

# GENOME-MAP: Real-World Test Data and Queries for Logic Databases<sup>\*†</sup>

Steve Rozen Mark J. Daly Mary-Pat Reeve Nathan Goodman

<http://jura.wi.mit.edu/rozen>

Whitehead Institute for Biomedical Research  
One Kendall Square  
Cambridge MA 02139

December 27, 1993

## Abstract

In the process of trying to find a logic query language to use with our genome-mapping database we extracted test data from our database and developed a set of representative queries over the data. We also synthesized data for a project that does not yet have released data.

This data is available to those developing or testing logic databases and logic programming languages.

## 1 Introduction

There are two data sets:

- Workflow Data
- Physical Map Data

Our primary purpose in making these data sets and queries available is to help those developing or evaluating logic database or logic programming languages.

---

<sup>\*</sup>This work was supported by funds from the National Institutes of Health, National Center for Human Genome Research, grant number P50 HG00098.

<sup>†</sup>©Whitehead Institute for Biomedical Research, 1993. All rights reserved.

## 1.1 How to Obtain the Data

The data sets are available [at http://jura.wi.mit.edu/rozen/papers/rozen-et-al-1993-genome-map/0.135](http://jura.wi.mit.edu/rozen/papers/rozen-et-al-1993-genome-map/0.135) in various subdirectories of [ftp://pub/steve/GENOME-MAP](ftp://pub.steve.GENOME-MAP), which we henceforth abbreviate as *GENOME-MAP*.

## 2 Workflow

The workflow data sets consist of released data from the Whitehead/MIT Center for Genome Research (Genome Center). The data sets are up-to-date as of December, 1993, but may become out-of-date in the future. The workflow data set records workflow in the Genome Center's genetic-linkage mapping project for the mouse (*Mus musculus* and *Mus musculus castaneus*). The purpose of this project is to find short DNA sequences called "markers" and determine their locations on the mouse chromosomes. The production information system that stores this data is Mapbase [2, 1], which is constructed on top of the object-oriented database management system ObjectStore [3]. Most of the sample queries on the workflow data are representative of queries that are commonly posed against MapBase. We explicitly note any queries that are not representative.

### 2.1 EDB Relations

Each EDB relation represents a particular workflow step in the process of finding a marker. Each potential marker is identified by a marker id. As we process a potential marker, we create entries for it in the workflow EDB relations. Each workflow EDB relation records the marker id, the date of the workflow step, the user that is responsible for the workflow step, and any data associated with that workflow step.

#### 2.1.1 Schema

In the arguments for EDB Relations below we use the following abbreviations:

D	date	an integer in the form YYYYMMDD
Desc	description	an uninterpreted text description
M	marker id	a string
L	left	
Len	length	an integer
Loc	location	usually a string describing some location <i>not</i> in the database
R	right	
U	user	a string representing the user or software responsible for some step

The workflow relations are:

```
box(M,D,U,Box_number)
choice(M,D,U,L_primer_start,L_primer_len,L_primer_seq,
       R_primer_start,R_primer_len,R_primer_seq,
```

```

    Outcome,X1,X2)
comment(M,D,U,Comment)
genbank_info(M,D,U,Source_species,Offset,Map_position,
             Desc,Accession_id)
marker_insert(M,D,U,Start,Len)
marker_BLAST(M,D,U,Genbank_name,Desc,Score)
marker_FASTN(M,D,U,Outcome,Match_type,Score)
mapmaker(M,D,U)
marker(M,D,U)
marker_locus(M,D,U,Locus,X1,X2)
marker_name(M,D,U,Alternative_name)
order(M,D,U)
primer(M,D,U,Result_file)
received(M,D,U)
sequence(M,D,U,Seq,Source_strain,Source,Source_loc)
sequence_hold(M,D,U,Hold_flag,Why_held)
target(M,D,U,Start,Len,Target_type)
typing(M,D,U,T_panel,T_panel_suffix,Rank,Typing,Nomenclature,Loc)

```

Most of these relations can have multiple entries for a particular marker. The exception is `marker`, which implements the extent of all markers. For most of these relations, the most recent entry for a particular marker is of primary interest. The exceptions are `comment`, `marker_name`, `marker_locus`, and `marker_BLAST`. For `comment`, `marker_name`, and `marker_locus`, all records are interesting. For `marker_BLAST`, usually all records with a non-empty `Genbank_name` are of interest.

