

YILDIZ TEKNİK ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

Object-Ori. Dil. Üze, bir Uyg.
Kütüp.-Arşiv Otom.

YÜKSEK LİSANS TEZİ

Zekiye Emiroğlu

1992

120-009
YILDIZ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

OBJECT - ORIENTED DİLLER
ÜZERİNE BİR UYGULAMA:
KÜTÜPHANE - ARŞİV OTOMASYONU

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜH. ZEKİYE EMİROĞLU

TEZ YÖNETİCİSİ
Prof.Dr. MEHMET YAHYA KARSLIGİL

İSTANBUL 1992

YILDIZ TEKNİK ÜNİVERSİTESİ
KÜTÜPHANE DOKÜMANTASYON
DAİRE BAŞKANLIĞI

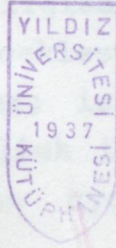
	- R 368
Kot	: 62
Alındığı Yer	: Fen Bil. Ens.
	
Tarih	: 13. 4. 95
Fatura	:
Fiyatı	: 120.000.
Ayniyat No	: 1-4
Kayıt No	: 51041
UDC	:
Ek	:

YILDIZ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

OBJECT - ORIENTED DİLLER

ÜZERİNE BİR UYGULAMA:

KÜTÜPHANE - ARŞİV OTOMASYONU



YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜH. ZEKİYE EMİROĞLU

TEZ YÖNETİCİSİ

Prof.Dr. MEHMET YAHYA KARSLIĞİL

Y.T.O.

KÜTÜPHANE DOK. DAİ. BAŞKANLIĞI

İSTANBUL 1992

Yakın gelecekte hasar görmüş elyazmalarının restorasyonuna
alt bir yazılış da bu ÖNSÖZ e eklenecektir.

Veritabanı uygulamaları bilgisayar dünyasının en hızlı
ilerleyen alanlarından biridir. Bilgisayarda saklanacak ve
üzerinde işlem yapılacak veri miktarının artması bu
ilerlemeyi hızlandırmış; bilgisayarların büyük miktarda veri
üzerinde hatasız ve hızlı işlem yapabilme özelliği
veritabanı uygulamalarını bankacılıktan kütüphane
otomasyonuna birçok alanda önemli bir yere getirmiştir.

Süleymaniye Kütüphanesi 125.000 El yazması eser ile
Türkiye'nin en zengin yazma eser kütüphanesidir. Türk ve
yabancı araştırmacılar için büyük önem taşıyan bu
kütüphanede , kütüphane hizmetlerini kolaylaştırmak amacıyla
bir bilgisayar sistemi kurulmuş, gereken veritabanı yazılımı
Yıldız Üniversitesi, Bilgisayar Bilimleri ve Mühendisliği
Bölümü tarafından gerçekleştirilmiştir. Ancak bilgisayar
dünyasındaki ilerlemeler; işlem hızı, çok kullanıcılık ve
daha büyük miktarda bellek ihtiyaçları bu sistemde günümüz
koşul ve ihtiyaçlarına uygun değişiklikler yapılmasını
gerektirmiştir.

Bu çalışmada daha güçlü bir veritabanı yönetim sistemi
kullanılarak Süleymaniye Kütüphanesi için yeni bir Kütüphane
Otomasyonu yazılımı geliştirilmesi planlanmıştır. Daha önce
girilmiş bilgilerin yeni sisteme aktarımı sağlanmış ve
gereken programlar ORACLE İlişkisel Veri Tabanı Ynetim
Sistemi altında ORACLE SQL ve C++ kullanılarak
geliştirilmiştir. Object Oriented Programlama dillerinin
verimlilik, uyumluluk, modülerlik ve yeniden kullanılma
özellikleri gözönünde tutularak C++ seçilmiş ve SQL ile
entegre edilmiştir. Bu yazılım çok kullanıcıli bir sistem
altında tüm kütüphanecilik fonksiyonlarını kapsamaktadır.

Yakın gelecekte hasar görmüş elyazmalarının restorasyonuna ait bir yazılım da bu programa eklenecektir.

Bu çalışmamda bana yol gösteren, tez yöneticim, Bilgisayar Bilimleri ve Mühendisliği Bölümü Başkanı Prof. M. Yahya KARSLIGİL'e ve çalışmalarımda bana destek olan tüm araştırma görevlisi arkadaşlarıma teşekkür ederim.

This importance comes from the amount of data which will be stored and manipulated on computers. The speed and correctness of computers make the database applications really important on different areas from banking to library systems.

Süleymaniye Library is the biggest manuscript library in Turkey over 125.000 manuscripts and assists many Turkish and foreign researchers on this area. A computer system installed and a library database application developed by Yıldız University, Computer Science and Engineering Department to support researchers. With the development on computer science, processing speed, multiuser support and need of more storage made this system obsolete and it's become a necessity to upgrade this software with a new, powerful relational database management system.

With this thesis, we are planning to write a new library automation software for Süleymaniye Library and make the data migration of previously entered information. The program are developed under the ORACLE RDBMS with ORACLE SQL and C++. Because of the productivity, compatibility, modularity and reusability of object oriented approach, we choose C++ and integrated with this powerful database software ORACLE SQL. This library automation software includes all library services in a multiuser system. In the near future, a software for restoration of damaged manuscripts will be integrated to this software.

I would like to thank my supervisor, Head of the Computer
Sciences and Engineering Department, Prof. Dr. N. Yahya
KARSLIGIL and all research assistants in my department who

ABSTRACT

Nowadays, database applications become one of the important areas of computer world. The reason of this importance comes from the amount of data which will be stored and manipulated on computers. The speed and correctness of computers make the database applications really important on different areas from banking to library systems.

Süleymaniye Library is the biggest manuscript library in Turkey over 125.000 manuscripts and assists many Turkish and foreign researchers on this area. A computer system installed and a library database application developed by Yıldız University, Computer Science and Engineering Department to support researchers. With the development on computer sciences, processing speed, multiuser support and need of more storage made this system obsolete and it's became a necessity to upgrade this software with a new, powerful relational database management system.

With this thesis, we are planning to write a new library automation software for Süleymaniye Library and make the data migration of previously entered information. The programs are developed under the ORACLE RDBMS with ORACLE SQL and C++. Because of the productivity, compatibility, modularity and reusability of object oriented approach, we choose C++ and integrated with this powerful database software ORACLE SQL. This library automation software includes all library services in a multiuser system. In the near future, a software for restoration of damaged manuscripts will be integrated to this software.

I would like to thank my supervisor, Head of the Computer Sciences and Engineering Department, Prof.Dr. M. Yahya KARSLIGİL and all research assistants in my department who supported me on my thesis.

1.2. Veritabanı Sisteminin Temel Elemanları	1
1.3. Veritabanı Kullanımının Getirdiği Kolaylıklar	3
1.4. Veritabanı Oluşturma Aşamaları	4

2 - KÜTÜPHANECİLİK NEDİR ?

2.1. Kütüphane nedir ?	9
2.2. Kütüphanecilik Tekniği	10
2.3. Kütüphanecilikte Bilgisayar Kullanımı	10

3 - BİR KÜTÜPHANE YAZILIMI

3.1. Süleymaniye Kütüphanesi	12
3.2. Süleymaniye Kütüphanesi Bilgisayar Yazılımı	12
3.3. Yazılımda Kullanılan Tablolar	16

4 - VERİTABANI UYGULAMALARINDA KULLANILAN BAŞLI YAZILIMLARIN ÖZELLİKLERİ

4.1. dBASE III PLUS	21
4.1.1. Teknik Özellikleri	21
4.1.2. Dosya Yapısı, Veri ve Dosya Tipleri	22
4.1.3. dBase III Plus Kullanımı	24
4.2. ORACLE Alışıkisel Veri Tabanı Yönetim Sistemi	26
4.2.1. ORACLE RDBMS Nedir ?	26
4.2.2. ORACLE Veri Tabanı ve ORACLE Instance	27
4.2.3. Paylaşılabilir ve Dağıtılmış Veri Tabanı Sistemleri	30
4.2.4. Entity Relationship ve Mantıksal Tablo Yapısı	32
4.2.5. ORACLE RDBMS Fiziksel Dosya Yapısı	33
4.2.6. ORACLE SQL	36
4.2.7. ORACLE SQL'deki Veri Tipleri	37
4.2.8. SQL'deki Komut Türleri	38
4.2.9. ORACLE RDBMS'te ki Yardımcı Programlar	39

5 - Süleymaniye Kütüphanesi Otomasyonu

6 - Programlar

KAYNAKLAR

ÖZGEÇMİŞ

ÖNSÖZ

ABSTRACT

1 - VERİTABANI NEDİR ?

1.1. Veritabanı Nedir ?	1
1.2. Veritabanı Sisteminin Temel Elemanları	1
1.3. Veritabanı Kullanımının Getirdiği Kolaylıklar	3
1.4. Veritabanı Oluşturma Aşamaları	4

2 - KÜTÜPHANECİLİK NEDİR ?

2.1. Kütüphane nedir ?	9
2.2. Kütüphanecilik Tekniği	10
2.3. Kütüphanecilikte Bilgisayar Kullanımı	10

3 - BİR KÜTÜPHANE YAZILIMI

3.1. Süleymaniye Kütüphanesi	12
3.2. Süleymaniye Kütüphanesi Bilgisayar Yazılımı	12
3.3. Yazılımda Kullanılan Tablolar	16

4 - VERİTABANI UYGULAMALARINDA KULLANILAN BAZI YAZILIMLARIN ÖZELLİKLERİ

4.1. dBASE III PLUS	21
4.1.1. Teknik Özellikleri	21
4.1.2. Dosya Yapısı, Veri ve Dosya Tipleri	22
4.1.3. dBase III Plus Kullanımı	24
4.2. ORACLE İlişkisel Veri Tabanı Yönetim Sistemi	26
4.2.1. ORACLE RDBMS Nedir ?	26
4.2.2. ORACLE Veri Tabanı ve ORACLE Instance	27
4.2.3. Paylaşılabilir ve Dağıtılmış Veri Tabanı Sistemleri	30
4.2.4. Entitiy Relationship ve Mantıksal Tablo Yapısı	32
4.2.5. ORACLE RDBMS Fiziksel Dosya Yapısı	33
4.2.6. ORACLE SQL	36
4.2.7. ORACLE SQL deki Veri Tipleri	37
4.2.8. SQL deki Komut Türleri	38
4.2.9. ORACLE RDBMS te ki Yardımcı Programlar	39

5 - Süleymaniye Kütüphanesi Otomasyonu

6 - Programlar

KAYNAKLAR

ÖZGEÇMİŞ

1 - VERİ TABANI NEDİR ?

1.1. Veri Tabanı Nedir ?

Veri tabanı uygulamaları bilgisayar bilimlerinin en hızlı ilerleyen dallarından biridir. Bilgisayarda saklanacak ve üzerinde işlem yapılacak veri miktarının normal file sistemlerinin limitlerini aşması sonucu veri tabanı oluşturma ihtiyacı ortaya çıkmıştır. Veri tabanı temelde bilgisayara dayalı bir veri saklama sistemidir. En önemli görevi veriyi depolamak ve işlenerek bilgi haline gelmesini sağlamaktır. Veri; veri tabanına kayıtlı herhangi bir işlem görmemiş fiziksel değer (sayılar, karakterler, şekil veya ses toplulukları gibi), bilgi ise bu ham değerlerin işlenerek anlamlı hale gelmiş şeklidir. Veri tabanını oluşturmak için öncelikle verinin mantıksal yapısı tarif edilir. Bu tarif sırasında veri modellerinden faydalanılır. Daha sonra bir bilgisayar dili yardımıyla bu veri yapısı oluşturulur ve üzerinde işlem yapılır. Veri tabanı uygulamalarında tüm bilgisayar dilleri kullanılabilir. Ancak yapılan veri tabanı uygulamasının daha verimli olabilmesi için özel veri tabanı dilleri oluşturulmuştur.

1.2. Veri Tabanı Sisteminin Temel Elemanları

Bir veri tabanı sisteminin Veri, Yazılım, Donanım ve Kullanıcı olmak üzere dört temel elemanı vardır.

a) Veri: Depolanan veri bir veya birkaç veri tabanına bölünebilir. Buna yapılacak uygulamaya göre karar verilir. Genelde veriler sistem içinde tek bir veri tabanında toplanır. Veri tabanı entegre ve paylaşılabilir bir topluluktur. Birden fazla yerde kullanılacak aynı tür verilerin tekrarlanması önlenerek tek dosyada toplanması veri tabanının entegre olma özelliğini gösterir. Veri tabanı içindeki veri parçalarının değişik kullanıcılar tarafından

aynı veya farklı amaçlarla kullanılması ise veritabanının paylaşılabirlik özelliğini gösterir. Burada dikkat edilmesi gereken önemli bir nokta aynı veri parçasına n tane kullanıcının aynı anda ulaşabilmesi gereğidir. Bu da veritabanı sisteminin çok kullanıcı (Multiuser) bir sistemle desteklenmesi gereğini ortaya çıkartmıştır.

b) Donanım : Veri tabanı uygulamalarında kullanılacak cihazın yüksek erişim hızına sahip yardımcı bellek birimlerine sahip olması gerekmektedir. Bu birimlerin kapasitesi ve cihaz seçimi yapılacak veri tabanı uygulamasının ihtiyaçlarına uygun olarak seçilebilir.

c) Yazılım : Fiziksel veri tabanı ile sistem kullanıcıları arasındaki iletişimi sağlayan yazılıma Veri Tabanı Yönetim Sistemi adı verilir. Veri tabanına ulaşabilmek için kullanıcıdan gelen tüm istekler Veri tabanı yönetim sistemi tarafından yerine getirilir.

d) Kullanıcılar : Bir veri tabanı yönetim sisteminde Uygulama Programcısı, Veri tabanı yöneticisi ve son kullanıcı olmak üzere 3 sınıf kullanıcı vardır. Uygulama programcısı bir veri tabanı dili kullanarak yapılacak uygulamanın gerektirdiği programı yazar. Veri tabanı yöneticisi, veri tabanı üzerinde hertürlü yetkiye sahip olan kişidir. Yazılımın kurulması ve servis verilmesi, optimum veri tabanı kullanımı için gerekli ayarlamalar, veri duyarlılığı, veri bütünlüğü, veri güvenliği, veri depolama alanı, verilerin yedeklenmesi ve yeniden yüklenmesi gibi işlemlerle ilgilenir. Son kullanıcı ise sistemin kullandığı sorgulama dili veya uygulama programcısının yazdığı program yardımıyla veritabanına giriş/çıkış yapan kişidir.

1.3. Veri Tabanı Kullanımının Getirdiği Kolaylıklar

İşletmelerde tüm departmanlarda gereken bilgilerin ayrı ayrı saklanması bu veriler üzerinde kontrolü zorlaştırmakta, hatta imkansız hale getirmektedir. Veri tabanı sistemi kullanımı ise işletmeye tüm veriler üzerinde merkezi kontrol imkanı verir. Veri tabanı sisteminin diğer avantajları ise şöyle sıralanabilir.

a) Veriler arasında tekrarlar önlenir.

Bir çok işletmede kullanılan veriler ayrı dosyalarda saklanmaktadır. Mesela bir işletmedeki personel bilgileri ayrı bir dosyada ve bu personelin maaşını hesaplamak için gerekli bilgiler, özel bilgilerle birlikte bir başka bölümde saklandığı takdirde aynı verilerin tekrarlanması kaçınılmazdır. Ancak bilgisayarda saklanan ve birbirleriyle entegre çalışan dosyalar yardımıyla bu tekrar önlenir.

b) Veriler arası uyumsuzluk ortadan kalkar.

Değişik birimlerde veriler üzerinde yapılan hesaplamalar ve işlemlerdeki hata veya tekrarlamalardan doğan problemler ortadan kalkar.

c) Veriler değişik birimler tarafından paylaşılabilir.

Aynı verinin farklı kısımlarının değişik birimler tarafından kullanılması veya aynı tür verilerin farklı işlemlerde kullanılması bilgisayarda saklanan verilerin paylaşılabilir olma özelliğinin getirdiği sonuçlardır. Ayrıca yeni uygulamalarda da önceden depolanmış veriler kullanılarak sürekli yeni dosyaların oluşması önlenir.

d) Veriler arasında standartizasyon gerçekleştirilir.

Aynı tür bilgiler, işletmelerin değişik bölümlerinde değişik şekillerde yorumlanarak kullanılmaktadır. Mesela bir departmanda işletmenin değişik bölümleri için E8,C7 gibi simgeler; diğer bir departmanda ise ELEKTRİK İŞLERİ, MUHASEBE gibi kelimeler kullanılıyorsa bu pekçok karışıklığa yol açar.Bilgisayar ve veri tabanı kullanımı bu karışıklığı önleyerek veriler üzerinde standart oluşmasını ve kullanımını sağlar.

e) Veriler üzerinde güvenlik önlemleri alınır.

Verilere erişimin kontrol edilebilebilmesi, önemli verilere erişimde yetkilerin verilebilmesi ve bu önlemlerin şifrelerle desteklenebilmesi bir veritabanı sisteminin getirdiği özelliklerdir. Eğer bu gibi önlemler alınmazsa bilgisayarda kurulan veri tabanı sistemi, geleneksel veri tabanı sistemlerinden daha fazla güvenlik riski altında kalır.

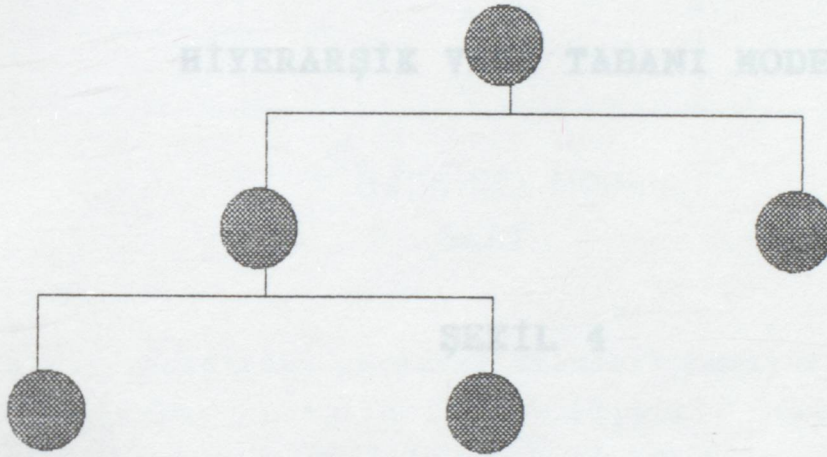
f) Verilerin doğruluğu kontrol edilebilir.

