UZMAN VERİ-TABANI SİSTEMLERİ VE ÖNERİLER

Talat Fırlar

***Özet*-Bu çalışmada uzman sistemler, ilişkisel veri-tabanı sistemleri ve nesneye-dayalı veri-tabanı sistemleri çalışıldı. Dinamik tip, Ad-hoc sorguları ve bildirimsellik , nesneye-dayalı sistemlerde ve uzman sistemlerde tartışıldı. Prolog ve SQL arasında bir front end tanımlamak için Arity Prolog ve SQL paketi kullanıldı. Bu proje yeni öneriler belirler.**

***Anahtar Kelimeler*-İlişkisel Veri-tabanı Sistemleri, Uzman Sistemler, Dedüktif Sistemler.**

***Abstract*-In this study expert systems (deductive systems), relational data-base systems and object-oriented data-base systems worked. Dynamic typing, Ad-hoc queries and declarativness are argued in object-oriented systems and expert systems. Arity Prolog and SQL package used for define a front end between Prolog and SQL. This project expresses new offers.**

***Keywords*-Object-oriented data-base systems, deductive systems.**

I. GİRİŞ

Uzman veri-tabam sistemi, uzman sistemlerin kullanılarak veri-tabanlarının bütünlük sınırlamalarının ve sorgulama optimizasyonlarının yapıldığı birleşik bir sistemdir. Günümüzdeki yazılımlarda, sorgulama gerektiren ortamlarda bu zeki sistemlerin ihtiyacı açıktır. Nesneye dayalı sistemlerde ise ilişkisel ve dedüktif veri-tabanı yönetim sistemlerinde bulunmayan özellikler vardır.Bu yüzden, uzman veri-tabanları ile nesneye dayalı sistemlerin ortak kullanımıyla birçok problem aşılabilmektedir. Bu sistemler ortak kullanılırken bazı dezavantajlar da söz konusu olmaktadır. Bu makale boyunca uzman veri-tabanı sistemleri ile nesneye dayalı sistemler arasındaki benzerlikler ve farklılıklar ele alınacaktır.

Talat Fırlar;İ.Ü.Teknik Bilimler M.Y.O,Avcılar,İstanbul

I.1 Uzman Sistemlerle Veri-tabanlarının Bütünleştirilmesi

Uzman sistemin geleneksel yazılım ile bütünleşmesi ve dışarıdaki dünya ile bağlanması için anahtar, zeki veri-tabanı geliştirmektir[1].
Uzman sistemlerin ve veri-tabanlarının, zeki veri-tabanları çalışmasının içinde bütünleştirilmesi kısmen ayıklanmalıdır; çünkü, kullanıcı direk olarak veri-tabanı yazılımları üretebilir[2].
Aşağıdaki yetenekler, bir uzman veri-tabanı sisteminin potansiyel yeteneklerinin kısmi bir listesidir :
Karmaşık sorgulamalar, geçişli-kapalı sorgular ve öz yenilemeli sorgular,Doğal dil ara yüzü gibi bir front-end dizayn edilebilir ve karmaşık front-end 'lerde bile sistem bir sorgulama için kısmi bir cevap verebilir,Veri ile uğraşan uzman kurallar, içsel hiçbir uzman sistem aracı kullanmaksızın aynı çevrede programlanabilir. Basit veri-tabanı uygulamaları (veri girişi, rapor üretme ve uygulama üretme) ve verinin zeki işleminde (intelligent processing) birbirleriyle karıştırılabilir [3],Bir uzman sistem ve ilişkisel bir veri-tabanı yönetim sistemi arasındaki ilk ilişki şemalarında, verinin bulunup getirilmesi için çok gevşek bütünleştirme teknikleri kullanılmıştır. Bu teknikler de dinamik SQL sorgulamaları iki sistem arasında birinden diğerine çevrilmiştir. Bu gevşekliği ortadan kaldırmak için nesneye dayalı programlama tekniklerinden yararlanılmıştır[4].

II. VERİ-TABANI TEKNOLOJİLERİ

Hem dedüktif hem de nesneye dayalı veri-tabanı sistemleri, veri-tabanı teknolojisinin yeni uygulamaları aracılığı ile motive edilmektedir. Eski kuşak veri-tabanı sistemleri örneğin bankacılık, hava-yolu rezervasyon sistemleri gibi büyüyen nüfuslara uygulanması gereken sistemlere cevap veremiyorlar.

