Documentation:	**Performance analysis report for sql-kekahr-prod-centralin server**
Author:	**Sabyasachi Choudhury**
Version:	**V 1.0**
Date of documentation:	**12 June 2025**

About Author of the report

Sabyasachi Choudhury, Database Administrator who has 3+ years of experience in Microsoft data platform. Specialist in SQL Server administration and Performance tuning. He has certifications as mentioned below

- Microsoft Certified Azure Database Administrator

Analysis Report

Azure SQL Database analysis was performed by DB team to identify whether SQL instances are configured as per the industry standards and follow the SQL best practices. Analysis took place at Bangalore office. The output of this analysis report would give you complete insight into current SQL Server instance configuration. The report also provides recommendations to achieve better performance as per Microsoft standard and current workload.

Technical Environment

Property	Value
Environment	Azure SQL Database (PaaS)
Server Name	sql-kekahr-prod-centralin.database.windows.net
Database Name	sqldb-kekahr
Service Tier	Hyperscale
Compute Size	18 vCores
Physical Memory in MB	933836
Deployment Model	Single Database
Azure Region	Central India
SQL Compatibility Level	SQL Server 2016(130)
Collation	SQL_Latin1_General_CP1_CI_AS
High Availability Replica	1 HA Replica
Zone Redundency	Enabled
Backup Retention	PITR Retention 35 Days
Automatic Tuning	**Force Last Good Plan:** ON **Create Index:** ON **Drop Index:** OFF
Query Store	Enabled
Auditing	Enabled (Database level)

Workload Characteristics

OLTP Workload: Day-to-day transactional operations with low latency
Reporting Workload: Complex queries, aggregations, dashboard generation
Batch Jobs: Nightly ETL, data processing, cleanup routines

1. SQL Server Configuration

Note: No Downtime Required for these changes.

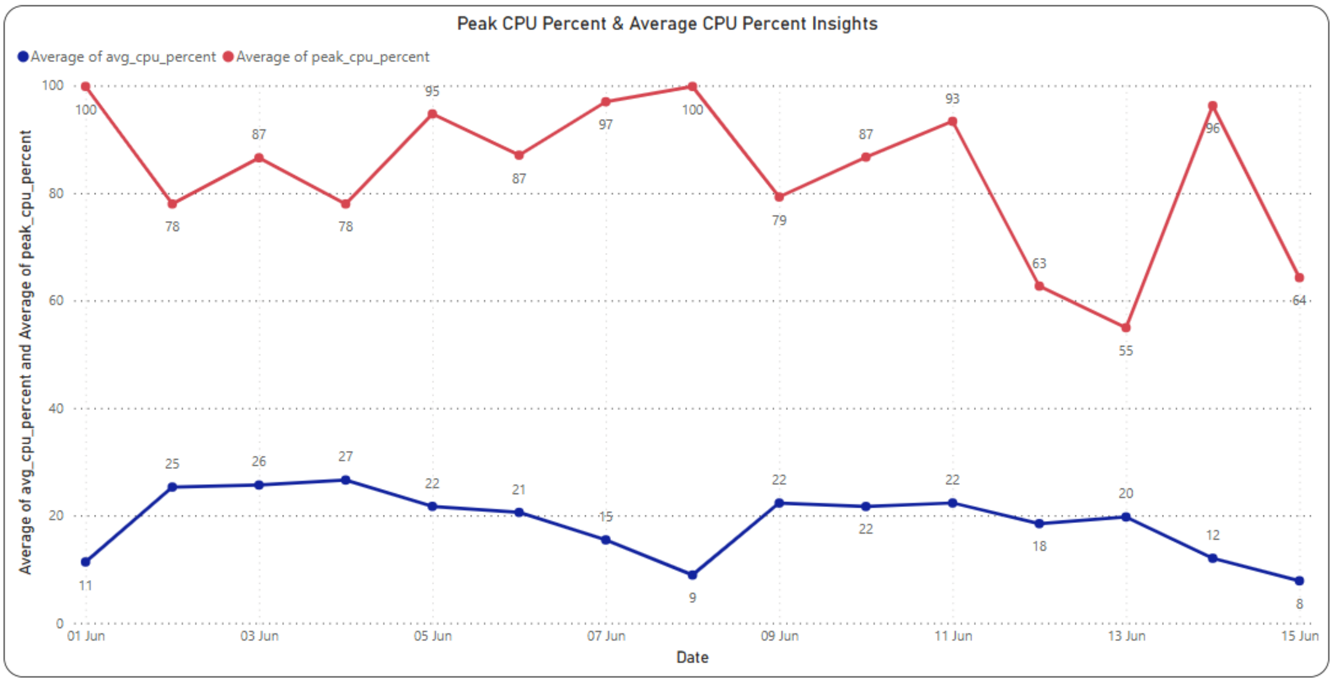
Configuration Name	Current Value	Description	Recommendation
Fill Factor	80 & 0	Fill Factor determines how full SQL Server will make each data page during index creation or rebuild.	Set to 95 to balance read and write performance in a mixed OLTP + Reporting environment.
ASYNC_STATS_UPDATE_WAIT_AT_LOW_PRIORITY	OFF	Controls whether asynchronous statistics updates wait at a low priority to reduce contention with other operations.	Keep OFF (unless Async Stats update is in used)
ELEVATE_ONLINE	OFF	With this OFF, index rebuilds or other maintenance operations may take the table offline, causing downtime and impacting performance during maintenance windows. For a production database, setting this to ON can minimize user impact during maintenance, especially since automatic tuning (index creation/drop) is enabled.	Change to ON
ELEVATE_RESUMABLE	OFF	When set to ON, automatically elevates certain operations to resumable mode, allowing them to be paused and resumed (e.g., during index rebuilds).	Change to ON
EXEC_QUERY_STATS_FOR_SCALAR_FUNCTIONS	OFF	When set to ON, collects execution statistics for scalar user-defined functions (UDFs) to help identify performance bottlenecks.	Change to ON
FORCE_SHOWPLAN_RUNTIME_PARAMETER_COLLECTION	OFF	When set to ON, forces the collection of runtime parameters in showplan (execution plan) data, aiding in query performance analysis.	Change to ON
LAST_QUERY_PLAN_STATS	OFF	When set to ON, collects statistics for the last query plan, providing detailed runtime stats for troubleshooting.	Change to ON
MAXDOP	8	Sets the maximum degree of parallelism for query execution.	Keep DB-level at 4; tune query-level
OPTIMIZE_FOR_AD_HOC_WORKLOADS	OFF	When set to ON, optimizes the plan cache for ad hoc queries by storing only a small stub initially, reducing memory usage.	Change to ON
QUERY_OPTIMIZER_HOTFIXES	OFF	When set to ON, applies query optimizer hotfixes for the current compatibility level.	Change to ON
XTP_PROCEDURE_EXECUTION_STATISTICS	OFF	Collects execution statistics for natively compiled T-SQL modules (used with In-Memory OLTP).	Keep OFF (unless In Memory OLTP Table used)
XTP_QUERY_EXECUTION_STATISTICS	OFF	Collects query execution statistics for natively compiled T-SQL modules.	Keep OFF (unless In Memory OLTP Table used)

2. Performance Metrics Analysis (Last 30 Days)

2.1 Resource Utilization

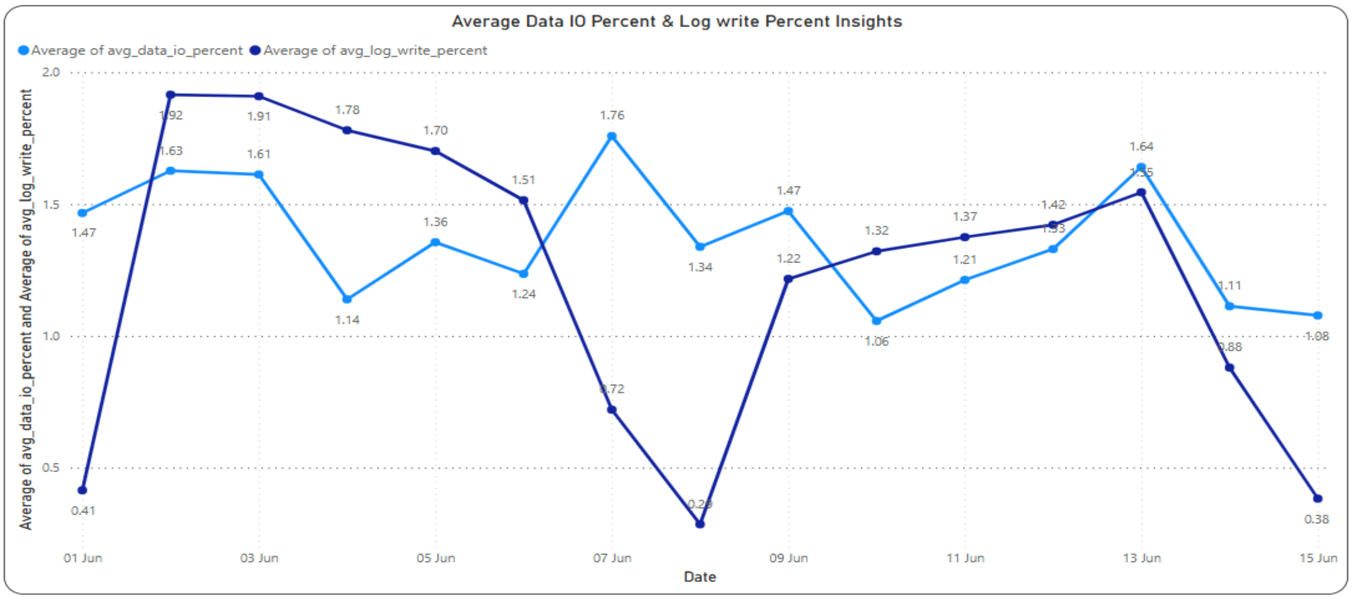
CPU Utilization:

The **average CPU utilization** over the period fluctuates between approximately 7.8% and 26.6%, with several days (e.g., June 3rd, 4th, 11th) showing sustained averages above 20%. This indicates moderate CPU usage overall but with some spikes that could impact performance during peak workload periods. The **peak CPU percentages** are quite high, reaching 99.8% on multiple days (June 1st, 7th, 8th, 14th), suggesting short bursts of very high CPU demand. These spikes may correspond to resource-intensive queries and could cause transient performance degradation or throttling if sustained.



