ESPECIFICACIÓ DEL FORMAT "DBF ESTESA"
(INCLOU LA IMPLEMENTACIÓ "NOMS ESTESOS")

Autors del document: Xavier Pons i Abel Pau
Proposta inicial: 23-12-2011
Darrera modificació i versió: 25-11-2018. **1.5**

Nota: Si existeix la paraula "PENDENT" en aquest document indica aspectes que encara no han estat implementats

1. Antecedents i motivació.

El **format DBF**, inicialment desenvolupat en el context del programari dBASE i seguit en altres *softwares* com FoxBase i FoxPro, i descrit en diversos llocs a la literatura especialitzada i a Internet, és, exceptuant les formes tabulars en text pla, potser el més popular dels formats de taules alfanumèriques de dades. El **MiraMon** pot accedir a dades tabulars en format DBF III, III+ i IV, així com també a dades tabulars (físiques o resultat d'expressions de consulta o càlcul) contingudes en altres formats de fitxers (com ara XLS, MDB, etc) i en bases de dades (com ara Oracle, SQL-Server, etc); en aquests darrers casos caldrà que existeixin i estiguin instal·lats els corresponents *drivers* ODBC i, amb l'excepció de l'MDB (que admet accés directe des del MiraMon), que es creï un fitxer DSN per a accedir a les dades.

Malgrat el gran potencial que presenta l'accés a les diferents fonts tabulars esbossades més amunt, en el context MiraMon sovint DBF és el format de taules d'elecció, tant per la seva senzillesa i velocitat, com pel fet que no cal dependre de tercers, o simplement perquè es vol tenir la certesa que es podrà obrir la informació en un ordinador del qual desconeixem quins *drivers* té instal·lats (motiu pel qual és l'opció per defecte en la creació de fitxers MMZ destinats a una àmplia difusió per Internet).

Cal recordar també que les taules DBF poden ser enllaçades entre si i amb altres taules (MDB, SQL-Server, etc) tot establint enllaços (*join*) en què és possible indicar la cardinalitat (1 a 1, 1 a molts, etc) i el nivell d'obligatorietat de la resolució de l'enllaç (tipus diccionari o no) en un conjunt de relacions que no està limitat ni en nombre de relacions per camp, ni en nombre de nivells de relacions. Aquests enllaços s'emmagatzemen en els corresponents **fitxers REL**.

Tanmateix, el format DBF, per la seva antiguitat, presenta una sèrie de limitacions. L'evolució del format DBF després de la versió IV va ser força complicada (venda del producte entre diferents empreses, estranyes decisions com la inclusió de nous tipus numèrics que no aportaven res de significatiu, etc [vegeu http://en.wikipedia.org/wiki/DBase]), per la qual cosa en general la comunitat d'usuaris i desenvolupadors continua fidel als formats III, III+ i IV tant per a lectura com per a exportació. Actualment hi ha encara un producte dBASE a la venda (http://www.dbase.com).

Algunes de les limitacions del DBF IV han estat superades gràcies a indicacions que també s'emmagatzemen en els fitxers REL i s'estableixen des del **Gestor de Metadades i Relacions del MiraMon**, **GeMM**. Algunes de les més destacades són:

- La longitud del nom del fitxer, que en l'especificació original era 8+3, i que en el MiraMon s'amplia a qualsevol longitud suportada pel sistema operatiu. A més, en el MiraMon els noms dels fitxers DBF i els directoris on són poden contenir espais, accents, etc.
- La longitud del nom de cada camp DBF està limitada a 10 caràcters que, a més, no poden contenir lletres accentuades, espais, caràcters especials, etc. En canvi, des del GeMM es pot indicar un descriptor de text lliure, sense limitacions d'accents, caràcters especials, etc, per a cada camp; a més, el descriptor pot ser multiidiomàtic si es desitja i admet una extensió, en caràcters, de:

  ```
  #define MAX_LON_DESCRIPCIO_CAMP_DBF max(_MAX_PATH+100,256)
  ```
- En les taules anteriors a dBASE IV el joc de caràcters utilitzat en els camps de tipus 'C' no estava especificat. El MiraMon permet una solució flexible i configurable per a aquests casos i, com en les taules dBASE IV, admet indicar el joc de caràcters, amb la qual cosa la interpretació d'accents i caràcters especials deixa de ser ambigua.
- En els camps amb contingut numèric no es pot indicar les seves unitats quan en tenen. En canvi, des del GeMM aquestes poden ser especificades, així com si es desitja que siguin mostrades en les consultes.
- En les taules DBF no es pot indicar la qualitat del contingut de cada camp. En canvi, des del GeMM aquesta pot ser especificada.
- En les taules DBF no es pot indicar el tractament (categòric, ordinal o quantitatiu continu) del contingut de cada camp. En canvi, des del GeMM aquesta propietat també pot ser especificada.

A pesar d'aquestes importants extensions introduïdes a través dels fitxers REL (de les quals també es poden beneficiar altres formats tabulars llegibles des del MiraMon), existeixen altres limitacions del format DBF que no poden ser resoltes excepte amb la

introducció de modificacions en el propi format. Entre les més importants podem destacar:

- La limitació del nombre de camps a 128 en dBASE III+ (llibre de dBASE de Jordi Abadal, p. 137 i criteri d'exportació de Excel 2000) o 255 (dBASE IV). Aquesta limitació no només pot ser trobada en taules de tota mena, sinó que encara és més freqüent en la taula única creada en resoldre tots els enllaços de l'arbre de relacions especificat des del GeMM quan les relacions són nombroses i especialment vinculades amb taules amb molts camps. Finalment, noteu que el MiraMon no estableix aquesta distinció entre versions dBASE III+ o dBASE IV i assumeix, per a qualsevol DBF clàssica, independentment de la versió:
    - #define MAX_N_CAMPS_DBF 255
- La limitació del nombre de caràcters, en camps de tipus 'C':
    - #define MAX_AMPLADA_CAMP_C_DBF_CLASSICA 254
- La limitació del format i longitud del nom dels camps de la DBF a 10 caràcters en majúscules (sense suport a lletres accentuades, ç, etc).
    - #define MAX_LON_NOM_CAMP_DBF_CLASSICA 11

