Ultra-Structure Theory를 이용한 데이터베이스 설계

이성수*, 송은정**, 이영재***

1. 서론

컴퓨터 소프트웨어 개발은 전략적 경쟁 우위의 수단으로 기업들 사이에 끊임없이 시도되고 있다. 이러한 시스템 개발로 인해 기업 등의 모든 조직은 비용 절하, 워크 프로세스의 지속적인 향상, 고용자, 밴더, 고객들과의 빠르고 지속적인 정보의 연계성, 컴퓨터 기술로 인한 새로운 시장 진출 등을 기대하고 있다. 이러한 목적을 만족 시키기 위해 더욱 복잡한 소프트웨어 시스템이 요구되고 있으며 데이터베이스, 프로그램 언어, 새로운 기술이 추가되는 지원 도구, 객체 지향 기술 등의 새로운 개발 환경들이 조성되고 있다. 성공적인 소프트웨어 시스템은 초기 개발 능력에 의한 측정뿐 만 아니라 사용자 요구들의 지속적인 변경에 의해서도 만족되어야 한다. 기존의 전통적인 개발 방법에 의하여 개발된 소프트웨어 시스템에서 시스템이 간단하다면 사용자 요구사항이 변해도 쉽게 시스템을 변경할 수 있을 것이다. 또는 시스템이 복잡하더라도 사용자의 요구사항이 변하지 않는다면 시스템은 성공적으로 구축될 수 있을 것이다. 그러나 초기 사용자 요구 사항에 맞추어 개발한 소프트웨어 시스템은 납기 후 단지 몇 달 만에 구식이 되어버리고 사용자에 의해 버려지게 되는 경우는 더 이상 특별한 일이 아니다. 빠르게 변화하는 오늘의 환경에서 소프트웨어 시스템은 최소한의 개발 기간과 비용을 적용하기 때문에 변경 유지 관리의 필요성이 커지고 있는 현실이다. 이러한 문제점을 완화해야 하는 관점에서 근본적이고 시간이 흘러도 변하지 않는 소프트웨어 개발 가능성을 고려한 접근 방법, 즉 Object Role Modeling (Halpern, 1996), Semantic Object Modeling (Kroenke, 1995), Ultra-Structure Theory (Long, 1995; Shostako, 1996/8) 방법이 그것이다. 이러한 접근 방법들은 전통적인 개발 방법의 문제점을 극복하는데 강점들을 제공하고 있고 그 이상의 연구를 할 가치를 제공한다.

이 논문은 새로운 접근 방법중의 하나인 Ultra-Structure Theory를 제조회사의 영업 주문 프로세스에 적용하여 데이터 베이스 설계의 가능성을 연구하고 아울러 전통적 개발 방법과의 차이점을 살펴보는 것이다.

2. Ultra-Structure Theory(UST)

2.1 개요

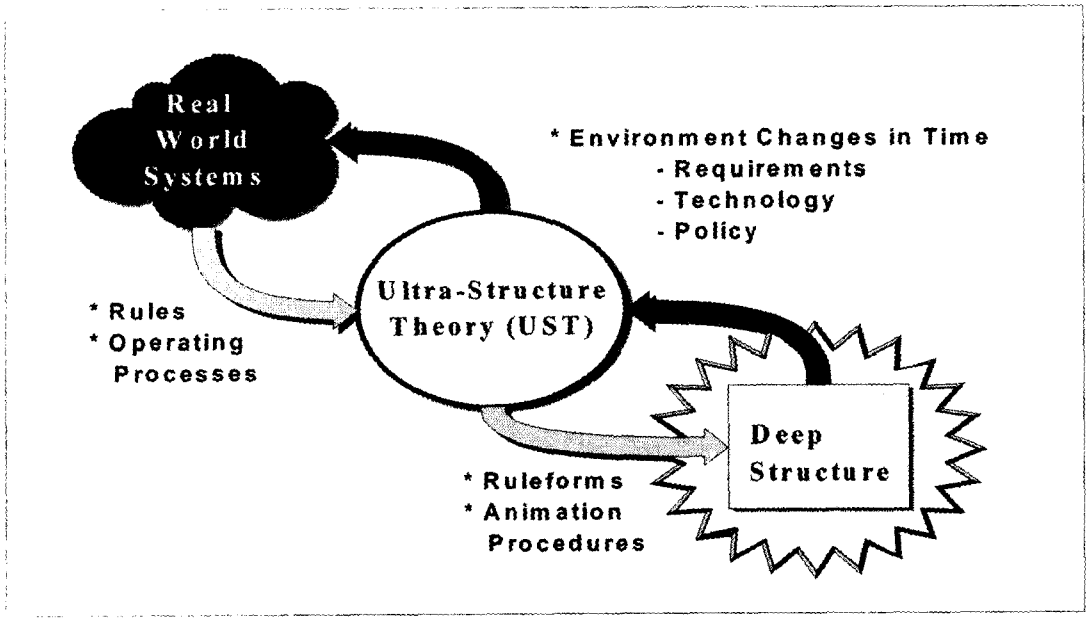
'Ultra-Structure'는 1995년 Long,J.G.과 Denning,D.E.에 의해 복잡하고 자주 변경되는 시스템의 구현을 위해 제시된 이론이다. UST는 물리적인 엔티티들을 일반화시켜 새로운 연관관계(relationship)를 보여주고 있다. UST 분석의 첫번째 접근방법은 프로세스를 이해하

