DP-420 · CHEAT SHEET · CERTIFYDECK

Microsoft Certified: Azure Cosmos DB AI Developer Associate

Apuntes basados en los objetivos oficiales y materiales clave del certificado DP-420, para repasar antes del examen.

Nivel: Intermediate (Associate) Duración: 100 min Nota de corte: 700/1000 Fuente: guía de estudio oficial de Microsoft Learn

1. Design and develop database solutions

40–45%

Evaluate resource models and SDK solutions

- Azure Cosmos DB ofrece cinco niveles de coherencia: alta, de obsolescencia limitada, de sesión, de prefijo coherente y posible. Los niveles alta y de obsolescencia limitada leen de dos réplicas y duplican el coste en RU de las lecturas; el nivel de sesión garantiza leer lo propio escrito mediante tokens de sesión. La invalidación por solicitud con ConsistencyLevel solo puede relajar el nivel predeterminado de la cuenta.
- El rendimiento aprovisionado puede ser estándar (manual) o con escalabilidad automática en el contenedor (recomendado) o compartido en la base de datos. Se aplica en todas las regiones de la cuenta. El modo sin servidor encaja con tráfico intermitente, en una sola región. La capacidad de ráfaga aprovecha la capacidad inactiva para picos breves y la ejecución basada en prioridades limita primero las solicitudes de prioridad baja.
- Las bases de datos y los contenedores se crean con CreateDatabaseIfNotExistsAsync y CreateContainerIfNotExistsAsync en .NET o con create_database_if_not_exists y create_container_if_not_exists en Python, y con az cosmosdb sql en la CLI. Las propiedades modificables se cambian con replace_container. Con Microsoft Entra ID, las cuentas de servicio usan la operación de plano de control RBAC predefinida.

Implement database operations

- Las consultas parametrizadas usan la notación @ (QueryDefinition.WithParameter en .NET, parameters en query_items de Python) y evitan la inyección SQL. Los resultados llegan en páginas limitadas por MaxItemCount; hay que leer hasta que no queden páginas y el token de continuación, único por consulta, permite reanudar en cualquier momento, salvo con GROUP BY o DISTINCT sin ORDER BY.
- Una lectura puntual (ReadItemAsync en .NET, read_item en Python) busca un elemento por id y valor completo de la clave de partición, salvo los ORDER BY sin filtro de igualdad. Es la lectura más barata: su coste depende del tamaño del elemento y del nivel de coherencia. Una consulta que filtra por id y clave de partición no es una lectura puntual; ReadMany (ReadManyItemsAsync) lee varios elementos a la vez.
- CreateItemAsync falla si el id ya existe. ReplaceItemAsync y patch solo funcionan sobre elementos existentes y UpsertItemAsync crea o reemplaza. La revisión parcial (patch) envía solo las rutas modificadas con Add, Set, Replace, Remove, Increment y Move, admite un predicado de filtro condicional y no puede tocar propiedades del sistema; delete_item falla si el elemento no existe.
- El lote transaccional (CreateTransactionalBatch en .NET, execute_item_batch en Python) agrupa operaciones de un mismo valor de clave de partición con garantías ACID: si una falla, se revierte todo y las demás devuelven 424. La compatibilidad masiva del SDK v3 (AllowBulkExecution, una tarea por operación) maximiza el rendimiento entre particiones sin atomicidad y sustituye a la biblioteca bulk executor.

