Chapter 6

A system for uniform treatment of clinical data

Morris A. Swertz¹, René J. Oostergo², Engbert O. de Brock³

¹Groningen Bioinformatics Centre, Groningen Biomolecular Sciences and Biotechnology Institute, University of Groningen, The Netherlands and ²Trial Coordination Center, University Medical Center Groningen, The Netherlands and ³Cluster Information Systems, Faculty of Management and Organization, University of Groningen, The Netherlands.

Submitted
Abstract

Background | A clinical trial coordination centre, such as TCC in Groningen, takes care of many heterogenic study databases. Typically, these databases are designed and managed individually because of different origins and needs. As a consequence, data preservation and distribution is time-consuming because operating procedures and extraction methods have to be re-developed for each new study database. Our objective is to obtain - make or buy - a flexible infrastructure to preserve clinical databases and make data extractions available to the researchers in a timely and labour extensive manner.

Methods | In order to elicit the detailed requirements, various stakeholders were interviewed and some archetypical examples of existing clinical trial research databases of TCC were analyzed. Literature and software systems regarding clinical data management were reviewed, looking for suitable systems and methods.

Results | We present a detailed description of the requirements, such as reconstruction of historic data extractions. Literature and system reviews showed no available clinical data infrastructures that met all requirements. We decided to make a generic system ourselves. We explain our design decisions, include a generic and flexible data model, and give mechanisms to preserve and extract data in a custom, reproducible and labor extensive way.

Conclusions | We designed a general way to bring different existing clinical trial databases together into one analytical, web-based system. We demonstrated how such a uniform system eases reuse of methods by data administrators and provides clinical researchers with a uniform (web-based) user interface to quickly extract custom datasets that suit their analysis needs. The detailed descriptions also constitute a foundation for local system developers to base their own "buy or make" decisions or "custom make" projects upon.

Availability | a basic implementation without data extraction user interface is available at www.molgenis.org/variant/metabase.

About Chapter 6

This chapter reports the third of four case-studies. The purpose of this case was to refine the ‘generative strategy’ (Chapter 2) for infrastructure to integrate, query and (re)distribute clinical trial data. Predictability of the variation between studies (in contrast to Chapters 4-5) allowed use of another mechanism than generation. We used a generic model where all details about study variants (e.g. questions, questionaires) as well as contents (answers) is stored. One could think of such ‘meta model’ as a special type of domain specific language.
6.1 Introduction

A trial coordination centre has to take care of many (heterogenic) clinical study databases. The Trial Coordination Centre (TCC) of the University Medical Centre Groningen (UMCG) in Groningen is a typical example (see TCC website). Currently TCC manages the results from questionnaires and measurements for about 40 studies, with an increase of 10 per year. Study size varies from 20 to thousands of patients, and 5 to 25 visits per patient. Until now these study databases have been managed and exploited individually which becomes cumbersome and labour intensive because operating procedures and (extraction) method development have to be repeated for each study database. TCC wanted a labour extensive, web-based system to preserve and distribute these heterogenic clinical trial data in a uniform way so clinical researchers can easily extract data across several trials while TCC developers can reuse structures, procedures, and methods. Literature and existing clinical data management systems were analyzed to answer the question: “Make or buy such a system?”.

Academic systems such as TRIALDB (Brandt, et al., 2000; Nadkarni, et al., 1998) and OpenSDE (Los, et al., 2005), and commercial systems such as Oracle Clinical (Oracle, 1999), INTEGRATED REVIEW, Phase Forward’s CLINTRIAL did not provide all required features, such as complete import of existing clinical trial databases, a flattening tool for extraction), versioning support so researchers can rerun their analysis at a later date using the original data, export to various statistical systems such as SPSS, SAS, or R, capabilities to query across studies (meta-analysis), and/or in-depth developer oriented documentation to allow TCC members and even TCC customers to extend the system themselves (see section 6.2 below). For these (and other) reasons, the conclusion for TCC was to “make such a system”.

This paper is based on (Swertz and de Brock, 2005) and is organized as follows. Section 6.2 describes the requirements definition, a definition of essential features for the system to reach the goals within the given context. Section 6.3 discusses the design decisions we have taken as well as the resulting functional design, a design of the functional parts of the system, necessary to provide the required features. The requirements definition and functional design form the blueprint for the implementation. Section 6.4 continues with discussion and future work. The paper ends with conclusions in Section 6.5.

