Firebird para Especialistas de bases de datos:
Episodio 1 - Indices.

by Ann Harrison
translated by Raúl Valencia

Resumen

Firebird difiere en formas muy significativas de otros sistemas de bases de datos relacionales. Entender las diferencias le permitirá crear aplicaciones de Firebird con mejores rendimientos.

Audiencia: Diseñadores experimentados de aplicaciones de bases de datos.

Cambiar hacia Firebird puede ser desconcertante para los diseñadores que han trabajado con otros sistemas de bases de datos relacionales. En teoría, estos sistemas separan el diseño lógico de una aplicación del almacenamiento físico de los datos, permitiendo a los diseñadores enfocarse en los datos que quieren que su aplicación acceda, en vez de cómo éstos deberían de recuperarse. En la práctica, la mecánica de cada sistema de base de datos hace que ciertos estilos de acceso sean mucho más rápidos que otros: Los diseñadores aprenden a usar métodos que trabajan con los sistemas de bases de datos que ellos conocen.

Los diseñadores que están familiarizados con Oracle, o SQL Server de Microsoft, encontrarán que los índices y los modelos de concurrencia y recuperación de fallos de Firebird se comportan de forma distinta a los de las bases de datos que ellos conocen. Entender y trabajar con estas diferencias hará su cambio hacia Firebird menos estresante y más exitoso. Este artículo se enfoca en las características inusuales de los índices de Firebird.

Tipos de índices

Firebird soporta únicamente un tipo de índice: una variante del árbol-B. Los índices pueden ser únicos, o también pueden permitir duplicados. Pueden conformarse de una sola clave, o de una clave compuesta, en forma ascendente o descendente.

Localización de Registros

Varios productos de bases de datos agrupan los registros según el índice de la clave primaria, ya sea almacenando directamente los datos en el índice o usando la clave para agrupar los registros. En un sistema bien balanceado, este agrupamiento hace que las búsquedas de clave primaria sean muy eficientes. Si la fila entera se almacena en el índice, el nivel de datos se amplía mucho, haciendo que todo el índice se vuelva profundo, y más costoso de recorrer que un índice menos profundo y denso. El agrupamiento de registros puede resultar en un menguado almacenamiento de overflows (desbordamientos), dependiendo de las especificaciones del diseño y la distribución de los datos.

Firebird almacena los registros en páginas de datos, usando la página más accesible que tenga el espacio suficiente. Los índices se almacenan en páginas de índice, y contienen un localizador de registros en el nodo hoja.

Los costos de acceso de los índices primarios y secundarios

Cuando los datos están agrupados según la clave primaria, el acceso a los datos por medio de ésta es muy rápido. El acceso por medio de los índices secundarios es más lento, especialmente cuando el índice de la clave secundaria usa la clave primaria como un localizador de registros: Una búsqueda por un índice secundario se convierte en dos.

En Firebird, el costo de las búsquedas según los índices primario y los secundarios es idéntica.

Estrategias de acceso a los índices

La mayoría de los sistemas de bases de datos leen un nodo de índice y recuperan los datos -técnica que puede llevar a un rebote entre las páginas de índice y las de datos, lo cual puede resolverse con un control adecuado de posicionamiento, asumiendo que el DBA tiene el tiempo y la habilidad para hacerlo. Para los índices no agrupados, esta técnica también resulta en lecturas repetidas de las páginas de datos.

Firebird recolecta los localizadores de las filas que califican desde el índice, construye un mapa de bits de localizadores de registros, y luego los lee en el orden en el que están almacenados físicamente.

Optimización de índices

La mayoría de los optimizadores de las bases de datos deben de seleccionar un índice por tabla como su ruta a los datos, por causa de sus estrategias de acceso, que enlazan muy estrechamente el acceso a los índices y a las filas de datos. Firebird puede usar varios índices en una tabla, haciendo operaciones de AND y OR en los mapas de bits que crea desde el índice antes de acceder los datos.

Si se tiene una tabla en donde varios campos diferentes se utilizan para restringir los datos recuperados desde una consulta, la mayoría de las bases de datos requieren que se defina un sólo índice que incluye todos los campos. Por ejemplo, si se busca una película que se lanzó en 1964, dirigida por Stanley Kubrick, y distribuida por Columbia, se necesita un índice por año, director y distribuidor. Si alguna vez se requiriese encontrar todas las películas dirigidas por Kubrick, también se necesitará un índice por solamente el director, y así sucesivamente. Con Firebird, se define un índice por Director, otro por Distribuidor, y otro por año, y se utilizarán en varias combinaciones.

Cadenas duplicadas largas

Algunas bases de datos (como por ejemplo Firebird 2) son mejores que otras (como por ejemplo Firebird 1) en la remoción de datos de las cadenas duplicadas largas (>1000) en los índices. Si se necesita un índice en un campo con selectividad baja para una base de datos Firebird 1.x, hay que crear una clave compuesta con la columna que se quiere indexar primero, y otra columna más selectiva después. Por ejemplo, si se tiene un índice en la columna FechaPago en la tabla Cheques, y cada fila es almacenada con dicho valor nulo cuando el cheque se emite, y luego modificado cuando el cheque se cobra, entonces se debe de crear un índice de dos partes, como por ejemplo (FechaPago, NumeroCuenta), en vez de una sola clave en FechaPago.