Hi ha d'altres millores possibles a esmentar que podrien ser objecte de discussió sobre l'oportunitat del seu disseny i implementació en un futur:

- Els nombres s'escriuen en realitat com a text que cal interpretar. Seria desitjable definir tipus numèrics binaris seguint els estàndards habituals per a enters i reals.
- Suport a camps de text de longitud variable amb un sistema d'indexació del fitxer que diu on comença cada registre (tot i que això faria una mica més lent l'accés).
- Suport a camps binaris de longitud il·limitada.
- Suport a compressió individual per registre amb un sistema d'indexació del fitxer que diu on comença cada registre.
- Suport a una marca d'esborrat d'un camp (columna).
- Optar per una descripció en text de la capçalera.
- La primera "minicapçalera" de 32 bytes podria passar a ser extensible per encabir ampliacions futures. Això implica codificar la mida de la pròpia minicapçalera.
- La descripció de cada camp podria ser extensible. Això implica codificar a la minicapçalera la mida de la descripció d'un camp.
- El nombre de registres podria ser un **unsigned __int64** (enter de 64 bits), , però també un "__int40" (o sigui, destinar 5 bytes del disc a definir la grandària. Detalls a continuació:
    - Es proposa que en la propera actualització de les llibreries del format, que probablement no implicaria ni una versió 0x91, s'implementi això. Com es pot veure al LOCUS_N_REGISTRES d'aquest document, hi ha un lloc a la capçalera d'un fitxer DBF on s'emmagatzema el nombre de registres com un unsigned __int32, i en el codi s'actualitza per exemple a ActualitzaNFitxesDBF(). Si volguéssim que en el disc s'emmagatzemés un enter de 64 bits sense signe necessitaríem buscar en la capçalera 4 bytes, ni que fossin esparsos, per a escriure els 4 bytes addicionals que

ens permetrien arribar a un unsigned __int64. Tanmateix, si mirem la següent taula (disponible a DBF_estesa_Calcul_NRegistres_i_Grandaria_v1.xlsx):

Tipus en llenguatge C	bytes	bits/valor	milions de registres possibles	Grandària aprox. d'un fixer, en Tbyte, amb 100 bytes per registre
unsigned __int32	**4**	32	4 295	0.4
	5	40	1 099 512	100.0
	6	48	281 474 977	25 600.0
	7	56	72 057 594 038	6 553 600.0
unsigned __int64	**8**	64	18 446 744 073 710	1 677 721 600.0

veiem que una altra possibilitat fóra que, ni que en l'estructura en memòria usem un enter de 64 bits sense signe, en el format en disc no destinem tants bytes a desar quants registres podem tenir. Concretament proposo usar un enter de 5 bytes en el disc, cosa que ens permetria anar fins al bilió (milió de milions) de registres i que faria que un fitxer de 100 bytes per registre ocupés, ell tot sol, 100 Tbyte, cosa que sembla més que suficient per uns quants anys. El byte suggerit per a aquesta ampliació és el marcat amb LOCUS_CINQUE_BYTE_N_REGISTRES, ja que no hem trobat exemples en què tingui un valor diferent a 0.

- La data interna del fitxer es podria eliminar. Sempre ha estat un maldecap i és redundant amb la del sistema de fitxers. A més, no conté l'hora i, doncs, no és útil ni per a restaurar la data-hora exacta d'un fitxer que s'ha enviat per correu electrònic i que ha perdut la data.
- Es podria introduir un camp de tipus data-hora.

L'equip del MiraMon ha decidit solucionar les més importants d'aquestes limitacions, tot establint una variació del format DBF que anomenarem "**DBF estesa**".

2. Característiques i utilització del format "DBF estesa".

2.1 Si una taula no necessita superar les limitacions de la DBF clàssica, és preferible escriure-la en aquest format a fi que sigui llegible per altres *softwares* que no suportin la DBF estesa. Noteu que el format no manté la compatibilitat descendent (una DBF estesa no pot ser llegida, ni parcialment, per un *software* que llegia DBF clàssiques).

2.2 L'**extensió** del fitxer és **.dbf** com en la DBF clàssica.

2.3 El **primer byte** és **0x90**. Per comprovar si el software pot llegir això, només comprova el '9', el que permet que canviï el segon número per modificacions compatibles descendents. No utilitzem valors menors per no entrar en conflicte amb altres numeracions que podeu trobar a l'Annex 2.

2.4 El **nombre de camps** possible passa a ser de 13.4 milions (el valor màxim exacte es justifica en el següent apartat).

2.5 La **longitud d'un registre** és un **unsigned __int32**, comptant el byte d'esborrat del registre.

2.6 La **mida d'un camp** C pot arribar a **unsigned __int32-1** (es renuncia a un byte per a encabir el byte d'esborrat).

2.7 El fitxer continuarà contenint la marca de final de definició de camps (*Header Record Terminator* - 0x0D) com en les DBF clàssiques ja que en el codi caldria fer una bifurcació arreu on es calcula OffsetPrimeraFitxa i es considera massa crític el problema que es generaria entre DBF clàssiques i esteses i, doncs, es prefereix sacrificar aquest byte d'emmagatzematge.

2.8 El fitxer NO conté la marca de final de fitxer (0x1A) que sovint es pot trobar en les DBF clàssiques.El fitxer contindrà els noms estesos de camps entre la marca de final de definició de camps (Header Record Terminator - 0x0D) i la primera fitxa. L'accés a aquest camps es farà a partir d'un *offset* i una mida (els noms estesos no contindran el '\0' final) emmagatzemats en la capçalera dels camps (l'*offset* en el reservat2+7, de longitud 4 bytes, i la mida, en el reservat2+11 de longitud un byte. La màxima longitud d'un nom estès d'un camp seria, doncs, de 255 caràcters, però com es comenta més endavant, acaba essent de 128 caràcters, suficient en comparació amb el que suporten altres gestors de grans bases de dades (Oracle 9.i en suporta 30, i SQL Server 2000, 128).

