



# MIAH:
## An Application Programming Interface for HGVbase

By
Iván Rodríguez Rodríguez

# ABSTRACT

Genetics is a fascinating field of science which covers a huge range of subjects. One area in particular has received a huge amount of interest: The area of genetic variation.

The study of polymorphisms, which are common variations in the sequence of DNA among individuals, has become more popular in the last few years. This area involves studying DNA sequences to find the relationship between polymorphisms in the genome and human physical characteristics such as height, weight or more important, the tendency to suffer from a disease or a method to cure it.

The study of polymorphisms has led to the development of databases to store the information; one such database is HGVbase. HGVbase is an online database that stores polymorphisms. Thanks to HGVbase, scientists all over the world can work together, sharing this powerful tool, to extract and to submit polymorphisms in a quick and easy way.

However, HGVbase has some features that can be upgraded to attain better performance. In this report we analyse the weak points and suggest some improvements that can be made. The objective of this report is to give a description of this tool by illustrating what, why and how things can be upgraded, with the final purpose of upgrading the HGVbase environment.

# FOREWORD

This report is a Master's Thesis in Computer Science at the department of Numerical Analysis and Computer Science of KTH, Stockholm. The work was carried out at the Center for Genomics and Bioinformatics of Karolinska Institutet in the research group directed by professor Anthony Brookes.

I would like to thank several people, without whom this thesis would not have been possible. First, to professor Henrik Eriksson who tracked my work, gave me inestimable advice and calmed me down in the moments I was close to insanity. And next, to David Fredman, who gave me accurate advice and directed me all through the thesis, to Gillian Munns, who helped me with my –terrible– English all along this report, to Daniel Ríos, who supported me with his experience from his previous Master Thesis, and to all those who patiently read all versions of this report and gave me adequate and valuable feedback.

Iván Rodríguez.

# ABBREVIATIONS

| | |
|---|---|
| API | Application Programming Interface. |
| CGB | Center for Genomics and Bioinformatics. |
| CPAN | Comprehensive Perl Archive Network. |
| DBI | Database Interface. |
| DNA | Deoxyribonucleic Acid. |
| GNU | A recursive acronym for “GNU is Not Unix”. |
| HGVbase | Human Genome Variation database. |
| HGVbaseID | HGVbase identifier. |
| HTML | Hypertext Markup Language. |
| I/O | Input / Output. |
| ID | Identifier. |
| KI | Karolinska Institutet. |
| KTH | Kungliga Tekniska Högskolan. |
| MIAH | Middleware Integration Application for HGVbase. |
| MS | Microsoft. |
| ODBC | Open Database Connectivity. |
| OO | Object Oriented. |
| OOD | Object Oriented Design. |
| PERL | Practical Extraction and Report Language. |
| SQL | Standard Query Language. |
| SNP | Single Nucleotide Polymorphism. |
| XML | Extensible Markup Language. |

# TABLE OF CONTENTS

# 1 INTRODUCTION

Genetics is a scientific field that has gained momentum in the last few years. The Human Genome Project is the largest scientific task ever, and the recent announcement of the complete human DNA sequence has been given extensive media coverage. It has also given rise to a common misunderstanding: That the human genome is no longer a research subject.

On the contrary, one of the main tasks in genetics is still the study of the genome: Now scientists are looking for differences between individual genomes. Those differences, called SNPs (Single Nucleotide Polymorphisms), may modify a gene and could be responsible for the different sets of characteristics that the human species presents all over the world.

However, the number of SNPs is immense. At present, more than two million SNPs have been reported, and there are still many more to be found. This is where computer science enters: Managing such a huge number of registers is a task for modern computers and database engines like HGVbase.

## 1.1 A Statement of the Problem

HGVbase is a database that stores SNPs and makes them available to the entire world. Its users range from a medical scientist performing direct queries for information to a statistician running programs interacting with the database.

However, as the range of information accommodated in it is growing, modifications and upgrades need to be done in order to maintain its usefulness. In principle, such modifications of the database may necessitate modifications of all application programs using it. This is the main problem addressed by this thesis.

Another difficulty with a database that is constantly being updated by different sources of varying quality is the need for data record history management, and we will also be studying that problem.

## 1.2 Goals of the Project

The primary goal of this thesis is to design a way of hiding the database implementation, thus minimizing the impact made by database modifications mostly to the applications that are now running directly against the database.

The secondary goal is to unify the different access methods to the database thus making the creation of new related functionalities easier.

The final goal is the design of an historical management system to grant optimal tracking of database records history and good browsing through it by the database user.

## 1.3 Contents of the Thesis

The objective of this thesis is to show how to design and to implement an application, which will be called MIAH as an acronym for Middleware Integration Application for HGVbase, to be placed between database users (computer programs mostly, but also human users) and the database itself, that is, acting as a gateway. To achieve it, several functionalities will have to be available: Input/output operations, up-to-the-user queries, general database modifications, and management of upgrades and history of the database.

As this is an ambitious project, we don't claim completeness in our treatment. However, all of the objectives will be reached at least in a theoretical approach. This means formulating the problem and describing a method to solve it.

In addition to this report, a user manual for MIAH will be supplied. The manual will serve as a guide for present and future users, showing the application's interface, main features in the code (to allow future programmers to extend and modify the application), and how to manage the application.

# 2 OVERVIEW OF HGVbase

Before starting with MIAH, we should give some facts about the environment in which it fits and in which this thesis has been written.

## 2.1 Karolinska Institutet and CGB

Karolinska Institutet (KI) is Sweden's best-known university for medicine. It offers several training programs as well as numerous further-education and independent courses. KI is also a research institute which allows students and postgraduate students to take part in advanced research under the supervision of established researchers.

The Center for Genomics and Bioinformatics (CGB) is a young academic department of KI (created in 1997) hosting over 100 researchers in the fields of functional, clinical and structural genomics, as well as genomic technologies and bioinformatics. Through different ongoing projects, the CGB creates and manages genetic information to discover connections between genes, proteins and their functions, that will lead to understanding of human disease and to the development of new drugs and methods to fight it.

## 2.2 HGVbase

HGVbase is one of the projects being carried out at CGB by Anthony Brookes' team, and MIAH is part of it. Consequently, a brief overview of HGVbase, its behaviour, structure and purpose may be useful in order to get into MIAH's context.

### 2.2.1 Purpose

In a nutshell, HGVbase is a database mounted over MySQL[1] that provides an accurate, highly useful and ultimately fully comprehensive catalogue of normal human gene and genome variation. Variations in the genome define the genotype[2] of each individual; as phenotype is the physical representation of the genotype, the variations of a genotype may be responsible for the observable traits of the owner such as hair colour or more importantly, the presence or absence of a disease. Thus, by summarizing all known variations in the human genome as a non redundant set of records, the genotype-phenotype association analyses are facilitated

HGVbase is supported by public (mainly) and private funds, and receives data from several sources (see figure 2.1 on the next page). Thanks to these periodical submissions, HGVbase has grown remarkably (from just several ten thousands to more than two million entries) and today is a major research tool which is used in the study of the genetic component of human phenotypic variation.

[1] MySQL is a database engine. See Appendix II, subsection 2 for a better explanation.
[2] See definitions of genetic terms in Appendix I.

