**RELATÓRIO DE ANALISE**

**TUNNING E PERFORMANCE.**

**CONSULTA ANALISADA – QUERY2.SQL**

|  |
| --- |
|  |

Execução inicial em 662 msec para 17 linhas afetadas;

Analisando o plano de execução:

|  |
| --- |
|  |

**LAUDO -** Essa consulta traz mais complexidade, pois pede os dados do intervalo de todas as apostas feitas ontem no filtro.

* No plano, podemos ver o **PARALLEL SEQ SCAN** na tabela **[BET]**. Neste caso a consulta está lendo todos os registros da tabela.
* No filtro não há utilização de um índice
* Vemos também que o item **ROWS REMOVED BY FILTER** trouxe **1.319.815** de registros **NÃO UTILIZADOS**. Isso indica que o banco varreu mais dados do que necessário e utilizou 13.518 registros uteis.
* No resultado parcial é feito o merge (**GATHER MERGE**) e então novamente os dados são agregados (**FINALIZE GROUPAGGREGATE**) e montando o resultado final.
* A função **DATE(PLACED\_AT)** prejudica muito em um índice normal

**SUGESTÃO DE MELHORIAS**

Com essas informações podemos sugerir a criação de um índice mais funcional.

**CREATE INDEX IDX\_BETS\_PLACED\_AT\_DAY ON BETS (PLACED\_AT)   
 WHERE PLACED\_AT >= CURRENT\_DATE - INTERVAL '7 DAY';**

Porém como o indice tem uma data **IMUTAVEL**, sugiro criar com um range de data fixo e utilizar o **CRON** para agendar um DROP INDEX e CREATE INDEX com a data de corte.

**CREATE INDEX IDX\_BETS\_PLACED\_AT\_DAY ON BETS (PLACED\_AT)   
 WHERE PLACED\_AT >= '2025-10-15';**

Há uma sugestão também de mudarmos a consulta usando **CTE** (**COMMON TABLE EXPRESSION**).

|  |
| --- |
|  |

Com essa alteração, evitamos a utilização da função **DATE()**, no qual não conseguimos usar um índice eficiente no campo **[PLACED\_AT]**.

O índice proposto ajuda a filtrar no intervalo do último dia de forma mais seletiva e assim evitando o **PARALLEL SEQ SCAN (TABLE SCAN)**.

Neste cenário, também indico o particionamento da tabela **[BETS]**. Caso haja a real oportunidade do particionamento, a estrutura que podemos seguir seria:

|  |
| --- |
|  |

**APÓS A APLICAÇÃO DAS SUGESTÕES**

Com a aplicação das sugestões, o plano de execução trouxe:

|  |
| --- |
|  |

* Tempo total diminuiu para aproximadamente **191 ms**, comparado aos **2.000 ms** anteriores, mostrando um ganho superior a **10 VEZES** no desempenho.
* O uso do **BITMAP INDEX SCAN** ainda ocorre no índice simples **IDX\_BETS\_PLACED\_AT**, mostrando que para o filtro temporal o banco continua aproveitando o índice **BTREE** sobre **PLACED\_AT**.
* O **BITMAP HEAP SCAN** continua acessando a mesma quantidade de dados, mas o tempo dessa etapa diminuiu muito, indicando melhor eficiência na leitura.
* A etapa de ordenar **(SORT)** por **STATUS** e **(PLACED\_AT)::DATE** que anteriormente consumia quase **2 segundos**, foi reduzida para **182 ms**, o que é um enorme ganho.
* A agregação (**GROUPAGGREGATE**) também ficou bem rápida, encerrando próximo a **8 ms** após a ordenação.