“Evolution and Future Trends for XBRL Development”
KU XBRL 2013

Herm Fischer

Abstract

The XBRL Abstract Model represents an effort to step above the issues of syntax, specific models and taxonomies, and software strategy. It tries to bridge the modeling semantics, financial ontology, data points modeling, and other XBRL communities, with a view to fitting into big-iron databases and OLAP/BI systems. Development got to a point where there are ideas worth prototyping to get initial feedback before moving ahead. Strategies for mapping to relational database (as used by XBRL-US) and to graph-oriented databases (as used by social networking) will be compared.

Validation likewise represents a step where alternative technologies each have their place. At the onset of XBRL, it was felt important to have something like Excel’s formulas for the business users, and XML’s query facilities, for the techies. XBRL Formula developed over a decade to provide a XML-centric declarative way of specifying the semantic constraints of a taxonomy, and has come into significant use in the European and Asian ecosystems. In the US, the more business-rules focused approach of the Sphinx language is fervently supported. The author’s contributions to the open source community find a third angle important, raw speed when supporting production environments, experienced in a Python-API based set of validation implementations. Together these provide a menu of choices of sufficient maturity and capability to support the XBRL ecosystem.

From XML to relational and then graph-based databases

We’ll examine the progression of XBRL from a technology for single business reports (XBRL instances) and their isolated validation and analysis, to databases based on relational technology (rigid slicing and dicing or extraction to fixed models), and then to databases based on massively-scale graphs (for social networking technology).

XBRL in XML

XML represents a way of formally modeling the syntax of an interchange document and validating the syntactical contents. It defines strongly typed data structures based on coupled models of typing and structural particles. Its stake holders use XML for message interchange, software as a service implementations, well-structured web interfaces, and general serialization of software artifacts.

XML is a character-based interface that serializes objects in a character-based stream which needs parsing and validating (compared to databases where things are stored in binary form for efficient processing).

XBRL represents a way of formally modeling the semantics of business information interchange, with the “Extensible” in its name most distinguishing difference from XML and technologies found in relational databases. The extensibility of XBRL comes from a compositional approach to modeling the data of each individual business report. Each instance of a report not only serializes the data, but also composes its model by a taxonomy set. The taxonomy of a US SEC filing extends and customizes the standard reporting information to represent how a specific business reports its financial performance and corporate actions, each different from the other. The model changes as regulatory and business requirements change, and varies according to the business report style of the reporting entities. No two companies, or reports provided in different time periods, are likely to have identical structures. That's the feature and benefit of XBRL (and its challenge).

Software that loads and processes an XBRL instance with its supporting taxonomy structure, from the character-based XML files, takes serious computing resources, from megabytes to gigabytes of memory and seconds to minutes per business report. Some filings of prudential data are multiple gigabytes in size. None can be loaded and processed in the memory footprint or timeframe allocated for a web service call, but that's an issue of storing data and taxonomy information in unparsed source form.

Relational technology

Databases using relational technology are pervasive. They are suited to well-defined data models and highly normalized tables of data. OLAP implementations provide support to dimensional analysis, but of defined-in-advance model structures. XBRL is the opposite, data where each new instance provides a newly different model (at least with some extension difference), and it's most troublesome to normalize. Many parties have implemented relational databases of XBRL data. For prudential (credit reporting) systems, and environments where the structure of business reports is not significantly extended by each submitter, this is much easier than for financial reporting, where reports seriously differ from each other.

The XBRL-US relational database [1] represents an extremely thorough and robust representation of XBRL, at a syntactical level. The XML artifacts of XBRL are captured in a highly normalized set of tables that are sufficient for retention of present SEC filings. It's not intended to handle non-US prudential filings, corporate action filings, or massively huge filings. Its highly normalized tables provide good access to retrieve and reconstruct single filings but need difficult query support for cross-filing semantics (where data is bound to linkbase graphs of different structures). It's not known to be in significant use by any stakeholders. The XBRL-US database structure is hosted by XBRL-US as the "XBRL-US Public Database", and available to their XBRL-Challenge open source tool developers.

Figure 1 shows an Entity-Relationship model of the relational database. It has 35 tables in a highly-normalized structure well-suited for SQL (relational) algebra. The brown background tables represent data about each XBRL filing (this database is usually used for US SEC filings, which capture metadata about the filing, from the RSS feed of the SEC public website). The green background tables represent the XBRL taxonomy content, focusing on element (concept) information and organization of relationships as determined by an XBRL processor. The blue background tables represent the instance data of the business report in terms of the facts reported and their contextualizing information.

This design is tailored for the SEC structure of filings, and could be used for other applications, but with some extension or tweaking required for taxonomies that use tuples, typed dimensions, table rendering, formula linkbases, and versioning modules not in present in XBRL use by SEC forms filings.

The XBRL-US design places the burden of XBRL parsing and semantic resolution at the database loading stage, so the querying and processing of data need not require use of a processor. Implementation of the Arelle interface required use of the XBRL processor for DTS (discoverable taxonomy set) resolution, linkbase relationships compilation and tree walking, and XBRL instance processing (correlating instance fact concepts and dimensions to the DTS object model). (Processor features for extension modules, such as table linkbase and formula compilation, are not used, but would be required for non-SEC filings such as FINREP or Solvency II.)

A key feature of relational technology today is support for limited forms of OMG's OLAP. OLAP supports dimensional analysis, and was designed as a general technology. Relational implementations require specification of the dimensional model at the design of the database schema, but XBRL instances provide dimensional models with each submission. An attempt was made to prototype with dynamic relational OLAP features, it did not hold promise.

OLAP has been successfully used on some XBRL projects where all filings have the same taxonomy (model), so that a predefined dimensional analysis model can be used.

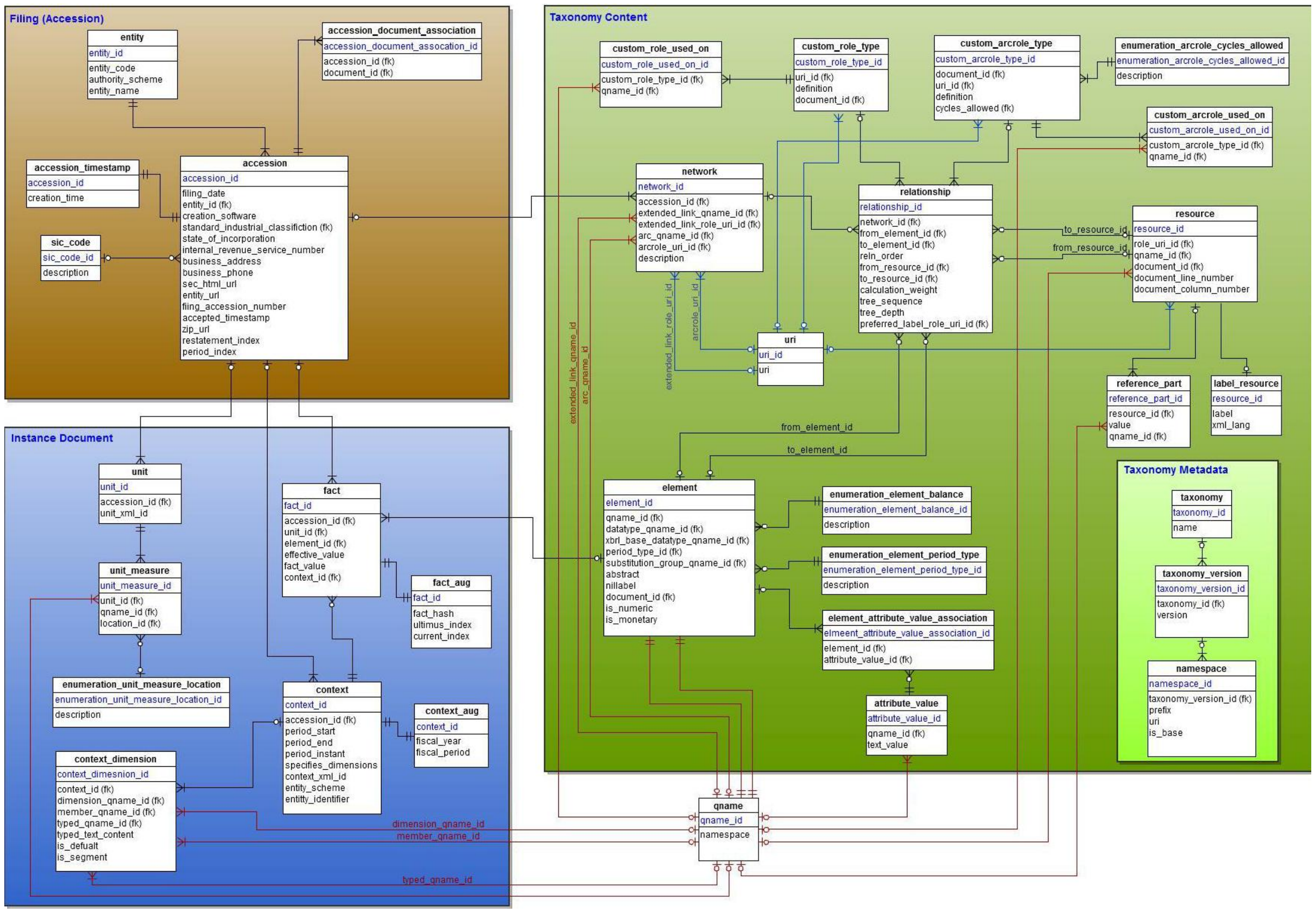


Figure 1. XBRL-US Data Dictionary diagram

Abstract Model as a meta-model for a graph database

The XBRL Abstract Model represents an effort to step above the issues of syntax, specific models and taxonomies, and software strategy. It tries to bridge the modeling semantics, financial ontology, data points modeling, and other XBRL communities, with a view to fitting into big-iron databases and OLAP/BI systems. The Abstract Model is based on OMG's CWM, of which OLAP is a subset, but the dimensions model in XBRL calls for a dynamic form of OLAP that hasn't been implemented by relational databases. Development got to a point where there are ideas worth prototyping to get initial feedback before moving ahead.

