Data Modeling and Data Communication in GenomEUtwin

Jan-Eric Litton[1], Juha Muilu[2], Ann Björklund[1], Anne Leinonen[2], and Nancy L. Pedersen[1]

[1] *Department of Medical Epidemiology and Biostatistics, Karolinska Insitutet, Stockholm, Sweden*
[2] *Finnish Genome Center, Biomedicum Helsinki, University of Helsinki, Finland*

Database infrastructure has become a critical component for competitive life sciences research and discovery. The explosion of data requires that the data are properly loaded, accessed, managed, queried, analyzed, and shared with others. The key purpose of the population-based twin cohorts housed at different institutions in Europe is to gather an extremely large quantity of information from their twin populations, and share it. Longitudinal research over a long period of time, hopefully generations, demands completely new methods and systems to handle the gathering of information and storing. These cohorts bring to the fore problems concerning the need for a standardization of research data and a computer and storage strategy. In the following we describe the preliminary strategy being implemented in the Database Core of GenomEUtwin.

Strategy

The workflow and integration between the phenotype and genotype databases are shown in Figure 1. At present the Genotype database is being built in Helsinki and the phenotype database in Stockholm using Oracle as the database engine. However, this is only the first step for implementing the format and variable standard and data communication used in GenomEUtwin. We are heading toward a distributed model using "middleware" software between different countries for federation of our local databases. The data can be either collected into a single common database or directly federated from the source database through the core schema. Both options can be synchronized by utilizing a separate extract-transform-load processes of local replicas, where the local database replicates can reside in a demilitarized zone. Access to the whole dataset can be limited using built in security features existing in relation database management systems. Data are only visible throught restricted views which filter out rows in a table the user has no right to use or see. We are also exploring possibilities to access data directly from the source database. This provides a way to get up-to-date information for the studies and can be used to validate and check consistency.

Common Variable Format and Standard

In order to facilitate communication among the various centers, one of the first steps is to establish a common variable format and standard.[1] Another key aspect is designing a unique identifier for all individuals that may potentially be included in the database. By applying the same variable names and value formats to variables in common to all databases, several advantages will be accomplished. Firstly, misunderstandings and mistakes will be fewer and the programming will be more efficient. Thus, doctoral students will learn programming more quickly and key persons, such as database managers, programmers and statisticians, can be used more efficiently, with less time spent on deciphering inconsistencies in coding styles. Data accuracy will increase, as will the comprehensibility of data. In all, this standardization will enable a smooth transfer of datasets between co-workers and facilitate the planning, analysis and archiving stages of GenomEUtwin.

In the long run, documentation and archiving of data will be much improved. Databases will be more useful for research in the future when information about the data is less dependent on the actual collectors. This project will focus on a computer and storage strategy allowing our twin data to be organic, so that they can grow as our needs grow and adapt as our needs change. This means we need to focus on the underlying architecture of how data will move, how it will be stored, and how it will be accessed.

A Phenotypic Database Prototype

The GenomEUtwin database core decided to build a prototype of a common twin database. At the beginning the database will consist of only a limited number of key variables, including the identifying number, date of birth, gender, date of death, zygosity, birthweight, birth length, weight and height. These items may appear simple and non-controversial; however, there are a number of considerations important to each variable that illustrate the challenges in creating databases for international, multidisciplinary collaboration.

The European Twin Identifier

Unique identification of common data is the most important matter in data integration. The identification is not trivial because of different conventions and requirements.