### 2.1.2 How to Obtain

The data is available in `GENOME-MAP/data/workflow/40/{sequence,other}.Z`. The file `sequence` contains the `sequence` relation, consisting of 7,254 rows in 3.7 Mbytes (uncompressed). The file `other` contains all the other relations, totaling 77,287 rows in 4.7 Mbytes (uncompressed).

## 2.2 Common Queries

A common type of query is one that retrieves the most recent values produced by several workflow steps for one or a list of markers. More formally, for workflow step  $S$  (i.e. EDB relation  $S$ ) let the relation `most_recent_S` be the subset of  $S$  containing only the most recent tuples<sup>1 2</sup> for each marker (which is always the first argument).

A simple example of this type of query would be:

---

<sup>1</sup>Because a step could be performed twice in one day, there could be more than one most recent entry for a particular marker.

<sup>2</sup>As discussed above, the definition of a `most_recent_S` does not make sense for  $S \in \{\text{marker}, \text{comment}, \text{marker\_name}, \text{marker\_locus}, \text{and } \text{marker\_BLAST}\}$ .

```
?- most_recent_sequence(M,--,Seq,--,--),
    most_recent_marker_insert(M,--,Start,Len).
```

where  $M$  is a constant marker id.

A realistic example involving more relations would be join (on  $M$ ) of

- `most_recent_sequence`
- `most_recent_marker_insert`
- `most_recent_choice`
- `most_recent_screening`
- `most_recent_typing`

Both queries would be useful for

- single markers,
- a list of 10 markers, or
- all markers in the database.

Finally, a another useful query would be:

- For a given marker, print
  - the `most_recent_sequence` entry (or entries)
  - the `most_recent_marker_insert` (or entries), if any
  - all `comment` entries, if any, and
  - all `marker_BLAST` entries with a non-empty `GenBank_name`, if any.

## 2.3 Data-Dredging Queries

Any *particular* data-dredging query is likely to be relatively infrequent (though as a class they may be relatively common). We include data-dredging queries because they are often thought to be a good application of logic query languages [4], and because the need for data-dredging arises frequently in the operation of MapBase. Because data-dredging queries are exploratory in nature and are often performed only a few times, they often needn't be particularly fast. For the same reasons, users will want to minimize programming time spent on them.

- Print out `marker_locus` tuples for all markers with more than one `marker_locus` tuple.
- For all `typing` steps prior to the last release (19931020), what percentage were re-done?

- Print all markers in which a `mapmaker` step is followed by a `typing` step, and in which the difference in the date of the `mapmaker` step and the `typing` step is a least 1 day, and in which the date of the typing is after the last release.
- What markers were ordered but have not been received (using information in the `most_recent_order` and `most_recent_received` relations).
- Assume the idealized sequence of steps for a marker is:
  1. `sequence` or `genbank_info`
  2. `marker_insert`
  3. `target`
  4. `marker_BLAST`
  5. `choice`
  6. `received`
  7. `screening`
  8. `typing`
  9. `mapmaker`

Using the `most_recent` version of each step, what percentage of markers drops out between each pair of steps?

- For those markers with a non-empty most-recent sequence, `Seq`, check that the subsequent most-recent choice tuple (if any) satisfies the following invariant:

$$\text{substring}(\text{Seq}, \text{L\_primer\_start}, \text{L\_primer\_len}) = \text{L\_primer\_seq}$$

## 2.4 Updates

It would be perfectly reasonable to materialize the `most_recent` versions of the EDB relations, especially since most queries do not involve historical data. However, in that case updates must ensure that the data in the historical and `most_recent` relations are consistent. Steps can be entered in any order, and there is no guarantee that a step being entered is the most recent.

Here are two representative updates:

- Add a new tuple to `marker_name`. Ensure that the 1st argument (`M`) indeed exists as the 1st argument of a tuple in the relation `marker`. Ensure that the 4th argument is not already the 1st argument of any tuple in `marker`, and is not already the 4th argument of any tuple in `marker_name` or `marker_locus`.
- Add a new tuple to `choice`. Ensure that the 1st argument exists in relation `marker` and that the marker has a sequence (i.e. that there is a `sequence` tuple with the same 1st argument).

## 3 Physical Map Data and Queries

The physical map data is synthesized, but the synthesis is designed to mimic realistic data. The physical map data resembles data that will (and has been) generated for the Genome Center’s physical mapping projects for mouse and human genomes.

