PROJECT

ZHTHMA 1

A)

Για να βρουμε τα physical reads απο το δισκο των συγκεκριμενων δειγματοληψιων εκτελουμε μια εντολη *select * from v$sysstat where name='physical read total IO requests';* πριν την ερωτηση στη βαση και μια μετα.Η διαφορα στο value του πεδιου που μας επιστρεφεται ειναι ο ζητουμενος αριθμος.

Πιο συγκεκριμενα για την δειγματοληψια :

```
SELECT year, count(*)
 FROM movie
GROUP BY year
ORDER BY year desc;
```

Εχουμε *previous_value=535263 , after_value=538777*

Αρα τα physical read που θα χρειαστουμε ειναι *538777 - 535263=3,514 physical read*

Για την δειγματοληψια :

```
SELECT companyID, count(*)
FROM distributedBy
GROUP BY companyID;
```

Εχουμε *previous_value=541327 , after_value=541966*

Αρα τα physical read που θα χρειαστουμε ειναι *541966- 541327=639 physical read*

*Για να παρουμε πιο ακριβη αποτελεσματα κανουμε clear την cache memory εκτελοντας τις εντολες:

```
alter system checkpoint;
alter system flush shared_pool;
alter system flush buffer_cache;
```

B)

Η εντολη που πρεπει να εκτελεστει για να υπολογισουμε το hit ratio ειναι η εξης:

```
SELECT (P1.value + P2.value - P3.value) / (P1.value + P2.value)
FROM   v$sysstat P1, v$sysstat P2, v$sysstat P3
WHERE  P1.name = 'db block gets'
AND    P2.name = 'consistent gets'
AND    P3.name = 'physical reads'
```

Εκτελοντας αυτη την εντολη παιρνουμε hit ratio= 0.998 αλλα ειναι το συνολικο που εχω μεχρι τωρα στην βαση. Για να παρουμε των συγκεκριμενων ερωτησεων θα κρατησουμε τις τιμες των ‘db block gets' ,'consistent gets', 'physical reads' πριν και μετα την εκτελεση της καθε εντολης.

Αρα εχουμε για την πρωτη δειγματοληψια:

	value1	value2	value3
πριν	8215684	56366219	119613
μετα	8215673	56361280	114713
διαφορα	11	4939	4900

Hit ratio: (P1.value + P2.value - P3.value) / (P1.value + P2.value)=
=(11+4939-4900)/(11+4939) = 50/4950=0,0101

Για την δευτερη δειγματοληψια εχουμε:

	value1	value2	value3
πριν	8217334	56455194	130627
μετα	8216918	56441196	122853
διαφορα	416	13998	7774

Hit ratio: *(P1.value + P2.value - P3.value) / (P1.value + P2.value)=*
=(416+13998-7774)/(416+13998)=6640/14414=0,46

Το συστημα χρειαζεται tuning διοτι αποδεκτες τιμες hit ratio θεωρουνται πανω απο 97-98% ενω τα δικα μας ποσοστα ειναι εξαιρετικα χαμηλα.

C)

Για να βρουμε το μεγεθος των table στη βαση εκτελουμε την εντολη:

```
select owner as "Schema"
  , segment_name as "Object Name"
  , segment_type as "Object Type"
  , round(bytes/1024/1024,2) as "Object Size (Mb)"
  , tablespace_name as "Tablespace"
from dba_segments where segment_name='the_name_of_the table';
```

table_name	**size(mb)**
distributedBy	47
movie	39
people	35
plays	54
producedby	72

Η μνημη cache που χρησιμοποιει η oracle ειναι ≈ 42 mb αρα οι πινακες που χωρανε ολοκληροι στην cache ειναι ο movie και ο people.

D)

Χρησιμοποιουμε την εντολη *alter table movie CACHE;* για να τοποθετησουμε τον πινακα movie στην cache.

Για την εκτελεση της πρωτης διαβαστηκαν 73 block απο το δισκο.
Για την εκτελεση της δευτερης διαβαστηκαν 900 block απο το δισκο.