* 동국대학교 대학원 정보관리학과 박사과정 (sslee@unimo.co.kr)

** 동국대학교 대학원 정보관리학과 석사과정 (silversong@empal.com)

*** 동국대학교 정보관리학과 교수 (yjlee@dongguk.edu)

고 매우 정확하게 Rule과 프로세스로 표현해야 하며 구현할 때에는 이런 표현을 단지 단편적인 프로세스가 아닌 유사한 시스템사이에서 나올 수 있는 가능한 모든 프로세스들을 고려해야 하는 것이다. 실제 실 세계 시스템에서 보여주는 복잡한 구조와 이벤트들은 내부 프로세스에 의한 부산물이다. 이러한 프로세스들은 많은 Rule의 실행 결과인데, 이 Rule들을 'Ruleform' 이라고 불리는 몇 개의 그룹으로 나눌 수 있다. (그림 1)은 시스템의 기능을 나타내는 Ruleform과 Animation Procedure로 구성되어 있는 UST의 Framework을 보여주고 있다.



(그림 1) UST 개념도

개념적으로 시스템 기능의 모든 면은 Operating rule과 Procedure에 나타나고 이들의 분석을 통해 Deep Structure(DS)를 개발하게 되는 것이다. DS는 Ruleform과 Control 로직이라고 부르는 Animated procedures로 구성된다. 실제 Rule과 Procedure는 데이터로써 저장된다. 현재와 환경에 따라 변화될 미래의 Operating rule 또는 Procedure는 Ruleforms에서 데이터를 조정함으로써 반영된다. 일단 DS가 시스템을 위해 적합하게 정해지면 Rule의 변경에도 불구하고 시스템의 근원적이고 안정적인 부분은 변하지 않는다고 할 수 있다. UST는 가장 기본적인 것에 초점을 맞추고 시간이 흘러도 변하지 않는 DS를 제안하고 있으며 복잡한 Rule의 개념을 활용하여 일반화 시킨 시스템 이론이다.

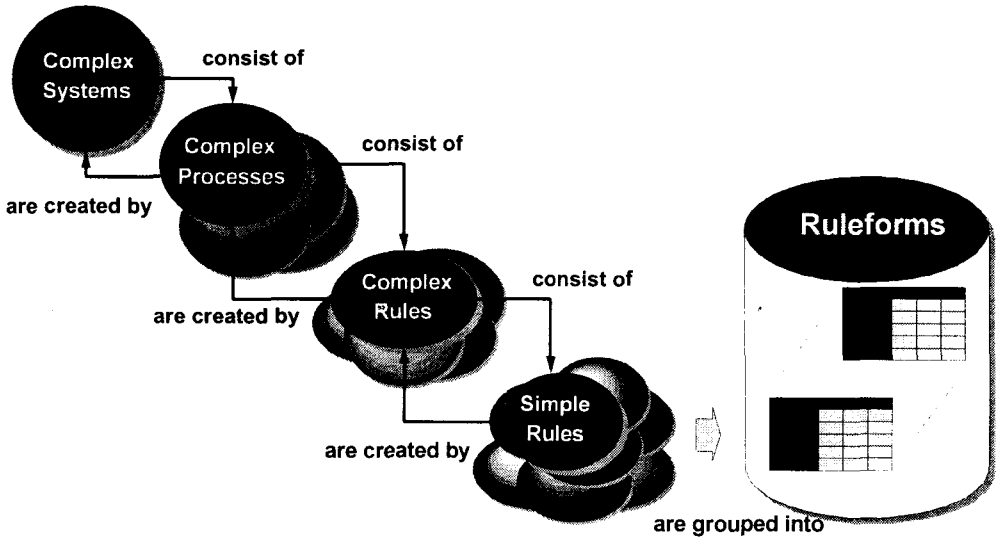
2.2 UST에서 두 가지 주요 가설

첫째가 Ruleform Hypothesis 이다. 복잡한 구조와 행동은 꼭 복잡하지는 않은 프로세스에 의해 발생된다. 그 프로세스는 Operating rule의 활성화를 나타낸다. Operating rule은 그 형식이 Ruleform에 의해 규정되는데, 몇 개의 종류로 나눌 수 있다. 그것은 일정하게 유

지되고 시스템의 DS를 구성한다.