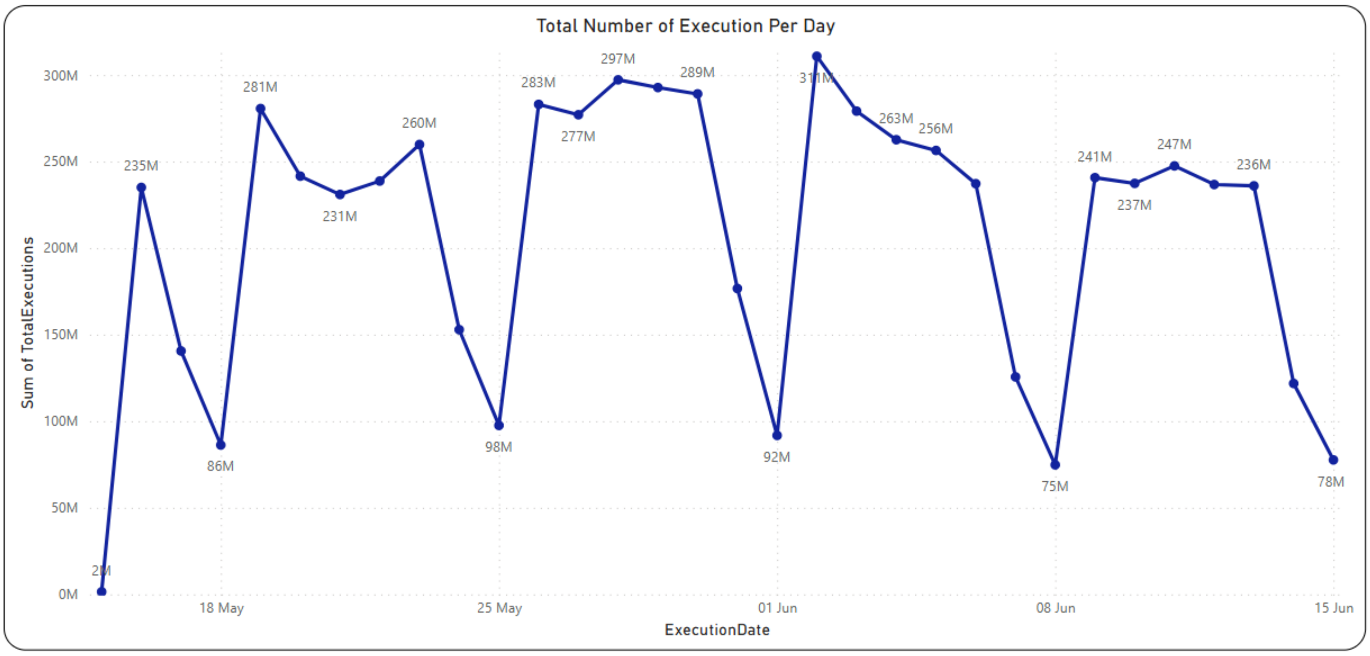
Data IO and Log Write Percentage:

The average Data IO percentage remains low and stable, generally between 1.0% and 1.7%, which suggests that data read/write operations are not a major bottleneck. Overall, IO subsystems appear healthy and unlikely to be the primary cause of performance issues.



Total Number of Execution per day:

Between May 15 and May 25, query executions started low at around 1.5 million and sharply increased to over 259 million by May 23, before dropping to about 97 million on May 25, indicating a rapid rise in workload followed by a brief reduction. From May 26 to June 10, executions remained consistently high, fluctuating between 237 million and 297 million daily, reflecting sustained heavy usage likely due to ongoing business processes or user activity. However, from June 11 to June 16, there was a steep decline in executions from approximately 247 million down to 9.5 million.

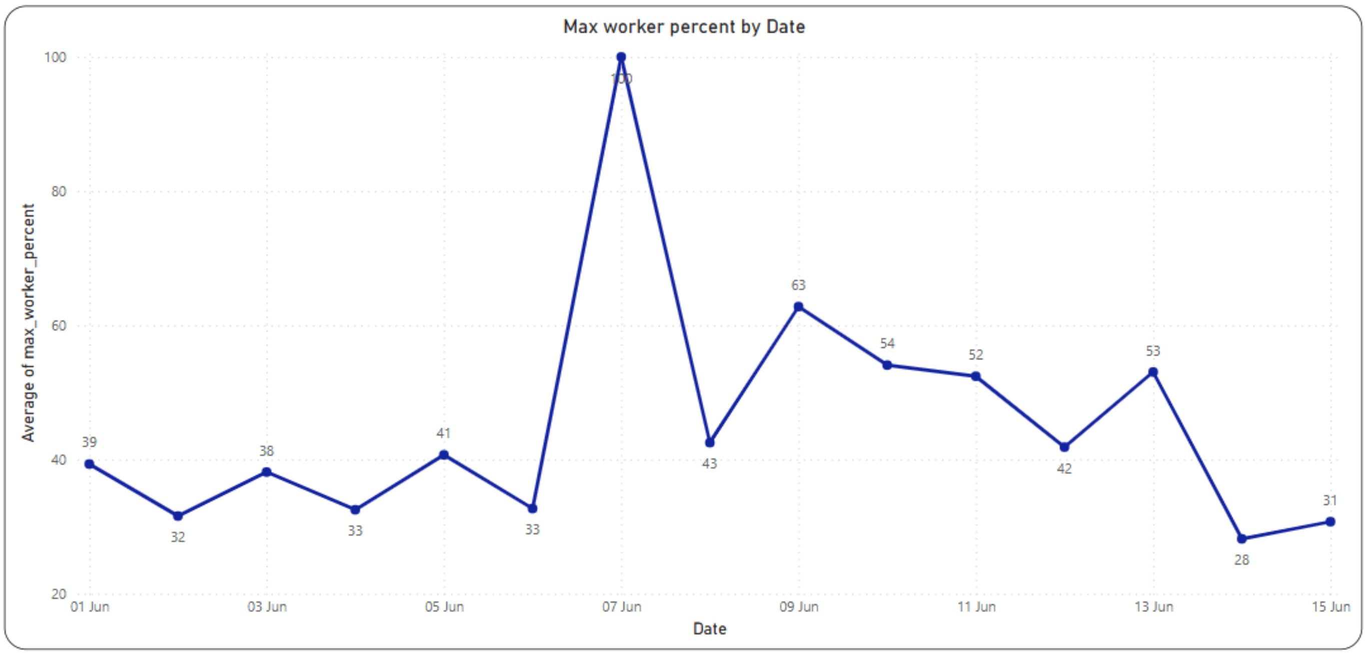


ExecutionDate	TotalExecutions
16-06-2025	9519476
15-06-2025	77673742
14-06-2025	121836223
13-06-2025	235951418
12-06-2025	236720202
11-06-2025	247462736
10-06-2025	237388897

09-06-2025	240687641
08-06-2025	74833487
07-06-2025	125616741
06-06-2025	237210362
05-06-2025	256413585
04-06-2025	262590582
03-06-2025	279120554
02-06-2025	310798962
01-06-2025	91826185
31-05-2025	176679213
30-05-2025	289055948
29-05-2025	292730937
28-05-2025	297197181
27-05-2025	277030191
26-05-2025	282998092
25-05-2025	97590155
24-05-2025	152889456
23-05-2025	259853070
22-05-2025	238738399
21-05-2025	230960820
20-05-2025	241545385
19-05-2025	280585981
18-05-2025	86283256
17-05-2025	140585053
16-05-2025	235045433
15-05-2025	1510179

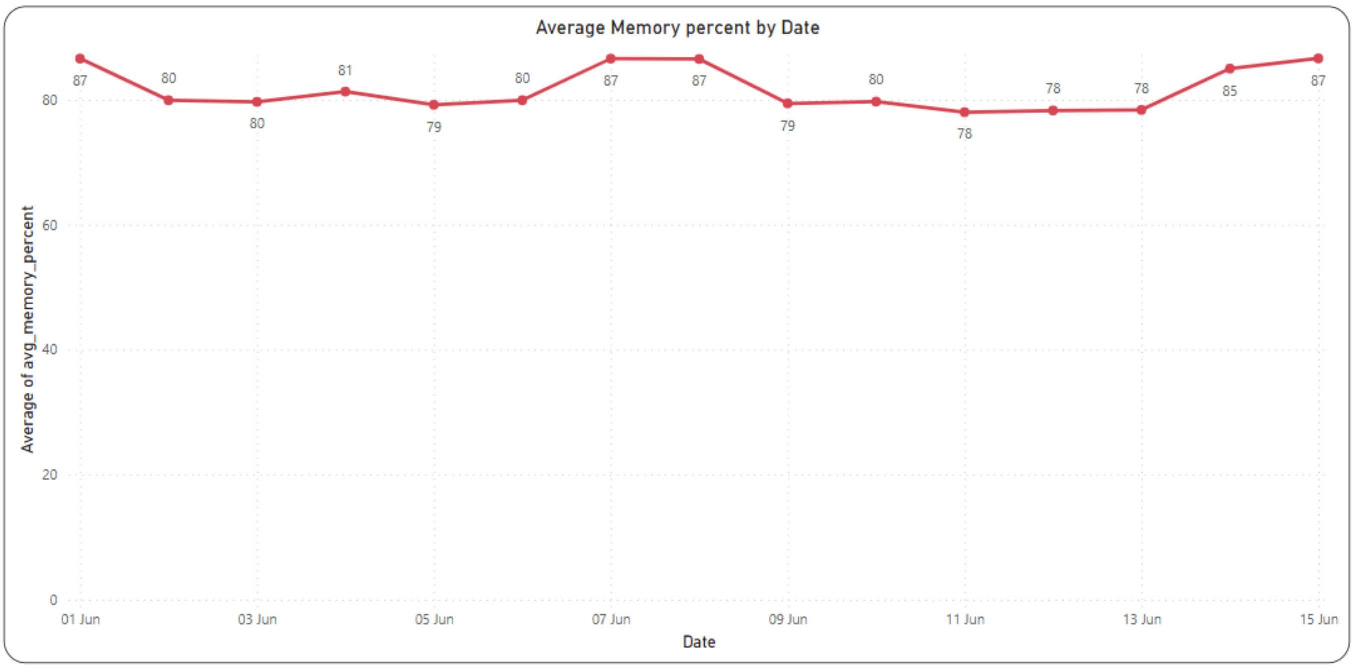
Max Worker Percent

The max_worker_percent metric, representing the maximum percentage of worker threads utilized, ranges from about 30% to 100%, with a notable spike to 100% on June 7th. Worker threads are essential for query execution concurrency; a spike to 100% indicates that the system reached its maximum concurrency limit on that day, which can cause query queuing and increased latency. Other days show moderate to high worker usage (30-60%), which may be acceptable depending on workload but warrants attention to concurrency and query parallelism.



