Títol del Treball Fi de Carrera : Creació d'un programari per la transformació de dades

Nom Estudiant : Federico José Alonso Padilla
ETIG

Nom Consultor : Ramón Cahiuelas Quiles

Data Lliurament : 18 de Juny de 2004

DEDICATÒRIA I AGRAÏMENTS

LA dedicatòria ...

Dedico aquest treball a Yolanda, la meva parella, i a Lluís, el seu fill, que m'han donat el suport i l'estimació necessària per portar a terme els estudis i aquest treball fi de carrera.

... i els agraïments ...

Al meu amic Alex, company de feina i d'estudis, el seu ajut durant aquest anys d'estudi sense horaris ni llocs, i, *especialment*, l'haver "parit" en el moment just l'idea de la "Data Transformation Machine – I " que ha donat peu a aquest TFC.

Al "altre" amic, l'Andreu, també company de feina i d'estudis, els seus ànims i el seu punt de vista optimista i *diferent*.

Als "jefes" Luis, Paco i Dani, que per aquest orde i cadascú en el seu moment, m'han donat facilitats i recolzament per tirar endavant en aquest estudis.

I a la tota la família la seva inesgotable comprensió i paciència.

RESUM

Creació d'un programari per la transformació de dades.

L'objectiu d'aquest TFC ha estat la creació d'un programari que permeti transformacions ràpides de gran volums de dades. Cada procés de transformació havia de dependrà d'un conjunt de regles de transformació preestablertes per subconjunts de dades, de forma que el procés fos repetible i verificable i així es pugues garantir la qualitat dels processos de transformació. Les regles de transformació s'havien de poder especificar de forma fàcil i pràctica per usuaris sense coneixements de programació. Les transformacions possibles haurien de cobrir requeriments derivats de la neteja, transformació i formatació de dades, a nivell de taula, registre i camp. Per cada subconjunt de dades i regles de transformació inicials, que hi conformen l'entrada del programari, aquest hauria de retornar el corresponent subconjunt de dades transformades.

Títol en castellà : Creación de un programa para la transformación de datos.
Títol en anglès : Building of software for data transformation.

ÍNDEX DE CONTINGUTS

ÍNDEX DE FIGURES

1 INTRODUCCIÓ

1.1 Justificació del TFC i context en el qual es desenvolupa

Punt de partida

A aquest TFC li donen marc les necessitats sorgides d'un gran multinacional del sector químic, que té un munt de diferents sistemes informàtics per donar suport a les seves activitats de negoci i que pren la decisió estratègica de integrar-los en un únic ERP (de l'anglès *Enterprise Resource Planning)*.

Per portar endavant aquesta integració s'han estat posant en marxa projectes que per països i/o unitats de negocis han anant migrant els processos de negoci des dels antics sistemes al nou sistema integrador.

L'ultimi d'aquest projectes posats en marxa ha estat sense dubte el més ambiciós i complex donat les seves dimensions : afecta a la unitat de negoci més gran de la companyia, en totes les seves seus a Europa. Això es tradueix en un munt de sistemes informàtics diferents: hi participen fins 15 països (Espanya, Portugal, França, Itàlia, Regne Unit, Alemanya, Bèlgica, Holanda, Suècia, Noruega, Àustria, Finlàndia,...) i a més, en alguns països, hi han fins 3 entitats legals independents, cadascuna de les quals, i per cadascú dels països, han anat evolucionant al llarg del temps, adoptant solucions informàtiques de forma independent, de vegades per falta de planificació, de vegades perquè en el moment que implementaven la seva solució informàtica eren empreses independents, que ara, amb motiu de diverses compres i vendes es troben formant una gran unitat de negoci d'una mateixa empresa.

Les dimensions d'aquest projecte han fet inviables les solucions adoptades en els projectes anteriors per portar a bon terme la necessària transformació de dades.

Aquesta es la raó per la qual la empresa va posar en marxa un projecte de

Mineria de Dades sobre la informació continguda en tots aquests sistemes, donant prioritat a la creació d'un magatzem comú de dades per albergar ja transformades les dades dels vells sistemes existents. Es clar que el nou magatzem de dades haurà de tenir com a model el nou sistema ERP. D'aquesta manera s'aprofita la preparació de les tasques de transformació de dades requerides per quan s'hagin d'integrar en el nou sistema.

En la pràctica, s'haurà de extraure informació de cada sistema i després, transformar-la tota per poder-la treballar de forma unificada. El problema que presenta fer aquestes transformació es complex. A manera de exemple, un producte que ara te el codi "1234" a França i "9876" a Itàlia haurà de tenir un mateix codi, possiblement, donat que posarem tots els productes en un únic "sac" el codi passarà a ser "PG1BC", i també hi han qüestions com la jerarquia dels productes: primer s'haurà de definir una nova jerarquia que pugui donar suport a tots els productes de tots el països i de totes els diferents sub-negocis que fins ara podien ser suportats per jerarquies locals e independents, i segon s'hauran de escriure las "regles de transformació" de manera que per cada jerarquia "i/o" producte vell es pugui canviar sistemàticament de la assignació jeràrquica vella a la nova. El mateix raonament, o similar s'haurà d'aplicar també als mestre de clients, a les unitats de mesura dels productes, a les condicions de preus (preus, descomptes, comissions,...), proveïdors, comptes financeres,... etc. De fet la llista complerta d'elements a transformar es realment immensa.

Aquesta transformacions s'hauran de poder fer de forma automatitzada per poder processar la gran quantitat d'informació que s'haurà de extreure de tots els sistemes de forma repetitiva, sistematitzada, verificable, i en el mínim temps possible.

Per començar a fer això hi havia un important requeriment a satisfer : el coneixement de si un producte haurà de pertanyé a una lloc o altre de la nova jerarquia, si els codis de productes hauran canviar capa un format o un altre, i tota la resta de transformacions, està en mans dels actuals usuaris, que es troben distribuïts en un munt de països, amb llengües diferents, sense

coneixements per "traduir" el que saben a un programa, o a una sentencia SQL (*Structured Query Language*), per tal “d’aplicar" les transformacions que saben s’han de fer a les dades d’una forma mínimament repetitiva i automatitzada.

Es important ressaltar la importància del requeriment per el programari de “ transformar les dades d’entrada en el mínim temps possible” : si les transformacions es realitzen correctament però no es fan en un temps “raonable” no es cobriran requeriments. Per aclarir-lo, si per transformar totes les dades que es volen posar en un magatzem comú de dades, per el seu posterior anàlisi, son necessàries 6 setmanes, llavors es clar que es redueix considerablement la utilitat dels possibles resultats que es puguin extreure.

Aportació del TFC

Amb totes aquestes premisses, es va plantejar que la resolució d’aquest problema requeria la creació d'un programari, que a partir del "coneixement" dels usuaris, expressat seguint uns esquemes simples i prefixats en fulls de càlcul, hauria de permetre fer la transformació de totes les dades extretes de cada sistema informàtic actual, per tal de poder-les carregar en una única base de dades comuna, d'una forma automatitzada i repetible, per tal de poder ara utilitzar aquesta base de dades comuna per fer anàlisis de la informació mitjançant la Mineria de Dades, i per posteriorment, una vegada definit el projecte d'implantació d'una aplicació ERP comuna, també pogués fer-se servir per obtenir el conjunt de dades "inicials" que alimentaran el nou sistema en el moment de la seva posta en marxa..

Per desenvolupar aquesta eina es va seleccionar Microsoft Accesss (*en davant Access*) amb Microsoft SQL Server. Per raons de facilitar el desenvolupament l'eina es crearà per treballar tan sols amb Access amb petits volums de dades per desenvolupar i provar el programari, deixant per una segona fase final les petites adaptacions necessàries per tal de que l'eina manegui una gran volum de dades mitjançant el suport de SQL Server.

El següent esquema es una representació general dels requeriments inicials i el

resultat del programari de transformació. Las línies puntejades representen elements que queden fora del abast del programari de transformació desenvolupat.

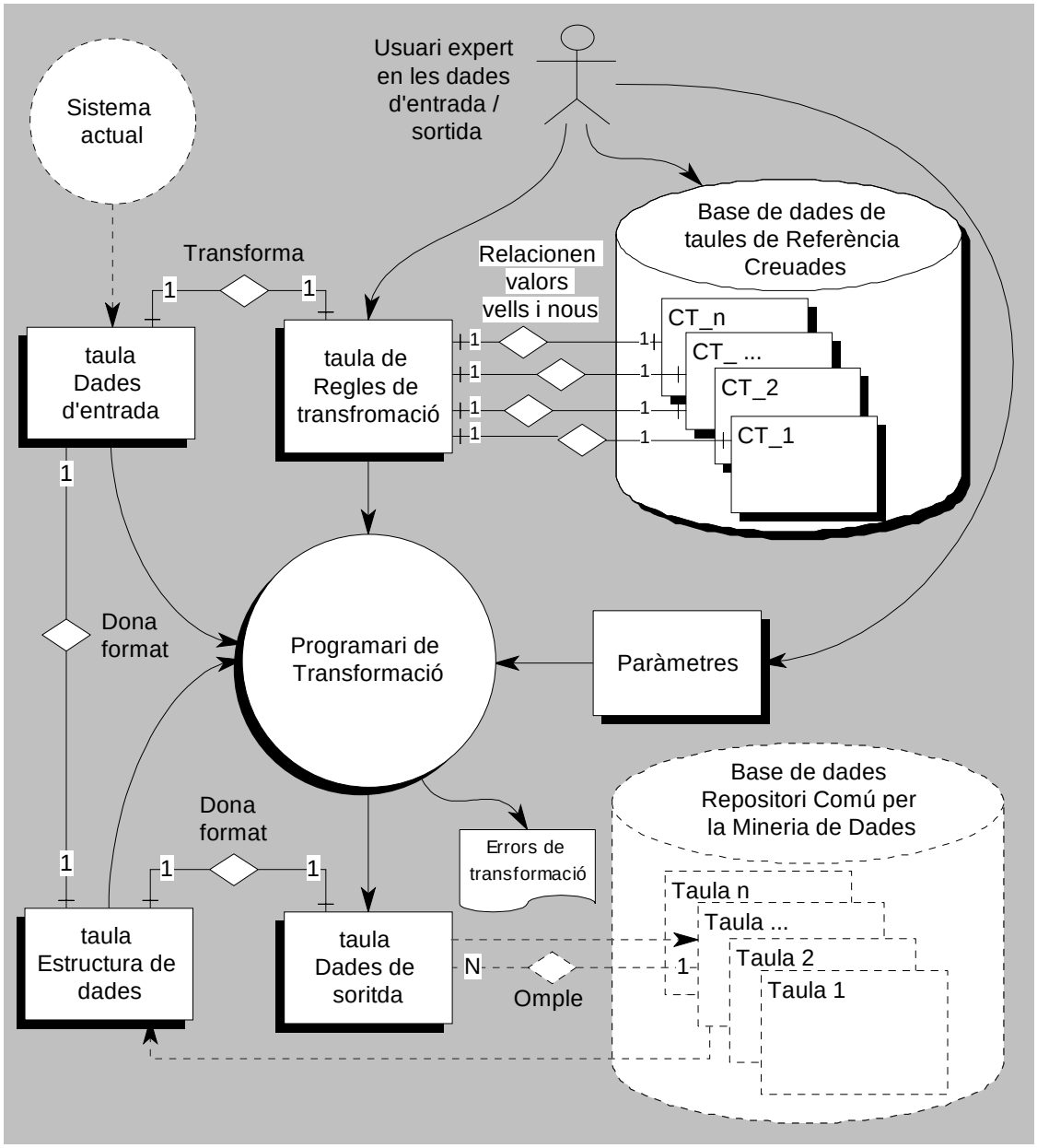


Figura 1-1: Esquema general del programari i dades d'entrada i sortida

1.2 Objectius del TFC.

Els objectius inicials d'aquest projecte van ser :

- Recollir els requisits de preparació de dades que plantegen els propietaris de les dades existents.
- Analitzar el requisits, classificar-los, i descriure a grans trets com es portarà a terme la preparació de les dades. les especificacions del nou programari.
- Definir i documentar com es podran especificar a través de fulls de càlcul simples les transformacions a aplicar a les dades.
- Desenvolupar el programari que de forma eficaç i eficient obtingui a partir d'un conjunt de dades i de transformacions un nou conjunt de dades transformades.