Yeni kuşak veri-tabanı sistemlerinin gelişmesinde bu iki yaklaşım göze çarpmaktadır; nesneye dayalı sistemler ve kural tabanlı sorgu dilleri.Gerçekte bu iki formalizmi, zıt gibi görünüyorlar; fakat birbirlerinin birçok eksik

yönlerini tamamlayan özellikleri ve bir tek kümede kaynaştırılabilmeleri, bir yaklaşımın içinde bütünleştirilebilmelerini sağlayabilir.Teorik olarak bu yaklaşımın ön fizibilite çalışmaları birçok araştırmacı tarafından yapılmıştır. Böyle bir yaklaşım şunları gerçekleyebilmelidir:Nesneye dayalı bir yaklaşım olmalı, nesnelerin sınıflarını hiyerarşileri genelleştirerek ve nesne paylaşımı ile desteklemelidir.Standart kural-tabanlı programlama dilleri kullanarak bütünlük sınırlamaları açıklanmalı.Kural-tabanlı programlama dili kümeleri çoklu kümeleri, sıraları desteklemeli ve reddin (negation) formlarını kontrol etmelidir.Kural-tabanlı paradigma, sorguları ve güncellemelerin her ikisini de icra edebilmelidir.Sistem, modülleri desteklemeli. Bir modül, tip eşitliklerinin ve kuralların bir koleksiyonudur.Burada sorguları ve güncellemeleri belirlemek için kural tabanlı yaklaşımın ve nesneye dayalı veri modellemenin bütünleştirilme şartları belirtilmiştir. Bu özellikler LOGRES dilinde gerçekleştirilmiştir.[1]

Geleneksel olmayan veri-tabanı uygulamalarında, dizayn ve planlamada nesneye dayalı veri-tabanlarının kullanımı denenir, çünkü karmaşık yapılar (nesneler) ortaya çıkmaktadır. Ancak, bu problemlerde bilinmeyen miktarlarda non-deterministik problemler oluşmaktadır.

Bir nesneye dayalı model ile bağlayıcı bilgi ve global sınırlamalar problemlerin planlarının veya şedüllerinin belirlenmesinde iyi bir ortamdır. Gerçekçi bir sınırlama çözücünün yürütümündeki, şu üç adım çözülmelidir :

Alan indirgemesi (domain reduction) için bazı tekniklere ihtiyacımız var. Gerçek hayattaki problemleri çözerken, geri-izlemeye (backtracking) başlamadan önce sınırlama yayılımlarıyla her bilinmeyen değişkenin alanlarının indirgenmesi için mükemmel tekniklere dayandırılmalıdır.Problemdeki karmaşıklık engeli çok yüksekse, problemler genellikle basitleştirilirler. Bu basitleştirme alana-bağımlı heuristic'lerdir (domain dependent heuristics). Problemle, heuristicler mümkün oldukça bildirimsellik özelliği kaybolmayacak şekilde ilişkilendirilirler, aksi taktirde ad-hoc programlama üzerindeki mantıksal programlamanın sınırlamalarının avantajları kaybedilebilir.Daha çok yönetilebilen kısımlarda sınırlamaların derlenmesine ihtiyacımız var.Yukarıdaki teknikler LAURE dilinde uygulanmakta ve başarı ile yürütülmektedir.[5]

II.1. İlişkisel Modelin Genişletilmesi

Hem nesneye dayalı (ND) hem de uzman veri-tabanları (UVT) yaklaşımları ilişkisel modeli bazı yönlerden genişletmeye kalkışır. ND sistem yaklaşımında, "sınıf " notasyonu "ilişkisel şema" ile değişir ve genişleme şöyle olur [6]:

1.Kullanıcıların tanımladığı fonksiyonların veya "metotların" keyfiliğiyle,
2. Basit veri yapılarını kullanarak ilişkiler (satırlar kümesi) aracılığı ile kullanıcının tanımlayabileceği tiplerin zengin bir kümesine genişler.

