M-CHIPS Storage Concept

Kurt Fellenberg

24th November 2001

Abstract

The M-CHIPS Database concept allows for data analysis of all of its components including the experimental annotations. It accounts for rapid increase in amount of hybridisation data, experimental descriptions becoming more detailed and new kinds of experiments we expect for the future. This report aims to present our work on storing microarray data from the more general aspects down to the practical details.

Definitions

Relations, attributes and tuples

A relational database consists of

- relations, also called classes or tables. Such a table relates between
- attributes also referred to as data fields or columns of such a table and may contain an arbitrary number of
- tuples, also called records or datasets, which are represented as the rows of a table.

Query

A database query retrieves a set of tuples from a specified relation which meet a certain condition (also specified in the query).

Indices

The purpose of an index is to allow rapid access to specified values within a relation. Without an index, the server process executing a query has to read from the beginning of the table to the end, looking for relevant tuples. An index is computed on a relation for one (or several) of its attributes, ordering the comprised values and storing pointers into the relation in an appropriate data structure (e.g. a b-tree or hash table). A query is directed to the matching rows by the index, thus performing much faster than without.

Inheritance

The PostgreSQL database management system (DBMS) allows for construction of table hierarchies by inheritance. Let the table B be created inheriting from table A, then

- B inherits all attributes of A and
- its tuples are accessible by querying the parental table A
- by anyone having read permission on A regardless of permissions on B.

Data integrity

Suppose a valid alteration of the data, defined by a block of sequentially permformed operations (such a block is called a 'transaction') breaks down after doing half of the work. Maybe a table has been deleted but remains existant in the table administrating system catalogue of the database system. Or the task was to add $500 to everyone's salary in a table containing employees and now it is unknown which row was updated and which not. In both cases data integrity (database consistency) is violated.

Contents

1 General Considerations concerning Data Storage 5

1.1 Storing transcription intensities 5

1.1.1 What to store 5

1.1.2 How to store it 5

1.2 Storing gene annotations 6

1.2.1 What to store 6

1.2.2 How to store it 6

1.3 Storing experimental annotations 6

1.3.1 What to store 6

1.3.2 How to store it 7

1.3.2.1 Flexibility 7

1.3.2.2 Performance 7

1.3.2.3 Analysis aspects 7

1.3.2.4 M-CHIPS storage 8

2 Realization concepts 10

2.1 Safety considerations 10

2.1.1 Global accidents 10

2.1.1.1 Transactions 10

2.1.1.2 Global backups 11

2.1.2 Private errors 11

2.1.3 Unauthorized access 11

2.2 Performance considerations . . . 12
2.2.1 Minimizing query space . . . 12
2.2.2 Separate or block-wise storage . . . 12
2.3 Flexibility considerations . . . 13
2.4 Considerations related with Analysis . . . 14

3 Implementation **15**
3.1 Databases . . . 15
3.2 Microarray Families . . . 23
3.2.1 Gene annotations . . . 23
3.2.2 Administration of the comprised multiconditional experiments . . . 25
3.3 Multiconditional Experiments . . . 26
3.3.1 Experimental annotations constant within the multiconditional experiment . . . 26
3.3.2 Administration of the comprised condiditions . . . 28
3.4 Experimental Conditions . . . 29
3.4.1 Administration of the comprised hybridizations and measurements . . . 29
3.4.2 Experimental annotations dependent on the condition . 29
3.5 Hybridisations . . . 31
3.5.1 Hybridisation intensities . . . 31
3.5.2 Hybridisation intensities, 'solidified' . . . 32
3.5.3 Measurement-dependent experimental annotations . . . 33

Chapter 1

General Considerations concerning Data Storage

1.1 Storing transcription intensities

1.1.1 What to store

Considering the ongoing development of analysis tools for array data, it would be unwise to store any processed form of the original data because it will be outdated when their calculation methods change. In order to perform our calculations on the fly on transcription data as raw as feasable, we decided to store the signal intensities derived from a hybridisation image by an imaging software. All of the recent releases of these software packages require human interaction for verification and adaption of a semi-automatically performed spot recognition, impeding on-the-fly calculations on the image data themselves.

1.1.2 How to store it

Storing intensity information appears to be easy. A hybridisation yields a huge amount of uniform data comprising, in our case, two intensities and two background values per gene or EST (being spotted in duplicate). Performance considerations would suggest hybridisation-wise storage in tab delimited files or array tuples in a database, dispensing with selective retrieval of particular values but being fast accessing whole hybridisations. Specified subsets of spots are not easily accessible in a hybridisation data file nor in an array. However it should be possible to selectively retrieve intensities above a certain threshold or within a specified interval, thus it is necessary to store the values for every gene/EST as separate tuples in a database relation. In this form indices can be calculated to perform fast score-dependent queries utilizing the database capability of b-tree search. When in future hybridisation databases will be too large to be loaded into computer memory, it will become essential to perform tuple selections as well as simple calculations on the database level before loading compressed results into the memory for visualization.

1.2 Storing gene annotations

1.2.1 What to store

Gene annotations may consist of clone numbers, accession numbers and different kinds of entries describing the spotted sequence or the encoded protein like chromosomal location, enzyme categorization number or protein structure. Here we only include identifiers serving as a key to connect to databases containing gene information, short variable length free text descriptions of the protein and its functional category and the spot location. Moreover it turned out to be necessary to explicitly keep control of the array the spot is located on, provided that the spotset comprises more than one array and each of them has been hybridized separately.

1.2.2 How to store it

Because complex sequence annotations or enzyme properties are found in linked gene databases, the gene annotations may be stored in only one relation containing attributes for the above values, and every spotted element (gene or EST) can be described by one tuple.

1.3 Storing experimental annotations

1.3.1 What to store

Experimental annotations comprise the description of environmental conditions, genotype, patient data, information about surgery, type of tissue (incl.

estimated degree of contamination by other celltypes), the sampling method and annotations related to hybridisation protocol, properties of the individual array or imaging process, to give some examples. They fall into the two realms of

1. Organism-specific annotations resembling the need of the specific research area such as e.g. 'transgene' and 'growth phase' for yeast or 'tumor type' and 'metastasis location' for human biopsies.
2. Common annotations that are useful for all fields of interest. These technique-related properties like array characteristics, description of labelling, hybridisation and washing conditions or detection of the signals are annotated by all the users.

