Universidad Carlos III de Madrid

PROYECTO FIN DE CARRERA

Ingeniería Técnica de Informática de Gestión

Validación de Informes Económicos/Contables/Financieros Semánticos y su Implementación en Base de Datos, de una Forma Automática.

Autor: Abel Nieto Cano

Tutores: Ignacio J. Santos Forner y Elena Castro Galán

Leganés, 13 de Octubre de 2015

Título: Validación de Informes Económicos/Contables/Financieros Semánticos y su Implementación en Base de Datos, de una Forma Automática.

Autor: Abel Nieto Cano

Director: Elena Castro Galán

EL TRIBUNAL

Presidente: María Dolores Cuadra Fernández

Vocal: Ana María Iglesias Maqueda

Secretario: Harith Al-Jumaily Taha Abdulla

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 13 de Octubre de 2015 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

Agradezco a Ignacio y a Elena, tutores de este proyecto, por todo el esfuerzo dedicado, por su compresión y apoyo en los peores momentos y por todo lo disfrutado. Igualmente, por la confianza depositada en mí, y por la paciencia que han tenido durante el desarrollo del trabajo.

Gracias al apoyo dado por mi familia, José Francisco y Rufina, que nunca desistieron en su insistencia para que realizara este proyecto, y a Naiara, que me aguantó y me dio ese empujón cuando más lo necesitaba para no desistir en el empeño de terminar el Proyecto final de Carrera.

A ellos va dedicado este proyecto que más allá de lo aprendido a nivel académico, Con este trabajo he conseguido eliminar barreras mentales y de motivación que me impedían mi propio reconocimiento.

Resumen

Este documento consta de 8 secciones, salvo la bibliografía y glosario. En La primera sección está Introducción y definimos conceptos básicos así con el alcance y objetivos del proyecto. La segunda sección hace referencia a los requerimientos Software y hardware empleados para desarrollar y desplegar el trabajo realizado. La tercera sección muestra el metamodelo definido por la Autoridad Bancaria Europea (EBA) a través de las taxonomías FINREP (Informe Financiero) y COREP (Informe de Solvencia Común) ambas son un Metamodelo de los informes requeridos por la Regulación Bancaria Europea y su mapeo en un modelo Relacional (implementado en MS SQL SERVER 2012). La cuarta Sección hace un estudio de la arquitectura de XBRL, la base de datos, su estructura de tablas como paso hacia un sistema gestor de Base de datos más apropiado (MS SQL Server 2012). La quinta sección se definen las condiciones y tecnología utilizada para la validación del *data Point Model (DPM) como paso de datos al modelo relacional, también se definen las pruebas de validación efectuadas. La sexta sección* se definen las condiciones utilizadas para el mapeo de *data Point Model (*DPM) hacia un Modelo de datos Multidimensional (MDM) y el MDM implementado en una base de datos relacional. La séptima sección atañe a los futuros *trabajos y mejoras del proyecto definido en este documento.* La Octava y última Sección engloban el Estudio de viabilidad, planificación del proyecto y Presupuesto.

Se espera que el lector tenga un conocimiento del DPM, así como conocimientos básicos de XBRL, aunque este trabajo, introducirá ambos modelos. También se espera que el lector pueda tener conocimiento en la creación de modelos conceptuales para bases de datos relacionales y multidimensionales.

Abstract

This document consists of 8 sections, except the bibliography and glossary. The first section is Introduction and basic concepts and define the scope and objectives of the project. The second section refers to the software requirements and hardware used to develop and deploy their work. The third section shows the metamodel defined by the European Banking Authority (EBA) through taxonomies FINREP (Financial Report) and COREP (Common Solvency Report) both are a metamodel of the reports required by the Banking Regulation and mapping a relational model (implemented in MS SQL Server 2012). The fourth section is a study of the architecture of XBRL, the database table structure as a step towards a management system more appropriate data base (MS SQL Server 2012). Section five conditions and technology used for validation of data Point Model (DPM) and pass data to the relational model are defined, the validation tests conducted are also defined. The sixth section the conditions used for mapping data Point Model (DPM) to a multidimensional data model (MDM) MDM and implemented in a relational database is defined. The seventh section concerns the future work and improvements to the project defined in this document. The eighth and final section covers the feasibility study, project planning and budget.

It is expected that the reader has a knowledge of DPM, as well as basic knowledge of XBRL, although this work will introduce two models. It is also expected that the reader may have knowledge in the creation of conceptual models for relational databases and multidimensional data.

Tabla de contenidos

1 INTRODUCCIÓN 18

1.1 XBRL 19

1.2 Introducción a los informes semánticos. Meta Modelo XBRL. 19

1.3 Objetivos. 21

1.4 Alcance del Desarrollo y medios empleados. 22

1.5 Relación con otros trabajos. 24

2 REQUERIMIENTOS SOFTWARE/HARDWARE 25

2.1 Requerimientos Hardware 25

2.2 Requerimientos Software 25

3 DISEÑO DE LOS METADATOS DE LOS INFORMES 27

3.1 Meta modelo definido por la EBA (FINREP y COREP). 27

3.2 Creación de la estructura y la carga de la DPM de la EBA en un modelo relacional PLSQL 28

4 SISTEMA GESTOR DE BASE DE DATOS. ESTRUCTURA DE LA BASE DE DATOS. 31

4.1 Elección del modelo de datos propio. 32

4.2 API Export de las Entidades, atributos y datos de la EBA a SQL Server 2012. 35

4.3 Creación de la Base de datos en SQL Server 2012 46

4.4 Análisis y definición del Modelo de datos. 51

4.4.1 Tablas y agrupaciones de Tabla 54

4.4.2 La presentación en lista 55

4.4.3 Dimensionalidad del modelo de datos 57

4.4.4 Análisis dimensional de las Plantillas de datos. 59

4.4.5 Estructuras de árbol en el Modelo 62

4.4.6 Tablas y descripción de campos. 65

4.4.7 Preocupaciones dentro del Modelo hacia XBRL 80

5 PROCESO DE VALIDACIÓN DE LOS DATOS DEL DPM 81

5.1 Representación del proceso 82

5.2 Reglas de validación aplicadas en el estudio. 83

5.3 Desarrollo de la solución. 85

5.3.1 Parámetros de la solución. 89

5.3.2 Paquete Master de la Solución. 94

5.3.3 Paquetes DTS de la solución. 97

5.3.4 Control de errores en los Paquetes DTS de la solución. 105
5.3.5 Control de errores en los Paquetes DTS con Validaciones de Estructuras en árbol. 114
5.4 PRUEBAS DE CONCEPTO EFECTUADAS 116
6 CREACIÓN DEL MODELO MULTIDIMENSIONAL 124
6.1 QUE ES UN MODELO MULTIDIMENSIONAL 124
6.2 ESPECIFICACIONES DEL MODELO EN ESTRELLA 125
6.3 CREACIÓN DEL MODELO EN ESTRELLA. 129
6.3.1 Creación de la Tabla de hechos. 131
6.3.2 Creación de las tablas Dimensiones. 134
6.3.3 Explotación de los Datos del MDM. 141
7 FUTURAS MEJORAS. CONCLUSIONES. 144
8 PLANIFICACIÓN Y SEGUIMIENTO. PRESUPUESTO 146
8.1 INTRODUCCIÓN 146
9 GLOSARIO. 147
10 REFERENCIAS. 148
11 ANEXOS 150

Índice de figuras

FIGURA 1. ESQUEMA GENERACIÓN DPM XBRL. 20

FIGURA 2. ESQUEMA PROYECTO. (EN AZUL) 22

FIGURA 3. DIFERENTES TIPOS DE DBMS'S. REFERENCIA [2] 23

FIGURA 4. REQUERIMIENTOS HARDWARE. 25

FIGURA 5. PÁGINA WEB EBA. REFERENCIA [1] 28

FIGURA 6. MODELO PROPUESTO POR LA EBA. REFERENCIA [2] 29

FIGURA 7. PARTE DEL EBA DPM 31

FIGURA 8. COMPARATIVA MS ACCESS VS MS SQL SERVER. REFERENCIA [14] 33

FIGURA 9. FLUJO DE LA BDD ACCESS A SQL SERVER 34

FIGURA 10. IMPORT AND EXPORT. EXPORTACIÓN DE LA BDD ACCESS A SQL SERVER 35

FIGURA 11. IMPORT AND EXPORT. DATA SOURCE DE LA BDD ACCESS A SQL SERVER 36

FIGURA 12. IMPORT AND EXPORT. LISTA DE DATA SOURCES DISPONIBLES PARA EXPORTAR. 36

FIGURA 13. IMPORT AND EXPORT. DESTINO BDD SQL SERVER. 37

FIGURA 14. IMPORT AND EXPORT. ESPECIFICACIÓN DE TABLA A BDD SQL SERVER. 38

FIGURA 15. IMPORT AND EXPORT. TABLAS Y VISTAS DESDE ACCESS HACIA SQL SERVER. 39

FIGURA 16. IMPORT AND EXPORT. MAPPING DE ATRIBUTOS Y TIPOS DE DATOS DESDE ACCESS A SQL SERVER 40

FIGURA 17. IMPORT AND EXPORT. GUARDAR PROYECTO EN PAQUETES ETL-SSIS 43

FIGURA 18. IMPORT AND EXPORT. VERIFICACIÓN DE LAS ESTRUCTURAS DE DATOS. 44

FIGURA 19. IMPORT AND EXPORT. EJECUCIÓN PROCESO DE EXPORTACIÓN DE LAS ESTRUCTURAS DE DATOS. 45

FIGURA 20. IMPORT AND EXPORT. ESTRUCTURA DE DATOS EBA EN SQL SERVER 2012. 46

FIGURA 21.1. COMPONENTE PARA LA GENERACIÓN DE SCRIPTS DE BDD EN SQL SERVER 2012. 47

FIGURA 21.2. COMPONENTE PARA LA GENERACIÓN DE SCRIPTS DE BDD EN SQL SERVER 2012. 48

FIGURA 21.3. COMPONENTE PARA LA GENERACIÓN DE SCRIPTS DE BDD EN SQL SERVER 2012. 49

FIGURA 21.4. COMPONENTE PARA LA GENERACIÓN DE SCRIPTS DE BDD EN SQL SERVER 2012. 50

FIGURA 22. ESQUEMA DE CREACIÓN DEL MODELO DE BDD EN SQL SERVER 2012. 52

FIGURA 23. DIAGRAMA EN SQL SERVER 2012. DBO.DIAGRAM_DPM2.2_TAXONOMY. REPRESENTA LA ASOCIACIÓN DE TABLAS DE UNA TAXONOMÍA. 54

FIGURA 24. DIAGRAMA EN SQL SERVER 2012. DBO.DIAGRAM_DPM2.2_AXISORDINATE. REPRESENTA LA ASOCIACIÓN DE TABLAS DE TIPO LISTA DEL DPM. 56

FIGURA 25. RELACIÓN 1..N TABLECELL → DATAPOINT 57

FIGURA 26. DIAGRAMA EN SQL SERVER 2012. DBO.DIAGRAM_DPM2.2_DIMENSIONAL. REPRESENTA LA ASOCIACIÓN DE TABLAS DEL MODELO DIMENSIONAL DEL DPM. 58

FIGURA 27. EJEMPLO DE REPRESENTACIÓN DE CONCEPTOS CONTABLES EN EL EJE X EN EL TIEMPO EJE Y Y SU MAGNITUD EJE Z. REFERENCIA [9] 60

FIGURA 28. EJEMPLO DE REPRESENTACIÓN DE DIMENSIONES PADRE EN EL EJE X CON LAS DIMENSIONES HIJO EN EL EJE Y Y SU MAGNITUD EJE Z DE TIPO MONETARIA. REFERENCIA [9] 60

FIGURA 29. DIAGRAM_DPM2.2_TEMPLATES. REPRESENTACIÓN DE DIMENSIONES, EJES Y PUNTO DE DATO 62

FIGURA 30. EJEMPLO DE REPRESENTACIÓN DE LAS TABLAS AXISORDINATE Y HIERARCHYNODE 63

FIGURA 31. REPRESENTACIÓN DE ENUMERACIONES DE CAMINOS VÁLIDOS Y NO VALIDOS EN UNA ESTRUCTURA PADRE-HIJO. 64

TABLA 1. EJEMPLO ESTRUCTURA EN ÁRBOL VALIDA. 64

TABLA 2. ESTRUCTURA TABLA AXIS. REFERENCIA [1] 65

TABLA 3. ESTRUCTURA TABLA AXISORDINATE. REFERENCIA [1] 66

TABLA 4. ESTRUCTURA TABLA CONTEXTDEFINITION. REFERENCIA [1] 66

TABLA 5. ESTRUCTURA TABLA CONTEXTOFDATAPOINTS. REFERENCIA [1] 67

TABLA 6. ESTRUCTURA TABLA DATAPOINT. 67

TABLA 7. ESTRUCTURA TABLA DATAPOINTVERSION. REFERENCIA [1] 68

TABLA 8. ESTRUCTURA TABLA DATATYPE. REFERENCIA [1] 68

TABLA 9. ESTRUCTURA TABLA DIMENSION. REFERENCIA [1] 69

TABLA 10. ESTRUCTURA TABLA DIMENSIONCOORDINATE. 69

TABLA 11. ESTRUCTURA TABLA DOMAIN. REFERENCIA [1] 70

TABLA 12. ESTRUCTURA TABLA FLOWTYPE. 70

TABLA 13. ESTRUCTURA TABLA HIERARCHY. REFERENCIA [1] 71

TABLA 14. ESTRUCTURA TABLA HIERARCHYNODE. REFERENCIA [1] 72

TABLA 15. ESTRUCTURA TABLA MEMBER. REFERENCIA [1] 72

TABLA 16. ESTRUCTURA TABLA METRIC. REFERENCIA [1] 73

TABLA 17. ESTRUCTURA TABLA MODULE. 73

TABLA 18. ESTRUCTURA TABLA MODULETABLEORGROUP. 73

TABLA 19. ESTRUCTURA TABLA MODULETABLEVERSION. 74

TABLA 20. ESTRUCTURA TABLA OPENAXISRESTRICTION. REFERENCIA [1] 74

TABLA 21. ESTRUCTURA TABLA OPENMEMBERRESTRICTION. REFERENCIA [1] 75

TABLA 22. ESTRUCTURA TABLA REPORTINGFRAMEWORK. REFERENCIA [1] 75

TABLA 23. ESTRUCTURA TABLA TABLE. REFERENCIA [1] 75

TABLA 24. ESTRUCTURA TABLA TABLECELL. REFERENCIA [1] 76

TABLA 25. ESTRUCTURA TABLA TABLEGROUP. REFERENCIA [1] 76

TABLA 26. ESTRUCTURA TABLA TABLEGROUPTEMPLATES. REFERENCIA [1] 77

TABLA 27. ESTRUCTURA TABLA TABLEVERSION. REFERENCIA [1] 77

TABLA 28. ESTRUCTURA TABLA TAXONOMY. REFERENCIA [1] 78

TABLA 29. ESTRUCTURA TABLA TEMPLATE. REFERENCIA [1] 78

FIGURA 32. MODELO DE DATOS COMPLETO. REFERENCIA [1] 79

FIGURA 33. CICLO DE VIDA. PROYECTO SW 81

FIGURA 34. FLUJO DE INSERCIÓN DE DATOS VALIDADOS DEL DPM 82

FIGURA 35. MODELO DE DATOS VALIDADO MEDIANTE LA HERRAMIENTA DE INTEGRACIÓN DE DATOS SQL SERVER INTEGRATION SERVICE. 83

TABLA 30. REGLAS DE VALIDACIÓN APLICADAS 84

FIGURA 36. FLUJO DE INSERCIÓN DE DATOS MEDIANTE SSIS. 85

FIGURA 37. ESTRUCTURA PAQUETE DTS. REFERENCIA 15 86

FIGURA 38. FLUJO DE EJECUCIÓN SOLUCIÓN SSIS EBA_XBRL_SSIS. 88

FIGURA 39. PARÁMETROS DE LA SOLUCIÓN. ARCHIVO PROJECT PARAMS. 89

FIGURA.40 CONFIGURACIÓN DE CONEXIONES EN LA SOLUCIÓN DE SSIS. 90

FIGURA.41 OPCIONES AVANZADAS. CONFIGURACIÓN DE CONEXIONES EN LA SOLUCIÓN DE SSIS. 91

FIGURA 42. MENSAJE TEST DE CONECTIVIDAD A LA BDD. 92

FIGURA 43. PANTALLA DE CONFIGURACIÓN DE CONEXIONES A FICHEROS EXCEL 93

FIGURA 44. PANTALLA DE CONFIGURACIÓN DE CONEXIONES A FICHEROS EXCEL 94

FIGURA 45. IS_MASTER_EBA.DTSX. PAQUETE DE INICIO. 95

FIGURA 46. FLUJO DE DATOS. CONEXIÓN AL FICHERO DE ENTRADA EXCEL. 98

FIGURA 47. FLUJO DE DATOS. CONEXIÓN AL FICHERO DE ENTRADA EXCEL. PREVIEW. 98

FIGURA 48. FLUJO DE DATOS. CONEXIÓN AL FICHERO DE ENTRADA EXCEL. COLUMNS 99

FIGURA 49. FLUJO DE DATOS. CONEXIÓN AL FICHERO DE ENTRADA EXCEL. ERROR OUTPUT 100

FIGURA 50. FLUJO DE DATOS. CONVERSIÓN DE DATOS DEL FICHERO DE ENTRADA. 101

FIGURA 51. FLUJO DE DATOS. TRANSFORMACIÓN DE DATOS DE COLUMNA DERIVADA. 102

FIGURA 52. FLUJO DE DATOS. DESTINO OLE DB. 104

FIGURA 53. FLUJO DE DATOS. DESTINO OLE DB. MAPPINGS. 105

FIGURA 54. FLUJO DE EJECUCIÓN. CONTROL DE ERRORES Y EJECUCIÓN DE UN PAQUETE DTS. 106

TABLA 31. ESTRUCTURA TABLA LOG_ERRORS. 107

FIGURA 55. FLUJO DE EJECUCIÓN. CONTROL DE ERRORES DE UN PAQUETE DTS. 109

FIGURA 56. FLUJO DE EJECUCIÓN. PARÁMETROS DEFINIDOS EN EL CONTROL DE ERRORES DE UN PAQUETE DTS. 110

FIGURA 57. FLUJO DE EJECUCIÓN. CONTROL DE FIN DE EJECUCIÓN DE UN PAQUETE DTS. 112

FIGURA 58. FLUJO DE EJECUCIÓN. CONTROL DE FIN DE EJECUCIÓN DE UN PAQUETE DTS. MAPPING 113

FIGURA 59. EJEMPLO LOG DE INSERCIÓN SATISFACTORIA EN LA TABLA LOG_ERRORS. 114

FIGURA 60. PROCESO DE VALIDACIÓN DE ERRORES EN EL CASO ESPECIAL DE TABLAS CON ESTRUCTURAS EN ÁRBOL. IS_HIERARCHYNODE.DTSX. 114

FIGURA 61. TEST1. PROCESO DE VALIDACIÓN DE LA TABLA CONCEPTOS. 117

FIGURA 62. TEST1. FICHERO DE ENTRADA MODIFICADO CON UN ERROR DE PK DUPLICADA. 117

FIGURA 63. TEST1. MENSAJE DE ERROR. 118

FIGURA 64. TEST2. PROCESO DE VALIDACIÓN DE LA TABLA MIEMBROS. 118

FIGURA 65. TEST2. PROCESO DE VALIDACIÓN DE LA TABLA MIEMBROS. ERROR INTEGRIDAD REFERENCIAL VIOLADA 119

FIGURA 66. TEST2. MENSAJE DE ERROR. 119

FIGURA 67. TEST4. PROCESO DE VALIDACIÓN DE LA TABLA DIMENSIONES. 120

FIGURA 68. TEST4. PROCESO DE VALIDACIÓN DE LA TABLA DIMENSIONES. ERROR PK CON VALOR NULL. 121

FIGURA 69. TEST4. FICHERO DE ENTRADA MODIFICADO CON UN ERROR DE PK NULL. **121**

FIGURA 70. TEST4. MENSAJE DE ERROR. **121**

FIGURA 71. TEST5. PROCESO DE VALIDACIÓN DE LA TABLA HIERARCHYNODE. ESTRUCTURA ÁRBOL VIOLADA **122**

FIGURA 72. TEST5. PROCESO DE VALIDACIÓN DE LA TABLA HIERARCHYNODE. ESTRUCTURA ÁRBOL VIOLADA **123**

FIGURA 73. TEST5. FICHERO DE ENTRADA MODIFICADO CON UNA VIOLACIÓN DE LA ESTRUCTURA EN ÁRBOL. **123**

FIGURA 74. TEST5. MENSAJE DE ERROR. **123**

FIGURA 75. MODELO EN ESTRELLA. REFERENCIA [9] **124**

FIGURA 76. TABLAS MAESTRAS. TIPO DATOS, UNIDAD Y BALANCE. **126**

TABLA 32. DEFINICIÓN DATOS MAESTROS DE LA TABLA DE HECHOS. **127**

FIGURA 77. EJEMPLO DE TABLA DIMENSION GENERADA DE FORMA DINÁMICA **127**

FIGURA 78. EJEMPLO DE TABLA DE HECHOS GENERADA DE FORMA DINÁMICA, CON TODAS SUS REFERENCIAS. **129**

FIGURA 79.PASOS PARA LA GENERACIÓN DEL MODELO EN ESTRELLA PROPUESTO EN EL ARTÍCULO ... **130**

FIGURA 80. FLUJO DE DATOS IS_DIMENSION. EJECUCIÓN PROCEDIMIENTO SQL DE CREACIÓN DE LA TABLA DE HECHOS. **132**

FIGURA 81. DIAGRAMA DE BDD TABLA DE HECHOS Y REFERENCIA A LAS TABLA MAESTRAS. **133**

FIGURA 82. DIAGRAMA DE BDD TABLA DE HECHOS Y REFERENCIA A LAS TABLA MAESTRAS DE DIMENSIÓN. **137**

FIGURA 83. DIAGRAMA DE BDD DEL MDM. **140**

FIGURA 84. DIAGRAMA DE BDD DEL MDM EN LA SOLUCIÓN DE ANALYSIS SERVICE. **143**

FIGURA 85. DIAGRAMA DE BDD TABLA DE HECHOS Y REFERENCIA A LAS TABLA MAESTRAS DE DIMENSIÓN. **145**

Índice de tablas

FIGURA 1. ESQUEMA GENERACIÓN DPM XBRL. 20

FIGURA 2. ESQUEMA PROYECTO. (EN AZUL) 22

FIGURA 3. DIFERENTES TIPOS DE DBMS'S. REFERENCIA [2] 23

FIGURA 4. REQUERIMIENTOS HARDWARE. 25

FIGURA 5. PÁGINA WEB EBA. REFERENCIA [1] 28

FIGURA 6. MODELO PROPUESTO POR LA EBA. REFERENCIA [2] 29

FIGURA 7. PARTE DEL EBA DPM 31

FIGURA 8. COMPARATIVA MS ACCESS VS MS SQL SERVER. REFERENCIA [14] 33

FIGURA 9. FLUJO DE LA BDD ACCESS A SQL SERVER 34

FIGURA 10. IMPORT AND EXPORT. EXPORTACIÓN DE LA BDD ACCESS A SQL SERVER 35

FIGURA 11. IMPORT AND EXPORT. DATA SOURCE DE LA BDD ACCESS A SQL SERVER 36

FIGURA 12. IMPORT AND EXPORT. LISTA DE DATA SOURCES DISPONIBLES PARA EXPORTAR. 36

FIGURA 13. IMPORT AND EXPORT. DESTINO BDD SQL SERVER. 37

FIGURA 14. IMPORT AND EXPORT. ESPECIFICACIÓN DE TABLA A BDD SQL SERVER. 38

FIGURA 15. IMPORT AND EXPORT. TABLAS Y VISTAS DESDE ACCESS HACIA SQL SERVER. 39

FIGURA 16. IMPORT AND EXPORT. MAPPING DE ATRIBUTOS Y TIPOS DE DATOS DESDE ACCESS A SQL SERVER 40

FIGURA 17. IMPORT AND EXPORT. GUARDAR PROYECTO EN PAQUETES ETL-SSIS 43

FIGURA 18. IMPORT AND EXPORT. VERIFICACIÓN DE LAS ESTRUCTURAS DE DATOS. 44

FIGURA 19. IMPORT AND EXPORT. EJECUCIÓN PROCESO DE EXPORTACIÓN DE LAS ESTRUCTURAS DE DATOS. 45

FIGURA 20. IMPORT AND EXPORT. ESTRUCTURA DE DATOS EBA EN SQL SERVER 2012. 46

FIGURA 21.1. COMPONENTE PARA LA GENERACIÓN DE SCRIPTS DE BDD EN SQL SERVER 2012. 47

FIGURA 21.2. COMPONENTE PARA LA GENERACIÓN DE SCRIPTS DE BDD EN SQL SERVER 2012. 48

FIGURA 21.3. COMPONENTE PARA LA GENERACIÓN DE SCRIPTS DE BDD EN SQL SERVER 2012. 49

FIGURA 21.4. COMPONENTE PARA LA GENERACIÓN DE SCRIPTS DE BDD EN SQL SERVER 2012. 50

FIGURA 22. ESQUEMA DE CREACIÓN DEL MODELO DE BDD EN SQL SERVER 2012. 52

FIGURA 23. DIAGRAMA EN SQL SERVER 2012. DBO.DIAGRAM_DPM2.2_TAXONOMY. REPRESENTA LA ASOCIACIÓN DE TABLAS DE UNA TAXONOMÍA. **54**

FIGURA 24. DIAGRAMA EN SQL SERVER 2012. DBO.DIAGRAM_DPM2.2_AXISORDINATE. REPRESENTA LA ASOCIACIÓN DE TABLAS DE TIPO LISTA DEL DPM. **56**

FIGURA 25. RELACIÓN 1..N TABLECELL → DATAPOINT **57**

FIGURA 26. DIAGRAMA EN SQL SERVER 2012. DBO.DIAGRAM_DPM2.2_DIMENSIONAL. REPRESENTA LA ASOCIACIÓN DE TABLAS DEL MODELO DIMENSIONAL DEL DPM. **58**

FIGURA 27. EJEMPLO DE REPRESENTACIÓN DE CONCEPTOS CONTABLES EN EL EJE X EN EL TIEMPO EJE Y Y SU MAGNITUD EJE Z. REFERENCIA [9]..... **60**

FIGURA 28. EJEMPLO DE REPRESENTACIÓN DE DIMENSIONES PADRE EN EL EJE X CON LAS DIMENSIONES HIJO EN EL EJE Y Y SU MAGNITUD EJE Z DE TIPO MONETARIA. REFERENCIA [9] **60**

FIGURA 29. DIAGRAM_DPM2.2_TEMPLATES. REPRESENTACIÓN DE DIMENSIONES, EJES Y PUNTO DE DATO..... **62**

FIGURA 30. EJEMPLO DE REPRESENTACIÓN DE LAS TABLAS AXISORDINATE Y HIERARCHYNODE **63**

FIGURA 31. REPRESENTACIÓN DE ENUMERACIONES DE CAMINOS VÁLIDOS Y NO VALIDOS EN UNA ESTRUCTURA PADRE-HIJO. **64**

TABLA 1. EJEMPLO ESTRUCTURA EN ÁRBOL VALIDA. **64**

TABLA 2. ESTRUCTURA TABLA AXIS. REFERENCIA [1]..... **65**

TABLA 3. ESTRUCTURA TABLA AXISORDINATE. REFERENCIA [1] **66**

TABLA 4. ESTRUCTURA TABLA CONTEXTDEFINITION. REFERENCIA [1]..... **66**

TABLA 5. ESTRUCTURA TABLA CONTEXTOFDATAPOINTS. REFERENCIA [1] **67**

TABLA 6. ESTRUCTURA TABLA DATAPOINT. **67**

TABLA 7. ESTRUCTURA TABLA DATAPOINTVERSION. REFERENCIA [1] **68**

TABLA 8. ESTRUCTURA TABLA DATATYPE. REFERENCIA [1] **68**

TABLA 9. ESTRUCTURA TABLA DIMENSION. REFERENCIA [1]..... **69**

TABLA 10. ESTRUCTURA TABLA DIMENSIONCOORDINATE. **69**

TABLA 11. ESTRUCTURA TABLA DOMAIN. REFERENCIA [1]..... **70**

TABLA 12. ESTRUCTURA TABLA FLOWTYPE. **70**

TABLA 13. ESTRUCTURA TABLA HIERARCHY. REFERENCIA [1]..... **71**

TABLA 14. ESTRUCTURA TABLA HIERARCHYNODE. REFERENCIA [1] **72**

TABLA 15. ESTRUCTURA TABLA MEMBER. REFERENCIA [1] **72**

TABLA 16. ESTRUCTURA TABLA METRIC. REFERENCIA [1] 73

TABLA 17. ESTRUCTURA TABLA MODULE. 73

TABLA 18. ESTRUCTURA TABLA MODULETABLEORGROUP. 73

TABLA 19. ESTRUCTURA TABLA MODULETABLEVERSION. 74

TABLA 20. ESTRUCTURA TABLA OPENAXISRESTRICTION. REFERENCIA [1] 74

TABLA 21. ESTRUCTURA TABLA OPENMEMBERRESTRICTION. REFERENCIA [1] 75

TABLA 22. ESTRUCTURA TABLA REPORTINGFRAMEWORK. REFERENCIA [1] 75

TABLA 23. ESTRUCTURA TABLA TABLE. REFERENCIA [1] 75

TABLA 24. ESTRUCTURA TABLA TABLECELL. REFERENCIA [1] 76

TABLA 25. ESTRUCTURA TABLA TABLEGROUP. REFERENCIA [1] 76

TABLA 26. ESTRUCTURA TABLA TABLEGROUPTEMPLATES. REFERENCIA [1] 77

TABLA 27. ESTRUCTURA TABLA TABLEVERSION. REFERENCIA [1] 77

TABLA 28. ESTRUCTURA TABLA TAXONOMY. REFERENCIA [1] 78

TABLA 29. ESTRUCTURA TABLA TEMPLATE. REFERENCIA [1] 78

FIGURA 32. MODELO DE DATOS COMPLETO. REFERENCIA [1] 79

FIGURA 33. CICLO DE VIDA. PROYECTO SW 81

FIGURA 34. FLUJO DE INSERCIÓN DE DATOS VALIDADOS DEL DPM 82

FIGURA 35. MODELO DE DATOS VALIDADO MEDIANTE LA HERRAMIENTA DE INTEGRACIÓN DE DATOS SQL SERVER INTEGRATION SERVICE. 83

TABLA 30. REGLAS DE VALIDACIÓN APLICADAS 84

FIGURA 36. FLUJO DE INSERCIÓN DE DATOS MEDIANTE SSIS. 85

FIGURA 37. ESTRUCTURA PAQUETE DTS. REFERENCIA 15 86

FIGURA 38. FLUJO DE EJECUCIÓN SOLUCIÓN SSIS EBA_XBRL_SSIS. 88

FIGURA 39. PARÁMETROS DE LA SOLUCIÓN. ARCHIVO PROJECT PARAMS. 89

FIGURA.40 CONFIGURACIÓN DE CONEXIONES EN LA SOLUCIÓN DE SSIS. 90

FIGURA.41 OPCIONES AVANZADAS. CONFIGURACIÓN DE CONEXIONES EN LA SOLUCIÓN DE SSIS. 91

FIGURA 42. MENSAJE TEST DE CONECTIVIDAD A LA BDD. 92

FIGURA 43. PANTALLA DE CONFIGURACIÓN DE CONEXIONES A FICHEROS EXCEL 93

FIGURA 44. PANTALLA DE CONFIGURACIÓN DE CONEXIONES A FICHEROS EXCEL 94

FIGURA 45. IS_MASTER_EBA.DTSX. PAQUETE DE INICIO. 95

FIGURA 46. FLUJO DE DATOS. CONEXIÓN AL FICHERO DE ENTRADA EXCEL. 98

FIGURA 47. FLUJO DE DATOS. CONEXIÓN AL FICHERO DE ENTRADA EXCEL. PREVIEW. 98

FIGURA 48. FLUJO DE DATOS. CONEXIÓN AL FICHERO DE ENTRADA EXCEL. COLUMNS 99

FIGURA 49. FLUJO DE DATOS. CONEXIÓN AL FICHERO DE ENTRADA EXCEL. ERROR OUTPUT 100

FIGURA 50. FLUJO DE DATOS. CONVERSIÓN DE DATOS DEL FICHERO DE ENTRADA. 101

FIGURA 51. FLUJO DE DATOS. TRANSFORMACIÓN DE DATOS DE COLUMNA DERIVADA. 102

FIGURA 52. FLUJO DE DATOS. DESTINO OLE DB. 104

FIGURA 53. FLUJO DE DATOS. DESTINO OLE DB. MAPPINGS. 105

FIGURA 54. FLUJO DE EJECUCIÓN. CONTROL DE ERRORES Y EJECUCIÓN DE UN PAQUETE DTS. 106

TABLA 31. ESTRUCTURA TABLA LOG_ERRORS. 107

FIGURA 55. FLUJO DE EJECUCIÓN. CONTROL DE ERRORES DE UN PAQUETE DTS. 109

FIGURA 56. FLUJO DE EJECUCIÓN. PARÁMETROS DEFINIDOS EN EL CONTROL DE ERRORES DE UN PAQUETE DTS. 110

FIGURA 57. FLUJO DE EJECUCIÓN. CONTROL DE FIN DE EJECUCIÓN DE UN PAQUETE DTS. 112

FIGURA 58. FLUJO DE EJECUCIÓN. CONTROL DE FIN DE EJECUCIÓN DE UN PAQUETE DTS. MAPPING 113

FIGURA 59. EJEMPLO LOG DE INSERCIÓN SATISFACTORIA EN LA TABLA LOG_ERRORS. 114

FIGURA 60. PROCESO DE VALIDACIÓN DE ERRORES EN EL CASO ESPECIAL DE TABLAS CON ESTRUCTURAS EN ÁRBOL. IS_HIERARCHYNODE.DTSX. 114

FIGURA 61. TEST1. PROCESO DE VALIDACIÓN DE LA TABLA CONCEPTOS. 117

FIGURA 62. TEST1. FICHERO DE ENTRADA MODIFICADO CON UN ERROR DE PK DUPLICADA. 117

FIGURA 63. TEST1. MENSAJE DE ERROR. 118

FIGURA 64. TEST2. PROCESO DE VALIDACIÓN DE LA TABLA MIEMBROS. 118

FIGURA 65. TEST2. PROCESO DE VALIDACIÓN DE LA TABLA MIEMBROS. ERROR INTEGRIDAD REFERENCIAL VIOLADA 119

FIGURA 66. TEST2. MENSAJE DE ERROR. 119

FIGURA 67. TEST4. PROCESO DE VALIDACIÓN DE LA TABLA DIMENSIONES. 120

FIGURA 68. TEST4. PROCESO DE VALIDACIÓN DE LA TABLA DIMENSIONES. ERROR PK CON VALOR NULL. 121

FIGURA 69. TEST4. FICHERO DE ENTRADA MODIFICADO CON UN ERROR DE PK NULL. 121

FIGURA 70. TEST4. MENSAJE DE ERROR. **121**

FIGURA 71. TEST5. PROCESO DE VALIDACIÓN DE LA TABLA HIERARCHYNODE. ESTRUCTURA ÁRBOL VIOLADA **122**

FIGURA 72. TEST5. PROCESO DE VALIDACIÓN DE LA TABLA HIERARCHYNODE. ESTRUCTURA ÁRBOL VIOLADA **123**

FIGURA 73. TEST5. FICHERO DE ENTRADA MODIFICADO CON UNA VIOLACIÓN DE LA ESTRUCTURA EN ÁRBOL. **123**

FIGURA 74. TEST5. MENSAJE DE ERROR. **123**

FIGURA 75. MODELO EN ESTRELLA. REFERENCIA [9] **124**

FIGURA 76. TABLAS MAESTRAS. TIPO DATOS, UNIDAD Y BALANCE. **126**

TABLA 32. DEFINICIÓN DATOS MAESTROS DE LA TABLA DE HECHOS. **127**

FIGURA 77. EJEMPLO DE TABLA DIMENSION GENERADA DE FORMA DINÁMICA **127**

FIGURA 78. EJEMPLO DE TABLA DE HECHOS GENERADA DE FORMA DINÁMICA, CON TODAS SUS REFERENCIAS. **129**

FIGURA 79.PASOS PARA LA GENERACIÓN DEL MODELO EN ESTRELLA PROPUESTO EN EL ARTÍCULO ... **130**

FIGURA 80. FLUJO DE DATOS IS_DIMENSION. EJECUCIÓN PROCEDIMIENTO SQL DE CREACIÓN DE LA TABLA DE HECHOS. **132**

FIGURA 81. DIAGRAMA DE BDD TABLA DE HECHOS Y REFERENCIA A LAS TABLA MAESTRAS. **133**

FIGURA 82. DIAGRAMA DE BDD TABLA DE HECHOS Y REFERENCIA A LAS TABLA MAESTRAS DE DIMENSIÓN. **137**

FIGURA 83. DIAGRAMA DE BDD DEL MDM. **140**

FIGURA 84. DIAGRAMA DE BDD DEL MDM EN LA SOLUCIÓN DE ANALYSIS SERVICE. **143**

FIGURA 85. DIAGRAMA DE BDD TABLA DE HECHOS Y REFERENCIA A LAS TABLA MAESTRAS DE DIMENSIÓN. **145**

1 Introducción

Este proyecto toma como punto de partida el Desarrollo realizado para representar el Modelo punto de datos (*Data Point Model, DPM*) creado por la Autoridad Bancaria Europea (*European Banking Authority, EBA*) y la representación de las validaciones técnicas de ejecución para proyectos en XBRL (Informes económico financieros semánticos).

Este documento tiene como objetivo proporcionar una introducción al tema de la creación de un modelo conceptual para el almacenamiento de datos multidimensional que es recibido como instancias XBRL que siguen las reglas definidas por taxonomías europeas publicadas por la EBA.

El Modelo de Datos Multidimensional (MDM) se presenta en este documento como prueba de concepto y está destinado a ser un punto de partida para un proceso de modelado posterior que será ajustado y ampliado a las necesidades analíticas o transaccionales específicos. Este Proyecto se ciñe únicamente a los conceptos del DPM y Taxonomías de Arquitectura XBRL, que construyen la base de la información europea de supervisión.

La estructura del modelo de datos se basa en clases de meta modelos.

En resumen, este trabajo propone un diseño de metadatos para los informes semánticos, y la creación de estos informes, desde el mundo real a través de la implementación física. Este trabajo de investigación estudia una parte de dicho modelo y se centrara en tres puntos:

- Migración del Modelo de datos de la EBA a un Modelo de datos en un Gestor de Base de datos más apropiado para este tipo de áreas de negocio.
- Creación de una interfaz de integración de los datos (incluidos en el DPM de la EBA) en el modelo creado previamente.
- Creación de un Modelo Multidimensional Estable y que puede ser explotada su información para la creación de reportes.

1.1 XBRL

Conocido por su acrónimo XBRL (*eXtensible Business Reporting Language*), este estándar nace de la propuesta lanzada en 1998 por *Charles Hoffman*, experto contable y auditor, para simplificar la automatización del intercambio de información financiera mediante el uso del lenguaje XML.

El aspecto más novedoso en XBRL es la estandarización, su meta es uniformizar la información financiera a nivel global. El concepto de taxonomía como indicador de las líneas maestras sobre las que se tiene que fundamentar el intercambio de información, hace que el tratamiento de los datos se simplifique enormemente.

Sobre estas taxonomías, podemos construir los informes, con los datos concretos o hechos económicos para un momento dado del tiempo.

1.2 Introducción a los informes semánticos. Meta Modelo XBRL.

El meta modelo engloba el universo del discurso, que es un conjunto de datos que el Regulador o Supervisor quiere obtener de las entidades financieras. El “Universo del discurso” puede definirse como una descripción abstracta y general de la parte o sector de un universo real y que será posible representar por un modelo de base de datos. En este nivel de análisis se está tratando con una descripción de la realidad, no con datos, y suele contener listas de tipos de entidades, de las relaciones existentes entre esas entidades y de las restricciones de integridad que se aplican sobre ellas.

Podemos considerar que el Meta modelo incluye las distintas maneras en que los individuos conectan sus pensamientos usando el lenguaje.

El DPM es un modelo lógico. El UD es el conjunto de requerimientos, los datos que quieren obtener los Supervisores o Reguladores. La implementación física son las taxonomías XBRL.

