# Microsoft SQL Server database

Like | Updated March 1, 2019 by Jay.Manaloto | Tags: database_performance, performance, sql_server, sql_server_performance, sql_tuning, system_performance Add or remove tags

Edit | Page Actions

| Performance | Performance Best Practices | Performance Decision Tree | Performance Analyzer | Workflow Analysis Utility |
|---|---|---|---|---|

*Best Practices for System Performance.*

**5 Database Server Tuning and Maintenance (continued)**



## 5.5 Microsoft SQL Server Database

### *5.5.1 Microsoft SQL Server Database Server Tuning*

IBM outlines recommendations for running TRIRIGA on a Microsoft SQL Server database. In addition to this wiki page, see the following IBM Support blog entry: **IBM TRIRIGA best practice recommendations for a Microsoft SQL Server database**.

### *a. Server and Memory Considerations*

IBM strongly recommends a **dedicated** server for the TRIRIGA database when using Microsoft SQL Server. Compared to other database platforms, Microsoft SQL Server was found to require up to **twice** the memory resources to achieve the same level of performance as other database platforms. Thus, a large memory allocation is crucial when choosing Microsoft SQL Server.

### *b. Snapshot Isolation*

Configure the database to allow **read committed isolation** to reduce blocking:

```
ALTER DATABASE <dbname>
SET ALLOW_SNAPSHOT_ISOLATION ON
ALTER DATABASE <dbname>
SET READ_COMMITTED_SNAPSHOT ON
```

For more information, see (1) Snapshot Isolation in SQL Server, (2) Row Versioning-based Isolation Levels in the Database Engine, and (3) Using Row Versioning-based Isolation Levels.

### *c. Implicit Conversions*

When SQL Server tries to join on or compare fields of different data types, if they are **not** the same data type, it will convert one to match the other. This is called **implicit conversion**. An implicit conversion is not desired in SQL Server, and can lead to poor performance due to SQL Server not using indexes optimally. For more information, see decimal and numeric (Transact-SQL).

If there are implicit conversions, the plans generated may still be cached in SQL Server. The following SQL will show the plans with the implicit conversions. The results will be shown in the query results. If you click on the XML link, you can search for and observe the implicit conversion. If you observe implicit conversions, notify IBM Support for further assistance.

```
--Clear Proc Cache, do not run in production unless you intend to wipe out cached plans. This
will require any new plans to recompile as they come in.
--DBCC FREEPROCCACHE
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

DECLARE @dbname SYSNAME
SET @dbname = QUOTENAME(DB_NAME()) ;

WITH XMLNAMESPACES
     (DEFAULT 'http://schemas.microsoft.com/sqlserver/2004/07/showplan')
SELECT  stmt.value('(@StatementText)[1]', 'varchar(max)') ,
t.value('(ScalarOperator/Identifier/ColumnReference/@Schema)[1]',
'varchar(128)'),
t.value('(ScalarOperator/Identifier/ColumnReference/@Table)[1]', 'varchar(128)'),
t.value('(ScalarOperator/Identifier/ColumnReference/@Column)[1]',  'varchar(128)'
), ic.DATA_TYPE AS ConvertFrom , ic.CHARACTER_MAXIMUM_LENGTH AS ConvertFromLength
, t.value('(@DataType)[1]', 'varchar(128') AS ConvertTo
,    t.value('(@Length)[1]', 'int') AS ConvertToLength , query_plan
```

```
FROM   sys.dm_exec_cached_plans AS cp
       CROSS APPLY sys.dm_exec_query_plan(plan_handle) AS qp
       CROSS APPLY
query_plan.nodes('/ShowPlanXML/BatchSequence/Batch/Statements/StmtSimple')
       AS batch ( stmt )
       CROSS APPLY stmt.nodes('.//Convert[@Implicit="1"]') AS n ( t )
       JOIN INFORMATION_SCHEMA.COLUMNS AS ic ON QUOTENAME(ic.TABLE_SCHEMA) =
t.value('(ScalarOperator/Identifier/ColumnReference/@Schema)[1]',
'varchar(128)') AND QUOTENAME(ic.TABLE_NAME) =
t.value('(ScalarOperator/Identifier/ColumnReference/@Table)[1]',
'varchar(128)') AND ic.COLUMN_NAME =
t.value('(ScalarOperator/Identifier/ColumnReference/@Column)[1]',
'varchar(128)')
WHERE
t.exist('ScalarOperator/Identifier/ColumnReference[@Database=sql:variable("@dbname")]
[@Schema!="[sys]"]') = 1
```

