

db

COLLABORATORS

	TITLE : db		
ACTION	NAME	DATE	SIGNATURE
WRITTEN BY		December 8, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	db	1
1.1	db - A small and fast database program	1
1.2	disclaimer	1
1.3	preface	1
1.4	introduction	3
1.5	features	4
1.6	future	4
1.7	system	5
1.8	installing	5
1.9	settings	6
1.10	ref	7
1.11	window	7
1.12	view	8
1.13	gadgets	8
1.14	keys	9
1.15	menus	10
1.16	arexx	14
1.17	rff	15
1.18	identifiers	17
1.19	speed	20
1.20	history	20

Chapter 1

db

1.1 db - A small and fast database program

Table of contents

=====

Disclaimer	db reference manual
Preface	The window
Introduction	Views
Features	The gadgets
Future plans	The keys
System requirements	The menus
Installing and starting	ARexx
Settings	

Technical info

- The RFF file format
- RFF identifier list
- A note about SpeedRender

History of changes

1.2 disclaimer

Disclaimer

db has been tested and found stable in everyday use.
However the author is not responsible for any loss of data, damages to
software or hardware that may result directly or indirectly from
the use of this program.

1.3 preface

Preface

This program is postcardware, ie like freeware, this means that you can copy it freely as long as you don't ask any more money for it than a nominal fee for copying. If you like the program, please send a postcard to the author (address below).

To you who have sent a postcard already:
Thanks, your response has encouraged me to continue improving this product.

If you want to distribute this program then you must supply the whole archive (packed in .lha or unpacked). You are free to supply a copy of db on coverdisk magazines. If you do, please just send me a copy of that magazine for my collections :-)

Special permission is hereby granted to include db in
Public-Domain collections such as Fred Fish's Amiga Library.

HOW TO GET THE LATEST VERSION OF db

The fastest way to get the most recent release of db is to
download it from an AmiNet ftp site. For example: ftp.luth.se (biz/dbase)

You may also call "Datormagazin" BBS:(in Sweden. Type 'download db<ver>.lha')

DMZ: int +46 8 654 99 50
DMZ: int +46 8 654 99 62
DMZ: int +46 8 654 61 18
DMZ: int +46 8 654 34 03
DMZ: int +46 8 653 32 28
DMZ: int +46 8 650 83 62

If you've made a nice ARexx script that enhances db, or maybe designed a practical database layout, do share it so other people can have use of it. Send your work to me so I can include it in the archive.

If you have suggestions or remarks about this program, or if
you find any bugs, please let me know. I really like response from users.

Write to the following address:

David Ekholm
Mantalsvägen 33
s-175 43 Järfälla
Sweden

You who prefer the faster way may use:

phone: int +46 8-580 15668
email: david-ek@dsv.su.se

Testing and/or suggestions by:

Richard Ludwig, Mikael Östergren, Anders Callertun, Fredrik Jervfors.

Icon artwork by Mikael Östergren.

Thanks to Jan van den Baard for GadToolsBox (used in v1.0 & v1.1)

Additional ARexx scripts by Richard Ludwig
email: md93-ril@nada.kth.se

Thanks to Andrew Leppard for the short (though hardwarebanging :-()
asm code for tonedialing using the loudspeaker.
email: 9405571x@lux.levels.unisa.edu.au

Thanks also goes to the following persons for translations:

German translation by Uwe Roehm
email: roehm@forwiss.uni-passau.de

Polish translation by Michal Letowski
email: pro37@ci3ux.ci.pwr.wroc.pl

Dutch translation by Edmund Vermeulen
email: edmundv@grafix.wlink.nl

Danish translation by Christian Hoj
email: cbh@vision.auc.dk

Swedish translation by David Ekholm (oops myself :-))
email: david-ek@dsv.su.se

Italian translation by Michele Rotellini (50%)
(no email)
and Piergiorgio Ghezze (50%)
email: pghezze@oink.dsi.unive.it

Finnish translation by Jukka Kauppinen
email: Grendel@Freenet.hut.fi

French translation by Goncalves A. Georges
email: melkor@ramses.fdn.org

Norwegian translation by Lars Magnus Nordeide
email: larsmn@oslonett.no

1.4 introduction

An Introduction to db

db is a small and fast database program that I wrote after having tested numerous other PD database programs and always found something lacking or irritating me. They might have dozens of features not found in db, but they lacked font sensitivity and a standard GUI look and OS 3.0 behaviour.

My main need was to keep record on addresses and telephone numbers of friends

and companies. Before v2.0 db was fixed to be just an address and telephone database, but that has changed. db was designed with user definable layout in mind from start. Currently you have to use a file editor to specify the database fields and layout as there is no GUI for that.

(See The RFF file format)

Anyway that's a one time job, then you just USE the database and have fun.

Complete GUI support will probably only be included in a commercial product I plan to call REG.

