**ESTÁNDARES DE DESARROLLO**

**DE BASE DE DATOS**

**Versión 1.0**

# 

Control de Cambios

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Autor | Versión | Referencia del cambio |
| 2018-ENE-16 | patricio.paccha | 2.0 | Creación del documento. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Revisiones

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Fecha | Revisado por | Versión | Observaciones | Sumilla |
|  |  |  |  |  |
|  |  |  |  |  |

Aprobaciones

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Fecha | Revisado por | Versión | Observaciones | Sumilla |
|  |  |  |  |  |
|  |  |  |  |  |

Contents

[1. MARCO GENERAL 5](#_Toc515541060)

[1.1 Generalidades 5](#_Toc515541061)

[2. NOMENCLATURA 6](#_Toc515541062)

[2.1 Regla General 6](#_Toc515541063)

[2.2 Reglas Especificas 6](#_Toc515541064)

[2.2.1 Instancia 6](#_Toc515541065)

[2.2.2 Base de Datos 6](#_Toc515541066)

[2.2.3 Esquema 7](#_Toc515541067)

[2.2.4 Tablas 7](#_Toc515541068)

[2.2.5 Columnas 7](#_Toc515541069)

[2.2.6 Vistas 8](#_Toc515541070)

[2.2.7 Procedimientos almacenados 8](#_Toc515541071)

[2.2.8 Funciones 9](#_Toc515541072)

[2.2.9 Triggers 9](#_Toc515541073)

[2.2.10 Índices 10](#_Toc515541074)

[2.2.11 Sinónimo 10](#_Toc515541075)

[2.2.12 Secuencia 11](#_Toc515541076)

[2.2.13 Primary Key Constraint 11](#_Toc515541077)

[2.2.14 Unique Constraint 11](#_Toc515541078)

[2.2.15 Foreign Keys Constraint 11](#_Toc515541079)

[2.2.16 Defaults constraint 12](#_Toc515541080)

[2.2.17 Check Constraint Constraint 12](#_Toc515541081)

[3. IDENTIFICADORES 13](#_Toc515541082)

[3.1 Parámetros 13](#_Toc515541083)

[3.2 Variables 14](#_Toc515541084)

[4. DOCUMENTACIÓN 15](#_Toc515541085)

[4.1 Encabezado: 15](#_Toc515541086)

[4.2 Nomenclatura de Scripts de bases 16](#_Toc515541087)

[4.3 Documentación de objetos y generación de diccionario de datos 17](#_Toc515541088)

[4.4 Snippet : Log de ejecución de script 19](#_Toc515541089)

[5. RELEASE - DatabaseScripts 20](#_Toc515541090)

[5.1 DatabaseScripts al GIT 20](#_Toc515541091)

[5.2 Resolución de incidentes 20](#_Toc515541092)

[5.3 Desarrollo de nuevas funcionalidades & mejoras 20](#_Toc515541093)

[5.4 Uso de la rama master (De uso exclusivo del área de SQM) 20](#_Toc515541094)

[5.5 Release 21](#_Toc515541095)

[5.6 Gestión de conflictos 21](#_Toc515541096)

[6. BUENAS PRÁCTICAS 23](#_Toc515541097)

[1.1 Variables columnas y campos 23](#_Toc515541098)

[6.1 Claves primarias 23](#_Toc515541099)

[6.2 Palabras reservadas 23](#_Toc515541100)

[6.3 Sangría 23](#_Toc515541101)

[6.4 Uso de espacios en blanco 23](#_Toc515541102)

[6.5 Comentarios de código 23](#_Toc515541103)

[6.6 Diagrama de entidad relación y diccionario de datos 24](#_Toc515541104)

[6.7 Uso de NOCOUNT 24](#_Toc515541105)

[6.8 Uso de sentencias SQL en el cliente 24](#_Toc515541106)

[6.9 Optimización de consultas 24](#_Toc515541107)

[6.10 Uso de Triggers 25](#_Toc515541108)

[6.11 Store Procedures 25](#_Toc515541109)

[6.12 Uso alias en las consultas 25](#_Toc515541110)

[6.13 Prohibición de uso select \* 25](#_Toc515541111)

[6.14 Recomendaciones Adicionales 26](#_Toc515541112)

[7. AUDITORÍA DE LA BASE DE DATOS 27](#_Toc515541113)

[7.1 A Nivel de Servidor 27](#_Toc515541114)

[7.2 A Nivel de Base de Datos 27](#_Toc515541115)

[8. MANEJO DE ERRORES 28](#_Toc515541116)

[8.1 Excepciones en Procedimientos Almacenados 28](#_Toc515541117)

[8.2 Registro de Log de Auditoría 28](#_Toc515541118)

[9. REFERENCIAS 29](#_Toc515541119)

[10. APROBACIÓN DEL DOCUMENTO 29](#_Toc515541120)

ESTÁNDARES PARA EL DESARROLLO DE

BASE DE DATOS DE LA DNTICS

# MARCO GENERAL

El propósito del presente documento es el de definir un patrón de normalización que deberá ser utilizado al momento de crear y actualizar objetos de base de datos (tablas, triggers, procedimientos almacenados, funciones, etc.). De este modo, se fomenta la facilidad de mantenimiento y la estandarización de elementos en la base de datos que se maneja en la institución enmarcada a los lineamientos generales de la Arquitectura General de TICs[1].

