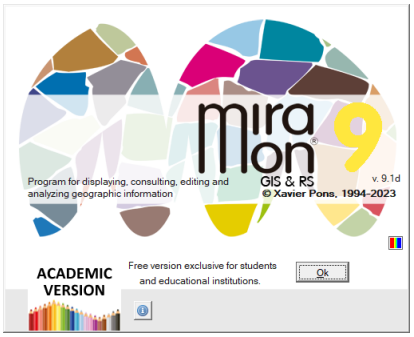


"EXTENDED DBF" FORMAT SPECIFICATION (INCLUDES "EXTENDED NAMES" IMPLEMENTATION)

Authors of the document: Xavier Pons and Abel Pau
First proposal: 23-12-2011
Last modification and version of the document: 16-02-2024. **1.9**

1. Background and motivation.

The **DBF format,** originally developed in the context of dBASE software, followed in other software such as FoxBase and FoxPro, and extensively described in the specialized literature and on the Internet, is, except for tabular forms in plain text, perhaps the most popular of the alphanumeric data table formats. MiraMon can access tabular data in DBF III, III+ and IV format (and supports some ulterior additions, as F fields, etc), as well as tabular data (physical or as a result of query expressions) contained in other file formats (such as XLS, MDB, etc) and in databases (such as Oracle, SQL-Server, etc); in these last cases, the corresponding ODBC drivers must exist and be properly installed and, with the exception of the MDB and ACCDB (which supports direct access from MiraMon), a DSN file must be created to access the data.

Despite the great potential given by the access to the various tabular sources outlined above, in the MiraMon context DBF is often the chosen table format, both for its simplicity and speed, as well as for not depending on third parties, or simply because the certainty that, when disseminating information, it can be opened on a computer with unknown installed drivers (so it is the default option for the creation of MMZX and MMZ files, intended for wide distribution on the Internet). It should also be remembered that the DBF tables can be linked to other DBF tables and to other tables (MDB, SQL-Server, etc) by establishing links (joins) in which it is possible to indicate the cardinality (1 to 1, 1 to many, etc) and the level of the compulsory nature

of the join resolution (dictionary type or not) in a set of relations that is not restricted neither in the number of relations per field, nor in the number of relational levels. These joins are stored in the corresponding **REL files.**

However, and due to its design time, the DBF format has several limitations. The evolution of the DBF format after version IV was quite complicated (selling the product between different companies, strange decisions such as the inclusion of new numeric types that often did not contribute anything meaningful, etc [see http://en.wikipedia.org/wiki/DBase]), so users' and developers' community generally remains loyal to the III, III+, and IV formats for both reading and exporting. Currently there is still one dBASE product for sale (http://www.dbase.com).

Some of the limitations of DBF IV have been overcome thanks to indications that are also stored in REL files and are set from the **MiraMon Universal Geospatial Metadata Manager, GeM+.** Some of the most outstanding are:

- The length of the file name, which in the original specification was 8+3, and which is extended in MiraMon to any length supported by the operating system. In addition, in MiraMon the names of the DBF files and the directories where they are can contain spaces, accents, etc.
- The name length of each DBF field is limited to 10 characters and cannot contain accented letters, spaces, special characters, etc. However, from the GeM+ a free text descriptor can be indicated, without limitations of accents, special characters, etc, for each field; in addition, the descriptor can be multilingual if desired and supports a length, in characters, of:
  max(_MAX_PATH+100,256)
- In tables previous to dBASE IV, the character set used in 'C' type fields was not specified. MiraMon allows a flexible and configurable solution for these cases and, as in dBASE IV tables, supports specifying the character set (byte at offset 29), whereby the interpretation of accents and special characters is no longer ambiguous.
- In fields with numerical content, their units cannot be indicated if they have them. Instead, from the GeM+ these can be specified, as well be shown in the queries, if desired.
- In DBF tables the quality of the content of each field cannot be indicated. However, from the GeM+ this can be specified.
- In the DBF tables the treatment (categorical, ordinal or continuous quantitative) of the content of each field cannot be indicated. However, from GeM+ this property can also be specified.

