The Journal of GIS Association of Korea, Vol. 12, No. 3, pp. 239-249, November 2004

다중축척 공간 데이터베이스의 데이터 갱신

권오제* · 강혜경** · 이기준*

Data Update on Multi-Scale Databases

O-Je Kwon* · Hae-Kyong Kang** · Ki-Joune Li*

요 약

다중축척 데이터베이스는 동일한 공간을 다른 축척으로 표현하는 공간데이터베이스의 집합이다. 이 다중축척 데이터베이스는 소축척의 정밀한 공간데이터베이스로부터 유도될 수 있다. 본 논문은 유도된 다중축척 데이터베이스의 갱신문제를 다루고자 한다. 다중축척 데이터베이스의 갱신은 수정된 원시데이터로부터 직접 유도된 데이터뿐만 아니라 갱신되지 않은 원시데이터로부터 유도된 데이터들까지 갱신해야 한다. 이것은 현재 데이터 갱신방법들을 다중축척 데이터베이스에 그대로 적용할 수 없는 이유가 되는데, 현재 방법들은 수정된 원시데이터로부터 직접 유도된 데이터 만을 갱신하기 때문이다. 이 다중축척 데이터베이스의 갱신관리는 공간 데이터베이스 관리시스템(혹은 GIS)에서 제공되어야 할 중요한 기능이다. 이 논문에서는 다중축척 데이터베이스 갱신을 위한 규칙 및 알고리즘을 제안하고, 이것을 ESRI의 ArcObject를 이용하여 개발한 프로토타입을 소개한다. 본 연구에서 제공하는 갱신방법은 다중축척 데이터베이스들간의 일관성을 유지시켜주고, 유도된 다중축척 데이터베이스의 무결성을 보장해 줄 수 있다는 점에서 의의가 있다.

주요어 : 다중축척 공간데이터 베이스의 무결성, 갱신, 유도된 데이터의 일관성

ABSTRACT : This paper discusses on the update problem of multi-scale databases when the multi-scale databases, which is several spatial databases covering the same geographic area with different scales, are derived from an original one. Although the integrity between original and derived multi-scale databases should be maintained, most of update mechanisms do not

*전기전자정보컴퓨터공학부, 부산대학교 jkwon@isel.cs.pusan.ac.kr
*전기전자정보컴퓨터공학부, 부산대학교 lik@pnu.edu
**지형정보협동과정, 부산대학교 hkang@pnu.edu

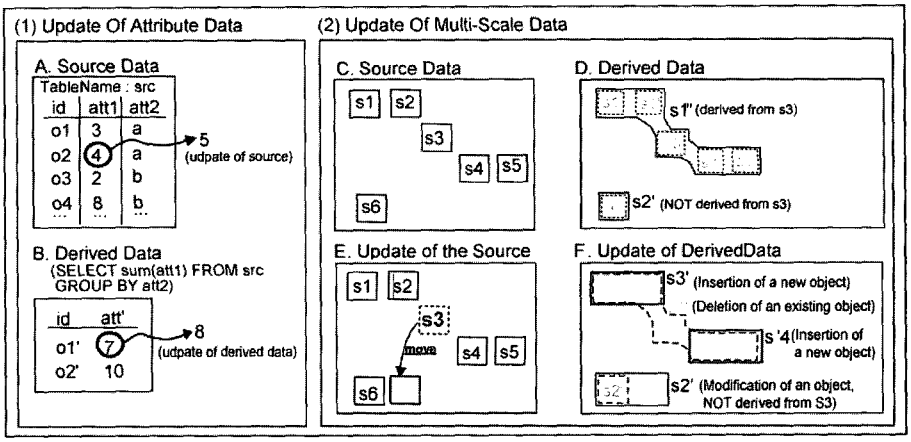
6respect it since the update mechanisms have assumed that the update of source objects propagates to objects directly derived from the source. In order to maintain the integrity of multi-scale databases during updates, we must propagate updates of sources to objects derived from both the updated source objects and other related objects. It is an important functional requirement of multi-scale database systems, which has not been supported by existing spatial database systems. In this paper, we propose a set of rules and algorithms for the update propagation and show a prototype developed on ArcGIS of ESRI. Our update mechanism provides with not only the consistency between multi-scale databases but also incremental updates.

Keywords : integrity of multi-scale spatial database, data update, consistency of derived data

1. 서 론

다중축척 데이터베이스는 동일한 지역(공간)을 다른 축척 및 공간형태로 표현한 공간데이터베이스들의 집합이다. 이 다중축척 데이터베이스는 이미 존재하는 원시 공간데이터베이스로부터 유도조건(derivation constraints)을 정의함으로써 자동적으로 생성될 수 있다. 이때 원시 데이터베이스와 유도 데이터베이스간의 일관성이란 유도조건이 항상 만족됨을 의미한다(Egenhofer 1997). 그런데 원시 데이터베이스의 갱신은 유도조건을 위배하는 원인이 될 수 있으므로 갱신관리를 해줄 필요가 있다. 이 논문은 원시 및 유도데이터베이스간의 일관성을 유지하기 위하여 유도조건이 보호될 수 있도록 갱신작업을 처리하는데 초점을 두고 있다.

