Memoria-practica-3-optimizacione...

yorer

Ficheros y Bases de Datos

2º Grado en Ingeniería Informática

Escuela Politécnica Superior. Campus de Leganés
Universidad Carlos III de Madrid

Titulación: GRADO INGENIERIA INFORMATICA Año Académico: 2020/2021 -- Curso: 2º Asignatura: Ficheros y Bases de Datos Título: Memoria Práctica 3 – Diseño Físico en Oracle	uc3m | Universidad Carlos III de Madrid

Profesor:		**Grupo**	
Alumno/a:		**NIA:**	
Alumno/a:		**NIA:**	
Alumno/a:		**NIA:**	

1 Introducción

2 Análisis

- **Medición de la base de datos**

Se ha llevado a cabo una medición de los tiempos de ejecución y número de bloques de la base de datos aplicando para ello el procedimiento run_test dado por el profesorado. Hay que mencionar que para realizar estas pruebas hemos utilizado la base de datos por defecto que se nos entregó en la práctica 2, evitando así cualquier tipo de manipulación que se haya podido hacer durante la realización de dicha práctica.

Para conseguir unos datos más consistentes y fiables, ejecutamos varias veces run_test pasándole como parámetro el número 10, para que mostrará los datos obtenidos en base a 10 iteraciones del test. El número de bloques obtenido a partir de las distintas ejecuciones rondaba los 2152980 y el tiempo de ejecución en torno a los 17000 y 18000 milisegundos, es decir unos 18 segundos.

Insertamos una captura de los resultados de la última ejecución que hicimos del run_test:

```
RESULTS AT 06/05/21
TIME CONSUMPTION: 18081,3 milliseconds.
CONSISTENT GETS: 2152976,8 blocks
```

- **Análisis general**

La carga de trabajo está constituida por varias operaciones, entre ellas nos encontramos las inserciones en la tabla edition y en la tabla participation, las 3 consultas y las operaciones de reseteo donde borra y altera tablas.

Nos encontramos ante una organización serial no consecutiva, con un tamaño de bloque de 8K, donde cada tabla creada se referencia por un índice a su clave primaria. Como nos encontramos ante una organización serial las operaciones de inserción son óptimas, mientras que las búsquedas, si no son identificativas por clave primaria, requieren hacer un full scan, además el aprovechamiento de espacio también es óptimo.

A la hora de analizar si este diseño es un buen diseño o no, debemos tener en cuenta varios aspectos, como los mencionados antes sobre la inserción o la búsqueda. Podríamos decir que como diseño inicial de nuestra base, es un buen diseño, donde se guarda la integridad relacional, y se expresan las relaciones necesarias, pero que a la hora de consultar resulta bastante costosa, al tener que hacer numerosos full scans y joins para obtener ciertos datos. Un posible arreglo ante este aspecto es la creación de clusters que permitan unir la información de varias tablas reduciendo así los accesos. Estas posibles mejoras las señalaremos en los siguientes análisis y en la propuesta de nuestro diseño físico.

Tenemos que mencionar que el número de frecuencias de los procesos es desconocido, por lo que a la hora de realizar nuestro diseño físico tomaremos como objetivo minimizar el número de bloques al que se accede.

Para las distintas selecciones realizadas en sql se usará el explain plan básico de oracle, que consiste en lo siguiente:

Un explain plan es una representación de la ruta de acceso que el optimizador de Oracle Database toma cuando es ejecutada una consulta SQL en la base de datos. El procesamiento de una consulta SQL se puede dividir en 7 fases:

1- Sintáctica: se verifica la sintaxis de la consulta

2- Semántica: comprueba que todos los objetos existan y sean accesibles

3- View Merging: se reescribe la consulta como join en tablas base en lugar de usar vistas

4- Transformación de sentencias: se reescribe la consulta transformando algunas construcciones complejas en otras más simples siempre que sea posible (por ejemplo, subconsulta unnesting, in/or transformation, etc.). Algunas transformaciones usan reglas mientras que otras tienen un costo basado en estadísticas.

5- Optimización: determina la ruta de acceso óptima para la consulta. El optimizador basado en costos (CBO) usa estadísticas para analizar los costos relativos de acceso a los objetos.

