TT2: AXUSTE DA BASE DE DATOS

Administracion de Bases de Datos – Traballos tutelados
Curso 2023/2024

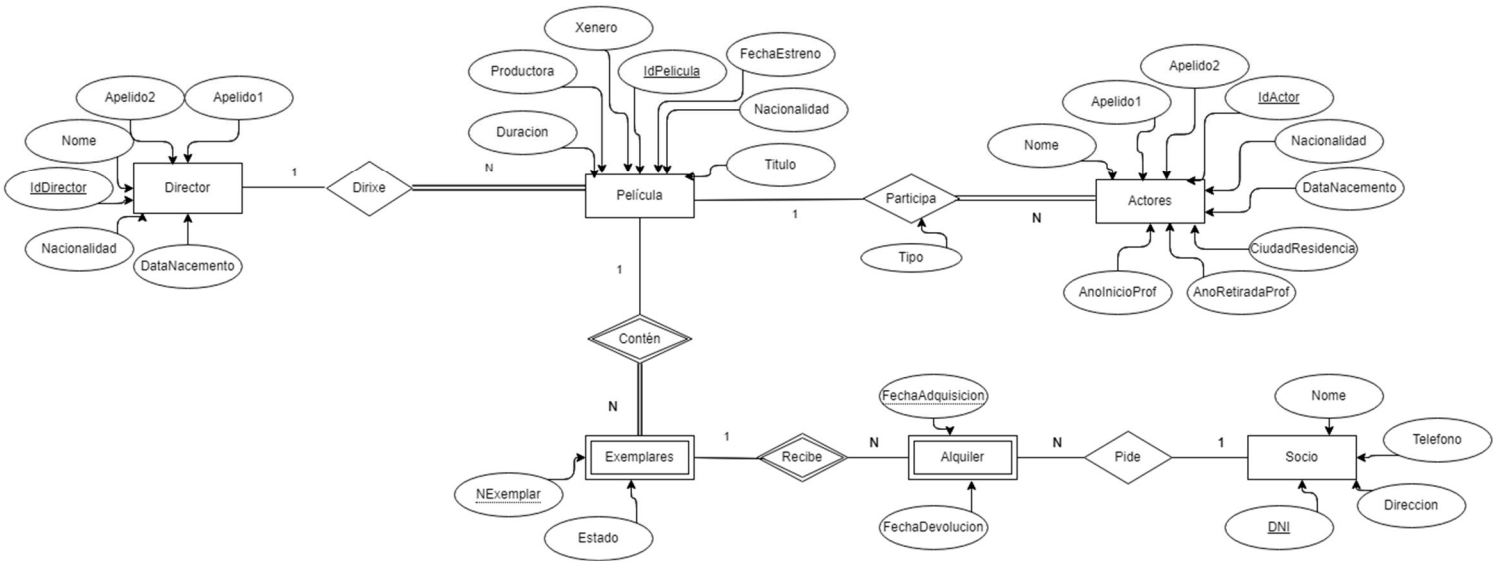
Olveira Miniño, Óscar | 54224120X | oscar.olveira@udc.es
Ansemil Barrós, Fátima María | 49676360V | fatima.ansemil@udc.es

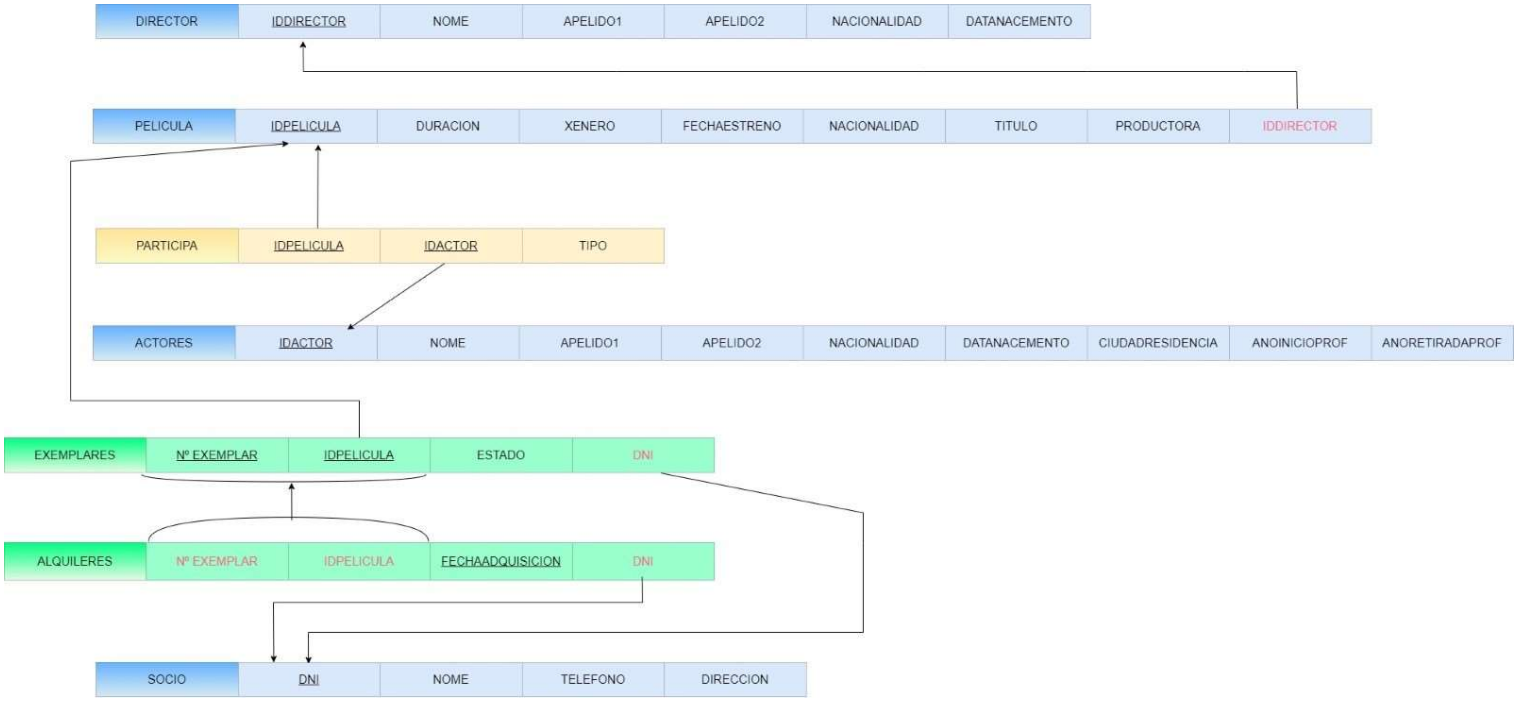
Índice:

1. INTRODUCION

Primeiramente imos determinar a carga de traballo da base de datos. Para iso, especificaremos cinco sentenzas **SELECT** pensadas para representar operacións comúns que se poderían realizar na vida real sobre a nosa base de datos.
A continuación, analizaremos os plans de execución de cada consulta **SELECT** antes dos axustes, empregando ademais en dúas delas, sentenzas hints de Oracle. Posteriormente, levaremos a cabo unha serie de axustes de optimización para ver os cambios que houbo e a mellora do rendemento en cada unha delas. Finalmente, tamén mediremos o tempo de execución dunha sentenza **DML** (antes e despois dos cambios) sobre unha táboa que recibiu cambios e así avaliar o impacto da optimización. Este proceso permitiranos entender as ventaxas ou consecuencias dos axustes realizados.