둘째 가설은 CORE(Competency Rule Engine) Hypothesis 이다. 50개의 Ruleform이하로 구성된 Competency Rule Engine(COREs)이 있다. 이것은 유사한 시스템 사이에 발견되는 모든 Rule를 나타낸다. 그 한정적인 DS는 동일 업종에 속하는 여러 시스템에 대해서 변하지 않고 일정하다. 다양한 구조와 성격의 차이는 Competency rule의 차이에 의해 나타난다.

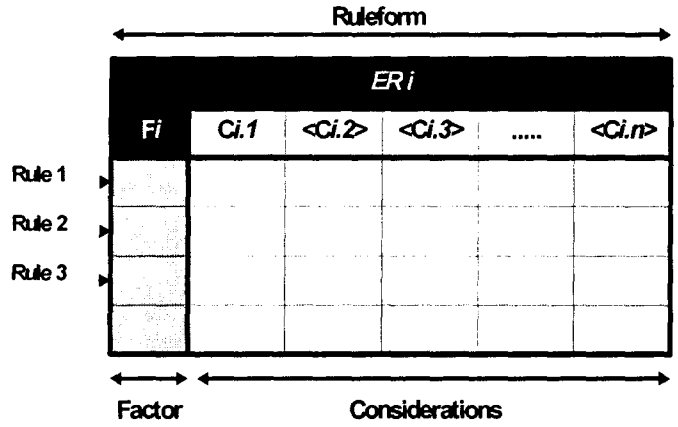
(그림 2)는 복잡한 어플리케이션 시스템에서 어떻게 Ruleforms을 분석하고 정의되고 있는지 보여준다. 복잡한 시스템은 Object와 Relationship 들에 의하여 정확하게 묘사될 수는 없다. 이러한 시스템 활동들은 Operating rule이나 Business rule에 의해 만들어진 복잡한 프로세스들의 실행 결과이다. Ruleform은 단순한 Rule 들의 집합으로 구성되고 Operating rule이나 Business rule을 편리하게 표현하기 위해 Ruleform을 구체화 한다. Ruleform은 Operating rule의 구조와 의미를 분석함으로써 구체화 시킬 수 있다. Rule은 시간이 지남에 따라 변할 수 있지만 Ruleform은 변하지 않고 일정하게 되는 것이다.



(그림 2) Ruleform Hypothesis

2.3 Ruleform 구조

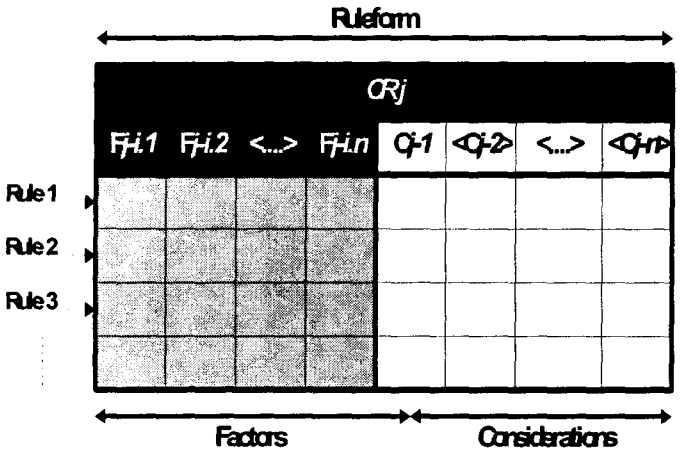
Existential rule은 엔티티들을 정의하는 데 사용된다. 그것은 시스템과 프로세스들의 설명을 정의하고 있다. 즉 시스템의 이벤트이나 프로세스의 행동을 설명하는데 사용된다. 일단 정의되면 엔티티의 특성들에 의해 분류된다. 유사한 엔티티들을 표현한 Existential rule은 한 개의 Existential Ruleform(ER) 구조로 그룹 지어진다. Existential Ruleform의 구조는 (그림 3)처럼 Factor와 Consideration들로 구성된다. 기능적인 관점으로 분류할때 Existential rule은 상위 레벨인 Compound Ruleform(CR)의 기초가 되는 단위이기도 한다.



- ER = Existential Ruleform
- i = ER #
- n = Integer (# of Considerations)
- F = Factors
- $C_{i.n}$ = Considerations for Existential Ruleform
- < Optional>

(그림 3) Existential Ruleform 구조(Oh & Scotti, 1999)

Existential rule의 조합은 "Compound Rule"이라고 불리어지고 (그림 4)에서 보여주는 Compound Ruleform(CR) 구조에 저장된다. Compound Ruleform은 Existential Ruleform에서 표현되는 두개나 그 이상의 다른 종류의 엔티티사이의 연관관계를 표현한다.



