11 - Oracle GoldenGate 26ai

Otimização de Performance, Divisão por Ranges e Tuning de I/O

Técnicas avançadas para ambientes de produção críticos com grandes volumes de dados — bancos, telecoms e e-commerces.

Objetivos da Aula

Ao final desta aula, você dominará as técnicas avançadas de otimização para grandes volumes no Oracle GoldenGate 26ai.

1	2	3
Divisão por @RANGE Fatiar uma única tabela massiva em múltiplos processos paralelos usando a função `@RANGE`, distribuindo a carga de forma segura e escalável.	**Tuning de I/O e Memória** Ajustar parâmetros críticos de I/O, checkpoint e o componente `CACHEMGR` para eliminar paginação em disco e gargalos de escrita sequencial.	**Tuning Fino do Replicat** Eliminar Full Table Scans com `KEYCOLS`, mitigar transações de barreira e proteger o ambiente com `TRANSACTIONTIMEOUT`.

O Problema: A "Tabela Monstro"

Em ambientes de produção críticos, é comum encontrar uma tabela central com **bilhões de registros** recebendo milhões de atualizações por minuto. Mesmo com múltiplos grupos por esquema, essa tabela única se torna um gargalo intransponível, gerando latência que afeta diretamente o negócio.

Sintomas no ambiente

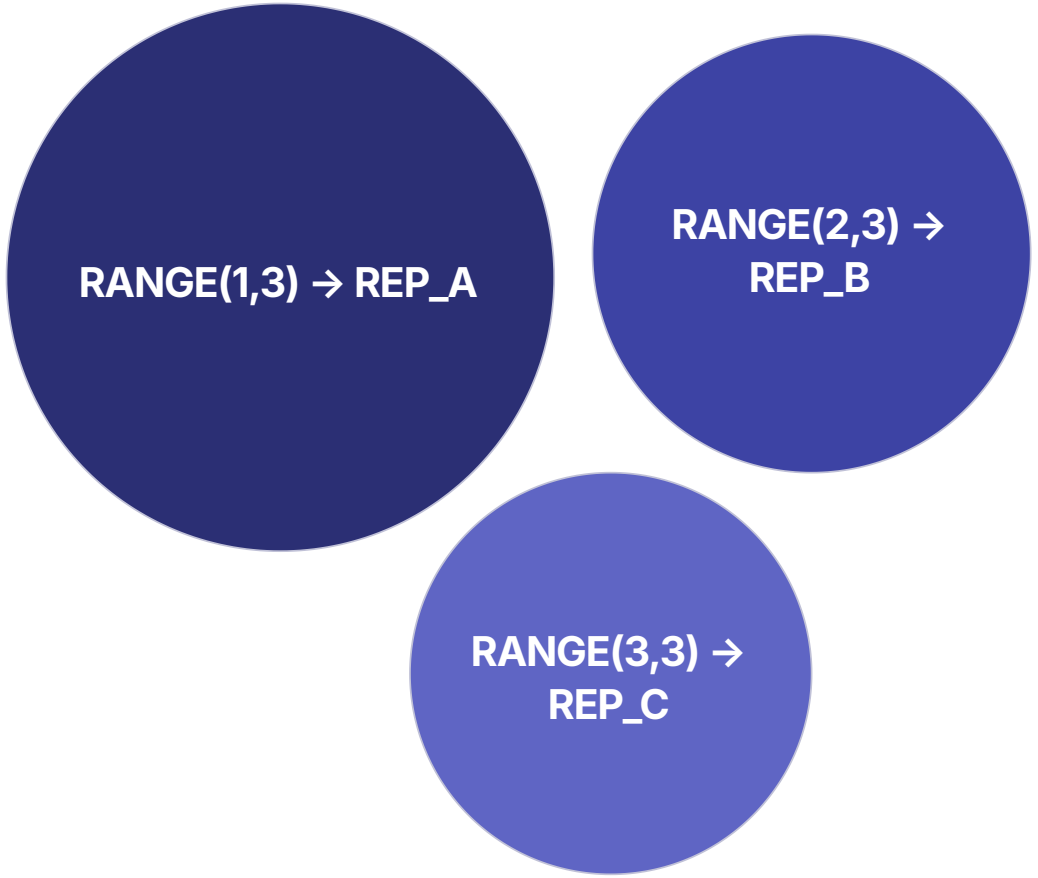
- Lag crescente e não recuperado no Extract/Replicat
- Alertas de performance na monitoração da madrugada
- Subsistema de disco saturado mesmo com hardware moderno

