PERÚ | Ministerio de Comercio Exterior y Turismo | Secretaría General | Dirección General de Informática

Estándares

Estándares de Base de Datos

Preparado por:

Oficina de Informática - MINCETUR

Preparado el: martes, 29 de Marzo de 2022
Nombre de Archivo: DE-01_EstandaresDesarrollo_V02R01.doc

1 CONTROL DE VERSIONES

Fecha	Versión	Descripción	Autor
18/01/2000	V01R01	Versión Inicial	
04/10/2012	V02R01	Revisión	Jose Robles Margarita Argumedo
16/11/2015	V03R01	Modificado	Jose Robles, Margarita Argumedo, Marco Hinojosa
26/08/2016	V04R1	Modificado	José Robles, Margarita Argumedo, Marco Hinojosa

2 TABLA DE CONTENIDOS

3 GENERALIDADES

3.1 DEFINICIÓN

El establecimiento de estándares posibilita la implementación de una base de datos ordenada, comprensible y de fácil administración.

A través del uso de nomenclaturas, se crearán nombres significativos que permitan clasificar los objetos, facilitar el afinamiento de base de datos, así como también la interpretación y mantenimiento de programas referidos a objetos del mismo.

3.2 OBJETIVOS

- Establecer patrones para la asignación de nombres de objetos de base de datos.
- Permite contar con documentación adecuada, facilitando de esta manera la adaptabilidad de la base de datos ante una eventual modificación.

3.3 ALCANCE

Todos los lineamientos y reglas de definición detallados en el presente documento, serán de aplicación a todos los objetos de la base de datos del MINCETUR.

Los estándares de base de datos están dirigidos a analistas y desarrolladores, con el fin de homogenizar las estructuras de objetos de base de datos; siendo estos lineamientos aplicados a todos los sistemas de la institución.

3.4 AMPLIACIONES O MODIFICACIONES

Toda vez que surja la necesidad de ampliar o modificar el presente documento, se hará llegar una propuesta al Director General de la Oficina de Informática, quien designará al personal que en coordinación con el administrador de base de datos, evaluarán las propuestas.

3.5 TIPO DE DOCUMENTO

DE: Documento Extra (Estándar)

3.6 CÓDIGO DE DOCUMENTO

DE-01

4 OBJETOS DE LA BASE DE DATOS

4.1 TABLESPACE

Un tablespace es una unidad lógica de almacenamiento dentro de una base de datos ORACLE.

Formato:

TBS_<Nombre_tablespace>_<DAT> ó TBS_<Nombre_tablespace>_<IND>
Dónde: <TBS> identificador de tablespace de base de datos. <Nombre_tablespace>, con un máximo de 6 caracteres. <DAT> Indicador almacenamiento lógico de datos. <IND> Indicador almacenamiento lógico de índices.

Ejemplo:

TBS_COM_DAT TBS_COM_IND
✓ TBS, identificador de tablespace base de datos. ✓ COM, tablespace del ambiente de Comunicaciones. ✓ DAT, almacena datos. ✓ IND, almacena índices.

4.2 DATAFILES

Se antepone el identificador de datafile, luego el nombre del datafile, seguido de un punto luego las letras "DBF", el nombre del datafile debe mantener el nombre del tablespace al cual representa.
Cuando un tablespace tiene varios datafiles se va numerando consecutivamente en la variable (NN).

Formato:

TBS_<Nombre_datafile>_<DAT>NN.DBF ó TBS_<Nombre_datafile>_<IND>NN.DBF
Dónde: <TBS> identificador de tablespace base de datos. < Nombre_datafile>, con un máximo de 6 caracteres. <DAT> Indicador almacenamiento físico de datos. <IND> Indicador almacenamiento físico de índices. <NN> Número consecutivo según secuencia de datafile correspondiente. <DBF> Indicador de datafile.

Ejemplo:

TBS_COM_IND01.DBA TBS_COM_IND02.DBA
✓ TBS, identificador de tablespace base de datos. ✓ COM, tablespace del ambiente de Comunicaciones. ✓ IND, almacena índices. ✓ NN, número consecutivo de datafile. ✓ DBF, Indicador de datafile.

4.3 USUARIOS ADMINISTRADOR (ESQUEMA)

Se antepone el prefijo de sistema o negocio seguido de un guion luego de las letras "ESQ", el prefijo deberá ser coordinado con el administrador de base de datos.

Formato:

<Prefijo sistema>_ESQ
Dónde: < Prefijo sistema>, prefijo de sistema o negocio con un máximo de 3 caracteres.

Ejemplo:

INS_ESQ
✓ INS, prefijo del esquema de sistemas institucionales. ✓ ESQ, identificador de objeto de base de datos.

Nota.- Se cuenta con esquemas por línea de negocio (turismo, comercio e institucionales), donde se despliegan los objetos de base de datos de aplicaciones de pequeña y mediana envergadura distinguiéndolos por el prefijo de origen. Las aplicaciones o sistemas de gran envergadura podrán tener su propio esquema de base de datos, adquiriendo el nombre de la dirección general o nacional que representan.

4.4 USUARIOS FINAL

Usuarios creados para aplicaciones de tipo clientes servidor; el empleo de usuarios del tipo clientes servidor debe ser coordinado con el administrador de base de datos y deberá contar con la justificación del caso.

Formato:

<Siglas_oficina>_<Nombre_usuario>
Dónde: <Siglas_oficinas>, con un máximo de 4 caracteres <Nombre_usuario>, debe ser igual a los nombres que figuran en el directorio Activo de la Institución.

