FREIE UNIVERSITÄT BOZEN
LIBERA UNIVERSITÀ DI BOLZANO
FREE UNIVERSITY OF BOZEN - BOLZANO

KRDB RESEARCH CENTRE
KNOWLEDGE REPRESENTATION MEETS DATABASES

Faculty of Computer Science, Free University of Bozen-Bolzano, Piazza Domenicani 3, 39100 Bolzano, Italy
Tel: +39 04710 16000, fax: +39 04710 16009, http://www.inf.unibz.it/krdb/

KRDB Research Centre Technical Report:

Granular information retrieval from the Gene Ontology and from the Foundational Model of Anatomy with OQAFMA

C. Maria Keet

Affiliation	KRDB Research Centre, Faculty of Computer Science Free University of Bozen-Bolzano Piazza Domenicani 3, 39100 Bolzano, Italy
Corresponding author	Maria Keet keet@inf.unibz.it

Keywords	Bio-ontologies, recursive queries, granularity
Number	KRDB06-1
Date	6 April 2006
URL	http://www.inf.unibz.it/krdb/pub/TR/KRDB06-1.pdf

Granular information retrieval from the Gene Ontology and from the Foundational Model of Anatomy with OQAFMA*

C. Maria Keet

KRDB Research Centre, Faculty of Computer Science, Free University of Bozen-Bolzano, Italy
keet@inf.unibz.it

Abstract. Current persistent storage implementations of large bio-ontologies fall short in meeting usability and reusability requirements. Limitations are demonstrated with two representative ontologies-stored-in-databases, the Gene Ontology and the Foundational Model of Anatomy. Their un-usability for granular information retrieval is not due to a lack of knowledge modelled in the respective ontologies, but stems from sub-optimal implementations how the ontologies are made persistent. We discuss and propose several improvements for ontology development, including support for recursive queries, insufficiencies in and improvements for ontology development environments, and finer-grained specification of goals of an ontology. Advantages of a better implementation are illustrated with granularity based querying. This granular information retrieval enables both retrieving more information quicker and can be used for improving and enlarging an ontology.

1 Introduction

The added-value of ontologies has been reiterated many times over, but at present, its effective usage falls short of the promises. Ontologies are not intended just for storing knowledge about the subject domain, browsing and (manual) annotation of data, but can be used for data integration, inferencing, Semantic Web agents mediation, ought/will be more than just an online encyclopaedia and so forth. While, of course, it is first a challenge to develop a good ontology, and a second to use it effectively, but *how* the captured knowledge is stored persistently affects what one can do with the ontology, despite the implementation-independence claim. In fact, the latter should be more precisely formulated: the knowledge modelled in an ontology should be implementation-independent, whereas the storage method commits to certain types of implementations. For instance, OWL files can be used for reasoning purposes to e.g. classify a taxonomy and check consistency, whereas this is not possible with ontologies stored in a (relational) database. On the other hand, relational databases have about 30 years of established research and implementations behind them, where one can take advantage of e.g. sophisticated querying. Taking into account ontology as an engineering artifact, then implementation decisions for ontology development are not trivial.

* A shorter version of the experiments but with more information about granularity was presented at the SBIOLBD Seminar at EPFL, Lausanne, Switzerland, d.d. 13-2-2006, with an extended abstract available at http://www.meteck.org/files/SBIOLBD06_extabs.pdf.

In addition, over the years there always is a 'requirements slip': desiring to use an engineering artifact for other purposes it was originally designed for and users moving the goal posts. For instance, data/information retrieval from ontologies – getting out what has been put in the ontology – with as a sub-goal granularity, i.e. querying an ontology at one's desired different levels of detail, which goes further than the "retrieve ancestors"-type of query. Querying an ontology stored in a database at the desired level of detail can be done 'on the fly' with database views that return e.g. all subtypes of tissue that reside in the *Tissue*-level, or with a full formal domain granularity framework respecting ontological differences in types of granularity [14] that contains different perspectives on the data or ontology, has its granular levels for each perspective predefined, and may be used for inferencing. A conceptually relatively simple method is to position a taxonomy and partonomy orthogonally and to use the *partOf* relation (or spatial *containedIn*, *involvedIn* for processes) to distinguish between levels of granularity. With this combination, one can structure and retrieve more information from the same resource that is otherwise available in the ontology but inaccessible.

With ontologies stored in databases, such change in content and structure of the queries, is, *in theory*, not a problem. However, with current *engineering practices* this is not possible, or only after elaborate reengineering. Two well-known large ontologies stored in relational databases are the Gene Ontology (GO) with more than 19,000 entities [7] and the Foundational Model of Anatomy (FMA) with about 72,000 entities and 1.9 million relations [16], which we analyze and discuss in the next section. We have chosen these two ontologies-stored-in-databases, because the GO is a *de facto* trendsetter for the Open Biomedical Ontologies (OBO) family of ontologies and the FMA is the frontrunner for expressive ontologies created with the Protégé ontology development environment. These ontologies are widely used for annotation but rarely used for ontology-driven information systems [8]. The issues that will be brought afore in the next section, however, are not unique, but representative of the challenges ahead and design considerations to take into account to bring ontologies to the next level to facilitate effective usage, which also has the advantage of being able to improve the ontologies themselves, discussed in section 3. We conclude in section 4.

