Desempeño de consultas SQL relacionales y objeto-relacionales en Oracle

The performance of relational and object-relational SQL queries when using Oracle

Francisco Javier Moreno,[1] Guillermo Ospina Romero,[2] Rafael Larios Restrepo[3]

RESUMEN

En este artículo se muestra por medio de consultas específicas el comportamiento del modelo relacional y objeto-relacional, y se presenta así un estudio en el que se mide y a la vez se compara la eficiencia (tiempo, uso de recursos del sistema) de operaciones que involucran cláusulas GROUP BY, subconsultas (con las cláusulas IN y EXISTS), y reuniones (*joins*).

Palabras clave: bases de datos relacionales y objeto-relacionales, optimización, SQL, Oracle.

ABSTRACT

This article presents a study in which the performance of several queries involving GROUP BY clause, sub-queries (using IN and EXISTS clauses) and joins are measured and compared by means of specific queries. The behaviour of relational and object-relational models is also shown.

Key words: relational and object-relational databases, tuning, SQL, Oracle.

Recibido: mayo 5 de 2005
Aceptado: septiembre 23 de 2005

Introducción

Buscar la optimización de las consultas para reducir tiempos de respuesta o evitar el uso exagerado de recursos del sistema, se ha constituido en un objetivo durante los últimos años. En muchas consultas, la realización de reuniones (*joins*), agrupamientos y subconsultas entre tablas, es indispensable para obtener la información solicitada.

El SGDB Oracle soporta el modelo relacional y el objeto-relacional a partir de la versión 8i (Oracle Corporation, 2003), sin embargo poco se sabe cuándo es mejor utilizar cada uno de ellos (Bodnar, 2000; Raghavan, 1999), es por esto que se procederá a analizar casos que proporcionarán una base para entender y aplicar de una mejor manera, las consultas que se construyan en uno u otro modelo (tablas relacionales, frente a tablas basadas en tipos y tablas anidadas del modelo objeto-relacional).

Para lograr tal análisis, Oracle provee varias herramientas que permiten observar cómo se comporta la ejecución de una determinada consulta. Estas herramientas son el EXPLAIN PLAN, SQL TRACE y el Tkprof. Estas herramientas entregan información detallada acerca de la ejecución de una consulta (tales como bloques leídos durante la ejecución de la consulta, filas procesadas, uso de la cpu, entre otros), la cual ayuda en gran manera a evaluar su desempeño. Sin utilizar estas herramientas no sería posible analizar cómo se comporta internamente la consulta, es decir, no se dispondría de información suficiente para conocer en qué forma se debe modificar una consulta para hacerla más eficiente.

1 M.Sc. Universidad Nacional de Colombia Sede Medellín, profesor asociado a la Universidad Nacional Sede Medellín, e-mail: fjmoreno@unalmed.edu.co
2 Aspirante a título de Ingeniería en Sistemas en la Universidad Nacional Sede Medellín, e-mail: glospina@unalmed.edu.co
3 Aspirante a título de Ingeniería en Sistemas en la Universidad Nacional Sede Medellín, rilarios@unalmed.edu.co

El SGBD Oracle también ofrece la oportunidad de modificar la forma en que se ejecutan internamente las consultas, esto se logra mediante la utilización de palabras clave dentro de la sentencia a ejecutar. Estas palabras clave son llamadas *hints*. A través de su utilización se puede influir en la ejecución parcial o total de la consulta con el objetivo de mejorar posiblemente el rendimiento de la misma.

El SGBD Oracle posee un optimizador interno que le permite definir un plan de ejecución para una consulta. Este optimizador, sin embargo, no siempre define el plan más óptimo. Es en estos casos en donde la utilización de los hints se hace necesaria para la búsqueda de consultas optimizadas.

El SGBD soporta una gran cantidad de *hints* para forzar al optimizador a realizar diferentes acciones específicas que ayudan al proceso de optimización (entre ellas se encuentran el forzar o no la utilización de índices, selección del método para realizar una reunión, entre otros); sin embargo, no es el objetivo de este artículo mostrar la variedad de *hints* que soporta el SGBD Oracle, sino mostrar y utilizar aquellos que apoyan el enfoque del artículo. Un análisis detallado puede verse en (Nyffenegger, 2001)

El artículo se concentra en el comportamiento del SGBD Oracle en algunas consultas específicas aplicadas en un modelo relacional y objeto-relacional. El cuerpo del artículo es el siguiente: en la sección 2 se presentan las tasas de rendimientos a analizar en el desempeño de una consulta, en la sección 3 los métodos de reunión disponibles en Oracle, en la sección 4 los modelos a utilizar, en la sección 5 los resultados, y finalmente, en la sección 6, las conclusiones y trabajos futuros. Todos los ejemplos y ejercicios se desarrollaron mediante la utilización de la herramienta JDeveloper de Oracle y la herramienta SQL*Plus, la interfaz de línea de comandos por defecto de Oracle.

Tasas utilizadas para medir el desempeño de una consulta

Mediante la herramienta Tkprof se puede obtener un archivo de salida como el mostrado en la Figura 1. A partir de él se analizan las siguientes tasas (Harrison, 2000):

1. Bloques leídos (f+g) sobre filas procesadas (h). Esta tasa indica de una manera general el costo relativo de la consulta. Mientras más bloques tienen que ser accedidos en relación en las filas retornadas, la fila traída será mucho más costosa. Una relación similar se puede deducir sobre la tasa de bloques leídos sobre ejecuciones (f+g)/e. El valor que se debe procurar para esta tasa debe ser menor a 10, sin embargo tasas con valores de 10 a 20 son aceptables. Tasas por encima de 20 pueden indicar alguna posibilidad de optimización en este campo.