유도된 데이터의 갱신에 관한 기존 연구들은 원시데이터의 갱신은 이로부터 직접 유도된 데이터만 파급된다고 가정해왔다. 예를 들어, [그림 1]의 (1)처럼 원시데이터 o_2의 갱신은 o_1'의 갱신원인이 되며, 다른 객체에는 영향을 주지 않는다. 반면, 다중축척 데이터베이스에서 원시 데이터베이스의 갱신은, 갱신된 원시객체뿐만 아니라 갱신되지 않은 원시객체로부터 유도된 객체들에 모두 영향을 미친다. 예를 들어 [그림 1]의 (2)에서처럼 S_3의 갱신은 S_3으로부터 유도된 S_1'뿐만 아니라 S_3으로부터 유도되지 않고 S_6로부터 유도된 S_2'에 영향을 미친다. 또, [그림 1] (2)-F처럼 새로운 객체(S_3', S_4')의 추가와, 존재하는 객체의 삭제(S_1')도 발생한다. 이러한 파급들이 다중축척 데이터베이스의 갱신처리에서 고려되어야 한다. 그러나 이 갱신문제는 다중축척 데이터베이스 연구분야에서 거의 연구되지 않은 분야이므로, 본 연구는 이 문제에 대한 해결방법을 제시하고자 한다.

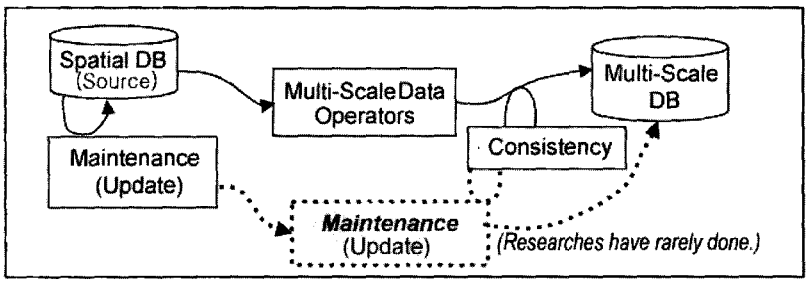


[그림 1] 현재 갱신방법과 다중축척 갱신방법의 비교

이 논문은 다음과 같이 구성되어 있다. 2장에서는 기존연구들을 간략히 기술한다. 3장에서는. 원시 데이터와 유도된 데이터간의 대응관계를 기술하는 다중축척 데이터 모델을 제시하고, 유도된 다중축척 데이터의 갱신 규칙 및 알고리즘을 기술한다 4장에서는 프로토타입을 소개하고 5장에서 이 논문을 결론 맺는다.

2. 기존 연구

이 논문은 다중축척 데이터베이스의 갱신에 대한 것으로, 원시데이터베이스의 갱신은 이로부터 유도된 데이터베이스를 갱신해야 하는 원인이 된다. 따라서 이 연구의 관련 연구 분야는 [그림 2]처럼 나눌 수 있다. 먼저 (a) 다중축척 데이터베이스의 유도를 위한 연산자(Multi-Scale Data Operators), (b) 다중축척 데이터베이스의 일관성(Consistency), (c) 공간데이터베이스의 갱신(Spatial DB Update), 그리고 (d) 유도된 다중축척 데이터베이스의 갱신파급(Maintenacne)이 있다. 이 분야들을 기준으로 몇몇 주목할만한 연구들을 간략히 소개한다.



[그림 2] 다중축척 데이터베이스 연구분야

다중축척 데이터베이스의 유도를 위한 연산자(Multi-Scale Data Operator)

기존의 대축척 데이터베이스로부터 새로운 소축척 데이터베이스는 지도일반화를 이용해서 유도될 수 있다. 지도일반화란 자세한 공간데이터를 축척에 적합하도록 단순화(simplify)시키는 것으로, 많은 연구들이 simplification 혹은 smoothing과 같은 연산자를 제안해왔다(McMaster 1992 and Muller 1995). (Davis 1999)는 이 연산자들은 지도 일반화, 기하 및 공간분석 연산자로 분류하였다. 더 복잡한 결과가 요구되는 경우, 새로운 축척의 데이터베이스는 공간데이터의 변형(distortion)뿐만 아니라 의미적 추상화(semantic abstraction)도 필요하다(Richardson 1994, Rigaux 1994, and Peng 1996). 의미적 추상화란 자세한 개념으로부터 보다 덜 자세한 개념을 유도해 내는 것으로, 아파트들로부터 생성된 아파트단지가 예가 된다.

