U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Mestrado em Engenharia Informática e Computação

Tecnologias de Bases de Dados

Otimização de Interrogações

PROJ1

Inês Ferreira de Almeida
João António Teixeira Coelho

March 2025

Índice

1 Introdução

A otimização de interrogações SQL é uma componente essencial da administração e desenvolvimento de bases de dados, sobretudo quando se lida com grandes volumes de informação. Este projeto tem como objetivo analisar o impacto de diferentes estratégias de estruturação de consultas SQL numa base de dados real, relacionada com as eleições legislativas portuguesas de 1999. Foram definidos 3 ambientes diferentes: Ambiente X, em que não existem índices nem restrições de integridade; Ambiente Y, em que existem apenas as restrições de integridade standard (chave primária e chave externa); e o ambiente Z, com as restrições de integridade standard e índices extra desenvolvidos para otimizar o desempenho, que serão expostos e explicados nas próximas secções.

2 Dados

A base de dados fornecida segue o esquema presente na figura 1. Para a criação dos ambientes, foram copiadas as tabelas presentes no utilizador GTD7. Serão agora apresentados os índices desenvolvidos para o ambiente Z, assim como a motivação para eles.

1. **Índices para a pergunta 1**

```
CREATE INDEX IDX_ZCONCELHOS_COD_NOME ON ZCONCELHOS (CODIGO, NOME);
CREATE INDEX IDX_ZFREGUESIAS_CONCELHO ON ZFREGUESIAS (CONCELHO);
CREATE INDEX IDX_ZLISTAS_DISTRITO_PARTIDO ON ZLISTAS (DISTRITO,
    PARTIDO);
CREATE INDEX IDX_ZVOTACOES_FREGUESIA_PARTIDO ON ZVOTACOES (FREGUESIA,
    PARTIDO);
```

Motivação

A primeira pergunta e as suas alíneas lidam com operação de seleção e junção. Por estes motivos, os índices acima permitem acelerar a pesquisa pois:

(a) **IDX_ZCONCELHOS_COD_NOME ON ZCONCELHOS(CODIGO, NOME)**:
Este índice permite otimizar tanto junções como filtragens relacionadas com a tabela `ZCONCELHOS`. Como o índice está ordenado por `CODIGO` e depois por `NOME`, ele é particularmente útil em junções que envolvem o código do concelho — como acontece na maioria das relações com freguesias — e também permite acelerar buscas pelo nome do concelho. Fisicamente, o índice permite localizar de forma eficiente os blocos onde constam os concelhos com o nome desejado, reduzindo significativamente o número de leituras necessárias, tanto para a filtragem como para a resolução da junção.

(b) **IDX_ZFREGUESIAS_CONCELHO ON ZFREGUESIAS(CONCELHO)**:
Este índice cria uma árvore ordenada por CONCELHO, o que permite localizar rapidamente todas as freguesias associadas a um concelho específico. Fisicamente, isso significa que a base de dados não precisa fazer um scan completo à tabela ZFREGUESIAS — ela percorre a árvore até ao valor do código e segue os apontadores diretamente para as linhas relevantes. Isto reduz drasticamente o número de leituras no disco.

(c) **IDX_ZLISTAS_DISTRITO_PARTIDO ON ZLISTAS(DISTRITO, PARTIDO)**:
Este índice é eficaz quando uma query filtra por DISTRITO e utiliza PARTIDO como chave de junção. Fisicamente, o índice permite encontrar rapidamente todas as linhas de um determinado distrito, e dentro desse subconjunto, aceder de forma eficiente às linhas de um partido específico. Isto reduz o número de blocos de dados que precisam de ser lidos e melhora a performance da operação de junção.

(d) **IDX_ZVOTACOES_FREGUESIA_PARTIDO ON ZVOTACOES(FREGUESIA, PARTIDO)**:
Este índice suporta bem uma query que filtra por uma lista de freguesias e por partido. Fisicamente, permite à base de dados saltar diretamente para os ramos relevantes da

árvore B onde estão armazenadas as freguesias indicadas, e dentro delas, aceder rapidamente aos registos do partido. Como o índice já está ordenado por FREGUESIA, ele também pode acelerar a agregação (GROUP BY) sem necessidade de ordenar os dados manualmente, o que poupa recursos e tempo.

2. **Índices para a pergunta 2**

```
CREATE INDEX IDX_ZVOTACOES_PARTIDO ON ZVOTACOES (PARTIDO);
CREATE INDEX IDX_ZVOTACOES_FREGUESIA ON ZVOTACOES (FREGUESIA);
CREATE INDEX IDX_ZVOTACOES_FREGUESIA_PARTIDO_VOTOS ON ZVOTACOES (
    FREGUESIA, VOTOS DESC, PARTIDO);
CREATE INDEX IDX_ZCONCELHOS_DISTRITO ON ZCONCELHOS (DISTRITO, CODIGO);
CREATE INDEX IDX_ZVOTACOES_PARTIDO_VOTOS ON ZVOTACOES (PARTIDO, VOTOS
    DESC);
```

Motivação

Os seguintes índices foram idealizados para operações de agregação:

(a) **IDX_ZVOTACOES_PARTIDO ON ZVOTACOES(PARTIDO)**:
Este índice é essencial para otimizar interrogações que filtram os votos de um partido específico, como o caso de consultas que calculam o total de votos de partidos individuais. Quando a query envolve um filtro baseado no partido, a base de dados pode aceder diretamente aos registos do partido sem a necessidade de fazer um scan completo da tabela de votos. Isso resulta em uma redução significativa no tempo de resposta, especialmente em tabelas grandes.

(b) **IDX_ZVOTACOES_FREGUESIA ON ZVOTACOES(FREGUESIA)**:
Este índice é útil em queries que envolvem junções entre a tabela de votos e a tabela de freguesias. Ao associar votos com as freguesias através do código de freguesia, o índice permite uma navegação rápida pelas entradas relevantes de cada freguesia. Em operações de agregação por freguesia ou quando se procura dados específicos de freguesias, o índice elimina a necessidade de fazer buscas completas, melhorando o desempenho das operações de agregação e contagem de votos.

(c) **IDX_ZVOTACOES_FREGUESIA_PARTIDO_VOTOS ON ZVOTACOES(FREGUESIA, VOTOS DESC, PARTIDO)**:
Este índice é criado para otimizar interrogações que precisam de identificar o partido com o maior número de votos em cada freguesia. Ao ordenar os votos dentro de cada freguesia de forma descendente, a base de dados pode rapidamente localizar o partido com o maior número de votos, sem precisar ordenar toda a tabela ou realizar cálculos extra.

(d) **IDX_ZCONCELHOS_DISTRITO ON ZCONCELHOS(DISTRITO, CODIGO)**:
Este índice melhora a performance das consultas que fazem junções entre freguesias, concelhos e distritos. Quando é necessário agregar ou filtrar votos por distrito, o índice ajuda a localizar rapidamente os concelhos dentro de um distrito específico. Isso acelera a execução de interrogações que envolvem agrupamento de votos por distrito, especialmente em grandes volumes de dados.

(e) **IDX_ZVOTACOES_PARTIDO_VOTOS ON ZVOTACOES(PARTIDO, VOTOS DESC)**:
Este índice foi criado para acelerar a identificação do partido mais votado em diferentes níveis de agregação. Ao ordenar os votos de forma descendente dentro de cada partido, ele melhora a performance de queries que precisam de agrupar votos por partido e identificar rapidamente os partidos mais votados. A ordenação por votos permite que a base de dados aceda diretamente aos registos com mais votos, acelerando a comparação entre partidos e o cálculo dos totais por distrito ou freguesia.

3. **Índices para a pergunta 3**

```
CREATE BITMAP INDEX BM_ZLISTAS_DISTRITO_PARTIDO ON ZLISTAS (DISTRITO,
    PARTIDO);
```

Motivação

Este índice bitmap foi criado para acelerar a query que procura partidos sem listas num distrito específico, como o distrito de Lisboa. O índice bitmap é particularmente eficiente em cenários em que as colunas 'DISTRITO' e 'PARTIDO' possuem um número limitado de valores distintos (o que é o caso, já que os distritos e partidos geralmente não são muitos - cardinalidade baixa). Ele permite que a base de dados rapidamente encontre que partidos estão ou não associados a um distrito específico, sem a necessidade de realizar scans completos da tabela.

4. **Índices para a pergunta 4**

```
CREATE INDEX IDX_ZVOTACOES_FREGUESIA_VOTOS ON ZVOTACOES (FREGUESIA,
    VOTOS DESC);
```

Motivação

Este índice foi criado para otimizar interrogações que visam identificar o partido vencedor em cada freguesia, especialmente aquelas que precisam de ordenar os votos de forma decrescente (como nas consultas que procuram o partido com o maior número de votos em cada freguesia). A base de dados pode aceder diretamente às linhas com os maiores valores de votos para cada freguesia, evitando a necessidade de fazer um scan completo da tabela para identificar os partidos vencedores.

5. **Índices para a pergunta 5**

```
CREATE INDEX IDX_ZCONCELHOS_DISTRITO ON ZCONCELHOS (DISTRITO);
CREATE INDEX IDX_ZVOTACOES_PARTIDO ON ZVOTACOES (PARTIDO);
CREATE BITMAP INDEX BM_ZVOTACOES_PARTIDO ON ZVOTACOES (PARTIDO);
```

Motivação

A utilização de 2 tipos de índices diferentes, como pedido no enunciado, permite avaliar as valências e fraquezas de cada tipo de índice

(a) **IDX_ZCONCELHOS_DISTRITO ON ZCONCELHOS(DISTRITO)**:
Este índice melhora as interrogações que filtrem por distrito na tabela concelhos, acelerando bastante a execução de queries que realizem junções ou filtros que utilizem o código de distrito.

(b) **IDX_ZVOTACOES_PARTIDO ON ZVOTACOES(PARTIDO)**:
Este índice melhora as interrogações que filtrem por partido na tabela votações, acelerando bastante a execução de queries que realizem junções ou filtros que utilizem o código de distrito.

(c) **BM_ZVOTACOES_PARTIDO ON ZVOTACOES(PARTIDO)**:
Este índice é eficiente para operações de consulta em colunas com poucos valores distintos (como 'PS', 'PSD', etc.). Pode ser muito útil para realizar operações de OR (em vez de uma pesquisa por cada valor de partido)

3 Método

Foram formuladas e executadas várias interrogações SQL, representando operações comuns como seleção, junção, agregação e negação. As consultas foram testadas nos três ambientes de experimentação. Para cada interrogação, foram analisados os seguintes aspetos:

- **Formulação em SQL** – Código da interrogação executada.

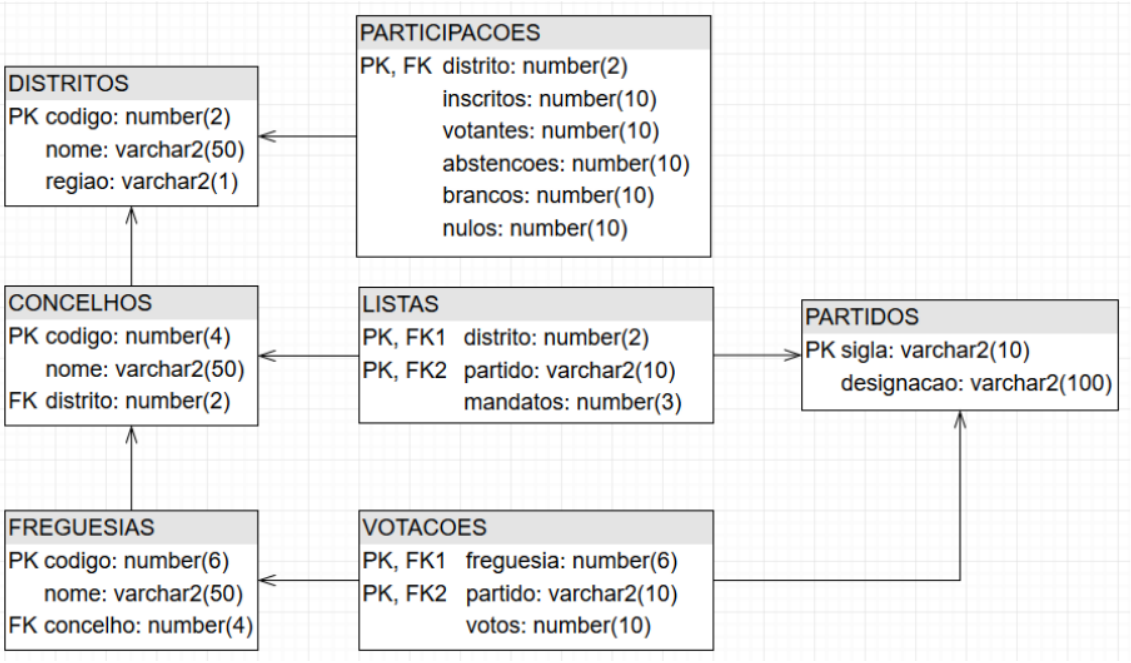


Figure 1: Esquema relacional da base de dados

- **Resultados** – Dados retornados pela interrogação.
- **Planos de execução** – Estrutura de execução gerada pelo otimizador do SGBD.
- **Discussão** – Análise do impacto das otimizações aplicadas.

