cid:image001.jpg@01CA6144.728946B0

***SQL Coding Standards***

***Microsoft Confidential***

***May 7, 2010***

Table of Contents

[Table of Contents 2](#_Toc245547188)

[Document History 3](#_Toc245547189)

[SQL Server Conventions 3](#_Toc245547190)

[1. General Object Naming Rules 3](#_Toc245547191)

[2. Stored Procedure Names 4](#_Toc245547208)

[3. Functions 5](#_Toc245547209)

[4. Table Names 5](#_Toc245547210)

[5. View Names 5](#_Toc245547211)

[6. Indexes 6](#_Toc245547212)

[7. Primary Keys 6](#_Toc245547213)

[8. Foreign Keys 7](#_Toc245547214)

[9. SQL Syntax Conventions 7](#_Toc245547215)

[10. Security 9](#_Toc245547216)

|  |
| --- |
| Document History |

|  |  |  |
| --- | --- | --- |
| **Last Updated By** | **Last Updated Date** | **Comments** |
| Kevin Johnston | 11/9/2009 | Draft |
|  |  |  |
|  |  |  |

|  |
| --- |
| SQL Server Conventions |

## General Object Naming Rules

### Objects shall be named clearly and unambiguously.

### The name shall provide as much information as possible about an object and provide an obvious connection to references in project documentation.

### Use camel case object naming.

### Form names by joining the words without spaces and capitalizing them within the compound:

#### lowerCamelCase where the first letter is left in lowercase should be used. Example: vwCustomerServiceRequest (view).

#### UpperCamelCase (aka PascalCase) where the first letter is capitalized could be used as well. Example: CAPCustomer (table).

### Object names should be singular.

### Example: Customer not Customers (table).

### Do not use special characters or space.

### Restrict names to alphanumeric characters.

### Keep the names relatively short.

### Note: Maximum object name supported by SQL Server is 128 characters.

### Avoid abbreviations.

### If used they should remain consistent throughout a model. When it is necessary to abbreviate due to length restrictions, follow the rules defined in [Appendix A](#_Appendix_A:_Class). Note: only Identifier is abbreviated (to ID) in the current EDW PDM.

### Using common acronyms is acceptable.

### Example: CAPCustomer, ISOCountryCode, etc.

### Don’t use reserved words.

### Refer to SQL Server Books Online for a complete list of reserved keywords (look for: “reserved words”) or check it online: <http://msdn2.microsoft.com/en-us/library/ms189822.aspx>

Example: Database, Table, Order, Current, etc.**Tabs**: Set tabbing in whatever development environment to use a tab size of 4 keeping the tab character rather than inserting 4 space characters.

## Stored Procedure Names

Name stored procedures using the same convention as described above for table, view and column names. Do not use SQL Server stored procedure group numbers. Restrict names to alphanumeric characters. Form names by capitalizing the first letter of each word. Also, list the “object” first, then the “action” so all database procedures, are grouped together when displaying sorted lists from the database.

Example Stored Procedure Names:

* spTravelUserGet
* spTravelUserSrchByName
* spTravelUserVld

Set of recommended verbs:

|  |  |
| --- | --- |
| **Verb** | **Description** |
| Add or Insert | Insert a specific row. |
| Delete | Delete a specific row or set of rows. (Usually based on a column for the table). |
| Get | Select and return a specific row or a set of related rows. (Usually based on a column for the table). |
| List | Select and return all rows for a table. |
| Update | Update a specific row or set of rows. UPDATE ONLY processing is recommended for Set stored procedures. |
| Validate | Perform validation of rows to be operated |
| Log | Log changes (update, insert, delete) to a specific row or a set of related rows. |

## Functions

|  |  |
| --- | --- |
| **Object Type** | **User function** |
| Naming | **fn***ObjectAction* |
| Description | Use fn prefix followed by the “Object” or “Area of Usage” and whenever possible followed by the “action”.  *Note*: User Function names must be unique within the database. |
| Example | fnCustomerAddressExtract (Object = CustomerAddress, Action = Extract)  fnCAPIncidentUpdate (Object = CAPIncident, Action = Update)  fnZIPGet (Object = ZIP, Action = Get) |

## Table Names

|  |  |
| --- | --- |
| **Object Type** | **Table** |
| Naming | *TableName* |
| Description | Don’t use prefix for table names.  To name associative tables, created to resolve many-to-many relationships, use two referenced table names. |
| Example | Customer, ServiceRequest, CustomerServiceRequest |

## View Names

|  |  |
| --- | --- |
| **Object Type** | **View** |
| Naming | **vw***ViewName* |
| Description | Use vw prefix followed by the SQL view name.  *Note*: View names must be unique within the submodel. |
| Example | vwCustomerList, vwServiceRequest, vwCustomerServiceRequest |