Microsoft recomienda el uso de la ISO 8859-1 para evitar los problemas de conversión de caracteres entre los sistemas; además contiene caracteres adecuados para manejar la mayor parte de idiomas que comprende América y la parte Oeste de Europa, incluso haciendo uso, de ser necesario, de tipos Unicode en algunos campos tales como nchar,nvarchar y ntext.[2].

## Generalidades

* Normaliza las tablas para asegurar que no exista duplicidad de datos y se aproveche al máximo el almacenamiento en las tablas.
* Usar campos de tipo bit para almacenar valores booleanos envés de enteros o varchar puesto que consume almacenamiento innecesariamente.
* usar tipos Unicode. Use siempre: nchar, nvarchar y ntext
* Definir filegroup específicos o particiones para entidades con almacenamiento extenso
* Imágenes y columnas de tipo blob no deben estar definidas en tablas frecuentemente requeridas para evitar problemas de rendimiento. Estos datos deben ser definidos en tablas separadas relacionadas por un identificador.

# NOMENCLATURA

## Regla General

* Usar nombres en singular para todos los objetos a crear
* Se debe declarar los nombres en español
* No usar palabras reservadas en los nombres de los objetos, en el caso de ser absolutamente necesario usar [], por ejemplo [Year]
* Los nombres de los objetos de base de datos utilizarán el **<sufijo>** (en mayúsculas) y estarán compuestos por palabras con la primera letra en mayúscula (**UpperCamelCase**). Se debe evitar el uso de abreviaciones que dificulten la comprensión.
* Si se requiere separar el nombrado de los objetos debe usar el underscore: “\_”
* Las palabras reservadas del lenguaje TRANSACT – SQL deben estar en mayúsculas.

## Reglas Especificas

Para la creación de nuevos objetos se debe considerar la nomenclatura de nombres que se describe a continuación:

### Instancia

**Reglas:**

Debe estar relacionado con el aplicativo en mayúsculas.

**Formato:**

<APLICATIVO>

**Ejemplo:**

SATJEU07

### Base de Datos

**Reglas:**

El nombre en mayúsculas y separado por underscore.

El nombre físico de la base de datos -Master Database Files y Log Database Files- debe corresponder al alias de la aplicación seguido de guion bajo más las palabras Data y Log respectivamente.

**Formato:**

BDD:<APLICATIVO>

MDF: <ALIAS>\_Data

LDF: <ALIAS>\_Log

**Ejemplo:**

BDD: EXPEDIENTE\_ELECTRONICO

MDF: EEL\_Data

LDF: EEL\_Log.

### Esquema

**Reglas:**

Nombre en acrónimo o nemónico.

**Formato:**

<ESQUEMA>

**Ejemplo:**

ADM

### Tablas

**Reglas:**

Nombre en singular y **UpperCammelCase**.

Si la tabla es relacional considere que primero es el nombre de la tabla principal.

**Formato:**

<TableName>

**Ejemplo:**

CategoriaJudicatura

### Columnas

**Reglas:**

Nombre en singular, **UpperCammelCase** y no usar espacios, ni guiones.

Para las claves primarias se debe usar el prefijo **Id** unido al <NombreTabla>

No sobrecargar la descripción. Es decir, si estamos en una tabla Provincia y necesitamos la columna nombre; debemos usar el campo nombre y no nombreProvincia.

Las columnas que sirven como referencia como un Foreign Key deberán ser nombradas como esta en la Tabla que hace referencia.

Ajustar al máximo el tamaño de los campos para no desperdiciar espacio.

**Formato:**

<ColumnName>

**Ejemplo:**

IdProvincia 🡪 primary Key

Ubicación 🡪 campo de la tabla Provincia

IdPais 🡪 en la tabla Provincia hace referencia a Id de la tabla País

### Vistas

**Reglas:**

Nombre en singular, **UpperCammelCase** sin espacios iniciando con el prefijo **VS\_**

**Formato:**

VS\_<NombreVista>

**Ejemplo:**

VS\_BuscarJuiciosReasignacion

### Procedimientos almacenados

**Reglas:**

Nombre en singular, **UpperCammelCase** sin espacios iniciando con el prefijo **PA\_** más el **acrónimo del aplicativo (3 letras mayúsculas)** más underscore el ‘nombre del procedimiento almacenado’

El ’nombre del procedimiento almacenado’ debe ser empezar con un verbo en infinitivo.