Βλεπουμε οτι διαβαστηκαν ελαχιστες σελιδες σε σχεση με πριν(3,514) για την δειγματοληψια πανω στη movie ενω για τη δευτερη δειγματοληψια χρειαστηκαν παραπανω I/O αυτο συνεβη διοτι η cache ειναι σχεδον γεματη απο τις πλειαδες της movie.

Για την εκτελεση της πρωτης ερωτησης για δευτερη φορα διαβαστηκαν 300 block απο το δισκο.

Αυτο γινεται διοτι η Oracle χρησιμοποιει αλγοριθμο αντικαταστασης σελιδας LRU και στην εκτελεση της δευτερης ερωτησης οταν χρειαζοταν block για να φερει πλειαδες της distributedBy διεγραφε block με πλειαδες της movie.

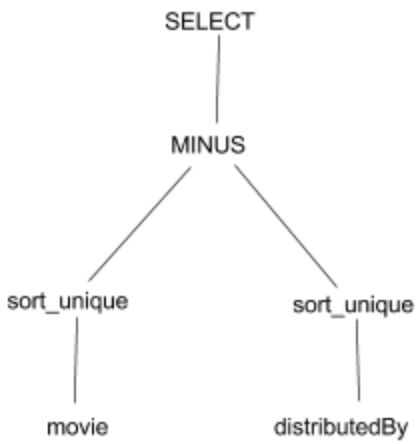
ZHTHMA 2

A)

i)

Το πλανο εκτελεσης για την ερωτηση :
SELECT movieID FROM movie MINUS SELECT movieID FROM distributedBy;
Ειναι το εξης:

Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT		217 K	3464 K
1	MINUS			
2	SORT UNIQUE		217 K	2765 K
3	TABLE ACCESS FULL	MOVIE	217 K	2765 K
4	SORT UNIQUE		143 K	698 K
5	TABLE ACCESS FULL	DISTRIBUTEDBY	143 K	698 K



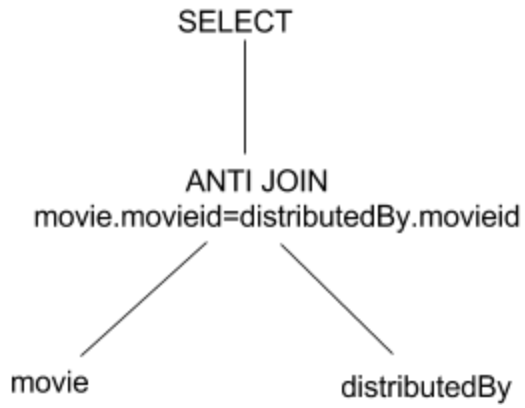
Περιγραφη πλανου εκτελεσης:

Διαβαζουμε ολοκληρο το table movie και με την sort_unique επιστρεφουμε ολα τα μοναδικα movieId που υπαρχουν σε αυτο ομοια και για το table distributedBy.Επειτα εκτελουμε την πραξη minus και γυριζουμε το αποτελεσμα της δειγματοληψιας στο select.

Το πλανο εκτελεσης για την ερωτηση :
SELECT movieID FROM movie WHERE movieID NOT IN
(SELECT movieID FROM distributedBy);

Ειναι το εξης:

Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT		217 K	3828 K
1	HASH JOIN RIGHT ANTI		217 K	3828 K
2	TABLE ACCESS FULL	DISTRIBUTEDBY	143 K	698 K
3	TABLE ACCESS FULL	MOVIE	217 K	2765 K



Περιγραφη πλανου εκτελεσης:
Διαβαζουμε ολοκληρα τα table movie και distributedBy και τους εφαρμοζουμε μια πραξη anti join βασισμενη στην κοινη τους στηλη movieId.Επειτα γυριζουμε το αποτελεσμα της δειγματοληψιας στο select.

ii)

