Database Design Standards

U.S. Small Business Administration
Office of the Chief Information Officer
Office of Information Systems Support

TABLE OF CONTENTS

1. Standards and Conventions

All the names will follow Hungarian naming convention i.e capitalize the first character of all the words. --See Data Naming Standards for description of Words (Prime, Modifier and Class)-

1.1.Database Objects and Data Structures

1.1.1.Databases

Database Definitions:

- A separate database may be considered wherever the level of independence with other systems is low.
- Database names should be brief, generally up to 8 characters.

1.1.2.Tables

- Table name will adhere to Data Administration Naming Standards.
- All table names will be suffix with “Tbl”.

1.1.3.Data Elements

- All column names will adhere to Data Administration Name Standards
- Columns that are used to specify foreign keys in a table will normally have the same name as they do in the referenced table. If there is more than one foreign key relationships between two tables the differentiation between the foreign key columns should be represented by prefixing the column name by a word or phrase describing the relationship implemented.

1.1.4.User Defined Data Types

- Names of the user defined data types should be suffixed by “Ud”

1.1.5.Triggers

- Trigger names should have the following structure
  <table name><type>Trig
  The table name will not include the “Tbl” suffix and type can be any one of the following:
  - “Ins” for insert trigger
  - “Upd” for update trigger
  - “Del” for delete trigger

1.1.6.Rules

- Rule name should have the following structure
  <Descriptive name >Rule

1.1.7.Defaults

- Default name should have the following structure
  <Descriptive name > Dflt

1.1.8.Indexes

- Index name should have the following structure
  <Column name/concatenated column description> XXInd
  | |
  ([U]nique/[N]onunique) <----------| |
  ([C]lustered/[N]on clustered) <----------|

1.1.9.Views

- Name of the view should be a descriptive word followed by the suffix "View". The descriptive word should follow the same naming conventions as table names.

1.1.10.Stored Procedures

Application Stored Procedures (ASP) :

All database operations will be done thru stored procedures and will be called Application Stored Procedures.
- Programmers will code stored procedures.
- Application Stored Procedures are divided into two categories:

Table Level Stored Procedures (TSP) :

- There are four type of operations possible on one table i.e. select, insert, delete, update. For each type of operation there are different SQL statements required in an application e.g.

Suppose in the application we want to display Business Id and Business Name for Business Id = "0000000001" , the SQL statement will be :

```
select BusId , BusNm from BusTbl
where BusId = "0000000001"
```

Also we want to know how many Businesses we have.

```
select count(*) from BusTbl
```

Since in both the examples the base table is BusTbl, both of these SQL statements will be part of one stored procedure with the identifier saying that which one to execute. In case of SQL statements involving multiple tables, programmer will identify the base table. In case of conflict, the table which comes first in alphabetical order will be the base table. E.g. Suppose we want to display SBG #, SicCd, State and the Name of the business for application Number 456783, the SQL statement will be :

```
select SBG#,SicCd,BusNm
from BndAppComnTbl  a, BusTbl b
where  a.BusId = b.BusId
   and App# = 456783
```

Now in this SQL statement "BndAppComnTbl" will be the base table.

All SQL statements of one type and for one table will go into one stored procedure. Always the first parameter in the stored procedure will identify, which batch of SQL statements will be executed.

- Name of the stored procedure for a table will have the following structure
<table name><type>TSP
where in <table name> drop "Tbl" . <type> can be any one of the following :-
-"Sel" for select
- "Ins" for insert
- "Upd" for update
- "Del" for delete
e.g BndAppComnSelTSP will be the name of stored procedure for "BndAppComnTbl" table and "Select" operation.

- During development programmer will submit changes and request for new stored procedures to the Database System Administration by completing a "Stored Procedure Request Form".

Customize Stored Procedures (For Transactions) (CSP) :

- All the Transactions where more then one SQL statements will be involved will use Customize Stored Procedures.

- In Customize stored procedures only Table Level Stored Procedures will be executed.

- Name of the Customize stored procedure will be a suitably coined word followed by the suffix "CSP". The coined word should follow same conventions as table names.

Business Rules Implemented Thru Stored Procedures :

- Name of the stored procedure for business rules will be a suitably coined word followed by the suffix "BSP". The coined word should follow same conventions as table names.

- Only Database Database Administration will write Stored Procedures for Business Rules.