Na formulação em SQL, até à questão 3, as interrogações são apresentadas utilizando o ambiente X como exemplo. No entanto, para executar a mesma consulta nos outros ambientes, basta substituir o prefixo "**x**" por "**y**" ou "**z**", conforme o contexto desejado.

4 Questões

Nesta secção, são apresentadas as questões do enunciado que guiam a análise e otimização das interrogações SQL. As perguntas abordam diferentes operações sobre a base de dados, incluindo seleção, junção, agregação e negação, permitindo avaliar o desempenho das consultas em diferentes configurações.

4.1 Seleção e junção

A seleção permite obter subconjuntos específicos de dados com base em condições definidas, enquanto a junção combina informação de múltiplas tabelas relacionadas, facilitando a extração de dados interligados.

Estas são as questões propostas:

a. Quais os códigos e nomes de freguesias do concelho 1103? E do concelho "Azambuja"?

b. Indique as siglas e designações dos partidos e o respetivo número de mandatos obtidos no distrito de Lisboa.

c. Indique o número de votos obtido pelo BE nas freguesias do distrito de Lisboa.

1. **Formulação em SQL**

```sql
-- a
SELECT codigo, nome
FROM xfreguesias
WHERE concelho = 1103;

SELECT f.codigo, f.nome
FROM xfreguesias f
JOIN xconcelhos c ON f.concelho = c.codigo
WHERE c.nome = 'Azambuja';

-- b
SELECT l.partido, p.designacao, l.mandatos
FROM xlistas l
JOIN xpartidos p ON l.partido = p.sigla
WHERE l.distrito = (SELECT codigo FROM xdistritos WHERE nome = 'Lisboa
    ');

-- c
SELECT v.freguesia, SUM(v.votos) AS total_votos
FROM xvotacoes v
JOIN xfreguesias f ON v.freguesia = f.codigo
WHERE f.concelho IN (SELECT codigo FROM xconcelhos WHERE distrito =
                     (SELECT codigo FROM xdistritos WHERE nome = '
                        Lisboa'))
AND v.partido = 'BE'
GROUP BY v.freguesia;
```

2. **Resultados**

CODIGO	NOME
110301	Alcoentre
110302	Aveiras de Baixo
110303	Aveiras de Cima
110304	Azambuja
110305	Manique do Intendente
110306	Vale do Paraiso
110307	Vila Nova da Rainha
110308	Vila Nova de São Pedro
110309	Maçussa

9 rows selected.

CODIGO	NOME
110301	Alcoentre
110302	Aveiras de Baixo
110303	Aveiras de Cima
110304	Azambuja
110305	Manique do Intendente
110306	Vale do Paraiso
110307	Vila Nova da Rainha
110308	Vila Nova de São Pedro
110309	Maçussa

9 rows selected.

Figure 2: Resultados da alínea a.

PARTIDO	DESIGNACAO	MANDATOS
PS	Partido Socialista	23
PSN	Partido Solidariedade Nacional	0
PPM	Partido Popular Monárquico	0
PPDPSD	Partido Social Democrata	14
POUS	Partido Operário de Unidade Socialista	0
PH	Partido Humanista	0
PCTPMRPP	Partido Comunista dos Trabalhadores Portugueses	0
PCPPEV	Partido Comunista Português	6
CDSPP	Partido Popular	4
BE	Bloco de Esquerda	2
MPT	Movimento Partido da Terra	0

11 rows selected.

Figure 3: Resultados da alínea b.

FREGUESIA	TOTAL_VOTOS
110716	43
110804	5
110903	3
110904	10
110910	31
110916	54
111104	229
111114	42
111107	815
111312	20
111411	173

FREGUESIA	TOTAL_VOTOS
111319	0
111002	276

222 rows selected.

Figure 4: Resultados da alínea c.

3. **Planos de execução**

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			14	7
TABLE ACCESS	XFREGUESIAS	FULL	14	7
Filter Predicates				
CONCELHO=1103				

(a) Primeira parte da alínea a no contexto de x

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			9	7
TABLE ACCESS	YFREGUESIAS	FULL	9	7
Filter Predicates				
CONCELHO=1103				

(b) Primeira parte da alínea a no contexto de y

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			9	2
TABLE ACCESS	ZFREGUESIAS	BY INDEX ROWID BATCHED	9	2
INDEX	IDX_ZFREGUESIAS_CONCELHO	RANGE SCAN	9	1
Access Predicates				
CONCELHO=1103				

(c) Primeira parte da alínea a no contexto de z

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			14	10
HASH JOIN			14	10
Access Predicates				
F.CONCELHO=C.CODIGO				
TABLE ACCESS	XCONCELHOS	FULL	1	3
Filter Predicates				
C.NOME='Azambuja'				
TABLE ACCESS	XFREGUESIAS	FULL	4241	7

(d) Segunda parte da alínea a no contexto de x

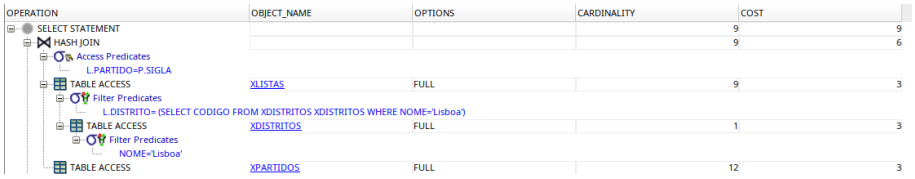
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			14	10
HASH JOIN			14	10
Access Predicates				
F.CONCELHO=C.CODIGO				
TABLE ACCESS	YCONCELHOS	FULL	1	3
Filter Predicates				
C.NOME='Azambuja'				
TABLE ACCESS	YFREGUESIAS	FULL	4241	7

(e) Segunda parte da alínea a no contexto de y

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			14	4
HASH JOIN			14	4
Access Predicates				
F.CONCELHO=C.CODIGO				
NESTED LOOPS			14	4
NESTED LOOPS			14	4
STATISTICS COLLECTOR				
INDEX	IDX_ZCONCELHOS_COD_NOME	FAST FULL SCAN	1	2
Filter Predicates				
C.NOME='Azambuja'				
INDEX	IDX_ZFREGUESIAS_CONCELHO	RANGE SCAN	14	1
Access Predicates				
F.CONCELHO=C.CODIGO				
TABLE ACCESS	ZFREGUESIAS	BY INDEX ROWID	14	2
TABLE ACCESS	ZFREGUESIAS	FULL	14	2

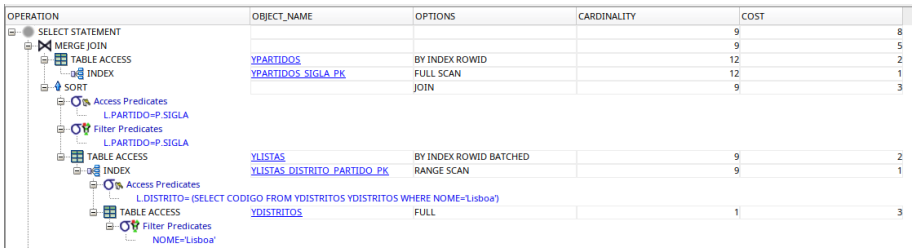
(f) Segunda parte da alínea a no contexto de z

Figure 5: Planos de execução para a alínea a.



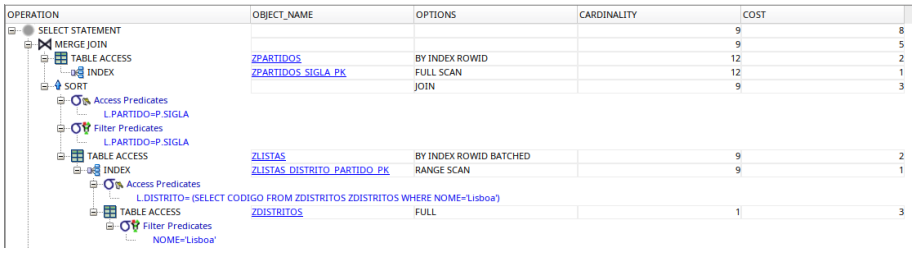
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			9	9
HASH JOIN			9	6
Access Predicates				
L.PARTIDO=P.SIGLA				
TABLE ACCESS	XLISTAS	FULL	9	3
Filter Predicates				
L.DISTRITO= (SELECT CODIGO FROM XDISTRITOS XDISTRITOS WHERE NOME='Lisboa')				
TABLE ACCESS	XDISTRITOS	FULL	1	3
Filter Predicates				
NOME='Lisboa'				
TABLE ACCESS	XPARTIDOS	FULL	12	3

(a) Alínea b no contexto de x



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			9	8
MERGE JOIN			9	5
TABLE ACCESS	YPARTIDOS	BY INDEX ROWID	12	2
INDEX	YPARTIDOS_SIGLA_PK	FULL SCAN	12	1
SORT		JOIN	9	3
Access Predicates				
L.PARTIDO=P.SIGLA				
Filter Predicates				
L.PARTIDO=P.SIGLA				
TABLE ACCESS	YLISTAS	BY INDEX ROWID BATCHED	9	2
INDEX	YLISTAS_DISTRITO_PARTIDO_PK	RANGE SCAN	9	1
Access Predicates				
L.DISTRITO= (SELECT CODIGO FROM YDISTRITOS YDISTRITOS WHERE NOME='Lisboa')				
TABLE ACCESS	YDISTRITOS	FULL	1	3
Filter Predicates				
NOME='Lisboa'				

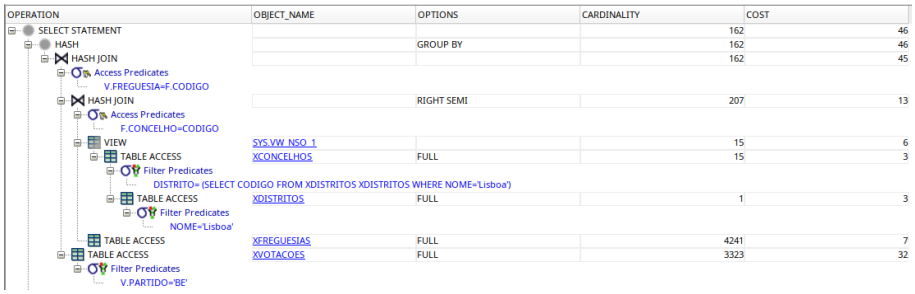
(b) Alínea b no contexto de y



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			9	8
MERGE JOIN			9	5
TABLE ACCESS	ZPARTIDOS	BY INDEX ROWID	12	2
INDEX	ZPARTIDOS_SIGLA_PK	FULL SCAN	12	1
SORT		JOIN	9	3
Access Predicates				
L.PARTIDO=P.SIGLA				
Filter Predicates				
L.PARTIDO=P.SIGLA				
TABLE ACCESS	ZLISTAS	BY INDEX ROWID BATCHED	9	2
INDEX	ZLISTAS_DISTRITO_PARTIDO_PK	RANGE SCAN	9	1
Access Predicates				
L.DISTRITO= (SELECT CODIGO FROM ZDISTRITOS ZDISTRITOS WHERE NOME='Lisboa')				
TABLE ACCESS	ZDISTRITOS	FULL	1	3
Filter Predicates				
NOME='Lisboa'				

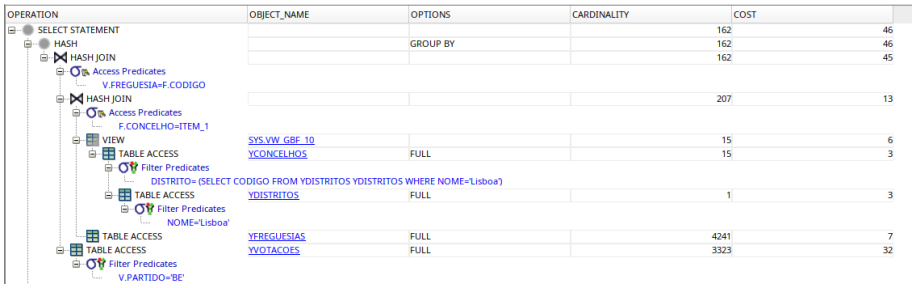
(c) Alínea b no contexto de z

Figure 6: Planos de execução para a alínea b.