2 Case studies: experiments with the GO and FMA

2.1 The GO database

Common types of queries of the GO in the DAG-Edit tool [31], AmiGO web page [30], or directly to its database are "find string x" in the tree, definition, comment, id, etc., or the supertype or sibling of x; querying for "descendants only" is problematic, as it relies on an implementation of recursive queries. The recursive query in SQL:1999 is defined using common table expressions as temporary views:

```
WITH [ RECURSIVE ] <query_alias_name> [ ( <column_list> ) ]
   AS ( <select_query> )
   <query_using_query_alias_name>
```

The interested reader can find examples of recursive queries in database fora on the

Web, where options are limited with the numbering scheme of the nodes, and performance tends to be 'slow' (e.g. [2] [19] [25]). Neater solutions than will be discussed here for the GO can be achieved with 'advanced databases' such as deductive databases, databases with Strudel/StruQL, and DL databases that both support recursive queries [26] [1]. SQL:1999 and SQL:2003 both also do support recursive queries, which are more or less implemented in currently available DBMSs, such as DB2 and Oracle. This can be combined with generating database views to construct a particular granular level containing data. Let Table 1 be the table `Producer_level`, then querying the column `Entity1` or `Entity2` for the root entity at the desired depth in the tree returns the contents for a chosen level. For instance, if `Entity2 = 'ToxinProducer'` corresponds to `D` in Table 1, then all entities it subsumes belong to that level. Using the sample data of Table 1, then the basic view is:

```
CREATE VIEW TheToxinProducers_level2D AS
   SELECT entity1 FROM Producer WHERE entity2 = 'D';
```

Together with the recursive query support, all its subtypes are also loaded into the *ToxinProducer*-level. Without support for recursive queries, only one type of granularity can be used: that of isA, where not only each layer in the taxonomic tree constitutes a level but also only those in the same branch of the tree. If the relation would not be isA but $partOf$, then views for each level can be constructed straightforwardly by retrieving only the immediate parts:

```
CREATE VIEW Producer_level3E AS
   SELECT entity1 FROM Producer WHERE entity2 = 'E';
```

Table 1. Sample data for a table that stores a partonomy, called `Producer_level`.

Entity1	Relation	Entity2
E	isA	D
F	isA	D
G	isA	E
H	isA	D
I	isA	E
J	isA	E
...	isA	...

Shortcuts can bypass recursive queries, as, for instance, the ontology-stored-in-a-database Gene Ontology (GO) does. The GO MySQL database itself does not support recursive queries (required for traversing trees), but offers a workaround with the additional `graph_path` table. In `graph_path`, "Paths between any two terms (traversing down only) this table states whether there exists a path between a parent and a child, and the distance between them. multiple paths mean multiple entries in this table" [34]. Querying a section of the biological_process GO [33] with the sample query of [34] using `ancestor.name = 'blood coagulation'` and adding `ORDER BY graph_path.distance ASC`, then

```
SELECT rchild.name, rchild.acc, graph_path.distance
FROM term AS rchild, term AS ancestor, graph_path
```

```
WHERE graph_path.term2_id = rchild.id
   AND graph_path.term1_id = ancestor.id
   AND ancestor.name = 'blood coagulation'
ORDER BY graph_path.distance ASC;
```

gives the output as in Table 2 (data from v3-8-2005 of the geneontology.obo (BP)), having fetched every descendant of *BloodCoagulation* by *listing* the descendants with distance to the assigned root instead of maintaining the hierarchical structure. A DAG-Edit "descendants only" can be retrieved by selecting

`has` *ancestor* `that equals` *blood coagulation* which returns an unordered set without the distances retrieved with the SQL query. Problematic is that `distance` puts together *any* type of relation: the in-/extrinsic pathways of blood coagulation are taxonomic subtypes whereas the other entities are *part of* blood coagulation. Utilising the taxonomy and partonomy separately or orthogonally is as of yet not possible without losing such crucial information. Hence, although the knowledge is stored in the database, the limited query answer is ontologically meaningless. To allocate entities correctly in the appropriate level of granularity, one requires a (depth-first) algorithm that during the search uses and returns the *type* of relation that is traversed. The translations of the GO database to make it browsable through Protégé or convert it into OWL-format does not automatically solve these issues, as will be described in the next sections.

Table 2. Output for fetching descendants of *Blood coagulation* in GO, using the database query.

Name	acc	Distance
blood coagulation	GO:0007596	0
platelet activation	GO:0030168	1
regulation of blood coagulation	GO:0030193	1
blood coagulation, intrinsic pathway	GO:0007597	1
blood coagulation, extrinsic pathway	GO:0007598	1
positive regulation of blood coagulation	GO:0030194	2
negative regulation of blood coagulation	GO:0030195	2
fibrinolysis	GO:0042730	3

2.2 The FMA database

2.2.1 Introduction

One can browse the FMA online [32] and search for a single entity, or get the database dump and install MySQL and Protégé to browse its contents locally through the Protégé software, which does not allow SQL database queries. The FMA database is tightly integrated with, and entirely dependent on, the Protégé application software, which is revealed not only upon inspecting the database dump, but even clearer after reverse engineering the conceptual model (with VisioModeler 3.1), shown in Fig. 1, which is devoid of semantics. For instance the `‘short_value‘ varchar(255) default NULL` in the definition of the `FMA` table may record values, among others, like

`'English'`, `'31227'`, and `'Thu Aug 12 11:30:30 PDT 1999'`, as illustrated in Fig 2 with a section of the database dump. The FMA MySQL database as distributed by the FMA developers is difficult to query as all knowledge is hidden in the application layer (Protégé) the database is supposed to be connected to: in addition to the attribute name as e.g. `slot_value`, the actual values in most other columns are numbers where mapping from number to string and meaning is not provided. Browsing the FMA in Protégé, one has to click each individual relation and navigate through the myriad of pop-up screens to find information. Finding the answer to a simple query like "how many organs does a human have?" is impossible, assuming one does not want to browse and count manually thousands of organ types in Protégé.