Ejemplo:

ESTA_JHERNANDEZ
✓ ESTA, abreviatura de la oficina de estadísticas a la cual pertenece el usuario. ✓ JHERNANDEZ, nombre de funcionario.

4.5 USUARIO DE CONEXIÓN

Todas las aplicaciones contarán con un usuario de conexión, el cual tendrá asignados los roles y atributos necesarios para su funcionamiento.
Solo si fuera necesario el administrador de base de datos en coordinación con el analista responsable crearán un usuario extra de conexión.

Formato:

<Nombre_Esquema>_<USER>_<XXX>
Dónde: <Nombre de Esquema>, prefijo de esquema con un máximo de 3 caracteres. <USER>, identificador de usuario de conexión. <XXX>, identificador de usuario extra.

Ejemplo:

INS_USER INS_USER_REP
✓ INS, prefijo del esquema institucional. ✓ USER, identificador de usuario de conexión. ✓ REP, Usuario reporteador.

4.6 USUARIOS DE CONEXIÓN DE ANALISTA

Detalle en anexo 1 adjunto (documento interno).

4.7 ROLES

Se podrán crear roles para otorgar permisos a los esquemas, usuarios finales y de conexión.

4.7.1 ROLES DE PERSONALIZADOS

El nombre del rol se construirá colocando el nombre del esquema de base de datos, seguido de un guión bajo, luego el nombre de la función del rol, seguido de un guión bajo, luego el prefijo del módulo y/o sistema, adicionalmente se puede indicar un guión bajo seguido de alguna particularidad propia del módulo y/o sistema.

Función del rol:

SEL	: Lectura.
UPD	: Actualización.
INS	: Insertar.
DEL	: Borrar.
EXE	: Ejecutar.

Formato:

<Nombre_esquema>_<Función_rol>_<Prefijo_modulo>[_ZZZ]
Dónde: <Nombre_esquema>, prefijo de quema con un máximo de 3 caracteres. <Función_rol>, con un máximo de 3 caracteres. <Prefijo_modulo>, prefijo de sistema o negocio con un máximo de 3 caracteres. [_ZZZ], opcional, entre 2 o 3 caracteres.

Ejemplo:

INSDBA_SEL_DP
✓ INSDBA, esquema institucional. ✓ SEL, función de consulta del rol. ✓ DP, prefijo del sistema de legajo de personal.

4.7.2 ROLES APLICACIÓN

Roles creados para otorgar permisos a usuarios finales o de conexión, generalmente los usuarios de conexión tendrán dos roles uno de consulta y otro de ejecución.

Formato:

ROL_<Usuario_final> **ROL_<Usuario_conexión>**
Dónde: <Usuario final>, definido en el punto 4.4 ó 4.5 <Usuario conexión>, definido en el punto 4.4 ó 4.5

Ejemplo:

ROL_TUR_JPEREZ
✓ ROL, Identificador de ROL. ✓ TUR_JPEREZ, Usuario final

4.8 TABLAS

Las tablas se nombrarán con un identificador genérico de 25 caracteres como máximo, donde se colocará el prefijo de origen o módulo o sistema en los primeros caracteres, seguido del identificador del tipo de tabla, luego se prosigue con el nombre descriptivo de la tabla, el cual podrá estar construido con los nemónicos necesarios. El nombre de tablas no podrá contener guiones bajos en su definición.

Los nombres de las tablas deberán ser lo más descriptivos posible, tratando de incluir las características más saltantes de la información que contendrán.

Tipos de tablas:

1	Maestro	Contiene información principal del Sistema y/o aplicación (tabla Maestra).
2	Detalle	Contiene información complementaria de otra tabla relacionada.
3	Transaccional	Contiene información de un determinado evento.
4	Temporal	Contiene información transitoria, luego de usarla se debe borrar.
5	Acumulado	Contiene Información acumulada.
6	Auditoría	Contiene información de un determinado periodo de tiempo o log.
7	Puente	Tabla que permite realizar un enlace de relación entre dos tablas.
8	Tipo	Contiene información descriptiva de datos básicos del sistema.
9	Histórico	Tablas históricas por un determinado tiempo

Formato:

<Prefijo_origen><Tipo_tabla><Nombre_descriptivo_tabla>
Dónde: <Prefijo_origen>, prefijo de esquema o negocio con un máximo de 3 caracteres. <Tipo tabla>, valor numérico de un carácter, detallado a continuación. <Nombre_descriptivo_tabla>, con un mínimo de 3 caracteres.

Ejemplo:

SGA1USU
✓ SGA, prefijo de origen. ✓ 1, tipo de tabla maestra. ✓ USU, abreviatura de la tabla de usuarios.

Nota.- Solamente se otorgarán permisos de REFERENCES de tablas entre esquemas, para poder generar las llaves foráneas, los permisos adicionales debe ser justificados al administrador de base de datos y al analista responsable del proyecto.
De preferencia crear vistas para poder consultar data entre esquemas.

4.9 COLUMNAS

El nombre de las columnas debe tener como máximo una longitud de 20 caracteres, donde al inicio se colocará un identificador de tipo de columna, seguido de un guion bajo "_", y a continuación el nombre descriptivo de columna.

Formato:

<Tipo_Columna>_<Nombre_descriptivo_columna>
Dónde: <Tipo columna>, con un máximo de 3 caracteres. < Nombre_descriptivo_columna>, identificador asociado a un campo.

Ejemplo:

DES_EQPO
✓ DES, indica que es un campo descripción. ✓ EQPO es la abreviatura de equipo.

Podrán usarse abreviaturas en el identificador asociado a un campo, pero éstas deben tener un significado consistente en toda la aplicación.