2. ESQUEMA RELACIONAL E MODELO RELACIONAL:





3. SENTENZAS SELECT, PLANS DE EXECUCIÓN E OPTMIZACIÓNS

Neste apartado, amosaremos cada unha das sentenzas que fixemos, explicaremos o seu propósito e xustificaremos as posibles optimizacións aplicadas según os plans de execución mostrados antes e despois dos axustes.

1º CONSULTA: Obter o nome e apelidos do director xunto co titulo e duración da película que dirixiu que teña como xénero 'Comedia'. Esta consulta nun caso real sería un usuario buscaría películas de comedia e lle interese saber o director de esa película xa que si o dirixiu alguén famoso ou que lle guste un director como dirixe, pois interesaralle máis esa película.

A sentencia sería a seguinte:

```
SELECT d.nome, d.apelido1, p.titulo, p.duracion
FROM director d JOIN pelicula p ON d.iddirector=p.iddirector
WHERE p.xenero = 'Comedia';
```

- **Resultado** da sentencia sería a seguinte:

```
OSCAROLVEIRA@abd>
select d.nome, d.apelido1, p.titulo, p.duracion
  2  from director d join pelicula p on d.iddirector=p.iddirector
  3  where p.xenero = 'Comedia';

NOME                 APELIDO1             TITULO                                   DURACION
-------------------- -------------------- ---------------------------------------- ----------
Eva                  Romero               Locuras en Mumbai                            92
Laura                Sánchez              Risas en Toronto                             98
Rocío                Martínez             Risas en el Palacio                          85
Rocío                Martínez             Risas en Londres                             85
Laura                Martínez             Locuras en Sydney                            88
Laura                Romero               Locuras en la Ciudad                         96
Javier               Hernández            Locuras en Ciudad de México                  88
Daniel               Díaz                 Locuras en Varsovia                          75
Elena                Romero               Risas en París                               95
Marta                Martínez             Locuras en Delhi                             75
María José           Martínez             Risas en Helsinki                            92
Juan                 Sánchez              Risas sin Fin                                92
María José           Sánchez              Locuras en Bollywood                         88
Ana Belén            Pérez                Risas en Ciudad del Cabo                     75
Javier               Pérez                Noche de Locura                              95
Laura                Martínez             Locuras en Teherán                           92
Marta                Gómez                Risas en Toronto                             92
Fernando             Gómez                Risas en Río                                 88
Pedro                Romero               Locuras en Zagreb                            92

19 filas seleccionadas.
```

- **Plan de execucion** da consulta antes de optimizar:

```
OSCAROLVEIRA@abd>
select d.nome, d.apelido1, p.titulo, p.duracion
  2  from director d join pelicula p on d.iddirector=p.iddirector
  3  where p.xenero = 'Comedia';

Plan de Ejecución
----------------------------------------------------------
Plan hash value: 3919706412

-------------------------------------------------------------------------------------------
| Id  | Operation                     | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |             |    19 |  1292 |    10  (10)| 00:00:01 |
|   1 |  MERGE JOIN                   |             |    19 |  1292 |    10  (10)| 00:00:01 |
|   2 |   TABLE ACCESS BY INDEX ROWID| DIRECTOR    |   100 |  2700 |     2   (0)| 00:00:01 |
|   3 |    INDEX FULL SCAN            | PK_DIRECTOR |   100 |       |     1   (0)| 00:00:01 |
|*  4 |   SORT JOIN                   |             |    19 |   779 |     8  (13)| 00:00:01 |
|*  5 |    TABLE ACCESS FULL          | PELICULA    |    19 |   779 |     7   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   4 - access("D"."IDDIRECTOR"="P"."IDDIRECTOR")
       filter("D"."IDDIRECTOR"="P"."IDDIRECTOR")
   5 - filter("P"."XENERO"='Comedia')
```

- **Plan de execución** da consulta usando os hints: /*+ **use_nl(d p)***/

```
OSCAROLVEIRA@abd>
select /*+ use_nl(d p)*/ d.nome, d.apelido1, p.titulo, p.duracion
  2  from director d join pelicula p on d.iddirector=p.iddirector
  3  where p.xenero = 'Comedia';

Plan de Ejecución
----------------------------------------------------------
Plan hash value: 44238485

-------------------------------------------------------------------------------------------
| Id  | Operation                     | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |             |    19 |  1292 |    26   (0)| 00:00:01 |
|   1 |  NESTED LOOPS                 |             |    19 |  1292 |    26   (0)| 00:00:01 |
|   2 |   NESTED LOOPS                |             |    19 |  1292 |    26   (0)| 00:00:01 |
|*  3 |    TABLE ACCESS FULL          | PELICULA    |    19 |   779 |     7   (0)| 00:00:01 |
|*  4 |    INDEX UNIQUE SCAN          | PK_DIRECTOR |     1 |       |     0   (0)| 00:00:01 |
|   5 |   TABLE ACCESS BY INDEX ROWID| DIRECTOR    |     1 |    27 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   3 - filter("P"."XENERO"='Comedia')
   4 - access("D"."IDDIRECTOR"="P"."IDDIRECTOR")

Hint Report (identified by operation id / Query Block Name / Object Alias):
Total hints for statement: 1 (U - Unused (1))
---------------------------------------------------------------------------

   3 -  SEL$58A6D7F6 / P@SEL$1
         U -  use_nl(d p)
```

Neste caso, as diferencias serían:

- Obrigamos a usar joins con bucles anidados polo que agora non fai un merge join
- O coste co hint aumenta xa que nas operacións do join, necesita recorrer máis veces o bucle (con merge ao ser as táboas case do mesmo tamaño redúcese o nº de operacións xa que se fusionan)
- Na táboa Director pasa de explorar todo o índice a buscar por igualdade en índice único

- **Optmización** da consulta:

  Decidimos crear un índice no atributo xénero xa que asi soamente recorrería as películas que se filtre polo xénero indicado:

  CREATE INDEX idx_xenero_pelicula ON Pelicula(xenero);

```
OSCAROLVEIRA@abd> select d.nome, d.apelido1, p.titulo, p.duracion
from director d join pelicula p on d.iddirector=p.iddirector
  3  where p.xenero = 'Comedia';

Plan de Ejecución
------------------------------------------------------------
Plan hash value: 3955527112

-----------------------------------------------------------------------------------------------------------
| Id  | Operation                            | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                     |                     |    19 |  1292 |     7  (15)| 00:00:01 |
|   1 |  MERGE JOIN                          |                     |    19 |  1292 |     7  (15)| 00:00:01 |
|   2 |   TABLE ACCESS BY INDEX ROWID        | DIRECTOR            |   100 |  2700 |     2   (0)| 00:00:01 |
|   3 |    INDEX FULL SCAN                   | PK_DIRECTOR         |   100 |       |     1   (0)| 00:00:01 |
|*  4 |   SORT JOIN                          |                     |    19 |   779 |     5  (20)| 00:00:01 |
|   5 |    TABLE ACCESS BY INDEX ROWID BATCHED| PELICULA            |    19 |   779 |     4   (0)| 00:00:01 |
|*  6 |     INDEX RANGE SCAN                 | IDX_XENERO_PELICULA |    19 |       |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   4 - access("D"."IDDIRECTOR"="P"."IDDIRECTOR")
       filter("D"."IDDIRECTOR"="P"."IDDIRECTOR")
   6 - access("P"."XENERO"='Comedia')
```

Como se pode observar, agora non recorre por completo a táboa Película, soamente recorre as películas filtradas polo xénero indicado e polo que o coste diminuíu un pouco con respecto á consulta sen optimizar.

2º CONSULTA: Neste caso o propósito seria **encontrar todas as películas dun xénero en específico as que se estrearon despois dunha data elixida.** Isto podería ser un caso real para un sistema de datos dun videoclub que quere recomendar novas películas dun xénero en concreto para os seus usuarios

A sentenza capaz de obter os datos requiridos polo enunciado seria esta:
SELECT Titulo, FechaEstreno
FROM Pelicula
WHERE Xenero = 'Acción' AND FechaEstreno > TO_DATE('2020-01-01', 'YYYY-MM-DD');

- **Resultado** desta sentencia sería a que podemos ver a continuación:

```
SQL> SELECT Titulo, FechaEstreno
  2  FROM Pelicula
  3  WHERE Xenero = 'Acción' AND FechaEstreno > TO_DATE('2020-01-01', 'YYYY-MM-DD');

TITULO                                   FECHAEST
---------------------------------------- --------
Aventura Explosiva                       07/08/24
Aventura Extrema                         11/05/25
Kung Fu Panda                            07/06/25
Misión Explosiva                         10/10/23
Misión Explosiva 2                       08/10/23
Misión Explosiva 3                       08/02/25
Misión Explosiva 4                       11/03/23
Misión Imposible 10                      02/08/24
Misión Imposible 11                      11/08/25
Misión Imposible 8                       07/01/24
Misión Imposible 9                       04/02/23
Misión Peligrosa                         03/05/23
Misión Peligrosa 2                       08/10/23
Misión Relámpago                         09/07/23
Misión al Límite                         07/10/24
Operación Relámpago                      09/07/25

16 filas seleccionadas.
```

- **Plan de execución** da consulta antes de optimizar:

```
EXPLAIN PLAN FOR
  2  SELECT Titulo, FechaEstreno
  3  FROM Pelicula
  4  WHERE Xenero = 'Acción' AND FechaEstreno > TO_DATE('2020-01-01', 'YYYY-MM-DD');

Explicado.

SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 2159660392

-----------------------------------------------------------------------------
| Id  | Operation         | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |          |    15 |   525 |     3   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| PELICULA |    15 |   525 |     3   (0)| 00:00:01 |
-----------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------

   1 - filter("XENERO"='Acción' AND "FECHAESTRENO">TO_DATE(' 2020-01-01
              00:00:00', 'syyyy-mm-dd hh24:mi:ss'))

14 filas seleccionadas.
```

Como se pode ver, o plan indica que a consulta vai ler toda a táboa “Película” (acceso completo á táboa) . O custo da consulta é 3, é dicir non é un custo excesivamente grande.

OPTIMIZACION: Para a optimización da sentenza da nosa táboa Película, decidimos crear un índice coas columnas Xénero e FechaEstreno. Este índice, ao cal chamamos **idx_pelicula_xenero_fechaestreno,** creemos que pode mellorar a eficiencia das consultas que buscan películas por Xénero e FechaEstreno, xa que en lugar de realizar un acceso completo á táboa, a base de datos pode usar o índice para atopar as filas que cumpran os criterios de búsqueda moito máis rápido. (O comando de creacion do indice mostrase na imaxe)

```
CREATE INDEX idx_pelicula_xenero_fechaestreno
  2  ON Pelicula (Xenero, FechaEstreno);

Índice creado.

SQL> explain plan for
SELECT Titulo, FechaEstreno
  3  FROM Pelicula
  4  WHERE Xenero = 'Acción'
  5    AND FechaEstreno > TO_DATE('2020-01-01', 'YYYY-MM-DD');

Explicado.

SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------
------------------------------------------------
Plan hash value: 796760126

--------------------------------------------------------------------------------------------------------------
| Id  | Operation              | Name                          | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |                               |    15 |   525 |     2   (0)| 00:00:01 |
|*  1 |  VIEW                  | index$_join$_001              |    15 |   525 |     2   (0)| 00:00:01 |
|*  2 |   HASH JOIN            |                               |       |       |            |          |
|*  3 |    INDEX RANGE SCAN    | IDX_PELICULA_XENERO_FECHAESTRENO |    15 |   525 |     1   (0)| 00:00:01 |
|   4 |    INDEX FAST FULL SCAN| UK_TITULO_PRODUCTORA          |    15 |   525 |     1   (0)| 00:00:01 |
--------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("XENERO"='Acción' AND "FECHAESTRENO">TO_DATE(' 2020-01-01 00:00:00', 'syyyy-mm-dd
              hh24:mi:ss'))
   2 - access(ROWID=ROWID)
   3 - access("XENERO"='Acción' AND "FECHAESTRENO">TO_DATE(' 2020-01-01 00:00:00', 'syyyy-mm-dd
              hh24:mi:ss') AND "FECHAESTRENO" IS NOT NULL)

20 filas seleccionadas.
```

Como se pode ver o optimizador de Oracle decidiu usar o índice porque mellora significativamente o rendemento da consulta ao permitir un acceso máis eficiente aos datos.
As diferenzas con respecto ao plan de execución anterior son as seguintes:

- Ahora emprégase un hash join para unir as táboas que resultan das operacións de búsqueda e escaneo do índice
- Lévase a cabo un escaneo e unha búsqueda rápida do índice creado. Como xa se mencionou esto é útil para atopar máis rápidamente as entradas que satisfagan a condición da consulta sen ter que ler toda a táboa
- Como se amosa no plan de execución, se reduce a cantidade de datos que se deben ler e procesar, diminuíndo o custo da operación (medido en termos de CPU e tempo).