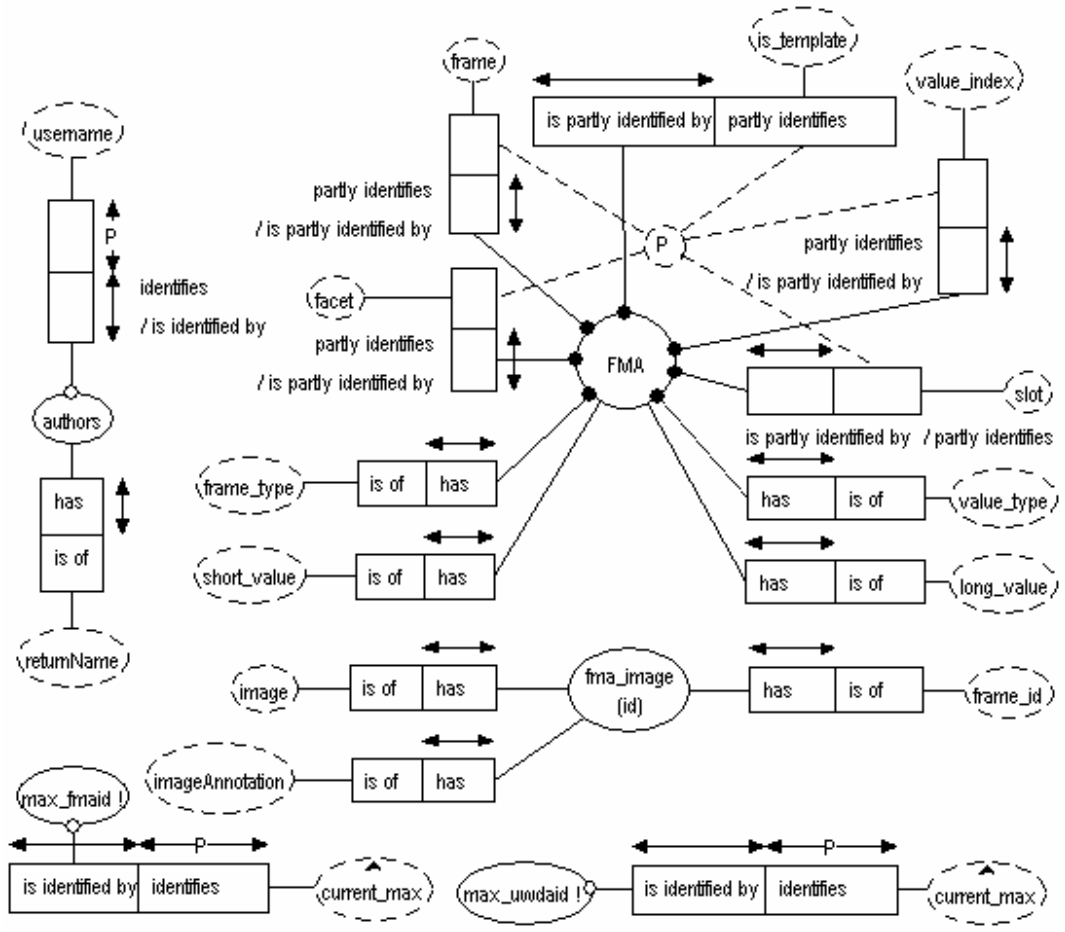

Fig. 1. Reverse engineered ORM conceptual model of the FMA database.

However, not all is lost. The purpose of this experiment is to illustrate how an applied domain granularity framework can be used with FMA and queried meaningfully. This engineering solution for the FMA, OQAFMA and as offered by its developers, does not allow full implementation of a domain granularity framework, but even in its

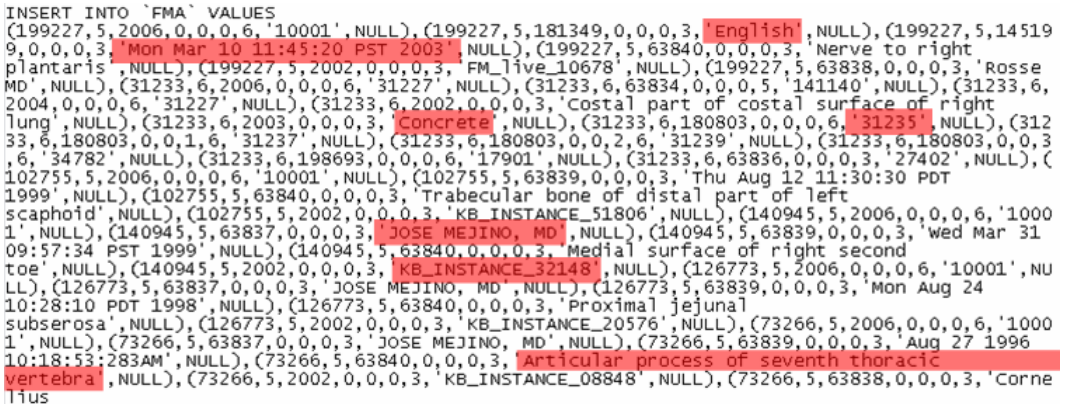

Fig. 2. Screenshot of a section of the FMA database dump with values to be inserted in its core table `FMA`. The marked text are some of the values of the column declared as `'short_value' varchar(255) default NULL,`

current form achieves more than IFOMIS' unsuccessful attempt to test their levels of granularity for human structural anatomy [15] with the FMA in Prolog[1].

The OQAFMA is a query agent for the FMA, that uses StruQL for querying the re-engineered database that stores the FMA ontology [16]. OQAFMA does not use the MySQL database in which the FMA is stored directly, but first performs several preprocessing steps before one can query the database. This combination has been re-engineered into a format usable for querying: the OQAFMA query agent [16]. Preprocessing steps involve optimizations such as the creation of a new table for each edge (relation) type, adding views and storing it in a PostgreSQL database [16], and can be queried through an online interface [35] using the StruQL query language that returns the answer in XML for further processing (e.g. visualization). Before testing OQAFMA features to support granular queries, several queries were formulated in natural language, bearing in mind how domain granularity for human anatomy may be defined with organ-tissue-cell-organelle and so forth based on the FMA partonomy, as shown in Table 3 (It is outside the scope of this technical report to go into details of ontological soundness of the levels and its contens, and of the notion of granularity in a granularity hierarchy[2], but not declared and hard coded in the database. Details about the materials and methods are provided in the next section.

[1] The recursive functions in particular caused some problems (Werner Ceusters, personal communication June 2005).

[2] First, the ontological (un)soundness of the presented level definition, naming, and allocation of the entities in their respective levels. For instance, the FMA contains *Hormone* as structural entity, but it is a functional one (structurally, it can be a peptide like insulin). Second, the granularity in a granularity hierarchy to 'skip' levels where one may identify in reality more levels than used in the software system, like $gp_{hsa}gl_2$ collapses three levels into one. Both [23] and [29] achieved better performance with software implementation using wider and shallower levels of granularity than with fewer perspectives that had more detailed levels; this is a topic of future research.