### *d. Sparse Columns*

Microsoft SQL Server has a limitation for row size at about **8060** bytes of data. This applies to data that is stored in the row. Most VARCHAR/NVARCHAR data is kept off row (just a pointer is stored on the row).

Some users have had issues storing data because the row data is too big on specific business object tables (e.g. T tables like T_TRIREALESTATECONTRACT, T_TRICAPITALPROJECT and T_TRIBUILDING). Users see the dreaded "Cannot Create a row of size NNNN which is greater than the allowable maximum row size of 8060" in the **server.log** and they cannot save their record. To remedy the situation, you must analyze the business object and delete unused fields.

Another alternative you can use is **sparse columns**. Sparse column support was introduced in SQL Server 2008. A sparse column is an ordinary column that has been optimized for **null** value storage. Null value storage is optimized at the expense of value storage; a sparse column with a null value takes up no storage space at all. However, if the column has a value, 2-4 extra bytes over the value size are required to save the field value. There is a trade-off and the ratio of non-null to null values needs to be significant for any benefit. Microsoft suggests **not** using sparse columns unless the space saved is at least 20-40%. There is also a cost when reading non-null values from sparse columns; table operations including this column may require more processing. That being said, depending on the data being stored, this could be an option for making the row sizes smaller.

IBM TRIRIGA Application Platform 3.5.x does **not** support sparse columns out of the box. To make a sparse column, you would need to work **outside** of TRIRIGA directly in the database. You also need to **avoid** publishing the business object as this could make the field revert to non-sparse. Use sparse columns **sparingly** as they could have a performance impact to your system depending on the data. For more information, see **Use Sparse Columns**.

## *5.5.2 Microsoft SQL Server Application Platform Indexes*

The following performance tuning indexes can be added to SQL Server databases that are running TRIRIGA Platform 3.5.x. These recommended SQL Server performance tuning indexes are the result of iterative performance tuning cycles and collaboration with the TRIRIGA development team. The indexes listed here are **not** included in the TRIRIGA base product unless otherwise stated.

These indexes provide significant performance improvements when measured against a broad performance test workload. While TRIRIGA recommends adding these indexes to the SQL Server database platform, performance gains might vary depending on an array of factors including application usage, load patterns, hardware sizing, application, database server configuration, and so on. SQL Server database administrators should monitor databases for efficient index usage to determine the overall impact produced by applying the recommended indexes, and to determine additional indexes that will improve performance based on situational and data composition needs.

