

1.

```

select /*+ GATHER_PLAN_STATISTICS */ /* prac4s10 consulta-1 */
A.ID, A.titulo,round(A.drama), H.genero
from PELISHIST H, PELISAHORA A
where A.ID = H.ID and (round(A.drama) = 43 or round(A.drama) = 50);
-----
SET LINESIZE 130
SELECT *
FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR
(format => 'ALLSTATS LAST'));

```

```

PLAN_TABLE_OUTPUT
1 SQL_ID 2axwxuc415p0k, child number 2
2 -----
3 select /*+ GATHER_PLAN_STATISTICS */ /* prac4s10 consulta-1 */ A.ID,
4 A.titulo,round(A.drama), H.genero from PELISHIST H, PELISAHORA A where
5 A.ID = H.ID and (round(A.drama) = 43 or round(A.drama) = 50)
6
7 Plan hash value: 1925733202
8
9 -----
10 | Id | Operation | Name | Starts | E-Rows | A-Rows | A-Time | Buffers |
11 |-----|-----|-----|-----|-----|-----|-----|-----|
12 | 0 | SELECT STATEMENT | | | 1 | | 7 | 00:00:00.01 | 25 |
13 | 1 | NESTED LOOPS | | | 1 | 1 | 7 | 00:00:00.01 | 25 |
14 | 2 | NESTED LOOPS | | | 1 | 1 | 7 | 00:00:00.01 | 18 |
15 | 3 | INLIST ITERATOR | | | 1 | | 7 | 00:00:00.01 | 9 |
16 | 4 | TABLE ACCESS BY INDEX ROWID BATCHED | PELISAHORA | 2 | 1 | 7 | 00:00:00.01 | 9 |
17 |* 5 | INDEX RANGE SCAN | IDX_PELISAHORA_FUN_DRAMA | 2 | 1 | 7 | 00:00:00.01 | 2 |
18 |* 6 | INDEX UNIQUE SCAN | PK_PELISHIST | 7 | 1 | 7 | 00:00:00.01 | 9 |
19 | 7 | TABLE ACCESS BY INDEX ROWID | PELISHIST | 7 | 1 | 7 | 00:00:00.01 | 7 |
20 -----
21
22 Predicate Information (identified by operation id):
23 -----
24
25 5 - access(("A"."SYS_NC00016$"=43 OR "A"."SYS_NC00016$"=50))
26 6 - access("A"."ID"="H"."ID")
27
28 Note
29 ----
30 - this is an adaptive plan

```

Sec 1.1.

Según nos marca la práctica:

- A-Rows : Filas reales (cuando ejecuta esa operación) de la operación de esa línea:
- Cuando están bien estimadas las E-Rows suele coincidir con el núm. filas de E-Rows multiplicado por *Starts*: si hay mucha diferencia es que ese plan no va a ser eficiente.

Las que están bien estimadas son: (PK_PELISHIST, PELISHIST)

Id 6: Starts 7 * E-Rows 1 = 7 -> 7 A-Rows

Id 7: Starts 7 * E-Rows 1 = 7 -> 7 A -Rows

Sec 1.2.

Usa muchos buffers porque al acceder por índice no hace falta acceder a toda la memoria de la tabla

Sec 1.3.

```

select /*+ GATHER_PLAN_STATISTICS */ /* prac4s10 consulta-2 */
A.ID, A.titulo, A.genero
from PELISAHORA A, PELISHIST H
where A.ID = H.ID and A.ID > 100 and A.genero = 'Terror' ;
SELECT *
FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR
(format => 'ALLSTATS LAST'));

```

PLAN_TABLE_OUTPUT									
1	SQL_ID	47dkngvpstcwm, child number 0							
2	-----								
3	select /*+ GATHER_PLAN_STATISTICS */ /* prac4s10 consulta-2 */ A.ID,								
4	A.titulo, A.genero from PELISAHORA A, PELISHIST H where A.ID = H.ID and								
5	A.ID > 100 and A.genero = 'Terror'								
6									
7	Plan hash value: 234004151								
8									
9	-----								
10	Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers	
11	-----								
12	0	SELECT STATEMENT		1		3	00:00:00.01	10	
13	1	NESTED LOOPS		1	3	3	00:00:00.01	10	
14	* 2	TABLE ACCESS BY INDEX ROWID BATCHED	PELISAHORA	1	3	3	00:00:00.01	5	
15	3	BITMAP CONVERSION TO ROWIDS		1		4	00:00:00.01	1	
16	* 4	BITMAP INDEX SINGLE VALUE	IDX_PELISAHORA_BIT_GENERO	1		1	00:00:00.01	1	
17	* 5	INDEX UNIQUE SCAN	PK_PELISHIST	3	1	3	00:00:00.01	5	
18	-----								
19									
20	Predicate Information (identified by operation id):								
21	-----								
22									
23	2	filter("A"."ID">100)							
24	4	access("A"."GENERO"='Terror')							
25	5	access("A"."ID"="H"."ID")							
26		filter("H"."ID">100)							