The Abstract Model as specified is a meta-model, defining the framework for constructing an instance model that can be implemented in an actual database. In this task, we chose to make an instance model closely track the organization of the meta model's data dictionary and instance features, but did not complete building out an instance model of the dimensional validation or data typing features. The initial point of the instance model in this project is to provide an efficient basis for investigating performant scalable access to large-scale XBRL data without the time and memory sever footprint needed if XBRL source information were loaded.

We chose to prototype an instance of the Abstract Model based on graph database technology. The author first became aware of this approach from a 1979 paper by Dr. Codd [2], on the need to extend relational technology with what today might be seen as the kind of graph database used in social network applications. This paper was used to guide a previous standardization effort (Portable Common Tool Environment) into an object model very similar to XBRL, and here is used as the basis for a graph instance model for XBRL semantics.

In the Abstract Model PWD 2.0 specification [3], two model layers are discussed, a primary model which is a meta-model, driven by semantics, and independent of syntax. Secondary models are provided, with some detail for XBRL syntax, Global Ledger syntax, and Charlie Hoffman's Financial model.

In the prototype project, the instance model used is an attempt to capture the primary model's data dictionary and instance models more or less as they are, with the addition of features from the XBRL-US model reflecting its thoroughness in representing Filing Accession data and such.

Model elements from the Abstract model are shown in Figure 2. The colored metaclasses are represented in the prototype project (except for AxisOrdinate, from the XBRL table linkbase, which has evolved in that effort and will be added later).   Each of these metaclasses will be examined in turn in their use.

The *DataPoint* metaclass defines a reportable item of business information contextualized by a set of aspects that identify or describe the item (this is an XBRL primary item fact), and a corresponding data point value (the reported item of business information).

The *Aspect* metaclass defines an identifying or describing characteristic of a business information item data point (which is instantiated as a Fact's AspectValue, and may be organized into an ontology and internal or external semantics model). The *AspectValue* metaclass defines a specific instance of value of an Aspect. The value is an identifying or classifying characteristic that a potentially realizable (or excluded) data point may have. If values are enumerable and known in advance, they may be organized into AspectValueRelationships which may form an ontology for the Aspect. If they have associated internal or external descriptions, they may form a semantics model for a related data point. An instance of AspectValue is also used to model the data point's value.

The *RelationshipType* metaclass defines a classification of relationships. The *RelationshipSet* metaclass defines a collection of relationships. The *Relationship* metaclass defines a logical relationship between pairs of Aspects, AspectValues, DataPoints, Resources, and between Aspect and AspectValue, Aspect or DataPoint and Resource.

The *AxisOrdinate* metaclass defines a correlation between AspectValue and *Table*, representing a view rendering. The rendering semantics of XBRL have evolved some of the terms since this modeling effort, and are not yet implemented by the prototype instance mode.

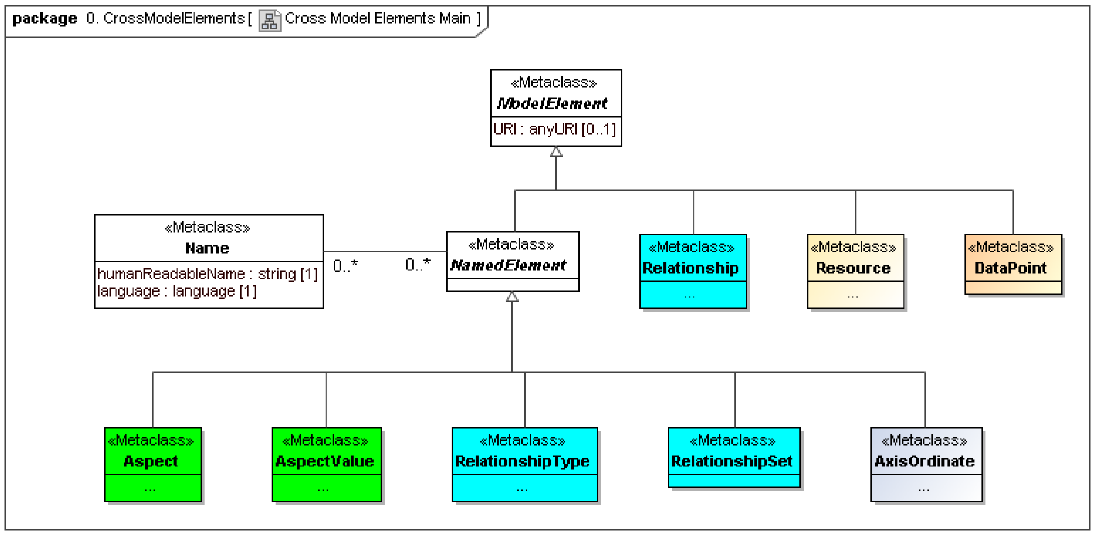


Figure 2. Abstract model elements.

The Abstract Model and prototype graph database both model aspects, Figure 3, which are a key feature of both the data dictionary and the instance data points. In the case of the data dictionary, these define the classes (data types) of reported business information and contextualizing information known to the data dictionary (dimensions). These are the classes of identifying or descriptive characteristics of data points, which may be reported as business information items (Facts). In the instance data points, aspect values are the reported information items and contextualizing information such as periods and units about the reported information.

Aspects may be enumerable or non-enumerable. If enumerable (such as for XBRL explicit dimensions) there is a fixed, finite, and known-in-advance set of AspectValues. If non-enumerable (such as with most fact values and XBRL typed dimension values), AspectValues are not specified within a data dictionary (class definition) and may be of a nature to be known only when business information is made available (such as from names, numbers, locations, and aspects whose instances are structured data).

For the case of enumerable Aspects, organized in AspectValueRelationships representing an identifying nature of a DataPoint (such as an element name or dimension) the AspectValueRelationships of that Aspect form an ontology (model of states of being), and may with other Aspects of the DataPoint, form a semantics model of the data point, if described formally (internally or externally). Otherwise, Aspects such as period date (such as when an account balance is taken), reporting entity (such as the business for whom the business information is reported), have no traditional ontological significance (they don't model states of the business information).

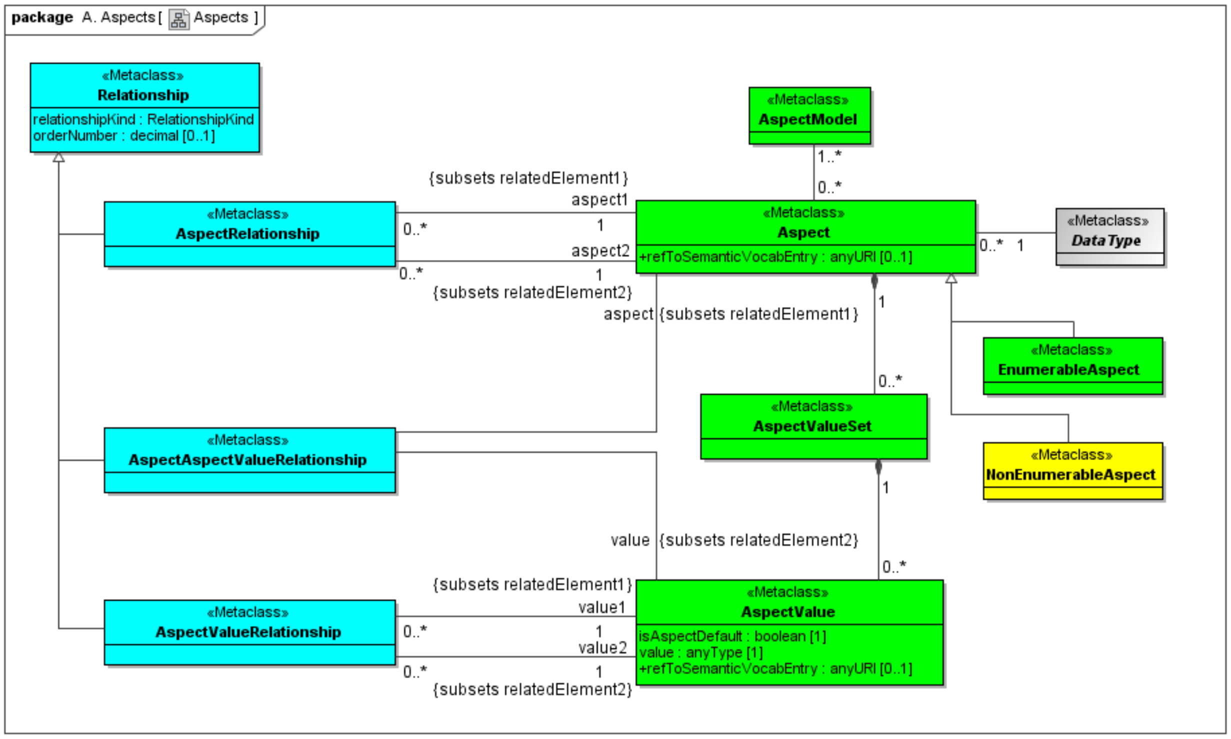

package A. Aspects [Aspects]
«Metaclass» Relationship
relationshipKind : RelationshipKind
orderNumber : decimal [0..1]
«Metaclass» AspectModel
1..*
0..*
{subsets relatedElement1}
aspect1
«Metaclass» AspectRelationship
0..*
1
aspect2
0..*
1
{subsets relatedElement2}
«Metaclass» Aspect
+refToSemanticVocabEntry : anyURI [0..1]
0..* 1
«Metaclass» Data Type
aspect {subsets relatedElement1}
1
0..*
«Metaclass» EnumerableAspect
«Metaclass» AspectValueSet
«Metaclass» NonEnumerableAspect
«Metaclass» AspectAspectValueRelationship
1
value {subsets relatedElement2}
0..*
{subsets relatedElement1}
value1
«Metaclass» AspectValue
isAspectDefault : boolean [1]
value : anyType [1]
+refToSemanticVocabEntry : anyURI [0..1]
«Metaclass» AspectValueRelationship
0..*
1
value2
0..*
1
{subsets relatedElement2}

Figure 3. Abstract model aspects, values and relationships

Relationships, Figure 4, occur both in the data dictionary (of taxonomies) and instance data (connecting facts to derived facts and footnotes). Relationships relate aspects, aspect values, resources, and datapoints.