Address for correspondence: Jan-Eric Litton, Department of Medical Epidemiology and Biostatistics, P.O. Box 281, Karolinska Insitutet, SE-171 77 Stockholm, Sweden. Email: Jan-eric.Litton@mep.ki.se

https://doi.org/10.1375/twin.6.5.383 Published online by Cambridge University Press

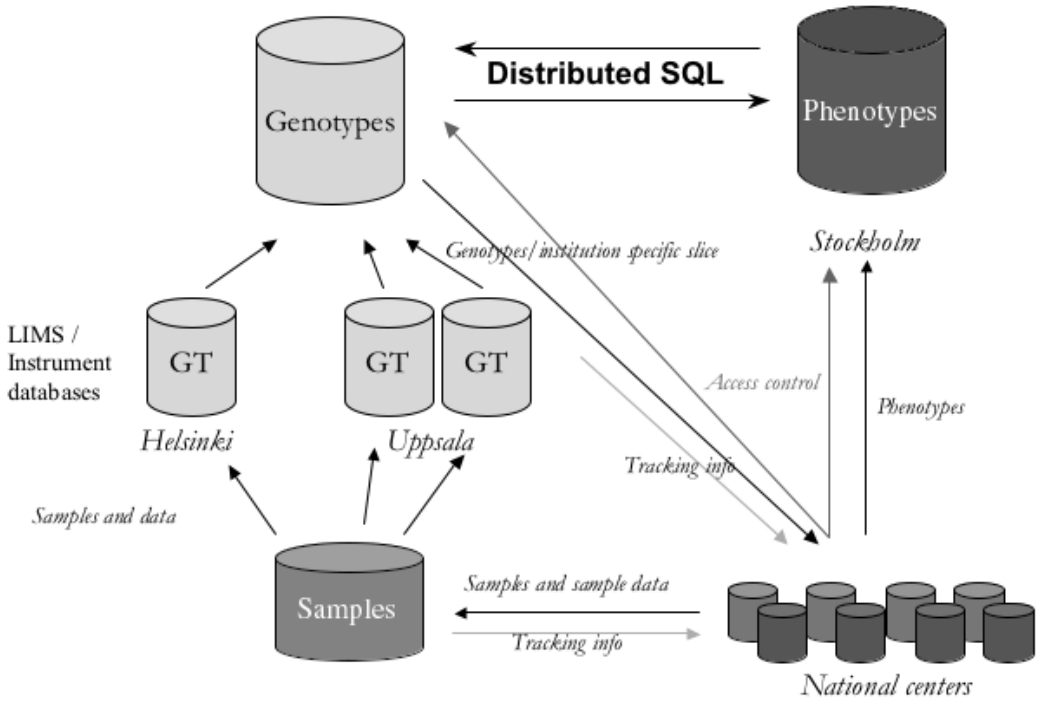

Figure 1

Illustrates the information flow between the national centers, DNA extraction unit and genotype and phenotype units. Besides the substantive data (black arrows) information on processing of samples and genotyping results will be sent back to national centers (light grey arrows). The national centers control the visibility of data accessed for a common study (grey arrows).

The database programmers do not like to code meaning into IDs because it is difficult to change the assigned identifiers once the data are stored. End users on the other hand, like to include key information to be able to "eyeball data".

The GenomEUtwin collaboration is based on cohorts that are ascertained using twins as the focus, not families. Most of the European twin registries do not have genotypic or phenotypic information from non-twin individuals. Some do, and GenomEUtwin will want to take advantage of those samples. Other registries will want to consider collecting data from non-twin siblings, other multiple births, and other biological family members. The identifier should be unique for each individual in the sample. It should be based not only on what country, institution, and cohort one belongs to, but also on the index pair that was the basis for being included in the study.

In some countries, there will be multiple twin pairs from the same family. However, a given twin center may not, at least at this point, know which twin pairs are related to which. This is another reason why the key relationship information cannot be a family identifier. Because twin pairs are the index unit, we will have some cases where, for example, an individual is included initially because she or he is ascertained as a twin, but this person may also be a parent to another set of twins. The identifier for a person should be based on his own initial ascertainment scheme, and not a function of relationship. Each registry will have to check that the file consists of unique individuals, although in some cases we may not know this until the genotyping is done.