UVT yaklaşımı, genişlemelerinde daha koruyucudur. Daha genel veri tipleri yaratma yetenekleri vardır, fonksiyon sembolleri ve argüman terimlerini mantıksal araçlar olarak kullanır. Kümeler, bazen bağımsız bir kavram gibi desteklenir ve listeler, fonksiyon sembollerinin mekanizması boyunca doğal görünürler. Ancak, UVT sistemlerindeki büyük ek güç mantığın güçlü bir kullanımından ileri gelir. İlişkisel sistemler birinci dereceden mantıksal ifadelerin üzerine sorgu dili gibi kurulurken, UVT sistemleri birinci dereceden ifadelerin koleksiyonlarını kullanır. Bunlar genellikle Horn ifadeleri gibi veya if-then kuralları gibi yazılır ve sabit noktalara dayalı bir yazılımları vardır.

Bu sınıf sistemler genellikle daha az ticarileşmiştir; fakat geliştirilen birçok deneysel sistemler vardır. Bunların ilki LDL'dir (Naqvi ve Tsur [1989]). Bu sınıfının en gelişmişidir, tüm dedüktif ve i-tabanlarının atası kabul edilir.

Bu tipteki diğer projeler,
CORAL (Ramakrishan [1990])
Aditi (Vagahari [1990])
LOGRES (Cacace [1990])
Ded Gin (Cefebre ve Vieille [1989])
NAIL ([7]], Phipps[1990], Derr & Ross [1991]).

II.2. İlişkisel Nesneler ve İlişkisel Yüklemler

Bir bilgi tabanı içindeki veri-tabanı ilişkilerinin simgelendirilmesi için bütünleştirme iki değişebilir modele dayanır.

Yüklem Modeli
Nesne Modeli

Bu iki model tam olarak değişebilir ve istenilen şekilde kullanılabilir. Örneğin, nesneler ile bir ilişki tanımlanıp yüklemler ile ilişkiyi bulup getirebilirsiniz.

Özel bir veri-tabanı için ilişkisel, yüklem ve nesne metotlarının şemalarını tanımlayalım :

a) İlişkisel Form :
İşçi = (IşçiNo, İsim, Maaş, BölümNo)
Bölüm = (BölümNo, Fonksiyon, Idareci)

b) Nesne Modeli :

Nesne	: İşçi
Nesne	: Bölüm
Aile	: Ilişki
Aile	: Ilişki
Anahtar	: IşçiNo

Anahtar : BölümNo
Sıfat : Isim
Sıfat : Fonksiyon
Sıfat : Maaş
Sıfat : Idareci
Sıfat : BölümNo

c) Mantıksal Yüklem Modeliyle :

Işçi IşçiNo, Isim, Maaş, BölümNo
Bölüm BölümNo, Fonksiyon, Idareci

III. ND VE UVT YAKLAŞIMLARI ARASINDAKİ REKABET

ND sistemlerindeki gibi, bu gruptaki UVT sistemleri sorgu dillerinde bir Turing makinasının gücünü önerir (ilişkisel sistemler gibi sistemler bunu sağlamaz) dolayısı ile yeterli genişleme olmalı. Ancak, sonunda üç önemli fark bulundu [3]:

1. Bir ND programlama dili programlarken, program oldukça gelenekseldir; mantıksal dillerdeki programlama program yazma anında özellikle yazılımlar en az sabitleştirilmiş nokta yazılımlarıyken , zor görülebilir.
2. ND sistemlerin uygulama programcıları tarafından kodlar yazılırken genellikle yazıldığı gibi çalıştırılabilir, bir UVT sistemindeki kod, rekabet ortamında icra edilecekse ciddi boyutta kod optimizasyonlarına ihtiyaç duyar.
3. Bir mantıksal programlama dilinde yazılan programlar, ND programlardan daha bildirimseldir (declarative). Herhangi bir kişi çok yüksek düzeydeki algoritmik fikirlerini hesaplamaların düzenli detaylarını belirtmeden açıklayabilir.1. nokta UVT sistemlerinin ümitsiz vaka olabileceklerini savunurken, 2. nokta UVT programcıları sayfalar yazarken ND programcıların neden sistem yazdıklarını soruyor. Aşağıdaki bölümlerde 1. noktaya ilk bakışımızın neden yanlış olabileceğini açıklamaya çalışacağız.

Bir sorgu dilinin hem mantıksal hem de nesneye dayalı olması mümkündür. Bu iki filozofiyi birleştirmek mümkündür, fakat bir ND sisteme mantık eklemeyi denemektense, bir mantıksal taban ile bütünleştirme yoluna gidilebilir.