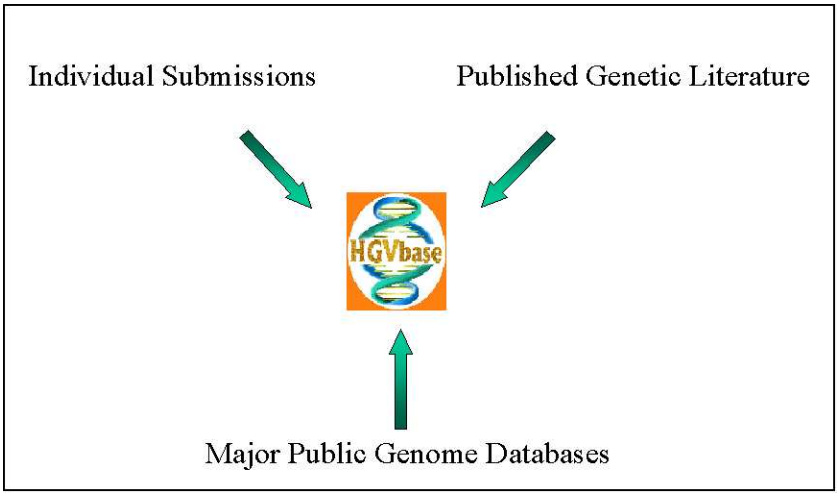


**Figure 2.1. Different sources for HGVbase.**

### 2.2.2 Structure

HGVbase is a set of non-redundant polymorphism records that accommodate single base polymorphisms (SNPs), insertion-deletion variants, simple tandem repeat polymorphisms and generic changes involving alterations not described by the preceding three alternatives. Identifiers for the database are created by adding a number (given by a positive counter) to a 3-letter code (that represents the variation). In addition to the identifier, several other information items are stored on each entry:

- Genomic DNA sequences and/or coding sequences.
- Gene name and symbol where the variation is located.
- An access number to an equivalent register in other relevant databases.
- Description and personal information of the submitter.
- Polymorphism's location within the gene.
- Information about whether the polymorphism is proven, or suspected and why.
- Allele frequency for "populations", and the number of individuals within a defined population.

The database has been designed in two levels: Local handling is performed over a MS Access database that implements an interface connected by ODBC protocol to the MySQL server; when the data is ready, it is transferred to HGVbase production database[3], which runs in MySQL on Linux[4].

### 2.2.3 Additional Environment

In addition to the small temporal local database (running in MS Access) and the HGVbase database (running in Linux), there exists a third database. It is called **Denormalized HGVbase,** and accommodates a snapshot of HGVbase at a certain

[3] See Appendix I for a definition of a production database.
[4] Linux is an operating system based on Unix, which is another operating system. Visit www.linux.com for further information.

moment. Its purpose is to be available in the server as a copy of HGVbase production database to be downloaded; in that way, two major problems are avoided:

1. No external users have direct access to the HGVbase production database.
2. No extra load is charged to the database server: Exporting a mirror of the database only involves file transfer, and not any kind of database query.

Denormalized HGVbase is also used as a source for MIAH's output function. For further information, go to section 3.5 on page 21.

# 3 MIAH: DESIGN

## 3.1 Before Start: Introduction to Object-Oriented Design

Object-Oriented[5] design is a method of designing software applications with a different approach. Instead of the classic structured programming or data-driven design, object-oriented software is all about objects. An object can be seen as an entity which has several attributes, and a way of communicating with its environment through sending and receiving messages. These messages define the interface to the object: Everything an object can do is represented by its message interface.

Thus, the aim of object-oriented design is to **encapsulate** private data and internal code by offering a common, public interface to provide access to it. An example can be seen in a coffee machine: It has several buttons (the interface) to provide different kinds of coffee; when you press the "coffee and milk" button, the machine receives your message, processes and executes it, and finally returns to you a glass and an auditory warning. So, to ask for coffee you need just a finger (a device to send a message to the machine) and a proper interface (several buttons with a description of their respective tasks): Internal details are solved by the machine without your involvement.

That's how OO works: When interacting with an object, you don't need to know the object's internal structure nor how the object works, just its interface. Consequently, complexity is managed using abstraction.

### 3.1.1 Classes

To create a new object you just need to define a class. A class determines everything about an object, while objects are individual instances of a class. Following our example, the class would be the abstract image of a coffee machine, defining how it is constructed internally and all the related messages it may act upon. An object would be "the coffee machine in the hall", which would instantiate the "coffee machine" class.

### 3.1.2 Methods and Attributes

A class definition includes attributes and methods of a certain type of object. Attributes are data related to the object stored within it without direct access. They can be public (for example, available types of coffee) or private (for example, the amount of coffee remaining). They are accessed through methods.

Methods, like attributes, can be public or private. A method is no more than code that is executed when somebody calls it; frequently, methods involve the use of attributes (the method "give_me_coffee_with_milk" would require consulting the attributes "milk_remaining" and "coffee_remaining"), but not always (a method to emit an auditory signal wouldn't use any attributes).

[5] Commonly referred as OO, see Abbreviations section.

### 3.1.3 In Summary

Object-Oriented design may be summarized as follows:

- A class is an abstraction for a design problem.
- Within the class, its attributes and methods are defined.
- Objects instantiate classes.
- Objects interact among themselves and with the environment.
- Objects can be composed of other objects (as in real life).

This method has several advantages:

- The overview increases. Consequently, modeling becomes simpler.
- Classes can be reused: This implies faster and cheaper development and maintenance.
- It is a powerful and elegant method of developing software.
- Distributing the coding task is easier.

However, drawbacks are also present. Changing from thinking procedurally to OO is a big learning task, and designing reusable classes is challenging.

Object-Oriented programming offers a new and powerful model for writing software. Although this chapter has mentioned the most important concepts, topics like inheritance, or polymorphisms, which are also interesting, are not included because they are not relevant to this report. If you want to broaden your knowledge of these and other concepts referring to Object-Oriented design, refer to the References section for several useful sources.

## 3.2 General Structure

The goal of MIAH is to present a solution for the problems related in section 1. Approaching them separately will make clearer and less complex solutions. That is why four modules have been created within MIAH:

- Interface Module
- SQL Translator Module
- I/O Support Module
- History Management Module

Figure 3.1 on the next page illustrates MIAH’s organization graphically.

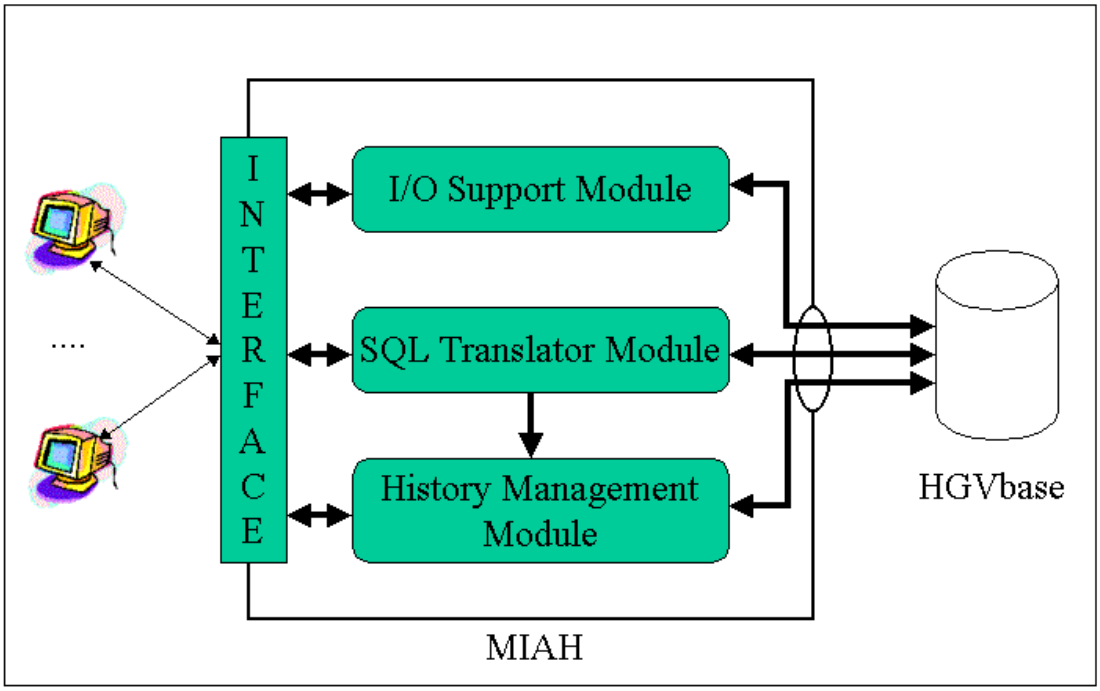


**Figure 3.1. MIAH's organization .**

These modules work together to provide the users with the same functionalities as before its implantation, but allowing the management team to take a more active approach to the database, optimising its use and content with minimum impact on users.