Design and implement data models and partitioning strategies

- El TTL se configura en segundos con DefaultTimeToLive en el contenedor (ausente o null: desactivado; -1: activado sin caducidad predeterminada; n: caducan a los n segundos) y se puede reemplazar por elemento con la propiedad ttl, contada desde la última modificación (_ts). Una tarea en segundo plano borra los caducados con RU sobrantes, aunque depende de las consultas de lectura.
- Empieza diseñando las consultas de lectura y escritura de la aplicación y mide la latencia y las RU de cada versión del modelo con ellas. En cargas con muchas lecturas, desnormaliza y duplica datos en contenedores con otra clave de partición, sincronizados con la fuente de cambios, y agrupa en una partición lógica los elementos que deben actualizarse juntos.
- Elige una propiedad inmutable, de tipo cadena y de alta cardinalidad que reparta por igual las RU y el almacenamiento. En contenedores grandes con muchas lecturas, debe aparecer como filtro de igualdad en la mayoría de las consultas. No encaja en cargas de lecturas puntuales y escrituras. La clave no se puede cambiar in situ: hay que mover los datos a un contenedor nuevo.
- Inserta los datos en relaciones contenidas o de uno a pocos que cambian poco, no crecen sin límite y se leen juntos. Usa mejor las lecturas. Referencia en relaciones de uno a muchos o de muchos a muchos, datos que cambian a menudo o colecciones ilimitadas, suele mejorar las escrituras. Los modelos híbridos desnormalizan solo los campos que se muestran y guardan agregados precalculados.
- El rendimiento aprovisionado se reparte por igual entre las particiones físicas y una partición lógica nunca se divide, así que una clave desequilibrada crea particiones activas con respuestas 429 aunque sobre capacidad. Detéctalas con NormalizedRUConsumption por PartitionKeyRangeId y CDBPartitionKeyRUConsumption, y vigila el almacenamiento con CDBPartitionKeyStatistics. La solución duradera es cambiar la clave migrando a un contenedor nuevo.
- Las claves de partición jerárquicas admiten hasta tres niveles (kind MultiHash), se definen al crear el contenedor y permiten que un prefijo como TenantId supere los límites de una partición lógica. Las consultas por prefijo se dirigen a un subconjunto de particiones y las lecturas puntuales usan la clave completa con PartitionKeyBuilder. Con pocos inquilinos o cargas de esta naturaleza conviene una clave sintética id.

Implement change feed

- La fuente de cambios está habilitada de forma predeterminada, no consume RU del rendimiento aprovisionado y garantiza el orden solo dentro de un valor de clave de partición. Se lee con Azure Functions, el procesador de fuente de cambios o el modelo de extracción, y sirve para notificaciones, vistas materializadas, movimiento de datos, migraciones sin tiempo de inactividad y Event Sourcing.
- El procesador de fuente de cambios (solo .NET y Java) guarda el estado en un contenedor de concesiones, reparte el trabajo entre instancias y reintenta los lotes fallidos. El modelo de extracción usa tokens de continuación que guardas tú, permite releer la fuente desde el principio o controlar el ritmo, y pagina con PeekNext, pero sin equilibrio automático ni garantía de entrega al menos una vez.
- El contenedor de concesiones guarda el estado y coordina los trabajadores, puede estar en la misma cuenta o en otra, vale está particionado debe usar /id, las instancias de una unidad comparten processorName y contenedor de concesiones con nombres de instancia distintos, varias unidades o desencadenadores comparten nombre y base y StartTime solo actúa la primera vez, y conviene usar el punto de conexión global.
- El modo de versión más reciente registra creaciones y actualizaciones, solo la última versión de cada elemento y sin eliminaciones (se expulsan con TTL), y puede empezar desde el principio. El modo de todas las versiones y eliminaciones exige copia de seguridad continua, se habilita en una cuenta existente, captura eliminaciones con metadatos y solo lee desde dentro de la ventana de retención.
- Las trabajos de copia de contenedor copian elementos a un contenedor nuevo para cambiar la clave de partición, las claves únicas, el nivel del rendimiento o aplanar claves jerárquicas, la copia sin conexión exige detener las escrituras en el origen; la copia en línea requiere copia de seguridad continua y el modo de todas las versiones y eliminaciones, y se integra con Azure Data Factory para orquestar.