2.Design Data Structures

2.1.Define User Defined Data Types

User defined data types may be used in columns representing
- entities which are widely used in the system.
- special purpose attribute, occurring in a number of tables, and may have same rules, defaults attached.

2.2.Table Design

2.2.1.Basic Relational Design

Before table design, the Entity - Relationship Diagram (ERD) should be complete along with the attributes of all the entities and their relationships.

The **Entities** and the **Relations** of ERD will help in identifying the tables as :

1. Entities : Entities (e.g. item, vendor etc.) identified during logical design lead to one or more tables as follows :

1.1. **Independent Entities** become independent tables with following properties

- Primary key contains no foreign keys.

1.2. **Dependent tables** may be created to eliminate repeating groups. e.g. children of an employee can be placed in a dependent table.

Dependent tables may have following properties:

- Foreign key refer to their corresponding parent table.
- Primary key contains these foreign key and additional columns.
- Foreign key rules
  - Nulls and defaults not allowed.
  - Please refer to integrity rules/triggers.

1.3. **Sub tables** may be created where applicability of a column is for a few rows only.

Sub tables may minimize data redundancy, but may increase join overhead. Sub tables may have following properties:

- Primary key contains a foreign key referring to super table (no additional column).
- Foreign key rules
  - Nulls and defaults not allowed.
  - please refer to integrity rules/triggers.
- A classifying column in super table is recommended because it eliminates frequent subtables lookups.

- Avoid sub tables, where join overhead is prohibitive.

2. Relations : Among Entities relations shall be represented as follows :

2.1 **Many - One relationship :** Place foreign key on many side. e.g. one department may have many employees. It may have following properties.

- Foreign key rules
  - delete restrict usually or delete cascade in rare cases.
  - update restrict
  - also see Integrity rules/triggers.

2.2 **One - One Relationship :** Place foreign key in table with fewer rows. e.g. each department is managed by a manager. It may have the following properties:

- Foreign key rules
  - delete restrict usually or delete cascade in rare cases.
  - update restrict
  - also see Integrity rules/triggers.

2.3 **Many - Many Relationship :** becomes an associative table. e.g. vendor and item. A vendor can supply many items and on the other hand an item can be supplied by many vendors.

- Two foreign keys refer to two related tables
- Primary key is composite of foreign keys, may contain additional columns.
- Foreign key rules
  - Nulls and defaults not allowed
  - delete restrict usually or delete cascade in rare cases.
  - update of foreign key which is part of the primary key is not permitted.
  - also see Integrity rules/triggers.

3. Identify columns and name it as per convention.

- all attributes of the entity or relationship becomes a column.

4. Identify datatype (integer, money, date etc.) and its length as per DATABASE for each column.

5. Identify whether NULL permissible in the column.

6. Identify Primary Key:
   6.1 NULL is not allowed in the primary key.

7. Identify all foreign keys.
   7.1 Domain Contains Primary key values from parent table or NULL.

2.2.2.Normalization

1. After completion of Basic Table Design, the tables have to be normalized to eliminate data redundancy. This process should address the following problems.

- Inconsistent data
- Update Anomalies
- Uncertainty in data maintenance procedures.

Normalization is undertaken in successive steps starting from First Normal Form (1NF) to 2NF to 3NF. Each subsequent form requires that the preceding form is true.

The stages of Normalization are as follows:

1. First Normal Form (1NF): requires that there are no repeating groups in the table. (All underlying domains contain atomic values only) i.e each cell in the table has only one value. By definition only 1NF tables can be defined under RDBMS.

   If there are repeating groups, create a new table for elements of the repeating group.

2. Second Normal Form (2NF) : requires that each column in a table is fully dependent on the full primary key.

   Problems of part dependency will occur in case of associative or dependent tables, where primary key is composed of more than one element.

   To eliminate part dependency, decompose the table.

3. Third Normal Form (3NF): requires that each column in the table is directly dependent on the primary key. Cases where a column is dependent on the primary key, but the dependence is via another column in the table, should be eliminated.

   For example, conveyance allowance may be dependent to an employee (primary key), but is directly dependent on the employee grade, which is another column in the employee table.

   To eliminate indirect dependency, decompose the table.

2.2.3.Design Tuning

Once the tables have been designed and normalized, it is necessary to review the design from a performance point of view.