**Notes:**

- Note that these indexes are based on **out-of-the-box** SQL queries. These may need to be altered to account for custom columns or other customizations that would alter the out-of-the-box query to which the index pertains.
- In addition, Microsoft SQL Server imposes different **restrictions** on the size of indexes depending on the version in use. If you try to apply these indexes and receive a warning about the length of the index, you may need to remove columns from the end of the recommended index to achieve an index size that will work for the version of Microsoft SQL in use. Multi-byte character sets will be especially vulnerable to these restrictions.

```
CREATE INDEX [PERF_APP_OBJECT_PERMISSION1] ON [APP_OBJECT_PERMISSION] ([APPLICATION_ID],
[TEMPLATE_ID], [TAB_ID], [SECTION_ID], [FIELD_ID],[SERVICE_ID], [GROUP_ID])
GO


CREATE INDEX [PERF_APP_OBJECT_PERMISSION2] ON [APP_OBJECT_PERMISSION] ([TAB_ID], [SECTION_ID],
[FIELD_ID],[TEMPLATE_ID], [SERVICE_ID], [GROUP_ID])
GO


CREATE INDEX [PERF_BUDGET_CODES1] ON [BUDGET_CODES] ([STATUS], [CODE_REF_ID], [TRANSACTION_ID])
GO


CREATE INDEX [PERF_BUDGET_CURRENCIES1] ON [BUDGET_CURRENCIES] ([TRANSACTION_ID],
[CURRENCY_CODE], [AMOUNT])
GO


CREATE INDEX [PERF_BUDGET_TRANSACTION1] ON [BUDGET_TRANSACTION] ([TRANSACTION_TYPE],
[REVERSE_FLAG], [SYSTEM_DATE])
GO


CREATE INDEX [PERF_BUDGET_TRANSACTION2] ON [BUDGET_TRANSACTION] ([OBJECT_ID], [REVERSE_FLAG])
INCLUDE ([TRANSACTION_ID], [TRANSACTION_TYPE], [DESCRIPTION], [TRANSACTION_DATE],
[SYSTEM_DATE], [COMPANY_ID], [PROGRAM_ID], [PROJECT_ID], [BO_ID], [OBJECT_VERSION],
[MODULE_ID], [ORGANIZATION_ID], [GEOGRAPHY_ID], [USER_ID], [REF_OBJECT_ID],
[REF_OBJECT_VERSION], [REF_MODULE_ID], [REF_BO_ID], [REVERSE_DATE], [LOCATION_ID])
GO


CREATE INDEX [PERF_GROUPMEMBER1] ON [T_GROUPMEMBER] ([PAR_SPEC_ID], [MEMBERTYPE],
[SYS_OBJECTID]) INCLUDE ([MEMBERID])
GO


CREATE INDEX [PERF_GUI_HEADER_PUBL1] ON [GUI_HEADER_PUBL] ([GUI_NAME])
GO


CREATE INDEX [PERF_GUI_HEADER_PUBL2] ON [GUI_HEADER_PUBL] ([SPEC_CLASS_TYPE],[ALT_PRINT_FORM])
GO


CREATE INDEX [PERF_IBS_SPEC1] ON [IBS_SPEC] ([ROOT_FLG]) INCLUDE ([SPEC_CLASS_TYPE], [SPEC_ID])
```

```
GO


CREATE INDEX [PERF_IBS_SPEC_TYPE1] ON [IBS_SPEC_TYPE] ([COMPANY_ID], [SPEC_CLASS_TYPE],
[DELETED_FLAG])

GO


