ERWin 4.0 사용 방법

현 대 정 보 기 술 ㈜

김연홍(k3701@hitel.net)

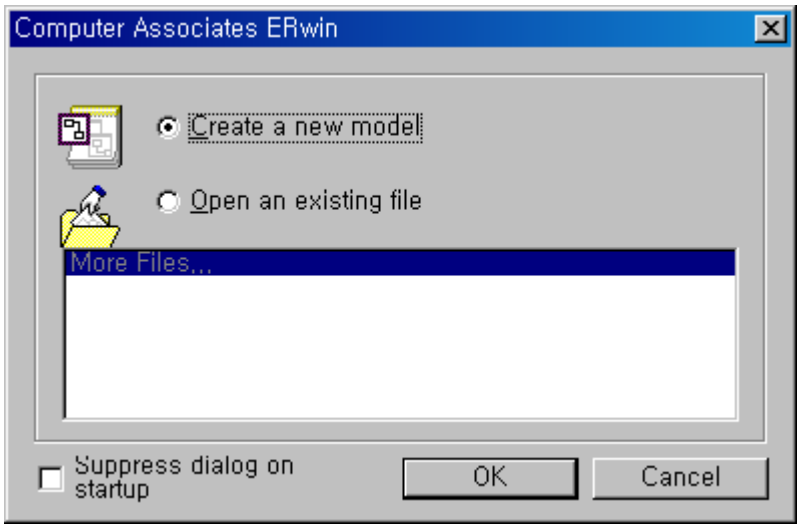
7. ERWin 사용 방법

이 장에서는 ERWin 의 사용방법과 기능에 대해서 설명할 것이며 단순한 ERWin 의 기능적인 접근이 아닌 실제 예제를 모델링하면서 관련된 ERWin 의 기능에 대해서 설명하고자 한다. 그리고 이 책은 ERwin 의 매뉴얼이 아니므로 ERwin 의 모든 기능이 아닌 실제 프로젝트를 하면서 자주 사용하게 되는 기능을 중심으로 설명하도록 하겠다.

여기서 사용하는 ERWin 의 버전은 4.0 으로 최신 버전이며 ERWin4.0 파일은 www.x-lution.com 사이트의 데이터 모델링 자료실에 올려 놓았으니 참조하기 바란다.

[ERWin4.0 Time Version 설치와 라이센스 설치방법 설명]
[제니시스와 협의]

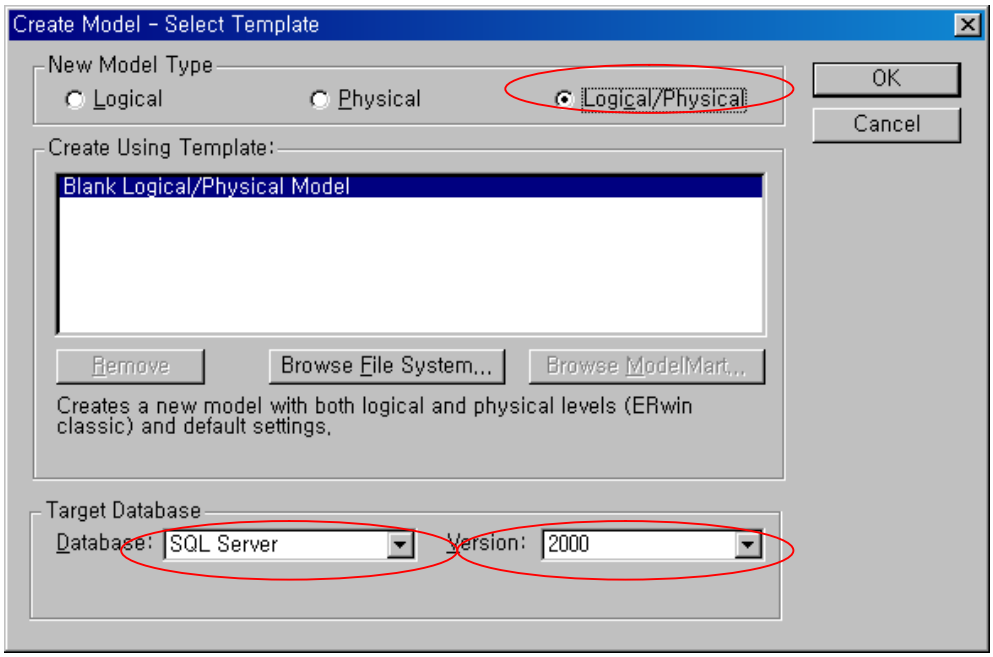
위 방법에 따라 ERWin 을 설치 했으면 시작 -> 프로그램 -> Computer Associates ERWin 4.0 -> ERWin4.0 을 선택하여 ERwin 을 실행 한다.



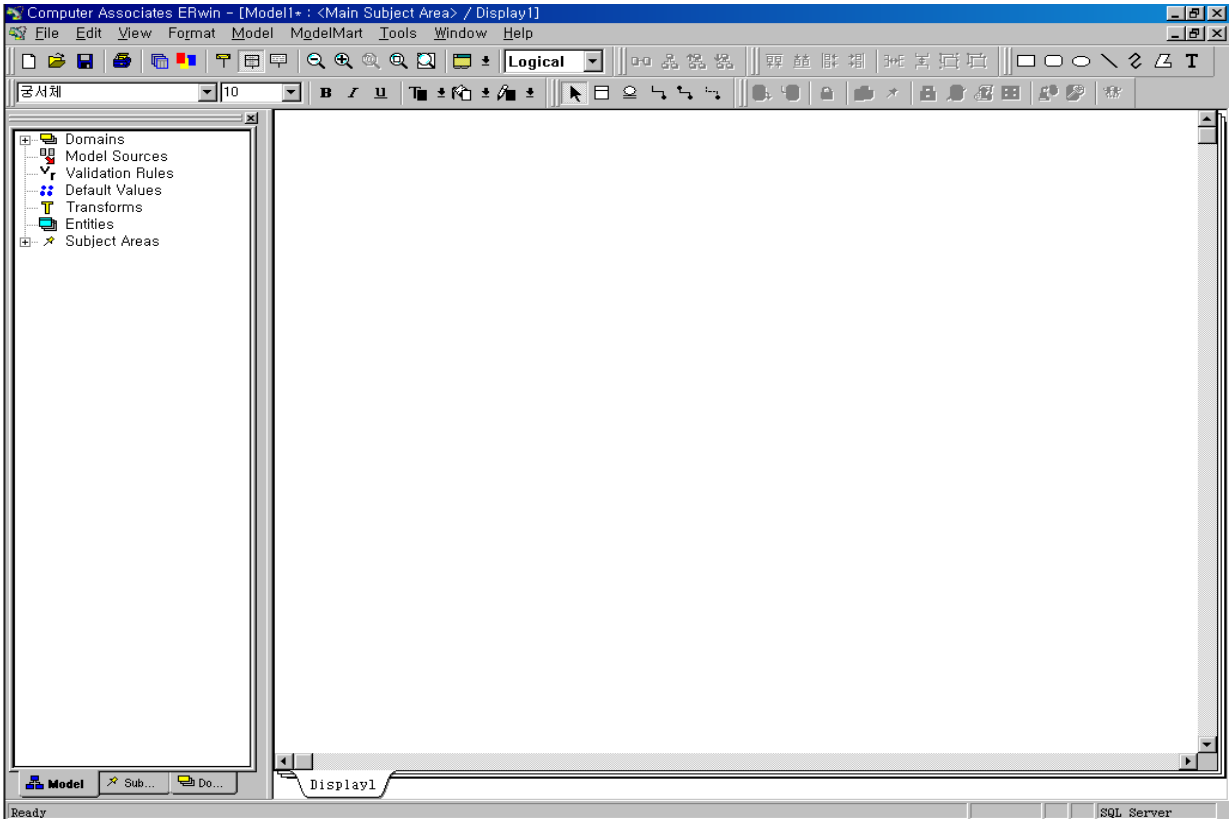
그러면 위 같이 ERwin 초기 화면이 나타나는데 'Create a new model'옵션 버튼을 선택하고 OK 버튼을 누르면 다음과 같이 Create Model 대화상자가 나타나게 되고 여기서 세번째 옵션 버튼인 Logical/Physical 옵션 버튼을 선택한다.

기존 ERwin 파일을 열고자 한다면 'Open an existing file'옵션 버튼을 선택한다. 그런 다음 Create Model 대화상자에서 모델의 유형으로 Logical/Physical 옵션 버튼을 선택한 뒤 폼 아래 부분에 개발 대상 데이터베이스를 선택해야 하는데 여기서는 SQL Server 2000

로 설명할 것이므로 DB 는 SQL Server 를 선택하고 버전은 2000 을 선택한다. (참고로 이전 버전인 ERwin3.5.2 는 SQL Server 7.0 까지 지원했다.)



그런 다음 OK 버튼을 누르면 다음과 같이 ERWin 메인 화면이 나타나게 된다.



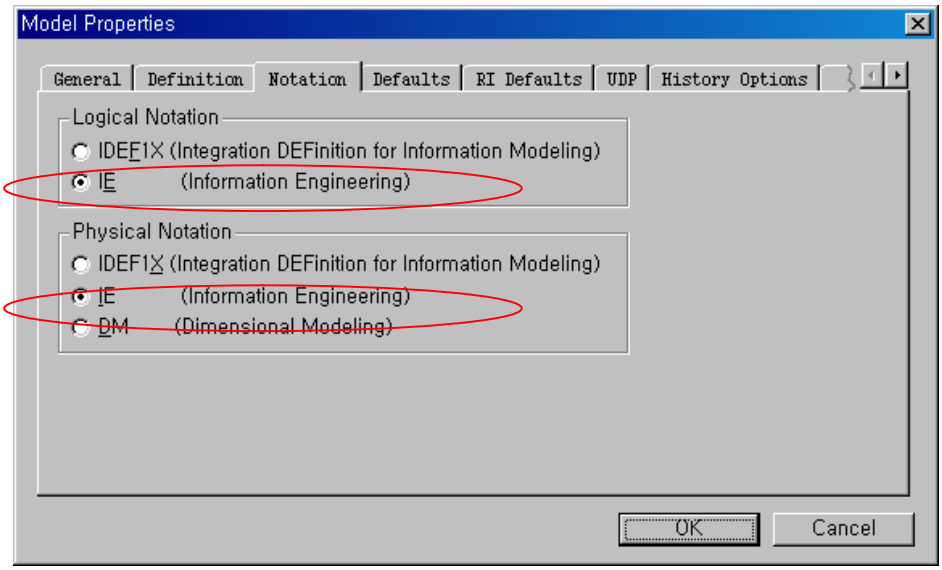
메인 화면은 도구 메뉴와 Model Explorer 그리고 다이어그램등 크게 세가지 부분으로 구성되어 있으며 Model Explorer 는 ERwin 4.0 에 새롭게 추가된 부분이다. 마치 SQL Server 의 쿼리 분석기와 비슷한 모습이며 하는 역할도 다이어그램에서 정의 된 개체들의 정보를 계층적으로 보여주고 있어서 기능적으로도 비슷한 역할을 하고 있다.

-. ERwin 표기방식

메인 화면이 열리면 가장 먼저 정의해야 할 내용은 어떠한 표기 방법을 사용할 것이냐 하는 것이다.

ERWin 은 크게 두 가지 표기법을 지원하는데 하나는 IE (Information Engineering)방식과 Idef1x (Integration DEFinition for Information Modeling)방식 이다.

IE 표기방식은 정보 공학 표기방식으로 우리가 일반적으로 모델링을 할 때 가장 많이 사용하는 유형이며 Idef1x 방식은 미 국방성에서 프로젝트 표준안으로 개발한 표기방식이다. 이 책에서는 일반적으로 널리 사용되는 IE 표기 방식을 이용하여 모델링을 사용도록 하겠다. 기본적으로 ERWin 을 설치하게 되면 Idef1x 방식이 선택되어지며 이 설정을 IE 표기방식으로 바꾸려면 ERWin 초기 화면의 메뉴에서 Model / Model Properties..를 선택하면 다음과 같은 Model Properties 대화상자가 나타나게 되는데 세 번째 Notation 탭에서 Logical Notation Physical Notation 영역 모두 IE 옵션 버튼을 선택하면 된다.



이렇게 하면 ERWin Toolbox 의 모습이 아래와 같이 바뀌게 된다.

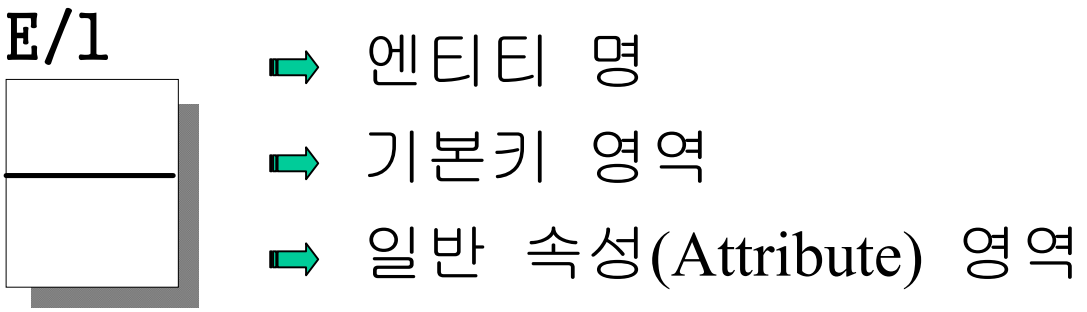
-. Logical Nation 과 Physical Nation

ERWin 은 기본적으로 개념적 데이터 모델링은 지원하지 않으며 논리적 데이터 모델링과 물리적 데이터 모델링을 지원한다.
그러므로 ERWin 을 사용하기 위해서는 먼저 업무적인 분석과 기본적인 엔티티와 Attribute 등이 정의된 양식이 있어야 하며 이를 ERWin 으로 옮기면서 관계형 데이터베이스 모델링 이론에 입각해서 ERD 를 작성하게 된다.
ERWin 에서 논리적 데이터 모델링과 물리적 데이터 모델링을 선택하기 위해서는 아래 그림에 나와있는 것처럼 ERwin Toolbar 오른쪽의 콤보 박스를 선택하면 된다.

-. 엔티티 생성

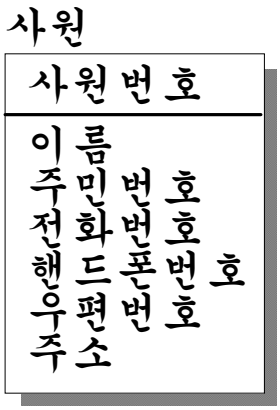
엔티티의 생성은 ERWin Tool Box 의 두 번째 버튼이며 두 번째 버튼을 선택한 뒤 화면상에 클릭하면 엔티티가 만들어 진다.

엔티티는 다음과 같이 세 영역으로 구성된다.



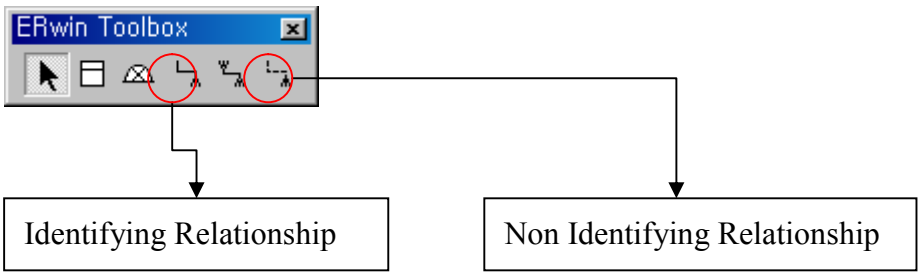
세 영역을 이동하기 위해서는 엔티티를 선택한 뒤 텝키를 이용하여 영역을 이동할 수 있으며 기본키와 일반 속성영역에 속성을 추가하고자 한다면 엔터키를 치면 새로운 속성을 기술할 공간이 만들어 진다.

그러면 다음과 같이 사원 엔티티를 만들어 보자

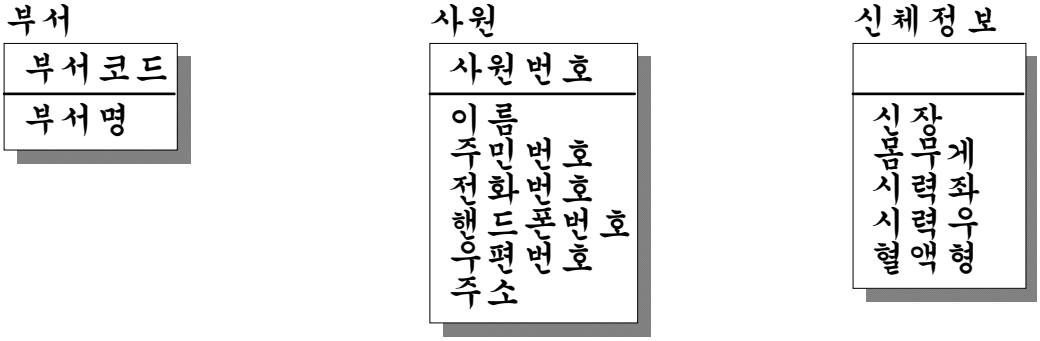


-. 식별관계, 비 식별관계

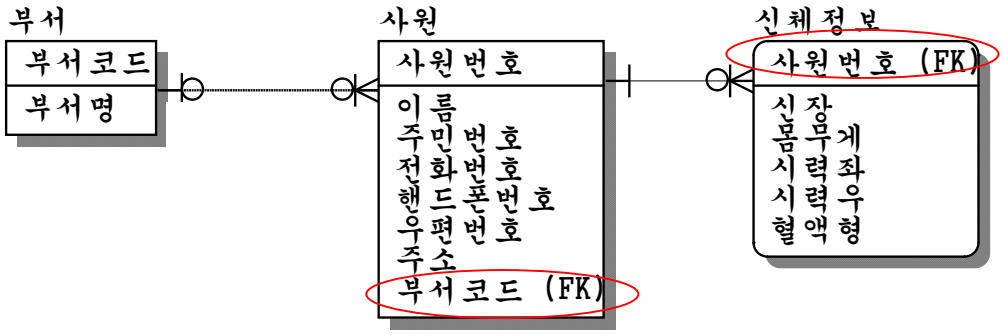
위에서 관계란 두 엔티티간의 업무적인 연관성이라 정의한 바 있다. 그리고 관계의 유형에는 부모 테이블의 기본키 혹은 복합키가 자식 테이블의 기본키 혹은 복합키의 구성원으로 전이되는 식별관계와 자식 테이블의 일반 속성(Attribute) 그룹의 구성원으로 전이되는 비 식별관계가 있다.
식별관계와 비 식별관계의 관계 선을 정의하기 위해서는 ERwin Toolbox 에서 네번째 버튼이 식별관계의 관계 선이고 여섯번째 버튼이 비 식별관계의 관계 선이다.



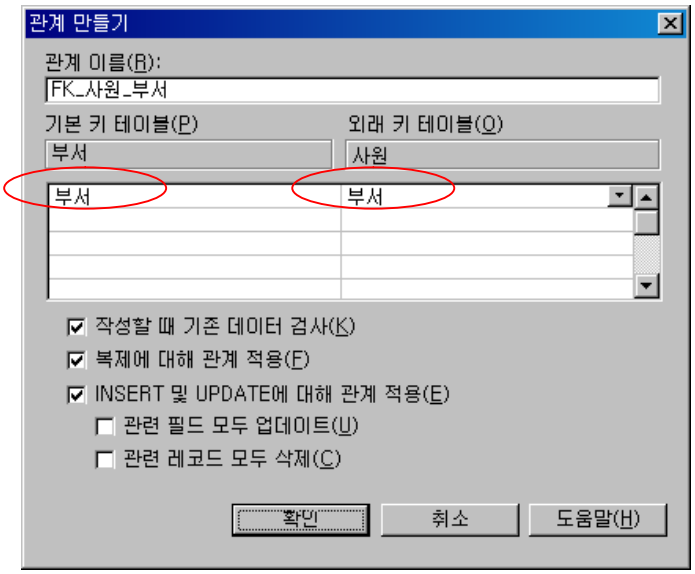
그러면 다음과 같이 사원의 신체조건과 부서를 정의하기 위한 부서 테이블을 추가해 보도록 하자.



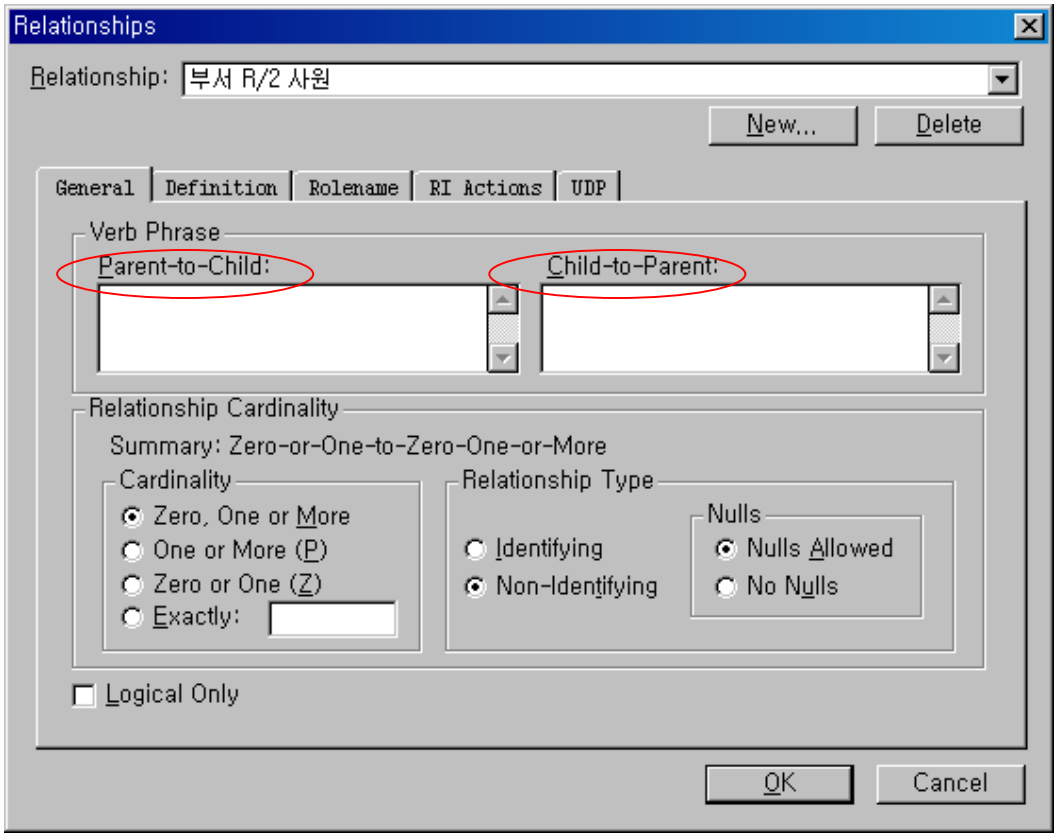
위 테이블에서 부서 테이블과 사원 테이블간의 관계는 부서에 사원이 소속됨으로 부서 테이블이 부모 테이블이 되고 사원들의 부가적인 신체 정보를 저장하기 위한 것이니 만큼 사원 테이블이 신체정보 테이블과의 관계에서는 부모 테이블이 된다.
그러면 관계의 유형은 어떻게 되는가 ?
부서 테이블과 사원 테이블에서 부서는 사원의 부분적인 정보를 표현함으로 비 식별관계 이며 사원 테이블과 신체정보 테이블에서는 사원들 개개인의 신체 정보를 저장하게 됨으로 식별관계가 된다. 식별관계나 비 식별관계 모두 관계를 형성하기 위해서는 ERwin Toolbox 에서 관계 유형에 맞는 관계선을 선택하고 부모 테이블을 선택한 뒤에 자식 테이블을 선택한다.



이처럼 관계를 형성하게 되면 부모 테이블의 기본키가 자식 테이블에 자동으로 전이되어지게 되는데 SQL Server 의 엔터프라이즈 관리자의 다이어그램에서는 관계를 통해 부모 테이블의 기본키가 자식테이블의 외래키(Foreign Key)로 자동으로 전이되어지지 않는다.
SQL Server 에서 관계를 형성하기 위해서는 자식 테이블에 부모 테이블의 기본키와 대응하는 컬럼을 만들어 주고 부모 테이블의 기본키를 드레그해서 자식테이블에 드롭하면 아래와 같이 관계 만들기 대화상자가 나타나게 되는데 여기서 자식 테이블(외래키 테이블)에서 부서 필드를 선택해 주어야 관계가 만들어지게 된다.



(SQL Server 의 엔터프라이즈 관리자는 SQL Server 의 관리도구 이지 CASE Tool 은 아니다.)
관계를 형성한 뒤에는 관계에 관한 옵션을 설정해야 하는데 부서 테이블과 사원 테이블간의 관계선을 선택한 후 오른쪽 버튼을 눌러 나타나는 팝업 메뉴에서 Relationship Properties 메뉴를 선택하면 다음과 같이 Relationships 대화상자가 나타난다.

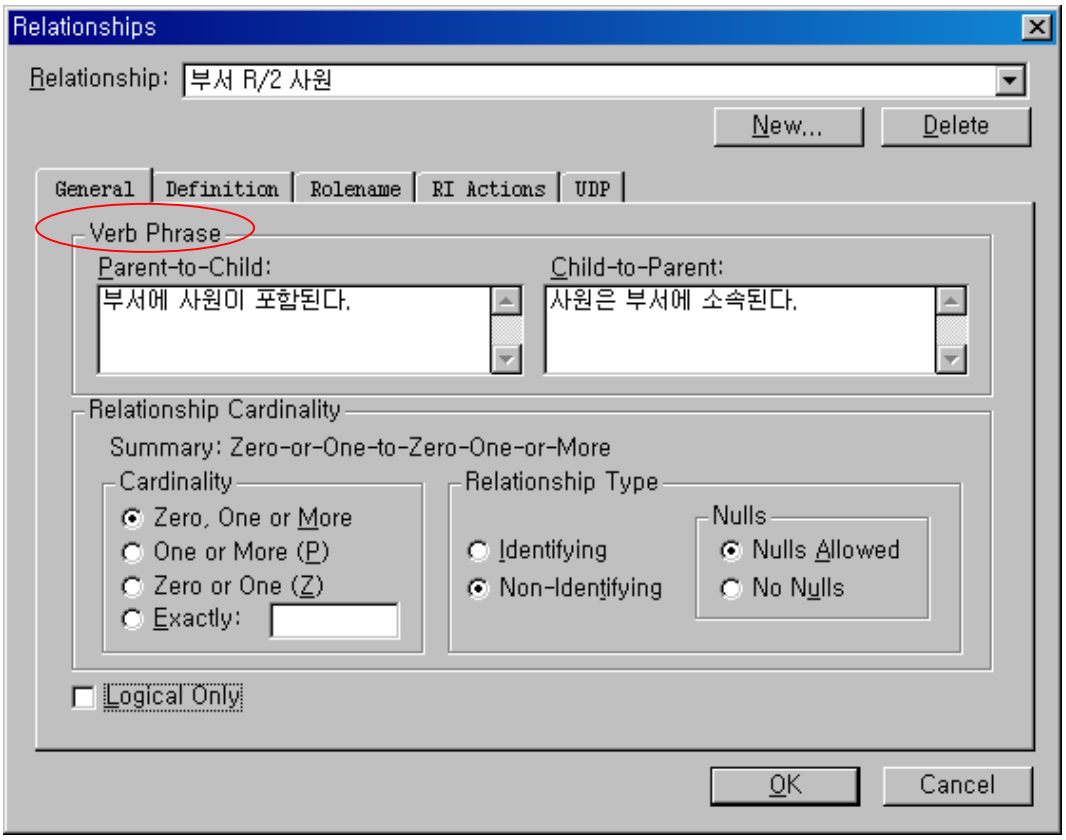


이 대화상자에서는 Cardinality 와 Relationship Type 이 주요한 옵션이며 Cardinality 는 두 테이블에서 레코드들의 메칭 정보를 보여주게 되는데 옵션을 바꾸어가면서 실제 다이어그램에서의 바뀌는 모습을 확인해 보도록 하자.

