CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

Departamento de Computação - DECOM

Laboratório de Banco de Dados I

Exercício 05 - Processamento e Otimização de Consultas0

Ana Clara Cunha Lopes - Engenharia da Computação
Thiago Ribeiro Corrêa - Engenharia da Computação

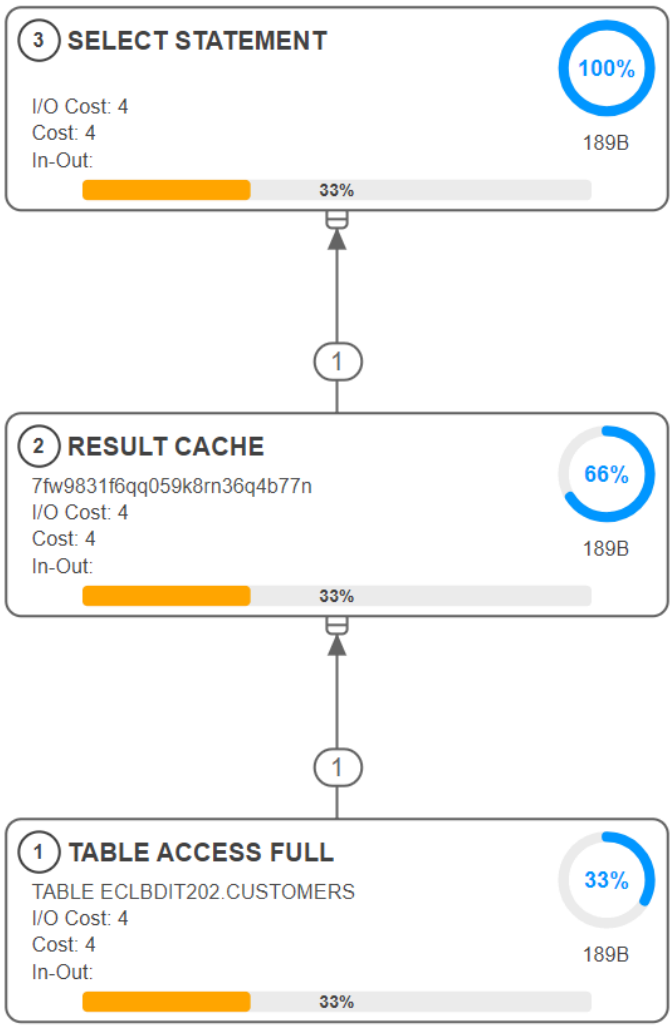
Professor: Evandrino Barros

Belo Horizonte

02 de agosto de 2024

Consulta 01

```
-- consulta01

select * from customers where cust_id=4090;
```



A consulta SQL busca todas as colunas da tabela customers para encontrar cust_id = 4090. O plano de execução possui três etapas: TABLE ACCESS FULL, RESULT CACHE, e SELECT STATEMENT.

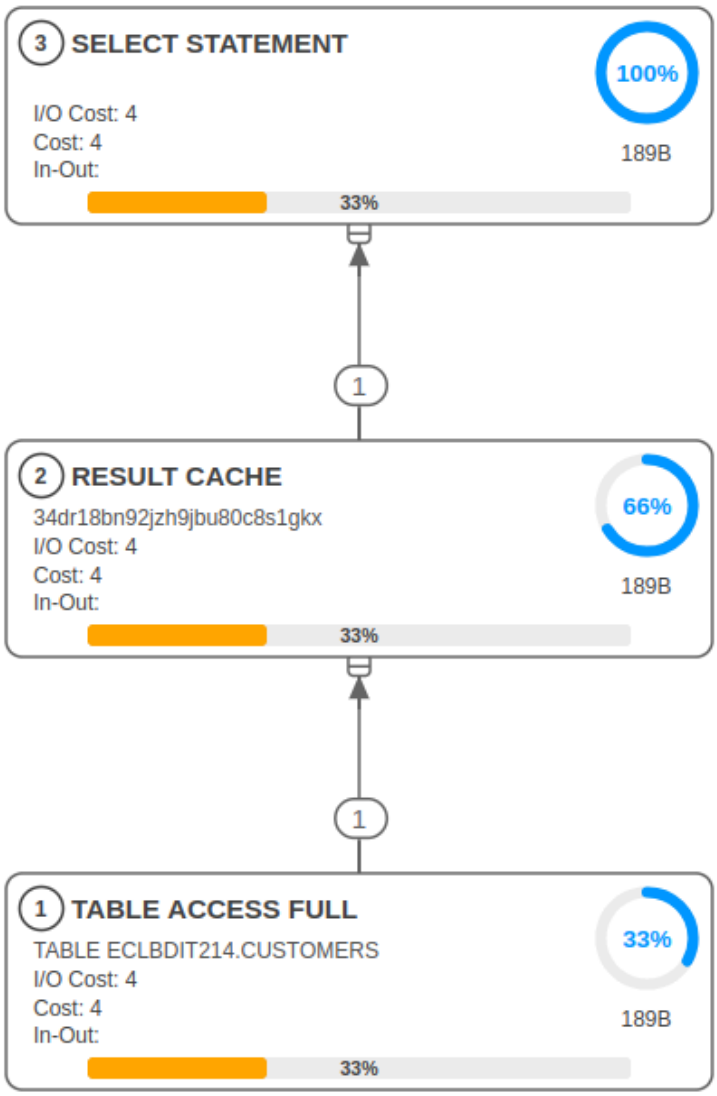
Na TABLE ACCESS FULL, a consulta realiza um acesso completo à tabela, lendo todas as linhas para encontrar a linha desejada. Este é um processo custoso em termos de I/O, representando 33% do custo total da consulta. O RESULT CACHE utiliza uma cache de resultados para armazenar dados intermediários (ou finais), acelerando execuções subsequentes da mesma consulta, representando 66% do custo total. Já o SELECT STATEMENT engloba todas as operações anteriores, resultando em um custo total de 4.

A seletividade da consulta, calculada como a razão entre o número de linhas que satisfazem a condição de busca e o número total de linhas na tabela, é muito baixa

(0,000018). Isso indica que a consulta é altamente seletiva, retornando apenas uma linha entre 55500, o que é bom para o desempenho.

Consulta 02

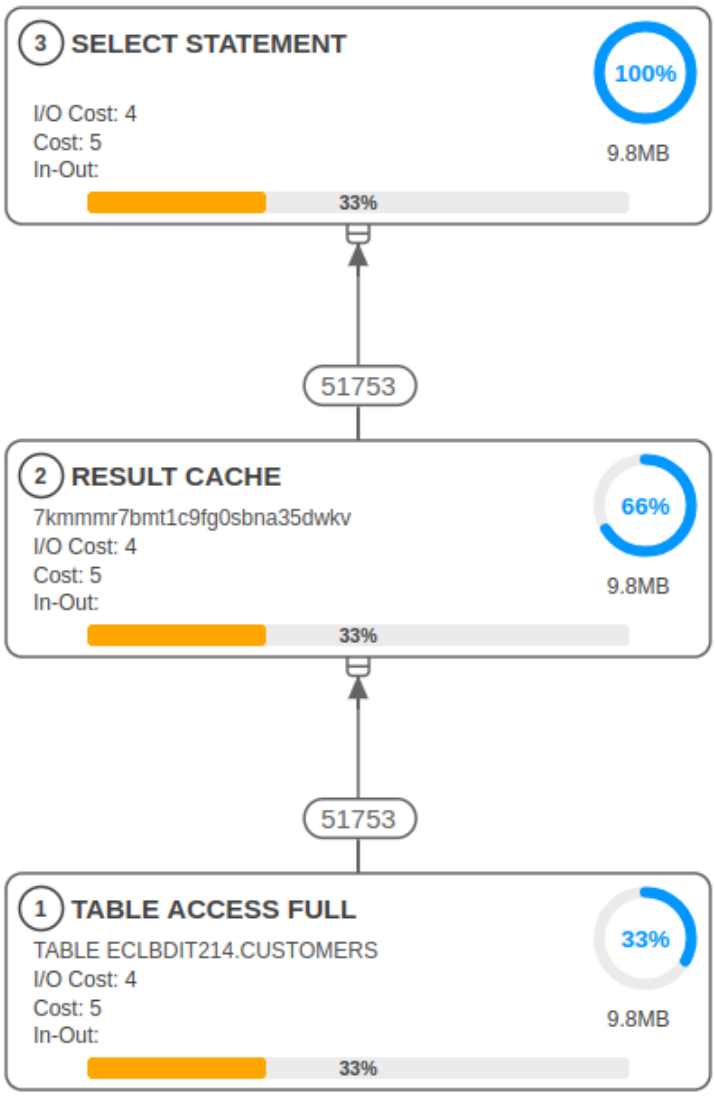
```
-- consulta 02

select * from customers where cust_id=4090;          -- gerar e analisar o plano:
consulta com igualdade
```



A explicação do plano e a seletividade da consulta são as mesmas da consulta anterior.

