T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

NESNE TEMELLİ VERİTABANLARININ UYGULAMASI

YÜKSEK LİSANS TEZİ

ELEK. ELEKTRONİK MÜH. LEYLA KURU

Enstitü Anabilim Dalı : ELEKTRİK-ELEKTRONİK MÜH.

Enstitü Bilim Dalı : ELEKTRONİK MÜH.

Bu tez 6../.8./1998 tarihinde aşağıdaki jüri tarafından Oybirliği/~~Oyçokluğu~~ ile kabul edilmiştir.

Jüri Başkanı	Jüri Üyesi	Jüri Üyesi
Prof. Dr. Füsun SELÇUK	Prof. Dr. A. Ferit KONAR	Doç.Dr. Ahmet BABANLI

ÖNSÖZ

Nesne temelli veri tabanları, geleneksel veri tabanı programlarından farklı olarak kullanıcının hem tasarım anında sistem yapısını kurmasını kolaylaştırmakta hem de sonradan sisteme çok kolay bir şekilde eklemelerin yapılabilmesine imkanı sağlamaktadır. Bu nedenle nesne temelli veri tabanları, veri tabanı problemleriyle ilgilenen kullanıcıların vazgeçemeyecekleri bir teknik olarak ortaya çıkmaktadır.

Çalışmalarımda her türlü teşvik ve fedakarlığı esirgemeyen, bilgi ve tecrübelerinden istifade ettiğim kıymetli hocalarım Prof. Dr. Füsun SELÇUK, Doç. Dr. Ahmet BABANLI ve Prof. Dr. A. Ferit KONAR' a sonsuz teşekkürlerimi sunuyorum.

Leyla KURU

İÇİNDEKİLER

KISALTMALAR

OOP	: Object Oriented Programming
OODB	: Object Oriented Database
OODBMS	: Object Oriented Database Managment System
SQL	: Structured Query Language
OOT	: Object Oriented Technology
OMG	: Object Managment Group
ODB	: Objectivity Database
ACID	: Atomicity, Consistency, Isolation and Durability
MDI	: Multiple Document Interface
OLE	: Object Linking and Embedding
VBX	: Visual Basic Custom Control
DLL	: Dynamic Link Library
API	: Application Programming Interface
RAD	: Rapid Application Development
LDM	: Logical Data Model
FDM	: Functional Data Model

ŞEKİLLER LİSTESİ

ÖZET

NESNE TEMELLİ VERİTABANLARININ UYGULAMASI

Anahtar kelimeler: veri kapsaması, bilgi gizleme, veri soyutlaması, veri tabanı, durgun veri, verinin sürekliliği, çokşekillilik, kalıtsallık, temel tablolar, türetilmiş tablolar, veri bütünlüğü, veri yönetimi, veri yapısı, verinin paylaşılması, verinin korunması, sorgu dili, nesne temelli programlama, nesne bağlama, nesne gömme, çoklu çalışma.

Bu çalışmada nesne temelli veritabanlarının karmaşık problemlerin çözümünde sağladığı üstünlükleri göstermek maksadıyla bir nesne temelli veritabanı tasarlanmış ve bu uygulamanın gerçeklenmesi sağlanmıştır. Genel hatları ile ilişkili veritabanları ve nesne temelli programlamadan söz edilmiş ve nesne temelli veritabanı sistemleri ele alınmıştır. Geliştirilen uygulama ile nesne temelli veritabanı sistemlerinin özellikleri sergilenmeye çalışılmıştır.

SUMMARY

APPLICATION OF OBJECT ORIENTED DATABASES

Keywords: data encapsulation, information hiding, data abstraction, database, static data, persistance of data, polymorphism, inheritance, base tables, derived tables, data integrity, data manupulation, data structure, data sharing, protection of data, the impedance mismatch, query, structured query language, komplex data, atomicity, consistency, isolation, durability, locks, version control, multi-tasking, object linking and embedding, dynamic link library.

In this work, an application is designed and executed to show the advantages of object oriented database systems on the solutions of complex problems. Relational databases and object oriented programming are introduced in general. While the methedology of appliance and development is being given, the steps of forming the foundation of object oriented databases and the facilities which are supplied for the user are examined.

BÖLÜM 1. GİRİŞ

Kompleks ve uzun olan veritabanı programları için uygun olan yapısal programlama tekniği, her ne kadar mükemmel sonuç veriyorsa da yine de belli bir limiti aşan programlar için yetersiz kalmaktadır. Çok daha karmaşık veritabanı programlarının yazılabilmesi için, ki teknolojik gelişim bu tip programların yazılmasını gerekli görmektedir, programlama işine yaklaşımda yeni bir metoda yönelmek gerekmiştir. İşte bu aşamada "Nesne Temelli Veritabanı" kavramı ortaya çıkmıştır.[1]

Bölüm 2'de veritabanı tanımı yapılarak, genel karakteristikleri anlatılacaktır. İlişkili veritabanı (Relational database) teorisi örneklerle açıklanıp ilişkili veritabanı modelinde verilerin tablo vasıtasıyla gösterildiği ve verilerin yönetiminin de çeşitli operasyonlarla sağlandığı anlatılacaktır.[1]

Bölüm 3'de nesne temelli programlama ortamı tanıtıldıktan sonra nesne temelli veritabanı sistemleri açıklanacaktır. Bu bölümde nesne temelli veritabanı sistemlerinin genel karakteristikleri üzerinde durulacak ve karmaşık problemler karşısında nesne temelli veritabanlarının sağladığı kolaylıklar anlatılacaktır.[4]

Bölüm 4'de Windows'un temel özellikleri verilerek yaygın nesne temelli programlama dillerinden olan; Visual Basic, C ++, Delphi ve Access programlama ortamları tanıtılacaktır.[2]

Bölüm 5'de nesne temelli veritabanlarının karmaşık problemlerin çözümünde sağladığı kolaylıkların görülmesi amacıyla Access ortamında bir veritabanı uygulaması tasarlanacaktır. Bu uygulamada tablolar vasıtasıyla oluşturulan veriler arasında ilişkiler oluşturarak yeni veritabanları elde edilecektir.[7]

Bölüm 6'da da Bölüm 5'de geliştirilen uygulama gerçeklenecek ve yapılacak sorgular ile de geliştirilen uygulamanın özellikleri sergilenecektir.

Böylece tezin amacı olan; nesne temelli veri tabanları tanıtılmış olacak ayrıca nesne temelli veritabanlarının nasıl kullanıldığı ve karmaşık problemlerin çözümlerinde sunduğu avantajlar açıklanmaya çalışılacaktır.

BÖLÜM 2. İLİŞKİLİ VERİTABANLARI

2.1 Veri Tabanı Nedir?

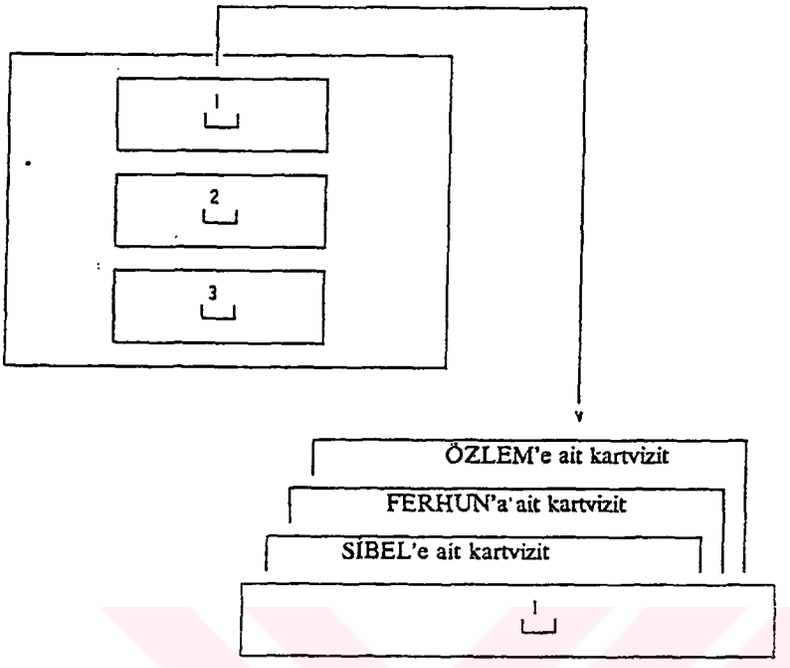
Gerektiği zaman; listelemek, sıralama yapmak veya arama yapmak için düzenlenmiş ve birbiriyle bağlantısı olan veriler topluluğuna Veri Tabanı (database) denilmektedir.[3]

Dosyalarda kayıtlı bilgilerin bilgisayar aracılığıyla sağlıklı olarak muhafaza edilebilmesi, bilgisayarların yaygın bir şekilde kullanılmaya başlanmasıyla birlikte daha da önem kazanmaya başladı. Önceleri Basic, Cobol, Pascal gibi geleneksel bilgisayar programlama dilleri ile sayfalar dolusu programlar yazılarak bilgiler depolanabiliyor, bu bilgilere çeşitli şekillerde ulaşabilmek mümkün olabiliyordu. Ancak kullanım kolaylığı, indekslemelerin yapılması, sayısal değerlerin etkin bir şekilde kullanılması, pratik ve kolay raporlar hazırlanması gibi ihtiyaçlar belirdikçe, bu dillerle yazılan programların yetersiz kaldığı görüldü. Bu alandaki ihtiyacın karşılanabilmesi için Veri Tabanı programları ortaya çıkmıştır. Veri tabanı kavramının daha iyi anlaşılabilmesi için şu şekilde basit bir örnek verilebilir:
Şekil 2.1'de görülen kartvizitlerin olduğu kabul edilsin.

FERHUN ÇETİN İncirli cad.No 1 B.KÖY TEL:572 33 33	ŞENGÜL AKCAY Altıyol cad.No 52 K.KÖY TEL:356 77 88	SEMA ESEN Ata cad.Ata ap No:10 AVCILAR TEL:591 24 42
ÖZLEM KIRCALI İncirli cad.No 27 B.KÖY TEL:572 42 67	ÇETİN METİN Nispetiye cad.No:54 LEVENT TEL:169 00 77	SİBEL DOĞAN Ortaköy cad.No 33 O.KÖY TEL:159 88 56

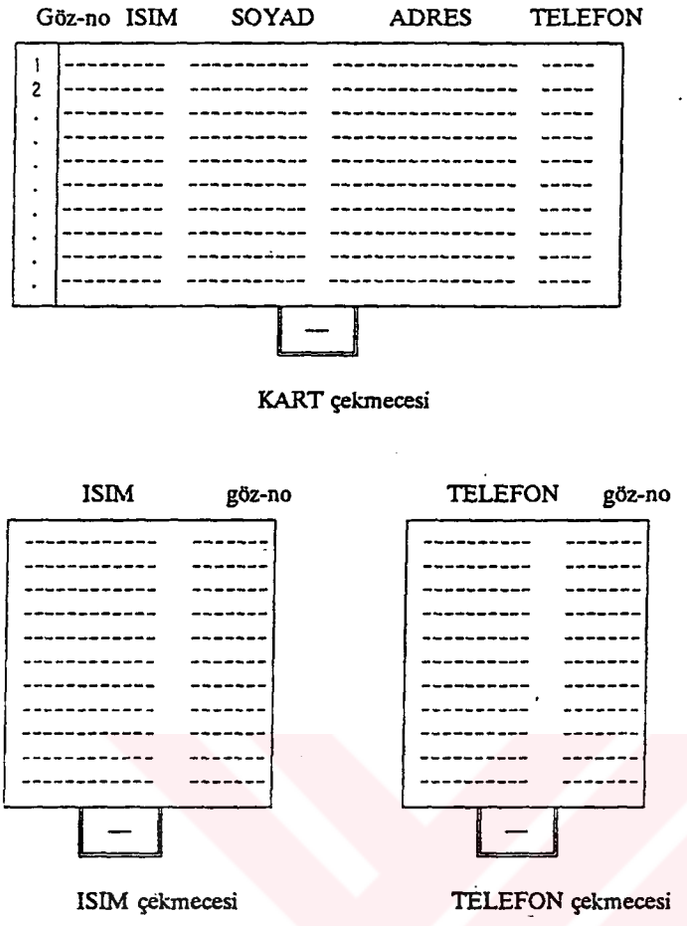
Şekil 2.1 Farklı kişilere ait kartvizitler

Şekil 2.1'deki kartvizitler Şekil 2.2'deki kart çekmecelerinden birine saklansın.



Şekil 2.2 Kart çekmeceleri

Aranılan bir kartın isme ve telefona göre kolayca bulunulması için 2 küçük çekmece daha düşünülsün. İSİM ve TELEFON çekmeceleri. KART çekmecesinde her kartvizite ait kartonun numaralanabileceği şekilde bölme olsun. Ayrıca her kartviziti içerecek kartonda da ismi, soyadı, adresi ve telefonu içerecek birer alan bulunsun. Diğer taraftan İSİM çekmecesinde ismi ve KART çekmecesindeki kartonun numarasını (göz-no); TELEFON çekmecesinde de telefonu ve KART çekmecesindeki kartonun numarasını (göz-no) içerecek alanlar bulunsun (Şekil 2.3).



Şekil 2.3 KART, İSİM ve TELEFON çekmeceleri

Şimdi KART çekmecesinden boş olan 1. Karton alınır ve 1. Karta göre İSİM, SOYAD, ADRES ve TELEFON için ayrılmış alanları yazılarak yerine yerleştirilir. Sonra İSİM çekmecesinden bir karton alınarak İSİM ve KART çekmecesindeki kartonun numarasına göre Göz-No yazılarak yerine yerleştirilir. Daha sonra ise TELEFON çekmecesinden bir karton alınarak TELEFON ve KART çekmecesindeki kartonun numarasına göre Göz-No yazılarak yerine yerleştirilir.

Birinci kartın yerleştirildiği şekilde 2, 3, 4 ve 5. Kartlar da yerleştirilir. Böylece Şekil 2.4'deki gibi kartlar elde edilmiş olunur.

KART Çekmecesi

	ISIM	SOYAD	ADRES	TELEFON
1	FERHUN	ÇETİN	İncirli Cad. No: 1 B.Köy	572 33 33
2	ŞENGÜL	AKKAYA	Altıyol cad.No 52 K.KÖY	356 77 88
3	SEMA	ESEN	Ata cad.Ata ap No:10 AVCILAR	591 24 42
4	ÖZLEM	KIRCALI	İncirli cad.No27 B.KÖY	572 42 67
5	ÇETİN	METIN	Nispetiye cad.No:54 LEVENT	169 00 77
6	SİBEL	DOĞAN	Ortaköy cad.No33 O.KÖY	159 88 56
I	--------	----------	----------	--------
v	--------	----------	----------	--------
	--------	----------	----------	--------

ISIM Çekmecesi

ISIM	göz-no
ÇETİN	5
FERHUN	1
ÖZLEM	4
SEMA	3
SİBEL	6
ŞENGÜL	2

TELEFON Çekmecesi

TELEFON	göz-no
159 88 56	6
169 00 77	5
356 77 88	2
572 33 33	1
572 42 67	4
591 24 42	3

Şekil 2.4 Elde edilen kartlar

İşte buradaki kartların bilgisayarda oluşturulduğu düşünüldüğünde KART çekmecesi bir Veri Tabanı olmaktadır. İSİM ve TELEFON dosyaları ise veri dosyası olmayıp, aranan kaydın veri dosyasındaki numarasını içermektedirler.[3]

2.1.1 Veri Tabanının Sağladığı Olanaklar

- Verileri içerecek dosyaya bir isim vererek bu dosyayı oluşturmak ve kullanmak.
- Dosyanın yapısını disk veya diskette saklamak.
- Gerektiği zaman dosyanın yapısını değiştirmek.

- Yapısını oluşturduğumuz dosyaya kayıt girerek bilgileri disk veya diskette depolamak.
- Girilen kayıtlardan istenilene ulaşmak.
- Ulaşılan herhangi bir kaydı değiştirmek veya gözlemek.
- Bir kaydı girerken, düzeltirken veya gözlerken istediğimiz ekran formunu oluşturmak.
- İşleme sokmak istemediğimiz kayıtları tamamen iptal etmek.
- Veri dosyasını istenilen bilgiye göre sıraya sokmak.
- İstenilen kayıtlardan yalnızca verilen koşulları sağlayanları ekran veya yazıcıya dökmek.
- Yalnız istenilen bilgileri görüntülemek.
- Bu dökümlerin çıktısını istenilen formda almak.
- Bu dökümleri istenilen sırada almak.
- Bir dosyadan diğer bir dosyaya bilgi aktarmak.[3]

2.2 İlişkili Veritabanı Sistemleri (Relational Database Systems)

Veritabanı sistemleri bir dizi farklı yaklaşımlara bağlı olabilirler. Burada sözü edilecek veritabanları ilişkili veritabanı (relational databases) sistemleri olacaktır. İlişkili veritabanı sistemleri 1970'lerin sonunda ortaya çıkmıştır. İlişkili veritabanı sistemi şunları içeren bir sistemdir:

• Veri kullanıcı tarafından yalnızca tablo (table) olarak algılanır

• Kullanıcı düzeni üzerindeki operasyonlar, eski tablolardan yeni tablolar üreten operasyonlardır. Bu operasyonlar SELECT, PROJECT ve JOIN'dan oluşmaktadır.

Bir örnek ilişkili veritabanı sistemi şekil 2.5'de görülmektedir. Bu veritabanı bölümler (departments) ve işçilerden (employees) oluşmaktadır. Buradan da görüldüğü gibi veritabanı tablo şeklindedir. Dept# veritabanının D1, D2 ve D3'den oluşmuş 3 bölümü vardır. Bu bölümlerin özellikleri Dname (bölüm adı) ve Budget (bütce) olarak belirlenmiştir. Aynı şekilde Emp# veritabanıda E1, E2, E3 ve E4'den oluşmuş işçileri vardır. Bu işçilerin özellikleri de Ename (işçi adı), Depth# (işçinin ait olduğu bölüm) ve salary (maaş)'dan oluşmaktadır.