2. Parsing (d), sobre ejecución (e). Idealmente, el conteo de *parsing* (análisis sintáctico) debe ser cercano a uno. Si este valor es alto en relación al conteo de ejecuciones, la sentencia entonces ha sido "parseada" varias veces sin necesidad, lo cual indica que puede haber problemas en el tamaño del shared pool[4] (muy pequeño). Otra razón para que esto ocurra es que es posible que no se utilicen variables tipo bind[5] en la sentencia (Niemiec, 2005).

3. Filas traídas (i) sobre traídas (j) (Rows Fetched over fetches). Esta tasa indica el nivel en el cual la capacidad del *array fetch*[6] ha sido utilizada. Una tasa cercana a uno, indica que hubo poco procesamiento a través de *arrays*, lo que significa que existe una buena oportunidad para optimizar.

4. Lecturas de Disco (k) sobre lecturas lógicas (f+g). Esta es una tasa de error (*miss rate*) dentro del *buffer* de datos en la zona de caché, es decir, es una tasa que muestra el porcentaje de ocasiones en el que SGBD no ha encontrado las filas solicitadas en el *buffer* de la zona de caché y por lo tanto ha tenido que recurrir a traer los bloques desde disco. Generalmente se busca que esta tasa no represente más de un 10%.

```
call        count   cpu   elapsed  disk  query  current   rows
____        ___     ___   ___ ___ ___ ____ ___

        Parse(a)     (d)     –      –     –      –       –      –
        Execute(b)   (e)     –      –     –      –       –      –
        Fetch(c)     (j)     –      –     –      –       –      (i)
        ____  ___ ___ ___ ___ ___ ____ ___

        Total        –     –      –     (k)    (f)     (g)     (h)
```

Figura 1. Estructura de la salida del TKPROF

Métodos de reunión (*join*) y *Hints*

Se realizará una comparación de los diferentes métodos de reunión que se encuentran disponibles en Oracle. Los métodos a evaluar son: *Hash, Nested Loops y Sort Merge* (Date, 2001).

Hash: Este método utiliza una tabla de HASH basada en las tablas implicadas en la reunión.

Nested Loops: En este método se utilizan ciclos anidados para la lectura de las tablas. Por cada elemento (tupla o bloque) de la tabla externa se leen todos los elementos (tuplas o bloques) de la tabla interna.

Sort Merge: En este método, las tablas, a las cuales se les efectuará la reunión, pasan por un proceso de ordenamiento previo a la reunión.

4 El *shared pool* (Nyffenegger, 2000) es una zona del SGBD Oracle donde se encuentran almacenados: planes de consultas optimizados, sentencias SQL parseadas e información de objetos, entre otros.

5 Debido a que las sentencias SQL se guardan en memoria, cuando se utilizan variables tipo *bind* en una consulta, estas permiten que a la misma no se le haga *parsing* cada vez que se ejecute ("la sentencia no cambia"), de manera que así libera recursos útiles (Kyte, 2000).

6 Array Fetch es la característica que posee el SGBD Oracle, que le permite traer más de una fila de la consulta por cada *fetch* realizado (Harrison, 2000).

Por defecto, el SGBD tomará en cuenta el camino de ejecución (*Execution Path*) determinado por el optimizador interno, y de acuerdo con este, seleccionará uno de los métodos de *join* para ejecutarlo. Por medio de la utilización de *Hints*, se puede forzar a que el SGBD ejecute una sentencia específica con un método deseado y por lo tanto, sea posible comparar las tasas de rendimiento antes mencionadas. Para utilizar los *Hints*, es necesario introducirlos dentro de la sentencia a ejecutar. La sintaxis para ello es:

SELECT /*+ [HINTS]*/ [columnas] FROM…

Los *Hints* que se utilizarán para las pruebas de comparación serán: (Oracle Corporation[2], 2003).

• USE_MERGE(nombre_tabla): Fuerza al optimizador a utilizar el método Sort Merge.

• USE_HASH(nombre_tabla): Fuerza al optimizador a utilizar el método Hash.

• USE_NL(nombre_tabla): Fuerza al optimizador a utilizar el método Nested Loops.

Sin embargo, debido a que en algunas consultas objeto-relacionales, en especial cuando se usan REFs (punteros), el nombre de la tabla no está disponible, se utilizará su nombre interno (alias) el cual es provisto por Oracle. Este nombre interno para las tablas puede ser obtenido a través del siguiente método (sólo funciona desde la versión Oracle 10 g en adelante):

1. Realizar un EXPLAIN PLAN[7] sobre la consulta especiífica

2. Efectuar la siguiente consulta:

SELECT plan_table_output

FROMTABLE(DBMS_XPLAN.DISPLAY ('PLAN_TABLE',NULL,'ALL'));

La función DBM-S_XPLAN.DISPLAY acepta tres parámetros:

table_name El nombre de la tabla donde se guardan los datos del EXPLAIN PLAN. El valor por defecto es 'PLAN_TABLE'.

statement_id Id del plan de la sentencia a mostrar, el valor por defecto es NULL.

Format Controla el nivel de detalle a mostrar, el valor por defecto es 'TYPICAL'. Entre otros valores se encuentran 'BASIC', 'ALL' y 'SERIAL'.

Esta consulta trae (además de otros valores) el alias de las tablas involucradas en las sentencias encontradas en la tabla del EXPLAIN PLAN.

