ESTANDARES DE DESARROLLO EN BASE DE DATOS
Versión 2.0

HISTORIAL DE LAS REVISIONES

Item	Versión	Fecha	Autor	Descripción	Estado	Responsable de Revisión	Responsable de Aprobación
1	1.0	06/09/2022	MB, CP	Versión 1.1	Revisado	MB	MB
2	2.0	02/02/2024	AS,CP,DF	Versión 2.0	Revisado	AS, CP. DF, EV	EV

Autor(es):
MB: Michael Boza
CP: Christiam Polo
AS: Almendra Saavedra

Revisor(es) WIN:
MB: Michael Boza
EV: Edwin Vásquez
CP: Christiam Polo
AS: Almendra Saavedra
DF: Deivis Fernández

Aprobación:
EV: Edwin Vásquez

Contenido

1. **INTRODUCCION** 4
2. **PROPÓSITO** 4
3. **OBJETIVO** 4
4. **ALCANCE** 4
5. **ITEMS CONSIDERADOS PARA ESTANDARIZACIÓN** 4
6. **AUDIENCIA** 5
7. **IMPLEMENTACIÓN** 5
   - **7.1. Nomenclatura de base de datos** 5
   - **7.2. Nomenclatura de esquemas** 6
   - **7.3. Nomenclatura de Objetos** 6
     - **7.3.1. Nomenclatura de las tablas** 6
     - **7.3.2. Nomenclatura de Índices** 8
     - **7.3.3. Nomenclatura de Procedimiento Almacenado:** 9
     - **7.3.4. Nomenclatura de Vista:** 10
     - **7.3.5. Nomenclatura de Triggers:** 10
     - **7.3.6. Nomenclatura de funciones:** 10
     - **7.3.7. Nomenclatura de índice:** 11
   - **7.4. Nomenclatura de linked server** 11
   - **7.5. Nomenclatura de Defaults** 11
8. **Consideraciones y convenciones de programación** 12
   - **8.1. Consideraciones para pases a producción** 12
   - **8.2. Codificación para mejorar la performance** 12
   - **8.3. Convenciones generales de codificación** 13
   - **8.4. Convenciones para nombres de variables** 13
   - **8.5. Manejo de performance índices** 14
   - **8.6. Gestión de transacciones en funciones/procedimientos almacenados** 14
   - **8.7. Normalización de tablas** 14
9. **Buenas prácticas para OLTP (Online Transaction Processing)** 15
10. **Consideraciones para tunning** 16

ESTÁNDARES DE BASE DE DATOS SQL

1. INTRODUCCION

El área de Ingeniería de Datos de la Gerencia de TI de WIN, con el objetivo de transferir las experiencias sucedidas dentro de la implementación de objetos de Base de datos, ha creído necesario documentar los estándares de Base de Datos SQL (Structured Query Language).

2. PROPÓSITO

Esta política establece los estándares para el diseño, implementación y uso de bases de datos. Los estándares de base de datos ayudan a garantizar la calidad, integridad y coherencia de los datos.

3. OBJETIVO

Proponer una forma de definir tanto los nombres de los objetos y/o estructuras necesarias para el desarrollo de un sistema de información a nivel bases de datos, para definir estructuras como funciones, procedimientos y demás objetos asociados.

4. 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. Esta documentación, deberá estar al alcance de todo el personal involucrado en el desarrollo de Sistemas de Información, tanto para arquitectura cliente servidor, web o arquitectura abierta, los cuales dan soporte a las actividades de los procesos de negocio de WIN.

La aplicación del presente estándar, es hacia los objetos nuevos, deberá ser aplicado desde su aprobación y ser comunicado a todo el personal mencionado líneas arriba.

5. ITEMS CONSIDERADOS PARA ESTANDARIZACIÓN

Las nuevas Base de Datos deberán contemplar la versión mínima con soporte. Para el dimensionamiento de nuevas Base de Datos se debe considerar cantidad de transacciones y cantidad de tablas para que Arquitectura Tecnológica nos apoye en obtener el tamaño de Base de Datos.

Este modelo de estándar de base de datos utiliza la nomenclatura "snake case" para los nombres de tablas, campos e índices. La nomenclatura "snake case" es una convención de nomenclatura que utiliza guiones bajos para separar las palabras en un nombre. Esta convención de nomenclatura es popular en el desarrollo de software porque es fácil de leer y recordar.

