

Guia de preparação DP-800

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

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

ANTES DE COMEÇAR

Como usar este guia

Este guia foi feito para ser estudado junto com as 6 provas completas do curso DP-800: 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 120 minutos, no formato do exame. Referência de aprovação: 70%.

Domínio (capítulo)	Peso	Neste guia
1. Implementar recursos de IA em soluções de banco de dados	25-30%	18 conceitos · 14 pegadinhas
2. Projetar e desenvolver soluções de banco de dados	35-40%	18 conceitos · 14 pegadinhas
3. Proteger, otimizar e implantar soluções de banco de dados	35-40%	18 conceitos · 14 pegadinhas

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

Estude um pouco todo dia, de graça

Instagram @strecht.simulados

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

Seguir no Instagram →

YouTube Strecht Simulados

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

Inscrever-se no canal →

DIAGNÓSTICO

Você já sabe isso?

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

D1 · Implementar recursos de IA em soluções de banco de dados

O DBA da Rede de Lojas Moda Viva executou uma consulta em sys.vector_indexes e viu que JSON_VALUE(build_parameters, '$.Version') de um índice vetorial antigo retorna um valor menor que 3. A documentação recomenda migrar para a versão mais recente. Como fazer a migração?

A. Executar ALTER INDEX ... REBUILD com a opção VERSION = 3 no índice
B. Atualizar o nível de compatibilidade do banco, que converte o índice
C. Executar ALTER VECTOR INDEX ... UPGRADE durante a janela de manutenção
D. Remover o índice com DROP INDEX e recriá-lo com CREATE VECTOR INDEX

D2 · Implementar recursos de IA em soluções de banco de dados

Antes de carregar milhões de vetores, o engenheiro de dados da Seguradora Rural Campo Seguro quer validar, em uma consulta, se os vetores de uma tabela de staging têm o número esperado de dimensões e qual é o tipo base de cada um. Qual função usar?

A. VECTORPROPERTY, com 'Dimensions' e 'BaseType'
B. VECTOR_DISTANCE, comparando cada vetor com um vetor de zeros
C. VECTOR_NORMALIZE, que retorna o vetor com o tamanho informado
D. VECTOR_NORM, com os tipos de norma 'norm1' e 'norm2' do vetor

D3 · Implementar recursos de IA em soluções de banco de dados

Na base de laudos da Clínica Ortopédica Movimento, a equipe quer encontrar documentos com qualquer forma do verbo fraturar (fraturou, fraturado, fratura) e que mencionem joelho próximo de ligamento. A coluna Laudo tem índice de texto completo com o idioma português. Qual condição atende?

A. FREETEXT(Laudo, 'fraturar joelho próximo de ligamento', LANGUAGE 1046)
B. Laudo LIKE '%fratur%' AND Laudo LIKE '%joelho%' AND Laudo LIKE '%ligamento%'
C. CONTAINS(Laudo, '"fraturar" AND "joelho" AND "ligamento"', LANGUAGE 1046)
D. CONTAINS(Laudo, 'FORMSOF(INFLECTIONAL, fraturar) AND NEAR((joelho, ligamento))')

D4 · Projetar e desenvolver soluções de banco de dados

A coluna Perfil (json) da Agência de Intercâmbio Mundo Afora tem documentos como {"idiomas": ["inglês", "espanhol", "italiano"]}. O analista quer filtrar os alunos que tenham algum idioma começando com ital, usando semântica de LIKE na comparação. Qual expressão atende?

A. JSON_PATH_EXISTS(Perfil, '$.idiomas.ital') = 1
B. JSON_VALUE(Perfil, '$.idiomas') LIKE 'ital%'
C. JSON_CONTAINS(Perfil, 'ital%', '$.idiomas[*]') = 1
D. JSON_CONTAINS(Perfil, 'ital%', '$.idiomas[*]', 1) = 1

D5 · Projetar e desenvolver soluções de banco de dados

Um procedimento da Escola de Idiomas Fala Fácil monta, para a API, a lista de telefones de cada aluno com JSON_ARRAY(TelefoneFixo, Celular, Recado). Quando o aluno não tem telefone de recado (NULL), o front-end espera receber null na terceira posição do array, mas recebe um array com só dois elementos. Qual ajuste resolve?

A. Envolver cada argumento com ISNULL(coluna, '') antes de montar o array
B. Acrescentar ABSENT ON NULL ao final da lista de argumentos do JSON_ARRAY
C. Acrescentar NULL ON NULL ao final da lista de argumentos do JSON_ARRAY
D. Trocar JSON_ARRAY por JSON_ARRAYAGG agrupando pelos três telefones

D6 · Projetar e desenvolver soluções de banco de dados

A equipe da Agropecuária Boi Gordo quer que todos os desenvolvedores do repositório usem, no GitHub Copilot do Visual Studio Code, o mesmo servidor MCP local, iniciado por linha de comando via npx. Onde e com qual estrutura a configuração deve ser gravada para valer no workspace?

A. No arquivo .github/copilot-instructions.md, descrevendo em texto o comando npx que inicia o servidor
B. No arquivo dab-config.json, na seção runtime.mcp, com o caminho do executável do servidor local
C. No arquivo .vscode/settings.json, com a chave mcpServers e a URL pública do repositório do servidor
D. No arquivo .vscode/mcp.json, com a chave raiz servers e o servidor do tipo stdio com command e args

D7 · Proteger, otimizar e implantar soluções de banco de dados

A Agência de Turismo Destino Certo hospeda o Data API builder no Azure Container Apps e quer enviar ao Application Insights as requisições, exceções e dependências de banco geradas pela API, sem gravar a cadeia de conexão do Application Insights no arquivo versionado. Qual configuração atende?

A. data-source.options.set-session-context true para enviar as dependências de banco ao Application Insights
B. runtime.cache com enabled true e ttl-seconds alto para registrar as requisições no Application Insights
C. runtime.telemetry.application-insights com enabled true e connection-string "@env('APPINSIGHTS_CONN')"
D. runtime.host.mode development para que o DAB grave a telemetria completa em arquivos de log locais

D8 · Proteger, otimizar e implantar soluções de banco de dados

Para acelerar um painel, um desenvolvedor da Concessionária de Rodovias Via Livre colocou WITH (NOLOCK) em todas as tabelas da consulta de faturamento. O financeiro passou a ver valores que depois desaparecem do relatório. O que explica isso?

A. NOLOCK bloqueia as transações de escrita até o fim da consulta, que então lê valores já confirmados
B. NOLOCK desliga o cache de planos e cada execução recalcula o faturamento a partir de dados de backup
C. NOLOCK força a leitura da versão confirmada mais antiga da linha, guardada no repositório de versões do banco
D. NOLOCK equivale a READ UNCOMMITTED e permite ler alterações ainda não confirmadas, que podem ser revertidas

D9 · Proteger, otimizar e implantar soluções de banco de dados

O arquiteto da Companhia de Gás Chama Azul quer que o build do projeto SQL execute análise de código estática e que uma regra específica de desempenho, SR0004, faça o build falhar em vez de só gerar aviso. Quais propriedades do arquivo .sqlproj ele deve configurar?

A. TreatWarningsAsErrors = False e DSP com o provedor SqlAzureV12 do Azure SQL Database
B. RunSqlCodeAnalysis = True e SqlCodeAnalysisRules com +!Microsoft.Rules.Data.SR0004
C. BlockOnPossibleDataLoss = True e DropObjectsNotInSource = True no perfil de publicação
D. RunSqlCodeAnalysis = True e SqlCodeAnalysisRules com -Microsoft.Rules.Data.SR0004

01

CAPÍTULO 1 · 25-30% DA PROVA

Implementar recursos de IA em soluções de banco de dados

Aqui eu preciso dominar o ciclo completo de IA dentro do banco: registrar modelos externos, gerar e manter embeddings, guardar vetores, buscar por texto completo, vetor ou híbrido e montar um RAG chamando o modelo com sp_invoke_external_rest_endpoint. A prova cobra sintaxe exata (VECTOR_SEARCH com TOP WITH APPROXIMATE e ORDER BY distance, credenciais com escopo de banco, caminho $.result na resposta) e decisões de arquitetura (kNN ou ANN, gatilho ou Change Tracking ou CDC, texto completo ou vetorial). Onde mais erro é no sentido das métricas, porque em distância menor é mais parecido, e nas restrições de índice vetorial. Também caem limites numéricos do tipo vector e do sp_invoke_external_rest_endpoint. Pensar sempre em menos tokens, menos segredos expostos e menos trabalho dentro da transação resolve boa parte dos cenários.



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

Tópicos-chave

Tipo vector

Guarda cada elemento como float32 por padrão (4 bytes, até 1.998 dimensões) ou float16 (meia precisão, até 3.996 dimensões). As dimensões da coluna têm de coincidir com as do modelo; não existe VECTOR(MAX). É exibido como array JSON e converte de e para char, nchar, varchar, nvarchar e json nos dois sentidos, desde que o array tenha só números e o número de elementos coincida com as dimensões.

VECTOR_DISTANCE

Calcula distância exata (kNN) para todas as linhas qualificadas e nunca usa índice vetorial. Métricas: cosine (0 a 2), euclidean (0 a infinito) e dot (produto escalar negativo, de menos a mais infinito); em todas, menor é mais parecido. Pode ser usada no WHERE com um limiar definido por testes, além do padrão TOP N ordenado.

VECTOR_NORM, VECTOR_NORMALIZE e VECTORPROPERTY

VECTOR_NORM devolve a magnitude (norm1, norm2 ou norminf). VECTOR_NORMALIZE devolve o vetor escalado para comprimento 1 na norma escolhida; modelos do Azure OpenAI já entregam vetores normalizados. VECTORPROPERTY aceita 'Dimensions' e 'BaseType' e ajuda a diagnosticar vetores de modelos diferentes.

Índice vetorial (CREATE VECTOR INDEX)

CREATE VECTOR INDEX nome ON tabela(coluna) WITH (METRIC = 'cosine' | 'dot' | 'euclidean', TYPE = 'DiskANN'). Só em tabela base, com pelo menos 100 linhas de vetor não nulo; não pode ser particionado, não replica para assinantes e impede TRUNCATE TABLE. A versão aparece em build_parameters de sys.vector_indexes, e índice antigo se migra com DROP e CREATE.

VECTOR_SEARCH

VECTOR_SEARCH(TABLE = tabela_base AS alias, COLUMN, SIMILAR_TO = variável ou coluna vector, METRIC) devolve as colunas da tabela mais distance. Com SELECT TOP (N) WITH APPROXIMATE e ORDER BY distance ASC usa o índice (ANN); sem APPROXIMATE faz kNN exato. Só usa o índice se métrica e coluna coincidirem; caso contrário faz kNN e emite aviso. Não aceita tabela temporária nem pode ficar em view: encapsule em procedimento.

kNN, ANN e recall

kNN calcula todas as distâncias e garante os vizinhos corretos; é recomendado até cerca de 50 mil vetores após os filtros. ANN usa índice e troca um pouco de exatidão por velocidade. Recall é a fração dos vizinhos exatos que o ANN também retorna; perto de 1 significa resultado quase igual ao kNN.

Modelos externos (CREATE EXTERNAL MODEL)

Objeto de banco com LOCATION (só HTTPS com TLS), API_FORMAT (Azure OpenAI, OpenAI, Ollama ou ONNX Runtime, este com LOCAL_RUNTIME_PATH no SQL Server 2025), MODEL_TYPE = EMBEDDINGS, MODEL, CREDENTIAL e PARAMETERS. Criar não chama o modelo; ALTER EXTERNAL MODEL ... SET troca a definição para todos; DROP não remove a credencial. Metadados em sys.external_models; uso exige GRANT EXECUTE ON EXTERNAL MODEL::nome.

AI_GENERATE_EMBEDDINGS

AI_GENERATE_EMBEDDINGS(origem USE MODEL nome [PARAMETERS json]) gera o vetor usando um modelo externo; PARAMETERS da chamada sobrepõe o do modelo, como dimensions. Novas tentativas ficam em PARAMETERS = '{"sql_rest_options": {"retry_count":n}}' (0 a 10). No SQL Server 2025 é preciso sp_configure 'external rest endpoint enabled'; no Azure SQL Database e no Fabric já vem ligado, e o diagnóstico usa o evento ai_generate_embeddings_summary.

AI_GENERATE_CHUNKS e estratégia de chunking

AI_GENERATE_CHUNKS(SOURCE, CHUNK_TYPE = FIXED, CHUNK_SIZE, OVERLAP, ENABLE_CHUNK_SET_ID) divide por caracteres e devolve chunk, chunk_order, chunk_offset, chunk_length e, opcionalmente, chunk_set_id; exige compatibilidade 170. OVERLAP (0 a 50%) repete parte do trecho anterior. Trechos grandes demais misturam ideias, pequenos demais perdem contexto: tamanho intermediário com sobreposição.

Projeto de armazenamento de embeddings

Padrão recomendado: tabela dedicada de chunks com chave para o documento, ordem do chunk, texto e coluna vector. Embuta só colunas com significado (títulos, descrições, texto livre); identificadores, preços e datas viram filtros relacionais. Pergunta e documentos precisam do mesmo modelo, e trocar o modelo exige regenerar todos os vetores.

Manutenção de embeddings

Gatilho de tabela atualiza logo após a mudança, mas pesa na escrita; Change Tracking indica quais linhas mudaram para lote; CDC traz valores antes e depois para alto volume assíncrono. Azure Functions com gatilho SQL geram fora do mecanismo, Azure Logic Apps orquestra com baixo código, e o Microsoft Foundry centraliza seleção, avaliação e hospedagem do modelo enquanto outro processo detecta mudanças.