1.3.2 How to store it

1.3.2.1 Flexibility

Experimental annotations are set up by the biologists working in the field. They tend to grow with every new type of experiment performed. To account for this, an implementation of any concept will be useless if it does not enable easy and quick addition of new annotations or the completion of values for already defined annotations without altering the database scheme and the analysis algorithms. If an annotating experimenter finds something he or she forgot to define before uploading an experiment, the database should have the flexibility to incorporate a new annotation or value within a minute.

1.3.2.2 Performance

Gene and experimental annotations taken together sum up to less than 0.35% storage space of the yeast database (May 2000). Since the share of data entered directly by human beings may in any case have a size far too small to be relevant for query performance, flexibility remains the only time saving aspect related to experimental annotations.

1.3.2.3 Analysis aspects

For the conceptualization of structures for data storage one might prefer formats supporting a wider range of analytical access to the data than others.

Let the experimental annotations, though ordered into various categories and subcategories, be text fields containing free text description of the annotation value, e.g. the yeast specific annotation

- growth phase - value: 'exponential'.

From people querying sequence databases in a high throughput manner, one can learn that there are severe problems like misspelling, different words having the same meaning, various types of abbreviations, making it hard to analyse the contents of a text field for a high number of datasets. On the other hand one would expect the number of tuples (hybridisations / multi-conditional experiments) of a public expression database, once established, to grow quite fast. People might cluster these tuples by the expression behavior of a set of genes and would want to know which growth conditions, experimental settings, genotypes or environmental conditions of the organism corresponded to a particular cluster. In other words: Which properties are common for hybridizations that share similar expression patterns? This question cannot be answered by visual inspection alone when looking on hundreds of hybridizations with huge numbers of sample properties. Sample descriptions are favourable that enable inclusion of these descriptions into the process of algorithmical analysis. To make them accessible to statistical analysis, the values of an experimental annotation should be directly comparable among the datasets. If we for example let the above annnotation 'growth phase' be an enumeration type variable comprising the defined values 'exponential', 'stationary' and 'pseudo-hyphal', the occurence of the value 'exponential' can be counted within the cluster and compared with it's overall occurence to determine if it is characteristic (either over- or under-represented) for the cluster. Prerequisite is that the annotation values are enumerable. Apart from enumeration type annotations already mentioned, floating point numbers can be made enumerable by mapping them to a set of bins, e.g in a way that each bin covers an equally spaced range of values or in another manner that seems suitable for the particular annotation in terms of biological relevance.

1.3.2.4 M-CHIPS storage

Our implementation works now for 33 yeast specific, 70 arabidopsis specific, 54 human tumor specific, 41 trypanosoma specific, and 76 common (technical) highly categorized experimental annotations. They were set up by biologists working in these fields enabling statistical analysis of the descriptions of nearly 700 hybridisations stored in a PostgreSQL database. The following more practical ascpects deal with the realization of such a database for a multiuser setting.

Chapter 2

Realization concepts

M-CHIPS provides a storage concept for unified analytical access to microarray experiments from different fields of research, an instance of which is a field-specific (organism-specific) database. Although these databases adopt different ontologies for experiment annotation, they can be accessed by the very same analysis algorithms. They are designed to be used by the people who generate the data. To meet the requirements of these users, they have to allow for multiuser access including safe management of simultaneus write access, short waiting periods and privacy (protection against unauthorized access).

2.1 Safety considerations

2.1.1 Global accidents

2.1.1.1 Transactions

DBMSs capable of the administration of more than one version of a database at the same time (like Oracle or PostgreSQL) protect integrity of the stored data by transactions. Transactions give databases an all-or-nothing capability when making modifications. A transaction can comprise one or multiple queries with every of the performed changes becoming valid upon successful execution of the whole transaction and none of them in case of an error. At the same time all other users are insulated from seeing the partially committed transaction until the very moment of commitment, preventing database consistency from being damaged by simultaneous write access. Although transaction-based database management slows down access performance, we recommend to use a transaction based DBMS.

2.1.1.2 Global backups

Although the choice of a transaction-based DBMS ensures a great amount of safety for the data, there is no way to guarantee absolute secureness. In case of a disk headcrash or failure in the server's power supply while updating important system catalogues it may well be that the integrity of all the databases managed by the server is destroyed at the same time. In such a case we will restore the status of the last night for the whole database system from tape backup.

2.1.2 Private errors

In case of accidently deleting hybridisations from a single database it would be inappropriate to reset the whole system to the state of the night before. To be prepared for such a case, SQL dumps are performed separately for each database overnight. They consist of SQL queries that can be used to restore data subsets from a whole database down to a single tuple of a particular table.

2.1.3 Unauthorized access

To ensure that data (which may be unpublished) cannot be altered nor read by unauthorized individuals, update and/or read permissions can be granted on any database table to a particular user. Granting such permissions to user groups rather than separately to each user is a common procedure to circumvent the necessity of changing permissions for each database table upon registration of a new user. In our implementation nearly all the relations inherit from few parental tables and are accessed via their parental table only. Permission inheritance enables the administrator to quickly grant e.g. read access to a new user by changing permissions for a few parental tables in place of dealing with many tables or user groups. However, the main reason for access via parental tables is to enable pooling of tuples from hybridisation tables into large blocks without syntax alteration of accessing queries (which will be described below 2.2.2).

2.2 Performance considerations

Since the overall extent of data referring to gene descriptions and experimental annotations is minimal (see 1.3.2.2), performance considerations are related only to hybridisation intensities.

2.2.1 Minimizing query space

It is already quite efficient to divide the entirety of spots into appropriate subsets related to the type of queries that are performed. Most of the analysis queries target genes rather than empty or control spots, so we recommend to store at least the genes separately from the rest. In our implementation the spots are kept in tables belonging to (and inheriting from) 5 different parental tables comprising

- genes (genes / ESTs - incl. housekeeping)
- empty spots (no DNA has been spotted)
- heterologous DNA (e.g. guide spots)
- heterologous DNA with known concentration (external control spots for 'spiking', i.e. assaying standard RNA aliquots added before the labelling step)
- reference spots (reserved for a novel category of control spots).

2.2.2 Separate or block-wise storage

As already mentioned in 1.1.2, fast querying of tuples is mediated by indices. If the above categories would contain the hybridisations stored so far as one big block table per category, adding a new hybridisation would be quite slow because of the time necessary for recomputing the indices. Because of this, every new hybridisation is inserted as 5 new separate relations, computing indices only for the new tuples.