When you use db you will notice that the user interface has been kept as compact as possible (few gadgets, menus and windows). Still the functionality in for example, searching and sorting is high. This is intentional. I prefer few buttons with high functionality than the opposite. The ASL requester is in my opinion an example of good design. It may look simple, but hides features like automatic drawer creation and filename completion. I hope you spend enough time with db to discover its hidden features.

1.5 features

Feature List

A partial list of db's features include:

- o Dynamic memory handling. Number of records and fields only limited by free memory.
- o GadTool based. (Use fields of string, checkbox and cycle type)
- o Mouse and keyboard driven.
- o User definable fields and layout.
- o Multiple views of the same database.
- o Commodore's Clipboard for flexible interaction with other programs.
- o AppWindow -just drag and drop database icons on db to load.
- o Online MenuHelp -Press HELP key when selecting a menu item.
- o Font sensitive.
- o ASL requesters for flexible loads and saves.
- o Localized.
- o ARexx support.
- o Dial numbers using a modem or loudspeaker.
- o WB and Shell usage with Commodore's template parsing.
- o Fast and flexible find function using AmigaDOS patterns.
- o Filter function.
- o Fast and flexible sort function. Multiple sort orders can be specified.
- o 'Export View' and two standard ASCII export features.
- o Automatic ASCII import (tab-separated ASCII).

1.6 future

Future plans

The following are plans for the future. The main changes will probably only be included in a commercial product called REG.

- o Sort numbers correctly.
- o Improved ARexx support.
- o List view of database.
- o Field and record merge.
- o Relations between databases.

1.7 system

System Requirements & Compatibility

db will only run on AmigaOS 2.04 or higher.
db needs at least AmigaOS 2.1 to use localized languages.
db needs to have the whole database in memory to run which limits the number of records to what the memory allows. However modern computers like the Amiga with, say 2 MB of memory, will typically allow databases with more than 5000 records -enough for most uses.

The idea is sort of: "one should not drive around with a big trailer in the middle of a city". db is a small and fast citycar.

db allocates exactly the amount of memory needed for each field. You just specify a maximum number of characters if you want a limit for some reason (zipcodes...). (If you don't specify a limit then there is a default limit of 200 characters)

This is a memory efficient model.

db has been tested with AmigaOS 3.0 and is written in SAS C 6.55.

1.8 installing

Installing and starting db

db runs from both Workbench and Shell. See Settings for parameter list.

Installing db is simple - just drag the db program icon and the Catalogs drawer to the desired destination drawer. (If you omit the Catalogs drawer, you will only get English text.)

To start db from Shell, just enter db followed by an optional parameter list. Enter db ? from Shell to see the parameters.

To start db manually from the Workbench simply double-click on its program icon.

You can also start db by clicking on a db project icon so that that project gets loaded automatically.

Another way to load a project is to simply drag that project over db's window and release the mouse.

1.9 settings

Settings

db can be set up either from Shell or from its ToolTypes. The following parameters apply to both environments:

DEVICE=<serial device name>	-Device used for dialing using a modem. The default is serial.device, but users of internal modems might use modem0.device or something else.
UNIT=<device unit number>	-For multiple serial units. 0 is default.
TONEDIAL	-Dial numbers using the loudspeaker instead.
TONEDIALSPEED=<number 1-10>	-Set the speed of the tonedialing. Lower values give higher speed.
CCITT5	-Use CCITT5 frequencies when dialing instead of DTMF frequencies.
DIALPREFIX=<prefix string>	-String to add before phonenumber when dialing. Default is ATDT (Hayes modems)
DIALPOSTFIX=<postfix string>	-As the above, but this will be added after the phonenumber (+ a return code) The default ;H returns the modem to command state and hangs up immediately. (you take over from there)
AREACODE=<areacode string>	-This will strip the <areacode string> from the beginning of phone numbers beginning with <areacode string> in order not to confuse some telephone systems. Let's say our area code is (08). Entering that code into AREACODE will make numbers like (08)123456 to be dialed as 123456.
PUBSCREEN	-Opens db on the specified public screen or the Workbench screen if not found.
FONTNAME	-You can specify a custom font to use other than the default which is the screen font. The name has to be typed with the .font

	addition, for example FONTNAME=times.font
FONTSIZE	-Size of font, see above. If the font makes the window to big to open, db will try the screenfont and at last topaz 8.
NOICONS	-This will suppress iconsaving for projects.
NOSPEEDRENDER	-Turns off SpeedRender .
NORETURNSTEP	-Turns off the ability to use the return key or enter key to step between fields. Holding down Alt when pressing return or enter disables returnstepping temporarily.
NOSERIAL	-This ToolType has currently no function.
HORIZBAR	-Places the record dragbar horizontally in the bottom of the window instead of to the right. Also allows the left and right keys to be used to move between records.
ESCQUIT	-Enables the Escape key to be used to quit db (in normal mode).
MAKEBACKUP	-Enables saving of a backup file of the database everytime the database is saved. (.bak added to the filename) I strongly recommend that you use this.

