Relational Association Programming

(Proposed) Paradigm Documentation

3rd Revision

Table Of Contents

1 Introduction 2 Composition 2.1 Modularization 2.2 Qualification 2.2.1 Data Qualification 2.2.2 Code Qualification 2.3 Quantification 2.3.1 Data Quantification 2.3.2 Code Quantification	Page 3
3 Definition 3.1 Modularization 3.1.1 Association 3.1.2 Relation 3.2 Qualification 3.2.1 Conformation 3.2.2 Situation 3.2.3 Classification 3.2.4 Abstraction	Page 4
3.3 Quantification 3.3.1 Information 3.3.2 Location 3.3.3 Objectification 3.3.4 Function 4 Application 4.1 Programming Model	Page 5
4.1.1 Illustrated Example	Page 6
4.2 Data Access 4.3 Code Call 5 Bibliography	Page 7

1. Introduction

Relational association programming is a proposed software development paradigm based on the procedural and modular programming paradigms. The paradigm allows the classification of abstract function reference groups which promotes effective function call management in large scale systems without incurring extra run time overhead. This document identifies and defines the compositional elements then describes their programmatic applications.

2. Composition

2.1 Modularization

Association	An association is a program module.
Relation	A relation is a qualification or quantification of another association.

2.2 Qualification

2.2.1 Data Qualification

Conformation	A conformation is a data type.
Situation	A situation is a reference type.
Classification	A classification is a reference table type.

2.2.2 Code Qualification

Abstraction	An abstraction is a type of procedure.

2.3 Quantification

2.3.1 Data Quantification

Information	An information is a data instance.
Location	A location is a reference instance.
Objectification	An objectification is a reference table instance.

2.3.2 Code Quantification

Function	A function is an instance of a procedure.

3 Definition

3.1 Modularization

Modularity facilitates the design of complex systems by allowing compartmentalization. It is very strongly recommended that implementations be capable of modularity, though not entirely necessary since single module programs remain within the scope of the paradigm.

3.1.1 Association

An association is a program module containing or capable of relating qualification or quantification elements. It is not required to be an identified element in the programming language.

3.1.2 Relation

A relation exists when an association contains or has access to a qualification or quantification of another association. It is not required to be identified element in the programming language.

3.2 Qualification

Qualification is a data or code type definition. In typeless programming languages it is not necessary but some facilitation may be required for completeness.

3.2.1 Conformation

A conformation defines a primitive and composite data unit type.

3.2.2 Situation

A situation defines a reference data unit type. The implementation can syntactically distinguish between a conformation situation, a classification situation, an abstraction situation and a situation of a situation (etc.).

3.2.3 Classification

A classification defines a code reference table type. It contains function locations and may also contain conformations whose instances have a one-to-one correspondence with instances of this classification.

3.2.4 Abstraction

An abstraction defines a procedure type. It may specify the return and parameter types for procedures.

3.3 Quantification

Quantification is a data or code instance.

3.3.1 Information

An information is a data unit. The term covers both primitive and composite data.

3.3.2 Location

A location is a reference or symbolic reference. The implementation may syntactically distinguish between a information location, objectification location, function location and location of a location.

3.3.3 Objectification

An objectification is a table of code procedure references and may contain data units. Implementations which are intended to allow run-time changes should allow the copying of objectification elements onto other objectification elements. The *default* access to the objectification should be for read only in such cases.

3.3.4 Function

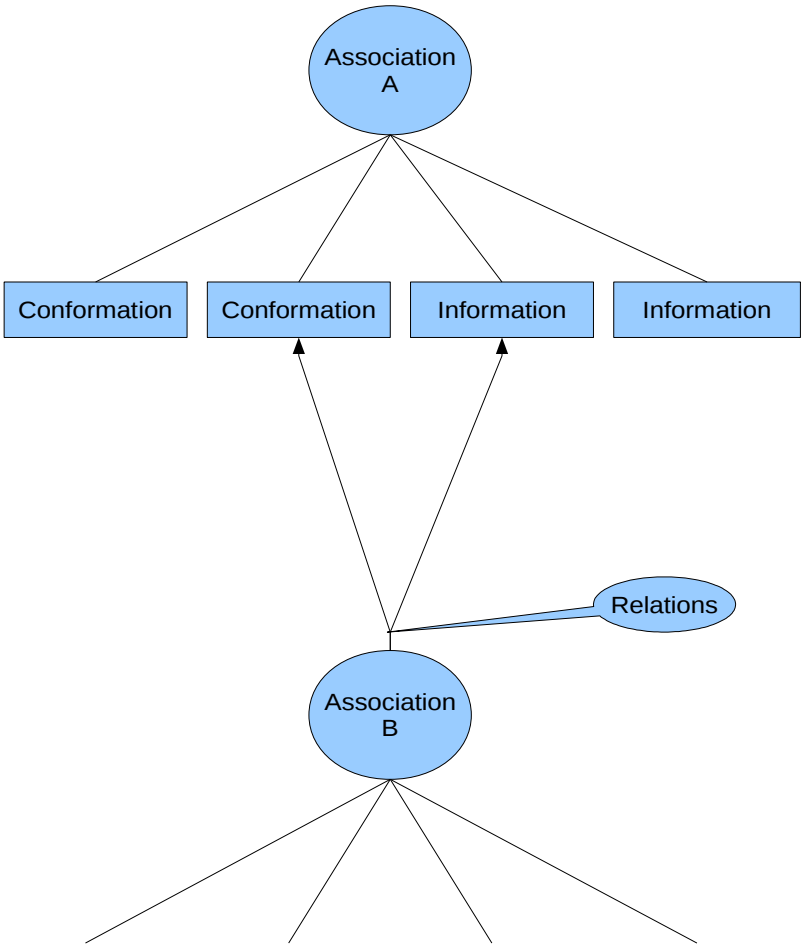
A function is simply a procedure. The ability to assign procedure references to function locations is a fundamental requirement of the paradigm.

4 Application

4.1 Programming Model

Modularization	**Qualification**	**Quantification**
Association	Relation(s)	
	Conformation(s)	Information(s)
	Situation(s)	Location(s)
	Abstraction(s)	Function(s)
	Classification(s)	Objectification(s)

4.1.1 Illustrated Example



4.2 Data Access

Association → *Relation* → ... → **Information**
Association → *Relation* → ... → **Location**
Association → *Relation* → ... → **Objectification** → *Element* → … → *Element*
Association → *Relation* → ... → **Function**

Accessing data through relational association programming begins within the context of an association as noted in the above table. If the information, location, objectification or function data being accessed is contained inside another association, access is tabled through one or more relation.

4.3 Code Call

Association → *Relation* → … → **Function**(*Parameter,* …)
Association → *Relation* → … → **Location**(*Parameter,* …)
Association → *Relation* → … → **Objectification** → *Element* → … → Element(*Parameter,* …)

Calling code through relational association programming also begins within the context of an association as noted in the above table. If the function, function location or objectification being used to call code are contained inside another association, access is tabled through one or more relations.

5 Bibliography

Data type	http://en.wikipedia.org/wiki/Data_type	February 23rd 2011
Primitive data type	http://en.wikipedia.org/wiki/Primitive_data_type	February 23rd 2011
Composite data type	http://en.wikipedia.org/wiki/Composite_data_type	February 23rd 2011
Subroutine (procedure)	http://en.wikipedia.org/wiki/Subroutine	February 23rd 2011
Procedural programming	http://en.wikipedia.org/wiki/Procedural_programming	February 23rd 2011
Modular programming	http://en.wikipedia.org/wiki/Modular_programming	February 23rd 2011