**1. 논리적인 데이터베이스 설계**

**-- 논리적인 데이터베이스 설계는 다음이 정의되어야 한다.**

**● 테이블 및 해당 이름(entity(개체)라 불린다).**

**● 각 테이블의 컬럼명(attribute라 불린다).**

**● 유일 값 요구, null 허용여부, 컬럼에 저장될 자료형 컬럼의 특성**

**● 각 테이블의 기본 키(primary key)**

**테이블의 컬럼을 유일하게 식별할 수 있는 값들을 저장하는 컬럼 (또는 컬럼 집합)이다. 테이블에 다른 유일한 컬럼이 존재할 수 있지만 primary key 만이 행들을 추출하기 위한 유일한 access key로서 식별된다. 각 테이블은 하나의 primary key만을 가질 수 있다. primary key가 엄격히 요구되는 것은 아니지만 항상 하나의primary key는 정의되어야 한다.**

**● 테이블간의 관계**

**어떤 테이블의 행들은 다른 테이블의 하나 이상의 행에 종속적이다.**

**이러한 테이블 사이의 종속성을 관계(relationship)라 한다. 관계를**

**정의하기 위해서는 다른 테이블의 primary key를 참조하는 이러한**

**컬럼을 참조 키(foreign key)라 한다.**

**예를 들어, 각 order(주문)는 customer(고객)에 의해 정해지므로**

**Orders테이블의 각 행은 Customers 테이블에 종속적이다.**

**이것이 orders 테이블과 customers 테이블의 관계이다.**

**orders 테이블은 customers 테이블의 각각의 행(각각의 고객)을**

**참조하는 값들을 가지는 컬럼을 가져야 한다. orders 테이블의 행들은**

**한명의 고객(customer)만을 참조하는 것이 보장되어야 하므로 관계는**

**customers 테이블의 primary key에 기초하여야 한다.**

**Customers 테이블의 primary key를 참조하는 Orders 테이블의 컬럼을**

**참조 키(foreign key)라 부른다.**

**(고객들 중에는 "동명이인" 이 존재할 수 있다. 주문한 물건은 한**

**사람인데, 여러 "동명이인"으로 인식한다면 곤란하다. 그러므로 고객의**

**유일한 "회원번호"를 체크 하는 것이 현명하다.)**

**2. 관계의 유형**

**-- 테이블 사이에는 3가지 유형의 관계가 가능하다.**

**● 일대 일(One-to-One)**

**기본 테이블의 각각의 행은 참조 테이블의 하나의 행과만 관계되어진다.**

**일대 일 관계는 참조키(foreign key)가 有一 하게 정의함으로써구현된다.**

**(중복을 허용하지 않는다.)**

**● 일대 다수(One-to-Many)**

**기본 테이블의 각 행들은 참조 테이블의 하나 또는 그 이상의 행들과**

**관계되어 진다. 예를들면 1명의 customer는 많은 주문을 할 수 있다.**

**그러나 어떤 하나의 order는 많은 customer에 의해 정해 질 수 없다.**

**● 다수대 다수(Many-to-Many)**

**한 테이블의 많은 행들은 다른 테이블의 많은 행들과 관계되어진다.**

**예를 들어 "홍길동" 이라는 작가는 여러 종류의 책을 쓸 수 있고,**

**또한 SQL7.0 이라는 책은 "홍길동"이외에 여러 작가에 의해 쓰여질**

**수 있다. 두 테이블간의 Many-to-Many 관계는 다른 세 번째 테이블**

**생성 및 각각의 초기화 테이블로 부터 연결된일대다수(One-to-Many)**

**관계생성에 의해 구현된다.**

**3. 정규화(Nomalization)**

**-- 테이블과 관계들을 설계할 때 일반적으로 어떤 논리적 비일관성들이 발생**

**한다. 정규화(Nomalization)라 부르는 과정은 이런 비일관성이 일어나지 않**

**도록 한다.**

**정규화는 일관성이 있는 데이터베이스 설계를 생성하기 위해 테이블, 키,**

**컬럼, 관계등을 정제하는 과정이다. 정규화는 테이블에 많은 테스트 적용을**

**통해 수행된다. 정규화 3레벨(제1, 제2, 제3 정규형식)이 일반적으로 적용**

**된다.**

**-- 어떠한 릴레이션 구조가 바람직한 것인지, 바람직하지 못한 릴레이션을**

**어떻게 분해하여야 하는지에 관한 구체적인 판단기준을 제공한다.**

**정규화이론에서는 보다 높은 단계의 정규형으로 나아 갈수록 보다**

**바람직한 릴레이션 구조를 가진다고 간주한다.**

**즉, 높은 단계의 정규형으로 나아갈수록**

**- 데이타의 본질적 의미가 릴레이션 구조에 보다 정확히 반영되고,**

**- 데이타의 중복을 줄이고,**

**- 데이타의 변경시 발생하는 문제점을 방지하고,**

**- 데이타의 무결성을 제고할 수 있다고 본다.