Timetable For Acedb 1 Day Introduction And 3 Day Workshop

June 19 - 22 2001

Day 1 Novice User Day

AM Navigating around window displays text, genetic map, physical map, grids, forest display and pathways

Dumping out data

Sequence display, DNA analysis, pepmap and genefinder

Blixem and dotter alignment tools

PM Querying the database

Introduction to other executables

Acedb resources

Acebrowser

Day 2 Advanced User Day

AM Introduction to using acedb, browsing, searching, displays, data-types

Maps and views

Displays for sequences

Dumping out DNA, proteins and others

PM Data structures: simple models

Querying, making tables, manipulating keysets

Editing objects interactively

Day 3 Curator Day

AM
- Acedb directories and specification files
- Reading in EMBL sequences and proteins
- Editing using .ace files
- Ace tools: acediff
- Theory of models and classes
- Changing models and classes
- Creating a new class

PM
- Special tags
- Using special tags: Making new maps and views
- Gmap exercise
- Grid exercise
- Fmap and method exercise
- Other user configuration
- Tace

Day 4 Administrator Day

AM
- Setting up a new database: Acquisition from ftp site, Installing software and data
- Server/client
- Other interfaces: giface and webace
- Binary versions
- www.acedb.org and Help
- AQL

PM
- Complex queries
- Designing your own database
- Windows version demonstration

Course organiser Sylvia Martinelli

Table Of Contents

The Acedb Course For Database Curators And Administrators

This course should enable you to set up your own database to use ACEDB software from the very beginning or enable you to take over the management of an existing database. After a brief introduction the courses starts with a description of the various window displays and ways of displaying these objects. A set of exercises has been designed to illustrate all that. These can be followed independently or they may be given as a demonstration depending on the facilities available.

The next part of the course concerns the acquisition of the software and the options acquisition of data from the Sanger Centre ftp site. After getting the necessary files, you will be able to run and INSTALL script to make the ACEDB database directories, untar and uncompress Update files. Then you will be able to run your own copy of the miniature worm database.

A description of the major directories and some of the most critical files will be followed by examples of simple models (meta-data), such as People and Publications. This will serve as an introduction to interactive editing of objects into the database.

Another exercise has been set for you to do independently, to familiarise you with the Query Language and TableMaker facilities. In these examples, both the questions and answers are given. The database can be queried from 4 different windows (1) Query_by_example (2) Query Builder (3) the plain query facility (4) Table Maker. All 4 will be demonstrated, if possible, although Query_by_example often collapses.

You can try dumping out data in ace format, a technique which is very useful for bulk editing outside ACEDB which can then be read back in.

Being able to handle keysets is an essential skill for a curator and is a natural adjunct to querying the database as well as data dumping out or reading in data.

As well as learning to edit objects interactively you will learn how to write .ace files using a text editor, then read these files into ACEDB. As well as reading in whole new objects you will see how to make minor alterations to existing objects and how to rename objects. The use of an executable file called acediff will be demonstrated. It can be used in data preparation, for comparing two files or two database.

The central part of the course is discussion of data modeling, class structure and magic tags; also a review of all the essential database directories and files. This is accompanied by an exercise in which an existing model is altered and new model or class added to your database.

Since the software was written with the storage and display of DNA sequences in mind, an important part of the course involves looking at the fmap or sequence display, learning how to customise this display, how to read in DNA sequences, display sequences, analyse them with the internal tools, how to dump out DNA and protein sequences, use of Genefinder package. Converting between EMBL format and .ace format will be included.

Another display which can be customised to a far greater extent is the Gmap (Genetic Map) display which has been used in the worm databases for displaying positions of Mendelian genes, genetics rearrangements, Contigs from the physical mapping project and the positions of predicted genes. It is also used to display clones selected for the sequencing project and their progress through the analysis. Different degrees of customisation will be shown.

All the aforementioned techniques will be shown on single user database run by XACE, the x-windows interface. ACEDB has several other interfaces: the text only interface TACE, the GIFACE interface, the WEBACE interface. ACEDB can also be used by multiple users through either the ACESERVER and ACECLIENT modules which use TACE or the GIFACESERVER and CLIENT. You will be able to run some commands and queries using the TACE interface and

possibly the ACESERVER/CLIENT interface. If time permits you will be asked to make your own database in which you can model any kind of data that you choose.

The Acedb Course Directories

Within your home directory, you will have a course2000 directory (or similar)

which contains subdirectories used for various parts of this course :

exercises/

diff/

add_data/

edit_models/

embl/

blast/

Gmap_teaching/

Fmap_teaching/

Grid_teaching/

models.used_in_code

The text explanations in these directories have not been updated in synchrony with the printed booklet which has recently been re-printed.

The course and mini-database will be placed on the Sanger ftp site

in ftp/pub/acedb/.

EXERCISES ON KEYSETS

(For the examples in question 2, you need a basic knowledge of the Class model for Author.)

1. Copy a keyset

Get a keyset of objects such as Gene or Journal, make a copy by pulling down the background menu with the right hand mouse button and selecting 'copy whole keyset'. The original will be highlighted pink at the top, the new copy will be white at the top indicating that it is not the selected keyset. You can make the copy the selected set by clicking your left mouse on it.

Get a keyset of authors (not all of them), then try using the **Select/modify** menu on the keyset window.

Select some of the authors, **copy selected items** to another window and work on this smaller set.

Select some authors. Choose **remove selected items**.

Select some authors. **Clear** or **reverse** the selection.

2.Logical operations

a) MINUS : see example (8) for ALL genes MINUS lethal genes.

For the next three exercises first get ALL authors and make a copy of the keyset.

b) AND : perform a query on Keyset 'ALL authors' to get Authors with email addresses (Tag = E_mail) and keep a copy of the result. Then perform a query on Keyset 'ALL authors' to get Authors with Laboratories and keep a copy of this (or get Authors with sequences Tag = Sequence). Highlight/Select one of these two subsets, move to the other and perform an AND operation. Look at the result. It should be authors which occur in both.

c) OR : similar to above but make subsets of Authors with Laboratories, then Authors with Sequence. Perform OR operation. Should get authors with either OR both.

d) XOR : try (c) above again but use XOR operation. What is the difference?

3. New keyset and add-remove

Get keyset of authors. Choose on background menu of window the New keyset option.

Choose Add-remove from same menu and try clicking on some authors in main keyset window.

Try it on the new keyset window. Try saving your new keyset or dumping it.

4. Select / Modify keysets

Get a list of genes. Highlight a few then pull down the Select/Modify menu and choose <Clear Selection>. Select some more and try <Copy selected items> You will get another keyset.

You can then afford to remove them from the first keyset using <Remove selected items>.I have never used < reverse selection>, but if you wanted to highlight most but not all of large set, it would pay to select the smaller group then use <Reverse> to get what you want.

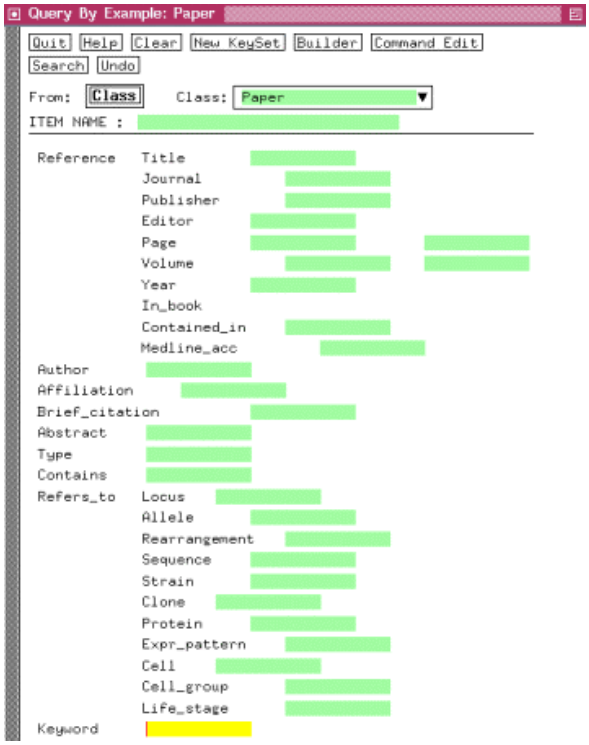
Most of these functions are also on the background menu found by using your right mouse.

EXERCISES WITH QUERY LANGUAGE AND TABLE-MAKER

Note : when using the full query facility or table_maker you are advised to have open a copy of the model of the class you are examining. With query by example, this is not necessary.

This series of examples will introduce the main features of the query language, the main window types associated with queries and most of the keywords used in the query language. The examples increase in complexity.

QUERY BY EXAMPLE



1. Get a listing of all the authors which have addresses attached.

Bring up the Query By Examples facility by choosing the option of that name on the Classes window pop-down menu. A class comes as a default. To choose author (or another class instead). There is a green box after word Class which has a right-hand down-arrow. Click on this arrow to get a menu of classes and select AUTHOR by clicking on it with left-mouse.

Now the Author tags and green data entry fields will appear. Highlight Address by a left-click to indicate that only Authors which have addresses attached should be listed. Then pick the Search(locally) button near the top of the window and a new keyset window will appear. This contains a list of the Authors which have addresses attached.

2. Get a listing of all papers published in 1991.

Bring up the Query By Examples facility (see question 1) and pick Paper class like you picked Author before. The Paper tags and data entry fields will appear. Highlight Year to indicate that you want only those papers which have the year of publication attached. Then in the green data entry field next to Year type 1990 to indicate that the Year entry must be 1990. Pick the Search button and the keyset will appear.

3. Get papers published in 1997 which have a number 9 in their object name.

4. What difference does it make if you ask for a new Keyset?

5. Find the authors who publish papers.

6. Find the authors whose email address contains *edu *.

7. Find all the cells which have daughters.

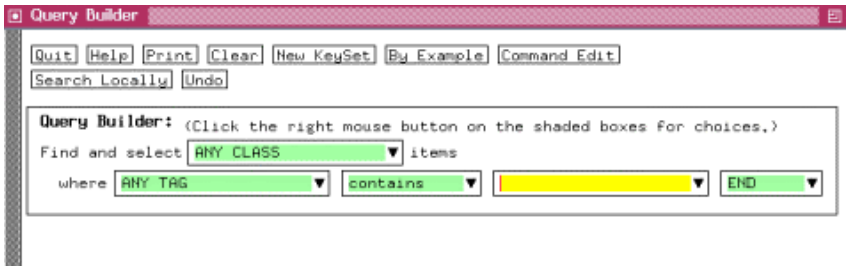
8. Choose cells beginning with C* on the main window the go back to the QBE window and change **From : Class** to **From : Keyset.** Then get Cells with daughters. Is it the same as the previous answer.

If you simply choose a Class on the QBE window, you will interrogate all objects in that class. If you wish to narrow the search, use the ITEM NAME box.

If you are not sure how to do any query, break it down into smaller portions and do a bit at a time.

QUERY_BUILDER

Do the simpler queries given under Query_by_example first.



1. Find papers published after 1991 whose author surname begins ba*

From drop-down menu on main acedb window, choose <Query> which brings up the query commands window which you can use as a reference point. Then also choose <Query_Builder> from the same drop-down menu. Keep both windows visible.

Working on the Query Builder window :

Choose A Class :

The <find and select> box has ANY CLASS as default. To choose a class pick on the right hand triangle with right-mouse and get class <Paper> by double-clicking. <Paper> appears in the <find and select> box.

Choose A Tag :

Go to the box labelled <where> and pick on the triangle to get menu. Double-click on <USE TREE TAG CHOOSER> which will bring up the model (within the <Tag_chooser> window) of Paper with all its tags and fields. Double-click on <?Author> (NOT the tag Author) so that Author gets written into the <where> window.

Choose A Condition For The Field After Tag Author :

The next box to the right of <where> box gives a drop-down menu of conditions to be applied to author. In this case choose <begins with> by double clicking. This appears in the box.

To follow up <begins with> type ba in the next rightwards box and press <return>. (NOTE no asterisk is needed, the system puts one in).

(also NOTE that you can test the query now by pressing return on the last yellow box when END is displayed OR by clicking on search locally.)

The query commands window should reflect these choices on a single line saying :

"Find Paper Author = ba*".

Second Stage To Query.

To finish the query with "published after 1991" pull down the menu from the last box and select <AND> to indicate that a second condition is being applied to this search. This brings up a second line of boxes for attributes and conditions. In the <where> box select from Paper the Year field-tag. In the condition box select <is greater than> in the text entry box type <1991><return>. The

query is finished.

To start the search you can either (1) pull down and choose END <return> in the last box or (2) go to the query commands window and put in a <return> at the end of the entry which now reads

"Find Paper Author = ba* & Year > 1991".

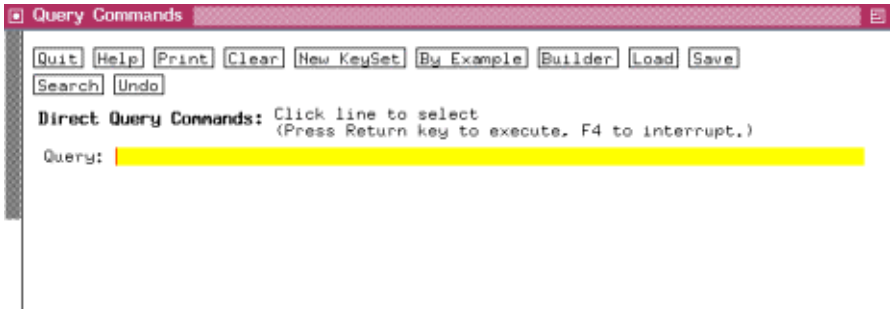
After a few seconds, a new keyset appears of the relevant papers.

2. Find all the cells with no daughters.

3. Find cells whose name contains ***v ** and that possess daughters.

PLAIN QUERY FACILITY

Note You are advised to always have the appropriate Class Model on your screen before attempting to use this plain Query facility.



These examples are being confined to as few classes as possible for simplicity but you will need to know the class model structure to be able to use most of them.

If you are unsure how to do a query, break it down into smaller parts and do the parts one at a time. Keep testing the answers for correctness. You may get the wrong answer because you have asked for the wrong thing!

When you are sure that you have got some right, try saving the queries and re-using them. This is especially helpful with tablemaker.

1a. Find all Loci starting with dp

On Class window, from drop-down menu, select Query.

On first line in yellow box after <Query:> type

Find Locus dp* //hit <enter> OR Search button.

Find Locus IS dp* // a better alternative

The result will come up in the Main Keyset window.

Note The ***** is the wild-card symbol for 'all letters' in this case.

1b. Find all Authors starting with ba*

follow above syntax.

2a. Find all Authors with email addresses

On Class window, form drop-down menu, select Query.

On first line in yellow box after <Query:> type

Find Author E_mail .....hit return OR Search button.

The result will come up in the Main Keyset window.

(A word of explanation : If you want to select objects which have data in a field you just need to give the tag name such as E_mail.)

2b. Find all Authors beginning with C, with email addresses

Find Author ("C*" AND E_mail).....hit return OR Search button.

(quote marks are not necessary)

Instead of Find, you can use the shorthand >?, i.e. >?Author.

3.Use of logical operators AND OR XOR.

Note down the number of objects found each time.

3a. Find all Authors that have either an address OR a Paper.

Find Author (Address OR Paper).

3b. Find all Authors starting with c, with address OR a Paper.

Find Author "C*" (Address OR Paper).

3c. Find all Authors starting with c, with address AND a Paper

Find Author "C*" (Address AND Paper).

3c. Find all Authors starting with c, with either an address or a Paper but NOT both.

Find Author "C*" (Address XOR Paper).

NOTE : you can substitute && for AND, || for OR.

4. USE OF NOT :

4a. Find all Authors starting with c, with NO address

Find Author “C*” (NOT Address).

NOTE :You can substitute ! for NOT.
brackets are not needed here.
try re-doing examples in 3 with nots.

5. CONDITIONS ON TAGS

5a. Find authors who live in CANADA or Japan

Find Author (Mail = *CA*).etc.

5b.Find authors who work in an American University.

Find Author E_mail = *edu*.

6. Find the Authors who have more than 5 papers published.

>?Author COUNT Paper >5

7. USE OF FOLLOW :

7a. Find papers published by Ian Hope.

>?Author IS *Hope*; FOLLOW Paper

7b. Find the co-authors of papers published by Ian Hope.

>?Author IS *Hope*; FOLLOW Paper; FOLLOW Author

(may not give much in minidb, can try another author)

Note : One can do this on subsequent lines using keysets and can use UNDO to go back up a set.

8. USE OF NEXT and HERE :

8. Find Sequences with Accession numbers.

Look at the Sequence model before starting this complex query.

DB_info Database ?Database Text Text

Where DB_info is a major tag similar to Address, Database is a sub-tag like Town and takes the name of the database, the text fields take the Identity name followed by the Accession number.

To ask questions of the text fields to the right of the Database sub-tag, it is necessary to use NEXT, meaning to the right of. Try building up the query part by part to get to the Accession number in the last field to the right. You use AND to make sure that each object selected has ALL of these things.

Find Sequence Database

Find Sequence Database AND NEXT

Find Sequence Database AND NEXT AND NEXT

Find Sequence Database AND NEXT AND NEXT AND NEXT

Find Sequence Database AND NEXT = *DDBJ* AND NEXT AND NEXT

Note the different number of retrieved objects each time.

Next can refer to a tag OR a field.

Try putting conditions at each stage of the search, and certainly

look at the result of each query before adding to complexity.

9.QUERIES ON OBJECTS CONTAINING MAP POSITIONS.

9a. Find genes on chromosome IV which have been sequenced.

>?Locus Map = IV & Sequence

9b. try III, OR, NOT sequenced , any other variations.

10. PIPING THE RESULT OF ONE QUERY TO ANOTHER.

find all the authors whose object name (surname) starts with 'a' and contains 'O'.

>?Author a*; a*o*;

OR better

>?Author a*; *o*;

you are sending the result of the first query into the second query with the ";".

11. GLOBAL SEARCH AND USE OF WHERE

11a .find all the references to dpy loci

grep dpy* <return>

11b. restrict the classes found to just loci

grep dpy* <return> ---> gives keyset of mixed object types

where CLASS Locus <return> ---> gives keyset of just genes

N.B. type both lines. Go back to grep line and press <return>, after getting a result, press <return> on the WHERE line.

12. WORKING WITH HASH STRUCTURES :

In worm databases, Locus and Rearrangement classes contain Maps where the map position is a hash.

12a. Find Rearrangements with a right ends beyond position 5 on chromosome IV

Class Rearrangement has the following tags where Map_position is a hash structure

?Rearrangement

Map ?Map XREF Rearrangement ?Map_position

?Map_position

UNIQUE Position UNIQUE Float

Ends Left UNIQUE Float

Right UNIQUE Float

>? Rearrangement (Map = IV* # Right > 5.0)

12b. Find Rearrangements with left ends before position 4 on chromosome IV

>? Rearrangement (Map = IV* # Left < 4.0)

12c. Find Rearrangements with left ends before position 4 on chromosome IV and right ends beyond position 5

>? Rearrangement (Map = IV* # Left < 4.0) &&

(Map = IV* # Right > 5.0)

12d. Find loci between positions 5 and 10 on chromosome IV

Class Locus has the following tags where Map_position is a # structure

?Locus

Map ?Map XREF Locus ?Map_position

?Map_position

UNIQUE Position UNIQUE Float

Ends Left UNIQUE Float

Right UNIQUE Float

>?Locus Map = IV* AND NEXT AND NEXT > 5.0 AND HERE < 10.0

SOME QUESTIONS WITHOUT ANSWERS

1. How many predicted genes (subsequences) have matching CDNA.?

2. How many predicted genes are equivalent to Mendelian loci.?

3. How may Clones have been finished by the Sanger centre sequencing team ?.

4. How many contigs make up chromosome IV ?

5. How many sequences have DNA ?

6. How many sequences have DNA longer than 300 base pairs.

7. How many cells have great grandchildren ?

8. Since Gene_classes are made specifically to catalogue Phenotypic descriptions, how is it that most objects in the mini database have no description?

When you are using the Query facility, there is a help box on the window. Try clicking on query_syntax to check on things like =, OR, AND. There are further examples for you to try within the HELP.