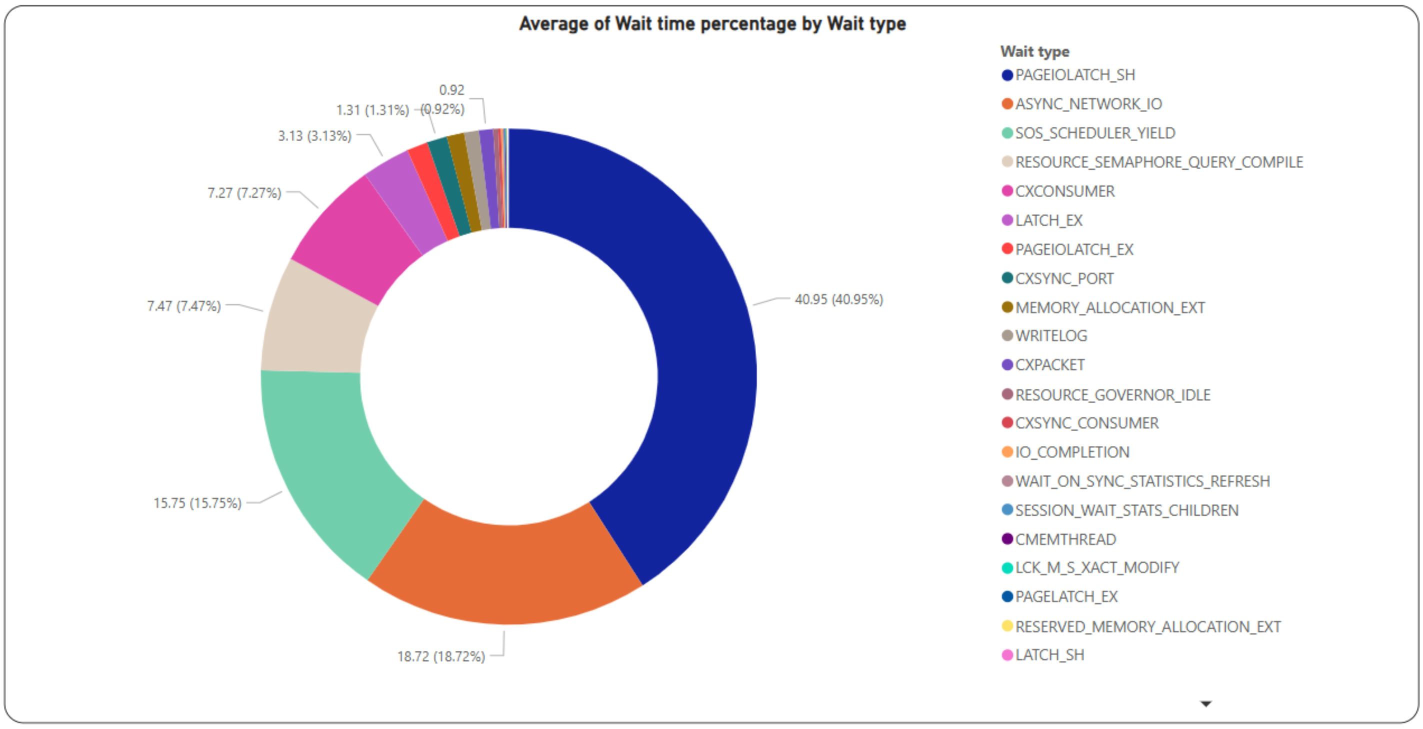
Memory Utilization

Memory usage averages between approximately 78% and 87%, with some fluctuations but generally staying in the high 70s to mid-80s percentile range. This indicates that the database utilizes a significant portion of allocated memory, which is typical for workloads with active caching and buffer pool usage. While not critically high, sustained memory usage near 80-85% suggests monitoring for potential memory pressure, especially if coupled with CPU spikes or query slowdowns.



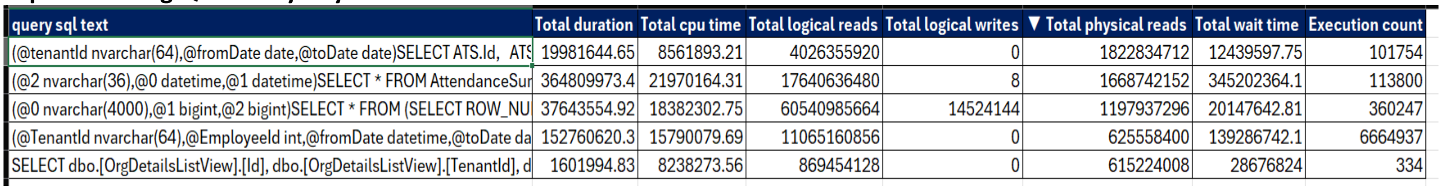
2.2 Wait Statistics and Bottlenecks:

The analysis of average wait time percentage by wait type reveals that the most significant contributor to overall wait times is PAGEIOLATCH_SH, accounting for approximately 41% of total wait time. This type of wait typically indicates that queries are spending considerable time waiting for data pages to be read from disk into memory, which is often symptomatic of storage subsystem latency or insufficient memory to cache frequently accessed data. The next largest contributors are ASYNC_NETWORK_IO (18.7%) and SOS_SCHEDULER_YIELD (15.8%), suggesting that some queries are experiencing delays due to slow network responses (potentially from the application layer) and CPU scheduling contention, respectively. Other notable wait types include RESOURCE_SEMAPHORE_QUERY_COMPILE (7.5%), which can point to resource contention during query compilation, and CXCONSUMER (7.3%), often associated with parallel query execution. The presence and proportion of these waits indicate that the current performance issues are likely multifactorial, with a primary bottleneck at the storage layer, compounded by network, CPU, and query compilation resource contention. Addressing these areas—particularly optimizing IO performance, reviewing memory allocation, and tuning high-impact queries—should be prioritized to alleviate the observed performance degradation.



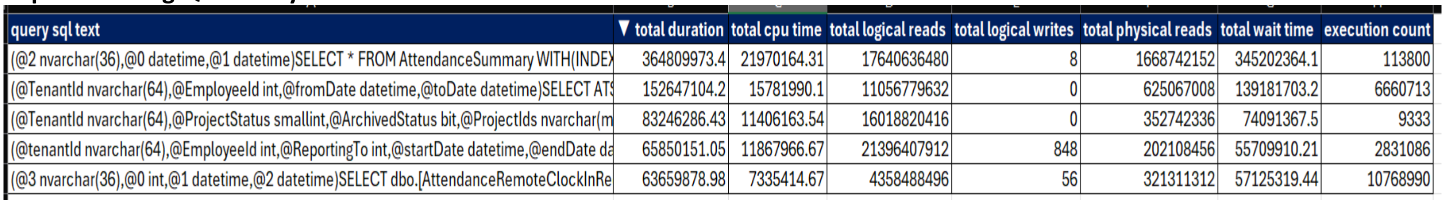
2.3 Top Queries and Wait Types

Top Consuming Queries by Physical Reads:



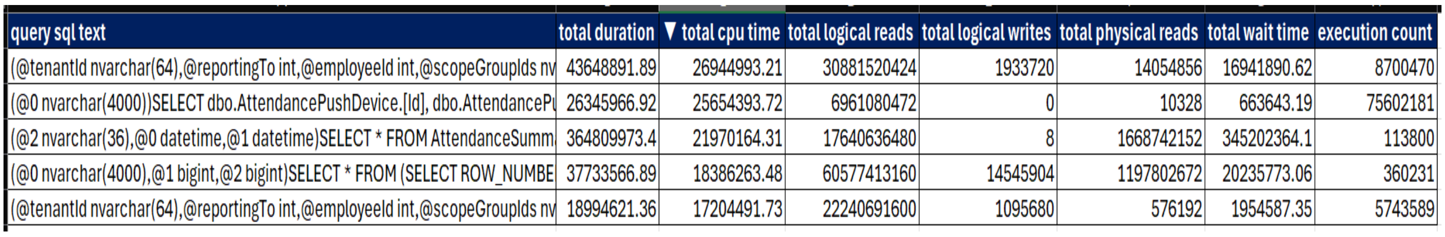
query sql text	Total duration	Total cpu time	Total logical reads	Total logical writes	▼ Total physical reads	Total wait time	Execution count
(@tenantId nvarchar(64),@fromDate date,@toDate date)SELECT ATS.Id, ATS	19981644.65	8561893.21	4026355920	0	1822834712	12439597.75	101754
(@2 nvarchar(36),@0 datetime,@1 datetime)SELECT * FROM AttendanceSur	364809973.4	21970164.31	17640636480	8	1668742152	345202364.1	113800
(@0 nvarchar(4000),@1 bigint,@2 bigint)SELECT * FROM (SELECT ROW_NU	37643554.92	18382302.75	60540985664	14524144	1197937296	20147642.81	360247
(@TenantId nvarchar(64),@EmployeeId int,@fromDate datetime,@toDate da	152760620.3	15790079.69	11065160856	0	625558400	139286742.1	6664937
SELECT dbo.[OrgDetailsListView].[Id], dbo.[OrgDetailsListView].[TenantId], d	1601994.83	8238273.56	869454128	0	615224008	28676824	334

Top Consuming Queries by Duration:



query sql text	▼ total duration	total cpu time	total logical reads	total logical writes	total physical reads	total wait time	execution count
(@2 nvarchar(36),@0 datetime,@1 datetime)SELECT * FROM AttendanceSummary WITH(INDEX	364809973.4	21970164.31	17640636480	8	1668742152	345202364.1	113800
(@TenantId nvarchar(64),@EmployeeId int,@fromDate datetime,@toDate datetime)SELECT ATS	152647104.2	15781990.1	11056779632	0	625067008	139181703.2	6660713
(@TenantId nvarchar(64),@ProjectStatus smallint,@ArchivedStatus bit,@ProjectIds nvarchar(m	83246286.43	11406163.54	16018820416	0	352742336	74091367.5	9333
(@tenantId nvarchar(64),@EmployeeId int,@ReportingTo int,@startDate datetime,@endDate da	65850151.05	11867966.67	21396407912	848	202108456	55709910.21	2831086
(@3 nvarchar(36),@0 int,@1 datetime,@2 datetime)SELECT dbo.[AttendanceRemoteClockInRe	63659878.98	7335414.67	4358488496	56	321311312	57125319.44	10768990