**

**\* 종류**

**3.1 비정규형 릴레이션**

**3.2 제 1정규형 릴레이션**

**3.3 제 2정규형 릴레이션**

**3.4 제 3정규형 릴레이션**

**3.5 Boyce/Codd 정규형 릴레이션**

**3.6 제 4정규형 릴레이션**

**3.7 제 5정규형 릴레이션**

**-- 3.1 비정규형 릴레이션**

**가장 기본적인 정규화 조건을 만족하지 못하는 릴레이션.**

**- multi-valued 속성 : 한 튜플이 여러개의 값을 취하는 것.**

**- composite속성 : 복합 값을 취하는 것.**

**표 3-1 비정규형 릴레이션(multi-valued 속성이 있음)**

**--------------------------------------------**

**학번 이름 학과 소속대학 과목명-성적**

**---------------------------------------------------**

**1000 갑 경영 상대 영어-A, 역사-B, 음악-C**

**2000 을 전산 공대 영어-D, 역사-A**

**3000 병 영문 문리대 음악-C**

**---------------------------------------------------**

**표 3-2 비정규형 릴레이션(composite 속성이 있음)**

**----------------------------------------------------**

**학번 이름 학과 소속대학 과목명-성적**

**----------------------------------------------------**

**1000 갑 경영 상대 영어-A**

**1000 갑 경영 상대 역사-B**

**1000 갑 경영 상대 음악-C**

**2000 을 전산 공대 영어-D**

**2000 을 전산 공대 역사-A**

**3000 병 영문 문리대 음악-C**

**----------------------------------------------------**

**\* 비정규형 릴레이션의 문제점**

**1) 사용자에게 표현되는 데이타의 구조가 복잡해 진다.**

**2) 데이타의 표현구조가 복잡해지면 그것을 물리적으로 지원하는**

**데이타베이스 구조 역시 복잡해지고 부담이 가중된다.**

**3) 데이타 관리와 사용이 복잡해 진다.**

**-------------------------------------------------------------- 3.2 제 1정규형 릴레이션**

**--------------------------------------------------------------**

**모든 속성이 오직 하나의 원자적 값(atomic value)만을 취하는**

**릴레이션.**

**표 6-3 학생 1 릴레이션**

**-------------------------------------------------------**

**학번 이름 학과 소속대학 과목명 성적**

**-------------------------------------------------------**

**1000 갑 경영 상대 영어 A**

**1000 갑 경영 상대 역사 B**

**1000 갑 경영 상대 음악 C**

**2000 을 전산 공대 영어 D**

**2000 을 전산 공대 역사 A**

**3000 병 영문 문리대 음악 C**

**-------------------------------------------------------**

**\* 제1정규형의 문제점**

**제1정규형은 가장 초보적인 정규형으로 그 차제만으로 바람직한**

**릴레이션 구조를 형성하고 있다고 보기 어렵고, 제2정규형의**

**조건을 만족하지 못하면 데이타의 사용과정에서 여러가지 문제를**

**발생시킨다.**

**1) 입력 문제**

**학생 1 릴레이션에서 각 튜플을 유일하게 식별할 수 있는**

**primary key는 "학번 + 과목명" 이다.**

**학생 1 릴레이션은 한 학생의 학번, 이름, 학과, 소속대학에**

**관한 데이타는 그 학생이 어떤 과목을 수강할 때까지 입력**

**하지 못한다.**

**릴레이션 구조 자체가 잘못되어 수강과목이 결정되지 않은**

**복학생, 신입생에 관한 데이타를 입력할 수 없다.**

**이것을 삽입이상(insertion anomaly)라 부른다.**

**2) 삭제 문제**

**학생 1 릴레이션에서 3000번 학생이 유일하게 수강하던**

**음악과목을 철회하면 해당 튜플을 삭제해야 한다.**

**그런데 이 튜플을 삭제하고 나면 "3000번 학생의 이름이**

**'병'이고, 영문학과에 소속하며, 문리대에 다닌다"는 데이타**

**마저 삭제되고 만다.**

**이것을 삭제이상(deletion anomaly)라 부른다.