The Relationship metaclass defines a graph of values of a RelatableElement, providing any hierarchy and organization. The graph nature of this is difficult to model in the relational database, and easy to model in a graph database. In the case of the relational database, the graph is flattened by a depth-first treewalk (ignoring cycles, which are allowed in some cases by XBRL), and stored in a table as graph nodes with depth level, sequence ordinate in depth-first tree walk, and from- and to- element surrogates (ids). In the graph model edges implement relationships with the database features.

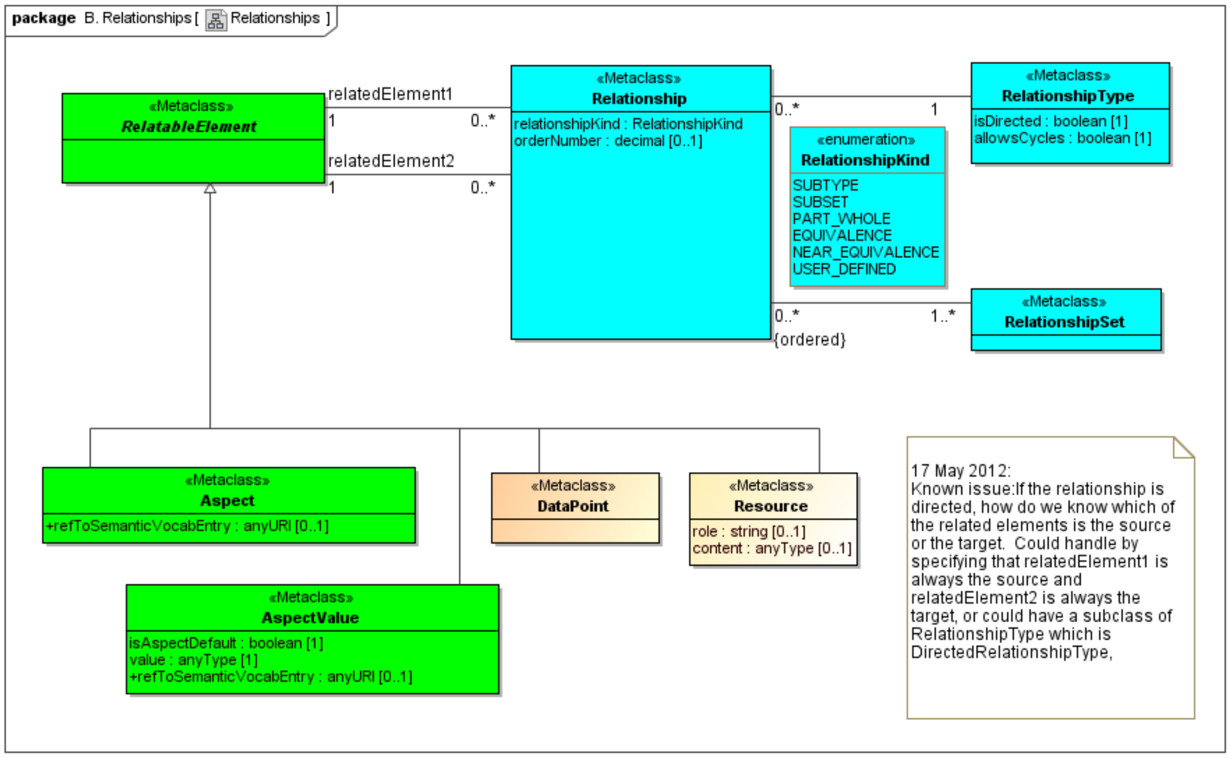

package B. Relationships [Relationships]
«Metaclass»
RelatableElement
relatedElement1
1
0..*
relatedElement2
1
0..*
«Metaclass»
Relationship
relationshipKind : RelationshipKind
orderNumber : decimal [0..1]
0..*
1
«Metaclass»
RelationshipType
isDirected : boolean [1]
allowsCycles : boolean [1]
«enumeration»
RelationshipKind
SUBTYPE
SUBSET
PART_WHOLE
EQUIVALENCE
NEAR_EQUIVALENCE
USER_DEFINED
0..*
{ordered}
1..*
«Metaclass»
RelationshipSet
«Metaclass»
Aspect
+refToSemanticVocabEntry : anyURI [0..1]
«Metaclass»
DataPoint
«Metaclass»
Resource
role : string [0..1]
content : anyType [0..1]
«Metaclass»
AspectValue
isAspectDefault : boolean [1]
value : anyType [1]
+refToSemanticVocabEntry : anyURI [0..1]
17 May 2012:
Known issue:If the relationship is directed, how do we know which of the related elements is the source or the target. Could handle by specifying that relatedElement1 is always the source and relatedElement2 is always the target, or could have a subclass of RelationshipType which is DirectedRelationshipType,

Figure 4. Abstract model relationships.

Resources, Figure 5, represent things that describe or specify Elements of the Abstract Model. Resources may be labels, documentation, references to specifications and legislation, and components of models that provide rendering and viewing, validation, transformation, documentation, and support of instances of abstract model features. Resources may be contextualized by a data type, may have a role and content (such as label).

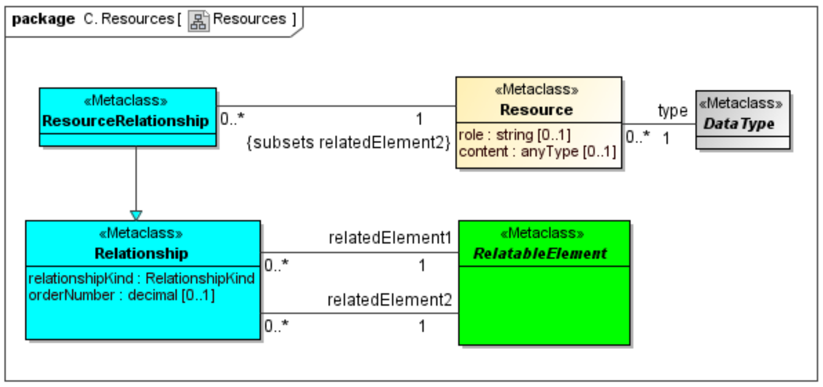


Figure 5. Abstract model resources.

This instances model, Figure 6, captures how instances of a DataPoint are identified or described by the value combinations of the identifying Aspects (from the ontology of the data dictionary model)

DataPoints may be modeled by Aspects in many ways, with Aspect modeling choices reflecting the architecture of a reporting system, politics of organizations and information technologists, opinions of the designers, and experience-set of the implementers. Earlier tradition has been to designate each ontologically identified Aspect class (such as a business reporting chart of accounts Aspect) as a DataPoint with a unique "primary item" AspectValue (e.g., a separate XBRL concept name for each chart of accounts entry). Aspect modeling provides flexibility in such approaches, allowing the architect to reduce a or expand a collection of AspectValues between aspect classes.

The DataPoint metaclass defines a class that is a unit of reported information (either an element of such information with FactValue or a structured aggregate of such information). DataPoint instances of reportable business information are Facts. A DataPoint may have an AspectValue instance that is its FactValue (reported business information).

Each DataPoint has a set of AspectValues that identify or describe the DataPoint. Each DataPoint is related to the AspectValues (that identify and describe it) whereas each DataPoint is related to the set of Aspects (that have AspectValues for the Fact instance). A DataPoint may have a StructuringRelationship with other DataPoints according to its Type.

