En esta publicación agrupo por categoría y menciono buenas practicas para el diseño de bases de datos relacionales.

>>***A NIVEL DE BASE DE DATOS***

-El charset de las bases de datos son ‘UTF-8′, en MS SQL SERVER=‘SQL_Latin1_General_CP1_CI_AI’, en ORACLE=’AL32UTF8’ o ’AL32UTF16’.

-El motor por defecto debe de ser relacional (INNO).

-Nombre de la base de datos en MAYÚSCULA ej ‘MIBASEDEDATOS’.

>>***A NIVEL DE TABLAS***

-El nombre de las tablas debe ser en mayúsculas o CamelCase.
-El nombre de las tablas debe ser en singular.

-El nombre de las tablas deben ser descriptivos, no importa que tan largos sean siempre y cuando sean soportados por la base de datos.

-Si la tabla es un catálogo puedes usar la puedes nombrar como ‘CATPRODUCTO’, ‘CAT_CLIENTE’ o en CamelCase ‘CatCliente’. Solo no abuses del uso de prefijos o sufijos, en algunas ocaciones pueden ser innecesarios.

-Si la tabla tiene más de 2 palabras estas se deben poner juntas o con un guión bajo, nunca se debe de usar espacios, ej ‘APELLIDO_PATERNO’ o en CamelCase ‘APaterno’.

-Si la tabla es una tabla muchos a muchos (multivaluada), se deben de utilizar los nombres de las tablas que generan la relación, deberán de ir con guiones bajos, ej ‘VENDEDOR_CLIENTE’, ‘VENDEDOR_CLIENTE_PRODUCTO’ otro ejemplo ‘CONTACTO_TELEFONO_TABLA1_TABLA2_TABLAN’, el guion nos ayuda a identificar las relaciones.

-Evita el uso de tablas temporales, mejor utiliza alguna estructura de datos del lenguaje de programación que estés usando.

>>A NIVEL DE CAMPOS

-Todas las tablas deben de tener un Primary Key (PK) y debe de ser el primer campo de la tabla y debe de ser único e irrepetible.

-Ademas del PK se sugiere que todas las tablas tengan los siguientes campos: Estatus, Fecha Alta, Fecha Ultima Modificación (FECHA_MOD), Fecha Baja, Usuario Alta, Usuario Modificación y Usuario Baja.

-El nombre de los campos/columnas deben ser en singular.

-PK o llave primaria de todas las tablas debe de ser un entero (INT o BIGINT), es único, autoincremental, indexado y dependiendo del uso de la tabla, será ascendente o descendente, si es una tabla que genera muchos registros lo mejor sería descendente.

-El nombre del PK empieza con ‘ID’ en mayúsculas guión bajo nombre de la tabla con mayúsculas, ej ‘ID_NOMBRE_DE_LA_TABLA’, ‘ID_APELLIDO_PATERNO’, ‘ID_CONTACTO_TELEFONO_TABLA1_TABLA2′.

-Las llaves foráneas (foreign keys) (FK), deben ser un entero (INT o BIGINT), indexadas, y con la misma nomenclatura que un PK pero agregando ‘FK_’ al inicio, ej ‘FK_ID_NOMBRE_DE_LA_TABLA’, ‘FK_ID_APELLIDO_PATERNO’, ‘ID_CONTACTO_TELEFONO_TABLA1_TABLA2′.

-El nombre de los campos deben de ser como métodos en JAVA(CamelCase), empiezan en minúsculas, no tienen espacios o guiones bajos, son descriptivos, y las siguientes palabras empiezan con mayúscula, ej ‘holaMundo’, ‘apellidoPaterno’, sin embargo también pueden ir en mayúscula usando un guión bajo como espacio, la única condición es que todos sean homogéneos.

-Evita usar el ‘NULL’ es decir campos en nulo, todos los campos deben de tener una inicialización, esta puede ser vacía y en caso de que tenga otra, debe de estar comentada en la tabla.

-Todos los campos de ESTATUS deben de estar comentados con valores, tratar de usar strings de por lo menos 10 caracteres e indexados.

-En ocasiones es muy bueno indexar algún campo de fecha.

>>A NIVEL DE TIPOS DE DATOS PARA SQL SERVER

-Para cadenas de caracteres se utilizará nvarchar.

-Para fechas se utilizará nvarchar(10) y con el formato aaaa-mm-dd

-Para horas se utilizará nvarchar(5) y con el formato HH:mm

-Estampa de tiempo simple nvarchar(8) y con el formato HH:mm:ss

-Bigint para estampa de tiempo completa (long en java).

-Para tipo de datos booleanos es bit (La sugerencia es no usarlo a menos que el dato a guardar indique que no tendrá cambios, ej sexo).

-Para textos largos se usará “nvarchar(MAX)” el cual equivale a 2 GB de información en SQL SERVER y en ORACLE nvarchar2(1024) esto quiere decir que puede almacenar 1024 caracteres de UTF8, en ORACLE 12c se puede ampliar a 32767 bytes = 8190 Characters.

