비즈니스 프로세스 질의 언어의 도입 가이드라인

A Guideline for incorporating business process query language

김민수*, 김훈태**

* 부경대학교 시스템경영공학과 (minsky@pknu.ac.kr)
** 대진대학교 산업시스템공학과 (hoontae@daejin.ac.kr)

Abstract

BPM 시스템의 활용이 확대되고 BPM 표준의 채택이 일반화되어가면서, 차츰 BPM 시스템의 관리와 유지보수 및 활용을 극대화할 수 있는 방안에 대한 요구가 높아지고 있다. 여기에는 비즈니스 프로세스 질의 언어의 표준화가 중요한 역할을 할 것으로 기대되고 있다. 비즈니스 프로세스 질의 언어는 마치 데이터베이스 관리 시스템의 SQL처럼 BPM 시스템에 대한 표준적인 접근 및 개발이 가능하게 해주며, 궁극적으로는 BPM 어플리케이션의 개발 생산성과 유지보수성을 향상시켜 줄 수 있을 것이다. 본 논문에서는 아직까지 표준화에 대한 명시적인 결과물이 제시되지 않고 있는 비즈니스 프로세스 질의 언어에 대한 기존의 연구 결과를 살펴보고, 어떠한 설계 요소들이 포함되어야 할지를 제시하고, 이를 BPM 시스템에 도입하기 위해 필요한 참조 아키텍처를 제안하고자 한다.

1. 서론

최근 몇 년간 BPM(Business Process Management)의 중요성에 대한 인식이 커지면서 BPM 시스템을 도입하는 기업들의 저변이 많이 확대되고 있는 추세이다. 이와 함께 BPM 솔루션 공급자의 개별적인 구현에 따른 상호 운용성과 호환성의 문제를 극복하기 위한 표준화 연구가 활발히 진행되었다[1].

BPM 관련 표준들은 프로세스 모형화, 실행과 운영, 모니터링과 통제 등의 다양한 프로세스 관리 영역에서 정의되었거나 정의되고 있는 상태인데, 몇몇 예를 든다면 BPMN(Business Process Modeling Notation), BPEL4WS(Business Process Execution Language for Web Services), BPML(Business Process Modeling Language), XPDL(XML Process Definition Language), BAML(Business Activity Monitoring Language) 등을 뽑을 수 있겠다[1,3,6,9].

BPM 관련한 다양한 표준들은 대부분 모델링과 실행 및 모니터링 분야에 관한 것들이 많은 부분을 차지하고 있으며, 상대적으로 많은 진척을 이루고 있지 못한 분야가 바로 비즈니스 프로세스 질의에 관련한 표준인 BPQL 분야이다.

BPQL은 BPM 시스템에 대한 표준적인 질의 및 조작언어라 할 수 있다. 오늘날 데이터베이스 관리 시스템들이 서로 다른 프로그래밍 언어와 공급자에 의해 제공되더라도, 일단 해당 시스템이 관계형 모델을 따르고만 있다면, SQL(Structured Query Language) 입력창이나 기타의 표준화된 데이터베이스 연결 도구 등을 통해 쉽게 접근하여 조작할 수 있는 것은 바로 데이터베이스의 접근을 위한 인터페이스와 사용하는 질의 언어가 표준의 형태로 잘 정의되어 있기 때문이다. BPQL은 데이터베이스 시스템의 SQL 표준처럼, BPM 시스템에 대한 접근과 조작을 표준화하는 역할을 하며, 이를 통해 서로 다른 공급자의 BPM 시스템이라도 동일한 방식으로 질의하고 관리할 수 있게 한다[1,5,7].

BPQL 표준이 적절히 정의되어 현장에 정착된다면 단순히 BPM 시스템간의 호환성과 상호 운영성의 문제만을 해결하는 것이 아닌 더 큰 파장을 미칠 것으로 예상되고 있다. 현재까지는 프로세스의 인출과 조작에 사용되는 비즈니스 로직이 개별 BPM 시스템들의 개발 도구에서 제공하는 프로그래밍 언어 속에 가려진채로 구현되고 있는 상태지만, BPQL을 통해 보다 명시적으로 정의하게 될 경우, 이러한 로직들이 표면으로 드러나게 될 것이다. 이것은 비즈니스 로직의 변화로 인한 애플리케이션의 개발과 유지 보수를 보다 효과적으로 할 수 있게 하며, 기업의 입장에서는 노출된 비즈니스 로직들을 귀중한 자산으로써 관리할 수 있게 된다. 이러한 BPQL 표준의 필요성과 기대 효과를 BPMI(Business Process Management Initiative)에서는 다음과 같이 밝히고 있다[1,4,6,7].

- BPMS에 대한 표준적인 접근 및 개발 방법론의 표준화
- 비즈니스 프로세스 애플리케이션의 개발 생산성 및 유지보수성 향상
- 조직내부의 프로세스 지식과 경험의 구조화된 관리를 가능하게 함
- 이종의 BPM 시스템간 연동 및 통합의 실제적인 도구로 활용됨
- 프로세스 마이닝을 통한 프로세스 관리 효율성의 증대

BPQL을 개발하고자 하는 노력이 몇몇 연구 단체와 상업용 WfMS(Workflow Management System) 및 BPM 시스템 공급자 등에 의해서 진행되고 있으

나 포괄적인 합의에 이르지 못한 상태이며, 표준화 단체에 의한 실질적인 성과를 보이고 있지는 않다.

본 연구에서는 기존의 대표적인 BPQL 관련 연구 결과를 살펴보고, 이들로부터 BPQL 표준이 갖추어야 할 공통 요소들을 찾아 BPQL 구현을 위한 참조 아키텍처를 제시하고자 한다. BPQL 표준 자체가 정의되어 있지 않은 현 시점에서도 여전히 BPM 시스템들은 비즈니스 프로세스에 대한 질의 처리를 나름대로 제공해야 되는 상황이다. 향후 BPQL 표준의 발표로 인해 기존의 사용 환경이 받게 될 영향을 최소화하기 위해서는 기존 연구로부터 공통점을 찾고 이로부터 도출해 낸 본 연구의 참조 아키텍처가 많은 도움이 될 것으로 기대된다.

본 논문의 구성은 다음과 같다. 이어지는 2장에서는 BPQL 관련한 대표적인 연구들을 살펴보고, 3장에서는 BPQL 설계를 위한 고려 사항을 제시한다. 4장에서는 이러한 설계 요소를 반영하기 위한 참조 아키텍처를 제안한다. 마지막으로 5장에서는 본 연구의 결론을 정리하였다.

2. BPQL 관련 기존 연구

비즈니스 프로세스를 위한 질의 언어와 관련한 대표적인 연구 활동으로는 BPMI.org의 표준화 작업과 ICONS(Intelligent Content Management System) 프로젝트에서의 BPQL 사례를 살펴보기로 하겠다. 그리고 질의 언어와 관련하여 중요하게 여겨지고 있는 SBQL(Stack Based Query Language)에 대해서도 살펴보기로 하겠다.