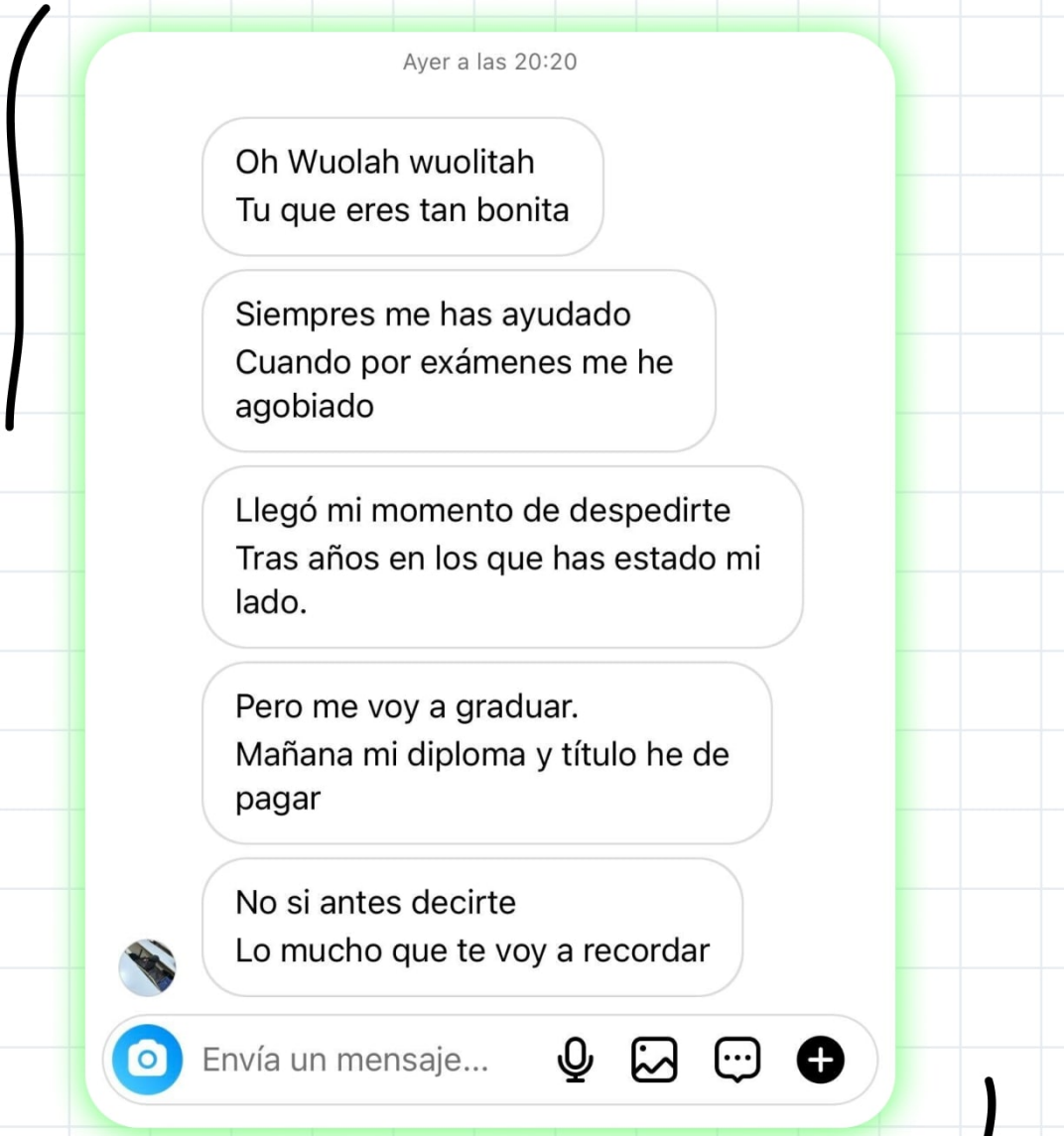
6- Query Evaluation Plan (QEP): Se evalúan las diferentes opciones de acceso y se genera el plan.

7- QEP execution: Se llevan adelante la acciones indicadas en el plan de ejecución determinado.

Los pasos del 1 al 6 a veces se agrupan y se denominan 'parsing', fase de 'parseo' o simplemente análisis.

Que no te escriban poemas de amor cuando terminen la carrera 🚩🚩🚩🚩🚩🚩

(a nosotros por suerte nos pasa)



WUOLAH

El paso 7 es la ejecución misma de la sentencia.

El “explain plan" es fundamentalmente una representación de la ruta de acceso generada en el paso 6.

Una vez que se ha determinado el “access path” o ruta de acceso, éste se almacena en memoria en la “library cache” junto con la sentencia correspondiente. Las consultas se almacenan en la “library cache” en función de una representación en hash del literal de la sentencia, este hash luego lo utiliza Oracle para ubicarla en memoria. Antes de realizar las fases 1-6 antes mencionadas para un SQL determinado, Oracle busca la sentencia en la “library cache” para ver si ya fue "parseada" y utilizar el plan ya generado. Primero aplica un algoritmo hash al texto del SQL en cuestión y luego se busca este valor hash en “library cache”. Si el hash coincide con alguno de los almacenados, se utilizará ese “Access path” o método de acceso para resolver la consulta. Este será siempre utilizado (para la misma consulta) hasta tanto la misma vuelva a parseada. Dependiendo de las circunstancias, el plan de ejecución tal vez varíe de un parseo a otro. (En futuros posteos veremos que determina que una consulta vuelva a ser parseada).

Por esto, entendemos que el sistema más común de búsqueda usado es el full scan, no necesariamente siempre el peor posible, pero con unos índices y algún clúster concreto, se podría optimizar esta selección de plan de búsqueda y mejorarlo en general.

- **Análisis inserciones**

Como se ha mencionado antes se realizan dos inserciones, ambas inserciones además requieren de una consulta, por ello las inserciones se basarán en la selección de los datos que buscamos y su posterior inserción al final (por ser una organización serial), en las respectivas tablas.

- Inserción en edition:

  Se realiza un full access a la tabla competición a la que previamente se ha realizado un cross join con categoría con un coste de 583, donde finalmente se filta por country y se insertan las respectivas filas.

  Con autotrace podemos ver que el coste total de la operación es 583 y el número de bloques accedidos es de 19019.

  En este caso al hacer un cross join podríamos suponer que un cluster beneficiaria a esta inserción, pero dado que no tienen ningun atributo en común el clúster entre ambas no tendría sentido crearlo, además de que no se podría.

  Como alternativa al cluster en competition podemos indizar en base al país de esta manera haremos que la inserción sea más eficiente al hacer un acceso indizado en lugar de un full scan.

- Inserción en participation:

  Se realiza un full Access de las tablas edition y section para encontrar las filas que queremos para luego hacer una combinación natural e insertar en la tabla.