DEPT

DEPT#	DNAME	BUDGET
D1	Marketing	10M
D2	Development	12M
D3	Research	5M

EMP

EMP#	ENAME	DEPT#	SALARY
E1	Lopez	D1	40K
E2	Cheng	D1	42K
E3	Finzi	D2	30K
E4	Saito	D2	35K

Şekil 2.5 Bölümler ve işçiler veritabanı

Şekil 2.6'da select, project ve join operasyonlarına ilişkin veritabanları görülmektedir. Select operasyonu veritabanındaki belirli satırları açmak için kullanılır. Project operasyonu veritabanındaki belirli sütunları açmak için kullanılır. Join operasyonu ise iki tabloyu ortak değerleri olan sütun altında birleştirmekte kullanılır.[1]

SELECT (RESTRICT): Result:

DEPTs where BUDGET > 8M

DEPT#	DNAME	BUDGET
D1	Marketing	10M
D2	Development	12M

PROJECT: Result:

DEPTs over DEPT#, BUDGET

DEPT#	BUDGET
D1	10M
D2	12M
D3	5M

JOIN:

DEPTs and EMPs over DEPT#

Result:

DEPT#	DNAME	BUDGET	EMP#	ENAME	SALARY
D1	Marketing	10M	E1	Lopez	40K
D1	Marketing	10M	E2	Cheng	42K
D2	Development	12M	E3	Finzi	30K
D2	Development	12M	E4	Saito	35K

Şekil 2.6 Select, project ve join operasyonları

Select operasyonu ile bütçesi 8 milyondan büyük olan satırlar seçilmiştir. Project operasyonu ile Depth# ve bütçe sütunları görüntülendirilmiştir. Dept ve Emp veritabanının Depth# adında ortak bir sütunu vardır. Öyleyse bu iki veritabanı bu ortak sütundaki ortak değerler altında birleştirilebilir. Şekilde bu görülmektedir. Burada Dept tablosundan verilmiş bir sıra ile Emp tablosundan bir sıra ile birleştirilmiştir. Böylece daha geniş yeni bir veritabanı üretilmiştir. Burada dikkat edilmesi gereken şey sütun içinde yalnız ortak olan değerlerin birleştirilebileceğidir. Dept tablosunda D3 diye bir bölüm olmasına rağmen Emp tablosunda D3 diye bir satır olmadığı için birleştirilmiş tabloda da D3 diye bir değer görülmemektedir. Her bir operasyon sonucu yeni bir tablo oluşmaktadır. Bu ilişkili veritabanlarının closure özelliğinden kaynaklanmaktadır.[1]

Şekil 2.6'da görülen bir başka özellik set-at-a-time özelliğidir. Bunun anlamı operasyonlar ve sonuçların hepsi tek bir satır halinde değil bir tablonun tümü şeklindedir. Tablolarda bir dizi satırlardan oluşmaktadır. Örneğin join operasyonu 3 ve 4 satıra sahip 2 tablo üzerine uygulanmış ve sonuç 4 satırlı yeni bir tablo şeklindedir. Bu set işleme kapasitesi (set processing capability) ilişkili veritabanlarının temel özelliklerindendir.[1]

İlişkili veritabanları için şunlar geçerlidir:

- İlişkili veritabanları yalnızca tablo şeklinde algılanabilirler. Tablolar ilişkili sistemdeki fiziksel değil mantıksal yapılardır. Sistem fiziksel yapısında istediği yapıları kullanabilir. Sıralama, indeksleme gibi. Burada fiziksel yapının tablo halinde olan mantıksal yapıya uyarlanabilmesi gerekir. Bir başka deyişle tablolar verinin fiziksel yapısından soyutlanmışlardır.
- Bir başka özellik veritabanında tüm bilginin bir ve yalnızca bir yolla gösterilebilmektedir. Bu da kesin veri değeri ile olmaktadır. Bu gösterim metodu ilişkili veritabanındaki tek yöntemdir. Bu gösterimde bir tabloyu diğer bir tabloyla ilişkili olup olmadığını gösteren işaretçiler yoktur. Bunu yerine verinin kendi değeri vardır. Örneğin Dept tablosunun D1 satırı ile Emp tablosunun E1 satırı arasında bir ilişki vardır. Çünkü E1 işçisi D1 bölümünde çalışmaktadır. Bu ilişki D1 değeri ile

Emp tablosundaki Dept# bölümünde gösterilmiştir. İlişkili olmayan veritabanlarında bu böyle değildir. Orada işaretçiler vardır. İlişkili veritabanı sistemlerinde işaretçiler yalnızca fiziksel kısımda olabilir.

• Bütün veri değerleri atomiktir. Bunun anlamı her bir veritabanındaki tablonun satır veya sütunda tek bir veri değerinin olmasıdır. Grup verisi bulunmamasıdır.

2.2.1 İlişkili model

İlişkili model veriye bakmanın bir yolu olarak tarif edilebilir. Yani; verilerin tablo vasıtasıyla gösterildiği ve yönetimde çeşitli operasyonların bulunduğu bir reçete olarak da tarif edilebilir. Daha doğru bir ifade ile ilişkili model verinin 3 görünüşü ile ilgilenecektir: Veri yapısı (Data structure), veri bütünlüğü (data integrity), ve veri yönetimi (data manipulation).

Veritabanı birçok bir çok bütünlük kuralları taşımaktadır. Örneğin şekil 2.5'de görüldüğü gibi dept veritabanında budget sütununda yer alan bilgiler belli bir sırada olmaktadır. İşçi maaşlarının da belli bir sırada olması gerekmektedir. Veritabanının ilişkili olabilmesi için:

• Dept tablosundaki her bir satır tek bir dept# değeri taşımalıdır. Aynı şekilde Emp tablosundaki her bir satır tek bir Emp# değeri taşımalıdır.

•Emp tablosundaki her bir Dept# değeri Dept tablosundaki Dept# değeri olarak yer almalıdır. Yani her bir işçinin mutlaka bir bölümü olmalıdır.

Dept tablosundaki Dept # sütunu ile Emp tablosundaki Emp# sütunu bulundukları tabloların ana anahtarıdırlar (primary key). Emp tablosundaki Dept# sütunu ise yabancı anahtardır (foreign key). Ana anahtarların altı çift çizilmiştir.[1]

2.2.2 Optimizasyon

İlişkili operasyonlar set-level (seviye set) operasyonudurlar. SQL gibi ilişkili diller çoğu zaman kuralsız (non procedural) olarak bilinirler. Kullanıcı istediği şeyi

procedur yazmadan elde etmektedir. Kullanıcı isteğini yerine getirmek için yüklenmiş veritabanında dolaşma işlemi sistem tarafından otomatik olarak yapılmaktadır. Bu nedenle ilişkili sistemlere otomatik dolaşma sistemleri (automatic navigation systems) de denilmektedir. Otomatik dolaşmanın nasıl olacağına karar vermek optimizer denilen Database management systemin (DBMS) bir nesnesinin görevidir. Bir başka deyişle kullanıcının her bir ilişkili isteği optimizer tarafından değerlendirilir ve en uygun yolla cevap yine optimizer tarafından verilir. Aşağıdaki örneği ele alalım:

Result:=(EMP WHERE EMP#='4') [Salary];

Burada "(EMP WHERE...)" isteği bir seçim (select)'dir. Emp tablosunda Emp# değeri 4 olan satır seçilmek istenmektedir. Köşeli parantez içindeki sütun ismi "salary" ise isteğin sonunda yalnızca bu sütunun görüntülenmesi isteğini ifade etmektedir. ":=" işareti ise sonucu result adlı bir tabloya aktarmaktadır. Sonuçta cevap tek bir satır ve sütundan oluşur ve E4 işçisinin maaşını gösterir.

Önemli veri erişimi sağlayan en az 2 yol vardır:

1- Emp tablosu üzerinde istenen kayıt bulununcaya kadar sıralı fiziksel bir tarama yapılarak
2- Ana anahtar (primary key) üzerinde indeksleme kullanılmış ise indeks kullanılarak doğrudan E4 verisine ulaşılabilir.

Optimizer tercihini yaparken şunlara dikkat eder:

- İstekte hangi tablolar yer almış
- Tabloların büyüklükleri nedir
- Ne tür indeksleme yapılmıştır
- Bu indeksleme ne kadar seçicidir
- Verinin disk üzerindeki fiziksel yapısı nedir
- Ne tür ilişkili operasyonlar kullanılmıştır

2.2.3 Katalog

Her DBMS mutlaka Katalog (catalog) yada sözlük (dictionary) sağlamalıdır. Katalog bütün şemaların bulunduğu bir yerdir. Daha açık bir ifade ile katalog sistem nesneleri hakkında detaylı bilgilerin bulunduğu bir yerdir. Bu nesneler tablolar, indekslemeler, kullanıcılar, bütünlük kuralları, güvenlik kuralları ve benzerleridir. İlişkili sistemlerde kataloğun kendisiside tablolardan oluşmuştur. Şekil 2.7'de Dept ve Emp veritabanının katoloğu bir tablo olarak gösterilmiştir. Burada katalog Tables ve columns adıyla 2 tabloda verilmiştir. Tables adıyla verilen veritabanında yer alan tabloların adları ve bu tablolarda bulunan satır ve sütun sayılardır. Columns tablosuyla verilen tabloda ise Dept ve Emp veritabanındaki sütun isimleridir.[1]

TABLES

TABNAME	COLCOUNT	ROWCOUNT	.....
DEPT	3	3	.....
EMP	4	4	.....
.....	....	....	.....

COLUMNS

TABNAME	COLNAME	.....
DEPT	DEPT#	.....
DEPT	DNAME	.....
DEPT	BUDGET	.....
EMP	EMP#	.....
EMP	ENAME	.....
EMP	DEPT#	.....
EMP	SALARY	.....
.....	.....	.....

Şekil 2.7 Dept ve Emp veritabanının katolog yapısı

2.2.4 Temel tablolar ve türetilmiş tablolar

Verilmiş olan veri tabanı tabloları kullanılarak başka tablolar elde edilebileceği görüldü. Verilmiş olan tablolara yani orijinal tablolara temel tablolar (base tables) denilmektedir. Bu tablolardan faydalanarak oluşturulan yeni tablolara türetilmiş tablolar (derived tables) denilmektedir. Temel tablolar bağımsız türetilmiş tablolar ise bağımlı tablolardır.[1]

İlişkili sistemler temel tablolar oluşturmaya ortam sağlamak zorundadırlar. SQL ilişkili dilinde bu Create komutu ile yapılmaktadır. Burada table olarak geçen tablolar temel tablolardır ve bu tablolar isimlendirilmelidir. Çoğu türetilmiş tablolar ise isimsizdir. İlişkili sistemler view adı verilen türetilmiş tabloları desteklemektedirler. View adlandırılmış olan fakat bağımsız olmayan bir türetilmiş tablodur. Örneğin;

CREATE VIEW TOPEMPS AS

(EMP WHERE SALARY >3K) [EMP#, ENAME, SALARY]

Bu örnekte adı topemps olan bir view üretilmiş olmaktadır. Şekil 2.8'de görüldüğü gibi gölgesiz bölümler bu program satırlarının işletilmesi sonucu oluşmuştur.

TOPEMPS

EMP#	ENAME	DEPT#	SALARY
E1	Lopez	D1	40K
E2	Cheng	D1	42K
E3	Finzi	D2	30K
E4	Saito	D2	35K

Şekil 2.8 Topemps adlı view türetilmiş tablonun oluşturulması

Temel tablolar ile viewler arasındaki farklar:

- Temel tablolar gerçekten vardırlar ve gerçek veriyi veitabanında saklarlar.
- Viewler ise gerçekte yokturlar fakat gerçek veriye farklı yollardan göz atmaya yarar.

2.2.5 SQL dili

SQL dili ilişkili operasyonları formüle etmekte kullanılır. Dept ve emp veritabanının SQL dilindeki veri tanımlama operasyonları nasıl tanımlandığı şekil 2.9'da gösterilmiştir. Her bir tablo için "create table" komutu kullanılmıştır.

Her bir komut temel tablonun ismini, sütunların isimlerini, veri tiplerini, ana anahtarları ve yabancı anahtarları tanımlamaktadır.[1]

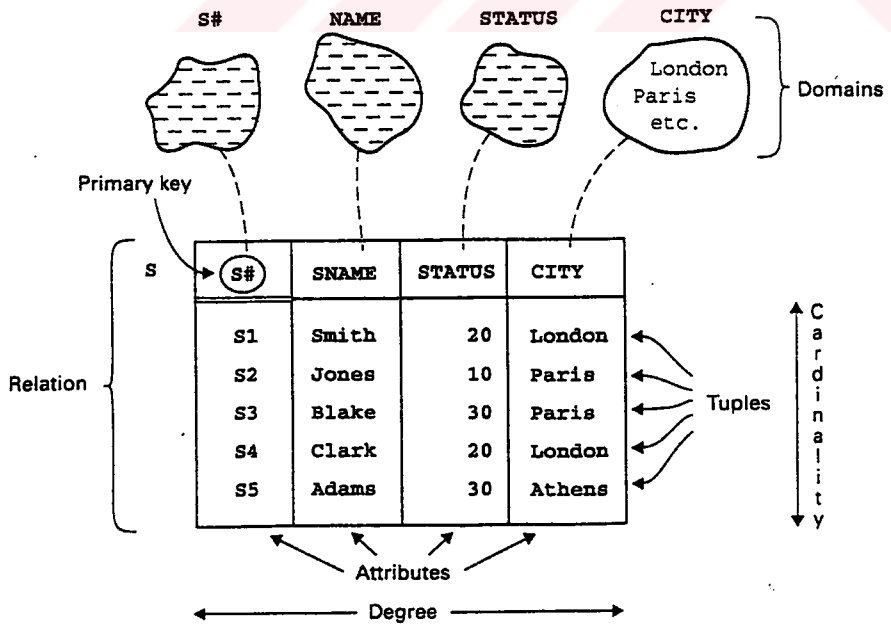
```
CREATE TABLE DEPT
     ( DEPT#    CHAR(2),
       DNAME    CHAR(20),
       BUDGET   DECIMAL(7),
     PRIMARY KEY ( DEPT# ) ) ;

CREATE TABLE EMP
     ( EMP#     CHAR(2),
       ENAME    CHAR(20),
       DEPT#    CHAR(2),
       SALARY   DECIMAL(5),
     PRIMARY KEY ( EMP# ),
     FOREIGN KEY ( DEPT# ) REFERENCES DEPT ) ;
```

2.9 Dept ve Emp veritabanının SQL'deki tanımı

2.2.6 Domen ve ilişkiler

İlişkili model 3 bölüme ayrılır: nesneler (objects), bütünlük (integrity) ve operasyonlar (operators). Her üç bölümün kendine has temelleri vardır. İlişkili modelde nesneler bölümünde en sık kullanılan terimler şekil 2.10'da gösterilmiştir.



Şekil 2.10 İlişkili modelde nesneler bölümünde kullanılan terimler

Bu terimlerin anlamları:

- Relation (ilişki): Tablo olarak düşünülmektedir. Veriler arasındaki ilişkiler tablo şeklinde gösterilmektedir.
- Tuple: Tablodaki satırları ifade etmektedir.
- Attribute: Tablodaki sütunları ifade etmektedir.
- Primary Key: (Ana anahtar): Tabloda altı çift çizgi ile gösterilmektedir. Tabloda her biri farklı olan özelliği göstermektedir.
- Domain (Domen): Bir veri havuzudur. Bu havuzda belli ilişkileri olan sütunlar bulunmaktadır.
- Degree (derece): Tablodaki sütun sayısıdır.
- Cardinality: Tablodaki satır sayısıdır.

2.2.7 Veri bütünlüğü (data integrity)

Herhangi bir zamanda verilmiş olan herhangi bir veritabanı genelde veri değerlerinin belli yapılarını içermektedir ve bu yapılar gerçeği yansıtmalıdır. Bu gerçek dünyanın model gösterimi olmalıdır. Fakat belirli verilerin yapıları gerçek dünyayla ilgili olmayabilir. Bu nedenle veritabanı tanımı doğruluk kurallarını içerecek şekilde genişletilmelidir ki saçma veri değerleri engellenebilsin. Veritabanları çok sayıda doğruluk kurallarından oluşmaktadır. Her doğruluk kuralı veri tabanı için özeldir. Her veritabanının farklı doğruluk kuralları vardır. Yalnızca ilişkili modellerde 2 ortak doğruluk kuralı vardır. Bunlar ana ve aday anahtarlar (Primary and candidate keys) ile yabancı anahtarlardır (foreign keys).[1]

BÖLÜM 3. NESNE TEMELLİ VERİTABANLARI

3.1 Nesne Temelli Programlama Nedir?

Nesne Temelli Programlama, program veya genel anlamda yazılım sistemlerinin geliştirilmesinde yeni bir yöntemdir. Nesne Temelli Programlamanın esas amaçları, yazılımın kolay büyümesini sağlamak, yazılımın karmaşıklığını azaltmak ve sistemlerin modellenmesini kolaylaştırmaktır. Nesne Temelli Programlama kullanıldığında, yazılım sisteminin tasarım aşaması, gerçekleme aşamasına daha sıkça bağlanmaktadır.

Nesne Temelli Programlamanın başarılı uygulamaları bugün artık birçok alanda görülmektedir.[2] Nesne Temelli Programlama kavramının başarılı olmasının arkasında temel düşünce tarzının insanların doğaları gereği sahip oldukları düşünce tarzına olan benzerliği de yatmaktadır. İnsanlar, çevrelerini nesneler topluluğu olarak algılarlar ve nesnelerin arasındaki ilişkileri sorgularlar. İnsanların problem çözmede kullandıkları temel birim nesnedir. Bu nedenle eğer programcı klasik programlama yöntemlerini bırakıp, nesne temelli bir platforma taşıyarak bu platformda problemlerini çözmeye çalıştığında, programcılık açısından yeni olan bu düşünce tarzının gerçekte kendi doğal düşünce tarzına çok benzer olduğunu görecektir. Bu düşünce tarzı benzerliği Nesne Temelli olarak problem çözümünün bilgisayar programcılığına uygulanmasını kolaylaştırmaktadır.

Nesne Temelli Programla, çok önemli ve temel niteliğinde birkaç kavram etrafında oluşmuştur. Esas konusu nesne kavramı ve nesne davranışıdır.[2]

3.2 Nesne Temelli Kavramlar

Nesne Temelli Kavramlar bir çok yazılım mühendisliği kavramlarının birleşmesinden oluşmuştur. Bu kavramlar Veri Kapsaması (data encapsulation), Bilgi Gizleme (information hiding), Veri Soyutlaması (data abstraction), Polymorphizm (Çok şekillilik), İnheritance (Kalıtsallık) gibi kavramlardır. Bu kavramlar nesneler, sınıflar ve sınıf hiyerarşisi kavramları ile birleştirilmiştir.[4]

Data Abstraction : Veri tipleri kavramının genişletilmesidir. Soyut veri tiplerinin tanıtılmasını sağlamaktadır. Soyut veri nesnesi soyut operasyonlarla nesne üzerinde tanımlanmaktadır.

İnformation Hiding : Bu bir yazılım tasarım tekniğidir. Bu teknik ile detaylı ve karmaşık uygulamalar saklanmakta böylece yazılım kolaylaştırılmaktadır. Bu aynı zamanda koruma maksadıyla da yapılmaktadır.

