Module 4

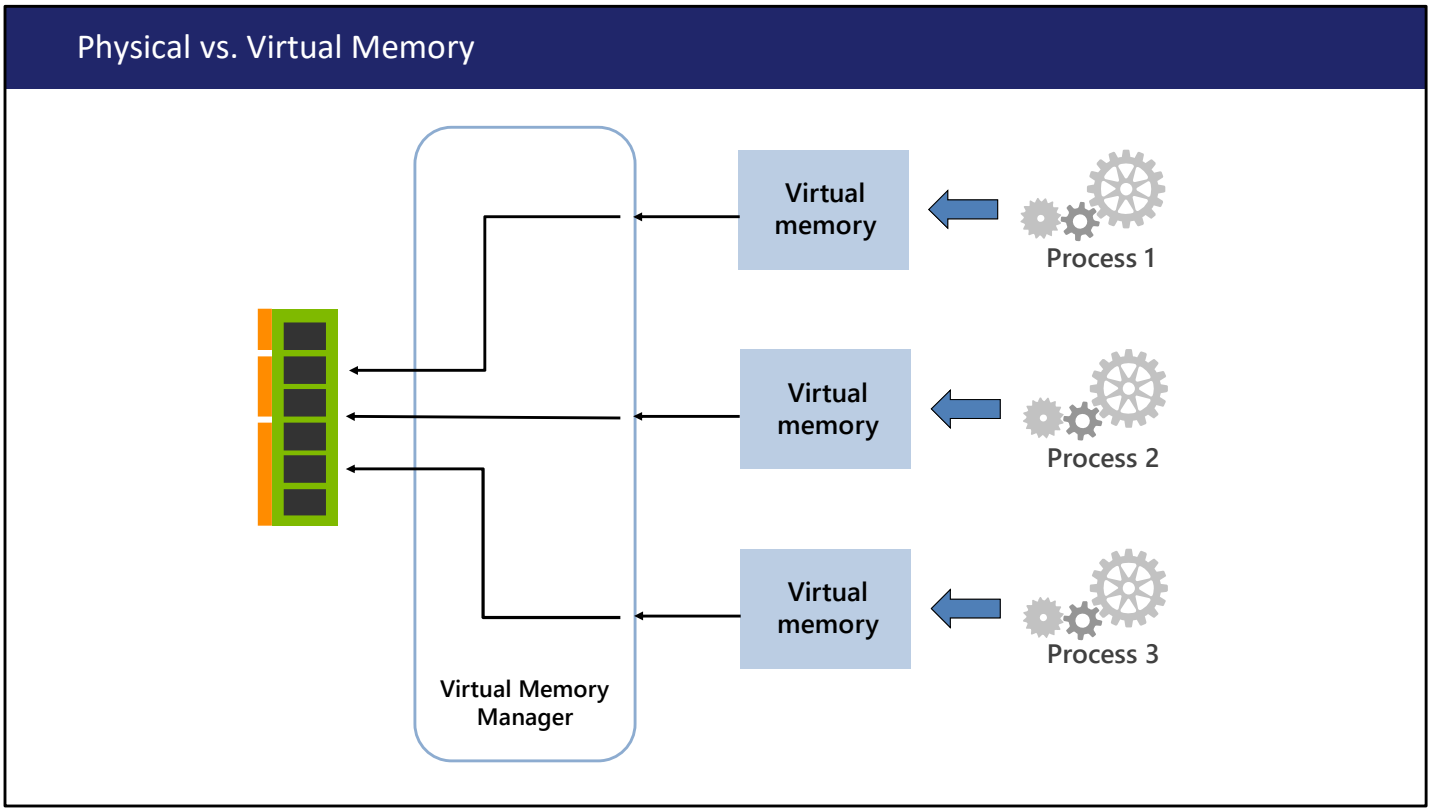
Memory usage

Module Overview

- Windows memory
- SQL Server memory
- In-Memory OLTP

Lesson: Windows memory

- Physical vs. virtual memory
- Virtual Address Space



Memory Management Architecture Guide
https://docs.microsoft.com/en-us/sql/relational-databases/memory-management-architecture-guide