  Con Autotrace vemos que el número de bloques accedidos es 15535.

  Una posible mejora sería crear un índice por el atributo year en edición, ya que a la hora de filtrar nos interesa saber los años por lo que podría reducir el número de accesos. En sección al seleccionar la clave primaria creemos que no sería necesario realizar ningún tipo de índice o estructura auxiliar.

- **Análisis borrados:**

  - De cara a los borrados realizados por el paquete hemos planteado como una mejora para los mismos crear dos índices más. En referencia a la tabla edition, hemos comprobado que se accede a esta con frecuencia usando como clave de búsqueda el año, por lo que creemos que este índice es fundamental para el desarrollo general de los procesos realizados por el script.

  - En cuanto al otro borrado, observamos que se realiza en la tabla participation búsqueda en base al nombre de la edición, por lo que será interesante tener un índice y ver si este favorece realmente al borrado.

- **Análisis Consultas:**

A continuación, realizaremos un análisis detallado de cada consulta proporcionada, indicando posibles cambios que se podrían realizar para su optimización.

- **Consulta 1:** Competiciones para las que no hay un campeón registrado, excluyendo las que están en curso.

  - **Análisis query:**

    - **Competiciones año sysdate:** Para la primera selección cogeremos los datos referentes al nombre de la competición, país, edición etc para aquellas competiciones que no estén en curso, es decir que su year sea menor a sysdate.
    - **Competiciones con ganadores:** Para la segunda selección cogeremos de la tabla de ganadores todas las competiciones con la intención de aplicar el minus a la selección 1.
    - La selección 3 dentro de la query 1 realiza un conteo de ediciones para cada competición dentro de la subquery 1 haciendo un join a la tabla de ediciones. Con esto logra determinar el número de ediciones sin ganador y

una lista de los años.

- **Autotrace y posibles mejoras:**

  - Con autotrace podemos observar como todas las tablas son accedidas con un full access, porque como se ha mencionado antes, se utiliza la organización serial no consecutiva que tiene por defecto oracle. El número de bloques accedidos son 7683.
  - Algún cambio que se podría realizar es crear un índice para el atributo año en edición que nos permitiría filtrar de forma más eficiente los años que queremos. Como la mayor parte de la consulta se basa en un MINUS y luego un JOIN, es necesario que se haga un full scan para seleccionar las filas que queremos, por lo que creemos que no podemos optimizar dichas partes.

- **Consulta 2:**

  - **Análisis query:**

    - **Cheaters:** La primera selección recoge los spic de los tramposos junto al año cuando cometieron las trampas.
    - **Secciones con cheaters:** Usaremos la subquery de cheaters junto a un join a signing para determinar qué secciones tienen cheaters en un año concreto. Para ello la selección cogerá la pk de la sección junto al año de trampa.
    - **Cheaters/año en club:** Cogemos el número de cheaters de cada club y calculamos en base al año la cantidad de ellos que ha habido en base al club que pertenezcan.
    - **Campeones:** Cogerá en base a los participantes totales sacados de participation y la tabla de winners. Después usa la tabla de ediciones para marcar qué edición de la competición es la que se ha seleccionado.
    - **Trofeos:** Obtenemos la cantidad de trofeos junto a la cantidad de cheater para cada club dentro de las subquerys anteriores.
    - La selección final esencialmente coge los datos de todas las subqueries para responder al enunciado en sí.

  - **Autotrace y posibles mejoras:**

    - Con un autotrace podemos observar que el número de operaciones que realiza esta consulta es bastante grande, vemos que se realizan operaciones de full access a tablas como winner, signing o full access a índices de clave primaria en edition o participation. El número de bloques accedidos es de
      120964.

- Hemos pensado en usar un cluster para el primer join usando cheaters y signing, pero dado que cheaters es una subquery creada en la consulta consideramos que no va a mejorar la situación y que potencialmente al ser un clúster de solo una tabla, la empeore. La misma lógica para la selección de cheaters por secciones.

- **Consulta 3:**

  - **Análisis query:**

  - **Atletas:** Selecciona los spics y sus respectivas secciones para cada deportista participante combinado con join al signing para obtener los datos del deportista.
  - **Carrera:** Coge de cada atleta de la subquery 1 su fecha de inscripción dentro de la federación concreta a la que referencia su ficha federativa.
  - **Min date:** Coge para cada deporte y país la fecha mínima de federación dentro del subquery de carrera.
  - Finalmente usa los datos obtenidos para seleccionar los deportistas que coincidan con esta consulta en base a country, sport y expedition.

  - **Autotrace y posibles mejoras:**

  - Con autotrace se observa, como en la anterior consulta, que el número de operaciones es considerable, y como es lógico por la organización serial que hay presente, en las operaciones de selección se hacen full scans sobre la tablas. El número de accesos que se realizan es de 10865810, este número es acorde a la consulta en sí, ya que se consulta una gran cantidad de registros.
  - Tenemos un claro cluster entre participation y edition ya que ambos forman un join que con un cluster permitirá acceder a la información de manera mucho más eficiente.
  - Podemos crear un índice para federation_card que se base en el SPIC de esta manera, consideramos que mejorará el proceso de búsqueda.