Busca de texto completo

Exige catálogo de texto completo e KEY INDEX único, de coluna única e não anulável. CONTAINS dá controle preciso (frases entre aspas, prefixo com asterisco, FORMSOF INFLECTIONAL ou THESAURUS, NEAR); FREETEXT busca pelo significado com formas flexionadas. CONTAINSTABLE e FREETEXTTABLE devolvem KEY e RANK. CHANGE_TRACKING AUTO propaga mudanças, MANUAL exige START UPDATE POPULATION; stoplists personalizadas ignoram palavras.

Busca híbrida e RRF

Executa texto completo e vetorial e combina as posições com Reciprocal Rank Fusion: cada lista contribui com 1/(k + rank), com k tipicamente 60, e as parcelas se somam. Use FULL OUTER JOIN e COALESCE(..., 0) para não perder quem está em uma só lista, aplique filtros antes das buscas e ajuste tamanho das listas, k e pesos (como 2.0/(k + rank)).

Métricas de qualidade de busca

Precisão é a fração dos retornados que é relevante; recall é a fração dos relevantes encontrada; MRR mede quão alto aparece o primeiro resultado relevante. Avalie com um conjunto de consultas de teste com relevância conhecida, e meça desempenho separadamente (tempo, leituras, CPU).

sp_invoke_external_rest_endpoint

Parâmetros: @url, @payload, @headers (JSON plano, nvarchar(4000)), @method (padrão POST), @timeout (1 a 230 s, padrão 30), @credential, @response OUTPUT e @retry_count (0 a 10). Retorna 0 para 2xx e o código HTTP nos demais; se não consegue chamar, lança exceção. Exige EXECUTE ANY EXTERNAL ENDPOINT e, no Azure SQL Database, só chama domínios permitidos (outros serviços via Azure API Management).

DATABASE SCOPED CREDENTIAL para endpoints

IDENTITY define como o segredo vai: HTTPEndpointHeaders (cabeçalhos, como api-key), HTTPEndpointQueryString (query string), Managed Identity (SECRET com resourceid) ou Shared Access Signature. O nome da credencial é uma URL em domínio permitido, sem query string, com o mesmo protocolo e FQDN da chamada e caminho igual ou mais genérico.

Fluxo RAG no banco

O modelo roda fora (por exemplo Azure OpenAI); o banco guarda dados e vetores, recupera o contexto (busca vetorial ou híbrida, junções, TOP), formata com FOR JSON PATH só com as colunas úteis, monta o payload messages com papéis system e user, chama o endpoint e extrai a resposta. A resposta do chat fica em $.result.choices[0].message.content; para texto acima de 4.000 caracteres use OPENJSON WITH nvarchar(max).

Parâmetros do modelo, grounding e saídas estruturadas

temperature (0 a 2) baixa para respostas factuais e max_tokens limita o tamanho da resposta. A mensagem de sistema delimita o escopo (responder só com o contexto e admitir quando faltar informação). Saídas estruturadas com JSON Schema garantem campos e tipos, extraídos com JSON_VALUE ou OPENJSON ... WITH. Tokens limitam o contexto e definem o custo.

Macetes para memorizar

💡 Métricas de distância

Distância é golfe: quanto menor, melhor. Cosseno vai de 0 (gêmeos) a 2 (opostos).

💡 Busca aproximada

TOP + WITH APPROXIMATE + ORDER BY distance ASC, e nada mais no ORDER BY.

💡 RRF

Cada lista paga 1 sobre (60 + posição); quem está nas duas recebe dos dois.

💡 Resposta do sp_invoke_external_rest_endpoint

response é o carimbo do correio; result é a carta. A resposta do modelo mora em result.choices[0].message.content.

💡 Credenciais

Headers põe a chave no cabeçalho, QueryString põe na URL, Managed Identity não tem chave, só resourceid.

💡 Tamanho do vetor

float32 = 4 bytes até 1.998; float16 = metade dos bytes e o dobro de dimensões (3.996).

Modelos de embeddings

Mesmo modelo, mesmo espaço: pergunta e documentos sempre com o mesmo modelo.

Índice vetorial

100 linhas, tabela base, sem partição, sem truncar, sem replicar; versão nova só com DROP e CREATE.

Tabela de memorização: não confunda

Quando é...	...e quando é
kNN (VECTOR_DISTANCE ou VECTOR_SEARCH sem APPROXIMATE) Poucos vetores após os filtros (até cerca de 50 mil) ou quando se precisa do resultado exato como referência.	**ANN (TOP N WITH APPROXIMATE com índice)** Grandes volumes sem filtro seletivo, aceitando recall menor que 1 em troca de velocidade.
Busca de texto completo Nomes, códigos e termos exatos digitados pelo usuário.	**Busca vetorial** Usuário descreve a necessidade com palavras diferentes das do catálogo; para intenção imprevisível, combine as duas (híbrida).
CONTAINS Controle preciso: frase exata, prefixo, FORMSOF, proximidade com NEAR e operadores AND/OR.	**FREETEXT** Texto livre buscado pelo significado, com formas flexionadas geradas automaticamente.
Gatilho de tabela para embeddings A busca precisa refletir a mudança logo após a gravação, aceitando custo na escrita.	**Change Tracking ou CDC para embeddings** Processamento em lote ou assíncrono: CT quando basta saber que mudou, CDC quando precisa de valores antes e depois em alto volume.
VECTOR_NORM Saber a magnitude do vetor (um número).	**VECTOR_NORMALIZE** Obter o próprio vetor escalado para comprimento 1.
JSON_VALUE na resposta Extrair escalar curto, como o content de até 4.000 caracteres ou o status.	**OPENJSON WITH nvarchar(max)** Extrair texto longo do modelo sem corte, ou converter os campos de uma saída estruturada em colunas tipadas.
RAG Dados que mudam com frequência, rastreabilidade das fontes e dados sob controle da organização, sem retreinar.	**Fine-tuning** Gravar conhecimento nos pesos do modelo, processo que o banco não faz.

Números que caem

Item	Valor
Tipo vector float32 (padrão)	**4 bytes por elemento, até 1.998 dimensões**
Tipo vector float16	**até 3.996 dimensões**

Faixa de VECTOR_DISTANCE com cosine	**0 (idênticos) a 2 (opostos)**
Mínimo para criar índice vetorial	**100 linhas com vetor não nulo**
kNN recomendado	**até cerca de 50 mil vetores após os filtros**
Constante k típica do RRF	**60**
@timeout do sp_invoke_external_rest_endpoint	**1 a 230 segundos, padrão 30**
@retry_count e retry_count em sql_rest_options	**0 a 10 (o @retry_count tem padrão 0)**
Limites do sp_invoke_external_rest_endpoint	**payload 100 MB, URL 8 KB, cabeçalhos 8 KB; conexões externas simultâneas: 10% dos workers, máximo 150**
Erros de limite de conexões externas	**10928 ou 10936**
OVERLAP do AI_GENERATE_CHUNKS	**0 a 50%**
temperature	**0 a 2**
Retorno do JSON_VALUE sem RETURNING	**nvarchar(4000)**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Ordenar a distância de cosseno de forma decrescente porque 'maior é mais parecido'.
Como não cair: Em VECTOR_DISTANCE e VECTOR_SEARCH, menor é mais parecido em todas as métricas; a ordem é crescente.

⚠ **Alerta de prova:** Esperar que VECTOR_DISTANCE use o índice vetorial recém-criado.
Como não cair: VECTOR_DISTANCE é sempre exato. Só VECTOR_SEARCH com TOP WITH APPROXIMATE usa o índice.

⚠ **Alerta de prova:** Colocar outras colunas ou DESC no ORDER BY da busca aproximada, ou omitir o ORDER BY.
Como não cair: Com WITH APPROXIMATE, ORDER BY é obrigatório, só aceita distance e só crescente. Outras ordenações vão para consulta externa.

⚠ **Alerta de prova:** Consultar com euclidean num índice criado com cosine e achar que o índice está corrompido.
Como não cair: Métrica diferente da do índice faz kNN completo com aviso. Use a mesma métrica e coluna do índice.

⚠ **Alerta de prova:** Somar o RANK do texto completo com a distância de cosseno, ou juntar as listas com INNER JOIN.
Como não cair: Escalas diferentes não se somam: use RRF com 1/(k + rank) e FULL OUTER JOIN com COALESCE.

⚠ **Alerta de prova:** Aumentar o k do RRF esperando recuperar itens perdidos.
Como não cair: k só suaviza o peso das posições. Itens perdidos voltam corrigindo a junção ou aplicando filtros antes das buscas.

⚠ **Alerta de prova:** Trocar o caminho para strict quando JSON_VALUE devolve NULL para uma resposta longa.
Como não cair: JSON_VALUE retorna até nvarchar(4000); strict só troca NULL por erro. Use OPENJSON WITH (coluna nvarchar(max) '$.result.choices[0].message.content').

⚠ **Alerta de prova:** Ler a resposta do modelo em $.response.choices ou direto na raiz de @response.
Como não cair: response guarda status e cabeçalhos HTTP; o corpo do modelo está em result: $.result.choices[0].message.content.

⚠ **Alerta de prova:** Interpretar retorno 429 do sp_invoke_external_rest_endpoint como sucesso ou como 'chamada não feita'.
Como não cair: 0 é só para 2xx; outro valor é o código HTTP. Quando nem chama, o procedimento lança exceção.

⚠ **Alerta de prova:** Definir MODEL_TYPE = CHAT ou API_FORMAT Azure Machine Learning/Hugging Face no CREATE EXTERNAL MODEL.
Como não cair: O tipo aceito é EMBEDDINGS e os formatos são Azure OpenAI, OpenAI, Ollama e ONNX Runtime.

⚠ **Alerta de prova:** Trocar o modelo de embeddings com ALTER EXTERNAL MODEL e continuar usando os vetores antigos.
Como não cair: Vetores de modelos diferentes ficam em espaços diferentes; regenere todos os vetores armazenados.

⚠ **Alerta de prova:** Achar que DROP EXTERNAL MODEL apaga a credencial ou falha por causa dela.
Como não cair: O DROP remove só o modelo; a credencial continua e deve ser excluída à parte se não for usada.

⚠ **Alerta de prova:** Guardar chave de API com IDENTITY = 'Managed Identity' ou escrevê-la em @headers em cada chamada.
Como não cair: Chave em cabeçalho é HTTPEndpointHeaders com SECRET em JSON; Managed Identity leva o resourceid, sem chave.

⚠ **Alerta de prova:** Aumentar max_tokens para o modelo 'ler' um contexto inchado, ou subir a temperatura para respostas melhores.
Como não cair: max_tokens limita a resposta, não a entrada. Reduza o contexto às colunas úteis e use temperatura baixa com grounding.

Como cai na prova

📝 **Como cai:** Consulta de busca vetorial com um erro de sintaxe ou de semântica para identificar.
Como resolver: Confira na ordem: tabela base no TABLE, SIMILAR_TO como variável ou coluna vector, métrica igual à do índice, TOP WITH APPROXIMATE, ORDER BY só distance crescente. Se a consulta usa VECTOR_DISTANCE, lembre que ela nunca usa índice.

📝 **Como cai:** Cenário de catálogo ou base de conhecimento pedindo o tipo de busca adequado.
Como resolver: Termos e códigos exatos levam a texto completo; descrições com palavras diferentes levam a vetorial; mistura imprevisível leva a híbrida com RRF. Depois confira se os filtros vêm antes das buscas.

📝 **Como cai:** Trecho T-SQL de RAG: montar payload, chamar o endpoint e extrair a resposta.
Como resolver: Payload com JSON_OBJECT('messages': JSON_ARRAY(...)) e papéis system e user; EXEC @ret = sp_invoke_external_rest_endpoint com @response OUTPUT; teste @ret = 0; extraia de $.result.choices[0].message.content (OPENJSON nvarchar(max) se puder passar de 4.000 caracteres).

📝 **Como cai:** Escolher o método de manutenção de embeddings para um requisito de latência e de carga.
Como resolver: Precisa refletir na hora: gatilho. Lote sem valores anteriores: Change Tracking. Alto volume assíncrono com antes e depois: CDC. Fora do mecanismo: Azure Functions com gatilho SQL. Baixo código agendado: Logic Apps. Nunca regenerar tudo periodicamente nem chamar o modelo dentro da transação quando o requisito é não pesar na escrita.

📝 **Como cai:** Configuração de segurança e permissões para IA (credencial, permissões, habilitação). **Como resolver:** Chamar endpoint: EXECUTE ANY EXTERNAL ENDPOINT. Usar modelo: EXECUTE ON EXTERNAL MODEL::nome. Criar ou alterar modelo: CREATE EXTERNAL MODEL ou ALTER ANY EXTERNAL MODEL. Restringir quem dispara o fluxo: procedimento com EXECUTE só para a função certa. SQL Server 2025: 'external rest endpoint enabled' via sp_configure.

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
Medir quanto a busca aproximada se aproxima da exata	**Recall comparando os vizinhos do ANN com os do kNN**
Poucos milhares de vetores depois de filtros seletivos	**kNN exato com VECTOR_DISTANCE tende a bastar**
Usuário descreve o que precisa em vez de nomear o item	**Busca vetorial semântica com embeddings**
Ora digitam códigos exatos, ora descrevem a necessidade	**Busca híbrida com RRF**
Ordenar resultados de texto completo por relevância	**Junção com CONTAINSTABLE por [KEY], ordenando por [RANK] DESC**
Busca vetorial dentro de uma view não compila	**Encapsular VECTOR_SEARCH em procedimento armazenado**
Vetores com mais de 1.998 dimensões	**Pedir menos dimensões ao modelo ou usar float16 (até 3.996)**
Mudar o modelo usado por vários procedimentos sem reescrevê-los	**ALTER EXTERNAL MODEL ... SET (MODEL = ...)**
Erros 10928 ou 10936 em chamadas externas simultâneas	**Limite de conexões externas proporcional aos workers**
Serviço fora da lista de domínios permitidos do Azure SQL Database	**Expor pelo Azure API Management e chamar o endpoint do APIM**
Chamar o modelo sem guardar segredo no banco	**DATABASE SCOPED CREDENTIAL com IDENTITY = 'Managed Identity' e resourceid**
Respostas inventadas pelo assistente	**Grounding na mensagem de sistema e temperatura baixa**
Resposta do modelo precisa virar colunas tipadas de forma confiável	**Saídas estruturadas com JSON Schema e OPENJSON ... WITH**
Contexto perdido nas fronteiras dos trechos	**OVERLAP no AI_GENERATE_CHUNKS**

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

- Em distância vetorial, menor é mais parecido, sempre em ordem crescente.
- VECTOR_DISTANCE é exato; só VECTOR_SEARCH com TOP WITH APPROXIMATE usa o índice, e com a mesma métrica.
- sp_invoke_external_rest_endpoint: 0 é sucesso 2xx, a resposta vem em @response OUTPUT e o conteúdo está em $.result.
- Trocar o modelo de embeddings exige regenerar todos os vetores.
- No RAG, envie ao modelo só as colunas úteis em JSON, com grounding no system e temperatura baixa.