Los campos que forman parte de la llave principal de la tabla deben ser creados definiéndolos como campos no nulos, a continuación mostramos un ejemplo;

```
ID_VISITA       NUMBER CONSTRAINT NN_CV6VISITA01 NOT NULL,  -- SIENDO CV6VISITA EL NOMBRE DE LA TABLA
```

Si en un campo se abrevia la palabra "EQUIPO" como EQPO, en el resto campos de tablas relacionadas deberán mantener la misma abreviatura.

Toda sentencia de SQL se realizará con el nombre de esquema o alias asociado. Así escribimos E.FLG_EST, suponiendo que ST1EQPO con el alias asociado "E", se debe usar el mismo alias para referenciar la misma tabla, eso quiere decir que en toda la aplicación se deberá identificar dicha tabla con le letra E.

Detalle de tipo de columnas:

ID	Campo Identificador o llave principal o foránea; tipo de dato numérico (number(n))
DES	Campo Descripción; tipo de dato carácter (varchar2(n)).
COD	Campo Identificador que no sea llave; tipo de dato carácter (varchar2(n)).
NUM	Campo Numérico (number(n)).

ABR	Campo Abreviatura; tipo de dato carácter (varchar2(n)).
FEC	Campo Fecha; tipo de dato fecha (date)
FLG	Campo flag cuyos valores deben ser 1 o 0; tipo de dato numérico (number(n)).

Los campos VARCHAR2 así como el campo NUMBER deben ser dimensionados.

Los campos de estado deben definirse de la siguiente manera FLG_EST, los cuales debe ser campos numéricos cuyos valores son 0,1.

Todas las tablas deben contar con campos de auditoría, los cuales deben ser nombrados de la siguiente forma:

- ✓ Fecha y usuario de creación de la siguiente manera FEC_CREA, USU_CREA.
- ✓ Fecha y usuario de modificación FEC_MODI, USU_MODI; los campos de usuario son de tipo NUMBER(8).

La nomenclatura de los campos de tipo LOB debe identificar el tipo de contenido de almacenamiento (Blob y Clob) y deben ser direccionados al TableSpaces adecuado, de igual manera los Índices LOB.

La creación de campos CLOB y/o BLOB en esquemas de negoció están restringidos y esto debido a la arquitectura de la red de la institución, dicha coordinación se debe realizar con el analista responsable del proyecto.

Se han creado un esquema de base de datos con una configuración dirigida a la actualización y recuperación de archivo con estas características, para tal fin se crearon 2 tablas:

- ✓ Una que contiene una columna BLOB donde guardaremos los archivos adjuntos, cuyo identificador es ID_DOC NUMBER(9).

- ✓ La otra con un campo CLOB para el guardado de formatos htm básicamente, cuyo identificador es ID_FORM NUMBER(9).

El acceso a mencionadas tablas se realiza por intermedio de la DLL. Los esquemas de negocio solo guardarán el ID único del archivo BLOB o CLOB, el cual será retornado por la DLL, deben ser levantadas referencias entra el id del BLOB o CLOB (la tabla negocio) y el ID de la tabla que contiene el BLOB o CLOB.

4.10 VISTAS

Se antepone las iniciales "**VI**", seguido de un guion bajo, luego el prefijo del módulo y/o sistema y seguido del nombre nemónico de la vista**.**

Formato:

<VI>_<Prefijo_origen><Nombre_descriptivo_tabla>
Dónde: <VI>, identificador de tipo de objeto de base de datos. <Prefijo_origen>, prefijo identificador del negocio, el cual debe tener entre 2 o 3 caracteres. <Nombre_descriptivo_tabla>, descripción de tabla u objeto de negocio a representar.

Ejemplo:

VI_DIMPERSONA

✓ VI, identificador de tipo de objeto de base de datos vista. ✓ DIM, prefijo del esquema de personas. ✓ PERSONA, vista que contienen todos los funcionarios de la institución.

4.11 INDICES

Se antepone las iniciales "IX", seguido de un guion bajo, luego el nombre de la tabla con un numero consecutivo de dos dígitos.

Formato:

<IX>_ Nombre_tabla><Correlativo>
Dónde: <IX> identificador de tipo de objeto de base de datos. <Nombre_tabla>, nombre de tabla a la que pertenece el índice. <Correlativo>, correlativo de índice por tabla.

Ejemplo:

XI_DIM1PERSONA01
✓ XI, identificador de tipo de objeto de base índice de llave primaria. ✓ DIMPERSONA, tabla de personas. ✓ 01, correlativo de índices.

Nota.- EL índice que hace referencia a la llave primaria de la tabla debe ser creada de la siguiente manera.

Formato:

<PK>_Nombre_tabla>
Dónde: <PK> identificador de tipo de objeto de base de datos, índice de llave primaria. <Nombre_tabla>, nombre de tabla a la que pertenece el índice.

Ejemplo:

PK_PER1MAEEMP
✓ PK, identificador de tipo de objeto de base de datos vista, índice de llave primaria. ✓ PER1MAEEMP, nombre de tabla.

4.12 CONSTRAINT

Los constraints nos indican ciertas condiciones o características especiales de algunos campos de una tabla. Los constraints utilizados con mayor frecuencia y que deben tener una nomenclatura fácil de reconocer, son: Llave Primaria (Primary Key), Llave Foránea (Foreign Key), Restricción de Unicidad (Unique Constraint) y Restricción de Verificación (Check Constraint).

a. **Primary Key**: Llave única de la tabla, la cual garantiza que no se ingresen 2 registros iguales con esta misma llave; el nombre del constraint se construirá con el Identificador PK seguido por un guion bajo "_" y luego el nombre de la tabla al que pertenece.