Table 3. Granular perspective gp_{hsa} human structural anatomy and nine defined levels with some examples.

Levels	Name	Examples
$gp_{hsa}gl_1$	Body	*MaleBody*
$gp_{hsa}gl_2$	Principal body part	*Head*, *Limb*
	Subdivision of principal body part	*LimbGirdle*, *Face*
	Organ system	*RespiratorySystem*
$gp_{hsa}gl_3$	Organ	*SalivaryGland*, *Pancreas*
$gp_{hsa}gl_4$	OrganPart	*Tendon*, *Cortex*, *LymphNode*
$gp_{hsa}gl_5$	Tissue	*Epithelium*, *SmoothMuscle*
$gp_{hsa}gl_6$	Tissue part	*HairFollicle*, *Nail*
$gp_{hsa}gl_7$	Cell	*MultiPotentStemCell*, *Melanocyte*
$gp_{hsa}gl_8$	Cell part	*Chromosome*, *Cytoskeleton*
$gp_{hsa}gl_9$	Molecule	*Hormone*, *Protein*, *Melanin*

2.2.2 Materials and methods

Relating the inaccessible but intuitively assumed FMA structure to domain granularity and some simple data manipulation operators, then for this experiment a granular level corresponds to one layer in the partonomy that has, by using $assignGrainLevel$, loaded into that level the relevant part of the taxonomy. For instance, the *Cell*-level contains the entity *Cell* and all entities that are (taxonomically) subsumed by it. I do not introduce new relations to the FMA but exploit the existing encodings. The $getLevel(HumanStructuralAnatomy)$ traverses the extant $partOf$ relations and returns the top common subsumer in the taxonomic structure residing in a level, e.g. *Organ*, *OrganPart* etc. Firing $getContents(Cell)$ corresponds to a recursive StruQL query that includes the statement

```
CellLevel->":DIRECT-SUBCLASSES"*->CellSubclasses
```

which returns all subtypes of *Cell* residing in the *Cell*-level. Further details on positioning the taxonomy and partonomy are given below.

An online query interface has been created for OQAFMA [35], and was used for the experiments, using Internet Explorer 6.0 accessing from a remote machine. Before testing, four queries were formulated in natural language, which are as follows:

I. Continuing with the running example about blood, the query is: "what are the cellular components of blood?", which can be decomposed into smaller tasks where *Blood* resides in another level as the cells it has as parts. In addition, we want to retrieve not an unordered set of blood cells at the *Cell*-level, but taxonomically structured.

II. The query "which cell type(s) do(es) not have a nucleus as part?" involves adjacent levels of granularity, being the *Cell*-level and *SubcellularOrganelle*-level in the human structural anatomy perspective.

III. "Which hormones are located in the kidney, and where in the kidney?" This uses two different perspectives, being a structural one for *Kidney* (an organ) with its parts and a functional one for *Hormone* (a molecule with a specific function).

IV. Last, "In which organs are macrophages located?", i.e. for each macrophage (and its subclasses), in which tissue (or its subclasses) is it located?

These questions were translated into StruQL[3], where possible, and the database queried through the OQAFMA interface to retrieve the result in XML format. Although one can use the XML output with other software, not provided with the OQAFMA query agent, in order to generate a graphical display, the results – if any – were manually processed because this was more time effective due to the small size of the query answers.

2.2.3 Results and discussion

Query I

The first query can be reformulated as shown in Table 4. The first part provides mappings required for use in the `WHERE` clause and the `CREATE` clause says what to return as the query answer, "+" is the closure operator following the specified type of relation an arbitrary number of times, and a "*" combines the "+" with the optional operator "?" for path alternation [16], hence it also supports recursive queries. A section of the XML result is included in Table 5, which in the real answer contains 40 entities[4] from the FMA that are not greyed-out in Fig 3. The FMA list of parts that are cells (as accessed with Protégé) is graphically depicted with a checkbox after the entity, amounting to 12 types of cells. Obviously, the FMA/Protégé parts list is incomplete and positioning the taxonomy orthogonally *automatically* does reveal the lacunas in the FMA/Protégé list. This also indicates the difficulty of constructing a parts list *de novo* in contradistinction to querying the extant modelled knowledge. Thus, through combining existing encoded information differently by using granularity of the subject domain in formulation of the query, one can retrieve or discover 'new' information that otherwise would have remained inaccessible and hidden in the database. Put differently: there is more information captured in the FMA than meets the eye when only browsed with Protégé. Considering the query answer has 40 entities as opposed to the incomplete 12 listed in FMA/Protégé, one can use database querying for verification of, and finding gaps in, the FMA contents as a first step. Following this, it can be useful to investigate the possibilities to use this approach to aid (semi-)automatic creation of new relations in the FMA.

Separate from this emerge some design considerations. Should one include in the parts list all 'intermediate' entities, only the leaves, or maybe only the shared common ancestor? For instance, only *Reticulocyte*, *Echinocyte* and *MatureErythrocyte*, or also their common subsumer *Erythrocyte*, or only *Erythrocyte* but not its subtypes? The answer depends on the goal(s) one wants to achieve: leaves only, minimal amount of (top-level) parts, or to be able to reconstruct the relevant section of the taxonomic tree? Deciding either way will improve reliable querying of the FMA. A second design consideration is that one may prefer that an end-user is allowed to browse the FMA though a nice interface only, whereas querying may be more suitable for developers. On the other hand, querying the FMA allows much more flexibility for different domain

[3] I acknowledge the helpful suggestions from Peter Mork in writing the queries.

[4] I am not sure about the ontological status of *Lymphoblast*, i.e. if it is really part of blood; lymphoblasts are proginetors of lymphocytes and might be part of the *LymphaticSystem* instead. See also the discussion of Query IV on phased sortals.

experts to retrieve desired information in a wider range of scenarios, hence increasing usefulness of the FMA. Furthermore, the XML output obtained from OQAFMA easily can be ported to a GUI tool to make an appealing graphical visualisation such that one can have the best of both: powerful querying and an appealing interface for the domain expert.