Η δευτερη ερωτηση ειναι πιο γρηγορη διοτι θα προσπελαστουν τα tables μονο για το anti join. Ενω στην πρωτη ερωτηση θα προσπελαστουν τα tables για να εφαρμοσουμε το sort_unique και για να εφαρμοσουμε το minus.
(πιθανο αν η ερωτηση δεν βασιζοταν πανω στο id της σχεσης και βασιζοταν σε αλλο πεδιο που ειχε ομοιες τιμες αναμεσα στις πλοιαδες η sort_unique θα επεστρεφε αποτελεσμα μικροτερου μεγεθους και θα ηταν πιο γρηγορος ο πρωτος τροπος)

Η πρωτη ερωτηση χρειαζεται : 6,36 sec
Η δευτερη ερωτηση χρειαζεται : 5,81 sec

iii)

Προσθετοντας ενα index στο table movie βασισμενο στη στηλη movieId και ενα index στο table distributedBy βασισμενο στη στηλη movieId, ο χρονος του πρωτου ερωτηματος γινεται 3.66 sec ενω ο χρονος της δευτερης ερωτησης γινεται 3,68 sec.

B)

i)

Το πλανο εκτελεσης για την ερωτηση :
SELECT pl.movieID FROM people p, plays pl WHERE p.personID = pl.personID and p.birthYear > χ;
Ειναι το εξης:

Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT		813 K	86 M
1	HASH JOIN		813 K	86 M
2	TABLE ACCESS FULL	PEOPLE	289 K	15 M
3	TABLE ACCESS FULL	PLAYS	810 K	43 M

Το πλανο εκτελεσης για την ερωτηση :
SELECT pl.movieID FROM plays pl WHERE pl.personID IN
(SELECT p.personID FROM people p WHERE p.birthYear > χ);

Ειναι το εξης:

Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT		810 K	83 M
1	HASH JOIN RIGHT SEMI		810 K	83 M

2	VIEW	VW_NSO_1	289 K	14 M
3	TABLE ACCESS FULL	PEOPLE	289 K	15 M
4	TABLE ACCESS FULL	PLAYS	810 K	43 M

Τιμες χ:	1930	1950	1970	1990	1995
Για την 1η	86,47	40,18	15,56	8,18	5,53
Για την 2η	76,91	37,05	15,89	6,94	4,69

Αρα η δευτερη ερωτηση εκτελειται πιο γρηγορα.

ii)

Για να αξιοποιησουμε την διαθεσιμη μνημη θα χωρισουμε την ερωτηση στις ακολουθες δυο ερωτησεις sql :

- *CREATE GLOBAL TEMPORARY TABLE tmp_table ON COMMIT PRESERVE ROWS AS SELECT personid FROM people WHERE birthyear>1950;*

- select pl.movieid from tmp_table p, plays pl where p.personid=pl.personid;

Οι χρονοι που παιρνουμε για τις νεες εντολες ειναι οι εξης(αρκετα βελτιομενοι απο πριν):

Τιμες χ:	1930	1950	1970	1990	1995
	30,1	26,47	13,59	5,3	4,67

ΖΗΤΗΜΑ 3

Α)

Το πλανο εκτελεσης της εντολης
select p.personid from movie m , plays p where p.movieid=m.movieid and m.movieid=0046790;
Μετα την αλλαγη του optimizer mode με την εντολη: *alter session set optimizer_mode=all_rows;*
Ειναι το εξης:

Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT		4022	3179
*1	HASH JOIN		4022	3179
*2	TABLE ACCESS FULL	MOVIE	54	1325
*3	TABLE ACCESS FULL	PLAYS	75	1854

1 - access(“p”.”MOVIEID”=”M”.”MOVIEID”)
2 - filter(“M”.”MOVIEID”=0046790)
3 - filter(“p”.”MOVIEID”=0046790)

Ο χρονος εκτελεσης ειναι 0,07 sec.
Αν προσθεσουμε το keyword DISTINCT στην παραπανω εντολη ο χρονος γινεται 0,06 sec.

