# ScriptsDB - Proceso: Creación de Scripts

Esta sección define el proceso de creación de scripts para soportar los desarrollos, ajustes, mejoras y solicitudes de cambios que impacten el modelo entidad relación de **SICOF ERP**.

## Notas de Versión

| Versión | Elabora | Revisa | Aprueba | Fecha Publicación | Observaciones |
|---|---|---|---|---|---|
| 1.0 | carlos.torres, abdimar.estrada | daberson.henao | daberson.henao | 05/10/2020 | Versión Inicial |

Antes de describir el proceso es importante la compresión de los siguientes conceptos.

## Conceptos Previos

- **Script de Actualización**: Son los scripts que contienen instrucciones DDL[1)]) o instrucciones DML[2)]) que serán aplicadas como solución en el modelo entidad relación de **SICOF ERP**, para soportar nuevos desarrollos, ajustes, mejoras o solicitudes.
- **Script de Reversión**: Son los scripts que contienen instrucciones DDL[3)]) o instrucciones DML[4)]) que serán aplicadas devolver el estado anterior del modelo entidad relación de **SICOF ERP**.
- **Funcionalidad Propietaria**: Se identifican como aquellos procesos un opciones que son utilizadas directamente por un usuario de **SICOF ERP**.
- **Archivo Leeme.txt**: Es un archivo interno que debe estar en el directorio raíz de cada Funcionalidad Propietaria. Este archivo es de obligatorio registro y gestión por cada cambio realizado y debe contener por cada actualización la lista ordena y secuencial de los scripts que integran la actualización además las reglas u observaciones a tener en cuenta.

## Proceso de Nombramiento de Scripts

A continuación se describe el proceso de nombramiento de scripts

### Síntesis

Asumiendo la siguiente Url genérica http://adacsc.co:1443/svn/repository/ADA/SICOF/ScriptsDB/SICOFERP/ se presentan los siguientes ejemplos:

- branches/Versión 1.0/00000000 FUNCIONALIDAD PROPIETARIA/1001 CREATE TABLE ESQUEMA.MAE_PRUEBAS.sql
- branches/Versión 1.0/00000000 FUNCIONALIDAD PROPIETARIA/Revert/1001 CREATE TABLE ESQUEMA.MAE_PRUEBAS REVERT.sql
- branches/Versión 1.0/00000000 FUNCIONALIDAD PROPIETARIA/1001a ALTER TABLE ESQUEMAMAE_PRUEBAS.sql

- branches/Versión 1.0/00000000 FUNCIONALIDAD PROPIETARIA/1002 CREATE OR REPLACE FORCE VIEW ESQUEMA.V_PRUEBAS.sql
- branches/Versión 1.0/00000000 FUNCIONALIDAD PROPIETARIA/1003a CREATE OR REPLACE PACKAGE ESQUEMA.PCK_PRUEBAS SPEC.sql
- branches/Versión 1.0/00000000 FUNCIONALIDAD PROPIETARIA/1003b CREATE OR REPLACE PACKAGE ESQUEMA.PCK_PRUEBAS BODY.sql

| Rama - Almacenamiento | Versión | Numeración Externa | Funcionalidad Propietaria. | Numeración Interna. | Evolución | Sentencia | Nombre | Reversión | Tipo |
|---|---|---|---|---|---|---|---|---|---|
| branches | Versión 1.0 | 00000000 | FUNCIONALIDAD PROPIETARIA | 1001 | | CREATE TABLE | ESQUEMA.MAE_PRUEBAS | | |
| branches | Versión 1.0 | 00000000 | FUNCIONALIDAD PROPIETARIA | 1001 | | CREATE TABLE | ESQUEMA.MAE_PRUEBAS | REVERT | |
| branches | Versión 1.0 | 00000000 | FUNCIONALIDAD PROPIETARIA | 1001 | a | ALTER TABLE | ESQUEMA.MAE_PRUEBAS | | |
| branches | Versión 1.0 | 00000000 | FUNCIONALIDAD PROPIETARIA | 1002 | | CREATE OR REPLACE FORCE VIEW | ESQUEMA.V_PRUEBAS | | |
| branches | Versión 1.0 | 00000000 | FUNCIONALIDAD PROPIETARIA | 1003 | a | CREATE OR REPLACE PACKAGE | ESQUEMA.PCK_PRUEBAS | | SPEC |
| branches | Versión 1.0 | 00000000 | FUNCIONALIDAD PROPIETARIA | 1003 | b | CREATE OR REPLACE PACKAGE | ESQUEMA.PCK_PRUEBAS | | BODY |

