Project Database Rules

Author:
Contributors:
Signed Off:

Table of Contents

Introduction

This document suggests a set of coding guidelines and database naming conventions and rules to be adhered to when writing SQL objects in .

[TYPE YOUR INTRODUCTION TO THE DOCUMENT HERE]

Database object naming conventions

1. Tables

[SPECIFY YOUR PROJECT SPECIFIC NAMING CONVENTIONS. HERE'S A SAMPLE]

- Keep table names short and simple as possible. Do not use numbers or spaces in the names.
- For the *[PRODUCT/PROJECT NAME]* we will keep all table names singular. For example "Customers" should be "Customer" and so on. This is because we don't want to have any confusion as to where to use plural and where not to.

  Prefix :
  The table naming convention in [PRODUCT/PROJECT NAME] will be in this format :

  dbo.MODULE ABBREVIATED NAME_TableName
  OR
  dbo.MODULE ABBREVIATED NAME_TableNames_XREF

  Eg :
  - dbo.PUR_PurchaseOrder
  - dbo.SAL_InvoiceHeader

  Some of the common module abbreviated names could be :
  [LIST ALL THE MODULE ABBRIEVIATIONS FOR YOUR PROJECT]
  - PUR - For Purchase
  - SAL - For Sales
  - INV - For Inventory

  And so on….

  Try to keep the module abbreviation between **2 – 5** characters.

  If the table doesn't belong to a specific module, we could group them into Logical functionalities and have those functional names as a part of the table name.
  Eg:
  - dbo.CORE_XYZZZ

- Notation : Use Pascal case. This reduces the need for underscores to visually separate words in names.
  Eg:
  - dbo.CORE_WorkflowProcessStage, which is more readable compared to core_workflowprocessstage
- If a table name could be long, try to abbreviate the words so that it's understandable by all developers. Over abbreviation is also disadvantageous as developers will not remember table names and have to keep referring the schema to identify tables.
  Eg:

- dbo.CORE_WorkflowProcessStage which is abbreviated for Workflow Process Stage
  This can be further abbreviated to CORE_WFProcessStage (*CLEAR*)
  But NOT CORE_WFPSt (*UNCLEAR*)

- Cross reference tables (XREF tables):
  For cross reference tables, which combines the Unique/Primary keys of 2 or more tables, we should suffix the table with _XREF.
  Eg:
  - dbo.CORE_SupplierLocation_XREF

- *[SPECIFY THE SCHEMAS USED IN YOUR PROJECT]*

2. Columns

[SPECIFY ALL THE COLUMN RULES HERE. HERE'S SOME EXAMPLES]

- Similar to table names, column names should be simple, short and understandable. Avoid abbreviations, acronymns, numbers, spaces and special characters. Column names should be in Pascal Case. No underscores or spaces between words.

- Primary key columns :
  Every table should have a primary key column.

  In *[PRODUCT/PROJECT NAME]* the primary key column name naming convention will either be :

  "Table Name"+ID

  Eg: For Location table . The primary column would be either LocationID.

  [Fix your own naming convention for the primary key column]
  [Fix your own datatype for the primary key column. Specify if it's a GUID, or an integer or a system generated value. Specify if it's an identity and so on.., here's a sample below]

  The column will be an **IDENTITY** column. The column type will be **INT**, so that it can hold 2^32 rows. The reason for using identity columns is that SQL maintains the sequence of newly inserted rows automatically instead of us writing code to generate the next sequence manually.

  [SPECIFY IF THE PK IS CLUSTERED OR NON CLUSTERED BASED ON THE PERFORMANCE TESTS AND YOUR PROJECT USAGE]

- Foreign key columns :
  FK column names should be the "TableName"+ID of the parent table where this column is the primary key.

- All tables will contain these columns : CreatedDate. This will be used for debugging and archiving purposes.
  [Specify any default columns that all your tables must have and what their datatypes are and their usage]

- Data Type Specific Naming :
  - Boolean fields should be given names like "IsHidden", "HasPermission" so that the meaning of the data in the field is not ambiguous.
  - If the field holds date and/or time information, the word "Date" or "Time" should appear somewhere in the field name.