Un ejemplo: Si se obtiene que los alias para dos tablas son: B@SEL$1 y P000003$@SEL$1, y se desea reunirlas mediante el método Sort Merge, la sentencia a utilizar será:

SELECT /*+ USE_MERGE(B@SEL$1 P000003$@SEL$1)*/ [columnas] FROM …

Modelos a utilizar

A continuación se compararán los modelos relacional y objeto-relacional (incluyendo el uso de tablas anidadas). Se analizarán estos modelos basándose en casos particulares, entre los cuales se posee un primer modelo de seis tablas donde se encuentran datos sobre vendedores, facturas, detalles, libros, editoriales y categorías.

Adicionalmente se analizará para una consulta especial, un segundo modelo con cinco tablas correspondientes a continentes, países, regiones, departamentos y municipios (véanse los modelos a utilizar en la Figura 2).

Se utilizarán para algunos casos y en ambos modelos, tablas con "muchos datos" y tablas más pequeñas (aproximadamente 10% del tamaño de las "grandes", a excepción de las tablas vendedor, categoría y editorial). La Tabla 1 muestra el tamaño de las tablas del Modelo 1 y la Tabla 2 muestra el tamaño de las tablas del Modelo 2.

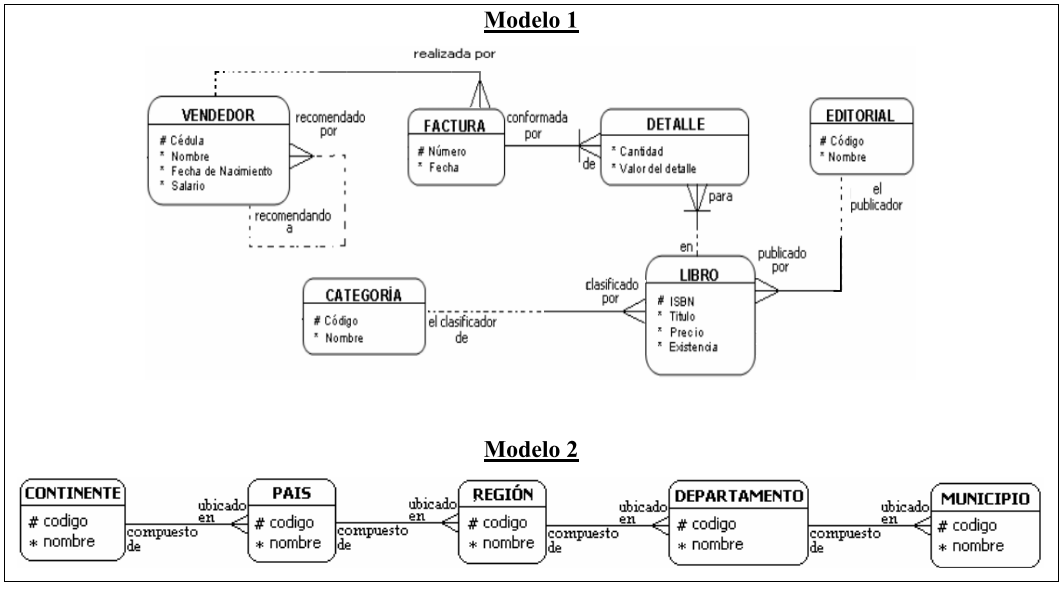


Figura 2: Modelos Entidad-Relación

7 El EXPLAIN PLAN es una herramienta proporcionada por Oracle que permite observar el plan de ejecución de una sentencia. Una descripción detallada de su uso puede verse en (Arnoff, 1995).

Tabla 1. Tamaño de las tablas a utilizar del modelo 1.

Tablas	Muchos Datos	Pocos Datos
Vendedor	192	192
Factura	12000	1200
Detalle	23481	2363
Libro	20000	2000
Categoría	50	50
Editorial	100	100

Tabla 2. Tamaño de las tablas a utilizar del modelo 2.

Tablas	Cantidad
Continente	5
País	200
Región	1000
Departamento	10000
Municipio	1000000

Resultados

A continuación se presentan los resultados obtenidos en ambos modelos; se comienza con los resultados obtenidos de las consultas realizadas sobre el Modelo 1, que comprenden pruebas de tipo GROUP BY, subconsultas y reuniones.

Pruebas con consultas de tipo GROUP BY

Los resultados que involucran la cláusula GROUP BY se dividen en dos. En la primera parte se mostrarán los resultados para una agrupación sencilla y luego se mostrarán los resultados cuando se utilizan las cláusulas de HAVING y ORDER BY.

Resultados para GROUP BY sencillo

Las consultas utilizadas se muestran a continuación:

Para el modelo relacional

```
SELECT    nombre,  COUNT(*)
FROM      libro 1, categoria c
WHERE     1.cod_cate  =  c.código
GROUP     BY  nombre
```

Para el modelo objeto relacional

```
SELECT    1.ref_cate.nombre,  COUNT(*)
FROM      libro 1
GROUP     BY  1.ref_cate.nombre
```

Se debe recordar que las pruebas se realizaron con tablas con muchos y pocos datos para ambos modelos. Nótese el uso de los punteros para la consulta sobre el modelo objeto relacional. En la Tabla 3 se presentan los resultados para las pruebas realizadas

Tabla 3. Resultados para las consultas de GROUP BY sencillo.