Only one module, the Interface module, interacts with users. This module distributes to the other modules the requests performed by users, and reports back appropriate feedback not only about the information requested, but also about errors that may have occurred during the execution.

The I/O Support Module provides import/export options: By supporting XML[6]-SQL translation, connectivity and usefulness of HGVbase are upgraded. Using this module users can request query results in XML format, or insert data in XML format into HGVbase.

SQL translator module supports the hottest SQL commands (like SELECT or CREATE TABLE) to allow users to query the database using standard syntax. Therefore, this module works closely with the History Management module. However, one of the standard SQL functions, the UPDATE command, must not be executed directly against the database; it must be managed properly to avoid deletions of data previous to the update by extracting, handling and properly storing historic data.

[6]A broader description of **XML** and **SQL** is given in Appendix II.

## 3.3 Interface Module

An interface is a software or hardware system that unrelated entities use to interact. The purpose of the Interface module is to properly bridge users with functions within MIAH, no matter which module they are located in. Consequently users will not distinguish between different modules because they will see MIAH as a **unit**. The interface takes care of their request by enrouting it to the correct module within MIAH and calling it correctly using the arguments supplied. In addition, the Interface module manages both the results from executed queries and any errors that may occur, and presents them properly to the user.

The Interface module interacts with all other modules within MIAH. In a theoretical approach, therefore, every function in every module has a mirror function in the Interface module to be called from the outside. This means that no user has direct access to any module but the Interface one. However, in a practical approach this is not completely true. For optimization reasons, finally the interface module has been merged with the SQL Translator Module to optimise interactivity between modules.

The design of this module answers one of the problems outlined in section 3.1: The non-existence of a definite and unique entry point to the database. Now, users can not see the database any more, only MIAH's interface. Thus, the database's structure and policies have been hidden from outside, reaching one of our objectives.

A description of the interface is available both electronically within the tool and as a User's manual.

## 3.4 SQL Translator Module

MIAH is basically an interface between the database and the set of users. To query the database, they must use the SQL language, the standard language accepted by the database engine, MySQL. Consequently, to grant the same access as before to users thus keeping consistency between the previous access system and MIAH SQL must be supported.

The SQL Translator module is in charge of executing the SQL queries requested by users against the database. Using a simple interface, this module takes the parameters supplied by the user and creates a valid SQL statement to be sent to the database. Once the database has processed it, the module provides the user with the result sent back from the database.

The SQL Translator module consists of a set of SQL language functions that are called from the Interface module. This set comprises almost all SQL functions and certainly all the most useful ones described in SQL standard. However, the addition of new functions if required is very easy due to the atomic character of functions within the module: each one is independent in the set, and can be removed, modified or upgraded without any side effect on the others. By extension, this also applies to new added functions.

The set of SQL functions has been enriched with a new, specific function: **select_all_from_HGVbaseID**. The task of this function is to extract all the information available within the database for a given HGVbaseID[7]. The inclusion of this function achieves two major goals:

1. Complex query (or join of subqueries) is no longer needed to perform an extraction of full data related to a HGVbaseID.
2. The structure containing the result of the operation is non-redundant, therefore saving useful –and sometimes vital– memory space and enhancing performance.

Furthermore, another advantage of this module is the automatic creation of valid SQL statements: By using statement templates, a considerable amount of frequent spelling mistakes and syntax errors from the SQL parser, which can be extremely annoying and time consuming, can be avoided. The SQL Translator module performs several checks on the parameters supplied looking for inconsistencies and errors, and returns feedback to the user if applicable.

## 3.5 I/O Support Module

This module has been included to support import/export operations on the database. In addition to queries on-line, HGVbase project also offers the possibility of downloading the whole database either as a mirror image of the production HGVbase database or as an XML translated package.

The XML translated package represents the possibility to download the full database in a format that, in addition to being non-redundant, machine-parsable and fully documented, can be interpreted directly by a very wide range of web browsers (including latest versions of Netscape Navigator and Microsoft Internet Explorer). This enhances HGVbase's value, and allows it to reach more potential users, now and in the future.

The export function is the principal feature of this module. It is HGVbaseID–focused due to design clauses: it returns XML referring to an HGVbaseID supplied as a parameter. This means that no other queries may be performed producing XML code. Far from being a disadvantage, this function fulfils a major need for HGVbase users, who often want to review all data related to a HGVbaseID.

As can be seen in figure 3.2 (on the next page), a user calls the export function through the interface supplying a HGVbaseID. This HGVbaseID is used as a key to query the Denormalized HGVbase[8] (storing HGVbaseID–focused data for easier and quicker access). The result of this query is a set of rows accommodating data related to a single HGVbaseID (for example, related alleles, frequencies, etc) in a redundant[9] way. Figure 3.3 on the next page illustrates it in a graphical way.

[7] See Appendix I for a definition of HGVbaseID.
[8] Definition and purpose of Denormalized HGVbase can be found on section 2.2.3 on page 4.
[9] See Appendix I for a definition of redundancy.