- La segunda es más eficiente porque tiene dos nested loops mientras que la primera solo tiene un único nested loop
- La segunda es más eficiente porque saca menos líneas reales
- La segunda es más eficiente ya que al tener un nested loop y al acceder por bitmap y index unique scan accederá menos a memoria
- Me parece más eficiente esta consulta por lo comentado en el apartado a,b,c

SECCIÓN 2

1.a) desc user_indexes

index_name, table_owner, table_name

Nombre	¿Nulo?	Tipo
INDEX_NAME	NOT NULL	VARCHAR2(128)
INDEX_TYPE		VARCHAR2(27)
TABLE_OWNER	NOT NULL	VARCHAR2(128)
TABLE_NAME	NOT NULL	VARCHAR2(128)
TABLE_TYPE		VARCHAR2(11)
UNIQUENESS		VARCHAR2(9)
COMPRESSION		VARCHAR2(13)
PREFIX_LENGTH		NUMBER
TABLESPACE_NAME		VARCHAR2(30)
INI_TRANS		NUMBER
MAX_TRANS		NUMBER
INITIAL_EXTENT		NUMBER
NEXT_EXTENT		NUMBER
MIN_EXTENTS		NUMBER
MAX_EXTENTS		NUMBER
PCT_INCREASE		NUMBER
PCT_THRESHOLD		NUMBER
INCLUDE_COLUMN		NUMBER
FREELISTS		NUMBER
FREELIST_GROUPS		NUMBER
PCT_FREE		NUMBER
LOGGING		VARCHAR2(3)
BLEVEL		NUMBER
LEAF_BLOCKS		NUMBER
DISTINCT_KEYS		NUMBER
AVG_LEAF_BLOCKS_PER_KEY		NUMBER
AVG_DATA_BLOCKS_PER_KEY		NUMBER
CLUSTERING_FACTOR		NUMBER
STATUS		VARCHAR2(8)
NUM_ROWS		NUMBER
SAMPLE_SIZE		NUMBER
LAST_ANALYZED		DATE
DEGREE		VARCHAR2(40)
INSTANCES		VARCHAR2(40)
PARTITIONED		VARCHAR2(3)
TEMPORARY		VARCHAR2(1)
GENERATED		VARCHAR2(1)
SECONDARY		VARCHAR2(1)
BUFFER_POOL		VARCHAR2(7)
FLASH_CACHE		VARCHAR2(7)
CELL_FLASH_CACHE		VARCHAR2(7)
USER_STATS		VARCHAR2(3)
DURATION		VARCHAR2(15)

1.b)

```
select INDEX_NAME, INDEX_TYPE, TABLE_NAME, UNIQUENESS
from user_indexes --- Índices de tu usuario
Where TABLE_NAME = 'PELISAHORA';
```

	INDEX_NAME	INDEX_TYPE	TABLE_NAME	UNIQUENESS
1	PK_PELISACTUAL	NORMAL	PELISAHORA	UNIQUE
2	IDX_PELISAHORA_UNICO_TITULO	NORMAL	PELISAHORA	UNIQUE
3	IDX_PELISAHORA_BIT_GENERO	BITMAP	PELISAHORA	NONUNIQUE
4	IDX_PELISAHORA_FUN_DRAMA	FUNCTION-BASED NORMAL	PELISAHORA	NONUNIQUE

1.b.1) Porque puede a ver más de un género/drama mientras que un título es único.

1.c)

```
select INDEX_NAME, INITIAL_EXTENT, BLEVEL, LEAF_BLOCKS, NUM_ROWS
from user_indexes
Where TABLE_NAME = 'PELISHIST';
```

	INDEX_NAME	INITIAL_EXTENT	BLEVEL	LEAF_BLOCKS	NUM_ROWS
1	IDX_PELISHIST_FUN_DRAMA	65536	0	1	521
2	IDX_PELISHIST_BIT_GENERO	65536	0	1	25
3	PK_PELISHIST	65536	1	2	521
4	IDX_PELISHIST_UNICO_TITULO	65536	1	3	521

1.c.1) Porque no están contruidos como un árbol si no que son un único elemento.

