



Universidad Nacional del Litoral

Facultad de Ingeniería y Ciencias Hídricas

Departamento de Informática

Bases de Datos

SQL: Guía de Trabajo Nro. 4

Stored Procedures y functions: Retorno

¿Que retorna un procedure o función?

El procedure o función va a llevar a cabo una tarea y retornar algo. A continuación analizaremos las opciones de retorno que tenemos.

1. Salida de un stored procedure en T-SQL

En T-SQL, tenemos tres posibilidades para obtener una salida de un stored procedure:

1.1. Return value

También llamado "status de retorno". El return value es un valor entero, y está pensado para proporcionar un "código de status" al programa invocante.

En un entorno cliente/servidor el return value sirve para indicar al programa invocante el motivo por el cual el procedimiento finalizó su ejecución.

Bajo condiciones normales, un SP finaliza su ejecución cuando alcanza el final del código del mismo o cuando ejecuta una sentencia `RETURN`.

Esta finalización normal (exitosa) retorna un status de 0.

Microsoft reserva un bloque de números de -1 a -99 para identificar status de error. Solo los primeros 14 valores poseen significado:

Valor	Significado
-1	Falta objeto
-2	Error de tipo de dato
-3	El proceso fue elegido como víctima de deadlock.
-4	Error de permisos
-5	Error de sintaxis
-6	Error de usuario de miscelánea
-7	Error de recursos (sin espacio, por ejemplo)
-8	Problema interno no fatal.
-9	El sistema ha alcanzado su límite.
-10	Inconsistencia interna fatal.
-11	Inconsistencia interna fatal.
-12	Tabla o índice corrupto.
-13	Base de datos corrupta.
-14	Error de hardware.

El valor del status de retorno de un SP puede capturarse mediante la siguiente sintaxis:

```
EXECUTE <@Variable-local-de-tipo-Int> = <Nombre-SP>  
    Valor-de-Parametro1,  
    Valor-de-Parametro2,  
    Valor-de-Parametro3
```

```
Declare @retorno integer
```

