OCL을 사용한 데이터베이스 제약사항 일관성 확인

박찬호° 최윤석 정기원
숭실대학교 대학원 컴퓨터학과
{afteroneday°, secooling}@hanafos.com, chong@computing.ssu.ac.kr

A Consistency Validation of Database Constraints Using OCL

Chanho Park° , Yunseok Choi , Kiwon Chong
Soongsil University.

요 약

데이터베이스에 대한 제약사항들은 소프트웨어의 개발 및 사용에 있어 개발자와 사용자가 모두 참고해야 하는 중요한 사항임에도 불구하고 데이터베이스 모델링과 소프트웨어 모델링의 불일치 및 표현 방법의 차이로 인해, 일관성있는 제약사항 명세 및 설계에 많은 어려움을 겪어왔다. OCL(Ocject Constraint Language)은 객체의 제약사항을 표현하는 언어로서 UML(Unified Modeling Language)의 일부분이다. 본 논문은 객체의 제약사항을 표현하는데 사용하던 OCL을 데이터베이스의 제약사항을 설계에 일관성있게 표현, 반영하기 위하여 사용하는 기법을 제시하고, 이에 대한 검증을 수행하는 방법에 대한 연구이다.

본 논문에서는 요구사항에서 추출한 데이터베이스의 제약사항을 정의한 규칙에 따라 추출해내고, 이를 OCL의 표현법을 이용하여 정형적으로 표현한다. 그리고 그 OCL의 형식적 표현을 ODL(Object Description Language)로 정의하는 기법을 정의함으로서 이에 대한 일관성 검증을 수행하는 기법을 제안한다.

1. 서 론

데이터베이스에는 설계, 구축되어 운용되어지는 동안 반드시 지켜져야 하는 제약사항들이 존재한다. 이를 일반적으로 Integrity Constraint[3][6]라고 한다. 해당 제약사항들은 소프트웨어를 개발하는 개발자 및 사용자들이 반드시 알아둬야 하는 사항이지만, 소프트웨어를 개발해 나가는 동안 이것이 어떻게 일관성이 지켜져 가는지에 대한 정의도 명확하지 않고, 수행하는 방법 역시 명확하게 정의되어지지 않는 경우가 많다. 그리고 이에 대한 정의가 약간씩 다르고 이를 표현하는 방법도 객체지향 표현방법인 UML[2][4][5]에서는 마땅치 않아 데이터베이스 설계 명세를 반드시 참조해야 한다는 불편이 존재한다.

따라서 본 논문에서는 객체지향방법론인 마르미[9]의 데이터베이스 명세서의 형식을 빌려 요구사항에서 데이터베이스의 제약사항을 추출해내고, 이를 UML의 일부로서 포함되어져 있는 OCL[1][7]의 표기법을 사용하여 데이터베이스의 제약사항을 나타낸다. 즉 요구사항, 형식적 정의, OCL 표현, ODL 명세로의 일관성 있는 변환 기법을 사용함으로서 제약사항의 일관성을 유지하고 검증하는 기법을 제시한다.

2. 관련연구

OCL은 UML Specification의 부분으로서[1][2], 객체지향 모델과 객체 모델 산출물상에서 제약사항을 묘사, 표현하기 위하여 사용하는 언어이다.[7] 기존 UML표현에서 제약사항은 '{}' 표기를 사용해서 해당 괄호안에 형식에 구현받지 않고 자유로운 기술을 통해 표현하고 있으며[5], OCL을 사용할 경우 제약사항을 정형적으로 표현할 수 있으며, 의미의 명확성을 부여할 수 있다[1][7].

현재 데이터베이스의 제약사항은 그 정의가 매우 넓고 방대해서 여러 저서에서 다소 다양한 범위에 걸쳐 분류하고 있다.[3][6] 그러나 데이터베이스에 대한 제약사항을 UML에서 따로 명세하거나 표현하지는 않고 있으며, 객체지향 방법론에서 데이터베이스에 대한 설계는 클래스 다이어그램이 어느 정도 나온 후 그에 대한 속성값을 사용하여 요구사항 명세서의 내용과 맞추어 설계하는 방법이 일반적이다.