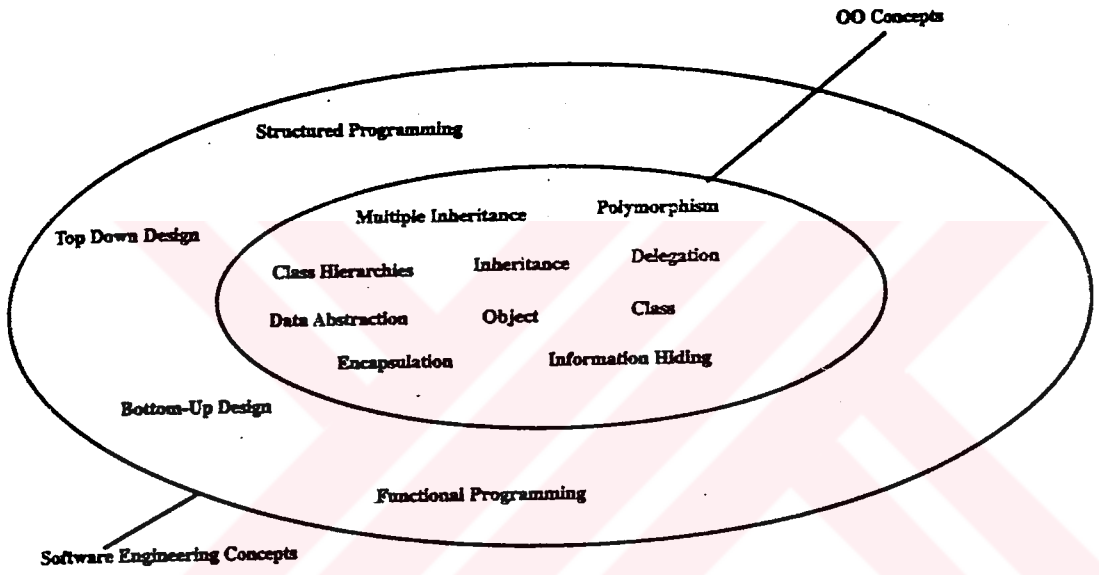
Encapsulation : Yazılım dünyasındaki nesneler, gerçek dünyadaki varlıklar gibi davranış gösteren program parçaları ve verilerdir. Bir nesnede onu tanımlayan bilgiler, veriler ve bunlar üzerinde işlem yapmaya yarayan metodlar bulunur. Nesnelerin en büyük avantajı, birçok kereler kullanılabilir olmasıdır. Böylelikle önceden hazırlanmış, deneyi yapılmış ve çalışır hale gelmiş programlar daha sonra çok kısa bir sürede yeni bir programa uyarlanabilir. Nesnelere mesaj göndermek ve bu mesajlara yapılan işlemlerin sonuçlarını görmek mümkündür.

Polymorphism : Aynı metodu farklı sınıflara uygulamak yeteneği veya farklı metodları aynı isimle farklı sınıflara uygulama özelliği polymorphism olarak adlandırılmaktadır. Bir ismin birbiri ile bağlantılı fakat aslında değişik olan birden fazla amaç için kullanılmasıdır. Çokşekilliliğin amacı; bir tek ismi genel bir işlemler kümesi tanımlamada kullanmaktadır.

İnheritance : Bir nesnenin bir başka nesneye ait özellikleri kullanabilmesi demektir. Bu özellik sınıflandırma kavramını sağladığı için çok önemlidir. Gerçek hayatta da birçok sınıflandırma örnekleri görmek mümkündür. Örneğin sarı uçak

uçak sınıfından belli bir nesnedir. Uçak nesnesi hava taşıtları sınıfına aittir. Sonuç olarak sarı uçak taşıt olduğundan taşıt sınıfının bazı özelliklerini almıştır. Bu sınıflandırma olmasaydı her nesne tüm özelliklerini açıkça belirtmek zorunda kalırdı. Fakat sınıflandırma kullanarak bir nesnenin yalnızca kendisini o sınıf içinde ayrıcalıklı yapan özelliklerini belirtmek yeterli olmaktadır.[4]

Şekil 3.1'de nesne temelli gösterimde birleştirilmiş yazılım mühendisliği kavramları görülmektedir.



Şekil 3.1 Nesne temelli gösterimde birleştirilmiş yazılım mühendisliği kavramları

3.3 Sınıf Kavramları ve Soyut Veri Tipleri

Soyut veri tipleri, Nesne Temelli Programlamanın merkezi olarak değerlendirilebilir Soyut bir veri tipi, bir veri tipinin bu veri tipi üzerinde uygulanabilecek fonksiyonlar kümesi ile bütünleştirilmiş halidir. Bu fonksiyonlar, Nesne Temelli Programlamada metodlar ya da operasyonlar olarak tanınırlar. Bundan böyle metod olarak adlandırılacak olan fonksiyonlar, soyut veri tipinin davranışını tanımlar ve belirlerler.

Birçok Nesne Temelli Programlama dilinde, soyut veri tipini sınıf kavramı temsil eder. Soyut veri tipinin veri yapısı ve gerçeklenmesine ilişkin ayrıntılar sınıf tanımı içinde tutulur. Bir sınıf tanımı, ayrıca tanımı yapılmakta olan soyut veri tipinin iç veri yapısını ve bu veri tipi üzerinde kullanılacak metodların topluca tanımıdır.[4]

Sınıf tanımı içinde bulunan bir başka bilgi de soyut veri tipi ile iletişim yaparken kullanılacak metodların nasıl kullanılacağı ve soyut veri tipinin, bu metodlara tepkisinin ne olacağıdır. Ohalde sınıfların bu görüşe göre iki parçadan oluştuğu düşünülebilir. Bir parçada sınıfın verileri diğer parçada da sınıfın metodları tutulmaktadır.

Bir başka açıdan bakıldığında, sınıfın başlıca iki bölümü vardır. Bunlardan biri sadece sınıf içinde erişilebilir olan bölümüdür ve "özel bölüm" adını alır. Bu bölüme dışarıdan erişmek için izin verilmez. Diğer bölüm ise, sınıfın dışarıdan da erişilmesine izin verilmiş olan bölümüdür ve "dışa açık bölüm" adını taşımaktadır. Her iki bölüm de veri ve metod saklayabilir.

Sınıfın içinde tanımlanmış olan metodlar, özel bölümde veya dışa açık bölümde tutulmuş olabilirler. Dışa açık bölümde yer almış olan metodlar, dışarıdan erişip uyarılmasına izin verilmiş olan metodlardır. Özel metodlar ise sadece sınıf içinde erişimi mümkün olan metodlardır.

Nesne Temelli Programlama dillerinin metodları, nesne temelli olmayan dillerin prosedür veya fonksiyonları ile benzerdir. Sınıf kullanılarak, veri ve metodlar bir araya getirilerek birleştirilmişlerdir. Bu da sınıf içindeki mekanizmayı kullanıcısından saklamaktadır. Ohalde bir sınıfı sadece yazılımı içinde kullanmak isteyen bir programcı, sınıfın içindeki verinin hangi veri yapıları kullanılarak ve ne şekilde gerçeklendiğini, gerçekleme ayrıntılarını, ayrıca kendisinin kullanmasına İzin verilen metodların nasıl çalıştıklarını ve veriye nasıl işlediğinin ayrıntısını bilmek durumunda değildir. [4]

3.4 Nesne Kavramı

Nesne belli bir sınıfa ait olduğu bildirimi yapılan bir değişken olarak ele alınabilir. Nesne kendi sınıfında tanımı yapılan soyut veri tipinin bir örneğidir. Sınıf tanımında var olan veri yapıları ve metodlar, özel ve dışa açık olanlar, bu sınıfa ait olan nesneye de geçmektedirler. Ohalde her nesne, ait olduğu sınıfın bütün karakterini kendi üzerinde taşımaktadır.[2]

Nesne, sınıf tanımı içinde tanımlanmış olan (özel ya da dışa açık) bütün verilerinin birer kopyasına kendi içinde sahiptir. Bunlar, nesnenin belli bir andaki durumunu saklarlar ve o an için nesnenin ne durumda olduğunu belirlerler.

Nesne üzerindeki çalışmalar, nesnenin sınıf tanımında tanımlı olan metodlarından biri veya birkaçı uyarılarak gerçekleştirilir. Nesne Temelli Programlamada bir metodu uyarmak, metoda mesaj yollamak olarak da bilinir. Nasıl metodlar Nesne Temelli olmayan dillerdeki prosedür veya fonksiyonlara benzetiliyorsa, metoda yollanan mesaj da prosedür veya fonksiyonlar çağrılırken kullanılan parametreler gibi parametreler içerir. Bir metodun uyarılması veya metoda mesaj gönderilmesi genellikle nesnede saklanan veri üzerinde değişiklik oluşturur. Bu nedenle nesnenin yapısı içindeki veriler, nesnenin yaratıldığı andan itibaren geçirdiği yaşantının bir anlamda sonucudur. Burada nesnenin yaşantısından kastedilen, çeşitli parametrelerle nesnenin metodlarına uygulanan mesajlar ve neden oldukları değişikliklerdir.

Her bir sınıf değişkeni veya yaygın olarak kullanılan adıyla "nesne" ait olduğu sınıfın bir örneğini temsil eder. Eğer aynı sınıfa ait birden fazla nesne bulunması söz konusu ise bu nesneler, belli bir anda, büyük olasılıkla o ana kadar farklı yaşantıları olduğundan, farklı değerler taşıyacaklardır.

Nesne Temelli bir dilin soyut veri tipleri yardımı ile sürekli genişletilebileceği düşünülebilir. Programcılar, çeşitli sınıflar ile çeşitli soyut veri tipleri yaratabilirler ve sınıf tanımları ile soyut veri tipleri diğer programcılar tarafından da kullanılabilir hale getirebilirler. Programcının yarattığı sınıf dile eklendiğinde, dilin kendi içinde

tanımlı olan veri tipleri gibi işlenecek ve kullanılabilecektir. Bu anlamda dil, kendisine eklenen sınıflar ile sürekli olarak genişleyecektir.[2]

3.5 Sınıf Hiyerarşisi

Eğer Y sınıfının her bir nesnesi aynı zamanda X sınıfınında bir nesnesi ise Y nesne sınıfı X nesne sınıfının alt sınıfıdır (subclass) denir. X nesne sınıfı ise Y nesne sınıfının süper sınıfıdır (superclass) denir. Y nesne sınıfı X nesne sınfının tüm özelliklerini kapsar. Yani özellikler kalıtsal olarak Y sınıfına geçer. X nesnesine ihtiyaç duyulan her yerde Y nesnesi kullanılabilir. Buna substitutability özelliği denir. Kalıtsal örnek değerleri yapısal kalıtsallık (structural inheritance), kalıtsal metodlar davranış kalıtsallığı (behaviroal inheritance) olarak adlandırılmaktadır.[1]

3.6. Geleneksel Veri Tabanlarından Beklenen Fonksiyonlar

Geleneksel veri tabanı modelleri uygulamalar için uygun destek sağlarlar. Bu çeşit veri tabanları aşağıdakileri sağlamak zorundadırlar:

- Verinin sürekliliği (persistence of data)
- Verinin paylaşılması (data sharing)
- Verinin korunması (protection of data)
- Performansın geçerli seviyede olması (acceptable levels of performance)

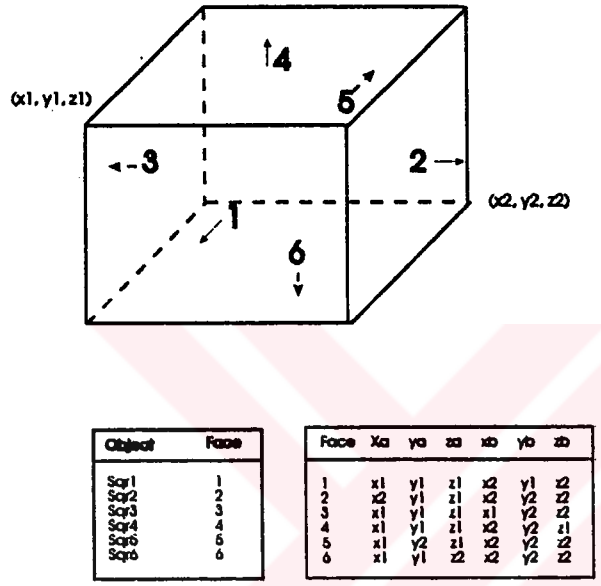
OODB (object-oriented datadase) sistemleri bütün bu özellikleri içermelidir. İlişkili DBMS (database managment systems) şu şekilde özetlenebilir:

1- Çevrim içi işlemlerin (online transaction) işlenmesi için uygunluk
2- Yapısal düzenli verinin performansı
3- Kolayca işlenebilir veri modeli
4- Endüstriyel standartlı veri tanımı ve veri işlem dilleri
5- Veri yönetim sisteminde deneyim ve bilgi

3.6.1 Geleneksel veri tabanlarının zayıf olduğu yerler

Verinin sürekli olarak değiştiği, kompleks olduğu veya çok boyutlu olduğu uygulamalarda, geleneksel veri tabanları yeterince esnek ve etkili değildirler.

Şekil 3.2'de çok boyutlu veri görülmektedir.



Object	Face
1	1
2	2
3	3
4	4
5	5
6	6

Face	Xa	ya	za	xb	yb	zb
1	x1	y1	z1	x2	y1	z2
2	x2	y1	z1	x2	y2	z2
3	x1	y1	z1	x1	y2	z2
4	x1	y1	z1	x2	y2	z1
5	x1	y2	z1	x2	y2	z2
6	x1	y1	z2	x2	y2	z2

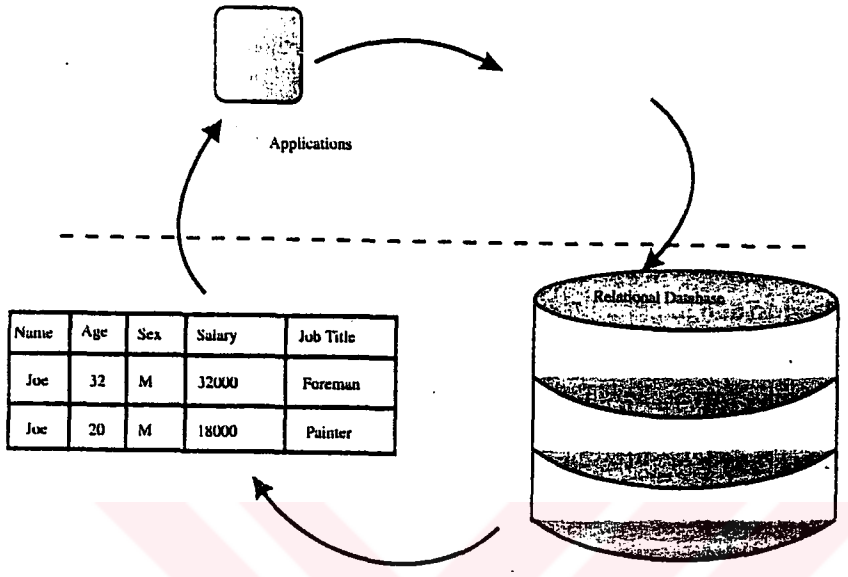
Şekil 3.2 Çok boyutlu veri

Özellikle yeni sistem tasarımı yapmada, araştırma çevrelerinde yetersiz kalmaktadır. Bu çevrelerde, tasarım işleminin yenilenebilir olması ve hem verinin yapısının hem de programın yapısının zamanla geliştirilebilir olması gerekmektedir.[4]

3.6.2 Empedans uyuşmazlığı problemi (The impedance mismatch problem)

OODBMS'lerin (object-oriented database management systems) gelişimi ve daha iyi programlama çevresi için empedans uyuşmazlığı problemi en önemli motivasyonlardan biridir. Empedans uyuşmazlığı problemi bugünkü programlama dillerindeki record-at-a-time ile SQL dilindeki set-at-a-time arasındaki seviye farkı

anlamına gelmektedir. Bu da ilişkili sistemlerde bir çok probleme sebebiyet vermektedir. Şekil 3.3'de empedans uyuşmazlığı problemi görülmektedir.



Şekil 3.3 Empedans uyuşmazlığı problemi

3.6.3 Programdaki sorgu ifadelerini gömme ile ilgili problemler

Birçok ilişkili veri tabanı sistemleri veri tabanlarına etkileşimli SQL (Structured Query Language-Yapılanmış Sorgu Dili) tabanlı arayüz sağladıkları gibi programlama arayüzü uygulamaları için de gömülmüş SQL tabanlı arayüz sağlamaktadır. Gömülmüş SQL arayüzleri veri tabanı dilleri ile çoklu programlama dilleri arasındaki söz dizimi ve anlam farklarını birleştirir. Genel amaçlı programlamada soru ifadelerini gömme ile ilgili 5 önemli problem vardır:

1- Veri tabanının veri modeli ve programlama dilinin sistem tipi uyuşmamaktadır. Örneğin, programlama dilleri nesne çeşitleri sağlamamaktadır.
2- Type sistem olmayınca, sınırlandırılmış type kontrolü bağlantı üzerinden yapılmaktadır.
3- Sorgulamalar, yalnızca sürekli veriler üzerinde formule edilebilir, kısa süreli veriler üzerinde formüle edilemezler. Uygulama adres uzayına giriş yaptıktan

4- sonra sürekli veriler üzerinde de formülasyon yapılamaz.
5- Sorgulama ve programlama dilleri ifadeleri serbestçe birleştirilemezler.
6- Her iki dilin söz dizimi anlamları birbirinden çok farklıdır ve program bu farklılıkları bilmelidir.

3.6.4 SQL DBMS problemleri

SQL DBMS farklılıklarının iç işletimini etkileyebileceği 5 alan şunlardır:[4]

1- Söz dizimi farklılıkları (Syntactial differences)
2- Anlam farklılıkları (Semantic differences)
3- Sözlük tabloları (Dictionary tables)
4- Dönüşümlü kodlar (Return codes)
5- Çoklu dillerin arayüzü (Host language interface)

3.6.5 Kompleks nesneler için veri modelleri

Veri tabanı modelleme şekilleri birçok boyutlarla karşılaştırılabilir:

- Nesne tanımlama yolları (Treatment of object identity)
- Artık geçersiz olan yapıların konuları (Issues of redundant structure)
- Tip ve sınıf inanışları (Notions of type and class)

Kompleks veriler için birçok modelleme şekilleri tanımlanmıştır.[4]

- Kompleks nesne tipleri
- Anlam modelleri
- Nesne tanımlayıcıları kullanan karmaşık nesne modelleri
- İlgili dillerin modelleri

Nesne modellemenin 2 değişik yolu vardır. Gerçek dünyadaki nesnelere benzer nesne gösterimi kullanarak doğrudan nesne erişimi sağlamak veya bu nesnelerin yalnızca tanımlarını sağlamak. Bu konunun şema gösterimi, kullanıcı haberleşimi ve işleme dilleri gibi sonuçları vardır. İlişkili modeller nesne tanımı gösterimi için anahtarlar kullanmaktadır. İlişkili modellerde nesnelerin belli özelliklerinin değeri

nesne tanımlamada ve nesne değer tespitinde kullanılmaktadır.