-Paro todos los datos de tipo decimal se usará float con precisión habilitada o la precisión que viene por defecto.

-Para cualquier cosa menor al 127, se utiliza tinyint, si en un futuro se requiere, se puede cambiar por el entero completo o bigint.

-Evita guardar los archivos en la base de datos, trata de guardar solo las rutas, en caso extraordinario de requerir un binario, el tipo de dato seria varbinary(max).

>> MAPEO DE TIPO DE DATOS DE MySQL EN JAVA
>> MAPEO DE TIPO DE DATOS DE MS SQL SERVER EN JAVA
>> MAPEO DE TIPO DE DATOS DE ORACLE EN JAVA
>> EQUIVALENCIAS DE TIPOS DE DATOS ENTRE MS SQL SERVER Y ORACLE

>>*A NIVEL DE TRANSACCIONES*

-El uso de transacciones es obligatorio cuando se inserta, actualiza o se elimina más de una tabla.

-Trata de usar niveles de bloqueo (isolation level) optimistas para reducir el tiempo de bloqueo de la tabla, puede ser ‘Read uncommitted’ o ‘Read committed’.

>>*A NIVEL DE RELACIONES*

-Todas las FK deben de tener relación con restricciones de updates en cascada y borrado restringido.

-Todas las llaves foráneas deben de ser del mismo tipo de dato y longitud que la llave primaria a la que hace referencia.

-Todas las llaves foráneas deben de ir indexadas en las tablas donde son foráneas, de preferencia con indices descendentes.

-Se deberá de evitar la redundancia cíclica, esto sucede cuando se tiene una actualización en cascada desde 2 o más tablas padre a un campo de la tabla hija, en este caso solo se debe de mantener una sola actualización en cascada o su defecto ninguna.

>>*A NIVEL DE PROCEDIMIENTOS ALMACENADOS*

-Evita (trata de no usar) el uso de procedimientos.

>>*A NIVEL DE FUNCIONES*

-Evitar (tratar de no usar) el uso de funciones (las desarrolladas por el usuario).

>>*A NIVEL DE TRIGGGERS (gatillos o disparadores)*

-Evitar (tratar de no usar) el uso de Triggers.

>>*A NIVEL DE VISTAS*

-Evitar (tratar de no usar) el uso de vistas.

>> *BUENAS PRÁCTICAS PARA RENDIMIENTO, SEGURIDAD Y FUNCIONALIDAD*

-Gasta todo el tiempo que se requiera para el proceso de análisis y diseño de base de datos. Es infinitamente más fácil hacer cambios cuando aún no hay datos y/o código . El objetivo es minimizar los cambios al diseño de la base.

-Separa toda la lógica de negocio de la base. La base de datos es eso, un lugar en donde se almacenan y se consultan datos. Usa un lenguaje de programación para ejecutar las reglas de negocio.

-Trata de normalizar tu base de datos a una forma normal 5, de no ser posible por lo menos llega a una forma nivel 3. En el siguiente post explico LAS REGLAS DE NORMALIZACIÓN EXPLICADAS FACILMENTE.

>>*SEGURIDAD*

-Para conectar un sistema, crea un **"usuario funcional"** que solo tenga privilegios de SELECT, INSERT, UPDATE, DELETE, SESSION para un esquema que va a utilizar, QUE NO TENGA darle privilegios de DROP y/o GRANT.
-Limita el privilegio al mínimo necesario para acceso y funciones a los usuarios de base de datos.

-Lleva un control de versiones de la base de datos.

-Para todo cambio en la base, también se debe de actualizar el diagrama de la base de datos.

-Bloquea el acceso a la base de datos a través del firewall.

-Escanea periodicamente la red.

-Limita el acceso fisico y logico a la red y al servidor de base de datos.

-Evita realizar tareas de administración remotas, si no es posible evitarlas utiliza una VPN.

-NUNCA expongas una base de datos a Internet.

-Deshabilitar el usuario súper administrador, sa, root o toor.

-Cambia los passwords que vienen por defecto.

-Habilitar la encriptación (SSL) de las base de datos y de la conexión. O en su defecto usa un túnel SSH.

-Para almacenar los password de los usuarios utiliza métodos de hasheo a nivel aplicación(vía lenguaje de programación) en vez de los algoritmos que proporciona la base de datos.

-Revisar periódicamente el espacio en disco y los logs de la base.

-Usa herramientas de monitoreo de consultas (revisar deadlocks) y log de conexiones a la base de datos.

-Antes de realizar un cambio importante a la base de datos, realizar un backup.

-Realizar backups periódicamente. Puedes programar un backup automático para que se ejecute cada determinado tiempo.

-Todos los password deberán de ser almacenados vía hash superior o igual a RSA-512, NO USES ningún algoritmo menor ej (MD5 ó SHA-256), **ya son obsoletos e inseguros**.
-Si se va a almacenar datos sensibles como puede ser un numero de tarjeta se deberá de almacenar de manera encriptada y deberá de ser desencriptada por la aplicación.