Essenciais para a prova (checklist)

☐ Sei declarar colunas vector, os limites de dimensões por tipo base e as conversões permitidas.

☐ Sei a faixa e o sentido de cosine, euclidean e dot no VECTOR_DISTANCE.

☐ Sei criar índice vetorial e listar suas limitações (100 linhas, tabela base, sem partição, sem TRUNCATE).

☐ Sei escrever VECTOR_SEARCH exato e aproximado e explicar quando o índice não é usado.

☐ Sei criar, alterar, remover e conceder EXECUTE em modelos externos e usar AI_GENERATE_EMBEDDINGS com PARAMETERS.

☐ Sei escolher entre gatilho, Change Tracking, CDC, Functions, Logic Apps e Foundry para manter embeddings.

☐ Sei montar índice de texto completo e usar CONTAINS, FREETEXT, CONTAINSTABLE e stoplists.

☐ Sei calcular RRF e combinar listas com FULL OUTER JOIN e COALESCE.

☐ Sei chamar sp_invoke_external_rest_endpoint com credencial adequada e extrair a resposta do caminho certo.

☐ Sei avaliar busca com precisão, recall e MRR.

Questões de revisão do capítulo

R1.1 · Implementar recursos de IA em soluções de banco de dados

O assistente de políticas internas da Indústria Química Segura Mais dá respostas diferentes para a mesma pergunta e, às vezes, respostas longas demais. O payload enviado ao modelo de chat usa temperature 1.2 e não limita o tamanho da resposta. Que ajuste no payload é o mais indicado para respostas factuais e concisas?

A. Aumentar temperature para 2 e remover a mensagem de sistema com as regras de grounding
B. Manter temperature e trocar o papel system por assistant na mensagem com as regras
C. Reduzir temperature para algo como 0.3 e definir max_tokens adequado ao tamanho desejado
D. Reduzir max_tokens para 1 e repetir a chamada até a resposta ficar completa no banco

R1.2 · Implementar recursos de IA em soluções de banco de dados

A Plataforma de Cursos Livres Educa Mais tem 2 milhões de embeddings na coluna Vetor e quer acelerar a busca aproximada por similaridade de cosseno no Azure SQL Database. Qual instrução cria o índice adequado?

A. CREATE VECTOR INDEX IX_Vetor ON dbo.Aulas(Vetor) WITH (METRIC = 'cosine', TYPE = 'DiskANN');
B. CREATE NONCLUSTERED INDEX IX_Vetor ON dbo.Aulas(Vetor) WITH (DATA_COMPRESSION = PAGE);
C. CREATE COLUMNSTORE INDEX IX_Vetor ON dbo.Aulas(Vetor) WITH (MAXDOP = 4, ONLINE = ON, DATA_COMPRESSION = COLUMNSTORE);
D. CREATE JSON INDEX IX_Vetor ON dbo.Aulas(Vetor) WITH (OPTIMIZE_FOR_ARRAY_SEARCH = ON);

R1.3 · Implementar recursos de IA em soluções de banco de dados

A Indústria Farmacêutica Cura Bem vai registrar no Azure SQL Database um modelo de embeddings implantado no Azure OpenAI com o nome emb-bulas, usando a credencial já criada. Qual instrução registra o modelo para uso com AI_GENERATE_EMBEDDINGS?

A. CREATE EXTERNAL MODEL ModeloBulas WITH (LOCATION = '...emb-bulas/embeddings?...', API_FORMAT = 'Azure OpenAI', MODEL_TYPE = EMBEDDINGS, MODEL = '...', CREDENTIAL = [...])
B. CREATE EXTERNAL MODEL ModeloBulas WITH (LOCATION = '...deployments/emb-bulas/chat/completions', API_FORMAT = 'Azure OpenAI', MODEL_TYPE = CHAT)
C. CREATE EXTERNAL DATA SOURCE ModeloBulas WITH (LOCATION = '...deployments/emb-bulas/embeddings', TYPE = BLOB_STORAGE, CREDENTIAL = [...]) apontando para o recurso
D. EXEC sp_invoke_external_rest_endpoint @url = '...deployments/emb-bulas/embeddings', @method = 'POST' para registrar o modelo no banco

R1.4 · Implementar recursos de IA em soluções de banco de dados

A Editora Saber Total usa o modelo externo ModeloLivros, com text-embedding-ada-002, em dezenas de procedimentos com AI_GENERATE_EMBEDDINGS. A equipe vai migrar para text-embedding-3-large no mesmo recurso e quer evitar alterar os procedimentos. Qual ação atende?

A. DROP e CREATE de cada procedimento trocando o nome do modelo em todas as chamadas
B. CREATE EXTERNAL MODEL ModeloLivros2 e um sinônimo com o nome antigo para o novo
C. ALTER EXTERNAL MODEL ModeloLivros SET (MODEL = 'text-embedding-3-large')
D. sp_rename 'ModeloLivros', 'text-embedding-3-large' para atualizar o modelo usado

Respostas comentadas

R1.1. C) Reduzir temperature para algo como 0.3 e definir max_tokens adequado ao tamanho desejado

Por quê: No payload de chat, temperature (0 a 2) controla a criatividade, com valores baixos para respostas factuais em RAG, e max_tokens limita o comprimento da resposta. Combine com instruções de grounding na mensagem de sistema.

✗ A) Incorreta. Temperatura alta aumenta a variabilidade, e retirar o grounding piora a fidelidade aos dados.

✗ B) Incorreta. As regras de comportamento pertencem à mensagem de sistema; mudar o papel não reduz a variabilidade.

✗ D) Incorreta. Um token por resposta inviabiliza a geração.

> **Dica de prova:** Respostas factuais e curtas: temperature baixa e max_tokens definido.

R1.2. A) CREATE VECTOR INDEX IX_Vetor ON dbo.Aulas(Vetor) WITH (METRIC = 'cosine', TYPE = 'DiskANN');

Por quê: CREATE VECTOR INDEX nome ON tabela(coluna_vector) WITH (METRIC = 'cosine' | 'dot' | 'euclidean', TYPE = 'DiskANN') cria o índice usado pelo VECTOR_SEARCH quando a métrica coincide. A tabela precisa ser base.

✗ B) Incorreta. O tipo vector não pode ser chave de índice rowstore comum.

✗ C) Incorreta. Columnstore não implementa busca por vizinhos mais próximos.

✗ D) Incorreta. Índices JSON são para colunas json, não para busca vetorial.

> **Dica de prova:** Índice para busca vetorial aproximada: CREATE VECTOR INDEX com METRIC e TYPE = 'DiskANN'.

R1.3. A) CREATE EXTERNAL MODEL ModeloBulas WITH (LOCATION = '...emb-bulas/embeddings?...', API_FORMAT = 'Azure OpenAI', MODEL_TYPE = EMBEDDINGS, MODEL = '...', CREDENTIAL = [...])

Por quê: Um modelo externo é um objeto de banco com a localização do endpoint, o formato da API (Azure OpenAI, OpenAI, Ollama ou ONNX Runtime), o tipo EMBEDDINGS, o nome do modelo, a credencial e parâmetros opcionais. Criá-lo não chama o modelo.

✗ B) Incorreta. O tipo de modelo aceito é EMBEDDINGS, e o caminho de chat não é o de embeddings.

✗ C) Incorreta. Fontes de dados externas servem para virtualização de dados, não para registrar modelos de IA.

✗ D) Incorreta. sp_invoke_external_rest_endpoint faz uma chamada HTTP; não registra um modelo reutilizável.

> **Dica de prova:** Registrar modelo para AI_GENERATE_EMBEDDINGS: CREATE EXTERNAL MODEL com MODEL_TYPE = EMBEDDINGS.

R1.4. C) ALTER EXTERNAL MODEL ModeloLivros SET (MODEL = 'text-embedding-3-large')

Por quê: ALTER EXTERNAL MODEL ... SET (...) muda a definição centralmente; consultas que usam USE MODEL continuam iguais. Lembre que trocar o modelo de embeddings exige regenerar os vetores armazenados, pois vetores de modelos diferentes não são comparáveis.

✗ A) Incorreta. Reescrever os procedimentos é exatamente o que a equipe quer evitar.

✗ B) Incorreta. Sinônimos não são o mecanismo documentado para modelos externos.

✗ D) Incorreta. Renomear o objeto não altera o modelo usado no endpoint.

> **Dica de prova:** Trocar o modelo sem mexer nas consultas: ALTER EXTERNAL MODEL ... SET (MODEL = ...).

02

CAPÍTULO 2 · 35-40% DA PROVA

Projetar e desenvolver soluções de banco de dados

Aqui eu preciso escrever e reconhecer T-SQL correto: tratamento de erros, funções, views, gatilhos, procedimentos, restrições e escolha de tipos e índices, além das tabelas especializadas (temporais, ledger, grafo, em memória, particionadas) e das funções novas de JSON, regex e correspondência aproximada. A prova mostra quatro trechos de código quase iguais e o detalhe que decide é um padrão de NULL, um quadro de janela, um flag de regex ou o lado da seta no MATCH. Também cai bastante a parte de ferramentas de IA: GitHub Copilot no SSMS e na extensão MSSQL, Copilot in Fabric, servidores MCP e arquivos de instruções. Onde mais erro é nos padrões invertidos (JSON_OBJECT x JSON_ARRAY com NULL) e nas faixas das funções de similaridade. Ler cada alternativa contra o requisito exato evita a maioria das quedas.

Neste capítulo

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

Tópicos-chave

TRY...CATCH, XACT_STATE e funções ERROR_*

Padrão: transação e COMMIT dentro do TRY; no CATCH, IF @@TRANCOUNT > 0 (ou XACT_STATE() <> 0) ROLLBACK e depois THROW. XACT_STATE() retorna 1 (confirmável), 0 (sem transação) ou -1 (só aceita ROLLBACK). ERROR_NUMBER, ERROR_MESSAGE, ERROR_LINE, ERROR_PROCEDURE, ERROR_SEVERITY e ERROR_STATE só funcionam no CATCH. Erros de compilação ou de resolução de nome no mesmo nível do TRY escapam; em nível inferior (procedimento, sp_executesql) são capturados.

THROW, XACT_ABORT e NOCOUNT

THROW sem parâmetros relança o erro capturado; com parâmetros, usa número de 50.000 a 2.147.483.647, severidade sempre 16, não exige sys.messages, não aceita formatação printf e a instrução anterior precisa terminar com ponto e vírgula. SET XACT_ABORT ON faz qualquer erro de execução reverter a transação inteira. SET NOCOUNT ON só suprime as mensagens de contagem de linhas.

Funções escalares e com valor de tabela

TVF inline tem um único SELECT e o otimizador a trata como view parametrizada; a de várias instruções é caixa-preta com estimativas ruins. O inlining de UDF escalar (compatibilidade 150+) evita execução linha a linha; is_inlineable em sys.sql_modules mostra a elegibilidade, e funções com GETDATE() não são elegíveis. Funções não determinísticas não servem para índice em coluna computada nem em view indexada.

Views: SCHEMABINDING, CHECK OPTION e views indexadas

SCHEMABINDING vincula a view à estrutura das tabelas base e bloqueia ALTER que a afete, sem impedir DML. WITH CHECK OPTION garante que modificações pela view continuem visíveis nela. View indexada exige SCHEMABINDING, nomes de duas partes, expressões determinísticas, índice clusterizado único primeiro e COUNT_BIG(*) com GROUP BY; não aceita LEFT JOIN.

Gatilhos DML e DDL

Gatilhos DML disparam uma vez por instrução; inserted e deleted trazem todas as linhas afetadas, então a lógica deve ser baseada em conjunto. UPDATE(coluna) indica que a coluna estava no SET, não que o valor mudou; para isso compare inserted com deleted. INSTEAD OF permite modificar views de várias tabelas; gatilhos DDL ON DATABASE usam EVENTDATA() em XML.

Procedimentos: OUTPUT, RETURN, TVP e SQL dinâmico

RETURN devolve só inteiro (um decimal perde os centavos); valores vão por parâmetro OUTPUT. Para mandar várias linhas numa chamada, use tipo de tabela definido pelo usuário como parâmetro READONLY. SQL dinâmico seguro é sp_executesql com @params; QUOTENAME serve para identificadores, não para valores.

Restrições e SEQUENCE

CHECK segue lógica de três valores: NULL gera UNKNOWN e passa, então campo obrigatório pede NOT NULL. UNIQUE aceita um único NULL; para vários NULLs use índice único filtrado WHERE coluna IS NOT NULL. FK aceita NO ACTION, CASCADE, SET NULL e SET DEFAULT em DELETE e UPDATE (não existe ON INSERT). SEQUENCE é compartilhável, obtida antes do INSERT com NEXT VALUE FOR ou sp_sequence_get_range, gera lacunas (números obtidos e não usados, inclusive por ROLLBACK, não voltam) e a unicidade na tabela depende de restrições.

