Strecht Simulados · Guia de preparação




# Guia de preparação DP-600

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-600: 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 modelos semânticos | 25-30% | 18 conceitos · 14 pegadinhas |
| 2. Manter uma solução de análise de dados | 25-30% | 17 conceitos · 14 pegadinhas |
| 3. Preparar dados | 45-50% | 20 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 modelos semânticos

**Na Cooperativa de Café Grão Nobre, uma tabela por mês precisa mostrar a média móvel de três meses (o mês da linha e os dois anteriores) da medida [Sacas]. A tabela Data tem a coluna Data[AnoMes]. Qual medida atende?**

A. AVERAGEX(WINDOW(0, REL, 2, REL, ALLSELECTED(Data[AnoMes]), ORDERBY(Data[AnoMes], ASC)), [Sacas])
B. AVERAGEX(WINDOW(1, ABS, 3, ABS, ALLSELECTED(Data[AnoMes]), ORDERBY(Data[AnoMes])), [Sacas])
C. AVERAGEX(OFFSET(-2, ALLSELECTED(Data[AnoMes]), ORDERBY(Data[AnoMes])), [Sacas])
D. AVERAGEX(WINDOW(-2, REL, 0, REL, ALLSELECTED(Data[AnoMes]), ORDERBY(Data[AnoMes])), [Sacas])

D2 · Implementar e gerenciar modelos semânticos

**No modelo Import da Rede de Lojas Ponto Final, a coluna NumeroPedido tem valores como SO45001, únicos por linha, e ocupa grande parte do tamanho do modelo. O que a orientação de redução de dados para modelos Import sugere?**

A. Mover a coluna para uma dimensão própria com relação um-para-um com os pedidos
B. Manter o texto e criar uma coluna calculada DAX numérica a partir dele no modelo Import
C. Dividir a coluna em duas colunas de texto, uma com o prefixo e outra com o número
D. Remover o prefixo SO e converter a coluna para número inteiro, com Do Not Summarize

D3 · Implementar e gerenciar modelos semânticos

**A Bolsa de Mercadorias Agro Futuro tem um modelo semântico Import com atualização incremental na tabela Negocios, atualizado uma vez por noite em uma capacidade Fabric. O refresh noturno leva cerca de 40 minutos, e a fonte é um SQL Server corporativo que aceita query folding. Os traders querem ver no relatório também os negócios fechados no dia corrente, sem aumentar a frequência de refresh do modelo. O que deve ser configurado na política de atualização incremental?**

A. Get the latest data in real time with DirectQuery, criando uma partição DirectQuery após o período
B. Only refresh complete days isolado, para incluir o dia corrente nas partições de importação
C. Detect data changes, com a mesma coluna de data usada para particionar a tabela de negócios
D. Archive data starting de um dia, para que o histórico seja consultado ao vivo pela fonte

D4 · Manter uma solução de análise de dados

**Em um deployment pipeline da Editora Livro Aberto, o analista seleciona só o relatório Vendas Mensais no estágio Desenvolvimento e o implanta para Teste. O modelo semântico do qual o relatório depende ainda não existe no estágio Teste, e a implantação falha. Como evitar a falha?**

A. Usar Select related para incluir na implantação o modelo semântico do qual o relatório depende
B. Publicar o relatório direto no workspace de Teste pelo Desktop antes de implantar pelo pipeline
C. Ativar o autobinding entre workspaces nas configurações do pipeline antes de repetir a implantação
D. Criar no estágio Teste uma regra de parâmetro que aponte o relatório para o modelo de Desenvolvimento

D5 · Manter uma solução de análise de dados

**No lakehouse RH da Indústria Têxtil Linha Fina, a OneLake security tem a função Analistas, com CLS na tabela Funcionarios liberando todas as colunas exceto Salario. Um analista da função roda SELECT * FROM Funcionarios no SQL analytics endpoint e depois lê a mesma tabela com spark.read em um notebook. O que acontece em cada caso?**

A. As duas leituras dão erro, porque a CLS bloqueia a tabela inteira para a função
B. As duas leituras trazem as colunas permitidas e escondem Salario sem gerar erro
C. O SQL analytics endpoint traz Salario com nulos; o Spark retorna erro de permissão
D. O SELECT * dá erro no SQL analytics endpoint; no Spark, a leitura traz as colunas permitidas

D6 · Manter uma solução de análise de dados

**Na Consultoria Elo Global, um relatório usa RLS dinâmica com a regra [Email] = USERPRINCIPALNAME(). Um consultor externo, convidado B2B, diz que não vê dados, mas o modelador testou com Test as role selecionando a função e tudo pareceu correto. Por que o teste não revelou o problema e o que fazer?**

A. O Test as role só funciona em DirectQuery; trocar o modelo para Import e testar de novo
B. A regra precisa de USERNAME(), que o Test as role simula para usuários externos
C. O Test as role usa a identidade de quem testa; validar com a conta do convidado e o UPN em um cartão
D. Convidados B2B não podem ser filtrados por RLS; criar um workspace separado para eles

D7 · Preparar dados

**Uma analista da Rede de Lojas Varejo Amigo usa o visual query editor do warehouse e quer, na mesma ferramenta, apagar linhas antigas de uma tabela de staging. O que acontece?**

A. O editor visual executa só consultas de leitura; o DELETE deve ir para o editor SQL
B. O editor visual gera um DELETE ao remover linhas pelo filtro e grava a tabela
C. O editor visual aceita DDL e DML quando o warehouse está no modo de edição
D. O editor visual executa o DELETE se a consulta for salva na pasta Shared queries do warehouse

D8 · Preparar dados

**A Seguradora Proteja Mais ligou a OneLake integration de um modelo semântico Import que tem funções de RLS. Analistas com o papel Viewer no workspace e permissões Read e Build no modelo tentam ler, por um atalho em um lakehouse, as tabelas Delta exportadas pelo modelo, sem sucesso. Qual é o motivo?**

A. Com RLS ou OLS no modelo, só quem tem escrita, como Contributor, lê as tabelas exportadas
B. A OneLake integration exporta medidas e tabelas calculadas, mas não as tabelas Import
C. As tabelas só são exportadas depois que o modelo é convertido para o modo Direct Lake
D. Atalhos não podem apontar para pastas .SemanticModel de outro workspace no OneLake

D9 · Preparar dados

**No SQL query editor do warehouse, um engenheiro da Cooperativa Leite Puro executa BEGIN TRAN e um UPDATE e, em uma execução separada, roda COMMIT TRAN, que falha. O que explica o comportamento?**

A. O COMMIT exige papel Admin no workspace para encerrar transações abertas no editor
B. O warehouse não suporta transações explícitas; cada instrução é confirmada sozinha
C. A transação expira após um minuto sem atividade no editor e é desfeita em seguida
D. Cada Run é um lote em sessão própria; a transação deve estar inteira em uma única execução

01

CAPÍTULO 1 · 25-30% DA PROVA

# Implementar e gerenciar modelos semânticos

Este domínio vale 25-30% e mistura modelagem, DAX e otimização. O tema que mais aparece é Direct Lake: preciso distinguir Direct Lake on SQL de Direct Lake on OneLake (fallback, fontes, segurança, composição com Import, regras de implantação), entender framing e guardrails. Depois vêm DAX (variáveis, CALCULATE com KEEPFILTERS, ALLSELECTED, funções de janela, grupos de cálculo), relações (ponte, role-playing, bidirecional com moderação) e atualização incremental com RangeStart e RangeEnd. As alternativas erradas costumam inverter as características de on SQL e on OneLake ou usar uma função DAX que existe mas faz outra coisa. Meu erro clássico é confundir o que tem fallback e o que falha.

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

## Tópicos-chave

### Modos de armazenamento

Import guarda cópia VertiPaq comprimida, responde da memória e só vê dados novos após refresh; o modelo inteiro precisa caber na memória. DirectQuery traduz as consultas DAX para a fonte a cada interação. Dual responde do cache ou da fonte conforme a consulta. Direct Lake lê Delta do OneLake e carrega só as colunas usadas (transcoding). No Desktop, uma tabela convertida para Import não volta para DirectQuery.

### Direct Lake on SQL x on OneLake

On SQL depende do SQL analytics endpoint, usa uma única fonte, não combina com DirectQuery nem Dual no mesmo modelo, pode cair para DirectQuery (fallback) e suporta regras de implantação para religar a fonte. On OneLake lê Delta pelas APIs do OneLake, aceita várias fontes (lakehouse, warehouse, espelhados, SQL database, banco KQL com OneLake availability), integra-se à OneLake security, combina com Import pela web modeling e não tem fallback.

### Fallback e DirectLakeBehavior

Em Direct Lake on SQL a consulta cai para DirectQuery quando há RLS, OLS ou DDM no SQL analytics endpoint, view SQL sem materialização, tabela acima dos guardrails ou tabela sem framing. DirectLakeBehavior: Automatic (fallback silencioso, para produção), DirectLakeOnly (falha em vez de cair, para achar problemas) e DirectQueryOnly (para medir). EVALUATE TABLETRAITS() mostra o motivo na coluna DirectLakeFallbackInfo.

### Guardrails e requisitos do Direct Lake

Exige capacidade Fabric (F SKU); PPU ou Pro não bastam e é o modelo que precisa estar na capacidade. Guardrails por SKU limitam arquivos Parquet, row groups e linhas por tabela: on SQL cai para DirectQuery, on OneLake falha no refresh; OPTIMIZE e VACUUM reduzem arquivos. Modelo e fonte na mesma região, só conexões de nuvem (sem gateway on-premises nem VNet).

### Framing e atualização do Direct Lake

Framing é uma operação barata de metadados, disparada pelo refresh, que fixa a versão das tabelas Delta que o modelo enxerga. Keep your Direct Lake data up to date (ligado por padrão) faz framing quando as tabelas mudam; desligue e atualize na última etapa do pipeline para não expor dados no meio do ETL. Se um refresh falha de forma irrecuperável, as atualizações automáticas são suspensas até um refresh manual dar certo.

### Permissões do Direct Lake com SSO

O Direct Lake confere a identidade de quem consulta. Em on SQL bastam Read no item e SELECT nas tabelas pelo SQL analytics endpoint; em on OneLake, Read e ReadAll no item ou uma função da OneLake security. Segurança definida no SQL analytics endpoint não vale para on OneLake; para regras em todos os motores use OneLake security ou RLS no modelo. Para dispensar permissões dos usuários, use conexão de nuvem com identidade fixa.

### Modelos compostos

Misturam modos e fontes (por exemplo, planilha em Import e vendas em DirectQuery). Make changes to this model cria um modelo local com DirectQuery para um modelo publicado e exige Build nele. Há implicação de segurança: valores de uma fonte podem ir nas consultas enviadas a outra. Relações entre grupos de origem diferentes são limitadas: usam INNER JOIN, não criam linha em branco para órfãos e não permitem RELATED. Com fato em DirectQuery e dimensões Import da mesma fonte, defina as dimensões como Dual.

### Esquema estrela e relações

Dimensões filtram e agrupam, fatos resumem, ligados por um-para-muitos. Fatos não se relacionam diretamente: compartilham dimensões. Muitos-para-muitos entre dimensões: tabela ponte com um-para-muitos e uma relação bidirecional para o filtro chegar ao fato, ocultando a ponte e os IDs. Fato de grão maior (meta mensal): coluna com o primeiro dia do período ligada à tabela Data.

### Relações bidirecionais, inativas e role-playing

Use bidirecional com moderação: só em ponte muitos-para-muitos, slicers que mostram só opções com dados e análise dimensão a dimensão; quando só uma medida precisa, CROSSFILTER(..., Both) dentro do CALCULATE. Só uma relação fica ativa por caminho; a inativa é usada com USERELATIONSHIP. Se o usuário precisa filtrar livremente pelos dois papéis, crie uma tabela de datas por papel.

### Tabela de datas e Auto date/time

Mark as date table valida a coluna: tipo data, valores únicos, sem nulos, datas contíguas (sem remover fins de semana) e, em Date/Time, o mesmo horário. Auto date/time cria uma tabela oculta por coluna de data, só usa calendário civil e impede filtro de tempo comum a várias tabelas; em modelos sérios, desligue e use tabela própria.

### Variáveis, iteradores e CALCULATE

Variável é avaliada uma vez, no contexto em que é definida, e vira constante: CALCULATE não a reavalia. Iteradores (SUMX, AVERAGEX) calculam linha a linha; AVERAGEX(VALUES(Cliente[ID]), [Medida]) dá média por cliente. Filtro booleano no CALCULATE substitui o filtro da mesma coluna; KEEPFILTERS intersecta com o contexto. Prefira filtro booleano em coluna a FILTER sobre a tabela inteira.

### Funções DAX de uso frequente

ALL/REMOVEFILTERS ignoram todos os filtros da coluna (total geral); ALLSELECTED respeita slicers externos (percentual do total visível). DIVIDE trata denominador zero ou BLANK (use / quando nunca for zero; evite IFERROR). SELECTEDVALUE(coluna, alternativo) substitui IF(HASONEVALUE(), VALUES()). TREATAS cria relação virtual sem relação física. RANKX(ALL(Produto[Nome]), [Vendas]) ranqueia ignorando o filtro do próprio item.