- CR = Compound Ruleform (CR can b one of the following ruleforms: NR Network Ruleform, AR: Authorizatio Ruleform, and MR: Meta Ruleform)
- i = ER #
- j = CR #
- n = Integer (# of Considerations)
- F = Factors
- $C_{j.n}$ = Considerations for Existential Ruleform
- < Optional>

(그림 4) Compound Ruleform 구조(Oh & Scotti, 1999)

Compound Ruleform에는 여러 종류의 Ruleform들이 있다. 첫째, 같은 종류의 Existential 엔티티 사이에 관계를 정의하는 Network Ruleform이다. 둘째, 다른 종류의 Existential 엔티티 사이에 관계를 정의하는 Authorization Ruleform이다. 셋째가 다른 Ruleforms 을 읽고 번역하고 평가하는 방법에 관한 Rule을 정의하는 Meta Ruleform이다. 이외에도 Protocol 그리고 Workorder Ruleform이 있다.

3. 데이터베이스 설계

한 제조회사의 영업 시스템에서 주문을 처리하는 프로세스에 Ultra-Structure Theory 를 적용하여 데이터베이스를 설계하는 과정은 다음과 같다.

3.1. 주문처리(Order Processing) 프로세스

주문처리 업무의 프로세스는 다음과 같이 나타낼 수 있다.

1) 처리자가 시스템에 로긴한다.
2) 시스템은 처리자의 부서와 처리 권한을 파악한다.
3) 시스템은 매출번호를 부여한다.
4) 처리자는 고객을 선택한다.
5) 시스템은 고객의 담당 부서와 로긴한 처리자의 부서를 비교하여 합당하지 않으면 에러를 낸다.
6) 시스템은 고객의 종류(대리점 또는 일반 고객에 대한 정보)를 알려준다.
7) 시스템은 대리점일 경우에는 신용도, 여신한도액을 파악한다.
8) 처리자는 세무 구분을 파악(일반, 간이, 영세율, 관납)하여 등록한다.
9) 처리자는 의뢰 제품의 재고수량을 등록한다
10) 시스템은 해당 제품의 재고 여부를 알려준다
11) 시스템은 제품의 수량과 대리점의 신용도, 입금방법(현금, 어음)에 따라 할인된 단가를 알려준다.
12) 시스템은 납품금액을 계산하고 영세율일 경우에는 부가세를 포함시키지 않는다
13) 시스템은 여신한도를 넘지 않는지 파악하고 여신한도가 넘으면 에러를 낸다
14) 처리자는 내용을 확인하고 등록처리 한다.
15) 처리자는 등록된 매출에 대한 거래명세서를 출력한다

전통적 개발 방법으로 접근하면 기본적으로 거래처, 사원, 단가, 재고, 품목마스터, 부서, 주문 마스터, 주문 상세 등의 관계테이블을 작성하고 주문 프로시져에 따라 각 테이블을 연결시켜 주문처리 프로세스를 프로그램화 할 것이다.

3.2. Existential Ruleform

3.2.1 Agency

전통적 개발 방법으로 구성한 관계형 테이블들 중에 거래처, 사원, 부서는 같은 성격을 가지고 있다고 판단하고 세 테이블을 [Agency] Ruleform으로 구성할 수 있다. 즉, <표 1>과 같이 표시 할 수 있는데 '114575', '100126'는 거래처 코드이며 Agency id에 Factor로 입력되며 거래처 명이 Description에 입력된다. '1900309', '2011279'는 사원 코드이며 성명이 Description에 입력되고 '210100', '210200'은 부서코드이며 부서명이 Description에 입력된다. 이들의 구분을 위해 Agency type의 Column을 추가 하였다.

전통적 개발 방법에 의한 3개의 테이블을 1개의 Ruleform으로 작성하였다.

<표 1> ER_1: Agency 구조

ER_1: Agency

F_1 : Agency ID	$C_{1.1}$: Description	$C_{1.2}$: Agency Type
114575	지티텔레콤	Customer
100126	하이테크 통신	Customer
1900309	홍길동	Person
2011279	김재욱	Person
210100	서울사무소	Depart
210200	부산사무소	Depart

3.2.2 Product

제품에 대한 정보는 [Product] Ruleform으로 작성하였는데 제품이나 반제품, 자재 등 모든 제품 코드가 레코드로 구성되게 된다. 각 제품에 대한 제품명은 Description에 입력된다. <표 2>에서 보여주듯 'RH020100P00'은 제품코드, 'RA000100'은 반제품 코드, '13300039'는 자재코드이다.