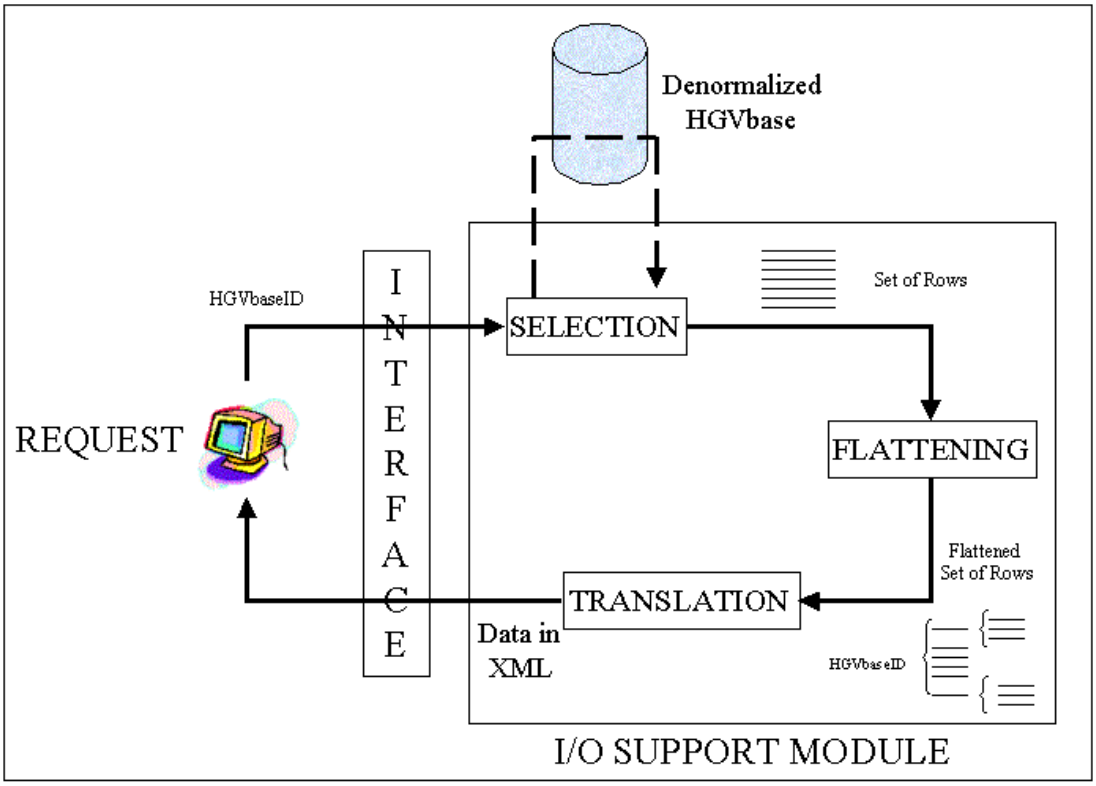


**Figure 3.2. Overview of how I/O Support Module works.**

Before the translation of the result into XML, redundancy on the result may be eliminated to allow the XML translator to work correctly. Consequently the redundant set of HGVbaseID–focused data obtained from the query is flattened into a non–redundant structure (refer to figure 3.3), and translated afterwards into XML. This result is then sent to the interface, which sends it to the calling user.

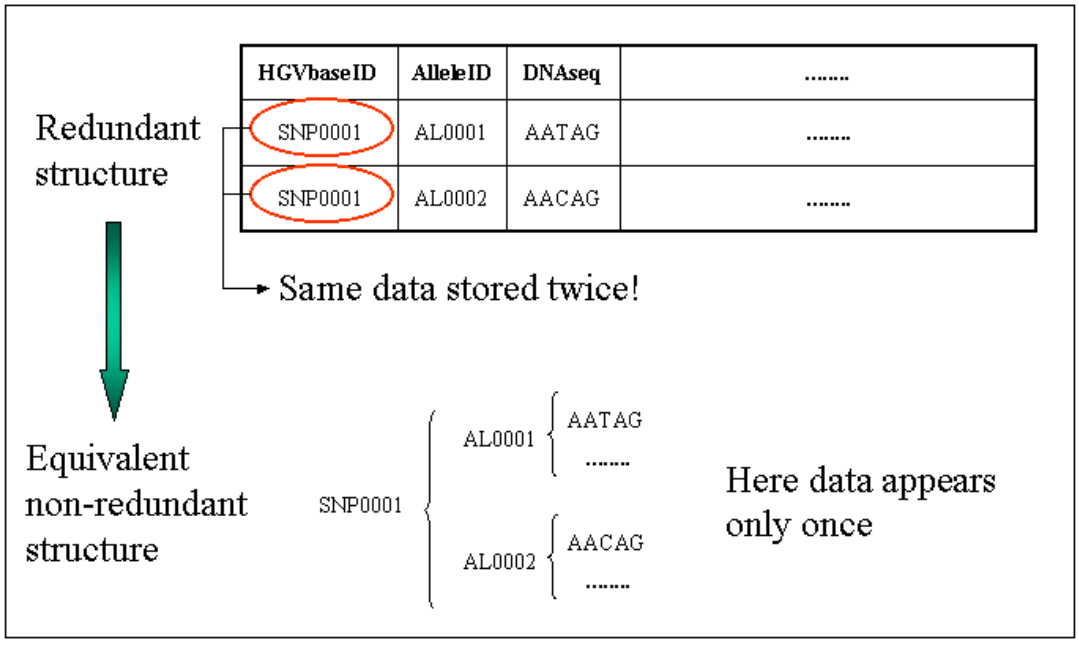


**Figure 3.3. Result from a SQL query against HGVbase, and two different ways to represent it.**

## 3.6 History Management Module

The History Management Module is in charge of tracking the history of the database: When a row is modified within HGVbase through a SQL "UPDATE" sentence, the old value is stored to allow access of it afterwards.

This utility offers two useful properties to the system:

- Protects the database against accidental modifications: Old values can be restored.
- Stores historical values: thus, information always remains within the database.

Basically, this module represents a call to the SQL "UPDATE" function, which has this behaviour:

**UPDATE TABLE table SET field1="newValue1", […], field2="newValue2" WHERE Key ="value"**

Consequently, a call to the "UPDATE" function is considered an atomic access by the History Management Module. The "UPDATE" statement presents some features that will be used to design the whole Module:

- Only one table in the same statement can be modified: therefore, any atomic modification can be done **just to one** table.
- A "WHERE clause"[10] may be used, in order to avoid a massive upgrade of all the rows in the table. Hence, the field used in the "WHERE clause" to address the target rows to be modified is also used as a key in the History Management Module to address the modification.
- An undetermined number of fields can be upgraded at the same time. Supposing that more than one will be realistic, the fields modified will be stored in a separate table to reduce redundancy (see below for further information).

Figure 3.4 on the next page illustrates an overview of the module and its internal organization. Note that the arrow going from the user to the module is **unidirectional**, this is, the user can only call the SQL "UPDATE" function, and receive feedback only about the correctness of it.

This leads us to an important point of this module: It has been designed to store the modifications in a small database of its own, but not to provide the user with an interface to browse this database. Such an interface may be part of a possible **future upgrade** for this module. In the current design, browsing through the Historical database must be done using direct SQL statements.

[10] WHERE clause is part of the SQL language. Refer to Appendix II for further details.

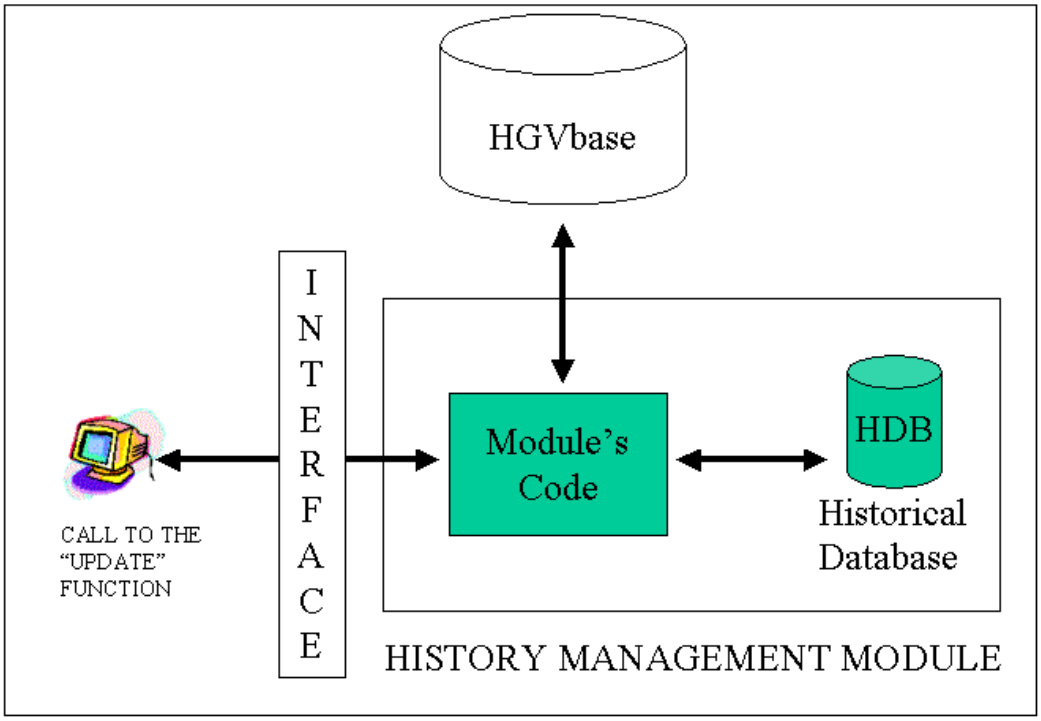


**Figure 3.4. Overview of History Management module.**

As can be seen in previous figure (figure 3.4), the module consists of an embedded database kept parallel to HGVbase in MySQL, and the code needed to manage it properly and to handle "UPDATE" requests from users. In order to explain in more detail how data is stored for future use, a quick overview of Historical Database is presented below.