1.3 Enfocament i mètode seguit.

Es va començar amb un anàlisi inicial sense concretar el disseny, donat que els detalls del requeriments inicials no hi eren massa clars, i també per que el projecte cercava aconseguit el recolzament de la direcció del projecte demostrant la viabilitat del programari.

Donada la urgència per l'obtenció de resultats, encara que parcials, el mètode aplicat per desenvolupar el programari ha estat la construcció de successius prototips, minimitzant les etapes d'anàlisi i disseny.

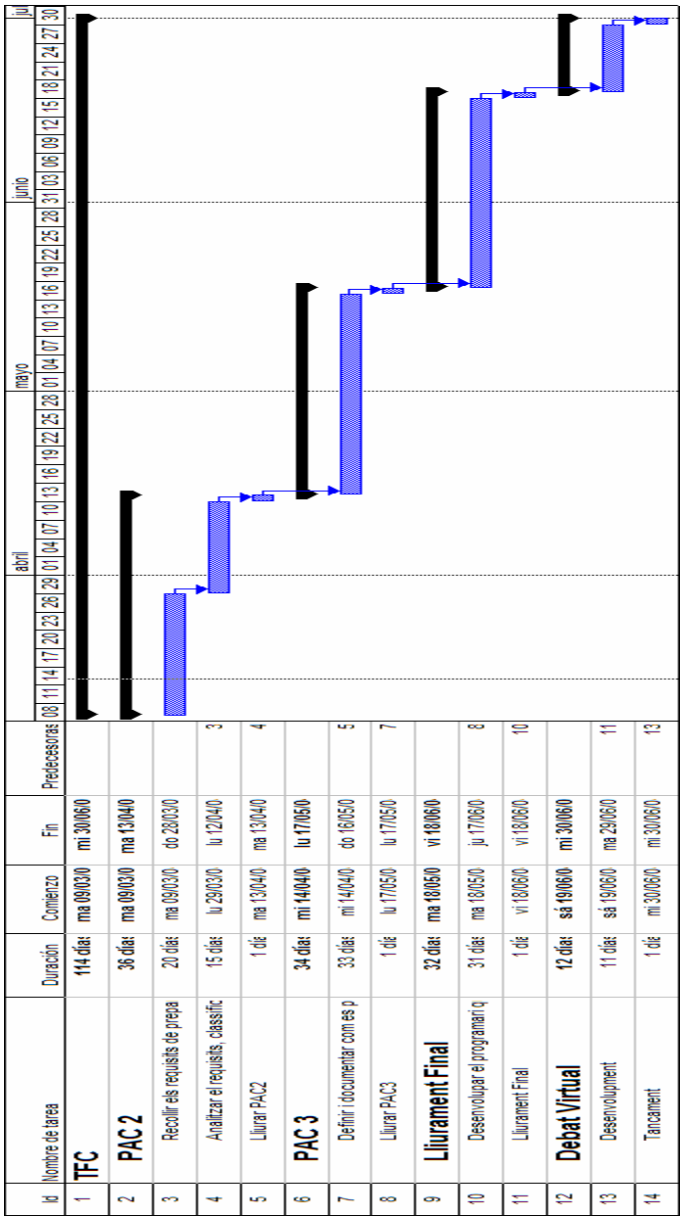
El primer prototip es va aconseguir aplicant un cicle de vida clàssic lleuger : definició i estudi del problema, anàlisis, disseny i implementació. Després es va sotmetre a la opinió dels usuaris, i a partir d'ella es va construir el següent prototip. I així successivament fins aconseguir la versió definitiva.

1.4 Planificació del projecte.

Per fer la temporalització i el posterior seguiment ens recolzem en la eina MS Project 98. La llista de processos es la següent:

id	Procés	Dies	Data inici	Data final	Depen-dències
1	TFC	114	9 març	30 juny	
2	PAC 2	36	9 març	13 abril	
3	Recollir els requisits de preparació de dades que plantegen els propietaris de les dades existents.	20	9 març	28 març	
4	Analitzar el requisits, classificar-los, i descriure a grans trets com es portarà a terme la preparació de les dades. les especificacions del nou programari.	15	29 març	12 abril	3
5	Lliurar PAC2	1	13 abril	13 abril	4
6	PAC 3	34	14 abril	17 mai	
7	Definir i documentar com es podran especificar a través de fulls de càlcul simples les transformacions a aplicar a les dades.	33	14 abril	16 mai	5
8	Lliurar PAC3	1	17 mai	17 mai	7
9	Lliurament Final	32	18 mai	18 juny	
10	Desenvolupar el programari que de forma eficaç i eficient obtingui a partir d'un conjunt de dades i de transformacions un nou conjunt de dades transformades.	31	18 mai	17 juny	8
11	Lliurament Final	1	18 juny	18 juny	10
12	Debat Virtual	12	19 juny	30 juny	
13	Desenvolupment	11	19 juny	29 juny	11
14	Tancament	1	30 juny	30 juny	13

La redacció la documentació i lliurables necessaris per el projecte (Memòria, Presentació Virtual, Producte i Manual d'usuari) es duran a terme en el transcurs del mateix, de forma progressiva, mantenint un control de versions per tal de poder fer un seguiment de l'evolució. El següent diagrama de Gant reflexa aquest fet.



Id	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	TFC	114 días	mié 09/03/0	mié 30/06/0	
2	PAC 2	36 días	mié 09/03/0	mié 13/04/0	
3	Recollir els requisits de prepa	20 días	mié 09/03/0	dom 29/03/0	
4	Analitzar el requisits, classific	15 días	lun 29/03/0	lun 12/04/0	3
5	Lliurar PAC2	1 día	mié 13/04/0	mié 13/04/0	4
6	PAC 3	34 días	mié 14/04/0	lun 17/05/0	
7	Definir i documentar com es p	33 días	mié 14/04/0	dom 16/05/0	5
8	Lliurar PAC3	1 día	lun 17/05/0	lun 17/05/0	7
9	Lliurament Final	32 días	mié 18/05/0	vié 18/06/0	
10	Desenvolupar el programari q	31 días	mié 18/05/0	jue 17/06/0	8
11	Lliurament Final	1 día	vié 18/06/0	vié 18/06/0	10
12	Debat Virtual	12 días	sáb 19/06/0	mié 30/06/0	
13	Desenvolupment	11 días	sáb 19/06/0	mié 29/06/0	11
14	Tancament	1 día	mié 30/06/0	mié 30/06/0	13

1.5 Productes obtinguts

El producte obtingut en aquest TFC es una fitxer de base de dades de Access.

Encara que l'idea final es transportar aquesta solució sobre un SGDB (Sistema Gestor de Base de Dades) com Microsoft SQL Server, es desestima incloure aquest transport del programari per manca de temps dins de la duració del TFC.

El fet de que el usuaris del programari parlen diferents llenguatges, i sent

l'anglès l'idioma dins de la companya, ha estat necessari que aquest idioma sigui l'emprat a la hora de donar nom a les taules, camps, consultes, formularis, funcions, variables, descripcions, comentaris, etc.

1.6 Breu descripció dels altres capítols de la memòria.

Capítol 2. EL CONJUNT DE DADES DE ENTRADA : detalla quin es el conjunt de dades amb el que s'ha d'alimentar el programari per tal d'obtenir les dades transformades.

Capítol 3. CARREGA DEL CONJUNT DE DADES D'ENTRADA : el conjunt de dades de entrada es subministrarà des de fitxers de text pla, fulls de càlcul (Microsoft Excel, en davant Excel) i /o taules en una base de dades (Access o Microsoft SQL Server), i en aquest capítol es descriuen els problemes superats en aquest aspecte.

Capítol 4. REGLES DE TRANSFORMACIÓ : es dona un repàs a les regles de transformació producte del anàlisis dels requeriments de transformació.

Capítol 5. SELECCIÓ DE REGLES PER SUBCONJUNTS DE DADES : cada taula de regles conté les transformacions a aplicar a varis subconjunts de dades amb la mateixa estructura, trobant-se entremesclades regles vàlides per tots els subconjunts i regles específiques per algun dels subconjunts, en aquest capítol s'explica com s'ha resolt aquesta funcionalitat

Capítol 6. PRIMERA APROXIMACIÓ DE RESOLUCIÓ : es descriu la primera solució, de tipus iteratiu (recorregut de totes les dades), sobre la que es va començar a treballar i que va quedar arraconada donat que la velocitat de transformació de dades feia inviable el seu ús pràctic al aplicar-la sobre volums de dades reals.

Capítol 7. RESOLUCIÓ FINAL MITJANÇANT LA CREACIÓ DE SENTENCIES SQL DE ACTUALITZACIÓ PER CADA REGLA : es descriu la solució final que ha resolt de forma categòrica el problema de la transformació de volums reals de dades (grans quantitats de dades a transforma en contra dels molt petits conjunt de dades utilitzats per fer proves) : s'han fet proves de transformació comparant la solució inicial amb recorregut de dades amb la solució final per sentències SQL, utilitzant els mateixos conjunts de dades d'entrada, i el segon algoritme ha resultat entre *100 i 600* vegades més ràpid que el primer. Aquest segon algoritme es la més important de les aportacions d'aquest TFC.

Capítol 8. SÉNTENCIES SQL PER TAULES DE REFERENÇA CREUADA : moltes de les transformacions a realitzar es basen en llistes on per cada valor en un camp de les dades d'entrada s'indica un nou valor per les dades de sortida; el desenvolupament del programari ha donat lloc a diferents tipus de taules de referència creuada. La part més complexa de la solució final ha estat ajustar la rutina que genera les sentències SQL per les regles que utilitzen taules de referència creuada. Per això es dedica aquest capítol apart.

2 ELS CONJUNTS DE DADES DE ENTRADA I SORTIDA

El conjunt de dades d'entrada del programari de transformació esta format per 5 grups de dades : Taula de Regles, Taules de Referència Creuada, Taula de Estructura de Dades, Taules de Dades d'Entrada, Taules auxiliars del programari. En els apartats d'aquest capítol s'utilitza la descripció de les taules principals com guia per descriure l'arquitectura del programari.

En cada apartat figura el nom del conjunt de dades, les sigles associades entre parèntesis i després dels dos punts el nom de la taula en el programari producte d'aquest TFC.

A cada conjunt de dades se l'hi ha assignat sigles, relacionades amb la seva descripció en anglès. Aquestes sigles estan recollides en el glossari i són RUL, CRT, RST, IDF i ODF.

2.1 Taula de Regles (RUL) : *FileRulesTable*

En la taula de regles es on es recopila la informació bàsica per fer les transformacions, incloent els camps afectats i les accions a dur a terme. La informació s'organitza de la següent manera:

Camp	Tipus [1]	Descripció
IdTR	Auto	Clau única per l'ús intern del programari
RECORD_TYPE	Text(3)	Identificador del tipus de registre[2] al que afectarà la regla. Es verifica amb la taula auxiliar "*RecType*". Si el valor es igual a '*' (*asterisc)* la regla s'aplica a qualsevol tipus de registre
SYSTEM_ID	Text(3)	Identificador del sistema[2] informàtic del que prové el fitxer d'entrada. Es verifica amb la taula auxiliar "*System*". Si el valor es igual a '*' (*asterisc)* la regla s'aplica sigui quin sigui els

[1] Els tipus de dades emprats estan recollits a l'annexa *12.1Abreviatures per tipus de camp*

[2] Per més informació veure Capítol 5. SELECCIÓ DE REGLES PER SUBCONJUNTS DE DADES

Camp	Tipus [1]	Descripció
		sistema origen de dades.
LEVEL	Text(3)	Nivell de regla. Les regles es poden aplicar en tres fases (Pre-procés amb codi '-', Transformació amb codi 0..n, i Post-procés amb codi +). Per el moment, tan sols les regles de la fase de transformació es poden aplicar en capes diferents, començant per la de mes petit nivell (0) fins la de més alt referit, de manera que s'asseguri que tota transformació es durà a terme avanç que aquelles que tinguin assignat un nivell superior.
S_FIELD	Text(30)	Nom del camp font (en anglès *Source Field*). Per que la regla sigui aplicada el nom, tal i com es trovi escrit, haurà d'estar inclòs en la taula d'estructura de dades.
T_FIELD	Text(30)	Nom del camp destí (en anglès *Target Field*). Per que la regla sigui aplicada el nom, tal i com es trobi escrit, haurà d'estar inclòs en la taula d'estructura de dades. Si es deixa en blanc o igual a "*" (asterisc) llavors es pren el mateix valor que figura en S_FIELD.
RULE	Text(3)	El valor ha d'estar inclòs en la taula auxiliar "*Rules*". Si no es defineix cap valor llavors es pren per defecte la regla de copia (CP). Més informació a l'apartat "Regles de Transformació"
VALUE	Text(80)	Depenent de la regla indicada en el camp anterior el valor indica: un valor per substituir, el nom d'una taula de referència creuada i opcionalment les claus de cerca, i finalment el nom d'una funció i el paràmetres necessaris per les regles de càlcul.