The unique European twin identifier — the EUid-number, is a 12-digit number. Relationships will be stored in separate tables, allowing multiple relationships for one person. The advantages of this construction are uniqueness for each individual in the database and facilitation of "eyeballing" twins, triplets or quadruplets. The EUidnumber consists of four parts:

- Country code 3 digits – using ISO 3166 standard
- Randomized number 7 digits
- Identification number 1 digit
- Check sum 1 digit

Randomized Number

In the database there will be twins and non twins. Members of multiple births (twins, triplets, or quads) will all share the same randomized number, but be uniquely identified by the identification number following the randomized number. Thus, each twin pair will share the same randomized number whereas a non twin will receive their own randomized number. This will be followed by an Identification number, indicating whether or not this is a twin. The potential values are: 1 = Twin 1, 2 = Twin 2, 3 = Triplet, 4 = Quadruplet. Singleton births are indicated with a 0 (zero).

Checksum

For data security and accuracy, we have chosen to apply an extra digit (checksum) that assures that assignment of identification numbers is correct. The checksum is calculated by

https://doi.org/10.1375/twin.6.5.383 Published online by Cambridge University Press

GUMM algorithm.[2] The GenomEUtwin Data Transfer Java application can be used to calculate GUMM check digits.

GenomEUtwin Data Communication

Key aspects of data communication within GenomEUtwin require common definition not only of variables and variable types, but also characteristics of the files. In both the phenotypic and genotypic databases, a number of key characteristics are standardized:

Principles for Missing Data

These different types of missing data are as follows:

- Irrelevant, non-participant, non-response: Used when data are irrelevant for a person either due to being not included in study base or due to non response to questionnaire/interview as a whole.
- Irrelevant, structural: Used when data are irrelevant in the context.
- Do not know: Used when the respondent explicitly states it.
- Unknown: Used when no answer is given to specific item, or when no data are available.

Error Codes for Dates

At least some of the date variables will have missing data due to some of the four reasons above. The easiest way to code the missing data is to use sequences of 9s, 8s etc. However, most database systems do not take these codes as valid for dates, which means that it is impossible to insert such a code in the database. To avoid this problem, an error code variable after each date variable is added.

Variable Names

Each variable will have a unique name in the database and in the transfer files. It is important that all GenomEUtwin users use the same variable names in the database and transfer files. The variable names should be as short as possible without losing its description of data. The maximum length is 18 columns. The variable names should, as far as possible, explain what data are in the variable. Thus, anonymous naming is forbidden.

File Format

The ideal file format for the phenotype and genotype data should be based on XML (eXtensible Markup Language) because it allows representation of the data in a structured and standard manner. In the preliminary study we decided, however, to start with simple semicolon-delimited format. This is because of a lack of existing XML specifications for the genotype and phenotype data. It was decided that it was too early to try to create structured XML formats which would be likely to change afterwards. XML would now bring in unnecessary complexity since the simple semicolon-delimited format can be easily created from the tools and databases available on each site.

File Transfer

To simplify tracing of data transfers between GenomEUtwin centers we use a standardized file naming convention as follows:

GT_country_centre_cohort_YYYYMMDD_NN, where GT is standard to all transfer within GenomEUtwin, YYYYMMDD is the year, month and date when the file was created and NN is the sequential number of a transfer.

We have developed a GenomEUtwin Data Transfer Java application that can be used to analyze and prepare data files.

File Encryption

To avoid unauthorized access to data, all files are encrypted with PGP[3] prior to transfer.

GenomEUtwin Prototype Phenotype Database — Table Description

The relational model uses the basic concept of a relation or table. The columns or fields in the table identify the attributes such as name, age, and so forth. A tuple or row contains all the data of a single instance of the table such as a twin person. In the relational model, every tuple must have a unique identification or key. The database model is seen in Figure 2.

To make the database as useful as possible, even a long time ahead, it is necessary not to limit the database to store only certain selected variables. Because of this the database structure can appear complicated at first sight. Nevertheless, the advantage with this structure is that it is possible to store completely new variables as soon as they appear without changing the database structure.