A lo largo de la sección se definirá cada fragmento del nombramiento de scripts

## Almacenamiento

Según el Flujo de Registro los scripts deben almacenarse según la siguiente sección Repositorio SVN

## Versionamiento

Toda Funcionalidad Propietaria debe estar versiona empezando con **Versión 1.0**. a Continuación se definen los siguientes lineamientos:

- La evolución de las versiones serán responsabilidad del líder técnico que soporte la funcionalidad.
- Cada versión debe tener su respectivo archivo Leeme.txt y su directorio Revert

## Extensión

Se deben utilizar las extensiones registradas en la sección Estandares y Prefijos PL/SQL Extensiones de archivos

## Agrupación

Cada script debe agruparse de acuerdo a la Funcionalidad Propietaria y estar contenido en el

directorio raíz de la funcionalidad.

## Numeración Externa

Cada directorio que represente una Funcionalidad Propietaria debe iniciar con una numeración de 8 dígitos, ser secuencial e iniciar a partir de 10000000.

Ejemplo:

```
00000000 FUNCIONALIDAD PROPIETARIA
```

### Nota

La numeración [00000000 - 09999999] está reservada para scripts de procesos internos y no debe ser utilizada para registrar scripts de lógica del negocio.

## Numeración Interna

Cada directorio que represente una Funcionalidad Propietaria deberá contener scripts de evolución de la funcionalidad, estos también deben estar numerados con 4 dígitos de forma secuencial y deben iniciar a partir de 1000.

Ejemplo:

```
1001 CREATE TABLE ESQUEMA.MAE_PRUEBAS.sql
```

### Nota

La numeración [0000 - 0999] está reservada para scripts de procesos internos y no debe ser utilizada para registrar scripts de lógica del negocio.

## Independencia / Segregación por Objeto

Los scripts deben ser generados por objetos, no se deben agrupar.

## Nombramiento por Sentencia

Cada script debe registrar en su nombre la sentencia general que describirá y debe estar en mayúsculas.

Ejemplo:

```
1001 CREATE TABLE ESQUEMA.MAE_PRUEBAS.sql
```

```
1002 CREATE OR REPLACE FORCE VIEW ESQUEMA.V_PRUEBAS.sql
```

## Evolución de Scripts

Según el tipo de objeto se pueden presentar los siguientes tipos de evoluciones de scripts:

### Evolución por sustitución

Este tipo de cambios modifican el script inicial. No es necesario crear un nuevo script ya que el control de versiones permitirá la gestión de cambios. Aplica para los siguientes tipos de objetos:

- FUNCTION
- PROCEDURE
- PACKAGE
- TRIGGERS
- VIEW
- MATERIALIZED VIEW
- TYPES
- Cualquier instrucción que permite el comando CREATE OR REPLACE

Ejemplo:

```
1002 CREATE OR REPLACE FORCE VIEW ESQUEMA.V_PRUEBAS.sql
```

**Nota** Todos los objetos que en su definición tengan especificación y cuerpo deben registrarse en archivos separados de la siguiente manera:

- Crear un script para la especificación y otro para el cuerpo o definición
- Deben tener la misma numeración
- La especificación debe tener la letra (a) después de la numeración y deben terminar con la palabra SPEC
- El cuerpo o definición debe tener la letra (b) después de la numeración y deben terminar con la palabra BODY
- Aplica para PACKAGE, TYPES

Ejemplo:

```
1003a CREATE OR REPLACE PACKAGE ESQUEMA.PCK_PRUEBAS SPEC.sql
1003b CREATE OR REPLACE PACKAGE ESQUEMA.PCK_PRUEBAS BODY.sql
```

### Evolución por estado

Este tipo de cambios se anexan al script inicial por medio de otro script. No es posible modificar el archivo inicial porque los objetos no permiten la instrucción CREATE OR REPLACE. Aplica para los objetos:

- TABLE
- ALTER TABLE
- Scripts DML (Insert, Delete, Update)

Los scripts de evolución por estado deben ser nombrados por el mismo nombre del script original pero la numeración debe estar acompañada de letras en minúsculas (al final de la numeración) de la [a-z] en caso de alcanzar el limite se procederá con las combinaciones [aa-az] y así sucesivamente.