TASA	(f + g) / h	d / e	i / j	k / (f + g)	DATOS
VALOR NORMAL	Menor que 10	1 (o cercano a 1)	Mientras mayor mejor	Menor del 10%	
MODELO RELACIONAL	3.82	1	10	0.03141361	Muchos
	0.6	1	10	0.2	Pocos
MODELO OBJETO-RELACIONAL	13.14	1	10	0.00913242	Muchos
	1.98	1	10	0.72727272	Pocos

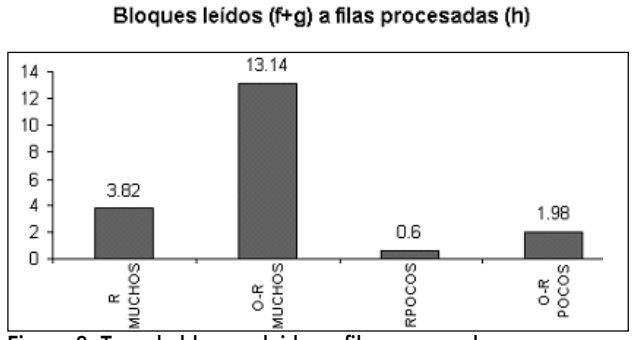


Figura 3. Tasa de bloques leidos a filas procesadas para GROUP BY sencillo

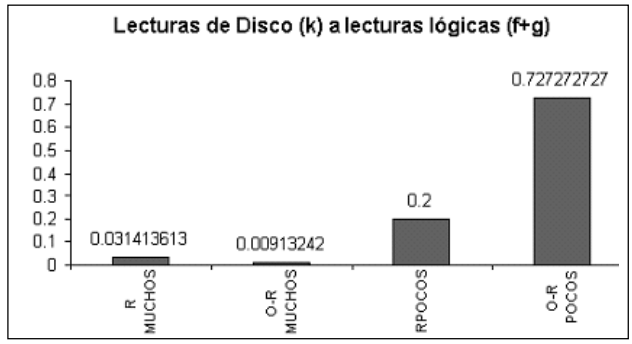


Figura 4. Tasa de lecturas de disco a lecturas lógicas para GROUP BY sencillo

Como se puede observar en la Figura 3, el modelo objeto-relacional se comporta con un rendimiento más bajo con respecto al modelo relacional, tanto con muchos como con pocos datos (aunque a medida que aumentan los datos, el rendimiento del modelo objeto-relacional mejora).

En la Figura 4, se observa que el modelo relacional aventaja al modelo objeto-relacional en la tasa de lecturas de disco a lecturas lógicas cuando se efectúan sobre pocos datos, sin embargo, al aumentar su cantidad, se observa que el modelo objeto-relacional aventaja al modelo relacional.

Resultados para GROUP BY cuando se utilizan las cláusulas HAVING y ORDER BY

Las consultas utilizadas se muestran a continuación:

Para el modelo relacional

```
SELECT    e.nombre,  count(*)
FROM      1ibro 1, editorial  e
WHERE     1.cod_edit  =  e.codigo
```

```
GROUP BY        e.nombre
HAVING          count(*) > 200
ORDER BY        e.nombre
```

Para el modelo objeto relacional

```
SELECT          l.ref_edit.nombre, count(*)
FROM            libro l
GROUP BY        l.ref_edit.nombre
HAVING          count(*) > 200
ORDER BY        l.ref_edit.nombre
```

En la Tabla 4 se presentan los resultados para las pruebas realizadas.

Tabla 4: Resultados para las consultas de GROUP BY con las cláusulas HAVING y ORDER BY.

TASA	(f + g) / h	d / e	i / j	k / (f + g)	DATOS
VALOR NORMAL	Menor que 10	1 (o cercano a 1)	Mientras mayor mejor	Menor del 10%	
MODELO RELACIONAL	4.340909091	1	11.25	0.97905759	Muchos
	0.581395349	1	11.25	0.92	Pocos
MODELO OBJETO-RELACIONAL	14.6	1	11.25	0.00761035	Muchos
	2.152173913	1	11.25	0	Pocos

En la tasa de bloques leídos a filas procesadas (Figura 5), se observa de nuevo la ventaja que posee el modelo relacional con respecto al modelo objeto-relacional especialmente cuando se ejecuta sobre muchos datos.

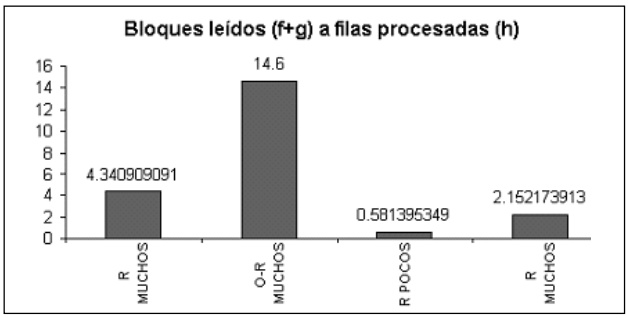


Figura 5. Tasa de bloques leídos a filas procesadas para GROUP BY con las cláusulas HAVING y ORDER BY

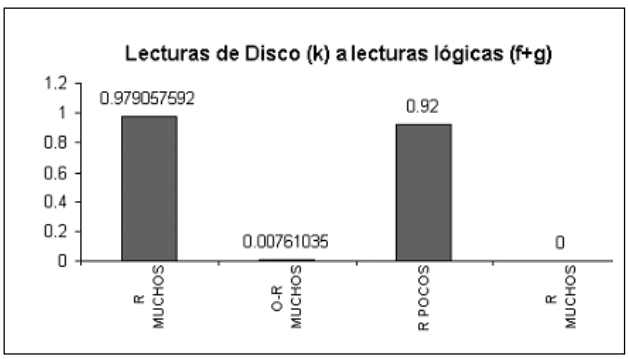


Figura 6. Tasa de lecturas de disco a lecturas lógicas para GROUP BY con las cláusulas HAVING y ORDER BY

De manera contraria, en la tasa de lecturas de disco a lecturas lógicas (Figura 6), se observa un comportamiento pobre en el modelo relacional, y muestra ser un modelo excesivamente costoso (en este caso) comparado con el modelo objeto-relacional, el cual presenta valores bastante eficientes.