Tipos de dados

decimal/numeric guarda valores exatos para dinheiro; float e real são aproximados (real é float(24)). datetime2 é o tipo recomendado sem fuso, com faixa de 0001 a 9999 e precisão de 100 ns; datetimeoffset preserva o deslocamento em relação ao UTC. nvarchar armazena Unicode; varchar só guarda Unicode com colação UTF-8.

Índices rowstore, columnstore e em memória

INCLUDE coloca colunas no nível folha e cobre a consulta, eliminando Key Lookup. Índice filtrado indexa só um subconjunto bem definido. Columnstore clusterizado é o armazenamento principal de tabelas analíticas; o não clusterizado dá análise em tempo real sobre a tabela OLTP. Em tabelas otimizadas para memória, índice hash (BUCKET_COUNT) atende igualdade pontual e o não clusterizado atende faixas e ordenação; DURABILITY = SCHEMA_ONLY perde os dados no reinício.

Particionamento

Usa função de partição (limites, RANGE LEFT ou RIGHT), esquema de partição e coluna de partição. Índice único alinhado precisa conter a coluna de partição na chave. SWITCH move uma partição para tabela vazia de estrutura idêntica, TRUNCATE ... WITH (PARTITIONS) esvazia partições, SPLIT RANGE cria o próximo limite e MERGE RANGE remove um limite.

Tabelas temporais e ledger

Temporal com versão do sistema exige duas colunas datetime2 (ROW START e ROW END), PERIOD FOR SYSTEM_TIME e SYSTEM_VERSIONING = ON; HISTORY_RETENTION_PERIOD limpa o histórico. Ledger torna adulteração detectável com hashes e digests: APPEND_ONLY = ON aceita só INSERT, a atualizável guarda histórico. Verificação com sp_verify_database_ledger_from_digest_storage (digest automático) ou sp_verify_database_ledger (manual).

Grafos (nós, bordas, MATCH e SHORTEST_PATH)

Nós têm $node_id; bordas têm $edge_id, $from_id e $to_id, preenchidos com os $node_id dos nós. No MATCH a seta vai do $from_id para o $to_id, e cada ocorrência de uma borda no padrão precisa de alias próprio. SHORTEST_PATH usa + ou {1,n}, FOR PATH nas tabelas e funções WITHIN GROUP (GRAPH PATH); restrições de borda CONNECTION (Origem TO Destino) limitam os tipos de nós ligados.

Funções JSON

JSON_VALUE retorna escalar e JSON_QUERY retorna objeto ou array; em lax o erro vira NULL, em strict vira erro. OPENJSON com WITH tipa colunas, e fragmentos pedem nvarchar(max) AS JSON. JSON_OBJECT tem NULL ON NULL como padrão e JSON_ARRAY tem ABSENT ON NULL; JSON_ARRAYAGG agrega linhas com ORDER BY interno. JSON_CONTAINS(alvo, valor, caminho, search_mode) usa 0 para igualdade e 1 para semântica de LIKE.

Tipo json e índice JSON

O tipo json guarda o documento já analisado em binário e só aceita objeto ou array no nível superior. O método modify altera valores no lugar. CREATE JSON INDEX ... FOR (caminhos) exige chave primária clusterizada e não aceita caminhos sobrepostos; para JSON em texto, crie coluna computada com JSON_VALUE e indexe. CHECK com JSON_PATH_EXISTS exige propriedades obrigatórias.

Expressões regulares e correspondência aproximada

REGEXP_LIKE valida (exige compatibilidade 170), REGEXP_SUBSTR extrai uma ocorrência ou grupo, REGEXP_REPLACE usa barra invertida + número para grupos e & para a correspondência inteira, REGEXP_INSTR devolve posição, REGEXP_MATCHES e REGEXP_SPLIT_TO_TABLE devolvem linhas. Flags: i, m, s e c (padrão, diferencia maiúsculas). EDIT_DISTANCE conta edições (Damerau-Levenshtein) com máximo opcional; JARO_WINKLER_DISTANCE vai de 0 (idênticas) para cima; EDIT_DISTANCE_SIMILARITY e JARO_WINKLER_SIMILARITY vão de 0 a 100.

Funções de janela, CTE, APPLY e EXISTS

ROW_NUMBER não repete, RANK repete e pula, DENSE_RANK repete sem pular; top N por grupo é ROW_NUMBER com PARTITION BY numa CTE filtrada fora. Com ORDER BY no OVER sem quadro, o padrão é RANGE UNBOUNDED PRECEDING AND CURRENT ROW. CTE recursiva para em 100 níveis por padrão (MAXRECURSION de 0 a 32767). OUTER APPLY mantém linhas sem resultado da TVF; NOT EXISTS não sofre com NULL como NOT IN.

GitHub Copilot no SSMS

Requer SSMS 22 ou posterior, a carga de trabalho AI Assistance e conta do GitHub com Copilot. Ask mode responde sem MCP e usa o contexto da janela conectada; Agent mode usa o servidor MCP local sql-tools, pede aprovação (Allow once, for this session, always) e não herda o contexto do editor. Executa sob o login conectado; o CONSTITUTION.md do banco com agentExecuteAsUser faz EXECUTE AS um usuário de menor privilégio (exige IMPERSONATE).

Extensão MSSQL, instruções e MCP

No VS Code, @mssql, Agent mode e Schema Designer conhecem o esquema; sugestões inline não. Plan mode raciocina antes da DDL. Instruções: .github/copilot-instructions.md (repositório), .github/instructions/nome.instructions.md com applyTo e, no SSMS, %USERPROFILE%/copilot-instructions.md. Servidores MCP: .vscode/mcp.json com a chave servers no VS Code; no SSMS, painel Tools ou %USERPROFILE%/.mcp.json, com as ferramentas vindo desmarcadas. O SQL MCP Server do DAB reaproveita entidades e RBAC, sem DDL nem NL2SQL.

Macetes para memorizar

💡 NULL nas funções JSON

Objeto mostra o nulo, array esconde o nulo: JSON_OBJECT é NULL ON NULL, JSON_ARRAY é ABSENT ON NULL.

💡 Ranking

ROW_NUMBER: fila do banco (cada um sua senha). RANK: pódio com buraco. DENSE_RANK: pódio sem buraco.

💡 Similaridade x distância

Distância: 0 é gêmeo. Similaridade: 100 é gêmeo.

💡 Seta do MATCH

A seta aponta do from para o to: quem segue -> quem é seguido.

💡 Copilot no SSMS

Ask pergunta e lê a janela; Agent age com ferramentas e MCP, mas precisa do banco no prompt.

💡 THROW

THROW: 50 mil para cima, severidade 16 fixa, ponto e vírgula antes.

💡 Views indexadas

SCHEMABINDING, duas partes, COUNT_BIG, nada de GETDATE e nada de LEFT JOIN.

💡 Temporal FOR SYSTEM_TIME

AS OF é um instante; BETWEEN pega quem esteve ativo no período; CONTAINED IN só quem nasceu e morreu dentro.

Tabela de memorização: não confunda

Quando é...	...e quando é
ROW_NUMBER Exatamente N linhas por grupo, com desempate explícito no ORDER BY.	**RANK / DENSE_RANK** Empatados devem dividir a posição: RANK pula números depois do empate, DENSE_RANK não pula.
JSON_VALUE Extrair um escalar (texto, número, booleano), como $.cliente.nome.	**JSON_QUERY** Extrair um objeto ou array, como $.itens.
CROSS APPLY Só interessam as linhas da esquerda para as quais a função retorna algo.	**OUTER APPLY** Todas as linhas da esquerda devem aparecer, com NULL quando a função não retorna nada.
TVF inline Um único SELECT, com estimativas boas porque o otimizador expande como view parametrizada.	**TVF de várias instruções** Lógica que precisa montar o resultado em etapas, aceitando a caixa-preta para o otimizador.
IDENTITY Numeração própria de uma tabela, conhecida só depois do INSERT.	**SEQUENCE** Número compartilhado por várias tabelas, obtido antes do INSERT ou reservado em blocos.

Columnstore clusterizado Tabela fato analítica grande, com compressão e modo batch como armazenamento principal.	**Columnstore não clusterizado** Análise operacional em tempo real sobre tabela OLTP que continua em rowstore, sem ETL.
Ask mode (Copilot no SSMS) Perguntas e consultas de leitura usando o contexto da janela conectada, sem MCP.	**Agent mode** Tarefas de várias etapas com ferramentas do sql-tools e servidores MCP, com aprovação e citando banco e servidor no prompt.

Números que caem

Item	Valor
Limite padrão de recursão de CTE	**100 níveis; MAXRECURSION de 0 a 32767 (0 = sem limite)**
Número de erro aceito pelo THROW	**50.000 a 2.147.483.647, severidade 16**
datetime2	**faixa de 0001 a 9999, precisão de até 100 ns**
real	**equivale a float(24)**
EDIT_DISTANCE_SIMILARITY e JARO_WINKLER_SIMILARITY	**0 a 100 (100 = idênticos)**
DIFFERENCE (SOUNDEX)	**0 a 4**
REGEXP_LIKE, REGEXP_MATCHES e REGEXP_SPLIT_TO_TABLE	**nível de compatibilidade 170 (as TVF, salvo ALLOW_BUILTIN_TVF_IN_ALL_COMPAT_LEVELS)**
GitHub Copilot no SSMS	**SSMS 22 ou posterior; Agent mode a partir do SSMS 22.7**
Copilot in Fabric	**capacidade paga F2 ou superior, ou P1 ou superior (não funciona em avaliação)**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Devolver um valor decimal com RETURN CAST(@x AS decimal(10,2)).
Como não cair: RETURN só devolve inteiro e corta os centavos. Use parâmetro OUTPUT para valores e RETURN para status.

⚠ **Alerta de prova:** Criar UNIQUE achando que ele ignora NULLs e aceita vários.
Como não cair: No SQL Server, UNIQUE aceita um único NULL. Unicidade só entre preenchidos: índice único filtrado WHERE coluna IS NOT NULL.

⚠ **Alerta de prova:** Achar que o CHECK foi criado com NOCHECK porque NULL passou.
Como não cair: NULL produz UNKNOWN e não viola CHECK. Declare a coluna NOT NULL.

⚠ **Alerta de prova:** Usar UPDATE(coluna) num gatilho para saber se o valor mudou.
Como não cair: UPDATE() só indica que a coluna estava no SET. Compare inserted e deleted pela chave.

⚠ **Alerta de prova:** Acrescentar NULL ON NULL ao JSON_OBJECT para tirar as chaves nulas, ou ABSENT ON NULL ao JSON_ARRAY para manter os nulos.
Como não cair: Os padrões são opostos: JSON_OBJECT já é NULL ON NULL (use ABSENT ON NULL para omitir); JSON_ARRAY já é ABSENT ON NULL (use NULL ON NULL para manter).

⚠ **Alerta de prova:** Usar JSON_CONTAINS com 'ital%' sem o quarto argumento.
Como não cair: O search_mode padrão é 0 (igualdade) e procura o texto literal. Para semântica de LIKE, passe 1.

⚠ **Alerta de prova:** Esperar que LAST_VALUE com ORDER BY no OVER devolva o último da partição.
Como não cair: O quadro padrão termina na linha atual. Use ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING.

⚠ **Alerta de prova:** Usar NOT IN numa subconsulta que pode retornar NULL.
Como não cair: Um único NULL faz o NOT IN não retornar nada. Use NOT EXISTS correlacionado.

⚠ **Alerta de prova:** Remover o limite da CTE recursiva com MAXRECURSION 0 quando o requisito pede manter a proteção.
Como não cair: Use OPTION (MAXRECURSION n) com o valor necessário, na instrução externa; 0 tira a proteção contra laços infinitos.

⚠ **Alerta de prova:** Passar severidade 10 ou 18 ao THROW ou exigir o número em sys.messages.
Como não cair: THROW não tem parâmetro de severidade (sempre 16) e o número de 50.000 em diante não precisa estar em sys.messages; isso é do RAISERROR.

⚠ **Alerta de prova:** Usar QUOTENAME(@valor) concatenado no WHERE para evitar injeção.
Como não cair: QUOTENAME é para identificadores. Valores vão como parâmetros de sp_executesql.

⚠ **Alerta de prova:** Filtrar JARO_WINKLER_DISTANCE entre 0,9 e 1 ou as funções de similaridade entre 0 e 10 para achar nomes parecidos.
Como não cair: Distância perto de 0 é parecido; similaridade (EDIT_DISTANCE_SIMILARITY, JARO_WINKLER_SIMILARITY) perto de 100 é parecido.

⚠ **Alerta de prova:** Usar REGEXP_LIKE com o flag 'c' para aceitar maiúsculas e minúsculas.
Como não cair: 'c' é o padrão e diferencia maiúsculas. Para ignorar, use 'i'.

⚠ **Alerta de prova:** Confiar na caixa de aprovação do Agent mode ou em instruções personalizadas como controle de segurança.
Como não cair: A aprovação confirma intenção e instruções só orientam. O limite real são as permissões do login, ou o usuário de menor privilégio definido no CONSTITUTION.md.

Como cai na prova

📝 **Como cai:** Quatro trechos de T-SQL quase idênticos que diferem num argumento, flag ou cláusula.
Como resolver: Isole a diferença entre as alternativas e verifique o padrão daquela peça: cláusula de NULL, search_mode, flag de regex, quadro da janela, occurrence/grupo. A resposta é quem cumpre o requisito sem efeito colateral.

📝 **Como cai:** Escolher o objeto de programação certo (função, procedimento, view, gatilho) para um requisito.
Como resolver: Cálculo reutilizável dentro de SELECT ou WHERE é função; modificação com controle de transação é procedimento; consulta encapsulada sem parâmetro é view; reação automática a modificação é gatilho.