Design and implement Azure Cosmos DB solutions by using AI-assisted tools

- El Azure Cosmos DB Agent Kit ofrece reglas para revisar claves de partición, consultas y código del SDK: aportas formas de documento, filtros, tasas de escritura, directiva de indexación, cargos de RU o diagnósticos saneados, sin credenciales. Sus recomendaciones son sugerencias: compara RU, latencia y resultados sobre los mismos datos y prueba fuera de producción antes de aplicarlas.
- El Azure Cosmos DB Agent Kit se instala con npx skills add AzureCosmosDB/cosmosdb-agent-kit o con el método de complementos del host, y se verifica buscando la skill cosmosdb-best-practices. Aporta guía, no acceso: no autentica en Azure ni consulta datos; el servidor MCP de Microsoft Learn sirve documentación. Conviene cargar solo la skill relevante para no saturar el contexto.

2. Secure, optimize, and deploy database solutions

30–35%

Implement database security controls

- Para exigir Microsoft Entra ID se establece disableLocalAuth en true (az resource update o Bicep) y se valida que una conexión con clave falle; las aplicaciones usan una identidad administrada y DefaultAzureCredential. Si aún hay claves, se rotan cambiando primero a la otra clave, y la rotación segura bloquea regenerar o deshabilitar claves que siguen en uso.
- El plano de datos usa roles nativos (Cosmos DB Built-in Data Reader y Data Contributor, o roles personalizados con dataActions) asignados con az cosmosdb sql role assignment create en el ámbito de cuenta, base de datos o contenedor. notDataActions no se admite, readMetadata es siempre necesario y las consultas del SDK requieren executeQuery y readChangeFeed. Los roles de Azure como Cosmos DB Operator solo administran.
- La red de Azure Cosmos DB for NoSQL se protege con firewall de IP (ipRules), puntos de conexión de servicio con reglas de red virtual y puntos de conexión privados con Private Link y la zona privatelink.documents.azure.com. Con publicNetworkAccess en Disabled se bloquea todo el tráfico público y de red virtual. Al activar un punto de conexión de servicio, el origen deja de ser la IP pública.
- Dynamic Data Masking enmascara en el servidor, al consultar, las rutas de la directiva con las estrategias Default, MaskSubstring o Email sin cambiar los datos almacenados. Requiere Microsoft Entra ID (no claves) y el desenmascarado se concede con la acción items/unmask, incluida en Data Contributor y no en Data Reader. Las rutas excluidas exigen incluir / y la fuente de cambios exige unmask.

Implement availability and recovery solutions

- La copia de seguridad periódica guarda instantáneas a intervalos configurables y solo se restaura en una cuenta nueva mediante una solicitud de soporte. La continua permite la restauración a un momento dado en autoservicio en una cuenta nueva o, para recursos eliminados, en la misma cuenta. Migrar de periódica a continua es irreversible, y la restauración no recupera firewall, puntos de conexión privados ni asignaciones de roles.
- El SDK envía las lecturas a la primera región disponible de su lista de preferencias (ApplicationPreferredRegions, o ApplicationRegion para ordenarlas por proximidad) y nunca lee de regiones que no estén en esa lista. Con escrituras en varias regiones no se admite la coherencia alta y los conflictos se resuelven con Last Write Wins, por _ts o una ruta numérica propia, o con una directiva personalizada que solo se define al crear el contenedor.
- La conmutación por error automática por partición (PPAF) mueve a otra región, según el orden de prioridad de conmutación por error, solo las particiones afectadas de una cuenta de NoSQL con una región de escritura, y vuelve atrás sola. Requiere al menos una región de lectura, rendimiento aprovisionado y un SDK compatible; no admite obsolescencia limitada y hay que deshabilitarla para combinar particiones o poner una región sin conexión.