-Aplica los parches de seguridad la base de datos por lo menos una vez cada 6 meses.

-Limita los accesos al sistema operativo y servidor en donde esta alojada la base de datos.

-Actualiza periodicamente el Sistema Operativo en donde esta alojada la base de datos.

>>*RENDIMIENTO*

-Evita usar vistas y/o procedimientos almacenados, esto significa que deberás arreglártelas a través de una consulta compleja (subquery's) o por programación (trata de agotar todas las opciones antes de usar un procedimiento, una vista o un trigger).

-Mantener drivers o conectores actualizados (JDBC y/o ODBC).

-Si vas almacenar binarios (Blob), estas columnas deben de estar en tablas separadas y deben de estar referenciados a través de FK, esto ayudará al rendimiento tanto de búsqueda como de backup.

-Aprovisionar RAM para el servicio de base de datos.

-En el servidor que aloja la base de datos habilita el uso de "large pages".

-Habilita el cache para consultas.

-Analiza y contabiliza las consultas que se usan constantemente, después intenta optimizarlas.

-Para aplicaciones grandes, el servidor de base de datos y el servidor de aplicaciones tienen que estar en máquinas físicas diferentes, PERO en la misma red local, de preferencia en el mismo switch conectados vía LCAP o Link aggregation con tamaño de paquetes "Mega-frame".

-Realiza mantenimientos programados a la base de datos por lo menos cada 6 meses o tan frecuente como sea necesario, puede ser incluso diario, depende del requerimiento. El mantenimiento debe de incluir calculo de estadísticas, reconstrucción de indices(Oracle es especial en este punto) y compactación de la base, JUSTO EN ESE ORDEN.

-Limpiar el log de la base de datos.

-Evita usar el asterisco(*) en los SELECT.

-Particiona las tablas grandes o las que no se usan frecuentemente y de preferencia ubicadas físicamente en discos duros diferentes, si no puedes particionar también puedes usar un archivo por tabla(MySQL), en oracle se llama TableSpace.

-Utiliza índices en todas las tablas. Para consultas que regresen más de un registro utiliza índices tipo “clustered”, para consultas que solo regresan un registro usa índices tipo “non-clustered”.

-Evita (tratar de no usar) indexar tipos de datos varchar o nvarchar excepto para los estatus y las fechas.

-No utilices mas indices de los necesarios, normalmente con lo que se mencionan en este articulo son mas que suficiente.

-De ser posible evita indexar columnas que son constantemente actualizadas.

-No indexes una columna en mas de un indice, es decir, una columna solo debe de ser indexada una vez.

-Cuando construyas consultas, trata de usar las columnas que se encuentran indexadas para realizar las búsquedas.

-Si vas a realizar muchos inserts y/o updates, utiliza un batch para mejorar el rendimiento.

-Utiliza un pool de conexiones en la aplicación.

-Investiga como configurar(tuning) el cachado de la base de base de datos que estes usando.

>>FUNCIONALIDAD

-Tratar de utilizar instrucciones estándar de SQL.

-El uso de alias es obligatorio en cualquier instrucción SQL, o en su defecto puedes referenciar al schema al momento de crear la conexión.

-Referencias a base de datos completas (NOMBREBASE.DBO.MITABLA AS MIALIAS).

-Documenta la base de datos, mínimo genera el diagrama Entidad-Relación donde se muestran los tipos de datos.

-Evita guardar caracteres extraños en los campos ej. ‘, /, //, &, ?, |, °, ¬, @, ↓, , etc…

-Si tu aplicación tiene alta concurrencia utiliza el modo OLTP (online transaction processing).

-Al momento de diseñar la base enfócate en los datos, no en la aplicación.

-Consulta diagramas Entidad-Relación de otros sistemas para tener un punto de referencia y te des una idea de cómo funcionan, de esta forma puedes copiar lo que mejor te funcione. Cada requerimiento es diferente.

-Pide la opinión de tus colaboradores.

-Utiliza las técnicas de modelado de datos, justo como en la programación orientada a objetos.

-Cuando insertes strings a la base de datos, trata de guardar toda la información en mayúsculas o en minúsculas, define un estándar para tu base.

-Recuerda que puedes hacer que una consulta sea sensible a mayúsculas CI (Case Insensitive) y/o sensible a acentos AI (Accent Insensitive).

-La mejor forma de usar fechas en cualquier base de datos es el String (aaaa-mm-dd), es compatible con cualquier base, sin generar problemas de rendimiento, estabilidad o usabilidad. Para las estampas de tiempo usa un BIGINT e introduce un Long que represente los milisegundos.

-Realiza pruebas de escritorio para verificar que tu diseño funcione correctamente y cumpla con los requerimientos funcionales y no funcionales.

-Ten la mente abierta y experimenta por ti mismo las sugerencias de este post y de otras fuentes de información. Toma aquello que te fue útil.