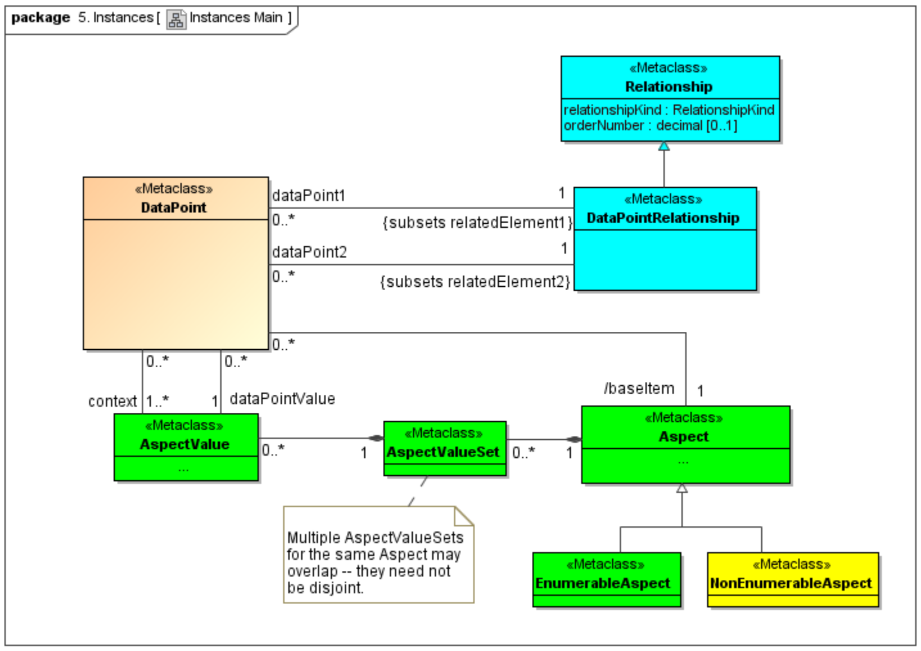

package 5. Instances [Instances Main]
«Metaclass» Relationship
relationshipKind : RelationshipKind
orderNumber : decimal [0..1]
«Metaclass» DataPoint
dataPoint1
0..*
{subsets relatedElement1}
1
«Metaclass» DataPointRelationship
dataPoint2
0..*
{subsets relatedElement2}
1
0..*
/baseItem
1
0..*
0..*
context
1..*
1
dataPointValue
«Metaclass» AspectValue
0..*
1
«Metaclass» AspectValueSet
0..*
1
«Metaclass» Aspect
Multiple AspectValueSets for the same Aspect may overlap -- they need not be disjoint.
«Metaclass» EnumerableAspect
«Metaclass» NonEnumerableAspect

Figure 6. Abstract model instances.

The document model represents a metamodel of a container for reported information and its data dictionary. In the reported project, the document model was extended to have accession submission properties (from the XBRL-US relational model) and sufficient information about taxonomy and linkbase packaging to represent shared common components and files in the real world. This allows the graph model of multiple submissions to interrelate common aspect classes of reported information by traversing edges form data points in one submission through the defining aspect class graph vertices to the data points of other submissions.

Metamodels not yet captured in prototype implementation

The valid combinations meta-model, Figure 7, represents the dimensions component of OMG OLAP. Its purpose is to model the semantics of dimensional validation. It does this by identifying cubes and valid cube regions for each reportable primary item Aspect class. As a meta-model, it conveys needed validation information, but to directly reflect this meta-model in the instance of a graph database would require expanding the full algebra of valid combinations by the reportable aspects and their valid combinations (this can be a huge hyperspace). In XBRL semantics a compact graph notation is used to represent this in dimensions tree graphs (which are captured in the project as relationships, so they can be conveyed).

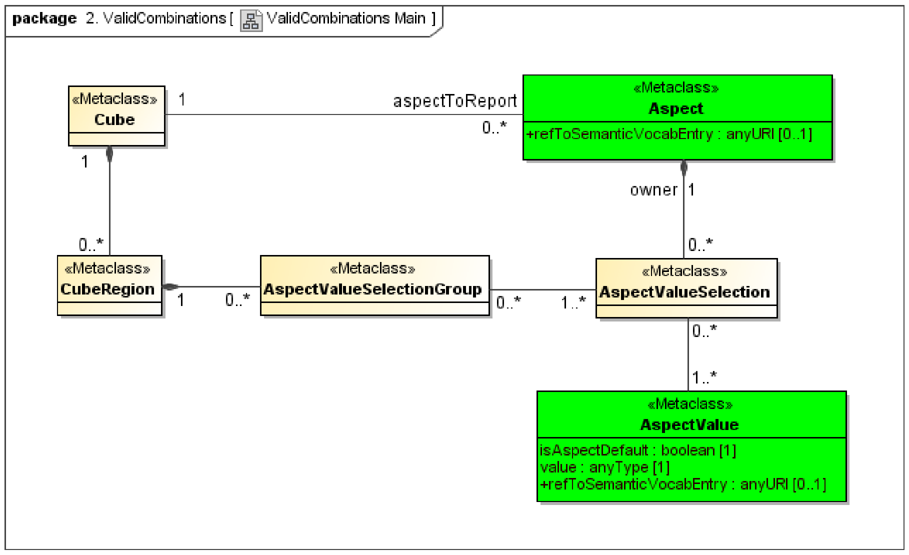

package 2. ValidCombinations [ValidCombinations Main]
«Metaclass» Cube
«Metaclass» Aspect
+refToSemanticVocabEntry : anyURI [0..1]
aspectToReport
1
0..*
owner
«Metaclass» CubeRegion
«Metaclass» AspectValueSelectionGroup
«Metaclass» AspectValueSelection
«Metaclass» AspectValue
isAspectDefault : boolean [1]
value : anyType [1]
+refToSemanticVocabEntry : anyURI [0..1]
1..*

Figure 7. Abstract model valid combinations

The data typing meta-model represents the aspect class compositional and constraining type features. It's likewise needed to model the semantics of data composition and validation, but not yet modeled in the reported project.

Initial use of the prototype graph database will be retrieval of information loaded from prior validated submissions (such as to the U.S. SEC). Since they were known to be validated in their XBRL syntax form, they are not intended to be re-validated (from the graph model form). Validation artifacts (messages and best practices advice from parsing the XBRL syntax) are intended to be captured and preserved in the graph model.

Instance semantic graph database from meta-model

The initial experimental instance model of the Abstract Model meta-model is shown in Figure 8. It's not an attempt to reproduce exactly the meta-model structure, but instead a practical equivalent that can be queried by the mechanisms available in graph databases.

A choice was made to use a graph database from the social networking field, designed for large data and having implementations using HADOOP technology. Titan [4] was chosen because it's intended for massive-scale graphs and support for advanced graph-based distributable query languages (such as Gremlin [5]).