2.1 BPMI.org의 표준화 작업

BPMI.org는 BPM 표준화와 관련하여 가장 많이 참조되는 단체의 하나로, BPML과 BPMN 등의 비즈니스 프로세스 모델링 관련 표준을 이미 발표한 바 있다. BPMI.org에서는 BPQL 역시 표준안의 개발에 착수할 것임을 3년 전부터 공표하여 왔는데, 여기에 참여하고 있는 다수의 대형 BPM 시스템 공급업체와 관련 기관 등의 영향력을 염두에 둘 때, 향후 BPQL 표준안 제정 가능성이 가장 높은 단체로 많은 기대를 모았었다. 그러나 2005년 초에 발표하기로 한 1.0 버전의 일정을 지키지 못한 채, 또 다른 표준화 단체인 OMG(Object Management Group)와의 통합이 이루어져 사실상 BPQL 표준안 발표에 대한 전체 계획이 모호한 상태에 있다.

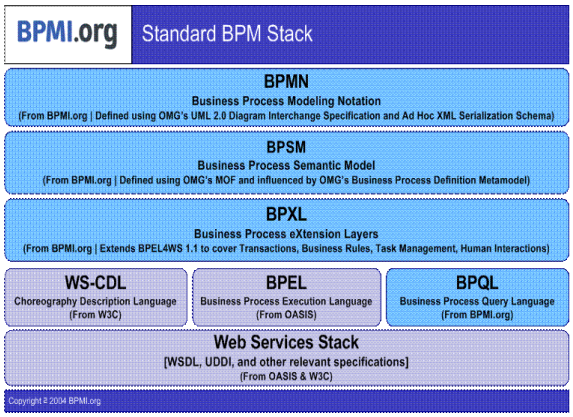
BPMI.org에서 정의하고 있는 BPQL 표준에 대한 명확한 일정은 없는 상태이지만 OMG와의 협력을 통해 BPM 분야의 표준화에 강력하게 매진할 것임을 밝힌바 있기 때문에, 조만간 BPQL의 표준안 제정에서도 주도적인 역할을 할 것으로 예상된다.

현재로서는 BPMI.org에서 BPQL과 관련하여 제시한 청사진만을 통해 어떠한 표준이 제정될 것인지를 추측할 수밖에 없겠다. BPMI.org가 밝혀온 BPQL 관련 내용들을 정리하면 다음과 같다.

BPQL은 프로세스의 실행을 맡는 프로세스 서버와 프로세스의 배치를 맡는 프로세스 저장소(Process Repository)를 모두 포함하는 BPM 기반 구조에 대한 관리 인터페이스다. 프로세스 서버에 대한 BPMI.org의 BPQL 인터페이스는 비즈니스 분석가들로 하여금 프로세서 서버에 의해서 관리되고 있는 프로세스 인스턴스의 실행에 대한 제어와 상태에 대한 질의를 가능하게 해 준다. 이 인터페이스는 또한 SOAP(Simple Object Access Protocol)에 기반을 둘 것으로 알려져 있다.

프로세스 저장소에 대한 BPQL 인터페이스는 비즈니스 분석가들이 저장소에서 유지되고 있는 프로세스 모델의 배치 및 회수를 관리할 수 있도록 해주며, WebDAV(Distributed Authoring and Versioning Protocol)에 기반을 둘 것으로 알려져 있다. BPQL 인터페이스를 통해 프로세스 저장소에 의해 관리되는 프로세스 모델들은 프로세스의 등록과 광고 및 발견을 위해 UDDI 서비스의 형태로 노출될 것으로 발표되었는데, 이것은 결국 BPQL이 비즈니스 프로세스 정의가 프로세스 서버 상에 배치되고 프로세스 인스턴스가 실시간적으로 조회되고 조작될 수 있도록 해 줄 것임을 의미한다.

이 외에도, BPMI.org에서 밝히고 있는 표준 BPM 스택 상에서 BPQL이 차지하는 위치와 그 의의, 그리고 이러한 BPQL 인터페이스가 WSDL 웹 서비스 정의를 사용하여 제시될 것이라는 점 등이 알려져 있다. 이와 함께, 차츰 그 중요성에 대한 인식이 높아지고 있는 BAM(Business Activity Monitoring)의 기초로써 BPQL이 활용될 것임도 밝히고 있다. <그림 1>은 BPMI.org에서 밝히고 있는 표준 BPM 스택을 나타낸 것으로, BPQL이 웹 서비스 스택 위에서 비즈니스 프로세스의 실행을 담당하는 BPEL 표준과 같은 수준으로 간주되고 있음을 확인할 수 있다[1,7].