6.2 Design objectives

We started with information analysis. The information analysis formed the basis of the requirements definition and functional design. We used the following methods:

- Interviews: Several interviews took place with various stakeholders: the general manager of TCC, some developers and managers of existing clinical trial research
databases at TCC, and some clinical researchers (future users). These interviews formed the basis for the requirements definition.

- **Systems analysis**: Simple and complex examples of existing clinical trial research databases at TCC were analyzed to identify commonalities that can be addressed uniformly and variation points that have to be supported, i.e., PREVEND (Hillege, et al., 2002), COACH (Jaarsma, et al., 2004), and TRAILS (de Winter, et al., 2005). These analyses produced technical details for the requirements definition.

- **Literature and systems review**: Literature and software systems regarding clinical data management were analyzed, looking for examples of the state-of-the-art in clinical trial data management that satisfied the requirements. However, as explained in the Introduction, it led us to the conclusion that no satisfactory system existed and, hence, not to buy but to make such a system. The literature review was also used as additional input for the functional design.

What was wrong with the existing situation?

- Ad hoc implementation of data extractions, e.g., duplicated efforts are needed in order to find, retrieve, reformat, convert, standardize, integrate, and aggregate data from each study for each data request.

- No versioning, so no possibility to reconstruct the database state at a specific point in history, e.g., at an extraction date as of 2001/12/23. This leads to problems when a researcher needs to rerun part of an analysis in relation to a publication under review.

- Data not directly accessible (via Internet) by clinical researchers so they always had to call TCC and wait for custom extractions.

- No standard data structures, so difficult for researchers to learn and hard for TCC developers to reuse previous work, etcetera.

- No additional information (“metadata”) about studies.

- No logging of extractions so researchers cannot not easily recall what data (version) and query was used for a certain analysis.

- Nearly impossible to analyze across multiple studies because of structural and semantic heterogeneity.

In summary, our information analysis revealed that TCC wanted a labour extensive system with a web-based interface to preserve and distribute such heterogenic clinical trial data in a standard way so clinical researchers can easily extract data across several trials while TCC developers can reuse structures, procedures, and methods. We will call the required system METABASE.

In the classical distinction between transaction processing and analytical processing, the METABASE system would provide the basis for (all) analytical processing within TCC. The role of METABASE is depicted in **Figure 1**. We will define the requirements in more detail.
The following main METABASE features were identified:

1. **Metadata**: provide a flexible database structure to store data in and about studies, i.e., data as well as metadata, without structural changes upon adding a new study.
2. **Study import**: support easy import of the collected study results data as well as study structure data into METABASE.
3. **Authorized extraction**: enable authorized clinical researchers to extract relevant subsets from several clinical trial databases for data analysis.
4. **Reproducibility**: support change history of study database releases as well as extraction logs in order to enable reconstruction of an extraction from a specific date.

Each feature will be treated in a separate subsection, starting with a short motivation and followed by an enumeration of required feature details, called “issues”. The issues may indicate structures to store specific data, functions to manipulate data, or important aspects; in combination they define the feature. With each issue a description and some comments are given.

**Metadata**

This feature has to enable addition of new studies without structural changes to METABASE, and should offer a stable basis for reusable extraction methods and operation procedures.

**Motivation** | In most study databases each question of a questionnaire is represented as a table column (structure) and all responses of a patient are represented in a table row (state). Import of such data in their original form would require the addition of new tables for every questionnaire added. This is undesirable because this requires a lot of work and hinders reuse of methods and operation procedures because they have to be hand-fitted to every table of each newly added study database.

**Issues** | Storage of new study databases without structural changes to METABASE requires that all necessary metadata describing the study database structure can be stored in METABASE. Analysis of example study databases as well as the literature resulted in a set of metadata issues. To allow for variations over the study databases, METABASE has to be...
able to store metadata on all these structural patterns. The Metadata issues are listed in Table 1.