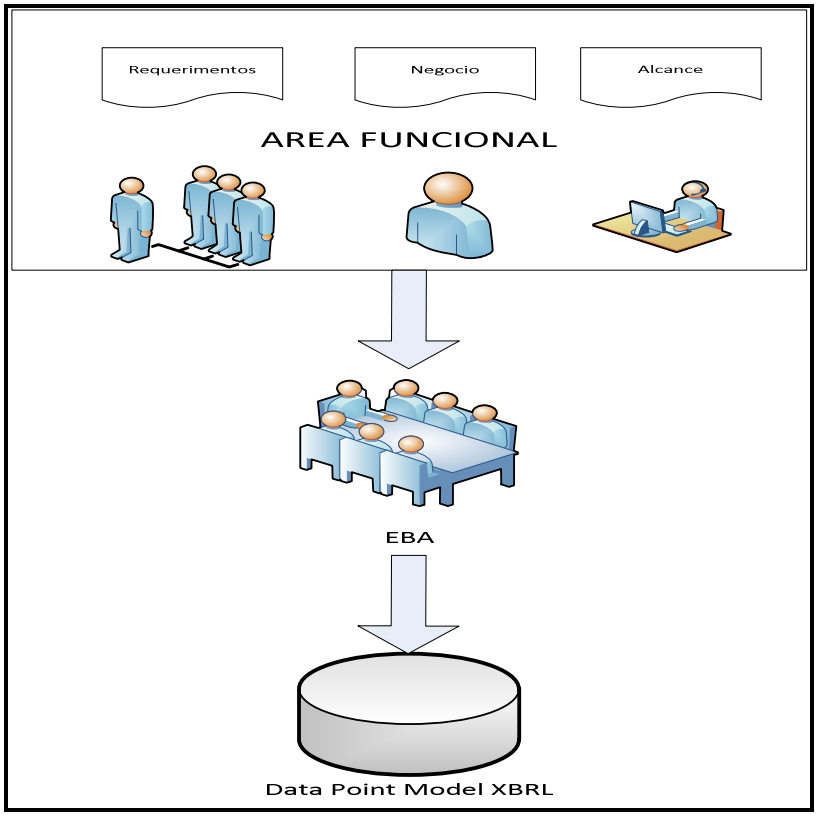


Figura 1. Esquema generación DPM XBRL.

El Meta-Meta modelo tiene su origen en el área Funcional una vez definido se abstrae a Metodología XBRL el último paso del Análisis es generar el DPM. A partir de que entran en juego las diferentes Tecnologías Software para Validación, Almacenamiento y Explotación de la Información.

En este trabajo se señala que el meta modelo de datos XBRL, no es una tecnología sino una metodología y por lo tanto no es exclusiva de un ámbito o área de Negocio Concreta. En este Proyecto final de carrera se centra en el DPM basado en un Meta modelo de ámbito económico-financiero, pero puede extrapolarse a otras áreas de una Organización o empresa, con esto quiero decir que XBRL tiene amplias posibilidades y se centra en la definición de reglas y buenas practicas a la hora de plasmar el universo del Discurso.

1.3 Objetivos.

El objetivo de este Proyecto es proporcionar un punto de partida en el tema referente a las validaciones del DPM y estructuras *XBRL* primeramente en un modelo multidimensional e implementarse en un modelo *ROLAP*. Sobre la base de un ejemplo fácilmente comprensible, cuestiones más complejas se deben tratar y habría que tenerlo en cuenta al definir una Solución para su uso en el mundo empresarial o institucional.

La versión analizada es la del DPM de la EBA versión 2.2. La EBA decidió mejorar aún más la aplicación del enfoque metodológico, introduciendo una base de datos relacional como repositorio para los metadatos DPM, en lugar de confiar únicamente en las estructuras de datos de MS Excel. Para mayor comodidad y por razones de comunicación, MS Access fue elegido para apoyar la difusión de esta base de datos.

En este análisis está Relacionado con este modelo Relacional bajo una Infraestructura en SQL Server, el enfoque está dirigido a mejorar las reglas de validación y el Modelo propuesto por la EBA, así como realizar un nuevo Modelo Multidimensional a partir de este con el fin de mejorar el desarrollo de informes de modelos económico financieros.

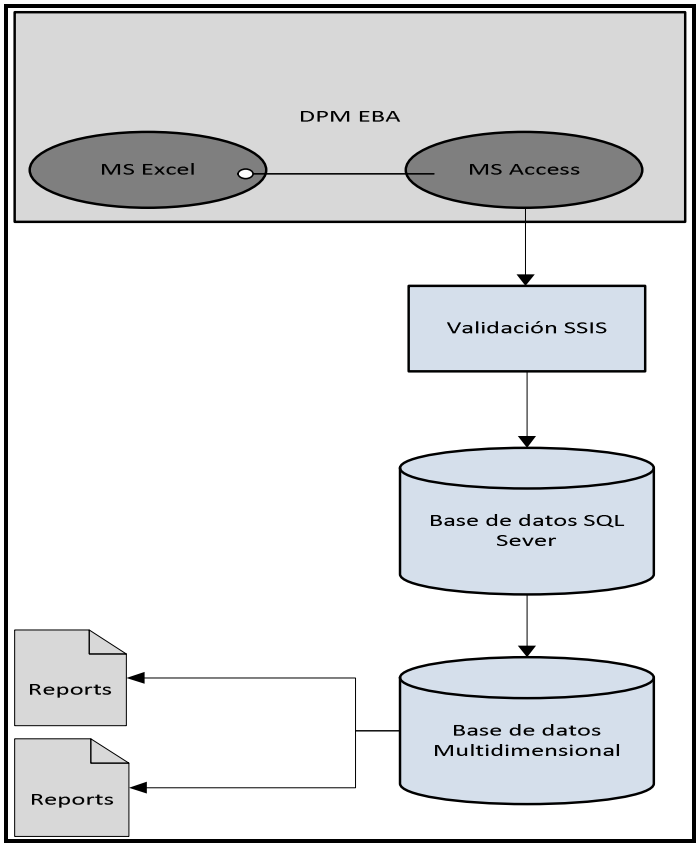


Figura 2. Esquema Proyecto. (En Azul)

1.4 Alcance del Desarrollo y medios empleados.

Este documento está dirigido a los usuarios de las taxonomías de supervisión europeos que tienen la necesidad de almacenar datos de informes basados en estas definiciones de datos y recuperarlos para fines analíticos o transaccionales. Expertos de bases de datos deben recibir información detallada acerca de los detalles a tener en cuenta a la hora modelar estructuras de bases de datos multidimensionales para almacenar datos de supervisión basada en XBRL. Por lo tanto, la audiencia de este documento puede ser instituciones, organismos financieros o económicos o universidades con la intención de proporcionar un análisis micro o macro prudencial en los datos de supervisión.

El DPM se trata de un modelo Relacional en MS Access y parte de Bases de datos jerárquicas. En este proyecto parto ya de este Modelo y las fases en las que se ha centrado es en:

- Validación del Modelo Conceptual y Modelo Lógico.
- Crear una Base de datos multidimensional.

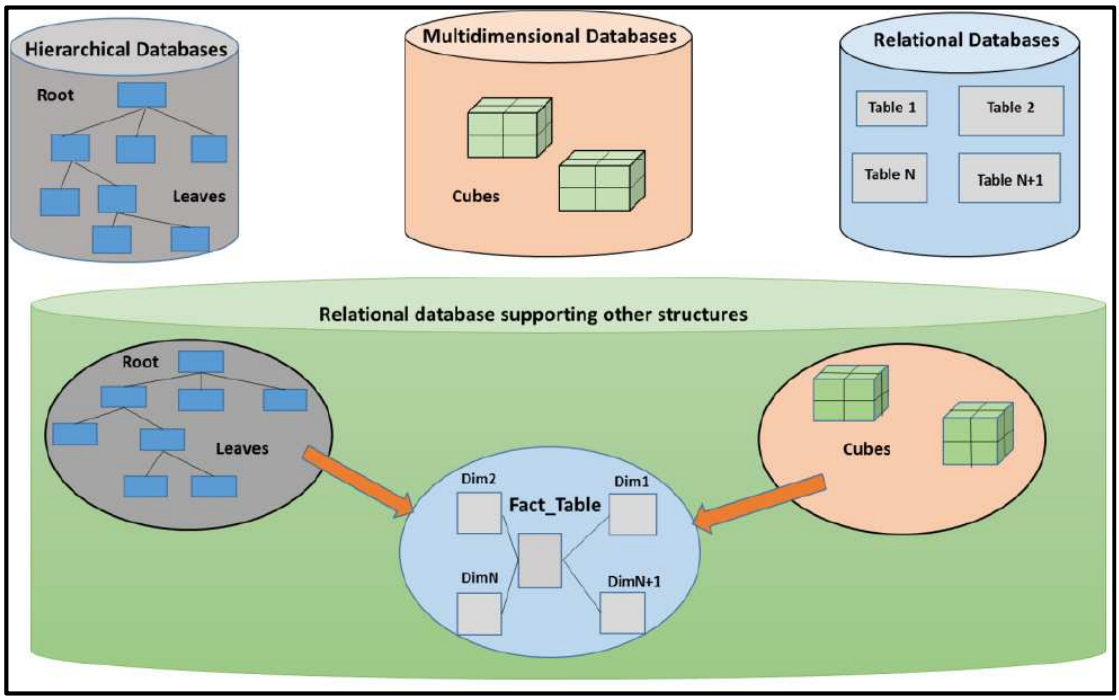


Figura 3. Diferentes tipos de DBMS's. Referencia [2]

El alcance final es llegar a una prueba de concepto de Bases de datos multidimensionales cuyo origen es un Modelo relacional validado a través de otro Modelo relacional creado por la EBA. Que a su vez tiene como origen un modelo Jerárquico.

Las Bases de datos Multidimensionales pueden o no estar implementadas en bases de datos relacionales. Su potencia reside en que tienen los datos almacenados en una matriz de almacenamiento multidimensional optimizado, y no en un formato relacional. Sin

embargo, es necesario organizar la información en un cubo de antemano. Estas bases de datos tienen tiempos de respuesta muy rápidos en las consultas. Los ejemplos de las bases de datos multidimensionales son: *Essbase*, *icCube*, Infor *BI OLAP Server*. En mi caso he utilizado SSAS Modulo que forma parte de la solución de *Microsoft SQL SERVER 2012*.

1.5 Relación con otros trabajos.

El punto de partida de este proyecto se sitúa el 24 de Noviembre en Bruselas, en la semana XBRL en Bruselas (*XBRL week in Brussels*). Referencia [12]

En esta semana Ignacio Santos y Abel Nieto, presentamos un estudio en conjunto llamado "*Formal Validation of Data Point Models*" (validación formal de modelos de punto de datos). Dicho estudio se trataba de una prueba de concepto acerca de la validación del DPM de la EBA pero en una versión anterior a la del estudio de este proyecto, en el estudio presentado en Bruselas la versión existente era la 2.0, la que se analiza en este proyecto es la 2.2. El documento de mejoras y cambios se encuentra en la Sección de anexos (Control de versiones DPM EBA).**La información relativa a este Estudio se encuentra en el sitio web de *Openfilling*.** Referencia [7]

La segunda parte de este Proyecto se presentó en la semana XBRL en Madrid en el banco de España, de igual forma Ignacio Santos y Abel Nieto, Presentamos un estudio pero este Basado en el paso del Modelo Relacional definido por la EBA a un Modelo de datos Multidimensional creado de forma dinámica. Referencia [13]

2 Requerimientos Software/Hardware

Las especificaciones Hardware y software con las que se ha llevado a cabo el proyecto son las siguientes.

Las licencias del Software empleadas han sido descargadas de la cuenta de MSDN, de la cual es propietario yo mismo Abel Nieto Cano.

2.1 Requerimientos Hardware

La máquina y SO con la que se han realizado las pruebas de concepto es:

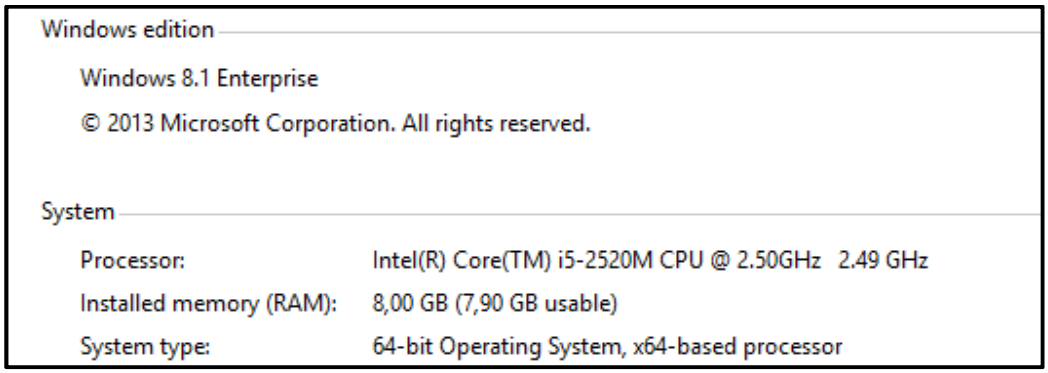


Figura 4. Requerimientos Hardware.

2.2 Requerimientos Software

Es necesario tener instalados los siguientes programas para el desarrollo y ejecución de código:

SQL Server 2012 Enterprise o Profesional.

- Para la Generación del Modelo Access del DPM EBA a SQL Server 2012.

  - Aplicación de SSIS *import and Export Data (32-bit)*

- Para la validación del Modelo conceptual del DPM EBA.
  - Para convertir el modelo *import and Export Data (32-bit)*
  - Instalación del Módulo de *Integration Service* del Motor de BDD de MS SQL SERVER. SQL Server Data Tools. SSDT.

- Para la generación del Modelo en estrella.
  - Instalación Módulo de *Analysis Service* del Motor de BDD de MS SQL SERVER

Microsoft Office.

3 Diseño de los metadatos de los informes

En esta sección se asigna el modelo relacional de la DPM suministrada por la EBA

La EBA publicó su modelo de meta el 15 de marzo de 2013, y después de varias modificaciones de la versión final el 29 de noviembre de 2013 para el año de referencia 2014. La publicación contiene una versión actualizada de las plantillas, instrucciones, reglas de validación y punto de datos modelo para la aplicación de técnicas Normas (ITS) en la presentación de informes de supervisión, COREP y FINREP. En ese mismo tiempo EBA publica la base de datos de DPM 0.1.1 como una estructura meta modelo utilizado como repositorio para todos los metadatos se define en la DPM de la que se derivan las taxonomías XBRL

3.1 Meta modelo definido por la EBA (FINREP y COREP).

La base de datos se construye a partir del DPM del EBA, se trata de un archivo Zip que se puede descargar de la Página web de la EBA.

Figura 5. Página Web EBA. Referencia [1]

Para una mejor comprensión de la aplicación pueden usar otros Sistemas gestores de Base de datos como en *MS SQL Server*, *DB2* u *ORACLE*. Sin embargo, el traslado a otro SGBD (Sistema gestor de Base de datos) es fácil, porque ANSI-SQL es un estándar. En el primer paso, se crea la estructura de la DPM, en este caso *MS SQL Server*. El segundo paso consiste en rellenar la DPM en la base de datos con el modelo de datos de la EBA (DPM 0.1.1 Base de datos) a través de un ETL (*Extract, Transform and Load*) herramienta.

La versión EBA para este ejemplo no contenía ningún documento XBRL de instancia, lo que hizo imposible llenar la tabla de hechos con un ejemplo, pero la estructura de la DPM se ha completado.

3.2 Creación de la estructura y la carga de la DPM de la EBA en un modelo relacional PLSQL

El archivo Zip con la estructura del modelo de metadatos, DPM Base de datos 0.1.1 está disponible para su descarga desde la página web EBA. Cuando se obtiene este archivo, su estructura y los datos son trasladados y validados a una Base datos SQL SERVER. En este documento se utiliza MS SQL Server (la edición gratuita, Microsoft® SQL Server®2012). Sin embargo, es posible utilizar otros de RDBMS, como Oracle, DB2, etc. A los efectos de este documento, Integration Services (IS) de MS SQL Server (hay una edición gratuita) se utiliza para pasar los datos y estructuras de MS Access a MS SQL Server. En esta herramienta, la fuente de datos es MS Access. La figura 6 muestra una vista general de la carga de acceso en un SGBD y el mapeo de DPM en una base de datos relacional.

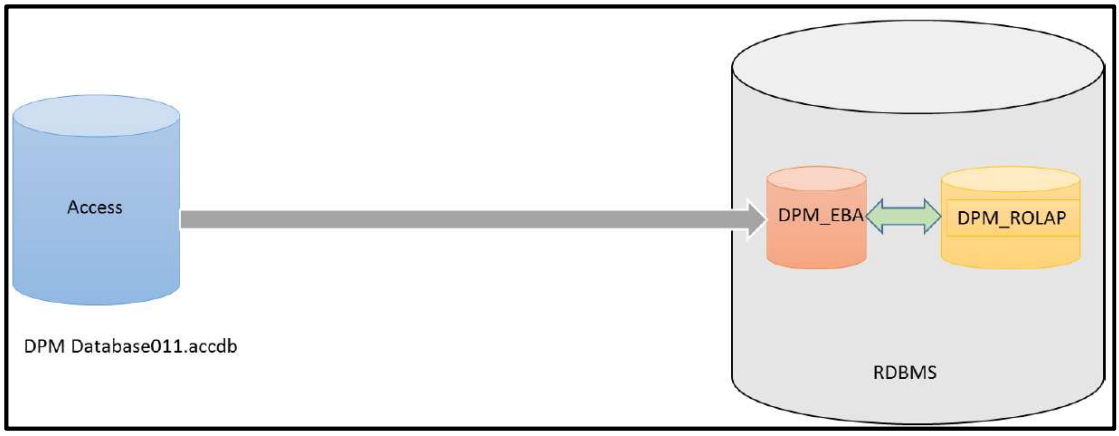


Figura 6. Modelo propuesto por la EBA. Referencia [2]

Una de las principales ventajas de este componente técnico es imponer una serie de restricciones lógicas en el modelo, y permitiendo la realización de una serie de comprobaciones de coherencia automáticos que no serían posibles de otro modo, contribuyendo así decisivamente a acortar el tiempo necesario para alcanzar el nivel de calidad deseado, de una DPM que categoriza más de 30.000 puntos de datos.

Otra ventaja considerable de la base de datos es la posibilidad de definir muchos puntos de vista diferentes sobre el mismo contenido de metadatos, de acuerdo con las necesidades del usuario que está tratando de comprender la estructura de información, y el vínculo entre

las plantillas de negocios y los puntos de datos dimensionales, que ahora se definen explícitamente en la DPM.

El modelo de base de datos Access propuesto por EBA, es un meta-modelo, con el fin de ser utilizados en cualquier dominio de informes que no sea COREP / FINREP, con un nivel relativamente bajo de abstracción, centrándose directamente en los principales conceptos que se utilizan en el modelado punto de datos (por ejemplo, marco, Tabla, celda de tabla, dimensión, miembro, dominio...). En cuanto a los conceptos dimensionales, que, básicamente, tienen las mismas definiciones que se encuentran en los sistemas de análisis, lo que hace posible una conexión muy directa entre ambos extremos de la cadena de información.

El análisis de este proyecto se centra en la taxonomía FINREP.

El meta-modelo no está vinculado a ninguna tecnología en particular, y por lo tanto a limitaciones específicas de XBRL, no se reflejan en el DPM ya que reduciría la claridad de la modelo. Con el fin de agilizar el proceso de traducción automática del DPM a las taxonomías XBRL, sin embargo, se han añadido algunos elementos adicionales del modelo, y varios campos que contienen propiedades específicas de XBRL (por ejemplo, códigos) están incluidos.

En comparación con la primera versión de la meta-modelo expresado en la base de datos publicada en noviembre de 2013, (referencia [7]) esta versión del modelo contiene algunos pequeños cambios, sobre todo alrededor de la representación de las tablas y Plantillas y los vínculos con los informes / módulos y taxonomías, y la representación de cambios entre versiones de taxonomías.

En particular, en esta versión de la evolución de los datos declarados, y en unos pocos casos se expresa la re-expresión de la información conceptualmente idéntica usando categorizaciones, y la indicación de la continuidad nocional de esta serie de datos a través de la re-categorización. Esto es para permitir la comprensión de la historia de puntos de datos únicos que es un requisito fundamental para el almacenamiento de datos y análisis de series de tiempo.

4 Sistema Gestor de Base de datos. Estructura de la base de datos.

El meta-modelo se estructura básicamente en torno a la representación de los metadatos de las Plantillas, los conceptos dimensionales utilizados para categorizar los datos, y los vínculos entre ellos, que es la clasificación actual.

El punto de partida es la Base de datos Access DPM Database.2.2.0.accdb descarga de la web de la EBA.

La figura 7 representa el EBA DPM.

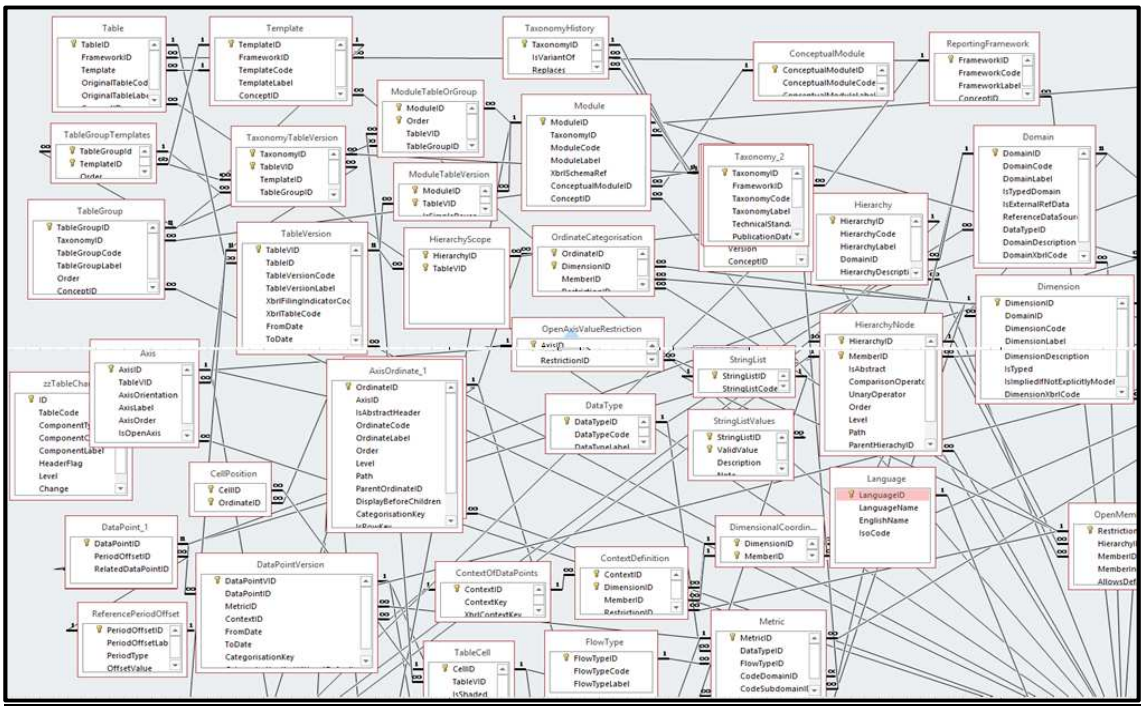


Figura 7. Parte del EBA DPM

El EBA DPM. Se trata de un modelo relacional, pero al haber empleado un motor de Base de datos como Access, carece de muchos aspectos técnicos de buenas prácticas en el

modelo de bases de datos. Algunas concernientes a la estructura del modelo, se puede apreciar en la figura un esquema muy caótico y en otros aspectos relacionados con validaciones, normalización de tipos de datos y campos, rendimiento y administración de la Base de datos.

4.1 Elección del modelo de datos propio.

Debido a que el SGBD (Sistema Gestor de Base de datos) empleado por la EBA para generar su Modelo de datos no es el más idóneos, decidimos Exportar este Modelo a un Sistema Gestor de Bases de datos con mejores características de diseño y rendimiento. Se eligió SQL Server 2012, a pesar de ser dos SGDB, Access y SQL server son muy distintos. Entre sus principales similitudes podemos encontrar:

1. Ambos son sistemas de almacenamiento de datos en forma de tablas relacionales.
2. Ambos admiten comandos en lenguaje SQL, aunque el de SQL Server está mucho más desarrollado, ya que realmente es la interfaz del servidor con el cliente.
3. Ambos son sistemas creados por Microsoft.

Aunque Las diferencias entre estos Sistemas de almacenamiento son más profundas y La siguiente tabla evidencia una comparativa entre MS Access y MS SQL Server:

Característica	MS Access	MS SQL Server
Nº de Procesadores en Paralelo	1	16
Instancias de servidores sobre un ordenador	Ninguna	Ilimitado
Procesamiento de consultas más rápido	bajo	Muy Alto
Arquitectura cliente-servidor	No	Sí
Consola o interfaz para administrar la base de datos	Si	Si
Escalabilidad	Ninguna	Si, Procesadores Multisimétricos
Nº de Usuarios conectados	255	Ilimitados
Limite de usuarios concurrentes	255	Ilimitados
Seguridad integrada	Si	Si, A nivel de SQL y de sistema operativo
Admite triggers	No	Si
Admite procedimientos almacenados	No	Si
Relaciones de tablas e integridad de referencia	Si	Si
Funciones Meta datos	No	Si

Figura 8. Comparativa MS Access vs MS SQL Server. Referencia [14]

La primera diferencia a mencionar es el enfoque de cada uno de éstos. *Access* está más enfocado en “uso de escritorio” o personal, de pequeña y mediana Empresa, pensado para manejarse desde un solo equipo (máximo una pequeña red local o Hacia el usuario experto no informático). Cabe mencionar que no está pensado para que muchos clientes accedan al mismo tiempo a las bases de datos. Por otro lado, SQL Server permite el acceso a las bases de datos a miles de usuarios simultáneamente, y no sólo eso, sino que también permite almacenar grandes volúmenes de datos (*Terabytes*) que incluyen elementos como fotografías, videos, textos, números, etc., con millones de registros.

Otra diferencia entre estos Sistemas de almacenamiento se aprecia a la hora de publicar bases de datos en Internet; *SQL Server* es autónomo, mientras que Access no lo es.

Otra diferencia Sustancial es que SQL Server Permite Instalar herramientas como:

- SSIS (*SQL Server Integration Service*), Para Validación e Integración de datos.
- DQS (*Data Quality Service*), calidad de datos.
- MDS (*Master Data Service*), consolidación de Datos Maestros.
- SSAS (*SQL Server Analysis Service*) para realizar análisis Multidimensionales de los datos y posteriormente ser Explotada la Información mediante Reporting Service.
- SSRS (*SQL Service Reporting Service*) Herramienta de Modelado de Informes. Se puede Atacar a un SSAS o directamente a la BDD.

Fases de adaptación y migración por las que va a pasar el Modelo de datos de la EBA.

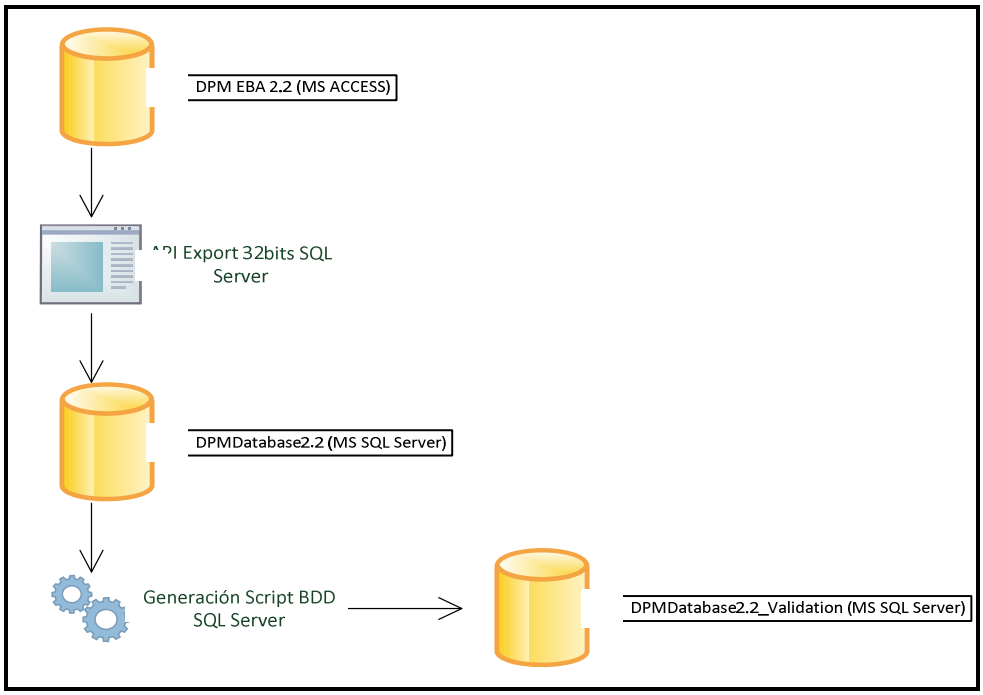


Figura 9. Flujo de la BDD Access a SQL Server

4.2 API Export de las Entidades, atributos y datos de la EBA a SQL Server 2012.

El propósito era reutilizar lo máximo posible el trabajo realizado por la EBA por lo que decidí utilizar la aplicación *Import & Export Data* (32 Bits) para la creación de la tablas y subtablas del DPM EBA, de esta forma creamos las Estructuras de datos pero sin relaciones por lo que está exento de FK (*Foreign Key*) y ninguna tabla tiene *Primary Key*. Este Paso Previo sirve para poder generar un Script de Base de datos Vacío, con las estructuras básicas, campos y atributos del modelo que vamos a crear.

Este es el primer paso para generar el modelo relacional en SQL Server 2012, con el mínimo impacto sobre las entidades y criterios de la EBA.

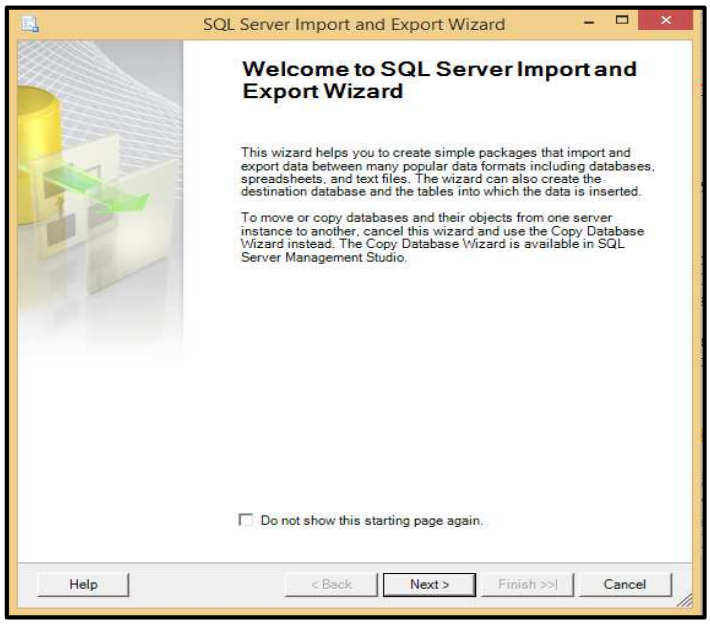


Figura 10. Import and Export. Exportación de la BDD Access a SQL Server

Esta es la pantalla de inicio para realizar la exportación de datos de MS Access a MS SQL Server.

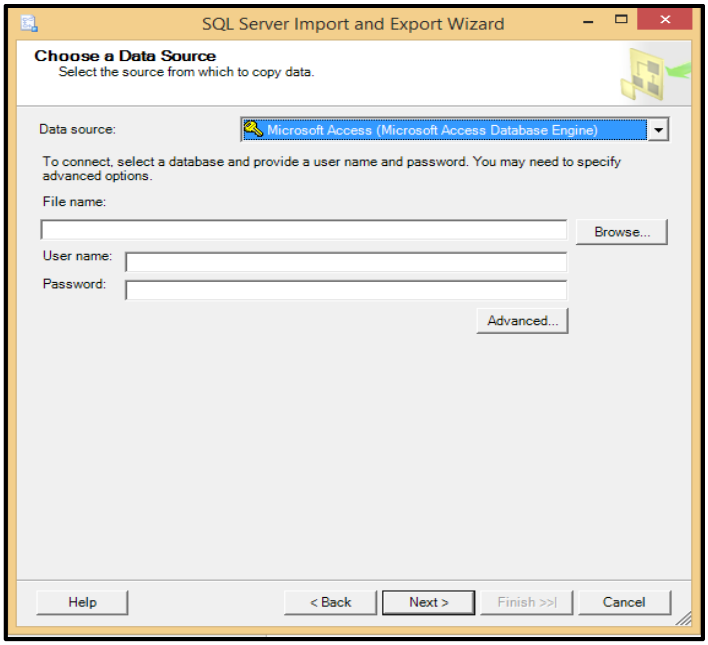


Figura 11. Import and Export. Data Source de la BDD Access a SQL Server

En esta pantalla se elige el Origen de datos, En *File name* se selecciona el fichero descargado de la EBA "**DPM Database.2.2.0.accdb**" que es el que posee el modelo de datos a Exportar.

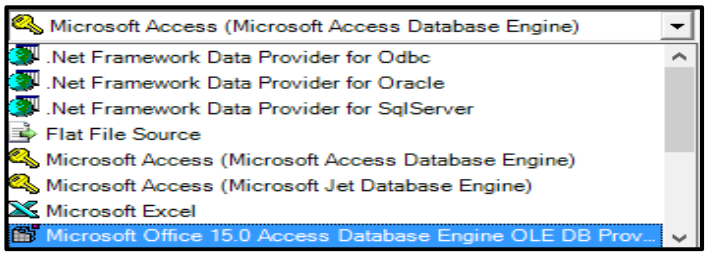


Figura 12. Import and Export. Lista de Data Sources disponibles para Exportar.

Hay varias opciones aparte de los orígenes MS Access.

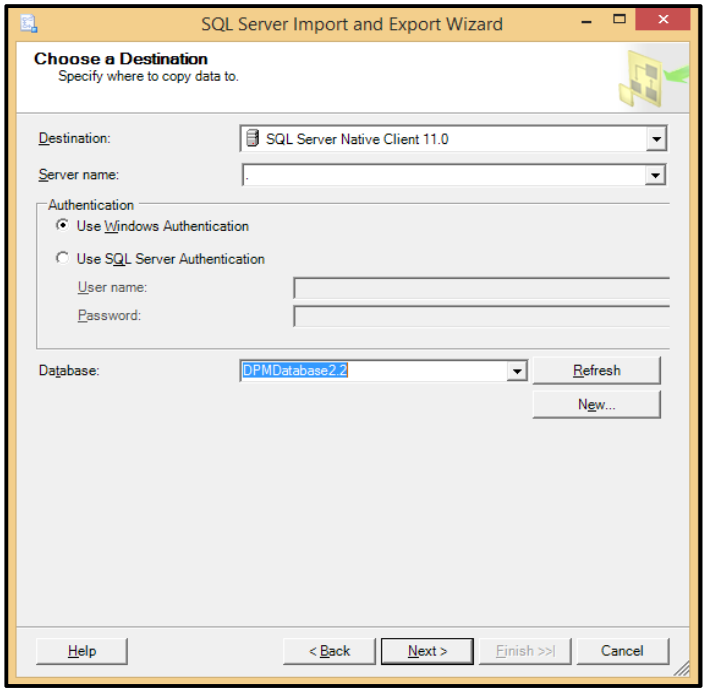


Figura 13. Import and Export. Destino BDD SQL Server.

El destino elegido es SQL Server en Modo Nativo 11.0, la conexión la realiza con el componente ODBC para Importación a *SQL server 2012*.

El parámetro servidor es mi maquina Local → "."

El parámetro *Database* es la BDD que he creado en Mi maquina desde el SQL Server Management Studio.

```
USE [master]
GO

/****** Object:  Database [DPMDatabase2.2]    Script Date: 03/09/2015 12:07:54
******/
CREATE DATABASE [DPMDatabase2.2]
 CONTAINMENT = NONE
 ON  PRIMARY
```

```
( NAME = N'DPMDatabase2.2', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL11.MSSQLSERVER\MSSQL\DATA\DPMDatabase2.2.mdf' , SIZE = 233472KB , MAXSIZE
= UNLIMITED, FILEGROWTH = 1024KB )

 LOG ON

( NAME = N'DPMDatabase2.2_log', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL11.MSSQLSERVER\MSSQL\DATA\DPMDatabase2.2_log.ldf' , SIZE = 7616KB ,
MAXSIZE = 2048GB , FILEGROWTH = 10%)

GO
```

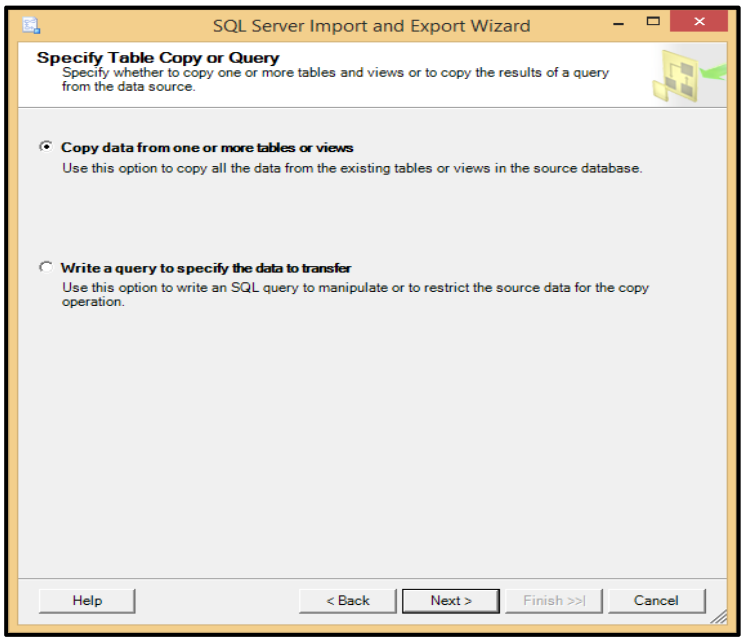


Figura 14. Import and Export. Especificación de tabla a BDD SQL Server.

El objetivo es copiar todos los datos y estructuras sin ningún tipo de restricción.

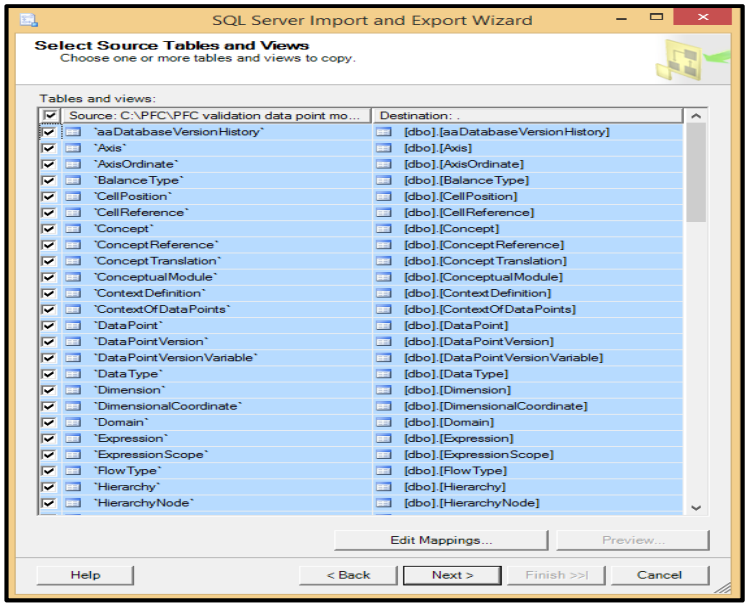


Figura 15. Import and Export. Tablas y vistas desde Access hacia SQL Server.

Las tablas pertenecen a un marco de información (en la actualidad, ya sea COREP o FINREP); la mayor parte del tiempo el concepto de Tabla será el mismo que el del modelo del asunto, salvo cuando, por razones de modelado, una Plantilla ha tenido que ser normalizada, ya sea porque el tipo de datos utilizado en Access no está soportado en SQL Server o no es el idóneo.

El objeto del análisis y las validaciones de datos efectuadas no engloba todo el conjunto del Modelo de datos creado por la EBA. Pero decidimos exportar el Modelo entero, en parte porque a priori las relaciones entre las diferentes entidades no eran del todo conocidas.

En los siguientes diagramas creados, ya desde el modelo de Datos en SQL Server, se encuentra el conjunto de entidades objeto de las pruebas de validación (las flechas representan las relaciones que deben ser leídos como "pertenece a un objeto" - es decir, lo que indica una relación n a 1.)

Los diagramas creados se representan en las siguientes figuras y son parte del EBA DPM. Y serán detallados.

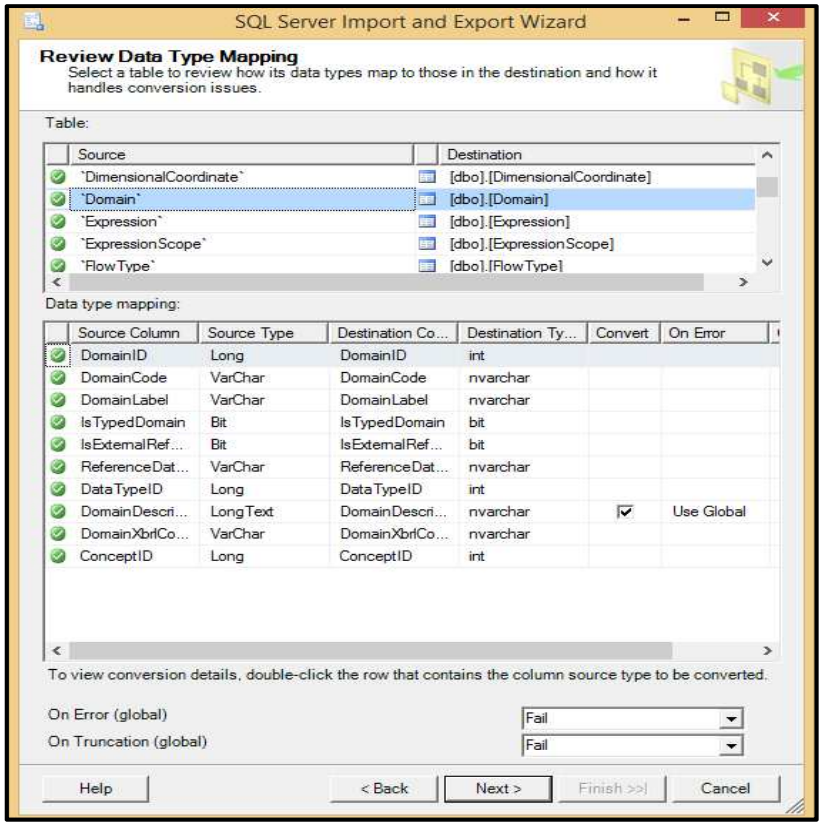


Figura 16. Import and Export. Mapping de atributos y tipos de datos desde Access a SQL Server

Dicho *Mapping* se realiza automática exceptuando para diferentes tipos de datos los cuales deben ser solucionados Manualmente. Los errores los proporciona la propia aplicación y deben ser solucionados para exportar los datos (fueron 32):

TITLE: SQL Server Import and Export Wizard

Column information for the source and the destination data could not be retrieved, or the data types of source columns were not mapped correctly to those available on the destination provider.

