지식기반형 전문가시스템을 이용한 CIM 데이타베이스의 통합†

박남규* · 김기동** · 박진우***

A Framework for the Intergration of CIM Databases using Knowledge-based Expert Systems†

Namkyu Park* · Ki Dong Kim** · Jin Woo Park***

ABSTRACT

One of the major issues in the implementation and maintenance of CIM databases is the sharing and exchange of information among the heterogeneous databases. This paper addresses some architectural aspects for integrating the heterogeneous multi-databases using knowledge-based expert systems. We propose a loosely integrated coupling system between databases and knowledge-based expert systems. Especially we suggest the architectural aspects of such a coupling methodology. We also present the structure and knowledge representation scheme for the proposed knowledge-based expert system. A prototype example is included to illustrate the framework and its mechanism for implementation.

1. 서 론

시장과 수요의 다양화에 따라 다품종 중·소량 생산 체제로 전환된 현대의 생산방식은 여러 면에서 매우 복잡한 양상을 나타내고 있다. 특히 급격하게 발달한 컴퓨터 기술은 생산시스템에도 큰 영향을 미쳤고, 이에 따라 컴퓨터 통합생산(CIM)이라는 새로운 생산방식이 등장하였다. CIM 시스템이란, 궁극적으로 수요예측에서 생산 및 판매까지의 제반 기업업무를 컴퓨터 정보 네트워크로 일체화시키는 것을 의미한다. 따라서 성능좋은 로보트나 공작기계만으로는 CIM 본래의 의미인 일체화를 달성하기가 어렵다. CIM 시스템 전체를 일체화 시켜주는 데는 "정보"라는 핵

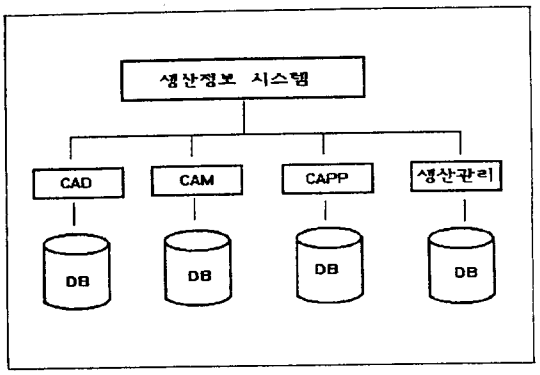
† 이 논문은 1992년도 재단법인 산학협동재단의 지원에 의하여 연구되었음.
* 서울대학교 산업공학과

심적인 매개체가 존재한다[3][7][16].

정보관리의 심장이라 할 수 있는 데이타베이스(이하 DB라 칭함)분야는 지금까지 많은 연구와 발전을 거듭하여 왔다. 그러나 생산시스템의 특성에서 기인하는 특수한 데이타 및 지식과 정보 등의 성질 때문에 기존의 DB만으로는 만족할 만한 정보관리를 하는 데 한계를 나타내고 있다[15]. 또한, 시간에 따라 끊임없이 변화하는 생산시스템 관련데이타의 특성때문에 정보자체의 양이 많아질 뿐만 아니라 미래의 정보에 대한 예측이 필요한 경우도 많아지게 된다. 따라서 CIM 시스템의 효율적인 운용을 위해서는, 방대한 양의 정보를 효과적으로 조직하고 처리할 수 있는 정보시스템이 필요하다. 제품의 생산과 관련된 정보들은 매우 복잡한 특성을 가지고 있기 때문에, 특별히 이와 같은 정보를 다루기 위한 생산정보시스템의 필요성이 커지고 있다. 본 연구에서는 생산정보시스템 중에서 생산현장관리 분야를 대상으로하여, 시스템 구축을 위해 필요한 이론적 토대를 개발함을 그 목적으로 한다. 특히, DB가 가지는 자료처리 능력과 전문가시스템이 가지고 있는 연역적 추론능력을 결합시킴으로써 제조시스템의 효율적 운영에 필요한 정보를 일관되게 유지하고 관리하도록 하였다.

2. 생산정보 시스템의 특성 및 필요사항

CIM은 제품수명주기상의 모든 활동사항을 포함하는 것이라고 생각할 수 있다. 따라서 신제품의 설계, 기획 및 생산 등에 관련된 모든 종류의 자료들이 CIM DB의 구성요소가 됨을 알 수 있다. 생산정보시스템이란 이처럼 다양하고 이질적인 특성을 가진 DB로부터 경영 및 제조공정상의 의사결정에 도움을 주는 정보를 창출해내는 시스템이다[13].



[그림 1] 생산정보 시스템의 구성도

생산정보 시스템은 [그림1]에서 보는 것처럼 다양한 종류의 DB를 이용하여 구축된다[12]. 이렇게 많은 종류의 DB를 데이타의 특성에 따라 분류해보면 크게 3가지로 나누어 볼 수 있다. 첫째는 CAD나 CAM 데이타처럼 복잡하고, 동적이며 제품의 외형상의 특징이나 물리적 특기사항 등을 수록하고 있는 design data이며, 둘째는 재무나 회계 및 행정상의 제어요인들과 같이 간단하고 정적인 business data이다. 셋째로 생산관리 시스템의 자재소요계획(MRP) DB처럼 생산계획이나 재고, 유지보수 등에 관련된 자료를 보관하는 planning data이다. Design data의 보관 및 검색을 위해서는 spatial data structure나 계층형 DB 등이 효율적이며 최근에는 객체지향 DB가 많이 이용되고 있다. 왜냐하면 어떤 제품에 대한 설계안은 부분품들의 모임으로 간주할 수 있으며, 각 부분품에 대한 설계안은 다른 부품의 설계시에 재사용이 가능해서, 각 설계안에 대한 데이타를 복합객체(complex object)로 생각할 수 있기 때문이다. 또 회계 DB와 같은 business data의 처리는 관계형 DB가 매우 효과적으로 사용되고 있다. Design data는 그래픽, 텍스트, 수

식표현 등의 데이타로 구성되며, 많은 경우 서로 다른 데이타 모형에 근거한 이종 (heterogeneous) DB로 구성되는 반면에 business data는 단순히 텍스트와 숫자 데이타 만으로 구성되는 간단한 형태이며 대부분이 동종(homogeneous) DB로 구성되고 있다. MRP DB와 같은 planning data는 이 두가지 형태의 특성을 모두 포함하고 있어서 이와 같은 제반 특성을 고려해 볼때 생산정보 시스템은 다음과 같은 특성을 가져야 할 것으로 여겨진다.

