NetRom

**Standard**

T-SQL Code Conventions

Version 2.0

12-Oct-2013



**T-SQL Code Conventions**

Standard

Caracal Blvd 178

200746, Craiova, Dolj

Romania

tel.: +40 251 437 738

fax.: +40 251 436 025

[soft@netrom.ro](mailto:soft@netrom.ro)

[www.netrom.ro](http://www.netrom.ro)

**Revision history**

|  |  |
| --- | --- |
| Title | T-SQL Code Conventions |
| Version | 2.0 Final |
| Subject | Standard |
| Author | NetRom |

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Author | Comments |
| 04-Jun-2009 | 0.1 | NetRom | Initial draft |
| 06-Jul-2009 | 1.0 | NetRom | First Revision |
| 08-Jul-2009 | 1.1 | NetRom | Final version |
| 28-Feb-2011 | 1.2 | NetRom | Update on section 3.7: Old point 5 removed and replaced with points 5 and 6. |
| 17-Mar-2011 | 1.3 | NetRom | Removed Best Practices from this document and created a separate one.  Renamed document (removed Best Practices from title) |
| 10-Dec-2013 | 1.4 | NetRom | Aligned to latest document template. |
| 10-June-2015 | 2.0 | NetRom | Updated dates format |

# Contents

[1. Contents 4](#_Toc374442286)

[2. About this document 5](#_Toc374442287)

[3. Database naming conventions 6](#_Toc374442288)

[4. SQL User Defined Objects Naming 7](#_Toc374442289)

[4.1. Tables 7](#_Toc374442290)

[4.2. Table Columns 7](#_Toc374442291)

[4.3. Store Procedures 7](#_Toc374442292)

[4.4. User defined Functions 7](#_Toc374442293)

[4.5. Views 8](#_Toc374442294)

[4.6. Foreign Keys 8](#_Toc374442295)

[4.7. Default constraints 8](#_Toc374442296)

[5. Indentation 9](#_Toc374442297)

[5.1. Wrap long lines 9](#_Toc374442298)

[5.2. Indentation characters 9](#_Toc374442299)

[6. Comments 10](#_Toc374442300)

[6.1. Database objects 10](#_Toc374442301)

[6.2. Script comments 10](#_Toc374442302)

[7. Blocks 11](#_Toc374442303)

[8. White spaces 12](#_Toc374442304)

[8.1. Blank lines 12](#_Toc374442305)

[8.2. Other spaces 12](#_Toc374442306)

[8.3. Constants 12](#_Toc374442307)

# About this document

This document describes the T-SQL code conventions and standards used within NetRom organization that must be followed for all database projects designed and created by NetRom (exceptions are allowed only for projects where other external standards are imposed by the customer).

# Database naming conventions

Use the following database name format on all NetRom database servers:

**ClientName**.(***ProjectName|Description***).**Purpose**.[Source].[Version].[SpecialReason]

*Note*: The separator should always be “.” but when (and only when) the database engine does not allow this (like for example MySQL) then “\_” can be used as a separator.

Meaning:

* ClientName (mandatory): Name of the NetRom’s client. It is preferred to NOT use “shortcuts”. If the name of the client is too long or composed from at least 2 words, a common sense abbreviation can be used (like the first 3 letters from each word, or even an acronym if the name has too many words). Some allowed abbreviations can be considered the following:
  + Event Software Benelux & Ticketing Software Benelux BV: ESBTSB (*this one really exists – not invented*)
  + IteMedical: IteMed
* ProjectName (mandatory if Description not provided): Name of the project - no abbreviations. Same logic as for ClientName can be applied.
* Description (mandatory if project name is not provided): A description of the database - usually linked to a project part: for example ItMobile.FieldDesk.BackOffice, where BackOffice is the description. You can see the combination *ProjectName.Description* as a single group which is mandatory (formed from two words from which only one or both can appear).
* Purpose (mandatory): This part is mandatory, should indicate the purpose of the database and should be only one of the following list of values:
  + Dev
  + QA
  + Stage
  + Accept
  + PreLive
  + Live
* Source (optional): Should indicate the provenience of the database (but only if other than NetRom). This can be used when for example bringing on our server a database from on the live servers from different customers or NetRom’s client.
* Version (optional): the version of the database/application.
* SpecialReason (optional): can be used when a database for special purposes is created/added on the server. Here you can use words like “Performance”, “UpdateScripts” etc.

# SQL User Defined Objects Naming

Use Camel casing for all User Defined Objects (UDO).

Use suggestive names and no abbreviations.

## Tables

Use ‘tbl’ as prefix for table names.

Primary Key (PK) will include the table name and will end with ‘ID’.

Use table relations as much as possible. This will ensure the quality of data (no orphan records will be possible).

Foreign keys will have the same name as related tables PK.

Use suggestive table naming for table relations. If necessary, tables can be grouped using the same prefix name (group name).

Avoid using null able columns. For varchar columns use as default value the empty value (‘’), not null.

Example:

1. tblEmployee will have the primary key EmployeeID
2. tblEmployeeContracts will have the primary EmployeeContractID and the foreign key EmployeeID which will refer to the tblEmployee primary key.
3. In the example above, the relation between Employee and EmployeeContracts it’s clear: one Employee can have more Contracts (one to many).

