

Guia de preparação DP-900

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.

4 capítulos por domínio	**28** 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-900: 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 45 minutos, no formato do exame. Referência de aprovação: 70%.

Domínio (capítulo)	Peso	Neste guia
1. Descrever conceitos fundamentais de dados	25-30%	18 conceitos · 14 pegadinhas
2. Descrever considerações para dados não relacionais no Azure	15-20%	18 conceitos · 14 pegadinhas
3. Descrever uma carga de trabalho de análise	30-35%	18 conceitos · 14 pegadinhas
4. Identificar considerações para dados relacionais no Azure	20-25%	18 conceitos · 14 pegadinhas

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

Estude um pouco todo dia, de graça

Instagram @strecht.simulados

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

Seguir no Instagram →

YouTube Strecht Simulados

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

Inscrever-se no canal →

DIAGNÓSTICO

Você já sabe isso?

Responda às 12 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 · Descrever conceitos fundamentais de dados

Qual profissional é responsável por criar pipelines que extraem, transformam e carregam dados de várias fontes em um repositório analítico?

A. Administrador de banco de dados
B. Analista de dados
C. Arquiteto de segurança
D. Engenheiro de dados

D2 · Descrever conceitos fundamentais de dados

Qual profissional automatiza as cargas de dados e monitora falhas nos pipelines que alimentam o data lake?

A. Engenheiro de dados
B. Analista de dados
C. Administrador de rede
D. Designer de relatórios

D3 · Descrever conceitos fundamentais de dados

Um banco não relacional organiza os dados em linhas e famílias de colunas, e cada linha pode ter colunas diferentes. Que tipo de banco é esse?

A. Banco de grafos
B. Banco de documentos
C. Banco chave-valor
D. Família de colunas

D4 · Descrever considerações para dados não relacionais no Azure

Uma região do Azure com a conta do Azure Cosmos DB fica indisponível. O que a distribuição global com failover automático garante?

A. A conta é convertida para uma única região
B. Os dados da região afetada são apagados
C. As gravações ficam suspensas até a volta
D. Outra região replicada assume as operações

D5 · Descrever considerações para dados não relacionais no Azure

Qual nível de consistência do Azure Cosmos DB oferece a menor latência, sem garantia de ordem nas leituras?

A. Sessão
B. Eventual
C. Forte
D. Prefixo

D6 · Descrever considerações para dados não relacionais no Azure

Um usuário apagou por engano um blob importante ontem. Qual recurso, se habilitado, permite recuperá-lo?

A. Camada de acesso Archive da conta
B. Exclusão reversível de blobs
C. Assinatura de acesso SAS
D. Namespace hierárquico

D7 · Descrever uma carga de trabalho de análise

No Real-Time Intelligence do Fabric, qual item permite salvar consultas KQL, executá-las e compartilhar os resultados com a equipe?

A. Dataflow Gen2 do Factory
B. Pipeline de cópia agendado
C. Conjunto de consultas KQL
D. Relatório paginado do BI

D8 · Descrever uma carga de trabalho de análise

Qual formato aberto adiciona transações ACID, versionamento e controle de esquema aos arquivos Parquet de um data lake?

A. Formato Avro
B. Delta Lake
C. Formato CSV
D. Formato XML

D9 · Descrever uma carga de trabalho de análise

Qual modo de armazenamento do Power BI lê as tabelas Delta direto do OneLake, sem importar os dados e sem enviar consultas SQL à fonte?

A. Direct Lake
B. Conexão ao vivo
C. DirectQuery
D. Importação

D10 · Identificar considerações para dados relacionais no Azure

Qual cláusula SQL apresenta o resultado de uma consulta em ordem decrescente de valor?

A. WHERE Valor DESC
B. ORDER BY Valor DESC
C. GROUP BY Valor DESC
D. HAVING Valor DESC

D11 · Identificar considerações para dados relacionais no Azure

A tabela Clientes já existe e precisa ganhar a coluna Email. Qual instrução deve ser usada?

A. INSERT INTO Clientes (Email) VALUES ('a@b.com')
B. CREATE TABLE Clientes (Email VARCHAR(100))
C. ALTER TABLE Clientes ADD Email VARCHAR(100)
D. UPDATE Clientes SET Email = ''

D12 · Identificar considerações para dados relacionais no Azure

Uma aplicação usa PostgreSQL e a equipe quer um serviço gerenciado no Azure, com backups automáticos e alta disponibilidade, sem mudar de mecanismo. Qual opção atende?

A. Azure Cosmos DB for Apache Gremlin
B. Instância Gerenciada de SQL do Azure
C. Banco de Dados do Azure para PostgreSQL
D. Armazenamento de Tabelas do Azure

01

CAPÍTULO 1 · 25-30% DA PROVA

Descrever conceitos fundamentais de dados

Aqui eu preciso classificar dados (estruturado, semiestruturado, não estruturado), reconhecer os formatos de arquivo (CSV, JSON, XML, Parquet, ORC, Avro), separar carga transacional (OLTP) de carga analítica, ligar cada letra do ACID a um cenário concreto e saber quem faz o quê entre DBA, engenheiro de dados e analista de dados. A prova traz um cenário curto e pede a classificação, o formato, a propriedade ou o papel certo, muitas vezes em 'Escolha DUAS'. Onde mais erro: achar que CSV é semiestruturado só por ser arquivo, tratar JSON como estruturado, trocar atomicidade por consistência e misturar engenheiro com analista. Também cai bastante a diferença entre os quatro tipos de NoSQL e entre esquema na gravação (relacional) e esquema na leitura (data lake). É um domínio de 25-30%, então cada detalhe vale ponto.

Neste capítulo

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

Tópicos-chave

Dados estruturados

Seguem esquema fixo: todos os registros têm as mesmas colunas, com os mesmos tipos, como uma tabela de funcionários com matrícula, cargo e salário. A classificação depende da organização, não do tipo de arquivo: um CSV exportado do ERP com as mesmas colunas em todas as linhas é estruturado. Vantagem: consultar e validar com SQL é simples. Limitação: incluir um campo exige alterar o esquema.

Dados semiestruturados

Têm campos nomeados e hierarquia, mas não exigem o mesmo esquema em todos os registros. JSON, XML e YAML são os exemplos clássicos; um pedido em JSON em que só alguns têm o campo cupom é semiestruturado. Aceitam listas aninhadas no mesmo registro (telefones e endereços dentro do cliente).

Dados não estruturados

Não têm campos definidos: imagens, áudios, vídeos, PDFs e texto livre (conversas de chat, e-mails, comentários). Costumam ficar em armazenamento de objetos, como o Blob Storage. Para analisar, é comum extrair atributos com serviços de IA (reconhecimento de texto, sentimento, entidades) e gravar o resultado de forma estruturada.

Esquema, schema-on-write e schema-on-read

Esquema é a definição da estrutura, dos campos e dos tipos. Bancos relacionais aplicam o esquema na gravação (schema-on-write), antes de aceitar os dados. Data lakes aplicam na leitura (schema-on-read): a carga é rápida e flexível, mas o trabalho de interpretar e validar vai para a consulta e dados inválidos podem entrar.

Metadados

São dados que descrevem outros dados, como tamanho, data, autor, esquema e classificação. Ajudam a encontrar, entender e governar os dados. Não confundir com réplicas, dados apagados na lixeira ou dados de sensores.

Formatos de texto: CSV, JSON e XML

CSV é texto delimitado, costuma ter cabeçalho na primeira linha, abre em planilhas e não guarda tipos; valor com vírgula vai entre aspas (ou troca-se o delimitador, como ponto e vírgula). JSON usa objetos com pares nome-valor e listas, é autodescritivo e é o formato das APIs REST. XML delimita cada campo com marcas de abertura e fechamento. Os três são legíveis em editor de texto.

Formatos binários: Parquet, ORC e Avro

Parquet e ORC são colunares, binários, compactados e guardam tipos: permitem ler só as colunas necessárias e são ideais para análise, mas ruins para editar linhas individuais com frequência. Avro é binário orientado a linhas, guarda o esquema junto com os dados e é comum na ingestão e serialização de eventos.

Formatos abertos (Parquet e Delta)

Formatos abertos permitem que vários mecanismos leiam os mesmos dados sem conversão nem cópias. No Fabric, o formato Delta Parquet é o que deixa todos os mecanismos usarem a mesma cópia. Continuam tendo esquema e são binários, não legíveis em editor de texto.

BLOB

Objeto binário grande, como imagem, vídeo ou PDF. No Azure, fica no Armazenamento de Blobs, que guarda esses objetos de forma escalável e barata. Fotos de produtos vão para armazenamento de objetos, não para tabelas normalizadas de SQL.

Carga transacional (OLTP)

Registra as operações do dia a dia na hora em que acontecem (compra no caixa, pagamento de fatura, atualização de endereço). Tem muitas operações pequenas e concorrentes de leitura e gravação, exige baixa latência e garantias ACID, usa dados normalizados e foca no estado atual. Serviços típicos: Banco de Dados SQL do Azure, PostgreSQL, MySQL e Cosmos DB.

Carga analítica (OLAP)

Lê e agrega grandes volumes de dados históricos de várias fontes para apoiar decisões, como comparar faturamento por região em cinco anos. É otimizada para leitura, recebe cargas periódicas (como lotes noturnos) e usa modelos desnormalizados, como o esquema estrela, ou cubos que pré-agregam medidas por dimensões. Serviços típicos: Microsoft Fabric, Azure Databricks e Power BI.

Transação, COMMIT, ROLLBACK e log

Transação é um conjunto de operações tratado como unidade: ou tudo é aplicado, ou nada. COMMIT confirma; ROLLBACK desfaz as operações da transação, sem apagar tabelas nem repetir nada. A durabilidade é garantida pelo log de transações em armazenamento persistente, que permite refazer o confirmado e desfazer o que ficou pela metade após uma falha.

Propriedades ACID

Atomicidade: tudo ou nada (débito desfeito se o crédito falhar). Consistência: o banco passa de um estado válido a outro, respeitando regras (saldo negativo rejeitado). Isolamento: transações simultâneas não veem efeitos parciais umas das outras (último item do estoque). Durabilidade: o que foi confirmado sobrevive a falhas (queda de energia).

Normalização

Separa entidades em tabelas próprias ligadas por chaves, para que cada dado fique em um só lugar e seja atualizado uma única vez. Reduz duplicação e anomalias, por isso é típica de OLTP. O custo é exigir mais joins em consultas de relatório, e por isso modelos analíticos desnormalizam. Em bancos relacionais, tabela = entidade, coluna = atributo e linha = instância.

Tipos de banco NoSQL

São quatro: chave-valor (acesso só por uma chave conhecida, valor opaco, como carrinho por ID de sessão), documento (entidade inteira em JSON consultável por campo, como um pedido completo), família de colunas (linhas com colunas agrupadas e esparsas, como Cassandra) e grafo (nós e arestas, para redes sociais, recomendações e fraude).

Esquema flexível e escala horizontal

Bancos NoSQL aceitam registros com estruturas diferentes, úteis quando os dados mudam com frequência. Escalam horizontalmente, distribuindo partições entre servidores; a escala vertical (servidor cada vez maior) tem limite e é mais comum em relacionais tradicionais. Em bancos de documentos, dados lidos juntos são incorporados no mesmo documento para ler tudo em uma operação, sem joins, mesmo que isso gere alguma duplicação.

Papéis: DBA, engenheiro de dados e analista de dados

O DBA cuida de disponibilidade, desempenho, backup, recuperação, segurança e permissões dos bancos. O engenheiro de dados cria, automatiza e monitora pipelines de ingestão e transformação e mantém o armazenamento analítico. O analista de dados explora os dados, identifica tendências, cria modelos semânticos, medidas DAX e relatórios. Sequência típica: o engenheiro prepara e o analista visualiza.

Dados em repouso, streaming e séries temporais

Dados em repouso já estão gravados em arquivos ou bancos e são processados em lote; dados em movimento chegam de forma contínua e são processados em streaming. Série temporal é a sequência de valores com carimbo de data e hora, comum em IoT e monitoramento. Manter os dados brutos no data lake permite reprocessar se as regras de transformação mudarem.

Macetes para memorizar

ACID

Atomicidade = tudo ou nada; Consistência = regra respeitada; Isolamento = cada um na sua; Durabilidade = confirmou, ficou.

Formatos de arquivo

Texto legível: CSV, JSON, XML. Binários: Parquet, ORC, Avro. Colunares só Parquet e ORC; Avro é linha com esquema.

Classificação

Tem coluna fixa? Estruturado. Tem nome de campo que varia? Semi. Não tem campo? Não estruturado.

Papéis

Engenheiro prepara, analista visualiza, DBA protege e mantém de pé.

Normalização

Normalizado = OLTP = um dado, um lugar, uma atualização. Desnormalizado = análise = menos joins.