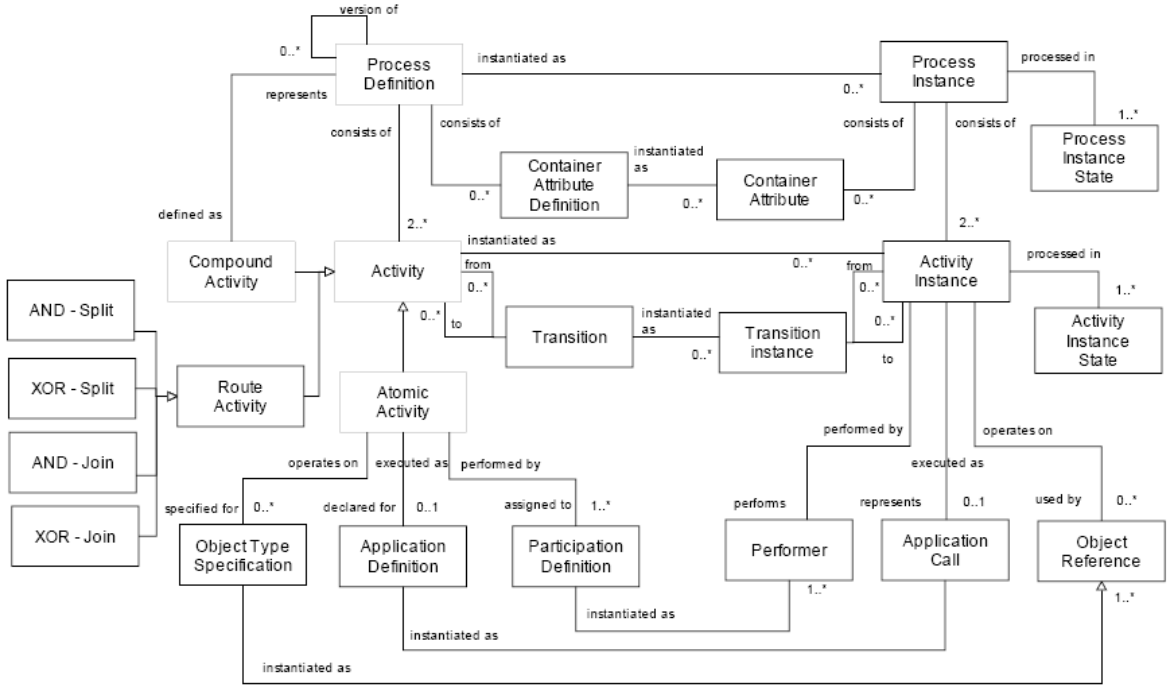


<그림 1> BPMI.org의 BPM 표준 구조

2.2 ICONS 프로젝트의 BPQL 사례

유럽연합의 ICONS 프로젝트는 2002년부터 2년여에 걸쳐 수행된 연구과제로, 다양한 형태로 분산되어 있는 정보 자원들에 대한 통일적이면서도 지식 기반의 접근이 가능하도록 함으로써, 정보 자원이 효과적으로 관리되며 활용될 수 있도록 하는데 목적이 있다. ICONS 프로젝트를 통해 웹 페이지나, 이기종의 데이터베이스, 구조화되어 있거나 비정형의 문서 혹은 멀티미디어 형태의 정보 자원을 대상으로 하는 지식기반의 운영관리 시스템이 프로토타입의 형태로 구현되었다. 특히 이 프로젝트의 일환으로 추진되었던 과제 중에서, 프로세스 지식의 효과적인 관리 및 활용을 위해 BPQL 개념을 정의하고 구현한 사례가 함께 발표되어 많은 연구자들의 주목을 받고 있다.

이 프로젝트를 통해 제안된 BPQL은 Rodan 시스템즈사의 OfficeObjects라는 상업용 WfMS 상에서 구현되어 실제 배치되어 사용되고 있다. 지금까지



<그림 2> ICONS 프로젝트의 프로세스 메타모델

알려진 ICONS 프로젝트의 BPQL 사례를 통해 그 특징을 간략히 살펴보기로 하겠다.

- **유연한 프로세스 정의**

기존의 WfMS가 다양한 업무처리 분야에서 상당한 성과를 거두었지만 여전히 지적되는 한계점의 하나가 바로 프로세스 정의가 동적으로 변화하는 상황에 대한 지원이 취약하다는 점이다. 대체적으로 WfMS에서는 프로세스의 정의가 일단 완성되면 변경되지 않으며, 수정되더라도 부분적인 수준으로 제한된다. 이러한 특징으로 인해서 관리 및 모니터링 프로세스에서와 같이 프로세스의 정의 자체가 실행환경에서 얻어지는 데이터를 토대로 자주 변경되어야 하는 상황에서 유연성을 보이고 있질 못하다. ICONS의 BPQL은 동적인 프로세스 변화가 발생하는 환경에서도 프로세스 정의가 유연성을 가질 수 있도록 프로세스 메타모델에서부터 실행환경 정보에 대한 명세를 포함하고 있다.

ICONS의 BPQL은 현재의 조직구조, 응용프로그램 데이터(Workflow relevant data) 뿐만 아니라 프로세스의 실행 이력(history)에 의존하는 복합적인 동적 요구사항을 프로세스의 정의 단계에서부터 명시적으로 표현할 수 있는데, 이를 위해서 프로세스 정의의 변경에 대한 요구사항을 정확히 형식화하여 표현할 수 있는 프로세스 메타모델을 정의하고, BPQL이 이러한 동적 모델에 대해서 질의할 수 있도록 구성되었다. 제시된 BPQL은 프로세스 정의에 포함되어, 기존에는 실행환경에서 응용프로그램의 변경을 통해 대처해야할 여러 사항들을 쉽게 대응할 수 있게 하였으며, 이미 폭넓게 사용되고 있는 프로세스 정의 형식인 XPDL과 통합하여 BPQL을 사용할 수 있는 방법까지 언급하였다[5].

- **실행환경 정보를 포함하는 프로세스 메타모델**

비즈니스 프로세스에는 크게 두 가지 측면이 존재하는데, 프로세스의 기능과 관련 데이터 및 수행자 등의 정보를 명시하는 프로세스 정의의 측면이 있고, 이를 실제로 프로세스 엔진이 실행(enactment)하여 필요한 자원을 할당하며 관련 데이터의 생성·삭제·변경을 통해 프로세스의 활동 상태를 변경시켜 나가는 실행단계의 측면이 있다. 모든 WfMS와 BPMS는 이러한 두 단계를 프로세스 정의와 프로세스 인스턴스라는 개념을 통해 설명하고 관련된 기능을 제공하는데, 프로세스 정의는 어떻게 프로세스를 쉽게 설계하여 실행시킬지에 주로 초점을 맞추고 있으며, 프로세스 인스턴스는 어떻게 프로세스의 모니터링과 분석을 효과적으로 할 수 있을지에 주로 초점을 맞추고 있다.

기존에 WfMC(Workfolw Management Coalition)에서 제시하는 메타모델이나 개별 업체에서 제시하는 프로세스 모델들 역시 이러한 두 단계의 측면을 각기 기술하고 있는데, 이 두 모델을 얼마나 유기적으로 정의하고 있는가에 따라 비즈니스 프로세스 모델의 표현력과 기능성이 큰 영향을 받게 된다. ICONS 프로젝트의 BPQL도 앞서 언급된 프로세스의 유연성을 제공하기 위해서는 이 두 단계의 모델을 통합하는 프로세스 메타모델을 가져야 하고, 실제의 BPQL 질의는 이런 메타모델을 대상으로 작성되고 실행될 수 있어야 한다.

프로세스 메타모델은 프로세스 엔터티(Entity)와 이들 간의 관계성 및 기본 속성(attribute)들을 정의하며 앞서 언급한 대로, 크게 프로세스 정의 메타모델과 프로세스 인스턴스 메타모델이라는 두 개의 부분으로 구성된다. 전자는 프로세스 정의와 이들 간의 관계성, 그리고 기본 속성들을 정의한다. 이것은 비즈니스 프로세스를 컴퓨터상에 설계하고 구현하기 위해 사용된다. 후자는 프로세스 인스턴스와 이들 간의 관계성 및 기본 속성들의 정의하며, 프

로세스 정의 모델에 따라서 수행된 프로세스의 실행을 표현하기 위해 사용된다. <그림 2>는 ICONS의 BPQL 프로젝트를 통해 두 개의 모델 요소를 통합하여 정리한 비즈니스 프로세스의 메타모델을 간략히 나타낸 것이다.

- **비즈니스 문맥을 반영한 질의 함수의 제공**

SQL이 다양한 함수를 제공하듯이 BPQL도 함수를 제공하는데, 이러한 함수의 호출 결과 또한 또 다른 BPQL 질의문에 자유로이 재사용될 수 있도록 정의된다. 표준함수로는 수학함수(COS, SIN, SQRT 등)와 문자열함수(CONCAT, SUBSTRING), 날짜 및 시간함수(CURRDATE, YEAR, MONTH), 집합함수(AVG, COUNT, MAX) 등이 모두 제공된다.

이런 기본 함수 외에도, ICONS의 BPQL에서는 워크플로우 관리 분야의 특수성을 반영한 별도의 함수 4개와 현재 질의가 처리되고 있는 실행 컨텍스트와 관련된 함수 2개가 추가로 제공되고 있다. 이러한 추가 함수는 BPQL 질의문을 보다 간결하고 명확해게 하는데 도움이 된다. <표 1>은 ICONS 프로젝트에서 추가된 함수들을 정리한 것이다.