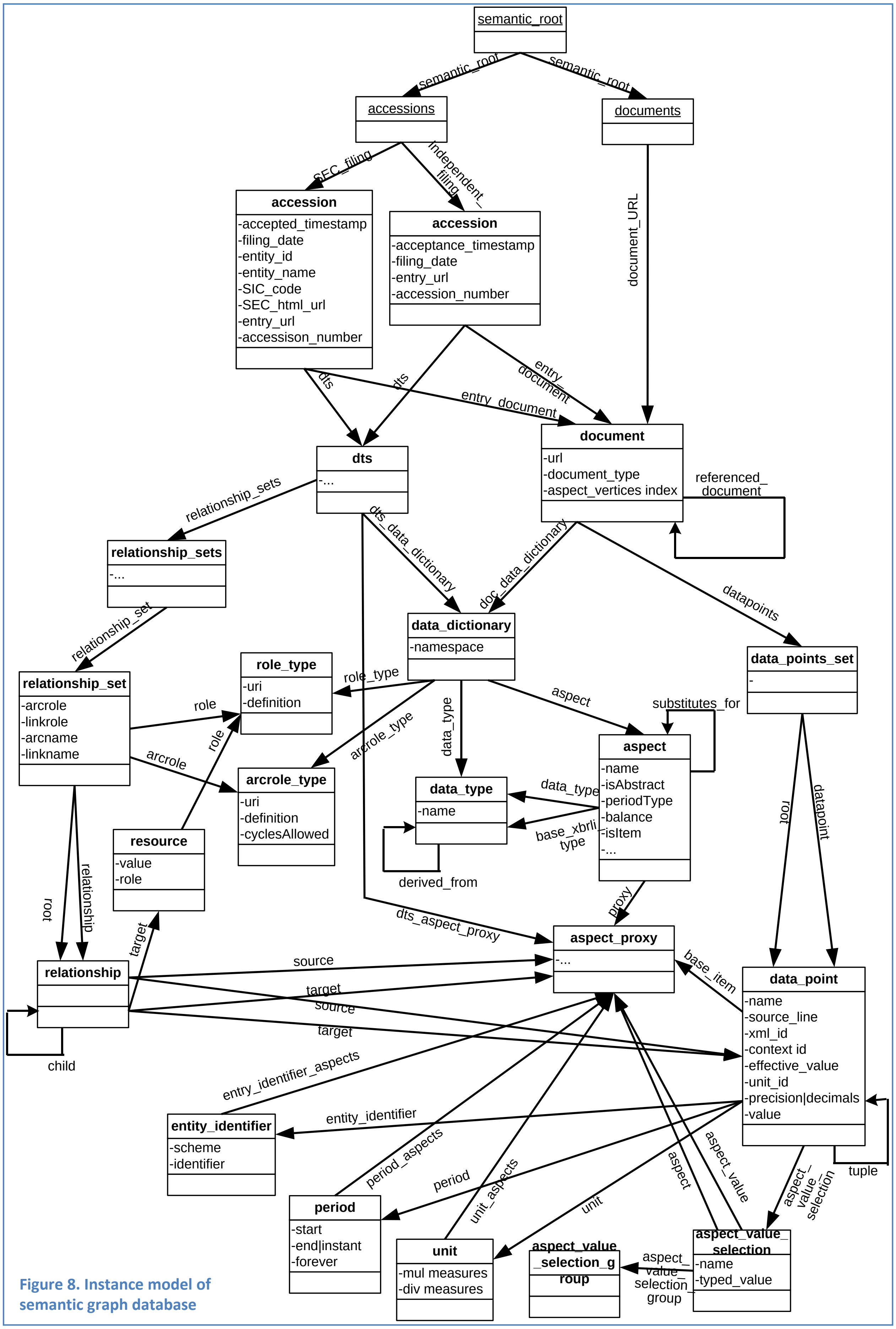


Figure 8. Instance model of semantic graph database

Graph database usage

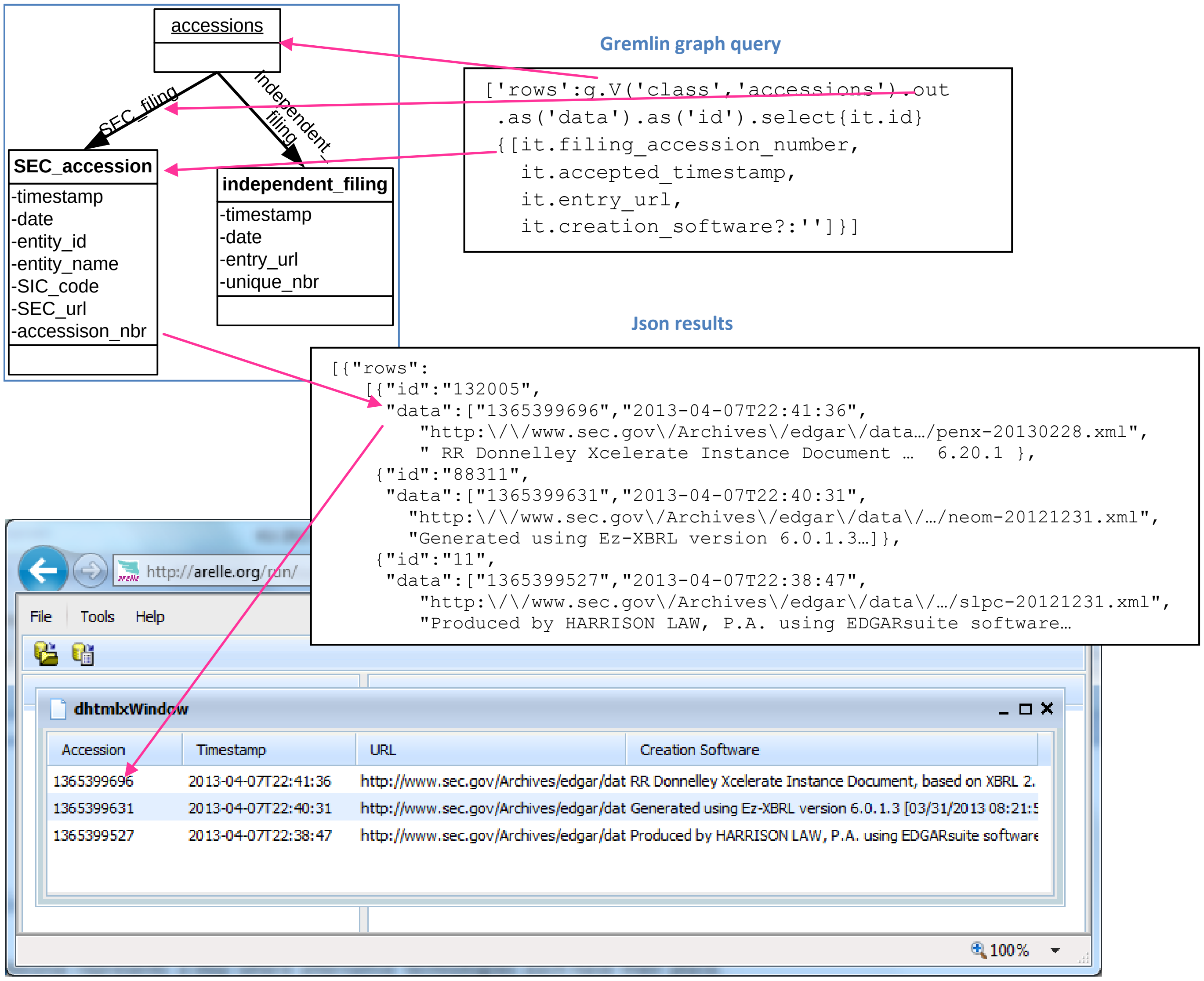
The Arelle open source project [6], [7] has implemented two interfaces, to the US-GAAP relational model and to a graph database based on the XBRL Abstract Model. It bulk-loads filings from the US SEC RSS feed, or individually loads taxonomies and instances as needed.

The relational model is intended to support stake holders who have already implemented uses, it has primarily been made available for the past two years to the open source submissions to the XBRL-US Challenge contests. The graph database is intended to by highly scalable, being designed for, and used in, social networking projects.

The graph database is intended to support web and cloud needs for fast browsing and cross-instance graph-query processing, and to experiment with implementations supportive of large-scale submissions.

Listing Accessions

This use case populates a table with a list of accessions based on the graph vertices and edges and Gremlin query, which returns results in the format required by the browser table construction json:



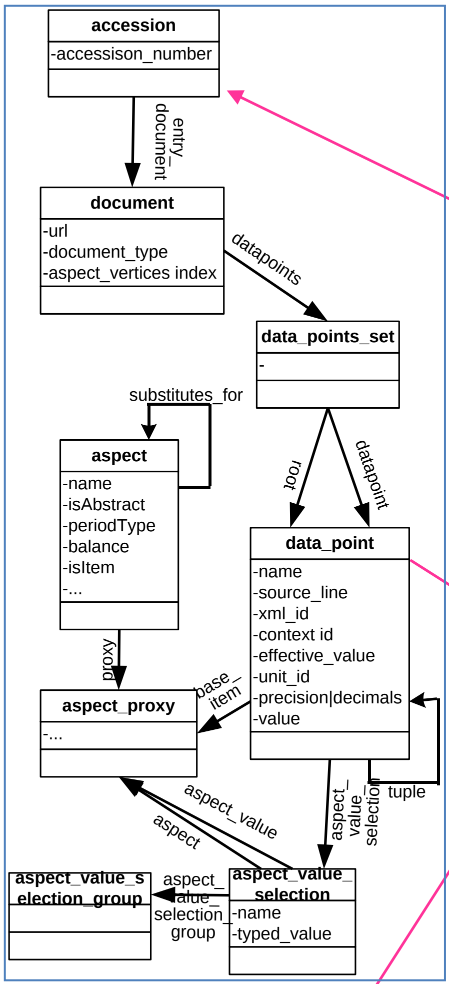
```
['rows':g.V('class','accessions').out
 .as('data').as('id').select{it.id}
 {[it.filing_accession_number,
   it.accepted_timestamp,
   it.entry_url,
   it.creation_software?:'']}]
```

```
[{"rows":
   [{"id":"132005",
     "data":["1365399696","2013-04-07T22:41:36",
        "http:\/\/www.sec.gov\/Archives\/edgar\/data…/penx-20130228.xml",
        " RR Donnelley Xcelerate Instance Document …  6.20.1 },
    {"id":"88311",
     "data":["1365399631","2013-04-07T22:40:31",
       "http:\/\/www.sec.gov\/Archives\/edgar\/data\/…/neom-20121231.xml",
       "Generated using Ez-XBRL version 6.0.1.3…]},
    {"id":"11",
     "data":["1365399527","2013-04-07T22:38:47",
        "http:\/\/www.sec.gov\/Archives\/edgar\/data\/…/slpc-20121231.xml",
        "Produced by HARRISON LAW, P.A. using EDGARsuite software…
```

Accession	Timestamp	URL	Creation Software
1365399696	2013-04-07T22:41:36	http://www.sec.gov/Archives/edgar/dat	RR Donnelley Xcelerate Instance Document, based on XBRL 2.
1365399631	2013-04-07T22:40:31	http://www.sec.gov/Archives/edgar/dat	Generated using Ez-XBRL version 6.0.1.3 [03/31/2013 08:21:5
1365399527	2013-04-07T22:38:47	http://www.sec.gov/Archives/edgar/dat	Produced by HARRISON LAW, P.A. using EDGARsuite software

Listing Accession Facts

This use case populates a table with a list of accession submission facts:



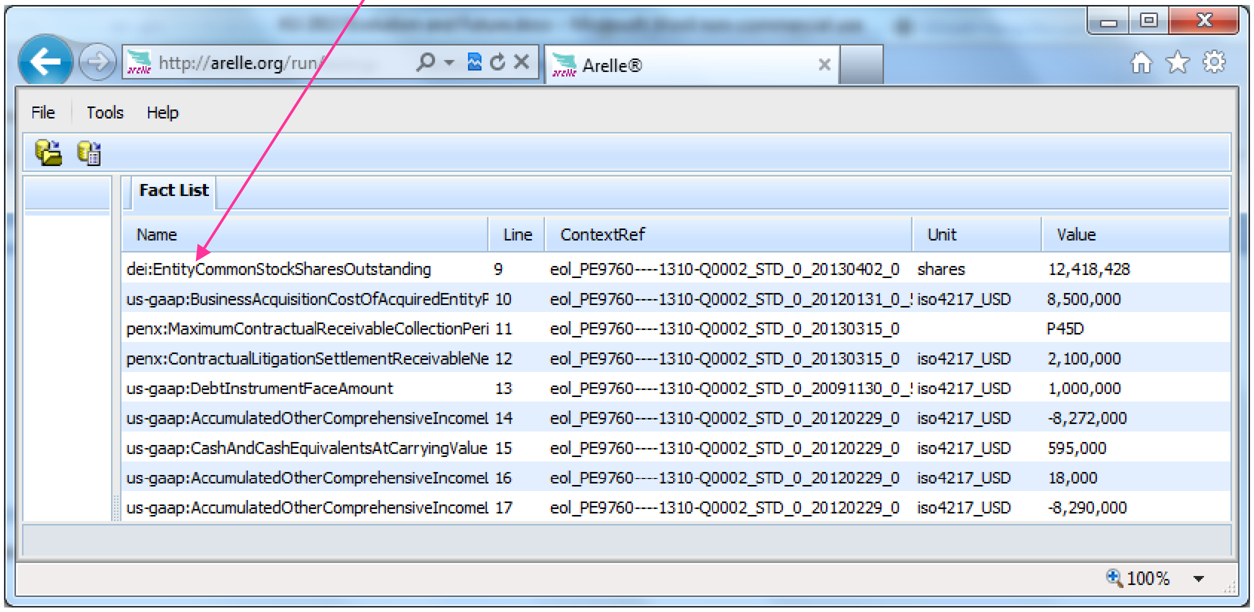
Using an accession vertex, v(132005), its entry document edge to data points set to set of data points, ordered (in this case) by source line (entry order), and with attributes of the data point put into the data for the display columns. (More sophistication, below, adds use the concept aspect's label, and could provide dimensional information instead of context ID, etc).