Kompleks nesne modelleri hiyerarşik veri tabanı yapıları oluşturmak için bir dizi elemandan oluşmaktadır. Bu şekilde içine yerleştirilmiş ilişkili model literatürdeki ilk modeldir. Anlam modelleri karmaşık nesne modellerinin dezavantajlarını çözümlemiştir. Örneğin nesne temelli fonksiyonel veri modeli (FDM-Functional data model) nesne içeren soyut sınıflardan oluşmaktadır. Mantıksal veri modeli (LDM-Logical data model) de bir çok yönden kompleks nesne modellerinin sistemini içermektedir.[4]

3.7 Nesne Temelli Veritabanı Sistemleri (OODBS)

Nesne temelli teknoloji (Object oriented technology) veritabanı yönetimi için yeni ve önemli bir alandır. Günümüzde artık ilişkili ürünler birçok yönüyle yetersiz kalmaktadır. Nesne temelli veritabanın temeli nesne temelli programlama dilindedir. Burada kullanıcılar kütükler ve kayıtlardan çok nesneler ve onların operasyonları ile uğraşmak zorundadır. Daha açık bir ifadeyle ilişkili modeldeki Dept veritabanındaki gibi ana anahtarlar gibi ifadeler yerine Dept nesnesi oluşturmak yeterli olmaktadır. Burada ilişki abstraction (genişletme) yolu ile yapılmaktadır. Nesne temelli veritabanı sistemleri (OODBS-object oriented database systems), nesne yükleme ve tekrar oluşturmada kullanılabilir ve yüklenmiş nesneler üzerinde işlem yapma olanakları sağlar. OODB hafızada var olan nesneleri yapıyı ve gösterimi değiştirmeden veri tabanına yüklemeyi sağlamaktadır. Bu da geleneksel veri tabanı modellerinin yerine neden OODB'nin tercih edildiğini daha iyi açıklamaktadır.[4]

OODB'nin yararlarından biri veri tabanı bilgisi içerisine içerdiği veri ile ilgili bilgi ve işlem yerleştirebilmesidir.

Nesneler, diğer nesnelerle kalıtsal, bağlama ve gömme yoluyla ilişkili olabilir. OODB, nesneler arasındaki kurulmuş olan ilişkiyi sürdürme özelliğine sahiptir. OODB sistemlerine giren müşteri uygulamalarını özelleştirmeye gerek yoktur. Çünkü bunlar zaten veri tabanında mevcuttur.

OODBM sistemlerin gücünü şöyle özetleyebiliriz:[4]

1- Veri ve ilişkileri karmaşık ve düzensiz olduğu zaman uygundur.
2- OOPL (nesne temelli programlama dili) arayüzleri hafifçe birleştirilmiştir.
3- PC/Workstation mimarilerini avantajlı hale getirecek şekilde tasarlanmıştır.
4- non-DBMS veri kaynakları ve uygulamalarının birleşimi sağlanmıştır.

3.6.1 OODB'lere ihtiyaç duyan uygulamaların karakteristikleri

OODB'lerin müşterileri aşağıdaki karakteristiklere sahip olmalıdır:

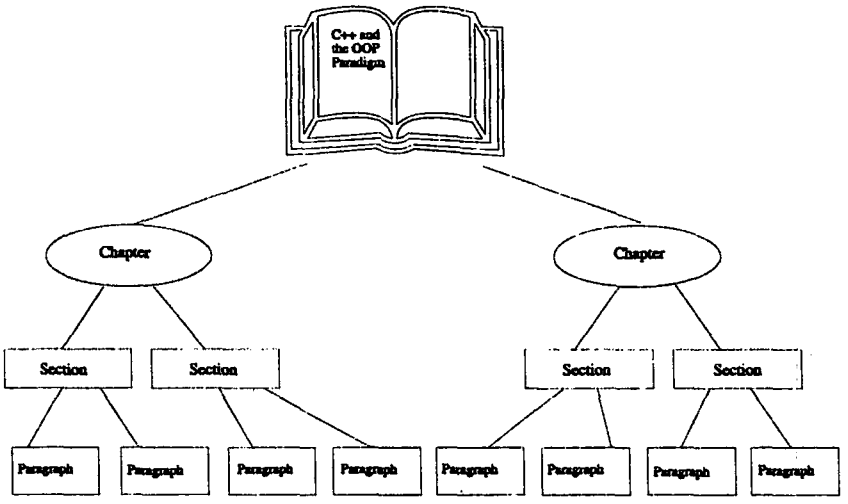
- Kompleks veri
- Verinin gelişim dizisi

Böyle sistemlere şu örnekler verilebilir:

- Bilgi tabanlı sistemler
- CAD/CAM tasarım sistemleri
- Çevre destekli program

Tipik tasarım sistemlerinin karakteristikleri aşağıda sıralanmıştır (Şekil 3.4).

- Nesneler diğer nesnelerden oluşmaktadır: modüller, altmodüller, kitaplar, bölümler gibi.
- İşlemler kalıtsaldır.
- Tasarım taslağı bir çok tasarımcı tarafından paylaşılabilir.



Şekil 3.4 Kompleks veri – bir kitap veya dökümanlar

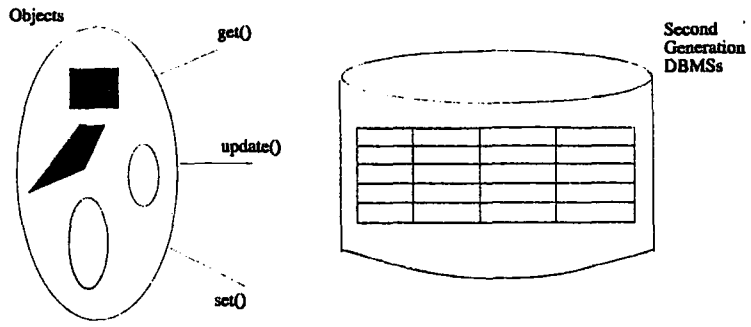
3.6.2 OOP/OODBMS çevresine göç ediş

Yeni şekle göç etme, kişilerin yaklaşımı daha iyi oldukça daha başarılı olacaktır. Bunun için OOP için gerekli olan nesneleri doğru seçmek önemlidir. Tasarımlarda OOP'in tercih edilmesinin sebepleri şunlardır:

1- Tasarımlarda tutarlılık.
2- Yazılım ve tasarımın kolay olması.
3- Koruma kolaylığı.[4]

3.7.3 OODBM sistemlerinin durumu

Son yıllarda OODB sistemleri hem akademik hem de endüstri çevrelerinden çok ilgi görmüşlerdir. Araştırmacılar henüz OODB sistemleri karakterize edecek güçlü bir veri modeli gerçekleştirememişlerdir. Gelecek için bir standartlaşmaya ihtiyaç vardır. Araştırmacılar 3'ncü Jenerasyon veri tabanı sistemi bildirisi adı altında bir yazı yayınlamışlardır. Burada şu amaçlanmıştır: 3'ncü Jenerasyon olan OODBM sistemleri daha zengin nesnelli yapılar ve kurallara destek sağlayacaktır ve bu sistem diğer sistemleri açacaktır. 3'ncü Jenerasyon DBM sistemleri 2'nci Jenerasyon DBM sistemleri (ilişkili sistemleri) kapsamalıdır (Şekil 3.5).[4]



Şekil 3.5 3'ncü Jenerasyon DBM sistemleri 2'nci Jenerasyon DBM sistemlerini kapsar

3.7.4 Mühendislik için DBM sistemlerin gerekliliği

Mühendislik uygulamaları DBM sisteme veri üzerindeki isteklerine işlemlerine ve performansına ulaşabilmek için ihtiyaç duymaktadırlar. Çoğu mühendislik uygulamaları şu özelliklere sahiptir:

- Kompleks veri: veri kompleks ve veriler arasındaki ilişki çok çeşitli olabilir. Mühendislik uygulamalarında bu tür verinin tanımlanması çok önemlidir. 3 boyutlu veri, uydu verisinin akışı bu tür karmaşık verilere örnektir.
- Kompleks işlem: karmaşık veri tanımlanması, oluşturulması ve silinmesi verinin doğrudan uygulamayla ilişkilendirilmiş olması gereğini göstermektedir.
- Dağıtılmış kontrol: mühendislik uygulamalarında çoğu zaman verilerin heterojen bir dağılımına ihtiyaç duyulmaktadır. Bu tip uygulamalar DBM sistemine ihtiyaç duymaktadır.
- Yüksek performans: çeşitli kompleks uygulamalar için buna ihtiyaç duyulmaktadır.

3.7.5 OOT ve OODBM sistemini kullanan uygulamalar

Bir çok mühendislik alanları esnek veri modellemeye, karmaşık veri ilişkilerine, büyük miktarda veriye yazılım ve donanımdan yüksek performans bekler. Genel olarak nesne temelli veri tabanları:

- Elektronik-Bilgisayar destekli tasarımda
- Mekanik-Bilgisayar destekli tasarımda
- Bilgisayar üretiminde
- Bilgisayar destekli yazılım tasarımında
- Bilgisayar destekli yayıncılıkta
- Genel olarak mühendislik tasarımlarında
- Laboratuvar biliminde ideal bir şekilde kullanılabilir.[4]

3.8 OODB'nin Özellikleri

3.8.1 OODBS

OODB'in çalışma alanı; sürekli veriye uygulanan programlama model problemlerini ve programlama modeli içerisinde görülen veri tabanı problemlerini içerir. Nesne temelli veri tabanı projeleri şu özelliklere sahiptir:

1- Nesne temelli modelleme
2- Süreklilik (persistence)
3- Nesne yorumlama
4- Çok kullanıcılar tarafından eş zamanlı erişim
5- Yerleşim saydamlığı ile dağılım
6- Sorgulama yeteneği (Query)
7- Yönetim değiştirme yeteneği
8- Veri sözlüğü
9- Sınıf kütüphanesi
10- Bütünlük ve doğruluk
11- İyileşme (Recovery)
12- Güvenlik
13- Kalıtsal veriye erişim
14- Kullanıcı arayüzü desteği

3.8.2 Nesne temelli veri modeli

OODBM sistemlerini kullanan uygulamaların gelişmesi nesne temelli veri modellerinin kullanımını gerekli kılmıştır. Nesne yönetim grubu (OMG-Object Managment Group) soyut nesne modeli, OMG uygulamalarında görülen hesaplama modelini tanımlamaktadır ve nesne tanımlamaları nesne gerçekleştirmeye ilişkin içeriklerin yerlerini bularak genelleştirilmiş bir nesne modeli sağlamaktadır. Bu model her biri nesne olarak tanımlanabilen ve bir işlem yada parametreyi tanımlayan ve müşterinin istediği bir şeydir. Metod seçimi bir çok nesneye ve müşterinin isteyine bağlıdır. OMG soyut nesne modeli şu şekilde tanımlanabilir.[4]

1- Nesnelerin ortak durumu ve bir dizi fonksiyonları vardır.
2- Nesneler müşteriye hizmet etmektedir.
3- Müşteriler istek yaparak bu hizmetten faydalanmaktadır.
4- İşlemler işlem isimleriyle tanımlanmaktadır.
5- Nesneler araçları ile tanımlanabilirler.
6- Servis için isteğin davranışı - sisteminin durumuna ve istekteki gerçek parametrelere bağlıdır.
7- Nesnelerin kimlikleri vardır.
8- Nesneler sürekli ve geçici olmak üzere iki gruba ayrılır
9- Atomicity : değişiklik işleme kütüklerinin bir özelliğidir. Bu bir işlemin İstenildiği taktirde bütün nesnelerin yer aldığı ortak durumlarını değiştirebilir ya da hiçbir değişiklik yapmaz.

Nesne veri yönetimi: [4]

Nesne modellerinin genel karakteristikleri şunlardır:

- Nesneler: işlemler, istekler, mesajlar, yöntemler, durumlar
- Çok şekillik ve bağlama (polymorphism and binding)
- Tanımlama (identity)
- Sınıflar ve tipler
- Dikkate değer nesneler: ilişkiler, özellikler, içerikler

- Genişletilebilir (extensibility)
- Bütünlük
- Nesne dili

Veri yönetim karakteristikleri şunlardır:

- Süreklilik (persistance)
- Eş zamanlılık ve değişiklikler
- Dağılım
- ODB dilleri ve sorgular
- Veri sözlüğü
- Yönetim değişimi: konfigürasyonlar, şema gelişimi
- Güvenilirlik (reliability)
- Güvenlik (security)

ODB sistem karakteristikleri:

- Sınıf kütüğü
- Program ve kullanıcı arayüzü
- Kullanıcı rolleri

3.8.3 Kompleks nesneler (complex objects)

Kompleks nesneler, aynı zamanda küme nesnelere de karşılık gelir. Bunlar, bütün sistemler tarafından sağlanan nesne yapıcılarının basit nesneler üzerine uygulanmasıyla oluşturulur. Nesne uzayını atomik nesneler üzerine kurulmuş olarak düşünebiliriz (tam sayılar, karakterler gibi).

Bileşik parçalardan bir tüm oluşturma işlemine "birleştirme" denilmektedir. Bu tasarımın karmaşıklığını basitleştirme avantajına sahiptir. Bir tümün birçok özelliği tümün parçalarına da uygulanabilir.

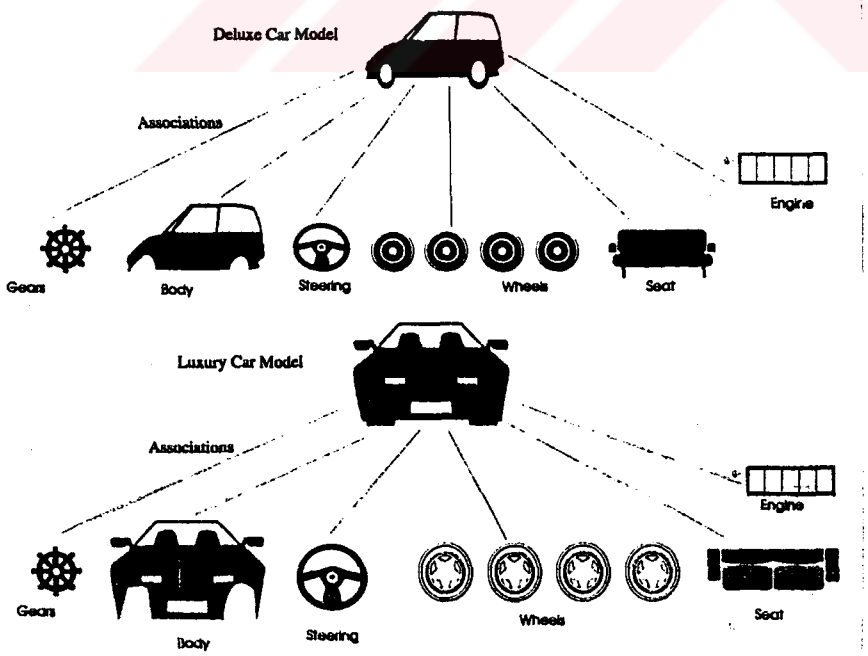
Kompleks nesnelerle ilgili önemli kavramlar aşağıda sıralanmıştır:[4]

1- Kompleks nesneler için kullanılan yükleme yapıları

2- Sorgulama işlemleri ve sonuç alma stratejileri
3- Eş zamanlı kontrol
4- Dilin işlemcileri için etkili sonuç alma stratejileri

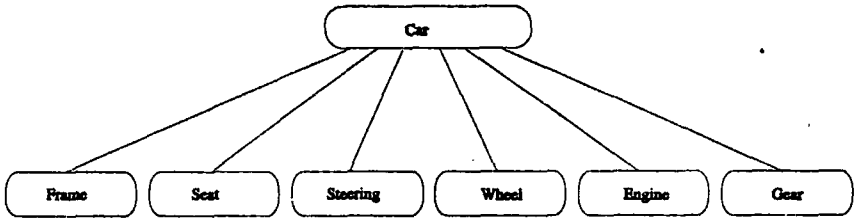
3.7.4 İlişkiler

İlişkiler, mantıksal olarak nesneleri bağlayan yapılardır. Birleştirilmiş (Composite) nesneler mantıksal olarak daha basit nesnelerin birleştirilmesi ile oluşmuş nesnelerdir. İlişkiler birçok durumlarda kullanışlıdır. Örneğin bir araba üreticisi, nesne temelli bir sistemi sade ve lüks olan iki modeli tasarlamada kullanır. Her iki modelinde ortak amaçlı bileşenleri vardır: araba çatısı, tekerlekler, motor, vites kutusu ve dişliler, koltuklar ve direksiyon gibi. Fakat her bir model için uygulanacak kalite ve fiyat farkı olacaktır. Sade olan modelde 4 silindirli motor, lüks olan modelde ise 6 silindirli motor kullanılacaktır. Buna benzer birçok farklılıklar da olacaktır. İlişkiler arabanın belli bileşenlerini belli bir araba modeli ile ilişkilendirmede kullanılabilir. Şekil 3.6'da araba parçaları için ilişkiler görülmektedir.[4]



Şekil 3.6 Araba parçaları için ilişkiler

Şekil 3.6'da görüldüğü gibi, iki araba sınıfının, nesneleri: ÇATI (FRAME), TEKERLEKLER (WHEELS), MOTOR (ENGINE), VİTES KUTUSU (GEAR), KOLTUK (SEAT), ve DİREKSİYON (STEERING) nesneleri ile ilişkilendirilmiştir. Bu ilişkileri oluşturmak için ilişki bağlantıları Şekil 3.7'deki gibi gösterilebilir.



Şekil 3.7 Sınıflar arasındaki ilişki bağlantıları

Bu bağlantıları oluşturabilmek için yedi sınıf arasındaki bağlantıların Şekil 3.7'deki gibi tanımlanması gerekir. ARABA sınıfları için aşağıdaki ortaklıklar gereklidir:

- Arabadan koltuğa
- Arabadan tekerleğe
- Arabadan vites kutusuna
- Arabadan çatıya
- Arabadan motora
- Arabadan direksiyona

3.7.5 İlişkilerin kardinalliği

Ortaklıkların kardinalliği, ortaklığın her iki yanında gösterilen nesne sayısıdır. Genelde ortaklıklar kardinaliteye göre 4 sınıfa ayrılır.[4]

- 1:1 = bire-bir
- 1m = bire-çok

- m:1= çoğa-bir
- n:m= çoğa-çok

Ortaklığın kardinalitesi sınıf tanımlandığı zaman belirtilir. Araba örneğimizde araba ile teker nesnesi arasındaki ilişki bire-çok, araba ile çatı nesnesi arasındaki ilişki ise bire-bir ortaklık ile ifade edilebilir.