Top Consuming Queries by CPU Time:



query sql text	total duration	▼ total cpu time	total logical reads	total logical writes	total physical reads	total wait time	execution count
(@tenantId nvarchar(64),@reportingTo int,@employeeId int,@scopeGroupIds nv	43648891.89	26944993.21	30881520424	1933720	14054856	16941890.62	8700470
(@0 nvarchar(4000))SELECT dbo.AttendancePushDevice.[Id], dbo.AttendancePu	26345966.92	25654393.72	6961080472	0	10328	663643.19	75602181
(@2 nvarchar(36),@0 datetime,@1 datetime)SELECT * FROM AttendanceSumm	364809973.4	21970164.31	17640636480	8	1668742152	345202364.1	113800
(@0 nvarchar(4000),@1 bigint,@2 bigint)SELECT * FROM (SELECT ROW_NUMBE	37733566.89	18386263.48	60577413160	14545904	1197802672	20235773.06	360231
(@tenantId nvarchar(64),@reportingTo int,@employeeId int,@scopeGroupIds nv	18994621.36	17204491.73	22240691600	1095680	576192	1954587.35	5743589

Key Observations

- **High Execution Count & High Duration:** The high execution count with high total duration suggests both frequent and slow queries, which is a major concern.
- **High Physical Reads:** All top queries have extremely high physical reads, explaining the dominance of PAGEIOLATCH_SH waits.
- **High Wait Time:** Wait times are very high for these queries, especially for those with high execution counts.

Recommendations

- Rewrite queries to be more selective, avoid SELECT *, and fetch only necessary columns/rows.
- If possible, increase memory allocation to SQL Server to allow more data to be cached, reducing disk I/O.
- Check if the application is processing data too slowly or fetching large result sets unnecessarily.
- Consider adjusting MAXDOP at the query level for problematic queries

3. Tables Without Clustered Index:

Below tables are highly fragmented. This situation happens because of missing cluster index and whole table relay on heap. The solution is to rebuild the tables. This cause high forward record reference results in high usage of IO and degrades the overall performance of database. We are recommending to create clustered index for these tables

DB Name: sqldb-kekahr

name	Type desc	Table Type	Create date
AppConsent	HEAP	USER_TABLE	15-11-2024 22:40
FileImport	HEAP	USER_TABLE	18-02-2019 16:42
NatureOfBusiness	HEAP	USER_TABLE	26-07-2019 18:59
PMSPIPRequest	HEAP	USER_TABLE	10-04-2025 15:41
PRAirTicketPolicy	HEAP	USER_TABLE	08-02-2024 17:08
PRPayrollProvider	HEAP	USER_TABLE	08-11-2022 16:58
PRSocialInsuranceOverride	HEAP	USER_TABLE	08-02-2024 17:08
PSProjectMonthlyRevenue	HEAP	USER_TABLE	12-12-2024 19:03
PSResourceCost	HEAP	USER_TABLE	20-11-2023 18:04
Sector	HEAP	USER_TABLE	26-07-2019 18:59
TicketInternalNote	HEAP	USER_TABLE	15-11-2024 22:40

4. Index Fragmentation:

The index fragmentation analysis reveals a significant number of indexes with elevated fragmentation levels in the database. Specifically, there is 1 index with fragmentation greater than 90%, 3 indexes between 80-90%, and 6 indexes between 70-80%. Additionally, 63 indexes fall within the 60-70% range, and 32 indexes are between 50-60%. Notably, a substantial portion of indexes—84 in the 40-50% range, 163 in the 30-40% range, and 293 in the 20-30% range—also exhibit moderate to high fragmentation. Furthermore, 526 indexes are in the 10-20% range, and 553 indexes show minimal fragmentation (0-10%). It is important to highlight that several indexes with fragmentation above 30% also have a high page count, which can further impact query performance by increasing I/O operations and reducing efficiency. Addressing these highly fragmented and large indexes through targeted maintenance, such as index rebuilding or reorganizations, is recommended to improve overall database performance and reduce resource consumption.

Fragmentation Range	Index Count
Greater than 90	1
80-90%	3
70-80%	6
60-70%	63
50-60%	32
40-50%	84
30-40%	163
20-30%	293
10-20%	526
0-10%	553

5. Duplicate Index:

Table name	Index name	Column list	Index name	Column list
APIUser	PK_APIUser	Id	nci_msft_1_APIUser_70AC8F20FD0DF3850831773ABA43ABA5	Identifier, IsDeleted, ExpiresOn
AttendanceDevice	PK_AttendanceDevice	Id	nci_wi_AttendanceDevice_667E873112632BB82A6106A43E1D9C16	Identifier, PremiseId
AttendanceEmployeeShiftAssignment	nci_wi_AttendanceEmployeeShiftAssignment_D95BA1490A5C893BA791FCD0FDE2E5F7	EmployeeId	uq_employee_shift_assignment	EmployeeId, FromDate, ToDate
Attendanc	nci_wi_AttendanceEm	EmployeeId	uq_employee_weekl	EmployeeId, FromDate, ToDate

eEmployeeWeeklyOffAssignment	ployeeWeeklyOffAssignment_D95BA1490A5C893BA791FCD0FDE2E5F7		yoff_assignment	
AttendanceOvertimeRequest	nci_wi_AttendanceOvertimeRequest_E4C665480A009584C11CB7E8E0E47904	TenantId, RequestStatus, ApprovalLog, ApproverId, CommentIdentifier, CreatedBy, DateApprovedRejected, DateCreated, DateModified, EmployeeId, FromDate, ModifiedBy, Note, NotifyTo, OvertimeHours, OvertimePolicyIdentifier, RejectReason, RequestedOn, RequesterId, RequireApproval, ToDate	nci_msft_1_AttendanceOvertimeRequest_74CFDD3605AEC2B000735E0296C4C444	TenantId, RequestStatus, ApprovalLog, ApproverId, CommentIdentifier, CreatedBy, DateApprovedRejected, DateCreated, DateModified, EmployeeId, FromDate, ModifiedBy, Note, NotifyTo, OvertimeHours, OvertimePolicyIdentifier, RejectReason, RequestedOn, RequesterId, RequireApproval, ToDate, TotalOvertimeHours
AttendanceTrackingPolicy	PK_AttendanceTrackingPolicy	Id	nci_wi_AttendanceTrackingPolicy_8A88EC13099F6F139FB089B9BC429AA2	Identifier
BlobMeta	pk_BlobMeta	Id	nci_wi_BlobMeta_4623D77EEB282433F455FE7E986E4075	Id, TenantId
CompositeView	PK__Composit__3214EC074CEEFBB0	Id	nci_wi_CompositeView_5A0676183141DBC1920AC85B6AAE9B4C	Identifier, Name
ContingentEmployee	UQ_ContingentEmployee_TenantId_EmployeeId	TenantId, EmployeeId	ContingentEmployee_TenantId_EmployeeId	TenantId, EmployeeId
ContingentEmployee	ContingentEmployee_TenantId_EmployeeId	TenantId, EmployeeId	UQ_ContingentEmployee_TenantId_EmployeeId	TenantId, EmployeeId
DocumentChan	nci_wi_DocumentChangeRequest_7E6DDA77	IsDeleted, Status, TenantId	nci_wi_DocumentChangeRequest_56F15	IsDeleted, Status, TenantId, ApproverId, Attributes, DateCreated, DocumentId, DocumentTypeId, EmployeeId, ExpiresOn, Files, RequestedOn

geRequest	DA1269986B153875672AD6F5		6F839722285039E2BD2028131F0	
DocumentType	DocumentType_TenantId	TenantId, DocumentFolderId, IsDeleted	DocumentTypeLookup	TenantId, DocumentFolderId, IsDeleted, Identifier, IsSystemGenerated, HasExpiryDate
Employee	PK_Employee	Id	nci_wi_Employee_17D0D99A4346653E211C6BDB87BD00A7	Identifier, OrgUserType, DisplayName
EmployeeBonusPayment	EmployeeBonusPayment_Tenant	TenantId	EmployeeBonusPaymentCurrencyLookup	TenantId, EmployeeId, IsDeleted, CurrencyId
EmployeeBonusPayment	SalaryIdentifierLookup	SalaryIdentifier, EmployeeId, TenantId	EmployeeBonusPaymentLookup	SalaryIdentifier, EmployeeId, TenantId, IsDeleted, Status, BonusTypeId
EmployeeEmailStatus	EmployeeEmailStatus_TenantId_EmployeeId_Email	TenantId, EmployeeId, Email	EmployeeEmailStatus_TenantId	TenantId, EmployeeId, Email, IsEmailDisabled
EmployeeModuleStatus	EmployeeModuleStatus_TenantId	TenantId	nci_wi_EmployeeModuleStatus_467A6A4568C136556640686DEFDF3FB3	TenantId, EmployeeId, Module, CreatedBy, DateCreated, DateModified, Enabled, LastDisabledOn, LastEnabledOn, ModifiedBy
EmployeeNumberSeries	ExpenseClaimNumberSeries_Tenant	TenantId	EmployeeNumberSeries_Tenant	TenantId
EmployeeNumberSeries	EmployeeNumberSeries_Tenant	TenantId	ExpenseClaimNumberSeries_Tenant	TenantId
EmployeePayrollProfile	UC_EmployeePayrollProfile	TenantId, EmployeeId	EmployeePayrollProfileLookup	TenantId, EmployeeId, BankDetails, Enable, ESI, PayGroupId, PF, ReviewPayrollSetUp

