Estándar de Programación SQL

Con este se busca estandarizar aspectos relevantes de los objetos como el uso de nombres, documentación, seguridad, y rendimiento. La implementación de los estándares en este manual es de seguimiento obligatorio para todas las Bases de Datos existentes.

1. Estructura de tablas.

Los campos de una tabla deben especificarse en orden natural, siendo el primer campo el campo de llave Natural y se considerara como un índice único, se recomienda que en la medida de lo posible este sea auto numérico.

Los campos para almacenar imágenes, binarios y textos largos deben incluirse en una tabla separada y relacionada mediante una relación uno-a-uno.

Implementación:

El orden de aparición de los campos debe ser:

1º. Llave Natural (Índice Único)

2º. Todos los demás campos en orden natural.

2. N□mbre de □bjet□s.

Los nombres de objetos deben ser lo más corto posible, fáciles de leer, y lo más descriptivo posible, evitando términos ambiguos o que se presten a distintas interpretaciones. Ej: ~~tipoMunicipio~~=>CategoriaMunicipio

Además de lo anterior también deben de ser significativos, es decir, que representen bien el propósito de ser del objeto en cuestión. Pueden emplearse abreviaturas o acrónimos pero éstos deben ser regulados para evitar su proliferación indiscriminada. Los nombres deben incluir sólo caracteres del alfabeto español excepto vocales con acento, eñes, y diérisis, y no deben utilizarse caracteres especiales ('#', '/', ';', '%', '+', '-', etc.) ni espacios, el único carácter especial que se permitirá y exclusivamente en los casos que se especificaran posteriormente será el underscore '_'; el uso de números debe evitarse de ser posible. Los caracteres deben

ser en minúscula excepto la primera letra de cada parte constitutiva del nombre (notación CamelCase); esta regla no aplica para abreviaturas de tipos de datos y de tipo de objetos los cuales siempre serán en minúscula; abreviaturas de instituciones o de uso internacional pueden aparecer en mayúscula, por ejemplo: ISO, MIDEPLAN, MAG, etc.

3. Nombres de base de datos:

Deben representar el propósito de la misma y no a los usuarios, departamentos o gerencias. No deben ser necesariamente iguales a los nombres de las aplicaciones informáticas. Ejemplos de nombres válidos: InsumosAgricolas, Inventarios, FinancieroContable, RecursosHumanos, MedicamentosVeterinarios.

4. Nombres de tabla:

Los nombres siempre serán sustantivos en singular. Deben empezar con un acrónimo que permita agrupar de alguna manera lógica o funcional las tablas que estén asociadas, seguidos del underscore '_'. El acrónimo Cat se usará siempre para representar catálogos, y Sys para tablas de uso interno de los sistemas. Ejemplo: Cat_Cliente, Emp_Empleado, Emp_Proyecto, Emp_Agenda, Prd_Producto, Prd_Detalle, Sys_Diccionario, Sys_Parametro, Cat_ClaseProducto.

5. Nombres de campo y variables:

Todos los campos deben iniciar con un acrónimo para el tipo de dato al que pertenecen según la siguiente tabla:

Tipo SQL	**Acrónimo**
Char	c
Varchar	c
Nchar	**no usar este tipo de dato**
Nvarchar	**no usar este tipo de dato**
Text	txt
Ntext	**no usar este tipo de dato**
Binary	bin
Varbinary	vb
Image	img

Datetime	dt
Smalldatetime	dt
Decimal	d
Numeric	**no usar este tipo de dato**
Float	f
Real	r
Int	n
Smallint	n
Tinyint	n
Money	m
Smallmoney	m
Bit	b
Uniqueiden	uid
Timestamp	ts

Los tipos nchar, nvarchar, ntext y Numeric no deben utilizarse.

En los nombres de todos los campos, el acrónimo de tipo debe ir seguido por un underscore '_', esto con el fin de permitir la correcta lectura del tipo de dato de este.

En los campos tipo fecha (datatime) es innecesario incluir la palabra Fecha ya que está implícita en el tipo de dato, por lo tanto, en lugar de usar dt_FechaIngreso basta con indicar dt_Ingreso.

Los campos deben especificar muy claramente que datos representan.

Implementación:

c_NombreEmpleado, b_Cancelado, d_Concentracion, dt_Inscripcion, c_Nota.

6. Nombres de Llaves primarias.

Todas las tablas deben incluir una llave artificial la cual será la llave primaria. Debe usarse el acrónimo "Pk".

Implementación:

PK_Pais, Pk_Provincia.

7. Nombres de Llaves Foráneas.

Siempre que se vaya a indicar una llave foránea en una tabla se debe indicar el acrónimo FK, seguido del carácter '_' y la descripción o nombre del campo.