CREATE INDEX [PERF_IBS_SPEC_TYPE2] ON [IBS_SPEC_TYPE] ([DELETED_FLAG], [EXT_MANAGED],
[PASS_THROUGH_FLAG], [SHOW_IN_MANAGER], [SPEC_CLASS_TYPE])

GO


CREATE INDEX [PERF_IBS_SPEC_TYPE3] ON [IBS_SPEC_TYPE] ([NAME], [DELETED_FLAG])

GO


CREATE INDEX [PERF_IBS_TEMP_SPEC_ASSIGNMENTS1] ON [IBS_TEMP_SPEC_ASSIGNMENTS] ([RAND_NO],
[SPEC_ID])

GO


CREATE INDEX [PERF_IBS_TEMP_SPEC_ASSIGNMENTS2] ON [IBS_TEMP_SPEC_ASSIGNMENTS] ([RAND_NO],
[SPEC_ID], [ASS_TYPE]) INCLUDE ([ASS_SPEC_ID], [ACTION])

GO


CREATE INDEX [PERF_LIST_VALUE1] ON [LIST_VALUE] ([LIST_ID], [COMPANY_ID], [LANGUAGE_ID])

GO


CREATE INDEX [PERF_ORGANIZATION1] ON [T_ORGANIZATION] ([SYS_OBJECTID], [SYS_GUIID]) INCLUDE
([TRISTATUSCL])

GO


CREATE INDEX [PERF_ORGANIZATION2] ON [T_ORGANIZATION] ([SYS_OBJECTID], [SYS_GUIID]) INCLUDE
([SPEC_ID], [SYS_TYPE1], [TRIIDTX], [TRINAMETX], [TRISTATUSCL], [TRIPATHTX], [TRISHORTNAMETX],
[TRIORGTYPECL], [TRIORGTYPECLOBJID])

GO


CREATE INDEX [PERF_ORGANIZATION3] ON [T_ORGANIZATION] ([SYS_PROJECTID],[SYS_OBJECTID]) INCLUDE
([SPEC_ID], [SYS_GUIID], [SYS_TYPE1], [SYS_ORGNAME], [SYS_ORGNAMEOBJID], [TRIIDTX],
[TRINAMETX], [TRISTATUSCL], [TRIFORMLABELSY])

GO


CREATE INDEX [PERF_RESOURCE_AVAILABILITY1] ON [RESOURCE_AVAILABILITY] ([SPEC_ID], [TASK_ID])

GO


CREATE INDEX [PERF_SCHEDULEDEVENTS1] ON [T_SCHEDULEDEVENTS] ([EVENTSTATUS],[SYS_OBJECTID],
[ENDDATETIME]) INCLUDE ([SPEC_ID], [SYS_GUIID], [SYS_TYPE1], [STARTDATETIME])

GO


CREATE INDEX [PERF_TRIBUILDING0] ON [T_TRIBUILDING] ([SYS_GUIID], [SYS_OBJECTID])

GO


CREATE INDEX [PERF_TRIBUILDING1] ON [T_TRIBUILDING] ([SYS_PROJECTID],[SYS_OBJECTID]) INCLUDE
([TRIGROSSMAREAMETNU], [TRIGROSSMAREAMETNU_UOM], [TRIGROSSMAREAIMPNU],
[TRIGROSSMAREAIMPNU_UOM], [TRIAREAUO], [TRIBUILDINGCOMMONAREAN], [SPEC_ID], [SYS_GUIID],
[SYS_TYPE1], [TRINAMETX], [TRIUSERMESSAGEFLAGTX], [TRIFORMLABELSY], [TRIPATHTX],
[TRIPARENTPROPERTYTX], [TRIPARENTPROPERTYTXOBJID], [TRIBUILDINGCLASSCL],
[TRIBUILDINGCLASSCLOBJID], [TRINUMBEROFFLOORSNU], [TRIGROSSAREAMETNU], [TRIGROSSAREAMETNU_UOM],
[TRIGROSSAREAIMPNU], [TRIGROSSAREAIMPNU_UOM], [TRILENGTHUO])

GO


CREATE INDEX [PERF_TRIBUILDING2] ON [T_TRIBUILDING] ([SYS_PROJECTID],[SYS_OBJECTID],
[SYS_GUIID]) INCLUDE ([TRINAMETX], [TRISTATUSCL])

GO


CREATE INDEX [PERF_TRIBUILDINGFACT1] ON [T_TRIBUILDINGFACT] ([TRICAPTUREPERIODTXOBJID]) INCLUDE