<표 2> ER_2: Product 구조

ER_2: Product

F_2 : Product ID	$C_{2.1}$: Description	$C_{2.2}$: Product Type
RH020100P00	휴대형무전기세트,220M,3W	제품
RH020200P00	휴대형무전기세트,220M,5W	제품
RB120100	BATT,PACK,700mAH	반제품
RA000100	BELT,SHOULDER	반제품
13300039	BATT,NICD	자재

3.2.3 Relationship

위와 같이 유사한 성격을 가진 데이터들을 구분하여 각각의 Existential Ruleform으로 생성할 수 있게 된다. 이러한 ER들 사이의 연관 관계를 나타내기 위해 [Relationship]의 Ruleform이 추가 되는데 <표 3>과 같이 생성된다. 전통적 개발 방법에서는 각 테이블의 Column으로 구성되는 내용과 유사하다. 예를 들어 FullName 이라는 Factor는 성명이나 상호명으로 사용되며, NID는 주민등록번호나 사업자 번호로 사용된다.

<표 3> ER_3: Relationship 구조

F_3 : Relation ID	$C_{3.1}$: Description
FullName	전체이름
NickName	약자
NID	주민등록번호/사업자 번호
ADDR	주소
TEL	전화번호
CHARGE_EMP	담당자
CHARGE_PRODUCT	담당품목
DEPART	부서코드
DEPART_NAME	부서명
ITEM_CD	품목코드
ITEM_NAME	품명
SPEC	규격
STAN_PRICE	표준단가
PART	구성품
CLASSIFY1	매출
CLASSIFY2	반품
PAYMENT_METHOD1	결재방법(현금)
PAYMENT_METHOD2	결재방법(외상)
TRADE_CLASS1	간판
TRADE_CLASS2	직판
CREDIT_AMT	여신한도금액
CURRENT_QTY	현재고 수량
SALES_CLASS1	현금매출
SALES_CLASS2	어음매출
TAX_CLASSIFY1	세무구분(일반)
TAX_CLASSIFY2	세무구분(간이)
TAX_CLASSIFY3	세무구분(영세율)
TAX_CLASSIFY4	세무구분(관납)

3.3 Compound Ruleform

이제 Existential rule의 연관관계를 나타내며 이들의 조합이 되는 Compound rule를 작성하게 된다. [Agency]과 [Relationship]의 연관관계를 나타낸 [AgencyRelationship]과 [Product]와 [Relationship]과의 조합인 [ProductRelationship] Network Ruleform을 생

성할 수 있다.

3.3.1 AgencyRelationship

[Agency]과 [Relationship]의 연관관계를 나타낸 것으로 <표 4>에서 보여주는데 ER_1의 Factor F_1인 Agency id 와 ER_3의 Relation id 의 Factor F_3 과의 관계를 보여주고 있다. 즉, '114575'의 거래처코드의 FullName은 '지티텔레콤'이고 사업자 번호는 '109-76-12357'임을 나타낸다. 거래처 뿐 만 아니라 사원과 부서에 대해서도 이와 같이 해석할 수 있다.

<표 4> CR_1: AgencyRelationship 구조

CR_1 : AgencyRelationship

$F_{1-1.1}$: AgencyID	$F_{1-3.2}$: Relation ID	C_{1-1} : Description
114575	FullName	지티텔레콤
114575	NickName	지티T.
114575	NID	109-76-12357
114575	ADDR	서울 서초구 방배동
114575	TEL	02-3479-5792
1900309	FullName	홍길동
1900309	NickName	홍길동
1900309	NID	721203-1593242
210100	FullName	서울사무소
210200	FullName	부산사무소

3.3.2 ProductRelationship

[Product]과 [Relationship]의 연관관계를 나타낸 것은 <표 5>에서 보여준다. ER_2의 Factor F_2인 Product id 와 ER_3의 Relation id 의 Factor F_3 과의 관계를 보여주고 있다. 'RH020100P00'의 품명은 '휴대형무전기세트,220M,3W,CLEAR' 이며 규격은 'PD-2000'이며 표준단가는 '360,000'원이다.