### Funções de janela e de informação

WINDOW devolve intervalo (início e fim relativos REL ou absolutos ABS), OFFSET uma linha deslocada (-1 é o anterior) e INDEX uma posição absoluta, todas com ORDERBY e PARTITIONBY opcional. RANK aceita ties DENSE ou SKIP; ROWNUMBER nunca repete. ISINSCOPE diz em que nível da hierarquia a linha está; ISFILTERED também fica verdadeiro com slicer; HASONEVALUE é verdadeiro no subtotal de um nível acima.

### Grupos de cálculo

Itens usam SELECTEDMEASURE (sem argumentos), SELECTEDMEASURENAME e ISSELECTEDMEASURE para aplicar só a certas medidas. Funcionam só em medidas explícitas; ao criar o grupo o Power BI liga Discourage implicit measures. Precedence ordena grupos não comutativos (moeda e YTD). Medidas viram variant: proteja medidas de texto com IF(ISNUMERIC(SELECTEDMEASURE()), ...). Cada item pode ter Dynamic format string. Aplica-se um item numa medida com CALCULATE e filtro na coluna do grupo.

### Dynamic format strings, field parameters e visual calculations

Dynamic format string em medida (Measure tools, Format, Dynamic) muda o formato por contexto mantendo o valor numérico, ao contrário de FORMAT, que devolve texto. Field parameters (Modeling, New parameter, Fields) deixam o leitor trocar medidas ou colunas; não aceitam medidas implícitas nem visuais de IA como Q&A. Visual calculations (RUNNINGSUM, MOVINGAVERAGE, PREVIOUS) operam no visual, sem criar objeto no modelo, e não têm RELATED nem USERELATIONSHIP.

### Large semantic model storage format

Permite passar do limite padrão até o tamanho da capacidade ou o máximo do administrador; o upload pelo Desktop continua limitado a 10 GB (publique menor e carregue o histórico no serviço). Melhora a escrita via XMLA e traz on-demand load (páginas carregadas sob demanda após despejo). Vale para Import, não para modelos de push. Liga-se no slider do modelo, no Default storage format do workspace ou por PowerShell.

### Atualização incremental

Parâmetros RangeStart e RangeEnd com esses nomes exatos, tipo date/time, em filtro manual que dobre para a fonte (sem folding, tudo é lido e filtrado localmente). Política: período de arquivo (quanto histórico guardar) e período incremental (o que atualizar), além de opcionais: partição DirectQuery em tempo real (tabela híbrida), Only refresh complete days e Detect data changes com coluna de auditoria diferente da de partição. Chave de data inteira yyyymmdd: converta os parâmetros dentro do filtro.

### Desempenho de modelos e relatórios

Performance analyzer divide o tempo do visual em DAX query, Direct query, Visual display e Other. Query caching é para Import em Premium/Embedded; automatic aggregations são para DirectQuery; agregações definidas pelo usuário ficam ocultas e exigem detalhe em DirectQuery. Reduza Import tirando colunas e prefixos, otimizando tipos e criando colunas no Power Query. Refresh completo pode usar o dobro da memória: use refresh por tabela ou partição via XMLA ou enhanced refresh API. Cada visual gera consulta: menos visuais por página.

## Macetes para memorizar

### 💡 Fallback

SQL Salva (cai para DirectQuery), OneLake Ousa e falha.

### 💡 DirectLakeBehavior

Automatic Atende produção, Only Lake mostra Onde quebra, Only Query mede.

### 💡 Cache x agregação

Cache é de Cópia (Import); Automatic aggregations ajudam o Ao vivo (DirectQuery).

### 💡 Atualização incremental

Arquivo é Antigo e longo, Incremental é Imediato e curto.

### 💡 Total do percentual

ALLSELECTED olha o Slicer; ALL olha o Universo.

### 💡 Janela DAX

WINDOW é faixa, OFFSET é vizinho, INDEX é posição fixa.

### 💡 Bidirecional

Só três convites: ponte, slicer com dados e dimensão por dimensão; no resto, CROSSFILTER na medida.

## Tabela de memorização: não confunda

| Quando é... | ...e quando é |
|---|---|
| **Direct Lake on SQL**<br>Quer fallback para DirectQuery, segurança do SQL analytics endpoint e regras de implantação para religar a fonte; uma única fonte. | **Direct Lake on OneLake**<br>Quer várias fontes Delta (inclusive banco KQL), OneLake security e combinar com tabelas Import; aceita que o refresh falhe acima do guardrail. |
| **ALLSELECTED**<br>Percentual do total respeitando o slicer escolhido pelo usuário. | **REMOVEFILTERS / ALL**<br>Percentual do total geral, ignorando também o slicer. |
| **USERELATIONSHIP**<br>Ativar uma relação inativa existente (data de envio x data de pedido). | **CROSSFILTER**<br>Mudar a direção de filtro de uma relação já ativa só naquela medida. |
| **TREATAS**<br>Não existe relação física: aplicar valores de uma coluna como filtro em outra tabela. | **USERELATIONSHIP / RELATED**<br>A relação existe no modelo (inativa para USERELATIONSHIP, ativa para RELATED). |
| **Query caching**<br>Modelo Import em Premium ou Embedded com página inicial consultada repetidamente. | **Automatic aggregations**<br>Modelo DirectQuery grande, com cache de agregações treinado pelo histórico de consultas. |

| | |
|---|---|
| **Dynamic format string**<br>Mudar o formato mantendo o valor numérico para gráficos e ordenação. | **FORMAT()**<br>Só quando texto é aceitável, pois devolve string e quebra gráficos. |
| **Tabela de datas por papel**<br>Usuário precisa filtrar livremente pelos dois papéis em slicers. | **Relação inativa com USERELATIONSHIP**<br>Só algumas medidas precisam do segundo papel. |

## Números que caem

| Item | Valor |
|---|---|
| Peso do domínio na prova | **25-30%** |
| Limite de upload de modelo pelo Power BI Desktop | **10 GB (mesmo com formato de modelo grande)** |
| Memória de um refresh completo | **até o dobro do tamanho do modelo** |
| Exemplo de política correta (guardar 5 anos, atualizar a última semana) | **Archive data starting 5 anos; Incrementally refresh data starting 7 dias** |

## Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Dizer que Direct Lake on OneLake cai para DirectQuery quando passa do guardrail ou usa view SQL.
**Como não cair:** On OneLake não tem fallback: acima do guardrail o refresh falha e view sem materialização nem é suportada. Fallback é de on SQL.

⚠ **Alerta de prova:** Colocar DirectLakeOnly em produção.
**Como não cair:** Produção usa Automatic; DirectLakeOnly serve para achar falhas e DirectQueryOnly para medir.

⚠ **Alerta de prova:** Achar que iteradores (SUMX) ou o papel Viewer causam fallback.
**Como não cair:** Fallback vem de RLS/OLS/DDM no endpoint, view sem materialização, guardrail ou falta de framing.

⚠ **Alerta de prova:** Tentar adicionar tabela DirectQuery ou Dual a um modelo Direct Lake on SQL.
**Como não cair:** Não é suportado; crie um modelo composto no Desktop sobre ele. Import com Direct Lake é pela web modeling em on OneLake.

⚠ **Alerta de prova:** Achar que Direct Lake funciona em workspace Pro ou PPU se a origem estiver em capacidade F.
**Como não cair:** O próprio modelo precisa estar em capacidade Fabric; e não há gateway nem acesso entre regiões.

⚠ **Alerta de prova:** Esperar que CALCULATE reavalie uma variável definida antes dele.
**Como não cair:** A variável já virou constante; aplique o filtro dentro da expressão da medida.

⚠ **Alerta de prova:** Usar filtro booleano simples e esperar que a linha do visual seja respeitada.
**Como não cair:** Filtro booleano substitui o filtro da coluna; use KEEPFILTERS para intersectar.

⚠ **Alerta de prova:** Ranquear com RANKX(VALUES(Produto[Nome]), ...) ou SUM sem transição de contexto.
**Como não cair:** VALUES no contexto da linha só tem o próprio produto (sempre 1); use ALL ou ALLSELECTED e uma medida.

⚠ **Alerta de prova:** Inverter períodos da atualização incremental ou usar a coluna de partição em Detect data changes.
**Como não cair:** Arquivo é o histórico longo, incremental é a janela curta; Detect data changes usa coluna de auditoria diferente.

⚠ **Alerta de prova:** Criar RangeStart/RangeEnd como texto ou inteiro, ou com outros nomes.
**Como não cair:** Nomes reservados e sensíveis a maiúsculas, tipo date/time; converta dentro do filtro se a chave for yyyymmdd.

⚠ **Alerta de prova:** Republicar pelo Desktop um modelo com atualização incremental para incluir medidas.
**Como não cair:** Republicar apaga partições e dados; use implantação só de metadados via XMLA (ALM Toolkit, TMSL). Nem dá para baixar o .pbix de volta.

⚠ **Alerta de prova:** Achar que query caching serve para DirectQuery ou que automatic aggregations servem para Import.
**Como não cair:** É o inverso: query caching é Import em Premium/Embedded; automatic aggregations otimizam DirectQuery.

⚠ **Alerta de prova:** Usar FORMAT() no item de cálculo ou na medida para mostrar moeda ou percentual.
**Como não cair:** FORMAT devolve texto e quebra gráficos; use Dynamic format string.

⚠ **Alerta de prova:** Achar que relação limitada em modelo composto mostra linha em branco para chaves sem par.
**Como não cair:** Relação limitada usa INNER JOIN: órfãos somem, e RELATED não funciona.

## Como cai na prova

📝 **Como cai:** Dois modelos Direct Lake (um on SQL, outro on OneLake) diante da mesma situação (RLS no endpoint, guardrail, view); pede o que acontece com cada um.
**Como resolver:** Aplique a regra: on SQL faz fallback e respeita a segurança do endpoint; on OneLake não tem fallback, falha no refresh e não enxerga a RLS do SQL. Descarte as alternativas com papéis invertidos.

📝 **Como cai:** Medida DAX com resultado errado (valor repetido, 100%, ranking sempre 1); pede a correção.
**Como resolver:** Identifique o contexto: filtro booleano substituindo (KEEPFILTERS), denominador com filtro do item (ALLSELECTED/REMOVEFILTERS), VALUES no ranking (ALL), variável avaliada antes do CALCULATE.

📝 **Como cai:** Requisito de modelagem (duas datas, metas em outro grão, M:M) com quatro desenhos.
**Como resolver:** Prefira estrela: ponte para M:M, tabela por papel ou USERELATIONSHIP, primeiro dia do período para grão maior. Elimine mesclar fatos, relacionar fatos e bidirecional em tudo.

📝 **Como cai:** Configuração de atualização incremental com períodos e opções.
**Como resolver:** Confira nomes e tipo dos parâmetros, folding, qual período é arquivo e qual é incremental, e se a opção pedida é tempo real (DirectQuery), dias completos ou detecção de mudanças.

📝 **Como cai:** Relatório lento; pede o primeiro passo ou a otimização certa.
**Como resolver:** Siga a ordem: está em capacidade? Capacity Metrics app. Depois Performance analyzer para separar DAX query de Visual display; reduza visuais, otimize modelo e fonte; escolha cache ou agregação conforme o modo.

## Gatilhos do enunciado

| Se o enunciado trouxer... | Pense em |
|---|---|
| descobrir por que o Direct Lake caiu para DirectQuery | **EVALUATE TABLETRAITS() e a coluna DirectLakeFallbackInfo** |
| dados parciais aparecem durante a carga noturna em Direct Lake | **Desligar Keep your Direct Lake data up to date e atualizar o modelo no fim do pipeline** |

| | |
|---|---|
| relatório mostra dados antigos após correção no lakehouse | **Último framing; os dados só aparecem no próximo refresh** |
| banco KQL do Eventhouse como fonte de Direct Lake | **Direct Lake on OneLake com OneLake availability ligada** |
| estender modelo certificado com planilha sem alterar o original | **Make changes to this model (modelo composto), com Build** |
| muitos-para-muitos entre cliente e conta | **Tabela ponte, relações um-para-muitos, uma bidirecional, ponte oculta** |
| meta mensal ou por categoria contra fato diário | **Primeiro dia do mês ligado à Data; ou TREATAS para relação virtual** |
| média móvel dos últimos três meses | **WINDOW(-2, REL, 0, REL, ...) com AVERAGEX** |
| comparação com o mês anterior sem inteligência de tempo | **OFFSET(-1, ...)** |
| leitor troca a dimensão ou a medida do gráfico pelo slicer | **Field parameter** |
| acumulado só no visual, sem mexer no modelo compartilhado | **Visual calculation com RUNNINGSUM** |
| dados de hoje em tempo real com histórico importado | **Get the latest data in real time with DirectQuery (tabela híbrida)** |
| atualizar só os dias que mudaram na fonte | **Detect data changes com coluna de última alteração** |
| subtotal da matriz deve mostrar outra coisa que o nível de loja | **ISINSCOPE** |

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

- Direct Lake exige capacidade Fabric, mesma região e conexões de nuvem; não há gateway.
- On SQL: uma fonte, fallback, regras de implantação. On OneLake: várias fontes, OneLake security, Import junto, sem fallback.
- Grupos de cálculo só afetam medidas explícitas e usam SELECTEDMEASURE; Precedence ordena grupos.
- RangeStart e RangeEnd: nomes exatos, date/time, filtro que dobra; republicar pelo Desktop apaga partições.
- FORMAT devolve texto; para formato variável use Dynamic format string.

