Strecht Simulados · Guia de preparação



Guia de preparação DP-700

Tudo o que a prova cobra, organizado por domínio: conceitos-chave, comparações que confundem, pegadinhas reais, gatilhos do enunciado, questões comentadas e um plano para chegar pronto ao exame.

3 capítulos por domínio	**21** questões comentadas	**70%** nota de corte
6 provas completas no curso		

ANTES DE COMEÇAR

Como usar este guia

Este guia foi feito para ser estudado junto com as 6 provas completas do curso DP-700: o guia explica o que cai e onde estão as armadilhas; as provas treinam o ritmo e mostram o que você ainda não domina.

Quando	O que fazer
Semana 1	Faça o teste diagnóstico (página seguinte) e leia os capítulos 1 e 2. No fim da semana, faça a Prova 1 do curso no tempo do exame.
Semana 2	Leia os demais capítulos, começando pelos domínios em que você errou mais no diagnóstico. Faça as provas 2 e 3.
Semana 3	Releia os alertas de pegadinha e as tabelas de comparação. Faça as provas 4 e 5 e revise cada erro pela explicação.
Semana 4	Estude a folha de cola, faça a Prova 6 e refaça as provas abaixo de 80%. Agende o exame quando estiver acima de 80% em todas (a nota de corte é 70%).

Mapa do exame

Cada prova do curso tem 50 questões e 100 minutos, no formato do exame. Referência de aprovação: 70%.

Domínio (capítulo)	Peso	Neste guia
1. Implementar e gerenciar uma solução de análise	30-35%	18 conceitos · 14 pegadinhas
2. Ingerir e transformar dados	30-35%	18 conceitos · 14 pegadinhas
3. Monitorar e otimizar uma solução de análise	30-35%	18 conceitos · 14 pegadinhas

Como ler os capítulos: os **tópicos-chave** são o que mais cai; a **tabela de memorização** separa o que a prova costuma confundir; os **alertas de prova** mostram as armadilhas das alternativas erradas; os **gatilhos** ligam palavras do enunciado à resposta; e o **checklist** diz se você está pronto para o domínio.

Estude um pouco todo dia, de graça

Instagram @strecht.simulados

Um post de prova por dia: questão comentada, pegadinhas e o "não confunda" de cada certificação.

Seguir no Instagram →

YouTube Strecht Simulados

Shorts diários com pegadinhas de prova e, aos sábados, um simulado comentado.

Inscrever-se no canal →

DIAGNÓSTICO

Você já sabe isso?

Responda às 9 questões sem consultar nada. As respostas, com a explicação e o capítulo para revisar, estão no fim do guia. Comece o estudo pelos capítulos em que errou.

D1 · Implementar e gerenciar uma solução de análise

A equipe de SQL da Distribuidora Atlântico quer gerenciar o esquema do warehouse do Fabric como código no Visual Studio Code, compilando um .dacpac e publicando as diferenças no warehouse. Qual configuração atende?

A. Extensão Power Query SDK e projeto no formato PBIP com o destino definido como Fabric Lakehouse
B. Extensão Azure Data Factory e modelo ARM com o destino definido como o warehouse do workspace
C. Extensões SQL Database Projects e mssql, projeto SDK-style e plataforma Synapse Data Warehouse in Microsoft Fabric
D. Extensão Fabric Data Engineering e notebook ipynb com o destino definido como SQL analytics endpoint

D2 · Implementar e gerenciar uma solução de análise

A Gestora de Investimentos Norte Capital moveu o workspace de análise, que tinha OneLake diagnostics ligado, para uma nova capacidade. Desde então, nenhum evento novo de diagnóstico aparece. O que você deve fazer?

A. Escolher um novo lakehouse de diagnóstico dentro da nova capacidade
B. Reiniciar a capacidade antiga para liberar a gravação no lakehouse
C. Habilitar a imutabilidade dos logs para retomar a gravação dos eventos
D. Esperar 24 horas para que os eventos voltem a ser gravados sozinhos

D3 · Implementar e gerenciar uma solução de análise

Na Têxtil Fio de Ouro, um engenheiro definiu memória de driver e de executor em um ambiente do Fabric e o anexou ao notebook, mas as sessões continuam usando a configuração do pool padrão do workspace. O admin confirma que desligou a opção Customize compute configurations for items. Qual é a explicação?

A. As propriedades de computação do ambiente só passam a valer depois que o ambiente é publicado em Quick mode
B. A memória de driver e executor só pode ser ajustada quando o pool padrão do workspace é de nó único
C. O ambiente anexado precisa estar em outro workspace para que as propriedades de computação sejam aplicadas
D. Com a opção desligada, o pool padrão e sua computação valem para os ambientes do workspace

D4 · Ingerir e transformar dados

No eventhouse da Empresa de Energia Luz do Vale, a tabela Leituras recebe medições irregulares por sensor. O painel precisa de uma série por sensor com a contagem de leituras por hora nas últimas 24 horas, com zero nas horas sem leitura. Qual consulta KQL atende?

A. Leituras | make-series Total=count() on Sensor from ago(1d) to now() step 1h by Instante
B. Leituras | summarize Total=count() by bin(Instante, 1h), Sensor | where Instante > ago(1d)
C. Leituras | make-series Total=count() default=0 on Instante from ago(1d) to now() step 1h by Sensor
D. Leituras | summarize Total=dcount(Instante) by Sensor | extend Horas = range(0, 23, 1)

D5 · Ingerir e transformar dados

Um analista da Rede de Postos Combustível Bom compara a tabela Vendas, que tem várias linhas por IdPosto, com a tabela Postos em uma consulta KQL usando Vendas | join (Postos) on IdPosto, e o resultado traz bem menos linhas de vendas do que o esperado. Qual é a explicação?

A. O join padrão é leftsemi, que limita o resultado a uma linha por posto
B. O join padrão é innerunique, que deduplica as chaves da tabela da esquerda
C. O join padrão é leftanti, que traz só vendas sem posto correspondente
D. O join padrão é inner, que descarta as colunas da tabela da direita

D6 · Ingerir e transformar dados

Uma analista da Rede Calçados Passo Firme conectou o SSMS ao SQL analytics endpoint de um lakehouse e tentou executar um INSERT para corrigir algumas linhas de uma tabela Delta, mas o comando falhou. Qual é a explicação?

A. Falta a permissão ReadData no lakehouse, que é a permissão exigida para executar comandos INSERT no endpoint
B. O SQL analytics endpoint é somente leitura; a gravação deve ser feita pelo Spark ou em um warehouse
C. O endpoint passa a aceitar INSERT depois que os metadados das tabelas Delta são sincronizados
D. O INSERT precisa estar dentro de uma transação explícita com BEGIN TRAN e COMMIT no endpoint

D7 · Monitorar e otimizar uma solução de análise

Na Logística Ponte Aérea, um job Spark disparado por pipeline entrou na fila porque a capacidade estava sem núcleos livres, e a capacidade continuou ocupada por mais de um dia. O que acontece com esse job na fila?

A. Ele vira sessão interativa e roda com um nó de tamanho Small
B. Ele sai da fila depois de 24 horas e precisa ser reenviado
C. Ele roda na hora usando burst acima do limite da capacidade
D. Ele permanece na fila até a capacidade liberar núcleos, sem prazo

D8 · Monitorar e otimizar uma solução de análise

Na Rede de Lojas Mundo Bebê, o pipeline noturno só precisa atualizar a tabela Vendas e a partição do mês corrente do modelo semântico, e, se uma tabela falhar, as demais devem ser confirmadas mesmo assim. Como configurar a atividade Semantic model refresh?

A. Selecionar a tabela e a partição desejadas e usar o modo Transactional nelas
B. Manter a atualização completa e desligar o Wait on completion
C. Manter a atualização completa padrão e usar o modo Transactional
D. Selecionar a tabela e a partição desejadas e usar o modo Partial Batch

D9 · Monitorar e otimizar uma solução de análise

Na Corretora Agro Futuro, um engenheiro tenta criar, em um banco KQL, um atalho do OneLake para uma tabela Delta que tem uma coluna chamada preco*unidade, e a criação falha. O que ele deve fazer?

A. Criar o atalho em Files do lakehouse e consultá-lo pelo eventhouse
B. Criar o atalho com Accelerate desligado para aceitar o nome da coluna
C. Mudar a caching policy do banco KQL antes de repetir a criação
D. Renomear a coluna para remover o asterisco antes de criar o atalho

01

CAPÍTULO 1 · 30-35% DA PROVA

Implementar e gerenciar uma solução de análise

Aqui eu preciso dominar três frentes: configurações do workspace (Spark, Apache Airflow, domínios, OneLake), ciclo de vida (Git integration, deployment pipelines, variable library, SQL Database Projects) e segurança e governança (papéis, permissões de item, OneLake security, RLS, CLS, máscara, rótulos, endosso, auditoria). Também entra a orquestração: escolher entre Dataflow Gen2, atividade Copy, notebook, Copy job e Apache Airflow job, e montar pipelines com ForEach, Until, Lookup, parâmetros e expressões. A prova cobra muito o 'quem pode' (Admin, Member, Contributor, Viewer, domain admin, domain contributor) e o 'onde configura'. Os distratores dão permissão demais (Admin, CONTROL) ou de outro tipo (Monitor no lugar de ReadData). Onde mais erro: regras de implantação no estágio errado e o comportamento da CLS no Spark contra o SQL endpoint.

Neste capítulo
18 tópicos-chave · 14 pegadinhas · 7 macetes · 14 gatilhos · 4 questões comentadas

Tópicos-chave

Papéis de workspace

Admin conecta e desconecta Git, gerencia admins, atualiza e exclui o workspace e cria pools Spark. Member adiciona pessoas como Member, Contributor ou Viewer e permite que outros compartilhem itens. Contributor escreve dados, cria itens e executa pipelines. Viewer lê dados do warehouse por T-SQL e vê definições de pipelines e notebooks, mas não lê com Spark, não executa e não escreve. Com vários grupos, vale o maior papel.

Permissões de item e compartilhamento

Compartilhar um warehouse sem permissões extras dá só Read (equivalente a CONNECT); para ler dados, ReadData ou GRANT nos objetos. ReadAll dá acesso a arquivos por Spark e OneLake; para usar a URL do lakehouse é preciso Read all with Apache Spark; compartilhar lakehouse nunca concede escrita. Monitor dá acesso a DMVs e insights (padrão em Admin, Member e Contributor); Audit trata auditoria SQL (padrão só Admin); Reshare permite recompartilhar (padrão em Admin e Member).

Links de compartilhamento

People with existing access só gera a URL, sem conceder acesso. Specific people aceita convidados do Entra ID, mas não externos não convidados. People in your organization não funciona para convidados. Links que dão acesso incluem pelo menos Read, com permissões adicionais opcionais.

OneLake security

Funções com regras por pasta (Selected folders), linhas (RLS) e colunas (CLS) para Viewers e usuários com Read; papéis com escrita veem tudo. RLS e CLS da mesma tabela vão na mesma função; funções com ReadWrite não aceitam RLS nem CLS. Para restringir quem tem ReadAll, tire o usuário da DefaultReader (sem excluir a função). Grupos do Entra podem ser membros, mas mudanças de associação levam cerca de uma hora.

CLS: Spark x SQL endpoint

Com CLS da OneLake security, o Spark retorna só as colunas permitidas e o SQL analytics endpoint dá erro no SELECT *. Com CLS por GRANT no warehouse, SELECT * que inclui coluna proibida falha com permissão negada. Nenhuma das duas mascara valores; mascarar é papel da máscara dinâmica.

Permissões de pasta no OneLake

A permissão numa pasta desce para todas as subpastas. Para cima, só concede travessia para navegar até ela, sem dar acesso aos demais itens do pai nem a pastas irmãs.

Segurança T-SQL no warehouse

CREATE USER não é suportado: GRANT direto ao usuário ou grupo do Entra, que vira usuário (CREATE ROLE é possível). Autenticação é pelo Microsoft Entra ID, sem logins SQL com senha. sys.fn_my_permissions mostra as próprias permissões. RLS usa função com valor de tabela e CREATE SECURITY POLICY com FILTER PREDICATE; para mudar a função: DROP da política, ALTER FUNCTION e CREATE da política. RLS do warehouse faz o Direct Lake cair para DirectQuery.

