Análisis de Consultas

Contenido

Notas para el instructor

Este módulo proporciona a los alumnos un conocimiento profundo de la forma en que el optimizador de consultas de Microsoft® SQL Server™ 2000 evalúa y procesa las consultas que contienen los operadores AND y OR así como las operaciones de combinación y si los alumnos deben omitir el optimizador de consultas.

En las prácticas los alumnos ejecutarán varias consultas y analizarán cómo el optimizador de consultas procesa las consultas que contienen los operadores lógicos AND y OR. Los alumnos también analizarán cómo se procesan las combinaciones de bucle anidado, mezcla y hash.

Después de completar este módulo, los alumnos serán capaces de:

- Analizar la ganancia de rendimiento de escribir consultas eficientes y crear índices útiles para consultas que contengan el operador lógico AND.
- Analizar la ganancia de rendimiento de escribir consultas eficientes y crear índices útiles para consultas que contengan el operador lógico OR.
- Evaluar de qué forma el optimizador de consultas utiliza distintas estrategias de combinación para la optimización de consultas.

Introducción

Objetivo del tema
Proporcionar una introducción a los temas y objetivos del módulo.

Explicación previa
En este módulo aprenderá la forma en que el optimizador de consultas procesa los operadores lógicos AND y OR, y cómo utiliza distintas estrategias de combinación.

- Consultas que utilizan el operador AND
- Consultas que utilizan el operador OR
- Consultas que utilizan operaciones de combinación

Después de realizar esta práctica, el alumno será capaz de:

- Analizar la ganancia de rendimiento de escribir consultas eficientes y crear índices útiles para consultas que contengan el operador lógico AND.
- Analizar la ganancia de rendimiento de escribir consultas eficientes y crear índices útiles para consultas que contengan el operador lógico OR.
- Evaluar de qué forma el optimizador de consultas utiliza distintas estrategias de combinación para la optimización de consultas.

Consultas que utilizan el operador AND

Objetivo del tema
Describir cómo se optimiza una consulta que contiene el operador AND.

Explicación previa
La forma en que el optimizador de consultas procesa el operador AND depende de si existen índices en algunas o todas las columnas a las que se hace referencia en la cláusula WHERE.

- Procesamiento del operador AND
  - Devuelve las filas que cumplen todas las condiciones de cada criterio especificado en la cláusula WHERE
  - Limita de forma progresiva el número de filas devueltas con cada condición de búsqueda adicional
  - Puede utilizar un índice por cada condición de búsqueda de la cláusula WHERE
- Directrices de indización y consideraciones de rendimiento
  - Definir un índice en un criterio de búsqueda altamente selectivo
  - Evaluar el rendimiento entre crear varios índices de una columna y un índice compuesto

La forma en que el optimizador de consultas procesa el operador AND depende de si existen índices en algunas o todas las columnas a las que se hace referencia en la cláusula WHERE.

Procesamiento del operador AND

Cuando una consulta contiene el operador AND, el optimizador de consultas:

- Devuelve las filas que cumplen todas las condiciones de cada criterio especificado en la cláusula WHERE.
- Limita de forma progresiva el número de filas devueltas con cada condición de búsqueda adicional.
- Puede utilizar un índice por cada condición de búsqueda de la cláusula WHERE.
- Siempre utiliza un índice si éste es útil.

  Si los índices no son útiles para ninguna columna de la cláusula WHERE, el optimizador de consultas realiza un recorrido de tabla o de índice agrupado.
- Puede utilizar varios índices si son útiles.

  Si hay varios índices y algunos son útiles para alguna columna de la cláusula WHERE, el optimizador de consultas determina la combinación de índices que se utilizará.

  El plan de ejecución puede mostrar que uno o la mayoría de los índices se utilizaron para procesar la consulta. La combinación de índices está determinada por:
  - La selectividad de la búsqueda.
  - El tipo de índices que existen, como agrupados o no agrupados.
  - Posibilidad de tratar el índice.
  - Existencia de una vista indizada.

- Puede utilizar un único índice incluso si existen varios útiles.

  Si el optimizador de consultas encuentra un índice que es altamente selectivo, utiliza dicho índice. A continuación, usa la operación de filtro para procesar las condiciones de búsqueda restantes en las filas que cumplen la condición.

Directrices de indización y consideraciones de rendimiento

La mejor forma de indizar para consultas que contienen el operador AND es disponer, como mínimo, de un criterio de búsqueda altamente selectivo y definir un índice en dicha columna.