## Table Columns

Use Camel casing for all table column definitions.

Use suggestive names and no abbreviations.

Column definition must use only logical names and should not make any reference to the column type.

## Store Procedures

Use following naming standard:

[dbo].[sp<Action>[<Group Name>\_]<table/logical\_instance>[<Description>]]

Example:

[dbo].[spUpdateEmployee]

[dbo].[spGetEmployeeInfo]

[dbo].[spDeleteEmployee]

## User defined Functions

Use following naming standard:

* Scalar valued functions:

[dbo].[sf<Action>[<Group Name>\_]<table/logical\_instance>[<Description>]]

* Table valued functions

[dbo].[tf<Action>[<Group Name>\_]<table/logical\_instance>[<Description>]]

Example:

[dbo].[sfCheckEmployeeName]

[dbo].[tfGetEmployeeContracts]

## Views

Use the following naming standards:

[dbo].[vw[<Group Name>\_]<table/logical instance>[<Description>]]

Example:

[dbo].[vwEmployeeRelatedData]

## Foreign Keys

Use following naming standards:

FK\_TableName1\_TableName2

TableName1 is the main table containing the foreign key and the TableName2 is the name of the table which has as Primary Key the above mentioned Foreign Key.

## Default constraints

Use following naming standards:

DF\_TableName\_ColumnName

It is used to specify default values for table columns.

# Indentation

## Wrap long lines

Declare variables one per line.

SELECT/INSERT/UPDATE columns should be one per line.

WHERE conditions should be one per line.

If more tables are used in FROM clause, each table should be in one line.

Examples:

declare

@FirstName varchar(50),

@LastName varchar(50),

@Initials varchar(20),

@FullName varchar(150)

SELECT

O.[Name] as [OrganizationName],

U.[Name] as [UnitName],

E.[Name] as [EmployeeName]

FROM

[dbo].[tblOrganization] O INNER JOIN

[dbo].[tblUnit] U on

U.[OrganizationID] = O.[OrganizationID] INNER JOIN

[dbo].[tblEmployee] E on

E.[UnitID] = U.[UnitID]

WHERE

O.[Name] LIKE '%Organization%' and

E.[Name] LIKE '%Employee%'

ORDER BY

O.[Name] asc,

U.[Name] asc,

E.[Name] asc

## Indentation characters

Use only tabs for indentation instead of spaces.

Use only one tab per indent level.

# Comments

## Database objects

When creating new objects in database (store procedures, user define functions) fill the description header with right values.

Example:

-- =============================================

-- Author: <Author,,Name>

-- Create date: <Create Date,,>

-- Description: <Description,,>

-- =============================================

For Author Name, use Dev1, Dev2 (generic names assigned to each developer).

Filling the Create Date is useful to keep track of the changes. When modifications are done, add Modify Date and a small description between brackets.

Object description should be clear and can contain information like:

* Detailed action
* Objects affected
* Page(s) / Other Database Objects where is used

## Script comments

Use comments whenever is necessary (for explaining the code or marking modifications).

Write comments in your stored procedures, triggers and SQL batches generously, whenever something is not very obvious. This helps other programmers understand your code clearly. Don’t worries about the length of the comments, as it won't impact the performance.

To add comments in SQL you can use:

* “--” all the text after this, on that line, will be treated as comment
* “/\* \*/” if you need to comment more lines; all the text between those will be treated as comment

Examples:

-- SET NOCOUNT ON added to prevent extra result sets from

SET NOCOUNT ON; -- added to prevent extra result sets from

/\* SET NOCOUNT ON;

added to prevent extra result sets from \*/

# Blocks

All scripts between BEGIN-END keywords will be treated as a block.

When creating local scope (script-block), always indent.

When using IF keyword always use BEGIN-END.

Example:

IF (@EmployeeID is not null)

BEGIN

SELECT

@EmployeeName = [Name]

FROM

[dbo].[tblEmployee]

WHERE

[EmployeeID]=@EmployeeID

END

# White spaces

## Blank lines

Blank lines should be used between logical sections of script.

Blank lines improve readability in general.

## Other spaces

Place a single space after a comma and surround operators with single spaces.

Examples:

SET @index = @index + 1

SELECT

[Name] as [EmployeeName]

FROM

[dbo].[tblEmployee]

WHERE

[EmployeeID] in (12, 13, 14)

## Constants

Though T-SQL has no concept of constants (like the ones in C language), variables will serve the same purpose. Using variables instead of constant values within your SQL statements, improves readability and maintainability of your code. Consider the following example:

UPDATE [dbo].[tblOrders]

SET [OrderStatus] = 5

WHERE [OrderDate] < '2001/10/25'

The same update statement can be re-written in a more readable form as shown below:  
  
DECLARE

@ORDER\_PENDING int

SET

@ORDER\_PENDING = 5

UPDATE [dbo].[tblOrders]

SET [OrderStatus] = @ORDER\_PENDING

WHERE [OrderDate] < '2001/10/25'