Optimize and remediate database performance

- Para bajar el cargo en RU de una consulta, filtra por la clave de partición, evita en WHERE funciones que no usan el índice (UPPER, LOWER, GetCurrentTimestamp) y añade índices compuestos con las igualdades primero y un único filtro de rango al final. Si filtras y ordenas por propiedades distintas, añade las del filtro al ORDER BY; en JOIN, filtra cada matriz con una subconsulta.
- Las métricas de consulta indican dónde se va el tiempo (TotalTime, IndexLookupTime, DocumentLoadTime, RuntimeExecutionTime) y si hubo exploración: si Retrieved Document Count supera a Output Document Count, parte de la consulta no usó el índice. En .NET, Diagnostics.GetQueryMetrics() devuelve métricas acumuladas y por partición de cada página, y hay que sumarlas en todos los viajes de ida y vuelta.
- Las métricas de índice se activan por consulta con PopulateIndexMetrics (populate_index_metrics en Python) y muestran los índices simples y compuestos utilizados y los potenciales. Los potenciales son recomendaciones que hay que añadir y comprobar, y la puntuación de impacto, alta o baja, solo depende de la forma de la consulta. Añaden sobrecarga, así que se usan solo para depurar.
- Para saber si la latencia de extremo a extremo se debe al servicio, compara las métricas Server Side Latency Direct o Gateway, según el modo de conexión, con lo que mide el cliente. Los diagnósticos del SDK separan el tiempo de red (Transit time frente a BELatencyInMs), la CPU y la falta de subprocesos del cliente, y en .NET CosmosThresholdOptions registra automáticamente las operaciones que superan tu objetivo.
- Azure Monitor permite alertas de métricas (por ejemplo, Normalized RU Consumption o Total Request Units filtrada por StatusCode 429), alertas del registro de actividad (rotación de claves) y alertas de búsqueda de registros sobre tablas de Log Analytics como CDBPartitionKeyStatistics. Las métricas se recopilan sin configuración; los registros necesitan una configuración de diagnóstico. Un grupo de acciones define correo, SMS o funciones.

Implement Fleet management and monitoring across multiple accounts and subscriptions

- Una flota agrupa cuentas de Azure Cosmos DB (una por inquilino) en fleetspaces. Un grupo de rendimiento del fleetspace comparte RU/s entre cuentas de distintas suscripciones y grupos de recursos: cada recurso consume primero sus RU/s dedicadas y luego las del grupo. Todas las cuentas deben tener las mismas regiones y el mismo modo de escritura, y el grupo rinde con muchos inquilinos de tráfico variado.
- Fleet analytics exporta cada hora datos de uso, aprovisionamiento y coste de todas las cuentas de una flota a un lakehouse de Microsoft Fabric (OneLake) o a ADLS Gen2, como tablas Delta Lake con esquema en estrella (FactRequestHourly, DimResource...). Hay que dar acceso de escritura a la entidad de servicio Cosmos DB Fleet Analytics; sirve para tendencias, no para solicitudes concretas.

3. Implement AI and analytics capabilities in database solutions

20–25%

Design and implement AI retrieval

- La búsqueda de texto completo necesita una directiva de texto completo (fullTextPaths con ruta e idioma) y un índice fullTextIndexes sobre esa ruta. FullTextContains, FullTextContainsAll y FullTextContainsAny filtran en WHERE, admiten búsqueda aproximada con distancia de edición, y FullTextScore clasifica por BM25 solo en ORDER BY RANK o RRF, sin poder proyectarse. Las facetas se obtienen con GROUP BY y COUNT.
- La directiva de incrustación de vectores declara ruta, tipo de datos (float32, float16, int8 o uint8), dimensiones y función de distancia (cosine, dotproduct o euclidean). El índice vectorial puede ser flat (exacto), quantizedFlat (fuerza bruta comprimida) o diskANN (aproximado y eficiente a escala); conviene excluir la ruta del vector del índice normal, y estas opciones no se cambian en su sitio: se quitan y se vuelven a añadir.
- Sharded DiskANN usa vectorIndexShardKey en el índice diskANN para crear un índice por cada valor de una propiedad (inquilino, categoría o el primer nivel de una clave jerárquica), con menos RU y latencia y mejor recuperación; la consulta debe filtrar ese valor en WHERE. Sin clave hay un índice por partición física. Conviene acotar las búsquedas a la clave de partición y evitar la distribución a todas las particiones.
- La búsqueda vectorial se activa en la cuenta (EnableNoSQLVectorSearch) y se ejecuta con VectorDistance en ORDER BY, siempre con TOP N para no disparar RU y latencia. VectorDistance admite un argumento booleano que fuerza fuerza bruta y un objeto de opciones (distanceFunction, dataType, multiplicadores, filterPriority); se combina con filtros WHERE y con la clave de partición. diskANN es aproximado y puede variar entre ejecuciones.
- La búsqueda híbrida combina VectorDistance (significado semántico) y FullTextScore (BM25, coincidencia de palabras) con la función RRF en una cláusula ORDER BY RANK. Requiere directiva vectorial, directiva de texto completo, índice vectorial e índice de texto completo. RRF admite pesos opcionales en el orden de las funciones, no se proyecta ni se combina con ORDER BY sobre otras propiedades y acepta filtros WHERE.