📝 **Como cai:** Tabela especializada para um requisito de negócio (histórico, imutabilidade, velocidade, arquivamento, relacionamentos).
Como resolver: Histórico consultável por tempo: temporal. Prova de não adulteração: ledger. Velocidade sem durabilidade: memória SCHEMA_ONLY. Arquivar por mês: particionamento com SWITCH. Muitos-para-muitos com caminhos: grafo.

📝 **Como cai:** Configurar ou usar uma ferramenta de IA (Copilot no SSMS, extensão MSSQL, Copilot in Fabric, MCP).
Como resolver: Identifique a ferramenta exata do enunciado e não misture: AI Assistance é do SSMS, configurações de tenant e capacidade são do Fabric, mcp.json é do VS Code, CONSTITUTION.md é por banco no SSMS. Depois confira o modo (Ask, Agent, Plan).

📝 **Como cai:** Tratamento de erro em transação de débito e crédito ou com XACT_ABORT ligado.
Como resolver: Procure TRY com COMMIT no fim, CATCH com ROLLBACK condicionado (@@TRANCOUNT ou XACT_STATE) e THROW. Descarte COMMIT no CATCH, @@ERROR lido tarde e SAVE TRANSACTION em transação condenada.

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em

Os três melhores de cada região	**ROW_NUMBER() OVER (PARTITION BY grupo ORDER BY critério, desempate) numa CTE filtrada fora**
Comparar com o mesmo mês do ano anterior	**LAG(coluna, 12, 0) OVER (ORDER BY AnoMes)**
Total acumulado que soma empates da mesma data	**ROWS UNBOUNDED PRECEDING e ordenação com coluna única de desempate**
Subordinados diretos e indiretos em profundidade variável	**CTE recursiva: âncora, UNION ALL e membro recursivo**
Remover duplicatas mantendo uma linha	**CTE com ROW_NUMBER() OVER (PARTITION BY chave) e DELETE onde o número > 1**
Registros que só podem ser inseridos e com prova de não adulteração	**WITH (LEDGER = ON, APPEND_ONLY = ON)**
Consultar como os dados estavam num momento passado, sem mudar o aplicativo	**Tabela temporal com SYSTEM_VERSIONING e FOR SYSTEM_TIME**
Enviar muitas linhas ao procedimento numa única chamada	**Tipo de tabela definido pelo usuário como parâmetro READONLY (TVP)**
Dividir texto por vários delimitadores diferentes	**REGEXP_SPLIT_TO_TABLE (value, ordinal)**
Nomes com erro de digitação ou abreviação	**EDIT_DISTANCE com máximo, EDIT_DISTANCE_SIMILARITY ou JARO_WINKLER**
Staging ou estado de sessão sem necessidade de durabilidade	**Tabela otimizada para memória com DURABILITY = SCHEMA_ONLY**
Menor caminho entre nós com até N saltos	**MATCH(SHORTEST_PATH(...{1,n})) com FOR PATH e WITHIN GROUP (GRAPH PATH)**
Convenções de T-SQL aplicadas automaticamente aos arquivos .sql no VS Code	**.github/instructions/nome.instructions.md com applyTo**
Copilot no SSMS deve executar com usuário de menor privilégio no banco	**CONSTITUTION.md com agentExecuteAsUser e IMPERSONATE**

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

- Padrões invertidos: JSON_OBJECT é NULL ON NULL, JSON_ARRAY é ABSENT ON NULL.
- CHECK deixa NULL passar e UNIQUE aceita um NULL; os dois pedem atenção a NOT NULL e índice filtrado.
- Gatilhos disparam uma vez por instrução: lógica em conjunto com inserted e deleted.
- No Agent mode do SSMS, MCP só funciona com as ferramentas marcadas no painel Tools e o agente roda com as permissões do login.
- Fora dos EUA e da França, o Copilot in Fabric precisa da permissão de processar dados fora da região geográfica.

Essenciais para a prova (checklist)

☐ Sei escrever TRY...CATCH com ROLLBACK condicionado e THROW, e interpretar XACT_STATE() = -1.

☐ Sei quando usar OUTPUT, RETURN, TVP READONLY e sp_executesql com parâmetros.

☐ Sei os requisitos de views indexadas e o efeito de SCHEMABINDING e WITH CHECK OPTION.

☐ Sei comportamento de NULL em CHECK, UNIQUE e NOT IN.

☐ Sei escolher entre IDENTITY e SEQUENCE e explicar as lacunas da sequência.

☐ Sei montar tabela temporal, consultar com FOR SYSTEM_TIME e escolher entre ledger append-only e atualizável.

☐ Sei inserir bordas com $from_id/$to_id e escrever MATCH com direção e aliases corretos.

☐ Sei a diferença entre JSON_VALUE, JSON_QUERY, OPENJSON WITH e os padrões de NULL de JSON_OBJECT e JSON_ARRAY.

☐ Sei usar REGEXP_* e as funções de distância e similaridade com as faixas corretas.

☐ Sei onde ficam instruções, mcp.json e CONSTITUTION.md e o que muda entre Ask, Agent e Plan mode.

Questões de revisão do capítulo

R2.1 · Projetar e desenvolver soluções de banco de dados

A Operadora de Cartões Paga Bem particionou a tabela Transacoes por mês com uma função RANGE RIGHT sobre DataTransacao (datetime2) e um esquema de partição. Ao criar a chave primária clusterizada só em TransacaoId sobre o esquema de partição, recebeu erro. Qual definição permite o índice clusterizado alinhado?

A. PRIMARY KEY CLUSTERED (TransacaoId) ON PS_Mes(TransacaoId)
B. PRIMARY KEY NONCLUSTERED (TransacaoId) ON [PRIMARY], sem esquema
C. PRIMARY KEY CLUSTERED (TransacaoId, DataTransacao) ON PS_Mes(DataTransacao)
D. PRIMARY KEY CLUSTERED (DataTransacao) ON PS_Mes(DataTransacao)

R2.2 · Projetar e desenvolver soluções de banco de dados

A tabela Chamados da Central de Suporte Resolve Já tem 20 milhões de linhas, mas só cerca de 2 mil ficam com Status = 'Aberto' ao mesmo tempo, e o painel dos atendentes consulta quase sempre esses chamados abertos por DataAbertura. Qual índice atende com o menor custo de armazenamento e manutenção?

A. CREATE INDEX IX_Data ON Chamados(DataAbertura) INCLUDE (Status, Descricao);
B. CREATE INDEX IX_Abertos ON Chamados(DataAbertura) WHERE Status = 'Aberto';
C. CREATE CLUSTERED COLUMNSTORE INDEX CCI_Chamados ON Chamados;
D. CREATE INDEX IX_Status ON Chamados(Status, DataAbertura) INCLUDE (Descricao);

R2.3 · Projetar e desenvolver soluções de banco de dados

Na Seguradora Auto Proteção, uma analista configurou no SSMS um servidor MCP que consulta a base de conhecimento interna e habilitou as ferramentas dele. Ela faz perguntas no modo Ask e o Copilot nunca consulta a base. O que ela deve fazer?

A. Repetir a pergunta incluindo o nome do servidor MCP entre colchetes no prompt
B. Trocar a conversa para o Agent mode (o Ask mode não usa MCP)
C. Adicionar a URL do servidor MCP ao arquivo de instruções do repositório
D. Criar a propriedade CONSTITUTION.md no banco para autorizar o servidor MCP

R2.4 · Projetar e desenvolver soluções de banco de dados

Na Fabricante de Móveis Madeira Nobre, o administrador do Fabric habilitou a configuração de tenant que permite usar o Copilot, mas delegou as configurações de tenant aos administradores de capacidade. Os desenvolvedores do SQL database no workspace da capacidade F64 de produção continuam sem o Copilot. O que falta?

A. O administrador da capacidade habilitar o Copilot nas configurações delegadas da própria capacidade
B. Cada desenvolvedor instalar a extensão MSSQL no Visual Studio Code e entrar com a conta do GitHub
C. Criar um banco SQL novo no workspace, porque o Copilot não vale para bancos criados antes da ativação
D. Converter a capacidade F64 para uma capacidade de avaliação, que já inclui o Copilot por padrão

Respostas comentadas

R2.1. C) PRIMARY KEY CLUSTERED (TransacaoId, DataTransacao) ON PS_Mes(DataTransacao)

Por quê: Particionamento usa função de partição (limites, RANGE LEFT ou RIGHT), esquema de partição (mapeia para filegroups) e coluna de partição. Índices exclusivos alinhados precisam conter a coluna de partição na chave; índices alinhados permitem SWITCH e manutenção por partição.

✗ A) Incorreta. A função de partição é sobre datetime2; usar TransacaoId como coluna de partição não corresponde à função.

✗ B) Incorreta. Um índice não alinhado impede a troca rápida de partições, essencial para o arquivamento mensal.

✗ D) Incorreta. Só a data como chave primária não garante unicidade das transações.

> **Dica de prova:** PK em tabela particionada: inclua a coluna de partição na chave.

R2.2. B) CREATE INDEX IX_Abertos ON Chamados(DataAbertura) WHERE Status = 'Aberto';

Por quê: Índices filtrados usam uma cláusula WHERE para indexar só um subconjunto de linhas bem definido, reduzindo armazenamento, custo de manutenção e estatísticas, e acelerando consultas cujo predicado corresponde ao filtro.

✗ A) Incorreta. Um índice completo por data também cobre 20 milhões de linhas sem necessidade.

✗ C) Incorreta. Columnstore é voltado a análises de grandes volumes, não a buscas pontuais em poucas linhas.

✗ D) Incorreta. Indexar a tabela inteira por Status cria um índice grande para servir a uma fração mínima das linhas.

> **Dica de prova:** Poucas linhas consultadas sempre com o mesmo filtro: índice filtrado.

R2.3. B) Trocar a conversa para o Agent mode (o Ask mode não usa MCP)

Por quê: O Agent mode do SSMS pode ser estendido com servidores MCP para integrar sistemas externos. É preciso estar no Agent mode e habilitar as ferramentas do servidor no painel Tools; o Ask mode responde sem usar ferramentas MCP.

✗ A) Incorreta. Citar o servidor no prompt não faz o Ask mode usar ferramentas MCP.

✗ C) Incorreta. Arquivos de instruções não configuram nem habilitam servidores MCP.

✗ D) Incorreta. O CONSTITUTION.md define regras e contexto de execução do banco, não habilita MCP.

> **Dica de prova:** Ferramenta MCP no SSMS: só no Agent mode.

R2.4. A) O administrador da capacidade habilitar o Copilot nas configurações delegadas da própria capacidade

Por quê: Habilitar o Copilot no Fabric envolve as configurações de tenant (que podem ser restritas a grupos de segurança), as configurações delegadas de capacidade quando há delegação e a atribuição do workspace a uma capacidade paga compatível.

✗ B) Incorreta. A extensão MSSQL com GitHub Copilot é outra ferramenta; o Copilot no editor do Fabric depende das configurações do tenant e da capacidade.

✗ C) Incorreta. Não há essa restrição por data de criação do banco.

✗ D) Incorreta. O Copilot no Fabric não é suportado em SKUs de avaliação; exige capacidade paga F2 ou superior, ou P1 ou superior.

> **Dica de prova:** Configuração de tenant delegada: o admin da capacidade também precisa habilitar o Copilot.

03

CAPÍTULO 3 · 35-40% DA PROVA

Proteger, otimizar e implantar soluções de banco de dados

Aqui eu preciso escolher o recurso certo de proteção pelo requisito exato (quem não pode ver, onde fica a chave, se é exibição ou criptografia), diagnosticar concorrência e desempenho pelo sintoma descrito e montar uma esteira de CI/CD segura para projetos SQL. Também entram o Data API builder (configuração REST, GraphQL, cache, permissões e telemetria) e o tratamento de mudanças com CES, CDC, Change Tracking, Azure Functions e Logic Apps. As alternativas erradas quase sempre são recursos reais que resolvem um problema vizinho: TDE no lugar de Always Encrypted, NOLOCK no lugar de RCSI, Change Tracking no lugar de CDC. Onde mais erro é em detalhe de configuração: o cache do DAB que precisa ser ligado em dois lugares, o prefixo +! da análise de código e os pré-requisitos do optimized locking. A regra de ouro é ler o requisito como contrato: menor privilégio, sem senha, sem alterar código, sem manter bloqueio.

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

Tópicos-chave

Always Encrypted

Criptografa colunas no driver do cliente; as chaves de criptografia de coluna são protegidas por uma chave mestra de coluna guardada fora do mecanismo (Azure Key Vault, repositório de certificados ou HSM). Nem o administrador do banco vê o texto claro. A aplicação usa Column Encryption Setting=Enabled na cadeia de conexão e passa os valores por parâmetros, que o driver cifra antes de enviar.

Determinística x aleatória, _BIN2 e enclaves

A determinística gera sempre o mesmo valor cifrado para o mesmo texto e permite igualdade, junções, GROUP BY e índices; a aleatória é mais segura e, sem enclave, não permite operação nenhuma. Colunas de texto cifradas exigem colação _BIN2, como Latin1_General_BIN2. Com enclaves seguros (por exemplo VBS no Azure SQL Database) e chave habilitada para enclave, passam a valer comparações, LIKE, ordenação e criptografia no local.

Transparent Data Encryption (TDE)

Protege dados em repouso: arquivos de dados, de log e backups, contra roubo de mídia ou acesso físico. É transparente para as aplicações e o mecanismo descriptografa as páginas na leitura, então quem tem SELECT lê tudo normalmente. Não trabalha por coluna nem por usuário.

Criptografia em nível de coluna (ENCRYPTBYKEY)

Usa chave mestra do banco, certificado, chave simétrica protegida por ele, OPEN SYMMETRIC KEY e ENCRYPTBYKEY/DECRYPTBYKEY. As chaves ficam dentro do banco e o controle é por permissão sobre as chaves, com mais código. Não exige driver especial, mas um administrador com acesso às chaves consegue descriptografar.