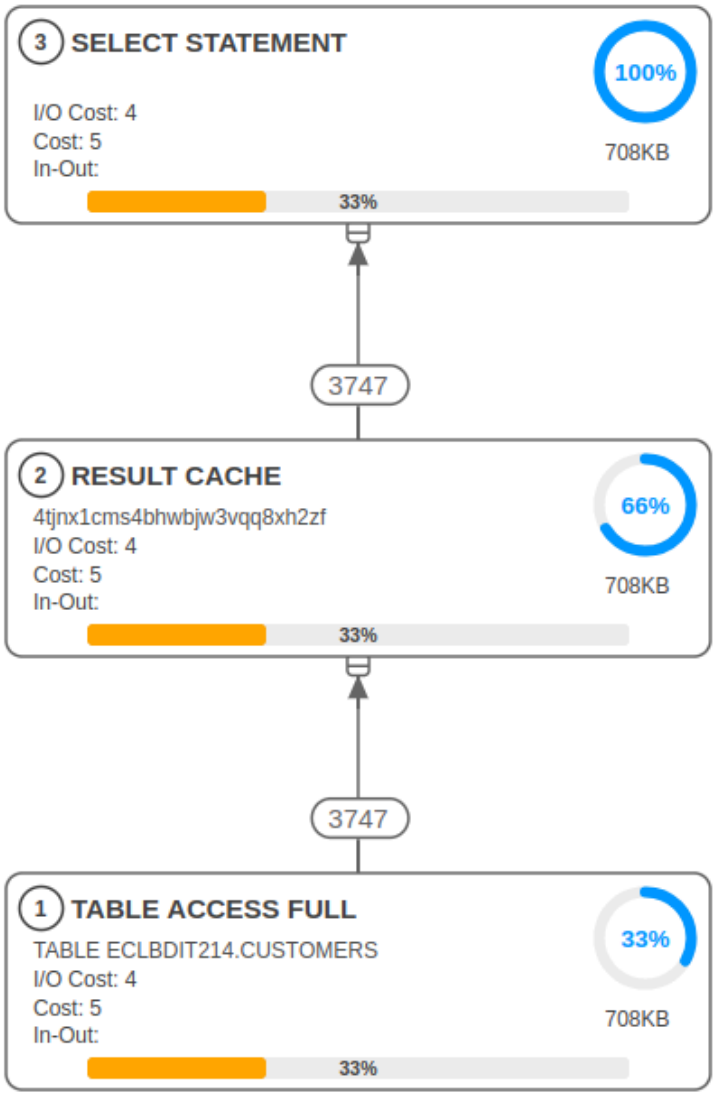
Consulta 03

```
-- consulta 03

select * from customers where cust_id>4090;          -- gerar e analisar o plano:
consulta com desigualdace: +90% da linhas
```



Nesse plano de execução para cust_id > 4090, a consulta realiza um acesso completo à tabela customers, com um custo total de 5 e uma cardinalidade de 51753, indicando que a condição de busca retorna a maioria das linhas da tabela. O uso do cache de resultados representa 66% do custo total. A seletividade da consulta é alta (0,933), o que significa que quase todas as linhas são retornadas.

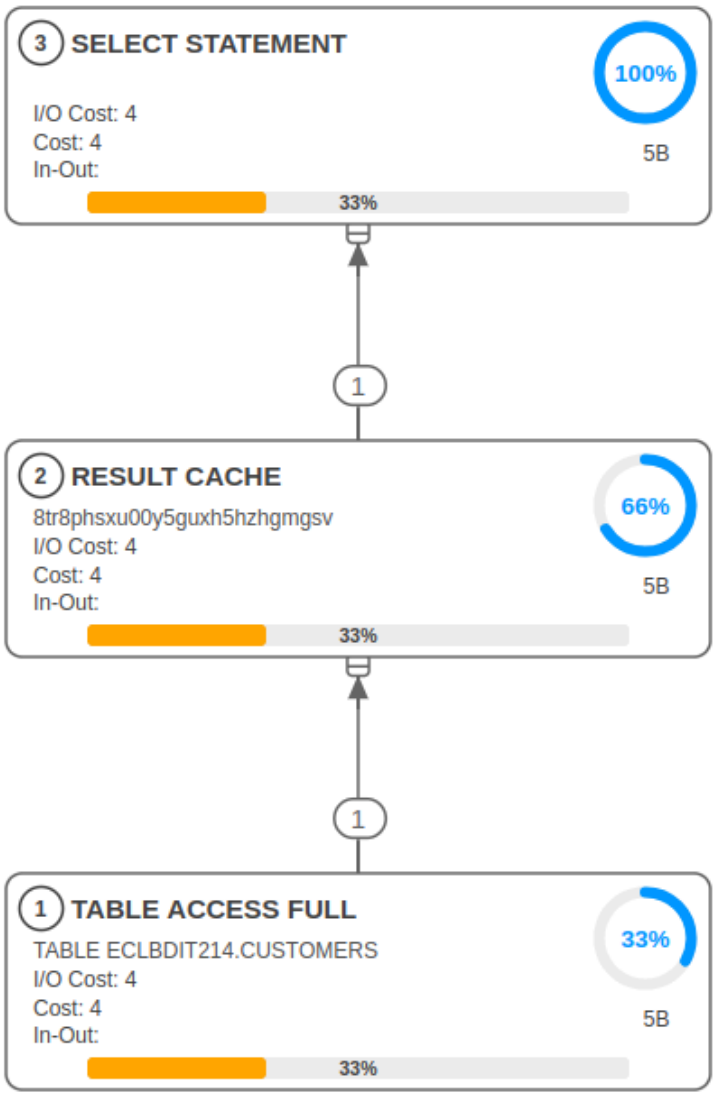
```
select * from customers where cust_id<4090;          -- gerar e analisar o plano:
consulta com desiguidade: -10% das linhas
```



A consulta no qual cust_id < 4090 realiza um acesso completo à tabela customers, com um custo total de 5 e uma cardinalidade de 3747, indicando que a condição de busca retorna uma pequena parte das linhas da tabela. O uso da cache de resultados representa 66% do custo total. A seletividade da consulta é baixa, pois cerca de 6.75% das linhas são retornadas.