3º CONSULTA: Obter o titulo da pelicula que teñan como xénero 'Drama', xunto co nome e o primer apelido do actor que participa en ela e ver o tipo de actor que é en esa pelicula. Esto nun caso real, o usuario buscaras peliculas do xenero de drama e mirar si en algunha de esas peliculas participa algún actor/actriz que lle interese ou sexa famoso e ademais ver o rol que desempeña en ela.

A sentenza que obtén os datos que se piden sería:

```
SELECT a.nome, a.apel1, p.tipo, pe.titulo
FROM actores a JOINparticipa p ON a.idactor=p.idactor
        JOIN pelicula pe ON p.idpelicula=pe.idpelicula
WHERE xenero = 'Drama' ;
```

- O **resultado** da sentenza sería a seguinte:

```
OSCAROLVEIRA@abd>
select a.nome, a.apel1, p.tipo, pe.titulo
  2  from actores a join participa p on a.idactor=p.idactor
  3                join pelicula pe on p.idpelicula=pe.idpelicula
  4  where xenero = 'Drama' ;

NOME                 APEL1                TIPO                 TITULO
-------------------- -------------------- -------------------- --------------------------------
Víctor               Vicente              Secundario           Lágrimas del Ganges
Richard              Parra                Anti-Héroe           Lágrimas del Kalahari
Peter                Gallardo             Secundario           Lágrimas del Café
Marina               García               Protagonista         Los Minions
Martin               Pérez                Anti-Héroe           Lágrimas en el Silencio
William              Díaz                 Secundario           Secretos Enterrados
Sunita               Gutiérrez            Secundario           Lágrimas del Pasado
Carlos               Ramos                Secundario           Secretos Desenterrados
Irina                Serrano              Secundario           Lágrimas del Corazón
Lin                  Castro               Secundario           Lágrimas en Budapest
Francisco            Nuñez                Protagonista         Lágrimas del Alma
Svetlana             Medina               Secundario           Lágrimas en Buenos Aires
Anita                Santos               Secundario           Lágrimas del Invierno
William              Cruz                 Héroe                Lágrimas del Océano
Robert               Gallego              De Reparto           Lágrimas del Tango
Mario                Reyes                Secundario           Lágrimas del Desierto
Irina                Aguilar              Secundario           Lágrimas en Venecia
Abdul                Herrero              Secundario           Lágrimas del Pasado
Sandra               Benitez              Secundario           Lágrimas en Rosario

19 filas seleccionadas.
```

- O **plan de execución** antes de optimizar:

```
OSCAROLVEIRA@abd>
select a.nome, a.apel1, p.tipo, pe.titulo
  2  from actores a join participa p on a.idactor=p.idactor
  3                join pelicula pe on p.idpelicula=pe.idpelicula
  4  where xenero = 'Drama' ;

Plan de Ejecución
----------------------------------------------------------
Plan hash value: 561197753

-------------------------------------------------------------------------------------------
| Id  | Operation                     | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |           |    19 |  1767 |    13   (8)| 00:00:01 |
|*  1 |  HASH JOIN                    |           |    19 |  1767 |    13   (8)| 00:00:01 |
|   2 |   MERGE JOIN                  |           |   100 |  5600 |     6  (17)| 00:00:01 |
|   3 |    TABLE ACCESS BY INDEX ROWID| ACTORES   |   100 |  2500 |     2   (0)| 00:00:01 |
|   4 |     INDEX FULL SCAN           | PK_ACTOR  |   100 |       |     1   (0)| 00:00:01 |
|*  5 |    SORT JOIN                  |           |   100 |  3100 |     4  (25)| 00:00:01 |
|   6 |     TABLE ACCESS FULL         | PARTICIPA |   100 |  3100 |     3   (0)| 00:00:01 |
|*  7 |   TABLE ACCESS FULL           | PELICULA  |    19 |   703 |     7   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("P"."IDPELICULA"="PE"."IDPELICULA")
   5 - access("A"."IDACTOR"="P"."IDACTOR")
       filter("A"."IDACTOR"="P"."IDACTOR")
   7 - filter("PE"."XENERO"='Drama')

Note
-----
   - this is an adaptive plan
```

- **Optimización da consulta:**
  Para axustar a consulta, puidemos reusar o índice anteriormente creado sobre o atributo xénero e creamos outro sobre o id do actor, o nome e o primer apelido para asi obter mellorar os tempos nos joins xa que indexando a táboa, faceríase menos operacións. Ademais este índice pode servir para futuras sentenzas onde se poda filtrar polo nome e apelido do actor.

  CREATE INDEX idx_xenero_pelicula ON Pelicula(xenero);
  CREATE INDEX idx_actor ON Actores(idactor, nome, apel1);

```
OSCAROLVEIRA@abd>
select a.nome, a.apel1, p.tipo, pe.titulo
  2  from actores a join participa p on a.idactor=p.idactor
  3                join pelicula pe on p.idpelicula=pe.idpelicula
  4  where xenero = 'Drama' ;

Plan de Ejecución
----------------------------------------------------------
Plan hash value: 555398271

-------------------------------------------------------------------------------------------------------
| Id  | Operation                             | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                      |                     |    19 |  1767 |     8   (0)| 00:00:01 |
|*  1 |  HASH JOIN                            |                     |    19 |  1767 |     8   (0)| 00:00:01 |
|*  2 |   HASH JOIN                           |                     |    19 |  1292 |     7   (0)| 00:00:01 |
|   3 |    TABLE ACCESS BY INDEX ROWID BATCHED| PELICULA            |    19 |   703 |     4   (0)| 00:00:01 |
|*  4 |     INDEX RANGE SCAN                  | IDX_XENERO_PELICULA |    19 |       |     1   (0)| 00:00:01 |
|   5 |    TABLE ACCESS FULL                  | PARTICIPA           |   100 |  3100 |     3   (0)| 00:00:01 |
|   6 |   INDEX FULL SCAN                     | IDX_ACTOR           |   100 |  2500 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("A"."IDACTOR"="P"."IDACTOR")
   2 - access("P"."IDPELICULA"="PE"."IDPELICULA")
   4 - access("PE"."XENERO"='Drama')

Note
-----
   - this is an adaptive plan
```

Como se pode observar, agora fai un hash join en vez de un merge join entre as táboas Actores e participa polo que non necesita recorrer toda a táboa. Ademais na táboa Película tampouco accede á táboa completa xa que co índice no atributo xénero soamente accede as películas que pertencen ao xénero filtrado. Gracias a acceder de forma máis simplificada ás táboas, tamén se reduciu o costo da consulta.