3 Diseño Físico

- Después de analizar toda la carga de trabajo, uno de los cambios que creíamos necesarios era aumentar el tamaño de cubo en aquellas tablas donde se estaban realizando las operaciones de inserción, para reducir así el número de cubos necesarios y con ello el número de accesos, por ello hemos aumentado de 8K a 16K el

tamaño del cubo en las tablas edition y participation. El nuevo resultado de ejecutar run_test(10) es:

```
TIME CONSUMPTION: 17076,5 milliseconds.
CONSISTENT GETS: 26127,3 blocks
```

- Como se puede observar supone un cambio muy significativo en el número de bloques, pasando de 2 millones de bloques a unos 26000. Como habíamos mencionado esto se debe a que necesitamos menos cubos para guardar los registros, esto unido a la reducción de desbordamientos, hace que esta significativa reducción sea algo esperable.

- Por probar también cambiaremos el tablespace a 2K esperando que efectivamente empeore la situación al aumentar significativamente el número de desbordamientos medio.

```
TIME CONSUMPTION: 89010,7 milliseconds.
CONSISTENT GETS: 4011472,8 blocks
```

- De cara a modificar el PCTFREE o el PCTUSED de oracle, hemos determinado que no compensa realizar ninguna modificación ya que el free por default se encuentra al 10% lo que es de por si un valor significativamente bajo. Aunque no planeamos realizar cambios sobre las bases en la práctica, de cara a un hipotético futuro, no compensaría bajar de ese 10% de espacio libre para ‘emergencias’ (updates etc)

- No nos hemos planteado crear índices de la PK de ninguna tabla ya que oracle de forma automática crea esos índices al crearse la tabla.

Diseño físico realizado sobre insert/delete:

- **Inserción en edition:**

  índice country de edition: create index ind_country on competition(country);
  Hemos empleado un hint para obligar a oracle utilizar el índice creado:
  /*+ INDEX(competition) */

- **Inserción en participation:**

  índice year de edition: create index ind2 on edition(year);
  Hemos empleado un hint para obligar a oracle utilizar el índice creado:
  /*+ INDEX(edition) */ +

- **Delete en participation:**

  índice en edition de participation: create index ind_edition on participation(edition);
  Hemos empleado un hint para obligar a oracle utilizar el índice creado:
  /*+ INDEX(edition) */

- **Delete en edition:**

  Índice year de edition (Ya ha sido creado al optimizar la inserción en participation).

Diseño físico realizado sobre querys:

- **Query 1:**

  Índice year de edition (Ya ha sido creado al optimizar la inserción en participation).

- **Query 2:**

  Dado que se usa el sistema full scan debido a que no existe otra manera de realizar las consultas y no podemos realizar un cluster de subquerys, no hemos encontrado nada que optimizar en esta consulta.

- **Query 3:**

  Indice spic federation_card: create index ind_spic on federation_card(spic);
  Hemos empleado un hint para obligar a oracle utilizar el índice creado:
  /*+ INDEX(federation_card) */

4 Evaluación

Antes de hacer la comparación global entre el diseño inicial y nuestro diseño físico, vamos a realizar un análisis de los cambios individuales efectuados en cada parte.

- **Inserciones:**
  - **Inserción en Edition:**
    Para la inserción en la tabla de edition observamos que si no forzamos el uso del índice el proceso hash elegirá el método de full scan de nuevo considerando que el índice no le sirve. Probamos forzando el índice usando hints y efectivamente vemos que empeora el resultado total en alrededor de 200 bloques.
  - Inserción sin Hint-Índice:

```
Plan de Ejecucion
----------------------------------------------------------
Plan hash value: 271390773

-------------------------------------------------------------------------------------------
| Id  | Operation                | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | INSERT STATEMENT         |             |  6121 |   824K|   594   (3)| 00:00:01 |
|   1 |  LOAD TABLE CONVENTIONAL | EDITION     |       |       |            |          |
|   2 |   MERGE JOIN CARTESIAN   |             |  6121 |   824K|   594   (3)| 00:00:01 |
|*  3 |    TABLE ACCESS FULL     | COMPETITION |   874 |   106K|   343   (1)| 00:00:01 |
|   4 |    BUFFER SORT           |             |     7 |    91 |   251   (6)| 00:00:01 |
|   5 |     INDEX FAST FULL SCAN | PK_CATEGORY |     7 |    91 |     0   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

- Inserción con Hint-Índice:

```
Plan de Ejecucion
----------------------------------------------------------
Plan hash value: 271390773

-------------------------------------------------------------------------------------------
| Id  | Operation                | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | INSERT STATEMENT         |             |  6121 |   824K|   594   (3)| 00:00:01 |
|   1 |  LOAD TABLE CONVENTIONAL | EDITION     |       |       |            |          |
|   2 |   MERGE JOIN CARTESIAN   |             |  6121 |   824K|   594   (3)| 00:00:01 |
|*  3 |    TABLE ACCESS FULL     | COMPETITION |   874 |   106K|   343   (1)| 00:00:01 |
|   4 |    BUFFER SORT           |             |     7 |    91 |   251   (6)| 00:00:01 |
|   5 |     INDEX FAST FULL SCAN | PK_CATEGORY |     7 |    91 |     0   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

- **Inserción en participation:**
  Para la inserción en la tabla participation hemos usado un índice sobre la variable year para facilitar la búsqueda de los datos requeridos.
  Observamos la diferencia en el hash a la hora de ejecutar el mismo insert y el resultado de bloques nos da una diferencia de casi 10K bloques, pasando de los 15K bloques iniciales a alrededor de 4k bloques.
  Esto se mantiene dentro de lo esperado ya que al crear el índice con year hemos ahorrado al proceso el full scan y en su lugar solo ha tenido que coger los datos referentes al año seleccionado.