Consulta 04

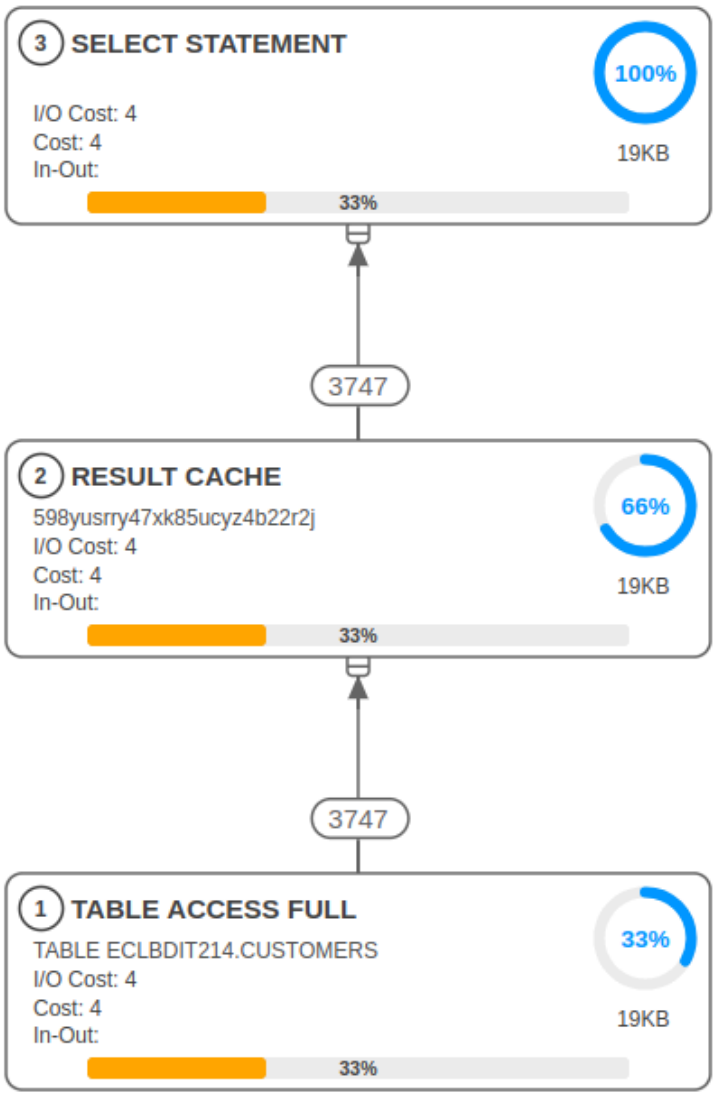
```
-- consulta 04

select cust_id from customers where cust_id=4090;    -- gerar e analisar o plano
```



A consulta realiza um acesso direto na coluna cust_id da tabela, com um custo total de 4 e uma cardinalidade de 1, indicando que apenas uma linha satisfaz a condição de busca. O uso do cache de resultados representa 66% do custo total. O valor "In-Out" é muito baixo devido à alta seletividade da consulta, que retorna apenas uma linha e solicita apenas a coluna cust_id.

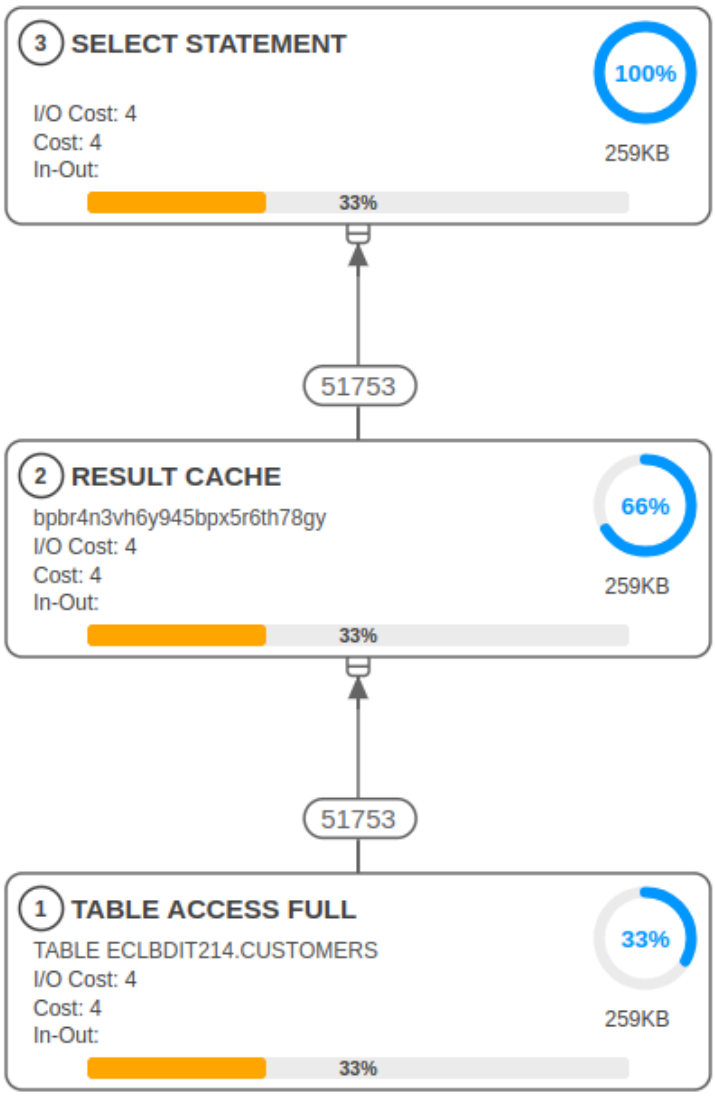
Consulta 05

```
-- consulta 05

select cust_id from customers where cust_id<4090;   -- gerar e analisar o plano
```



A consulta realiza um acesso à tabela customers com um custo total de 4 e uma cardinalidade de 3747, retornando uma parte significativa das linhas da tabela. O valor "In-Out" é relativamente baixo, refletindo a quantidade total de dados lidos, já que a consulta acessa apenas a coluna cust_id.

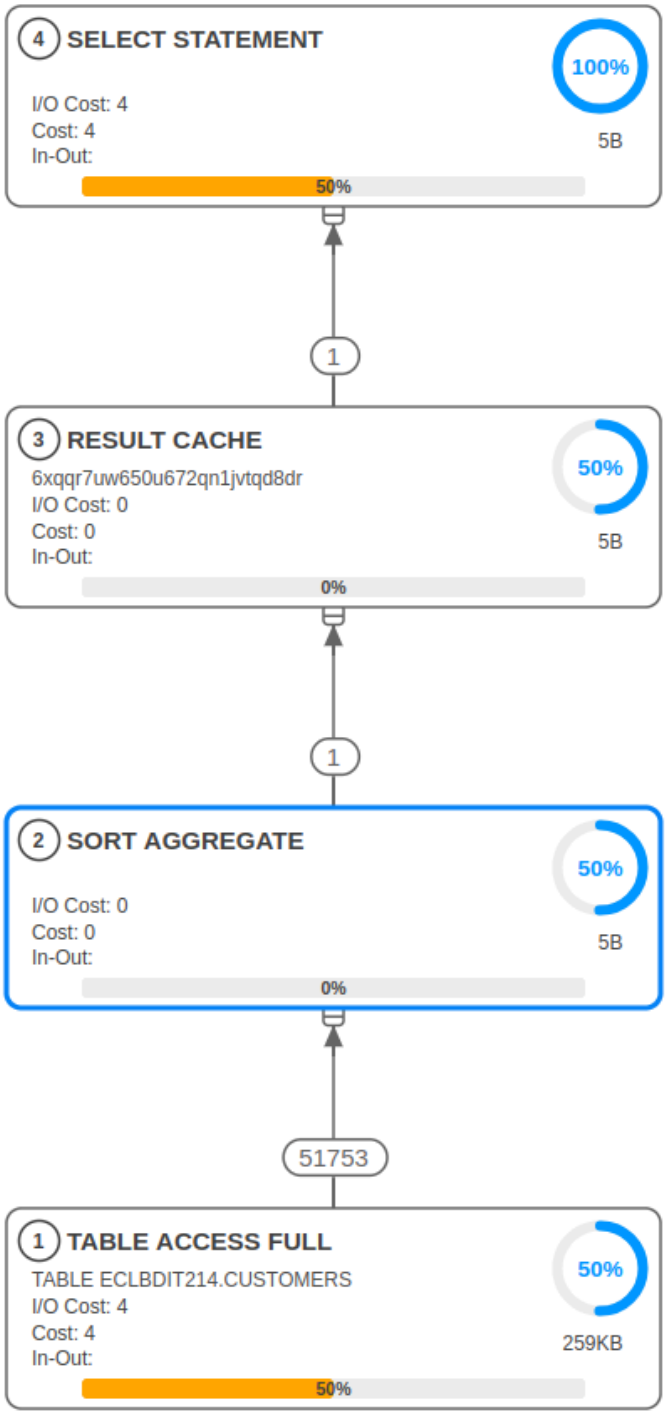
```
select cust_id from customers where cust_id>4090;   -- gerar e analisar o plano
```



A consulta realiza um acesso à tabela customers com um custo total de 4 e uma cardinalidade de 51753, retornando uma grande quantidade de linhas. O valor "In-Out" é de 259KB, refletindo a quantidade total de dados transferidos, pois a consulta seleciona apenas a coluna cust_id. Em contraste, a consulta [select * from customers where cust_id > 4090] acessa toda a tabela, mas com um "In-Out" muito maior (9.8MB), já que seleciona todas as colunas da tabela. A diferença no "In-Out" destaca como a seleção de menos colunas reduz a quantidade de dados manipulados, melhorando a eficiência da consulta, mesmo quando o custo de execução é o mesmo.