Buffer Management:
SQL Server's buffer management is crucial for performance, handling the storage and retrieval of 8-KB data or index pages in memory, mirroring disk page sizes. The buffer cache stores these pages, managed by the buffer manager, which reads from or writes to disk as needed. Pages are only written back if modified, potentially multiple times, enhancing efficiency. Upon startup, SQL Server calculates the buffer cache size based on system memory and other parameters, only using physical memory as needed. This adaptive approach, alongside optimizations like bundling read requests for quicker ramp-up, ensures efficient memory use and fast data access. Cooperation with other system components ensures balanced memory and resource usage.

SQL Server's Virtual Memory Manager (VMM) optimizes how the database system uses physical and virtual memory. It coordinates memory allocation, ensuring SQL Server processes have the necessary memory resources without overextending system capacities. VMM allows SQL Server to use more memory than physically available by managing a virtual address space, swapping data between RAM and disk as needed (paging). This mechanism helps in handling large databases and complex operations that exceed physical memory limits. Essentially, VMM balances between performance and available memory resources, dynamically adjusting to workload demands to maintain efficient operations and prevent system overload.

Virtuellt minne är en teknik som låter datorns operativsystem använda hårddiskutrymme som tillfälligt minne för att kompensera för brist på fysiskt RAM. Detta gör att datorn kan köra fler

applikationer samtidigt och hantera större data än vad den fysiska minneskapaciteten tillåter, vilket ökar systemets flexibilitet och effektivitet.

Virtual Address Space, VAS

- VAS, abstraction layer between application and physical memory
- VAS size
  - 8 TB before Windows 2012
  - 128 TB Windows 2012 and later
  - Theoretical limit is 16 Exabyte, but OS sets limitation

Denna slide handlar om **Virtual Address Space (VAS)**, som är ett abstraktionslager mellan en applikation och fysisk minne. VAS låter operativsystemet hantera minnesallokering utan att applikationer behöver direktåtkomst till fysiskt RAM.

VAS-storlek:

- **Före Windows Server 2012**: VAS var begränsat till **8 TB**.
- **Från Windows Server 2012 och senare**: Ökades till **128 TB** för bättre skalbarhet.
- **Teoretisk gräns**: **16 exabyte**, men operativsystemet sätter praktiska begränsningar.

VAS är avgörande för minneshantering i SQL Server och andra applikationer, särskilt för **stora databasservrar** med mycket RAM.

Virtual address space
https://en.wikipedia.org/wiki/Virtual_address_space

Lesson: SQL Server memory

- SQL Server memory configuration
- SQL Server memory models
- SQL Server memory architecture

SQL Server memory configuration

- Min Server Memory:
  - Memory level that, once reached, will not be released
- Max Server Memory:
  - Maximum amount of memory that SQL Server will request
  - Various formulas exists, for how much to leave to OS in dedicated machine:
    - Leave 1 GB for every 4 GB memory until 16 GB. And 1 GB for every 8 GB after that.
    - Leave 4 GB or 10% of the memory in the machine, whichever is the largest.
- Signs of ”not enough memory”
  - Significant waits for
    - PAGEIOLATCH (reading data from disk)
    - RESOURCE_SEMAPHORE (memory grants for sorts, hash, …)
    - RESOURCE_SEMAPHORE_QUERY_COMPILE (producing execution plans)

PAGEIOLATCH
Detta väntetillstånd inträffar när en SQL Server-process väntar på att en data- eller indexsida ska läsas in från disk till bufferpoolen, som är SQL Servers interna cache. PAGEIOLATCH indikerar vanligtvis att SQL Server begär en sida som inte finns i minnet och måste vänta på att I/O-operationen ska slutföras. Om du ser mycket höga väntetider för PAGEIOLATCH, kan det tyda på flaskhalsar i I/O, som långsamma diskar eller att du inte har tillräckligt med minne och därför måste läsa från disken oftare än önskat.