However querying for certain values is slowed down by increasing number of separate tables, because there is no global index guiding the search immediately to the one containing the tuple. This structure, while enabling high performance for write / delete operations impedes a fast read access. In order to optimize both for writing and reading operations, we

- write / delete hybridisations as separate tables, but
- read from large blocks,

which are produced by over-night jobs that join those tables (hybridizations) that are not to be altered or deleted any more. Thus, computing of large indices is performed at times of low traffic (as an investment in query performance).

While storing hybridizations into blocks includes alteration of the database structure (decreasing the number of tables), it remains totally insulated from and invisible to the accessing software (algorithmic layer): Since every access to the intensity tables is directed via one of the five parental tables listed in 2.2.1, query syntax does not change with the 'assembly' of a set of tables into one block: This block will be a child of a specific parental table as have been the collected tables, summarized within the new block.

2.3 Flexibility considerations

To meet the requirements described in 1.3.2.1, the categorization of experimental annotations should be kept in definition tables rather than mapped to database structure itself. In our concept annotations along with their defined values are stored in a definition table. Each annotation has a unique identification number. They are stored as a linked list including an attribute pointing to the ID of the annotation next in sequence. The ID serves as a key for querying the annotations, the defined sequence allows for a clear list structure facilitating the annotation process. The annotations are structured by a set of headings and subheadings with an arbitrary nesting depth, which are stored in a second table. The linked list structure enables adding of a new annotation at an arbitrary position by linking of the desired predecessor to a new element that points to the ID of the element following in the list. In a similar manner the whole set of defined values is numbered sequentially to enable rapid queries and stored by a linked list in the same table as the annotations.

To prepare for the administration of experiments related to a new field of research, it is sufficient to generate an empty database with definition tables containing the up-to-date list of common annotations along with a new second half of both annotation definition and heading table containing the 'organism-specific' annotations for the new field of experiments. A growing number of already assembled definition lists facilitate to compile new ones by serving as templates for the description of similar experimental procedures.

2.4 Considerations related with Analysis

As described in 1.3.2.3, the annotation values should be categorized down to an enumerable level, either directly by creating an enumeration type annotation, or by storing a floating point number. These numbers are stored along with a unit if this is required for a unique meaning/message of the value. They don't necessarily have to be non-integer. Discretizing numbers will be reasonable in cases where

- similar values are expected to have the very same meaning in terms of their biological impact and
- the probability of those equivalent values to match the very same number is low because of measurement errors.

Chapter 3

Implementation

This chapter will deal with how to suit the action to the world proposing an implementation of the above concepts. I will start by describing database specific tables before showing schemes of the tables related to multiconditional experiments and hybridizations. The entirety of microarray data can be divided into the following sections. Each section is a subset of the one ahead of it in terms of hybridisation intensities, but comes along with a unique set of annotations (Tab. 3.1).

3.1 Databases

Our microarray databases are administered by a PostgreSQL database server process running on a SUN E450. Data are uploaded, annotated and analyzed by users working in different fields of research using samples from different organisms. A separate database is created for each organism / field and endowed with particular definitions of experimental annotations appropriate for the attended sort of sample. Figure 3.1 shows the definition of experimental annotations in relation to other major parts and gives a rough overview of a database. A detailed scheme is given at `http://www.dkfz-heidelberg.de/tbi/services/mchips/scheme2.pdf`. Apart from the experiment annotation definition-tables (red boxes on top), the latter scheme shows two more relations occuring only once per database (on top in green 'DATABASE MANAGEMENT'-box). The first stores archive flags reporting any write access to either tables or BLOBSs (binary large objects) for an overnight job producing a new backup of the database. It also holds

Table 3.1: Database sections

Section	Intensity Data	Annotations
database	containing all data derived from / related to one particular field of research (organism)	definition of valid experiment annotations along with a set of valid values for each of these annotations
microarray family	data obtained from one array type comprising a defined set of genes/ESTs in a particular spotting scheme	gene annotations (spot location, brief description and keys relating to external databases)
multiconditional experiment	set of measurements comprising two or more experimental conditions incl. one 'control' condition	experiment annotations common throughout the experiment (unchanged in all of the conditions)
experimental condition	consists of one or more hybridizations repeatedly performed under the very same conditions	condition dependent experimental annotations (e.g. the timepoints in a timecourse)
measurement (image)	one image, i.e. one channel in case of multichannel data - consists of genes / ESTs, empty spots and different kinds of reference spots, all of which are spotted in duplicate (referred to as 'primary' and 'secondary' spots)	measurement dependent experiment annotations (e.g. labelling efficiency, individual array no., number of previously performed hybridisations on the individual array

Figure 3.1: Overview scheme

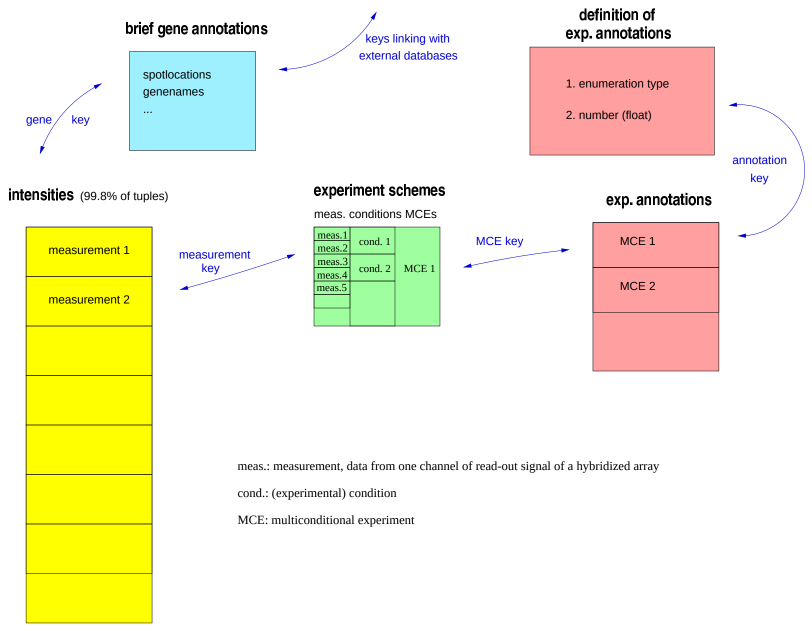


Table 3.2: Database-related information (table structure)

```
                          Table    = archive
+----------------------------------+----------------------------------+-------+
|              Field               |               Type               | Length|
+----------------------------------+----------------------------------+-------+
| tablesflag                       | bool                             |     1 |
| blobsflag                        | bool                             |     1 |
| structure_version                | int4                             |     4 |
| headingsnestdepth                | int4                             |     4 |
+----------------------------------+----------------------------------+-------+

                          Table    = master
+----------------------------------+----------------------------------+-------+
|              Field               |               Type               | Length|
+----------------------------------+----------------------------------+-------+
| family                           | text                             |   var |
+----------------------------------+----------------------------------+-------+
```

the database's structure version and nesting depth of its annotation hierarchy. The second is a register of the microarray families within the database (Tab. 3.2).