- Inserción sin Hint-índice:

```
Plan hash value: 54202617

-----------------------------------------------------------------------------------------
| Id  | Operation                | Name          | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | INSERT STATEMENT         |               | 24851 |    10M|       |  2093   (1)| 00:00:01 |
|   1 |  LOAD TABLE CONVENTIONAL | PARTICIPATION |       |       |       |            |          |
|*  2 |   HASH JOIN              |               | 24851 |    10M|  2688K|  2093   (1)| 00:00:01 |
|*  3 |    TABLE ACCESS FULL     | EDITION       | 14303 |  2514K|       |   961   (1)| 00:00:01 |
|   4 |    VIEW                  |               | 59349 |    13M|       |   276   (2)| 00:00:01 |
|   5 |     SORT GROUP BY        |               | 59349 |    13M|       |   276   (2)| 00:00:01 |
|   6 |      TABLE ACCESS FULL   | SECTION       | 59349 |    13M|       |   273   (1)| 00:00:01 |
-----------------------------------------------------------------------------------------
```

- Inserción con Hint-índice:

```
Plan de Ejecucion
----------------------------------------------------------
Plan hash value: 711928405

-----------------------------------------------------------------------------------------------------
| Id  | Operation                                | Name          | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | INSERT STATEMENT                         |               | 66003 |    20M|       |  5217   (1)| 00:00:01 |
|   1 |  LOAD TABLE CONVENTIONAL                 | PARTICIPATION |       |       |       |            |          |
|*  2 |   HASH JOIN                              |               | 66003 |    20M|  3928K|  5217   (1)| 00:00:01 |
|   3 |    TABLE ACCESS BY INDEX ROWID BATCHED   | EDITION       | 43206 |  3417K|       |  2693   (1)| 00:00:01 |
|*  4 |     INDEX RANGE SCAN                     | IND_YEAR      | 43206 |       |       |    95   (0)| 00:00:01 |
|   5 |    VIEW                                  |               | 54794 |    12M|       |  1664   (1)| 00:00:01 |
|   6 |     SORT GROUP BY                        |               | 54794 |  5993K|  6968K|  1664   (1)| 00:00:01 |
|   7 |      TABLE ACCESS FULL                   | SECTION       | 54794 |  5993K|       |   273   (1)| 00:00:01 |
-----------------------------------------------------------------------------------------------------
```

- **Borrados:**
  - Para los dos borrados realizados por el paquete hemos implementado un índice nuevo sobre la tabla participation en la variable edition. De esta manera estimamos que este nuevo índice junto al índice Year tan frecuentemente usado en edition, podremos mejorar el uso de bloques.

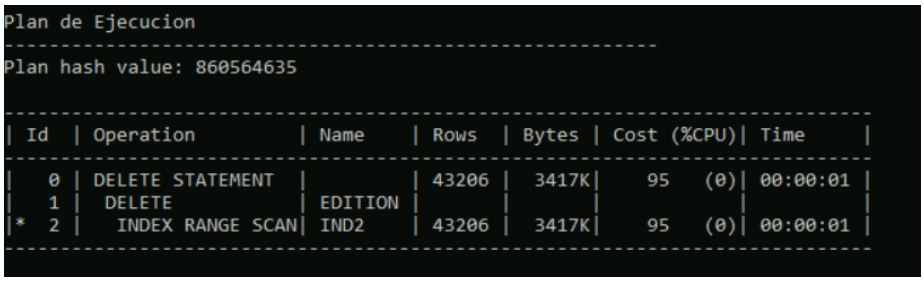
  - Delete en edition:

    - Usando índice obervamos una reducción de bloques significativa de casi 3K, teniendo en cuenta que pasamos de 3.5K a 0.6K, es un cambio bastante significativo que se ajusta a nuestras estimaciones.

Sin índice:

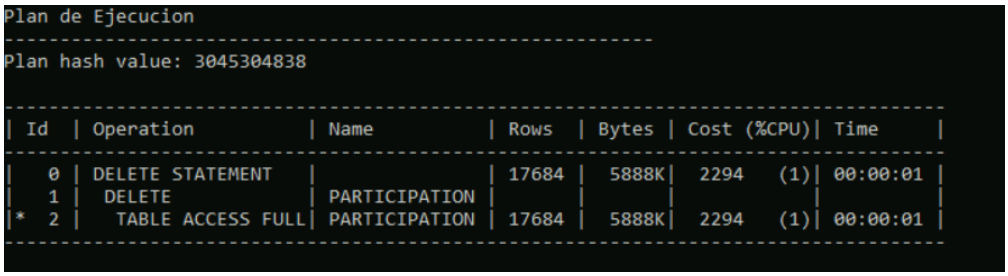
```
Plan de Ejecucion
----------------------------------------------------------
Plan hash value: 1943749504

-------------------------------------------------------------------------------
| Id  | Operation          | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | DELETE STATEMENT   |         |  4637 |   815K|   960   (1)| 00:00:01 |
|   1 |  DELETE            | EDITION |       |       |            |          |
|*  2 |   TABLE ACCESS FULL| EDITION |  4637 |   815K|   960   (1)| 00:00:01 |
-------------------------------------------------------------------------------
```

Con índice: Efectivamente pasamos de un Access full a un index range scan, lo cual aligera el proceso.



- Delete en participation:
  - Para este delete creamos el índice en base a la edición de la competición, esperando reducir los costes de búsqueda a la hora de borrar las filas.
  - Observaremos otra vez un cambio, solo que esta vez apenas significativo (de 132K a 131K) Esto entra dentro de lo esperado teniendo en cuenta que como índice, la edición no debería generar mejoras tan significativas como el año.
  - Dado que el borrado de año es de forma más específica, mientras que la edición no afecta tanto al borrado en si.