Es recomendable comparar la diferencia de rendimiento al crear varios índices de una columna y un índice compuesto. El rendimiento de las consultas no mejora necesariamente con la indización de todas las columnas que forman parte del operador AND. Sin embargo, se puede obtener ventaja de tener varios índices si las columnas a las que hace referencia el operador AND son de selectividad más baja.

Consultas que utilizan el operador OR

Objetivo del tema
Describir cómo se optimiza una consulta que contiene el operador OR.

Explicación previa
La forma en que el optimizador de consultas procesa el operador OR también depende de si existen índices en algunas o todas las columnas a las que se hace referencia en la cláusula WHERE.

- Devuelve las filas que cumplen cualquier condición de cada criterio especificado en la cláusula WHERE
- Aumenta de forma progresiva el número de filas devueltas con cada condición de búsqueda adicional
- Puede utilizar un índice o índices diferentes para cada parte del operador OR
- Siempre realiza un recorrido de tabla o de índice agrupado si una columna a la que se hace referencia en el operador OR no tiene un índice o si éste no es útil
- Puede utilizar varios índices

La forma en que el optimizador de consultas procesa el operador OR también depende de si existen índices en algunas o todas las columnas a las que se hace referencia en la cláusula WHERE.

Cuando una consulta contiene el operador OR, el optimizador de consultas:

- Devuelve las filas que cumplen cualquier condición de cada criterio especificado en la cláusula WHERE.
- Aumenta de forma progresiva el número de filas devueltas con cada condición de búsqueda adicional.
- Puede utilizar un índice que cumpla todas las partes del operador OR o utiliza índices distintos para cada parte del operador OR.
- Siempre realiza un recorrido de tabla o de índice agrupado si una columna a la que se hace referencia en el operador OR no tiene un índice o si éste no es útil.
- Si hay varios índices y todos son útiles, el optimizador de consultas:
  - Busca en una tabla mediante un índice para cada columna.
  - Ordena los valores que cumplen la condición para cada columna.
  - Combina los resultados.
  - Recupera las filas que cumplen la condición mediante la operación de búsqueda de marcadores.

Consultas que utilizan operaciones de combinación

Objetivo del tema
Presentar los temas de esta sección.

Explicación previa
En esta sección describiremos las optimizaciones de consultas que utilizan operaciones de combinación.

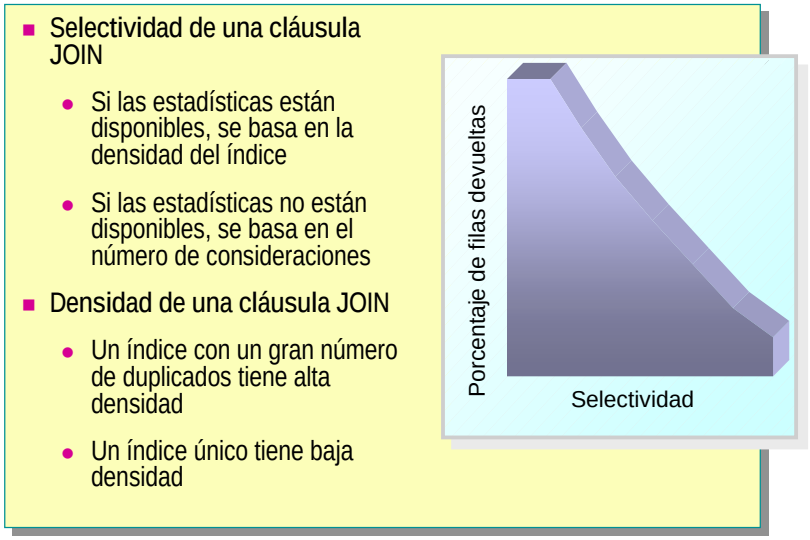
- Selectividad y densidad de una cláusula JOIN
- Cómo se procesan las combinaciones
- Cómo se procesan las combinaciones de bucle anidado
- Presentación multimedia: Cómo se procesan las combinaciones de mezcla
- Consideraciones acerca del uso de combinaciones de mezcla
- Cómo se procesan las combinaciones hash

En esta sección describiremos cómo el optimizador consultas optimiza las consultas que utilizan operaciones de combinación.

Selectividad y densidad de una cláusula JOIN

Objetivo del tema
Describir la selectividad y la densidad de una cláusula JOIN.