<표 1> ICONS의 BPQL에 추가된 함수[5]

함수명		인자 반환값
WF 특화 함수	FirstActInst	ProcessInstance ActivityInstance
	PrevActInst	ActivityInstance ActivityInstance[]
	ActInst	ProcessInstance, ActID ActivityInstance[]
	NextInstance	ActivityInstance ActivityInstance[]
컨텍스트 관련함수	ThisProcessInst	 ProcessInstance
	ThisActivityInst	 ActivityInstance

2.3 SBQL 연구

SBQL은 질의 언어의 구현과 관련하여 비교적 최근에 발표된 연구 결과로, 이후 많은 질의처리 시스템의 구현에 영향을 주고 있다. ICONS의 BPQL 구현에서도 SBQL의 연구 결과를 활용하고 있는데, SBQL은 질의처리 시스템의 구현에 스택기반의 접근법(SBA: Stack-Based Approach)을 사용한 것을 의미한다.

SBQL은 기존의 질의 언어 구현에 비해 간단하면서도 형식적이고 일관된 의미론을 제공하며, 질의 재작성, 인덱스 활용, 결과를 만들어 내지 않는 죽은 질의문의 삭제등과 같은 다양한 질의 최적화 방법론을 제공한다는 장점을 가지고 있다.

SBQL은 일반적인 프로그래밍 언어에서 중요시되는 스코프(scope), 네임(name) 및 바인딩(binding) 요소를 질의 언어의 설계에 적극적으로 수용한 것으로, 자체적인 데이터 모델과 저장 공간을 가지는 추상적인 기계(Abstract Machine)의 개념을 도입한 다음, 스택에 기반을 두어 연산을 진행하도록 하여 스코프를 처리할 수 있게 하였다. SBQL은 질의에서 발생하는 개별 네임이 해당 네임에 대한 스코프와 함께 적절한 실행시간 요소(즉 특정 객체, 속성, 메쏘드의 인자 등)와 결합되어 스택을 통해 유지되도록 한다.

SBA에서는 질의 언어를 프로그래밍 언어의 특별한 형태로 간주한다. 따라서 질의의 의미는 프로그래밍 언어에서 함수의 호출 스택과 같이 잘 알려진 메커니즘에 기반을 두고 있다. 거의 모든 프로그래밍 언어에 구현되어 있는 전통적인 스택의 개념이 데이터베이스 컬렉션과 SQL 및 OQL의 예에서와 같이 모든 타입의 질의 연산자들을 지원하기 위해 확장되었다. SBA는 현재 selection, projection, navigation, join, quantifier 및 다양한 질의 연산자를 제공하고 있다. 스택은 또한 재귀와 파라메터를 지원하므로, SBA에서 정의되는 모든 함수, 프로시져, 메소드 및 뷰에서 재귀가 가능해 진다.

SBA를 사용하여 객체지향 개념에 관계성과 질의 명령어 요소들을 포함시키고, 질의를 근본적으로는 프로시저, 함수, 뷰, 메소드, 모듈 등과 같은 프로그래밍에서 자주 언급되는 요소들에 개념적으로 대응시켜 정의함으로써 질의 언어의 운영상의 의미를 명확히 결정할 수 있다는 장점이 있다.

SBQL은 추상 문법과 연산자의 완전한 온톨로지에 기반을 두고 있기 때문에 관계형 연산 및 계산법과 같은 수학적인 성질을 가지고 있다. SBQL은 명령형의 확장과 추상화와 함께 프로그래밍 언어와 같은 계산 능력도 가진다. 구체적인 문법, 특별한 기능성, 저장모델의 두드러진 특징 및 구체적인 메타모델은 SBQL로부터 비교적 쉽게 BPQL과 같은 구체적인 질의 언어를 만들 수 있을 것으로 기대되어 왔다. 현재 SBQL은 유럽 프로젝트와 DOM 모델을 기반으로 하는 XML 저장소, 그리고 객체지향 DBMS인 Objectivity/DB 상에서 대표적인 구현 사례를 가지고 있다[5,8].

3. BPQL 설계를 위한 고려 사항

BPQL 표준이 갖추어야 할 요구 사항들은 매우 다양하다. 이러한 요구사항들 중에서 개념적으로 언급되는 요소들 중에서 가장 중요한 요소는 바로 질의 언어의 문법과 의미가 명확하게 정의되어 혼동의 여지가 없어야 한다는 것이다. 이것은 다양한 공급자와 사용자 사이에서 중립적인 표준으로 사용되어, 그 의미와 기능에 혼란이 없이 질의 언어의 해석과 실행이 일치해야 하기 때문이다. 이와 함께 질의 언어가 개발자뿐만 아니라 프로세스의 실행과 모니터 및 관리를 담당하는 다양한 이해 관계자 즉, 설계자, 관리자, 분석가 등이 쉽게 이해할 수 있어야 한다는 점도 중요하게 언급되고 있다. 특히 비즈니스 프로세스의 모델링 언어가 주로 객체지향 관점에서 기술된다는 점에서 질의 언어 역시 이에 부합하도록 설계될 필요가 있다.

이외에도 기존의 데이터베이스에서 SQL을 통해 데이터의 조작을 쉽게 할 수 있도록 다양한 함수를 제공하는 것과 유사하게 프로세스 질의 언어에서도 프로세스의 조작을 위한 특화된 함수와 API 등을 제공해야 한다는 기능적인 요구사항들이 존재한다.

여기에서는 이들 개념적 및 기능적인 요구사항들에 대한 내용을 하나하나 찾아서 나열하는 것보다는 이들 개념과 기능들을 제공하기 위해 BPQL 표준이 설계 단계에서부터 기술적으로 고려해야 할 요구사항을 다루고자 한다. 설계 개념의 정립을 위한 기술적 요구사항들을 살펴보는 것은 BPQL 표준

이 갖추어야 할 요구사항들을 보다 명확하고 간결하게 이해하는데 도움을 줄 수 있을 것이다.

3.1 질의를 위한 프로세스 메타모델

ICONS 프로젝트에서 살펴보았듯이 성공적인 BPQL 표준이 정의되기 위해서는 유연한 프로세스 메타모델의 개발이 매우 중요하다. 대부분의 워크플로우 관리 시스템들은 비즈니스 프로세스가 일단 배치되면 해당 프로세스의 정의가 그다지 변경되지 않거나 고정된다는 가정을 두고 있기 때문에 동적인 요소들에 의해 영향을 크게 받는 비즈니스 환경에서 많은 한계를 드러내게 되었다. 기존의 메타모델로는 프로세스 정의가 실행 환경의 잦은 변경 요소들을 반영해야 하는 상황에서 매번 응용프로그램의 코드를 재작성하는 방식을 사용해야 했으며, 이것이 프로세스의 적응성과 활용도를 떨어뜨리는 결과를 가져왔다. 만약 프로세스의 정의에 실행시간 데이터, 실행한 결과 이력 데이터 그리고 현재의 실행 환경 정보를 모두 포함할 수 있다면 프로세스의 적응성은 비약적으로 증가하게 될 것이며, 잦은 응용프로그램의 변경으로 인한 프로세스 대응성 저하도 피할 수 있게 될 것이다. 특히 이러한 요소들이 사전에 정의된 표준화된 용어와 방법론을 통해 프로세스 정의 속에 명시적으로 기술될 수 있다면 프로세스 정의의 가시성을 높이게 될 것이다.

