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 either the Microsoft Access database system, Microsoft SQL Server database system or any other database system you choose. 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 an integer data type as your primary key for each table. This will be a unique number 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.

Another option for your primary key is a uniqueidentifier. This 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

ConcurrencyNumber

The last field can be used if you wish to track multi-user access to this row yourself. This might be used in place of a TimeStamp field. If the database system you have does not support TimeStamp, then you could build your own incrementing field. Of course this means you need to fill in this field and update it every time you modify this table.

## 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:

Employee

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 Tables

Sometimes you will need to create some temporary, or work tables for use in your application. Temporary/work tables should always be created in a separate schema. 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.

## 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 begin with a prefix of “idx”. This prefix should be followed by a name that will identify the field or fields contained within this index:

idxLastName

idxCustID

## Stored Procedure Naming

Stored procedures should begin with the prefix such as “usp” or “proc” followed by a name that will identify the purpose of the stored procedure.

uspGetCustomerID  
procGetCustomerID

uspCustomerInsert

procGetNextID

**TIP**: It is recommend you 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 begin with a prefix of “trg” followed by the name of the table, leaving off that tables prefix. They should then be followed by “del”, “ins”, or “upd” for the Delete, Insert and Update trigger respectively.

trgCustomers\_del

trgCustomers\_ins

trgCustomers\_upd

## View Naming

Views should begin with a prefix of “vw”.

vwCustomerShow

vwCustomerInvoices

## Database Diagram Naming

Database diagrams should begin with a prefix of “diag”. Follow this with a good description of the diagram usage.

diagUsersRoles

diagWinFormControls

## Default Naming

Defaults should begin with a prefix of “def”.

defState

defCustType

## Rule Naming

Rules should begin with a prefix of “rul”.

rulState

rulCustType

## User-Defined Types or Domains

When you create your own user-defined type (Domain), you should prefix it with “typ”.

typName

typAddress

## User-Defined Functions

When you create your own user-defined functions prefix those with “udf”.

udfGetMessage

udfCalculateSquareFootage

## SQL Statements

All SQL keywords and built-in functions need to be 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 and correct naming of tables can make code generation easier as well.