TABLEMAKER

1. Make a table with only those authors having an e_mail address in the first column and the author's Laboratories and e_mail addresses in the second and third columns.

In the first group of lines, or "Definition zone", which defines the first column, pick Author on the pop-down menu accessed by depressing the right mouse button on the box marked <Class..> This indicates that you want authors listed in the first column.

Then make a second Definition zone by picking the Create Definition button. This defines the second column. Pick the <Tag:> button with the left mouse button.

The tags and data fields for Author will be displayed. Pick <?Laboratory> to the right of the Laboratory tag twice with the left mouse button to indicate that you want to see the Laboratories associated with the authors. (If the Laboratory tag had been picked instead, you would have indicated that you just wanted the table to indicate whether there was a Laboratory or not, not to display the actual laboratory.).

Create another definition zone. Pick the <Tag:> button with the left mouse button. Pick the <Text> data field to the right of E_mail twice with the left mouse button to indicate that you want the E_mail address shown in the third column. Pick the Optional button once thus changing it to Mandatory. This indicates that only those authors which have E_mail addresses should be listed.

Pick the <Search Whole Class> button and the new table should appear.

(The layout/width of columns may not be ideal but this can be adjusted by going back to Define Table and playing with widths. You should also try altering <optional> to <mandatory>).

2a. Make a table, putting all genes with Loci beginning with "d" in the first column, and their Allele in the 2nd column

(if continuing from end of question 1, pick Define table and write over previous definitions, OR USE THE BUTTON FOR CLEARING COLUMNS)

In the first definition zone, pick Gene/Locus on the pop-down menu accessed by picking the section marked <Class:> with the right mouse button, to indicate that you want genes listed in the first column. Create a second definition zone, pick the <Tag:> button, which will bring up a window with all the Gene tags and data types. Pick ?Allele next to the <Allele> tag to indicate that you want the Alleles listed in the second column.

2b. Narrow the choice to alleles starting with e* by putting a condition under the Tag Reference_allele. type e* in the green data entry field next to Condition to indicate that the Alleles must begin with e. Also pick the Optional button once, changing it to Mandatory, so that only those genes which contain alleles will be listed.

2c. For the 3rd column : Pick ?Clone next to the <Positive_Clone> tag to indicate that you want the Clones listed in the third column. Then going back to the definition zone, type c* in the green data entry field next to Condition to indicate that the clones must begin with c. Also pick the Optional button once, changing it to Mandatory, so that only those genes which contain clones will be listed. Pick the Search Whole Class button and a new window containing the table should appear.

2d. try narrowing the whole search by restricting the Loci

starting with d.

3. Try making a table of all the “Mendelian”* genes on a particular chromosome and get them in order of their map position from top to bottom.

*those identified by mutation hence they have alleles.

column 1 : Select Class Locus (visible)

column 2 : Select Tag (blue) !Map (hidden and mandatory)

condition *IV

column 3 : Select Tag by Clicking on (green) ?Map position to get the underlying details then

Select Field Float after (dark-blue) Position

change From 1 to Right_of 2 (visible, mandatory)

column 4 : Select field ?Allele (or Tag Reference_alleles and look at the difference) (visible, mandatory)

Give the columns some Headers

Sort on column 3 and process.

save query and the table.

Query_syntax

A detailed description is presented in the manual doc /query_language_guide.txt

There are 3 possibilities

Find Class xxxxxx

The command xxx is applied to all the keys of the (sub)Class

Abbreviation: >?Class xxxxxx

Follow Tag xxxx

The command xxx is applied to all the keys following

tag in the objects of the active key set.

Abbreviation: > Tag xxxxxx

XXXXX

The command xxx is applied directly to the active key set.

A command is a logical expression evaluating to True or False.

it is applied to a key which is discarded or retained

in the resulting list.

A composite query can be formed by chaining a series of queries

separated by semicolons, in which case the active keyset obtained so far is passed into the next query.

Recognised operators are, in order of increasing precedence:

$| SETOR

$& SETOR

$^ SETXOR

$- SETMINUS

$= SETDEFINE (possibly bugged)

NEIGHBOURS

| OR

^ XOR

& AND

! NOT

Jumps in constructed types.

< <= > >= to compare numbers

= to compare numbers or match the left hand side to a

template, i.e. a word with wild chars *, ?.

COUNT Counts the number or entries to the right of its rhs

IS xxx Checks if the name of the active object matches xxx

CLASS xxx Checks if the active object belongs to (sub)class xxx

Parentheses of all sorts "([" can be used freely, but must be matched.

Words are matched to a tag or treated just as text.

They must be put in "double quotes" if they include

spaces or any operator &, |, ^, <, >, =, (,), [,], {, }.

Wild chars can be used in words but then they are not

at present matched to tags except in a redirection

Numbers are parsed as floats.

Examples (as given in $ACEDB/wquery/examples.qry)

>?Chrom* will list all chromosomes

>?Author s* | a* all authors whose name begins with s or a

>?Au* s?s* OR b*s* all authors whose name matches s?s* or b*s*

>Pap* Journal = N* OR Year > 1987

Checks the previous list : redirect to papers and lists

all their papers published in N(ature) or after 1987

Year = 1988 Restricts to the papers of exactly 1987

>?Gen* myo* Clone All the cloned myosin genes

Find Author IS "Sulston JE" ; >Paper ; >Author

Finds all the co-authors of papers by Sulston.

Sub fields

To find the value of sub fields you must first

locate yourself on the tag then move right from it

using NEXT (move right) or HERE (move here) for multiple checks.

For example the genes to the end of

chromosome X are found with

Find Gene gMap = X AND NEXT > 12 AND HERE < 19 ;

Warning

This is clearly hard to read, also it will be evaluated

at execution left to right as it should only if the precedence of the operators allows it. Be careful.

Subtypes

To locate a tag in a subtype, you must indicate the path using the operator #, example:

Find Gene Lab AND NEXT#freezer = 4

To find all the genes in your freezer number 4

COUNT

To count entries, example, find prolific authors:

Find Author COUNT Paper > 100

Data Input / Output

This section covers the addition of data, amending data and deleting data; dumping out normal textual data also the input and output of DNA sequences which can be dumped out in special formats. The use of the ACEDIFF executable for finding the difference between two sets of .ace formatted objects in ASCI files is discussed.

There are two different ways of adding, amending or deleting data with an ACEDB database :

(1) interactively using the **ADD / ALIAS / RENAME** option :

usually on one object at a time, or on groups of objects in the Keyset window

(2) indirectly by reading in an ASCI file in **.ace** format :

used for several objects and the preferred method for bulk data.

Data can be dumped out in several different formats, the most common being in ASCI text for B-type objects in a **.ace** file. Peptide and DNA sequences can be dumped out in .ace format or in FastA format. There is an **export** or **dump** facility on most object windows and on the Keyset window.

GENERAL PRINCIPLES OF DATA INPUT / OUTPUT

ADDING NEW OBJECTS TO ACEDB

1. Display the Class model of the object to be added, for consultation.
2. Select or name the Class to be used.
3. Give the object a name or unique identifier.
4. Fill in as many data fields as required.

In the .ace format ASCI file, the record for a single object will follow this pattern :

```
Class_name1 Object_name3
Field_1 “aaa bbb ccc” //Field_1 is the Tag name for the field
Field_2 “ xxxxxx gg hhhh jjjjj” -C “this is nonsense”
                                                    // after -C comes a comment
```

There must be blank lines between objects in text files, so after 1 or more blank lines then the next object can start.

Anything written after // is ignored by the acedb parser.

```
Class_name2 Object_name7
Field_a “fhjfjhhj”
Field_b “ads dgfk askkjkj klkl!”
```

DELETING OBJECTS FROM ACEDB

The acedb command syntax is :

```
-D Class_name Object_name
```

One way of deleting objects is via an .ace file. This will need to contain a list of the Class and Object names, preceded in each case by the command -D and separated by blank lines :

```
-D Locus abc-3

-D Allele  s45
```

To delete objects interactively :

Get a Keyset of objects you wish to delete, select them all, or a subset, by highlighting them on this window and pick Other from the top of the window.

then pick **Edit/kill** and respond to the menu of choices.

Amending Objects In Acedb

Changing an object’s name can be done using the acedb command :

```
-R Class_name Object_name1 Object_name2
```

Changing a data field inside an object can be done as follows :

Class_name Object_name
-D Field_2 // where Field_2 is the Tag name
Field_2 “this is the new data item”

Adding Comments To Objects

Open an object in the usual way and select the update option on the background menu (right-hand mouse button), select **add comment**, type into the field given and press return.

Deleting Comments From Objects

As for Adding comments but select the **delete** option after highlighting the comment.

Detailed Instructions For Addition, Deletion, Amendment Of Data

I. READING IN A PREPARED .ACE FILE

II. MAKING YOUR OWN .ACE FILE

III. EDITING INTERACTIVELY

I. Reading In A Prepared .ace File

To read a text file of .ace formatted objects into acedb, open your database, then get **Write access** by pulling down the menu under **Edit,** then open this menu again and select **Read .ace files**, A new window appears **(ace file parser**), set by default to look for a file in the home directory, if that directory exists. If not, the pathway at the top may be given as “.” for this directory or the directory you were in when acedb was opened. In either case you can type in the complete pathway to the directory containing your file, or navigate around your computer files using the a menu on this window. If you press the TAB key, Only files ending .ace will be displayed for you to choose an input file.

To choose a file, either click on the filename or type the file into the File box (yellow area). Then you can either Press <Return> on the filename box or the **OK** box. Another new window will appear which is fairly self-explanatory. Clicking on **Read all** will cause the whole file to be read in. If you get an error message, edit the .ace file in a text editor, save the changes, close and re-open the file and start again with reading in. Or just click to continue and ignore the errors. (in this exercise, they will be caused by trying to delete, rename or edit objects which are not already in the database.

When finished, you can quit the **READ (or parser)** window and have a look at some of the new objects.

II. Editing Acedb By Reading In Your Own .ace Files

Open a new file using a text editor such as vi, emacs, jot, word etc.. You can give the file any name you please BUT you must remember to add .ace to the file name. One file can contain any number of objects of any type. For example Author, Paper, Sequence, Gene objects can all appear in the same file. The file can be of any length.

An example of two simple objects is given. The first line after a blank line must contain the Class name followed by the object's unique name. After that, each consecutive line must contain a Tag followed by data. Tags are shown in bold for emphasis. When the data field is of the text type (as opposed to integer, floating point number, date), the data must be enclosed within double quote marks. Although, only the first or opening quote is essential, it would be very bad practice to leave out the second one. (Why?)

Author "Patel B"

Full_name "Bala Patel"

Laboratory CB

Paper [cgc1011]

Paper [cgc533]

Mail "Laboratory of Molecular Biology"

Mail "Hills Road, Cambridge"

Fax "050 3456789"

Paper [cgc533]

Title "Yet more of those Genes"

Journal "Cell Reports"

Volume 3

Year 1993

After saving the file, you can read it into your acedb database.

Get **Write access** in ACEDB, by picking the option of the same name on the Classes window pop-down menu, accessed by picking anywhere on the Classes window with the right mouse button. Then choose the **read .ace** files option on the same menu. A new window will appear. Pick the **Open File** button in this window. Another window will appear. Here you should indicate the file you want to read in. Press <Return>. Then pick the **Read all** button in the previous window to read in the file. If you are happy with the new data you should immediately choose the **Save** option on the Classes window. this button is only visible if you have write access.

The next file contains some suggestions for amendments using the ace commands to be written into a text file fix.ace. (You can make this file yourself).

```
// EDIT COMMANDS in file fix.ace
//======================

Chromosome A
Non_graphic                              //set a Boolean value

Paper [wbg10.1p142]
-D Author “Arhinger J”    //delete data from object

-D Author “Micky Mouse”          // delete whole object

-R Sequence  zk643   ZK644     //re-name object

-A Sequence  ZK637.9  lin-9      // making an alias
```

II. Editing Acedb Interactively

With an open database, obtain **Write access** using the right mouse button and the drop-down menu on the class window. This write access option is NOT available unless you are an accredited curator, i.e. your login name is in the passwrd.wrm file in the wspec directory.

If you wanted to put in the following Author information manually, you could start by displaying the Author class model in order to see which Tags and fields might fit your information.

Alan S. Jones is at the CB Lab, he has written two papers, "Big Genes" which appeared in Cell, Volume 3, 1989, and "Small Genes" in vol 7 of Nature 1989. His address is Genome Studies, Medical Research Council, Hills Road, Cambridge, and his fax number is 050 3456789.

1. Display the Author class model

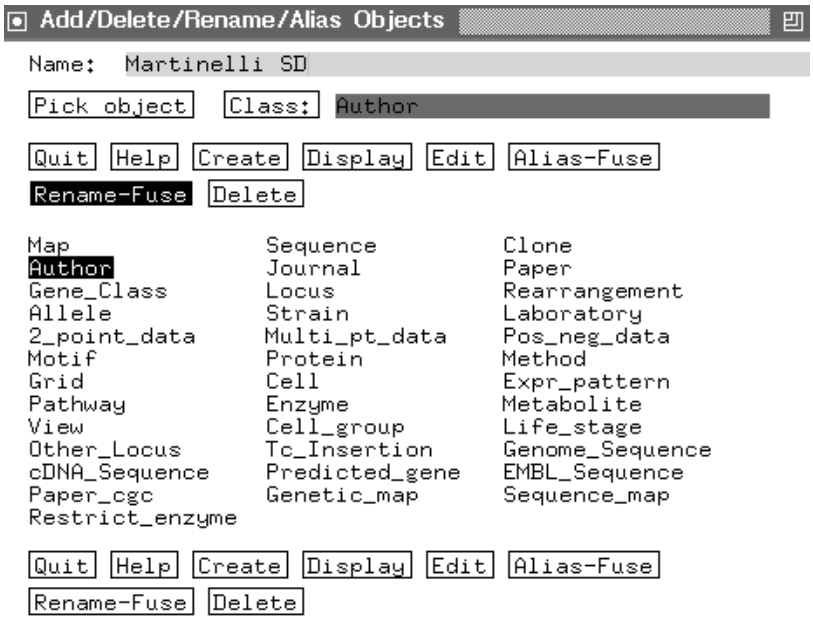
Highlight the Model class in the Classes window, type ?Author in

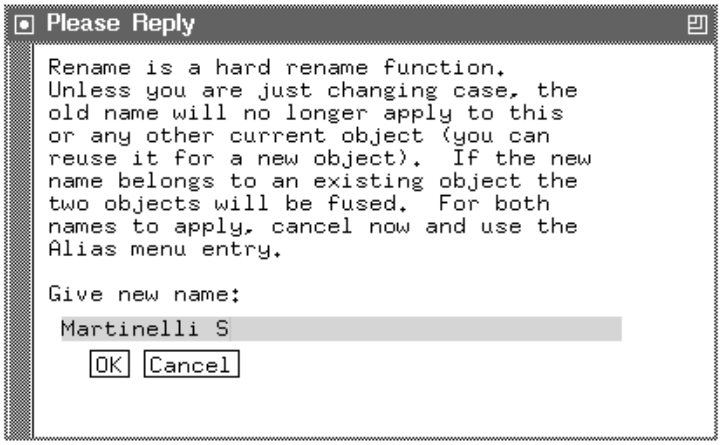
the Search field (yellow box), and press <Return>. This shows you the model on which the update form is based.

2. Make a new Author entry

In the C.elegans database, we reverse peoples names to be surname followed by their initials, hence we would create an object called Jones AS".

To do this, you need to choose the **Add / Alias / Rename** option on the **Edit** menu which is accessed by picking on the Edit box on the Class window with the right mouse button. A new window appears called **Add/Delete/Rename/Alias Objects** which contains a list of class names will appear. If the class you want is not listed, click on the **Class:** box towards the top of this window to get a full list.





Click once on Author and type "Jones AS"<return> (quote marks not needed) in the yellow data entry field after **Name:.** A new window will appear asking if you want to create a new object; answer **yes** if you have typed the name correctly. A new window will appear with the name "Jones AS" at the top. There is a background drop-down menu for addition/alteration of data, on all B-tree windows.

On this “Jones AS” window, the model is shown in blue, both the tags and data fields. (Getting the object window by this route, you only have to click once on a data field for it to turn yellow).

Pick a data field with the left mouse button, bringing up the yellow data entry field thus permitting you to type in something. Press return to finish that data field, then click on another and so on.. Type the data relating to "Jones AS" in the data entry fields next to the appropriate tags. Each time you have finished correctly typing a field, press return and move on to another one.

If you make a mistake in a particular field, highlight the offending data then choose **Delete** on the drop down menu. Choose **Save** from the drop-down menu before exiting the object. If you forget, and select the **quit** box, or quit on the drop-down menu, you will be reminded to save the data. You can elect NOT to save if you are merely trying something out or have made a mistake.

(I have sometimes opened a data field by mistake only to find that to exit from this field I have to enter something, any rubbish will do, press return then go back and delete the data.)

The Author data given above might be added to the "Jones AS" object as follows:

Author Jones AS

Full_name Alan S. Jones

Laboratory CB

Mail Genome Studies

Mail Medical Research Council // (Get a new text field for each line)

Mail Hills Road, Cambridge

Fax 050 3456789

Paper [cgc101] //(one paper entry per line of text)

Paper [cgc532] //square brackets for papers are a convention
//adopted by the worm database

When you get to the **Paper** tag, type [cgc101] on one line <return> then reply **yes** to the prompt about a new object. Put [cgc532] on another line. The details about the papers should not be added here since they relate to the paper objects rather than the Author object. Because of the cross-referencing between paper and Author objects in the models, the papers [cgc101] and [cgc532] will have been created if they did not already exist.

3. To add data to an existing object

Two ways exist of getting to an object window for the purpose of editing the data. To add data to the recently created Papers [cgc101] and [cgc532], it is necessary to use the **Add/Delete/Rename/Alias Objects** window. If you try to access them as part of a list of objects on the Keyset window, they will appear in feint print (and are popularly known as ghosts). They are empty objects and cannot be opened this way.

Using the **Add/Delete/Rename/Alias Objects,** select the paper class then type in the object name [cgc101] or whatever in the yellow box next to **Name:** pick the **Display** or **Edit** boxes option to get the Paper [cgc101] window. Picking **Display** gives an empty window apart from the Object name. To visualise the Tags and fields, on the pop-down menu accessed by picking anywhere on the background of this window, select **Update**. All the tags and fields appear shadowed in blue. Picking **Edit,** on the other hand, automatically brings the object window up in update mode with blue shadowed tags and fields. Add data as for the Author “Jones AS” in the previous section. The save and quit the object, as before.

Paper [cgc101]

Title "Big Genes"

Journal Cell

Volume 3

Year 1990

Paper [cgc532]

Title "Small genes"

Journal Nature

Volume 7

Year 1989

4. To add/amend data in old established objects using the interactive method

Bring up an object’s text window in the usual way, (select Class and object on main window, or main plus Keyset windows). Then select update. If you want to delete a field, highlight this field then choose the **Delete** option on the pop-down background menu If you want to alter data, either highlight the field then select **Edit** or double click to get yellow background and type new data. The old data is lost. To add a comment, highlight the field

and select **Add Comment** which gives a yellow box for typing to the right of the data. To delete or replace a comment, highlight the comment and select the **Delete** option. If wanted, select add comment and carry on as usual. Don’t forget to **Save** your changes.

(cancel, contract and expand on these menus not mentioned)

5 To add/amend data in old established objects using the .ace file method

Open a text editor and add lines similar to those shown in boxes

This next line re-names the "Jones AS" object as "Jones BS"

```
-R Author "Jones AS" "Jones BS"
```

This delete Jones BS's E_mail address and replaces it.

```
Author "Jones BS"
-D E_mail
E_mail "jones@mrc-lmb.cam.ac.uk"
```

This adds an abstract to a paper object

```
Paper [cgc15]
Abstract [cgc15]

LongText [cgc15]
This is the full text of a fascinating article by Leon Avery on nuclear protein extracts.

It can contain blank lines, because the following special symbol marks the end of the entry.
***LongTextEnd***
```

This adds a comment to a Paper's previous Author tag entry.