(1) 설계 및 생산업무는 매우 복잡하고 다량의 데이타 처리를 요구한다. 대개의 업무는 고도로 분산화된 환경하에서 병행적으로 이루어진다. 따라서 분산화된 각각의 작업을 통합환경내에서 관리하기 위해서는 강력하고 효율적인 client/server 구조가 필요하다. 즉, 이종 분산 DB의 효율적처리를 위한 시스템 구조가 필요하다.

(2) 설계 및 생산과 관련된 대부분의 업무는 반복되거나, 기존의 know-how를 이용하는 것이 많다. 따라서 기존의 데이타에 대한 재사용이나, 시간에 따라 변화하는 양상, 기존 자료의 추적 등이 쉽게 이루어질 수 있는 구조가 되어야 한다[10].

(3) 새로운 제품의 추가나, 새로운 제조공정의 도입등으로 인하여 design data나 planning data 등에 변화가 생겨, 근본적으로 DB의 구조가 바뀌어야 하는 경우에도, 기존의 DB 확장을 통하여 이런 상황을 지원할 수 있어야 한다. 규칙기반형 전문가 시스템을 이용하여 DB 시스템의 meta knowledge source를 구성하는 것도 한가지 방법이 될 수 있을 것이다.

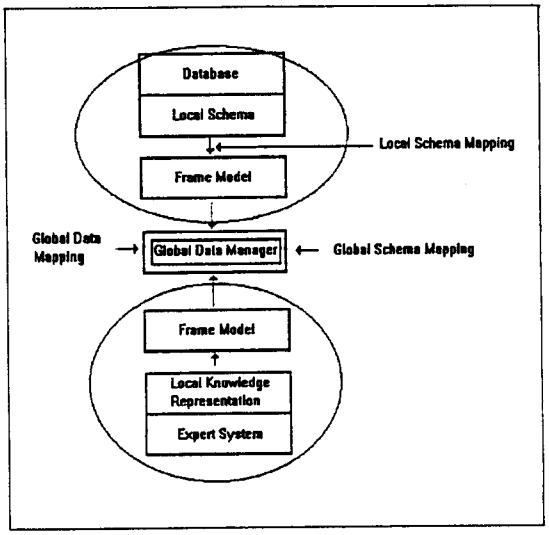
(4) 생산정보 시스템의 효율향상을 위하여 많은 양의 데이타 및 지식 등의 정보를 효율적으로 처리할 수 있어야 한다. 이종 DB간의 정보교환 및 검색을 위하여, 통합된 정보모형에 근거한 새로운 개념의 시스템 구조가 도입되어야 한다.

가장 이상적으로 생각해 본다면, 생산정보 시스템을 구성하는 모든 종류의 CIM DB는 동시에 설계, 구축, 운영 및 유지되어야 한다. 그러나 대부분의 기업이나 조직에서는 CIM의 도입이전에 이미 전산화 시스템을 갖추고 있었고, 시스템이 고도화 될수록 기존의 각 시스템은 '자율성을 가지는 정보의 섬'으로 변해가게 되었다[11]. 각 시스템은 고유의 DB 관리시스템을 가지며 다른 시스템과 분리되어 독립적으로 운영되기 때문에 실제로 각 시스템 간의 정보흐름은 극히 제한된 형태로 이루어지고 있는 것이 현실이다[13]. 이종 분산 환경으로 이루어진 생산정보 시스템의 특성으로 인하여, DB의 관점에서 볼때 몇가지 중요한 문제점이 발생하고 있다. 첫째는 데이타의 공유(data sharing)에 관한 문제이다. 데이타가 각각의 DB에 분산 저장되어 있고 각 DB의 데이타 모형이 서로 다르기 때문에, 서로 다른 DB에 있는 정보를 이용하여야 하는 경우에는 원활한 정보의 교환이 시스템 효율 향상의 관건이라 할 수 있다. MRP와 CAD, 생산계획 DB 간의 상호작용에서, 자재명세(bill of material)의 작성이나 대일정계획의 전개 등이 그 예가 될 수 있다. 둘째는 데이타공유 개념을 구현시키기 위한 데이타 교환에 관한 문제다. 이종 DB 간의 데이타 교환이 필요할 때, 많은 경우 데이타 변환을 통한 재입력 등과 같은 인위적 개입이 이루어지고 있어서 원활한 정보흐름이 차단되고 있는 실정이다. 따라서 이 부분을 자동화시킴으로써 정보시스템의 효율향상을 이룰 수가 있을 것으로 여겨진다. 셋째로는 앞에 언급한 두 가지 문제들로 인하여 각 DB 내에 저장되어 있는 정보들 중에서 서로 관련인

있는 정보들 사이에 정보불일치(inconsistency) 현상이 생겨 내용상의 일관성이 결여될 수가 있는 문제다. 이종 분산 DB를 동시에 여러개 이용하여 자료를 검색하고 수정 및 삭제작업을 하는 경우, 서로 연관되어 있는(dependent) DB 사이의 관계 및 의미를 추적하고 관리할 수 있는 시스템이 필요하다. 이와 같은 문제들을 해결할 수 있는 효율적 방법은 데이타 모형의 통합에 근거한 통합정보 시스템의 구축이라 할 수 있다. 다음 절에서 통합정보 시스템의 구축을 위한 데이타 모형의 통합에 대하여 자세히 알아 보기로 하겠다.