The definition of experimental annotations consists of a table listing the annotations along with enumeration type values, a table containing the 'annotation headings' which provide a hierarchy of topics categorizing the actual annotations, and one recording those annotations usually being measurement dependent (Tab. 3.3).

The annotation headings show a nesting depth of 3 heading levels. Here the fourth level of the hierarchy comprises the annotations themselves, the fifth their annotation values. For the annotation of an experiment the nested headings and annotations are compiled into one HTML form by a web interface. To accelerate the recursive CGI script, starting and end points of blocks consisting of elements to be sequentially listed in the form (but not necessarily being sequentially numbered in the linked list), are precompiled into arrays and recorded after updating the definition tables (Tab. 3.4). To give an expample how experiment annotation definitions may look like in practice, Tab. 3.5lists the first part of common annotations. The common annotations are used commonly for yeast, arabidopsis and human cancer biopsies to describe the more technical part of the experiment. The HTML output compiled from the table contents is shown in Fig. 3.2. The complete set of common annotations can be found in the first part of each annotation definition list on our web site[1], e.g. in the yeast list[2]. The actual experiment annotations which are entered via similar HTML forms are stored elsewhere

[1]http://www.dkfz.de/tbi/services/mchips

[2]http://www.dkfz.de/tbi/services/mchips.yeast.html, http://www.dkfz.de/tbi/services/mchips.yeast.txt

Table 3.3: Definition of experimental annotations (table structure)

```
                          Table    = annotations
+----------------------------------+----------------------------------+-------+
|              Field               |              Type                | Length|
+----------------------------------+----------------------------------+-------+
| lastheadingno                    | int4                             |     4 |
| ano                              | int4                             |     4 |
| nextano                          | int4                             |     4 |
| annotation                       | text                             |   var |
| vno                              | int4                             |     4 |
| nextvno                          | int4                             |     4 |
| value                            | text                             |   var |
+----------------------------------+----------------------------------+-------+

                       Table    = annotationheadings
+----------------------------------+----------------------------------+-------+
|              Field               |              Type                | Length|
+----------------------------------+----------------------------------+-------+
| heading1no                       | int4                             |     4 |
| heading1                         | text                             |   var |
| heading2no                       | int4                             |     4 |
| heading2                         | text                             |   var |
| heading3no                       | int4                             |     4 |
| heading3                         | text                             |   var |
+----------------------------------+----------------------------------+-------+

                        Table    = measdep_defaults
+----------------------------------+----------------------------------+-------+
|              Field               |              Type                | Length|
+----------------------------------+----------------------------------+-------+
| alwaysdep                        | int4                             |     4 |
+----------------------------------+----------------------------------+-------+
```

Table 3.4: Script acceleration tables (table structure)

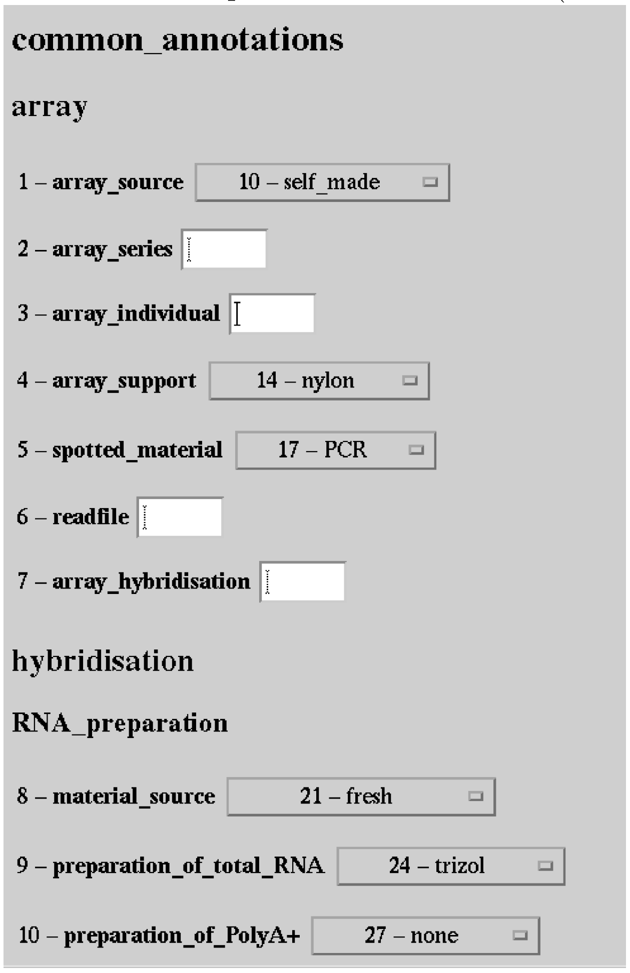
```
                                Table    = minnext
+----------------------------------+----------------------------------+-------+
|              Field               |              Type                | Length|
+----------------------------------+----------------------------------+-------+
| j                                | int4                             |     4 |
| h1                               | text                             |   var |
| h2                               | text                             |   var |
| h3                               | text                             |   var |
| ano                              | text                             |   var |
| vno                              | text                             |   var |
+----------------------------------+----------------------------------+-------+


                                Table    = maxnext
+----------------------------------+----------------------------------+-------+
|              Field               |              Type                | Length|
+----------------------------------+----------------------------------+-------+
| j                                | int4                             |     4 |
| h1                               | text                             |   var |
| h2                               | text                             |   var |
| h3                               | text                             |   var |
| ano                              | text                             |   var |
| vno                              | text                             |   var |
+----------------------------------+----------------------------------+-------+
```

Heading1 is the highest level of the annotation hierarchy followed by lower heading levels, annotations and values. For any number j of a hierarchy element, the number of its first child in the next lower level is recorded in the relation minnext. Here, it is stored under the attribute depicting this next lower level. ‘H1‘ to ‘h3‘ take the numers of elements in the three heading levels, ‘ano’ contains the annotation numbers and vno the value numbers. Likewise, the number of the last child in the next lower level is recorded in the relation maxnext.

