Abstract: Como usar indices Bitmap Join

Autor: Martín Cabarique

Entre los mejores métodos de acceso a la información se encuentra el uso de los índices de tipo Bitmap Join. Este nuevo método de acceso requiere crear un índice de tipo bitmap sobre columnas que no son de la tabla indexada.

Para obtener dicho índice se realiza el join al momento de la creación del índice, entre la tabla base y las tablas que contienen la información de las columnas indexadas. El resultado es que tenemos almacenado en el índice el resultado del join correspondiente, mejorando los tiempos de consulta en índices superiores al 600%.

Esta técnica es muy poderosa para joins entre tablas sobre columnas con muy baja cardinalidad (Ej: menos de 300 valores distintos). Dado que el índice creado es de tipo bitmap, este deberia ser usado solo sobre tablas de consulta y no de tipo OLTP, dado el costo de las actualizaciones y la limitación en la concurrencia de los usuarios que modifican.

Bitmap Join Indexes en Acción

Echemos un vistazo más de cerca a la forma como este tipo de índices funciona.

Para mostrar el concepto, usare un sencillo modelo entidad relación entre tablas con una relación muchos-a-muchos, correspondiente a un Datawarehouse de ventas. Se cuenta con las tablas clientes, productos y una tabla de ventas que es el resultado de la relación muchos-a-muchos entre estas ultimas (Ilustración 1).

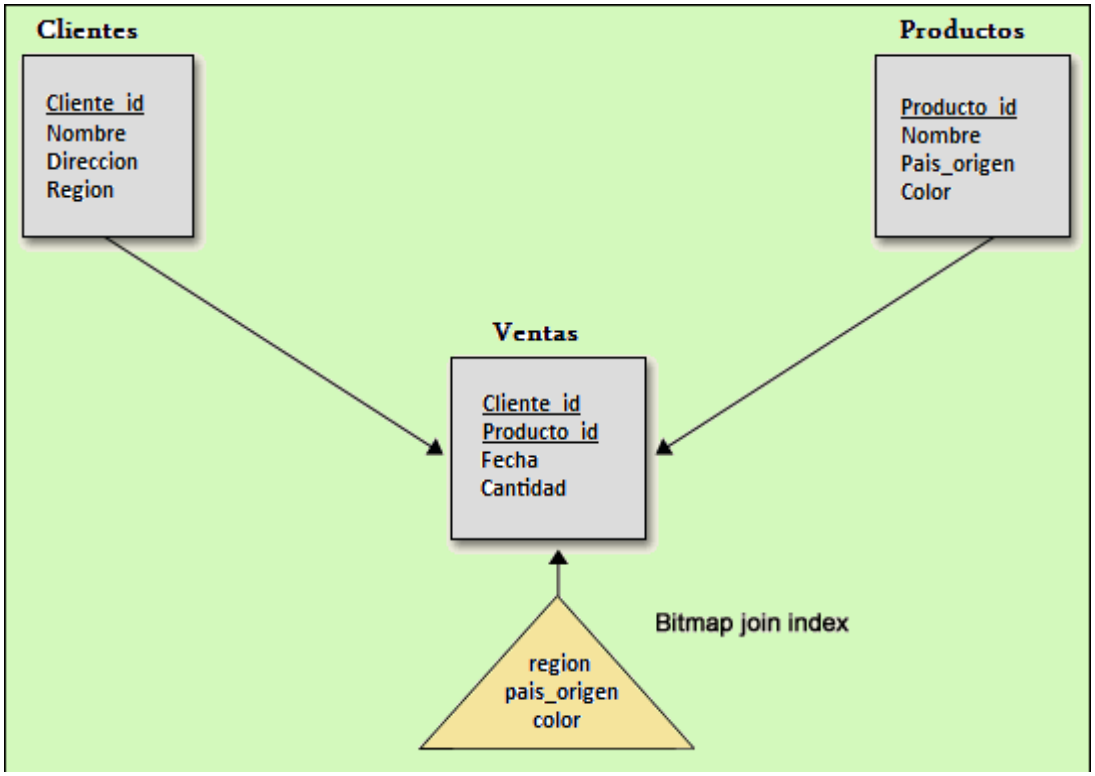


Ilustración 1: Modelo Entidad Relación

Las regiones son solamente 4: Sierra, Costa, Oriente, Galápagos.
Los países de origen son en su mayoría solo 5 valores: Ecuador, China, USA, Colombia y Perú.
Los colores son menos de 100 valores distintos.