2.1.1 Filtrat de regles

El valor en el camp RECORD_TYPE per el tipus de registre, i del camp SYSTEM_ID per el sistema, permet la combinació en una única taula la informació de transformacions generals i altres d'específiques per un determinat tipus de registre i/o sistema

2.1.2 Fases

Com conseqüència de requeriments apareguts durant el desenvolupament del

programari es va fixar la necessitat de distingir 3 fases de transformació. A continuació es detalla les diferències de cadascuna d'elles:

Pre-procés

En aquesta fase les regles duen a terme transformacions que tenen com origen i destí la taula de entrada de dades, permeten netejar i preparar les dades si es cal. Normalment s'eliminen els registres erronis, caducats, fora d'ús, etc., per tal de minimitzar el treball de transformació. També es dupliquen o explosionen d'altres registres, per exemple, per el mestre de materials algun d'ells s'ha de duplicar assignant una nova organització de vendes virtual utilitzada per l'àrea financera. En certa manera es duen a terme las tasques conegudes com *Cleansing* (en anglès *neteja*) en l'entorn de Mineria de Dades per la preparació de dades.

Transformació

En aquesta fase les regles duen a terme transformacions que tenen com origen els valors a la taula d'entrada i com destí la actualització dels valors a taula de sortida de dades.

Avanç d'executar la primera transformació i just després d'haver finalitzat les transformacions corresponents a la fase de pre-procés s'aplica preliminarment la transformació per defecte : copiar. Tots els registres de la taula de dades d'entrada no marcats per esborrar es copien a una nova taula de dades de sortida.

Post-procés

En aquesta fase les regles duen a terme transformacions que tenen com origen i destí la taula de sortida de dades, permeten netejar i preparar les dades si es cal. Per exemple, en aquesta fase es poden eliminar columnes de la taula que en estat necessàries per aplicar les transformacions però que no han de estar

presents en el fitxer de sortida.

2.2 Taules de Referència Creuada (CRT) : *CT_Tables*

Complementen la informació de transformació especificada a les taules de regles. També utilitzen els filtres *RECORD_TYPE* i SYSTEM_ID de la mateixa manera que la taula de regles. Las taules son carregades per el programari des de la base de dades Access que s'indiqui per els paràmetres, rebent com nom el que tenien originalment amb el prefix "*CT_*".

Camp	Tipus	Descripció
IdTR	Auto	Clau única per l'ús intern del programari
RECORD_TYPE	Text(3)	Identificador del tipus de registre al que afectarà la regla. Es verifica amb la taula auxiliar *RecType.* Si el valor es igual a '*' (*asterisc)* la regla s'aplica a qualsevol tipus de registre
SYSTEM_ID	Text(3)	Identificador del sistema informàtic del que prové el fitxer d'entrada. Es verifica amb la taula auxiliar Systyem. Si el valor es igual a '*' (*asterisc)* la regla s'aplica sigui quin sigui els sistema origen de dades.
KEY1 (opcional)	Text(255)	Primera clau ce cerca, el nom de la clau, tal i com es trobi escrit, haurà d'estar inclòs en la taula d'estructura de dades.
KEY2 (opcional)	Text(255)	Segona clau de cerca, el nom de la qual, tal i com es trobi escrit, haurà d'estar inclòs en la taula d'estructura de dades.
---		...
KEYn (opcional)	Text(255)	Enèsima clau ce cerca, el nom de la qual, tal i com es trobi escrit, haurà d'estar inclòs en la taula d'estructura de dades.
NEW (opcional)	Text(255)	Conté el nou valor que haurà de substituir el del camp de destí indicat a la regla de transformació. El nom d'aquest camp es NEW, i no coincideix amb cap dels definits a la taula de estructura de dades.
NEW_*SKEY1* (regla CR)	Text(255)	S'utilitza tan sols per la regla CR *Clone Record* (veure apartat Regles de Registre). Conté el nou valor que haurà de substituir el del camp clau de destí *SKEY1*, es a dir que el nom d'aquest camp es NEW_SKEY1, i treien l'inici del nom NEW_ ens queda el nom del camp clau, que tal i com es trobi escrit, haurà d'estar inclòs en la taula d'estructura de dades.

Camp	Tipus	Descripció
NEW_*SKEY2* (regla CR opcional)	Text(255)	Nou valor per segon camp clau ídem anterior.
---		...
NEW_*SKEYm* (regla CR opcional)	Text(255)	Nou valor per emèsim camp clau ídem anterior.

El format final de les taules depèn de la seva finalitat, directament definida per la regla de transformació que l'utilitza. La taula auxiliar "*CrossTableType*" recull les columnes requerides en cadascuna d'elles. En tot cas les columnes RECORD_TYPE i SYSTEM_ID estan presents a tots el tipus de taules.

2.3 Taula de Estructura de Dades (RST) :

FileRecordStructureTemplate

En aquesta taula es recull la definició de la taula de dades d'entrada i també de la de sortida.

Camp	Tipus	Descripció
IdRST	Auto	Clau única per l'ús intern del programari
NAME	Text(255)	Nom del camp. En quasi bé totes les ocasions coincideix amb el nom del camp destí de l'ERP SAP
TYPE	Text(255)	Tipus de dada seguint les especificacions de format utilitzades per SAP. Es verifica amb la taula auxiliar *FieldType*.
LENGHT	Text(255)	Longitud del camp seguint les especificacions de format utilitzades per SAP.
DESCRIPTION	Text(255)	Descripció opcional del camp. Informació de suport opcionalment introduïda per l'usuari. No es utilitzat per el programari.

2.4 Taula de Dades d'Entrada (IDF) : *InputData*

Aquesta taula conté uns camps fixes per l'ús intern del programari i la resta depenen dels que es defineixen dinàmicament dins la taula de Estructura de Dades descrita en l'apartat anterior.

Camp	Tipus	Descripció
IdID	Auto	Clau única per l'ús intern del programari
RnID	Num()	Número de registre en el fitxer d'entrada. Donat que els registres es poden duplicar serveix per fixar la procedència exacta de qualsevol registre. Per tant, en la columna es poden trobar valors duplicats.
DeleteFlag	Bol	També per qüestions de seguiment de les dades, quan un registre s'ha de esborrar com resultat de l'aplicació de un transformació, tan sols es marca com esborrat, però es conserva per tal de poder fer verificacions posteriors.
ValidFlag	Bol	S'utilitza per la regla VR per marcar els registres validats.
CAMP1	*Text(?)*	*Camp creat dinàmicament segons la informació proporcionada per la RST*
CAMP2	*Text(?)*	*Ídem anterior*
,,,		*...*
CAMPn	*Text(?)*	*Ídem anterior*

2.5 Taula de Dades d'Entrada (ODF) : *OutputData*

A diferència de les taules anteriors, esta es resultat del programari. Com la taula IDF aquesta taula conté uns camps fixes per l'ús intern del programari i la resta depenen dels que es defineixen dinàmicament dins la taula de Estructura de Dades descrita en l'apartat anterior.

Camp	Tipus	Descripció
IdOD	Auto	Clau única per l'ús intern del programari
IdID	Num()	Número de registre en la taula d'entrada. Donat que els registres es poden duplicar serveix per fixar la procedència exacta de qualsevol registre. Per tant, en la columna es poden trobar valors duplicats.
RnID	Num()	Ídem taula IDF
DeleteFlag	Bol	Ídem taula IDF, però en aquest cas la marca s'utilitza per no bolcar les dades al fitxer de sortida de dades.
ValidFlag	Bol	Ídem taula IDF
CAMP1	*Text(?)*	Ídem taula IDF
CAMP2	*Text(?)*	Ídem taula IDF
,,,		*...*
CAMPn	*Text(?)*	Ídem taula IDF

2.6 Taules auxiliars del programari

Les taules auxiliars recullen valors validats per verificar la informació de transformació.

Nom de la taula Informació continguda	Descripció
RecType Tipus de Registre	Conté els tipus de registre vàlids. Els principals estan associats als diferents tipus de materials i de clients. Permeten establir regles específiques per subconjunts de dades que provenen d'un mateix sistema.
System Sistemes	Conté els identificadors dels diferents sistemes que proveiran les dades d'entrada. En principi, els codis fan referència al nom dels països, però realment es tracta d'un inventari de sistemes. Permeten establir regles específiques per les dades de cada sistema.
Rule Regles	Conté els codis de les regles de transformació, la seva descripció e informació per l'ús intern del programari.
Delimiter Separadors	Llista de separadors per els camps de text dels fitxers de text pla en el que es subministren les dades d'entrada.

Nom de la taula Informació continguda	Descripció
FieldMaskDate Màscares per dates	Llista de formats en els que el programari pot importar i exportar le dades de tipus numèric. En concret permet definir quins son els separadors de milers i decimals i la posició del signe, al principi o al final del número.
FieldMaskIs_Number Màscares numèriques	Similarment a l'anterior camp però per el format de les dates. En concret defineix l'ús o no de separadors entre l'any, el mes i el dia, i l'ordre d'aquests valors dins de la data.
FieldType Tipus de dades	Llista amb el tipus de dades admesos per el programari i amb els que han de coincidir el emprats a la taula de estructura de dades. El tipus corresponen als emprat per l'ERP SAP.

3 CARREGA DEL CONJUNT DE DADES D'ENTRADA

Ha estat un requisit del programari aconseguir la màxima flexibilitat per introduir la informació de transformació per part dels usuaris.

Per aquesta raó s'havien de poder recollir la informació necessària a partir del fitxer de text pla, fulls de càlcul (Excel) i /o taules en una base de dades (Access).

La dificultat a estat en que si bé Access disposa de funcions que permeten importar fitxers dels tipus esmentat de forma automàtica, ha resultat que aquestes funcions han presentat problemes insuperables quan s'ha tractat de importar fitxers de text pla o fulls de càlcul amb certes peculiaritats.

En els fitxers de text pla hi han problemes amb el formats numèrics, les dates i els separadors. En concret el problemes són:

- els valors numèrics provenint de SAP porten el signe a la dreta
- hi han fitxers de dades d'entrada amb dades numèriques amb i d'altres sense separador de milers, uns on el separador decimal es una coma i d'altres és un punt.
- hi han columnes que representen codis alfa-numèrics, que en les primers registres són tan sols numèrics i que després presenten cadenes de caràcters, en aquest cas las rutines d'importació defineixen la columna de la taula de destí con numèrica, amb el conseqüent error quan s'intenta afegir una cadena alfa-numèrica.
- el programari s'executa en ordinadors amb configuracions regionals diverses

Problemes similars han aparegut a l'hora d'importar els fulls de càlcul.

Per tot això s'ha perdut l'avantatge de poder utilitzar les funcions imbuïdes en Access per la importació i s'han tingut que escriure funcions *ad hoc* per realitzar aquestes tasques.

Totes aquestes rutines es troben al mòdul "*ModLoatToTable*", on també s'ha creat una funció "*LoadToTable*" que permet unificar la crida per la càrrega de dades des de un fitxer extern a una taula interna.