Máscara dinâmica de dados

Funções: default (máscara completa, como XXXX), email (primeira letra e formato de e-mail), random (números em intervalo) e partial. UNMASK permite ver o dado real; ALTER ANY MASK com ALTER na tabela gerencia máscaras sem ver dados. Admin, Member e Contributor têm CONTROL, que inclui UNMASK. A máscara não impede inferência por filtros: combine com CLS, RLS e permissões granulares.

Rótulos de confidencialidade

Item criado a partir de outro herda o rótulo dele; com herança downstream, o rótulo do lakehouse propaga para modelo e relatório dependentes (do Power BI para itens do Fabric não é suportado). Rotulagem obrigatória é plena só em itens do Power BI. Se o usuário não vê rótulos, falta publicá-los para ele numa política do Purview. Rótulo padrão por área de negócio vai nas Delegated settings do domínio.

Endosso

Promoted: quem tem escrita no item. Certified: só revisores autorizados pelo admin. Master data: fonte única de verdade, só em itens com dados (lakehouse, modelo semântico). Dashboards do Power BI não recebem endosso; certificação e master data dependem de habilitação do admin do Fabric; certificadores por área podem ser delegados ao domínio.

Auditoria: Purview x OneLake diagnostics

O log de auditoria do Fabric é pesquisado no portal do Microsoft Purview com o papel Audit Logs, horários em UTC e intervalo padrão de sete dias; mostra, por exemplo, quem compartilhou um item. Quem leu qual arquivo do OneLake vem do OneLake diagnostics, que exige Admin do workspace, Contributor no lakehouse de destino na mesma capacidade; ao trocar de capacidade, escolha novo lakehouse. A imutabilidade dos logs é irreversível.

Domínios

Organizam e governam, mas não mudam quem acessa: todos os itens do workspace associado ganham o atributo do domínio, sem mover nada. Default domain associa os workspaces atuais e futuros de um grupo; os que já estão em outro domínio continuam nele se a configuração de tenant que permite sobrescrever associações não estiver habilitada. Domain admin gerencia contributors e associa workspaces, mas não renomeia, exclui nem cria admins. Domain contributor associa só workspaces em que é Admin, nas configurações do workspace. Subdomínio: criado por admin do Fabric ou do domínio, com admins herdados do pai.

Git integration

Só o Admin conecta e desconecta; provedores: Azure DevOps e GitHub na nuvem; uma branch e uma pasta por vez. Na conexão inicial com conteúdo nos dois lados, você escolhe a direção. Undo volta ao último sync e apaga itens novos. Conflito: Accept incoming changes ou Keep current content (Commit fica desabilitado); sem saber o que manter, crie branch nova, faça merge no Git e volte. Commit to new branch guarda mudanças sem trocar a conectada; a troca de branch pode ser delegada a Contributor. Boa prática: workspace e branch por desenvolvedor.

Deployment pipelines

A quantidade de estágios é definida na criação e fixa. Regras ficam no estágio de destino, nunca em Development, e exigem ser dono do item: lakehouse padrão para notebook e regra de fonte de dados para modelos semânticos (inclusive Direct Lake). Pareamento por nome e tipo, com a pasta como desempate; itens criados após a atribuição não são pareados e o deploy cria cópia. Implantar exige admin do pipeline e Contributor na origem e no destino; implantar para trás só para estágio vazio. Primeira implantação não leva dados: atualize os modelos no destino.

Variable library e SQL Database Projects

A variable library tem um value set ativo por workspace, definido por estágio, sem regras de parâmetro. SQL Database Projects (extensões SQL Database Projects e mssql, projeto SDK-style, plataforma Synapse Data Warehouse in Microsoft Fabric) versionam o esquema do warehouse: Build valida e gera o .dacpac; Publish compara e aplica as diferenças; Create Project From Database parte de um warehouse existente.

Configurações de Spark do workspace

Starter Pool usa nós Medium pré-aquecidos; pool personalizado vai de Small a XX-Large (X-Large e XX-Large indisponíveis em avaliação), com autoscale entre mínimo e máximo ou nó único (mínimo 1). vCores = CU x 2 x 3 (burst), divididos pelo tamanho do nó: na F64 com nós Large, 24 nós. Só Admin cria pools e altera o ambiente padrão. Reserve maximum cores garante núcleos por job; Set maximum job lifetime cancela jobs longos; alta simultaneidade para pipelines com session tag reutiliza sessões.

Apache Airflow job

Orquestração code-first com DAGs em Python. A versão não muda em job existente: crie outro job. Starter pool desliga após 20 minutos sem uso (desenvolvimento); custom pool fica ligado até ser pausado (produção) e ganha concorrência com extra nodes. O pool é configurado em Workspace settings > Data Factory > Apache Airflow Runtime Settings.

Orquestração: pipelines e notebookutils

Gatilho de evento de armazenamento com filtro no Subject e @pipeline()?.TriggerEvent?.FileName dispara na chegada de arquivo. ForEach não sequencial com Batch count controla paralelismo; Until repete até uma condição com timeout; Lookup sem First row only entrega output.value. Interpolação: vendas_@{pipeline().parameters.ano}.csv. Notebook recebe valores pela célula de parâmetros e Base parameters; notebookutils.notebook.run devolve string; exit fica fora de try-except; runMultiple com DAG dá dependências, paralelismo e retry, e falhas parciais vêm em RunMultipleFailedException (ex.result).

Macetes para memorizar

Papéis de workspace

Admin = Git e admins; Member = gente; Contributor = mão na massa; Viewer = só olha (T-SQL).

Regras de implantação

Regra mora no destino, nunca no Dev, e só o dono do item cria.

Máscara

UNMASK vê; ALTER ANY MASK mexe; CONTROL faz os dois (por isso não serve).

Endosso

Promoted = qualquer escritor; Certified = revisor autorizado; Master data = verdade única (só itens com dados).

Auditoria

Purview conta quem fez a ação; diagnostics conta quem leu o arquivo.

vCores Spark

CU x 2 x 3 dividido pelo tamanho do nó (F64 com Large = 384 / 16 = 24 nós).

> **Pasta no OneLake**
> Permissão desce como água; para cima, só a escada para chegar.

Tabela de memorização: não confunda

Quando é...	...e quando é
Dataflow Gen2 Analista com Power Query, transformações visuais em volumes baixos a médios e vários destinos.	**Notebook Spark** Transformação complexa com Python e bibliotecas sobre dados grandes e variados.
Atividade Copy em pipeline Muito volume e muitas fontes, sem código e sem transformação.	**Apache Airflow job** Orquestração code-first com DAGs em Python.
UNMASK A pessoa precisa ver o dado real da coluna mascarada.	**ALTER ANY MASK** A pessoa gerencia máscaras sem poder ver os dados (com ALTER na tabela).
Log de auditoria do Purview Ações de gestão, como quem compartilhou um item.	**OneLake diagnostics** Quem leu qual arquivo do OneLake, operação por operação.
Regra de lakehouse padrão Notebook implantado apontando para o lakehouse errado.	**Regra de fonte de dados** Modelo semântico (inclusive Direct Lake) que precisa apontar para o lakehouse do estágio.
Accept incoming changes Substituir a versão do workspace pela que vem do Git.	**Keep current content** Manter a versão do workspace como mudança não commitada.
Reserve maximum cores for active Spark jobs Garantir núcleos máximos para cada job ativo.	**Set maximum job lifetime** Cancelar jobs que passam de um limite de duração.

Números que caem

Item	Valor
Fórmula de vCores Spark	**CU x 2 x 3 (burst), dividido pelo tamanho do nó**
F64 com nós Large (16 vCores)	**24 nós; com Medium (8 vCores), 48; com X-Large (32 vCores), 12**
Mudança de grupo em função da OneLake security	**cerca de uma hora**
Agendamentos por pipeline	**até 20**

Starter pool do Apache Airflow job	**desliga após 20 minutos sem uso**
Log de auditoria no Purview	**horários em UTC; intervalo padrão dos últimos sete dias**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Criar a regra de implantação no estágio Development para que seja herdada.
Como não cair: Regras não existem em Development; são criadas no estágio de destino por quem é dono do item.

⚠ **Alerta de prova:** Converter o modelo Direct Lake para Import porque pipelines de implantação não o suportam.
Como não cair: Direct Lake é implantado normalmente; o vínculo é ajustado com regra de fonte de dados.

⚠ **Alerta de prova:** Dar Contributor para que a função da OneLake security passe a valer.
Como não cair: Papéis com escrita ignoram as restrições; o usuário precisa sair da DefaultReader e ficar só na função restrita.

⚠ **Alerta de prova:** Conceder ReadWrite para combinar RLS e CLS em funções diferentes.
Como não cair: ReadWrite não aceita RLS nem CLS; as regras de linhas e colunas vão na mesma função.

⚠ **Alerta de prova:** Esperar que a CLS da OneLake security mascare valores ou que o SQL endpoint omita a coluna.
Como não cair: No Spark a coluna é omitida; no SQL endpoint o SELECT * dá erro; nada é mascarado.

⚠ **Alerta de prova:** Dar CONTROL ou Contributor a quem só deve gerenciar máscaras.
Como não cair: CONTROL inclui UNMASK e Contributor tem CONTROL; o certo é ALTER ANY MASK mais ALTER na tabela.

⚠ **Alerta de prova:** Executar CREATE USER ou criar login SQL com senha no warehouse.
Como não cair: Não há CREATE USER nem login SQL: GRANT direto ao usuário ou grupo do Entra.

⚠ **Alerta de prova:** Desabilitar a security policy com STATE = OFF e alterar a função de RLS.
Como não cair: A função continua vinculada; é DROP da política, ALTER da função e CREATE da política.

⚠ **Alerta de prova:** Achar que associar o workspace a um domínio muda quem vê os itens.
Como não cair: Domínio não concede nem retira permissões; acesso vem de papéis e permissões de item.

⚠ **Alerta de prova:** Usar o log de auditoria do Purview para saber quem leu um arquivo do OneLake.
Como não cair: Leitura de arquivo é com OneLake diagnostics; o Purview registra ações como compartilhamento.

⚠ **Alerta de prova:** Tentar mudar a versão do Airflow ou adicionar estágios num recurso já criado.
Como não cair: Versão nova = novo Apache Airflow job; estágios novos = novo deployment pipeline.

⚠ **Alerta de prova:** Deixar a data de término vazia para o agendamento não acabar.
Como não cair: Não existe agendamento sem fim; use uma data de término distante.

⚠ **Alerta de prova:** Converter o valor para int antes do exit ou envolver o exit em try-except.
Como não cair: Valores de saída são sempre string, e exit dentro de try-except não tem efeito.

⚠ **Alerta de prova:** Esperar o e-mail de falha do agendamento numa execução manual.
Como não cair: Failure notifications cobrem só execuções agendadas.

Como cai na prova

📝 **Como cai:** Usuário não consegue fazer algo; quatro permissões ou papéis possíveis.
Como resolver: Escolha o menor privilégio que resolve: elimine Admin e CONTROL quando o requisito fala em mínimo, e confira se a permissão é do tipo certo (ReadData lê dados, Monitor só DMVs, ReadAll é Spark).

📝 **Como cai:** Problema depois de implantar entre estágios (item duplicado, lakehouse errado, visuais vazios).
Como resolver: Duplicado = item não pareado; lakehouse errado = regra de lakehouse padrão ou de fonte de dados no destino; vazio = implantação não leva dados, atualize o modelo.

📝 **Como cai:** Conflito ou necessidade de guardar trabalho no Git integration.
Como resolver: Identifique o que deve ser preservado: workspace (Keep current content ou Commit to new branch), Git (Accept incoming changes), os dois (nova branch, merge no Git, volta).

📝 **Como cai:** Escolher a ferramenta de orquestração ou transformação pelo perfil da equipe.
Como resolver: Sem código e Power Query: Dataflow Gen2; cópia em massa sem transformação: atividade Copy; Python complexo: notebook; DAG em Python: Apache Airflow job; carga inicial mais CDC sem pipeline: Copy job.

📝 **Como cai:** Expressão dinâmica em pipeline com quatro sintaxes parecidas.
Como resolver: Dentro de texto use @{...}; @@ escapa; sem @ é literal; dado do evento vem de TriggerEvent com o operador ?. para nulos; item do loop é @item().