Los parámetros inician con i (entrada)/ o (salida) junto Nombre en singular, **UpperCammelCase**

**Formato:**

PA\_<ACRONIMO >\_<VerboNombreProcedimiento >

**Ejemplo:**

PA\_EEL\_ **Resortear**Firmante

PA\_EEL\_ **Obtener**Firmante

**Notas:**

Los nombres de los Stored Procedures NO deben comenzar con SP\_, esto porque generalmente el SQL piensa que son system procedures y los busca primero en la base de datos Master

Se recomienda usar procedimientos almacenados cuando sea estrictamente necesario. Se debe desarrollar toda la lógica de negocio en clases **java**, en los paquetes de servicio[3].

No usar prefijo xp\_, para extended stored procedures, cual es un prefijo reservado para identificar system extended stored procedures. No se permiten sentencias SQL en el cliente. Esto genera baja en el performance de los sistemas.

Evitar en lo posible usar cursores, dado que éstos consumen muchos recursos

### Funciones

**Reglas:**

Nombre en singular, **UpperCammelCase** sin espacios iniciando con el prefijo **FU\_** más underscore el ‘NombreFunción’

El ‘NombreFunción’ debe ser empezar con un verbo en infinitivo.

Los parámetros inician con i (entrada)/ o (salida) junto Nombre en singular, **UpperCammelCase**

**Formato:**

FU\_< VerboNombreFunción>

**Ejemplo:**

FU\_ConvertirFirmantePonente

**Notas:**

Se recomienda usar funciones cuando sea estrictamente necesario. Se debe desarrollar toda la lógica de negocio en clases **java**, en los paquetes de servicio[3].

### Triggers

**Reglas:**

Usar el prefijo **TR\_** más **el nombre de la tabla** seguido de guion bajo y a continuación la **acción** (Insert, Delete, Update) y nombre en singular, **UpperCammelCase**

**Formato:**

TR\_<TableName>\_<ActionNombreDescriptivo>

**Ejemplo:**

TR\_Judicatura\_UpdateDeleteAuditado

**Notas:**

<Action> : Debe especificar la acción principal que desarrolla

Insert

Update

Delete

### Índices

**Reglas:**

Usar el prefijo **IX\_** más el nombre de la tabla seguido de guion bajo y a continuación el nombre del campo afectado

**Formato:**

IX\_<TableName>\_<Column>

**Ejemplo:**

IX\_MovimientoJuicio\_IdJudicaturaDestino

**Notas:**

Por cada tabla transaccional importante no exceder más de 12 índices.

Requerimientos de reportes, consultas o inserciones deberá utilizar los índices existentes.

### Sinónimo

**Reglas:**

Iniciará con SYN\_ más el nombre de la tabla o seudónimo más el nombre en singular, **UpperCammelCase**

**Formato:**

SYN\_<TableName>

**Ejemplo:**

SYN\_Provincia

**Nota:**

Se utilizará sinónimos sólo para los objetos de los que no seamos propietarios; es decir a las tablas a las cuales solo se tiene permiso de lectura.

### Secuencia

**Reglas:**

Tendrán como prefijo la palabra “**SEQ**” seguido de underscore más el nombre en singular, **UpperCammelCase**

**Formato:**

SEQ\_<TableName>

**Ejemplo:**

SEQ\_Provincia

### Primary Key Constraint

**Reglas:**

Usar el prefijo PK\_ mas el nombre de la columna.

**Formato:**

PK\_<NombreColumna>

**Ejemplo:**

PK\_IdProvincia

### Unique Constraint

**Reglas:**

Usar el prefijo UK\_ seguido por el nombre del campo. En caso de tener varios campos usar **UpperCammelCase** para denotar cada uno de ellos.

**Formato:**

UK\_<NombreColumnas>

**Ejemplo:**

UK\_LoginCedula

### Foreign Keys Constraint

**Reglas:**

Usar el prefijo FK\_ seguido por el nombre de campo. En caso de tener varios campos usar **UpperCammelCase** para denotar cada uno de ellos.

**Formato:**

FK\_<NombreColumna>

**Ejemplo:**

FK\_IdProvincia

### Defaults constraint

**Reglas:**

Usar el prefijo DF\_ seguido por el nombre de la tabla más underscore con el nombre del campo seguido de guión bajo con el valor a ser asignado.

**Formato:**

DF\_<TableName>\_<ColumnName>\_<Value>

**Ejemplo:**

DF\_Judicatura\_Nombre\_Nulos

### Check Constraint Constraint

**Reglas:**

Usar el prefijo CK\_ seguido por el nombre de la tabla más underscore con el nombre del campo seguido de guión bajo con el valor a ser evaluado.