4º CONSULTA: Neste caso planteamos outra sentenza que busca **atopar unha lista de todas as películas que foron alquiladas co seu estado actual e a súa data de devolución.** Esto podería ser un caso real para un sistema de datos dun videoclub que quere saber que películas foron alquiladas, en que estado se encontran e cando teñen que ser devoltas.

A sentenza que obtén os datos requeridos sería esta:

```
SELECT  P.Titulo, P.FechaEstreno, E.ESTADO, A.FECHADEVOLUCION
FROM Alquiler A
JOIN Exemplares E ON A.IDPELICULA = E.IDPELICULA AND A.NEXEMPLAR = E.NEXEMPLAR
JOIN Pelicula P ON E.IDPELICULA = P.IdPelicula
WHERE E.ESTADO IN ('OPTIMO', 'BUENO')
ORDER BY A.FECHADEVOLUCION DESC;
```

- **Resultado** que proporciona é o seguinte :

```
SQL> SELECT  P.Titulo, P.FechaEstreno, E.ESTADO, A.FECHADEVOLUCION
  2  FROM Alquiler A
  3  JOIN Exemplares E ON A.IDPELICULA = E.IDPELICULA AND A.NEXEMPLAR = E.NEXEMPLAR
  4  JOIN Pelicula P ON E.IDPELICULA = P.IdPelicula
WHERE E.ESTADO IN ('OPTIMO', 'BUENO')
  6  ORDER BY A.FECHADEVOLUCION DESC;

TITULO                                   FECHAEST ESTADO          FECHADEV
---------------------------------------- -------- --------------- --------
Locuras en Teherán                       07/10/25 BUENO           05/04/24
Lágrimas del Kalahari                    02/07/25 BUENO           05/04/24
Los Minions                              07/07/25 OPTIMO          02/04/24
Secretos Ocultos                         03/05/23 BUENO           02/04/24
Lágrimas del Alma                        05/03/23 BUENO           02/04/24
Lágrimas del Alma                        05/03/23 BUENO           02/04/24
Lágrimas en Budapest                     09/12/23 BUENO           02/04/24
Amor en Dublin                           02/05/25 BUENO           02/04/24
Locuras en Bollywood                     05/10/24 OPTIMO          29/03/24
Lágrimas del Alma                        05/03/23 BUENO           25/03/24
Locuras en Teherán                       07/10/25 BUENO           21/03/24
Secretos Ocultos                         03/05/23 BUENO           21/03/24
Los Minions                              07/07/25 BUENO           21/03/24
Amor en Ciudad de México                 12/05/24 BUENO           21/03/24
Lágrimas del Kalahari                    02/07/25 BUENO           20/03/24
Amor Eterno                              09/05/23 BUENO           18/03/24
Risas en Toronto                         10/12/24 BUENO           18/03/24
Amor en Dublin                           02/05/25 BUENO           16/03/24
Amor en Vancouver                        01/04/25 BUENO           14/03/24
Aventuras en California                  12/03/23 BUENO           14/03/24
Lágrimas del Café                                 BUENO           14/03/24
Lágrimas del Café                                 BUENO           14/03/24
Lágrimas del Alma                        05/03/23 BUENO           12/03/24
Secretos Ocultos                         03/05/23 BUENO           12/03/24
Amor en Ciudad de México                 12/05/24 BUENO           12/03/24
Secretos Ocultos                         03/05/23 BUENO           12/03/24
Los Minions                              07/07/25 BUENO           12/03/24
Locuras en Teherán                       07/10/25 BUENO           12/03/24
Amor en Dublin                           02/05/25 BUENO           01/01/01
Amor en Dublin                           02/05/25 BUENO           01/01/01
Lágrimas en Budapest                     09/12/23 BUENO           01/01/01
Lágrimas en Budapest                     09/12/23 BUENO           01/01/01
Lágrimas en Budapest                     09/12/23 BUENO           01/01/01
Lágrimas del Alma                        05/03/23 BUENO           01/01/01
Lágrimas del Alma                        05/03/23 OPTIMO          01/01/01
Lágrimas del Alma                        05/03/23 BUENO           01/01/01
Risas en Río                             04/12/24 BUENO           01/01/01
Amor Eterno                              09/05/23 BUENO           01/01/01
Amor Eterno                              09/05/23 BUENO           01/01/01
Amor Eterno                              09/05/23 OPTIMO          01/01/01
Amor Eterno                              09/05/23 BUENO           01/01/01
Secretos Ocultos                         03/05/23 BUENO           01/01/01
Secretos Ocultos                         03/05/23 BUENO           01/01/01
Risas en Toronto                         10/12/24 BUENO           01/01/01
Risas en Toronto                         10/12/24 BUENO           01/01/01
Risas en Toronto                         10/12/24 BUENO           01/01/01
Risas en Toronto                         10/12/24 BUENO           01/01/01
```

- **Plan de execución** sería o seguinte:

```
SQL> EXPLAIN PLAN FOR
SELECT  P.Titulo, P.FechaEstreno, E.ESTADO, A.FECHADEVOLUCION
  3  FROM Alquiler A
  4  JOIN Exemplares E ON A.IDPELICULA = E.IDPELICULA AND A.NEXEMPLAR = E.NEXEMPLAR
  5  JOIN Pelicula P ON E.IDPELICULA = P.IdPelicula
WHERE E.ESTADO IN ('OPTIMO', 'BUENO')
  7  ORDER BY A.FECHADEVOLUCION DESC;

Explicado.

SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Plan hash value: 2307398764

-----------------------------------------------------------------------------------------------
| Id  | Operation                     | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |            |    50 |  4050 |    10  (20)| 00:00:01 |
|   1 |  SORT ORDER BY                |            |    50 |  4050 |    10  (20)| 00:00:01 |
|*  2 |   HASH JOIN                   |            |    50 |  4050 |     9  (12)| 00:00:01 |
|   3 |    MERGE JOIN                 |            |    50 |  2200 |     6  (17)| 00:00:01 |
|*  4 |     TABLE ACCESS BY INDEX ROWID| EXEMPLARES |    50 |  1050 |     2   (0)| 00:00:01 |
|   5 |      INDEX FULL SCAN          | PK_EX      |   100 |       |     1   (0)| 00:00:01 |
|*  6 |     SORT JOIN                 |            |   100 |  2300 |     4  (25)| 00:00:01 |
|   7 |      TABLE ACCESS FULL        | ALQUILER   |   100 |  2300 |     3   (0)| 00:00:01 |
|   8 |    TABLE ACCESS FULL          | PELICULA   |   100 |  3700 |     3   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("E"."IDPELICULA"="P"."IDPELICULA")
   4 - filter("E"."ESTADO"='BUENO' OR "E"."ESTADO"='OPTIMO')
   6 - access("A"."IDPELICULA"="E"."IDPELICULA" AND "A"."NEXEMPLAR"="E"."NEXEMPLAR")
       filter("A"."NEXEMPLAR"="E"."NEXEMPLAR" AND "A"."IDPELICULA"="E"."IDPELICULA")

Note
-----
   - this is an adaptive plan

27 filas seleccionadas.
```

Neste plan de execución inclúense unha serie de operacións entre as cales distinguimos unha unión HASH JOIN, unha unión MERGE JOIN e un acceso completo á taboa. O custo total asociado a esta consulta é relativamente baixo, con valores individuais que oscilan entre 1 e 10. Ademais o tempo estimado para cada operación é moi curto.