Despite these important extensions introduced through REL files (which can also benefit other tabular formats readable by MiraMon), there are other limitations of the DBF format that cannot be solved except by introducing slight modifications to the format itself. Among the most important, the following ones can be highlighted:

- Limiting the field number to 128 in dBASE III+ (dBASE book by Jordi Abadal, p. 137, and Excel 2000 export criteria) or 255 (dBASE IV). This limitation

cannot only be found in tables of all kinds, but is even more frequent in the unique table created to resolve all the joins in the relation tree specified from GeM+ when the relations are numerous, and especially when tables with many fields are joined. Finally, it should be noticed that MiraMon does not establish this distinction between dBASE III+ or dBASE IV versions and assumes, for any classic DBF, regardless of version:
    - 255
- The limitation of the number of characters, in fields of type 'C':
    - 254
- The limitation of the format and length of the name of the DBF fields to 10 characters in capital letters (without supporting accented letters, ç, etc).
    - 11 (including the null string terminator)

There are other possible improvements to mention that could be the subject of discussion about the opportunity of their design and future implementation:

- Numbers are currently written as text that must be interpreted. It would be convenient to define binary numeric types following the usual standards for integer and real numbers.
- Support for variable-length text fields with a file indexing system that tells where each record begins (although this would make access a bit slower).
- Support for unlimited length binary fields.
- Support for individual compression per record with a file indexing system that tells where each record begins.
- Support for a mark of deletion of a field (column).
- Choose a text description of the header.
- The first 32-byte "miniheader" could be extended to accommodate future extensions. This involves coding the size of the miniheader itself.
- The description of each field could be extended. This involves encoding the size of a field description in the miniheader.
- Extend the maximum number of records to an **unsigned __int64** (64-bit integer). The value can be written by combining the 4 bytes of the classic DBF plus the 4 bytes 16-19 of the header.
- The internal date of the file could be removed. It has always been an important problem and is redundant with the one of the file system. In addition, it does not include the time and, therefore, it is not useful for restoring the exact date-time of a file that has been emailed and has lost the time stamp.
- A new date-time field could be created.

The MiraMon proposes to solve the most important of these limitations and adopt some improvements, while establishing a variation of the DBF format that has been called "**extended DBF**".

2. Characteristics and use of the "Extended DBF" format.

2.1 If a table does not need to overcome the limitations of the classic DBF, it is preferable to write it in this format so that it is readable by other software that does not support the extended DBF. Note that the format does not maintain backward compatibility (a software that reads classic DBF will not read an extended DBF, even partially, unless it has implemented the proposal explained in this document).

2.2 The file **extension is .dbf** as in classic DBF.

2.3 The **first byte** is **0x90.** To check if the software can read this, it just checks for the '9', which allows it to change the second number for backward compatible modifications. Smaller values are not used to avoid conflict with other numberings.

2.4 The **number of** possible fields becomes 13.4 million (the exact maximum value is justified in the next section).

2.5 The **length of a record** is an **unsigned __int32,** counting the record deleted byte.

2.6 The **size of a C field** can reach **unsigned __int32 - 1** (one byte is given up to fit the record deleted byte).

2.7 The file will continue to contain the end of field definition mark (Header Record Terminator - 0x0D) as in the classic DBF since a fork would have to be made in the code everywhere where the offset to the first record is calculated and the problem that would be generated between classic and extended DBF is considered too critical and, therefore, it is preferred to sacrifice this byte of storage.

2.8 The file does NOT contain the end-of-file mark (0x1A) that can often be found in classic DBF. The file will contain the extended field names between the field definition end mark (Header Record Terminator - 0x0D) and the first record. Access to these fields will be based on an offset and a size (the extended names will not contain the final '\0') stored in the header of the fields (the offset in the reserved2+7, of 4-byte length, and the size, in the reserved2+11 of 1-byte length. The maximum length of an extended name of a field would therefore be 255 characters, but as discussed later, it ends up being 128 characters, enough compared to what other large database managers support (Oracle 9.i supports 30, and SQL Server 2000, 128).