**

**3) 데이타 중복 문제**

**학생 1 릴레이션은 학생의 이름, 학과, 소속대학에 관한**

**데이타가 여러 튜플에 중복되어있다. 이것은 저장공간과**

**관리비용의 부담이 가중된다.**

**만약, 1000번 학생이 경영학과에서 전산학과로 소속학과를**

**변경했을 경우 그 학생에 해당하는 모든 튜플을 찾아서**

**일일히 학과 값을 갱신해야한다.**

**이것은 갱신에 따른 비용도 문제지만, 갱신작업이 철저히**

**이루어 지지 않았을 경우 데이타 값 불일치 현상이 발생하게**

**되고, 데이타베이스 정확성, 일관성, 무결성을 회손한다.**

**-- 3.3 제 2정규형 릴레이션**

**제1정규형에 속하고(single-values, atomic속성들로 구성), 모든**

**nonkey 속성들이 candidate key에 함수종속되어야 한다.**

**nonkey속성은 candidate key 구성에 참가하지 않는 속성**

**(예: 이름,학과, 소속대학, 성적)**

**표 3-4 학생 2 릴레이션 표 3-5 수강 릴레이션**

**-----------------------------------------------**

**학번 이름 학과 소속대학 학번 과목명 성적**

**------------------------------------------------**

**1000 갑 경영 상대 1000 영어 A**

**2000 을 전산 공대 1000 역사 B**

**3000 병 영문 문리대 1000 음악 C**

**2000 영어 D**

**2000 역사 A**

**3000 음악 C**

**------------------------------------------------**

**\* 제1정규형인 학생1릴레이션과 제2정규형인 학생2, 수강 릴레이션**

**과의 차이점.**

**- 한 학생의 학번, 이름, 학과, 소속대학 등에 관한 데이타가**

**그 학생이 수강하는 과목과는 아무런 상관없이 학생 2릴레이션**

**에 입력될 수 있다.**

**- 한 학생이 수강하는 과목명, 성적에 관한 데이타가 수강**

**릴레이션에서 삭제되어도 그 학생의 학번, 이름, 학과,**

**소속대학에 관한 데이타가 손상되지 않는다.**

**- 학생 2 릴레이션과 같이 데이타의 중복을 방지하고 있다.**

**\* 제2정규형의 문제점**

**제2정규형은 제3정규형의 조건을 충족하지 못하면 데이타의**

**사용 과정에서 여러가지 문제를 발생시킨다.**

**1) 입력 문제**

**학생 2 릴레이션의 학번은 Primary Key이며, NULL값을 취할**

**수 없고, 학번없이는 자료입력이 불가하다.**

**즉, "학생의 데이타없이 어떤 학과가 어떤 대학에 소속한다는**

**데이타가 입력될 수 없다." 예로 문리대학에 불문학과를**

**신설할 경우 불문학과 학생이 나타날 때까지 학생 2 릴레이션**

**에 입력될 수 없다.**

**2) 삭제 문제**

**학생 2 릴레이션에서 3000번 학생 튜플을 삭제할 경우 학생**

**데이타와 "영문학과가 문리대학에 소속한다"는 사실마저**

**사라져 버리게 된다.**

**3) 데이타 중복 문제**

**학생 2 릴레이션에서 경영학과 학생 튜플들이 많을 경우 각**

**튜플마다 "경영학과가 상대에 소속한다"라는 사실이 중복되게**

**된다. 중복 데이타로 인한 관리비용, 갱신비용, 데이타의**

**불일치 현상등의 문제점을 계속적으로 안고 있다.**

**-- 3.4 제 3정규형 릴레이션**

**제2정규형에 속하고, 모든 nonkey 속성들 사이에 함수 종속 관계가**

**없어야 한다.**

**표 3-6 학생 3 릴레이션 표 3-7 학과 릴레이션**

**------------------------------------------**

**학번 이름 학과 학과 소속대학**

**------------------------------------------**

**1000 갑 경영 경영 상대**

**2000 을 전산 전산 공대**

**3000 병 영문 영문 문리대**

**------------------------------------------**

**표 3-8 수강 릴레이션**

**----------------------------**

**학번 과목명 성적**

**----------------------------**

**1000 영어 A**

**1000 역사 B**

**1000 음악 C**

**2000 영어 D**

**2000 역사 A**

**3000 음악 C**

**----------------------------**

**\* 제2정규형인 학생 2 릴레이션과 제3정규형인 학생 3, 학과, 수강**

**릴레이션과의 차이점.**

**- 학생 데이타와 상관없이 어떤 학과가 어떤 대학에 소속한다는**

**사실을 학과 릴레이션에 입력할 수 있다.