Resultados para las pruebas con subconsultas

Para las sentencias que utilizan las subconsultas, se tratarán aquellas que se realizan con las cláusulas IN y EXISTS. Primero se mostrarán los resultados con las subconsultas cuando se utiliza la cláusula EXISTS, y luego los resultados cuando se utiliza la cláusula IN.

Resultados de las subconsultas cuando se utiliza la cláusula EXISTS

Las consultas utilizadas se muestran a continuación:

Para el modelo relacional

```
SELECT *
FROM cliente c
WHERE NOT EXISTS ( SELECT * FROM factura
f WHERE f.ced_cli = c.cédula )
```

Para el modelo objeto relacional

```
SELECT *
FROM cliente c
WHERE NOT EXISTS ( SELECT * FROM factura
f WHERE f.ref_cli = REF(c) )
```

En la Tabla 5 se presentan los resultados para las pruebas realizadas

Tabla 5. Resultados para las subconsultas cuando se utiliza la cláusula EXISTS

TASA	(f + g) / h	d / e	i / j	k / (f + g)	DATOS
VALOR NORMAL	Menor que 10	1 (o cercano a 1)	Mientras mayor mejor	Menor del 10%	
MODELO RELACIONAL	0.978723404	1	11.5	0	Muchos
	1.4375	1	11.5	0	Pocos
MODELO OBJETO-RELACIONAL	1862.413043	1	11.5	0	Muchos
	207.2857143	1	11.5	0	Pocos

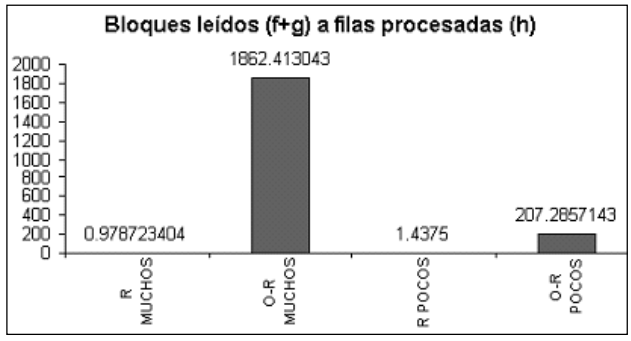


Figura 7. Tasa de bloques leídos a filas procesadas para las subconsultas cuando se utiliza la cláusula EXISTS

En este caso se puede observar que en la tasa de bloques leídos a filas procesadas (Figura 7), hay una diferencia bastante significativa entre ambos modelos donde se muestra una ganancia bastante visible del modelo relacional sobre el modelo objeto-relacional, que en este caso muestra una eficiencia baja.

Resultados de las subconsultas cuando se utiliza la cláusula IN

Las consultas utilizadas se muestran a continuación:

Para el modelo relacional

```
SELECT *
FROM     cliente
WHERE    cédula NOT IN ( SELECT ced_cli
         FROM factura )
```

Para el modelo objeto-relacional

```
SELECT *
FROM     cliente c
WHERE    REF(c) NOT IN (SELECT ref_cli
         FROM factura )
```

En la Tabla 6 se muestran los resultados para las pruebas realizadas

Tabla 6: Resultados para las subconsultas cuando se utiliza la cláusula IN

TASA	(f + g) / h	d / e	i / j	k / (f + g)	DATOS
VALOR NORMAL	Menor que 10	1 (o cercano a 1)	Mientras mayor mejor	Menor del 10%	
MODELO RELACIONAL	0.978723404	1	11.5	0	Muchos
	1.4375	1	11.5	0	Pocos
MODELO OBJETO-RELACIONAL	1862.413043	1	11.5	0	Muchos
	207.2857143	1	11.5	0	Pocos

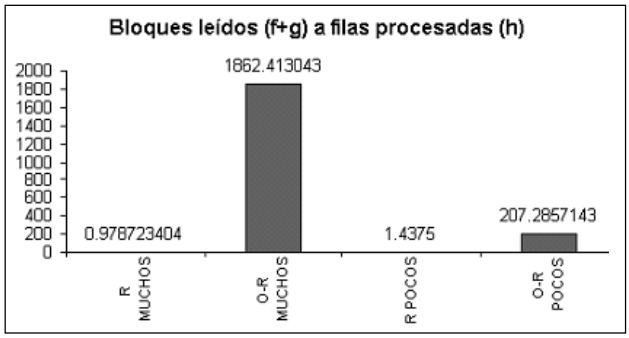


Figura 8. Tasa de bloques leídos a filas procesadas para las subconsultas cuando se utiliza la cláusula IN

De manera análoga a las pruebas realizadas sobre las subconsultas cuando se utiliza la cláusula EXISTS, las pruebas realizadas con la cláusula IN muestran que en la tasa de bloques leídos a filas procesadas (Figura 8) el modelo relacional ofrece de nuevo un rendimiento bastante superior con respecto al modelo objeto-relacional.

