Las 12 reglas de Codd.

Regla 1: la regla de la información.

Toda la información en la base de datos es representada unidireccionalmente, por valores en posiciones de las columnas dentro de filas de tablas. Toda la información en una base de datos relacional se representa explícitamente en el nivel lógico exactamente de una manera: con valores en tablas.

Regla 2: la regla del acceso garantizado.

Todos los datos deben ser accesibles sin ambigüedad. Esta regla es esencialmente una nueva exposición del requisito fundamental para las llaves primarias. Dice que cada valor escalar individual en la base de datos debe ser lógicamente direccionable especificando el nombre de la tabla, la columna que lo contiene y la llave primaria.

Dice que cada valor escalar individual en la base de datos debe ser lógicamente direccionarle especificando el nombre de la tabla, la columna que lo contiene y la llave primaria. Refuerza la importancia de las claves primarias para localizar datos en la base de datos.

Cada tabla debe tener su nombre único.

No puede haber dos filas iguales. No se permiten los duplicados.

Todos los datos en una columna deben ser del mismo tipo.

Código	Nombre	Posición	Salario
1	Edgardo Trujillo	Gerente	19000
2	Lidimarie Fonsi	Empleada	12000
3	Jean Piaget	Empleado	13500
4	Jerome Bruner	Empleado	14000

Regla 3: tratamiento sistemático de valores nulos

El sistema de gestión de base de datos debe permitir que haya campos nulos. Debe tener una representación de la "información que falta y de la información inaplicable" que es sistemática, distinto de todos los valores regulares.

Ejemplo: Cuando la base no puede manejar la entrada de datos nulos, lo cual podría producirnos un error, por esto es conveniente tener en cuenta los dominios de cada tipo de atributo o campo de la tabla así como la inexistencia del dato

requerido. Hay problemas para soportar los valores nulos en las operaciones relacionales, especialmente en las operaciones lógicas.

Regla 4: catálogo dinámico en línea basado en el modelo relacional

El sistema debe soportar un catálogo en línea, el catálogo relacional debe ser accesible a los usuarios autorizados. Es decir, los usuarios deben poder tener acceso a la estructura de la base de datos (catálogo).

Ejemplo: El uso de clave para controlar lo que cada usuario puede ver o manejar dentro de la base de datos es muy importante y debe poder ser accesible en cualquier momento por quien así lo requiera. Debe de ser posible el acceso a datos y metadatos.

Regla 5: La regla del sub-lenguaje Integral

Debe haber al menos un lenguaje que sea integral para soportar la definición de datos, manipulación de datos, definición de vistas, restricciones de integridad, y control de autorizaciones y transacciones". Esto significa que debe haber por lo menos un lenguaje con una sintaxis bien definida que pueda ser usado para administrar completamente la base de datos.

Regla 6: La regla de la actualización de vistas

Todas las vistas que son teóricamente actualizables, deben ser actualizables por el sistema mismo. La mayoría de las RDBMS permiten actualizar vistas simples, pero deshabilitan los intentos de actualizar vistas complejas.

La actualización debe de ser automática, sin necesidad de que el usuario tenga que estar actualizando manualmente. Una vista puede ser el conjunto de socios de la biblioteca que viven en Ciudad Capital. Si quiero añadir un socio que vive en Izabal a la vista (sería actualizable), debo poder hacerlo sin notarlo, debe encargarse el SGBD de manejarlo.

Regla 7: alto nivel de inserción, actualización, y cancelación.

El sistema debe soportar suministrar datos en el mismo tiempo que se inserte, actualiza o esté borrando. Esto significa que los datos se pueden recuperar de una base de datos relacional en los sistemas construidos de datos de filas múltiples y/o de tablas múltiples.

Esto significa que las cláusulas SELECT, UPDATE, DELETE e INSERT deben estar disponibles y operables sobre los registros, independientemente del tipo de

relaciones y restricciones que haya entre las tablas.

Regla 8: Independencia física de datos:

El acceso de usuarios a la base de datos a través de terminales o programas de aplicación, debe permanecer consistente; lógicamente cuando quiera que haya cambios en los datos almacenados, o sean cambiados los métodos de acceso a los datos.

Regla 9: Independencia lógica de datos:

La independencia lógica de los datos especifica que los programas de aplicación y las actividades de terminal deben ser independientes de la estructura lógica, por lo tanto los cambios en la estructura lógica no deben alterar o modificar estos programas de aplicación.

Regla 10: independencia de la integridad, las limitaciones de la integridad se deben especificar por separado de los programas de la aplicación y se almacenan en la base de datos. Debe ser posible cambiar esas limitaciones sin afectar innecesariamente las aplicaciones existentes.