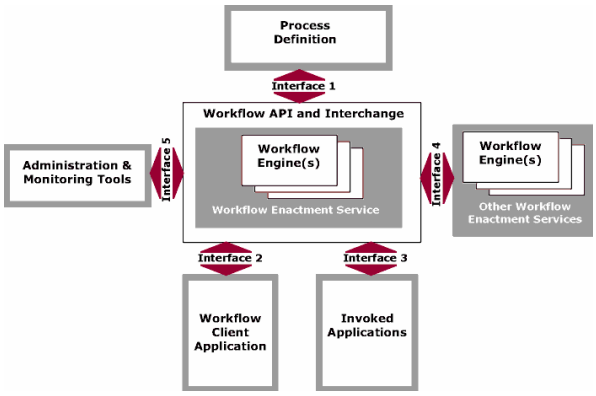
현재 프로세스 정의 메타모델로 제안된 것 중 널리 알려져 있는 것으로는 WfMC 표준의 메타모델이 있다. 이 메타모델은 비교적 잘 정의되어 있지만 프로세스의 실행을 설명하는 메타모델이 빠져 있는 단점이 있다. 이외에도 대부분의 WfMS에 의해 제공되는 프로세스 실행 모델들은 개별적인 도구에 의존적인 경우가 많으며, 상당부분이 주어진 시스템 내에서 해당 요소들의 구현에 초점을 두고 작성된 것들이 대부분이다. 최근 이러한 프로세스 실행모델의 정의를 메타모델에 포함시키려는 노력이 나타나고 있지만, 여전히 시간 관련한 요소의 도입과 동적인 참여자 할당, 실행 환경 정보의 정의와 같은 내용들이 부족한 상태이다. BPQL 표준은 이러한 모든 요소들을 포함하는 프로세스 메타모델로부터 유도되어야 할 것이다.

3.2 표준 인터페이스의 지원

BPQL을 통해 어떤 일을 할 수 있을 지를 파악하는 가장 정확한 방법은 바로 BPQL이 제공하는 인터페이스를 살펴보는 것이다. 프로세스 서버와 주변 액터들의 인터페이스를 살펴보기 위한 좋은 예로써 WfMC에서 제시하는 워크플로우 참조 구조와 다섯 인터페이스를 생각해 볼 수 있겠다. <그림 3>은 WfMC에서 정의하고 있는 워크플로우 참조모델의 다섯 인터페이스를 나타낸 것이다[9].

인터페이스 1번은 프로세스 정의가 모델링 및 프로세스 정의 도구와 런타임 워크플로우 소프트웨어 간에 임포트/익스포트 될 수 있도록 하는 것이다. 이 인터페이스는 기본적으로 프로세스 정의 및 API 호출의 형태와 관련된 것으로, 특정 공급자의 그래픽 모델링 도구 등을 통해 생성된 프로세스 정의가 다양한 물리적 혹은 전자적 교환 매체를 통해 교환되어 다른 공급자의 런타임 실행 엔진에 의해서 실행될 수 있음을 의미한다. 이때, 해당 프로세스 정의의 전체 혹은 일부가 교환될 때 관련된 일련의 속성(attribute)과 활동(activity)들도 함께 교환될 수 있어야 한다.

인터페이스 2번은 워크플로우 클라이언트 응용프로그램과의 인터페이스를 정의하는 것으로 실제 클라이언트 응용프로그램이 어떻게 구현되었는지에 무관하게 워크플로우 엔진과 워크리스트 등에 일관된 형태로 접근할 수 있도록 하는 일련의 표준적인 API 들을 포함하게 된다.



<그림 3> WfMC의 참조모델과 인터페이스[9]

인터페이스 3번은 기존에 존재하는 응용프로그램의 호출을 위한 것으로, 로컬 혹은 원격의 응용프로그램과의 연결을 위해 에이전트 등을 사용하는 형태로 나타나게 된다. 물론 해당 인터페이스를 지원하는 응용프로그램들은 바로 워크플로우 실행 엔진과 통신할 수 있다.

인터페이스 4번은 서로 다른 이종의 워크플로우 시스템 간에도 하나의 비즈니스 프로세스를 성공적으로 진행시키기 위해 상호운용성을 보장하기 위한 것이다. 이를 위해서는 서로 다른 두 실행 엔진 간에 동기 혹은 비동기적인 메시징을 사용하여 액티비티나 서브 프로세스 등을 동기/비동기적인 형태로 호출하여 실행시킬 수 있도록 해야 한다.

마지막으로 인터페이스 5번은 관리 및 모니터링을 위한 인터페이스로써, 서로 다른 워크플로우 영역 간에라도 동일한 접근 API를 통해 사용자, 역할, 감사, 자원 제어 및 프로세스 제어 등의 작업을 할 수 있음을 의미한다.

BPQL은 이상의 다섯 가지 인터페이스에 모두 관여하게 되는데, 이들 인터페이스의 구성을 보다 명백하게 파악할 수 있게 한다. 모든 인터페이스가 명확히 모호성 없이 정의되기 위해서는 인터페이스의 정의에 사용되는 입력 파라미터와 출력 결과 데이터에 대한 용어 정의 및 형태에 대한 결정이 필요하다. 이것들은 또한 BPQL이 기본적으로 이해해야할 프로세스 메타모델의 스키마에 명시되어야 할 것들이다. 따라서 원칙적으로 앞서 언급된 모든 인터페이스에서 나타나는 데이터들은 프로세스 메타모델의 구성요소가 되며, BPQL은 이를 활용하여 개별 인터페이스들의 API 함수들을 대체할 수 있어야 한다.

우선 인터페이스 1과 관련해서는 프로세스에 대한 질의를 진행시키기 위해서 프로세스 모델의 표준적인 기술이 필요하다. SQL이 DDL을 통해 데이터 스키마를 정의하듯, BPQL은 인터페이스 1과 호환되는 접근 방식을 통해 프로세스 스키마를 정의하여 서로 다른 BPM 시스템 간에도 중립적으로 교환할 수 있어야만 한다. BPQL의 관점에서는 프로세스의 정의를 저장하고 교환하기 위한 스키마가

명시되어야 함을 의미하는 것으로, 대부분의 BPMS에서는 이를 제공하기 위한 프로세스 리포지토리의 개념을 함께 고려하고 있다.

