db

COLLABORATORS

	TITLE : db	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		January 19, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

db

1.1 db - A small and fast database program

```
                          Welcome to
        db 3.1
  the database program
    Table of contents
    =================
```

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

Since v3.0 this program is ShareWare. It no longer has the save function
disabled if unregistered (for evaluation purposes), but instead shows a
ShareWare reminder for at least 10 seconds when db is started. You will have
to
register
in order to receive a password that will remove
the ShareWare reminder. By registering you support the continued developement
of this product. Please support it.

Congratulations to all of you who've sent me a postcard before 96-01-06, you are
hereby entitled to register for half the fee!

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 :-) Seriously I'm just asking for one copy. Magazines seem to be
lazy here as db has appeared on a number of magazines, but I haven't received
a singe one but "Amiga Format".

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.netnet.net (biz/dbase)

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. I especially look
for translations of the "Addresses" example file.

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
GSM: +46 70 755 34 33
email: david-ek@dsv.su.se

Thanks to all of you who've given me suggestions and found bugs, among those
Postcard senders, Anders Callertun and Mikael Östergren.

Icon artwork:
Program & Project icons by Mikael Östergren
Additional icons by Roger McVey (r.mcvey@genie.geis.com)

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

The excellent textfield gadget was made by Mark Thomas, thanks Mark!
Here is how to contact him:

    Mark Thomas
    9036 N. Lamar, Apt. #125

```
    Austin, TX  78753

    or

    mthomas@zilker.net

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: letowski@ci-1.ci.pwr.wroc.pl

Dutch translation by Edmund Vermeulen
  email: edmundv@grafix.xs4all.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 Ghezzo (50%)
  email: pghezzo@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@sn.no

Spanish translation by Gonzalez Rocha
  email: alu756@csi.ull.es
```

1.4 register

```
How to register
---------------

Since v3.0 this program is ShareWare. It no longer has the save function
disabled if unregistered, but instead shows a ShareWare reminder for at least
10 seconds when db is started. When you register, you will receive a password
that will remove the ShareWare reminder.

The normal fee is 20 USD (see orderform). This is how you register:

1. Just fill in the orderform that is enclosed in this archive.
2. Send this form and any postal money order or bills to:
```

David Ekholm
Mantalsvägen 33
s-175 43 JÄRFÄLLA
SWEDEN

3. I will return a name, serial number and password for you. Use this
   information together with the program "Keymaker" (in this archive)
   to make a keyfile that will remove the ShareWare reminder. Keymaker
   will write the keyfile to l: but you can later on move it to s:
   or the same directory as db resides in if you prefer that.

Here are some guidelines on choosing payment method:

 - SEK / USD / DEM and GBP Bills of any kind are gladly accepted. No other
   currencies are accepted. Putting bills in letters is quite safe, but
   remember to put enough paper in the envelope so that the bill cannot
   be easily spotted simply by using a lamp. If you want to be extra sure,
   use registered mail. This is probably the cheapest and easiest way of
   paying the fee. It is probably also the fastest way.

 - Please, only *INTERNATIONAL* Postal Money orders in SEK can be cashed.
   NO other money orders are useful.

 - *NO* national cheques can be accepted. Since I cannot cash foreign
   checks or money without having to pay substantial fees to the bank.

 - Swedish residents can use PostGiro 624 18 13-2. Foreign transfers
   via PostGiro cost _me_ 25SEK, so add 25SEK to the fee if you pay
   this way (= 155 SEK).

    (As of this writing (January 1996), one US Dollar is ~= 6.465 SEK)

                Please allow 1-3 weeks for password delivery.

Regards

/David Ekholm

1.5 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 dozzens 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. Since v3.0 I've added

GUI-based database design
 to db that
releaves the user from the work of specifying the layout by writing RFF code
(See
The RFF file format
).

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.

1.6 features

Feature List

A partial list of db's features include:

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

1.7 future

Future plans

The following are my plans for future versions of db:

    o Completing the GUI-design part.
    o Adding datatypes so pictures can be displayed in a db window.
    o (A simpler-to-read file format.)

1.8 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 city car.

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 3.1 and is written in SAS C 6.55.

1.9 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.