OPTIMIZACION: Para a optimización desta consulta decidimos crear o seguinte índice, ao que chamamos **idx_alquiler_exemplares.** Está definido sobre as columnas IDPELICULA, NEXEMPLAR e FECHADEVOLUCION da táboa Alquiler, co obxectivo de mellorar o rendemento da consulta dada.

CREATE INDEX idx_alquiler_exemplares on Alquiler(IdPelicula, NEXEMPLAR, FechaDevolucion);

```
SQL> EXPLAIN PLAN FOR
  2  SELECT  P.Titulo, P.FechaEstreno, E.ESTADO, A.FECHADEVOLUCION
  3  FROM Alquiler A
  4  JOIN Exemplares E ON A.IDPELICULA = E.IDPELICULA AND A.NEXEMPLAR = E.NEXEMPLAR
  5  JOIN Pelicula P ON E.IDPELICULA = P.IdPelicula
WHERE E.ESTADO IN ('OPTIMO', 'BUENO')
  7  ORDER BY A.FECHADEVOLUCION DESC;

Explicado.

SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------
Plan hash value: 3278363352

-------------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name                    | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                         |    72 |  5832 |     8  (25)| 00:00:01 |
|   1 |  SORT ORDER BY                |                         |    72 |  5832 |     8  (25)| 00:00:01 |
|*  2 |   HASH JOIN                   |                         |    72 |  5832 |     7  (15)| 00:00:01 |
|   3 |    MERGE JOIN                 |                         |    72 |  3168 |     4  (25)| 00:00:01 |
|*  4 |     TABLE ACCESS BY INDEX ROWID| EXEMPLARES              |    72 |  1512 |     2   (0)| 00:00:01 |
|   5 |      INDEX FULL SCAN          | PK_EX                   |   100 |       |     1   (0)| 00:00:01 |
|*  6 |     SORT JOIN                 |                         |   100 |  2300 |     2  (50)| 00:00:01 |
|   7 |      INDEX FULL SCAN          | IDX_ALQUILER_EXEMPLARES |   100 |  2300 |     1   (0)| 00:00:01 |
|   8 |    TABLE ACCESS FULL          | PELICULA                |   100 |  3700 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("E"."IDPELICULA"="P"."IDPELICULA")
   4 - filter("E"."ESTADO"='BUENO' OR "E"."ESTADO"='OPTIMO')
   6 - access("A"."IDPELICULA"="E"."IDPELICULA" AND "A"."NEXEMPLAR"="E"."NEXEMPLAR")
       filter("A"."NEXEMPLAR"="E"."NEXEMPLAR" AND "A"."IDPELICULA"="E"."IDPELICULA")
```

As principais diferencias unha vez optimizada a consulta son as seguintes:

- Ao ter un índice que inclúe IDPELICULA e NEXEMPLAR, aceleramos a xunta entre as táboas Alquiler e Exemplares, xa que o índice permite un acceso rápido ás filas relevantes.
- Ademais, incluír FECHADEVOLUCION no índice tamén axuda a optimizar a cláusula ORDER BY, xa que as filas xa estarán preordenadas polo índice, reducindo así a necesidade de operacións de ordenación adicionais.
- Tamén destacar o paso 7, no cal en vez de facerse un acceso completo á táboa, fai emprego do índice creado, implicando do mesmo xeito unha redución do custo da execución.

5º CONSULTA: Para este caso planteamos unha sentenza coa cal o videoclub podería **consultar as películas que foron alquiladas por un socio en concreto, empreganndo para a identificación do mesmo o seu DNI**. A sentenza que permite obter esto é a seguinte:

```
SELECT P.Titulo, FECHADQUISICION, FECHADEVOLUCION , NEXEMPLAR
FROM Alquiler A
JOIN Socios S ON A.DNI = S.DNI
JOIN Pelicula P ON A.IDPELICULA = P.IdPelicula
WHERE S.DNI = '44580534F';
```

- **O resultado** da consulta sería este que vemos a continuación:

```
SQL> SELECT P.Titulo, FECHADQUISICION, FECHADEVOLUCION , NEXEMPLAR
  2  FROM Alquiler A
  3  JOIN Socios S ON A.DNI = S.DNI
  4  JOIN Pelicula P ON A.IDPELICULA = P.IdPelicula
  5  WHERE S.DNI = '44580534F';

TITULO                                   FECHADQU FECHADEV NEXE
---------------------------------------- -------- -------- ----
Lágrimas del Alma                        08/03/24 02/04/24 0007
Lágrimas del Alma                        12/02/24 02/04/24 0009
Trama Oscura                             16/01/24 12/03/24 0002
```

- O seu **plan de execución**, sería o seguinte:

```
SQL> EXPLAIN PLAN FOR
  2  SELECT P.Titulo, FECHADQUISICION, FECHADEVOLUCION , NEXEMPLAR
  3  FROM Alquiler A
  4  JOIN Socios S ON A.DNI = S.DNI
  5  JOIN Pelicula P ON A.IDPELICULA = P.IdPelicula
  6  WHERE S.DNI = '44580534F';

Explicado.

SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Plan hash value: 2106598247

-------------------------------------------------------------------------------------------------
| Id  | Operation              | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |                     |     4 |   280 |     5  (0)| 00:00:01 |
|*  1 |  HASH JOIN             |                     |     4 |   280 |     5  (0)| 00:00:01 |
|*  2 |   TABLE ACCESS FULL    | ALQUILER            |     4 |   164 |     3  (0)| 00:00:01 |
|   3 |   VIEW                 | index$_join$_004    |   100 |  2900 |     2  (0)| 00:00:01 |
|*  4 |    HASH JOIN           |                     |       |       |           |          |
|   5 |     INDEX FAST FULL SCAN| PK_PELICULA         |   100 |  2900 |     1  (0)| 00:00:01 |
|   6 |     INDEX FAST FULL SCAN| UK_TITULO_PRODUCTORA |   100 |  2900 |     1  (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("A"."IDPELICULA"="P"."IDPELICULA")
   2 - filter("A"."DNI"='44580534F')
   4 - access(ROWID=ROWID)

20 filas seleccionadas.
```

Neste esquema podemos ver que, nesta execución, se usa unha combinación de operacións de acceso completo á táboa, ademais dunha unión hash xunto cun escaneo rápido e completo do índice, para recuperar os datos necesarios. En canto aos custos podemos observar que estes oscilan entre valores moi baixos, polo que en xeral, podemos dicir que ten un moi bo rendemento.