Design and implement agent memory stores

- Azure Cosmos DB sirve como almacén duradero de memoria del agente: memoria a corto plazo (turnos, con TTL) y a largo plazo (resúmenes, hechos, preferencias). Se recomienda un documento por turno; la clave de partición puede ser un GUID (escrituras masivas), threadId o una clave jerárquica tenantId/threadId para multiinquilino. La fuente de cambios con Durable Functions procesa los turnos nuevos para derivar memorias.
- El estado de la conversación se guarda como un elemento por turno con partición por threadId y se recupera con TOP @k filtrando por hilo y ordenando por marca de tiempo (o _ts) descendente. El TTL por elemento caduca turnos sueltos; el lote transaccional actualiza turno y metadatos del hilo de forma atómica en la misma partición lógica, y la coherencia de sesión requiere reutilizar el CosmosClient para leer las propias escrituras.
- La memoria semántica guarda incrustaciones por turno o por hecho y las recupera con VectorDistance en ORDER BY con TOP, filtrando por hilo, usuario o inquilino. diskANN conviene a gran escala y quantizedFlat a conjuntos pequeños o filtrados; vectorIndexShardKey crea un índice por inquilino. La búsqueda híbrida combina VectorDistance y FullTextScore con RRF.

Design and implement operational analytics on Azure Cosmos DB

- Cosmos DB in Microsoft Fabric usa el mismo motor que Azure Cosmos DB for NoSQL, pero se administra dentro de Fabric: refleja sus datos en OneLake de forma automática, se factura en unidades de capacidad, solo admite Microsoft Entra ID y el modo Gateway, usa escalabilidad automática y una sola región. Para escrituras multirregión, sin servidor o procedimientos almacenados, usa Azure Cosmos DB y refléjalo en Fabric.
- La creación de reflejo replica una base de datos de Azure Cosmos DB for NoSQL en OneLake en formato Delta sin consumir RU ni usar la fuente de cambios. Requiere copia de seguridad continua y se conecta con claves de lectura y escritura o Microsoft Entra ID (readMetadata y readAnalytics). Monitor replication muestra el estado y la última actualización; reiniciar la replicación vuelve a cargar todo desde cero.
- En el punto de conexión de análisis SQL, cada contenedor reflejado es una tabla de solo lectura y los datos anidados llegan como cadenas JSON que se expanden con OPENJSON y CROSS APPLY u OUTER APPLY, con o sin cláusula WITH. Para Spark, un acceso directo de lakehouse a la base de datos reflejada permite usar cuadernos sin acceder a la cuenta. Las combinaciones entre bases reflejadas usan nombres de tres partes.
- El Spark connector de Cosmos DB (formato cosmos.oltp) se conecta directamente al punto de conexión de Cosmos DB, a diferencia de leer la copia reflejada en OneLake, y sirve para ETL inverso con baja latencia; sus escrituras consumen RU. La Catalog API (CosmosCatalog) crea contenedores y permite spark.sql, y spark.cosmos.write.strategy elige ItemOverwrite o ItemPatch. Con Cosmos DB in Fabric se usa AccessToken y el modo Gateway.