3. 제약사항의 정의 및 분류

요구사항에서 나타난 데이터베이스의 제약사항을 변환하는 기법에 대해 기술하기 전에, 제약사항에 대한 여러 관점을 살펴보도록 한다. 데이터베이스 관점에서 보는 제약사항과 객체지향 모델링 관점에서 보는 제약사항, 그리고 객체지향 모델의 표현 관점에서 보는 제약사항의 정의를 살펴보도록 한다. 일차적으로 각 관점에서 보는 정의를 살펴보면 [표 1]과 같다.

[표 1] 관점에 따른 제약사항의 정의

	제약사항의 정의
데이터베이스 관 점	값이 반드시 True가 되어야 하는 Boolean function이다. 제약사항을 위반하는 데이터베이스 수정은 DBMS에 의해 거부되어지며, 모델상에서 표현되지 못하는 실세계에서의 중요한 단면적이고 부가적인 정보들을 제약사항의 형태로 사용하여 나타낸다. [3]
OO 모델링 관 점	객체모델의 entity사이의 기능적 관계로서, 반드시 유지되어야 하는 상태의 의미적 표현이다. Design by Contract개념에 의거하여 사용되어진다. [4]
OO 모델의 표현 관점	객체모델 및 시스템에 대한 하나이상의 제한사항으로서 구성요소의 묵적의 표현이다. 제약사항은 Assertion과 rule의 조합으로서 하나의 객체가 다른 것에 의해 영향을 받는 방법의 의미를 표현하는 것이다. [7]

한편 데이터베이스의 완전 제약사항[3](Integrity Constraint)을 항목별로 분류하여 보면 [표 2]와 같다. 데이터베이스의 완전 제약사항은 일반적으로 데이터베이스가 지켜야 할 사항들이지만, 앞에서 명시한 대로 기준이나 분류기준이 워낙에 다양하므로 본 논문에서는 [3]에서 제시한 분류기준을 기본으로 사용한다.

[표 2] 제약사항의 분류

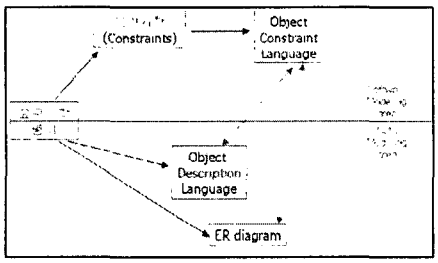
	정 의
Key	key는 entity set내에서의 entity또는 Class내에서의 object를 유일하게 식별하는 속성의 집합이다.
Single value	어떤 role에서의 값이 유일해야 한다는 요구사항이다. 여기에 속하는 것으로는 unique과 null이 있다.
Referential Integrity	데이터베이스에서 실제로 존재하는 다른 객체에 연관된 값에 대한 요구사항이다.
Domain	속성의 값이 반드시 명시된 값의 범위내에 놓이거나, 명시된 set 내에서의 값이 되어야 한다는 것이다.
General	데이터베이스 내에서 지켜져야 하는 일종의 assertion으로서 고객의 요구사항에 의거해서 매우 다양한 형태를 취한다.

객체 및 데이터베이스에 대해 제약사항을 표현하는 장점에는 향상된 문서화, 개선된 정확성, 오해 없는 의사소통 등이 있다.