Ejemplo:

```
EXECUTE spBuscarPublicaciones '1389'
```

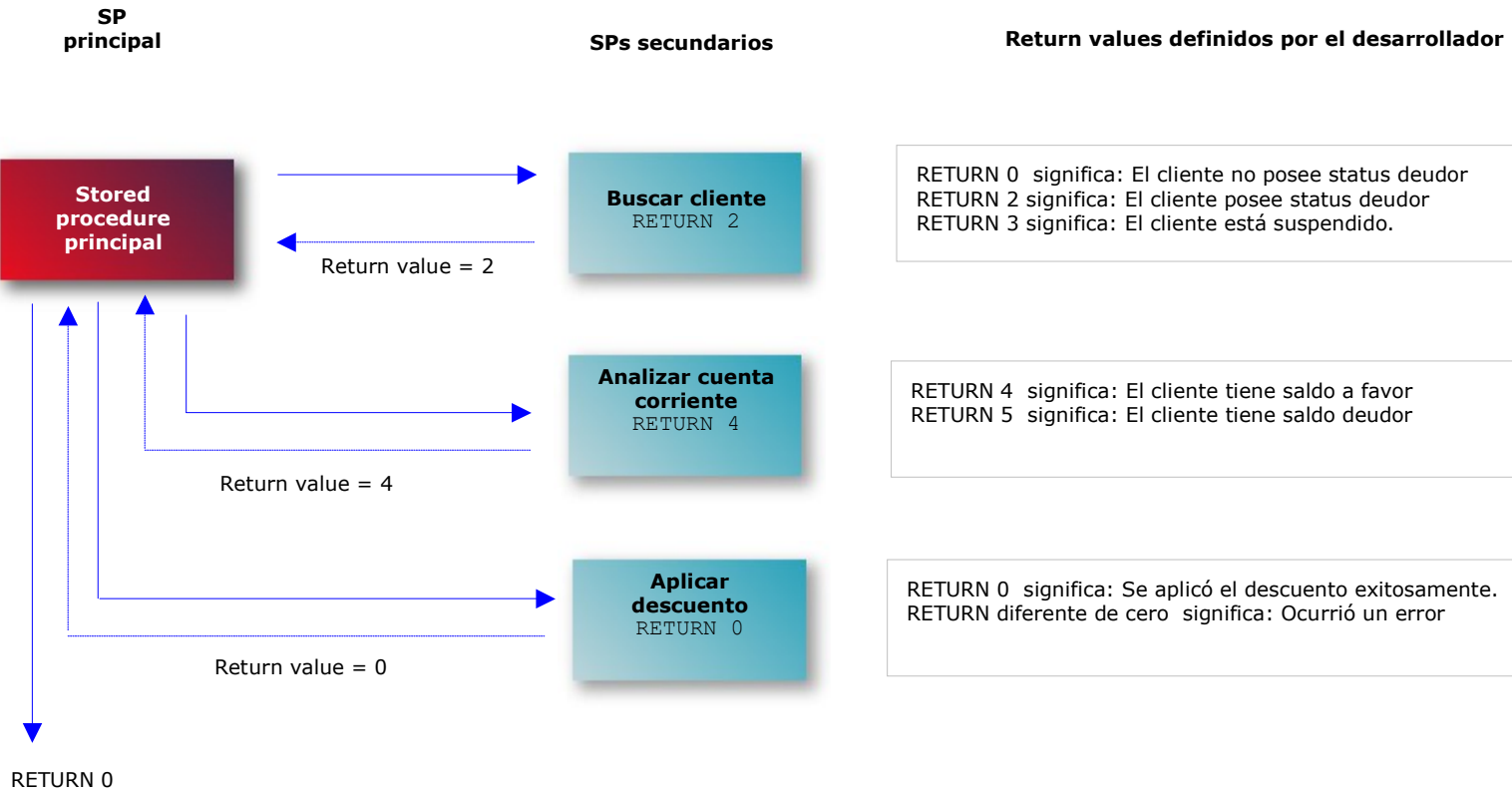
```
Declare @retorno integer
```

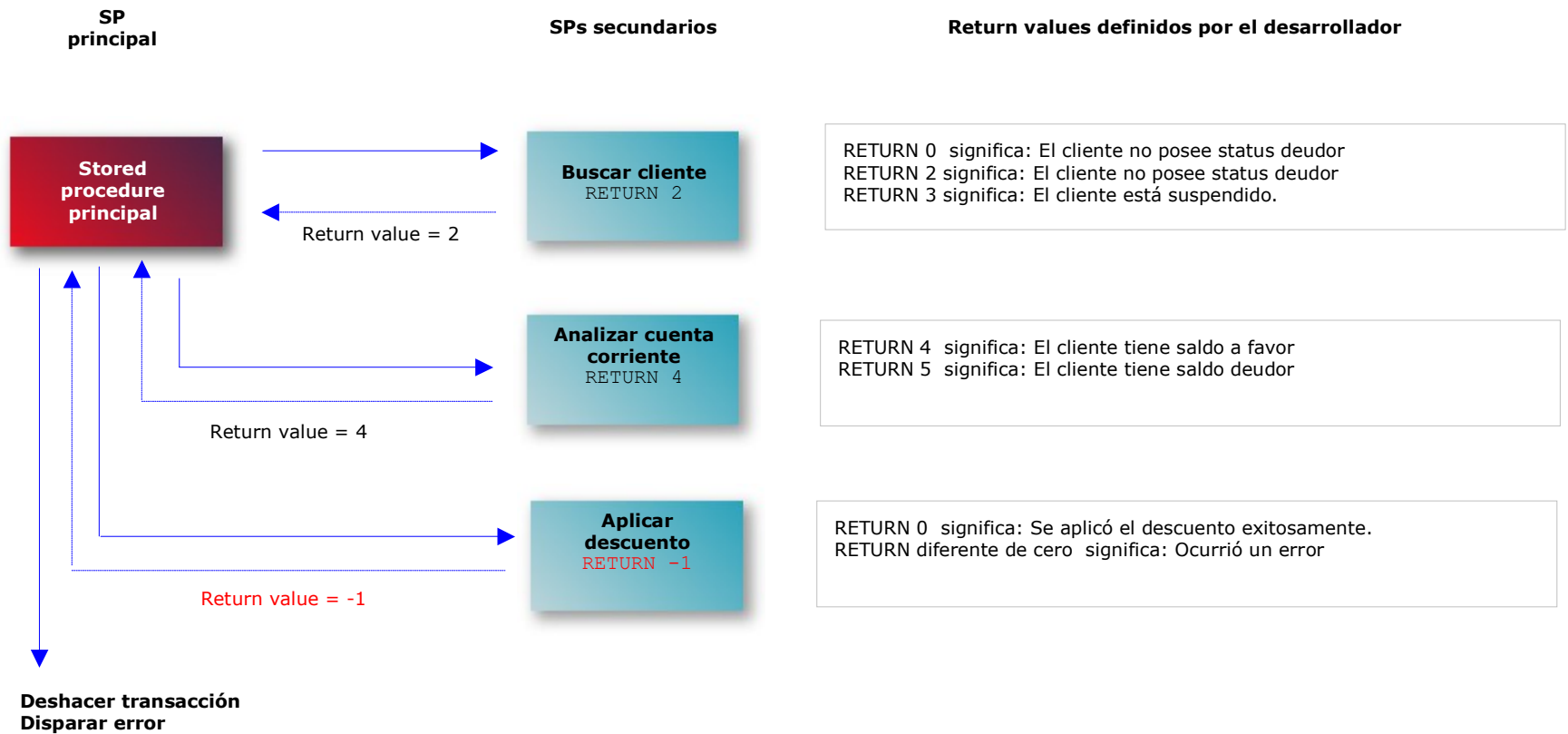
```
EXECUTE @retorno =spBuscarPublicaciones '1389'
```