The database is divided into two modules, phenotype and import. In the module "phenotype" the correct data are stored. The module "import" is used to store temporarily the data from the centers while the data are error-checked. If the data are correct they are inserted into the tables in the module "phenotype".

Phenotype Module

Fixed_data

The table *Fixed_data* contains the immutable (non-changeable) information stored for the twins/relatives in the database. With non-changeable information we mean information that does not vary with time. For example, the variable *Birthdate* has the same value regardless of when it is collected or evaluated. The *county* is where the data for a particular person is collected. The *randomnum* is the same unique number as in the *EUidnum. Idnum* indicates if the person is twin #1, twin #2, triplet, quadruplet or even a non-twin. *Birthdate_error* and *Death_date_error* are variables that indicate if these dates are correct or not.

Variable_type

The names of all the changeable variables in the database are stored into the *variable_type* table. No answers are stored, only the variable's full name and variable name. The variable *weight_self* has the following structure: WEIGHT_SELF refers to self reported weight which may have a value between 1-800 kilos, 990 refers to do not know, 999 to unknown and blank/NULL to irrelevant, not participant non-response.

An example row in the variable_type table may be seen in Figure 3.

https://doi.org/10.1375/twin.6.5.383 Published online by Cambridge University Press

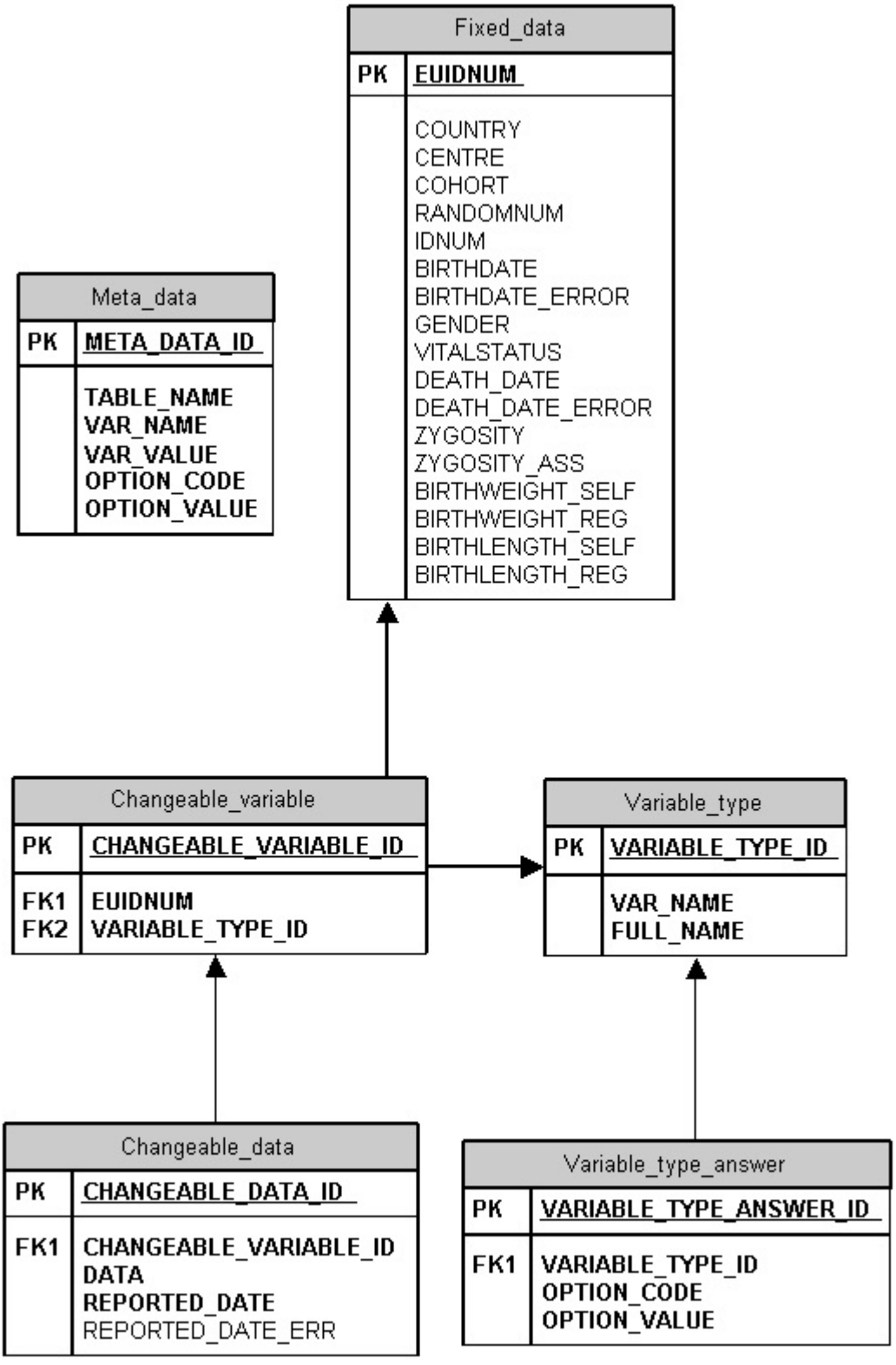

Figure 2

Data model Phenotype Database, module Phenotype.

https://doi.org/10.1375/twin.6.5.383 Published online by Cambridge University Press

VARIABLE_TYPE ID	VAR_NAME	FULL_NAME
379	Weight_self	Weight – self reported

Figure 3
Example row from the table variable_type.

VARIABLE_TYPE_ANSWER_ID	VARIABLE_TYPE_ID	OPTION_CODE	OPTION_VALUE
79	379	1-800	Weight in kilos
80	379	998	Do not know
81	379	999	Unknown
82	379	Blank/NULL	Irrelevant, non-participant, non-response

Figure 4
Example row from the table variable_type_answer.