1.c.2) Porque solo hay 25 tipos de género, lo podemos comprobar haciendo un count de los tipos de genero

```
SELECT COUNT(*) FROM PELISHIST
GROUP BY GENERO
```

	COUNT(*)
1	24
2	30
3	18
4	33
5	10
6	4
7	7
8	77
9	15
10	6
11	41
12	166
13	1
14	4
15	2
16	1
17	1
18	4
19	1
20	6
21	3
22	2
23	10
24	54
25	1

1d)

```
select INDEX_NAME, DISTINCT_KEYS, AVG_LEAF_BLOCKS_PER_KEY,
AVG_DATA_BLOCKS_PER_KEY, CLUSTERING_FACTOR
from user_indexes Where TABLE_NAME = 'PELISHIST';
```

INDEX_NAME	DISTINCT_KEYS	AVG_LEAF_BLOCKS_PER_KEY	AVG_DATA_BLOCKS_PER_KEY	CLUSTERING_FACTOR
1 IDX_PELISHIST_FUN_DRAMA	52	1	5	310
2 IDX_PELISHIST_BIT_GENERO	25	1	1	25
3 PK_PELISHIST	521	1	1	194
4 IDX_PELISHIST_UNICO_TITULO	521	1	1	516

1.d.1) 521 indica cuántas pelis hay, cada peli tiene un título distinto, de ahí el 521 con su respectivo género, que como ya hemos comentado antes había 25 tipos de género.

1.d.2)

Nos indica como de ordenada está la tabla, para saber si está bien ordenada nos tenemos que fijar en el número de bloque y en el de cantidad de registros de la tabla, en caso de que se acerque al primero estará bien ordena, mientras que si se acerca a los valores del segundo estará mal ordenada.

1.e)

OWNER	INDEX_NAME	BLEVEL	LEAF_BLOCKS	NUM_ROWS
1 SYS	I_WRI\$_OPTSTAT_H_OBJ#_ICOL#_ST	2	3630	451592
2 SYS	I_WRI\$_OPTSTAT_H_ST	2	1939	451592
3 SYS	I_SOURCE1	2	731	287454
4 SYS	I_DEPENDENCY2	2	680	141079
5 SYS	I_DEPENDENCY1	2	663	141079
6 SYS	I_COL1	2	706	120264
7 SYS	I_COL3	1	273	120264
8 SYS	I_COL2	1	356	120264
9 SYS	I_ACCESS1	1	441	112260
10 SYS	I_WRI\$_OPTSTAT_HH_ST	2	369	106015
11 SYS	I_WRI\$_OPTSTAT_HH_OBJ_ICOL_ST	2	699	106015
12 SYS	I_ARGUMENT1	2	792	105304
13 SYS	I_ARGUMENT2	2	565	105304
14 SYS	WRH\$_SYSMETRIC_HISTORY_INDEX	2	807	87561
15 SYS	I_OBJ1	1	207	73183
16 SYS	I_OBJ5	2	919	73183
17 SYS	I_OBJ2	2	913	73183
18 SYS	I_OBJ4	1	310	73183
19 SYS	I_IDL_UB11	1	198	61838
20 SYS	I_H_OBJ#_COL#	1	324	55661
21 SYS	I_OBJAUTH2	1	252	52273
22 SYS	I_OBJAUTH1	1	188	52273
23 SYS	I_SETTINGS1	1	144	50088

Buscando en la web he encontrado:

Al no estar comprimido, un leaf block del índice irá guardando entradas con sus claves y el rowid de la fila en la tabla correspondiente, por ejemplo:

```
online,0,AAAPvCAAFAAAAFaAA
aonline,0,AAAPvCAAFAAAAFaAAg
online,0,AAAPvCAAFAAAAFaAAI
online,2,AAAPvCAAFAAAAFaAAm
online,3,AAAPvCAAFAAAAFaAAq
online,3,AAAPvCAAFAAAAFaAAt
```

La compresión de índices será muy beneficiosa cuando las columnas iniciales del índice tienen muchos valores repetidos dentro de un leaf block, y cuanto más grande el índice más espacio ocupado se liberará.

La columna de num_rows nos muestra las filas por cada índice.

En este caso nos muestran los leaf block de cada índice y el num_rows de estos mismos, de promedio el primer índice tendrá 124 filas por cada leaf block.