1.10 ref

db Reference Manual
=====

This section contains information most people figure out themselves by testing a program. db has on-line help that should be sufficient for most people.

I recommend that you experiment with the program first and turn to this section if there is a problem of some kind.

1.11 window

The window

The db window always opens on the default public screen, normally the Workbench screen (Unless you use the PUBSCREEN ToolType). It uses the screen font (or your specified font) and all text and gadgets are sized accordingly.
db has a zoom gadget next to the depth gadget to enable the window

to be minimized if db isn't needed for a while.

db will currently always display one record at a time that we call the current record. db can work in one of three modes:

- o Normal mode. Enter records and look at the database here.
- o Find mode. Information entered in the fields will serve as a search pattern. AmigaDOS patterns are accepted. Press enter to start search or press escape to cancel find mode. (In order for Enter to work, you will have to leave any activated string field first)
- o Sort mode. Numbers entered in the fields will serve as a sort-order description. Press enter to start sort or press escape to cancel sort mode. (In order for Enter to work, you will have to leave any activated string field first)

The window titlebar can look like this:

Adresses/Main view 16/43

It indicates a number of things namely:

Filename/Viewname Current record number/Number of records.

1.12 view

Views

The visual information in the database can be divided into many views. This enables the user to view the database in different ways.

One view may only contain address specs, while another view contains special information on the person's study-results for example. This is also useful when doing exports for label-printing if a label-looking view is designed.

Another reason to use views is to enable db to have more fields than is possible to show on the screen at one time.

If you don't understand the idea, try some of the example programs and try playing with the menuitems under the View menu and see what happens. (Only some of the examples have multiple Views)

1.13 gadgets

Gadget Operations

db's window is made up of two kinds of gadgets, program gadgets and field gadgets. Currently There are only three program gadgets -the record dragbar with it's two arrow keys. These gadgets has two uses, one is to

navigate in the database in a quick way. The second is to give a visual feel of where the user is in the database and how big the database is. This is illustrated by the size and position of the dragbar knob.

The field gadgets can be of three types: string, checkbox and cycle type. They can be activated in one of three ways:

- o Clicking the mouse over the desired field.
- o Pressing the key that corresponds to the underlined character in the field name. (Will toggle checkbox fields and cycle cycle-fields)
- o Pressing tab or shift-tab. (string fields only)

See Key operations for more information.

Use the HELP key or right mouse button to deactivate a string field. Checkbox fields store their state as 0's and 1's in the database. Cycle fields store the active choice as numbers from 0 and up. (In ASCII representation as always in db)

By using string fields to enter information makes it possible to enter more information than is visible in the gadget. If so, the gadget will start to scroll the entered text. Pressing shift-left or shift-right will move the cursor to the first and last character in a field as usual. Amiga-X will also clear the active gadget and Amiga-Q will undo what you've typed. By using PD programs like NewEdit makes it also possible to copy and paste information between fields, not just records as supported by db.

You don't have to click outside a string gadget or press enter to be able to use the other keys to navigate in the database. When finished entering data, there is no need to press enter to leave a string gadget either. Any information entered will be recorded in the database.

1.14 keys

Key Operations

The following keys can be used to control db (apart from the window shortcuts): This is the action performed in normal mode:

Up	- Previous record.
Down	- Next record.
Shift Up	- First record.
Shift Down	- Last record.
Return	- Forward search.
Shift Return	- Backward search.

In find and sort mode these are the active keys:

ESC	- Leave find or sort mode
Return/Enter	- Start search or sort.

To move between fields you use Tab and Shift-tab as is the standard today.

You may also use the return or enter key if the NORETURNSTEP ToolType is not specified. If you don't use NORETURNSTEP you have to deselect any string field to be able to use the return key to search in the database.

1.15 menus

Menu Operations

Note that much of this information can be found online in db by selecting a menuitem and pressing HELP.

Project/New:

Clears the database from all records, but keeps all fields.
Will ask before clearing the old database if it has unsaved changes.

Project/Open....:

Brings up a standard ASL file-requester to allow the user to select a new database file to load. Will warn the user before loading ontop of an unsaved database. db databases can also be loaded by dragging an icon over a db window. db accepts plain tab-separated ASCII files and files in the RFF file format.

Project/Save:

Saves the current database under a known name using db's RFF file format
If no name has been specified, db will automatically call Project/Save as...

Project/Save as....:

Performs the same behavior as Save item described above except brings up a standard ASL file-requester to let the user select a file and path name for the database. The user will be warned if he types the same name as an existing file. Otherwise a new file and icon will be created and the Workbench will be informed of the optional icon creation.