Performance Tuning of data structure consists of :

- selective denormalization
- Over - Normalization

Selective denormalization : For frequently used queries a join may be causing unacceptable overhead. In such cases, the tables may be denormalized to 1NF or 2NF level to eliminate joins. Denormalization requires prior knowledge of how the data will be used. Denormalization should be last choice for performance tuning.

Denormalisation is possible through following methods:

- **Duplicate columns:** Same columns may be duplicated in other tables where it is referenced, either as exact copies or as summarized/derived data. Triggers should be implemented to ensure integrity.

- **Redefine columns:** By substituting large columns with artificial or contrived columns.

Over Normalization: The process of decomposing a large table which is already in 3NF is called over normalization. We may over normalize to:

- remove unimportant data from table.
- remove long descriptive columns.
- when single row exceeds page size.
- can improve concurrent access to single table when simultaneous, frequent, non overlapping columns, especially when user is updating.

Care should be taken to define proper triggers to ensure integrity.

2.2.4.Physical Design

After completion of design tuning, it is necessary to undertake the physical design to properly implement the logical design on the target system. This process includes:

- defining indices
- defining partitions
- assignment of storage devices

The objective is to achieve optimum performance of

- disk i/o
- disk storage
- CPU Time

Issues of physical performance tuning may be taken up even after application development is complete. These measures do not affect the logical design. They will not affect the integrity or consistency of database in any manner.

Define indices : Before identification of indices it is necessary to understand the queries properly. Availability of indices improves access time, but they consume disk space and also have additional overhead during inserts and updates. So provision of an index is a compromise between these factors :

1. List all access parameters on each column:

- Frequency of usage
  - High, say of daily access
  - medium, say of weekly access
  - low, say of access with lesser frequency
- On-line or batch
- single table Vs joins

- ordered or random

2. Calculate potential size of the index during normal operations.

3. To identify whether an index is to be provided, the first three criteria may be used.

For high frequency on-line and batch applications indexes should always be considered, irrespective of the size.

For moderate access create indexes where index size is below a predefined size, say 1MB.

For infrequent access, index could be created at the time of execution, or not created at all.

Clustered index may be provided where an ordered access is required.

2.3.Define Defaults and Rules

2.4.Define Integrity Rules / Triggers

1. Integrity rules (IR) between tables should be implemented with the help of triggers.

Following IR will have to be identified for all foreign keys :-

For implementation at independent tables :

PK UPDATE CASCADE
PK UPDATE RESTRICT
PK UPDATE NULLIFY

PK DELETE CASCADE
PK DELETE RESTRICT
PK DELETE NULLIFY

For implementation at dependent tables :

FK INSERT CASCADE
FK INSERT RESTRICT

FK UPDATE CASCADE
FK UPDATE RESTRICT

Note: ERWin will not be used to generate triggers Because it updates tables in a sequence, which can be different from what designer has decided.

2.5.Define Views

2.6.Define Business Rules / Triggers

Business rules will be implemented in the database through triggers.

2.7.Define Strategy To Avoid Dead Locks

To avoid dead locks Database Designers will propose the order in which the tables will be updated. It may be alphabetical order or some sequence in which the tables will get updated to avoid dead locks.

3.Programming Guide Lines For Database Programmer

3.1.Avoid Hold Lock

Avoid hold lock. Instead of selecting with hold lock and updating, update first and then select in a transaction.

3.2.Use Templates For Triggers

Triggers will be used to take care of multiple rows updates / deletes / inserts as far as referential integrity is concerned. Use templates to implement Integrity Rules.

3.3.Specify the Column names in the insert statement

In the INSERT statement always specify the column names in which data will be inserted.
e.g. instead of using INSERT into TABLE values (var1,var2,var3)
use INSERT into TABLE (COL1,COL2,COL3) values (var1,var2,var3)

3.4.Avoid Select *

Always specify column names in select statement.

3.5.Use Upper Case For DATABASE Reserved Words.

Database reserved words will be in upper case and should be aligned to improve readability of the code. All source code scripts will use a consistent standard indentation format.

3.6.Never use Print statement in stored procedures/Triggers.

3.7.Avoid Cursors.

4.Back Up Strategy.

Dump database will be taken on weekly basis. Back up to tape/floppy
Dump transaction will be taken on daily basis. Back up to tape / floppy
Dump database after the commands which are not logged.
Dump database of master after updating master database.