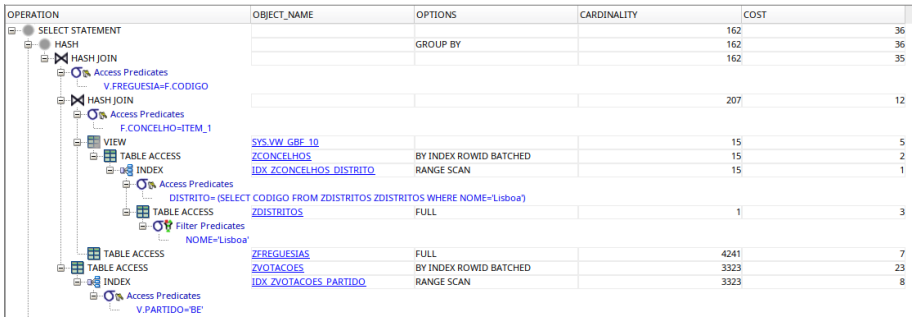
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			162	46
HASH		GROUP BY	162	46
HASH JOIN			162	45
Access Predicates				
V.FREGUESIA=F.CODIGO				
HASH JOIN		RIGHT SEMI	207	13
Access Predicates				
F.CONCELHO=CODIGO				
VIEW	SYS.VW_NSO_1		15	6
TABLE ACCESS	XCONCELHOS	FULL	15	3
Filter Predicates				
DISTRITO= (SELECT CODIGO FROM XDISTRITOS XDISTRITOS WHERE NOME='Lisboa')				
TABLE ACCESS	XDISTRITOS	FULL	1	3
Filter Predicates				
NOME='Lisboa'				
TABLE ACCESS	XFREGUESIAS	FULL	4241	7
TABLE ACCESS	XVOTACOES	FULL	3323	32
Filter Predicates				
V.PARTIDO='BE'				

(a) Alínea c no contexto de x



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			162	46
HASH		GROUP BY	162	46
HASH JOIN			162	45
Access Predicates				
V.FREGUESIA=F.CODIGO				
HASH JOIN			207	13
Access Predicates				
F.CONCELHO=ITEM_1				
VIEW	SYS.VW_GBF_10		15	6
TABLE ACCESS	YCONCELHOS	FULL	15	3
Filter Predicates				
DISTRITO= (SELECT CODIGO FROM YDISTRITOS YDISTRITOS WHERE NOME='Lisboa')				
TABLE ACCESS	YDISTRITOS	FULL	1	3
Filter Predicates				
NOME='Lisboa'				
TABLE ACCESS	YFREGUESIAS	FULL	4241	7
TABLE ACCESS	YVOTACOES	FULL	3323	32
Filter Predicates				
V.PARTIDO='BE'				

(b) Alínea c no contexto de y



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			162	36
HASH		GROUP BY	162	36
HASH JOIN			162	35
Access Predicates				
V.FREGUESIA=F.CODIGO				
HASH JOIN			207	12
Access Predicates				
F.CONCELHO=ITEM_1				
VIEW	SYS.VW_GBF_10		15	5
TABLE ACCESS	ZCONCELHOS	BY INDEX ROWID BATCHED	15	2
INDEX	IDX ZCONCELHOS_DISTRITO	RANGE SCAN	15	1
Access Predicates				
DISTRITO= (SELECT CODIGO FROM ZDISTRITOS ZDISTRITOS WHERE NOME='Lisboa')				
TABLE ACCESS	ZDISTRITOS	FULL	1	3
Filter Predicates				
NOME='Lisboa'				
TABLE ACCESS	ZFREGUESIAS	FULL	4241	7
TABLE ACCESS	ZVOTACOES	BY INDEX ROWID BATCHED	3323	23
INDEX	IDX ZVOTACOES_PARTIDO	RANGE SCAN	3323	8
Access Predicates				
V.PARTIDO='BE'				

(c) Alínea c no contexto de z

Figure 7: Planos de execução para a alínea c.

4. **Discussão**

   (a) **1a)**

   Esta pergunta pede para selecionar os códigos e nomes de freguesias do concelho 1103 e da Azambuja.

   Ao olhar para o plano de execução dos ambientes X e Y e comparando com o ambiente Z, percebemos que os 2 primeiros realizam scans à tabela inteira (custos 7 e 10 para cada parte da pergunta), enquanto que o terceiro utliza índices para reduzir o custo da operação (custos 2 e 4). É de realçar o menor custo e cardinalidade da operação em que o filtro é o código, para o ambiente Z, indicando que é mais eficiente filtrar através deste método do que pelo nome do concelho, para os índices implementados.

   Ao procurar pelo nome "Azambuja", embora o filtro seja aplicado apenas sobre o `NOME`, o otimizador escolhe o índice *IDX_ZCONCELHOS_COD_NOME* porque o `CODIGO` (a primeira coluna do índice) participa na junção com a tabela `ZFREGUESIAS`.

   (b) **1b)**

   Esta pergunta pede para selecionar siglas e designações dos partidos e o respetivo número de mandatos obtidos no distrito de Lisboa.

   É de realçar que a introdução de restrições standard (chave primária e externa) foi o maior diferenciador de desempenho, já que o otimizador não utilizou nenhum dos índices que foram desenhados para o ambiente Z, apenas fazendo um *range scan* com base na chave primária composta da tabela LISTAS, nos ambientes Y e Z.

   (c) **1c)**

   Esta pergunta pede para indicar o número de votos obtido pelo BE nas freguesias do distrito de Lisboa.

   Neste caso, os índices *IDX_ZCONCELHOS_DISTRITO* e *IDX_ZVOTACOES_PARTIDO* permititam reduzir os custos das operações em que foram utilizados de 3 para 2 e 32 para 23, respetivamente, levando a uma redução de custo de 10 no ambiente Z.

   Estes índices permitiram acelerar o acesso aos concelhos com o codigo de Lisboa e o acesso às votações do BE, facilitando as operações de junção e filtro.

4.2 Agregação

A agregação em SQL permite resumir dados através de funções como COUNT(), SUM(), AVG(), MIN() e MAX(). Estas operações são frequentemente utilizadas com GROUP BY para calcular métricas específicas por grupo, facilitando a análise e interpretação dos dados.

Estas são as questões propostas:

a. Quantos votos teve o 'PS' a nível nacional?

b. Quantos votos teve cada partido, em cada distrito?

c. Qual o partido que, ao nível de freguesia, registou o maior número de votos? Indique a sigla do partido, o nome da freguesia e os votos correspondentes.

d. Para cada distrito indique qual o seu nome e a designação e número de votos do partido que nele teve melhor votação.

1. **Formulação em SQL**

```sql
-- a
SELECT SUM(v.votos) AS total_votos
FROM xvotacoes v
WHERE v.partido = 'PS';
```

```sql
-- b
SELECT c.distrito, v.partido, SUM(v.votos) AS total_votos
FROM xvotacoes v
JOIN xfreguesias f ON v.freguesia = f.codigo
JOIN xconcelhos c ON f.concelho = c.codigo
GROUP BY c.distrito, v.partido
ORDER BY c.distrito, total_votos DESC;

-- c
SELECT v.freguesia, v.partido, f.nome, v.votos
FROM xvotacoes v
JOIN xfreguesias f ON v.freguesia = f.codigo
WHERE (v.freguesia, v.votos) IN (
    SELECT v1.freguesia, MAX(v1.votos)
    FROM xvotacoes v1
    GROUP BY v1.freguesia
)
ORDER BY v.votos DESC;

-- d
SELECT d.codigo AS distrito, d.nome, v.partido, SUM(v.votos) AS
    total_votos
FROM xvotacoes v
JOIN xfreguesias f ON v.freguesia = f.codigo
JOIN xconcelhos c ON f.concelho = c.codigo
JOIN xdistritos d ON c.distrito = d.codigo
GROUP BY d.codigo, d.nome, v.partido
HAVING SUM(v.votos) = (
    SELECT MAX(total_votos)
    FROM (
        SELECT SUM(v2.votos) AS total_votos
        FROM xvotacoes v2
        JOIN xfreguesias f2 ON v2.freguesia = f2.codigo
        JOIN xconcelhos c2 ON f2.concelho = c2.codigo
        WHERE c2.distrito = d.codigo
        GROUP BY v2.partido
    ) subquery
)
ORDER BY d.codigo;
```

2. **Resultados**

```
TOTAL_VOTOS
-----------
    2359939
```

Figure 8: Resultados da alínea a.

```
        30 PCTPMRPP           657
        30 PSN                570
        30 MPT                475
        40 PS               49947
        40 PPDPSD           33564

  DISTRITO PARTIDO    TOTAL_VOTOS
---------- ---------- -----------
        40 CDSPP             5215
        40 PCPPEV            1612
        40 BE                 992
        40 PDA                437
        40 PCTPMRPP           330
        40 MPT                178

182 rows selected.
```

Figure 9: Resultados da alínea b.

```
    130125 PCTPMRPP   Olo                                                0
    130125 PCPPEV     Olo                                                0
    130125 MPT        Olo                                                0
    130125 CDSPP      Olo                                                0
    130125 BE         Olo                                                0
    130713 MPT        Paços de Gaiolo                                    0
    130713 PSN        Paços de Gaiolo                                    0
    130713 PS         Paços de Gaiolo                                    0
    130713 PH         Paços de Gaiolo                                    0

 FREGUESIA PARTIDO    NOME                                           VOTOS
---------- ---------- -------------------------------------------------- ----------
    130713 POUS       Paços de Gaiolo                                    0
    130713 PCTPMRPP   Paços de Gaiolo                                    0

4.358 rows selected.
```

Figure 10: Resultados da alínea c.

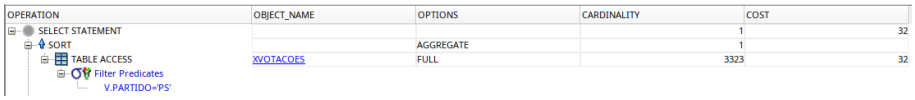
DISTRITO	NOME	PARTIDO	TOTAL_VOTOS
1	Aveiro	PS	145575
2	Beja	PS	39728
3	Braga	PS	195602
4	Bragança	PPDPSD	36841
5	Castelo Branco	PS	63398
6	Coimbra	PS	109956
7	Évora	PS	42257
8	Faro	PS	87162
9	Guarda	PS	44254
10	Leiria	PPDPSD	99091
11	Lisboa	PS	480410

DISTRITO	NOME	PARTIDO	TOTAL_VOTOS
12	Portalegre	PS	36545
13	Porto	PS	440162
14	Santarém	PS	110326
15	Setúbal	PS	170193
16	Viana do Castelo	PS	55132
17	Vila Real	PPDPSD	56507
18	Viseu	PPDPSD	90116
30	Madeira	PPDPSD	56302
40	Açores	PS	49947

20 rows selected.

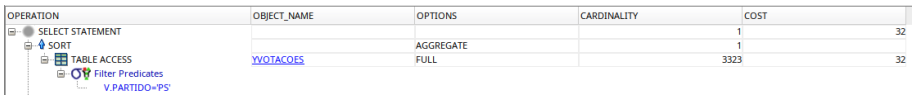
Figure 11: Resultados da alínea d.

3. **Planos de execução**



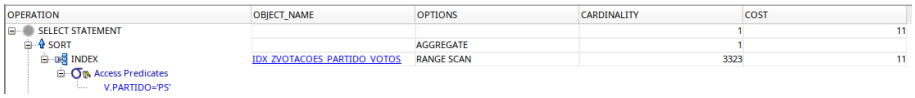
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	32
SORT		AGGREGATE	1	
TABLE ACCESS	XVOTACOES	FULL	3323	32
Filter Predicates				
V.PARTIDO='PS'				

(a) Alínea a no contexto de x



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	32
SORT		AGGREGATE	1	
TABLE ACCESS	YVOTACOES	FULL	3323	32
Filter Predicates				
V.PARTIDO='PS'				

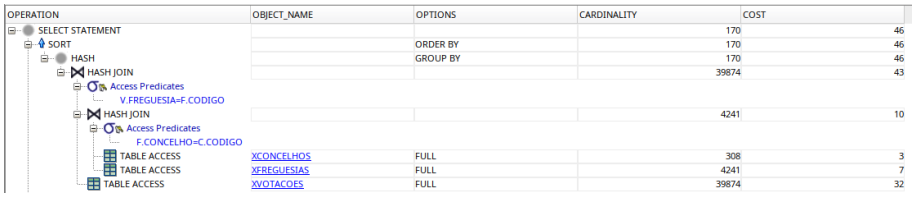
(b) Alínea a no contexto de y



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	11
SORT		AGGREGATE	1	
INDEX	IDX_ZVOTACOES_PARTIDO_VOTOS	RANGE SCAN	3323	11
Access Predicates				
V.PARTIDO='PS'				

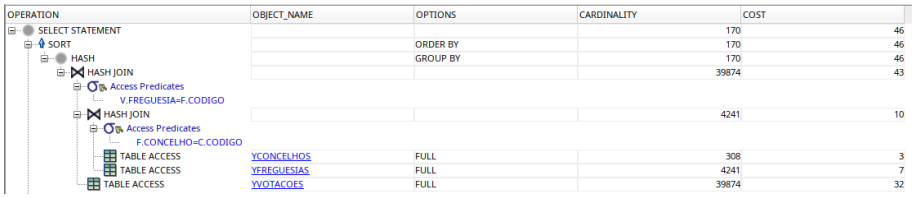
(c) Alínea a no contexto de z

Figure 12: Planos de execução para a alínea a.