Per la càrrega de les dades d'entrada del fitxers de text pla, s'ha crear una rutina específica per raons de rendiment, ja que s'ha aprofita per incloure la preparació de cada valor avanç d'escriure-ho en la taula de dades d'entrada. Aquesta funció és "*LoadInputData*" que es troba inclosa al mòdul "*ModDTM*".

La resolució satisfactòria d'aquests problemes, desconeguts fins l'últim moment ha implicat un esforç suplementari per recopilar la informació necessària i posar a punt les funcions.

En la versió actual no tots els conjunt de dades d'entrada es poden llegir des de qualsevol format d'entrada. La taula a continuació o reflexa.

Conjunt de Dades E/S	Text Pla	Excel	Access
RUL – Taula Regles	Disponible	Disponible	Disponible
CRT - Taules Referència Creuada	-	Pendent	Disponible
RST – Taula Estructura de Dades	Disponible	Disponible	Disponible
IDF – Taula Dades Entrada	Disponible	Pendent	Pendent
ODF – Taula Dades Sortida	Disponible	Pendent	*preparat*

Figura 3-1: Tipus fitxers suportats per grups de dades d'entrada i sortida

4 REGLES DE TRANSFORMACIÓ

Les regles de transformació s'agrupen en 4 tipus segons el tipus de transformació que codifiquin.

- Tipus Taula : defineixen transformacions com afegir, eliminar o reanomenar una columna, permeten "adaptar" els fitxers de dades extrets dels sistemes font de dades al format del magatzem comú de dades. També hi ha una regla per la creació de una nova taula a partir d'una taula d'entrada de dades, per tal de facilitar la creació de taules de referència creuada utilitzades per el programari i evitar haver d'extreure informació repetidament dels sistemes origen.
- Tipus Registre : defineixen transformacions que, bàsicament esborren o creen nous registres.
- Tipus Camp : són les regle bàsiques, i defineixen transformacions com esborrar el valor d'un camp, substituir-lo per un valor constant, copiar en un camp el valor que es troba en un altre (del mateix registre).
- Tipus Càlcul : son la porta oberta ha introduir transformacions més complexes, que precisin d'un codi de programació complex, i que per tant no queden recollides per les funcionalitats proveïdes per les regles anteriors.

En l'aplicatiu les possibles regles es troben codificades a la taula auxiliar "*Rules*". Seguidament es descriuen breument les regles. En el moment de tancar aquest TFC encara no estan totes implementades, encara que si les més importants.

4.1 Regles de Taula

Aquestes regles esdevenen una eina per automatitzar canvis en el format de les taules d'entrada i sortida de dades, ja que totes dues comparteixen el mateix format, segons s'ha definit en les especificacions del programari. La seva finalitat es adaptar el format d'una taula de entrada de dades al format de

sortida requerit per el magatzem comú de dades final. Tot i això, la recomanació es que les extraccions de dades des de els sistemes origen es facin utilitzant el format de dades que hauran de tenir en el magatzem comú al que van destinades, i per tant les regles següents s'han d'utilitzar com últim recurs.

Codi Regla : descripció en anglès	Descripció
AC : Add Column	afegir una nova columna a la estructura de la taula de dades. (pendent implementació)
DC : Delete Column	esborrar una columna existent a la estructura de la taula de dades.
RC : Rename Column	canviar el nom d'una columna de la taula de dades. (pendent implementació)
CCT : Create Crosstable	a partir de la taula de entrada o sortida crea automàticament una taula del tipus CRT (*Cross-Reference Table*) que després pot ser utilitzada per altres regles de transformació. (pendent implementació)

4.2 Regles de Registre

Per definició de les regles que dupliquen registres requereixen indicar els valors nous en taules CT per les claus del registre, evitant que el amb la creació de nous registres es dupliquin claus existents.

Codi Regla : descripció en anglès	Descripció
CR : Clone Record	Afegir registres com a copia d'uns altres : a una taula s'indica per quins valors clau s'haurà de duplicar un registre i quins nous valors clau haurà de portar. Es conserva sense canvis el registre original, així si tenim un registre per el que tenim dos encerts a la llista de claus a "clonar" obtindrem finalment 1 + 2 = 3 registres.

Codi Regla : descripció en anglès	Descripció
EX : Explode records	Explosionar registres com a copia d'un altres : similar a la regla CR, amb la diferència de s'elimina el registre original, així si tenim un registre per el que tenim dos encerts a la llista de claus a "clonar" obtindrem finalment 1 + 2 – 1 = 2 registres. (pendent implementació)
DR : Delete Record	Eliminar registres : a una taula s'indica per quins valors clau s'haurà de eliminar un registre.
DX : Delete records eXcept	Eliminar registres no validats : aquesta regla és complementaria de l'anterior, i en una taula s'indica per quins valors clau NO s'haurà de eliminar un registre.

4.3 Regles de Camp

Codi Regla : nom en anglès	Descripció
CP : Copy rule	Per un registre donat, copia al camp destinació especificat el valor contingut en el camp d'origen especificat. El camp origen i destí poden ser el mateix.
SB : Substitution rule	El camp de destí s'omple amb el valor indicat.
DE : DEletion rule Field	El camp de destí s'omple amb una cadena buida.
CT : Cross-reference Table	Per un registre donat, omple el camp destinació especificat amb el valor indicat mitjançant una taula de referències creuades, en la que es selecciona una entrada segons el valor/s contingut/s en el/s camp/s d'origen. El camp destí pot coincidir o no amb un dels camps d'origen.

Codi Regla : nom en anglès	Descripció
ST : Substitution Table	Similar a la regla SB permet agrupar en una taula valors de substitució per els diferents subconjunts de dades.

4.4 Regles de Càlcul

Codi Regla : nom en anglès	Descripció
CA : Calculation	Té associada una funció escrita en VBA (Visual Basic for Applications), la qual te com paràmetres, constants i variables que permeten recuperar el valor d'un o més camps per cada registre d'entrada, i que després de l'execució del codi implementat, retorna un valor que s'assigna al camp de destí. El codi contingut a la funció s'executarà tantes vegades com registres tingui el fitxer d'entrada.
CAC : Calculation Column	Similar a l'anterior, té associada una funció escrita en VBA que tan sols té com paràmetres valors constants. El codi contingut en una funció s'executarà 1 sola vegada per cada referència que s'hi faci en les regles de transformació validades, independentment del nombre de registres que tingui el fitxer d'entrada. Es una porta oberta a la execució de subprogrames per portar a terme transformacions que, per la seva complexitat, no es poden resoldre amb les regles anteriors.

En el programari producte d'aquest TFC s'han inclòs funcions per donar suport a diversos requeriments. D'aquestes funcions les més rellevants per la seva

importància i complexitat serveixen per fer transformacions en el mestre de materials, ja que es modifica la unitat bàsica de mesura i llavors s'han de poder convertir les quantitats de/a litres, quilograms, peces, bots, capsa, etc. utilitzant per cada valor taules de conversió d'unitats. Aquestes funcions es "criden" des de les regles de transformació utilitzant una regla CAC. Les funcions són:

- *CA_RULE_fctMARMUpdatesDependingOnnewBUOM* : actualitza els factors de conversió d'unitats depenen de la nova unitat base de mesura per cada producte. Inicialment aquest era un programa extern amb una solució també de recorregut iteratiu de la taula de dades d'entrada. Després de analitzar els requeriments, s'ha aconseguit resoldre amb la combinació d'una consulta i d'una sentència d'actualització SQLs que es creen i executen per la funció esmentada. Es requereix que aquest factors siguin el números enters menor possibles, per el que s'ha implementat també la funció *MinimumFactors* per calcular el màxim comú divisor utilitzant l'algoritme d'Euclides.
- *CA_RULE_fctQuantConversionBUOM* : converteix les quantitats guardades a un camp especificat a unes altres segons el canvi d'unitats base de mesura especificat per cada material. El desenvolupament d'aquesta funció a seguit els passes de l'anterior.
- *CA_RULE_fctUOMConversionDependingOnBUOM* : dona nou format les unitats de mesura alternatives guardades a un camp especificat segons les noves unitats base de mesura especificades per cada material. El desenvolupament d'aquesta funció a seguit els passes de l'anterior.

5 SELECCIÓ DE REGLES PER SUBCONJUNTS DE DADES

Com ja hem vist, cada una de les regles recopilades en una taula de regles pot afectar a dades de qualsevol sistema origen, o bé estar definides específicament para un sistema en particular, això es combina amb que a la vegada també cada regla pot estar definida per qualsevol tipus de registre o be ser especifica per un tipus de registre en particular.

Com sigui que cada vegada que s'executa el programari és obligatori indicar com a paràmetre quin es el sistema origen del fitxer, i quin tipus de registre tracta, el que fa el programari es començar per descartar les regles que específicament estan destinades a un altre sistema o tipus de registre

5.1 El problema de la combinació de regles

Cada taula destí en el magatzem comú de dades final prové de la combinació de al menys una taula per cada sistema origen existent. Aquestes taules origen comparteixen, en general, la mateixa estructura de dades, encara que, òbviament, contindran informació diferent.

És per aquesta raó que els usuaris van afegir un nou requeriment per tal de poder agrupar la informació al respecte de les transformacions requerides per el mateix tipus de dades, encara que vinguessin de diferents sistemes, per tal de facilitar la seva feina a l'hora d'especificar-les.

La mateixa problemàtica apareix per dades d'entrada que venen d'un mateix sistema, i que amb la mateixa estructura, pertanyen a subconjunts diferents i com a conseqüència requereixen transformacions diferents.

Aquests requeriments es van plantejar quan ja s'havien desenvolupat varies versions del programari.

Així que encara que el funcionament bàsic del programari de transformació es va definir i construir per transformar en cada execució una única taula de dades

d'entrada, provenint d'un únic sistema, segons la informació de transformació especificada en la corresponents taula de regles i taules de referència creuada, s'ha modificat el programari per tal de que la informació de transformació es pugui especificar de manera conjunta. És per això que tan la taula de regles com la taules de referència creuada tenen dos camps, anomenats RECORD_TYPE i SISTEM_ID, que permeten que cada regla i cada entrada en les taules de referència creuada quedi designades per us general, amb el valor '*' (asterisc), o per un grup de dades específic.

Com a conseqüència l'esquema inicial de requeriments es modifica de la segons es reflexa a l'esquema en la propera pàgina (Figura 5-1).

En l'esquema s'indica que tan la taula de regles com las taules de referència creuada contenen informació de transformació genèriques i també específiques per dos tipus de registres FP i RW (en aquest cas productes acabats i matèries primeres), , i també per els sistemes actuals DE, AT i ES (en aquest cas el alemany, el austríac i l'espanyol).

El camp RECORD_TYPE permet especificar informació de transformació específica per un subconjunts de dades del mateix tipus i sistema. És el cas del mestre de clients, per el que s'especifiquen transformacions genèriques i d'altres que depenen si el client es un deutor o un proveïdor. També és el cas del mestre de productes, on hi han transformacions que dependran del valor i del sistema origen, i d'altres que depenen del tipus de producte.

El camp SYSTEM_ID funciona con l'anterior, però esta destinat a distingir els diferents sistemes actuals des del que es recull la informació d'entrada.