Explicación previa
El orden en que el optimizador de consultas procesa las combinaciones viene determinado por la existencia de índices y una cláusula WHERE, además de la selectividad y densidad de los datos.



El orden en que el optimizador de consultas procesa las combinaciones viene determinado por la existencia de índices y una cláusula WHERE, además de la selectividad y densidad de los datos.

Sugerencia
Utilice la ilustración de la diapositiva para reiterar que una selectividad baja devuelve muchas filas y una selectividad alta devuelve menos.

Selectividad de una cláusula JOIN

La selectividad de una cláusula JOIN es el porcentaje de filas de una tabla que se combinan en una única fila de otra. La selectividad se deriva del número de filas previstas que se devolverán, como se ve con la cláusula WHERE.

Una selectividad baja devuelve muchas filas y una selectividad alta devuelve menos. La base es la multiplicación de las filas en ambas tablas después de aplicar los predicados locales (cláusula WHERE) en las agregaciones y tablas combinadas. Este algoritmo es distinto del que se utiliza para determinar cuántas filas cumplen una condición de búsqueda.

Cómo se determina la selectividad de una cláusula JOIN

Puede calcular la selectividad de una cláusula JOIN mediante la densidad de los datos. El optimizador de consultas determina la selectividad de una cláusula JOIN según los siguientes parámetros:

- Si las estadísticas están disponibles, la selectividad de combinación se basa en la densidad del índice para todas las columnas.
- Si las estadísticas no están disponibles porque no existen índices, los índices existentes no resultan útiles o si no se incluye una cláusula WHERE en la consulta, el optimizador de consultas procesa la misma de forma más eficiente mediante:
  - Aplicación de una estrategia de combinación adecuada.
  - Uso de otros operadores físicos.
  - Generación de estadísticas de columna dinámicamente.
  - El número de filas en cada tabla de la combinación.

Sugerencia
Señale las consideraciones que el optimizador de consultas estima para determinar la selectividad de una cláusula JOIN cuando no están disponibles las características.

Densidad de una cláusula JOIN

La densidad de una cláusula JOIN es el porcentaje promedio de duplicados entre las tablas internas y externas. El optimizador de consultas utiliza la densidad de una cláusula JOIN para determinar la tabla que se procesará como interna y la que se procesará como externa.

- Un índice con un gran número de duplicados tiene una alta densidad de combinación, lo que no resulta muy selectivo para las combinaciones.

  Por ejemplo, la tabla **orders_details** contiene muchos pedidos para un solo cliente.

- Un índice único tiene una baja densidad de combinación, por lo que es altamente selectivo.

  Por ejemplo, la tabla **customer** sólo enumera una vez cada cliente. La columna **customer ID** es única.

Si un índice tiene una baja densidad de combinación, el optimizador puede tener acceso a los datos mediante un índice agrupado o no agrupado. Sin embargo, sólo un índice agrupado normalmente resulta útil para índices con una alta densidad de combinación.

Ejemplo

En este ejemplo utilice los siguientes supuestos para determinar cómo el optimizador de consultas produce un plan de ejecución:

- La tabla **employee** contiene 1.000 filas.
- La tabla **department** contiene 100 miembros filas (departamentos únicos).
- Los datos están distribuidos de forma uniforme (10 empleados por departamento).
- No hay índices ni estadísticas.

```
USE credit
SELECT *
FROM department AS dept INNER JOIN employee AS empl
ON dept.deptno = empl.deptno
```

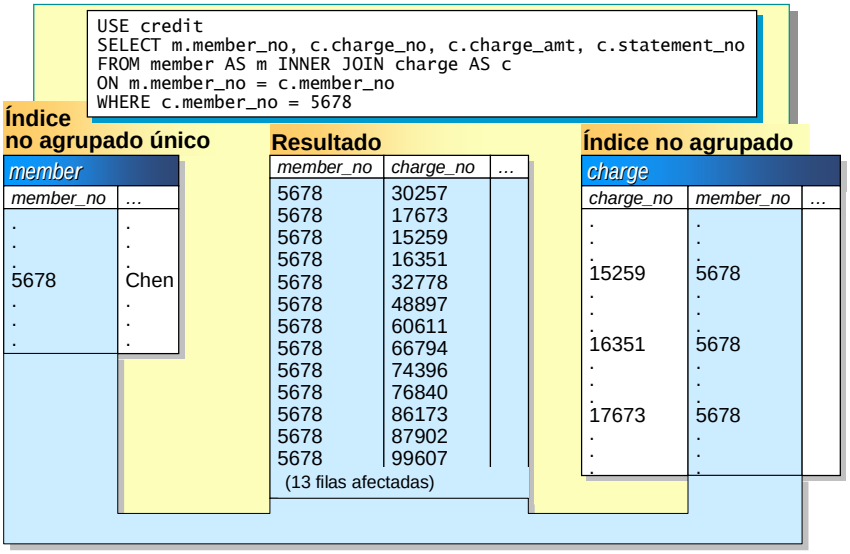
Cuando no existen índices en las columnas que están combinadas, el optimizador de consultas utiliza una estrategia de combinación que determina la tabla externa y la tabla interna. Lo realiza mediante la evaluación de la proporción de filas entre las tablas.

