**SQL Database standards**

**Overview:**

Singular follows a data directed approach to software development and a significant amount of time and emphasis is placed on correct and efficient database design at the onset of any project.

Where possible table relationships and check constraints are built into each database to enforce simple business rules and drive strong data integrity. These rules are also built into the business layer along with more complex business logic that is impossible (or cumbersome) to create within the database.

Referential integrity is integral to relational databases and Singular’s database design principles. Every database is usually fully normalised (2NF – 2nd normal form) however, occasionally we will purposefully de-normalise aspects of a database for speed purposes (common in development of data warehouses). In these scenarios, data integrity is controlled using check queries.

Indexes (both clustered and non-clustered) are utilised to improve query performance on a database. In Enterprise versions of SQL Server, we make use of Indexed Views to enhance the speed of aggregation lookups otherwise, we make use of Table Value Functions. These views and functions form the building blocks with which we can build complex stored procedures within a system.

All database interaction is done through stored procedures which are defined within specific schemas for enhanced security. The use of stored procedures is considered the best way to combat against attacks on the database through an application (such as SQL Injection Attack).

**Standards:**

**Table naming:**

* Give the table a meaningful name
* The table’s name should always be plural as it holds many records
* Where the table name does not adequately describe the data set it holds, write a detailed table description to describe the data that the table holds and give it context
* Table names should not contain spaces e.g. ‘SecurityGroups’ not “Security Groups”
* “Camel case” the table names e.g. ‘SecurityGroups’ NOT ‘securitygroups’
* Avoid abbreviations

**Table design:**

* The primary key column must be the first column in the table
* The primary key column must be an identity column (auto number)
* It must be named as the singular (i.e. not plural) of the table name with ‘ID’ appended to it (Table: ‘Details’; PK: ‘DetailID’)
* The primary key must only be one column
* Composite unique indexes can be used in the cases where the combination of multiple columns needs to be unique. Do not make these columns the primary key to enforce uniqueness
* “Camel case” the field names and use capital letters where appropriate e.g. ‘StockDetailID’ and NOT ‘STOCKDetailId’
* Place meaningful descriptions on all columns that are not self expalantory e.g. StockCount: The amount of stock left in the warehouse NOT StockCount: Count of stock.
* The description on foreign key columns should be. Link to <entity> (where <entity> is the singular (not plural) form of the table name.
* All ID (primary key and foreign keys) fields should be the first columns in any table
* All bit fields should be the last columns in any table

**Relationships:**

* Place relationships on ID columns only
* Use relationships where necessary. Cascade deletes should be used between Parent/Child tables e.g. Use cascade deletes between Stocks and StockDetails. Not between Employees and RaceID

**VarChar (Text) Columns:**

* Where the text fields need to cater for international characters e.g. a foreign language, use nVarChar(n). Else, use varChar(n)

**Bit columns:**

* Should not allow nulls and a default value should be set
* In the cases where a Tri-state field is used, a bit field can be used which allows nulls. The three states will then be NULL, False and True

**Int (Number) columns:**

* Nulls should generally be used on foreign key columns

**DateTime Columns:**

* For dates that have no time components, name the date fields XDate
* For dates with time components name the fields XDateTime

**Decimal/ Money/ Float Columns:**

* Column type depends on the amount of precision needed for data to be stored:
  + For currency information, use the Money data type
  + For percentages use the decimal(18-n, n) data type (where n is the precision required)

**Stored Procedures/ Functions:**

* At the top of each custom stored procedure/function, write your name, the date and give an overview of what the stored procedure/function does
* Throughout the stored procedure/function, comment the logic to explain what each step of the T-SQL aims to do (This adds immense value when other developers have to debug/make changes to your work)

**Common Schemas:**

* **GetProcs** – Used for ‘read’ stored procedures (fetching records from the database)
* **InsProcs** – Used for ‘insert’ stored procedures (inserting new records into the database)
* **UpdProcs** – Used for ‘update’ stored procedures (updating existing data in the database)
* **DelProcs** – Used for ‘delete’ stored procedures (deleting records from the database)
* **RptProcs** – Used for ‘report’ stored procedures (fetching data sets from the database for reporting purposes)
* **CmdProcs** – Used for ‘command’ stored procedures
* **Fn** – Used for functions