El modelo también incluye otros estándares para el diseño, implementación y uso de bases de datos. Estos estándares ayudan a garantizar la calidad, integridad y coherencia de los datos. Es importante adaptar este modelo a las necesidades específicas de su organización.

6. AUDIENCIA

Los estándares del presente documento están dirigidos a:

a) Arquitectos de software que requieran consultar los estándares para conocer cómo desarrollar soluciones usando correctamente los objetos de base de Datos SQL.
b) Desarrolladores de soluciones de tecnología de información que requieran implementar soluciones que usen objetos de Base de Datos.
c) Directores y Líderes de proyectos que requieran conocer los estándares de SQL**.**

7. IMPLEMENTACIÓN

7.1. Nomenclatura de base de datos

Prefijo: <empresa>_<objetivo de la BD>_db

Ejemplo: win_erp_db

- Los nombres deben seguir la nomenclatura snake case.
- Los nombres de la base de datos deben ser representativo.

- Modificación de datos
    - Los datos deben modificarse utilizando las herramientas y procedimientos estándar.
    - Los datos deben modificarse siguiendo las reglas de nomenclatura y formato estándar.
    - Cada bloque procedimental desarrollado debe estar documentado, mostrando de manera detallada parámetros, uso, valores retornados, si los hay, histórico de modificaciones indicando autor, fecha y labor desarrollada. La documentación al interior de la construcción procedimental facilitara su entendimiento al momento de realizarse una revisión, en búsqueda de situaciones que se deban corregir. Los comentarios en el código siempre seguirán el siguiente estándar:
        - 1º. Al inicio del código se debe indicar:
            - Nombre de quien lo escribió
            - Fecha de escritura
            - Nombre de la persona que lo modifico
            - Fecha de modificación

        - Breve descripción de cada uno de los parámetros de entrada, si los hay
        - Breve descripción de cada uno de los parámetros de salida, si los hay
        - Breve, pero clara y completa, descripción de lo que realiza ese código.
        - En el caso de que algún cambio que se realice, alterase la descripción de alguno de los aspectos 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.

7.2. Nomenclatura de esquemas

Prefijo: <nombre>

Ejemplo: conta, log, erp, rrhh, crm, ventas.

- Debe ser un nombre representativo.
- En mysql el esquema y base de datos son equivalentes.

7.3. Nomenclatura de Objetos

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. Los nombres de los objetos de datos deben seguir la nomenclatura snake case. Ej: tipo_empresa=>cat_empresa (categoría de empresa).

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 espacial 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.

7.3.1. Nomenclatura de las tablas

Prefijo: <prefijo de esquema>.x_<nombretabla>

Donde x:

t: transaccional

m: maestra

p: paramétrica

Ejemplo: crt.t_plan, crm.m_planilla, gn.t_usuario

- Los nombres siempre serán sustantivos en singular. Deben empezar con el esquema correspondiente, seguido de la letra del tipo de tabla que la identifica, seguidos del underscore '_'
- Para tablas temporales locales el límite máximo es 30 caracteres incluyendo al #, respetando el estándar.
- El prefijo del esquema debe tener máximo 3 caracteres (ct = contabilidad, rh = RRHH, gn = General, tmp = temporal)
- El nombre de tabla debe de ser como máximo 30 caracteres.
- El prefijo de esquema debe ir en minúsculas.
- Las tablas deben tener un nombre descriptivo en minúsculas y que utilice la nomenclatura "snake case".
- Las tablas deben tener una clave principal que sea única y no nula.
- Las tablas deben tener una clave foránea para cada relación.

7.3.1.1. Nomenclatura de Columnas

Prefijo: <prefijo del uso del campo>_<nombre campo>

Ejemplos: id_entidad, desc_entidad, cod_producto, desc_producto, fec_crea, flg_activo, desc_razon_social

- Las columnas deben tener un nombre descriptivo que comience con una letra minúscula y utilice la nomenclatura "snake case".
- Las columnas de una tabla deben especificarse en orden natural, siendo el primer campo el campo de llave primaria, se recomienda que en la medida de lo posible este sea auto numérico.
- Las columnas deben tener un tipo de datos adecuado para el valor que almacenan.
- Las columnas deben tener una longitud adecuada para el valor que almacenan.
- Uso de Nulos, debe evitarse en lo posible.
- Las columnas flag deben ser de tipo bit.
- Los tipos de datos deben ser apropiados para el tipo de datos que representan. Deben ser consistentes con otros objetos de datos relacionados.
- El tamaño de los datos debe ser suficiente para almacenar los datos necesarios.
- El tamaño de los datos debe ser lo más pequeño posible para optimizar el rendimiento.
- El prefijo debe ser en minúsculas, se usarán los siguientes acrónimos:
  - ✓ id: Para identificadores numéricos.
  - ✓ cod: Para códigos.