Ejemplos:

```
1002 CREATE TABLE PRESUP01.MAE_PRUEBAS.sql
1002a ALTER TABLE PRESUP01.MAE_PRUEBAS.sql
1002b ALTER TABLE PRESUP01.MAE_PRUEBAS.sql
1002c ALTER TABLE PRESUP01.MAE_PRUEBAS.sql
```

## Reversión de Scripts

Cada Funcionalidad Propietaria contiene un directorio llamado **Revert** el cual almacenará los scripts de reversión. A continuación se dictan los lineamientos para crearlos:

- El nombre del Script de Reversión debe ser el mismo del script de actualización y debe finalizar con la palabra REVERT
- El Script de Reversión deberá devolver el estado anterior del objeto modificado por el Script de Actualización, esta situación puede tener algunas excepciones con sentencias DML en los casos donde se afecten secuencias o disparadores concatenados
- En las sentencias DML los Script de Reversión deben considerar cargas previas a una modificación.

Ejemplos:

```
Script de Actualización: 1001 CREATE TABLE ESQUEMA.MAE_PRUEBAS.sql
Script de Reversión: 1001 CREATE TABLE ESQUEMA.MAE_PRUEBAS REVERT.sql
```

## Personalizaciones / Clientes

En algunas situaciónes algunos procesos tienen particularidades entre clientes. En estas situaciones la Funcionalidad Propietaria tendra en sus versiones el código estandar que aplica a todos los clientes y la particularidades deben registrarse en un directorio interno con las siguientes condiciones:

- Debe empezar con la descripción *ConfigCliente* seguido del nombre del cliente Ej: **ConfigClienteADA**
- Debe contener internamente su respectivo archivo **Leeme.txt**
- Debe contener internamente su directorio **Revert**
- Desde el archivo Leeme.txt de la Funcionalidad Propietaria se debe orquestar la ejecución del directorio de personalización según la actualización que aplique.

## Archivo Leeme.txt

Este archivo es muy importante para la gestión de script ya que en el se deben registrar las actualizaciones que se realizan en las Funcionalidades Propietarias para comprender el archivo se explicará por medio de una plantilla y finalmente se mostrará un archivo diligenciado según los ejemplos utilizados a lo largo de la sección:

## Plantilla Leeme.txt

```
################################################################################
###############
# ARCHIVO DE IMPLEMENTACION - FUNCIONALIDAD PROPIETARIA - EJ: 00000000
FUNCIONALIDAD PROPIETARIA
################################################################################
###############
# Versión: Número de la Versión liberada. EJ: 1.0
# Fecha: Fecha de liberación de la Funcionalidad Propietaria, por lo generar
tiene la fecha del primer scritp registrado. EJ:06/10/2020
# Desarrollador: Desarrollor que publica la funcionalidad propietaria.
EJ:carlos.torres@ada.co
# Módulos: Módulos de la funcionalidad propietaria. EJ:Presupuesto
12.5.2.5.0
################################################################################
###############
# A continuación se describe el proceso de implementación de los scripts
requeridos para
# la actualización del proceso. Favor tener presente el orden descrito en
este
# archivo ya que de no seguir el orden pueden generarse errores de
compilación por
# referencias invalidas.
################################################################################
###############
# PASOS DE EJECUCION DE SCRIPTS DB
################################################################################
###############
# Alcance:
# Indica el alcancela de la funcionalidad propietaria. EJ: Aplica para todos
los clientes.
################################################################################
###############
# Ejecutar Script:  En esta sección se registran de forma secuencial a lista
de script
#                   que deben ser ejecutados para soportar la solución
implementada.
# Ejemplo:
#           1.    1001 CREATE TABLE ESQUEMA.MAE_PRUEBAS.sql
#           2.    1002 CREATE OR REPLACE FORCE VIEW ESQUEMA.V_PRUEBAS.sql
#           3.    1003a CREATE OR REPLACE PACKAGE ESQUEMA.PCK_PRUEBAS
SPEC.sql
#           4.    1003b CREATE OR REPLACE PACKAGE ESQUEMA.PCK_PRUEBAS
```

```
BODY.sql
################################################################################
###############
# OBSERVACIONES
#-------------------------------------------------------------------------------
--------------
# Aqui se deben registrar observacione sobre la ejecución de los scripts.
# EJ: Alguna ocnsideración externa a la ejecución del script, roles,
permisos, etc
# Se debe registrar la revisión inicial
# EJ: Revision History: 1.0 - carlos.torres@ada.co - 06/10/2020 07:58:34 :
Liberación de versión
################################################################################
###############
#   * Validar objetos invalidos al terminar la actualización.
#   * Ejecutar actualización con un usuario que tenga permisos con el
usuario ESQUEMA
################################################################################
###############
# HISTORIAL DE ACTUALIZACIONES
# En esta sección se registran las evoluciones de script
################################################################################
###############
# Se debe registrar la revisión de la actulización
# EJ: Revision History: 1.1 - carlos.torres@ada.co - 07/10/2020 08:58:34 :
Adición de columnas
#-------------------------------------------------------------------------------
--------------
# Ejecutar Script:  En esta sección se registran de forma secuencial a lista
de script
#                   que deben ser ejecutados para soportar la solución
implementada.
# Ejemplo:
#           1.    1001a ALTER TABLE ESQUEMAMAE_PRUEBAS.sql
# OBSERVACIONES
#-------------------------------------------------------------------------------
--------------
# Aqui se deben registrar observacione sobre la ejecución de los scripts.
#   * Ejecutar los scripts en el orden propuesto
#   * Validar objetos invalidos al terminar la actualización.
#   * Ejecutar actualización con un usuario que tenga permisos con el
usuario ESQUEMA
```