Si hay alguna condición de búsqueda en la cláusula WHERE, el optimizador de consultas puede utilizarlas en primer lugar para determinar cómo combinar las tablas. Esta determinación se basa en la selectividad.

Cómo se procesan las combinaciones

Objetivo del tema
Ilustrar cómo el optimizador de consultas procesa una combinación.

Explicación previa
Veamos cómo el optimizador de consultas procesa una combinación.



Sugerencia
Indique que el optimizador de consultas convierte los criterios de búsqueda en la cláusula WHERE de modo que la tabla **member** sea la tabla externa.

Explique que, mediante esta conversión, el optimizador de consultas limita la búsqueda. Limita la búsqueda porque la tabla **member** sólo tiene una fila que cumple la condición, mientras que la tabla **charge** tiene muchas filas.

La comprensión de cómo el optimizador de consultas procesa las operaciones de combinación le permitirá determinar los tipos de índices que resultan útiles crear.

Las combinaciones se procesan como parejas. Independientemente de las tablas que combine, las combinaciones siempre se realizan entre dos tablas. El resultado de estas combinaciones se denomina *resultado intermedio*. A continuación, los resultados intermedios se pueden combinar con otra tabla mediante cualquier algoritmo de combinación. Para cada combinación, el optimizador de consultas determina el algoritmo de combinación adecuado que se utilizará.

Al procesar operaciones de combinación, el optimizador de consultas normalmente:

- Determina el orden en que se procesan las tablas, basado en índices, selectividad y densidad.

  El orden no se determina por el orden de la tabla a la que se hace referencia en la instrucción SELECT.
- Identifica la tabla que resulta óptima como tabla externa.
- Busca todas las filas coincidentes en la tabla interna para cada fila que cumpla la condición en la tabla externa.

Evaluación del uso de índices

La selectividad y la densidad de una cláusula JOIN afecta al tipo de índice que resulta más útil para procesar la consulta.

- Un índice en la columna que está especificada en la cláusula WHERE puede influir en la tabla que se utiliza como tabla externa y la estrategia de combinación que se utiliza. La selectividad determina qué tabla es la interna.
- El optimizador de consultas considera automáticamente el uso de cláusulas JOIN redundantes y las condiciones en la cláusula WHERE.

Ejemplo

En este ejemplo hay un índice único no agrupado en la columna **member_no** en la tabla **member** y un índice no agrupado en la columna **member_no** en la tabla **charge**. Ambos índices resultan útiles para procesar la consulta.

```
USE credit
SELECT m.member_no, c.charge_no, c.charge_amt, c.statement_no
FROM member AS m INNER JOIN charge AS c
ON m.member_no = c.member_no
WHERE c.member_no = 5678
```

El optimizador de consultas convierte el criterio de búsqueda en la cláusula WHERE de modo que la consulta se procese como:

