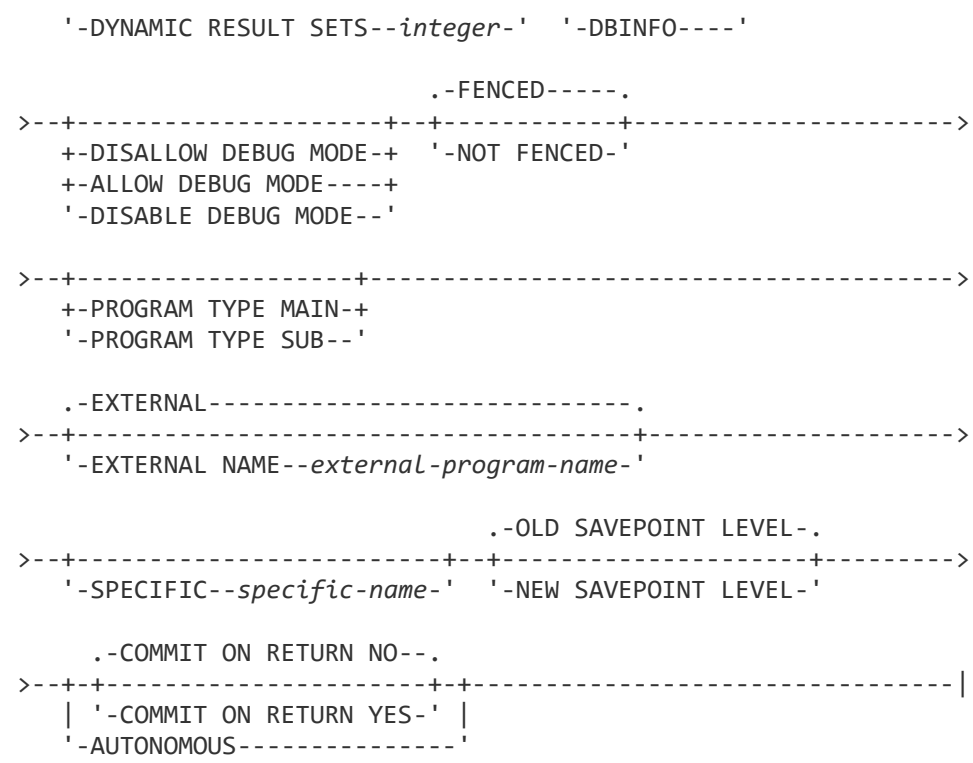
4.13.2 Externo

```
>>-CREATE--+------------+--PROCEDURE--procedure-name------------>
           '-OR REPLACE-'

>--+-----------------------------------------+--option-list----->
   '-(--+-------------------------------+--)-'
        | .-,-------------------------. |
        | V                           | |
        '-----parameter-declaration---+-'

>--+----------------------+-----------------------------------><
   '-SET OPTION-statement-'


parameter-declaration

   .-IN----.
|--+-------+--+----------------+--data-type--+-----------------------+--+----------------+--|
   +-OUT---+  '-parameter-name-'             '-AS -+-LOCATOR-------+-'  '-default-clause-'
   '-INOUT-'                                      '-XML-cast-type-'

data-type

|--+-built-in-type------+--------------------------------------|
   +-distinct-type-name-+
   '-array-type-name----'

default-clause

|--DEFAULT--+-NULL-------------+-------------------------------|
            +-constant---------+
            +-special-register-+
            +-global-variable--+
            '-(--expression--)-'

XML-cast-type

                     .-(--1--)-------.

|--+-+-+-CHARACTER-+--+---------------+--+--------------+-------------------------------------------+-+--|
   | | '-CHAR------'  '-(--integer--)-'  '-ccsid-clause-'                                           | |
   | +-+-+-CHARACTER-+--VARYING-+--(--integer--)--+--------------+-------------------------------------+ |
   | | | '-CHAR------'          |                '-ccsid-clause-'                                     | |
   | | '-VARCHAR----------------'                                                                    | |
```

