WHEEL OF DEATH DATABASE NAMING CONVENTIONS

ABSTRACT

This document sets out the naming conventions to be used within all database projects for Wheel of Death.

Each naming convention listed is followed by an explanation for its usage.

There are many different database naming conventions used in the industry. Wheel of Death has adopted this in-house policy due to its uniformity and self-descriptiveness.

PARTICULARS

1. Camel Casing to be used for all object names, with lowercase first letter

- Examples:
  - tblClientPhoto
  - tblTeamMember

2. Object Entity Names to be singular – NOT plural

- Representative of a single entity instance
- Examples:
  - tblPhoto – NOT tblPhotos
  - vwProcessItem – NOT vwProcessItems

3. Table names to commence with the prefix 'tbl'

- Self-describing in code

4. Stored Procedure names to commence with the prefix 'usp'

- Self-describing in code

5. View names to commence with the prefix 'vw'

- Self-describing in code

6. Index names to commence with the prefix 'idx'

- Self-describing in code

7. Primary Key in all tables to be named 'ID'

Each table must have a 'surrogate' primary key column named 'ID', of type long/big integer. This provides various benefits, including:

- Creation of generic application code elements (typically list pickers and list maintenance forms). This reduces coding effort as every table is guaranteed to have the same field name and type for the primary key.
- Class abstraction in project code. A base table class can be setup with an ID property mapped to the table's primary key (ID). Derived classes can then inherit from this base table class the relevant methods built in to the parent class, rather than re-build it for every table (e.g. Object-Relational Mapper – ORM).
- Simplified SELECT joins with surrogate primary keys than with compound primary keys comprising two or more columns in the key.
- Bridging tables (aka intersection tables/associative entities) must also include a surrogate 'ID' primary key. These tables often evolve over time to contain additional data in additional to foreign keys. Having the **ID** wired up in place in addition to the compound unique key constraint allows the intersection table itself to then be easily referenced by other tables (and processed in project code) in consistent fashion.
- If a non-numeric natural (candidate) primary key exists for that table, nothing is lost – as the natural primary key can still be defined as UNIQUE.

Note: ID will usually be a system-generated column. However, in some cases, where a natural key exists that is numeric and its value is meaningful, the ID column may be defined as a long integer, but not necessarily system-generated.

8. Foreign Key columns named FK[parent-table- root-name]ID

<child table>.FK[parent-table-root-name]**ID**

references **tbl**[parent-table- root-name].**ID** (i.e. its parent table primary key)

- Self-describing and aids code manageability. No need to include **tbl** in the foreign key name, as it is redundant – the context is always a table for a foreign key.

- The trailing **ID** appears redundant, but serves as a visual reminder that the foreign key always references the **ID** primary-key column of the parent table. Moreover, in cases where there are multiple foreign keys from a child table to the same parent table (see section 9 below), the position of 'ID' in the name serves to differentiate the root table name from the relationship descriptor.
- SQL join statements become consistent and self-describing – E.g.:

```
SELECT
        P.*
FROM
        tblUserPhoto UP INNER JOIN
        tblPhoto P ON UP.FKPhotoID = P.ID;
```

9. Multiple Foreign Key columns to the same Parent named FK[parent-table- root-name]ID[relationship descriptor]

Where more than one foreign key exists to the same parent table, the naming convention is as before, but with additional naming to clarify the function of the foreign key.

For example, a child table might have two foreign keys to tblUser – one to indicate which user first created the record and another to indicate which user last edited it. In this case, the table would have two foreign keys, named as follows:

```
FKUserIDCreatedBy
FKUserIDLastEditedBy
```