5.Directory Structure To Store Design Documents

The directory structure will be like this:

<HOME DIRECTORY>: It may be developers' home directory or may be the directory where all the applications reside.

<PHASE OF DEVELOPMENT>:

Phases	Directory Name
Development	dbdevl
System Test	dbtest
Acceptance Test	dbacpt
Production	dbprod

<PROJECT NAME> : Name of the Project in brief.

<OBJECT TYPE> :

Object Type/Others	Directory Name
Default	dflt
Index	idx
Rule	rule
Stored Procedure	sp
Table	tab
Trigger	trg
User defined Data Type	ud
Data	dat
Documentation	doc
Miscellaneous	msc

Database Object script file names will have following format : <OBJECT TYPE/PURPOSE>.xxx
where xxx will have following format wherever required :

CRE: CREATE
DRP: DROP
UBN: UNBIND
BIN: BIND

Object Type	File Name	Example
Default	dflt.xxx	dflt.cre / dflt.drp / dflt.bin / dflt.ubn
Index	idx.xxx	idx.cre / idx.drp
Rule	rule.xxx	rule.cre / rule.drp / rule.bin / rule.ubn
Stored Prcedure	Name of the SP ([T/C/B]SP is not required)	
Table	tab.xxx	tab.cre / tab.drp
Trigger	Name of the table	
User Defined Data Type	ud.xxx	ud.cre / ud.drp

Database designers will make a batch files to install / remove all the database objects related to the project. The Structure of the batch file will be < install / remove > and will be stored in /<HOME DIRECTORY>/<PHASE OF DEVELOPMENT>/<PROJECT NAME>

Appendix A

Template For Table Level Stored Procedure

```
/* Date          : 1st Nov, 1994
   Programmer    : Deepak Bhargava.
   Remarks       :
   Parameters    : No.   Name          Data Type      Description
                   1     @Identifier   int            Identifies which batch to execute.
                   2     @RetVal       int            No. of rows returned/affected.
                   3     @BusId        char(10)       Business Id
*/
Create Procedure client.dbo.BusSelSP(@Identifier int = 0, @RetVal int output, @BusId char(10) = Null)
as
begin
        /* Variable Declaration */
        declare  @error  int, @tran_flag int
        /* End of Variable Declaration */

        select @tran_flag = @@trancount

        if @tran_flag = 0       /*Transaction has not begun */
                begin transaction BusSel
        else        /* Transaction has already initiated */
                save transaction BusSel

        if @Identifier = 0
        /* Select all the columns from Bus Table for specific Business based on primary key*/
        begin
            select BusId,
                   BusNm,
                   BusTrdNm,
                   BusVcePhnNmb,
                   BusFormedDt
            from client.dbo.BusTbl
            where BusId = @BusId

             select @RetVal = @@rowcount,@error = @@error

           if @error != 0
                   begin
                      if @@trancount > 0 rollback tran BusSel
                      return @error
                   end
        end
```

```
        else
        if  @Identifier = 1
         /*Count Total Number Of Businesses*/
            begin
                select @RetVal = count(*)
                from BusTbl

                select @RetVal = @@rowcount,@error = @@error

               if @error != 0
                  begin
                     if @@trancount > 0 rollback tran BusSel
                     return @error
                  end
            end
    if @tran_flag = 0
                commit tran BusSel
    return @error
end
```

1. @Identifier and @RetVal are two system parameters which will be incorporated in all the Application Stored Procedures (ASP).
2. Print statements will not be used in stored procedures.
*3. For Select stored procedure, use the following @Identifiers :

@Identifier	Select Statement
0	Select all the columns on the basis of primary key.
1	Select Total No. Of Rows in the table.
2	Check Existance of Row On the basis of Primary Key.
3	Select Description of the ID from Independent tables. eg. Select the Description on the basis of BusId.
4	Select All ID's and Description from Independent tables (For Drop down datawindows.) eg. Select all the Sic Codes and Description.
11 Onwards	For Other Select Statements.

*5 For Insert, update and Delete (if approved by Data Administrator) stored procedures, follow the same template as above. Use 0 as Identifier, where operation is on the basis of primary key.

6 The @RetVal parameter will return the no of rows selected or affected.

* To add any other SQL statement Contact Database Administrator.