1.10 settings

```
                Settings
--------

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

  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. These two
                                 ToolTypes have to be specified together.

  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.

  NOBORDER                      -Display string fields without the fancy
                                 v37 borders. Use this if you think there
                                 is to much GUI in your database.
                                 Note: this option uses
                 SpeedRender
                .

  HIGHLABEL                     -Display fieldnames in highlighted
                                 colour instead. This option goes well
                                 together with the NOBORDER option.

  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.

  LOCALESORT                     -Makes db use the locale.library string
                                  compare function when sorting making db
                                  handle language specific characters
                                  correctly. However, due to a bug in the
                                  Swedish language support in the OS, Swedes
                                  should NOT use this ToolType. LOCALESORT
                                  can also improve sorting speed a lot.

  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 phonenumbers when
                                  dialing. Default is ATDT (Hayes modems)

  DIALPOSTFIX=<postfix string>   -As the above, but this will be added after
                                  the phonenumbers (+ 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.
```

1.11 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.

1.12 window

The window

The db window usually 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 display one record at a time that we call the current record.
However you can see one selected field from many records at a time using
the "Browser" which is a window with a listview inside (v2.7).
db can work in one of three modes:

- Normal mode. Enter records and look at the database here.

- 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)

- 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.

1.13 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)

1.14 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 five types: string, checkbox, cycle, text (read-only) and textfield type (multi-line). They can be activated in one of three ways:

- Clicking the mouse over the desired field.
- Pressing the key that corresponds to the underlined character in the field name. (Will toggle checkbox fields and cycle cycle-fields)
- 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 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, it is 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.

1.15 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.
```

1.16 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 draging 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 window 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.

A "fast find" function has been added to the space key. Upon pressing space, db will enter find mode and clear all fields automatically. (v2.5)

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.

Note! since v2.10: If you leave all fields empty db will restore the record order to the state the database was when it was loaded. A kind of "undo sort".

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 concist of doubleclicking the number you want to dial. See the example scripts.

Keyboard fans, note this: As an alternative to doubleclicking, you can use L-Amiga + the hotkey defined for the desired field (marked by an _underscore in the field name).

Action/Browse...

The browser is a window with a listview that allows you to browse through
the database, seeing fields from more than one record at a time.
To use it, select a field to browse on, and select Browse...

You can also doubleclick on a field to start the browser or use L-Amiga
+ a hotkey if one is defined for that field.
(unless an ARexx command is installed for doubleclicking).

The main window is disabled when using the browser, but the window stays
opened so you can see all fields of the selected record as the browser is
limited to only showing one field from each record at a time.

The following might not be obvious:
· Click once using the mouse to see that record in the main window.
· Doubleclick to select a record AND leave the browser.
  (The Return key works as well)
· You can use the arrow keys to browse. Try combining Shift-arrow and
  Alt-arrow to see the effect.
· The browser has a 'completion' function: Just type the string you
  look for and the browser will search simultaneously. Use Del to
  clear the completion buffer and backspace to delete one character.
  Entered text will be shown in the titlebar of the browser.
  Use the Tab, Shift-Tab key sequence to jump between multiple matches.
· You can leave the browser by pressing Escape.

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/Field definition...

THIS FEATURE IS CURRENTLY UNDER CONSTRUCTION AND THEREFORE DISABLED

This is the door to db's field definition section. It allows you to add,
edit and remove fields using a user friendly GUI.

Note! You are just changing the internal fields here. Use "View design..."
to design the visual apperance of your fields.

Settings/View design...

This is the door to db's View design section. It allows you to add,
edit and remove visual fields in the current View using a user friendly GUI.
The visual fields are handled similar to how character are handled in a
wordprocessor. This is WYSIWYG (What You See Is What You Get).

Note! You don't work with any real fields here, just a visual presentation
of a subset of the database fields. Use "Field definition..."
to define the database fields. Click
here
to read more on view design.

ARexx/Execute ARexx command...

This item will bring up a standard ASL requester for executing ARexx scripts.
Below the horizontal bar is space for user defined ARexx scripts. See

Install a custom ARexx menu
.

1.17 arexx

ARexx support
---

db's ARexx support has finally been improved to allow users write scripts that
check for duplicate records, implement simple relations, connect db to WWW
browsers and the like.

Previously db didn't have a full-featured ARexx support as I intended to release
a shareware product later. I still do, but that version will have other
advantages. The old ARexx support is/was not bad at all. It 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 three ways:

* When the user selects an item from the ARexx menu or presses a corresponding
  function key. (db2.10) See
below
for help on installing a custom ARexx menu.

* 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. 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'..)
You can change the basename (i.e. "DB") to something else by specifying the RFF tag RXPORTNAME in the database file.

1.18 installarexxmenu

Install a custom ARexx menu

The ARexx menu is a new feature since db2.10. It allows you to add ARexx commands of your choice to it's items. By doing it this way, the commands are no longer "tied" to a specific field. Add the commands you lack in db here.

To make a custom ARexx menu you currently have to use a text editor and write one line of RFF code into the file. An example will show this clearly:

Let's say you want a menu looking like this:
  Merge
  Check for duplicates

Suppose you have two ARexx files for these functions on your harddisk. They are named Merge and dupl. The single RFF line that installs these commands into the menu could then look like this:

'@'rff=1.2,type=rxmenu rxfile=Merge rxfile=dupl,name="Check for duplicates"

(Ignore the single quotes. Note there are tabs above, NOT spaces)

In the first case we wanted to use the filename "Merge" in the menu, so we just specified the filename by using the RFF tag rxfile. In the second case we didn't want the weird looking name "dupl" in the menu so we gave this option a new name. You can also use the tag rxstring to specify a stringfile to be executed instead of a disk based file. This way you can enter ARexx commands directly into the RFF code.

The ten first menu items are automatically tied to the function keys.

1.19 arexxcommands

ARexx command list:

This is the list of all currently supported ARexx commands in db.