다중축척 데이터베이스의 일관성 (Consistency of Multi-Scale DB)

데이터베이스의 일관성이란 데이터 모델에 논리적 모순이 없는 상태를 말한다(Egenhofer 1997). 이 논리적 모순은 제약조건을 정의함으로써 없앨 수 있는데, 만약 데이터들이 데이터모델에 기술된 제약조건을 만족시키지 않으면 일관성이 없는 것으로 간주된다. 이러한 관점에서 다중축척 데이터베이스의 일관성이 의미하는 것은 (i) 다중축척 데이터베이스 유도데 사용된 제약조건의 보장, (ii) 유도된 관계(relations)의 정확성이다. (Egenhofer 1993)는 다중축척에서의 일관성을 평가하는 방법을 제안하였으며, (Tryfona, 1994)는 다중축척 데이터베이스에 적합한 관계를 유도하는 방법들 제시하였다. 또, (Egenhofer 1994 and Ubeda 1997)는 위상관계의 일관성을 평가하는 방법들을 제안하였다.

공간데이터 갱신 (Update maintenance of spatial DB)

(Belussi 2000)는 공간 데이터베이스의위상적 무결성을 보장하는 관계(relation) 유도 시스템을 제안하였다. 이 시스템은 원시데이터베이스의 위상관계들이 갱신과정에서 변형되는 것을 금지하며, 갱신으로 인해 관계가 변할 경우 무결성이 보장되는 범위 내에서 변할수 있도록 관계를 유도한다. (Peled 1996 and 1998)은 지형도의 자동 갱신관점에서 공간데이터베이스의 갱신문제를 고려하였다.

Update of Derived Multi-scale Data

(Kidner 1994)는 다중축척 데이터베이스를 유지관리하는 프로토타입을 소개하였다. 이 시스템은 원시 데이터베이스에 기술된 규칙과 제약조건으로부터 생성된 규칙을 이용해서 다중축척 데이터베이스를 갱신한다. 이 연구는 (Jones 1996)에 의해서 개선되었다.

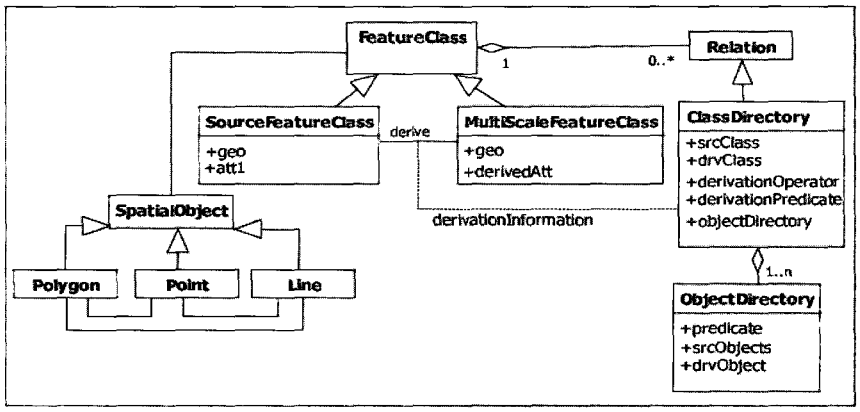
3. 다중축척 데이터베이스의 갱신

3.1 다중축척 데이터 모델

다중축척 데이터모델은 원시객체, 유도된 객체, 유도과정을 표현하는 클래스들로 구성되며, [그림 3]은 이들간의 관계의 다이어그램으로 보여준다. <표 1>은 이 논문에서 사용될 심볼과 이들의 의미를 정리한 것이다. 다이어그램에 기술된 각 클래스들의 의미는 다음과 같다.

MultiScaleFeatureClass(FC_{DRV})는 SourceFeature Class(FC_{SRC})으로부터 의미추상화를 통해유도된 클래스이다. 여기서 의미추상화란 객체들의 공통되는 속성을 이용하여 집합화하는 것을 말한다. 공간데이터베이스에서 이 속성으로 사용할 수 있는 정보는 인접(adjacency), 근거리(proximity), 동일한 속성값이다. [그림 2]에서 처럼 다중축척 데이터베이스는 다중축척 연산자(일반화 연산자)를 통해서 유도될 수 있으며, 이를 위해 많은 연산자들이 기존연구에 의해 제시되었다. 이 중에서 본 연구는 의미추상화의 관점에서 두 가지 연산자, GeoAggregation(집합)과 Classification(분류)을 고려한다. *GeoAggregation*은 근거리에 있는 객체들을 통합하여 새로운 객체를 생성하고, *Classification* combines 은 동일한 속성값을 가진 객체를 결합시켜 새로운 객체를 생성한다.