Índices en vez de datos

Las bases de datos que no utilizan tecnologías de versionamiento de filas resuelven algunas consultas (conteos por ejemplo) leyendo el índice sin leer realmente los datos. Los índices de Firebird (Como los de PostgreSQL y otras bases de datos que nativamente implementan versionamiento de filas) contienen entradas que no son todavía visibles a otras transacciones, y entradas que ya no son relevantes para algunas transacciones activas. La única forma de saber cuándo una entrada de índice representa datos visibles para una transacción en particular es leer la fila misma.

El tópico de las versiones de las filas merece una larga discusión. Brevemente, cuando Firebird almacena un nuevo registro, lo etiqueta con el identificador de la transacción que lo creó. Cuando modifica una fila, crea una nueva versión de la misma, marcada con el identificador de la transacción que hizo la modificación. Ese registro apunta a la versión previa. Hasta que la transacción que creó la nueva versión emite el comando COMMIT, todas las otras transacciones continuarán leyendo la versión anterior de la fila.

En el ejemplo anterior, cuando una transacción modifica el campo indexado FechaPago, Firebird crea una nueva versión conteniendo los nuevos datos y el identificador de la transacción que hizo el cambio. El índice en ese campo tiene dos entradas para ese registro, una para el valor nulo original y otra para el nuevo valor de FechaPago.

El índice, entonces, no tiene la suficiente información para determinar qué entrada debe de contarse en respuesta a una consulta como "`select count(*) from Cheques where FechaPago is not null`".

Longitud de la clave de índice.

En Firebird Versión 1.x, la longitud total de una clave de índice debe de ser menor a 252 bytes. Los índices de clave compuesta y los índices con secuencias de ordenamiento no binarias son más restrictivos, por las razones que se describen en la sección de compresión de claves. Firebird 2 permite claves de hasta 1/4 del tamaño de la página, o un máximo de 4KB.

Representación de las claves de los índices

Firebird convierte todas las claves de los índices a un formato que puede compararse byte-por-byte. Con la excepción de las columnas enteras de 64 bits, todas las columnas numéricas y de fecha se almacenan como claves enteras de doble precisión, y el número de doble precisión es manipulado para compararse byte por byte. Cuando se ejecuta una búsqueda indexada, Firebird convierte el valor de entrada al mismo formato que la clave almacenada. Lo que esto significa para el programador es que no hay diferencia de velocidad entre los índices en cadenas, números y fechas. Todas las claves se comparan byte por byte, sin importar las reglas de su tipo original de datos.

Compresión de claves de índice

Las claves de índice de Firebird se almacenan siempre con compresión de prefijos y sufijos.

La compresión de sufijos remueve los espacios al final de los campos de cadena, y los ceros al final de los campos numéricos. (Hay que recordar que la mayor parte de los valores numéricos se almacenan como de doble precisión, y, por lo tanto, los ceros al final no son significativos). La compresión de sufijos se hace por cada campo clave en una clave compuesta, sin perder los límites entre cada uno. Tras remover los espacios o los ceros al final, la compresión de los índices ajusta el valor a un múltiplo de cuatro bytes, e inserta bytes "marcadores" cada cuatro, para indicar la posición del campo en la clave.

Considerando el caso de una clave de tres columnas con este conjunto de valores:

"abc ", "def ", "ghi " "abcdefghi ", " ", " "

El simplemente eliminar los espacios al final hace que estos dos conjuntos de valores sean iguales. En vez de ello, Firebird cambia el primer conjunto a “abc 1def 2ghi 3”, y el segundo en “abcd1efgh1i 1 2 3”.

Firebird 1.x comprime el prefijo de las claves de índice conforme se almacenan en las páginas en el índice. Almacena la primera clave en una página sin comprimir su prefijo. Las claves subsiguientes se almacenan tras reemplazar los bytes del inicio que coinciden con los bytes del inicio de la clave anterior con un sólo byte, que contiene el número de bytes que se han saltado. Las dos claves se almacenarían de esta forma: "0abc 1def 2ghi 3" "3d1efgh1i 1 2 3"

Una entrada de índice que coincide exactamente con la entrada anterior es almacenada como un sólo byte que contiene su longitud. Firebird 2 también ejecuta compresiones de prefijos, pero usa representaciones más densas.

La combinación de las técnicas de compresión elimina algunas de las reglas acerca de la construcción de las claves. La compresión de sufijos ocurre en todos los segmentos de una clave, de manera que los valores largos de tipo varchar se deben de ubicar en su ubicación lógica en una clave compuesta, no forzarse al final. Por otro lado, si parte de una clave compuesta tiene un número largo de valores duplicados, debe de estar al frente de una clave compuesta, para tomar ventaja de la compresión de prefijos.