Gremlin graph query

```
['rows':g.v(132005).out('entry_document')
  .out('data_points').out('data_point')
  .order{it.a.source_line <=> it.b.source_line}
  .as('data').as('id').select{it.id}{
  [it.name,
   it.source_line,
   it.context?:'',
   it.unit?:'',
   it.effective_value?:it.value?:'']}]
```

Json results

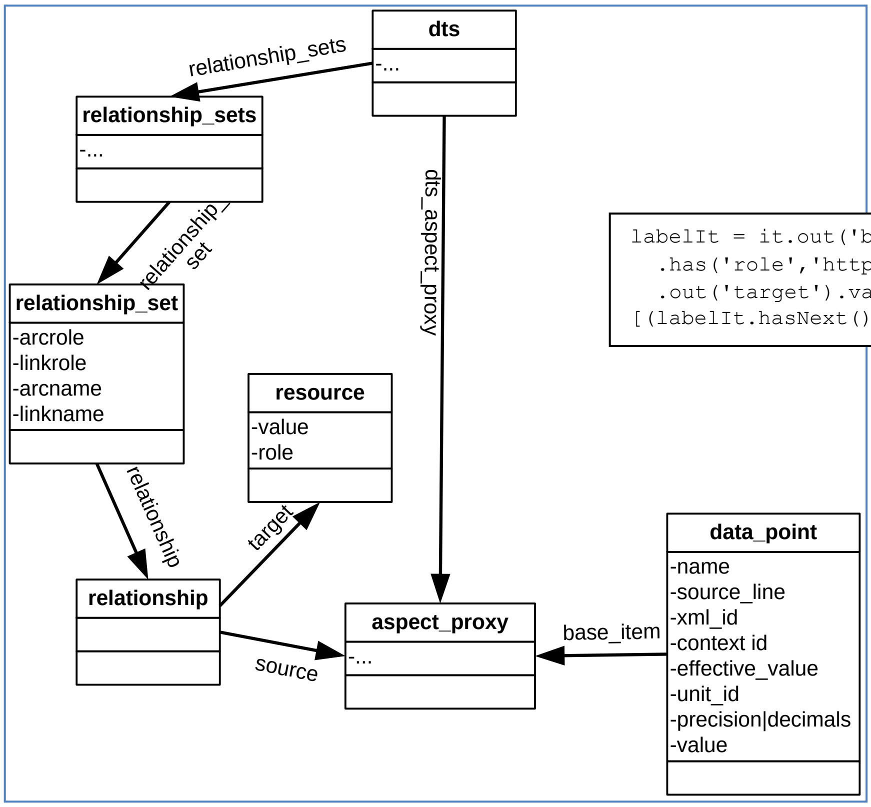
```
[{"rows":
 [{"id":"134195",
   "data":["dei:EntityCommonStockSharesOutstanding",9,
           "eol_PE9760----1310-Q0002_STD_0_20130402_0",
           "shares",
           "12,418,428"]},
  {"id":"134197",
    "data":["us-gaap:BusinessAcquisitionCostOfAcquired…",
            10,
            "eol_PE9760----1310-Q0002_STD_0_20120131…",
            "iso4217_USD",
            "8,500,000"]},
```



Name	Line	ContextRef	Unit	Value
dei:EntityCommonStockSharesOutstanding	9	eol_PE9760----1310-Q0002_STD_0_20130402_0	shares	12,418,428
us-gaap:BusinessAcquisitionCostOfAcquiredEntityF	10	eol_PE9760----1310-Q0002_STD_0_20120131_0_!	iso4217_USD	8,500,000
penx:MaximumContractualReceivableCollectionPeri	11	eol_PE9760----1310-Q0002_STD_0_20130315_0		P45D
penx:ContractualLitigationSettlementReceivableNe	12	eol_PE9760----1310-Q0002_STD_0_20130315_0	iso4217_USD	2,100,000
us-gaap:DebtInstrumentFaceAmount	13	eol_PE9760----1310-Q0002_STD_0_20091130_0_!	iso4217_USD	1,000,000
us-gaap:AccumulatedOtherComprehensiveIncomel	14	eol_PE9760----1310-Q0002_STD_0_20120229_0	iso4217_USD	-8,272,000
us-gaap:CashAndCashEquivalentsAtCarryingValue	15	eol_PE9760----1310-Q0002_STD_0_20120229_0	iso4217_USD	595,000
us-gaap:AccumulatedOtherComprehensiveIncomel	16	eol_PE9760----1310-Q0002_STD_0_20120229_0	iso4217_USD	18,000
us-gaap:AccumulatedOtherComprehensiveIncomel	17	eol_PE9760----1310-Q0002_STD_0_20120229_0	iso4217_USD	-8,290,000

Linkbase information (e.g., label)

This use case adds a dts-discovered linkbase label to each fact (shown before with just its QName):

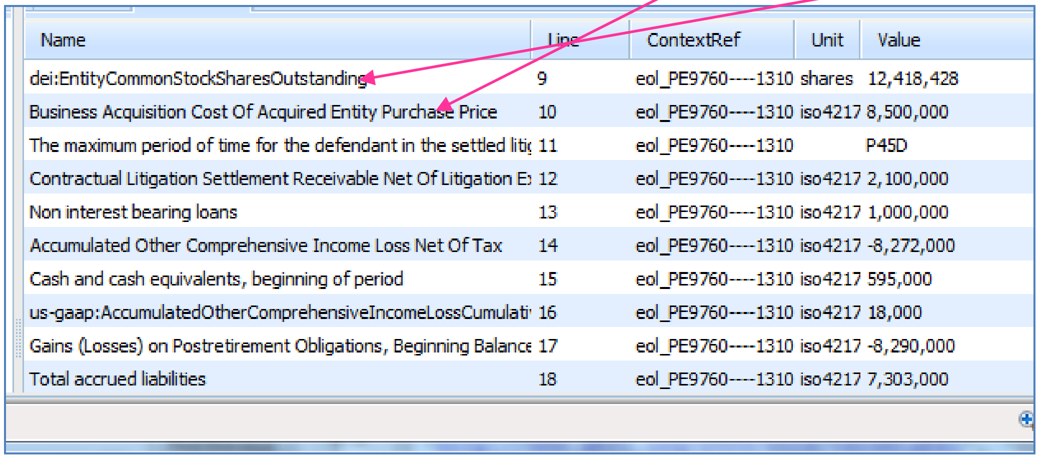


The portion of the above query that simply accessed the name property of the data_point now navigates the graph as follows:

Gremlin graph query

```
labelIt = it.out('base_item').in('source')
  .has('role','http://www.xbrl.org/2003/role/label')
  .out('target').value
[(labelIt.hasNext() ? labelIt.next() : it.name)
```