## Essenciais para a prova (checklist)

☐ Sei listar as diferenças entre Direct Lake on SQL e on OneLake (fontes, fallback, segurança, composição, regras de implantação).

☐ Sei as causas de fallback, os três valores de DirectLakeBehavior e como usar TABLETRAITS().

☐ Sei explicar framing, Keep your Direct Lake data up to date e o que acontece após refresh com erro.

☐ Sei montar ponte muitos-para-muitos, dimensão role-playing e relação com fato de grão maior.

☐ Sei quando usar KEEPFILTERS, ALLSELECTED, REMOVEFILTERS, TREATAS, USERELATIONSHIP e CROSSFILTER.

☐ Sei escrever WINDOW, OFFSET, INDEX e RANK com ties DENSE ou SKIP.

☐ Sei criar itens de cálculo com SELECTEDMEASURE, ISSELECTEDMEASURE, Precedence e Dynamic format string.

☐ Sei configurar atualização incremental (parâmetros, períodos, tempo real, Detect data changes) e manter partições ao implantar.

☐ Sei o que o formato de modelo grande muda e onde ligá-lo.

☐ Sei escolher entre query caching, automatic aggregations, agregações definidas pelo usuário e redução de modelo.

## Questões de revisão do capítulo

R1.1 · Implementar e gerenciar modelos semânticos

**Um analista da Rede de Lojas Mega Casa escreveu a medida Vendas Quiosque = VAR Total = [Vendas] RETURN CALCULATE(Total, Loja[Tipo] = "Quiosque"). O resultado mostra as vendas de todas as lojas, sem o filtro de quiosque. Por quê?**

A. O filtro booleano em Loja[Tipo] exige KEEPFILTERS para ser aplicado à variável
B. Medidas com VAR são avaliadas em contexto de linha e perdem os filtros do visual atual
C. A variável é avaliada uma vez, fora do CALCULATE, e passa a valer como constante
D. Variáveis não podem ser usadas como primeiro argumento do CALCULATE em medidas

R1.2 · Implementar e gerenciar modelos semânticos

**Na Seguradora Rumo Firme, o SQL analytics endpoint de um lakehouse tem RLS em T-SQL na tabela Apolices. Há dois modelos semânticos Direct Lake sobre esse lakehouse: Riscos, em Direct Lake on SQL com o comportamento padrão, e Atuária, em Direct Lake on OneLake. O que acontece com as consultas à tabela Apolices em cada modelo?**

A. Riscos não aplica a RLS; Atuária cai para DirectQuery para aplicar a RLS do endpoint
B. Os dois falham, porque Direct Lake não lê tabelas com RLS no SQL analytics endpoint
C. Os dois aplicam a RLS do SQL em memória, sem mudar o modo de leitura da tabela
D. Riscos cai para DirectQuery e aplica a RLS; Atuária lê o OneLake sem a RLS do SQL

R1.3 · Implementar e gerenciar modelos semânticos

**No modelo composto da Rede de Hotéis Vista Mar, a tabela Reservas, em DirectQuery para um banco SQL, se relaciona com a tabela Hoteis, importada de uma planilha. A analista percebe que as reservas cujo HotelID não existe em Hoteis não aparecem na linha (Blank) dos visuais, como acontecia quando o modelo era todo Import. Qual é a explicação?**

A. A tabela Hoteis em Import elimina as reservas órfãs a cada atualização do modelo semântico
B. A relação entre grupos de origem diferentes é limitada e usa INNER JOIN, sem linha em branco
C. O modelo composto exige integridade referencial e bloqueia a carga dessas reservas
D. Relações com tabelas DirectQuery passam a ser bidirecionais e descartam valores sem par

R1.4 · Implementar e gerenciar modelos semânticos

**A Rede de Lojas Mega Ofertas quer acelerar a abertura dos relatórios de um modelo semântico Import em capacidade Premium, reaproveitando resultados de consultas já calculados pelo serviço. Qual recurso atende e em que tipo de modelo?**

A. Query caching, disponível em modelos DirectQuery e live connection ao AAS
B. Automatic aggregations, que guardam em cache os resultados de modelos Import
C. On-demand load, que guarda no OneLake os resultados das consultas dos relatórios
D. Query caching, disponível em Premium ou Embedded para modelos Import

## Respostas comentadas

**R1.1. C) A variável é avaliada uma vez, fora do CALCULATE, e passa a valer como constante**
Por quê: Variáveis DAX guardam o resultado da expressão no contexto em que são definidas. Funções que mudam o contexto, como CALCULATE, não reavaliam a variável; por isso, filtros devem ser aplicados na expressão da medida, não ao valor já guardado.
✗ A) Incorreta. KEEPFILTERS altera como o filtro se combina, mas não reavalia a variável.
✗ B) Incorreta. Medidas com VAR continuam no contexto de filtro do visual.
✗ D) Incorreta. A expressão é válida; o problema é que a variável já foi avaliada.

> **Dica de prova:** VAR é constante depois de avaliada: CALCULATE não a recalcula.

**R1.2. D) Riscos cai para DirectQuery e aplica a RLS; Atuária lê o OneLake sem a RLS do SQL**
Por quê: Segurança definida no SQL analytics endpoint afeta só quem consulta por SQL. Para regras que valham em todos os motores, inclusive Direct Lake on OneLake, a recomendação é usar OneLake security ou RLS no modelo semântico.
✗ A) Incorreta. É o contrário: quem tem fallback para DirectQuery é o Direct Lake on SQL.
✗ B) Incorreta. Com o comportamento padrão (Automatic), o Direct Lake on SQL faz fallback em vez de falhar.
✗ C) Incorreta. O Direct Lake não aplica a RLS do SQL analytics endpoint em memória.

> **Dica de prova:** RLS no SQL endpoint: on SQL vira DirectQuery; on OneLake não a vê.

**R1.3. B) A relação entre grupos de origem diferentes é limitada e usa INNER JOIN, sem linha em branco**

Por quê: Relações regulares usam expansão de tabela com LEFT OUTER JOIN e criam um membro desconhecido (Blank) para chaves sem correspondência. Relações limitadas, como muitos-para-muitos ou entre grupos de origem, usam INNER JOIN e têm restrições, como não permitir RELATED.

✗ A) Incorreta. Atualizar a tabela Import não remove linhas de outra tabela.

✗ C) Incorreta. Não há bloqueio de carga; o efeito vem da semântica de junção da relação limitada.

✗ D) Incorreta. A direção do filtro não muda por causa do modo, e não explica o desaparecimento.

> **Dica de prova:** Relação limitada não gera a linha (Blank) de integridade.

**R1.4. D) Query caching, disponível em Premium ou Embedded para modelos Import**

Por quê: Query caching guarda os resultados de consultas da página inicial de relatórios em um cache local da capacidade, reduzindo o trabalho repetido. É um recurso de Power BI Premium e Embedded, só para modelos Import.

✗ A) Incorreta. Query caching não se aplica a DirectQuery nem a live connection a Analysis Services.

✗ B) Incorreta. Automatic aggregations otimizam modelos DirectQuery, não Import.

✗ C) Incorreta. On-demand load carrega páginas de modelos grandes despejados; não guarda resultados no OneLake.

> **Dica de prova:** Cache de resultados de consultas: Query caching em modelos Import.

02

CAPÍTULO 2 · 25-30% DA PROVA

# Manter uma solução de análise de dados

Este domínio vale 25-30% e é sobre segurança, governança e ciclo de vida. Preciso dominar a diferença entre papéis de workspace e permissões de item (Read, Build, Reshare, Write), a OneLake security (nega por padrão, só Grant, não restringe quem tem escrita), RLS e OLS do modelo, rótulos de confidencialidade e endosso. Na parte de ALM caem Git integration, projetos .pbip com TMDL, deployment pipelines (o que é copiado, regras, autobinding), impact analysis, ponto de extremidade XMLA e arquivos .pbit e .pbids. A pegadinha mais comum é achar que segurança de linha vale para quem tem papel de edição, ou que a implantação copia dados e permissões.

**Neste capítulo**

17 tópicos-chave · 14 pegadinhas · 7 macetes · 14 gatilhos · 4 questões comentadas

## Tópicos-chave

### Papéis de workspace

Viewer vê; Contributor vê e modifica; Member também compartilha e adiciona Member, Contributor e Viewer; Admin faz tudo, adiciona outros Admins, exclui o workspace, conecta ao Git e cria a workspace identity. Papéis podem ir para usuários, grupos de segurança, grupos do Microsoft 365 e listas de distribuição; quem está em vários grupos fica com o maior papel. Remover o compartilhamento de um item não tira o papel no workspace.

### Permissões de item em modelos semânticos

Read vê relatórios e exporta dados resumidos; Build cria conteúdo (relatórios, Analyze in Excel), exporta dados subjacentes e lê via XMLA; Reshare compartilha; Write altera. O diálogo Share semantic model tem três caixas (modificar, compartilhar, criar conteúdo) e sem marcar nada dá só Read. Build vem de papel Contributor ou acima, concessão direta, audiência de app ou compartilhamento; sair da audiência do app não revoga Build e Reshare.

### Leitura de dados do lakehouse por papel

Admin, Member, Contributor e Viewer leem com T-SQL pelo SQL analytics endpoint. Ler por Spark, APIs do OneLake ou Lakehouse explorer exige ReadAll, que Admin, Member e Contributor têm e Viewer só recebe por compartilhamento ou função da OneLake security.

### OneLake security

Modelo de negação por padrão: acesso só por funções do tipo Grant (não existem funções Deny). Cada lakehouse nasce com a função DefaultReader, que transforma ReadAll em acesso; excluí-la corta os Viewers. Não restringe Admin, Member e Contributor. Funções têm Read ou ReadWrite (ReadWrite não aceita RLS nem CLS), dados (tabelas, pastas, esquemas) e membros (inclusive grupos do Entra). Mudanças levam alguns minutos, sem reinício.

### RLS, CLS e pastas na OneLake security

RLS vale para tabelas Delta, com regras estáticas (sem usuário logado, sem consulta dinâmica ou multitabela); arquivos usam segurança de pasta. Na CLS o Spark mostra só as colunas permitidas e o SQL analytics endpoint e os modelos dão erro num SELECT *. Dentro de uma função, OLS, RLS e CLS se intersectam; entre funções vale a união (regras de RLS sobre a mesma tabela viram OR). Pasta: herança para baixo e travessia para cima, sem acesso às irmãs.

### RLS no modelo semântico

Funções e regras DAX são criadas no Desktop; membros são atribuídos no serviço, na página Security, depois de publicar (usuários, grupos de segurança, de distribuição e mail-enabled; não grupos do Microsoft 365 nem entidades de serviço). Funções são aditivas e não há regra de negação que prevaleça. Só vale para quem tem leitura: Admin, Member e Contributor não são filtrados. Filtre dimensões e deixe relações ativas propagarem; Apply security filter in both directions só em uma relação bidirecional.

### RLS dinâmica e testes

Uma função com regra comparando coluna de e-mail com USERPRINCIPALNAME(). No serviço, USERNAME() e USERPRINCIPALNAME() devolvem o UPN; no Desktop USERNAME() vem como DOMÍNIO\usuário. Convidados B2B podem ter UPN no formato #EXT#, e Test as role usa a identidade de quem testa. CUSTOMDATA() é para embed. Com poucas regras estáticas, vários modelos por audiência em workspaces separados são alternativa.

### OLS (segurança em nível de objeto)

Esconde tabelas ou colunas e seus metadados para membros de funções com leitura, definida na TMDL view ou no Tabular Editor: modelPermission: read, tablePermission, columnPermission e metadataPermission: none. Sem columnPermission, none esconde a tabela inteira. Não restringe Contributor. Visuais com o campo mostram que ele não foi encontrado; modelos com OLS não suportam Quick insights, Smart narrative nem Excel Data Types.

### Rótulos de confidencialidade

Mecanismos: manual, padrão, obrigatório, programático, herança na criação (relatório herda do modelo), herança downstream (inclusive de itens do Fabric para Power BI, nunca o inverso) e herança de fontes (só modelos Import, configuração do locatário ligada, sem gateway ou VNet, vale o mais restritivo, nunca sobrescreve rótulo manual). Rótulo pai não pode ser aplicado: use o subrótulo. PBIP não suporta rótulos.

### Rótulos fora do serviço

No serviço o rótulo não altera quem acessa. A proteção acompanha os dados por caminhos suportados: exportar para Excel, PDF e PowerPoint como imagem, Analyze in Excel, tabela dinâmica com conexão ao vivo e download do .pbix (que leva o rótulo mais restritivo entre relatório e modelo). Exportação para .csv e PowerPoint com dados ao vivo não levam rótulo.