IV. ND VE UVT SİSTEMLERİNİ BİRLEŞTİRMEK NEDEN ZORDUR

Hem tamamen dedüktif hem de tamamen nesneye dayalı bir sistem niçin yoktur, nedenleri şunlardır :

1. Nesne tanımlamada mantıksal sabitleşmiş noktalar sizi istenmeyen zorluklarla karşılaştırır.
2. Tip sisteminizi genişletmekte ciddiyseniz, bildirimsel bir programlama yöntemi kullanamazsınız.

IV.1. Mantıksal Olarak Nesne-Özdeşliği

Yukarıdaki örnekte kurallar (1) ve (2) üzerinde **nesne** özdeşliği oluşturmak için, kuralları fonksiyon sembolleri ile aşağıdaki gibi dönüştürebiliriz.

s(1) : gidiş (X,Y,F) :- direk_gidiş(X,Y).
s(2) : gidiş (X,Y,g(I,Z)) :- direk_gidiş(Z,Y), gidiş(X,Z,I).

Burada, **gidiş**'in üçüncü argümanı bir "nesne özdeşliği" gibi davranır, her **gidiş** gerçeği için üçüncü argüman, aşağıdaki formda olduğu gibidir,

$$g(g(.....g(F,Z_1),.....Z_{k-1}),Z_k)$$

Buradaki $Z_1,.....,Z_k$ orta düğümleri, birinci argümandan ikinci argümana gidilirken bulunan gidişlerin listesini verir. Bir mantıksal dil kullanırsak nesne özdeşliği çok büyür. Herhangi bir kimse isterse nesne özdeşliğini benzetebilir (s(1) ve s(2) kuralları aracılığı ile davranış önerilir). Fonksiyon sembolleri, bir gerçeğin türevlerinin bir izini sağlamada fazladan bir argüman gibi kullanılır.

IV.2. Dinamik Tip ve Ad-hoc Sorguları

Mantık ve nesneye dayalı sistemler karışamaz, varsayımı dinamik tip sistemlerle ilgilidir. Bilindiği gibi nesneye dayalı programlarda, çalıştırma (run) anında yeni tipler bildirmek mümkündür. UVT sistemlerinin çok önemli bir özelliği, ad-hoc sorgularını kullanıcıların hizmetine sunmasıdır. Ad-hoc sorguları kısadır, yüksek düzeyli bir dilde kolayca yazılırlar. Ad-hoc sorguları ile dinamik tipler karıştırılamaz.

Bir $R_1(A,B)$ ilişkisini $R_2(B,C)$ ilişkisi ile birleştirirsek yeni bir ilişki T(A,B,C) oluşur. Benzer şekilde mantıksal ifade olarak,

$$t(A,B,C) :- r_1(A,B), r_2(B,C).$$

T ilişkisinin yeni bir tipi vardır; çünkü daha önce A,B,C sıfatlarını içeren bir ilişki yoktu. İlişkisel cebrin operasyonları T ilişkisi için hemen kullanılabilirdir.

Operasyonların bu ön tanımı, ilişkisel veya mantıksal sistemlerde ad-hoc sorgularını mümkün kılar. Zor ifadeleri hızlı ve keyfi olarak kurabiliriz. Her defasında yeni bir tip yaratmaya gerek kalmaz.

ND sistemler, bu argümana şüphesiz hile olarak bakar. Sebebi ise, ilişkisel veya mantıksal sistemlerin yeni "tip" yaratmaları oldukça sınırlıdır. Tip sistemleri çok zayıftır. İlişkilere istediğiniz ekleri yapamazsınız, dolayısı ile kullanacağınız tüm operasyonları bir defada tanımlamalısınız.

Bir ND sistemin bu noktada avantajları tartışılmazdır. İlişkilerin ihtiyacı olan yeni tip ve yapıları tanımlayabilirsiniz ve sonra ad-hoc sorgularını yapabilirsiniz. Ayrıca, ND sistemleri herhangi bir tipin elemanları için okuma ve yazma benzeri birçok basit operasyonları tipik olarak tanımlar, dinamik olarak bildirilmiş tipleri dahil eder.

Hem dinamik tiplerimiz hem de ad-hoc sorgularımız olsun. Bunları beraber kullanmak mümkün değildir. Çünkü ad-hoc sorguları dinamik tipleri yok eder.

V. BİLDİRİMSELLİK İÇİN GEREKENLER

Birçok sorgulama dilinde argümanlar, bildirimselliğe dayanmaktadır. Bildirimsellik göreceli bir kavramdır. SQL, APL veya mantıksal diller bildirimseldir.