The comment will be shown with a dark background in ACEDB. You can add a comment next to any kind of entry.

```
Paper [cgc12]
Author "Smith JK"
Author "Jones AC" -C "And the rest of the gang"
```

This deletes a comment and adds a new one.

```
Paper [cgc12]
Author "Smith JK"
- D Author "Jones AC" -C "And the rest of the gang"
Author "Jones AC" -C “et al.”
```

To edit a group of objects interactively

Get a list of objects that you want to edit in the Keyset window, for example all the papers from the Worm Breeder's Gazette, known in the database as Paper [wbg*] where * stands for any character. Click on the **Other** box, so that the **Edit/Kill** box is revealed. Use the **Select/Modify** box menu to select all these papers, then from the Edit/kill box drop-down menu, select **Edit only these selected objects.** This will give a new box for typing in a .ace command, such as

```
-D Title //which is usually Worm Breeder's Gazette
```

Reselect your set of Papers, get the Edit box again and type something like this, not forgetting the quote marks

```
Title "The new Worm Breeder's Gazette"
```

Adding DNA sequences

These are rather special in that they can be added in .ace format or in Fasta format. On the Fmap window, there is a drop-down menu which includes **Import sequence.** The window called allows you to import plain or fasta sequences from the directory of your choice..

V. DUMPING DATA FROM ACEDB

Normal Textual Information

To dump (or export) a single object like "Jones AS", it is just necessary to select the window for Author "Jones AS" on the main window then on the "Jones AS" window, pull down the background drop-down menu and select **Export data.** The **Dump** window will appear, asking you where you want to dump the object**.** Work your way around the directory tree if the default suggestion does not suit you. Give a name to the file. (I use the object's name or the class name and a date). Click on OK or press return on the yellow bar next to **File:** which should contain your filename. The Dump window should disappear when the object has been successfully dumped.

A set of objects on the keyset window can be dumped directly from that using the **Export** buttons drop-down menu at the top. Either a list of object names can be dumped out into a file or the set of objects in their **.ace** format. You have to respond to questions about the place to dump the file and you have to give the file a name.

Saving a list of object names can be useful if you want to add the same amendment to them all using a script, or feed the list back into acedb as a keyset at a later date.

To Dump A DNA Or Protein Sequence

From the keyset window, with a list of sequences (or proteins), do a query to FOLLOW DNA (or FOLLOW Peptide), then on the resulting Keyset of DNA (Peptide) objects, from the background menu under **Export data** select **DNA in FASTA format (protein in FASTA format)**. It is also possible to dump the sequence in .ace format by selecting for **ace file** on the dump menu.

DNA sequences can also be dumped from the **Fmap** or sequence display. The drop-down menu menu on exons gives the possibility of either dumping the cDNA sequence in FASTA format or the peptide sequence. The genomic DNA can be dumped from the drop-down background menu opened on the whole Fmap window.

Proteins be dumped from Pepmap by clicking on **Analysis..** and selecting **Fasta dump of Active zone**. The sequence of the peptide named at the top of the window is written to standard out (your active screen). The coordinates of the **Active Zone** could be changed to dump out a smaller segment.

Translations of predicted genes can be dumped out from the Fmap. There is a drop-down menu on each exon which includes **export translation.**

VI. USING the ACEDIFF PROGRAM

The executable binary file of **acediff** comes with the bin/ directory of your standard ACEDB distribution.

acediff is a program that takes two ace files (here called old and new), finds the differences between the two files then writes out the differences to standard output (screen). An output file to receive this data can be specified, such as difference.ace. Using this file would effectively convert a database containing the old file into a database containing the other new file. The command line SYNTAX assuming that **acediff** is on your path:

```
% acediff old.db new.db > difference.db

                        //Equivalent to : old.db  +  diff.db = new.db

% acediff old.file new.file > difference.file
```

Running ACEDIFF with some small example files

```
//short.1.ace       a short practice ace file


Author : "Angstadt JD"
Laboratory "BC"
Paper    "[wm83p5]"
Paper    "[wm85p104]"
Paper    "[wbg8.2p31]"
Paper    "[medline10]"


Author "O'Callaghan M"
```

```
Remark "ACEDB curator"
```

```
//short.2.ace         another short practice  .ace file

Author : "Angstadt JD"
Laboratory          "CB"
Mail        "4 Bridge Street"
Mail        "Cambridge"
E_mail    "jang@mrc-lmb.cam.ac.uk"
Paper     "[wm83p5]"
Paper     "[wm85p104]"
Paper     "[wbg8.2p31]"
Paper     "[medline10]"

Author "Martinelli SD"
E_mail "sylvia@sanger.ac.uk"
Remark "ACEDB curator"
```

Look at these two files. The first Author has changed laboratories and acquired an email address. Two different authors occur in second place. To get a file containing the differences just between these two files you can type on the command line.

```
%         acediff short.1.ace short.2.ace > short.diff.ace
```

BEWARE if **acediff** is NOT given an output filename it puts the result into the second file given to it. Here this is equivalent to your newdatabase which you have then LOST.

The file short.diff.ace should only contain the differences between the two files. Notice how ACEDB conveniently puts the file names at the top of the file for you of the two files used by acediff.

```
//short.diff.ace
// acediff difference from short.1.ace to short.2.ace
```

```
Author "Angstadt JD"
-D Laboratory "BC"
Laboratory "CB"
Mail "4 Bridge Street"
Mail "Cambridge"
E_mail "jang@mrc-lmb.cam.ac.uk"

Author "Martinelli SD"
E_mail "sylvia@sanger.ac.uk"
Remark "ACEDB curator"

Author "O'Callaghan M"
-D Remark "ACEDB curator"
```

If the first file comes from an old database and the second file from a newer version of the database, reading the difference file into the old database will bring this up to date.

Examples of the use of acediff in the C. elegans project

I used to dump out the Sequence Class from the Cambridge sequence database each month, then compare it with the previous month's dump. The resulting difference file was used to make an Update file, containing all the new data added to ACeDB since the previous public release. This update file was then placed on the public ftp site.

(Details of making updates are given in the next section).

Use of acediff to prepare a file containing just the data alterations meant that a much smaller amount of data had to be sent around the internet. A similar technique was used for the physical map database.

I have used acediff to look at two different versions of what should have been the same database. I used the dump mechanism for the whole database to prepare two large dump files then compared them and got a summary of the deviations in the databases. I believe this is still an underused but powerful resource. It could even be used to synchronise databases kept at different sites.

VII. Reading EMBL Sequence Information Into Acedb.

This is one of many examples whereupon a curator wishes to transfer data from another database into their version of ACEDB. Databases store their data in fixed formats so that a program written to convert one public format into another database format should have lasting value. These kind of programs are relatively easy to write. The programming language of choice for such tasks is **PERL.**

Although this exercise demonstrates conversion of EMBL sequence information for incorporation into ACEDB, similar methods would be used for GenBank, or for literature references held in Medline, protein data in Swissprot etc.

BASIC METHOD using the command line on a unix machine, assuming that you already have perl program to convert EMBL to .ace format :

```
> perl convert2ace < input.file > output.ace
```

Here is a real example where a curator wishes to add the record for the C.elegans calmodulin-like gene to a database

and has obtained the EMBL record cecal1.embl. The perl program is called embl2ace, executable program for conversion of embl format into .ace worm database format.

If the perl interpreter is on your pathway, you can type the following on the command line interface :

```
% embl2ace < cecal1.embl > cecal1.ace
```

If you don't give an output file, the script will write the .ace file to standard output (the screen).

Compare the input and output files, then read the .ace files into your database and look at them.

We (Sanger C.elegans DB) only extract certain bits of information from EMBL records, but the whole record is read in as a piece of LongText which can be pulled up from within the Sequence object. We could make more use of EMBL information such as cross links to the medline references but we don't routinely at the moment. Also we could make more use of references to Enzymes.

VIII.Parsing And Saving Text

Whether you are editing interactively, or adding ace files with xace or tace versions of the code, the parser compares your data structure and data content with the model for consistency. It also checks your models.wrm file for consistency when your re-read models.

WARNING : parsing is not an 'all or nothing' process. Unless it reaches EOF, the XACE parser will continue to process objects even if previous objects contain mistakes. You can then save the partially altered data, or exit without saving in order to start again. The TACE parser in 'Parse' mode will stop at the first mistake and only alter the previous objects. IN 'Pparse' mode, it will read the whole file but report the mistakes like it does in XACE. Quitting from TACE saves any data changes automatically.

In xace, the parser is rather ‘kind’ and reports what is wrong and inside which object and that it is near line so_and_so. You can ignore the mistakes and keep adding in objects from a file. Everything correct will go into the database and you can choose to save immediately or be given the choice of saving (or not) when you exit from the database. I tend to keep open data files in a text editor while entering data so that I can correct mistakes on the fly, as the parser finds them. If you have a lot of mistakes and don’t want any of the changes to take place, remember to exit without saving.

Common mistakes

When filling in a text field, forgetting to put quote marks around 2 or more words. The parser usually reports :

ERROR : PARSE ERROR NEAR LINE XX IN OBJECT : TAG SO_AND_SO : ‘the second word’ - OFF THE END OF THE MODEL.

Putting a float into an int field will result in truncation.

Putting an int into a float; the parser will add ‘.0’.

If you have typed a class name or tag name incorrectly, the parser

Will say “unknown tag/class”.

If you have left out a tag but included some data starting at the margin, the parser will take the first word and report that this Tag is unknown.

ERROR : PARSE ERROR NEAR LINE XX IN OBJECT : UNKNOWN TAG ‘first word on the line’.

A blank line inside an object is assumed to end an object and the next line of text is assumed to start another one. Since the next line will start with a tag the message given is

ERROR : PARSE ERROR NEAR LINE XX IN NO OBJECT : UNRECOGNISED CLASS TAG_WHATEVER.

The same effect will be given if there is a line starting with // (indicating a comment) in the middle of an object.

From edgrif@sanger.ac.uk Mon Jan 15 10:47:27 2001

Saving data in acedb

You've learnt how to make updates to an acedb database in the preceding pages but it's also important to know how to save, or how to throw away, changes that you've made. It helps to understand a little about acedb works.

When you make changes to objects in the database acedb makes copies of those objects by reading them from the disk where the database is stored into the memory of your computer. It's on these copies in the computers memory that you make your changes. This is crucial to understand because its not until you explicitly "save" your work that the changes will be copied back to the database stored on the disk.

In xace you select the "Gain write access" item from the "Edit.." Menu button on the main panel, once you have done this you will see that a "Save.." menu button has appeared on the main panel. You can use the "Save.." menu button to save your changes to disk. This means that you can make changes to objects, but then if you decide that you don't like the changes you've made, you can simply exit xace without saving and the database will be as it was before you started. This is particularly useful for when you wish to load large amounts of data from an ace file and you either want the whole of the data to be loaded correctly or none of it to be loaded. You can select the "Read .ace file" menu item, read in the ace file and if there are any errors you can then quit xace, fix the errors in the ace file and try again.

The Many (Inter) Faces Of Acedb

TACE text ace : command line acedb.

Some of the more frequently used commands on which this exercise is based.

Classes	**Find class name**
List	**Show**
Parse	**Edit**
count	**clear**

To start the textual database front end :

% setenv ACEDB ~/teachace (or whatever is relevant i.e. home acedb directory)

% tace (or full pathway to the binary tace)

You will see a short introduction, followed by

Type ? for a list of options

then

acedb> (the prompt on the line for entering commands)

type classes then choose a class with a few objects for the next part.

find, list, show and clear

having chosen a class with a few objects

each of which has a few used tags to make it interesting

type :

find Locus (for example. find gets the Class indicated)

list -a (-a = ace format, equivalent to name dump)

(-p perl)

(-h looks similar to ace)

(-j has lots of ?)

type :

find author (for example)

show -a (ace format, equivalent to ace dump)

then type :

show -a -f authors (makes written ace dump called authors in the current directory)
(equivalent to Write authors)

to narrow down a search within a class, type :

find Author *Wild A* (for example) [wild a] [“wild a”] will also work
show -a (then shows just this author)

another example :
find Genome_sequence
show -a Subsequence (this restricts the data displayed to just that Tag and its field)

to return to having no selected objects type :

clear (quite useful, although every time you select
it overwrites previous keyset anyway)
show/list -a (to check empty list)

Parse (=write)

put in a short acefile e.g. short.2.ace from the diff directory
type :
Parse short.2.ace (N.B. you may need to put in the full pathway)

Only objects which have been parsed correctly will be included in the list/show

edit
Find a locus such as vab-7 and edit it
find Locus vab-7
list -a (to check the selection)
show -a(this is so you can see the object before and after editing)
Edit Allele sdm1 (adds this allele to vab-7)
show -a
edit Enzyme "lipase" (adds the enzyme to vab-7)
show -a
edit -D Enzyme (or something else already there in object)

show -a
edit Reference [wbg15.7p99]
show -a
clear

query

(N.B. some of the keywords used in queries are case-sensitive
so you are advised to use CAPITALS)

syntax : QUERY FIND class_name [condition1] [tag_name[condition2]] etc.
[condition 1 would restrict the objects selected using their object identifier
e.g. author containing *a*c*]
[condition2 is restricting the selection according to the value in a field]

the following are general examples for my personal database
you may need to modify them to suit yours.

QUERY FIND Sequence *c*
QUERY FIND Sequence *c* AND COUNT DNA_homol > 50
QUERY FIND Sequence *c* AND DNA_homol & Feature
QUERY FIND Sequence *c* AND (DNA_homol OR Feature)

QUERY FIND protein *wp* AND DNA_homol

QUERY FIND protein *wp* AND Title
show -a Title

QUERY FIND protein *wp* AND COUNT DNA_homol > 50

QUERY FIND Author NOT Sequence AND NOT Paper [try using brackets]

spush, spop, sminus

QUERY FIND Locus Map = IV # Position > 1.0
spush
QUERY FIND Locus Map = IV # Position > 3.0 [and I mean >]

sminus

spop

list -a

this should show the loci between 1 and 3 on chromosome IV

count

can use this anytime to find out how many objects are in the current keyset.

dump

dump -s (takes up a lot of space and time from a big database and can run into problems)

(I advise dumping out individual classes using a script)

quit to leave the database

List Of Commands Available In Tace Mode.

? : List of commands

Help : for further help

Classes : Give a list of all the visible class names and how many objects they contain.

Model model: Shows the model of a class or subtype, useful before Follow or Query commands

Find class [name]: Creates a new list with all objects from class, optionally matching name

Follow Tag : i.e. Tag Author gives the authors of the papers of the current list

Grep template: searches for *template* everywhere in the database except LongTexts

LongGrep: searches for *template* also in LongTexts

List [-h | -a | -p | -j] [-c max] [-f filename] [template] : lists for [human | ace | perl | java] [in file] current names [matching template]

Show [-h | -a | -p | -j] [-c max] [-f filename] [tag] : shows for [human | ace | perl | java] the [tag-branch of] objects of current list

Is template : keeps in list only those objects whose name match the template

Remove template : removes from list those objects whose name match the template

Query query_string : performs a complex query - 'help query_syntax' for further info

Where query_string : same as Query, kept for backwards compatibility, do NOT use

Table-Maker [-active] [-j | -p] [-c max] [-title] [-n name | [-f] command-file] [params] : [Search active list] [java|perl style] table defn [%d->param]

Biblio [-abstracts] : shows the associated bibliography [including the abstracts]

Dna [file] : Fasta dump of related sequences

KeySet-Read [filename]: read keyset from file filename or from stdin

spush : push active keyset onto stack

spop : remove top keyset from stack and make it the active keyset

swap : swap keyset on top of stack and active keyset
sand : replace top of stack by intersection of top of stack and active keyset
sor : replace top of stack by union of top of stack and active keyset
sxor : replace top of stack by exclusive or of top of stack and active keyset
sminus : remoevs from top of stack members of and active keyset
Parse [file] : parse an ace or ace.Z or ace.gz file or stdin, until ctrl-D or EOF
PParse [file] : as above, but don't stop on first error
Write filename : acedump current list to file
Edit ace_file_command: Apply the command to every member of the active list
EEdit ace_file_command: idem No Check
Read-models
Kill : Kill and remove from acedb all objects in the current list
Status [on | off] : toggle memory statistics
Time_Stamps [on | off] : toggle time stamps creation
New Class Format: i.e New Plate ya\%dx.y creates ya8x.y if ya7x.y exists
Count : number of objects in active keyset
Clear : Clear current keyset
Save : Save current state otherwise the system saves every 600 seconds
Undo : returns the current list to its previous state
// : comment, do nothing
Depad [seqname] : depends an assembly, or currently padded assembly
Pad seqname : pads an assembly
// 0 Active Objects

ACEDB Directories And Their Contents

rawdata/ **assumed by code**

bin/ **essential**

essential files : xace, tace,

optional : acediff, aceserver, aceclient, gifaceserver, xaceserver, gnbkclient, jade2ace.

database/ **essential**

ACEDB.wrm

block1.wrm etc. //as many blocks as needed

database.map //may have to add some blocks in large DB

log.wrm

touched/

new/

wspec/ **essential**

cache.wrm // GraphPackage (can be removed)

copyright

database.wrm

displays.wrm

layout.wrm

models.wrm

options.wrm

passwd.wrm

psfonts.wrm

server.wrm

subclasses.wrm

xfonts.wrm // only needed on some unix X-terms

wscripts/ **optional**

display_script // only relevant for worm jpeg pictures in pictures/

wgf/ // contains worm specific tables needed by the genefinder package //**optional**

whelp/ // context sensitive help pages in html format**.** **// optional**

wquery// table.definitions and queries can be stored here // **optional**

Looking At Some Other Important Directories Of ACEDB

database/ contains the storage information, and handling records

wspec/ has classes and features and types of display

wgf/ has genefinder tables (mainly applicable to worm)

wquery/ stores and retrieves Queries and Table Definitions

whelp/ directory comes with the binary code.

wspec

Database manager can change many of theses files (marked *) to customise the database but it isn't necessary, especially to utilise worm data.

>cd wspec/ // to examine contents :

various files some or all of :

graphpackage, cachesize.wrm, copyright, database.wrm, displays.wrm, help.wrm models.wrm, options.wrm, passwords.wrm, server.wrm, classes.wrm, subclasses.wrm, *font*,

Essential files for personalising one's own database are :

models.wrm, options.wrm, displays.wrm and subclasses.wrm

Ignore Graphpackage : config of X properties which can be deleted (also README).

cachesize.wrm *

sets memory requirements. HAS BY DEFAULT :

cache1 = 4000 kb (minimum is 2000kb)

cache2 = 4000 kb

disk = 4000 kb

so 8000kb total cache unless you choose to set them differently. to increase performance, try increasing cache2.

to run on small MAC you could try shrinking them a bit. a bigger cache gives faster working but slower writing back.

disk = 4000 kb means the block sizes allocated, every time db goes above this another 4000kb memory allocated to it. when the db is first run, disk memory allocation is 4000 kb. Note : disk size and cache size are independent so that block size does not reflect the cache size.

constraints.wrm

This is not yet fully supported.

database.wrm

makes several files of blocks of data which must be big enough to hold all the data. Can add more blocks as the data grows and one can increase the block size. It can hold the NAME of the database for use when adding UPDATE files.

displays.wrm * :

an important file which can be changed. It gives the names of the window types. Every window type in acedb is a display which can be changed without changing the database itself. One line allocated per type in file.

Window size is configurable.

can set width and height of windows (= screen coordinates in notional units) However, all text objects use the same window so if one window is enlarged, all the others will be.

can change -t TITLE for title name to be changed e.g. to WORM (can set a help name for each window?)

DO NOT CHANGE the -g option for example : -g TEXT-FIT

because the program expects it.

help.wrm :

help works 2 ways :

(1) via help.wrm the original version;1 page for each of these wspec files

(2) via Mosaic (you can write your own help pages)

a future version will have Mosaic with html.

can add new help pages oneself.

The version in the wspec directory is really redundant now there is a whelp directory of html pages.

layout.wrm

You can type into a normal text file your exact class window layout, with the classes in columns, as you want them to appear on the window.

The class put in the top left-hand corner will be the default class whose object list appears on the keyset window when the database is first opened.

You can use classes or subclasses here.

