UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

BASES DE DATOS
1644
GRUPO: 01

ING. FERNANDO ARREOLA FRANCO

TAREA 10
REGLAS DE CODD, INDICES

SÁNCHEZ MANJARREZ ANDREW

REGLAS DE CODD: Sistema de 13 reglas (numeradas del 0 al 12) definidas por Edgar F. Codd para determinar la fidelidad de un sistema de base de datos al modelo relacional.

Regla 1: Información • Todos los datos deben estar almacenados en las tablas • Esas tablas deben de cumplir las premisas del modelo relacional •No puede haber información a la que accedemos por otra vía

Regla 2: Acceso garantizado •Cualquier dato es accesible sabiendo la clave de su fila y el nombre de su columna o atributo • Por ejemplo el "Sánchez" es un dato al que podremos acceder conociendo la clave de la persona en concreto y usando el atributo "Primer apellido" • Si a un dato no podemos acceder de esta forma, no estamos usando un modelo relacional

Regla 3: Tratamiento sistemático de los valores nulos • Esos valores pueden dar significado a la columna que los contiene (una persona sin teléfono, tendrá valor nulo en el teléfono) • El SGBD tiene que tener la capacidad de manejar valores nulos • El SGBD reconocerá este valor como un valor distinto de cualquier otro • El SGBD sabrá aplicarle la lógica apropiada • Es un valor independiente del tipo de datos de la columna

Regla 4: Catálogo en línea relacional • El catálogo en línea es el diccionario de datos • El diccionario de datos se debe de poder consultar usando las mismas técnicas que para los datos • Los metadatos, por tanto, se organizan también en tablas relacionales • Si SELECT es la instrucción que consulta datos, también será la que consulta los metadatos

Regla 5: Sublenguaje de datos completo •Al menos tiene que existir un lenguaje capaz de hacer todas las funciones del SGBD •No puede haber funciones fuera de ese lenguaje • Puede haber otros lenguajes en el SGBD para hacer ciertas tareas • Pero esas tareas también se deben poder hacer con el "lenguaje completo"

Regla 6: Vistas actualizadas • Las vistas tienen que mostrar información actualizada •No puede haber diferencia entre los datos de las vistas y los datos de las tablas base

Regla 7: Inserciones, modificaciones y eliminaciones de alto nivel • La idea es que el lenguaje que maneja la BD sea muy humano • Eso implica que las operaciones DML trabajen con conjuntos de filas a la vez • Para modificar, eliminar o añadir datos no hará falta programar de la forma en la que lo hacen los lenguajes de tercera generación como C o Java

Regla 8: Independencia física •Cambios en la física de la BD no afecta a las aplicaciones ni a los esquemas lógicos • El acceso a las tablas (elemento lógico) no cambia porque la física de la base de datos cambie

Regla 9: Independencia lógica •Cambios en el esquema lógico (tablas) de la BD no afectan al resto de esquemas • Si cambiamos nombres de tabla, o de columna o modificamos información de las filas, las aplicaciones (esquema externo) no se ven afectadas • Es más difícil de conseguir

Regla 10: Independencia de integridad • Las reglas de integridad (restricciones) deben de ser gestionadas y almacenadas por el SGBD

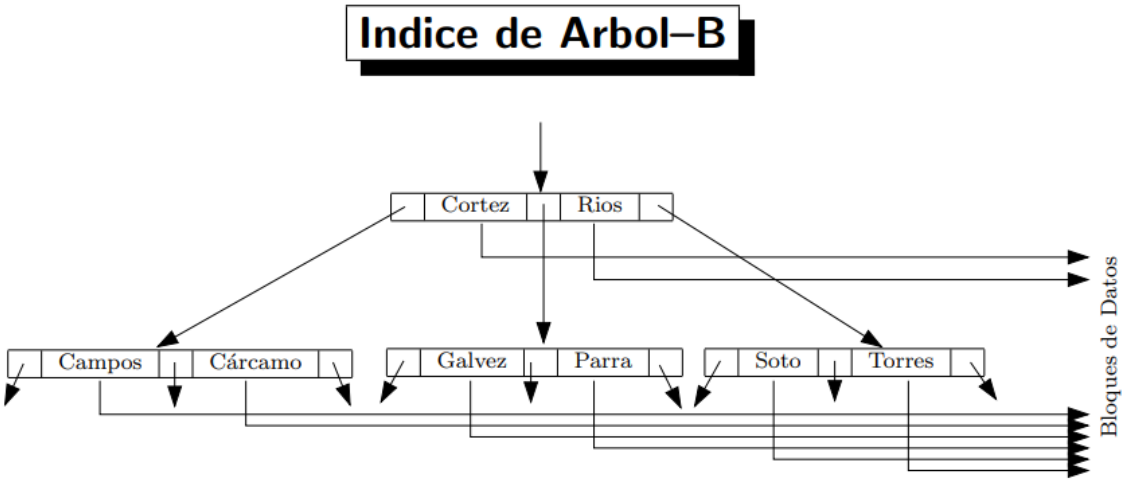
Regla 11: Independencia de distribución •Que la base de datos se almacene o gestione de forma distribuida en varios servidores, no afecta al uso de la misma ni a la programación de las aplicaciones de usuario • El esquema lógico es el mismo independientemente de si la BD es distribuida o no

Regla 12: No subversión • La base de datos no permitirá que exista un lenguaje o forma de acceso, que permita saltarse las reglas anteriores