ClassDirectory(R_{CDRV})객 FC_{SRC}로부터 FC_{DRV}를 유도하는 관계를 표현한다. 이 클래스의 속성 *derivationPredicate*은 유도과정에서 사용된 제약조건을 기술하는데, 이 제약조건을 기술하는 방법에는 (i) 속성($FC_{SRC}.att$), (ii) 속성값($FC_{SRC}.att=v$), (iii) 사용자가 명시한 값(v)을 이용하는 방법이 있다. 속성($FC_{SRC}.att$) 제약조건의 예를 들면 FC_{SRC}가 행정경계이고, 동이름을 $FC_{SRC}.att$로 가지고 있을 때, 이 속성을 이용해서 동경계(FC_{DRV})를 유도($FC_{DRV} \leftarrow FC_{SRC}.att$)하는 것이다. 값(v) 제약조건의 예를 들면 1미터 이내의 주택들을 집단화하여 주택단지를 유도하는 것이 해당된다. 앞의 두 가지 방법은 f의 속성을 이용하지만, 값(v)을 이용하는 방법은 f의 공간



[그림 3] 다중축척 공간데이터 모델

<표 1> 심볼 및 의미

심 볼	의 미	기 술
FC	Feature Class	Features의 집합; FC_{SRC} : 원시 features 집합; FC_{DRV} :: FC_{SRC}로부터 유도된 feature들의 집합
f	feature집합 ∈ *FC*	f_i : i^{th} feature, *f.geo* : a spatial shape of a feature f_{SRC} : 원시feature의 집합, FC_{SRC}의 객체 f_{DRV}: 유도된 feature의 집합, FC_{DRV}의 객체
RCDRY *RODRY*	Class Directory Class Object Directory Class	FC_{SRC}와 FC_{DRV}간의 관계(유도대응관계) 집합 f_{SRC}와 f_{DRV}간의 관계(유도대응관계) 집합
att	속성(Attribute)	*FC.att* : *FC*의 속성; *f.attj* : feature *f* 의 j^{th} 번째 속성
dom(att)	도메인(Domain) of *att*	*att*의 속성값의 범위
v	속성값(Attribute value)	*f.a* = *v* : *v*는 *f.a* 의 속성값

정보를 이용하는 방법이다. FC_{SRC}가 갱신된 경우에도 이 제약조건들은 지켜져야 하므로 FC_{DRV}를 갱신해야 하는 것이다. 다은 장에서는 이 갱신을 위한 규칙들을 제시한다. 즉, 각각의 제약조건에 의해서 새로운 클래스 FC_{DRV}가 FC_{SRC}로부터 유도된 경우 FC_{SRC}의 갱신을 FC_{DRV}에 파급시키는데 필요한 규칙들을 제시하겠다.

3.2 갱신 파급 규칙

이 논문에서 고려할 갱신연산자는 삽입(insert), 삭제(deletE), 공간수정(geometry modify), 속성수정(attribute modify)이다. 갱신 규칙은 FC_{SRC}의 갱신내용을 FC_{DRV}에 파급시키기 위한 것이므로, 각 갱신연산자에 대해서 규칙을 제시할 필요가 있다. <표 2>에서 규칙 1은, FC_{SRC}로부터 속성제약조건에 의해 FC_{DRV}유도되었을 때, FC_{SRC}에 새로운 f_{ins}가 삽입(insert)된 경우 FC_{DRV}의 갱신을 위한 것이다. 각 갱신규칙은 부록 1의 알고리즘에 의해서 구현되었다. 값(*v*)을 이용한 제약조건은 *f*의 공간정보를 이용하는 방법이므로 속성이 수정된 경우를 고려할 필요가 없다.

<표 2> 갱신 파급을 위한 규칙 및 알고리즘

제약조건	갱신유형	규칙	알고리즘	제약조건	갱신유형	규칙	알고리즘
속성	삽입	규칙1	A1	속성값	삽입	규칙5	A1
	삭제	규칙2	A2		삭제	규칙6	A2
	공간정보수정	규칙3	A3		공간수정	규칙7	A3
	속성수정	규칙4	A1, A2		속성수정	규칙8	A1, A2
값	삽입	규칙9	A4	값	공간수정	규칙11	A4, A5
	삭제	규칙10	A5				

경우 1 : 속성 제약 조건

FC_{DRV}는 FC_{SRC}·att제약 조건에 의해 FC_{SRC}로부터 유도된 클래스이다.

[규칙1] 새로운 f_{ins}가 FC_{SRC}에 삽입(insert)된다. 이 삽입된 f_{ins}는, f_{ins}·att=$v1$, f_{drv}·att=$v1$인 경우, f_{drv}와 통합(merge)된다. f_{ins}·att=$v1$이 dom(f_{drv}·att)에 포함되지 않으면 FC_{DRV}에 새로 입력한다.

[규칙2] f_{drv}는 f_{src}로부터 유도되었을 때, f_{src}의 부분집합인 f_{del}가 FC_{SRC}로부터 삭제(delete)된다. 그러면 f_{drv}는 f_{src} f_{del}를 이용해서 수정된다. f_{src}=f_{del}이면 f_{drv}를 삭제한다.