Project/Output/View....:

db has no internal label-layout generator. The idea is that other programs, better at layout, like DTP programs should handle that. Instead db outputs an ASCII file formatted as the current View which may look like a label or something else suitable for importing in other programs. A label template for PageStream is included. This will bring up a standard ASL requester for the View file.
The user will be asked if all records should be exported or only those matched by the 'Find record' function.

Project/Output/View with names....:

This menuitem works like 'Output view' above but adds fieldnames from the current view before each field. This is good for general printouts from the database.

Project/Output/Tab-separated ASCII...:

This item will bring up a standard ASL requester for saving a plain ASCII file, ie a tab separated textfile with one record per line and the fieldnames in the first line of the file. This is suitable for exporting db databases to programs like Excel for list printouts.

The user will be asked if all records should be exported or only those matched by the 'Find record' function.

Project/Output/Comma-separated ASCII...:

This item will bring up a standard ASL requester for saving comma-separated ASCII file, ie like the format above, but with commas as separators and quotes "" around the names. This is suitable for exporting db databases to programs like ProWrite for mailmerge printouts.

The user will be asked if all records should be exported or only those matched by the 'Find record' function.

Project/About...:

Brings up a requester showing information about the author, hidden features, revision number and ARexx port name. Click Ok to make the requester disappear.

Project/Quit:

First prompts the user with a requester if there is unsaved data and if the action is confirmed removes any currently installed database and exits the program.

Edit/Cut:

db uses Commodore's standard Clipboard to allow copying of information between db and other programs (wordprocessors, DTP...).
db writes to two clipboards, Unit 0 (the default clipboard) and Unit 1.
Reads are only made from Unit 1. This is what is written:

Unit 0: The current record with fields formatted like the current view.

Unit 1: The whole record in a record format like this:

fieldname <TAB> contents <NL>

fieldname <TAB> contents <NL>

...

Cut will copy the contents of the current record to the clipboard and delete the current record.

Edit/Copy:

Just copies the current record to the clipboard. See Edit/Cut for more information.

Edit/Paste:

Will add a new record and paste the contents of the clipboard that matches the fields in this database into the new record. Edit/Paste will paste from Clipboard unit 1 which has a format as described in Edit/Cut.

If there is no data in the clipboard that suits the fields, nothing will happen at all.

Using the public clipboard not only allows copying and pasting between db and wordprocessors. Many instances of db can move records between them also.

Edit/Add:

Adds a new empty record to the database. The new record will be added after the current record.

Edit/Kill:

The current record is deleted in normal mode.
In find and sort mode the fields are cleared instead.
You cannot kill a record if it is the only one.

View menu:

Use these menuitems to switch between different Views of the database (if there are more than one of course).
See Views section for more information.

Action/Find...

Will turn db into find mode. The current record will now indicate the desired search pattern.

- * Enter a search pattern into one or many of the fields.
- * Press Enter/Return to leave the fields
- * At the 2:nd press of the Enter/Return key db will then search the database starting from the top and stop at the first occurence that has a match.

Usually only a few letters will do as a search pattern. For example:

'da' will match both 'David' and 'Daniel'.

You may also use AmigaDOS patterns. For example:

'#?d' or '*d' will match fields ending with a d.

(david|micke)' will match both 'David' and 'Micke'.

Pressing the escape key or clicking the window closegadget returns db to normal mode.

Note: The export function uses this function to filter records

Note! Fields of checkbox and cycle type are ignored in find and sort modes. To be able to use all fields, user must make and switch to a view which only contains stringgadgets.

Action/Find next

This menu item is only supplied for compatibility with other programs. It will continue searching for other matching records. It is better to use enter/return instead and shift-enter/return to search backwards.

Action/Sort...

Will turn db into sort mode. The current record will now indicate the desired sort order. Just enter numbers in the fields. Anything else than numbers is ignored.

For example, entering a '1' in the Zip field and a '2' in the Name field indicates that you want the database sorted on zipcodes in the first hand and sorted on names in the second hand.

db will sort the Swedish ÅÄÖ characters correctly unlike most other programs. Pressing the escape key or clicking the window closegadget returns db to normal mode.

Note! Fields of checkbox and cycle type are ignored in find and sort modes. To be able to use all fields, user must make and switch to a view which only contains stringgadgets.

Action/Dial number

db will dial a phonenumber using your modem or loudspeaker. See Settings for how to configure db to your modem and serial device settings etc.

To dial a number using the modem, do the following:

1. Select the gadget containing a valid phonenumber (ie, at least one digit)
2. Select this menuitem or press Amiga-D.
3. When db starts to dial, lift the hook and wait. As soon as there is a connection, db will hang up and you can take over.