RESOURCE_SEMAPHORE
Detta väntetillstånd uppstår när en tråd väntar på att få ett minnesbidrag för att utföra operationer som sorterar eller bygger hash-tabeller, vilket ofta är en del av komplexa frågeprocesser som JOINs eller aggregeringar. Om tråden måste vänta för länge på att minnesbidraget ska beviljas, kan det tyda på minnestryck på servern. Detta kan vara ett tecken på att servern inte har tillräckligt med tillgängligt minne eller att frågor begär mer minne än vad som finns tillgängligt, vilket kan leda till prestandaproblem.

RESOURCE_SEMAPHORE_QUERY_COMPILE
Detta väntetillstånd innebär att en tråd väntar på tillstånd för att kompilera eller återkompilera en exekveringsplan för en fråga. Kompilering av exekveringsplaner är en resursintensiv operation, och SQL Server begränsar antalet samtidiga kompileringar för att förhindra att systemet blir överbelastat. Om en process hamnar i RESOURCE_SEMAPHORE_QUERY_COMPILE väntetillstånd för ofta eller för länge, kan det vara ett tecken på att servern upplever ett högt tryck på kompilering, vilket kan vara på grund av en stor mängd komplexa frågor som behöver kompileras ofta.

Memory Management Architecture Guide
https://docs.microsoft.com/en-us/sql/relational-databases/memory-management-architecture-guide

sp_configure (Transact-SQL)
https://docs.microsoft.com/en-us/sql/relational-databases/system-stored-procedures/sp-configure-transact-sql

Server memory configuration options
https://docs.microsoft.com/en-us/sql/database-engine/configure-windows/server-memory-server-configuration-options

Does My SQL Server Need More Memory?
https://www.erikdarlingdata.com/sql-server/does-my-sql-server-need-more-memory/

Setting max server memory (includes link to Jonathan Kehayias' blog post)
https://sqlblog.karaszi.com/setting-max-server-memory/

Memory Dangerously Low or Max Memory Too High
https://www.brentozar.com/blitz/max-memory/

SQL Server Memory Models

	Conventional	Locked	Large
OS Page Size	4 kB	4 kB	2 MB
Static/Dynamic	Dynamic	Dynamic	Static Memory is committed at startup Rarely used
Majority of SQL Server memory subject to OS swapping	Yes	No	No
Lock Pages In Memory (LPIM) for service account	Not set	Set	Required
Prerequisites			Enterprise Edition Trace Flag 834 Trace flag 874 if col-store ix are used

Demo Locked pages

I SQL Server refererar "Locked Pages in Memory" till en funktion som tillåter SQL Server att låsa sidor i fysiskt minne, vilket förhindrar att dessa sidor pagineras ut till disk av operativsystemet. När sidor pagineras ut kan det orsaka prestandaproblem eftersom det tar längre tid att återhämta sidorna från disk jämfört med att direkt komma åt dem i RAM.

När SQL Server-konfigurationen är inställd för att tillåta låsta sidor i minnet, och SQL Server har rätt behörigheter via Windows-användargruppen "Lock Pages in Memory", kommer den att försöka reservera minne som inte kan pagineras. Detta är speciellt användbart för databasservrar med hög arbetsbelastning, där konsekvent snabb åtkomst till data är kritisk.

Fördelarna med att låsa sidor i minnet innefattar:

1. **Förutsägbar Prestanda: Det minimerar risken för att systemet påverkas av prestandafluktuationer på grund av paginering.**

2. **Förhindrar Minnespress: Det förhindrar att operativsystemet tar minne från SQL Server för att använda det till andra processer, vilket kan vara viktigt på system som körs på gränsen för sina minnesresurser.**

3. **Bättre Prestanda under Minnesbelastning: Om servern har tillräckligt med fysiskt minne, kan användningen av låsta sidor förbättra systemets prestanda genom att undvika disk-I/O som är associerad med att paginera data till och från swap-filen.**

Men det finns också potentiella risker:

1. **Systemstabilitet: Om för mycket minne låses kan det påverka andra processer som körs på samma server, inklusive operativsystemets kärnprocesser, vilket kan leda till instabilitet.**

2. **Felkonfiguration: Om funktionen används på en server som inte har tillräckligt med fysiskt minne kan det faktiskt leda till prestandaproblem.**

Därför rekommenderas det att endast använda "Locked Pages in Memory" när du har noggrant övervägt dina minneskrav och prestandamål och endast på servrar som är dedikerade till SQL Server. Dessutom bör man övervaka systemets beteende noggrant efter att ha aktiverat denna funktion.