3.7.6 Süreklilik (Persistance)

Süreklilik, sürekli şeklinde tasarlanan nesnelerin, kendilerini oluşturan işlemlerden sonra da varlıklarını devam ettirmeyi gerekli kılmaktadır. Süreklilik, sanki program dışında bir nesne havuzu varmış ve nesnelerin buradan istenildiğinde programa taşınabilir ya da program dışına alınabilir olduğunu varsaymaktadır. Bu şekildeki nesneler program yerleştikten sonra da varlıklarını sürdürürler.[4]

3.7.7 İşlem kütüğü (Transactions)

İşlem kütüğü (Transactions), bölünemez bir mantıksal ünite gibi davranabilir ve birlikte gruplandırılmış bir işlemler dizisi olarak düşünülebilir. Nesne temelli uygulamalar çoğunlukla karmaşık işlemlere ihtiyaç duymaktadırlar. Karmaşık işlem kütükleri saatler ve günlerce kalabilmesine rağmen ilişkili sistemlerde bu mili saniyelerdir.[1] Yani;

- Bir işlem kütüğünü tekrar baştan başlatmak kabul edilemez bir yığın iş demektir.
- İlişkili kilitlemeler kabul edilemez bir yığın bekletmeye sebep olmaktadır.

3.7.8 Eş zamanlılık kontrolü (Concurrency Control)

Eş zamanlılık kontrol mekanizmaları, ACID işlem kütüklerinin sıralı bir şekilde olması açısından önemlidir. Eş zamanlılık modları şunlardır:

- Pessimistic mod (Kötümser mod)
- Optimistic mod (İyimser mod)

- Mixed mod (Karma mod)
- Semioptimistic mod (Yarı iyimser mod)

Kötümser eş zamanlılık kontrol modu, uyuşmazlık olduğu zaman işlem kütüğünü beklemeye zorlar ve uyuşmazlık çözümlendiğinde işlem kütüğünün devamına izin vererek eş zamanlılığı önlemektedir.

İyimser eş zamanlılık kontrol modu, uyuşmazlık olmadığında işlem kütüğüne ilerleme izni verir ve bağlantı sırasında oluşmuş olan uyuşmazlıkları çözümler.

Karma eş zamanlılık kontrol modu, bir çok farklı veri kontrolü yapar. Farklı işlem kütükleri, farklı eş zamanlılık kontrol modları kullanabilirler. Karma modda sistem farklı veri veya veri tipleri için farklı eş zamanlılık kontrol modları çalıştırır.

Yarı iyimser mod karma mod'un bir diğer versiyonudur. Nesne üzerindeki kilit nesnenin kullanımı biter bitmez serbest bırakılabilir.

Bir çok eş zamanlılık kontrol algoritmaları vardır:

1- Veri parçalarını kilitleyerek, uyuşmazlık girişimini engellemek iki fazlı anahtarlama tekniklerinde daha çok kullanılır.
2- Nesne verilerinin elde edilebilir bütün versiyonlarını tutar.

3.7.9 Kilitler (Locks)

OODB sistemlerinde kilitleri kullanmanın bir çok sebebi vardır:

- Eş zamanlılık kontrolü
- Senkronizasyon

Aşağıda OODBM sistemlerinde kullanılan başlıca önemli kilit tipleri sıralanmıştır.

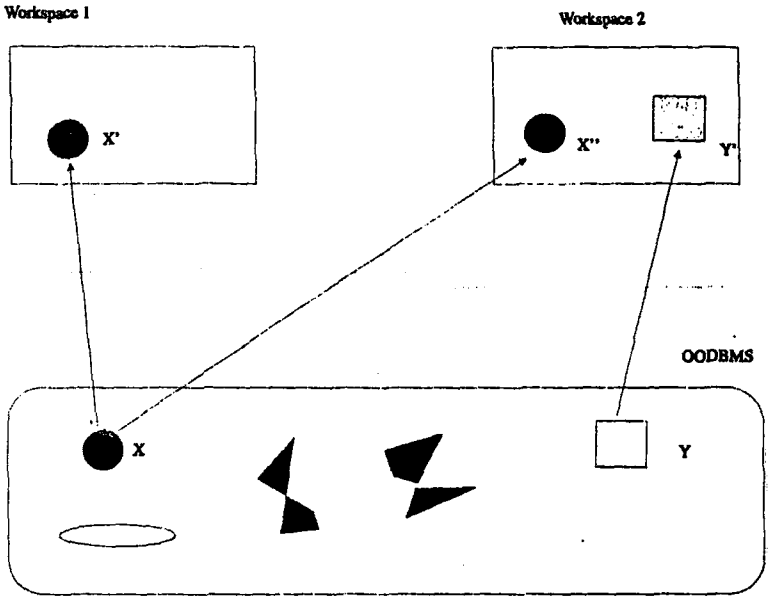
- Okuma kilitleri (Read locks): işlem kütüğü bir nesneyi okurken kullanılır. Bir çok işlem kütüğü bir okuma kilidini paylaşabilir.

• Bildirim kilitleri (Notify locks): Bu kilitler, işlem kütüğünün dışındaki cepte başarılmış olan nesneler için kullanılmaktadır.
• Boş kilitler (Null locks): Bu kilitler, nesneler anahtarlanmadığı ve nesneler işlem kütüğünün dışındaki cepte başarıldığı zamanlarda kullanılmaktadır.

3.7.10 Çeşit kontrolü (Version Control)

Bir çok uygulamalar, verilmiş olan nesnelerin bir çok versiyonlarına ihtiyaç duymaktadır. Nesne temelli sistemler bunu doğrudan desteklemektedir. Bu şekildeki destek şunları içermektedir:
•Verilmiş olan bir nesnenin yeni versiyonunu oluşturma yeteneği çıkış denetimi sağlanarak kopya çıktısını veritabanından alarak kullanıcı workstationa taşıyarak mümkün olmaktadır.
• Verilmiş olan bir nesnenin versiyonu kurma yeteneği ise nesnenin giriş denetimi sağlanarak workstationdan tekrar veritabanına taşıyarak mümkün olmaktadır.

Çeşit yönetimi, verinin gelişimini saptamak için kullanılır. Veri değiştirmeleri bazen yeni arızalara sebep olmaktadır. Ve böyle bir durumda tam olarak neyin değişmiş olduğunu bilmek çok önemlidir. Çok kullanıcılar aynı zamanda bir dizi nesne üzerinde çalıştıklarında, kullanıcılar genelde kendi çalışma alanlarını oluşturarak nesne tanımı yaparlar (Şekil 3.8). Aynı nesnelerin değişik çeşitlerini oluşturmak ve korumak nesne özelliklerinin sık değiştiği ortamlarda çok önemlidir. Bu tasarımların yenileyici işlemlerden geçtiği tasarım sistemlerinde yaygındır.



Şekil 3.8 Paylaşılmış OODBM sistemlerinden çalışma uzayına giriş denetimi

3.7.11 Konfigürasyon kontrolü

Bileşenleri daha büyük bir yapıyla birleştirmek "Konfigürasyon" adını almaktadır. Verilmiş olan bir kritere göre konfigürasyon nesne seçimini gerektirir. Konfigürasyon yönetimi, değişim yönteminin boyut oluşturması yöntemi olarak düşünülebilir. Bu yöntem başarılı nesnelerin diğer başarılı nesnelerden nasıl oluştuğunu tanımlamaktadır.

3.7.12 Özel veri tabanlarında giriş ve çıkış denetimi

Tipik mühendislik uygulamaları işbirliği içinde bir çalışma grubu tarafından yürütülmektedir. Uygulamalar, toplum paylaşımlı veri tabanı ile müşteri servis mimarisine sahiptir. Belli bir çalışma alanı için gerekli nesneler toplum veri tabanından özel veri tabanı yada çalışma uzayına çıkış denetimini işlem kütüğünün başında yapabilmektedir. Bir sıra tasarım işleminin sonunda bir çok tasarım bölümlerini birleştirmek için bu nesneler çıkış denetiminin yapıldığı toplum veri tabanına giriş denetimi yaparlar.

3.7.13 Şema gelişimi

İlişkili veritabanları yalnızca küçük değişimlere destek vermelerine rağmen nesne temelli sistemler şema gelişimini daha çok desteklemektedirler. Verilmiş olan nesne sınıfına giriş yapan uygulamaların sayısı arttığında genellikle uygulamaların gereksinimini daha iyi açıklamak için nesnelerin yapılarını tanımlamaya ihtiyaç vardır. Eğer sürekli nesneler sınıfının yapısı değişirse bu sınıfın tüm örnekleri bundan etkilenir. Sürekli sınıfların yapılarını ya da davranışlarını değiştirme işlemine şema gelişimi denir.[4]

3.7.14 Veri sözlüğü

Veri sözlüğü; kullanıcı uygulamaları yada diğer nesneler tarafından ihtiyaç duyulan bilgilerin depolanmasını sağlar. Veri sözlüğü; veri modelleri, tipler, örnekleri üzerindeki bilgileri korumak maksadıyla kullanılmaktadır.[4]

3.7.15 Yerleşim saydamlığı ile dağılım

Eğer OODB nesnelerin dağılımını destekliyorsa yerleşim saydamlığı ile dağılım önem kazanmaktadır ki; uygulamalar nesneleri yerleştirmede birleştirme mekanizmalarını kullanmaya ihtiyaç duymadan tasarlanabilsin. Eğer OODBM sistem uygulamaları dengeli değilse o zaman nesnelerin yerlerini denge sağlayıncaya kadar değiştirmek mümkün olmalıdır. Dağıtılmış sistemlerde nesneleri isimlendirmenin bir çok yolu vardır. Nesne isimlendirmede iki önemli tercih söz konusudur. Nesneler bağımlı yada bağımsız yerleştirilebilirler. Bağımsız yerleştirme isimleri "nesne tanımlayıcılar (Object Identifiers)" olarak adlandırılır. Bunlar nesne hareket ettirilse bile aynı kalma üstünlüğüne sahiptir. Fakat maalesef çok büyük oldukları için silindikleri taktirde tekrar kullanımları zordur.[4]

3.7.16 Güvenlik

Bütün ortamlarda olduğu gibi nesne temelli ortamlarda da güvenlik çok önemlidir. Güvenlik nesneleri ve işlemleri tehlikeden korumak için gereklidir.[4]

BÖLÜM 4. NESNE TEMELLİ PROGRAMLAMANIN SUNDUĞU AVANTAJLAR VE YAYGIN PROGRAMLAMA DİLLERİ

4.1 Çoklu Çalışma (Multi-tasking)

DOS tabanlı sistemlerde kullanıcı bir anda sadece bir uygulama programı ile çalışabiliyordu. Kullanıcının bir diğer uygulama programına geçebilmesi için ilk önce ilk uygulama programını kapaması ve sonra da yeni uygulama programını açması gerekiyordu. İki uygulama arasında bilgi taşınması ancak kağıt üzerinde alınan notların diğer uygulamada tekrar girilmesiyle sağlanabiliyordu. Windows programlarında ise uygulamalar zaman paylaşımlı olarak, paralel olarak çalışmaktadır. Kullanıcı diğer uygulamaya geçebilmek için sadece görev değişim komutunu çalıştırmak zorundadır. Aynı zamanda kullanıcı ön planda bir uygulama ile çalışırken, arka planda bir diğer görev işletim sistemi tarafından yerine getirilebilmektedir.[5]

4.2 Görsel Kullanıcı Standartları (Visual Standarts)

Bilgisayarların kullanım alanları arttıkça, kullanıcılar bir çok değişik uygulama ile çalışır hale geldiler. Bu da her bir uygulama için eğitim süreçlerinin gerekliliğini ortaya çıkardı. Farklı uygulamalar arasında geçişlerin kolaylaşması amacıyla görsel ortamlarda program geliştirme ile ilgili standartlar tanımlandı. Bu standartlar bir komut butonu veya araç çubuğunun büyüklüğünden, kullanılacak renklere hatta bu renklerin birbirleri arasındaki geçişlere kadar uzandı.[5]

4.3 Çoklu Ekranla Çalışabilme

Ayrı ayrı uygulamalarda veya aynı uygulama üzerinde birden fazla ekranla

çalışabilme standart windows desteğidir. Windows tarafından sağlanan Multiple Document Interface (MDI) tekniği birçok pencere ile çalışabilme ve bu pencereler arasında bir tuş veya fare hareketi ile geçiş yapabilme olanağı sağlamaktadır. Ayrıca kesme, kopyalama, yapıştırma işlemleriyle de pencereler arası veri aktarımı çok kolay bir şekilde sağlanmaktadır.[5]

4.4 Nesne Bağlama ve Gömme (Object Linking and Embedding-OLE)

Nesne gömme ve bağlama teknikleri ile uygulamalar arasında veri taşınması ve gerektiğinde bu veriler üzerinde tekrar işlem yapılabilmesi mümkün olmaktadır. Windows tarafından sağlanan bu teknikler sayesinde kullanıcı metin işleme programı içine, bir tablolama programının verisini koyabilmekte ve gerektiğinde metin üzerindeki bu tabloyu tekrar değiştirebilmektedir.[5]

4.5 Genel Sürücü Desteği

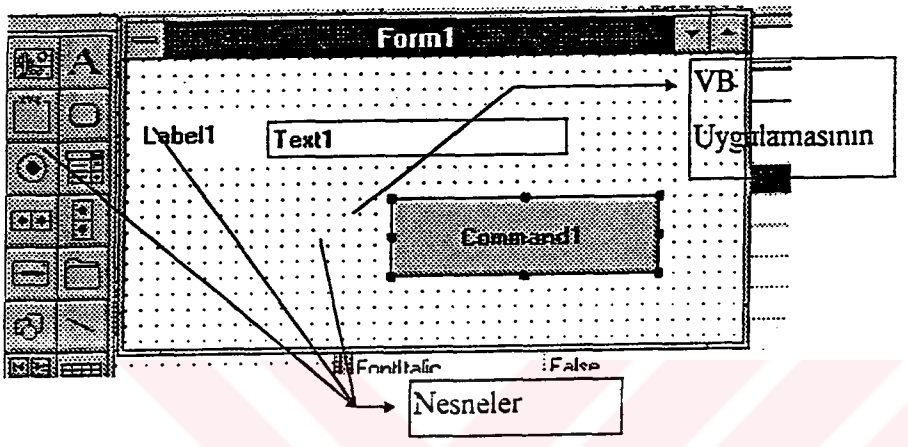
Windows kullanıcıya bakımı, kurulumu ve kullanımı kolay ve ucuz olan bir ortam sağlamaktadır. Printer, plotter, scanner ve digitizer gibi yardımcı aygıtlar ek bir çaba gerektirmeden windows altında çalışan uygulamalar tarafından kullanılabilmektedir. Bu cihazların özellikleri sürücüler tarafından windows'a aktarılmakta ve windows bunları standart bir yapıda uygulamaya geçirmektedir. Böylece her bir uygulama için bu işlemlerin tekrarlanması gerekliliği ortadan kaldırılarak, hem kullanıcılar rahatlatılmış hem de kullanıcıların daha fazla teknik bilgi gerekliliği ortadan kaldırılmıştır.[5]

Visual Basic, C++ ve Delphi görsel programlama ortamlarının en popüler olanlarıdır.

4.6 Visual Basic

Visual Basic Olay-Sürümlü (Event-Driven), Doküman-Bazlı (Document-Centered) bir geliştirme ortamıdır. Program akışı klasik prosedür bazlı programlama dillerinden farklı olarak durmadan çalışan bir döngü içerisinde değildir. Uygulama

Programı içerisinde bulunan nesneler kendileri ile ilgili bir olay gerçekleştiğinde canlanırlar ve kendilerine atanmış fonksiyonları çalıştırırlar. Dolayısıyla sistemde herhangi bir olay oluşana kadar uygulama bekleme durumunda kalır. Ayrıca Doküman-Bazlı bir yapıda olduğu için yazılacak herhangi bir uygulamanın bir form üzerinde kurulması gerekmektedir (Şekil 4.1).[5]

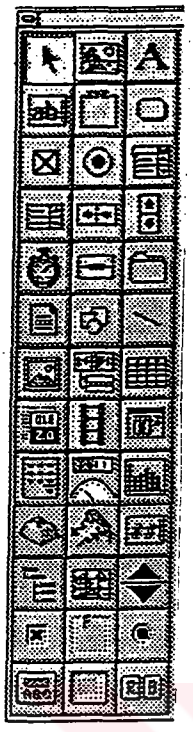


Şekil 4.1 Visual Basic formu

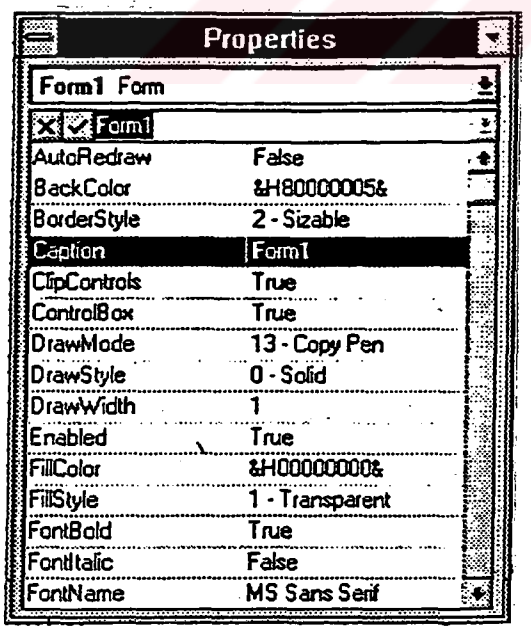
Visual Basic gerçek anlamda bir Nesne Temelli Program geliştirme ortamı değildir. Visual Basic'de varolan nesneler üzerinde oldukça esnek bir programcı hakimiyeti bulunmaktadır. Fakat Visual Basic ile yeni bir nesne tanımlanamaz ve üzerinde modifikasyon yapılamaz. Sistemde tanımlı nesneler uygulama içinde dinamik olarak yaratılabilirler. Form üzerinde kullanılacak nesneler ancak Nesne Kutusundan (Toolbox) seçilir (Şekil 4.2).

Sisteme yeni nesneler eklenebilir. Bunlar Visual Basic Custom Control (VBX) diye adlandırılan ek rutinlerdir. Eklenen yeni VBX'ler sayesinde programcı uygulamasına yeni boyutlar katabilir. Ancak nesnelerin özellikleri VBX ile sağlanan standart özelliklerdir. Bu özelliklere ekleme yapılmaz. Sistemde varolan bir VBX'in özellikleri Özellikler Penceresinden (Property Window) görülebilir.[5]

Şekil 4.3'de Özellikler Penceresi görülmektedir.



Şekil 4.2 Nesne Kutusu (Toolbox)



Şekil 4.3 Özellikler Penceresi (Property Window)

Visual Basic C++'da yazılmış Dynamic Link Library (DLL) rutinlerini kullanarak programlama ortamında bulunmayan fonksiyonaliteyi ve Windows'un Application Programming Interface (API) fonksiyonlarını da kullanabilmektedir. Ancak Visual Basic ile DLL yazılamaz ve varolan DLL'ler üzerinde geliştirme yapılamaz.