CHANGEABLE_VARIABLE_ID	EUIDNUM	VARIABLE_TYPE_ID
567	75200021210	379

Figure 5
Example row from the table changeable_variable.

CHANGEABLE_DATA_ID	CHANGEABLE_VARIABLE_ID	DATA	REPORTED_DATE	REPORTED_DATE_ERR
275	567	88	20020512	1

Figure 6
Example row from the table changeable_variable.

Variable_type_answer

The "answers," values, or alternatives that belong to the variables in table *variable_type* are stored here. The row for the same example variable as above may be seen in Figure 4.

Changeable_variable

This table stores which questions the individual has answered. For example, the twin with EUIDNUM 752000021210 and self reported weight 88 kg will have the row in the table as seen in Figure 5 (Note: the value in *changeable_variable_id* is just an example).

Changeable_variable_data

In this table the answers to the changeable variables are stored. The row for the twin in the example can be seen in Figure 6. (Note: the value in *changeable_data_id* is just an example).

Because of the variable *reported_date* it is possible to follow a person's history for a variable.

https://doi.org/10.1375/twin.6.5.383 Published online by Cambridge University Press

Meta_data

The purpose of the *Meta_data* table is to describe the tables in the database. For many accustomed to less rigorous descriptions of databases, this table represents the "data on the data", or in general the documentation of the various tables in the database.

Import module

Import_data, error_data, error_code

The table *import_data* is used to import the data from the centers (Figure 7). The data are inserted into the table as it is delivered, no error checks are done. When the data are imported to *import_data* a SQL-script is run to manage the error checking and insert data to the tables in the phenotype module. If the script discovers any errors in a row, the row is not inserted into the phenotype module. Instead, an error code and the EUidnum are inserted into the table *error_data*. In this way, the database administrator can easily see which twins/relatives do not have correct data and contact the responsible center to get the correct data.