**Formato:**

CK\_<TableName>\_<ColumnName>\_<Value>

**Ejemplo:**

CK\_MovimientoJuicio\_Cantidad\_NoNegativo

# IDENTIFICADORES

Todos los nombres de identificadores de variables, constantes, cursores y demás, deben ser escritos usando **LowerCammelCase**

**Formato**:

<nombreParámetro>

**Ejemplo:**

cedulaEcuatoriana.

## Parámetros

**Reglas:**

Para un mejor entendimiento de la lógica de los programas, los parámetros deberán llevar la sintaxis **prefijo** + **UpperCammelCase**

**Formato**:

<tipodato><NombreParámetro>

**Ejemplo:**

iCedulaEcuatoriana. 🡪 parámetro de entrada

**Nota:**

|  |  |
| --- | --- |
| **Tipo de Parámetro** | **Identificador** |
| Input/ Entrada | i |
| Output/salida | o |
| Input Output | io |

**Ejemplo:**

CREATE PROCEDURE PA\_EEL\_ResortearFirmante (

@iCedula VARCHAR(10) INTPUT,

@oSecretario INT OUTPUT )

AS

BEGIN

<Declaración de variables>

<Sentencias SQL>

END

## Variables

**Reglas:**

La declaración de variables deberá llevar la sintaxis **lowerCammelCase**

**Formato:**

<nombreParámetro>

**Ejemplo:**

Declaración de una variable para manejar el número de autorización de SRI de tipo numérico.

Declare NumeroAutorizacion integer;

# DOCUMENTACIÓN

* Los scripts deben tener encabezado
* Los procedimientos y funciones deben tener encabezado

## Encabezado:

/\*----------------------------------------------------------------------------------------\

| ©Copyright FUNCION JUDICIAL |

| Todos los Derechos Reservados República del Ecuador |

|-----------------------------------------------------------------------------------------|

| Este código está protegido por las leyes y tratados internacionales de derechos de autor|

| por lo que el uso, reproducción o distribución del código total o parcial no autorizada |

| se atendrá a las sanciones contempladas en la ley con su máximo rigor. |

\-----------------------------------------------------------------------------------------/

[App] : Auditoria-Alert

[Developer]

- Author :

- Descripction :

- validate : [X] Stress o Stress concurrente [ ] ETL

[ ] DataBase Object [X] Log & Audit

[x] Quality Data

[SQM-Tester]

Author :

Date :

Observation :

Tested : [ ] Stress o Stress concurrente [ ] ETL

[ ] DataBase Object [ ] Log & Audit

[ ] Quality Data

[Operaciones]

Operator :

Date :

Observation :

-------------------------------------------------------------------------------------------\*/

--

-- do code

--

## Nomenclatura de Scripts de bases

Para que un script de bases de datos sea válido, debe cumplir con las siguientes recomendaciones:

1. Los scripts de modificación de vistas, procedimientos y funciones se crean en un solo directorio “\_DATABASE” sin numeración y con la siguiente nomenclatura:

**CJ-DNTICS-SDSW-<DATABASE>-<SHEMMA>**-<Vista, función, procedimiento>.sql”

**Ejemplos:**

**CJ-DNTICS-SDSW-BADALEX1-CLEX1-vs\_CN\_SorteoFirmantes.sql**

2. El script debe ser enviado en un archivo con extensión “.sql”

3. Los nombres de los scripts deben tener la siguiente sintaxis:

**Sintaxis:**

##-CJ-DNTICS-SDSW-BaseDeDatos-SHEMMA-ModuloProposito-AÑO-MES-DIA.sql

**Ejemplos:**

05-CJ-DNTICS-SDSW-Badalex-ReportesInternosSPIngresoCausas-2016-JUN-17.sql

06-CJ-DNTICS-SDSW-Adm-ArchivoCrearRolesParaArchivo-2016-AGO-28.sql

07-CJ-DNTICS-SDSW-Adm-TramiteAlterTableJuicios-2016-SEP-15.sql

4. Mantener la secuencia del número de script y las fechas no son relevantes

5. Verificar que el valor de la variable @Filename definida en el script, debe ser idéntico al nombre físico del archivo.

**Ejemplo:**

**exec sp\_AddLogScript '@Filename', '03-CJ-DNTICS-SDSW-Badalex-ArchivoSp\_MotivoSolicutd-2016-SEP-07.sql'**

6. Para almacenar evidencia que el script haya sido ejecutado especificadamente en la creación de objetos de base de datos deben utilizarse las sentencias:

**Ejemplo:**

exec sp\_AddLogScript ‘acción realizada sobre el objeto’

Esto no es necesario dentro de objetos de base de datos como sp, vista y funciones

## Documentación de objetos y generación de diccionario de datos

Cuando en un script se realicen acciones de creación o modificación de tablas o campos, se debe realizar de forma obligatoria la documentación de los objetos con la finalidad de tener actualizado el diccionario de datos; para lo cual se debe cumplir con las siguientes directrices