Esquema

Relacional valida na entrada (write); data lake interpreta na saída (read).

Horizonte temporal

OLTP: agora. Análise: histórico.

NoSQL

Quatro tipos: chave-valor, documento, família de colunas e grafo; o resto da alternativa é distrator.

Tabela de memorização: não confunda

Quando é...	...e quando é
Dados estruturados Todas as linhas têm as mesmas colunas e tipos, mesmo que estejam em um arquivo CSV.	**Dados semiestruturados** Há campos nomeados, mas eles variam entre registros ou há listas aninhadas (JSON, XML, YAML).

Dados semiestruturados Existem chaves ou marcas nomeadas que dão alguma estrutura.	**Dados não estruturados** Não há campo algum: imagem, áudio, vídeo, texto livre de chat ou e-mail.
OLTP Muitas transações pequenas e concorrentes, ACID, dados normalizados, estado atual (registrar compra, pagar fatura).	**Carga analítica** Leitura e agregação de histórico de várias fontes, cargas periódicas, modelo desnormalizado (relatório anual, tendências, treinar modelo).
Parquet / ORC Formato colunar e compactado para consultas analíticas que leem só algumas colunas.	**Avro** Formato binário orientado a linhas, com esquema embutido, para gravação e serialização de eventos.
Atomicidade O cenário fala em tudo ou nada: se uma parte falha, a outra é desfeita.	**Consistência** O cenário fala em regra ou restrição respeitada: transação que violaria a regra é rejeitada.
Isolamento Duas transações ao mesmo tempo não podem ver o efeito parcial uma da outra.	**Durabilidade** Depois de confirmada, a transação não se perde, mesmo com queda de energia.
Engenheiro de dados Pipelines de ingestão, limpeza e transformação em escala, monitorar falhas de carga.	**Analista de dados** Tendências e padrões, modelos semânticos, medidas DAX, relatórios e visualizações.

Números que caem

Item	Valor
Peso do domínio no exame	**25-30%**
Tipos principais de banco NoSQL	**4 (chave-valor, documento, família de colunas, grafo)**
Propriedades ACID	**4 (atomicidade, consistência, isolamento, durabilidade)**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Achar que um CSV é semiestruturado ou não estruturado só porque é um arquivo.
Como não cair: A classificação depende da organização: colunas fixas com o mesmo tipo em todas as linhas é dado estruturado, esteja em tabela ou em arquivo.

⚠ **Alerta de prova:** Classificar JSON como estruturado porque tem campos nomeados.
Como não cair: Estruturado exige o mesmo esquema em todos os registros; JSON permite campos que variam, por isso é semiestruturado.

⚠ **Alerta de prova:** Marcar 'Colunares' para um YAML ou JSON.
Como não cair: Colunar é forma de armazenamento (Parquet, ORC), não classificação de estrutura; YAML, JSON e XML são semiestruturados.

⚠ **Alerta de prova:** Dizer que texto livre de chat é semiestruturado ou 'relacional'.
Como não cair: Sem campos nomeados é não estruturado; relacional é modelo de banco, não classificação de dado.

⚠ **Alerta de prova:** Acreditar que a normalização faz as consultas analíticas dispensarem joins.
Como não cair: É o contrário: tabelas normalizadas exigem mais joins. Tabela única e larga é desnormalização.

⚠ **Alerta de prova:** Achar que incorporar os itens dentro do documento do pedido evita duplicação ou garante esquema igual.
Como não cair: A vantagem é ler o pedido inteiro em uma operação, sem joins; incorporar pode até gerar alguma duplicação.

⚠ **Alerta de prova:** Pensar que schema-on-read dispensa processamento na consulta ou impede dados inválidos.
Como não cair: A interpretação acontece na consulta (exige processamento) e, sem validação na gravação, dados inválidos podem entrar.

⚠ **Alerta de prova:** Escolher atomicidade para o pedido confirmado que não pode se perder na queda de energia.
Como não cair: Confirmado e sobrevive a falha é durabilidade; atomicidade é o tudo ou nada antes da confirmação.

⚠ **Alerta de prova:** Aceitar normalização, replicação ou escalabilidade como propriedade ACID.
Como não cair: ACID são só atomicidade, consistência, isolamento e durabilidade; o resto é distrator.

⚠ **Alerta de prova:** Marcar 'cargas em lote noturnas' ou 'esquema estrela' como característica de OLTP.
Como não cair: Lotes noturnos, agregações sobre histórico e esquema estrela são de carga analítica; OLTP é transação pequena, concorrente e ACID.

⚠ **Alerta de prova:** Dizer que banco chave-valor não aceita JSON ou não aguenta grande volume.
Como não cair: O valor pode ser JSON (tratado como opaco) e escala bem; a limitação real é consultar de forma eficiente só pela chave.

⚠ **Alerta de prova:** Escolher banco de grafos para operações que atualizam clientes, pedidos e pagamentos na mesma transação.
Como não cair: Relações fortes com transação entre várias tabelas pedem banco relacional; grafo é para percorrer redes de relações.

⚠ **Alerta de prova:** Achar que NoSQL escala trocando por um servidor cada vez maior.
Como não cair: NoSQL escala horizontalmente, somando nós e partições; escala vertical tem limite.

⚠ **Alerta de prova:** Atribuir relatórios ou medidas DAX ao DBA ou ao engenheiro, ou backup ao analista.
Como não cair: Relatórios e DAX são do analista, pipelines do engenheiro, backup, alta disponibilidade e permissões do DBA.

Como cai na prova

📝 **Como cai:** Cenário descreve um conjunto de dados (tabela, JSON, foto, áudio, YAML) e pede a classificação.
Como resolver: Pergunte: há campos? Se não, não estruturado. Se há, os campos são iguais em todos os registros? Se sim, estruturado (mesmo em arquivo); se variam, semiestruturado.

📝 **Como cai:** Cenário de transação com falha ou concorrência e pergunta qual propriedade ACID se aplica.
Como resolver: Localize o verbo: desfez tudo (atomicidade), rejeitou por regra (consistência), simultâneo sem interferência (isolamento), confirmou e não perdeu (durabilidade). Descarte termos que não são ACID.

📝 **Como cai:** Descrição de tarefas e pergunta qual papel é responsável.
Como resolver: Pipeline, ingestão, transformação: engenheiro. Relatório, tendência, DAX: analista. Backup, disponibilidade, acesso ao banco: DBA. Cuidado com alternativas de papéis invertidos.

📝 **Como cai:** 'Escolha DUAS' sobre formatos: colunares, legíveis em editor ou semiestruturados.
Como resolver: Colunares: Parquet e ORC. Legíveis: CSV, JSON, XML. Semiestruturados: JSON e XML. Avro é sempre a isca: binário e por linha.

📝 **Como cai:** Cenário de negócio e pergunta se é carga transacional ou analítica.
Como resolver: Registrar ou atualizar na hora é OLTP; somar, comparar períodos, relatório mensal ou treinar modelo com histórico é analítica.

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
imagem, foto, áudio, vídeo, raio-x, texto livre de conversa	**Dado não estruturado**
JSON em que alguns registros têm um campo e outros não, listas aninhadas, YAML	**Dado semiestruturado**
mesmas colunas e mesmo tipo em todas as linhas (inclusive CSV do ERP)	**Dado estruturado**
colunar e compactado, ler só as colunas necessárias	**Parquet (ou ORC)**
binário, orientado a linhas, esquema junto dos dados, ingestão de eventos	**Avro**
texto delimitado com cabeçalho, sem tipos / marcas de abertura e fechamento	**CSV / XML**
débito desfeito porque o crédito falhou	**Atomicidade**
regra que proíbe saldo negativo, transação rejeitada	**Consistência**
dois vendedores reservando o último item ao mesmo tempo	**Isolamento**
queda de energia depois da confirmação	**Durabilidade (log de transações)**
pipeline de ingestão, carga automatizada, falha na carga do data lake	**Engenheiro de dados**
backup, disponibilidade, logins e permissões dos bancos de produção	**Administrador de banco de dados (DBA)**
recuperar pelo ID da sessão, sem consultar outros campos	**Banco chave-valor**
nós e arestas, quem comprou o quê, recomendação, rede social, fraude	**Banco de grafos**

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

- CSV com colunas fixas é estruturado; JSON, XML e YAML são semiestruturados; imagem, áudio, vídeo e texto livre são não estruturados.
- Parquet e ORC são colunares; Avro é binário por linha com esquema; CSV não guarda tipos.
- Queda de energia após confirmar = durabilidade; tudo ou nada = atomicidade; regra respeitada = consistência; simultâneo = isolamento.
- Normalização serve ao OLTP e custa joins; análise usa modelo desnormalizado.

Essenciais para a prova (checklist)

☐ Sei classificar qualquer exemplo em estruturado, semiestruturado ou não estruturado, inclusive CSV e YAML.

☐ Sei dizer quais formatos são colunares, quais são texto legível e qual guarda esquema por linha.

- ☐ Sei associar cada letra do ACID a um cenário concreto e descartar distratores como replicação e normalização.
- ☐ Sei explicar COMMIT, ROLLBACK e o papel do log de transações na durabilidade.
- ☐ Sei diferenciar OLTP de carga analítica por volume, horizonte temporal e modelagem.
- ☐ Sei a vantagem e a desvantagem da normalização.
- ☐ Sei os quatro tipos de NoSQL e um cenário típico de cada.
- ☐ Sei a diferença entre schema-on-write e schema-on-read.
- ☐ Sei atribuir tarefas a DBA, engenheiro de dados e analista de dados.

Questões de revisão do capítulo

R1.1 · Descrever conceitos fundamentais de dados

O que é o esquema de um conjunto de dados?

A. A cópia de segurança feita diariamente pelo DBA
B. O relatório com os gráficos gerados pelos dados
C. A definição da estrutura, campos e tipos
D. O conjunto de usuários que podem acessar

R1.2 · Descrever conceitos fundamentais de dados

Qual dos itens abaixo é um exemplo de dado não estruturado?

A. Gravação em vídeo de uma reunião
B. Documento JSON com dados de pedido
C. Arquivo XML com notas fiscais
D. Tabela de produtos com preço e código

R1.3 · Descrever conceitos fundamentais de dados

Qual é uma vantagem de aplicar o esquema apenas na leitura (schema-on-read), como em um data lake?

A. Carregar os dados rapidamente, sem modelar tudo antes
B. Dispensar qualquer processamento na hora da consulta
C. Impedir que os arquivos tenham formatos diferentes
D. Garantir que nenhum dado inválido seja gravado no lake

R1.4 · Descrever conceitos fundamentais de dados

Por que o formato JSON é considerado autodescritivo?

A. As colunas ficam só na primeira linha, como um cabeçalho
B. O banco valida o esquema antes de aceitar qualquer gravação
C. O arquivo guarda os dados em colunas binárias compactadas
D. Os nomes dos campos acompanham os valores em cada registro

Respostas comentadas

R1.1. C) A definição da estrutura, campos e tipos

Por quê: O esquema define como os dados estão organizados. Dados estruturados têm esquema fixo; semiestruturados, flexível; não estruturados, nenhum.

✗ A) Incorreta. Cópias de segurança são backups.
✗ B) Incorreta. Relatórios são visualizações.
✗ D) Incorreta. Usuários com acesso são definidos por permissões.

> **Dica de prova:** Esquema: estrutura, campos e tipos.

R1.2. A) Gravação em vídeo de uma reunião

Por quê: Dados não estruturados não têm organização por campos, como vídeos, imagens, áudios e documentos livres. Em geral, ficam em armazenamento de objetos, como o Blob Storage.

✗ B) Incorreta. JSON é semiestruturado.
✗ C) Incorreta. XML é semiestruturado.
✗ D) Incorreta. Tabelas com colunas definidas são dados estruturados.

> **Dica de prova:** Vídeo, áudio e imagem: não estruturados.

R1.3. A) Carregar os dados rapidamente, sem modelar tudo antes

Por quê: Schema-on-read acelera a ingestão e dá flexibilidade, mas transfere para a consulta o trabalho de interpretar e validar os dados.

✗ B) Incorreta. A interpretação ocorre na consulta, o que exige processamento.
✗ C) Incorreta. Data lakes aceitam vários formatos.
✗ D) Incorreta. Sem validação na gravação, dados inválidos podem entrar.

> **Dica de prova:** Schema-on-read: entra rápido, estrutura na leitura.

R1.4. D) Os nomes dos campos acompanham os valores em cada registro

Por quê: Formatos semiestruturados como JSON e XML carregam os nomes dos campos junto com os valores, o que permite registros com estruturas diferentes.

✗ A) Incorreta. Cabeçalho na primeira linha é típico de CSV.

✗ B) Incorreta. Validar esquema na gravação é típico de bancos relacionais.