`Axis` -> [dbo].[Axis]:

- Column "AxisOrientation": Source data type "130" was not found in the data type mapping file.

`AxisOrdinate` -> [dbo].[AxisOrdinate]:

- Column "OrdinateCode": Source data type "130" was not found in the data type mapping file.

`DataPointVersionVariable` -> [dbo].[DataPointVersionVariable]:

- Column "VariableCode": Source data type "130" was not found in the data type mapping file.

`Dimension` -> [dbo].[Dimension]:

- Column "DimensionCode": Source data type "130" was not found in the data type mapping file.

`FlowType` -> [dbo].[FlowType]:

- Column "FlowTypeLabel": Source data type "130" was not found in the data type mapping file.

`HierarchyNode` -> [dbo].[HierarchyNode]:

- Column "ComparisonOperator": Source data type "130" was not found in the data type mapping file.

- Column "UnaryOperator": Source data type "130" was not found in the data type mapping file.

`OrdinateVariable` -> [dbo].[OrdinateVariable]:

- Column "VariableCode": Source data type "130" was not found in the data type mapping file.

`SpecificCellVariable` -> [dbo].[SpecificCellVariable]:

- Column "VariableCode": Source data type "130" was not found in the data type mapping file.

`VariableOfExpression` -> [dbo].[VariableOfExpression]:

- Column "VariableCode": Source data type "130" was not found in the data type mapping file.

…

…

…

`qDPM_zAxes` -> [dbo].[qDPM_zAxes]:

- Column "AxisOrientation": Source data type "130" was not found in the data type mapping file.

BUTTONS:

OK

Dicho *mapping* tiene las siguientes consideraciones:

- El tipo de datos *BIT* de *ANSI SQL* no se corresponde con el tipo de datos *BIT* de *Microsoft Access SQL*. En su lugar, se corresponde con el tipo de datos *BINARY*. No hay equivalente en *ANSI SQL* para el tipo de datos *BIT* de *Microsoft Access SQL*. En estos casos se ha relacionado como *Bolean*.
- *TIMESTAMP* ya no se admite como sinónimo de *DATETIME*.
- Si se usa el nombre de tipo de datos *TEXT* sin especificar la longitud opcional, por ejemplo *TEXT* (25), se crea un campo *Nvarchar* (25). Esto permite escribir instrucciones *CREATE TABLE* que produzcan tipos de datos coherentes con *Microsoft SQL Server*. En los casos que no se especifica longitud o de tipo *LONG TEXT*, da error, en esos casos los he creado como *nvarchar* (MAX) en SQL Server.
- Los campos *CHAR* siempre se almacenan en el formato de representación Unicode, que es el equivalente del tipo de datos *CHAR* de *ANSI SQL*.

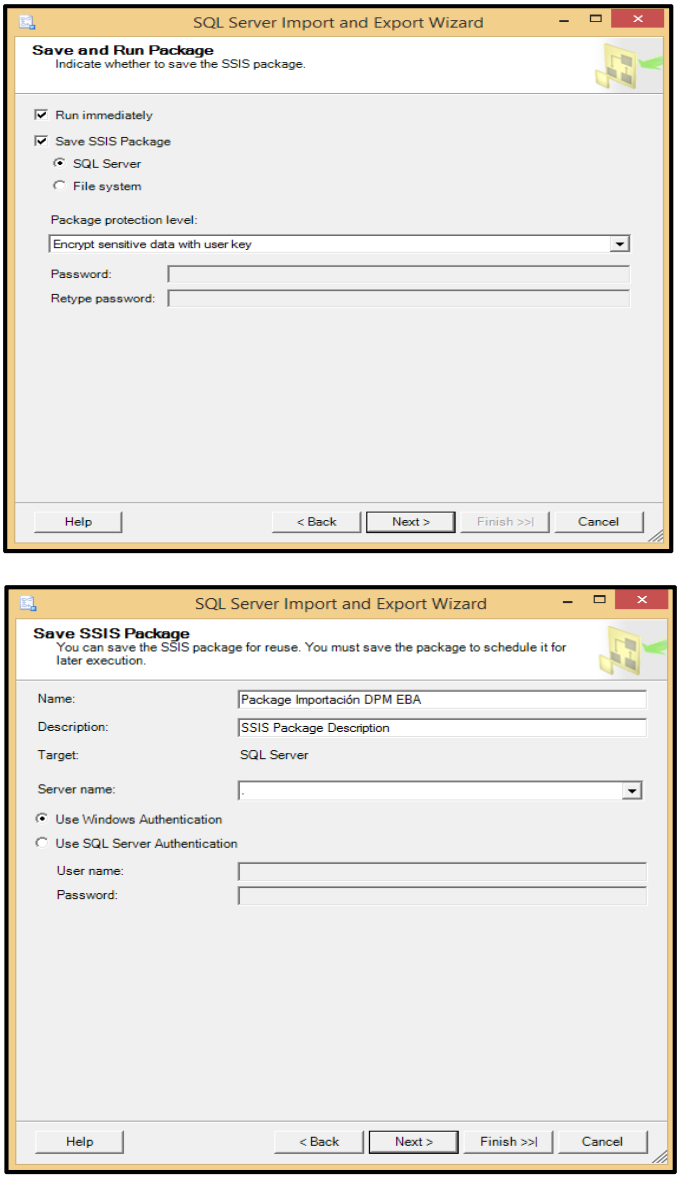


Figura 17. Import and Export. Guardar Proyecto en Paquetes ETL-SSIS

Una vez resuelto es posible guardar el proyecto en una solución SSIS para realizar más ejecuciones de la Exportación de este origen de datos en otro destino de BDD *SQL Server*.

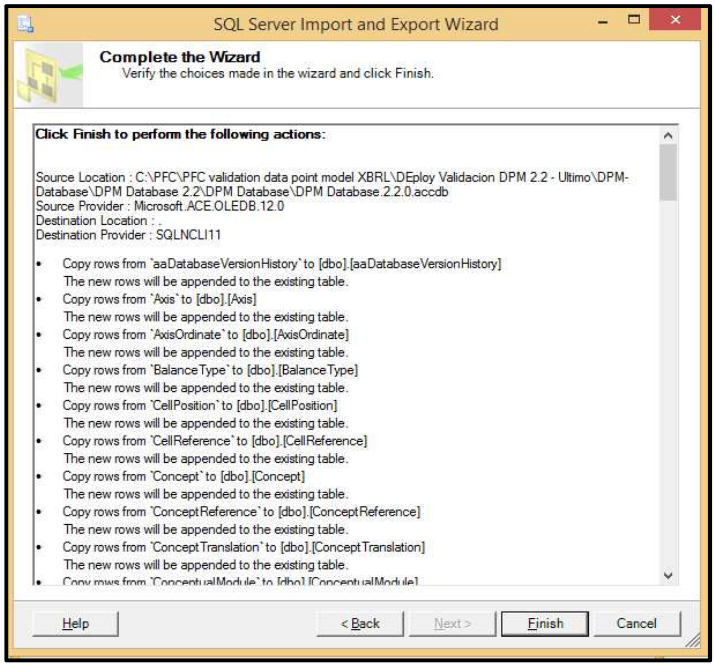
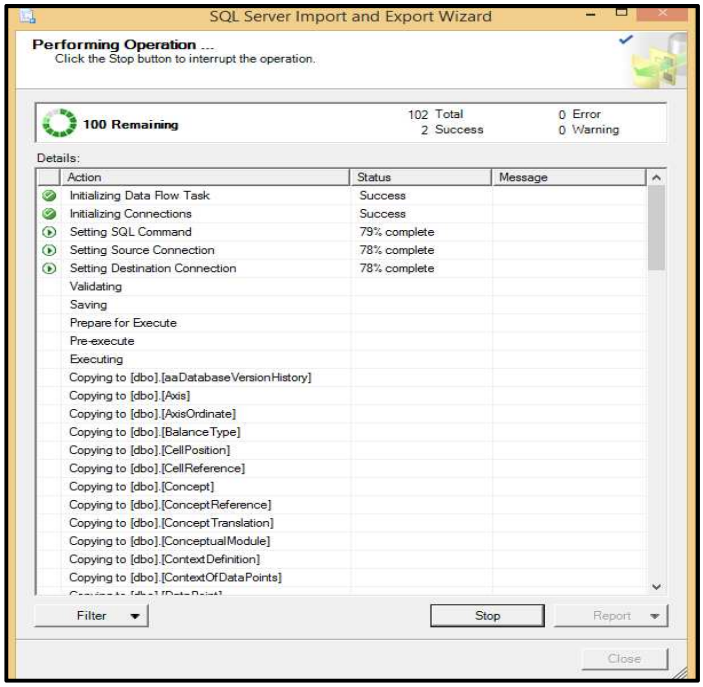


Figura 18. Import and Export. Verificación de las estructuras de datos.



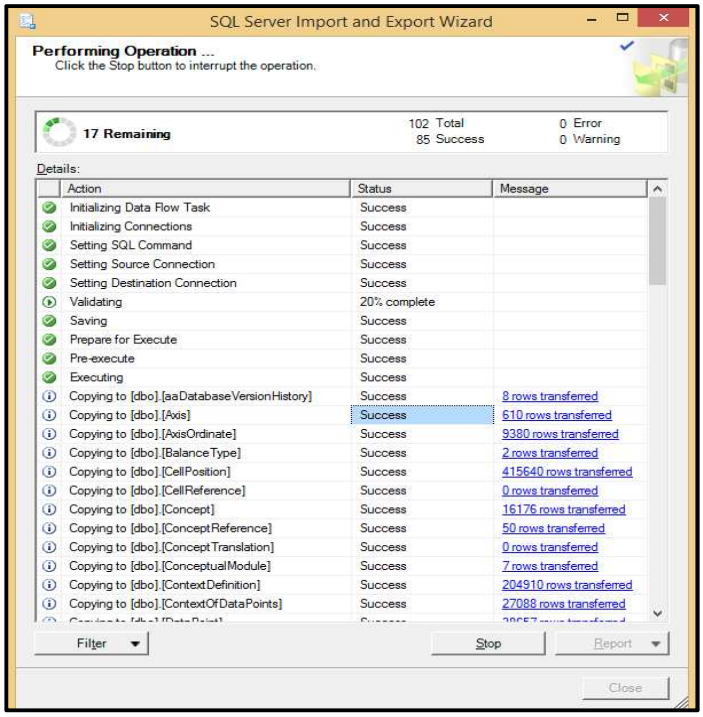


Figura 19. Import and Export. Ejecución proceso de exportación de las estructuras de datos.

Al final del proceso tenemos las Estructuras de datos Inicialmente en *Access* Exportadas en una BDD *SQL Server*.

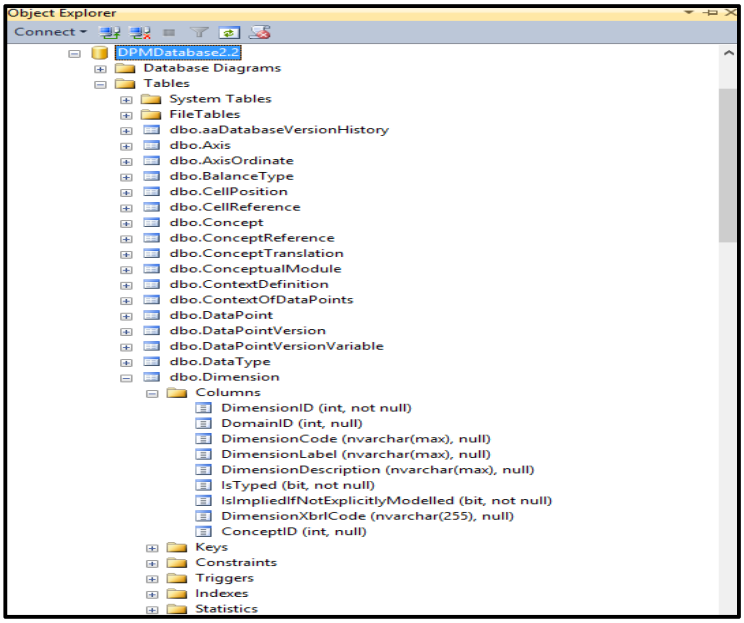


Figura 20. Import and Export. Estructura de datos EBA en SQL Server 2012.

Dicha estructura posee los datos Reales económico Financieros prefijados Por la EBA. Pero no existe un modelo Relacional como tal, ni los datos han sido validados en su Inserción.

4.3 Creación de la Base de datos en SQL Server 2012

Una vez migradas las estructuras de datos de Access a *SQL Server*, ya es Posible Generar un *Script* en *T-SQL* para migrar esta estructura de tablas en otro entorno.

La generación del Script se realiza mientras un componente de SQL Server. Tal y como detallo a continuación.

Botón derecho sobre la BDD DPMDatabase2.2, que es sobre la que hemos migrado la estructura de datos del *Access*. A continuación en *Task → Generate Script*. Aparece el siguiente asistente.

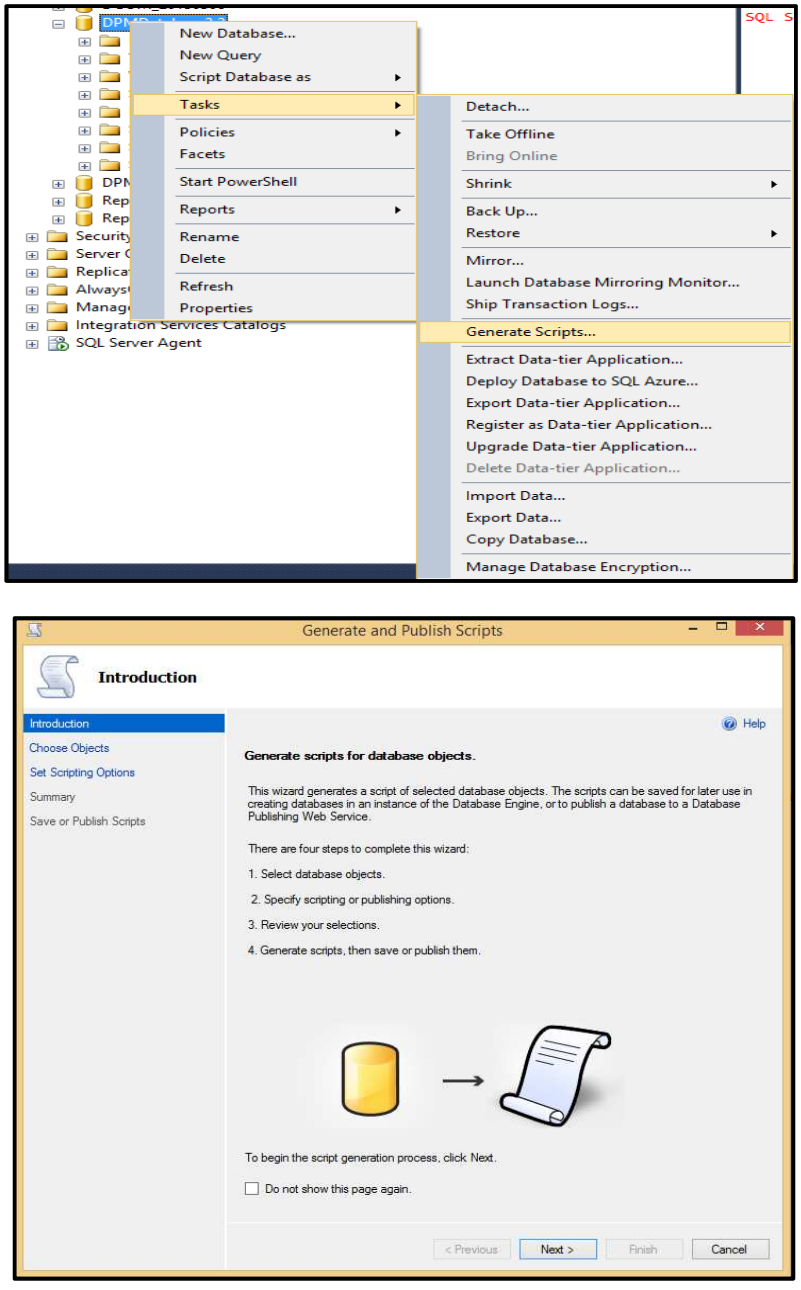


Figura 21.1. Componente para la generación de Scripts de BDD en SQL Server 2012.

Al ejecutar *Next* aparece la Opción de generar el script completo o solo parte. En nuestro caso generamos un script con todas las entidades del Modelo.

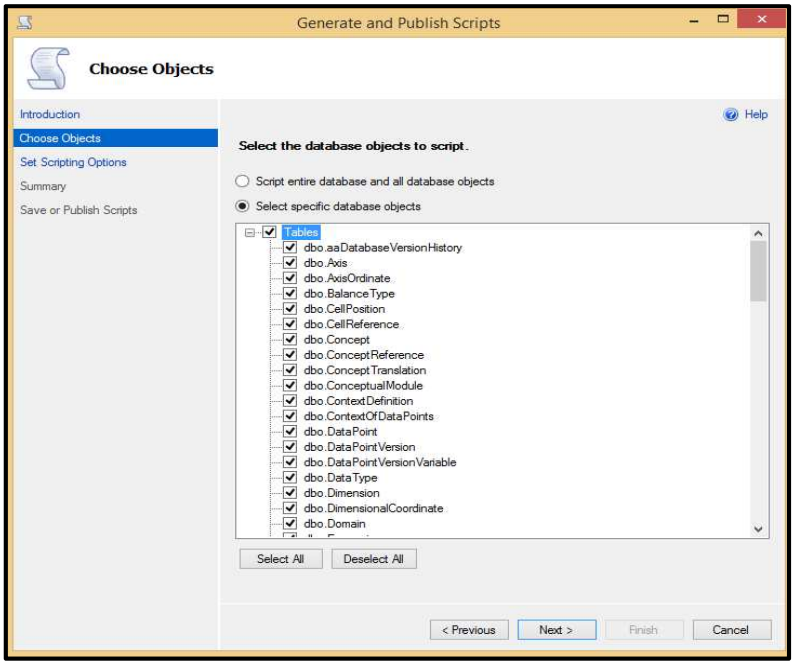


Figura 21.2. Componente para la generación de Scripts de BDD en SQL Server 2012.

Al ejecutar *Next* se especifica si se quiere publicar como un servicio Web o salvarlo en una localización especifica. Elegimos la primera.

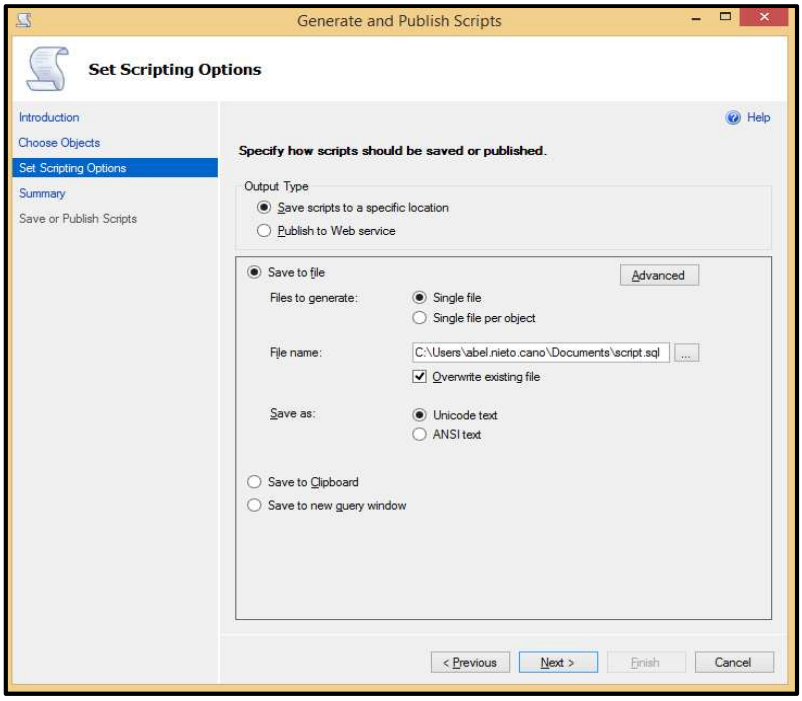


Figura 21.3. Componente para la generación de Scripts de BDD en SQL Server 2012.

Al ejecutar *next* se empieza a generar el Script en un Archivo de tipo *.SQL. Al finalizar existe la opción de salvar la generación en un *report*.

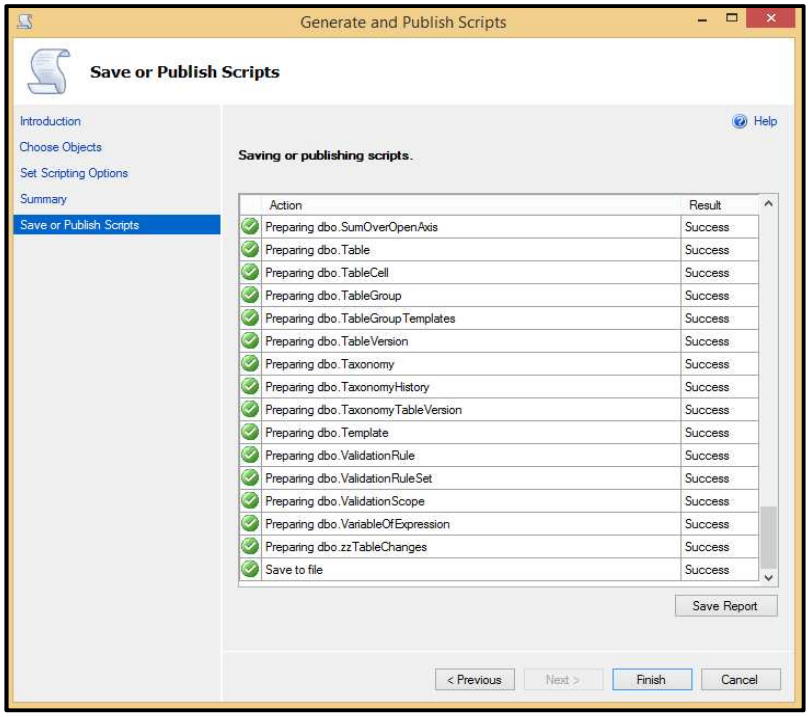


Figura 21.4. Componente para la generación de Scripts de BDD en SQL Server 2012.

En este punto ya poseemos, el conjunto de entidades y atributos definidos por el EBA, en un *Script* de BDD para *SQL Server*, El cual estará vacío (sin datos), para Posteriormente realizar la carga de datos económico-financieros y validaciones propias del Modelo así como el control de errores.

Antes de Ejecutar el *Script* procedemos a crear una nueva BD desde el *SQL Server Management Studio*.

```
USE [master]
GO


/****** Object:  Database [DPMDatabase2.2_Validation]    Script Date: 04/09/2015
11:21:11 ******/
CREATE DATABASE [DPMDatabase2.2_Validation]
 CONTAINMENT = NONE
```

```
 ON  PRIMARY
( NAME = N'DPMDatabase2.2_Validation', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL11.MSSQLSERVER\MSSQL\DATA\DPMDatabase2.2_Validation.mdf' , SIZE = 9216KB , MAXSIZE = UNLIMITED, FILEGROWTH = 1024KB )
 LOG ON
( NAME = N'DPMDatabase2.2_Validation_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL11.MSSQLSERVER\MSSQL\DATA\DPMDatabase2.2_Validation_log.ldf' , SIZE = 63424KB , MAXSIZE = 2048GB , FILEGROWTH = 10%)
GO
```

Dicha BD se llamará DPMDatabase2.2_Validation, será el repositorio de los datos ya validados del Modelo de la EBA. Una vez creado ejecutamos el script de la BDD generado, en los pasos anteriores se encuentra en el Apartado Anexos. Anexo

En el Script hay que cambiar la cláusula *Use* por la Nueva BD de Validación de datos.

```
USE [DPMDatabase2.2]  → USE [DPMDatabase2.2_Validation]
GO                      GO
```

4.4 Análisis y definición del Modelo de datos.

En este puntos las acciones solo se realizarán sobre la BD **DPMDatabase2.2_Validation.** El alcance no es todo el modelo prefijado por la EBA, solo atiende a parte de este modelo y a una serie de buenas prácticas a la hora de realizar el modelado de datos.

Esta BD tiene las siguientes características:

- **No tiene Datos.**
- **Solo posee las estructuras de tablas y vistas definidas por el EBA.**
- **No tiene Relaciones entre tablas (FK no definidas).**
- **No posee Claves Primarias definidas.**
- **No posee índices.**

Los próximos Pasos van orientados a crear un Modelo Relacional en SQL Server atendiendo a las relaciones entre tablas definidas en el Modelo relacional en Access. Cuyo fin es:

- **Crear un Modelo Relacional Estable.**
- **Crear diagramas E/R por cada área que serán objetos del Estudio.**

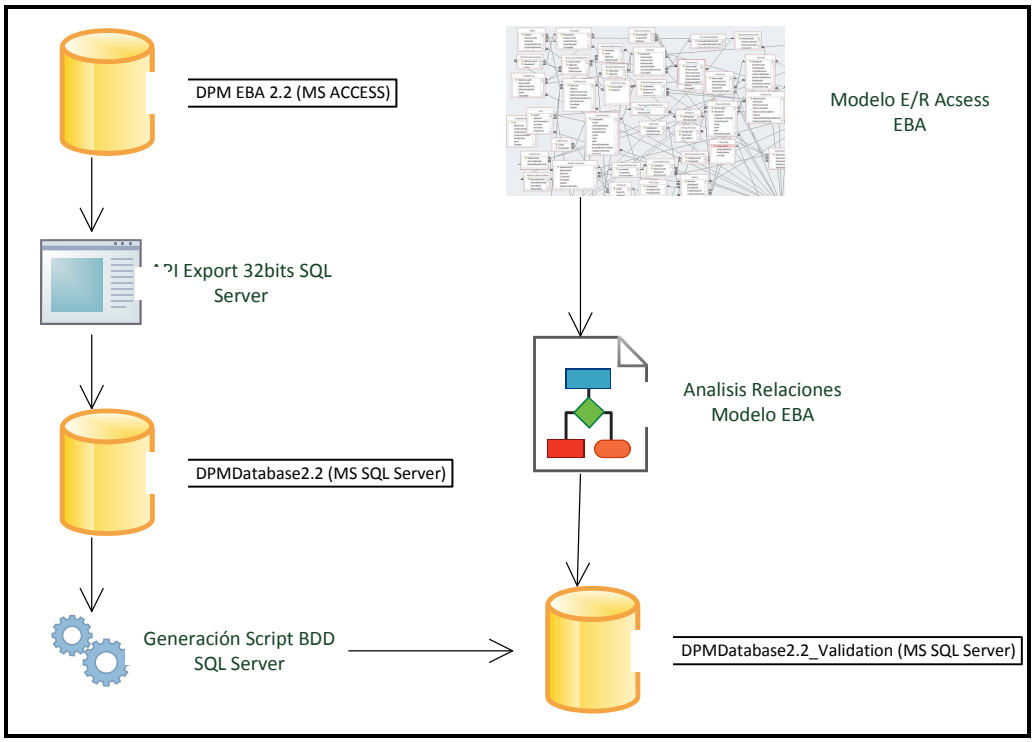


Figura 22. Esquema de creación del Modelo de BDD en SQL Server 2012.

Una de las principales ventajas de este componente técnico es imponer una serie de restricciones lógicas en el modelo, y permitiendo la realización de una serie de comprobaciones de coherencia automáticos que no serían posibles de otro modo, contribuyendo así decisivamente a acortar el tiempo necesario para alcanzar el nivel de calidad deseado, de una DPM que categoriza más de 30.000 puntos de datos.

Otra ventaja considerable de la base de datos es la posibilidad de definir muchos puntos de vista diferentes sobre el mismo contenido de metadatos, de acuerdo con las necesidades

del usuario que está tratando de comprender la estructura de información, y el vínculo entre las Plantillas de negocio y los puntos de datos dimensionales, que ahora se definen explícitamente en la DPM.

El modelo de base de datos Access propuesto por EBA, es un meta-modelo, con el fin de ser utilizados en cualquier dominio de informes que no sea COREP / FINREP, con un nivel relativamente bajo de abstracción, centrándose directamente en los principales conceptos que se utilizan en el modelado punto de datos (por ejemplo, marco, tabla, celda de tabla, dimensión, miembro, dominio...). En cuanto a los conceptos dimensionales, que, básicamente, tienen las mismas definiciones que se encuentran en los sistemas de análisis, lo que hace posible una conexión muy directa entre ambos extremos de la cadena de información.

El análisis de este proyecto se centra en la taxonomía FINREP.

El meta-modelo no está vinculado a ninguna tecnología en particular, y por lo tanto a limitaciones específicas de XBRL, no se reflejan en el DPM ya que reduciría la claridad del modelo. Con el fin de agilizar el proceso de traducción automática del DPM al de taxonomías XBRL, sin embargo, se han añadido algunos elementos adicionales del modelo, y varios campos que contienen propiedades específicas de XBRL (por ejemplo, códigos) están incluidos.

En comparación con la primera versión de la meta-modelo expresado en la base de datos publicada en noviembre de 2013, esta versión del modelo contiene algunos pequeños cambios, sobre todo alrededor de la representación de las tablas y Plantillas y los vínculos con los informes / módulos y taxonomías, y la representación de cambios entre versiones de taxonomías. Los cambios del DPM actual esta adjuntos en el apartado Anexos. (*Anexo Control de versiones DPM EBA*).

La Base de datos se estructura básicamente en torno a la representación de los metadatos y de los conceptos dimensionales utilizados para categorizar los datos y los vínculos entre ellos, cuyo origen es el DPM de la EBA tal y como representa la figura 20

Las Tablas pertenecen a un marco de información (en nuestra prueba de concepto FINREP); la mayor parte del tiempo el concepto de Tabla será el mismo que el del modelo

del asunto, salvo cuando, por razones de modelado, una Plantilla ha tenido que ser normalizada.

(En los siguientes diagramas, las flechas representan las relaciones que deben ser leídos como "pertenece a un objeto" - es decir, lo que indica una relación n a 1)

4.4.1 Tablas y agrupaciones de Tabla

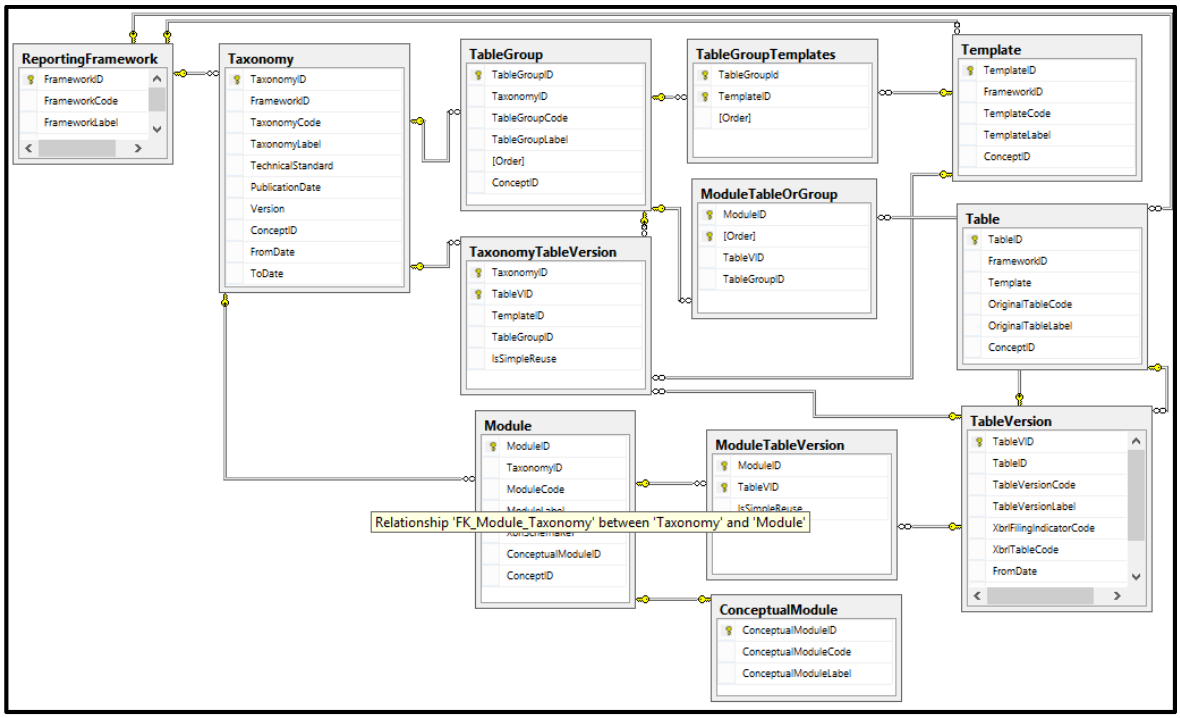


Figura 23. Diagrama en SQL Server 2012. Dbo.Diagram_DPM2.2_Taxonomy. Representa la asociación de tablas de una taxonomía.

El marco de información referido tanto a claves Primarias como a relaciones entre tablas, son conceptos relativamente estables, que pueden persistir durante varios cambios de requisitos de información práctica específicos o implementaciones técnicas.

La construcción del modelo tiene como punto de partida el análisis del Modelo de datos de la EBA.

Por otro lado una descripción específica de la clasificación de estas tablas y los puntos de datos dentro de ellos se trata de un objeto de estudio distinto y relativizado por la Taxonomía y el área de Negocio.

De cualquier modo voy a dar una breve descripción de la clasificación de estas tablas y sus Agrupaciones.

En un punto / período determinado en el tiempo se hace referencia en el modelo como una taxonomía, la descripción específica de una tabla en particular dentro de una taxonomía es representado por un *TableVersion*, varios de los cuales puede representar la evolución de una Tabla conceptual particular en el tiempo.

Las Taxonomías pueden ser agrupadas, por información en *TableGroups*, en representación de las materias (por ejemplo, la adecuación del capital, etc. Riesgo de Crédito)

La tabla *Module* representan las principales unidades de información, que potencialmente podrían ser reportados en una sola presentación. Las tablas incluidas en un módulo en particular se indican mediante las relaciones *ModuleTableOrGroup*, que se vincula a un *TableVersion* específico, o un *TableGroup* (y contienen *TableGroups* o *TableVersions*) al módulo.

La Tabla *ModuleTableVersion* aclara las versiones individuales de la Tabla que se incluyen en un módulo (n ID *tableversion* de un grupo de la tabla se incluyen en 1 módulo en particular).

4.4.2 La presentación en lista

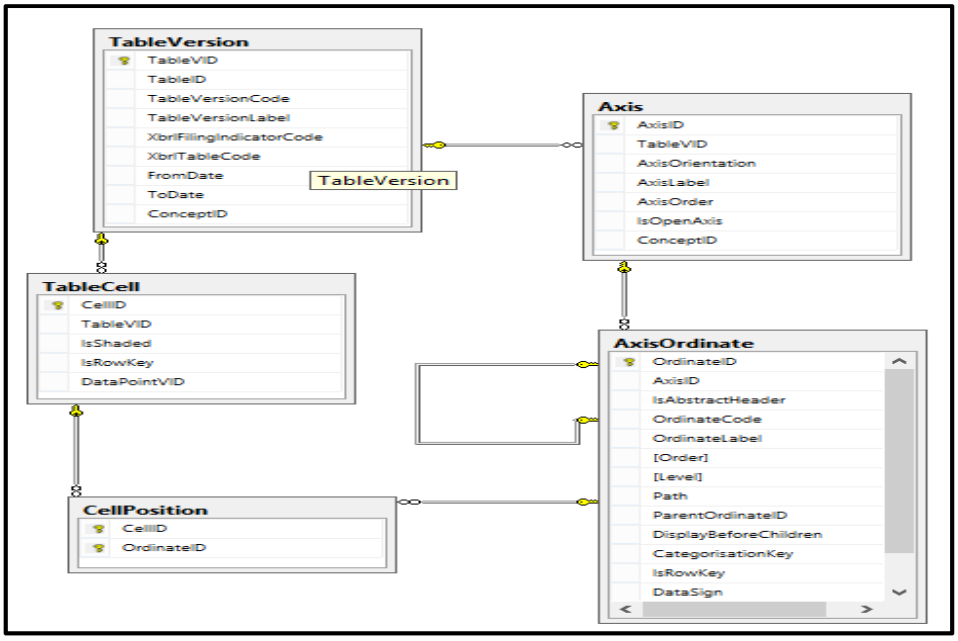


Figura 24. Diagrama en SQL Server 2012. Dbo.Diagram_DPM2.2_AxisOrdinate. Representa la asociación de tablas de tipo lista del DPM.

La disposición física de la tabla (*TableVersion*) se describe en términos de ejes. Un eje representa ya sea las filas, columnas u hojas de una Tabla, que son "X", "Y" y "Z" ejes respectivamente (los valores posibles de *AxisOrientation*). Cada valor posible a lo largo de cada eje (es decir, el individuo fila, columna o lámina) se llama un *AxisOrdinate*.

Esta descomposición de las tablas es clave para el proceso de modelado, que clasifica a cada valor individual sobre un eje (alrededor de 6.000 valores), en lugar de cada célula individual (hay más de 80.000 celdas de la tabla).

En el marco de varios tipos de tablas. La mayoría tienen una estructura fija, con una sola hoja, mientras que otros pueden tener múltiples hojas con la misma estructura o incluso un número variable de hojas. Además, algunas Tablas tienen un formato de "lista", es decir, una estructura abierta donde las filas son identificadas por los datos clave insertados, y repitiendo un número indeterminado de veces, dependiendo de los datos que se informaron.

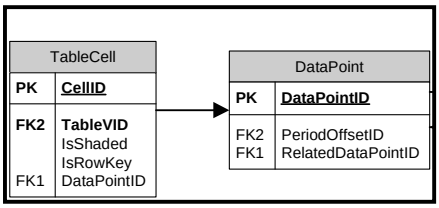


Figura 25. Relación 1..n tableCell → Datapoint

Cada celda de la tabla corresponde a uno, y sólo uno, hecho informativo, llamado *DataPoint* (Un hecho en el MDM); Sin embargo, hay algunos puntos de datos representados en múltiples celdas de la tabla. En este último caso las celdas de la tabla contienen exactamente la misma información, y así comparten exactamente la misma clasificación en el DPM.

4.4.3 Dimensionalidad del modelo de datos

Los datos asociados a las tablas del dimensionamiento del DPM es el conjunto de datos elegido para el estudio de las validaciones de integración de los datos que debe seguir el Modelo completo del EBA.

Los conceptos dimensionales representados son Dominios, Dimensiones, miembros y jerarquías.

Las dimensiones son las diferentes categorías que se usan para describir los puntos de datos (por ejemplo, del sector de contraparte), y de los miembros son los casos reales de esas categorías (por ejemplo, bancos centrales).

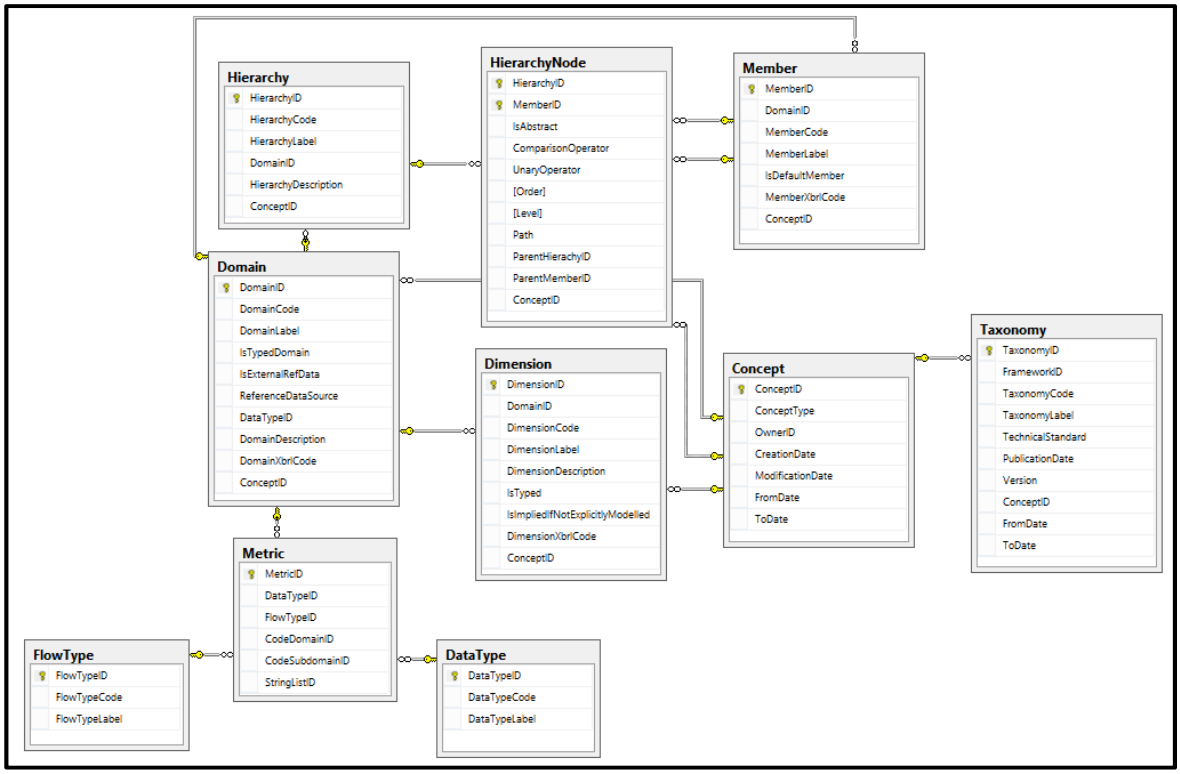


Figura 26. Diagrama en SQL Server 2012. Dbo.Diagram_DPM2.2_Dimensional. Representa la asociación de tablas del modelo dimensional del DPM.