1. **Declarar las siguientes variables de trabajo**

declare @esquema nvarchar (50)

declare @tabla nvarchar (100)

declare @comentario nvarchar (2000)

declare @columna nvarchar (100)

1. Para documentar una tabla, se debe ejecutar el procedimiento almacenado **sp\_DocumentarTabla**, que tiene la siguiente sintaxis:

exec **dbo.sp\_DocumentarTabla** elEsquema, nombreTabla, elComentario

**Ejemplo 1: Ejecución con variables seteadas**

set @esquema = 'clex1'

set @tabla = 'Flujo'

set @comentario = 'Entidad que representa un flujo procesal, el diagrama en sí del flujo procesal y sus estados básicos.'

exec dbo.sp\_DocumentarTabla @esquema, @tabla, @comentario

**Ejemplo 2: Ejecución de forma directa**

exec dbo.sp\_DocumentarTabla 'clex1', 'Flujo', 'Entidad que representa un flujo procesal, el diagrama en sí del flujo procesal y sus estados básicos.'

1. Para documentar un campo, se debe ejecutar el procedimiento almacenado **sp\_DocumentarColumna**, que tiene la siguiente sintaxis:

exec **dbo.sp\_DocumentarColumna** elEsquema, nombreTabla, nombreCampo, elComentario

**Ejemplo 1: Ejecución con variables seteadas**

set @esquema = 'clex1'

set @tabla = 'Flujo'

set @columna = 'Flujo\_JSON'

set @comentario = 'Cadena de texto que representa el diagrama del flujo procesal. Este campo también almacenará las plantillas de cada actividad en formato base64 (ver utilerías de la bbdd para codificar/decodificar estas plantillas dentro de la bbdd.  Tener cuidado con este campo en consultas pesadas, ya que ocupa tiempo su recuperación.'

exec dbo.sp\_DocumentarColumna @esquema, @tabla, @columna, @comentario

**Ejemplo 2: Ejecución de forma directa**

exec dbo.sp\_DocumentarColumna 'clex1', 'Flujo', 'Flujo\_JSON', 'Cadena de texto que representa el diagrama del flujo procesal. Este campo también almacenará las plantillas de cada actividad en formato base64 (ver utilerías de la bbdd para codificar/decodificar estas plantillas dentro de la bbdd.  Tener cuidado con este campo en consultas pesadas, ya que ocupa tiempo su recuperación.'

1. En el caso de que se necesite consultar la documentación de los objetos (tablas y campos), se debe ejecutar el procedimiento almacenado **sp\_DescribirTabla**, que tiene la siguiente sintaxis:

**exec dbo.sp\_DescribirTabla elEsquema, nombreTabla, [visualizarComentarioTabla]**

**Ejemplo 1: Ejecución con variables seteadas**

set @esquema = 'clex1'

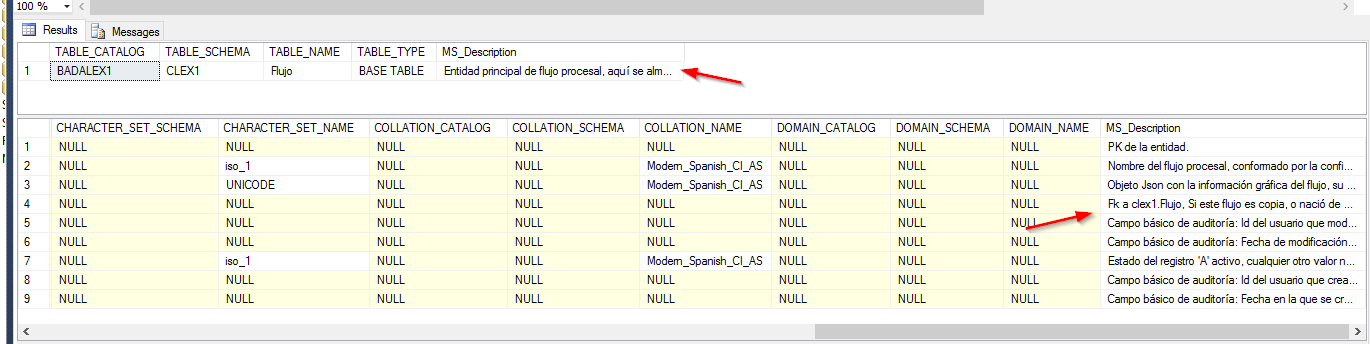
set @tabla = 'Flujo'

exec dbo.sp\_DescribirTabla @esquema, @tabla, 1

**Ejemplo 2: Ejecución de forma directa**

exec dbo.sp\_DescribirTabla 'clex1', 'Flujo', 1

El resultado, será el siguiente:



## Snippet : Log de ejecución de script

**Considerar la que el script tenga log de ejecución, el manejo de errores y transacciones.**

***Ejemplo:***

-------------------------------------------------------------------------------------------\*/

USE [BADALEX1\_ALERT]

GO

EXEC sp\_AddLogScript '@filename', '00 Audit.sql' 🡪 poner nombre del archive para obtener un log de exec.