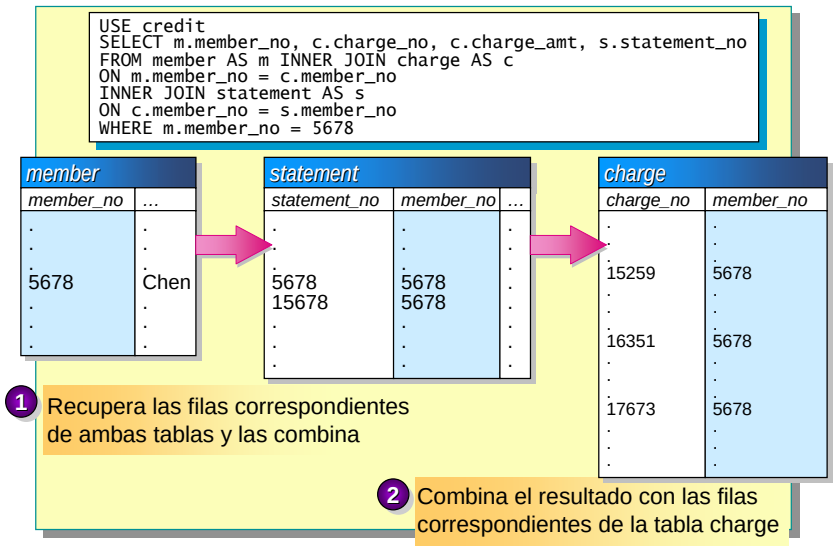
WHERE m.member_no = 5678

Al convertir la tabla **member** en la tabla externa, el optimizador de consultas limita la búsqueda, ya que la tabla **member** sólo tiene una única fila que cumple la condición, mientras que la tabla **charge** tiene muchas filas.

Cómo se procesan las combinaciones de bucle anidado

Objetivo del tema
Describir cómo se procesan las combinaciones anidadas.

Explicación previa
Si hay una cláusula JOIN en la consulta, el optimizador de consultas evalúa el número de tablas, índices y combinaciones para determinar el orden óptimo y la estrategia de combinación que se utilizará.



Si hay una cláusula JOIN en la consulta, el optimizador de consultas evalúa el número de tablas, índices y combinaciones para determinar el orden óptimo y la estrategia de combinación que se utilizará. El optimizador de consultas procesa las combinaciones de bucle anidado como iteraciones anidadas.

Definición de iteración anidada

Una *iteración anidada* se produce cuando el optimizador de consultas construye un conjunto de bucles y el conjunto de resultados crece a media que avanza por las filas. El optimizador de consultas realiza los siguientes pasos:

1. Busca una fila de la primera tabla.
2. Utiliza dicha fila para recorrer la tabla siguiente.
3. Utiliza el resultado de la tabla anterior para recorrer la tabla siguiente.

Evaluación de combinaciones

El optimizador de consultas evalúa al menos cuatro o más posibles combinaciones, incluso si no están especificas en el predicado de combinación. No tiene que agregar cláusulas redundantes. El optimizador de consultas equilibra el costo y utiliza estadísticas para determinar el número de combinaciones que evalúa. La evaluación de todas las combinaciones posibles es ineficiente y costosa.

Evaluación del costo del rendimiento de las consultas

Cuando el optimizador de consultas realiza una combinación debe tener en cuenta que se producen determinados costos. Las combinaciones de bucle anidado son superiores a las combinaciones de mezcla y a las combinaciones hash cuando se ejecutan transacciones pequeñas, como, por ejemplo, la que sólo afectan a un pequeño conjunto de filas. El optimizador de consultas:

- Utiliza combinaciones de bucle anidado si la entrada externa es bastante pequeña y la entrada interna está indizada y es bastante grande.
- Utiliza la entrada menor como la tabla externa.
- Requiere que exista un índice útil en el predicado de combinación para la tabla interna.
- Siempre utiliza una estrategia de combinación de bucle anidado si la operación de combinación utiliza un operador que no sea de igualdad.

Ejemplo

En este ejemplo, la tabla **member** (10.000 filas) se combina con la tabla **charge** (100.000 filas) y la tabla **charge** se combina con la tabla **statement** (20.000 filas). Hay índices no agrupados en la columna **member_no** en cada tabla. El optimizador de consultas procesa la combinación con la tabla **member** combinada con la tabla **statement** y el resultado de dicha combinación se combina con la tabla **charge**.

```
USE credit
SELECT m.member_no, c.charge_no, c.charge_amt, s.statement_no
FROM member AS m INNER JOIN charge AS c
      ON m.member_no = c.member_no
INNER JOIN statement AS s
      ON c.member_no = s.member_no
WHERE m.member_no = 5678
```

El optimizador de consultas realiza los siguientes pasos para procesar la consulta:

1. Recupera las filas que cumplen la condición de las tablas **member** y **statement**; a continuación, combina el resultado mediante la estrategia de combinación de bucle anidado.
2. Recupera las filas que cumplen la condición de la tabla **charge** y, a continuación, combina dicho resultado con el obtenido de la primera combinación de bucle anidado mediante otra estrategia de combinación de bucle anidado.