4. 일관성 유지를 위한 DB제약사항 표현 절차

본 논문에서는 [표 1]에서의 데이터베이스 관점에서 [표 2]와 같이

분류된 제약사항을 정의한 정형적인 명세로 변환하고 이를 다시 OCL 형태로 변환한다. 그리고 이로부터 ODL로 변환하는 기법을 제시한다. OCL과 ODL에서의 제약사항을 명시하는 기법을 매핑시킴으로서 제약사항의 일관성을 유지한다. 이러한 절차를 그림으로 나타낸 것이 [그림 1]이다. 요구사항에서 제약사항을 추출해내기 위해서 객체지향 방법론인 마르미의 데이터베이스 설계 명세서의 양식을 사용한다.[9] 즉 객체관점의 제약사항을 표현하던 OCL을 사용하여 데이터베이스의 제약사항을 표현하고 이를 사용하여 데이터베이스 제약사항 설계의 일관성 확인을 위한 수단으로 사용한다.



[그림 1] 소프트웨어 관점과 DB관점의 매핑

5. 제약사항의 변환

고객의 요구사항은 워낙 다양하고 일정한 형식의 형태가 갖추어진 것이 아니다. 따라서 일단 제약사항에 관한 요구사항을 형식에 맞게 표현하기 위해서는 이를 일단 정형화된 형태로 정제해야 한다. 이를 위해서 마르미 방법론에서의 데이터베이스 설계서[9]의 형식을 사용하여 요구사항 명세서에 명시되거나 또는 설계가 진행되면서 나타나는 제약사항을 기입한다.

데이터베이스 명세서에 기입된 제약사항을 정형적으로 추출해내기 위하여 [표 3]과 같이 정의된 규칙을 사용하도록 한다.

[표 3] 제약사항의 정형적 표현 규칙

	정형적 규칙에 의한 제약사항 표현방법
key	PK -> ta_name.attr_name
single value	UNIQUE -> ta_name.attr_name
	NULL -> ta_name.attr_name
referential	RIC -> referential_ta_name.referential_attr_name => referenced_ta_name.key_attr_name
domain	RANGE -> minimum_value <= ta_name.attr_name <= maximum_value
	FIELD -> ta_name.attr_name = [FIELD1_name(range_value)|FIELD2_name(range_value)|...]
	LENGTH -> ta_name.attr_name = length_value
general	ASSERTION -> [condition]
	MULTI -> ta1_name.attr_name[number_of_attr1] : ta2_name.attr_name[number_of_attr2]

[표 3]에서 예약어는 모두 대문자로 표시하였으며, ta는 해당 테이블을 나타내고, attr은 해당 속성을 나타낸다. 마르미에서는 데이터베이스 설계서의 테이블정의서에서 "기본키"컬럼을 따로 명시해서 표현하므로 이를 Key로 추출한다. single value는 "Null/Unique"컬럼에서, referential은 테이블 정의서의 유도 데이터 컬럼을 정의해두었다가 사용한다. domain 제약사항은 테이블 정의서의 제약사항 컬럼과 길이 컬럼에 표현된 것을 변환한다. 마지막 general은 요구사항에서 고객이 필요로 하는 것을 찾아서 정의해야 하는데 이것은 너무 범위가 넓고 형태도 다양하므로, 본 논문에서는 데이터베이스의 assertion에 해당하는 것과 multiplicity에 해당하는 것, 두가지로 분류하고 변환하는 것으로 한다.

정의한 방법을 사용하여 정형화 형태로 제약사항을 표현한 후 이것을 이용하여 OCL표현으로 변환한다. 앞에서 밝힌 바와 같이 OCL은 객체의 제약사항을 나타내기 위한 명세언어이기 때문에[1][7], 데이터베이스의 제약사항을 완벽하게 정의할 수 없다. 따라서 본 논문에서는 OCL의 표현방법을 가져다 쓰되, 그 문맥적 의미는 다시 정의하여 사용하기로 한다. 그 변환을 나타낸 것이 [표 4]이다. OCL은 테이블 이름을 명시하고 그에 대해 제약사항을 명시해주는 형식을 취하고 있는데[1] 그에 대한 예시로서 key constraint를 명시하는 OCL 표현을 나타낸 것이 [그림 2]로서 [표 4]의 다른 제약사항들도 [그림 2]와 같은 형식에 내용만 변경된 형태로서 나타낸다.

```
Table_name
self.primary_key_attribute = table_name
```