GO

------------------------------------------------------------------------

-- do code 🡪 procedure, function,...

BEGIN TRAN

BEGIN TRY

-- Body script …

--

COMMIT TRAN

END TRY

BEGIN CATCH

ROLLBACK TRAN

EXEC sp\_AddLogScript 'Error...'

END CATCH

go

-------------------------------------------------------------------------------------------------------

EXEC sp\_AddLogScript 'fin' 🡪 final del script

--[ TestLog: ] ---------------------------------------------------------------------------------------

select \* from LogScript where NameQuery = (select varValue from LogGlobalVar where varName = '@filename' )

-------------------------------------------------------------------------------------------------------

# RELEASE - DatabaseScripts

El objetivo principal es definir el uso del servidor de control de versiones GIT para él envió de scripts de bases de datos hacia el área de SQM. También se describe el uso de GIT para la atención de incidentes y nuevas funcionalidades.

## DatabaseScripts al GIT

Se debe hacer uso del repositorio GIT: **http://xxxxxxx/root/esatje-script** que contiene las siguientes ramas:

* **development**.- Para el **desarrollo de nuevas** **funcionalidades** y **resolución de incidentes** (De uso exclusivo del área de DESARROLLO)
* **master**.- Para el paso a SQM (De uso exclusivo del área de SQM)

## Resolución de incidentes

1. Ingresar al directorio de incidentes creado en el GIT
2. Crear una nueva carpeta con el número de incidente generado en el sistema MANTIS
3. Cree los scripts respetando la nomenclatura
4. Sincronizar su código con el GIT (add, commit, pull & push)

## Desarrollo de nuevas funcionalidades & mejoras

1. Crear una nueva carpeta con el nombre de la funcionalidad
2. Cree los scripts respetando la nomenclatura
3. Sincronizar su código con el GIT (add, commit, pull & push)

## Uso de la rama master (De uso exclusivo del área de SQM)

1. Para enviar a SQM los scripts de incidentes o nuevos desarrollos, se debe realizar un **merge** entre la rama **development** en la rama **master**, con la finalidad de que el área de SQM haga uso de esta rama para descargar los scripts de bases de datos.
2. En el documento de paso a producción en la sección de BDD: especificar la Instalación y configuración con la ruta exacta en la cual están alojados los scripts de bases de datos.

## Release

Para la liberación de los scripts es necesario:

1. Ingrese a: **~\ \_RELEASES**
2. Seleccione o crear el archivo que contiene la liberación: **SQLScriptRelease\_YYYYMMDD.sql**

* Poner su nombre usando “**@user\”**
* Use “**@**” agrega comentario respectivo a tus scripts
* Registra tu script partiendo del directorio raíz del GIT

1. Haz clic en **checkSQLScripts.bat** para validar las rutas de tus scripts
2. Subir los cambios al GIT

**Ejemplo:** selecciona **🡪** SQLScriptRelease\_2018Feb15.sql

**@user\** pepito.perez : DocumentosInviables

**@** primer script …

**\sorteos\DocumentosInviables\01\_CJ-DNTICS-SDSW-CLEX1.DocumentoInviable.sql**

**\sorteos\DocumentosInviables\01\_CJ-DNTICS-SDSW-CLEX1.pa\_getDocumentos.sql**

## Gestión de conflictos

1. **Mantener un solo directorio para los objetos de la base de datos como:**

Vistas

CJ-DNTICS-SDSW-BADALEX1.CLEX1.vs\_litigantes.sql

…

Procedimientos

CJ-DNTICS-SDSW-BADALEX1.CLEX1.pa\_getJuicios.sql

..

Funciones

CJ-DNTICS-SDSW-BADALEX1.CLEX1.fn\_getJuiciosPorLitigante.sql

..

**Nota:**

**Estos scripts no llevan numeración**

1. **Mantener los incidentes en el directorio \_ INCIDENTE crear un directorio con el número de incidente del Matiz y crear los scripts respectivos. Ejemplo:**

**\_INCIDENTE**

<**3242**>

CJ-DNTICS-SDSW-BADALEX1.CLEX1.RecoveryLitigante.sql

…

<**3224**>

< … >

< Reléase >

…

1. **En el caso de nuevos productos o módulos cree su propio directorio.**
2. **Subir los cambios el GIT validar que no existan conflictos.**
3. **Registrar los scripts a liberar en SQLScriptRelease\_YYYYMMDD.sql .**
4. **Validar las rutas de los scripts registrados. Use checkSQLScripts.bat**

# BUENAS PRÁCTICAS

## Variables columnas y campos

usar tipos Unicode. Use siempre: nchar, nvarchar y ntext

## Claves primarias

No se permite el uso de claves primarias compuestas. Se recomienda usar secuenciales.