Genotype Database

The genotype database is designed to store results for different types of genetic markers (alleles and genotypes) and samples. The main focus is on high quality and integrity of the data produced at one or many genotyping centers. The database is not designed in the first place for fast analytical queries. The analysis requirement is typically addressed by separate problem centric data warehouses, which are partially preprocessed and transformed with external data for fast online query response. In these kinds of databases data are often denormalized and precalculated (aggregated over some attributes). In GenomEUtwin we will explore different ways to create the data warehouses or study specific "data marts". This is partly necessary already because data are distributed over different centers as shown in Figure 1.

The database structure is kept lightweight and modular, trying to capture just the essential consensus information available in the existing database schemata the authors were aware of.[4–9] None of the existing systems were chosen because they had data structures designed for other purposes or were too method specific. It was decided to create a new database which was suitable for different genotyping methods and which provided orthogonal and minimalist structures for extensions. The development work is done under an open source project and the source code is freely available from the website.[10]

Currently, two modules are implemented: the core module captures the minimum information needed for gene mapping studies and the annotation module provides tables for quality control and assessment. It is possible to extend the database further in a consistent manner to include instrument and laboratory specific data if needed later. The database will be implemented in Oracle 9i, but can be exported easily to other relational database systems.

Core Module

The core module contains tables for basic analysis data like genotype, marker, sample and subject as shown in Figure 8.

The lines indicate relationships between entities and numbers cardinalities of the relationships. In the database there are zero to many (0..*) samples from a single subject and one to many (1..*) alleles (typically two) from the same genotype measurement over genetic markers and samples.

The genotype contains identified sequence variation, alleles, measured from a DNA sample at a specific marker locus. The identifier of allele can be for example name of SNP or size of microsatellite repeat, it just is a unique "name" for the variation. Same allele combinations are normalized under same allele result entity, which speeds up comparison of genotypes between individuals. In the database there can be multiple genotype versions from same sample-marker tuple, but only one is to be considered as valid. It is the role of external quality control to select and mark the valid genotypes.

The marker identifies the genomic locus. There can be different types of markers depending on which variation the marker is designed for. In the GenomEUtwin the type can be either SNP or microsatellite. The marker data can have also associated information about the primers and genomic maps (not shown in the Figure 8).

The sample is DNA from a subject on which the genotype measurement is done. It is possible to have multiple samples from the same subject in the database. The quality control selects the correct samples and resolves conflicts which may arise when data are pooled from different genotyping centers. The subject table contains the identifier and sex of the person. The identifier (EUid number) is used to join the genotypes with the phenotype data.

Besides the core module, the database has tables for family and twin information as needed for quality control and also generic tables for annotating and cross-referencing the database objects. Family information is compared against the information stored in the phenotype database to assure the consistency between the databases. This is done as part of quality control. The annotation module can be used, for example, to store information on quality checks and mark out the genotypes having inconsistent information.

Import of data will follow a similar pattern developed for the phenotype database. Submitted data will be imported into a submission database which maintains original unstructured data with submission information like dates and name of contact persons. The data are then transferred into the GTdb genotype database in a separate process. During the transfer some minor modifications can be done to the data if needed, for example, to harmonize allele coding for analysis (especially microsatellite data). In the transfer process the data are also transformed into the final relational format. The genotype database will have the sample, person and marker information preloaded and genotypes with unknown marker/sample/person identifiers cannot be inserted into the database. This is enforced by using integrity constraints implemented into the relational database management system. In the future we will explore other more automated possibilities to synchronize or federate multiple heterogeneous data sources.

https://doi.org/10.1375/twin.6.5.383 Published online by Cambridge University Press