1.1.1. Return value personalizado

Podemos valernos de esta característica para explotarla a nuestro favor.

Si tenemos un programa principal que invoca a otro secundario, el secundario puede indicarle al principal cómo resultó la ejecución a través de su return value:





Si un SP interno -invocado por otro SP- genera una determinada condición que consideramos un error para nuestra aplicación (un no cumplimiento de una regla de negocio, por ejemplo) el desarrollador puede notificar de esta situación al SP invocante retornando un valor de retorno "personalizado" que el SP invocante pueda interpretar.

En T-SQL, estos códigos de error pueden ser cualquier valor de tipo Int fuera de los reservados (-99 a -1) y se especifican a continuación de la(s) cláusula(s) RETURN en el SP invocado.

1.2. Parámetros OUTPUT

T-SQL permite definir parámetros OUTPUT de cualquiera de los tipos de datos T-SQL.

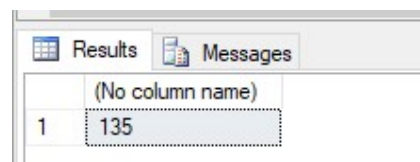
Si necesitamos retornar valores individuales, podemos usar este recurso.

Ejemplo 8

```
CREATE PROCEDURE obtenerCantidadVendida2
(
    @pub_id CHAR(4),
    @cantidad INTEGER OUTPUT
)
AS
    SET @cantidad = (
        SELECT SUM(qty)
        FROM sales S INNER JOIN titles T
            ON S.title_id = T.title_id
        WHERE T.pub_id = @pub_id
    )

    RETURN 0
```

```
DECLARE @cantidad2 INTEGER
EXECUTE obtenerCantidadVendida2 '1389', @cantidad2 OUTPUT
SELECT CONVERT(VARCHAR, @cantidad2)
```



	(No column name)
1	135

1.3. La salida de una sentencia SQL cualquiera sea la forma de la relación

T-SQL permite retornar directamente la salida de una sentencia `SELECT`.

Ejemplo 9

```
CREATE PROCEDURE ListarTitles
AS
    SELECT * FROM titles
```

```
EXECUTE ListarTitles
```

	title_id	title	type	pub_id	price	advance	royalty	ytd_sales	notes
1	BU1032	The Busy Executive's Database Guide	business	1389	19.99	5000.00	10	4095	An overview of available database systems with e...
2	BU1111	Cooking with Computers: Surreptitious Balance Sh...	business	1389	11.99	5000.00	10	3876	Helpful hints on how to use your electronic resourc...
3	BU2075	You Can Combat Computer Stress!	business	0736	2.99	10125.00	24	18722	The latest medical and psychological techniques f...
4	BU7832	Straight Talk About Computers	business	1389	19.99	5000.00	10	4095	Annotated analysis of what computers can do for y...
5	MC2222	Silicon Valley Gastronomic Treats	mod_cook	0877	19.99	0.00	12	2032	Favorite recipes for quick, easy, and elegant meals.
6	MC3021	The Gourmet Microwave	mod_cook	0877	2.99	15000.00	24	22246	Traditional French gourmet recipes adapted for mo...

2. Salida de una function PL/pgSQL

En una función PL/pgSQL podemos obtener la salida a través de OUTPUT PARAMETERS o con un query con una sintaxis especial-
En un caso u otro, el resultado se debe ajustar al tipo de retorno de la function.