En el caso de las llaves foráneas que venga de una tabla tipo "Catálogo de Tablas" al acrónimo "FK" hay que concatenarle "Tbl" , quedando de la siguiente manera FkTbl seguido del carácter '_' y la descripción o nombre de la tabla virtual que se referencia, esto con el fin de hacer más descriptivos los campos a utilizar.

Implementación:

Fk_Empleado, Fk_Factura, Fk_Exportador y

FkTbl_TpPuesto, FkTbl_Moneda y

FkTbl_TpInstitucion.

8. Nombres de Relaciones entre las Tablas

Para los nombres de las relaciones se comenzara con Rl, el carácter '_' y seguidamente el nombre de la tabla en que se encuentra la llave primaria a relacionar, seguido del nombre de la tabla donde se encuentra la llave foránea.

Las relaciones siempre serán entre una llave principal natural y el campo o llave foránea, siguiendo siempre lo establecido en los apartados correspondiente a la nomenclatura de estos.

9. Nombres de Procedimiento Almacenado:

Todos los nombres de procedimientos almacenados deben iniciar con el acrónimo usp ('User

Stored Procedure') y siempre debe ir seguido por un underscore '_'. El nombre debe ser un verbo seguido de uno o más sustantivos. Ejemplo: usp_CalcularSalarioBase, usp_ReservarAuto, usp_CancelarCuenta.

10. Funciones

Para la creación de vistas se debe contemplar el prefijo “fn_” seguido del nombre de la función ej. **fn_**CalcularDiasTranscurridos.

Se deben crear funciones cuando se requiere procesar un número considerable de datos y este procesamiento devuelve un solo valor único. Por ejemplo cuando se quiere obtener un porcentaje de desviación.

Como buena práctica se debe evitar lo menos posible utilizar funciones en consultas donde se regrese un cantidad de registros.

11. Nombres de Vista:

Las vistas iniciarán con el acrónimo vw ('view'), siempre debe ir seguido por un underscore ‘_’ y seguirán las mismas convenciones generales para el uso de nombres.

12. Nombres de Triggers:

Los triggers iniciarán con el acrónimo trg, siempre debe ir seguido por un underscore ‘_’, el nombre de la tabla a la que pertenecen, en caso de ser de una sola operación (Insert, Update, Delete) esta debe indicarse en el nombre, seguido del nombre bajo las mismas convenciones generales para el uso de nombres.

13. Nombres de índice:

Para índices se iniciara con IX (léase ix), seguido del nombre del campo o campos involucrados, el carácter '_' se podrá usar para separar el acrónimo ‘IX’ del resto del nombre. Además, cuando exista más de un índice declarado sobre una tabla y un mismo campo, se seguirá la recomendación de MS SQL Server de usar números consecutivos.

14. Uso de Nulos.

El uso del valor nulo debe evitarse a toda costa.

Implementación:

Para indicar nulidad en un campo se recomienda emplear valores explícitos distintos de nulo como por ejemplo: “desconocido”, “no existe”, “no aplica”, “no ingresado” para campos alfanuméricos, o cualquier otro valor no ambiguo. Cualquiera que sea el valor seleccionado debe emplearse consistentemente en TODAS las Bases de Datos.

15. Pr□gramación en las Bases de Dat□s

Todos los accesos a las tablas se harán a través de elementos programáticos (entiéndase trigger, procedimientos almacenados, funciones de cualquier tipo)

Para todas las formas de programación en las bases de datos, siempre se deben acatar las disposiciones de nombres mencionadas en los apartados correspondientes indicados anteriormente.

Toda creación, modificación o borrado de los objetos programáticos deberá estar claramente comentado en el diccionario de datos dentro de la base de datos, en la documentación pertinente, en el mismo código y respetar lo indicado en el estándar de análisis.

Los comentarios en el código siempre seguirán el siguiente estándar:

1º. Al inicio del código se debe indicar:

a. Nombre de quien lo escribió
b. Fecha de escritura
c. Nombre de la persona que lo modifico
d. Fecha de modificación
e. Breve descripción de cada uno de los parámetros de entrada, si los hay
f. Breve descripción de cada uno de los parámetros de salida, si los hay
g. Breve, pero clara y completa, descripción de lo que realiza ese código.
h. En el caso de que algún cambio que se realice, alterase la descripción de alguno de los aspecto anteriormente mencionados, se agregaran líneas indicando como dicha modificación afecta las cosas, pero la descripción original ó anteriores no debe modificarse en ningún momento

2º. Entre más comentarios tenga el código, más clara será la lectura, siempre y cuando no se exceda en la cantidad y longitud de estos.

La identación deberá ser siempre de 2 o 4 espacios. Pero siempre en lo subsiguiente se deberá utilizar la misma identación en todas las bases de datos de la

institución.