Formato:

PK_<Nombre_tabla>
Dónde: <PK>, identificador de tipo de objeto de base de datos de constraints de llave primaria. <Nombre_tabla>, nombre de la tabla.

Ejemplo:

PK_ST1EQPO
✓ PK, identificador de tipo de objeto de base de datos, constraints de llave primaria. ✓ ST1EQPO, nombre de tabla.

b. **Foreign Key**: Llave que sirve para hacer referencia a otra tabla a través de su llave primaria; el nombre del contraints se construirá con el identificador FK seguido de un guion bajo "_", luego el nombre de la tabla origen o hija, a continuación un guion bajo y finalmente el nombre de la tabla a la que hace referencia o padre.

Formato:

FK_<nombre_tabla_hija>_<nombre_tabla_padre>
Dónde: <FK>, identificador de tipo de objeto de base de datos constrain de llave primaria. <nombre_tabla_hija>, nombre de la tabla referenciada. <nombre_tabla_padre>, nombre de la tabla que contiene la llave primaria.

Ejemplo:

FK_ST7PARTEQPO_ST1EQPO
✓ FK, identificador de tipo de objeto de base de datos, constraints de llave foránea. ✓ ST7PARTEQPO, nombre de tabla hija ✓ ST1EQPO, nombre de tabla padre.

c. **Unique Constraint**: Asegura la unicidad de un registro al igual que la llave primaria, pero no sirve para referencias; el nombre del constraints se construirá con el identificador UN seguido de un guion bajo, luego el nombre de la tabla al cual pertenece y finalmente un correlativo de dos dígitos.

Formato:

UN_<nombre_tabla><correlativo>
Dónde: <UN>, identificador de tipo de objeto de base de datos contraints único. <nombre_tabla>, nombre de la tabla

<correlativo>, nombre de la tabla, valor de dos dígitos.

Ejemplo:

UN_ST7PARTEQPO01
✓ UN, identificador de tipo de objeto de base de datos, constraints único. ✓ ST7PARTEQP, nombre de tabla ✓ 01, correlativo de constrain único.

d. **Check Constraint**: Verifica los valores que ingresan en un campo. El nombre del constraints se construirá con el identificador CK, seguido de un guion bajo, luego el nombre de la tabla al cual pertenece y finalmente un correlativo de dos dígitos.

Formato:

CK_<nombre_tabla><correlativo>
Dónde: <CK>, identificador de tipo de objeto de base de datos de contraints check. <nombre_tabla>, nombre de la tabla <correlativo>, correlativo de check constraints, valor de dos caracteres**.**

Ejemplo:

CK_ST7PARTEQPO01
✓ CK, identificador de tipo de objeto de base de datos, constraints único. ✓ ST7PARTEQP, nombre de tabla ✓ 01, correlativo de constrain único.

Nota.- Para el caso de Constraints **NOT NULL,** específicamente para las columnas de llaves primarias, se define el nombre anteponiendo las iniciales “NN”, seguido de un guion bajo, luego el nombre de la tabla con un número consecutivo de dos dígitos.

Formato:

NN_<nombre_tabla><correlativo>
Dónde: <NN>, identificador de tipo de objeto de base de datos de contraints de not null. <nombre_tabla>, nombre de tabla <correlativo>, correlativo, valor de dos dígitos.

Ejemplo:

NN_PER1MAEEMP01 NN_PER1MAEEMP02
✓ NN, identificado de objeto de base de datos constraints no nulo. ✓ PER1MAEEMP, nombre de tabla. ✓ 01, correlativo.

4.13 SINÓNIMOS

Se antepone las iniciales "SI", seguido de un guion bajo, luego el nombre de la tabla y/o vista, la creación de sinónimos debe ser justificado y coordinar administrador de base de datos y analista responsable del proyecto, ya que la institución no trabaja con sinónimos.

Formato:

SI_<nombre_tabla>
Dónde: <SI>, identificador de tipo de objeto de base de datos sinónimos. <nombre_tabla>, nombre de tabla.

Ejemplo:

SI_PER1MAEEMP SI_VI_PER1MAEEMP
✓ SI, identificador de tipo objeto de base de datos de sinónimos de una tabla o vista. ✓ PER1MAEEMP o VI_PER1MAEEMP, nombre de tabla o vista.

4.14 SECUENCIAS

Se antepone las iniciales "SE", seguido de un guion bajo, luego el prefijo del módulo y/o sistema seguido del nombre nemónico de la secuencia.

Formato:

SE_<nombre_sequencia>
Dónde: <SE>, identificador de tipo de objeto de base de datos secuencias. <nombre secuencia>, nombre de secuencia.

Ejemplo:

SE_PERNUMCOD
✓ SE, identificador de tipo objeto de base de datos de secuencia. ✓ PER, prefijo de la secuencia. ✓ NUMCOD, nombre de secuencia.

4.15 PAQUETES

Los paquetes deberán tener un nombre que indique claramente el uso de los mismos o el propósito general de los objetos que contiene. La definición del nombre se realizará anteponiendo al inicio el identificador PKG, seguido de guion bajo para luego indicar el prefijo de origen y finalmente especificar en nombre descriptivo del paquetes. El nombre no debe exceder a 25 caracteres como máximo.

Formato:

PKG_<prefijo_origen><nombre_descriptivo_package><tipo_package>
Dónde: <PKG>, identificador de tipo de objeto de base de datos paquete. <prefijo_origen>, prefijo de origen asignado al sistema. <nombre paquete>, nombre descriptivo del paquete.