([TRIFACTCAPITALFIXEDASS], [TRIDIMBUILDINGTENURETXOBJID], [TRIDIMBUILDINGCLASSTXOBJID],
[TRIDIMLOCATIONTXOBJID], [TRIFACTREPLACEMENTVALU])

GO


CREATE INDEX [PERF_TRIBUILDINGFACT2] ON [T_TRIBUILDINGFACT] ([TRICAPTUREPERIODTXOBJID]) INCLUDE
([TRIFACTMAINTENANCECOST], [TRIDIMBUILDINGTENURETXOBJID], [TRIDIMBUILDINGCLASSTXOBJID],
[TRIDIMLOCATIONTXOBJID], [TRIFACTREPLACEMENTVALU])

GO


CREATE INDEX [PERF_TRIBUILDINGSYSTEMITEMFACT1] ON [T_TRIBUILDINGSYSTEMITEMFACT]
([TRICAPTUREPERIODTXOBJID]) INCLUDE ([TRIDIMBUILDINGTENURETXOBJID],
[TRIDIMBUILDINGCLASSTXOBJID], [TRIFACTREPLACEMENTVALU], [TRIDIMLOCATIONTXOBJID],
[TRIFACTESTIMATEDREPAIR], [TRIDIMBUILDINGSYSTEMCLOBJID])

GO


CREATE INDEX [PERF_TRICAPITALPROJECT1] ON [T_TRICAPITALPROJECT] ([SYS_PROJECTID], [SYS_GUIID],
[SYS_OBJECTID]) INCLUDE ([TRIDATEDA], [SPEC_ID], [SYS_TYPE1], [TRIIDTX], [TRINAMETX],
[TRISTATUSCL], [TRISTATUSCLOBJID])

GO


CREATE INDEX [PERF_TRICAPITALPROJECT2] ON [T_TRICAPITALPROJECT] ([SYS_PROJECTID], [SYS_GUIID],
[SYS_OBJECTID]) INCLUDE ([TRINAMETX], [TRISTATUSCL])

GO


CREATE INDEX [PERF_TRICAPITALPROJECTFACT1] ON [T_TRICAPITALPROJECTFACT] ([TRIDIMUSERIDTXOBJID],
[TRIDIMSTATUSTX], [TRICAPTUREPERIODTXOBJID], [TRIDIMPROGRAMTXOBJID]) INCLUDE
([TRIFACTBUDGETCURRENTAM], [TRIFACTCOMMITMENTCHANG], [TRIDIMPROGRAMTX])

GO
```

```
CREATE INDEX [PERF_TRICAPITALPROJECTFACT2] ON [T_TRICAPITALPROJECTFACT] ([TRIDIMUSERIDTXOBJID],
[TRIDIMSTATUSTX], [TRICAPTUREPERIODTXOBJID], [TRIDIMPROGRAMTXOBJID]) INCLUDE
([TRIFACTCURRENTBUDGETTO], [TRIFACTBUDGETCURRENTAM], [TRIDIMPROGRAMTX])

GO


CREATE INDEX [PERF_TRICAPITALPROJECTFACT3] ON [T_TRICAPITALPROJECTFACT] ([TRIDIMUSERIDTXOBJID],
[TRIDIMSTATUSTX], [TRICAPTUREPERIODTXOBJID], [TRIDIMPROGRAMTXOBJID]) INCLUDE
([TRIFACTORIGINALBUDGETT], [TRIFACTBUDGETORIGINALA], [TRIDIMPROGRAMTX])

GO


CREATE INDEX [PERF_TRICAPITALPROJECTFACT4] ON [T_TRICAPITALPROJECTFACT] ([TRIDIMUSERIDTXOBJID],
[TRIDIMSTATUSTX], [TRICAPTUREPERIODTXOBJID], [TRIDIMPROGRAMTXOBJID]) INCLUDE
([TRIFACTSCHEDULEVARIANC], [TRIFACTCOUNTTOTALNUMBE2], [TRIDIMPROGRAMTX])

GO


CREATE INDEX [PERF_TRICLAUSETYPE1] ON [T_TRICLAUSETYPE] ([TRINAMETX])

GO


CREATE INDEX [PERF_TRIPEOPLE1] ON [T_TRIPEOPLE] ([TRIIDTX])

GO


CREATE INDEX [PERF_TRIPEOPLE2] ON [T_TRIPEOPLE] ([TRIRECORDNAMESY])

GO