Por ejemplo, la celda en la tabla FINREP 8.01.a, la fila 090, la columna 010, esta categorizada en el DPM por los 5 pares de relaciones:

[Base].[Liabilities]

[Metric].[Carrying amount]

[Main category].[Deposits. Redeemable at notice]

[Accounting portfolio].[Financial liabilities held for trading]

[Counterparty sector].[Central banks]

Todos los miembros de una dimensión deben pertenecer al mismo dominio. Un dominio de grupos de miembros del mismo tipo, que corresponden a conceptos con carácter semántico

similares, ya sean abstractas como Tipo de riesgo, o más concreto como moneda. Algunos (la mayoría) de los dominios son "cerrados", es decir, tener un número predefinido y restringido de miembros (por ejemplo, países), y otras son "abiertas", ya que no podemos enumerar todas las instancias posibles (por ejemplo, personas jurídicas).

Las dimensiones no siempre son equivalentes a dominios, ya que pueden representar un papel particular en el modelo. Por ejemplo Residencia, la ubicación de las actividades y País del mercado, son todas ellas diferentes dimensiones.

Las Jerarquías indican cómo los miembros o conceptos de un dominio que se relacionan entre sí y definen subconjuntos de los miembros, y también pueden definir las agregaciones de menor a mayor los niveles superiores de la jerarquía.

4.4.4 Análisis dimensional de las Plantillas de datos.

Al describir las Plantillas de datos, expertos en negocios definen el conjunto de pares de las dimensiones y los miembros que categorizan cada fila y columna. Si hay un "eje z ', se generan varias hojas, y la dimensión y los miembros asociados a cada hoja en su conjunto también se deben especificar.

Es posible rastrear desde una celda de la tabla, a través de las coordenadas de los ejes de la celda para identificar la clasificación completa de cada celda de la tabla individual, que resulta de la unión de la categorización de sus coordenadas de ejes (hoja, fila y columna).

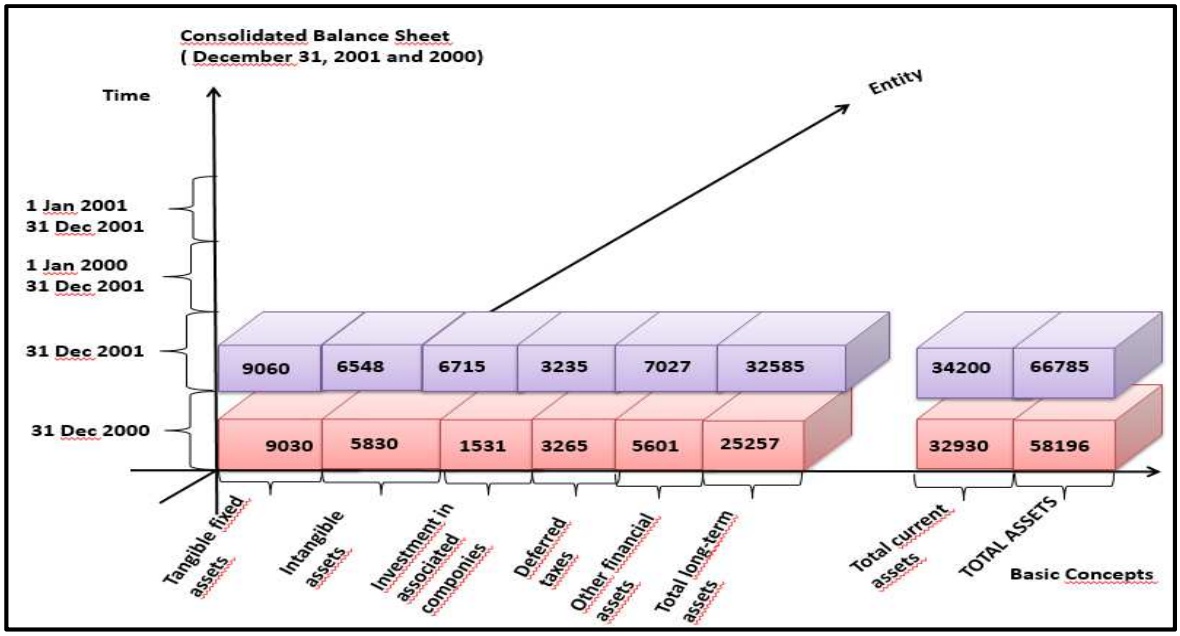


Figura 27. Ejemplo de representación de conceptos contables en el eje X en el tiempo eje Y y su Magnitud eje Z. Referencia [9]

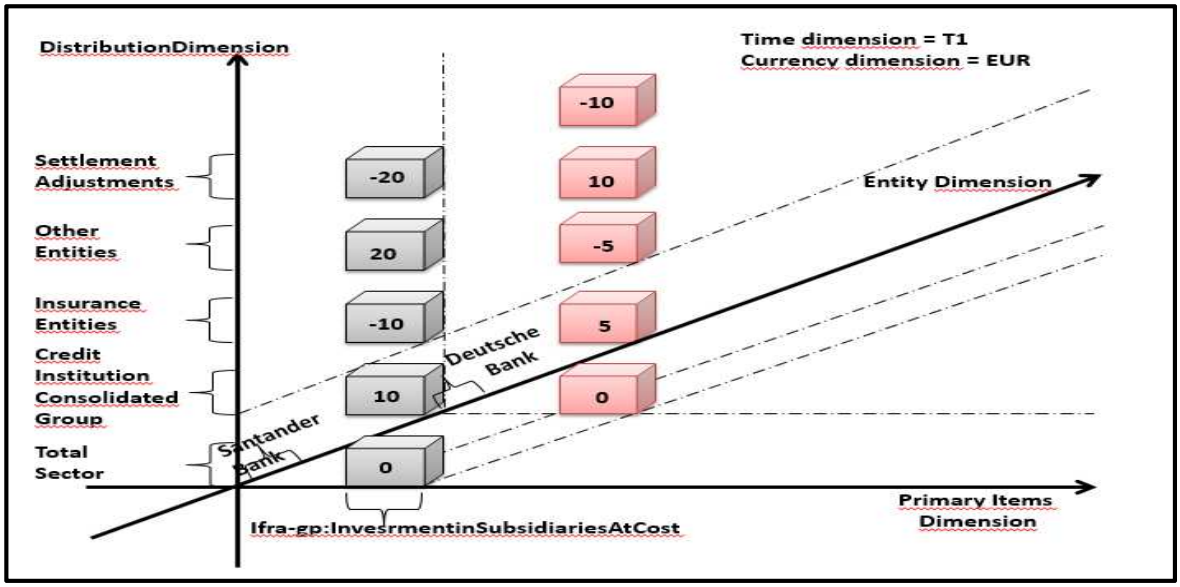


Figura 28. Ejemplo de representación de dimensiones Padre en el eje X con las dimensiones hijo en el eje Y y su Magnitud eje Z de tipo monetaria. Referencia [9]

El modelo contiene dos tipos principales de tablas, "cerrado" - donde cada eje de las Tablas tiene todos sus valores requeridos y figuran explícitamente, y por lo tanto el tamaño exacto de la tabla reportado es conocido, y "abierto", donde uno o más ejes es "abierto" (permite un número variable de entradas, elegidos ya sea de una lista restringida por ejemplo, los sectores de contrapartida (), o de un tipo en particular (por ejemplo, cualquier número entero).

Tablas cerradas se dividen en dos tipos principales, los que tienen ejes X e Y sólo (que son simplemente tablas naturales), o los que también especifican un eje Z, y se componen de varias hojas, cada una con una copia completa de la Tabla, uno para cada ordenada en el eje Z.

Para las tablas de ejes abiertos, el número no determinado de entradas en el eje está representado en la base de datos de DPM por un *AxisOrdinate* con *OrdinateCode* de "999".

Los valores aceptables para el valor "clave" identifican una fila / columna / hoja específica (según el caso) en una Tabla abierta, que por supuesto debe ser único y dependiendo del tipo de datos puede ser la dimensión clave, o bien una cadena de caracteres o un entero (por ejemplo), o pueden ser un valor de un "dominio" predefinido en una lista de valores. Cuando se requiere un valor de un dominio, las posibilidades válidas de ese dominio pueden ser aún más limitada al ser un valor contenido en una "jerarquía" específica es decir, un árbol de valores. Esto se indica a través de la Tabla *OpenAxisValueRestriction*, que también se indicará si el miembro de la raíz de la jerarquía es un valor aceptable o no.

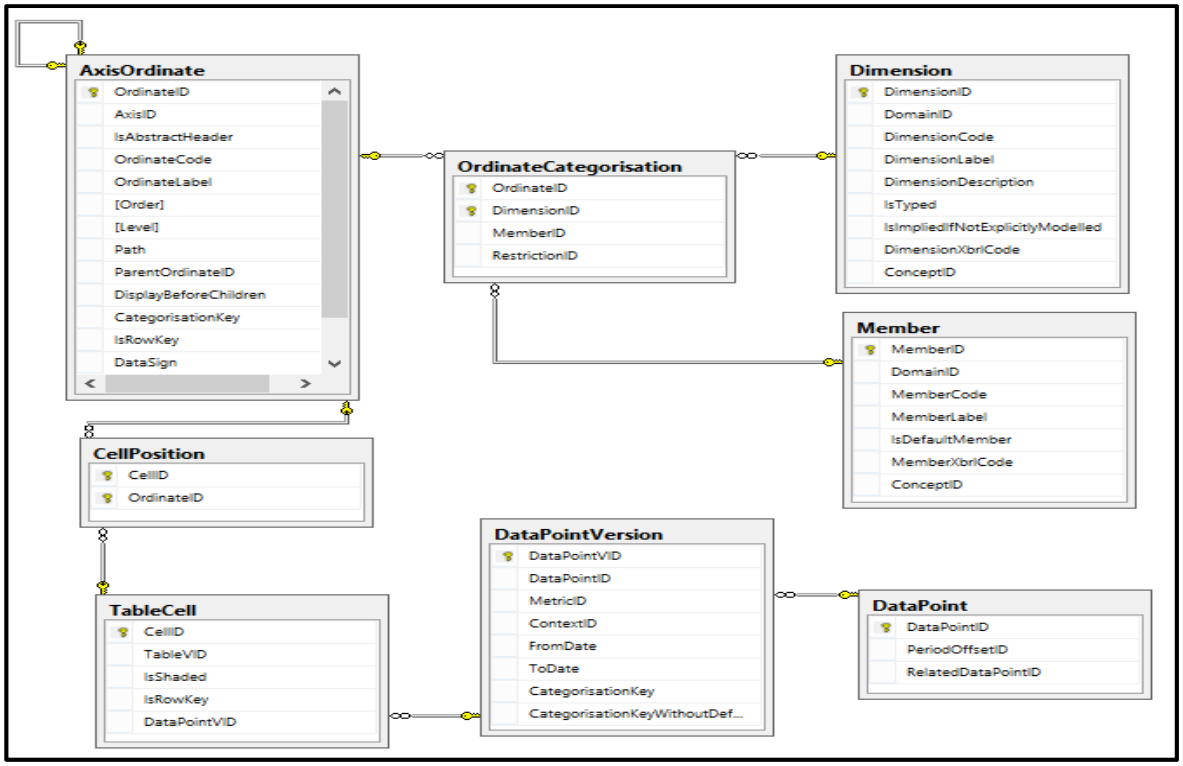


Figura 29. Diagram_DPM2.2_Templates. Representación de dimensiones, ejes y punto de dato

El Modelado del punto de datos es un proceso iterativo. En cada iteración se aplica un conjunto de comprobaciones de coherencia a todas las celdas del marco, para validar el modelo desde el punto de vista lógico, la comprobación de casos de desaparecidos, dimensiones obligatorias, o duplicar las dimensiones, por ejemplo.

Finalmente los puntos de datos se alcanzan mediante la identificación de las combinaciones únicas de pares [dimensión]. [Miembro] en todo el conjunto completo de celdas categorizadas.

4.4.5 Estructuras de árbol en el Modelo

Dos tablas de la base de datos de DPM representan “estructuras árbol" / "padre-hijo" (es decir, cuando una entrada puede tener varias entradas secundarias, cada una de las cuales

también pueden tener hijos, etc.). Los dos casos se dan en las tablas *HierarchyNode* y *AxisOrdinate*.

En las reglas de validación, objeto de nuestro estudio. Solo validaremos los datos referentes a la tabla *HierarchyNode*

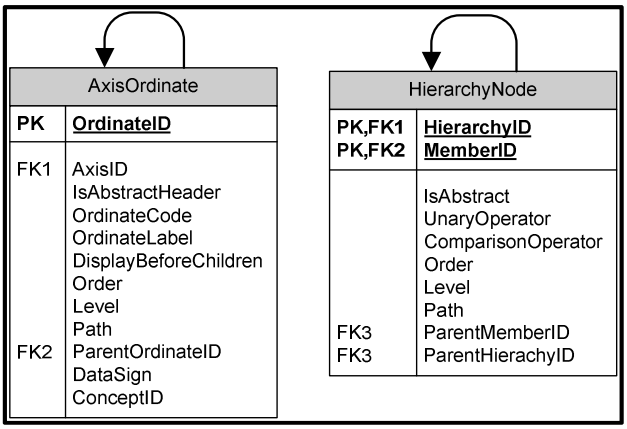


Figura 30. Ejemplo de representación de las tablas AxisOrdinate y HierarchyNode

En cada caso, la estructura de árbol está representado en la base de datos de dos maneras, por un enlace a una entrada de "Padre" de cada hijo y con un campo "*Path*" para dar la ruta completa de un hijo a través de sus antepasados al primer nivel, los cuales utilizan el campo Orden para indicar el orden global de los nodos en el árbol (y así también el orden dentro de cada grupo de hermanos).

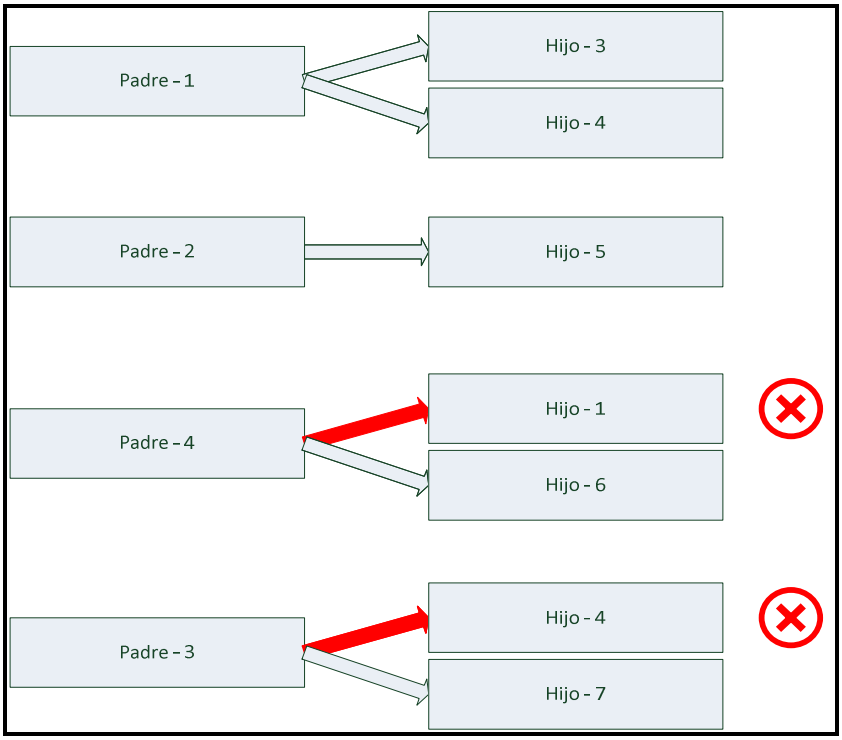


Figura 31. Representación de enumeraciones de caminos válidos y no Validos en una estructura Padre-Hijo.

Las enumeraciones de caminos Validos referentes a los dos primeros ejemplos sería Representado como:

Nodo	Padre	Ruta	Orden	Nivel
1		1.	1	1
3	1	1.3.	2	2
4	1	1.4.	3	2
2		2.	4	1
5	2	2.5.	5	2

Tabla 1. Ejemplo estructura en árbol Valida.

Las enumeraciones de caminos no validos correspondientes a los 2 últimos casos de la figura 31, esta será una de las validaciones que se deben llevar a cabo en la inserción de los datos que se efectuaran más adelante.

4.4.6 Tablas y descripción de campos.

Estructura y Descripción de las tablas más Importantes objeto del estudio de validación de datos.

Las descripciones han sido sacadas de los análisis de la EBA sobre el área de negocio.

Axis

Representa si el dato es una fila, columna o una hoja de una tabla en particular.

Nombre Campo	Tipo	Long.	Descripción
AxisID	Integer	4	ID
TableVID	Integer	4	Tabla a la que pertenece el Eje
AxisOrientation	Nvarchar	1	X,Y o Z fila, columna o plantilla respectivamente
AxisLabel	Nvarchar(MAX)	255	Descripción en Inglés
AxisOrder	Smallint	2	Para Múltiples ejes Z indica el orden que deberían mostrar.
IsOpenAxis	Bit	1	Indica si está “Abierto” (1) vs. “Cerrado” (0) **e.g. Figura 27**

Tabla 2. Estructura Tabla Axis. Referencia [1]

AxisOrdinate

Representa una posición específica en un eje "cerrado" (o un marcador de posición genérico para un eje "abierto". En estructuras de árbol representan la anidación de filas o columnas.

Nombre Campo	Tipo	Long.	Descripción
OrdinateID	Integer	4	ID

AxisID	Integer	4	Eje de ordenadas al que pertenece
IsAbstractHeader	Bit	1	Si es verdad, esto ordenada no representa un "reportable data", ej. Se puede representar como un título no como una columna de valores.
MetricID	Integer	4	Naturaleza del Dato.
OrdinateCode	Nvarchar	4	Código Corto
OrdinateLabel	Nvarchar(MAX)	255	Descripción en Inglés
Order	Integer	4	indica el orden en estructuras Árbol
Level	Integer	4	Nivel de parentesco en estructuras Árbol
Path	Nvarchar	255	Ruta desde el nodo Raíz en estructuras Árbol
ParentOrdinateID	Integer	4	Nodo Padre (Si aplica)
DisplayBeforeChildren	Bit	1	
DataSign	Nvarchar	255	
ConceptID	Integer	4	Referencia a la tabla concept

Tabla 3. Estructura Tabla Axisordinate. Referencia [1]

ContextDefinition

Cuando Un par de miembros de una dimensión específica se utiliza para categorizar una o más versiones de puntos de datos.

Nombre Campo	Tipo	Long.	Descripción
ContextID	Integer	4	ID
DimensionID	Integer	4	Dimension Considerada
MemberID	Integer	4	Categorización en esa dimension.

Tabla 4. Estructura Tabla ContextDefinition. Referencia [1]

ContextOfDataPoints

Una combinación específica de los pares de miembros de dimensión (excluyendo la dimensión métrica) utilizada para categorizar una o más versiones de puntos de datos. La intención de ilustrar el enfoque destinado a la cartografía de XBRL

Nombre Campo	Tipo	Long.	Descripción
ContextID	Integer	4	ID
ContextKey	Nvarchar	255	Concatenación de los [DimensionCode MemberID]
XbrlContextKey	Nvarchar(MAX)	255	Concatenación de los códigos usados en XBRL.

Tabla 5. Estructura Tabla ContextOfDataPoints. Referencia [1]

DataPoint

Un dato único de la información. Un hecho en el MDM.

Nombre Campo	Tipo	Long.	Descripción
DataPointID	Integer	4	ID
PeriodOffsetID	Integer	4	
RelatedDataPointID	Integer	4	

Tabla 6. Estructura Tabla DataPoint.

DataPointVersion

La clasificación de un DataPoint que es válido durante un período de tiempo específico.

Nombre Campo	Tipo	Long.	Descripción
DataPointVID	Integer	4	ID
DataPointID	Integer	4	Referencia a la tabla Data Point
MetricID	Integer	4	Referencia a la tabla Metric
ContextID	Integer	4	Categorización dimensional

FromDate	Date	8	Fecha desde que el Data Point es valido
ToDate	Date	8	Fecha hasta que el Data Point es valido
CategorisationKey	Nvarchar	255	Concatenación de [DimensionCode MemberID]
CategorisationKeyWithoutDefaults	Nvarchar	255	Concatenación de [DimensionCode MemberID]

Tabla 7. Estructura Tabla DataPointVersion. Referencia [1]

DataType

Tipo de datos asociado a una métrica.

Nombre Campo	Tipo	Long.	Descripción
DataTypeID	Integer	4	ID
DataTypeCode	Nvarchar	1	Código corto
DataTypeLabel	Nvarchar	50	Descripción en Inglés

Tabla 8. Estructura Tabla DataType. Referencia [1]

Dimension

Categoría / aspecto utilizado para describir y diferenciar los puntos de datos, cada uno se refiere a una característica específica. Los valores permitidos se toman de un dominio. Si éstos se enumeran explícitamente son llamados miembros.

Nombre Campo	Tipo	Long.	Descripción
DimensionID	Integer	4	ID
DomainID	Integer	4	Dominio del que se toman los valores permitidos para esta dimensión
DimensionCode	Nvarchar	3	Short code
DimensionLabel	Nvarchar	255	Descripción etiqueta en Ingles

DimensionDescription	Nvarchar(MAX)	255	Descripción larga en Inglés
IsTyped	Bit	1	Dimensiones "tipadas" permiten cualquier valor de una forma particular (es decir, cualquier cadena de cierta longitud o patrón, cualquier número, fecha, etc.), las dimensiones "explícitas" sólo permiten una opción de una lista dada de miembros.
DimensionXbrlCode	Nvarchar	255	Código usado en documentos XBRL.
ConceptID	Integer	4	Referencia a la tabla conceptos

Tabla 9. Estructura Tabla Dimension. Referencia [1]

DimensionalCoordinate

Especificación de los pares Dimensión y miembro.

Nombre Campo	Tipo	Long.
DimensionID	Integer	4
MemberID	Integer	4

Tabla 10. Estructura Tabla DimensionCoordinate.

Domain

Conforma grupos de valores de un tipo particular o un concepto particular. Puede tener una lista explícita de los valores permitidos (miembros), o bien especificar los valores de un determinado tipo o patrón (un dominio). Proporciona los valores permitidos para una dimensión.

Nombre Campo	Tipo	Long.	Descripción
DomainID	Integer	4	ID
DomainCode	Nvarchar	3	Código Corto

DomainLabel	Nvarchar	255	Descripción etiqueta en Inglés
IsTypedDomain	Bit	1	Dominios "Tipados" permiten cualquier valor de una forma particular (es decir, cualquier cadena de cierta longitud o patrón, cualquier número, fecha, etc.), las dimensiones "explícitas" sólo permiten una opción de una lista dada de los miembros.
IsExternalRefData	Bit	1	Indica si se obtiene la lista de valores de dominio de una lista autorizada externamente
ReferenceDataSource	Nvarchar	255	Indica dónde se obtiene la lista de valores de (si no se define por el propietario del dominio).
DataTypeID	Integer	4	Indica los tipos de los valores permitidos
DomainDescription	Nvarchar(MAX)	255	Descripción larga en Inglés
DomainXbrlCode	Nvarchar	255	Código usado en documentos XBRL.
ConceptID	Integer	4	Referencia a la tabla conceptos

Tabla 11. Estructura Tabla Domain. Referencia [1]

FlowType

Los tipos de flujo pueden ser tipo “Stock” o “Flow”.

Nombre Campo	Tipo	Long.
FlowTypeID	Integer	4
FlowTypeCode	Nvarchar	1
FlowTypeLabel	Nvarchar	50

Tabla 12. Estructura Tabla FlowType.

Hierarchy

Las jerarquías especifican cómo los miembros se relacionan entre sí, y también pueden definir las agregaciones de menores niveles a los niveles superiores de la jerarquía.

Nombre Campo	Tipo	Long.	Descripción
HierarchyID	Integer	4	ID
HierarchyLabel	Nvarchar	255	Descripcion etiqueta en Inglés
HierarchyCode	Nvarchar	50	Código Corto
DomainID	Integer	4	Dominio al que se refiere la Jerarquía
HierarchyDescription	Nvarchar(MAX)	0	Descripción larga en Inglés
ConceptID	Integer	4	Referencia a la tabla conceptos

Tabla 13. Estructura Tabla Hierarchy. Referencia [1]

HierarchyNode

Representa un nodo en una jerarquía de miembros, especificando cómo los miembros se relacionan entre sí, y también pueden definir las agregaciones de menores niveles a los niveles superiores de la jerarquía. Es donde está integrada la estructura en Árbol.

Nombre Campo	Tipo	Long.	Descripción
HierarchyID	Integer	4	Jerarquía a la que este nodo Pertenece
MemberID	Integer	4	El Miembro que este nodo representa
IsAbstract	Bit	1	
ComparisonOperator	Nvarchar	2	Indica la relación entre este nodo y la agregación de sus hijos
UnaryOperator	Nvarchar	1	Indica el operados de este nodo para la agregación de sus hermanos
Order	Integer	4	indica el orden en estructuras Árbol
Level	Integer	4	Nivel de parentesco en estructuras Árbol
Path	Nvarchar	255	Ruta desde el nodo Raíz en estructuras Árbol

ParentHierachyID	Integer	4	Siempre debe ser el mismo que HierarchyID (incluido puramente debido a las limitaciones de MS Access).
ParentMemberID	Integer	4	Indica el padre de este nodo, en su caso o el nivel inmediatamente superior (información de la estructura de árbol)

Tabla 14. Estructura Tabla HierarchyNode. Referencia [1]

Member

Un valor posible dentro de un dominio.

Nombre Campo	Tipo	Long.	Descripción
MemberID	Integer	4	ID
DomainID	Integer	4	Dominio al que es miembro pertenece
MemberCode	Nvarchar	50	Código corto
MemberLabel	Nvarchar	255	Descripcion etiqueta en Inglés
MemberXbrlCode	Nvarchar	255	Código usado en documentos XBRL.
IsDefaultMember	Bit	1	
ConceptID	Integer	4	Referencia a la tabla conceptos

Tabla 15. Estructura Tabla Member. Referencia [1]

Metric

El significado conceptual de un dato de información.

Nombre Campo	Tipo	Long.	Descripción
MetricID	Integer	4	ID - Coincide con un MemberID desde la que se pueden obtener las descripciones
DataTypeID	Integer	4	Tipo de dato presentado

FlowTypeID	Integer	4	Es un valor en un punto específico en el tiempo (stock) o un cambio en el valor (flow)
SubdomainID	Integer	4	

Tabla 16. Estructura Tabla Metric. Referencia [1]

Module

Unidad de presentación de los informes, representa el posible contenido de una instancia XBRL particular. Asociado con un valor *SchemaRef* específico en una instancia XBRL.

Nombre Campo	Tipo	Long.
ModuleID	Integer	4
TaxonomyID	Integer	4
ModuleCode	Nvarchar	255
ModuleLabel	Nvarchar	50
XbrlSchemaRef	Nvarchar(MAX)	
ConceptuallModuleID	Integer	4
ConceptID	Integer	4

Tabla 17. Estructura Tabla Module.

ModuleTableOrGroup

Nombre Campo	Tipo	Long.
ModuleID	Integer	4
TableVID	Integer	4
TableGroupId	Integer	4
Order	Integer	4

Tabla 18. Estructura Tabla ModuleTableOrGroup.

ModuleTableVersion

Nombre Campo	Tipo	Long.
ModuleID	Integer	4
TableVID	Integer	4

Tabla 19. Estructura Tabla ModuleTableVersion.

OpenAxisValueRestriction

Para las Tablas con "ejes abiertos" (es decir, los que permiten una selección de un número variable de hojas / filas / columnas cada uno con un valor de un dominio particular), los valores permitidos para ser informado no puede ser todos los valores de un dominio, pero sólo un subconjunto. Esta tabla indica la restricción aplicada a un eje abierto particular.

Nombre Campo	Tipo	Long.	Descripción
AxisID	Integer	4	Ejes a los que se aplica la restricción
RestrictionID	Integer	4	Enlace a los detalles de la restricción y a los posibles valores de la dimensión relevante

Tabla 20. Estructura Tabla OpenAxisRestriction. Referencia [1]

OpenMemberRestriction

Para las Tablas con "*ejes abiertos*" (es decir, los que permiten una selección de un número variable de hojas / filas / columnas cada uno con un valor de un dominio particular), los valores permitidos para ser informado no puede ser todos los valores de un dominio, pero sólo un subconjunto. Esta tabla indica el subconjunto permitido por la referencia a un miembro de una jerarquía, todos los miembros por debajo del miembro de referencia son valores aceptables, si *MemberIncluded* es cierto, el miembro de referencia es también un valor válido, de lo contrario, no lo es.

Nombre Campo	Tipo	Long.	Descripción
RestrictionID	Integer	4	ID
HierarchyID	Integer	4	Valores restringidos para una jerarquía dada

MemberID	Integer	4	Valores para un miembro dado de una jerarquía
MemberIncluded	Bit	1	En caso afirmativo, el miembro vinculado es un valor válido, si no, sólo son descendientes.
AllowDefaultMember	Bit	1	

Tabla 21. Estructura Tabla OpenMemberRestriction. Referencia [1]

ReportingFramework

Marco de información general - de alto nivel, el concepto estable

Nombre Campo	Tipo	Long.	Descripción
FrameworkID	Integer	4	ID
FrameworkCode	Nvarchar	255	Código corto
FrameworkLabel	Nvarchar	255	Descripción etiqueta en Inglés
ConceptID	Integer	4	Referencia a la tabla conceptos

Tabla 22. Estructura Tabla ReportingFramework. Referencia [1]

Table

La mayoría de las veces la tabla será la misma, como una plantilla de negocios, excepto cuando, por razones de modelado, las plantillas tuvieron que ser normalizado y se dividió en dos o más tablas

Nombre Campo	Tipo	Long.	Descripción
TableID	Integer	4	ID
FrameworkID	Integer	4	Reporting framework al que pertenece
OriginalTableCode	Nvarchar	255	Código corto
OriginalTableLabel	Nvarchar	255	Descripción etiqueta en Inglés
Template	Nvarchar	255	Nombre / Código de la plantilla de la cual esta tabla es una representación.
ConceptID	Integer	4	Referencia a la tabla conceptos

Tabla 23. Estructura Tabla Table. Referencia [1]

TableCell

Representa una intersección individual de fila, columna (y hoja) para una tabla particular.

Nombre Campo	Tipo	Long.	Descripción
CellID	Integer	4	ID
TableVID	Integer	4	
IsShaded	Bit	1	¿Hay datos que se espera que se introducirán en esta celda? - Ya sea porque no se requiere, o porque esta célula forma parte de una misma partida, o la intersección de la fila y la columna (y hoja) no tiene sentido lógico.
IsRowKey	Bit	1	¿Esta celda representa el código / ID utilizado para identificar una fila de datos?
DataPointID	Integer	4	Información empresarial contenida en esta celda

Tabla 24. Estructura Tabla TableCell. Referencia [1]

TableGroup

Agrupar (sólo con fines de información) de las plantillas dentro de una taxonomía

Nombre Campo	Tipo	Long.	Descripción
TableGroupID	Integer	4	ID
TaxonomyID	Integer	4	Taxonomía a la cual pertenece
TableGroupCode	Nvarchar	255	Código corto
TableGroupLabel	Nvarchar	255	Descripcion etiqueta en Inglés
Order	Integer	4	Posición en la tabla
ConceptID	Integer	4	Referencia a la tabla *concept*

Tabla 25. Estructura Tabla TableGroup. Referencia [1]

TableGroupTemplates

Vinculación de Tabla entre *TableGroup* y *Template*

Nombre Campo	Tipo	Long.
TableGroupID	Integer	4
TemplateID	Integer	4

Tabla 26. Estructura Tabla tableGroupTemplates. Referencia [1]

TableVersion

La descripción específica de una tabla en particular a partir de un marco de información, dentro de una taxonomía, válida durante un periodo de tiempo determinado. Varios TableVersions pueden representar la evolución de una tabla particular en el tiempo.

Nombre Campo	Tipo	Long.	Descripción
TableVID	Integer	4	ID
TableID	Integer	4	
TableVersionCode	Nvarchar	255	Código corto
TableVersionLabel	Nvarchar(MAX)		Descripción en Inglés.
XbrlFilingIndicatorCode	Nvarchar	255	Código que debe ser incluido en la presentación de una instancia XBRL.
FromDate	Date	8	Fecha desde que es válido
ToDate	Date	8	Fecha hasta que es válido
ConceptID	Integer	4	Referencia a la tabla conceptos

Tabla 27. Estructura Tabla TableVersion. Referencia [1]

Taxonomy

Una descripción específica de la clasificación de las tablas y datos de puntos de un marco de información, en un punto / período determinado en el tiempo

Nombre Campo	Tipo	Long.	Descripción
TaxonomyID	Integer	4	ID
FrameworkID	Integer	4	Informes marco que esta taxonomía describe
TaxonomyCode	Nvarchar	255	Código corto
TaxonomyLabel	Nvarchar	50	Descripción en Inglés.
ConceptID	Integer	4	Referencia a la tabla conceptos
FromDate	Date	8	Fecha desde que es válida
ToDate	Date	8	Fecha hasta que es válida

Tabla 28. Estructura Tabla Taxonomy. Referencia [1]

Template

Plantillas usadas para los informes.

Nombre Campo	Tipo	Long.
TemplateID	Integer	4
FrameworkID	Integer	4
TemplateCode	Nvarchar	255
TemplateLabel	Nvarchar	255
Template	Nvarchar	255
ConceptID	Integer	4

Tabla 29. Estructura Tabla Template. Referencia [1]

Finalmente adjunto el modelo de datos relacional completo definido. Se trata de solo una parte del Modelo de datos desarrollado por la EBA.

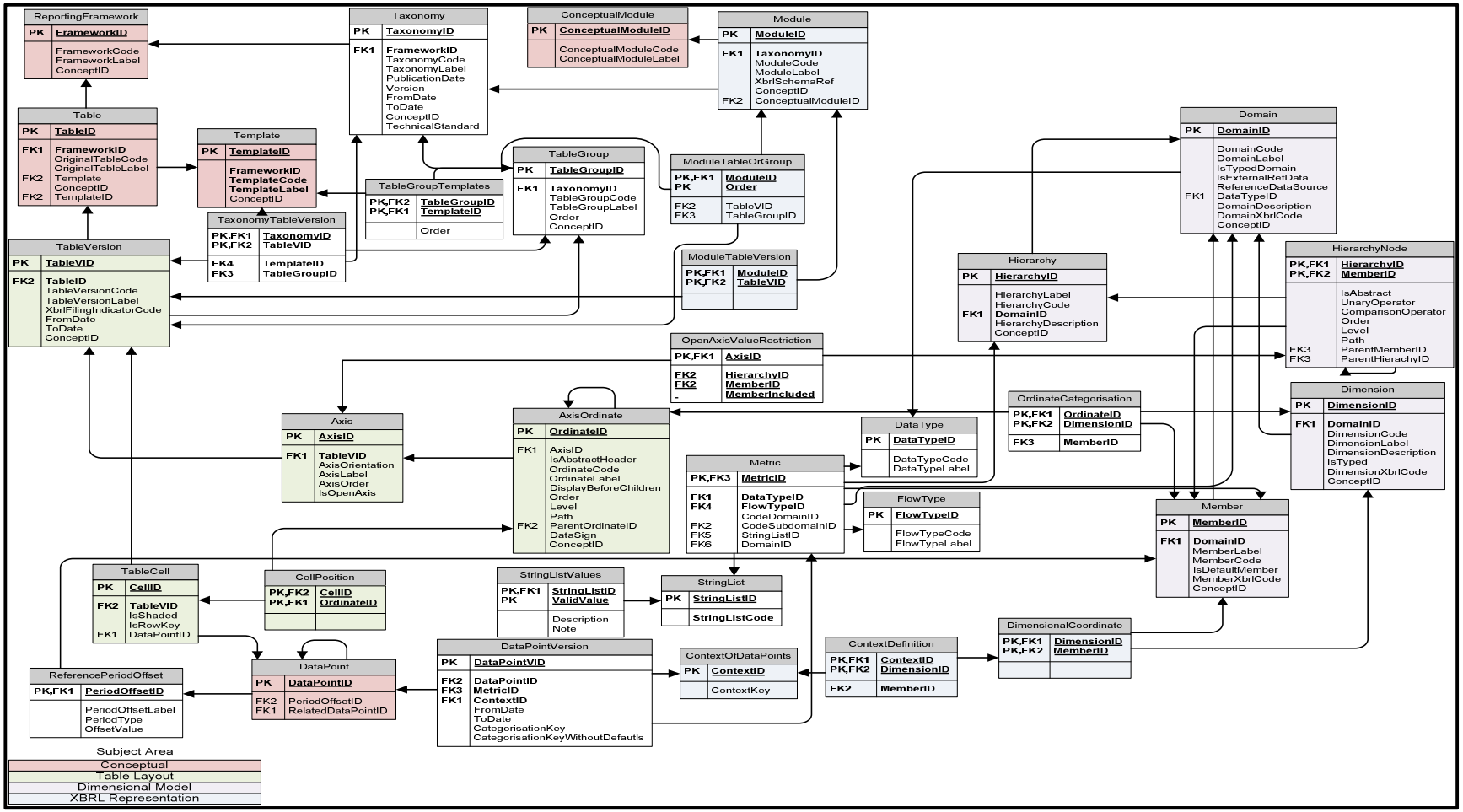


Figura 32. Modelo de datos Completo. Referencia [1]

4.4.7 Preocupaciones dentro del Modelo hacia XBRL

Una limitación de XBRL es el tamaño de datos a tratar por cada informe. Si el informe es grande hay dificultades en el procesamiento de la instancia de documento. El problema está con grandes instituciones financieras que deben de entregar al Regulador informes XBRL, muy grandes y no son fácilmente tratables por las máquinas, al necesitar gran cantidad de memoria.

La elección de tecnologías ágiles es importante, esta es una de las razones por la que el SGBD *SQL Server* es una buena Opción, especialmente en la validación de los cálculos de los informes. Cuando los informes XBRL son muy grandes, que son complejos o incluso imposible de aplicar por medio de la tecnología XML sin embargo su implementación en una base de datos de este tipo hace que el problema sea solucionable.

Otra de las mejoras que aporta nuestro estudio es la validación de los datos insertados en un Modelo así como las reglas bajo las que se rigen. La Utilización de motores de BDD como Access hace que sean imposible ciertas validaciones o incluso mejorar la calidad de los datos insertados. *SQL Server* posee una herramienta de programación denominada SSIS (*SQL Server Integration Service*) que posee las cualidades perfectas para conseguir tanto la calidad del dato así como verificar reglas de validación de los mismos.

Otra de las mejoras propuestas tiene que ver con el uso de esta información en un modelo multidimensional o modelo en estrella. La explotación de los datos posee ratios de efectividad y respuesta muy rápidos para ser representado mediante informes y puestos a disposición de usuarios finales.

Ambas Mejoras son Explicadas a continuación.

5 Proceso de validación de los Datos del DPM

El proceso de inserción de datos es una de las partes más importantes ya que deben ser datos consistentes y deben abordar todos los requerimientos de negocio.

El enfoque para un proyecto de esta envergadura debe seguir el ciclo de vida de desarrollo de software (Figura 33).