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
Vários notebooks com dependências, paralelismo e repetição	**notebookutils.notebook.runMultiple com DAG**
Disparar o pipeline quando um arquivo chega	**Gatilho de evento de armazenamento com filtro no Subject**
Esperar uma condição externa com repetição e limite de tempo	**Atividade Until com Lookup e Wait**
Mesmo pipeline com valores diferentes por horário	**Vários agendamentos com parâmetros (nome idêntico ao parâmetro)**
Valor diferente por estágio sem regras de implantação	**Variable library com value set ativo por workspace**
Esquema do warehouse como código com .dacpac	**SQL Database Projects (SDK-style)**
Desenvolvedores se atropelando no mesmo workspace	**Workspace e branch por desenvolvedor, integração por pull request**
Fonte única de verdade de dados	**Selo Master data**
Workspaces atuais e futuros de um grupo no domínio	**Default domain**
Leitura de uma pasta específica para Viewers	**Função da OneLake security com Selected folders**
Relatório Direct Lake sobre warehouse em branco para o leitor	**ReadData no warehouse ou GRANT nas tabelas**
Egress alto com atalhos para S3 ou GCS	**Cache de atalhos na aba OneLake do workspace**
Automação que cria e executa notebooks	**Entidade de serviço com papel Contributor no workspace**
Notebook filho com lakehouse padrão diferente bloqueado no run	**useRootDefaultLakehouse: True**

🎯 **Se você lembrar de só uma coisa deste capítulo...**

- Só o Admin conecta ou desconecta o Git e cria pools Spark.
- Regras de implantação: no estágio de destino, pelo dono do item.
- Domínio não muda permissões.
- CLS da OneLake security: Spark omite a coluna, SQL endpoint dá erro.
- Valores de exit são sempre string, e exit nunca dentro de try-except.

Essenciais para a prova (checklist)

- ☐ Sei o que cada papel de workspace pode fazer e que vale o maior papel entre grupos.
- ☐ Sei a diferença entre Read, ReadData, ReadAll, Monitor, Audit e Reshare.
- ☐ Sei configurar OneLake security com pastas, RLS e CLS e o papel da DefaultReader.
- ☐ Sei implementar RLS, CLS e máscara dinâmica no warehouse e quem pode ver ou gerenciar máscaras.
- ☐ Sei como rótulos são herdados e onde configurar rótulo padrão e certificação por domínio.
- ☐ Sei quando usar o log de auditoria do Purview e o OneLake diagnostics.
- ☐ Sei resolver conflitos e trocas de branch no Git integration.
- ☐ Sei configurar deployment pipelines, regras, pareamento e permissões de implantação.
- ☐ Sei calcular nós Spark por capacidade e escolher entre starter pool, pool personalizado e nó único.
- ☐ Sei montar pipelines com ForEach, Until, Lookup, gatilho de evento e expressões dinâmicas.

Questões de revisão do capítulo

R1.1 · Implementar e gerenciar uma solução de análise

No warehouse da Loja Virtual Clique Já, a coluna Email da tabela dbo.Clientes deve aparecer para os atendentes no formato aXXX@XXXX.com, expondo só a primeira letra. Qual definição de máscara atende?

A. ALTER COLUMN Email ADD MASKED WITH (FUNCTION = 'hash()')
B. ALTER COLUMN Email ADD MASKED WITH (FUNCTION = 'email()')
C. ALTER COLUMN Email ADD MASKED WITH (FUNCTION = 'default()')
D. ALTER COLUMN Email ADD MASKED WITH (FUNCTION = 'random(1, 9)')

R1.2 · Implementar e gerenciar uma solução de análise

Na Seguradora Cuidado Total, o lakehouse LH_Sinistros tem o rótulo Confidencial. Uma engenheira cria um notebook diretamente a partir desse lakehouse. Que rótulo o notebook recebe?

A. O rótulo Confidencial, herdado do lakehouse na criação
B. Nenhum, pois notebooks não aceitam rótulos de confidencialidade
C. O rótulo padrão do tenant, já que o notebook é um item novo
D. O rótulo do workspace, que vale para todos os itens novos dele

R1.3 · Implementar e gerenciar uma solução de análise

Na Corretora de Seguros Bem Assistido, o engenheiro Rafael tem o papel Contributor no workspace e quer compartilhar um lakehouse com uma colega de outra área, mas a opção Share não aparece para ele. O que resolve, mantendo o papel atual de Rafael?

A. Rafael publicar o lakehouse em um aplicativo do Power BI para a colega
B. Um Admin habilitar o OneLake diagnostics para liberar o compartilhamento do item
C. Um Admin ou Member conceder a ele a permissão de recompartilhar o lakehouse
D. Rafael criar um atalho do lakehouse em um workspace pessoal da colega

R1.4 · Implementar e gerenciar uma solução de análise

No warehouse da Construtora Base Firme, a coluna Salario tem máscara default(), e os analistas sem UNMASK veem 0. Um auditor demonstrou que, filtrando faixas de valor no WHERE, consegue descobrir salários exatos. Qual conclusão e ação seguem a orientação da Microsoft?

A. A máscara não impede inferência; combine-a com CLS, RLS e permissões granulares
B. A máscara default() falhou; troque-a pela função random() para impedir a inferência
C. A máscara vale só para SELECT *; proíba os analistas de usar a cláusula WHERE
D. A máscara só vale no Power BI; aplique-a de novo no modelo semântico do warehouse

Respostas comentadas

R1.1. B) ALTER COLUMN Email ADD MASKED WITH (FUNCTION = 'email()')

Por quê: A máscara dinâmica de dados oferece default(), email(), random() e partial(prefixo, preenchimento, sufixo). Os dados armazenados não mudam; só o resultado da consulta é mascarado.

✗ A) Incorreta. hash() não está entre as funções de máscara (default, email, random e partial).
✗ C) Incorreta. default() aplica a máscara completa do tipo, como XXXX para textos.
✗ D) Incorreta. random() serve para colunas numéricas, trocando o valor por um número aleatório no intervalo.

> **Dica de prova:** Máscara no formato de e-mail com a primeira letra: email().

R1.2. A) O rótulo Confidencial, herdado do lakehouse na criação

Por quê: Outros exemplos de herança na criação: pipeline criado a partir de lakehouse ou notebook, atalho de lakehouse a partir de lakehouse e KQL queryset a partir de banco KQL.

✗ B) Incorreta. Rótulos podem ser aplicados a todos os itens do Fabric.

✗ C) Incorreta. Na herança na criação, prevalece o rótulo do item de origem.

✗ D) Incorreta. Workspaces não têm rótulo próprio aplicado aos itens.

> **Dica de prova:** Item criado a partir de outro herda o rótulo dele.

R1.3. C) Um Admin ou Member conceder a ele a permissão de recompartilhar o lakehouse

Por quê: Contributor escreve e executa itens, mas o compartilhamento é controlado pela permissão de recompartilhamento, concedida por Admins ou Members.

✗ A) Incorreta. Aplicativos não são a forma de compartilhar um lakehouse.

✗ B) Incorreta. O OneLake diagnostics registra acessos; não altera permissões.

✗ D) Incorreta. Criar atalho exige permissões no destino do atalho e não concede acesso ao lakehouse.

> **Dica de prova:** Contributor sem Share: precisa da permissão de recompartilhar o item.

R1.4. A) A máscara não impede inferência; combine-a com CLS, RLS e permissões granulares

Por quê: A documentação alerta que usuários sem privilégios, mas com permissão de consulta, podem inferir dados mascarados. A máscara não deve ser a única proteção de dados sensíveis.

✗ B) Incorreta. Trocar a função não impede a inferência por filtros.

✗ C) Incorreta. A máscara vale para qualquer consulta; o risco de inferência continua com filtros.

✗ D) Incorreta. A máscara é aplicada no warehouse para os resultados das consultas, não só no Power BI.

> **Dica de prova:** Máscara dinâmica não protege contra inferência: combine com CLS e RLS.

02

CAPÍTULO 2 · 30-35% DA PROVA

Ingerir e transformar dados

Aqui eu preciso escolher a ferramenta certa de ingestão (Copy job, mirroring, atalho, pipeline, Dataflow Gen2, eventstream, notebook) e o armazenamento certo (lakehouse, warehouse, eventhouse, SQL database in Fabric) a partir de poucos detalhes do enunciado: 'sem código', 'sem copiar dados', 'em tempo real', 'só as mudanças', 'T-SQL com transações'. A prova também cobra modelagem dimensional de verdade: tipo de fato, tipo de dimensão, SCD, membro inferido e chave substituta com IDENTITY no warehouse. Na parte de transformação caem código PySpark (MERGE do Delta, joins, nulos), T-SQL (COPY INTO, OPENROWSET, CTAS) e muito KQL (summarize com bin, make-series, arg_max, mv-expand, parse_json). No streaming, o foco é o tipo de janela temporal, watermark e checkpoint no Spark e update policy, materialized view e Query acceleration no eventhouse. Onde mais erro: confundir marca d'água com CDC, achar que o SQL analytics endpoint aceita INSERT e trocar hopping por tumbling ou session.

Neste capítulo
18 tópicos-chave · 14 pegadinhas · 8 macetes · 14 gatilhos · 4 questões comentadas

Tópicos-chave

Copy job

Assistente que move dados de muitas fontes para muitos destinos sem montar pipeline, nos modos Full copy ou Incremental copy (por marca d'água ou CDC). Ele guarda sozinho o estado da última execução: se uma execução falha, a próxima retoma do fim da última execução bem-sucedida, sem redefinir a marca d'água à mão. O método padrão no destino é Append; Merge atualiza linhas existentes usando uma coluna de chave (a chave primária, se houver, é usada por padrão em Merge e SCD Type 2). As audit columns gravam a linhagem por linha, como o ID da execução.

Marca d'água x CDC na cópia incremental

A marca d'água exige uma coluna incremental confiável (data de modificação, por exemplo) e não captura exclusões. O CDC usa o rastreamento de mudanças da própria origem e replica inserções, atualizações e exclusões, sem exigir coluna de data. Por padrão o Copy job não exclui dados do destino; truncar antes da carga completa é opcional.

Mirroring (database, metadata e open)

Database mirroring replica continuamente bancos como Azure SQL Database em tabelas Delta no OneLake, sem ETL, com SQL analytics endpoint somente leitura; dá para replicar o banco inteiro ou só tabelas escolhidas. Metadata mirroring sincroniza só metadados de catálogos suportados e usa atalhos. Open mirroring recebe mudanças que uma aplicação grava na landing zone, servindo para sistemas sem conector. No database mirroring e no open mirroring, a computação da replicação não consome CU e há armazenamento grátis proporcional à capacidade (1 TB por CU); consultas aos dados espelhados (SQL, Power BI ou Spark) são cobradas normalmente.

Atalhos do OneLake (shortcuts)

Referenciam dados sem copiar. Em Tables, o atalho só pode ficar no nível superior; em Files, em qualquer nível. Excluir o atalho mantém os dados no destino, mas excluir um arquivo dentro do atalho apaga na origem; mover, renomear ou excluir o caminho de destino pode quebrar o atalho (aponte-o para o caminho novo). Para criar, é preciso escrita no caminho onde o atalho fica (ReadWrite da OneLake security) e leitura no destino.

Autenticação de atalhos

Atalhos internos do OneLake usam pass-through (padrão, identidade de cada usuário, que precisa de acesso ao destino) ou delegated identity com uma conexão, quando os consumidores não têm acesso direto à origem. Para ADLS Gen2 em outro tenant do Entra, funcionam entidade de serviço ou token SAS; workspace identity e conta organizacional não. Para leitura com Entra ID, o papel mínimo na conta é Storage Blob Data Reader.

Lakehouse com esquemas

Esquemas vêm habilitados por padrão em lakehouses criados pelo portal, e saveAsTable sem esquema grava no dbo. Para outro esquema, use esquema.tabela. Entre workspaces, o Spark SQL usa nome de quatro partes: workspace.lakehouse.esquema.tabela. Schema shortcut (só em lakehouse com esquemas) expõe todas as tabelas de um esquema de outro lakehouse e reflete tabelas novas automaticamente.