Consulta 06

```
-- consulta 06

select count(*) from customers where cust_id>4090;  -- gerar e analisar o plano
```

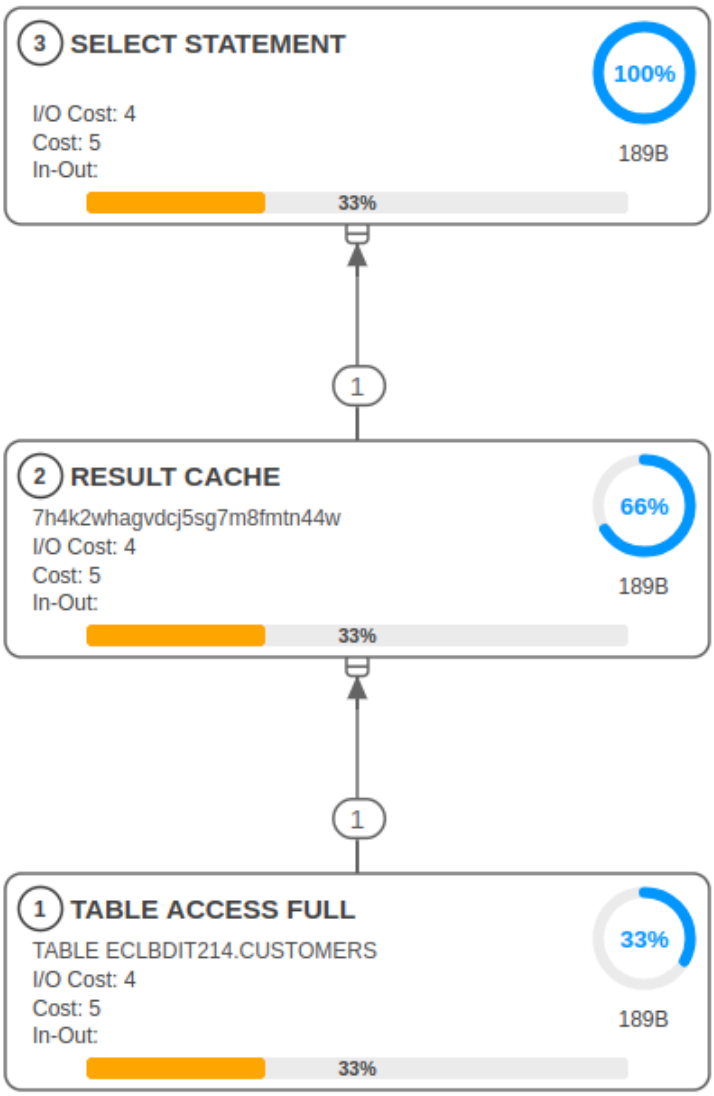


A consulta realiza um acesso à tabela com um custo total de 4 e uma cardinalidade de 51753, indicando que a condição de busca corresponde a essa quantidade de linhas. O valor "In-Out" é de 259KB, refletindo a quantidade de dados lidos, pois a consulta precisa acessar todas as linhas relevantes para contar os registros. A etapa SORT AGGREGATE calcula a contagem final a partir dos dados lidos, resultando em um "In-Out" muito baixo (5B). O uso da cache de resultados também é significativo (50% do custo total), já que os resultados intermediários podem ser armazenados e reutilizados. A diferença principal em relação a consultas que retornam dados detalhados é que aqui a consulta realiza uma contagem eficiente

sem necessidade de transferir grandes volumes de dados, refletindo no custo reduzido para operações de agregação e cache.

Consulta 11

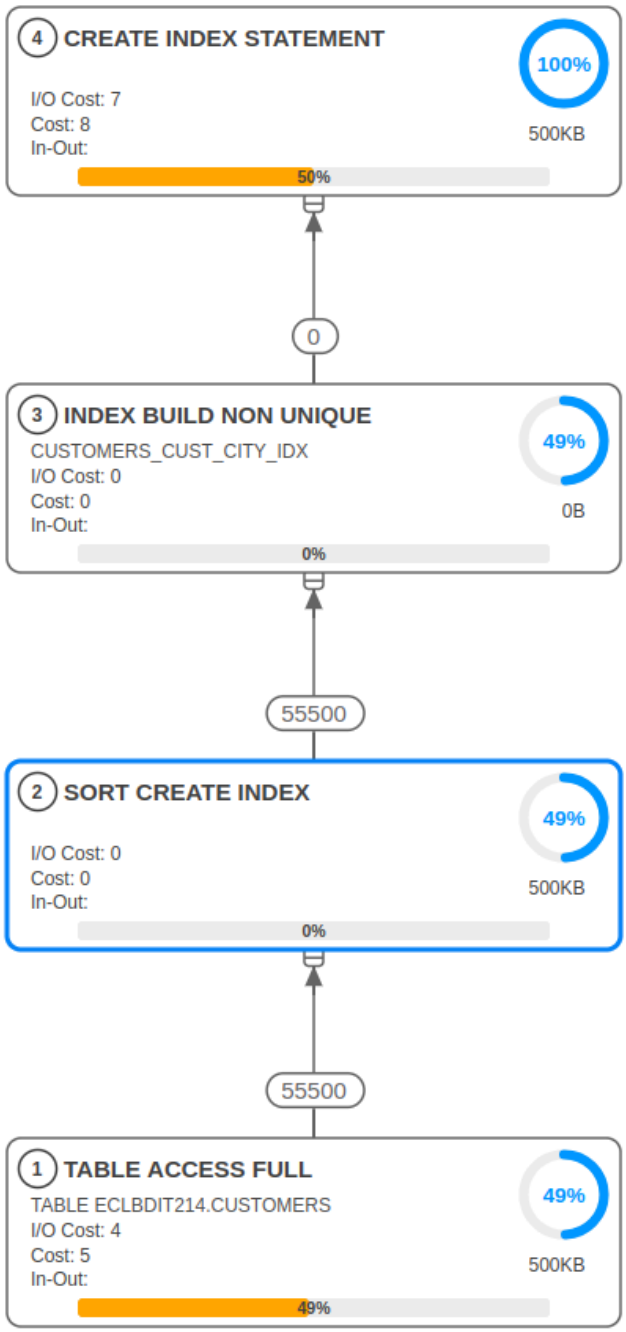
```
-- consulta 11

select * from CUSTOMERS where CUST_CITY='Tokyo'; -- gerar e analisar o plano
```



A consulta realiza um acesso à tabela com um custo total de 5 e uma cardinalidade de 1, indicando que a condição de busca corresponde a uma única linha. O valor "In-Out" é de 189B, refletindo a quantidade total de dados lidos, que é relativamente baixa porque apenas uma linha é retornada e todos os dados desta linha são transferidos. O uso da cache de resultados também é significativo (66% do custo total), ajudando a armazenar e reutilizar resultados intermediários. Apesar da

consulta acessar toda a tabela, a baixa cardinalidade e o uso eficiente da cache contribuem para a eficiência na execução.

```
create index customers_cust_city_idx on customers(cust_city); -- gerar e analisar o plano
```