Table 1 | **Requirements for the Metadata feature**

<table>
<thead>
<tr>
<th>Issue</th>
<th>Description</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Studies</td>
<td>Store data about individual studies.</td>
<td>Data about studies are typically stored in separate files or databases. In METABASE data about studies have to be represented as rows in a table.</td>
</tr>
<tr>
<td>Forms</td>
<td>Store question forms (modules of questions or measurements within a study).</td>
<td>In study databases, forms (questionnaires) are typically stored as separate tables. In METABASE these have to be represented as rows in a table.</td>
</tr>
<tr>
<td>Questions</td>
<td>Store individual questions (may have repeating answer elements).</td>
<td>In study databases questions are typically represented as table columns and repeating answers are typically modeled by adding extra columns. In METABASE these table columns have to be represented as values in a table.</td>
</tr>
<tr>
<td>Responses</td>
<td>Store responses to the questions for a specific patient, visit, or study.</td>
<td>In study databases such responses are typically stored as column values. In METABASE both column head and column value have to be represented as values in a table with additional information identifying the metadata (i.e., questions, visits, patients).</td>
</tr>
<tr>
<td>Codes and code lists</td>
<td>Store codes and code lists.</td>
<td>In study databases responses to questions are often coded and restricted to a limited set of answers, e.g., sex = {man, woman}. These lists may be stored in a table but often we need the study protocol codebook for it. In METABASE these have all to be represented as rows in a table, in one table.</td>
</tr>
<tr>
<td>Visits and visit schemas</td>
<td>Store the moments when patients should be and are actually questioned.</td>
<td>The study protocol states which questionnaires should be worked through at which visits. The visiting schema may be available in a separate table. However, incidental unplanned visits can occur. In the study databases the individual visits are typically identified by a value in a response row. In METABASE all these data have to be represented as rows in tables.</td>
</tr>
</tbody>
</table>

**Study import**

This feature has to enable easy import of various clinical research study data and study structure data into METABASE.
Motivation | Study databases are, after validation, offered to TCC for preservation and distribution. Typically, the data is shipped to TCC in the form of delimited text files where each file represents an individual database table. To complicate matters, individual study databases may know subsequent releases. Finally, each study database may come with additional materials such as an experimental setup or planning. Organizing these data for preservation is challenging. Simply transforming these data by hand into METABASE would incur unnecessary cost for TCC because the similarities of the imported study databases are not exploited.

Issues | The Metadata feature, as described above, provides for storage of the data in and about the study databases. What is needed in addition is to provide support for the import process. Issues regarding the import feature are listed in Table 2. The order of issues corresponds to the sequence of actions needed for import.

Table 2 | Requirements for the Study import feature

<table>
<thead>
<tr>
<th>Issue</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preparation</td>
<td>Prepare data before import into METABASE. For import into METABASE, the data is expected to be validated and shipped in delimited text files.</td>
</tr>
<tr>
<td>Mapping</td>
<td>Associate source files with columns and elements in METABASE. The source dataset will be tagged with metadata as a basis for transformation into METABASE. These tags are needed to identify which columns map database elements such as visits, patients, and questions. As the source databases are very similar (though not identical) in structure and format, a generic way to design such mappings is feasible.</td>
</tr>
<tr>
<td>Annotation</td>
<td>Annotate imported study databases with relevant information. Relevant information about the study databases for TCC members and/or applied researchers are added and may include study name, release version, release date, and additional descriptions on metadata elements (such as the complete question text).</td>
</tr>
<tr>
<td>Unflattening</td>
<td>Transform study data to the METABASE structure. Data is transformed from the table-like source data model to an “unstructured” METABASE form. The imported data will also be archived in its original format.</td>
</tr>
<tr>
<td>Upload</td>
<td>Upload the transformed data into METABASE. After transformation the data will be loaded into METABASE. At this point there will be room to validate the transformation process.</td>
</tr>
</tbody>
</table>
Authorized extraction

This feature has to enable authorized clinical researchers to extract their relevant subsets from clinical trial databases.

Motivation | Researchers need data from the clinical trial databases for analysis, but typically only need (or are authorized to) a certain subset of the data. So the data needs to be selected from the study database and then packaged into a format suitable to the researchers’ analysis software. Researchers typically need a two dimensional data matrix. This process is also known as “flattening”. The returned data is a package that contains a selection of responses from selected visits and includes the relevant code lists. Building such extractions is currently cumbersome and labor intensive. The extraction methods have first to be fitted to the specific database structure of the study database. Then, at request, TCC members have to determine if the requester is actually authorized to get that extraction and finally they need to implement a program to realize the extraction and transformation into the suitable format. Little of this work can be reused for later extractions and there is no good administration of executed extracts.