EmployeePayrollProfile	nci_wi_EmployeePayrollProfile_5FEF727B51057B3D8BECFCB4AAE17130	PayGroupId, Enable, EmployeeId	IX_EmployeePayrollProfile	PayGroupId, Enable, EmployeeId, TenantId
EmployeePayrollProfile	nci_wi_EmployeePayrollProfile_60E8548144B90FE89E47601AD89E87C1	TenantId, Enable, EmployeeId	nci_wi_EmployeePayrollProfile_3740AEA0D3047B12EACCBCE5C0CAF8E6	TenantId, Enable, EmployeeId, BankDetails, ESI, PF, PTConfigurationId
EmployeePraiseAssignment	EmployeePraiseAssignment_EmployeeId_IsDeleted	EmployeeId, IsDeleted	nci_wi_EmployeePraiseAssignment_596B0E99F5B016FAB1B8346C7A494C13	EmployeeId, IsDeleted, PraiseId
EmployeeSalary	EmployeeSalary_Tenant	TenantId, EmployeeId, IsDeleted, EffectiveFrom	EmployeeSalary_Currency	TenantId, EmployeeId, IsDeleted, EffectiveFrom, CurrencyId
EmployeeSalary	EmployeeSalary_Tenant	TenantId, EmployeeId, IsDeleted, EffectiveFrom	EmployeeSalary_Identifier	TenantId, EmployeeId, IsDeleted, EffectiveFrom, Identifier
EmployeeWish	PK_Wish	Id	nci_msft_1_EmployeeWish_D8EDAB3A769FD25E03342CEDFF3250C3	Identifier, TenantId
EntityFieldGroup	EntityFieldGroup_Tenant_UniqueIdentifier	TenantId, UniqueIdentifier	UNQ_EntityFieldGroup	TenantId, UniqueIdentifier, EntityType, CountryCode, IsDeleted
ExitProcessActionTracker	UX_ExitProcessActionTracker_TenantId_ExitRequestId_ActionType	TenantId, ExitRequestId, ActionType	Idx_ExitProcessActionTracker_TenantId_RequestId	TenantId, ExitRequestId, ActionType, Status
ExitRequest	UX_ExitRequest_ExitRequestStatu	TenantId, EmployeeId, ExitRequestStatus	IDX_ExitRequest_TenantId	TenantId, EmployeeId, ExitRequestStatus, LastWorkingDate, RequestedOn, SettlementDate, TerminationType, ExtendedDays

	s_EmployeeId			
Expense	ExpenseTenantLookup	TenantId, EmployeeId	EmployeeExpenseClaimLookup	TenantId, EmployeeId, ExpenseClaimId, IsAdvanceRequest
Expense	ExpenseTenantLookup	TenantId, EmployeeId	Idx_Expense_Usage	TenantId, EmployeeId, ExpenseTypeId, IsAdvanceRequest, IsDeleted, Id, BillingDate, Amount
ExpenseClaim	ExpenseClaimTenantLookup	EmployeeId, TenantId	nci_wi_ExpenseClaim_C5144C61D67D3AFDA336AF74A66BB651	EmployeeId, TenantId, ApprovalStatus
ExpensePolicy	nci_wi_ExpensePolicy_65E1AA5D71DC57DF69E1834BE22631F1	TenantId	ExpensePolicy_Tenant	TenantId, IsDeleted, HasCategoryLevelApproval, HasMaxAmountApproval
GroupEmploye e	nci_wi_GroupEmployee_D95BA1490A5C893BA791FCD0FDE2E5F7	EmployeeId	DF__GroupEmployee__UniqueRows	EmployeeId, GroupTypeId, GroupId
GroupType	Idx_GroupType_TenantId	TenantId	Idx_GroupType_HasVisibilityRestriction	TenantId, HasVisibilityRestriction
InboxListItem	Inbox_Employee	TenantId, EmployeeId, ItemType, Status, IsDeleted, RequestId	Inbox_RequestIdentifier	TenantId, EmployeeId, ItemType, Status, IsDeleted, RequestIdentifier
KBTag	UQ_ArticleTag	Id	PK_ArticleTag	Id
KBTag	PK_ArticleTag	Id	UQ_ArticleTag	Id
LeaveType	PK_LeaveType	Id	LeaveType_Dayoff	Id, IsSick, IsPaid
LeaveType	nci_wi_LeaveType_D162F476A89A006FAA25AE60CE00075E	TenantId, IsDeleted	LeaveType_TenantId	TenantId, IsDeleted
LeaveType	LeaveType_TenantId	TenantId, IsDeleted	nci_wi_LeaveType_D162F476A89A006F	TenantId, IsDeleted

			AA25AE60CE00075E	
LoanTaxableInterestRate	LoanTaxableInterestRate_Tenant	TenantId	LoanTaxableInterestRate_Category	TenantId, LoanCategoryId
LoanTaxableInterestRate	LoanTaxableInterestRate_Category	TenantId, LoanCategoryId	UC_LoanTaxableInterestRate	TenantId, LoanCategoryId, InterestRate, FinancialYear
LoanTaxableInterestRate	LoanTaxableInterestRate_Tenant	TenantId	UC_LoanTaxableInterestRate	TenantId, LoanCategoryId, InterestRate, FinancialYear
OnboardEmployee	Idx_OnboardEmployee_TenantId	TenantId	Idx_OnboardEmployee	TenantId, Status
OnboardEmployee	Idx_OnboardEmployee	TenantId, Status	nci_msft_1_OnboardEmployee_0A5FBC7B9B38CC6089EF12AA79AD7E26	TenantId, Status, CancelledOn, CancelledReason, CompletedOn, EmployeeId, FlowId, InitiatedOn
OnboardEmployee	Idx_OnboardEmployee_TenantId	TenantId	nci_msft_1_OnboardEmployee_0A5FBC7B9B38CC6089EF12AA79AD7E26	TenantId, Status, CancelledOn, CancelledReason, CompletedOn, EmployeeId, FlowId, InitiatedOn
OnboardEmployee	Idx_OnboardEmployee_TenantId	TenantId	nci_wi_OnboardEmployee_1DBF843D7E30EC7852339277CF36B635	TenantId, FlowId, Status, DateModified, EmployeeId
OnboardEmployee	Idx_OnboardEmployee_TenantId	TenantId	nci_wi_OnboardEmployee_C3FCE98B5B0503A06EB972AC79D254A9	TenantId, IsOnboardSkipped, Status, FlowId, EmployeeId

OnboardFlow	Idx_OnboardFlow_TenantId	TenantId	Idx_OnboardFlow	TenantId, IsDefault, IsDeleted
PMSEmployeeReview	nci_wi_PMSEmployeeReview_B1B82A45E5D9DBDF8E9D6E991971C36B	ReviewCycleId	nci_wi_PMSEmployeeReview_74ECDCF59C317D27A9FC72B24F85A28A	ReviewCycleId, TenantId, BandAllocationStatus, ReviewCalibrationStatus, Status
PMSEmployeeReview	PMSEmployeeReview_Tenant	TenantId, ReviewCycleId	nci_wi_PMSEmployeeReview_9A9CCC03E174B306982932AD1E96E508	TenantId, ReviewCycleId, EmployeeId, Status
PMSExternalReviewerProfile	PK_PMSReviewExternal	Id	nci_wi_PMSExternalReviewerProfile_8DA3F32B69728A2FF9D3F7E129303487	Identifier, ReviewId, TenantId, Email, FirstName, JobTitle, LastName
PMSObjective	PMSObjective_Tenant	TenantId, TimeFrameId	PMSObjective_TenantId_TimeFrameId_IsDeleted_IsDraft_ParentRelationType_Index	TenantId, TimeFrameId, IsDeleted, IsDraft, ParentRelationType, OwnerId
PMSObjectiveProgressUpdateLog	PMSObjectiveProgressUpdateLog_Tenant	TenantId, TimeFrameId	nci_wi_PMSObjectiveProgressUpdateLog_619BA4FAE6E56D9A12BE0621B12E7D26	TenantId, TimeFrameId, EmployeeId, ObjectiveProgress, ParentRelationType, UpdatedOn
PMSObjectiveProgressUpdateLog	PMSObjectiveProgressUpdateLog_Tenant	TenantId, TimeFrameId	nci_wi_PMSObjectiveProgressUpdateLog_B6C3866553D969D0BC271891C3F7B936	TenantId, TimeFrameId, CreatedBy, DateCreated, DateModified, EmployeeId, GroupId, KeyResultId, KeyResultProgress, ModifiedBy, ObjectiveId, ObjectiveProgress, ParentObjectiveId, ParentObjectiveProgress, Status, UpdatedBy, UpdatedOn

PMS ObjectiveTag	PMSObjectiveTag_Tenant	TenantId	PMSObjectiveTag_TenantId_Identifier_IsDeleted	TenantId, Identifier, IsDeleted
PMS ObjectiveTagMapping	PMSObjectiveTagMapping_Tenant	TenantId	PMSObjectiveTagMapping_Objective_Tag	TenantId, ObjectiveId, TagId
PMS ObjectiveTagMapping	PMSObjectiveTagMapping_Tenant	TenantId	PMSObjectiveTagMapping_TenantId_TagIdentifier_ObjectiveIdentifier	TenantId, TagIdentifier, ObjectiveIdentifier, IsDeleted
PMS ObjectiveTimeFrame	PMSObjectiveTimeFrame_Tenant	TenantId	PMSObjectiveTimeFrame_Identifier_Index	TenantId, Identifier, IsDeleted
PMS OneOnOneMeeting	PMSOneOnOneMeeting_Tenant	TenantId, Id	PMSOneOnOneMeeting_Identifier_Index	TenantId, Identifier
PMS RatingMetricUnit	PK_PMSRatingMetricUnit	Id	nci_wi_PMSRatingMetricUnit_2E1B33D3550793E92088F0839FC4AA01	Identifier, TenantId, Title
PMS RecurringMeetingConfiguration	PMSRecurringMeetingConfiguration_TenantId	TenantId, Id	PMSRecurringMeetingConfiguration_Identifier_Index	TenantId, Identifier
PMS ReviewGroupCycle	Idx_PMSReviewGroupCycle_TenantId	TenantId	PMSReviewGroupCycle_Identifier_Index	TenantId, Identifier, IsDeleted
PMS ReviewRatingTra	PMSReviewRatingTransaction_Tenant	TenantId, ReviewId	PMSReviewRatingTransaction Unique	TenantId, ReviewId, ReviewerId, ReviewObjectiveId, ReviewObjective, ReviewObjectiveGroupId, ReviewerType, ExternalReviewerIdentifier, RatingIdentifier