**

**- 학생 데이타를 삭제할 때 학과의 소속대학에 관한 데이타가**

**함께 삭제 안된다.**

**- 어떤 학과가 어떤 대학에 소속한다는 사실의 데이타 중복을**

**방지할 수 있다.**

**(예, 경영학과가 상대에 소속한다.)**

**-- \*\*\* 릴레이션의 분해요령**

**릴레이션을 낮은 단계에서 높은 정규형으로 전환한다는 것은 결국**

**원래의 릴레이션을 여러개의 작은 릴레이션으로 분해하는것.**

**1) 무손실 분해(nonloss decomposition)**

**무손실 분해는 project연산에 의해 분해된 여러 projection들이**

**join연산에 의해 원래의 릴레이션으로 되돌아 갈 수 있어야 함을 의미.**

**예)**

**학생3(학번,이름,학과) --+**

**|→ join → 학생2(학번,이름,학과,소속대학)**

**학과 (학과,소속대학) --+**

**학생2(학번,이름,학과,소속대학)--+**

**|→ join →학생1(학번,이름,학과,소속대학,**

**수강 (학번,과목명,성적) --+ 과목명, 성적)**

**무손실 분해는 때로는 원래의 릴레이션 구조로는 표현할 수**

**없었던 새로운 사실도 담을 수 있다.**

**예로 "영문과는 문리대학에 소속한다" 는 제2정규형인 학생 2**

**릴레이션에서는 단독으로 표현될 수 없었지만, 제3정규형인**

**학과 릴레이션을 통해서 적절히 표현되고 있다.**

**그래서 정규화는 비단 데이타 삽입/삭제 또는 중복의 문제를**

**방지할 뿐만아니라, 릴레이션 구조 자체가 보다 강력한 표현력을**

**지니고 데이타의 본질적 구조로 근접해가는 과정으로 이해할**

**수 있다.**

**2) 손실 분해**

**손실 분해는 무손실 분해와는 반대로, 원래의 릴레이션을**

**분해함으로써 기존에 있는 데이타가 상실되는 것.**

**즉, 분해되어진 projection들을 join을 하더라도 원래의**

**릴레이션이 복구되지 않는다.**

**예) 학생 2 릴레이션의 손실 분해**

**표 3-9 학생 4 릴레이션 표 3-10 학과 릴레이션**

**-------------------------- ----------------------**

**학번 이름 소속대학 학과 소속대학**

**-------------------------- ----------------------**

**1000 갑 상대 경영 상대**

**2000 을 공대 전산 공대**

**3000 병 문리대 영문 문리대**

**4000 정 상대 경제 상대**

**-------------------------- ----------------------**

**학생 4, 학과 릴레이션과 같이 정규화 이루어진 경우 두**

**projection을 join 하더라도 원래의 학생 2 릴레이션으로**

**되돌아 갈 수 없다.**

**학번이 1000, 4000번 학생의 학과가 "경영" 인지 "경제"**

**인지를 알 수 가없기 때문에 원래의 릴레이션을 분해하는**

**과정에서 데이타가 손실 되었다.**

**--> ansi join**

**select 학생.이름, 학과.학과, 학과.소속대학, 수강.과목명, 수강.성적**

**from 학생 inner join 수강**

**on 학생.학번 = 수강.학번**

**inner join 학과**

**on 학생.학과 = 학과.학과**

**select a.이름, c.학과, c.소속대학, b.과목명, b.성적**

**from 학생 a inner join 수강 b**

**on a.학번 = b.학번**

**inner join 학과 c**

**on a.학과 = c.학과**

**--> sql server join**

**select 학생.이름, 학과.학과, 학과.소속대학, 수강.과목명, 수강.성적**

**from 학생, 학과, 수강**

**where 학생.학번 = 수강.학번 and**

**학생.학과 = 학과.학과**

**select 이름, 학과.학과, 소속대학, 과목명, 성적**

**from 학생 a, 학과 b, 수강 c**

**where a.학번 = c.학번 and**

**a.학과 = b.학과**

**◈ 정규화의 이점은 다음을 포함한다.**

**● 테이블들이 좀더 좁혀지기 때문에 정렬 및 index를 생성할 수 있다.**

**● 더 많은 테이블이 있기 때문에 더 많은 clustered index를 사용할 수 있다.**

**● index가 작고 경제적이다.**

**● 테이블당 좀더 적은 index는 update 성능을 향상시킨다.**

**● 더 적은 Null 및 불필요한 데이터의 감소는 데이터베이스의 조밀성을 증가시**

**킨다.**

**● 더 적은 데이터에 영향을 주기 때문에 동시 테이블 잠김이 감소한다.**