Issues | The availability of a standardized database enables a standardized and reusable implementation of extraction algorithms. The consequence is that applied researchers do not have to “call TCC” every time they want an extraction, reducing the time to realize a new extraction. Meanwhile TCC can spend time on more specialized tasks. To enable the authorized extraction facility, a set of extraction issues is identified, see Table 3.

Table 3 | Requirements of the Authorized extraction feature

<table>
<thead>
<tr>
<th>Issue</th>
<th>Description</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorization</td>
<td>Assign extraction rights for (parts of) study databases to groups/individuals.</td>
<td>The administration of these rights should be maintainable by the study owner.</td>
</tr>
<tr>
<td>Selection</td>
<td>Select metadata elements of interest as columns of the “flattened” result table.</td>
<td>Each question (or measurement or question repeat) is represented as a column in the flattened table. Metadata about patients and visits are automatically added.</td>
</tr>
<tr>
<td>Filtering</td>
<td>Define filters on the extracted result rows.</td>
<td>Filter study data based on patients, visits, and the responses on specific measurements. These selections are equivalent to setting conditions in the “WHERE clause” in an SQL statement.</td>
</tr>
<tr>
<td>Flattening</td>
<td>Execute the flattening and filtering based on extract definitions.</td>
<td>Transformation from the generic “structure-less” database of METABASE to the specific “structured” or “flat” representation to which the filtering is applied.</td>
</tr>
<tr>
<td>Provenance</td>
<td>Provide extractions with annotations that describe the extracted data.</td>
<td>Extractions include annotations such as extraction date, extraction definition, database (release) used, and “metadata” (such as code lists).</td>
</tr>
</tbody>
</table>
Catalogue extraction definitions for traceability and reuse purposes.  
Log executed extractions for tracking and reporting purposes.  
Enable TCC members, study owners, and researchers to create or use pre-defined extraction functions.

**Reproducibility**

This feature supports the change history of study database releases and extractions executed in order to enable reconstruction of an import or extraction as of a specific date.

**Motivation** | Although study databases are validated before they are preserved and distributed by TCC, there are often unforeseen events that require later changes to the study database. It is therefore not uncommon that a clinical trial database is updated and offered again to TCC and overrules the previous release. Another reason for several stored releases of the same study is that some researchers already want to delve into intermediate results while the study is not yet finished. However, this complicates matters for TCC and the biomedical researcher because analysis of the new release may yield different results.

**Issues** | It should be clear to the researcher whether extracted data comes from other study database releases than the previous extraction and there should be the possibility to get an extraction as if it was run at a specific date, e.g., because some published results were based on the data as known on a specific date and needed to be redone following comments from reviewers. By means of questions, Table 4 explains what is needed to enable reproducibility.

**Table 4 | Requirements of the Reproducibility feature**

<table>
<thead>
<tr>
<th>Issue</th>
<th>Description</th>
<th>Comments</th>
</tr>
</thead>
</table>
| Releases       | When was data known inside the original study database?                                                                                             | Tag the data in the database in order to identify and separate study releases.  
See also Section 2.1.                                                          |
| Import log     | When and how was data imported inside the meta database?                                                                                                | Annotate releases in order to identify and separate the raw data as known by TCC and extractable from METABASE on a specific date. The imported data files are archived separately.  
See also Section 2.2.                                                            |
| Extraction log | When and how was data extracted by a researcher?                                                                                                     | Log extractions in order to execute extractions as of a specific date. For instance, because a researcher needs to do another extraction on the same data as a previous example.  
See also Section 2.3.                                                            |
6.3 Functional design

The functional design of METABASE describes the working of and relationships between the functional parts of the system. The essence of the requirements definition (Section 6.2) is to prevent the need for structural changes to METABASE when new study databases are added or new applications are needed.

The problem that TCC faces is challenging because of the variation both in the supply of study databases as well as in the needs of the consumers of study data. Instead of the current ad-hoc solutions for these variations in terms of creating new modules for each study and each extraction, a more permanent solution is needed. METABASE should account for variation coming from suppliers and from consumers of study data. At the same time the level of abstraction should not be that high that the users of METABASE (TCC members, study providers, applied researchers) cease to understand its structure.