2.9 S'avisarà amb un missatge en el cas que una DBF passi de no contenir noms estesos a contenir-los:

El nom del camp que vols crear (%s) té característiques no compatibles amb la definició clàssica dels fitxers DBF (com ara accents, espais, longitud més gran que 10, etc). Recordeu que també podeu generar noms de format lliure des de la casella "Descriptor" o des del Gestor de Metadades.

Desitges utilitzar el nom igualment? [Sí] [No]

2.10 En el cas concret que el nom tingui la longitud clàssica (inferior a 10 caràcters) però tingui alguna lletra minúscula (sense accentuar, i sense presència d'espais ni caràcters especials) s'avisarà i es convidarà a passar-lo a majúscules per a major compatibilitat amb el format clàssic DBF però es tolerarà donat que altres programes que generen DBF clàssiques ho estan fent.

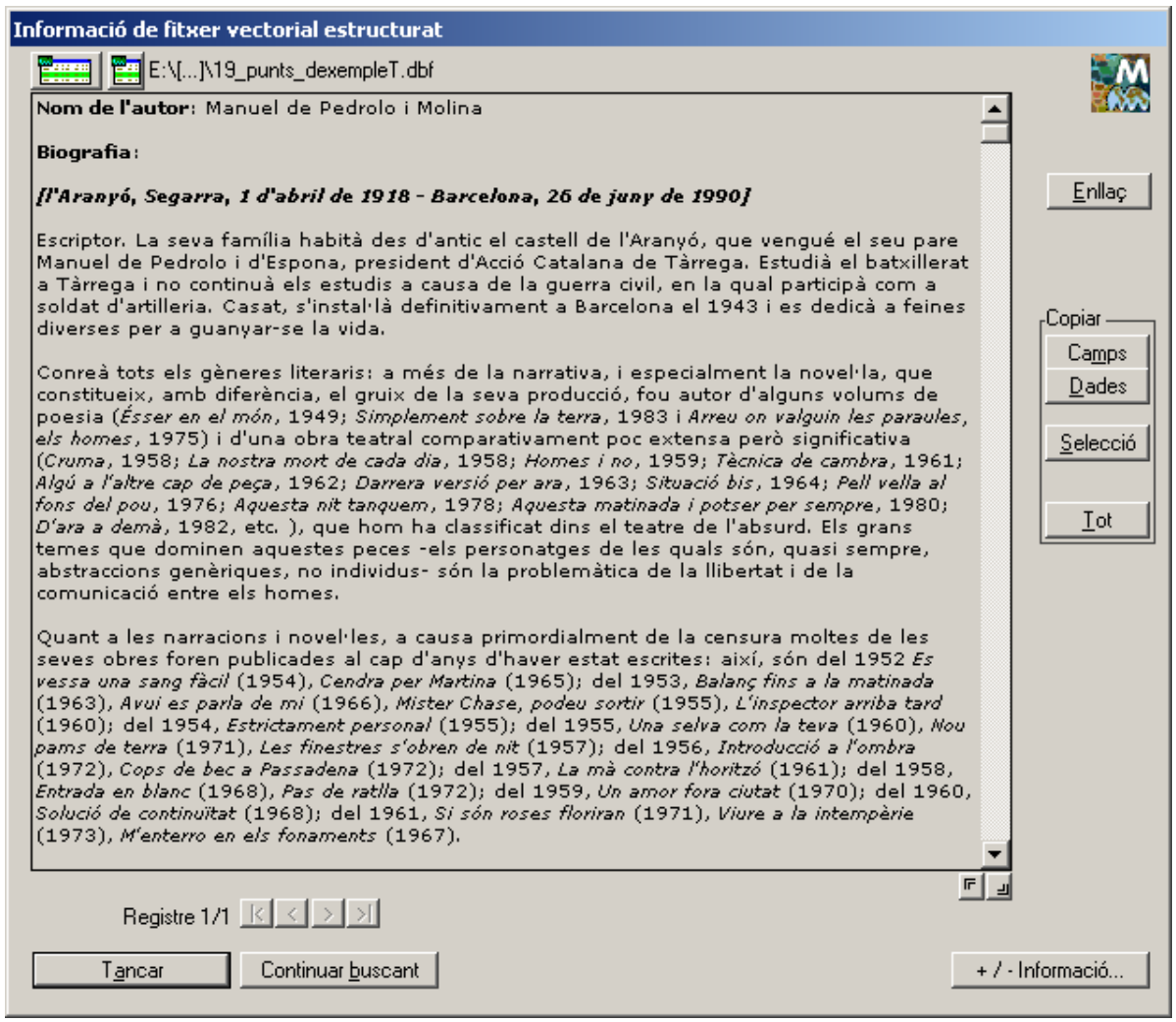
Els camps de C amb longituds esteses permeten posar text especialment llarg i complex, com ara codificació HTML, expressions xpath() per a l'accés a recursos de dades obertes (*open data*), etc, en els camps. En el següent exemple extret d'una base hipotètica on se situa el lloc de naixement de diversos escriptors, la consulta permet accedir a la seva biografia (molt més extensa, uns 100 000 caràcters, que el que apareix a la captura, com podeu deduir per la grandària del botó de la barra de desplaçament vertical), en aquest cas extreta de l'Enciclopèdia Catalana a Internet. Naturalment també es podria afegir una foto, enllaços a altres recursos a Internet o a una intranet, etc.