- **Plan de execución** da consulta usando os hints: **/*+ use_nl(a s p)*/**

```
SQL> EXPLAIN PLAN FOR
SELECT /*+ USE_NL(A S P) */ P.Titulo, FECHADQUISICION, FECHADEVOLUCION , NEXEMPLAR
  3  FROM Alquiler A
  4  JOIN Socios S ON A.DNI = S.DNI
  5  JOIN Pelicula P ON A.IDPELICULA = P.IdPelicula
  6  WHERE S.DNI = '44580534F';

Explicado.

SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------
Plan hash value: 3450732440

-----------------------------------------------------------------------------------------
| Id  | Operation                    | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |              |     3 |   210 |     6   (0)| 00:00:01 |
|   1 |  NESTED LOOPS                |              |     3 |   210 |     6   (0)| 00:00:01 |
|   2 |   NESTED LOOPS               |              |     3 |   210 |     6   (0)| 00:00:01 |
|*  3 |    TABLE ACCESS FULL         | ALQUILER     |     3 |   123 |     3   (0)| 00:00:01 |
|*  4 |    INDEX UNIQUE SCAN         | PK_PELICULA  |     1 |       |     0   (0)| 00:00:01 |
|   5 |   TABLE ACCESS BY INDEX ROWID| PELICULA     |     1 |    29 |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   3 - filter("A"."DNI"='44580534F')
   4 - access("A"."IDPELICULA"="P"."IDPELICULA")

Hint Report (identified by operation id / Query Block Name / Object Alias):
Total hints for statement: 2 (U - Unused (2))
---------------------------------------------------------------------------

   1 -  SEL$7EC67111 / S@SEL$1
         U -  USE_NL(A S P)

   3 -  SEL$7EC67111 / A@SEL$1
         U -  USE_NL(A S P)

28 filas seleccionadas.
```

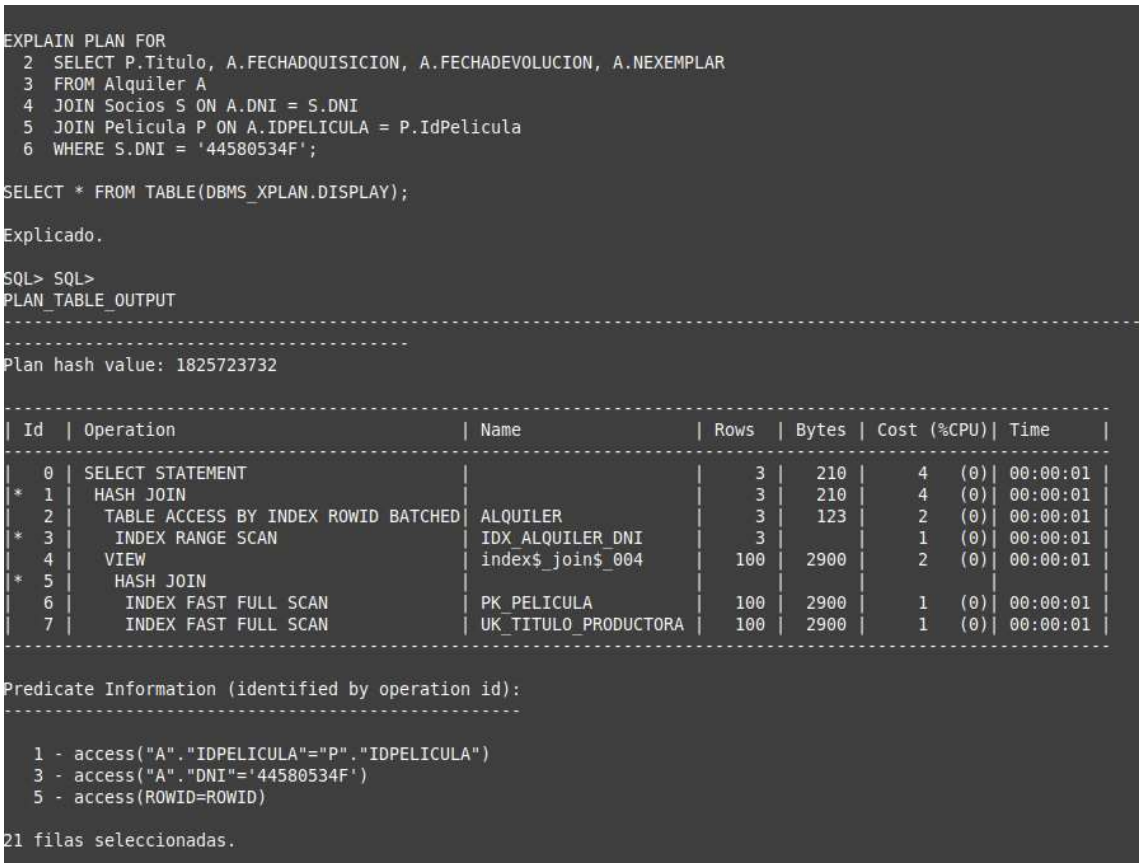
As principales diferencias con respecto ao plan de execución anterior son as seguintes:

- "USE_NL" non mellorou a optimización porque o método de "nested loops" (xuntas anidadas) é menos eficiente para conxuntos de datos grandes en comparación cos "hash joins"
- Os "nested loops" implican moitas operacións de busca repetitivas, o que aumenta o custo de execución.
- Non se fai un escaneo rápido e completo do índice senón que se usa un TABLE ACCESS BY INDEX ROWID no que se usa o ROWID almacenado no índice para buscar a fila completa na táboa.