### Endosso

Promoted: quem tem escrita no item. Certified: revisores definidos pelo administrador do Fabric (com delegação a domínios) e com escrita no item; quem não é revisor segue o link How do I get content certified. Master data: revisores autorizados, só itens que contêm dados. Certified e Master data precisam ser habilitados pelo administrador. Dashboards do Power BI não aceitam endosso. Endossar dá destaque, não acesso. Make discoverable mostra o modelo para quem ainda não tem acesso.

### Git integration

Conecta o workspace inteiro a um branch e pasta do Azure DevOps ou GitHub na nuvem (não GitHub Enterprise Server local, não My workspace). Itens não suportados são ignorados. Admin conecta, sincroniza e desconecta; Member e Contributor fazem commit e update; Viewer não vê informações de Git. Na conexão inicial com conteúdo dos dois lados, escolhe-se uma direção. Update traz o branch inteiro e bloqueia o commit enquanto pendente; item mudado dos dois lados fica em Conflict. Branch out to another workspace isola o desenvolvimento.

### Power BI Desktop project (.pbip) e TMDL

Gera as pastas nome.Report e nome.SemanticModel em texto, um .gitignore que exclui .pbi/cache.abf e localSettings.json, e o .pbip (opcional). O modelo pode ser TMSL (model.bim) ou TMDL (pasta definition, arquivos por tabela, função, perspectiva e cultura), melhor para diff. Clonado sem cache, abre sem dados. Limitações: sem rótulos, caminho de 260 caracteres e sem salvar direto no OneDrive. Git e APIs enviam só metadados; Publish do Desktop leva também o cache. TMDL view: Preview mostra diff, Apply altera só metadados.

### Deployment pipelines

Copiam metadados; dados, credenciais, agenda de refresh, membros de funções e permissões não vão: atualize o modelo no destino. Mantêm ID, URL e permissões do item de destino e preservam dados quando possível (troca de conexão exige refresh completo). Select related inclui o modelo de que o relatório depende; autobinding liga ao item do mesmo estágio e não se desliga por opção. Regras de fonte de dados, de parâmetro e de lakehouse padrão: criadas pelo dono no estágio de destino, nunca em Desenvolvimento, nunca para relatórios. O app de cada estágio precisa ser atualizado.

### Impact analysis e lineage view

Impact analysis mostra itens e workspaces afetados (filhos diretos ou all downstream), agrupa por tipo ou workspace e tem Notify contacts; para fontes de dados mostra só filhos diretos e a cadeia de conexão. Exige escrita no item; itens sem acesso aparecem como Limited access mas contam. A lineage view mostra os itens do workspace e upstream externo um nível acima, não downstream de outros workspaces; Viewers não veem fontes.

### Ponto de extremidade XMLA

URL powerbi:// copiada em Workspace Connection nas configurações do workspace, usada como servidor Analysis Services no SSMS com autenticação Microsoft Entra. Leitura é padrão; escrita exige XMLA Endpoint em Read Write na capacidade. Contributor ou acima é admin do banco (TMSL, deploy, processar partições); Build é leitor com RLS. Não acessíveis: push, Excel, live connection a AAS/SSAS ou a outro modelo. Deploy pelo Visual Studio não leva credenciais: defina no serviço e processe.

### Arquivos .pbit e .pbids e modelos compartilhados

.pbit leva páginas, visuais, definição do modelo e consultas, sem dados, credenciais nem membros de funções; ao abrir pede parâmetros. .pbids traz uma única conexão, sem credenciais nem tabelas, pode fixar DirectQuery ou Import e não suporta Live Connect. Modelo compartilhado: relatórios de outros workspaces via Build; Publish to web não funciona; não pode ser excluído se usado por relatório de outro workspace; usuário gratuito com modelo em F64 cria relatório só no My workspace.

## Macetes para memorizar

### 💡 Papéis de workspace

Viewer Vê, Contributor Cria, Member Multiplica (compartilha), Admin Administra (Git, admins, excluir).

### 💡 Quem a segurança não alcança

Quem edita não é filtrado: Admin, Member e Contributor passam por RLS, OLS e OneLake security.

### 💡 O que a implantação leva

Pipeline leva a planta, não a mobília: metadados sim; dados, credenciais, agenda, membros e permissões não.

### 💡 Combinação de funções

Dentro da função é E (interseção); entre funções é OU (união).

### 💡 Selos de endosso

Promoted autor com escrita no item, Certified revisor, Master data revisor e só item com dados; dashboard fica de fora.

### 💡 Herança de rótulo

Rótulo desce o rio: da fonte para o modelo, do modelo para o relatório, do Fabric para o Power BI; nunca sobe.

### 💡 .pbit x .pbids

pbiT é Template (páginas, modelo e consultas, sem dados, credenciais nem membros de funções); pbiDS é Data Source (só a conexão).

## Tabela de memorização: não confunda

| Quando é... | ...e quando é |
|---|---|
| **Papel no workspace**<br>Dá acesso a todo o conteúdo do workspace; para bloquear alguém, retire o papel. | **Permissão de item / compartilhamento**<br>Dá acesso a um item específico; remover o compartilhamento não mexe no papel. |
| **RLS**<br>Restringir linhas com regra DAX. | **OLS**<br>Esconder tabelas ou colunas inteiras e seus metadados (metadataPermission: none). |
| **Read**<br>Ver relatórios e exportar dados resumidos. | **Build**<br>Criar relatórios, Analyze in Excel, exportar dados subjacentes e consultar via XMLA. |
| **Impact analysis**<br>Saber o que será afetado em outros workspaces e avisar contatos. | **Lineage view**<br>Ver relações entre itens do próprio workspace e fontes upstream. |
| **Git integration / APIs**<br>Implantar PBIP só com metadados; o modelo precisa de refresh. | **Publish do Power BI Desktop**<br>Implantar metadados e o cache de dados local, como um PBIX. |

| | |
|---|---|
| **.pbit**<br>Reaproveitar relatório e modelo completos sem dados. | **.pbids**<br>Só facilitar a conexão a uma fonte; o usuário escolhe tabelas no Navigator. |
| **Regra de parâmetro**<br>Servidor e banco parametrizados no Power Query, inclusive fontes não suportadas por regra de fonte. | **Regra de fonte de dados**<br>Trocar a fonte de modelo ou dataflow quando a fonte é suportada e do mesmo tipo. |

## Números que caem

| Item | Valor |
|---|---|
| Peso do domínio na prova | **25-30%** |
| Capacidade a partir da qual usuário gratuito usa Build em modelo compartilhado | **F64 ou superior (ou Premium)** |
| Limite de caminho padrão no Windows para projetos PBIP | **260 caracteres** |

## Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Colocar gerentes como Contributor e esperar que a RLS os filtre.
**Como não cair:** RLS e OLS do modelo e OneLake security não restringem quem edita; consumidores ficam como Viewer ou recebem acesso por compartilhamento ou app.

⚠ **Alerta de prova:** Criar função FALSE() ou Deny para bloquear quem já está em outra função.
**Como não cair:** Funções de RLS são aditivas e a OneLake security só tem Grant; entre funções vale a união.

⚠ **Alerta de prova:** Atribuir membros às funções no Desktop (Manage roles ou View as).
**Como não cair:** Membros só no serviço, na página Security do modelo; o Desktop define regras e testa.

⚠ **Alerta de prova:** Adicionar grupo do Microsoft 365 como membro de função de RLS.
**Como não cair:** Não é aceito; use grupo de segurança, de distribuição ou mail-enabled. (Papéis de workspace aceitam M365.)

⚠ **Alerta de prova:** Revogar o compartilhamento direto e achar que o usuário perdeu acesso.
**Como não cair:** O papel no workspace continua dando acesso; retire o papel.

⚠ **Alerta de prova:** Achar que Build dá escrita via XMLA ou que conectar por XMLA escapa da RLS.
**Como não cair:** Build é leitor com RLS; escrita exige Contributor ou acima e XMLA em Read Write.

⚠ **Alerta de prova:** Esperar que a implantação copie dados, credenciais, agenda ou membros de funções.
**Como não cair:** Só metadados vão; atualize o modelo no destino e configure o resto lá.

⚠ **Alerta de prova:** Criar regra de implantação no estágio Desenvolvimento ou para um relatório.
**Como não cair:** Regras ficam no estágio de destino e valem para modelos, dataflows e lakehouse padrão.

⚠ **Alerta de prova:** Procurar uma opção para desligar o autobinding ou incluir o app na implantação.
**Como não cair:** Não existem; evite autobinding pelo desenho das conexões e atualize o app do estágio após implantar.

⚠ **Alerta de prova:** Fazer Update from Git na conexão inicial esperando mescla item a item.
**Como não cair:** Update sobrescreve o workspace; commit leva o workspace ao Git e pode ser revertido lá. Só uma direção por vez.

⚠ **Alerta de prova:** Achar que a herança downstream vai do Power BI para o Fabric ou que a herança de fontes troca o rótulo aplicado manualmente.
**Como não cair:** A downstream vai do Fabric para o Power BI, nunca o inverso; a herança de fontes (só modelos Import) não sobrescreve rótulo manual nem rebaixa rótulo.

⚠ **Alerta de prova:** Supor que exportar .csv ou PowerPoint com dados ao vivo leva o rótulo.
**Como não cair:** CSV não leva rótulo; PowerPoint só com Export as image.

⚠ **Alerta de prova:** Achar que endosso dá acesso ou que dashboard pode ser certificado.
**Como não cair:** Endosso só dá destaque; dashboards não aceitam endosso.

⚠ **Alerta de prova:** Esperar ver itens downstream de outros workspaces na lineage view.
**Como não cair:** Use a impact analysis do item.

## Como cai na prova

📝 **Como cai:** Usuário vê dados que não deveria (ou não vê o que deveria); pede a causa.
**Como resolver:** Verifique na ordem: papel no workspace (edição ignora RLS/OLS/OneLake security), funções aditivas ou união, DefaultReader, relação ativa e direção do filtro, formato do UPN.

📝 **Como cai:** Implantação em deployment pipeline com erro ou resultado inesperado.
**Como resolver:** Lembre: só metadados, dependência precisa estar no destino (Select related), regras só no destino e só para modelos/dataflows/lakehouse padrão, app precisa ser atualizado.

📝 **Como cai:** Situação de Git integration (status, conflito, conexão inicial, quem pode fazer o quê).
**Como resolver:** Mapeie o papel: Admin conecta e desconecta, Member e Contributor commit e update, Viewer nada. Update é do branch inteiro e bloqueia commit; conflito exige resolver antes.

📝 **Como cai:** Qual permissão mínima conceder para uma ação.
**Como resolver:** Ver relatório: Read. Criar relatório, Analyze in Excel, XMLA leitura: Build. Compartilhar: Reshare. Alterar: Write ou Contributor. Escolha a menor que atende.

📝 **Como cai:** Rótulo de confidencialidade em item novo, exportado ou baixado.
**Como resolver:** Identifique o mecanismo (criação, downstream, fonte) e se o caminho de saída é suportado; no download do .pbix vale o mais restritivo.

📝 **Como cai:** Escolha DUAS ou TRÊS sobre limitações (OLS, XMLA, PBIP, Git).
**Como resolver:** Julgue cada alternativa: OLS bloqueia Quick insights, Smart narrative e Excel Data Types; XMLA não acessa push, Excel e live connection; PBIP não tem rótulos.

## Gatilhos do enunciado

| Se o enunciado trouxer... | Pense em |
|---|---|
| conectar o SSMS ou o Tabular Editor ao modelo | **URL powerbi:// em Workspace Connection; escrita exige XMLA Read Write** |
| relatório implantado falha porque o modelo não está no estágio | **Select related** |
| trocar servidor do banco por estágio sem ajuste manual | **Parametrizar e criar regra de parâmetro no estágio de destino** |
| testadores veem modelos vazios após a primeira implantação | **Implantação copia só metadados; atualizar o modelo** |
| desenvolver isolado sem afetar o workspace compartilhado | **Branch out to another workspace** |
| versionar modelo com diff legível por tabela | **PBIP com TMDL** |
| avisar donos de relatórios de outros workspaces antes de mudar o modelo | **Impact analysis com Notify contacts** |

| | |
|---|---|
| esconder a coluna CPF de uma função | **OLS com columnPermission e metadataPermission: none** |
| Viewer perdeu acesso às tabelas após criar função personalizada | **DefaultReader excluída ou editada na OneLake security** |
| usuário em duas funções da OneLake security com RLS por estado | **União: vê as linhas das duas funções (OR)** |
| RLS dinâmica precisa atravessar tabela ponte | **Apply security filter in both directions** |
| compartilhar modelo para criar relatórios e repassar o acesso, sem alterar | **Marcar share (Reshare) e build content (Build), sem modify (Write); sem caixa marcada fica só Read** |
| pedidos de acesso devem seguir o processo da empresa | **Request access com instruções em vez de e-mail ao dono** |
| alterar modelo com atualização incremental sem perder histórico | **Implantação só de metadados via XMLA (ALM Toolkit, TMSL)** |

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

- Admin, Member e Contributor não são filtrados por RLS, OLS nem OneLake security.
- Membros de funções de RLS são atribuídos no serviço (página Security), nunca no Desktop.
- Deployment pipelines copiam só metadados; regras ficam no estágio de destino.
- Conectar o workspace ao Git é ação do Admin; Viewer não vê Git.
- Rótulos não funcionam em PBIP e não acompanham exportação para .csv.