Warehouse x SQL analytics endpoint

O warehouse oferece T-SQL de leitura e escrita com transações multitabela. O SQL analytics endpoint do lakehouse é gerado pelo sistema e somente leitura: INSERT, UPDATE, DELETE, CREATE TABLE e ALTER TABLE falham, e nenhuma permissão muda isso. Dados do lakehouse são gravados com Spark, pipelines ou dataflows. Para OLTP com chaves estrangeiras impostas, a escolha é SQL database in Fabric.

Ingestão no warehouse (COPY INTO, OPENROWSET, CTAS)

COPY INTO é o caminho de maior throughput para arquivos de um container, mas não aceita WHERE, CAST nem joins. Para filtrar e converter durante a carga, use INSERT INTO ... SELECT ... FROM OPENROWSET(BULK ...). CTAS cria tabela nova (não acrescenta linhas) e aceita join entre itens do workspace com nomes de três partes. INSERT linha a linha degrada consultas: acumule em arquivos de pelo menos 4 MB e carregue em lote.

IDENTITY no warehouse do Fabric

Só aceita BIGINT, sem seed nem incremento personalizados, e garante unicidade, não ordem nem sequência sem lacunas. Não dá para adicionar IDENTITY com ALTER TABLE: crie tabela nova e carregue com INSERT...SELECT. Para inserir o membro -1, use SET IDENTITY_INSERT ON com lista de colunas, desligue e rode DBCC CHECKIDENT com RESEED (sem valor personalizado). Só uma tabela por sessão pode ter IDENTITY_INSERT ligado.

Chaves NOT ENFORCED e membros especiais

No warehouse, PRIMARY KEY, UNIQUE e FOREIGN KEY só existem como NOT ENFORCED, então o ETL testa a integridade. Se o lookup de dimensão falha, a linha do fato não é descartada nem gravada com chave nula: entra com o membro especial Desconhecido (inserido pelo processo de carga, não pelo motor) e é corrigida depois. Dimensões com chaves substitutas geradas nunca devem ser truncadas e recarregadas.

Tipos de tabela fato

Transacional registra eventos que não mudam depois de inseridos. Snapshot periódico registra medições em intervalos, como saldo diário, que é semiaditivo e não pode ser somado ao longo do tempo. Snapshot acumulado tem várias chaves de data, uma por marco do processo, para medir durações. Factless fact table não tem medidas e registra só ocorrências.

Tipos de dimensão

Dimensão degenerada é um atributo de mesmo grão no próprio fato, como número do pedido ou da nota. Junk dimension junta vários atributos pequenos de baixa cardinalidade. Role-playing é a mesma dimensão ligada várias vezes ao fato (datas de pedido, envio, entrega). A dimensão de data usa chave int AAAAMMDD, e hora fica em dimensão separada. Para Direct Lake, dimensões desnormalizadas com todos os atributos de filtro, já que colunas calculadas não são suportadas.

SCD, membro inferido e soft delete

SCD tipo 1 sobrescreve, tipo 2 cria nova versão com validade e flag de atual (por isso exige chave substituta, já que a chave natural se repete), tipo 3 guarda o valor anterior em coluna. Membro inferido é criado pela carga de fatos quando a chave natural chega antes dos atributos; quando os detalhes chegam, a linha é atualizada e a flag de inferido vira falsa, sem nova versão SCD. Membro excluído na origem com fatos ligados recebe soft delete (IsDeleted).

Dataflow Gen2: destinos, staging e fast copy

Métodos de atualização: Replace e Append; esquema dinâmico ou fixo só se aplica ao Replace. Warehouse aceita só esquema fixo; banco KQL aceita só Append. Para lakehouse, o staging vem desligado por padrão; ligue em consultas com junções e agregações pesadas. Fast copy usa o backend da Copy activity (arquivos de 100 MB ou mais, ou 5 milhões de linhas em bancos), carrega direto só em lakehouse e não aceita agrupamentos: ingira em uma consulta e transforme em outra que a referencia.

Incremental refresh do Dataflow Gen2

Extract data from the past define o horizonte da carga inicial; Bucket size, a granularidade; a coluna de detecção de mudança compara o valor máximo por bucket; Only extract data for concluded periods evita carregar o período em andamento. Quando um bucket muda, o dataflow apaga e insere o bucket inteiro (só Replace, nunca MERGE por chave), e dados fora do intervalo não são afetados.

MERGE em tabelas Delta

whenMatchedUpdateAll atualiza, whenNotMatchedInsertAll insere e whenNotMatchedBySourceDelete apaga do destino o que sumiu da origem. Esta última só cabe quando a origem é o retrato completo; se o arquivo traz só mudanças, use UPDATE e INSERT, sem DELETE por origem. Para acrescentar só o inédito (evitar reenvios), use MERGE com WHEN NOT MATCHED THEN INSERT.

Eventstream

Captura eventos contínuos sem código e roteia para vários destinos no mesmo item. Operadores: Filter, Join (enriquecer com referência), Union (empilhar fluxos de mesmos campos), Expand (array em linhas), Group by (agregar em janelas) e Manage fields. O default stream recebe os eventos brutos; derived streams publicam o fluxo transformado no Real-Time hub. Fontes CDC trazem mudanças de bancos como Azure SQL Database; custom endpoint oferece endpoint Kafka. Para filtrar antes de gravar no eventhouse, use Event processing before ingestion, não Direct ingestion. Entrega at least once, mensagens de até 1 MB.

Spark Structured Streaming e KQL no eventhouse

No Spark, checkpointLocation guarda o progresso para retomar, withWatermark mais window tolera dados atrasados, trigger com intervalo maior reduz arquivos pequenos, e um job 24x7 roda como Spark job definition com política de repetição. No eventhouse, update policy enriquece na ingestão (com join, desabilite a streaming ingestion), materialized view com arg_max deduplica de forma gerenciada, extent tags ingest-by evitam reingerir o mesmo lote e atalhos com Query acceleration (Accelerate = On, consulta por external_table()) dão desempenho próximo ao nativo, sem suportar materialized views nem update policies.

Macetes para memorizar

Janelas temporais

Tumbling = Tijolos lado a lado; Hopping = Hops que se sobrepõem; Sliding = Sai ou entra evento, emite; Session = Silêncio fecha a sessão; Snapshot = 'foto' do instante: agrupa só eventos com o mesmo timestamp.

Mirroring

Database Copia, Metadata Referencia, Open Recebe.

SCD

1 sobrescreve, 2 duplica a linha (versões), 3 guarda o 'ex' na coluna ao lado.

Dimensões especiais

Degenerada = número da nota no fato; Junk = gaveta de flags; Role-playing = mesma data em vários papéis.

Dataflow Gen2 e destino

Warehouse só Fixo, KQL só Append, incremental só Replace.

Duplicados no KQL

Lote repetido = ingest-by; repetido sempre = materialized view; olhar rápido = arg_max.

IDENTITY no Fabric

Bigint, único, sem ordem: 'BUS' (Bigint, Único, Sem sequência).

Atalhos em Tables

Tables só no térreo; Files aceita qualquer andar.

Tabela de memorização: não confunda

Quando é...	...e quando é
Warehouse Precisa de T-SQL de escrita (INSERT, UPDATE, DELETE), transações multitabela e dados estruturados.	**Lakehouse (SQL analytics endpoint)** Engenharia com Spark em big data; o endpoint só lê, e a gravação é feita por Spark, pipelines ou dataflows.
Copy job com marca d'água A origem tem coluna incremental confiável e exclusões não precisam ser propagadas.	**Copy job com CDC** Precisa replicar inserções, atualizações e exclusões usando o rastreamento de mudanças da origem.
COPY INTO Carga de arquivos do container com maior throughput, só com opções de formato.	**INSERT ... SELECT FROM OPENROWSET** Precisa filtrar linhas e converter colunas durante a carga em uma tabela existente.
Eventstream Eventos contínuos sem código, com Filter, Join, Group by e vários destinos.	**Spark Structured Streaming** Lógica complexa em Python ou bibliotecas próprias, com checkpoint e watermark.
Janela tumbling Contagem por período fixo, sem sobreposição (um evento entra em um único resultado).	**Janela hopping** Resultado frequente sobre um período maior, com sobreposição (ex.: a cada 5 minutos, os últimos 15).

Janela session Agrupar rajadas de atividade separadas por silêncio (timeout e duração máxima).	**Janela sliding** Emitir resultado sempre que um evento entra ou sai da janela.
Materialized view com arg_max Duplicados constantes entre lotes diferentes, com solução gerenciada.	**Extent tags ingest-by** O mesmo lote inteiro pode ser reenviado e não deve entrar de novo.

Números que caem

Item	Valor
Arquivos para ingestão em lote no warehouse	**pelo menos 4 MB**
Fast copy no Dataflow Gen2	**arquivos de 100 MB ou mais, ou 5 milhões de linhas em bancos**
Mensagem no eventstream	**máximo de 1 MB**
Retenção de eventos no eventstream	**padrão de um dia; até 90 dias**
Garantia de entrega do eventstream	**at least once (pelo menos uma vez)**
Query acceleration	**até 900 colunas; arquivo Parquet acima de 6 GB não entra no cache**
Armazenamento grátis de mirroring	**1 TB por CU da capacidade (ex.: 64 TB em 64 CUs)**
Chave da dimensão de data	**int no formato AAAAMMDD (ex.: 20261022)**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Achar que, com a permissão certa ou dentro de BEGIN TRAN, o SQL analytics endpoint aceita INSERT.
Como não cair: O endpoint é somente leitura sempre; escrita em T-SQL é no warehouse, e no lakehouse se grava com Spark.

⚠ **Alerta de prova:** Usar a marca d'água para propagar exclusões da origem.
Como não cair: Marca d'água não vê linhas apagadas; exclusões exigem CDC.

⚠ **Alerta de prova:** Colocar WHERE e CAST no COPY INTO para filtrar a carga.
Como não cair: COPY INTO não transforma; filtro e conversão vão em INSERT...SELECT FROM OPENROWSET.

⚠ **Alerta de prova:** Usar CTAS para acrescentar linhas a uma tabela que já existe.
Como não cair: CTAS sempre cria tabela nova; para acrescentar, INSERT...SELECT.

⚠ **Alerta de prova:** Definir IDENTITY(1,1) em int ou adicionar IDENTITY com ALTER TABLE.
Como não cair: No Fabric, IDENTITY é só BIGINT, sem seed/incremento, e exige tabela nova carregada com INSERT...SELECT.

⚠ **Alerta de prova:** Usar whenNotMatchedBySourceDelete quando o arquivo traz só os registros alterados do dia.
Como não cair: Exclusão por origem só quando a origem é o retrato completo; senão apagaria quem não veio no arquivo.

⚠ **Alerta de prova:** Marcar ativo = false com whenNotMatchedBySourceUpdate quando o requisito pede remover.
Como não cair: Se a tabela deve ficar idêntica ao cadastro, é whenNotMatchedBySourceDelete.

⚠ **Alerta de prova:** Criar um atalho por tabela para expor um esquema inteiro de outro lakehouse.
Como não cair: Atalhos por tabela não pegam tabelas novas; schema shortcut reflete o esquema todo.

⚠ **Alerta de prova:** Achar que excluir um arquivo dentro do atalho só remove uma cópia local.
Como não cair: O atalho não guarda cópia: apagar arquivo dentro dele apaga na origem; apagar o atalho não apaga dados.

⚠ **Alerta de prova:** Escolher Append ou bucket diário para não ver números parciais do mês no incremental refresh.
Como não cair: Incremental refresh só suporta Replace; para períodos fechados, Only extract data for concluded periods.

⚠ **Alerta de prova:** Contar por hora com summarize ... by bin e esperar zeros nas horas sem leitura.
Como não cair: bin não cria linhas vazias; make-series com default=0 preenche os buracos.

⚠ **Alerta de prova:** Estranhar que Vendas | join (Postos) trouxe menos linhas e culpar leftsemi ou inner.
Como não cair: O join padrão do KQL é innerunique, que deduplica as chaves da tabela da esquerda.

⚠ **Alerta de prova:** Resolver duplicados com arg_max() em cada relatório ou com retention policy curta.
Como não cair: arg_max na consulta só esconde; retenção apaga por idade. Use materialized view de deduplicação ou ingest-by para lotes reenviados.

