Standards/Guidelines Document

Coding standards and guidelines for Database Programming

Contents

[I. Database Object Naming Conventions and Style 3](#_Toc407976219)

[II. Coding Practices 5](#_Toc407976220)

# Database Object Naming Conventions and Style

1. Use Pascal casing type for all objects in SQL:

CREATE TABLE dbo.**P**ayment**T**ransaction

(

**C**ap**C**ontrol**N**umber varchar(32)

)

1. Use Pascal casing for Column name and local variable names in stored procedure:

**C**ap**C**ontrol**N**umber in table PaymentTransaction

@**C**ap**C**ontrol**N**umber in stored procedure GetPaymentDetails

1. Use Pascal casing for stored procedure name :

Exec dbo.**G**et**P**ayment**D**etails

1. Use Pascal casing for Schema name:

**S**cm.Table where **S**cm is schema.

1. Put schema specific objects into respective schemas.
2. Schema provides proficient way of managing the objects.
3. Schema level access can be granted to the User so database security is even more manageable.
4. Use primary key constraint name as :

PaymentTransaction table has CapControlNumber as a primary key in it then the constraint name will look like

**PK\_TableName\_KeyColumnName** (eg. PK\_PaymentTransaction\_CapControlNumber)

1. Use foreign key constraint name as :

HealthPlanCode is a foreign key in the PaymentTransaction table but is a primary key in the HealthPlan table so it should look like

**FK\_FKTableName\_PKTableName\_FKeyColumnName**

(eg. FK\_PaymentTransaction\_HealthPlan\_HealthPlanCode)

1. Use default constraint name as :

In PaymentTransaction table column UpdatedDate has a default constraint with current date in it. so constraint name should look like

**DF\_TableName\_KeyColumnName** (eg. DF\_PaymentTransaction\_UpdatedDate)

***Note: Use same naming convention for all constraints.***

1. Use Pascal casing for function name :

e.g. dbo.**R**eturn**L**ine**I**d()

1. Use descriptive variable names :

* Avoid single character variable names, such as i or t. Use index or count.
* Do not abbreviate words (such as num instead of number).
* Do not use prefix like temp new or old for table creation instead use “Temp” schema
* Do not use temp tables unless extremely required.
* When executing an UPDATE or DELETE statement, use the primary key in the WHERE condition, if possible. This reduces error possibilities.

1. Maintain strict indentation. Do not nonstandard indentation, such as one space. Recommended to use tabs (4 spaces) for alignment.
2. Indent comments at the same level of indentation as the code you are documenting.

Eg. --Below update will eliminate all the records which do not have ID’s

Update Temp.Table

Set IsVoid = 1,

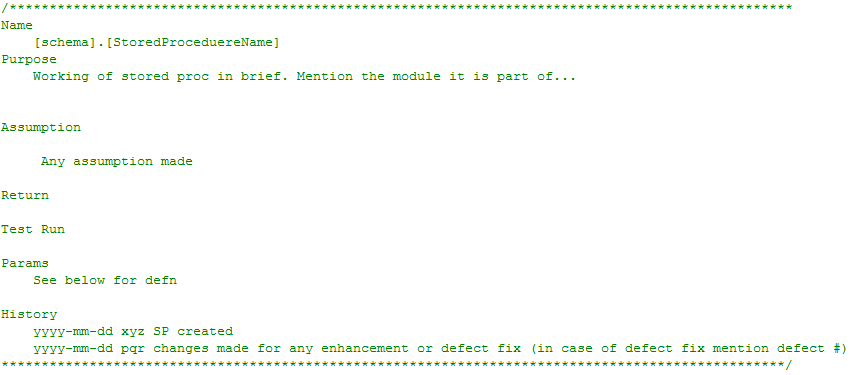
IsRemoved = 1,

Where ID is null or ID = ’’ -- marking void for missing ID’s

Note above the comments & code should align in same way so it can be easily readable.

Do mention if the changes are made for any defect fix or change request. These updates should go in History block of SP header.

1. All comments should pass spell checking. Misspelled comments indicate sloppy development.
2. All the stored procedures & functions should have header which details out attributes like Name, Purpose, Assumptions, Return values, Params, History. See example below.



1. All local variables in stored procedure and function should be declared at the top. (which will be used throughout the procedure)

CREATE PROCEDURE dbo.GetPaymentDetails

@CapControlNumber VARCHAR(12)

AS

BEGIN

DECLARE @IndividualProviderID VARCHAR(14)

,@GroupPorviderID VARCHAR(14)

<………..

…….>

END

1. Declare a local variable in stored procedure or function as close as possible to its first use.

# Coding Practices

1. Write good comments which should detail out your scripts. Give examples where ever necessary.
2. Use Try/Catch for Exception handling.
3. Always use a Begin/End in ”if” statement, even if it conditions a single statement.
4. Avoid using the ternary conditional operator.
5. Never use ”goto” in a stored procedure as it is absolute practice.
6. Always write common code which gets called repeatedly in a function and call that in stored procedure whenever necessary.
7. Try to avoid multiple hits to the table whenever possible i.e. perform table search at once and retrieve the needed information preferabley at one go.
8. Always try to first drop the temporary table and create again and at end of the stored procedure drop the temporary table created*.(only if needed to create one mostly try to find alternative)*

CREATE PROCEDURE dbo.GetPaymentDetails

@CapControlNumber VARCHAR(12)

AS

BEGIN

DECLARE @IndividualProviderID VARCHAR(14)

,@GroupPorviderID VARCHAR(14)

IF OBJECT\_ID('TEMPDB..#TempCCNDetails') IS NOT NULL

BEGIN

DROP TABLE #TempCCNDetails

END

CREATE TABLE #TempCCNDetails

(

CapControlNumber VARCHAR(12)

)

<………..

…….>

DROP TABLE #TempCCNDetails

END

1. Before finalizing the code snippets, always check the execution plan of the code and ensure it has optimal performance.
2. No Code should be checked in unless reviewed by DBA or lead.
3. Developer must write a test harness for complex stored procedures or functions.
4. The scripts should include logging & debugging so that performance measures or debugging the errors will be easier for developer.