[규칙3] f_{drv}는 f_{src}로부터 유도된다. 이때 f_{src}의 부분집합인 f_{chgeo}의 공간기하가 변경된다. 그러면 f_{drv}는 f_{chgeo}를 포함한 f_{src}로부터 다시 유도된다.

[규칙4] f_{drv}는 f_{src}로부터 제약조건 f_{src}·att을 통해 유도된다. 이때 f_{src}의 부분집합, f_{chatt}의 속성값이 $v_{original}$에서 v_{change}로 갱신되었다. 이 f_{chatt}은 f_{drv}·att-$v_{original}$인 f_{drv}와 f_{drv}·att=v_{change}인 f_{drv_other}의 갱신원인이다. v_{change}가 dom()인 경우에는, f_{drv}는 로 부터 다시 생성하고, 은 와 를 통합(merge)하여 수정한다. 가 dom(f_{src}·att)에 없는 경우에는, f_{drv}는 f_{src}·att=$v_{original}$로부터 다시 생성하고, 새로운 객체 f_{drv_new}를 f_{chatt}으로부터 생성하여 FC_{DRV}에 삽입(insert)한다.

경우 2 속성값 제약조건

FC_{DRV}는 FC_{SRC}로부터 제약조건 FC_{SRC}·att=v에 의해 유도되었다. 이때 v는 dom(FC_{SRC}·att)의 부분집합이다.

[규칙5] 속성값이 v인 새로운 객체 f_{ins}가 FC_{SRC}에 삽입(insert)된다. 그러면 인 f_{drv}는 f_{ins}와 통합(merge)을 통해 수정(f_{drv}←f_{drv}∪f_{ins})된다.

[규칙6] f_{drv}는 속성값이 v인 f_{src}로부터 유도된다. 이때 f_{src}의 부분집합인 f_{del}이 FC_{SRC}로부터 삭제(delete)된다. 그러면 att=v인 f_{drv}는 f_{src}-f_{del}인 객체들을 통합(merge)함으로서 수정(f_{drv}←f_{src}←f_{del})된다.

[규칙7] f_{drv}는 f_{SRC}·att=v인 f_{src}로부터 유도된다. 이때 f_{src}의 부분집합인 f_{chgeo}의 공간기하가 변경된다. 그러면 f_{drv}는 f_{chgeo}를 포함한 f_{src}로부터 다시 유도된다.

[규칙8] f_{drv}는 f_{src}로부터 제약조건 f_{src}·att=$v_{original}$을 통해 유도된다. 이때 f_{src}의 부분집합, f_{chatt}의 속성값이 $v_{original}$에서 v_{change}로 갱신되었다. 그러면 f_{drv}는 f_{src}·att=$v_{original}$로부터 다시 생성한다.

규칙 3 : 사용자가 명시한 값 제약조건

FC_{DRV}는 FC_{SRC}로부터 제약조건 v에 의해 유도되었다.

[규칙9] f_{drv}는 제약조건 v를 만족시키는 f_{src}로부터 유도된다. 이때 새로운 객체 f_{ins}가 FC_{SRC}에 삽입(insert)된다. f_{drv}와 f_{ins}의 관계가 제약조건 v를 만족시키면, f_{drv}와 f_{ins}를 통합(merge)하여 f_{drv}를 수정한다.

[규칙10] f_{drv}는 제약조건 v를 만족시키는 f_{src}로부터 유도된다. 이때 f_{src}의 부분집합인 f_{del}이 FC_{SRC}로부터 삭제(delete)된다. 그러면 f_{drv}는 f_{src}-f_{del}인 객체들을 통합(merge)함으로서 수정(f_{drv}←f_{src}-f_{del})된다.

[규칙11] f_{drv}는 제약조건 v를 만족시키는 f_{src}로부터 유도된다. 이때 공간정보가 수정

된 f_{src}의 부분집합을 f_{ch}라고 하면, 이 f_{ch}는 f_{drv}와 f_{src}를 제외한 원시객체 f_{src_other}로부터 유도된 f_{drv_other}의 갱신원인이 된다. 즉, f_{drv}는 $f_{src} - f_{ch}$로부터 새로 유도된다. f_{drv_other}는 f_{drv_other}와 f_{ch}가 제약조건 v를 만족시킬 경우, 이 f_{drv_other}와 f_{ch}를 통합(merge)함으로서 수정된다.

3.3 갱신 파급 규칙사용 예

이 예는 규칙 11을 적용한 예를 보여준다. [그림 4]는 객체수준에서의 다중축척 데이터 모델을 보여주고 있다.

주어진 데이터 집합: *BuildingBlock* 클래스는 *Building* 으로부터 *GeoAggregation* 연산자와 사용자가 명시한 *값*을 이용한 제약조건, *distance* < *50*에 의해서 유도되어 졌다.