Table 4. OQAFMA query to retrieve the parts of *Blood* that are cells.

StruQL syntax for *Query I*
`Blood binds to the symbol for blood.`
`Cell binds to the symbol for cell.`
`BloodCellPart binds to all parts of the blood that are cells.`
`BloodCellPartSubclasses binds to the subclasses thereof.`
`Result binds to the human-readable name of these symbols.`
`WHERE`
`Blood->":NAME"->"Blood",`
`Blood->"part"+->BloodCellPart,`
`BloodCellPart->":DIRECT-SUPERCLASSES"+->Cell,`
`Cell->":NAME"->"Cell",`
`BloodCellPart->":DIRECT-SUBCLASSES"*->BloodCellPartSubclasses,`
`BloodCellPartSubclasses->":NAME"->Result`
`CREATE`
`BloodShouldContain(Result)`

Query II

Querying "which cell type does not have a nucleus as part?" is for illustrative purpose to mention that such negation is not possible with OQAFMA. StruQL *does* support negation when NOT is combined with a atomic boolean expressions or a membership expressions [5], and assumes a closed-world assumption. In contradistinction, the FMA takes an open-world assumption because it is incomplete in representing human anatomy knowledge, hence absence of a relation between entities does not mean it does not exist in reality, only that there is no such relation defined in the FMA. Second, negation is not used in the FMA, hence no relation has been defined alike *not hasPart Nucleus*: this is textually addressed with naming terms, like *Non-nucleated cell*[5] to categorise those cells that do not have as part a nucleus. Also GALEN's GRAIL formalism does not deal properly with negation, but uses terms like "presence" and "absence" as a workaround. To use GALEN's workaround in the FMA, one has to have for the non-nucleated cell a predicate alike *isAbsent Nucleus*, or in DL: $NonNulceatedCell \doteq Cell \sqcap \forall isAbsent.Nucleus$.

Query III

The third query contains as test item the combination of two granular perspectives: structural and functional human anatomy. This is supposedly *not* possible, as the FMA contains only structural parts. However, the latter is not true: several functional entities *are* included in the FMA. Philosophically it is debatable [12] if functions exist

[5] Solution of this negation issue is being investigated (Anand Kumar, personal communication 30-8-2005).

Fig. 3. Difference in query answers on of *Query I*: Protégé/FMA browsing for blood cells (indicated with check-boxes) and OQAFMA answer (in black).

Table 5. Section of the output of *Query I* with OQAFMA, without the taxonomic structure of the blood cells.

```
<results>
<BloodShouldContain>
<Result>Reticulocyte</Result>
</BloodShouldContain>
<BloodShouldContain>
<Result>Echinocyte</Result>
</BloodShouldContain>
<BloodShouldContain>
<Result>Mature erythrocyte</Result>
</BloodShouldContain>
...
<BloodShouldContain>
<Result>Delayed type hypersensitivity-related T lymphocyte</Result>
</BloodShouldContain>
...
<BloodShouldContain>
<Result>Interleukin 2-activated natural killer cell</Result>
</BloodShouldContain>
</results>
```

at all and if yes what their nature is; one may prefer an ontological trade-off that aside from its ongoing investigations, one assumes their existence as perceived by biologists and correct or update its representation at a later date, if deemed necessary. Regarding the FMA, at the level of *BiologicalMacromolecule*[6], it contains entities such as hormones (*Epinephrine*, *Estrogen*,*Melatonin* etc), receptors (= protein with a specific function) like *DihydropyridineReceptor*, and other functional molecules like *SignalRecognitionParticle* (a ribonucleoprotein), and *BiogenicPeptide* with its subtypes (like *Vasopressin* and *Insulin*). The adherence to the structural perspective is better in the higher levels of granularity, although not always correct. For instance, the note in the *Pancreas* "clinical part" section mentions "Exocrine and endocrine need to be transferred to functional concepts", and of the two organ systems that have a definition, *RespiratorySystem* is a "*Functional system* which consists of structures involved in respiration" (emphasis added). The other organ system with a definition is *CardiovascularSystem* defined as "Organ system which consists of the heart, the systemic and pulmonary arterial and venous system, the lymphatic and the portal venous system." hence defined as a *structural* system. Even if one were to define a granular perspective for functional anatomy, then there still remains the lacuna to be filled to link the functional entities subsumed by *BiologicalMacromolecule* as part of higher-level entities as well as relating them as structural parts of coarser-grained entities, as none of the molecules are part of anything in the current version (of 2003).

[6] *BiologicalMacromolecule* is ontologically an incorrect label, as molecules are first subdivided into organic or anorganic – not biological and non-biological – and 'macro' has a fiat boundary; in addition, the FMA definition is inconsistent as it states that a macromolecule is build up from certain listed smaller molecules, but those molecules are also subsumed by *BiologicalMacromolecule*. It is beyond the scope to elaborate on this further here, but *BiologicalMacromolecule* should be read as *OrganicMolecule*, or at a later date with an extended FMA, be simply labeled as *Molecule*.

For these reasons, any query similar to *Query III* cannot return an answer that is correct with respect to reality – querying the system with OQAFMA and using a closed-world assumption, then the empty set it returns as query answer is correct for the presently encoded knowledge in the FMA.