En l'exemple mostrat, el camp comença així:

```
<HTML><BR><BR><b><i>[l'Aranyó, Segarra, 1 d'abril de 1918 - Barcelona, 26 de juny de 1990]</i></b><BR><BR>Escriptor....
```

i acaba així:

```
...difusió de la seva obra, així com a la del seu ideari.</HTML>
```


Informació de fitxer vectorial estructurat
E:\[...]\19_punts_dexempleT.dbf
Nom de l'autor: Manuel de Pedrolo i Molina
Biografia:
[l'Aranyó, Segarra, 1 d'abril de 1918 - Barcelona, 26 de juny de 1990]
Escriptor. La seva família habità des d'antic el castell de l'Aranyó, que vengué el seu pare Manuel de Pedrolo i d'Espona, president d'Acció Catalana de Tàrrega. Estudià el batxillerat a Tàrrega i no continuà els estudis a causa de la guerra civil, en la qual participà com a soldat d'artilleria. Casat, s'instal·là definitivament a Barcelona el 1943 i es dedicà a feines diverses per a guanyar-se la vida.
Conreà tots els gèneres literaris: a més de la narrativa, i especialment la novel·la, que constitueix, amb diferència, el gruix de la seva producció, fou autor d'alguns volums de poesia (Ésser en el món, 1949; Simplement sobre la terra, 1983 i Arreu on valguin les paraules, els homes, 1975) i d'una obra teatral comparativament poc extensa però significativa (Cruma, 1958; La nostra mort de cada dia, 1958; Homes i no, 1959; Tècnica de cambra, 1961; Algú a l'altre cap de peça, 1962; Darrera versió per ara, 1963; Situació bis, 1964; Pell vella al fons del pou, 1976; Aquesta nit tanquem, 1978; Aquesta matinada i potser per sempre, 1980; D'ara a demà, 1982, etc.), que hom ha classificat dins el teatre de l'absurd. Els grans temes que dominen aquestes peces -els personatges de les quals són, quasi sempre, abstraccions genèriques, no individus- són la problemàtica de la llibertat i de la comunicació entre els homes.
Quant a les narracions i novel·les, a causa primordialment de la censura moltes de les seves obres foren publicades al cap d'anys d'haver estat escrites: així, són del 1952 Es vessa una sang fàcil (1954), Cendra per Martina (1965); del 1953, Balanç fins a la matinada (1963), Avui es parla de mi (1966), Mister Chase, podeu sortir (1955), L'inspector arriba tard (1960); del 1954, Estrictament personal (1955); del 1955, Una selva com la teva (1960), Nou pams de terra (1971), Les finestres s'obren de nit (1957); del 1956, Introducció a l'ombra (1972), Cops de bec a Passadena (1972); del 1957, La mà contra l'horitzó (1961); del 1958, Entrada en blanc (1968), Pas de ratlla (1972); del 1959, Un amor fora ciutat (1970); del 1960, Solució de continuïtat (1968); del 1961, Si són roses floriran (1971), Viure a la intempèrie (1973), M'enterro en els fonaments (1967).
Enllaç
Copiar
Camps
Dades
Selecció
Tot
Registre 1/1
Tancar
Continuar buscant
+ / - Informació...

3. Especificació del format “DBF estesa”.

El **primer byte** és **0x90**. Aquesta és la **marca de DBF estesa**. Millores posteriors podrien implicar numeracions hexadecimals successives.
[Programadors useu MARCA_VERSIO_1_DBF_ESTESA i ES_DBF_ESTESA().]

Els **2 bytes** anomenats **reservat_1** (bytes 12-13) seran llegits conjuntament amb els 10 i 11 com un sol paquet de 4 bytes (*unsigned __int32*) que definirà el **nombre de bytes per registre** (emmagatzemat al codi del MiraMon en el membre **BytesPerFitxa** de l’estructura emprada per a les taules). L’extensió a un enter llarg és necessària per a poder encabir, per exemple, diversos camps C amb una amplada important.
[Programadors useu **sizeof**(TIPUS_BYTES_ACUMULATS_DBF) en cas de llegir una capçalera d’una DBF estesa.]

Els **2 bytes** anomenats **reservat_2** (bytes 30-31) seran llegits conjuntament amb els 8 i 9 com un sol paquet de 4 bytes que definirà **on s’inicia l’emmagatzematge dels registres** (al codi del MiraMon en el membre **OffsetPrimeraFitxa** de l’estructura emprada per a les taules). A diferència d’en la DBF clàssica, on el **nombre de camps de la taula** (emmagatzemat al codi del MiraMon en el membre **ncamps** de l’estructura emprada per les taules) determinava on acabava la capçalera DBF, ara això ja no és així ja que cal preveure que encara hi haurà, després de la marca de final de la capçalera clàssica (*header record terminator*, 0x0D), els noms estesos. El nombre de camps que suporta ara, doncs, la DBF estesa queda condicionat pels 32 bytes destinats a descriure cada camp i a aquest *offset* on comencen els registres, dintre el qual cal incloure, a més de la descripció de cada camp, la capçalera de 32 bytes, el byte final 0x0D que marca el final de la descripció dels camps i els noms estesos. Nota 3.1: En l’estructura de la DBF clàssica OffsetPrimeraFitxa era un enter curt amb signe; per evitar haver de repassar tots els seus usos per si en algun cas se’n feia un ús amb signe s’ha optat per redefinir el membre com a un enter llarg també amb signe (*signed __int32*); això redueix a la meitat el nombre de camps possible, però com que ja és un valor molt gran, no sembla problemàtic i, en qualsevol cas, sempre es podria fer aquesta repassada per a suportar un ús sense signe si arribava a fer falta. Així doncs, de moment queda: (2147483648-32-1)/(32+128)= 13 421 772, o sigui **uns 13.4 milions de camps**. Nota 3.2: Si es continuava emprant només els bytes 8 i 9, el nombre de camps suportats seria 1022 [=(32767-32)/32], que es contempla com a insuficient en el context de dotar-se d’una nova DBF que no quedi limitada fàcilment en prestacions.
[Programadors useu SEGON_OFFSET_a_OFFSET_1a_FITXA en cas de voler adreçar-vos directament als bytes 30 i 31 d’una capçalera d’una DBF estesa, i/o PRIMER_OFFSET_a_OFFSET_1a_FITXA per accedir al primer paquet, o per a DBF clàssiques.]