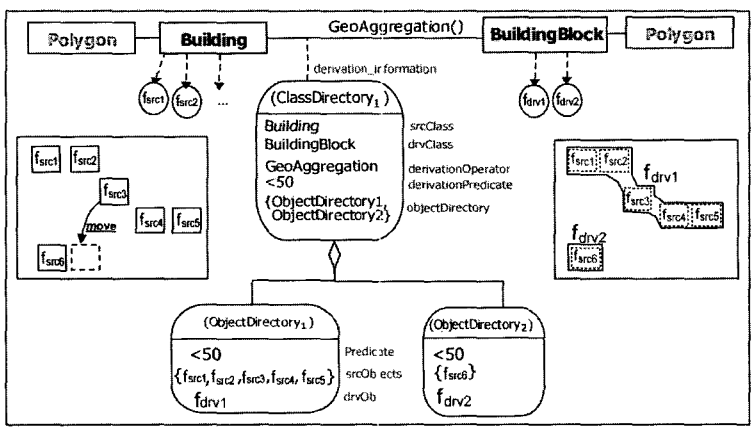
원시데이터 갱신: [그림 4]에서 *fsrc3* 이 *fsrc6*옆으로 이동 (change geometry)하였다.

갱신 파급: 제약조건의 유형이 값을 이용한 것이고, 갱신유형은 공간정보수정이므로, 규칙11 이 BuildingBlock 의 갱신을 위해 적용된다. 규칙11 은 알고리즘 4와 5를 수행한다. 예를들어 오른쪽 알고리즘 5의 line9는 GeoAggretaion() 연산자를 수행한다. 그 결과, 두개의 새로운 객체 f_{drv3}와 f_{drv4}가 FC_{DRV}에 삽입되고, f_{drv1}은 FC_{DRV}로부터 삭제된다.

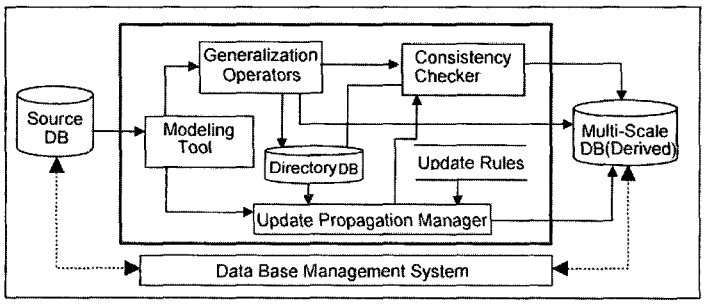
Algorithm 5
1: ListUpdateObj := {f_{src3}}
2: while(ListUpdateObj $\neq \phi$)
3: f_{del} := f_{src3}
4: f_{drv} := f_{drv1}
5:
6: ListSrcObjs := { f_{src1}, f_{src2}, f_{src3}, f_{src4}, f_{src5} }
7: ListObjs := { f_{src1}, f_{src2}, f_{src4}, f_{src5} }
8: ListUpdateObj := { ϕ }
9: f_{drv3} := { f_{src1}, f_{src2}} and f_{drv4} := { f_{src4}, f_{src5} }
//new objects are created.
10: delete f_{drv1} from FC_{DRV}
11: End while

4. 프로토타입 구현

본 논문에서는 ESRI에서 개발한 ArcGIS의 컴포넌트인 ArcObjects라는 공간 라이브러리를 이용하여 다중축척 데이터베이스 관리자의 프로토타입을 만들었다. 프



[그림 4] 갱신파급 규칙 11 사용 예



[그림 5] 다축척 데이터 관리자의 구조

로토타입의 주요 컴포넌트는 다음 [그림 5]와 같다.

Modeling Tool 컴포넌트는 사용자로 하여금 그래픽 상호작용을 통해 유도 과정을 수행할 수 있게 해준다. 여기서는 불러온 원시 클래스나 이 원시클래스로부터 유도된 새로운 클래스, 그리고 유도 과정에서 사용되는 일반화 연산자(generalization operators)들과 조건자(predicates)들을 명시할 수 있다. Generalization Operators컴포넌트는 6개의 일반화 연산자(Kang 2001)를 지원해준다. 이 연산자들은 원시 공간 데이터베이스로부터 다중축척 데이터베이스를 유도한다. Directory DB컴포넌트는 원시 데이터베이스와 다중축적 데이터베이스 간에 대응관계를 제공한다. [그림 3]에서 *ClassDirectory*클래스와 ObjectDirectory 클래스 정보가 여기에 저장된다. Consistency Checker(일관성 검사) 컴포넌트는 다중축척 데이터베이스 내의 관계가 올바른지 검사한다. Update Rule(갱신 규칙) 컴포넌트는 3장에서 제안한 갱신 파급 규칙을 제공해준다. 이 규칙들은 update propagation manager(갱신 파급 관리) 컴포넌트에 의해 적용되어 진다. Update Propagation Manager 컴포넌트는 directory DB컴포넌트를 참조하고 갱신 파급 규칙을 적용시킴으로써 원본 데이터베이스의 갱신을 유도된 다중축척 데이터베이스에 파급한다.

