Batch Checklist

☐ A configuração de dados sensíveis foi externalizada?

☐ O RunIdIncrementer foi removido? Só utilizar esse incrementador nos testes ou quando o restart não for necessário, pois ele impede a reinicialização do Job do ponto em que ele parou.

☐ O batch está logando informações importantes que facilitem o monitoramento e tratamento de erros? Em alguns casos é importante customizar o log, por exemplo, adicionando métricas de desempenho.

☐ A lógica do job não está concentrada em uma única classe? Dividir a lógica em diferentes classes melhora a manutenibilidade do código.

☐ O job possui testes unitários e de integração? Não é possível fazer mudanças rápidas e com segurança sem testes.

☐ Não repetiu lógicas comuns entre batches e sistemas online? Se aplica quando os sistemas dependem um do outro.

☐ O job está com a lógica mais simples possível? Se a lógica estiver muito complexa, pode ser sinal de que é necessário dividir o job em vários jobs.

☐ O processamento é realizado no mesmo lugar em que o dado está? De preferência, manter o dado ao lado do processamento para evitar problemas de latência e indisponibilidade.

☐ O uso de operações I/O foi minimizado? Equilibre o número de operações I/O com o uso de memória para obter o melhor desempenho possível:

- ☐ Foi utilizado cache para evitar operações desnecessárias de I/O?
- ☐ Dados não foram relidos na mesma transação?
- ☐ Scans desnecessários de tabelas e índices foram evitados?
- ☐ Foi utilizado leitor do tipo cursor em todas as leituras? O cursos é interessante quando não são trazidos muitos registros do banco de dados, caso contrário é melhor utilizar o leitor paginado.

☐ Foram utilizados valores chave nas cláusulas WHERE das consultas, sempre que possível?

☐ Não existe repetição de lógica? Não se repetir, por exemplo, realizando a leitura de uma mesma tabela duas vezes para realizar operações diferentes no mesmo batch.

☐ Foi alocado de antemão a memória suficiente para executar o batch? Isso é importante para evitar operações de realocação de memória que consomem tempo durante a execução do Job.

☐ A integridade dos dados foi considerada? Sempre zelar pela integridade do dado adicionando validações e checagens no Job.

☐ Foram utilizados checksums (checagem de somas para número de registros esperados, etc) para validação interna, sempre que possível?

☐ Foram planejados e realizados testes de stress com massa real? Isso permite identificar problemas de desempenho antes do batch chegar ao ambiente de produção.

☐ Foram realizados backups de arquivos, no caso de batches que processam arquivos?

☐ Foram evitadas operações de leitura de dados nos processadores? Uma forma de evitá-las é adicionar as informações consultadas na própria leitura e customizar o leitor para tratar o resultado.

☐ A janela de execução do batch foi considerada? É importante que a execução do batch não afete a disponibilidade do sistema online que depende dos dados.

☐ O desempenho foi otimizado utilizando técnicas de processamento concorrente? Nem sempre é interessante utilizá-las, essa deve ser a última alternativa caso o desempenho do batch não esteja aceitável.

☐ Foram considerados problemas de processamento concorrente? Se um sistema online pode atualizar o mesmo dado que um batch, ou o batch executa em múltiplas threads, é importante pensar em soluções de lock físico ou lógico dos dados.

☐ Para implementações com particionamento, foi adotada uma estratégia que otimize o processamento dos dados? Dependendo da natureza do dado, é possível optar por particionamentos fixos, por coluna chave, por views, por arquivo, etc.

Referências

https://docs.spring.io/spring-batch/docs/4.2.x/reference/html/spring-batch-intro.html#batchArchitectureConsiderations