Table 3.5: Definition of experimental annotations (table contents)

```
yeast=> select * from annotationheadings order by heading1no, heading2no, heading3no;

heading1no|heading1                     |heading2no|heading2     |heading3no|heading3

----------+-----------------------------+----------+-------------+----------+-----------------------

         1|common_annotations           |         1|array        |         1|-

         1|common_annotations           |         2|hybridisation|         2|RNA_preparation

         1|common_annotations           |         2|hybridisation|         3|labeling

         1|common_annotations           |         2|hybridisation|         4|hybridisation_conditions

         1|common_annotations           |         2|hybridisation|         5|stringency_wash

         1|common_annotations           |         2|hybridisation|         6|detection

         1|common_annotations           |         3|sample       |         7|-

         1|common_annotations           |         4|submission   |         8|-

         2|organism_specific_annotations|         5|genotype     |         9|-

                                        ...  skipping ...

   yeast=> select * from annotations order by lastheadingno, ano, vno;


   lastheadingno| ano|nextano|annotation               | vno|nextvno|value

   -------------+----+-------+-------------------------+----+-------+-----------------------------

              1|   1|      2|array_source             |  10|     11|self_made

              1|   1|      2|array_source             |  11|     12|genome_systems

              1|   1|      2|array_source             |  12|     13|clontech

              1|   1|      2|array_source             |  13|     14|research_genetics

              1|   2|      3|array_series             |   0|      0|[]

              1|   3|      4|array_individual         |   0|      0|[]

              1|   4|      5|array_support            |  14|     15|nylon

              1|   4|      5|array_support            |  15|     16|polypropylene

              1|   4|      5|array_support            |  16|     17|glass

              1|   5|      6|spotted_material         |  17|     18|PCR

              1|   5|      6|spotted_material         |  18|     19|colonies

              1|   5|      6|spotted_material         |  19|     20|DNA-oligo

              1|   5|      6|spotted_material         |  20|     21|PNA-oligo

              1|   6|      7|readfile                 |   0|      0|[]

              1|   7|      8|array_hybridisation      |   0|      0|[]

              2|   8|      9|material_source          |  21|     22|fresh

              2|   8|      9|material_source          |  22|     23|frozen
                                        ...  skipping ...
```

Figure 3.2: Definition of experimental annotations (HTML output)



common_annotations

array

1 – array_source: 10 – self_made

2 – array_series:

3 – array_individual:

4 – array_support: 14 – nylon

5 – spotted_material: 17 – PCR

6 – readfile:

7 – array_hybridisation:

hybridisation

RNA_preparation

8 – material_source: 21 – fresh

9 – preparation_of_total_RNA: 24 – trizol

10 – preparation_of_PolyA+: 27 – none

Table 3.6: Spot categories

category	table name
genes	y1_genes
empty spots	y1_empty
heterologous DNA	y1_hetrl
heterologous DNA with known concentration	y1_hetkc
reference spots	y1_refgs

as described below (3.3.1, 3.4.2 and 3.5.3).

3.2 Microarray Families

3.2.1 Gene annotations

A database can comprise different sorts of microarrays. Each family represents a unique spotting scheme including genes or ESTs and reference spots. For a family referred to as 'y1' by the master table of the database yeast, there are 5 gene annotation tables corresponding to the categories mentioned in 2.2.1 (Tab. 3.6). All of these tables share the same scheme (Tab. 3.7). An index has been computed for every attribute with the name of each index relation consisting of the family, the spot category and an abbreviation of the indexed attribute (attributes and their indexes are listed in the same sequence).

The attribute 'spotno' serves as a key connecting to the tables which contain hybridisation intensities. 'field', 'plate', 'letter' and 'number' correspond to the spot location on the array as well as to the DNA stock kept in microtiter plates. Two fields of fix length ('ext_link7' and 'ext_link10') are reserved for keys linking to external databases and 'description' and 'functional_catalogue' contain a brief description of the protein and its function of variable form and size. Certain spotsets may have to be normalized separately. In such cases the partition of the spots is recorded by the attribute 'partition'. In the example given (Fig. 3.3), which was taken from the database 'humanbiopsy' (containing data derived from renal clear cell carcinoma, family 'hb1'), there are two partitions of the entire set leading to a 'bifurcation' of data points in a scatter plot. In this example the partitions correspond to the location of the spots on two different nylon filters (the spotset being too big to be spotted on one filter) which have to be hybridised in separate tubes.

Figure 3.3: Partitions showing differential slope in a scatterplot

- tumor, pooled fit to median of prim & sec of normal
- sharp bifurcation can be observed even better in linear scale
- primary spots fitted as separate partitions (filtersets)
- sec spots fitted as separate partitions (filtersets)