```
Some commands perform different tasks depending on if they are called
with or without parameters (e.g. 'CurrentRecord' will return the number
of the current record and 'CurrentRecord <x>' will set the current
record to <x>)

The commands return answers in the variable 'Result'. In case of a warning
condition, 'RC' will be set to 5. Don't forget to specify 'Options Results'
in your scripts to get results in these variables.

Record related commands:
------------------------
Add
  Like its menu counterpart.

Kill
  Like its menu counterpart.
  (Note: this command will clear the fields when in FIND or SORT mode)

FirstRecord
  Jump to the first record in the project. Return "1"

NextRecord
  Jump to the next record in the project and return that record number.
  RC will be set if there are no more records.

CurrentRecord [<number>]
  Return the number of the current record if no argument is given.
  Jump to the specified record number otherwise. RC will be set if <number>
  is illegal.

RecordSum
  Return the number of records there are in the database.

FindFirst
  Search forward from start for a matching record. (Set search criteria in
  'Find' mode by filling the fields first).
  If there is a match and db is in 'Find' mode, db will return to 'Normal'
  mode automatically. Return the number of the current record.
  If there is no hit, db will set RC.

FindNext
  Search forward for a matching record. Otherwise like FindFirst.

Cut
  Like its menu counterpart.

Copy
  Like its menu counterpart.

Paste
  Like its menu counterpart.

Merge
  Merge the current record with the contents of the clipboard.


Field related commands:
```

```
-----------------------
(These commands work on internal fields. There are no "Views" in ARexx.)

FirstField
  Set the "current field" to be the first field and return the name of the
  first field.

NextField
  Set the "current field" to be the next field and return the name of this
  field. Set RC if there are no more fields.

CurrentField [<field>]
  Set the "current field" to be <field>. If no argument is specified, return
  the name of the "current field". From start, "current field" is the same
  as the last activated field.

GetField [<field>]
  The contents of the current field (usually the 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>]
  Put <var> or nothing into the current field. Use 'CurrentField' first
  to specify another destination field.


Other commands:
---------------
Mode [<mode, one of "Normal", "Find" or "Sort">]
  Set db in one of its three main modes. This works like in the menus.
  Will return the name of the current mode if called without parameter.

Save
  Save the current project to disk.

Sort
  Sort the database based upon the sortorders specified in 'Sort' mode.