2.9 The user will be warned with a message if a DBF goes from not containing extended names to containing them:
"The field name you want to create (%s) has characteristics not supported by the classic definition of DBF files (such as accents, spaces, length greater than 10, etc). Remember that you can also generate free-form names from the "Descriptor" box or from the Metadata Manager (GeM+).
Do you want to use the name anyway? [Yes/No]"

C fields with extended lengths allow putting particularly long and complex text content into the fields, such as HTML encoding, xpath() expressions for accessing open data resources, etc. In the following example taken from a hypothetical

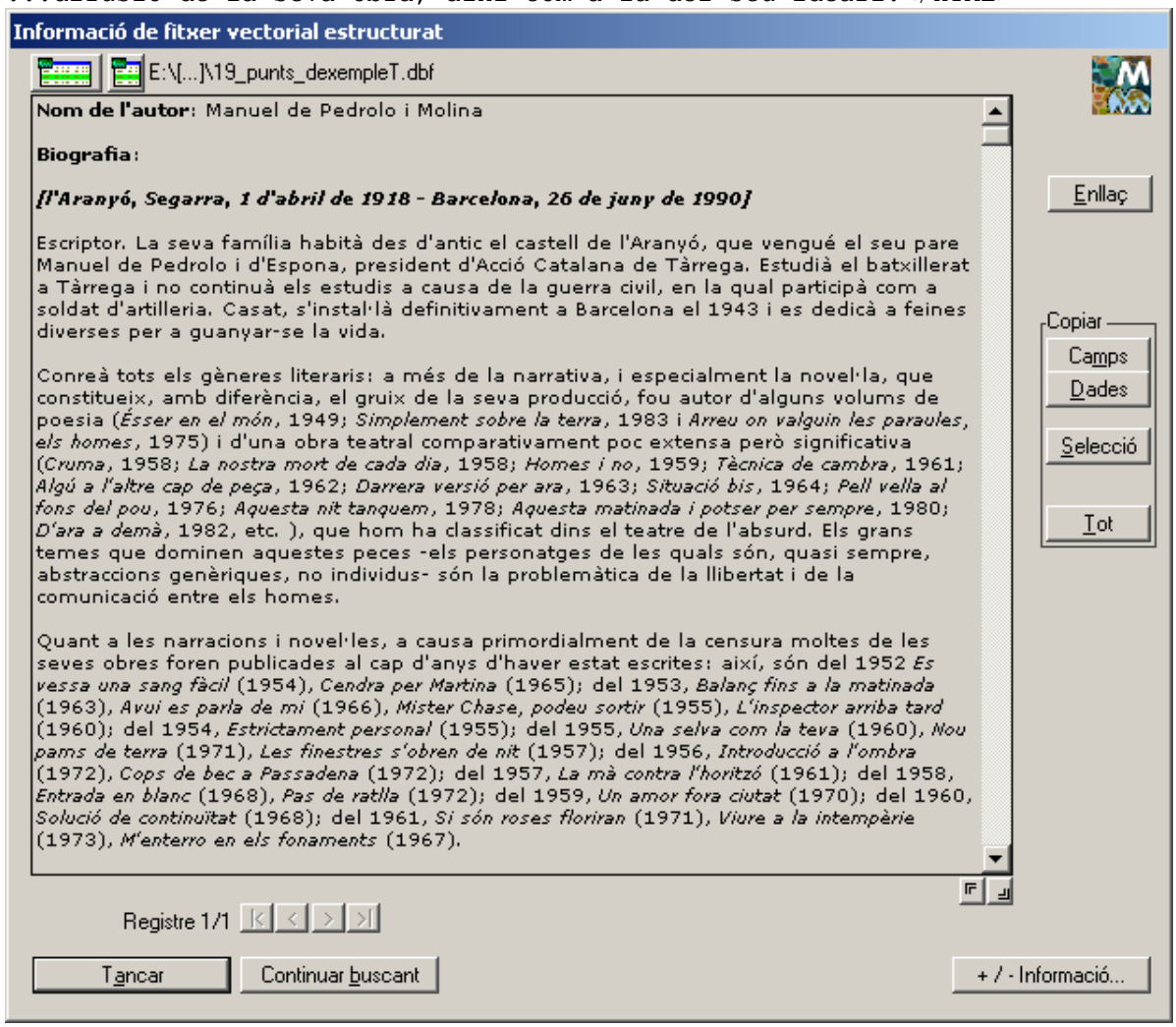
database where the birthplace of several writers is specified, the query allows accessing to their biography (much longer, about 100 000 characters, than what appears in the screen shot, as it can be deduced by the size of the vertical scroll bar button), in this case taken from the Catalan Encyclopedia on Internet. Naturally, adding a link to a photo, links to other resources on the Internet or to an intranet, etc, can also be possible.