Visual Basic şu anda 16 Bit Windows platformunda çalışmaktadır. Ve yine yapılan uygulamalar 16 Bit olarak çalışmaktadır.

Visual Basic uygulamaları tam anlamıyla Executable (EXE) tipli program olmamaktadır. Hazırlanan programlar her zaman bir run-time modüle ihtiyaç duymaktadır. Bu yorumlayıcı desteğinden dolayı da performans olarak hiçbir zaman gerçek derlenmiş bir program düzeyine erişememektedir.

Visual Basic çabuk ve kolay geliştirme sağladığından özellikle veri tabanı uygulama geliştiricilerin tercih ettiği bir platform olmaktadır. Ancak donanımın daha fazla niteliğinin kullanılması gereken uygulamalarda Visual Basic yeterli olmamaktadır.[5]

4.7 Visual C++

C++ Windows platformunda programcıya sonsuz esneklik sağlayarak gerçek bir görsel program geliştirme ortamı sağlamaktadır. C++ ile hem 16 Bit hem de 32 Bit platformlarda çalışabilen programlar yazılabilmektedir. Ayrıca hazırlanacak programlar aşağıdaki program tiplerinde olabilmektedir.

- Windows EXE
- DOS EXE
- Windows DLL
- Windows VBX
- Win 32 Static Library

C++ tam anlamıyla bir Nesne Temelli Programlama (OOP) dilidir. Bir OOP dilinin

taşıması gereken tüm özellikleri taşır. Windows işletim sisteminin gerekli tüm özelliklerini kullanabilir.

C++ ile yazılan programların en önemli noktası veri yapılarının bulunduğu sınıfların doğru kurulması ve sınıflar arası ilişkilerin doğru tanımlanması olmaktadır. Bu yapı kurulduktan sonra sınıfların üyelik değişkenlerinin ve fonksiyonlarının yazılması ile program oluşturulacaktır. Her uygulama için bu sınıfların yeniden kurulması bir yük getireceği için hazır sınıf kütüphanelerinin bulunması, uygulama geliştirmek için büyük bir avantaj sağlayacaktır. Microsoft bu amaçla genel amaçlı bir Class Library hazırlamıştır.[5]

4.8 Delphi

Delphi, Borland'ın derleyici teknolojisindeki en iyi yanlarını alan, bunları Borland'ın veri tabanı teknolojisindeki en iyi yanlarıyla birleştiren ve yeni görsel programlama araçlarını kullanan özgün bir üründür. Bu özellikleriyle karma bir ürün olan Delphi, programcılara hem standart uygulamaları hem de istemci/sunucu uygulamalarını hızlı hazırlama olanağını eşi görülmemiş bir mükemmellikle sunmaktadır.[6]

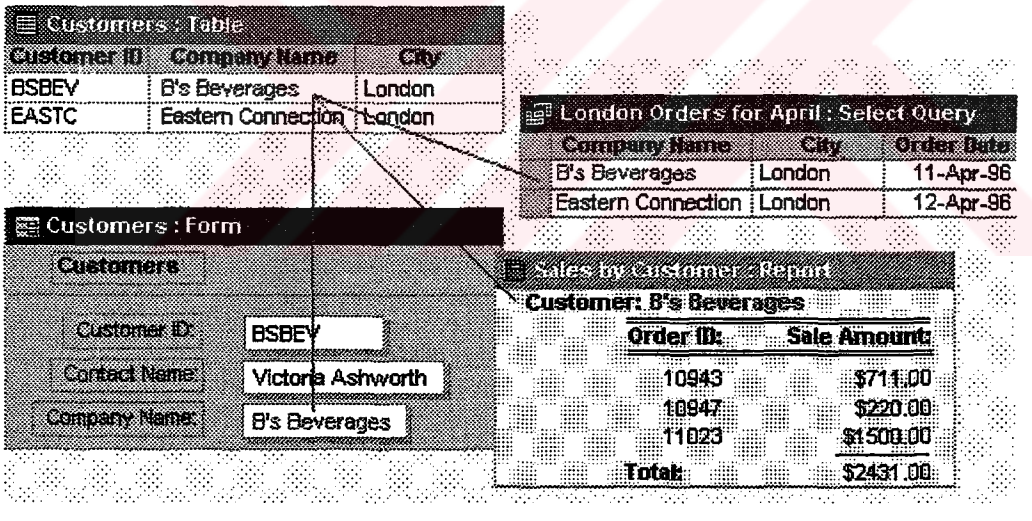
Delphi'nin varlığı üç nedene dayanmaktadır:

1- Dünyanın güçlü ve uzmanlaşmış bir istemci/sunucu aracına ihtiyacı olması.
2- Programcıların karmaşık projeleri kısa zaman dilimi içinde bitirebilmek için görsel araçları kullanmaya gerek duymaları.
3- Programcıların sağlam ve uzmanlaşmış programları yazabilmeleri için yeniden kullanılabilir, nesneye dayalı program parçalarına ihtiyaç duymaları.

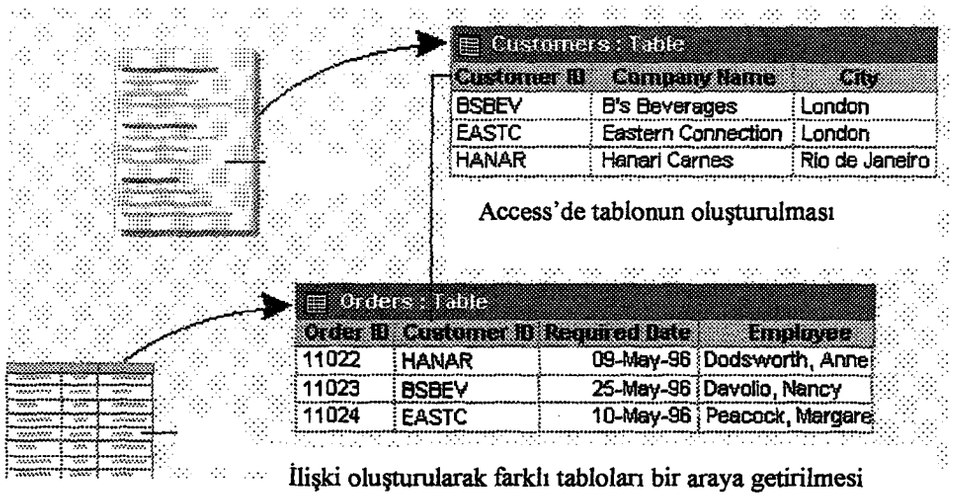
Delphi, büyük ve karmaşık problemleri bile çözebilen iyi bir istemci/sunucu aracıdır. Çok yetenekli, nesneye dayalı bir dilin gücünden yararlanan gerçek bir derleyicidir. Ayrıca Delphi veri tabanı işlemlerini kısa zamanda yerine getirebilmek için Borland Veri Tabanı Motorunu kullanır. Bu da Delphi'nin istemci/sunucu kısmının oluşturulduğu veri tabanı araçlarının sağlam ve uzmanlaşmış bir teknolojinin parçaları olduğu anlamına gelir.[6]

4.9 Access

Access bir nesne temelli veritabanı programıdır. Burada veri tabanı oluşturulurken program satırları otomatik olarak hazırlanmaktadır. Bu tezde geliştirilen uygulama access'de geçeklenmiştir Microsoft access kullanarak bütün bilgiler bir tek veritabanı kütüğünden elde edilebilir (Şekil 4.4) . Kütük veriyi farklı kutular içine yüklemektedir. Bu kutulara tablo denilmektedir. Veri bir kere yüklendikten sonra farklı yerleşim yerlerinde kullanılabilir. Access'de yeni bir veritabanı oluşturmak için önce yeni komutu seçilir sonrada table sekmesi tıklanarak veri tabanı tasarımı yapılır. Veri tabanında bulunulması istenen sınıflar, nesneler ve diğer özellikler belirtilir. Farklı tablolardaki veriyi bir araya getirmek için tablolar arasında ilişki kurulmaktadır (Şekil 4.5). Belli özellikteki verileri bir araya getirmek için sorgular (query) oluşturulmaktadır. Veriyi farklı biçimlerde sunmak için ise form nesnesinden faydalanılır. Burada istenilen nesne temelli ortam oluşturulur.



Şekil 4.4 Verinin birçok yerde kullanılabilmesi



Şekil 4.5 Tablolar arasında ilişki kurulması

BÖLÜM 5. NESNE TEMELLİ VERİTABANI İLE BİR UYGULAMA TASARIMI

5.1 Uygulamanın Genel Tanıtımı

Bu tezde nesne temelli veritabanının bir uygulaması olarak, oyuncak mağazasındaki oyuncakların, personelin ve oyuncakların alındığı fabrikaların veritabanları Access'de tasarlanmıştır. Bu uygulamada verilerin bütün özellikleri veritabanında belirtilerek ve aralarında gerekli ilişkiler kurulacaktır. Daha sonra karmaşık sorgular hazırlanacaktır.

5.2 Uygulamanın Tasarımı

Oyuncak mağazasındaki oyuncakların, mağazada çalışan personelin ve oyuncakların alındığı fabrikaların geniş bilgilerini içeren veritabanını oluşturmak amacıyla yapılacak tasarımda öncellikle tek tek veri sınıfları ve bu sınıfların özellikleri belirtilecektir. Daha sonra sınıflar arası ile ilişkiler belirlenip veritabanı tasarımı tamamlanacaktır.

5.2.1 Oyuncaklar sınıfı tasarımı

Oyuncaklar sınıfında oyuncakların belirtilmesi istenen yada ihtiyaç duyulan bütün özellikleri belirtilir.
Oyuncak sınıfının nesneleri:

- Oy kodu
- Oy genel adı
- Oy türü
- Ambalaj adı

- Cins
- Yaş grubu
- Oy malzemesi
- Fab kodu
- Oy fiyatından oluşmaktadır.

Oy kodu nesnesi: Bu nesne oyuncağın kodunu içermektedir. Tam sayı değerli bilgi içerir. 3 karakterden oluşturulmuştur.

Oy genel adı nesnesi : Bu nesne oyuncağın genel adını içermektedir.Araba, bebek ve uçak gibi.Bu nesnenin içerdiği değerin uzunluğu 15 karakterden oluşturulmuştur.

Oy türü nesnesi: Oyuncağın türünü belirtmektedir. Oyuncağın pilli olup olmadığı gibi bilgileri içerir. Metin değerli bir bilgidir. 15 karakterden oluşturulmuştur.

Ambalaj adı nesnesi: Oyuncağın ambalajdaki ismini vermektedir. Metin değerli bilgi içermektedir. Karakter uzunluğu 15 'dir.

Cins nesnesi: Oyuncağın kız, erkek veya her ikisi için olup olmadığı bilgisini içermektedir. 10 karakterden oluşmuş metin değerli bilgi içermektedir.

Yaş grubu nesnesi: Oyuncağın hangi yaş grubuna ait olduğunu belirtmektedir. Metin değerli bilgi olup 10 karakterden oluşturulmuştur.

Oy malzemesi nesnesi: Oyuncağın yapıldığı malzemeyi belirtmektedir. Metin değerli bilgi olup 15 karakterden oluşturulmuştur.

Fab kodu nesnesi: Oyuncağın alındığı fabrikanın kodunu vermektedir. Sayı değerli bilgi olup 3 karakterden oluşturulmuştur.

Oy fiyatı nesnesi: Oyuncağın satış fiyatını içermektedir.

Şekil 5.1'de oyuncak sınıfının veritabanı yapısı gösterilmiştir.

Alan Adı	Veri Türü	Tanım
OY KODU	Sayı	
OY GENEL ADI	Metin	OYUNCAK ADI
OY TÜRÜ	Metin	OYUNCAK TÜRÜ
AMBALAJ ADI	Metin	OYUNCAĞIN AMBALAJI ÜZERİNDEKİ İSİM
CİNS	Metin	KIZ-ERKEK
YAŞ GRUBU	Metin	OYUNCAĞIN İLGİLİ YAŞ GRUBU
OY MALZEMESİ	Metin	OYUNCAĞIN YAPILDIĞI MALZEME
OY FİYATI	Metin	OYUNCAĞIN FİYATI
FAB KODU	Sayı	OYUNCAĞIN ALINDIĞI FABRİKANIN KODU

Şekil 5.1 Oyuncak sınıfının veritabanı yapısı

Oyuncak sınıfı şekil 5.2'den de görüldüğü gibi 10 çeşit farklı özellikte oyuncaklardan oluşturulmuştur.

5.2.2 Personel sınıfı tasarımı

Personel sınıfında mağazada çalışan personel ile ilgili bilgiler bulunmaktadır. Personel sınıfının nesneleri :

- Pers adı
- Soyadı
- Cins
- Görevi
- Pers adresi
- Tel 'den oluşmaktadır.

Pers adı nesnesi: Personelin adını içermektedir. Metin değerli bilgi olup 15 karekterden oluşturulmuştur.

Soyadı nesnesi: Personelin soyadı bilgisini içermektedir. Metin değerli bilgi olup 15 karakterden oluşturulmuştur.

Cins nesnesi: Personelin cinsiyeti bilgisini içermektedir. Metin değerli olup 10 karakterden oluşturulmuştur.

Kimlik	OY	OY GENEL	OY TÜRÜ	AMBALAJ ADI	CİNS	YA	OY	OY	FAB
1	201	BEBEK	BATARYALI	BARBY BEBEK	KIZ	5-7	PLASTİK	8000000	101
2	202	ARABA	BATARYALI	ROY ROYS	ERKE	6-9	ALÜMİNYUM	5000000	602
3	203	BEBEK	KURMALI	CINDY	KIZ	3-6	PAMUK	4000000	220
4	204	TREN	BARAYALI	AUTO TREN	ERKE	6-9	PLASTİK	15000000	305
5	205	UÇAK	BATARYALI	FLIGHT	ERKE	7-9	PLASTİK	15000000	104
6	206	TREN	KURMALI	TRAIN	ERKE	4-6	PLASTİK	3000000	101
7	207	ARABA	PİLLİ	CAR	ERKE	6-9	PLASTİK	7000000	101
8	208	YAP BOZ	100 PARÇA	PUZZLE 100	ORTA	6-9	KARTON	2000000	602
9	209	YAP BOZ	1000 PARÇA	PUZZLE 1000	ORTA	10-	KARTON	3000000	602
10	210	BEBEK	YÜRÜYEN	WALK GIRL	KIZ	4-8	PLASTİK	20000000	305

Şekil 5.2 Oyuncak sınıfı ve kayıtları

Görevi nesnesi: Personelin görevi bilgisini içermektedir. Metin değerli bilgi olup 15 karakterden oluşturulmuştur.

Pers adresi nesnesi: Personel adresi bilgisi içermektedir. Metin değerli bilgi içermektedir. 25 Karakterden oluşturulmuştur.

Tel nesnesi: Personelin telefon numaraları bilgisini içermektedir. Sayısal bilgi olup 7 karakterden oluşturulmuştur.

Şekil 5.3'de personel sınıfının veritabanı yapısı gösterilmiştir.

Alan Adı	Veri Türü	Tanım
KODU	Sayı	ÇALIŞAN PERSONELİN KODU
PERS ADI	Metin	ÇALIŞAN PERSONELİN ADI
SOYADI	Metin	PERSONELİN SOYADI
CİNS	Metin	PERSONELİN CİNSİYETİ
GÖREVİ	Metin	PERSONELİN GÖREVİ
PERS ADRESİ	Metin	PERSONELİN ADRESİ
TEL	Sayı	PERSONELİN TELEFONU

Şekil 5.3 Personel sınıfının veritabanı yapısı

Personel sınıfı şekil 5.4'de görüldüğü gibi 8 adet personel kaydından oluşmaktadır.

PERS ADI	SOYADI	CİNS	GÖREVİ	PERS ADRESİ	TEL
LEYLA	KURU	KIZ	MAĞAZA SAHİBİ	KÜLTÜR M. AKÇAM S. 10/5	4547854
ERSEN	KURU	ERKEK	MAĞAZA SAHİBİ	KÜLTÜR M. AKÇAM S. 10/5	7841255
FATMA	KOŞAR	KIZ	SATIŞ ELEMANI	AYDINPINAR M. 15/7	5578785
HATİCE	AKSOY	KIZ	SATIŞ ELEMANI	CUMHURİYET BULV. 4/10	4541555
AHMET	PİROĞLU	ERKEK	SATIŞ ELEMANI	ATATÜRK BULV.8/4	3321414
HASAN	TARTAR	ERKEK	MUHASEBECİ	ÇAMLIEVLER 16/4	6968474
AYNUR	BİLLUR	KIZ	HİZMETLİ	HUZUR C. 8/14	8963333
AYNUR	KISA	KIZ	HİZMETLİ	YEŞİL M. 12/4	6694754

Şekil 5.4 Personel sınıfı ve kayıtları

5.2.3 Fabrika sınıfı tasarımı

Fabrika sınıfında oyuncakların üretildiği fabrikalar ile ilgili bilgiler bulunmaktadır. Fabrika sınıfının nesneleri :

- Fab kodu
- Fab adı
- Fab adresi
- Fab ili
- Fab tel'den oluşmaktadır.

Fab kodu nesnesi: Oyuncakların alındığı fabrikanın kodunu göstermektedir. Sayısal bilgi olup 3 karakterden oluşturulmuştur.

Fab adı nesnesi: Oyuncakların alındığı fabrikanın ismini içermektedir. Metin değerli bilgi olup 15 karakterden oluşturulmuştur.

Fab adresi: Fabrikaların adres bilgilerini içermektedir. Metin değerli bilgi olup 20 karakterden oluşturulmuştur.

Fab ili nesnesi: Fabrikaların bulunduğu il ile ilgili bilgileri içermektedir. Metin değerli bilgi olup 10 karakterden oluşturulmuştur.

Fab tel nesnesi: Fabrikaların telefon bilgilerini içermektedir. Sayı değerli bilgi olup 7 karakterden oluşturulmuştur.

Şekil 5.5'de fabrika sınıfına ait veritabanı yapısı gösterilmektedir.

Alan Adı	Veri Türü	Tanım
FAB KODU	Sayı	OYUNCAĞIN ALINDIĞI FABRİKANIN KODU
FAB ADI	Metin	FABRİKANIN ADI
FAB ADR	Metin	FABRİKANIN ADRESİ
FAB İL	Metin	FABRİKANIN İLİ
FAB TEL	Sayı	FABRİKANIN TELEFONU

Şekil 5.5 Fabrika sınıfının veritabanı yapısı

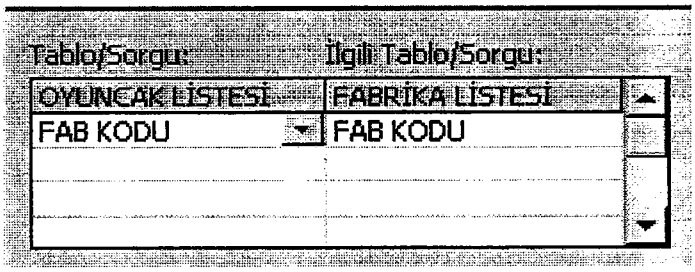
Fabrika sınıfı şekil 5.6'dan da görüldüğü gibi 5 adet fabrika kaydından oluşmaktadır.