Ninguna línea de código será superior a 80 caracteres

16. Buenas prácticas

- Nunca utilizar la sentencia SELECT *. Seleccionar solo aquellos campos que se necesiten ya que cada campo extra genera tiempo extra.
- Escribir las consultas con estructura ANSI y NO con estructuras T-SQL

  Ejemplo estructura ANSI:
  - SELECT <Campo1>, <Campo2> FROM <Nombre Tabla1> INNER JOIN <Nombre Tabla2> ON
  - <PK_TBL1> = <FK_Tbl2>
  - WHERE <Condición>

  Ejemplo estructura T-SQL:
  - SELECT <Campo1>, <Campo2> FROM <Nombre Tabla1>, <Nombre Tabla2> WHERE
  - <PK_Tbl1> = <FK_Tbl2> and <Condición>

- Dado lo anterior, para realizar las relaciones entre las tablas, se deben utilizar las instrucciones:
  - INNER JOIN
  - LEFT JOIN
  - RIGHT JOIN
  - CROSS JOIN

- Las cláusulas SQL principales deben estar en líneas separadas, de modo que las instrucciones sean más fáciles de leer y editar. Por ejemplo:

  SELECT Nombre, ApellidoMaterno
  FROM Cliente
  WHERE ID = 1

- Si se utiliza varias tablas en la consulta, hay que especificar siempre a que tabla pertenece cada campo.
- No utilizar frecuentemente la cláusula LIKE.
- Utilizar en vez de la cláusula IN la cláusula BETWEEN cuando sea posible en la instrucción WHERE.
- Utilizar lo menos posible las cláusulas IN (SELECT), NOT, IS NULL, != , <>, !>, !<, NOT EXISTS, NOT IN, NOT LIKE

- No utilizar tablas temporales públicas. En el caso que se utilicen tablas temporales locales en algún procedimiento, estas siempre se deben eliminar al terminar de utilizarse en el procedimiento en cuestión, de igual modo antes de crear alguna tabla temporal local, siempre se debe ver la opción de utilizar variables tipo tabla como prioridad.
- Utilizar la instrucción TOP si necesita una cantidad limitada de filas.
- Usar la cláusula EXISTS en lugar de la cláusula IN.
- Utilizar la cláusula EXISTS en vez de la sentencia SELECT Count(*) FROM…
- En el caso de que se esté utilizando la cláusula EXIST de este modo:
    - SELECT <Campo1> FROM <Nombre Tabla1>
    - WHERE EXISTS (SELECT * FROM <Nombre Tabla2>)

  Lo óptimo es hacer lo siguiente:
    - SELECT <Campo1> FROM <Nombre Tabla1>
    - WHERE EXISTS (SELECT TOP 1 <Campo1> FROM <Nombre Tabla2>)
- Evitar usar la instrucción UNION, a menos que este eliminando filas duplicadas.
- No utilizar la instrucción ROUND, LOWER, UPPER, SUBSTRING en el WHERE.
- Reemplazar Count(*) por Count(1) o Count(<Nombre del campo>)
- No Utilizar WITH NOLOCK o WITH ROWLOCK porque esta sentencia puede leer la tabla aun cuando tenga transacciones pendientes (update, delete e insert), por ende podría mostrarse y leerse información que puede no sea real.
- Definir variables con el tipo de datos adecuados al dato a almacenar.
- Como buena práctica también incluir en el código la instrucción SET NOCOUNT ON, ya que esta instrucción evita que se devuelva el mensaje que muestra el recuento del número de filas afectadas por una instrucción o un procedimiento almacenado como parte del conjunto de resultados. Si se establece SET NOCOUNT en ON, no se devuelve el recuento. Cuando SET NOCOUNT es OFF, sí se devuelve ese número.
- No utilizar ejecuciones de código DINAMICO, ya que con esta forma no se sabe si la sintaxis es correcta hasta cuando se ejecutan los procesos, El Servidor tiene que compilar y luego ejecutar.
- Utilizar CASE de tal modo de suplir la necesidad de usar código dinámico.
- Evite poner la sentencia SELECT dentro de los campos de selección.
- No use la cláusula INTO nombre_de_tabla (“SELECT… INTO”). Esto bloqueará mientras se ejecuta la consulta las tablas del sistema. En su lugar cree primero las tablas y luego re-escribe la sentencia como INSERT INTO tabla_name SELECT.
- Es recomendable usar joins a un subquery.
- Las columnas filtro dentro de la cláusula WHERE, TIENEN QUE SER del mismo tipo de la columna que existe en la tabla.
- No usar concatenaciones de cadenas en la cláusula WHERE.

1. **SPs:**
    - Un SP solo deberá contener 1 funcionalidad
    - No utilizar Openrowset

2. **Versión de SQL para aplicaci nes nuevas:**
    - Microsoft SQL Server 2008 R2 (SP2) - 10.50.4000.0 (X64)