nsaction				
PRBonusAccrual	PRBonusAccrual_Index	TenantId, EmployeeId, SalaryIdentifier, ComponentIdentifier, CycleId	UC_PRBonusAccrual	TenantId, EmployeeId, SalaryIdentifier, ComponentIdentifier, CycleId, ProcessedCycleId
PRBonusPayment	PRBonusPayment_index	EmployeeId, TenantId, ProcessedCycleId	PRBonusPaymentCurrencyLookup	EmployeeId, TenantId, ProcessedCycleId, CurrencyId
PRBonusTransaction	PRBonusTransaction_TenantId	TenantId	PRBonusTransaction_StatusLookup	TenantId, employeeId, BonusId, ProcessedCycleId, Status, PaymentMode, IsDeleted
PRBonusType	PK__PRBonusType	Id	nci_wi_PRBonusType_2E1B33D3550793E92088F0839FC4AA01	Identifier, TenantId, Title
PRBudgetEstimationReport	Idx_PRBudgetEstimationReport	TenantId, IsDeleted, ReportSavedBy	PRBudgetEstimationReportCurrencyLookup	TenantId, IsDeleted, ReportSavedBy, CurrencyId
PRCompensationPlanningDetails	Idx_PRCompensationPlanningDetails	TenantId, Month, EmployeeId	PRCompensationPlanningDetailsCurrencyLookup	TenantId, Month, EmployeeId, CurrencyId
PREmployeeAirTicketPolicyAssignment	PREmployeeAirTicketPolicyAssignment_TenantId	TenantId	UQ_Tenant_Employee_Air_Ticket_Policy_Assignment	TenantId, EmployeeId
PREmployeeBenefit	PREmployeeBenefit_TenantId	TenantId	nci_msft_1_PREmployeeBenefit_D63180E5A458C078BBECA3FA630DEFD1	TenantId, IsDeleted, SalaryStructureIdentifier, Amount, BenefitConfigurationId, CreatedBy, CurrencyId, DateCreated, DateDeleted, DateModified, EffectiveFrom, EmployeeId, EndDate, FinancialYear, IsOneTimePaymentPerk, ModifiedBy, PayGroupId, ProcessedCycleId, ShowEmployerTaxToEmployee, TaxAmount, Title
PREmplo	SalaryIdentifierLookup	SalaryStructureIdentifier, EmployeeId, FinancialYear, TenantId	PREmployeeBenefit	SalaryStructureIdentifier, EmployeeId, FinancialYear, TenantId, CurrencyId

yeeBenefit			CurrencyLookup	
PREmployeeCurrentCompensationDetails	Idx_PREmployeeCurrentCompensationDetails	TenantId, EmployeeId	PREmployeeCurrentCompensationDetailsCurrencyLookup	TenantId, EmployeeId, CurrencyId
PREmployeeForm16GenerationRequest	PREmployeeForm16GenerationRequest_TenantId	TenantId	UC_PREmployeeForm16GenerationRequest	TenantId, FinancialYear, EmployeeId, LegalEntityId
PREmployeeLegalEntityAssignment	PREmployeeLegalEntityAssignment_UniqueIndex	TenantId, EmployeeId	PREmployeeLegalEntityAssignment_NC_Index	TenantId, EmployeeId
PREmployeeLegalEntityAssignment	PREmployeeLegalEntityAssignment_NC_Index	TenantId, EmployeeId	PREmployeeLegalEntityAssignment_UniqueIndex	TenantId, EmployeeId
PREmployeeLoan	PREmployeeLoan_TenantId	TenantId	nci_wi_PREmployeeLoan_7DBAC6391EF1A0C07259AD21BA8FA1BB	TenantId, RepaymentStatus, EmployeeId
PREmployeeLoan	PREmployeeLoan_TenantId	TenantId	nci_wi_PREmployeeLoan_A302A5B30960CD89F0CAC44C0E9E6799	TenantId, RequestedOn
PREmployeeLoan	PREmployeeLoan_TenantId	TenantId	PREmployeeLoanCurrencyLookup	TenantId, CurrencyId

PREmployeeLoan	PREmployeeLoan_TenantId	TenantId	PREmployeeLoanTaxableInterestRateLookup	TenantId, EmployeeId, LoanRequestStatus, OutstandingAmount, TaxableInterestRate
PREmployeeOverrides	UC_PREmployeeOverrides	TenantId, EmployeeId	PREmployeeOverrides_TenantId	TenantId, EmployeeId
PREmployeeOverrides	PREmployeeOverrides_TenantId	TenantId, EmployeeId	UC_PREmployeeOverrides	TenantId, EmployeeId
PREmployeeRunOnHold	PREmployeeRunOnHold_TenantId	TenantId	nci_wi_PREmployeeRunOnHold_3889BB6BA2A431DDE70DBE79239DCD8A	TenantId, IsDeleted, Month, ProcessedCycleId, Year, EmployeeId, CycleId
PREmployeeRunOnHold	PREmployeeRunOnHold_TenantId	TenantId	PREmployeeRunOnHold_UniqueIndex	TenantId, PayGroupId, Year, Month, CycleId, EmployeeId
PRESIOverride	PRLWFOverride_index	EmployeeId, TenantId, From	PRESIOverride_index	EmployeeId, TenantId, From
PRESIOverride	PRESIOverride_index	EmployeeId, TenantId, From	PRLWFOverride_index	EmployeeId, TenantId, From
PRFinancialYearWiseEmployeeCompensation	Idx_PRFinancialYearWiseEmployeeCompensation	TenantId, PayGroupId, From, To, EmployeeId	PRFinancialYearWiseEmployeeCompensationCurrencyLookup	TenantId, PayGroupId, From, To, EmployeeId, CurrencyId
PRLeaveEncashment	PRLeaveEncashment_TenantId	TenantId	nci_wi_PRLeaveEncashment_8EA13E5A764FAF69AEB44B08F8753159	TenantId, ProcessedCycleId, EmployeeId, PayDate, Amount, Comments, CreatedBy, DateCreated, DateModified, Days, LeaveTypeId, ModifiedBy, PaidOn, PayAction
PRLeaveEncas	PRLeaveEncashment_TenantId	TenantId	PRLeaveEncashment_LeaveEn	TenantId, EmployeeId, LeaveEncashmentRequestId

hment			cashmentRequestId	
PRLeaveEncashment	PRLeaveEncashment_TenantId	TenantId	PRLeaveEncashmentCurrencyLookup	TenantId, CurrencyId
PROneTimeTransaction	PROneTimeTransaction_index	EmployeeId, TenantId, CycleId, ProcessedCycleId	PROneTimeTransactionCurrencyLookup	EmployeeId, TenantId, CycleId, ProcessedCycleId, CurrencyId
PSEmployeeTimesheetProfile	UC_PSEmployeeTimesheetProfile	TenantId, EmployeeId	PSEmployeeTimesheetProfile_TenantId	TenantId, EmployeeId, TimesheetPolicyId, EffectiveFrom, Enabled
PSProject	nci_wi_PSProject_16BE9FBD4747EC27A5C8D8DF1A857C66	TenantId, IsDeleted, AssignedToAll	nci_wi_PSProject_F04FD0670DB0EF1C19ED491B1DD67524	TenantId, IsDeleted, AssignedToAll, AllowNonBillableHours, BillingRate, BillingType, ClientId, Code, CommentsRequired, CreatedBy, DateCreated, DateDeleted, DateModified, Description, EnableTimer, EndDate, IsBillable, ModifiedBy, Name, StartDate, Status, TimesheetSettings, TrackTimeForTask
PSProjectHealthSettings	PK_PSProjectHealthSettings	Id	PSProjectHealthSettings_Index	Id, TenantId
PSProjectManager	nci_wi_PSProjectManager_FE51BB284825E0B9A2DB18C79926D4E7	IsDeleted, ProjectId, EmployeeId	nci_msft_1_PSProjectManager_750A7E0000ED6E78B5A9D2ED55BE77D4	IsDeleted, ProjectId, EmployeeId, ManageTasks, ManageTeam
PSProjectTask	PK_ProjectTask	ID	nci_msft_1_PSProjectTask_1597C5F995E09C6209044F268BDCE881	Identifier, IsDeleted, TenantId
PSProjectTask	IX_PSProjectTask_TimeEntry	TenantId, ProjectId, TaskType, IsDeleted, ApprovalStatus	PSProjectTask_TimeEntryTask	TenantId, ProjectId, TaskType, IsDeleted, ApprovalStatus, DateModified
PSRetainerBilli	PK_PSRetainerBillingC	Id	PSRetainerBillingCh	Id, TenantId, BillingChargeId, StartDate, EndDate, IsCharged, IsDeleted, ProjectId

ngChargeLineItem	hargeLineItem		argeLineItem_Index	
PSRetainerChargeLineItem	PK_PSRetainerChargeLineItem	Id	PSRetainerChargeLineItem_Index	Id, TenantId, ChargeId, StartDate, EndDate, IsDeleted, ProjectId
PSTaskStage	PSTaskStage_Tenant	TenantId, IsDeleted	IX_PSTaskStage_IsDeleted	TenantId, IsDeleted
PSTaskStage	IX_PSTaskStage_IsDeleted	TenantId, IsDeleted	PSTaskStage_Tenant	TenantId, IsDeleted
PSTaskStage	IX_PSTaskStage_IsDeleted	TenantId, IsDeleted	PSTaskStage_tenantId_IsDeleted	TenantId, IsDeleted, DateModified
PSTaskStage	PSTaskStage_Tenant	TenantId, IsDeleted	PSTaskStage_tenantId_IsDeleted	TenantId, IsDeleted, DateModified
PSTimesheetEntry	IX_PSTimesheetEntry_TimeEntryStatus	TenantId, ProjectId, TaskId, Status, IsDeleted	PSTimesheetEntry_TimeEntryStatus	TenantId, ProjectId, TaskId, Status, IsDeleted, DateModified
PSTimesheetPolicyPeriod	nci_wi_PSTimesheetPolicyPeriod_61801BA27E7CEEAC8745808BA79F0953	IsDeleted, TimesheetPolicyId, EndDate, StartDate	nci_msft_1_PSTimesheetPolicyPeriod_4EA4E171615CE5B8FDD57D07FA32FD87	IsDeleted, TimesheetPolicyId, EndDate, StartDate
PSTimesheetPolicyPeriod	nci_msft_1_PSTimesheetPolicyPeriod_4EA4E171615CE5B8FDD57D07FA32FD87	IsDeleted, TimesheetPolicyId, EndDate, StartDate	nci_wi_PSTimesheetPolicyPeriod_61801BA27E7CEEAC8745808BA79F0953	IsDeleted, TimesheetPolicyId, EndDate, StartDate
RequestApprover	nci_wi_RequestApprover_EDE387D76C44C6F37F7AB9BFC1E590A6	ApproverId, TenantId, ApprovalRequestType, ApproverType, CreatedBy, DateCreated, DateModified, Level, ModifiedBy, RequestId	nci_wi_RequestApprover_62244BF986E9F09E10CA6669FA02D27D	ApproverId, TenantId, ApprovalRequestType, ApproverType, CreatedBy, DateCreated, DateModified, Level, ModifiedBy, RequestId, RequestIdentifier