2.1. Retornar un valor escalar

2.1.1. Usando un OUTPUT parameter

Ejemplo

```
CREATE FUNCTION Ejemplo31
(
    IN prmPub_id VARCHAR(4),
    OUT prmCantidad INTEGER
)
RETURNS INTEGER

AS
$$
DECLARE
BEGIN
    SELECT SUM(qty)
    INTO prmCantidad
    FROM sales S INNER JOIN titles T
        ON S.title_id = T.title_id
    WHERE T.pub_id = prmPub_id;

END;
$$
LANGUAGE plpgsql
```

```
SELECT Ejemplo31 ('1389');
```

Data Output		Messages	Notifications
ejemplo31 integer			
1	135		

Como vemos:

- El tipo del parámetro **OUT** DEBE COINCIDIR con el tipo de retorno definido para la function.
- No es necesario especificar sentencia RETURN.

2.1.2. Salida directa sin parameter OUT

Podemos retornar directamente un valor escalar:

```
CREATE FUNCTION Ejemplo31B
(
    IN prmPub_id VARCHAR(4)
)
RETURNS INTEGER

AS
$$
DECLARE
    cantidad INTEGER;
BEGIN
    cantidad := (SELECT SUM(qty) FROM sales S INNER JOIN titles T
                  ON S.title_id = T.title_id WHERE T.pub_id = prmPub_id);
    RETURN cantidad;
END;
$$
LANGUAGE plpgsql
```

```
SELECT Ejemplo31B ('1389');
```

Data Output		Messages	Notifications
	ejemplo31b integer		
1	135		

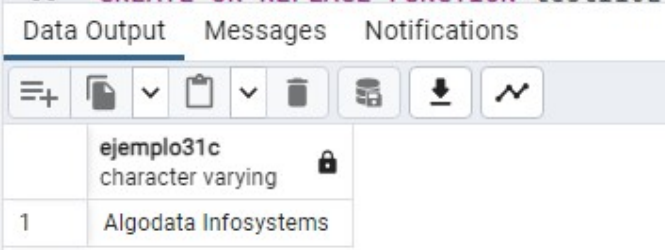
Como vemos:

- Debemos especificar sentencia **RETURN**.
- El tipo del valor retornado a través de la cláusula **RETURN** debe coincidir con el tipo de retorno definido para la function.

También tenemos la posibilidad de definir como tipo de retorno un tipo anclado:

```
CREATE OR REPLACE FUNCTION Ejemplo31C
(
    IN prmPub_id VARCHAR(4)
)
RETURNS Publishers.pub_name%TYPE
AS
$$
DECLARE
    pub_name1 Publishers.pub_name%TYPE;
BEGIN
    pub_name1 := (SELECT pub_name FROM Publishers P
                  WHERE P.pub_id = prmPub_id);
    RETURN pub_name1;
END;
$$
LANGUAGE plpgsql
```

```
SELECT Ejemplo31C('1389')
```



	ejemplo31c character varying
1	Algodata Infosystems

...o bien usar la misma estrategia que para una relación unaria (Ver 2.2), retornar `setof` del tipo de dato correspondiente.

2.2. Functions con más de un OUTPUT parameter

Cuando la function posee más de un OUTPUT parameter, debe retornar un RECORD.

```
CREATE OR REPLACE FUNCTION getMaxMinPrecio
(
    OUT minPrice public.Titles.price%TYPE,
    OUT maxPrice public.Titles.price%TYPE
)
RETURNS RECORD
AS
$$
DECLARE
BEGIN
    minPrice := (SELECT MIN(price) FROM titles);
    maxPrice := (SELECT MAX(price) FROM titles);
END
$$
LANGUAGE plpgsql
```

```
SELECT * FROM getMaxMinPrecio ();
```

Data Output			Messages	Notifications
	minprice	maxprice		
	numeric	numeric		
1	2.99	22.95		

2.3. Retornar una relación unaria



Relación unaria

Ver "2. Relación unaria" en el documento "Relations y SQL".



Relations y SQL

PL/pgSQL permite definir el valor de retorno de una función como `Setof <Tipo-De-Dato>`. Esto permite que una función retorne una especie de "lista" de valores.

Por un lado definimos el tipo de retorno de la función. Por ejemplo:

```
RETURNS setof FLOAT
```











Y en el cuerpo del procedimiento debemos utilizar una cláusula RETURN especial: `RETURN QUERY`. Por ejemplo:

Ejemplo

```
CREATE FUNCTION Ejemplo30 ()
  RETURNS setof date

AS
$$
DECLARE
BEGIN
  RETURN QUERY
    SELECT ord_date FROM Sales;
END;
$$
LANGUAGE plpgsql

SELECT Ejemplo30 ();
```

Data Output		Messages	Notifications
<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>			
	<div><div>ejemplo30</div><div>date</div><div></div></div>		
1	1994-09-14		
2	1994-09-14		
3	1994-09-14		
4	1994-09-14		
5	1994-09-14		

2.4. Retornar sets de tuplas completas



Conjunto de tuplas completas

Ver "4. Conjunto de tuplas completas" en el documento "Relations y SQL".