To interrupt db when dialing. Simply press Amiga-D once again. This should work with most modems

Dialing using the loudspeaker is similar:

1. Select the gadget containing a valid phonenumber (ie, at least one digit)
2. Lift the hook and hold the phone next to the loudspeaker with a suitable

volume setting.

3. Select this menuitem or press Amiga-D.

You may also use ARexx to simplify dialing to just consist of doubleclicking the number you want to dial. See the example scripts.

Settings/Display warnings:

With this item selected, the user will be warned before a Kill is performed on a non-empty record.

Settings/Sort direction

The user may also choose a backward sort direction.
db will sort the Swedish ÅÄÖ characters correctly.

Settings/Save changes...

This option is currently not available. When it is implemented db will save the changes in it's icon as tooltypes. Edit the tooltypes manually!

1.16 arexx

ARexx support

db doesn't have a full-featured ARexx support as I intend to release a commercial/shareware product later, but the commands actually implemented allows you to do things like this:

- * Show pictures, texts and play sounds etc from db.
- * Ensure that data entered in fields gets formatted as you like (UPPER, Caps..)
- * Expand codes to their full names etc (sort of a "filename completion")

To achieve this, ARexx programs can be invoked in two ways:

- * When the user doubleclicks a string field or hits LAmiga + key where key corresponds to the underlined character of a fieldname in a view. This is called requested invocation and is suitable for showing pictures like above. The invocation is asynchronous, so the ARexx program will run simultaneously with db. Use BLOCKINPUT and FREEINPUT to prevent possible problems here.
- * When the user hits Esc, Help, Enter or Tab to leave a string field. or manipulates a checkbox or cycle field.
This is called automatic invocation and can be used to format input in different ways. Here the invocation is synchronous so don't write slow scripts as they will lock out the user during execution.

You can specify ARexxfiles or stringprograms to be executed for each field in each view, for a whole view or for a whole db project. Currently you have to

edit the RFF lines to control db. Four new RFF tags have been implemented for this. See The RFF file format for more info.

There are readymade ARexx scripts for common uses in this archive. Try playing with the example project that makes use of the scripts.

The first time db is invoked it will open a port named 'DB.1'. If several programs are started they will get higher numbers ('DB.2', 'DB.3'..)

ARexx commands in db:

GETFIELD [<field>]	-The contents of the current field (or last activated field) or the specified field (internal fieldname) will be returned to the Result variable. For checkbox and cycle fields, their value as an ordernumber will be returned, not an X/space or a name.
PUTFIELD <var>	-The opposite, but you must specify a variable or ''. Currently you can't specify other destinations. Information is always put in the current string field.
CUT, COPY, PASTE	-Like their menu counterparts.
DIAL <number>	-Dials the given number using your modem or loudspeaker.
DISPLAYBEEP	-Flashes the screen to indicate that something is wrong.
RETRYINPUT	-Flashes the screen and reactivates the last gadget used. Use only in AUTORXSTRING and AUTORXFILE scripts, otherwise it makes no sense.
OKAY1 <message>	-Puts up a requester to inform the user of something. Nothing is returned from this function. The requester has one 'Ok' reply button.
OKAY2 <message>	-Puts up a requester asking the user to make some choice. The requester has two buttons. One 'Ok' button and one 'Cancel' button returning 1 and 0 respectively.
BLOCKINPUT	-Block user input and puts up a wait pointer to prevent user from modifying things while the ARexx program runs
FREEINPUT	-Frees user input, restoring the old pointer.
QUIT	-Quits db unconditionally (doesn't ask for saving)

(Use Options Results in your scripts to get results)

1.17 rff

Technical info
=====

The RFF file format

db is a general database program, ie it has not a fixed set of fields. But currently there is no GUI (graphical user interface) for editing the set of fields you want.

In order to make a custom database using db you therefore have to edit a db datafile manually. db stores all information other than fieldnames in the so called RFF lines of a file. (The fieldnames are stored in the first line of a file as in the ASCII-text standard for database files)

db uses an extended version of the standard ASCII database format called RFF. The difference between the two is that RFF is capable of storing information on things like layouts, visual fieldnames (as compared to internal fieldnames in standard ASCII file format), maximal fieldlengths, fieldtypes and more.

However an RFF file can be converted to a plain ASCII database file by just deleting all lines beginning with @RFF. In the RFF standard all RFF lines has to be in the beginning of the file, before any data lines, but after the first line which is the fieldname line, according to the ASCII database standard.

Note, in the early versions (1.1 & 1.0) db ignored the information specific to the RFF standard, but wrote files in an RFF compatible manner.

Normally there is one RFF line per View in a file, but the first RFF line describes internal information (as opposed to visual information) like maximum fieldlengths so an RFF file consisting of only two RFF lines, has one view, ie one window displaying just one way to look at the database.