- ✓ mnt/imp: Para valores numéricos con decimales.
- ✓ cant: Para cantidades.
- ✓ cat: Para categorías.
- ✓ num: Para números (enteros).
- ✓ fec: Para fechas (date y datetime)
- ✓ hrs: Para horas.
- ✓ desc: Para nombres y descripciones.
- ✓ flg: Para campos lógicos (flag)
- ✓ tip: Para tipo.

7.3.1.2. Nomenclatura de llaves primarias

Prefijo: pk_<tabla>

Ejemplo: pk_gn_t_usuario = llave primaria id_usuario

- Debe usarse el acrónimo “pk” para dicho campo antecedido por el tipo de dato y de un undercore ‘_’.
- Todo constraint debe tener un nombre de acuerdo al estándar especificado.
- Sin embargo. Para el caso de DEFAULTS y CHECKS triviales, la nominación de los mismos es opcional. Se recomienda la definición directa a nivel de diseño. Por ejemplo, DEFAULTS (df_tabla_campo), CHECKS (ck_tabla_campo)
- No puede incluir al #.
- El constraint primary key genera un índice clustered, unique con el mismo nombre.

7.3.1.3. Nomenclatura de llaves foráneas

Prefijo: fk_<tab. refer.>_<tab. origen>

Ejemplo: fk_gn_t_usuario_gn_t_grupo

- Siempre que se vaya a indicar una llave foránea en una tabla se debe indicar poniendo primero el acrónimo de tipo de campo, el carácter ‘_’, el acrónimo fk, seguido del carácter ‘_’ y la descripción o nombre del campo.
- El Constraint Foreign Key Genera un índice nonclustered, con el mismo nombre.
- La Tabla Referencia es aquella cuyos campos deben existir en la tabla origen. (existe una relación de referencia)

7.3.2. Nomenclatura de Índices

Prefijo: ix_<nombre tabla>_<definición indice>

Ejemplo: ix_gnt_usuario_nombre

- Los nombres de los índices deben ser adjetivos.
- Los índices deben utilizarse para mejorar el rendimiento de las consultas.

- Los índices deben utilizarse para optimizar las relaciones entre tablas.
- Usar esta nomenclatura para índices que no dependen de un constraint.
- Evitar usar excesivos INDICES, en lo posible reusar los que se tengan. Muchos índices degradan la Inserción y actualización de información.

7.3.3. Nomenclatura de Procedimiento Almacenado:

Prefijo: spx_<nombretabla>_<nombreopcional>

X puede ser:
S = SP Select (por llave completa). Debe traer un solo registro.
I = SP Insert
U = SP Update
D = SP Delete
P = SP Proceso. Afecta a varias tablas
A = SP Selección total de registros, sin ninguna condición
L = SP Listado (selección por llaves compuestas).
C = SP Selección para utilizar en controles de listado (combos)
T = SP para tablas recursivas (Tree)
H = SP para tablas recursivas en controles de listado.
R = SP Reportes

Ejemplo:

spi_gn_t_usuario : sp insertar usuarios
spc_gn_t_tpidentidad: sp lista de documentos de identidad para combos
spl_gn_t_usuariosgrupo: sp lista de usuarios por grupo
spp_rh_t_planillas_calculo: sp para el cálculo de planillas