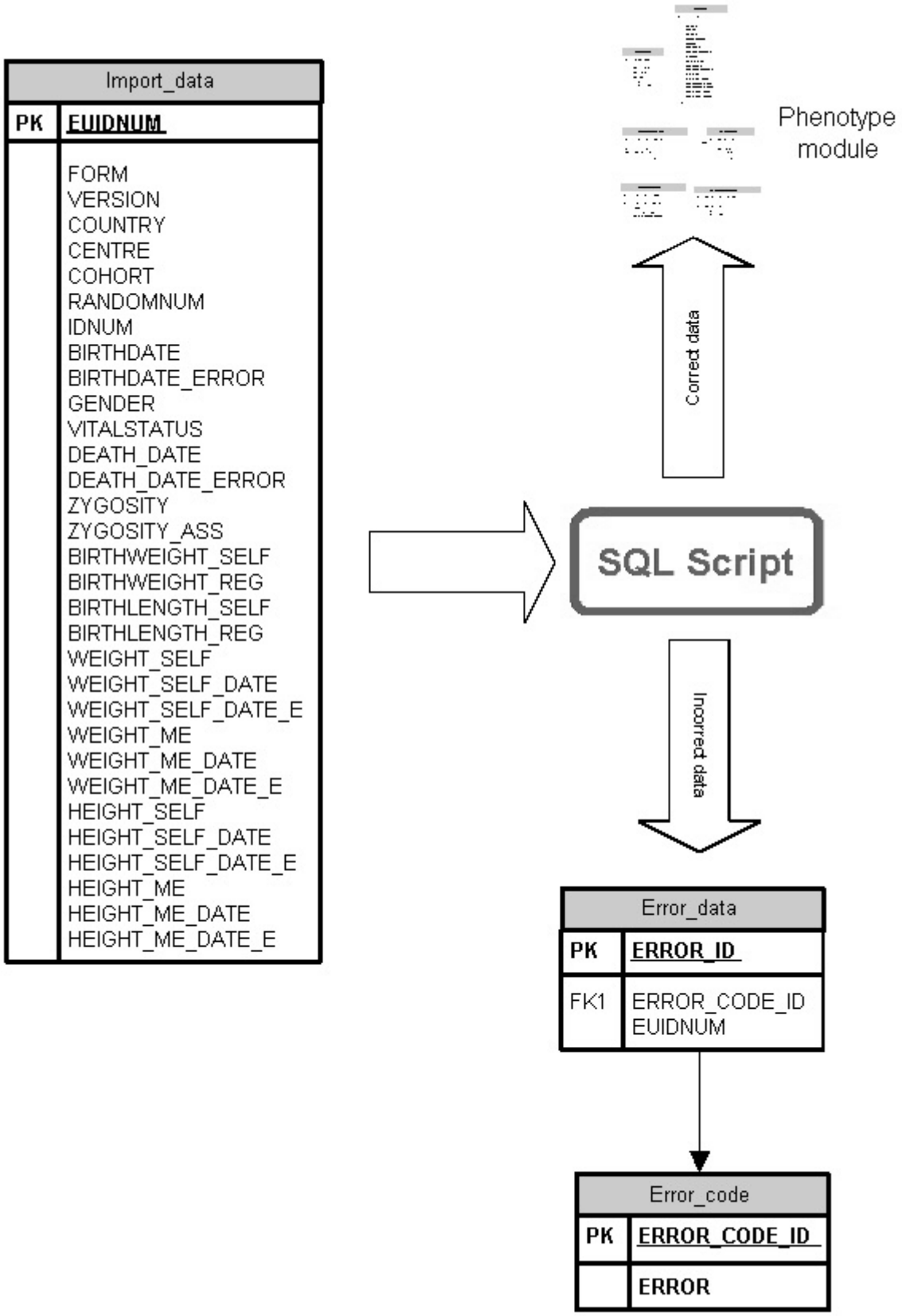

Figure 7
Data model Phenotype Database, module import.

https://doi.org/10.1375/twin.6.5.383 Published online by Cambridge University Press