Query IV

Retrieving the types of organ residing in the *Organ*-level that has as part a type of macrophage in the *Cell*-level faces several problems. Table 6 depicts the StruQL query and Table 7 the query answer. Note that it is not required to include the label with the type of macrophage in the result; in fact, the correct CREATE argument for the for *Query IV* is `CREATE OrganWithMacrophages(OrganSubclassesName)`, which returns *Liver* only, but the additional information is included for illustrative purpose. Except for *HepaticMacrophage* in the *Liver*, discovered during querying with OQAFMA and *not* when browsing the FMA in Protégé, the FMA lacks *any* relation from other types of macrophages to other entities (except for subtyping), even though in reality macrophages are located in/part of organs. Second, the FMA is phased sortal ignorant, which complicates the question: individual *Monocyte* cells that are part of blood 'dock' in tissue and due to the changes in micro-environment switch on/off genes resulting in other behaviour and morphology [28] [24] [20] and are then called instances of *Macrophage*; the scientific community does neither agree on the terminology nor on the status of the affected cells [11] [22]. From an ontological analysis viewpoint, it is clear that the monocyte/macrophage is a phased sortal (refer to [9] [10] for further information on phased sortals). Regardless phased sortalness of the entity, assuming the FMA contains all relations relevant for human structural anatomy, and using database technology that supports recursive queries, one can retrieve the answer. *Query IV* realises more advanced information retrieval compared to *Query I*: for each iteration, the query's high-level evaluation strategy is to traverse the taxonomic structure for macrophages as well as the isA for organs, then it checks if there is an orthogonally positioned $partOf$ path between any of the entities in the two sub-trees.

Table 6. OQAFMA query to retrieve organs that have as part a (subtype of) *Macrophage*.

Query for macrophages in organs
`WHERE`
`Macrophage->":NAME"->"Macrophage",`
`Macrophage->":DIRECT-SUBCLASSES"+->MacrophageSubclasses,`
`MacrophageSubclasses->":NAME"->MacrophageSubclassesName,`
`Organ->":NAME"->"Organ",`
`Organ->":DIRECT-SUBCLASSES"+->OrganSubclasses,`
`OrganSubclasses->":NAME"->OrganSubclassesName,`
`OrganSubclasses->"part"+->MacrophageSubclasses`
`CREATE`
`OrganWithMacrophages(OrganSubclassesName, MacrophageSubclassesName)`

It is worth noting that the FMA is at present one of the most comprehensive bio-ontologies, but it is incomplete both regarding the relations between entities and

Table 7. OQAFMA result for the query in *Table 4.7.*

Macrophages in organs
<results>
<OrganWithMacrophages>
<MacrophageSubclassesName>Hepatic macrophage</MacrophageSubclassesName>
<OrganSubclassesName>Liver</OrganSubclassesName>
</OrganWithMacrophages>
<results>

particularly concerning the *Cell*-level and below, although it is arguable if molecules belong in the subject domain – the present FMA cut-off 'level' is *BiologicalMacromolecule*[7]. The issue with negation is to be implemented correctly as well. Positioning orthogonally the structural and functional anatomy granular perspectives is an outstanding task in its entirety. With this in place, one can query for e.g. the types of hormones residing in the kidney.

3 Discussion

Two main issues are ontology development and diversification of usage of ontologies. In the past few years, much emphasis in (bio-)ontologies has been, and is being, put on ontological correctness of the modeled subject domain (e.g. [10, 21]), which is certainly still an important issue. However, *how* an ontology is to be made persistent for both off-line and online use has received much less attention, possibly because it ought not matter, although in practice it does.

3.1 Engineering

Looking at alternatives for storing ontologies in databases, adoption of OWL and RDF as standard ontology languages has its advantages concerning data and ontology interoperability, and ontology integration is facilitated when they are written in the same language. Also, its reasoning support is a welcome addition to improve ontology development. On the other hand, expressivity of query languages for both OWL and RDF lags behind relational database query languages. For instance, OWL-QL 'merges' network protocol approaches with querying distributed OWL-based knowledge bases [6], which is useful for the Semantic Web, but this is not necessarily a main query aim of a bio-ontology. RDF query languages include SPARQL [38] (a W3C draft), RQL [36] [13] and SeRQL [37]. SPARQL focuses on sub-graph isomorphisms for query answering of RDF graphs in databases, but for recursive queries it relies on

[7] The principal investigator Cornelius Rosse is convinced it does and that it is a part yet to be developed properly (personal communication, 1-5-2005). I am convinced that cell biology and biochemistry are separate disciplines in the core life sciences and that if one wants to use it for human anatomy (an applied life science, like the other specialisations in biomedicine), it is better to develop the former separately and afterwards to use those sections applicable to human anatomy, but not that human anatomy does or should take precedence. This because if the subcellular entities are ordered first to fit human anatomy, it will be less (or even: not) generic and less/not reusable for other subject domains, which is contrary to one of the purposes of developing an ontology.

SQL (and e.g. StruQL) and the DBMSs that support it. RQL supports "recursive traversal of class and property hierarchies (for advanced pattern-matching)" of trees with arbitrary depth [36]. SeRQL is a variation on RQL, and both return a bag as query answer [13], although with an additional `construct` one can rebuild the tree [37]. However, paths have to be specified with query formulation and RQL thus cannot 'discover' the path(s) between e.g. some macrophage and an organ. This considered, storing large ontologies in (relational) databases is a sensible option. It has an additional advantage when one desires to develop interoperable software and integrate databases, because querying over ontologies-stored-in-databases with data-databases is easier. But for this to realise, it is essential to have the ontology stored in a *good* database. A good database, as one can learn already in undergraduate courses in database development [18], has a meaningful conceptual model and ensures there is a separation of the database layer from the application layer in order to achieve relative implementation independence, thereby increasing reusability and simplifying maintenance of both. Unfortunately, FMA/Protégé meets neither. A workaround may be to generalise Mork et al's pre-processing operations [16], but this has to be carried out for each Protégé-database, therefore it will be more efficient and user-friendly if the Protégé software already takes care of this by modifying the development environment to map frames and slots to meaningful conceptual models. For instance, to create separate entities/relations in an ER or ORM model for each type of relation in the ontology and giving meaningful names to widely used attributes that end up in separate table columns, such as 'comment', 'language', 'user' and so forth. Aside from desirable optimization, an advantage is that then one can define database views for each granular level to simplify granular querying. For instance, the *Organ*-level will contain all taxonomic subtypes of *Organ*, and the tuples and values for the leaves in the taxonomic tree can be counted automatically and we would finally know how many organs we have.

