Family name: Sánchez Janés Given name: Sheila

Family name: Fortes Hidalgo Given name Alejandro

Question 1

For each question in the 3rd Lab SQL quiz:

1. Explain what structures you created
2. Explain Oracle’s execution plan considering the structures you created
3. For the execution plan given, discuss what steps would be executed differently by a column-oriented databases (refer to the physical data structures as well as query processing techniques used)
4. *Per l’exercici 1, hem utilitzat aquestes estructures:*

**(20%) SELECT pobl, MIN(edat), MAX(edat), COUNT(\*) FROM poll\_answers GROUP BY pobl;**

Aquesta es la última consulta que hem mirat, i amb els Bitmaps realitzats, ja millorava el cost, pero intentant millorant al màxim aquest cost i intentant no arribar als màxims de blocs possibles. Així, hem provat una **Vista** veient que només hi havien 120 files i ha resultat que millorava en gran medida el cost amb 5 i arribant a els blocs màxims que es poden utilitzar 275.

**(30%) SELECT pobl, edat, cand, MAX(val), MIN(val), AVG(val) FROM poll\_answers GROUP BY pobl, edat, cand;**

Aquesta consulta es la que hem provat a fer en segona instància i ha resultat la més complicada. Hem vist que n’hi havien més de 1000 files, per tant la Vista quedava descartada. La quantitat de vegades que es repeteixen els atributs per això, eran superiors a 100 i per tant, l’opció del Bitmap era una opció que podía ser la correcta.

Així, hem vist que realitzant 3 bitmaps, reduint així l’impacte de la creació de Hash Join a l’hora de fer els Bitmaps amb més d’un atribut. Per tant, l’opció que hem triat ha estat la realització de:

* **Bitmap (pobl)**
* **Bitmap (edat)**
* **Bitmap (cand, val)**

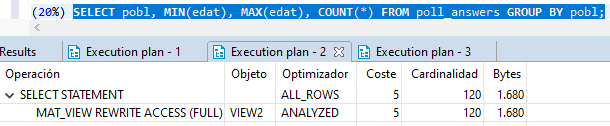
Creiem que la realització d’aquests Bitmaps, i veient que la quantitat de vegades que poden sortir els atributs de cand i val eren 2000 cadascun hem decidit posar-los junts i els altres separats. D’aquesta forma, el cost es de 42 i la quantitat de blocs aumenta en 24.

**(50%) SELECT cand, AVG(val) FROM poll\_answers GROUP BY cand;**

Aquesta consulta es la que hem provat fer primer ja que es la consulta amb el percentatge més alt i es podia veure en la seva execució, que només hi havien 10 files. Per tant, el primer que se'ns ha passat per la ment és l’utilització d’una Vista de la query. L’hem provat i veiem que el cost era 5 i l’augment dels blocs només de 208 a 216, així que la **Vista** ha estat l'opció correcta.

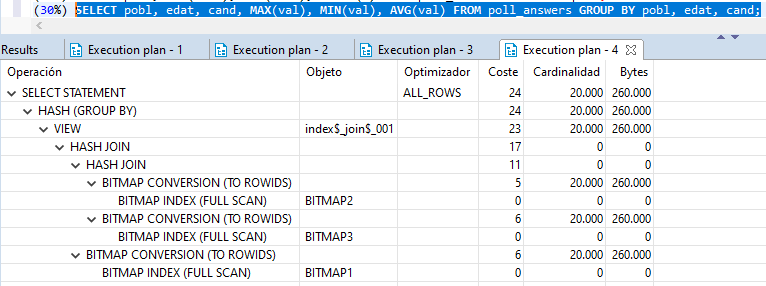
1. *El plan d’execució per cadascuna de les consultes ha estat aquesta:*

**(20%) SELECT pobl, MIN(edat), MAX(edat), COUNT(\*) FROM poll\_answers GROUP BY pobl;**



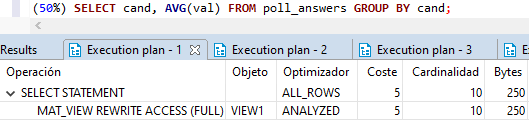
Simplement s’utilitza una Vista per la Consulta.

**(30%) SELECT pobl, edat, cand, MAX(val), MIN(val), AVG(val) FROM poll\_answers GROUP BY pobl, edat, cand;**



Aprofitant l’apartat d’execution plan que ens ofereix DBeaver, podem veure que s’utilitzen varies combinacions de Bitmaps en Hash Join. Es pot apreciar que el Bitmap2(edat) i Bitmap3(cand, val) estàn en una Hash Join i aquesta Hash Join amb el Bitmap1(pobl) i finalment aquesta última amb el Hash del Group By.

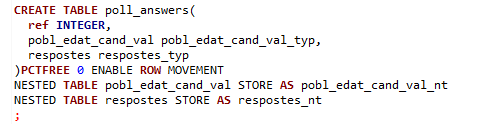
**(50%) SELECT cand, AVG(val) FROM poll\_answers GROUP BY cand;**



Aquí també s’utilitza una Vista per la consulta

1. *Per l’exercici 2, hem utilitzat aquestes estructures:*

Per aquests exercici, com a l’enunciat ens indicava que utilitzesim vertical fragmentation, per tant, hem decidit utilitzar nested table per simular-ho. Hem agrupat els atributs pobl, edat, cand i val i les respostes en dues nested table com podem veure a la imatge.