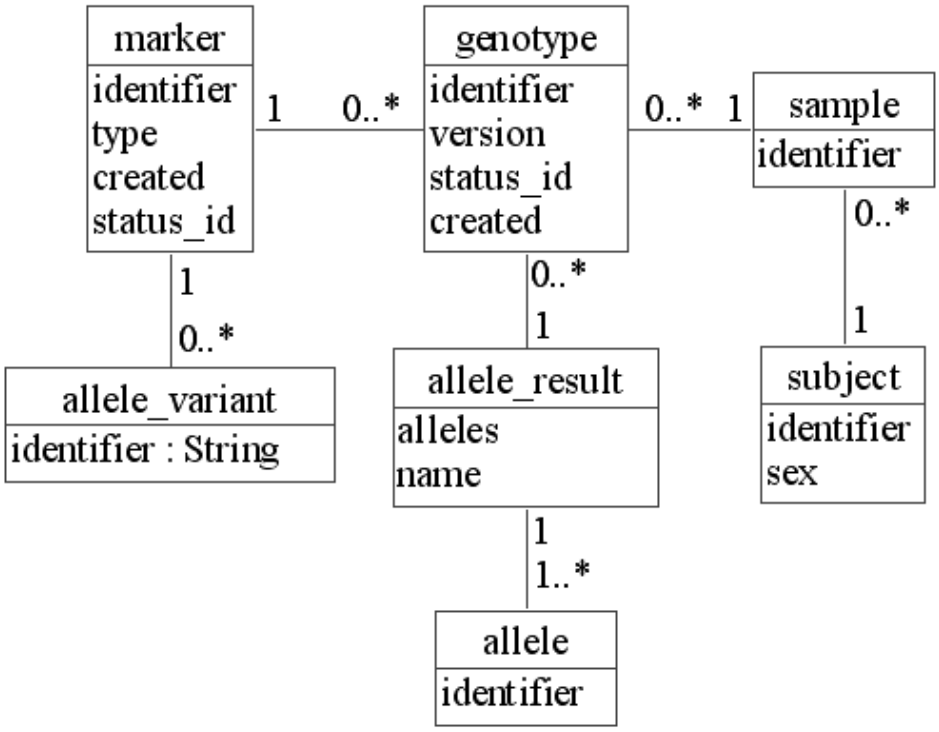

Figure 8
Conceptual data model of core Genotype data.

Software

Application program interface (API) for the database is under development. The Java based API provides methods for writing and reading the data to/from the databases. A function of the API is to simplify access to the data and hide database specific features. The abstraction makes it possible to modify the database schema without breaking implementations based on the API. The API will be used in flat file importers/exporters and also in possible web interfaces made for the database. The API and the database are developed as open source project.

Discussion

This European database project will allow conclusive, desperately needed breakthroughs concerning methods for securing and storing the value, quality and usefulness of large quantities of information that can be collected with the aid of modern information technology. The project contains important elements of work concerning the definition, structure and standardization of information that has been gathered from a multitude of sources from European twin registries. Setting up a new common storage infrastructure and computer strategy and a format and variable standard for the European twin community will have implications for future database handling and European population-based registries like the twin registries and for collaborative epidemiologic research within Europe.

Acknowledgments

This work has been possible thanks to all member of the GenomEUtwin database core. Members of the core can be found on http://www.genomeutwin.helsinki.fi/research_core_database.htm

This project was supported by the European Commission under the programme "Quality of Life and Management of the Living Resources" of 5th Framework Programme (no. QLG2-CT-2002-01254).

References

1. Björklund, A., & Litton, J-E. (2003*). Data format and variable standard for GenomEUtwin's phenotype database prototype*. Internal document GenomEUtwin, version 3.2.
2. Gumm, H. P. (1985). A new class of check digit methods for arbitrary number systems, IEEE. *Transactions on Information Theory, 31,* 102–105.
3. http://www.pgp.com/
4. Jeanette Papp pers. comm.
5. Juha Saharinen pers. comm.
6. David Fange per. comm.
7. http://www.mucosa.de/ Mucosa LIMS
8. http://hapmap.cshl.org/xml-schema/ HapMap XML schema for genotype data
9. http://www.bioinformatics.med.uu.nl/index.html#/db/GIDS/index.shtml GIDS database
10. http://gtdb.sourceforge.net

https://doi.org/10.1375/twin.6.5.383 Published online by Cambridge University Press