## Essenciais para a prova (checklist)

- ☐ Sei o que cada papel de workspace pode fazer e qual vale quando a pessoa está em vários grupos.
- ☐ Sei o que Read, Build, Reshare e Write permitem e como cada um é concedido e revogado.
- ☐ Sei como a OneLake security nega por padrão, usa DefaultReader e combina funções.
- ☐ Sei configurar RLS estática e dinâmica, atribuir membros no serviço e testar com Test as role.
- ☐ Sei definir OLS em TMDL e quais recursos ela bloqueia.
- ☐ Sei os tipos de herança de rótulo e quais exportações levam o rótulo.
- ☐ Sei quem aplica Promoted, Certified e Master data.
- ☐ Sei operar Git integration (conexão, commit, update, conflito, branch out) e a estrutura de um PBIP.
- ☐ Sei o que deployment pipelines copiam, como funcionam regras, Select related e autobinding.
- ☐ Sei usar impact analysis, lineage view, XMLA, .pbit e .pbids.

## Questões de revisão do capítulo

R2.1 · Manter uma solução de análise de dados

**Na Atacadista Grão Dourado, a função de RLS do modelo semântico foi escrita na tabela de fatos Pedidos, usando LOOKUPVALUE para buscar em outra tabela a região do vendedor logado, e os relatórios ficaram lentos para os usuários filtrados. Qual mudança segue a orientação da Microsoft para RLS eficiente?**

A. Criar uma relação inativa entre Vendedor e Pedidos e ativá-la com USERELATIONSHIP na regra
B. Mover a regra para uma coluna calculada em Pedidos que concatena a região ao e-mail do vendedor
C. Aplicar o filtro na dimensão Vendedor e deixar a relação ativa propagá-lo até a tabela Pedidos
D. Repetir a mesma regra em cada tabela do modelo para que o filtro não dependa de relações

R2.2 · Manter uma solução de análise de dados

**A Empresa de Telecom Conecta Já mantém em um lakehouse a lista oficial de clientes, usada como fonte única por várias áreas. A governança quer sinalizar isso no OneLake catalog. Qual selo é indicado e quem pode aplicá-lo?**

A. Master data, aplicado pelo dono do lakehouse sem precisar de autorização
B. Certified, aplicado automaticamente aos itens usados por várias áreas
C. Master data, aplicado por revisores autorizados pelo administrador do Fabric
D. Promoted, aplicado por qualquer pessoa com permissão de escrita no lakehouse oficial

R2.3 · Manter uma solução de análise de dados

**Na Agência de Dados Mapa Claro, o lakehouse Clientes recebeu o rótulo de confidencialidade Confidential. O modelo semântico Direct Lake e o relatório criados a partir dele existem há meses, sem rótulo. Com a herança downstream ligada, o que acontece com esses itens?**

A. Ficam sem rótulo, pois a herança downstream só vale entre itens do Power BI
B. Recebem o rótulo, pois a herança vai de itens do Fabric para os do Power BI
C. Ficam sem rótulo até um admin aplicar rótulos em lote pelas APIs do locatário
D. Recebem o rótulo, que depois volta ao lakehouse quando o relatório é alterado

R2.4 · Manter uma solução de análise de dados

**Na Indústria de Embalagens Boa Caixa, o dono de um modelo semântico recebe muitos e-mails com pedidos de permissão Build e quer que os usuários sigam o fluxo de solicitação de acesso da empresa, feito em um portal interno. O que ele deve configurar?**

A. Trocar, em Request access, o e-mail ao dono por instruções em texto simples
B. Remover o modelo do OneLake catalog para que os pedidos parem de chegar ao dono
C. Configurar uma regra no Microsoft Purview que bloqueie os e-mails de solicitação
D. Desligar a opção Make discoverable para que ninguém consiga pedir acesso ao modelo

## Respostas comentadas

### R2.1. C) Aplicar o filtro na dimensão Vendedor e deixar a relação ativa propagá-lo até a tabela Pedidos

Por quê: A RLS aplica filtros a toda consulta DAX, por isso um bom desenho de modelo é essencial. Em geral, é mais eficiente filtrar dimensões do que fatos e deixar relações ativas levarem o filtro adiante; LOOKUPVALUE deve ser evitado quando a relação do modelo produz o mesmo resultado.

✗ A) Incorreta. Filtros de RLS só se propagam por relações ativas, e o uso de USERELATIONSHIP com RLS pode gerar erros.
✗ B) Incorreta. Uma coluna calculada na tabela de fatos aumenta o modelo e a regra continuaria sendo avaliada linha a linha no fato.
✗ D) Incorreta. Repetir regras em todas as tabelas multiplica filtros e piora o desempenho; o filtro na dimensão já se propaga.

> **Dica de prova:** RLS rápida: filtro na dimensão e propagação pela relação ativa.

### R2.2. C) Master data, aplicado por revisores autorizados pelo administrador do Fabric

Por quê: Os selos de endosso são Promoted, Certified e Master data. Master data vale só para itens que contêm dados, como lakehouses e modelos semânticos, e precisa ser habilitado pelo administrador.

✗ A) Incorreta. Mesmo o dono precisa estar autorizado para aplicar Master data.
✗ B) Incorreta. Nenhum selo é aplicado automaticamente; Certified exige revisor autorizado.
✗ D) Incorreta. Promoted indica que o criador considera o item pronto para reúso, não fonte única de verdade.

> **Dica de prova:** Fonte única de verdade, como lista de clientes: selo Master data.

### R2.3. B) Recebem o rótulo, pois a herança vai de itens do Fabric para os do Power BI

Por quê: A proteção de informações do Fabric combina rotulagem manual, padrão, obrigatória, programática, herança downstream, herança na criação e herança de fontes. A herança downstream não ocorre de itens do Power BI para itens do Fabric.

✗ A) Incorreta. A herança downstream também vale de itens do Fabric para itens do Power BI.
✗ C) Incorreta. Não é preciso intervenção administrativa; a herança downstream é automática.
✗ D) Incorreta. Não há propagação de Power BI para Fabric; o rótulo segue para baixo.

> **Dica de prova:** Herança downstream: do lakehouse para o modelo e o relatório, não ao contrário.

### R2.4. A) Trocar, em Request access, o e-mail ao dono por instruções em texto simples

Por quê: Quando alguém sem Build tenta uma ação que exige essa permissão, o padrão é oferecer um e-mail ao dono do modelo. Admin, Member ou Contributor do workspace podem trocar esse comportamento por instruções, útil quando a organização usa um sistema automatizado de solicitações.

✗ B) Incorreta. Não há como esconder do catálogo os itens a que o usuário já tem acesso, e isso não organiza os pedidos.
✗ C) Incorreta. Bloquear e-mails não mostra aos usuários o caminho correto para pedir acesso.
✗ D) Incorreta. Tirar a descoberta reduz a visibilidade, mas não direciona os pedidos para o processo da empresa.

> **Dica de prova:** Direcionar pedidos de Build a um processo: Request access com instruções.

03

CAPÍTULO 3 · 45-50% DA PROVA

# Preparar dados

Este é o domínio mais pesado da prova (45-50%), então é aqui que eu ganho ou perco o exame. Preciso saber escolher o armazenamento certo (lakehouse, warehouse, Eventhouse, SQL database, Cosmos DB), o jeito certo de trazer os dados (atalho, mirroring, Copy job, pipeline, Dataflow Gen2, notebook) e escrever consultas em quatro linguagens: T-SQL, KQL, Power Query e DAX. A prova quase sempre traz um cenário com uma restrição (sem cópia, sem código, sem quebrar a consulta, mantendo todas as colunas) e as alternativas erradas são sintaxes parecidas que fazem outra coisa. Onde eu mais erro: limites da superfície T-SQL do warehouse (o que não é suportado), ordem dos operadores KQL e detalhes de OneLake availability e OneLake integration. Leio cada palavra do requisito antes de olhar as alternativas.

**Neste capítulo**

20 tópicos-chave · 14 pegadinhas · 8 macetes · 14 gatilhos · 4 questões comentadas

## Tópicos-chave

### Guia de decisão de armazenamento

Eventhouse para eventos de streaming, telemetria e séries temporais com KQL; SQL database in Fabric para OLTP com chaves estrangeiras impostas (mesmo motor do Azure SQL Database); Warehouse para data warehouse corporativo em T-SQL com transações ACID multitabela; Lakehouse para big data, dados de todos os tipos (inclusive não estruturados) e equipes Spark; Cosmos DB in Fabric para NoSQL. SQL database e Cosmos DB também atendem IA com vetores. Todos disponibilizam dados no OneLake.

### Lakehouse x Warehouse

Os dois guardam Delta no OneLake e compartilham o mesmo motor SQL. O lakehouse é desenvolvido com Spark e tem um SQL analytics endpoint gerado automaticamente, somente leitura (sem DML, sem CREATE TABLE); o warehouse tem T-SQL completo com DML, transações multitabela, views, funções e procedimentos. Se os consumidores só leem com T-SQL, o SQL analytics endpoint dispensa um warehouse separado.

### SQL analytics endpoint

Somente leitura para dados e com DDL limitado: aceita views e funções com valor de tabela, mas não triggers, materialized views, CREATE TABLE AS SELECT nem DML. Mapeia tipos Delta: LONG para bigint, DOUBLE para float, TIMESTAMP para datetime2 e STRING para varchar(8000) no lakehouse (varchar(max) em itens espelhados). Tipos sem mapeamento não aparecem como colunas.

### Tipos de dados do Warehouse

Tabelas aceitam um subconjunto: bit, inteiros, decimal, float, real, date, time, datetime2, char, varchar, varbinary e uniqueidentifier. money vira decimal, datetime/smalldatetime/datetimeoffset viram datetime2, nvarchar vira varchar; text, nchar e smallmoney também não são aceitos em tabelas. datetime2 e time têm até 6 dígitos de fração; varchar(max) e varbinary(max) até 16 MB. uniqueidentifier é aceito, mas vira binário no Delta e não é lido como GUID pelo SQL analytics endpoint.

### Limites da superfície T-SQL do Warehouse

Não há triggers, sinônimos, materialized views T-SQL, consultas recursivas (CTE comum funciona, recursiva não) nem SET TRANSACTION ISOLATION LEVEL. ALTER TABLE só adiciona colunas anuláveis, remove colunas (DROP COLUMN) e cria PRIMARY KEY, UNIQUE e FOREIGN KEY com NOT ENFORCED; renomear coluna é com sp_rename. IDENTITY exige bigint, não aceita seed nem increment e pode ter lacunas e valores fora de ordem.

### Sintaxes analíticas do Warehouse

Além do T-SQL tradicional, aceita QUALIFY (filtrar pelo resultado de uma função de janela, ótimo para deduplicar), GROUP BY ALL, ORDER BY ALL, FROM antes de SELECT, APPROX_COUNT_DISTINCT, APPROX_MEDIAN, STRING_AGG com WITHIN GROUP e TRY_CAST/TRY_CONVERT (devolvem NULL em vez de erro). Consultas entre itens do mesmo workspace usam nomes de três partes, sem cópia. Já em Spark SQL, tabelas de lakehouses com esquemas de outros workspaces usam quatro partes (workspace.lakehouse.esquema.tabela).

### Views, funções e procedimentos

View expõe uma consulta e não recebe parâmetros. Função com valor de tabela embutida usa RETURNS TABLE AS RETURN SELECT, é chamada no FROM e, para usar o valor padrão de um parâmetro, a chamada precisa de DEFAULT (omitir só funciona em procedimentos). Parâmetros substituem constantes, nunca nomes de tabela. Procedimento armazenado encapsula DML com parâmetros e é a escolha para cargas chamadas por pipelines.

### SQL query editor e visual query editor

O visual query editor monta consultas sem código com o diagrama do Power Query, só executa leitura e só aceita passos que dobram para T-SQL; existe no SQL analytics endpoint, no Warehouse e em bancos espelhados, e View SQL mostra o T-SQL gerado. O SQL query editor executa DDL, DML e DCL, mas cada Run é um lote em sessão própria: transação dividida entre execuções falha. A prévia mostra só as primeiras 10.000 linhas; Save as view cria view, Save as table usa CTAS, Open in Excel baixa arquivo que roda a consulta, Visualize results cria relatório e não aceita ORDER BY.

### OneLake availability (Eventhouse/banco KQL)

Expõe o banco ou a tabela KQL como cópia lógica em Delta no OneLake, sem custo extra de armazenamento, para atalhos em lakehouses e warehouses, notebooks e Direct Lake on OneLake. A representação Delta é somente leitura e a política de retenção do banco KQL vale também para ela. Algumas mudanças de tabela ficam bloqueadas enquanto está ligada: desligue, altere e religue. Para gerar Parquet de 200 a 256 MB o Eventhouse pode atrasar a gravação em até 3 horas; TargetLatencyInMinutes da policy mirroring reduz esse atraso.