Sin índice usa el Access full:



Con índice usa el Index como era esperado:

```
Plan de Ejecucion
----------------------------------------------------------
Plan hash value: 3319160751

-------------------------------------------------------------------------------------
| Id  | Operation         | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | DELETE STATEMENT  |               |     1 |   161 |     3   (0)| 00:00:01 |
|   1 |  DELETE           | PARTICIPATION |       |       |            |          |
|*  2 |   INDEX RANGE SCAN| IND_EDITION   |     1 |   161 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------
```

Querys

- **Query 1:**
  - Para la query 1 hemos estudiado para las dos variantes 16/8K cómo afecta la creación del índice, sin embargo hemos observado que el plan seleccionado por oracle independientemente de que creemos el índice es siempre el mismo.

  - Esto nos lleva a pensar que el índice en este caso concreto no aporta beneficios. Creemos que esto se debe a que ya que la consulta pide todas las rows con años por debajo de sysdate, la cantidad de rows va a ser demasiado grande como para que le compense usar el índice tantas veces en lugar de realizar el full scan (que es el que acaba realizando).

  - Esto se mantendría dentro de la lógica una vez calculados los datos, pero no era lo que estimamos que pasaría. Sin embargo, en los datos obtenidos observamos reducciones de costes y bloques entre 16 y 8K lo cual si esperábamos, y de cara a con índice creado y sin índice creado también se observan reducciones pero que no se deben a los índices ya que el plan hash se mantiene usando el full Scan. Suponemos que esos pequeños cambios se deben a procesos internos del sistema.

  - Sin índice elige como sistema un access full como era de esperar:

```
Plan de Ejecucion
----------------------------------------------------------
Plan hash value: 2175445794

-------------------------------------------------------------------------------------------
| Id  | Operation                    | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |             |     1 |   347 |     5  (60)| 00:00:01 |
|   1 |  SORT GROUP BY               |             |     1 |   347 |     5  (60)| 00:00:01 |
|   2 |   NESTED LOOPS               |             |     1 |   347 |     4  (50)| 00:00:01 |
|   3 |    NESTED LOOPS              |             |     1 |   347 |     4  (50)| 00:00:01 |
|   4 |     VIEW                     |             |     1 |   167 |     4  (50)| 00:00:01 |
|   5 |      MINUS                   |             |       |       |            |          |
|   6 |       SORT UNIQUE            |             |     1 |   180 |            |          |
|*  7 |        TABLE ACCESS FULL     | EDITION     |     1 |   180 |     2   (0)| 00:00:01 |
|   8 |       SORT UNIQUE            |             |     1 |   167 |            |          |
|   9 |        INDEX FULL SCAN       | PK_WINNER   |     1 |   167 |     0   (0)| 00:00:01 |
|* 10 |     INDEX UNIQUE SCAN        | PK_EDITION  |     1 |       |     0   (0)| 00:00:01 |
|  11 |    TABLE ACCESS BY INDEX ROWID| EDITION     |     1 |   180 |     0   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

- Con índice vemos que no ha diferencias a la hora de elegir el proceso a seguir

```
Plan de Ejecucion
----------------------------------------------------------
Plan hash value: 1155367247

-------------------------------------------------------------------------------------------
| Id  | Operation                | Name      | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |           |    10 |  2480 |       | 11925   (1)| 00:00:01 |
|   1 |  SORT GROUP BY           |           |    10 |  2480 |       | 11925   (1)| 00:00:01 |
|*  2 |   HASH JOIN              |           |    10 |  2480 |   25M| 11924   (1)| 00:00:01 |
|   3 |    TABLE ACCESS FULL     | EDITION   |   288K|    22M|       |   960   (1)| 00:00:01 |
|   4 |    VIEW                  |           |   288K|    45M|       |  7254   (1)| 00:00:01 |
|   5 |     MINUS                |           |       |       |       |            |          |
|   6 |      SORT UNIQUE         |           |   288K|    22M|    27M|            |          |
|*  7 |       TABLE ACCESS FULL  | EDITION   |   288K|    22M|       |   966   (2)| 00:00:01 |
|   8 |      SORT UNIQUE         |           | 37320 |  2806K|  3528K|            |          |
|   9 |       INDEX FAST FULL SCAN| PK_WINNER | 37320 |  2806K|       |   157   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

- Con índice year forzado usando hint observamos que efectivamente empeora, en alrededor de 13K bloques ,por lo que determinamos que no compensa usarlo en esta query. Sin embargo, el índice se mantendrá en la base de datos ya que es muy útil para la inserción ya comentada anteriormente.

```
Plan de Ejecucion
----------------------------------------------------------
Plan hash value: 1908000469

-------------------------------------------------------------------------------------------
| Id  | Operation                          | Name      | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |           |    10 |  2480 |       | 28817   (1)| 00:00:02 |
|   1 |  SORT GROUP BY                     |           |    10 |  2480 |       | 28817   (1)| 00:00:02 |
|*  2 |   HASH JOIN                        |           |    10 |  2480 |   25M| 28816   (1)| 00:00:02 |
|   3 |    TABLE ACCESS FULL               | EDITION   |   288K|    22M|       |   960   (1)| 00:00:01 |
|   4 |    VIEW                            |           |   288K|    45M|       | 24146   (1)| 00:00:01 |
|   5 |     MINUS                          |           |       |       |       |            |          |
|   6 |      SORT UNIQUE                   |           |   288K|    22M|    27M|            |          |
|   7 |       TABLE ACCESS BY INDEX ROWID BATCHED| EDITION |   288K|    22M|       | 17858   (1)| 00:00:01 |
|*  8 |        INDEX RANGE SCAN            | IND2      |   288K|       |       |   608   (1)| 00:00:01 |
|   9 |      SORT UNIQUE                   |           | 37320 |  2806K|  3528K|            |          |
|  10 |       INDEX FAST FULL SCAN         | PK_WINNER | 37320 |  2806K|       |   157   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

- **Query 2:**

- Observamos que la única manera óptima de realizarla es usando el sistema de full scan debido a la cantidad de datos que maneja y al frecuente uso de subquerys . Por otro lado esta es la query con mayor número de subprocesos por lo que su Hash es significativamente más complejo que el resto

