Database Standards and Conventions
(Version 1.0)

For

iNova

Document Profile

Project iNova

Prepared By : Sirish Kumar

Date Prepared : 30-Jun-06

Targeted Audience :

1. Shaji
2. Balaji
3. Vergis
4. Developers or iNova Project, this includes current developers and developers who shall be associated in the future with the project development and/or subsequent maintenance activities

Table of Contents

1. Introduction

The goal of the Database Standards is to document new databases and applications in a consistent manner across platforms. These methods and requirements should provide a template for better documentation and make it easier to identify objects during daily maintenance and crisis events.

Having a good set of naming conventions for database objects is one of the most vital things to a project. In the long duration of a business, it saves money and time as programmers are transferred internally and don't need to relearn object names. This document enumerates all the database conventions that shall be required for iNOVA.

2. Purpose of the Document

This document serves as a guideline to the database implementers. It is essential that standards be prepared such that database creation, change management is made easier and flexible

3. Need for Standards

The idea of standardization within the database arena is, extremely, essential to software applications. All sorts of standardization activities have been focused on this arena

4. Standards

The standards documented here are based on the lowest common denominator between all databases, keeping in view that the application is required to be compatible with any third party database engine.

5. Table

To have record and table names that are consistent, clear, and descriptive of the data maintained in the record or table. A record or table is a data representation of an entity with an existence in the real world. Containers, Vessels, Enquiry, Customers etc. have an existence, and the application must keep data about each occurrence of those entities. A record or table name should indicate what entity that record or table contains data about.

1. Table names should be as descriptive as possible.

2. Maximum length for a table shall not exceed 30 characters.

3. If a table name requires multiple tokens, separate each token with an underscore.

4. Unlike Stored procedures, Triggers and other database objects, don't prefix or suffix a table object with any qualifiers unless otherwise the tables belong to a generic component. For instance, it is advisable to prefix all workflow related table with 'WF_' only to distinguish that these tables are not specific to any one business process.

5. Each token in the table name shall be in proper case.

6. Avoid using plurals in the table name.

7. A table represents a unique entity. Therefore, the table names shall be in singular form.

8. Avoid using additional tokens like 'Master' in an attempt to make the table name more descriptive.

9. The table name shall, ideally, represent a master or a business activity. Users familiar with business process shall be in a position to identify the difference between 'Invoice' and 'Customer'. The former is a transaction as the name suggests whereas, the later is a master. No additional effort is required to name a customer table as 'Customer_Master'

10. Sometimes there is a need for more than one table to contain data about an entity. For example, we may have a table containing data about Container master information, a table with Container Tracking data, and a table with Container Allocation data. The table name should be expanded. In the above example, we would have the table names: Container, Container Tracking and Container_Allocation.

6. User Defined Data Types

The entire effort to create standards and conventions is to reduce maintenance time – maintenance that calls for database objects. Also, well defined database standards not only cut down the development time but also aid in familiarizing new developers with the nuances of a database system.

Whenever there is a change request to be applied on a column name the developer should be in a position to analyse the nature of the column, in terms of its data type. The UDT, may alternatively, be referred to as a **Column domain Descriptor**.

There is another significant advantage in maintaining a checklist of user defined data types. It creates a standard approach in maintaining consistency as far as the field length and type are concerned. A name field is defined to be Varchar(20) and it shall be uniform across the entire application. Whenever a change is needed to alter the size and type of fields that may pertain to names, the developers shall have minimum difficulty in identifying or locating such fields in the database.

A check list of UDT shall be maintained. The checklist comprises a description of a UDT and a suffix code.

Annexure-I shall elicit the UDT standards proposed for iNova.

7. Columns

To have data element and table column names that are consistent, clear, and descriptive of the data maintained in the element or column. An element or column should have the same name every place it appears in all databases across all platforms. Customers and programmers should not have to guess what name is used in a specific database on a specific platform. Additionally, standard names will improve productivity, facilitate sharing of programmer resource among groups, and provide coherent table joins.

1. Field names again should be in lower case.

2. Max column length = 30 Chars

3. They should be descriptive to their purpose and only use abbreviations for a select few names.

4. If the column name requires multiple tokens, separate them by an underscore.

5. The column name must be clear without the context of its membership in a table

6. Abbreviations for domain specific column names are allowed, anything else, use the entire word(s).

7. Use underscores to separate words.

8. It would make sense in using abbreviating CAN or BL instead of elaborating them to be 'Cargo Arrival Note' or 'Bill of Lading'.