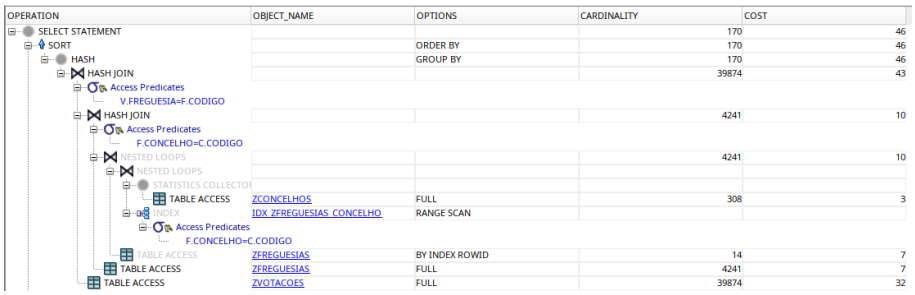
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			170	46
SORT		ORDER BY	170	46
HASH		GROUP BY	170	46
HASH JOIN			39874	43
Access Predicates				
V.FREGUESIA=F.CODIGO				
HASH JOIN			4241	10
Access Predicates				
F.CONCELHO=C.CODIGO				
TABLE ACCESS	XCONCELHOS	FULL	308	3
TABLE ACCESS	XFREGUESIAS	FULL	4241	7
TABLE ACCESS	XVOTACOES	FULL	39874	32

(a) Alínea b no contexto de x



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			170	46
SORT		ORDER BY	170	46
HASH		GROUP BY	170	46
HASH JOIN			39874	43
Access Predicates				
V.FREGUESIA=F.CODIGO				
HASH JOIN			4241	10
Access Predicates				
F.CONCELHO=C.CODIGO				
TABLE ACCESS	YCONCELHOS	FULL	308	3
TABLE ACCESS	YFREGUESIAS	FULL	4241	7
TABLE ACCESS	YVOTACOES	FULL	39874	32

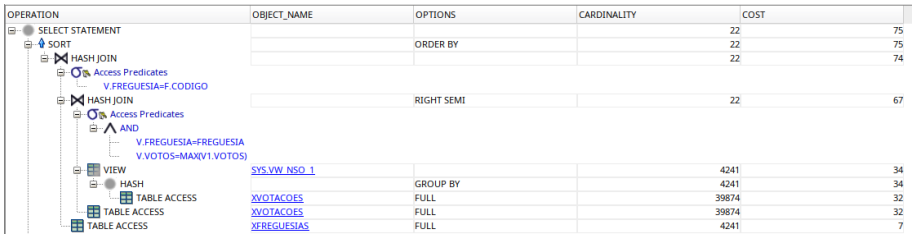
(b) Alínea b no contexto de y



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			170	46
SORT		ORDER BY	170	46
HASH		GROUP BY	170	46
HASH JOIN			39874	43
Access Predicates				
V.FREGUESIA=F.CODIGO				
HASH JOIN			4241	10
Access Predicates				
F.CONCELHO=C.CODIGO				
NESTED LOOPS			4241	10
NESTED LOOPS				
STATISTICS COLLECTOR				
TABLE ACCESS	ZCONCELHOS	FULL	308	3
INDEX	IDX_ZFREGUESIAS_CONCELHO	RANGE SCAN		
Access Predicates				
F.CONCELHO=C.CODIGO				
TABLE ACCESS	ZFREGUESIAS	BY INDEX ROWID	14	7
TABLE ACCESS	ZFREGUESIAS	FULL	4241	7
TABLE ACCESS	ZVOTACOES	FULL	39874	32

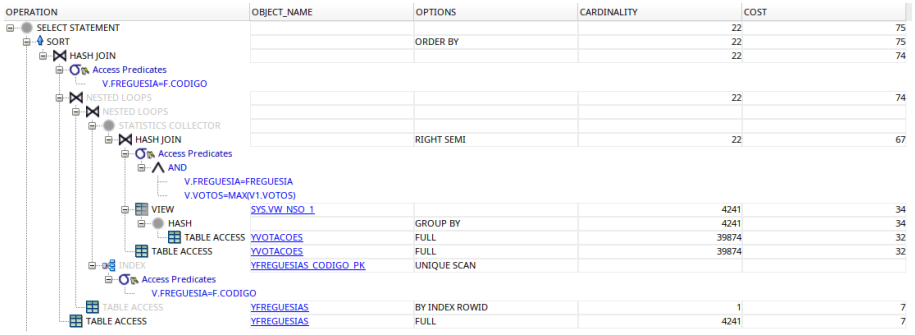
(c) Alínea b no contexto de z

Figure 13: Planos de execução para a alínea b.



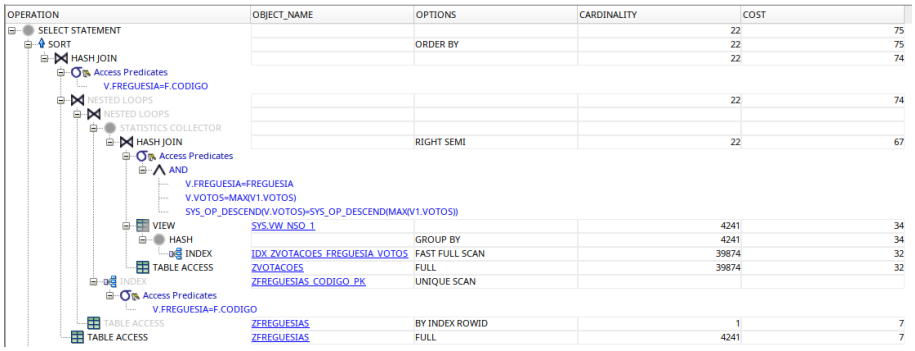
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			22	75
SORT		ORDER BY	22	75
HASH JOIN			22	74
Access Predicates				
V.FREGUESIA=F.CODIGO				
HASH JOIN		RIGHT SEMI	22	67
Access Predicates				
AND				
V.FREGUESIA=FREGUESIA				
V.VOTOS=MAX(V1.VOTOS)				
VIEW	SYS.VW_NSO_1		4241	34
HASH		GROUP BY	4241	34
TABLE ACCESS	XVOTACOES	FULL	39874	32
TABLE ACCESS	XVOTACOES	FULL	39874	32
TABLE ACCESS	XFREGUESIAS	FULL	4241	7

(a) Alínea c no contexto de x



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			22	75
SORT		ORDER BY	22	75
HASH JOIN			22	74
Access Predicates				
V.FREGUESIA=F.CODIGO				
NESTED LOOPS			22	74
NESTED LOOPS				
STATISTICS COLLECTOR				
HASH JOIN		RIGHT SEMI	22	67
Access Predicates				
AND				
V.FREGUESIA=FREGUESIA				
V.VOTOS=MAX(V1.VOTOS)				
VIEW	SYS.VW_NSO_1		4241	34
HASH		GROUP BY	4241	34
TABLE ACCESS	YVOTACOES	FULL	39874	32
TABLE ACCESS	YVOTACOES	FULL	39874	32
INDEX	YFREGUESIAS_CODIGO_PK	UNIQUE SCAN		
Access Predicates				
V.FREGUESIA=F.CODIGO				
TABLE ACCESS	YFREGUESIAS	BY INDEX ROWID	1	7
TABLE ACCESS	YFREGUESIAS	FULL	4241	7

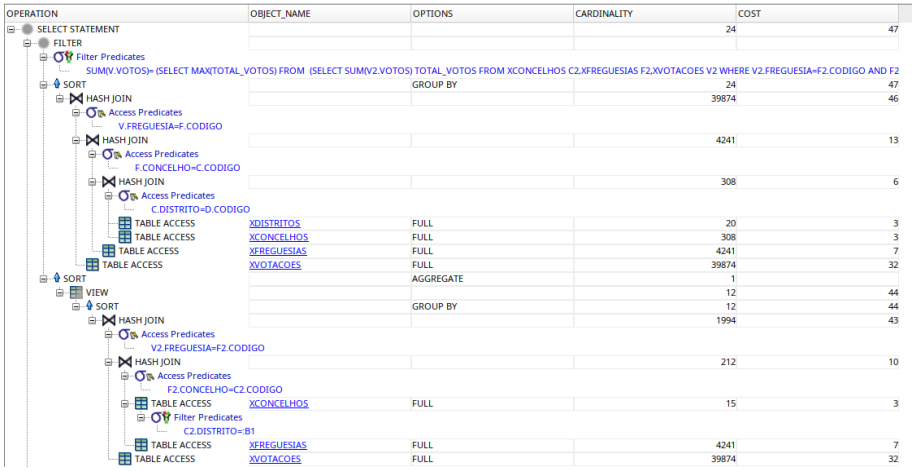
(b) Alínea c no contexto de y



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			22	75
SORT		ORDER BY	22	75
HASH JOIN			22	74
Access Predicates				
V.FREGUESIA=F.CODIGO				
NESTED LOOPS			22	74
NESTED LOOPS				
STATISTICS COLLECTOR				
HASH JOIN		RIGHT SEMI	22	67
Access Predicates				
AND				
V.FREGUESIA=FREGUESIA				
V.VOTOS=MAX(V1.VOTOS)				
SYS_OP_DESCEND(V.VOTOS)=SYS_OP_DESCEND(MAX(V1.VOTOS))				
VIEW	SYS.VW_NSO_1		4241	34
HASH		GROUP BY	4241	34
INDEX	IDX_ZVOTACOES_FREGUESIA_VOTOS	FAST FULL SCAN	39874	32
TABLE ACCESS	ZVOTACOES	FULL	39874	32
INDEX	ZFREGUESIAS_CODIGO_PK	UNIQUE SCAN		
Access Predicates				
V.FREGUESIA=F.CODIGO				
TABLE ACCESS	ZFREGUESIAS	BY INDEX ROWID	1	7
TABLE ACCESS	ZFREGUESIAS	FULL	4241	7

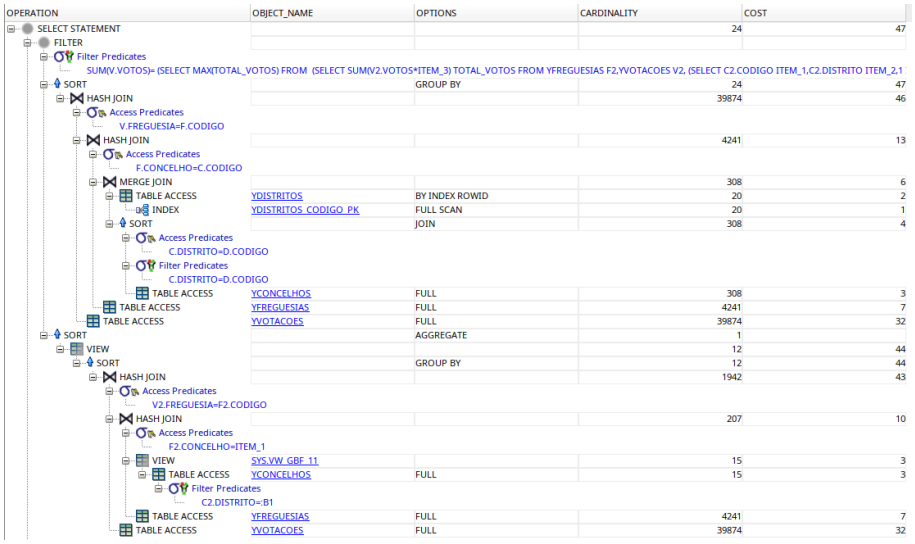
(c) Alínea c no contexto de z

Figure 14: Planos de execução para a alínea c.