Ejemplo:

PKG_CEPROMOCONSU
✓ PKC, identificador de tipo objeto de base de datos de paquete. ✓ CE, prefijo de origen del sistema de promotores. ✓ PROMO, nombre descriptivo del paquete. ✓ CONS, tipo de paquete de consulta.

Tipos de paquetes:

CONS	Paquetes que realizaran únicamente consultas.
MANT	Paquetes que agregarán, actualizarán o eliminarán registros.
REP	Paquetes que generan reportes, siempre y cuando la envergadura del proyecto lo justifique.

Deben ser exclusivos los paquetes que realizan consultas de los que realizan registros, actualizan o eliminan registros (toda eliminación de data debe ser eliminado lógico).

Los objetos de los paquetes deben estar declarados en el mismo orden tanto en la cabecera como en el cuerpo; los objetos contenidos en los paquetes deben estar ordenados alfabéticamente, en 2 grupos primero las funciones y luego los procedimientos.

4.16 PROCEDIMIENTO

Se antepone las iniciales “PRC”, seguido de un guion bajo, luego el prefijo del módulo y/o sistema, seguido del nombre del procedimiento, el nombre no debe exceder los 30 caracteres en total.

Formato:

PRC_<prefijo_origen><nombre procedimiento>
Dónde: <PRC>, identificador de tipo de objeto de base de datos procedimiento. <prefijo_origen>, prefijo de esquema o negocio con un máximo de 3 caracteres. <nombre_procedimiento>, nombre de procedimiento.

Ejemplo:

PRC_DPALTABAJAGENERAL
✓ PRC, identificador de tipo objeto de base de datos de procedimiento. ✓ DP, prefijo de origen de legajo de personal.

✓ ALTABAJAGENERAL, nombre descriptivo del procedimiento.

4.17 FUNCION

Se antepone las iniciales “FNC”, seguido de un guion bajo, luego el prefijo del módulo y/o sistema, seguido del nombre nemónico de la función, el nombre no debe exceder los 30 caracteres en total.

Formato:

FNC_<prefijo_origen><nombre_funcion>
Dónde: <FNC>, identificador de tipo de objeto de base de datos función. <prefijo_origen>, prefijo de esquema o negocio con un máximo de 3 caracteres. <nombre_funcion>, nombre descriptivo de la función.

Ejemplo:

FNC_SGATRAE_ROL_USU
✓ FNC, identificador de tipo objeto de base de datos de función. ✓ SGA, prefijo de origen de sistema. ✓ TRAE_ROL_USU, nombre descriptivo de la función.

4.18 DISPARADOR

Se antepone las iniciales “TRG”, seguido de un guion bajo, luego el prefijo del módulo y/o sistema, seguido del nombre nemónico del disparador, el cual no debe exceder los 25 caracteres.

Formato:

TRG_<prefijo_origen><nombre_disparador>
Dónde: <TRG>, identificador de tipo de objeto de base de datos de disparador. <prefijo_origen>, prefijo de esquema o negocio con un máximo de 3 caracteres. <nombre_disparador>, nombre descriptivo del disparador

Ejemplo:

TRG_SGAREGCOMI
✓ TRG, identificador de tipo objeto de base de datos de disparador. ✓ SGA, prefijo de origen de sistema. ✓ REGCOMI, nombre descriptivo del disparador.

5 CONSIDERACIONES

5.1 OBSERVACIONES GENERALES

5.1.1 Estructura general de los objetos

El prefijo de origen: indicador del origen de desarrollo y/o área de negocio (nombre del esquema o abreviación del mismo), los límites en cuanto a cantidad de caracteres serán de 2 y 3 dígitos. EL prefijo será asignado por el administrador de base de datos y analista responsable del proyecto.

<Prefijo_origen>
Dónde : SGA, prefijo origen del esquema seguridad.

5.1.2 Especificaciones generales de los objetos

Todos los objetos de base de datos deben de ser creados con las siguientes especificaciones:

- ✓ En mayúsculas.
- ✓ En singular.
- ✓ En castellano.
- ✓ Sin utilizar espacios en blanco.
- ✓ El empleo de nuevos usuarios del tipo clientes servidor debe ser coordinado con el administrador de base de datos y deberá contar con la justificación del caso.
- ✓ Los nombres de tablas no debe contener el símbolo (-) guion bajo.
- ✓ Los datos se deben guardar en mayúsculas y mediante las consultas manejar el formato de visualización.
- ✓ Los *nemónicos* deben estar compuestos de cadenas cortas de caracteres, los cuales a simple vista deben expresar un significado.
- ✓ No usar caracteres especiales como #%&/Ñ para los nombres descriptivos de los objetos de base de datos.
- ✓ Toda eliminación de tabla debe ser eliminado lógico.
- ✓ Cuando se referencia a los objetos se debe indicar el nombre del esquema seguido del nombre del objeto.

Formato:

<esquema_base_datos>.<nombre_objeto>
Dónde : <esquema_base_datos>, esquema donde se crean los objetos de base de datos. <nombre_objeto>, nombre de objeto de base de datos.

Ejemplo:

SGADBA.SG8USISROL

SGADBA, esquema de seguridad. SG8USISROL, nombre de tabla de sistemas roles.

- ✓ Las variables globales, variables locales, parámetros que requerimos en los paquetes, procedimientos, funciones y disparadores deben tener las siguientes características:

  Variables Globales: son las variables definidas en la cabecera de los paquetes, cuyo nombre se construirá antecediendo el identificador G, seguido de la descripción del nombre del campo.

Formato:

<G><Nombre_descriptivo_variable>
Dónde: <G>, Indicador de variable global <Nombre_descriptivo_variable>, nombre del objeto de base de datos.