**models.wrm *

see later, this is a large section.

options.wrm *

(for full details read the file wspec/options.wrm)

This file is read every time database is run and has 1 line per class for options which can be set at the class level. This file therefore reveals how to declare a new class.

default for most is VISIBLE -V rather than HIDDEN -H

visible classes occur in the main class menu.

One can change the order in which these are presented (HOW?) e.g. to alphabetical. Now the first one in the list is the default when acedb turned on. They all follow in the sequence seen in this file.

default = -VClassname

Class structure (DOMAIN) :

Btree (B) most classes are B-tree which is the default.

each has a model in the models.wrm file

Array (A) some are arrays = DNA, Peptide, Keysets and LONGTEXT.

These are not accessed directly by name. Only a programmer can specify new arrray type classes, so new classes must normally be of the Btree type, the default..

Other classes/OPTIONS :

XREF a system class used to set up 2-way referencing

DISPLAY (D) a default = tree in text form e.g. for authors.

Map is given as default for genes, clones, sequences.

Otherwise they are displayed first as record card (tree text display).

e.g. for gene on map need to set to gMap display :

_VLocus -V -D GMAP

-T (Tag)option = a type of alias feature e.g.

-VStrain -T genotype

When a strain is called by another object, the full genotype is the first thing displayed as the default NOT not a strain number (object identifier) which is meaningless. For Paper the title is displayed rather than the cgcnumber this default means that if there is a title it is used, if not then the database number.

Vmap is an old map. It subsumes into gMap.

-R can be used to rename a class but BEWARE of confusion here.

could change class Map to class Chromosome.

STRONGLY advise against renaming classes.

_VClone -R Foo

would re-name Clone to Foo in the main window.

this all dates back to when acedb was less configurable.

-Case sensitive object names are not case-sensitive by default, however, this can be changed but once changed it cannot be reverted.

This would be necessary for a database containing Drosophila genes for instance.

to make a new class, add a line to this options.wrm file

passwds.wrm*

list of usernames for people with write access

psfonts And xfonts

specific post script fonts and standard fonts for X windows. can change either but should not have to. COLOUR under ps fonts controls the default for printing. psfonts is included in case other printers don't have the defaults assumed by acedb. (can change with the printer controller which comes up in acedb when printing.)

xfonts doesn't scale fonts but ps does. if the fonts work, best not to meddle with them otherwise can use command :

acedb -acefont 12 x 16 (or whatever you choose).

standard acefont is 8 x 13.

server.wrm

a multi-user version of acedb for multiple editors. usually used with scripts which read and write to the database automatically. a client-server version of acedb has 1 server database and lots of clients, server controls who has access to it.. the clients each talk to the server. only works for textace (straight UNIX, no windows) NOT for graphical ace, i.e. a version for xace has been written but is not ready for distribution yet. Some help for server-client in given in nalusda ftp and later here.

subclasses.wrm *

Important features in this, see lecture.

That there are subclasses does not mean that this is a proper object oriented database. The subclasses are read in after the models.wrm file is read. Once defined, these can be used in queries or data selection where the normal class could otherwise be used. Can have maximum of 8 subclasses per class at the moment. Visible means that the subclass is displayed in class lists. Filter is the QUERY to get the objects matching this subclass definition.

psfonts and xfonts

specific post script fonts and standard fonts for X windows. can change either but should not have to. COLOUR under ps fonts controls the default for printing. psfonts is included in case other printers don't have the defaults assumed by acedb. (can change with the printer controller which comes up in acedb when printing.)

xfonts doesn't scale fonts but ps does. if the fonts work, best not meddle with them otherwise can use the command :

acedb -acefont 12 x 16 (or whatever you choose).

standard acefont is 8 x 13.

LOCK/NOLOCK

Locks are used so that no 2 people can alter data at the same time and therefore corrupt the database.

2 ways of setting up locks. UNIX way is there by default. This UNIX method does not guarantee that on a Network people's updates have not crossed in mid-path, either due to time taken to travel OR the caching by UNIX.

You can install a different mechanism.

Best system is NOLOCK i.e. only 1 writer and all the rest READERS.

Mac version is all single user so no locking.

models.wrm, A Brief Resume Of Models

For more on models : see lectures on theory and editing _models, also next document in this booklet : Models_guide, Models _used_in _code.

This file is a VERY important feature of acedb ; it is also the heart of the adaptability of acedb to store any kind of data.

MODELS.WRM is read EITHER when the database is fired up for the very first time OR when you ask it to read models during a session.

When you name a new object, an empty object is created which can be filled out as needed.

You can add data or comments to the database in 2 ways

(1) interactively

(2) through .ace files.

You can add extra information to objects after they have been initialised, because you don't have to fill out every field in each object. Its a permissive system, models don't require things to be filled.. You can add extra classes at any time. At the start, empty objects are created which can be filled out as needed. can add data/comments to db in 2 ways (1) interactively (2) through .ace files.

The models file controls the type of data put in and its structure. Model has same tree layout as object. Each model in the common set of models contains ALL possible tags and their structure. In different databases, different sets of tags will be selected and used. It is a strongly typed system. (meaning here that one can't put letters in number field).

EXPLANATION Of Some Terms Used

text = alphanumeric and punctuation marks. Some punctuation needs protecting by back slash e.g. inside a piece of text, quote marks must be given thus :

Remark : "Every \"dumpy\" gene is in the database."

int = integer number, whole negative or positive number.

float = decimal point number

UNIQUE = only a single value can be entered.

(in the tag or field immediately to the right of UNIQUE.

It can occur anywhere in a model. See below)

?Gene or **?Author** = a class

OR inside an object, it is a pointer to a gene or an author object.

Tags

The label given to a data field. (field description). Within a person's address, tags could be Phone, E_mail, Street etc.

Each tag label must be unique within 1 class model but the same tag name can be used again in another class. So can use the Tag Reference in Locus, Author, Rearrangement, Sequence etc. to

lead to field ?Paper. If you want to refer to a paper or a clone or an author twice in a model of one class have to use different tag names e.g.

Class Sequence

Hybridises_to ?Clone

Positive_probe ?Clone

Sent_to ?Author

Made_by ?Author

N.B. Can't have a Tag followed immediately by a Tag, because the

2nd one would not be rooted to anything.

?Text

2 types of text :

text without ? is data stored inside object and displayed with it.

can't search for it. ?Text are pointers to data NOT stored there but allows rapid searching of text. ?Text supports searching by global search which searches names of objects.

--->25

a display short-hand, all 25 whatevers are stored in object but not automatically displayed on the screen

UNIQUE

it can occur anywhere in a model, and governs the tag or field immediately to the right of UNIQUE.

Example :

UNIQUE int

only one integer can follow this tag, if a second is given, it overwrites the first unique one in context of this TAG

Example :

CLASS TAG *SUB_TAGS UNDER TAG REFERENCE*

?Paper Reference **Title** UNIQUE ?Text

Journal UNIQUE ?Journal XREF Paper

Publisher UNIQUE Text

Page UNIQUE Text UNIQUE Text

Volume UNIQUE Text Text

Year UNIQUE Int

Example :

?Clone

Cosmid_type UNIQUE YAC

BAC

Cosmid

where YAC, BAC and Cosmid are subTags. Subtags YAC BAC Cosmid can't be pointers AND only one of them can be given HERE.

CANNOT DO THIS

hybridises_to UNIQUE ?YAC

?BAC

CAN GET AROUND THIS AND USE POINTERS AS FOLLOWS

hybridises_to Type UNIQUE YAC UNIQUE ?YAC

BAC UNIQUE ?BAC

COSMID UNIQUE ?COSMID

i.e. the object type is specified, then that it is unique, then the pointer

UNIQUE specifies how many TAGS are to the right of it = ONE ONLY.

BUT if one has a construct like this e.g.

CONTAINS CLONE ?CLONE

GENE ?GENE

can give both types and more than one of each.

XREF

SETS UP CROSS-REFERENCES. If clone refers to a specific gene, one also wants the gene to refer back to that clone....reciprocal cross references. Using the XREF system, one doesn't have to enter the data twice, i.e. once in clone and once in gene.

to use, enter XREF followed by class/tag name.(the one used to set up XREF) If one XREFS a unique tag, it overwrites whatever is already there.

If one deletes one end of a X-reference, the system deletes the other end.

When the models.wrm file is read, the code checks the consistency of the cross-references.

DATE

a new type in acedb, datetype

format for dates : yy-mm-dd OR yy OR yy-mm

format when time is added after the underscore:

yy-mm-dd_dd:dd:dd

yy-mm-dd_dd

yy-mm-dd_dd:dd

where everything after _ is time i.e. hh:mm:ss.

date parser also recognises 1995 for 95. recognises NOW and gives whole date and time recognises TODAY and fetches this

REPEAT

rarely used but a legal keyword

e.g. ?Author Text REPEAT means many pieces of text on same line

Normally Gene ?Gene would put a list of genes broken into 1 per line but the use of REPEAT would put them all on the same line in DISPLAY

it is easier to add and delete objects displayed all in the same column. all text or object names are considered to be 1 to right of TAG.

#constructed_types e.g. Lab_location

refers to another model with own tags and data. it means import this model into here.

N.B. within some worm locations there are 2 freezer tags which seems illegal. freezer tags are not part of this model but only of the sublocation object . (see later in course)

```
/*@(#)displays.wrm        1.29 2/1/97 */

 /* wspec/displays.wrm

    This file is read only at execution time

    It holds the display definitions necessary to the graphic interface.

    You can freely edit this file without recompiling
    but it must remain consistent with the other wspec files
    and you must never change the graphType (-g) of a class
    unless you modify the corresponding application source code.

    Line starting with _DDispayType are parsed, they are expected
    to contain a la Unix a set of -Option parameters.
    DisplayTypes must match their enumeration in wspec/disptype.wrm

    Options names can be abbreviated, recognized options are
    -GraphType : One of PLAIN TEXT_SCROLL TEXT_FIT MAP_SCROLL as
                 enumerated in wh/graph.h. You cannot change that
                parameter unless you change the way the graph is
                used in the source code.
    -Title text : A default title for that graph, sometimes overridden
                by the applications.
    -Width w -Height h -Xposition x -Yposition y :
                Real numbers in the following range
                0 < x,w  < 1.3   ;  0 < y,h < 1.0
                They indicate the initial proportion of the screen width and height
                used to display objects of that class in their preferred display type.
                   The full screen is normalised to 1, 1.3

    -Menu text    Provide this display type, under name text, in the keyset "display as" menu.

    -Help section : The correct section of wspec/help.wrm,

    GraphType default as TEXT_FIT. x and y as 0, w h as .3 .5
```

The displays can be registered as the preferred display type of a class in the file wspec/options.wrm. In that case, you must provide a display Function and register it in wspec/quovadis.wrm.

```
_DDtMain -g TEXT_FIT -t "C. elegans 7/97 "  -w .44 -height .23 -help acedb

_DDtChrono -g TEXT_SCROLL -t "Chrono" -x .0125 -y .25 -w 0.6 -height 0.74 -help Chronometer

_DDtFile_Chooser  -g TEXT_SCROLL -t "File chooser" -w 0.5 -height 0.7 \
        -help File_Chooser

_DDtHelp -g TEXT_SCROLL -t "Help" -x .5 -y .15 -w  .9 -hei .8 -help Help

_DDtAce_Parser -g TEXT_SCROLL -t ".ace file parser" -x 0.4 -w 0.5 -heig 0.3 -help Read_files

_DDtDump -g TEXT_SCROLL -t Dump  -x .65 -y .04 -w 0.6 -height 0.6 -help Dump

_DDtKeySet -g TEXT_FIT -t KeySet -x .0001 -y 0.25 -w 0.44 -height 0.4 -help KeySet

_DDtSession -g TEXT_FULL_SCROLL -t "Session Control" -x  0.41 -w 0.64  -height 0.64  -help  "Session_control"

_DDtStatus  -g TEXT_SCROLL -t "Status" -x .25 -y .4 -w 0.8 -heig .7 -help Status

_DDtUpdate -g TEXT_SCROLL -t "Update" -x .65 -y .04 -w 0.6 -hei 0.6 -help "Official_Update"

_DDtLongText  -g TEXT_SCROLL -t "Text" - y 0.3 -w 0.75 -height 0.68  -help "Long-Text"

_DDtQuery   -g TEXT_SCROLL -t "Query" -w .65  -height 0.35 -help Query

_DDtQueryBuilder   -g TEXT_SCROLL -t "Query, by Gary Aochi" -w .73  -height 0.35 -help GaryQuery

_DDtQueryByExample   -g TEXT_SCROLL -t "Query by Example" -w .55  -height 0.35 -help Query_By_Example

_DDtSpreadSheet   -g TEXT_FULL_SCROLL -t "Table_Maker" -w .60  -height 0.35 -help Table_Maker

_DDtMULTIMAP   -g TEXT_FIT -t "Multi-map" -w 1.15  -height 0.65 -help Multi_Map

_DDtAction    -g TEXT_FIT -t "Action" -w .45  -height 0.65 -help Action

_DTREE  -g TEXT_SCROLL -w 0.55 -height 0.35   -x .2 -y .3 -help Tree

_DDtBiblio  -g TEXT_SCROLL -t Biblio -x 0.72 -y 0.22 -w 0.7 -height 0.75 -help Bibliography
```

_DCMAP -g TEXT_FIT -t Chromosome -y 0.3 -w 0.6 -height 0.75 -help Chromosome-map

_DFMAP -g TEXT_FIT -t Features -y 0.3 -w 0.80 -height 0.93 -help Feature-map

_DPEPMAP -g TEXT_FIT -t Protein -y 0.3 -w 0.80 -height 0.93 -help Protein-map

_DDtColControl -g TEXT_FIT -t "Column Control" -w .75 -height .5 -help Column-control

_DDtGel -g TEXT_FIT -t "Agarose Gel" -y 0.3 -w 0.35 -height 0.68 -help Agarose-Gels

_DGMAP -g TEXT_FIT -y 0.3 -w 0.75 -height 0.68 -help Genetic-map -m Genetic-map

_DVMAP -g TEXT_FIT -y 0.3 -w 0.75 -height 0.80 -help Vertical-map

_DGRID -g TEXT_FULL_SCROLL -w 0.54 -height 0.78 -help Grid -m Grid

_DPMAP -g TEXT_FIT -x 0.41 -w 0.88 -height 0.7 -help Physical-map

_DDtPmapPadSheet -g TEXT_SCROLL -t "Autoposition Clones" -w 0.70 -height 0.30 -help Contig-editor

_DDtPmapFingerprint -g TEXT_SCROLL -t "Fingerprint" -w 0.50 -height 0.93 -x .1 -y .03 -help Gel-Patterns

_DDtAlign -g TEXT_SCROLL -t Align -y .64 -w 0.55 -height 0.40 -help Make_maps

_DDtCodons -g TEXT_FIT -t "Codon Usage" -x .1 -y .5 -w .7 -height .85 -help Codon_usage

_DDtDnaTool -g TEXT_SCROLL -t DNA_Analysis -y .61 -w 0.32 -height 0.38 -help DNA_analysis

_DDtAlias -g TEXT_FIT -t "Add/Delete/Rename/Alias Objects" -w 0.5 -height 0.25 -help Add_New_Objects

_DDtPlotPolygon -g TEXT_FIT -t "Curve Plot" -x .01 -y .6 -w .3 -height .3 -help Plot_Curve

_DDtHistogram -g TEXT_FIT -t "Histogram" -x .01 -y .33 -w .5 -height .33 -help Histogram

_DDtMultiTrace -g TEXT_FIT -t "Traces" -x .01 -y .03 -w .9 -height .93 -help Traces

_DDtImage -g PIXEL_SCROLL -t "Image" -width .6 -height .4 -help Image

_DMETAB -g TEXT_FULL_SCROLL -t "Pathways" -width .6 -height .4 -help Pathways

_DSYN -g TEXT_FIT -x 0.05 -y 0.05 -w 1.2 -height 0.92 -help syn

_DDtAlignment -g TEXT_FULL_SCROLL -t "Alignment" -width .6 -height .4 -help Alignment

_DDtForest -g TEXT_FIT -x 0.05 -y 0.05 -w 1.2 -height 0.92 -help forest

```
/* @(#)options.wrm  1.19 12/1/96 */

 /* wspec/options.wrm

    This file is read only at execution time

    It holds the class definitions necessary to the applications.
    The kernel classes are defined in  wspec/sysoptions.wrm
    which can serve as a model when editing this one.

    You can freely edit this file without recompiling
    but it must remain consistent with the other wspec files
    and you must never change the type (A, B, X) of a class
    unless you reconstruct the database from ace files.

    Line starting with _VClassName are parsed, they are expected
    to contain a la Unix a set of -Option parameters.
    Class names must match their enumeration in wspec/classes.wrm

    The order in which classes appear here will be used in displays.

    Options names can be abbreviated, recognized options are
    -Hidden    : this class will be listed in the Main Window triangle.
    -Visible   : this class will be listed in the acedb Main Window.
    -Array     : this class is of type A, (Array or tuple).
    -Btree     : this class is of type B, it must be further defined in models.wrm
    -XREF      : Auto cross referencing Hidden B system class. Do not use.
    -Display  displayType : The preferred display type of the class,
                                as enumerated in wspec/disptype.wrm
    -Title  Tag : Tag must be a tag name listed in wspec/tags.wrm (or systags)
                  If present in some object of that class, the text following it
              will be used preferentially when referring to that object.
    -Symbol Tag : Tag must be a tag name listed in wspec/tags.wrm (or systags)
              If present in some object of that class, the text
              or key following it will be used preferentially
              when referring to that object in Maps.
    -Rename otherName : Other name will show in lists, old and new
                  name will be recognized in ace files and queries, old
                  is used in the code. In case some newName
                                matches some old name, the new name takes precedence.
    -CaseSensitive : The names in this class will be case sensitive.
                Never undo this option once it has been used.
```

```
    -Known : Prevents indirect creation of new object in ace files
          You can't enter a new object in this class directly
          but not by quoting it in another object.

    Classes default as -B -H -D TREE
    A classes further default as -H -D ZERO

    Displays:
      Every display type needs a display function.
    Parsers:
      Special code must be provided for A Classes
    Dumps:
      Special code must be provided to dump A classes

These routines must be registered in file wspec/quovadis.h before
    recompilation.
 */

// use the following for Arun Aggrawal's synmap
// but note that they will not work with tace unless you
// register the -M options in displays.wrm, with dummy values
//  _VMap             -V -D SYN -M STSMAP
//  _VAssembly        -V -D SYN -M DOGMAP
//  _VClone           -V -D SYN -M CloneMAP

_VMap             -V -D GMAP
_VSequence        -V -D FMAP
_VClone           -V -D PMAP

_VAuthor          -V
_VJournal         -V
_VPaper           -V -T Title

_VGene_Class      -V
_VLocus           -V -D GMAP
_VRearrangement   -V -D GMAP

_VAllele          -V
_VStrain          -V -T Genotype
_VLaboratory      -V

_V2_point_data    -H -T Genotype
_VMulti_pt_data   -H -T Genotype
```