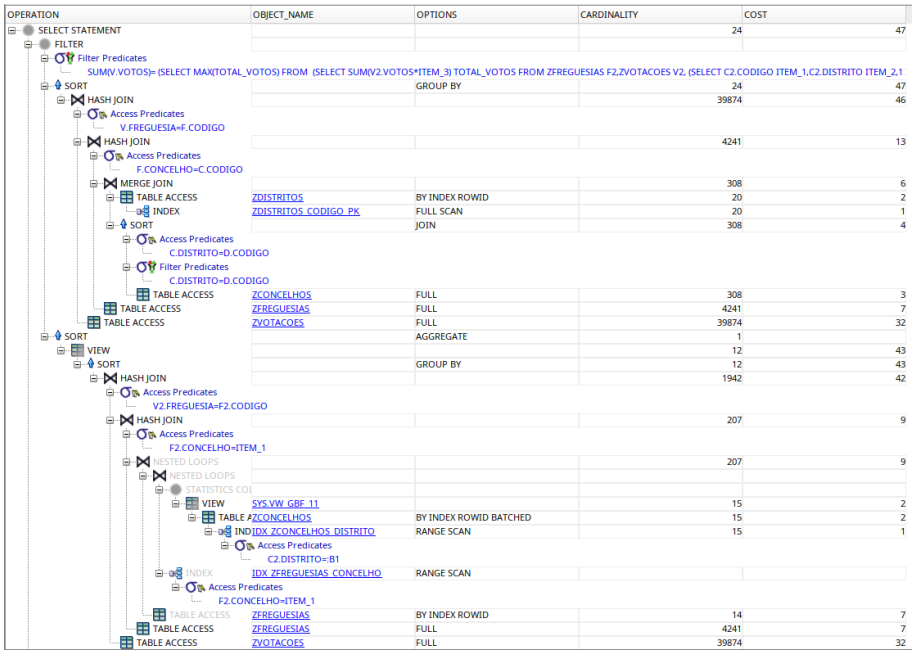
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			24	47
FILTER				
Filter Predicates				
SUM(V.VOTOS)= (SELECT MAX(TOTAL_VOTOS) FROM (SELECT SUM(V2.VOTOS) TOTAL_VOTOS FROM XCONCELHOS C2,XFREGUESIAS F2,XVOTACOES V2 WHERE V2.FREGUESIA=F2.CODIGO AND F2				
SORT		GROUP BY	24	47
HASH JOIN			39874	46
Access Predicates				
V.FREGUESIA=F.CODIGO				
HASH JOIN			4241	13
Access Predicates				
F.CONCELHO=C.CODIGO				
HASH JOIN			308	6
Access Predicates				
C.DISTRITO=D.CODIGO				
TABLE ACCESS	XDISTRITOS	FULL	20	3
TABLE ACCESS	XCONCELHOS	FULL	308	3
TABLE ACCESS	XFREGUESIAS	FULL	4241	7
TABLE ACCESS	XVOTACOES	FULL	39874	32
SORT		AGGREGATE	1	
VIEW			12	44
SORT		GROUP BY	12	44
HASH JOIN			1994	43
Access Predicates				
V2.FREGUESIA=F2.CODIGO				
HASH JOIN			212	10
Access Predicates				
F2.CONCELHO=C2.CODIGO				
TABLE ACCESS	XCONCELHOS	FULL	15	3
Filter Predicates				
C2.DISTRITO=:B1				
TABLE ACCESS	XFREGUESIAS	FULL	4241	7
TABLE ACCESS	XVOTACOES	FULL	39874	32

(a) Alínea d no contexto de x



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			24	47
FILTER				
Filter Predicates				
SUM(V.VOTOS)= (SELECT MAX(TOTAL_VOTOS) FROM (SELECT SUM(V2.VOTOS*ITEM_3) TOTAL_VOTOS FROM YFREGUESIAS F2,YVOTACOES V2, (SELECT C2.CODIGO ITEM_1,C2.DISTRITO ITEM_2,1				
SORT		GROUP BY	24	47
HASH JOIN			39874	46
Access Predicates				
V.FREGUESIA=F.CODIGO				
HASH JOIN			4241	13
Access Predicates				
F.CONCELHO=C.CODIGO				
MERGE JOIN			308	6
TABLE ACCESS	YDISTRITOS	BY INDEX ROWID	20	2
INDEX	YDISTRITOS_CODIGO_PK	FULL SCAN	20	1
SORT		JOIN	308	4
Access Predicates				
C.DISTRITO=D.CODIGO				
Filter Predicates				
C.DISTRITO=D.CODIGO				
TABLE ACCESS	YCONCELHOS	FULL	308	3
TABLE ACCESS	YFREGUESIAS	FULL	4241	7
TABLE ACCESS	YVOTACOES	FULL	39874	32
SORT		AGGREGATE	1	
VIEW			12	44
SORT		GROUP BY	12	44
HASH JOIN			1942	43
Access Predicates				
V2.FREGUESIA=F2.CODIGO				
HASH JOIN			207	10
Access Predicates				
F2.CONCELHO=ITEM_1				
VIEW	SYS.VW_GBF_11		15	3
TABLE ACCESS	YCONCELHOS	FULL	15	3
Filter Predicates				
C2.DISTRITO=:B1				
TABLE ACCESS	YFREGUESIAS	FULL	4241	7
TABLE ACCESS	YVOTACOES	FULL	39874	32

(b) Alínea d no contexto de y



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			24	47
FILTER				
Filter Predicates				
SUM(V.VOTOS)= (SELECT MAX(TOTAL_VOTOS) FROM (SELECT SUM(V2.VOTOS*ITEM_3) TOTAL_VOTOS FROM ZFREGUESIAS F2,ZVOTACOES V2, (SELECT C2.CODIGO ITEM_1,C2.DISTRITO ITEM_2,1				
SORT		GROUP BY	24	47
HASH JOIN			39874	46
Access Predicates				
V.FREGUESIA=F.CODIGO				
HASH JOIN			4241	13
Access Predicates				
F.CONCELHO=C.CODIGO				
MERGE JOIN			308	6
TABLE ACCESS	ZDISTRITOS	BY INDEX ROWID	20	2
INDEX	ZDISTRITOS_CODIGO_PK	FULL SCAN	20	1
SORT		JOIN	308	4
Access Predicates				
C.DISTRITO=D.CODIGO				
Filter Predicates				
C.DISTRITO=D.CODIGO				
TABLE ACCESS	ZCONCELHOS	FULL	308	3
TABLE ACCESS	ZFREGUESIAS	FULL	4241	7
TABLE ACCESS	ZVOTACOES	FULL	39874	32
SORT		AGGREGATE	1	
VIEW			12	43
SORT		GROUP BY	12	43
HASH JOIN			1942	42
Access Predicates				
V2.FREGUESIA=F2.CODIGO				
HASH JOIN			207	9
Access Predicates				
F2.CONCELHO=ITEM_1				
NESTED LOOPS			207	9
NESTED LOOPS				
STATISTICS CO				
VIEW	SYS.VW_GBF_11		15	2
TABLE A	ZCONCELHOS	BY INDEX ROWID BATCHED	15	2
IND	IDX_ZCONCELHOS_DISTRITO	RANGE SCAN	15	1
Access Predicates				
C2.DISTRITO=:B1				
INDEX	IDX_ZFREGUESIAS_CONCELHO	RANGE SCAN		
Access Predicates				
F2.CONCELHO=ITEM_1				
TABLE ACCESS	ZFREGUESIAS	BY INDEX ROWID	14	7
TABLE ACCESS	ZFREGUESIAS	FULL	4241	7
TABLE ACCESS	ZVOTACOES	FULL	39874	32

(c) Alínea d no contexto de z

Figure 15: Planos de execução para a alínea d.

4. **Discussão**

   (a) **2a)**

   Esta pergunta pede para aferir os votos do PS a nível nacional.

   Ao olhar para o plano de execução dos ambientes X e Y e comparando com o ambiente Z, percebemos que os 2 primeiros realizam scans à tabela inteira (custo 32), enquanto que o terceiro utliza o índice que ordena os votos por partido para reduzir o custo da operação (11 - 1/3 apenas). Este índice permite aceder diretamente ao PS na tabela votações e somar todos os valores de votos, resultando numa perfomance menos custosa e superior.

   (b) **2b)**

   Esta pergunta pede para contabilizar os votos de cada partido em cada distrito.

   Neste caso, a performance acabou por ser a mesma para todos os ambientes. Apesar do uso do índice em ZFREGUESIAS, o plano geral continua a ter o mesmo custo estimado porque o ganho do índice é mínimo (afeta poucas linhas) e outras operações mais pesadas (como joins e full scan da tabela grande ZVOTACOES) continuam a dominar o custo.

   (c) **2c)**

   Esta pergunta pede para indicar qual o partido que, ao nível de freguesia, registou o maior número de votos.

   Neste caso, a performance acabou por ser a mesma para todos os ambientes. Apesar do uso do índice em ZVOTACOES que levou a um *fast full scan*, o custo global permanece igual (75) ao do caso y e x, porque o ganho local do scan rápido é insuficiente para alterar o custo total e as outras operações (JOIN, SORT, agregação) dominam o custo.

   (d) **2d)**

   Esta pergunta pede para, para cada distrito, indicar qual o seu nome e a designação e número de votos do partido que nele teve melhor votação.

   Neste caso, a performance acabou por ser a mesma para todos os ambientes. Apesar do uso do índice em ZCONCELHOS E ZFREGUESIAS que levaram a *range scans* com base em *rowid*, o custo global permanece igual (47) ao do caso y e x, porque o ganho local é de apenas 1 unidade.

   Neste caso, o otimizador optou por vários scans completos apesar da presença de índices na tabela, o que indica que para esta operação em específico, estes poderiam ser pouco vantajosos.

4.3 Negação

A negação é utilizada para filtrar dados que não atendem a uma determinada condição. É frequentemente implementada com operadores como NOT, <> (diferente de) e NOT IN, permitindo excluir valores específicos ou grupos de resultados de uma consulta.

Esta é a questão proposta:

Quais os partidos que não concorreram no distrito de Lisboa?

1. **Formulação em SQL**

```sql
SELECT p.sigla, p.designacao
FROM xpartidos p
WHERE p.sigla NOT IN (
    SELECT DISTINCT l.partido
    FROM xlistas l
    JOIN xdistritos d ON l.distrito = d.codigo
    WHERE d.nome = 'Lisboa'
);
```

2. **Resultados**

SIGLA	DESIGNACAO
PDA	Partido Democrático do Atlântico

Figure 16: Resultados da pergunta 3.

3. **Planos de execução**

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			3	9
HASH JOIN		ANTI	3	9
Access Predicates				
P.SIGLA=PARTIDO				
TABLE ACCESS	XPARTIDOS	FULL	12	3
VIEW	SYS.VW_NSO_1		9	6
HASH JOIN			9	6
Access Predicates				
L.DISTRITO=D.CODIGO				
TABLE ACCESS	XDISTRITOS	FULL	1	3
Filter Predicates				
D.NOME='Lisboa'				
TABLE ACCESS	XLISTAS	FULL	182	3

(a) Pergunta 3 no contexto de x

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			3	7
MERGE JOIN		ANTI	3	7
TABLE ACCESS	YPARTIDOS	BY INDEX ROWID	12	2
INDEX	YPARTIDOS_SIGLA_PK	FULL SCAN	12	1
SORT		UNIQUE	9	5
Access Predicates				
P.SIGLA=PARTIDO				
Filter Predicates				
P.SIGLA=PARTIDO				
VIEW	SYS.VW_NSO_1		9	4
HASH JOIN			COST=1 9	4
Access Predicates				
L.DISTRITO=D.CODIGO				
NESTED LOOPS			9	4
STATISTICS COLLECTOR				
TABLE ACCESS	YDISTRITOS	FULL	1	3
Filter Predicates				
D.NOME='Lisboa'				
INDEX	YLISTAS_DISTRITO_PARTIDO_PK	RANGE SCAN	9	1
Access Predicates				
L.DISTRITO=D.CODIGO				
INDEX	YLISTAS_DISTRITO_PARTIDO_PK	FULL SCAN	9	1

(b) Pergunta 3 no contexto de y

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			3	7
MERGE JOIN		ANTI	3	7
TABLE ACCESS	ZPARTIDOS	BY INDEX ROWID	12	2
INDEX	ZPARTIDOS_SIGLA_PK	FULL SCAN	12	1
SORT		UNIQUE	9	5
Access Predicates				
P.SIGLA=PARTIDO				
Filter Predicates				
P.SIGLA=PARTIDO				
VIEW	SYS.VW_NSO_1		9	4
HASH JOIN			9	4
Access Predicates				
L.DISTRITO=D.CODIGO				
NESTED LOOPS			9	4
STATISTICS COLLECTOR				
TABLE ACCESS	ZDISTRITOS	FULL	1	3
Filter Predicates				
D.NOME='Lisboa'				
INDEX	ZLISTAS_DISTRITO_PARTIDO_PK	RANGE SCAN	9	1
Access Predicates				
L.DISTRITO=D.CODIGO				
INDEX	ZLISTAS_DISTRITO_PARTIDO_PK	FULL SCAN	9	1

(c) Pergunta 3 no contexto de z

Figure 17: Planos de execução para a pergunta 3.