Daha verilerin girişi sırasında verilerin ve yapılan işlemlerin doğruluğu kontrol edilebilir. Mesela bazı alanların (verilerin), boş-0 değer taşıması, karakter ve nümerik bilgiler arasındaki karışıklıklar, yanlış tarih girişleri, nümerik değerlerde basamak hataları kontrol edilerek geleneksel yöntemlerle kontrol edilemeyen kullanıcı hataları ortadan kalkar.

1.4. Veri Tabanı Oluşturma

Yapılan veri tabanı uygulamasında; seçilen yardımcı bellek birimleri üzerinde birtakım 0 ve 1 değerlerinden oluşan fiziksel verinin kullanıcı tarafından anlaşılır, kullanılır hale gelmesi için veri modellerinden yararlanarak verinin mantıksal yapısı oluşturulur. Veri modeli,verinin

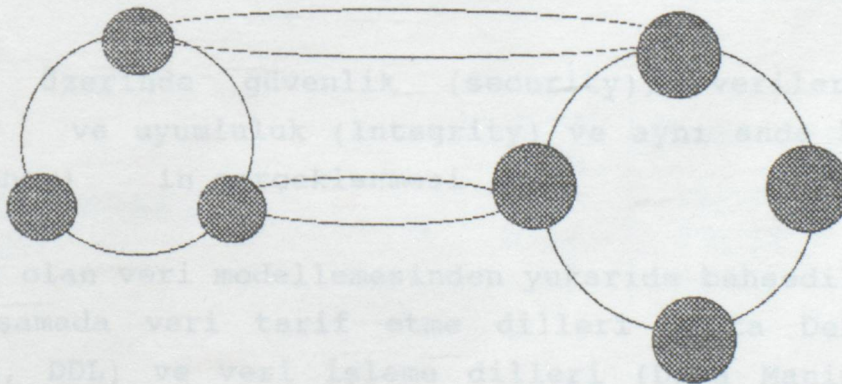
organizasyon içindeki girişlerinin, diğer veriler ile ilişkilerinin ve işlemlerinin tarifinden oluşur. Veri modelleri genel olarak Hiyerarşik Veri Modeli (Hierarchical Data Model), İlişkisel Veri modeli (Relational Data Model) , Ağ Veri Modeli (Network Data Model) olarak 3 grupta toplanabilir. Hiyerarşik Veri Modeli, ağaç yapısına benzer olarak, veriler arasında birebir ve birden çokluya ilişkilerden oluşur.



HİYERARŞİK MODEL

ŞEKİL 1

Ağ Veri Modeli, record yapıları ve record yapıları arasındaki ilişkiler ile gösterilir.

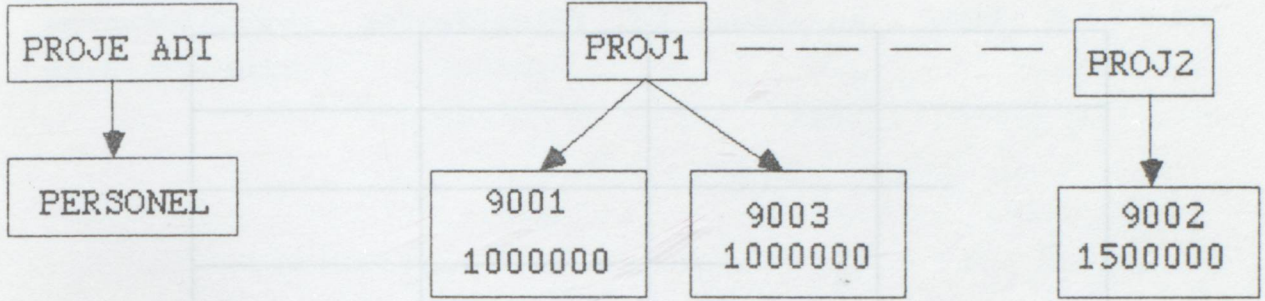


AĞ MODELİ

ŞEKİL 2

İlişkisel Veri Modelinde ise veri iki boyutlu tablolar halinde tutulur ve bu tablolar arasındaki ilişkiler tarif edilir.

PERSONEL BİLGİLERİ



HİYERARŞİK VERİ TABANI MODELİ

İLİŞKİSEL MODEL

ŞEKİL 3

ŞEKİL 4

Elimizde bir şirketteki personel bilgileri, maaşları ve çalıştıkları bölümler bilgileri bulunduğu düşünelim. Bu bilgilerin hiyerarşik ve ilişkisel yöntemlerle modellenmesi şöyle gösterilebilir.

Bir veri tabanı yönetim sistemi; veriyi saklamak görevinin yanı sıra kullanıcıyı - programcıyı donanım seviyesindeki ayrıntılardan uzak tutarak fiziksel veri tabanına erişimi sağlar. Bir veritabanı yönetim sistemi yazılımı 3 aşamada gerçekleştirilir.

- Veri modelinin oluşturulması
- Bir veritabanı dilinden yararlanarak verinin tarifi ve işlenmesi
- Veri üzerinde güvenlik (security), veriler arası ilişkiler ve uyumluluk (Integrity) ve aynı anda kullanım (concurrency) in gerçekleştirilmesi

İlk aşama olan veri modellemesinden yukarıda bahsedilmiştir. İkinci aşamada veri tarif etme dilleri (Data Definition Languages, DDL) ve veri işleme dilleri (Data Manipulation

İlişkisel Veri Modelinde ise veri iki boyutlu tablolar içinde tutulur ve bu tablolar arasındaki ilişkiler tarif edilir.

İLİŞKİSEL MODEL
ŞEKİL 3

Elimizde bir şirketteki personel isimleri, maaşları ve çalıştıkları bölümler bilgileri bulunduğunu düşünelim. Bu bilgilerin hiyerarşik ve ilişkisel yöntemlerle modellenmesi şöyle gösterilebilir.

PERSONEL BİLGİLERİ

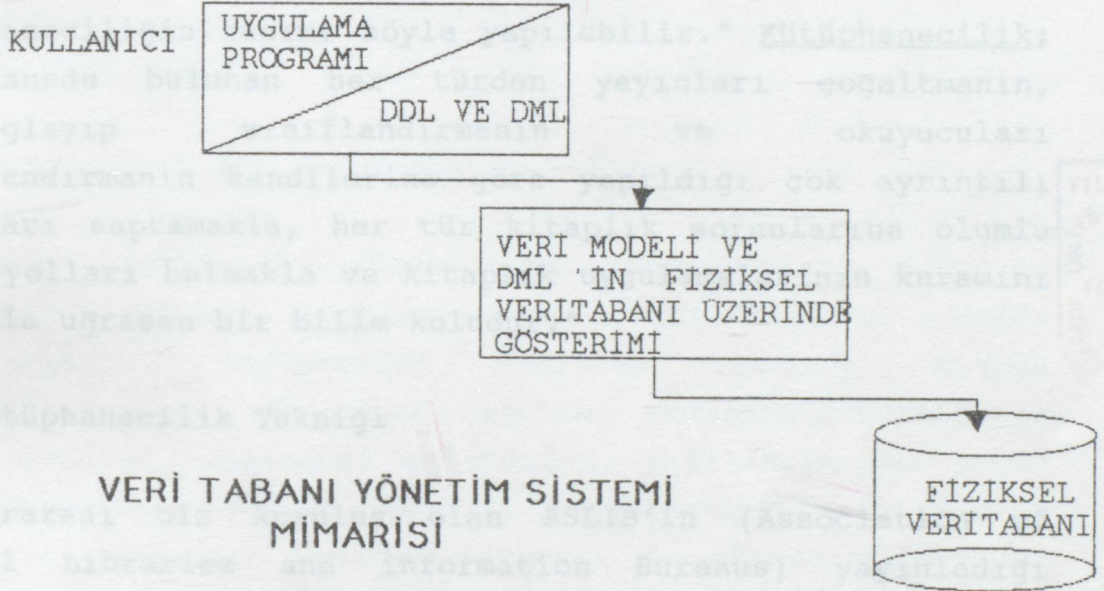
KİŞİLER	MAAŞLAR
{ SICIL NO İSİMLER PROJELER }	{ MAAŞLAR }

SICIL NO	ADI	MAAŞ
9001	Z.EMİROĞLU	1000000
9002	E.KARSLIGİL	1500000
9003	G.ESENDAL	1000000

İLİŞKİSEL VERİTABANI
MODELİ

SICIL NO	PROJE ADI
9001	PROJ1
9002	PROJ2
9003	PROJ1

Languages, DML) kullanılır. Bunlar fiziksel veri tabanı ile kullanıcı arasındaki ilişkiyi sağlar.Üçüncü aşama ise daha çok veri tabanı yöneticisi (Database Administrator,DBA) in sorumluluğunda gerçekleşir. Bu aşamalar Şekil 5 ' te görülmektedir.

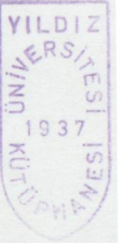


ŞEKİL 5

2 - KÜTÜPHANECİLİK NEDİR ?

2.1. Kütüphane Nedir ?

Kütüphane kelimesi Arapçada kitaplar anlamına gelen Kütüb ve Farsçada ev anlamına gelen hane kelimelerinden türetilmiştir. Kitapların toplandığı yer anlamındadır. Kütüphaneler, bilgi taşıyan yayınlardan herkesin kolayca ve çok hızlı bir şekilde yararlanmasını sağlar. Kütüphanecilik mesleği İ.Ö.2500 lü yıllarda ortaya çıkmıştır.II.Ramses Ozymandias'ın bu yıllarda Teb'de kurduğu kütüphane tarihteki en eski kütüphanedir.Çeşitli gelişmelerle günümüze ulaşan kütüphaneciliğin tanımı şöyle yapılabilir." Kütüphanecilik; kütüphanede bulunan her türden yayınları çoğaltmanın, kataloglayıp sınıflandırmanın ve okuyucuları yararlandırmanın kendilerine göre yapıldığı çok ayrıntılı kuralları saptamakla, her tür kitaplık sorunlarına olumlu çözüm yolları bulmakla ve kitaplık uygulamalarının kuramını koymakla uğraşan bir bilim koludur."



2.2.Kütüphanecilik Tekniği

Uluslararası bir kuruluş olan ASLIB'in (Association of Special Libraries and information Bureaus) yayınladığı kütüphanecilik sistem şeması şöyledir.

- Kütüphane
 - Yönetsel sistem
 - Uygulama sistemi
 - Yayınları Çoğaltma (Sağlama)
 - Seçme
 - İsmarlama
 - Alma
 - Yayınları Düzenleme
 - Envanter
 - Kataloglama

- Etiketleme
- Depolama
- Yararlandırma
- Yer Belirleme
- Listeleme
- Ödünç verme
- Okuyucu için yayın ayırma
- Anımsatma
- Fotokopi
- Kitaplıklararası ödünç verme
- Koruma
- Ciltletme
- Ödeme
- Kayıttan düşme

2.3 Kütüphanecilikte Bilgisayar Kullanımı

Günümüzde Bilgisayarlar yüksek hıza sahip olmaları, doğru ve güvenilir işlem yapmaları ve getirdikleri diğer kolaylıklarla endüstriden eğitime pekçok alanda kullanılmaktadır. Yukarıda sistem şemasını verdiğimiz kütüphanecilik alanındaki gelişmeler, yayınların her geçen gün hızla çoğalması ve çağımızın gereklerine uygun olarak insanların zamandan kazanma ihtiyacı bu alanda bilgisayar kullanımını gerektirmektedir. Kütüphaneler özellikle son 20 yılda bilgisayar kullanımı artmış ve kompleks organizasyonlardır. Bilgisayarlar kütüphaneciliğin tüm işlem safhalarında kullanılabilir. Günümüzde kütüphane otomasyonunun gerektirdiği yazılım ihtiyaçları profesyonel programcılar veya yazılım firmaları tarafından sağlanmaktadır. Veritabanı uygulamalarının güzel bir örneği olan kütüphane yazılımlarında veritabanı uzmanları ve uzman kütüphaneciler birlikte çalışarak ihtiyaca uygun yazılım üretmektedirler. Özellikle kütüphaneleri bilgisayar ağı yardımıyla birbirine bağlayarak kısa zamanda birçok yerde yayın taraması yapılabilmesinin sağlanması bilgisayar

kullanımının kütüphaneciliğe getirdiği bir yeniliktir. Ayrıca kataloglama, ödünç verme veya yararlandırmada, kitap seçiminde, çeşitli istatistikî bilgiler sağlanmasında bilgisayar kullanımının getirdiği kolaylıklardır. Bir kütüphanede kullanılacak kütüphane yazılımı genel olarak şu aşamaları içerir.

a) Kütüphanede bulunan kitap, periodik, vs. tüm eserlerin bilgisayara kaydedilmesi. Bu aşamada yapılan programlar yardımıyla bir veya birkaç kütüphane personeli tarafından eserlerin gereken bilgileri bilgisayara kaydedilir.

b) Kataloglama, kitap taraması, sipariş, muhasebe işlemleri. Yazılan programlar yardımıyla okuyucuların kitap taraması işlemleri çok kısa bir sürede yapılmaktadır. Hatta bir ağ yoluyla birbirine bağlı kütüphanelerde kitap taraması yapmakta mümkündür.

c) Ödünç verme, ödünç verilen kitapların kontrolü, hatırlatmalar.

Bu işlemlerde bilgisayar yoluyla çok kısa bir zamanda yapılmakta, ödünç verilen kitapların kontrolü daha hassas bir şekilde gerçekleştirilmektedir. Ayrıca hangi kitapların hangi sıklıkta okunduğu buna bağlı olarak kitap çoğaltma ihtiyacı gibi istatistikî bilgilerin; araştırmacılar için çok uzun süreli taramalar yoluyla elde edilebilecek bilgilerin çok kısa zaman aralıkları içinde edinilmesi mümkündür. Yine tarihi değer ve önem taşıyan eserlerin bilgisayara aktarılması bu yolla erişim ve restorasyonu bilgisayarın kütüphaneciliğe getirdiği yeniliklerdendir.

3- BİR KÜTÜPHANE YAZILIMI

3.1. Süleymaniye Kütüphanesi

Cumhuriyetin ilanından sonra Osmanlı döneminden kalan İstanbul Kütüphaneleri Süleymaniye Kütüphanesi, Millet Kitaplığı ve Beyazıt kütüphanesi olmak üzere 3 büyük kütüphane çevresinde toplanmıştır. Bunlardan Süleymaniye Kütüphanesi Türkiye'nin en geniş el yazması eserler koleksiyonuna sahip tek kütüphanesidir. 1557 yılında Kanuni Sultan Süleyman'ın isteğiyle Süleymaniye Külliyesi içinde kurulmuştur. Kütüphaneye Atıf Efendi, Hacı Selim Ağa, Köprülü, Nuruosmaniye ve Ragıppaşa Kütüphaneleri de bağlıdır.

Basımevleri kurulup yayılmadan önce elle yazılıp çoğaltılmış kitaplara El Yazması veya Yazma adı verilir. Resim ve tezhiplerle süslenmiş olup, giriş ve bölüm başlarında süslü başlıklar konulmuş ve metnin içine minyatürler yapılmıştır. Çok değerli olan bu el yazması eserler kütüphanelerde özel bölümlerde saklanarak, özel yöntemlerle kataloglanır. Süleymaniye Kütüphanesi İslami yazmalar açısından dünyanın en önemli kütüphanelerinden biridir. Süleymaniye'de 100.000, diğer bağlı kütüphanelerde 25.000 olmak üzere toplam 125.000 el yazması eser bulunmaktadır.

3.2. Süleymaniye Kütüphanesi Bilgisayar Yazılımı

Yerli ve yabancı araştırmacılar, öğrenciler ve ilgili kişiler tarafından bu el yazması eserler üzerine yapılan araştırmalarda bilgisayar kullanımı büyük kolaylık getirmektedir. Bu eserlere ait bilgilerin bilgisayara aktarılması, çeşitli sorgulama yöntemleriyle kolayca erişilebilmesi hem kütüphaneciler hem de bu kütüphaneden yararlanan kişiler için büyük zaman tasarrufu sağlamakta ve aynı zamanda daha sağlıklı bir araştırma yapılmasına imkan vermektedir. Bu amaçla 1988 yılında Yıldız Üniversitesi ,

Bilgisayar Bilimleri ve Mühendisliği Bölümünce dBase III Plus kullanılarak yazma bir kütüphane yazılımı gerçekleştirilmiştir. Şu anda kütüphanede el yazması eserlerin bilgisayara aktarılması, aranması, listelenmesi, sıralanması, sorgulanması gibi işlemlerde bu yazılım kullanılmaktadır. Bu yazılım Arş.Grv. Füsun Yaman tarafından geliştirilmiştir. Kütüphane elemanlarından Melek Gençboyacı tarafından bugüne kadar 30.000'i aşkın yazma eser bilgisi bilgisayara kaydedilmiştir. Süleymaniye Kütüphanesinde bulunan el yazması eserler dillerine göre Türkçe, Arapça, Farsça ve diğer diller olmak üzere 4 gruba ayrılmıştır. Konuları ise İslam Dilleri Tasnif Cetveline göre 11 ana başlıkta toplanmıştır. Bu temel konular aynı zamanda kendi içlerinde toplam 130 konuya ayrılmaktadır.(İslam İlimleri Tasnif Cetveline Göre Konu Başlıkları,Tablo 1)

Yazmaların bulunduğu kütüphaneler ise Süleymaniye, Atıf Efendi, Hacı Selim Ağa, Köprülü, Nuruosmaniye ve Ragıp Paşa olmak üzere 6 tanedir. Ayrıca bu yazmalar 98 ayrı koleksiyonda toplanmıştır. (Yazmaların Bulunduğu Koleksiyonlar, Tablo 2)

El yazması eserler ayrıca hat tipine göre de gruplanmaktadır. Yazma Eserlerde kullanılan 17 ayrı hat tipi vardır.(Yazma Eserlerde Kullanılan hat Tipleri,tablo 3)

Yazma eserlerin kaydı ve üzerinde gerekli sorgulamalar yapılması için geliştirilen kütüphane veritabanı Arapça, Farsça,Türkçe ve diğer dillerdeki kitapların saklandığı 4 veritabanı dosyası üzerine kurulmuştur.Veritabanına bilgi girişi, güncelleme ve sorgulama işlemlerini gerçekleştiren programlar aşağıda belirtilen 7 grupta toplanmıştır.