[그림 2] key constraint의 OCL 표현

[표 4] 제약사항의 OCL 표현 규칙

[표 4]에서도 [표 3]에서와 마찬가지로 ta는 테이블을, attr은 속성

	OCL표현에 의한 제약사항 표현방법
key	SELF.primary_key_attr = ta_name
single value	attr_name -> COUNT (attr_name) = 1
	attr_name -> COUNT (attr_name) >= 0
referential	Referenced_ta_name.key_attr_name -> INCLUDES (SELF.attr_name)
domain	attr_name >= minimum_value AND attr_name <= maximum_value
	attr.SUBSTRING (field1_start_value , field1_end_value) = 'field1_name'
	attr_name.SIZE = length_value
general	SELF.attr_name = related_ta_name.related_attr_name
	attr_name -> EXISTS(RANGE_EXPRESSION)
	SELF.attr_name.ALLINSTANCES.COUNT = number_of_tuple1 : ta2_name.ALLINSTANCES.COUNT = number_of_tuple2

를 나타내며, 대문자로 표시한 것은 예약어이다. Key처럼 OCL의 원래 의미로 적합하게 나타낼 수 없는 것은 표현에 해당하는 의미를 바꾸어서 나타내었다. 여기까지의 과정을 통해 요구사항에서 시작하여 5가지로 분류한 데이터베이스 제약사항을 정형적 규칙을 거쳐 OCL표현으로 변환하는 기법을 제시하였다. 위에서 제시한 OCL표현기법은 사실상 타 데이터 모델링 기법에서의 상세 설계 내용을 거의 모두 포함한다. 앞에서 말한 것처럼 ODL 자체가 명세언어이니만큼 모든 제약사항을 나타낼 수는 없다. 하지만 현재까지의 진행에서 표현된 세부명세에서 각각의 attribute, table, relationship에 대해 정의한 제약사항을 가지고 있으므로, 이것을 사용하여 ODL 명세를 표현하여, 데이터베이스 설계 과정에서의 제약사항의 일관성을 유지할 수 있다.

6. 기존 방법 분석

데이터베이스 제약사항을 명시하던 기존의 모델링 방법으로 대표적인 것은 ER(Entity Relationship) 다이어그램과 ODL(Object Description Language) 명세가 있다. [표 5]는 해당 방법으로 제약사항의 명세 및 표현이 가능한지의 여부를 나타낸 것이다.

[표 5] 제약사항의 표현가능 여부

		ODL	ER	결과
key		○	○	○
single value	unique	○	△	○
	null	×	△	△
referential		○	○	○
domain	range	×	△	△
	field	×	△	△
	length	×	○	○
general	assertion	×	▲	▲
	multiplicity	▲	○	○

○ : 명세가능 표현가능 , △ : 명세가능 표현불가능
▲ : 일부만 명세가능 표현불가능 , × : 명세불가능 표현불가능

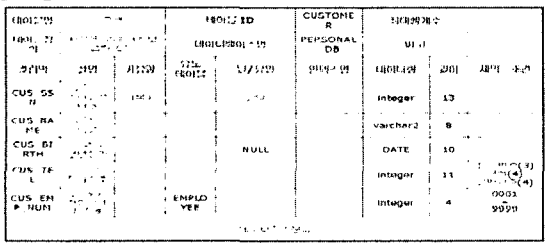
즉, 기존에 데이터 모델링을 수행하던 두가지 방법에서 제약사항의 일부는 표현이 가능하지만, 일부는 표현하거나 구체적으로 명세하는

것이 불가능하다. 이때 다음 단락에서 제시하는 설계과정 전반에 걸쳐 일관성 있는 데이터베이스 제약사항 명세 및 표현이 가능하다. 기존에 데이터모델링의 수단으로 사용하던 두 가지 방법, ODL과 ER에서 표현이 가능한 것과 안되는 것을 명시하였다. 그리고 기존 방법으로 표현이 가능한지의 여부를 파악하기 위해 결과 항목을 따로 두었다. 즉 결과 항목에서 표현이 가능하다고 나온 것은 ER이나 ODL을 사용하여 표현하는 것이 가능한 것이다. [표 5]에서 보는 바와 같이 결과에서 명시된 바에 따라 해당 제약사항의 표현가능 여부를 확인할 수 있다.