✗ C) Incorreta. Colunas binárias compactadas descrevem formatos como Parquet.

> **Dica de prova:** Nome do campo junto do valor: autodescritivo.

02

CAPÍTULO 2 · 15-20% DA PROVA

Descrever considerações para dados não relacionais no Azure

Aqui eu preciso escolher entre os serviços de armazenamento (Blob Storage, Data Lake Storage Gen2, Azure Files, Table Storage) e dominar o Azure Cosmos DB: APIs, níveis de consistência, RU/s, modos de capacidade, chave de partição e recursos como feed de alterações, TTL e backup contínuo. A prova dá um cenário com um requisito claro (log que cresce no fim, compartilhar por 24 horas, não pode apagar por cinco anos, app MongoDB) e pede o serviço ou recurso. Onde mais erro: confundir exclusão reversível com controle de versão, achar que namespace hierárquico reduz custo, colocar pool elástico ou DTU no Cosmos DB e trocar sessão por prefixo consistente. O peso é de 15-20%, mas os itens são bem previsíveis se eu decorar os pares requisito-recurso.

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

Tópicos-chave

Armazenamento de Blobs

Armazenamento de objetos acessado por HTTP, ideal para imagens, vídeos, backups e arquivos de data lake a baixo custo. A hierarquia é conta de armazenamento, contêiner e blob; o contêiner agrupa os blobs como uma pasta de nível superior. Partição, fila e tabela são conceitos de outros serviços.

Tipos de blob

Blob de blocos: tipo geral para arquivos. Blob de acréscimo: para logs que só crescem no fim. Blob de páginas: acesso aleatório, como discos de VM. Não existe 'blob de disco gerenciado'.

Camadas de acesso

Hot para dados lidos com frequência; Cool, Cold e Archive reduzem o custo de armazenamento mas encarecem o acesso. Archive é offline e exige reidratação. A conta tem camada padrão (Hot ou Cool) e cada blob de blocos pode ter a sua própria camada, sem recriar a conta.

Gerenciamento do ciclo de vida

Políticas que movem blobs automaticamente para Cool, Cold ou Archive e podem excluir dados expirados. É a forma de reduzir custo de arquivos raramente lidos depois de um tempo, sem apagá-los.

Proteção de dados do Blob Storage

Exclusão reversível recupera blob apagado; controle de versão guarda versões anteriores e restaura conteúdo sobrescrito; instantâneos e bloqueios de recurso também ajudam. Armazenamento imutável (WORM) aplica retenção por tempo ou retenção legal: gravar uma vez, ler muitas, sem alterar nem excluir.

SAS e criptografia

Assinatura de acesso compartilhado (SAS) delega acesso a recursos de armazenamento com permissões, prazo e, opcionalmente, IPs, sem entregar a chave da conta. Todos os dados do Armazenamento do Azure são criptografados em repouso por padrão com AES de 256 bits, em qualquer redundância e camada; o cliente pode usar chaves próprias no Key Vault.

Azure Data Lake Storage Gen2

É o Blob Storage com namespace hierárquico, que traz diretórios reais, compatibilidade com Hadoop e base para Spark. Renomear um diretório com milhões de arquivos é atômico, sem copiar cada objeto. Permissões finas por diretório e arquivo usam ACLs no estilo POSIX, junto com RBAC.

Azure Files

Compartilhamentos de arquivos gerenciados montados como unidade de rede, por SMB (Windows, Linux e macOS) ou NFS (cargas Linux), ideais para levar pasta de rede legada para a nuvem sem mudar a aplicação. Tem camada padrão em HDD e premium em SSD para baixa latência e alto IOPS.

Armazenamento de Tabelas

Repositório NoSQL chave-valor barato dentro de uma conta de armazenamento, sem esquema fixo: cada entidade tem PartitionKey, RowKey e carimbo de data e hora, mais as propriedades que quiser. Só PartitionKey e RowKey são indexadas, por isso as consultas eficientes filtram por elas. Não usa chaves estrangeiras.

Azure Cosmos DB

Banco NoSQL totalmente gerenciado (PaaS), com distribuição global, leituras e gravações de baixa latência em várias regiões, escala elástica e SLAs de latência, throughput, consistência e disponibilidade, com leituras pontuais em milissegundos. É agnóstico a esquema e indexa todas as propriedades automaticamente; a política de indexação pode ser ajustada para reduzir custo.

Modelo de recursos do Cosmos DB

Hierarquia: conta, banco de dados, contêiner e itens. O contêiner é a coleção de itens com throughput e partições próprias, a unidade de escala. Itens do mesmo contêiner podem ter estruturas diferentes e várias delas podem ficar na mesma partição lógica.

APIs do Cosmos DB

NoSQL (nativa, documentos JSON, consulta no estilo SQL como SELECT * FROM c WHERE c.cidade = 'Recife'), MongoDB, PostgreSQL, Apache Cassandra (família de colunas, CQL), Apache Gremlin (grafos) e Table (chave-valor). A API é escolhida por conta, na criação; usar outra API exige outra conta e migração.

Cosmos DB for Table

Permite migrar aplicações do Table Storage sem reescrever o acesso a dados, ganhando distribuição global, indexação automática de todas as propriedades e SLAs de latência. A limitação de filtrar bem só por PartitionKey e RowKey é do Table Storage, não do Cosmos DB for Table.

Unidades de Solicitação e modos de capacidade

Toda operação consome Unidades de Solicitação; o throughput é medido em RU/s. Modo provisionado: fixo ou com escala automática até um limite definido (picos imprevisíveis); modo sem servidor: paga só as RUs consumidas (tráfego baixo e intermitente). O throughput pode ser provisionado no contêiner (dedicado) ou no banco (compartilhado entre contêineres).

Chave de partição

Distribui os dados e a carga entre partições. Deve ter muitos valores distintos e espalhar leituras e gravações de forma equilibrada, como o ID do usuário. Valor fixo, data igual para muitos ou país com poucos valores concentram a carga.

Níveis de consistência

Cinco, do mais forte ao mais relaxado: forte (sempre a versão mais recente confirmada, maior latência), desatualização limitada (atraso máximo em versões ou segundos), sessão (padrão, o cliente lê as próprias gravações), prefixo consistente (pode atrasar, mas nunca fora de ordem) e eventual (menor latência, sem garantia de ordem). Quanto mais forte, maior latência e custo.

Feed de alterações, TTL e backup contínuo

Feed de alterações permite reagir a inserções e alterações em tempo quase real, por exemplo com Azure Functions. Vida útil (TTL) no contêiner ou no item exclui itens expirados sem consumir RUs do cliente. Backup contínuo permite restauração pontual após exclusões ou alterações acidentais; também existe backup periódico.

Distribuição global e failover

O Cosmos DB replica os dados entre as regiões escolhidas. Com failover automático ou gravações em várias regiões, se uma região cair outra região replicada assume e a aplicação continua operando, sem apagar dados nem converter a conta para uma região.

Macetes para memorizar

💡 Consistência do Cosmos DB

Do mais forte ao mais fraco: Forte, Desatualização limitada, Sessão, Prefixo, Eventual. Frase: 'Fui Dormir Sem Pagar a Energia'. Sessão fica no meio e é o padrão.

💡 Tipos de blob

Blocos = arquivo comum; Acréscimo = log que só cresce; Páginas = disco de VM.

💡 Camadas de acesso

Hot todo dia, Cool pouco, Cold raro, Archive offline (reidratar antes de ler).

💡 Hierarquias

Blob: conta > contêiner > blob. Cosmos: conta > banco > contêiner > itens.

💡 APIs do Cosmos DB

Mongo = documento, Cassandra = colunas, Gremlin = grafo, Table = chave-valor, NoSQL = nativa em JSON com sintaxe SQL.

💡 Data Lake Gen2

ADLS Gen2 = Blob + namespace hierárquico.

💡 Proteção de blobs

Apagou: exclusão reversível. Sobrescreveu: versão. Não pode mexer: imutável. Emprestar por prazo: SAS.

💡 Sintaxe por API

SELECT ... FROM c = NoSQL; db.x.find() = MongoDB; g.V() = Gremlin.

Tabela de memorização: não confunda

Quando é...	...e quando é
Armazenamento de Blobs Objetos acessados por HTTP: mídia, backups, data lake.	**Azure Files** Pasta de rede montada por SMB ou NFS, sem mudar a aplicação.

Exclusão reversível Blob foi apagado por engano.	**Controle de versão** Blob foi sobrescrito com conteúdo errado e é preciso voltar ao anterior.
Armazenamento de Tabelas Chave-valor simples com o menor custo, consultas por PartitionKey e RowKey.	**Azure Cosmos DB for Table** Mesma API Table com distribuição global, latência menor e indexação de todas as propriedades.
Consistência de sessão O cliente precisa ler as próprias gravações; é o padrão da conta.	**Prefixo consistente** As leituras podem atrasar, mas nunca podem ver gravações fora de ordem.
Consistência forte Toda leitura precisa da versão mais recente confirmada, aceitando maior latência.	**Consistência eventual** Menor latência e a ordem não importa, como contadores de curtidas.
Provisionado com escala automática Picos imprevisíveis em que o throughput sobe e desce sozinho até um limite.	**Sem servidor** Tráfego baixo e intermitente, pagando só as RUs consumidas.
API para Apache Gremlin Pessoas e relações, consultas que percorrem conexões (grafo).	**API para Apache Cassandra** Aplicação Cassandra existente com consultas CQL (família de colunas).

Números que caem

Item	Valor
Peso do domínio no exame	**15-20%**
Níveis de consistência do Cosmos DB	**5 (forte, desatualização limitada, sessão, prefixo consistente, eventual)**
APIs do Cosmos DB	**6 (NoSQL, MongoDB, PostgreSQL, Apache Cassandra, Apache Gremlin, Table)**
Tipos de blob	**3 (blocos, acréscimo, páginas)**
Camadas de acesso de blobs	**Hot, Cool, Cold e Archive**
Criptografia padrão em repouso do Armazenamento do Azure	**AES de 256 bits**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Escolher a camada Archive para recuperar um blob apagado.
Como não cair: Archive só guarda blobs existentes offline; recuperar apagado é exclusão reversível.

⚠ **Alerta de prova:** Achar que o namespace hierárquico reduz custo ou protege contra exclusão.
Como não cair: Ele organiza diretórios e permissões; custo cai com política de ciclo de vida e proteção vem de exclusão reversível, versão ou imutabilidade.

⚠ **Alerta de prova:** Usar ACLs para cumprir exigência regulatória de não alterar nem apagar.
Como não cair: Com ACLs um administrador ainda poderia excluir; o requisito pede armazenamento imutável (WORM).

⚠ **Alerta de prova:** Usar o feed de alterações para apagar itens antigos ou restaurar dados.
Como não cair: O feed só lista mudanças; apagar após um tempo é TTL e voltar no tempo é backup contínuo.

⚠ **Alerta de prova:** Marcar pool elástico ou Hyperscale como modo de capacidade do Cosmos DB.
Como não cair: São recursos do Banco de Dados SQL; no Cosmos DB os modos são provisionado (fixo ou escala automática) e sem servidor.

⚠ **Alerta de prova:** Dizer que o Cosmos DB mede throughput em DTU, vCore ou gigabytes transferidos.
Como não cair: O Cosmos DB usa Unidades de Solicitação (RU/s); DTU e vCore são do Azure SQL.

⚠ **Alerta de prova:** Achar que a API do Cosmos DB é escolhida por consulta, por item ou muda sozinha.
Como não cair: A API é definida por conta, na criação.

⚠ **Alerta de prova:** Escolher país, data de criação ou valor fixo como chave de partição.
Como não cair: Poucos valores ou valores repetidos concentram a carga; prefira propriedade com muitos valores distintos, como ID do usuário.

⚠ **Alerta de prova:** Responder 'partição' como o agrupador de blobs.
Como não cair: No Blob Storage quem agrupa é o contêiner; partição é conceito do Table Storage e do Cosmos DB.

⚠ **Alerta de prova:** Achar que a criptografia em repouso só vale com GRS ou camada premium.
Como não cair: Vale por padrão para todos os dados, em qualquer redundância e camada.

⚠ **Alerta de prova:** Escolher o Banco de Dados SQL para app global com gravações em várias regiões porque ele tem réplicas geográficas.
Como não cair: Gravações de baixa latência em várias regiões com replicação automática é Cosmos DB.

⚠ **Alerta de prova:** Achar que camada de acesso se aplica a blobs de páginas de VMs.
Como não cair: Camadas de acesso se aplicam a blobs de blocos, e podem ser definidas por blob.

⚠ **Alerta de prova:** Marcar prefixo consistente ou sessão quando o requisito é sempre a última versão.
Como não cair: Última versão confirmada em toda leitura é consistência forte; prefixo garante só a ordem e sessão só as próprias gravações.