V.1. Nesne Tabanları

Yeni uygulamalar, sorgu dillerinde bildirimsellik isterler. Açıkça bazıları da istemez. Örneğin, bir UVT veya ilişkisel sistemdeki bir derleyicinin veriyi yüklemeye ihtiyacı olmayabilir; çünkü yapılar sabittir. Derleyicinin ad-hoc sorgularına ihtiyacı yoktur ve performans hayatidir. Benzer argümanlar, yeni veri-tabanı teknolojisinin uygulamalarını icra eder, bir derleyiciden daha gerçekçidirler.[6]. Örneğin, bir CAD sisteminin birçok elemanında ad-hoc sorguları faydalı olmayabilir.

Zengin bir yapıya sahip uygulamalar için, verimlilik hayatidir ve ad-hoc sorgularına ihtiyaç yoktur, gereken bir "nesne tabanı" veya "nesneye dayalı bir alt yapı (back end)'dir". Bu çeşit bir sistem, önemli ve faydalı bir gelişmedir. Nesne-tabanları güçlü bir tip sistemini destekler; fakat direk olarak güçlü bir sorgu dilini desteklemez. Nesne tabanları bir UVT veya ND veri-tabanı sisteminin alt yapısına faydalı olabilir.

V.2. Ad-hoc Sorgularına İhtiyacı Olan ve Olmayan Uygulamalar :

Ad-hoc sorgularına ihtiyacı olan uygulamalar bildirimseldir, bir ND programlama dili ile bunu test etmeye çalışılırsa, sonuçta bunun mümkün olmadığı görülür (bir ND programlama dili ile ad-hoc sorgulamalı ve bildirimsel bir uygulama geliştirilemez). Bir uygulamayı geliştirirken, UVT veya ND sistemlerin güçlü sorgulama dilleriyle ve de ad-hoc sorgu becerileriyle nelere dikkat edildiğine bakalım.

Aşağıdaki sınıflar, planlanmamış ve beklenmeyen sorgulara cevap verecek sistemlere göre önerilmiştir :
1. Bilimsel veri-tabanları : Bu sınıftaki uygulamalar, veriler arasında yeni ilişkiler bulunmasını sağlar. Bilimsel buluşların doğasından dolayı, hangi soruların sorulacağını önceden bilemeyiz.

2. Ticari veri-tabanları : Ticari verilerin ad-hoc sorguları değişik seviyelerde finansal kararları geliştirmeye izin verir. Örneğin, ticari pazarlardaki yatırımcıların bazı kalıpları keşfetmeye ihtiyaçları vardır ve onları başkalarından önce kullanmak isterler. Önceden planlanmış bir sorgunun bir avantajı yoktur, çünkü pazarda , "yaygın bir bilgi" konumuna gelmiştir ve etkisini kaybeder. Bu yüzden, pazara yönelik veri-tabanları ad-hoc sorgulamalarını desteklemez[8].

V.3. Bildirimsellik ve Metotlar

ND sistemlere, UVT sistemlerinin bildirimsellik ve ad-hoc sorgulama becerilerini kazandırabilirmiyiz ? Bu, mümkün görünmektedir. ND sistemleri kuram aracı olan Smaltalk veya C++ programlama dilleri, Pascal veya C'den daha bildirimsel değillerdir[9].

Ancak, genelde nesneye dayalı programlama için önemli bir argüman; soyutlamanın yüksek seviyelerde ilerleyen sınıflarının yaratılmasına veri tipleri ve bu tiplerin operasyonlarına (ilişkisel veya mantıksal dillerde bulunan) sahiptir. Bu tamamen doğrudur. Örneğin, ilişkilerin bir sınıfının tanımlandığını düşünelim, birleştirme (join) metodu ile, S ilişkisinin R ilişkisine birleştirilmesi için bir mesaj gönderelim (R JN S). Başarılı bir bildirimsel dilde, sorguları optimize edebilmeliyiz ve programcının planladığı zaman aralıklarında sonuçları alabilmeliyiz.

SQL ve benzeri ilişkisel diller muhtemelen, Pascal gibi diğer dillerden sorguları optimizasyona daha yatkındır. İlişkisel diller tamamen Turing değillerdir, dolayısı ile ilişkisel model veya eşdeğeri mantıksal kurallar, oldukça açık basit yazılımlar ve göreceli olarak basit stratejiler izlerler, ilişkisel cebrin verilen bir ifadesinin eşdeğer optimalini üretmek için yukarıdakileri kullanırlar.