In a nutshell, the Historical Database is a very simple database used to store old values from records present in HGVbase. It has two tables to keep the old values and metadata[11] about them to allow them to be addressed and linked properly. Figure 3.5 shows graphically the database structure as a Star Schema[12].

The Historical_Table is the main table in the structure. It stores the ID for modifications, the table in which the modification has taken place, and the date of the modification.

A second table is necessary because a single update may involve several fields.

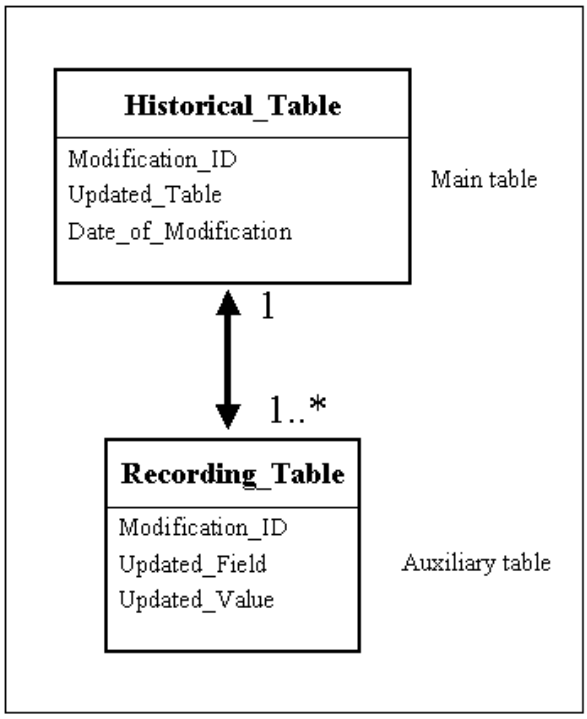


**Figure 3.5. Star-schema of the Historical Database.**

[11] See Appendix I for a definition of metadata.
[12] See Appendix I for a definition of a Star Schema.

To insert a set of modifications in a successful call to the "UPDATE" function, several steps are taken. Let's take next "UPDATE" statement as an example, to follow the procedure:

**UPDATE TABLE** Curators **SET** CuratorAddress="Sveavägen 15**",** CuratorPhone ="+46 123 45 67" **WHERE** CuratorID ="MJ"

Table Curators in HGVbase will be modified as next figure (figure 3.6) shows:

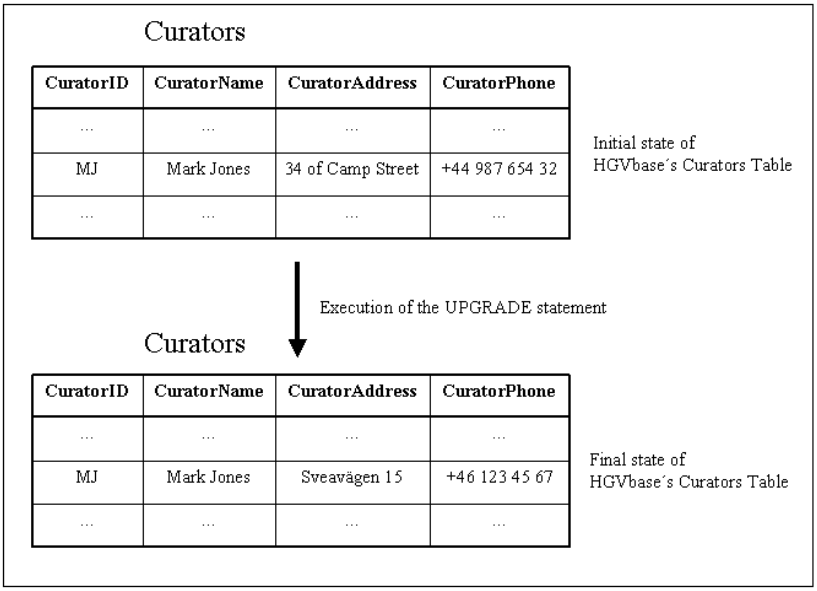


**Figure 3.6. Initial and final state of the table "Curators" during a call to the UPGRADE function.**

During the execution of the "UPDATE" statement, and before overwriting information in HGVbase, History Management Module extracts the fields to be overwritten, and inserts them properly into the Historical Database. Next figure (figure 3.7) shows how these old values are stored in the Historical database, continuing with our example:

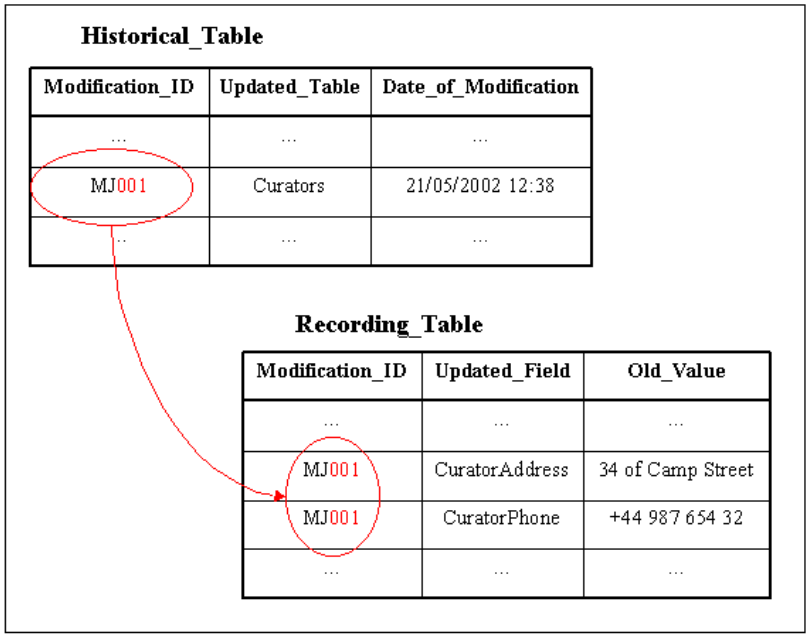


**Figure 3.7. How values are stored in the Historical database.**

Note that the Modification_ID is created by adding a counter as a suffix to the key used in the "WHERE" dause on the UPDATE statement. Thus, it is easier to track changes: updates of the same record will have the same root but a different counter suffix (see figure 3.8).