<표 5> CR_2: ProductRelationship 구조

CR_2 : ProductRelationship

$F_{2-2.1}$: Product ID	$F_{2-3.2}$:Relation ID	C_{2-1}:Description
RH020100P00	ITEM_NAME	휴대형무전기세트,220M,3W,CLEAR
RH020100P00	SPEC	PD-2000
RH020100P00	STAN_PRICE	360000
RH020200P00	ITEM_CD	RH020200P00

RH020200P00	ITEM_NAME	휴대형무전기세트,220M,5W,CLEAR
RA000100	ITEM_NAME	BELT,SHOULDER
RB120100	ITEM_NAME	BATT,PACK,700mAH
13300039	ITEM_NAME	BATT,NICD
13300039	SPEC	N-700AACL

3.3.3 AgencyNetwork

[Agency]과 [Agency]의 연관관계를 나타낸 것은 <표 6>에서 보여주고 있다. ER_1의 Factor F_1인 Agency id 와 ER_1의 Agency id 의 Factor F_1 과의 관계를 Relation ER_3가 연관관계를 나타내고 있다. '114575' 거래처는 담당부서가 '210100' 담당자는 '2011279'이며 '2100179' 사원의 부서는 '210100' 이다.

<표 6> CR_3: AgencyNetwork 구조

CR_3 : AgencyNetwork

$F_{3-1.1}$:Agency ID	$F_{3-3.2}$: Relation ID	$F_{3-1.3}$:Agency ID
114575	DEPART	210100
114575	CHARGE_EMP	2011279
2011279	DEPART	210100
1900309	DEPART	210100

3.3.4 ProductNetwork

[Product]과 [Product]의 연관관계를 나타낸 CR은 <표 6>에서 보여주고 있다. ER_2의 Factor F_2인 Product id 와 ER_2의 Product id 의 Factor F_2 과의 관계를 Relation ER_3가 연관관계를 나타내고 있고 관련된 조건이 추가 된다. 'RH020100P00' 이라는 제품은 '13300039'라는 자재가 3개, 'RA000100' 이라는 반제품이 1개 소요된다는 것을 나타내고 있다.

<표 6> CR_4: PorductNetwork 구조

CR_4 : ProductNetwork

$F_{4-2.1}$:Product ID	$F_{4-3.2}$: Relation ID	$F_{4-2.3}$:Product ID	C_{4-1} : QTY(소요량)
RH020100P00	PART	13300039	3
RH020100P00	PART	RA000100	1
RH020100P00	PART	RB120100	2
RB120100	PART	13300039	8

3.3.5 AgencyProduct

Authorization Ruleforms(AR)은 다른 종류의 ER 사이의 관계를 정의하고 있는데 어떤 매출 담당자에게 특정 담당 제품이 있다면 [Agency] 와 [Product] 사이에 [AgencyProduct] Compound Ruleform을 생성한다. 즉, '2011279'라는 사원의 담당 제품은 'RH020100P00'와 'RH020200P00' 이 되는 것이다. 또한 여기에 현재 재고를 관리 하고자 한다면 현재재고의 Relation id 와 연관시켜 전일재고, 금일입고, 금일출고, 재고수량 등에 대한 조건을 추가 할 수 있다. 부서 '210100'에 품목 'RH020100P00'의 현재고는 '2000'이라는 Rule을 동시에 적용시킬 수 있다.

<표 7> CR_5: AgencyProduct 구조

CR_5 : AgencyProduct

$F_{5-1.1}$: Agency ID	$F_{5-3.2}$: Relation ID	$F_{5-2.3}$: Product ID	C_{7-1} : S_Qty (전일재고)	C_{7-2} : I_Qty (금일입고)	C_{7-3} : O_Qty (금일출고)	C_{7-4} : Qty (재고수량)
2011279	CHARGE_PRODUCT	RH020100P00				
2011279	CHARGE_PRODUCT	RH020200P00				
1900309	CHARGE_PRODUCT	RH020100P00				
210100	CURRENT_QTY	RH020100P00	2000	2000	2000	2000
210100	CURRENT_QTY	RH020200P00	3500	0	500	3000
210200	CURRENT_QTY	RH020100P00	0	1400	0	1400

3.3.6 Credit_Rule

[Agency] 와 [Product] 사이에 여러 개의 연관관계가 성립되고 그 연관관계에 따른 조건들이 발생하면 <표 8>과 같이 나타낼 수 있는 [Credit_Rule] Compound Ruleform을 생성한다. 즉, 어떠한 거래처(여기서 (ANY)라고 표현)와 어떠한 제품(ANY)에 대해서 대리점(간판)으로 매출이 발생하고 외상 매출 이라면 금액 할인율 0% 이고 현금 매출이고 미수잔액이 40% 이상이면 5%, 미수잔액이 40% 이하면 추가로 3% 해주는 매출 Rule에 대하여 정의하고 있다. 만약 특정제품에 대하여서는 추가 할인 Rule이 적용되지 않는다면 그 내용도 포함하여야 한다.