Dynamic Data Masking

Altera só o que é exibido a quem não tem UNMASK; o dado continua em texto claro. Funções: default() oculta tudo, email() mostra a primeira letra e sufixo constante, random() para tipos numéricos, partial(prefixo,'preenchimento',sufixo) e datetime. UNMASK pode ser concedido no banco, schema, tabela ou coluna, e a máscara não impede inferência por filtros em consultas ad hoc.

Row-Level Security (RLS)

Combina uma função com valor de tabela inline criada WITH SCHEMABINDING (o predicado) e uma CREATE SECURITY POLICY com STATE = ON. FILTER PREDICATE oculta silenciosamente linhas em SELECT, UPDATE e DELETE; BLOCK PREDICATE (AFTER INSERT, AFTER UPDATE, BEFORE UPDATE, BEFORE DELETE) barra gravações fora do escopo. Com usuário de aplicação compartilhado, use sp_set_session_context com @read_only = 1 e leia SESSION_CONTEXT no predicado.

Permissões e menor privilégio

GRANT no nível de schema cobre todos os objetos, inclusive os criados depois; GRANT aceita lista de colunas para SELECT, UPDATE e REFERENCES. DENY prevalece sobre qualquer GRANT, venha de onde vier. Pela cadeia de propriedade, EXECUTE no procedimento basta para acessar tabelas do mesmo dono; prefira funções de banco personalizadas a db_owner, db_datareader e db_datawriter.

Acesso sem senha com Microsoft Entra

Identidades gerenciadas eliminam credenciais: a atribuída pelo sistema vive e morre com o recurso, a atribuída pelo usuário pode ser compartilhada por vários aplicativos. O usuário contido é criado com CREATE USER [nome] FROM EXTERNAL PROVIDER, a cadeia usa Authentication='Active Directory Default' e a autenticação somente do Microsoft Entra desabilita a autenticação SQL. Para o banco chamar endpoints de modelo, use DATABASE SCOPED CREDENTIAL com IDENTITY = 'Managed Identity' e SECRET com o resourceid (vale a identidade do usuário primária; sem ela, a do sistema); no SQL Server 2025 a instância precisa do Azure Arc e de 'allow server scoped db credentials'.

Auditoria

No SQL Server Audit, CREATE SERVER AUDIT define o destino, SERVER AUDIT SPECIFICATION captura eventos de servidor (logins, permissões) e DATABASE AUDIT SPECIFICATION captura ações em objetos do banco, como SELECT numa tabela; o arquivo é lido com sys.fn_get_audit_file. No Azure SQL Database os destinos são Azure Storage, Log Analytics ou Event Hubs, configurados no servidor ou no banco.

Níveis de isolamento

READ UNCOMMITTED (equivale a NOLOCK) permite leitura suja; READ COMMITTED evita só a leitura suja; REPEATABLE READ ainda permite fantasmas; SERIALIZABLE evita fantasmas com bloqueios de intervalo. SNAPSHOT exige ALLOW_SNAPSHOT_ISOLATION ON, dá a foto do início da transação e pode gerar conflito de atualização. RCSI (READ_COMMITTED_SNAPSHOT) dá versão por instrução sem mudar código e já vem ligado no Azure SQL Database.

Optimized locking

Combina bloqueio de ID de transação (TID), que mantém um único bloqueio por transação e libera os de linha logo após cada atualização, com lock after qualification (LAQ), que avalia predicados na versão confirmada mais recente sem bloquear. Exige ADR habilitado antes; o LAQ depende de RCSI. No SQL Server 2025 vem desligado por padrão; no Azure SQL Database e no Fabric já está ligado.

Concorrência, bloqueios e deadlocks

Concorrência otimista sem bloqueio: coluna rowversion no WHERE do UPDATE e conferência de @@ROWCOUNT. Ler para atualizar sem corrida: WITH (UPDLOCK, HOLDLOCK). Deadlock é ciclo de bloqueios; a vítima recebe o erro 1205, e a prevenção é ordem consistente de acesso, transações curtas, índices e versão de linha, com nova tentativa na aplicação. Na cadeia de bloqueio, a cabeça é quem não espera ninguém (blocking_session_id em sys.dm_exec_requests).

Query Store e ajuste automático

Guarda planos e estatísticas por consulta ao longo do tempo. sp_query_store_force_plan fixa um plano anterior; sp_query_store_set_hints aplica hints (MAXDOP, RECOMPILE, MAX_GRANT_PERCENT) sem mudar código e tem precedência sobre plan guides. Ao atingir MAX_STORAGE_SIZE_MB ele vira READ_ONLY em silêncio. O ajuste automático tem FORCE_LAST_GOOD_PLAN, CREATE_INDEX e DROP_INDEX, e o Query Performance Insight depende do Query Store.

Nível de compatibilidade e PSP

O nível de compatibilidade é por banco e libera recursos de processamento inteligente: 150 traz inlining de UDF escalar e modo batch em rowstore; 160 traz a otimização de plano sensível a parâmetro (PSP), que cria um plano dispatcher com variantes por cardinalidade; 170 traz otimização de parâmetros opcionais. O Azure não sobe o nível sozinho: capture linha de base no Query Store, teste e force planos antigos se regredir.

Projetos SQL SDK-style (Microsoft.Build.Sql) e build

Projetos declarativos, um arquivo .sql por objeto, globbing automático, referências em pacotes NuGet e build multiplataforma com .NET 8 ou superior. O build é offline: valida sintaxe, referências entre objetos e recursos da plataforma de destino (DSP, como Sql170 para SQL Server 2025 ou SqlAzureV12 para Azure SQL Database) e gera o .dacpac. RunSqlCodeAnalysis liga a análise de código e SqlCodeAnalysisRules usa - para desabilitar e +! para transformar a regra em erro.

SqlPackage e implantação

Publish compara o .dacpac com o destino e aplica só as diferenças. Script gera o T-SQL e DeployReport gera o XML das operações, ambos sem alterar o banco; Extract com ExtractTarget=SchemaObjectType mais git status detecta drift. BlockOnPossibleDataLoss (padrão True) interrompe mudanças com perda de dados e DropObjectsNotInSource (padrão False) mantém objetos fora do projeto. Pre e post-deployment são um de cada, rodam em toda implantação, incluem arquivos via :r e devem ser idempotentes (MERGE).

Data API builder (DAB)

dab init cria o arquivo (connection-string com @env()), dab add inclui entidades (--source.type table, view ou stored-procedure) e dab start executa. Views precisam dos campos de chave; procedimentos usam a ação execute, com padrão POST no REST e mutation no GraphQL. Permissões combinam role, actions, fields include/exclude e policy.database com @item e @claims; relacionamentos (cardinality, linking.object) só navegam no GraphQL.

Tratamento de mudanças: CES, CDC, Change Tracking

Change event streaming publica DML como CloudEvents (JSON ou Avro) no Azure Event Hubs ou Fabric Eventstream, para vários consumidores; configura-se com sp_enable_event_stream, sp_create_event_stream_group e sp_add_object_to_event_stream_group. CDC lê o log e grava valores antes e depois no schema cdc (jobs do Agent no SQL Server). Change Tracking só diz que a linha mudou e a versão, é síncrono e é o requisito do gatilho SQL das Azure Functions.

Macetes para memorizar

💡 TDE x Always Encrypted

TDE tranca o disco; Always Encrypted tranca o dado até chegar ao cliente.

💡 Escada de isolamento

RU lê sujo, RC tira o sujo, RR tira o não repetível, SERIALIZABLE tira o fantasma; SNAPSHOT tira o fantasma com foto, não com bloqueio.

💡 Cache do DAB

Cache tem duas chaves: runtime E entidade. Uma só não abre a porta.

💡 Análise de código

Menos desliga, mais-exclamação grita: -Regra desabilita, +!Regra quebra o build.

💡 Detecção de mudanças

CT conta QUE mudou; CDC conta COMO era; CES avisa a TODOS.

💡 Optimized locking

ADR antes de tudo; LAQ só anda com RCSI.

💡 Padrões do SqlPackage

Bloqueia perda (BlockOnPossibleDataLoss True), não apaga estranho (DropObjectsNotInSource False).

💡 Máscaras do DDM

email() mostra a letra; partial(0,'XXX',4) mostra o fim; random() só para número; default() esconde tudo.

Tabela de memorização: não confunda

Quando é...	...e quando é
Always Encrypted Nem o DBA pode ler o valor e a chave precisa ficar fora do mecanismo, num cofre controlado pela segurança.	**TDE** Proteger arquivos de dados, log e backups contra roubo de mídia, sem mudar nada na aplicação.
Criptografia determinística A aplicação precisa de igualdade, junção, GROUP BY ou índice na coluna cifrada sem enclave.	**Criptografia aleatória** Máxima segurança e nenhuma operação na coluna, ou operações ricas usando enclave seguro.
FILTER PREDICATE Ocultar silenciosamente as linhas de outros locatários em leituras, atualizações e exclusões.	**BLOCK PREDICATE** Impedir que alguém insira ou atualize linhas para fora do próprio escopo (AFTER INSERT, AFTER UPDATE).
SNAPSHOT Todas as consultas da transação precisam ver a mesma foto do início dela; a sessão pede o nível e aceita conflitos de atualização.	**READ COMMITTED SNAPSHOT (RCSI)** Leitores e escritores deixarem de se bloquear sem alterar código, com consistência por instrução.
Change Tracking Basta saber quais linhas mudaram e a operação desde uma versão (sincronização incremental, gatilho SQL das Functions).	**Change Data Capture** É preciso o histórico com valores antes e depois de cada alteração, lido em lote.

sp_query_store_force_plan Houve regressão após troca de plano e existe um plano anterior bom para fixar.	**sp_query_store_set_hints** É preciso moldar a execução de uma consulta (MAXDOP 1, RECOMPILE) sem mexer no código nem no banco inteiro.
Identidade gerenciada atribuída pelo sistema Um único recurso que precisa de identidade própria, com ciclo de vida atrelado a ele.	**Identidade gerenciada atribuída pelo usuário** Vários aplicativos com exatamente as mesmas permissões, reduzindo os usuários de banco a gerenciar.

Números que caem

Item	Valor
Erro devolvido à vítima de deadlock	**1205**
Nível de compatibilidade da PSP	**160 ou superior**
Inlining de UDF escalar e modo batch em rowstore	**nível de compatibilidade 150**
Otimização de parâmetros opcionais	**nível de compatibilidade 170**
Paginação do DAB: default-page-size / max-page-size (padrões)	**100 / 100.000**
$first=-1 no DAB	**usa o max-page-size configurado**
Hyperscale (modelo vCore)	**até 128 TB**
Database connections do Static Web Apps	**aposentado em 30/11/2025**
Plataforma de destino (DSP) para SQL Server 2025 / Azure SQL Database	**Sql170 / SqlAzureV12**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Aceitar o pull request porque o Git fez o merge sem conflito nos arquivos .sql.
Como não cair: O Git compara texto, não semântica. Rode o build do projeto SQL depois do merge para validar referências entre objetos.

⚠ **Alerta de prova:** Achar que o TDE falhou porque um analista com SELECT leu os dados.
Como não cair: TDE é dado em repouso. Esconder dado de quem consulta é papel de Always Encrypted, permissões, máscara ou RLS.

⚠ **Alerta de prova:** Usar Dynamic Data Masking como se fosse criptografia ou como barreira contra administradores.
Como não cair: A máscara só muda a exibição; o dado fica em texto claro, filtros usam o valor real e admins veem tudo.

⚠ **Alerta de prova:** Colocar NOLOCK achando que lê a versão confirmada mais antiga ou que evita a corrida de 'ler e atualizar'.
Como não cair: NOLOCK é READ UNCOMMITTED: leitura suja. Versão confirmada é RCSI/SNAPSHOT; corrida se resolve com UPDLOCK, HOLDLOCK.

⚠ **Alerta de prova:** Aumentar o ttl-seconds global do DAB esperando que o cache passe a valer.
Como não cair: O cache precisa estar ligado em runtime.cache e também em cache.enabled na entidade; o TTL da entidade prevalece.

⚠ **Alerta de prova:** Colocar o app em db_datareader e db_datawriter como solução de menor privilégio.
Como não cair: Essas funções fixas valem para o banco inteiro e não dão EXECUTE. Crie função personalizada com permissões no schema.

⚠ **Alerta de prova:** Conceder VIEW DEFINITION achando que o usuário passa a ler os dados.
Como não cair: VIEW DEFINITION mostra a definição do objeto; ler dados é SELECT (no schema, se precisar cobrir objetos futuros).

⚠ **Alerta de prova:** Escolher REPEATABLE READ para impedir leituras fantasmas.
Como não cair: REPEATABLE READ segura as linhas lidas, mas aceita linhas novas no intervalo. Sem fantasma: SERIALIZABLE ou SNAPSHOT.

⚠ **Alerta de prova:** Resolver um plano que serve a um valor e não a outro com OPTION (RECOMPILE), DBCC FREEPROCCACHE ou forçando o plano atual.
Como não cair: Sem mudar código e com planos por cardinalidade: PSP, ligando o nível de compatibilidade 160 ou superior.

⚠ **Alerta de prova:** Ligar só OPTIMIZED_LOCKING acreditando que ele habilita ADR e RCSI sozinho.
Como não cair: ADR é pré-requisito e precisa vir antes; o LAQ só funciona com RCSI habilitado.

⚠ **Alerta de prova:** Acreditar que o Change Tracking guarda os valores anteriores ou depende do SQL Server Agent.
Como não cair: Change Tracking registra chave, operação e versão, de forma síncrona e leve. Valores anteriores e jobs do Agent são do CDC.

⚠ **Alerta de prova:** Usar SqlCodeAnalysisRules com -Microsoft.Rules.Data.SR0004 para a regra quebrar o build.
Como não cair: O prefixo - desabilita a regra; +! transforma a detecção em erro. E RunSqlCodeAnalysis = True liga a análise.