인터페이스 2와 관련하여서는 상당히 다양한 BPQL의 활용 예를 생각해 낼 수 있다. 단순히 워크리스트의 인출을 위한 질의로부터 매우 복잡한 조건을 제시하여 검토시키는 질의의 작성이 BPQL을 통해 가능해 질 것이다. 이것은 인터페이스 2에서 정의되는 API를 사용하여 코딩하는 형태보다도 훨씬 선언적인 형태가 될 것이고 활용 면에서 효과적인 방법일 것이다. 특히 현재 프로세스 서버의 실행 환경에 대한 정보를 프로세스 메타모델에 정의된 스키마대로 접근하여 얻을 수 있게 될 경우 클라이언트 응용프로그램의 요구에 맞춰 보다 유연한 BPQL 질의 문의 작성도 가능해 질 것이다.

인터페이스 3의 경우에는 Embedded SQL과 같은 형태의 Embedded BPQL을 유사한 형태로 생각해 볼 수 있겠다. 응용프로그램들은 Embedded BPQL을 통해 복잡한 비즈니스 프로세스 질의를 프로그래밍 코드로부터 분리시켜 모아 놓을 수 있게 되며, 이를 해석하여 실행하고 그 결과를 반환하는 방법 역시 표준화됨으로써 응용프로그램의 중재를 위한 에이전트의 작성이 훨씬 쉬워지게 될 것이다. 따라서 인터페이스 3의 경우, 그 정의가 다른 인터페이스에 비해 나중에 나타날 수도 있겠지만 비즈니스 응용프로그램들과의 유기적인 통합을 위해서는 궁극적으로 Embedded BPQL과 같은 형태로 BPQL의 큰 구조 속에 포함되어 기술되어야 하겠다.

인터페이스 4의 경우에는 유사한 예로 분산된 데이터베이스들 상에서 발생하는 분산 트랜잭션을 생각해 볼 수 있겠다. 서로 다른 데이터 소스로부터 단일의 트랜잭션을 진행시켜 나가기 위한 과정이 분산된 BPM 시스템 상에서 단일의 비즈니스 프로세스를 실행시켜 나가기 위한 상호 협력 과정과 유사하게 설명될 수 있을 것이다. 이러한 분산 비즈니스 프로세스의 진행을 위해서는 물론 분산 트랜잭션의 제어를 위한 프로토콜의 개발에 버금가거나 혹은 그 이상의 프로토콜 요소와 기술이 개발되어야 하겠지만, BPQL 등을 통해서 분산된 프로세스 자원들을 쉽게 지정하고 접근할 수 있는 방법이 제시된다면, 분산 비즈니스 프로세스의 정의 및 실행이 훨씬 수월하게 될 것이다. 최근 웹 서비스 등을 통한 상호운용성의 개선이 비즈니스 프로세스 전 분야에서 활발히 진행되고 있음을 고려할 때, 이에 대한 고려가 BPQL 내부의 설계 시에 반영될 필요가 있다.

마지막으로 인터페이스 5의 경우는 다양한 프로세스 통계와 제어를 위한 BPQL 질의문의 작성과 실행으로 설명될 수 있을 것이다. 여기에는 실행 이력에 대한 데이터가 주로 분석 등에 활용될 것으로 예상되는데, 실행 이력 또한 비즈니스 프로세스의 메타모델을 통해 정의되었기 때문에 BPQL을 통해 쉽게 접근할 수 있을 것으로 예상된다.

이처럼 BPQL은 워크플로우 참조모델에서 언급하고 있는 5가지 표준 인터페이스의 정의 및 활용에 상당한 영향을 주게 되며, 이들 인터페이스를 대체하거나 보완하는 역할을 할 정도의 표현력을 갖추어야 할 것이다.

3.3 기존 프로세스 정의와의 호환성

BPQL을 위해 새로이 정의되는 프로세스 메타모델과 이를 통해 얻어지게 되는 프로세스 정의는 개념적으로는 기존의 프로세스 정의를 포괄할 수 있어야 한다. 호환성 혹은 상위 통합성은 또한 BPQL의 활용도를 더욱 넓히기 위해서도 필요한 요소가 된다. 여기에는 두 가지 측면이 있는데, 하나는 기존의 프로세스 정의 언어로 기술된 비즈니스 프로세스들에 대해서도 BPQL 표준으로 작성된 질의가 작동할 수 있는 실행시간에 대한 측면과 또 다른 측면으로는 BPQL 표준이 기존에 널리 사용되고 있는 XPDL, BPML, BPEL4WS, 혹은 WSDL과 같은 프로세스 정의 언어들의 내부에서 프로세스 정의를 위해서 이용될 수 있어야 한다는 작성시간에 대한 측면이 그것이다.

실행시간에 대한 측면은 각 BPMS를 제공하는 공급자들의 BPQL 표준 적합성 및 기능의 지원 여부에서 그 해답을 얻을 수 있다. BPQL 표준이 모든 BPMS 공급자들에 의해 수용되어, 각각의 BPMS에서 구현된다면 실행시간 측면에서의 호환성은 얻어질 수 있다. BPQL 표준 질의가 기존에 개별 공급자들의 특화된 시스템에 맞도록 작성된 프로세스 정의에 대해 실행될 경우, 해당 BPMS의 공급자들은 BPQL에 명시된 프로세스 데이터와 용어를 자신들의 레거시 포맷에 맞도록 대응시켜 해당 질의의 결과를 성공적으로 실행하여 제공할 책임을 지게 된다. 이것은 BPQL 질의를 수용하는 확장된 프로세스 메타모델을 이해하고 이를 자사의 레거시 형식과 대응시키는 모종의 기술적 구현이 해당 BPMS 공급자에 의해 제동되어야 가능할 것이다.

정의시간 측면은 개별적인 프로세스 정의 언어의 내부에서 BPQL이 어떻게 사용될 수 있을지를 결정한다. 일단 BPQL 표준이 제시될 경우 해당 BPQL의 문법과 의미론에 따라 프로세스 정의의 형태가 중립적으로 결정될 것이다. 그러나 기존의 BPMS 업체에서 널리 사용하고 있는 모든 프로세스 정의 언어가 바로 이러한 새로운 프로세스 정의 언어로 변화되지는 않을 것이다. 이것은 BPQL 표준에서 제공하는 표현력보다 개별 BPMS 업체에서 제공하는 고유 프로세스 정의 언어의 표현력에 더 뛰어난 요소가 있을 경우 더욱 그러할 것이다. 이런 경우 BPMS 제공 업체에서는 자신들의 프로세스 정의 언어의 장점을 활용하고, BPQL 표준에서 제공하는 장점을 수용하기 위한 방안으로 두 프로세스 정의 언어를 혼합하여 사용하고자 시도할 가능성이 높아진다. 즉, 기존의 프로세스 정의 언어의 일부를 변형하거나 확장하여 BPQL 요소를 추가하는 시도가 일어나게 될 것이다. 물론 이렇게 혼합된 형태의 프로세스 정의는 여러 BPMS 시스템간의 호환성을 위해 BPQL 표준에서 정의되는 중립적인 형태의 프로세스 정의로 변환될 수 있을 것이다. 이때 개별 BPMS에서 제공하던 부가적인 요소들이 누락될 위험은 여전히 상존하게 된다.