Regla 11: independencia de distribución: El sistema debe poseer un lenguaje de datos que pueda soportar que la base de datos esté distribuida físicamente en distintos lugares sin que esto afecte o altere a los programas de aplicación. El soporte para bases de datos distribuidas significa que una colección arbitraria de relaciones, bases de datos corriendo en una mezcla de distintas máquinas y distintos sistemas operativos y que esté conectada por una variedad de redes, pueda funcionar como si estuviera disponible como en una única base de datos en una sola máquina.

Esta regla es responsable de tres tipos de transparencia de distribución: Transparencia de localización. El usuario tiene la impresión de que trabaja con una BD local. (Aspecto de la regla de independencia física) Transparencia de fragmentación. El usuario no se da cuenta de que la relación con que trabaja está fragmentada. (Aspecto de la regla de independencia lógica de datos). Transparencia de replicación. El usuario no se da cuenta de que pueden existir copias (réplicas) de una misma relación en diferentes lugares. Ejemplo: Las mismas órdenes y programas se ejecutan igual en una BD centralizada que en una distribuida. Voy a sacar dinero de un cajero. El dinero de mi cuenta se guarda en un servidor. Saco un extracto de movimientos. Y esa información se guarda ¿En el mismo servidor de antes? No tiene por qué y yo no tengo porqué saberlo (transparencia). En un servidor, no tiene por qué guardarse todos los datos de

todos los clientes de un banco. 12 Es más lógico que se guarden en varios servidores, uno por zona geográfica, por ejemplo. Los clientes no tienen por qué saberlo. ¿Ellos lo notan?, No y además se tiene que encargar el SGBD de que sea transparente.

Regla 12: regla de la no subversión:

Si un sistema relacional tiene un lenguaje de bajo nivel (un registro de cada vez), ese bajo nivel no puede ser usado para saltarse (subvertir) las reglas de integridad y los limitantes expresados en los lenguajes relacionales de más alto nivel (una relación (conjunto de registros) de cada vez). Algunos productos solamente construyen una interfaz relacional para sus bases de datos No relacionales, lo que hace posible la subversión (violación) de las restricciones de integridad. Esto no debe ser permitido. Algunos problemas no se pueden solucionar directamente con el lenguaje de alto nivel. Normalmente se usa SQL inmerso en un lenguaje anfitrión para solucionar estos problemas. Se utiliza el concepto de cursor para tratar individualmente las tuplas de una relación. En cualquier caso no debe ser posible saltarse los limitantes de integridad impuestos al tratar las tablas a ese nivel.

Ejemplo: No debe ser posible saltarse los limitantes de integridad impuestos al tratar las tuplas a ese nivel. Si se puede, con las facilidades que da el SGBD utilizar un sistema para acceder a los registros (desde aplicaciones externas al SGBD), éste sistema debe respetar todas las reglas anteriores. Debe seguir manteniendo todas las integridades de los datos.

Bit-Map y Bit-tree

Los índices B-Tree son los tipos utilizados por los sistemas OLTP y se implementan principalmente de forma predeterminada. Bitmap, por el contrario, es un formato de índice altamente comprimido utilizado en la mayoría de las bases de datos.

El mapa de bits es la opción preferida para elementos de columna con parámetros bajos, como el género, que son solo 2 valores y son preferidos. Las estadísticas en el repositorio también son una buena característica de los datos que se pueden hacer muy bien con Bitmap. Otra característica de Bitmap es el flujo de bits, donde cada bit se aplica al valor de la columna en una fila de la tabla.

Los índices de maps de bits se usan principalmente en aplicaciones de almacenamiento de datos, donde la database es de solo lectura, excepto los processs ETL, y generalmente necesita ejecutar consultas complejas contra un esquema en estrella , donde los índices de bitmap pueden acelerar el filtrado

según las condiciones en sus tablas de dimensiones. que normalmente no tienen demasiados valores distintos.

El índice de árbol B es un índice creado en columnas que contienen valores muy únicos. El índice B-Tree tiene registros que contienen cada palabra clave y una cadena y un valor. Si el servidor encuentra una restricción apropiada asociada con el valor solicitado, se establece un puntero para recuperar la cadena.

Cada índice B-Tree impone una pequeña penalización al insert / actualizar valores en la tabla indexada. Esto puede ser un problema si tiene muchos índices en una tabla muy ocupada.

Estas características hacen que los índices B-Tree sean muy útiles para acelerar las búsquedas en aplicaciones OLTP, cuando se trabaja con sets de datos muy pequeños a la vez, la mayoría de las consultas se filtran por ID y se desea un buen performance simultáneo.

Referencias:

https://esacademic.com/dic.nsf/eswiki/4866

https://www.mindmeister.com/es/1079684487/las-12-reglas-de-codd-del-modelo-relacional?fullscreen=1

https://usacdatospb.files.wordpress.com/2015/09/grupo-4.pdf

https://es.bccrwp.org/compare/difference-between-b-tree-and-bitmap-97a604/

https://sql.dokry.com/b-tree-vs-ndices-de-database-de-bitmap.html