The essence of the solution is to store the \textit{data structures} as well as the \textit{data states} of study databases as one uniform and integrated data state in METABASE. In order to store incoming study data structures (such as tables from study databases with each column representing a question) without structural modifications to METABASE, the structural “metadata” about studies has to be imported as well as the study data itself. In the database inside the METABASE system there should be room to account for the structural variation of study databases. In short, the database of METABASE will have to contain data as well as metadata of multiple studies, each with patients, visits, and questionnaires.

Below we detail the functional design that realizes this essential functionality. The following main aspects of the solution are described:

1. **Architecture:** an overview of the structure and components of the METABASE system.
2. **(Un)Flattening:** design of the patterns that underlie the import and export functions, i.e., transformation of imports of so-called “flat” study data (with specific structure) into “unflattened” METABASE data (with uniform structure) and exported to (re)flattened data extractions for analysis (with, typically different, specific structures).
3. **Database:** design of the uniform structure of the database in the METABASE system that allows METABASE to store study structure data as well as study results data.
4. **Other important design aspects:** design of the versioning of released data, logging, and authorization aspects that allows for reproducibility, traceability, and authorized extractions.

In the next subsections, each of these aspects will be described.

**Architecture**

As **Figure 1** in Section 6.2 already emphasized, METABASE will operate in the context of the studies it preserves, and the research interface (a website) to which it distributes (used
by the biomedical researchers). The architecture explains globally the components of the METABASE system, see Figure 2.

![Diagram](image_url)

**Figure 2 | The architecture of the METABASE system.** Database: the heart of METABASE, which enables flexible storage of study data as well as structure data (metadata); Import module: enables mapping and unflattening the data coming from the studies into the database of METABASE; Extract module: enables selection and (re)flattening study data from the database of METABASE into table-like data sets that biomedical researchers use for analysis; Import/Extract logs: enable traceability of study release imports and data extractions; Archiving module: archives the source data as they come in; Authorization module: enables the access rights of users and user groups to the database, e.g., based on a specific study, specific patients, specific visits, and/or specific questions.

**(Un)Flattening**

The essence of the METABASE is to allow for structural variations, both in the imported study databases as well as in the extracted data sets. To provide this functionality the inverse-like algorithms for unflattening and (re)flattening are needed, i.e., to transform study structures into the database upon import and, vice versa, into “flat” result sets upon extraction. This is a proven design pattern for clinical trial data and is also known as “Entity Attribute Value (EAV) modelling” or “Row modeling” (Nadkarni and Brandt, 1998) and are applied in most contemporary clinical trial information systems such as TrialDB, Oracle Clinical and now also METABASE. However, in METABASE the EAV modelling is not located in the source data but in the (off-line) database, which acts as a data warehouse (Inmon, 2002), see Figure 2.

The design pattern of (un)flattening is illustrated via a general example, see Figure 3.

**Explanation of flattening/unflattening:**

- **Key** with value $K_i$ identifies a subject, for example patientID = 1 or visitID = 1.
- **Attribute** $Q_j$ identifies a question, for example $Q_1$ = “WEIGHT”.
Value $R_{ij}$ identifies the response of subject $K_i$ to question $Q_j$, for example $R_{1,1} = 69$ where $Q_1 = \text{“WEIGHT”}$. 

In most study databases all responses $R_{ij}$ of all subjects $K_i$ are modelled in tables with questions $Q_j$ as columns. Thus, questions are hard-coded in the structure, so adding questions requires adding an extra column.

In the data model of METABASE both the questions $Q_j$ (column heads) and responses $R_{ij}$ (column values) are transformed into column values, under column heads Question and Response respectively. The questions are not coded into the structure anymore, so new questions can be added without restructuring.

In a study Response table, column Key provides data that identifies the subject answering a question, e.g., the visit of a specific patient in the context of a specific study and questionnaire.

Only a limited collection of such flattening and un-flattening functions will be needed:

For each table of the imported study databases, typically many tables for each set of questions (columns) and responses (rows), a table with code lists, and a table with visits.

---

**Figure 3 | Functional design of flattening and unflattening**
• For each type of extraction. Probably a generic tree representation of studies will suffice, i.e., selections of question per visit per study.

Given meta information on what data values represent K, Q, and R respectively, the unflattening functions for import are automatically implemented using software generators. Similarly, tailor made flattening functions for extraction are generated from this K, Q, R meta data in the database (see below). This prevents much of the laborious handwork.