För att konfigurera SQL Server för att använda "Locked Pages in Memory", måste du utföra följande steg:

1. **Tilldela behörigheten "Lock Pages in Memory":**
   - Öppna "Local Group Policy Editor" genom att köra `gpedit.msc`.
   - Navigera till "Computer Configuration" > "Windows Settings" > "Security Settings" > "Local Policies" > "User Rights Assignment".
   - Hitta policyinställningen "Lock pages in memory".
   - Dubbelklicka på inställningen och klicka på "Add User or Group".
   - Lägg till SQL Server-tjänstens konto (t.ex., NT SERVICE\MSSQLSERVER för standardinstansen) och klicka på "OK".

2. **Konfigurera SQL Server:**
   Det finns inga specifika inställningar inom SQL Server som måste ändras för att aktivera användningen av "Locked Pages in Memory". När behörigheten är tilldelad kommer SQL Server att automatiskt använda denna funktion om den är tillgänglig och om SQL Server-tjänsten startas om efter att behörigheten har tilldelats.

3. **Starta om SQL Server-tjänsten:**
   - Gå till "Services" genom att köra `services.msc`.
   - Hitta SQL Server-tjänsten (t.ex., SQL Server (MSSQLSERVER) för en standardinstans).
   - Högerklicka på tjänsten och välj "Restart".

Kom ihåg att endast användarkonton eller tjänster som kräver denna behörighet bör få den, och den bör hanteras noggrant på grund av dess potentiella påverkan på systemets stabilitet.

Det är också viktigt att notera att vissa SQL Server-versioner (särskilt SQL Server Standard Edition från och med version 2012) kanske inte kräver manuell konfiguration för "Locked Pages in Memory" eftersom de automatiskt använder denna funktion under vissa omständigheter. Dessutom, om du använder SQL Server på en Azure Virtual Machine eller en annan molnbaserad tjänst, kanske inställningen inte är tillgänglig eller lämplig att använda i den miljön.

Slutligen, efter att du har aktiverat "Locked Pages in Memory", är det klokt att övervaka serverns prestanda och beteende för att säkerställa att den inte negativt påverkar andra applikationer eller systemets övergripande stabilitet.

"Large Pages" i SQL Server är en funktion som används för att förbättra prestanda och minneshantering, särskilt när du arbetar med stora mängder minne. Den här funktionen nyttjar större minnesblock (sidor) istället för standardstorleken på 4 KB, vilket leder till mer effektiv minnesallokering och mindre overhead för att hantera många små sidor.

Viktiga fördelar med Large Pages:

1. **Förbättrad prestanda:** Genom att använda större minnesblock minskas behovet av TLB (Translation Lookaside Buffer)-inträden, vilket är en cache i processorn som översätter virtuella minnesadresser till fysiska. Med färre, större sidor kan SQL Server hantera dessa adresseringar snabbare och effektivare, vilket leder till ökad prestanda.

2. **Minskad fragmentering:** Vid hantering av många små sidor kan fragmentering uppstå, där minnet blir ineffektivt fördelat. Genom att använda stora sidor minskar SQL Server risken för fragmentering, vilket leder till mer effektivt utnyttjande av tillgängligt minne.

3. **Mindre CPU-overhead:** Eftersom färre minnesallokeringar och -hanteringsoperationer krävs för stora sidor, kan SQL Server minska den mängd CPU-resurser som går åt till att hantera minnesoperationer. Detta frigör CPU-kraft till andra viktiga operationer och förbättrar övergripande prestanda.

4. **Optimal för stora arbetsbelastningar:** Funktionen är särskilt användbar för system med mycket RAM och höga arbetsbelastningar, såsom stora OLTP-system, datalager eller applikationer som kräver intensiv minnesanvändning.