Nossa responsabilidade como DBAs

Não podemos simplesmente afirmar "essa tabela é grande demais". Precisamos fatiar a tabela para que o hardware trabalhe em paralelo e dimensionar corretamente a memória virtual e o subsistema de I/O.

Conceito: Como o @RANGE Funciona

A função `@RANGE` aplica um algoritmo de **hash matemático distribuído** sobre a chave primária. Cada linha é roteada deterministicamente para exatamente um processo Replicat — sem duplicatas, sem conflitos, preservando a ordem cronológica.



A lógica é simples: se o resto da divisão da chave por N for igual ao índice da fatia, a linha pertence àquele processo. Uma linha específica **sempre cairá no mesmo Replicat**, garantindo integridade transacional e paralelismo seguro.

Implementação: Divisão por Range no Replicat

Para dividir a carga da tabela `CRM.CLIENTES` entre dois processos paralelos, especifique `@RANGE` dentro da cláusula `FILTER` do comando `MAP`. O primeiro argumento é a fatia atual e o segundo é o total de fatias.

rep_fa_1.prm — Fatia 1

```
REPLICAT rep_fa_1
USERIDALIAS ogg_target_profile
MAP crm.clientes,
  TARGET crm.clientes,
  FILTER (@RANGE(1, 2, id_cliente));
```

rep_fa_2.prm — Fatia 2

```
REPLICAT rep_fa_2
USERIDALIAS ogg_target_profile
MAP crm.clientes,
  TARGET crm.clientes,
  FILTER (@RANGE(2, 2, id_cliente));
```

Para escalar além de 2 processos, basta criar novos grupos Replicat incrementando o primeiro argumento. O segundo argumento (total de fatias) deve ser atualizado em **todos** os arquivos de parâmetro simultaneamente para evitar sobreposição de dados.

Eliminando Gargalos de I/O e Disco

O GoldenGate grava dados **sequencialmente** nos arquivos de trail. A arquitetura do subsistema de disco impacta diretamente a vazão do Extract e do Replicat.

Arquitetura de Disco

Aloque os trail files nos controladores mais rápidos disponíveis. Prefira **RAID 0+1** (striping + mirroring) — o RAID 5 calcula paridade em cada escrita, degradando seriamente o I/O sequencial do OGG.

CHECKPOINTSECS

Controla a frequência de gravação dos checkpoints de recuperação. Aumentar o intervalo reduz o I/O, mas amplia o reprocessamento em caso de abend. **Não se aplica ao Integrated Replicat.**

GROUPTRANSOPS

Agrupa múltiplas pequenas transações em uma única transação maior no destino (padrão: 250 ops). Elevar para **1000–2000** reduz drasticamente a frequência de commits e o I/O no banco alvo.

Gerenciamento de Memória Virtual: CACHEMGR

O Cache Manager é o componente responsável por manter em memória as transações abertas (não commitadas). Com transações gigantescas, a RAM física pode estourar e o sistema operacional passa a **paginar em disco (Swap)**, destruindo a performance.