⚠ **Alerta de prova:** Escolher Table Storage ou Gremlin para migrar uma aplicação MongoDB.
Como não cair: App MongoDB com mínima mudança de código vai para o Azure Cosmos DB for MongoDB.

Como cai na prova

📝 **Como cai:** Cenário de arquivos (mídia, logs, backups, pasta de rede, data lake) e pergunta qual serviço ou tipo de blob.
Como resolver: Acesso por HTTP: Blob (escolha o tipo pelo padrão de escrita). Montagem SMB ou NFS: Azure Files. Spark, diretórios e ACLs: ADLS Gen2. Pouco acesso: camada fria ou Archive.

📝 **Como cai:** Requisito sobre o que a leitura precisa garantir e pergunta o nível de consistência.
Como resolver: Última versão: forte. Atraso máximo definido: desatualização limitada. Ler as próprias gravações: sessão. Nunca fora de ordem: prefixo consistente. Mais rápido, sem ordem: eventual.

📝 **Como cai:** Empresa já usa uma tecnologia (MongoDB, Cassandra, Table Storage) e quer migrar com mínima mudança.
Como resolver: Escolha a API do Cosmos DB com o mesmo nome; para app nova com JSON, API para NoSQL. Descarte serviços relacionais.

📝 **Como cai:** Incidente com dados (apagou, sobrescreveu, gravou errado, precisa expirar) e pergunta o recurso.
Como resolver: Blob apagado: exclusão reversível. Blob sobrescrito: versão. Cosmos com dado errado: backup contínuo. Expirar: TTL. Reagir a mudança: feed de alterações.

📝 **Como cai:** Padrão de tráfego ou custo no Cosmos DB.
Como resolver: Intermitente: sem servidor. Picos imprevisíveis: escala automática. Vários contêineres pequenos: throughput no banco. Unidade de medida: RU/s.

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
log gravado continuamente no fim do mesmo arquivo	**Blob de acréscimo**
compartilhar arquivo por 24 horas sem entregar a chave da conta	**Token SAS com validade**
não pode ser alterado nem excluído por anos (regulatório)	**Armazenamento imutável (WORM)**
apagou por engano / sobrescreveu com conteúdo errado	**Exclusão reversível / controle de versão**
raramente lido depois de 90 dias, reduzir custo sem apagar	**Política de ciclo de vida para camadas frias**
pasta de rede SMB legada ou montagem NFS em Linux	**Azure Files (premium SSD se exigir baixa latência)**
data lake para Spark, Hadoop, diretórios, permissão por diretório	**ADLS Gen2 (namespace hierárquico e ACLs POSIX)**
milhões de registros simples acessados por chave, sem joins, menor custo	**Armazenamento de Tabelas**
leituras e gravações de baixa latência em várias regiões do mundo	**Azure Cosmos DB**
app MongoDB / Cassandra com CQL / rede social de amizades / JSON nativo em app nova	**Cosmos DB for MongoDB / for Apache Cassandra / for Apache Gremlin / for NoSQL**
tráfego baixo e intermitente no Cosmos DB / picos imprevisíveis	**Sem servidor / throughput com escala automática**
vários contêineres com pouco tráfego dividindo RU/s	**Throughput provisionado no banco, compartilhado**
apagar itens 24 horas após a gravação / reagir a cada inserção	**TTL / feed de alterações**
gravou dados errados há horas e quer voltar a conta para antes	**Backup contínuo com restauração**

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

- Sessão é a consistência padrão; forte é a mais lenta; eventual é a mais rápida e sem ordem.
- API do Cosmos DB é escolhida por conta; throughput é medido em RU/s; chave de partição precisa de muitos valores distintos.
- ADLS Gen2 = Blob + namespace hierárquico; Azure Files = SMB/NFS; Table Storage = PartitionKey + RowKey.
- Pool elástico, Hyperscale, DTU e vCore nunca são do Cosmos DB.

Essenciais para a prova (checklist)

- ☐ Sei escolher entre Blob Storage, ADLS Gen2, Azure Files e Table Storage a partir do cenário.
- ☐ Sei os três tipos de blob e as quatro camadas de acesso, e que a camada pode ser definida por blob.
- ☐ Sei diferenciar exclusão reversível, controle de versão, armazenamento imutável, SAS e ciclo de vida.
- ☐ Sei o que o namespace hierárquico acrescenta (diretórios, rename atômico, ACLs POSIX).
- ☐ Sei a hierarquia do Cosmos DB e que o contêiner é a unidade de escala.
- ☐ Sei as seis APIs do Cosmos DB e qual modelo de dados cada uma atende.
- ☐ Sei ordenar e explicar os cinco níveis de consistência e qual é o padrão.
- ☐ Sei escolher entre provisionado fixo, escala automática e sem servidor, e onde provisionar throughput.
- ☐ Sei escolher uma boa chave de partição.
- ☐ Sei para que servem feed de alterações, TTL e backup contínuo.

Questões de revisão do capítulo

R2.1 · Descrever considerações para dados não relacionais no Azure

Uma aplicação precisa guardar milhões de registros simples de configuração de dispositivos, acessados por chave, sem joins e com o menor custo possível. Qual serviço atende?

A. Azure Databricks com Delta Lake
B. Instância Gerenciada de SQL do Azure
C. Armazenamento de Tabelas do Azure
D. Banco de Dados do Azure para MySQL

R2.2 · Descrever considerações para dados não relacionais no Azure

A empresa quer compartilhar um arquivo do Blob Storage com um parceiro por 24 horas, sem entregar a chave da conta de armazenamento. Qual recurso atende?

A. Namespace hierárquico da conta
B. Redundância geográfica (GRS)
C. Token SAS com validade
D. Camada de acesso Archive

R2.3 · Descrever considerações para dados não relacionais no Azure

No Armazenamento de Blobs do Azure, qual elemento agrupa os blobs de forma parecida com uma pasta de nível superior?

A. Uma partição
B. Uma fila
C. Um contêiner
D. Uma tabela

R2.4 · Descrever considerações para dados não relacionais no Azure

No Azure Cosmos DB, o que é um contêiner?

A. Uma máquina virtual que executa o banco de dados
B. Um arquivo compactado com o backup da conta
C. Uma coleção de itens com throughput e partições próprias
D. Uma região do Azure reservada para o banco

Respostas comentadas

R2.1. C) Armazenamento de Tabelas do Azure

Por quê: O Armazenamento de Tabelas é um repositório NoSQL chave-valor barato para dados estruturados não relacionais, acessados pela PartitionKey e RowKey.

✗ A) Incorreta. O Databricks é plataforma de análise.
✗ B) Incorreta. Um banco relacional gerenciado custa mais e não é necessário sem joins.
✗ D) Incorreta. O MySQL é relacional.

> **Dica de prova:** Chave-valor barato em conta de armazenamento: Table Storage.

R2.2. C) Token SAS com validade

Por quê: Assinaturas de acesso compartilhado (SAS) delegam acesso a recursos de armazenamento com permissões, prazo e, opcionalmente, IPs definidos.

✗ A) Incorreta. O namespace hierárquico organiza diretórios.
✗ B) Incorreta. A redundância replica dados.
✗ D) Incorreta. A camada Archive deixa o blob offline.

> **Dica de prova:** Acesso temporário sem chave da conta: SAS.

R2.3. C) Um contêiner

Por quê: A hierarquia do Blob Storage é conta de armazenamento, contêiner e blob. Com namespace hierárquico, os contêineres ganham diretórios reais.

✗ A) Incorreta. Partição é conceito do Table Storage e do Cosmos DB.

✗ B) Incorreta. Fila é do Queue Storage.

✗ D) Incorreta. Tabela é do Table Storage.

> **Dica de prova:** Conta > contêiner > blob.

R2.4. C) Uma coleção de itens com throughput e partições próprias

Por quê: A hierarquia do Cosmos DB é conta, banco de dados, contêiner e itens. O contêiner é a unidade de escala de throughput e armazenamento.

✗ A) Incorreta. O Cosmos DB é PaaS, sem VMs gerenciadas pelo cliente.

✗ B) Incorreta. Backups não são contêineres.

✗ D) Incorreta. Regiões são locais de replicação da conta.

> **Dica de prova:** Cosmos DB: conta > banco > contêiner > itens.

03

CAPÍTULO 3 · 30-35% DA PROVA

Descrever uma carga de trabalho de análise

Este é o domínio mais pesado (30-35%) e eu preciso cobrir três frentes: os conceitos de análise (data lake, data warehouse, lakehouse, Delta Lake, ETL/ELT, esquema estrela, lote versus streaming e janelas de tempo), as plataformas (Microsoft Fabric com OneLake, Lakehouse, Warehouse, Data Factory e Real-Time Intelligence, Azure Databricks, Hubs de Eventos e Stream Analytics) e o Power BI (componentes, modelagem, modos de armazenamento e visuais). A prova descreve uma necessidade e pede o item do Fabric, o recurso do Power BI ou o visual certo. Onde mais erro: Lakehouse versus Warehouse, Eventstream versus Eventhouse, DirectQuery versus Direct Lake, relatório versus dashboard e as janelas tumbling versus hopping. Decorar os pares cenário-item resolve a maior parte.

Neste capítulo

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

Tópicos-chave

Data lake

Repositório de arquivos em vários formatos, inclusive brutos, com baixo custo e esquema aplicado na leitura. Não é painel, não registra vendas na hora e não é banco relacional com fatos e dimensões.

Data warehouse

Repositório relacional otimizado para consultas analíticas, normalmente em esquema estrela, alimentado por cargas periódicas. Diferente do data lake, que guarda arquivos de qualquer formato.

Lakehouse e Delta Lake

O lakehouse guarda os dados como arquivos abertos no data lake (Delta Lake sobre Parquet) e permite consultá-los com SQL e transações, como em um warehouse; Fabric e Databricks usam essa abordagem. Delta Lake acrescenta transações ACID, histórico de versões e aplicação de esquema; o log de transações permite viagem no tempo para consultar ou restaurar a tabela como era antes.

Arquitetura medallion

Organiza o lakehouse em bronze (dados brutos, como chegaram), prata (limpos e validados) e ouro (agregados e prontos para o negócio). Não existe camada platina. Guardar o bronze dá rastreabilidade e permite reprocessar quando a regra muda.

ETL, ELT e carga incremental

ETL: extrair, transformar e carregar (transforma antes da carga). ELT: extrair, carregar e transformar no destino, aproveitando a escala do data lake ou warehouse. Carga incremental traz só os registros novos ou alterados desde a última execução, usando marcas d'água como datas de alteração; a carga completa recarrega tudo.

Esquema estrela

Tabela de fatos no centro com medidas numéricas (quantidade, valor) e chaves; tabelas de dimensão ao redor descrevem quem, o quê, onde e quando. Desnormalizado para simplificar e acelerar consultas de BI. Granularidade é o nível de detalhe que cada linha de fato representa; chave substituta é gerada pelo warehouse, enquanto a chave natural vem da origem. Floco de neve é quando a dimensão é quebrada em tabelas encadeadas.

Lote versus streaming

Lote processa conjuntos acumulados em intervalos (folha de pagamento mensal, carga noturna): eficiente com grandes volumes, mas com resultados atrasados. Streaming processa cada evento ao chegar, com dados contínuos e sem fim definido e baixa latência (fraude em segundos, sensores, cliques).

Janelas de tempo no streaming

Em cascata (tumbling): intervalos fixos sem sobreposição. De salto (hopping): tamanho fixo que avança menos que o tamanho, com sobreposição (10 minutos atualizados a cada 2). Deslizante (sliding): move-se conforme os eventos e se sobrepõe. De sessão: blocos separados por períodos de inatividade. De instantâneo: eventos com o mesmo carimbo de tempo.

Streaming no Azure

Arquitetura típica: ingestão (Hubs de Eventos ou IoT Hub), processamento (Azure Stream Analytics, com consultas no estilo SQL e janelas, ou Fabric) e visualização (Power BI). Spark Structured Streaming processa fluxos com a mesma API de DataFrames, no Databricks e no Fabric. Uma solução tem fonte, processamento e coletor (sink), o destino dos resultados.

Microsoft Fabric e OneLake

Plataforma SaaS de análise que integra Data Factory, engenharia de dados, data warehouse, Real-Time Intelligence, ciência de dados e Power BI. O OneLake é o data lake único e lógico, um por locatário, com tabelas em Delta Parquet lidas por Spark, T-SQL, KQL e Power BI sem cópias. Itens ficam em workspaces (funções Administrador, Membro, Colaborador e Visualizador) e a computação vem de capacidades com unidades de computação, sem VMs nem clusters manuais.