3. 이종 분산 CIM 데이타베이스의 통합방법

생산정보 시스템을 구성하고 있는 다양한 DB를 통합하는 방법으로서 3가지를 생각할 수가 있다[6][8]. 첫번째 방법은 통일된 하나의 데이타모형을 기반으로 새롭게 개발된 동종의 DB들을 이용하여 생산정보 시스템을 구성하는 것이다. 이 방법을 이용하면 완전하게 통합된 정보시스템 모형을 구축할 수 있는 장점이 있으나, 기존의 DB와, 이 위에서 운용되는 모든 응용 프로그램 및 각종 관련 시스템들을 모두 포기해야 하는 단점이 있다. 그러나 design data, business data, planning data 사이의 본질적인 특성 차이로 인하여 이 3종류의 데이타를 한 번에 관리해주는 통합 데이타 모형의 존재 가능성이 의문시 될 뿐만 아니라, 가능하다 하여도 그 효율이 매우 낮을 것으로 생각된다. 두번째로, 통합된 데이타 관리(administration) 시스템을 개발하여 기존의 이종 분산 DB를 통합하는 방법을 생각해 볼 수 있다. 이 방법은 기본적으로는 통합된 데이타 관리 시스템에 맞추어서 개개의 DB를 새로 설계하여야 하고, 이에 맞추어 해당 응용 프로그램 및 응용 시스템을 재구축해야 하기 때문에 많은 비용과 시간을 필요로 하고 있다. 그러나 기존의 이종 DB를 이용할 수 있다는 점에서는 중요한 의미를 주고 있다. 따라서 기존의 이종 DB를 연결하여 사용하되 DB의 재설계나 각종 응용 프로그램 및 응용 시스템의 재구축 없이 통합 정보 시스템을 구축하는 방법을 생각해 보는 것이 자연스러운 일이다. 이것이 세번째 방법이다.



[그림 2] Global Data Manager

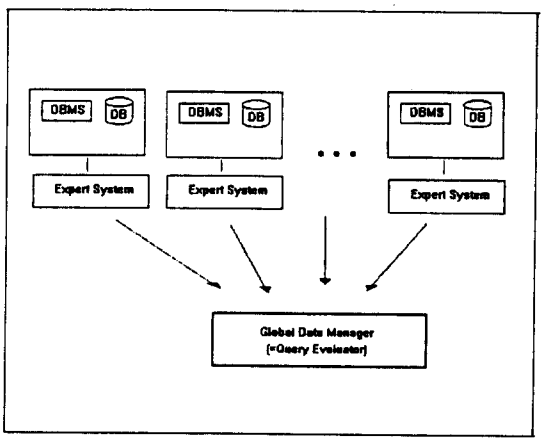
[그림2]와 같이 기존의 DB의 골격을 유지하면서 통합정보시스템을 구축할 수 있는 방법으로 global data manager 개념을 도입한 통합된 DB 접속을 생각해 볼 수 있다. Global data manager 개념이란 모든 DB가 어떤 일정한 형식을 통하여 자료를 교환하는 개념이다. 다시 말하면 local user 입장에서는 자기자신 중심의 DB(centralized DB)를 사용하는 것처럼 되고, global user 입장에서는 분산 DB(distributed DB)를 사용하는 것이 된다. 보다 근본적인 관점에서 파악해 본다

면 전통적인 file processing system과 DB 관리 시스템이 병존하면서부터 이종 분산 DB의 문제는 시작되었다고 볼 수 있다. 물론 DB를 연구하는 사람들 사이에서는 이종 분산 DB를 지원할 수 있는 DB 관리시스템 자체를 개발하려는 이론적인 시도와 소프트웨어 개발이 추진되고 있으나, 생산시스템이나 기타 응용시스템의 경우처럼 이미 DB 시스템이 구축되어 있는 곳에서는 그와 같은 DB 관리시스템의 개발이 현실적인 해결책이 되지 못한다[14]. 따라서 본 연구에서는 이종 분산 DB의 구성 자체에 대한 이론적인 접근이 아니라, 현실적인 대안으로서 DB 관리시스템 역할을 수행해줄 수 있는 개념적 대체 모형을 제시하였다[그림3].

4. 통합 생산정보시스템의 구조

4.1 전문가 시스템을 이용한 CIM DB의 구조

이미 존재하는 이종 분산 DB를 통합하는데는 통합환경의 특성상 2가지를 생각해 볼 수 있다. 만약 DB의 통합이 정적(static)인 상태에서 이루어진다면 간단한 문제가 된다. "Static"이라는 것은 기존 DB 구조나 schema에 대한 언급이 없이 데이타 값만이 교환됨을 의미한다. 그러나 DB의 통합이 동적(dynamic)인 상태에서 이루어져, 기존 DB의 구조나 schema가 시간에 따라 수시로 변경되는 경우는 데이타의 무결성 및 일관성 유지가 문제로 대두되기 때문에 DB 통합 작업이 매우 복잡하게 된다. CIM DB의 경우는 많은 경우 동적인 상태에서의 통합작업을 요한다. 이는 CIM 자체가 원래부터 유연하고 적응적인 시스템을 지향하고 있기 때문이다. 본 연구에서는 동적인 상태에서의 DB 통합을 데이타 값의 교환작업과 DB 구조 및 schema의 변경에 관련된 작업으로 구분하고 이 작업을 수행하기 위해 전문가 시스템을 이용하였다. [그림 3]에서 전문가 시스템을 이용하여 CIM DB를 통합하는 개략적 시스템 구조를 표현하였다.