- 1 - Eserlerin Kayıtlarının yapılması
- 2 - Eser bilgisinin silinmesi
- 3 - Eser bilgisinin düzeltilmesi

- 4 - Eserin aranması
- 5 - Çeşitli listelerin yazıcıdan alınması
- 6 - Eserler üzerinde çeşitli sorgu ve listeler
- 7 - Eserlerin sıralanması

1 - Eserlerin Kayıtlarının Yapılması

Bu bölümde her bir yazma eser için yukarıdaki özelliklerde dahil olmak üzere 55 ayrı özellik girişi yapılmaktadır. (Yazma Eserlerin Kaydında Girişi Yapılan Özellikler, Tablo 4)

Bu girişlerde giriş hızını kolaylaştırmak üzere Eserin Dili, Tasnif No, Konusu, Kütüphanesi, Koleksiyonu, Hat tipi bilgileri otomatik olarak seçilmektedir. Her yazma eser için 55 alanda yaklaşık 2600 karakterlik bilgi girilmektedir. Bu bilgilerin 24 tanesi aramalarda key olarak kullanım özelliğine sahiptir.

2 - Eser kayıtlarının silinmesi

Bu bölümde yanlış veya hatalı yapılan kaydın silinmesi işlemleri yapılmaktadır.

3 - Eser bilgilerinin değiştirilmesi

Bu bölümde hatalı yapılan bir kaydın gerekli alanları değiştirilebilmektedir.

4 - Eserin aranması

Bu bölüm kütüphanecilikte karşılaşılan en önemli problem olan bir eserin aranması işlemini gerçekleştirmektedir. Eserlerin aranmasında tek veya iki key üzerinden olmak üzere 2 ayrı yöntem kullanılmaktadır. Tek Key ile arama işleminde

24 ayrı özellik üzerinden arama işlemi yapılabilir. Ve verilen özelliğe sahip tüm kitaplara ulaşmak mümkündür. (Tek Özelliğe Göre Aramada Seçilebilen Özellikler, Tablo 5)

Birden fazla özellik verilerek yapılan arama işlemine çapraz arama adı verilir. Bu yazılımda önceden belirlenen 14 ayrı key çifti üzerinden çapraz tarama yapılarak , verilen özelliğe sahip tüm kitaplar bulunur. (İki Özelliğe Göre Çapraz Aramada Seçilebilen Özellikler, Tablo 6)

5 - Çeşitli listelerin yazıcıdan alınması

Eserlerin çeşitli listeleri yazıcıdan çıkış olarak alınabilir

6 - Eserler üzerinde çeşitli sorgu ve listeler

Bu bölümde çeşitli keylere göre aynı özelliğe sahip eserler özet listeler halinde ekranda görülebilir. Bu bölümde 20 ayrı özellik üzerinden listeleme işlemi yapılabilir. (Sorgulama ve Listelemede Kullanılabilen Özellikler, Tablo 7)

7 - Eserlerin sıralanması

Bu bölümde ise eserler istenilen keye göre sıralanarak daha önce sırasız olarak girilen bilgilerin düzenlenmesi sağlanmaktadır.

İslam İlimleri Tasnif Cetveline Göre Konu Başlıkları	
UMUMİYAT	KUR'AN VE KUR'AN İLİMLERİ
TEFSİR İLMİ	HADİS İLMİ
AKAİD VE KELAM	FIKİH İLMİ
İSLAM MEZHEPLERİ	TASAVVUF
İÇTİMAİYAT VE AHLAK	İSLAM DİNİ TARİHİ
DİĞER KONULAR	

TABLO 1

Yazmaların Bulunduğu Koleksiyonlar	
ABDÜLGANİ AĞA	İZMİR
ALİ GALİP BEY	İZMİRLİ İSMAİL HAKKI bey
ALİ N. TARLAN	İZMİRLİ MUSTAFA EFENDİ
AMCA HÜSEYİN PAŞA	KADIZADE BURHANETTİN
ANTALYA TEKELİOĞLU	KADIZADE MEHMED
AŞİR EFENDİ	KARAÇELEBİZADE HÜSAMEDDİN
AYASOFYA	KASİDECİZADE SÜLEYMAN SIRRI
BAĞDATLI VEHBİ EFENDİ	KILIÇ ALİ PAŞA
BEŞİR AĞA	LALA İSMAİL
BEŞİR AĞA (EYÜB)	LALELİ
CARULLAH EFENDİ	MAHMUD PAŞA
CELAL ÖKTEN	MEHMET AĞA CAMİİ
ÇELEBİ ABDULLAH EFENDİ	M.HİLMİ-M.FEHMİ
ÇORLULU ALİ PAŞA	M.MURAD-M.ARİF
DAMAT İBRAHİM PAŞA	MESİH PAŞA
DARÜLMESNEVİ	MİHRİŞAH SULTAN
DENİZLİ	MOLLA ÇELEBİ

DÜĞÜMLÜ BABA	MUĞLA
EFGANİ ŞEYH HAYDAR EFENDİ	MURAD BUHARİ
EL-HAC MUSTAFA EFENDİ	MUSALLA MEDRESESİ
ERZİNCAN	NAFİZ PAŞA
ESAD EFENDİ	NUREDDİN AĞA
EYÜB CAMİİ (HZ.HALİT)	OSMAN HULDİ ÖZTÜRKLER
FATİH	PERTEV PAŞA (SELİMİYE)
DR.FEYZULLAH PAŞA	PERTEVNİYAL SULTAN
GELİBOLULU TAHİR EFENDİ	REİSÜLKÜTTAP MUSTAFA EFENDİ
GİRESUN	REŞİD EFENDİ
GÜLNUŞ VALDE SULTAN	RÜSTEM PAŞA
HACI MAHMUD EFENDİ	SALİHA HATUN
HACI REŞİD BEY	SEREZ
HAFİZ AHMED PAŞA	SERVİLİ
HAFİD EFENDİ	SEYYİD NAZİF EFENDİ
HAFİD EFENDİ EKİ	SULTAN AHMED I.
HALET EFENDİ	SÜLEYMANİYE
HALET EFENDİ EKİ	ŞAZELİ TEKKEŞİ
HAMİDİYE	ŞEHİD ALİ PAŞA
HARPUT	ŞEHZADE MEHMED
HASAN HAYRİ VE ABDULLAH EFENDİ	ŞEYHÜLİSLAM ESAD EFENDİ
HASAN HÜSNÜ PAŞA	TAHİR AĞA TEKKEŞİ
HASİB EFENDİ TEKKEŞİ	TIRNOVALI
HAŞİM PAŞA	TURHAN VALDE SULTAN
HEKİMOĞLU ALİ PAŞA	UŞŞAKİ TEKKEŞİ
HİDİV İSMAİL PAŞA	YAHYA TEVFİK EFENDİ
HÜSEYİN KAZIM BEY	YAZMA BAĞIŞLAR

HÜSREV PAŞA	YENİ CAMİ
IYD MEHMED EFENDİ	YENİ MEDRESE
İBRAHİM EFENDİ	YOZGAT
İHSAN MAHVİ BALKIR	YUSUF AĞA
İSMİHAN SULTAN	ZÜHDÜ BEY

TABLO 2

Yazma Eserlerde Kullanılan hat Tipleri	
KUFİ	İCAZET
MAGRİBİ	NESTALİK
ARAP HATTI	MUHAKKAT
SÜLÜS	REYHANİ
NESİH	TEVKİ
NESİH KIRMASI	DİVANİ
RİK'A	CELİ DİVANİ
TALİK	GUBARİ
SİYAKAT	DİĞER HAT ÇEŞİTLERİ

TABLO 3

Yazma Eser Kaydında Girişi Yapılan Özellikler		
DEMİRBAŞ NO	* DİLİ	ÖLÇÜSÜ
TASNİF NO	MÜELLİF HATTI	EKSİK SAYFA
CİLT KAYDI	MÜTERCİMİ	YAPRAK ADEDİ
MÜELLİF ADI	* HATTI	SATIR SAYISI
TELİF TARİHİ	* KÜTÜPHANESİ	SÜTUN SAYISI
ESERİN MEŞHUR ADI	* KOLEKSİYONU	CİLT ADEDİ
ESERİN ADI	* KONUSU	KOPYA ADEDİ
İSTİNSAH TARİHİ	FOTOKOPİ NO	KAĞIT ÖZELLİĞİ

MÜSTENSİHLERİ	MİKROFİLM NO	MÜZEHHİBİN ADI
TERCÜME TARİHİ	RİSALE YERİ	TEZHİPLİ SAYFA
MİNYATÜR ADEDİ	KIRAAT KAYDI	VAKIF KAYDI
MİNYATÜR ÖZELLİKLERİ	SİMA KAYDI	RESİM PLAN VS.ÖZ.
MİNYATÜRLÜ SAYFALAR	RİVAYET KAYDI	KİTABIN MUHTEVASI
CİLT ÖZELLİĞİ	FEVAİD	DİĞER AÇIKLAMALAR
HATTATI	MÜTALEA KAYDI	
HAŞİYE YAZAN(ŞARİH)	MÜTALEA TARİHİ	
TALİKAT YAZAN	TEMELLÜK TARİHİ	
İHTİSAR EDEN	MUKABELE TARİHİ	
ZEYL YAZAN	MÜLKİYE KAYDI	
ŞERH YAZAN	VAKIF KAYDI TARİHİ	

TABLO 4

Tek Özelliğe Göre Aramada Seçilebilen Özellikler		
DEMİRBAŞ NO	İKİ TARİH ARASI ESERLER	TEZHİP ÇEŞİDİ
TASNİF NO	İSTİNSAH YERİ	RESİM, PLAN, HARİTALI
ESERİN ADI	DİLİ	TEK NÜSHA ESERLER
ESERİN MEŞHUR ADI	HATTAT ADI	CİLT ÇEŞİDİ
KOLEKSİYONU	HATTIN CİNSİ	MÜELLİF HATTI
KÜTÜPHANESİ	TERCÜME TARİHİ	HAŞİYE YAZARI(ŞARİH)
MÜELLİF ADI	MÜTERCİM ADI	ZEYLLERİ
TELİF TARİHİ	MİNYATÜRLÜ ESERLER	ŞERHLERİ

TABLO 5

İki Özelliğe Göre Aramada Seçilebilen Özellikler	
ESER ADI & MÜELLİF ADI	KOLEKSİYON ADI & DEMİRBAŞ NO

ESERİN ADI & ESERİN DİLİ YERİ	İKİ TARİH ARASI & İSTİNSAH
ESERİN ADI & SATIR SAYISI	İKİ TARİH ARASI & TASNİF NO
TASNİF NO & YAZILDIĞI YER	İKİ TARİH ARASI & HATTIN CİNSİ
TASNİF NO & MÜELLİF ADI	İKİ TARİH ARASI & TEZHİP.ÖZ.
HATTAT & HATTIN CİNSİ	İKİ TARİH ARASI & HATTAT ADI
KOLEKSİYON & KÜTÜPHANE	İKİ TARİH ARASI & CİLD ÖZ.

TABLO 6

Eserler Üzerinde Çeşitli Sorgu Ve Listeler	
DEMİRBAŞ NUMARASI	DİLİ
TASNİF NUMARASI	HATTAT ADI
ESERİN ADI	HATTIN CİNSİ
ESERİN MEŞHUR ADI	TERCÜME TARİHİ
KOLEKSİYONU	MÜTERCİM ADI
KÜTÜPHANESİ	MİNYATÜRLÜ ESERLER
MÜELLİF ADI	TEZHİP ÇEŞİDİ
TELİF TARİHİ	RESİM, PLAN, HARİTALI
İKİ TARİH ARASI ESERLER	TEK NÜSHA ESERLER
İSTİNSAH YERİ	CİLD ÇEŞİDİ

TABLO 7

4 - VERİ TABANI UYGULAMALARINDA KULLANILAN BAZI YAZILIMLAR

Aktif bellek değişkeni sayısı :Maximum 256

4.1. dBASE III PLUS

5000 byte kullanılabilir.

Karakter tipi bellek değişkenleri : string uzunluğa + 2 byte

4.1.1. dBASE III PLUS'ın Teknik Özellikleri

Date tipi bellek değişkenleri : 9 byte

dBASE III PLUS mikrobilgisayarlar için ASHTON-TATE tarafından geliştirilmiş bir veritabanı yazılımıdır. IBM PC uyumlu bilgisayarlarda, DOS işletim sistemi altında çalışır. Gereken minimum bellek 256K olmasına rağmen genelde 384K RAM bellek kullanır. Aşağıdaki teknik özellikler kullanılan RAM belleğe göre değişebilir.

Veritabanı dosyaları .DBF olarak veritabanı

dosyalarında saklanır. Bu dosyalar RECORD ve FIELD lardan

Record Sayısı : Maximum 1 Billion

Byte Sayısı : Maximum 2 Billion

Record uzunluğu : Maximum 4000 byte

Field sayısı : Maximum 128

Field Özellikleri

Karakter : Maximum 254 byte

Date : Maximum 8 byte

Logical : Maximum 1 byte

Memo : Maximum 5000 byte

Numeric : Maximum 19 byte

Dosyalama İşlemleri

Aynı anda her tipten 15 dosya açık olabilir.

Aynı anda 10 veritabanı dosyası kullanılabilir.

Her aktif veritabanı dosyası için 15 ayrı Index dosyası açılabilir.

Her aktif veritabanı dosyası için bir format dosyası aktiftir.

Nümerik Hassasiyet :15.9 basamaktır.

Karşılaştırmalarda ise 13 basamaktır.

Bellek Değişkenleri

Aktif bellek değişkeni sayısı :Maximum 256

Bellek değişkenleri : 6000 byte kullanılabilir.

Karakter tipi bellek değişkenleri : string uzunluğu + 2 byte

Nümerik bellek değişkenleri : 9 byte

Date tipi bellek değişkenleri : 9 byte

Lojik bellek değişkenleri : 2 byte

Komut satırı uzunluğu : Maximum 254 karakter

4.1.2. dBASE III PLUS Dosya yapısı,Veri ve Dosya Tipleri

dBASE III PLUS'ta veri;tipi .DBF olan veritabanı dosyalarında saklanır. Bu dosyalar RECORD ve FIELD lardan oluşur. Bir veritabanı dosyasına yapılan her bir giriş bir RECORD'u, bu recordu oluşturan her birim ise bir FIELD'i oluşturur. Başka bir deyişle dosyayı oluşturan her bir satır bir RECORD (KAYIT),her sütun ise bir FIELD (ALAN) dır.

FIELD1	FIELD2	FIELD3		FIELDN

RECORD 1

RECORD 2

RECORD N

DOSYA (FILE) YAPISI

ŞEKİL 6

Bir veritabanı dosyası oluştururken önce bu dosyayı oluşturacak FIELD'lara, FIELD isimlerine, tiplerine ve uzunluklarına karar verilerek dosya yapısı oluşturulur.dBASE III PLUS'ta; Character, Numeric, Logical, Date ve memo olmak üzere 5 tip alan vardır.

Character alanları : Tüm alfabetik karakterler, noktalama işaretleri ve hesaplamada kullanılmayacak nümerik değerler bu tip alanlarda saklanır.

Nümerik alanlar : Bu tür alanlarda integer ve real sayılar saklanır.

Date alanları : Tarih bilgilerinin saklanması için kullanılır. Bu tür bilgiler AA/GG/YY formatında saklanmaktadır.

Logical alanlar : Doğru/Yanlış (True/False) türü bilgilerin saklandığı mantıksal alanlardır.

Memo : Bir text editorde yazılmış, büyük miktarda text bilgisinin saklanması için kullanılır.

dBASE III PLUS'ta veritabanı dosyasının uzunluğu;

Dosya Uzunluğu = File Header + Record Sayısı * Record Uzunluğu + 1

File Header Uzunluğu = (32 * Field sayısı) + 34

formülleri ile hesaplanır. Bir field'in içeriği ne olursa olsun kendisine ayrılan karakter sayısı kadar yer kaplar. Yani 15 karakter ayırdığımız bir fieldın sadece 1 karakterini kullansak bile bellekte 15 karakterlik yer tutar. Bu yüzden kullanılmayan alanlar disk üzerinde boşuna yer harcamaktadır. dBASE III PLUS'ta Binary, Catalog, Command, Database, Format, Index, Label, Memory, Query, Report, Screen, Text ve View olmak üzere 14 tür dosya kullanılabilir.

Binary (.BIN) Dosyaları : Link edilmiş ve binary forma dönüştürülmüş Assembler programlarıdır. CALL komutuyla çağrılarak kullanılır.

Catalog (.CAT) Dosyaları : Birbiriyle ilişkili dosyaları bir arada tutmak-görebilmek amacıyla oluşturulurlar.