Al learn hem vist que hem arribat a un 95% de millora, per tant, podriem haver possat algun index.

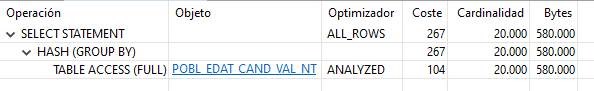
Hem volgut provar amb tres bitmaps en pobl, edat i cand,val i ho hem pujat al learn però ens hem adonat que no millora tant.

1. *El plan d’execució per cadascuna de les consultes ha estat aquesta:*

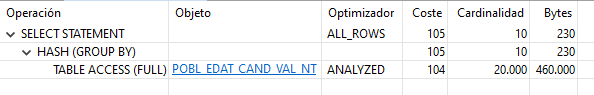
**(20%) SELECT pobl, MIN(edat), MAX(edat), COUNT(\*) FROM poll\_answers GROUP BY pobl;**

****

**(30%) SELECT pobl, edat, cand, MAX(val), MIN(val), AVG(val) FROM poll\_answers GROUP BY pobl, edat, cand;**

****

**(50%) SELECT cand, AVG(val) FROM poll\_answers GROUP BY cand;**

****

Primer voliem recalcar que les querys que hem posat es modifiquen una mica perquè la taula poll\_answers ha estat modificada per poder fer les nested tables. Per tant, com podem veure a les execution plans s’accedeix a les nested table.

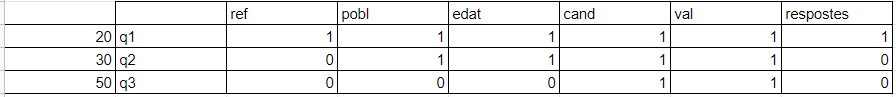
1. *Diferencia entre execution plan amb i sense column-oriented*

Veurem que es fan hash join abans d’accedir a les taules o als bitmaps. Això és deu a què oracle és row-oriented però si fos column-oriented aquesta transacció no faria falta.

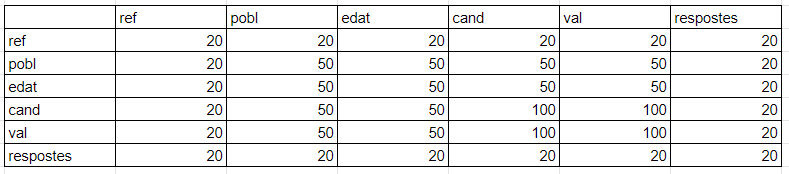
Question 2

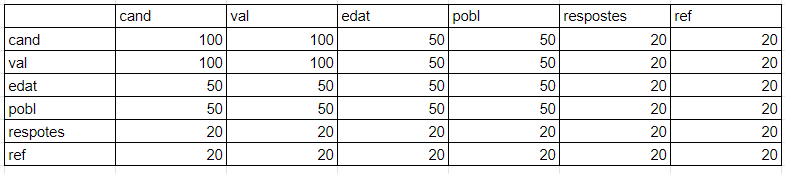
For the second exercise, uevvse the affinity matrix method to decide how to fragment the database vertically. Consider now your solution for Exercise 2 in the 3rd Lab SQL quiz. Is Oracle yielding the best result when using the same vertical fragmentation strategy as suggested by the affinity matrix method? **Justify your answer.**

Aquesta primera taula, anomenada attribute usage matrix, podem veure si els atributs han aparegut a les queries donades.



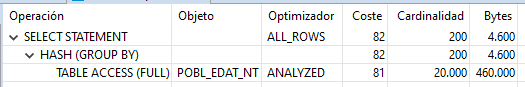
A continuació, podem veure l’attribute affinity matrix i, aquesta ordenada. Aquí podem veure l’afinitat de cada atribut tenint en compte la freqüencia de les queriess i si han aparegut.



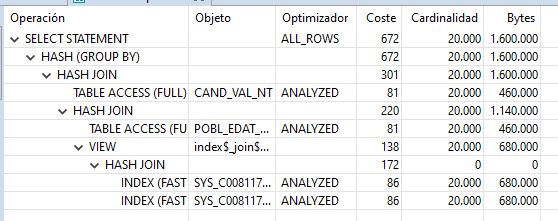


Així que hem fet 3 nested table, una amb pobl i edat, altra amb cand i val i altre amb les respostes. Com podem veure a les imatges el cost de la query 1 i 2 ha disminuït però el de la query 3 dona 672, moltíssim més que amb la fragmentation que hem fet nosaltres. Així que podem veure que no és el millor resultat amb l’affinity matrix method.

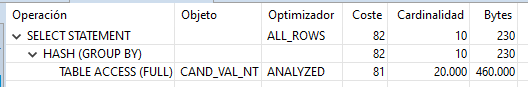
**Query 1**



**Query 2**

****

**Query 3**

****