우선, 제시한 기법에 따라 생성한 OCL표현에서 ER이나 ODL로 표현할 수 있는 제약사항은 표현하도록 한다. 예를 들면 Key같은 경우, ODL과 ER로 모두 명세가 가능하므로 둘중 하나에 표현한다. 그리고 전체 내용에 대한 명세 또는 일부에 대한 명세만 가능하고 표현은 되지 않는 제약사항에 대해서는 OCL표현을 노트형식으로 부가한다. 이렇게 하면 별도의 데이터베이스 제약사항을 다른 문서를 찾아보지 않아도 되기 때문에 편리할 뿐 아니라, OCL표현의 정형성을 통해 제약사항의 의미를 정확하게 명시할 수 있다.

7. 실제 사례 적용 및 분석

이해를 돕고자 실제 사례에 위 기법을 적용한다. 다음은 .NET기반 방법론인 HySTEP에 포함된 예제의 일부를 발췌한 사례에 제안한 기법을 적용한 예이다.



[그림 3] 예제용 마르미의 데이터베이스 설계명세서

[그림 3]은 예제를 마르미의 산출물 양식을 사용해서 나타낸 것이다. 고객이 정의한 요구사항 명세서를 보고 [그림 3]과 같은 형태로 데이터베이스의 제약사항을 추출한다. 추출한 제약사항을 일단 정형적 언어로 표현하기 위해 [표 3]에서 명시한 규칙에 의해 정형적 표현으로 변환한다. 변환 결과는 [그림 4]와 같다.

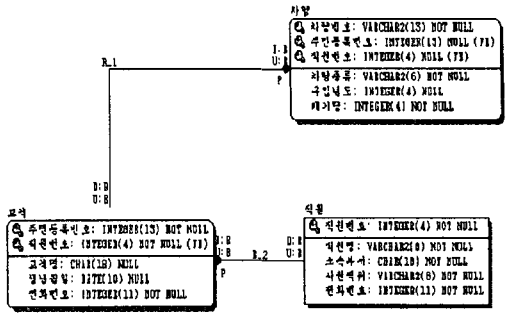
```
PK -> CUSTOMER.CUS_SSN
NULL -> CUSTOMER.CUS_BIRTH
RIC -> CUSTOMER.CUS_EMP_NUM=>EMPLOYEE.EMP_NUM
RANGE -> 0001 <= CUS_EMP_NUM <= 9999
FIELD -> CUSTOMER.CUS_TEL = [(3)|(4)|(4)]
LENGTH -> CUS_SSN = 13
LENGTH -> CUS_NAME = 8
LENGTH -> CUS_BIRTH = 10
LENGTH -> CUS_TEL = 11
LENGTH -> CUS_EMP_NUM = 4
MULTI -> CUSTOMER.CUS_SSN[1] : CAR.OWNER_SSN[*]
```