위의 예에서는 하나의 부서에 사원이 없을 수도, 한명만 있을 수도 아니면 여러 명 있을 수도 있으므로 첫번째 Zero, One or More 옵션이 올바르다.

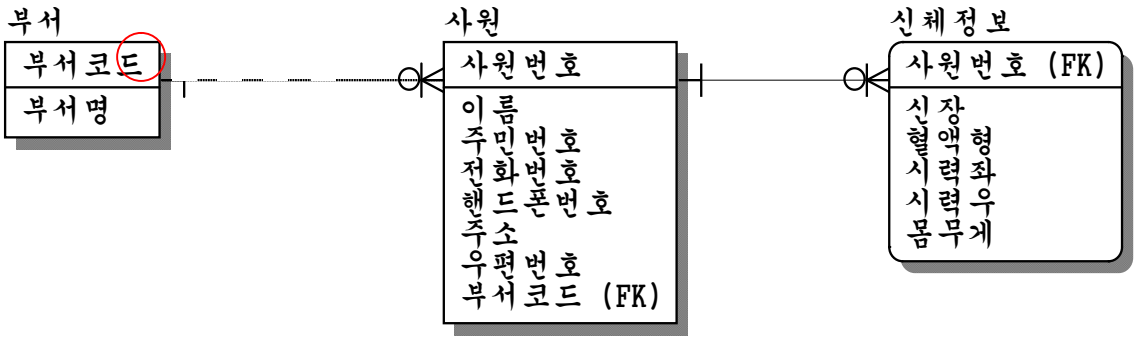
그리고 위의 화면에서 Verb Phrase 는 부모와 자식(Parent-to-Child) 그리고 자식과 부모(Child-to-Parent)와의 관계에 대한 설명적인 문구를 입력하는 곳이다.

해서 아래의 그림과 같이 두 테이블간 적당한 내용을 정의해 두면 된다.



Verb Phrase 는 실제 모델링에서 중요한 옵션이라기보다는 관계에 관한 가독성을 높이기 위한 주석 정도로 이해하면 될 것 이다.

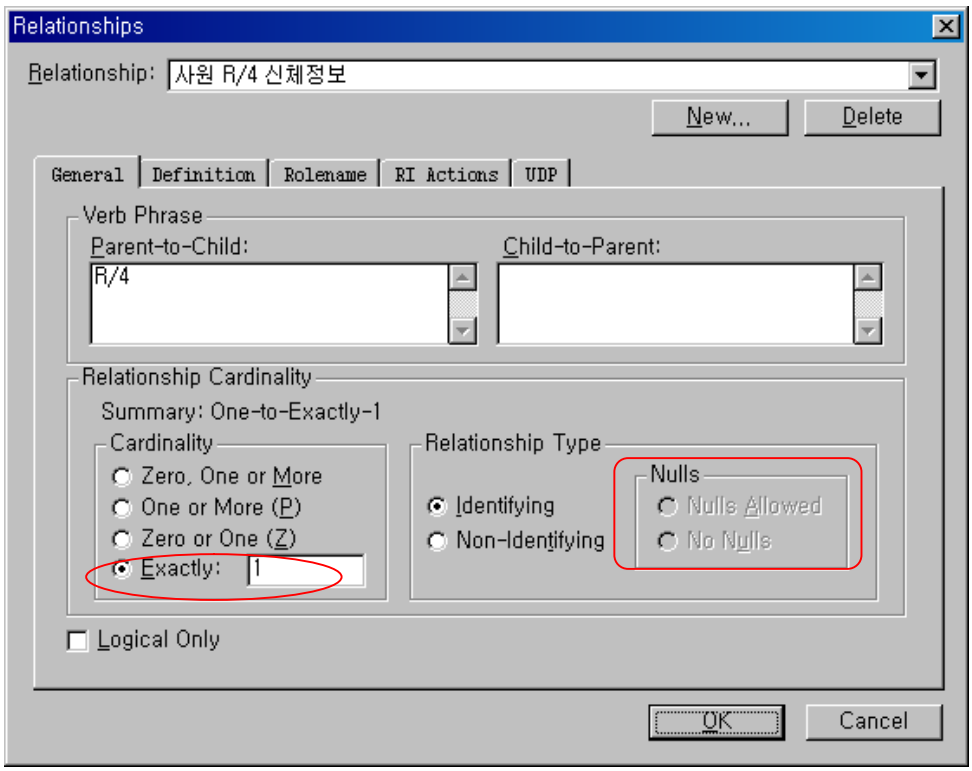
그 다음은 Relationship Type 인데 부서 테이블과 사원 테이블의 관계 유형이 비 식별 관계이므로 Non-Identifying 옵션이 선택되어 있다. 그리고 Null 에 대한 옵션을 선택할 수 있는며 기본적으로 비 식별관계에서는 부모 테이블에서 널을 허용할 수 있게끔 옵션이 선택되어 있는데 이는 대부분의 경우에 올바른 옵션이 아니다.
Null 에 대한 옵션을 No Nulls 로 선택하도록 하자.
그러면 아래와 같은 모습이 될 것이다.



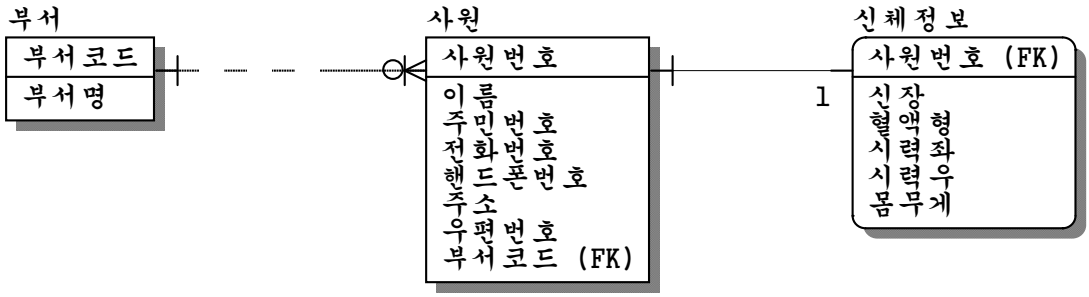
이번에는 사원 테이블과 신체정보 테이블 사이의 관계선을 살펴보도록 하자.

사원과 신체정보 테이블의 관계선에 대한 Relationships 대화상자를 보면 식별 관계로 정의되어 있으며 식별 관계로 관계가 형성되었을 때는 널에 대한 옵션 설정이 비 활성화 된다.

그리고 Cardinality 옵션은 사원 테이블에 하나의 레코드는 반드시 신체정보 테이블에도 대응되는 하나의 레코드가 존재해야 하므로 아래의 그림 처럼 Exactly 1 관계로 정의해야 한다.

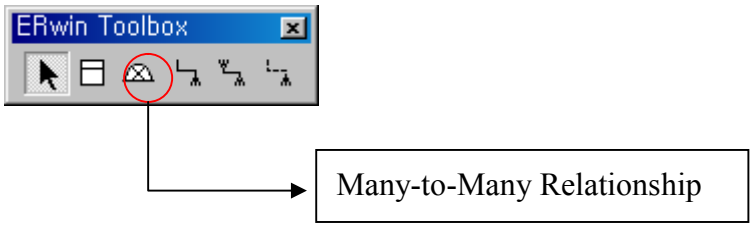


Exactly 을 선택하고 임의의 숫자를 입력하면 자식 테이블쪽의 관계선 끝이 직선으로 표시되는데 이를 정확하게 보기 위해서는 ERwin 다이어그램 빈공간에서 오른쪽 버튼을 누른 뒤 팝업 메뉴에서 Relationship Display / Cardinality 를 선택하면 된다.

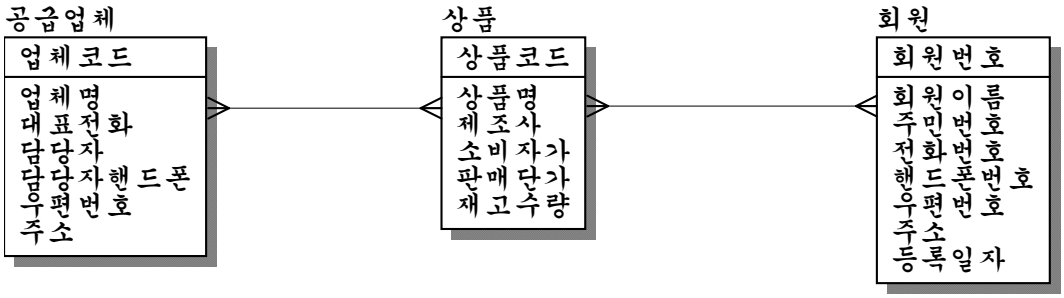


-. 다대다 해소 방법

앞에서도 설명했지만 다대다 관계는 논리적으로는 존재할 수 있지만 물리적으로는 존재할 수 없다.
ERwin 에서도 역시 ERwin Toolbox 의 관계선을 이용하여 다대다관계를 표현할 수 있다.



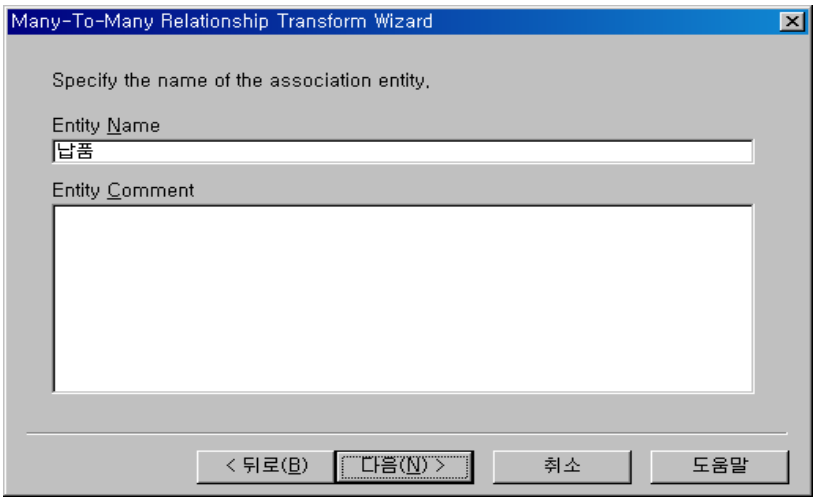
그러면 다대다 관계선을 이용하여 아래의 그림처럼 다이어그램을 만들어 보자.
(다대다 관계에 있는 엔티티들은 부모와 자식의 관계가 아니므로 다대다 관계선을 선택한 후 순서에 상관없이 두 엔티티를 차례로 선택하면 관계가 형성된다.)



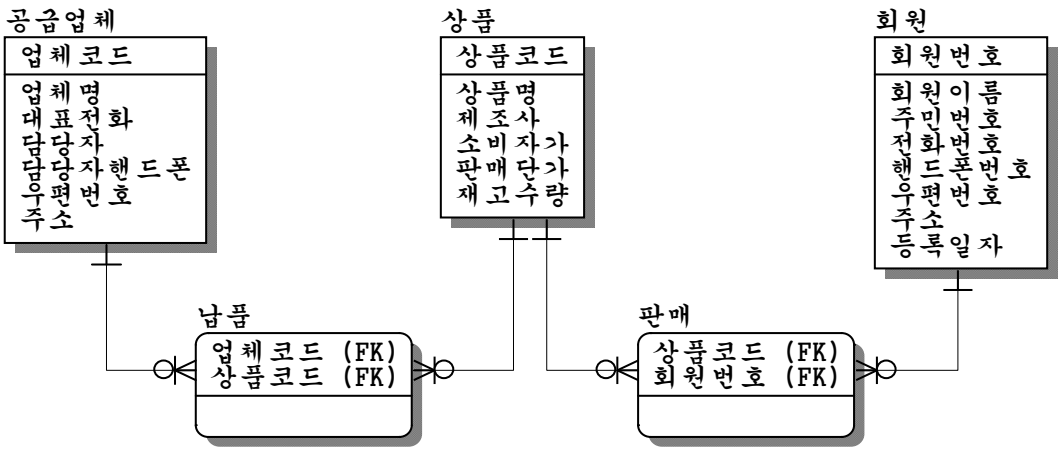
하나의 공급업체는 여러 개의 상품을 납품할 수 있고 하나의 상품은 여러 공급업체에서 납품 받을 수 있기 때문에 공급업체와 상품 엔티티간은 다대다 관계이며 하나의 상품은 여러 회원에게 판매할 수 있고 한명의 회원은 여러 상품을 구매할 수 있으므로 이 역시 다대다 관계이다.

그러나 대부분의 경우에 논리적 모델링에서도 다대다 관계를 풀어서 교차실체(행위실체)를 정의해야 한다. 왜냐하면 대부분의 업무적 프로세스와 상세 정보가 바로 이 교차 실체에서 정의되기 때문이다.

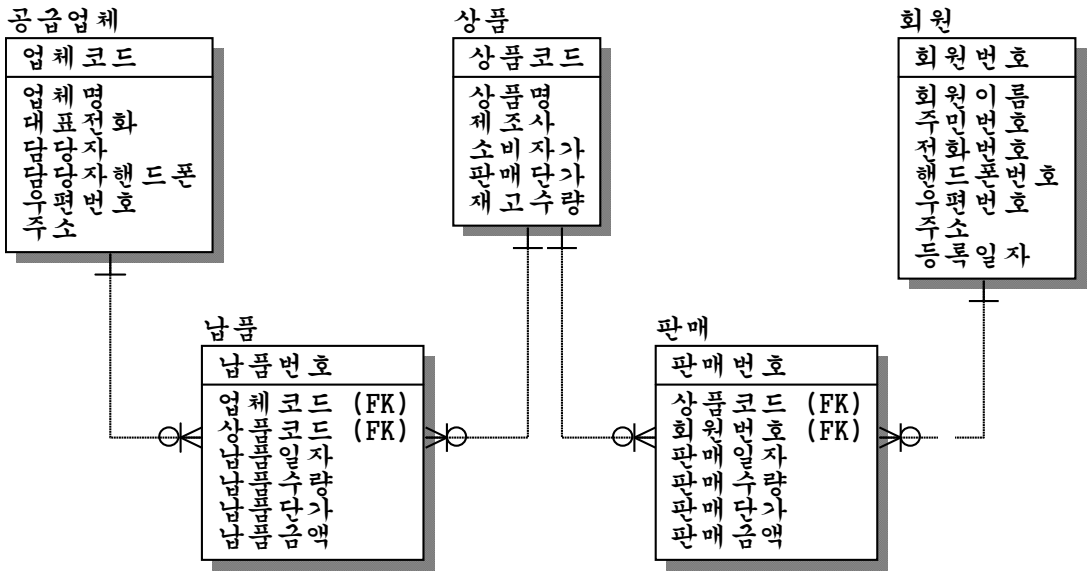
해서 다대다 관계를 해소하려면 위의 다대다 관계선에서 오른쪽 버튼을 누른 뒤 팝업 메뉴에서 Create Association Entity 메뉴를 선택하면 다음과 같이 마법사가 나타나는데 여기서는 새롭게 추가될 교차 실체의 실체명을 정의하고 다음 버튼을 눌러 작업을 완료한다.



공급업체와 상품 엔티티 간에 정의될 수 있는 프로세스는 납품이고 회원과 상품 엔티티 간에 정의될 수 있는 프로세스는 판매이므로 각각 납품과 판매를 교차실체명으로 정의하여 다대다 관계를 해소하면 다음과 같은 모습이 된다.



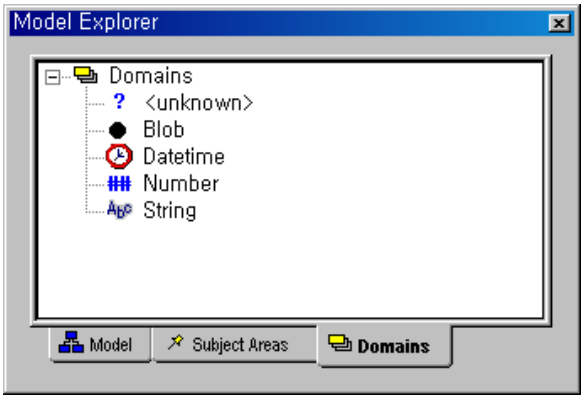
마지막으로 납품과 판매 엔티티에 속성(Attribute)를 추가해 보도록 하겠다.
일반적으로 납품 엔티티는 납품번호로 그리고 판매 엔티티는 판매번호로 관리되어지기 때문에 납품번호와 판매번호를 기본키로 정의하고 필요한 속성을 추가해 보도록 하자.



(납품번호와 판매번호를 기본키로 정의하려면 관계선이 비식별관계로 정의되어야 한다.)

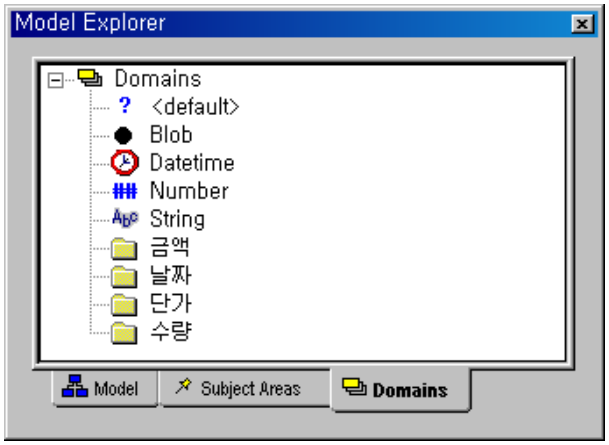
-. Domains 설정

납품 엔티티와 판매 엔티티의 속성들을 보면 일자, 수량, 단가, 금액등의 컬럼이 같이 쓰여지고 있는 모습을 볼 수가 있는데 이렇게 여러 엔티티에서 공통적으로 적용되는 속성이 존재하면 그것을 하나의 개체로 만들어 적용시키는 것이 훨씬 더 편리할 수 있다. SQL Server 에서 사용자 정의 데이터 타입을 정의해서 필요한 컬럼에 바인딩하는 것과 같은 내용이다. 이러한 내용을 정의하기 위해서는 Model Explorer 의 Domain 텝에서 정의한다.

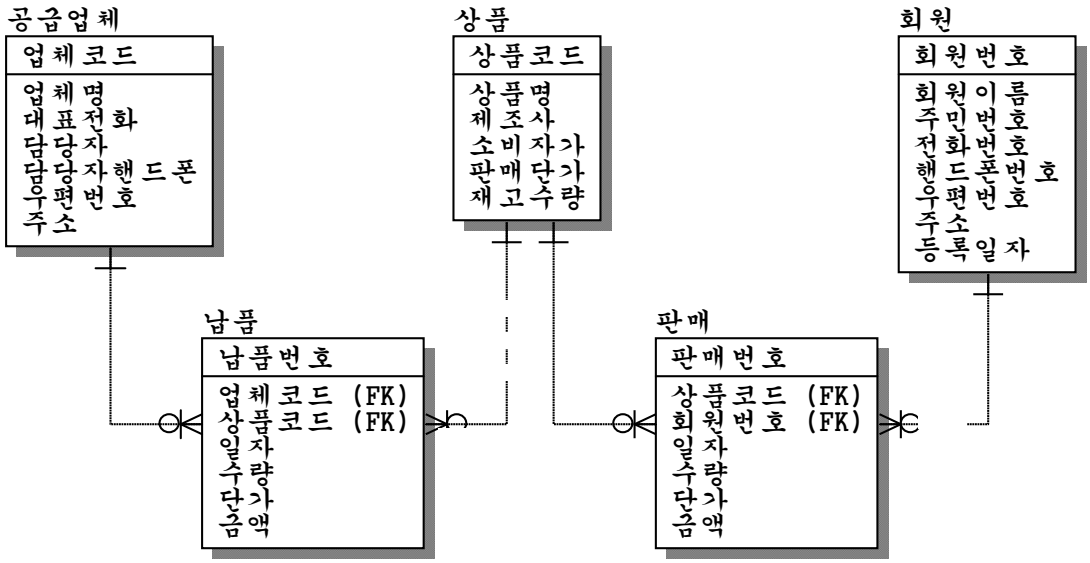


일자는 문자열로 정의할 것이며 수량, 단가, 금액은 숫자형 속성들이다.
아직은 논리적(Logical) 모델링 단계이므로 정확한 Data Type 과 Size 를 정의할 단계는 아니지만 그레도 문자형인지 숫자형인지는 구분을 해서 만들어 주어야 한다. 우선 일자를 정의해 보도록 하자. 일자는 문자형으로 정의할 것이므로 String 에서 오른쪽 버튼 New 를 선택한 뒤 폴더 모습의 아이콘이 만들어지면 일자라고 정의한다. 그런 다음 수량, 단가, 금액 등은 Number 에서 오른쪽 버튼 New 를 선택한 뒤 차례로 만들어 나간다.

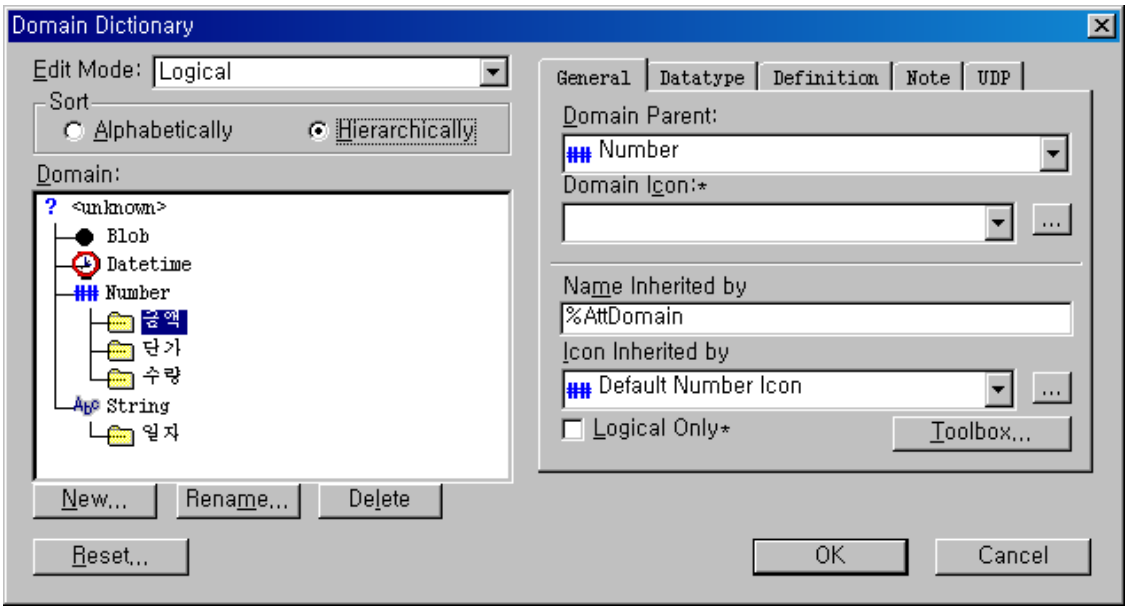
그러면 아래와 같은 모습이 될 것이다.



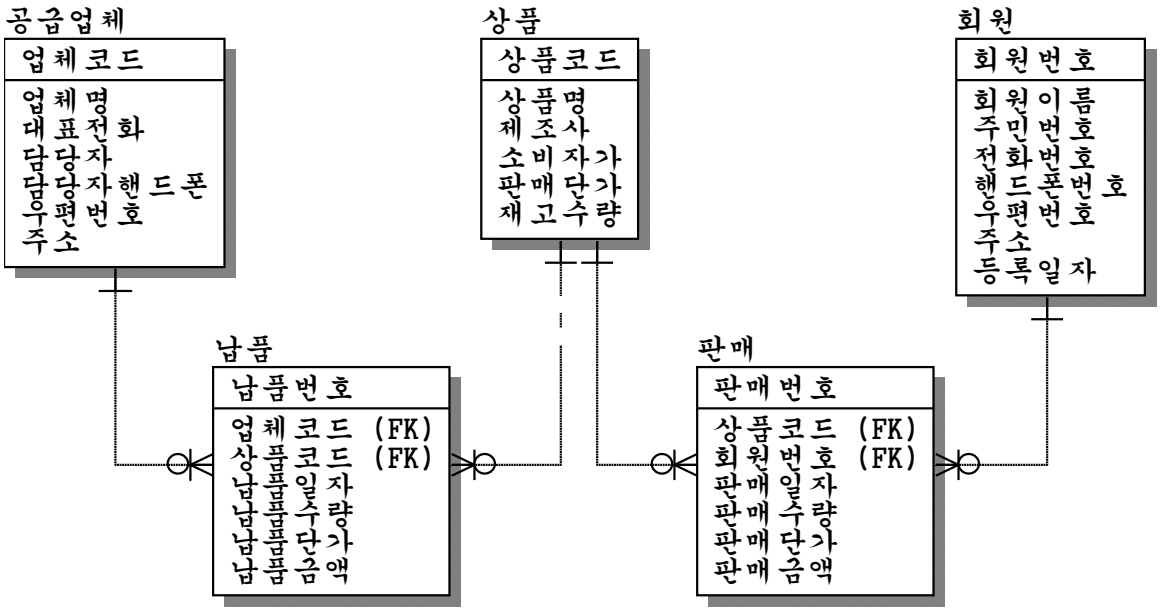
그러면 납품 엔티티와 판매 엔티티에 있는 납품 일자, 단가, 수량, 금액 속성과 판매 일자, 단가, 수량, 금액 속성들을 제거한 뒤 Model Explorer 에 있는 금액과 단가수량, 일자등을 선택해서 드래그 한 다음 납품 엔티티와 판매 엔티티에 차례로 드롭하면 다음과 같은 모습이 된다.



여기서 속성(Attribute)의 의미를 좀더 명확하게 하기 위해서 속성(Attribute)명 앞에 해당 엔티티명을 추가할 수 있는데 이렇게 하려면 Model Explorer 의 Domains 하위 폴더 중 하나를 선택한 후 오른쪽 버튼을 누른 뒤 Properties 메뉴를 선택하면 Domain Directory 대화상자가 나타나게 되는데 여기서 Hierarchically 옵션 버튼을 선택하면 아래와 같은 모습이 된다.