A separate issue is the use of RDBMS. A common characteristic of ontologies are their hierarchical structures, be it taxonomies, partonomies, or e.g. a hierarchy of *developsFrom*. In this setting, recursive queries are necessary for useful information retrieval. SQL:1999 supports recursive queries, which has been gradually implemented in extant RDBMSs like Oracle, DB2, MS SQL, and in deductive databases. While MySQL is freely available, it lacks support for recursive queries, which may not be an issue when focusing on ontology creation and enlargement only, but is a serious limitation if one actually wants to use the ontology to go beyond finding an entity or retrieve an ambiguous distance to other entities.

3.2 Diversification of use

The demand for recursive queries brings afore the second point: diversification of uses of ontologies. Ontologies are not just an inert repository, even if the ontology is a reference ontology: if it offers nothing more than an online encyclopaedia, then why put in the effort to build the ontology? An encyclopaedia cannot answer questions like which macrophages are part of some organ, or be able to anticipate other *ad hoc* peculiar questions like hormones located in the kidney or molecules in the mitochon-

drion. An ontology can, provided it is implemented in a usable format. Ontologies can – and should – have a variety of uses (e.g. [3] [4] [17] [27]), which has to be taken into account during the development stage. Top-level ontologies may not need computational support as they are intended to guide modelling decisions off-line, but developing ontology-mediated query formulation and information retrieval, such as utilising the encoded knowledge better through granular queries, requires run-time access to the ontology. At present, this is not possible with the GO and FMA/Protégé implementations, but is with the reengineered-FMA OQAFMA. With the current engineering limitations, offering an ontology in several storage formats (RDF, OWL, XML, etc.) to simplify its use and reuse to match the desired task can be a feasible approach, with the preconditions to have a clear versioning system in place and informing the user what is lost or gained with one storage format compared to the other. Applying this to the FMA case study, the gain one obtains with OQAFMA is that it has 'added' the semantics of the application layer to the database such that one can examine the information contained in the FMA more easily and comprehensively, experiment with e.g. applied domain granularity to explore its contents in a novel way, and enables wider use and reuse of the FMA to, for instance, link it with other ontologies at the subcellular levels or query medical data through this ontology. Aside from this, now having the opportunity to position orthogonally the taxonomy with the partonomy is useful for discovering lacunas and inconsistencies shown in the browsers and can be used for semi-automatic addition of the newly suggested relations.

Aside from information retrieval scenarios and sophisticated querying, the desire for and offers of 'reasoning support' for ontologies is underspecified: what exactly is available where and what type of reasoning does the user want/need? Automation for classification of a taxonomy, automated instance categorization, tree comparisons, test a hypothesised theory or relation for satisfiability, finding missing relationships, more? A *finer-grained goal specification* as part of the requirements analysis process will not only help deciding how one best can store the ontology with the presently available technology, but also provide an impetus for logicians and software developers to address and solve the most pressing users' needs first.

4 Conclusions

Current persistent storage implementations of large bio-ontologies fall short in meeting usability and reusability requirements. Some main limitations were demonstrated with two representative ontologies-stored-in-databases: the Gene Ontology and the Foundational Model of Anatomy. The Gene Ontology is in its current database format not usable for performing advanced querying. Comparing browsing the FMA in Protégé with OQAFMA querying, the latter is certainly a useful improvement, especially for testing the information contained in the FMA and to experiment with applied domain granularity in a relatively easy manner to explore and illustrate its potential. However, intersecting – or positioning orthogonally – taxonomic relations with the partonomy currently is more promising for finding lacunas and inconsistencies and may be used for possible (semi-)automatic addition of the newly suggested relations than to retrieve reliable, consistent, and complete information.

Improvements are possible with regards to ontology development environment software, taking into account general database development good practice, using RDBMSs that support expressive and recursive queries, and finer-grained specification of goals of what the ontology and system it may be part of are and may be intended for. We demonstrated the added-value of a variation of querying, granularity, which, by availing of extant database technology and implementation (OQAFMA), enables retrieval of more information quicker and can be used for improving and enlarging an ontology compared to the more common bio-ontology usage and implementations stored in relational databases.

Using applied domain granularity with other ontologies, a fully specified domain granularity framework, and the functions defined in the preceding paragraphs will take even more work and time to realise. Current and future work involves improving ontology usage and developing granularity further to provide a reliable and reusable framework for granular information retrieval.

Acknowledgments
I thank Peter Mork, Sergio Tessaris, and Scott Marshall for their helpful comments on querying.

References

1. Borgida, A., Lenzerini, M., Rosati, R. Description Logics for Data Bases. In: *Description Logics Handbook*, Baader, F. Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds). Cambridge University Press, 2002. pp 472-494.
2. Brouard, F. Recursive Queries in SQL:1999 and SQL Server 2005. SQLServerCentral.com. 25-04-2005. http://www.sqlservercentral.com/columnists/fBROUARD/recursivequeriesinsql1999andsqlserver2005.asp
3. Catarci, T., Dongilli, P., Di Mascio, T., Franconi, E., Santucci, G., Tessaris, S. An Ontology Based Visual Tool for Query Formulation Support. In: *Proceedings of ECAI 2004*, Valencia, Spain, August 2004.
4. Daraselia, N., Egorov, S., Yazhuk, A., Novichkova, S., Yuryev, A., Mazo, I. Extracting Protein Function Information from MEDLINE Using a Full-Sentence Parser. *Proceedings of the Second European Workshop on Data Mining and Text Mining in Bioinformatics*. 2004. pp11-18.
5. Fernández, M.F. StruQL Query language. http://www.research.att.com/∼mff/strudel/doc/sitegraph.genoid_8.html.
6. Fikes, R., Hayes, P., Horrocks, I. OWL-QL – a language for deductive query answering on the Semantic Web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2004, 2(1):19-29.
7. Gene Ontology Consortium. The Gene Ontology (GO) project in 2006. *Nucleic Acids Research*, 2006, 34:D322-D326.
8. Guarino, N. Formal Ontology and Information Systems. *Proceedings of Formal Ontology in Information Systems (FOIS'98)*. Amsterdam: IOS Press, 1998.
9. Guarino, N., Welty, C. A Formal Ontology of Properties. *Proceedings of 12th Int. Conf. on Knowledge Engineering and Knowledge Management*, Dieng, R (ed.), Lecture Notes in Computer Science, Springer Verlag 2000.
10. Guarino, N., Welty, C.A. An overview of OntoClean. In: *Handbook on ontologies*. Stab, S., Studer, R. (eds.). Springer Verlag, 2004. pp151-159.
11. Hoffbrand, A.V., Pettit, J.E. *Atlas of Clinical Heamatology*. 3rd ed. Elsevier/Mosby, 2000.
12. Johansson, I., Smith, B., Munn, K., Tsikolia, N., Elsner, K., Ernst, D., Siebert, D. Functional Anatomy: A Taxonomic Proposal. *Acta Biotheoretica*, 2006, preprint: http://ontology.buffalo.edu/medo/Functional_Anatomy.pdf.