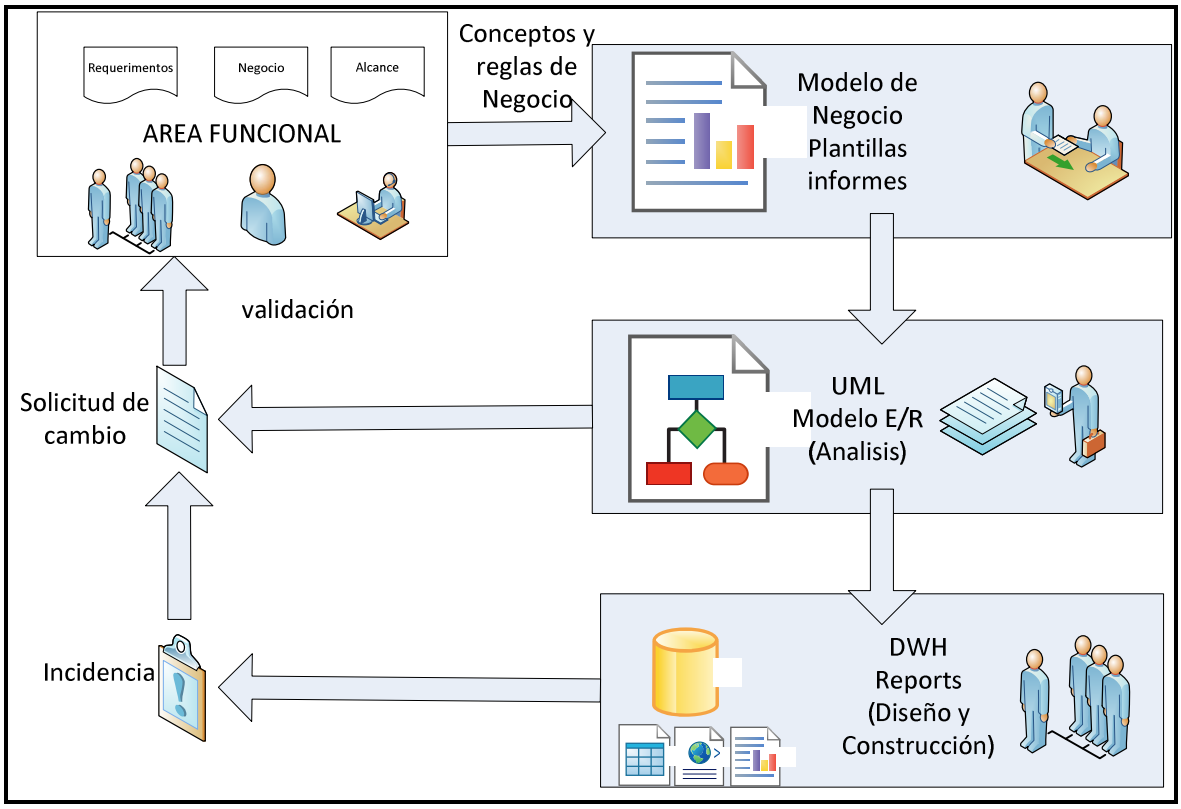


Figura 33. Ciclo de Vida. Proyecto SW

El mundo real es el conjunto de normas de contabilidad, leyes, directivas europeas, etc., que se define en un conjunto de datos necesarios en un informe, a través de plantillas. Que definen los usuarios Expertos, De acuerdo con esto se analiza el conjunto de definiciones, reglas de usuario y datos.

Los datos que recopila el EBA ya tienen una consistencia y un enfoque determinado, de cualquier modo nuestro desarrollo está orientado a predecir cómo debería ser la inserción de datos proporcionados por una determinada área de negocio por lo que se han migrado estos datos a plantillas Excel, con el fin de asemejar el proceso en un entorno lo más real posible, Cuyo fin será Insertar estos datos en nuestra BD en *SQL server*.

La plataforma específica elegida es SQL Server y el modo de integración de los datos será mediante SSIS (*SQL server Integration Service*).

5.1 Representación del proceso

El proceso propuesto sigue el esquema de la figura 34.

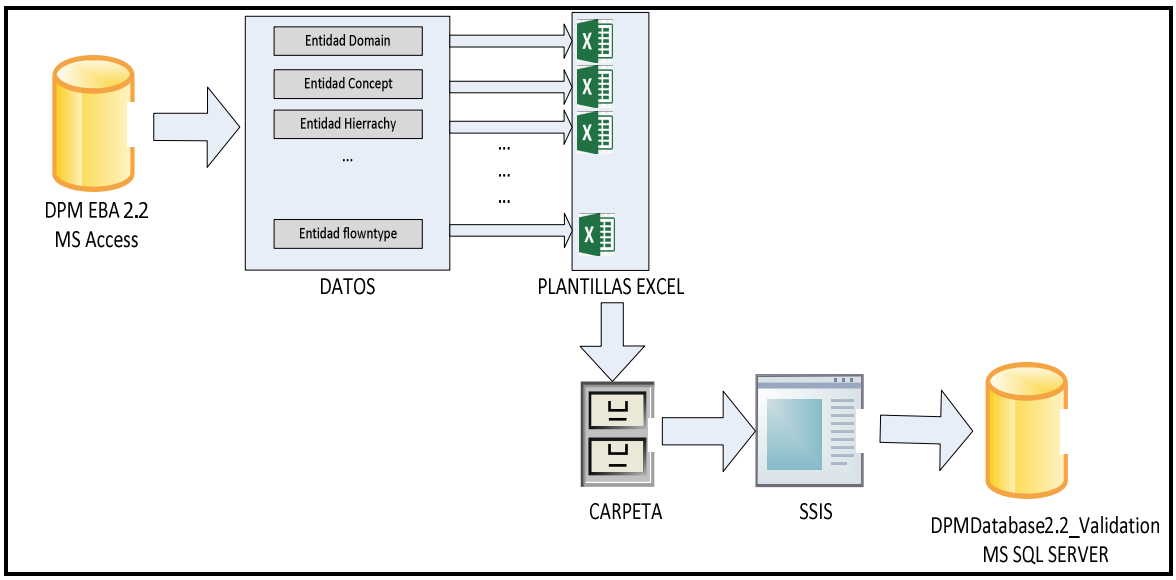


Figura 34. Flujo de inserción de datos Validados del DPM

El proceso se centra en la inserción y validación de datos de una parte del Modelo. Representado en la figura 35.

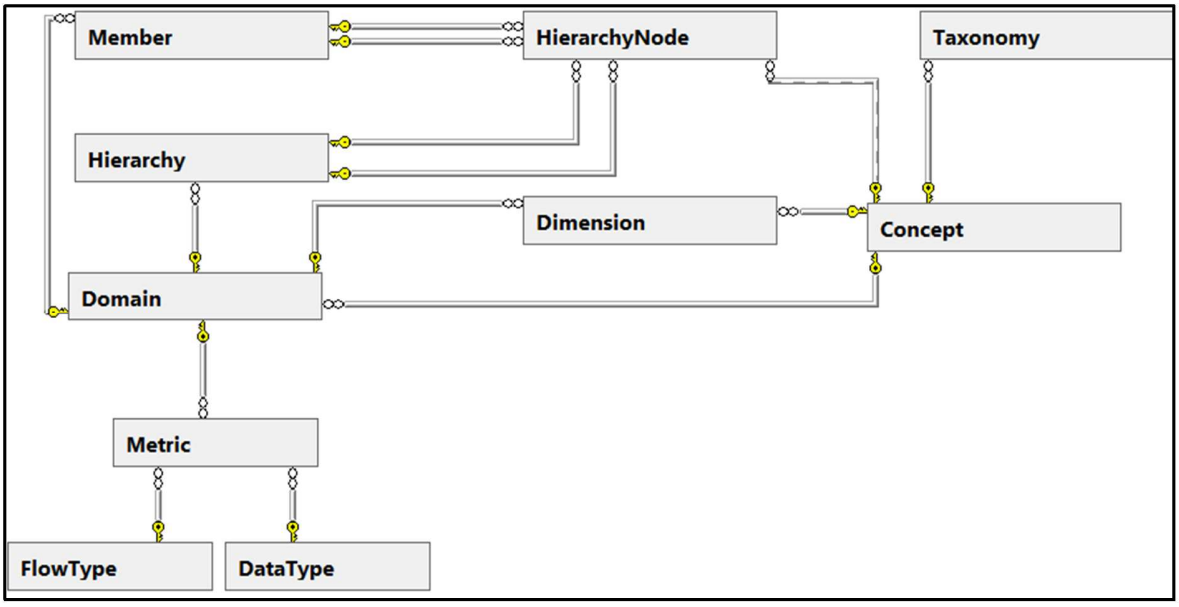


Figura 35. Modelo de Datos Validado mediante la herramienta de integración de datos SQL Server Integration Service.

5.2 Reglas de validación aplicadas en el estudio.

En esta Sección se propone todas las definiciones y reglas de validación. Estas definiciones se basan en el modelo de datos XBRL. Aunque, la sección presenta una nueva forma de Insertar y validar los datos del DPM.

Tabla	Orden Inserción	Regla de validación	Error
Concept	1	No existe IDconcept Duplicados	Primary Key restricted
Domain	2	No existe IDDomain Duplicados	Primary Key restricted
Domain	2	1 Concepto referencia N Dominios	foreign key restricted
Member	3	No existe IDMember Duplicados	Primary Key restricted
Member	3	1 Dominio referencia N Miembros	foreign key restricted

Dimension	4	No existe IDimension Duplicados	Primary Key restricted
Dimension	4	1 Dominio referencia N Dimensiones	foreign key restricted
Dimension	4	1 Concepto referencia N Dimensiones	foreign key restricted
Hierarchy	5	No existe IDhierarchy Duplicados	Primary Key restricted
Hierarchy	5	1 Dominio referencia N Jerarquias	foreign key restricted
HierarchyNode	6	No existe IDhierarchyNode Duplicados	Primary Key restricted
HierarchyNode	6	No existe IDhierarchyNode Padre con orden mayor de 1	Tree structure Restricted
HierarchyNode	6	No existe IDhierarchyNode hijo con orden igual 1	Tree structure Restricted
HierarchyNode	6	1 Miembro referencia N Nodos(Hijo)	foreign key restricted
HierarchyNode	6	1 Jerarquía referencia N Nodos(Hijo)	foreign key restricted
HierarchyNode	6	1 Jerarquía referencia N Nodos (Padre)	foreign key restricted
ReportingFramework	7	No existe IDReportingFramework Duplicados	Primary Key restricted
Taxonomy	8	No existe IDTaxonomy Duplicados	Primary Key restricted
Taxonomy	8	1 concept referencia N Taxonomías	foreign key restricted
Taxonomy	8	1 tipo de informe referencia N Taxonomías	foreign key restricted
Flowtypes	9	No existe IDFlowtype Duplicados	Primary Key restricted
DataTypes	10	No existe IDdatatype Duplicados	Primary Key restricted
Metric	11	No existe IDMetric Duplicados	Primary Key restricted
Metric	11	1 Dominio referencia N Métricas	foreign key restricted
Metric	11	1 flujo de datos referencia N Métricas	foreign key restricted
Metric	11	1 tipo de datos referencia N Métricas	foreign key restricted

Tabla 30. Reglas de validación aplicadas

El flujo de inserción de los datos sigue la secuencia de la figura 36.

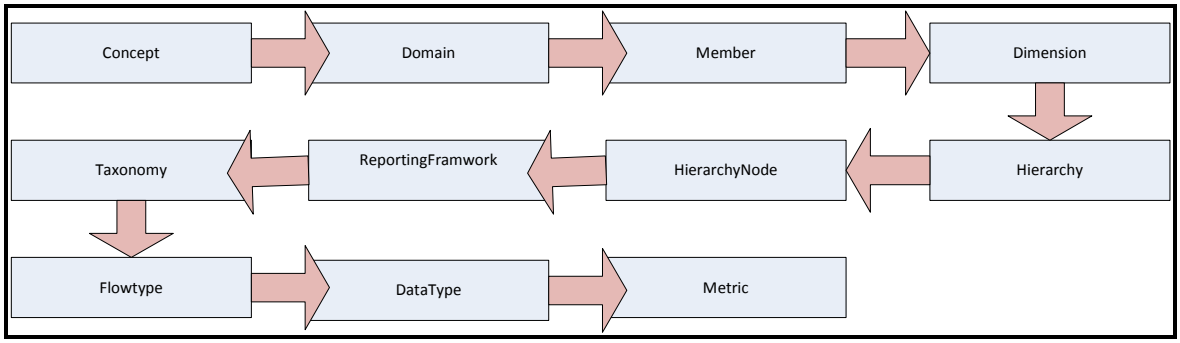


Figura 36. Flujo de inserción de datos mediante SSIS.

5.3 Desarrollo de la solución.

***SQL Server* proporciona *SQL Server Data Tools (SSDT)* para el desarrollo de paquetes de *Integration Services.* Comúnmente denominados paquetes ETL (*Extract Transform Load*)**

Una de las ventajas de utilizar SSIS es entre otras cosas:

- Entorno amigable para el programador. Y por tanto ahorro de tiempo en el desarrollo.
- Tareas automatizadas en la propia aplicación que solo necesitan parametrización. Conexiones *SFTP*, *SMTP*, *TELNET*, *OLEDB*, *ADO*,…
- Asistente de Importación y Exportación.
- Posibilidad de hacer Transformaciones y conversiones de datos en tiempo real.
- Posibilidad de trabajar con Script PL/SQL
- Posibilidad de trabajar con Script componente en .Net
- Ejecutar Servicios WEB
- Herramientas de Calidad de datos (*DQS data quality Service*).

Estas son algunas de las características.

Un paquete es una colección organizada de conexiones, elementos de flujo de control, elementos de flujo de datos, controladores de eventos, variables, parámetros y configuraciones que se pueden ensamblar con la ayuda de las herramientas gráficas de diseño proporcionadas por *SQL Server Integration Services* o mediante programación. El paquete es la unidad de trabajo que se recupera, ejecuta y guarda.

Al crear por primera vez un paquete, es un objeto vacío que no hace nada. Para agregar funcionalidad a un paquete, debe agregarle un flujo de control y, opcionalmente, uno o más flujos de datos.

El siguiente diagrama muestra un paquete individual que contiene un flujo de control con una tarea Flujo de datos que, a su vez, contiene un flujo de datos.

Esta es la estructura Básica que se repite a lo largo de toda la solución desarrollada.

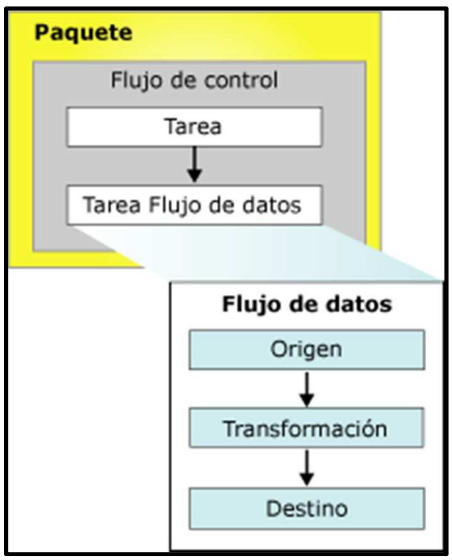


Figura 37. Estructura Paquete DTS. Referencia 15

Una vez creado el paquete básico, puede agregarle características avanzadas como registro y variables para extender su funcionalidad. Para obtener más información, vea la sección Objetos que extienden la funcionalidad de un paquete.

Lo primero es crear un proyecto de *Integration Service*.

1. Abrir *SQL Server Data Tools (SSDT).*
2. En el menú Archivo, seleccione Nuevo y haga clic en Proyecto.

Nuestro Proyecto se denomina EBA_XBRL_SSIS. Está en el apartado de Anexos. Para su ejecución es necesario ver los requerimientos SW y HW.

La estructura de nuestra solución sigue los estándares de programación descritos por la MSDN de Microsoft (Figura 37).

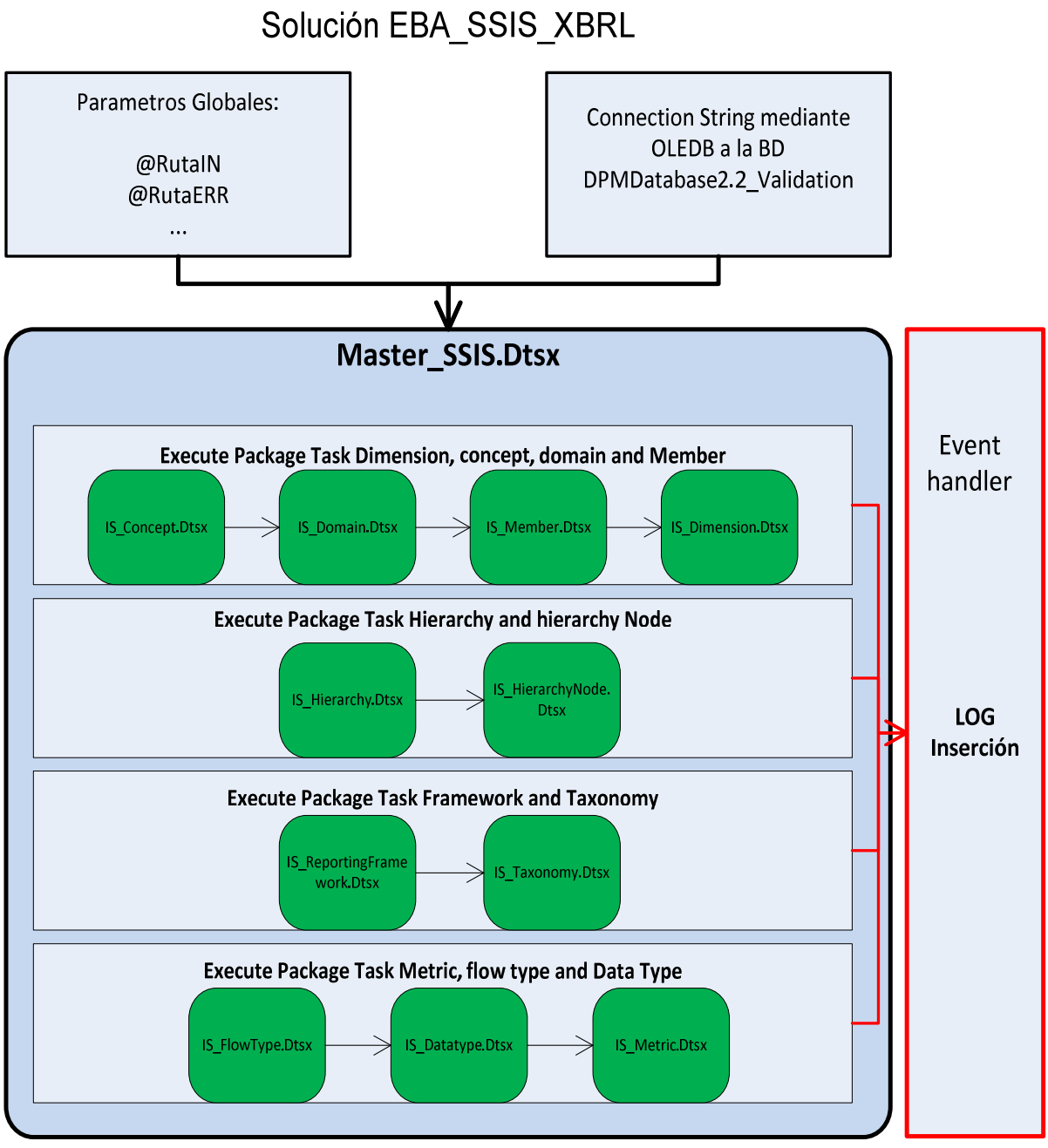
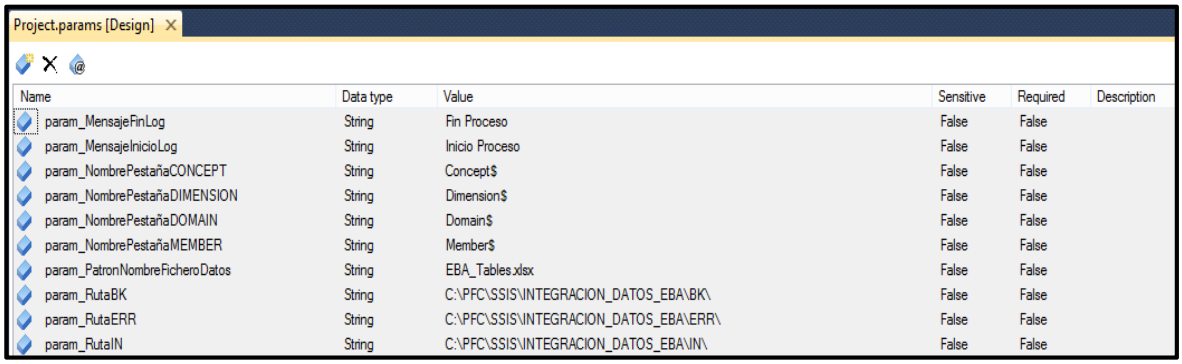


Figura 38. Flujo de ejecución Solución SSIS EBA_XBRL_SSIS.

La estructura de la solución engloba parámetros o variables de la solución, Archivo de configuración de conexiones a BDD, un paquete Maestro denominado IS_MASTER_EBA.Dtsx que es el paquete ETL de Inicio y el resto de Paquete ETL, por último se encuentra la herramienta de SSIS, controladora de eventos, que gestionará los mensajes de salida de nuestro programa.

5.3.1 Parámetros de la solución.

Existen dos archivos de configuración globales en las Soluciones de SSIS, el archivo de parámetros de la solución *Project.params* y el archivo de conexiones a BDD que en nuestro caso se denomina DPMDatabase2.2_Validation.conmgr.


Project.params [Design]

Name	Data type	Value	Sensitive	Required	Description
param_MensajeFinLog	String	Fin Proceso	False	False	
param_MensajeInicioLog	String	Inicio Proceso	False	False	
param_NombrePestañaCONCEPT	String	Concept$	False	False	
param_NombrePestañaDIMENSION	String	Dimension$	False	False	
param_NombrePestañaDOMAIN	String	Domain$	False	False	
param_NombrePestañaMEMBER	String	Member$	False	False	
param_PatronNombreFicheroDatos	String	EBA_Tables.xlsx	False	False	
param_RutaBK	String	C:\PFC\SSIS\INTEGRACION_DATOS_EBA\BK\	False	False	
param_RutaERR	String	C:\PFC\SSIS\INTEGRACION_DATOS_EBA\ERR\	False	False	
param_RutaIN	String	C:\PFC\SSIS\INTEGRACION_DATOS_EBA\IN\	False	False	

Figura 39. Parámetros de la Solución. Archivo Project Params.

Como se ve en la figura 39, en el archivo de configuración declararemos los parámetros que vamos a ir necesitando a lo largo del proceso. En nuestra solución hemos definido por un lado dos constantes de inicio y fin de proceso, para etiquetar el principio y el fin de la ejecución de cada Dtsx, El nombre del fichero de entrada que en nuestro caso será de tipo Excel y el nombre de sus Pestañas y las rutas de Entrada del Fichero, Salida y Error.

También se puede decidir si los parámetros serán sensibles a Minúsculas y Mayúsculas o si se tratan de parámetros obligatorios para la ejecución. Nosotros no hemos restringido ninguno de estos aspectos al no tener una orden explicita.

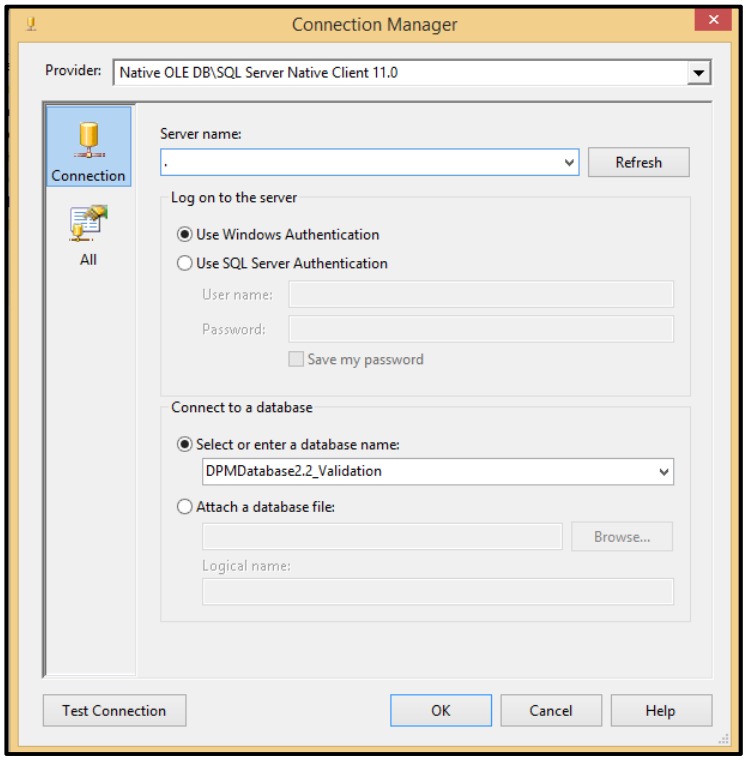


Figura.40 Configuración de conexiones en la solución de SSIS.

Con el administrador de conexiones generamos el archivo DPMDatabase2.2_Validation.conmgr. Primero rellenamos el servidor donde está alojada la BDD con la que vamos a interaccionar, en nuestro caso es el Local (.) y el Nombre de la BDD es DPMDatabase2.2_Validation, que es la BDD donde hemos creado un modelo relacional en *SQL Server 2012* sin datos (Definiciones en el apartado 5).

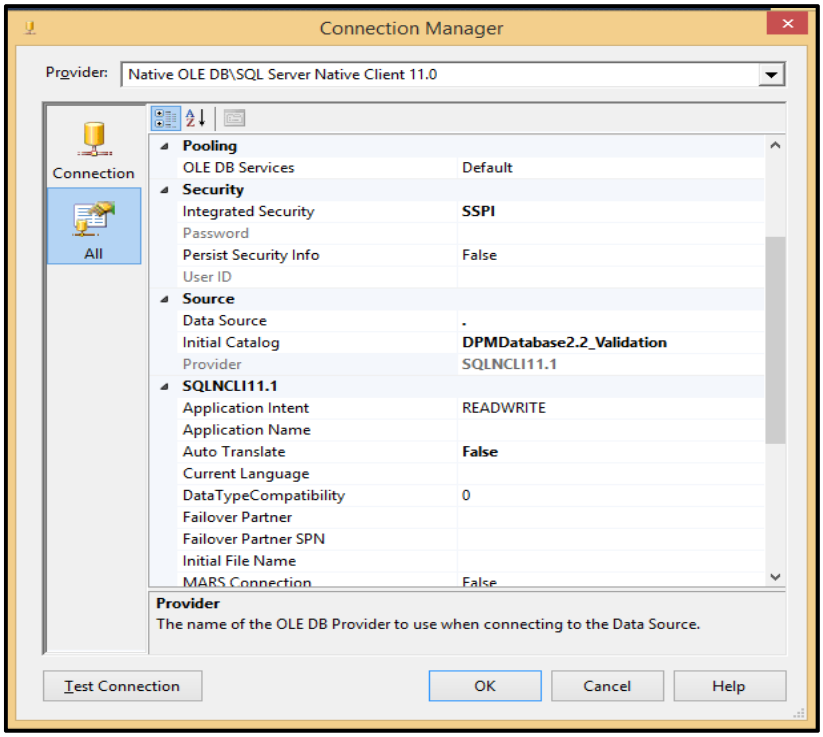


Figura.41 Opciones avanzadas. Configuración de conexiones en la solución de SSIS.

En las opciones avanzadas del archivo de configuración de conexiones, se puede elegir los tipos de conexión en nuestro caso hemos elegido *OLEDB*, pero existen más tipos de conexión como *ADO*, *SMTP*, *ODBC*,….

Existen más opciones de configuración en nuestro caso hemos decido tener la configuración por defecto, Por ejemplo en un entorno productivo lo habitual es que exista un usuario de servicio que ejecute las conexiones a BDD, en este caso debería definirse Usuario y clave.

Una vez configurada la conexión, es importante probarla para ello existe el botón *Test Connection*, con el cual podemos comprobar que la configuración es la idónea. Figura 42.

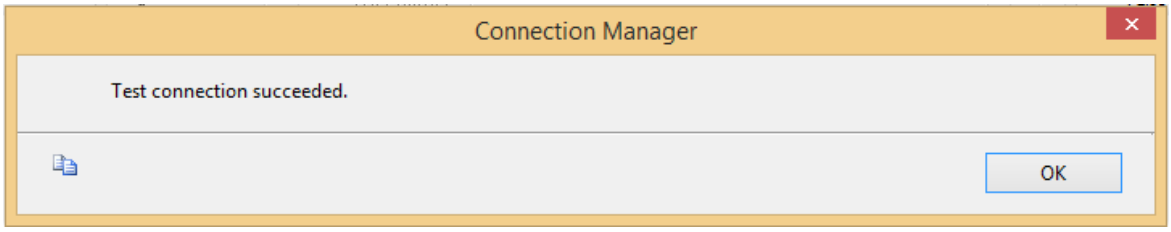


Figura 42. Mensaje Test de conectividad a la BDD.

El archivo de conexión a BDD posee el siguiente código fuente.

```
<?xml version="1.0"?>
<DTS:ConnectionManager xmlns:DTS="www.microsoft.com/SqlServer/Dts"
  DTS:ObjectName="W8-AVDEA03021.DPMDatabase2.2_Validation"
  DTS:DTSID="{2BE5256B-112A-4D39-8C4A-098213B8392B}"
  DTS:CreationName="OLEDB">
  <DTS:ObjectData>
    <DTS:ConnectionManager
      DTS:ConnectionString="Data Source=.;Initial
Catalog=DPMDatabase2.2_Validation;Provider=SQLNCLI11.1;Integrated Security=SSPI;Auto
Translate=False;" />
  </DTS:ObjectData>
</DTS:ConnectionManager>
```

Como se puede ver en el código el archivo de conexión a BDD es un XML.

Existen más tipos de archivos de configuración de conexión utilizados, pero estos han sido definidos a nivel de cada Paquete ETL. Se pueden Programar a nivel Global, pero en nuestro caso no era un requerimiento.

Por cada Paquete ETL creamos una conexión al fichero Excel con los datos de Entrada ya sean datos de Conceptos, Miembros, taxonomías, dimensiones…

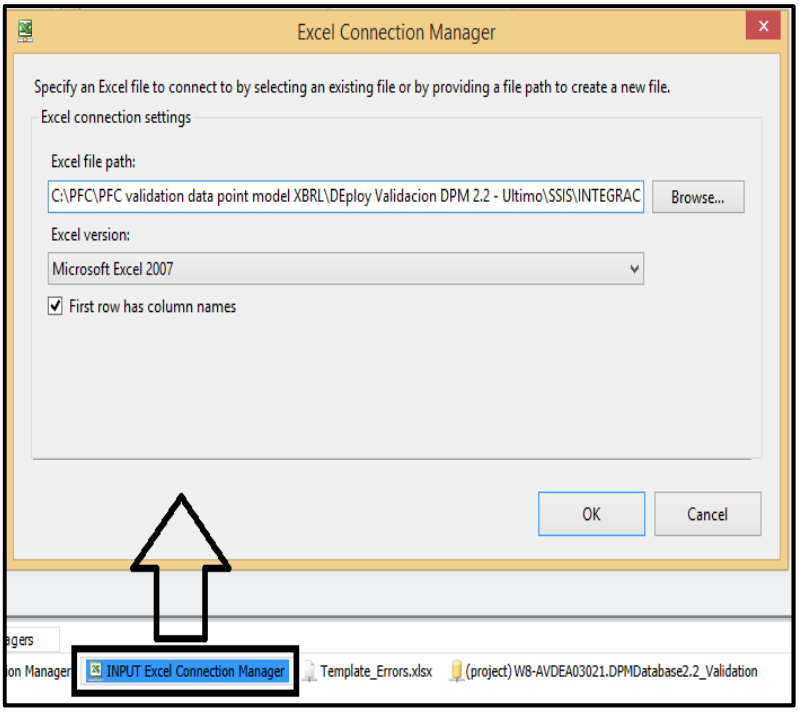


Figura 43. Pantalla de configuración de conexiones a ficheros Excel

El "*Excel file path*" de entrada puede definirse con los Parámetros Param_RutaIN + Param_PatronNombreFicheroDatos, definidos en el archivo de configuración de parámetros.

El otro tipo de conexión definido son los mensajes de salida de la ejecución de cada Paquete ya sea para la ejecución del paquete ETL de Conceptos, Miembros, taxonomías, dimensiones…

El tipo de conexión elegido es a ficheros de texto plano, en ellos se guardará el Log de ejecución del Paquete ETL.

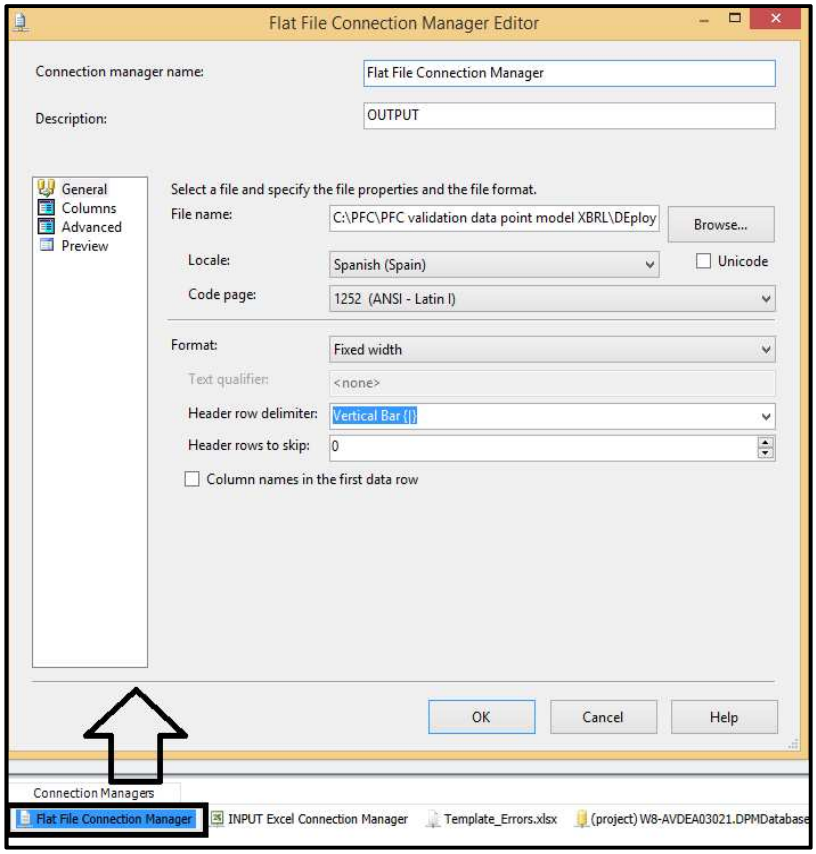


Figura 44. Pantalla de configuración de conexiones a ficheros Excel

5.3.2 Paquete Master de la Solución.

El paquete de Inicio de la solución es el denominado IS_MASTER_EBA.Dtsx.

Se encarga de ejecutar de forma Secuencial el Script de Borrado y las tareas de ejecución que corresponden a los distintos bloques de ejecución de cada Paquete ETL.

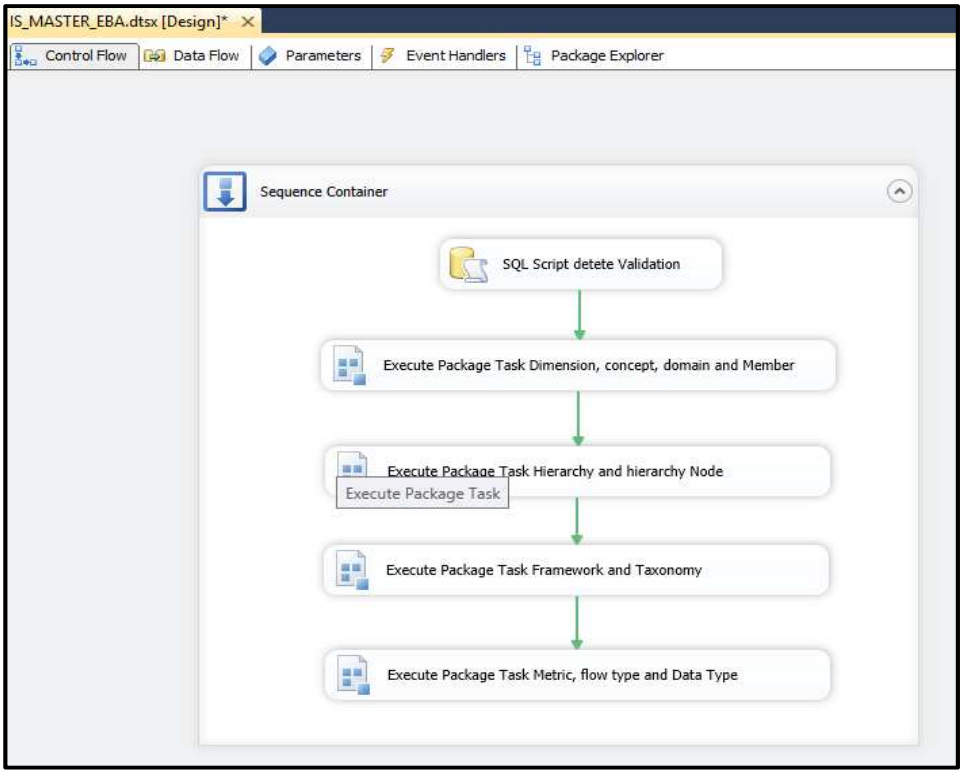


Figura 45. IS_MASTER_EBA.dtsx. Paquete de Inicio.

La primera tarea a ejecutar es un Script de borrado (*SQL Script delete Validation*) en cascada de los datos (En el caso de que existan) referentes a las tablas objeto del estudio, Esto se realiza debido a que uno de los requerimientos es que los datos se validan en conjunto es decir para cada ejecución validamos todos los datos de nuevo.

También borramos la tabla Log de errores, para que por cada ejecución solo quede el último Log. Como veremos más adelante La tabla está preparada para registrar cada Nueva Inserción pero preferimos realizar el borrado por cada ejecución para que fuese más claro y sencillo y de esta forma la tabla no crecerá de forma incremental.

El Script es el siguiente:

```
DELETE Taxonomy;
DELETE ReportingFramework;
```

```
DELETE Metric;
DELETE DataType;
DELETE FlowType
DELETE HierarchyNode;
DELETE Hierarchy;
DELETE DIMENSION;
DELETE MEMBER;
DELETE DOMAIN;
DELETE CONCEPT;
DELETE STG.LOG_ERRORS;
```

El resto de cajas de la figura 38 corresponde a las tareas de ejecución de cada Paquete ETL, los procesos son:

- Execute Package Task Dimension, concept, domain and Member. Que ejecuta los siguientes paquetes ETL de forma secuencial:
    - IS_CONCEPT.Dtsx.
    - IS_DOMAIN.Dtsx.
    - IS_MEMBER.Dtsx.
    - IS_DIMENSION.Dtsx.

- Execute Package Task Hierarchy and hierarchy Node. Que ejecuta los siguientes paquetes ETL de forma secuencial:
    - IS_HIERARCHY.Dtsx
    - IS_HIERARCHYNODE.Dtsx

- Execute Package Task Framework and Taxonomy. Que ejecuta los siguientes paquetes ETL de forma secuencial:
    - IS_REPORTINGFRAMEWORK.Dtsx.
    - IS_TAXONOMY.Dtsx.

- Execute Package Task Metric, flow type and Data Type. Que ejecuta los siguientes paquetes ETL de forma secuencial:

- IS_FLOWTYPE.Dtsx.
- IS_DATATYPE.Dtsx.
- IS_METRIC.Dtsx.

En la figura 38 está representado el flujo de ejecución.

5.3.3 Paquetes DTS de la solución.

Los paquetes DTS de la solución independientemente de los datos tienen la misma Estructura, a excepción del IS_DIMENSION, que posee una tarea de ejecución adicional que crea la tabla de hechos del modelo de datos multidimensional, la cual se explicará más adelante.

Por esta razón vamos a tomar como punto de Partida a las Explicaciones el Dts IS_CONCEPT.

La estructura del flujo de ejecución de los DTS sigue el estándar tal y como está definido en la figura 38. En este apartado vamos a explicar en profundidad como se define el flujo de datos.

La primera caja del flujo de datos corresponde al fichero de entrada de los datos a validar. Como se puede ver en la figura 46. El Paquete Dts establece una conexión de tipo Excel tal y como explicamos en el apartado 6.3.1.

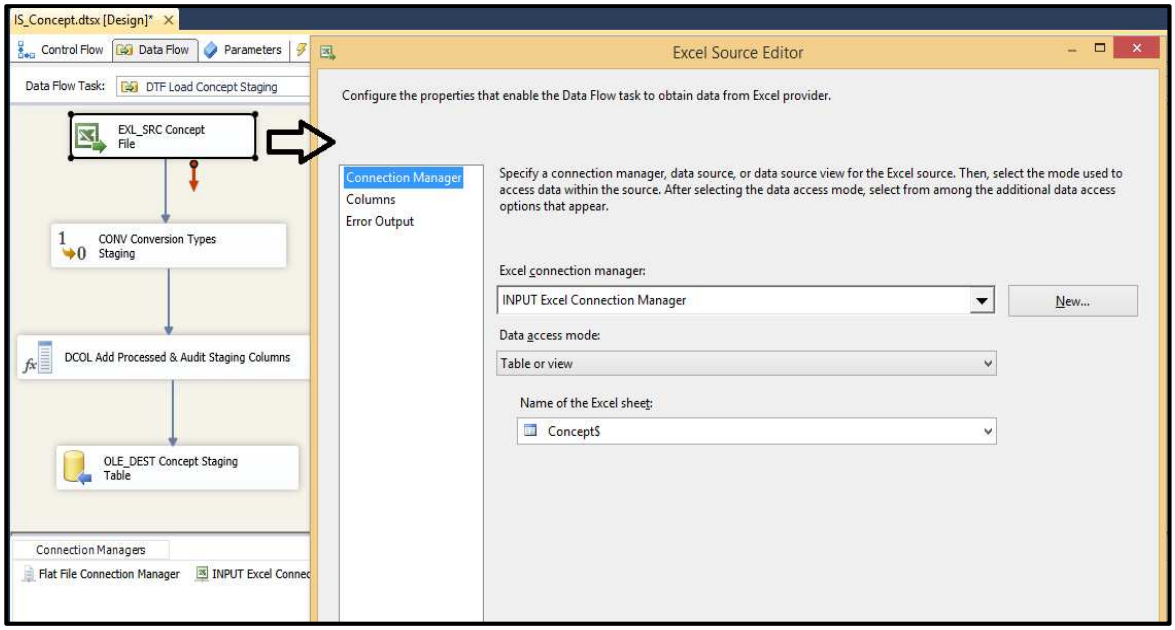


Figura 46. Flujo de datos. Conexión al fichero de entrada Excel.