[그림 3] CIM 데이타 베이스 통합을 위한 시스템 구조

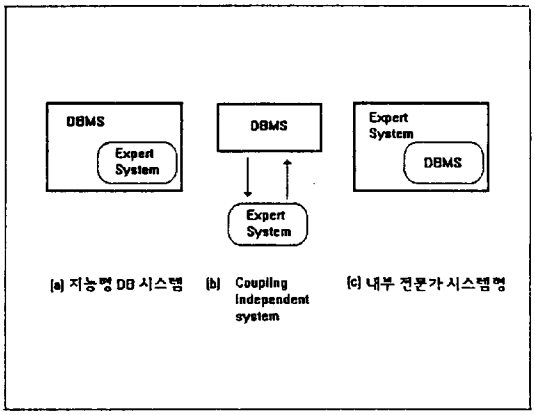
각 DB는 자기자신만의 고유한 데이타 모형과 DB 관리시스템을 가지고 있으며, 각 DB 시스템마다 거기에 해당하는 지식기반형 전문가 시스템을 갖고 있다. 각각의 DB 시스템마다 존재하는 전문가 시스템은 DB 사이의 데이타 교환이나 공유, 다른 DB로부터의 질의어 접수 및 해석, 자료검색을 위한 DB 관리시스템 보조 등의 역할을 수행한다. 또한 정보시스템을 구성하는 환경의 변화에 따라 DB를 조정해주는 역할을 하는 등 일종의 이종 분산 DB 관리시스템으로서의 기능을 수행한다. 각 DB에 전문가시스템을 구축하기 위해서는 DB의 구조와 데이타의 특성, 질의어의 종류, DB나 지식베이스의 수정을 요하는 외적 요인의 변화 등을 분석해야만 한다. 각 DB 시스템의 전문가 시스템은 다른 DB와는 무관하다. 즉, 각각의 DB 시스템과 전문가 시스템 사이에는 일대일의 대응관계가 존재한다. DB와 전문가 시스템

사이에 이렇게 일대일의 관계를 설정한 이유는, 이종 DB가 각각 고유의 데이타 모형을 가지며 이에 따라 DB의 설계나 응용프로그램의 작성, 데이타의 검색 등에 자기자신만의 독특한 방법을 사용하기 때문이다. [그림 3]의 시스템 구조에서 통합된 접속구조를 제공하는 global data manager는 일종의 query evaluator라 할 수 있다. Global data manager는 각 DB 사이의 데이타 불일치나 schema mapping 등의 역할을 수행한다. 또한 각각의 전문가 시스템이 서로 자료를 교환하는 데 있어서 조정자 역할을 수행하며 사용자로서는 규칙의 검색 형태로 이루어진 global query를 받아들여서 이것을 분해하고 각각의 DB에 할당해 주는 역할을 수행한다. 이 때 global data dictionary를 참고하여 데이타의 위치를 찾아준다. 따라서 사용자의 입장에서는 어떤 site에 어떤 정보가 있는지, 각 DB의 구조가 어떻게 되어있는지를 몰라도 간단하게 규칙(rule)의 형태로 질의어를 보내기만 하면 된다.

4.2 전문가 시스템의 구조 및 특징

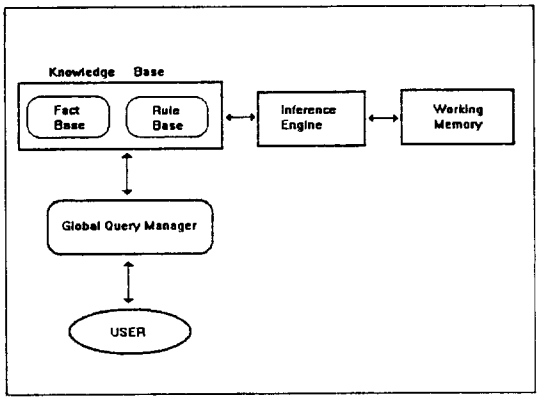
본 연구에서 제시한 시스템은, 4.1절에서 언급한 것처럼 이종 분산 DB와 전문가시스템이 1:1 대응 관계에 있고, 각 전문가 시스템은 모두 똑같은 구조를 가지고 있다. DB 관리시스템 기술과 인공지능 기술(특히 전문가시스템)이 결합되면 매우 방대한 양의 DB를 효율적으로 검색할 수 있다. 또한 DB 관리시스템 자체에 추론능력을 부여함으로서 DB 관리시스템의 기능과 능력을 확장시켜주고 있다.

DB 관리시스템과 전문가시스템 사이의 이와 같은 상호작용은 [그림4]에서 보는 것처럼 3가지 형태로 분류할 수 있다[4]. 본 연구에서 제시한 시스템은 이종 DB의 DB 관리시스템에는 관여하지 않은 채 전문가 시스템을 통해서만 전체 시스템의 데이타 관리를 하는 coupling 독립시스템 방식이다.



[그림 4] 데이타베이스와 전문가시스템의 결합모형

전문가 시스템은 [그림5]와 같은 구조로 이루어져 있으며 지식베이스는 이 시스템의 핵심적인 구성요소가 되고 있다.



[그림 5] DB-Coupled 전문가 시스템의 구조

지식베이스는 크게 3종류의 지식으로 구성되어 있다. 첫째, CIM DB 사이의 데이타 교환시에 발생하는 데이타 자체와 그 domain에 관한 지식,