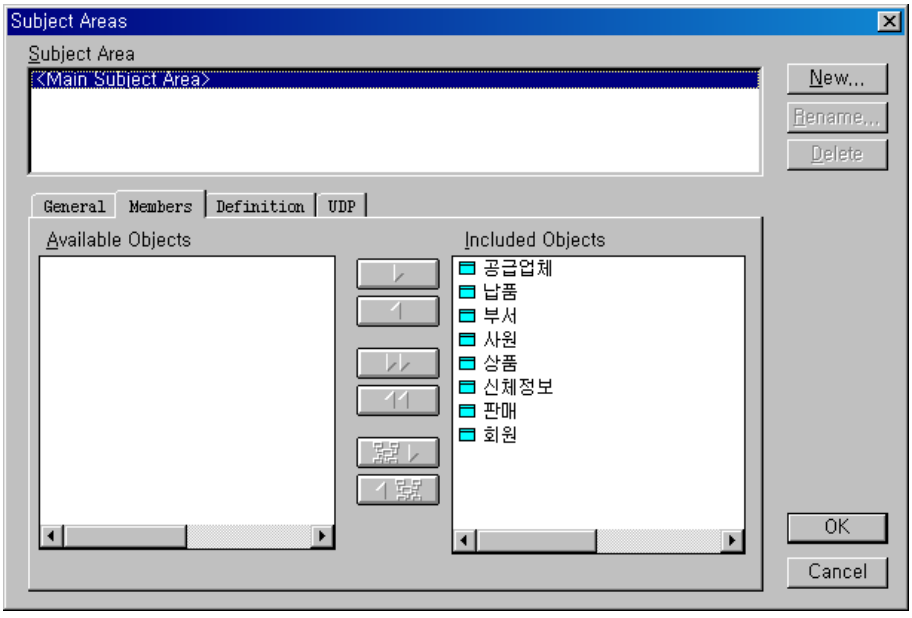
그리고 오른쪽에 Name Inherited by 입력 상자의 내용이 기본적으로 %AttDomain 으로 정의되어 있는데 이 앞에 %EntityName 을 붙여서 %EntityName%AttDomain 이라고 정의하면 적용되는 모든 속성명 앞에 엔티티명이 함께 붙어서 정의 되어진다.



-. Subject Area

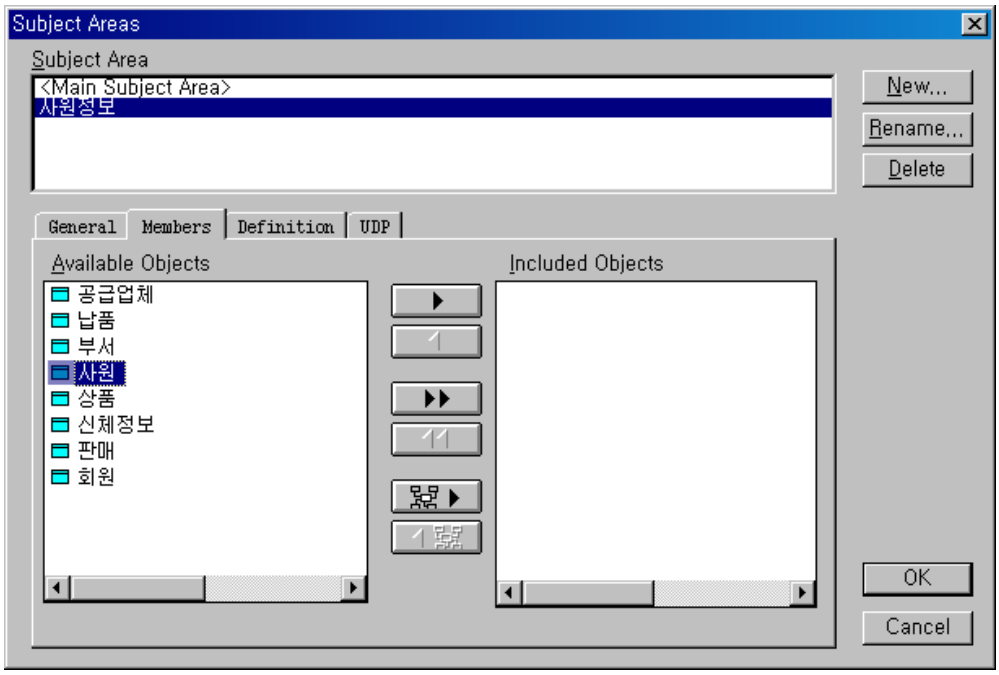
일반적으로 업무 분석을 해서 엔티티(Entity)를 추출하게 되면 적게는 10 개 미만부터 많게는 100 개 이상의 엔티티가 정의되는 모습을 보게 되는데 이러한 많은 엔티티를 한 화면에서 모두 관리한다면 너무나 복잡할 것이다. 이럴 경우 Subject Area 를 활용하면 편리한 점이 많이 있다. Subject Area 는 업무적으로 관련이 있거나 혹은 개발자가 보고자 하는 내용만을 가지고 새로운 화면을 구성한다. 그러므로 좀더 편리하게 엔티티와 관계를 확인할 수 있으며 Subject Area 에서 어떠한 내용을 변경한다 하더라도 이 변경 사항이 전체 ERD 에 반영됨으로 보다 편리하게 작업할 수 있다. 그리고 또 한가지 특징은 나중에 데이터베이스 스키마를 생성할 때 Subject Area 별로 스키마를 데이터베이스에 생성할 수 있다는 점이다. (새로운 엔티티를 데이터베이스에 추가하고자 하는 경우 편리하게 사용할 수 있다.)
Subject Area 를 생성하거나 관리하려면 ERwin Toolbar 의 Create Subject Area 버튼과 Model Explorer 의 Subject Areas 탭 그리고 Model 메뉴 / Subject Areas 메뉴에서 작업할 수 있다.

ERwin Toolbar 에서 Create Subject Area 버튼을 누르면 다음과 같이 Subject Areas 대화상자가 나타나게 되는데 현재 <Main Subject Area>가 리스트에 등록되어진 모습을 볼 수 있다. Subject Areas 대화상자 에서 Members 탭을 누르면 여태까지 작성한 엔티티들이 모두 포함되어 있는 모습을 볼 수 있다. 이번 예제에서는 부서, 사원, 신체정보 엔티티를 포함하는 '사원정보' Subject Area 와 공급업체, 상품, 회원, 납품, 판매 엔티티를 포함하는 '상품정보' Subject Area 를 만들어보도록 하겠다.



위 Subject Areas 대화상자에서 New 버튼을 누른 후 새로운 Subject Area 의 이름을 '사원정보'라고 입력한 후 Members 탭을 선택한다.

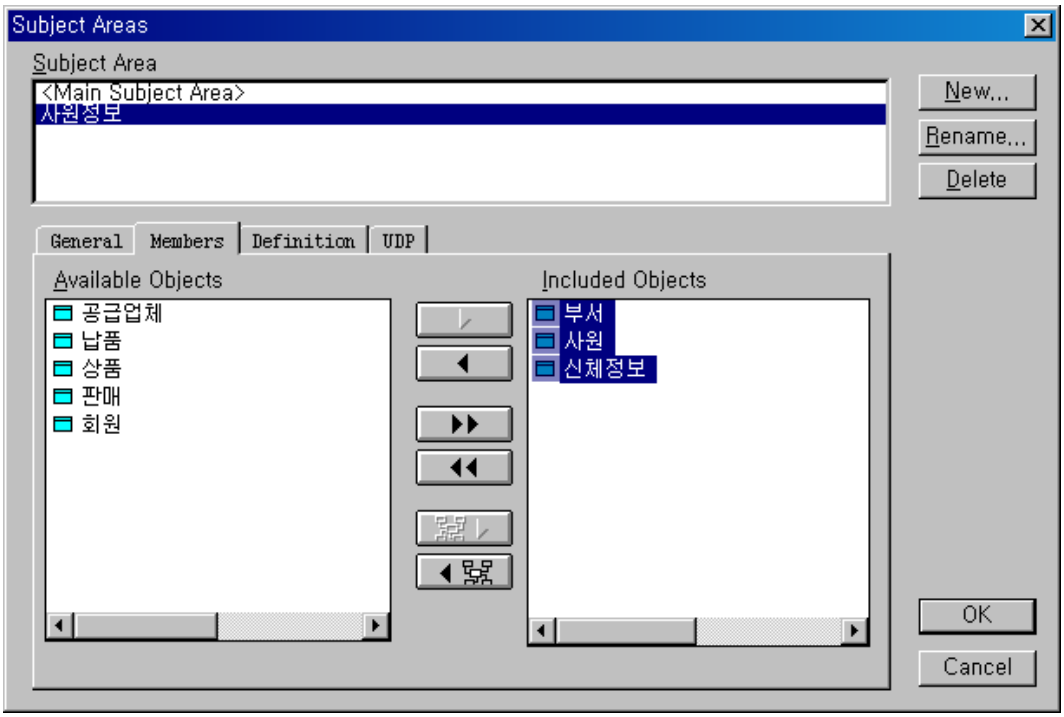
그런 다음 Available Objects 리스트 상자에서 원하는 엔티티를 선택해서 화살표 버튼을 이용하여 Included Objects 리스트 상자로 옮길 수 있는데 가운데 밑에서 두번째 버튼을 활용하면 좀 더 편리하게 관계를 맺고있는 엔티티들을 추가할 수 있다.



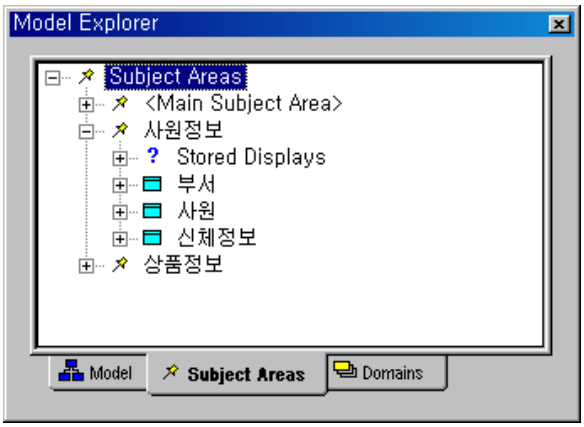
위의 화면처럼 사원 엔티티를 선택하면 밑에서 두번째 버튼이 활성화 되어지는데 그 버튼을 누르면 다음과 같이 Spanning Neighborhood 대화상자가 나타나는데 여기서 선택한 엔티티를 중심으로 관계를 맺고 있는 부모와 자식 엔티티를 단계별로 추가할 수 있다.

여기서는 사원 엔티티를 중심으로 부서와 신체정보 엔티티가 1 단계 걸쳐서 정의되어 있으므로 Level 1 로 선택한 후 확인버튼을 누르면 아래의 그림처럼 사원을 중심으로 부서와 신체정보 엔티티가 '사원정보' Subject Area 에 포함되어 있는 것을 확인할 수 있다.

그런 다음 'OK'버튼을 누르면 부서, 사원, 신체정보 엔티티들만 화면에 보여지게 된다.



'상품정보' Subjcect Area 는 독자 스스로 만들어 보도록 하자.



Subject Area 는 편하게 원하는 엔티티들만 보여주지만 정말 중요한 기능은 Physical Modeling 단계에서 Subject Area 별로 DB 스키마를 생성할 수 있다는 점이다.

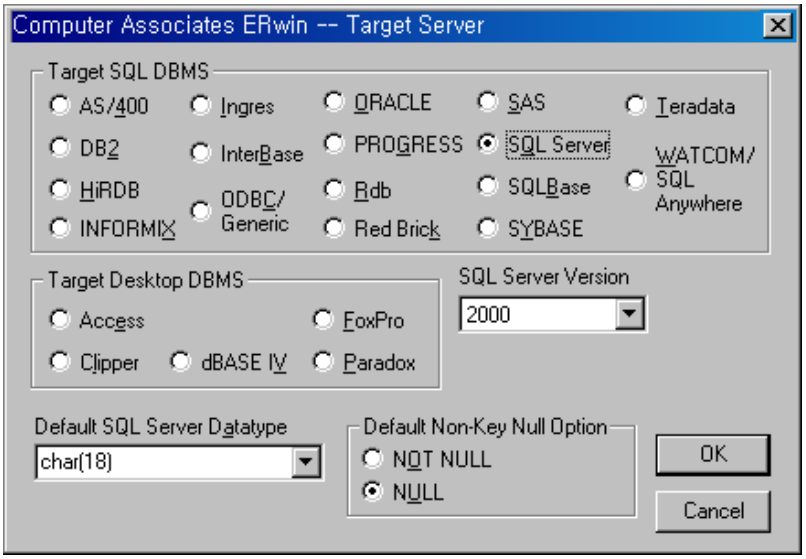
예를 들어 기존에 만들어진 데이터베이스 스키마에 새로운 테이블만 생성하려고 할 때 생성하고자 하는 엔티티들만 Subject Area 로 구성한 후 Generate 할 수 있다.

-. Physical 모델링 (이전에 보기 옵션)

ERwin 에서 물리적 모델링으로 전환하려면 앞에서 설명했던 것 처럼 ERwin Toolbar 의 오른쪽 콤포박스를 Physical 로 선택하면 된다.

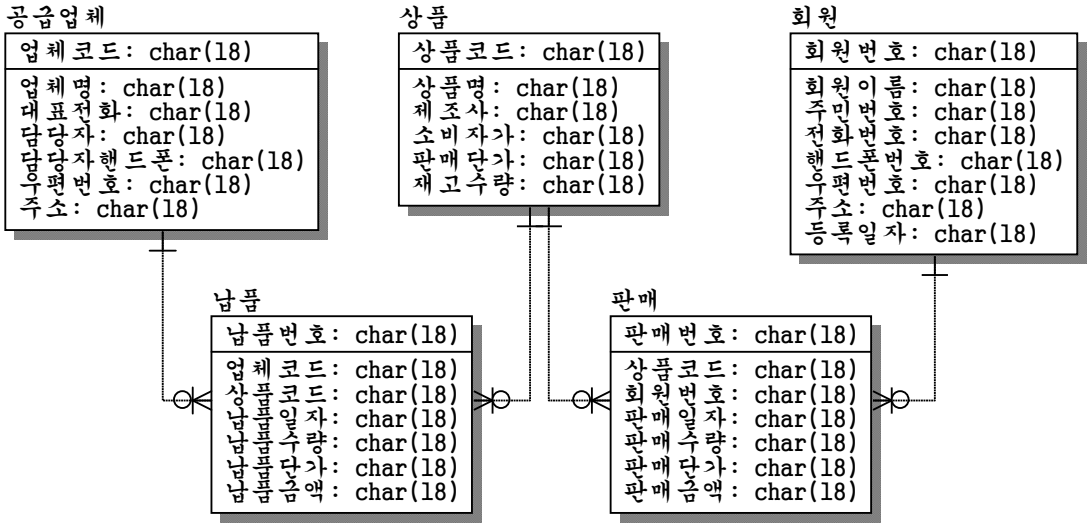
ERwin 을 처음 실행할 때 Target Database 를 SQL Server 2000 으로 선택했으므로 다시 데이터베이스를 선택할 필요는 없다. 만약 처음에 선택하지 않고 지나쳤다면 Database 메뉴에서 Choose Database..메뉴를 선택하면 다음과 같이 개발 DBMS 를 선택하는 Target Server 대화상자가 나오게 되는데 여기서 원하는 RDBMS 와 Version 을 선택하면 된다. (Database 메뉴는 Physical 에서만 나타난다.)

물리적 모델링으로 전환할 경우 용어가 엔티티(Entity)에서 테이블(Table)로 속성(Attribute)에서 컬럼(Column)으로 바뀌게 된다.

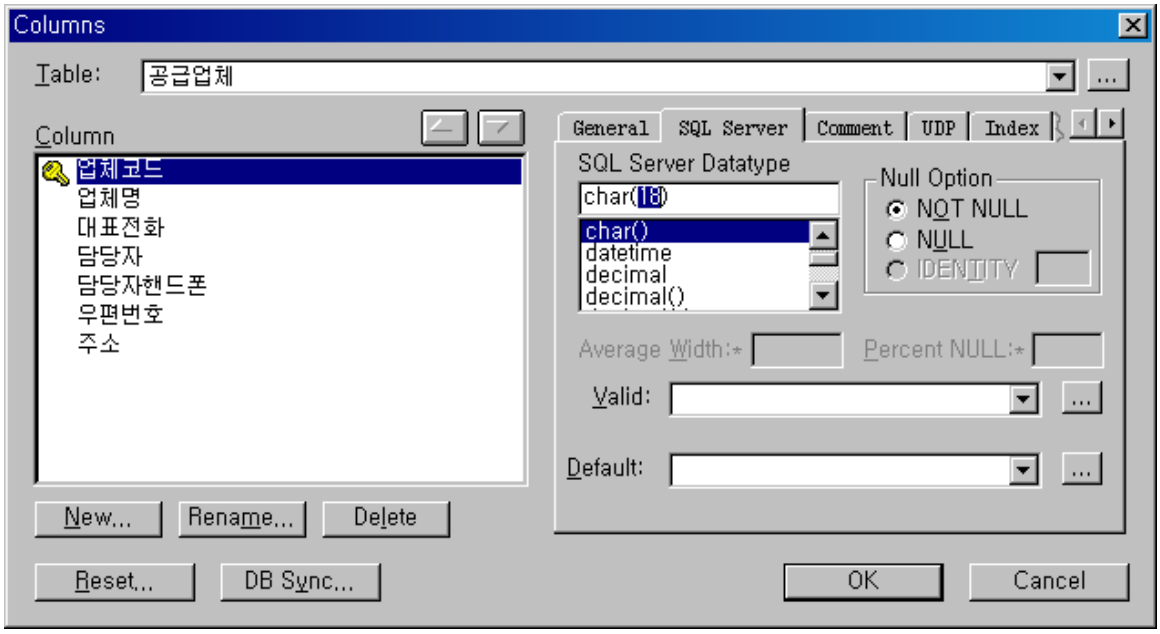


Logical 모델링에서 Physical 모델링으로 전환하면 기본적으로 다음과 같이 컬럼 Data Type 과 Size 가 함께 보여지게 된다.

-. Column Data Type 과 Size

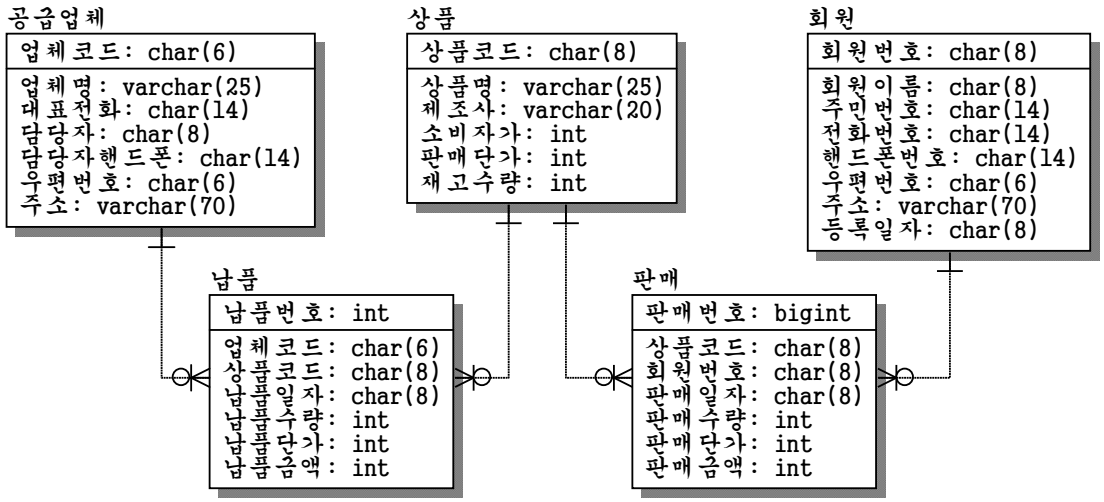


기본적으로 char(18)로 모든 Data Type 과 Size 가 정의되어 있는데 이를 각 컬럼에 입력될 데이터의 성격에 따라 적절하게 변경해 주어야 한다.
컬럼의 Data Type 과 Size 를 정의하려면 테이블을 선택한 뒤 오른쪽 버튼을 누르고 팝업메뉴에서 Columns..메뉴를 선택하면 다음과 같이 Columns 대화상자가 나타나게 된다.



위 Columns 대화상자 왼쪽에서 컬럼을 선택하고 오른쪽에서 Data Type 과 Size 를 정의하면 되고 이 대화상자에서 컬럼의 Null 허용여부, IDENTITY 속성설정, Rule 과 Check, Default 등을 정의할 수 있으며 인덱스도 정의할 수 있다.

그러면 다음과 같이 컬럼의 Data Type 과 Size 를 정의해 보도록 하자.

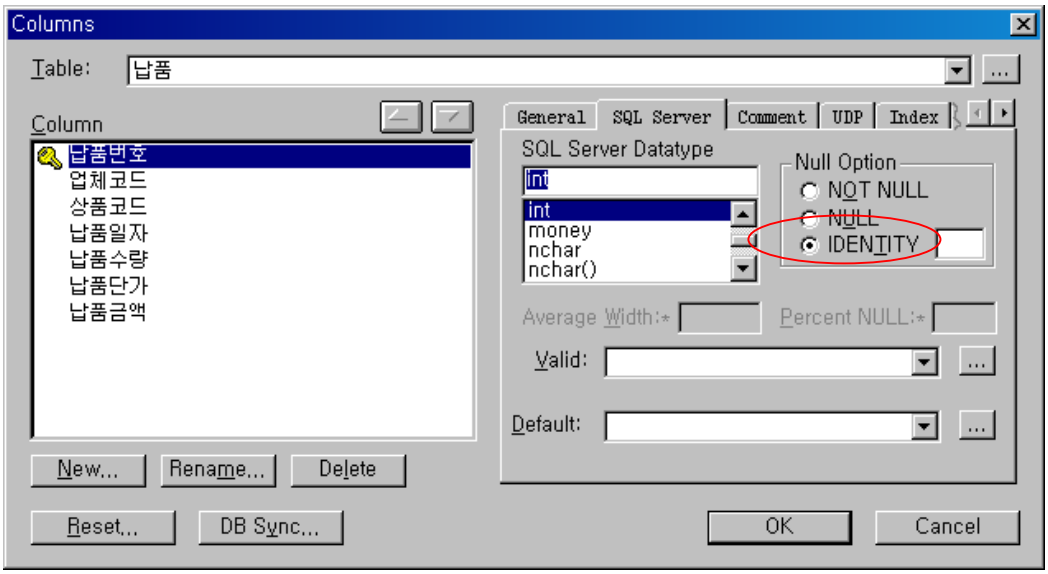


(임의적인 상황을 가정해서 정의했으므로 모든 유형에서 위의 데이터 형식이 항상 올바르다고 할 수는 없다.)

-. Identity, Null Option

위의 예에서 납품 테이블의 납품번호 컬럼과 판매 테이블의 판매번호 컬럼은 Identity 컬럼이다. (Identity 컬럼을 정의하기 위해서는 컬럼의 Data Type 이 정수형 Data Type 이어야만 한다.)

ERwin 에서 Identity 컬럼을 정의하기 위해서는 Columns 대화상자에서 정의하는데 납품 테이블을 더블클릭 하던지 아니면 납품 테이블에서 오른쪽 버튼을 눌러 팝업 메뉴에서 Columns 메뉴를 선택해서 Columns 대화상자를 연 다음 납품번호를 선택한 후 Identity 옵션버튼을 선택하면 된다.

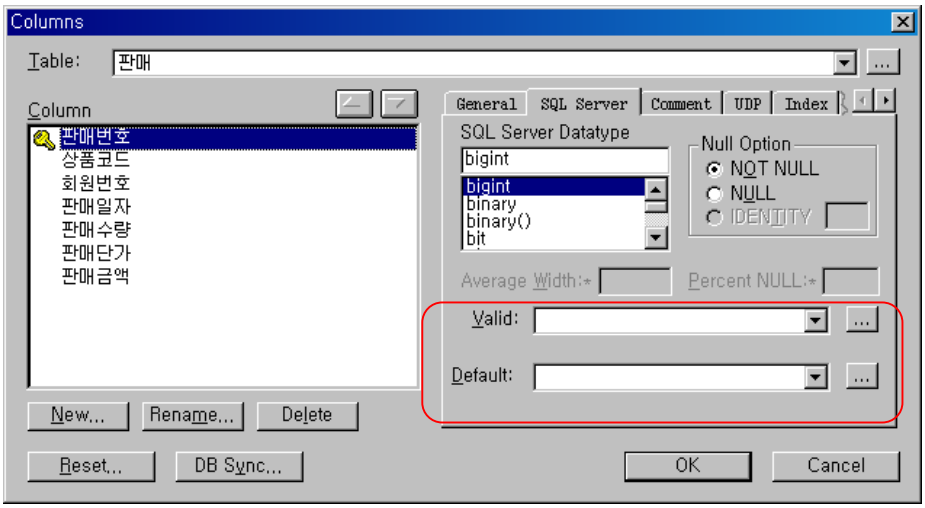


이렇게 Identity 컬럼을 정의할 경우 기본값과 증가 값은 1 이다. 만일 Identity 컬럼의 초기값과 증가 값을 임의로 정의하고자 한다면 Identity 옵션버튼의 뒤쪽 입력 상자에 초기값과 증가 값을 컴마로 구분해서 정의해 주어야 한다. (초기값이 1 이고 증가값이 10 일 경우 입력 상자에 1, 10 으로 정의한다.)
Identity 컬럼에는 널값이 입력될 수 없으므로 Null 옵션이 의미가 없으며 사용자가 임의로 값을 입력할 수도 없다.

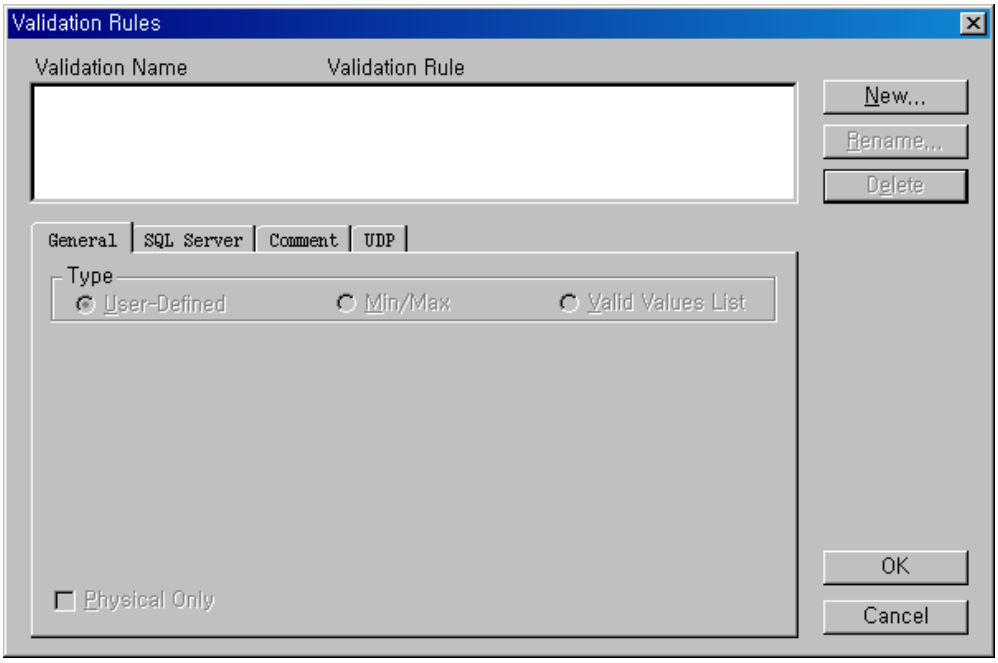
- 참고 : 판매 테이블의 판매 번호는 Data Type 이 Bigint 형이다. Bigint 는 정수형 Data Type 으로 – 9,223,372,036,854,775,808 부터 9,223,372,036,854,775,807 까지의 엄청나게 큰 정수형 Data Type 이다. 실제로 SQL Server 에서는 Bigint Data Type 도 Identity 컬럼으로 정의할 수 있으나 ERwin 4.0 에서는 Int 까지만 Identity 컬럼을 지원하고 Bigint Data Type 은 Identity 속성을 정의할 수 없도록 비활성화 되어 있다. 아마도 패치 버전에서는 이를 지원하지 않을까 생각 한다.