4. **Discussão**

   Esta pergunta pedia para indicar os partidos que não concorreram no distrito de Lisboa.

   É de realçar que a introdução de restrições standard (chave primária e externa) foi o maior diferenciador de desempenho.

   Curiosamente, o otimizador não utilizou nenhum dos índices bitmap definidos para o ambiente Z, optando por um range scan baseado na chave primária composta da tabela ZLISTAS, mesmo com a operação sendo um NOT IN, onde índices bitmap costumam ser bastante eficazes - ideais para operações do tipo NOT IN, já que permitem verificar de forma eficiente a ausência de combinações (como partidos que não concorreram num distrito específico).

   O otimizador pode ter optado por não os usar devido a heurísticas internas, como:

   A baixa cardinalidade do filtro (D.NOME = 'Lisboa'), o que reduz o benefício do bitmap;

   O custo estimado de usar o índice bitmap ser maior que a simples varredura pela chave composta;

   A presença de restrições bem definidas, como a chave primária composta em ZLISTAS(DISTRITO, PARTIDO), que permite ao otimizador realizar um acesso direto e eficiente, com baixa cardinalidade (9) e baixo custo (1).

   Portanto, embora o bitmap seja teoricamente vantajoso para este tipo de operação, o plano escolhido foi suficientemente eficiente e, no entendimento do otimizador, não justificava o uso do índice bitmap neste cenário específico.

4.4 Vistas

A pergunta "Houve algum partido a vencer em todos as freguesias de um concelho do distrito do Porto?" é de natureza universal.

Indique código do concelho e sigla do partido que é de natureza universal.

Compare do ponto de vista temporal e de plano de execução as estratégias da dupla negação e da contagem em três situações diferentes (só no contexto Z):

a. Sem vista para calcular o vencedor em cada freguesia.

b. Com vista.

c. Com vista materializada (eventualmente com índices).

1. **Formulação em SQL**

```sql
-- a.
-- Double Negation
SELECT c.codigo AS concelho, v.partido
FROM zconcelhos c
JOIN zfreguesias f ON f.concelho = c.codigo
JOIN zdistritos d ON c.distrito = d.codigo
JOIN zvotacoes v ON v.freguesia = f.codigo
WHERE d.nome = 'Porto'
AND NOT EXISTS (
    SELECT 1
    FROM zfreguesias f2
    WHERE f2.concelho = c.codigo
    AND NOT EXISTS (
        SELECT 1
        FROM zvotacoes v2
        WHERE v2.freguesia = f2.codigo
        AND v2.partido = v.partido
        AND v2.votos = (
```

```
            SELECT MAX(v3.votos)
            FROM zvotacoes v3
            WHERE v3.freguesia = f2.codigo
        )
    )
)
GROUP BY c.codigo, v.partido;

-- Count
SELECT c.codigo AS concelho, v.partido
FROM zconcelhos c
JOIN zfreguesias f ON f.concelho = c.codigo
JOIN zdistritos d ON c.distrito = d.codigo
JOIN zvotacoes v ON v.freguesia = f.codigo
WHERE d.nome = 'Porto'
AND v.votos = (
    SELECT MAX(v2.votos)
    FROM zvotacoes v2
    WHERE v2.freguesia = v.freguesia
)
GROUP BY c.codigo, v.partido
HAVING COUNT(DISTINCT v.freguesia) = (
    SELECT COUNT(*)
    FROM zfreguesias f2
    WHERE f2.concelho = c.codigo
)

-- b.
CREATE OR REPLACE VIEW zvencedor_freguesia AS
SELECT v.freguesia, v.partido
FROM zvotacoes v
WHERE v.votos = (
    SELECT MAX(v2.votos)
    FROM zvotacoes v2
    WHERE v2.freguesia = v.freguesia
);

-- Double Negation
SELECT c.codigo AS concelho, vf.partido
FROM zconcelhos c
JOIN zfreguesias f ON f.concelho = c.codigo
JOIN zdistritos d ON c.distrito = d.codigo
JOIN zvencedor_freguesia vf ON vf.freguesia = f.codigo
WHERE d.nome = 'Porto'
AND NOT EXISTS (
    SELECT 1
    FROM zfreguesias f2
    WHERE f2.concelho = c.codigo
    AND NOT EXISTS (
        SELECT 1
        FROM zvencedor_freguesia vf2
        WHERE vf2.freguesia = f2.codigo
        AND vf2.partido = vf.partido
    )
)
GROUP BY c.codigo, vf.partido;

-- Count
SELECT c.codigo AS concelho, vf.partido
FROM zconcelhos c
JOIN zfreguesias f ON f.concelho = c.codigo
```

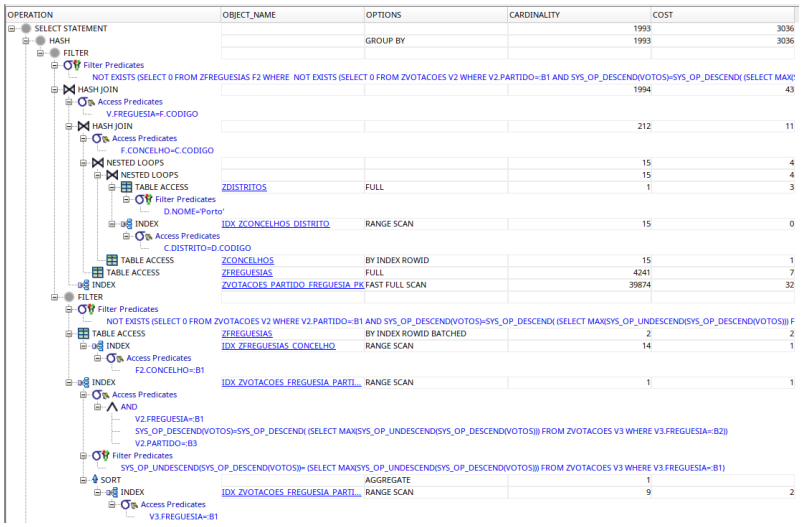
```
JOIN zdistritos d ON c.distrito = d.codigo
JOIN zvencedor_freguesia vf ON vf.freguesia = f.codigo
WHERE d.nome = 'Porto'
GROUP BY c.codigo, vf.partido
HAVING COUNT(DISTINCT vf.freguesia) = (
    SELECT COUNT(*) FROM zfreguesias f2 WHERE f2.concelho = c.codigo
);

-- c.
CREATE MATERIALIZED VIEW zmv_vencedor_freguesia
BUILD IMMEDIATE REFRESH COMPLETE AS
SELECT v.freguesia, v.partido
FROM zvotacoes v
WHERE v.votos = (
    SELECT MAX(v2.votos)
    FROM zvotacoes v2
    WHERE v2.freguesia = v.freguesia
);

CREATE INDEX IDX_MV_VENCEDOR_FREGUESIA ON zmv_vencedor_freguesia (
   freguesia, partido);

-- Double Negation
SELECT c.codigo AS concelho, mv.partido
FROM zconcelhos c
JOIN zfreguesias f ON f.concelho = c.codigo
JOIN zdistritos d ON c.distrito = d.codigo
JOIN zmv_vencedor_freguesia mv ON mv.freguesia = f.codigo
WHERE d.nome = 'Porto'
AND NOT EXISTS (
    SELECT 1
    FROM zfreguesias f2
    WHERE f2.concelho = c.codigo
    AND NOT EXISTS (
        SELECT 1
        FROM zmv_vencedor_freguesia mv2
        WHERE mv2.freguesia = f2.codigo
        AND mv2.partido = mv.partido
    )
)
GROUP BY c.codigo, mv.partido;

-- Count
SELECT c.codigo AS concelho, mv.partido
FROM zconcelhos c
JOIN zfreguesias f ON f.concelho = c.codigo
JOIN zdistritos d ON c.distrito = d.codigo
JOIN zmv_vencedor_freguesia mv ON mv.freguesia = f.codigo
WHERE d.nome = 'Porto'
GROUP BY c.codigo, mv.partido
HAVING COUNT(DISTINCT mv.freguesia) = (
    SELECT COUNT(*) FROM zfreguesias f2 WHERE f2.concelho = c.codigo
);
```

2. **Resultados**

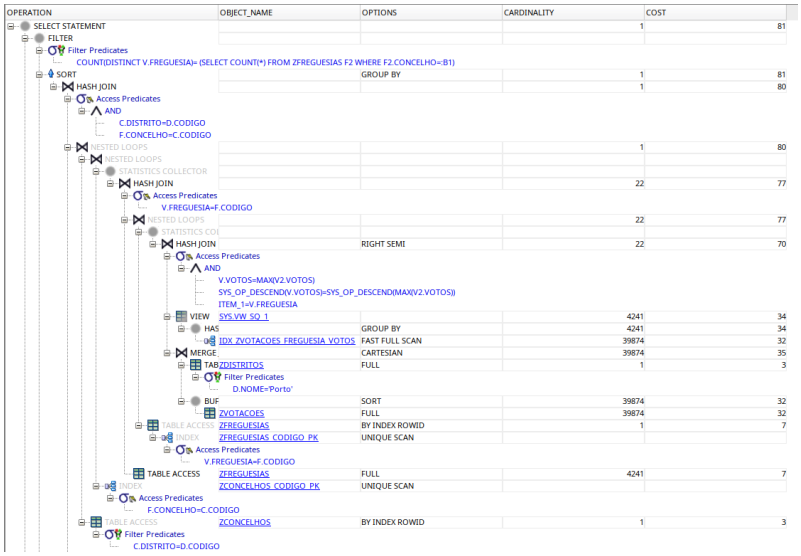
CONCELHO	PARTIDO
1304	PS
1308	PS

Figure 18: Resultados da pergunta 4.

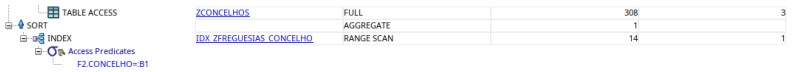
3. Planos de execução



(a) Alínea a utilizando dupla negação

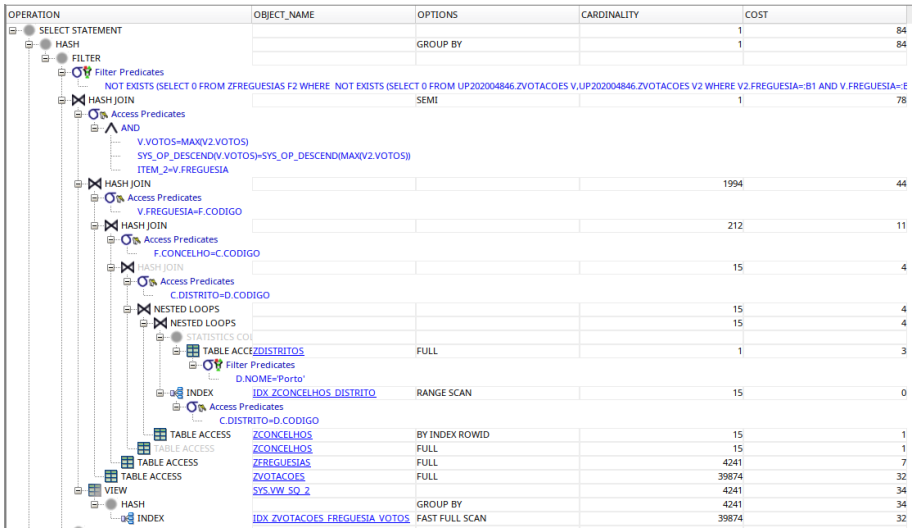


(b) Alínea a utilizando contagem



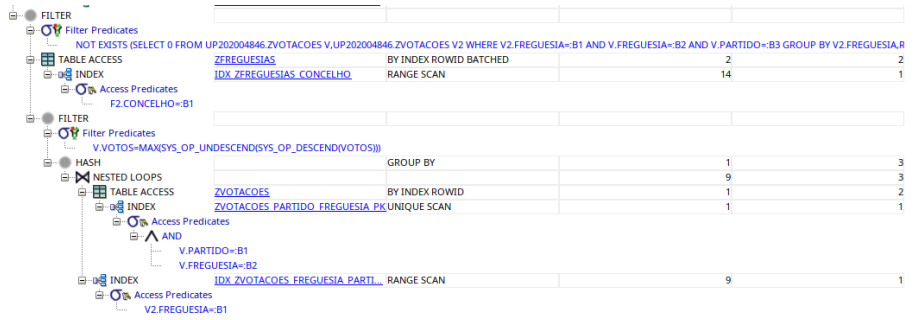
(c) Alínea a utilizando contagem (continuação)

Figure 19: Planos de execução para a alínea a.