Atalhos e espelhamento

Atalhos do OneLake referenciam dados em outros lakehouses, no ADLS Gen2 ou em outras nuvens sem copiá-los. Espelhamento replica continuamente, quase em tempo real, bancos operacionais como Azure SQL, Cosmos DB e Snowflake para o OneLake. Com tudo em Delta, o Fabric permite consultas T-SQL entre warehouse e lakehouse do mesmo workspace.

Lakehouse e Warehouse no Fabric

O Lakehouse combina arquivos de qualquer formato (área Files, para brutos) e tabelas Delta (área Tables), é trabalhado com Spark em notebooks e tem um endpoint de análise SQL criado automaticamente, somente leitura. O Warehouse oferece T-SQL completo, incluindo INSERT e UPDATE. Lakehouses, warehouses, eventhouses e bancos SQL armazenam dados; pipelines, notebooks e dataflows processam.

Data Factory no Fabric

Pipelines orquestram ingestão e transformação com conectores, agendamentos, dependências e monitoramento; a atividade de cópia move dados de uma fonte para um destino; gatilhos executam por agendamento ou por evento. Dataflow Gen2 transforma com Power Query em interface de baixo código. Notebooks executam código Spark (Python).

Real-Time Intelligence

Eventstream captura, transforma e roteia eventos sem código para Eventhouse, Lakehouse ou Activator. Eventhouse armazena dados de eventos em bancos KQL para consultas rápidas. Conjunto de consultas KQL salva e compartilha consultas; painel em tempo real mostra visuais de consultas KQL com atualização automática; Activator dispara ações (e-mail, alerta) quando uma condição ocorre nos dados.

Azure Databricks

Plataforma baseada em Apache Spark com notebooks colaborativos e lakehouse com Delta Lake, para engenharia de dados, ciência de dados e aprendizado de máquina. O Unity Catalog centraliza governança, permissões e linhagem entre workspaces; Delta Sharing compartilha dados com terceiros.

Componentes do Power BI

Fluxo típico: criar no Power BI Desktop, publicar no serviço e consumir no navegador, em apps ou no aplicativo móvel (iOS e Android, com layout de celular). Modelo semântico (antes conjunto de dados) reúne tabelas, relacionamentos e medidas e alimenta vários relatórios. Relatório tem uma ou mais páginas sobre um único modelo; dashboard é uma tela única com blocos de vários relatórios; aplicativo distribui pacotes de relatórios; relatório paginado (Report Builder) tem layout fixo para imprimir e exportar em PDF.

Modelagem no Power BI

Power Query (linguagem M) limpa e transforma antes do modelo; DAX cria medidas, colunas calculadas e tabelas calculadas. Coluna calculada é avaliada linha a linha e ocupa espaço; medida é dinâmica e respeita o contexto de filtro. Dimensão para fatos é um para muitos; hierarquias permitem drill down; tabela de datas contínua habilita inteligência de tempo; RLS usa funções com filtros DAX; atualização agendada renova dados importados, com gateway para fontes locais.

Modos de armazenamento e interação

Importação copia os dados para o modelo; DirectQuery consulta a fonte a cada interação; modo duplo combina os dois por tabela; Direct Lake lê as tabelas Delta direto do OneLake sem importar e sem enviar SQL à fonte. Interações: realce e filtragem cruzados entre visuais, segmentação de dados na página, filtros por visual, página ou relatório, detalhamento (drillthrough) para página de detalhe e Q&A em linguagem natural.

Macetes para memorizar

Medallion

Bronze bruto, prata limpo, ouro pronto.

ETL x ELT

A letra do meio diz o que vem primeiro depois de extrair: ETL transforma, ELT carrega.

Janelas

Tumbling tomba bloco a bloco sem sobrepor; Hopping pula menos que o tamanho e sobrepõe; Session fecha na inatividade; Sliding desliza com os eventos.

Power BI

Power Query (M) prepara, DAX calcula; Desktop cria, serviço compartilha, móvel consome.

Visuais

Linha = tempo; barra = comparar categorias; pizza/rosca = parte do todo com poucas fatias; dispersão = relação entre duas medidas; cartão = um número; KPI = valor contra meta; matriz = linhas x colunas com subtotais; cascata = do saldo inicial ao final; funil = etapas; mapa de árvore = retângulos aninhados; combinado = colunas + linha.

Tempo real no Fabric

Eventstream entra, Eventhouse guarda, KQL consulta, painel mostra, Activator age.

Streaming no Azure

Hubs de Eventos recebe, Stream Analytics processa, Power BI mostra.

Esquema estrela

Números no centro (fatos), descrições ao redor (dimensões); dimensão para fato é um para muitos.

Tabela de memorização: não confunda

Quando é...	...e quando é
ETL Transformar antes de carregar no destino.	**ELT** Carregar bruto no data lake ou warehouse e transformar lá, com a escala do destino.
Lakehouse (Fabric) Arquivos variados e tabelas Delta trabalhados com Spark; T-SQL só de leitura pelo endpoint de análise SQL.	**Warehouse (Fabric)** Equipe SQL que precisa de T-SQL completo, com INSERT e UPDATE.
Eventstream Capturar, transformar e encaminhar eventos em tempo real sem código.	**Eventhouse** Armazenar grandes volumes de telemetria e consultar com KQL.
DirectQuery Consultar a fonte a cada interação, sem copiar os dados.	**Direct Lake** Ler tabelas Delta do OneLake direto, sem importar e sem enviar SQL à fonte.
Relatório Uma ou mais páginas interativas sobre um único modelo semântico.	**Dashboard** Uma tela única com blocos fixados de vários relatórios.
Janela em cascata (tumbling) Intervalos fixos sem sobreposição, como contar a cada cinco minutos.	**Janela de salto (hopping)** Intervalos fixos que se sobrepõem, como média de dez minutos atualizada a cada dois.
Medida Total ou indicador que muda conforme o filtro do relatório.	**Coluna calculada** Valor calculado linha a linha, útil para filtrar e agrupar.

Números que caem

Item	Valor
Peso do domínio no exame	**30-35% (o maior)**
OneLake por organização	**1 por locatário**
Camadas da arquitetura medallion	**3 (bronze, prata, ouro)**

Alertas de prova (pegadinhas)

> ⚠ **Alerta de prova:** Usar pipelines de cópia para usar no Fabric dados que já estão no ADLS sem copiá-los.
> **Como não cair:** Pipeline de cópia duplica os dados; usar sem copiar é atalho do OneLake.

⚠ **Alerta de prova:** Achar que o endpoint de análise SQL do lakehouse aceita INSERT e UPDATE.
Como não cair: Ele é somente leitura; T-SQL completo é o Warehouse.

⚠ **Alerta de prova:** Confundir Direct Lake com DirectQuery.
Como não cair: DirectQuery envia consultas à fonte; Direct Lake lê os arquivos Delta do OneLake sem enviar SQL.

⚠ **Alerta de prova:** Escolher 'modo cache' como modo de armazenamento do Power BI.
Como não cair: Não existe; os modos são importação, DirectQuery, duplo e Direct Lake.

⚠ **Alerta de prova:** Escolher relatório quando o gestor quer uma tela com blocos de vários relatórios.
Como não cair: Isso é dashboard; o relatório pode ter várias páginas, mas vem de um único modelo.

⚠ **Alerta de prova:** Usar realce cruzado ou segmentação para que cada gerente veja só a sua região.
Como não cair: Restringir dados por usuário é segurança em nível de linha (RLS); realce e segmentação são interação.

⚠ **Alerta de prova:** Marcar 'platina' como camada da arquitetura medallion.
Como não cair: São só bronze, prata e ouro.

⚠ **Alerta de prova:** Colocar os valores numéricos de vendas na tabela de dimensão ou na tabela de datas.
Como não cair: Medidas numéricas ficam na tabela de fatos; dimensões (inclusive datas) descrevem.

⚠ **Alerta de prova:** Usar gráfico de pizza para comparar 25 categorias.
Como não cair: Pizza e rosca são para poucas fatias; muitas categorias pedem barras.

⚠ **Alerta de prova:** Escolher Dataflow Gen2 ou notebook para eventos em tempo real sem código.
Como não cair: Dataflow Gen2 é lote com Power Query e notebook exige código; eventos sem código é Eventstream.

⚠ **Alerta de prova:** Chamar de 'carga por espelho' a carga diária só do que mudou.
Como não cair: Espelhamento é replicação contínua; só os registros novos ou alterados é carga incremental.

⚠ **Alerta de prova:** Achar que a chave natural é gerada pelo warehouse.
Como não cair: A natural vem da origem; a gerada pelo warehouse é a chave substituta.

⚠ **Alerta de prova:** Achar que o Fabric cobra por relatório, exige VMs ou clusters Spark manuais.
Como não cair: É SaaS; workspaces usam uma capacidade com unidades de computação compartilhadas.

⚠ **Alerta de prova:** Achar que há um OneLake por workspace ou por usuário.
Como não cair: É um por locatário; workspaces são pastas lógicas dentro dele.

Como cai na prova

📝 **Como cai:** Cenário de equipe e necessidade no Fabric e pergunta qual item usar.
Como resolver: Identifique a linguagem e o tipo de dado: T-SQL completo = Warehouse; Spark e arquivos = Lakehouse/Notebook; KQL e eventos = Eventhouse; eventos sem código = Eventstream; Power Query = Dataflow Gen2; orquestração = pipeline; ação automática = Activator.

📝 **Como cai:** Descrição do que o gráfico precisa mostrar e pergunta o visual.
Como resolver: Ache o verbo: evolução no tempo (linhas), comparar categorias (barras), parte do todo (pizza/rosca, poucas fatias), relação entre medidas (dispersão), localização (mapa), um número (cartão), meta (KPI), etapas (funil), variação do início ao fim (cascata).

📝 **Como cai:** Requisito de tempo de resposta e pergunta lote ou streaming.
Como resolver: Reagir em segundos ou dados contínuos: streaming. Resultado pode esperar, volume acumulado, fechamento periódico: lote.

📝 **Como cai:** Modelo analítico com fatos e dimensões e pergunta onde fica algo ou qual a cardinalidade.
Como resolver: Valores numéricos: fatos. Atributos descritivos: dimensões. Produto uma vez e vendas muitas: um para muitos. Dimensão quebrada em cadeia: floco de neve.

📝 **Como cai:** Requisito de atualidade e desempenho e pergunta o modo de armazenamento do Power BI.
Como resolver: Copiar e atualizar por agendamento: importação. Consultar a fonte a cada clique: DirectQuery. Tabelas Delta no OneLake sem importar: Direct Lake.

📝 **Como cai:** Requisito de agregação em streaming e pergunta o tipo de janela.
Como resolver: Sem sobreposição: tumbling. Com sobreposição e tamanho fixo: hopping. Separado por inatividade: sessão.

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
T-SQL completo com INSERT e UPDATE no Fabric	**Warehouse**
Spark com JSON, imagens e tabelas Delta no Fabric	**Lakehouse (Files para brutos, Tables para Delta)**
telemetria em tempo real consultada com KQL	**Eventhouse**
capturar e encaminhar eventos sem código	**Eventstream**
disparar e-mail quando a temperatura passar de um limite	**Fabric Activator**
réplica contínua de Azure SQL ou Cosmos DB no OneLake	**Espelhamento (mirroring)**
transformar com Power Query em baixo código e carregar no lakehouse	**Dataflow Gen2**
orquestrar cópia diária com agendamento e monitoramento	**Pipelines do Data Factory (atividade de cópia + gatilho de agendamento)**
governança, catálogo, permissões e linhagem no Databricks	**Unity Catalog**
milhões de eventos por segundo / consultas estilo SQL sobre o fluxo	**Hubs de Eventos / Azure Stream Analytics**
faturas e extratos com layout fixo para imprimir em PDF	**Relatório paginado**
ano > trimestre > mês no mesmo gráfico / acumulado no ano	**Hierarquia de datas / tabela de datas contínua**
clique direito no cliente abre página de detalhes	**Detalhamento (drillthrough)**
tabela Delta como estava na semana passada	**Viagem no tempo do Delta Lake**

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

- Warehouse = T-SQL completo; Lakehouse = Spark + arquivos, com endpoint SQL só de leitura.
- Atalho usa sem copiar; espelhamento replica continuamente; carga incremental traz só o que mudou.
- Direct Lake lê Delta do OneLake; DirectQuery consulta a fonte; importação copia.
- Relatório = várias páginas de um modelo; dashboard = uma tela com blocos de vários relatórios; paginado = impressão.