-. Check(Rule), Default

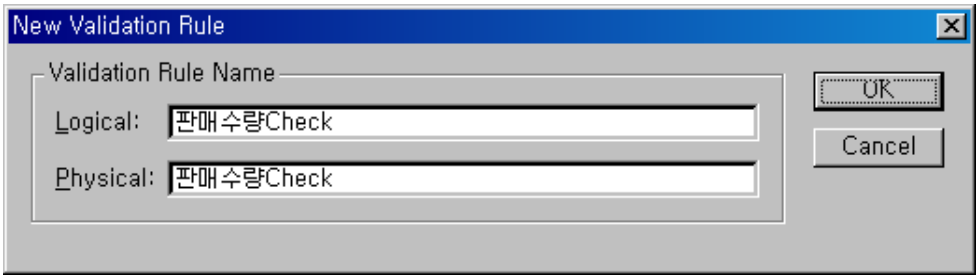
Check 제약 조건은 테이블을 만들거나 수정하면서 정의하는 제약조건이고 Rule 은 데이터베이스 내의 Object 로 우선 데이터베이스 내에 Rule 이라는 Object 를 만든 후에 이를 필요한 테이블의 컬럼에 바인딩 해서 사용하게 된다. 둘 다 데이터가 컬럼에 들어올 수 있는 경우의 수를 제한해서 데이터베이스의 무결성을 강화하기 위한 방법으로 사용 한다.
Default 또한 마찬가지로 사용자가 특정 컬럼에 데이터를 입력하지 않았을 때 기본적으로 그 컬럼에 들어가지는 값을 정의하는데 사용되는 옵션이다. 이들도 마찬가지로 Columns 대화상자에서 관리한다.

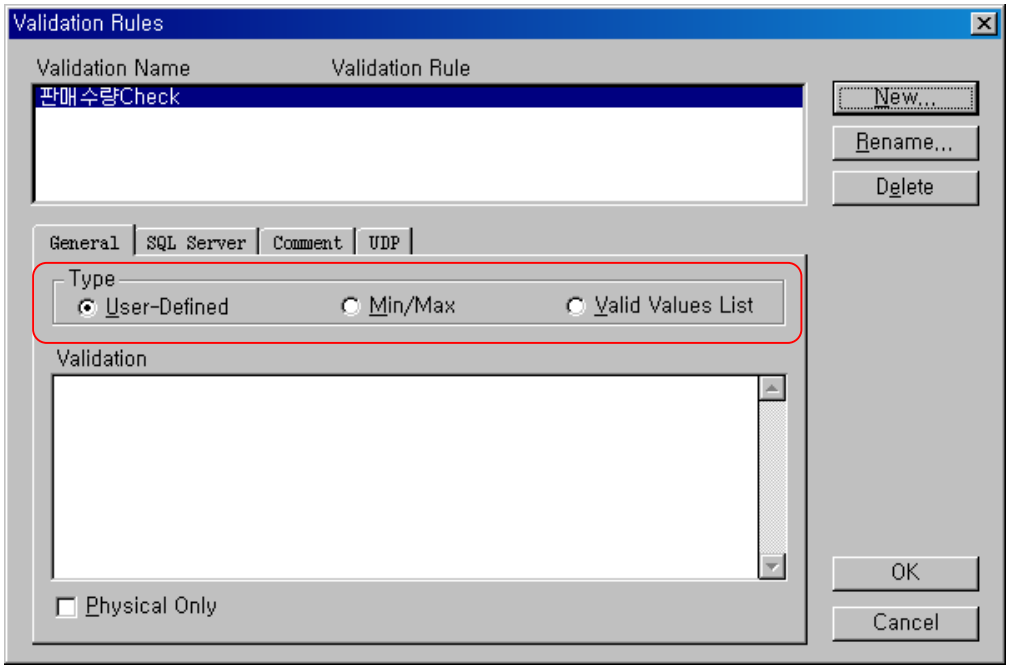


우선 Check(Rule) 제약조건 부터 정리해 보도록 하자.
이번 예에서는 위의 판매 테이블에 판매수량과 판매단가 그리고 판매금액 컬럼에 0 이상의 값이 입력되어야 한다는 내용을 정의해 보도록 하겠다. 그리고 다른 예로 값의 범위를 정하는 예와 특정한 몇 개의 데이터만 입력할 수 있도록 정의하는 내용에 대해서 다루어 보도록 하겠다.
Check(Rule)제약조건을 정의하려면 위의 그림에서 Valid 뒤쪽의 버튼을 클릭하면 다음과 같이 Validation Rules 대화상자가 나타나게 된다.

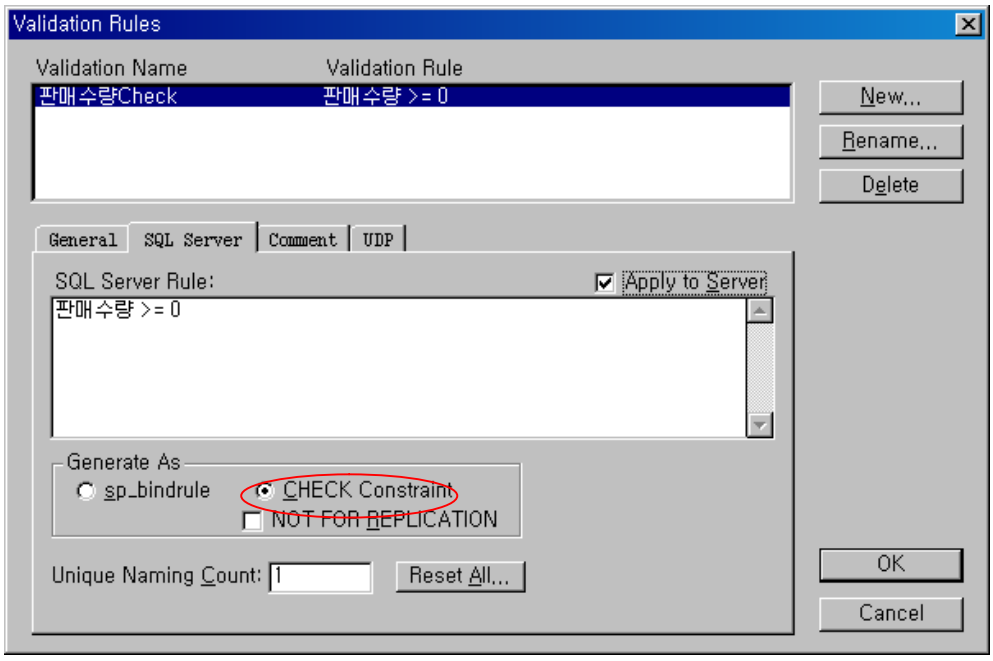


우선 테이블에 판매수량과 판매단가 그리고 판매금액 컬럼에 0 이상의 값이 입력되어야 한다는 내용을 정의하기 위해 New 버튼을 누른 뒤 New Validation Rule 대화상자에서 Logical 과 Physical 입력상자에 '판매수량 Check'이라고 입력한다.





그러면 아래의 Type 이 활성화 되어지는데 여기서 첫 번째 User-Defined 옵션버튼을 선택한 후 Validation 입력상자에 '판매수량 >= 0'이라고 입력한다.

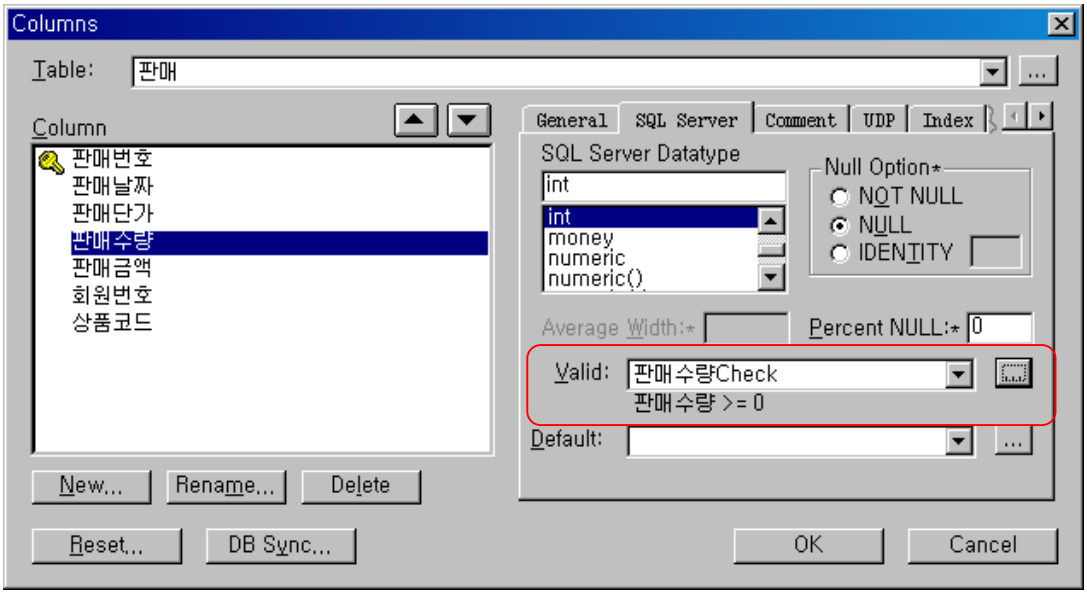


그런 다음 위의 그림처럼 SQL Server 탭을 선택하고 Generate As 영역에서 CHECK Constraint 옵션을 선택한다.

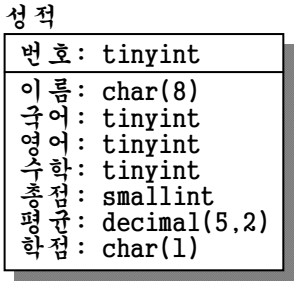
Sp_bindrule 옵션은 Rule 을 만들때 적용하는 옵션이며 위의 예는 판매수량 컬럼에 적용되는 제약 조건을 정의하는 것이므로 CHECK Constraint 옵션을 선택해야 한다.

위와 같은 요령으로 '판매단가 Check', '판매금액 Check'를 만든다.

그런 다음 OK 버튼을 눌러서 Columns 대화상자로 돌아간 뒤 아래의 그림처럼 판매수량 컬럼을 선택한 후 Valid 콤보상자에서 '판매수량 Check'를 선택하고 판매단가 컬럼을 선택한 후 '판매단가 Check'를 판매금액 컬럼을 선택한 후 '판매금액 Check'를 선택해서 각각의 필드에 적합한 Check 제약 조건을 매칭시킨다.



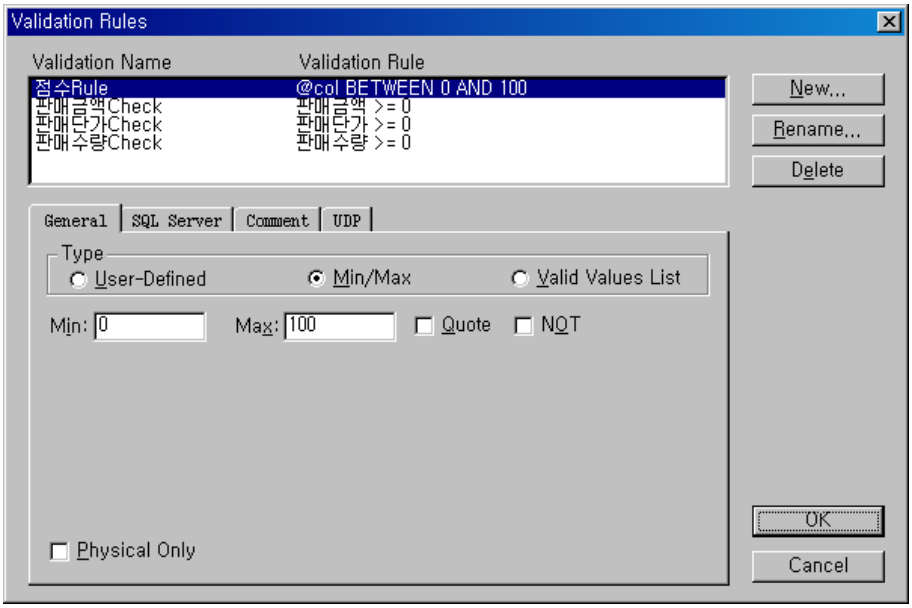
이번에는 값의 범위를 제한하거나 특정한 몇 개의 데이터만이 입력되어야 하는 내용을 정의하는 유형에 대해 알아보도록 하자.
우선 다음과 같은 성적 테이블이 있다고 하자.



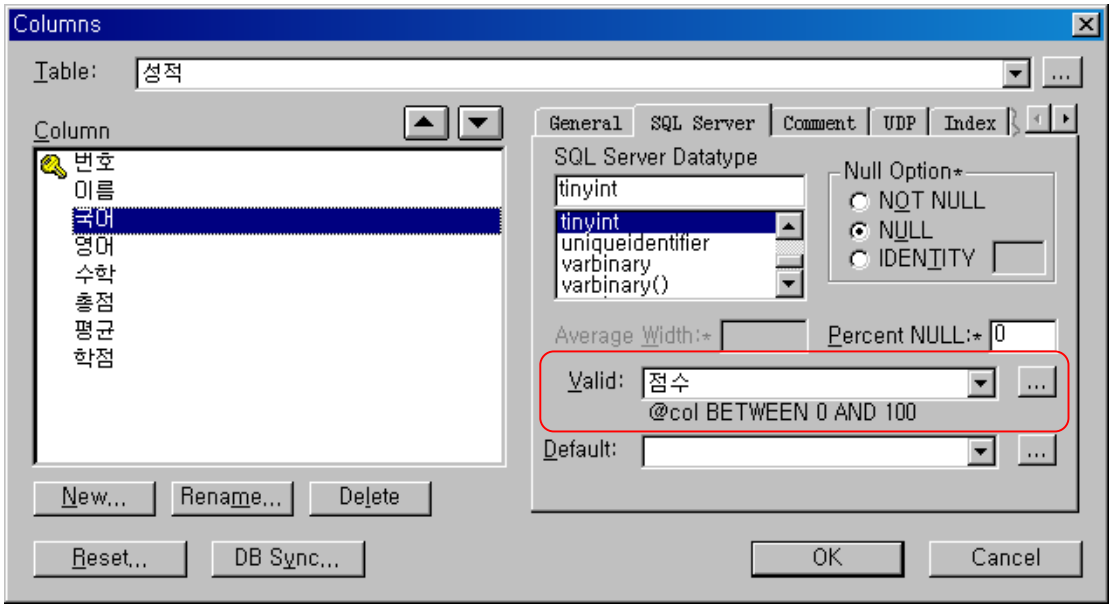
국어, 영어, 수학 컬럼의 경우 입력될 수 있는 데이터의 범위는 0 ~ 100 사이의 정수이다. 그리고 총점과 평균은 국어, 영어, 수학의 합과 평균이므로 계산하면 될 것이고 학점은 평균에 따라 A, B, C, D 그리고 F 만이 입력될 수 있다고 하자.

이를 정의하기 위해서 다시 Validation Rules 대화상자를 띄운 후에 국어, 영어, 수학 컬럼에 입력될 수 있는 값의 범위를 정의하기 위해 New 버튼을 누른 후 '점수 Rule'이라

고 입력한 뒤 두 번째 옵션 버튼인 Min/Max 를 선택한다.
그러면 아래 그림처럼 Min 값과 Max 값을 입력할 수 있는 입력상자가 나타나는데 여기에 0 과 100 을 입력하도록 하자.

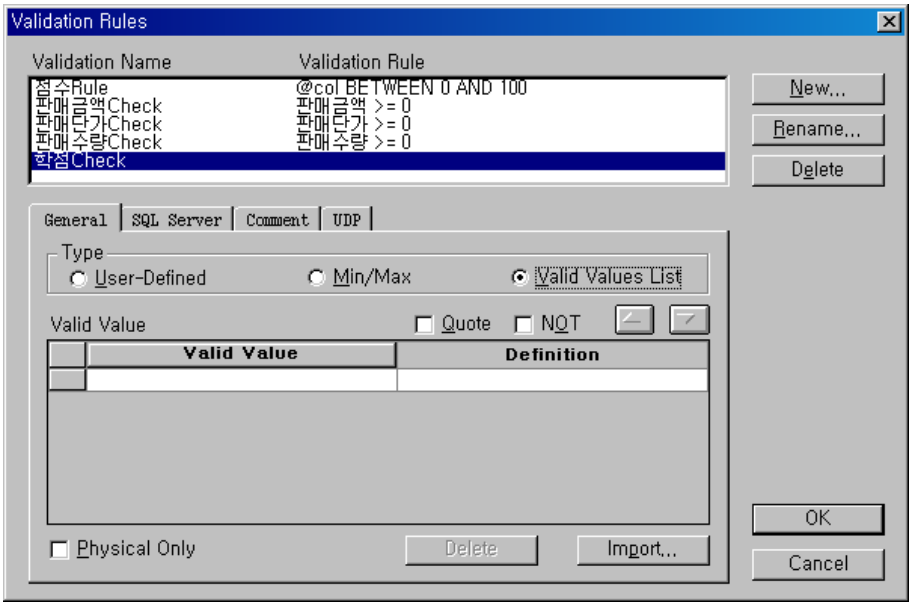


여기서는 숫자형 데이터의 입력 범위를 지정하고 있으므로 'Quote'체크 박스는 선택하지 않으며 더군다나 'NOT'체크 박스도 선택할 필요가 없다.
'점수 Rule'은 국어, 영어, 수학 컬럼에 모두 적용될 제약조건 이므로 Rule 로 만들어서 필요한 컬럼에 바인딩해야 하므로 SQL Server 탭에서 sp_bindrule 옵션을 선택하면 된다.
이렇게 정의하고 'OK'버튼을 누른 뒤 Columns 대화상자에서 국어, 영어, 수학 컬럼의 Valid 콤보상자에서 점수를 선택하면 된다.

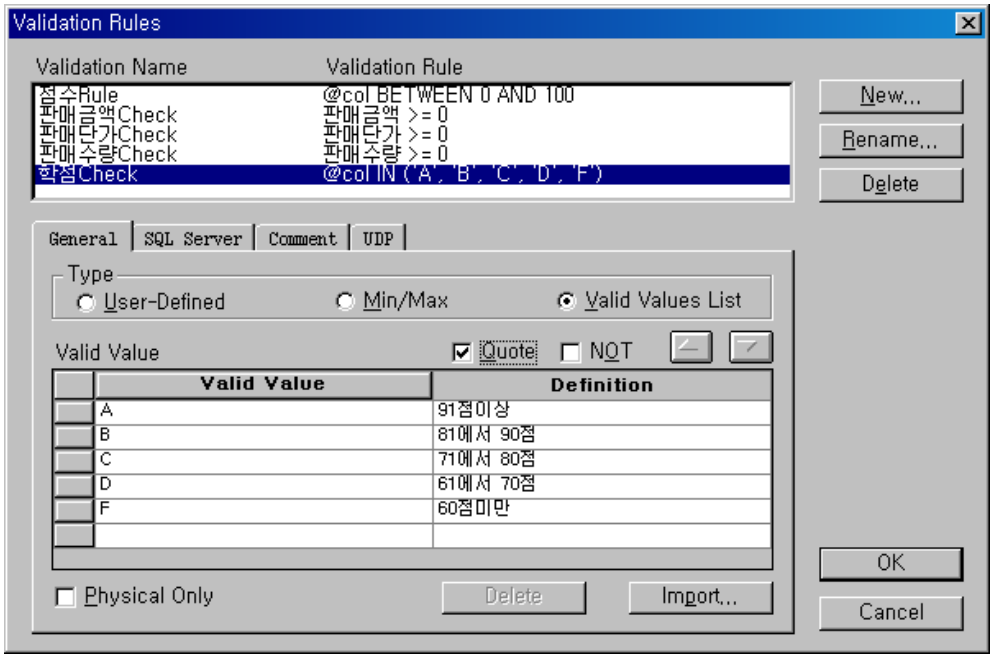


그 다음 학점 필드에 입력될 수 있는 데이터를 정의해 보도록 하자.

다시 Validation Rules 대화상자를 열고 New 버튼을 누른 후 '학점 Check'라고 이름을 입력한 후 이번에는 마지막 옵션 버튼인 Valid Value List 버튼을 선택하면 다음과 같이 값을 입력할 수 있는 그리드가 나타나게 된다.



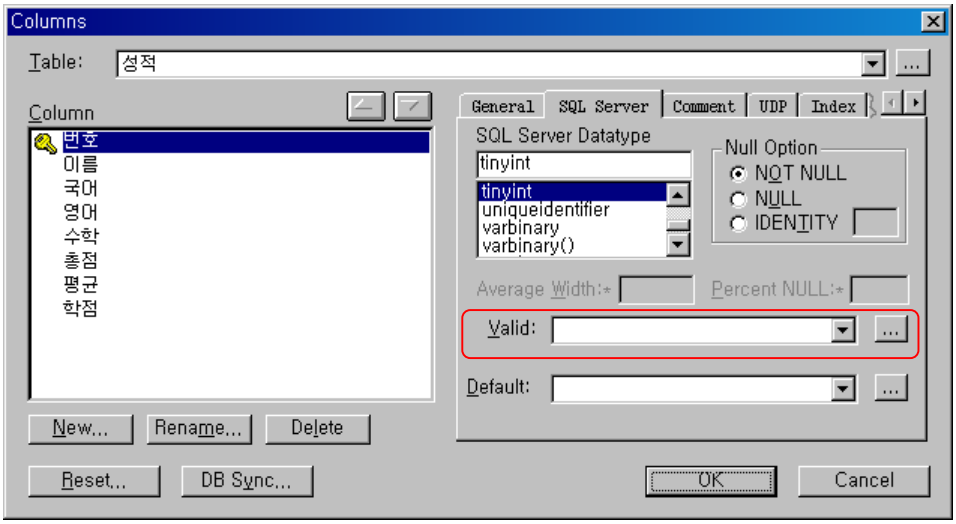
여기의 Valid Value 컬럼에 값의 리스트를 아래와 같이 정의하면 된다.
이는 문자열이기 때문에 Quote 체크박스를 선택하도록 한다. NOT 이라는 옵션을 사용하면 특정 데이터 이외의 값만을 받아들일 수 있도록 정의할 수 도 있을 것이다.



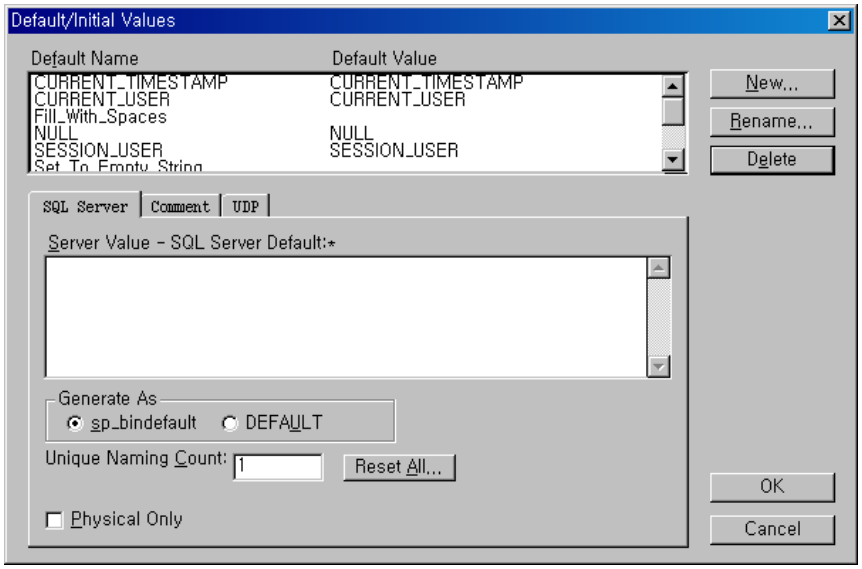
‘학점 Check’역시 학점 컬럼에 적용되는 Check 제약조건이기 때문에 SQL Server 텝에서 CHECK Constraint 옵션을 선택한다. 그런 다음 ‘OK’버튼을 누른 뒤 Columns 대화상자에서 학점 컬럼을 선택하고 Valid 콤보상자에서 ‘학점 Check’ 제약 조건을 선택하면 된다.

이제 Default 값을 정의해 보도록 하자.

위의 성적 테이블에서 국어, 영어, 수학 컬럼에 데이터가 입력되지 않을 때 기본적으로 0 값이 입력될 수 있도록 해야 한다. 이를 정의하기 위해서 이번에는 Columns 대화상자에서 Default 뒤쪽의 버튼을 클릭한다.



그러면 이번에는 Default / initial Values 대화상자가 나타난다. 그런데 Default / initial Values 대화상자는 Validation Rules 대화상자와는 달리 기본적으로 Default 값을 정의할 수 있도록 TIMESTAMP 나 USER 등이 미리 정의되어 있다. 이들을 사용하려면 원하는 컬럼에 바로 Default 값을 지정하면 될 것이다.



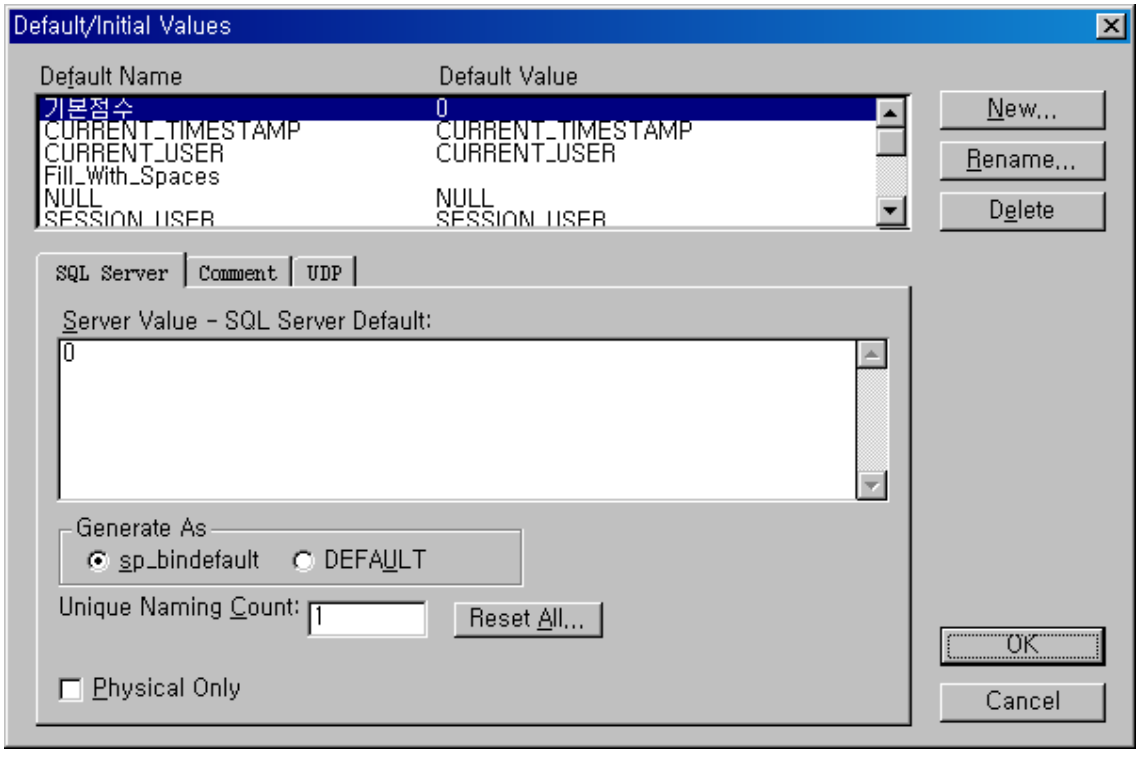
여기서는 기본적으로 국어, 영어, 수학 컬럼에 적용될 Default 값을 새로 정의할 것이므로 New 버튼을 누른 뒤 Default 이름을 '기본점수'라고 입력하고 확인버튼을 눌러 새로운 Default 값을 정의하도록 하자.