Quit
  Quit db unconditionally (don't ask for saving)

BlockInput
  This command does two things:
  1. Block user input and put up a wait pointer to prevent
     user from modifying things while the ARexx program runs.
  2. Turn off automatic GUI updates as you control the database from
     ARexx. db becomes 'quiet'. Updates can now be performed on demand by
     issuing UpdateGUI.

FreeInput
  Free user input, restoring the old mouse pointer and turn on
  automatic GUI updates.

UpdateGUI
  If you issue BlockInput. The GUI won't be updated when you control db
  through ARexx unless you explicitly ask for it by issuing this command.
```

```
WindowToFront
  Bring db's window to front.

ScreenToFront
  Bring db's screen to front.

ActivateWindow
  Activate db's window

DisplayBeep
  Flash the screen to indicate that something is wrong.

Okay1 <message>
  Put up a requester to inform the user of something. Nothing is returned
  from this function. The requester has one 'Ok' reply button.

Okay2 <message>
  Put 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.

CurrentGadget [<gadgetname or index>]
  Select the gadget with the matching name or index as the current gadget.
  The current gadget is the one that gets activated when the user presses
  the TAB key. Return the index of the current gadget.

CurrentView [<viewname or index>]
  Select the view with the matching name or index as the current view.
  Return the index of the current view. (db3.1)

RetryInput
  Reactivate the last gadget used. To activate another gadget, use
  CurrentGadget first. This command is often used to enforce correct input.

Dial <number>
  Dial the given number using your modem or loudspeaker.

(Remember to use 'Options Results' in your scripts to get results. Also:
 The DOS commands TCO, TCC, TS and TE help a lot when you debug scripts.)
```

1.20 viewdesign

```
View Design
-----------

The purpose of this part of db is to allow you to add, edit, rearrange
and remove visual fields (gadgets) in the current view in a simple way
by using common mouse and key operations (db will write the RFF code).

Quick Overview
You pick new visual fields from the new toolbox to the right. They may
later be edited with a simple doubleclick. To Resize and move them
around, point at them and drag the mouse (the right border of any visual
field is resize-sensitive).
```

You can cut, copy and paste visual fields as expected and there is
also an undo to help you out.

Visual fields are either separated by a space, tab, newline or double
newline. Use the keyboard to add and remove these separators.

Limitation
In this release you still need to use a text editor to specify the
INTERNAL fields, but that's simple; just write all names in one line,
separate them with tabs, add a newline, save it and load it into db.
I intend to fix this later on. "View Design" isn't localized either,
but will be when the code is completed. Also note that there currently
is no support for making fields of type textfield and text using this
GUI either (Add the RFF identifiers FTYP=textfield and ROWS=<number of
rows) in the meantime.

Quick tutorial
This text will guide you through the steps needed in order to make a small
database.

1. Decide which fields you will need initially. Write the fieldnames to the
   first line of a file like this:
   name address zip phone

   (Don't forget to separate the fields by tabs and to add a newline.
   Don't use the editor Ed as Ed destroys tabs)

2. Save that one-liner to a file and load it into db.

3. Select "View design" from the "Settings" menu.

4. You are now faced with a blank window and a toolbar window to the right.

5. Select the topmost tool to create a plain string field.

6. A new window will appear. Here you must select which of the fields you
   wish to "connect" this "visual field" to. Select the "Select..." button.

7. A new window will appear with a listview containing your fields (in our
   case "name", "address", "zip" and "phone". Select one of them by pressing
   enter (the cursor keys also works) or by doubleclicking.

8. Your "Visual field" is now connected properly. Adjust it's size and
   name/title as you wish and select Ok. Your "visual field" is created.

9. When you proceed in the same manner with the other fields you will need
   to arrange the fields. This is very simple. In the main window you will
   find a cursor. You can move the cursor about and insert newlines, tabs
   and double newlines as you please by using the keyboard but there can only
   be one separator between each field, i.e. you may not use two successive
   tabs. Fields can also be moved about and resized by using the mouse. Pay
   attention to the shape of the mouse pointer while you move it across your
   window.

10. Just doubleclick on a field if you like to edit it further.

11. When your are satisfied, just return to db and save the new database
    design.

1.21 textfield.gadget/User_Docs

```
Using the textfield gadget
--------------------------

The textfield gadget is a new fieldtype since db3.1. It allows you to enter
information using several lines of text, -a small editor one could say.
The code for the textfield gadget is not my own. It is an external BOOPSI
object by Mark Thomas. It is implemented as a library called textfield.gadget.
You have to have the file textfield.gadget in your sys:Classes/Gadgets drawer
or in db's Gadgets drawer (the default), otherwise textfields will show up like
ordinary string fields instead.