- Todos los nombres de procedimientos almacenados deben iniciar con el acrónimo spx ('Stored Procedure X') y siempre debe ir seguido por un underscore '_'. El nombre debe ser un verbo seguido de uno o más sustantivos. Ejemplo: spp_calcular_salario_base.
- Los parámetros pasados a los STORED PROCEDURE deben ser descritos cuando sus funciones no sean obvias y cuando la rutina espera que el parámetro esté en un rango especifico de valores. El valor de retorno del STORED PROCEDURE, los parámetros pasados por referencia (OUTPUT), deben también ser descritos al inicio de cada STORED PROCEDURE.
- La longitud máxima del Nombre Opcional, en algún caso extremo, será de 20 caracteres.
- El Nombre opcional se utilizará para aquellos procedimientos que realicen un trabajo diferenciado.
- No se recomienda el uso del prefijo "sp_" para nombrar un stored procedure, debido a que este se utiliza para los de sistema y que SQL Server los coloca en un caché y al no encontrarlos procede a recompilarlos.
- Todo procedimiento almacenado deberá ser documentado con la siguiente estructura.

```
/*****************************************************************'*
*Nombre SP  : <Nombre del SP>
* Propósito  : Explicar en forma detallada
* Input      : <Parámetro> - Descripción de los parámetros
* Output     : <Descripción de la Salida>
* Creado por : <Responsable>
* Fec Creación: <Fecha Creación>
```

* Fec Actualización: <Fecha de Actualización>
*Actualizado por:<Responsable de la Actualización>
*Ticket/Proyecto: <Número de ticket o proyecto al que hace referencia>
* Incluir detalle de cambios
**********/

7.3.4. Nomenclatura de Vista:

Prefijo: <prefijo del esquema>_v_<nombrevista>

v: vista

Ejemplo: vt_v_clientes_activos

- Las vistas iniciarán con el prefijo del esquema, seguido del acrónimo v ('view') y el nombre de la vista, siempre debe ir seguido por un underscore '_' y seguirán las mismas convenciones generales para el uso de nombres.
- En el caso que la vista sea sobre una única tabla, se adopta el nombre de la tabla, en caso contrario se utiliza los criterios para nombrar tablas.

7.3.5. Nomenclatura de Triggers:

Prefijo: trg_x_<nombretabla>_<nombreopcional>

Donde x puede ser: i(insert), u(update), d (delete), combinación de algunos de ellos.

Ejemplo: trg_i_gnt_usuario, trg_i_ud_ctt_voucher_actualiza_saldos.

- Los triggers iniciarán con el acrónimo trg, siempre debe ir seguido por un underscore '_', seguido de la inicial de la función que realizará, 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.
- La longitud máxima del Nombre del opcional, en algún caso extremo, será de 20 caracteres.

7.3.6. Nomenclatura de funciones:

Prefijo: <nombreesquema>.<prefijo de función>_<nombre de función>

Ejemplo:

crm.fn_proper

crm.ft_ctsaldos

- El prefijo debe ser de 2 caracteres:
  - fn: Función escalar, es una función que te devuelve un valor.
  - ft: Función tablas, es una función que te devuelve una tabla.
- La longitud máxima del nombre de la función, en algún caso extremo, será de 20 caracteres.
- Toda Función deberá ser documentada con la siguiente estructura.

```
/****************************************************************** '*
* Nombre FUN    : <Nombre de la Función>
* Propósito     : Explicar en forma detallada
* Input     : Descripción de los parámetros
* Output  : <Descripción de la Salida>
* Creado por    : <Responsable>
* Fec Creación  : <Fecha Creación>
* Fec Actualización : <Fecha de Actualización>
*Actualizado por: <Responsable de la actualización>
*Ticket/Proyecto: <Número de ticket o proyecto al que hace referencia>
* Incluir detalle de cambios
'*******************************************************************/
```

7.3.7. Nomenclatura de índice:

Prefijo: ix_<nombretabla>_<definiciónindice>

Ejemplo: ix_gn_t_usuario_nombre

- Para índices se iniciará con el acrónimo ix, seguido del nombre de la tabla, el carácter '_' se podrá usar para separar el acrónimo del resto del nombre. Además, cuando exista más de un índice declarado sobre una tabla y un mismo campo, se seguirá la recomendación de usar números consecutivos.
- Usar esta nomenclatura para índices que no dependen de un constraint.

7.4. Nomenclatura de linked server

Prefijo: <acrónimodelmotor>_<descripciónbasededatos>

Ejemplo: MYSQL_MIPORTAL, SQLSRV_CONCAR

Solo para la creación de linked server se admite el uso de mayúsculas. El área de Infraestructura es responsable de crear los linked servers. Toda creación de linked servers debe ser aprobada a través del sistema de solicitudes de accesos(tickets) y ejecutada por el área de Infraestructura.