Default 를 적용시키는 방법도 두 가지가 있는데 하나는 Default 를 데이터베이스 안에 개체(Object)로 생성한 뒤 여러 컬럼에 바인딩(Binding)하는 방법이고 다른 한가지는 일반적으로 사용하는 방법으로 테이블을 만들거나 수정할 때 직접 컬럼에 Default 속성을 정의하는 방법이다. (전자를 절차적 방법이라고 하고 후자를 서술적 방법이라고 한다.)

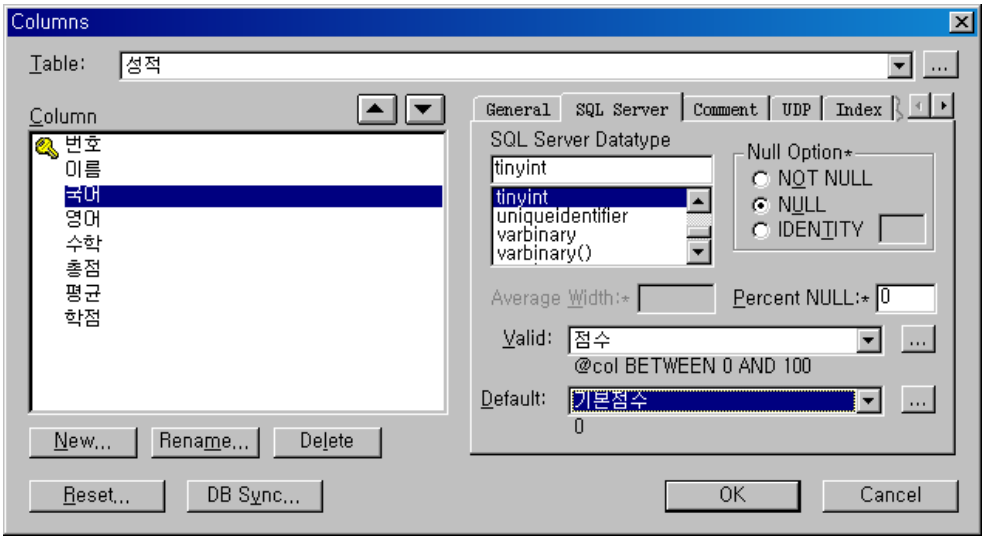
이러한 내용을 구현하기 위해서는 Default / initial Values 대화상자에서 Generate As 내에 옵션을 선택해서 구현할 수 있다.

전자가 sp_Bindefault 옵션이고 후자가 Default 옵션이다.

여기서는 어떠한 방법으로 정의한다 해도 무방하지만 여러 테이블의 여러 컬럼에 동일한 Default 속성을 적용시키고자 한다면 절차적 방법을 통해 정의하는 것이 바람직하므로 절차적 방법의 Default 를 정의해 보도록 하겠다.
우선 New 버튼을 누른 뒤 Default 이름에 '기본점수'를 입력하고 확인버튼을 누른 뒤 아래 처럼 값은 0 을 입력하고 sp_Bindfault 를 선택한다.



그런 다음 국어, 영어, 수학 필드에 Default 를 기본점수로 선택해주면 된다.

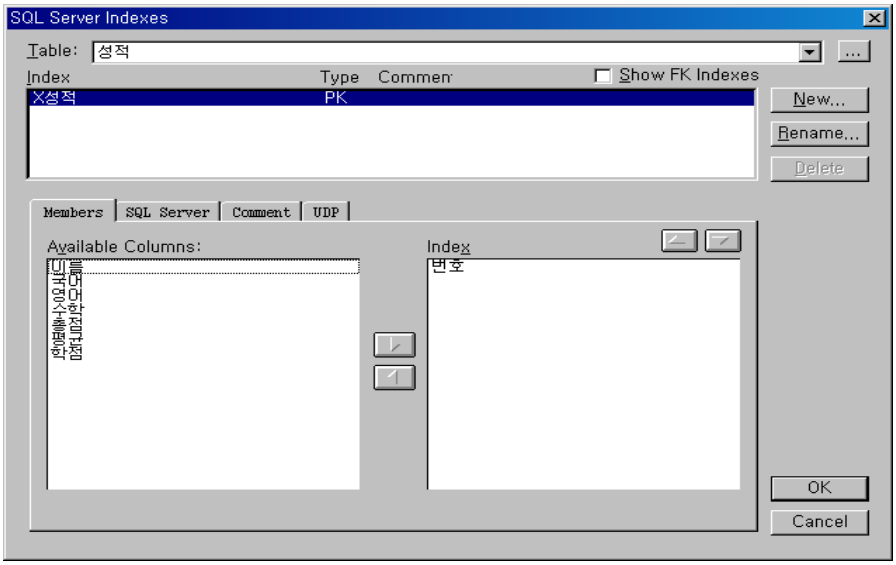


-. 인덱스 정의 방법

인덱스에 대한 개념과 이론은 위 (몇 장)에서 설명했으므로 여기서는 부가적인 설명은 제외하고 기능적인 부분에 대해서 적용해보도록 하겠다.

ERwin 에서 인덱스를 정의하려면 Physical 모델링에서 테이블을 선택한 뒤 오른쪽 버튼을 누른 후 팝업 메뉴에서 Indexes 메뉴를 선택하면 된다. (물론 Columns 대화 상자의 Index 탭에서 정의할 수도 있고 Model 메뉴의 Indexes 메뉴를 선택해도 된다.)

이번 예제는 성적 테이블을 가지고 인덱스를 적용시켜 보도록 하겠다. 우선 성적 테이블을 선택한 후 오른쪽 버튼을 누른 후 팝업 메뉴에서 Indexes 를 선택하면 다음과 같이 SQL Server Index 대화상자가 나타나게 된다.

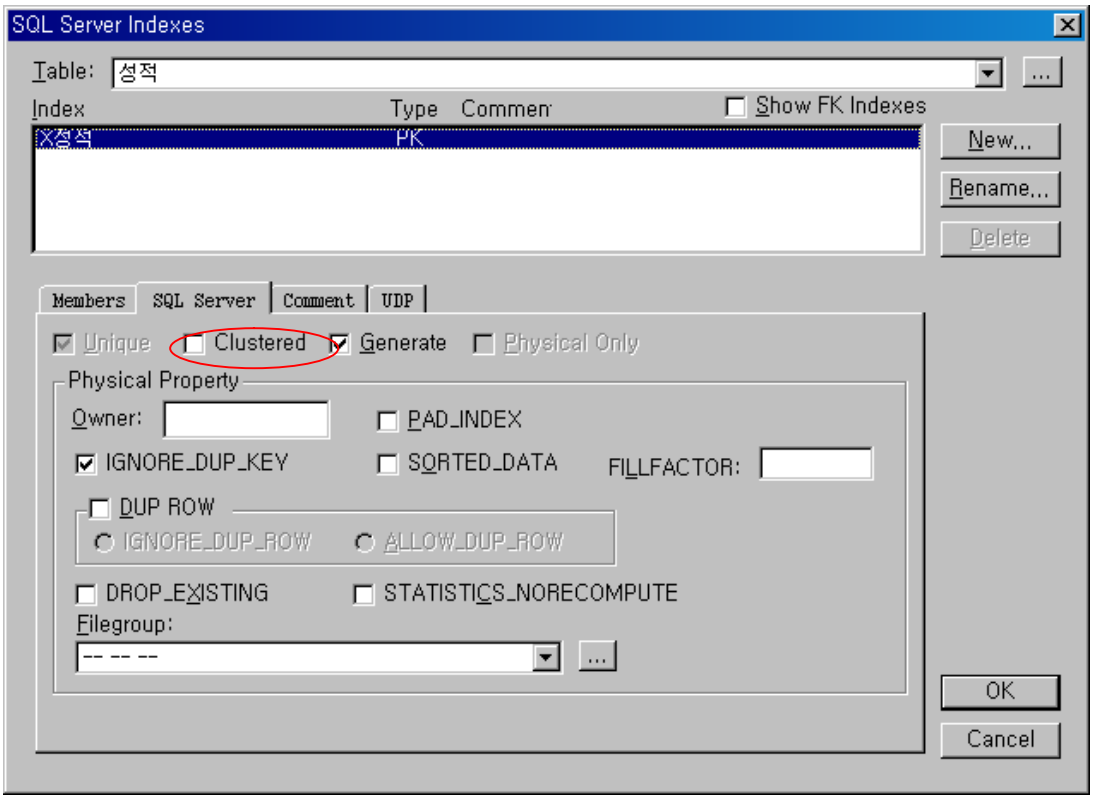


앞장에서 설명 했듯이 현재 우리는 성적 테이블에서 어떠한 인덱스도 정의한 적이 없지만 이미 하나의 인덱스가 만들어져 있는 모습을 볼 수 가 있다.
이 인덱스는 기본키에 의해 자동으로 만들어진 인덱스이다.

그 의미는 인덱스의 이름은 'X 성적'이고 인덱스의 유형은 'PK' (기본키) 인덱스이며 인덱스에 포함된 컬럼은 '번호'컬럼 이라는 뜻이다.

그런데 여기서 한가지 주의해야 할 내용이 있는데 ERwin 에서 인덱스를 정의할 때 기본키 컬럼에 기본적으로 넌 클러스터드 인덱스가 적용된다는 점이다.

물론 클러스터드 인덱스로 만들 수도 있는데 그 옵션은 아래와 같이 SQL Server Indexes 대화 상자의 SQL Server 텝에서 확인해 볼 수 있다.
(기본적으로 기본키 인덱스는 Unique 속성에 선택된 상태에서 바꿀 수 없게끔 비활성화 되어 있다.)



성적 테이블에서 만약 번호 컬럼이 클러스터드 인덱스로 적용된다면 어떻겠는가?
클러스터드 인덱스는 범위 쿼리(Range Query)할 때에 뛰어난 성능을 나타낸다고 했다.
그렇다면 번호를 기준으로 범위 쿼리를 할 경우가 얼마나 되는가?
1 번부터 10 번까지의 성적을 조회한다던가 아니면 20 번부터 35 번까지의 성적을 조회하는 일이 아마도 별로 많지는 않을 것이다.

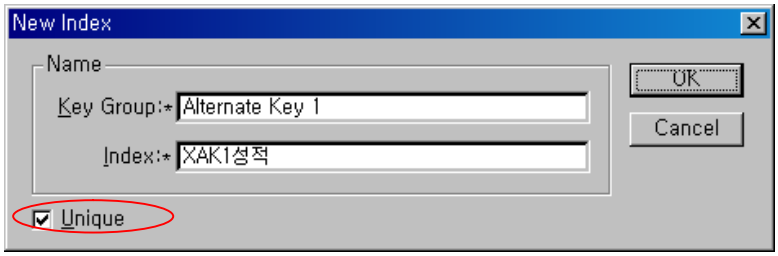
오히려 총점이나 평균 혹은 학점을 기준으로 범위 쿼리를 하는 경우가 많을 것이다. 그러므로 번호 컬럼에는 넌 클러스터드 인덱스가 적합하며 총점과 평균, 학점 컬럼 중에는 업무적인 프로세서를 살펴본 후 클러스터드 인덱스를 정의해야 할 것이다. 여기서는 학점 컬럼에 클러스터드 인덱스를 적용해 보도록 하겠다.

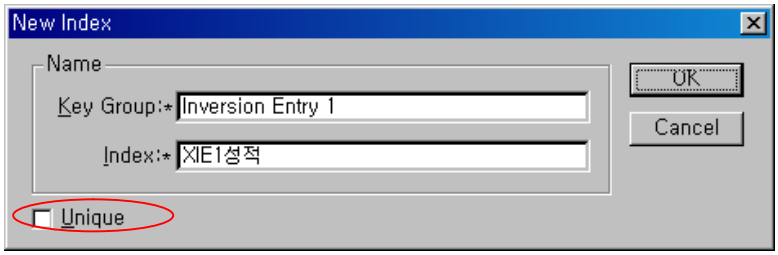
새로운 인덱스를 만들려면 SQL Server Indexes 대화상자에서 New 버튼을 누른다.

그러면 아래의 그림 처럼 New Index 대화 상자가 나타나게 되는데 여기서 중요한 옵션이 바로 Unique 옵션이다.

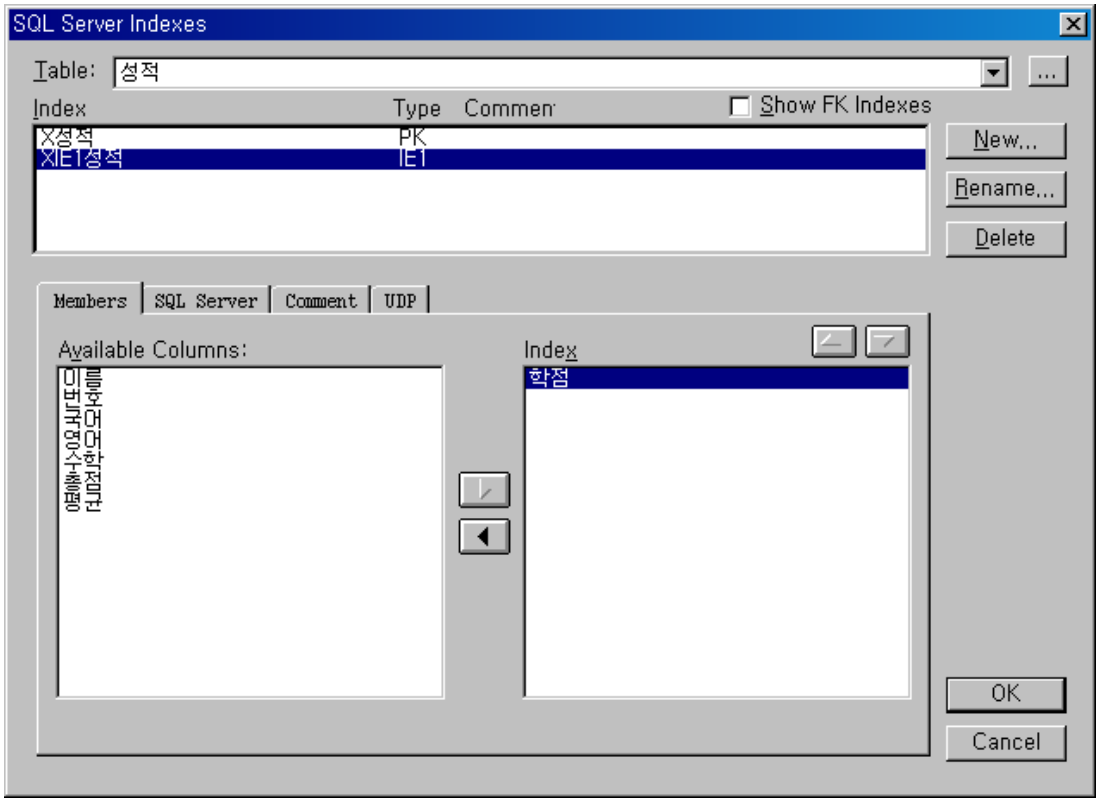
기본적으로 이 옵션은 선택되어 있으며 인덱스로 정의할 컬럼의 데이터가 Unique 하다면 선택을 해주어야 할 것이다.

만일 Unique 옵션을 선택하지 않으면 Key Group 의 이름이 'Invension Entry'로 바뀌는 것을 확인할 수 있는데 이는 데이터가 Unique 하지는 않지만 자주 액세스 되는 컬럼에 인덱스를 정의할 때 사용하는 용어이다.

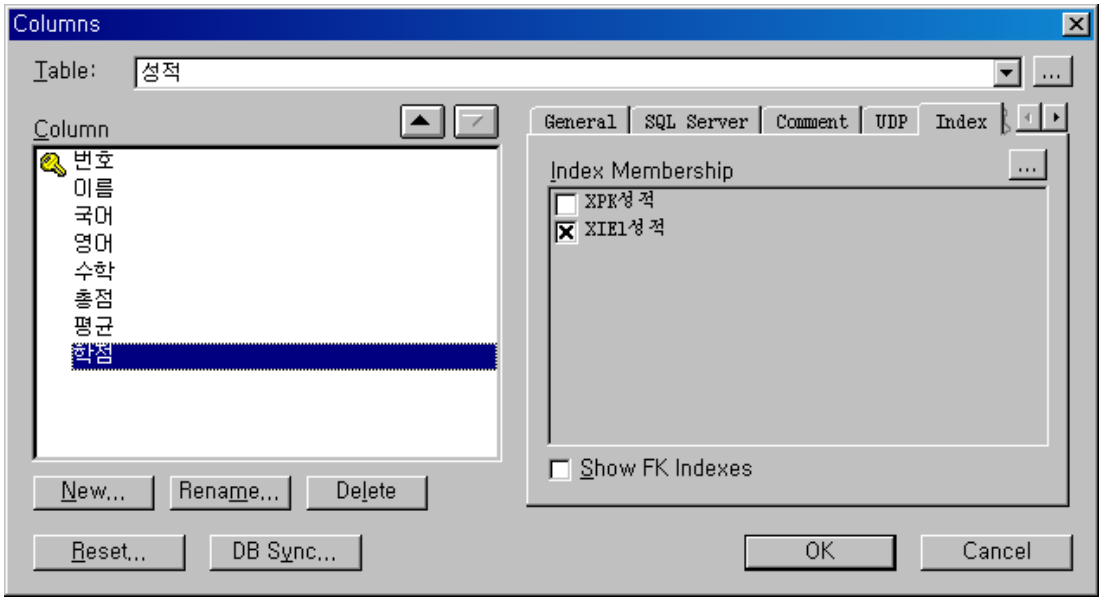




우리가 지금 인덱스를 정의할 학점 컬럼은 데이터가 Unique 하지 않으므로 두번째 그림처럼 Unique 옵션을 해제한 후 OK 버튼을 누른다.



그러면 위의 그림처럼 새로운 인덱스가 만들어 지게 되는데 Members 탭에서 학점컬럼을 선택한 후 추가 화살표 버튼을 눌러 오른쪽 Index 리스트 상자로 옮긴 후에 SQL Server 탭에서 Clustered 체크박스를 선택하면 될 것이다.



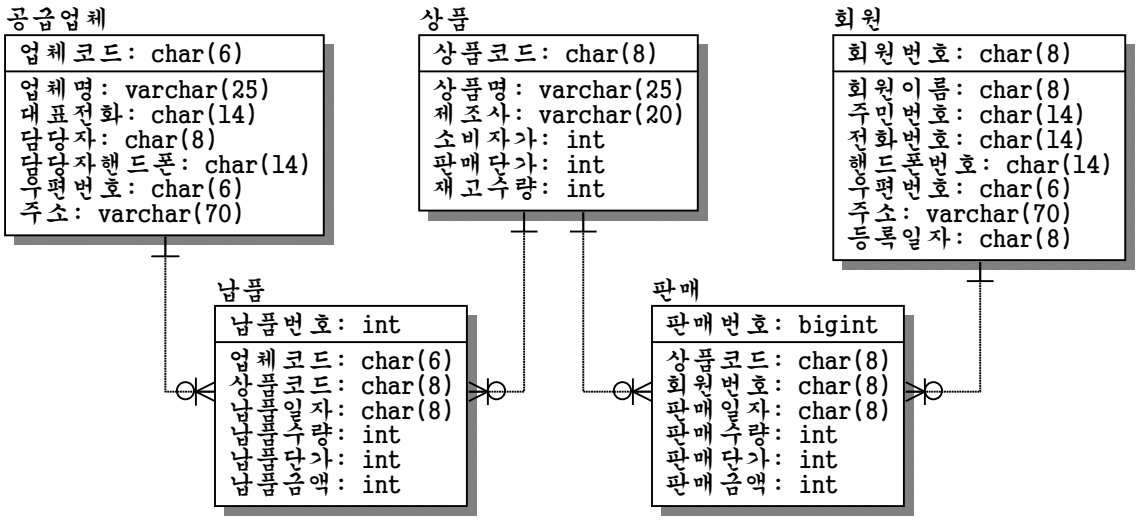
이렇게 만들어진 인덱스는 Columns 대화상자의 Index 탭에서도 확인할 수 있다.

-. 트리거(Trigger) 정의 방법

트리거에 대한 개념과 이론은 위 (몇 장)에서 설명했으므로 여기서는 부가적인 설명은 제외하고 기능적인 부분에 대해서 적용해보도록 하겠다.

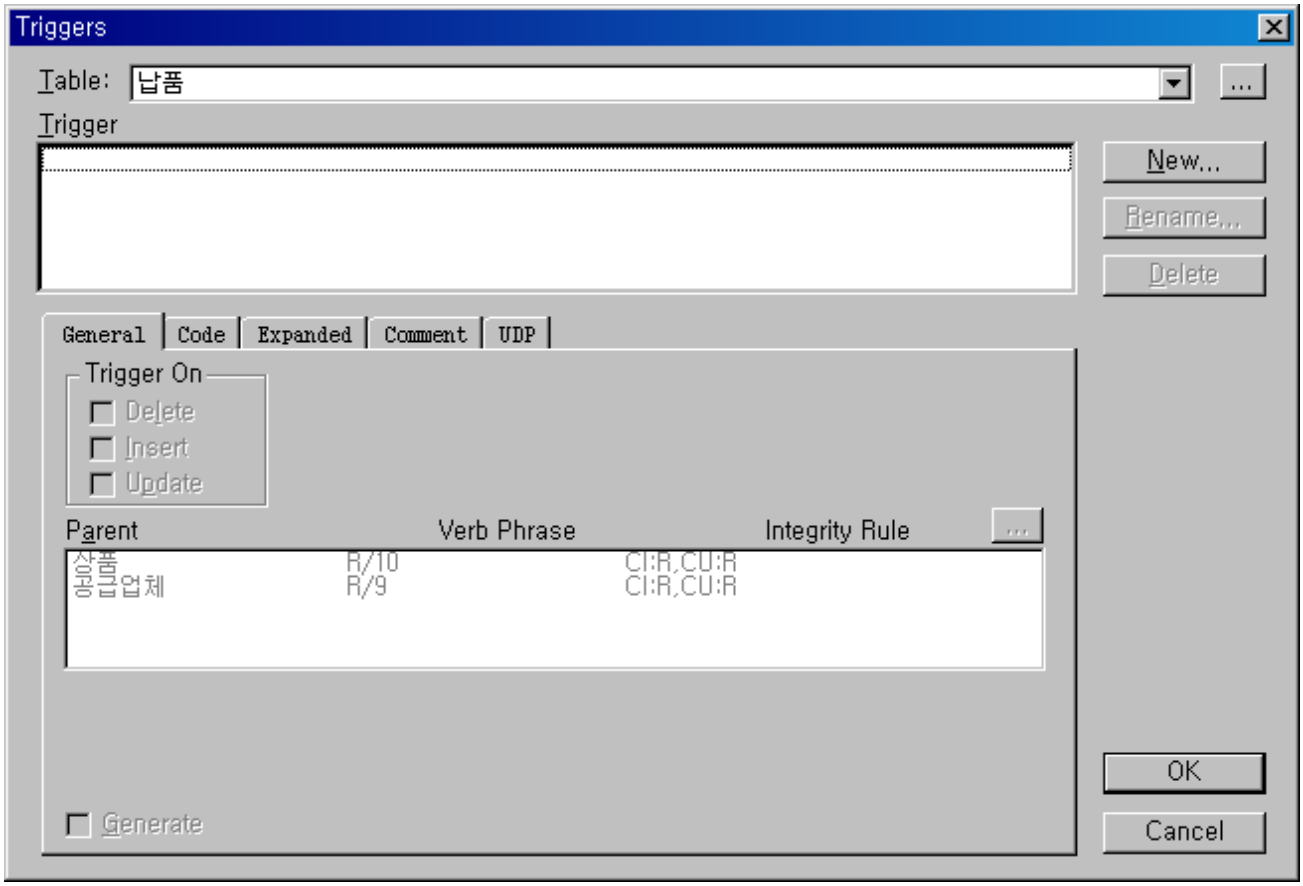
ERwin 에서 트리거를 정의하려면 Physical 모델링에서 테이블을 선택한 뒤 오른쪽 버튼을 누른 후 팝업 메뉴에서 Triggers 메뉴를 선택하면 된다.

이번 트리거에 대한 예제는 앞에서 다대다 관계를 해소하기 위해서 사용했던 예제를 통해 트리거를 정의해 보도록 하겠다.



기본적인 로직은 앞서 트리거를 설명할 때 했던 내용이므로 설명을 하지 않아도 될 것이다. 여기서는 납품 테이블에 데이터가 입력 되어질 때 상품 테이블의 재고 수량에 더해지는 트리거만 만들어 보도록 하겠다.

납품 테이블에서 오른쪽 버튼을 누른 뒤 팝업 메뉴에서 Triggers 메뉴를 선택하면 다음과 같이 Triggers 대화상자가 나타나게 된다.



지금 현재 납품 테이블은 공급업체 테이블, 상품 테이블과 관계를 맺고 있으며 참조 무결성과 관련한 내용이 기본적으로 적용되어 있는 모습을 볼 수 있다.

여기서 New 버튼을 누른 뒤 트리거의 이름을 '납품 INS'로 정의하고 확인 버튼을 누른다.