También se puede observar de acuerdo con los resultados que, aunque la cláusula EXISTS debería ser más eficiente que la cláusula IN (Oracle corporation[2], 2003), en este caso particular los dos tipos de subconsultas ofrecen un rendimiento exactamente igual.

En las demás tasas, en ambos casos, los dos modelos se comportaron de manera eficiente

Resultados para consultas con auto-rreuniones (*self joins*)

En este caso sólo se realizaron las pruebas con una cantidad fija de datos. A continuación se muestran las consultas utilizadas para este caso.

Para el modelo relacional

```
SELECT    a.nombre, d.nombre as bisabuelo
FROM      vendedor a, vendedor b,
          vendedor c, vendedor d
WHERE     a.ced_rec = b.cédula
AND       b.ced_rec = c.cédula
AND       c.ced_rec = d.cédula
```

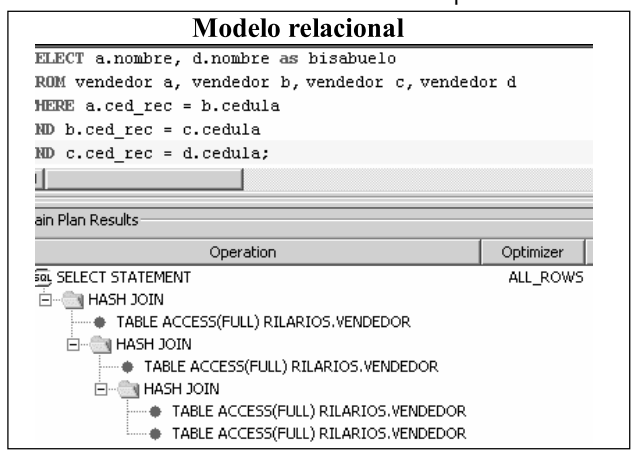
Para el modelo objeto-relacional.

```
SELECT    v.nombre, v.ref_rec.ref_rec.re-
          f_rec.nombre as bisabuelo
FROM      vendedor v
WHERE     v.ref_rec.ref_rec.ref_rec.nom-
          bre IS NOT NULL
```

Se debe observar que en la consulta realizada para el modelo objeto-relacional se debió utilizar la cláusula IS NOT NULL, para que ambos modelos mostraran la misma información y asegurar así igualdad de condiciones.

A pesar de que para mostrar los mismos resultados las consultas se escriben de manera diferente, se puede observar (Figura 9) que los planes de ejecución (EXPLAIN PLAN) para ambos modelos son iguales, por lo que se puede concluir que internamente las consultas se ejecutan de la misma manera.

En la Tabla 7 se muestran los resultados para este caso.



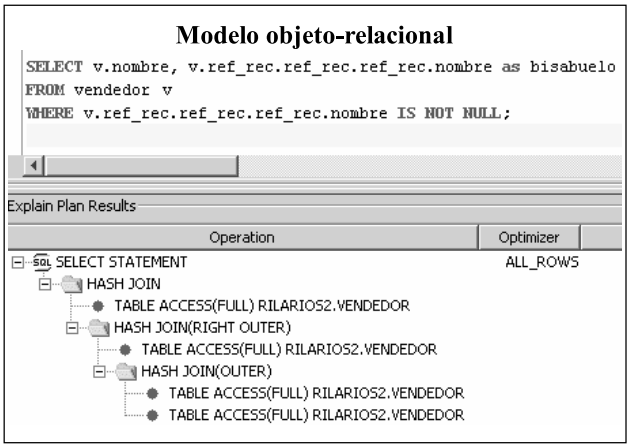


Figura 9. EXPLAIN PLAN para los modelos relacional y objeto-relacional para la consulta e auto-reunión

Tabla 7. Resultados para las consultas de auto-reunión

TASA	(f + g) / h	d / e	i / j	k / (f + g)
VALOR NORMAL	Menor que 10	1 (o cercano a 1)	Mientras mayor mejor	Menor del 10%
MODELO RELACIONAL	0.50617284	1	11.57142857	0
MODELO OBJETO-RELACIONAL	0.802469136	1	11.57142857	0

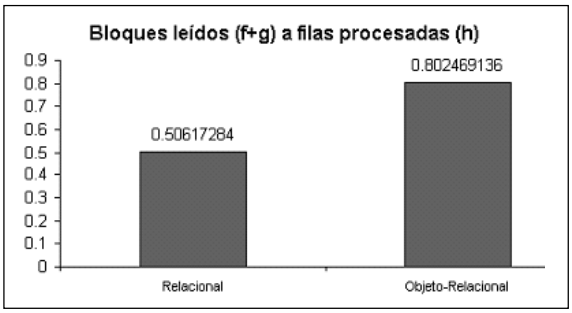


Figura 10. Tasa de bloques leídos a filas procesadas para las consultas con auto-reunión.

Como se puede observar en la Figura 10, en la tasa de bloques leídos a filas procesadas, aunque ambos modelos presentan rendimientos bastante eficientes y la diferencia no es muy significativa, el modelo relacional aventaja al modelo objeto-relacional. En las otras tasas, de la misma manera que en casos anteriores, ambos modelos presentan muy buen rendimiento.

Resultados para el caso del modelo 2

En este punto, se analizarán los resultados para el modelo que involucra a los continentes, países, regiones, etc. Se pretende observar la eficiencia de los punteros dentro del modelo objeto-relacional en comparación con la reunión de tablas en el modelo relacional. Se utilizaron tablas desde muy pequeñas hasta grandes para tratar de combinar (y cubrir) un espectro grande de resultados. Se muestran a continuación las consultas que se utilizaron para este caso en particular.