### OneLake integration de modelos semânticos

Grava em Delta, após cada refresh, as tabelas Import do modelo. Exige capacidade Premium P ou Fabric F (não funciona em PPU). Não exporta medidas, tabelas DirectQuery, híbridas, de grupo de cálculo nem agregações gerenciadas pelo sistema. Se o modelo tem RLS ou OLS, só quem tem escrita no modelo (como Contributor) lê os arquivos exportados.

### Atalhos do OneLake

Funcionam como links simbólicos: não copiam dados. Excluir o atalho não apaga o destino, mas excluir um arquivo dentro dele apaga na origem se o usuário tiver escrita lá; renomear a pasta de destino pode quebrar o atalho. Em Tables ficam só no nível superior (com descoberta de tabelas Delta); em Files podem ficar em qualquer nível e não convertem formatos. Atalhos apontam para armazenamento (OneLake, ADLS, S3, GCS), não para tabelas de bancos SQL.

### Mirroring e Copy job

Database mirroring replica bancos (como Azure SQL) em Delta quase em tempo real; metadata mirroring sincroniza catálogos e usa atalhos (Azure Databricks, sem duplicar dados); open mirroring recebe mudanças numa landing zone gravada por aplicações. A computação de replicação é gratuita e as consultas consomem capacidade normalmente; com a capacidade pausada a replicação para. Copy job faz cópia em massa, incremental (carga completa na primeira execução) e CDC, sem código e sem pipeline.

### Escolha da ferramenta de ingestão

Copy activity em pipeline e Copy job: ingestão sem código, inclusive volumosa. Dataflow Gen2: transformação com Power Query, para analistas. Notebook Spark: lógica complexa em código (Python, Scala, SQL, R), como logs semiestruturados. Eventstream: streaming. Os destinos do Dataflow Gen2 com configurações automáticas usam Replace; para acumular, desligue as automáticas e escolha Append.

### Conexões, gateways e Real-Time hub

Conexões de nuvem compartilháveis são criadas em Settings, Manage connections and gateways, com papéis próprios (User usa, User with resharing usa e compartilha, Owner gerencia); papéis de workspace não dão acesso a elas. OAuth2 pode falhar em consultas longas por expiração do token. On-premises data gateway atende fontes da rede local (pipelines exigem versão compatível e saída para *.frontend.clouddatahub.net); VNet data gateway alcança fontes Azure com private endpoint sem instalar máquina. O Real-Time hub é o catálogo único de streaming do locatário, com conectores (Event Hubs, Kafka, Kinesis, CDC de bancos) e Fabric events e Azure events que disparam alertas.

### OneLake catalog

Tem três abas: Explore (descobrir itens, com filtros All, My items, Endorsed items, Favorites, workspace, domínio, tags e categoria de tipo como Data types; expande lakehouses e warehouses até as tabelas e abre os detalhes com as colunas, sem abrir o item), Govern (estado de governança, insights e ações recomendadas) e Secure (papéis de workspace e funções da OneLake security). Modelos marcados com Make discoverable aparecem para quem ainda não tem acesso, para pedir acesso.

### Modelagem dimensional no warehouse

O grão do fato é definido pelos valores das chaves de dimensão; fatos de grãos diferentes ficam em tabelas separadas. Dimensões são desnormalizadas (hierarquia achatada) e o snowflake é exceção. A chave substituta isola o warehouse da origem (SCD tipo 2, várias fontes) e não deve ter significado, exceto em data e hora (AAAAMMDD); a chave natural liga a linha à origem no ETL. Como FKs não são impostas, o ETL testa integridade (LEFT JOIN com IS NULL) e usa membro Desconhecido ou inferido em vez de chave nula.

### Power Query: tipos, perfil e limpeza

A detecção automática de tipos olha as 200 primeiras linhas; as ferramentas de perfil (Column quality, distribution e profile) olham as 1.000 primeiras, a menos que se mude para o conjunto inteiro. Formato estrangeiro de data ou número: Change type Using locale. Fixed decimal number guarda 4 casas e 19 dígitos (exato para dinheiro); Decimal number é ponto flutuante. Fill down só preenche null (vazio precisa virar null com Replace values antes); Remove duplicates diferencia maiúsculas e não garante manter a primeira ocorrência.

### Power Query: combinar e agregar

Merge queries junta colunas por chave (Left outer, Inner, Left anti, Right anti, Full outer); chave composta exige mesma ordem de seleção e mesmo tipo, não mesmo nome. Fuzzy matching só em texto, Jaccard, Similarity threshold padrão 0,80 (1,00 equivale a exato). Append queries empilha linhas casando colunas pelo nome e põe null onde falta; Append queries as new cria consulta nova. Group by Advanced aceita várias colunas e várias agregações; All rows guarda as linhas do grupo como tabela; Count distinct values e Percentile só no Power Query Online.

### KQL essencial

Filtre cedo com where (tempo primeiro) e só depois agregue. has usa o índice de termos (3 ou mais caracteres) e é mais rápido que contains. project mantém só o que lista, project-away remove poucas colunas, extend acrescenta. summarize com bin agrupa por intervalo, mas omite intervalos vazios; make-series preenche lacunas com default. arg_max(Timestamp, *) by Id traz o último registro completo. O join padrão é innerunique (deduplica a esquerda), lookup enriquece fato com dimensão, union empilha mesmo com esquemas diferentes, render é sempre o último operador.

### Consultas DAX no DAX query view

Toda consulta tem EVALUATE com uma expressão de tabela; escalar precisa de chaves, como EVALUATE {[Medida]}. Opcionais: DEFINE (medidas valem só na consulta), ORDER BY e START AT (valores constantes por coluna do ORDER BY; não existe OFFSET). SUMMARIZECOLUMNS recebe agrupamentos, depois filtros (TREATAS, FILTER, KEEPFILTERS), depois pares nome e expressão. Sort by column não vale em consultas DAX; INFO e INFO.VIEW listam metadados; Update model with changes grava a medida testada no modelo.

## Macetes para memorizar

### 💡 Warehouse: tipos que trocam

Dinheiro vira decimal, data velha vira datetime2, N some: money para decimal, datetime para datetime2, nvarchar para varchar.

### 💡 O que o warehouse não tem

TSMR, Tem Sempre Mais Restrição: sem Triggers, sem Sinônimos, sem Materialized views, sem Recursão.

### 💡 Quantas linhas o Power Query olha

200 para tipar, 1.000 para perfilar, 100 para Column from examples.

### 💡 Guia de decisão de armazenamento

Evento na Eventhouse, OLTP no SQL database, T-SQL no Warehouse, Spark no Lakehouse, NoSQL no Cosmos.

### 💡 Ordem KQL

Where primeiro, summarize depois, render por último.

### 💡 Funções de classificação T-SQL

ROW_NUMBER nunca repete, RANK repete e pula, DENSE_RANK repete e não pula, NTILE divide em grupos.

### 💡 Atalho

Atalho é ponteiro: apagar o atalho é seguro; apagar dentro dele não.

### 💡 WHERE x HAVING

WHERE filtra linha antes do grupo; HAVING filtra grupo depois do SUM.

## Tabela de memorização: não confunda

| Quando é... | ...e quando é |
|---|---|

| | |
|---|---|
| **Warehouse**<br>Precisa de DML em T-SQL, transações ACID em várias tabelas, procedimentos de carga e equipe que trabalha com SQL. | **Lakehouse com SQL analytics endpoint**<br>Dados de todos os tipos com Spark, e consumidores que só leem em T-SQL (views e funções), sem precisar de outro item. |
| **SQL database in Fabric**<br>Aplicativo OLTP com alta concorrência de gravação e chaves estrangeiras realmente impostas. | **Warehouse**<br>Data warehouse analítico em T-SQL; chaves só NOT ENFORCED e integridade testada no ETL. |
| **OneLake availability (Eventhouse)**<br>Expor tabelas KQL em Delta para T-SQL, Spark e Direct Lake sem cópia; retenção do banco vale para o Delta. | **OneLake integration (modelo semântico)**<br>Exportar as tabelas Import de um modelo em Delta após o refresh; só em capacidade P ou F, sem medidas nem DirectQuery. |
| **Atalho do OneLake**<br>Apontar para arquivos ou tabelas Delta em armazenamento (OneLake, ADLS, S3, GCS) sem copiar. | **Mirroring**<br>Replicar um banco operacional (Azure SQL e outros) em Delta quase em tempo real; atalho não aponta para tabela de banco SQL. |
| **On-premises data gateway**<br>Fonte em rede local sem acesso pela internet (SQL Server na fábrica). | **VNet data gateway**<br>Fonte Azure protegida por private endpoint ou Private Link, sem querer instalar e manter máquina de gateway. |
| **summarize ... by bin()**<br>Agregar por intervalo quando dias ou horas sem dados podem simplesmente não aparecer. | **make-series**<br>Série com eixo regular (from, to, step) que precisa mostrar zero nos intervalos vazios. |
| **ROW_NUMBER com PARTITION BY (ou QUALIFY)**<br>Manter a linha mais recente inteira por chave, com todas as colunas coerentes. | **MAX por coluna com GROUP BY**<br>Só o valor máximo de uma coluna; MAX em várias colunas mistura valores de linhas diferentes. |

## Números que caem

| Item | Valor |
|---|---|
| Peso do domínio na prova | **45-50%** |
| Prévia de resultados do SQL query editor | **primeiras 10.000 linhas** |
| Linhas usadas pelo perfil de dados do Power Query (padrão) | **1.000 primeiras** |
| Linhas inspecionadas na detecção automática de tipos | **200 primeiras** |
| Linhas usadas por Column from examples | **100 primeiras da prévia** |
| Similarity threshold padrão do fuzzy merge | **0,80 (escala de 0 a 1)** |
| Precisão de datetime2 e time no warehouse | **até 6 dígitos de fração** |
| varchar(max) e varbinary(max) no warehouse | **até 16 MB** |

| | |
|---|---|
| STRING do Delta no SQL analytics endpoint do lakehouse | **varchar(8000)** |
| Fixed decimal number (Currency) | **4 casas decimais, 19 dígitos** |
| Arquivos Parquet da OneLake availability | **200 a 256 MB, com atraso de até 3 horas** |
| has em KQL usa o índice para termos com | **3 ou mais caracteres** |

## Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Usar CAST ou CONVERT para converter coluna de staging com valores como N/D.
**Como não cair:** CAST e CONVERT dão erro e derrubam a consulta; TRY_CAST/TRY_CONVERT devolvem NULL. ISNULL não converte nada, só troca nulos.

⚠ **Alerta de prova:** Filtrar ROW_NUMBER no WHERE ou no HAVING para deduplicar.
**Como não cair:** Função de janela não entra no WHERE (avaliado antes) nem no HAVING; use subconsulta com rn = 1 ou QUALIFY. E particione pela chave, ordenando pela data DESC.

⚠ **Alerta de prova:** Achar que o warehouse aceita triggers, materialized views, CTE recursiva ou PRIMARY KEY imposta.
**Como não cair:** Nenhum deles é suportado; chaves só NOT ENFORCED, recursão não existe nem com MAXRECURSION, e ALTER TABLE não adiciona coluna NOT NULL.

⚠ **Alerta de prova:** Criar tabela, rodar UPDATE ou CTAS no SQL analytics endpoint do lakehouse.
**Como não cair:** O endpoint é somente leitura: só views e funções com valor de tabela. DML e tabelas novas são do warehouse.

⚠ **Alerta de prova:** Esperar que o visual query editor execute DELETE ou qualquer passo do Power Query.
**Como não cair:** Ele só executa leitura e só aceita passos que dobram para T-SQL. Alterar dados é no SQL query editor.

⚠ **Alerta de prova:** Dividir BEGIN TRAN, UPDATE e COMMIT em execuções separadas no SQL query editor.
**Como não cair:** Cada Run é um lote em sessão nova; a transação inteira precisa estar em uma única execução.

⚠ **Alerta de prova:** Achar que os dados do banco KQL com OneLake availability ficam no Delta além da retenção.
**Como não cair:** A política de retenção do banco vale também para a cópia Delta; para histórico maior, aumente a retenção ou grave em outro destino.

⚠ **Alerta de prova:** Supor que a OneLake integration de modelo funciona em PPU ou exporta medidas e tabelas DirectQuery.
**Como não cair:** Só P ou F, e só tabelas Import. Com RLS ou OLS, Read e Build não bastam para ler o Delta: precisa de escrita no modelo.

⚠ **Alerta de prova:** Achar que excluir um arquivo dentro do atalho nunca afeta a origem.
**Como não cair:** Apagar o atalho é seguro; apagar dentro dele apaga na origem se houver permissão de escrita.

⚠ **Alerta de prova:** Usar contains ou matches regex para procurar palavra inteira em KQL.
**Como não cair:** has usa o índice de termos e é mais rápido; contains varre substrings e regex não aproveita índice.

⚠ **Alerta de prova:** Usar take ou limit para pegar os N maiores, ou colocar render antes do summarize.
**Como não cair:** take/limit pegam linhas quaisquer e limit não aceita by; use top N by coluna. render vai sempre no fim.