⚠ **Alerta de prova:** Usar update policy para gravar em lakehouse ou rodar Python.
Como não cair: Update policy grava só em tabelas do mesmo banco KQL e roda KQL; lógica Python vai para Spark.

Como cai na prova

📝 **Como cai:** Cenário de empresa com requisitos como 'sem código', 'sem copiar dados', 'sem pipeline' e quatro ferramentas parecidas.
Como resolver: Sublinhe cada restrição e elimine a opção que viola uma delas: Copy job e mirroring dispensam pipeline, atalho dispensa cópia, eventstream dispensa código; depois confira se a sobrevivente atende latência e exclusões.

📝 **Como cai:** Quatro trechos de KQL ou PySpark quase iguais.
Como resolver: Compare ordem dos argumentos (eixo de tempo em make-series, parâmetros de row_window_session), operador (bin x make-series, arg_max x arg_min) e cláusulas do MERGE; a errada costuma inverter parâmetros ou trocar um operador vizinho.

📝 **Como cai:** Escolha do armazenamento (lakehouse, warehouse, eventhouse, SQL database).
Como resolver: T-SQL com escrita e transações: warehouse; Spark e big data: lakehouse; telemetria com consultas interativas: eventhouse; OLTP com FKs impostas: SQL database in Fabric.

📝 **Como cai:** Problema de modelagem dimensional descrito em linguagem de negócio.
Como resolver: Traduza: marcos com datas = snapshot acumulado; saldo diário = snapshot periódico semiaditivo; só ocorrências = factless; atributo único do fato = degenerada; chave ausente = membro Desconhecido.

📝 **Como cai:** Múltipla resposta (Escolha TRÊS) sobre características de um recurso.
Como resolver: Avalie cada afirmação isolada como verdadeiro ou falso; as falsas costumam dizer 'exatamente uma vez', 'aceita INSERT', 'exige pipeline' ou trocar o padrão (retenção, método).

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
Assistente sem pipeline, carga inicial completa e depois só mudanças	**Copy job com Incremental copy**
Sistema sem conector de mirroring, mas a aplicação exporta mudanças	**Open mirroring com landing zone**

Réplica contínua de Azure SQL no OneLake, sem ETL, para Spark e Power BI	**Database mirroring (endpoint somente leitura)**
Usar banco do Azure Data Explorer no Fabric sem reingestão	**Database shortcut (follower)**
Mudanças de linhas de banco em tempo real, sem código, para um eventhouse	**Fonte CDC do eventstream**
Aplicação Kafka existente precisa enviar para o Fabric	**Custom endpoint com endpoint Kafka**
Compartilhar o fluxo já transformado no Real-Time hub	**Derived stream**
Arquivos pequenos no destino lakehouse do eventstream	**Aumentar Minimum rows e Maximum duration**
Enriquecer na ingestão do eventhouse sem orquestração	**Update policy com função KQL**
Histórico no OneLake com consulta KQL rápida, sem duplicar	**Atalho no banco KQL com Query acceleration**
Processo com marcos e tempo entre eles	**Fato de snapshot acumulado**
Último registro por chave no KQL	**summarize arg_max(tempo, *) by chave**
Coluna com JSON guardado como texto no KQL	**parse_json antes de acessar propriedades; array em linhas com mv-expand**
Retomar streaming de onde parou / tolerar eventos atrasados	**checkpointLocation / withWatermark mais window**

🎯 Se você lembrar de só uma coisa deste capítulo...

- SQL analytics endpoint só lê; warehouse escreve com T-SQL.
- Exclusões na cópia incremental exigem CDC; marca d'água não vê DELETE.
- Linha de fato nunca é descartada por falta de dimensão: vai para o membro Desconhecido.
- Atalho acelerado continua sendo tabela externa: sem materialized view e sem update policy.
- Join padrão do KQL é innerunique.

Essenciais para a prova (checklist)

- ☐ Sei quando usar Copy job, mirroring, atalho, atividade Copy, Dataflow Gen2, eventstream e notebook.
- ☐ Sei a diferença entre database, metadata e open mirroring e o que o mirroring cobra.
- ☐ Sei as regras de atalhos: nível em Tables, exclusão, autenticação entre tenants e permissões para criar.
- ☐ Sei escolher entre COPY INTO, OPENROWSET com INSERT...SELECT e CTAS no warehouse.
- ☐ Sei as limitações de IDENTITY no Fabric e a sequência para inserir o membro -1.

- ☐ Sei identificar fato transacional, snapshot periódico, snapshot acumulado e factless.
- ☐ Sei montar um MERGE do Delta com whenNotMatchedBySourceDelete e quando não usá-lo.
- ☐ Sei as opções de destino do Dataflow Gen2 (método, esquema, staging, fast copy) e do incremental refresh.
- ☐ Sei escolher entre tumbling, hopping, sliding, session e snapshot.
- ☐ Sei escrever KQL com summarize/bin, make-series, arg_max, dcount, mv-expand e parse_json.

Questões de revisão do capítulo

R2.1 · Ingerir e transformar dados

A Rede de Lojas Calçar Bem quer levar para o Fabric, em tempo real, as inserções, atualizações e exclusões de linhas de três tabelas de um Azure SQL Database, sem escrever código, e roteá-las para um eventhouse. Qual fonte do eventstream atende?

A. A fonte Fabric workspace item events
B. A fonte Fabric job events
C. A fonte Fabric OneLake events
D. A fonte Azure SQL Database CDC

R2.2 · Ingerir e transformar dados

No warehouse da Construtora Novo Lar, cada contrato de obra passa por assinatura, liberação do terreno, início da obra e entrega das chaves, em datas diferentes ao longo de meses. A diretoria quer analisar o tempo entre esses marcos. Qual tipo de tabela fato é o mais indicado?

A. Fato de snapshot periódico diário
B. Fato transacional
C. Fato de snapshot acumulado
D. Factless fact table

R2.3 · Ingerir e transformar dados

Um engenheiro da Rede de Cafeterias Grão Torrado vai criar, em um banco KQL, um atalho do OneLake para uma tabela Delta de um lakehouse e já quer consultas com desempenho próximo ao de tabelas nativas. Como habilitar isso na criação?

A. Definir a retention policy do atalho em 365 dias
B. Criar o atalho em Files em vez de Tables
C. Usar o comando .ingest into apontando para o atalho
D. Definir Accelerate como On ao criar o atalho

R2.4 · Ingerir e transformar dados

No eventhouse da Plataforma de Streaming Tela Viva, a tabela Cliques tem IdUsuario e Instante. O time de produto quer marcar o início de cada sessão por usuário, em que uma nova sessão começa após 30 minutos sem cliques ou após 2 horas desde o início da sessão. Qual consulta KQL atende?

A. Cliques | sort by IdUsuario asc, Instante asc | extend Inicio = row_window_session(Instante, 2h, 30m, IdUsuario != prev(IdUsuario))
B. Cliques | summarize Inicio = min(Instante) by IdUsuario, bin(Instante, 30m) | order by IdUsuario asc, Inicio asc
C. Cliques | extend Inicio = row_window_session(Instante, 30m, 2h) | summarize count() by IdUsuario, Inicio
D. Cliques | make-series Inicio = count() on Instante step 30m by IdUsuario | mv-expand Inicio

Respostas comentadas

R2.1. D) A fonte Azure SQL Database CDC

Por quê: Eventstreams têm fontes de CDC para Azure SQL Database, Azure SQL Managed Instance, SQL Server em VM, PostgreSQL, MySQL, Cosmos DB e MongoDB, além de eventos do próprio Fabric.

✗ A) Incorreta. Workspace item events tratam criação, atualização e exclusão de itens do Fabric.
✗ B) Incorreta. Job events tratam execuções de jobs, como pipelines e notebooks.
✗ C) Incorreta. OneLake events tratam mudanças em arquivos e pastas do OneLake.

> **Dica de prova:** Mudanças de linhas de um banco em tempo real, sem código: fonte CDC do eventstream.

R2.2. C) Fato de snapshot acumulado

Por quê: Fatos de snapshot acumulado têm várias chaves de data, uma por marco, e algumas podem ficar como N/A até o processo chegar ao marco.

✗ A) Incorreta. O snapshot periódico registra medições em intervalos, como saldos diários.
✗ B) Incorreta. O fato transacional registra eventos que não mudam depois de inseridos.
✗ D) Incorreta. A factless fact table não tem medidas e registra apenas ocorrências.

> **Dica de prova:** Processo com marcos e durações: snapshot acumulado.

R2.3. D) Definir Accelerate como On ao criar o atalho

Por quê: No banco KQL: + > New > OneLake shortcut, escolha a origem, os dados e, opcionalmente, Accelerate = On; a consulta é feita com external_table().

✗ A) Incorreta. A retention policy não acelera tabelas externas.
✗ B) Incorreta. No banco KQL, o atalho aparece em Shortcuts; a pasta não habilita aceleração.
✗ C) Incorreta. Ingerir copia os dados para uma tabela nativa, o que não é o uso do atalho acelerado.

> **Dica de prova:** Atalho no KQL já acelerado: Accelerate = On na criação.

R2.4. A) Cliques | sort by IdUsuario asc, Instante asc | extend Inicio = row_window_session(Instante, 2h, 30m, IdUsuario != prev(IdUsuario))

Por quê: row_window_session(Expr, MaxDistanceFromFirst, MaxDistanceBetweenNeighbors, Restart) exige linhas serializadas, por exemplo após sort, e devolve o valor de início de cada sessão.

✗ B) Incorreta. bin cria intervalos fixos de 30 minutos, não sessões por inatividade.

✗ C) Incorreta. Os parâmetros estão invertidos, falta ordenar o conjunto e não há reinício por usuário.

✗ D) Incorreta. make-series cria séries de contagem em passos fixos, não sessões.

Dica de prova: Sessões por inatividade no KQL: row_window_session sobre dados ordenados.

03

CAPÍTULO 3 · 30-35% DA PROVA

Monitorar e otimizar uma solução de análise

Este domínio mistura diagnóstico e ajuste fino: eu preciso saber onde olhar (Monitoring hub, workspace monitoring, Capacity Metrics app, queryinsights, Data insights do eventstream, logs do Dataflow Gen2) e qual alavanca puxar (V-Order, OPTIMIZE, VACUUM, estatísticas, CLUSTER BY, caching policy, native execution engine). A prova descreve um sintoma concreto, como erro 430, conflito 24556, arquivos pequenos, primeira execução lenta ou alerta que não dispara, e oferece quatro ferramentas reais, das quais só uma atende ao escopo pedido. Muitos distratores trocam a ferramenta de monitoramento pela de outro serviço: queryinsights para Spark, Data insights para pipelines, Capacity Metrics para logs. Também caem configurações de atividade de pipeline (Retry, Timeout, Secure output, Deactivate, Fail) e o Activator para alertas. Onde mais erro: confundir caching policy com retention policy e os três lugares de configurar o V-Order.

Neste capítulo
18 tópicos-chave · 14 pegadinhas · 7 macetes · 14 gatilhos · 4 questões comentadas

Tópicos-chave

Monitoring hub

Mostra execuções de itens que a pessoa tem permissão de ver, inclusive de outros usuários e sob demanda. Tem colunas Upstream run e Downstream runs para ver quem disparou o quê, visão Gantt para comparar durações e sobreposições, exportação para CSV, entrada e saída de cada atividade e Rerun a partir da atividade que falhou. A manutenção de tabela feita pelo portal aparece como TableMaintenance, e métricas de micro-batch do Spark Streaming ficam na aba Structured Streaming.

Workspace monitoring e alertas

Grava logs em um eventhouse, como ItemJobEventLogs, sem backfill de atividades anteriores à ativação e com retenção padrão de 30 dias. Um alerta único para falhas de todos os pipelines do workspace sai de uma consulta KQL em ItemJobEventLogs usada por uma regra do Activator. Failure notifications do agendamento cobrem só execuções agendadas e só falhas.

Activator

Detecta condições e dispara ações: executar notebook ou pipeline do Fabric, Power Automate, Teams e e-mail (não executa T-SQL avulso nem muda SKU). Com job events do Fabric, alerta sucesso e falha de pipelines ou atualização de modelo semântico sem alterar o pipeline. Objetos agrupados por ID com regras de mudança de estado e de ausência de dados dão alerta por dispositivo; no eventstream, o destino Activator alerta direto do fluxo.