⚠ **Alerta de prova:** Supor que o build do projeto SQL conecta ao banco ou detecta drift.
Como não cair: O build é offline: valida o modelo e gera o .dacpac. Drift se vê com comparação de esquema (.scmp) ou SqlPackage Extract.

⚠ **Alerta de prova:** Tratar deadlock com SET LOCK_TIMEOUT maior, DEADLOCK_PRIORITY ou desligando a detecção.
Como não cair: Esperar mais não desfaz o ciclo e a prioridade só escolhe a vítima. Previna com ordem de acesso, transações curtas e versão de linha; trate o 1205 com nova tentativa.

Como cai na prova

📝 **Como cai:** Cenário de dado sensível com várias tecnologias de proteção nas alternativas (Always Encrypted, TDE, máscara, criptografia de coluna).
Como resolver: Identifique quem não pode ver (usuário comum, admin, ladrão de backup) e onde a chave deve ficar. Admin bloqueado e chave fora do banco: Always Encrypted. Só exibição: máscara. Arquivo e backup: TDE. Chave dentro do banco com controle por permissão: ENCRYPTBYKEY.

📝 **Como cai:** Sintoma de concorrência descrito em linguagem de negócio (valor que some, venda duplicada, sobrescrita silenciosa, relatório bloqueando).
Como resolver: Traduza o sintoma para a anomalia (leitura suja, atualização perdida, fantasma, bloqueio leitor x escritor) e escolha a menor intervenção que respeite as restrições do enunciado, como não mudar código ou manter READ COMMITTED no resto do sistema.

📝 **Como cai:** Trecho de configuração do dab-config.json ou requisição REST/GraphQL com uma peça faltando.
Como resolver: Confira o nível certo (runtime x entidade), as palavras-chave exatas ($select, $filter, $orderby, $first, $after; first, filter, orderBy, after no GraphQL) e se segredos estão em @env(). Desconfie de nomes inventados como $top, where ou sort.

📝 **Como cai:** Escolha de ferramenta de diagnóstico de desempenho (DMV, Query Store, Extended Events, Query Performance Insight).
Como resolver: Pergunte o que precisa ser medido: CPU acumulada por consulta é sys.dm_exec_query_stats com sys.dm_exec_sql_text; regressão de plano é Query Store; deadlock no Azure SQL Database é database_xml_deadlock_report em XE de escopo de banco; visão no portal é Query Performance Insight.

📝 **Como cai:** Controle de CI/CD: impedir merge sem revisão, implantar só da main, aprovar antes de produção.
Como resolver: Merge e revisão na branch: política de branch ou CODEOWNERS. Credenciais de produção só da main: verificação de controle de branch na service connection. Aprovação e espera antes de implantar: regras de proteção do ambiente. Segredos: secrets de ambiente, AzureKeyVault@2 ou OIDC.

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
Nem os administradores podem ler e as chaves ficam em cofre fora do banco	**Always Encrypted com chave mestra de coluna no Azure Key Vault**
Busca por igualdade ou junção na coluna cifrada, sem enclave	**Criptografia determinística (texto com colação _BIN2)**
LIKE ou comparação de faixa em coluna do Always Encrypted	**Enclaves seguros com chave habilitada para enclave**
Valores que aparecem no relatório e depois somem	**Leitura suja: NOLOCK equivale a READ UNCOMMITTED**
Leitores não podem bloquear escritores e não se pode mudar o código	**ALTER DATABASE ... SET READ_COMMITTED_SNAPSHOT ON**
Detectar que outro usuário salvou antes, sem manter bloqueio durante a edição	**rowversion no WHERE do UPDATE e conferência de @@ROWCOUNT**
Duas sessões leem o mesmo saldo e vendem o último item duas vezes	**Ler o saldo com WITH (UPDLOCK, HOLDLOCK) dentro da transação**
Rápido para valores pequenos e lento para o valor que concentra a maioria das linhas	**Plano sensível a parâmetro (PSP), compatibilidade 160+**
Query Store parou de registrar consultas novas	**Ficou READ_ONLY por MAX_STORAGE_SIZE_MB; aumentar o limite com limpeza AUTO**
Vários sistemas devem receber cada alteração com baixa latência, sem polling	**Change event streaming para Event Hubs (CloudEvents)**

Vários aplicativos com as mesmas permissões e sem senha	**Uma identidade gerenciada atribuída pelo usuário e um único usuário no banco**
Revisar exatamente o que será executado antes de publicar, sem alterar o banco	**SqlPackage /Action:Script e /Action:DeployReport**
Alterações em certos caminhos precisam de revisão obrigatória de um time	**CODEOWNERS (GitHub) ou revisores incluídos automaticamente (Azure DevOps)**
Workflow precisa autenticar no Azure sem guardar segredo	**OpenID Connect com credencial federada e azure/login (client-id, tenant-id, subscription-id)**

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

- DENY sempre vence GRANT, venha o GRANT de função ou direto.
- O build do projeto SQL é offline; quem compara com o banco é SqlPackage ou a comparação de esquema.
- No Azure SQL Database o RCSI e o optimized locking já vêm ligados; no SQL Server 2025 o optimized locking vem desligado e exige ADR.
- Relacionamentos do DAB só navegam no GraphQL; GET de item inexistente devolve 200 com value vazio.
- Change Tracking é o requisito do gatilho SQL das Azure Functions; os gatilhos do conector do Logic Apps pedem IDENTITY (criado) e ROWVERSION (modificado).

Essenciais para a prova (checklist)

☐ Sei escolher entre Always Encrypted, TDE, criptografia de coluna e Dynamic Data Masking pelo requisito de quem não pode ver.

☐ Sei quando usar criptografia determinística, aleatória e enclaves seguros, e que texto cifrado exige colação _BIN2.

☐ Sei montar RLS com função inline WITH SCHEMABINDING, FILTER e BLOCK PREDICATE e SESSION_CONTEXT com @read_only = 1.

☐ Sei conceder permissão no schema, por coluna, via EXECUTE em procedimento e sei que DENY prevalece.

☐ Sei criar usuário de identidade gerenciada com FROM EXTERNAL PROVIDER e escolher entre identidade do sistema e do usuário.

☐ Sei associar cada anomalia (suja, não repetível, fantasma) ao nível de isolamento que a evita, e diferenciar SNAPSHOT de RCSI.

☐ Sei os pré-requisitos do optimized locking (ADR antes, RCSI para LAQ).

☐ Sei usar Query Store para forçar plano, aplicar hints e recuperar o modo READ_ONLY.

☐ Sei o que fazem Publish, Script, DeployReport e Extract do SqlPackage e os padrões de BlockOnPossibleDataLoss e DropObjectsNotInSource.

☐ Sei configurar entidades, permissões, cache, paginação, GraphQL seguro e telemetria no Data API builder.

Questões de revisão do capítulo

R3.1 · Proteger, otimizar e implantar soluções de banco de dados

O procedimento usp_CancelarPedido da Loja de Móveis Casa Linda deve gerar um erro de severidade 16 quando o pedido já foi enviado. A equipe quer um teste unitário do SQL Server que passe só quando esse erro acontecer e falhe se o procedimento terminar sem erro. Como marcar o teste?

A. Adicionar SET XACT_ABORT ON ao script de pré-teste do teste unitário do procedimento
B. Adicionar uma condição Row Count com zero linhas esperadas ao final do teste unitário
C. Adicionar ao método de teste o atributo ExpectedSqlException com a severidade
D. Adicionar uma condição Scalar Value com o valor 16 na primeira célula do resultado

R3.2 · Proteger, otimizar e implantar soluções de banco de dados

O time de BI da Rede de Postos Bom Caminho cria novas views no schema Relatorios toda semana. A função LeitoresBI deve conseguir ler todas as views desse schema, inclusive as que forem criadas no futuro, sem o DBA precisar conceder permissão a cada nova view. Qual comando atende?

A. GRANT SELECT ON SCHEMA::Relatorios TO LeitoresBI;
B. GRANT VIEW DEFINITION ON SCHEMA::Relatorios TO LeitoresBI;
C. ALTER ROLE db_datareader ADD MEMBER LeitoresBI;
D. GRANT SELECT ON DATABASE::BomCaminho TO LeitoresBI;

R3.3 · Proteger, otimizar e implantar soluções de banco de dados

O aplicativo da Rede de Cinemas Grande Tela consome o Data API builder e precisa listar, da entidade Filme, só os campos titulo e duracao dos filmes com classificacao igual a 'Livre', ordenados por titulo, 20 por página. Qual requisição REST atende?

A. GET /api/Filme/classificacao/Livre?$orderby=titulo&$top=20&$select=titulo,duracao
B. GET /api/Filme?fields=titulo,duracao&where=classificacao='Livre'&sort=titulo&limit=20
C. GET /api/Filme?$select=titulo,duracao&$filter=classificacao eq 'Livre'&$orderby=titulo&$first=20
D. POST /api/Filme com o corpo {"select": "titulo,duracao", "filter": "classificacao = Livre"}

R3.4 · Proteger, otimizar e implantar soluções de banco de dados

A Transportadora Rota Sul habilitou Transparent Data Encryption (TDE) no banco de entregas. Um analista com permissão SELECT na tabela de clientes executou uma consulta e viu os endereços normalmente. O gestor achou que o TDE tinha falhado. Qual afirmação explica o comportamento?

A. O TDE criptografa as colunas, mas libera a leitura para quem tem SELECT por ter uma exceção configurada
B. O TDE só passa a valer depois que todas as consultas forem refeitas com parâmetros e a aplicação for reconectada
C. O TDE precisa de uma política de máscara em cada coluna para que o texto apareça cifrado nos resultados
D. O TDE protege dados em repouso, como arquivos e backups, e não impede que um usuário autorizado leia os dados

Respostas comentadas

R3.1. C) Adicionar ao método de teste o atributo ExpectedSqlException com a severidade

Por quê: Testes negativos validam que o código falha corretamente com entradas inválidas. Em testes unitários do SQL Server, marque o método de teste com ExpectedSqlException (Severity, State, MatchFirstError) no arquivo C# gerado.

✗ A) Incorreta. XACT_ABORT muda o tratamento de transações, mas não transforma o teste em teste negativo.
✗ B) Incorreta. Contar linhas não verifica se o erro esperado ocorreu.
✗ D) Incorreta. A severidade de um erro não é retornada como valor do conjunto de resultados.

> **Dica de prova:** Teste que espera erro: atributo ExpectedSqlException.

R3.2. A) GRANT SELECT ON SCHEMA::Relatorios TO LeitoresBI;

Por quê: A hierarquia de permissões vai de servidor a banco, schema e objeto. GRANT no nível de schema cobre todos os objetos do schema, inclusive os criados depois, o que simplifica a gestão sem ampliar demais o acesso.

✗ B) Incorreta. VIEW DEFINITION permite ver a definição dos objetos, não ler os dados.
✗ C) Incorreta. db_datareader dá leitura em todas as tabelas e views do banco, mais do que o necessário.
✗ D) Incorreta. SELECT no banco também abrange todos os schemas, violando o menor privilégio.

> **Dica de prova:** Ler tudo de um schema, inclusive objetos futuros: GRANT SELECT ON SCHEMA::.

R3.3. C) GET /api/Filme?$select=titulo,duracao&$filter=classificacao eq 'Livre'&$orderby=titulo&$first=20

Por quê: Palavras-chave REST do Data API builder: $select (campos), $filter (condição), $orderby (ordem), $first (itens por página) e $after (continuação). Os nomes de caminho e parâmetros diferenciam maiúsculas e minúsculas.

✗ A) Incorreta. O caminho com nome de coluna e valor serve para chave primária, e $top não é a palavra-chave de paginação do DAB.

✗ B) Incorreta. Esses nomes de parâmetro não são as palavras-chave da API REST do DAB.

✗ D) Incorreta. POST cria registros; não é usado para consultar com filtros.

> **Dica de prova:** DAB REST: $select, $filter, $orderby, $first e $after.

R3.4. D) O TDE protege dados em repouso, como arquivos e backups, e não impede que um usuário autorizado leia os dados

Por quê: Transparent Data Encryption protege os arquivos de dados, de log e os backups contra acesso físico ou roubo de mídia. Não protege contra quem tem permissão de consulta; para isso use Always Encrypted, permissões, máscara ou RLS.

✗ A) Incorreta. O TDE não trabalha por coluna nem com exceções por usuário; ele cifra os arquivos do banco inteiro.

✗ B) Incorreta. Consultas parametrizadas são exigência do Always Encrypted, não do TDE, que é transparente para as aplicações.

✗ C) Incorreta. Máscara de dados é outro recurso; o TDE não altera o resultado das consultas.

> **Dica de prova:** TDE = dado em repouso; não esconde nada de quem consulta com permissão.

DIA DA PROVA

Estratégia de prova

A prova do DP-800 é de cenário: uma empresa com um requisito de negócio e várias opções técnicas reais, das quais só uma cumpre todas as restrições do enunciado. As restrições decisivas aparecem como frases curtas, como sem alterar o código, sem senha, menor privilégio, sem manter bloqueios ou sem alterar o banco. Muitas alternativas trazem trechos de T-SQL ou de configuração quase idênticos, e a diferença está num argumento, num padrão ou num nível de configuração. Há também questões de múltipla resposta (Escolha DUAS ou TRÊS) em que cada afirmação deve ser julgada isoladamente. Nomes atuais importam: Microsoft Entra ID, Microsoft Foundry, SQL MCP Server, Copilot in Fabric.

Como responder

✓ Administre os 120 minutos: são cerca de 50 itens (a Microsoft não divulga o número exato), então mire pouco mais de 2 minutos por item e marque para revisão o que travar.

✓ Distribua o esforço pelos pesos: Projetar e desenvolver (35-40%) e Proteger, otimizar e implantar (35-40%) somam a maior parte; IA (25-30%) é menor, mas tem muita sintaxe decorável.

✓ Sublinhe mentalmente as restrições do enunciado (sem senha, sem mudar código, menor privilégio, sem bloqueio) e descarte primeiro as alternativas que as violam.