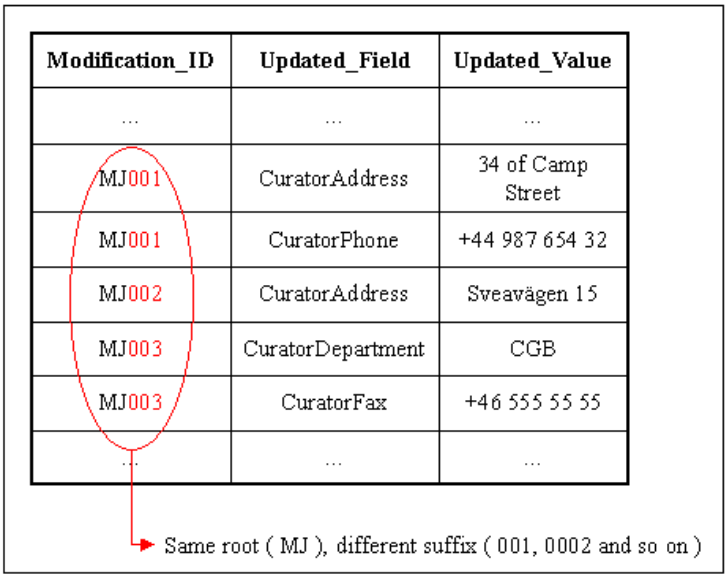


**Figure 3.8. Similarities and differences of several updates on the same HGVbase register.**

# 4 MIAH: IMPLEMENTATION

Although this report describes MIAH mostly on the level of design (which is useful for giving the essence of the solutions), some brief implementation details are also important to give a complete description of MIAH.

## 4.1 Programming Language Used

The language used to implement the required software was Perl, a versatile, easy, portable and free programming language. These two last reasons were crucial when making the decision, in addition to the large amount of code libraries[13] that are available through the Internet for it, highly valuable for any Perl developer.

During the implementation of MIAH some of these modules were used. This subsection lists the modules used and their use.

- Module DBI: Is a database interface module for Perl. It defines a set of methods, variables and conventions that provide a consistent database interface independent of the actual database being used.

- Module XML: Allows the use of XML in Perl applications by providing an efficient and easy way to parse XML documents.

- Module I/O: Is the module used to make the Input/Output in Perl easier and faster.

## 4.2 Running Environment

MIAH has been designed to work with "MySQL". MySQL is a relational management system that follows the main SQL syntax standard. However, some syntax rules differ from other SQL providers (like Oracle or Sybase), and migrating MIAH to these providers may require modifications of internal code.

## 4.3 Security Features

MIAH is not designed to serve as a security management tool. Actually, user and password by default are stored permanently within the structure, which allows direct connections to the production database[14] without explicit authentication and saves time and effort bypassing redundant re-authentications.

Consequently, security must be managed outside the tool, either by removing implicit authentication by the tool (probably needed in a distributed version) or limiting execution privileges on MIAH.

[13] These libraries are formed by a set of modules, each one performing a determined task.
[14] See Appendix I for a definition of a production database.

## 4.4 Test Suite

A very important and final step in the development of any software is the test suite. In it, the author or the team who implemented the software develops a set of tests, in addition to the code, to prove that everything within the program runs as expected.

MIAH has a set of tests to validate all the modules which compound it. Taking advantage of MIAH's modularity, each module has its own set of tests that can be run independently. This allows tests to be made during the development of each module in spite of the fact that others may not be implemented yet.

The pattern followed to carry out (and design) the tests goes from inside to outside:

1. Single functions are checked: Internal code, range of possible parameters[15], performance, optimisation and output are carefully considered.

2. Sets of related functions are checked: Interaction between them is carefully tracked.

3. The entire module is checked: Through its interface, the module's functionalities are called to prove, finally, that the module works perfectly.

4. Interaction between modules: Several tests are performed to check how modules communicate and relate to each other when necessary.

5. MIAH's functionality: In the final test the full tool is tested from the outside.

During this battery of tests, errors and flaws found are corrected and reported properly to give a good overview of MIAH's evolution.

[15] With a special attention on dangerous, special or boundary values.

# 5 CONCLUSION

Genetics is an outstanding field of science that will become one of the most important fields of the 21st Century due to the relevance and impact of its investigations. New technologies allow genetic scientists to study in great detail many things about how we are made: cloning and cures for many diseases are only two examples of break-throughs achieved in the last years in genetics. These discoveries always lead to new investigations, which go a step beyond, thus closing the circle of science.

However, we have arrived at a point where human limitations are critical in current and future investigations. To advance in knowledge of what constitutes life, scientist have to rely, not only on specific devices such as powerful microscopes, but also on general knowledge handling technology.

Computer support is such an example of vital technology used in science. Currently, computers are used in every project to support scientists in their experiments, not only for modeling reality, but also to furnish the scientist with an invaluable 'behind the scenes' help. By using computers, scientists can dedicate more time to research, using computers for mundane repetitive tasks.

HGVbase was designed for this purpose: By keeping a comprehensive account of polymorphisms, genetic scientists all over the world will save precious time during their research by quickly accessing accurate information on HGVbase.

This project was intended to go a step beyond. MIAH was developed to upgrade interconnectivity between HGVbase users and HGVbase. By reviewing its weak points, and providing a good solution for them, HGVbase functionality may be upgraded to serve its users better.

MIAH, then, provides several upgrades to the HGVbase environment:

- Database implementation details have been hidden behind MIAH: Now, users interact with MIAH's interface to use the database.

- Access to the database has been unified. Now users and applications using the database go through the same point: MIAH's interface.

- A Historical Management system has been designed. Now, the history of database records can be successfully tracked.

The main problem during the realization of this Thesis has been the search for the correct features to implement within MIAH. Sometimes it was very difficult to clarify what the user' s needs are, and much time was spent looking for the right and useful features to be implemented.

MIAH has not been designed to be a static tool. Indeed, some extensions for it are quite obvious and useful. The first one may be the adaptation of MIAH to work over the Internet, accepting and serving requests from a web form or web page. Then, a careful design of a friendly interface to serve as an entrance to the database through MIAH will

turn out as an improvement of HGVbase usefulness. Another extension may be the upgrade of the History Management module by providing it with a better interface. Currently it must be used through the common MIAH interface, limiting its functionality because most of it was designed for computer applications. Thus, a better interface to access the module will make its use easier and more convenient.

HGVbase is a useful tool that will help genetic scientists to make progress in their research. By making upgrades such as MIAH, we are not only helping HGVbase to grow and develop faster, but also helping the community of scientists to obtain a better understanding of genetics, and consequently, of human nature!

# 6 REFERENCES


[ 1 ] Brookes, Anthony J., Lehväslaiho, Heikki, Siegfried, Marianne, Boehm, Jana G., Yuan, Yan P., Sarkar, Chandra M., Bork, Peer & Ortigao, Flavio. 2000. **HGBASE: a database of SNPs and other variations in and around human genes**. In *Nucleic Acids Research*, 356-360, vol. 28, No. 1. Ed. Oxford University Press.

[ 2 ] **Cambridge Dictionary Online**. [online] in *Cambridge University Press.* Available: http://dictionary.cambridge.org/

[ 3 ] Castellano, J.G. **Tutorial de DBI**. [online] in *Universidad de Granada.* Available: http://geneura.ugr.es/~javi/dbi/index.htm

[ 4 ] **Center For Genomics and Bioinformatics**. [ online ] in *CGB.com*. Available: http://www.cgb.ki.se

[ 5 ] Conway, Damian 1999. **Object Oriented Perl**. Ed. Manning Publications Co.

[ 6 ] **Definitions for the most current IT-related words**. [ online ] in *Whatis.com*. Available: http://whatis.techtarget.com/

[ 7 ] Flynn, Peter. 2002. **The XML FAQ**. [ online ] in *University College, Cork.* Available: http://www.ucc.ie/xml/#acro

[ 8 ] Fredman, D., Siegfried, M., Yuan, Y.P., Bork, P., Lehväslaiho, H, & Brookes, J. 2002. **HGBASE: A human sequence variation database emphasizing data quality and a broad spectrum of data sources**. In *Nucleic Acids Research*, 387-391, vol. 30, No. 1. Ed. Oxford University Press.