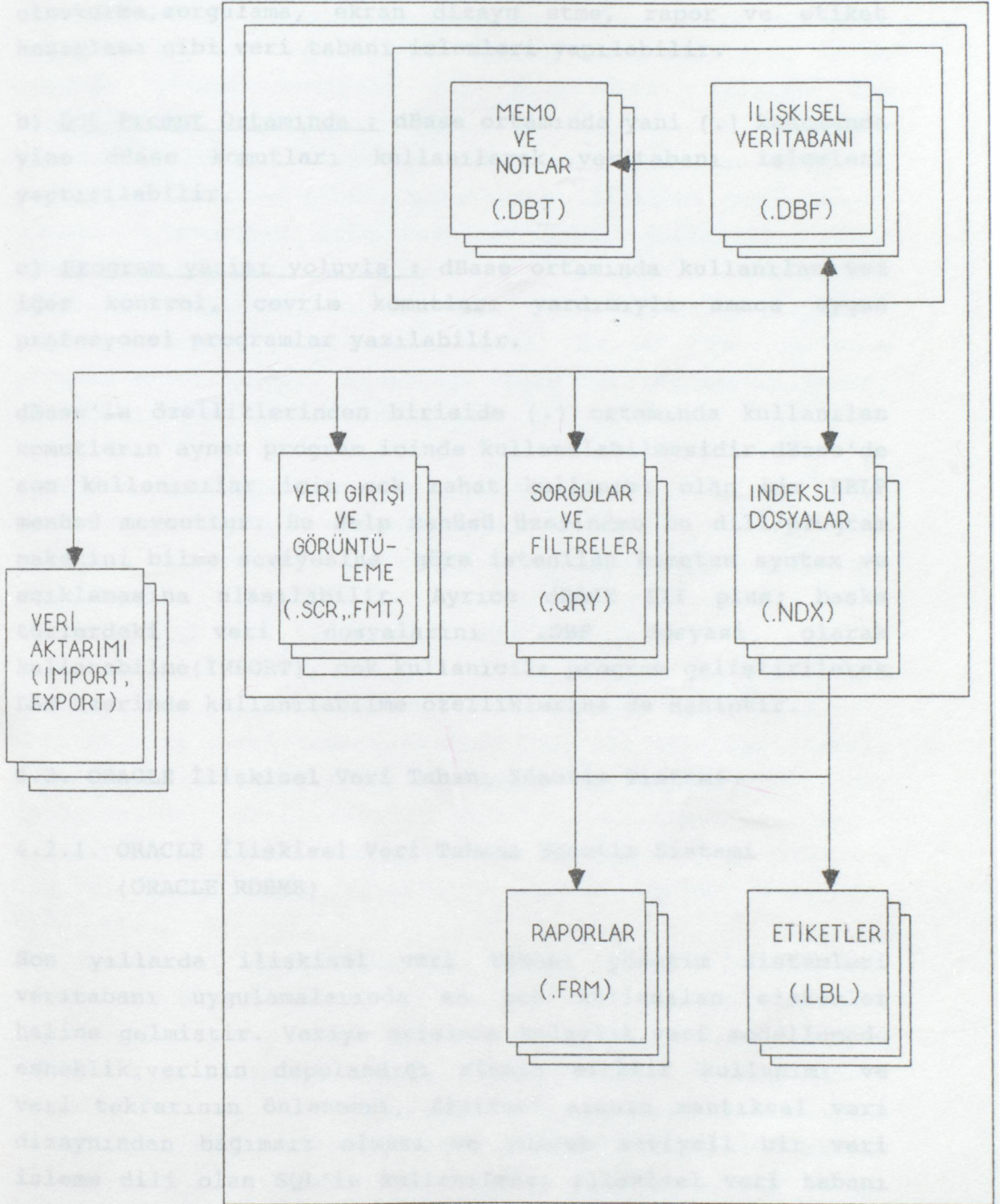
Command (.PRG) Dosyaları : dBASE komutlarından oluşan program dosyalarıdır.ASCII formatında dosyalardır.
Database (.DBF) Dosyaları: Veritabanı dosyalarıdır.
Format (.FMT) dosyaları : Ekran düzenlemek amacıyla .SCR dosyalarından oluşturulmuş dosyalardır.
Index (.NDX) dosyaları : .DBF türündeki veri tabanı dosyaları üzerinde mantıksal bir sıra(index) oluşturmak üzere kullanılan dosyalardır. .DBF dosyaları ile birlikte kullanılırlar.
Label (.LBL) dosyaları : Etiket hazırlamak üzere kullanılan dosyalardır.
Memory (.MEM) dosyaları : Bellek değişkenlerini saklamak üzere kullanılan dosyalardır.
Query (.QUE) dosyaları : Bir .DBF dosyası üzerinde sabit sorgulamalar yapmak amacıyla kullanılır.
Report (.FRM) dosyaları :Rapor hazırlama amacıyla kullanılır.
Screen (.SCR) dosyaları : Ekran düzenleme amacıyla oluşturulur.
Text (.TXT) dosyaları : ASCII formatında bilgilerin saklanması için kullanılır.
View (.VUE) dosyaları : Bir veritabanı dosyasından belli bazı alanların görülmesini sağlamak için kullanılan dosyalardır.

Sekil 7 de dbase III plus veritabanı yönetim sistemi yapısı görülmektedir.

4.1.3. dBASE III PLUS Kullanımı

dBase III PLUS 3 değişik yöntemle kullanılabilir.

a) ASSISTANCE menüsü yardımıyla : Daha çok programlamayı bilmeyen son kullanıcılar için geliştirilmiş bu menü yardımıyla menüler üzerinden seçim yaparak; dosya



dBASE III PLUS VERİ TABANI YAPISI

ŞEKİL 7

oluřturma,sorgulama, ekran dizayn etme, rapor ve etiket hazırlama gibi veri tabanı işlemleri yapılabilir.

b) Dot Prompt Ortamında : dBase ortamında yani (.) konumunda yine dBase komutları kullanılarak veritabanı işlemleri yaptırılabilir.

c) Program yazımı yoluyla : dBase ortamında kullanılan ve diğer kontrol, çevrim komutları yardımıyla amaca uygun profesyonel programlar yazılabilir.

dBase'in özelliklerinden biriside (.) ortamında kullanılan komutların aynen program içinde kullanılabilmesidir.dBase'de son kullanıcılar için çok rahat kullanımı olan bir HELP menüsü mevcuttur. Bu Help menüsü üzerinden bu dili/program paketini bilme seviyesine göre istenilen komutun syntax ve açıklamasına ulaşılabilir. Ayrıca dBASE III plus; başka türlerdeki veri dosyalarını .DBF dosyası olarak kullanabilme(IMPORT), çok kullanıcılı program geliştirilerek LAN üzerinde kullanılabilme özelliklerine de sahiptir.

4.2. ORACLE İlişkisel Veri Tabanı Yönetim Sistemi

4.2.1. ORACLE İlişkisel Veri Tabanı Yönetim Sistemi

(ORACLE RDBMS)

Son yıllarda ilişkisel veri tabanı yönetim sistemleri veritabanı uygulamalarında en çok kullanılan sistemler haline gelmiştir. Veriye erişimde kolaylık,veri modellemede esneklik,verinin depolandığı alanın efektif kullanımı ve veri tekrarının önlenmesi, fiziksel alanın mantıksal veri dizaynından bağımsız olması ve yüksek seviyeli bir veri işleme dili olan SQL'in kullanılması ilişkisel veri tabanı dillerinin getirdiği kolaylıklardır.ORACLE İlişkisel Veri Tabanı Yönetim Sistemi yukarıda belirtilen özelliklere sahip bir veri tabanı sistemidir.Temeli SQL dilidir. SQL dili, bu

tip uygulamalara yeni başlayanlar için veriye erişimde kolaylık ve hız sağlar. Aynı zamanda programcılarının gerek duyduğu yüksek seviyeli bir programlama dilinin tüm özelliklerine sahiptir. SQL ilk olarak IBM Research tarafından geliştirilmiş ve daha sonra American National Standards Institute (ANSI) tarafından ilişkisel veri tabanı yönetim sistemleri için standart kabul edilmiştir. ORACLE RDBMS; veri tabanı yönetim ve servislerinde veritabanı yönetici (Database Administrator, DBA) ve kullanıcılarına yardımcı olan programları, veri izleme ve kullanım programlarını içerir. Çekirdeği Kernel'dir. Kernel'in yerine getirdiği fonksiyonlar şunlardır.

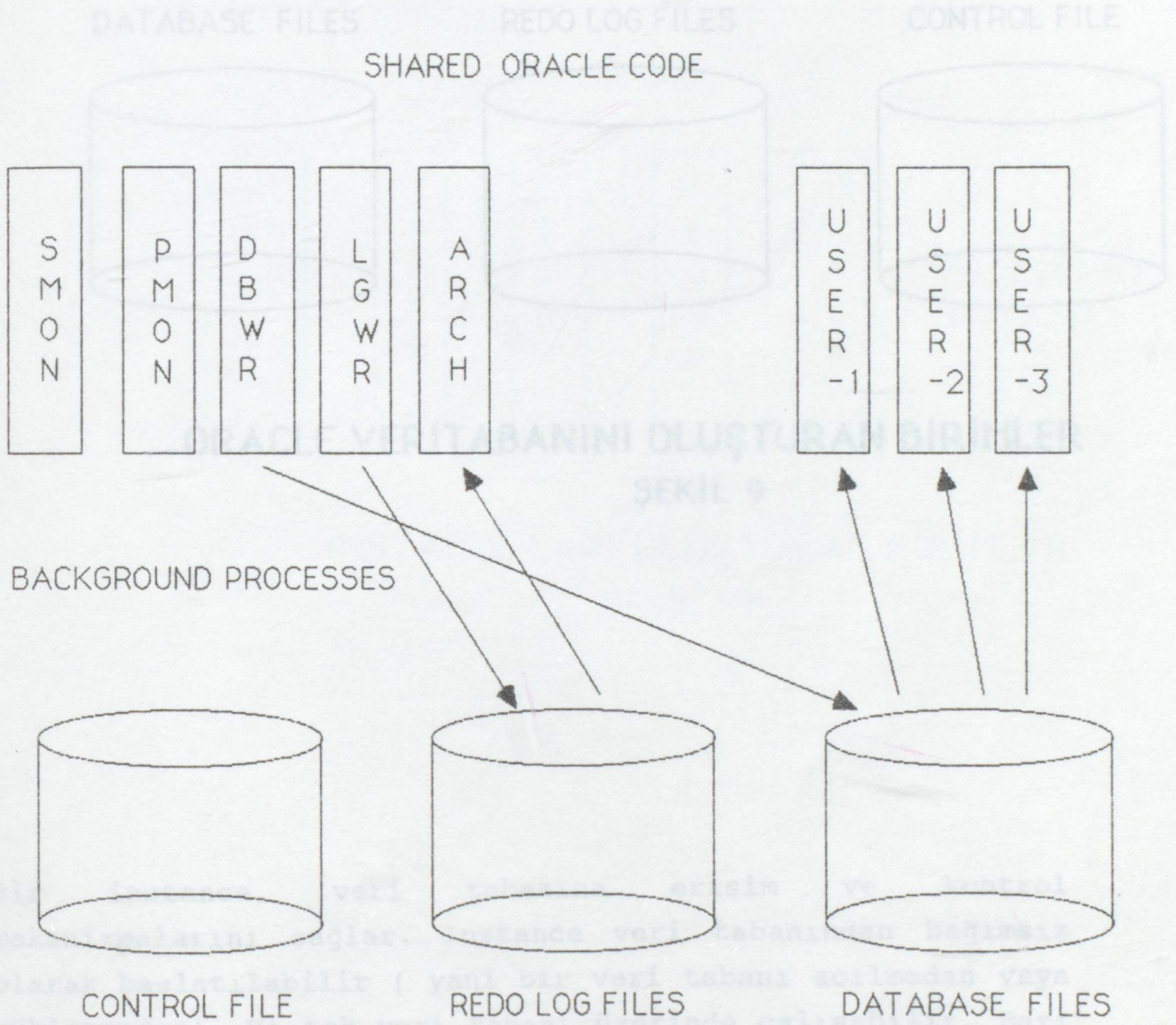
- Veri tanımlanması, yönetimi ve depolanması
- Veriye erişim ve çoklu çalışmanın kontrolü
- Backup ve Recovery imkanı- SQL ve PL/SQL çalışma imkanları

4.2.2. ORACLE Veri tabanı ve ORACLE Instance

ORACLE veri tabanı sisteminin temelinde bir kullanıcı, veritabanına (Veri tabanı ve Redo log file'leri) veri tabanı üzerinde işlem yapan yazılım (Instance) yoluyla erişir. bir Instance ve veri tabanı birbirinden bağımsız olarak çalışır. Ancak her instance sadece bir veri tabanına bağlıdır. Diğer veri tabanlarına erişim SQL*NET üzerinden olur. (Şekil 8)

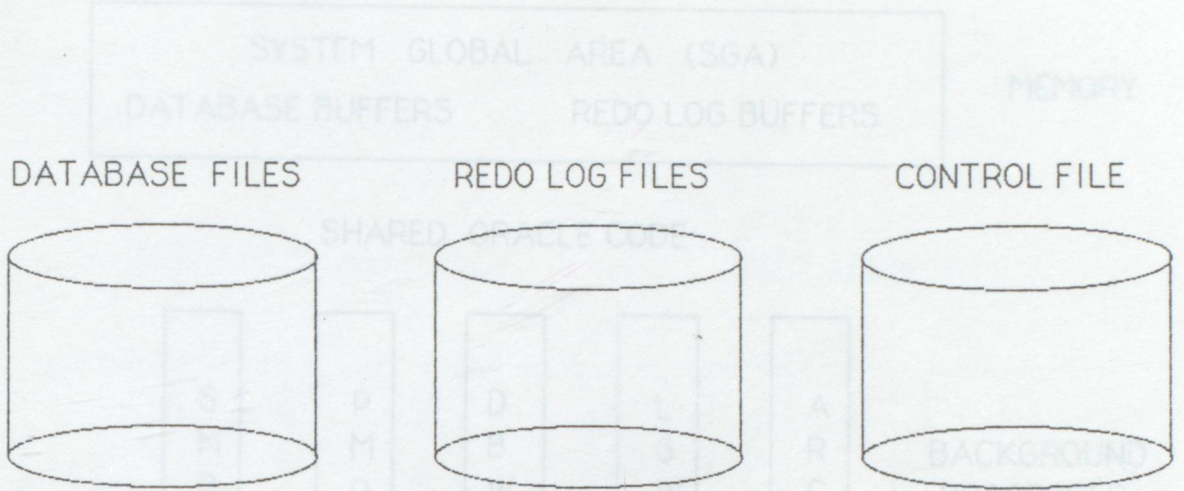
Instance'ların başlatılma ve durdurulması işlemi DBA tarafından SQL*DBA yazılımı ile yapılır. Veri tabanı UNIT adı verilen verilerden oluşur ve aynı zamanda işletim sistemi file'lerini da içerir. Fiziksel olarak veri tabanı file'leri ve redo log file'lerini kapsar. Mantıksal olarak; veri tabanı file'lerini, kullanıcı table'lerini, sözlükleri ve recovery bilgilerini içeren redo log file'lerini tutar. Aynı zamanda veri tabanını tanımlayan bilgileri tutan kontrol file'lerinin bir veya daha fazla sayıda kopyasını da içerir. (Şekil 9)

SYSTEM GLOBAL AREA (SGA)



ORACLE VERİTABANI SİSTEMİ YAPISI

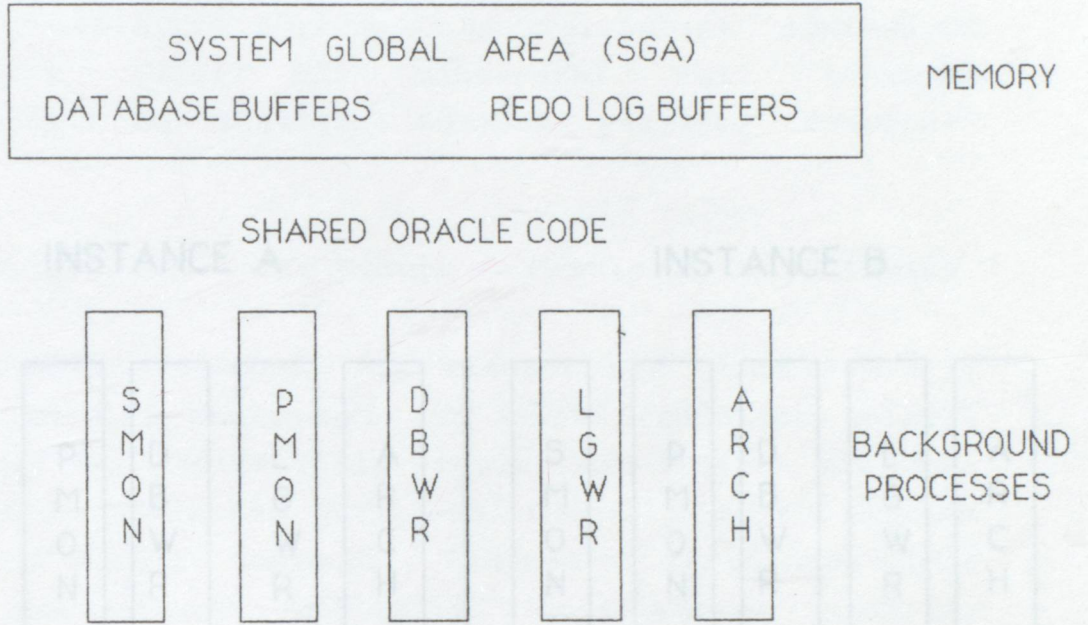
ŞEKİL 8



ORACLE VERİTABANINI OLUŞTURAN BİRİMLER
ŞEKİL 9

Bir instance, veri tabanına erişim ve kontrol mekanizmalarını sağlar. Instance veri tabanından bağımsız olarak başlatılabilir (yani bir veri tabanı açılmadan veya yüklenmeden). Ve tek veri tabanı üzerinde çalışabilir. Bazı işletim sistemleri bir veri tabanının birden fazla instance tarafından açılmasına imkan vermektedir. Bir instance; yapılan işlemler arasında iletişimi sağlayan paylaşılabılır tek bellek alanından (System Global Area, SGA) ve diğer kullanıcılar tarafından paylaşılan beş ayrı işlem den oluşur. (Database Writer, DBWR; Log Writer, LGWR; System Monitor, SMON; Process Monitor, PMON; Archiver, ARCH)

Her instance'ın , diğer instance'lar tarafından ortak kullanılan ORACLE RDBMS koduna erişebilmesi gereklidir.