**Database**

The functional design of the METABASE data model details the structure of the database to store study *structure* data as well as study *results* data from the different study database *releases*. The functional design of the data model is summarized in Figure 4 and explained below:

- The data model consists of 11 tables (also called “entities”), ranging from whole **Studies** to individual **Responses**.
- Arrows in Figure 4 denote references, d-a-s-h-e-d arrows denote optional references. The underlying foreign key attributes are marked by => respectively (=>).
- We decided to use automatically generated, “meaningless” keys, consisting of only one attribute. They are underlined in Figure 4, e.g., StudyID.
- Figure 4 distinguishes levels – L0 to L4 – and, from L2 on, four columns – C1 to C4.
- The seven tables of Studies (L0), PlannedVisits and VisitForms (C2), Forms and Questions (C3), and CodeLists and Codes (C4) contain the results of the study design.
- The tables of Releases (L1), Patients, Visits, and Responses (C1) contain the results of study execution. (Responses will, by far, be the table with the largest number of rows.)
- The core of METABASE is the administration of Responses during Visits given by Patients. Responses, Visits, and Patients represent the actual study results.
- Studies can have several Releases. A study will typically have a general goal attribute while a release will typically have a creation date and a date of import. A release may be locked and/or retrievable.
- Code Lists and Codes limit the lists of allowed Responses to certain Questions. Code lists map codes values to code labels (descriptions), e.g., 1 ≡ “male” and 2 ≡ “female”.
- During study planning, lists of Questions will have been determined, typically organized in one or more questionnaires, also known as Forms.
- Some Questions may allow several (repeating) Responses. Some Questions may not be answered (have missing values).
- During study planning, a protocol of Planned Visits may have been defined, possibly including the Forms that have to be (partially) treated then (Visit Forms). It may be that the actual Visits deviated from planning, that the planned visit was spread over more than one physical visit (so-called “dynamic visits”), or that there was no planning at all.
Figure 4 | Data model for the database of METABASE
(as in a hospital environment). It is therefore possible that actual Visits could not be linked to Planned Visits or vice versa.

- **Visits** can be nested: Visits refer to Visits (by ParentID).
- **NodeID** contains the path of VisitID’s to the topnode/topvisit (separated by dots).
- **OriginalKey** holds the key-value for this field in the original data and is used for import-purposes only.
- **VisitDate** holds the date the visit has occurred.
- Patients may have filled out more than one form per visit. This means that there could have been recurring forms at each visit.
- **Sequence** is an ordinal number denoting the order of the VisitID-FormID combination per patient. Therefore, (PatientID, FormID, Sequence) is another key in Visits.

**Other important design aspects**

The functional design provides developers of clinical trial information systems with guidelines and does not aim for a complete cookbook description of the implementation. For example, the following aspects are considered relatively easy and therefore are not explained in detail in this paper:

- Logging of events such as imports and extracts from the database.
- Archiving of imported study datasets, i.e., the data files.
- Provenance descriptions, i.e., annotation of extractions with accompanying code lists and study-related metadata.
- Extraction catalogue, i.e., a list of predefined extraction queries that can be reused by others.

However, there are some design aspects that need to be addressed, see **Table 5**.

**Table 5 | Functional design of reproducibility and authorization**

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Design summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reproducibility</td>
<td>To enable historic views (not only the most recent release of study data), the design pattern is as follows:</td>
</tr>
<tr>
<td></td>
<td>Each table (except Studies) will - directly or indirectly - have a “release” column. This shows which element was known in which specific releases.</td>
</tr>
<tr>
<td></td>
<td>Releases are annotated with the date the release was created and the date the release was imported into METABASE.</td>
</tr>
<tr>
<td>Authorization</td>
<td>To assign rights to extract specific study data and/or rights to authorize others to do so, the design pattern is as follows:</td>
</tr>
<tr>
<td></td>
<td>Users and Users Groups can be administrated.</td>
</tr>
<tr>
<td></td>
<td>Extraction rights can be granted on study elements, i.e., on studies, visits, patients, etcetera.</td>
</tr>
<tr>
<td></td>
<td>Execution rights can be granted on pre-defined or historical</td>
</tr>
</tbody>
</table>

Basic mathematical notation can be used in the online Reader. Notes are saved in a personal or shared workspace.
Administration rights can be granted on the level of studies to study owners to allow them to grant rights to other users. The consequence is that data extractions can only be executed by authorized people and that TCC members are not required to manage all individual rights and extraction requests.