In the example shown, the field starts:

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

and ends:

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



3. Specification of the “Extended DBF” format.

The **first byte** is **0x90.** This is the **extended DBF mark.** Further improvements could involve successive hexadecimal numbering.

The **2 bytes** called **reserved_1** (bytes 12-13, numbered from 0) will be read together with 10 and 11 as a single 4-byte package (*unsigned __int32)* which will define the **number of bytes per record**. The extension to a 32-bit integer is necessary to be able to fit, for example, several C fields with a large width.

The **4 bytes** located in bytes 16-19 will be read together with those located at bytes 04-07 as a single 8-byte package (*unsigned __int64)* that will define the **number of records**. The extension to a 64-bit integer is necessary to be able to fit, for example, the point attributes of high-density lidar files over very large countries.

The **2 bytes** 30-31 will be read together with 8 and 9 as a single 4-byte package that will define **where to start storing the records**. Unlike in classic DBF, where the **number of fields in the table** determines where the DBF header ends, this is no longer the case since it is necessary to anticipate that there will still be, after the end mark of the classic header (Header Record Terminator, 0x0D), the extended names. The number of fields now supported by the extended DBF is conditioned by the 32 bytes intended to describe each field and this offset where the records begin, within which it is necessary to include, in addition to the description of each field, the header of 32 bytes, the final byte 0x0D marking the end of field descriptions and extended names. So, for now, it remains: (2147483648-32-1)/(32+128)= 13 421 772, or **about 13.4 million fields.**

In **special fields** in the extended DBF (the **C** ones in the current version, 0x90), the number of **bytes per field** is not defined in byte 16, but in bytes 21-24 (*unsigned __int32)* of each 32-byte package that defines each of the fields. The 16th byte remains, in these cases, with value 0.

The **extended names** in the new DBF tables allow a length of 128 characters and practically any character (note that in other large database managers the field name width is equal to or less than the proposed in the extended DBF and therefore maximum compatibility is achieved). The only characters that are not allowed are the open accent alone (without accenting any letter), square brackets, diaeresis, and non-printable characters (such as carriage return or DEL), thus making DBF compatible with field names allowed in the Oracle, MySQL or SQL Server databases. The access to the extended field names is done through bytes 25, 26, 27, 28 and 29 of the header of the field. More specifically, bytes 7, 8, 9 and 10 define the offset where to look for the field extended name. Byte 11 refers to the size of the name, which will be, as said before, a maximum of 128 characters. Allowing more characters (up to 255) does not seem necessary when neither SQL Server nor Oracle exceed this value and would decrease the total number of possible fields in the table. The **character set** (ANSI, OEM, UTF-8) with which an extended name is

written is the one consistent with that defined in byte 29 of the table header (the same with which C fields are written). Note for the classic DBF case: Although special characters (accented letters, etc) in field names do not conform to the classic DBF standard, ArcGIS and QGIS create and tolerate accented field names in classic DBF. The philosophy at MiraMon is, in a classic DBF, not to generate accented letters, etc, in the names of the fields, as it is too far from the standard, but in case there are these extended characters, to tolerate them and display them according to the character code of byte 29 of the header.

For additional information, please consult MiraMon databases.