Essenciais para a prova (checklist)

- ☐ Sei diferenciar data lake, data warehouse e lakehouse, e o que o Delta Lake acrescenta.
- ☐ Sei as camadas medallion e a ordem de ETL e ELT.
- ☐ Sei montar um esquema estrela e explicar granularidade, chave substituta e floco de neve.
- ☐ Sei diferenciar lote de streaming e os tipos de janela de tempo.
- ☐ Sei a arquitetura de streaming no Azure (Hubs de Eventos, Stream Analytics, Power BI).
- ☐ Sei escolher o item certo do Fabric: Lakehouse, Warehouse, Notebook, Pipeline, Dataflow Gen2, Eventstream, Eventhouse, Activator.
- ☐ Sei o que são OneLake, atalhos, espelhamento, workspaces e capacidades.
- ☐ Sei diferenciar relatório, dashboard, aplicativo, relatório paginado e modelo semântico.
- ☐ Sei os modos de armazenamento do Power BI e a diferença entre medida e coluna calculada.
- ☐ Sei escolher o visual certo para cada necessidade.

Questões de revisão do capítulo

R3.1 · Descrever uma carga de trabalho de análise

Uma equipe de desenvolvedores SQL quer um repositório analítico no Microsoft Fabric com suporte completo a T-SQL, incluindo INSERT e UPDATE. Qual item atende?

A. Warehouse
B. Dashboard
C. Notebook
D. Eventstream

R3.2 · Descrever uma carga de trabalho de análise

Qual fonte gera tipicamente dados de streaming?

A. Sensores enviando leituras a cada segundo
B. Cadastro de filiais alterado raramente
C. Backup mensal do sistema de vendas
D. Planilha de metas atualizada todo ano

R3.3 · Descrever uma carga de trabalho de análise

Qual visual é o mais indicado para mostrar as vendas de cada estado do Brasil em sua localização geográfica?

A. Linhas
B. Funil
C. Mapa
D. Cartão

R3.4 · Descrever uma carga de trabalho de análise

Qual visual mostra ao mesmo tempo o total mensal e a composição por canal de venda, com os canais sobrepostos em cada coluna?

A. Mapa de preenchimento
B. Cartão com um valor
C. Colunas empilhadas
D. Gráfico de dispersão

Respostas comentadas

R3.1. A) Warehouse

Por quê: No Fabric, o Warehouse oferece T-SQL completo. O Lakehouse é gerenciado com Spark e tem um endpoint de análise SQL só de leitura.

✗ B) Incorreta. Dashboards mostram visualizações.
✗ C) Incorreta. Notebooks usam Spark e linguagens como Python.
✗ D) Incorreta. O Eventstream ingere eventos em tempo real.

> **Dica de prova:** T-SQL completo no Fabric: Warehouse.

R3.2. A) Sensores enviando leituras a cada segundo

Por quê: Fontes de streaming incluem sensores de IoT, cliques em sites, transações financeiras e logs de aplicações, que geram eventos continuamente.

✗ B) Incorreta. Cadastros pouco alterados não são fluxos contínuos.
✗ C) Incorreta. Backups mensais são processados em lote.
✗ D) Incorreta. Metas anuais são dados estáticos.

> **Dica de prova:** Eventos contínuos (sensores, cliques): streaming.

R3.3. C) Mapa

Por quê: Mapas e mapas de preenchimento mostram dados geográficos, como vendas por estado ou cidade.

✗ A) Incorreta. Linhas mostram tendências no tempo.
✗ B) Incorreta. O funil mostra etapas de um processo.
✗ D) Incorreta. O cartão mostra um único número.

> **Dica de prova:** Dado por localização: mapa.

R3.4. C) Colunas empilhadas

Por quê: Gráficos empilhados combinam comparação entre períodos com a composição de cada total.

✗ A) Incorreta. Mapas mostram dados geográficos.

✗ B) Incorreta. O cartão mostra um número.

✗ D) Incorreta. A dispersão relaciona duas medidas.

> **Dica de prova:** Total + composição por período: colunas empilhadas.

04

CAPÍTULO 4 · 20-25% DA PROVA

Identificar considerações para dados relacionais no Azure

Neste domínio eu preciso dominar dois blocos: os conceitos relacionais (chaves, normalização, NULL, SQL com DDL, DML e DCL, joins, agregações, views, procedimentos e índices) e os serviços relacionais do Azure (família Azure SQL e os bancos gerenciados de PostgreSQL e MySQL). A prova mostra instruções SQL nas alternativas e pede a que faz o que o cenário pede, ou descreve requisitos de migração, custo e segurança e pede o serviço ou recurso certo. Onde mais erro: confundir Banco de Dados SQL com Instância Gerenciada, trocar DROP por DELETE e WHERE por HAVING, e misturar os recursos de segurança (TDE, Always Encrypted, mascaramento dinâmico, RLS, firewall e ponto de extremidade privado). O peso é de 20-25%.

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

Tópicos-chave

Chave primária

Identifica cada linha de forma única e estável; seus valores são únicos e não podem ser nulos. Pode ser simples (uma coluna) ou composta (várias colunas, como IdPedido + IdProduto). Sua função é identificar a linha: acelerar buscas é papel do índice, e criar o relacionamento é papel da chave estrangeira, que aponta para a chave primária da outra tabela.

Chave estrangeira e integridade referencial

Coluna que aponta para a chave primária de outra tabela, como IdCliente em Pedidos. Garante integridade referencial: inserir um pedido com IdCliente inexistente é rejeitado, sem criar o cliente nem mover o pedido. Não acelera consultas.

Normalização e anomalias

Objetivo: reduzir duplicação e manter a integridade, separando entidades em tabelas ligadas por chaves. Endereço repetido em 50 pedidos causa risco de atualizar só parte das linhas e deixar dados inconsistentes; normalizar leva os dados do cliente para outra tabela. Lista separada por vírgula em uma coluna viola a normalização.

Relacionamento muitos para muitos

Modelado com uma tabela associativa, como Matriculas entre Alunos e Cursos, com chaves estrangeiras para as duas e espaço para atributos da relação, como a data. Não se resolve com chave primária igual nem com tabela única repetindo tudo.

NULL

Representa ausência de valor; não é zero, não é texto vazio e não indica erro. Comparações usam IS NULL ou IS NOT NULL, não o sinal de igual.

DDL, DML e DCL

DDL define estrutura: CREATE, ALTER, DROP e RENAME. DML manipula dados: SELECT, INSERT, UPDATE e DELETE. DCL controla permissões: GRANT (conceder), DENY (negar explicitamente) e REVOKE (retirar a permissão registrada).

SELECT, WHERE, ORDER BY e LIKE

WHERE filtra linhas; ORDER BY ordena (ASC é o padrão, DESC decrescente) e não filtra nem remove repetições; DISTINCT retorna valores únicos. LIKE usa curingas: % para qualquer sequência e _ para um caractere; 'começa com Ana' é LIKE 'Ana%'. O sinal de igual não usa curinga.

Agregações, GROUP BY e HAVING

COUNT, SUM, AVG, MIN e MAX resumem várias linhas em um valor; COUNT(*) conta as linhas. Total por região é SUM com GROUP BY. WHERE filtra linhas antes do agrupamento e HAVING filtra os grupos depois da agregação, como clientes com mais de dez pedidos.

Joins

Combinam tabelas por colunas em comum, normalmente chave estrangeira e chave primária. INNER JOIN traz só as correspondências; LEFT JOIN traz todas as linhas da esquerda, com nulos quando não há par (todos os clientes, inclusive sem pedidos); CROSS JOIN combina todas as linhas com todas.

View

Consulta salva que se comporta como tabela virtual, sem copiar os dados. Simplifica consultas (junção de três tabelas usada sempre) e ajuda na segurança expondo só as colunas e linhas permitidas. Não acelera a consulta por si só e não recebe parâmetros.

Procedimento armazenado

Encapsula lógica no banco, com parâmetros e várias instruções SQL que podem ler e gravar, reutilizável por diferentes aplicações. Centraliza regras de negócio, reduz tráfego de rede e facilita o controle de permissões.

Índice

Acelera buscas e filtros, como consulta lenta por CPF em milhões de linhas. Custo: ocupa espaço e deixa inserções e atualizações mais lentas, por isso deve ser criado com base nas consultas mais importantes.

T-SQL

Transact-SQL é o dialeto da Microsoft, usado pelo SQL Server e pelos serviços Azure SQL. O padrão é ANSI e cada fornecedor tem o seu: PL/SQL (Oracle) e PL/pgSQL (PostgreSQL). KQL é outra linguagem, do Kusto.

Família Azure SQL

Três opções: SQL Server em VMs do Azure (IaaS, controle total do SO, versões específicas e software de terceiros, com patches por conta do cliente), Instância Gerenciada (PaaS com compatibilidade quase total com SQL Server) e Banco de Dados SQL (PaaS para bancos individuais e aplicações novas). Cosmos DB e Table Storage não fazem parte da família.

Instância Gerenciada de SQL do Azure

PaaS para migrar SQL Server com recursos de instância: restauração de backups nativos .bak, SQL Server Agent e consultas entre bancos. É implantada em uma sub-rede da rede virtual do cliente, com IP privado por padrão. Não permite instalar software no servidor nem pausa automaticamente.

Banco de Dados SQL do Azure: modelos e camadas

A Microsoft aplica patches, faz backups e oferece alta disponibilidade integrada, sem cluster, sempre na versão mais recente do mecanismo. Modelos de compra: DTU (pacotes de CPU, memória e E/S) e vCore (escolha separada de núcleos, memória e armazenamento). Opções: pool elástico (muitos bancos com picos diferentes), sem servidor (pausa em inatividade e cobra computação por segundo) e Hyperscale (bancos muito grandes).

Segurança no Azure SQL

TDE criptografa em repouso e vem habilitado em bancos novos; TLS protege em trânsito; Always Encrypted protege colunas de modo que nem os administradores vejam; mascaramento dinâmico mostra só parte do valor sem alterar o gravado; segurança em nível de linha filtra linhas conforme quem consulta. Acesso: regras de firewall do servidor (quais IPs), ponto de extremidade privado (IP privado via Private Link), autenticação do Microsoft Entra (identidade corporativa com MFA), auditoria (registro de eventos) e Microsoft Defender para SQL (ameaças como injeção de SQL).

Continuidade e serviços de código aberto

Restauração pontual (PITR) volta o banco a um momento dentro da retenção; replicação geográfica ativa e grupos de failover mantêm cópia legível em outra região; o Serviço de Migração de Banco de Dados do Azure migra com pouca parada. Banco de Dados do Azure para PostgreSQL e para MySQL (servidor flexível) oferecem backup, alta disponibilidade, janela de manutenção personalizada, parar e iniciar, réplicas de leitura e extensões como PostGIS.

Macetes para memorizar

Categorias de SQL

DDL mexe na estrutura (CREATE, ALTER, DROP); DML mexe no conteúdo (SELECT, INSERT, UPDATE, DELETE); DCL mexe na chave da porta (GRANT, DENY, REVOKE).

Filtros

WHERE antes do grupo, HAVING depois do grupo.

Apagar

DROP derruba a tabela; DELETE esvazia linhas.

Família Azure SQL

VM = você manda e você trabalha; MI = migração de instância; SQL Database = app nova.

Proteção de dados no Azure SQL

TDE em repouso, TLS em trânsito, Always Encrypted nem o DBA vê, máscara esconde parte, RLS esconde linhas.

Custo no Banco de Dados SQL

Muitos bancos: pool. Pouco uso: sem servidor. Banco gigante: Hyperscale.

Chaves

PK: única e não nula, olha para dentro. FK: aponta para fora e barra o órfão.

Tabela de memorização: não confunda

Quando é...	...e quando é
DROP TABLE Remover a tabela inteira, com a estrutura (DDL).	**DELETE FROM** Remover linhas mantendo a tabela (DML).
WHERE Filtrar linhas antes do GROUP BY.	**HAVING** Filtrar grupos depois da agregação, com COUNT ou SUM.
INNER JOIN Trazer só as linhas que têm correspondência nas duas tabelas.	**LEFT JOIN** Trazer todas as linhas da esquerda, inclusive sem par (clientes sem pedidos).
View Consulta salva usada como tabela virtual ou para expor só algumas colunas.	**Procedimento armazenado** Lógica executável com parâmetros e várias instruções que leem e gravam.
Banco de Dados SQL do Azure Aplicação nova, banco individual ou pool elástico, sem administrar servidores.	**Instância Gerenciada de SQL do Azure** Migrar SQL Server com recursos de instância: .bak nativo, SQL Server Agent, consultas entre bancos, rede virtual por padrão.

Instância Gerenciada de SQL do Azure Compatibilidade quase total em PaaS, com a Microsoft cuidando de patches.	**SQL Server em VM do Azure** Versão específica do SQL Server, software de terceiros ou acesso ao SO; o cliente aplica as atualizações.
Mascaramento dinâmico de dados Usuário sem privilégio vê o valor parcialmente oculto (XXXX-XXXX-XXXX-1234).	**Always Encrypted** Nem os administradores do banco podem ver o valor em texto claro.