RequestApprover	RequestApprover_RequestId	TenantId, RequestId	nci_wi_RequestApprover_F7C9B5BF801959D032344C1FE52E9B17	TenantId, RequestId, ApprovalRequestType, ApproverId, ApproverType, CreatedBy, DateCreated, DateModified, Level, ModifiedBy
ROAnnouncement	PK__tmp_ms_x__3214EC07CA2E681C	Id	nci_msft_1_ROAnnouncement_1597C5F995E09C6209044F268BDCE881	Identifier, IsDeleted, TenantId
ROAnnouncement	nci_wi_ROAnnouncement_FFEE839B0FF7B36E2BD69DBC45193770	IsDeleted, TenantId, Status	nci_msft_1_ROAnnouncement_C6A9A6966D43D9CB37A6A89E7FE2822B	IsDeleted, TenantId, Status, AcknowledgementCount, AddedBy, AddedOn, AllDepartments, AllEmployees, AllLocations, AllowLikes, Attachments, CommentIdentifier, Content, Excerpt, HeaderImageUrl, HideAnnouncement, HideAnnouncementAfter, Identifier, IsAddedFromWall, PublishedBy, PublishedOn, ReactionIdentifier, RequiresAcknowledgement, SelectedDepartments, SelectedEmployees, SelectedLocations, Title, ViewsCount
UserProfile	Idx_UserProfile_TenantId	TenantId	nci_msft_1_UserProfile_9E70040F91F19979BA05B7869469B546	TenantId, PersonalEmail, EmployeeId, BloodGroup, ContactInfo, CreatedBy, CurrentAddress, DateCreated, DateModified, DateOfBirth, DialCode, Education, Experience, FirstName, Gender, HomePhone, LastName, MaritalStatus, MarriageDate, MiddleName, MobilePhone, ModifiedBy, Nationality, PermanentAddress, PersonalInfo, ProfessionalInfo, ProfessionalSummary, Relations, SkypeId, SocialNetworks, Status
UserProfile	Idx_UserProfile_TenantId	TenantId	nci_wi_UserProfile_7032B260F9A4BB9AD2215352CA91792C	TenantId, UserId, FirstName, LastName, MiddleName, MobilePhone

6. Top 10 table size & Archival Recommendation

The database exhibits significant performance degradation primarily due to extensive disk I/O waits, as evidenced by the high PAGEIOLATCH_SH wait times. This is largely driven by very large table sizes—several tables exceed hundreds of gigabytes—resulting in frequent page reads from disk and increased query latency.

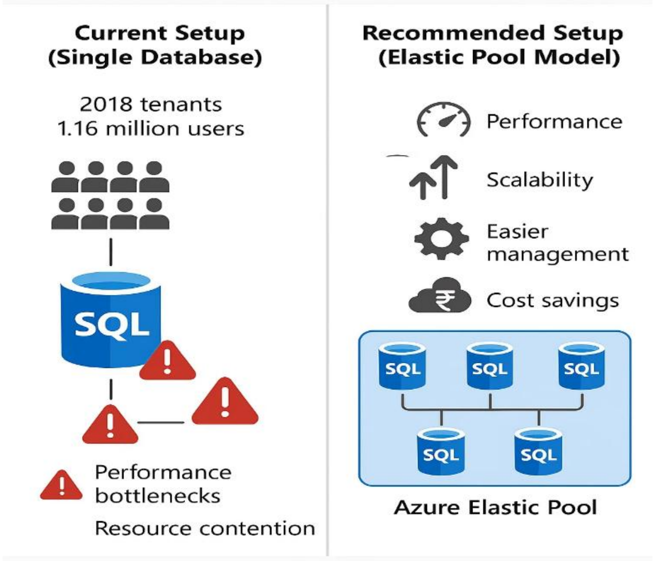
Table	Used MB	Allocated MB
dbo.AttendanceSummary	797380.05	852854.97
dbo.PRPayAdvice	698016.13	755266.82
dbo.AttendanceRemoteClockInRequest	350976.42	361900.32
dbo.PREmployeeSalaryStructure	264893.5	273844.12
dbo.PSTimesheetEntry	235905.07	240770.95
dbo.PREmployeeRunAggregates	209903.61	215639.67
dbo.PRPaySlip	199950.18	201350.63
dbo.LeaveRequest	193069.23	214421.69
dbo.PSEmployeeTimesheet	178168.41	192763.77
dbo.AttendanceLog	176129.3	177284.61

Recommendation:
To address this, it is recommended to implement a data archival strategy that moves historical and infrequently accessed data to separate archive storage. Archiving older data will reduce the active dataset size, lower I/O pressure, and improve overall query performance. Additionally, consider scheduling regular index maintenance and monitoring memory usage to further optimize system responsiveness.

7. Tenant Management and Multi-Tenancy Recommendations

Current Tenant Landscape:

The client currently supports **2018** tenants within a single Azure SQL Database of approximately 5 TB in size. The total user base across all tenants exceeds **1.16** million users, including **361,066** active users. This single-database multi-tenant approach can lead to resource contention, performance bottlenecks, and challenges in scalability and maintenance.



Recommendation:
To enhance performance, scalability, and manageability, it is recommended to adopt a **multi-tenant model** using **multiple databases**, where tenants are distributed across several databases rather than sharing one large database. Leveraging **Azure SQL Database Elastic Pools** will allow these databases to share resources efficiently, reducing costs while maintaining performance.

Benefits of Multi-Database Multi-Tenancy Approach:
Improved Performance: Tenant workloads are isolated, reducing noisy neighbor effects and resource contention.
Scalability: Databases can be scaled independently based on tenant needs, supporting growth without impacting others.
Cost Efficiency: Elastic pools optimize resource utilization by sharing compute and storage among multiple databases.
Simplified Management: Easier backup, restore, and maintenance operations at the tenant or group level.

8. Missing Index:

Below are the missing index details where most of index suggestions are 90% plus user impact. So we are suggesting to test on UAT first & create those indexes.

create_index_statement	avg_user_impact
CREATE INDEX missing_index_6_5 ON [sqldb-kekahr].[dbo].[BlobMeta] ([TenantId], [BlobName])	99.91
CREATE INDEX missing_index_3641_3640 ON [sqldb-kekahr].[dbo].[AttendanceRequestTransaction] ([TenantId],[RequestType]) INCLUDE ([RequestId], [TimeEntries])	98.71
CREATE INDEX missing_index_10_9 ON [sqldb-kekahr].[dbo].[AttendanceRequestTransaction] ([TenantId], [RequestId],[RequestType]) INCLUDE ([TimeEntries])	99.29
CREATE INDEX missing_index_4_3 ON [sqldb-kekahr].[dbo].[AttendanceLog] ([TenantId], [EmployeeAttendanceNumber],[Timestamp]) INCLUDE ([AttendanceSyncLogId], [DeviceIdentifier], [Status])	99.54
CREATE INDEX missing_index_3737_3736 ON [sqldb-kekahr].[dbo].[PSTimesheetEntry] ([TenantId], [IsDeleted],[Date]) INCLUDE ([EmployeeId], [TaskId], [ProjectId])	99.27
CREATE INDEX missing_index_3740_3739 ON [sqldb-kekahr].[dbo].[PSTimesheetEntry] ([TenantId], [IsDeleted],[Date]) INCLUDE ([EmployeeId], [TaskId], [Status], [Comments], [Billable], [StartTime], [EndTime], [ProjectId], [Identifier], [TotalMinutes])	71.66
CREATE INDEX missing_index_570_569 ON [sqldb-kekahr].[dbo].[InboxListItem] ([TenantId], [ItemType], [IsDeleted], [RequestIdentifier]) INCLUDE ([Status], [RequestId], [EmployeeId], [RequestDetails], [DateCreated], [DateModified], [CreatedBy], [ModifiedBy], [ArchivedOn], [DeletedOn])	92.98
CREATE INDEX missing_index_180_179 ON [sqldb-kekahr].[dbo].[LeaveRequest] ([TenantId], [IsDeleted],[StatusId])	87.29
CREATE INDEX missing_index_26_25 ON [sqldb-kekahr].[dbo].[LeaveRequest] ([Identifier], [TenantId], [IsDeleted])	97.94
CREATE INDEX missing_index_448_447 ON [sqldb-kekahr].[dbo].[PSDashboardInfo] ([TenantId], [IsDeleted]) INCLUDE ([DashboardRequestId], [DashboardName], [Persona], [ParentId], [Data])	94.33
CREATE INDEX missing_index_406_405 ON [sqldb-kekahr].[dbo].[ExpenseCategoryRequest] ([TenantId], [IsDeleted],[RequestStatus]) INCLUDE ([ClaimId])	59.46
CREATE INDEX missing_index_1668_1667 ON [sqldb-kekahr].[dbo].[Expense] ([TenantId], [IsDeleted]) INCLUDE ([Amount], [LinkedEntityId], [IsReimbursed])	98.89
CREATE INDEX missing_index_4676_4675 ON [sqldb-kekahr].[dbo].[PREmployeeMaxRunStatus] ([EmployeeId], [CycleId]) INCLUDE ([Status])	99.95
CREATE INDEX missing_index_468_467 ON [sqldb-kekahr].[dbo].[EmployeeAssetAssignment] ([TenantId], [AssetAssignmentId], [AssignmentStatus]) INCLUDE ([ActionTakenOn])	95.97