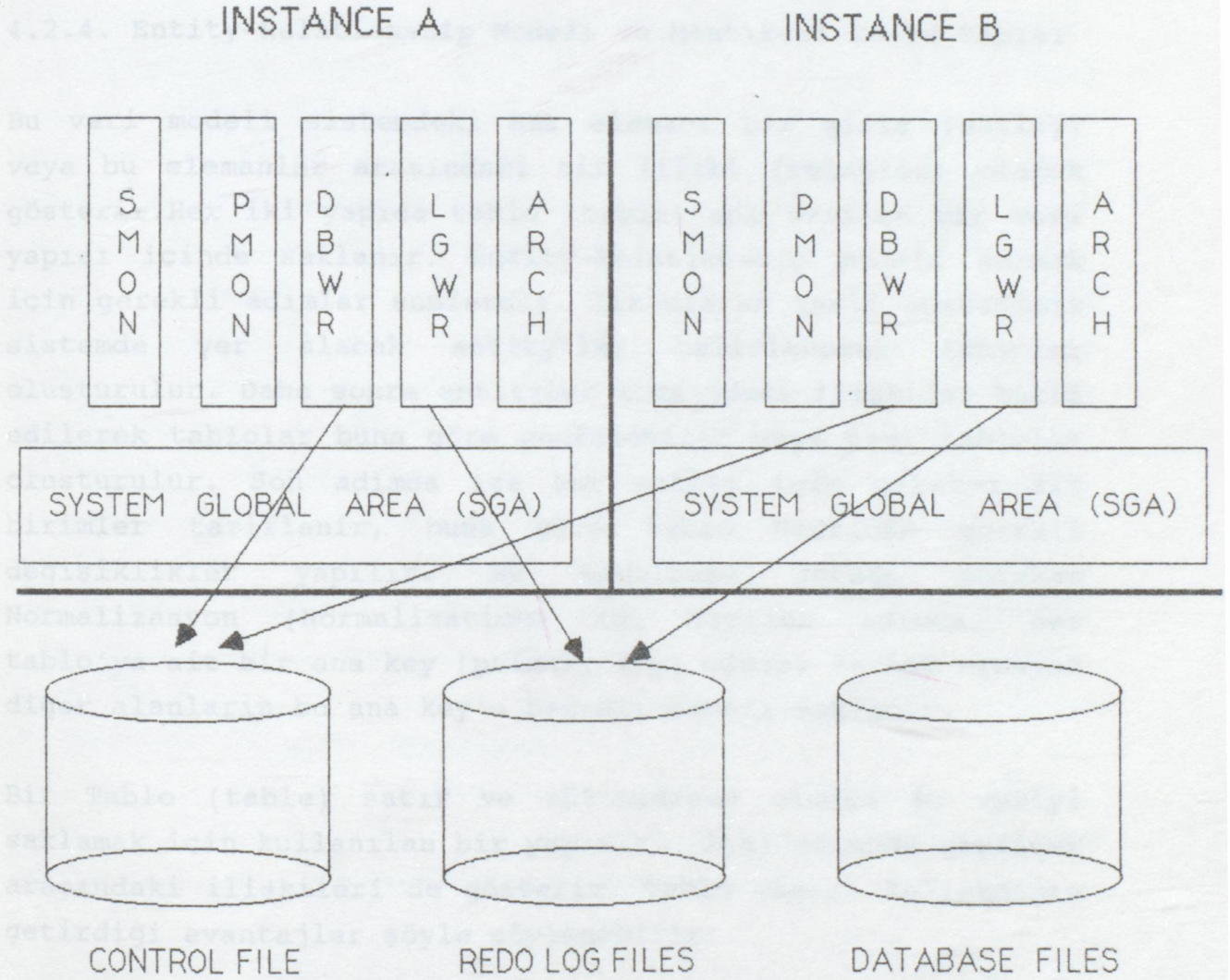
INSTANCE'LARI OLUŞTURAN BİRİMLER

ŞEKİL 10

4.2.3. Paylaşılabilir ve Dağıtılmış Veri Tabanı Sistemleri

Bir veri tabanına sadece bir instance tarafından erişilebilir. Ancak bazı işletim sistemleri birden fazla instance tarafından erişime imkan vermektedir. Eğer bir veri tabanına birden fazla instance tarafından erişilebiliyorsa Paylaşılabilir Disk Sistemi adını alır. Dağıtılmış veritabanları ise ağ yazılımı tarafından bağlanmış birden

çokveri tabanı 've instance'a ilişkin verilerdir. Bir ORACLE veri tabanı diğer veritabanı ile eş kurulumda iletişim kurabilir. Böylece bir veritabanı sisteminin kullanıcıları diğer bir sistemindeki veri tabanına ulaşabilirler. Bu iletişimi SQL*Net yapısı sağlar.



PAYLAŞILABİLİR VERİ TABANI YAPISI

ŞEKİL 11

çokveri tabanı ve instance'a imkan verir.(Sekil 11) Bir ORACLE veri tabanı diğer veritabanları ile ağ üzerinden iletişim kurabilir. Böylece bir veritabanı sisteminin kullanıcıları diğer bir sistemdeki veri tabanına ulaşabilirler. Bu iletişim SQL*Net yazılımı üzerinden yapılır.

4.2.4. Entity-Relationship Modeli ve Mantıksal Dosya Yapısı

Bu veri modeli sistemdeki her elemanı bir giriş (entity) veya bu elemanlar arasındaki bir ilişki (relation) olarak gösterir. Her iki yapıda tablo (table) adı verilen bir veri yapısı içinde saklanır. Entity-Relationship modeli kurmak için gerekli adımlar şunlardır. İlk olarak tarif edeceğimiz sistemde yer alacak entity'ler belirlenerek tablolar oluşturulur. Daha sonra entityler arasındaki ilişkiler tarif edilerek tablolar buna göre genişletilir veya yeni tablolar oluşturulur. Son adımda ise her entity için gereken alt birimler tariflenir, buna göre tablo üzerinde gerekli değişiklikler yapılır. Bu modelleme içinde gereken Normalizasyon (Normalization) adı verilen adımda, her tablo'ya ait bir ana key (primary key) olması ve key olmayan diğer alanların bu ana key'e bağımlı olması sağlanır.

Bir Tablo (table) satır ve sütunlardan oluşan ve veriyi saklamak için kullanılan bir yapıdır. Aynı zamanda girişler arasındaki ilişkileri de gösterir. Tablo yapısı kullanmanın getirdiği avantajlar şöyle söylenebilir:

Satır-Sütun formatındaki tablo, verilerin normal gösterimine de uygundur. Veri arasındaki ilişkileri göstermek için pointer kullanmak gerekmez. Veri zaten kendi üzerindeki ilişkileri gösterir. Kısaca veri çok kolay anlaşılabilir. Table yaratılırken table, sütun, index ve cluster'lar için anlamlı isimler kullanılmalı; modellenecek ilişkinin gerektirdiği kadar sütun oluşturarak fazla sütundan

kaçınılmalı; tablolarda aynı verilerin tekrarlanmamasına dikkat edilmeli; açıklama (Comment) satırlarında gereken kısaltmalar açıklanmalıdır. Bir tablo üzerinde her satır için ayrı bir değere sahip olan sütuna Key adı verilir (Primary Key). Bir tablo oluştururken key olacak sütun, 1. olarak tarif edilmelidir. Bazen birden fazla sütun birleşerek key oluşturabilirler (Composite Primary Key). Bir tableda yer alan her satır (row) ayrı değerler taşımalıdır. Bu da primary keyler ile sağlanır. İki veya daha fazla tablo arasında ilişkiyi sağlayan keylere ise Foreign key adı verilir. Viewlar bir table içindeki bilgilerin değişik şekilde görüntülenmesini sağlamak için kullanılır. Kendileri veri içermez ancak base table adı verilen, kullandıkları temel tablodaki verileri gösterirler. Bir table'a hızlı erişimi sağlamak için index yapıları kullanılır. Indexler aynı zamanda satırlar arasındaki tekrarları önlerler. Sorgulamalarda %10-%15 oranında hız kazandırırılar. Clusterlar ise bir veya daha fazla table üzerinde fiziksel olarak birbirine yakın recordlar tarafından oluşturulurlar. Bu özellikle sık erişilen record blokları için kullanılır. Ancak bir uygulamanın performansını arttırırken diğer bir uygulamanın performansını düşürmesi kaçınılmazdır. Oracle Query Optimizer'ları index ve clusterları gerektiği anlarda kullanarak yapılan uygulamanın performansını arttırırılar. Dolayısıyla bir table üzerinde index ve clusterlar oluşsa veya kaldırılrsa da yapılan uygulama programlarında değişiklik yapılmasına gerek yoktur.

4.2.5.ORACLE RDBMS Fiziksel Dosya Yapıları

ORACLE'ın kullandığı dosyalar Veri tabanı dosyaları, kontrol dosyaları,redo log dosyaları ve Oracle RDBMS kod dosyaları olmak üzere 4 grupta toplanabilir (Database Files, Control Files, Redo Log Files, ORACLE RDBMS code Files). Bu fiziksel dosyalarla veri tabanı yöneticileri ilgilenir. Diğer kullanıcıların bunlar üzerinde çalışmasına gerek yoktur. Ve

büyük kapsamlı veri tabanı uygulamalarında veritabanı yöneticisinin bunları göz önüne alarak uygulama yapması gerekir. Bir backup işlemi sırasında diğer kullanıcıların çalışmasına imkan verecek şekilde bir veri tabanı dizayni gibi. ORACLE Veritabanında bulunması gereken filelar şunlardır.

Veritabanı fileları : Max 1 tane, Minimum 500KB uzunluğunda
Online Redo Logları : Max 2 tane, Minimum 50KB uzunluğunda
Kontrol fileları : Max 1 tane, Uzunluğu yükleme sırasında belirlenir.

Program fileları : Sayısı değişebilir, Uzunlukları ortalama 17400KB dir.

Bir veritabanının fiziksel yapısı bu verilerin disk üzerinde yerleşimiyle ilgilidir. Genellikle veritabanı yöneticisi tarafından düzenlenir. Böylece bir tablespace'in veritabanı ihtiyaçlarını karşılayıp karşılamadığı, yeni tablespace ihtiyaçları belirlenmiş olur. Kullanıcılar genellikle giriş/cıkış performansını arttırmak için fileların fiziksel yapısını incelerler. Mantıksal yapı ile fiziksel yapı birbirinden çok farklıdır. Mesela kullanıcı tarafından tek bir table üzerinde görülen veriler ,fiziksel olarak birden fazla file halinde tutulabilir. Genelde kullanıcılar veriye daha yakın olması sebebiyle mantıksal yapı ile ilgilenirler.

ORACLE RDBMS Program Dosyaları: Bunlar değişik tiplerdeki veri dosyaları, kütüphaneler, program ve executable filelardır. Genelde DBA veya sistem programcısının sorumluluğu altındadır. ORACLE yazılımının yüklenmesi ve upgrade edilmesi ile ilgili işlemlerin bulunduğu program filelarıdır.

Veritabanı Dosyaları : Bir ORACLE veritabanı bir veya daha fazla veritabanı dosyasından oluşabilir. Bunlar veritabanındaki verileri saklamak amacıyla kullanılır.

Özellikleri şöyle sıralanabilir. Veritabanındaki her file sadece bir veritabanına ait olmalıdır. Kullanılan tablolar tablespace adı verilen birimlerde toplu olarak saklanırlar. Bir instance başlatıldığı zaman ilgili veritabanı açık olmalıdır. Bir veritabanı disk üzerinde fiziksel olarak sürekli bir bölgede yer alıyorsa bu veritabanı daha yüksek bir performansa sahiptir. Bir veritabanı fileinin büyüklüğü sonradan değiştirilemez. Bir veritabanındaki ilk veritabanı file'i SYSTEM Tablespace'inde bulunur. Yapılacak uygulamalar için yeni tablespaces yaratmak daha uygundur. Oluşturulacak veritabanında saklanacak maximum dosya sayısı bu veritabanı oluşturulurken belirlenir. Maximum limiti 255 dir. ORACLE veritabanı BLOK veya PAGE adı verilen birimlere ayrılmıştır. Bunlar kullanılan işletim sisteminin blok uzunluğundan farklı olabilir. Bu koşulda işletim sistemi blok uzunluğunun bir çarpanı olmalıdır. Eğer işletim sistemi ve oracle blok uzunluğu aynı seçilirse performans daha yüksek olur.

Kontrol Dosyaları : ORACLE veritabanının her açılışında bir veya daha fazla kontrol file'ı kontrol edilir. Bir kontrol file'ı sadece bir veritabanına bağlıdır. Bu filelar bir instance veritabanına erişmek istediğinde veritabanı ile ilgili bilgileri sağlamak için tutulurlar. Fiziksel veritabanının ve redo logfilelerinin isimleri, veritabanı adı ve oluşturulma tarihi gibi bilgileri içerirler. Oracle tarafından otomatik olarak güncellenirler.

Redo Log Dosyaları : Bu dosyalar veritabanında bir değişiklik yapıldığında (transaction commit) bu değişikliği saklayan dosyalardır. Online ve offline olmak üzere iki gruba ayrılırlar. Online Redo dosyaları yapılan değişiklikleri sürekli kaydederler. Bu özellikle bilgilerin beklenmedik bir şekilde hasara uğraması durumunda yeniden yüklenebilmeleri için tasarlanmıştır. Off line redo log dosyaları ise online redo log dosyalarının arşivlenmek

amacıyla tutulan kopyalarıdır. Rolling Forward ve Rolling Back işlemleri ile Redo log Dosyaları da kullanılarak bir instance'in herhangi bir nedenle kesildiği durumlarda verilerin kaybolmaması-yeniden yüklenmesi sağlanır. Bu tüm veritabanı uygulamalarında gereken bir özelliktir.

4.2.6. ORACLE SQL

SQL daha öncede belirtildiği gibi 70'li yıllarda IBM araştırma laboratuvarları tarafından geliştirilmiş ve daha sonra Ekim 1986 da ANSI tarafından ilişkisel veri tabanı uygulamalarının standart dili olarak kabul edilmiştir. İlk kez 1979 da ORACLE tarafından ORACLE SQL'in ticari uygulamaları tanıtılmıştır. SQL tüm veri tabanı uygulamalarında sistem yöneticileri, veritabanı yöneticileri, uygulama programcıları, yönetim personeli, karar destek sistemi personeli ve diğer kullanıcıların istek ve ihtiyaçlarına cevap verebilen bir dildir. SQL bir defada bir record üzerinde değil bir defada bir record kümesi üzerinde işlem yapar ve veri üzerinde istenilen sorgulamanın yapılmasını sağlar. SQL komutlarının girişi bir record kümesi çıkışıda yine bir record kümesidir. Ayrıca çıkış olarak alınan bu recordlar bir başka SQL komutunda giriş olarak kullanılabilirler. SQL kullanırken sorgulama sırasında veriye erişimde ne tür bir erişim metodu kullanılacağına söylenmesine gerek yoktur. Çünkü tüm sorgulama komutlarında Query Optimizer kullanılır ve veriye erişimden hızlı yola karar verir. Query optimizer bir table üzerinde hangi keylerin index olarak tutulduğunu bilir ve gereken yerlerde bu indexleri kullanır. Eger yapılan uygulama index gerektirmezse kullanıcı index kullanma isteğini belirtse bile index kullanmaz. SQL çok kolay öğrenilen ve uygulanabilen bir veri tabanı dilidir. Temel komutlar bir kac saat içinde ayrıntılı komutlar ise birkaç günde tamamiyle öğrenilebilir. Ayrıca tüm ilişkisel veritabanı uygulamalarında kullanılabilecek standart dil

kabul edildiği için bir uygulamada kazanılan tecrübeler başka bir veritabanı uygulamasında da kullanılabilir. Yazılan programların bir sistemden diğerine aktarımı çok az bir değişiklikle sağlanabilir. ORACLE SQLin getirdiği bir yenilikte değişik veritabanı sistemleri arasında veri aktarımının (data migration) da sağlanmasıdır. Oracle precompilerları üçüncü nesil bilgisayar dilleri(3GL) ile SQL in birarada kullanılmasını sağlamaktadır. Bu precompilerlar PRO*ADA, PRO*C, PRO*COBOL, PRO*FORTRAN, PRO*Pascal, PRO*PL/I olarak sıralanabilir.

SQL; sorgulama; tablolara veri girişi; veri silme ve güncelleme; veritabanı objeleri oluşturma, silme ve değiştirme; veritabanına erişim kontrolü ve veriler arası uyumluluk gibi işlemleri yerine getirebilir. ORACLE ilişkisel veritabanı sisteminde kullanılan veritabanı objeleri şöyle sıralanabilir. Clusters, Columns, Constraints, Databases, DatabaseLinks, Indexes, Rollback segments, Save points, Sequences, Synonyms, Tables, table spaces, Users, Views. Tüm veritabanı objeleri veritabanı içinde bir yer tutar. Tablespace gibi bazı objeler kendi bünyelerinde Table gibi diğer objeleri tutabilirler. Her objenin bir yaratıcısı ve bir sahibi vardır (Creator ve Owner). Genelde yapılan veritabanı uygulamalarında bir objenin yaratıcısı ve sahibi aynı kişilerdir.

4.2.7. ORACLE SQL deki Veri Tipleri

CHAR, VARCHAR : Belirtilen uzunlukta karakter tipindeki veritanımlamaları için kullanılır. Maksimum uzunluk 255, default uzunluk ise 1 dir.

NUMBER : Numerik tipteki değişkenleri saklamak için kullanılırlar. 38 basamak hassasiyet sağlanmaktadır. Sabit veya floating point sayılar saklanabilir.

DATE : Tarih bilgileri saklamak için kullanılır. Bu tip değişkenlerin uzunluğu sabit ve 7 byte tir.

LONG : LONG olarak tarif edilen sütunlarda değişken uzunluktaki karakter verileri saklanır. Maximim uzunluk 65635 karakterdir.

RAW ve LONG RAW : Byte tipindeki verilerin saklanması için kullanılır.NULL değer taşıyan sütun, herhangi bir değer taşımayan sütun anlamına gelir. NULL değerler sıfır anlamına gelmez. Bir kolonun NOT NULL olarak tanımlanması o kolona bir değer girilmek zorunluluğunu gösterir. Veritabanında null sütunlar ancak içinde veri taşıyan sütunlar arasında kalmışlarsa saklanırlar. Bu koşulda saklanmaları için sadece 1 BYTE gerekir. Eger bir row'un sonuna düşmüşlerse saklanmazlar fakat belirtilirler. Çok kolona sahip olan tablelarda null deger taşıyan kolonlar en sonda tarif edilmelidir.

4.2.8. SQL deki Komut türleri

SQL deki komut türlerini 4 grupta toplamak mümkündür.
a) Sorgulama Komutları (Queries) Veriyi herhangi bir kombinasyon oluşturarak sorgulamak için kullanılan komutlardır. Sorgulamalar SQL'in SELECT komutu ile yapılır. Bu komutta görülmek istenen veri, sorgulanan tablo adı ve istenen koşul da belirtilir.

Örnek : SELECT NO,ADI FROM OGRENCI WHERE SINIF="1A"b)

Veri işleme Komutları (Data manipulation statements,DML) Bir tabloya yeni bir bilgi-satır eklemek, bir satır silmek veya olan bir bilgiyi değiştirmek amacıyla kullanılan komutlardır.

Ornek : INSERT INTO OGRENCI VALUES (333,'AYSE GUL','3B')
DELETE FROM OGRENCI WHERE ADI='AYSE GUL'C)

Veri tanımlama komutları(Data definition statements,DDL)
Yeni veritabanı objeleri tanımlamak , kullanılmayanları

silme ve objeler üzerinde deęişiklik yapmak için kullanılır. Tüm CREATE, DROP ve ALTER komutları bu tür komutlardır.

Ornek : CREATE TABLE OGRENCI

(NO NUMBER, ADI CHAR(10), SINIF CHAR(2))

DROP TABLE OGRENCI

d) Veri kontrol komutları (Data control statements, DCL) Bu tür komutlar veritabanına erişim ve veritabanındaki bilgiye erişim olmak üzere 2 tür erişimi kontrol ederler. Veritabanına erişim komutlarına örnek olarak GRANT CONNECT ve bilgiye erişim komutlarına örnek olarak ta GRANT SELECT veya REVOKE DELETE gibi komutlar verilebilir. Kısaca bu tür komutlar; bir kullanıcıya, diğer kullanıcıların tablolarını görme, deęiştirme ve kullanma yetkisi vermek için kullanılır.