```
TIME CONSUMPTION: 2364,66666666666666666666666666666666666667 milliseconds.
CONSISTENT GETS: 89166,33333333333333333333333333333333333333 blocks
```

- El Hash efectivamente es como suponíamos, realizará los full para todas las consultas y cuando pueda usará índices de PK, pero en general será un proceso muy pesado.

```
Plan de Ejecucion
----------------------------------------------------------
Plan hash value: 3870690049

-------------------------------------------------------------------------------------------------------------------
| Id  | Operation                                | Name                        | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |                             |     1 |   278 |       |  8403   (1)| 00:00:01 |
|   1 |  TEMP TABLE TRANSFORMATION               |                             |       |       |       |            |          |
|   2 |   LOAD AS SELECT (CURSOR DURATION MEMORY)| SYS_TEMP_0FD9D8B03_CE643DD  |       |       |       |            |          |
|   3 |    HASH GROUP BY                         |                             |     1 |   134 |       |   346   (2)| 00:00:01 |
|   4 |     VIEW                                 | VM_NWVW_1                   |     1 |   134 |       |   345   (1)| 00:00:01 |
|   5 |      HASH GROUP BY                       |                             |     1 |   161 |       |   345   (1)| 00:00:01 |
|*  6 |       HASH JOIN                          |                             |     1 |   161 |       |   344   (1)| 00:00:01 |
|   7 |        INDEX FULL SCAN                   | PK_EVIDENCE                 |     1 |    22 |       |     1   (0)| 00:00:01 |
|   8 |        TABLE ACCESS FULL                 | SIGNING                     | 57164 |  7759K|       |   342   (1)| 00:00:01 |
|   9 |   SORT ORDER BY                          |                             |     1 |   278 |       |  8057   (1)| 00:00:01 |
|* 10 |    HASH JOIN                             |                             |     1 |   278 |       |  8056   (1)| 00:00:01 |
|  11 |     NESTED LOOPS                         |                             |     1 |   195 |       |     3  (34)| 00:00:01 |
|  12 |      NESTED LOOPS                        |                             |     1 |   195 |       |     3  (34)| 00:00:01 |
|  13 |       VIEW                               |                             |     1 |    69 |       |     3  (34)| 00:00:01 |
|  14 |        HASH GROUP BY                     |                             |     1 |    69 |       |     3  (34)| 00:00:01 |
|  15 |         VIEW                             |                             |     1 |    69 |       |     2   (0)| 00:00:01 |
|  16 |          TABLE ACCESS FULL               | SYS_TEMP_0FD9D8B03_CE643DD  |     1 |   134 |       |     2   (0)| 00:00:01 |
|* 17 |       INDEX UNIQUE SCAN                  | PK_CLUB                     |     1 |       |       |     0   (0)| 00:00:01 |
|  18 |      TABLE ACCESS BY INDEX ROWID         | CLUB                        |     1 |   126 |       |     0   (0)| 00:00:01 |
|  19 |     VIEW                                 |                             |     1 |    83 |       |  8053   (1)| 00:00:01 |
|  20 |      HASH GROUP BY                       |                             |     1 |   511 |       |  8053   (1)| 00:00:01 |
|* 21 |       HASH JOIN RIGHT OUTER              |                             | 37320 |    18M|       |  8051   (1)| 00:00:01 |
|  22 |        VIEW                              |                             |     1 |   138 |       |     2   (0)| 00:00:01 |
|  23 |         TABLE ACCESS FULL                | SYS_TEMP_0FD9D8B03_CE643DD  |     1 |   134 |       |     2   (0)| 00:00:01 |
|* 24 |        HASH JOIN                         |                             | 37320 |    13M|  8168K|  8049   (1)| 00:00:01 |
|* 25 |         HASH JOIN                        |                             | 37320 |  7726K|  5216K|  2688   (1)| 00:00:01 |
|  26 |          TABLE ACCESS FULL               | WINNER                      | 37320 |  4774K|       |   205   (1)| 00:00:01 |
|  27 |          TABLE ACCESS FULL               | EDITION                     |  288K |   22M |       |   960   (1)| 00:00:01 |
|  28 |         TABLE ACCESS FULL                | PARTICIPATION               |  342K |   52M |       |  2157   (1)| 00:00:01 |
-------------------------------------------------------------------------------------------------------------------
```

- **Query 3:**
  - Al probar si el índice surte efecto sobre el proceso hemos observado que no lo selecciona. Esto se debe a que Oracle considera que el índice perjudica más de lo que ayuda al proceso, esto tiene lógica teniendo en cuenta que el full scan es prácticamente inevitable de cara al proceso total.
  - De todas maneras, intentamos probar usando un hint para ver que sucedía y no lo cogió, lo que significa que empeora tanto la situación que Oracle lo rechaza.
  - De cara al número total de bloques, ronda alrededor de 13K, una cifra razonable teniendo en cuenta el espacio de bloque (8k) y la cantidad de datos que la query recoge.