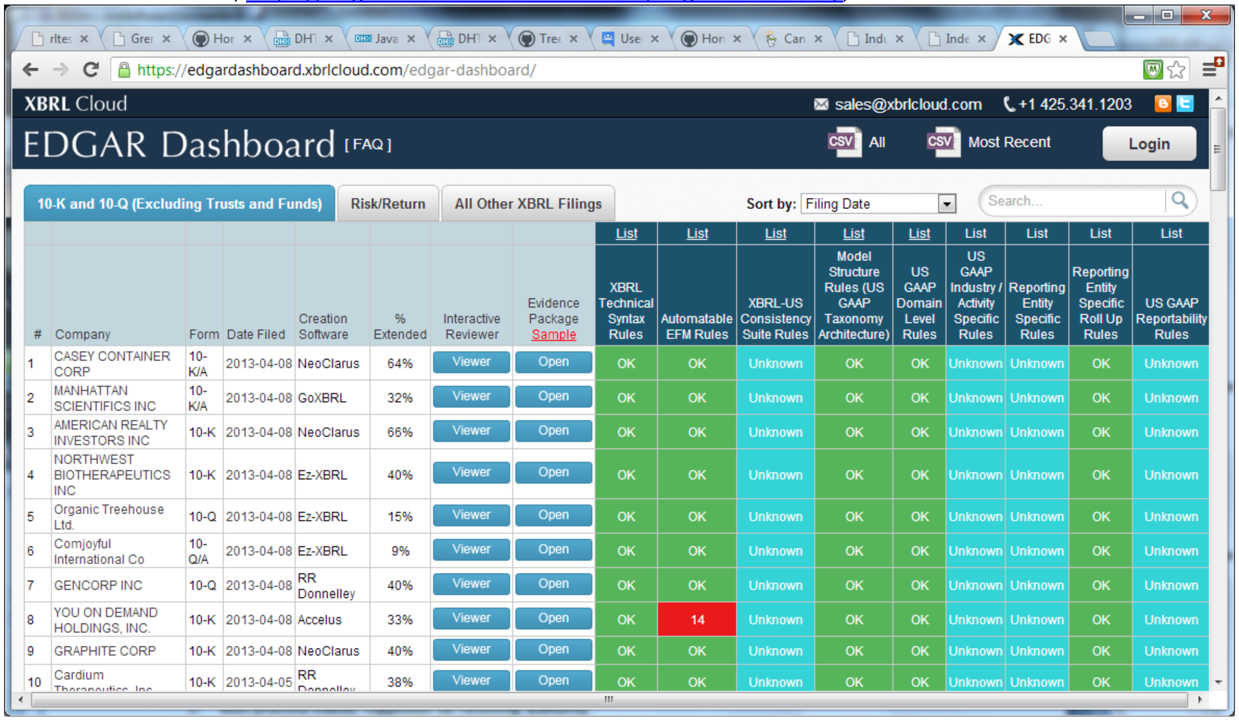
From data point, the base_item edge connects to an aspect_proxy (for the aspect, or XBRL concept of the data point), which is the source of a concept-label relationship to a label resource. To achieve this, from the data_point (fact) node, we form an iterator (for all 'standard' labels, there may be multiple). It represents the data_point, .out('base_item') navigates the edge labeled 'base_item', to the aspect_proxy, where .in('source') navigates an in-coming edge labeled 'source' to a relationship node, then .out('target') navigates the out-going edge from relationship to target resources, where the .has() filters to just those labels having standard label role, and .value represents the label's text contents. For the case where there are no labels, the iterator.hasNext() will be false, and the fact's name (qname) is taken instead.



Name	Line	ContextRef	Unit	Value
dei:EntityCommonStockSharesOutstanding	9	eol_PE9760----1310	shares	12,418,428
Business Acquisition Cost Of Acquired Entity Purchase Price	10	eol_PE9760----1310	iso4217	8,500,000
The maximum period of time for the defendant in the settled litiç	11	eol_PE9760----1310		P45D
Contractual Litigation Settlement Receivable Net Of Litigation E:	12	eol_PE9760----1310	iso4217	2,100,000
Non interest bearing loans	13	eol_PE9760----1310	iso4217	1,000,000
Accumulated Other Comprehensive Income Loss Net Of Tax	14	eol_PE9760----1310	iso4217	-8,272,000
Cash and cash equivalents, beginning of period	15	eol_PE9760----1310	iso4217	595,000
us-gaap:AccumulatedOtherComprehensiveIncomeLossCumulati	16	eol_PE9760----1310	iso4217	18,000
Gains (Losses) on Postretirement Obligations, Beginning Balance	17	eol_PE9760----1310	iso4217	-8,290,000
Total accrued liabilities	18	eol_PE9760----1310	iso4217	7,303,000

Validation technologies, Formula, Sphinx and API, and graph databases

Validation has been a completely disjoint subject from the XBRL Abstract Model, which this author feels can now change and become an integrated discussion. Perhaps the issue is that validation was perceived as an independent workflow task from consumption of accepted data submissions, which might pertain to situations where submitted data is not publically available (such as tax returns and credit risk information). However public financial reports are public, and today SEC filings are publically analyzed for filing quality (or berated in public) and expected to serve investors and shareholders consuming its contents. An example commercial service providing public examinations of filings is the XBRL Cloud Dashboard, https://edgardashboard.xbrlcloud.com/edgar-dashboard/, shown here:



								List	List	List	List	List	List	List	List	List
#	Company	Form	Date Filed	Creation Software	% Extended	Interactive Reviewer	Evidence Package Sample	XBRL Technical Syntax Rules	Automatable EFM Rules	XBRL-US Consistency Suite Rules	Model Structure Rules (US GAAP Taxonomy Architecture)	US GAAP Domain Level Rules	US GAAP Industry / Activity Specific Rules	Reporting Entity Specific Rules	Reporting Entity Specific Roll Up Rules	US GAAP Reportability Rules
1	CASEY CONTAINER CORP	10-K/A	2013-04-08	NeoClarus	64%	Viewer	Open	OK	OK	Unknown	OK	OK	Unknown	Unknown	OK	Unknown
2	MANHATTAN SCIENTIFICS INC	10-K/A	2013-04-08	GoXBRL	32%	Viewer	Open	OK	OK	Unknown	OK	OK	Unknown	Unknown	OK	Unknown
3	AMERICAN REALTY INVESTORS INC	10-K	2013-04-08	NeoClarus	66%	Viewer	Open	OK	OK	Unknown	OK	OK	Unknown	Unknown	OK	Unknown
4	NORTHWEST BIOTHERAPEUTICS INC	10-K	2013-04-08	Ez-XBRL	40%	Viewer	Open	OK	OK	Unknown	OK	OK	Unknown	Unknown	OK	Unknown
5	Organic Treehouse Ltd.	10-Q	2013-04-08	Ez-XBRL	15%	Viewer	Open	OK	OK	Unknown	OK	OK	Unknown	Unknown	OK	Unknown
6	Comjoyful International Co	10-Q/A	2013-04-08	Ez-XBRL	9%	Viewer	Open	OK	OK	Unknown	OK	OK	Unknown	Unknown	OK	Unknown
7	GENCORP INC	10-Q	2013-04-08	RR Donnelley	40%	Viewer	Open	OK	OK	Unknown	OK	OK	Unknown	Unknown	OK	Unknown
8	YOU ON DEMAND HOLDINGS, INC.	10-K	2013-04-08	Accelus	33%	Viewer	Open	OK	14	Unknown	OK	OK	Unknown	Unknown	OK	Unknown
9	GRAPHITE CORP	10-K	2013-04-08	NeoClarus	40%	Viewer	Open	OK	OK	Unknown	OK	OK	Unknown	Unknown	OK	Unknown
10	Cardium	10-K	2013-04-05	RR	38%	Viewer	Open	OK	OK	Unknown	OK	OK	Unknown	Unknown	OK	Unknown

This commercial site provides subscription services to check filings before submission and to examine their rule validations after submission. The "Automatable EFM Rules" are proprietary checks in addition those required by the SEC for formal submission.

A simple classification of validations is:

- File validation, character encoding, BOM acceptability, compressed archive content accessibility
- XML document construction, syntax correctness, schema references and DTS composition of XML documents (inter-document references, availability of referenced files, acceptability of referenced files and acceptability of any active content (HTML or scripting) (in terms of their version and suitability for receiving authority).
- XBRL 2.1 base specification validation, including side effects of DTS composition, role and arcrole definition, type, particle, and concept construction, relationship resolution and constraints.
- XBRL 2.1 calculation linkbase and essence-alias validation, where specified
- XBRL Dimensions validation, where specified
- Receiving authority validation (such as U.S. SEC Edgar validation)
    - Syntactical and compositional checks
    - Semantic checks required for acceptance
    - Consistency checks required for post-acceptance audit and verification
    - Best practice checks suggested by receiving authority
- Industry practices validation
    - Best practices checks (such as XBRL International and XBRL-US Best Practice Guidelines)
    - Commercial validation service practices guidelines (such as XBRL Cloud)
- Data consumer validation

    - Ratio, trend, and cross-filings validations (such as by auditors, investors, credit risk advisors)

Typical implementation of such validations requires progressive installation of layers of technology:

- Workflow and receiving authority document management (are files or archives intact, acceptable encoding, etc)
- XML level parsing, usually embedded in an XBRL processor
- XBRL processor to resolve DTS composition, role and arcrole definitions, relationship resolution, element and particle composition (tuples, typed dimensions, and resources)
- Calculation and dimensions are usually embedded as part of an XBRL processor as they require complete access to the DTS object model, and at least partial (or streaming) access to an XBRL instance.
- Remaining validations can utilize alternative technologies:
    - XML syntax rules (such as Schematron [8])
        - Composition of documents, presence of instance elements and content
    - XBRL-aware rules (XBRL Formula, Sphinx)
        - Instance aspects validation
        - Assertions of completeness or fallback for missing instance values
        - Relationship tree based rules
    - Processor API implementation may be required for
        - Validation of DTS, schema, or relationships (absent of any use by facts)
        - Cross-checking relationships (missing calculation relationships, disallowed dimensional constructions)

Regarding the technologies for those validations not performed by XML and XBRL processors (the last major bullet), at the onset of XBRL, it was felt important to have something like Excel's formulas for the business users, and XML's query facilities, for the techies. The author participated in the initial U.S. FDIC XBRL project [9], where formulas were based on Excel syntax. This provided a familiar approach to writing expressions that was sufficient at the time (before the advent of XBRL dimensions). The approach was not deemed sufficient by the Formula Working Group after dimensions.

XBRL Formula next developed over a six years to provide a XML-centric declarative way of specifying the semantic constraints of a taxonomy, and has come into significant use in the European and Asian ecosystems. XBRL formula [10] is based on an extensible linkbase syntax, expressing declaratively assertions and derived output instance facts, based on an aspect model of XBRL instance facts. Formula has a robust predicate logic approach, and includes the notion of fallback in the absence of facts (so that assertion and output instance fact expressions can denote and respond to missing input facts).