Here is the format of an RFF line:

```
{<identifier>=<data>[,<identifier>=<data>]...[<tab>]}...<NL>
```

Example:

```
NAME=_Firm,OFFS=0,SIZE=37 NAME=_Name,OFFS=1,SIZE=14,NEXT=space
```

That should read: One or more comma separated 'identifier=data' items. Groups of commaseparated identifier=data items may also be tab separated. Case is not significant in identifier names. If you need to use a comma or space as data, enclose the data in "quotes"

The idea is that all information that belongs to a view is collected in a single RFF line, and all information that belongs to a single field in a view is collected between two tabs just like the field data itself.

IDENTIFIER SCOPE: GLOBAL AREA AND LOCAL AREA

An RFF 1.1 line is divided into two areas, the global area and the local area. The global area is the area BEFORE the first tab character, and the local area is the rest of the line. Identifiers put in the global area affects the whole view (or whole project for internal RFF lines). Identifiers put in the local area only affects that field. It works like global and local variables in computer languages like C and Pascal.

Some identifiers may appear in many different areas, even multiple times,

and some may only appear once in one special area. This is indicated next to the tag specification in the list further below.

To explain how the tags may be used I have included a Status: entry in the identifier list. Here is an explanation:

global	-can appear in the global area of an RFF line.
local	-can appear in the local area of an RFF line.
first	-must be the first tag.
internal	-can also appear in an internal RFF line.
internal-only	-must only appear in an internal RFF line.
required	-must not be omitted

Unknown identifiers are ignored but kept in the program for saving. This is somewhat like the IFF file way of thinking and allows for future enhancements without loosing backward compatibility. In v2.4, user is able to specify fields of checkbox and cycle type, not just string types. Such a file can be loaded into an old version of db, edited and then reloaded in a modern version of db without any error codes or lost information.

I also choosed RFF because it is a READABLE format. Readable to both men and machines of different types. To other database programs an RFF file should show up like a normal ASCII file with some funny records in the beginning, not that bad, right?

1.18 identifiers

Identifiers in RFF:

'@'RFF=<version.revision> (Ignore the 'quotes')

The RFF line identifier itself. Has a version and revision number as it's parameter. Must be the first identifier of an RFF line. A new version number tells an old RFF parser that so big changes has been made to this line that the entire line should be ignored.

Status: global, first, internal, required

TYPE=<type of data described>

This identifier describes what the current RFF line describes.

The parameters can be any of the following:

internal	-Information not concerning any view.
form	-This is a form view
list	-This is a list view (currently not implemented)

Status: global, internal, required

LNAM=<layoutname>

This is the name of the view to be specified. It will show up in the menus and in the titlebar of a database window when that view is selected.

(layoutname was the old name for a view)

If you omit LNAM the filename is used as name.

Status: global

TABSIZE=<number of characters>

New tag for db2.1!

When using a tabstep to visually separate fields (see NEXT tag below), this tag sets the distance between two tab positions (measured in characters).

If you omit TABSIZE a default of 6 is used.

Status: global

RXFILE=<filename>

New tag for db2.2!

Name of ARexx file to execute if user doubleclicks a string field or hits LAmiga+key, where key corresponds to the underlined character in a fieldname.

Status: global, local, internal

RXSTRING=<ARexx string program>

New tag for db2.2!

Name of ARexx string program to execute if user doubleclicks a string field or hits LAmiga+key, where key corresponds to the underlined character in a fieldname. This identifier has priority over RXFILE if both occurs in the same scope.

Status: global, local, internal

AUTORXFILE=<filename>

New tag for db2.2!

Like RXFILE but executes whenever user hits Esc, Help, Enter or Tab to leave a string field.

Status: global, local, internal

AUTORXSTRING=<ARexx string program>

New tag for db2.2!

Like RXSTRING but executes whenever user hits Esc, Help, Enter or Tab to leave a string field.

Status: global, local, internal

FLEN=<maximum field length>

This identifier describes the maximum allowed fieldlength. It is used to calculate the buffersize for the stringgadgets. Note: db will never allocate more memory or disk-space than needed to fit a string, so you may use large FLEN lengths without consuming space.

If you omit FLEN a default of 200 is used.

Status: local, internal-only

NAME=<visual field name>

This identifier describes not only the fieldname that should be displayed (could be different from the internal fieldname), but also describes what hotkey to be used to activate it's gadget. This is done by placing an underscore character before the character that is to act as a hotkey. Example: NAME=E_mail, will make the m key act as a hotkey to activate that field.

If you omit NAME, the internal fieldname is used (the first line of the file)

Status: local

OFFS=<offset to field in database>

This identifier is very important. It helps db "connect"

a field gadget to the right field in the database as the visual fields and the internal fields can be in different order. There can even be less visual fields than internal fields. This is of course only used on multiple views.

If you omit OFFS, the last OFFS+1 is used.

Status: local

SIZE=<visual fieldsize in characters>

This identifier is used to calculate the horizontal size of stringgadgets.