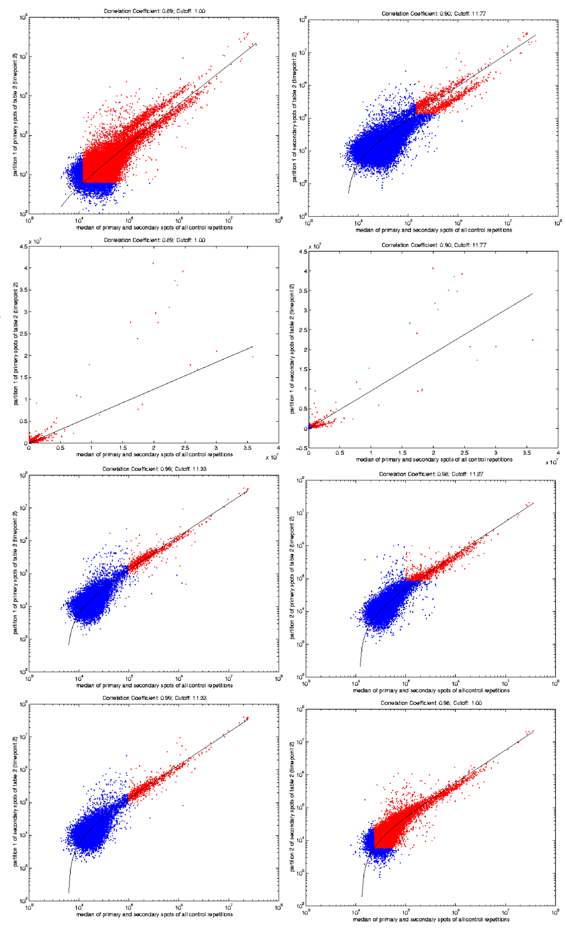


Table 3.7: Gene annotations (table structure)

```
                          Table    = y1_genes
+----------------------------------+----------------------------------+-------+
|              Field               |              Type                | Length|
+----------------------------------+----------------------------------+-------+
| spotno                           | int4                             |     4 |
| field                            | int4                             |     4 |
| plate                            | int4                             |     4 |
| letter                           | char()                           |     1 |
| number                           | int4                             |     4 |
| ext_link7                        | char()                           |     7 |
| ext_link10                       | char()                           |    10 |
| partition                        | int4                             |     4 |
| description                      | text                             |   var |
| functional_catalogue             | text                             |   var |
+----------------------------------+----------------------------------+-------+
Indices:  y1_genes_isn
          y1_genes_if
          y1_genes_ip
          y1_genes_il
          y1_genes_in
          y1_genes_in7
          y1_genes_in10
          y1_genes_ipart
          y1_genes_id
          y1_genes_ifc
```

Table 3.8: Information about an array family (table structure)

```
                          Table    = y1
+----------------------------------+----------------------------------+-------+
|              Field               |              Type                | Length|
+----------------------------------+----------------------------------+-------+
| experiments                      | int4                             |     4 |
| tables                           | int4                             |     4 |
| i_fileformat                     | int4                             |     4 |
+----------------------------------+----------------------------------+-------+
```

3.2.2 Administration of the comprised multiconditional experiments

There are two more tables belonging to an array family (see detailed scheme `http://www.dkfz-heidelberg.de/tbi/services/mchips/scheme2.pdf`, also in 'DATABASE MANAGEMENT'). To stick to the example family 'y1' (comprehensive yeast filter), there is a table named 'y1' (Tab. 3.8) storing the number of multiconditional experiments as well as the number of measurements in the family. Since each measurement is initially stored in a separate table (see 3.5.1) and identified with a unique table number, their quantity is attributed as 'tables'. Generally, 'measurement' 5 identifies the 5th measurement of a particular experiment (see 3.4.1), whereas 'tables' / 'tableno' hold quantity / IDs of measurements on a family-wide scale (even when the initial tables have been merged into a block).

Table 3.9: Experiments contained in an array family (table structure)

```
                                 Table    = y1_master
+----------------------------------+----------------------------------+-------+
|              Field               |              Type                | Length|
+----------------------------------+----------------------------------+-------+
| experiment                       | int4                             |     4 |
| ex_name                          | text                             |   var |
| ex_table                         | int4                             |     4 |
| conditions                       | int4                             |     4 |
| condep_ano                       | text                             |   var |
+----------------------------------+----------------------------------+-------+
Indices:   y1_master_iex
          y1_master_ien
          y1_master_iet
          y1_master_ico
          y1_master_ica
```

The third attribute ('intput file format') stores the version number of the script capable of reformatting an output file of a particular imaging software into the format of a database table. This matlab function exists in different versions enumerated sequentially for different imaging software types and spotting schemes.

The second table lists the multiconditional experiments contained by the family (Tab. 3.9). Each experiment is assigned a number and a name. 'ex_table' links to the administration table for the hybridisation intensities as well as to the experimental annotations. For convenience in algorithmical handling, we redundantly included here also the number of comprised conditions as well as the varied experimental parameters ('condition-dependent annotations').

3.3 Multiconditional Experiments

3.3.1 Experimental annotations constant within the multiconditional experiment

There may be an arbitrary number of multiconditional experiments hybridised on a particular filter family. They may be timecourses, variations of agent concentrations in culture media, comparisons of different genotypes just to give some examples, consisting of several experimental conditions intended to be directly comparable. To learn something from such a comparison not too many parameters should be altered among the conditions performed. Hence most of the experimental conditions are constant for the entire experiment, some are condition dependent and some are measurement dependent, i.e. they can take different values for each single measurement, like e.g. the label

Table 3.10: Experimental annotations constant throughout the experiment (table structure)

```
                    Table    = y1_constant_categoricalvalue_65
+----------------------------------+----------------------------------+-------+
|              Field               |              Type                | Length|
+----------------------------------+----------------------------------+-------+
| experiment                       | int4                             |     4 |
| ano                              | int4                             |     4 |
| annotation                       | text                             |   var |
| vno                              | int4                             |     4 |
| cvalue                           | text                             |   var |
+----------------------------------+----------------------------------+-------+
```

```
                         Table    = y1_constant_number_65
+----------------------------------+----------------------------------+-------+
|              Field               |              Type                | Length|
+----------------------------------+----------------------------------+-------+
| experiment                       | int4                             |     4 |
| ano                              | int4                             |     4 |
| annotation                       | text                             |   var |
| vno                              | int4                             |     4 |
| nvalue                           | float8                           |     8 |
+----------------------------------+----------------------------------+-------+
```

incorporation rate. For fast annotation via html questionaire, the data are required in the form of these three sets of annotations. For statistical analysis, they are needed hybridization wise. Redundancy caused by hybridization wise storage of the entire set of annotations would have little effect in terms of storage space or performance because these annotations are of negligible volume. However, we decided to store them in separate relations for convenient algorithmical handling: Splitting up a uniform set of hybridization wise stored annotations into hybridization-dependent, condition-dependent and constant annotations requires repeated value comparison, whereas the distribution of constant and condition-dependent annotations to each hybridization is a trivial task.