둘째, DB의 개념적 모형이나 논리적 모형에 대한 구조를 표현하고 외부환경 변화시에 이러한 구조와 모형을 조정해 주는 지식 그리고 마지막으로 개개의 DB 관리시스템과 관련된 질의어(Query language)와 문법(syntax), 의미(semantics)와 관련된 지식 등이다. 본 연구에서는 이 세가지 중에서 앞의 두가지 지식을 분석하고 모형화하였다. 위 세가지의 지식 중에서 두번째 사항에 관련된 부분이 지식베이스의 핵심이자 가장 중요한 내용이라 할 수 있다. 또한 이종 DB 간의 원활한 데이타 교환 및 의미 불일치 등을 제거할 수 있는 근거가 되기 때문에 대단히 중요한 의미를 가진다고 볼 수 있다. 따라서 4. 3절에서 이에 관한 지식표현을 중심으로 하여 전문가시스템의 지식표현 및 규칙의 구성에 관한 내용을 전개하고자 한다. 첫번째 항에 관련된 지식은 대부분이 특정 영역의 데이타나, DB의 데이타사전에 등록된 것들이기 때문에 해당영역의 전문가나 사용자 또는 해당 DB로부터 쉽게 얻을 수 있다. 다만 본 연구에서는 지식표현 및 추론상의 효율을 생각하여 단순한 규칙대조(pattern matching)로 얻을 수 있는 특징을 가지는 데이타(또는 지식)와 생성규칙(production rule)을 통하여 얻을 수 있는 특징을 가지는 데이타로 구분을 하여 지식 베이스를 구축하였다. 세번째 항에 관련된 지식은 global query의 분해를 통하여 특정 DB 관리시스템에 할당된 subquery를 생성하는 일이나, 이 query의 수행을 위한 DB 검색, 문법 및 여러 종류의 query들에 대한 의미해석 등과 관련되어 있다. 최근의 DB 관리시스템들은 고급 질의어(high level query language)를 제공하고 있고, DB 관리시스템의 효율향상을 위하여 질의어 최적화를 많이 연구하고 있어서 위 세번째 항에 관련된 지식도 중요하게 생각된다. 그러나 본 연구의 목적은 DB 관리시스템의 효율향상이 아니라, 데이타 손실이나 불일치를 배제하면서 이종 DB간의 통합을 이룰 수 있는 방법을 찾는 것이기 때문에 세번째 항에 대한 분석 및 연구는 본 논문의 주제를 벗어나므로 더 이상 언급하지 않기로 한다.

4.3 지식의 표현 및 규칙의 구성

활동의 계획이나 제어, 제품의 설계, 생산, 검사 등과 같은 일들에 관계된 데이타는 대부분이 DB로 부터 연역적으로 도출된 정보들로서 여러 종류의 데이타들의 집합체로 구성되어 있다. 또한 이러한 정보들은 서로 연관되어 있어서 상호 영향을 준다. 따라서 DB 관리시스템에 연역적 능력을 부가해 준다면 이와 같은 정보를 보다 효율적으로 검색하고 관리할 수 있을 것이다. 뿐만 아니라 여러 개의 기능이 복합되어 도출되는 특성을 가진 데이타들도 존재한다. 이와 같은 데이타들의 효율적인 관리를 위해서 DB 관리시스템에 생성규칙을 갖춘 전문가시스템을 결합시켜서 DB 관리시스템의 기능을 확장시켜줄 필요가 있다. 본 연구에서는 지식의 표현을 위하여 규칙(rule)과 술어논리(predicate logic)를 사용하였다. 규칙은 이해하기가 쉽고 모듈화 되어 있어서 추가나 삭제 등 유지 보수 작업이 쉽고 동적인 상황에 맞도록 변경하기가 쉬워서 선택하였다. DB 구조의 표현은 실체(entity)와 관계(relationship)로 이루어지는데 관계의 표현은 술어논리를 이용하면 대단히 효과적으로 표현할 수 있기 때문에 선택하였다.

4.3.1 CIM 데이타베이스의 데이타와 domain에 관한 지식

본 연구에서는 DB의 검색과 자료교환시에 발생하는 데이타의 종류를, 어떤 사실을 표현하는

것과 사실의 조합에 의한 기능적 결과를 나타내는 것 등 2가지로 나누어서 각각 연역적 규칙(deductive rule)과 연산적 규칙(operative rule)으로 표현하였다.

① 데이타 정의

relation ⟨relation name⟩
⟨attribute domain list⟩
is_key ⟨relation name, key_attr_list⟩
(예) ***relation*** ⟨EMPLOYEE⟩
⟨E_no, E_name, SALARY⟩
relation ⟨PART⟩ ⟨Part_no, P_name, COLOR⟩

② 연역적 규칙의 정의

deduct-rule ⟨rule name⟩
⟨attribute_variable_list⟩
:→ ⟨$cond_1$⟩
:→ ……
:→ ⟨$cond_i$⟩
:→ ……
:→ ⟨$cond_n$⟩

attribute_variable_list의 형식은 $a_i=v_i$이고 ⟨$cond_i$⟩는 변수에 대한 조건식으로 conjunction이나 negation으로 형성된다.

여러개의 조건식으로 이루어지는 연역규칙은, 각 조건식 사이에 논리적 OR의 관계가 존재한다. 따라서 조건식들 중에서 한 개라도 성립(만족)이 되면 연역규칙도 성립하게 된다.

(예) ***deduct-rule*** ⟨Nation⟩
⟨Person_name, Country⟩
deduct_rule ⟨View⟩ ⟨Person_name, Age⟩
deduct_rule ⟨Xview⟩
⟨Person_name, Age, Country⟩
:→ ⟨View⟩ ⟨Person_name⟩,
⟨Nation⟩ ⟨Person_name, Country⟩,!.

위의 예는 개인의 이름과 나이 그리고 국적을 나타내는 DB를 규칙으로 표현한 것이다. 3번째 규칙에 나오는 !는 PROLOG의 cut라는 기능으로 어떤 기준을 만족시키는 상황을 찾아내면, 더 이상의 backtracking을 차단해주는 기능을 한다. 이렇게 함으로서 지식베이스를 소모적으로 탐색하는 비효율을 제거할 수 있기 때문이다.

③ 연산적 규칙의 정의

operative_rule [operation name]
with ⟨relation name⟩
⟨attribute_variable_list⟩
:→ ⟨$cond_1$⟩{set of operations for attribute 1}
:→ ……
:→ ⟨$cond_i$⟩{set of operations for attribute i}
:→ ……
:→ ⟨condn⟩{set of operations for attribute n}

우변에 있는 조건식이 하나라도 성립하면 ***operative_rule*** [operation name]도 성립한다. 단 조건식 i는 그 조건식 i에 대한 연산집합들이 모두 성립할 때 성립한다.