If you omit SIZE, a default of 25 characters is used.

Status: local

NEXT=<visual separator between fields>

This tag controls the position of the fields in the window. Not by x-y coordinates as in some other programs, but by telling db how to move it's invisible "pen" when a field has been drawn.

db starts drawing in the top-left corner.

Here are the currently defined parameters (newline is the default):

space	-move slightly to the right before drawing the next gadget.
tab	-move a tab-step to the right before drawing the next gadget.
para	-move two lines down before drawing the next gadget.

Status: local

CMNT=<"comment string">

This gives us the ability to insert invisible comments in databases.

Any string that contains spaces tabs or commas should be enclosed in "".

Status: global, local, internal

FTYP=<type of field>

Determines the type of the field. These are the currently defined types (string is the default):

checkbox	-displays a checkbox. A 0 or 1 is entered in the database.
cycle	-displays a cyclegadget with choices as specified below. The active choice is stored as an order number starting from 0 in the database.
calc	-a calculated field, contains a formula. (not implemented. Use ARexx instead)
external	-this field stores the filename of some external file. (not implemented. Use ARexx instead)

Status: local

SFMT=<string format>

This identifier gives special formatting of strings like the following (no formatting is the default):

upper	-all capital letters. (Not implemented. Use ARexx instead)
caps	-capital initial letters in words. (Not implemented. Use ARexx instead)
right	-right justified text. (db2.6)

center -centered text. (db2.6)

Status: local

CENT=<cyclegadget entry>

This identifier can occur several times and tells db what choices should appear in a cyclegadget. If you use FTYP=cycle and don't specify this one, you will get an error when the file is loaded.

Status: local, multiple

1.19 speed

A note about SpeedRender

Someone might wonder what the (NOSPEEDRENDER) ToolType in db's tool icon mean, well here it is:

I've tried to program db in such a way that it shall work with any future OS version. But in order to achieve resonable speeds in redrawing the window I have adopted a technique called SpeedRender. What SpeedRender does is to copy all the string gadget border pointers to a private list and then clear the GadgetRender field pointers in the gadgetstructs. Now Intuition doesn't have to redraw the gadgetborders everytime a gadget is updated (happens every time the user uses the dragbar). Setting this flag turns off SpeedRender.

1.20 history

History of changes

=====

95-07-11 v2.6 NEW_FEATURES:

- * An Italian, Finnish and Norwegian catalog is now added to the archive.
- * A handy ARexx script to simplify entering of dates is added to the ARexxdemos drawer.
- * Data in fields can now be shown centered and right-justified (nice for numbers) by setting the RFFtag SFMT For more information, refer to the section on the RFF file format. To see right justified fields, look at the "Music" and "Game reviews" examples.

BUGS FIXED:

- * It was not possible to specify (%) to search for empty fields (note: db uses AmigaDOS search patterns. You can for example specify ~(%) to search for non-empty fields) This very old bug was found by Fredrik Jervfors.
- * The Norwegian catalog incorrectly used 'S' as a shorthand for both 'Save' and 'Search'.

- 95-02-18 v2.5 NEW_FEATURES:
- * The sort function can now be set to consider language specific character sets using the LOCALESORT tooltype. Note: This might not work in your country if Commodore hasn't set up your language's specific character set. If set, characters like áâã will typically be sorted next to the character a. LOCALESORT sorts more than five times faster than the normal sort when sorting the 250 records-in-size "Music" example on an Amiga 4000 with the cache turned off (!). I hope that speed is enough for you. Currently I'm using an improved shaker-sort algorithm. This one works fast if only a few records are misplaced, but I know that the quick-sort algorithm wins when a lot of records are misplaced.
 - * A "fast find" function has been added to the space key. Upon pressing space, db will enter find mode and clear all fields automatically.
- BUG FIXED:
- * The routines to update the database and gadgets have been rewritten to eliminate possible destruction of the contents of the current record when jumping between sort and find modes. This is an old bug.
- 94-11-20 v2.4 NEW_FEATURES:
- * db now handles fields of checkbox and cycle type. as well as string fields. Take a look at the "Game Reviews" file in the Examples drawer.
 - * db now also dials numbers using the loudspeaker. Thanks goes to Andrew Leppard for the dial code. See Settings for more info.
(There are three new ToolTypes to accommodate this)
 - * Added an 'Output view with names' menuitem allowing the database to be outputted with the fieldnames of the current view preceeding each field.
 - * Added a MAKEBACKUP ToolType that makes db save a backup of the database every time a database is saved.
 - * Added an OKAY1 and OKAY2 ARexx command enabling the script programmer to put up information and selection requesters.
 - * On-line help somewhat improved.
 - * An Italian and Finnish catalog is now added to the archive, making db speak 8 languages!
(Where are you French and Spanish guys?)
- BUG FIXED:
- * Reading the contents of an empty string field from ARexx returned garbage.
- 94-09-28 v2.3 NEW FEATURES:
- * Several language catalogs added. See Preface .
 - * Clicking the closewindow gadget when in sort- or find mode returns db to normal mode instead of quitting as suggested by Edmund Vermeulen.
In normal mode this quits db as always.
 - * Added a HORIZBAR ToolType that places the record-
-

dragbar horizontally at the bottom of the window.
Suggested by Edmund Vermeulen.