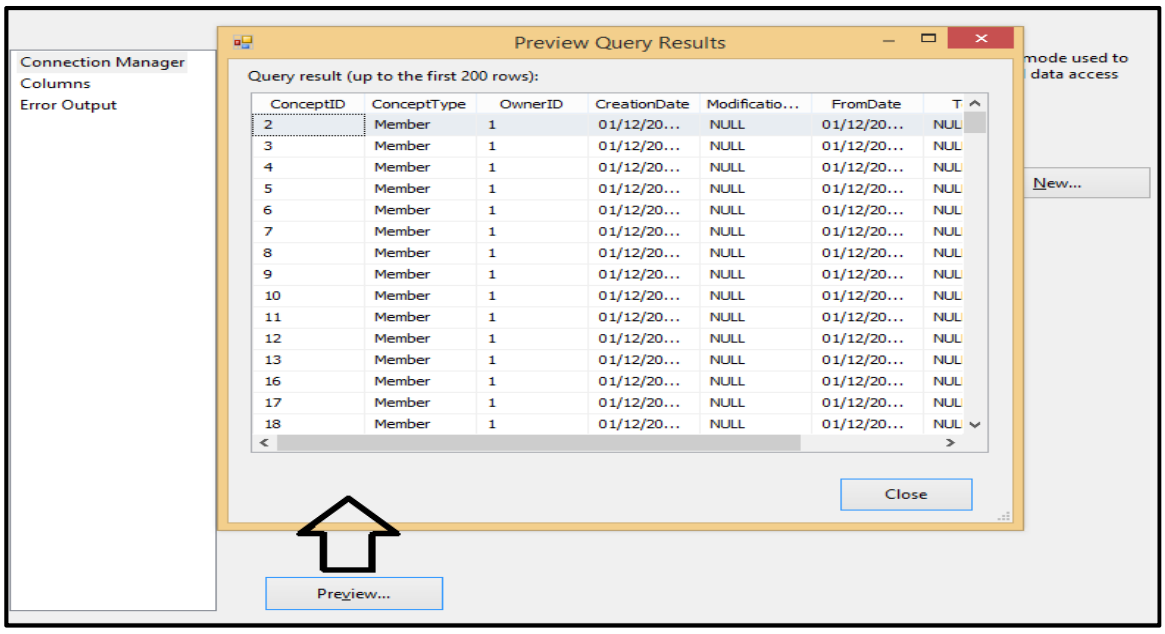


Figura 47. Flujo de datos. Conexión al fichero de entrada Excel. Preview.

Existe la opción de vista preliminar, es una buena práctica utilizar dicha vista para saber que hay datos susceptibles de ser insertados y es una prueba de que la conexión al fichero de datos es correcta.

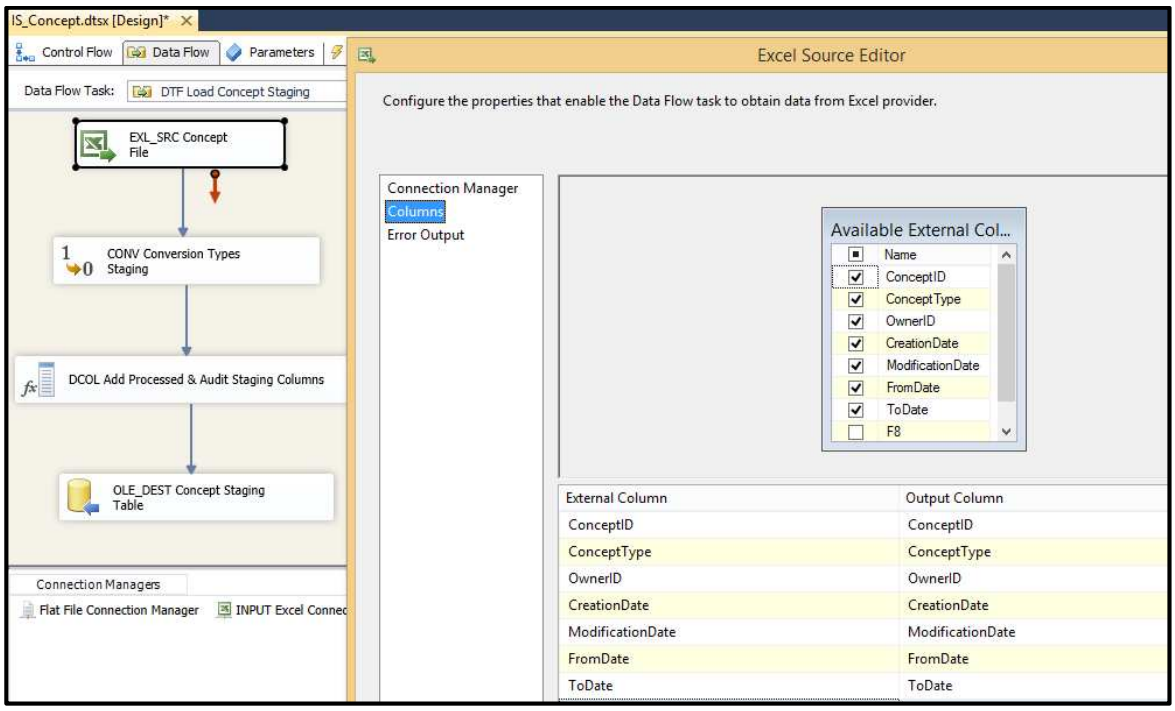


Figura 48. Flujo de datos. Conexión al fichero de entrada Excel. Columns

En la Opción *Columns*, podemos ver el nombre de las columnas del fichero de entradas, en nuestro caso poseemos nombres descriptivos en el fichero de entrada lo cual facilitará el posterior Mapeo campos de entrada vs campos Tabla SQL. También permite obviar columnas marcando o desmarcando el *check* al efecto.

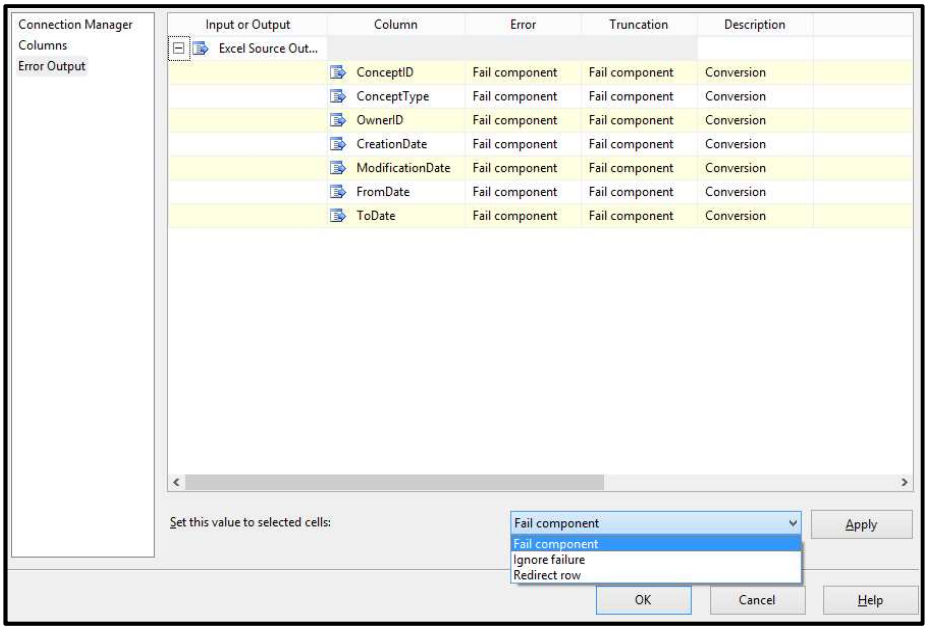


Figura 49. Flujo de datos. Conexión al fichero de entrada Excel. Error Output

Existe la Opción de parametrizar errores en este punto, pero en nuestro caso hemos gestionado los errores de forma homologa mediante un proceso de control de errores programado en el propio DTS por el cual cada error quedará reflejado en una tabla de BDD y un fichero Texto de salida, pero será explicado en más detalle en el apartado 6.3.4.

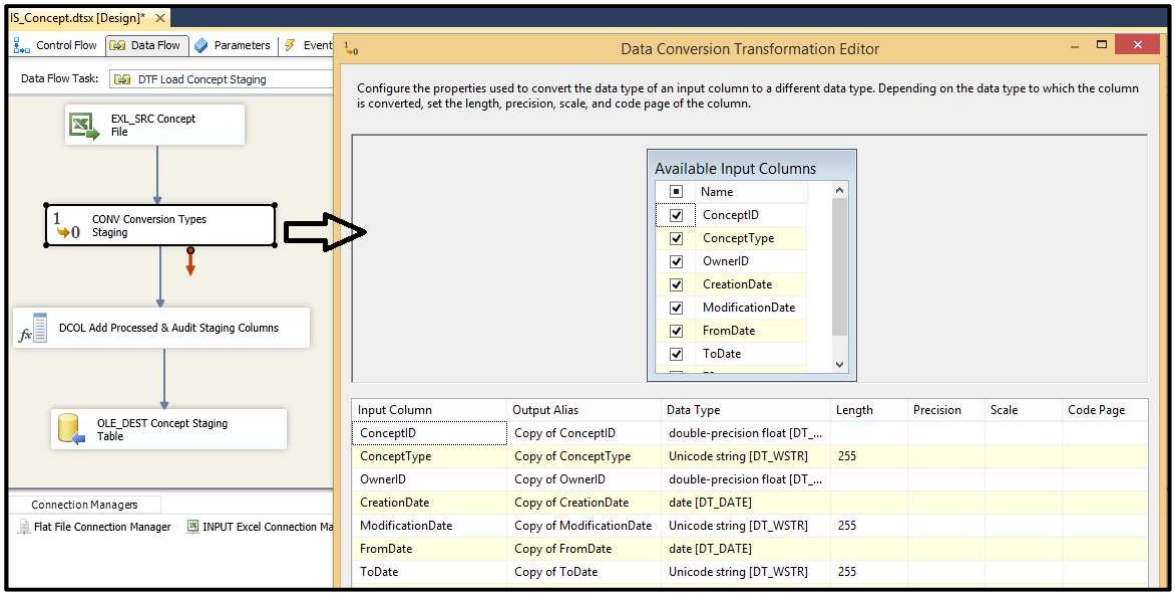


Figura 50. Flujo de datos. Conversión de datos del fichero de entrada.

La segunda caja realiza la parte de la transformación de los datos para que estos sean entendibles según las especificaciones de la BDD, ya que los tipos de datos y longitudes provenientes de un Excel pueden ser diferentes a los definidos en el modelo.

La transformación Conversión de datos convierte los datos de una columna de entrada a otro tipo de datos diferente y después los copia a una nueva columna de salida. Como podemos ver en la figura 50, el paquete extraer los datos de Origen de datos Excel y después usa esta transformación para convertir las columnas al tipo de datos necesario para el almacén de datos de destino. Se Pueden aplicar múltiples conversiones a una sola columna de entrada.

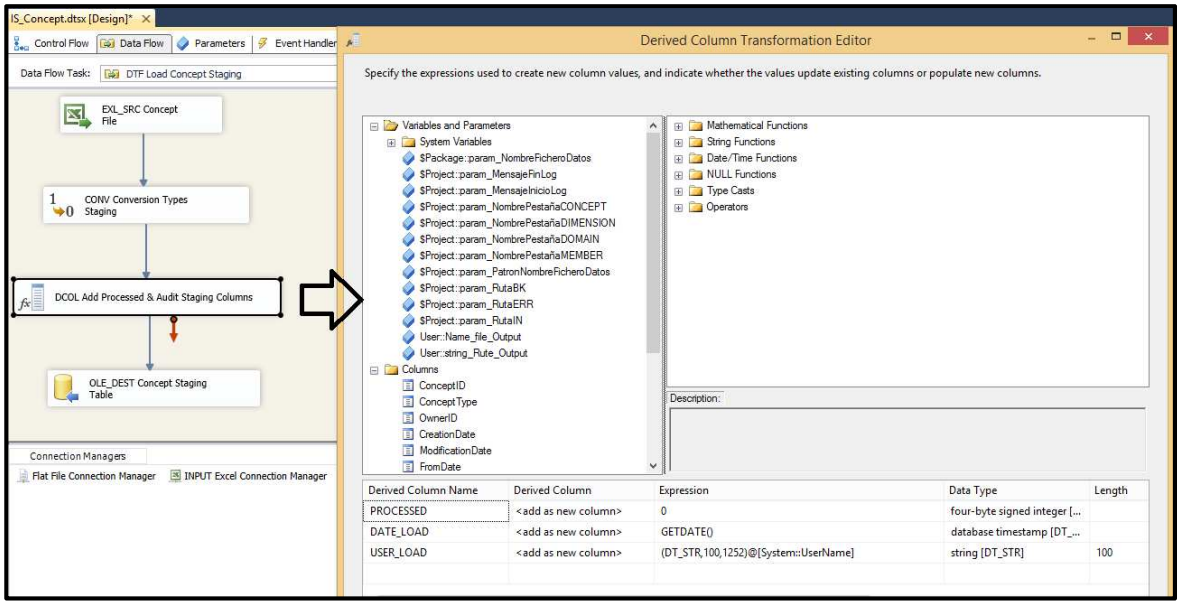


Figura 51. Flujo de datos. Transformación de datos de columna derivada.

La transformación Columna derivada crea nuevos valores de columna aplicando expresiones a las columnas de entrada de la transformación. Una expresión puede contener cualquier combinación variables, funciones, operadores y columnas de la entrada de transformación. El resultado puede agregarse como una nueva columna o insertarse en una columna existente como un valor de reemplazo. La transformación Columna derivada puede definir varias columnas derivadas, y cualquier variable o columna de entrada puede aparecer en varias expresiones.

En nuestro caso debido a que los datos de entrada son consistentes no ha sido necesario transformar los datos de entrada, pero si ha sido de utilidad para crear los campos de auditoria.

Hemos utilizado esta transformación de columna derivada para realizar las siguientes tareas:

- Campo PROCESSED. Es un campo nuevo. Posee una constante asociada 0.

- Campo DATE_LOAD. Es un campo nuevo. Qué referencia la fecha de ejecución le hemos asignado el valor *Getdate*() fechaactual con el formato dd-mm-yyyy hh:mi:ss, pero se puede utilizar las funciones GETDATE y DATEPART para extraer el año actual mediante la expresión *DATEPART("year",GETDATE()).* O similares si fuera necesario.
- Campo USER_LOAD. Es un campo nuevo. Hace uso de la variable del sistema *Username* la cual contiene el nombre de usuario que ejecuta el proceso. Es necesario convertir este valor debido a que el sistema lo proporciona como UNICODE STRING y el campo de la tabla de BDD es tipo VARCHAR.

Se puede utilizar para otras tareas como:

- Concatenar datos de distintas columnas en una columna derivada.
- Extraer caracteres de datos de cadena mediante funciones como SUBSTRING.
- Aplicar funciones matemáticas a datos numéricos y almacenar el resultado en una columna derivada.
- Crear expresiones que comparen columnas de entrada y variables

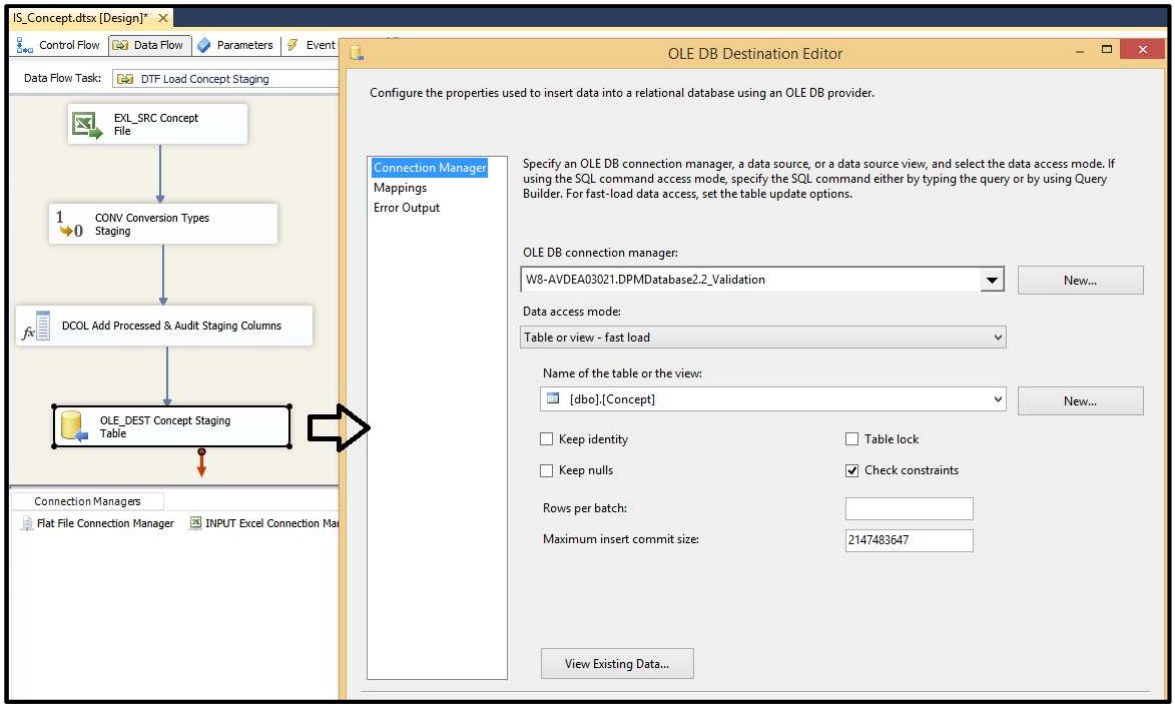


Figura 52. Flujo de datos. Destino OLE DB.

Por último, se produce la inserción en la BDD. Mediante la caja de tipo Destino OLE DB, se puede parametrizar la tabla de destino de los datos origen y el tipo acceso a la BDD. En Nuestro caso hemos elegido modo de acceso carga rápida lo cual nos proporciona especificar las siguientes opciones en el Editor de destino de OLE DB para el destino:

- Mantener los valores de identidad del archivo de datos importado o usar valores exclusivos asignados por SQL Server.
- Conservar un valor NULL durante la operación de carga masiva.
- Comprobar las restricciones en la tabla o vista de destino durante la operación de importación masiva.
- Adquirir un bloqueo de nivel de tabla durante la operación de carga masiva.
- Especificar la cantidad de filas del lote y el tamaño de confirmación.

Para cual modo de acceso a la BDD permite ver los datos existentes en nuestro caso estarán vacía a la hora de insertar, por el script de borrado previo definido en el IS_MASTER_EBA.Dtsx

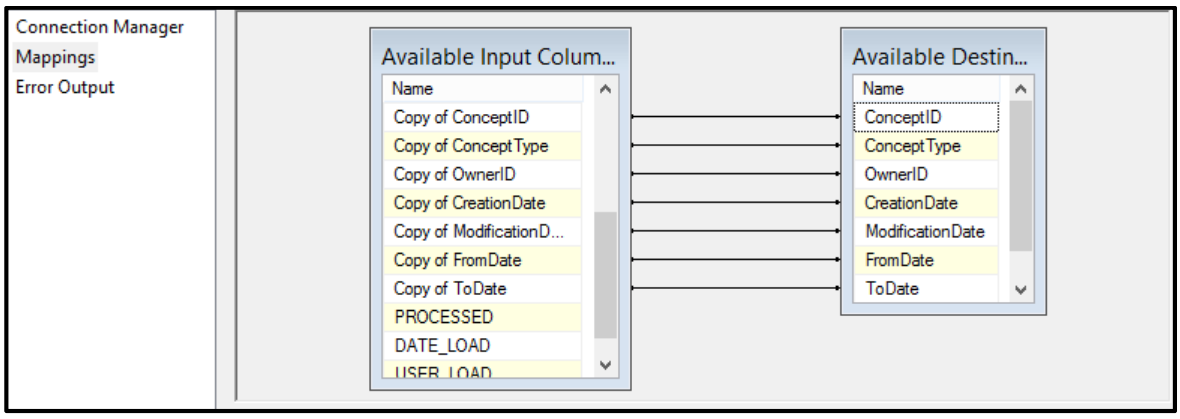


Figura 53. Flujo de datos. Destino OLE DB. Mappings.

Para terminar de Programar el OLE DB, se deben mapear los datos de origen (Que en nuestro caso serán los transformados en la conversión de datos) contra los campos destino en la BDD, el Mapeo se realiza de forma sencilla debido que los campos poseen la misma raíz en el nombre.

5.3.4 Control de errores en los Paquetes DTS de la solución.

El control de errores es parte muy Importante, dentro de cualquier validación de datos debido a que es el encargado de controlar las restricciones propias de nuestro modelo.

Para ello SSIS posee el controlador de eventos (*Event handler* en Inglés). En el controlador de eventos está programado un flujo de datos que se ejecutará siempre y cuando haya un error en la ejecución de cualquier paquete en cualquier punto y/o la ejecución de dicho DTS haya terminado.

El esquema de ejecución es el que se puede ver en la figura

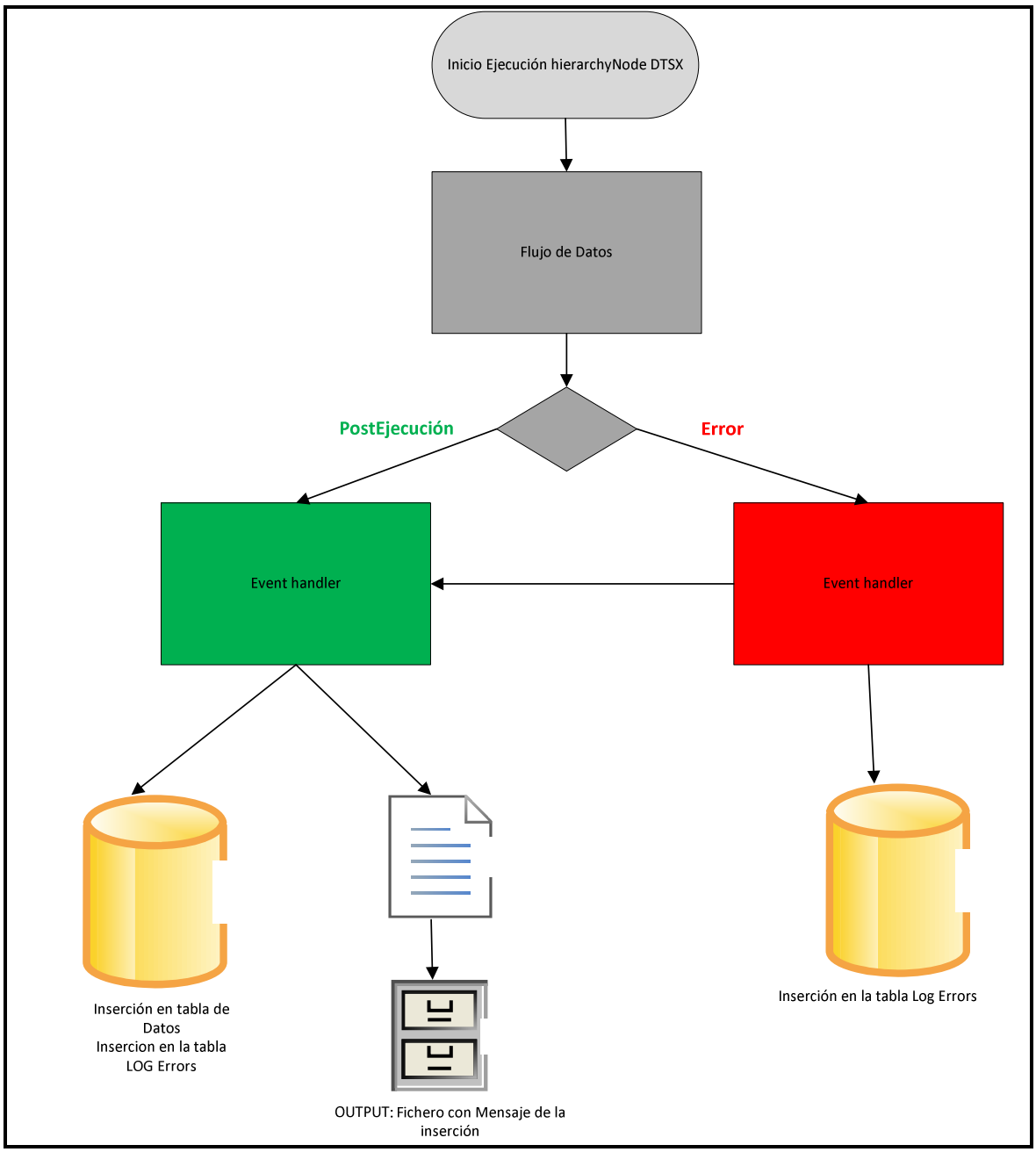


Figura 54. Flujo de ejecución. Control de errores y ejecución de un paquete DTS.

Tal y como apunta el esquema de ejecución fue necesario crear una estructura de datos en la BDD para almacenar estos LOG.

Siguiendo una buena práctica, esta tabla está asociada a un esquema distinto que el resto del Modelo. EL esquema empleado es STG.

Script Esquema STG:

```
USE [DPMDatabase2.2_Validation]
GO

/****** Object:  Schema [STG]    Script Date: 15/09/2015 1:31:37 ******/
CREATE SCHEMA [STG]
GO
```

La definición de la tabla es la siguiente:

Nombre Campo	Tipo	Long.	Descripción
ID_ERROR	Integer		ID auto numérico. Es la clave primaria
CODE_TABLE	varchar	255	Código de la tabla
ERROR_DATE	Datetime		Fecha del error.
DES_PROCESS	varchar	100	Nombre Proceso
DES_TASK	varchar	255	Nombre Tarea
COD_ERROR	varchar	100	Código de error
DES_ERROR	varchar	255	Descripción del Error
COD_USER	varchar	255	Usuario de ejecución

Tabla 31. Estructura Tabla LOG_ERRORS.

La tabla posee el siguiente Script:

```
USE [DPMDatabase2.2_Validation]
GO
/****** Object:  Table [STG].[LOG_ERRORS]    Script Date: 17/09/2015 4:00:38 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

SET ANSI_PADDING ON
GO

CREATE TABLE [STG].[LOG_ERRORS](
	[ID_ERROR] [int] IDENTITY(1,1) NOT NULL,
	[CODE_TABLE] [varchar](15) NULL,
	[ERROR_DATE] [datetime] NULL,
	[DES_PROCESS] [varchar](100) NULL,
	[DES_TASK] [varchar](max) NULL,
	[COD_ERROR] [varchar](100) NULL,
	[DES_ERROR] [varchar](max) NULL,
	[COD_USER] [varchar](max) NULL,
 CONSTRAINT [PK_LOG_ERRORS] PRIMARY KEY CLUSTERED
(
	[ID_ERROR] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO

SET ANSI_PADDING OFF
GO
```

El primer caso, susceptible de ocurrir es que el DTS de error en este caso el controlador de eventos está programado del siguiente modo.

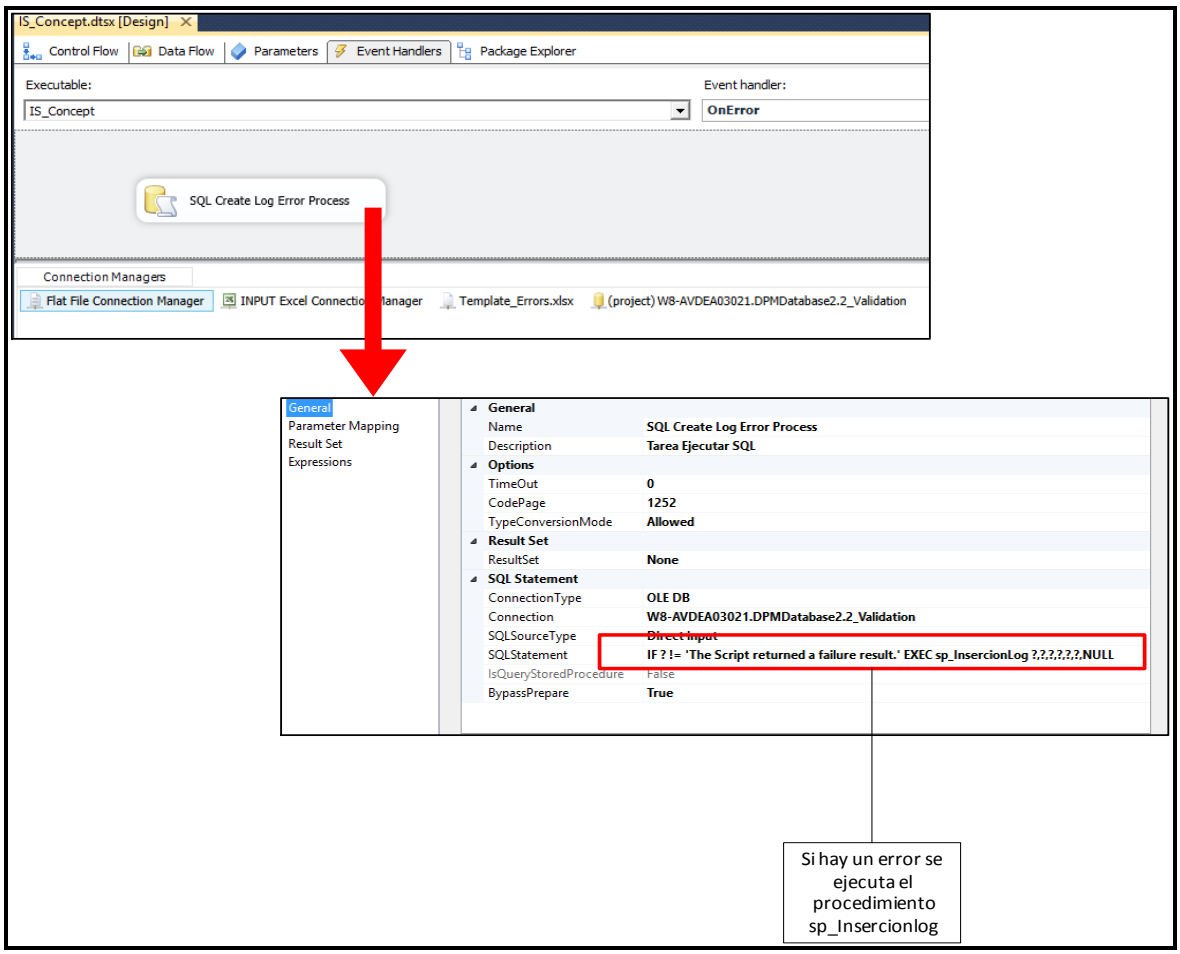
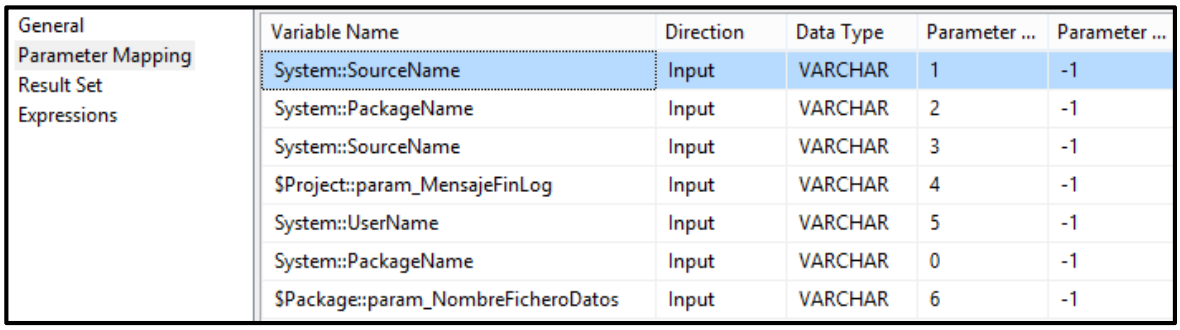


Figura 55. Flujo de ejecución. Control de errores de un paquete DTS.

Tal y como se dispone en la figura 55, está activado **Event handler → on Error. En el momento del error, se ejecuta un SQL Script de SSIS con el Código:**

```
IF ? != 'The Script returned a failure result.'  EXEC sp_InsercionLog ?,?,?,?,?,?,?,NULL
```

La ? (Interrogación) corresponden al orden de los parámetros de la figura 56.



Variable Name	Direction	Data Type	Parameter ...	Parameter ...
System::SourceName	Input	VARCHAR	1	-1
System::PackageName	Input	VARCHAR	2	-1
System::SourceName	Input	VARCHAR	3	-1
$Project::param_MensajeFinLog	Input	VARCHAR	4	-1
System::UserName	Input	VARCHAR	5	-1
System::PackageName	Input	VARCHAR	0	-1
$Package::param_NombreFicheroDatos	Input	VARCHAR	6	-1

Figura 56. Flujo de ejecución. Parámetros definidos en el Control de errores de un paquete DTS.

El pseudocódigo de la operación viene a decir si existe error Ejecutar el procedimiento almacenado *sp_InsercionLog* con los parámetros de entrada de la Figura 55.

El procedimiento almacenado tiene el siguiente código cuyo fin es insertar los valores en la tabla Log_Errors:

```
USE [DPMDatabase2.2_Validation]
GO
/****** Object:  StoredProcedure [dbo].[sp_InsercionLog]    Script Date: 15/09/2015
2:57:49 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[sp_InsercionLog]
       @CODE_TABLE [varchar](15),
       @DES_PROCESS [varchar](80),
       @DES_TASK [varchar](80),
       @COD_ERROR [int],
       @DES_ERROR [nvarchar](max),
       @COD_USER [varchar](40),
       @FILE_NAME [varchar](250)
WITH EXECUTE AS CALLER
AS
BEGIN
              SET NOCOUNT ON;

              DECLARE @ERROR_DATE datetime = GETDATE();
```

```
		DECLARE @ERROR_CHILD INT = 0;
		DECLARE @DES_ERROR_FINAL nvarchar(max)

		-- Check if error was previously inserted by child package
		SELECT
			@ERROR_CHILD = COUNT(0)
		FROM
			STG.LOG_ERRORS WITH (NOLOCK)
		WHERE
			CODE_TABLE IS NOT NULL AND
			ID_ERROR = (SELECT MAX(ID_ERROR) FROM [STG].[LOG_ERRORS] WITH
(NOLOCK)) AND
			DES_TASK = @DES_TASK AND
			COD_ERROR = @COD_ERROR AND
			DES_ERROR = @DES_ERROR

		-- If error description refers to start or end process, filename is
included on it
		IF @ERROR_CHILD = 0 AND @FILE_NAME IS NOT NULL
			SET @DES_ERROR_FINAL = @DES_ERROR + ' | ' + @FILE_NAME
		ELSE
			SET @DES_ERROR_FINAL = @DES_ERROR

		IF @ERROR_CHILD = 0
		BEGIN

			-- Insert log entry
			INSERT INTO STG.LOG_ERRORS (CODE_TABLE,
								DES_PROCESS,
								DES_TASK,
								COD_ERROR,
								DES_ERROR,
								ERROR_DATE,
								COD_USER)
			VALUES
				(REPLACE(@CODE_TABLE,'$',''),
				 @DES_PROCESS,
				 @DES_TASK,
				 @COD_ERROR,
				 @DES_ERROR_FINAL,
				 @ERROR_DATE,
				 @COD_USER)
		END

END
```

Haya o no haya error el controlador de eventos tiene programado un flujo de control pos ejecución, el cual posee un flujo de datos asociado tal y como aparece en la figura 57.

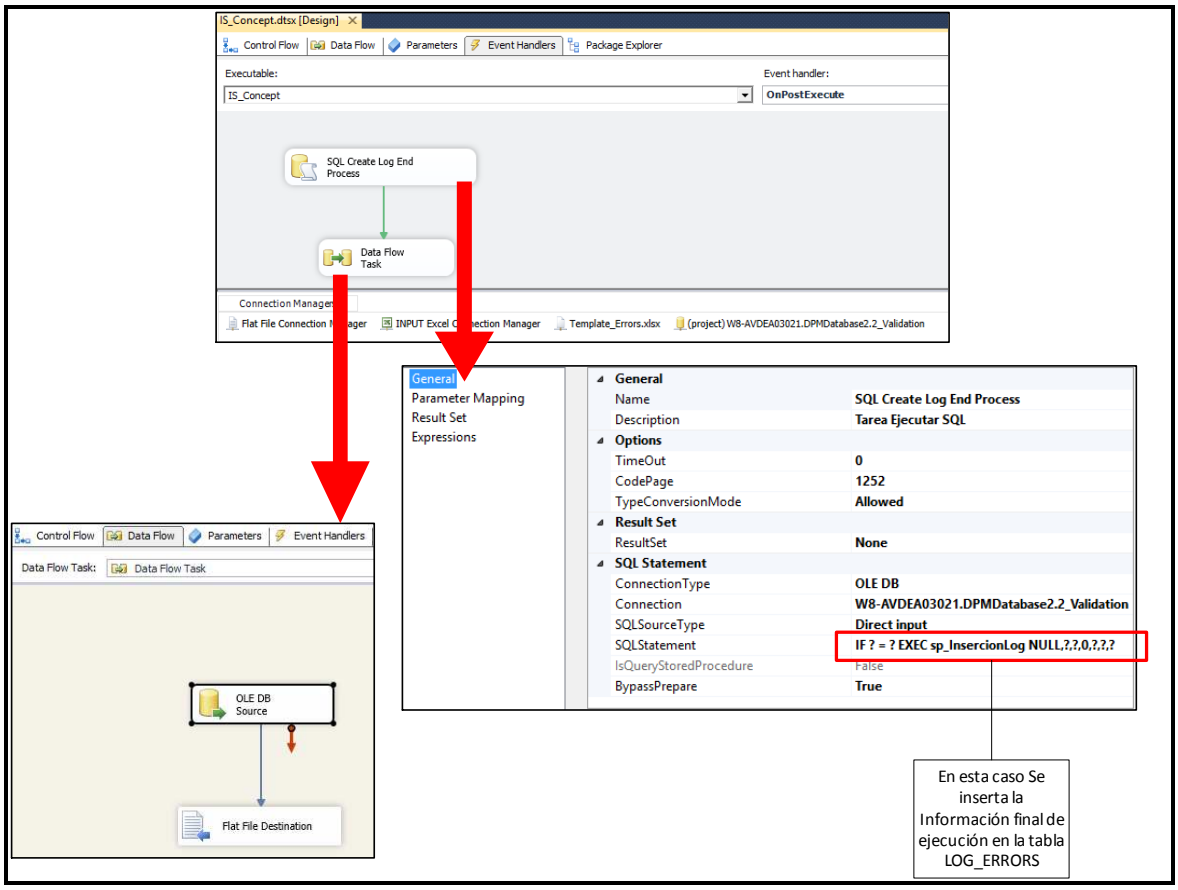


Figura 57. Flujo de ejecución. Control de fin de ejecución de un paquete DTS.

Primeramente, se ejecuta un SQL Script de SSIS, pero en este caso se ejecutará siempre no solo en caso de error.

```
IF ? = ? EXEC sp_InsercionLog NULL,?,?,0,?,?,?
```

Después se ejecuta el flujo de datos con Origen OLE DB tabla LOG_ERRORS destino una conexión a un fichero de texto plano.

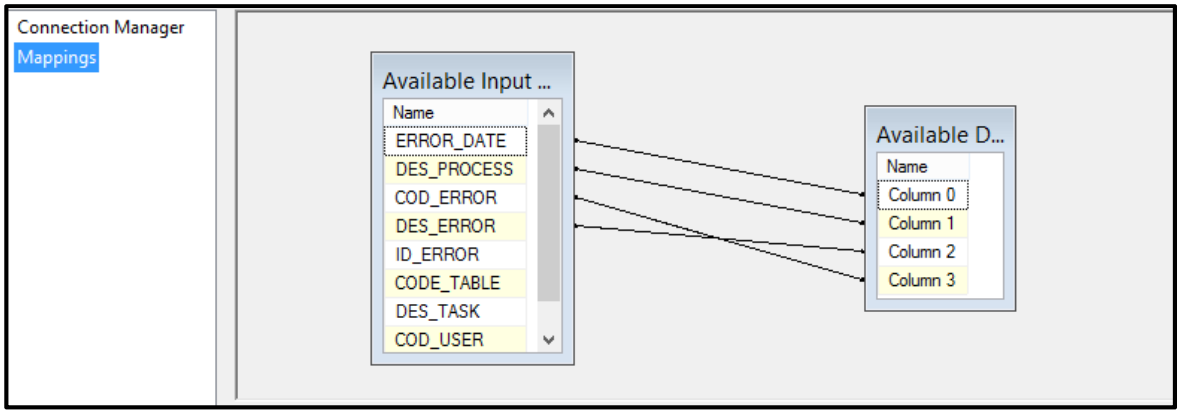


Figura 58. Flujo de ejecución. Control de fin de ejecución de un paquete DTS. Mapping

Tal y como se muestra en la figura 57, en el fichero de texto se guardan los campos ERROR_DATE, DES_PROCESS, COD_ERROR y DES_ERROR.