Cómo se procesan las combinaciones de mezcla

Objetivo del tema
Describir cómo se procesan las combinaciones de mezcla.

Explicación previa
En esta presentación, verá cómo se procesan las combinaciones de mezcla.

Las columnas de las condiciones de combinación se utilizan como entrada para procesar una combinación de mezcla. SQL Server lleva a cabo los pasos siguientes cuando utiliza una estrategia de combinación de mezcla:

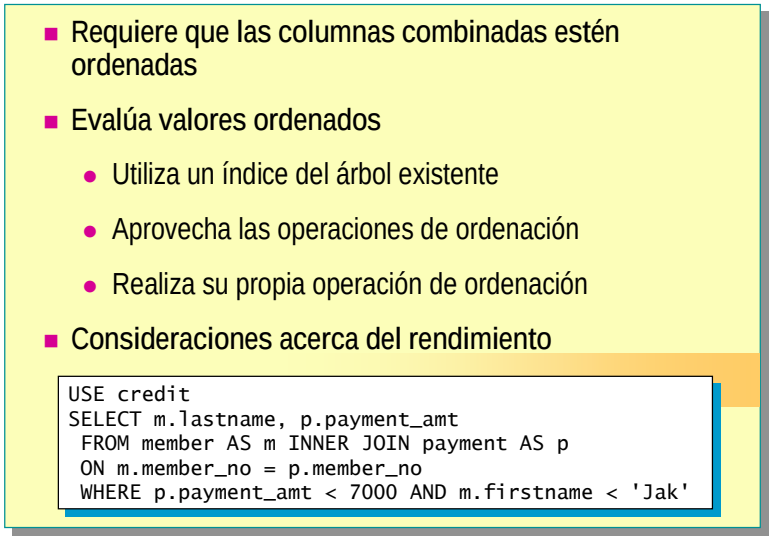
1. Obtiene los primeros valores de entrada de cada conjunto de entrada.
2. Compara los valores de entrada.
3. Realiza un algoritmo de mezcla.
   - Si los valores de entrada son iguales, se devuelven las filas.
   - Si son distintos, se descarta el valor menor y el siguiente valor de dicha entrada se utiliza para la siguiente comparación.
4. Repite el proceso hasta que se procesan todas las filas de uno de los conjuntos de entrada.
5. Evalúa las condiciones de búsqueda restantes de la consulta y sólo devuelve las filas que cumplen la condición.

Nota Sólo se realiza un paso por cada entrada. La operación de combinación de mezcla finaliza después de que se hayan evaluado todos los valores de una entrada. Los valores restantes de la otra entrada no se evalúan.

Consideraciones acerca del uso de combinaciones de mezcla

Objetivo del tema
Presentar los detalles de la operación de combinación de mezcla.

Explicación previa
Una combinación de mezcla utiliza dos entradas ordenadas y, a continuación, las mezcla.



Una mezcla utiliza dos entradas ordenadas y, a continuación, las mezcla.

Requiere que las columnas combinadas estén ordenadas

Si ejecuta una consulta con operaciones de combinación y las columnas combinadas están ordenadas, el optimizador de consultas procesa la consulta mediante una estrategia de combinación de mezcla. Una combinación de mezcla es muy eficiente ya que las columnas ya están ordenadas y requiere menos operaciones de E/S de página.

Evalúa valores ordenados

Para que el optimizador de consultas pueda utilizar la combinación de mezcla, las entradas deben estar ordenadas. El optimizador de consultas evalúa los valores ordenados en el orden siguiente:

1. Utiliza un índice del árbol existente (lo más habitual). El optimizador de consultas puede utilizar el índice del árbol de un índice agrupado o un índice no agrupado tratado.
2. Aprovecha las operaciones de ordenación que utilizan las cláusulas GROUP BY, ORDER BY y CUBE. La operación de ordenación sólo se tiene que efectuar una vez.
3. Realiza su propia operación de ordenación en la que aparece un operador SORT cuando se muestra gráficamente el plan de ejecución. El optimizador de consultas realiza esta operación en muy pocas ocasiones.