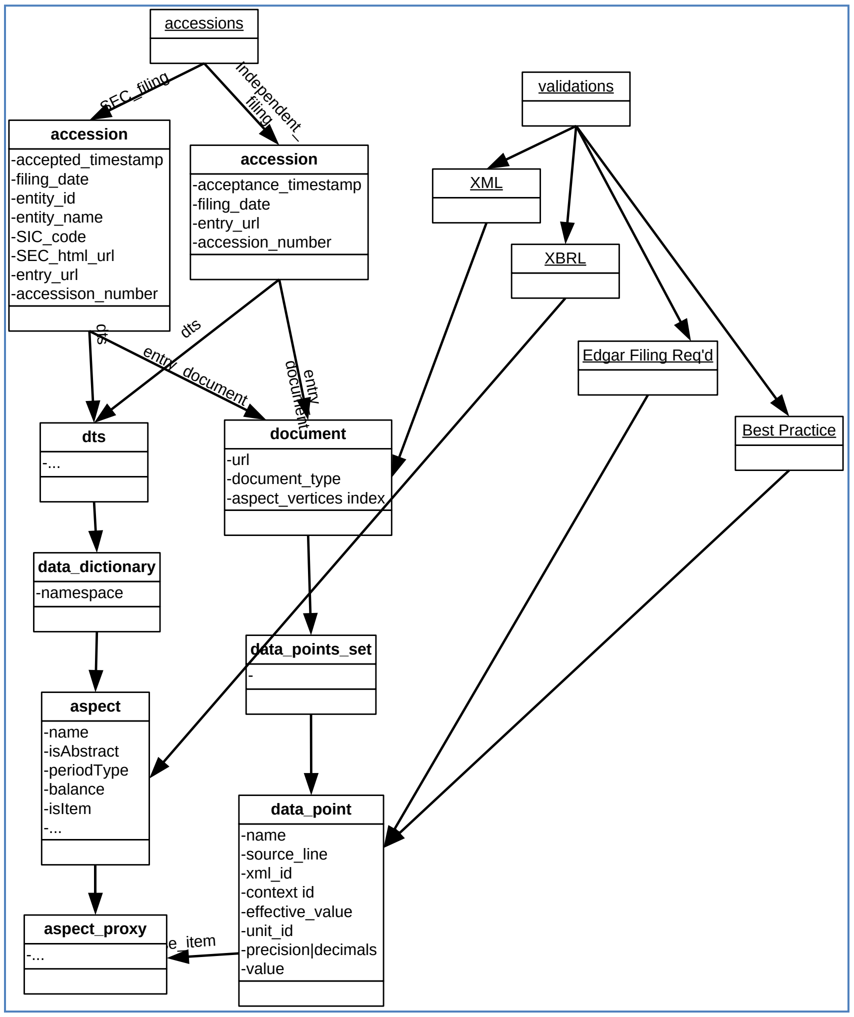
In the US, the more business-rules focused approach of the Sphinx language [11] is fervently supported. Sphinx is a syntax-only semi-procedural expression of rules, somewhat Pythonic in nature. Compared to the syntactical contrivances needed in a Formula Linkbase, it is a substantially compact expression.

The author's contributions to the open source community find a third angle important, raw speed when supporting production environments, experienced in a API based set of validation implementations. A significant set of validations required by the U.S. SEC and the XBRL-US Best Practices working groups involve analysis of the relationships graphs modeled in taxonomies of filings, and make no sense for XBRL Formula (as they do not relate to instance facts, but instead to relationship graph structures). API that is robust in set-based operations makes these validations feasible. In the case of such implemented validations, they are based on an XBRL processor's object model, which is distilled from the XBRL Semantics after processing. The Arelle project has implemented plug in validations in this manner.

With the prototyping of an instance of the Abstract Model graph database, a different paradigm of validation becomes interesting in two stages, that of (1) integrating validation results into the graph database, and (2) eventual validation upon the graph models directly. Present day validation technologies consider a single instance or filing with its extension taxonomies by itself. They validate and produce textual or structured messages from XML, XBRL, formula, and rules processors. These results may be consumed by workflow products of producers, or in some cases simply handled as text (for example, the SEC Edgar receipt processing simply e-mails text-based result messages). Some present-day dashboard systems also capture results as text messages (organized by single filings).

With consideration of the abstract model-based graph database, and a graph manipulation language robust in graph structure manipulation, one can consider integrating validation results so they can be queried across large collections of filings. This case may not care about ordinary simple validations, such as file errors that would block submission of filings, but instead more advanced semantical checks (such as for non-negative items or mismatches between presentation format and calculation linkbases). Such validations are deeply-connected to the relationship models in a filing taxonomy, and the graph database allows a fine-grain linkbase. Validation issues linked to underlying model and fact data would allow easy instant review of single filings, and also cross-filing parallel reviews (such as which other filings of similar companies had same issues in similar parts of their financial report).

A research effort now under way to integrate the graph database and validation technologies is to provide a clear way to drill down to examine the nature of difficult validations. While all of the present validation technologies provide their outputs in some human readable form (reports, web pages) backed up perhaps by XML files, they don't support public query and analysis.



Here elements of the prior instance model are integrated with skeletal elements representing validation objects. Validation will follow some hierarchy, eventually expanded to represent those error conditions and rules validated. The lower level breakdown of validations can be edge connected to the graph model objects of the validation result. For example rules that identify data points may also identify aspects (concepts or dimensions) and would be edge connected to both.

A dashboard or error list in this environment could not only connect error conditions with the model objects involved, but could be navigated in a different manner to connect multiple instances, for example to query multiple filings involving the same error criteria or code, and to categorize them (such as by industry code or business similarity to the compared filing).

A logical progression from rule-based validation that relates to submission construction (such as Edgar Filing rules) is to extend it to the content of data itself. A dauntingly difficult task has been to assess financial ratios across large collections of filings. Multiple communities would benefit from easy and fast ways to compare such information, but it is difficult to access without model-based query access to the fact's taxonomy structure. Debreceney, et al. [12] discussed this at the prior KU conference. His paper notes the improvement in ability to determine how data is classified with taxonomy relationship model information at hand. This paper explores a start at such technology. The author is enticed at the potential of matching the technologies of graph storage for XBRL in the large, for comparing of graph information of models across many filings, and the start of technology to obtain financial ratio based on model-described data.

References

[1] XBRL-US 2011. XBRL US Public Database. Available from: xbrlchallenge@xbrl.us

[2] Codd, E.F., 1979. Extending the Database Relational Model to Capture More Meaning, published by IBM Research Laboratory and also by ACM Transactions on Database Systems, Vol. 4, No. 4, December 1979, pages 397-434.

[3] Frankel, D., Fischer, H., Foster, W., and Lam, R. XBRL Abstract Model 2.0, June 2012. Available at http://xbrl.org/Specification/abstractmodel-primary/PWD-2012-06-06/abstractmodel-primary-pwd-2012-06-06.html.

[4] Titan Distributed Graph Database. Available from http://thinkaurelius.github.com/titan/.

[5] Gremlin Query Language. See https://github.com/thinkaurelius/titan/wiki/Gremlin-Query-Language for introduction and further references.

[6] Fischer, H., and Mueller, D. 2011. Open Source & XBRL: the Arelle® Project, 2011 Kansas University XBRL Conference, April 29-30 2011, Overland Park, Kansas. Avalable from: http://web.ku.edu/~eycarat/myssi/_pdf/4-Fischer%20-Mueller-Open-source-ArelleProject.pdf

[7] Fischer, H. and Muller, D., 2011. Enabling Comparability and Data Mining with the Arelle® Open Source Unified Model, First Conference on Financial Reporting in the 21st Century: Standards, Technology, and Tools, September 8-9, 2011, Macerata, Italy. Available from: http://arelle.org/downloads/11.

[8] Schematron. ISO/IEC 19757-3:2006 Information technology -- Document Schema Definition Language (DSDL) -- Part 3: Rule-based validation – Schematron. See: http://www.schematron.com.

[9] Federal Deposit Insurance Corporation (FDIC) (undated). Implementing XBRL Formulas. Available at: http://www.fdic.gov/bank/ImplementingXBRLFormulas.pdf.

[10] Fischer, H. Formula Overview 1.0, et al. XBRL International, December 2011. Available at http://xbrl.org/WGN/XBRL-formula-overview/PWD-2011-12-21/XBRL-formula-overview-WGN-PWD-2011-12-21.html.

[11] CoreFiling, 2012. Sphinx 2 Primer. Available from: Corefiling, Oxford UK, http://blogs.corefiling.com/2011/06/sphinx-an-xbrl-expression-language/.

[12] Debreceney, R., Alessandro, d'E., Felden, C., Farewell, S., and Piechocki M. 2011. Feeding the Information Value Chain: Deriving Analytical Ratios from XBRL filings to the SEC, 2011 Kansas University XBRL Conference, April 29-30 2011, Overland Park, Kansas. Avalable at http://web.ku.edu/~eycarat/myssi/_pdf/2-Debreceny-XBRL%20Ratios%2020101213.pdf.

About the author:

Herm Fischer has contributed to the XBRL base specification, and its dimensions, formula, versioning, and rendering modules. He currently is a member of the XBRL Standards Board, chairs the Formula and Rendering Working Groups and is vice chair of the Base Specification and Maintenance Working Group. He contributed to the 2007 XBRL US GAAP Project. He currently contributes to and coordinates the Arelle open source XBRL processor project. Formerly with UBmatrix, Inc, he developed XBRL tools. He participated in development of SEC validation and its test suite.