CREATE INDEX missing_index_470_469 ON [sqldb-kekahr].[dbo].[EmployeeAssetAssignment] ([TenantId], [AssignmentStatus]) INCLUDE ([AssetAssignmentId], [ActionTakenOn])	92.76
CREATE INDEX missing_index_323_322 ON [sqldb-kekahr].[dbo].[PREmployeeRunAggregates] ([TenantId], [GenerateInCycle]) INCLUDE ([EmployeeId], [CycleId], [Aggregate], [HasPF], [HasESI], [HasPT], [ProcessedCycleId], [RunId], [HasLWF], [CurrencyId], [OffCycleId])	83.77
CREATE INDEX missing_index_1196_1195 ON [sqldb-kekahr].[dbo].[LeaveRequest] ([Identifier], [TenantId])	99.84
CREATE INDEX missing_index_1528_1527 ON [sqldb-kekahr].[dbo].[InboxListItem] ([TenantId], [Status], [ItemType], [RequestIdentifier]) INCLUDE ([RequestId], [EmployeeId], [RequestDetails], [DateCreated], [DateModified], [CreatedBy], [ModifiedBy], [IsDeleted], [ArchivedOn], [DeletedOn])	84.75
CREATE INDEX missing_index_1216_1215 ON [sqldb-kekahr].[dbo].[LetterPublished] ([TenantId], [Identifier])	99.65
CREATE INDEX missing_index_32_31 ON [sqldb-kekahr].[dbo].[ExitRequest] ([TenantId],[EmployeeId], [ExitRequestStatus]) INCLUDE ([RequestedOn], [TerminationType], [TerminationReason])	93.55

9. Index and Statistics Maintenance Strategy

Current Practice:

Index maintenance is performed using a combination of manual scripts, with index rebuilds by fragmentation levels and table usage. Due to the large database size (~5 TB), full index maintenance for all tables can take several days. Currently, high-usage tables are maintained weekly, and other tables as needed.

Recommendation and Action Plan:

Frequency	Target Tables	Action (Fragmentation %)	Mode/Options	Tool/Script
Daily	Most-used, high-fragmented	Rebuild (>30%), Reorganize (10–30%), Update Statistics	ONLINE, RESUMABLE for rebuilds;	Ola Hallengren IndexOptimize
Weekly	All large/high activity	Rebuild (>20–25%), Reorganize (10–20%), Update Statistics	ONLINE, RESUMABLE for rebuilds;	Ola Hallengren IndexOptimize

10. Page Compression:

Overview:

Page compression is an advanced SQL Server and Azure SQL Database feature that reduces the storage footprint of tables and indexes by compressing data at the page level. This not only saves storage costs but can also improve I/O performance, especially for large databases like yours (~5 TB), by reducing the amount of data read and written to disk.

Recommendation:

Implement page compression for **AttendanceSummary(852 GB)** table where performance analysis shows high storage usage and significant I/O activity.

11. Table Partitioning

Current Situation:

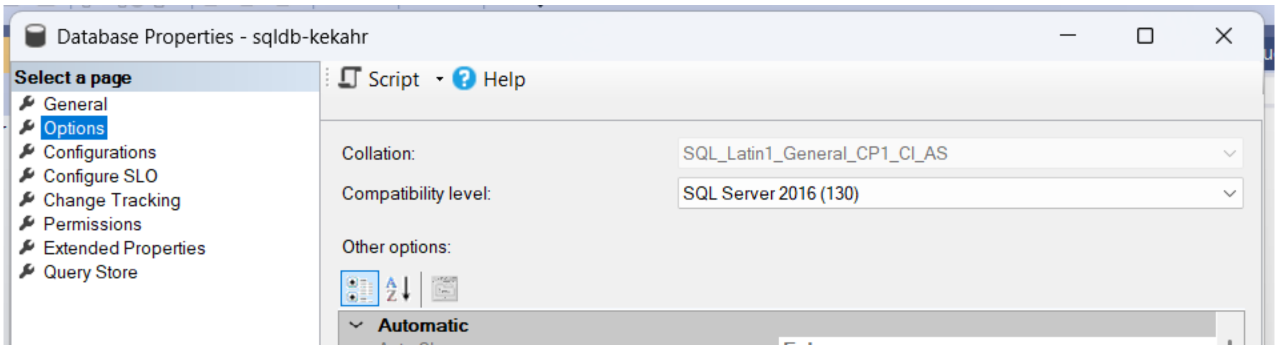
The table dbo.AttendanceSummary is very large, occupying approximately 797 GB used out of 853 GB allocated, which can lead to slower query performance and longer maintenance windows. In contrast, the PREmployeeMaxRunStatus table is already partitioned, benefiting from improved manageability and performance.

Recommendation:

Partition the large **dbo.AttendanceSummary** table based on **tenant_id** to improve query performance and reduce maintenance time.

12. Upgrade Database Compatibility Level to SQL Server 2019:

The current database compatibility level is set to **SQL Server 2016**, which will reach end of mainstream support. It is strongly recommended to upgrade the compatibility level to **SQL Server 2019** to leverage significant improvements in query optimization, performance, and security.



Database Properties - sqldb-kekahr

Collation: SQL_Latin1_General_CP1_CI_AS

Compatibility level: SQL Server 2016 (130)

13. Consolidated Recommendations

13.1 Infrastructure and Platform

- Consider migrating from Azure SQL Database PaaS to SQL Server on Azure Virtual Machine 2019:
- Full control over OS and SQL Server instance-level configurations
- Use of unsupported features in PaaS (e.g., cross-database queries, CLR integration, SQL Server Agent jobs with full control)
- Custom software dependencies or strict compliance requirements.
- Plan and execute a lift-and-shift migration with minimal code changes using Azure VM images pre-configured for SQL Server 2019, leveraging Azure Hybrid Benefit for cost savings

13.2 Utilize the Existing HA Configuration for Read Scale-out

- Your existing HA solution should be used to its full potential to offload read operations and provide high availability, avoiding unnecessary additional HA deployments.
- Configure readable secondary replicas and route read-only workloads accordingly.

13.3 Tenant Management and Multi-Tenancy Recommendations

- To improve performance and scalability, it is recommended to adopt a multi-tenant architecture using multiple databases with Azure SQL Database Elastic Pools. This approach isolates tenant workloads, optimizes resource utilization, and simplifies management, effectively reducing resource contention in a large multi-tenant environment.

13.4 SQL Server Configuration Tuning

- Set Fill Factor to 95 for balanced performance.
- Enable online and resumable index rebuilds (ELEVATE_ONLINE and ELEVATE_RESUMABLE ON) to reduce maintenance impact.
- Enable detailed query stats and optimizer

hotfixes (EXEC_QUERY_STATS_FOR_SCALAR_FUNCTIONS, FORCE_SHOWPLAN_RUNTIME_PARAMETER_COLLECTION, QUERY_OPTIMIZER_HOTFIXES ON) for better troubleshooting and plan quality.
- Set MAXDOP to 6 at the database level and tune at query level to optimize parallelism without CPU contention.
- Enable OPTIMIZE_FOR_AD_HOC_WORKLOADS to reduce plan cache bloat.

13.5 Index and Table Optimization

- Create clustered indexes on all heap tables to reduce fragmentation and improve IO efficiency.
- Implement regular index maintenance: rebuild indexes with fragmentation >30%, reorganize those with moderate fragmentation. We did not observe any maintenance job in Azure portal.
- Identify and remove duplicate indexes to reduce write overhead and storage.

13.6 Archival Plan and Data Management

- Implement a data archival and purging strategy to reduce the volume of data actively queried, thereby reducing IO and CPU load.
- Archive historical or infrequently accessed data to cheaper storage or separate databases.

13.7 Reduce ASYNC_NETWORK_IO Waits

- Optimize client applications to consume query results promptly and efficiently, avoiding delays in reading data from SQL Server
- Improve network infrastructure to reduce latency and increase bandwidth between application and database servers.

13.8 Reduce SOS_SCHEDULER_YIELD Waits

- Identify and tune CPU-intensive queries by analyzing execution plans for expensive operations like table scans, sorts, or scalar functions.
- Adjust MAXDOP settings Query level to optimize parallel query execution and reduce CPU contention.
- Rewrite or optimize queries to use indexes effectively and avoid unnecessary CPU work.

13.9 Reduce PAGEIOLATCH_SH Waits

- **Create clustered indexes on heap tables** to reduce fragmentation and improve data locality.
- **Regularly rebuild or reorganize fragmented indexes**, especially those above 30% fragmentation, to reduce IO overhead.
- **Optimize queries to reduce physical reads** by:
  - Selecting only necessary columns (avoid SELECT *)
  - Adding appropriate WHERE clauses and filters
  - Using covering indexes to satisfy queries without key lookups

13.10 Index and Statistics Maintenance Strategy

- Implement a scheduled index and statistics maintenance plan using Ola Hallengren's scripts. Perform daily maintenance on frequently used, highly fragmented tables—rebuilding indexes when fragmentation exceeds 30%, reorganizing when between 10–30%, and updating statistics. On a weekly basis, extend this to all large or high-activity tables, using lower thresholds (rebuild >20–25%, reorganize 10–20%) to ensure consistent performance and reduce query inefficiencies. Use ONLINE and RESUMABLE options where applicable.

13.11 Table Partitioning
- Partition the large dbo.AttendanceSummary table based on tenant_id to improve query performance and reduce maintenance time.

13.12 Page Compression

Implement page compression for **AttendanceSummary(852 GB)** table where performance analysis shows high storage usage and significant I/O activity.

13.13 Upgrade Database Compatibility Level to SQL Server 2019

The current database compatibility level is set to SQL Server 2016, which will reach end of mainstream support. It is strongly recommended to upgrade the compatibility level to SQL Server 2019 to leverage significant improvements in query optimization, performance, and security.

14.Task Prioritization

Tasks	Effort	Priority
Optimize top 5 CPU, Logical Reads, Duration intensive queries **Note: Attached the query details with execution plan**	High	High
Enable page compression on AttendanceSummary	High	High
Consider migration to SQL Server on Azure VM	High	High
Adopt multi-tenant architecture using Elastic Pools	High	High
Optimize client applications to reduce ASYNC_NETWORK_IO waits	High	High
Partition large tables (e.g., AttendanceSummary by tenant_id)	High	High
Enable database-scoped configurations	Low	High
Set MAXDOP to 4 at database level & Tune Query level as per requirement	Low	High
Remove duplicate indexes	Medium	High
Implement Ola Hallengren maintenance scripts	Medium	High
Upgrade compatibility level to 150 (2019)	Medium	High
Use existing HA configuration for read scale-out	Medium	High
Create clustered indexes on heap tables	Medium	Low