⚠ **Alerta de prova:** Testar string vazia em KQL com isnull ou == null.
**Como não cair:** String no Kusto nunca é nula; use isempty() ou isnotempty(). isnull serve para outros tipos.

⚠ **Alerta de prova:** Concluir que a coluna está limpa porque Column quality mostrou 100% Valid.
**Como não cair:** O perfil olha só as 1.000 primeiras linhas por padrão; mude para o conjunto inteiro antes de concluir.

⚠ **Alerta de prova:** Deixar o destino do Dataflow Gen2 nas configurações automáticas e esperar acumular dados.
**Como não cair:** O automático é Replace; para carga acumulativa, desligue as automáticas e escolha Append.

## Como cai na prova

📝 **Como cai:** Cenário de empresa com restrições (sem cópia, sem código, equipe que só sabe SQL) pedindo o item de armazenamento ou a ferramenta de ingestão.
**Como resolver:** Sublinhe as restrições. Mapeie o tipo de carga no guia de decisão (streaming, OLTP, DW T-SQL, Spark, NoSQL) e elimine as alternativas que copiam dados ou exigem código quando isso é proibido.

📝 **Como cai:** Quatro consultas T-SQL quase iguais; pede a que devolve exatamente o resultado.
**Como resolver:** Confira PARTITION BY (reinicia por grupo?), ORDER BY (direção certa?), moldura (ROWS UNBOUNDED PRECEDING para acumulado), WHERE x HAVING e se a função existe no warehouse.

📝 **Como cai:** Quatro pipelines KQL com os mesmos operadores em ordens diferentes.
**Como resolver:** Simule a saída de cada operador: where antes de summarize, summarize perde colunas não agrupadas, top/take antes do filtro muda o resultado, render só no fim.

📝 **Como cai:** Pergunta sobre o que acontece com dados ou permissões ao ligar OneLake availability, OneLake integration ou atalho.
**Como resolver:** Lembre que Delta exposto é somente leitura, segue a retenção do KQL, que integration só exporta Import em P/F, e que operações dentro do atalho afetam a origem.

📝 **Como cai:** Erro inesperado no Power Query (tipos com erro, perfil limpo mas carga falha, Fill down sem efeito).
**Como resolver:** Pergunte: qual localidade? quantas linhas foram olhadas (200 ou 1.000)? o valor é null ou texto vazio? Ajuste o passo certo sem mexer no gateway ou no destino.

📝 **Como cai:** Escolha DUAS ou TRÊS sobre comportamento de recurso (atalhos, visual query editor, Real-Time hub).
**Como resolver:** Avalie cada alternativa isoladamente como verdadeiro ou falso; as erradas costumam trocar Tables por Files ou listar fonte que não é de streaming.

## Gatilhos do enunciado

| Se o enunciado trouxer... | Pense em |
|---|---|
| converter sem que linhas inválidas façam a consulta falhar | **TRY_CAST / TRY_CONVERT** |
| registro mais recente de cada chave, mantendo todas as colunas | **ROW_NUMBER() OVER (PARTITION BY chave ORDER BY data DESC) = 1, ou QUALIFY** |
| subtotais por nível e total geral numa só consulta | **GROUP BY ROLLUP** |

| | |
|---|---|
| posição com empate sem pular a seguinte (1, 2, 2, 3) | **DENSE_RANK; RANK pula, ROW_NUMBER não repete** |
| dados do Eventhouse em T-SQL ou Spark sem pipeline de cópia | **Ligar a OneLake availability e criar atalho** |
| OLTP com chaves estrangeiras impostas dentro do Fabric | **SQL database in Fabric** |
| telemetria, streaming, séries temporais com análise interativa | **Eventhouse com banco KQL** |
| Azure Databricks sem duplicar dados nem ETL | **Metadata mirroring** |
| trazer só o que mudou, sem código e sem pipeline | **Copy job em modo incremental** |
| banco Azure sem endpoint público, sem manter máquina de gateway | **VNet data gateway** |
| reduzir egress em atalhos para S3 ou GCS com leituras repetidas | **Cache de atalhos na aba OneLake das configurações do workspace** |
| último estado de cada entidade em KQL | **summarize arg_max(Timestamp, *) by Id** |
| série diária com zero nos dias sem eventos | **make-series ... default = 0 from ... to ... step 1d** |
| nomes digitados com variações na junção de textos | **Merge com fuzzy matching (Jaccard, threshold menor que 1,00)** |

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

- O SQL analytics endpoint do lakehouse é somente leitura; DML, procedimentos de carga e transações multitabela são do warehouse.
- No warehouse, chaves são sempre NOT ENFORCED, IDENTITY é bigint sem seed e com lacunas, e não existem triggers, sinônimos, materialized views nem recursão.
- Atalhos e OneLake availability não copiam dados; mirroring replica; Copy job e pipelines copiam.
- Em KQL: where cedo, has em vez de contains, arg_max para último registro, isempty para strings, render no fim.
- Em consultas DAX: EVALUATE precisa de tabela, DEFINE vale só na consulta e Sort by column não é aplicado.

## Essenciais para a prova (checklist)

- ☐ Sei escolher entre Lakehouse, Warehouse, Eventhouse, SQL database e Cosmos DB a partir do cenário.
- ☐ Sei quais tipos T-SQL não existem em tabelas do warehouse e por qual tipo trocar cada um.

☐ Sei deduplicar com ROW_NUMBER/QUALIFY e calcular acumulado com SUM() OVER (PARTITION BY ... ORDER BY ... ROWS UNBOUNDED PRECEDING).

☐ Sei criar função com valor de tabela, chamá-la com DEFAULT e quando usar procedimento em vez de view.

☐ Sei o que o visual query editor e o SQL query editor podem e não podem executar, e o que faz cada botão da barra de resultados.

☐ Sei o efeito de OneLake availability, OneLake integration, atalhos e cache de atalhos sobre dados e permissões.

☐ Sei escolher entre atalho, database/metadata/open mirroring, Copy job, pipeline, Dataflow Gen2 e notebook.

☐ Sei resolver tipos com locale, nulos com Fill down, duplicados, Merge (anti, fuzzy) e Append no Power Query.

☐ Sei escrever em KQL filtros, summarize com bin, make-series, arg_max, join/lookup/union, top e countif.

☐ Sei montar uma consulta DAX com EVALUATE, DEFINE MEASURE, SUMMARIZECOLUMNS, ORDER BY e START AT.

## Questões de revisão do capítulo

R3.1 · Preparar dados

**Uma consulta KQL da Operadora de Logística Via Rápida filtra a coluna Mensagem com contains 'timeout' e varre bilhões de linhas. O termo procurado aparece como palavra inteira nas mensagens. Qual ajuste segue as boas práticas de consulta?**

A. Trocar contains por contains_cs, que desconsidera maiúsculas e minúsculas
B. Mover o filtro para depois do summarize, para reduzir as linhas antes da busca no texto
C. Trocar contains por matches regex, que aproveita o índice de termos da coluna
D. Trocar contains por has, que usa o índice de termos com três ou mais caracteres

R3.2 · Preparar dados

**Um analista da Agência de Viagens Rota Azul escreveu uma consulta KQL que primeiro agrega bilhões de eventos por país e cidade e só depois filtra o país Brasil, e ela está lenta. Qual versão segue as boas práticas de consulta?**

A. Eventos | sort by Cidade | where Pais == 'BR' | summarize Total = count() by Cidade
B. Eventos | where Pais == 'BR' | summarize Total = count() by Cidade | top 10 by Total
C. Eventos | summarize Total = count() by Pais, Cidade | where Pais == 'BR' | top 10 by Total
D. Eventos | top 10 by Cidade | where Pais == 'BR' | summarize Total = count() by Cidade

R3.3 · Preparar dados

**Uma analista da Rede de Shoppings Praça Central precisa listar, na tabela KQL Sensores, as leituras das últimas 24 horas com temperatura de 18 a 25 graus, incluindo os dois limites. Qual consulta atende?**

A. Sensores | where ago(1d) > Timestamp and Temperatura between (18 .. 25)
B. Sensores | where Timestamp > ago(1d) and Temperatura between (18 .. 25)
C. Sensores | where Timestamp > ago(1d) and Temperatura in (18, 25)
D. Sensores | where Timestamp > ago(1d) and Temperatura !between (18 .. 25)

R3.4 · Preparar dados

**Um modelador da Clínica Veterinária Pata Amiga testou no DAX query view uma nova versão da medida [Ticket Médio] usando DEFINE MEASURE e agora quer gravá-la no modelo semântico. O que ele deve usar?**

A. O comando EVALUATE com a palavra-chave SAVE no final da consulta
B. A opção Copy query do Performance analyzer, colando a medida no Model view
C. Nada, pois medidas criadas com DEFINE MEASURE são salvas ao executar
D. A ação Update model (CodeLens) ou o botão Update model with changes

## Respostas comentadas

**R3.1. D) Trocar contains por has, que usa o índice de termos com três ou mais caracteres**
Por quê: Em KQL, has procura termos inteiros com o índice de termos e é bem mais rápido que contains, que procura substrings varrendo os dados. Também é boa prática filtrar cedo com where.
✗ A) Incorreta. contains_cs diferencia maiúsculas e minúsculas e também não usa o índice de termos.
✗ B) Incorreta. Filtrar antes é que reduz as linhas processadas; depois do summarize os detalhes já foram agregados.
✗ C) Incorreta. Expressões regulares não aproveitam o índice de termos e costumam ser mais caras.

> **Dica de prova:** Palavra inteira em KQL: has em vez de contains.

**R3.2. B) Eventos | where Pais == 'BR' | summarize Total = count() by Cidade | top 10 by Total**
Por quê: As boas práticas de KQL recomendam aplicar filtros (where) o mais cedo possível, principalmente sobre colunas de tempo e colunas indexadas, e só depois agregar, ordenar e limitar o resultado.
✗ A) Incorreta. Ordenar a tabela inteira antes de filtrar é caro e não ajuda a agregação.
✗ C) Incorreta. Agregar tudo antes de filtrar processa dados desnecessários.
✗ D) Incorreta. top antes do filtro e da agregação pega 10 linhas quaisquer pela cidade e altera o resultado.

> **Dica de prova:** KQL eficiente: where primeiro, summarize depois.

### R3.3. B) Sensores | where Timestamp > ago(1d) and Temperatura between (18 .. 25)

Por quê: Em KQL, between filtra valores dentro de uma faixa inclusiva, em números ou datas, e !between exclui a faixa. ago() calcula um instante relativo ao momento da consulta, sendo comum filtrar o tempo cedo, com where, para ler menos dados.

✗ A) Incorreta. A comparação invertida traz leituras anteriores às últimas 24 horas.

✗ C) Incorreta. O operador in compara com os valores exatos 18 e 25, sem os valores intermediários.

✗ D) Incorreta. !between devolve as leituras fora da faixa pedida.

> **Dica de prova:** Faixa inclusiva em KQL: between (mínimo .. máximo).

### R3.4. D) A ação Update model (CodeLens) ou o botão Update model with changes

Por quê: DEFINE MEASURE permite testar medidas novas ou alterar medidas existentes no escopo da consulta, sem afetar o modelo. No DAX query view, as ações de CodeLens e o botão Update model with changes levam essas medidas ao modelo semântico.

✗ A) Incorreta. Não existe palavra-chave SAVE em consultas DAX.

✗ B) Incorreta. Copy query copia a consulta de um visual; não grava medidas no modelo.

✗ C) Incorreta. Medidas de DEFINE são do escopo da consulta e não alteram o modelo ao executar.

> **Dica de prova:** Medida testada no DAX query view vai ao modelo com Update model with changes.

DIA DA PROVA

# Estratégia de prova

O DP-600 é uma prova de cenário: cada questão descreve uma empresa, um requisito e uma restrição (sem cópia de dados, sem código, mínimo privilégio, sem afetar outros consumidores) e pede a ação, a consulta ou a configuração exata. As alternativas erradas são quase sempre recursos reais do Fabric usados no lugar errado ou com uma característica invertida (on SQL x on OneLake, Replace x Append, Read x Build). Preparar dados pesa quase metade da prova, com muita sintaxe de T-SQL, KQL, Power Query e DAX em que as quatro opções diferem por um detalhe. Estude pelos nomes atuais (OneLake catalog, Microsoft Entra ID, Real-Time hub, Dataflow Gen2, Direct Lake on OneLake e on SQL analytics endpoint); recursos aposentados ou renomeados, como Azure AD, Datamart ou OneLake data hub, não são a resposta certa.

## Como responder

✓ Conte com cerca de 50 questões em 100 minutos (a Microsoft não divulga o número exato): ritmo de uns 2 minutos por questão, marcando para revisão as longas de sintaxe.

✓ A nota mínima é 700 numa escala de 1 a 1000 e o formato mistura múltipla escolha e múltipla resposta com cenários de engenharia de análise: leia quantas alternativas o enunciado pede antes de marcar.

✓ Priorize Preparar dados (45-50%): é onde está quase metade dos pontos; os outros dois domínios valem 25-30% cada.