Alerta clássico no log do OGG 26ai:

```
2026-06-24 14:16:22 WARNING OGG-01842
CACHESIZE PER DYNAMIC DETERMINATION (32G)
LESS THAN RECOMMENDED: 64G. Check swap space.
```

Solução — adicionar ao parâmetro do Extract ou Replicat:

```
CACHEMGR CACHESIZE 32G,
    CACHEDIRECTORY /u02/ogg_fast_cache
```

⚠ O diretório definido em `CACHEDIRECTORY` deve residir obrigatoriamente em um disco de alta velocidade (SSD/NVMe). Apontar para um volume lento anulará completamente o benefício do parâmetro.

Tuning Fino do Replicat no Banco Destino

Três técnicas essenciais que eliminam os principais vilões de performance no lado do banco de dados alvo.

Transações de Barreira

Updates em chaves primárias forçam o Coordinated Replicat a sincronizar e commitar **todas as threads ativas** antes de prosseguir. Solução: isole tabelas com muitos PK updates em um grupo Replicat exclusivo com poucas threads.

Full Table Scans com KEYCOLS

Tabelas sem PK ou índice único no destino forçam o OGG a usar todas as colunas no `WHERE`, causando Full Scans por linha. Solução: declare colunas altamente seletivas com `KEYCOLS`.

```
MAP hr.emp, TARGET hr.emp,
  KEYCOLS (FIRST_NAME,
LAST_NAME,
        DOB, ID_NO);
```

Locks Presos: TRANSACTIONTIMEOUT

Se o Extract sofrer abend no meio de uma transação longa, o Replicat pode ficar aguardando indefinidamente com locks abertos no banco alvo. Solução: defina um timeout automático.

```
TRANSACTIONTIMEOUT 5m
```

💡 Dica de Ouro: PCTFREE em Ambientes RAC

O problema em Oracle RAC

Em clusters Oracle RAC, a tabela de checkpoints do Replicat (`OGG.GGSCHKPT`) pode se tornar um ponto crítico de contenção entre os nós. Múltiplos Replicats atualizam essa tabela agressivamente a cada commit, causando **Buffer Busy Waits** e **GC Buffer Busy** no interconnect do cluster.

A solução técnica

Antes de iniciar os Replicats pela primeira vez, altere o atributo `PCTFREE` da tabela de checkpoint para **90**. Isso força o Oracle a armazenar pouquíssimas linhas por bloco, espalhando os registros por vários blocos e eliminando a contenção de interconnect.

```
ALTER TABLE ogg.ggschkpt
  PCTFREE 90;
```

⚠️ Atenção ao momento certo

Este ajuste **deve ser feito antes** de rodar os Replicats pela primeira vez. Se os processos já tiverem sido iniciados e a tabela já estiver populada, será necessário recriar a tabela ou executar um REORG completo nela.

Este é um dos ajustes mais negligenciados em implantações RAC — e um dos que mais causam chamados de emergência em produção.

Resumo: Checklist de Otimização

Aplique estas técnicas sistematicamente em ambientes com tabelas de alto volume para garantir performance e estabilidade em produção.

Divida tabelas massivas com @RANGE

Crie N grupos Replicat com `FILTER (@RANGE(i, N, chave_pk))` para paralelizar o processamento sem risco de duplicação.

Configure disco e I/O corretamente

Trail files em RAID 0+1, ajuste de `CHECKPOINTSECS` e aumento do `GROUPTRANSOPS` para 1000–2000 operações.

Dimensione o CACHEMGR

Defina `CACHESIZE` compatível com a RAM disponível e aponte `CACHEDIRECTORY` para um disco rápido (SSD/NVMe).

Aplique tuning fino no Replicat

Use `KEYCOLS` para evitar Full Scans, isole tabelas com PK updates, defina `TRANSACTIONTIMEOUT` e ajuste `PCTFREE 90` em RAC.

Estas otimizações são cumulativas. A máxima performance é obtida aplicando todas as camadas em conjunto — desde a infraestrutura de disco até o tuning de parâmetros individuais do Replicat.