SQL Server Database Naming Standards

Database naming standards are used to help the DBA and the programmers who will be using that database. By creating standards for the different objects you will create in a database, each person using the database will know which type of object they are dealing with just by looking at the name of the object.

## Overview

In addition to programming standards, you may wish to define some standards for naming the objects in your database. This will help you identify particular objects when you are reviewing your source code without having to look at a printed data dictionary. Below are some naming standards that can be used with any database system. Of course some of these standards may need to be modified depending on limitations of your particular database. If you use another database system, you may have to change, delete or add additional information for the types of objects used in that database system.

## Database Naming Conventions

Name your database something that will make sense to you and your programmers. Try to avoid cryptic names with numbers, as these are not easily recognized by anyone who is not familiar with your application. Below are some names that would be good database names.

Customer

Invoice

GeneralLedger

## Table Naming

Define your tables using a “singlar” name, not plural. For example, use **Employee** or **Customer**, not **Employees** or **Customers**. This makes it easier to create a class that holds a single Employee or Customer. Then you can create a collection class using Employees or Customers for the name or EmployeeCollection or CustomerCollection.

### Primary Key Definition

It is recommended you use a uniqueidentifier (GUID) data type as your primary key for each table. This will be a unique id that the user never sees, but will be used internally to uniquely identify each row. It will cut down on the amount of fields in your WHERE clause. From a performance standpoint this is very efficient since most database systems perform JOINs on integer types quicker than any other type.

Using uniqueidentifiers is useful when your database will need to be combined with other databases for a corporate roll-up for example. Using a unique identifier ensures that no records from the different databases ever have the same identifier.

### Common Columns

Each table in your system should contain columns that will allow you to track who inserted a record as well as who made the last update and at what time and date that insert/update was performed. The field names below are put into each of the tables in a database.

InsertName

InsertDate

UpdateName

UpdateDate

## Schema Usage

For large databases you should break up the tables into schemas. Group your tables by the sub-system in which the table is modified directly by the application. Below are some examples of schema names:

Sales

Payables

Receivables

Audit

Obviously there will be times when a table will be used in more than one module. In that case you need to make a decision about which schema the table should be placed into. Either choose one module or the other, or maybe make a new prefix to identify those tables that are used in more than one module. A good example of this, are tables that hold validation codes such as states, and types. You may wish to create a schema named **Maintenance** or **Validation** for these tables.

## Temporary or Work Tables

Sometimes you will need to create some temporary, or work tables for use in your application. These are not temporary tables you create in a stored procedure. These are tables that are physically created. Temporary/work tables should always be created in a separate schema named “temp”. It is highly recommended that these tables be created in a separate filegroup (if using SQL Server) so you can move these to a different physical controller and disk.

temp.Customers

temp.Employees

temp.Orders

## Column Naming

Name your columns with good long names. Avoid abbreviations. Use Pascal Case for your column names. Do NOT use all lower or all upper case. Here are some examples:

EmployeeID

FirstName

LastName

Street1

## Index Naming

Indexes should end with a suffix of **Index**. The name of the index should be representative of the fields that are contained within the index.

LastNameIndex

FirstLastNamesIndex

CustomerIdIndex

## Stored Procedure Naming

### Modification Stored Procedures

Stored procedures should begin with the main table that they are affecting followed by what they are doing

{TableName}Insert

{TableName}Update

{TableName}Delete

{TableName}Register

### Select Stored Procedures

When selecting from a single table use a name that is descriptive of what you are doing.

{TableName}SelectAll

{TableName}Search

{TableName}SearchBy{Something}

### Feature-Specific Stored Procedures

When selecting from a group of tables use a name that is descriptive of what you are doing. Include as many primary tables as makes sense.

{MainTableName}{ChildTableName}Search

{MainTableName}{ChildTableName}SearchBy{Something}

{MainTableName}{ChildTableName}List

**TIP**: Do NOT use a prefix of “sp” or “sp\_”, as this prefix is used for the built-in SQL Server system procedures and may cause confusion.

## Trigger Naming

Triggers should start with the name of the table, and end with a suffix of **Trigger**. In between, identify what the trigger is used for; delete, insert or update.

CustomersDeleteTrigger

CustomersInsertTrigger

CustomerUpdateTrigger

NOTE: We STRONGLY discourage the use of triggers. Only use them if necessary.

## View Naming

Views should end with a suffix of **View**.

CustomerListView

OrdersView

## Database Diagram Naming

Database diagrams should end with a suffix of **Diagram**.

UsersRolesDiagram

CustomerSystemDiagram

## Default Naming

Defaults should end with a suffix of **Default**.

StateCodeDefault

CustomerTypeDefault

## Rule Naming

Rules should end with a suffix of **Rule**.

StateCodeRule

CustTypeRule

## User-Defined Types or Domains

When you create your own user-defined type (Domain), suffix it with **Type**.

FirstNameType

PostalCodeType

StreetType

## User-Defined Functions

When you create your own user-defined functions suffix those with **Function**.

GetMessageFunction

CalculateSquareFootageFunction

## SQL Statements

All SQL keywords and built-in functions should be written in upper case.

SELECT CustomerName FROM dbo.Customer

DO NOT use the following:

select CustomerName from dbo.Customer

## Summary

Database standards can help make the job of the DBA and the application programmer much easier. Prefixes, suffixes and correct naming of tables can make code generation easier as well.