13. Karvounarakis, G., Magkanaraki, A., Alexaki, S., Christophides, V., Plexousakis, D., Scholl, M., Tolle, K. RQL: A Functional Query Language for RDF. In: *The Functional Approach to Data Management: Modelling, Analyzing and Integrating Heterogeneous Data*, Gray, P.M.D., Kerschberg, L., King, P.J.H., Poulovassilis, A. (eds.), LNCS, Springer-Verlag, 2004.
14. Keet, C.M. A taxonomy of types of granularity. *IEEE Conference on Granular Computing (GrC2006)*, 10-12 May 2006, Atlanta, USA.
15. Kumar, A., Smith, B., Novotny, D.D. Biomedical Informatics and Granularity. *Comparative and Functional Genomics*, 2005, 5(6-7): 501-508.
16. Mork, P., Brinkley, J.F., Rosse, C. OQAFMA Query Agent for the Foundational Model of Anatomy: a prototype for providing flexible and efficient access to large semantic networks. *Journal of Biomedical Informatics*, 2003, 36: 501-517.
17. Mungall, C.J. Obol: integrating language and meaning in bio-ontologies. *Comparative and Functional Genomics*, 2004, 5(6-7):509-520.
18. Newton, M., Armstrong, S., Kwee Brown, J., Cooke, I., Field, M., Findlay, J., Lewis, S., Taylor, J., Yazdanjoo, K. *M358: Relational Databases*. Open University UK. 2001.
19. Sampath, S. Recursive Queries in SQL Server 2005. SQLServerCentral.com. 3-3-2005. http://www.sqlservercentral.com/columnists/sSampath/recursivequeriesinsqlserver2005.asp
20. Scheuerer, B., Ernst, M., Drrbaum-Landmann, I., Fleischer, J., Grage-Griebenow, E., Brandt, E., Flad H.-D., Petersen, F. The CXC-chemokine platelet factor 4 promotes monocyte survival and induces monocyte differentiation into macrophages. *Blood*, 2000, 95(4):1158-1166.
21. Smith, B., Ceusters, W., Klagges, B., Kohler, J., Kumar, A., Lomax, J., Mungall, C.J., Neuhaus, F., Rector, A., Rosse, C. Relations in Biomedical Ontologies. *Genome Biology*, 2005, 6:R46.
22. Stvrtinová V., Jakubovský J., Hulín, I. Inflammation and Fever. In: *Pathophysiology: Principles of Diseases*. Academic Electronic Press, 1995. http://nic.sav.sk/logos/books/scientific/Inffever.html.
23. Tange, H.J., Schouten, H.C, Kester, A.D.M. and Hasman, A. The Granularity of Medical Narratives and Its Effect on the Speed and Completeness of Information Retrieval. *Journal of the American Medical Informatics Association*, 1998, 5(6): 571-582.
24. Tuttle, D.L., Harrison, J.K., Anders, C., Sleasman, J.W., Goodenow, M.M. Expression of CCR5 Increases during Monocyte Differentiation and Directly Mediates Macrophage Susceptibility to Infection by Human Immunodeficiency Virus Type 1. *Journal of Virology*, 1998, 72(6):4962-4969.
25. Venigalla, S. Expanding Recursive Opportunities with SQL UDFs in DB2 v7.2. 1-3-2005. http://www-128.ibm.com/developerworks/db2/library/techarticle/0203venigalla/0203venigalla.html
26. Voronkov, A. *Advanced Deductive Databases*. International Masters Programme on Computational Logic. http://rpc25.cs.man.ac.uk/voronkov/teaching/dresden/2002/index.html.
27. Wolstencroft, K., Lord, P., Tabernero, L., Brass, A., Stevens, R. Using ontology reasoning to classify protein phosphatases [abstract]. *8th Annual Bio-Ontologies Meeting 2005*, 24 June 2005, Detroit, USA.
28. Yang, J.-H., Sakamoto, H., Xu E.C., Lee, R.T. Biomechanical Regulation of Human Monocyte/Macrophage Molecular Function. *Am. J. of Pathology*, 2000, 156:1797-1804.
29. Zukerman, I., Albrecht, D., Nicholson, A., Doktor, K. Trading off granularity against complexity in predictive models for complex domains. In: *Proceedings of the 6th International Pacific Rim Conference on Artificial Intelligence (PRCAI 2000)*.
30. AmiGO. http://www.godatabase.org/cgi-bin/amigo/go.cgi.
31. DAG-Edit. http://www.godatabase.org/dev/.
32. Foundational Model of Anatomy (FMA). 2003. http://fme.biostr.washington.edu:8089/FME/index.html.
33. Gene Ontology Consortium. http://www.geneontology.org.
34. Gene Ontology – optimisations. http://www.godatabase.org/ dev/sql/modules/go-optimisations.sql.
35. Ontology Query Agent for the Foundational Model of Anatomy. http://sig.biostr.washington.edu/projects/oqafma/.
36. RQL. http://139.91.183.30:9090/RDF/RQL/.
37. Sesame RQL. http://www.openrdf.org/doc/rql-tutorial.html.
38. SPARQL. http://www.w3.org/TR/rdf-sparql-query/.