FAB KODU	FAB ADI	FAB ADR	FAB İL	FAB TEL
602	ASAL AŞ.	YÜZÜNCÜYIL C	İZMİT	3125412
220	TEKİN HOL	MALAZGİRT C.	KAYSERİ	9666511
101	KURU AŞ	YILDIRIM C.3/7	BOLU	5471369
104	MELAN LTD.ŞT	BİLLUR C. 20/8	ADAPAZARI	6987410
305	YANMAZ AŞ	KÜLTÜR C.	DÜZCE	5241221

Şekil 5.6 Fabrika sınıfı ve kayıtları

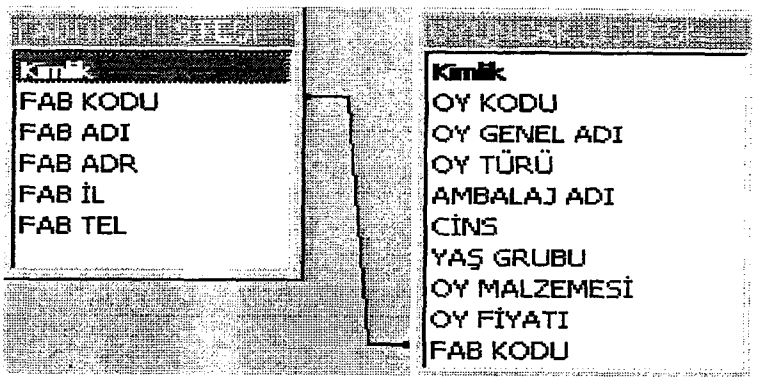
5.3 Sınıflar Arası İlişkiler

Sınıflar arasındaki ilişki tasarım anında belirtilmelidir. Bu uygulamada oyuncak sınıfı fabrika sınıfı ile ilişkilendirilmiştir. Oyuncak sınıfı ile fabrika sınıfı "FAB KODU" adı ile ortak bir nesneye sahiptirler. Bu ortak nesne tasarım anında oluşturulmuştur. İstenirse ortak nesne sayısı arttırılabilir. Bu tasarımcının isteğine bırakılmıştır. Nesne temelli veritabanları tasarım anında kurulmuş olan bu ilişkiyi bilgisayara tanımlamada büyük kolaylıklar getirmiştir. Oyuncak sınıfı ile fabrika sınıfı arasındaki ilişkiyi bilgisayara tanımlarken Access ortamında yapılması gereken tek şey bu iki sınıfın hangi nesneyle ilişkilendirildiğini söylemektir. Şekil 5.7'de oyuncak sınıfı ile fabrika sınıfı arasındaki ortaklığın Access ortamında tanıtılması görülmektedir.



Şekil 5.7 Fabrika sınıfı ile oyuncak sınıfı arasında ilişkinin oluşturulması

Şekil 5.8'de de oluşturulmuş olan ilişki görülmektedir.



Şekil 5.8 Oyuncak sınıfı ile fabrika sınıfı arasındaki ilişki

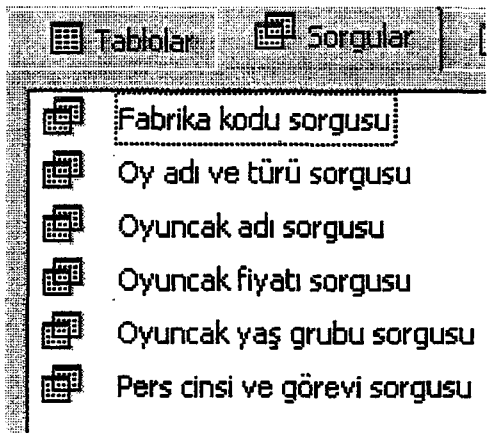
BÖLÜM 6. SORGULARIN TASARIMI VE GERÇEKLENMESİ

6.1 Giriş

Bu bölümde istenen sorgular hazırlanacak ve sorguların gerçeklenmesi sağlanacaktır. Sorgular hem tablo şeklinde hem de nesne temelli ortamda bir form olarak gerçeklenmiştir.

6.2 Sorguların Hazırlanması

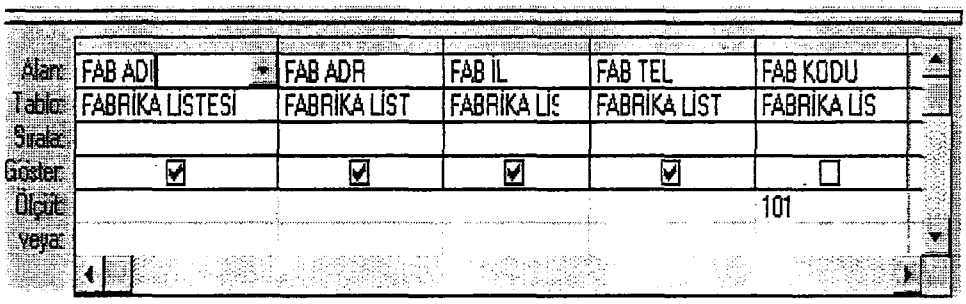
Sorgular hazırlanırken her bir sorgu için sorgular nesnesi seçili durumda iken yeni sonrada tasarla sekmesi seçilir. Bu yapıldıktan sonra tasarlanacak olan sorguda bulunması istenen bilgiler seçili duruma getirilir. Tasarımda sorgulanacak olan bilgiler belirtilerek de tasarım hazırlanması tamamlanmış olur. Oyuncaklar veritabanı uygulamasında şu sorgular hazırlanmıştır:



Şekil 6.1 Hazırlanan sorgular

6.3 Fabrika Kodu Sorgusu Tasarımı

"Fabrika kodu sorgusu" adlı sorguda Fab kodu 101 olan fabrikanın adı, adresi, bulunduğu il ve fabrikanın telefon numarası adlı bilgiler istenmektedir. Şekil 6.1'de bu sorgunun tasarımı gösterilmiştir. Bunun için FAB KODU'nda ölçüt olarak belirtilen yere 101 yazılır ve istenen diğer nesneler seçili duruma getirilir.



Şekil 6.2 Fabrika kodu sorgusunun tasarımı

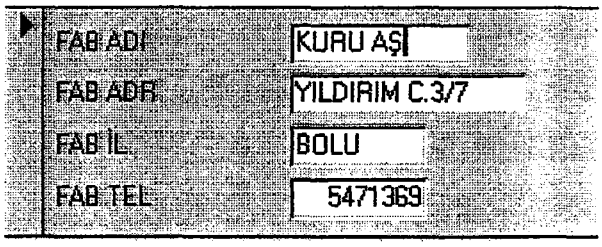
6.3.1 Fabrika kodu sorgusunun gerçeklenmesi

Şekil 6.3'de fabrika kodu sorgusunun tablo olarak gerçeklenmesi görülmektedir.

FAB ADI	FAB ADR	FAB İL	FAB TEL
KURU AŞ	YILDIRIM C.3/7	BOLU	5471369

Şekil 6.3 Fabrika kodu sorgusunun tablo olarak gerçeklenmesi

Burdan da görüldüğü gibi kayıtlarda bulunan ve kodu 101 olan fabrikanın adı, adresi ve telefon bilgileri tablo olarak gerçeklenmiştir. Şekil 6.4'de gerçekleme olayı nesne temelli ortamda istenilen form seçilerek gerçeklenmiştir.



Şekil 6.4 Nesne temelli olarak fabrika kodu sorgusunun gerçeklenmesi

6.4 Oyuncak Genel Adı Sorgusu Tasarımı

"Oyuncak genel adı sorgusu" adlı sorguda oyuncak genel adı "ARABA" olan oyuncakların ambalaj adı, oyuncağın hangi cinsiyetteki çocuk için olduğu, oyuncağın hangi yaş grubundan çocuğa ait olduğu, oyuncağın alındığı fabrikanın adı ve oyuncağın fiyatı bilgileri istenmektedir. Şekil 6.5'de bunun tasarımı hazırlanmıştır. OY GENEL ADININ ölçüt bölmesine "ARABA" yazılarak diğer nesneler seçili duruma getirilmiştir.

Alan:	FAB ADI	OY GENEL AD	CİNS	YAŞ GRUBU	AMBALAJ ADI	OY FİYATI
Tablo:	FABRİKA LİS	OYUNCAK LİS	OYUNCAK LİS	OYUNCAK LİS	OYUNCAK LİS	OYUNCAK L
Sırala:						
Göster:	☑	☑	☑	☑	☑	☑
Ölçüt:		"ARABA"				
veya:						

Şekil 6.5 Oyuncak genel adı sorgusunun tasarımı

Bu sorguda hem oyuncak sınıfı hem de fabrika sınıfı ile ilgili bilgiler bulunmaktadır. Böyle bir sorgunun tasarlanabilmesi için daha önce bu iki sınıfın ilişkilendirilmiş olması gerekmektedir.

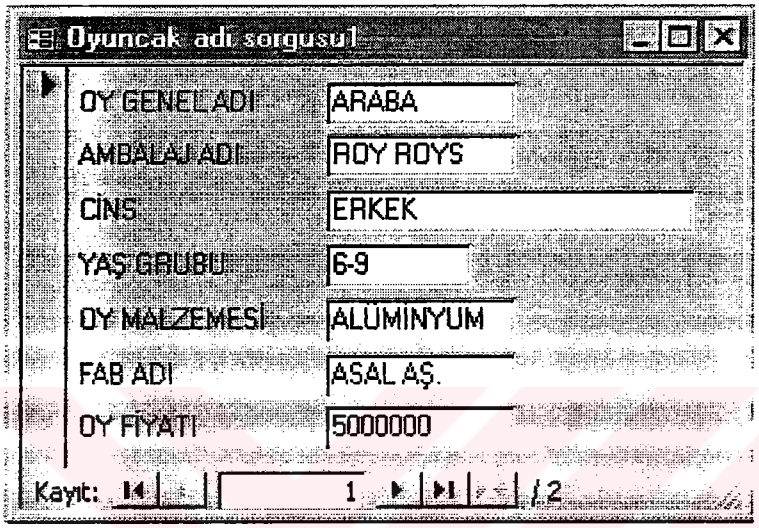
6.4.1 Oyuncak genel adı sorgusunun gerçeklenmesi

Şekil 6.6'da oyuncak genel adı sorgusunun tablo olarak gerçeklenmesi görülmektedir.

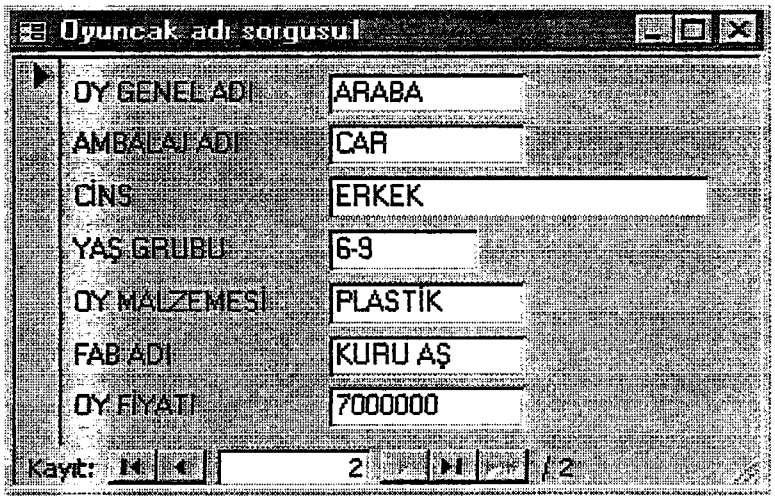
OY GENEL ADI	AMBALAJ ADI	CİNS	YAŞ GR	OY MALZEME	FAB ADI	OY FİYATI
ARABA	ROY ROYS	ERKEK	6-9	ALÜMİNYUM	ASAL AŞ.	5000000
ARABA	CAR	ERKEK	6-9	PLASTİK	KURU AŞ	7000000

Şekil 6.6 Oyuncak genel adı sorgusunun tablo olarak gerçeklenmesi

Buradan da görüldüğü gibi kayıtlarda bulunan ve genel adı "araba" olan oyuncağın ambalaj adı, cins, yaş grubu, fab adı ve oyuncak fiyatı bilgileri tablo olarak gerçeklenmiştir. Şekil 6.7'de gerçekleme olayı istenilen form seçilerek nesne temelli ortamda gerçeklenmiştir. Bu sorguyu sağlayan 2 adet kayıt vardır. Şekil 6.8'de 2'nci kayıt gösterilmiştir.



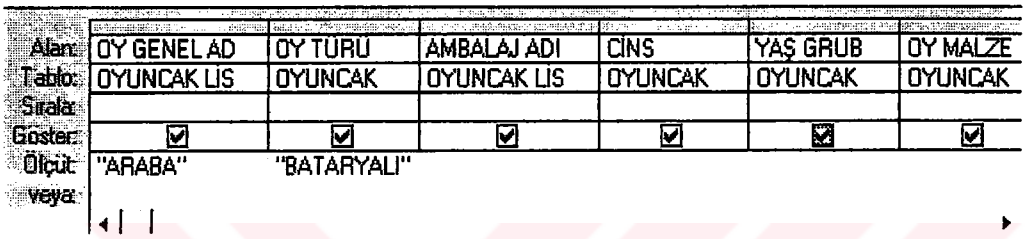
Şekil 6.7 Oyuncak adı sorgusunun 1'inci kaydının nesne temelli olarak gerçeklenmesi



Şekil 6.8 Oyuncak adı sorgusunun 2'nci kaydının nesne temelli olarak gerçeklenmesi

6.5 Oyuncak Adı ve Türü Sorgusu Tasarımı

Bu sorguda sorguda oyuncak genel adı "ARABA" ve oyuncak türü "BATARYALI" olan oyuncağın Ambalaj adı, cins, yaş grubu, oy malzemesi, fab adı ve oyuncak fiyatı bilgileri istenmektedir. Burada fabrika sınıfı ile oyuncak sınıfına ait bilgiler bulunmaktadır. Şekil 6.9'da bu tasarım görülmektedir. Bu sorgu bir önceki sorgudan daha karmaşıktır. Nesne temelli veritabanı sistemlerinde karmaşık sorgu oluşturma kolaylığı vardır.



Alan:	OY GENEL AD	OY TÜRÜ	AMBALAJ ADI	CİNS	YAŞ GRUB	OY MALZE
Tablo:	OYUNCAK LİS	OYUNCAK	OYUNCAK LİS	OYUNCAK	OYUNCAK	OYUNCAK
Sırala:						
Göster:	☑	☑	☑	☑	☑	☑
Ölçüt:	"ARABA"	"BATARYALI"				
veya:						

Şekil 6.9 Oyuncak adı ve türü sorgusu tasarımı

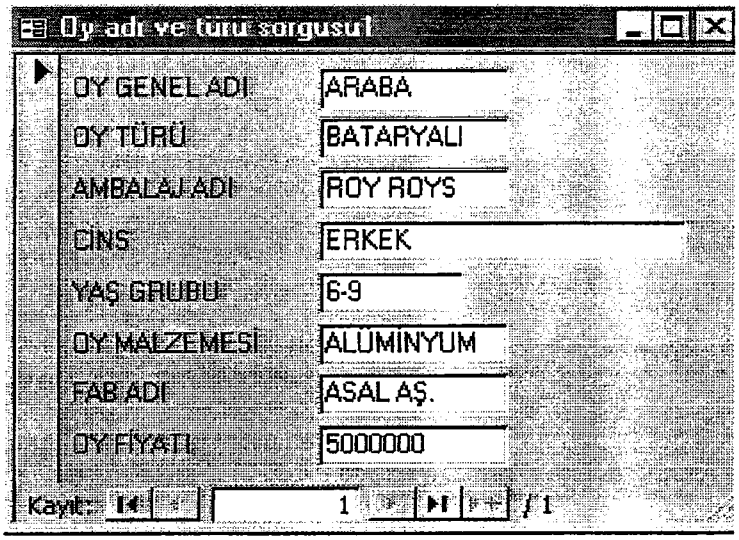
6.5.1 Oyuncak adı ve türü sorgusunun gerçeklenmesi

Şekil 6.10'da oyuncak adı ve türü sorgusu tablo olarak gerçekleştirilmiştir.

OY GENEL	OY TÜRÜ	AMBALAJ A	CİNS	YAŞ	OY MALZEME	FAB ADI	OY FİYATI
ARABA	BATARYALI	ROY ROYS	ERKEK	6-9	ALÜMİNYUM	ASAL AŞ.	5000000

Şekil 6.10 Oyuncak adı ve türü sorgusunun tablo olarak gerçeklenmesi

Buradan da görüldüğü gibi oyuncak genel adı "ARABA" ve oyuncak türü "BATARYALI" olan bir adet oyuncak bulunmuştur. Şekil 6.11'da istenilen form seçilerek gerçekleme nesne temelli olarak yapılmıştır.



Şekil 6.11 Oyuncak adı ve türü sorgusunun nesne temelli olarak gerçeklenmesi

6.6 Oyuncak fiyatı sorgusu tasarımı

Bu sorguda oyuncak genel adı "BEBEK" ve oyuncak fiyatı 5000000 'dan küçük oyuncakların ambalaj adı, cins, yaş grubu bilgileri istenmektedir. Bu sorguda yalnız oyuncak sınıfına ait bilgiler sorgulandığı için başka bir sınıf ile ilişki söz konusu değildir. Şekil 6.12'de oyuncak fiyatı sorgusunun tasarımı görülmektedir. Burada OY GENEL ADININ ölçüt bölmesine "BEBEK" ve OY FİYATI ölçüt bölmesine de <=5000000 yazılarak diğer nesneler seçili duruma getirilmiştir.

Alan:	OY GENEL ADI	OY TÜRÜ	AMBALAJ ADI	CİNS	YAŞ GRUBU	OY FİYATI
Tablo:	OYUNCAK LİST	OYUNCAK LİS	OYUNCAK LİS	OYUNCAK LİS	OYUNCAK LİS	OYUNCAK
Sırala:						
Göster:	☑	☑	☑	☑	☑	
Ölçüt:	"bebek"					<="500000
veya:						

Şekil 6.12 Oyuncak fiyatı sorgusu tasarımı

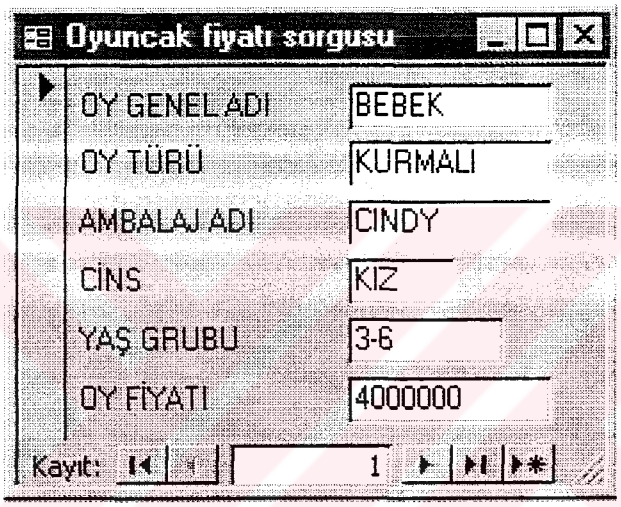
6.6.1 Oyuncak fiyatı sorgusunun gerçeklenmesi

Şekil 6.13'de oyuncak fiyatı sorgusu tablo olarak gerçeklenmiştir.