- It is sometimes appropriate to add the unit of time to the field name also, especially if the field holds data like whole numbers ("3" or "20"). Those fields should be named like "WorkedHours" or "FreightMinutes".

3. Indexes

- Since indexes are always related to a table or view, it makes the most sense to use the name of the table or view, as well as the column(s) they index.

  The index naming convention in *[PRODUCT/PROJECT NAME]* will be :

  IDX_TableName_ColumnNames

  For composite indexes that span across multiple columns type the column names in Pascal Case. Underscores are therefore not required in the column names. If there are too many columns, try to abbreviate the columns so that the index name is not too long.
  Try to keep the index name short, maximum around 30-40 characters.

  [SPECIFY ALL THE RULES FOR CHOOSING CLUSTERED/NON CLUSTERED INDEXES.
  MENTION IF YOU ALLOW FILTERED INDEXES AND WHERE THEY SHOULD BE USED]

4. Constraints

- Constraints are at the field/column level so the name of the field the constraint is on should be used in the name. The type of constraint (Check, Referential Integrity a.k.a Foreign Key, Primary Key, or Unique) should be noted also. Constraints are also unique to a particular table and field combination, so you should include the table name also to ensure unique constraint names across your set of database tables.

- Constraints naming convention in *[PRODUCT/PROJECT NAME]* will be :

  ConstraintTypePrefix_TableName_ColumnName

  Eg:
  For SAL_InvoiceHeader
  The primary key should be : PK_InvoiceHeader_InvoiceID

  The ConstraintType Prefixes as as follows :

  - PK - primary key (PK_TableName_ColumnName)
  - FK - foreign key (FK_Table1Name_Table2Name_ColumnName)
  - UN - unique key constraint (UN_TableName_ColumnName)
  - CHK - check constraint (CHK_TableName_ColumnName)
  - DF - default (DF_TableName_ColumnName)

5. Views

- Views follow similar rules that apply to naming tables

- Prefix : As a standard in *[PRODUCT/PROJECT NAME]* table naming convention we will prefix all tables with “vw_”
  [SPECIFY YOUR NAMING CONVENTION HERE]
  The table naming convention will therefore be :

  dbo.vw_MODULE ABBREVIATED NAME_ViewName

If the view combines just 1 or 2 tables then try to add the table names in the view name in Pascal notation.
Eg : A view for PUR_Supplier and Location can be named as vw_PUR_SupplierLocation_XREF

If the view combines several tables then the name can be a understandable and developers should be able to easily understand what the view is meant for.
Eg: vw_SAL_MonthlyStockAndBalanceInfo

6. Rules for tables in [PRODUCT/PROJECT NAME]

Master tables :

[Specify all the rules for the master tables, here some samples below]

- All tables will have a primary key which is of int datatype and usually an identity column of seed = 1 and increment value = 1
  [SPECIFY THE TYPE OF PK YOU WILL BE USING IN YOUR PROJECT]

- For all master records in *[PRODUCT/PROJECT NAME]*, we will never physically delete a row from the table .
  We will always do a soft delete, i.e. there will be a bit column called isInActive which will be updated to 0 for an inactive record.
  [SPECIFY ANY COMMON RULES]

- There will be new columns for XYZ reasons:
  [SPECIFY ALL NEW/DEFAULT COLUMNS THAT GO INTO ALL MASTERS AND THEIR DATATYPES AND USAGE]
  [GIVE EXAMPLES HOW THESE ARE USED, WHEN THEY ARE POPULATED/CHANGED ETC]

All tables :

[SPECIFY COMMON RULES FOR ALL THE TABLES IN YOUR PROJECT]

- Any column that can accept only a Boolean value will have to be a bit datatype. The column name should have “is” prefix. For example, isActive, isUsed etc...

7. Stored Procedures

- Unlike a lot of the other objects described above, stored procedures are not logically tied to any table or column. Typically though, stored procedures perform one of the CRUD (Create, Read, Update, and Delete) operations on a table, or another action of some kind. Since stored procedures always perform some type of operation, it makes sense to use a name that describes the operation they perform. Use a verb to describe the type of operation, followed by the table(s) the operations occur on.