_VPos_neg_data -H -T Results

_VMotif -V

_VContig -H -D PMAP

_VProtein -V -D PEPMAP
_VAlignment -V
_VMethod -V

_VGrid -V -D GRID

_VCell -V
_VExpr_pattern -V

_VMultiMap -H -D DtMULTIMAP
_VManyMap -H -D GMAP

_VPathway -V -D METAB
_VEnzyme -V
_VMetabolite -V

_VView -V
_VCell_group -H
_VLife_stage -H
_VPicture -H

_VHomology_group -H
_VMap_set -H

// The following lines are now obsolete and declared in kernel (chopped out now from text file).

```
// $Id: server.wrm,v 1.7 1997/11/12 11:42:53 mieg Exp $

// This file contains information needed to
// control the credentials of the clients connecting
// to the acedb server

////////////////////////////////////////////////////////////
// DATA_VERSION
// If data_version is larger than the one encountered previously
// the graphic clients will attempt to refresh their data
// If you run a rather static server, for distant clients
//  reset this number only after every major data change
// If you run a continuous server on which clients have write access
// set this number to zero, then the session number of the server
// will be used to trigger refresh,
// anyway all objects on the client are refreshed before any edition

DATA_VERSION 0  // ZERO for continuous refresh mode

////////////////////////////////////////////////////////////
// WRITE ACCESS:
// You must specify a directory that will be readable by
// any client to which you want to grant access

WRITE_ACCESS_DIRECTORY  /tmp

// On first connection from a client
// the server will write there very short temporary pass file 640
// The client must read this file before his second connection
// the server then destroys it.

// hence the correct way to set up things is to
// ensure that all users and the server process
// belong to the same group and that the client can mount the
// WRITE_ACCESS_DIRECTORY.
// This effectivelly restrict access to a set of user on trusted
```

```
// machines.

// Since the pass file is dynamic and modified each time
// you cannot crack it by multiple retries.

// Example
//  WRITE_ACCESS_DIRECTORY  /tmp
// everybody logged on the server machine can write
//
// ATTENTION
//  /tmp is not a good choice, because clients on other machine will
wait
//  for a few seconds for the pass file to appear in their own /tmp
//  A more fancy directory name is likely to be absent on the client
//  machine, and the client will proceed without waiting

////////////////////////////////////////////////////////////////
// READ ACCESS:
// There are 3 possibilities:
// 1: If you set READ_ACCESS_DIRECTORY PUBLIC

READ_ACCESS_DIRECTORY PUBLIC

// any client will have read access

// 2: If you specify a READ_ACCESS_DIRECTORY directory
//   for example READ_ACCESS_DIRECTORY /tmp
// The same strategy as for write access is used, and
// you can therefore limit read access to those users who
// can read a given directory.

// 3: If you set READ_ACCESS_DIRECTORY RESTRICTED
//   or if the READ_ACCESS_DIRECTORY clause is missing
//   only client with write access will have read access.

// Example
//  READ_ACCESS_DIRECTORY  /tmp
// everybody logged on the server machine can read
```

```
////////////////////////////////////////////////////////////////
// The following lines are used by the xclient
// To run this system, you need to have rpc installed on your
// UNIX machine.
// Normally these lines are automatically filled
// by the script wtools/xcl which as of may 97 is the way
// to start the graphic xclient
//
// As an example we give here the Montpellier worm server address

// EXTERNAL_SERVER 193.49.189.62
// EXTERNAL_PORT   20000100

// In case of problems or comments mail mieg@kaa.crbm.cnrs-mop.fr
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
```

```
// @(#)subclasses.wrm  1.8    2/19/97
// subclasses.wrm
// this is in standard ace file format
Class Gene
Visible
Is_a_subclass_of Locus
Filter Gene

Class Other_Locus
Visible
Is_a_subclass_of Locus
Filter "NOT Gene"

Class Tc_Insertion
Visible
Is_a_subclass_of Allele
Filter Transposon_insertion

Class Genome_Sequence
Visible
Is_a_subclass_of Sequence
Filter Genomic_Canonical

Class cDNA_Sequence
Visible
Is_a_subclass_of Sequence
Filter cDNA_EST

Class Repeat_Sequence
Visible
Is_a_subclass_of Sequence
Filter Repeat_consensus

Class wormpep
Visible
Is_a_subclass_of Protein
Filter Worm_pep
```

```
 // Reduced Model For Clone
?Clone  Remark General_remark  ?Text
                    PCR_remark  ?Text            // for PCR
        Position Map ?Map XREF Clone ?Map_position
                    Pmap UNIQUE ?Contig XREF Clone UNIQUE Int UNIQUE Int
                    Pos_neg_data ?Pos_neg_data
        Positive Hybridizes_to ?Clone XREF Positive_probe ?Grid
                    Positive_probe ?Clone XREF Hybridizes_to ?Grid
        Negative Negative_probe ?Clone XREF Does_not_hybridize_to ?Grid
                    Does_not_hybridize_to ?Clone XREF Negative_probe ?Grid
        Sequence ?Sequence  XREF  Clone
        Length Seq_length  UNIQUE  Int
                    Gel_length  UNIQUE  Float
        Location ?Laboratory #Lab_Location
        Gridded ?Grid
        FingerPrint Gel_number UNIQUE Int
                     Bands UNIQUE Int UNIQUE Int
        Type UNIQUE Cosmid
                    PAC
                    BAC
                    YAC Text                 // text is library name

 // model for the Lab_location subtree
#Lab_Location  Freezer  Text UNIQUE DateType
              LiquidN2  Text
              Minus70  Text
              Remark  ?Text


// example ace file to illustrate hash for Location

Clone cm15a7
Hybridizes_to POLY1 Y7B9
Hybridizes_to POLY1 Y9F5
Hybridizes_to POLY1 Y51B4
Sequence cm15a7 -C “5’ end only”
Lab_Location CB Freezer A1
```

Lab_Location CB LiquidN2 3:435

Lab_Location RW Freezer 4

Lab_Location Remark “regrown 7/92”

More Clone Examples

R219

PositionpMap	**ctg313**	-798	-785	
Fingerprint	Gel_number	364		
	Bands	163936		13

Y9F5

PositionpMap	**ctg313**	-806	-712	
	Positive-probe	**cm5c12**		
			m01f6	
			cm12g4	
			cm15a7	
Length	Gel_length	175		
Gridded	**POLY1**			

cm15a7

PositionHybridizes-to		POLY1	Y7B9	
				Y9F5
				Y51B4
Sequence	**cm15a7**		*5’ end only*	
Location	CB	Freezer	A1	
		LiquidN2		3:435
	RW	Freezer	4	
		Remark	regrown 7/92	

A Guide To Models And Ace Files

Mary O'Callaghan, October 1993
amended by Sylvia D. Martinelli May 1995, 1996, June 1997.

A) Objects, Classes and Models.

B) Contents of the Models

- Tags
- Data fields
- "UNIQUE" entries
- Tag entries which are objects
- "REPEAT" entries
- Tags which have no entries
- INCLUDED MODELS
- Editing models

C) Ace Files

- Adding/Editing ACEDB data
- Ace file operations
- Adding data
- Renaming data
- Deleting data
- Adding comments
- Ace file data that ACEDB does not see
- General Syntax Rules

D) Array classes

E) SUBCLASSES

Objects, Classes And Models

In ACEDB an "object" is an item, such as a sequence, a paper or an author, together with its attached data.

Illustration 1 shows a typical paper object, [cgc12].

^^

Illustration 1

[cgc12]

Reference Title "Critical oxygen tension of C. elegans"

Journal "Journal of Nematology"

Year 1977

Volume 9

Page 253 254

Author "Anderson GL"

"Dusenbery DB"

Type ARTICLE

Keyword "ENVIR CONDITIONS"

"OXYGEN"

ACEDB groups similar objects into "classes". Papers and multi-point-data objects, for instance, have a Paper class and a Multi_pt_data class respectively.

Each class has a "model", which provides the format for the contents of any object which belongs to that class.

There are two ways to see a class's model:

A) Pick the Model class in the main ACEDB window. This gives a listing of the various classes. Pick the class which is of interest and its model will be displayed.

B) To pick the relevant class in the main ACEDB window. :

Type ?Paper (or other classname) in yellow Search box, press return and you should get the Class Model Name in the main keyset window for selection.

Illustration 2

?Paper

Reference **Title** UNIQUE ?Text
Journal UNIQUE ?Journal XREF Paper
Publisher UNIQUE Text
Contained_in ?Paper XREF Contains
Year UNIQUE Int
Volume UNIQUE Int Text
Page UNIQUE Int UNIQUE Int
Author ?Author XREF Paper
Abstract ?LongText
Type UNIQUE Text
Contains ?Paper XREF Contained_in
Refers_to Locus ?Locus XREF Reference
Rearrangement ?Rearrangement XREF Reference
Sequence ?Sequence XREF Reference
Keyword ?Keyword

Contents Of The Models

Tags

Models state the "tags", or labels, that can be attached to the contents of objects in a particular class. The model for objects in the Paper class, for instance, includes Journal and Volume tags (See Illustration 2 above).

Tags are positioned to the left of the data entries they label. Tags are underlined in the Illustrations and other examples, so that readers can clearly distinguish between tags and their entries. Some tags are subtags of other tags. Illustration 3 shows the ?Author model, in which the Address tag contains several subtags. Illustration 4 shows a typical Author object, "Jones A".

Illustration 3

?Author
Full_name Text
Laboratory ?Laboratory XREF Staff
Address **Mail** Text
E_mail Text

Phone Text

Fax Text

Paper ?Paper

Illustration 4

Author "Jones A"

Full_name "Alan Jones"

Laboratory CB

Address **Mail** "25 Green Road, Cambridge"

E_mail "alan@mrc-lmb"

Phone "30535"

Fax "30536"

Paper [cgc12]

[cgc120]

Why have subtags?

An object is displayed in a clearer and more orderly fashion in ACEDB when the system groups together items which have something in common, such as parts of an address, as subtag entries within a main tag.

Also, if a group of tags are subtags of a main tag and we want to delete that group (and hence their related data), rather than having to delete each tag individually as we would have to if they were not subtags, we can simply delete the main tag. For instance, rather than deleting Mail, E-mail, Phone and Fax each time we want to delete address data in an Author object, we can simply delete Address.

Deletion details will be explained in section C.

Data fields

Models also indicate the nature of the data that can be given for each tag entry.

An "Int" data field to the right of a tag indicates that only an integer can be given for that data field. "Float" means that a floating point number must be used. If "Text" or "?Text" is stated, any ASCII characters can be entered. "?Text" data, unlike "Text" data, can be searched with the "Text Search" option in the main ACEDB window. There might be a few Ints and/or (?)Texts for one tag entry.

For example, the ?Sequence model includes the following:

promoter Int Int ?Text

A possible entry would be:

promoter 2520 2526 TATA

"UNIQUE" entries

In many cases, there is no restriction on the number of entries for a tag in an object. Sometimes, however, there can only be one entry. This makes sense. In the case of objects belonging to the Paper class, for instance, whereas it is important that the Author tag can include several entries, there should obviously be only one entry in the Journal or Volume tags.

If the word "UNIQUE" is used in a model for objects in a class, there can only be one item to the immediate right of the "UNIQUE" for every item to the left.

Often "UNIQUE" is positioned between a tag and a data field.

This means that there can only be one entry in that data field for that tag.

For instance, the ?Paper model includes the following:

Volume UNIQUE Int Text

This indicates that each Paper object can only contain one entry in the Int field of the Volume tag, though there can be several entries in the Text field.

"UNIQUE" can also be placed between two data fields in a model's tag. For instance, the ?Sequence model contains:

misc-feature Int UNIQUE Int

In this case, a misc-feature entry in a Sequence object can have only one integer in its second field for each one in its first.

Some sub_tag entries in one TAG are mutually exclusive (i.e. a multiple-choice type tag). The options are labelled with subtags. In the model there will be a UNIQUE between the tag and the subtags to indicate that in a given object a tag can only have one of the subtags (and its entry). The ?Allele model, for instance, includes:

Source UNIQUE **Gene** ?Locus

Gene_class ?Gene-Class

An Allele object might include the following:

Source Gene_Class unc

let

or:

Source Gene unc-33

let-45

However, it should not contain the following:

Source Gene unc

Gene_Class unc-33

Tag entries which are objects

Objects can have entries which are themselves objects, in the same or another class. For instance, Paper objects have an Author tag, the entries for which are objects in the Author class. In models for objects which include other objects in their tags, the syntax is as follows:

Tag ?Class

(for instance, **Author** ?Author in the case of the Paper model)

This indicates that within an object with that model, any entry in the stated tag will be an object in the stated class.

What implications does the fact that a tag's entry is an object have for the user? An example will help to make this clear.

The Mapper tag for a Multi-pt-data object "ABC" contains the entry "Jones A". The Mapper tag section of the Multi-pt-data model states that Mapper entries are members of the Author class (i.e. **Mapper** ?Author). As a result, when the text relating to the "ABC" Multi-pt-data object is displayed in a window, an entry in its Mapper tag, "Jones A", will be in bold face to indicate that it is an object in its own right. If "Jones A" is selected with the mouse, the "Jones A" object, together with its related tags and data, will be displayed in a new window.

XREF

Sometimes when tag entries are described in models as objects in a particular class, the words XREF and a tag name appear to the right of the class. e.g. the Sequence model contains:

Clone ?Clone XREF Sequence.

This establishes a cross-reference from the tag entry, which is an object in its own right, back to the object in which it is a tag entry.

In the example just given, Clone tag entries are described in the ?Sequence model as "?Clone", or objects belonging to the Clone class. What exactly do the words "XREF Sequence" mean? They indicate that the model for Clone objects contains a Sequence tag, entries for which are defined as "?Sequence", or objects belonging to the Sequence class. If a Sequence object contains a Clone object in its Clone tag, then that Clone object will automatically have the same sequence as an entry in its own Sequence tag. e.g. If we have a Sequence object "ABC", whose Clone tag entry is "Clone35", then the Clone object, "Clone35", will automatically include the Sequence object "ABC" in its own Sequence tag.

When models.wrm is read into the database, code checks the consistency of the models to one another.

There are some pitfalls to avoid when using XREF. If both the classes/tags referred to are UNIQUE or non_UNIQUE, there is no problem. If one is UNIQUE and the other isn't, there can be overwriting and deletion problems, without one realising.

REPEAT entries

If the word **REPEAT** is stated after a data field in a tag, then that data field's entries can each contain several items of a particular kind on a single line.

The Grid model includes the following TAG Row :

Row Int ?Grid_row

where Class Grid_row is a referred_to class which itself contains use of REPEAT :

UNIQUE **Clone** UNIQUE ?Clone XREF Gridded REPEAT

Looking at the second half of the line, we see that there is a REPEAT statement for ?Clone entries Hence the ?Clone part of two row entries might contain the following:

Clone1 Clone2 Clone3 Clone4

Clone5 Clone6 Clone7 Clone8 Clone9

Why have a UNIQUE before ?Clone and REPEAT after it?

We learnt above that UNIQUE means there can only be one of what is to the right of UNIQUE for each element on the left. Since the row tag entry begins with Int, there can only be a single-line group of clones for each integer given.

Hence you could have the following:

Row 1 A1 A2 A3 A4
2 A6 A7 A8 A9 A10

but not:

Row 1 A1 A2 A3 A4
A6 A7 A8 A9 A10
2 B1 B3 B6 A5

Tags which have not got field data descriptions entries.

Some subtags in objects do not contain data definition entries (such as int, text). This is deliberate. In computer terms, they are flags which can be set to a value. In Pos_neg_data, each piece of data gives a Positive or Negative result (such as deleted/non deleted) which could be used in the graph drawing to display genes deleted in one colour, those n ot deleted in another. In Class Cell_group one flag can be set to UNIQUELY to Tissue, Region or Organ, meaning that the specified group is ONE of those categories. No further text explanation is needed there. The Allele model, for instance, includes the following:

Description Recessive
Dominant
Semi-dominant
Weak

meaning that a particular allele can be described adequately by one of these words without further description.

INCLUDED MODELS (and UNROOTED TAGS)

UNROOTED TAGS are not permitted.

All the tags in a model must be rooted, i.e. all the nodes in a model between a tag and the root must be other tags, not data or pointer nodes. An example of an unrooted tag would be "Location" in the following model:

?Clone Laboratory ?Laboratory **Location** Freezer Text

Liquid_N2 Text

Where you want to record the name of the Lab and whether the clone is in a freezer, fridge, liquid nitrogen and which one of them etc.

The model shown above is invalid.

To get round this limitation, an included model (sometimes called a constructed subtype) can be used instead. e.g.

Clone Laboratory ?Laboratory **#Lab_location**

?Lab_location Location Freezer Text

Liquid_N2 Text

Other_loc Text

The # symbol means something like "put the following model inside the current model in this position. Lab_location is the name of a second model. However all the data will be contained inside the Clone class. Lab_location is NOT a class, and there is no data directly associated with it. An example of an ace file with data for these models is:

Clone XYZ

Laboratory "CB" Freezer "F1"

Laboratory "CB" Liquid_N2 "AS2"

Laboratory "RW" Other_loc "fridge2"

Laboratory "RW" Freezer "F1"

(the two freezer1 designations do not cause a problem, both are unique since they have different laboratories.)

This whole system of including models is recursive, i.e. the included model (introduced by the #) can contain an included model of its own, etc..

Querying data which corresponds to part of an included model is more complicated than querying the simple data. A possible query is

Clone XYZ Laboratory = CB # Freezer = F1

i.e. go to specific laboratory, give a '#', then go into the included model. One can't XREF into an included class but one can XREF out of it.

N.B. in case you hadn't got it....

Lab_location is NOT a class but a model or sub_type.

Editing the models

If you want to change a class model, or perhaps add one, this must be done in the models.wrm file. If a new class is created there should also be a new line in options.wrm.

(**Note** it is possible to NOT use a tag in any object if no data is given for that field (for a normal data field) or one cannot decide which flag to set for these NO-data tags.)

Ace Files

Adding/editing ACEDB data

There are two ways of editing data in ACEDB.

First, objects can be edited manually using the Add/ Delete/Rename

option in the main ACEDB window menu and/or the Update option in the text window for any object. The second solution is to import data using files in "Ace" format.

What is an Ace file?

If a file is in Ace format, ACEDB can interpret the message it contains. Ace files can be used to add, delete, and rename objects, and to add data and comments to objects and delete data from them. The files, which must have the ".ace" extension, are read into ACEDB using the "Read Ace Files" option in the menu of the main ACEDB window.

Ace File Operations Adding data

To add an object to the database, first state its class and name (e.g. Journal Cell) on a single line in an Ace file. Data can be attached to this object on immediately subsequent lines. The data must be prefixed with suitable tags, such as Author, Journal, Volume and Page in the case of Paper objects. An object's potential tags are listed in the model for the class of objects to which that object belongs. The steps are the same when the object is already in ACEDB and we simply want to add to its data. If, for instance, we wanted to add an author and a volume number to the paper "[wbg6]", we could put the following:

Paper [wbg6]

Author "Jones A"

Volume 39

Note: I have only used bold for the tags in examples in order to make the examples clearer. There should not be any bold in an Ace file.

If you want to add several entries for a particular tag in an object, each of the entries must be listed on a separate line and each line must begin with the tag in question. The order of the lines doesn't matter if, for instance, we needed to add several Gene tag entries to the Paper object, [wbg6], we might put the following in an Ace file:

Paper [wbg6]

Gene let-31

Gene let-32

Gene let-45

As mentioned earlier, some tags have subtags. In this case only label the data with the subtags; there is no need to mention the tag containing the subtags since ACEDB will know the tag to which the subtags belong, from the models. If a subtag has no description to its right in a model, there should be no data to the right of that subtag when it is appended to an object in an Ace file. Your choice of a subtag or subtags is adequately descriptive in itself. When adding data in an Ace file, it is important to have the correct data types (e.g. integer, floating-point) for each part of an object's tag entry, and to have those parts in the correct order. The format is defined in the object's class model (e.g. see the definitions given to the right of the tag entries shown in the Paper class model in Illustration (2).

If the word "UNIQUE" is in the model, you can only have one of what is on the right of the "UNIQUE" for each element on the left.

The Paper model contains:

Publisher UNIQUE Text.

This means that there can only be one entry for Publisher in a Paper object.

Usually in ACEDB when a new entry is added for a particular tag in an object (e.g. "Jones A" for the Author tag in a Paper object), the new tag entry will be added to any previous entries. However, if a tag can only contain one entry, the new entry will overwrite the old one.

For a full discussion of tags, data types, their order, "UNIQUE" etc, see section above.

Note if mistakes are made in an Ace file, ACEDB will point them out when attempts are made to read in the file.

Renaming data

If you want to rename an object the Ace file syntax is as follows:

-R Classname Old_object New_object

e.g. -R Author "Jones A" "Jones AB"

Deleting data

The Ace file syntax for deleting an object is as follows:

-D Classname Objectname

e.g. -D Author "Jones A"

To delete data which is attached to an object, first give the object's class and name on a single line. Then on subsequent lines say "-D Tagname" for each of the tags whose entries you want to delete. You only need to say "-D Tagname" once for each tagname in an object no matter how many entries that tag has. Each "-D Tagname" should be on a separate line. For example, the following statements will delete all the papers and laboratories associated with the Author "Jones A":

Author "Jones A"

-D **Paper**

-D **Laboratory**

If you want to delete entries for a particular tag in an object and add in new entries for that tag, the order of the Ace file statements is important. The deletion statement should come before you add in the new data, since if the deletion comes after the new data is added, both the new and old data will be deleted. For instance, if, in addition to wanting to delete the current papers for the Author object "Jones A", you wanted to add in some new Paper tag entries, you could say:

Author "Jones A"

-D **Paper**

Paper "Worm tales"

Paper "Worm mysteries"

If it is stated in a model that there can only be one entry for a particular tag (e.g. "Title UNIQUE Text" in the Paper model - see the discussion of "UNIQUE" above), and you want to replace an object's entry for that tag, there is no need to say "-D tagname", since the new entry will automatically overwrite the old.