5. 결 론

본 논문에서는 다중축척 데이터베이스가 원본 데이터베이스로부터 유도되어 질 때 발생하는 다중축척 데이터베이스의 갱신 문제를 다룬다. 원본 데이터베이스와 유도된 다중축척 데이터베이스 간에 무결성을 보장해주기 위해 원본 데이터베이스에서 발생한 갱신을 갱신된 원본 객체와 다른 관계된 객체들로부터 유도된 객체들에게 파급해줘야 한다. 그럼에도 불구하고 대부분의 갱신 메커니즘은 원본 데이터베이스에서의 갱신은 바로 객체들에게 파급되어 진다고 가정하고 있기 때문에 위의 문제를 다루고 있지 않다. 그래서 본 논문에서는 다중축적 데이터베이스에서 갱신 파급을 조절해주기 위한 갱신 법칙과 알고리즘을 제시한다. 본 논문에서는 ESRI의 ArcObject를 통해 프로토타입을 개발하였다. 본 논문에서 제안한 갱신 메커니즘

은 데이터베이스 간의 일관성 및 다중축적 데이터베이스에서 발생하는 연속적인 갱신 문제를 해결하는데 기여할 것이다.

참고문헌

Belussi A., Negri M., and Pelagatti G., 2000: An Integrity Constraints Driven System for Updating Spatial Databases. *ACM Int. Workshop on Advances in Geographic Information Systems*, 121-128.

Chen I. and McLeod D., 1989: Derived Data Update in Semantic Database. *Proc. of the 5th International Conference on Very Large Data Bases*, 225-235.

Chen I., Hull R. and Mcleod D., 1995: An Execution Model for Limited Ambiguity Rules and Its Application to Derived Data Update. *ACM Transactions on Database Systems*, 20(4), 365-413.

Davis Jr. C. A. and Laender A. J. F., 1999: Multiple Representations in GIS: Materialization Through Map Generalization, Geometric, and Spatial analysis Operations. *ACM Int. Workshop on Advances in Geographic Information Systems*, 60-65.

Egenhofer M. and Sharma J., 1993: Assessing the Consistency of Complete and Incomplete Topological Information. *Geographical Systems 1(1)*, 47-68.

Egenhofer M., Clementini E. and Felice P., 1994: Evaluating Inconsistencies Among Multiple Representation. *Int. Symposium on Spatial Data Handling*, 901-920.

Egenhofer M. J., 1997: Consistency Revisited, *GeoInformatica*, 1(4), 323-325.

Davis Jr. C. A. and Laender A. H. F., 1999: Multiple Representation in GIS : Materialization Through Map Generalization, Geometric, and Spatial Analysis Operations. *ACM Int. Workshop on Advances in Geographic Information Systems*, 60-65.

Jones C. B., Kidner D. B., Luo L. Q., Bundy G. Ll. and Ware J. M., 1996: Database Design for a Multi-scale Spatial Information System, *Int. Journal of Geographical Information Systems*, 10(8), 901-920.

Kang H., Do S., and Li K., 2001: Model-Oriented Generalization Rules. *Proc.(in CD) ESRI User Conference*, San Diego, USA, July.

Kidner D. B. and Jones C. B., 1994: A Deductive Object-Oriented GIS for Handling Multiple Representations. *Int. Symposium on Spatial Data Handling*, 882-900.

McMaster R. and Shea K., 1992: Generalization in Digital Cartography. *The Association of American Geographers*, 1-134.

Muller J. C., Lagrange J. P. and Weibel R., 1995: GIS and Generalization:Methodology and Practice, *Taylor & Francis Inc.*, 1-252.

Peng W., and Tempfli K., 1996: An Object-Oriented Design for Automated Database Generalization". *Int. Symposium on Spatial Data Handling*, 199-213.

Peled A., 1996: Spatial Database Update-The key to effective automation. *Int. Archives of Photogrammetry and remote sensing*, XXXI(B4), 955-961.

Peled A., 1998: Toward Automatic Updating of the Israel National GIS Phase II. *Symposium on GIS Between Visions and Applications*, 32(4). 467-472.

Richardson D., 1994: Generalization of Spatial and Thematic data using inheritance and classification hierarchies. *Int. Symposium on Spatial Data Handling*, 957-972.

Rigaux P., and Scholl M., 1994: Multiple Representation Modeling and Querying. Geographic Information Systems: *Int. Workshop on Advanced Information Systems*, 59-69.

Tryfona N. and Egenhofer M. J., 1997: Consistency among Parts and Aggregates: A Computational Model. *Transactions in GIS*, 1(3), 189-206.

Ubeda Th., Egenhofer M. J., 1997: Topological Error Correcting in GIS. *In: Advances in Spatial Databases(SSD), Int. Symposium on Large Spatial Databases*, 283-297.