2.- Tabla user_tables :

2.a) Table_name, tablespace_name y status

Nombre	¿Nulo?	Tipo
TABLE_NAME	NOT NULL	VARCHAR2(128)
TABLESPACE_NAME		VARCHAR2(30)
CLUSTER_NAME		VARCHAR2(128)
IOT_NAME		VARCHAR2(128)
STATUS		VARCHAR2(8)
PCT_FREE		NUMBER
PCT_USED		NUMBER
INI_TRANS		NUMBER
MAX_TRANS		NUMBER
INITIAL_EXTENT		NUMBER
NEXT_EXTENT		NUMBER
MIN_EXTENTS		NUMBER
MAX_EXTENTS		NUMBER
PCT_INCREASE		NUMBER
FREELISTS		NUMBER
FREELIST_GROUPS		NUMBER
LOGGING		VARCHAR2(3)
BACKED_UP		VARCHAR2(1)
NUM_ROWS		NUMBER
BLOCKS		NUMBER
EMPTY_BLOCKS		NUMBER
AVG_SPACE		NUMBER
CHAIN_CNT		NUMBER
AVG_ROW_LEN		NUMBER
AVG_SPACE_FREELIST_BLOCKS		NUMBER
NUM_FREELIST_BLOCKS		NUMBER
DEGREE		VARCHAR2(10)
INSTANCES		VARCHAR2(10)
CACHE		VARCHAR2(5)
TABLE_LOCK		VARCHAR2(8)
SAMPLE_SIZE		NUMBER
LAST_ANALYZED		DATE
PARTITIONED		VARCHAR2(3)
IOT_TYPE		VARCHAR2(12)
TEMPORARY		VARCHAR2(1)
SECONDARY		VARCHAR2(1)
NESTED		VARCHAR2(3)
BUFFER_POOL		VARCHAR2(7)
FLASH_CACHE		VARCHAR2(7)
CELL_FLASH_CACHE		VARCHAR2(7)
ROW_MOVEMENT		VARCHAR2(8)
GLOBAL_STATS		VARCHAR2(3)

2.b)

	TABLE_NAME	NUM_ROWS	BLOCKS	AVG_ROW_LEN
1	PELISHIST	521	244	2228
2	PELISAHORA	142	65	2476
3	NOTIFICACIONES	4	5	85
4	DICCION	9	5	53
5	EMPRESA	0	5	0
6	CLIENTE	0	5	0
7	CLIENTE_N	0	5	0
8	CLIENTE_I	0	5	0
9	MOROSO	0	5	0
10	OFERTA	0	5	0
11	INVIERTE	0	5	0
12	TARJETA	0	5	0
13	COMPRAS	0	5	0
14	PUESTO	0	5	0
15	TIENET	0	5	0
16	TIENETEL	0	5	0

2.b.1)

Blocks: Tamaño de los bloques.

Num_rows: Son las filas que tienen esa tabla

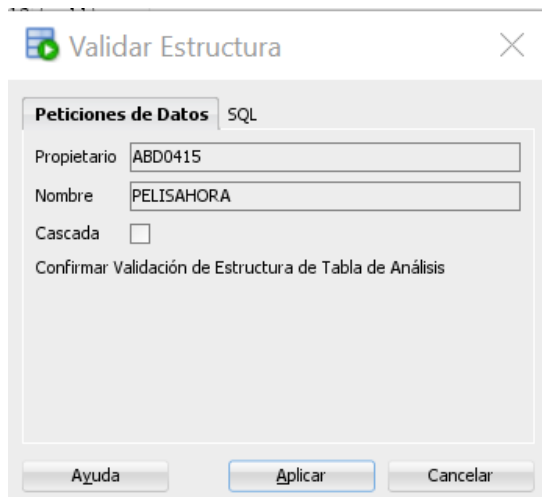
Los bloques de 5 con 0 rows es porque no han superado un límite y Oracle pone ese número por defecto

2.b.2)



Avg_row_len -> longitud media de la fila

Entonces pelishist tendrá una longitud media de las palabras menor que la tabla de pelisahora

SECCIÓN 3.



a) Validado correctamente.

 **Recopilar Estadísticas** 

Peticiones de Datos

SQL

Propietario

ABD0415

Nombre

PELISAHORA

Porcentaje de Ejemplo

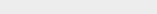
1

Confirmar Recopilación de Estadísticas de Tabla

Ayuda

Aplicar

Cancelar

A confirmation dialog box with a title bar containing a document icon and a close button. The title is "Confirmación". The main text says "Las estadísticas de la tabla 'PELISAHORA' se han recopilado". At the bottom right is an "Aceptar" button.