<표 8> CR_6: AgencyProduct 구조

CR_6 : Credit_Rule

$F_{6-1.1}$: Agency ID	$F_{6-2.2}$: Product	$F_{6-3.3}$: Relation ID	$F_{6-3.4}$: Relation ID	C_{6-1} : 여신한도액	C_{6-2} : 미수잔액	C_{6-3}: 할인율	C_{6-4}: 할인율
(ANY)	(ANY)	TRADE_CLASS2 (간판)	PAYMENT_METHOD2 (외상)	CREDIT_AMT	40%	0%	0
(ANY)	(ANY)	TRADE_CLASS2 (간판)	PAYMENT_METHOD1 (현금)	CREDIT_AMT	41%	5%	0%
(ANY)	(ANY)	TRADE_CLASS2 (간판)	PAYMENT_METHOD1 (현금)	CREDIT_AMT	40%	5%	3%
(ANY)	(ANY)	TRADE_CLASS1 (직판)	PAYMENT_METHOD1 (현금)	CREDIT_AMT	40%	0	0
(ANY)	Y1019999	(ANY)	(ANY)	CREDIT_AMT	40%	0%	0

이밖에도 [Customer_Rule], [Tax_Rule], [Serial_Rule] [Discount_Rule]등이 만들어 질 수 있고 시스템의 프로세스와 로직에 따라 연관된 Rule들이 생성 될 수 있다. 초기에는 이러한 Rule들을 찾는 것이 쉽지 않지만 여러 번 반복하여 생각하면 쉽게 적용될 수 있을 것이다.

3.3.7 Sale_Workorder

실제 고객이 제품에 대한 주문을 하면 별첨과 같은 매출 Ruleform이 생성된다. 이것은 Working Order에 대한 실제 데이터가 저장되는 것으로 각 Column에 대한 정보는 ER에 근거하여 저장되는 것이다. 예를 들어 수불부서에 대한 내용은 ER_1 의 Agency id의 부서코드('210100')가 입력 되고 담당자도 Agency에 정의된 사번('1900309')이 입력 된다. 제품에 대한 내용을 다룰 때는 ER_2 의 Product id를 적용하고 납품, 현금매출 등에 관련된 내용은 ER_3 의 Relation id로 적용한다.

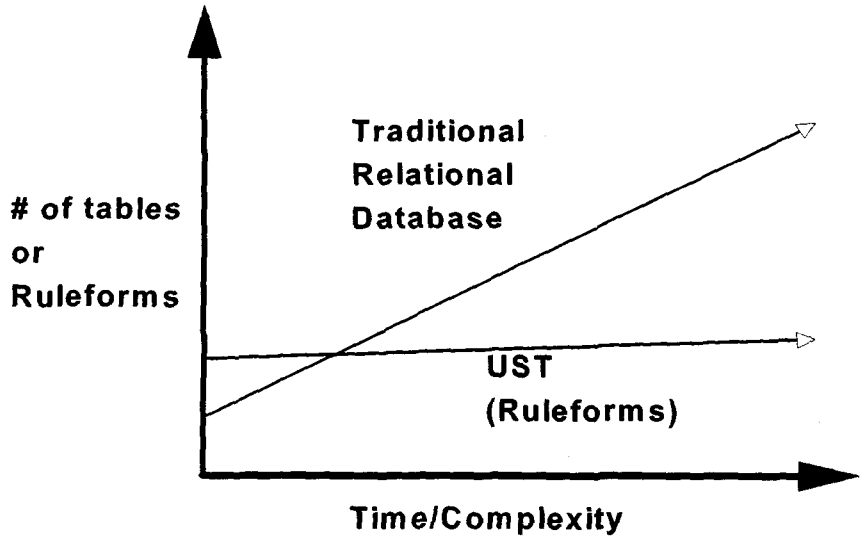
또한 프로그램에서 데이터를 정의된 CR에 있는 적합한 Rule을 적용하여서 처리하여 데이터베이스에 저장되며 아울러 각종 조회, 출력물을 생성할 때에도 CR에서 정의된 Rule이 적용된다.

4. 연구 결과

시스템을 개발 후 거래처에 대한 새로운 정보 추가를 원하든지 사원에 대한 관리항목이 늘어나게 된다든지 등에 대한 사용자 요구 사항이 변하는 경우가 발생하게 된다. 이때 전통적 개발 방법에 의하면 Column을 추가 해야 하며 관련된 모든 프로그램을 수정해야만 한다. 그러나 UST에서는 관련 ER에 관리를 원하는 항목을 레코드로 추가하기만 하면 된다.

더불어 추가된 항목과 기존 혹은 새로운 항목과의 연관관계가 발생한다면 CR 에 레코드로 추가하기만 하면 된다. 즉 Ruleform의 숫자를 증가 시킬 필요없이 변경될 수 있음을 의미한다. 정의된 ER의 주요관점은 유사한 엔티티/묘사/역할 등에 대한 정보와 데이터의 저장소가 되는 것이다. 이러한 ER의 연관성을 나타낸 것이 CR 이 되는 것이다. 개발 초기에는 프로세스에 따라 여러 개의 ER 과 CR이 만들어 질 것이다. 분석과 설계과정에서 사용자 참여로 여러 번의 과정을 거쳐 Ruleform들은 정리가 된다.