Si especificamos un valor de retorno como `setof <nombre-de-tabla>` la función podrá retornar cualquier conjunto de tuplas de la tabla especificada.

Por un lado definimos el tipo de retorno de la función:

```
RETURNS setof titles
```

Y en el cuerpo del procedimiento utilizamos `RETURN QUERY`. Por ejemplo:

Ejemplo

```
CREATE FUNCTION Ejemplo32 ()  
  RETURNS setof titles  
  
  AS  
  $$  
  DECLARE  
  BEGIN  
    RETURN QUERY  
    SELECT * FROM titles;  
  END;  
  $$  
LANGUAGE plpgsql
```

```
SELECT *  
  FROM Ejemplo32()  
 WHERE pub_id = '1389' ;
```

Data Output						Messages	Notifications
<div> <div> <div>+</div> <div> <div></div> <div></div> <div></div> </div> </div> <div> <div></div> <div></div> <div></div> </div> <div> <div></div> <div></div> <div></div> </div> <div> <div></div> <div></div> <div></div> </div> <div> <div></div> <div></div> <div></div> </div> <div> <div></div> <div></div> <div></div> </div> <div> <div></div> <div></div> <div></div> </div> </div>							
	title_id character varying (6)	title character varying (80)	type character	pub_id character	price numeric (10,2)		
1	PC1035	But Is It User Friendly?	popular_comp	1389	22.95		
2	PC8888	Secrets of Silicon Valley	popular_comp	1389	20.00		
3	BU1032	The Busy Executive's Database Guide	business	1389	19.99		
4	BU1111	Cooking with Computers: Surreptitious Balance Shee...	business	1389	11.95		

2.5. Retornar sets de projections de tuplas



Conjunto de Projections de tuplas

Ver “5. Conjunto de Projections de tuplas” en el documento “Relations y SQL”.

Si se necesita obtener una relación que posee un conjunto de Projections de tuplas podemos definir un composite-type con la estructura a retornar:

Schema level

```
CREATE TYPE publisherCT
AS (
    pub_id CHAR(4),
    totalPrice numeric
);
```



Composite Types

Ya vimos como crear y utilizar Composite types en “2.4. Sentencias *SELECT* Single-row que retornan una Projection de una tupla” en este mismo documento.

Luego nuestra function es definida para retornar este composite-type.

En el cuerpo del procedimiento también utilizamos `RETURN QUERY`:

Ejemplo

```
CREATE FUNCTION Ejemplo33 ()  
  RETURNS setof publisherCT  
  
  AS  
  $$  
  DECLARE  
  BEGIN  
    RETURN QUERY  
      SELECT pub_id, SUM(price) AS totalPrice  
      FROM titles  
      WHERE price IS NOT NULL  
      GROUP BY pub_id;  
  
  END;  
  $$  
  LANGUAGE plpgsql
```

```
SELECT *  
  FROM Ejemplo33();
```

Data Output			Messages	Notifications
	pub_id character	totalprice numeric		
1	0877	92.46		
2	1389	94.88		
3	0736	48.92		

2.6. Return Next

En PL/pgSQL también podemos retornar un record que asuma la forma de un Composite Type.

Es una forma que debemos usar cuando lo que queremos retornar NO ES resultado de un query, sino que se trata de algo que estamos armando por nosotros mismos.

Primero definimos un composite-type con la estructura deseada. Por ejemplo:

Ejemplo 14

```
CREATE TYPE publisherCT
AS (
    pub_id CHAR(4),
    totalPrice numeric
);
```

Y luego:

```
CREATE FUNCTION Ejemplo34 ()
    RETURNS setof publisherCT (A)

AS
$$
DECLARE
    fila publisherCT%ROWTYPE; (B)
BEGIN
    fila.pub_id := '0736';
    fila.totalPrice := 240.00;












    RETURN NEXT fila; (C)
END;
$$
LANGUAGE plpgsql
```

Nuestra function es definida para retornar sets de este composite-type (A).

Definimos una variable especial de tipo <composite-type>%rowtype (B).

En el cuerpo del procedimiento utilizamos una cláusula RETURN NEXT <tupla-del-composite-type> (C).

```
SELECT *  
FROM Ejemplo34();
```

Data Output		Messages	Notifications
        			
	pub_id character 	totalprice numeric 	
1	0736	240.00	