✓ Quando os trechos de código forem quase iguais, compare só o que muda entre eles e verifique o padrão daquela peça.

✓ Em múltipla resposta, avalie cada afirmação como verdadeira ou falsa sozinha; não escolha por eliminação de pares.

✓ Desconfie de recursos reais aplicados ao problema errado: TDE para esconder dado de usuário, máscara como criptografia, Change Tracking para valores anteriores, VECTOR_DISTANCE para usar índice.

✓ Desconfie de palavras-chave inventadas ou de outra plataforma ($top, where, sort, VECTOR(MAX), MODEL_TYPE CHAT, ALTER VECTOR INDEX UPGRADE).

✓ A nota mínima é 700 numa escala de 1 a 1000: não deixe nenhum item em branco.

Erros que derrubam candidatos

✗ Inverter o sentido de métricas: distância vetorial menor é mais parecido; similaridade de texto 100 é idêntico.

✗ Confundir os padrões de NULL de JSON_OBJECT e JSON_ARRAY e o search_mode padrão do JSON_CONTAINS.

✗ Trocar SNAPSHOT por RCSI ou achar que REPEATABLE READ evita fantasmas.

✗ Configurar algo no nível errado: cache do DAB só no runtime, MAXDOP de instância em vez de banco, UNMASK no banco inteiro em vez da coluna.

✗ Escolher a solução que funciona mas fere o requisito: db_owner, MAXRECURSION 0, regenerar todos os embeddings toda noite quando só algumas linhas mudam, chamar o modelo dentro da transação.

✗ Misturar ferramentas de IA: carga AI Assistance é do SSMS, configurações de tenant são do Fabric, mcp.json é do VS Code, CONSTITUTION.md é do banco no SSMS.

✗ Ler a resposta do modelo no caminho errado de @response ou aceitar o corte de 4.000 caracteres do JSON_VALUE.

Na semana da prova

- ☐ Liste os recursos que você ainda confunde e, para cada um, escreva em uma frase por que o recurso vizinho não atende ao mesmo requisito.
- ☐ Revise os pares que a prova confunde: Always Encrypted x TDE, SNAPSHOT x RCSI, CT x CDC x CES, kNN x ANN, texto completo x vetorial.
- ☐ Decore os números de bolso: 1205, compatibilidades 150/160/170, 1.998 e 3.996 dimensões, 100 linhas, k = 60, @timeout 1 a 230, retry 0 a 10.
- ☐ Releia a sintaxe exata de VECTOR_SEARCH, CREATE EXTERNAL MODEL, sp_invoke_external_rest_endpoint e das funções JSON e REGEXP.
- ☐ Na véspera, revise só os checklists de cada domínio e durma; no dia, leia cada enunciado até o fim antes de olhar as alternativas.

REVISÃO FINAL

Folha de cola

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

1. Implementar recursos de IA em soluções de banco de dados

‣ Medir quanto a busca aproximada se aproxima da exata → **Recall comparando os vizinhos do ANN com os do kNN**

‣ Poucos milhares de vetores depois de filtros seletivos → **kNN exato com VECTOR_DISTANCE tende a bastar**

‣ Usuário descreve o que precisa em vez de nomear o item → **Busca vetorial semântica com embeddings**

‣ Ora digitam códigos exatos, ora descrevem a necessidade → **Busca híbrida com RRF**

‣ Ordenar resultados de texto completo por relevância → **Junção com CONTAINSTABLE por [KEY], ordenando por [RANK] DESC**

⚠ Em VECTOR_DISTANCE e VECTOR_SEARCH, menor é mais parecido em todas as métricas; a ordem é crescente.

⚠ VECTOR_DISTANCE é sempre exato. Só VECTOR_SEARCH com TOP WITH APPROXIMATE usa o índice.

⚠ Com WITH APPROXIMATE, ORDER BY é obrigatório, só aceita distance e só crescente. Outras ordenações vão para consulta externa.

2. Projetar e desenvolver soluções de banco de dados

‣ Os três melhores de cada região → **ROW_NUMBER() OVER (PARTITION BY grupo ORDER BY critério, desempate) numa CTE filtrada fora**

‣ Comparar com o mesmo mês do ano anterior → **LAG(coluna, 12, 0) OVER (ORDER BY AnoMes)**

‣ Total acumulado que soma empates da mesma data → **ROWS UNBOUNDED PRECEDING e ordenação com coluna única de desempate**

‣ Subordinados diretos e indiretos em profundidade variável → **CTE recursiva: âncora, UNION ALL e membro recursivo**

‣ Remover duplicatas mantendo uma linha → **CTE com ROW_NUMBER() OVER (PARTITION BY chave) e DELETE onde o número > 1**

⚠ RETURN só devolve inteiro e corta os centavos. Use parâmetro OUTPUT para valores e RETURN para status.

⚠ No SQL Server, UNIQUE aceita um único NULL. Unicidade só entre preenchidos: índice único filtrado WHERE coluna IS NOT NULL.

⚠ NULL produz UNKNOWN e não viola CHECK. Declare a coluna NOT NULL.

3. Proteger, otimizar e implantar soluções de banco de dados

‣ Nem os administradores podem ler e as chaves ficam em cofre fora do banco → **Always Encrypted com chave mestra de coluna no Azure Key Vault**

‣ Busca por igualdade ou junção na coluna cifrada, sem enclave → **Criptografia determinística (texto com colação _BIN2)**

‣ LIKE ou comparação de faixa em coluna do Always Encrypted → **Enclaves seguros com chave habilitada para enclave**

‣ Valores que aparecem no relatório e depois somem → **Leitura suja: NOLOCK equivale a READ UNCOMMITTED**

‣ Leitores não podem bloquear escritores e não se pode mudar o código → **ALTER DATABASE ... SET READ_COMMITTED_SNAPSHOT ON**

⚠ O Git compara texto, não semântica. Rode o build do projeto SQL depois do merge para validar referências entre objetos.

⚠ TDE é dado em repouso. Esconder dado de quem consulta é papel de Always Encrypted, permissões, máscara ou RLS.

⚠ A máscara só muda a exibição; o dado fica em texto claro, filtros usam o valor real e admins veem tudo.

GABARITO

Respostas do diagnóstico

D1. D) Remover o índice com DROP INDEX e recriá-lo com CREATE VECTOR INDEX → revise o capítulo 1

Por quê: A versão de um índice vetorial aparece em build_parameters de sys.vector_indexes. Índices antigos não são atualizados no lugar: remova e recrie com CREATE VECTOR INDEX, de preferência em janela de manutenção, porque a busca aproximada fica indisponível enquanto o índice não existe.

✗ A) Incorreta. Não existe opção VERSION para reconstruir índices vetoriais; a reconstrução não muda o formato.
✗ B) Incorreta. O nível de compatibilidade não converte o formato de índices vetoriais existentes.
✗ C) Incorreta. Não há instrução ALTER VECTOR INDEX ... UPGRADE; a migração documentada é DROP e CREATE.

> **Dica de prova:** Índice vetorial em versão antiga: DROP INDEX e CREATE VECTOR INDEX de novo.

D2. A) VECTORPROPERTY, com 'Dimensions' e 'BaseType' → revise o capítulo 1

Por quê: VECTORPROPERTY(vetor, propriedade) aceita 'Dimensions' e 'BaseType' e ajuda a validar dados e diagnosticar incompatibilidade de dimensões, comum quando modelos diferentes geraram os vetores.

✗ B) Incorreta. A distância até um vetor de zeros não informa as dimensões e exige vetores de mesmo tamanho.
✗ C) Incorreta. VECTOR_NORMALIZE escala o vetor para comprimento 1; não informa propriedades.
✗ D) Incorreta. VECTOR_NORM retorna a magnitude do vetor, não as dimensões nem o tipo.

> **Dica de prova:** Quantas dimensões e qual tipo base: VECTORPROPERTY.

D3. D) CONTAINS(Laudo, 'FORMSOF(INFLECTIONAL, fraturar) AND NEAR((joelho, ligamento))') → revise o capítulo 1

Por quê: CONTAINS aceita termos simples, frases entre aspas, prefixos com asterisco, FORMSOF(INFLECTIONAL ou THESAURUS, ...) e NEAR para proximidade, combinados com AND, OR e AND NOT.

✗ A) Incorreta. FREETEXT busca pelo significado de um texto livre e não garante que joelho apareça próximo de ligamento; proximidade é expressa com NEAR em CONTAINS.
✗ B) Incorreta. LIKE não usa o índice de texto completo nem entende flexões ou proximidade.
✗ C) Incorreta. Termos entre aspas buscam a palavra ou frase exata, sem flexões nem proximidade.

> **Dica de prova:** Flexões e proximidade no texto completo: FORMSOF(INFLECTIONAL) e NEAR dentro de CONTAINS.

D4. D) JSON_CONTAINS(Perfil, 'ital%', '$.idiomas[*]', 1) = 1 → revise o capítulo 2

Por quê: JSON_CONTAINS(alvo, valor, caminho, search_mode) procura um valor SQL em um caminho do documento e retorna 1, 0 ou NULL (se o caminho não existe). search_mode 0 usa igualdade e 1 usa semântica de LIKE para valores de texto; para arrays, use curinga no caminho.

✗ A) Incorreta. JSON_PATH_EXISTS testa se um caminho existe, não o conteúdo dos valores do array.

✗ B) Incorreta. JSON_VALUE não retorna arrays; para $.idiomas devolve NULL em modo lax.

✗ C) Incorreta. Sem search_mode, o padrão é 0, igualdade; o texto 'ital%' seria procurado literalmente.

> **Dica de prova:** Procurar texto com padrão dentro de array JSON: JSON_CONTAINS com search_mode = 1.

D5. C) Acrescentar NULL ON NULL ao final da lista de argumentos do JSON_ARRAY → revise o capítulo 2

Por quê: JSON_ARRAY constrói um array a partir de expressões. A cláusula de nulos controla o tratamento de NULL: ABSENT ON NULL (padrão do JSON_ARRAY) omite o elemento e NULL ON NULL gera o literal null.

✗ A) Incorreta. ISNULL com string vazia gera "" em vez de null, o que muda o significado esperado pelo front-end.

✗ B) Incorreta. ABSENT ON NULL é justamente o comportamento padrão que está removendo o elemento.

✗ D) Incorreta. JSON_ARRAYAGG agrega valores de várias linhas; aqui os valores estão em colunas da mesma linha.

> **Dica de prova:** JSON_ARRAY descarta NULL por padrão; para manter a posição, use NULL ON NULL.

D6. D) No arquivo .vscode/mcp.json, com a chave raiz servers e o servidor do tipo stdio com command e args → revise o capítulo 2

Por quê: Clientes MCP leem a lista de servidores de um arquivo de configuração. No VS Code, o arquivo do workspace é .vscode/mcp.json com a chave servers; servidores HTTP usam url e servidores locais usam stdio com command e args. Credenciais devem vir de inputs ou variáveis de ambiente.

✗ A) Incorreta. Arquivos de instruções orientam as respostas; não registram servidores MCP.

✗ B) Incorreta. O dab-config.json configura o Data API builder, não o cliente MCP do VS Code.

✗ C) Incorreta. A configuração de MCP fica em mcp.json, e no VS Code a chave raiz é servers, não mcpServers.

> **Dica de prova:** MCP no VS Code para o time: .vscode/mcp.json com "servers".

D7. C) runtime.telemetry.application-insights com enabled true e connection-string "@env('APPINSIGHTS_CONN')" → revise o capítulo 3

Por quê: O DAB integra-se ao Application Insights por runtime.telemetry.application-insights (enabled e connection-string). Use a cadeia de conexão completa via @env(); a documentação alerta para possível dupla instrumentação no App Service e recomenda contêineres, Container Apps ou AKS.

✗ A) Incorreta. set-session-context passa claims ao SQL; não tem relação com telemetria.

✗ B) Incorreta. O cache reduz idas ao banco; não envia telemetria.

✗ D) Incorreta. O modo development altera detalhes de erro e recursos de desenvolvimento, não configura o Application Insights.

> **Dica de prova:** DAB no Application Insights: runtime.telemetry.application-insights com a cadeia via @env().

D8. D) NOLOCK equivale a READ UNCOMMITTED e permite ler alterações ainda não confirmadas, que podem ser revertidas → revise o capítulo 3

Por quê: READ UNCOMMITTED (equivalente ao hint NOLOCK) é o nível mais permissivo: permite leituras sujas, leituras não repetíveis e fantasmas. Para leituras consistentes sem bloquear escritores, prefira READ COMMITTED SNAPSHOT ou SNAPSHOT.

✗ A) Incorreta. NOLOCK não adquire bloqueios compartilhados e não bloqueia escritores.
✗ B) Incorreta. O hint não tem relação com cache de planos nem com backups.
✗ C) Incorreta. Ler versões confirmadas é o comportamento do RCSI ou do SNAPSHOT, não do NOLOCK.

> **Dica de prova:** Valor que aparece e some com NOLOCK: leitura suja (READ UNCOMMITTED).

D9. B) RunSqlCodeAnalysis = True e SqlCodeAnalysisRules com +!Microsoft.Rules.Data.SR0004 → revise o capítulo 3

Por quê: A análise de código de projetos SQL identifica problemas de design, nomenclatura e desempenho. RunSqlCodeAnalysis a executa no build; SqlCodeAnalysisRules desabilita regras com - e transforma detecções em erro com +!.

✗ A) Incorreta. Essas propriedades não ativam a análise de código nem elevam a regra a erro.
✗ C) Incorreta. São propriedades de implantação do SqlPackage, não de análise de código no build.
✗ D) Incorreta. O prefixo - desabilita a regra, o oposto do desejado.

> **Dica de prova:** Regra de análise de código quebrando o build: SqlCodeAnalysisRules com +!RegraId.

Agora, treine no formato real

Faça as 6 provas completas do curso DP-800: 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.