Para el modelo relacional

```
SELECT    /*+ ORDERED NO_INDEX(c)
          NO_INDEX(p)*/c.nombre, p.nombre,
          r.nombre, d.nombre, m.nombre
FROM      municipio m, departamento d,
          región r, pais p, continente c
WHERE     c.id = p.id_cont
AND       p.id = r.id_pais
AND       r.id = d.id_regi
AND       d.id = m.id_depto
```

Para el modelo objeto-relacional

```
SELECT
     m.ref_depto.ref_regi.ref_pais.-
     ref_cont.nombre,
     m.ref_depto.ref_regi.ref_pa-
     is.nombre,
     m.ref_depto.ref_regi.nombre,
     m.ref_depto.nombre,
     m.nombre
FROM municipio m
```

Nótese la gran diferencia entre la sintaxis de las dos sentencias, esto es debido a la naturaleza de ambos modelos, es decir, para la consulta en el modelo relacional se hace necesario realizar la reunión de tablas explícitamente en las cláusulas FROM y WHERE de la sentencia; sin embargo, cuando se aprovecha la utilización de punteros en el modelo objeto-relacional, es posible "navegar" hacia las demás tablas involucradas en la consulta, por lo que no es necesario especificar todas las tablas en la cláusula FROM de la sentencia, sino que basta solamente con utilizar la tabla municipio en la cláusula FROM de la sentencia.

La Tabla 8 muestra los resultados para las pruebas

Tabla 8. Resultados para las pruebas con el Modelo 2

TASA	(f + g) / h	d / e	i / j	k / (f + g)
VALOR NORMAL	Menor que 10	1 (o cercano a 1)	Mientras mayor mejor	Menor del 10%
MODELO RELACIONAL	0.003805	1	14.99970001	0.88436268
MODELO OBJETO-RELACIONAL	0.074357	1	14.99970001	0.08000591

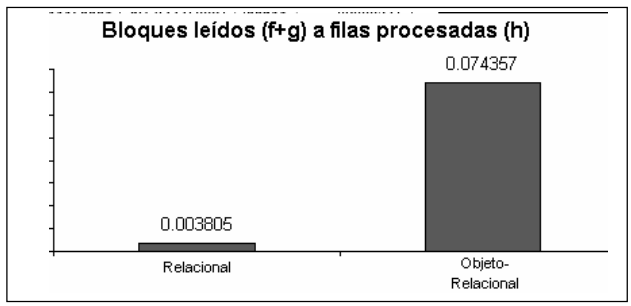


Figura 11. Tasa de bloques leídos a filas procesadas para las pruebas con el Modelo 2

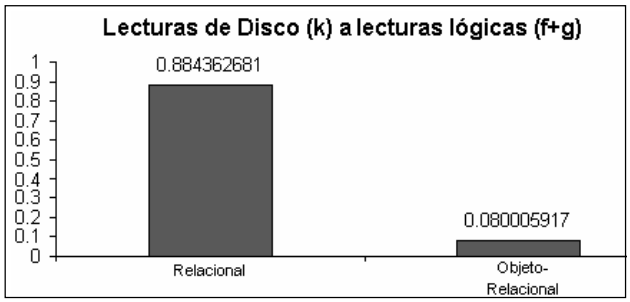


Figura 12. Tasa de lecturas de disco a lecturas lógicas para las pruebas con el Modelo 2

A pesar de que en la tasa de bloques leídos a filas procesadas (Figura 11) para las consultas con el modelo geográfico, tanto el modelo relacional como el objeto-relacional presentaron rendimientos bastante buenos y eficientes, se observa que el modelo relacional presenta un mejor desempeño en general.

Todo lo contrario sucede en la tasa de lecturas de disco a lecturas lógicas (Figura 12), en la cual el rendimiento del modelo objeto-relacional es mucho mejor que el del modelo relacional, el cual presentó un rendimiento bastante pobre, lo cual debe ser tomado en cuenta a la hora de escoger el modelo más adecuado para trabajar.

Resultados para la reunión (*join*) sencilla

Aunque dentro de los ejemplos observados se desarrollaron reuniones, es importante analizar detenidamente la comparación de este caso, en el cual se analizará el rendimiento de una de las colecciones del modelo objeto-relacional: las tablas anidadas. Para esta prueba se realizará la reunión de las tablas factura y detalle (del Modelo 1), en donde la tabla detalle en el modelo objeto-relacional es una tabla anidada. Las consultas realizadas para estas pruebas son:

Para el modelo relacional

```
SELECT    f.fecha, d.cant, d.valor_deta-
          lle
FROM      factura f, detalle d
WHERE     f.numero = d.num_fact
```

Para el modelo objeto-relacional

```
SELECT    f.fecha, d.cantidad, d.valor-
          _detalle
FROM      factura f, TABLE(f.detalle) d
```

Los resultados para las pruebas con la reunión sencilla se muestran en la Tabla 9

Tabla 9. Resultados para reunión sencilla

TASA	(f + g) / h	d / e	i / j	k / (f + g)	DATOS
VALOR NORMAL	Menor que 10	1 (o cercano a 1)	Mientras mayor mejor	Menor del 10%	
MODELO RELACIONAL	0.073634002	1	14,87106918	0	**Muchos**
	0.076597545	1	14,87106918	0	**Pocos**
MODELO OBJETO-RELACIONAL	0.085101021	1	14,87106918	0.21026653	**Muchos**
	0.103550296	1	14,87106918	0.20816326	**Pocos**