7.5. Nomenclatura de Defaults

Prefijo: d_nombre_default

Ejemplo: d_importe

- Usar DEFAULT solo para tipos de datos definidos por el usuario, en caso de columnas usar DEFAULT (Declarativo); es decir utilizando únicamente la expresión.

  Ejemplo dt_usuariocrea DEFAULT SYSTEM_USER

8. Consideraciones y convenciones de programación

8.1. Consideraciones para pases a producción

- Todo pase a producción debe cumplir con los estándares fijados en el presente documento. Previo a la validación y aprobación respectiva del equipo de Ingeniería de Datos.
- Todos los aplicativos deben estar documentados (contar con un diccionario de datos y diagramas de procesos).
- La creación de cualquier objeto (procedures, funciones, tablas, índices, etc.) debe contener el esquema definido en el presente documento, de no contar con lo indicado, no será válido para su aprobación.
- La creación de cualquier objeto (procedures, funciones) debe contener un historial de versiones con el formato indicado líneas arriba, así mismo contar de manera obligatoria con los campos de auditoria señalados los cuales deben ser actualizables, de no ser el caso no se aprobará la creación de los objetos.
- Incluir campos de auditoría son valores por defectos y deben ir incluidos de manera obligatoria en la creación de tablas:

Campo	**Tipo de datos**
desc_usuario_crea	Varchar(50)
desc_host_crea	Varchar(50)
desc_host_mdf	Varchar(50)
desc_usuario_modf	Varchar(50)
fec_creacion	Datetime
fec_modf	Datetime
nom_app	Varchar(50)
nom_app_modf	Varchar(50)

- Es MUY importante que cada pase contenga su procedimiento y script de ROLLBACK, el cual debe considerar:
    - Script de rollback para las operaciones DMLs (Inserts, Updates, Deletes). Ejemplo: Si el proceso implica operaciones de Insert, el rollback será el Delete de los registros insertados. Asimismo, tomar en cuenta el recálculo de valores autonuméricos, contadores, acumuladores, etc.
    - Script de rollback para los objetos nuevos (tablas, índices, procedures, etc).
    - Para escenarios con gran cantidad de transacciones, que implican muchos objetos, el rollback sería la restauración de la base de datos obtenida previamente al inicio del proceso.
    - En la medida de lo posible, se debe preparar estructuras de migración que faciliten la creación de scripts que puedan funcionar en forma automática y, asimismo, establecer el orden de ejecución de los mismos.

8.2. Codificación para mejorar la performance

- Se usarán procedimientos almacenados, funciones definidas por el usuario. **Evitar el uso de scripts SQL desde la aplicación**.

- Cuando se codifiquen transacciones grandes se debe usar savepoints (SAVE TRANSACTION) en los lugares adecuados para evitar rollbacks costosos.

8.3. Convenciones generales de codificación

- Todas las rutinas (STORED PROCEDURE, TRIGGER, VIEW, FUNCIONES) deben empezar con un comentario corto describiendo las características funcionales de la rutina (Qué es lo que hace). Este comentario no debe describir los detalles de la implementación de la rutina (Cómo lo hace) porque la implementación puede cambiar con el tiempo, resultando que tenemos un trabajo innecesario de mantenimiento del comentario o peor aún que tenemos comentarios erróneos. El código por sí mismo o cualquier comentario en línea describirá la implementación de la rutina. Debe contener también el nombre del autor, tal como se indicó líneas arriba para cada tipo de rutina.
- El uso de los SERVIDORES VINCULADOS se recomienda, siempre que se analice la necesidad y con la aprobación del área de Infraestructura.
- Todos los comentarios de más de una línea deben hacerse con los caracteres "/* */"de la siguiente manera:

```
CREATE PROCEDURE si_gnt_usuario
…
/*
      Desarrollado por el generador.
*/
…
```

- Todos los comentarios en línea deben ser escritos con los caracteres "--", de la siguiente manera:

```
DECLARE
@id_usuario int,        -- id Usuario
@ds_usuario varchar(100),   -- Nombre Usuario

SELECT
        @desc_usuario = desc_usuario
FROM crm.gn_t_usuario
WHERE id_usuario = @id_usuario and flg_activo =1
```

- Los nombres deben ser lo suficientemente largos para autodocumentar su función.
- Para reflejar la estructura lógica del código, debe usarse el carácter TAB. NO usar espacios. En el entorno de edición, el TAB debe configurarse como ocho caracteres a fin de tener una vista más ordenada del script.
- Las sentencias SQL que hacen referencia a varias tablas deben utilizar alias pasa identificar a cada una de ellas.