Números que caem

Item	Valor
Peso do domínio no exame	**20-25%**
Tamanho máximo de banco na camada Hyperscale	**até 128 TB**
Opções da família Azure SQL	**3: duas PaaS (Banco de Dados SQL e Instância Gerenciada) e uma IaaS (SQL Server em VM)**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Usar CREATE TABLE para acrescentar uma coluna a uma tabela que já existe.
Como não cair: CREATE TABLE falharia porque a tabela existe; estrutura existente muda com ALTER TABLE ... ADD.

⚠ **Alerta de prova:** Usar INSERT para mudar o preço de um produto já cadastrado.
Como não cair: INSERT cria linha nova; alterar valor existente é UPDATE com WHERE (sem WHERE, altera todas as linhas).

⚠ **Alerta de prova:** Usar ORDER BY ou GROUP BY para filtrar, ou ORDER BY para remover repetições.
Como não cair: Filtrar é WHERE (linhas) ou HAVING (grupos); valores únicos é DISTINCT; ORDER BY só ordena.

⚠ **Alerta de prova:** Escrever WHERE Nome = '%Ana' para 'começa com Ana'.
Como não cair: O sinal de igual não usa curinga; o correto é LIKE 'Ana%'.

⚠ **Alerta de prova:** Escolher DROP ou DELETE para retirar uma permissão.
Como não cair: Permissão é DCL: REVOKE retira, GRANT concede, DENY nega.

⚠ **Alerta de prova:** Achar que o Banco de Dados SQL individual restaura .bak ou tem SQL Server Agent.
Como não cair: Esses recursos de instância estão na Instância Gerenciada.

⚠ **Alerta de prova:** Achar que a Instância Gerenciada permite instalar software de terceiros no servidor.
Como não cair: É PaaS, sem acesso ao servidor; controle total é SQL Server em VM.

⚠ **Alerta de prova:** Dizer que o DBA aplica os patches do mecanismo no Banco de Dados SQL do Azure.
Como não cair: No PaaS, a Microsoft atualiza automaticamente; no SQL Server em VM, o cliente cuida do SO e dos patches (o Azure Advisor só recomenda).

⚠ **Alerta de prova:** Achar que regra de firewall por IP dá acesso privado sem internet.
Como não cair: Firewall libera IPs no ponto público; acesso por IP privado é ponto de extremidade privado.

⚠ **Alerta de prova:** Usar mascaramento dinâmico para que cada vendedor veja só as próprias vendas.
Como não cair: Mascaramento oculta parte do valor sem filtrar linhas; filtrar linhas por usuário é segurança em nível de linha.

⚠ **Alerta de prova:** Achar que chave estrangeira ou view aceleram uma consulta lenta.
Como não cair: Quem acelera é o índice; a chave estrangeira garante integridade e a view não acelera por si só.

⚠ **Alerta de prova:** Aceitar valores repetidos ou vazios na chave primária.
Como não cair: Chave primária é única e não nula.

⚠ **Alerta de prova:** Tratar NULL como zero ou texto vazio.
Como não cair: NULL é ausência de valor; zero e texto vazio são valores.

⚠ **Alerta de prova:** Escolher Hyperscale para centenas de bancos pequenos ou para banco de testes usado poucas horas.
Como não cair: Muitos bancos com picos diferentes: pool elástico. Uso intermitente com pausa: sem servidor. Hyperscale é para bancos muito grandes.

Como cai na prova

📝 **Como cai:** Cenário de migração ou aplicação nova com requisitos (compatibilidade, controle do SO, gestão mínima) e pergunta a opção Azure SQL.
Como resolver: Precisa de SO ou software extra: VM. Precisa de recurso de instância (.bak, Agent, consultas entre bancos) em PaaS: Instância Gerenciada. App nova sem servidor para administrar: Banco de Dados SQL. Mecanismo PostgreSQL ou MySQL: o serviço gerenciado do mesmo mecanismo.

📝 **Como cai:** Alternativas com instruções SQL completas e pergunta qual faz o que se pede.
Como resolver: Identifique o verbo primeiro (SELECT consulta, INSERT inclui, UPDATE altera, DELETE remove linhas, ALTER muda estrutura) e depois a cláusula (WHERE, GROUP BY, HAVING, ORDER BY, LIKE).

📝 **Como cai:** Requisito de segurança ou acesso e pergunta o recurso.
Como resolver: Separe a camada: rede (firewall, ponto de extremidade privado), identidade (Microsoft Entra), dado em repouso (TDE), coluna sensível (Always Encrypted ou mascaramento), linha (RLS), monitoração (auditoria, Defender para SQL).

📝 **Como cai:** Requisito de custo, tamanho ou padrão de uso no Banco de Dados SQL.
Como resolver: Muitos bancos com picos diferentes: pool elástico. Intermitente: sem servidor. Muito grande: Hyperscale. Escolher vCores separado: vCore; pacote pronto: DTU.

📝 **Como cai:** Problema de modelagem (dado repetido, muitos para muitos, coluna que aponta para outra tabela).
Como resolver: Repetido: normalizar em outra tabela. Muitos para muitos: tabela associativa. Aponta para outra tabela: chave estrangeira. Identifica a linha: chave primária, simples ou composta.

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
restaurar .bak nativo, SQL Server Agent, consultas entre bancos, rede virtual por padrão	**Instância Gerenciada de SQL do Azure**
versão específica do SQL Server, software de terceiros, acesso ao SO	**SQL Server em Máquinas Virtuais do Azure (extensão SQL IaaS Agent para automatizar)**
aplicação nova, SQL totalmente gerenciado, sem administrar servidores	**Banco de Dados SQL do Azure**
centenas de bancos pequenos, um por cliente, picos em horários diferentes	**Pool elástico**
usado poucas horas por dia, pausa automática, cobrança por segundo	**Camada de computação sem servidor**

dezenas de terabytes, escalar e restaurar rápido	**Hyperscale**
escolher vCores e armazenamento separadamente	**Modelo baseado em vCore**
nem os administradores podem ver CPF e cartão	**Always Encrypted**
número do cartão aparece como XXXX-XXXX-XXXX-1234	**Mascaramento dinâmico de dados**
cada vendedor vê só as próprias linhas sem mudar a aplicação	**Segurança em nível de linha**
acesso por IP privado, sem passar pela internet	**Ponto de extremidade privado (Private Link)**
excluiu linhas há duas horas, voltar o banco a um momento	**Restauração pontual**
cópia secundária legível em outra região	**Replicação geográfica ativa**
PostGIS, PostgreSQL ou PHP com MySQL sem trocar de mecanismo	**Banco de Dados do Azure para PostgreSQL / para MySQL (servidor flexível)**

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

- Instância Gerenciada = .bak nativo, SQL Server Agent, consultas entre bancos e rede virtual; VM = controle do SO e software extra.
- DROP remove a tabela; DELETE remove linhas; REVOKE retira permissão; ALTER muda estrutura.
- WHERE filtra linhas antes; HAVING filtra grupos depois; LEFT JOIN mantém todos da esquerda.
- No PaaS a Microsoft aplica patches; no SQL Server em VM é o cliente.

Essenciais para a prova (checklist)

- ☐ Sei a função e as regras da chave primária, da chave composta e da chave estrangeira.
- ☐ Sei classificar instruções em DDL, DML e DCL e escolher a certa para cada cenário.
- ☐ Sei usar WHERE, HAVING, GROUP BY, ORDER BY, DISTINCT, LIKE e as funções de agregação.
- ☐ Sei a diferença entre INNER JOIN e LEFT JOIN.
- ☐ Sei quando usar view, procedimento armazenado e índice, e o custo do índice.
- ☐ Sei escolher entre SQL Server em VM, Instância Gerenciada e Banco de Dados SQL.
- ☐ Sei quando usar pool elástico, sem servidor, Hyperscale, DTU e vCore.
- ☐ Sei associar cada requisito de segurança ao recurso do Azure SQL.
- ☐ Sei para que servem restauração pontual, replicação geográfica ativa e o Serviço de Migração de Banco de Dados.
- ☐ Sei o que os servidores flexíveis de PostgreSQL e MySQL oferecem.

Questões de revisão do capítulo

R4.1 · Identificar considerações para dados relacionais no Azure

No Banco de Dados SQL do Azure, quem aplica as atualizações e correções do mecanismo de banco de dados?

A. O analista, ao publicar relatórios
B. O desenvolvedor, a cada deploy
C. A Microsoft, de forma automática
D. O DBA, em janelas agendadas

R4.2 · Identificar considerações para dados relacionais no Azure

Uma empresa quer que os usuários se conectem ao Azure SQL com as mesmas identidades corporativas, com MFA e gestão centralizada. Qual opção de autenticação atende?

A. Contas locais no servidor
B. Chaves de acesso compartilhadas
C. Logins SQL com senha local
D. Autenticação do Microsoft Entra

R4.3 · Identificar considerações para dados relacionais no Azure

Para atender a uma auditoria, a empresa precisa registrar os eventos do Banco de Dados SQL do Azure, como logins e consultas executadas. Qual recurso atende?

A. Pool elástico do SQL
B. Camada Hyperscale
C. Índice columnstore clusterizado
D. Auditoria do Azure SQL

R4.4 · Identificar considerações para dados relacionais no Azure

Qual instrução SQL altera o preço do produto de código 10 para 99,90?

A. SELECT Preco FROM Produtos WHERE Id = 10
B. UPDATE Produtos SET Preco = 99.90 WHERE Id = 10
C. INSERT INTO Produtos (Id, Preco) VALUES (10, 99.90)
D. ALTER TABLE Produtos ADD Preco DECIMAL(10,2)

Respostas comentadas

R4.1. C) A Microsoft, de forma automática

Por quê: Nos serviços PaaS do Azure SQL, a Microsoft cuida de patches, backups e alta disponibilidade. No SQL Server em VM, o cliente gerencia o SO e o SQL Server.

✗ A) Incorreta. Relatórios não interferem no mecanismo.
✗ B) Incorreta. O deploy da aplicação não atualiza o mecanismo.
✗ D) Incorreta. Atualizar o mecanismo manualmente é típico do SQL Server em VM.

> **Dica de prova:** PaaS: patches do mecanismo por conta da Microsoft.

R4.2. D) Autenticação do Microsoft Entra

Por quê: O Azure SQL aceita autenticação do SQL e do Microsoft Entra. Esta última centraliza identidades, permite MFA e dispensa senhas no banco.

✗ A) Incorreta. Contas locais não trazem gestão centralizada.
✗ B) Incorreta. Chaves compartilhadas são usadas em contas de armazenamento.
✗ C) Incorreta. Logins SQL ficam isolados no banco, sem gestão central.

> **Dica de prova:** Identidade corporativa no Azure SQL: Microsoft Entra.

R4.3. D) Auditoria do Azure SQL

Por quê: A auditoria do Azure SQL acompanha eventos do banco e grava os registros para conformidade, análise e investigação.

✗ A) Incorreta. O pool elástico compartilha recursos.
✗ B) Incorreta. Hyperscale é camada para bancos grandes.
✗ C) Incorreta. O índice columnstore acelera consultas analíticas.

> **Dica de prova:** Registrar eventos para conformidade: auditoria.

R4.4. B) UPDATE Produtos SET Preco = 99.90 WHERE Id = 10

Por quê: UPDATE altera valores de linhas existentes. Sem WHERE, a alteração vale para todas as linhas da tabela.

✗ A) Incorreta. SELECT só consulta o preço.
✗ C) Incorreta. INSERT cria uma nova linha.
✗ D) Incorreta. ALTER TABLE muda a estrutura da tabela.

> **Dica de prova:** Alterar valor existente: UPDATE com WHERE.

DIA DA PROVA

Estratégia de prova

O DP-900 é uma prova de fundamentos baseada em cenários curtos: cada enunciado descreve uma necessidade de negócio e uma palavra-chave decide a resposta (pausar, global, sem copiar, nem o DBA, T-SQL completo, fora de ordem). As alternativas erradas costumam ser serviços reais de outro contexto, como pool elástico oferecido para o Cosmos DB, RU/s misturado com DTU ou Eventstream no lugar de Eventhouse. Há itens de múltipla escolha e de múltipla resposta ('Escolha DUAS'), e muitos pedem para classificar algo (tipo de dado, formato, carga, papel, propriedade ACID). A análise (30-35%) e os conceitos fundamentais (25-30%) somam mais da metade da prova.

Como responder

✓ Conte com 40 a 60 itens em 45 minutos (a Microsoft não divulga o número exato) e pontuação de 1 a 1000 com mínimo de 700: com cerca de 50 itens, o ritmo é de menos de um minuto por item.

✓ Sublinhe mentalmente a palavra-chave do cenário antes de ler as alternativas: ela costuma apontar direto para o serviço ou recurso.