CREATE INDEX [PERF_TRIPEOPLE3] ON [T_TRIPEOPLE] ([SYS_GUIID],[SYS_OBJECTID]) INCLUDE
([TRIWORKFAXTX], [TRIWORKPHONETX], [TRIEMAILTX], [TRISTATUSCL], [TRITITLETX],
[PRIMARYORGANIZATIONSYSKEY], [SPEC_ID], [SYS_TYPE1], [TRIUSERMESSAGEFLAGTX], [TRINAMETX])

GO


CREATE INDEX [PERF_TRIPEOPLE4] ON [T_TRIPEOPLE] ([SYS_GUIID],[SYS_OBJECTID], [TRIIDTX])

GO


CREATE INDEX [PERF_TRIPEOPLE5] ON [T_TRIPEOPLE] ([SYS_OBJECTID], [SYS_GUIID]) INCLUDE
([TRIFIRSTNAMETX], [TRILASTNAMETX], [TRISTATUSCL], [PRIMARYORGANIZATIONSYSKEY], [SPEC_ID],
[SYS_TYPE1], [TRIUSERMESSAGEFLAGTX], [TRINAMETX], [TRIIDTX], [TRIFORMLABELSY])

GO


CREATE INDEX [PERF_TRIPEOPLE6] ON [T_TRIPEOPLE] ([TRINAMETX]) INCLUDE ([SPEC_ID])

GO


CREATE INDEX [PERF_TRIPROJECTBUDGETCHANGE1] ON [T_TRIPROJECTBUDGETCHANGE] ([SYS_PROJECTID],
[SYS_GUIID],[SYS_OBJECTID]) INCLUDE ([SPEC_ID], [SYS_TYPE1], [TRIDATEDA], [TRINAMETX],
[TRIIDTX], [TRISTATUSCL], [TRISTATUSCLOBJID], [TRIUSERMESSAGEFLAGTX], [TRIREVISIONNU])

GO


CREATE INDEX [PERF_TRIPROJECTBUDGETCHANGE2] ON [T_TRIPROJECTBUDGETCHANGE] ([SYS_PROJECTID],
[SYS_GUIID],[SYS_OBJECTID]) INCLUDE ([TRISTATUSCL])

GO


CREATE INDEX [PERF_TRIPROJECTORIGINALBUDGET1] ON [T_TRIPROJECTORIGINALBUDGET] ([SYS_PROJECTID],
[SYS_GUIID],[SYS_OBJECTID])

GO


CREATE INDEX [PERF_TRIRECONTRACTFACT1] ON [T_TRIRECONTRACTFACT] ([TRIFACTACCOUNTINGTYPET])
INCLUDE ([TRIDIMORGANIZATIONTXOBJID], [TRIDIMPRIMARYUSETXOBJID], [TRIFACTTOTALCONTRACTRE],
[TRIFACTTOTALCOSTNU])

GO


CREATE INDEX [PERF_TRIREPAYMENTFACT1] ON [T_TRIREPAYMENTFACT] ([TRICAPTUREPERIODTXOBJID])
INCLUDE ([TRIDIMCONTRACTTYPETXOBJID], [TRIDIMPAYMENTTYPETXOBJID], [TRIFACTOUTSTANDINGRECE],
[TRIFACTOUTSTANDINGDAYS])

GO


CREATE INDEX [PERF_TRIREPAYMENTFACT2] ON [T_TRIREPAYMENTFACT] ([TRIDIMCONTRACTADMINISTOBJID],
[TRIDIMISPAIDTXOBJID],[TRICAPTUREPERIODTXOBJID]) INCLUDE ([TRIDIMCONTRACTTYPETXOBJID],
[TRIDIMPAYMENTTYPETXOBJID], [TRIFACTTOTALPAYMENTSNU], [TRISCOREONTIMENU])

GO


CREATE INDEX [PERF_TRISPACE1] ON [T_TRISPACE] ([SYS_OBJECTID], [SYS_GUIID], [TRINAMETX],
[TRIIDTX])

GO


CREATE INDEX [PERF_TRISPACEALLOCATIONFACT1] ON [T_TRISPACEALLOCATIONFACT]
([TRICAPTUREPERIODTXOBJID], [TRIDIMSPACECLASSTXOBJID], [TRIDIMLOCATIONTXOBJID],
[TRIDIMWORKPOINTFLAGLI], [TRIFACTALLOCWORKPOINTS], [TRIFACTALLOCAREAIMPNU])

GO


CREATE INDEX [PERF_TRISPACEALLOCATIONFACT2] ON [T_TRISPACEALLOCATIONFACT]