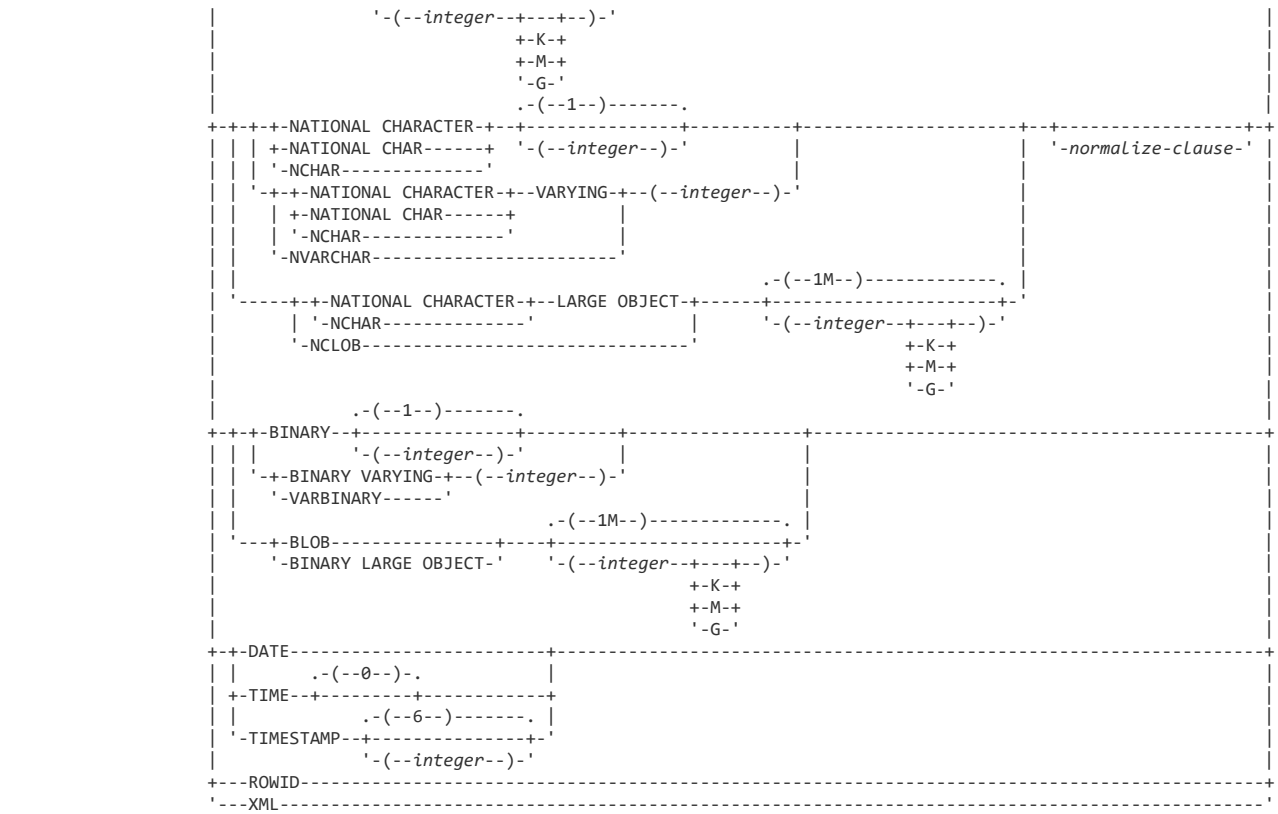

.-(--1M--)-------------.
'-----+-+-CHARACTER-+--LARGE OBJECT-+------+---------------------+--+--------------+--+---------+-'
'-CHAR------'
'-CLOB------------------------'
'-(--integer--+---+--)-'
+-K-+
+-M-+
'-G-'
'-ccsid-clause-'
'-LOCATOR-'
.-(--1--)-------.
+-+---GRAPHIC----+---------------+--+--------------+-------------------+-----------------------------+
'-(--integer--)-'
'-ccsid-clause-'
+-+-GRAPHIC VARYING-+--(--integer--)--+--------------+----------------+
'-VARGRAPHIC------'
'-ccsid-clause-'
.-(--1M--)-------------.
'---DBCLOB----+----------------------+--+--------------+--+---------+-'
'-(--integer--+---+--)-'
'-ccsid-clause-'
'-LOCATOR-'
+-K-+
+-M-+
'-G-'
.-(--1--)-------.
'-+-+-BINARY--+---------------+---------+-----------------------------+-----------------------------'
'-(--integer--)-'
'-+-BINARY VARYING-+--(--integer--)-'
'-VARBINARY------'
.-(--1M--)-------------.
'---+-BLOB----------------+----+----------------------+--+---------+-'
'-BINARY LARGE OBJECT-'
'-(--integer--+---+--)-'
'-LOCATOR-'
+-K-+
+-M-+
'-G-'

option-list

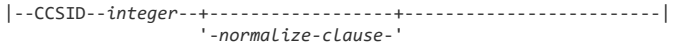

(1)
|--+------------------------+------------------------------------->
'-LANGUAGE--+-C--------+-'
+-C++------+
+-CL-------+
+-COBOL----+
+-COBOLLE-+
+-JAVA-----+
+-PLI------+
+-REXX-----+
+-RPG------+
'-RPGLE----'
.-PARAMETER STYLE SQL--------------------.
>--+--+-------------------->
'-+-PARAMETER STYLE GENERAL-------------+-'
+-PARAMETER STYLE GENERAL WITH NULLS-+
+-PARAMETER STYLE JAVA---------------+
'-PARAMETER STYLE DB2GENERAL---------'
.-NOT DETERMINISTIC-. .-MODIFIES SQL DATA-.
>--+-------------------+--+--------------------+----------------->
'-DETERMINISTIC-----' +-READS SQL DATA----+
+-CONTAINS SQL------+
'-NO SQL------------'
.-CALLED ON NULL INPUT . .-INHERIT SPECIAL REGISTERS-.
>--+----------------------+--+---------------------------+------>
.-DYNAMIC RESULT SETS--0-------. .-NO DBINFO-.
>--+-----------------------------+--+-----------+-------------->
'-1-------'
'-2-------'
'-3-------'



built-in-type



ccsid-clause



normalize-clause

Ejemplos:

```
CREATE PROCEDURE PARTS_ON_HAND (IN  PARTNUM  INTEGER,
                                    OUT COST     DECIMAL(7,2),
                                    OUT QUANTITY INTEGER)
          LANGUAGE JAVA
          PARAMETER STYLE JAVA
          EXTERNAL NAME 'parts.onhand'


CREATE PROCEDURE ASSEMBLY_PARTS (IN  ASSEMBLY_NUM  INTEGER,
                                    OUT NUM_PARTS     INTEGER,
                                    OUT COST DOUBLE)
          LANGUAGE C
          PARAMETER STYLE GENERAL
          DYNAMIC RESULT SETS 1
          FENCED
          EXTERNAL NAME ASSEMBLY
```