Constant annotations are stored in two separate tables per multiconditional experiment just to be more readable rather than for computational reasons (Tab. 3.10). These tables are children of parental tables 'y1_constant_categoricalvalue' and 'y1_constant_number' respectively. The numbers within their names as well as the content of the field 'experiment' correspond to the according key in y1_master. The first table takes the enumeration type ('categorical') annotations, the second one those consisting of a number. This is reflected by the type of the attributes 'cvalue' and 'nvalue' which is the only difference among the above schemes. As a representative of intended redundancy both number ('ano') and name ('annotation') are enlisted for an annotation as well as for its value. For the small extent of the annotations (1.3.2.2) this does

Table 3.11: Association of experiments, conditions and measurements (table structure)

```
                              Table    = y1_experiment_65
+----------------------------------+----------------------------------+-------+
|              Field               |               Type               | Length|
+----------------------------------+----------------------------------+-------+
| experiment                       | int4                             |     4 |
| condition                        | int4                             |     4 |
| hybridization                    | int4                             |     4 |
| measurement                      | int4                             |     4 |
| tableno                          | int4                             |     4 |
+----------------------------------+----------------------------------+-------+
```

not have major consequences for storage space nor for performance. However the redundancy might serve to reconstruct experimental annotations (which would be very time consuming to re-enter by hand) if an error occurs in the numbering of annotations or values. Redundant storage appears advisable here because as new kinds of experiments evolve, annotation definitions are under constant change.

3.3.2 Administration of the comprised condiditions

For each condition in a multiconditional experiment, there is a table like the following which for our example family y1 inherits from a parental relation y1_experiment. For the above experiment no. 65 it will be named y1_ex_65 (Tab. 3.11). 'experiment' will contain a 65 as well for the entirety of tuples to identify the experiment in a family-wide context, since the experiment tables can be merged into big block relations as for the intensities (see 2.2.2, 3.5.2). The comprised conditions have been studied by several repeatedly performed hybridizations which themselves consist of one (radioactive labelling, monochannel) or more (multichannel fluorescence data) measurements (frequently called channels or images). While 'measurement' identifies a measurement in the context of its particular experiment, 'tableno' holds its family-wide ID. Both remain unchanged when the initial tables are merged into a block.

Table 3.12: Association of experiments, conditions and measurements (table content)

```
yeast=> select * from y1_experiment_65 order by tableno;
experiment|condition|hybridization|measurement|tableno
----------+---------+-------------+-----------+-------
        65|        0|            1|          1|    576
        65|        0|            2|          2|    577
        65|        0|            3|          3|    578
        65|        1|            4|          4|    579
        65|        1|            5|          5|    580
        65|        1|            6|          6|    581
        65|        2|            7|          7|    582
        65|        2|            8|          8|    583
        65|        2|            9|          9|    584
        65|        2|           10|         10|    585
        65|        2|           11|         11|    586
        65|        3|           12|         12|    587
        65|        3|           13|         13|    588
        65|        3|           14|         14|    589
        65|        3|           15|         15|    590
        65|        3|           16|         16|    591
                (16 rows)
```

3.4 Experimental Conditions

3.4.1 Administration of the comprised hybridizations and measurements

The number of successfully performed hybridizations and measurements may vary among the conditions. As an example we show the content of the above relation which outlines an experiment with radioactive (monochannel) hybridizations (Tab. 3.12). The control condition is identified by a zero whereas numbering of hybridizations and measurements starts at one. While in the above case the measurement IDs correspond to those of the hybridizations, they are different in multichannel experiments where each hybridization comprises more than one measurement belonging to different conditions. Whereas the sequence recorded in 'measurement' is due to the experiment (with the first one of a hybridization usually being the 'red' channel), the purpose of 'tableno' is rather technical. It simply corresponds to the order in which they were uploaded into the database, being a unique ID.

3.4.2 Experimental annotations dependent on the condition

The condition dependent annotations describing experiment no. 65 are stored in y1_conditiondependent_65 (Tab. 3.13). It shows the same structure as

Table 3.13: Condition dependent annotations (table structure)

```
                    Table    = y1_conditiondependent_65
+----------------------------------+----------------------------------+-------+
|              Field               |              Type                | Length|
+----------------------------------+----------------------------------+-------+
| experiment                       | int4                             |     4 |
| condition                        | int4                             |     4 |
| ano                              | int4                             |     4 |
| annotation                       | text                             |   var |
| vno                              | int4                             |     4 |
| cvalue                           | text                             |   var |
| nvalue                           | float8                           |     8 |
+----------------------------------+----------------------------------+-------+
```

Table 3.14: Condition dependent annotations (table content)

```
yeast=> select * from y1_conditiondependent_65 order by ano, condition, vno;
experiment|condition| ano|annotation        | vno|cvalue            |nvalue
----------+---------+----+------------------+----+------------------+------
        65|        0|1035|strain            |1091|3E2               |NaN
        65|        1|1035|strain            |1091|3E2               |NaN
        65|        2|1035|strain            |1092|702               |NaN
        65|        3|1035|strain            |1092|702               |NaN
        65|        0|1037|genetic_variation |1100|WT                |NaN
        65|        1|1037|genetic_variation |1100|WT                |NaN
        65|        2|1037|genetic_variation |1099|inducible_promoter|NaN
        65|        3|1037|genetic_variation |1099|inducible_promoter|NaN
        65|        0|1038|transgene         |   0|***               |0
        65|        1|1038|transgene         |   0|***               |0
        65|        2|1038|transgene         |   0|***               |4111
        65|        3|1038|transgene         |   0|***               |4111
        65|        0|1049|glucose           |   0|***               |2
        65|        1|1049|glucose           |   0|***               |0
        65|        2|1049|glucose           |   0|***               |2
        65|        3|1049|glucose           |   0|***               |0
        65|        0|1050|galactose         |   0|***               |0
        65|        1|1050|galactose         |   0|***               |2
        65|        2|1050|galactose         |   0|***               |0
        65|        3|1050|galactose         |   0|***               |2
(20 rows)
```

for the constant annotations (described in 3.3.1), except for including both numbers (stored in 'nvalue') and enumeration type values (in 'cvalue') into one table. Moreover it contains an additional attribute accounting for the condition. Enumeration of conditions starts at zero for the control condition (Tab. 3.14). In this particular experiment both the genotype of the yeast cells and the carbon source of their medium had been varied. For enumeration type annotations like 'strain', a valid valuenumber ('vno') is listed but the field 'nvalue' contains 'not-a-number'. Conversely, floating point number annotations like 'transgene' or 'glucose' have valueno 0 and a dummy entry for 'cvalue', but a meaningful 'nvalue' (namely the floating point value, wich happens to be always a natural number in the above table).