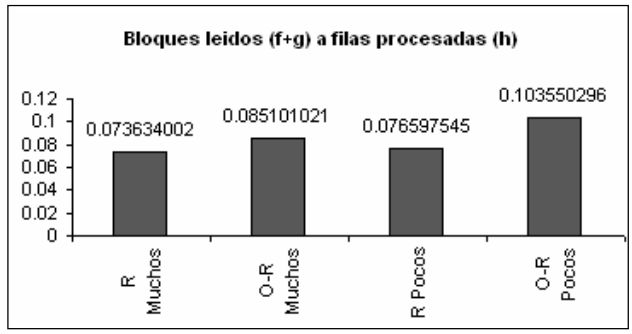


Figura 13. Tasa de bloques leídos a filas procesadas para la reunión sencilla

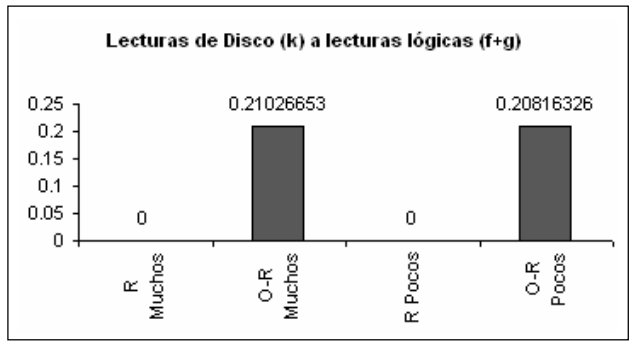


Figura 14. Tasa de lecturas de disco a lecturas lógicas para la reunión sencilla

En la tasa de bloques leídos a filas procesadas (Figura 13), se observa que ambos modelos presentan rendimientos bastante buenos y las diferencias entre ellos no es significativa, aunque el modelo relacional aventaja ligeramente al objeto-relacional. Sin embargo en la tasa de lecturas de disco a lecturas lógicas (Figura 14) se nota una gran diferencia entre ambos modelos, donde el rendimiento del modelo objeto-relacional se encuentra por encima del valor que se debe procurar (menor al 10%), lo que influye bastante en el rendimiento general del modelo.

Conclusiones y trabajos futuros

En general, en las pruebas realizadas, ambos modelos se comportaron de manera bastante aceptable; sin embargo, en la tasa de lecturas de disco a lecturas lógicas, am-

bos modelos presentaron deficiencias en algunos casos determinados, lo que influye en su rendimiento.

A través de todo el proceso se ha encontrado una constante bastante interesante: el modelo objeto-relacional, a pesar de ofrecer una nueva forma de modelar los datos (Stonebraker, 1996) y obtener rendimientos bastante interesantes, no posee aún la madurez (en lo que a ejecución se refiere) necesaria para tomar ventaja sobre el modelo relacional, especialmente en la tasa de bloques leídos sobre filas procesadas.

Puede decirse que el ganador en estas pruebas ha sido el modelo relacional, el cual a través de todas las pruebas ha mostrado ser un modelo más maduro (posiblemente porque ha habido más tiempo para optimizarlo) y en general mostró rendimientos bastantes buenos.

Aunque es fácil caer en la tentación de adoptar modelos nuevos, es necesario contar con criterios que permitan decidir cual de ellos es más ventajoso. Es necesario realizar más pruebas, por ejemplo, el comportamiento de los *varrays* (Burleson, 2004) del modelo objeto-relacional no ha sido analizado para tener cada vez una mejor idea sobre el rendimiento de ambos modelos. Igualmente se hace necesario examinar como el uso de índices puede influir en los resultados de ambos modelos.

Bibliografía

Aronoff, E., EXPLAIN PLAN: Everything you wanted to know and had no-one to ask. Quest Software, 1995.

Bodnar, R., "*Modeling Customers, an Object Lesson*", Oracle Magazine, Vol. XIV, N° 4, Jul/Ago 2000, pp.91-98.

Burleson, D., "*Phisical DB Design Using Oracle*", CRC Press, 2004, pp. 154-196.

Date, C., "*Introducción a los sistemas de bases de datos*", Pearson Educación, 2001, pp. 554-560.

Harrison, G., "*Oracle SQL High-Performance Tuning*", 2a ed., Prentice Hall, 2000, pp. 107-113, 134-159.

Kyte, T., "*Use of Bind Variables on 8.1.6*", Página electrónica Ask Tom, 2000.

Niemiec, R., "*Comunicación privada a través de correo electrónico*", febrero 2005.

Nyffenegger[1], R., Shared Pool [Oracle], http://www.adp-gmbh.ch/ora/concepts/shared_pool.html, 2000.

Nyffenegger[2], R., Oracle SQL Hints, http://www.adp-gmbh.ch/ora/sql/hints.html, 2001.

Oracle Corporation[1], Oracle DB Application Developer's Guide Object Relactional Features, 2003, pp. 27-106.

Oracle Corporation[2], Oracle DB Performance Tuning Guide, 2003, pp.303-307, 411-544.

Raghavan, G., Oracle Tips, Comunicación electrónica con usuario, http://www.arikaplan.com/oracle/ari52499c.html, 1999.

Stonebraker, M., "*Object-Relational DBMSs: The Next Great Wave*", Morgan Kauffman, 1996, pp. 18-39.