- * Added a PUBSCREEN ToolType that opens db on the specified public screen or Workbench if not found.
- * Added an ESCQUIT ToolType that enables the Escape key to quit db (in normal mode).
- * Improved the GETFIELD ARexx command to accept an optional from-field as argument.
- * Added a RETRYINPUT ARexx command that can be used in AUTORX scripts to enforce correct input. RETRYINPUT will reactivate the last field entered.
- * Added CUT COPY and PASTE Arexx commands.
- * Closing large databases is now much faster. Records are de-allocated in the reverse order from how they got loaded thus helping exec in it's job.
- * Cut can now also be used in sort and find mode.

BUGS FIXED:

- * Pasting in sort mode now works.
- * A theoretical problem with trashing of global program-variables has been fixed (has never happened though)
- * Since v2.2: ARexx scripts could be started even in sort and find mode, that "feature" has been removed.

94-09-05

v2.2b NEW CATALOG:

A German catalog added. Thanks goes to Uwe Roehm.

BUG FIXED:

- * In v2.2 Fixed problem where an ARexx program (asynchronous commands only) could suddenly halt if it sent a command to db at exactly the same time as some IDCMP event occurred (mouse click etc).

94-09-03

v2.2 NEW FEATUERS:

- * ARexx support added. ARexx programs can be invoked when the user leaves a field, doubleclicks a field or presses LAmiga+key. Example ARexx scripts are included that shows text and pictures, plays sound, dials numbers and adjust fields in different ways.
- * Added an edithook routine to make GadTool string-fields smarter (like in ASL requesters). User may now perform the following operations IN string-fields:
 - ESC deactivates (without the ugly square)
 - Up/down keys (w shift) moves between records
 - Menus can be accessed with RAmiga+Key.
 - Doubleclicking performs a special action
 - Pressing Enter/Shift-Enter cycles around just like pressing Tab/Shift-Tab
- * Tab key now remembers the last active string-field and re-activates that one if de-activated instead of activating the first gadget.
- * Improved the parsers tolerance to "misplaced" spaces
- * db will now block input and show a waitpointer when needed (when loading, sorting..)

BUGS FIXED:

- * In v2.1: If the user made a change to a project by using cut, kill or paste and then quit. db wouldn't put up a project-not-saved warning.
- * Since v2.0: If the user tried to dial a number before any field had been selected, db would cause Enforcer read-hits.
- * Before v2.2: The Default Tool field of icons created by db wouldn't always get a correct path (to db) if db was invoked by doubleclicking on a project icon.

No Gurus so far anyway :-)

- | | | |
|----------|------|--|
| 94-08-20 | v2.1 | <ul style="list-style-type: none">* The return and enter keys can now be used to step between fields. (Use the NORETURNSTEP ToolType to only allow the tab key for this like before v2.1)* Added a TABSIZE tag to the RFF format to simplify layout work.* The RFF parser will parse all 1.x files not just RFF 1.1 files.* Saves and loades are faster. |
| 94-08-13 | v2.0 | <ul style="list-style-type: none">* Major changes. db is now a general database, not just a telephone and address database. db now uses the field- and layout specifications found in the datafiles (the RFF lines)* Multiple views implemented.* Added Amiga-W shortcut for "Save as..." menuitem* Commaseparated ASCII export implemented (user request)* When exporting records: filtering of records can be specified.* Custom fontname and fontsize can be specified* db now moves one titlebar down from the top left corner of the screen when zoomed.* NOICONS can be specified to suppress icon saving (user request)* before 2.0 db didn't intentionally remember the last exported filename. This has been changed.* Before v2.0: If the user made a change to a project and performed an ASCII export and then quit. db wouldn't put up a project-not-saved warning. This has been corrected.* Fixed bug where db sometimes sorted records incorrectly in projects where there were records which had blank fields which were also part of the sort order. |
| 94-06-04 | v1.1 | <ul style="list-style-type: none">* Added modem-dialing feature.* db now runs from Shell as well as from Workbench* Settings can now be made from Shell using Commodore's template parsing or from db's icon using ToolTypes.* db now moves to the top left corner of the screen when zoomed.* Gadget positions and sizes somewhat adjusted.* Selecting Cut won't incorrectly put up the delete warning requester.* minor code changes made. |
-

94-03-04 v1.0 Initial release.