(예) ***relation*** ⟨EMPLOYEE⟩
⟨E_no, E_name, Salary⟩
relation ⟨***PROJECT***⟩
⟨***P_no, P_name, Leader***⟩
relation ⟨WORKS_ON⟩
⟨E_no, P_no, Role⟩

일 때 레코드 W를 ***relation*** ⟨WORKS_ON⟩에 삽입하거나 수정하고자 하는 경우를 생각해 보자. WORKS_ON Relation에는 2개의 foreign key가 있기 때문에 이 relation을 수정할 때는 refer-

ential integrity를 유지하기 위하여 다음과 같은 규칙을 만족시켜야 한다.

operative_rule [insert]
with ⟨EMPLOYEE⟩
⟨E_no, E_name, Salary⟩
:→ ⟨EMPLOYEE⟩
⟨E_no = WORKS_ON. E_no⟩
:→ ⟨PROJECT⟩
⟨P_no = WORKS_ON. P_no⟩

4.3.2. 데이타베이스의 모형 및 schema 통합에 관한 지식의 표현

DB의 개념적 모형으로 부터 E-R(entity relationship) 모형을 작성하면, 곧바로 DB를 설계한다. 따라서 DB를 구성하는 실체인 데이타나 그 관계가 변하면 DB의 구조에도 영향을 주게 된다. 그러므로 E-R 모형에 대한 정보를 유지하고 있어야 만이 향후에 DB의 구조나, 분산 DB 사이의 schema 통합을 이룰 수가 있게 된다. 본 논문에서는 술어논리를 이용하여 E-R 모형을 다음과 같이 표현하였다.

entity ⟨entity name⟩ ⟨attribute list⟩
relationship ⟨relationship name⟩
⟨entity1, entity2, mapping, attribute list⟩

E-R 모형은, 이미 성립되어 있는 사실을 기술하는 fact가 되기 때문에 fact database에 저장된다. 술어논리에 의하여 표현된 E-R 모형과, 앞의 4.3.1절에서 데이타 및 domain 정의에 사용된 규칙을 이용하면 DB의 구조를 쉽게 규칙으로 표현하여 지식베이스에 저장할 수 있게된다. 이런 식으로 표현되어 지식베이스에 저장된 DB의 논리적 구조도 이미 성립되어 있는 사실을 기술하는 fact가 된다. 다시 말하면 靜的(static)인 지식이다. 그러나, 앞절에서 이미 여러 차례 기술한 바와 같이 entity의 삭제나 추가, 혹은 entity 사이의 관계 변화와 같이 DB의 구조에 영향을 주는 외부적 변화가 생기게 되거나, 이종 DB 간에 자료교환이 이루어지는 경우에는 DB의 논리적 구조가 動的(dynamic)으로 변하게 되므로, 이미 저장되어 있는 fact를 위배하게 된다. 따라서 이와 같은 상황을 처리하기 위해서는 E-R 모형의 수정과 DB의 구조를 동시에 자동으로 수정해야 한다. 본 연구의 전문가시스템에서는 DB 구조의 수정을 위한 지식표현을 위해 다음과 같은 절차를 사용하였다. 예를 들어 part의 가공을 위한 제조공정이 미리 정해져 있고 기계는 여러 종류의 part를 가공할 수 있다고 할 때 이런 상황을 지식베이스에 표현하려면

entity ⟨MACHINE⟩ ⟨m/c_no, m/c_name,.⟩
entity ⟨PART⟩
⟨part_no, part_name,····, price⟩
relationship ⟨ROUTING⟩
⟨MACHINE, PART, 1:n, [ ]⟩

라는 fact를 먼저 정의한다.

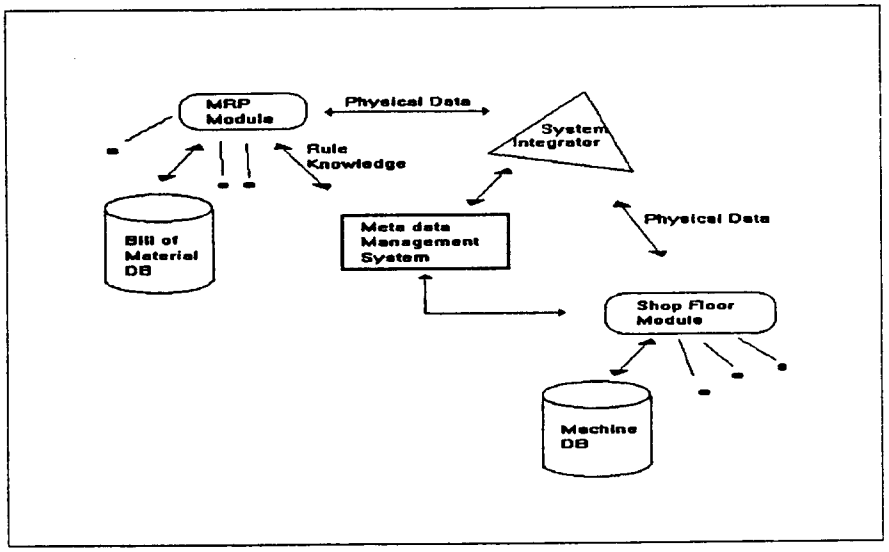
따라서 어느 기계에서 가공되는 part에 대한 정보를 알고 싶다면, ***relationship*** ⟨ROUTING⟩에서 [ ]로 표시된 attribute_list에, ***entity*** ⟨PART⟩의 모든 attribute_list와 entity ⟨MACHINE⟩의 attribute_list중 m/c_no를 두 entity사이의 link field로 포함시켜서 ***entity*** ⟨PART⟩를 수정하고 이에 따른 새로운 relation을 만들어야 한다. 새로운 relation은 ***relation*** ⟨STATUS⟩ ⟨p_no, ···, m/c_no⟩와 is_key ⟨STATUS⟩ ⟨p_no, m/c_no⟩가 된다. 이 fact가 DB의 논리적 구조에 대한 지식 베이스에 저장되게 된다.