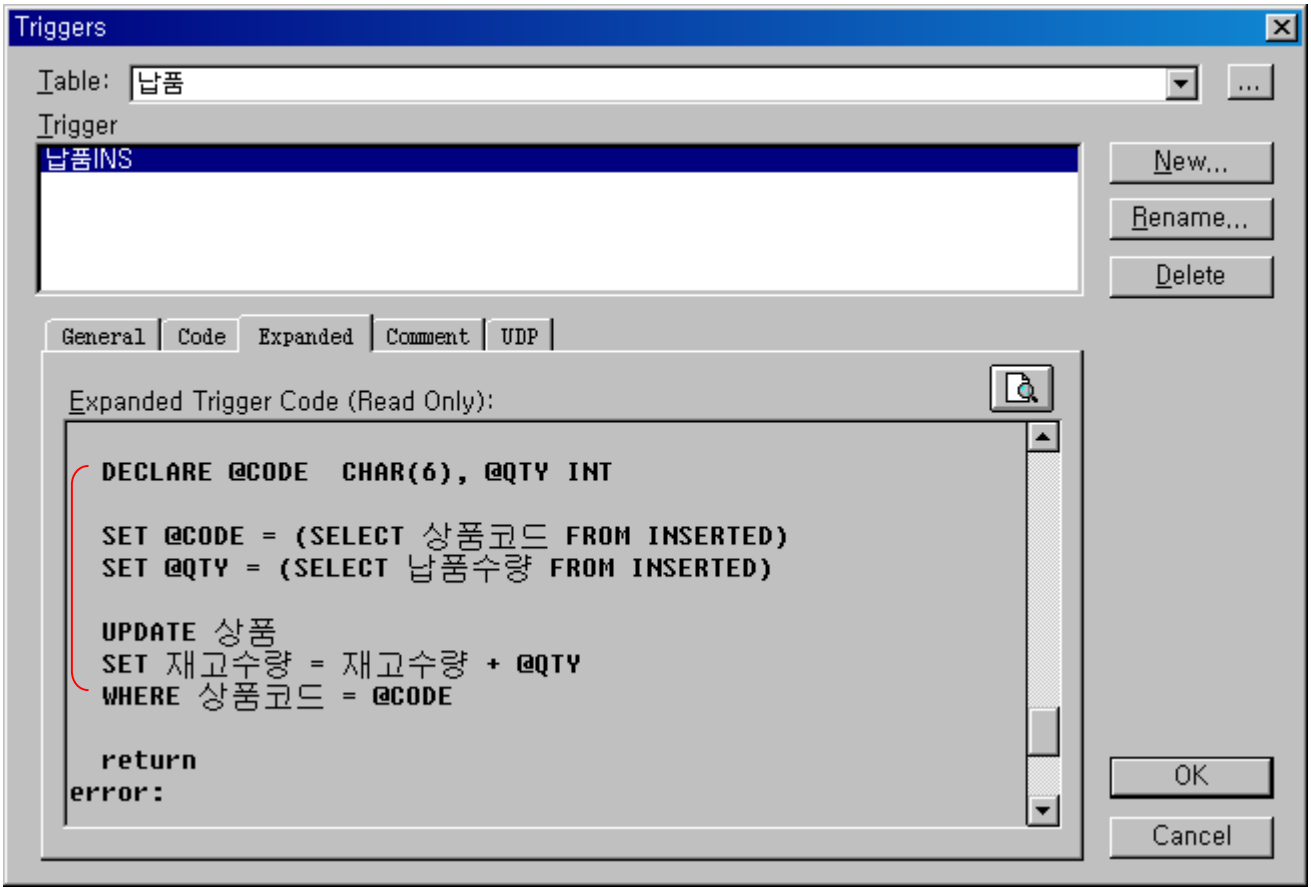
그런 다음 Code 탭으로 전환한 후 다음과 같이 트리거를 정의하는 코딩을 Trigger Code 입력 상자의 마지막 부분 Return 문장 바로 위에 다음 코트를에 추가로 삽입한다.

```
DECLARE @CODE    CHAR(6), @QTY INT

SET @CODE = (SELECT  상품코드  FROM INSERTED)
SET @QTY = (SELECT  납품수량  FROM INSERTED)

UPDATE  상품
SET  재고수량  =  재고수량  + @QTY
WHERE  상품코드  = @CODE
```

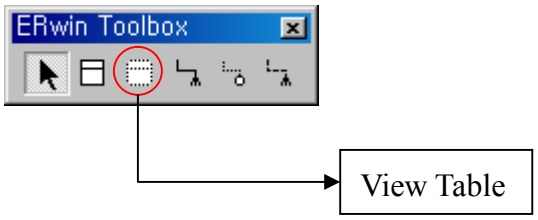
그런 다음 Expanded 텝으로 전환한 해 보면 실제로 만들어질 트리거에 대한 내용들 정의되어 있는 모습을 확인할 수 있다.
코딩을 점검해 보고 이상이 없는 지 확인하자.



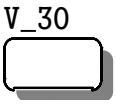
이 이후의 업데이트 트리거와 삭제트리거는 같은 요령으로 적용을 하면 될 것이다.

◦ 뷰(View) 정의 방법

: ERwin 에서 뷰(View)를 만들려면 물리적(Physical)모델링 단계에서 ERwin Toolbox 에 있는 ViewTable 을 선택해서 만들 수 있다.

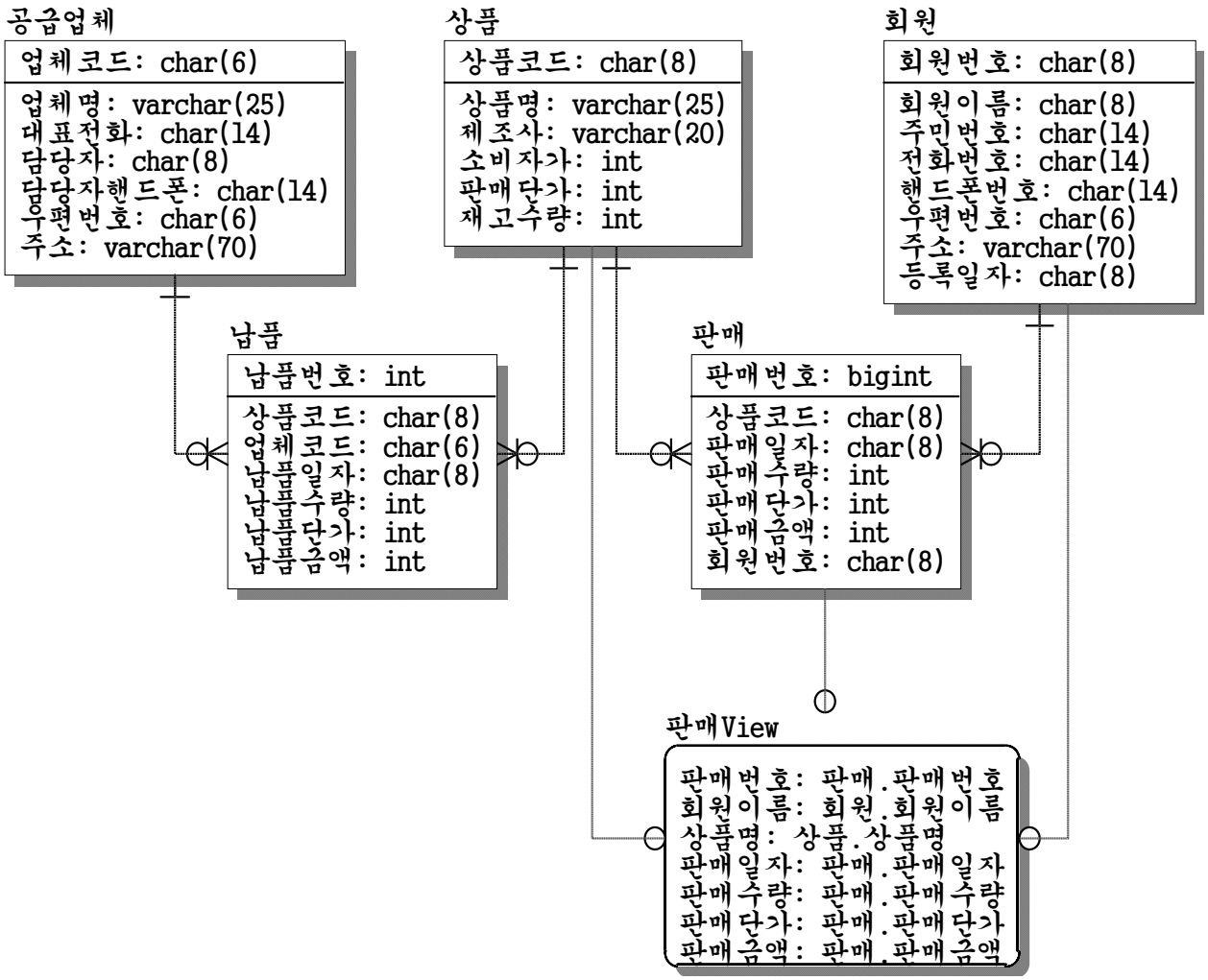


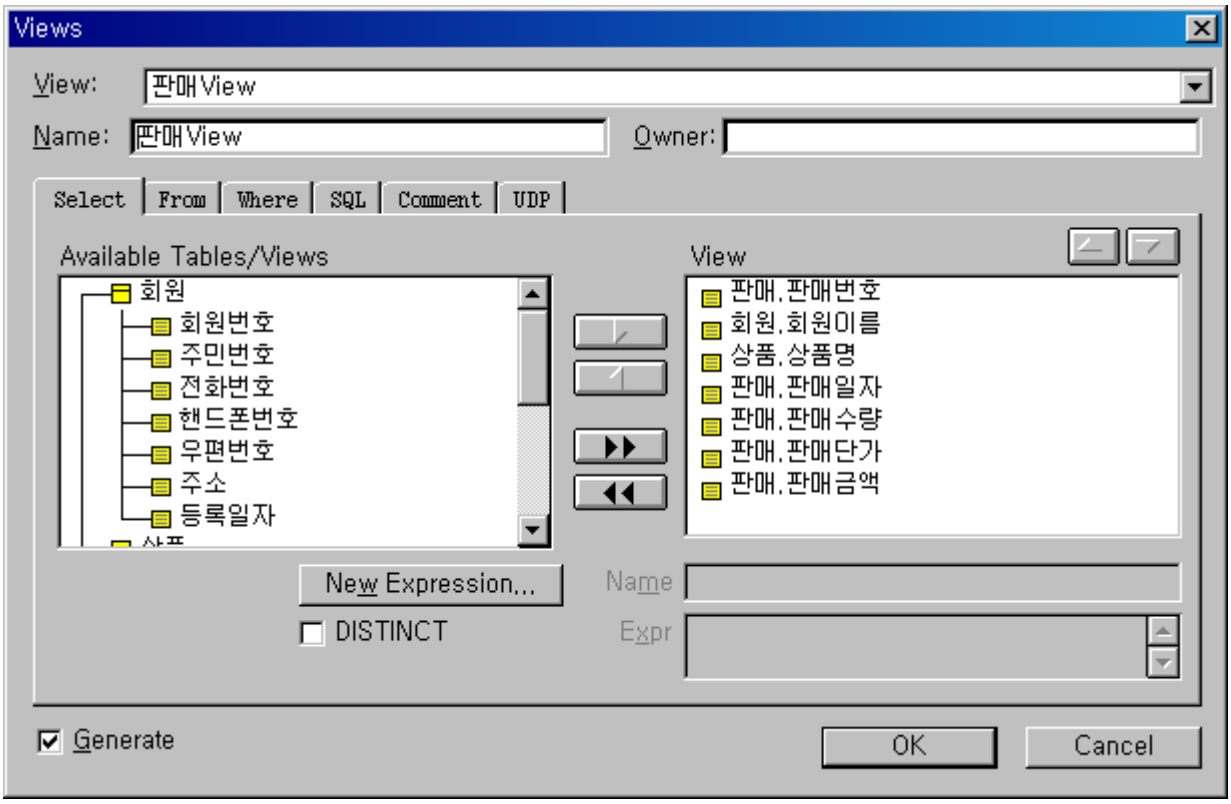
엔티티를 만들 때 처럼 View Table 을 선택한 후 다이어그램에 클릭하면 다음과 같은 모습으로 뷰가 만들어지게 되는데 이 뷰에 포함하고자 하는 컬럼을 드레그 엔 드롭하게되면 뷰필드가 구성이되어 진다.



우리는 여기서 판매테이블의 데이터를 조회하는 뷰를 만들것이며 이름을 '판매 View'로 정의할 것이다.
(뷰의 이름을 바꾸려면 엔티티 이름을 바꿀 때 처럼 바꾸어주면 된다.)

뷰의 이름을 '판매 View'로 바꾼 뒤 뷰로 구성할 필드(판매.판매번호, 회원.회원이름, 상품.상품명, 판매.판매일자, 판매.판매수량, 판매.판매단가, 판매.판매금액)들을 '판매 View'에 끌어다 놓으면 다음과 같은 아래와 같은 모습이 될 것이다.

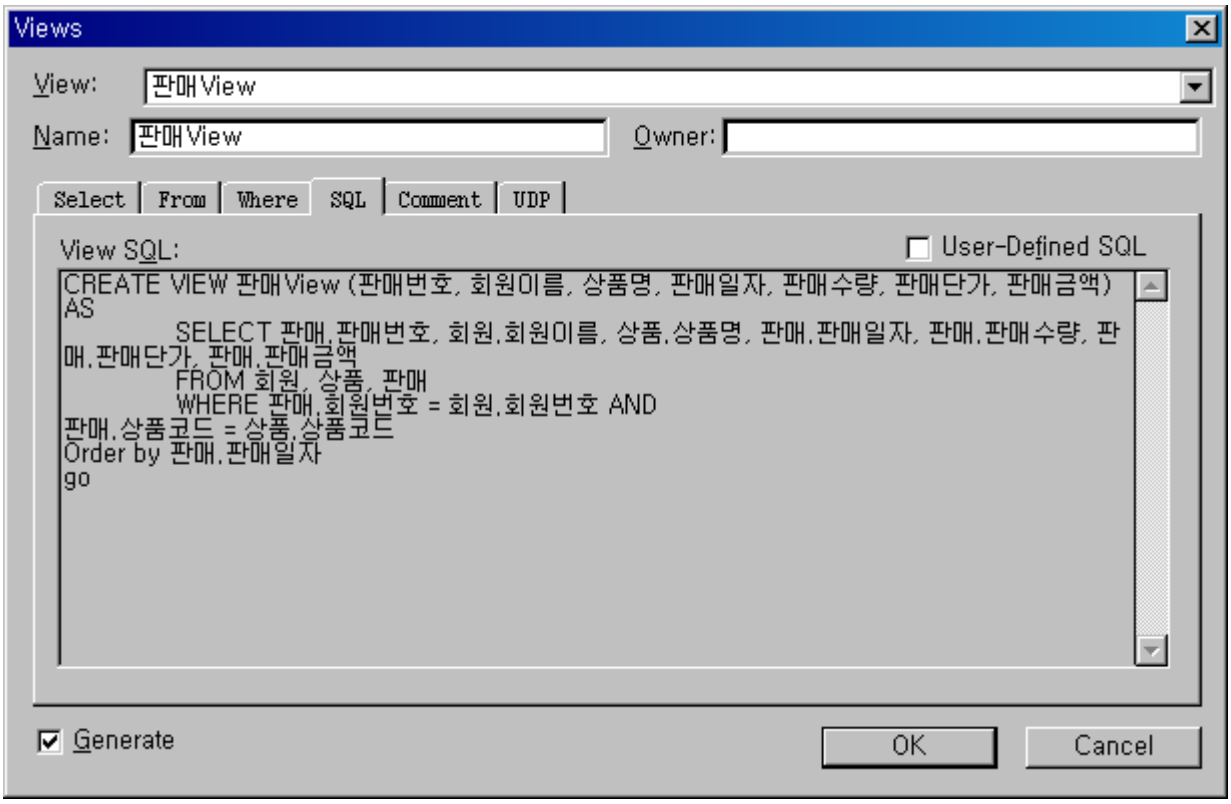

공급업체
업체코드: char(6)
업체명: varchar(25)
대표전화: char(14)
담당자: char(8)
담당자핸드폰: char(14)
우편번호: char(6)
주소: varchar(70)
상품
상품코드: char(8)
상품명: varchar(25)
제조사: varchar(20)
소비자가: int
판매단가: int
재고수량: int
회원
회원번호: char(8)
회원이름: char(8)
주민번호: char(14)
전화번호: char(14)
핸드폰번호: char(14)
우편번호: char(6)
주소: varchar(70)
등록일자: char(8)
납품
납품번호: int
상품코드: char(8)
업체코드: char(6)
납품일자: char(8)
납품수량: int
납품단가: int
납품금액: int
판매
판매번호: bigint
상품코드: char(8)
판매일자: char(8)
판매수량: int
판매단가: int
판매금액: int
회원번호: char(8)
판매View
판매번호: 판매.판매번호
회원이름: 회원.회원이름
상품명: 상품.상품명
판매일자: 판매.판매일자
판매수량: 판매.판매수량
판매단가: 판매.판매단가
판매금액: 판매.판매금액



Views 대화상자의 Select 탭을 보면 뷰에 사용된 테이블의 나머지 컬럼들과 뷰로 구성된 컬럼들의 모습이 보여지게 된다.
From 탭에는 뷰에 사용된 테이블들이 보여지게 되며 조인의 조건을 포함한 조회의 조건등을 지정하기 위해서는 Where 탭에서 관련된 내용을 정의해 주어야 한다.

정상적으로 조회가 이루어 질 수 있도록 Where 탭의 Where 절에 다음과 같은 구문을 추가해 보도록 하자.

판매.회원번호 = 회원.회원번호 AND
판매.상품코드 = 상품.상품코드



그런 다음 SQL 탭에서 완성된 뷰 문장을 확인해 보도록 하자.

◦ 스토어드 프로시저(Stored Procedure) 정의 방법

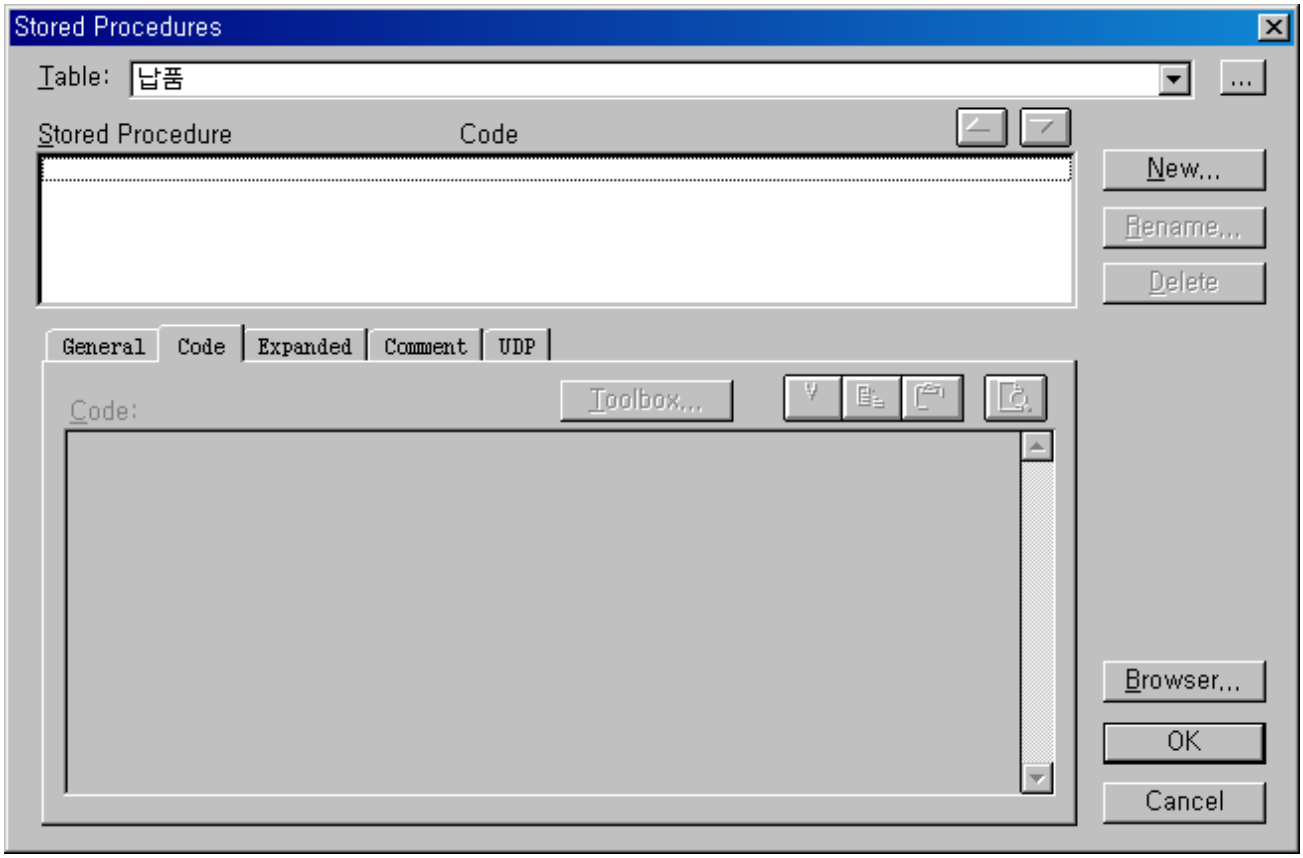
: ERwin 에서 스토어드 프로시저(Stored Procedure)를 만들려면 물리적(Physical)모델링 단계에서 DataBase 메뉴 / Stored Procedure 메뉴를 선택하면 Model-Level 메뉴와 Table-Level 의 두 가지 메뉴를 만나게 되는데 테이블(Table)이나 뷰(View)에 기반한 스토어드 프로시저를 만들고자 한다면 Table 레벨을 선택하면 되고 값의 계산이나 부가적인 프로세스를 처리하기 위한 스토어드 프로시저를 정의하고자 한다면 Model-Level 을 선택하면 된다.

하지만 이 둘의 차이는 궁극적으로 없으며 SQL Server 에서 스도어드 프로시저를 만들 경우 예를들어 판매 현황을 조회하는 스토어드 프로시저를 만들고자 하는데 지금 당장 판매 테이블이 데이터베이스에 없더라도 만들 수 있는 기능 즉 이름 지연이 가능하기 때문이다.
(중요한 내용이 아니므로 중략하기로 한다.)

위에서 뷰를 만들 때 판매 테이블의 데이터를 조회하는 뷰를 만들었으므로 여기서는 납품 테이블의 납품 현황을 날짜별로 조회할 수 있는 스토어드 프로시저를 만들어 보도록 하겠다.

Table-Level 의 스토어드 프로시저는 위의 방법으로도 만들 수 있지만 해당 테이블의 오른쪽 버튼을 누른 뒤 팝업 메뉴에서 Stored Procedure 를 선택해도 만들 수 있다.

여기서는 납품 테이블을 기반으로 한 스토어드 프로시저를 만들 것이므로 납품 테이블을 선택하고 오른쪽 버튼을 누른 뒤 Stored Procedure 메뉴를 선택하면 다음과 같이 Stored Procedure 대화상자가 나타나게 된다.



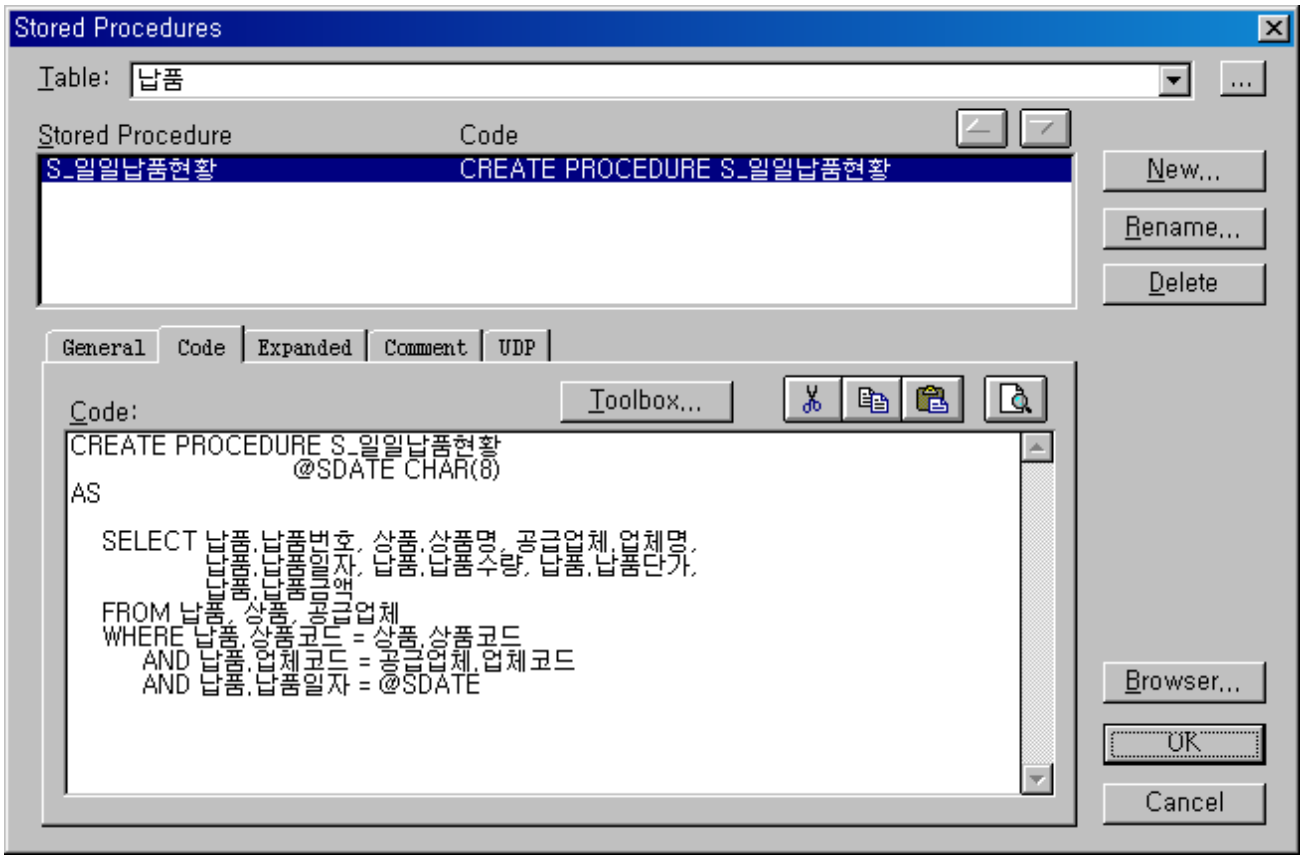
여기서 New 버튼을 누른 뒤 'S_일일납품현황'이라는 이름을 입력한 뒤 확인 버튼을 누르면 다음 그림과 같이 Code 입력 상자가 사용할 수 있게끔 활성화 되어지게 되는데 여기에 다음과 같은 코드를 입력해서 스토어드 프로시저를 정의해 보도록 하자.

```
CREATE PROCEDURE S_일일납품현황
                                @SDATE CHAR(8)
AS

    SELECT  납품.납품번호,  상품.상품명,  공급업체.업체명,
                    납품.납품일자,  납품.납품수량,  납품.납품단가,
```

```
                    납품.납품금액
  FROM 납품, 상품, 공급업체
  WHERE 납품.상품코드 = 상품.상품코드
        AND 납품.업체코드 = 공급업체.업체코드
        AND 납품.납품일자 =@SDATE
```

그러면 다음과 같은 모습이 되며 이렇게 하면 'S_일일납품현황'이라고 하는 스토어드 프로시저가 만들어지게 된다.



여기서 간단하게 스토어드 프로시저를 하나 만들어 봤는데 실제로 스토어드 프로시저의 사용은 속도 향상, 유지 보수의 편의성, 네트워크 트레픽의 감소등등 단점 보다는 장점이 월등하게 많은 아주 훌륭한 데이터베이스 개체(Object)이다.