(그림 5)에서 보여주는 내용은 Long, J. G.의 연구결과(1995)를 보여주고 있다. 전통적인 데이터베이스 설계 접근 방법에서는 초기에는 테이블 수가 적지만 시간이 흐르고 시스템이 복잡해 질수록 테이블의 수는 현저히 증가하게 되며 따라서 관련된 프로그램은 모두 수정되어야 할 것이다. UST에 의해 개발된 시스템은 초기에 많은 변화 가능성을 내포해야 하기 때문에 많은 Ruleform이 생성되어야 할 것이다. 그러나 시간이 흘러감에 따라 거의 일정한 Ruleform의 수를 가질 수 있게 되는 것이다.



(그림 5) 예상되는 테이블 증가 비교

향후 연구는 UST를 바탕으로 생성된 Ruleform들을 가지고 실행 프로그램을 만들어 기존에 만들어진 시스템과 성능(performance)을 비교하며 장/단점을 발견하고 Usability 가능성을 찾아보는 것이다.

5. 결론

이 연구에서 UST에 기초하여 전통적인 데이터베이스 시스템과 비교하여 데이터베이스를 설계하였다. 새로운 UST를 기본으로 하는 시스템의 경우에는 시스템의 요구 변경 사항을 쉽게 반영 할 수 있을 뿐만 아니라 또한 이러한 시스템은 뛰어난 확장성과 탄력성을 보여주

고 있다. 이 연구는 Ultra-Structure 이론의 Ruleform Hupothesis를 지원하고 시스템의 Deep Structure를 구체화 하여 설계하였다. Deep Structure는 전통적인 데이터베이스 시스템의 이익보다 훨씬 많은 잠재이익을 제공할 수 있다는 것을 보여주었다. 이러한 UST를 기본으로 개발된 시스템의 특성과 이익에 대한 연구는 소프트웨어 시스템 개발 방법론에 혁명을 불러일으킬 것이다.

참고문헌

Conger, S., *The New Software Engineering*, International Thomson Publishing, 1994

Davis, A.M., *Software Requirements: Objects, Functions, & States (Revision)*, Prentice Hall, 1993

Halpin, T., *Conceptual Schema and Relational Database Design (2nd edition)*, Prentice Hall, 1995

Hunger, J.W., *Engineering the System Solution: A practical Guide to developing Systems*, Prentice Hall, 1995

Kroenke, D., *Database Processing: Fundamentals, Design and Implementation*, Prentice Hall, 1995

Long, J. G. and Denning, D.E., *Ultra-Structure: A design Theory for Complex Systems and Processes*, Communications of the ACM, January 1995

Pressman, S.L., *Software Engineering: a Practitioner's Approach (3rd edition)*, McGraw-Hill, 1992

Rechtin, E., *Systems Architecting: Creating & Building Complex Systems*, Prentice Hall, 1991

Shostko, A., *Design of an Automatic Course Scheduling System Using Ultra-Structure*, 1996

Shostko, A., *Information and Psychology*, WESS, 1998

Oh, Youngsuck and Scotti, Richard, *"Analysis and Design of a Database using Ultra-Structure Theory(UST) - Conversion of a Traditional System to One Based on UST"*, *Proceeding of the 20th Annual Conference*, American Society for Engineering Management(1999)

<별 첨>

Ruleform : Sales_Workorder

COLUMN	SALES_NO (매출번호)	SEQ (순번)	Relation ID 1 (납품)	Relation ID 2 (현금매출)
DATA	41772	1	CLASSIFY 1	SALES_CLASS1

ORDER_NO (수주번호)	Agency ID1 (수불부서코드)	Agency ID2 (담당자 사번)	SALES_DATE (납품일자)	Agency ID3 (거래처 코드)
263210	210100	1900309	2002.07.03	114575

Relation ID 3 (결재방법(현금))	RECEIPT_DATE (수금예정일)	CONTRACTOR_NO (계약번호)	CONTRACTOR_DATE (계약일자)	Agency ID4 (매출부서)
PAYMENT_METHOD1	2002.07.03	U02-21-1100	2002.06.30	210100

Relation ID 5 (세무구분(일반))	Relation ID 6 (간판)	VAT (부가세)	Product ID (품목코드)	QTY(수량)
TAX_CLASSIFY1	TRADE_CLASS2	465600	RH142200P00	20

PRICE(단가)	AMOUNT(금액)	SERIAL_FROM	SERIAL_TO
204300	4086000	3232	3251