1. Usar nombres consistentes y bien definidos para los nombres de los campos y tablas (e.g. Colegio, EsudianteCurso, CursoId, etc).
2. Usar nombres de tabla en singular e singular (e.g. usa EstudianteCurso en vez deEstudianteCursos). La tabla representa una colección de entidades, no es necesario el nombre en plural.
3. No se debe usar espacios en los nombres de las tablas, esto fuerza a usar ‘{‘, ‘[‘, ‘“’ lo cual puede conllevar a la introducción de errores.
4. No se debe usar prefijos o sufijos innecesarios para los nombres de las tablas (e.g. use Escuela en vez de TblEscuela o TablaEscuela).
5. Por seguridad, almacenar información sensible encriptada (como las claves de acceso). Usar el valor desencriptado solo en la aplicación, cuando se requiera.
6. Usar tipo “integer” para los campos “id” en todas las tablas.
7. Si necesita indexar tablas elija tipo de datos “integer” (o sus variantes). El uso de “varchar” puede causar problemas de desempeño.
8. Use tipo “bit” valores Sí/No. Usar “integer” o “varchar” conlleva consumo de espacio innecesario. Se recomienda empezar el nombre del campo con “Es”.
9. Se recomienda proveer una autenticación específica para el acceso a la base de datos. No se debe asignar role “admin” a todos los usuarios.
10. Evite el uso de “select \*” a menos que sea realmente necesario. Use "select [columnas requeridas]" para un mejor desempeño.
11. Use un ORM (object relational mapping) framework (i.e. hibernate, iBatis ...) si el código de la aplicación es suficientemente grande. Problemas de desempeño en ORM frameworks se pueden evitar a través de la correcta configuración de los parámetros.
12. Para tablas grandes o de poco uso se recomienda particionarlas físicamente para un mejor desempeño de las columnas.
13. Para sistemas de base de datos grandes, sensitivos o críticos use métodos de recuperación como: clustering, auto respaldo, replicación, etc.
14. Use constraints (foreign key, check, not null) para la integridad de los datos. No mantener todo el control al código de la aplicación.
15. Uno de los principales problemas es la falta de documentación. Se debe documentar el diseño de base de datos con esquemas Entidad-Relación. También describa los elementos creados (triggers, stored procedures y scripts).
16. Use índices para consultas frecuentas en tablas grandes. Herramientas de análisis puedes ser utilizadas para determinar donde se deben definir índices. Para consultas que devuelven un conjunto de registros, indices clustered suelen ser mejor. Para consultas puntuales, indices non-clustered son usualmente mejores.
17. El servidor de base de datos y web deben estar en equipos diferentes. Esto provee mayor seguridad (los ataques no se pueden acceder directamente a los datos) y el desempeño de procesador y memoria será mejor.
18. Columnas del tipo “Image” and “blob data” no deben definirse en tablas que se consultan frecuentemente, esto genera problemas de desempeño. Esta información debe estar en tablas separadas.
19. Normalizar la base de datos para un mejor desempeño. En caso contrario se podría generar data duplicada, del otro lado se debe tener cuidado de no generar demasiadas relaciones de tablas, en ambos casos se puede impactar generando un pobre desempeño.
20. Tenga cuidado en el diseño y modelamiento de la base de datos, el tiempo invertido en diseño puede salvar información de mantenimiento y rediseño.