Örnek 1: Üç ilişkinin birleşimini (join) ele alalım. R(A,B), S(B,C) ve T(C,D) ilişkileri A ve D sıfatları üzerine iz düşürülürse (cebirde $PJ_{A,D}$ (R JN S JN T)) mantıksal kural olarak eşdeğeri;

cevap(A,D) :- r(A,B), s(B,C), t(C,D).

Veriye bağlı olarak R ile S ilişkilerini ilk önce birleştirmek daha verimli olabilir, veya S ile T ilk olarak birleştirilebilir. Bir optimizer, verinin istatistiklerine göre kolayca bir seçim yapabilir. Belki de birinci birleştirme ikincisinden daha maliyetlidir, dolayısı ile bu optimizasyon hayati önemlidir. Cebirdeki veya mantıktaki açıklamalar, notasyonun yazılımılarını daha açık yapar.

Bir ND sistemde, ikili bir ilişki gibi davranan bir sınıf ve bu sınıf için "birleştir (join)" metodu tanımlansın . Dolayısı ile ifade şöyle yazılabilir :

R birleştir S birleştir T

Bu ifade ya

(R JN S) JN T

ya da

R JN (S JN T)

tipinde hesaplanabilir.

Bu değişme özelliğini ND sistemde kullanabilirmiyiz ? Bir ND yöntemi programlamada kullanmak istersek, ve program ilişkiler seviyesindeyse sisteme en azından ilişkisel cebrin, cebirsel kurallarını yerleştirmeye ihtiyaç vardır. Bu bilgiler sistem tarafından çıkarılamaz.

VI. UVT VE ND SİSTEMLERDE MODELLEME

Başlangıçta, veri modelleri bir "eleman kümesi" veya benzer varlıkların koleksiyonunun bir notasyonunda bir araya getirildi. İlişkiler, ilişkisel modeldeki bir eleman kümesinin simgesidir ve yüklemler UVT sisteminde aynı işi yaparlar. Açıkça, sınıf ND sistemde işaret edilen notasyondur. Ancak bu çeşitli eleman kümelerinin simgelerinin farklı derecede esneklileri ve kullanım kolaylıkları vardır[4].
Bir ilişkinin sabit sayıda argümanları vardır ve argümanlar verilmiş sıfat isimleridir.Bir yüklem bir ilişki simgeler ve de sabit argümanlıdır, fakat argümanlarının isimleri yoktur ve anlam bir argümanın pozisyonu ile birleştirilir. Bir sınıfın genellikle elemanları sabit kümeler değildir, çünkü alt sınıflar ile farklı kümelerin alanlarına sahip olabilir. Elemanlar pozisyonlarından çok isimleriyle bilinirler.

Özellikle bir eleman kümesinin birçok sıfatı (attribute) varsa bir listedeki elemanları pozisyonları yerine isimleri ile kullanmak daha uygun olabilir. Bu ilişkisel notasyonun mantıksal notasyonu üzerindeki özelliğidir. Ancak, elemanları pozisyonları ile işaret etmek kötü bir yöntem değildir.

Elemanlar için isim/pozisyon probleminden çok daha önemlidir. Elemanlar başka eleman kümelerinin alt özelliklerine sıfatlarına veya metotlarına bir varlık kümesine birleştirilerek eleman kümelerini genişletebilirler.Bir metodun birçok genel halinde, bir ilişkisel sistem, ND sisteme benzetilemez; ilişkisel diller yeterince açıklayıcı değillerdir.

Eleman kümelerindeki değişkenliğe, ND sistemlerinde alt sınıf sistemi ile erişilir. Sınıflar, metotların farklı kümeleri ile alt sınıflara bölünür. UVT sistemleri için bir kaltım (inheritance) mekanizması aynı amaç için kullanılır.

Örnek 2: Aşağıdaki gibi bir yüklemimiz olsun,

öğrenci(kod,a)

kod, öğrencinin numarası ve a bir başka ilişkinin ismini göstermek üzere yukarıdaki bahsedilenleri desteklemektedir.