Configurações de atividade de pipeline

Retry repete só após falha, com intervalo padrão de 30 segundos; Timeout padrão de 12 horas e máximo de sete dias. Secure input/output esconde valores nos logs, como tokens. Deactivate pula a atividade sem apagá-la. Fail activity encerra o pipeline com mensagem e código de erro próprios. Atividade Outlook ou Teams ligada à saída de falha alerta uma atividade específica.

Capacity Metrics app e throttling

O Microsoft Fabric Capacity Metrics app mostra o consumo de CU da capacidade e embasa a decisão de SKU. O smoothing distribui o consumo ao longo do tempo. Na primeira fase do throttling, depois de consumir 10 minutos de uso futuro, operações interativas novas recebem atrasos de 20 segundos; a recusa de operações em segundo plano é uma fase posterior.

Fila e admissão de jobs Spark

Só jobs em lote entram na fila, em ordem FIFO, e expiram após 24 horas. Notebooks interativos sem núcleos disponíveis recebem erro 430: cancele um job ativo no Monitoring hub ou tente depois. Com optimistic job admission, pedidos de escala dependem de núcleos livres; para computação previsível por job, desligue o autoscale e use nós fixos que caibam na capacidade.

V-Order e perfis de recurso

Três escopos: sessão (spark.sql.parquet.vorder.default), tabela (TBLPROPERTIES delta.parquet.vorder.enabled, padrão para qualquer sessão) e uma única gravação (opção parquet.vorder.enabled no write). A configuração antiga spark.sql.parquet.vorder.enable foi removida no runtime 1.3. Workspaces novos usam o perfil writeHeavy, com V-Order desligado; para tabelas lidas pelo Power BI, spark.fabric.resourceProfile = readHeavyForPBI.

OPTIMIZE, VACUUM e Optimized Write

OPTIMIZE compacta Parquet pequenos (e pode aplicar V-Order); VACUUM remove arquivos não referenciados mais antigos que a retenção, padrão de sete dias, reduzindo o time travel. Para VACUUM abaixo de sete dias, desligue spark.databricks.delta.retentionDurationCheck.enabled no ambiente. Sem código, use Maintenance no Lakehouse explorer. Optimized Write (spark.databricks.delta.optimizeWrite.enabled) gera arquivos de tamanho adequado sem repartition manual.

Particionamento e arquivos pequenos

Partição em disco deve usar coluna de cardinalidade moderada, como data, nunca um identificador único. Mais partições de shuffle ou repartition em memória não corrigem partição por ID. Tabela com incremental refresh do dataflow deve ter o dataflow como único gravador e sem OPTIMIZE.

Native execution engine

Acelera Parquet, Delta e CSV, mas não JSON, XML nem Structured Streaming. Para muitos itens, habilite na aba Acceleration do ambiente. Confirme no plano procurando nós com sufixo Transformer, NativeFileScan ou VeloxColumnarToRowExec (Photon é do Databricks).

Inicialização de sessões Spark

Starter pool parte em segundos; pool personalizado sem live pool leva cerca de três minutos; Quick mode instala bibliotecas na partida (30 segundos a 5 minutos); com Private Link, starter pools não são suportados e a partida leva de 2 a 5 minutos. Partida previsível com bibliotecas: custom live pool ativo na janela, com ambiente publicado em Full mode.

Desempenho do warehouse

Result set caching vem ligado em warehouses e SQL analytics endpoints, vale só para SELECT puro, é invalidado automaticamente e pode ser desligado por consulta com DISABLE_RESULT_SET_CACHE. O cache em memória e disco é automático, consistente e sem limpeza manual. Após grande carga, UPDATE STATISTICS em estatística de coluna única (só essas podem ser criadas manualmente). Filtros repetidos em coluna de alta cardinalidade: CLUSTER BY. Strings do Spark viram varchar(8000) no endpoint: crie colunas varchar(n).

Concorrência e erros no warehouse

O isolamento é sempre snapshot; SET TRANSACTION ISOLATION LEVEL é ignorado. O erro 24556 é conflito de escrita por tabela, mesmo em linhas diferentes: repita a transação ou escalone os processos. Erro de tempdb: estatísticas atualizadas e menos colunas em GROUP BY e ORDER BY, pois o tempdb não é configurável. KILL usa o session_id e só Admin do workspace executa. Falha de conexão transitória: espere de 5 a 10 minutos e reconecte.

Query insights

Visões do warehouse e do SQL analytics endpoint (não de Spark): long_running_queries para as mais demoradas por formato, frequently_run_queries para as frequentes, exec_sessions_history para sessões concluídas. query_hash agrupa a mesma consulta com valores diferentes no filtro; distributed_statement_id é único por execução; data_scanned_remote_storage_mb maior que zero explica a primeira execução lenta.

Políticas do eventhouse

Caching policy define o que fica quente em SSD; retention policy define até quando o dado existe. Dado fora do cache e dentro da retenção continua consultável, só mais lento; política de cache nula na tabela herda a do banco.
query_datascope='hotcache' restringe ao cache quente. Partitioning policy por hash é para filtros frequentes em string ou guid de alta cardinalidade, como TenantId. Capacity Planner com capacidade mínima evita latência ao reativar.

Otimização de consultas KQL e materialized views

materialize() evita recalcular um let usado várias vezes; has busca termo inteiro pelo índice e é mais barato que contains ou regex; lookup é indicado para tabela grande com dimensão pequena do lado direito. Consultar a materialized view pelo nome junta a parte materializada com o delta e sempre retorna resultado atualizado; materialized_view() lê só a parte materializada, mais rápida. A retenção da view não afeta a tabela de origem.

Métricas e limites do eventstream

Backlogged input events crescendo = processamento atrás da entrada; Deserialization errors = entrada ilegível; erros de conversão = saída incompatível com o esquema do destino. Acima de 100 MB/s, ajuste Event throughput setting para High. Eventos acima de 1 MB não passam; consumidor parado por mais que a retenção (padrão de um dia) perde eventos.

Dataflow Gen2: monitoramento e erros

O histórico de atualizações mostra até 50 execuções ou seis meses; Download detailed logs (motor Mashup) fica disponível por 28 dias nos Recent runs. Exclamação vermelha por validação: abra no editor e valide. Origem sobrecarregada: reduza avaliações paralelas na aba Scale. Require fast copy faz a atualização falhar se fast copy não puder ser usado. Parâmetro obrigatório impede o agendamento: execute pela atividade Dataflow do pipeline ou pela API.

Macetes para memorizar

💡 V-Order em três escopos

Sessão = vorder.default; Tabela = TBLPROPERTIES; Uma escrita = opção no write.

💡 Cache x retenção

Cache decide a velocidade; retenção decide se o dado ainda existe.

💡 OPTIMIZE x VACUUM

OPTIMIZE junta; VACUUM varre (sete dias).

💡 Erros do eventstream

Backlog = fila atrasada; Deserialização = entrada ilegível; Conversão = saída não cabe no destino.

💡 Pipeline: números de atividade

Retry a cada 30 s, Timeout de 12 h, teto de 7 dias.

💡 Spark: fila

Lote espera (FIFO, 24 h); interativo não espera (erro 430).

Onde olhar

Capacidade = Capacity Metrics; T-SQL = queryinsights; eventstream = Data insights; jobs = Monitoring hub.

Tabela de memorização: não confunda

Quando é...	...e quando é
Caching policy Decidir quanto do dado fica em SSD para consultas rápidas.	**Retention policy** Decidir até quando o dado existe e pode ser consultado.
OPTIMIZE Compactar arquivos pequenos e aplicar V-Order.	**VACUUM** Remover arquivos não referenciados mais antigos que a retenção (padrão de sete dias).
Consultar a view pelo nome dela Resultado sempre atualizado (materializado mais delta).	**Chamar materialized_view() (função)** Mais velocidade aceitando algum atraso, lendo só a parte materializada.
Activator com job events Alertar sucesso e falha, inclusive de execuções manuais, sem mexer no pipeline.	**Failure notifications do agendamento** Só falhas de execuções agendadas de um pipeline, por e-mail.
Partial Batch no Semantic model refresh Atualizar tabelas e partições escolhidas confirmando em lotes.	**Transactional** Confirmar tudo de uma vez.
Rerun a partir da atividade com falha Reprocessar só o que falhou sem mudar a definição do pipeline.	**Deactivate** Pular temporariamente uma atividade sem apagá-la.
Capacity Metrics app Consumo de CU da capacidade inteira e decisão de SKU.	**Query insights** Histórico e desempenho de consultas T-SQL de um warehouse ou endpoint.

Números que caem

Item	Valor
Atividade de pipeline: intervalo padrão de Retry	**30 segundos**
Atividade de pipeline: Timeout	**padrão de 12 horas; máximo de sete dias**
Fila de jobs Spark em lote	**expira em 24 horas**

Primeira fase do throttling	**após 10 minutos de uso futuro, atrasos de 20 segundos em operações interativas**
Retenção padrão do VACUUM	**sete dias**
Workspace monitoring	**retenção padrão de 30 dias, sem backfill**
Histórico de atualizações do Dataflow Gen2	**até 50 execuções ou seis meses**
Logs detalhados do Dataflow Gen2	**disponíveis por 28 dias**
.show ingestion failures	**falhas de comandos de ingestão dos últimos 14 dias**
Eventstream com throughput High	**acima de 100 MB/s**
Falha de conexão transitória no warehouse	**esperar de 5 a 10 minutos e reconectar**
Pool personalizado sem live pool	**cerca de três minutos para iniciar**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Usar queryinsights para investigar um notebook Spark ou Data insights para um pipeline.
Como não cair: Query insights é só T-SQL; Data insights é só eventstream; Spark tem monitoramento contextual na célula e o Monitoring hub.

⚠ **Alerta de prova:** Aumentar a caching policy para manter dados que estão sumindo da tabela.
Como não cair: Cache não amplia o período consultável; quem apaga é a retention policy.

⚠ **Alerta de prova:** Achar que VACUUM compacta arquivos ou aplica V-Order.
Como não cair: Compactar e V-Order são do OPTIMIZE; VACUUM só limpa arquivos antigos.

⚠ **Alerta de prova:** Supor que o V-Order já vem ligado em workspaces novos.
Como não cair: Workspaces novos usam writeHeavy, com V-Order desligado; para leitura pelo Power BI, readHeavyForPBI.

⚠ **Alerta de prova:** Tratar o erro 430 de notebook interativo como throttling ou falta de permissão.
Como não cair: Interativos não entram na fila: o erro é limite de computação; cancele um job ativo ou tente depois.

⚠ **Alerta de prova:** Mudar o nível de isolamento para READ COMMITTED para resolver conflito no warehouse.
Como não cair: O warehouse ignora o SET e usa snapshot; o conflito é por tabela, então repita ou escalone.

⚠ **Alerta de prova:** Criar CREATE STATISTICS com várias colunas ou rodar DBCC SHOW_STATISTICS para recalcular.
Como não cair: Manualmente só estatísticas de coluna única; SHOW_STATISTICS apenas exibe; atualize com UPDATE STATISTICS.

⚠ **Alerta de prova:** Aumentar o tamanho do tempdb para corrigir erro de espaço.
Como não cair: tempdb não é configurável; atualize estatísticas e reduza GROUP BY e ORDER BY.

⚠ **Alerta de prova:** Usar Retry para repetir uma atividade que deu certo ou Timeout zero para pular etapa.
Como não cair: Retry só repete após falha; para pular, Deactivate.

⚠ **Alerta de prova:** Esperar que o workspace monitoring mostre execuções de antes da ativação.
Como não cair: Não há backfill; os dados começam na ativação e ficam 30 dias por padrão.

⚠ **Alerta de prova:** Achar que a retenção definida na materialized view apaga dados da tabela de origem.
Como não cair: A retenção da view vale só para a view.

⚠ **Alerta de prova:** Particionar por id_evento e repartition para resolver arquivos pequenos.
Como não cair: Partição em disco por cardinalidade moderada, como data; ID único fragmenta sempre.