Regla 0 •Un SGBD relacional debe gestionar sus BD de forma completa usando el modelo relacional

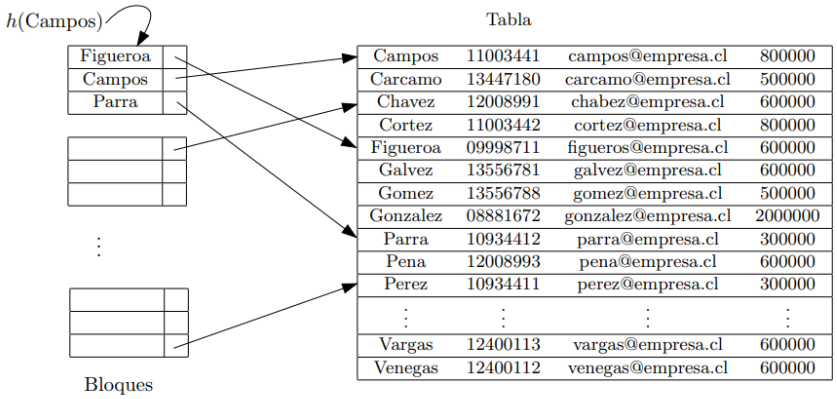
INDICES

INDICE B-TREE: Los Arboles–B (B ´ +Trees y BTrees) son una de las estructuras de datos más usadas para mantener índices de acceso a BD. Son excelentes para hacer búsquedas externas en memoria secundaria (en disco). Aseguran un tiempo eficiente, de orden logarítmico en el peor caso, en cuanto al tiempo de procesamiento y en cuanto a la cantidad de accesos al disco. Estructura de Árbol balanceado, todas las ´ hojas del árbol están a la misma profundidad. Cada nodo del árbol se mantiene en una unidad (página) de memoria secundaria. Cada nodo tiene un máximo y un mínimo de punteros a tuplas de la base de datos. El árbol completo funciona como índice. Las inserciones y eliminaciones de registros en el índice se pueden realizar de forma muy eficiente, no es necesario reconstruir el índice.



INDICE HASH: Una desventaja de los esquemas de índices es que debemos acceder a una estructura adicional para localizar los datos. La técnica de Hashing (o asociación) nos permite evitar este acceso adicional. Disponemos de una función (función de hash) que tiene como dominio las posibles llaves de búsqueda y como recorrido las posibles posiciones en memoria secundaria donde se encuentran los datos. El principio fundamental del hashing es que, a pesar de que el conjunto de las posibles llaves puede ser muy grande, el conjunto de llaves efectivamente almacenadas en la BD es mucho más pequeño. Ejemplo: Todos los nombres posibles (cualquier secuencia de caracteres) vs. los nombres de los empleados de la empresa. Si h es la función de hash, se busca que la probabilidad de que h(k1) = h(k2) para dos elementos k1 /= k2 almacenados en la BD sea pequeña. A pesar de que la probabilidad de que h(k1) = h(k2) para dos elementos k1 /= k2 es pequeña, de todas maneras puede ocurrir que h(k1) = h(k2). A esto se llama colisión. Todos los valores que colisionan se almacenan en un mismo bloque (como una lista ligada por ejemplo). Cada elemento en el bloque apunta a la dirección física representada por la llave de

búsqueda. Una buena función de hash ⇒ pocas colisiones. Para encontrar los datos asociados a una llave de búsqueda se debe: • Aplicar h a la llave para obtener la dirección de un bloque. • Buscar secuencialmente dentro del bloque el puntero asociado a esa llave. • Seguir el puntero para acceder a los datos. Es bastante eficiente si h distribuye uniformemente las llaves en los bloques (minimiza colisión). La función h debe ser construida antes de comenzar a ocupar la BD y no puede cambiar. La cantidad de bloques utilizables se debe especificar previamente. Si la BD crece mucho, la probabilidad de colisión aumenta ⇒ baja en la eficiencia.



INDICE BITMAP: Un índice de mapa de bits es un tipo especial de índice de base de datos que utiliza mapas de bits. Tradicionalmente se han considerado para trabajar bien para datos como el género, que tiene un pequeño número de valores Distintos, por ejemplo, hombres y mujeres, pero muchas ocurrencias de esos valores. Esto sucedería si, por ejemplo, había datos de género para cada residente en la ciudad. Sin embargo, algunos investigadores sostienen que los índices de mapa de bits también son útiles para los datos de valor único que no se actualiza con frecuencia. Índices de mapa de bits tiene un espacio importante y una ventaja de rendimiento sobre otras estructuras para tales datos.

REFERENCIAS:

Que es un Indice BitMap en Oracle?. (2020). Retrieved 21 October 2020, from http://williamaltamiranop.blogspot.com/2009/09/que-es-un-indice-bitmap-en-oracle.html

Yaseen, A. (2020). Uso de diferentes tipos de índices SQL Server. Retrieved 21 October 2020, from https://www.sqlshack.com/es/uso-de-diferentes-tipos-de-indices-sql-server/

BPC, V. (2020). 12 reglas de Codd para bases de datos Relacionadas. Retrieved 21 October 2020, from https://medievalstrucos.wordpress.com/2013/07/18/12-reglas-de-codd-para-bases-de-datos-relacionadas/

Perez Rojas, J. (2005). *Indexación y Hashing* (2nd ed.). Universidad de Talca.