## Indexes

|  |  |
| --- | --- |
| **Object Type** | **Index** |
| Naming | **ix[type]***ColumnName(s)* |
| Description | Use ix prefix followed by [optional] index type (c-clustered = ixc, u-unique = ixu, cu-clustered and unique = ixcu) and followed by the “Column Name(s)”. The prefix ix use alone means: non-clustered, non-unique index.  *Note*: The index type is optional since it can be derived from metadata and the type in the name of the index doesn't guarantee the type is accurate.  *Note*: Index names must be unique within the table. |
| Example | ixZIP (index on Column = ZIP)  ixuProductID (unique index on Column = ProductIdentifier) |

## Primary Keys

|  |  |
| --- | --- |
| **Object Type** | **Primary Key** |
| Naming | **ix[type]***ColumnName(s)* |
| Description | Use ix prefix followed by [optional] index type (c-clustered = ixc, u-unique = ixu, cu-clustered and unique = ixcu) and followed by the “Column Name(s)”. The prefix ix use alone means: non-clustered, non-unique index.  *Note*: The index type is optional since it can be derived from metadata and the type in the name of the index doesn't guarantee the type is accurate.  *Note*: Index names must be unique within the table. |
| Example | ixZIP (index on Column = ZIP)  ixuProductID (unique index on Column = ProductIdentifier) |

## Foreign Keys

|  |  |
| --- | --- |
| **Object Type** | **Foreign Key** |
| Naming | **fk***TableNameColumnName(s)* |
| Description | Use fk prefix followed by the “Table Name” and followed by the “Column Name(s)”.  *Note*: Foreign Key names must be unique within the submodel (database). |
| Example | fkCustomerRegionID (Table = Customer, Column = RegionID)  fkCAPIncidentProductCode (Table = CAPIncident, Column = ProductCode) |

## SQL Syntax Conventions

TOBE added: COMMENTS

1. Indent Spacing

Indent blocks of code which are blocked within *BEGIN* and *END* statements:

Example: Code indention for multiple columns and SQL JOIN.

SELECT

A.ColumnName1

,A.ColumnName2

,B.ColumnName3

,B.ColumnNameN

FROM tblTableNameA AS A

INNER JOIN tblTableNameB AS B

ON A.ColumnName1 = B.ColumnName1

INNER JOIN tblTableNameC AS C

ON B.ColumnName2 = C.ColumnName2

WHERE A.ComumnName2 = ...

OR B.ColumnName3 = ...

GROUP BY...

HAVING ...

WHILE (...)

BEGIN

statement1

statement2

END

1. Format of SQL code

Keep all coding within the project consistent. Use the same formatting and indents for all pieces of SQL code source files.

Line up all keywords *SELECT, FROM, WHERE, GROUP BY, HAVING* clauses. Line up all columns in a select clause and indent the AND and OR statements.

Specify column list in INSERT statement

* + Explicitly specifying column list prevents code failure when the table structure is changed.

**Replace**

INSERT dbo.authors

VALUES( 1, ‘last name for author 1’)

With:

INSERT dbo.authors

(au\_id, au\_lname)

VALUES(1, ‘last name for author 1’)

* + Insert Statement with a Column List:

Never rely on the physical order of the columns in making an insert. Always specify the column’s order in the insert clause.

Example: Insert statement with explicit column list.

INSERT INTO Table1 (

ColumnName1

,ColumnName2

,ColumnName3

,ColumnNameN )

SELECT

A.ColumnName21

,A.ColumnName22

,A.ColumnName23

,A.ColumnName2N

FROM Table2 AS A

WHERE ...

Example: Insert statement with value list.

INSERT INTO Table1 (

ColumnName1

,ColumnName2

,ColumnName3

,ColumnNameN )

VALUES (

value1

,value2

,value3

,valueN )

1. *Use Column Prefixes in SELECT Lists*
   * Use prefixes for columns appearing in the SELECT list. It is a general coding best practice that leads to more maintainable applications.

**Replace**

SELECT au\_id, au\_lname FROM dbo.authors

With:

SELECT a.au\_id, a.au\_lname FROM dbo.authors a

## Security

Data is a corporate asset. Security must be designed into applications and databases from the beginning of the design process. Secure databases don’t just happen, and effective security controls rarely happen as an afterthought to a development effort. Security must be made a priority in the design process, even if the design becomes more complex as a result of the security controls. Simplicity of design is not an excuse for lack of security controls.

Toward the objective of ensuring the maximum security possible, the following generalized standards should apply:

1. User access to a server and database should be implemented using only the built-in SQL Server security features (standard, mixed, integrated security, GRANTS to users or db groups etc.). Only authorized users may be granted access to a server and its database(s) using SQL Server’s built-in security features. For example, allowing all users to login to the server and then use non-system (application-specific) tables to validate a user should be prohibited.
2. Database Developer’s should be aliased to DBO in the databases they are developing to ensure unbroken ownership chains.
3. Grants on objects should fully comply with the project’s security requirements. The security requirements should explicitly specify what grants are *and* are not allowed on all objects for users and/or database groups. Grants not identified by the specification/design should not be granted.

Resources: MSIT and related resources.