([TRICAPTUREPERIODTXOBJID], [TRIFACTALLOCMOVESNU], [TRIFACTALLOCWORKERSNU])

GO


CREATE INDEX [PERF_TRISPACEFACT1] ON [T_TRISPACEFACT] ([TRICAPTUREPERIODTXOBJID]) INCLUDE
([TRIDIMGEOGRAPHYTXOBJID], [TRIDIMSPACECLASSTXOBJID], [TRIFACTSPACEAREAIMPNU],
[TRIFACTSPACEALLOCATEDA])

GO


CREATE INDEX [PERF_TRISPACEFACT2] ON [T_TRISPACEFACT] ([TRICAPTUREPERIODTXOBJID]) INCLUDE
([TRIDIMSPACECLASSTXOBJID], [TRIDIMLOCATIONTXOBJID], [TRIFACTSPACEAREAIMPNU],
[TRIFACTSPACEALLOCATEDA])

GO
```

```
CREATE INDEX [PERF_TRISPACEPEOPLEFACT1] ON [T_TRISPACEPEOPLEFACT] ([TRICAPTUREPERIODTXOBJID])
INCLUDE ([TRIDIMSPACECLASSTXOBJID], [TRIFACTALLOCWORKERSNU], [TRIFACTALLOCAREAIMPNU],
[TRIDIMWORKERTYPETXOBJID], [TRIDIMLOCATIONTXOBJID])

GO


CREATE INDEX [PERF_TRISURVEYFACT1] ON [T_TRISURVEYFACT] ([TRIDIMSURVEYTYPETX]) INCLUDE
([TRIDIMREQUESTCLASSTXOBJID], [TRIFACTRESPONSESCORENU], [TRIFACTMAXIMUMSCORENU],
[TRICAPTUREPERIODTXOBJID])

GO


CREATE INDEX [PERF_TRISURVEYFACT2] ON [T_TRISURVEYFACT] ([TRIDIMSURVEYTYPETX],
[TRICAPTUREPERIODTXOBJID]) INCLUDE ([TRIDIMLOCATIONTXOBJID], [TRIDIMREQUESTCLASSTXOBJID],
[TRIFACTRESPONSESCORENU], [TRIFACTMAXIMUMSCORENU])

GO


CREATE INDEX [PERF_TRITASKDETAILFACT1] ON [T_TRITASKDETAILFACT] ([TRICAPTUREPERIODTXOBJID])
INCLUDE ([TRIDIMTASKTYPETXOBJID], [TRIFACTPREVENTIVETASKC], [TRIFACTPREVENTIVETASKS],
[TRIDIMLOCATIONTXOBJID])

GO


CREATE INDEX [PERF_TRIWORKTASK1] ON [T_TRIWORKTASK] ([SYS_PROJECTID], [SYS_GUIID],
[SYS_OBJECTID]) INCLUDE ([TRIMATRIXSERVICECLAS], [TRIMATRIXSERVICECLASOBJID],
[TRIWORKINGLOCATIONTX], [TRIWORKINGLOCATIONTXOBJID], [SPEC_ID], [SYS_TYPE1], [TRINAMETX],
[TRIUSERMESSAGEFLAGTX], [TRIIDTX], [TRISTATUSCL], [TRISTATUSCLOBJID], [TRIACTUALPERCENTCOMP],
[TRIACTUALPERCENTCOMP_UOM], [TRIACTUALENDDT], [TRIACTUALSTARTDT], [TRIPLANNEDSTARTDT],
[TRIPLANNEDENDDT])

GO


CREATE INDEX [PERF_WEB_LABEL1] ON [WEB_LABEL] ([APPLICATION_ID], [BO_ID]) INCLUDE
([LANGUAGE_ID], [LABEL_NAME], [LABEL_VALUE], [UPDATED_BY], [UPDATED_DATE])

GO


CREATE INDEX [PERF_WEB_MESSAGE1] ON [WEB_MESSAGE] ([LANGUAGE_ID], [USE_NAME])

GO


CREATE INDEX [PERF_WF_EVENT_HISTORY1] ON [WF_EVENT_HISTORY] ([COMPLETED_DATE])

GO


CREATE INDEX [PERF_WF_TEMPLATE1] ON [WF_TEMPLATE] ([STATUS_ID], [TEMPLATE_FLAG],[UPDATED_DATE])
INCLUDE ([WF_TEMPLATE_ID], [WF_TEMPLATE_VERSION])

GO
```