B)

Εκτελουμε τις παρακατω εντολες για να δημιουργησουμε το index:

```
CREATE CLUSTER trial_movie2 (movieID NUMERIC(7))
 HASH IS movieID HASHKEYS 1000;
drop table movie;
create table movie
(       movieID numeric(7) not null,
        movieTitle char(110) not null,  color char(45),
        language char(20),
        year numeric(4)

)   CLUSTER trial_movie2 (movieID);
```

Και ξανα κανουμε insert ολες τις πλειαδες στο table του movie.
Το πλανο εκτελεσης ειναι το εξης:

Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT		4372	333K
1	MERGE JOIN CARTESIAN		4372	333K
*2	TABLE FULL ACCESS	PLAYS	75	4875
3	BUFFER SORT		58	754
*4	TABLE ACCESS HASH	MOVIE	58	754

2 - filter("P"."MOVIEID"=0046790)

4 - access("M"."MOVIEID"=0046790)

Ο χρονος εκτελεσης ειναι 0,05 sec.

C)

Εκτελουμε τις εντολες :

```
DROP table plays ;
create table plays
(
        personID char(50) not null,
        movieID numeric(7) not null
)CLUSTER trial_movie2 (movieID);
```

Για να ενωσουμε το plays με το index και επειτα κανουμε insert τις πλειαδες.

Το πλανο εκτελεσης ειναι το εξης:

Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT		18525	1411 K
1	MERGE JOIN CARTESIAN		18525	1411 K
*2	TABLE ACCESS HASH	MOVIE	136	1768
3	BUFFER SORT		136	8840
*4	TABLE ACCESS HASH	PLAYS	136	8840

Ο χρονος εκτελεσης ειναι 0,02 sec.

D)

Το πλανο εκτελεσης της 1ης ερωτησης ειναι:

Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT		62656	7648 K
*1	TABLE ACCESS FULL	MOVIE	62656	7648 K

1 - filter("YEAR">1987)

Το πλανο εκτελεσης της 2ης ερωτησης ειναι:

Id	Operation	Name	Rows	Bytes

0	SELECT STATEMENT		199k	12 M
*1	TABLE ACCESS FULL	PEOPLE	199k	12 M

1 - filter("BIRTHYEAR">1940)

Παρατηρουμε οτι στην ερωτηση πανω στο table movie παρολο που και στις δυο ερωτησεις εχουμε table access full διαβαζουμε μερος των πλειαδων ενω στη δευτερη ερωτηση διαβαζουμε πραγματι ολες τις πλειαδες του table people που βρισκονται στη βαση.

Μετα την προσθηκη των unclustered B+ ευρετηριων τα πλανα εκτελεσης ειναι :

Για την 1η ερωτητηση:

Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT		62656	7648 K
*1	INDEX FAST FULL SCAN	movie_year_idx	62656	7648 K

Ο χρονος εκτελεσης ειναι: 9,81 sec.

Για την 2η ερωτηση:

Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT		199k	12 M
*1	INDEX FAST FULL SCAN	people_birthyear_idx	199k	12 M

Ο χρονος εκτελεσης ειναι: 24,73 sec.
Για να χρησιμοποιηθουν τα ευρετηρια πανω στα γνωρισματα year και birthyear και να μην εχουμε table access full καναμε τα ευρετηρια με διπλο κλειδι(το δευτερο πεδιο ειναι αυτο που θα επιστραφει στο select).Οι εντολες που χρησιμοποιηθηκαν ειναι :

CREATE INDEX movie_year_idx ON movie (year, movieTitle);

CREATE INDEX people_birthyear_idx ON people (birthYear , personName);

Ε)

Το πλανο εκτελεσης της 1ης ερωτησης ειναι:

Id	Operation	Name	Rows	Bytes

0	SELECT STATEMENT		71298	9608 K
*1	TABLE ACCESS FULL	MOVIE	71298	9608 K

1 - filter("MOVIEID"<>0046790 AND "YEAR">1983)

Το πλανο εκτελεσης της 2ης ερωτησης ειναι:

Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT		82	11316
*1	TABLE ACCESS HASH	MOVIE	82	11316

1 - access("MOVIEID"=0046790)
filter("YEAR">1983)

Στη δευτερη ερωτηση χρησιμοποιειται το hash table διοτι η πραξη πανω στο movieID
ειναι ισοτητα αρα μπορουμε να βρουμε τις συγκεκριμενες πλειαδες πολυ γρηγορα.Στην πρωτη ερωτηση δεν χρησιμοποιειται κανενα index διοτι ειναι range search και οι πλειαδες δεν ειναι συσταδοποιημενες αρα θα χρειαστουμε μια προσπελαση στο δισκο για καθε πλειαδα για να παρουμε τις τιμες movieID , movieTitle.