이상의 두 측면이 BPQL 표준의 설계시에 기존 프로세스 정의 언어와의 호환성과 관련하여 원활히 이루어질 수 있도록 고려되어야 할 것이다.

이 외에도 BPQL 표준이 채용하게 될 문법과 개념이 관계형 모델 보다는 객체지향 모델에 기반을 두어야 할 필요가 있다. 현재 널리 사용되고 있는 대부분의 프로세스 정의 언어들이 객체지향 개념에 근거하여 제시되고 있으며, 특히 기존의 SQL 및 OQL 표준에 대한 기능적인 포함관계를 위해서는 객체지향 개념에 근거하는 것이 보다 자연스럽다고 하겠다. 특히 프로세스 메타모델이 객체와 객체 인스턴스 등의 객체지향적인 개념을 통해서 쉽게 정

의될 수 있고, 객체간의 관계성(association)에 의해 상호작용이 쉽게 기술될 수 있다는 측면에서 더욱 그러하다.

3.4 시간(Temporal) 요소의 지원

WfMC의 워크플로우 참조모델 뿐만 아니라 많은 BPM 모델에서 요구하고 있는 프로세스 정의 요소로써 시간이 있다. 시간 요소는 기존에 실시간 프로세스 데이터의 인출과 같은 연구에서 실행시간 요소에서 많이 그 필요성이 언급되어 왔으나, 프로세스의 정의시간 요소로써도 BPQL에서 매우 중요하게 다루어지고 있다.

BPQL 표준을 통해 BPMS의 실행시간 데이터와 응용프로그램의 실행환경 정보가 프로세스 질의에 포함될 수 있기 때문에 해당 데이터에 대한 절대시간 및 상대시간의 제어 요소가 명확히 정의될 필요가 있는 것이다.

이미 몇몇 BPMS에서 프로세스의 실행 조건이나 트랜지션 조건을 위해 제한적으로 시간 요소를 도입해서 사용해 왔는데, BPQL에서는 이들 요소를 모두 포함하되 시간 요소에 대한 체계를 제시할 필요가 있다. SQL에서 CurrDate와 같이 현재 시스템의 절대 시간을 반환하는 함수가 제공되고 있는데, BPQL에서는 이보다 훨씬 복잡한 시간 요소의 정의와 함수들이 제공되어야만 할 것이다[2].

예를 들어, 특정 프로세스나 활동의 실행 조건 등이 절대 시간에 기초해 정의될 수도 있지만, 병렬적으로 진행되는 다른 프로세스나 활동의 상태 등에 기초할 수도 있다. 이처럼 BPQL 내에서 시간 요소를 지원하기 위해서는 프로세스의 런타임 데이터에 대한 시간 정보를 프로세스 메타모델을 통해 표준으로 정의할 필요가 있다. 즉 프로세스와 활동의 실행시간, 종료시간 등에 대한 정보들이 메타모델에서 정의되어 프로세스 정의에 활용될 수 있어야 하며, BPQL은 이러한 시간 요소를 질의 속에 포함할 수 있어야 한다.

4. BPQL 참조 아키텍처

BPQL에 대한 표준화 움직임이 아직 결과를 보여주지 않은 상태이지만 BPM 시스템 공급 업체들은 이미 나름대로의 질의 처리 시스템을 갖춘 제품을 시장에 내놓고 있으며, 향후 BPQL 표준이 제정되었을 때 자사의 제품에 이를 효과적으로 반영하길 원한다. 이를 위해서는 업체마다의 개별적인 특징을 살리는 구현보다는 여러 BPQL 관련 연구들로부터 장점과 공통점을 찾아내고, 이를 포괄할 수 있는 구현 아키텍처를 가지는 게 중요하다.

본 연구에서 제시하는 <그림 4>의 참조 아키텍처는 이러한 BPM 솔루션 제공 업체들의 표준 대응력을 높이는데 도움이 될 것이다. 참조 아키텍처 속에 표현된 전체 요소들에 대해서 개략적인 기능과 내용들을 살펴보면 다음과 같다.

- **Client Access/Legacy Client Access**

BPQL 질의처리 시스템에 접근하는 클라이언트는 사용자뿐만 아니라 응용 프로그램 및 BPQL 시스템 자체가 모두 될 수 있는데, 이들은 사용하는 인터페이스에 따라 크게 두 가지로 구분할 수 있을 것이다. 새로운 BPQL 표준을 인식하고 이에 준해서 접근을 하는 Client Access가 있고, 과거에 사용하던 질의 언어나 해당 BPMS 업체의 독자적인 질의 방식에 따라 접근을 하는 Legacy Client Access가 있다.

- **Query Rewriter**

과거의 레거시 형식으로 BPQL 시스템을 접근하는 클라이언트의 경우, 기존의 질의문을 새로운 BPQL 질의문으로 재작성해주는 컴포넌트가 필요하다. Query Rewriter는 기존 사용자를 위한 배려차원에서 제공되는 기능으로, XPDL과 같은 기존의 프로세스 정의 표준과의 호환성을 위해서도 중요하게 다루어져야 한다.

- **BPQL Graphic Window**

인간 사용자의 즉각적인 BPQL 질의 처리를 위한 그래픽 인터페이스로, 과거의 데이터베이스관리 시스템에서 SQL Window나 SQL 명령창과 같은 형태로 제공되는 컴포넌트에 해당된다.

- **Embedded BPQL API**

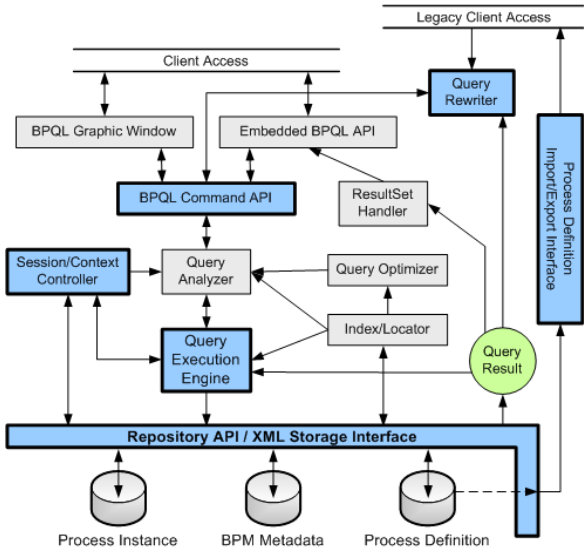
기존의 Embedded SQL과 같이 다른 응용 프로그램의 코드 속에 BPQL 질의를 내포하여 사용할 수 있도록 제공된다.