### *5.5.3 Reserve Indexes for SQL Server*

Any implementation of **Reserve** should be tuned to include appropriate indexes for performance improvement. The following indexes were identified to help increase performance dramatically for reserve queries by the TRIRIGA performance team on SQL Server, but you should review and tune for your specific implementation.

The following performance tuning indexes can be added to SQL Server databases that are running TRIRIGA Platform 3.5.x. These recommended SQL Server performance tuning indexes are the result of iterative performance tuning cycles and collaboration with the TRIRIGA development team. The indexes listed here are **not** included in the TRIRIGA base product unless otherwise stated.

These indexes provide significant performance improvements when measured against a broad performance test workload. While TRIRIGA recommends adding these indexes to the SQL Server database platform, performance gains might vary depending on an array of factors including application usage, load patterns, hardware sizing, application, database server configuration, and so on. SQL Server database administrators should monitor databases for efficient index usage to determine the overall impact produced by applying the recommended indexes, and to determine additional indexes that will improve performance based on situational and data composition needs.

**Notes:**

- Note that these indexes are based on **out-of-the-box** SQL queries. These may need to be altered to account for custom columns or other customizations that would alter the out-of-the-box query to which the index pertains.
- In addition, Microsoft SQL Server imposes different **restrictions** on the size of indexes depending on the version in use. If you try to apply these indexes and receive a warning about the length of the index, you may need to remove columns from the end of the recommended index to achieve an index size that will work for the version of Microsoft SQL in use. Multi-byte character sets will be especially vulnerable to these restrictions.

```
CREATE INDEX [PERF01_TRIRESERVATIONINSTANCE] ON [T_TRIRESERVATIONINSTANCE]
([triPlannedStartDT],[SYS_OBJECTID],[SYS_GUIID],[SYS_PROJECTID])

GO


CREATE INDEX [PERF01_TRIRESERVATIONRESOURCE] ON [T_TRIRESERVATIONRESOURCE] ([SPEC_ID],
[SYS_OBJECTID])

GO


CREATE INDEX [PERF03_TRIPEOPLE] ON [T_TRIPEOPLE] ([SPEC_ID],[SYS_OBJECTID])

GO


CREATE INDEX [PERF01_MYPROFILE] ON [T_MYPROFILE] ([SPEC_ID],[SYS_OBJECTID])

GO


CREATE INDEX [PERF01_TRIRESERVATIONDEF] ON [T_TRIRESERVATIONDEFINITION] ([SPEC_ID],
[SYS_OBJECTID],[SYS_GUIID],[SYS_PROJECTID])

GO


CREATE INDEX [PERF01_TRIROLE] ON [T_TRIROLE] ([SPEC_ID],[triNameTX])

GO


CREATE INDEX [PERF01_TRICONTACTROLE] ON [T_TRICONTACTROLE] ([SPEC_ID],[SYS_OBJECTID],
[ClassifiedByRoleSysKey])

GO
```

### *5.5.4 Lease Indexes for SQL Server*

Performance benchmark testing for **Lease** was performed on the DB2 database platform. However, the findings from that platform may also be applicable to Microsoft SQL Server. Your database administrator can take the identified queries from the

DB2 results and use the index adviser for your database platform to see what indexes are recommended on that platform.

**Notes:**

- The DB2 results in this wiki are based on **out-of-the-box** queries and does not take into account any additional columns that may be in your deployment. For more information, see the **5.3.4 Lease Indexes for DB2**.

Next >

**Comments (0)** | Versions (9) | Attachments (0) | About

*There are no comments.*

Add a comment

Feed for this page | Feed for these comments