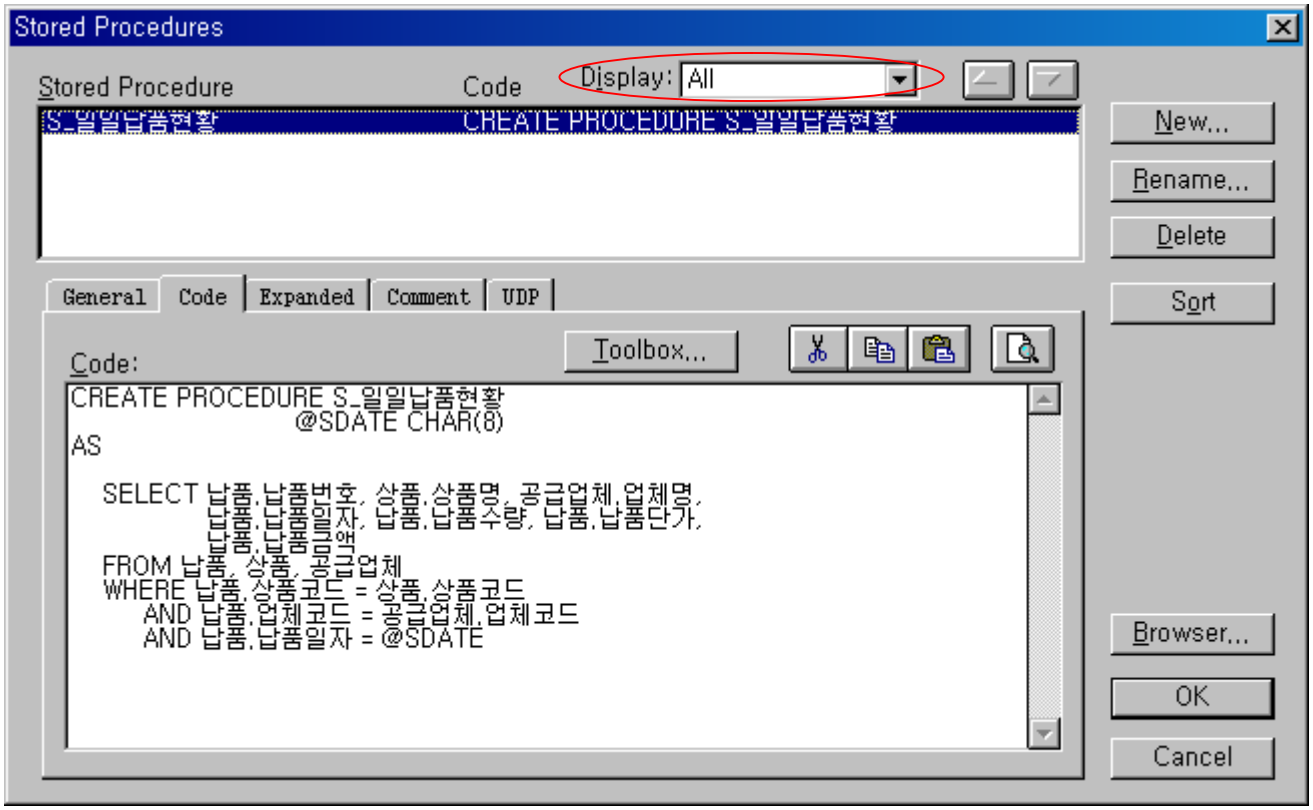
여태껏 직접 애플리케이션에서 SQL 문장을 정의해서 사용 했다면 앞으로는 스토어드 프로시저를 사용해서 코딩을 해야 할 것이다.

업무와 관련한 모든 SQL 문장을 스토어드 프로시저로 작성했다면 당연히 스토어드 프로시저의 숫자는 한 데이터베이스 안에 아주 많은 수의 스토어드 프로시저가 만

들어지게 될 것이다.

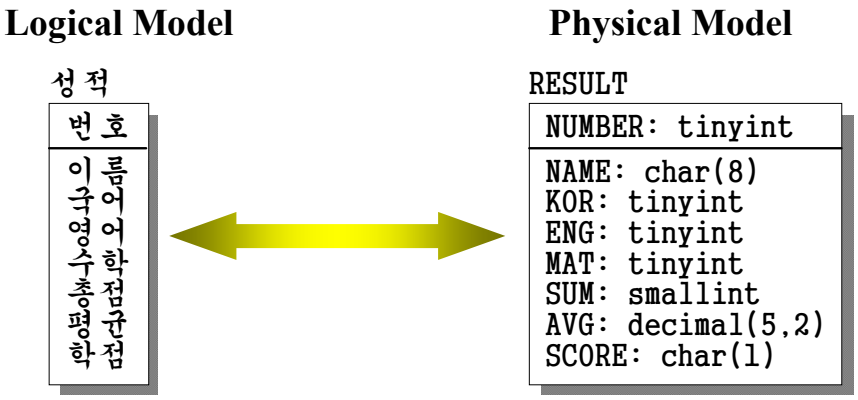
그러한 스토어드 프로시저를 모두 모아서 관리하려면 DataBase 메뉴 / Stored Procedure / Model-Level 을 선택하면 된다.

그러면 위와 비슷한 Stored Procedure 대화 상자가 나타나게 되는데 위와 다른게 없어 보이지만 위에는 없는 Display 콤보상자가 있으며 이 다이어그램 안에서 정의된 모든 스토어드 프로시저 목록과 함께 코드도 나와 있으며 물론 이 대화상자에서 새로운 스토어드 프로시저도 만들 수 있다.



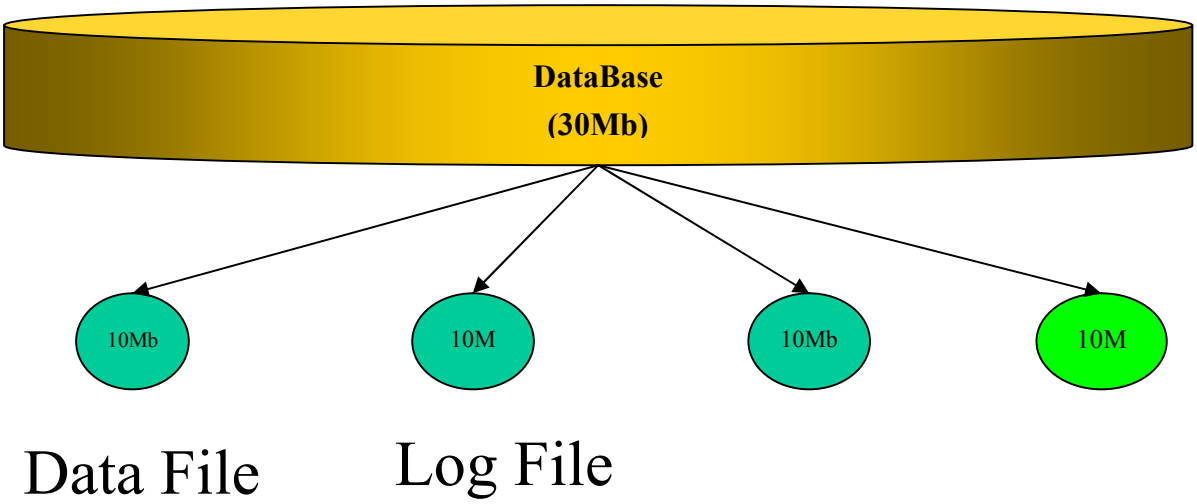
* 참고 : 여태까지의 예를 보면 테이블이나 컬럼의 이름등을 모두 한글로 정의해서 사용했다. 그러나 실제로 현업에서 데이터베이스 내의 개체(Object)명을 한글로 정의해서 사용하는 경우는 그리 많지는 않다.
물론 한글을 사용해서 반드시 문제가 되는 것은 아니지만 과거 한글 지원이 부족했던 시절부터 내려온 하나의 관습이라고 볼 수 있다.
해서 ERwin 을 사용할 때도 논리적(Logical)모델링에서는 한글로 엔티티(Entity)나 어트리뷰트(Attribute)를 정의하고 물리적(Physical) 모델링에서는

그 이름들을 영문으로 전환해서 사용하는 경우가 많은데 이렇게 물리적 모델링 단계에서 Object 명을 영문으로 바꾼다 하더라도 아래의 그림처럼 논리적 모델링 단계에서 정의된 한글명은 그대로 유지 되어진다.



◦ 파일그룹 정의

: 하나의 데이터베이스는 최소한 하나 이상의 데이터 파일과 최소한 하나 이상의 로그 파일로 구성되어 진다.



SQL Server 는 위의 그림처럼 하나의 데이터베이스에 여러 개의 데이터 파일을 구성할 수 있으며 이때 데이터 파일의 확장자는 .mdf 나 .ndf 이고 로그 파일의 확장자는 .ldf 이다.

.mdf 확장자를 갖는 데이터 파일은 데이터베이스를 생성할 때 가장 먼저 정의된 데이터 파일에 기본적으로 적용되는 디폴트 확장자이며 추가적으로 데이터 파일이 만들어지는 데이터 파일들은 .ndf 라는 확장자를 갖게 된다.

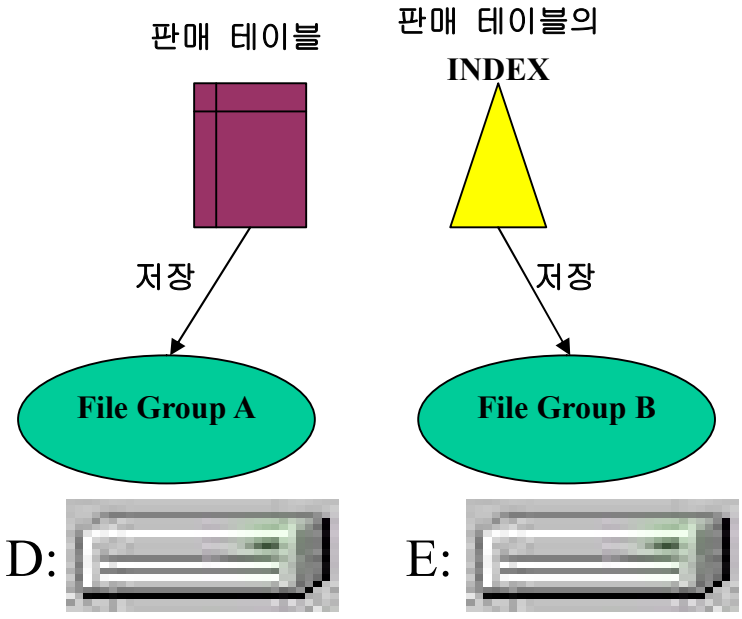
그리고 이러한 파일들은 그룹으로 관리되어 지는데 기본적으로 SQL Server 에서 만들어지는 모든 데이터 파일들은 주(Primary)파일 그룹의 구성원이 되며 새로운 파일 그룹을 만들어서 데이터 파일들을 관리자 임의로 파일 그룹에 포함시킬 수 있다.

* 참고 : 기본적으로 주(Primary) 파일 그룹에 데이터베이스 시스템 테이블이 존재하게 되며 새로운 테이블이나 뷰등을 만들 때 기본적으로 만들어지는 파일그룹 역시 주(Primary) 파일 그룹이다.

SQL Server 에서는 개별적인 파일이나 파일 그룹 별로 백업이나 복구를 할 수 있으며 저장 될 위치들을 개별적으로 정의할 수 있다.

이는 각각의 하드디스크에 파일들의 위치를 분산 시킴으로 해서 디스크에서 데이터를 읽고 쓰는데에 있어서의 경쟁을 피할 수 있음으로 시스템의 성능을 극대화 시킬 수 있는 방법으로 사용된다.

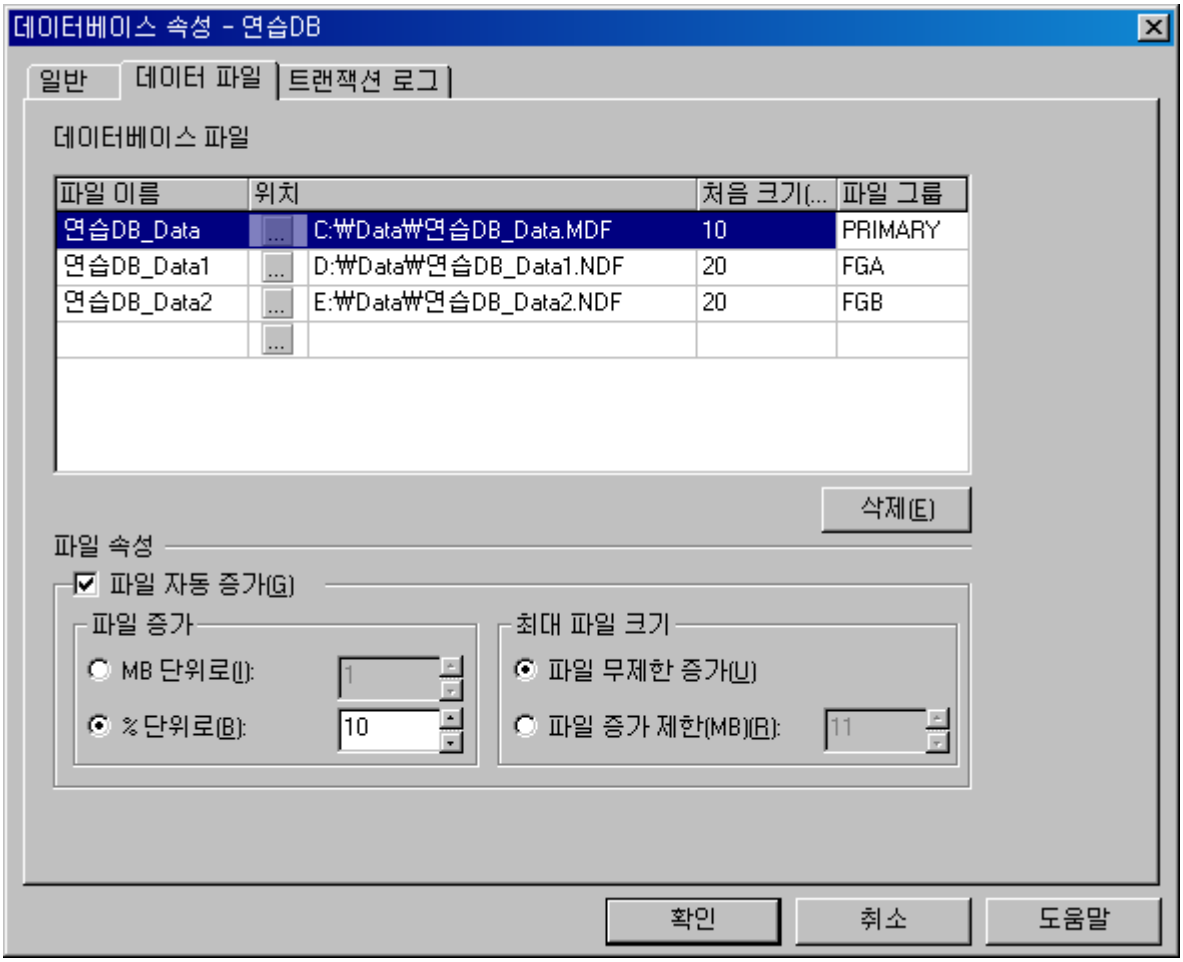
그리고 더군다나 데이터베이스의 테이블이나 인덱스가 저장되는 파일 그룹을 관리자가 임의로 지정할 수 있는데 예를 들어 아래의 그림처럼 파일 그룹 A 는 D:드라이브에 파일 그룹 B 는 E:드라이브에 만든 다음 판매 테이블은 파일 그룹 A 에 저장하고 판매 테이블의 인덱스 들은 파일 그룹 B 에 저장할 수 있다.



이렇게 저장하게 되면 판매 테이블에 데이터가 입력 되었을 경우 실제 데이터의 변경 사항은 D:드라이브가 인덱스의 변경 사항은 E:드라이브가 담당하게 되므로

보다 낳은 성능향상을 기대할 수 있다.
다음 예는 SQL Server 에서 데이터베이스를 만들면서 파일 그룹을 정의한 후에 ERwin 에서 테이블과 인덱스를 메칭 시키는 예제이다.

아래의 그림과 같이 엔터프라이즈 관리자에서 '연습 DB'라는 이름의 데이터베이스를 만들어 보자.

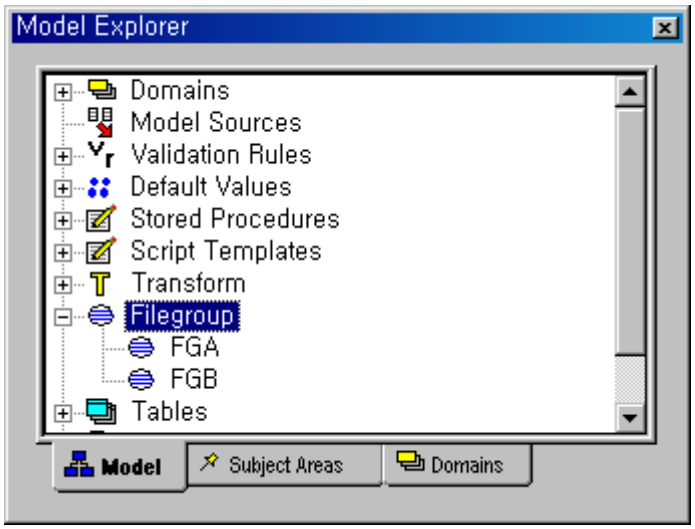


파일이름	위치	처음크기	파일그룹
연습 DB_Data	C:\Data\연습 DB_Data.MDF	10	PRIMARY
연습 DB_Data1	D:\Data\연습 DB_Data1.NDF	20	FGA
연습 DB_Data2	E:\Data\연습 DB_Data2.NDF	20	FGB

PRIMARY 파일 그룹의 이름은 변경할 수 없으며 두 번째 파일 그룹은 'FGA' 세 번째 파일 그룹은 'FGB'명으로 정의했다.
이제 다음으로 ERwin 에서 파일 그룹을 정의해 보도록 하자.

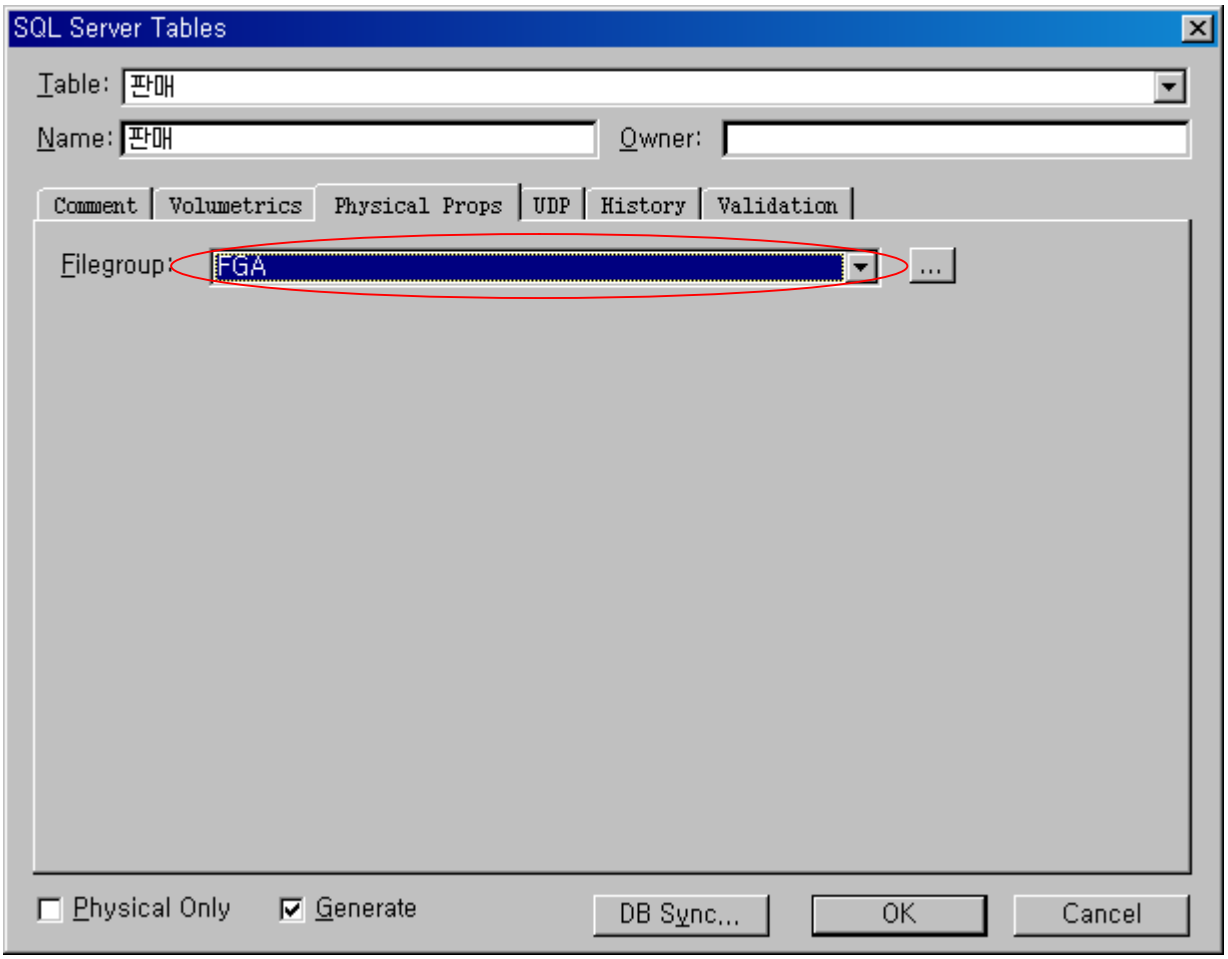
ERwin 에서 파일 그룹은 Model Explorer 의 ‘Filegroup’에서 오른쪽 버튼을 누른 뒤 New 메뉴를 눌러 정의할 수 있다.

그럼 다음 그림처럼 차례로 ‘FGA’, ‘FGB’라는 파일 그룹 두개를 만들어 보자.



그런 다음 각각 판매 테이블은 ‘FGA’ 그리고 판매 테이블의 인덱스는 ‘FGB’ 파일 그룹에 저장될 수 있도록 정의해 보도록 하겠다.

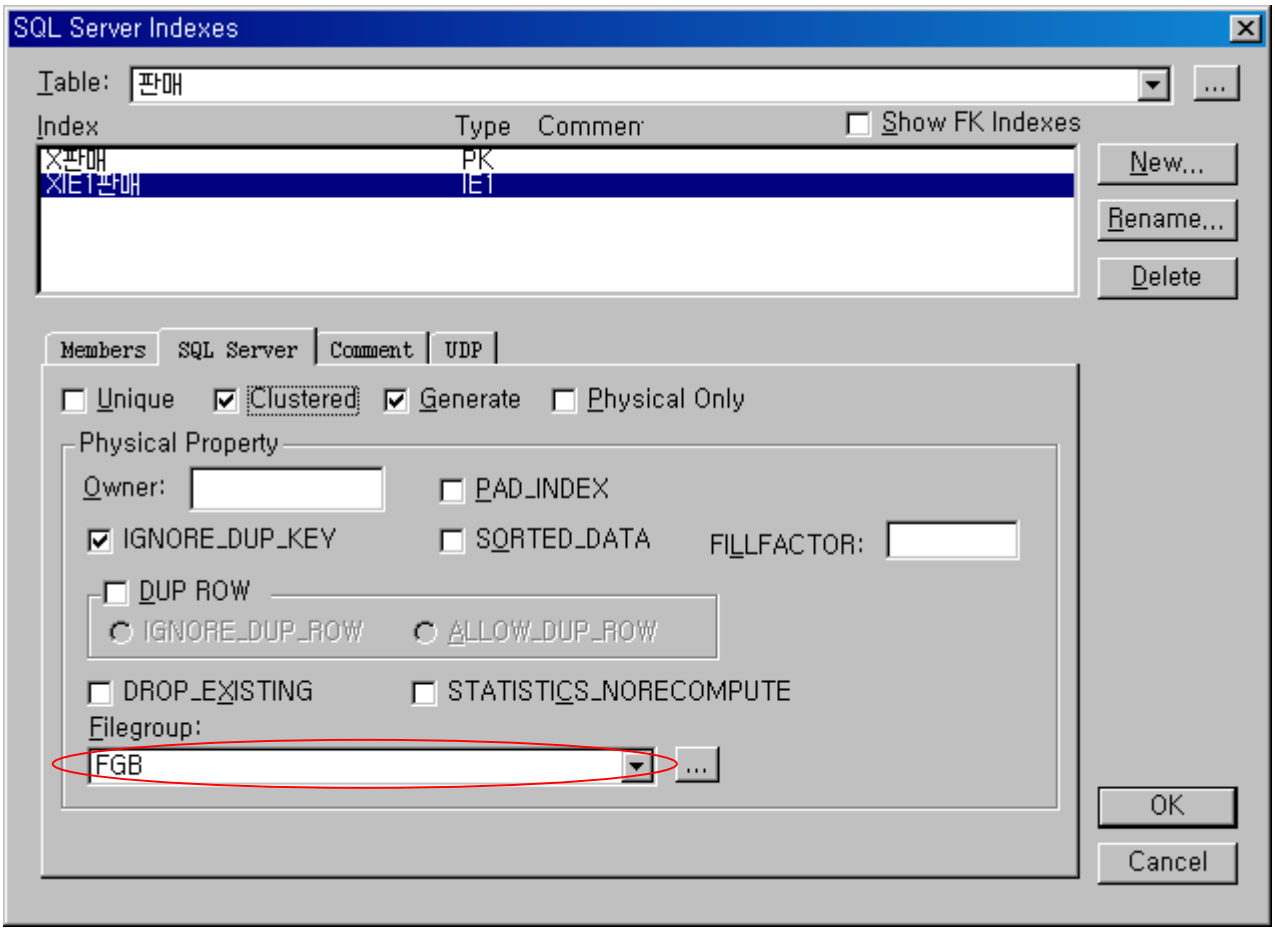
우선 판매 테이블을 정의하려면 판매 테이블에서 오른쪽 버튼을 누른 뒤 팝업 메뉴에서 Table Properties / Physical Property 메뉴를 선택하면 다음과 같이 SQL Server Tables 대화 상자가 나타나게 되는데 Filegroup 콤보상자에서 ‘FGA’목록을 선택하면 된다.



그런 다음 인덱스를 정의해야 하는데 현재 판매 테이블에는 기본키에 기본적으로 적용되는 인덱스만이 존재하고 있고 여기에 추가로 아래의 표에 나와있는 것 처럼 판매일자에 넌 클러스터드 인덱스를 만들어 보자.

인덱스	유형	유니크	컬럼	종류	파일그룹
X 판매	기본키	Unique	판매번호	Non Clustered Index	FGB
XIE1 판매	Inversion Entry	Non Unique	판매날짜	Clustered Index	FGB

그런 다음 아래의 그림 처럼 'X 판매'인덱스와 'XIE1 판매'인덱스 모두 Filegroup 콤보 상자에서 'FGB'를 선택하여 인덱스의 파일 그룹을 정의한다.



이상으로 파일 그룹의 의미와 파일 그룹을 정의하는 방법에 대해서 살펴보았다.
그런데 위의 예에는 한가지 크게 잘못된 내용이 있다.
그 내용은 다음에 소개할 데이터베이스 스키마 생성에서 설명하도록 하겠다.

만일 RAID 장비를 사용한다면 구지 복잡하게 파일 그룹등을 정의해서 사용해야 할 필요는 없을 것이다.

하지만 RAID 장비는 비교적 고가이고 일반적으로 소규모 기업에서는 RAID 장비를 이용하기가 쉽지 않으므로 위와 같이 여러 개의 물리적인 하드 디스크에 파일 그룹을 위치하는 방법으로 추가적인 비용부담 없이 보다 나은 성능 향상을 기대할 수 있을 것이다.

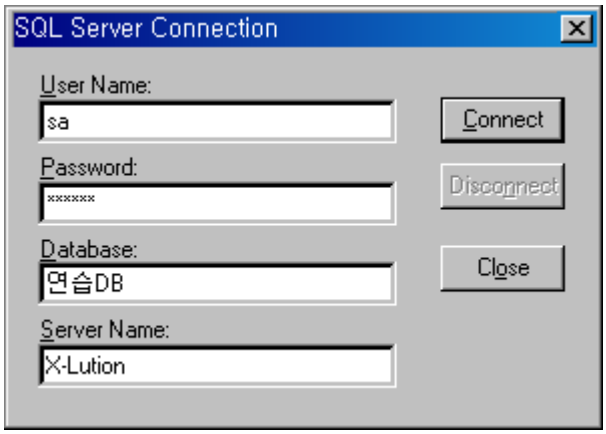
◦ 데이터베이스 스키마 생성

: 물리적 데이터 모델링 단계를 거치면서 데이터베이스의 스키마를 디자인 했다면

이제 이렇게 정의된 내용을 실제 데이터베이스 객체로 만들어질 수 있도록 해야 하는데 이를 데이터베이스 스키마 생성이라고 한다.

