DSE 250 Final – Fall 2016

December 6, 2016

For this final, you will fill in a "cheat sheet" of fundamental questions you need to consider when evaluating a novel (data model, query language) pair.

1 Models

1.1 What are the entities and the properties?

At the very least, we expect every model to have a notion of entities/objects, with properties/attributes associated to them. The first thing to ask about a model is what notion coresponds to such entities, and what to their properties.

1.2 What notions of collections of entities exist?

A model would be useless if it couldn't describe a collection of entities.

Questions: what kinds of collections are supported? How do they relate to the standard programming language notions of record, set, multiset, list, array, map?

1.3 Is the model complex-valued?

The model is complex-valued if the properties of an entity can in turn be collections of entities.

1.3.1 Is the nesting of collections bounded?

Can we have collections within collections etc. up to arbitrary nesting depth, or is this depth bounded?

1.3.2 What notions of equality are defined?

For atomic values (values of base type such as numeric, string, date, etc.) the notions of equality will likely be the standard ones adopted from classical programming languages.

However, for complex values it makes sense to ask how they are compared as this is less standard.

Possibilities are *deep* equality, which checks that two entities are equal by recursively checking that all their properties are pairwise equal. Related: what is "pairwise"? What are possible interpretations thereof?)

Alternatively, *shallow* equality checks only that the non-collection properties of the two entities are pairwise equal.

1.4 Does there exist a notion of identity?

One thing to ask early on is whether entities have an identity, i.e. a designated property (subset of properties) whose value is unique across all entities.

1.5 If the model is complex-valued and identity-aware, are there corresponding equality notions?

Can one compare entities by deep-value equality? By identity?

1.6 Is there a means to connect/link/relate entities?

Can entity sets be related according to many-to-one, many-many, one-one relationships?

1.7 Is there a notion of type/kind/label associated to entities/properties?

Start from your intuition gained from programming languages and see whether the standard notion of type has a counterpart in the model, or how it differs.

1.8 Are all entities necessarily typed, or are types optional?

1.8.1 Can the same entity have more than one type?

Or are types mutually exclusive?

1.8.2 Is there a subtyping relationship on types?

A type t_1 is a subtype of t_2 if all entities of type t_1 are implicitly also entities of subtype t_2.

1.8.3 Can a type be a subtype of several types that are not in direct/indirect subtyping relationships themselves?

1.9 How does the expressivity of this model compare to that of other reference ones?

Model m is at least as expressive as model m' if each value v' in m' can be translated to a value v in m such that a faithful inverse translation from v to v'

is possible.

2 Query Languages

2.1 Is the QL procedural or declarative?

Procedural means that it explicitly states the operations (and their order) required to find the data. Declarative QLs only specify what conditions the data need to satisfy to be returned by the query, without any hint on how to find this data.

2.2 What kind of navigation is supported?

2.2.1 How can one access the properties of an entity?

Think of the analogous record projection in standard programming languages. Call this access a navigation step. In particular, it is a projection step.

2.2.2 If relationships between entities are supported, can one access an entity from another entity by following a relationship between them?

Is the crossing of a relationship link between two entities possible? Classify this crossing also as navigation step, in particular a traversal step.

2.2.3 Are relationships directed or undirected?

That is, can relationships be crossed in either direction, or are they assymmetric?

2.2.4 What corresponds to the notion of entry point into the data?

How do we specify the entities from which we wish to start navigating the data?

2.2.5 Is there a notion of path expression?

A path expression describes a (possibly singleton) sequence of navigation steps. A path expression evaluated at entity e returns the collection of entities reachable from e via the sequence of steps described by the expression.

2.2.6 Is it legal for the result of a path expression starting at a source entity to be a collection of target entities?

2.2.7 Is it legal for the result of a path expression to be the empty collection?

2.2.8 If the results of path expressions are sets of entities, how are duplicates removed?

2.2.9 How are duplicates defined? by value- or by identity-eqality?

2.2.10 Are path expressions structure-preserving or flattening?

Suppose that the result of evaluating path expression p_1 starting from entity e is a collection C of entities. Suppose that for the entities in C, the result of evaluating path expression p_2 is in turn a (possibly non-empty) collection of entities. Then what does the path p obtained by concatenating p_1 with p_2 return when evaluated at e? A collection of collections? Or is this result a single collection, obtained by merging the nested collections? In the former case, we say that path expressions are structure-preserving, in the latter we say that they are flattening.

2.2.11 Can path expressions traverse collections?

If a navigation step s starting from entity e returns a collection, are path expressions starting with s and continuing with at least one more step legal at e?

2.2.12 Can path expressions specify traversal along unboundedly many navigation steps?

As opposed to bounded by the size of the expression?

2.3 Does the language have variables bound to the results of path expressions?

2.3.1 Can path expressions start from previously defined variables, or do they have to start from entry points only?

2.3.2 Can variables be compared to each other and to constants?

For equality comparisons, what kind of equality is supported? Value-based or identity-based?

2.3.3 Can one give a pattern-match/construct semantics to queries?

Can one conceptually think of queries as performing a two-phase computation as follows?

The query is specified by a pattern, possibly containing variables, and a return expression.

In the first phase, the pattern is matched in all legal ways against the data. Each match yields a tuple of values providing the binding for the tuple of pattern variables. Call *match table* the collection of the tuples yielded by the match phase.

In the second phase the return expression is executed for each tuple of the match table. If the return expression depends on the variables, the match tuple provides the corresponding variable binding as input to the return expression.

2.4 Is the language compositional?

Does the result of queries belong to the same data model as the query inputs?

2.5 If the language is compositional, and the model is complex-valued, can the query construct complex-values?

What constructors are there for building the output complex values from input complex values?

2.5.1 What are the semantics of executing a call to a constructor taking a variable as argument? Value- or reference-based?

Suppose a constructor is called with a variable x as argument, with the intended meaning that a new entity is built whose contents is given by the bindings of x. What exactly does "given by" mean here? Is x output by value, i.e. does the constructor use a copy of the value x binds to (deep copy if the value is complex)? Or does the constructor output x by reference, i.e. outputs the identity of x's binding (of course this only makes sense if the data model has a notion of identity).

2.6 Can constructor calls take as argument collections of entities returned by queries?

We call these queries "nested" within the "outer query" that performs the constructor call.

2.6.1 Is the nesting depth of queries bounded?

3 The Survey

Answer all the above questions for the following (model,language) pairs studied during this course:

model	language
relational	SQL
OO	OQL
XML	XPath
XML	XQuery
neo4j graphs	Cypher
JSON	N1QL (Couchbase's QL)

Refer to the questions via the section numbers whose titles they appear in.

Feel free to answer N/A or 'it depends' as needed. In the latter case, give a very brief justification.