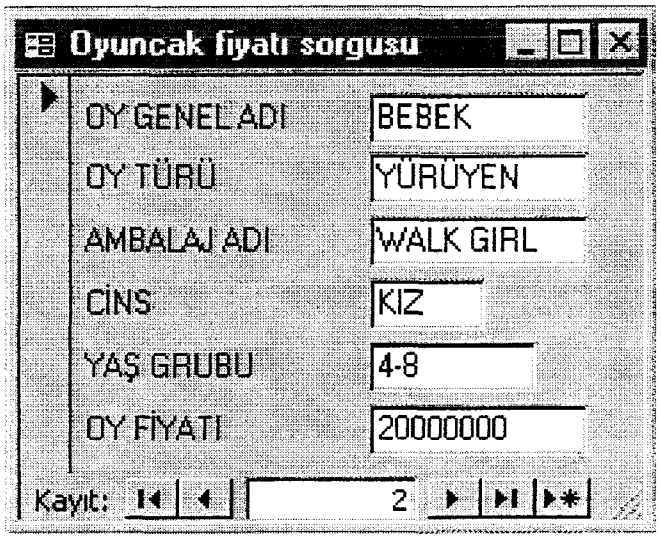
OY GENEL ADI	OY TÜRÜ	AMBALAJ ADI	CİNS	YAŞ GRUBU	OY FİYATI
BEBEK	KURMALI	CINDY	KIZ	3-6	4000000
BEBEK	YÜRÜYEN	WALK GIRL	KIZ	4-8	20000000

Şekil 6.13 Oyuncak fiyatı sorgusunun tablo olarak gerçeklenmesi

Buradan da görüldüğü gibi oyuncak fiyatı en çok 5000000 TL. olan "BEBEK" adında 2 oyuncak bulunmaktadır. Şekil 6.14 ve şekil 6.15'de istenilen form seçilerek gerçekleme nesne temelli olarak yapılmıştır.



Şekil 6.14 Oyuncak fiyatı sorgusunun 1'inci kaydının nesne temelli olarak gerçeklenmesi



Şekil 6.15 Oyuncak fiyatı sorgusunun 2'nci kaydının nesne temelli olarak gerçeklenmesi

6.7 Oyuncak Yaş Grubu Sorgusu Tasarımı

Bu sorguda 5 ile 7 yaş arasındaki çocuğa uygun olan oyuncaklar sorgulanacaktır. Bunun için yaş grubu "6-9" olan oyuncakların genel adı, ambalaj adı, cins, oy fiyatı bilgileri istenmektedir. Yaş grubu bölmesindeki ölçüt kısmına "6-9" yazılarak sorgu oluşturulmuştur. Şekil 6.16'da oyuncak yaş grubu sorgusunun tasarımı görülmektedir.

Alan:	OY GENEL AD	OY TÜRÜ	AMBALAJ ADI	CİNS	YAŞ GRUBU	OY FİYATI
Tablo:	OYUNCAK LİS	OYUNCAK LİS	OYUNCAK LİS	OYUNCAK	OYUNCAK LİS	OYUNCAK L
Sırala:						
Göster:	☑	☑	☑	☑	☑	☑
Ölçüt:					"6-9"	
veya:						

Şekil 6.16 Oyuncak yaş grubu sorgusunun tasarımı

6.7.1 Oyuncak yaş grubu sorgusunun gerçeklenmesi

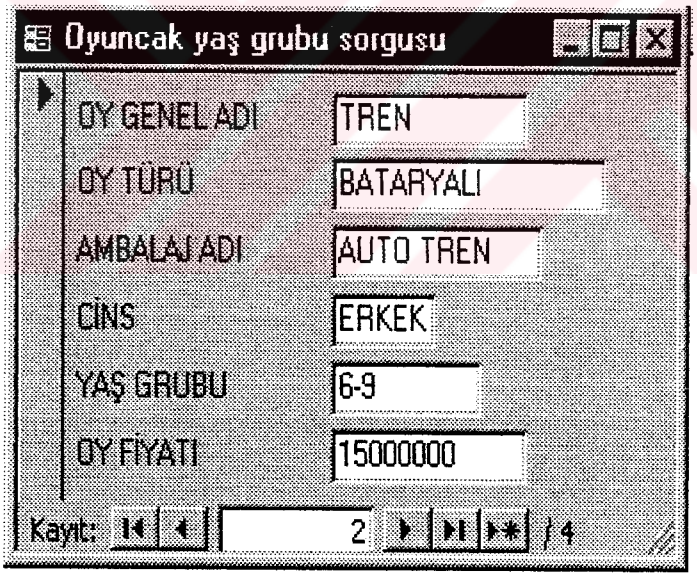
Şekil 6.17'de oyuncak yaş grubu sorgusu tablo olarak gerçeklenmiştir.

OY GENEL ADI	OY TÜRÜ	AMBALAJ ADI	CİNS	YAŞ GR	OY FİYATI
ARABA	BATARYALI	ROY ROYS	ERKEK	6-9	5000000
TREN	BARAYALI	AUTO TREN	ERKEK	6-9	15000000
ARABA	PİLLİ AMBULANS	CAR	ERKEK	6-9	7000000
YAP BOZ	100 PARÇA	PUZZLE 100	ORTAK	6-9	2000000

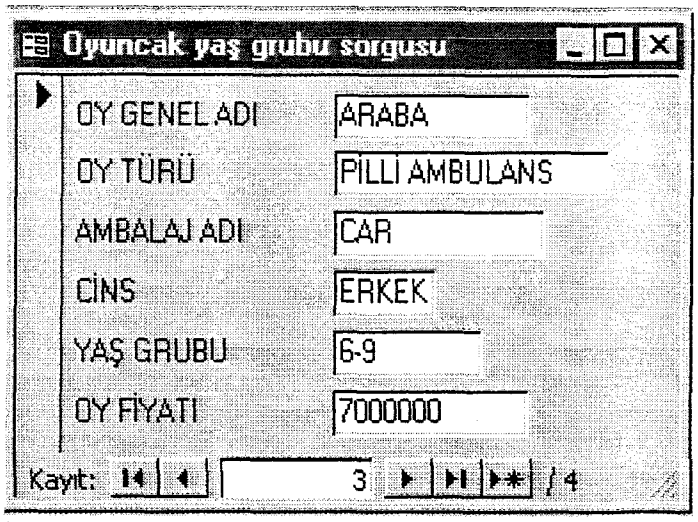
Şekil 6.17 Oyuncak yaş grubu sorgusunun tablo olarak gerçeklenmesi

Buradan da görüldüğü gibi oyuncak yaş grubu "6-9" olan 4 adet oyuncak çeşidi bulunmaktadır. Şekil 6.18, 6.19, 6.20 ve 6.21'de istenilen form seçilerek nesne temelli olarak 4 adet kaydın sırayla gerçeklenmeleri görülmektedir.

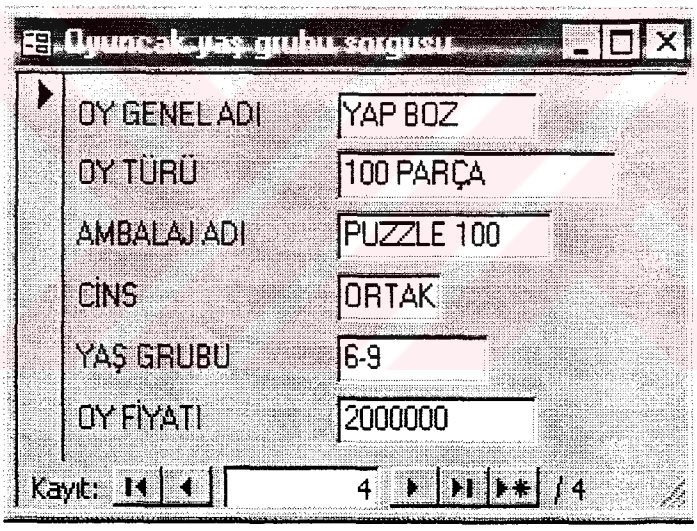
Şekil 6.18 Oyuncak yaş grubu sorgusu 1'inci kaydının nesne temelli olarak gerçeklenmesi



Şekil 6.19 Oyuncak yaş grubu sorgusu 2'nci kaydının nesne temelli olarak gerçeklemesi



Şeki 6.20 Oyuncak yaş grubu sorgusu 3'üncü kaydının nesne temelli olarak gerçeklenmesi



Şekil 6.21 Oyuncak yaş grubu sorgusu 4'üncü kaydının nesne temelli olarak gerçeklenmesi

6.8 Personel Cinsiyeti ve Görevi Sorgusu Tasarımı

Bu sorguda personel cinsi "KIZ" ve görevi "SATIŞ ELEMANI" olan personelin adı, soyadı, tel numaraları, adresi bilgileri istenmektedir. Şekil 6.22'de bu tasarım görülmektedir. Cinsiyeti bilgisi istenmemektedir.

Alan:	PERS ADI	SOYADI	GÖREVI	PERS ADRES	TEL	CINS
Tablo:	PERSONEL LIST	PERSONEL LIS	PERSONEL LIS	PERSONEL LI	PERSONE	PERSONEI
Sırala:						
Göster:	☑	☑	☑	☑	☑	☐
Ölçüt:			"satış elemanı"			"kız"
veya:						

Şekil 6.22 Personel görevi ve cinsiyeti sorgusu tasarımı

Sorgu hazırlanırken görevi bölmesindeki ölçüt hanesine "satış görevlisi" ve cins bölmesindeki ölçüt hanesine "kız" yazılarak diğer nesneler seçili duruma getirilir.

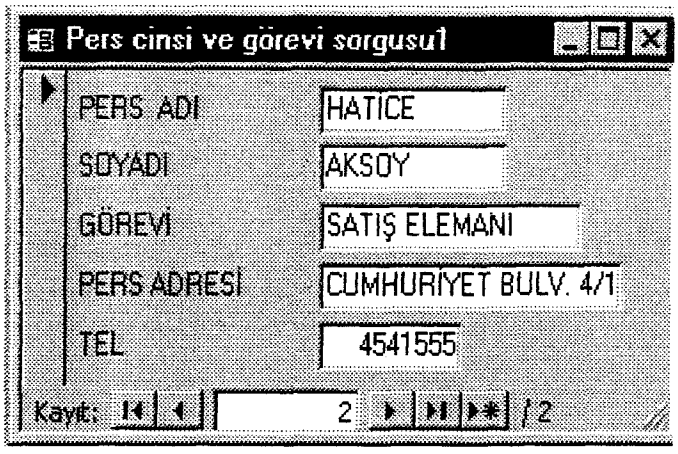
6.8.1 Personel görevi ve cinsiyeti sorgusunun gerçeklenmesi

Şekil 6.23'de personel cinsiyeti ve görevi sorgusu tablo olarak gerçeklenmiştir. Burada sorguyu sağlayan 2 adet kayıt bulunmuştur. Şekil 6.24 ve 6.25'de istenilen form biçimleri seçilerek sırayla sorgular nesne temelli olarak gerçeklenmiştir.

PERS ADI	SOYADI	GÖREVİ	PERS ADRESİ	TEL
FATMA	KOŞAR	SATIŞ ELEMANI	AYDINPINAR M. 15/7	5578785
HATİCE	AKSOY	SATIŞ ELEMANI	CUMHURİYET BULV. 4/10	4541555

Şekil 6.23 Personel görevi ve cinsiyeti sorgusunun tablo olarak gerçeklenmesi

Şekil 6.24 Personel görevi ve cinsiyeti sorgusu 1'inci kaydının nesne temelli gerçeklenmesi



Şekil 6.25 Personel görevi ve cinsiyeti sorgusu 2'nci kaydının nesne temelli gerçeklenmesi

6.9 Doğrudan SQL İle Sorgu Hazırlanması

6.9.1 Mevcut sorguyu değiştirme

Hazırlanmış olan bir sorguya SQL'de müdahele ederek değiştirebiliriz. Bunun için sorguda SQL seçilerek istenilen değişiklik yapılır. Daha önce hazırlanmış olan Oyuncak fiyatı sorgusunda oyuncak genel adı "bebek" ve oyuncak fiyatıençok 5milyon olan bir sorgu hazırlanmıştı. Bu sorguyu oyuncak genel adı "bebek" veya "araba" olan ve oyuncak fiyatıençok 5milyon olarak SQL'de değiştirilecektir. Bunun için sorgunun SQL'de hazırlanmış formuna ulaşılarak doğrudan müdahele ile istenilen satırlar silinir veya eklenir.

6.9.2 Sorgu yazma

Oyuncak fiyatı sorgusunun SQL deki karşılığı şu şekildedir.

Oyuncak fiyatı sorgusu: Seçme Sorgusu

```
SELECT [OYUNCAK LİSTESİ].[OY GENEL ADI], [OYUNCAK LİSTESİ].[OY TÜRÜ],
[OYUNCAK LİSTESİ].[AMBALAJ ADI], [OYUNCAK LİSTESİ].CİNS, [OYUNCAK
LİSTESİ].[YAŞ GRUBU], [OYUNCAK LİSTESİ].[OY FİYATI]
FROM [OYUNCAK LİSTESİ]
WHERE ((([OYUNCAK LİSTESİ].[OY GENEL ADI])="bebek") AND (([OYUNCAK
LİSTESİ].[OY FİYATI])<="5000000"));
```

Burada oyuncak fiyatı ençok 5 milyon olan ve oyuncak genel adı "bebek" olan sorgu hazırlanmıştı. Şimdi bu sorgu oyuncak genel adı "bebek" veya "araba" olan ve fiyatı ençok 5 milyon olarak değiştirilecektir.

Oyuncak fiyatı sorgusu: Seçme Sorgusu

```
SELECT [OYUNCAK LİSTESİ].[OY GENEL ADI], [OYUNCAK LİSTESİ].[OY TÜRÜ],
[OYUNCAK LİSTESİ].[AMBALAJ ADI], [OYUNCAK LİSTESİ].CİNS, [OYUNCAK
LİSTESİ].[YAŞ GRUBU], [OYUNCAK LİSTESİ].[OY FİYATI]
FROM [OYUNCAK LİSTESİ]
WHERE ((([OYUNCAK LİSTESİ].[OY GENEL ADI]="bebek") AND (([OYUNCAK
LİSTESİ].[OY FİYATI])<="5000000")) OR ((([OYUNCAK LİSTESİ].[OY GENEL
ADI])="araba"));
```

Bunun için SQL formu bu şekilde değiştirilir. Buradan görüldüğü gibi "veya" ile başlayan bölüm eklenmiştir.

BÖLÜM 7. SONUÇLAR

Bu çalışmada önce geleneksel veritabanı sistemlerinden ilişkili veri tabanı sistemleri tanıtılmış daha sonrada nesne temelli sistemler üzerinde durulmuştur. Nesne temelli sistemler ve nesne temelli programlama ortamları tanıtıldıktan sonra nesne temelli veritabanı sistemlerinin karmaşık veritabanı problemlerinin çözümüne getirdiği kolaylıklar irdelenmiş ve bir nesne temelli veritabanı uygulaması gerçekleştirilerek bu özellikler sergilenmiştir.

Görüldüğü üzere bir veri tabanı dosyasının nesne temelli bir programlama sistemi ile gerçeklenmesi hem tasarım anında tasarımcının sistem yapısını kurmasını kolaylaştırmakta hem de daha sonradan eklenebilecek ek özelliklerin sisteme çok kolay bir şekilde eklenebilmesini sağlamaktadır. Ayrıca geleneksel veri tabanı programları ile yapılması mümkün olmayan veya yapılması çok uzun program satırları gerektirecek özelliklerin de kolayca yapılabilmesini sağlamaktadır.

BÖLÜM 8. TARTIŞMA VE ÖNERİLER

Nesne temelli veri tabanlarının kullanıcıya tasarım anında sağladığı büyük kolaylıklar ve geleneksel veri tabanı programlarına karşı olan üstünlükleri nedeniyle; veri tabanı sistemlerinde nesne temelli veri tabanı programlarının kullanımı kaçınılmaz olmaktadır.

Nesne temelli veri tabanı programlarının daha fazla tatmin edici olabilmesi için veri tabanı programlarının hızlı bir şekilde işletilebilmesi gerekmektedir. Nesne temelli veri tabanı programlarının işletim hızlarının arttırılmasına yönelik yapılacak olan akademik çalışmalar büyük ilgi ve takdir toplayacaktır.

KAYNAKLAR

[1] DATE, C. J., "Database Systems", Addison-Wesley Publishing Company, November, 1994.

[2] TÜZÜN, M. Tunç., "Nesne Temelli C++ Programlama Dilini Kullanarak Problem Çözümüne Bir Örnek", Y. Lisans Tezi, İ.T.Ü., 1990.

[3] ÖZEL, G., "dBASE III-IV", Beta, İSTANBUL, 1990.

[4] BINDU, R. Rao., "Object-Oriented Databases", McGraw-Hill,Inc., 1994.

[5] ÇALBAŞ, S., "Görsel Programlama Teknikleri ve Bir Kalıp Tasarım Uygulaması", Y. Lisans Tezi, İ.T.Ü., 1996.

[6] CALVERT, C., "Delphi Unleashed", Sistem Yayıncılık, İSTANBUL, 1997.

[7] YANIK, M., "Microsoft Access", Beta, İSTANBUL, 1997.

ÖZGEÇMİŞ

1973 yılında Belçika'da doğdu. İlk öğrenimini Belçika'da, Orta ve Lise öğrenimini ise Kayseri Atatürk Lisesinde tamamladı. 1995 yılında İ.T.Ü. Sakarya Mühendislik Fakültesi Elektrik-Elektronik Mühendisliği bölümünü ikincilik derecesiyle bitirdi. 1996-1998 yılları arasında Abant İzzet Baysal Üniversitesinde Öğretim Görevlisi olarak çalıştı. Milli Eğitim Bakanlığının 1997 yılında yapmış olduğu Yurt Dışı Lisans Üstü Sınavını kazanarak, SAKARYA ÜNİVERSİTESİ adına Bilgisayar Donanımı Alanında A.B.D.'de Doktora yapma hakkı kazandı.