ZHTHMA 4

A)

Το πλανο εκτελεσης για την πρωτη ειναι:

Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT		-	-
*1	INDEX RANGE SCAN	movie_year_idx	-	-

1 - access("YEAR">1987 AND "YEAR" IS NOT NULL)

Ο χρονος εκτελεσης ειναι: 9,58 sec. λιγο βελτιωμενος σε σχεση με πριν (9,81 sec.)

Το πλανο εκτελεσης για την δευτερη ειναι:

Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT		-	-
*1	INDEX RANGE SCAN	people_birthyear_idx	-	-

1 - access("BIRTHYEAR">1940 AND "BIRTHYEAR" IS NOT NULL)
Ο χρονος εκτελεσης ειναι: 24,59 sec. λιγο βελτιομενος σε σχεση με πριν (24,73 sec.)

B)

Το πλανο εκτελεσης ειναι :

Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT		-	-
1	NESTED LOOPS		-	-
2	TABLE ACCESS FULL	PRODUCEDBY	-	-
*3	TABLE ACCESS HASH	MOVIE	-	-

3 - access("M"."MOVIEID"="P"."MOVIEID")
filter("M"."MOVIETITLE"='Black Tuesday (1954)' AND
"M"."MOVIEID"=0046790 AND "M"."MOVIEID"="P"."MOVIEID")

Χρησιμοποιειται το hash table διοτι ειναι συσταδοποιημενο και δεν θα χρειαστει μια προσπελαση στο δισκο για καθε πλειαδα που τηρει τα κριτηρια των συνθηκων. Πιο συγκεκριμενα αν χρησιμοποιουσαμε το B+ index θα επρεπε για καθε πλειαδα που ισχυει "MOVIETITLE"='Black Tuesday (1954)' να προσπελασουμε το δισκο για να παρουμε την πλειαδα και να ελενξουμε αν ικονοποιουνται και οι "M"."MOVIEID"=0046790 AND "M"."MOVIEID"="P"."MOVIEID".

ZHTHMA 5

A)

OPTIMIZATION_MODE =RULE

Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT		715k	81 M
*1	HASH JOIN		715k	81 M
*2	TABLE ACCESS FULL	PEOPLE	40352	2167 K

3	TABLE ACCESS FULL	PLAYS	715k	44 M

1 - access("P"."PERSONID"="PL"."PERSONID")
2 - filter("P"."BIRTHYEAR">1990)

Ο χρονος εκτελεσης ειναι: 24,17 sec

OPTIMIZATION_MODE =FIRST_ROWS

Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT		715k	81 M
*1	HASH JOIN		715k	81 M
2	TABLE ACCESS BY INDEX ROWID BATCHED	PEOPLE	40352	2167 K
*3	INDEX RANGE SCAN	PEOPLE_BIRTHYEAR_IDX	40352	
4	TABLE ACCESS FULL	PLAYS	715 K	44 M

1 - access("P"."PERSONID"="PL"."PERSONID")
3 - access("P"."BIRTHYEAR">1990)

Ο χρονος εκτελεσης ειναι: 23,64 sec

OPTIMIZATION_MODE =ALL_ROWS

Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT		715k	81 M
*1	HASH JOIN		715k	81 M
*2	TABLE ACCESS FULL	PEOPLE	40352	2167 K
3	TABLE ACCESS FULL	PLAYS	715 K	44 M

1 - access("P"."PERSONID"="PL"."PERSONID")
2 - filter("P"."BIRTHYEAR">1990)