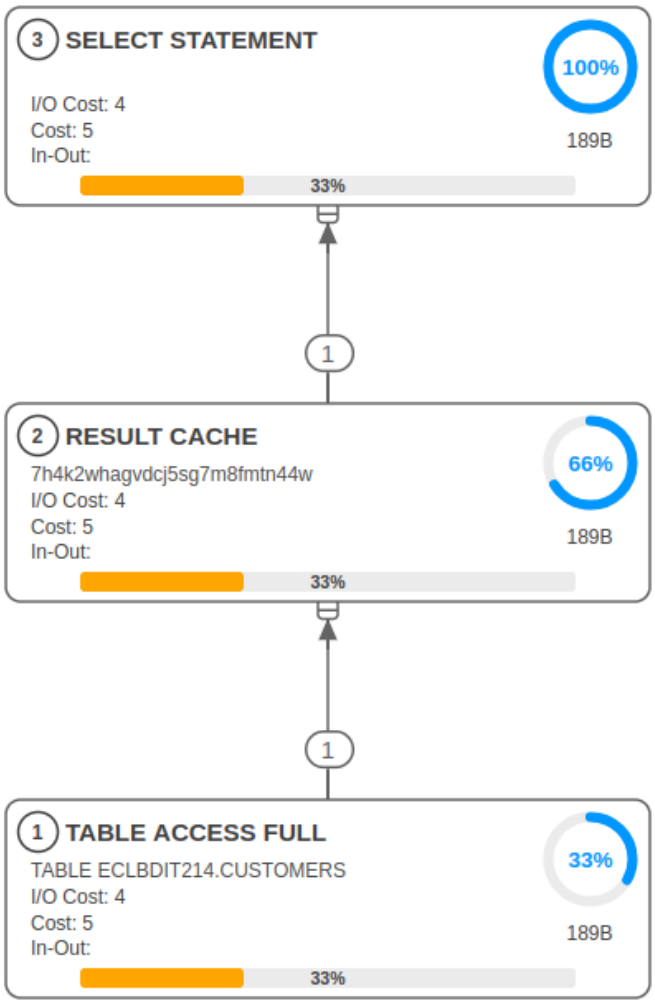


O comando cria um índice na coluna cust_city da tabela customers. O plano de execução mostra que a criação do índice envolve várias etapas. Primeiro, há um TABLE ACCESS FULL com um custo de 5, onde a tabela é lida completamente para coletar os dados necessários para o índice, resultando em um "In-Out" de 500KB e uma cardinalidade de 55500. Em seguida, a etapa SORT CREATE INDEX

organiza os dados para a construção do índice, também com um "In-Out" de 500KB. A etapa INDEX BUILD NON UNIQUE efetivamente constrói o índice, mas não requer mais I/O, refletindo um "In-Out" de 0B. Finalmente, a etapa CREATE INDEX STATEMENT completa a criação do índice com um custo total de 8 e um "In-Out" de 500KB, que representa o total de dados processados.

Consulta 13

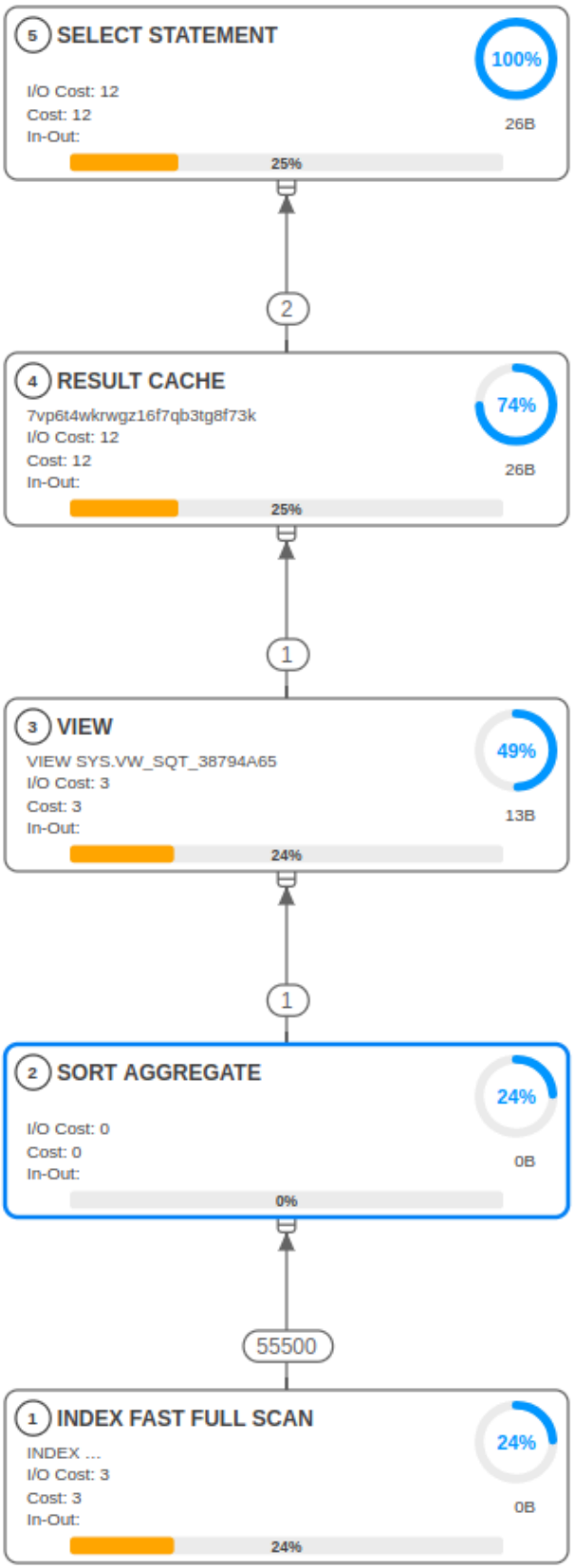
```
-- consulta 13

select * from CUSTOMERS where CUST_CITY='Tokyo'; -- gerar e analisar o plano: seletiva
```



A consulta realiza um acesso à tabela customers com um custo total de 5 e uma cardinalidade de 1, indicando que apenas uma linha corresponde ao critério de busca. O uso da cache de resultados representa 66% do custo total, ajudando a armazenar e reutilizar os dados lidos para melhorar a eficiência. O baixo "In-Out" e a

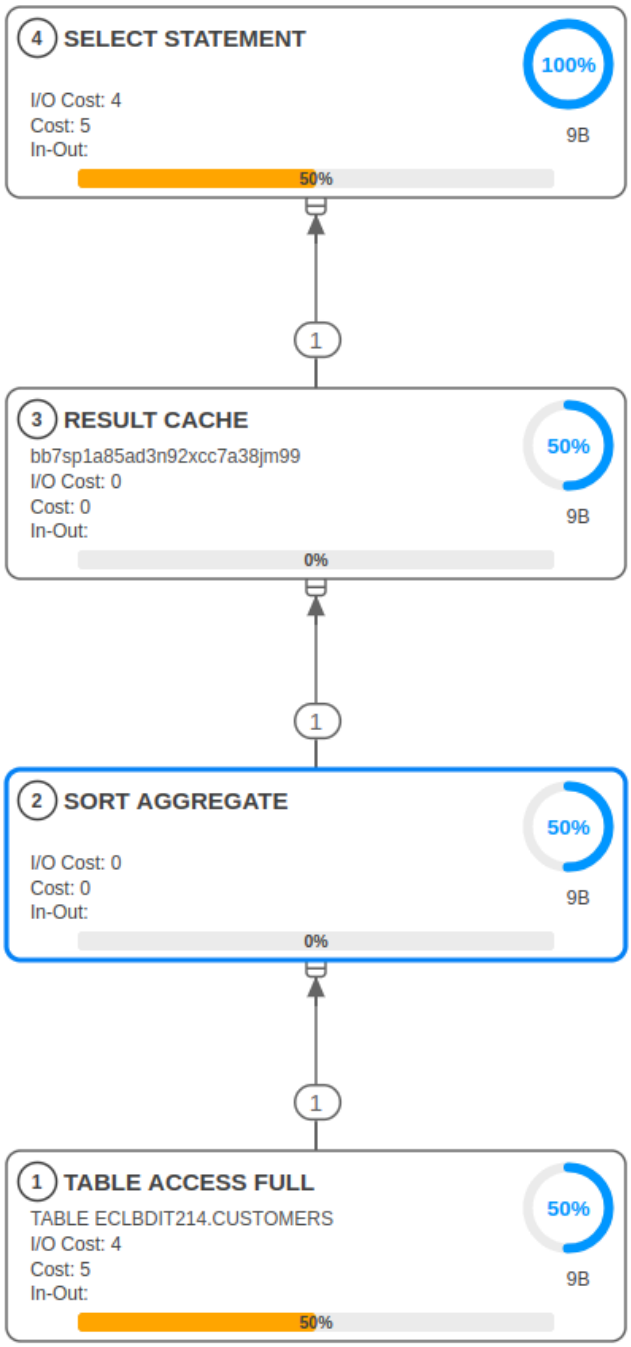
baixa cardinalidade contribuem para uma execução eficiente da consulta, embora o acesso completo à tabela ainda seja necessário para encontrar a linha específica.

```
-- cardinalidade s, s=16
select count(*) from customers;
```