En **camps especials** a les DBF esteses (els **C** a la v. actual, 0x90), el nombre de **BytesPerCamp** no es defineix al byte 16, sinó als bytes 21-24 (*unsigned __int32*) de cada paquet de 32 bytes que defineix cadascun dels camps. El 16è byte queda, en aquests casos, amb valor 0.

[Programadors useu OFFSET_BYTESxCAMP_CAMP_ESPECIAL o bé OFFSET_RESERVAT2_BYTESxCAMP_CAMP_ESPECIAL en cas de voler adreçar-vos directament als bytes per camp d'un camp C d'una capçalera d'una DBF estesa.]

Els **noms estesos** en les noves taules DBF permeten una longitud de 128 caràcters (MAX_LON_NOM_CAMP_DBF-1) i pràcticament qualsevol caràcter (cal notar que en altres grans gestors de bases de dades l'amplada del nom dels camps és igual o inferior a la proposa en la DBF estesa i, doncs, s'aconsegueix una màxima compatibilitat). Els únics caràcters que no es permeten són l'accent obert sol (sense accentuar cap lletra), els claudàtors, la dièresi i els caràcters no imprimibles (com ara el retorn de carro o el DEL), fent així la DBF compatible amb els noms de camps permesos en les bases de dades Oracle, MySQL, SQL Server. Per tal d'accedir als noms estesos de camp es fa a través dels bytes 25, 26, 27, 28 i 29 de la capçalera del camp en qüestió. Més concretament, els bytes 7, 8, 9 i 10 del membre **reservat2** de l'estructura Camp defineixen l'*offset* on cal anar a buscar el nom estès del camp. El byte 11 fa referència a la mida del nom, que com a màxim serà, com hem dit de 128 caràcters. Permetre més caràcters (fins a 256) no sembla necessari quan SQL Server ni Oracle no depassen aquest valor i faria que disminuís el nombre total de camps possible a la taula. El **joc de caràcters** (ANSI, OEM) amb què s'escriu un nom estès és el coherent amb el definit a la capçalera de la taula (el mateix amb què s'escriu els camps C); en cas que la taula no estigui marcada amb cap joc de caràcters (cosa no recomanable) s'utilitza el joc indicat a la clau `JocCaracDBFPerDefecte=` de la secció `[MiraMon]` del MiraMon.par.
[Programadors useu OFFSET_RESERVAT2_OFFSET_NOM_ESTES per al byte 7 de reservat_2 de l'estructura Camp i useu OFFSET_RESERVAT2_MIDA_NOM_ESTES per al byte 11 del mateix membre].

4. Implantació al MiraMon (secció per a programadors)

S'ha fet les proves inicials amb el **MiraDades**, a fi de demostrar que és capaç de generar, escriure i llegir taules DBF esteses (amb un **nombre de camps superior a 255, camps C amb longitud pràcticament il·limitada** [en teoria limitada per 4200 milions caràcters] i **noms estesos**). Des del moment en què les llibreries disposen de les funcionalitats per a treballar amb aquestes noves taules, les aplicacions passaran per 3 possibles fases d'adaptació:

- Adaptació **natural**: Consisteix en la simple recompilació de l'aplicació, que fa que treballi amb les noves llibreries i, per tant, de manera directa amb les noves taules. Això passarà sobretot en les aplicacions que llegeixen taules DBF i no n'escriuen.
- Adaptació **funcional**: Probablement algunes aplicacions necessitaran retocs específics que aniran essent realitzats a mesura que ho anem detectant. Potser passarà en algunes aplicacions que escriuen taules.
- Adaptació de la **grandària màxima del fitxer DBF**. Els canvis introduïts poden fer (tant perquè les taules poden tenir més camps com perquè aquests poden ser més amples) que sigui menys improbable que una taula assoleixi els 2 Gbytes de grandària. En aquest cas caldrà imperativament fer una revisió de la codificació de l'aplicació amb les llibreries corresponents MM_fopen(),

MM_fread(), etc. Com que aquest fet no s'espera que ocorri amb freqüència, aquesta darrera fase s'anirà obrint a mesura que sigui possible i sempre que sigui necessari. La disponibilitat d'un punter a fitxer de 64 bits, pf64, indicarà que és aquest el que cal usar i no el tradicional "pf".

Els programadors han de llegir amb especial atenció també els **Annexos 1** i **2**.

El mòdul CGI.h/c ha estat visitat només superficialment, però és evident que adaptar-lo rendiria beneficis en cerques en formularis, etc, per la qual cosa se suggereix de valorar un impuls per no arrossegar durant més temps les limitacions de la DBF clàssica en les aplicacions a Internet.

Pel que fa als noms estesos cal també parar atenció als Annexos 1 i 2, on s'explica quina repercussió té aquesta implementació en l'ús dels noms dels camps per a taules DBF.

5. Nota final

S'ha fet proves de funcionament per mirar de garantir el perfecte comportament de les noves funcionalitats:

- Creació de camps C especialment amples (més de 100 000 caràcters), és a dir amb longituds que clarament depassen el límit de les DBF clàssiques i fins i tot els 65536 caràcters que quedarien definits en entorns de memòria de 16 bits.
- Creació de taules amb un nombre gran de camps (més de 300, etc).
- Canvi de mida de camps C i del nombre de camps per comprovar que en cas de no ser necessària la marca de DBF estesa, la DBF torna a ser una DBF clàssica.
- Repassada exhaustiva del que es pot fer amb el GestBD o amb el propi MiraD.
- Respecte dels camps de la DBF: Crear, eliminar, omplir, canviar, buscar i substituir, concatenar, duplicar, moure i mostrar valors.
- Respecte de les taules DBF: Fusionar, *join*, ordenar, suprimir un o més registres, generació de taula única, duplicar registres, converteix expressió en resultat, crea registre buit DBF, etc.
- Cal que JM repassi les funcions del GestBD que encara són antigues i les passi a modernes tenint en compte que potser no caldrà fer res més: per exemple TEST_CONTINGUT_REGISTRES_DBF.