그런데 만약 여기에 대체공정을 허용하게 되면, 위에 열거한 지식표현을 단순히 반복적으로 적용

함으로서는 이 상황을 표현할 수가 없게 된다. 왜냐하면 대체공정이 허용됨으로써 MACHINE과 PART사이의 mapping이 1:n 에서 m:n으로 바뀌기 때문이다. 이 사실은 E-R 모형의 수정이 필요함을 의미하고, 이것은 곧 DB의 논리적 구조가 변경됨을 의미한다. 이 경우에 위의 지식베이스는 ***relationship*** 〈ROUTING〉 〈MACHINE, PART, m:n, [m/c_no, p_no, proc_time]〉와 같이 수정되어야 한다. ***entity*** 〈PART〉에 link attribute로 들어와 있던 m/c_no는, (대체 공정을 나타내는 새로운 ***relation*** 〈STATUS〉에 의하여 표현이 가능하므로) 삭제되어야 한다. 따라서 ***entity*** PART 〈p_no, p_name,.〉가 되고, 이 사실은 곧 DB의 논리적 schema가 변경됨을 의미한다. 또한 새로운 "many:many"관계의 표현을 위하여 relation도 ***relation*** 〈STATUS〉 〈p_no, m/c_no, proc_time〉와 is_key 〈STATUS, p_no, m/c_no〉로 수정되어야 한다. 이와 같이 변경된 새로운 fact를 지식베이스에 첨가함으로써 외부조건이나 상황의 변화에 따른 DB의 구조 변경을 자동적으로 수행하게 된다. 이와 같은 논리적 절차를 그대로 적용함으로써 DB의 구조나 E-R 모형의 수정, 추가, 삭제 등을 정의할 수 있다.

5. 규칙처리에 대한 예제

본 연구에서는 [그림1]에 표시한 생산정보 시스템 중에서 자재소요계획과 관련된 부분에 대한 prototype을 구성하였다. [그림6]에서 자재소요계획과 관련된 부분은 Foxpro를 사용하였고 shop floor와 관련된 부분은 Excel과 같은 spreadsheet에 의한 file processing system을 사용하였고 지식베이스(meta data management system)의 구성은 PROLOG를 이용하였다. 현재 지식베이스의 구성은 전문가 시스템 개발도구인 CLIPS를 이용하여 재구축하고 있다. 또 실제로 데이타를 dump하는 부분인 System Integrator는 BASIC과 C언어로 작성하였다.



[그림 6] 데이타 처리를 위한 규칙의 실행

예를 들어 [그림7]과 같은 DB 시스템에서 part_no = P인 part를 삭제하는 경우를 보자.

part_no	p_name	color
		blue
P	a001	wht
Q	b001	red
R	c001	yell
S	d001	scar
T	e001	oran
U	f001	cyan

Parent.part_no	Child.part_no
a001	b001
b001	c001
b001	d001
c001	NONE
d001	e001
d001	f001
e001	NONE
f001	NONE

part_no	Qty
P	20
Q	20
R	120
S	40
T	40
U	80

[그림 7] Part의 추가 및 삭제에 대한 예제

먼저 해당 part가 stock에 유지되고 있는지를 나타내는 규칙이 정의 되어야 한다. 따라서

deduct_rule ⟨IN_STOCK⟩ ⟨part_no = P⟩
:→ *relation* ⟨PART⟩ ⟨part_no = P⟩,
relation ⟨STOCK⟩ ⟨part_no = P, Qty = Q⟩
& Q ⟩ 0.

로 정의할 수 있다. 위 규칙의 우변에서 ","는 logical AND를 의미한다.

마찬가지로 NOT_IN_STOCK 규칙은

deduct_rule ⟨NOT_IN_STOCK⟩
⟨part_no = P⟩
:→ *relation* ⟨PART⟩ ⟨part_no = P⟩
:→ *relation* ⟨STOCK⟩
⟨part_no = P, QTY = Q>&Q<1.

로 정의할 수 있다.

part_no = P인 part를 삭제하기 위해서는 지식베이스에서다음과 같은 규칙을 찾아 추론해야 한다.

operative_rule [delete]
with ⟨PART⟩ ⟨part_no = P⟩
:→ IN_STOCK ⟨part_no = P⟩
do {해당 part가 존재하여 지울수 없음}
:→ NOT_IN_STOCK ⟨part_no = P⟩
do { relation PART 에서 해당 part를 삭제}

좀더 복잡한 예를 들어보자. 자재명세(bill of material) 를 나타내는 relation 에 (parent) Part_no = PP, child Part_no = CP인 part를 추가하고자 하는 경우를 생각해보자. 먼저 두 part 사이의 parent, child 관계를 나타내는 규칙을 정의하자.

deduct_rule ⟨TREE⟩ ⟨parent. part_no =
PP, child. part_no = CP⟩
:→ ⟨BOM⟩ ⟨parent. part_no = PP, child.
part_no = CP⟩
:→ ⟨BOM⟩ ⟨parent. part_no = PP, child.
part_no = X⟩&

〈TREE〉 〈parent. part_no = X, child. part_no = CP〉

BOM의 parent/child 관계는 어떤 parent part가 또 다른 part의 child part가 되는 transitive closure의 특징을 가진다. 이것을 SQL같은 DB시스템 고급질의어를 사용하여 표현하기란 대단히 복잡하고 비효율적인 일이 된다. 만약 깊이가 n인 BOM 계층구조에 대하여 이와 같은 자기되부름(recursion)을 조회한다면 n+1 번의 nested loop가 형성된다. 위에서 정의한 연역적 규칙 〈TREE〉의 두번째 조건식은 자기되부름 특성을 나타내기 위한 것이다. 지식베이스는 PROLOG를 이용하여 구축하였는데 PROLOG에서는 cut(!)를 사용함으로서 어떤 상황에 도달하면 backtracking을 차단하도록 해준다. cut을 사용하면 이와 같은 자기되부름구조를 대단히 효율적으로 제어해줄 수 있다. part를 추가하기위해서는 다음과 같은 규칙을 지식 베이스에 저장하고 있어야 한다.