Hur large pages används:
SQL Server aktiverar inte Large Pages som standard; det krävs specifika konfigurationer och förutsättningar, såsom att använda **Lock Pages in Memory**

(LPIM)-rättigheter och ha tillräckligt med fysiskt minne. Det används oftast i scenarier med SQL Servers Buffer Pool Extension eller när du kör stora instanser som kräver stabil och snabb minneshantering.

Large Pages är ett kraftfullt verktyg för att optimera minneshanteringen och förbättra prestandan i system med stora minneskrav och höga arbetsbelastningar.

SQL Server memory models
https://techcommunity.microsoft.com/t5/core-infrastructure-and-security/sql-server-memory-models-part-i/ba-p/370321

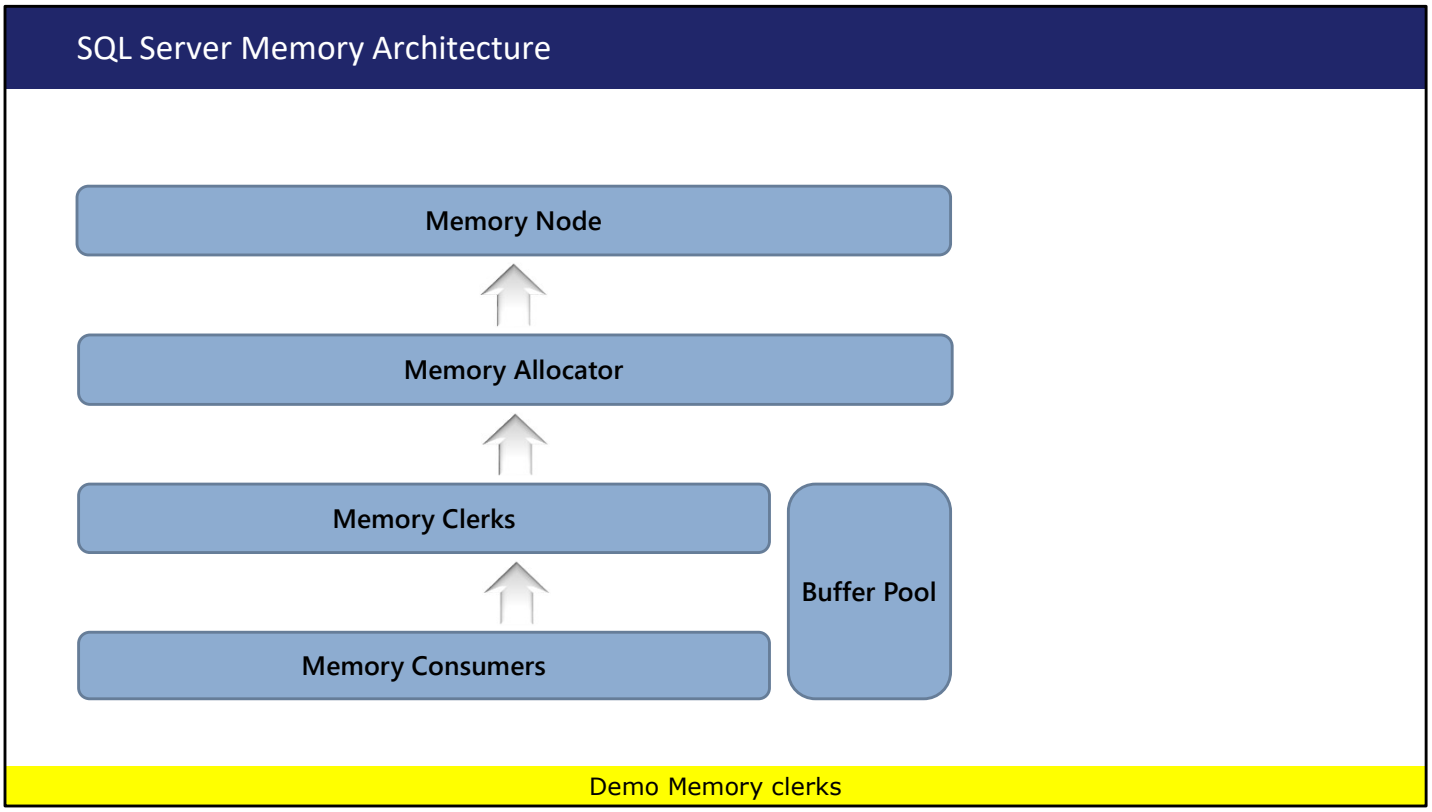
Enable the Lock Pages in Memory Option (Windows)
https://docs.microsoft.com/en-us/sql/database-engine/configure-windows/enable-the-lock-pages-in-memory-option-windows