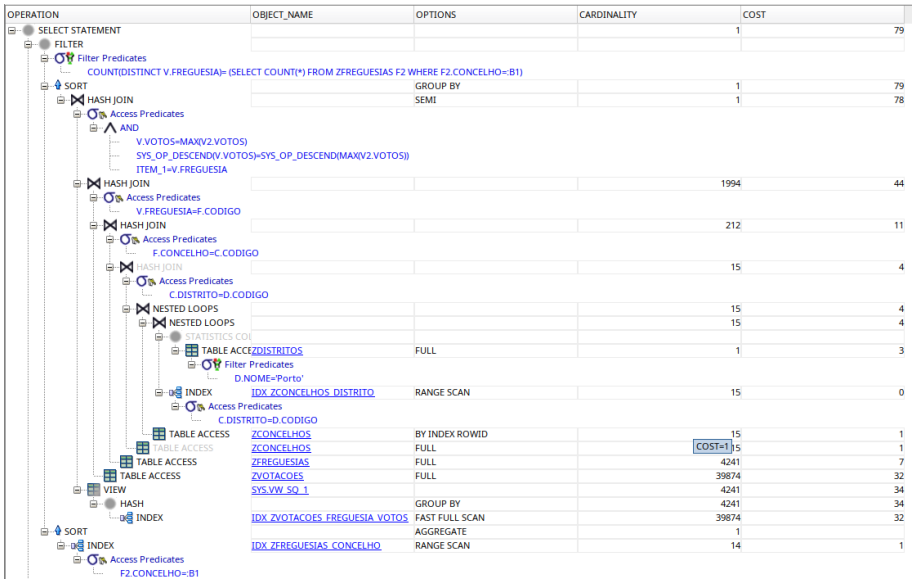
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	84
HASH		GROUP BY	1	84
FILTER				
Filter Predicates				
NOT EXISTS (SELECT 0 FROM ZFREGUESIAS F2 WHERE NOT EXISTS (SELECT 0 FROM UP202004846.ZVOTACOES V,UP202004846.ZVOTACOES V2 WHERE V2.FREGUESIA=:B1 AND V.FREGUESIA=:E				
HASH JOIN		SEMI	1	78
Access Predicates				
AND				
V.VOTOS=MAX(V2.VOTOS)				
SYS_OP_DESCEND(V.VOTOS)=SYS_OP_DESCEND(MAX(V2.VOTOS))				
ITEM_2=V.FREGUESIA				
HASH JOIN			1994	44
Access Predicates				
V.FREGUESIA=F.CODIGO				
HASH JOIN			212	11
Access Predicates				
F.CONCELHO=C.CODIGO				
HASH JOIN			15	4
Access Predicates				
C.DISTRITO=D.CODIGO				
NESTED LOOPS			15	4
NESTED LOOPS			15	4
STATISTICS COL				
TABLE ACCE	ZDISTRITOS	FULL	1	3
Filter Predicates				
D.NOME='Porto'				
INDEX	IDX ZCONCELHOS_DISTRITO	RANGE SCAN	15	0
Access Predicates				
C.DISTRITO=D.CODIGO				
TABLE ACCESS	ZCONCELHOS	BY INDEX ROWID	15	1
TABLE ACCESS	ZCONCELHOS	FULL	15	1
TABLE ACCESS	ZFREGUESIAS	FULL	4241	7
TABLE ACCESS	ZVOTACOES	FULL	39874	32
VIEW	SYS.VW_SQ_2		4241	34
HASH		GROUP BY	4241	34
INDEX	IDX_ZVOTACOES_FREGUESIA_VOTOS	FAST FULL SCAN	39874	32

(a) Alínea b utilizando dupla negação



FILTER				
Filter Predicates				
NOT EXISTS (SELECT 0 FROM UP202004846.ZVOTACOES V,UP202004846.ZVOTACOES V2 WHERE V2.FREGUESIA=:B1 AND V.FREGUESIA=:B2 AND V.PARTIDO=:B3 GROUP BY V2.FREGUESIA,R				
TABLE ACCESS	ZFREGUESIAS	BY INDEX ROWID BATCHED	2	2
INDEX	IDX_ZFREGUESIAS_CONCELHO	RANGE SCAN	14	1
Access Predicates				
F2.CONCELHO=:B1				
FILTER				
Filter Predicates				
V.VOTOS=MAX(SYS_OP_UNDESCEND(SYS_OP_DESCEND(VOTOS)))				
HASH		GROUP BY	1	3
NESTED LOOPS			9	3
TABLE ACCESS	ZVOTACOES	BY INDEX ROWID	1	2
INDEX	ZVOTACOES_PARTIDO_FREGUESIA_PK	UNIQUE SCAN	1	1
Access Predicates				
AND				
V.PARTIDO=:B1				
V.FREGUESIA=:B2				
INDEX	IDX_ZVOTACOES_FREGUESIA_PARTI...	RANGE SCAN	9	1
Access Predicates				
V2.FREGUESIA=:B1				

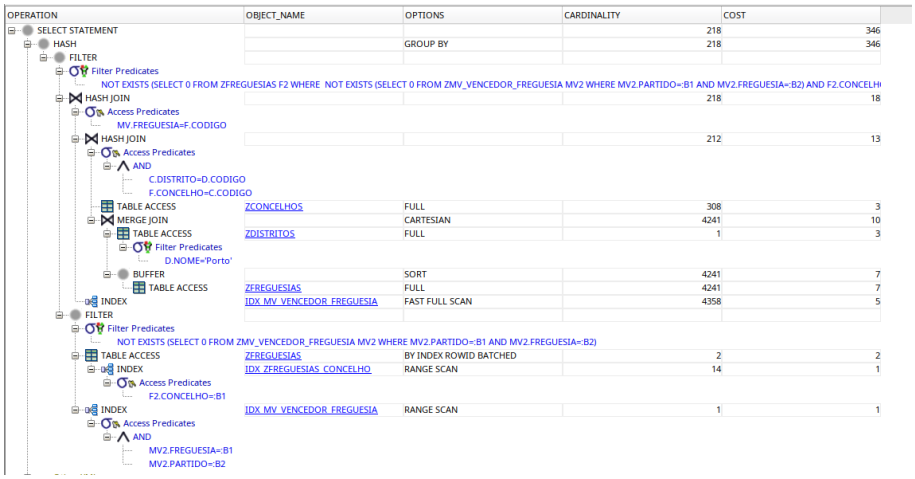
(b) Alínea b utilizando dupla negação (continuação)



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	79
FILTER				
Filter Predicates				
COUNT(DISTINCT V.FREGUESIA)= (SELECT COUNT(*) FROM ZFREGUESIAS F2 WHERE F2.CONCELHO=:B1)				
SORT		GROUP BY	1	79
HASH JOIN		SEMI	1	78
Access Predicates				
AND				
V.VOTOS=MAX(V2.VOTOS)				
SYS_OP_DESCEND(V.VOTOS)=SYS_OP_DESCEND(MAX(V2.VOTOS))				
ITEM_1=V.FREGUESIA				
HASH JOIN			1994	44
Access Predicates				
V.FREGUESIA=F.CODIGO				
HASH JOIN			212	11
Access Predicates				
F.CONCELHO=C.CODIGO				
HASH JOIN			15	4
Access Predicates				
C.DISTRITO=D.CODIGO				
NESTED LOOPS			15	4
NESTED LOOPS			15	4
STATISTICS COL				
TABLE ACCE	DISTRITOS	FULL	1	3
Filter Predicates				
D.NOME='Porto'				
INDEX	IDX ZCONCELHOS_DISTRITO	RANGE SCAN	15	0
Access Predicates				
C.DISTRITO=D.CODIGO				
TABLE ACCESS	ZCONCELHOS	BY INDEX ROWID	15	1
TABLE ACCESS	ZCONCELHOS	FULL	COST=1 15	1
TABLE ACCESS	ZFREGUESIAS	FULL	4241	7
TABLE ACCESS	ZVOTACOES	FULL	39874	32
VIEW	SYS.VW_SQ_1		4241	34
HASH		GROUP BY	4241	34
INDEX	IDX_ZVOTACOES_FREGUESIA_VOTOS	FAST FULL SCAN	39874	32
SORT		AGGREGATE	1	
INDEX	IDX_ZFREGUESIAS_CONCELHO	RANGE SCAN	14	1
Access Predicates				
F2.CONCELHO=:B1				

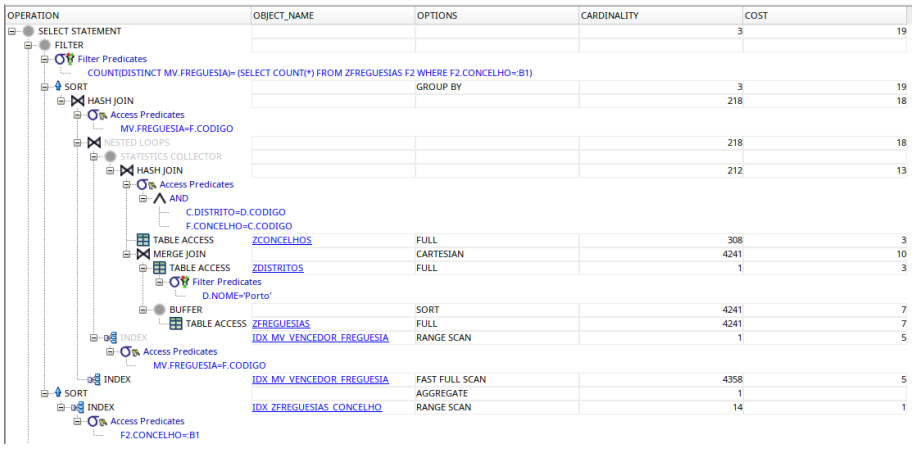
(c) Alínea b utilizando contagem

Figure 20: Planos de execução para a alínea b.



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			218	346
HASH		GROUP BY	218	346
FILTER				
Filter Predicates				
NOT EXISTS (SELECT 0 FROM ZFREGUESIAS F2 WHERE NOT EXISTS (SELECT 0 FROM ZMV_VENCEDOR_FREGUESIA MV2 WHERE MV2.PARTIDO=:B1 AND MV2.FREGUESIA=:B2) AND F2.CONCELH				
HASH JOIN			218	18
Access Predicates				
MV.FREGUESIA=F.CODIGO				
HASH JOIN			212	13
Access Predicates				
AND				
C.DISTRITO=D.CODIGO				
F.CONCELHO=C.CODIGO				
TABLE ACCESS	ZCONCELHOS	FULL	308	3
MERGE JOIN		CARTESIAN	4241	10
TABLE ACCESS	ZDISTRITOS	FULL	1	3
Filter Predicates				
D.NOME='Porto'				
BUFFER		SORT	4241	7
TABLE ACCESS	ZFREGUESIAS	FULL	4241	7
INDEX	IDX_MV_VENCEDOR_FREGUESIA	FAST FULL SCAN	4358	5
FILTER				
Filter Predicates				
NOT EXISTS (SELECT 0 FROM ZMV_VENCEDOR_FREGUESIA MV2 WHERE MV2.PARTIDO=:B1 AND MV2.FREGUESIA=:B2)				
TABLE ACCESS	ZFREGUESIAS	BY INDEX ROWID BATCHED	2	2
INDEX	IDX_ZFREGUESIAS_CONCELHO	RANGE SCAN	14	1
Access Predicates				
F2.CONCELHO=:B1				
INDEX	IDX_MV_VENCEDOR_FREGUESIA	RANGE SCAN	1	1
Access Predicates				
AND				
MV2.FREGUESIA=:B1				
MV2.PARTIDO=:B2				

(a) Alínea c utilizando dupla negação



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			3	19
FILTER				
Filter Predicates				
COUNT(DISTINCT MV.FREGUESIA)= (SELECT COUNT(*) FROM ZFREGUESIAS F2 WHERE F2.CONCELHO=:B1)				
SORT		GROUP BY	3	19
HASH JOIN			218	18
Access Predicates				
MV.FREGUESIA=F.CODIGO				
NESTED LOOPS			218	18
STATISTICS COLLECTOR				
HASH JOIN			212	13
Access Predicates				
AND				
C.DISTRITO=D.CODIGO				
F.CONCELHO=C.CODIGO				
TABLE ACCESS	ZCONCELHOS	FULL	308	3
MERGE JOIN		CARTESIAN	4241	10
TABLE ACCESS	ZDISTRITOS	FULL	1	3
Filter Predicates				
D.NOME='Porto'				
BUFFER		SORT	4241	7
TABLE ACCESS	ZFREGUESIAS	FULL	4241	7
INDEX	IDX_MV_VENCEDOR_FREGUESIA	RANGE SCAN	1	5
Access Predicates				
MV.FREGUESIA=F.CODIGO				
INDEX	IDX_MV_VENCEDOR_FREGUESIA	FAST FULL SCAN	4358	5
SORT		AGGREGATE	1	
INDEX	IDX_ZFREGUESIAS_CONCELHO	RANGE SCAN	14	1
Access Predicates				
F2.CONCELHO=:B1				

(b) Alínea c utilizando contagem

Figure 21: Planos de execução para a alínea c.