✓ Antes de ler as alternativas, sublinhe a restrição do enunciado (sem cópia, sem código, sem alterar o modelo, mínimo privilégio) e use-a para eliminar opções.

✓ Em questões de sintaxe, simule a saída de cada alternativa: onde está o PARTITION BY, a ordem dos operadores KQL, a posição do filtro no SUMMARIZECOLUMNS.

✓ Em múltipla resposta (Escolha DUAS ou TRÊS), julgue cada alternativa como verdadeira ou falsa isoladamente; as erradas costumam trocar um detalhe (Tables x Files, Import x DirectQuery).

✓ Em questões de Direct Lake, decida primeiro se é on SQL ou on OneLake; isso já define fallback, fontes, segurança e composição.

✓ Desconfie de alternativas com nome antigo (Azure AD, Datamart, OneLake data hub) ou de opções e botões que não existem (desligar autobinding, copiar dados na implantação, função Deny).

## Erros que derrubam candidatos

✗ Achar que RLS, OLS ou OneLake security restringem quem tem papel Admin, Member ou Contributor.

✗ Confundir quem tem fallback: Direct Lake on SQL cai para DirectQuery; on OneLake falha.

✗ Supor que o warehouse aceita triggers, materialized views, recursão, chaves impostas ou todos os tipos do SQL Server.

✗ Escrever KQL com o filtro depois do summarize, contains em vez de has, ou take em vez de top.

✗ Esperar que deployment pipelines copiem dados, credenciais e membros de funções.

✗ Usar FORMAT para formatar medida e quebrar gráficos, em vez de Dynamic format string.

✗ Republicar pelo Desktop um modelo com atualização incremental e perder as partições.

## Na semana da prova

- ☐ Refaça as provas práticas por domínio e anote cada erro como uma regra curta (ex.: 'on OneLake não tem fallback').
- ☐ Revise as tabelas de comparação: Direct Lake on SQL x on OneLake, papéis de workspace, Read x Build x Reshare, Lakehouse x Warehouse.
- ☐ Reescreva de memória as sintaxes mais cobradas: ROW_NUMBER com QUALIFY, SUM() OVER acumulado, arg_max, make-series, SUMMARIZECOLUMNS, WINDOW e OFFSET.
- ☐ Releia a lista do que o warehouse e o SQL analytics endpoint não suportam e o que a implantação não copia.
- ☐ Na véspera, revise só os macetes e o checklist de cada domínio e descanse; no dia, controle o tempo em blocos de 25 questões.

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 modelos semânticos

‣ descobrir por que o Direct Lake caiu para DirectQuery → **EVALUATE TABLETRAITS() e a coluna DirectLakeFallbackInfo**

‣ dados parciais aparecem durante a carga noturna em Direct Lake → **Desligar Keep your Direct Lake data up to date e atualizar o modelo no fim do pipeline**

‣ relatório mostra dados antigos após correção no lakehouse → **Último framing; os dados só aparecem no próximo refresh**

‣ banco KQL do Eventhouse como fonte de Direct Lake → **Direct Lake on OneLake com OneLake availability ligada**

‣ estender modelo certificado com planilha sem alterar o original → **Make changes to this model (modelo composto), com Build**

⚠ On OneLake não tem fallback: acima do guardrail o refresh falha e view sem materialização nem é suportada. Fallback é de on SQL.

⚠ Produção usa Automatic; DirectLakeOnly serve para achar falhas e DirectQueryOnly para medir.

⚠ Fallback vem de RLS/OLS/DDM no endpoint, view sem materialização, guardrail ou falta de framing.

## 2. Manter uma solução de análise de dados

‣ conectar o SSMS ou o Tabular Editor ao modelo → **URL powerbi:// em Workspace Connection; escrita exige XMLA Read Write**

‣ relatório implantado falha porque o modelo não está no estágio → **Select related**

‣ trocar servidor do banco por estágio sem ajuste manual → **Parametrizar e criar regra de parâmetro no estágio de destino**

‣ testadores veem modelos vazios após a primeira implantação → **Implantação copia só metadados; atualizar o modelo**

‣ desenvolver isolado sem afetar o workspace compartilhado → **Branch out to another workspace**

⚠ RLS e OLS do modelo e OneLake security não restringem quem edita; consumidores ficam como Viewer ou recebem acesso por compartilhamento ou app.

⚠ Funções de RLS são aditivas e a OneLake security só tem Grant; entre funções vale a união.

⚠ Membros só no serviço, na página Security do modelo; o Desktop define regras e testa.

## 3. Preparar dados

‣ converter sem que linhas inválidas façam a consulta falhar → **TRY_CAST / TRY_CONVERT**

‣ registro mais recente de cada chave, mantendo todas as colunas → **ROW_NUMBER() OVER (PARTITION BY chave ORDER BY data DESC) = 1, ou QUALIFY**

‣ subtotais por nível e total geral numa só consulta → **GROUP BY ROLLUP**

‣ posição com empate sem pular a seguinte (1, 2, 2, 3) → **DENSE_RANK; RANK pula, ROW_NUMBER não repete**

‣ dados do Eventhouse em T-SQL ou Spark sem pipeline de cópia → **Ligar a OneLake availability e criar atalho**

⚠ CAST e CONVERT dão erro e derrubam a consulta; TRY_CAST/TRY_CONVERT devolvem NULL. ISNULL não converte nada, só troca nulos.

⚠ Função de janela não entra no WHERE (avaliado antes) nem no HAVING; use subconsulta com rn = 1 ou QUALIFY. E particione pela chave, ordenando pela data DESC.

⚠ Nenhum deles é suportado; chaves só NOT ENFORCED, recursão não existe nem com MAXRECURSION, e ALTER TABLE não adiciona coluna NOT NULL.

GABARITO

# Respostas do diagnóstico

**D1. D) AVERAGEX(WINDOW(-2, REL, 0, REL, ALLSELECTED(Data[AnoMes]), ORDERBY(Data[AnoMes])), [Sacas])** → revise o capítulo 1

Por quê: As funções de janela do DAX (WINDOW, OFFSET, INDEX) trabalham sobre uma relação ordenada. WINDOW recebe início e fim, cada um relativo (REL, padrão) ou absoluto (ABS), além de ORDERBY e, opcionalmente, PARTITIONBY para reiniciar por grupo.

✗ A) Incorreta. De 0 a 2 relativo pega o mês atual e os dois seguintes, não os anteriores.

✗ B) Incorreta. Posições absolutas (ABS) 1 a 3 pegam sempre os três primeiros meses, não uma janela que acompanha a linha.

✗ C) Incorreta. OFFSET devolve uma única linha (dois meses antes), não o intervalo de três meses.

> **Dica de prova:** Média móvel em DAX: AVERAGEX sobre WINDOW(-n, REL, 0, REL, ...).

**D2. D) Remover o prefixo SO e converter a coluna para número inteiro, com Do Not Summarize** → revise o capítulo 1

Por quê: Técnicas de redução de modelos Import incluem remover colunas e linhas desnecessárias, agrupar e resumir, otimizar tipos de dados, preferir colunas personalizadas no Power Query, desabilitar carga de consultas auxiliares e desligar Auto date/time.

✗ A) Incorreta. Uma tabela à parte com a mesma cardinalidade não reduz o modelo e acrescenta uma relação.

✗ B) Incorreta. Manter o texto e acrescentar uma coluna calculada aumenta o modelo.

✗ C) Incorreta. Duas colunas de texto não reduzem a cardinalidade do número.

> **Dica de prova:** Coluna de alta cardinalidade em texto: converta para inteiro quando possível.

**D3. A) Get the latest data in real time with DirectQuery, criando uma partição DirectQuery após o período** → revise o capítulo 1

Por quê: A política de atualização incremental divide a tabela em partições históricas e de atualização com base em RangeStart e RangeEnd. Opcionalmente, uma partição DirectQuery cobre o período após o refresh, formando uma tabela híbrida com dados em tempo real.

✗ B) Incorreta. Only refresh complete days faz o oposto: considera só dias completos no refresh.

✗ C) Incorreta. Detect data changes usa uma coluna de auditoria diferente da de partição e só reduz o que é atualizado; não traz dados do dia.

✗ D) Incorreta. O período de arquivo define quanto histórico é guardado; não torna o histórico DirectQuery.

> **Dica de prova:** Import com dados do dia sem refresh extra: partição DirectQuery da atualização incremental.

**D4. A) Usar Select related para incluir na implantação o modelo semântico do qual o relatório depende** → revise o capítulo 2

Por quê: Durante a implantação, os pipelines mantêm as conexões entre itens: se o item que fornece os dados já está no estágio de destino, o item implantado é ligado a ele; se não está, a implantação falha. O botão Select related evita esse erro.

✗ B) Incorreta. Publicar por fora cria item não pareado e não resolve a dependência ausente no pipeline.

✗ C) Incorreta. Autobinding é automático, não uma configuração a ser ligada; ele precisa do modelo no estágio de destino.

✗ D) Incorreta. Regras de parâmetro não existem para relatórios; valem para modelos semânticos e dataflows.

> **Dica de prova:** Implantação de relatório falhou por falta do modelo: Select related.

**D5. D) O SELECT * dá erro no SQL analytics endpoint; no Spark, a leitura traz as colunas permitidas** → revise o capítulo 2

Por quê: A OneLake security aplica CLS e RLS nos motores do Fabric. Para CLS, o Spark mostra só as colunas autorizadas; o SQL analytics endpoint e os modelos semânticos retornam erro quando a consulta pede colunas proibidas, como num SELECT *.

✗ A) Incorreta. A CLS restringe colunas; consultas às colunas permitidas funcionam.

✗ B) Incorreta. O comportamento difere por motor: o SQL analytics endpoint gera erro no SELECT *.

✗ C) Incorreta. Não há substituição por nulos; o endpoint bloqueia, e o Spark filtra.

> **Dica de prova:** CLS da OneLake security: Spark filtra colunas; SQL endpoint dá erro no SELECT *.

**D6. C) O Test as role usa a identidade de quem testa; validar com a conta do convidado e o UPN em um cartão** → revise o capítulo 2

Por quê: A tabela de mapeamento da RLS dinâmica deve conter exatamente o valor que USERPRINCIPALNAME() devolve. Para convidados B2B, esse valor pode ser o e-mail ou um UPN no formato #EXT#, e o Test as role não reproduz a identidade do convidado.

✗ A) Incorreta. O Test as role funciona em Import; a limitação documentada é com DirectQuery com SSO.

✗ B) Incorreta. USERNAME() também reflete a identidade de quem testa e, para convidados, costuma devolver o mesmo valor.

✗ D) Incorreta. RLS se aplica a convidados B2B; o cuidado é com o formato do UPN e com grupos que contêm convidados.

> **Dica de prova:** Convidado sem dados com RLS dinâmica: teste com a conta real e mostre o UPN em um cartão.

**D7. A) O editor visual executa só consultas de leitura; o DELETE deve ir para o editor SQL** → revise o capítulo 3

Por quê: O visual query editor monta consultas sem código com o diagrama do Power Query e só executa SELECT. O SQL query editor executa DDL, DML e DCL, com limitações para controle de transações entre execuções separadas.

✗ B) Incorreta. Filtrar linhas no editor visual muda o resultado da consulta, não os dados da tabela.

✗ C) Incorreta. Não há modo que libere DDL ou DML no editor visual.

✗ D) Incorreta. Salvar a consulta em uma pasta não muda o que ela pode executar.

> **Dica de prova:** Alterar dados no warehouse: SQL query editor, não o editor visual.

**D8. A) Com RLS ou OLS no modelo, só quem tem escrita, como Contributor, lê as tabelas exportadas** → revise o capítulo 3

Por quê: Com a OneLake integration, as tabelas Import do modelo são gravadas em Delta no OneLake após cada refresh. Para não expor dados protegidos, se o modelo tem RLS ou OLS, apenas quem tem escrita no modelo lê esses arquivos.

✗ B) Incorreta. Medidas, tabelas de grupo de cálculo e tabelas DirectQuery não são exportadas; as tabelas Import são.

✗ C) Incorreta. A OneLake integration exporta justamente as tabelas Import do modelo depois de um refresh.

✗ D) Incorreta. É possível criar atalhos para as tabelas exportadas, inclusive de outros workspaces.

> **Dica de prova:** Modelo com RLS exportado ao OneLake: Read e Build não bastam para ler o Delta.

**D9. D) Cada Run é um lote em sessão própria; a transação deve estar inteira em uma única execução** → revise o capítulo 3

Por quê: O SQL query editor executa DDL, DML e DCL, mas cada execução é um lote em nova sessão: SET de sessão, sp_set_session_context e transações divididas em execuções separadas não funcionam como no SSMS. GO também inicia um lote novo.

✗ A) Incorreta. Não há exigência de papel para COMMIT.

✗ B) Incorreta. O warehouse suporta transações; a limitação é do editor quando os comandos são enviados em execuções separadas.

✗ C) Incorreta. Não há esse tempo de expiração documentado; a sessão termina ao fim do lote.

> **Dica de prova:** Transação no editor SQL do Fabric: tudo em uma execução só.


## Agora, treine no formato real

Faça as 6 provas completas do curso DP-600: 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.