Sometimes a tag has subtags. How is data deleted when this is the case? If you only want to delete the data in some of a tag's subtags then the Ace file syntax is:

Classname Objectname

-D **subtag1**

-D **subtag3**

There is no need to mention the tag which contains the subtags. If however, you want to get rid of the contents of all the subtags in a tag, rather than saying -D subtag for each of the subtags, you could have the simpler equivalent, "-D tag". e.g. in the case of Author objects you can have -D Address, rather than:

-D Mail

-D E_mail

-D Phone

-D Fax

Adding comments

Comments can be added to any tag entry for an object. These comments will be added into ACEDB. In an Ace file the syntax is:

Tag TagEntry -C Comment

e.g. we could have the following for a Paper object, [cgc12]:

Paper [cgc12]

Author "Jones A" -C "and the rest of the gang"

Page 456 467 -C "C. elegans growth patterns"

In ACEDB comments are displayed with a dark background.

Ace file data that ACEDB does not see

If you want to put something in an Ace file which you don't want ACEDB to try to interpret, such as a description of the contents of the Ace file, prefix your note with "//". Anything following this to the end of the line will be ignored.

General Syntax Rules

Objects listed in the Ace files should

be separated by a blank line, as in the case of the

Paper objects below.

Paper [cgc12]

Journal Nature

Page 10 16

Paper [cgc13]

Journal Nature

Page 17 24

A data item should be enclosed in quotes if it contains a space. Otherwise the data after the space will be lost. The following Author object is an example:

Author "Jones A"

Mail "6 Blackheath Park"

Phone "044 656565"

If a tag entry has multiple parts there should only be, quotes around individual parts e.g. the Sequence model includes:

promoter Int Int ?Text

In this case you might have the following:

promoter 2520 2526 "TATA signal"

But you should not have:

promoter "2520 2526 TATA signal"

Sentences in quotes can spread over several lines, as long as no carriage returns are included.

Array Classes

Some ACEDB classes have an array structure rather than the standard tree structure. Objects in these classes need special treatment in an ace file.

LongText and DNA

The LongText array class objects are long pieces of text (usually abstracts). The syntax is as follows:

LongText [wbg11.1p68]

This is an intricate worm family saga spanning several generations.

It can contain blank lines, because there is a special symbol for the end of the entry.

LongTextEnd

DNA objects are pieces of dna. The syntax is as follows:

DNA CEMSA02.f

TCGTTAAGAATTGGAAGTTCCGATGTTAGTGAAAATGAGAAGAAGGAGCTGAAGAAGAGAAAGCTTATCAGTGAAGTAA

ACATCAAAGCATTGGTGGTTTCCAAGGGAACATCTTTCACCACTAGTCTTGCAAAGCAGGAAGCTGATTTGACTCCGGA

AATGATTGCTTCTGGTTCATGGAAAGACATGCAATTCAAAAAGTATAATTTCGATTCACTCGGAGTTGTTCCGTCATCT

GGGCATCTGCATCCATTAATGAAAGTGCGGTCTGAATTCCGACAAATCTTCTTCTCAATGGGATTTTCTGAAATGGCGA

CAAATCGATACGTGGAGTCGTCTTTCTGGAACTTTGATGCCCTTTTCCAACCTCAACAGCATCCTGCAAGAGATGCTCA

TGATACTTTCTTCGGTTCTGATCCCGCGATTAGCACGAGTTCCCTG

Everything is taken up to the next blank line. Letters can either be upper or lower case, and should be IUPAC codes for nucleotides, e.g. A, C, G, T, N for anything, R for purine.

DNA and LongText objects should be attached to other objects. The model for Paper objects includes the following:

Abstract **?LongText**
A Paper and its related LongText abstract might be listed in an ace file as follows:

Paper [wbg11.1p68]
Abstract [wbg11.1p68]
Author "Jones J"
Author "Smith T"

LongText [wbg11.1p68]
This is the full text of a fascinating article by Leon Avery on
nuclear protein extracts. The only important thing about it is that
the final line is as follows, exactly, without any spaces after the
stars at the end.
*****LongTextEnd*****

The Paper and LongText objects are usually given the same name.

Sequence objects include the following:

DNA ?DNA

A sequence and its DNA might be listed in an ace file as follows:

Sequence CEMSA02.f
From_Author "Kerlavage AR
Library Genbank M79466
Reference [cgc1567]
DNA CEMSA02.f
Related_sequence CEMSA02.r
Brief_Identification "Phenylalanyl-tRNA synthetase beta

DNA CEMSA02.f
TCGTTAAGAATTGGAAGTTCCGATGTTAGTGAAAATGAGAAGAAGGAGCTGAAGAAGAGAAAGCTTATCAGTGAAGTAA
ACATCAAAGCATTGGTGGTTTCCAAGGGAACATCTTTCACCACTAGTCTTGCAAAGCAGGAAGCTGATTTGACTCCGGA
AATGATTGCTTCTGGTTCATGGAAAGACATGCAATTCAAAAAGTATAATTTCGATTCACTCGGAGTTGTTCCGTCATCT
GGGCATCTGCATCCATTAATGAAAGTGCGGTCTGAATTCCGACAAATCTTCTTCTCAATGGGATTTTCTGAAATGGCGA

CAAATCGATACGTGGAGTCGTCTTTCTGGAACTTTGATGCCCTTTTCCAACCTCAACAGCATCCTGCAAGAGATGCTCA
TGATACTTTCTTCGGTTCTGATCCCGCGATTAGCACGAGTTCCCTG

The Sequence and DNA objects are usually given the same name.

The other main array class is the KeySet class. The syntax for adding a KeySet via an ace file is as follows:

KeySet nameofkeyset

Classname Objectname // repeated up to next blank line

The following is an example of an ace file keyset:

KeySet GenesfromBill

Locus unc-32

Locus lin-19

Locus dpy-40

Locus cll-3

KeySet DatafromBen

Locus unc-31

Clone C05G2

// OLD end of file

E) SUBCLASSES :

They are not true subclasses (in the object-oriented sense), merely a pre-specified subset of objects from a class. they can appear in the main window as 'classes', be used in queries and in .ace files. When the models.wrm file is read, subclasses.wrm is also read. Only 8 subclasses are permitted per class at the moment (limited by 8 bits per key which can be set in memory).

Example :

?Clone Type UNIQUE **Cosmid**

BAC

PAC

where cosmid, pac and bac are subclasses. When a clone object is added to the database, a tag is set specifying which type in the .ace file. The database tests whether the tag is set when the object is stored, they are cached separately and can be retrieved quickly.

The Class Syntax

Class1 **Tag1** Field1
Tag2 Field2
Tag3 **Tag3a** Field3a //subtags
Tag3b Field3b
Tag3c Field3c
Tag4 Field4a Field4b Field4c
Tag5 // a flag

Where Field = Float, Int, Text, ?Text, Object_Identifier.

Each Tag name must be Unique within one class.
The same Tag name can be used in another class

Class1 Tag1 Field1
Tag2 Class2 XREF Tag_in_Class2
Tag3 **Tag3a** Field3a
Tag3b UNIQUE Field3b
Tag3c Field3c REPEAT
Tag4 Field4 #Sub_model
Tag5

?Sub_model Tag6 Field6

Altering An Existing Class

You have to alter **models.wrm** and after reading in the new model, read in a .ace file which uses the new tags. Make the following changes in wspec/models.wrm, quit ACEDB and restart. Then pick the "Read Models" option from the menu in the main window, then try to read in the file changemodels.ace, using the "Read .ace files" tool.

In **models.wrm** edit the ?Journal model to look like this:

```
?Journal          Paper ?Paper    XREF    Journal
        Address  Mail Text
                 E_mail Text
                 Phone Text
                 Fax Text
         Editor  Text
        Brief_description  ?Text
```

//new data for changed model of Journal

```
Journal Cell
Mail "5 Green Road"
Mail "Cambridge"
E_Mail "cell@cam.ac.uk"
Paper [cell1]
Paper [cell2]
Brief_description "This  Journal describes proteins, nucleic acids, carbohydrates and lipids"

Journal Nature
Mail "5 Green street"
Mail "Islington, London"
Phone "010 353641"
Editor "John Maddox"

Paper [cell1]
Title "Proteins and their functions"
Author "Albertson DG"
```

```
Paper [cell2]
Title "Lots of lipids"
Author "Foley S"
Author "Sweetnam N"
```

Adding A New Class (Tissue)

To add a new class to ACEDB open relevant files from **wspec** directory in a text editor. (It is possible to add a new class to the end of your **options.wrm** file using the CAT command [cat >> options.wrm << EOF] but this method is very inconvenient for altering the models.wrm file.)

//inside the text editor, add to the end of the **options.wrm** file these lines :

```
_VTissue -V
```

//inside the text editor, add to the end of the **models.wrm** file these lines :

```
//blank line
?Tissue         Remark ?Text
                Contained_in ?Tissue XREF Contains
            Contains ?Tissue XREF Contained_in
            Expression Locus ?Locus XREF Expressed_in
                        Sequence ?Sequence XREF Expressed_in
//blank line
```

// NB still in **models.wrm**, add lines to the classes ?Locus and ?Sequence somewhere in the middle of the model

?Sequence

Expressed_in ?Tissue XREF Sequence

?Locus

Expressed_in ?Tissue XREF Locus

These new lines can go anywhere in these class models, but choose somewhere that looks logical, or a position that you can find again easily.

//Edit **layout.wrm** by adding this new class to the main class window display

WARNING left justification of Class names and TAGs is very important.

If it is not precisely correct, you will get error messages from acedb.

```
// This is an annotated version of the models.wrm file for version 4
// of ACEDB that contains in principle all and only those tags that
// are used in the code somewhere, with brief comments on what they do.

// 6/6/96 : amended by Simon Kelley. more to do !

////////////////////////////////////////////////////////////////////// Genetic map

?Map    No_cache
               // Prevents cacheing if maps as objects in class gMap
               // Use if your map chnges frequently and you don't want to
               // have to manually rebuild it
        Display Non_graphic
                       // Prevents graphic display
               Title UNIQUE ?Text
                       // displayed on top line
               Flipped
                       // Then coordinates go upwards
               Centre UNIQUE Float UNIQUE Float
                       // First number is default coordinate to centre on
                       // when displaying the map not from a gene.  Second number
                       // is default display width.
               Extent UNIQUE Float UNIQUE Float
                       // The coordinates of the two ends of the chomosome,
                       // for drawing the locator image at the left of the display.
               View ?View
                       // list of view objects for controlling display of this map
               Default_view UNIQUE ?View
                       // view use when map first displayed.
               Minimal_view UNIQUE ?View
                       // use this when more than 1 map displayed
        Inherits From_map UNIQUE ?Map
                Author Text
                // Add objects from the inherits_from map to the
                // display, if a version of the object doesn't exist
                // in this map. Can be used to make different objects
                // which display the same map, with different views.
                // These have only Default_view, and inherits from.
                // Also used by the "Private save" function in gMap.
        Main_Marker <tag2> <object>
               // items shown left of locator at full map resolution
        Map ?Map Xref Map_shown #Map_position
            // Maps may contain maps, or include them.
        Includes ?Map
            // Put all the objects in the map into this map, positions
            // suitably scaled. The map must have a position on this
```

```
            // map, and an extent.
        Contains <tag2> <object>
                // list of all primary items drawable on the map
                // each needs a "Map ?Map #Map_position" line

// the following can occur in any class, to make it display on a map

<class> Map ?Map #Map_position
        Positive <tag2> <object>
                // object will highlight appropriately when parent does
        Negative <tag2> <object>
                // object will highlight appropriately when parent does
        Mapping_data   2_point ?2_point_data
                       Multi_point ?Multi_pt_data
                       Pos_neg_data ?Pos_neg_data
                // these three refer to mapping data objects that can
                // be displayed and take part in likelihood calculations
        More_data <object>
                 // look for Positive, Negative and mapping_data in
                 // <object> too.

// the position to draw the object, and whether it is a point or a line,
// are determined by the #structure in the object to the right of the
// specific map.

?Map_position UNIQUE   Position UNIQUE Float #Map_error
                       Ends    Left UNIQUE Float #Map_error
                               Right UNIQUE Float #Map_error
                       Multi_Position  Float #Map_error
                       Multi_Ends Float UNIQUE Float
                       With UNIQUE <tag2> UNIQUE <object> #Map_offset
                               // tag2 system for "burying"

?Map_error Error UNIQUE Float

?Map_offset Relative #Map_position

// multimaps - work via table maker
```

```
?MultiMap Map ?Map
          Min Int // keep loci appearing on at least min maps. Default = 2
          Anchor UNIQUE Text UNIQUE Text UNIQUE Text  // Class tag1 tag2
                           // i.e Locus Homeology_group Homelogs
                           // class members on different maps with the
                           // same Anchor tag will be chained

////////////////////////////////////////////Views - control Map and Grid displays

// Almost everything in these is meaningful.  The data is best set for
// Map displays by the interactive view editor.

?View   Type UNIQUE    Gmap
                       // display and columns below belong to this option
                       Grid Grid_map Int #View_tags
                              // Int is priority
                              // View_tags structure gives coloiurs and tags
                            Grid_edit_default UNIQUE Text
                              // Text is tag to set/unset on click editing
                            Grid_edit_menu Text
                              // List of tags to set/unset from menu
       Display Submenus
                       // if set get submenus on item boxes
              Cambridge
                       // if set get minor variations in drawing style
              No_buttons
                       // if set suppress header buttons for WWW
       Name    UNIQUE  Text
                       // ??
       Columns Text UNIQUE Int #Column
                       // the key section
                       // Text is name; Int 0 for Hidden, 1 for Visible


// the column structure is a list of different column types, with
// any parameters

?Column UNIQUE  Scale   Scale_unit UNIQUE Float       // minimum increment
                       Cursor Cursor_on
                              Cursor_unit UNIQUE Float
              Locator Magnification UNIQUE Float
                       Projection_lines_on
              Marker_points
              Marker_intervals
              Contigs
```

```
Reversed_physical
Physical_genes
Two_point
Multi_point
Likelihood
Points  Point_query UNIQUE Text
        Point_yellow UNIQUE Text
        Point_width UNIQUE Int // max width if not at RHS
        Point_error_scale UNIQUE Float
        Point_segregate_ordered
        Point_show_marginal
        Point_pne #Colour // positive, no error
        Point_pe  #Colour // positive, error
        Point_nne #Colour // negative, no error
        Point_ne  #Colour // negative, error
        Point_symbol UNIQUE Text
Interval_JTM   #Interval_col_conf
Interval_RD    #Interval_col_conf
Interval_SRK   #Interval_col_conf
Derived_tags   DT_query UNIQUE Text
               DT_width UNIQUE Int
               DT_no_duplicates
               DT_tag Text Int
Spacer  Spacer_colour #Colour
        Spacer_width Float
RH_data        RH_query UNIQUE Text
               RH_spacing UNIQUE Float
               RH_show_all
               RH_positive #Colour
               RH_negative #Colour
               RH_contradictory #Colour
```

```
?Interval_col_conf     Query UNIQUE Text // Query
                       Names_on
                       Width UNIQUE Int
                       Symbol UNIQUE Text
                       Colours Text #Colour // Only for chrom_bands.
                       Pne #Colour // positive, no error
                       Pe  #Colour // positive, error
                       Nne #Colour // negative, no error
                       Ne  #Colour // negative, error

////////////////////////////// Physical map - PMAP display - will be subsumed in Map

?Contig pMap    UNIQUE Int UNIQUE Int
                // min, max contained objects, in pMap units
         Clone  ?Clone XREF Contig
                // list of clones

?Clone  Remark General_remark ?Text
                       // show always in the Remark field of PMAP display
                Y_remark ?Text
                       // only show when "Show all remarks"
                PCR_remark ?Text
                       // only show when "Show all remarks"
        Position  pMap UNIQUE  ?Contig XREF Clone  UNIQUE Int UNIQUE Int
                // endpoints of the clone
        Positive       Hybridizes_to ?Clone_Grid ?Clone XREF Positive_probe ?Text
                // used by GRID display to determine pattern, and by PMAP
                // display to generate implicit positions for probes (top
                // lines of display).
                       Positive_probe ?Clone
                // reciprocal of Hybridizes_to - see above
                       Positive_locus ?Locus XREF Positive_clone
                // used to attach Loci to the physical map
        Sequence ?Sequence XREF Clone
                // used to draw yellow sequence boxes on PMAP display
        Gridded ?Clone_Grid
                // if Clone_Grid == POLY1, makes the clone line thicker!
        FingerPrint    Canonical_for ?Clone UNIQUE Int UNIQUE Int
                               // will draw with a (*) indicating buried clones
                               // and can fetch the buried clones
                       Bands
                               // will draw the clone in the "Cosmid" part of the map
        Contig9 Vaxmap UNIQUE Float
                       // used to find contig order in physical chromo map
                In_Situ  UNIQUE Int UNIQUE Int
                       // percent of physical chromosome
```

```
                        // used by obscure physical chromo map function
                Cosmid_grid
                        // makes the clone line thicker
                Canon_for_cosmid
                // makes the clone line thicker

////////////// grids of objects, traditionally clones gridded for filter hybridisation

?Grid Title ?Text
        Layout  Lines_at Int Int       // x, y spacing
                Space_at Int Int       // x, y spacing
                No_stagger
                        // default is with alternate lines staggered
                A1_labelling
                        // label spaced blocks a-h down LHS, 1-12 across top
        Row Int #Grid_row
                // the items in the grid
        Virtual_row Int UNIQUE ?Grid XREF In_grid REPEAT
                // alternatively, a set of interleaved component grids
        View ?View
                // controls the colours and editing process

?View_tags  Colour #Colour
            Surround_colour #Colour //colour when surround
            Tag Text  //could be many tags

?Grid_row UNIQUE <tag2> UNIQUE <object> XREF Gridded REPEAT
                        // objects of a single class in one row - one tag only
                 Mixed #Mixed_grid_row
                        // a row of objects from mixed classes

?Mixed_grid_row UNIQUE <tag2> UNIQUE <object> XREF Gridded #Mixed_grid_row
                // the objects in a mixed row, all with their own tag

?Pool   Contains <tag2> <object> XREF In_pool
                 Subpool ?Pool XREF In_pool
                        // etc. Contains is tag2
        In_pool ?Pool XREF Subpool

////////////////////////////////////////////////////////////////// sequence stuff

?Motif  Match_sequence UNIQUE Text
                // e.g. restriction site, used by DNA analysis window
```

```
?Sequence // displayed in text column at right of fmap
          DNA UNIQUE ?DNA UNIQUE Int  // Int is the length
               // used to get the DNA
          Structure  From      Source UNIQUE ?Sequence
               // connects sequences together
                              Source_Exons Int UNIQUE Int // start at 1
               // defines splicing pattern
                    Subsequence ?Sequence XREF Source UNIQUE Int UNIQUE Int
               // connects sequences together
          Visible <tag2> <object>
               // object name will be shown with tag2 in the text section
               // at right, and will be clickable
          Properties   Transposon Text
                              // makes subsequence visible
                       Coding  CDS UNIQUE Int UNIQUE Int
                              // makes subsequence visible, splicable and translatable
                       Transcript
                              // makes subsequence visible and splicable
          Assembly_tags        Text Int Int Text
               // type, start, stop, comment
               // all these will be drawn in the Assembly tags column
          Allele ?Allele UNIQUE Int UNIQUE Int UNIQUE Text
               // allele object, start, stop, replacement sequence
               // if an insertion point Text is transposon name (distinguished
               // by containing non ACTG letters), and (n, n+1) = T A, so indicates
               // direction (if known).
               // if a deletion, put '-' as the replacement sequence
          EMBL_feature  <tag2> Int Int Text
               // tag2 is typically the feature key
               // a box is drawn between the Int positions, and the Text
               // is shown with the key to the right
          Homol <tag2> <object> ?Method Float Int UNIQUE Int Int UNIQUE Int
               // creates the homology boxes.
               // the Method determines which column to drawn in and how
               // the Float is the score, next two ints are the position
               // in the current sequence, and next two the position
               // in the target sequence.
          Feature ?Method Int Int UNIQUE Float UNIQUE Text
               // Method determines column and display style
               // Ints are start, end of feature
               // Float is score
               // Text is shown on select, and same Text are neighbours
          Feature ?Method
               // all features will be drawn in the same style
```

```
?Method Display Colour #Colour
                                // colour of box
                Frame_sensitive
                                // if set, then three columns, one per frame
                Strand_sensitive
                                // if set, only show positive strand boxes
                Score   Score_by_offset
                                // higher score pushes to right (default)
                        Score_by_width
                                // higher score boxes are wider
                        Score_bounds UNIQUE Float UNIQUE Float
                                // scores for saturation at min width, max width
                        Percent
                                // adds '%' in text
                Bumpable
                        // boxes bump right, don't stack (unimplemented)
                Width UNIQUE Float
                        // width of box
                Symbol UNIQUE Text
                        // at top of column; first char only is used
                Right_priority UNIQUE Float
                        // position of column; the greater the more right
        Blastn          // can calculate percent from score if blastn
        Blixem  Blixem_X
                        // can show in Blixem as protein vs DNA
                Blixem_N
                        // can show in Blixem as DNA vs DNA

////////////////////////////////////////////////////////////// bibliographic stuff

?Author Address E_mail Text    // used for mailing in treedisp.c, action.c
        Paper ?Paper

?Paper  Reference       Title UNIQUE ?Text
                        Journal UNIQUE ?Journal XREF Paper
                        Year UNIQUE Int
                        Volume UNIQUE Text Text
                        Page  UNIQUE  Text UNIQUE Text
        Author ?Author XREF Paper
                // all the above used by biblio display
```

```
////////////////////////////////////////////////////////////////////// Map data --
////////////////////all the following fields are used by the genetic map data

// calculation package

?2_point_data  Point_1 <tag2> <object> XREF 2_point
               Point_2 <tag2> <object> XREF 2_point
                       // the two objects that are mapped
               Calculation UNIQUE Full UNIQUE Int UNIQUE Int UNIQUE Int UNIQUE Int // WT X Y XY
                                   One_recombinant UNIQUE Int UNIQUE Int // WT X
                                   Selected UNIQUE Int UNIQUE Int // X XY
                                   One_all UNIQUE Int UNIQUE Int // X ALL
                                   Recs_all UNIQUE Int UNIQUE Int UNIQUE Int UNIQUE Int // X Y ALL
                                   One_let UNIQUE Int UNIQUE Int // X ALL
                                   Tested UNIQUE Int UNIQUE Int // X H
                                   Selected_trans UNIQUE Int UNIQUE Int // X XY
                                   Backcross UNIQUE Int UNIQUE Int UNIQUE Int UNIQUE Int // WT X Y XY
                                   Back_one UNIQUE Int UNIQUE Int // WT X
                                   Sex_full UNIQUE Int UNIQUE Int UNIQUE Int UNIQUE Int // WT X Y XY
                                   Sex_one UNIQUE Int UNIQUE Int // WT X
                                   Sex_cis UNIQUE Int UNIQUE Int // X ALL
                                   Dom_one UNIQUE Int UNIQUE Int // WT nonWT
                                   Dom_selected UNIQUE Int UNIQUE Int // WT X
                                   Dom_semi UNIQUE Int UNIQUE Int // XD ALL
                                   Dom_let UNIQUE Int UNIQUE Int // WT ALL
                                   Direct UNIQUE Int UNIQUE Int // R T
                                   Complex_mixed UNIQUE Int UNIQUE Int // X ALL
               Calc    Calc_distance UNIQUE Float
                       Calc_lower_conf UNIQUE Float
                       Calc_upper_conf UNIQUE Float

?Pos_neg_data  Item_1 <tag2> <object> XREF Pos_neg_data
               Item_2 <tag2> <object> XREF Pos_neg_data
                       // the two items that are mapped
               Calculation UNIQUE Positive
                                   Negative
                       // the result

?Multi_pt_data Results A_non_B #Multi_counts
                       B_non_A #Multi_counts
                       Combined #Multi_counts

?Multi_counts  UNIQUE <tag2> <object> XREF Multi_point UNIQUE Int #Multi_counts

// end of file
```