- **BPQL Command API**

BPQL 표준에서 정의하는 모든 함수와 확장 기능, 상수 등에 대한 핵심 인터페이스를 제공한다. 외형적으로 특정 BPMS가 표준으로 정의된 BPQL API를 완전히 지원하는 지의 여부를 결정하는 핵심 컴포넌트가 된다. 사용자로부터 BPQL Graphic Window를 통해 입력된 BPQL 질의와 Embedded BPQL API를 통해 전달된 BPQL 질의가 모두 이 Command API를 사용하여 처리될 것이다. Legacy Client Access를 통해 생성되어 Query Rewriter를 통해 재작성 된 BPQL 질의도 이것을 사용하여 BPQL 질의를 시스템에 전달하게 될 것이다.

- **Query Analyzer**

질의문을 파싱하여 실제의 실의 실행 경로를 작



<그림 4> BPQL 구현 참조 아키텍처

성하는 분석기이다. 모든 질의처리 시스템에서 제공해야만 하는 요소로 대부분 질의 최적화를 위한 분석을 추가로 제공한다. 종종 Query Execution Engine과 결합되어 설명되기도 하지만, 본 연구에서는 두 요소를 개념상 분리하여 도시하였다.

- **Query Optimizer**
  최적의 인덱스 선정, 질의 경로의 재작성, 무의미한 질의 처리의 생략, 조인 및 네비게이션 순서의 변경 등을 통해 BPQL 질의의 수행 성능을 높이기 위한 기법들을 제공한다. 실제 BPMS 내에 Materialize된 인덱스와 데이터가 저장된 물리적인 페이지의 위치를 확인하기 위해 Index/Locator와 상호작용을 할 필요가 있다.

- **Index/Locator**
  BPMS 상에서 관리되는 데이터 요소들에 대한 인덱스 생성 및 관리를 담당한다. 저장된 데이터 요소들의 물리적 페이지에 대한 정보를 제공하며 새로운 물리적 페이지의 할당과 해제에 관여한다.

- **Session/Context Controller**
  현재 처리 중에 있는 질의문의 문맥과 사용자 세션에 대한 제어를 담당한다. 특히 BPQL 질의와 관련해서는 지금 실행 중에 있는 프로세스 인스턴스의 실행 시간 문맥 정보를 제공하기 위해 ICONS BPQL에서 추가되었던 함수들과 같은 요소를 제공하게 된다. 사용자 세션과 문맥 정보를 저장소에 기록하기 위해 Repository API를 사용하게 된다.

- **Query Execution Engine**
  Query Analyzer에 의해 최적화되어 작성된 최종 질의문을 처리하기 위해 Index/Locator를 사용하여 물리적인 데이터베이스 페이지에 접근하는 실행 엔진이다. SBQL과 같이 스택기반의 접근법을 사용할 경우에는 Session/Context Controller와 함께 스택의 형태로 통합 구현될 것이다. 중첩된 BPQL 질의의 처리를 위해서 Query Result를 다시 활용하여 후속 처리를 진행할 수 있도록 작성되어야 한다.

- **Repository API/XML Storage Interface**
  실제 비즈니스 프로세스의 정의, 실행 인스턴스, BPMS 제어 및 메타정보가 저장되는 데이터베이스와의 중계 인터페이스의 역할을 한다. 물리적인 저장소는 관계형 혹은 객체지향형에 무관하며, 이 인터페이스가 제공하는 서비를 통해 객체-관계형의 매핑이 제공될 것이다. 일반적으로 프로세스 정의와 관련 첨부문서 등이 XML 형태로 작성될 수 있기 때문에 XML 저장 인터페이스가 함께 제공될 필요가 있다. 물론 이를 위해 XML DB라 XML 저장소가 사용되는 것도 무방하다. 일반적으로 XPath와 XQuery와 같은 저수준의 인터페이스가 포함되어 구현될 것이다.

- **Process Definition Import/Export Interface**
  외부의 BPMS나 WfMS와의 정보 교환을 위한 인터페이스로 주로 BPEL4WS나 XPDL 등과 같은 표준 형식을 통해 프로세스 정의를 생성하거나 읽어 들이기 위해 사용하게 된다.

- **Query Result**
  지금까지 설명된 요소들이 소프트웨어 컴포넌트였다면, Query Result는 특정 소프트웨어 컴포넌트가 아니라 실행 시간에 구체화되는 객체를 나타내는 것이다. 일반적으로 질의 처리의 결과로 얻어지는 데이터는 다시 부속 질의의 처리를 위해 재사용될 수 있기 때문에, Query Result가 BPQL 질의에 재사용될 수 있는 형식으로 작성되어야 함을 표현하기 위해 도시되었다.

- **ResultSet Handler**
  Embedded BPQL 질의문의 경우, 응용 프로그램 내부에서 커서를 사용하여 Query Result를 접근하고 조작할 수 있어야 한다. Embedded SQL 표준에서와 같이 개념으로 BPQL에서도 제공되는 것으로, Query Result를 응용 프로그램이 접근할 수 있도록 커서 기능을 제공하고 질의 처리 결과를 버퍼링하여 관리할 수 있어야 한다.

5. 결론

현재까지의 BPM 시스템들이 갖추고 있는 질의 처리 기능은 SQL 기반위에 대표적인 질의 유형을 미리 폼과 같은 형태의 그래픽 환경으로 구현하여 제공하는 수준으로 그 유연성과 확장성에 많은 문제점을 지니고 있다. 스크립트 기반의 질의 처리 환경을 제공하는 몇몇 제품들의 경우에도 비즈니스 프로세스 환경에 맞도록 컨텍스트를 반영한 특화 함수들까지 지원하는 경우는 매우 드물다.

이제라도 BPQL 표준이 등장할 상황에 대비하기 위해서는 기존 BPQL 연구로부터 장점과 공통점을 추려내어 이를 중립적인 참조 아키텍처로 구현한 필요가 있다. 본 연구에서 제시하는 설계 요소와 참조 아키텍처는 이러한 BPM 시스템 제공 업체의 표준 대응력을 높이는데 기여할 것이다.

참고문헌

[1] 한국전산원, (2004), 비즈니스 프로세스 관리기술 표준적용을 위한 지침 연구.
[2] Eder, J., Paganos, E., (2001), Managing Time in Workflow Systems, in Workflow Handbook 2001, Layna Fischer (Ed.), Future Strategies Inc., USA.
[3] Dieter König, (2005), Web Services Business Process Execution Language(WS-BPEL), OASIS Open Standards Day XTech 2005 Conference Amsterdam.
[4] H. Smith and P. Fingar, (2003), Business Process Management: The third wave, Meghan-Kiffer Press.
[5] ICONS, IST-2001-32429, 5th EC Framework Programme, www.icons.rodan.pl
[6] Martin Owen and Jog Raj, (2003), BPMN and Business Process Management, http://www.bpmi.org
[7] Momotko M., Subieta K., (2004), Business Process Query Language - a Way to Make Workflow Process More Flexible, ADBIS'2004, Budapest, Hungary.
[8] Subieta K, Beeri, C., Matthes, F., Schmidt, J., W., (1994), A Stack-Based Approach to Query Language, East-West Database Workshop.
[9] Workflow Management Coalition: The Workflow Reference Model, WfMC-TC-1003 issue 1.1, Jan 1995.