En el caso de que la ejecución sea satisfactoria los campos poseerán los siguientes valores:

- DES_ERROR = “Fin Proceso | EBA_Tables.xlsx”
- DES_PROCESS = “Nombre DTSX”
- COD_ERROR = “0”
- ERROR_DATE= “fecha Actual”

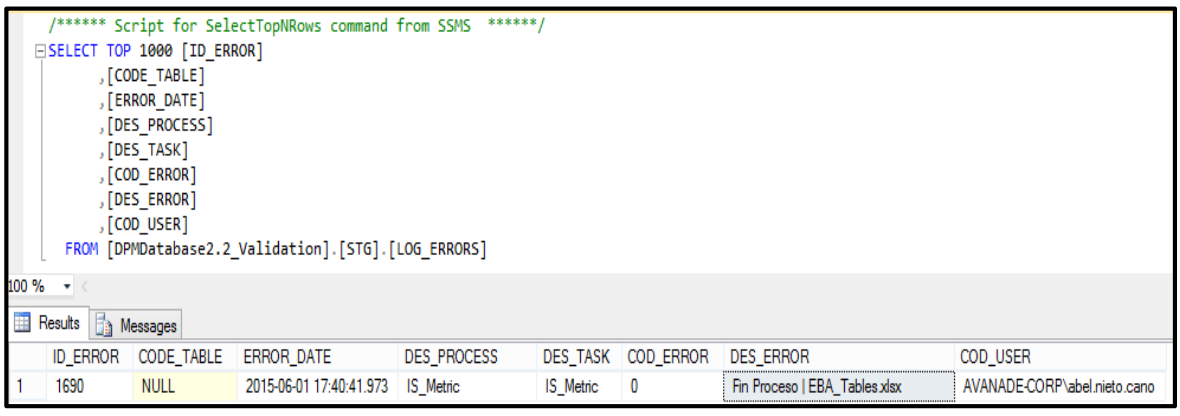


Figura 59. Ejemplo LOG de inserción satisfactoria en la tabla LOG_ERRORS.

5.3.5 Control de errores en los Paquetes DTS con Validaciones de Estructuras en árbol.

Para las validaciones de la tabla *Hierarchynode* la cual posee unas restricciones distintas a las del resto de tablas Optamos por un modelo de control diferente, tal y como se muestra en la figura 60

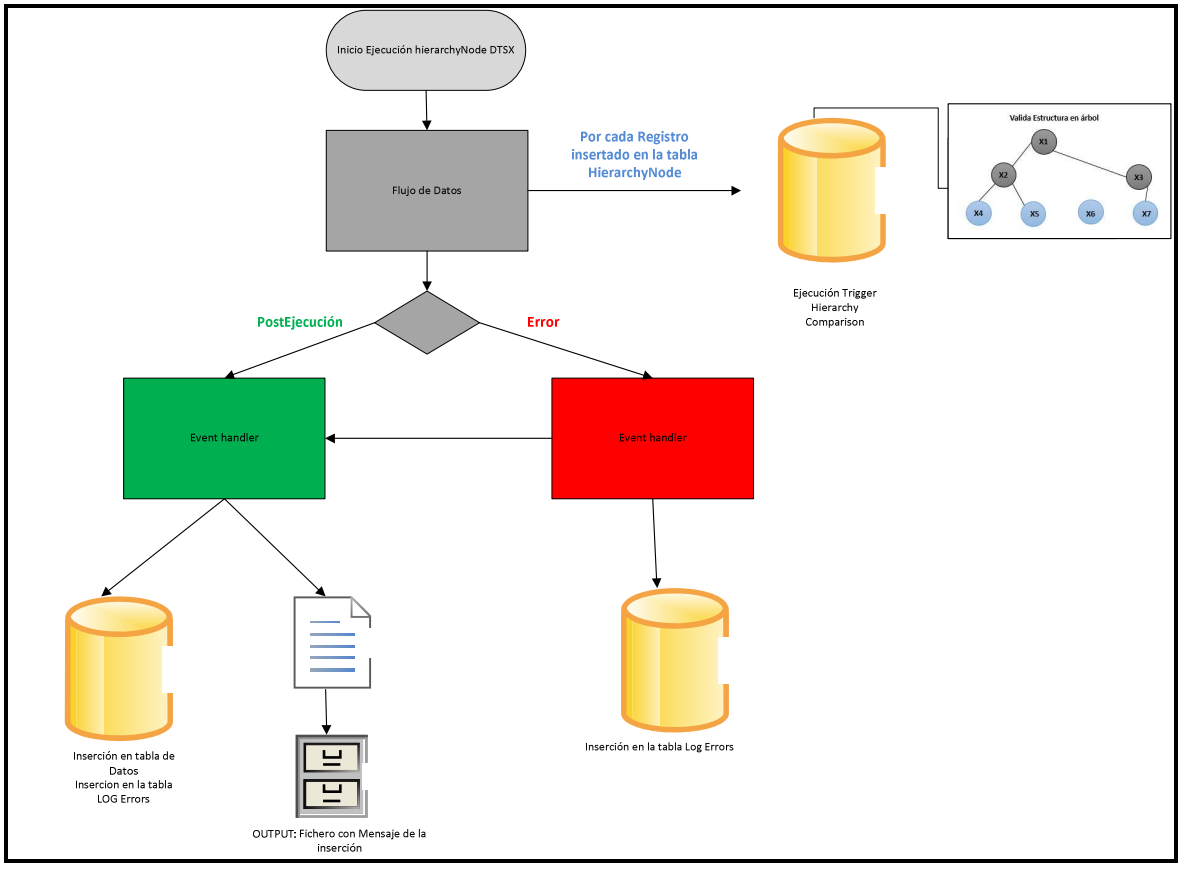


Figura 60. Proceso de validación de errores en el caso especial de tablas con estructuras en Árbol. IS_HierarchyNode.DTSX.

La estructura del DTS es igual pero con la salvedad que en la tabla de destino de la conexión *OLE DB* posee un *trigger*.

El *trigger* se encuentra programado en la tabla *HierarchyNode* de la BDD DPMDatabase2.2_Validation.

```
USE [DPMDatabase2.2_Validation]
GO

/****** Object:  Trigger [dbo].[HIERARCHY_COMPARISION]    Script Date: 15/09/2015
3:43:07 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO


CREATE TRIGGER [dbo].[HIERARCHY_COMPARISION]
ON [dbo].[HierarchyNode]


AFTER INSERT AS

DECLARE  @CUENTA AS INT
DECLARE  @Hid AS VARCHAR(6)
DECLARE  @HNid AS VARCHAR(6)

SET @CUENTA = 0

SELECT     @CUENTA = COUNT(*)
FROM  inserted  WHERE INSERTED.HierarchyID <> INSERTED.ParentHierachyID

SELECT     @Hid = cast(INSERTED.HierarchyID as varchar) ,@HNid =
cast(INSERTED.ParentHierachyID as varchar)
FROM  inserted  WHERE INSERTED.HierarchyID <> INSERTED.ParentHierachyID and
INSERTED.ParentHierachyID != null

IF @CUENTA <> 0

BEGIN

--INSERT INTO ERROR (CADENA) VALUES ('ERROR')

INSERT INTO [STG].[LOG_ERRORS]
           ([CODE_TABLE]
           ,[ERROR_DATE]
           ,[DES_PROCESS]
           ,[DES_TASK]
           ,[COD_ERROR]
           ,[DES_ERROR]
```

```
           ,[COD_USER])
     VALUES
           (NULL
           ,GETDATE()
           ,'IS_HierarchyNode'
           ,null
           ,'-000000001'
           ,'Description: "Cannot insert the value distinct to HierarchyID = '+ @Hid
+' into column ParentHierarchyID = '+  @HNid + ' , table DPM_Testing_Database
2.0.1.dbo.HierarchyNode; column does not allow distinct values into two columns.
INSERT fails.'
           ,null)

END

GO
```

Mediante este trigger se controla que no se insertan Nodos Hijo que posean la categoría de Padre de otro nodo de nivel Inferior. Y desde el trigger se produce la inserción directa en la tabla LOG_ERRORS en el caso de que se incumpla.

5.4 Pruebas de concepto efectuadas

Esta sección tiene como objetivo recopilar algunas de las pruebas de concepto efectuadas del código.

Como buena Practica hemos creado Dts Homólogos para los ficheros de entrada con un error introducido de forma deliberada.

Dts Concept.

Hay dos Dts:

- IS_Concept.dtsx → Datos EBA sin Modificar.
- IS_Concept Error 1.dtsx → Datos EBA modificados con una Clave Duplicada.

La prueba de concepto consiste en:

1. Ejecutar Paquete IS_Concept Error 1.dtsx. Pulsar F5

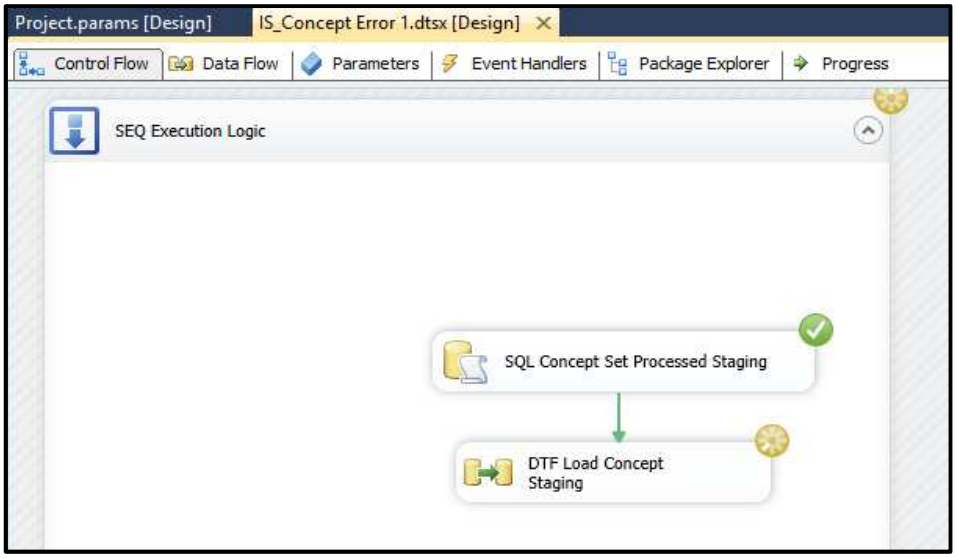


Figura 61. Test1. Proceso de validación de la tabla conceptos.

2. Este Dts dará un error debido a que posee una clave Duplicada en los datos a Insertar. Mediante el fichero.

EBA_Tables - Concept - Error.xlsx

	A	B	C	D	E	F	G
1	ConceptID	ConceptType	OwnerID	CreationDate	ModificationDate	FromDate	ToDate
2	2	Member	1	01-dic-13		01-dic-13	
3	2	Member	1	01-dic-13		01-dic-13	
4	3	Member	1	01-dic-13		01-dic-13	

Figura 62. Test1. Fichero de Entrada Modificado con un error de PK duplicada.

3. Comprobar la salida.

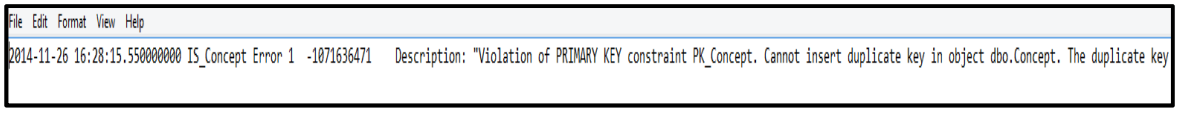


Figura 63. Test1. Mensaje de error.

Dts Member

Hay tres Dts:

- IS_Member.dtsx → Datos EBA sin Modificar. Con tablas Concept y Domain Rellenas
- IS_Member Error 1.dtsx → Datos EBA modificados con una Clave Duplicada.
- IS_Member Error 2.dtsx → Datos EBA sin Modificar Con tabla Concept Rellena y Domain vacía.

La prueba de concepto IS_Member Error 1 es la Misma que en el Dts Anterior. Se trata de una violación de Primary Key.

La prueba de concepto del IS_Member Error 2 consiste en:

1. Ejecutar Paquete IS_Member Error 2.dtsx. Pulsar F5

   Primero se integran los datos referentes a los datos concept.

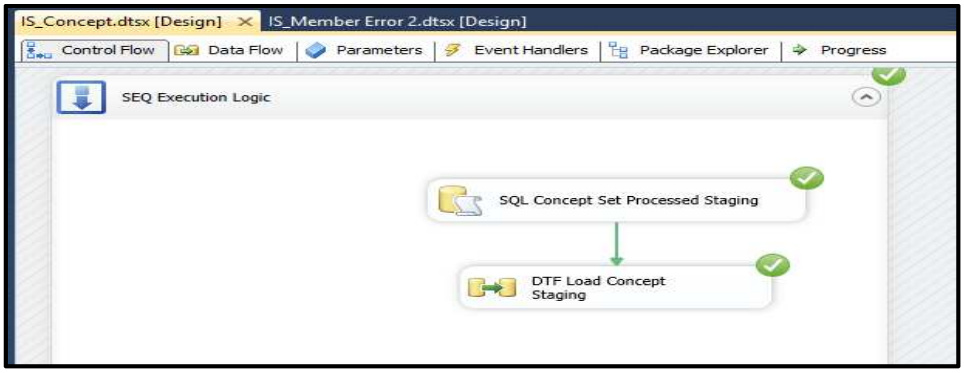


Figura 64. Test2. Proceso de validación de la tabla Miembros.

2. Este Dts dará un error debido a que posee una relación de clave ajena con las Tablas Concept y Domain, según estipula el EBA.

   Pero en nuestro caso solo hemos insertados lo datos referentes a los concept.

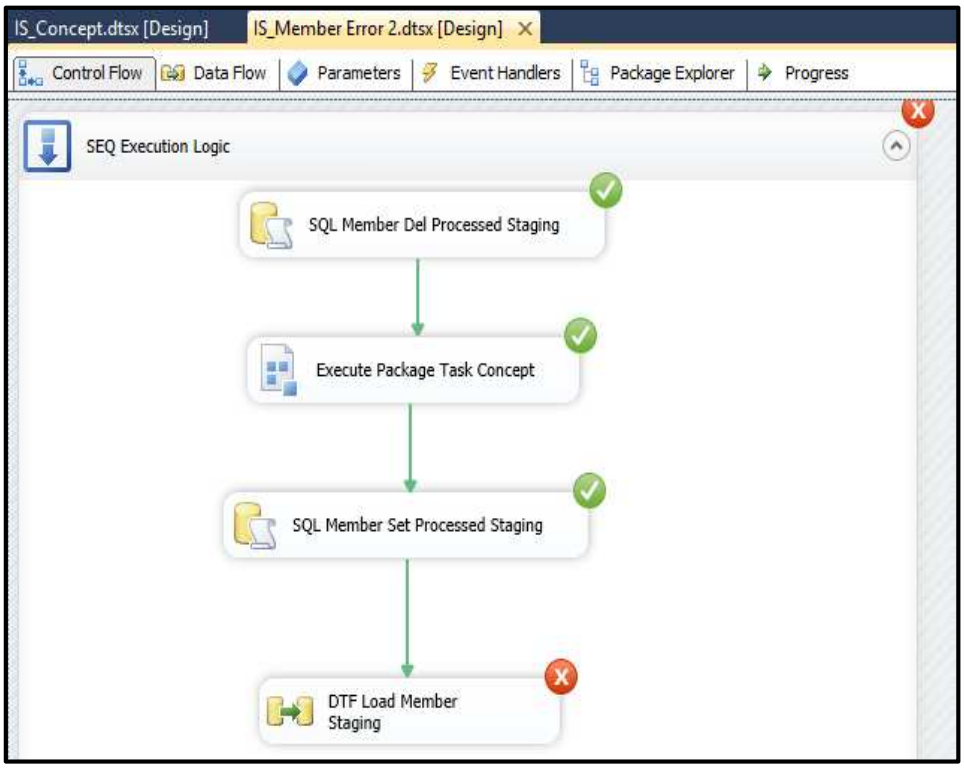


Figura 65. Test2. Proceso de validación de la tabla Miembros. Error Integridad referencial violada

3. Comprobar la salida.

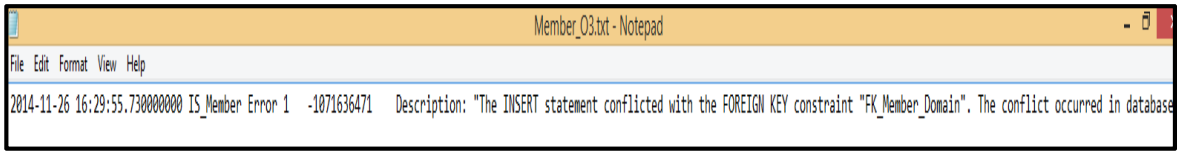


Figura 66. Test2. Mensaje de error.

Dts Dimension

Hay un Dts:

- IS_Dimension.dtsx → Datos EBA sin Modificar.

La prueba de concepto consiste en:

4. Ejecutar Paquete IS_Dimension.dtsx. Pulsar F5.

   Primero se integran los datos referentes a los datos Domain. Para evitar el error de FK ya validado en otra prueba que en este caso también está controlado. Y queremos que el proceso continúe, hasta la siguiente Validación.

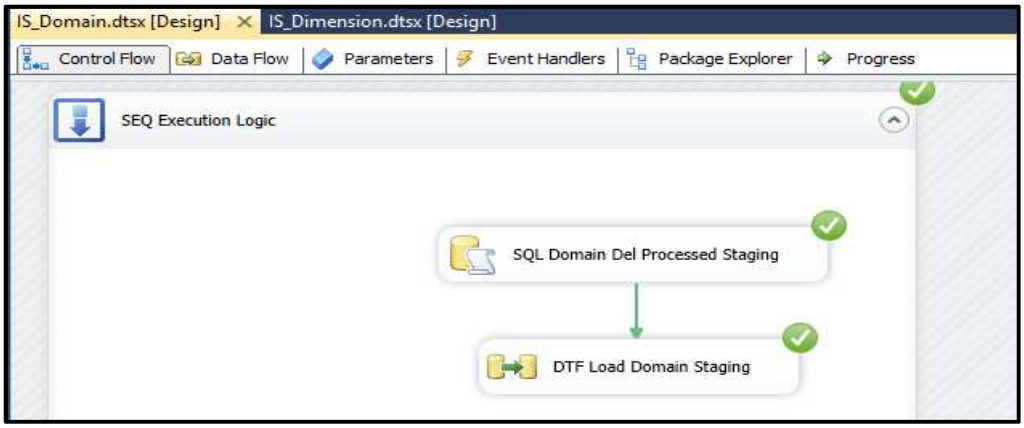


Figura 67. Test4. Proceso de validación de la tabla Dimensiones.

5. Este Dts dará un error debido a que posee una PK clave primaria con valor *Null*.

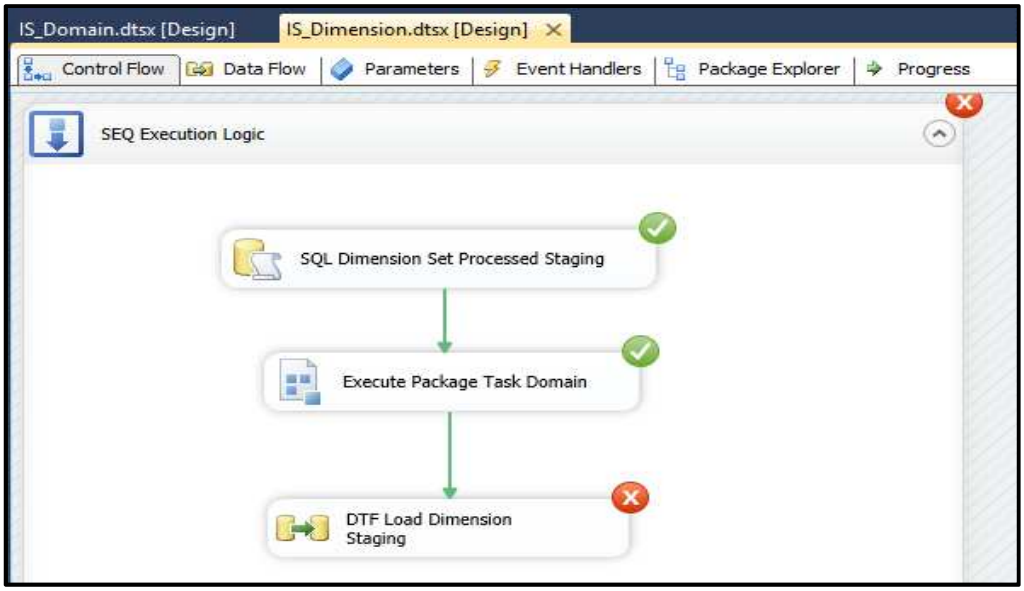


Figura 68. Test4. Proceso de validación de la tabla Dimensiones. Error PK con valor NULL.

EBA_TablesDimension.xlsx

105	790	380	ACT	Accountin	Defines w	FALSE	FALSE	eba_dim:ACT
106	795	530	INC	Individual	Indicate tl	TRUE	FALSE	eba_dim:INC
107	800	530	GCC	Group of c	Indicate tl	TRUE	FALSE	eba_dim:GCC
108	805	540	AST	Accountin	The accou	FALSE	TRUE	eba_dim:AST
109			NULL	NULL	NULL	NULL	NULL	NULL
110								

Figura 69. Test4. Fichero de Entrada Modificado con un error de PK NULL.

6. Comprobar la salida.

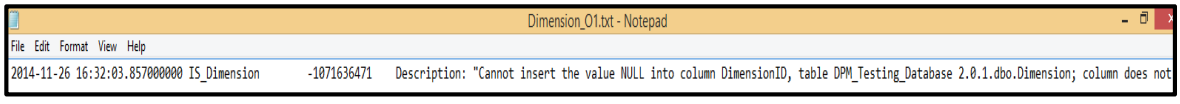


Figura 70. Test4. Mensaje de error.

Dts HierarchyNode.

Hay tres Dts:

- IS_HierarchyNode.dtsx → Datos EBA sin Modificar. Con tabla Hierarchy Rellena
- IS_HierarchyNode Error 1.dtsx → Datos EBA modificados con Campo IDHierarchyParent <> IDHierarchy.

La prueba de concepto del IS_HierarchyNode Error 1 consiste en:

4. Ejecutar Paquete IS_HierarchyNode Error 1.dtsx. Pulsar F5

   Primero se integran los datos referentes a los datos Hierarchy.

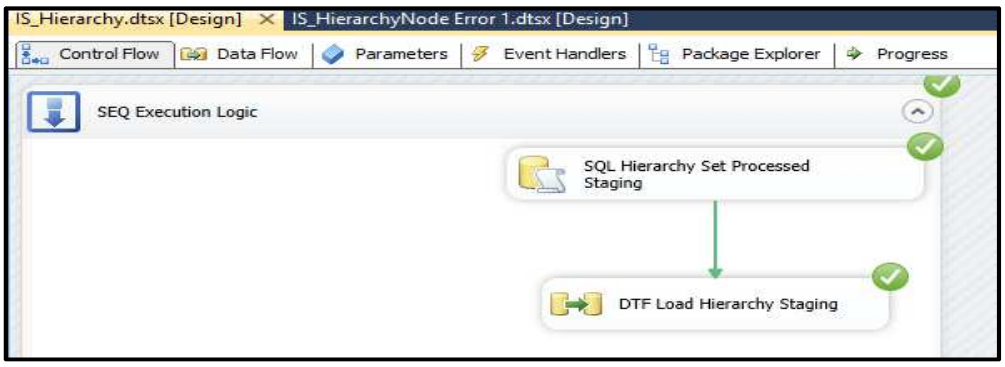


Figura 71. Test5. Proceso de validación de la tabla HierarchyNode. Estructura árbol Violada

5. Este Dts no dará un error debido a que posee un error Funcional entre los datos IDParentHierarchy e IDHierarchy. Insertará los datos que cumplen el criterio IDParentHierarchy = IDHierarchy. Pero Localizará y no insertará los que no cumplen dicha validación.

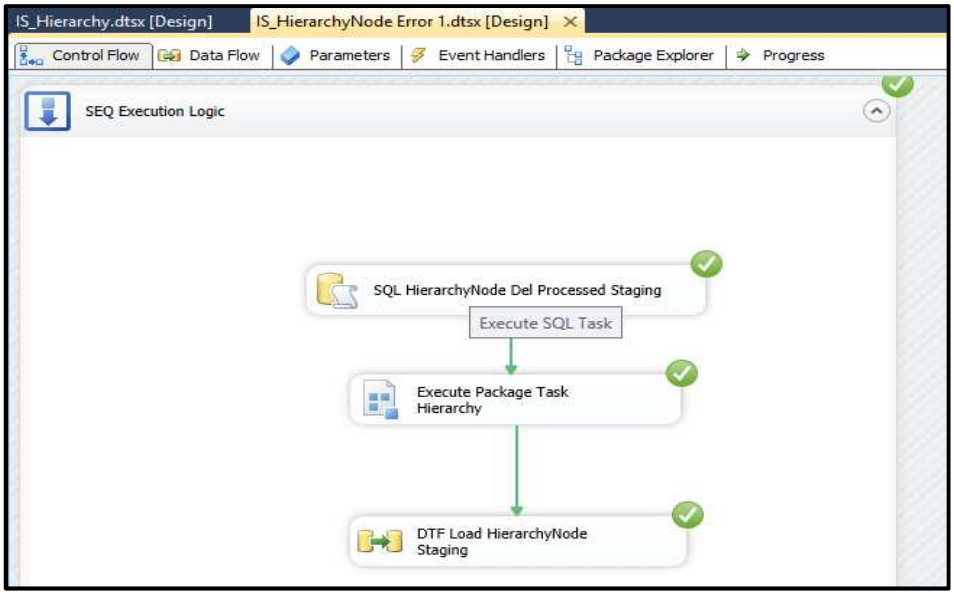


Figura 72. Test5. Proceso de validación de la tabla HierarchyNode. Estructura árbol Violada

EBA_Tables - HierarchyNode Error.xlsx

	A	B	C	D	E	F	G	H	I	J	K
1	HierarchyID	MemberID	IsAbstract	ComparisonOperator	UnaryOperator	Order	Level	Path	ParentHierachyID	entMemberID	
2	110	1000	FALSE		+	165	4	3677.2963	45	1055	
3	45	1006	FALSE		+	75	3	3677.3022	45	3022	
4	80	1007	FALSE		+	95	3	3677.3023	80	3023	

Figura 73. Test5. Fichero de Entrada Modificado con una violación de la estructura en árbol.

6. Comprobar la salida.

Avisa y localiza, los registros con el error Mencionado.

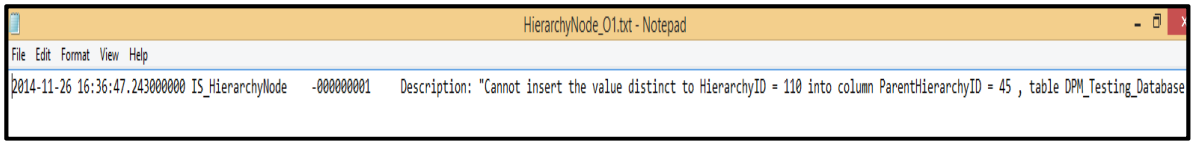


Figura 74. Test5. Mensaje de error.

6 Creación del Modelo multidimensional

6.1 Que es un modelo Multidimensional

El modelo de datos multidimensional (MDM) se presenta en este apartado y está destinado a ser un punto de partida para un proceso de modelado con posterioridad a las necesidades analíticas. Únicamente se refiere a los conceptos del DPM como hechos, que se concretaran en una tabla, para formar una tabla de hechos y todas las referencias asociadas a este Hecho, todo englobado en esta única tabla de hechos. Figura 75.

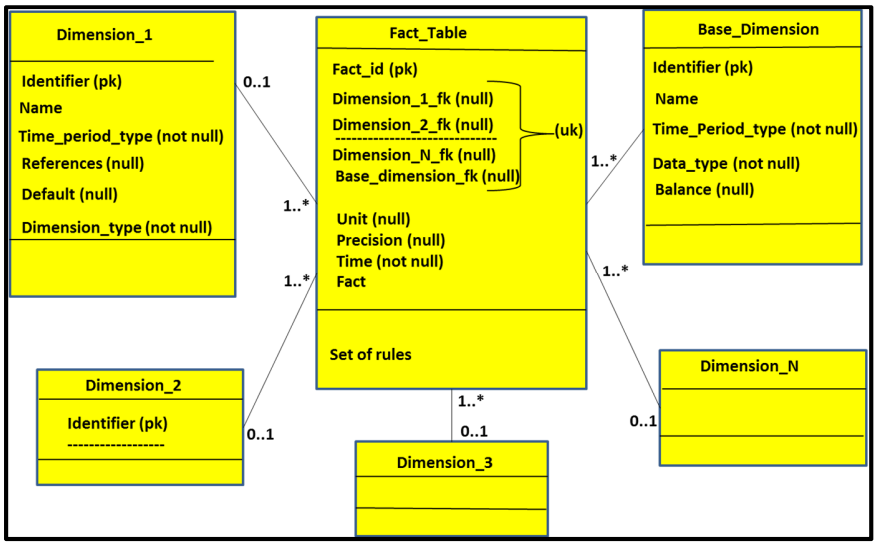


Figura 75. Modelo en estrella. Referencia [9]

La base de datos multidimensional se utiliza principalmente para crear OLAP (proceso analítico en línea) de las aplicaciones y sus bases de datos las cuales utilizan una tabla de hechos y un conjunto de dimensiones, es decir, cubos. Un hecho es una intersección que consta de elementos que forman la dimensión (s) que a su vez forman un cubo. Un hecho puede tener cero o más medidas.

El modelo de datos multidimensional (MDM) se utiliza en lugar del Modelo Relacional, porque la arquitectura europea de los informes económico-financieros se basa en gran medida de las dimensiones, lo que hace que la implementación de MDM sea la opción lógica. Por otra parte, el rendimiento de consultas es mejor en este tipo de base de datos.

El objetivo de este documento es la de almacenar el punto de modelo de datos en una base de datos, de una manera eficiente y fácil.

6.2 Especificaciones del Modelo En estrella

El XBRLDM Dimensión Taxonomía (XDT) define dos tipos de dimensiones .Las dimensiones pueden ser explícitas e implícitas (de acuerdo en XBRL Data Model - XBRLDM) referencia [17].

Los Atributos de dimensiones explícitas se definen de manera explícita en el modelo de metadatos. Una dimensión se define como implícita cuando sus atributos de dimensión no se definen explícitamente en el modelo de metadatos, sin embargo, pertenecen a un dominio particular. Si una dimensión está implícita, no hay posibilidad de establecer jerarquías.

Esto se puede encuentra definido formalmente: Tesis de I.Santos

La Figura 75 muestra el modelo estrella de partida pero nuestro MDM. Posee las siguientes Especificaciones y Normalizaciones de tablas.

Existen 3 tablas Maestras asignadas al Esquema MASTER. Con las siguientes atributos de tabla y Clave PK ID de tipo auto numérica.

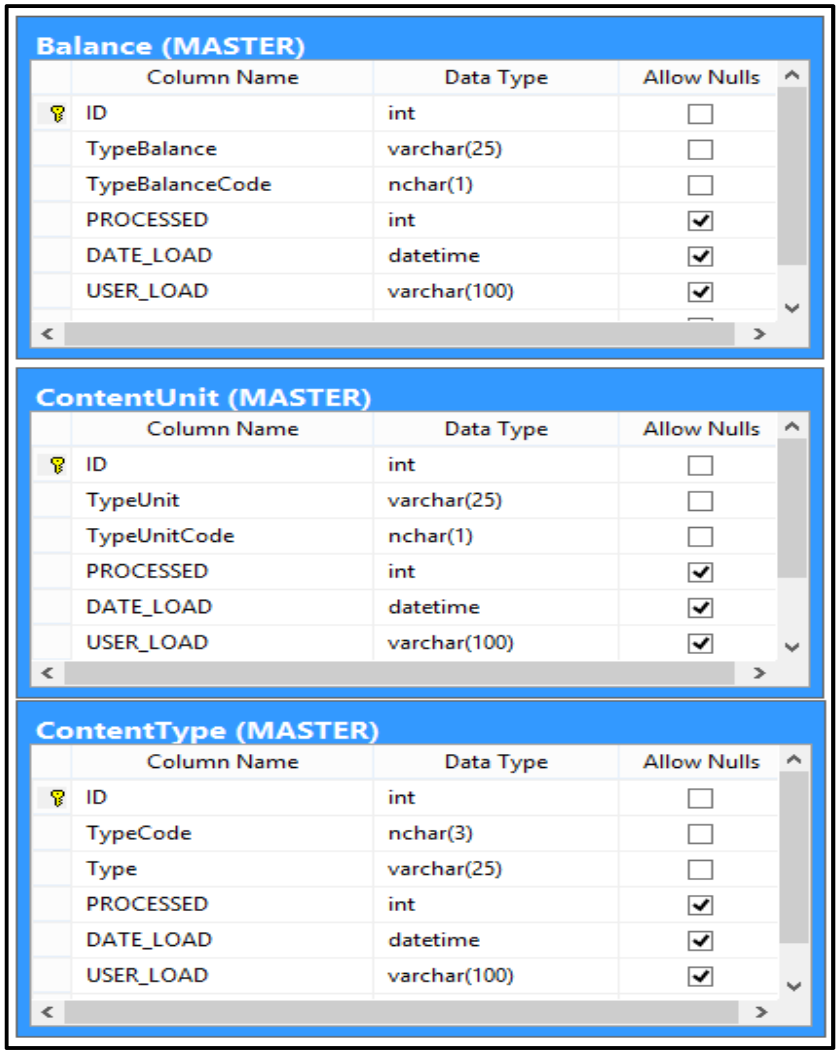


Figura 76. Tablas Maestras. Tipo datos, Unidad y Balance.

Los datos son Insertados desde unas plantillas de datos Excel Mediante SSIS, con proceso homologo al Explicado en el apartado 6 sobre la validación de los datos del modelo. Esta Solución está Incluida en el apartado de Anexos.

Los datos que se integran de forma automática en las tablas son los siguientes:

ContentUnit		ContentType		Balance	
TypeUnit	TypeUnitCode	TypeCode	Type	TypeBalance	TypeBalanceCode
Euro	€	bln	Boolean	Credit	+
Dollar	$	dat	Date	Debit	-

Not Value	–	int	Integer	N/A	|
		mon	Monetary		
		per	Percentage		
		cod	Code		
		str	String		
		dec	Decimal		

Tabla 32. Definición Datos Maestros de la tabla de Hechos.

ContentUnit: Hace referencia a la unidades monetaria del valor del Hecho en el caso de que aplique

ContentType: tipo de datos del Hecho

Balance: Balance del Hecho en el caso de que aplique

Existe n tablas de dimensión asignadas al Esquema DIM. Cada tabla dimensión tiene los siguientes atributos:

Dimension365 (DIM)

Column Name	Data Type	Allow Nulls
Dimension365ID	int	☐
DomainID	int	☑
DimensionCode	nvarchar(MAX)	☑
DimensionLabel	nvarchar(MAX)	☑
DimensionDescription	nvarchar(MAX)	☑
IsTyped	bit	☐
IsImpliedIfNotExplicitly...	bit	☐
DimensionXbrlCode	nvarchar(255)	☑
ConceptID	int	☑
		☐

Figura 77. Ejemplo de tabla Dimension generada de forma dinámica

Dichos atributos son los mismos que los definidos en el Modelo Relacional del apartado 5, (ID de dimensión como PK y FK a la tabla Concept y FK a la tabla *Domain*) salvo que en este caso cada Dimensión estará compuesta por un único registro que será referenciado a

la tabla de Hechos y habrá tantas tablas de dimensiones como registros en la tabla Dimensiones del modelo relacional.

En la creación del MDM veremos en mayor profundidad la singularidad del proceso.

Existe 1 tabla de hechos asignadas al Esquema *FACT*. La tabla de Hechos tiene los siguientes atributos:

Identificador (clave principal, pk), la referencia a la dimensión (clave externa, FK), unidad Monetaria (euros, libras, dólares,...), de tipo, de balance, de precisión y valor del hecho. La unidad Monetaria, el balance y la precisión pueden ser N/A o NULL, porque el hecho puede ser no numérico.

FactTable (FACT) *

Column Name	Data Type	Allow Nulls
IdFactTable	bigint	☐
FactValue	varchar(255)	☑
ContentType	int	☐
ContentUnit	int	☑
Precision	float	☑
Balance	int	☐
Dimension100ID	int	☑
Dimension105ID	int	☑
Dimension110ID	int	☑
Dimension115ID	int	☑
Dimension120ID	int	☑
Dimension125ID	int	☑
Dimension130ID	int	☑
Dimension135ID	int	☑
Dimension140ID	int	☑
Dimension145ID	int	☑
Dimension150ID	int	☑
Dimension155ID	int	☑
Dimension160ID	int	☑
Dimension165ID	int	☑
Dimension170ID	int	☑
Dimension180ID	int	☑
Dimension205ID	int	☑
Dimension225ID	int	☑
Dimension230ID	int	☑

Figura 78. Ejemplo de tabla de Hechos generada de forma dinámica, con todas sus referencias.

La clave externa que referencia a la dimensión puede ser NULL debido a que no todos los Hechos poseerán referencia a todas y cada una de las Dimensiones.

6.3 Creación del Modelo en Estrella.

Los términos utilizados directa o indirectamente en el mapeo del DPM en nuestro MDM son:

Un hipercubo es una lista ordenada de dimensiones, definido por el conjunto de cero o más declaraciones de dimensiones vinculadas al hipercubo por relaciones hipercubo-dimensión en un conjunto de relaciones dimensionales

Un hipercubo se refleja en el cubo de datos. Un cubo de datos es un conjunto de Hechos con sus dimensiones y miembros apropiados.

Un hipercubo en el MDM es un conjunto de pares <dimensión, atributos de dimensión> y atributos calculados definidos para uno o más hechos.

En el MDM, el contexto se define como un conjunto de dimensión de un hecho o grupo de hechos. Un contexto pertenece a una entidad o institución financiera, por un período, un significado para el negocio (segmento), y un escenario. El escenario muestra los pares específicos de dimensión y el atributo de dimensión de la lógica de negocio

El constructor de la tabla de hechos de la figura es equivalente al conjunto de puntos de datos en el DPM. Es un modelo de la estrella porque posee un conjunto de elementos primarios, que están relacionados en la tabla de hechos en n dimensiones.

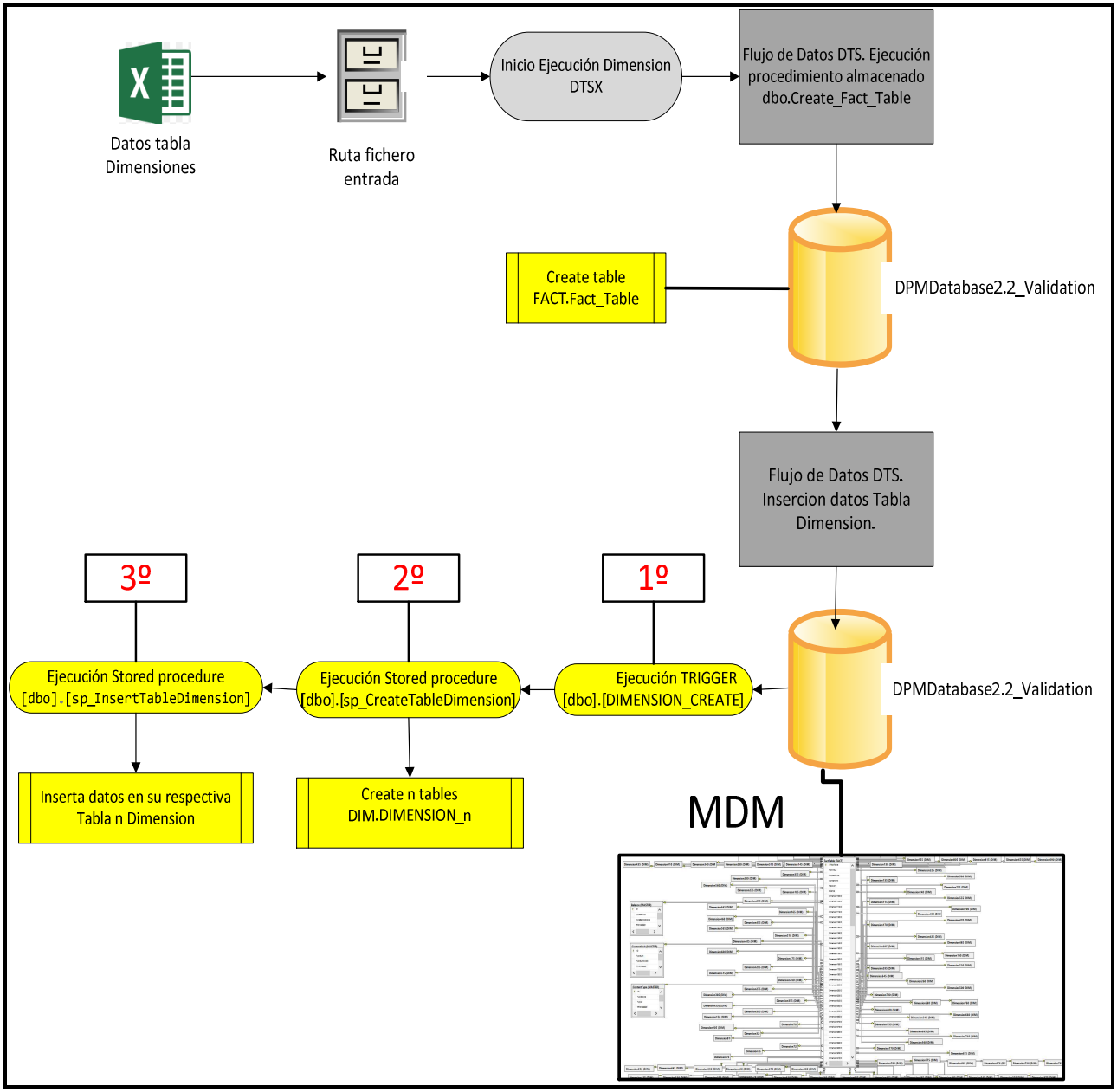


Figura 79.Pasos para la generación del Modelo en estrella propuesto en el artículo

El inicio de la creación del MDM comienza al ejecutar el IS_DIMENSION.DTSX, dentro del Dts hay programado Un script de SQL que ejecuta un procedimiento almacenado alojado

en la BDD de validación de datos, el cual crea la tabla de hechos con las referencias a las tablas Maestras.