Configuring And Making Genetic Maps And Their Views.

This will be in FIVE parts

1. changing between views
2. altering views
3. exploring the highlighting facility
4. making new views of existing maps
5. making completely new maps.

Map making is explained in more detail under USER_INTERFACE_CONFIGURATION, elsewhere in this book.

Personalising The Genetic Map Display

The genetic map is designed as a set of columns which can be rearranged, deleted, narrowed etc. Colours can be changed.

1). make new views of the genetic map

a) To have no rearrangements :

Select the views box (top left) with R-mouse and slide down) Views box menu window and select MAP-no_rearr. Map is automatically redrawn without them. To get back to normal close this views window, choose Views box with right mouse and select MAP-default.

b) to have just rearrangements :

Choose views button with Left-mouse, you get a new window for changing the genetic map display. The round buttons under column type can be toggled off by clicking on them. Click on Intervals (goes black), then remove Loci (and whatever else you like) and pick on the re-draw box.

c) intervals (rearrangements) can be drawn in other formats by moving another type of interval across from the left-hand side of the columns menu and moving it to where you want it on the right hand side. Ask for help with this.

d) The ordered gene set gives a much better genetic map.

to get the ordered gene set choose Views again, the views button within that and then the Map-ordered option. Re-draw. Left hand set is well ordered. If you zoom in you may get a clear linear sequence.

Map Tools : Highlighting

1. get a map of chromosome IV.

preserve it.

On the class window, select let* Loci.

Go back to the map and using the Highlight button's menu, Highlight selected loci in magenta.

then try the other choices on the menu.

for example, highlight all the deletions with 's' in their name.

Altering The Gmap Configuration And Making New Views.

With the minidatabase to which you have write access

1. get a map of chromosome IV.

Use the top left VIEWS button and Right mouse button to move to different views (i.e. with or without rearrangements etc.) These are commonly required views in the worm community and therefore supplied with the worm data.

2. With a map of Chromosome IV.

Using the same button but Left-hand mouse button, get <view control> menu.

Looking at right hand columns : Try toggling some of buttons on and off and re-drawing the map.

3. Again, using the same button, get <view control> menu.

Try moving columns up or down the list by selecting them and clicking on another arrow to move them to that arrow position. Redraw.

4. Use a different method of displaying intervals.

This involves getting a new column definition from the left hand side by highlighting it (say Interval_JTM) then clicking on an arrow on the right hand side to interpolate it.

Re_draw.

5. Make a map with just 3 different ways of displaying intervals

and a scale bar and locator and Marker_loci.

(I called mine 3-ints.)

6. Save this new map-type.

Try saving on view control, on map IV itself.

I could only see the view-type 3-ints after closing db and saving then re-opening.

7. Using the 3-interval map, configure the Interval_SRK column.

Click on green box below the column to get the configure window.

Next to Query for display, type FOLLOW rearrangement.

Find a query for the colouring of objects.

I have tried asking for presence of Locus mapped to this rearrangement (Locus_inside) and for a clone (Clone_inside). You could also try Deletion and Duplication as different colours.

(every time you add something to the map view, you should save it by overwriting the previous save or give a new name if relevant.)

8. Using the 3-interval map, try changing column widths.

Instead of invoking the view control menu, you can go straight to configure by clicking on the column name in green at the bottom of the map.

File : Gmap_teaching/3_ints.view.ace could be used.

9. try similar things with Map Sequence-IV.

adding columns, changing widths, changing queries and colours, and try saving some of these.

NONE OF THE ABOVE ADDS ANYTHING NEW TO THE MAP, IT MERELY CREATES A NEW VIEW WHICH CAN BE SAVED AND JUST RE-DRAWS EXISTING OBJECTS ATTACHED TO THE MAIN MAP OBJECTS.

Adding New Objects To An Existing Map

10. To add new objects to the map.

Look at the Map Sequence-IV text version.

Sequence-IV shows the clones used for sequencing and gives a quick visual progress report.

Its a very simple display to which you could add several other classes of objects.

10a. Add a column to the Sequence_IV map view.

Get the view control menu.

To add a piece of text, try adding Derived_tags to the active column list. Re-draw.

Nothing happens, so then configure the new column.

Fill in the Query (CLASS) at the top, also the Tag in the first <Tags to show> list and try various radio buttons on the right side for their effect. Re-draw.

(I used FOLLOW Clone and General_remark because the only objects contained in Map_sequence are clones and sequences. You could try Y_remark).

Save the view under a new name.

10b. Add another column with POINTS.

Here it gets more labour intensive.

The only things in the worm database with Point positions are Loci. So add a points column to the active list and then try configuring. Nothing happens however hard you try unless the Loci are listed under the Map Sequence-IV object AND they have positions on the Sequence map. So you need to get the text of the Sequence-IV map object and add some loci. They also need to have positions on the Sequence Map.

(file locseqmapIV.ace can be read in with this data.)

To get this to work, I dumped out the Loci resulting from the query :

>?Clone Map = IV Positive_locus

into the file rawdata/locseqmapIV.ace.

I added some data to each locus which has a position on the gMap :

Map Sequence-IV Position xyz.

Position xyz was estimated as the mid-position of the relevant clone. This was read back in and the names of these loci were added to the Map Sequence-IV object.

Try adding a background colour to some of the Loci.

(>FOLLOW Locus ; Alleles; OR > FOLLOW Locus ; C.elegansII_text.)

You may need to re-calculate the map to see things displayed then select the view which has points column added.

Making A New Map Object

11. to make a completely new map.

Its a good idea to look at the models used by the code to get an idea of what you have to specify in maps and views.

You have to make a map object and a view object, put some objects under contains in the map object and give them positions relative to the scale you choose for the map object.

I used clone (cosmid) B0303 with a notional length of 40000 DNA bases, centred at 20000. I used the fMap display to work out some positions for the predicted gene sequences which I have put as objects into the Contains section of the map object. I added Left and right end positions to the new map object within each sequence object.

To make a view object, you can use view-control on the map you make, i.e. add columns using view control : intervals, points and derived tags. intervals_srk would be suitable for the

sequences. Or you can create the text of a new view object using the update mechanism on a tree window of a newly named view object.

You should be able to make at least 1 new map with more than 1 view and more than 1 column on each view.

Files are available for you to take short cuts in Gmap_teaching/

b0303.ace

View.Map_cosmid.ace

View.match_cDNA.ace

B0303.map. additions.ace

To understand the map workings , they need to be read in this order.

Maps Within Maps

12. If you feel really confident about making new maps, you should be able to make a map of 1 predicted_gene sequence and its cDNAs which is contained within the map of Seq-B0303.

At the end of this exercise, you should be able to

1. alter an existing view,
2. Make and save a new view,
3. Add new objects to a map
4. Add new columns to a map
5. make a new map.
6.

gmap tutorial from sylvia martinelli, sanger centre. 250797.

revised dec 1997.

Fmap

1. Basic navigation (See previous)
2. DNA Analysis (See previous)
3. Changing the Fmap display : Column handling
4. Changing the Fmap display : Method alteration
5. Dumping out sequences (see previous)
6. Reading in new sequences (see previous)
7. Using GENEFINDER

Column handling

Use the top left hand Columns button WITH LEFT MOUSE to get the boxes at the base of the fmap display. Click again to remove the

Column boxes at the base. Right mouse gives a long list of columns which can be selected but goes off the screen and is no longer convenient to use.

Toggle columns on and off to get the display you want.

With 4_8c code(or later), the columns can be controlled near the top of the window, above each column with a drop down menu.

Method alteration

Open a method object and change some of the parameters set, save and re-draw the same fmap and look at the changes.

suggestions : colour of boxes, change right priority, toggle Bumpable and Score_by*.

Gene prediction and Genefinder

Also, I have provided a set of files in the directory cosmid_teaching which can be read into your acedb and used with genefinder.

C01G6_0_ver4.ace	DNA, this cosmid's sequence
	assembly tags
C01G6_1_ver4.ace	Sequence, subsequences features
	(exons appear and repeats)
C01G6_0_ver4.ace	Homologies, repeats etc.
	(ESTs appear)
WP:C01G6.prot.ace	wormpep sequences
methods_ver4.ace	a new set of methods you could use
ykforC01G6.ace	EST sequence objects

ykforC01G6.dna.ace EST DNA

Try reading them in one at a time using the **Edit menu : Add .ace files** and looking at the sequence object in fmap display, starting with the ver4.ace files. Blixem wont work for the ESTs unless you add the yk* files.

Pick a gene, zoom in and check the genefinder prediction made outside acedb.

Do you agree with it.?

If so, why?

If you agree, try another gene.

If you don't agree, re-predict the gene.

Select the ATG start site, gene splicing sites, coding frame for the last exon (effectively selecting a stop codon) and if necessary the Active Zone you want to work with. Justify your choice of sites.

When picking a slice site (drop-down menu on each feature), use the splice_colour_table for preferred colour to move towards. Use Blixem to look at the EST matches, do they confirm your splice sites.

When you selected everything you want, go to the Genefinder menu (right button drop-down and choose Selected-->Temp gene. Look at the result and decide whether to fix it or dispose of it and try again. Hints can also be found in
http://www.sanger.ac.uk/Projects/C_elegans/RUNGENE.shtml

In an ideal world, to predict a gene, you need to have blast homologies from searching a protein database for matches, DNA homologies to ESTs, experimentally determined cDNAs, positions of repeats (to avoid), gene structure prediction from an algorithm such as Genefinder, halfwise, postwise, FgeneH.

Table for splice site colours using genefinder

i \ o	Blue	Red	Green
Blue	-	L	R
Red	R	-	L
Green	L	R	-

L = leftwards even in a circle

R = rightwards even in a circle

7. Creating a new method

Design a new method to link together in the same fmap display a genome canonical sequence and an EMBL sequence for a specific Locus, the gene for ribosomal protein L13.

The EMBL sequence is available in cosmid_teaching/rpl13.dna.

The corresponding genome sequence is in cosmid_teaching/c32e8.dna.

If you run out of time or fail miserably, help is on hand in files :

method_emblmrna.ace, c32e8.ace, rpl13.ace.

To help you check the gene prediction, files c32e8*.ace can be used.

Reading EMBL Sequence Information Into Acedb.

Although this exercise demonstrates conversion of EMBL sequence information for incorporation into ACEDB, similar methods would be used for GenBank, or for literature references held in Medline, protein data in Swissprot etc.

BASIC METHOD using the command line on a unix machine :

```
> perl convert2ace < input.file > output.file
```

The EMBL DIRECTORY for the course has 3 input files and an executable file (program) written in PERL.

cecal1.embl = C.elegans calmodulin-like gene (calcium binding)

dmboss.embl = D.melanogaster gene for "bride of sevenless" protein

dmpcgene.embl = D.melanogaster gene "Pc" gene for polycomb protein

embl2ace = executable program for conversion of embl format into .ace worm database format

If you go into the embl directory, and providing that a perl interpreter is on your pathway, you can type the following on the command line interface :

```
% embl2ace < cecal1.embl > cecal1.ace
```

Grid Display

A very good explanation of the magic tags associated with Grids is given in

http://ars-genome.cornell.edu:80/cedocs/magic/grid.html

For these exercises, you may find it useful to add Grid and View to your layout.wrm file

in order to see them displayed on your class window.

Making A New GRID

In theory, his can be made in an .ace file or with the database open and using the Add/Alias/Rename choice on the Edit menu. However, if you are starting with the canonical models.wrm file, or the C.elegans models, only Clone objects can be put into grids. To do anything else, you need to make changes to the Grid_row model and to the models of all the classes that you want to associate with grid displays.

To display objects on a grid, you need to make a GRID object, a VIEW object, add any non Clone classes you want to display to Grid_row, and make changes to the models of the new classes you want to display.

```
?Grid Title ?Text
      Layout      Lines_at Int Int  // x, y spacing
                  Space_at Int Int  // x, y spacing
                  No_stagger
                  // default is with alternate lines staggered
                  A1_labelling
                  // label spaced blocks a-h down LHS, 1-12 across top
      Row Int #Grid_row
            // the items in the grid
      Virtual_row Int UNIQUE ?Grid XREF In_grid REPEAT
            // alternatively, a set of interleaved component grids
      View ?View
            // controls the colours and editing process

?Grid_row UNIQUE <tag2> UNIQUE <object> XREF Gridded REPEAT
                  // objects of a single class in one row - one tag only

?View Type UNIQUE Gmap
                  // display and columns below belong to this option
                  Grid Grid_map Int #View_tags
                        // Int is priority
                        // View_tags structure gives coloiurs and tags
                       Grid_edit_default UNIQUE Text
                        // Text is tag to set/unset on click editing
```

```
                        Grid_edit_menu Text
                         // List of tags to set/unset from menu

?View_tags  Colour #Colour
            Surround_colour #Colour //colour when surround
            Tag Text  //could be many tags
```

Worm Clone Grid

In the worm database, real YAC clones have positions on the grid which represents a polytene filter, to which cDNAs and other YACS have been hybridised in experiments. The YACs with grid positions have the Tag Gridded. The cDNAs which do or do not hybridise to various YACS (both positive and negative data are stored here) contain one of the tags listed here :

```
Positive Hybridizes_to
         Hybridizes_weak
Negative Does_not_hybridize_to
```

Those YACS which have hydridized to a test cDNA (also stored as a clone object here) the tag Positive_probe followed by the name of the cDNA clone.(similarly for negative results). There has to be a View object to control the grid for the POLY1 grid, it is GRID-1. The type Grid is set and under Grid_map, the relationship between the two types of clone is defined and given a colour code.

Typing into the Probe box on the POLY1 grid will give dark blue colour in any YAC box which hybridises well to the chosen probe, a pale blue colour is used for weak hybridisation.

As an exercise, try making a grid for yourself. It can display any class of object in rows and columns, which you choose. You can make any kind of relationship that you like between two classes of object.

General guide to making a Clone grid (the default in our database)

1. Choose Add/Alias/Rename under the Edit menu.
2. Choose Grid class and type a name in the <Name:> box.
3. Choose create and update.
4. Type in a title, this could be a longer explanation of the object's name.
5. Choose the number of columns if you wish to restrict them.
6. Choose the horizontal and vertical positioning of the small boxes.
7. Give a View name for the grid.
8. Start filling in the row data.

For each line you will have to give the row number, then keep putting in the Clone items

Guide to making a grid with a new kind of data e.g. Cells or Genes

1. Alter the models.wrm file so that your chosen class is listed under the Grid_row class.

for example :

```
?Grid_row UNIQUE Clone UNIQUE ?Clone XREF Gridded REPEAT
                 Mixed #Mixed_grid_row
                 Cell UNIQUE ?Cell XREF Gridded REPEAT
```

2. In the model for that class, add the tag Gridded.

```
?Cell Some_other_tags
      Gridded
```

3. Read models to get the new Grid_row recognised.

4. Follow the instructions above for Clone in 7 above, but substituting Cell for Clone.

The cell grid proposed here is a very simple one with no positive/negative relationships
So that a VIEW object is not needed.

If you don't have time to do this yourself, you can read in the file cell_grid.ace,
From course2001/grid_teaching.

Guide To Creating Relationships Between Other Classes. Example Cell And Gene

To create a Gene expression relationship between Gene and Cell, such as Expressed_in, NOT_expressed_in, it is necessary to alter the model for Cell and Gene and to define the tag for the relationship in the view object. Here, you will make full use of the Positive and negative tags to define the relationships on the grid.