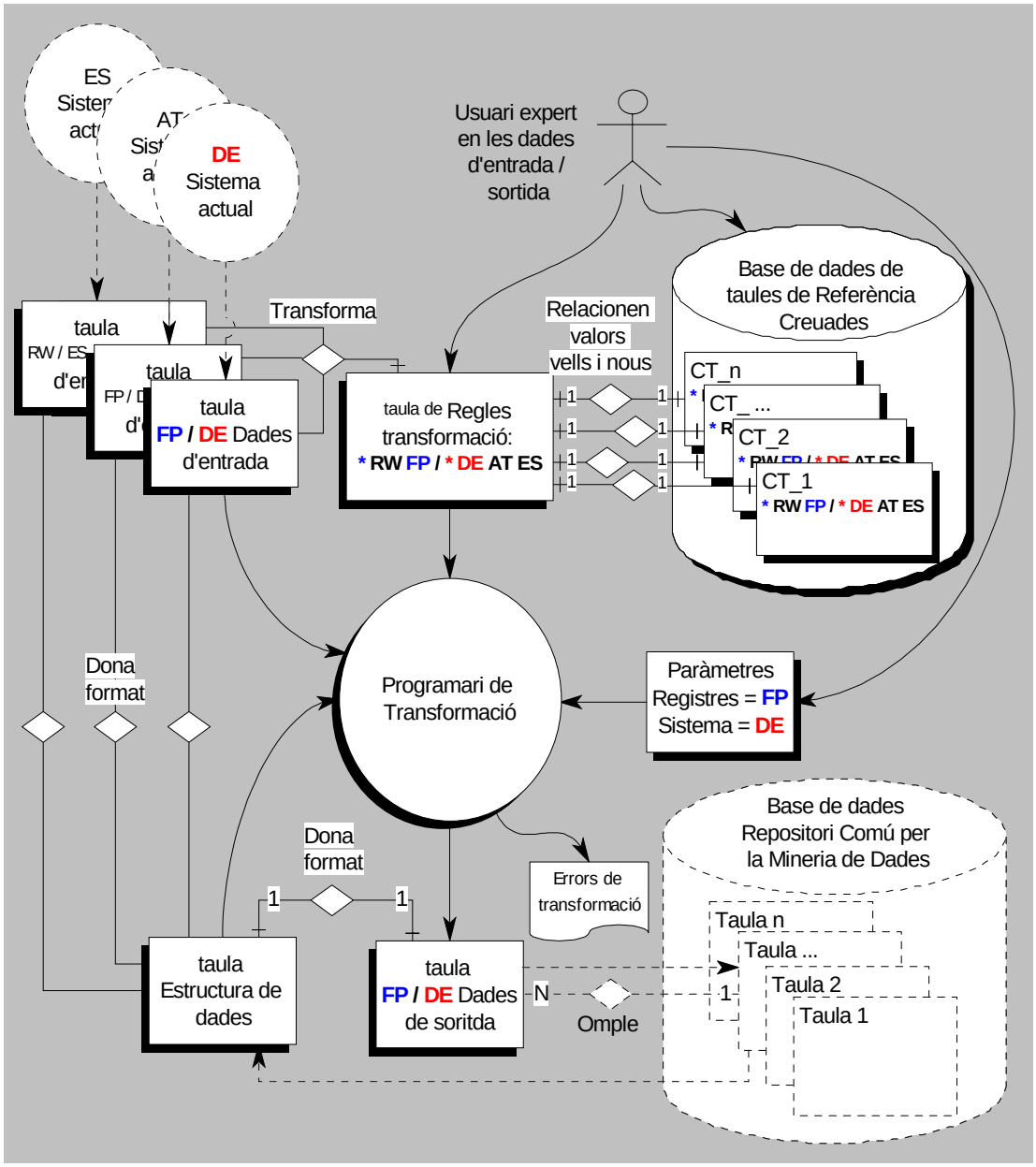


Figura 5-1: Esquema del programari amb regles combinades

Amb tot això una mateixa taula de Regles, i un mateix conjunt de Taules de Referència Creuada, podem transformar diferents grups de Dades d'Entrada provenint de diferent sistemes i/o que pertanyen subconjunts de dades (tipus de registres) diferents.

6 PRIMERA APROXIMACIÓ DE RESOLUCIÓ

6.1 L'algoritme primer

L'algoritme inicialment proposat per fer la transformació de les dades d'entrada a partir de la informació especificada a la taula de regles i a les taules de referència creuada constava del següents processos :

- Carregar la taula de estructura de dades (RST)
- Crear una taula "intermitja" amb el mateix nombre de columnes que l'indicat a la taula RST per tal de carregar les dades d'entrada des de el fitxer de text indicat. La finalitat d'aquesta taula intermitja era portar el contingut del fitxer de text a dins de la base de dades per després poder accedir directament. Aquesta taula era creada fen ús de les fucnions d'importació proveïdes per Access, i per això hi havia moltes limitacions.
- Crear una taula per les dades d'entrada amb el format especificat a la taula RST.
- Transferir el contingut de les dades d'entrada des de la taula intermitja fins la taula de dades d'entrada, fent un recorregut iteratiu registre per registre i columna per columna. Durant el procés es detectaven possibles error de format, com rebre una cadena de 20 caràcters per posar-la en un camp amb longitud màxima de 5, o rebre una cadena de caràcters per posar-la en un camp numèric.
- Recorrer la taula de regles (RUL) cercant tant possibles errors com regles no definides, o noms de camps no inclosos en la RST
- Aplicant la regla explicita per defecte de la transformació, copiar la taula d'entrada de dades sobre la taula de sortida.
- Seguidament es posava en marxa el procés propi de transformació : la taula de dades d'entrada es recorria iterativament, registre per registre, camp per camp, i per cada valor a cada camp es cercava a la taula de regles la corresponent al primer nivell o capa corresponent al camp en

qüestió, llavors, depenent de si la regla era del tipus de taula de referència creuada, es cercava el valor en tractament per tal d'obtenir el nou valor corresponent. En cas de que hi haguessin regles de segon nivell el recorregut de la taula de dades d'entrada es tornava a fer.

6.2 El Problema de Rendiment

El procés de transformació descrit comportava, en el millor dels casos l'exploració d'un nombre de valors igual al nombre de columnes de la taula de dades d'entrada, multiplicat per el nombre de registres, i tenim casos com el mestre de materials, en el que el nombre de registres en una taula de entrada de dades es de més de 235.000, amb 63 columnes, el que dona un total de quasi 15 milions de iteracions a processar. Altres dades d'entrada consten de 100.000 registres i 135 columnes: es a dir 13,5 milions de iteracions. Al temps propi de fer aquest recorregut per recuperar cada valor de la taula d'entrada s'ha de afegir, en molts casos, el temps de cerca a les taules de referència creuada per trobar el valor per el qual s'ha de substituir. Així és que per aquesta única la taula de 100.000 registres i 135 columnes, que conté informació d'un sol sistema, i que es una de les moltes taules que es necessari transformar, el algoritme inicial de recorregut iteratiu descrit va estar en funcionament durant 5 dies, després dels quals es va decidir aturar el programa, ja que la segona i definitiva solució havia fet la mateixa feina en ¡ 35 minuts !

7 RESOLUCIÓ FINAL MITJANÇANT LA CREACIÓ DE SENTENCIES SQL DE ACTUALITZACIÓ PER CADA REGLA

El programari producte d'aquest TFC ha esta dissenyat i desenvolupat partint de 0 amb una cura especial per aconseguir la màxima velocitat de transformació.

Per aconseguir-lo l'idea principal va ser aprofitar la potencia interna del gestor de base de dades per tal d'eliminar tot el temps de procés que implica accedir a la base de dades des de VBA.

A partir d'aquí el següent pas era poder transformar cadascuna de les regles de transformació en una sentencia de actualització SQL.

Amb això desapareix la necessitat de recorre la taula d'entrada. En el seu lloc s'explora la taula de regles i es preparen de dinàmicament les sentències de actualització SQL adients.

Ràpidament es va comprovar que hi havia tota una sèrie de regles de transformació que tenien una solució "trivial", ja que la sentència SQL d'actualització es podia crear directament amb una altre sentencia SQL.

D'altres regles de transformació no son "trivials" i han requerit d'un programa que les construeixi dinàmicament. A més per les regles CT s'ha de comprovar el contingut de la taula CT referida per que, depenent de l'ús que es faci a les claus de cerca dins de la taula CT (KEY1, KEY2,...) de comodins i de valors nuls (NULL), es poden necessitar des de 1 fins a 3 elevat a (nombre de claus)

Totes les sentències SQL generades es guarden en la taula *Transformation_SQLcmd* on queden relacionades amb la regla de la que las son filles, i on també es guarden els detalls relatius a las combinacions de comodins i nuls a que donen suport, (aquest últims referits per camps que inclouen la partícula BLANK, encara que, per el moment son nuls, en anglès

NULL).

D'aquesta manera, prèviament a fer qualsevol transformació, ja es disposa de totes les sentencies SQL necessàries per dur a terme la transformació.

Això també facilita en gran mesura verificar possibles errors de transformació, no ja del programari, si no de la pròpia definició de la informació de transformació : cada camp en la taula de sortida ha estat actualitzat per un reduït nombre de sentències SQL, i resulta fàcil verificar els canvis aplicats per cadascuna d'elles i trobar els esmentats errors.

El fet que les transformacions es facin a través de sentències SQL també permet que en una propera versió del programari, tot el volum de dades quedi en un servidor de dades, per després executar les transformacions directament "dins" del servidor, lliurant als usuaris de la despesa de temps que significa fer les transformacions al seu ordinador i del temps que necessiten per "pujar" fitxer de dades entrada i "baixar" fitxers de sortida, amb les dades transformades, des de un servidor de fitxers. Addicionalment, si es volgués millorar encara més la velocitat de les transformacions, és molt més fàcil aconseguir augmentar la potència de processament en un servidor que fer-ho en un munt de ordinadors de sobretaula o portàtils.

7.1 Seqüència de processos en la execució del programari previs a la transformació

Prèviament a fer la transformació pròpiament dita, el programari processa paràmetres, estructura de dades, regles, taules de referència creuada, per crear el conjunt de sentències SQL que després duran a terme la transformació. Aquest processos, sense entrar en detall, són:

- Carregar el paràmetres especificats per l'usuari per l'execució del programari de transformació : tipus de registre, sistema, carpeta i nom

dels fitxers (regles, base de dades de taules creuades, estructura de dades, dades d'entrada), format numèric i de data de les dades d'entrada i el requerit per les dades de sortida i nom del fitxer i taula en el que es volen obtenir les dades de sortida.

- Carregar el fitxer d'estructura de dades (RST)
    - Verificar la informació de RST : si es fa referència a un tipus de registre no predefinit, o hi ha altres possible problemes el programari reporta l'error i s'atura.
- Carregar el fitxer de regles (RUL)
    - Verificar la informació de regles : si es fa referència a un regla desconeguda, o hi ha altres possible problemes el programari reporta l'error i s'atura.
    - Filtrar el fitxer de regles seleccionar tan sols aquelles designades com genèriques, o que igualen el tipus de registre i de sistema indicats en els paràmetres d'execució.
    - Eliminar de la taula de regles aquelles que el camp font, o el destí, o qualsevol clau referida en una regla CT, CA o CAC no estigui contingut a l'estructura de dades especificada per l'execució.
- Carregar les taula de referència creuades (CRTs) que es trobin entre les regles que s'hagin validat per els processos anteriors, es a dir, si al fitxer de regles original hi ha referida una taula CRT que després queda eliminada per el processos anteriors, la taula CRT en qüestió no es carregarà. La funció "*CrossReference_Tables"* del mòdul "*ModDTM*" porta a terme la feina.
    - Durant el procés de càrrega també es comprova el format de les taules CRT, s'eliminen entrades duplicades, i si hi han errors es reporten i el programa s'atura.
    - També durant la càrrega s'analitzen el contingut de comodins i del nuls dins de cada taula, essent aquesta informació guardada per utilitzar-la en la generació de les sentències SQL a la taula "*CrossReferenceKeyStarCombination* "
- Creà les sentencies SQL per regles relacionades amb taules CRT. En

funció de la informació preparada per el processos anteriors, s'executa la funció "*CreateSQLcmd_CT*" del mòdul "*ModDTM*" que genera dinàmicament i guarda les sentències de actualització SQL.