Configuration: Lock Pages in Memory (LPIM)
https://www.brentozar.com/training/fundamentals-database-administration/lock-pages-memory-lpim-6m/

SQL Server and Large Pages Explained….
https://docs.microsoft.com/en-us/archive/blogs/psssql/sql-server-and-large-pages-explained

Memory video with Bob Ward (1 hour 17 minutes)
https://www.youtube.com/watch?v=CRAx73LiXTc

Log Page Life Expectancy over time
https://karaszi.com/log-page-life-expectancy-over-time



Monitoring Memory Clerk and Buffer Pool Allocations in SQL Server
https://www.sqlshack.com/monitoring-memory-clerk-and-buffer-pool-allocations-in-sql-server/

sys.dm_os_memory_clerks (Transact-SQL)
https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-os-memory-clerks-transact-sql

I SQL Server hanterar memory clerks internminnets allokering och användning, fungerande som bokhållare för minnesanvändning inom olika delar av servern. Varje memory clerk är specialiserad på ett visst område, som buffer poolen, execution plan caching eller CLR-integration, och övervakar minneskonsumtionen för att säkerställa optimal prestanda och effektivitet. Denna modulära hantering möjliggör detaljerad insikt och kontroll över minnesanvändningen, vilket hjälper i diagnostik och prestandaoptimering. Genom att isolera minnesförbrukningen per funktion kan SQL Server mer effektivt hantera begränsade minnesresurser och förbättra den totala databashanteringsupplevelsen.

En "memory allocator" i SQL Server är ett system som ansvarar för att tilldela och frigöra minne för serverns olika processer och operationer. Den ser till att minnet distribueras effektivt mellan olika uppgifter och hjälper till att förhindra minnesläckor genom att återanvända och frigöra minnesutrymmen som inte längre behövs. En "memory node" å andra sidan, refererar till en del av serverns minnesarkitektur som organiserar och hanterar minne på en mer granulär nivå, ofta i samband med NUMA (Non-Uniform Memory Access) arkitekturer. Detta möjliggör en mer effektiv minneshantering och tillgång baserat på serverns fysiska och logiska minneslayout, vilket optimerar prestanda för databasoperationer.

Memory grants

- Some operators in execution plan need explicit memory grants before the query starts executing
  - Sort
  - Hash
- SQL Server estimates how much memory will be needed
  - The operator will spill to disk in tempdb if it underestimated
  - You waste memory if it overestimated
- Wait stats when waiting for grant: RESOURCE_SEMAPHORE
- Query hints:
  - MAX_GRANT_PERCENT
  - MIN_GRANT_PERCENT

Den här delen av sliden handlar om hur SQL Server hanterar **memory grants** och väntetider kopplade till dem.

RESOURCE_SEMAPHORE (Wait Stats)

•**RESOURCE_SEMAPHORE** är en **wait type** som uppstår när en fråga väntar på att få allokerat minne innan den kan börja exekvera.

•När många frågor konkurrerar om minnesresurser kan vissa fördröjas, vilket kan orsaka **prestandaproblem**.

•Detta är särskilt vanligt för frågor som använder **Sort** och **Hash Join**, som behöver **explicita memory grants**.

Query Hints för att styra memory grants

•**MAX_GRANT_PERCENT** – Begränsar mängden minne som en fråga får använda, uttryckt i procent av **serverns totala minne**.