OPTIMIZACION: Para a optimización desta sentenza decidimos de novo levar a cabo a creación doutro índice, ao cal chamamos **idx_alquiler_dni.**

CREATE INDEX idx_alquiler_dni ON Alquiler(DNI);

Este foi creado na columna DNI da táboa Alquiler para mellorar o rendemento das consultas que inclúen esta columna, especialmente nas unións (JOIN) coa táboa Socios. Ao ter un índice en DNI, a base de datos pode localizar rapidamente as filas de Alquiler que coinciden co DNI específico dun socio, sen ter que escanear toda a táboa



```
EXPLAIN PLAN FOR
  2  SELECT P.Titulo, A.FECHADQUISICION, A.FECHADEVOLUCION, A.NEXEMPLAR
  3  FROM Alquiler A
  4  JOIN Socios S ON A.DNI = S.DNI
  5  JOIN Pelicula P ON A.IDPELICULA = P.IdPelicula
  6  WHERE S.DNI = '44580534F';

SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

Explicado.

SQL> SQL>
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------
--------------------------------------
Plan hash value: 1825723732

-------------------------------------------------------------------------------------------------------------------
| Id  | Operation                             | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                      |                     |     3 |   210 |     4   (0)| 00:00:01 |
|*  1 |  HASH JOIN                            |                     |     3 |   210 |     4   (0)| 00:00:01 |
|   2 |   TABLE ACCESS BY INDEX ROWID BATCHED | ALQUILER            |     3 |   123 |     2   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN                   | IDX_ALQUILER_DNI    |     3 |       |     1   (0)| 00:00:01 |
|   4 |   VIEW                                | index$_join$_004    |   100 |  2900 |     2   (0)| 00:00:01 |
|*  5 |    HASH JOIN                          |                     |       |       |            |          |
|   6 |     INDEX FAST FULL SCAN              | PK_PELICULA         |   100 |  2900 |     1   (0)| 00:00:01 |
|   7 |     INDEX FAST FULL SCAN              | UK_TITULO_PRODUCTORA |   100 |  2900 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("A"."IDPELICULA"="P"."IDPELICULA")
   3 - access("A"."DNI"='44580534F')
   5 - access(ROWID=ROWID)

21 filas seleccionadas.
```

Como se pode ver no plan de execución (INDEX_RANGE SCAN) o optimizador de Oracle está a aproveitar o índice creado na columna DNI da táboa ‘Alquiler’ para buscar as filas relacionadas co `DNI` específico ‘44580534F’.

Sen o índice, a base de datos tería que escanear toda a táboa ‘Alquiler’ en busca das filas co ‘DNI’ especificado, o que é moito máis custoso en termos de recursos e tempo.

3. CONSULTA DML

Como se explicou anteriormente, vamos a comprobar se os cambios que se fixeron na base de datos afectaron á sentencia DML que vamos a levar a cabo.
A consulta DML faríase sobre a táboa **Pelicula** xa que foi nunha das táboa na que fixemos máis operacións de optmización .

Para ver o impacto que tivo vamos a mostrar o plan de execucion antes e despois de optmizar ademais de lanzar un bucle facendo esa operación DML varias veces tamen antes e depois da optmización para medir o tempo e asi ver as ventaxas ou consecuencias que tiveron os axustes na base de datos:

A consulta a facer sería insetar peliculas:
INSERT INTO pelicula VALUES('123456abc', 113, 'Aventura', to_date('05/05/2024', 'dd-mm-yyyy'), 'España', 'En busca Galicia', 'Universal Pictures', 'b6a1c6743');

Primeiro mostramos o plan de execución desta consulta sin ningunha optmización:

```
OSCAROLVEIRA@abd> insert into pelicula values('123456abc', 113, 'Aventura', to_date('05/05/2024', 'dd-m

1 fila creada.


Plan de Ejecución
----------------------------------------------------------


-------------------------------------------------------------------------------------
| Id  | Operation                | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | INSERT STATEMENT         |          |     1 |    81 |     1   (0)| 00:00:01 |
|   1 |  LOAD TABLE CONVENTIONAL | PELICULA |       |       |            |          |
-------------------------------------------------------------------------------------
```

Ahora mostramos os tempos de execución da sentencia antes de optmizar:

```
OSCAROLVEIRA@abd> set timing on;
begin
  2  for i in 1 .. 1000 loop
insert into pelicula values('abc'||i, 113, 'Aventura', to_date('05/05/2024', 'dd-mm-yyyy'),
'España', 'En Galicia'||i, 'Universal Pictures', 'b6a1c6743');
  5  end loop;
  6  end;
  7  /

Procedimiento PL/SQL terminado correctamente.

Transcurrido: 00:00:00.11
```

As seguintes duas imáxenes fariamos o mesmo que nas primeiras pero aplicando os cambios que fixemos sobre a táboa **Pelicula** e ver a diferencia que hai entre a BD optmizada e a BD sen optmizar:

```
OSCAROLVEIRA@abd> INSERT INTO pelicula VALUES('123456abc', 113, 'Aventura', to_date('05/05/2024', 'dd-

1 fila creada.


Plan de Ejecución
----------------------------------------------------------


-------------------------------------------------------------------------------------
| Id  | Operation                | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | INSERT STATEMENT         |          |     1 |    81 |     1   (0)| 00:00:01 |
|   1 |  LOAD TABLE CONVENTIONAL | PELICULA |       |       |            |          |
-------------------------------------------------------------------------------------
```

```
OSCAROLVEIRA@abd> set timing on
begin
  2  for i in 1 .. 1000 loop
insert into pelicula values('abc'||i, 113, 'Aventura', to_date('05/05/2024', 'dd-mm-yyyy'),
'España', 'En Galicia'||i, 'Universal Pictures', 'b6a1c6743');
  5  end loop;
  6  end;
  7  /

Procedimiento PL/SQL terminado correctamente.

Transcurrido: 00:00:00.14
```

Como se pode observar, a creación de índices na táboa **Pelicula** tivo un pequeno impacto na carga de operacións (aumentou de 11 centésimas a 14 centésimas de segundo) polo que poderiamos manter as operacións de optmización levadas a cabo na táboa xa que as consultas optmizadas sobre as películas recibiron unha boa mellora.

4. CONCLUSIÓNS

Unha vez chegados a este punto podemos concluír que en xeral, todas as consultas puidéron ser optimizadas (algunhas sufrindo máis cambios que outra) creando soamente un índice por consulta ou reusando algún xa creado polo que así evitamos a sobreindexación da base de datos (algo pouco beneficioso).

Tamén se pode ver que nos nosos casos (xeralmente en case todos en calquera base de datos), o uso de hints empeora a execución das consultas xa que o propio Oracle xa dispón de un optimizador propio polo que solo se daría en casos moi concretos onde o uso de hints chegue a mellorar o plan de execución da consulta.

Ademais, como se pode comprobar e como se mencionou no apartado anterior, poderíamos manter os axustes sobre a base de datos xa que en unha carga alta de operacións, non se percibe gran diferencia (3 centésimas) polo que vale a pena manter esos axustes xa que as consultas na gran maioría conseguiron mellorar o seus costos.

Por último, tamén cabe destacar que probamos outros índices sobre algunhas táboas (por exemplo sobre o atributo Duracion na táboa Pelicula) pero como Oracle non as chegou a usar na execución das consultas, decidimos non usalos.