4.2.9. ORACLE İlişkisel Veritabanı Yönetim Sistemindeki Yardımcı Programlar

ORACLE RDBMS yazılımında veritabanı işlemlerini kolaylaştırıp, hızlandırmak için bazı özel yazılımlarda hazırlanmıştır. Bunlar şöyle sıralanabilir.

SQL*DBA : Bu yazılım veritabanını başlatmak ve durdurmak, kullanımını gözlemek, backup ve recovery işlemlerini gerçekleştirmek kısaca bir veritabanı yöneticisinin görevlerinin yerine getirilmesini sağlamak amacıyla gerçekleştirilmiştir.

CRT : Full-Screen uygulamaları için gerekli gösterim özelliklerini ve keyboard tuş fonksiyonlarını oluşturmak-deęiştirmek için kullanılan yazılımlardır.

Export/Import : ORACLE verisini ORACLE dışına çıkarmak ve tekrar ORACLE'a yüklemek için yani işletim sistemi ile ORACLE arasında veritabanı transferi yapmak için kullanılır.

SQL*Loader : Değişik veritabanı formatındaki fileları ORACLE standart file'ı haline getirmek için kullanılır. Yani data migration işlemini gerçekleştirir.

SQL*Forms : Veritabanı üzerinde form tabanlı uygulamalar geliştirmek için kullanılan bir yazılımdır.

SQL*Menu : Veritabanı üzerinde menu tabanlı uygulamalar geliştirmek için kullanılan bir yazılımdır.

SQL*ReportWriter : Rapor veya listeler dizayn etmek için kullanılır.

PL/SQL : Standart SQL komutları geliştirilerek ve yeni komutlar eklenerek oluşturulmuş ve bu komutların kullanımını sağlayan bir yazılımdır.

ORACLE PreCompilers: Değişik üçüncü nesil dillerinin SQL ile birlikte kullanımını gerçekleştirmek için kullanılan Precompilerlardır. Bunlar rPRO*ADA, PRO*C, PRO*COBOL, PRO*FORTRAN, PRO*Pascal, PRO*PL/I olarak sıralanabilir.

5 - Süleymaniye Kütüphane Otomasyonu

Veritabanı uygulamalarında asıl amaç veriyi depolamak ve bu veri üzerindeki işlemleri optimum erişimlerle sağlamaktır. Bu aşamada sistem tasarımcılarının karşısına çıkan iki büyük problem vardır: Kullanılan yardımcı bellek cihazının kapasitesi ve büyük miktardaki veriye erişimde hız faktörü. Veritabanı yazılımı hazırlanırken yapılan sistem analizinde kurulacak sisteme uygun konfigürastona sahip bilgisayar sistemleri seçilir. Bu seçim sırasında verinin sistemde artış hızı gözönüne alınır ve yardımcı bellek birimini maksimum surede, efektif bir şekilde kullanabilmek için gereken önlemler alınır. Depolanan veriye erişimde de yine istekler ve ihtiyaçlar doğrultusunda indekslemek gibi özel yöntemler kullanılarak gereken hız sağlanır.

Şu anda Süleymaniye Kütüphanesinde kullanılan bilgisayar yazılımı dBase III Plus ile yapılmış ve daha sonra FoxBasePlus ile geliştirilmiştir. Sistem tek kullanıcı olup, yazma eserler dillerine göre gruplanarak 4 temel dosya üzerine oturan bir veritabanı kurulmuştur. Sorgulamalarda kullanılan key sayısı çok fazla olduğu ve aynı özelliğe sahip birden fazla kayıt bulunduğu için indeksleme yöntemine gidilmemiş ve sıralı arama yöntemi kullanılmıştır. Gerektiğinde dosyaların sıralanmasına imkan verilmiş ve bu işlemin belirli aralıklarla yapılması tavsiye edilmiştir. Yazılım da her kayıt için yani veritabanına kaydedilen her bir yazma eser için 2627 karakterlik yani 2627 bytelik yer ayrılmıştır.

Şu anda 30.000 yazma eserin kayıtlı olduğu veritabanında bu eserlere ait tüm bilgilerin girilmediği yani bazı alanların boş kaldığı görülmüştür. dBase türü dillerde veri yardımcı bellek/depolama birimi üzerinde saklanırken veri yapısı üzerinde değişken uzunlukta bilgilerin tanımlanmasına izin verilmez. Yani bir kayıt için 100 karakterlik bir yer

ayrılmış ve bu 100 karakterin sadece 1 karakteri kullanılmış olsa bile bu bilgi disk üzerinde 100 karakterlik veya 100 bytelik yer tutmaktadır. Bu da disk üzerindeki alanın verimsiz kullanımına sebep olmaktadır. Yazma eserlerin girişinde her kaydın 50-60'ını oluşturan bilgilerin girilmediği görülmüştür. Bu her eser için yaklaşık 1.5KB belleğin boşa harcanması demektir. Eğer gerçek verilerle hesaplama yaparsak 30.000 eser için harcanan bellek file header uzunlukları ile birlikte yaklaşık 78.8 MB tir. Ve bunun yaklaşık 39.4 MB'i bilgi girişi yapılmamış ve ileride de yapılmayacak olan boş alanlar için tutulmaktadır. Eğer kütüphanenin 125.000 adet tüm yazma eserlerinin kaydedileceği düşünülürse , mevcut sistem kullanılmaya devam edildiği takdirde 328 MB bellek alanı gerekecek ancak bunun 164MB'i boş olduğu halde kullanılmış gibi işlem görecektir. Ayrıca bu büyük miktarda kullanılmayan veri alanı Backup alınması sırasında da sorun yaratmaktadır. Bu saklanacak veri miktarı büyüdükçe karşı karşıya gelinen bu rproblem mevcut sistemi kısıtlayan problemlerden biridir.

Süleymaniye Kütüphanesi için tasarlanan sistemde 08A012 Bilgisayar üzerinde yapılan veritabanı uygulamalarının temel işlemi sorgulamadır. tek veya çok key üzerinden özellikleri belirtilen kayıt veya kayıt gruplarına kısa zamanda erişmek tüm bilgisayar kullanıcılarının ortak isteğidir. Aramada kullanılan tekli keyler ve çapraz key grupları çoğu kez önceden düzenlendiği için yazılımın kullanım aşamasında değişik key grupları isteklerine cevap veremezler. Oysa kullanıcılar key değiştirme esnekliğine sahip olmak isterler. Çoğu programlama dili bu isteğe cevap vermemmektedir. Nitekim Süleymaniye Kütüphanesi Yazılımında da tek key ile yapılan aramalarda 24 ayrı özellik, çapraz aramalarda 14 key çifti üzerinden sorgulama yapılabilir. Önceden belirlenmiş olan bu key çiftleri kullanıcının değişik key kombinasyonu isteklerine cevap verememektedir.

cluster veya non-cluster yapılarında olmasının sorgulama sonuçlarını etkilemesine gerek yoktur.

Bu yazılımda karşımıza çıkan üçüncü büyük problem hız problemidir. Özellikle fazla recorda sahip dosyalarda yapılan sorgulamalarda sonuç alınması çok uzun sürmektedir. Bu aramada kullanılan sıralı (Sequential) arama yönetiminin bir sonucudur. Kayıtlar dosyalara sırasız girildiği için arama seri olarak baştan sona yapılmaktadır. Ayrıca sıralama yapılmış dosyalarda da bunun için bir yöntem geliştirilmemiştir.

Kütüphanecilik yazılımlarının ortak özelliği çok kullanıcı olma gereğidir. Gerek bilgisayara veri girişi sırasında gerekse kullanım sırasında tek kullanıcı bir sistem yeteri kadar yararlı olmamaktadır. Çok kullanıcı bir sistemde aynı anda birden fazla operatörün bilgi girişi yapabilmesi giriş yapısını hızlandıracaktır. Aynı zamanda kütüphaneden yararlanmak isteyen araştırmacıların birden fazla terminal aracılığıyla veritabanına erişebilmeleri çok önemli bir konudur.

Süleymaniye Kütüphanesi için tasarlanan sistemde ORACLE İlişkisel Veritabanı Sistemi kullanılacaktır. Bu sistemin Çekirdeğini oluşturan SQL dili veritabanı işlemleri için çok kolay öğrenilebilen bir dildir. ORACLE yapısı gereği yapılan sorgulamalarda efektif yolu seçme özelliğine sahiptir. Yani bir tablonun indexi varsa ve girişi yapılan sorgulama komutu bu indexten yararlanmayı gerektirmiyorsa ORACLE RDMBS bu yolu seçerek en kısa zamanda erişimi sağlayacaktır. Ayrıca verilerin CLUSTER yapısı içinde saklanması erişimi hızlandıran başka bir özelliktir. Sık kullanılan ortak kolonlara sahip tablolar bir araya gelerek cluster yapısını oluştururlar. Birbiri ile ilişkili kolonların fiziksel olarak birbirlerine yakın yerde bulunması erişim hızını arttırır. Kullanıcı ve yapılan uygulama için şeffaf olan bu özellikler uygulama yazılımını etkilemez. Bir veri tabanının cluster veya non-cluster yapısında olmasının sorgulama komutlarını etkilemesine gerek yoktur.

ORACLE2da boş olan değerlerin saklanmaması gereken fazla miktarda bellek ihtiyacı için güzel bir çözümdür. Ayrıca verinin bellek blokları üzerindeki yerleşiminde gerektiğinde sıkıştırmalar yapılabilmektedir. Bu da uygulamalara hız ve bellek kazandırmaktadır. Veri tabanı üzerinde yapılan Insert, Delete ve Update işlemleri bir süre sonra bellek yerleşiminin süreksiz (delik-deşik) hale gelmesine sebep olur. Bu yüzden zincir yapısında saklanmış verilere erişmek için birden fazla bloğun okunup yazılması uygulama performansını düşürür. ancak belirli zamanlarda yapılan sıkıştırma işlemi ile veritabanına gereken verim kazandırılır.

ORACLE uygulama programları kullanılarak yapılan sorgulamalarda key sayısı değişkendir, hangi keylerin seçileceği kullanıcıya kalmıştır. Bu erişim sırasında verilen yetkilerle programı kullanan kişilerin değişik sınırlar içinde kalması sağlanmaktadır. Tüm kullanıcılar aynı uygulamayı kullansa bile bazılarının sadece erişim diğerlerinin güncelleme yetkisine sahip olmaları mümkündür. Bu yetkilerle kullanıcının veritabanına erişiminde gerekli güvenlik önlemleri alınmaktadır.

ORACLE RDBMS'teki Commit ve Rollback özellikleri sayesinde veritabanı işlemlerinin beklenmeyen şekillerde durmasına karşın gereken önlemler alınmaktadır. Yapılan değişiklikler sistem elde olmayan bir sebeple kapansada (power-off) saklanmakta tekrar normal konuma döndüğünde yüklenmektedir. Bu özellik birçok veritabanı yazılımında görülmez. Ayrıca Üçüncü Nesil Dilleri ORACLE SQL ile entegre edilerek kullanılabilir. Bu özellik hız ve grafik-veritabanı kombinasyonu gerektiren uygulamalar için önemli bir özelliktir. Oracle'ın çeşitli precompilerları ile 3GL bir dille yazılan programlar SQL'e entegre edilebilmektedir.

3GL dillerle yazılan programların veritabanı ile ilişkili olarak kullanılması son yılların önemli bir gelişmesinide veritabanı uygulamalarına katmamızı sağlar.Object Oriented Programlama.

OOP yöntemi başkaları tarafından kolayca anlaşılacak, akıllı programlar yazmamıza yardımcı olan bir yöntemdir. Bir veri tabanı uygulamasından işletim sistemi yazılımına kadar pekçok alanda kullanılabilir. Bu yöntemde problemin gerçek dünyadaki gösterimi oluşturulur ve bu gösterim yazılımla gerçekleştirilir. Öncelikle bir uygulamada kullanılması gereken objeler oluşturulur bu objeler veri yapılarını ve bu veri yapıları üzerindeki işlemleri bir arada toplarlar. Örnek olarak bir sözlüğü ele alalım. Sözlük Objesinin oluşturulmasında özel veri yapısı ve işlemler kullanılacaktır. Bir sözlük için gereken veri yapısı kelime, telaffuzu ve tanımlarını içerir.Üzerinde gerçekleşecek işlemler ise KELİME_EKLE, KELİME_SİL, KELİME_BUL, ANLAMINI_YAZ gibi işlemlerdir. OOP Dillerinde aynı tip işlemleri gerçekleştiren objeler bir araya toplanarak SINIFlar (CLASS) oluşturur. Bu sınıflar aynı özellikleri ve kendilerini karakterize eden özellikleri gösteren objelerde oluşur. Yine sözlük örneğini ele alırsak sözlük KİTAP sınıfının bir üyesidir Tüm kitaplar gibi yazarı, basımyeri, basım yılı gibi özelliklere sahiptir ve ayrıca yukarıda belirtilen karakteristik özelliklere de sahiptir. OOP yönteminde, kısaca, problem birim olarak bağımsız fakat topluca çözümü meydana getiren parçalara ayrılır. Bu parçalar bilgisayardaki objelere karşılık gelir. Veri sistem içinde açık olarak akmaz, sistemde veriden çok mesajlar görülür. Bu mesajların gerektirdiği işlemler ise objelerin metodları yoluyla gerçekleştirilir.(Şekil 12-13).

Veri tabanı teknolojisinin gerçek dünyadaki tüm uygulamalara yayılmasıyla bu alanda yeni yöntemler kullanma zorunluluğu ortaya çıkmıştır. Mesela üretim alanında 3 boyutlu geometrik

BİLGİSAYARDA
MODELLENCEK
GERÇEK OBJE



SÖZLÜK

OBJECT: SÖZLÜK

MESAJLAR

KELİME-BUL (ARANAN)
ANLAMINI-YAZ (ARANAN)

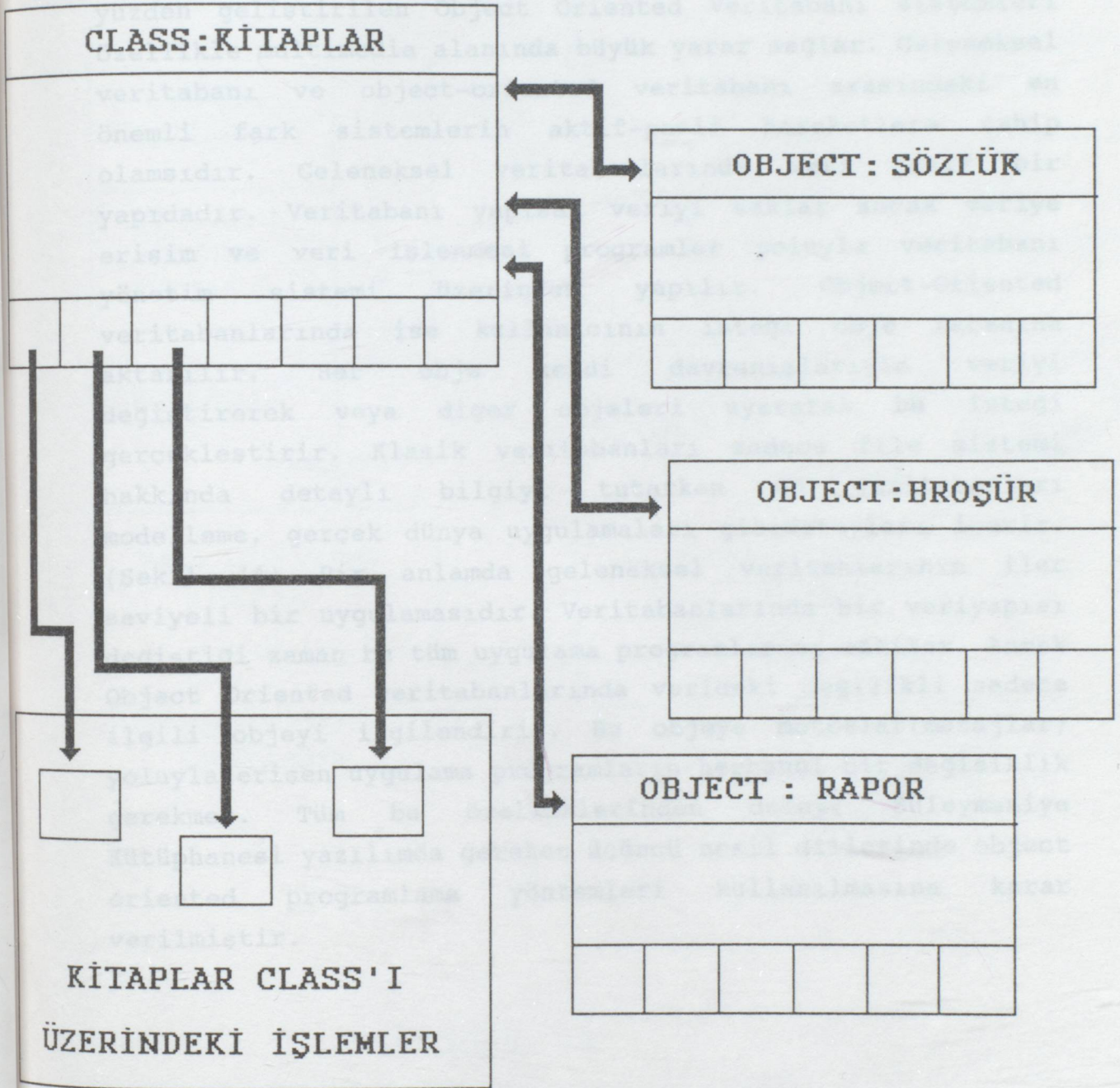
ARABİRİM
SÖZLÜK VERİ YAPISI
İŞLEMLER
KELİME-EKLE
KELİME-SİL
KELİME-BUL
ANLAMINI-YAZ
⋮