Les sentències SQL per las regles que no utilitzen taules CRT estan generades a la vegada per altres sentències SQL, ja que donada la seva simplicitat no precisen de cap rutina complexa. En la present versió es troben guardades com consultes d'Access, i són :

- "*QrySQLcmd_CA", "QrySQLcmd_CAC", "QrySQLcmd_CP", "QrySQLcmd_DC", "QrySQLcmd_DE", "QrySQLcmd_SB*"

El sufix del nom de cada consulta és el codi de la regal de transformació que genera. Obrin aquestes consultes en mode de disseny es pot veure com es generen les sentències SQL "simples" veient el contingut assignat al camp *SQLcmd* :

```
CA → "UPDATE InputData AS I INNER JOIN OutputData AS O ON I.idID = O.idID SET
     O.["+[T_FIELD]+"] = "+IIf([CA_FUNCTION]>"";
     "CA_RULE_"+[CA_FUNCTION]+[CA_ARGUMENTS]; "ErrorNotDefinedFunction")

CAC → "UPDATE Transformation_SQLcmd AS T SET T.EXECUTED = "+IIf([CA_FUNCTION]>"";
     "CA_RULE_"+[CA_FUNCTION]+[CA_ARGUMENTS]; "ErrorNotDefinedFunction")

CP → "UPDATE InputData AS I INNER JOIN OutputData AS O ON I.idID = O.idID SET
     O.["+[T_FIELD]+"] = I.["+[S_FIELD]+"]"

DE → "UPDATE OutputData AS O SET O.["+[T_FIELD]+"] = Null"

DC → "UPDATE RecordStructureTemplate AS RST SET RST.Deleted = True WHERE
     ((RST.NAME)='"+[TRANSFORMATION].[T_FIELD]+"')"

SB → "UPDATE OutputData AS O SET O.["+[T_FIELD]+"] = "+IIf([VALUE]>"";
     Chr(34)+[VALUE]+Chr(34); "Null")
```

Figura 7-1: Sentències SQL per regles sense taules CRT

7.2 Seqüència de processos en la execució del programari per la transformació

Encara que el programa no té una interfície d'usuari que ho faciliti, arribat a aquest punt tenim tota la informació necessari per fer la transformació, es a dir, el conjunt de sentencies SQL de actualització i les taules de referència creuada

ja filtrades, que s'han guardat amb el nom "CT_*nom_taula_original*". Tan sols queda carregà les dades d'entrada i executar les transformacions codificades amb sentències d'actualització SQL.

Aquestos processos, sense entrar en detall, són:

- Crear la taula d'entrada de dades "*InputData"* segons la informació continguda a la taula d'estructura de dades (RST).
- Carregar la taula "*InputData*" amb el contingut del fitxer de dades d'entrada indicat als paràmetres d'execució.
    - Durant aquest procés es comproven les dades d'entrada segons el tipus i longitud indicades per cada camp a la RST, en cas d'error el programari s'atura. (En el moment de tancar el TFC està pendent fer que es guardi l'error i el programa continuï.)
    - També es transformen els números i les dates del format que puguin tenir al fitxer d'entrada (indicat amb paràmetres d'usuari validats per les taules auxiliars "*FieldMaskIs_Number*" i "*FieldMaskDate*") al format intern per els números i les dates (-12345.6 i YYYYMMDD) respectivament. (En el moment de tancar el TFC tan sols transformen el números d'entrada)
    - Respecte a la primera versió hi ha un estalvi de temps significatiu quan els fitxer d'entrada de dades son grans.
- Inicialitzar variables per l'execució de les funcions associades a les regles de càlcul CA i CAC.
- Transformar les dades executant les sentències SQL en las 3 fases possibles:
    - En la fase de pre-procés aplicant les regles corresponents sobre la taula d'entrada.
    - Just finalitzada la fase anterior i prèviament a la següent, es crea la taula de dades de sortida "*OutputData*" i es copia el contingut la taula de entrada a la de sortida, deixant de banda els registres marcats per esborrar per les regles DR o DX.
    - En la fase de transformació s'apliquen les sentencies

corresponents a aquesta fase, començant per el nivell més baix i fins el més alt, involucrant tant la taula d'entrada i la de sortida.

- En la fase de post-procés aplicant les regles corresponents sobre la taula de sortida.

- Formatar i gravar les dades de sortida des de la taula "*OutputData*" al fitxer indicat al paràmetre corresponent.
  - Durant aquest procés es donen format a les dades de sortida segons la informació a la RST, incloent el format indicat en els paràmetres d'execució per els números i les dates.
  - Les columnes marcada per esborrar per una regla DC no es transfereixen, quedant a la pràctica esborrades en el fitxer de sortida.

Amb això conclou l'execució del programari producte d'aquest TFC.

Durant la codificació del programari s'ha evitat, sempre que ha estat possible, fer recorreguts iteratius de taules des de VBA, i s'han utilitzat al màxim sentències SQL creades dinàmicament. Així, per exemple, el processament preliminar del les taules CRT, incloent la carrega, filtrat, el anàlisis de comodins i nuls, i la seva preparació es fa íntegrament executant sentències SQL.

Totes aquestes tècniques donen un millor rendiment que no es pot comparar al de la primera solució. L'objectiu de millora de rendiment s'ha aconseguit.

8 SÉNTENCIES SQL PER TAULES DE REFERÈNCIA CREUADA

Els algoritmes necessaris per resoldre els requeriments finals per les taules CRT han estat d'una complexitat elevada. La varietat de tipus de taules CRT

Les regles CT apliquen transformacions basades en llistes on estan relacionats valors únics o combinats de les dades d'entrada, amb altres de nous que ompliran un camp o més de les dades de sortida.

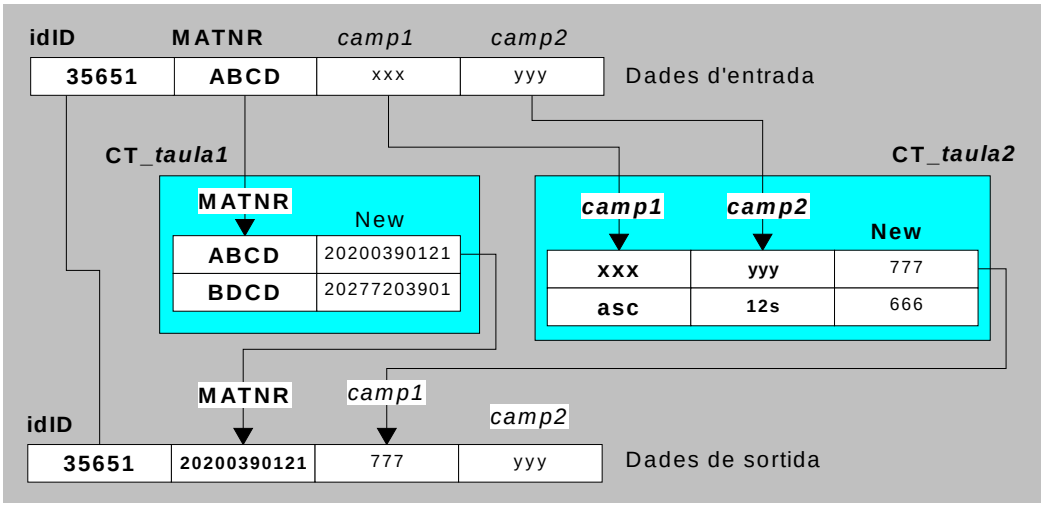


Figura 8-1: Funcionament bàsic de les taules creuades

Com ja s'ha apuntat en el capítol anterior, al igual que succeeix amb les regles de transformació, també les entrades de totes les taules CRT inclouen columnes on s'indica si valor per especificar si la transformació es específic per algun tipus de registre i sistema. La solució es la mateixa que per les regles de transformació.

8.1 L'ús de comodins i nuls com valors clau a les CRT

Però les CRT tenen una problemàtica afegida com a conseqüència d'altre requeriment per part dels usuaris fet després de la quarta versió del programari.

Es usuaris van requerir que per no haver d'especificar tots el valors possibles per una determinada clau requerien poder introduir en les CRT el valor '*' per significar "qualsevol valor". Com ha resultat es poden mesclar valors concrets

amb '*' dins de les claus. En el següent esquema es mostra un cas pràctic.

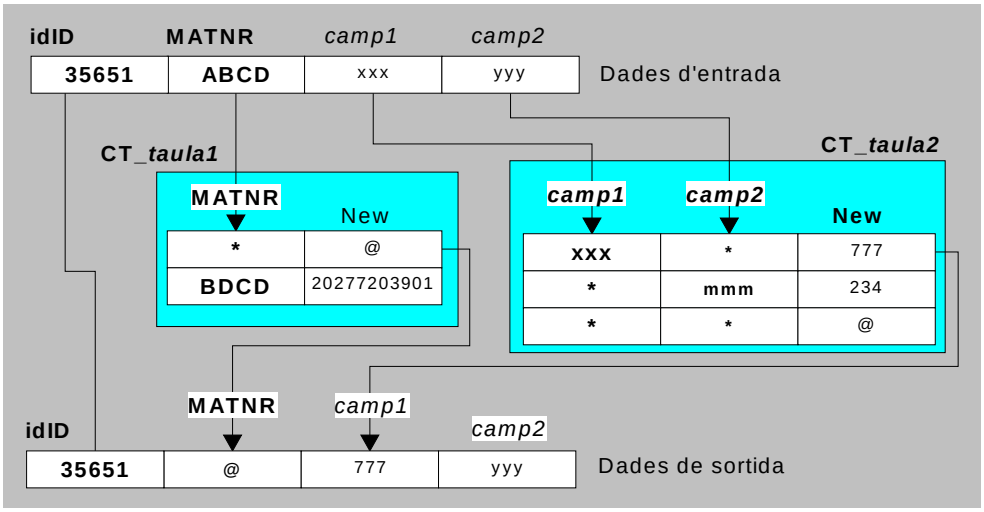


Figura 8-2: Funcionament de les taules creuades amb comodins

El programari fa una anàlisi a cada CRT utilitzada de l'ús que es fa per columnes d'aquest símbol comodí. Si no hi ha cap comodí en cap columna clau de una CRT tan sols es necessita una única sentència SQL de actualització per la regla CT que la utilitzi, no importa quin sigui el nombre de columnes clau involucrades. Així les actualitzacions representades a la Figura 8-1 necessiten 1 sentència SQL d'actualització per cada regla que utilitzi cada taula. La solució que donaria el programari per les regles requerides és :

Regla 1 per CT_taula1 :

RECORD_TYPE	SYSTEM_ID	LEVEL	S_FIELD	T_FIELD	RULE	VALUE
*	*	0	MATNR	MATNR	CR	CT_taula1[MATNR]

Sentencia SQL d'actualització

```
UPDATE ( [CT_taula1] AS C INNER JOIN [Dades_d'entrada] AS I ON (C.[MATNR] = I.[MATNR]) )
INNER JOIN [Dades_de_sortida] AS O ON I.idID = O.idID SET O.[MATNR] = C.NEW
```

Regla 2 per CT_*taula2* :

RECORD_TYPE	SYSTEM_ID	LEVEL	S_FIELD	T_FIELD	RULE	VALUE
*	*	0	camp1	camp1	CR	CT_taula2[camp1;camp2]

Sentencia SQL d'actualització

```
UPDATE ( [CT_taula2] AS C INNER JOIN [Dades_d'entrada] AS I ON (C.[camp1] = I.[camp1])
AND (C.[camp2] = I.[camp2])) INNER JOIN [Dades_de_sortida] AS O ON I.idID = O.idID SET
O.[camp1] = C.NEW
```

Figura 8-3: Sentències SQL per regles amb taules CRT sense comodins

Mentre que per fer les actualitzacions representades a la Figura 8-2 es necessiten 1 sentència SQL d'actualització per cada taula. La solució del programari requereix una única regla per taula però es generen més d'una sentencia SQL d'actualització

Regla 1 per CT_*taula1* :

RECORD_TYPE	SYSTEM_ID	LEVEL	S_FIELD	T_FIELD	RULE	VALUE
*	*	0	MATNR	MATNR	CR	CT_taula1[MATNR]

Sentencies SQL d'actualització :

1	UPDATE ([CT_taula1] AS C INNER JOIN [Dades_d'entrada] AS I ON (C.[MATNR] = I.[MATNR])) INNER JOIN [Dades_de_sortida] AS O ON I.idID = O.idID SET O.[MATNR] = C.NEW WHERE (C.[MATNR] <> '*')
2	UPDATE [CT_taula1] AS C, [Dades_d'entrada] AS I INNER JOIN [Dades_de_sortida] AS O ON I.idID = O.idID SET O.[MATNR] = C.NEW WHERE (C.[MATNR] = '*')

Regla 2 per CT_taula2 :

RECORD_TYPE	SYSTEM_ID	LEVEL	S_FIELD	T_FIELD	RULE	VALUE
*	*	0	camp1	camp1	CR	CT_taula2[camp1;camp2]

Sentencies SQL d'actualització :

1	UPDATE [CT_taula2] AS C, [Dades_d'entrada] AS I INNER JOIN [Dades_de_sortida] AS O ON I.idID = O.idID SET O.[camp1] = C.NEW WHERE ((C.[camp1] = '*') AND (C.[camp2] = '*'))
2	UPDATE ([CT_taula2] AS C INNER JOIN [Dades_d'entrada] AS I ON (C.[camp2] = I.[camp2])) INNER JOIN [Dades_de_sortida] AS O ON I.idID = O.idID SET O.[camp1] = C.NEW WHERE ((C.[camp1] = '*') AND (C.[camp2] <> '*'))
3	UPDATE ([CT_taula2] AS C INNER JOIN [Dades_d'entrada] AS I ON (C.[camp1] = I.[camp1])) INNER JOIN [Dades_de_sortida] AS O ON I.idID = O.idID SET O.[camp1] = C.NEW WHERE ((C.[camp1] <> '*') AND (C.[camp2] = '*'))

Figura 8-4: Sentències SQL per regles amb taules CRT amb comodins

El número que acompanya les sentències SQL reflexa l'ordre en el que s'han d'aplicar per tal d'aconseguir el resultat desitjat, tenint en compte que les coincidències amb valors específics han de prevaldré sobre les aconseguides amb comodins. Així es possible que els valors a la taula de destí s'actualitzin per totes les sentències SQL que donen solució a una regla.