[그림 4] 예제 테이블의 제약사항의 정형 명세

[그림 4]에 나타난 정형명세를 [그림 2]와 [표 4]에서 명시한 방법대로 나타내면 [그림 5]와 같다. [그림 5]에서는 앞에서 분류한 대부분의 제약사항을 포괄하고 있다. 제안한 기법의 간단한 적용을 통해 제약사항을 OCL의 표현을 이용하여 정형화되어진 방법으로 표현하는 것이다. 그리고 요구사항에서 정형표현, OCL표현을 거쳐 [그림 6]의 (ㄱ)과 (ㄴ)에 나온 것과 같은 데이터베이스 제약 모델링 표현으로 변환함으로써 데이터베이스 제약사항 모델의 일관성을 추구할 수 있다.

```
CUSTOMER
self.CUSTOMER.CUS_SSN = CUSTOMER
CUS_BIRTH -> count(CUS_BIRTH) >= 0
EMPLOYEE.EMP_NUM -> includes(self.CUS_EMP_NUM)
CUS_EMP_NUM >= 0001 and CUS_EMP_NUM <= 9999
CUS_TEL.substring(1,3)=
CUS_TEL.substring(4,7)=
CUS_TEL.substring(8,11)=
CUS_SSN.size = 13
CUS_NAME.size = 8
CUS_BIRTH.size = 10
CUS_TEL.size = 11
CUS_EMP_NUM.size = 4
CUS_SSN.allInstances.count=1 : CAR.allInstances.count=3
```

[그림 5] 예제 테이블의 제약사항의 OCL명세



(ㄱ) 예제의 ER 표현

```
interface CUSTOMER {
    unique attribute integer CUS_SSN;
    attribute string CUS_NAME;
    attribute Date CUS_BIRTH;
    attribute integer CUS_TEL;
    attribute integer CUS_EMP_NUM;
    (key (CUS_SSN));
    relationship SET<CAR> ownCars;
    relationship EMPLOYEE sellCars;
};
```

(ㄴ) 예제의 ODL 표현

[그림 6] 예제의 데이터베이스적 표현

기본 카테고리에 있는 제약사항을 두 가지 방법으로 나누어서 매핑이 수행되어진 것을 알 수 있고, 제약사항이 요구명세에 나온 것과 일관성 있게 표현되었음을 알 수 있다.

8. 결론 및 향후연구방향

본 논문에서는 데이터베이스 제약사항의 일관성을 분석 및 설계 과정에서 유지하기 위해 제약사항 추출규칙, OCL표현규칙, OCL표현의 데이터베이스 모델로의 변환 기법등을 제시하고 예제를 사용하여 실제 사례에 적용하여 보였다. 기존에 객체 모델의 제약사항을 명시하기 위한 OCL의 의미를 변경하여 데이터베이스의 제약사항을 명시하기 위하여 사용하였다. 이를 통해 객체지향 모델에서 표현한 DB제약사항을 일관성있게 DB모델로 적용할 수 있다. 향후연구로는 추출한 제약사항을 UML 다이어그램에 부가하는 방법에 대한 연구와, 제약사항 카테고리를 더 세부적이고 정확하도록 발전시키는 연구가 필요하다. 정확한 정의 및 분류기준 확립 후 이에 대한 구현 및 검증절차를 자동화하는 것에 대한 연구가 필요하다.

참고문헌

[1] OMG. Object Constraint Language Specification. In OMG Unified Modeling Language Specification version 1.4, 2001

[2] OMG. OMG Unified Modeling Language Specification version 1.4, Object Management Group, Inc. 2001.

[3] Jeffrey D.Ullman. A First Course in Database. 1997.

[4] Martin Fowler. UML Distilled Third Edition. 2004.

[5] Grady Booch, James Rumbaugh, Ivar Jacobson, The Unified Modeling Language User Guide, Addison-Weslay, 1999

[6] Silberschatz, Korth, Sudarshan. Database System Concepts 4th Edition. McGrawHill, 2002.

[7] Jos B.Warmer, Anneke G. Kleppe. The Object Constraint Language Precise Modeling with UML, Addison-Weslay, 1999.

[8] 하일규, 강병욱, OCL을 이용한 UML Diagram의 일관성 및 정확성 검증 방법, 제18회 한국정보처리학회 추계학술발표대회 논문집 제9권 2호, 2002.11

[9] 객체지향 방법론 마르미(MARMI)-II