D'altra banda una DBF estesa oberta amb el MiraDades mostra aquesta característica en la caixa que s'obre en fer "Informació | Informació de la taula...". Noteu en la següent captura els elements marcats en color vermell:

```
Nom del fitxer: E:\aa.dbf
[DBF estesa (no compatible amb dBASE)]
Darrera modificació: 18-12-2011
Nombre de registres que conté: 8
Nombre de camps per registre: 4
Joc de caràcters: Windows ANSI (88, 0x58)
```

```
Característiques dels camps:
-----------------------------------------------------
NÚM |     NOM           | DESCRIPTOR | T |    AMP | REL
-----------------------------------------------------
  1 |  CAMP_C           |            | C | 106383 |
  2 | R_COLOR           |            | N |      3 |
  3 | G_COLOR           |            | N |      3 |
  4 | B_COLOR           |            | N |      3 |
  5 | Nom estès de camp |            | N |      3 |
```

Tanmateix, agrairem especialment als usuaris que ens avisin de qualsevol possible error o disfunció que nosaltres no haguem detectat, així com de la necessitat de la tercera fase d'adaptació per a necessitats concretes.

ANNEX 1: Implicacions en els codis

Canvis i addendes efectuats:

BD_XP.h:

// Els 2 següents define són nous:
#define MAX_N_CAMPS_DBF_CLASSICA 255
#define MAX_N_CAMPS_DBF_ESTESA 13421772L

// Noms estesos
#define MAX_LON_NOM_CAMP_DBF_CLASSICA 11 // Inclou el \0 final
#define MAX_LON_NOM_CAMP_DBF 129 // Inclou el \0 final

#define MAX_N_CAMPS_DBF MAX_N_CAMPS_DBF_ESTESA

typedef unsigned short int TIPUS_NUMERADOR_CAMP_DBF;
→ **typedef unsigned __int32** TIPUS_NUMERADOR_CAMP_DBF;

#define printf_TIPUS_NUMERADOR_CAMP_DBF "%hu" → printf_UINT32
#define scanf_TIPUS_NUMERADOR_CAMP_DBF "%hu" → printf_UINT32
#define MAX_TIPUS_NUMERADOR_CAMP_DBF USHRT_MAX → _UI32_MAX

typedef unsigned char TIPUS_BYTES_PER_CAMP_DBF; → unsigned __int32
#define MAX_TIPUS_BYTES_PER_CAMP_DBF 255 → (_UI32_MAX-1)

typedef unsigned short int TIPUS_BYTES_ACUMULATS_DBF;
→ **typedef unsigned __int32** TIPUS_BYTES_ACUMULATS_DBF;
#define MAX_TIPUS_BYTES_ACUMULATS_DBF USHRT_MAX → _UI32_MAX

// Els 2 següents define són nous:
#define MAX_AMPLADA_CAMP_C_DBF_CLASSICA 254
#define MAX_AMPLADA_CAMP_C_DBF_ESTESA (_UI32_MAX-1)
#define MAX_AMPLADA_CAMP_C_DBF
passa de 254 a MAX_AMPLADA_CAMP_C_DBF_ESTESA

#define MAX_AMPLADA_CAMP_DBF MAX_AMPLADA_CAMP_C_DBF
// Els 2 següents define són nous:
#define MAX_AMPLADA_CAMP_DBF_CLASSICA
passa de 254 a MAX_AMPLADA_CAMP_C_CLASSICA
#define MAX_AMPLADA_CAMP_DBF_ESTESA MAX_AMPLADA_CAMP_C_DBF_ESTESA

A struct BASE_DADES_XP

short int OffsetPrimeraFitxa; → **unsigned __int32**

Nota:

BytesPerFitxa s'adapta sol en ser de TIPUS_BYTES_ACUMULATS_DBF.

A struct CAMP

char NomCamp[11]→ char NomCamp[129] usant els defines corresponents.

Creació de char NomCampDBFClassica[11], usant el define corresponent, per a emmagatzemar els possibles noms curts en la lectura de les DBF que necessitin conservar-lo. Seria estrany obrir una DBF amb el MiraD i guardar-la i que desapareguessin els noms curts.

Nous recursos de programació:

- La macro ES_DBF_ESTESA(versio_dbf) (BD_XP.h) retorna TRUE/FALSE.
- La definició MARCA_VERSIO_1_DBF_ESTESA indica el contingut del primer byte en una DBF estesa.
- Existeix un **nou tipus**: TIPUS_OFFSET_PRIMERA_FITXA.
- La definició SEGON_OFFSET_a_OFFSET_1a_FITXA indica on començar a llegir els dos bytes que, en una DBF estesa, complementen els 2 d'una DBF clàssica per a definir **OffsetPrimeraFitxa**.
- La lectura/escriptura dels BytesPerFitxa es pot marcar, en codi o en comentaris, amb **sizeof**(TIPUS_BYTES_ACUMULATS_DBF); això permet trobar més fàcilment els llocs del codi implicats amb aquest canvi.
- La definició OFFSET_BYTESxCAMP_CAMP_ESPECIAL indica on començar a llegir els 4 bytes que, en una DBF estesa, defineixen, en cada paquet de 32 bytes que defineix un camp, els **BytesPerCamp** (i OFFSET_RESERVAT2_BYTESxCAMP_CAMP_ESPECIAL ho indica des de reservat_2), mentre que OFFSET_BYTESxCAMP_CAMP_CLASSIC indica el lloc habitual per a la resta de camps.
- La funció RevisaCapcaleraDBF() pot ser útil perquè, com a novetat, omple el membre versio_dbf de la capçalera en funció de les característiques de la taula i, doncs, pot ser útil en la creació/revisió de taules creades. De fet, però, s'ha treballat perquè les funcions de creació i tancament marquin adequadament les taules DBF com a clàssiques o com a esteses.
- La funció DonaBytesDelCampMesLlarg() resulta útil quan voleu fer una reserva de memòria genèrica per a qualsevol dels camps d'un arbre de relacions REL1 o REL4.
- El fitxer capcalera.c ha estat revisat perquè les funcions que hi ha tinguin en compte els noms estesos.
- Cal tenir en compte que el nombre de camps ja no es calcula com abans, és a dir, que es va llegint camps fins que la suma de mides dels camps coincideix