## Ejemplo Leeme.txt

```
################################################################################
###############
# ARCHIVO DE IMPLEMENTACION - 00000000 FUNCIONALIDAD PROPIETARIA
################################################################################
###############
```

```
# Versión: 1.0
# Fecha: 06/10/2020
# Desarrollador: carlos.torres@ada.co
# Módulos: Presupuesto 12.5.2.5.0
################################################################################
###############
# A continuación se describe el proceso de implementación de los scripts
requeridos para
# la actualización del proceso. Favor tener presente el orden descrito en
este
# archivo ya que de no seguir el orden pueden generarse errores de
compilación por
# referencias invalidas.
################################################################################
###############
# PASOS DE EJECUCION DE SCRIPTS DB
################################################################################
###############
# Alcance:
# Aplica para todos los clientes.
################################################################################
###############
# Ejecutar Script:
#           1.    1001 CREATE TABLE ESQUEMA.MAE_PRUEBAS.sql
#           2.    1002 CREATE OR REPLACE FORCE VIEW ESQUEMA.V_PRUEBAS.sql
#           3.    1003a CREATE OR REPLACE PACKAGE ESQUEMA.PCK_PRUEBAS
SPEC.sql
#           4.    1003b CREATE OR REPLACE PACKAGE ESQUEMA.PCK_PRUEBAS
BODY.sql
################################################################################
###############
# OBSERVACIONES
#-------------------------------------------------------------------------------
--------------
# Revision History: 1.0 - carlos.torres@ada.co - 06/10/2020 07:58:34 :
Liberación de versión
################################################################################
###############
#   * Validar objetos invalidos al terminar la actualización.
#   * Ejecutar actualización con un usuario que tenga permisos con el
usuario ESQUEMA
################################################################################
###############
# HISTORIAL DE ACTUALIZACIONES
# En esta sección se registran las evoluciones de script
################################################################################
###############
# Revision History: 1.1 - carlos.torres@ada.co - 07/10/2020 08:58:34 :
Adición de columnas
#-------------------------------------------------------------------------------
```

```
---------------
# Ejecutar Script:
#           1.    1001a ALTER TABLE ESQUEMAMAE_PRUEBAS.sql
# OBSERVACIONES
#-------------------------------------------------------------------------------
---------------
# Aqui se deben registrar observacione sobre la ejecución de los scripts.
#   * Ejecutar los scripts en el orden propuesto
#   * Validar objetos invalidos al terminar la actualización.
#   * Ejecutar actualización con un usuario que tenga permisos con el
usuario ESQUEMA
```

←Volver atrás

[1] [3]
,
lenguaje de base de datos o lenguaje de definición de datos (Data Definition Language, DDL por sus siglas en inglés
[2] [4]
,
Lenguaje de Manipulación de Datos (Data Manipulation Language, DML