9. Each column must be categorized with a descriptor (class code) identifying the type of data it contains.

10. If a domain ends with a descriptor, e.g., ORGANIZATION_NAME, you may omit the descriptor, since it is included in the column name.

11. Wherever the column name is a non-abbreviated remove all intermediate vowels in each token.

12. Suffix the field name with suitable UDT code indicating the size and type of the column. The code shall be preceded by an underscore.

13. An example for the above mentioned guideline is as follows

14. Assume that there is a need for a field called 'Customer Name', the physical name is
15. 'Cstmr_nm_txt'. Here, the prefix 'txt' indicates that the column type belongs to a UDT text.

8. Stored Procedure

1. The same rules apply here for length of fields.
2. Also if a component is being developed to an application versus the entire application, prefix the stored procedure name with a 2 letter prefix.
3. Begin stored procedures with SP_ (1 underscore) and then follow it up with the component prefix in upper case.
4. Use descriptive names and if your name denotes whether your stored procedure selects, inserts, updates or deletes data.

   Examples:

- SP_WFtrading_partner(WF is the workflow system)
- SP_container_upd
- SP_customer_ins

Best Practices

1. Indentation: Indentation is a primary requirement for writing and editing stored procedures.
2. Each keyword, Token, expression and parameter should be started in a new line.
3. Maintain comments where needed. Each block in the stored procedure should be preceded with suitable comments which explain the functionality of the block they precede.
4. The code lines that form part of a loop or a selection block should be intended by one tab more than the first line.
5. The first few lines of the stored procedure should contain the profile of the stored procedure.

- Name of the author
- Date Created
- Version No
- Summary of the procedure

6. An additional suffix in the stored procedure name shall convey the nature of operation the stored procedure may perform.

7. The following suffices are recommended

   - INS – if the stored procedure is intended to insert a record
   - UPD – if the stored procedure updates one of more tables
   - DEL- if the stored procedure deletes one or more records
   - SEL- if the stored procedure acts as a means to retrieve one or more record sets.

Often it will be required to identify the module to which a stored procedure belongs to. An additional prefix/suffix will be required to do so. However, this approach shall not be encouraged in the current application as it is believed that the database is a central repository, sharable across the application. One or more modules may access the same data store, frequently. Module descriptor, therefore, does not make any meaning here.

9. Triggers

Triggers like stored procedures should designate what happens to the data
i = insert
u = update
d =delete.

Begin all triggers with the letter T and the table name. For example a trigger that inserts into the 'Enquiry' table would be TI_enquiry.

10. Views

Rules guiding the naming of views shall be similar to that of a trigger. Prefix each view with alphabet 'V'.

11. Constraints and References

All constraints should be prefixed by the string 'CHK_' (denoting check). The name of the constraint shall be descriptive enough in defining the functionality. If a Foreign Key is defined on a table prefix the foreign key with 'FK_'. The name of the foreign key shall be same as the table it refers.

Indexes

Create indexes wherever needed. Ideally, all foreign keys should be created as non-clustered indexes, needless to mention that all primary keys are created as clustered indexes. Further optimizations shall be addressed, subsequently, in Database optimization document.

12. Annexure -1 – Column Domain and Class Descriptor

Field Name/Description	Datatype	Prefix
Identity Column - Masters	int	id
Identity Column - Transactions	BigInt	bin
Code - Masters	Varchar(6)	cd
Name - Masters	Varchar(20)	nam
Description	varchar(100)	vc
Transaction Code	Varchar(20)	tcd
Date	Datetime	dt
Revision No	Small Int	si
Quantity	int	qty
Remarks, Notes	varchar(255)	vcr
Boolean fields	bit	bt
Amount	Numeric(6,2)	nc
Container No	varchar(11)	cno
Address	varchar(20)	add
EmailID	varchar(50)	eml
Website	varchar(100)	web
Voyage No, Stowage No	varchar(10)	vn
ZipCode	Varchar(6)	zip
Phone No	varchar(10)	phn
STD, ISD	varchar(6)	std
Mobile	varchar(20)	mob