8.4. Convenciones para nombres de variables

- Los nombres de variable deben ser usar la nomenclatura snake case, en la medida de lo posible, equivalentes los nombres de columna correspondientes.
  Ejemplo:
  @id_cliente, @cod_producto, @desc_ape_pat

- Para el caso de contadores utilizar @i, @j, @k, @n
- En la medida de lo posible no usar abreviaturas para variables no triviales, pues dificultan la lectura del código. En casos donde sea conveniente el uso de abreviaturas, estas deben respetar los estándares del proyecto.
  Ejemplo:
  @desc_nom_ven_real --Nombre del vendedor real.

8.5. Manejo de performance índices

- Se deberá indizar las columnas que sean estrictamente necesarias.
- No crear índices en los siguientes casos:
  - Columnas que son raramente referenciadas en una consulta.
  - Columnas que tienen pocos valores únicos.
  - Columnas varchar (max)
- Utilizar índices INCLUDE para campos que no sean parte de la ordenación, pero formen parte de la selección de datos. Mediante esta práctica una consulta agrupada, por ejemplo, requiere únicamente la información almacenada en el índice y no tiene que realizar “saltos” a la tabla.

8.6. Gestión de transacciones en funciones/procedimientos almacenados

- Las transacciones (BEGIN TRANSACTION… COMMIT TRANSACTION … ROLLBACK TRANSACTION) deben gestionarse desde la aplicación; salvo aquellas que implican procesos, como, por ejemplo: cálculo de planillas, cierres contables, etc.
- El no incluir esta gestión de transacciones, posibilita que data residual se almacene aun cuando el proceso haya sido cancelado.
- Tener en cuenta, que, para el manejo de transacciones desde las aplicaciones cliente, los mensajes enviados al usuario, los procesos de emisión, etc., deben ejecutarse posteriormente al cierre de las mismas.

8.7. Normalización de tablas

Para aquellas tablas en las que se incluya datos de gran volumen (text, image, etc). Éstas deben estar separadas en 2 partes, una con los datos estándar y otra con los datos de gran volumen. Por ejemplo, separar el maestro de empleados de sus fotografías.

Para tablas que incluyen imágenes como fondos, logos, íconos, etc. éstas deben almacenarse en tablas de imagen y los maestros que las requieran, utilizarán campos de referencia.

Asimismo, un criterio para este tipo de partición de tablas es la normalización, para evitar la repetición de datos. Toda creación de nueva tabla debe indicar con que tabla se va a relacionar.

Para tablas en las que se utilicen campos varchar(max) para el ingreso de información variada, se recomienda utilizar una tabla que permita el ingreso de detalles ordenado y no utilizar un solo registro de gran tamaño.

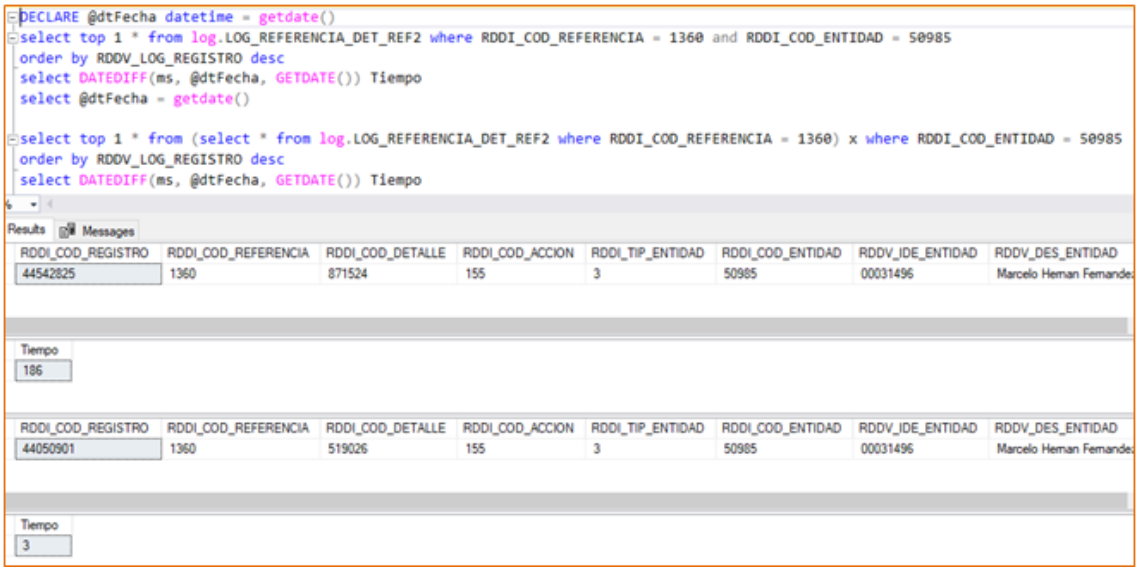
De esta manera evitamos este tipo de registros:

Para tablas con gran volumen de información, se recomienda el particionamiento de las mismas utilizando campos de tipo fecha o identificadores de período.

9. Buenas prácticas para OLTP (Online Transaction Processing)

- ✓ Evitar el uso de asterisco (*) en con la cláusula SELECT.
- ✓ Todo trigger debe tener control de errores. Ante una falla debe realizarse el rollback de la transacción.
- ✓ Usar UNION ALL en lugar de UNION para evitar ordenamiento innecesario y pérdida de datos.
- ✓ Evitar las conversiones de tipos de datos innecesarios, sobre todo en los filtros.
- ✓ Evitar usar usuarios y esquemas con roles de DBA o tener más privilegios de lo que se requieren.
- ✓ Usar tablas temporales en lugar de tablas físicas "temporales", para reportes, estadísticas, etc.
- ✓ Usar Tablas Temporales o variables tipo tabla para reescribir subqueries complejos.
- ✓ Evitar el uso de LIKE en el predicado. A menos que exista un índice por ese campo. Utilizar "XXX%"
- ✓ Evitar innecesarios FULL TABLE SCAN en tablas grandes.
- ✓ En el caso del uso de columnas identity, se recomienda utilizar la cláusula OUPUT para obtener el valor del (los) registro(s) insertado(s). Evitar el uso de @@IDENTITY, IDENT_CURRENT(),
- ✓ Anidar una subconsulta para mejorar el rendimiento.
- ✓ En la imagen se puede evidenciar que para obtener un registro aplicando un filtro compuesto y la misma operación mediante una consulta anidada el rendimiento de esta última es más eficiente que la anterior.
- ✓ Los scripts de actualización (INSERT, UPDATE, DELETE) no deberían incluir más de 100 instrucciones individuales. De requerir operaciones masivas se

deberá utilizar procedimientos BULK, para evitar encolamiento en las bases de datos productivas.



- ✓ Evitar la generación de Producto Cartesiano.
- ✓ Evitar comparar un campo a null en la cláusula WHERE.
- ✓ Evitar el uso de cursores. Es más eficiente recorrer una tabla temporal o una variable tabla y en la mayoría de los casos, se podría realizar los procesos masivamente y no uno a uno.
- ✓ Usar la sentencia Count con un Valor no con asterisco (*)
  Ejemplo: Select Count(1)
- ✓ No se debe actualizar los campos que son llaves primarias en una tabla.

10. Consideraciones para tunning

a) Mantener actualizadas las estadísticas e índices.
b) Revisar los planes de ejecución.
c) Evitar el uso de funciones definidas por el usuario en los filtros. En esos casos, utilizar variables con los valores de dichas funciones previamente a la ejecución del filtro.
d) SQL Profiler nos permite revisar las trazas (procesos) en ejecución.
e) Una buena práctica es el uso de tablas de log para realizar el seguimiento de un proceso. Estas tablas deben tener una estructura que contenga un identificador autonumérico, un campo para el detalle y otro para la hora de ejecución. Este último nos ayudará, asimismo, a identificar los puntos en los que nuestros procesos son más lentos.
f) Utilizar scripts para la detección de bloqueos.
g) Almacenar la información de bloqueos para el análisis posterior correspondiente.
h) Los campos de auditoria deben ser obligatorios, y deben ser actualizados con los datos de fecha de modificación y usuario de modificación.