amb els bytes per fitxa. Això es fa així perquè la primera fitxa ja no comença al final del byte 0x0D, sinó després de l'últim nom estès.

- La funció **ActualitzaTotaCapcaXerraire()** és l'encarregada, entre d'altres coses que ja feia, de detectar si existeixen noms estesos, d'omplir els offsets i mides dels noms estesos, així com de col·locar-los en el lloc corresponent dins el fitxer, ampliant o reduint l'espai entre 0X0D i els noms dels camps estesos.
- Per saber si un nom de camp és estès cal usar la funció **EsNomEstesBD_XP(NomCamp)**
- La funció **LlegeixCapcaleraDBF_i_RELOptativament()** calcula el nombre de camps d'una manera diferent a com ho feia abans. Ara, enlloc d'aplicar una fórmula el que fa és anar sumant els bytes per camp fins a arribar als bytes per fitxa. A més, posa el nom de camp clàssic en el membre NomCampDBFClassica de l'estructura CAMP i el nom estès en el membre NomCamp (afegint el '\0' al final en tots dos casos).
- Per saber si una DBF conté noms estesos o no, es pot cridar la funció **ES_DBFAmbNomsEstesos().**
- Si es vol convertir una DBF que conté noms estesos a una DBF sense noms estesos, per exemple per a la DBF dels fitxers SHP, es pot usar la funció **EliminaPartNomsEstesosDeDBF()**.
- Existeixen també funcions d'accés a informacions relacionades amb l'*offset* i la mida dels camps estesos: **DonaOffsetNomCampEstes(), DonaBytesCampsNomEstesCamp(), CalculaBytesCampsNomsEstesos(), InicialitzaOffsetCampsNomEstesCamp(), InicialitzaBytesCampsNomEstesCamp().**
- Existeixen macros per a escriure directament l'offset i la mida dels noms estesos a l'estructura CAMP. Els escriurem a disc només usant **ActualitzaTotaCapcaXerraire()**, **EscriuMidaNomCampEstesBD_XP()** i **EscriuOffsetNomCampEstesBD_XP()**.

Valors que cal buscar a TOTS els codis que NO són llibreries:

En les llibreries els següents canvis/consideracions ja han estat fets, però és imprescindible fer una repassada per aplicar-los als codis específics dels diferents mòduls:

- Valorar el nou valor de MAX_TIPUS_NUMERADOR_CAMP_DBF. S'ha trobat a 20 mòduls de les llibreries, però sempre era un "if(MAX.." que en realitat comprovava el retorn de la funció DonaIndexCampDBF(), que ja està automàticament adaptada a la DBF estesa; també ha estat trobada en DonaIndexCampGeoTopoBD_XP(), que igualment ha resultat automàticament adaptada.
- Quan MAX_N_CAMPS_DBF sigui un valor usat per a inicialitzar un *array* estàtic o per a una reserva dinàmica de memòria caldrà aplicar MAX_N_CAMPS_DBF_CLASSICA. Aquesta situació és molt poc probable (de fet, no ha estat trobada a les llibreries) i, doncs, no es preveu que es trobi.
- Buscar MAX_AMPLADA_CAMP_DBF per a valorar implicacions (*arrays* estàtics, etc) del canvi a ulong i eventualment deixar-la en MAX_AMPLADA_CAMP_DBF_CLASSICA. S'ha fet una cerca pels següents textos (noteu que els parèntesis/claudàtors estan descompensats expressament):

    - "[MAX_AMPLADA_CAMP_DBF" (Trobat a molts mòduls: StructMM.h, TaulaSmb.h, etc.)
    - "(MAX_AMPLADA_CAMP_DBF" (Trobat a 9 mòduls de les llibreries, però en alguns casos no ha calgut fer el canvi.)
    - "MAX_AMPLADA_CAMP_DBF)" (Trobat a 16 mòduls de les llibreries.)
    - "MAX_AMPLADA_CAMP_DBF," (Trobat a 5 mòduls de les llibreries, per exemple en crides com a argument a OmpleCampBD_XP().)
    - "*MAX_AMPLADA_CAMP_DBF" (Trobat a 1 mòduls de les llibreries.)
    - "MAX_AMPLADA_CAMP_DBF*" (No trobat a les llibreries.)
    - "+MAX_AMPLADA_CAMP_DBF" i "-MAX_AMPLADA_CAMP_DBF" (Trobat al MiraD.)
    - "MAX_AMPLADA_CAMP_DBF+" i "MAX_AMPLADA_CAMP_DBF-" (Trobat a 6 mòduls)

  Aquesta cerca s'ha demostrat molt útil per a detectar llocs on calia incidir amb un retoc del codi i s'aconsella MOLT que la utilitzeu sobre els codis de les MSA.
- Com en el cas anterior per a MAX_AMPLADA_CAMP_C_DBF: S'ha trobat a 17 mòduls de les llibreries, que majoritàriament han esdevingut MAX_AMPLADA_CAMP_C_DBF_CLASSICA excepte en les funcions en les quals tenia sentit suportar valors més grans (creació de nous camps, canvi d'amplada, etc).
- Buscar MAX_TIPUS_BYTES_PER_CAMP_DBF per a valorar l'oportunitat d'usar MAX_TIPUS_BYTES_PER_CAMP_DBF_CLASSICA. En les llibreries s'ha aplicat el canvi a DXFTop.c, GMLTop.c, GPXTop_f.c, KMLTop.c i SmbCmp.c.
- Si cal modificar un nom de camp és altament recomanable utilitzar les funcions adequades que gestionen correctament els noms estesos i els curts relacionats que no intentar manipular directament Camp->NomCamp.