Ejemplo:

G_DESPACKGE
G, identificador de variable global. SG8USISROL, nombre de tabla de sistemas roles.

Variables Locales: son las variables requeridas en los procedimientos, funciones y disparadores, las cuales deben iniciar con el identificador L, seguido de la descripción del nombre del campo.

Formato:

<L><Nombre_descriptivo_variable>
Dónde: <L>, Indicador de variable local <Nombre_descriptivo_variable**>,** nombre del objeto de base de datos.

Ejemplo:

L_ID_EQUIPO
L, identificador de variable local. ID_EQUIPO, variable utilizada para almacenar el identificador de equipo.

Parámetros: son utilizados para ingresar (IN) o extraer (OUT) valores cuando ejecutamos algún procedimiento, función o disparador, se identifican iniciando con el carácter X.

Formato:

<X><Nombre_descriptivo_parámetro>

Dónde: <X>, Indicador de parámetro <Nombre_descriptivo_variable>, nombre del objeto de base de datos.

Ejemplo:

X_ID_USU
L, identificador de parámetro. ID_USU, variable utilizada para almacenar el identificador de usuario.

- ✓ De preferencia la descripción de los nombres de variables o parámetros se construirán bajo las mismas especificaciones detalladas para la creación de columnas de tablas y si estas representaran alguna columna deberán ser la copia exacta de la misma.
- ✓ Se debe mostrar las variables o parámetros en el siguiente orden, si corresponden a columnas de alguna tabla, parámetros que no corresponden a alguna tabla y finalmente variables de retorno.
- ✓ Siempre que una variable sea copia de un campo definido para una tabla de la Base de datos, se empleará la cláusula %TYPE en lugar de declarar explícitamente el tipo y la longitud de la misma. Esto facilita el mantenimiento de los programas, siempre y cuando estén dentro del mismo esquema de base de datos.
- ✓ Los índices empleados para bucles (loop, for,...) que no tengan un significado especial se nombrarán simplemente por la letra REG o X,Y,Z. Para bucles anidados se añadirán números secuenciales que indiquen la profundidad.
- ✓ Las variables se declaran en la sección definida para tal fin entre el AS o IS y el BEGIN;
- ✓ Para el manejo de excepciones a nivel de base de datos, se considerará el siguiente procedimiento.

```
----------------------PROCEDIMIENTO PARA EL REGISTRO DE EXCEPCIONES

SGADBA.PKG_SGA_AUDIT.PRC_SGA6LOGERROR(
    X_OPR                   => 'I',                             --REGISTRO DE EXCEPCION
    X_ID_SIS                => <CODIGO_SISTEMA>,
    X_COD_FUENTE            => 'B',                             --ERROR DE ORIGEN DE BASE DE DATOS
    X_DES_PAGINAOBJETO      => <NOMBRE_PAQUETE>||'.'||<NOMBRE_PROCEDIMIENTO>,
    X_DES_ERROR             => DESCRIPCIÓN DEL ERROR,           --DESCRIPCIÓN DE LA EXCEPCIÓN DE
BASE DE DATOS
    X_USU_CREA              => USUARIO QUE REALIZA LA ACCIÓN,
    X_IP_CREA               => IP DONDE SE RELIZA LA ACCIÓN,
    X_ID_LOGERROR           => TICKET DE ERROR,
    ERROR                   => RETORNO DEL ERROR);
```

Ejemplo:

```
CREATE OR REPLACE PACKAGE BODY PKC_<PREFIJO_ORIGEN><NOMBRE_DESCRIPTIVO_PACKAGE>
--------------------------------------------------------------------------------
--DESCRIPCIÓN:    BREVE DESCRIPCIÓN DEL OBJETIVO DEL PACKAGE
--PARAMETROS:    DESCRIPCIÓN DE PARÁMETROS CUYA UTILIDAD NO SEA EVIDENTE
--------------------------------------------------------------------------------

PROCEDURE PRC_<PREFIJO_ORIGEN><NOMBRE_DESCRIPTIVO_PROCEDIMIENTOS> (
    <PARMETRO>              <DATA TYPE>,
    <PARMETRO>              IN OUT <DATA_TYPE>,
    <PARMETRO>              OUT <DATA_TYPE>
--------------------------------------------------------------------------------
--DESCRIPCIÓN:    BREVE DESCRIPCIÓN DEL OBJETIVO DEL PROCEDIMIENTOS
--PARAMETROS:    DESCRIPCIÓN DE PARÁMETROS CUYA UTILIDAD NO SEA EVIDENTE
--------------------------------------------------------------------------------
)
IS
```

```
    <VARIABLE_LOCAL>        <VARCHAR2 (TAMAÑO)>,
    <VARIABLE_LOCAL>        <NUMBER (TAMAÑO)>
BEGIN
    BEGIN
        UPDATE ESQUEMA.TABLA SET
            CAMPO1 = L_CAMPO1,
            CAMPO2 = L_CAMPO2,
            CAMPO3 = L_CAMPO3
        WHERE
              CAMPO4 = L_CAMPO4
          AND CAMPO5 = L_CAMPO5;
    EXCEPTION WHEN OTHERS THEN
        SGADBA.PKG_SGA_AUDIT.PRC_SGA6LOGERROR (
            X_OPR                   => 'I',                             --REGISTRO DE EXCEPCION
            X_ID_SIS                => <CODIGO_SISTEMA>,
            X_COD_FUENTE            => 'B',                             --ERROR DE ORIGEN DE BASE
DE DATOS
            X_DES_PAGINAOBJETO      => <NOMBRE_PAQUETE>||'.'||<NOMBRE_PROCEDIMIENTO>,
            X_DES_ERROR             => DESCRIPCIÓN DEL ERROR,           --DESCRIPCIÓN DE LA
EXCEPCIÓN DE BASE DE DATOS
            X_USU_CREA              => USUARIO QUE REALIZA LA ACCIÓN,
            X_IP_CREA               => IP DONDE SE RELIZA LA ACCIÓN,
            X_ID_LOGERROR           => TICKET DE ERROR,
            ERROR                   => RETORNO DEL ERROR);
    END;
END;

END PKC_<PREFIJO_ORIGEN><NOMBRE_DESCRIPTIVO_PACKAGE>;
```