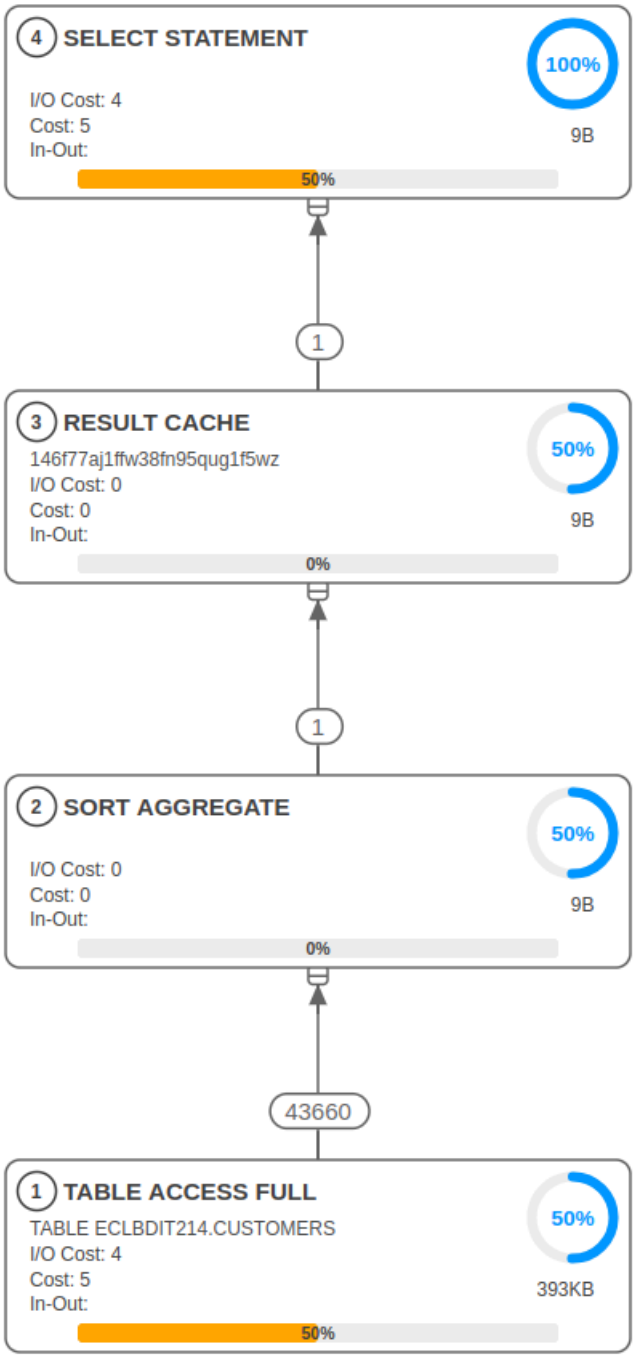
A consulta utiliza um plano de execução que começa com um INDEX FAST FULL SCAN com um custo de 3, onde o índice é lido completamente sem precisar acessar os dados da tabela, resultando em um "In-Out" de 0B, já que apenas a estrutura do índice é examinada. A cardinalidade nesta etapa é de 55500, representando o total de linhas na tabela. Em seguida, a etapa SORT AGGREGATE realiza a contagem das linhas, com um custo de 0 e sem leitura adicional de dados ("In-Out" de 0B). A etapa VIEW mostra o resultado da contagem, com um custo de 3. O uso da cache de resultados representa 74% do custo total, ajudando a armazenar e reutilizar o resultado final com um "In-Out" de 26B. No total, a execução tem um custo de 12, refletindo a eficiência ao utilizar o índice para contar o número de linhas, minimizando o acesso direto aos dados da tabela.

```
-- quantidade de linhas r=55500

-- seletividade sl, sl=s/r, 16/55500=0,00288%

select count(*) from CUSTOMERS where cust_city='Tokyo';   -- gerar e analisar o plano
```



A consulta acessa completamente a tabela customers com um custo total de 5 e uma cardinalidade de 1, indicando que a condição de busca corresponde a apenas uma linha. A seletividade é extremamente baixa (0,00288%), refletindo que apenas uma pequena fração das linhas da tabela atende ao critério. O uso da cache de resultados representa 50% do custo total, indicando que os dados intermediários são armazenados para melhorar a eficiência. Apesar do acesso completo à tabela, a baixa cardinalidade e o uso eficiente da cache ajudam a manter o custo total relativamente baixo.

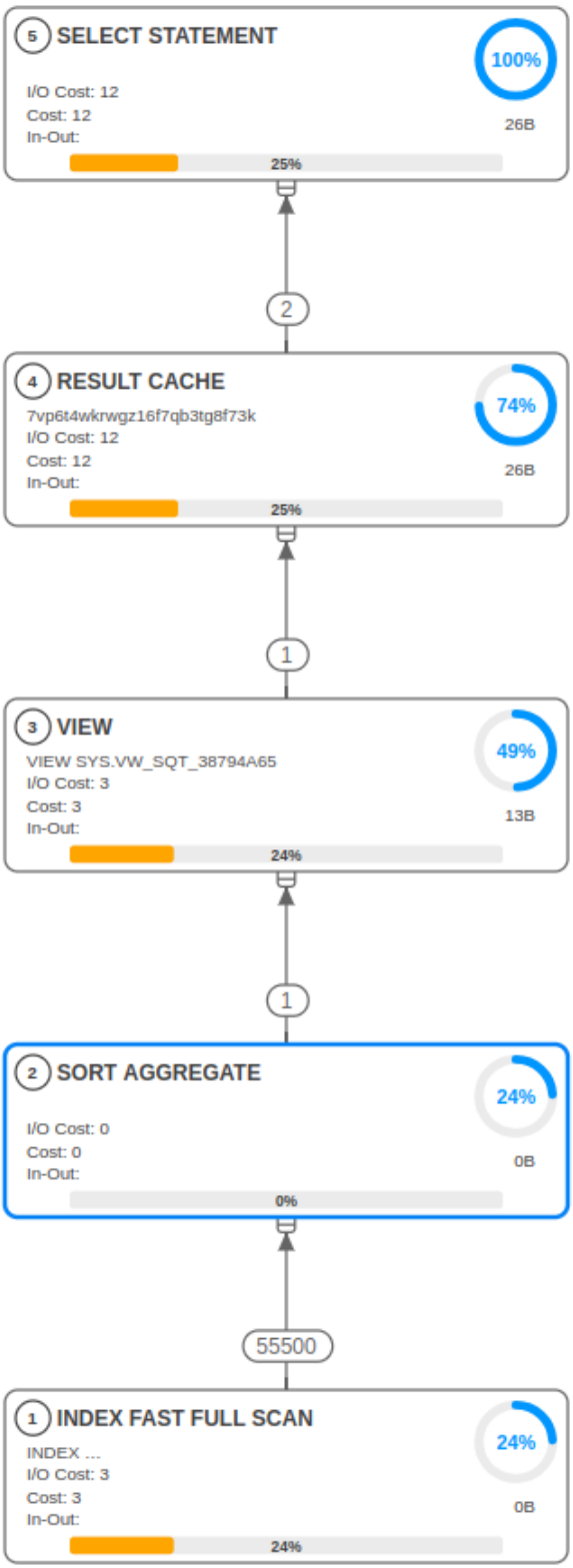
```
-- Selectividade para select * from customers where cust_city = 'Los Angeles';

select count(*) from customers where cust_city = 'Los Angeles';
```