De forma paral·lela s'ha solucionat el problema que suposa l'ús de valors nuls en les claus de cerca de les taules CRT. En la solució final, la rutina que genera dinàmicament les sentències SQL per les CRTs combina la informació sobre la distribució de valors concrets, comodins i nuls per clau de cada taula CRT i genera tantes sentències SQL com combinacions diferents es donin.

9 CONCLUSIONS.

Com TFC eminentment pràctic, el desenvolupament d'aquest programari ha comportat més del 98% del temps dedicat. I això ha anat en detriment de la seva documentació, que haurà de ser completada el més aviat possible, donada la rellevància i ús que tindrà (segons apunten els responsables de projectes)

Els objectius inicials van quedar en un segon pla quan amb la primera aproximació de resolució es va verificar que si bé es podia mecanitzar el procés de transformació i fer-lo repetitiu, el temps que necessitava era tan dilatat que els usuaris van tornar a fer transformacions "a mà".

Des de llavors "*El Objectiu*" ha estat construir un enginy amb un rendiment tal que el fessin una "eina pràctica per la transformació"

La feliç idea de convertir cada regla de transformació en una sentència SQL per no haver de fer recorreguts iteratius, i la no menys afortunada solució donada al manejament dels valors comodins i nuls (o blancs) en les claus de taules de referència creuada, han estat els fonaments de la solució definitiva.

La part negativa ha estat que han quedat coses por finalitzar per manca de temps, començant per la documentació i acabant per funcionalitats de tots tipus, encara que el nucli del programari es totalment operatiu, i ja fa setmanes que s'està utilitzant productivament. Tampoc s'ha pogut complir la planificació incial.

En un principi es van també valorar la possibilitat d'utilitzar eines ETL, però es va decidir que la creació d'una eina ad hoc era la millor opció per aconseguir la flexibilitat que la complexitat de la situació requeria i per poder disposar d'una solució productiva en un mínim període de temps. Per sort el fiasco inicial s'ha pogut superar i finalment, podem dir que no va ser una decisió equivocada.

Són moltes les idees sorgides durant el desenvolupament del programari per tal de completar-lo i millorar-lo. A continuació es descriuen les mes rellevants.

9.1 Línies de Treball

Són molts els aspectes en que el programari es pot desenvolupar. S'han enumerat les més urgents, però podrien afegir-se un munt més.

9.1.1 Noves Regles de transformació

A més de les regles pendents d'implementar, totes elles sorgides dels usuaris que fa servir el programari des de el primera versió, també s'han apuntat noves regles com producte de l'experiència durant el desenvolupament:

Codi Regla : **nom anglès** **(nivell de regla)**	**Descripció** **Breu**
DD : Delete Duplicate Records (nivell registre)	Dirigida a la fase de pre-procés haurà d'esborrar els registres duplicats.
GR : Group Records (nivell registre)	Permetrà agrupar registres, definint per quines columnes es fa l'agrupació i per quines altres es duen a terme operacions d'agrupació com sumar, comptar, etc. Una aplicació pràctica : al agrupar materials en un mateix nou codi, també serà necessari agrupar el registres amb el mateix nou codi de material en un únic registre en el qual el camp de vendes haurà d'acumular les que hi havien per cada codi vell
VF : Validate Field (nivell camp)	Validació de camp : aquesta es una regla especial, orientada a la creació d'un informe amb valors que NO s'hi troben entre els especificats a una taula de validació (llista positiva). (pendent implementació)

9.1.2 Millora d l'entorn gràfic per l'execució del programari

El programari actual compte amb un simple formulari que permet recorre el contingut de cada conjunt de paràmetres predefinits i executar el programari per que aquell que s'estigui visualitzant (botó amb l'avió). Hi ha un segon botó (la mascota dels ulls grans) que executa una funció que revisa tos els conjunts de paràmetres i fa una execució bàsica del programari de transformació amb cada conjunt de paràmetres que estigui marcat com seleccionat (taula "*UserInterfaceSetList*", valor "*ParameterSetSelected*").

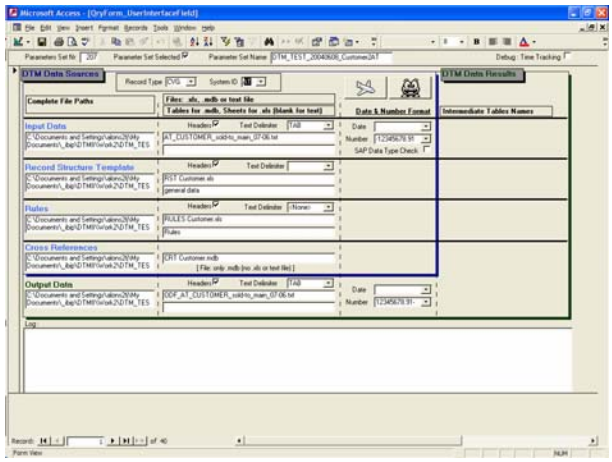


Figura 9-1: Formulari de selecció paràmetres i execució del programari

Però aquest formulari es part d'una solució temporal e inacabada (la finestra al peu amb títol "*Log:*" no es utilitzada en cap cas). I se eliminarà en substituir-la per una altre formulari en que tan sols ha de apareixia la informació relativa als conjunts de paràmetres per tal que permetin l'execució d'un d'ells, o la dels que prèviament es marquin per ser executats. En aquesta nova solució, la informació detallada continguda en cada conjunt de paràmetres es podrà consultar i editar a través del formulari detallat en l'apartat següent.

9.1.3 Creació d'un entorn gràfic per la edició dels paràmetres

Fins al moment, l'entrada de paràmetres per l'execució del programari. tan sols està "recolzada" per una parell de consultes :

- "*QryCreate_NewUserSet_from_existing*" : per copiar un conjunt de paràmetres sobre una altre.
- "*QryUserInterfaceSet_Nr*" : per editar un determinat conjunt de paràmetres, acompanyant cadascú d'ells de la seva descripció.

A més, la creació d'un nou conjunt s'ha de fer directament a la taula on son definits : "*UserInterfaceSetList".*

Per tot això és prioritari afegir un entorn d'entrada de dades que simplifiqui aquesta tasca, ja que els usuaris finals qualifiquen de difícil aquest procés.

Aquesta tasca ja es troba bastant avançada : s'ha fet l'anàlisi del conjunt de paràmetres i també el disseny; i a més està pendent acabar l'implementació. El formulari de entrada de dades ja està dissenyat i té el nom "*FrmParameterSetList*", i manca modificar el programa per que recuperi el paràmetres de la nova estructura que li dona suport.. aquest es el seu aspecte:

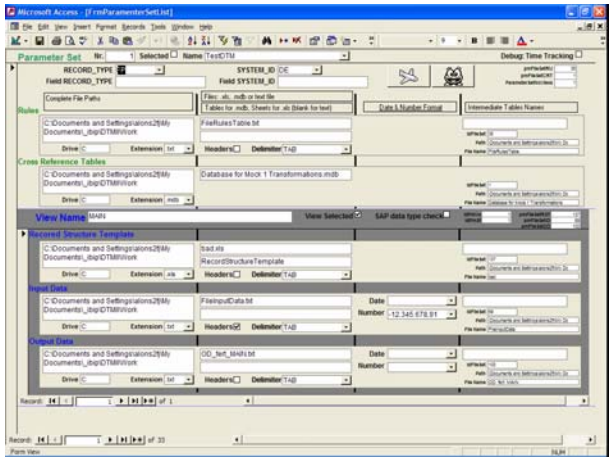


Figura 9-2: Formulari per l'entrada de paràmetres

Aquest formulari té botons una parell de botons, l'avió per executar la transformació del conjunt de dades amb la vista que es visualitza, que correspon a una execució bàsica del programari de transformació, i altre botó (en el que apareix una espècie de mascota) que llançaria una funció que faria una execució bàsica del programari de transformació per cada una de las vistes del conjunt de dades seleccionat.

De tota manera, quan estigui desenvolupat l'entorn d'execució, comentat en l'apartat següent, aquest botons desapareixen i es posa un de nou que simplement posi en marxa el formulari d'execució.

S'ha de dir que el conjunt de taules que donen suport als paràmetres en la versió del programa producte d'aquest TFC va ser una solució temporal. Ara ja s'han creat una estructura de taules per donar suport els paràmetres, en las que es basa aquesta solució per la edició dels paràmetres, sent aquestes taules les següents: *"ParameterSetList, ParameterViewList, ParameterFileSet, i ParameterSetViewRunning"*

9.1.4 Creació d'un subsistema de control de errors

El programari actual fa tota una sèrie de comprovacions per tal d'informar a l'usuari de possibles errors en la codificació de la informació de transformació, i o en les dades d'entrada. De vegades però el programari s'atura amb un error que encara no està cobert per una rutina de control d'errors.

Es necessita dotar al programari de les funcionalitats necessàries per tal de que les rutines de control d'errors cobreixin tots els possibles errors, incloent la preparació d'informes d'error, distingint aquells que s'han de quedar en un simple avís, i aquells que ha de forçar l'avortament del procés de transformació.

L'anàlisi de les necessitats de control d'errors ja s'ha començat a reflexar en taules creades per donar suport a les dades relacionades: "*LogInformation, Error, ErrorReportType i LogMessage*". Manca ara el gros de la feina d'implementar en el codi del programari, l'ús d'aquestes estructures de dades i la creació del codi per donar suport a la consulta i presentació dels informes d'errors.

9.1.5 Creació d'un entorn gràfic per la creació i edició de les dades de transformació

Després de la implementació de les millores indicades en l'apartat anterior s'ha considerat de molta utilitat la possibilitat de que la actual edició de regles i taules de referència creuada es pogui fer o completar des de dins del programari. D'aquesta manera es poden eliminar els errors mecànics sorgits per introduir malament el nom d'un camp, d'una taula, d'una funció, o be d'escriure malament l'argument d'una funció. Aquesta opció no substituiria en cap cas l'actual possibilitat de crear i manipular aquesta informació.

9.1.6 Millora de les sentencies d'actualització SQL

A mig analitzar hi han millores en el que respecte a la generació de les sentències SQL.

El problema de la cerca per valors nuls

La primera està relacionada amb la problemàtica que presenta treballar amb valors nuls, ja que podem verificar si un valor és nul, però no podem verificar si dos variables, o el contingut des dos camps son "iguals" quan al menys una d'elles és igual al valor nul. El petit codi a continuació il·lustra aquest problema.