✓ **Indentación de sentencias SQL**

La codificación de cualquier sentencia DML (SELECT, UPDATE, DELETE o INSERT) debe emplear la indentación para su clarificación, facilitando de esta manera la búsqueda de las tablas implicadas (FROM) y de las condiciones impuestas (WHERE). Cuando está implicada más de una tabla, deben emplearse los alias asignados a cada tabla o vista para cualificar los campos ayudando de este modo, al optimizador a realizar el PARSE de la sentencia, en lo posible usar el mismo alias asignado a una tabla o vista.

A continuación mostramos un formato genérico para todas las sentencias que SQL nos permite construir, con la finalidad de contar con sentencias legibles y homogéneas a través de toda la base de datos.

SELECT
Si es necesario recuperar muchos campos, podrían agruparse varios por línea:

```
-----------------------------SELECT--------------------------
SELECT
    I.CAMPO1,
    I.CAMPO2,
    I.CAMPO3
INTO
    L_ICAMPO1,
    L_ICAMPO2,
    L_ICAMPO3,
    L_TCAMPO1,
    L_TCAMPO2
FROM
     ESQUEMA.TABLA1 I
WHERE
  --COMENTARIO 1ERA CLAUSULA WHERE
      I.CAMPO4 = 1000
  --COMENTARIO 1ERA CLAUSULA WHERE
  AND I.CAMPO5 = 25;
```

JOIN
Como se ha comentado en diferentes ocasiones, se emplean las abreviaturas o alias para cualificar los campos cuando se trata de un *join*:

```
-----------------------------SELECT JOIN--------------------------
SELECT
    I.CAMPO1,
```

```
    I.CAMPO2,
    I.CAMPO3,
    T.CAMPO1,
    T.CAMPO2
INTO
    L_ICAMPO1,
    L_ICAMPO2,
    L_ICAMPO3,
    L_TCAMPO1,
    L_TCAMPO2
FROM
     ESQUEMA.TABLA1 I
     LEFT JOIN ESQUEMA.TABLA2 T ON I.CAMPO1 = T.CAMPO2
WHERE
  --COMENTARIO 1ERA CLAUSULA WHERE
      I.CAMPO4 = 1000
  --COMENTARIO 1ERA CLAUSULA WHERE
  AND I.CAMPO5 = 25;
```

INSERT

En los INSERT es importante destacar que deben especificarse los nombres de los campos para mantener la independencia lógica de la sentencia con respecto a los cambios en la definición de la tabla:

```
----------------------------INSERT-----------------------------------
INSERT INTO ESQUEMA.TABLA (
    CAMPO1,
    CAMPO2,
    CAMPO3
) VALUES (
    'J',
    'DOCUMENTO A JUSTIFICAR',
    'A JUSTIFICAR'
);
```

UPDATE

Igualmente, se especifican los nombres de los campos para mantener la independencia lógica en la Sentencia UPDATE:

```
----------------------------UPDATE-----------------------------
UPDATE ESQUEMA.TABLA SET
    CAMPO1 = L_CAMPO1,
    CAMPO2 = L_CAMPO2,
    CAMPO3 = L_CAMPO3
WHERE
      CAMPO4 = L_CAMPO4
  AND CAMPO5 = L_CAMPO5;
```

SUBQUERYS

Es interesante mostrar, por último, como podrían formatearse los *subquery* anidados:

```
----------------------------SELECT-----------------------------
SELECT
    I.CAMPO1,
    I.CAMPO2,
    I.CAMPO3
INTO
    L_CAMPO1,
    L_CAMPO2,
    L_CAMPO3
FROM
      ESQUEMA.TABLA1 I
WHERE
      I.CAMP4 = 56
  AND T.CAMPO4 NOT IN (
               SELECT T.CAMPO1
               FROM ESQUEMA.TABLA1 T);
```