VII. ÖNERİLER

Arity Prolog mantıksal programlama dilini ve SQL paketini kullanarak ilişkisel veritabanları ile uzman sistemler arasında bir ilişki kurmaya çalıştık. Bu iki sistem arasında bir front-end aracılığı ile doğal dil işlemede kullanıldığı gibi SQL komutları ile mantıksal dil komutları arasında bir dönüştürme yapıldı. Bu dönüştürmeler esnasında bütünleştirme sınırlamalarının güçlendirilmesi için ND sistemlerin özelliklerinden olan dinamik tip bildirimi, ND sistem geliştirme aracı kullanmaksızın kısmen gerçekleştirilmeye çalışılmıştır. Ullman 'nın fikirlerini temel alarak, mantıksal programlama ile nesneye dayalı programlama çevresinin bir kısmının kullanımından ne gibi sonuçların elde edilebileceği araştırıldı. Bu çalışmalar sırasında karşılaşılan problemlerden yola çıkarak şu önerilerde bulunabiliriz [7] :

Ad-hoc sorgularına ihtiyaç olan bölümlerde mümkün oldukça dinamik tiplerden kaçınılmalıdır. Eğer dinamik tip kullanımı zorunlu ise bir nesneye dayalı programlama dili C++ ile , mantıksal programlama dili Prolog kullanılarak bazı program parçalarını C++'da yazıp Prolog programının altına yerleştirmek mümkün olabilir.Bildirimselliğin kaçınılmaz olduğu durumlarda nesneye dayalı metotlardan kaçınılmalıdır.

VIII. SONUÇ

Bu makalede şu noktalar tartışılamaya çalışılmıştır.

UVT ve ND paradigmalarını birleştirmek, ND sistemlerin özelliklerinden olan nesne özdeşliğini (object identity) ve dinamik tip özelliklerinde ısrar edildiğinde mümkün değildir.

Bildirimselliğin ve ad-hoc sorgularının önemli olmadığı uygulamalar vardır. Bu durumlarda, bir ND sistem kullanılabilir. Bu gibi uygulamalarda, "nesne tabanına" veya "nesneye dayalı veri yüklemeye" gerçekten ihtiyaç vardır. Sorgu diline ihtiyaç yoktur.

Bildirimsellik ve ad-hoc sorgularının önemli olduğu uygulamalarda, nesneye dayalı bir sorgu dili kullanılamaz. Yüksek düzeyli operasyonlar sabit anlamda sisteme verilirse, sorgu işleme verimliliği artacaktır.

Tipik uzman sistem geliştirme özellikleri ekrana getirmeyi düzenlemek, hata kontorolu ve kural izleme gibi veri-tabanı etkileşimlerine yayılmalıdır. Veri-tabanı sistemindeki veri girişi veya rapor üretme gibi geleneksel modüller, bir uzman sistem ile etkileşerek modife de edilmelidir. Bilgi tabanlı sistemin çözümleme işlemi için bir açıklama mekanizması rapor ve uygulama üretme gibi veri-tabanı aktivitelerinde birleştirilmelidir. Bunları gerçekleştirirken dinamik nesne tipleri kullanılmalıdır.

Nesneye dayalı sistemlerin özelliklerinin çoğu uzman sistemlerde birleştirilebilir; fakat, bunun tersi doğru değildir.

KAYNAKLAR

[1]- Cacace F., Ceri S., Crespi-Reghizzi S., Tanca L., Zicari R. ; Integrating Object-Oriented Data Modeling With a Rule Based Programming Paradigm (press unknown),s.225

[2]-Parsaye Kamran - Chignell Mark- Khoshafian Setrag-Wong Harry; Intelligent Databases (New York, Wiley,1989)

[3]- Karahoca, A.; A Connection Between Expert Systems and Active Data-base Systems (Istanbul : M.S. thesis,Dept. of Computer Sci. Eng. I.Ü. ,1995)

[4]- Kim Won & H. Locohovsky Frederic ; Object-Oriented Concepts Database and Applications (New York : ACM Press, 1989)

[5]- Caseau Yves; Constraints in an Object-Oriented Deductive Database (press unknown),s.292-293

[6]- Date, C. J. ; An Introduction to Database Systems, (USA : Addison Wesley Publishing Company Inc. ,1986)

[7]-Ullman Jeffrey D.; A Comparison Between Deductive And Object Oriented Database Systems (press unknown),s.261-275

[8]-Özkarahan, Esen; Database Management (London :Prentice Hall International Edition,1990)

[9]-Van Le T. ; Techniques of Prolog Programing (Canada, John Wiley & Sons Inc.),1993