```
13312  consistent gets
35419  physical reads
```

```
Plan de Ejecucion
----------------------------------------------------------
Plan hash value: 2924075988

-------------------------------------------------------------------------------------------------------------
| Id  | Operation                                | Name                       | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |                            |  3451 |   458K|       |  5297   (1)| 00:00:01 |
|   1 |  TEMP TABLE TRANSFORMATION               |                            |       |       |       |            |          |
|   2 |   LOAD AS SELECT (CURSOR DURATION MEMORY)| SYS_TEMP_0FD9D8B57_CE643DD |       |       |       |            |          |
|*  3 |    HASH JOIN                             |                            |  3567 |   181K|       |  5278   (1)| 00:00:01 |
|   4 |     VIEW                                 |                            |  1205 | 10845 |       |  5141   (1)| 00:00:01 |
|   5 |      HASH UNIQUE                         |                            |  1205 |   448K|       |  5141   (1)| 00:00:01 |
|*  6 |       HASH JOIN SEMI                     |                            |  1205 |   448K|       |  5140   (1)| 00:00:01 |
|*  7 |        HASH JOIN                         |                            |  1205 |   258K|  8432K|  2981   (1)| 00:00:01 |
|   8 |         TABLE ACCESS FULL                | SIGNING                    | 57164 |  7759K|       |   342   (1)| 00:00:01 |
|   9 |         TABLE ACCESS FULL                | EDITION                    |  288K |    22M|       |   960   (1)| 00:00:01 |
|  10 |        TABLE ACCESS FULL                 | PARTICIPATION              |  342K |    52M|       |  2157   (1)| 00:00:01 |
|  11 |     TABLE ACCESS FULL                    | FEDERATION_CARD            | 57163 |  2400K|       |   137   (1)| 00:00:01 |
|* 12 |   HASH JOIN                              |                            |  3451 |   458K|       |    19   (6)| 00:00:01 |
|  13 |    VIEW                                  |                            |  3451 |   229K|       |    10  (10)| 00:00:01 |
|  14 |     HASH GROUP BY                        |                            |  3451 |   104K|       |    10  (10)| 00:00:01 |
|  15 |      VIEW                                |                            |  3567 |   107K|       |     9   (0)| 00:00:01 |
|  16 |       TABLE ACCESS FULL                  | SYS_TEMP_0FD9D8B57_CE643DD |  3567 |   149K|       |     9   (0)| 00:00:01 |
|  17 |    VIEW                                  |                            |  3567 |   236K|       |     9   (0)| 00:00:01 |
|  18 |     TABLE ACCESS FULL                    | SYS_TEMP_0FD9D8B57_CE643DD |  3567 |   149K|       |     9   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------------
```

Evaluación general:

- A continuación usaremos todas las medidas implementadas para comprobar cuanto mejora el total desde el 8K sin índices hasta el nuevo 16K manteniendo el pctfree y creando los índices señalados:

```
TIME CONSUMPTION: 16960,4 milliseconds.
CONSISTENT GETS: 31604,2 blocks
```

- sertUna vez realizado la prueba general observamos una reducción muy significativa de los bloques totales y del tiempo de ejecución total. Esto se debe mayormente a los cambios realizados, desde los índices situados, hasta el aumento del tamaño de bloque. Con esto observamos que el proceso total tenía y potencialmente aún tenga mejoras posibles.

5 Conclusiones Finales

- Hemos observado durante la práctica que cuando realizamos reiteradas ejecuciones de un mismo proceso los bloques necesarios se van estabilizando hacia un número X, esto suponemos que se debe al sistema de memoria interna de Oracle. Debido a esto para todos los estudios hemos cogido siempre el primer dato obtenido ya que es el que realmente accede a la base de datos pura y no a un sistema de memoria intermedio.

- De cara a los resultados obtenidos hemos determinado que la mayoría de las operaciones que considerábamos iban a mejorar el desempeño general, realmente podían llegar a empeorarlo, por ello nuestras conclusiones son que en general el full

scan es la solución más optima en la mayor parte de los casos. Salvo puntualidades donde observamos que indizar ayuda a el proceso de búsqueda/ inserción.

- Para resolver los problemas que creíamos que se podía evitar nos hemos basado en el uso de hints con índices ya que no hemos encontrado ninguna oportunidad de poner en práctica los clústeres. Ninguna situación de join de tablas frecuente ni con atributos lo suficientemente identificativos como para considerar un clúster como una solución viable. Lo mismo a la hora de modificar los datos de pctfree.

- Además de esa puntualidad durante la práctica hemos tenido bastantes problemas para estudiar los datos necesitando en varias ocasiones consultar tanto en las diapositivas de clases como en manuales de internet. Hemos aumentado considerablemente nuestro nivel de conocimiento pero nos ha requerido más tiempo del que habíamos estimado

- En lo respectivo a las demás prácticas, de forma general nos han supuesto bastante costosas llegando a quitarnos demasiado tiempo para otras asignaturas. Qui´zas reducir la cantidad de trabajo podría ayudar a solventar este problema. En cuanto a la dificultad de las mismas, la práctica 1 fue la más problemática debido a nuestro desconocimiento general, pero una vez aprendimos las mecánicas básicas fue generalmente bien.

- En general es una asignatura interesante y las prácticas ayudan a afianzar los conocimientos recibidos.