Elegir columnas con tipos enteros (o sus variantes) para indexar. Una columna con tipo varchar puede causar problemas de rendimiento.

## Palabras reservadas

Todas las palabras reservadas deberán escritas ser en mayúsculas.

## Sangría

Las instrucciones serán alineadas de acuerdo a la profundidad los diferentes bloques que se vayan construyendo. Se puede utilizar alguna herramienta para dar formato al código.

## Uso de espacios en blanco

Siempre incluya un espacio entre cada identificador y separador.

**Ejemplo:**

En lugar de esto:

WHILE (totalSales < maximumSales AND companyType='NEW') LOOP

Escriba esto:

WHILE (totalSales < maximumSales AND companType = 'NEW') LOOP

## Comentarios de código

Se debe tomar en cuenta las siguientes consideraciones para comentar el código TRANSACT SQL.

* No se debe comentar cada línea del código
* Comentar solamente las partes importantes del código.
* Los comentarios deben ser completos y coherentes
* Los comentarios deben ser escritos usando la sintaxis, gramática, puntuación y ortografía correcta.
* No use un comentario para reafirmar las acciones de una pieza de código.

El código SQL podría comentarse adicionalmente en las siguientes instancias:

* Antes de cada estructura de loop
* Antes de cada bloque BEGIN…END
* Antes de cada expresión lógica condicional (IF <condición> THEN).
* Antes de cualquier otra sentencia que se considere significativa e importante

## Diagrama de entidad relación y diccionario de datos

Generar para la BDD un Diagrama ER y un Diccionario de Datos

Script que crean objetos de base de datos deben ser documentados

## Uso de NOCOUNT

En los Store Procedures se debe tener en cuenta que SET NOCOUNT ON elimina la notificación del nro. de registros afectados por cada sentencia SQL lo cual incrementa el performance. Esto exceptúa los procedimientos que, por la naturaleza de su función, requieran esta funcionalidad.

## Uso de sentencias SQL en el cliente

No se deberán utilizar sentencias SQL en el cliente. Esto genera una gran baja en la performance de los sistemas.

## Optimización de consultas

* Se procurará elegir en la cláusula WHERE aquellos campos que formen parte de la clave primaria.
* Si se desea consultar por campos pertenecientes a índices compuestos es mejor utilizar todos los campos que componen el índice.
* Cuando se utilizan varias tablas dentro de la consulta hay que tener cuidado con el orden empleado en la cláusula FROM. Si deseamos saber cuantos alumnos se matricularon en el año 1996 y escribimos: FROM Alumnos, Matriculas WHERE Alumno.IdAlumno = Matriculas.IdAlumno AND Matriculas.Año = 1996 el gestor recorrerá todos los alumnos para buscar sus matriculas y devolver las correspondientes. Si escribimos FROM Matriculas, Alumnos WHERE Matriculas.Año = 1996 AND Matriculas.IdAlumno = Alumnos.IdAlumnos, el gestor filtra las matrículas y después selecciona los alumnos, de esta forma tiene que recorrer menos registros.

## Uso de Triggers

Los triggers deberán ser utilizados solo en caso de que se quiera hacer alguna acción en cascada sobre la integridad referencial o cuando se quiera evitar que una clave primaria sea modificada o si la funcionalidad del aplicativo así lo requiere, tomando en cuenta que el uso de los mismos degrada según sea el caso la velocidad de actualización de las tablas.

## Store Procedures

En la programación de Store Procedures se debe utilizar mayúsculas para las sentencias propias y palabras reservadas del SQL.

## Uso alias en las consultas

* Si se utiliza varias tablas en la consulta siempre se debe especificar a qué tabla pertenece cada campo, con ello se ahorra tiempo al gestor en localizar a la tabla pertenece el campo.

SELECT distinct ofe.ofe\_id,

ofe.pac\_id,

ofe.ofe\_fecha\_desde,

ofe.ofe\_fecha\_hasta,

ofe.ofe\_estado,

ofe.ofe\_version\_registro,

cen.pra\_id

FROM tsga\_oferta **ofe**,

tsga\_oferta\_centro\_ejecucion **oce**,

tsga\_centro\_stu\_ejecucion **cen**

WHERE ( (ofe.ofe\_id = oce.ofe\_id)

AND (cen.cee\_id = oce.cee\_id)

AND (ofe.ofe\_id = en\_id))

En el caso de varias referencias a una misma tabla agregar un número secuencial luego del prefijo de la tabla, esto aplica también para subconsultas o selects anidados.