	Nombre	Valor
1	NUM_ROWS	142
2	BLOCKS	65
3	AVG_ROW_LEN	2476
4	SAMPLE_SIZE	142
5	LAST_ANALYZED	07/05/20
6	LAST_ANALYZED_SINCE	07/05/20

OWNER	TABLE_NAME	COLUMN_NAME	NUM_DISTINCT	LOW_VALUE	HIGH_VALUE
1	ABD0415	PELLISARORA	ID	142	C100E
2	ABD0415	PELLISARORA	TITULO	142	C100E
3	ABD0415	PELLISARORA	SEXERO	17	142
4	ABD0415	PELLISARORA	DESCRIPCION	142	C100E
5	ABD0415	PELLISARORA	ACCION	40	C100
6	ABD0415	PELLISARORA	ANIMACION	10	C100
7	ABD0415	PELLISARORA	AVERTIDAS	50	C100
8	ABD0415	PELLISARORA	CREDITA	71	C100
9	ABD0415	PELLISARORA	DOCUMENTAL	7	C100
10	ABD0415	PELLISARORA	DRAMA	30	C100
11	ABD0415	PELLISARORA	FANTASIA	23	C100
12	ABD0415	PELLISARORA	ROMANTICA	40	C100
13	ABD0415	PELLISARORA	TEJERO	29	C100
14	ABD0415	PELLISARORA	THRILLER	55	C100
15	ABD0415	PELLISARORA	CIENCIAFICCION	12	C100
16	ABD0415	PELLISARORA	SYS_BACKLOG	31	C100

Refrescando cada 5 segundos:

[illegible]

C)Pestaña de Detalles:

	Nombre	Valor
1	CREATED	29/04/20
2	LAST_DDL_TIME	07/05/20
3	OWNER	ABD0415
4	TABLE_NAME	PELISAHORA
5	TABLESPACE_NAME	ESPABD0415
6	CLUSTER_NAME	(null)
7	IOT_NAME	(null)
8	STATUS	VALID
9	PCT_FREE	10
10	PCT_USED	(null)
11	INI_TRANS	1
12	MAX_TRANS	255
13	INITIAL_EXTENT	65536
14	NEXT_EXTENT	1048576
15	MIN_EXTENTS	1
16	MAX_EXTENTS	2147483645
17	PCT_INCREASE	(null)
18	FREELISTS	(null)
19	FREELIST_GROUPS	(null)
20	LOGGING	YES
21	BACKED_UP	N
22	NUM_ROWS	142
23	BLOCKS	65
24	EMPTY_BLOCKS	0
25	AVG_SPACE	0
26	CHAIN_CNT	0
27	AVG_ROW_LEN	2476
28	AVG_SPACE_FREELIST_BLOCKS	0
29	NUM_FREELIST_BLOCKS	0
30	DEGREE	1
31	INSTANCES	1
32	CACHE	N
33	TABLE_LOCK	ENABLED
34	SAMPLE_SIZE	142
35	LAST_ANALYZED	07/05/20
36	PARTITIONED	NO
37	IOT_TYPE	(null)
38	OBJECT_ID_TYPE	(null)
39	TABLE_TYPE_OWNER	(null)
40	TABLE_TYPE	(null)
41	TEMPORARY	N
42	SECONDARY	N

SECCIÓN 4:**a)**

```
EXEC DBMS_STATS.GATHER_INDEX_STATS('ABD0415', 'PK_PELISHIST');
```

Nos permite analizar las estadísticas de los índices de manera más óptima. Sus atributos en este caso serían:

OwnName: Esquema dónde se encuentra la tabla a analizar. ('ABD0415')

Index: Index a analizar. ('pk_pelishist')

Procedimiento PL/SQL terminado correctamente.

b)

Nos permite analizar las estadísticas de tablas ,columnas e índices de manera más óptima. Sus atributos en este caso serían

OwnName: Esquema dónde se encuentra la tabla a analizar. ('ABD0415')

TabName: Nombre de la tabla a analizar. ('pelishist')

Procedimiento PL/SQL terminado correctamente.

c)

```
select OWNER,INDEX_NAME,NUM_ROWS,LAST_ANALYZED,
BLEVEL,LEAF_BLOCKS,DISTINCT_KEYS
from dba_indexes
where owner = 'ABD0415'
and index_name ='PK_PELISHIST';
```

Esta consulta nos selecciona el propietario de la tabla, el nombre del index, las filas de ese index (521 porque hay 521 pelis), las estadísticas de la la tabla (si fuera nulo significaría que las estadísticas de la tabla aún no se han recopilado), el nivel en el que está, los leaf_block y las distinct_keys (una fila por cada distinct_keys, si distinct_keys es un porcentaje más bajo que num_rows significaría que el índice es “inútil”, en este caso no lo es).

	OWNER	INDEX_NAME	NUM_ROWS	LAST_ANALYZED	BLEVEL	LEAF_BLOCKS	DISTINCT_KEYS
1	ABD0415	PK_PELISHIST	521	07/05/20	1	2	521