Codi :

```
Function testNull_Behavior()

    Dim var1, var2, var3 As Variant

    var1 = Null
    var2 = Null
    var3 = "test"
    Debug.Print _
        "IsNull(var1)", "IsNull(var2)", "(var1 = var2)", "(var1 = var3)"
    Debug.Print _
        IsNull(var1), IsNull(var2), (var1 = var2), (var1 = var3)

End Function
```

Resultat de l'execució:

```
IsNull(var1)  IsNull(var2)  (var1 = var2) (var1 = var3)
True          True          Null          Null
```

Figura 9-3: Codi font per valors nuls

La solució passa per substituir els valors nuls per cadenes buides. Llavors si es pot comprovar si dues variables "en blanc" son iguals com es veu a continuació:

Codi :

```
Function testBlank_Behavior()

    Dim var1, var2, var3 As Variant

    var1 = ""
    var2 = ""
    var3 = "test"
    Debug.Print _
        "IsNull(var1)", "IsNull(var2)", "(var1 = var2)", "(var1 = var3)"
    Debug.Print _
        IsNull(var1), IsNull(var2), (var1 = var2), (var1 = var3)

End Function
```

Resultat de l'execució:

```
IsNull(var1)  IsNull(var2)  (var1 = var2) (var1 = var3)
False         False         True          False
```

Figura 9-4: Codi font per valors cadenes en blanc

El següent pas és modificar la rutina de càrrega des del fitxer a la taula de dades d'entrada per tal de que una cadena en blanc, o un valor nul quan es recuperi la informació des d'altre taula, esdevinguin valors blancs a la taula de dades d'entrada. Després s'haurà d'aplicar el mateix raonament per el contingut de les claus de cerca de les taules de referència creuada. Així finalment no serà necessari tenir sentències SQL especials per el valors nuls.

Tots dos codis anterior es troben al mòdul *ModTFCTools* del producte del TFC.

El problema de la cerca per els comodins.

A les taules de referència creuada els valors comodí comporten la creació de sentències de actualització SQL addicionals per manipular-los. L'idea és explorar l'ús de l'operador LIKE en substitució de l'operador =, el qual permet l'ús de comodins.

9.1.7 Eliminació de la creació de copies de taules CRT

En la solució actual el contingut de les taules CRT es filtrat i copiat en una taula dins de la base de dades que és el producte d'aquest TFC.

Les últimes modificacions fetes al programari, que utilitzen dues consultes enllaçades i creades dinàmicament per fer el filtrat, han obert la porta a la possibilitat de que finalment, durant la transformació, la sentències SQL d'actualització utilitzin una consulta creada prèviament sobre la taula CRT original per tal d'utilitzar la taula CRT creada localment com a copia filtrada de l'original.

Tècnicament, la solució és clara, però s'ha d'implementar i fer proves de rendiment avanç de donar-la per vàlida, ja que preparar una taula CRT de forma local comporta un temps que depenent de la freqüència amb que es modifiquin les dades i amb la que s'utilitzi per fer les transformacions pot tenir sentit deixar-la preparada en associació a cada conjunt de dades per transformar o no.

9.1.8 Incorporació a les dades d'entrada del tipus de registre

Els usuaris han demanat poder transformar fitxers d'entrada de dades en els que la informació sobre el tipus de registre ja es trobi codificada a cada entrada. La solució ja es troba recollida en la nova interfície d'usuari avança esmentada: es podrà seleccionar un tipus de registre especial "*<field>*", que obrirà la possibilitat de seleccionar el nom del camp del fitxer de dades de entrades d'on

s'haurà de recollir el tipus de registre.

L'adaptació del programari passa per separar les dades d'entrada en subconjunts amb un mateix tipus de registre, i realitzar successives execucions del programari per cada subconjunt.

Per si de cas, la mateixa funcionalitat, també s'implementarà per el filtre de sistema.

9.2 Opinió Personal

El producte obtingut s'ha convertit de fet en una peça clau dins del projecte de migració posat en marxa per l'empresa.

Penso que una vegada finalitzades las millores esmentades anteriorment, seria molt interessant traslladar el programari a un entorn distribuït, en el que l'únic programari necessari per utilitzar-lo fos un navegador Web, i en el que les dades es ponguessin manegar independentment que estiguessin en un únic SGDB o varis.

La proposta inicial d'aquesta solució, feta per Alejandro Sánchez, un company de feina i també d'estudis a la UOC (i per damunt de tot un amic) va aportar un mètode perquè un munt de persones tinguessin una forma fàcil d'especificar les transformacions i posar-les en comú.

La primera versió del programa, basada en recorreguts del fitxer d'entrada, el de regles, etc. va demostrar que l'idea es podia dur a terme.

El greus problemes de rendiment de la primera versió em van posar al davant un repte que em va engrescar des del primer moment, donant-me l'oportunitat de "crear i programar" una solució que finalment ha estat "la bona" i de la que soc autor al 100%. Ha estat un repte amb el que he gaudit, ja que ha el resultat és una eina que està tenint un important ús pràctic ja en l'actualitat, i que he pogut desenvolupar gràcies als coneixements adquirits en aquests any d'estudi a la UOC, sent aquest al cap i a la fi el propòsit final que em va moure a començar el estudis en el grup pilot de ETIG a la UOC : aprendre per després utilitzar l'*enginy* per resoldre problemes a la vida real.

10 GLOSSARI.

Terme	Definició
ABAP	*Advanced Business Application Programming*, llenguatge de programació emprat per l'ERP SAP
CT	Cross reference Table : codi de la regla bàsica que utilitza taules de referència creuada, les quals, dins d'aquest programari són molt importants. Quan aquest tipus de taula són carregades dins de la base de dades del programari s'afegeix al nom original aquest prefix. Guarden la informació que relaciona els valors d'entrada amb els seus corresponents valors de sortida i per tant son base de la majoria de transformacions.
CRT CRTs	Cross Reference Table/s : sigles per referir a una d'aquestes taules o a la base de dades externa al programari que las conté, i des de qual la que es carrega la informació per les taules CT_. També és el prefix per defecte del fitxer que conté les taules.
DTM	Data Transformation Machine : Maquina de transformació de dades, nom amb que es va batejar el programari producte del TFC.
ERP	Enterprise Resource Planning, eina
IDF	Input Data File : Fitxer de Dades d'Entrada. També el prefix per defecte d'aquest fitxer.
ODF	Output Data File : Fitxer de Dades d'Sortida. També el prefix per defecte d'aquest fitxer.
RST	Record Structure Template : Plantilla de Estructura de Registre, sigles utilitzades per referir la informació que defineix el format de les dades d'entrada i sortida. També el prefix per defecte del fitxer que conté aquesta informació.
RUL	Rules : Regles, sigles utilitzades per referir la informació de la transformació. També el prefix per defecte del fitxer que conté aquesta informació.
SAP	Programari líder en el mercat de les eines ERP.
SGDB	Sistema de Gestió de Bases de Dades.

Terme	Definició
Soft Coding	Nom donat a l'estil de programació que mitjançant l'ús exhaustiu de taules evita al màxim haver de codificar, dins del codi font, valors i/o estructures. Per exemple, si els valors "separadors de camps de text" es posen en una taula, i el programari es prepara per que els recuperi d'aquesta taula, es podran afegir, canviar, o eliminar separadors modificant l'esmentada taula i sense haver de modificar el codi del programa.
SQL	Structured Query Language.
VBA	Visual Basic for Applications, llenguatge de programació de Microsoft imbuït en les seves aplicacions d'oficina, i també a Microsoft.

11 BIBLIOGRAFIA.

11.1 Llibres

- [1] Ramón Sangüesa i Solé. *Data mining. Una introducció.* 2000. Fundació per a la Universitat Oberta de Catalunya.
- [2] Jaume Sistac Planes. *Bases de dades I*. 1999. Fundació per a la Universitat Oberta de Catalunya.
- [3] Jaume Sistac Planes. *Bases de dades II*. 1999. Fundació per a la Universitat Oberta de Catalunya.
- [4] M.Jesús Marco Galindo. *Enginyeria del programri II.* 2000. Fundació per a la Universitat Oberta de Catalunya.
- [5] Ralf Albrecht, Natash Nicol. *Desarrollo de Soluciones de Accesss con Microsoft SQL Server.* McGraw-Hill Profesional
- [6] John Viescas. *Quick Reference Guide to SQL*. 1990. Versió en castellà de Ediciones Anaya Multimiedia

11.2 Adreces http consultades

- http://www.datashape.net/
- http://www.kdnuggets.com/
- http://www.sql-server-performance.com/
    - http://www.sql-server-performance.com/ec_data_mining.asp
- http://www.hummingbird.com/
    - http://www.hummingbird.com/products/etl/overview.html?cks=y
    - http://www.hummingbird.com/products/etl/index.html
- http://www.cognos.com/products/decisionstream/architecture.html
- http://www.computerworld.com/databasetopics/businessintelligence/datawarehouse/story/0,10801,89534,00.html
- http://www.informationbuilders.com/products/webfocus/pdf/Data_Mgmt_Solutions.pdf

12 ANNEXES.

12.1 Abreviatures per tipus de camp

En aquest document s'han utilitzat les següents abreviatures per referir-se als diferents tipus de camp utilitzats a les taules de dades :

Tipus de camp	Descripció
Text(n)	Cadena de entre n = 1 i n = 255 caràcters, en Access correspon al tipus *Text.*
AutoN	Nombre generat automàticament per crear un índex únic per cada registre de la taula, en Access correspon al tipus *AutoNumber.*
Bol	Valors lògics Vertader o Fals, en Access correspon al tipus *Yes/No.*
Num(n,m)	Valors numèrics, on n i m son opcionals, on n representa el nombre de dígits totals i m els decimals, en Access es tradueix al tipus *Number*
Memo	Camp de text lliure, Access dona suport de fins 65.535 caràcters per aquest tipus de camp.

12.2 Altres taules auxiliars

S'ha deixat per aquest annexa la descripció d'algunes de les taules auxiliars que o bé no son les principals, o bé estan presents en el programari però la versió producte del TFC no fa ús.

Nom de la taula Informació continguda	Descripció
Application Aplicatius	Inventari diferents aplicacions existents. No utilitzada en la present versió.
Països Country	Codificació de països tal i com apareixen a l'ERP SAP
RuleType Tipus de Regla	Us intern programari. Les regles s'agrupen en tres tipus: relacionades amb taules de referència creuada (C), les que utilitzen una funció creada en VBA (F), i les que utilitzen el valor escrit directament a la regla (V).
CalculationFunctions Funcions de Càlcul	Us intern programari. Porta l'inventari de funcions creades per se utilitzades per una regla de càlcul o tipus (F).
FieldType_ABAP4_data_type Tipus de dades ABAP	No utilitzada encara.Us intern del programari. Conté informació sobre els tipus de dades en ABAP
FieldType_Mapping_external_to_ABAP4 Tipus de dades ABAP i SAP	No utilitzada encara, Us intern del programari. Conté informació sobre la relació dels tipus de dades en ABAP i el tipus de dades SAP.
FieldType_Mapping_for_SQL_Server Tipus de dades SAP i SQL	No utilitzada encara. Us intern del programari. Conté informació sobre la relació dels tipus de dades en SAP i el tipus de dades SQL.
UserInterfaceField *Definició paràmetres*	Us intern del programari. En la actual versió, defineix els diferent paràmetres emprats per el programari.
UserinterfaceParameterGroup Agrupació de paràmetres	Us intern del programari. En la actual versió, defineix els diferent grups del paràmetres emprats per el programari.
ParameterGroup Agrupació de paràmetres	Us intern del programari. Per la propera versió del programari, defineix els diferent grups del paràmetres emprats per el programari.

Nom de la taula Informació continguda	Descripció
ParameterLevel Nivell del paràmetres	Us intern del programari. Per la propera versió del programari, defineix els diferent nivels del paràmetres emprats per el programari.
ParameterFileType Tipus fitxer	Us intern del programari. Per la propera versió del programari, defineix els diferent tipus de fitxer suportats per els paràmetres.
ParameterFileSetDefaults Valors per defecte per paràmetres per fitxers	Us intern del programari. Per la propera versió del programari, defineix els tipus de fitxer, els noms, i altres detalls que defecte ds'0asignaran a cada grup de paràmetres relacionat amb els fitxers de dades externs.
LogInformation *Error* *ErrorReportType* *LogMessage*	Taules per l'ús intern del programari. Per propera versió del programari, definiran els tipus de missatges d'error, que després el programari utilitzarà per generar els informes d'errors d'execució.