GERÇEK OBJENİN YAZILIM MODELİ

GERÇEK BİR OBJENİN
YAZILIM İÇİN MODELLENMESİ

ŞEKİL

12



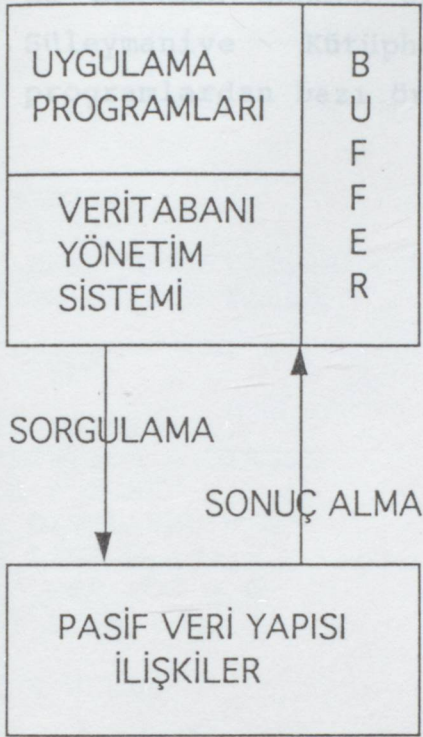
OBJELER, SINIFLAR ve ÖZELLİK AKTARIMI

ŞEKİL 2 13

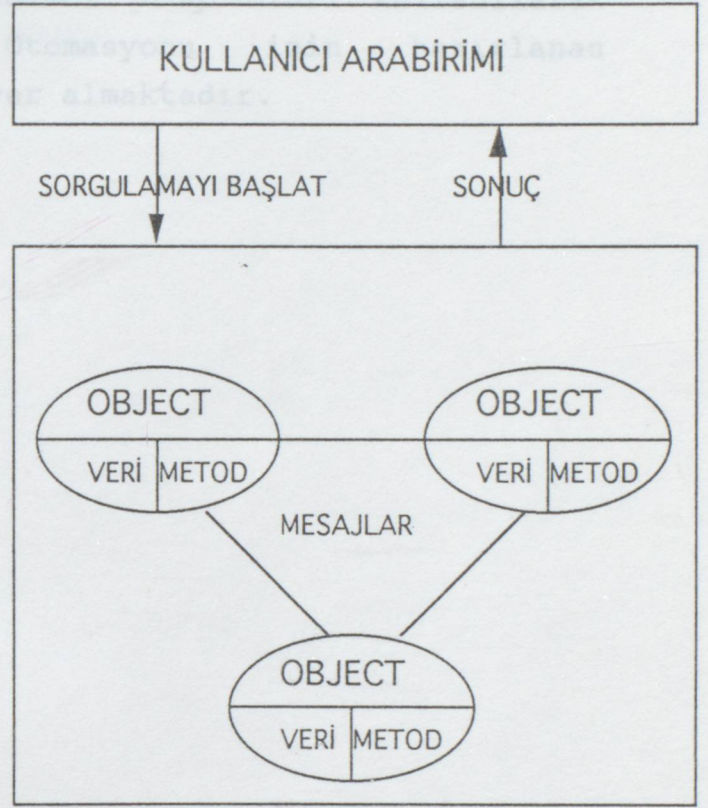
modellere çalışma ihtiyacı geleneksel veritabanı sistemlerinde kolayca sağlanamayacak bir problemdir. Bu yüzden geliştirilen Object Oriented Veritabanı sistemleri özellikle multimedia alanında büyük yarar sağlar. Geleneksel veritabanı ve object-oriented veritabanı arasındaki en önemli fark sistemlerin aktif-pasif hareketlere sahip olamsıdır. Geleneksel veritabanlarında veri pasif bir yapıdadır. Veritabanı yapısal veriyi saklar ancak veriye erişim ve veri işlenmesi programlar yoluyla veritabanı yönetim sistemi üzerinden yapılır. Object-Oriented veritabanlarında ise kullanıcının isteği obje tabanına aktarılır. Her obje kendi davranışlarıyla veriyi değiştirerek veya diğer objeleri uyararak bu isteği gerçekleştirir. Klasik veritabanları sadece file sistemi hakkında detaylı bilgiyi tutarken OO veritabanları modelleme, gerçek dünya uygulamaları gibidetayları içerir. (Şekil 14) Bir anlamda geleneksel veritabanlarının iler seviyeli bir uygulamasıdır. Veritabanlarında bir veriyapısı değiştiği zaman bu tüm uygulama programlarını etkiler. Ancak Object Oriented veritabanlarında verideki değişikli sadece ilgili objeyi ilgilendirir. Bu objeye metodlar(mesajlar) yoluyla erişen uygulama programların herhangi bir değişiklik gerekmez. Tüm bu özelliklerinden dolayı Süleymaniye Kütüphanesi yazılımda gereken üçüncü nesil dillerinde object oriented programlama yöntemleri kullanılmasına karar verilmiştir.

ŞEKİL 14

6. PROGRAMLAR



GELENEKSEL
VERİTABANI



OBJECT-ORIENTED
VERİTABANI

ŞEKİL 14

FORMS VERSION = 3.0.16.4.1

6. PROGRAMLAR

Bu bölümde ORACLE RDBMS yardımcı programları kullanılarak Süleymaniye Kütüphanesi Otomasyonu için hazırlanan programlardan bazı örnekler yer almaktadır.

```
NAME = GIRISI
TITLE = GIRISI
DEFAULT_MENU_APPLICATION = DEFAULT
VALIDATION_UNIT = FIELD
```

FINE BLOCK

```
NAME = GIRISI
DESCRIPTION = GIRISI
TABLES = TURK
ROWS_DISPLAYED = 1
PAGE_LINE = 1
LINES_PER_ROW = 0
ARRAY_SIZE = 0
```

DEFINE FIELD

```
NAME = DEMIRBASNO
DATATYPE = CHAR
LENGTH = 10
DISPLAY_LENGTH = 10
QUERY_LENGTH = 10
PAGE = 1
LINE = 3
COLUMN = 10
HELP = Enter value for : DEMIRBASNO
```

ENDDEFINE FIELD

DEFINE FIELD

```
NAME = TASNIFNO
DATATYPE = CHAR
LENGTH = 7
DISPLAY_LENGTH = 7
QUERY_LENGTH = 7
PAGE = 1
LINE = 3
COLUMN = 42
HELP = Enter value for : TASNIFNO
LOV_TEXT = <<<
SELECT DEWEX INTO GIRISI.TASNIFNO FROM KONULAR
>>>
```

ENDDEFINE FIELD

DEFINE FIELD

```
NAME = MLLINAYDI
```


*FORMS_VERSION = 3.0.16.4.1

SE = ON

LINE FORM

COMMENT = <<<

AMAC : TURKCE YAZMA ESER GIRISI YAPAR.

TABLE ADI : TURK

YAZAN : ZEKIYE EMIROGLU

>>>

NAME = GIRIS1

TITLE = GIRIS1

DEFAULT MENU_APPLICATION = DEFAULT

VALIDATION_UNIT = FIELD

DEFINE BLOCK

NAME = GIRIS1

DESCRIPTION = GIRIS1

TABLE = TURK

ROWS_DISPLAYED = 1

BASE_LINE = 1

LINES_PER_ROW = 0

ARRAY_SIZE = 0

DEFINE FIELD

NAME = DEMIRBASNO

DATATYPE = CHAR

LENGTH = 10

DISPLAY_LENGTH = 10

QUERY_LENGTH = 10

PAGE = 1

LINE = 3

COLUMN = 18

HELP = Enter value for : DEMIRBASNO

ENDDEFINE FIELD

DEFINE FIELD

NAME = TASNIFNO

DATATYPE = CHAR

LENGTH = 7

DISPLAY_LENGTH = 7

QUERY_LENGTH = 7

PAGE = 1

LINE = 3

COLUMN = 42

HELP = Enter value for : TASNIFNO

LOV_TEXT = <<<

SELECT DEWEY INTO GIRIS1.TASNIFNO FROM KONULAR

>>>

ENDDEFINE FIELD

DEFINE FIELD

NAME = CILTKAYDI

DATATYPE = CHAR
LENGTH = 15
DISPLAY_LENGTH = 15
QUERY_LENGTH = 15
PAGE = 1
LINE = 3
COLUMN = 64
HELP = Enter value for : CILTKAYDI

ENDDEFINE FIELD

DEFINE FIELD

NAME = MUELLIFAD
DATATYPE = CHAR
LENGTH = 80
DISPLAY_LENGTH = 30
QUERY_LENGTH = 80
PAGE = 1
LINE = 5
COLUMN = 18
HELP = Enter value for : MUELLIFAD

ENDDEFINE FIELD

DEFINE FIELD

NAME = TELIFTAR
DATATYPE = CHAR
LENGTH = 20
DISPLAY_LENGTH = 20
QUERY_LENGTH = 20
PAGE = 1
LINE = 5
COLUMN = 59
HELP = Enter value for : TELIFTAR

ENDDEFINE FIELD

DEFINE FIELD

NAME = ESMESHURAD
DATATYPE = CHAR
LENGTH = 30
DISPLAY_LENGTH = 30
QUERY_LENGTH = 30
PAGE = 1
LINE = 7
COLUMN = 24
HELP = Enter value for : ESMESHURAD

ENDDEFINE FIELD

DEFINE FIELD

NAME = ESERADI
DATATYPE = CHAR
LENGTH = 140
DISPLAY_LENGTH = 60
QUERY_LENGTH = 140
PAGE = 1
LINE = 9
COLUMN = 17
HELP = Enter value for : ESERADI

ENDDEFINE FIELD

DEFINE FIELD

NAME = ISTAR
DATATYPE = CHAR
LENGTH = 20
DISPLAY_LENGTH = 20
QUERY_LENGTH = 20
PAGE = 1
LINE = 11
COLUMN = 22
HELP = Enter value for : ISTAR

ENDDEFINE FIELD

DEFINE FIELD

NAME = ISTYER
DATATYPE = CHAR
LENGTH = 20
DISPLAY_LENGTH = 20
QUERY_LENGTH = 20
PAGE = 1
LINE = 11
COLUMN = 60
HELP = Enter value for : ISTYER

ENDDEFINE FIELD

DEFINE FIELD

NAME = MUSTAD1
DATATYPE = CHAR
LENGTH = 40
DISPLAY_LENGTH = 25
QUERY_LENGTH = 40
PAGE = 1
LINE = 14
COLUMN = 7
HELP = Enter value for : MUSTAD1

ENDDEFINE FIELD

DEFINE FIELD

NAME = MUSTAD2
DATATYPE = CHAR
LENGTH = 40
DISPLAY_LENGTH = 25
QUERY_LENGTH = 40
PAGE = 1
LINE = 14
COLUMN = 47
HELP = Enter value for : MUSTAD2

ENDDEFINE FIELD

DEFINE FIELD

NAME = TERCTAR
DATATYPE = CHAR
LENGTH = 20


```
DISPLAY_LENGTH = 20
QUERY_LENGTH = 20
PAGE = 1
LINE = 16
COLUMN = 21 value for : KUTUPHANE
HELP = Enter value for : TERCTAR
```

ENDDEFINE FIELD

DEFINE FIELD

```
NAME = MUTERAD
DATATYPE = CHAR
LENGTH = 40
DISPLAY_LENGTH = 40
QUERY_LENGTH = 40
PAGE = 1
LINE = 18
COLUMN = 21 value for : KOLEKSIYON
HELP = Enter value for : MUTERAD
```

ENDDEFINE FIELD

DEFINE FIELD

```
NAME = MUELHAT
DATATYPE = CHAR
LENGTH = 1
DISPLAY_LENGTH = 1
QUERY_LENGTH = 1
PAGE = 1
LINE = 20
COLUMN = 21
HELP = Enter value for : MUELHAT
```

ENDDEFINE FIELD

DEFINE FIELD

```
NAME = HATCINS value for : YAZKONU
DATATYPE = CHAR
LENGTH = 15 INTO GIRISI.YAZKONU
DISPLAY_LENGTH = 15 DEWEY--TASNIFNO
QUERY_LENGTH = 15
PAGE = 1
LINE = 20
COLUMN = 39
HELP = Enter value for : HATCINS
LOV_TEXT = <<<
SELECT HATADI
      INTO GIRISI.HATCINS FROM HATLAR
>>>
LOV_TITLE = HAT_SEC
LOV_X = 10
LOV_Y = 10
```

ENDDEFINE FIELD

DEFINE FIELD

```
NAME = KUTUPHANE
DATATYPE = CHAR
LENGTH = 20
DISPLAY_LENGTH = 20
```



```
QUERY_LENGTH = 20
PAGE = 1
LINE = 22
COLUMN = 21
HELP = Enter value for : KUTUPHANE
```

ENDDEFINE FIELD

DEFINE FIELD

```
NAME = KOLEKSIYON
DATATYPE = CHAR
LENGTH = 15
DISPLAY_LENGTH = 15
QUERY_LENGTH = 15
PAGE = 1
LINE = 22
COLUMN = 58
HELP = Enter value for : KOLEKSIYON
LOV_TEXT = <<<
SELECT KOLKISAADI
      INTO GIRIS1.KOLEKSIYON FROM KOLEKS
>>>
LOV_TITLE = KOLEKS_SEC
LOV_X = 10
LOV_Y = 10
```

ENDDEFINE FIELD

DEFINE FIELD

```
NAME = YAZKONU
DATATYPE = CHAR
LENGTH = 60
DISPLAY_LENGTH = 60
QUERY_LENGTH = 60
PAGE = 2
LINE = 2
COLUMN = 12
HELP = Enter value for : YAZKONU
LOV_TEXT = <<<
SELECT KONUADI INTO GIRIS1.YAZKONU
      FROM KONULAR WHERE DEWEY=:TASNIFNO
>>>
```

ENDDEFINE FIELD

DEFINE FIELD

```
NAME = FOTOKOPI
DATATYPE = CHAR
LENGTH = 5
DISPLAY_LENGTH = 5
QUERY_LENGTH = 5
PAGE = 2
LINE = 4
COLUMN = 17
HELP = Enter value for : FOTOKOPI
```

ENDDEFINE FIELD

DEFINE FIELD

```
NAME = MIKROFNO
```



```

DATATYPE = CHAR
LENGTH = 5
DISPLAY_LENGTH = 5
QUERY_LENGTH = 5
PAGE = 2
LINE = 4
COLUMN = 41
HELP = Enter value for : MIKROFNO

ENDDEFINE FIELD

DEFINE FIELD

NAME = RISALEYER
DATATYPE = CHAR
LENGTH = 9
DISPLAY_LENGTH = 9
QUERY_LENGTH = 9
PAGE = 2
LINE = 4
COLUMN = 63
HELP = Enter value for : RISALEYER

ENDDEFINE FIELD

DEFINE FIELD

NAME = OLCU
DATATYPE = CHAR
LENGTH = 15
DISPLAY_LENGTH = 15
QUERY_LENGTH = 15
PAGE = 2
LINE = 6
COLUMN = 17
HELP = Enter value for : OLCU

ENDDEFINE FIELD

DEFINE FIELD

NAME = EKSIKSAYFA
DATATYPE = CHAR
LENGTH = 30
DISPLAY_LENGTH = 30
QUERY_LENGTH = 30
PAGE = 2
LINE = 6
COLUMN = 49
HELP = Enter value for : EKSIKSAYFA

ENDDEFINE FIELD

DEFINE FIELD

NAME = SATSAYI
DATATYPE = CHAR
LENGTH = 5
DISPLAY_LENGTH = 5
QUERY_LENGTH = 5
PAGE = 2
LINE = 8
COLUMN = 50
HELP = Enter value for : SATSAYI

```


ENDDEFINE FIELD

DEFINE FIELD

NAME = SUTSAYI
DATATYPE = CHAR
LENGTH = 1
DISPLAY_LENGTH = 1
QUERY_LENGTH = 1
PAGE = 2
LINE = 8
COLUMN = 76
HELP = Enter value for : SUTSAYI

ENDDEFINE FIELD

DEFINE FIELD

NAME = YAPADET
DATATYPE = CHAR
LENGTH = 13
DISPLAY_LENGTH = 13
QUERY_LENGTH = 13
PAGE = 2
LINE = 8
COLUMN = 17
HELP = Enter value for : YAPADET

ENDDEFINE FIELD

DEFINE FIELD

NAME = CILTKAC
DATATYPE = CHAR
LENGTH = 2
DISPLAY_LENGTH = 2
QUERY_LENGTH = 2
PAGE = 2
LINE = 10
COLUMN = 17
HELP = Enter value for : CILTKAC

ENDDEFINE FIELD

DEFINE FIELD

NAME = KOPYASAYI
DATATYPE = CHAR
LENGTH = 2
DISPLAY_LENGTH = 2
QUERY_LENGTH = 2
PAGE = 2
LINE = 10
COLUMN = 49
HELP = Enter value for : KOPYASAYI

ENDDEFINE FIELD

DEFINE FIELD

NAME = KAGIT
DATATYPE = CHAR
LENGTH = 60

DISPLAY_LENGTH = 60
QUERY_LENGTH = 60
PAGE = 2
LINE = 12
COLUMN = 20
HELP = Enter value for : KAGIT

ENDDEFINE FIELD

DEFINE FIELD

NAME = MUZEHHIPAD
DATATYPE = CHAR
LENGTH = 40
DISPLAY_LENGTH = 40
QUERY_LENGTH = 40
PAGE = 2
LINE = 14
COLUMN = 20
HELP = Enter value for : MUZEHHIPAD

ENDDEFINE FIELD

DEFINE FIELD

NAME = TEZSAYFA
DATATYPE = CHAR
LENGTH = 50
DISPLAY_LENGTH = 50
QUERY_LENGTH = 50
PAGE = 2
LINE = 16
COLUMN = 20
HELP = Enter value for : TEZSAYFA

ENDDEFINE FIELD

DEFINE FIELD

NAME = TEZUSLUP
DATATYPE = CHAR
LENGTH = 135
DISPLAY_LENGTH = 60
QUERY_LENGTH = 135
PAGE = 2
LINE = 18
COLUMN = 20
HELP = Enter value for : TEZUSLUP

ENDDEFINE FIELD

DEFINE FIELD

NAME = MINADET
DATATYPE = CHAR
LENGTH = 3
DISPLAY_LENGTH = 3
QUERY_LENGTH = 3
PAGE = 2
LINE = 20
COLUMN = 20
HELP = Enter value for : MINADET

ENDDEFINE FIELD

DEFINE FIELD

NAME = MINUSLUP
DATATYPE = CHAR
LENGTH = 40
DISPLAY_LENGTH = 40
QUERY_LENGTH = 40
PAGE = 2
LINE = 20
COLUMN = 40

ENDDEFINE FIELD

DEFINE FIELD

NAME = MINSAYFA
DATATYPE = CHAR
LENGTH = 50
DISPLAY_LENGTH = 50
QUERY_LENGTH = 50
PAGE = 2
LINE = 22
COLUMN = 25
HELP = Enter value for : MINSAYFA

ENDDEFINE FIELD

DEFINE FIELD

NAME = CILTOZ
DATATYPE = CHAR
LENGTH = 90
DISPLAY_LENGTH = 60
QUERY_LENGTH = 90
PAGE = 3
LINE = 2
COLUMN = 19
HELP = Enter value for : CILTOZ