Ejemplo 1

En este ejemplo existe un índice agrupado en la columna **member_no** de la tabla **payment** y un índice agrupado único en la columna **member_no** de la tabla **member**. El optimizador de consultas recorre la tabla member y la tabla payment mediante el índice agrupado de cada tabla. Tras recorrer el contenido de cada tabla, el optimizador de consultas realiza una combinación de mezcla entre ambas tablas, ya que las dos entradas ya están ordenadas mediante los índices agrupados. Se trata de una combinación de mezcla/combinación interna.

```
USE credit
SELECT m.lastname, p.payment_amt
 FROM member AS m INNER JOIN payment AS p
      ON m.member_no = p.member_no
 WHERE p.payment_amt < 7000 AND m.firstname < 'Jak'
```

Ejemplo 2

En este ejemplo existe un índice agrupado único en la columna **member_no** de la tabla **member** y la consulta especifica de forma explícita una cláusula ORDER BY en la columna **member_no** de la tabla **payment**.

```
USE credit
SELECT m.lastname, m.firstname, p.payment_dt
  FROM member AS m INNER JOIN payment AS p
      ON m.member_no = p.member_no
  ORDER BY p.member_no
```

Consideraciones acerca del rendimiento

Tenga en cuenta los siguientes hechos acerca del uso que hace el optimizador de consultas de la combinación de mezcla:

- SQL Server realiza una combinación de mezcla para todos los tipos de operaciones de combinación (excepto las operaciones de combinación cruzadas o completas), incluidas las operaciones UNION.
- Una operación de combinación de mezcla puede ser una operación de uno a uno, uno a varios o varios a varios.

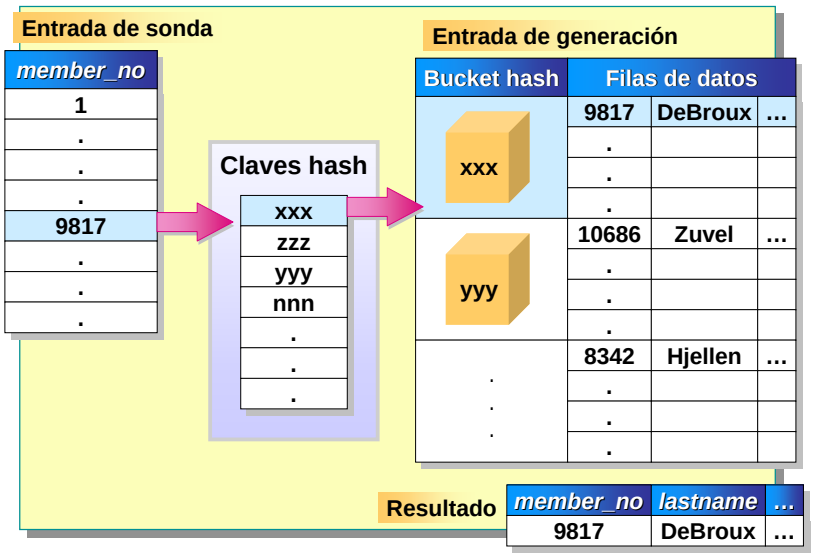
  Si la combinación de mezcla es una operación de varios a varios, SQL Server utiliza una tabla temporal para almacenar las filas. Si hay valores duplicados de cada entrada, una de las entradas vuelve al principio de los duplicados a medida que se procesa un valor duplicado de la otra entrada.
- El rendimiento de consulta de una combinación de mezcla es muy rápido, pero el costo puede ser alto si el optimizador de consultas debe realizar su propia operación de ordenación.

  Si el volumen de datos es grande y los datos deseados se pueden obtener ya ordenados de los índices de árbol B (árbol equilibrado) existentes, la combinación de mezcla suele ser el algoritmo de combinación más rápido.
- La combinación de mezcla normalmente se utiliza si las dos entradas de combinación tienen una gran cantidad de datos y están ordenadas en sus columnas de combinación (por ejemplo, si las entradas de combinación se han obtenido mediante el recorrido de índices ordenados).
- Las operaciones de combinación de mezcla sólo se pueden realizar con un operador de igualdad en el predicado de combinación.

Cómo se procesan las combinaciones hash

Objetivo del tema
Describir cómo se procesan las combinaciones hash.

Explicación previa
El optimizador de consultas utiliza una combinación hash cuando considera que esta opción es más eficiente que procesar las consultas mediante una combinación de bucle anidado o de mezcla.



La aplicación del algoritmo hash es una estrategia para dividir los datos en conjuntos iguales con un tamaño fácil de administrar según una determinada propiedad o característica. Los datos agrupados se pueden utilizar para determinar si un elemento de datos concreto coincide con un valor existente.