4. **Discussão**

   Esta pergunta pede para aferir se houve algum partido a vencer em todas as freguesias de um concelho do distrito do Porto. É também pedido para utilizar uma estratégia de dupla negação e uma de contagem.

   (a) **4a)**

   Os planos de execução mostram que a operação de dupla negação tem um custo muito superior à de contagem (3036 vs 81), assim como cardinalidade (1993 vs 1).

   Isto pode ser explicado pela maior complexidade lógica da abordagem com dupla negação, que envolve múltiplas subconsultas aninhadas e verificações linha-a-linha para cada combinação de concelho e partido. Estas subconsultas tornam difícil para o otimizador prever corretamente a cardinalidade e aplicar estratégias eficientes de execução, resultando num maior custo computacional e uso de recursos. Em contraste, a abordagem baseada em contagem utiliza agregações simples (GROUP BY e HAVING), que são mais facilmente otimizáveis, permitindo ao sistema gerar planos de execução mais eficientes e com menor custo estimado.

   (b) **4b)**

   A dupla negação melhorou imensamente a sua performance com a vista, passando de uma cardinalidade de 1993 para 1 e um custo de 3036 para 84. A contagem também melhorou ligeiramente o seu custo, de 81 para 79.

   Isto deve-se ao retirar do cálculo pesado de MAX(votos) de dentro da subconsulta, graças à vista. No entanto, a abordagem com COUNT continua ligeiramente mais eficiente, pois usa operações com que o otimizador lida melhor (agregações e junções simples), sem lógica de negação ou subconsultas aninhadas.

   (c) **4c)**

   A vista materializada melhorou bastante a perfomance da contagem, já que o seu custo passou para apenas 19. No entanto, a operação de dupla negação piorou bastante, aumentando o seu custo para 346 e tendo uma cardinalidade de 218.

   Isto deriva de como o otimizador de consultas trata subconsultas correlacionadas e operações de exclusão (NOT EXISTS) quando trabalha com dados pré-agregados.

   No caso da contagem, a vista materializada fornece dados já reduzidos aos vencedores por freguesia, o que permite ao otimizador aplicar diretamente operações de GROUP BY e COUNT sobre um conjunto menor de dados, tirando bom proveito dos índices e gerando um plano de execução simples e eficiente.

   Já na dupla negação, o uso de duas subconsultas aninhadas com NOT EXISTS torna o plano de execução mais dependente de nested loops e lookups múltiplos na vista materializada. Como o otimizador não pode reescrever a vista como faria com uma view normal, a consulta perde oportunidades de otimização. Além disso, a verificação da existência de combinações ausentes entre freguesias e partidos exige varrimentos mais complexos que não beneficiam tanto dos índices criados, o que contribui para o aumento expressivo do custo e da cardinalidade.

4.5 Índices

Compare os planos de execução da pesquisa "Quantos votos tiveram o PS e o PSD nos distritos 11, 15 e 17", considerando no contexto Z

a. Com índices árvore-B em zconcelhos.distrito e zvotacoes.partido.

b. Com índices bitmap.

1. **Formulação em SQL**

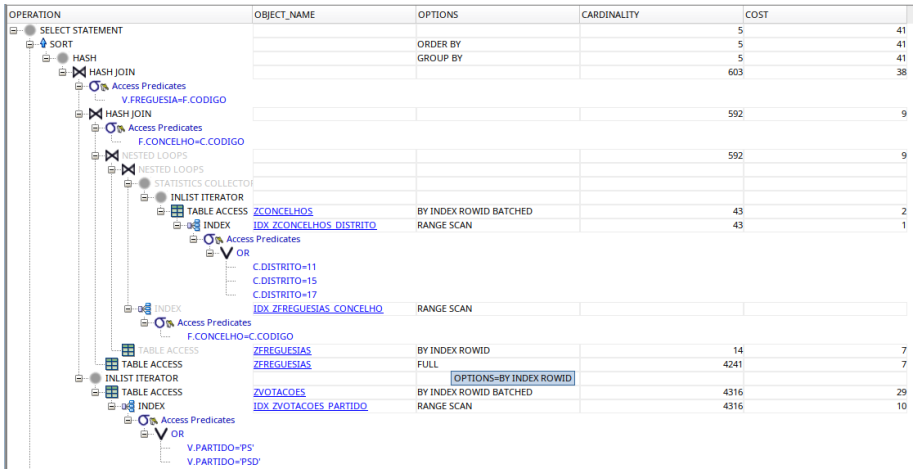
```sql
SELECT c.distrito, v.partido, SUM(v.votos) AS total_votos
FROM zvotacoes v
JOIN zfreguesias f ON v.freguesia = f.codigo
JOIN zconcelhos c ON f.concelho = c.codigo
WHERE c.distrito IN (11, 15, 17)
AND v.partido IN ('PS', 'PSD')
GROUP BY c.distrito, v.partido
ORDER BY c.distrito, total_votos DESC;

-- Force Bitmap
SELECT /*+ INDEX(v BM_ZVOTACOES_PARTIDO) */
       c.distrito, v.partido, SUM(v.votos) AS total_votos
FROM zvotacoes v
JOIN zfreguesias f ON v.freguesia = f.codigo
JOIN zconcelhos c ON f.concelho = c.codigo
WHERE c.distrito IN (11, 15, 17)
AND v.partido IN ('PS', 'PSD')
GROUP BY c.distrito, v.partido
ORDER BY c.distrito, total_votos DESC;
```

2. **Resultados**

DISTRITO	PARTIDO	TOTAL_VOTOS
11	PS	480410
15	PS	170193
17	PS	50691

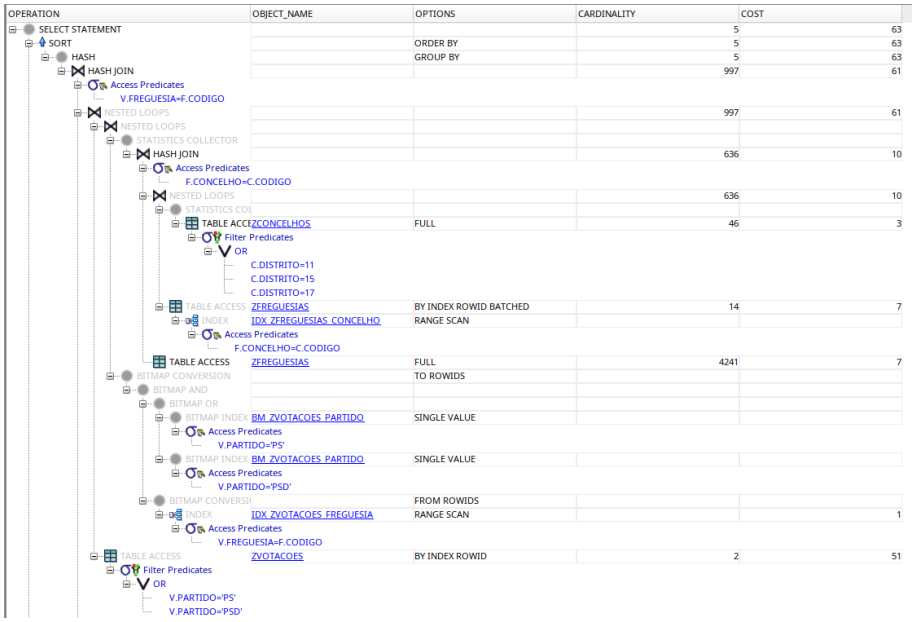
Figure 22: Resultados da pergunta 5.

3. Planos de execução

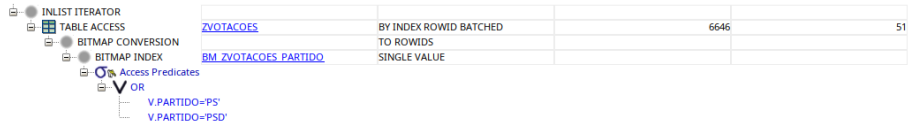


OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			5	41
SORT		ORDER BY	5	41
HASH		GROUP BY	5	41
HASH JOIN			603	38
Access Predicates				
V.FREGUESIA=F.CODIGO				
HASH JOIN			592	9
Access Predicates				
F.CONCELHO=C.CODIGO				
NESTED LOOPS			592	9
NESTED LOOPS				
STATISTICS COLLECTOR				
INLIST ITERATOR				
TABLE ACCESS	ZCONCELHOS	BY INDEX ROWID BATCHED	43	2
INDEX	IDX_ZCONCELHOS_DISTRITO	RANGE SCAN	43	1
Access Predicates				
OR				
C.DISTRITO=11				
C.DISTRITO=15				
C.DISTRITO=17				
INDEX	IDX_ZFREGUESIAS_CONCELHO	RANGE SCAN		
Access Predicates				
F.CONCELHO=C.CODIGO				
TABLE ACCESS	ZFREGUESIAS	BY INDEX ROWID	14	7
TABLE ACCESS	ZFREGUESIAS	FULL	4241	7
INLIST ITERATOR		OPTIONS=BY INDEX ROWID		
TABLE ACCESS	ZVOTACOES	BY INDEX ROWID BATCHED	4316	29
INDEX	IDX_ZVOTACOES_PARTIDO	RANGE SCAN	4316	10
Access Predicates				
OR				
V.PARTIDO='PS'				
V.PARTIDO='PSD'				

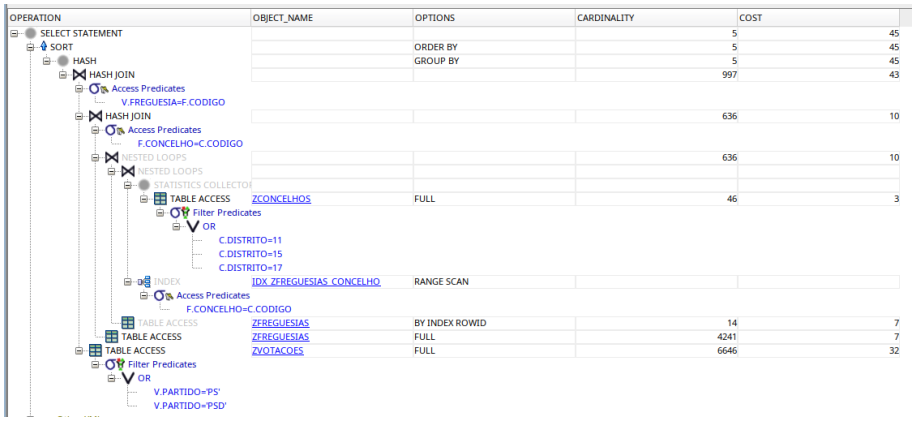
Figure 23: Planos de execução para a alínea a da pergunta 5



(a) Alínea b - com indíces bitmap



(b) Alínea b - com indíces bitmap (continuação)



(c) Alínea b - sem indíces bitmap

Figure 24: Planos de execução para a alínea b.

4. **Discussão**

   Esta pergunta pedia para contar os votos de PS e PSD nos distritos 11, 15 e 17, com índices B-Tree e BitMap e comparar.

   (a) **5a)**

   Os índices B-Tree definidos em zconcelhos.distrito e zvotacoes.partido são utlizados para range scans e acessos por rowid às tabelas. Resultam em boa perfomance, com um custo de apenas 41.

   (b) **5b)**

   Curiosamente, o otimizador optou por não utilizar o índice bitmap, então a query foi reformulada para forçar o seu uso. O custo acabou por ser maior. Isto deve-se ao facto de índices bitmap não serem necessariamente mais eficientes em contextos com junções e operações de agregação em grandes volumes de dados, como GROUP BY e SUM, especialmente quando o número de linhas envolvidas e os valores distintos são baixos (como apenas dois partidos).

   Enquanto os índices bitmap são muito eficazes em consultas sobre colunas com baixa cardinalidade e sem atualizações frequentes, a sua aplicação em contextos com múltiplas tabelas e filtros combinados pode introduzir overhead adicional.

5 Conclusão

Em conclusão, este trabalho foi importante para perceber a influência que a otimização de interrogações em SQL tem sobre a perfomance do motor de base de dados, e o quão importantes são índices personalizados para reduzir o número de linhas a serem analisadas.