[ 9 ] **Free On-Line Dictionary of Computing**. [ online ] in *The Imperial College Department of Computing.*
Available: http://foldoc.doc.ic.ac.uk/foldoc/index.html

[ 10 ] **Glossary of Genetic terms**. [ online ] in *The National Human Genome Research Institute.*
Available: http://www.nhgri.nih.gov/DIR/VIP/Glossary/pub_glossary.cgi

[ 11 ] Gressly, René. 2000. **An Introduction to the Java Technology**. [ online ] in *Gressly Systems.*
Available: http://www.gressly.ch/systems/download/Introduction1.4.pdf

[ 12 ] **Introduction to SQL.** [ online ] in *W3Schools.com.*
Available : http://www.w3schools.com/sql/sql_intro.asp

[ 13 ] **Introduction to XML**. [ online ] in *W3Schools*.
Available: http://www.w3schools.com/xml/xml_whatis.as

[ 14 ] **Karolinska Institutet**. [ online ] in *KI.com*.
Available: http://www.ki.se/

[ 15 ] **Manual de Perl**. [online] in *Universidad de Oviedo*.
Available: http://www.etsimo.uniovi.es/perl/tutor/

[ 16 ] **Manual de SQL en Español**. [ online ] in *Webexperto*. In Spanish.
Available: http://www.webexperto.com/manuales/sql/index.asp?capitulo=1

[ 17 ] Montlick, Terry. 1999. **What is Object-Oriented Software?**. [ online ] in *Catalog.com*.
Available: http://catalog.com/softinfo/objects.html

[ 18 ] **MySQL: the most popular open source database**. [ online ] in *MySQL.com*.
Available: http://www.mysql.com/

[ 19 ] Rios, Daniel. 2002. **RIOSNPS: An automated validation tool for HGVbase**. KTH.

[ 20 ] Schwartz, Randal, Olson, Erik, Christiansen, Tom. 1997. **Learning Perl on Win32 Systems**. Ed. O'Reilly.

[ 21 ] **SQL Interpreter & Tutorial**. [ online ] in *SQLCourse.com.*
Available : http://www.sqlcourse.com/intro.html

[ 22 ] **The DBI Interface**. [online] in *MySQL Official Webpage.*
Available: http://www.mysql.com/doc/P/e/Perl_DBI_Class.html

[ 23 ] **The language center**. [ online ] in *Merriam-Webster online*.
Available: http://www.m-w.com

[ 24 ] **The monastery gates: Perl Monks**. [ online ] in *Perl Monks.com.*
Available: http://www.perlmonks.com

[ 25 ] **The source for Perl**. [ online ] in *Perl.com*.
Available: www.perl.com

[ 26 ] Torkington, Nathan. 1999. **Pragmata II**. [online] in *Zdnet.*
Available: http://www.zdnet.com/filters/printerfriendly/0,6061,2377498-84,00.html

[ 27 ] **XML Basics**. [ online ] in *Software AG*.
Available: http://www.softwareag.com/xml/about/starters.htm

[ 28 ] Wall, Larry, Christiansen, Tom, Schwartz, Randal L. 1996. **Programming Perl**. Ed. O'Reilly.

# APPENDIX I. DEFINITIONS

## A) Genetic Terms

[ 1 ] **ALLELE**: One of the variant forms of a gene at a particular locus, or location, on a chromosome. Different alleles produce variation in inherited characteristics such as hair color or blood type.

[ 2 ] **DNA**: The chemical inside the nucleus of a cell that carries the genetic instructions for making living organisms.

[ 3 ] **GENE**: The functional and physical unit of heredity passed from parent to offspring. Genes are pieces of DNA, and most genes contain the information for making a specific protein.

[ 4 ] **GENOME:** All the DNA contained in an organism or a cell, which includes both the chromosomes within the nucleus and the DNA in mitochondria.

[ 5 ] **GENOTYPE**: The genetic identity of an individual that does not show as an outward characteristic.

[ 6 ] **PHENOTYPE**: The observable traits or characteristics of an organism, for example hair color or the presence or absence of a disease. Phenotypic traits are not necessarily genetic.

[ 7 ] **POLYMORPHISM:** A common variation in the sequence of DNA among individuals.

[ 8 ] **SNP**: Acronym for Single Nucleotide Polymorphisms. These are common, but minute, variations that occur in human DNA at a frequency of one over 1.000 bases. These variations can be used to track inheritance in families.

## B) General Terms

[ 1 ] **API:** An application program interface (API) is the specific method used by a computer operating system or by an application program by which a user can make requests of the operating system or another application.

[ 2 ] **BLESS**: An action performed in some programming languages for which the blessed target changes its type, becoming a new instance of the blessing type. For example, in Perl you can bless an array as a certain object: From then, the array is no more an array but a instance of the object it has been blessed to.

[ 3 ] **DATABASE:** A collection of data that is organized so that its contents can be accessed, managed and updated.

[ 4 ] **HGVbaseID:** A unique ID in HGVbase that refers to an unique polymorphism within HGVbase.

[ 5 ] **JOIN:** A relational database operation which selects rows from two (or more) tables such that the value in one column of the first table also appears in a certain column of the second table.

[ 6 ] **KEY:** A value used to identify a record in a database, derived by applying some fixed functions to the record.

[ 7 ] **METADATA:** Term used in database environments to refer to data that provides information about or documentation of other data (i.e. size, position, etc) managed within an application or environment.

[ 8 ] **ODBC:** Is an open standard application programming interface for accessing a database.

[ 9 ] **PRODUCTION DATABASE**: Database that stores real data. This term is used as opposed to the term Development Database, which is a database storing dummy data used to perform tests.

[ 10 ] **REDUNDANCY:** Referring to a state in where something is unnecessary because it is more than is needed.

[ 11 ] **STAR SCHEMA:** Data model used in databases for which a main table connects to several secondary tables, thus creating a star-like diagram called Star Schema.

# APPENDIX II. IMPLEMENTATION EXTENSIONS

This section tries to cover several gaps concerning the implementation left out during the report, through the explanation of some concepts, techniques and languages used to materialize the design ideas and solutions specified.

## II.1 PERL: An Overview

Perl plays a very important role within MIAH: It is the programming language used to implement and run it. That is why the following subsection gives the reader a brief overview about Perl and its functionalities, to allow the reader to take advantage just in case he/she would like to take a look on MIAH´s code or run a Perl program him/herself.

### II.1.1 Description

Perl, which is an acronym for Practical Extraction and Report Language, is an interpretive[1] language developed –and maintained– by Larry Wall especially for processing text. Intended to be practical rather than beautiful, it combines some of the best features of *C* –its syntax and C syntax corresponds quite closely–, *sed*, *awk* and *sh*, so people familiar with these languages should have little difficulty with it.

Another factor that makes Perl very popular is that it is distributed under the GNU license[2], which means not only that is freely available but has an online code repository[3] where many free modules covering a wide range of issues can be found to be used easily in your programs, thus saving time and effort.

In addition to these features, Perl is also portable to many different platforms like Unix, DOS and even Windows with minor modifications.

In a nutshell, Perl is free, easy to understand, well documented, powerful, robust, flexible, little constrained, and is constantly being improved. Those are the reasons why it has been chosen as one of the standard programming languages in the CGB.

Perl also allows you to implement object-oriented programs, although it was not designed in such a way. Thanks to some modifications in the original package, Perl provides the user with two methods to develop software: The classic structured programming[4] method or the Object-Oriented method.

[1] It means that programs run through a dataflow tracing mechanism, which prevents many undesired situations.
[2] See the Perl Homepage, www.perl.com .
[3] This repository is called CPAN, and is located at www.perl.com/CPAN.
[4] Also called Imperative programming.