operative_rule [insert]
with 〈BOM〉 〈parent. part_no = PP, child. part_no=CP〉
:→ 〈TREE〉 〈parent. part_no = PP, child. part_no = CP〉
:→ ¬〈TREE〉 〈parent. part_no=PP, child. part-no = CP〉 &
¬〈PART〉 〈parent. part_no = PP〉
do { Insert
relation 〈PART〉 〈parent. part_no = PP〉 &
Add
relation 〈BOM〉〈parent. part_no =PP, child. part_no = CP〉}

이 규칙에서 첫번째 조건식은 새로운 part를 추가함으로써 BOM hierarchy 상에 cycle이 발생하는지를 확인하기 위한 것이다. 일단 cycle이 없는 것으로 판명되면 두번째 조건식에 의하여 *relation* 〈PART〉에 part에 대한 master가 기록되고나서 *relation* 〈BOM〉에 제품구성상의 구조가 기록된다. 위의 예에서 보듯이 연산적 규칙의 실행은, 연역적 규칙에 의해 제공되는 지식들을 사용함을 알 수 있다.

6. 결 론

생산정보 시스템은 이종의 분산 DB로 구성되어 있다. 이종의 각 DB들의 구조와 특성을 그대로 유지하면서 전체정보 시스템을 원활히 하는 것은 대단히 어려운 일이지만, 동시에 대단히 현실적인 문제다. 본 연구에서는 그 방법의 하나로 전문가 시스템의 연역 및 추론 능력과 DB의 자료관리 및 검색능력을 결합함으로써 생산정보 시스템의 효율을 향상시킬 수 있는 일대일 대응 구조를 채택하고, 생산 시스템 운영에 필요한 정보사이의 관계를 일관되게 유지하고 관리하도록 하였다. 자료의 검색 뿐만 아니라, 생산시스템을 구성하는 외적요인의 변화가 DB의 논리적구조에 변화를 초래하는 경우에도 데이타의 무결성 및 schema 통합을 유지할 수 있도록 하였다. 본 연구에서 다루지 않은 질의어의 처리를 위한 부분도 전체 정보시스템의 효율에 큰 영향을 미치는 부분이므로 향후 연구가 필요한 부분으로 여겨진다. 본 연구에서 제시한 방법은 이종 분산 DB에 대한 이론적 DB 관리시스템이 아니라, 현실적으로 이미 존재하고 있는 이종 DB에 대하여 기존의 DB와 응용시스템을 그대로 유지하면서, 시스템 통합 및 정보의 공유를 이룩할 수 있는 접속구조라 하겠다. 이 방법은 기존의 시스템을 그대로 수용할 수 있고 시간에 따라 변하는 상황에 유연하고 적응적으로 대처할 수 있는 장점이 있다.

참 고 문 헌

1. Bancilhon, F., C. Thanos, and D. Tsichritzis, 「*Advances in database technology－EDBT '90 : session 3, 4, International Conference on Extending database technology*」, Springer－Verlag, 1990. 12.
2. Barghouti, Naser S. and Gail E. Kaiser, "Modeling concurrency in rule－based development envirionments", *IEEE Expert*, 1990. 12.
3. Beach, Mark J., "A flexible manufacturing technical data management system", *AUTOFACT '90 Conference Proceedings, SME.*, 1990. 11.
4. Campbell, J. A. and J. Cuena, 「*Perspectives in artificial intelligence : volume* Ⅱ, *Part 3*」, John Wiley & Sons, 1989.
5. Held, James P. and John V. Carlis, "Conceptual data modeling of expert systems", *IEEE Expert*, 1989. 9.
6. Kaiser, Gail E., et. al., "Database support for knowledge－based engineering environments", *IEEE Expert*, 1988.
7. Kaula, R. and J. Chin, "A database approach towards flexible manufacturing : A conceptual framework", *Computer Ind. Eng.*, vol 24, no 2., 1993.
8. Navathe, S. B. and R. Elmasri, 「*Fundamentals of database systems*」, The Benjamin/Cummings Publishing Co., 1989.
9. Nof, Shimon Y., O. Etzion, and D. Dori, "Active coordination of a CIM multi－database system", *Research Memorandum*, No 93－17, School of Industrial Engineering, Purdue University, 1993. 9.
10. Perkinson, Richard C., 「*Data analysis : The key to database design*」, QED Information Sciences Inc., 1984.
11. Ranky, Paul G., 「*Manufacturing database management and Knowledge－based expert systems*」, CIMware limited, England, 1990.
12. Roussopoulos, Nick , et. al., "An architecture for high performance engineering information system", *IEEE Transc, Software Engineering*, Vol 17, No 1., 1991.
13. Scheer, A. －W. and Alexander Hars, "From CIM to enterprise－wide data modeling", *ICCIM '91 Conference Proceedings*, 1991.
14. Yee, Lester and Cheng Hsu, "A metadata system for information modeling and integration", *IEEE systems integration '90*, 1990.
15. 박진우, 박주석, "A Preliminary Study : Database technologies for computer integrated manufacturing", 「한국 데이타베이스 학회 '91 동계학술대회 논문집」, 1991. 12.
16. 매일경제신문, "클라이언트/서버 기술과 컴퓨터 혁명", 1993. 12. 8.
17. 박진우, 박남규, 김기동, "데이타베이스와 전문가 시스템의 결합에 의한 생산정보시스템 개발 (최종 보고서)", 서울대학교 산업공학과 / 재단법인 산학협동재단, 1993. 7.