A consulta realiza um acesso à tabela com um custo total de 5 e uma cardinalidade de 43660. O valor "In-Out" é de 393KB, refletindo a quantidade total de dados lidos durante a operação, que é relativamente alta devido ao número significativo de linhas correspondentes. A etapa SORT AGGREGATE calcula a contagem final das linhas, com um "In-Out" reduzido para 9B, pois apenas o resultado da contagem é

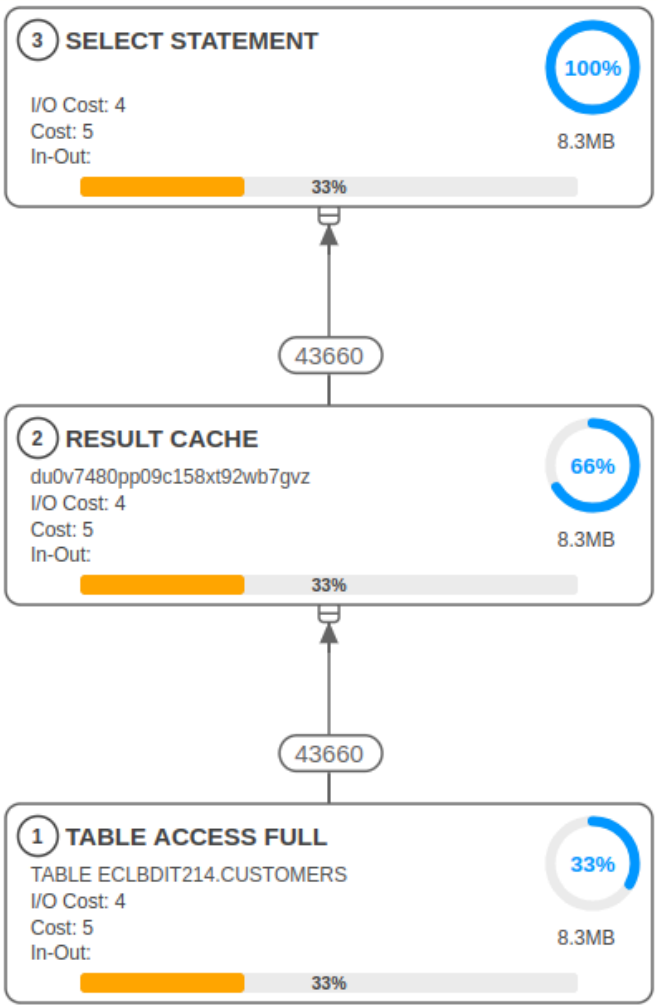
processado. O uso da cache de resultados representa 50% do custo total, ajudando a armazenar e reutilizar os dados intermediários.

```
-- cardinalidade s, s=932

select count(*) from customers;
```



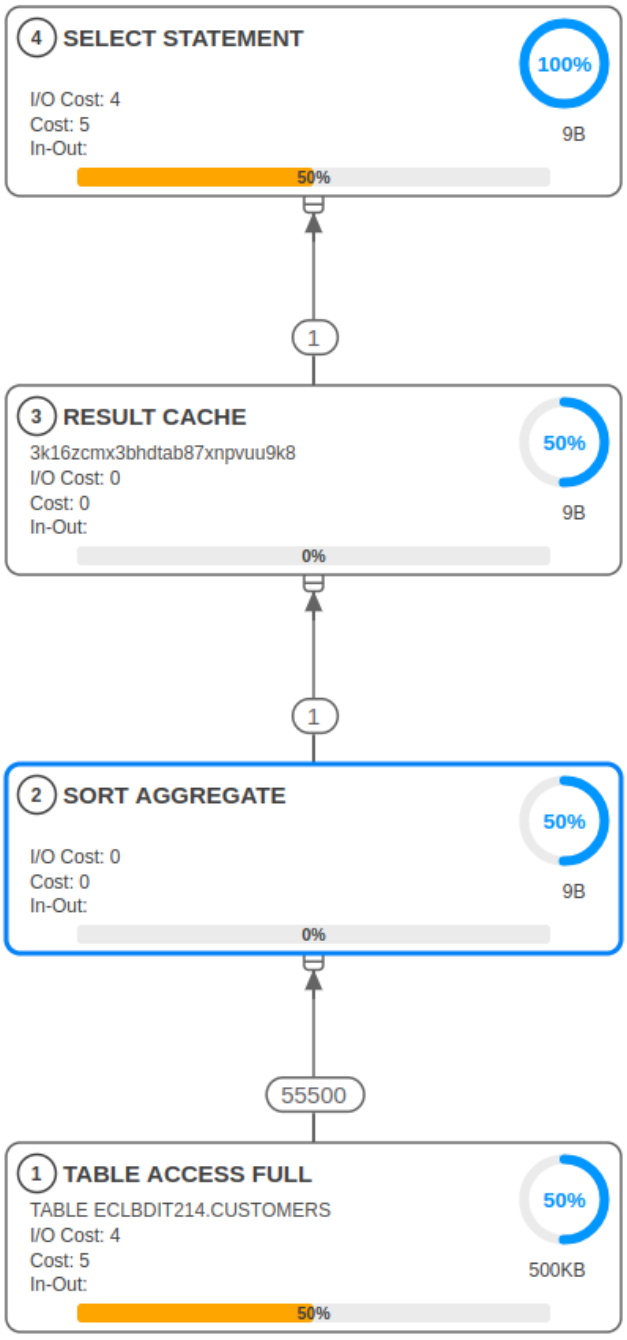
A consulta utiliza um plano de execução que começa com um INDEX FAST FULL SCAN com um custo de 3 e uma cardinalidade de 55500, refletindo o total de linhas na tabela. O valor "In-Out" é 0B, indicando que o índice é lido sem necessidade de acessar diretamente os dados da tabela. A etapa SORT AGGREGATE calcula a contagem, sem custo adicional de I/O e com "In-Out" também de 0B. A etapa VIEW mostra o resultado da contagem, com um custo de 3 e um "In-Out" de 13B. O uso da cache de resultados é significativo, representando 74% do custo total, com "In-Out" de 26B, e contribui para a eficiência ao armazenar e reutilizar o resultado final. Apesar do custo de I/O total ser relativamente alto (12), a execução é otimizada pela utilização do índice para contar as linhas, reduzindo a necessidade de leitura direta de dados.

```
-- quantidade de linhas r=55500

select * from CUSTOMERS where cust_city='Los Angeles';
```