## Prohibición de uso select \*

No utilizar nunca SELECT \* porque el gestor debe leer primero la estructura de la tabla antes de ejecutar la sentencia

## Recomendaciones Adicionales

Para que un script de bases de datos sea de calidad, debe cumplir con las siguientes recomendaciones:

1. El script debe contener las descripciones del caso al inicio de archivo, detallando si contiene varias acciones (creaciones, modificaciones de objetos de base de datos) o únicamente una descripción antes de la creación o modificación del objeto.
2. Tomar en consideración de que si se crea o modifica un objeto debe cerrarse con GO
3. En general para cambios de estructura, gestión de datos deben realizarse las verificaciones de caso pueden utilizar la cláusula IF EXISTS
4. En cambios de datos utilizar transacciones para asegurar que se cumpla lo indicado, así mismo tener en consideración que si la aplicación controla las transacciones ya no sería necesario en el gestor de base de datos.
5. En inserciones de datos en tablas deben especificar las columnas de las tablas para el mapeo el cual les ayudara a asegurar que los datos se registrar en la columna especificada (mapeo), en las inserciones masivas a partir de una selección deben contar con el nombre de columna correspondiente.
6. En transacción de datos deben asegurar que los datos que estén modificando o eliminando (cambio de estado) sea el correcto recomendado que utilicen la pk para su modificación.
7. Finalmente tener presente que los scripts que se envié deben asegurar la información.

# AUDITORÍA DE LA BASE DE DATOS

SQL Server proporciona varias características útiles para auditar las actividades y los cambios en el sistema de SQL Server. Estas características permiten a los administradores implementar una estrategia de defensa optimizada para los riesgos de seguridad específicos de su entorno.

SQL Server permite crear auditorías a nivel de servidor o a nivel de base de datos. La configuración es sencilla ya que solo se requiere especificar donde se almacenará la auditoría (en un archivo, la bitácora de eventos de aplicaciones o la bitácora de eventos de seguridad), que objeto se desea auditar (Base de datos, Esquema, Tablas, Funciones, Procedimientos Almacenados o Vistas) y para cual Principal (entidades que pueden solicitar recursos de SQL Server). [4]

Se debe tomar en cuenta que el costo de la auditoría es la degradación del rendimiento de servidor de la BDD.

## A Nivel de Servidor

En las especificaciones de auditoría de servidor debemos indicar el objeto de auditoría que deseamos rastrear. Por ejemplo, en el caso de un objeto de auditoría que rastree los login fallidos, debemos crear una especificación que busque los logins que no son exitosos. Para más detalle revisar [4]

## A Nivel de Base de Datos

En este tipo se especifica la clase de objeto se utiliza para indicar lo que se quiere auditar. Las opciones son: Objeto, Base de datos o Esquema. Seleccione objeto para auditar Tablas, Funciones, Procedimientos Almacenados o Vistas. Para más detalle revisar [4]

# MANEJO DE ERRORES

## Excepciones en Procedimientos Almacenados

En caso de requerirlo, los procesos deberán levantar la excepción que será manejada dentro de un proceso principal a nivel e aplicación y no a nivel de base de datos, por lo tanto todos los procesos de consulta, actualización o inserción deben tener manejo de excepciones para que se pueda lanzar una excepción que pueda ser entendida por un usuario o desarrollador.

**Algoritmo de Excepción**

WHEN exception1 THEN

Executable\_statements1;

WHEN exception2 THEN

Executable\_statements2;

. . .

WHEN OTHERS THEN

Otherwise\_code;

END

Para el manejo de excepciones considerar:

BEGIN TRY

-- do something

--

-- RAISERROR ('Error raised in TRY block.', -- Message text.

16, -- Severity.

1 -- State.

); -- Forzar un salto

END TRY

BEGIN CATCH

-- ROLLBACK

-- considerer that you catch of error

-- ERROR\_MESSAGE(),

-- ERROR\_LINE()

-- ERROR\_SEVERITY(),

-- ERROR\_STATE();

-- RAISERROR (@ErrorMessage, -- Message text.

-- @ErrorSeverity, -- Severity.

-- @ErrorState -- State.

-- );

END CATCH

Considerar los niveles de anidamiento para la captura de errores con el respectivo retorno de mensajes del error para la captura o administración del mismo.

## Registro de Log de Auditoría

De ser necesario, se deberá establecer a nivel de proyecto, los registros de auditoría a nivel de tablas, para lo cual se deberá crear un paquete para centralizar estas funcionalidades.

# REFERENCIAS

[2] J. Gabillaud, SQL Server 2008: Administración de una base de datos con SQL Server Management Studio. Ediciones ENI, 2010.

[4] «SQL Server Audit (motor de base de datos)». [En línea]. Disponible en: https://msdn.microsoft.com/es-es/library/cc280386.aspx. [Accedido: 30-mar-2015].

# APROBACIÓN DEL DOCUMENTO

Los firmantes certifican la información presentada en este documento

|  |  |  |
| --- | --- | --- |
| **Elaborado por:** |  | **Revisado por:** |
| Patricio Paccha  **ESPECIALISTA DE DESARROLLO** |  | **SUBGERENTE DE DESARROLLO** |