부록 1. 갱신 파급 알고리즘

Algorithm 1 : Insertion
Input : $FC_{SRC}, FC_{DRV}, R_{CDRY}, R_{ODRY}$
Output : FC_{DRV}, updated
Method :
1: Let *ListUpdateObj* be a list of inserted objects(f_{ins}) into FC_{SRC}
2: *constraint* $C \leftarrow R_{CDRY}.derivationPredicate$
3: **While** ($ListUpdateObj \neq \phi$) **Do**
4: Get f_{ins} from *ListUpdateObj*
5: *constraint* $Obj \leftarrow R_{ODRY}.derivationPredicate$ of f_{ins}
6: **If** (v of $f_{ins}.att \in dom(FC_{SRC}.att)$), where att = *constraint* C
7: Get f_{drv} which $R_{ODRY}.predicate = v$
8: $FC_{DRV} \leftarrow$ merge(f_{drv}, f_{ins})
7: **Else If** (v of $f_{ins}.att \notin dom(FC_{SRC}.att)$)
8: $FC_{DRV} \leftarrow$ Insert(f_{ins})
9: **End if**
10: **End while**

Algorithm 2 : Deletion
Input : $FC_{SRC}, FC_{DRV}, R_{CDRY}, R_{ODRY}$
Output : FC_{DRV}, updated
Method :
1: Let *ListUpdateObj* be a list of deleted objects(f_{del}) from FC_{SRC}
2: Let *ListSrcObjs* and *ListObjs* be a list of objects each.
3: Let f_{drv} be a subset of FC_{DRV}, where f_{drv} is derived from f_{del}
4: **While** ($ListUpdateObj \neq \phi$) **Do**
5: Get f_{del} from *ListUpdateObj*
6: *ListSrcObjs* $\leftarrow R_{ODRY}.srcObjects$ of f_{drv}
7: *ListObjs* $\leftarrow$ {*ListSrcObjs*} − {*ListUpdateObj*}
8: *ListUpdateObj* $\leftarrow$ {*ListUpdateObj*} − {*ListSrcObjs*}
9: **While** ($ListObjs \neq \phi$) **Do**
10: $FC_{DRV} \leftarrow$ union(*ListObjs*)
11: **End while**
12: **End while**

Algorithm 3 : Change Geometry
Input : $FC_{SRC}, FC_{DRV}, R_{CDRY}, R_{ODRY}$
Output : FC_{DRV}, updated
Method :
1: Let *ListUpdateObj* be a list of changed objects(f_{chgeo})
2: Let *ListSrcObjs* be a list of objects.
3: Let f_{drv} derived from f_{chgeo} be a subset of FC_{DRV}
4: **While** ($ListUpdateObj \neq \phi$) **Do**
5: Get f_{chgeo} from *ListUpdateObj*
6: *ListSrcObjs* $\leftarrow R_{ODRY}.srcObjects$ of f_{drv}
7: **While** ($ListSrcObjs \neq \phi$) **Do**
8: $FC_{DRV} \leftarrow$ union(*ListSrcObjs*)
9: **End while**
10 **End while**

Algorithm 4 : Insertion
Input : $FC_{SRC}, FC_{DRV}, R_{CDRY}, R_{ODRY}$
Output : FC_{DRV}, updated
Method :
1: Let *ListUpdateObj* be a list of objects(f_{chgeo}) in FC_{SRC} which geometry is changed.
2: Let f_{drv} be all objects of FC_{DRV}.
3: *constraint* $\leftarrow R_{CDRY}.derivationPredicate$
4: **While** ($ListUpdateObj \neq \phi$) **Do**
5: Get f_{del} from *ListUpdateObj*
6: $FC_{DRV} \leftarrow$ union($f_{chgeo} \cup f_{drv}$), where ($f_{chgeo} \cup f_{drv}$) satisfies with *constraint*
7: **End while**

Algorithm 5 : Deletion
Input : $FC_{SRC}, FC_{DRV}, R_{CDRY}, R_{ODRY}$
Output : FC_{DRV}, updated
Method :
1: Let *ListUpdateObj* be a list of objects(f_{del}) deleted from FC_{SRC}
2: Let *ListSrcObjs* and *ListObjs* be a list of objects each.
3: Let f_{drv} be a subset of FC_{DRV}, where f_{drv} is derived from f_{del}
4: **While** ($ListUpdateObj \neq \phi$) **Do**
5: Get f_{del} from *ListUpdateObj*
6: *ListSrcObjs* $\leftarrow R_{ODRY}.srcObjects$ of f_{drv}
7: *ListObjs* $\leftarrow$ {*ListSrcObjs*} − {*ListUpdateObj*}
8: *ListUpdateObj* $\leftarrow$ {*ListUpdateObj*} − {*ListSrcObjs*}
9: **call GeoAggretaion(** *ListObjs* **)**
10: delete f_{drv}
11: **End while**