- ✓ **Indentación en las declaraciones de las funciones y/o Procedimientos**

  Al declarar una función y/o procedimiento debe conservarse el orden de indentación tal como se muestra a continuación

```
----------------------------CABECERA DE PAQUETE---------------------------------

CREATE OR REPLACE PACKAGE PKC_<PREFIJO_ORIGEN><NOMBRE_DESCRIPTIVO_PACKAGE>
G_NOMPACK     VARCHAR2(25) := 'PKG_DIMCONSU';
--------------------------------------------------------------------------------
--DESCRIPCIÓN:    BREVE DESCRIPCIÓN DEL OBJETIVO DEL PACKAGE
--PARAMETROS:     DESCRIPCIÓN DE PARÁMETROS CUYA UTILIDAD NO SEA EVIDENTE
--------------------------------------------------------------------------------

    G_NOMPACK               VARCHAR2(25 BYTE);
--------------------------------------------------------------------------------
PROCEDURE PRC_<PREFIJO_ORIGEN><NOMBRE_DESCRIPTIVO_PROCEDIMIENTO> (
    <PARMETRO>              <DATA_TYPE>,
    <PARMETRO>              IN OUT <DATA_TYPE>,
    <PARMETRO>              OUT <DATA_TYPE>
--------------------------------------------------------------------------------
--DESCRIPCIÓN:    BREVE DESCRIPCIÓN DEL OBJETIVO DEL PROCEDIMIENTOS
--PARAMETROS:     DESCRIPCIÓN DE PARÁMETROS CUYA UTILIDAD NO SEA EVIDENTE
--------------------------------------------------------------------------------
);
--------------------------------------------------------------------------------
FUNCTION FNC_<PREFIJO_ORIGEN><NOMBRE_DESCRIPTIVO_PROCEDIMIENTO> (
    <PARMETRO>              <DATA_TYPE>,
    <PARMETRO>              IN OUT <DATA_TYPE>,
    <PARMETRO>              OUT <DATA_TYPE>
--------------------------------------------------------------------------------
--DESCRIPCIÓN:    BREVE DESCRIPCIÓN DEL OBJETIVO DEL FUNCION
--PARAMETROS:     DESCRIPCIÓN DE PARÁMETROS CUYA UTILIDAD NO SEA EVIDENTE
--                DE PREFERENCIA ESPECIFICAR VARIALBE DE RETORNO
--------------------------------------------------------------------------------
)
RETURN NUMBER;

END PKC_<PREFIJO_ORIGEN><NOMBRE_DESCRIPTIVO_PACKAGE>;



----------------------------CUERPO DE PAQUETE-----------------------------------

CREATE OR REPLACE PACKAGE BODY PKC_<PREFIJO_ORIGEN><NOMBRE_DESCRIPTIVO_PACKAGE>
--------------------------------------------------------------------------------
--DESCRIPCIÓN:    BREVE DESCRIPCIÓN DEL OBJETIVO DEL PACKAGE
--PARAMETROS:     DESCRIPCIÓN DE PARÁMETROS CUYA UTILIDAD NO SEA EVIDENTE
--------------------------------------------------------------------------------

PROCEDURE PRC_<PREFIJO_ORIGEN><NOMBRE_DESCRIPTIVO_PROCEDIMIENTOS> (
    <PARMETRO>              <DATA_TYPE>,
    <PARMETRO>              IN OUT <DATA_TYPE>,
    <PARMETRO>              OUT <DATA TYPE>
--------------------------------------------------------------------------------
--DESCRIPCIÓN:    BREVE DESCRIPCIÓN DEL OBJETIVO DEL PROCEDIMIENTOS
--PARAMETROS:     DESCRIPCIÓN DE PARÁMETROS CUYA UTILIDAD NO SEA EVIDENTE
--------------------------------------------------------------------------------
)
IS
    <VARIABLE_LOCAL>        <VARCHAR2(TAMAÑO)>,
    <VARIABLE_LOCAL>        <NUMBER(TAMAÑO)>
BEGIN
    <CUERPO DEL PROCEDIMIENTO>
END;

--------------------------------------------------------------------------------
FUNCTION FNC_<PREFIJO_ORIGEN><NOMBRE_DESCRIPTIVO_FUNCION> (
    <PARMETRO>              <DATA_TYPE>,
    <PARMETRO>              IN OUT <DATA_TYPE>,
    <PARMETRO>              OUT <DATA_TYPE>
--------------------------------------------------------------------------------
--DESCRIPCIÓN:    BREVE DESCRIPCIÓN DEL OBJETIVO DEL FUNCION
--PARAMETROS:     DESCRIPCIÓN DE PARÁMETROS CUYA UTILIDAD NO SEA EVIDENTE
--                DE PREFERENCIA ESPECIFICAR VARIABLE DE RETORNO
--------------------------------------------------------------------------------
)
RETURN BOOLEAN
IS
    <VARIABLE_LOCAL>        <VARCHAR2(TAMAÑO)>,
    <VARIABLE_LOCAL>        <NUMBER(TAMAÑO)>
BEGIN
    <CUERPO DE LA FUNCION>
END;

END PKC_<PREFIJO_ORIGEN><NOMBRE_DESCRIPTIVO_PACKAGE>;
```

Cuando se declaran variables dentro de los procedimientos y/o funciones, debe guardarse el orden en la indentación.

✓ **Comentarios del código**

Se incluirán comentarios que clarifiquen el procesamiento. Los comentarios se especificarán con "----". Se añadirán comentarios en el margen derecho de la definición de variables cuya utilidad no sea evidente. Asimismo, se incluirán las líneas de comentario en blanco con el fin de espaciar el comentario de manera tal que respete el indentado del bloque lógico de procesamiento al cual se refiere.

5.1.3 Nomenclatura por tipo objeto

Objeto de base de datos	Prefijo
Tablespace	TBS
Usuarios administrador (esquema)	DBA
Usuario de conexión	USER
Vistas	VI
Índices	IX
Constraint	
Primary key	PK
Foreign key	FK
Unique constraint	UN
Check constraint	CK
Check constraint (not null)	NN
Sinónimos	SI
Secuencias	SE
Paquetes	PKG
Procedimiento	PRC
Función	FNC
Disparador	TRG

6 FIRMAS DE AUTORIZACIÓN

Los abajo firmantes dan su conformidad del contenido del presente documento para todos los efectos del proyecto:

_____ Alfredo Dávila Director General Oficina de Informática	_____ Ricardo Palacios Director Oficina de Informática
_____ Marco Hinojosa Administrador de Base de Datos Oficina de Informática	_____ Jose Robles Administrador de Base de Datos Oficina de Informática

Fecha: _____