Report September 24th

Improving the formalization of the algorithm and Building our architecture

The basic input for the Rhone algorithm is: (1) a query; and (2) a list of concrete services. Query and concrete service are formally defined as follows.

Definition 1 (Query): A query Q has the form:

$$Q(\overline{I}, \overline{O}) := A_1(\overline{I}, \overline{O}), A_2(\overline{I}, \overline{O}), .., A_n(\overline{I}, \overline{O}), C_1, C_2, .., C_m[P_1, P_2, .., P_k]$$

The left side of the definition is the head of the query; and the right side is the body. $\overline{I}$ and $\overline{O}$ are a set of *input* and *output* parameters, respectively. A query is expressed as set of abstract services (A) which specify abstract functions performed by the query. Constraints over the *input* or *output* parameters are defined in C. The user preferences over the services are signed in P. C and P are in the form $x \otimes constant$ such that $\otimes \in \{\geq, \leq, =, \neq, <, >\}$.

Definition 2 (Concrete service): A concrete service (S) has its form similar to a query:

$$S(\overline{I}, \overline{O}) := A_1(\overline{I}, \overline{O}), A_2(\overline{I}, \overline{O}), .., A_n(\overline{I}, \overline{O})[P_1, P_2, .., P_k]$$

A concrete service (S) is defined as a set of abstract services (A), and by its quality constraints P. These quality constraints associated to the service represent the service level agreement exported by the concrete service.

Given the query and the concrete services, the algorithm tries to match abstract services in S with the same abstract service in Q.

Definition 3 (Abstract service equivalence): Given an abstract service A_i such that $A_i \in Q$ and $A_i \in S$. $A_i.Q$ is equivalent to $A_i.S$, denoted $A_i.Q = A_i.S$, iff (1) $A_i.Q$ and $A_i.S$ have the same abstract function name; (2) the number of *input* and *output* variables of $A_i.Q$ is equal to the same of $A_i.S$.

Once a matching between abstract services is found, the algorithm checks if the concrete service associated in the matching is a candidate to be part of the rewriting process.

Definition 4 (Candidate service): Given a query Q and a concrete service S, S is a candidate service iff: (1) $\nexists A_i\ s.t.\ A_i \in S\ and\ A_i \notin Q$; and (2) the quality constraints in S does not violate the user preferences in Q.

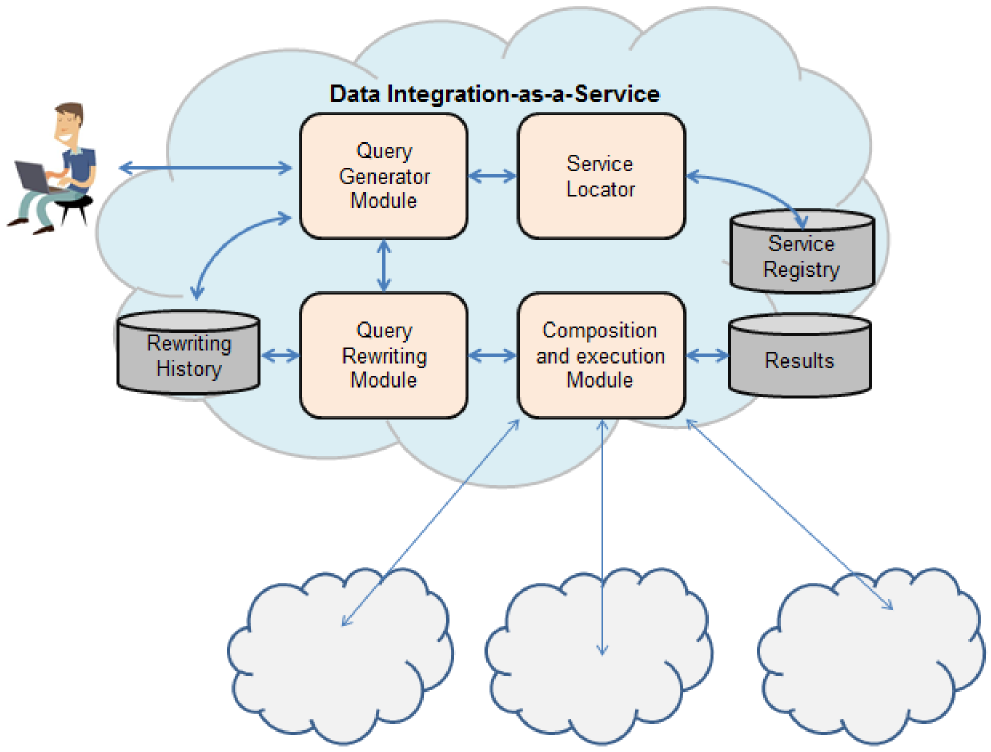
In the next step, the algorithm builds the *partial descriptors* (PD). Each descriptor describes how a *candidate* concrete service can be used in the query rewriting process.

Definition 5 (Partial Descriptor): PD is a complex data structure defined as follows: $\langle S, h, \varphi, G, P\rangle$ where S is a concrete service. h are mappings between terms in the head of S to terms in the body of S. φ are mapping between terms from the abstract composition to terms in the concrete service definition. G is a set of abstract services covered by S. P is a set quality constraints associated to the service S.

Intuitively, a rewriting is a set of *partial descriptors* that fully covers the original query, and do not violates the user preferences.

Architecture

Query Generator Module helps the user to create his queries, and to define his preferences. Once we have a query this module will try to find if there are previous rewritings to a equivalent query. If true, *the composition and execution module* can publish and execute the new composition retrieving and integrating the new data. If there are no previous rewritings, the module interacts with the *service locator* in order to identify in our *service registry* services that can answer the query or part of it. The query and the selected services are used in the *Query Rewriting Module* to generate the rewritings for the query. The rewritings are sent to *the composition and execution module* to be published and executed (perhaps we can use BPEL to compose and execute). The integration process could be done in our database or in the database of one related services (somehow we should analyze which one is the best option).


Data Integration-as-a-Service
Query Generator Module
Service Locator
Service Registry
Rewriting History
Query Rewriting Module
Composition and execution Module
Results