Nota Los datos duplicados o los intervalos de datos no resultan útiles para las combinaciones hash ya que los datos no están organizados de forma conjunta o en orden.

Cuándo se utiliza una combinación hash

El optimizador de consultas utiliza una combinación hash cuando considera que es más eficiente que procesar las consultas mediante una combinación de bucle anidado o de mezcla. Normalmente utiliza una combinación hash cuando no existe un índice o los índices existentes no resultan útiles.

Asigna una entrada de generación y sonda

El optimizador de consultas asigna una *entrada de generación* y *sonda*. Si el optimizador de consultas asigna incorrectamente la generación y entrada de sonda (puede suceder a causa de previsiones de densidad imprecisas), las invierte dinámicamente. Esta posibilidad de cambiar las funciones de entrada dinámicamente se denomina *inversión de funciones*.

La entrada de generación consta de los valores de columna de una tabla con el número menor de filas. La entrada de generación crea una *tabla hash* en memoria para almacenar estos valores.

El bucket hash es un lugar de almacenamiento en la tabla hash en que se inserta cada fila de la entrada de generación. Las filas de una de las tablas de combinación se colocan en el bucket hash donde el valor clave hash de la fila coincide con el valor clave hash del bucket. Los buckets hash se almacenan como una lista vinculada y sólo contienen las columnas que son necesarias para la consulta.

Una tabla hash contiene buckets hash. La tabla hash se crea a partir de la entrada de generación.

La entrada de sonda consta de los valores de columna de la tabla que tiene más filas. La entrada de sonda es lo que comprueba la entrada de generación para buscar una coincidencia en los buckets hash.

Nota El optimizador de consultas utiliza estadísticas de columna o índice para determinar cuál de las dos entradas es la menor.

Procesamiento de una combinación hash

La lista siguiente es una descripción simplificada de cómo el optimizador de consultas procesa una combinación hash. No pretende ser extensa ya que el algoritmo es muy complejo. SQL Server:

1. Lee la entrada de sonda. Cada entrada de sonda se procesa fila a fila.
2. Realiza el algoritmo hash en cada entrada de sonda y genera un valor clave hash.
3. Busca el bucket hash que coincide con el valor clave hash.
4. Obtiene acceso al bucket hash y busca la fila que cumple la condición.
5. Devuelve la fila si encuentra una coincidencia.

Consideraciones acerca del rendimiento

Tenga en cuenta los siguientes hechos acerca de las combinaciones hash que utiliza el optimizador de consultas:

- Al igual que las combinaciones de mezcla, una combinación hash es muy eficiente ya que utiliza buckets hash, que son como un índice dinámico pero con menos carga de trabajo para combinar filas.
- Las combinaciones hash se pueden realizar para todos los tipos de operaciones de combinación (excepto las operaciones de combinación cruzadas), incluidas las operaciones UNION y DIFFERENCE.
- Un operador hash puede quitar duplicados y agrupar datos, como SUM (salary) GROUP BY department. El optimizador de consultas sólo utiliza una entrada para las funciones de generación y sonda.
- Si las entradas de la combinación son grandes y de tamaño similar, el rendimiento de la operación de combinación hash es parecido a una combinación de mezcla con ordenación anterior. Sin embargo, si el tamaño de las entradas de la combinación es ligeramente distinto, el rendimiento de una combinación hash es, a menudo, mucho más rápido.

- Las combinaciones hash pueden procesar, de forma eficiente, entradas grandes, sin ordenar y sin indizar. Las combinaciones hash resultan útiles en consultas complejas porque los resultados intermedios:
  - No están indizados (a menos que se guarden directamente en disco y se indicen posteriormente).
  - No suelen estar ordenados para la siguiente operación del plan de ejecución.
- El optimizador de consultas puede identificar las previsiones incorrectas y realizar correcciones dinámicamente para procesar la consulta más eficientemente.
- Una combinación hash reduce la necesidad de *desnormalización* de la base de datos. La desnormalización normalmente se utiliza para obtener mejor rendimiento mediante la reducción de las operaciones de combinación a pesar de la redundancia, como las actualizaciones incoherentes. Las combinaciones hash permiten crear particiones verticales de los datos como parte del diseño de la base de datos física. Las particiones verticales representan grupos de columna de una única tabla en archivos o índices independientes.

Nota Para obtener información acerca de las combinaciones hash, busque "explicación de las combinaciones hash" en los Libros en pantalla de SQL Server.