- Prefix :
  As a standard [PRODUCT/PROJECT NAME] procedure naming convention we will prefix all procs with “sproc_”
  The stored procedure naming convention will therefore be :
  [SPECIFY YOUR OWN NAMING CONVENTION HERE]

  dbo.sproc_MODULE ABBREVIATED NAME_ProcedureName

  Do not prefix your stored procedures with something that will cause the system to think it is a system procedure. For example, in SQL Server, if you start a procedure with "sp_", "xp_" or "dt_" it will cause SQL Server to check the master database for this procedure first, causing a performance hit.

The module abbreviated name is similar to the one used above for tables.
The procedure name should be understandable and describe the purpose of the procedure.
Eg:

sproc_CORE_ExecuteWorkflowStage

7. User defined functions (UDFs)

- UDFs follow similar naming convention to stored procedures.

- Prefix :
  The prefix for [PRODUCT/PROJECT NAME] functions will be “UDF_”
  [SPECIFY YOUR OWN NAMING CONVENTION HERE]
  Therefore the naming convention for functions will be :

  dbo.udf_MODULE ABBREVIATED NAME_FunctionName

  For functions that are not tied to a module but rather used as a utility, they can be just named as

  dbo.udf_TaskName

8. Summarising naming conventions

- Limit the name to 30 characters (shorter is better)
- Use only letters or underscores (avoid numbers)
- Try to use underscore characters as little as possible. PascalCase notation achieves the same word separation without them.
- Use a letter as the first character of the name. (don't start names with underscores)
- Avoid abbreviations (can lead to misinterpretation of names)
- Avoid acronyms (some acronyms have more than one meaning eg. "ASP")
- Makes the name readable (they shouldn't sound funny when read aloud)
- Avoid using spaces in names even if the system allows it.

9. Data types used in [PRODUCT/PROJECT NAME]

All primary data types are acceptable. There are rules while choosing a column datatype. These are the ones to be used in [PRODUCT/PROJECT NAME] :
[SPECIFY ALL THE DATATYPES THAT YOU ARE GOING TO USE IN YOUR PROJECT. Here’s a sample list]

Data Type	Storage	Value Range	Purpose
Bit	1 bit	Stores 0, 1 or null	This is used for all Boolean columns. True is converted to 1 and False is converted to 0.
Tinyint	1 byte	0 to 255	Use this if to store a small range of positive whole numbers
Smallint	2 bytes	-32768 to 32767	Stores whole numbers that are positive and negative
Int	4 bytes	-2^31 to 2^31-1	Stores whole numbers that are positive and negative
Bigint	8 bytes	-2^63 to 2^63-1	Stores very large positive/negative whole numbers. [PRODUCT/PROJECT NAME] will use this datatype for identity columns
Float	4 or 8 bytes	-2.23^308 to 2.23^308	Stores large, floating point numbers that exceed the capacity of a decimal data type. Use this datatype in [PRODUCT/PROJECT NAME] to store decimal values as well as columns to hold money values.

datetime	8 bytes	Jan 1, 1753 , through Dec 31 9999	Stores large date and time values
varchar(n)	1-8000 bytes	Maximum of 8000 characters	ANSI data type of variable length. Use this for values that will not hold Non English characters
nvarchar(n)	2-8000 bytes	Maximum of 4000 characters	Unicode data type of variable lengh. Use this data type for columns that could store other languages. But use this datatype cautiously.. as it occupies 2 byes per character
nvarchar(max)	Upto 2 GB	Upto 536,870,912 characters	Unicode variable width data type. Use this to store very large data. NOTE: to be used cautiously. This can now be used instead of text datatype for accepting XML data into stored procedures. SQL is doing away with text datatype in the future.
image	Upto 2 GB		Stores variable-size binary data. To be used cautiously.
XML	Upto 2 GB	XML documents of upto 2 GB	Stores xml documents.

References

http://weblogs.asp.net/jamauss/articles/DatabaseNamingConventions.aspx#Tables
http://www.sql-server-performance.com/articles/dev/sql_best_practices_p3.aspx
http://databases.aspfaq.com/database/what-naming-convention-should-i-use-in-my-database.html
http://searchsqlserver.techtarget.com/tip/0,289483,sid87_gci1277452,00.html?track=NL-417