Physical mapping involves finding overlapping DNA fragments. In the data we supply, each DNA fragment is a yeast artificial chromosome (YAC)—DNA that is part of an organism under study (e.g. human), and that is being propagated in a yeast cell. We don’t know the DNA sequences of the YACs because they are too numerous and too long. So we determine overlap by detecting the presence of a unique “probe” in different YACs. (The probe is unique within the genome.) When a probe is detected in a YAC we say that the probe “hits” the YAC.

Detecting the presence of a probe involves a chemical assay, during which errors can occur, so the data distinguishes “ambiguous hits” from (unambiguous) “hits”. An additional problem is “chimerism”: a YAC can be the concatenation of separate fragments of DNA. Therefore, even when two probes hit the same YAC, we cannot assume that they are near each other in vivo. Therefore we usually insist that two probes be connected by more than one YAC.

### 3.1 EDB Relations

- `hit(Probe,YAC)` Probe definitely hits YAC.
- `amb_hit(Probe,YAC_set)` Probe ambiguously hits each YAC in `YAC_set`.

In the data, YACs and probes are identified by a string.

#### 3.1.1 How to Obtain

Physical-mapping data is in `GENOME-MAP/data/physical/r0/{hit,amb_hit}`. Each file contains the EDB relation of the same name: `hit` contains 5,388 tuples in 3.0 Mbytes (uncompressed), and `amb_hit` contains 1,766 tuples in 1.8 MBytes (uncompressed).

### 3.2 Queries

1. Two probes,  $p$  and  $q$ , are in the same *group* if either

- $p$  and  $q$  both (unambiguously) hit two different YACs,  $Y$  and  $Z$ , i.e.

$$\text{hit}(p,Y), \text{hit}(q,Y), \text{hit}(p,Z), Y \neq Z, \text{hit}(q,Z)$$

or

- $p$  and  $q$  both hit one YAC,  $Y$ , and  $p$  hits a second YAC,  $Z$ , which is contained in a YAC set that  $q$  ambiguously hits, i.e.

$$\text{hit}(p,Y), \text{hit}(q,Y), \text{hit}(p,Z), Y \neq Z, \text{amb\_hit}(q,S), Z \in S$$

Print all equivalence classes of probes over the reflexive and transitive extension of the group relation.

2. If two probes,  $p$  and  $q$ , both hit YAC  $Y$ ,  $p$  hits YAC  $W$ , and  $q$  ambiguously hits a YAC set containing  $W$ , then we say that  $q$  *probably hits*  $W$ . Inductively define an  $n$ -linked YAC path between probes  $p$  and  $q$  as a list of probes beginning with  $p$  and ending with  $q$ , as follows:

- (a)  $[p, q]$  is an  $n$ -linked YAC path between  $p$  and  $q$  if  $p$  and  $q$  both hit or probably hit each YAC in a set of (at least)  $n$  YACs.
- (b)  $[p, \dots, r, \dots, q]$  is an  $n$ -linked YAC path if both  $[p, \dots, r]$  and  $[r, \dots, q]$  are  $n$ -linked YAC paths.

A minimal  $n$ -linked YAC path is one in which removing any probe yields a list that is not an  $n$ -linked YAC path.

Print all minimal 2-linked YAC paths between two given probes.

3. An unambiguous  $n$ -linked YAC path between probes  $p$  and  $q$  is one that involves no ambiguous (or probable) hits.

Print all minimal unambiguous 2-linked YAC paths between two given probes.

## References

- [1] N. Goodman. An object oriented DBMS war story: Developing a genome mapping database in C++. In W. Kim, editor, *Modern Database Management: Object-Oriented and Multidatabase Technologies*. ACM Press, 1994.
- [2] N. Goodman, S. Rozen, and L. Stein. Requirements for a deductive query language in the MapBase genome-mapping database. In *Proceedings of the Workshop on Programming with Logic Databases In Conjunction with ILPS, Vancouver, B.C.*, pages 18–32, Oct. 1993. Available by anonymous ftp from `genome.wi.mit.edu` as `pub/steve/requirements.ps`. Entire proceedings available as Technical Report #1183, Computer Sciences Department, University of Wisconsin, Madison WI 53706, USA.
- [3] C. Lamb, G. Landis, J. Orenstein, and D. Weinreb. The ObjectStore database system. *Communications of the ACM*, 34(10):50–63, Oct. 1991.
- [4] S. Tsur. Data dredging. *Data Engineering*, 13(4), Dec. 1990.