Como puede observarse la tabla de ventas tiene mucha información, correspondiente a cada transacción de ventas realizada mientras que las tablas cliente y productos tienen pocos valores en comparación de dicha tabla. Para el caso de nuestra prueba, se crean los objetos con el siguiente script:

```
CREATE TABLE PRODUCTOS (
        PRODUCTO_ID NUMBER PRIMARY KEY,
        NOMBRE VARCHAR2(30),
        PAIS_ORIGEN VARCHAR2(15),
        COLOR VARCHAR2(10)
   );

CREATE TABLE CLIENTES (
        CLIENTE_ID NUMBER PRIMARY KEY,
        NOMBRE  VARCHAR2(30),
        DIRECCION VARCHAR2(15),
        REGION VARCHAR2(10)
  );

CREATE TABLE VENTAS
  (     CLIENTE_ID NUMBER,
        PRODUCTO_ID NUMBER,
        CANTIDAD NUMBER,
        FECHA DATE
  );

ALTER TABLE VENTAS
        ADD CONSTRAINT
        FOREIGN KEY (CLIENTE_ID) REFERENCES CLIENTES(CLIENTE_ID);

ALTER TABLE VENTAS
        ADD CONSTRAINT
        FOREIGN KEY (PRODUCTO_ID) REFERENCES PRODUCTOS(PRODUCTO_ID);
```

Para poblar las tablas con datos de prueba se usa el siguiente script:

```
-- Inserta 10000 filas en la tabla clientes
declare
  i number := 0;
begin
  for i in 1 .. 10000 loop
    insert into clientes values (i,'cliente' || i, 'direccion ' || i,
      decode(mod(i,4), 0, 'sierra', 1, 'costa', 2, 'oriente', 3, 'galapagos'));
  end loop;
end;

-- Inserta 20000 filas en la tabla productos
declare
  i number := 0;
begin
  for i in 1 .. 20000 loop
    insert into productos values (i,'producto' || i,
      decode(mod(i,4), 0, 'usa', 1, 'ecuador', 2, 'colombia', 3, 'peru'),
      decode(mod(i,5), 0, 'blanco', 1, 'amarillo', 2, 'verde', 3, 'azul', 4, 'rojo')
      );
  end loop;
end;

-- Inserta 2'0000000 filas en la tabla ventas
declare
  i number := 0;
  j number := 0;
begin
  for i in 1 .. 20000 loop
```

```
    for j in 1 .. 10 loop
    insert into ventas values (j,i, j, sysdate);
  end loop;
  end loop;
end;
```

Ahora veamos nuestro indice en funcionamiento. Para ello consultemos cuantas ventas fueron realizadas en la region 'sierra', cuya procedencia es de USA y cuyo color es 'rojo'.

```
Select count(*)
from ventas v, clientes c, productos p
where  v.cliente_id=c.cliente_id
          and  v.producto_id=p.producto_id
          and p.pais_origen = 'usa'
          and c.region = 'sierra'
        and p.color = 'rojo';
```

Su plan de ejecución sin el indice bitmap es el siguiente:

Operation	Optimizer	Cost
SELECT STATEMENT	ALL_ROWS	257
SORT(AGGREGATE)		
HASH JOIN		257
TABLE ACCESS(FULL) SCOTT.CLIENTES	ANALYZED	19
HASH JOIN		237
TABLE ACCESS(FULL) SCOTT.PRODUCTOS	ANALYZED	30
TABLE ACCESS(FULL) SCOTT.VENTAS	ANALYZED	206

Ilustración 2: Plan de Ejecución sin Bitmap Join Index

Observamos que su costo es de 257 y usa como método de ejecución el HASH JOIN.

Ahora veamos su comportamiento cuando creamos el bitmap index con el siguiente script:

```
CREATE BITMAP INDEX REGION_PAIS_COLOR
          ON VENTAS( C.REGION, P.PAIS_ORIGEN, P.COLOR)
          FROM  VENTAS V,
                    CLIENTES C,
                    PRODUCTOS P
          WHERE  V.CLIENTE_ID=C.CLIENTE_ID
                    AND  V.PRODUCTO_ID=P.PRODUCTO_ID;
```

Su nuevo plan de ejecución es ahora:

Operation	Optimizer	Cost
SELECT STATEMENT	ALL_ROWS	1
SORT(AGGREGATE)		
BITMAP CONVERSION(COUNT)		1
BITMAP INDEX(SINGLE VALUE) SCOTT.REGION_PAIS_COLOR		