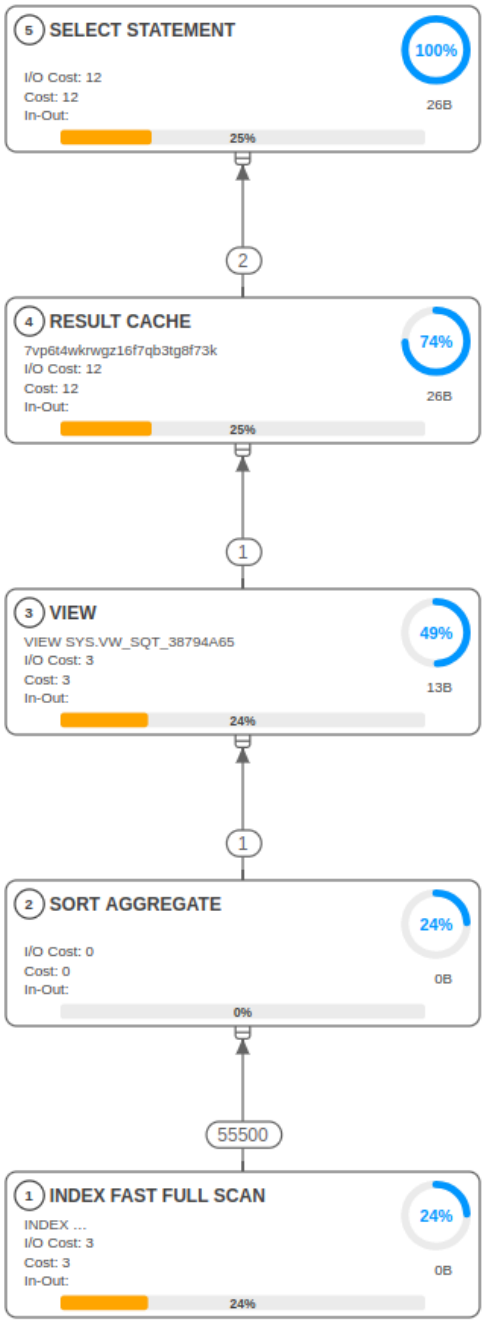
A consulta realiza um acesso completo à tabela customers com um custo total de 5 e uma cardinalidade de 43660, indicando que essa é a quantidade de linhas que correspondem ao critério de busca. A etapa RESULT CACHE representa 66% do custo total e armazena em cache os dados lidos, ajudando a melhorar a eficiência ao reutilizar esses dados em vez de recalcular. Embora a consulta tenha um custo de I/O e processamento considerável devido ao grande volume de dados, o uso da cache contribui para a eficiência da execução.

```
select count(*) from customers where cust_city >= 'Los Angeles';
```



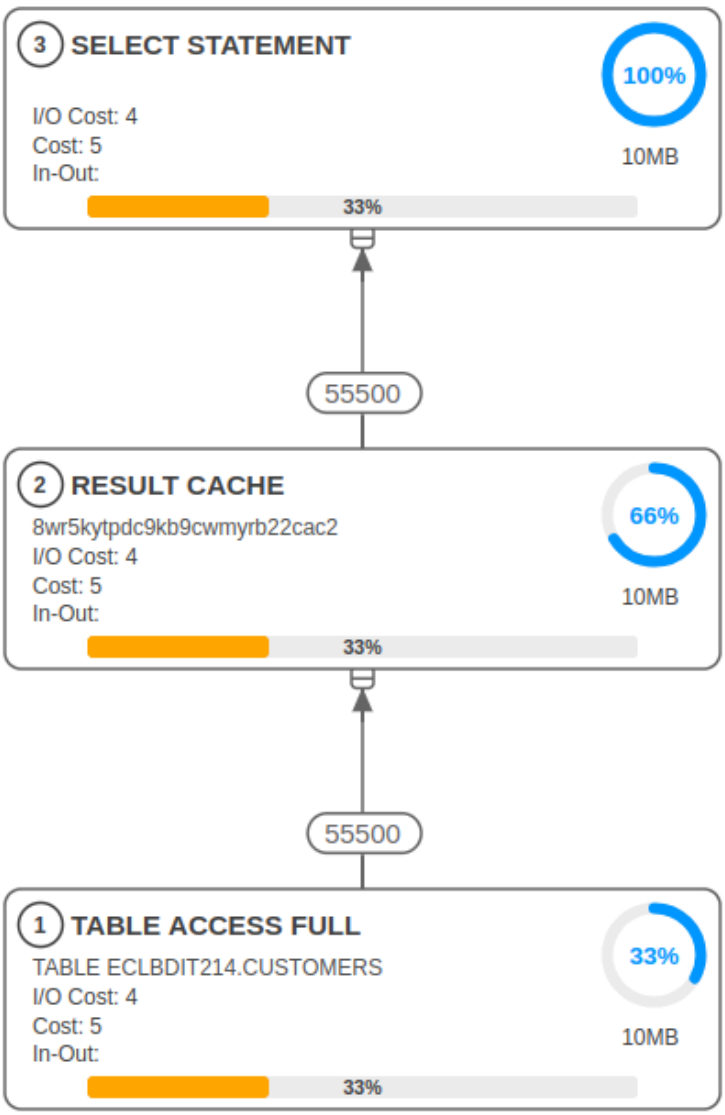
A consulta executa um acesso completo à tabela customers com um custo total de 5 e uma cardinalidade de 55500, indicando que a condição de busca corresponde a todas as linhas da tabela, já que a comparação com >= 'Los Angeles' inclui muitas entradas. A etapa SORT AGGREGATE calcula a contagem final das linhas, com um "In-Out" reduzido para 9B, já que apenas o resultado da contagem é processado. O uso da cache de resultados é significativo, representando 50% do custo total, e armazena o resultado final para melhorar a eficiência.

```
-- cardinalidade s, s=25304

select count(*) from customers;
```



A consulta usa um plano de execução que começa com um INDEX FAST FULL SCAN com um custo de 3, onde o índice é examinado para obter a contagem das 55500 linhas na tabela, sem acessar diretamente os dados da tabela (In-Out = 0B). A etapa SORT AGGREGATE calcula a contagem final com um custo de 0 e sem leitura adicional de dados. A etapa VIEW mostra a contagem resultante, com um custo de 3 e um "In-Out" de 13B. O uso da cache de resultados é significativo, representando 74% do custo total e armazenando o resultado final com um "In-Out" de 26B.

```
-- quantidade de linhas r=55500

-- seletividade da consulta - sl=25304/55500, aprox 50%

select * from CUSTOMERS where cust_city>='Los Angeles';
```



A consulta realiza um acesso completo à tabela customers com um custo total de 5 e uma cardinalidade de 55500. A seletividade é cerca de 50%, refletindo que aproximadamente metade das linhas atendem ao critério. O uso do RESULT CACHE representa 66% do custo total, ajudando a armazenar e reutilizar os dados lidos para melhorar a eficiência. Apesar do custo de I/O total ser elevado, o uso do cache contribui para otimizar a execução ao evitar leituras repetidas de dados.

Consulta 15

```
-- consulta 15

select cs.cust_id, cs.cust_first_name, cs.cust_last_name, cs.cust_city , ct.country_name from customers cs join countries ct on ct.country_id=cs.country_id where ct.country_name='Canada';
```

A consulta utiliza um plano de execução que inicia com um INDEX RANGE SCAN para encontrar as entradas na tabela countries com o nome de país 'Canada', com um custo baixo de 1 e sem leitura de dados (In-Out = 0B). Em seguida, a consulta

realiza um SORT CLUSTER BY ROWID para organizar os dados encontrados e um TABLE ACCESS BY INDEX ROWID para acessar diretamente as linhas da tabela customers associadas ao índice, com um custo de 2 e leitura de 15B. O acesso completo à tabela customers ocorre em 20% do custo, lendo 1.9MB de dados para encontrar todas as linhas correspondentes. A etapa de HASH JOIN combina os resultados das duas tabelas, com um custo de 7 e leitura de 118KB. O uso do RESULT CACHE representa 73% do custo total, ajudando a armazenar e reutilizar os dados resultantes da junção.

Consulta 16

```
-- consulta 16

select cs.cust_id, cs.cust_first_name, cs.cust_last_name, cs.cust_city , ct.country_name from customers cs join countries ct on ct.country_id=cs.country_id where ct.country_name='Canada';
```

A consulta segue um plano de execução que começa com um INDEX RANGE SCAN na tabela countries para localizar as linhas com o nome do país 'Canada', com um custo baixo e sem leitura de dados. Em seguida, um SORT CLUSTER BY

ROWID organiza os dados encontrados, e um TABLE ACCESS BY INDEX ROWID é realizado para acessar as linhas correspondentes na tabela customers, com um custo de 2 e leitura de 15B. A HASH JOIN combina os resultados das tabelas customers e countries, com um custo de 7 e leitura de 118KB. O uso do RESULT CACHE é significativo, armazenando e reutilizando esses dados para melhorar a eficiência.