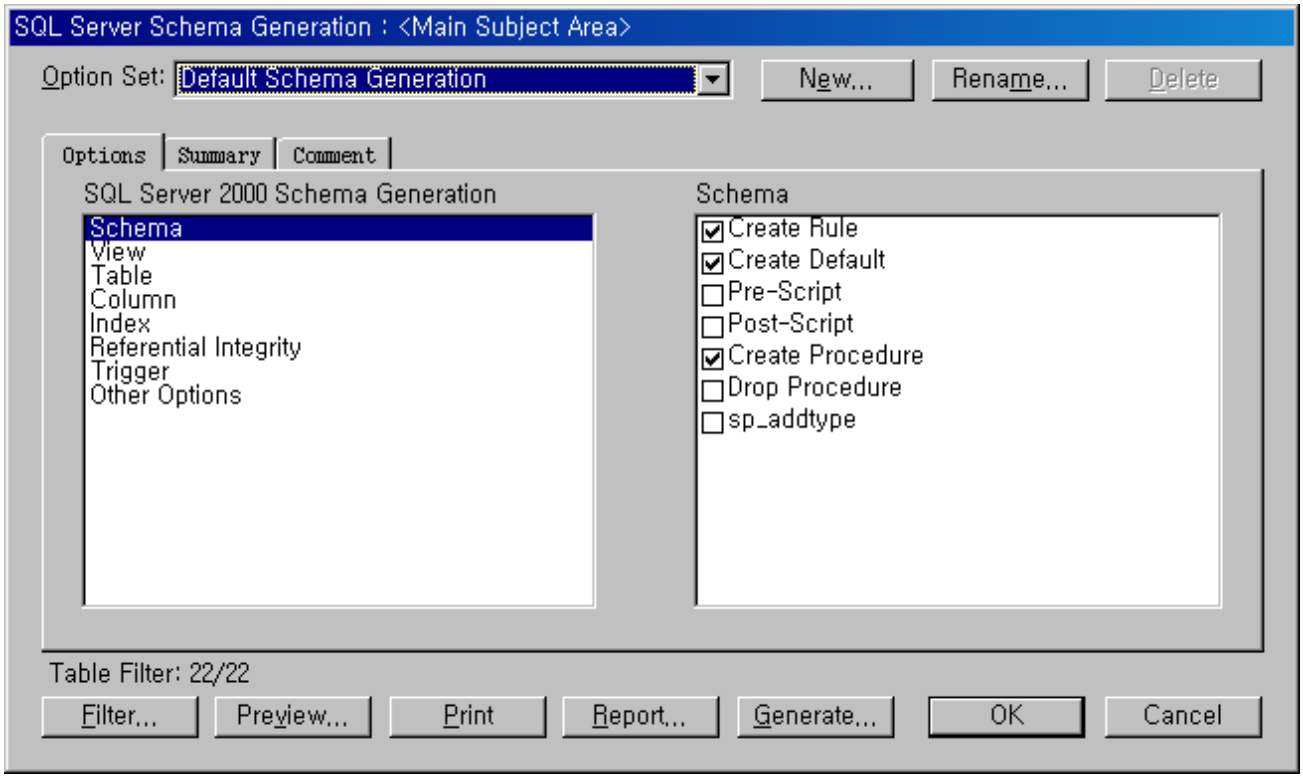
우선 디자인 된 스키마를 데이터베이스 스키마로 만들기 위해서는 우선 목적 데이터베이스에 연결해야 한다.

데이터베이스에 연결하기 전에 우선 SQL Server 에서 '연습 DB'라는 이름의 데이터베이스를 만든후 ERwin 에 Database 메뉴 / Database Connection 메뉴를 선택하면 아래와 같이 SQL Server Connection 대화상자가 나오게 되는데 여기서 User Name 은 데이터베이스 내에 스키마를 생성할 수 있는 권한이 있는 사용자 계정을 정의하면 되는데 여기서는 'sa'로 사용자 계정을 사용하도록 하겠다.
'sa'는 SQL Server 에서 최고 관리자 계정이며 SQL Server 를 설치할 때 비밀번호를 정의했다면 Password 입력상자에도 비밀번호를 입력해야한다.
'sa' 계정에 아무런 비밀번호도 정의하지 않았다면 입력하지 않는다.
그런 다음 Database 입력상자에는 스키마를 생성 할 데이터베이스 이름을 정의하며 Server Name 은 데이터베이스 서버 시스템의 컴퓨터 이름을 정의한다.



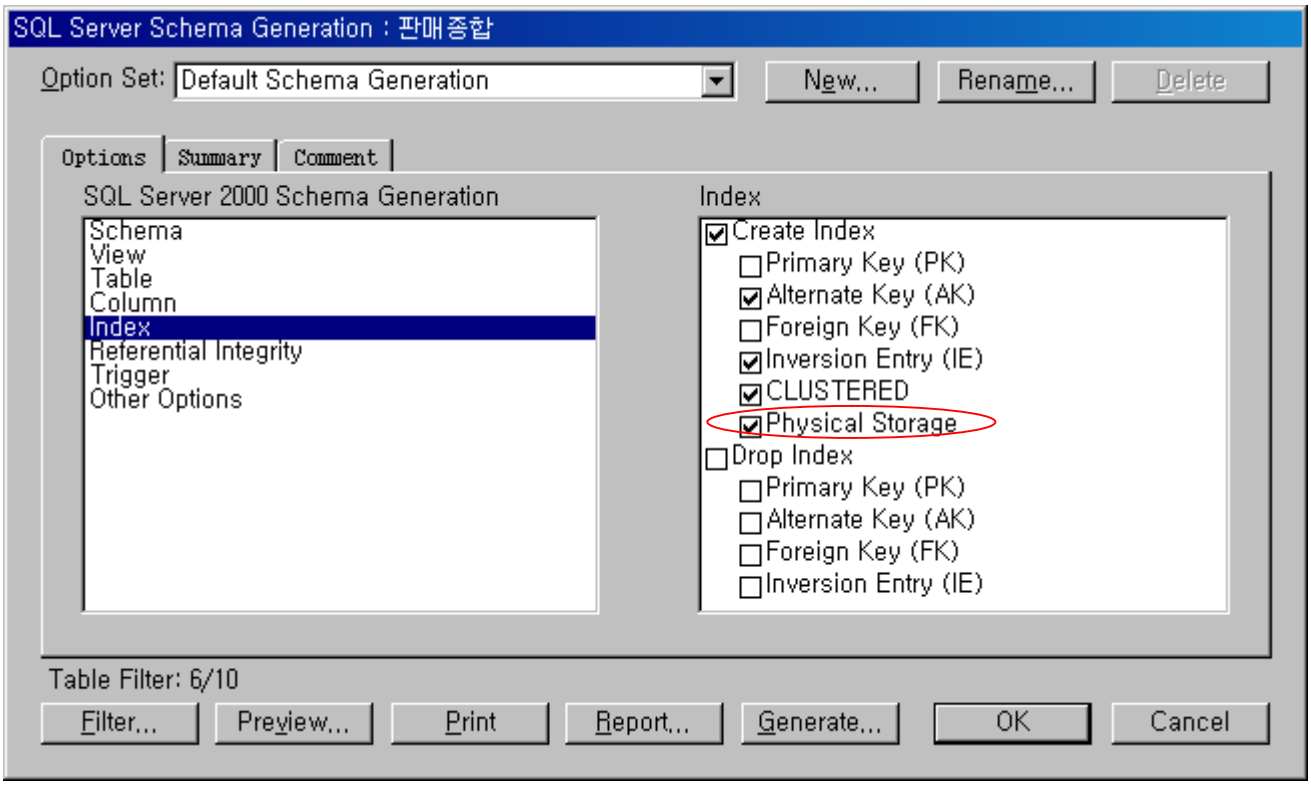
그런 다음 Connect 버튼을 누르면 연결을 시도하게 되는데 연결이 정상적으로 이루어지면 SQL Server Connection 대화상자가 사라지게 된다.
아니면 에러 메시지가 나타난다.

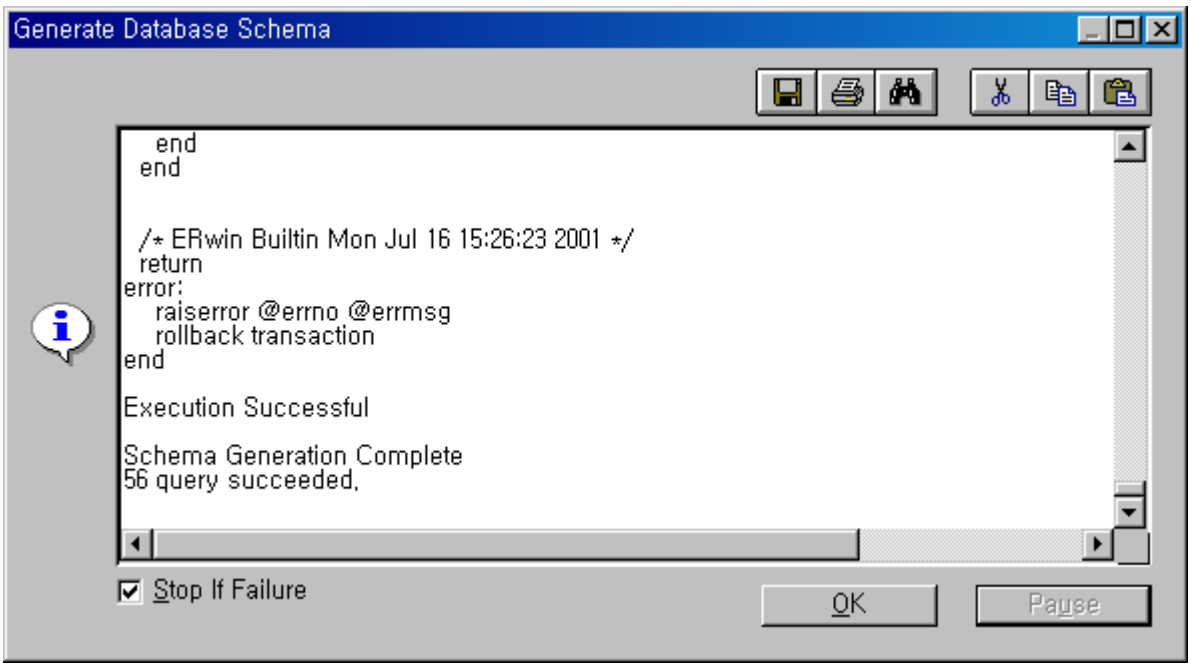
정상적으로 연결이 됐다면 이제 데이터베이스 스키마를 생성해야 한다.
데이터베이스 스키마를 생성하려면 Tools 메뉴 / Forward Engineer / Schema generation 메뉴를 선택하면 다음과 같이 SQL Server Schema Generation 대화상자가 나타나게 된다.



여기서 Generatie 버튼을 누르면 스크립트가 실행 되면서 '연습 DB' 데이터베이스에 ERwin 에서 정의한 스키마가 만들어 지게 되는데 한가지 주의할 점은 이 상태에서는 파일 그룹이 적용되지 않는다는 점이다.

앞에서 정의한 파일그룹의 설정 내용이 실제 데이터베이스에 적용되도록 하려면 아래의 그림처럼 Table 과 Index 에 있는 Schema 옵션 중에서 Physical Storage 옵션을 선택 해야만 한다.



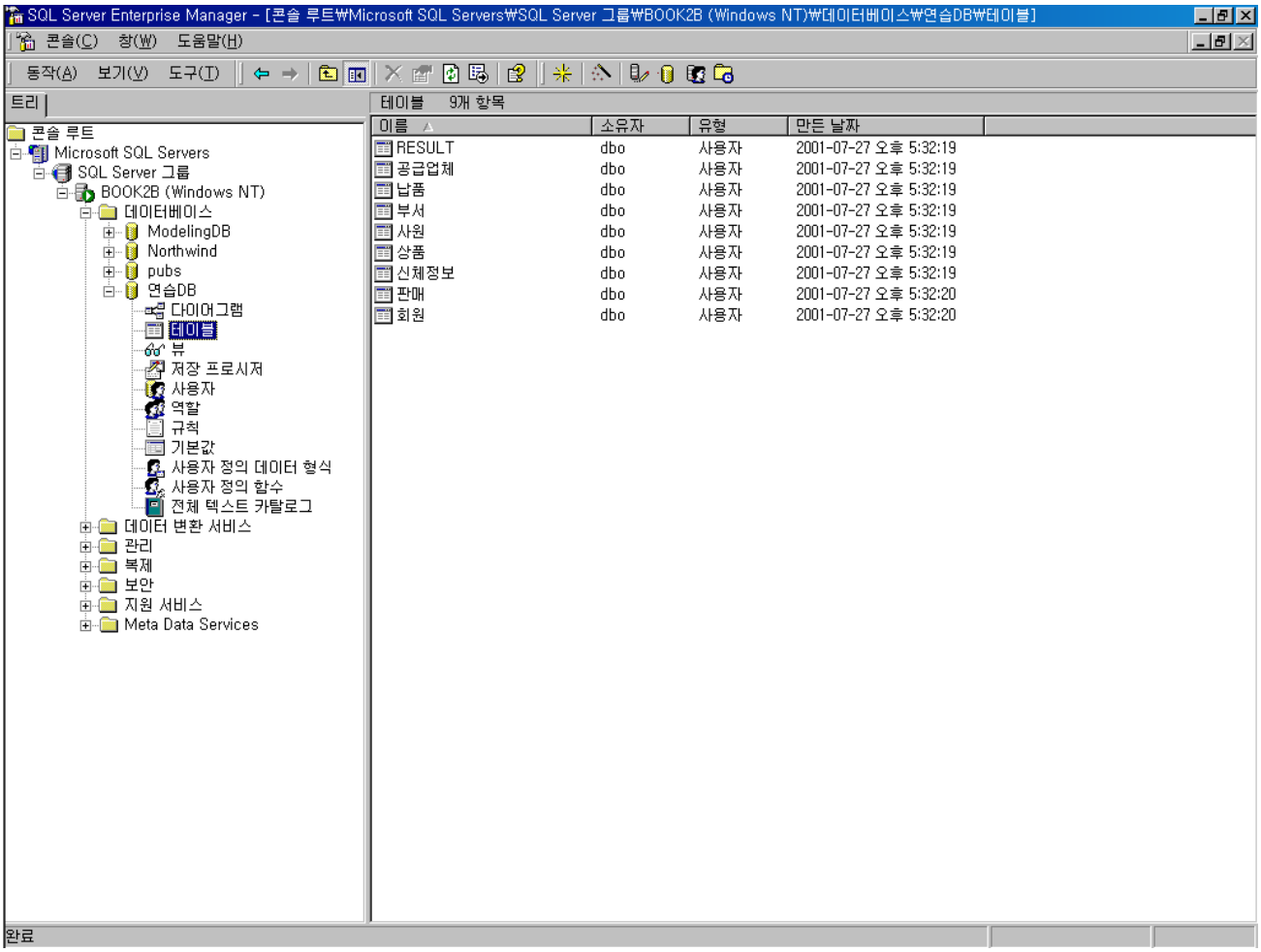


만약 중간에 에러가 나면 어떤 종류의 에러메시지인지 잘 확인해 보도록 하자. 그런 다음 해당 내용을 다시 ERwin 에서 수정해 주어야 하며 위의 Generate Database Schema 대화상자에서 저장 버튼을 눌러 위의 스크립트를 저장해 두도록 하자.

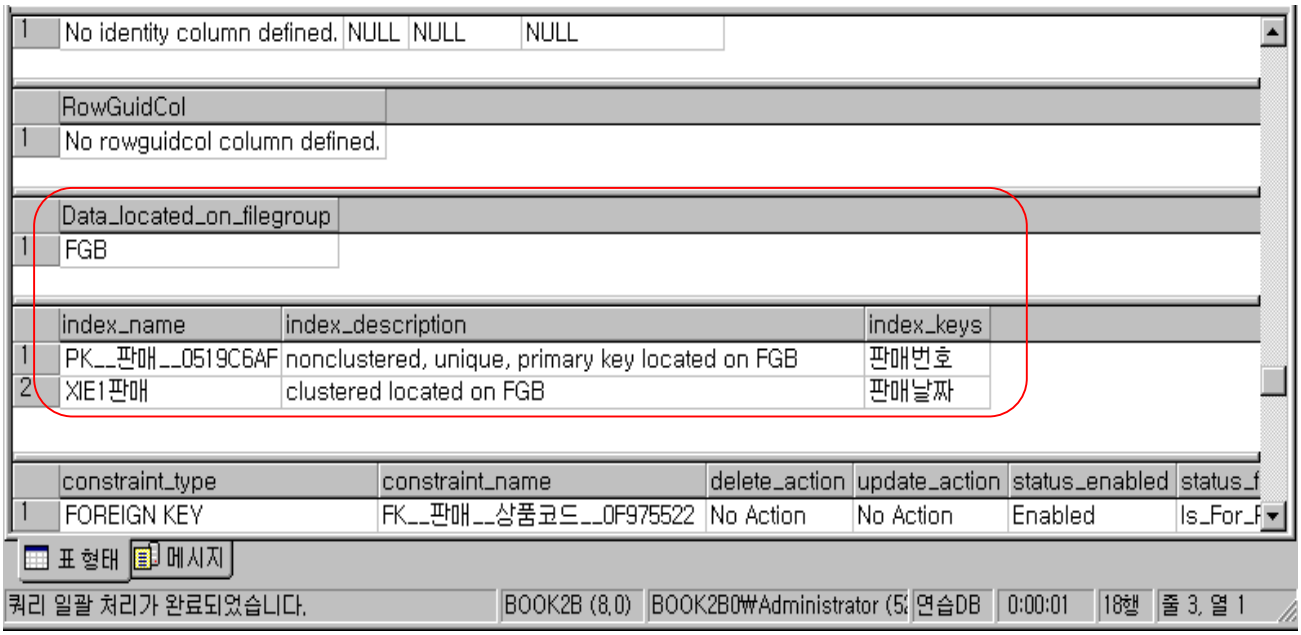
저장 파일의 확장자는 'ere'이며 메모장으로 열어서 내용을 확인 할 수 있다.

모두 정상적으로 작업을 마쳤다면 이제 실제 데이터베이스 스키마가 정상적으로 생성되었는지를 SQL Server 에서 확인해 보도록 하자.
만일 SQL Server 에서 아래와 같이 개체(Object)들이 보이지 않는다면 F5 키를 눌러서 REFRESH 한 다음 다시한번 확인해 보자.

테이블(Table), 뷰(View), 저장 프로시저(Stored Procedure), 규칙(Rule), 기본값(Default), 트리거(Trigger)등등 모든 내용들을 살펴보자.



* 참고 : 앞에서 파일 그룹을 정의해서 사용했는데 실제로 판매 테이블과 판매테이블의 인덱스가 원하는 파일 그룹에 저장되었는지를 확인하려면 쿼리 분석기에서 'SP_HELP 판매' 를 실행하면 아래와 같이 테이블과 인덱스들의 파일 그룹에 대한 정보를 확인할 수 있다.



위의 화면에서 Data_located_on_filegroup 은 판매 테이블이 위치해 있는 파일 그룹에 대한 정보를 보여주고 있으며 그 밑에 인덱스와 인덱스 유형 그리고 인덱스가 위치해 있는 파일 그룹에 대한 정보를 보여주고 있다.

그런데 판매 테이블의 위치가 'FGB'로 정의되어 있다.
이전 실습에서 분명히 판매 테이블의 파일 그룹을 'FGA'로 정의했었다.
ERwin 으로 가서 다시 한번 확인해 보도록 하자.
그런데 이게 왠일인가? 혹시 버그 ?

그렇다면 데이터베이스 스키마를 생성할 때 저장했던 스크립트를 열어서 판매 테이블과 관련한 내용을 확인해 보자.

```
CREATE TABLE 판매 (
        판매번호        bigint NOT NULL,
        판매날짜        char(8),
        판매단가        int,
        판매수량        int,
        판매금액        int,
        회원번호        char(8) NOT NULL,
        상품코드        char(6) NOT NULL
)
    ON "FGA"

Execution Successful
```

```
CREATE CLUSTERED INDEX XIE1 판매  ON  판매
(
        판매날짜
)
    ON "FGB"

Execution Successful


ALTER TABLE  판매
        ADD PRIMARY KEY NONCLUSTERED (판매번호)
    ON "FGB"

Execution Successful
```

아마도 인덱스에 대해서 제대로 공부하신 분들이라면 위 상황에 대한 대답을 할 수 있을 것이다.

클러스터드 인덱스는 인덱스의 리프 레벨(Leaf Level)이 데이터 페이지(Data Page)이다. 해서 위의 소스코드에서 테이블을 만들 때는 'FGA'파일 그룹에 만들어 지지만 판매날짜 컬럼에 클러스터드 인덱스를 만들면서 인덱스의 리프 레벨 (Leaf Level) 즉 데이터 페이지 (Data Page)가 'FGB'파일 그룹으로 재 정의 되어졌다.

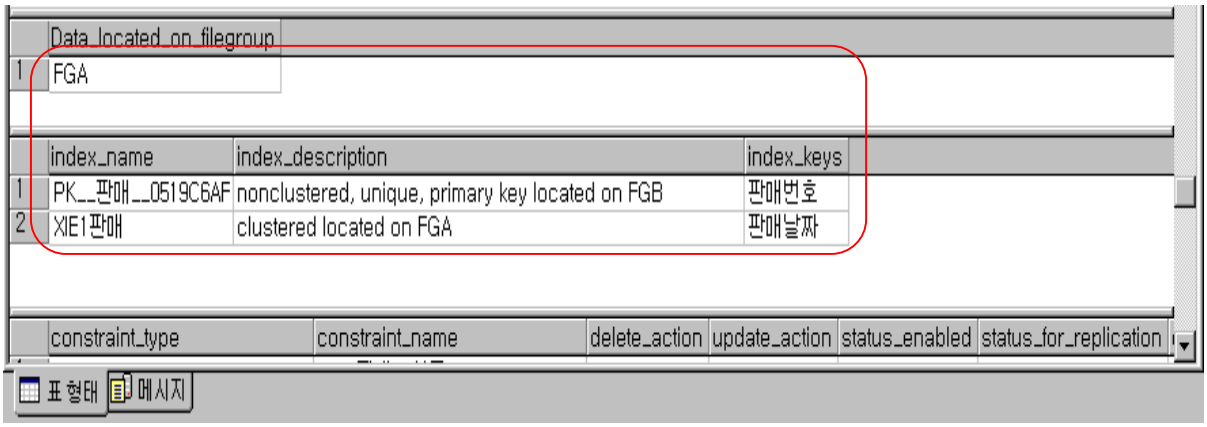
해서 판매 테이블이 최종적으로 위치하게 되는 파일 그룹은 'FGB'인 것이다.

그렇기 때문에 아무리 테이블의 파일 그룹과 인덱스의 파일 그룹을 구분한다 하더라도 클러스터드 인덱스가 만들어지면 클러스터드 인덱스가 정의된 파일 그룹에 테이블이 위치하게 되기 때문에 클러스터드 인덱스는 테이블을 저장하고자 하는 파일 그룹에 정의해야 하며 넌 클러스터드 인덱스를 테이블과 다른 파일 그룹에 저장해야 한다.

위의 파일 그룹에 대한 예는 다음 도표처럼 판매 테이블에 인덱스를 정의해야 한다.

인덱스	유형	유니크	컬럼	종류	파일그룹
X 판매	기본키	Unique	판매번호	Non Clustered Index	FGB
XIE1 판매	Inversion Entry	Non Unique	판매날짜	Clustered Index	FGA

그런 다음 다시한번 쿼리 분석기에서 'SP_HELP 판매'문을 실행하면 다음과 같이 테이블과 판매날짜 컬럼의 인덱스는 'FGA' 파일 그룹에 그리고 판매 테이블의 기본키 인덱스는 'FGB'파일 그룹에 있는 것을 확인할 수 있다.



	Data_located_on_filegroup
1	FGA

	index_name	index_description	index_keys
1	PK__판매__0519C6AF	nonclustered, unique, primary key located on FGB	판매번호
2	XIE1판매	clustered located on FGA	판매날짜

	constraint_type	constraint_name	delete_action	update_action	status_enabled	status_for_replication

표 형태 메시지

◦ 프린트 하기

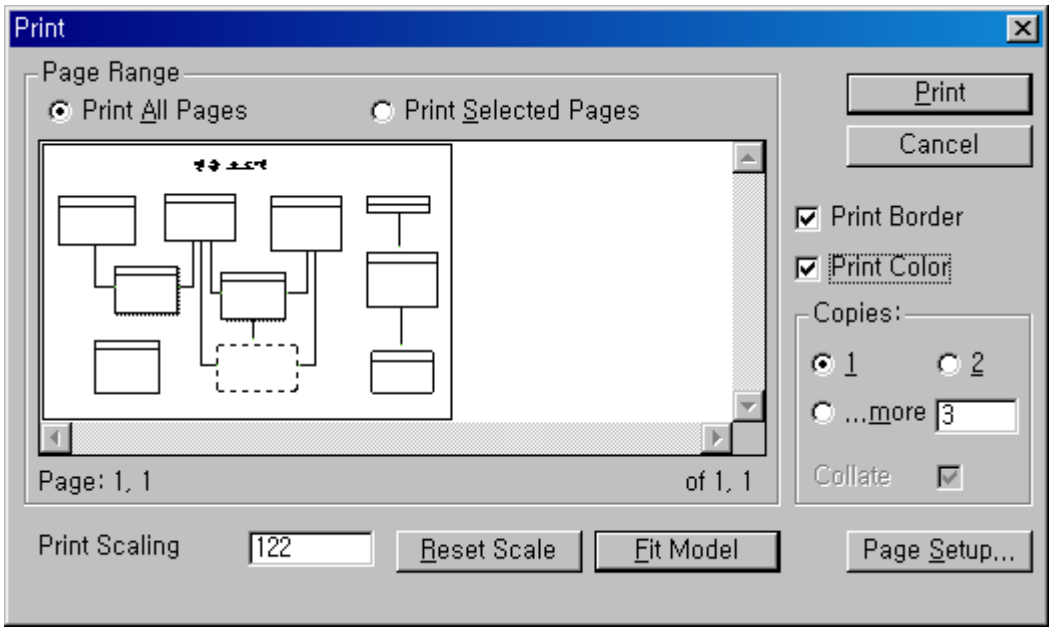
: 이제 어느정도 ERwin 의 기능에 대해서 살펴 보았다. ERwin 은 이러한 기능 외에도 아주 많은 다양한 기능을 가지고 있으며 그러한 기능들의 사용 방법이나 내용들은 ERwin 정식 매뉴얼을 참조하도록 하자.

이제 마지막으로 프린트 출력물에 관해서 정리해 보도록 하겠다.

출력을 하려면 우선 가로 방향으로 출력할 것인지 아니면 세로 방향으로 출력할 것인지에 관해 먼저 선택을 해주어야 하는데 이에 대한 설정은 File / Print Setup 메뉴에서 설정할 수 있다. 여기서는 가로 방향을 선택한다.

출력 방향이 정해진 다음 File 메뉴 / Print 메뉴를 누르면 다음과 같이 Print 대화상자가 나타나는데 출력 용지의 외곽선을 선택해서 출력물의 위치와 사이즈를 정의할 수 있으며 오른쪽 밑에 Fit Model 버튼을 누르면 전체 디자인 했던 스키마가 모두 보여질 수 있도록 자동으로 사이즈를 최적화 하게 된다.

그런 다음 Print 버튼을 눌러 출력하면 된다.


Print
Page Range
Print All Pages
Print Selected Pages
Print
Cancel
Print Border
Print Color
Copies:
1
2
...more
3
Collate
Page: 1, 1
of 1, 1
Print Scaling
122
Reset Scale
Fit Model
Page Setup...