Amb tot això cal anar treballant amb les aplicacions posant-les en situació de trobar-se amb una DBF estesa. Un bon exemple és ANSIOEM.exe.

Pistes sobre errors en compilar:

- Si en compilar un mòdul apareix "*Internal compiler error*" en un punt del codi probablement serà perquè teniu una matriu estàtica amb MAX_AMPLADA_CAMP_DBF que hauria de ser MAX_AMPLADA_CAMP_DBF_CLASSICA.

Aspectes a tenir presents en els recursos "EDIT" de les caixes de diàleg:

- Si un recurs destinat a editar o mostrar el contingut d'un camp ha estat declarat en el fitxer RC, caldrà que en despatxar l'esdeveniment WM_INITDIALOG feu:
  SendMessage(GetDlgItem(h_dlg,IDC_EDIT), **EM_LIMITTEXT**, Camps[i].BytesPerCamp+1);

  Si no feu això el recurs d'edició estarà limitat a 30000 caràcters i, per tant, no podreu enganxar-hi continguts de camps C realment grans.

- Si el recurs l'heu generat dinàmicament, per exemple cridant la funció CreateWindowExCharMissatge(), convé que també feu el procediment anterior o no suportarà textos llargs.

ANNEX 2: Informació complementària

Peter Mikalajunas, kd9fb@xnet.com, http://www.xnet.com/~kd9fb facilitava, el 26/5/98, aquesta informació, que no hem validat, però que pot servir de referència orientativa i complementària per a les BDF clàssiques. Ha estat complementada amb el que es recull en la definició de la nova DBF estesa (en vermell les coses a notar que han canviat i com, i en blocs de colors, els "paquets" que s'han de llegir junts):

DBF FILE STRUCTURE
~~~~~~~~~~~~~~~~~~
BYTES DESCRIPTION
00 FoxBase+, FoxPro, dBaseIII+, dBaseIV, no memo - 0x03
FoxBase+, dBaseIII+ with memo - 0x83
FoxPro with memo - 0xF5
dBaseIV with memo - 0x8B
dBaseIV with SQL Table - 0x8E
0x90 a les DBF esteses

01-03 Last update, format YYYYMMDD **correction: it is YYMMDD**
04-07 Number of records in file (32-bit number) LOCUS_N_REGISTRES
08-09 Number of bytes in header (16-bit number). Posició PRIMER_OFFSET_a_OFFSET_1a_FITXA → 32-bit a les DBF esteses: vegeu 30-31.
10-11 Number of bytes in record (16-bit number) → 32-bit a les DBF esteses: vegeu 12-13.
12-13 Reserved, fill with 0x00 → Conjuntament amb 10-11 defineix el membre **BytesPerFitxa** a la DBF estesa.
14 dBaseIV flag, incomplete transaction
Begin Transaction sets it to 0x01
End Transaction or RollBack reset it to 0x00

15 Encryption flag, encrypted 0x01 else 0x00
Changing the flag does not encrypt or decrypt the records
LOCUS_CINQUE_BYTE_N_REGISTRES → Si s'usa aquest byte per a passar el n de registres a un enter de 40 bytes, podem indicar un valor de fins a 1 bilió (10^{12}) registres.

16-27 dBaseIV multi-user environment use
28 Production index exists - 0x01 else 0x00
29 dBaseIV language driver ID
30-31 Reserved fill with 0x00 Posició SEGON_OFFSET_a_OFFSET_1a_FITXA → Conjuntament amb 08-09 defineix el membre **OffsetPrimeraFitxa** a la DBF estesa.
32-n Field Descriptor array
n+1 Header Record Terminator - 0x0D

FIELD DESCRIPTOR ARRAY TABLE
~~~~~~~~~~~~~~~~~~

BYTES DESCRIPTION
0-10 Field Name ASCII padded with 0x00
11 Field Type Identifier (see table)
12-15 Displacement of field in record
16 Field length in bytes. Posició OFFSET_BYTESxCAMP_CAMP_CLASSIC → Defineix **BytesPerCamp**, excepte en camps especials (els C a la v. 0x90) a les DBF esteses, en què **val 0** i BytesPerCamp s'emmagatzema als bytes 21-24.
17 Field decimal places
18-19 Reserved
20 dBaseIV work area ID
21-30 Reserved Posició OFFSET_BYTESxCAMP_CAMP_ESPECIAL (o OFFSET_RESERVAT2_BYTESxCAMP_CAMP_ESPECIAL des de reservat_2) → En camps especials (els C a la v. 0x90) a les DBF esteses defineix, als bytes 21-24 (ulong), els **BytesPerCamp**. S'usa els bytes 25 al 28 per a l'offset del nom estès i el 29 per a la mida d'aquest. Queda, doncs, encara lliure, el byte 30.
31 Field is part of production index - 0x01 else 0x00

FIELD IDENTIFIER TABLE

ASCII	DESCRIPTION
C	Character
D	Date, format YYYYMMDD
F	Floating Point
G	General - FoxPro addition
L	Logical, T:t,F:f,Y:y,N:n,?-not initialized
M	Memo (stored as 10 digits representing the dbt block number)
N	Numeric
P	Picture - FoxPro addition

Note all dbf field records begin with a deleted flag field. If record is deleted → 0x2A (asterisk), else 0x20 (space)
End of file is marked with 0x1A Això ha estat eliminat.