Ilustración 3: Plan de Ejecucion con Bitmap Join Index

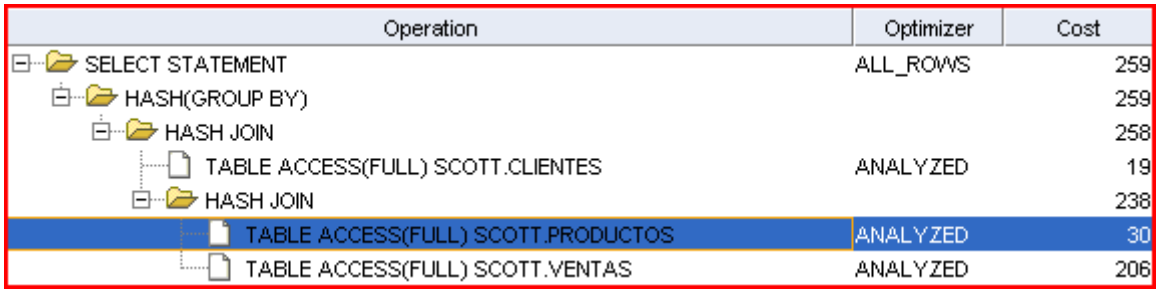
Vemos como en este caso no se requiere leer las tablas sino que la consulta es enteramente resuelta usando el indice, teniendo un costo de 1.

Otro posible ejemplo es consultar las ventas diarias para productos con un origen, color y región específicos. La consulta sería:

```
Select trunc(fecha), sum(cantidad)
from ventas v, clientes c, productos p
where  v.cliente_id=c.cliente_id
          and  v.producto_id=p.producto_id
          and p.pais_origen = 'usa'
```

```
        and c.region = 'sierra'
    and p.color = 'rojo'
group by trunc(fecha);
```

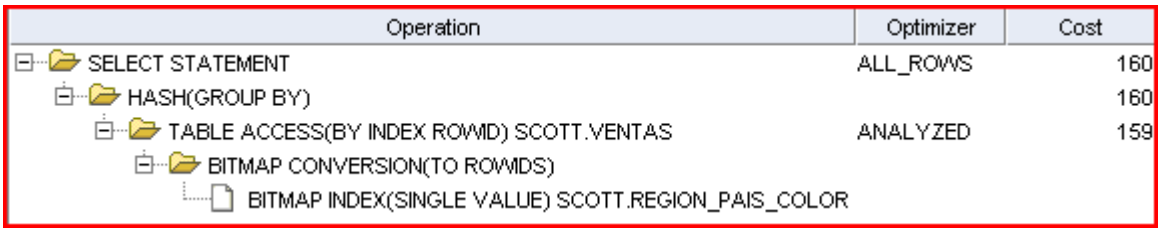
Su plan de ejecución sin indice es:



Operation	Optimizer	Cost
SELECT STATEMENT	ALL_ROWS	259
HASH(GROUP BY)		259
HASH JOIN		258
TABLE ACCESS(FULL) SCOTT.CLIENTES	ANALYZED	19
HASH JOIN		238
TABLE ACCESS(FULL) SCOTT.PRODUCTOS	ANALYZED	30
TABLE ACCESS(FULL) SCOTT.VENTAS	ANALYZED	206

Ilustración 4: Plan de ejecución sin Bitmap Index

Ahora veamos como se ejecutaría una vez creado el bitmap index:



Operation	Optimizer	Cost
SELECT STATEMENT	ALL_ROWS	160
HASH(GROUP BY)		160
TABLE ACCESS(BY INDEX ROWID) SCOTT.VENTAS	ANALYZED	159
BITMAP CONVERSION(TO ROWIDS)		
BITMAP INDEX(SINGLE VALUE) SCOTT.REGION_PAIS_COLOR		

Ilustración 5: Plan de Ejecución con Bitmap Index

En este caso observamos que las tablas productos y clientes no son leídas pues la información del join esta en el índice, requiriéndose solo leer a la tabla ventas.

CONCLUSIONES

Los bitmap join indexes son estructuras poderosas para acelerar el proceso de consulta sobre tablas que se consultan juntas (join) y cuyos atributos son de baja cardinalidad. Para hacer uso de los mismos es importante un análisis de las consultas y el modelo de tablas de manera que asegure la correcta creación de los mismos. No se recomienda su uso en ambientes OLTP salvo un estudio detallado que garantice que sus limitaciones de concurrencia y desempeño en las actualizaciones no sean un factor crítico a las aplicaciones.