Caveats
The textfield gadget doesn't behave exactly like an ordinary db gadget (I'm not
the author of that code). Keep the following in mind when using it:

* You have to press tab or use the mouse to leave a textfield gadget.
* You can't doubleclick a textfield gadget in order to run ARexx scripts.
* The  _underscore character won't show up below the label as the textfield
  gadget doesn't support a real gadget label.


    DOCS FOR USERS

        You can mark text for cutting, copying, and erasing by simply clicking
        and dragging.  Hitting alphanumeric keys replaces the text that is
        highlighted.  Hitting cursor keys moves you to the front or end of the
        highlighted text.

        If your cursor is already somewhere in the textfield, you can hold the
        SHIFT key and click to mark the text from the current cursor position
        to the place where you clicked.

        And the last way to mark text is to double-click, which will mark the
        word you clicked on.  If you didn't click on a word, but rather you
        clicked on spaces, the whole block of spaces is marked.  And if you
        clicked on word delimiters, the whole block of delimiters is marked.

        While you drag to scroll, the farther away from the gadget your mouse
        pointer is, the faster the gadget will scroll.

        For key sequences, the Amiga Style Guide was followed.  Anywhere the
        undo buffer is mentioned, the statement is only valid if the
        UndoStream is supplied (see tag section below).

        Key Sequence              Function
        ------------------------------------------------------------------
               TAB                Activate next gadget (if GA_TabCycle)