Section 3.1 in this report introduced OO principles. Perl follows them with its own nomenclature:

- Classes are named *Packages*.
- Methods are called *subroutines*.
- Attributes remain attributes.
- To create an object, you may bless[5] a reference. It will be used to access the object's attributes and methods.

Next figure (figure II.1) illustrates these principles:

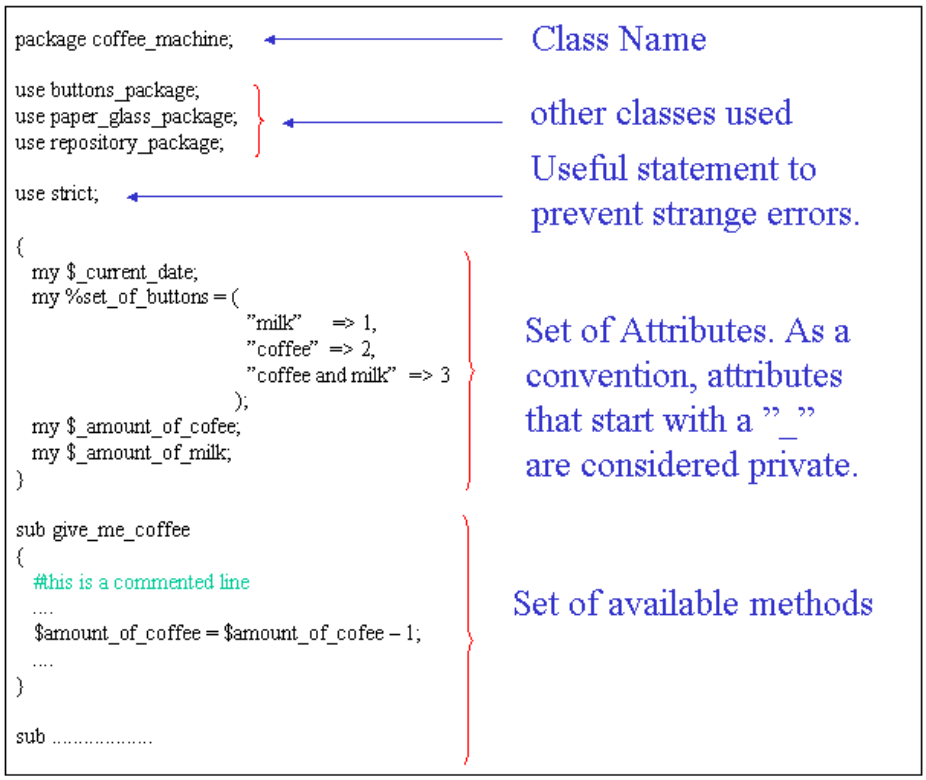


**Figure II.1. An example of Object Oriented Perl code.**

# II.2 SQL: An Overview

HGVbase is mounted over a database management system called MySQL. This system uses a standard language called SQL as an interface between database users and the database engine. Consequently, to query HGVbase MIAH must use SQL. That is why a lot of terms referring to SQL appear during the report, and why the following section gives a brief overview of the basics of SQL.

[5] See Appendix I for a definition of a blessing.

## II.2.1 Description

SQL[6], which is the acronym for Structured Query Language, is a standard language for accessing and managing relational database systems. SQL statements are used to perform tasks such as retrieving data or updating data in a database. This flow of information between the user and the database using SQL is similar to that illustrated in Figure II.2.

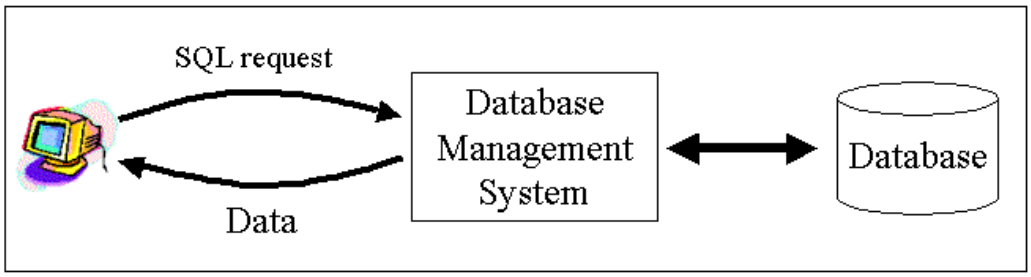


**Figure II.2. The flow of commands and data between the user and the database.**

Some common database engines that use SQL are: Oracle, Sybase, Microsoft SQL Server, Access and many others. Although they use the SQL standard syntax, unfortunately most of them also have their own proprietary extensions to the language that may cause confusion to users jumping between them.

The data contained by the database is stored as records in a set of tables. Tables are identified by descriptive names (like "Employees" or "Customers") and are divided into columns and rows. Rows contain records (like one record per customer or product) and columns contain related data (like "first name", "address", or "price"). Figure II.3 illustrates an example using a table called "Customers":

| Full Name | Address | City | Account Number |
|---|---|---|---|
| Johansson, Gudrun | Sveavägen, 15 | Stockholm | 0046 125 4568 |
| Sveaborg, Pia | Proffessorslingan, 6 | Umeå | 0046 198 4532 |
| Glick, Marcus | Main Street, 124 | New York | 0001 11 45875 |

**Figure II.3. Table "Customers" with four columns and three rows containing records of three customers.**

SQL gives the user full access to the database. Using SQL statements, all kind of operations relating the database such as queries or updates can be performed. However, the user must know the syntax because the interface is not graphical but in command line mode.

[6] Pronounced "ess-que-el".

The most important SQL functions and their syntax are enumerated below[7]:

1.- SELECT: Extracts data from a database.

**SELECT** fields **FROM** table **WHERE** constrains **ORDER BY** criterium

2.-UPDATE: Updates data in a database.

**UPDATE** table **SET** Field1=Value1, Field2=Value2, ... FieldN=ValueN
**WHERE** constrains

3.- DELETE: Deletes data from a database.

**DELETE FROM** table **WHERE** constrains

4.- INSERT: Inserts new data into the database coming from:

- Outside the database:

  **INSERT INTO** table (field1, field2, .., fieldN) **VALUES** (value1, value2, ..., valueN)

- Inside the database as a result of a nested SELECT statement.

  **INSERT INTO** table **SELECT** fields **FROM** SourceTable

5.- CREATE TABLE: Creates a new database table.

**CREATE TABLE** table ( Field1 type (length), Field2 type (length), ... )

6.- DROP TABLE: Deletes a database table.

**DROP TABLE** table

[7] Note that bold words represent reserved words within SQL syntax.

# II.3 XML: An Overview

XML is the acronym for eXtensible Markup Language. XML is a standard, simple and self-describing way of encoding and structuring data and text so they can be exchanged across diverse hardware, operating systems and applications, and used within a wide range of development tools and utilities.

Similar to HTML[8], XML is designed to improve the functionality of the web filling the gaps in HTML by providing more flexible and adaptable information identification. In fact, it is called extensible because its format is not fixed like HTML: Actually XML is a "metalanguage" [9] which lets you design your own customized markup languages for limitless different types of documents.

For this reason, XML was chosen to be the language used for exports and imports over HGVbase: It can be used to store or enclose any kind of structured information just what we need when exporting/importing data in order to pass it between different computing systems. For example, most web browsers, as well as Perl[10] support XML.

After reading this subsection, some strong points may remain in your mind about XML:

- It stands for eXtensible Markup Language.
- It is a markup language very similar to HTML.
- It has been designed to describe data.
- Its tags are not predefined. You must define your own tags.
- It has been designed to be self-descriptive.

[8] Hypertext Markup Language. Indeed, XML and HTML come from the same language.
[9] A language for describing other languages.
[10] The programming language used for implementing MIAH.