For example

```
?Gene Expressed_in ?Cell XREF Expresses  ?Grid

?Cell Expresses ?Gene XREF Expressed_in ?Grid
```

You might also want to display definite negative results such as

```
?Gene NOT_expressed_in ?Cell XREF Does_not_express  ?Grid
?Cell Does_not_express ?Gene XREF NOT_expressed_in ?Grid
```

To make a View object, you can use either an .ace file or the interactive mode (Add/alias/delete). In the View object for your grid, your most important relationship should be given the highest

integer i.e. this takes precedence. Choose your colour and carry on for all the relationships you wish to define here. Fill in the menu choice and the default menu, save the view and try it.

If you find this very difficult to do, don't understand it or don't have time, there is a file in the course directory for you to read into the database (gene_expression_grid.ace). Although you will have to make the model changes by hand first (given in models.diff),

Then read models before you can add the data file.

```
//models.diff

?Locus  Positive Expressed_in ?Cell XREF Positive_expr ?Grid
        Negative NOT_expressed_in ?Cell XREF Negative_expr ?Grid

?Grid_row Cell UNIQUE ?Cell XREF Gridded REPEAT

?Cell   Gridded ?Grid
        Positive_expr ?Locus XREF Expressed_in ?Grid
        Negative_expr ?Locus XREF NOT_expressed_in ?Grid
```

The file : gene_expression_grid.ace contains a grid object, a view object and data added to both cells and genes to enable them to interact on the grid.

When you have a gene expression grid, try using it in the edit mode to add more gene interactions. There will be a pop-up menu on the individual boxes, so that you can choose the positive or negative relationship.

User_Interface_Configuration

Note This section contains all the things which can be altered by the curator which are not data alterations, *per se*, but presentation issues.

Class Or Main Window Display

At the top of the main window (the window which appears first on opening the database), is written the name ACEDB then the code version used to open the database and finally anything determined by the curator and written into the file **displays.wrm** in the **wspec** directory. Examples are :

```
ACEDB 4_7h WORM geneace

ACEDB 4_8b C. elegans 4/01/2000 WS8
```

ACEDB is hard coded to display which version of the code is being used to run the database (4_7h or 4_8b in the above example), in addition to the, curator definable, database title.

Underneath all the comments at the top of the **displays.wrm** file, the first specification line read by the ACEDB program is the title of the main window. Inside the quote marks, a curator can write anything.

```
_DDtMain -g TEXT_FIT -t "C. elegans 7/97"  -w .44 -height .23 -help acedb
```

The other file contributing to the look of the main window is **layout.wrm**, also in the **wspec** directory. You can open a text file using any editor, write anything in it, save it as **layout.wrm**. Then if you open the database, you should see what you have written in the file displayed on the main window. It should be displayed exactly as you wrote it.

Each curator could open the same database using their own **wspec** directory and display different class lists on this window as well as different database titles. This would be useful if different curators all had write access to the same database but all looked after different classes of data. Each person might want their own data classes listed on the left hand side on the window but every-one else's on the right hand, or may want other peoples classes hidden from view.

For instance, on my own database, I list the map data classes, but on the public database, these are hidden from the user and can be accessed only from the full class list on the main window.

Example of a **layout.wrm** file :

```
//layout.wrm

Locus       Sequence    Author
Clone       Map         Picture
```

With the database open, it is also possible to alter the main window class list. Click on the **Selection** box to get the editing window.

Altering The Default Size Of Windows.

Key to **displays.wrm** abbreviations :

-g Graphical display type

-t Title

-x Xposition <<<< to explain

-y Yposition <<<< ditto

-w (-width)

-h (-height)

If you did not like the default size of the FMAP window for instance, you could increase the size of height and width options. But, beware, the size of all FMAP windows will be larger in future.

-help is said to bring in the relevant part of the file from **wspec/help.wrm**. This file no longer exists but the display file is read by the program dealing with the new **whelp/** directory. (Check recent versions of **displays.wrm** handed out with ACEDB code.)

Altering Classes To Be Listed On Other Occasions.

In previous code versions, every class to be displayed on the main window had to have the -V (for visible) option rather than the -H (hidden) option. With the advent of **layout.wrm** the main window layout is specified purely by that. All user classes are now listed on other class lists, in **query** and **tablemaker** facilities and on the main window under **Other classes**.

for example.

Only on the Add/Delete/Rename/Alias window does this option (-V or -H) seem to take any effect now. Class Cell will appear on this window, class Cell_group will not.

```
_VCell -V
_VCell_group -H
```

Two features of **options.wrm** are still important the -D option and the -T option. For example :

```
_V Strain -V -T Genotype
```

If a locus object cross references to a strain object, when the Locus text window is open, the strain will be listed there by its Genotype not its strain number (object identifier in ACEDB). This is more human readable.

```
_V Locus -V -D Gmap
```

This -D option means that every time a Locus object is called, it will be displayed graphically on a **Gmap**, rather than as a text object. This only works if the locus has a map position on a **Gmap**, otherwise, the text appears. If the option is removed, all loci will appear first in text windows, on which one can summon the genetic map

by clicking on the Map object listed within.

Making Subclasses For Short Cuts

If your database contains some classes which are very big, it pays to divide them into sub-classes which can be brought up from the main window, or queried by the query language. For example, in the C. elegans database there are over 175,000 sequences. Calling for all sequences from the main window takes some time and it would take a very long time to scroll down this list. Similarly, performing a query which involves opening 175,000 objects then reading down each one to a particular Tag, also takes a long time, and uses a lot of memory and CPU. Consequently we have divided sequences into a few categories : Genome_sequence, cDNA_sequence, EMBL_sequence. There are a few thousand of the first one, nearly 140,000 of the second and 40,000 of the third. A query on just EMBL_sequence will be very quick. These subclass names can be displayed on the main window if defined in **layout.wrm** and in any list of classes where the parent object would normally appear, as defined in **options.wrm** e.g. **Query_builder**, **TableMaker**.

Preferences : Main Window And B-Tree Display

A curator, or even a straight user, can change the superficial look of the database by changing the default settings in the **Preferences** window. This can be accessed on the main window from the **Admin..** box. If these changes are saved, a file is written in the user's account as .acedbrc in the home directory. The user does not need to be listed in the **passwd.wrm** file. When changes are required, the file must be altered interactively through ACEDB not by editing this file.

The choices available in code 4_8b are

OLD_STYLE_MAIN _WINDOW	
HORIZONTAL_TREE	
ACTION_MENU_IN_TREE_DISPLAY	???
NO_MESSAGE_WHEN_DISPLAY_BLOCK	(unavailable)
TAG_COLOUR_IN_TREE_DISPLAY	

Code 4_7h also had Show_aliases and Show_empty_pbjects. (I think the latter is subsumed into 4_8 code. <<<<

The OLD_STYLE_MAIN _WINDOW has no buttons but every choice of action is listed on the drop-down background menu including Preferences which you might need in order to return to the original preferences.

HORIZONTAL_TREE on/off button changes the display from a widely spaced object to one with the subtags just inset a little way from the main ones and underneath them.

TAG_COLOUR_IN_TREE_DISPLAY allows a user to choose the colour of the tags from a list of 20 or so colours.

If a user/curator makes a default preferences file, it will be used for all the ACEDB databases in that account when running the databases with **xace**.

An administrator can set up default preferences by copying an acedbrc file to the home_acedb_database_dir/ wspec/acedbrc.

*SESSION CONTROL to be written

GMAP VIEWS

Creating And Saving New Views

The basic pattern behind the Gmap display is based on columns. Each column has a prescribed width, a given position from left to right and contains pre-defined objects. This prescription is determined by the curator. Each column is independent of every other column. Although a view object has a predetermined set of columns, not all have to be displayed as they can be toggled on and off. To be displayed on the Gmap, most objects have to be overtly given positions on that map, although a few objects are hard-wired into ACEDB code. The map object (Class ?Map) itself needs to “know” about all the objects that it will be asked to display (such as Loci, clones, chromosomes). They are listed within the Map object. It also needs to “know” how to display them through a view. The map display is made using a View object (Class ?View) which is connected to that map object.

Map IV

Display etc.

Main_Marker etc.

Contains Locus ---> 240

Rearrangement -----> 63

Contig ctg291

These map views can be made by a curator, or even a user, although a user will not be able to save them for future use within the database. A user could dump out the map view as a text file for future reference. Making a view is best done interactively, although it is perfectly possible to create a new view object using an ace file or using the add/alias/rename menu.

What can a mere user do ? They can use the **View control** menu to turn on/off columns then re-draw the map, or add new columns and configure them (see User_Guide section ???). Users cannot save views, so their preferences will be lost on closing the map. A curator with write access can do all this and save the view and change it to the default view.

(I need screen dumps for this part).

It would be better to open a map of the subtype Gmap (genetic) before reading this section. If you are using the worm database there are two types of Gmap,- the chromosome maps with Loci and Deletions etc. displayed and the Sequence_maps with sequenced clones displayed. Pepmap is based on the same code and looks very similar.

You can use the top left **VIEWS** button and Right mouse button to move to different views (on the worm chromosome maps, there are views with or without rearrangements etc. These are commonly required views in the worm community and therefore supplied with the worm data.). Any of these views can be altered using the same **VIEWS** button with Left-hand mouse, to get the **View control** menu.

The **View control** menu is divided into two halves vertically. Looking at right hand columns : there is a list of those columns already used in that view. Dark centres to the buttons indicate that these columns are displayed currently. You can try toggling some of buttons on and off then re-drawing the map. Another simple exercise : try moving columns up or down the list by selecting them with one click (they become highlighted) and then clicking on another arrow to move them to that arrow position. Redrawing the map will give the new view.

To add new columns to the view, you need to go to the left hand side of the **View control** window. The simplest exercise involves using a different method of displaying intervals. Intervals are any object which has a length defined by the position of its left-hand end and its right-hand end on that map. Familiar examples are clones, sequences and genetic deletions.

Configuring Interval Columns

There are 3 kinds of interval display provided by the ACEDB code : Interval_RD, Interval_SRK and Interval_JTM. The hard-coded default is Interval_RD. If your interval column is of this type, you may wish to change to Interval_JTM. This involves getting a new column definition, from the

left hand side of the view definition window, by highlighting it with a click, then clicking on an arrow on the right hand side to interpolate it into the column list. Even if you redraw the map, nothing will happen at this stage because the column does not contain anything,(it has not been configured). Click (right mouse) on the new column type Interval_JTM so that a drop-down menu appears. Select **Configure column**. By default, this column will show the contents of the previous Intervals column. To alter it, use the configure window. For example, it the previous column is worm genetic Rearrangements called *Df* and *Dp* (deletions or deficiencies and duplications), you could just display one type in the new column by typing next to **Query for display** : FOLLOW Rearrangement *Dp*, then clicking on **Apply.** The map will reappear in an altered form.

To restrict the width of a column, just type a number into the box next to **Column width restriction.** Remember to press return on this number, then click on **Apply** or **OK.**

Toggling the **Display names** button is self-explanatory. Other features will be described later. Toggling the **Display interval** button will remove the intervals leaving their just names in the column**.**

Selecting Interval_SRK as a new column type adds a new dimension, that of colour to the intervals. They are drawn as boxes rather than lines and the configure menu includes the facility to run one or more queries on the boxes and choose a colour for display. For instance, you could type in the query for a worm Rearrangement and choose a colour from the pop-up menu open the colour box, to the right of the window(see example). *s* means any Rearrangement whose name contains an s.

Query for Display : FOLLOW Rearrangement *s*

Queries

Locus_inside	MAGENTA
NOT Reference	LIGHT BLUE

To configure the Interval_SRK column.

You can also go straight to configure by clicking on the column name in green at the bottom of the map. Warning, you may need to open the model for the class first in order to design a sensible query. Locus_inside refers to the presence of a Locus mapped to this rearrangement. Reference means that a Paper object is connected to this object. You could also try Deletions and Duplications as different colours.

Saving Issue !

Right mouse on the background will give a menu which includes **Save view.** Selecting this provokes the display of a window with **Please Reply** at the top, and a yellow box in which to give the name of this new view**.** Closing the map and re-opening it should add the new view to the list under the **Views** box. Every time you add something to the map view, you should save it by overwriting the previous save or give a new name to the view, if relevant.

Changing Column Name

On the **Views control** window, on the right hand side, click once on a column name to make it turn yellow then type in the yellow box.

Changing Colours For Related Data

(more introduction needed here)

The four coloured boxes on Interval and Point column menus are :

Positive data, no error
Positive data, error in coordinates
Negative data, no error
Negative data, error in coordinates

Each can be allocated any colour you choose. The worm database has the default colours, in order, green, dark green, mid-blue, dark blue. Clicking on a locus in the default map view may highlight some intervals in any of these 4 colours. Clicking on an interval may light up loci in any of these colours. If Locus XX has been used in a cross with Deletion xxDF99 and found to be deleted by this deficiency, and in another cross with xxDF55, the locus was not deleted, then highlighting Locus XX should turn xxDF99 green (positive result) and xxDF55 to mid blue. If xxDF99 had been crossed with Locus YY, mapping to the same point as xxDF99 and found NOT to delete this other locus, highlighting xxDF99 would turn locus XX green and Locus YY dark green, for conflicting data. In order for this to work, there are tags in the Pos_neg_data class called Positive and Negative. A deficiency deleting a locus is given a Positive attribute when reporting the result in a Pos_neg-datga object, not deleting it is a Negative attribute.

Note none of the above adds anything new to the map, it merely creates a new view which can be saved and just re-draws existing objects attached to the main map objects.

Adding New Objects To An Existing Map

presumably needs write access and so a user would not be able to do this.

To add new types of object to the map, you have to work on the Map object, the view object and the mappable objects themselves. Here it is a good idea to look at the text version of a map such as Map Sequence-IV or chromosome Map IV in the worm database. The tag **Contains** is the key to this manoeuvre. Whatever you want to display on the map has to be listed under contains.

```
Map IV
Display   etc.
Main_Marker   etc.
Contains Locus ---> 240
                Rearrangement -----> 63
                Contig  ctg291
```

You could decide, for instance. to add a column for Predicted_genes. These are DNA sequences from sequenced cosmids, with exons and introns predicted by DNA analysis, Protein homology and DNA homology tests. They have coordinates which allow them to be drawn on the FMAP but not on the Gmap.

In the worm database, Predicted_genes have been defined as a subclass of Class Sequence and so the name Predicted_Gene can be used wherever the word Sequence would normally appear. Since the model of Class Map in the worm database, specifies that the following Classes of object can be placed here : Locus, Rearrangement, Contig, Clone, Sequence, Allele, Map, you can add some sequences, without altering the models. If you add the names of six sequences to the Map object using the interactive **Update** option, the Map object would then appear so, (putting the subclass Predicted-gene under Sequence) :

```
Map IV
Display   etc.
Main_Marker   etc.
Contains Locus ---> 240
                Rearrangement -----> 63
                Contig  ctg291
                Sequence  ---->6
```

Which would expand to:

```
Map IV
Display   etc.
Main_Marker   etc.
Contains Locus ---> 240
                Rearrangement -----> 63
                Contig  ctg291
                Sequence  F20G6.3
                                ZK691.4
                                Y70H8A.15
                                C06G9.7
                                AH6.1
                                B0303.3
```

Next you have to decide whether you want these sequences to appear as point object or bar objects (intervals). For simplicity, if you choose point object you only have to give the Predicted_genes one position in the object. Although in real life, these would have real and quite specific positions on the Maps, I am assuming for now that they can be given arbitrary positions.

Open each of the 6 Predicted_genes in turn, choose **Update** and move to the tag Map, put the name of the map object here (IV in my example), then move along to Position and click once to open up the interpolated model for Map position. Click once on **Float** to turn this yellow and type in a position which is within the extent of your map.

Note if there is no tag for Map in the Class you wish to work with, you need to alter your models before going any further.

When you have several objects containing map positions, you can open the Map (IV in my example) and add a points column to the active list of columns on the left hand side of the **View control** window. and then try configuring. You could give a name to this column as well. When configuring the new column, you could type FOLLOW Sequence, click on **Apply** and then from

the map's background menu select **Recalculate.** The sequence will appear on the map and the view on the View list.

Other Right Hand Columns Available And Not Already Discussed : Marker_Points, Marker_Intervals, Spacer, RH_Data And Submaps.

Another way of adding a column to a map view is via the **Derived_tags** route. If you get the **view control** menu and on the right click on **Derived_tags**, having already clicked on an arrow on the left, you will get a column on the right-hand side called **Derived_tags**. Click on Re-draw.

Nothing happens until you configure the new column. On the configure menu : fill in the **Query for display** at the top, for example FOLLOW Sequence. Also fill in one of the **Tags to show** lines for example for class Sequence, use DB_remark. Click on **Apply** and see the remarks written on the map. There are various radio buttons on the right side to try **(and for me to describe).** Re-draw the map but don't forget to save the view under a new name.

Try adding a background colour to some of the Loci.

```
>FOLLOW Locus ; Alleles;
```

Or

```
> FOLLOW Locus ; C.elegansII_text.)
```

You may need to re-calculate the map to see things displayed then select the view which has points column added.

Advanced Queries

Please use 20ace for these queries.

1. How many STSs have been originated by Deloukas P ?

2. How many Sequence objects have no DNA sequence associated ?

3. How many STSs have the Hybridisation_screen flags PAC and Passed ?

4. How many ESTs (STS Sequence_type 3'cDNAs) are on the 1998 RH map ?

5. How many Sequences are longer than 3000 bp ?

.... another way ??

6. Using keysets, how many new STSs were placed on the RHmap (03/11/98)
when compared to the one dated 01/12/97 ?

7. How many STSs are positioned on the 98 transcript map in the region
180 - 200 cR ?

8. How many STSs, on the map SANGER_rhmap_01_12_97, detect a cluster ?

9. Find all STSs screened with PAC poly grids after 1998-09-20
(hint: DateType ... and this one is HARD, check your results carefully).

10. Construct a tablemaker query to give:

STS name
Parent sequence name
the two oligos
PCR buffer protocol
STS length
Parent Sequence

... don't forget to title each column and adjust the spacing.

11. Create a map view for the Chr20 map dated 08.04.97 to show the framework markers (STS Framework_marker) in green boxes.

12. Change the above view to just show the framework_marker(s).

ANSWER SHEET FOR QUERIES WITHOUT ANSWERS.

FOR MANY OF THESE NEED COMMUNAL DATABASE OR PUBLIC WORM DATABASE.

(impossible without looking at the models.)

1. How many predicted genes (Subsequences) have matching cDNA.

>?Sequence Subsequence === 4681

(to be correct need AND NEXT after Subsequence since some objects had Tag but no info !!!)

(equivalent in worm database to Genomic_Sequence subclass (plus some EMBL)

follow subsequence 28547

Matching_cDNA 7304

2. How many predicted genes are equivalent to Mendelian Loci

>?Sequence Subsequence; follow subsequence; Locus 1434;
follow Locus 910 ; Allele 332
>?Predicted_gene Locus 783 (no EMBL !)
>?Sequence *.* Locus (will include EMBL.)

3. How many clones have been finished by the Sanger centre finishing team. (HINXTIN)

>?Clone Finished ; follow Sequence; From_Laboratory = *HX*

2991; 1344;

4. How many contigs make up Chromosome IV?

>?MAP IS IV; FOLLOW Contig;

5. How may sequences have DNA ?

>?Sequence DNA AND NEXT

6. How many sequences have DNA longer than 300 base pairs ?

>?Sequence DNA AND NEXT AND NEXT > 300

FOR THIS ONE NEED TO MAKE SURE THAT THE CELLS ARE IN DATABASE FIRST OR USE BIG DATABASE.

7. How many cells have great-grandchildren?

You can do this from top to bottom or bottom to top.

first how many cells have grandchildren ?
>?Cell Daughter 1110 == cells with daughters
follow Daughter 2215 == these daughter cells
Daughter 1095 == daughters with daughters
follow Parent 645 == the parents with grandchildren

then how many cells have great-grandchildren ?
>?Cell Daughter 1110 == cells with daughters
follow Daughter 2215 == these daughter cells
Daughter 1095 == daughters with daughters
follow Daughter 2189 == the grand daughters
Daughter 1077 == granddaughters with daughters
follow Parent 643 == the original daughters
follow Parent 403 == the original parents

NOT RELEVANT IF YOU ARE USING THE PUBLIC DATABASE

8. In the worm database, Gene_classes are specifically made to hold Phenotypic descriptions. So why are there so few in the mini-database ?