⚠ **Alerta de prova:** Trocar Low por High ou aumentar a retenção para recuperar eventos expirados no eventstream.
Como não cair: Throughput define vazão; retenção define tempo de guarda; eventos expirados não voltam.

⚠ **Alerta de prova:** Contornar rótulo restritivo no COPY INTO trocando para BULK INSERT ou JSON.
Como não cair: BULK INSERT corresponde ao COPY INTO e o formato do arquivo não muda a restrição; remova do destino o rótulo que causa a falha e repita o comando.

Como cai na prova

📝 **Como cai:** Sintoma de falha ou lentidão e quatro ferramentas de monitoramento.
Como resolver: Identifique o motor do sintoma (Spark, T-SQL, KQL, eventstream, pipeline, capacidade) e descarte as ferramentas de outros motores; a sobrevivente é a que olha exatamente esse escopo.

📝 **Como cai:** Alerta com restrição 'sem alterar os pipelines' ou 'uma regra só'.
Como resolver: Atividade Outlook/Teams altera o pipeline e Failure notifications só cobre agendadas; com essas restrições a resposta é Activator com job events ou com ItemJobEventLogs.

📝 **Como cai:** Ajuste de desempenho com várias configurações Spark plausíveis.
Como resolver: Confira o escopo pedido (uma gravação, uma tabela, a sessão, vários itens) e o efeito real: V-Order otimiza leitura, Optimized Write ajusta tamanho, native engine acelera Parquet/Delta/CSV.

📝 **Como cai:** Erro de warehouse com número ou mensagem (24556, tempdb, conexão).
Como resolver: Lembre que isolamento é snapshot fixo, tempdb não é configurável e falhas transitórias pedem espera e reconexão; descarte opções que mudam configurações inexistentes.

📝 **Como cai:** Pergunta sobre eventhouse com cache, retenção e partição.
Como resolver: Separe o objetivo: velocidade (caching e hotcache), existência do dado (retention), filtro por string de alta cardinalidade (partitioning hash), reativação (Capacity Planner).

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
Quem disparou qual execução	**Upstream run e Downstream runs no Monitoring hub**
Comparar durações e sobreposições de execuções	**Visão Gantt**
Um alerta único para falhas de todos os pipelines do workspace	**KQL em ItemJobEventLogs mais regra do Activator**
Alerta por dispositivo, inclusive por falta de dados	**Objetos e regras de estado no Activator**
Segredo aparecendo na saída da atividade	**Secure output**
Falhar o pipeline de propósito com código próprio	**Fail activity**
Atualizar o modelo semântico depois da carga, antes do e-mail	**Semantic model refresh com Wait on completion ligado**
Erro de comando de ingestão nos últimos 14 dias	**.show ingestion failures**
Mesma consulta com valores diferentes no filtro	**Agrupar por query_hash**

Primeira execução lenta, as seguintes rápidas	**data_scanned_remote_storage_mb maior que zero**
Filtro frequente por string de alta cardinalidade no eventhouse	**Partitioning policy por hash**
Filtros repetidos em coluna de alta cardinalidade no warehouse	**CLUSTER BY**
Arquivos pequenos sem escrever código	**Maintenance no Lakehouse explorer (OPTIMIZE com V-order)**
Atalho Delta aparece como pasta / atalho KQL falha na criação	**Espaço no nome do atalho / coluna com asterisco no nome**

🎯 Se você lembrar de só uma coisa deste capítulo...

- Query insights é só T-SQL; Data insights é só eventstream.
- Cache não estende retenção; retenção da materialized view não mexe na origem.
- V-Order vem desligado em workspaces novos (writeHeavy).
- Isolamento do warehouse é sempre snapshot; conflito de escrita é por tabela.
- Notebook interativo não entra na fila; jobs em lote esperam até 24 horas.

Essenciais para a prova (checklist)

- ☐ Sei o que o Monitoring hub mostra e como fazer Rerun a partir da atividade com falha.
- ☐ Sei montar alertas com Activator (job events, ItemJobEventLogs, objetos e regras de estado).
- ☐ Sei as configurações de atividade: Retry, Timeout, Secure input/output, Deactivate e Fail.
- ☐ Sei configurar V-Order por sessão, tabela e gravação e quando usar readHeavyForPBI.
- ☐ Sei a diferença entre OPTIMIZE, VACUUM e Optimized Write.
- ☐ Sei usar result set caching, estatísticas, CLUSTER BY e varchar(n) no warehouse.
- ☐ Sei interpretar erros do warehouse (24556, tempdb, conexão) e quem pode usar KILL.
- ☐ Sei diferenciar caching, retention e partitioning policy no eventhouse.
- ☐ Sei ler as métricas do eventstream e quando usar throughput High.
- ☐ Sei diagnosticar Dataflow Gen2 (histórico, logs detalhados, aba Scale, Require fast copy).

Questões de revisão do capítulo

R3.1 · Monitorar e otimizar uma solução de análise

Um engenheiro da Indústria Têxtil Algodão Bom tentou, pelo portal, rodar VACUUM em uma tabela Delta com retenção de dois dias, para liberar espaço, e a operação falhou. Ele aceita perder o histórico de time travel. O que precisa ser configurado?

A. O papel Admin no workspace para liberar retenções menores que sete dias
B. spark.sql.parquet.vorder.default = true no ambiente usado pelo workspace inteiro
C. Apply V-order marcado junto com o VACUUM no Run maintenance commands
D. spark.databricks.delta.retentionDurationCheck.enabled = false no ambiente

R3.2 · Monitorar e otimizar uma solução de análise

Na Indústria de Bebidas Gelada, um engenheiro criou na pasta Tables de um lakehouse o atalho Vendas Mensais, apontando para uma tabela Delta válida no ADLS Gen2, mas ele não aparece como tabela, e sim como pasta não identificada. Qual é a causa?

A. O nome do atalho tem espaço, o que impede o reconhecimento como tabela Delta
B. A tabela de destino precisa ter V-Order habilitado para ser reconhecida pelo lakehouse
C. Atalhos em Tables só são reconhecidos como tabela quando o destino está no OneLake
D. O atalho precisa ser recriado em Files para que a descoberta de tabelas aconteça

R3.3 · Monitorar e otimizar uma solução de análise

A capacidade da Rede de Ensino Futuro Brilhante passou de 100% de uso durante um pico de cargas. Os usuários contam que consultas interativas ficaram mais lentas, mas não foram recusadas. Segundo a política de throttling, o que explica esse primeiro efeito?

A. Após consumir 10 minutos de uso futuro, operações interativas novas recebem atrasos de 20 segundos
B. A capacidade pausa sozinha os jobs agendados e retoma depois que o uso volta para menos de 50%
C. O smoothing distribui as consultas interativas por 24 horas e só depois volta a executá-las
D. O Fabric recusa todas as operações em segundo plano assim que a capacidade passa de 100% do uso contratado

R3.4 · Monitorar e otimizar uma solução de análise

Na Rede de Entregas Leva e Traz, um aplicativo que consome um eventstream por custom endpoint ficou fora do ar por três dias. Ao voltar, os eventos desse período não estavam mais disponíveis para leitura. O eventstream usa as configurações padrão. O que explica e como evitar?

A. O custom endpoint não guarda eventos; troque por um lakehouse
B. A retenção padrão é de um dia; aumente a retenção do eventstream
C. A entrega é exatamente uma vez; reconfigure o consumidor do aplicativo
D. O throughput Low descarta eventos; mude o nível para High

Respostas comentadas

R3.1. D) spark.databricks.delta.retentionDurationCheck.enabled = false no ambiente

Por quê: VACUUM remove arquivos não referenciados mais antigos que a retenção (padrão de sete dias); retenções curtas reduzem o time travel e podem afetar leitores e gravadores concorrentes.

✗ A) Incorreta. A verificação de retenção não depende do papel no workspace.

✗ B) Incorreta. Essa configuração trata do V-Order.

✗ C) Incorreta. Apply V-order acompanha o OPTIMIZE e não altera a verificação de retenção.

> **Dica de prova:** VACUUM com menos de sete dias: desligue retentionDurationCheck no ambiente.

R3.2. A) O nome do atalho tem espaço, o que impede o reconhecimento como tabela Delta

Por quê: Outras restrições de atalhos: nomes e caminhos sem os caracteres % e +, sem caracteres não latinos, até 10 atalhos por caminho e até 5 níveis de atalho para atalho.

✗ B) Incorreta. V-Order é otimização de leitura, não requisito de reconhecimento.

✗ C) Incorreta. Atalhos em Tables para Delta externo, como ADLS, também são reconhecidos como tabelas.

✗ D) Incorreta. Em Files não há descoberta de tabelas.

> **Dica de prova:** Atalho Delta aparecendo como pasta: verifique espaço no nome.

R3.3. A) Após consumir 10 minutos de uso futuro, operações interativas novas recebem atrasos de 20 segundos

Por quê: Smoothing espalha o uso de CU por pontos de tempo futuros; quando a capacidade esgota o uso dos próximos 10 minutos, começam os atrasos interativos, seguidos de recusas em fases posteriores.

✗ B) Incorreta. A capacidade não pausa jobs sozinha nessa situação.

✗ C) Incorreta. O smoothing distribui o consumo de CU ao longo do tempo para reduzir throttling; não adia a execução das consultas.

✗ D) Incorreta. A recusa de operações em segundo plano é uma fase posterior, não a primeira reação.

> **Dica de prova:** Primeira fase do throttling: atrasos de 20 segundos em operações interativas.

R3.4. B) A retenção padrão é de um dia; aumente a retenção do eventstream

Por quê: Nas configurações do eventstream há retenção, throughput, endosso e rótulo de confidencialidade. Eventos não podem ser excluídos explicitamente; expiram pela retenção.

✗ A) Incorreta. O consumo por custom endpoint lê eventos retidos no eventstream; a perda veio da retenção.

✗ C) Incorreta. A garantia de entrega é at least once, e o problema é a expiração por retenção.

✗ D) Incorreta. O nível de throughput escala o desempenho; não define por quanto tempo os eventos ficam retidos.

> **Dica de prova:** Consumidor parado por dias perdeu eventos: verifique a retenção (padrão de um dia).

DIA DA PROVA

Estratégia de prova

O DP-700 é uma prova de cenário: cada enunciado descreve uma empresa, um requisito e restrições ('sem código', 'sem copiar dados', 'com o menor privilégio', 'sem alterar os pipelines') e oferece quatro opções que existem de verdade no Fabric. A resposta certa é a única que atende a todas as restrições; as erradas costumam funcionar parcialmente, dar permissão demais ou pertencer a outro motor (Spark, T-SQL, KQL). Há muito código para ler (PySpark, T-SQL, KQL, expressões de pipeline), em que a diferença está num parâmetro invertido ou num operador vizinho. Os três domínios têm o mesmo peso (30-35% cada), então não dá para compensar um ponto fraco. A prova usa os nomes atuais dos produtos (Apache Airflow job, Microsoft Entra ID, Eventhouse, OneLake security); nomes aposentados aparecem só como distrator.

Como responder

✓ Planeje o tempo: são 100 minutos de prova, e a Microsoft não divulga o número exato de questões (a maioria dos exames tem entre 40 e 60), então reserve em torno de 2 minutos por item; nota mínima de 700 numa escala de 1 a 1000.

✓ Leia primeiro a pergunta final e depois o cenário, marcando cada restrição; elimine a opção que viola qualquer uma delas antes de comparar as restantes.

✓ Em múltipla resposta (Escolha DUAS ou TRÊS), julgue cada afirmação isoladamente como verdadeira ou falsa; não procure 'combinações' que façam sentido juntas.

✓ Em código, compare as opções linha a linha: ordem de parâmetros, operador (bin x make-series, arg_max x arg_min, has x contains) e cláusulas do MERGE costumam ser a única diferença.

✓ Em permissões, aplique o menor privilégio: Admin, CONTROL e Storage Blob Data Owner quase sempre são distratores quando o requisito pede o mínimo necessário.

✓ Use a marcação para revisão nas dúvidas, mas atenção: depois de iniciar uma pausa (há 5 minutos embutidos para isso, e o relógio continua correndo), não dá para voltar a nenhum item já visto.

✓ Em exames baseados em função como este, o botão do Microsoft Learn fica disponível durante a prova; use só para confirmar um detalhe pontual, pois consome o mesmo relógio.