- Exempel: OPTION (MAX_GRANT_PERCENT = 25) Begränsar frågan till högst **25 % av tillgängligt minne**.

•**MIN_GRANT_PERCENT** – Ställer in en **minimigräns** för memory grants.

- Exempel: OPTION (MIN_GRANT_PERCENT = 5) Kräver att frågan får **minst 5 %** av tillgängligt minne, annars kan den behöva vänta.

Dessa inställningar används för att **balansera minnesanvändning** och **förhindra att stora frågor monopoliserar serverns resurser**. 🚀

```
SELECT * FROM person.Person
ORDER BY 1,2,3,4
OPTION (MAX_GRANT_PERCENT = 25)
```

Understanding SQL server memory grant
https://techcommunity.microsoft.com/t5/sql-server-blog/understanding-sql-server-memory-grant/ba-p/383595

Memory Grants: The mysterious SQL Server memory consumer with Many Names
https://techcommunity.microsoft.com/t5/sql-server-support-blog/memory-grants-the-mysterious-sql-server-memory-consumer-with/ba-p/333994

Starting SQL: Query Memory Grants In SQL Server Execution Plans
https://www.erikdarlingdata.com/starting-sql-memory-grants-in-execution-plans/

SQL Server Community Tools: How To Use sp_WhoIsActive To Get Memory Grant Information
https://www.erikdarlingdata.com/sql-server-community-tools-how-to-use-sp_whoisactive-to-get-memory-grant-information/

Troubleshooting SQL Server With High Memory Grants using free monitoring tool
https://ajaydwivedi.com/performance-tuning/troubleshooting-sql-server-with-high-memory-grants/

Lesson: In-Memory OLTP

- What Is In-Memory OLTP?
- Memory-Optimized tables
- Natively Compiled stored procedures

What Is In-Memory OLTP?

- In-Memory OLTP tables:
- Transactional tables stored in-memory
- Access same as disk-based tables
- Fully transactional
- Durable
- Very low latency

In-Memory OLTP overview and usage scenarios
https://docs.microsoft.com/en-us/sql/relational-databases/in-memory-oltp/overview-and-usage-scenarios

Plan your adoption of In-Memory OLTP Features in SQL Server
https://docs.microsoft.com/en-us/sql/relational-databases/in-memory-oltp/plan-your-adoption-of-in-memory-oltp-features-in-sql-server

Transact-SQL Constructs Not Supported by In-Memory OLTP
https://docs.microsoft.com/en-us/sql/relational-databases/in-memory-oltp/transact-sql-constructs-not-supported-by-in-memory-oltp

Bucket är antalet hinkar eller platser där datana kan sorteras. sikta på att ha bucket count lika med eller större än det högsta antalet unika värden du förväntar dig i den kolumn som indexet baseras på.

Memory-Optimized tables

- Durable Memory-Optimized Tables:
  - ACID compliant
  - Persisted to disk
- Non-Durable Memory-Optimized Tables:
  - NOT ACID compliant
  - Not persisted to disk (server crash = data loss)
  - No disk IO

Table and Row Size in Memory-Optimized Tables
https://docs.microsoft.com/en-us/sql/relational-databases/in-memory-oltp/table-and-row-size-in-memory-optimized-tables

Natively Compiled Stored Procedures

- Compiled at create time
- Improved performance
- Support for atomic blocks and NOT NULL parameter constraints

Demo Memory optimized tables

Native Compilation Advisor
https://docs.microsoft.com/en-us/sql/relational-databases/in-memory-oltp/native-compilation-advisor

Supported Features for Natively Compiled T-SQL Modules
https://docs.microsoft.com/en-us/sql/relational-databases/in-memory-oltp/supported-features-for-natively-compiled-t-sql-modules

Creating Natively Compiled Stored Procedures
https://docs.microsoft.com/en-us/sql/relational-databases/in-memory-oltp/creating-natively-compiled-stored-procedures

V1

- No lab for this module

Lab 4: Memory usage

- Ex 1: Set max server memory
- Ex 2: Work with query that has "memory issues"

Estimated Time: 30 minutes