DB2 SQL PL Guidelines and Best Practice

Document History

|  |  |  |  |
| --- | --- | --- | --- |
| Ver. No | Author | Date | Remarks |
| 1.0 | Raymund Mallonga | 7/21/2019 | Initial Document |

# 1. Overview

This document defines the standards and guidelines that can be used as reference by developers when creating database objects/programs. Hopefully, with these general standards, every developer would be able to write very readable, self-documenting and easily maintained code.

# 2. General Standards

**Naming Conventions**

* Always name function or procedure that would suggest its intended purpose.
* Always specify specific names within the CREATE procedure or function header.
* Procedures and Functions will be usually be implemented as part of a module. Module Names would be specified in the Technical Specs to be provided by the project team.
* Make program names as meaningful as possible.
* Although SQL PL is not case sensitive, kindly use uppercase characters only for modules, procedures, udfs and udt names .
* Parameter inputs should be prefixed with “P\_” + <Mode> +<AAAAAA>

Example: P\_IN\_CIF\_NO

* Variable Names should be prefixed with “V\_”.
* Variable names should be meaningful, if based from a table, name should follow that of the table. Avoid vague names.
* If Variables will be used as the source value that would be saved to a table, it is good practice to anchor the variable name to the target column.

Example: DECLARE V\_NAME ANCHOR DATA TYPE TO TABLE SAMPLE.NAME

* Avoid re-using declared variables for a different purpose other than what was initially intended.
* User Defined Functions would always return a value, so names like GET\_DOCSTAMPS, GET\_SEQNO should be avoided, instead use DOCSTAMPS and SEQNO.
* Variable names identical to table names should be avoided to prevent confusion.
* Variable names should only be initialized with default values in the variable declaration section.

Example:

|  |
| --- |
| P1: BEGIN  –-  –- Declare variable in this section  –  DECLARE SQLCODE INTEGER DEFAULT 0; –- used for error handler  DECLARE SQLSTATE CHAR(5) DEFAULT ‘ ‘; –- used for error handler  DECLARE V\_UNITCODE CHAR(5) DEFAULT ‘10617‘; |

**Setting Variables**

* Variable names should only be initialized in the declaration section.

**Use of Temporary Tables**

Temporary tables can be declared in stored procedures to process large amounts of data that is not intended to be stored in the database. The data contained in the temporary table is only available for the current application session.

* Does not require locking.
* Can have minimal or no logging.
* Only visible to the current application.
* Can have index created against it.

**Code Readability**

* Prologue/Program Headers should appear at the start of the script. You may configure your IBM Data Studio templates to include this portion

|  |
| --- |
| -- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  -- Description: Describe the purpose of the object. If necessary,  -- describe the design of the object at a very high level.  --  -- Input Parameters:  --  -- Output Parameters:  --  -- Error Conditions Raised:  --  -- Author: <your name>  -- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  -- Revision History  -- Date Author Reason for Change  -- ----------------------------------------------------------------  -- 03 JAN 2015 J.Schmoe Created. |

* Use proper indention
* Always use comments
* Replace default labels generated by IBM Data Studio template with more meaningful labels.
* Avoid duplicate variable declaration within different compound statements of the same program.
* Avoid using GOTO statement since this would often lead to spaghetti code and would be difficult maintain and debug.

**Code Size**

* If a stored procedure body becomes too large, usually the stored procedure is doing too much. Try breaking down the functionality of a stored procedure into smaller and more manageable components.

**Exception Handling**

* Always declare exception handlers (EXIT or CONTINUE).
* Log SQL Exceptions using an autonomous utility logger (logger name may vary per Project team).
* Declaring custom conditions with its accompanying handlers would lend to a more readable code.
* You may use nested exceptions to localize exception handling.

**Commit and Rollback**

* Transaction control normally is handled by the caller.

**Value Assignment**

* SET statement allows parallel assignment of variables. Instead of SELECT ... INTO or VALUES... INTO, use SET when applicable.

|  |
| --- |
| SET V\_FIRST\_NAME, V\_LAST\_NAME = (SELECT FIRST\_NAME, LAST\_NAME  FROM EMPLOYEES WHERE EMP = 1);  \* this assumes that only a single record is returned. |

**Adhoc/Bootstrap Scripts**

* A single insert statement may be used to insert multiple rows at the same time.

|  |
| --- |
| INSERT INTO EMPLOYEE (FIRST\_NAME, LAST\_NAME)  VALUES (‘JUAN’,’DELACRUZ’),  (‘JOHN’,’SMITH’),  (‘JOSE’,’RIZAL’); |

**Deterministic vs Not Deterministic, External Action vs No External Action**

* Include “DETERMINISTIC” or “NOT DETERMINISTIC” Clause in the function/stored procedure header. A stored procedure/function is considered as Deterministic if the same input parameter values and same database state causes the program to return the same result all the time. Providing this clause would help the database manager by caching the result from the first execution.

Example: A procedure computing Docstamps would always return the same value for an input base amount of 1000. The only time the output would be different is if the parameters containing the docstamp rates would be updated to reflect a new value

* Include NO EXTERNAL ACTION clause to enable DB2 to use certain optimizations that assumes that the procedure has no external program impact. By default, clause is set to EXTERNAL ACTION.

**Result Sets**

* Avoid ambiguous cursor declarations. Specify the type of cursor by providing the “**FOR READ ONLY”** or “**FOR UPDATE”** clause in the select statement. Appropriate optimization will be done depending on the type declared. If not declared, DB2 will perform minimal optimizations.
* Minimize result set columns. Retrieve only the data that is needed by the calling application.
* Do not use “Select \* FROM”.

**Other useful tools**

* Use available unit testing tools. IBM Data Studio has unit testing capablities.
* For JUNIT style unit testing, you may use **db2unit** to create your test fixtures and test cases. Note, db2unit is an open source unit testing tool that needs to be installed in the database.

# 3. References

1. **Developing Error-Free Native Stored Procedures** by Tony Andrews <https://www.ibmbigdatahub.com/blog/developing-error-free-native-stored-procedures>
2. **DB2 SQL PL: Essential Guide for DB2 UDB, Linux, Unix Windows, I5/OS and z/OS, 2nd Edition** *by Fraser Mc Arthur, Paul Yip, Michael Gao, Drew Bradstock, Clara Liu, Raul Chong, Zamil Janmohamed.* Published by IBM Press 2004.