         SHIFT TAB                Activate previous gadget (if GA_TabCycle)
```

SHIFT CURSOR UP	Move to the top line in the current page, or scroll up one page if cursor is on top line
SHIFT CURSOR DOWN	Move to the bottom line in the current page, or scroll down one page if cursor is on top line
CTRL or SHIFT CURSOR RIGHT	Move to the right end of the current line
CTRL or SHIFT CURSOR LEFT	Move to the left end of the current line
SHIFT BACKSPACE	Delete all text to the left of cursor on the current line
SHIFT DELETE	Delete all text to the right of the cursor on the current line (in block cursor mode this also includes the highlighted character)
CTRL CURSOR UP	Move to the top line of the text
CTRL CURSOR DOWN	Move to the bottom line of the text
ALT CURSOR RIGHT	Move to the next word (using the delimiter characters provided by the programmer)
ALT CURSOR LEFT	Move to the previous word (using the delimiter characters provided by the programmer)
ALT CURSOR UP	Move to first character in gadget
ALT CURSOR DOWN	Move to last character in gadget
ALT BACKSPACE	Deletes the word to the left of the cursor starting at the current cursor position
ALT DEL	Deletes the word to the right of the cursor starting at the current cursor position
CTRL X	Deletes the whole line that the cursor is on
RAMIGA [	Switch to left justification (if TEXTFIELD_UserAlign is set)
RAMIGA \ or RAMIGA =	Switch to center justification (if TEXTFIELD_UserAlign is set)
RAMIGA]	Switch to right justification (if TEXTFIELD_UserAlign is set)
RAMIGA E	Erase all text in gadget (saved in undo buffer) (no read-only)

RAMIGA V	Paste text from clipboard to current cursor position (no read-only)
RAMIGA A	Mark all text
RAMIGA U	Undeletes (pastes) the last block of text marked, or recover from RAMIGA E (no read-only)

When text is highlighted the following keys have functions:

BACKSPACE	Erase marked text (saved in undo buffer)
DEL	Erase marked text (saved in undo buffer)
RAMIGA X	Cut marked text to clipboard (no read-only)
RAMIGA C	Copy marked text to clipboard
RAMIGA V	Replace marked text with text from clipboard (save marked text in undo buffer) (no read-only)
(any text key)	Replace marked text with that character

1.22 rff

Technical info
==============

The RFF file format
----

db is a general database program, ie it has not a fixed set of fields.
In order to make a custom database using db you have the option of either
using the built-in database design system (v3.0) which features an easy
to use graphical user interface (GUI) or write the specification manually
in a plain text file. The first method is a lot simpler, but there are
those who prefer the non-graphical approach. This text is for you.

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 format, 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 fieldlenghts 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 was 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 designed 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?

However I've noticed that the readability isn't as good as it was intended to be. This is especially true for large views where an RFF line easily becomes thousands of characters wide. Therefore I've been thinking of changing this format to use newlines instead of tabs. The code looks much better then, but I would then loose compatibility with older versions of db (sigh!). Maybe it's better to spend time improving the GUI for the design instead. That would solve the readability problem.

1.23 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
  rxmenu   -Here you install entries for the ARexx menu (db2.10)
  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

XPOS=<x position>, YPOS=<y position>
If both these tags are present, db will position the window for each view accordingly. (db3.1)
Status: global

TABSIZE=<number of characters>
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. (db2.1)
Status: global

RXPORTNAME=<portname>
Basename of db's Rexxport. db will add .1, .2 and so on to the basename if needed. (db2.9)
Status: global internal only.

RXFILE=<filename>
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. (db2.2)
This tag can also be used in the custom ARexx menu (db2.10)
Status: global, local, internal

RXSTRING=<ARexx string program>
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. (db2.2)
This tag can also be used in the custom ARexx menu (db2.10)
Status: global, local, internal

AUTORXFILE=<filename>
Like RXFILE but executes whenever user hits Esc, Help, Enter or Tab to leave a string field. (db2.2)
Status: global, local, internal

AUTORXSTRING=<ARexx string program>
Like RXSTRING but executes whenever user hits Esc, Help, Enter or Tab to leave a string field. (db2.2)
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=<object name>
This identifier is used to name different objects in db.
Before v2.10 it was only used to specify a visible field name (as opposed to internal fields). From now on this tag is "context sensitive" i.e. the exact meaning of this tag is dependent on the situation it is used in. In db2.10 this tag also specifies the name of a user defined menu item in the ARexx menu. However It's most common usage is to define the label that should show up next to a field in a view but it 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.
I you omit NAME, the internal fieldname is used (the first line of the file)
Status: local


PLACE=<direction>
This tag controls where the label of a field is to show up. db defaults
to put the label to the left of the field, but you can specify "above"
to put it above the field instead. (db2.10)


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):
  integer    -Do a numeric, not an alphabetic sort on this field.
              (internal-only, v2.8)
  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.
```

```
  text       -displays a textgadget. A textgadget is a read-only stringgadget.
              However it can be written to from ARexx. (db2.10)
  textfield  -displays a textfield gadget. This is a multi-line string gadget.
              (See the ROWS identifier) (v3.1)
  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, internal and global (v3.0)


ROWS = <number of rows>
This identifier determines how many rows a gadget should use. Default is
one line. Use this identifier together with the textfield gadget above.
Status: local, internal and global. (v3.1)


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
```

1.24 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.
```

1.25 history

```
                History of changes
==================

96-03-05  v3.1       NEWS:
                     * Multi-line fields are now supported using the
                       textfield.gadget (included) by Mark Thomas. Check out
                       the "Game Reviews" and "Windsurfing95" examples and
                       you'll see what I mean.

                     * You may now fix the window position for each view by
                       adding XPOS=<x position> and YPOS=<y position> to the
                       RFF lines of your database.

                     * Added the ARexx command 'CurrentView' to control views
                       from ARexx.

                     SHAREWARE NOTE:
                     * db is still ShareWare, but the save save limitation is
                       removed. Instead db will display a ShareWare reminder
                       for at least 10 seconds when started without a keyfile.


96-01-06  v3.0       NEWS:
                     * You may finally design your own databases from within
                       db instead of manually enter RFF code in a textfile.
                       This should be a MAJOR relief to most users. Read
                       about this in
                View design
                .

                     * The field type may now be specified internally
                       instead of per-view. This is often preferred,
                       especially in databases with multiple views.

                     SHAREWARE NOTE:
                     * db is ShareWare starting from this release. Please
                       read the
                text
                 on how to register.