Like in the above tables, a field is included that denotes the experiment

Table 3.15: Hybridization intensities (table structure)

```
                          Table    = y1_g_589
+----------------------------------+----------------------------------+-------+
|              Field               |              Type                | Length|
+----------------------------------+----------------------------------+-------+
| tableno                          | int4                             |     4 |
| spotno                           | int4                             |     4 |
| prim                             | float8                           |     8 |
| sec                              | float8                           |     8 |
| prim_bkg                         | float8                           |     8 |
| sec_bkg                          | float8                           |     8 |
+----------------------------------+----------------------------------+-------+
Indices:  y1_g_589_ipr
          y1_g_589_ise
          y1_g_589_isn
```

number for every tuple for identification in block context. The according parental tables (in the above case 'y1_conditiondependent' is the name of the parent) are themselves empty but mediate queries on all of their children (see in 2.2.2). This means that the query syntax given on top of the table list is never used. Instead all the algorithms involved would query this table by

```
yeast=> select * from y1_conditiondependent* where ex=65 order by ano, condition, vno;
```

resulting in the very same list.

3.5 Hybridisations

3.5.1 Hybridisation intensities

As listed in the administration table for experiment 65 (see 3.4.1), the third measurement of the last condition is hybridisation number `589`. The corresponding intensities are stored in 5 separate tables (compare 2.2.1 & 3.2.1), being accessed via the parental tables 'y1_g', 'y1_e', 'y1_h', 'y1_k' and 'y1_r'. They are of a uniform structure they inherited from their uniform parents, in example listed in Tab. 3.15. Since this kind of tables is also accessed by querying the parental table, 'tableno' mediates identification in block context, linking to the administration table (y1_experiment). 'Spotno' identifies the spot, corresponding to the identically named attribute of the gene annotation table 'y1_genes' (3.2.1). In the tables 'y1_e_589', 'y1_h_589', 'y1_k_589' and 'y1_r_589' this attribute corresponds to the 'spotno' in 'y1_empty', 'y1_hetrl', 'y1_hetkc' and 'y1_refgs', respectively. The remaining attributes contain the hybridisation intensities. Each gene or EST has been spotted in duplicate resulting in two intensities ('prim' and 'sec') per hybridisation. The last two

Table 3.16: Hybridization intensities in a block (table structure)

```
                              Table    = y1_g_block1
+----------------------------------+----------------------------------+-------+
|              Field               |               Type               | Length|
+----------------------------------+----------------------------------+-------+
| tableno                          | int4                             |     4 |
| spotno                           | int4                             |     4 |
| prim                             | float8                           |     8 |
| sec                              | float8                           |     8 |
| prim_bkg                         | float8                           |     8 |
| sec_bkg                          | float8                           |     8 |
+----------------------------------+----------------------------------+-------+
Indices:   y1_g_block1_ipr
          y1_g_block1_ise
          y1_g_block1_isn
          y1_g_block1_itn
```

attributes are intended to take a local background value which is delivered by most of the imaging software packages. Three indices have been computed. 'y1_g_589_ipr' and 'y1_g_589_ise' facilitate the search for specific hybridisation intensities ('pr' and 'se' for primary and secondary spots), 'y1_g_589_isn' querying certain spot numbers.

Many imaging software packages yield more than one intensity score and background per spot. Commonly, they provide differently calculated intensities (e.g. pixel mean, median), background intensities and various kinds of quality or reliability measures. From these, the contents of the above tables are either choosen or calculated as a starting point for standardized analysis in the process of database upload.

3.5.2 Hybridisation intensities, 'solidified'

As experiments are analysed and valued, hybridisations are deleted e.g. for bad signal quality, written into another context or kept in the experiments and conditions they were uploaded in. When a set of hybridisations is not to be altered any more, it is solidified, that is written into large block tables over night, as mentioned in 2.2.2.The separation into the 5 spot categories is kept resulting in 5 block tables. Tuples of the above table will go e.g. into y1_g_block1 (Tab. 3.16). These tables have exactly the same structure as the normal hybridisation tables. The only difference is that an index was computed for the table numbers (named 'y1_g_block1_itn') enabling rapid hybridisation wise retrieval of the tuples from the block. Such a block was tested with up to 538 hybridisations of the y1 type (comprising 6103 genes), speeding up retrieval of an entire multiconditional experiment up to 15fold compared to the unsolidified version depending on how many hybridisations

Table 3.17: Measurement-dependent annotations (table structure)

```
                        Table    = y1_measurementdependent_65
+----------------------------------+----------------------------------+-------+
|              Field               |              Type                | Length|
+----------------------------------+----------------------------------+-------+
| experiment                       | int4                             |     4 |
| condition                        | int4                             |     4 |
| measurement                      | int4                             |     4 |
| ano                              | int4                             |     4 |
| annotation                       | text                             |   var |
| vno                              | int4                             |     4 |
| cvalue                           | text                             |   var |
| nvalue                           | float8                           |     8 |
+----------------------------------+----------------------------------+-------+
```

are comprised, and on its position in the database.

3.5.3 Measurement-dependent experimental annotations

For measurement-dependent annotations, structures mentioned for the condition dependent annotations (3.4.2) apply as well. The table y1_measurementdependent_65 contains the measurement-dependent annotations of multiconditional experiment 65 (Tab. 3.17). It inherits from y1_measurementdependent and has the same structure as y1_conditiondependent_65 except for one additional attribute 'measurement', which is related to the intensity tables by relation y1_experiment_65. 'Condition' is related to 'measurement' here as well to secure this important information by repeated storage[3]. Although all defined annotations have to be annotated for a multiconditional experiment, their distribution among the hybridisation-dependent, condition-dependent and constant database relations may vary from experiment to experiment. Annotation starts by choosing the annotations which shall become measurement-dependent and thereafter assigning a value to each of those annotations for each measurement. Thereafter, the condition dependent annotations are selected and annotated before the remaining constant annotations are entered. The annotation process is mediated by a web interface such that annotation can be performed from remote sites, enabling annotation even before uploading of intensities, re-editing of assigned values and copy from similar experiments to save the user from re-entering identical values.

[3]'Hybridization' is not included in this table, because assignment to the hybridizations may readily be recovered from the sequence given by attribute 'measurement'.