**Estándar de Codificación PLSQL**

Soluciones Técnicas Vital

# Versiones

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Autor** | **Version** | **Comentarios** |
| 23/03/2014 | Juan Bodnar | 1.0.0 | Version Inicial |
| 02/06/2014 | Juan Bodnar | 1.0.1 | Nuevas definiciones |
| 09/01/2015 | Juan Bodnar | 1.0.2 | No duplicar funciones |

# Indice

[Versiones 2](#_Toc29912624)

[Indice 3](#_Toc29912625)

[Objetivo 4](#_Toc29912626)

[Alcance 4](#_Toc29912627)

[Descripción General 4](#_Toc29912628)

[Nuevos Objetos 4](#_Toc29912629)

[Escritura de código en PL/SQL: 4](#_Toc29912630)

[Duplicado de Nombres (Funciones / Procesos): 5](#_Toc29912631)

[Nombre de los columnas 5](#_Toc29912632)

[Claves Primarias: 6](#_Toc29912633)

[Tablas en Sucursal y en AC 6](#_Toc29912634)

[Nombres de las FK y PK 6](#_Toc29912635)

[Packages en Sucursal y en AC 7](#_Toc29912636)

# Objetivo

Describir estándares y buenas prácticas para la creación de código c# en las nuevas soluciones Vital

# Alcance

El presente documento está destinado a Analistas, Desarrolladores, Altos mandos.

# Descripción General

## Nuevos Objetos

Para la creación de nuevos objetos tengamos presente usar los siguientes prefijos.

|  |  |
| --- | --- |
| Tipo | Nomenclatura |
| Packages | PKG\_(Nombre) |
| Procedure | No debería haber nuevos procedures independientes |
| Trigger | TRG\_(Nombre) |
| View | VIEW\_(Nombre) |
| Table | TBL(Nombre) |
| Sequence | SEQ\_(Nombre) |
| Function | No debería haber nuevas funciones independientes |
| Temporary Table | TBLTMP(Nombre) |

## Escritura de código en PL/SQL:

* El código tiene que estar auto-documentado. Fecha, Nombre de Programador, y una breve descripción de lo que hace el procedure/function.
* Cada modificación debe indicarse con un comentario que indique %v fecha – usuario y descripción de la modificación.
* Los parámetros de in/out de los procedures/function son p\_NombreParametro.
* Las variables locales/globales son v\_NombreVariable.
* Las constantes son c\_NombreConstante.
* Los nombres de procedures/function con letra capital y sin guiones. Ejemplos: AltaDeCliente, ObtenerDomicilio, GetContactos.
* Los procedures/function no deben superar en lo posible la página de líneas de código.
* Solo se publicarán en la especificación los procedures que sea estrictamente necesario para la comunicación con la aplicación cliente.
* Todos los procedures/function tienen que tener el logueo del módulo y el control de exception. Ejemplo:

PROCEDURE GetCompromisosPago (cur\_out out cursor\_type)

as

**v\_Modulo varchar2(100) := 'PKG\_CREDITOS.GetCompromisoPago';**

begin

--Conjunto de líneas de plsql a ejecutar

**exception**

**when others then**

**n\_pkg\_vitalpos\_log\_general.write(2,'Modulo: ' || v\_Modulo ||' Error: ' || SQLERRM);**

**Raise;**

END GetCompromisosPago;

## Cursores y ciclos

No utilizaremos la definición de cursores en la sección declarativa.

Para mayor claridad en la lectura del código, utilizaremos la forma implícita en el for ... loop, ejemplo:

for r\_xxx in (select ...

from ...)

loop

end loop

## Duplicado de Nombres (Funciones / Procesos):

No utilizaremos mismo nombre de proceso para funciones y procesos dentro del mismo paquete.

Si tenemos que elegir una de las 2 opciones tenemos que evaluar si el método requiere ser llamado desde afuera (Ejemplo: .NET). Desde las aplicaciones solo se puede llamar procedures.

Si no se requiere un llamado desde afuera de la base de datos se puede optar por la opción de una función que se llame desde dentro del bloque PL y no se publique en la especificación.

## Nombre de los columnas

El nombre de las columnas deberá comenzar siempre con:

am = importe/monto (ammovimiento)

cd = código que hace referencia a una tabla de configuración.  Siempre debe tener una referencia contra otra tabla. (cdcomprobante)

ds = descripción (dsrazonsocial)

dt = fecha (dtmovimiento)

ic = indicador o flag. No debe hacer FK con una tabla  (ej: icmuevecaja que puede contener 0 si no mueve o 1 si mueve caja).

id = identificador.  Clave de la tabla o referencia contra otra tabla. (idingreso)

nm = nombre (nmtarea)

qt = cantidad numérica (qtreimpresiones)

sq = número correlativo, puede provenir de una secuencia o de un numerador propio

vl = número no importe (vlrecargo)

## Claves Primarias:

Tablas transaccionales:

La PK debe estar formada por una sola columna que debe siempre comenzar con “id” y luego el nombre de la tabla del tipo varchar2(40) y los valores se generán con la función sys\_guid().

Además de ayudarnos a estandarizar también evitaremos duplicación de PK al centralizar la información en AC.

Ej: para una tabla llamada tblMovCuenta el campo de la PK debe ser idMovCuenta.

Tablas de configuración:

La PK debe estar formada por un solo campo que debe siempre comenzar con “cd” definido como varchar2.

Ej: para la tblEstadoAlivio el campo de la PK debe ser cdEstadoAlivio.

## Tablas en Sucursal y en AC

Cuando una tabla se replica desde AC a las sucursales o viceversa, debe ser igual en ambos ambientes y tener el mismo nombre.

Tablas transaccionales:

Las tablas transaccionales deberán contener siempre el cdSucursal.

Tablas de configuración:

Las tablas de configuración deben crearse en AC y como vista materializada de solo lectura en la sucursal.

Aquellas configuraciones que pueden diferir de una sucursal a otra deben tener el cdSucursal.

## Nombres de las FK y PK

El nombre de la PK de una tabla debe ser PK\_ y el nombre de la tabla (sin el tbl).

Ej: para la tabla tblMovCuenta la PK debe llamarse PK\_MovCuenta.

Los nombres de la FK dese ser FK\_TablaOrigen\_TablaDestino.

Ej: para una FK creada en la tblCobranza que la relaciona con la tblIngreso la FK debe llamarse FK\_Cobranza\_Ingreso.

## Packages en Sucursal y en AC

Si la funcionalidad de un pkg es exactamente igual en ambos ambientes, tiene que tener el mismo nombre. Si la funcionalidad coincide en algunos procedures pero hay diferencias, en AC debe tener el sufijo \_CENTRAL (ej: pkg\_cuenta\_central)