✓ Distribua o estudo igualmente entre os três domínios, já que todos têm a mesma faixa de peso.

Erros que derrubam candidatos

✗ Escolher a opção que funciona, mas ignora uma restrição do enunciado (copia dados quando era para referenciar, exige código quando era sem código, cria pipeline quando era para evitar).

✗ Confundir ferramentas de monitoramento entre motores: queryinsights para Spark, Data insights para pipeline, Capacity Metrics para logs de execução.

✗ Tratar o SQL analytics endpoint como gravável e o warehouse como se aceitasse IDENTITY com seed, CREATE USER ou mudança de isolamento.

✗ Confundir caching policy com retention policy no eventhouse e OPTIMIZE com VACUUM no lakehouse.

✗ Dar permissão demais (Admin, CONTROL, Contributor) quando o cenário pede só leitura ou só gestão de máscara.

✗ Criar regras de implantação no estágio de desenvolvimento ou esperar que a implantação copie dados.

✗ Responder com nomes antigos (Azure AD, Data workflows, Datamart) que hoje só aparecem como distrator.

Na semana da prova

- ☐ Revise os itens que você errou no treino e reescreva, com suas palavras, por que cada alternativa errada não atende.
- ☐ Revise as tabelas mentais de decisão: armazenamento (lakehouse, warehouse, eventhouse, SQL database), ingestão (Copy job, mirroring, atalho, eventstream) e orquestração (Dataflow Gen2, Copy, notebook, Airflow).
- ☐ Treine leitura de código: KQL (summarize, bin, make-series, arg_max, mv-expand), MERGE do Delta e T-SQL de carga no warehouse.
- ☐ Revise os números que caem: Retry de 30 segundos, Timeout de 12 horas, VACUUM de sete dias, eventstream de 1 MB e retenção padrão de um dia, fila Spark de 24 horas.
- ☐ Na véspera, revise só os 'lembrar' de cada capítulo e as pegadinhas de permissões; durma bem.

☐ Antes do dia, use o exam sandbox para conhecer a interface (marcação para revisão, tempo e questões restantes) e lembre que pausas impedem voltar aos itens já vistos.

REVISÃO FINAL

Folha de cola

Leia na véspera e na manhã da prova: os gatilhos e as pegadinhas mais importantes de cada domínio.

1. Implementar e gerenciar uma solução de análise

‣ Vários notebooks com dependências, paralelismo e repetição → **notebookutils.notebook.runMultiple com DAG**

‣ Disparar o pipeline quando um arquivo chega → **Gatilho de evento de armazenamento com filtro no Subject**

‣ Esperar uma condição externa com repetição e limite de tempo → **Atividade Until com Lookup e Wait**

‣ Mesmo pipeline com valores diferentes por horário → **Vários agendamentos com parâmetros (nome idêntico ao parâmetro)**

‣ Valor diferente por estágio sem regras de implantação → **Variable library com value set ativo por workspace**

⚠ Regras não existem em Development; são criadas no estágio de destino por quem é dono do item.

⚠ Direct Lake é implantado normalmente; o vínculo é ajustado com regra de fonte de dados.

⚠ Papéis com escrita ignoram as restrições; o usuário precisa sair da DefaultReader e ficar só na função restrita.

2. Ingerir e transformar dados

‣ Assistente sem pipeline, carga inicial completa e depois só mudanças → **Copy job com Incremental copy**

‣ Sistema sem conector de mirroring, mas a aplicação exporta mudanças → **Open mirroring com landing zone**

‣ Réplica contínua de Azure SQL no OneLake, sem ETL, para Spark e Power BI → **Database mirroring (endpoint somente leitura)**

‣ Usar banco do Azure Data Explorer no Fabric sem reingestão → **Database shortcut (follower)**

‣ Mudanças de linhas de banco em tempo real, sem código, para um eventhouse → **Fonte CDC do eventstream**

⚠ O endpoint é somente leitura sempre; escrita em T-SQL é no warehouse, e no lakehouse se grava com Spark.

⚠ Marca d'água não vê linhas apagadas; exclusões exigem CDC.

⚠ COPY INTO não transforma; filtro e conversão vão em INSERT...SELECT FROM OPENROWSET.

3. Monitorar e otimizar uma solução de análise

‣ Quem disparou qual execução → **Upstream run e Downstream runs no Monitoring hub**

‣ Comparar durações e sobreposições de execuções → **Visão Gantt**

‣ Um alerta único para falhas de todos os pipelines do workspace → **KQL em ItemJobEventLogs mais regra do Activator**

‣ Alerta por dispositivo, inclusive por falta de dados → **Objetos e regras de estado no Activator**

‣ Segredo aparecendo na saída da atividade → **Secure output**

⚠ Query insights é só T-SQL; Data insights é só eventstream; Spark tem monitoramento contextual na célula e o Monitoring hub.

⚠ Cache não amplia o período consultável; quem apaga é a retention policy.

⚠ Compactar e V-Order são do OPTIMIZE; VACUUM só limpa arquivos antigos.

GABARITO

Respostas do diagnóstico

D1. C) Extensões SQL Database Projects e mssql, projeto SDK-style e plataforma Synapse Data Warehouse in Microsoft Fabric → revise o capítulo 1

Por quê: Projetos de banco de dados (SQL Database Projects) permitem desenvolver o warehouse como código: criar o projeto do zero ou a partir do warehouse existente, compilar para .dacpac e publicar.

✗ A) Incorreta. Power Query SDK e PBIP não geram .dacpac para esquema de warehouse.

✗ B) Incorreta. Modelos ARM implantam recursos do Azure, não objetos T-SQL de um warehouse do Fabric.

✗ D) Incorreta. Notebooks não são projetos de banco de dados e o SQL analytics endpoint não é alvo de publicação de esquema.

> **Dica de prova:** Esquema do warehouse como código com .dacpac: SQL Database Projects, SDK-style.

D2. A) Escolher um novo lakehouse de diagnóstico dentro da nova capacidade → revise o capítulo 1

Por quê: O lakehouse de diagnóstico precisa estar na mesma capacidade dos workspaces monitorados; se ele for excluído, os workspaces que apontam para ele desabilitam o diagnóstico.

✗ B) Incorreta. A capacidade antiga não tem relação com a gravação após a mudança.

✗ C) Incorreta. A imutabilidade protege os arquivos; não reativa o diagnóstico.

✗ D) Incorreta. O diagnostics não volta sozinho após a troca de capacidade.

> **Dica de prova:** Mudou de capacidade: escolha um novo lakehouse de diagnóstico na nova capacidade.

D3. D) Com a opção desligada, o pool padrão e sua computação valem para os ambientes do workspace → revise o capítulo 1

Por quê: A configuração Customize compute configurations for items, nas configurações de Spark do workspace, permite que itens como notebooks ajustem núcleos e memória de driver e executor por meio do ambiente. Desligada, o pool padrão prevalece.

✗ A) Incorreta. Quick mode trata da publicação de bibliotecas; não libera ajustes de computação bloqueados pelo admin.

✗ B) Incorreta. Pool de nó único é só uma opção de tamanho mínimo; não é requisito para ajustar memória.

✗ C) Incorreta. Ao usar ambiente de outro workspace, a configuração de computação dele é ignorada e vale a do workspace atual.

> **Dica de prova:** Ajuste de computação no ambiente ignorado: verifique Customize compute configurations for items no workspace.

D4. C) Leituras | make-series Total=count() default=0 on Instante from ago(1d) to now() step 1h by Sensor → revise o capítulo 2

Por quê: make-series agrega valores ao longo de um eixo, preenchendo intervalos ausentes com o valor default (0, se não especificado); o resultado é um array por grupo.

✗ A) Incorreta. O eixo e o agrupamento estão invertidos: o eixo deve ser a coluna de tempo.
✗ B) Incorreta. summarize com bin não gera linhas para horas sem leitura, então não aparecem zeros.
✗ D) Incorreta. dcount de Instante e range não produzem a contagem por hora.

> **Dica de prova:** Série temporal com buracos preenchidos no KQL: make-series com default.

D5. B) O join padrão é innerunique, que deduplica as chaves da tabela da esquerda → revise o capítulo 2

Por quê: Os sabores de join do KQL incluem innerunique (padrão), inner, leftouter, rightouter, fullouter, leftanti, rightanti, leftsemi e rightsemi.

✗ A) Incorreta. leftsemi não é o padrão.
✗ C) Incorreta. leftanti não é o padrão.
✗ D) Incorreta. inner não é o padrão e mantém as colunas das duas tabelas.

> **Dica de prova:** Join no KQL sumiu com linhas repetidas da esquerda: o padrão é innerunique.

D6. B) O SQL analytics endpoint é somente leitura; a gravação deve ser feita pelo Spark ou em um warehouse → revise o capítulo 2

Por quê: Warehouse oferece T-SQL de leitura e escrita com transações multitabela; o SQL analytics endpoint do lakehouse é gerado pelo sistema e somente leitura. Dados do lakehouse são gravados com Spark, pipelines ou dataflows.

✗ A) Incorreta. ReadData permite ler pelo T-SQL; nenhuma permissão torna o endpoint gravável.
✗ C) Incorreta. A sincronização de metadados expõe as tabelas para consulta; não libera escrita.
✗ D) Incorreta. Transações explícitas não mudam o fato de o endpoint ser somente leitura.

> **Dica de prova:** INSERT, UPDATE ou DELETE em T-SQL: warehouse; SQL analytics endpoint só lê.

D7. B) Ele sai da fila depois de 24 horas e precisa ser reenviado → revise o capítulo 3

Por quê: Jobs de pipelines, agendador e Spark job definitions entram em fila FIFO e são repetidos automaticamente quando há núcleos. Notebooks interativos não entram na fila, e capacidade em throttling rejeita novos jobs.

✗ A) Incorreta. Jobs em lote não são convertidos em sessões interativas.
✗ C) Incorreta. O burst tem limite por SKU; sem núcleos disponíveis, jobs em lote entram na fila.
✗ D) Incorreta. A fila tem expiração de 24 horas.

> **Dica de prova:** Fila de Spark no Fabric: só jobs em lote, FIFO, expira em 24 horas.

D8. D) Selecionar a tabela e a partição desejadas e usar o modo Partial Batch → revise o capítulo 3

Por quê: Configurações avançadas da atividade: Wait on completion, Max parallelism, Retry Count, tempo limite e modo de confirmação (Transactional ou Partial Batch); tabelas e partições podem vir de parâmetros.

✗ A) Incorreta. O modo Transactional confirma tudo de uma vez, contrariando o requisito.
✗ B) Incorreta. Desligar a espera não restringe o escopo nem muda a forma de confirmação.
✗ C) Incorreta. Por padrão a atividade faz atualização completa, e Transactional confirma tudo de uma vez.

> **Dica de prova:** Refresh parcial e confirmação em lotes: tabelas e partições escolhidas com Partial Batch.

D9. D) Renomear a coluna para remover o asterisco antes de criar o atalho → revise o capítulo 3

Por quê: No banco KQL, cada atalho conecta uma subpasta ou tabela, não pode ser renomeado e é criado um por vez; as colunas Delta são mapeadas para tipos do Eventhouse.

✗ A) Incorreta. Atalhos em lakehouse não são o mesmo que atalhos no banco KQL e não resolvem a criação pedida.
✗ B) Incorreta. A restrição vale com ou sem aceleração.
✗ C) Incorreta. A caching policy não interfere nos nomes de colunas.

> **Dica de prova:** Atalho KQL falhando por nome de coluna com asterisco: renomeie a coluna.

Agora, treine no formato real

Faça as 6 provas completas do curso DP-700: 300 questões inéditas, tempo do exame, explicação de cada alternativa e link da documentação para revisar.

Ver as 6 provas completas e o cupom do mês →

Até o dia da prova, uma revisão por dia no seu feed

Instagram @strecht.simulados

Um post de prova por dia: questão comentada, pegadinhas e o "não confunda" de cada certificação.

Seguir no Instagram →

YouTube Strecht Simulados

Shorts diários com pegadinhas de prova e, aos sábados, um simulado comentado.

Inscrever-se no canal →

Material independente de preparação, sem afiliação ou endosso de Microsoft. As questões são inéditas e não reproduzem a prova real.