Una vez creada comienza la inserción y validación de los datos. En la tabla *Dimension* hay programado un *trigger* que se ejecuta por cada registro llamado *DIMENSION_CREATE* que recoge los datos a insertar por cada registro y ejecuta de forma secuencial dos procedimientos, *sp_CreateTableDimension* que crea una nueva tabla de dimensiones con los datos de 1 registro y su referencia a la tabla de Hechos y *sp_InsertTableDimension* que inserta los datos en dicha tabla y los valida según el modelo definido para el MDM.

6.3.1 Creación de la Tabla de hechos.

El proceso de creación del MDM se realiza de forma automática desde el proceso de validación de la tabla dimensiones en el Modelo Relacional.

De esta forma está sujeto al control de errores. Explicado en el apartado 5.3.4.

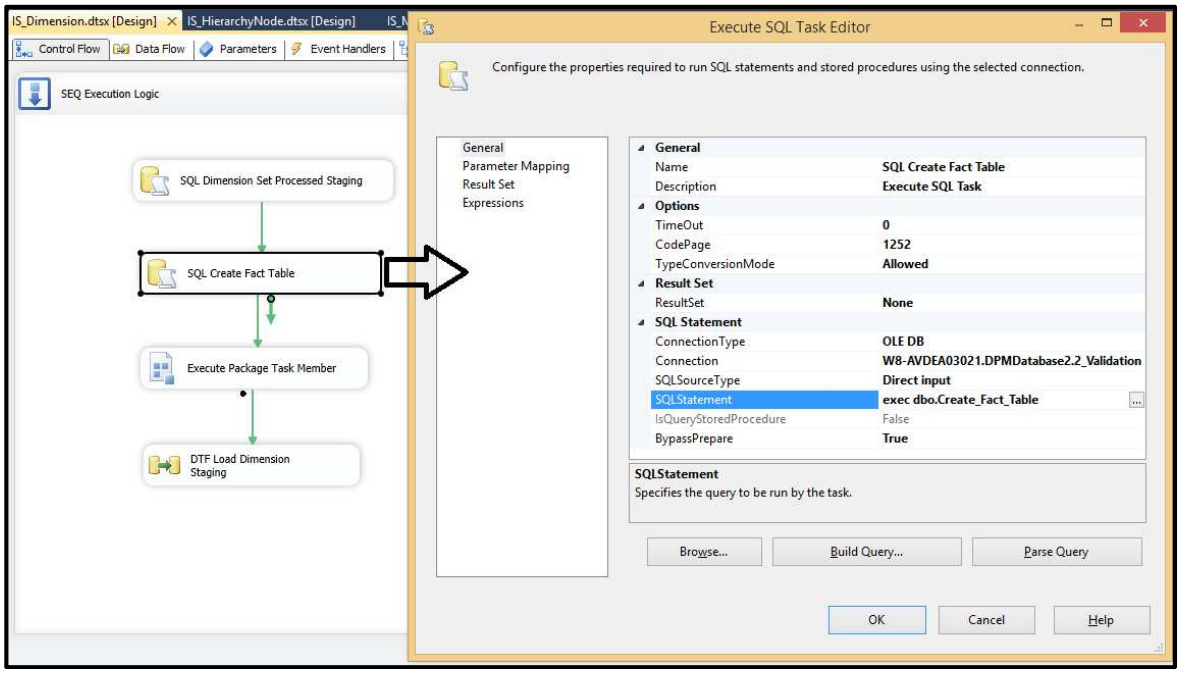


Figura 80. Flujo de datos IS_DIMENSION. Ejecución procedimiento SQL de creación de la tabla de Hechos.

En este punto se ejecuta el siguiente Script:

```
USE [DPMDatabase2.2_Validation]
GO

/****** Object:  StoredProcedure [dbo].[Create_Fact_Table]    Script Date: 21/09/2015
18:53:42 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

-- =============================================
-- Author:          <Abel Nieto Cano>
-- Create date: 01-05-2015
-- Description:     Create Fact Table
-- =============================================
CREATE PROCEDURE [dbo].[Create_Fact_Table]


AS
BEGIN

/****** Object:  Table [FACT].[FactTable]    Script Date: 23/05/2015 14:05:50 ******/
DROP TABLE [FACT].[FactTable]


CREATE TABLE [FACT].[FactTable](
       [IdFactTable] [bigint] IDENTITY(1,1) NOT NULL,
       [FactValue] [varchar](255) NULL,
       [ContentType] [int] NOT NULL,
       [ContentUnit] [int] NOT NULL,
       [Precision] [float] NULL,
       [Balance] [int] NOT NULL
 CONSTRAINT [PK_FactTable] PRIMARY KEY CLUSTERED
(
       [IdFactTable] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]


ALTER TABLE [FACT].[FactTable]  WITH CHECK ADD  CONSTRAINT [FK_FactTable_Balance]
FOREIGN KEY([Balance])
```

```
REFERENCES [MASTER].[Balance] ([ID])


ALTER TABLE [FACT].[FactTable] CHECK CONSTRAINT [FK_FactTable_Balance]


ALTER TABLE [FACT].[FactTable]  WITH CHECK ADD  CONSTRAINT [FK_FactTable_ContentType]
FOREIGN KEY([ContentType])
REFERENCES [MASTER].[ContentType] ([ID])


ALTER TABLE [FACT].[FactTable] CHECK CONSTRAINT [FK_FactTable_ContentType]


ALTER TABLE [FACT].[FactTable]  WITH CHECK ADD  CONSTRAINT [FK_FactTable_ContentUnit]
FOREIGN KEY([ContentUnit])
REFERENCES [MASTER].[ContentUnit] ([ID])

ALTER TABLE [FACT].[FactTable] CHECK CONSTRAINT [FK_FactTable_ContentUnit]
```

En este punto se crean las referencias FK a cada tabla Maestra y la definición de la tabla de Hechos.

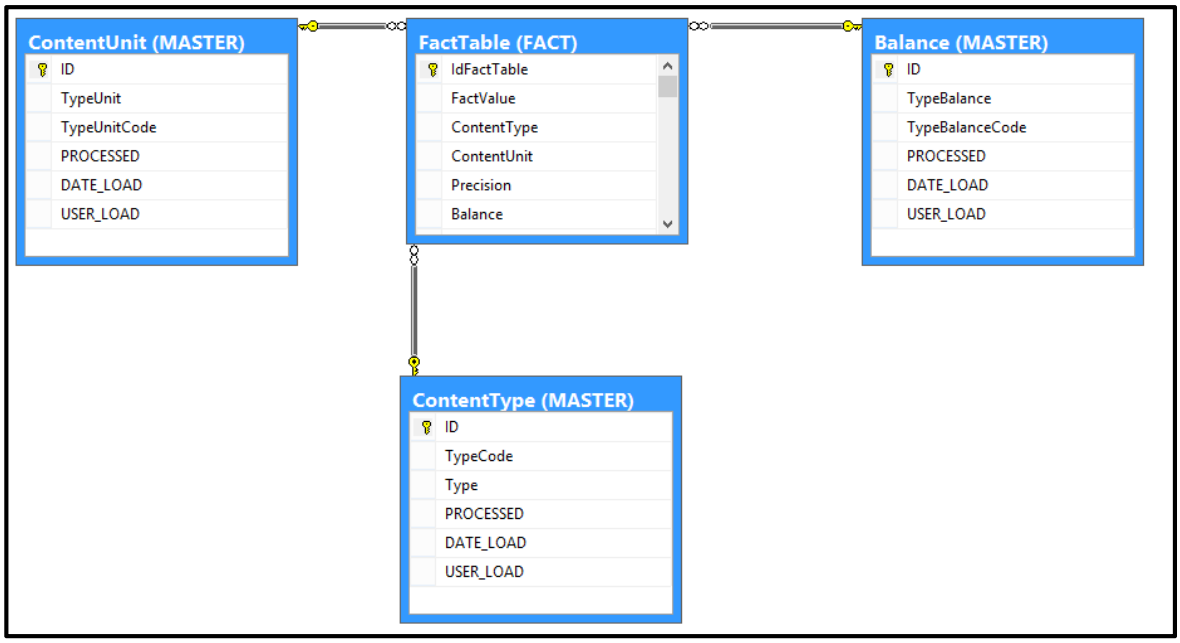


Figura 81. Diagrama de BDD Tabla de hechos y referencia a las tabla Maestras.

El MDM posee en este punto la estructura que se muestra en la figura 81.

6.3.2 Creación de las tablas Dimensiones.

La creación de las tablas de Dimensiones y del MDM, se realiza de forma dinámica mediante el siguiente *Trigger* que se ejecuta por cada registro Insertado en la tabla dimensión del modelo relacional.

```
USE [DPMDatabase2.2_Validation]
GO

/****** Object:  Trigger [dbo].[DIMENSION_CREATE]    Script Date: 21/09/2015 18:55:59
******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO



CREATE TRIGGER [dbo].[DIMENSION_CREATE]
ON [dbo].[Dimension]


AFTER INSERT AS

DECLARE @SqlInsert NVARCHAR(MAX)
DECLARE @CUENTA AS INT
DECLARE @DimensionID AS                     int
DECLARE @DomainID AS                         int
DECLARE @DimensionCode AS                   nvarchar(max)
DECLARE @DimensionLabel AS                  nvarchar(max)
DECLARE @DimensionDescription AS            nvarchar(max)
DECLARE @IsTyped AS                          bit
DECLARE @IsImpliedIfNotExplicitlyModelled AS bit
DECLARE @DimensionXbrlCode AS               nvarchar(255)
DECLARE @ConceptID AS                        int
DECLARE  @Dimid AS VARCHAR(6)


SET @CUENTA = 0

SELECT     @CUENTA = COUNT(*)
FROM  inserted
```

```
SELECT
 @DimensionID = cast(INSERTED.DimensionID as int)
,@DomainID = cast(INSERTED.DomainID  as int)
,@DimensionCode = INSERTED.[DimensionCode]
,@DimensionLabel = INSERTED.[DimensionLabel]
,@DimensionDescription = INSERTED.[DimensionDescription]
,@IsTyped = INSERTED.[IsTyped]
,@IsImpliedIfNotExplicitlyModelled = INSERTED.[IsImpliedIfNotExplicitlyModelled]
,@DimensionXbrlCode = INSERTED.[DimensionXbrlCode]
,@ConceptID = cast(INSERTED.[ConceptID]  as int)
  FROM inserted


IF @CUENTA <> 0

BEGIN

EXEC   [dbo].[sp_CreateTableDimension] @DimensionID = @DimensionID

EXEC   [dbo].[sp_InsertTableDimension] @DimensionID = @DimensionID, @DomainID =
@DomainID, @DimensionCode = @DimensionCode,
@DimensionLabel = @DimensionLabel, @DimensionDescription = @DimensionDescription,
@IsTyped = @IsTyped, @IsImpliedIfNotExplicitlyModelled =
@IsImpliedIfNotExplicitlyModelled,
@DimensionXbrlCode = @DimensionXbrlCode, @ConceptID = @ConceptID


END
GO
```

Este *trigger* se encarga de recoger la Información de la que se compondrá el MDM y lo realiza en dos pasos.

Primero, crea de forma dinámica la tabla de Dimensiones por cada Registro, dicha tabla tendrá su propia PK y su referencia a la tabla de hechos.

```
USE [DPMDatabase2.2_Validation]
GO

/****** Object:  StoredProcedure [dbo].[sp_CreateTableDimension]    Script Date:
21/09/2015 18:59:13 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
```

```
GO

-- =============================================
-- Author:          Abel Nieto Cano
-- Create date: 23/05/2015
-- Description:     Creación de Tablas Dimension
-- =============================================
CREATE PROCEDURE [dbo].[sp_CreateTableDimension]
       @DimensionID [int]
WITH EXECUTE AS CALLER
AS
BEGIN

SET NOCOUNT ON;

DECLARE @sqlcreate NVARCHAR(MAX)
DECLARE @CUENTA AS INT
DECLARE @DimID AS  varchar(30)

SET @Dimid = cast (@DimensionID as varchar)
SET @CUENTA = 0

SELECT @CUENTA = count(*) FROM INFORMATION_SCHEMA.TABLES WHERE
substring(TABLE_NAME,10,7) in (@Dimid) and
TABLE_SCHEMA = 'DIM'

IF @CUENTA = 0

BEGIN


SET @sqlcreate = '

CREATE TABLE [DIM].[Dimension'+@Dimid+'](
       [Dimension'+@Dimid+'ID] [int] NOT NULL,
       [DomainID] [int] NULL,
       [DimensionCode] [nvarchar](max) NULL,
       [DimensionLabel] [nvarchar](max) NULL,
       [DimensionDescription] [nvarchar](max) NULL,
       [IsTyped] [bit] NOT NULL,
       [IsImpliedIfNotExplicitlyModelled] [bit] NOT NULL,
       [DimensionXbrlCode] [nvarchar](255) NULL,
       [ConceptID] [int] NULL,
 CONSTRAINT [PK_Dimension'+@Dimid+'] PRIMARY KEY CLUSTERED
(
       [Dimension'+@Dimid+'ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]


ALTER TABLE FACT.FactTable
ADD Dimension'+@Dimid+'ID int NULL
```

```
ALTER TABLE [FACT].[FactTable]  WITH CHECK ADD  CONSTRAINT
[FK_FactTable_Dimension'+@Dimid+'] FOREIGN KEY([Dimension'+@Dimid+'ID])
REFERENCES [DIM].[Dimension'+@Dimid+'] ([Dimension'+@Dimid+'ID])


ALTER TABLE [FACT].[FactTable] CHECK CONSTRAINT [FK_FactTable_Dimension'+@Dimid+']'

 EXECUTE (@sqlcreate);

END

END

GO
```

En este punto se va creando esta estructura por cada Registro de dimensión que se integra de forma dinámica.

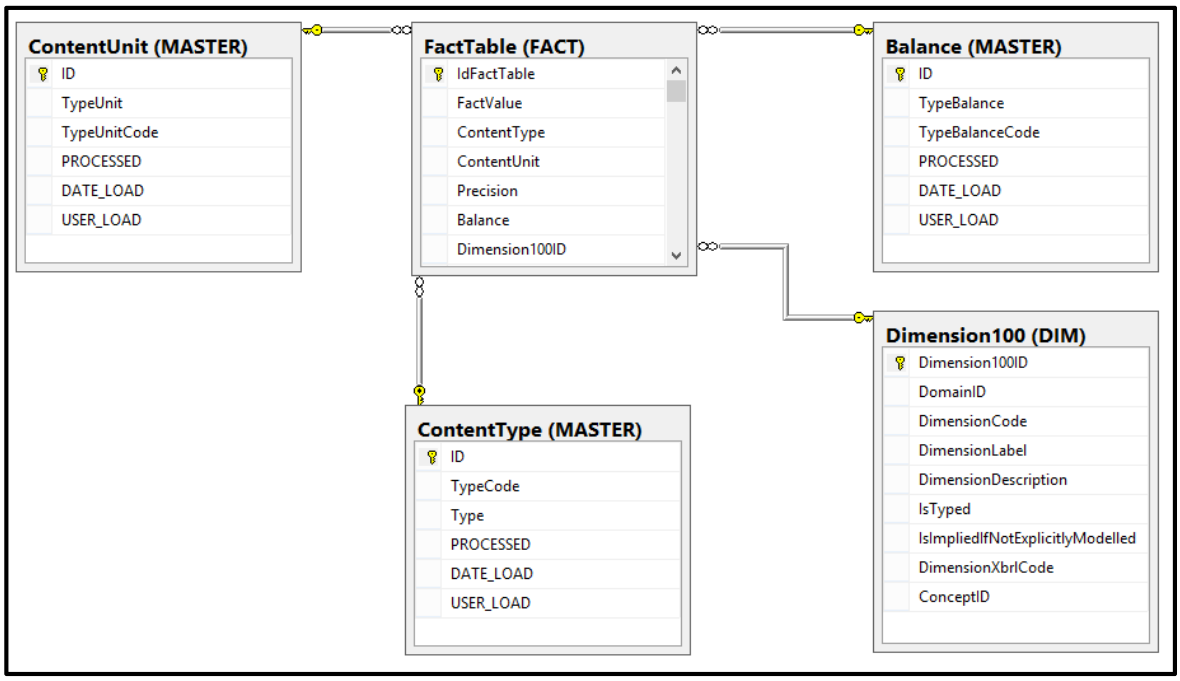


Figura 82. Diagrama de BDD Tabla de hechos y referencia a las tabla Maestras de dimensión.

El segundo paso es insertar los datos en la tabla creada.

```
SET QUOTED_IDENTIFIER ON
GO

-- ============================================
-- Author:          Abel Nieto Cano
-- Create date: 23/05/2015
-- Description:     Creación de Tablas Dimension
-- ============================================
CREATE PROCEDURE [dbo].[sp_InsertTableDimension]
        @DimensionID [int] ,
        @DomainID [int] ,
        @DimensionCode nvarchar(255),
        @DimensionLabel  nvarchar(255),
        @DimensionDescription nvarchar(255),
        @IsTyped                          bit    ,
        @IsImpliedIfNotExplicitlyModelled  bit,
        @DimensionXbrlCode                nvarchar(255),
        @ConceptID                         [int]

WITH EXECUTE AS CALLER
AS
BEGIN

SET NOCOUNT ON;


DECLARE @sqlinsert NVARCHAR(MAX)
DECLARE @CUENTA AS INT
DECLARE @DimID AS  varchar(30)
DECLARE @DomID AS  varchar(30)
DECLARE @ConID AS  varchar(30)
DECLARE @IsTID AS  varchar(1)
DECLARE @IsID AS  varchar(1)
DECLARE @TABLECHECK AS VARCHAR (50)
DECLARE @TABLA AS  varchar(100)
DECLARE @t TABLE ( resultado VARCHAR(MAX) )

SET @Dimid = @DimensionID
SET @DomID = @DomainID
SET @ConID = @ConceptID
SET @IsTID = @IsTyped
SET @IsID  = @IsImpliedIfNotExplicitlyModelled
SET @CUENTA = 0
SET @TABLECHECK =  'DIM.Dimension'+@Dimid
SET @TABLA = 'SELECT count(*) FROM '+@TABLECHECK
SET @DomID = @DomainID


insert into @t EXEC(@TABLA)
```

```
SELECT @CUENTA=resultado FROM @t

SELECT @CUENTA

IF @CUENTA = 0

BEGIN

SET @sqlinsert = '

INSERT INTO [DIM].[Dimension'+@Dimid+']
           ([Dimension'+@Dimid+'ID]
           ,[DomainID]
           ,[DimensionCode]
           ,[DimensionLabel]
           ,[DimensionDescription]
           ,[IsTyped]
           ,[IsImpliedIfNotExplicitlyModelled]
           ,[DimensionXbrlCode]
           ,[ConceptID])
     VALUES
           ('+@Dimid+'
           ,'+@DomID+'
           ,'+char(39)+@DimensionCode+char(39)+'
           ,'+char(39)+@DimensionLabel+char(39)+'
           ,'+char(39)+@DimensionDescription+char(39)+'
           ,'+@IsTID+'
                ,'+@IsID+'
           ,'+char(39)+@DimensionXbrlCode+char(39)+'
           ,'+@ConID+');'

 EXECUTE (@sqlinsert);

END

END

GO
```

El proceso se repite tantas veces como dimensiones posea la tabla de hechos, en este caso son 815 veces que coincide con el número de tablas de dimensión creadas y referenciadas en la tabla de hechos. Una vez finalizado el proceso el modelo en estrella queda validado y creado tal y como se muestra en la figura 83.

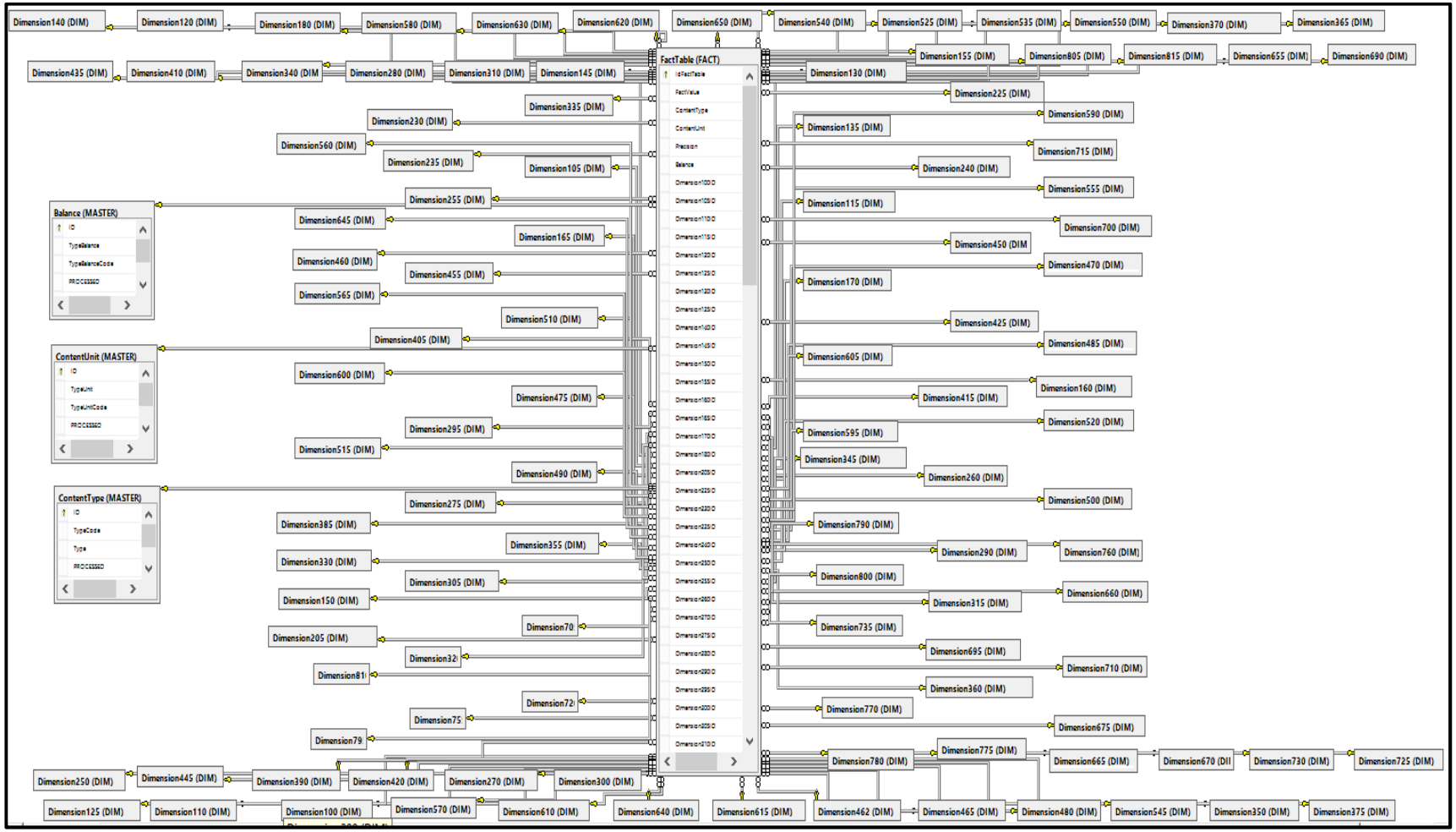


Figura 83. Diagrama de BDD del MDM.

6.3.3 Explotación de los Datos del MDM.

En este punto hay varias herramientas para tratar y explotar esta información para que los usuarios finales puedan verlos en Informes.

Cualquier herramienta desde la cual se pueda realizar una conexión a Base de datos SQL Server.

En este Proyecto hemos tratado el MDM con la Aplicación Analysis Service. La razón más inmediata es por compatibilidad, porque pertenece a la familia de productos Microsoft y además se trata de una herramienta integrada en MS SQL SERVER 2012. Con lo que la instalación se Puede realizar de forma directa.

Analysis Services es un motor de datos analíticos en línea que se usa en soluciones de ayuda a la toma de decisiones y *Business Intelligence* (BI), y proporciona los datos analíticos para informes empresariales. Un flujo de trabajo típico para *Analysis Services* incluye la creación de un modelo de datos OLAP o tabular, la implementación del modelo como base de datos en una instancia de *Analysis Services*, el procesamiento de la base de datos para cargarla con datos y, a continuación, la asignación de permisos para permitir el acceso a datos. Cuando esté listo, se puede obtener acceso a este modelo de datos con varios fines desde cualquier aplicación cliente que admita *Analysis Services* como origen de datos.

Para crear un modelo, se usa *SQL Server Data Tools* (vea Anexos y aplicaciones utilizadas en *Analysis Services*) y elija la plantilla de proyecto Multidimensional y Minería de datos que hemos se ha realizado para ver el MDM.

La plantilla de proyecto contiene las carpetas de todos los objetos necesarios en un modelo. Puede utilizar asistentes para crear todos los elementos básicos, como orígenes de datos, vistas de origen de datos, dimensiones, cubos y roles.

El modelo se rellena con datos procedentes de sistemas de datos externos, en este caso de datos hospedados en el motor de base de datos relacional de SQL Server, pero puede ser de Oracle.

Los modelos especifican objetos de consulta, como los cubos, pero también especifican las dimensiones que se pueden usar en varios cubos, cálculos y KPI que encapsulan la lógica del negocio, así como interacciones, como los comportamientos en navegación y obtención de detalles.

Puede instalar una instancia de Analysis Services en uno de estos tres modos de servidor:

- *Como instancia tabular, ejecutando modelos tabulares.*
- *Como una instancia multidimensional y de minería de datos, ejecutando cubos OLAP y modelos de minería de datos (es el valor predeterminado).*
- *Como PowerPivot para SharePoint, ejecutando modelos de datos PowerPivot y de Excel en SharePoint (PowerPivot para SharePoint es un motor de datos de nivel intermedio que carga, consulta y actualiza modelos de datos hospedados en SharePoint).*

El mismo motor de datos; tres formas de usarlo. Hay que tener en cuenta que los modos de servidor se establecen durante la instalación y no se pueden cambiar posteriormente. Debe instalar una nueva instancia si necesita otro modo diferente.

(Referencia [16])

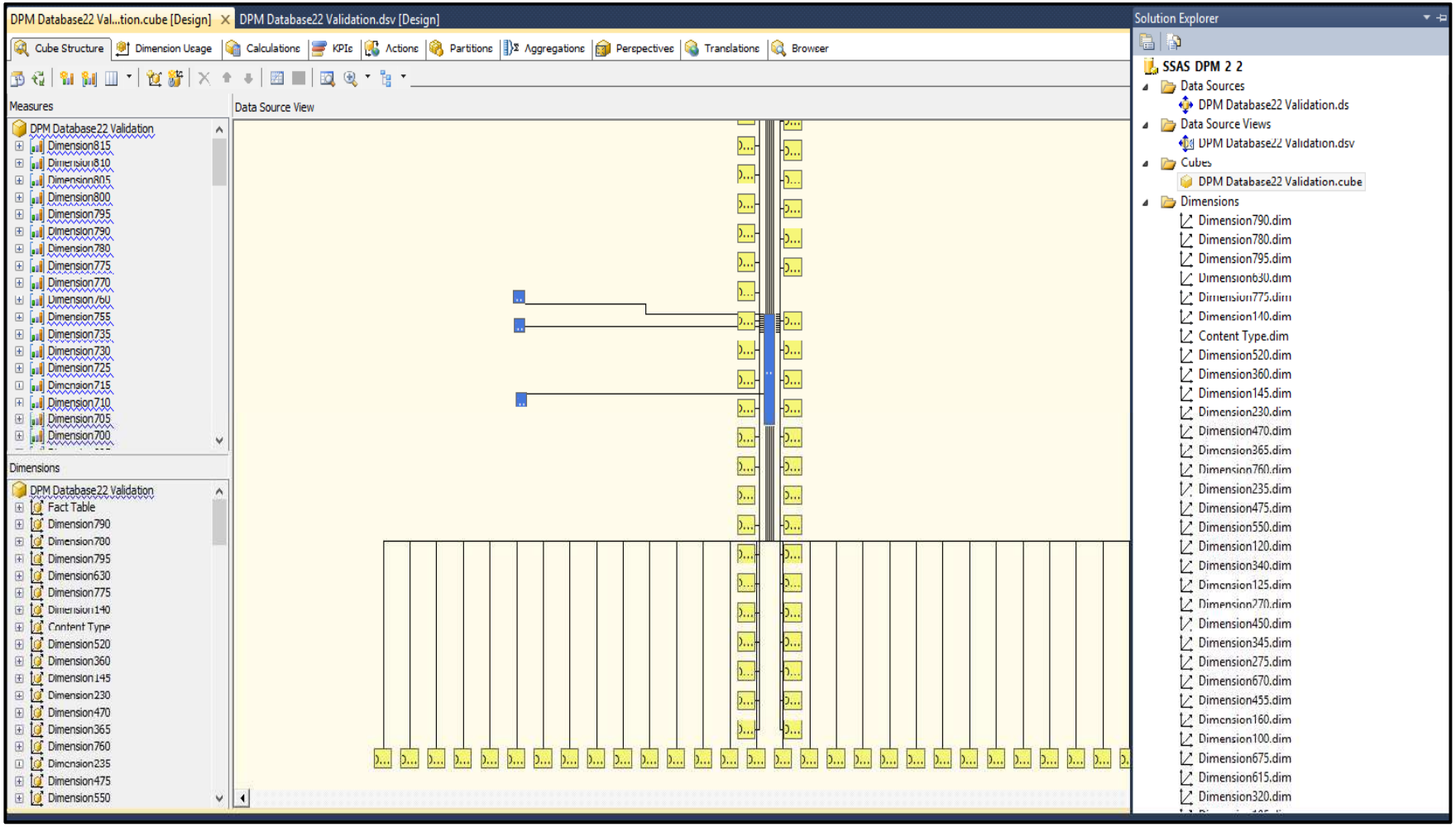


Figura 84. Diagrama de BDD del MDM en la solución de Analysis Service.

7 Futuras mejoras. Conclusiones.

Los proyectos de Ingeniería del SW en el que se están Implicados usuarios de negocio nunca están carentes de mejoras o cambios debido a lo cambiante de las estructuras y áreas en las que se desempeña su labor. Pero desde un punto de vista Técnico, hemos conseguido recopilar una serie de futuros desarrollos orientados a mejorar más el entorno planteado en este Proyecto.

El diseño lógico de la base de datos, que incluye las tablas y sus relaciones, es la clave de una base de datos relacional optimizada. Un buen diseño lógico de la base de datos puede ser la base de un rendimiento óptimo de la aplicación y de la base de datos. Un diseño lógico deficiente puede comprometer el rendimiento de todo el sistema.

La normalización de un diseño lógico de la base de datos implica la utilización de métodos formales para separar los datos en varias tablas relacionadas. Una característica de una base de datos normalizada es la existencia de varias tablas pequeñas con menos columnas. En las bases de datos no normalizadas, existen menos tablas más amplias con más columnas.

Por lo general, una normalización razonable permite mejorar el rendimiento. Cuando se dispone de índices útiles, el optimizador de consultas de SQL Server es una herramienta adecuada para la selección rápida y eficaz de combinaciones entre tablas.

La normalización ofrece diversas ventajas, entre las que se incluyen:

- Mayor rapidez en la ordenación y en la creación de índices.
- Un número mayor de índices clúster.
- Índices más estrechos y compactos.
- Menor número de índices por tabla. De esta forma, se mejora el rendimiento de las instrucciones INSERT, UPDATE y DELETE.
- Menor número de valores NULL y menos oportunidades para generar incoherencias. De esta forma, aumenta el rendimiento.

Aplicando una restructuración de bases de datos y normalización de las bases de datos del modelo actual, evitando la duplicación sistemática de los datos, se podría estimar un ahorro alrededor de un 50% del almacenamiento estimado, lo cual suponen un gran ahorro de costes en el almacenamiento

Como los futuros trabajos esta ampliar el espectro de este esquema a todo el modelo relacional propuesto por el EBA.

Otra mejora tiene que ver con la forma de integrar y acceder a los datos. A continuación mostramos un diagrama del flujo de datos recomendado.

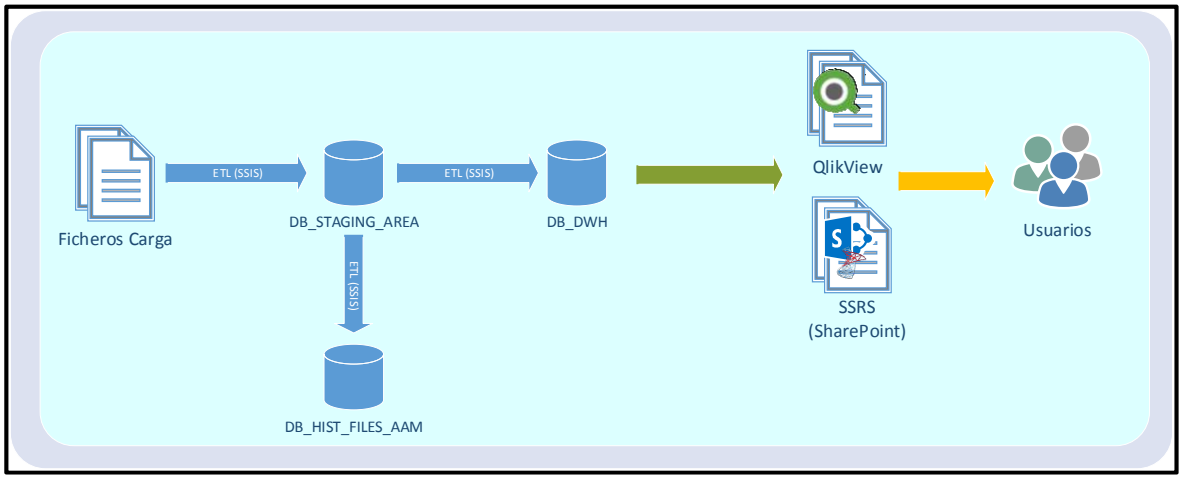


Figura 85. Diagrama de BDD Tabla de hechos y referencia a las tabla Maestras de dimensión.

Dicho proyecto se trata de un estudio en el que se profundiza en buenas prácticas, lo cual me ha servido para un mejor desarrollo técnico y académico, Por lo que el número de horas empleadas en el estudio han sido muy amplias pero muy agradecidas a nivel personal. Finalmente el estudio ha sido muy completo y del que me siento orgulloso.

8 Planificación y Seguimiento. Presupuesto

8.1 Introducción

La planificación engloba las 8 secciones del Proyecto, así como los congresos en los que se ha expuesto este proyecto con dos hitos claramente diferenciados. En el congreso de Bruselas en el que se mostraron varias pruebas de concepto sobre la validación del DPM y el congreso en Madrid en el que se profundizó en el MDM.

En el detalle de Planificación, Seguimiento y horas empleadas del Proyecto se detallan los Recursos de Personal.

El inicio del proyecto se sitúa el 19 de Octubre y se contabilizan aproximadamente unas 490 horas.

Planificación Proyecto.

El presupuesto total de este proyecto asciende a la cantidad de 80.000 Euros.

Presupuesto.

Leganés a 14 de Octubre de 2015

El ingeniero proyectista

Fdo. Abel Nieto Cano.

9 Glosario.

ADO	*ActiveX Data Objects*
ANSI	*American National Standards Institute*
BDD	*Base de Datos*
BI	*Bussiness Iteligence*
COREP	*Common Solvency Report*
DB	*DataBase*
DPM	*Data Point Model*
DQS	*Data Quality Service*
DTS/DTSX	*Data Transformation Services*
DWH	*Data WareHouse*
EBA	*European Banking Authority*
ETL	*Extract, Transform and Load*
FIREP	*Financial Report*
FK	*Foreign Key*
FTP	*File Transfer Protocol*
HW	*Hardware*
KPI	key performance indicator
MDM	*Multidimensional Data Model*
MDS	*Master Data Service*
MS	*Microsoft*
MSDN	*Microsoft Developer Network*
OLAP	*On-Line Analytical Processing*
OLE DB	*Object Linking and Embedding for Databases*
PK	*Primary Key*
PLSQL	*Procedural Language/Structured Query Language*
ROLAP	*Relacional On-Line Analytical Processing*
SGBD	*Sistema Gestor Base de Datos*
SP	*Stored Procedure*
SQL	*Structured Query Language*
SSAS/AS	*Sql Server Analisys Service*
SSDT	*SQL Server Data Tools*
SSIS/IS	*Sql Server Integration Service*
SSRS	*Sql Server Reporting Service*
SW	*Software*
XBRL	*Extensible Business Reporting Language*
XBRLDM	*Extensible Business Reporting Language Data Model*
XML	*eXtensible Markup Language*
ADO	*ActiveX Data Objects*

10 Referencias.

[1] COREP/FINREP XBRL Taxonomy v2.0.0. COREP/FINREP XBRL Taxonomy v2.2.0 http://www.eba.europa.eu/regulation-and-policy/supervisory-reporting/implementing-technical-standard-on-supervisory-reporting-data-point-model-

[2] Improving transparency in financial and business reporting — Harmonisation topics — Part 5: Mapping between DPM and MDM. TC XBRL WI XBRL001 CEN/TC XBRL Secretariat: NEN

[3] XBRL21. Extensible Business Reporting Language (XBRL) 2.1. Engel P, Hamscher W., Shuetrim G., Vun Kannon D., Wallis H. July 2nd, 2008. XBRL International. Extensible Business Reporting Language (XBRL) 2.1.

[4] DMMATRIXSCHEMA. Data Model and Matrix Schemas. Schmehl K. November 16th, 2009. XI European Banking Supervisor, XBRL Workshop hosted by the Oesterreichische Nationalbank, Vienna. XI European Banking Supervisors XBRL Workshop

[5] XBRLDIM. XBRL Dimensions 1.0 XBRL International. Hernandez-Ros I, Wallis H. April 26th, 2006. http://www.xbrl.org/Specification/XDT-CR3-2006-04-26.rtf.

[6] EURXBRLTAXONARCHIT. European XBRL Taxonomy Architecture V2.0. Declerck T, Homes R, Heinze K, 2013. CEN Workshop Agreement. European XBRL Taxonomy Architecture V3.0.

[7] ACADEMY. Academy works. 2013. Openfiling. Openfiling Academy. http://www.openfiling.info/academy/

[8] POC1. Proof of concept of mapping a XBRL report versus a RDBMS. Santos I, Castro E. Openfiling 1st General Assembly, organized by XBRL Operational Network of the European Banking Authority, and hosted by Bank of Italy. September 5th, 2011. Banca d'Italia, Rome, Italy. http://www.openfiling.info/?page_id=286.

[9] XBRL Meta-metadata Model. Valencia J. 2011. Final project of Computer Engineering Technology Management, Carlos III University of Madrid. Date September 27th, 2011. Tutors: Santos I, Castro E. XBRL Meta-metadata Model.

[10] BUILDDW. Building the Data Warehouse. Inmon W. H, 4th Edition. John Wiley & Sons 2005.

[11] Data Point Model and Taxonomies for Implementing Technical Standard (ITS) on Supervisory Reporting. Diciemnbre 2013. http://www.eba.europa.eu/regulation-and-policy/supervisory-reporting/implementing-technical-standard-on-supervisory-reporting-data-point-model-

[12] **Congreso XBRL en Bruselas** Eurofiling 2015. Ponentes: Ignacio J. Santos Forner y Abel Nieto Cano. http://www.eurofiling.info/201411/

[13] **Congreso XBRL en Madrid** Eurofiling 2015. Ponentes: Ignacio J. Santos Forner y Abel Nieto Cano. http://eurofiling.info/201506/index.shtml

[14] Artículo publicado en 2015 http://www.computacionynegocios.com/2012/09/semejanzas-y-diferencias-entre-ms.html

[15] Documentación de *MS SQL SERVER 2012* del Sitio Oficial https://msdn.microsoft.com/es-es/library/dd692930(v=sql.10).aspx

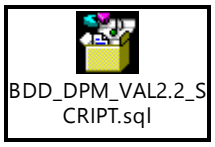
[16] Documentación sobre Analysis service del sitio Oficial https://msdn.microsoft.com/es-es/library/bb522607(v=sql.120).aspx

[17] Hernández-Ros y Wallis, 2006; Schmehl, 2009.

11 Anexos

Control de versiones DPM EBA.

Script BDD Validación DPM 2.2



Estructura de Carpetas Integración y Validación

Solución SSIS de Integración y validación.

Solución SSIS Carga Maestros MDM

Modelo MDM en SSAS