### 6.4 Discussion and future work

While the requirements analysis and functional design cover a large part of the development of METABASE there are some remaining questions.

**How about extractions by researchers?** | TCC also needed a tool so researchers can easily navigate the generic METABASE concepts (i.e., studies, forms, visits, etc.) and choose what data they want to have in a flat result file (for statistical analysis). It is well-known that information extraction from “unflattened” clinical studies is not straightforward (Los, et al., 2005; Nadkarni and Brandt, 1998). All data elements have to be joined together leading to complicated queries (in SQL). Furthermore, it must be made possible for clinical researchers with little informatics background to define new extraction queries, hence the need for a visual query system. Because existing query systems from TrialDB (Nadkarni and Brandt, 1998), OpenSDE (Los, et al., 2005), Oracle Discoverer for life science data (Oracle, 2004), and IBM Data Discovery and Query Builder (IBM, 2007) did not fit the uniform METABASE data model, the need for flattened extraction and the need for export to SPSS, it was decided to develop such a tool (which development will be described in detail in future publications).

**How about data from the clinic?** | Originally, METABASE was targeted at clinical data that was collected during research studies (project-like). However, researchers wanted to include “production” data from the daily clinic into their analysis as well. Evaluations showed that the proposed solutions can also be used to distribute snapshots of data collected during daily operations within the hospital. Essential are the notion of Releases, the fact that from a research perspective everything boils down to “questions and answers”, and the point that visits can be nested (i.e., Visits refer to Visits by ParentID).

**How about calculated fields?** | TCC is often asked not only to redistribute study data in their original format but also to transform it to better suit the researchers needs. Examples of these transformations include data aggregations, but also reformatting for specific analysis applications (e.g., SPSS), conversion, standardization, homogenization, and integration (e.g., equivalent data from separate studies). METABASE does not pre-compute at import time but the source database can of course include pre-computed values as answer to a kind of question (such as percentage response, drop outs, bias). Therefore there is a
need to support and ease extension of METABASE with such post-processing functionality when necessary. The exact forms of these transformations are not yet foreseen because it depends on specific studies and specific researcher requests. The expectation is that using (an extended version of) the METABASE concept we can build up a reusable post-processing catalogue, i.e., a list of available post-processing algorithms.

**How about meta analysis?** | Controlled vocabularies, such as the National Library of Medicine’s Unified Medical Language System (UMLS) (Lindberg, et al., 1993), Medical Subject Headings (MeSH) (Nelson, et al., 2004), or Diagnostic and Statistical Manual of Mental Disorders, Fourth Edition (DSM-IV) (Association, 1994), are indispensable to ease integration and meta-analysis. They play a major role in the code and code list tables. Furthermore, we added “meta code lists and meta codes” to allow for the mapping of different codes upon each other.

**How about implementation?** | Although it is outside the scope of this conceptual paper, we will say a few words about our implementation. Since METABASE is not a *transaction* processing system but the basis for *analytical* processing, we can quite freely add indices to the database of METABASE in order to boost performance. In order to reduce the number of joins during extraction and post-processing, we added the (redundant) attribute ReleaseID to all four tables on level L3. We decided to completely load each study release into METABASE. Alternatively, an approach could have been chosen where only the *differences* between releases were stored. While it may save storage space, this will make the queries much harder to formulate and to compute. We implemented the one-attribute keys as auto-numbers to improve performance and reduce memory usage.

### 6.5 Conclusions

In conclusion, a hybrid data model that mixes traditional relational tables for “common” information over studies with the inverse notions of *unflattening* and *(re)*flatting, respectively needed to transform the “specifics” of study structures into a uniform system upon import and, vice versa, into “flat” result sets upon extraction, was essential for designing a system satisfying all requirements. Thus, we succeeded in developing a flexible, uniform, and web-based system to treat clinical trial databases and their historic releases in order to make data extractions available in a labour *extensive* way. The design decisions outlined in this paper can be of great value for other developers embarking to develop a METABASE-like system.

**Acknowledgements**

We thank our collaborators at the Trial Coordination Center (TCC) of the University Medical Center Groningen (UMCG), especially Hans Hillege en Stefan van Oosten, for the fruitful discussions and the opportunity to do this research.