                     LIMITATION:
                     * The new graphical database design part isn't 100%
                       completed in this release. You still have to define the
                       internal fields and cycle entries manually, but I can
                       assure you that the tedious work of making the database
                       layout is gone.


95-11-22       v2.10 NEWS:
                     * There is now a user defineable ARexx menu in db.
                       Please look at the "Addresses2" example.

                     * Needed RAM has dropped by 40% per record for a
                       typical file as a result of a new memory handling
```

scheme.

* Read-only string fields are now supported (They can be written to from ARexx though).
  These fields are great to use as calculated fields.
  Their type in db is "text" (FTYP=text). Please look at the example "ARexxdemos/MagicFieldsIII".

* The visible name of a field can be positioned above the field by specifying PLACE=above in the RFF code.
  Please Look at the example "ARexxdemos/MagicFieldsIII".

* It is now considerably faster to quit a large database.

* Switching between views is now faster as db no longer closes and reopens the window.

* If you leave all fields blank when you sort a database it will be un-sorted to the state it was in upon loading. A kind of "undo sort".

* Smaller code than in 2.8. =)

BUG FIXED:
* Reading the contents of an empty string field in a newly created record from ARexx incorrectly resulted in a warning result.

95-09-27 v2.9 NEWS:
* The ARexx support has been improved a lot.
  db now supports more than
30 ARexx commands
.
  You are now able to easily make advanced applications with relations and build complete economy systems.
  (There is an invoice system comming for Swedish users.)
  Check out the new example in Examples/Relations.

* All icons are NewIcons from now on. NewIcons look much better than the original ones as they seldom show up in wrong colour. To see the new icons you have to install the NewIcons package. Otherwise the icons will look like before with the exception of some funny tooltypes in the icons.
  (You can get NewIcons at Aminet:util/wb/NewIcons.lha)

* I've included the original .ct files next to the .catalog files for each country to make it simpler for helpful and skilled translators to provide me with language updates as I release new versions.

95-08-26 v2.8 NEWS:
* Sorting now uses the superior Quicksort algorithm (ACM 271). Alan Wigginton (author of QuickFile)

notified me on the terribly bad sort benchmarks
he got when sorting a large database using db.
It could take hours on an Amiga 500. Now even large
databases should sort in seconds. If you get into
trouble when sorting a large database now, please
set the stack to a higher value.

* db now sorts numbers correctly if you add the
FTYP=integer RFF tag to the internal field
specifcation. Look at the "Music" example!

* The whole database will be sorted on the active
field if you hold down the shift key while activating
the Browser. (A shortcut to sorting, one could say.)
(Some users want to have just the Browser list
sorted and some users want to have the whole database
sorted simpler. This is an intermediate better-than-
nothing solution until I figure out the best way to
do it.)

* Minor speed improvements when entering and exiting the
Browser has been made.

BUGS FIXED:
* NEWS, new bugs they say...
When using the keys to navigate in the Browser under
OS 2.0 or 2.1 (below v39) the system locks completely.
My first Guru in db :-(This has been fixed.
Thanks to Alan Wigginton for telling me about it.
(My ordinary testers all seem to use v39+ today.)

* Since v2.6 db didn't sort the Swedish ÅÄÖ characters
correctly. This is fixed.
(Don't use the LOCALESORT ToolType if you want Swedish
characters sorted correctly as there is a bug in
the Swedish language support in the OS. But DO use
LOCALESORT otherwise as this speeds up sorting and
should make db sort your language specific text
correctly.)

95-08-16 v2.7 NEWS:
* Added a browser window that consists of a listview
where you can see fields from many records at a time.
The browser is handy when you don't even remember a
keyword to search for or when you want an overview of
the database. One way to start the browser is by
doubleclicking a field. There is on-line help also.

* Added two new tooltypes to further control the
apperance of fields (HIGHLABEL & NOBORDER).
See
Settings
for more info.

* The following catalogs are updated (please copy them to
your directory): Norwegian, French, Polish, Swedish.

(Text that has changed will show up in English if
you use an old catalog.)

95-07-11 v2.6 NEWS:

* 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 NEWS:

* 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 áàã 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 NEWS:

* 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 NEWS:
* 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 occured (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

* 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

* 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
* 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.