✓ Elimine primeiro as alternativas de outro serviço: pool elástico, Hyperscale, DTU e vCore são do Azure SQL; RU/s, API e consistência são do Cosmos DB; KQL é do Eventhouse; DAX é do Power BI.

✓ Em 'Escolha DUAS', avalie cada alternativa isoladamente como verdadeira ou falsa; distratores comuns são características da carga oposta (OLTP versus análise).

✓ Em itens de SQL, identifique primeiro o verbo (SELECT, INSERT, UPDATE, DELETE, ALTER, DROP, GRANT, REVOKE) e só depois a cláusula.

✓ Quando o cenário pede menor esforço de administração, prefira o serviço PaaS ou SaaS; só escolha SQL Server em VM se houver exigência de SO, versão específica ou software extra.

✓ Distribua o estudo pelos pesos: análise e conceitos fundamentais primeiro, depois relacional e não relacional.

✓ Quando o enunciado pergunta o que vale 'por padrão', lembre os padrões: sessão é a consistência padrão do Cosmos DB, o Cosmos DB indexa todas as propriedades, TDE vem habilitado em bancos novos do Azure SQL e o Armazenamento do Azure criptografa tudo em repouso.

Erros que derrubam candidatos

✗ Confundir Banco de Dados SQL com Instância Gerenciada em cenários de migração com .bak, SQL Server Agent ou consultas entre bancos.

✗ Trocar os níveis de consistência do Cosmos DB, principalmente sessão (padrão) com prefixo consistente.

✗ Escolher Lakehouse quando o requisito é T-SQL completo com escrita, ou Eventstream quando o requisito é armazenar e consultar com KQL.

✗ Classificar CSV de colunas fixas como semiestruturado ou JSON como estruturado.

✗ Misturar recursos de segurança: mascaramento dinâmico, Always Encrypted, RLS, TDE, firewall e ponto de extremidade privado.

✗ Confundir relatório com dashboard e DirectQuery com Direct Lake no Power BI.

✗ Usar ferramentas de proteção erradas no Blob Storage: Archive ou namespace hierárquico no lugar de exclusão reversível, versão ou imutabilidade.

Na semana da prova

- ☐ Refazer as provas práticas e anotar cada erro como par 'palavra-chave do cenário → resposta'.
- ☐ Revisar as tabelas de comparação de cada domínio: ACID, formatos, família Azure SQL, consistência do Cosmos DB, itens do Fabric e visuais do Power BI.
- ☐ Treinar SQL básico lendo instruções e dizendo em voz alta o que cada uma faz e a qual categoria (DDL, DML, DCL) pertence.
- ☐ Revisar os mnemônicos (consistência, medallion, janelas, papéis) na véspera, sem estudar conteúdo novo.
- ☐ No dia, controlar o ritmo: são 45 minutos para 40 a 60 itens, o que dá menos de um minuto por item com cerca de 50 itens.

REVISÃO FINAL

Folha de cola

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

1. Descrever conceitos fundamentais de dados

‣ imagem, foto, áudio, vídeo, raio-x, texto livre de conversa → **Dado não estruturado**

‣ JSON em que alguns registros têm um campo e outros não, listas aninhadas, YAML → **Dado semiestruturado**

‣ mesmas colunas e mesmo tipo em todas as linhas (inclusive CSV do ERP) → **Dado estruturado**

‣ colunar e compactado, ler só as colunas necessárias → **Parquet (ou ORC)**

‣ binário, orientado a linhas, esquema junto dos dados, ingestão de eventos → **Avro**

⚠ A classificação depende da organização: colunas fixas com o mesmo tipo em todas as linhas é dado estruturado, esteja em tabela ou em arquivo.

⚠ Estruturado exige o mesmo esquema em todos os registros; JSON permite campos que variam, por isso é semiestruturado.

⚠ Colunar é forma de armazenamento (Parquet, ORC), não classificação de estrutura; YAML, JSON e XML são semiestruturados.

2. Descrever considerações para dados não relacionais no Azure

‣ log gravado continuamente no fim do mesmo arquivo → **Blob de acréscimo**

‣ compartilhar arquivo por 24 horas sem entregar a chave da conta → **Token SAS com validade**

‣ não pode ser alterado nem excluído por anos (regulatório) → **Armazenamento imutável (WORM)**

‣ apagou por engano / sobrescreveu com conteúdo errado → **Exclusão reversível / controle de versão**

‣ raramente lido depois de 90 dias, reduzir custo sem apagar → **Política de ciclo de vida para camadas frias**

⚠ Archive só guarda blobs existentes offline; recuperar apagado é exclusão reversível.

⚠ Ele organiza diretórios e permissões; custo cai com política de ciclo de vida e proteção vem de exclusão reversível, versão ou imutabilidade.

⚠ Com ACLs um administrador ainda poderia excluir; o requisito pede armazenamento imutável (WORM).

3. Descrever uma carga de trabalho de análise

‣ T-SQL completo com INSERT e UPDATE no Fabric → **Warehouse**

‣ Spark com JSON, imagens e tabelas Delta no Fabric → **Lakehouse (Files para brutos, Tables para Delta)**

‣ telemetria em tempo real consultada com KQL → **Eventhouse**

‣ capturar e encaminhar eventos sem código → **Eventstream**

‣ disparar e-mail quando a temperatura passar de um limite → **Fabric Activator**

⚠ Pipeline de cópia duplica os dados; usar sem copiar é atalho do OneLake.

⚠ Ele é somente leitura; T-SQL completo é o Warehouse.

⚠ DirectQuery envia consultas à fonte; Direct Lake lê os arquivos Delta do OneLake sem enviar SQL.

4. Identificar considerações para dados relacionais no Azure

‣ restaurar .bak nativo, SQL Server Agent, consultas entre bancos, rede virtual por padrão → **Instância Gerenciada de SQL do Azure**

‣ versão específica do SQL Server, software de terceiros, acesso ao SO → **SQL Server em Máquinas Virtuais do Azure (extensão SQL IaaS Agent para automatizar)**

‣ aplicação nova, SQL totalmente gerenciado, sem administrar servidores → **Banco de Dados SQL do Azure**

‣ centenas de bancos pequenos, um por cliente, picos em horários diferentes → **Pool elástico**

‣ usado poucas horas por dia, pausa automática, cobrança por segundo → **Camada de computação sem servidor**

⚠ CREATE TABLE falharia porque a tabela existe; estrutura existente muda com ALTER TABLE ... ADD.

⚠ INSERT cria linha nova; alterar valor existente é UPDATE com WHERE (sem WHERE, altera todas as linhas).

⚠ Filtrar é WHERE (linhas) ou HAVING (grupos); valores únicos é DISTINCT; ORDER BY só ordena.

GABARITO

Respostas do diagnóstico

D1. D) Engenheiro de dados → revise o capítulo 1

Por quê: No DP-900, os papéis principais são: administrador de banco de dados (operação dos bancos), engenheiro de dados (pipelines e armazenamento) e analista de dados (modelos e relatórios).

✗ A) Incorreta. O DBA cuida de disponibilidade, desempenho e backup dos bancos.

✗ B) Incorreta. O analista explora os dados e cria relatórios e visualizações.

✗ C) Incorreta. O arquiteto de segurança define controles de proteção.

> **Dica de prova:** Pipeline de dados: engenheiro de dados.

D2. A) Engenheiro de dados → revise o capítulo 1

Por quê: O engenheiro de dados projeta, automatiza e opera os pipelines de ingestão e transformação, garantindo que os dados cheguem corretos.

✗ B) Incorreta. O analista consome os dados preparados.

✗ C) Incorreta. O administrador de rede cuida da conectividade.

✗ D) Incorreta. O designer de relatórios cuida da visualização.

> **Dica de prova:** Pipeline falhou: chame o engenheiro de dados.

D3. D) Família de colunas → revise o capítulo 1

Por quê: Bancos de família de colunas são indicados para grandes volumes com muitas colunas esparsas. No Azure, o Cosmos DB for Apache Cassandra atende a esse modelo.

✗ A) Incorreta. Grafos guardam nós e relações.

✗ B) Incorreta. Documentos guardam entidades em JSON.

✗ C) Incorreta. Chave-valor guarda um valor por chave.

> **Dica de prova:** Colunas agrupadas e esparsas: família de colunas.

D4. D) Outra região replicada assume as operações → revise o capítulo 2

Por quê: O Cosmos DB replica os dados entre as regiões escolhidas. Com failover automático ou gravações em várias regiões, a aplicação continua funcionando se uma região falhar.

✗ A) Incorreta. A conta não é convertida.

✗ B) Incorreta. Os dados estão replicados, não apagados.

✗ C) Incorreta. O failover permite continuar operando.

> **Dica de prova:** Região caiu: outra região replicada assume.

D5. B) Eventual → revise o capítulo 2

Por quê: A consistência eventual converge com o tempo e serve para casos em que a ordem não importa, como contadores de curtidas.

✗ A) Incorreta. Sessão garante as próprias gravações.
✗ C) Incorreta. Forte tem a maior latência.
✗ D) Incorreta. Prefixo consistente garante a ordem.

> **Dica de prova:** Mais rápido, sem ordem: eventual.

D6. B) Exclusão reversível de blobs → revise o capítulo 2

Por quê: Exclusão reversível, controle de versão e instantâneos ajudam a proteger blobs contra exclusões e alterações acidentais.

✗ A) Incorreta. A camada Archive guarda blobs existentes, sem recuperar excluídos.
✗ C) Incorreta. SAS delega acesso.
✗ D) Incorreta. O namespace organiza diretórios.

> **Dica de prova:** Recuperar blob apagado: exclusão reversível.

D7. C) Conjunto de consultas KQL → revise o capítulo 3

Por quê: O conjunto de consultas KQL (KQL queryset) é o espaço para escrever, salvar e compartilhar consultas sobre eventhouses e bancos KQL.

✗ A) Incorreta. O Dataflow Gen2 transforma dados com Power Query.
✗ B) Incorreta. Pipelines movem dados.
✗ D) Incorreta. Relatórios paginados são para impressão.

> **Dica de prova:** Salvar e compartilhar consultas KQL: conjunto de consultas.

D8. B) Delta Lake → revise o capítulo 3

Por quê: O Delta Lake é a base das tabelas de lakehouse no Fabric e no Databricks, com transações ACID, histórico de versões e aplicação de esquema.

✗ A) Incorreta. Avro é formato de serialização orientado a linhas.
✗ C) Incorreta. CSV é texto simples, sem transações.
✗ D) Incorreta. XML é texto orientado a documentos.

> **Dica de prova:** Parquet com ACID e versões: Delta Lake.

D9. A) Direct Lake → revise o capítulo 3

Por quê: O Direct Lake combina desempenho próximo da importação com dados atualizados, lendo os arquivos Delta do OneLake diretamente.

✗ B) Incorreta. A conexão ao vivo usa um modelo publicado.
✗ C) Incorreta. O DirectQuery envia consultas à fonte.
✗ D) Incorreta. A importação copia os dados para o modelo.

> **Dica de prova:** Power BI lendo Delta do OneLake: Direct Lake.

D10. B) ORDER BY Valor DESC → revise o capítulo 4

Por quê: ORDER BY define a ordem do resultado, ASC (padrão) ou DESC. WHERE filtra linhas e HAVING filtra grupos.

✗ A) Incorreta. WHERE filtra linhas.
✗ C) Incorreta. GROUP BY agrupa, não ordena.
✗ D) Incorreta. HAVING filtra grupos após a agregação.

> **Dica de prova:** Ordenar: ORDER BY (ASC ou DESC).

D11. C) ALTER TABLE Clientes ADD Email VARCHAR(100) → revise o capítulo 4

Por quê: Mudanças de estrutura usam DDL, como ALTER TABLE. Mudanças de conteúdo usam DML, como INSERT e UPDATE.

✗ A) Incorreta. INSERT inclui linhas.
✗ B) Incorreta. CREATE TABLE falharia, pois a tabela já existe.
✗ D) Incorreta. UPDATE altera valores em colunas que já existem.

> **Dica de prova:** Nova coluna em tabela existente: ALTER TABLE.

D12. C) Banco de Dados do Azure para PostgreSQL → revise o capítulo 4

Por quê: O Azure oferece serviços gerenciados para bancos de código aberto, como o Banco de Dados do Azure para PostgreSQL e para MySQL, com backup, escala e alta disponibilidade.

✗ A) Incorreta. A API Gremlin é para grafos.
✗ B) Incorreta. A Instância Gerenciada roda SQL Server, não PostgreSQL.
✗ D) Incorreta. O Table Storage é chave-valor, não relacional.

> **Dica de prova:** PostgreSQL gerenciado: Banco de Dados do Azure para PostgreSQL.

Agora, treine no formato real

Faça as 6 provas completas do curso DP-900: 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.