Ο χρονος εκτελεσης ειναι: 23,92 sec

B)

OPTIMIZATION_MODE =RULE

Εχουμε το ιδιο execution plan

Ο χρονος εκτελεσης ειναι: 24,26 sec

OPTIMIZATION_MODE =FIRST_ROWS

Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT		715k	81 M
*1	HASH JOIN		715k	81 M
*2	TABLE ACCESS FULL	PEOPLE	40352	2167 K
3	TABLE ACCESS FULL	PLAYS	715k	44 M

1 - access("P"."PERSONID"="PL"."PERSONID")
2 - filter("P"."BIRTHYEAR">1990)

Ο χρονος εκτελεσης ειναι: 24,53 sec

OPTIMIZATION_MODE =ALL_ROWS

Εχουμε το ιδιο execution plan

Ο χρονος εκτελεσης ειναι: 22,95 sec

Μονο στο mode first rows αλλαζει το execution plan και γινεται ιδιο με τα υπολοιπα mode

C)

OPTIMIZATION_MODE =RULE

Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT		113k	12 M
*1	HASH JOIN		113k	12 M
*2	TABLE ACCESS FULL	PEOPLE	40352	2167 K
3	TABLE ACCESS FULL	PLAYS	810k	44 M

1 - access("P"."PERSONID"="PL"."PERSONID")
2 - filter("P"."BIRTHYEAR">1990)

Ο χρονος εκτελεσης ειναι: 16,62 sec

OPTIMIZATION_MODE =FIRST_ROWS

Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT		113 k	12 M
*1	HASH JOIN		113 k	12 M
2	TABLE ACCESS BY INDEX ROWID	PEOPLE	40352	2167 K

	BATCHED			
3	BITMAP CONVERSION TO ROWIDS			
*4	BITMAP INDEX RANGE SCAN	PEOPLE_BIRTHYEAR_IDX		
5	TABLE ACCESS FULL	PLAYS	810K	43 M

1 - access("P"."PERSONID"="PL"."PERSONID")
4 - access("P"."BIRTHYEAR">1990)
filter("P"."BIRTHYEAR">1990)

Ο χρονος εκτελεσης ειναι: 18,02 sec

OPTIMIZATION_MODE =ALL_ROWS

Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT		113 K	12 M
*1	HASH JOIN		113 K	12 M
*2	TABLE ACCESS FULL	PEOPLE	40352	2167 K
3	TABLE ACCESS FULL	PLAYS	810 K	43 M

1 - access("P"."PERSONID"="PL"."PERSONID")
2 - filter("P"."BIRTHYEAR">1990)
Ο χρονος εκτελεσης ειναι: 19,06 sec

Βλεπουμε οτι παρολο που στα mode ALL_ROWS , RULE δεν εχουμε αλλαγη στο execution plan επειδη εχουμε το bit map το μεγεθος των αποτελεσματων του hash join ειναι σημαντικα μικροτερο

D)

Συνοψη χρονων για καθε mode:

	A	**B**	**C**
RULE	24,17 sec	24,26 sec	16,62 sec
FIRST ROWS	23,64 sec	24,53 sec	18,02 sec
ALL ROWS	23,92 sec	22,95 sec	19,06 sec

Σημαντικη διαφορα στους χρονους παρατηρειται μετα την δημιουργια του bit map διοτι μειωνεται το μεγεθος των αποτελεσματων του hash join.

Επισης εχουμε διαφορα στο mode first rows μετα τη διαγραφη του ευρετηριου PEOPLE_BIRTHYEAR_IDX διοτι ειναι το μονο mode που το χρησιμοποιουσε.

E)

Χρησιμοποιουμε τις παρακατω εντολες για να ξαναδημιουργησουμε τον πινακα people και να φτιαξουμε το ευρετηριο:

i) *drop table people;*

ii) CREATE CLUSTER birthyear_cluster_index (birthYear NUMERIC(4));

iii)
```
create table people
(
        personID char(50) not null,
        personName char(50) not null,
        birthYear numeric(4) l,
        deathYear numeric(4)
)CLUSTER birthyear_cluster_index(birthYear) ;
```

iv) create index birthyear_index_cl on cluster birthyear_cluster_index;

Και τα 3 mode εχουν το ακολουθο πλανο εκτελεσης.

Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT		104k	12 M
*1	HASH JOIN		104k	12 M
2	TABLE ACCESS CLUSTER	PEOPLE	37146	2357 K
*3	INDEX RANGE SCAN	BIRTHYEAR_INDEX_CL	2956	
4	TABLE ACCESS FULL	PLAYS	810 K	43 M

1 - access("P"."PERSONID"="PL"."PERSONID")
3 - access("P"."BIRTHYEAR">1990)

Ο χρονος εκτελεσης για το mode rule ειναι: 15.59 sec
Ο χρονος εκτελεσης για το mode first rows ειναι: 15.66 sec
Ο χρονος εκτελεσης για το mode all rows ειναι: 13.78 sec

Παρατηρουμε οτι οι χρονοι ειναι πιο βελτιωμενοι απο πριν. Αυτο οφειλεται στο γεγονος οτι με το unclustered ευρετηριο το range search που εφαρμοζοταν ηταν πιθανο να χρειαζεται ενα I/0 για καθε πλειαδα που ικανοποιουσε τη συνθηκη ωστε να παρει και τα αλλα δεδομενα (person Name κλπ) ενω τωρα που ειναι clustered ως προς το birthyear οι πλειαδες που ικανοποιουν τη συνθηκη βρισκονται σε σειριακες θεσεις και block στη μνημη.

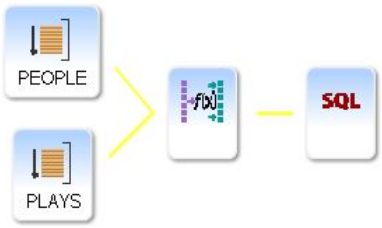
ZHTHMA 6

A)

Πλανο εκτελεσης:

SELECT STATEMENT		0				927K	120M
HASH JOIN		1			3,501	927K	120M
TABLE ACCESS FULL	PEOPLE	2			1,209	33K	2,177K
TABLE ACCESS FULL	PLAYS	3			1,852	927K	60M

γραφημα πλανου εκτελεσης:



Πλανο εκτελεσης για την ιδια sql ερωτηση απο το sql access advisor :

Operation	Object	Li...	Pre...	Pr...	Operation Cost	Estimated Rows	Estimated Bytes
SELECT STATEMENT		0				927K	120M
PX COORDINATOR		1					
PX SEND QC (RANDOM)	:TQ10001	2				927K	120M
HASH JOIN		3			2	927K	120M
PX RECEIVE		4				33K	2,177K
PX SEND BROADCAST	:TQ10000	5				33K	2,177K
PX BLOCK ITERATOR		6				33K	2,177K
TABLE ACCESS FULL	PEOPLE	7			671	33K	2,177K
PX BLOCK ITERATOR		8				927K	60M
TABLE ACCESS FULL	PLAYS	9			1,028	927K	60M

Γραφημα πλανου εκτελεσης απο sql access advisor:

Ουσιαστικα το sql access advisor προτεινει την παραλληλη εκτελεση του statement.

B)

Ο χρονος εκτελεσης αν αποδεκτουμε το execution plan που προτεινεται ειναι 16,56 sec . Βλεπουμε οτι ειναι πιο γρηγορο απο ολες τις περιπτωσεις του ζητηματος 5 εκτος απο την περιπτωση που χρησιμοποιειται clustered index στο γνωρισμα birthYear.

C)

To kostos pou plhrwsame sthn parapanw erwthsh einai oti xreiasthke na prospelasoume ola ta stoixeia apo to disko.

Dikh mou prosegkish gia thn sugkekrimenh erwthsh;

Ευρετηριο B+ tree gia thn people panw sta gnwrismata birthyear , personid
Eyrethrio gia thn plays panw sto gnwrisma personid