ENDDEFINE FIELD

DEFINE FIELD

NAME = HATTAT
DATATYPE = CHAR
LENGTH = 40
DISPLAY_LENGTH = 40
QUERY_LENGTH = 40
PAGE = 3
LINE = 4
COLUMN = 19
HELP = Enter value for : HATTAT

ENDDEFINE FIELD

DEFINE FIELD

NAME = SARIH
DATATYPE = CHAR
LENGTH = 40
DISPLAY_LENGTH = 40
QUERY_LENGTH = 40
PAGE = 3

LINE = 6
COLUMN = 19
HELP = Enter value for : SARIH

ENDDEFINE FIELD

DEFINE FIELD

NAME = TALIKATYAZ
DATATYPE = CHAR
LENGTH = 40
DISPLAY_LENGTH = 40
QUERY_LENGTH = 40
PAGE = 3
LINE = 8
COLUMN = 19
HELP = Enter value for : TALIKATYAZ

ENDDEFINE FIELD

DEFINE FIELD

NAME = IHTISARYAZ
DATATYPE = CHAR
LENGTH = 40
DISPLAY_LENGTH = 40
QUERY_LENGTH = 40
PAGE = 3
LINE = 10
COLUMN = 19
HELP = Enter value for : IHTISARYAZ

ENDDEFINE FIELD

DEFINE FIELD

NAME = ZEYLYAZ
DATATYPE = CHAR
LENGTH = 40
DISPLAY_LENGTH = 40
QUERY_LENGTH = 40
PAGE = 3
LINE = 12
COLUMN = 19
HELP = Enter value for : ZEYLYAZ

ENDDEFINE FIELD

DEFINE FIELD

NAME = SERH
DATATYPE = CHAR
LENGTH = 40
DISPLAY_LENGTH = 40
QUERY_LENGTH = 40
PAGE = 3
LINE = 14
COLUMN = 19
HELP = Enter value for : SERH

ENDDEFINE FIELD

DEFINE FIELD

NAME = KIRAAT
DATATYPE = CHAR
LENGTH = 1
DISPLAY_LENGTH = 1
QUERY_LENGTH = 1
PAGE = 3
LINE = 16
COLUMN = 19
HELP = Enter value for : KIRAAT

ENDDEFINE FIELD

DEFINE FIELD

NAME = SIMA
DATATYPE = CHAR
LENGTH = 1
DISPLAY_LENGTH = 1
QUERY_LENGTH = 1
PAGE = 3
LINE = 16
COLUMN = 35
HELP = Enter value for : SIMA

ENDDEFINE FIELD

DEFINE FIELD

NAME = RIVAYET
DATATYPE = CHAR
LENGTH = 1
DISPLAY_LENGTH = 1
QUERY_LENGTH = 1
PAGE = 3
LINE = 16
COLUMN = 54
HELP = Enter value for : RIVAYET

ENDDEFINE FIELD

DEFINE FIELD

NAME = FEVAID
DATATYPE = CHAR
LENGTH = 1
DISPLAY_LENGTH = 1
QUERY_LENGTH = 1
PAGE = 3
LINE = 16
COLUMN = 66
HELP = Enter value for : FEVAID

ENDDEFINE FIELD

DEFINE FIELD

NAME = MUTALEA
DATATYPE = CHAR
LENGTH = 1
DISPLAY_LENGTH = 1
QUERY_LENGTH = 1
PAGE = 3
LINE = 18
COLUMN = 19


```
HELP = Enter value for : MUTALEA
ENDDEFINE FIELD
DEFINE FIELD
NAME = MUTALEATAR
DATATYPE = CHAR
LENGTH = 20
DISPLAY_LENGTH = 20
QUERY_LENGTH = 20
PAGE = 3
LINE = 18
COLUMN = 39
HELP = Enter value for : MUTALEATAR
ENDDEFINE FIELD
DEFINE FIELD
NAME = TEMELTAR
DATATYPE = CHAR
LENGTH = 20
DISPLAY_LENGTH = 20
QUERY_LENGTH = 20
PAGE = 3
LINE = 20
COLUMN = 21
HELP = Enter value for : TEMELTAR
ENDDEFINE FIELD
DEFINE FIELD
NAME = MUKABTAR
DATATYPE = CHAR
LENGTH = 20
DISPLAY_LENGTH = 20
QUERY_LENGTH = 20
PAGE = 3
LINE = 20
COLUMN = 60
HELP = Enter value for : MUKABTAR
ENDDEFINE FIELD
DEFINE FIELD
NAME = MULKIYET
DATATYPE = CHAR
LENGTH = 110
DISPLAY_LENGTH = 60
QUERY_LENGTH = 110
PAGE = 3
LINE = 22
COLUMN = 20
HELP = Enter value for : MULKIYET
ENDDEFINE FIELD
DEFINE FIELD
NAME = VAKIFTAR
DATATYPE = CHAR
```


LENGTH = 20
DISPLAY_LENGTH = 20
QUERY_LENGTH = 20
PAGE = 4
LINE = 2
COLUMN = 24
HELP = Enter value for : VAKIFTAR

ENDDEFINE FIELD

DEFINE FIELD

NAME = VAKIFKAYDI
DATATYPE = CHAR
LENGTH = 50
DISPLAY_LENGTH = 50
QUERY_LENGTH = 50
PAGE = 4
LINE = 4
COLUMN = 17
HELP = Enter value for : VAKIFKAYDI

ENDDEFINE FIELD

DEFINE FIELD

NAME = RESIMPLAN
DATATYPE = CHAR
LENGTH = 40
DISPLAY_LENGTH = 40
QUERY_LENGTH = 40
PAGE = 4
LINE = 6
COLUMN = 33
HELP = Enter value for : RESIMPLAN

ENDDEFINE FIELD

DEFINE FIELD

NAME = MUHTEVASI1
DATATYPE = CHAR
LENGTH = 210
DISPLAY_LENGTH = 60
QUERY_LENGTH = 210
PAGE = 4
LINE = 9
COLUMN = 16
HELP = Enter value for : MUHTEVASI1

ENDDEFINE FIELD

DEFINE FIELD

NAME = MUHTEVASI2
DATATYPE = CHAR
LENGTH = 210
DISPLAY_LENGTH = 60
QUERY_LENGTH = 210
PAGE = 4
LINE = 10
COLUMN = 16
HELP = Enter value for : MUHTEVASI2

ENDDEFINE FIELD

DEFINE FIELD

NAME = MUHTEVASI3
DATATYPE = CHAR
LENGTH = 180
DISPLAY_LENGTH = 60
QUERY_LENGTH = 180
PAGE = 4
LINE = 11
COLUMN = 16
HELP = Enter value for : MUHTEVASI3

ENDDEFINE FIELD

DEFINE FIELD

NAME = ACK1
DATATYPE = CHAR
LENGTH = 210
DISPLAY_LENGTH = 60
QUERY_LENGTH = 210
PAGE = 4
LINE = 14
COLUMN = 16
HELP = Enter value for : ACK1

ENDDEFINE FIELD

DEFINE FIELD

NAME = ACK2
DATATYPE = CHAR
LENGTH = 140
DISPLAY_LENGTH = 60
QUERY_LENGTH = 140
PAGE = 4
LINE = 15
COLUMN = 16
HELP = Enter value for : ACK2

ENDDEFINE FIELD

DEFINE BLOCK

INE SCREEN

DEFINE PAGE

PAGE = 1
PAGE_XS = 0
PAGE_YS = 0
PAGE_PX0 = 0
PAGE_PY0 = 0
PAGE_PXS = 0
PAGE_PYS = 0
PAGE_SX0 = 0
PAGE_SY0 = 0
MODE = TEXT
LINE = 2
BOILER = <<<

TURKCE YAZMA ESER GIRISI

DEMIRBAS NO :

TASNIF NO :

CILT KAYDI :

>>>

LINE = 5

BOILER = <<<

MUELLIF ADI :

TELIF T:

>>>

LINE = 7

BOILER = <<<

ESERIN MESHUR ADI :

>>>

LINE = 9

BOILER = <<<

ESERIN ADI :

>>>

LINE = 11

BOILER = <<<

ISTINSAH TARIHI :

ISTINSAH YERI :

>>>

LINE = 13

BOILER = <<<

MUSTENSIHLERI :

1.

2.

>>>

LINE = 16

BOILER = <<<

TERCUME TARIHI :

>>>

LINE = 18

BOILER = <<<

MUTERCIMI :

>>>

LINE = 20

BOILER = <<<

MUELLIF HATTI :

HATCINS :

>>>

LINE = 22

BOILER = <<<

KUTUPHANESI :

KOLEKSIYONU :

>>>

MODE = BOX

BOILER = <<<

p-----

>>>

LINE = 16

BOILER = <<<

TEZHIPLI SAYFA :

>>>

LINE = 18

BOILER = <<<

TEZHIPLI OZELLIGI :

>>>

LINE = 20

BOILER = <<<

MINYATUR ADEDI :

MINYATUR USULU :

>>>

LINE = 22

BOILER = <<<

MINYATURLU SAYFALAR :

>>>

MODE = BOX

BOILER = <<<

b-----

>>>
ENDDEFINE PAGE
DEFINE PAGE

PAGE = 2
PAGE_XS = 0
PAGE_YS = 0
PAGE_PX0 = 0
PAGE_PY0 = 0
PAGE_PXS = 0
PAGE_PYS = 0
PAGE_SX0 = 0
PAGE_SY0 = 0
MODE = TEXT
LINE = 2
BOILER = <<<
KONUSU :

>>>

LINE = 4
BOILER = <<<
FOTOKOPI NO :

MIKROFILM NO :

RISALE YERI :

>>>

LINE = 6
BOILER = <<<
OLCUSU :

EKSIK SAYFA :

>>>

LINE = 8
BOILER = <<<
YAPRAK ADEDI:

SATIR SAYISI :

SUTUN SA

>>>

LINE = 10
BOILER = <<<
CILT ADEDI :

KOPYA ADEDI :

>>>

LINE = 12
BOILER = <<<
KAGIT OZELLIGI :

>>>

LINE = 14
BOILER = <<<
MUZEHHIBIN ADI :

>>>

LINE = 16
BOILER = <<<
TEZHIPLI SAYFA :

>>>

LINE = 18
BOILER = <<<
TEZHIP OZELLIGI:

>>>

LINE = 20
BOILER = <<<
MINYATUR ADEDI : MINYATUR USLUP:

>>>

LINE = 22
BOILER = <<<
MINYATURLU SAYFALAR :

>>>

MODE = BOX
BOILER = <<<

P-----
|


```
>>>
LINE = 16
BOILER = <<<
KIRANT KAYDI : SIMA KAYDI : RIVAYET KAYDI : PEVAID :
>>>
LINE = 18
BOILER = <<<
MUTALEA KAYDI : MUTALEA TARİHİ :
>>>
LINE = 20
BOILER = <<<
TEMELLOK TARİHİ : MUKABELE TARİHİ:
>>>
LINE = 22
BOILER = <<<
MULKIYET KAYDI :
>>>
MODE = BOX
BOILER = <<<
```

```
b-----
>>>
```

ENDDEFINE PAGE

DEFINE PAGE

```
PAGE = 3
PAGE_XS = 0
PAGE_YS = 0
PAGE_PX0 = 0
PAGE_PY0 = 0
PAGE_PXS = 0
PAGE_PYS = 0
PAGE_SX0 = 0
PAGE_SY0 = 0
MODE = TEXT
LINE = 2
BOILER = <<<
CILT OZELLIGI :
>>>
LINE = 4
BOILER = <<<
HATTATI :
>>>
LINE = 6
BOILER = <<<
HASIYE YAZAN :
>>>
LINE = 8
BOILER = <<<
TALIKAT YAZAN :
>>>
LINE = 10
BOILER = <<<
IHTISAR EDEN :
>>>
LINE = 12
BOILER = <<<
ZEYL YAZAN :
>>>
LINE = 14
BOILER = <<<
SERH YAZAN :
```



```
>>>
LINE = 16
BOILER = <<<
    KIRAAT KAYDI :      SIMA KAYDI :      RIVAYET KAYDI :      FEVAID :
>>>
LINE = 18
BOILER = <<<
    MUTALEA KAYDI :      MUTALEA TARIHI :
>>>
LINE = 20
BOILER = <<<
    TEMELLUK TARIHI :      MUKABELE TARIHI:
>>>
LINE = 22
BOILER = <<<
    MULKIYET KAYDI :
>>>
MODE = BOX
BOILER = <<<
```

p-----

b-----

```
>>>
```

```
ENDDEFINE PAGE
```

```
DEFINE PAGE
```

```
PAGE = 4
PAGE_XS = 0
PAGE_YS = 0
PAGE_PX0 = 0
PAGE_PY0 = 0
PAGE_PXS = 0
PAGE_PYS = 0
PAGE_SX0 = 0
PAGE_SY0 = 0
MODE = TEXT
LINE = 2
BOILER = <<<
    VAKIF KAYDI TARIHI :
>>>
LINE = 4
BOILER = <<<
```


VAKIF KAYDI :

>>>

LINE = 6

BOILER = <<<

RESIM, PLAN VS. OZELLIKLERI :

>>>

LINE = 8

BOILER = <<<

KITABIN

MUHTEVASI :

>>>

LINE = 13

BOILER = <<<

DIGER

ACIKLAMALAR :

>>>

MODE = BOX

BOILER = <<<

p-----

b-----

>>>

ENDDEFINE PAGE

DEFINE PAGE

PAGE = 5

PAGE_XS = 0

PAGE_YS = 0

PAGE_PX0 = 0

PAGE_PY0 = 0

PAGE_PXS = 0

PAGE_PYS = 0

PAGE_SX0 = 0

PAGE_SY0 = 0

MODE = TEXT

LINE = 2

BOILER = <<<

>>>

===== GIRIS1 =====

ENDDEFINE PAGE

UNDEFINE SCREEN

UNDEFINE FORM

YAZMA ADI :

TEKİF T :

YAZMA MESHUR ADI :

YAZMA ADI :

YAZMA TARİHİ :

YAZMA YERİ :

YAZMA STENSİLERİ :

2.

YAZMA TARİHİ :

YAZMA

YAZMA HATTI :

HATCINS

YAZMA

KOLEKSİYONU :

*0

<Replac

YAZMA

KOPI NO :

MIKROFILM NO :

RİSALE YERİ :

YAZMA

EKSİK SAYFA :

YAZMA ADEDİ :

SATIR SAYISI :

SÜTUN SAYISI :

YAZMA ADEDİ :

KOPI ADEDİ :

YAZMA ÖZELLİĞİ :

YAZMA ADI :

YAZMA SAYFA :

YAZMA ÖZELLİĞİ :

YAZMA ADEDİ :

MINYATÜR ÜSLUP :

YAZMA SAYFALARI :

*0

<Replac

LT OZELLIGI : _____

TTATI : _____

SIYE YAZAN : _____

LIKAT YAZAN : _____

ETISAR EDEN : _____

YVL YAZAN : _____

HRH YAZAN : _____

RAAT KAYDI : _ SIMA KAYDI : _ RIVAYET KAYDI : _ FEVAID : _

TALEA KAYDI : _ MUTALEA TARIHI : _____

MELLUK TARIHI : _____ MUKABELE TARIHI: _____

MLKIYET KAYDI : _____

tt: *0 _____ <Replac

KIF KAYDI TARIHI : _____

KIF KAYDI : _____

SIM,PLAN VS. OZELLIKLERI : _____

TABIN
HTEVASI : _____

GER
IKLAMALAR : _____

tt: *0 _____ <Replac

1. TUPECE BILGI GIRISI
2. ARAPCA BILGI GIRISI
3. FARSCA BILGI GIRISI
4. DIGER BILGI GIRISI

Purpose:

GIRIS

Appendix A

Full Menu Application Information

A.1 Application Information

Application: ZMENU
Short Name: ZMENU
File Name: ZMENU
Creation Date: 20-AUG-92
Creator: ZEKIYE
Last Release Date:
Menu Directory:
Version: 1
Identification: ZMENU

Full Menu Application Information

Page A-1

A.2 Menu overview

ZMENU

Full Menu Application Information

Page A-2

A.3 ZMENU

SULEYMANIYE KUTUPHANESI BILGI GIRISI

BILGI GIRISI

- 1 TURKCE BILGI GIRISI
- 2 ARAPCA BILGI GIRISI
- 3 FARSCA BILGI GIRISI
- 4 DIGER BILGI GIRISI

Purpose:

GIRIS

Short Name: TURKCE

Command Type: Runform.

Command Line:

=====

RUNFORM C:\ZEK\ZEKFORM\GIRIS1 &UN/&PW

Hint:

=====

No detailed help available for this item.

Item Roles:

=====

KULLANICI

A.3.2 Item Number: 2

Short Name: ARAPCA

Command Type: Runform.

Command Line:

=====

RUNFORM C:\ZEK\ZEKFORM\GIRIS2 &UN/&PW

Hint:

=====

No detailed help available for this item.

Full Menu Application Information

Page A-3

Item Roles:

=====

KULLANICI

A.3.3 Item Number: 3

Short Name: FARSCA

Command Type: Runform.

Command Line:

=====

RUNFORM C:\ZEK\ZEKFORM\GIRIS3 &UN/&PW

Hint:

=====

No detailed help available for this item.

Item Roles:

=====

No roles listed for this item.

A.3.4 Item Number: 4

Short Name: DIGER


```
*****
program SQL*FORMS ile hazirlanan bir formdan trigger yardimiyla
n Attributelarini degistiren C programidir
YE EMIROGLU
```

```
***** */
```

```
lude <stdio.h>
lude <stdlib.h>
lude <string.h>
lude <conio.h>
lude <dos.h>
lude <io.h>
lude <bios.h>
lude <math.h>
lude <mem.h>
lude <fcntl.h>
lude <sys\stat.h>
line ATRIB 0xFA
QL stmt #1
SQL INCLUDE sqlca;
```

```
opyright (c) 1985,1986 by Oracle Corporation. */
```

LCA : SQL Communications Area.

TION

ntains no code. Oracle fills in the SQLCA with status info
ring the execution of a SQL stmt.

S

the symbol SQLCA_STORAGE_CLASS is defined, then the SQLCA
ll be defined to have this storage class. For example:

```
#define SQLCA_STORAGE_CLASS extern
```

ll define the SQLCA as an extern.

the symbol SQLCA_INIT is defined, then the SQLCA will be
atically initialized. Although this is not necessary in order
use the SQLCA, it is a good pgming practice not to have
itialized variables. However, some C compilers/OS's don't
low automatic variables to be init'd in this manner. Therefore,
you are INCLUDE'ing the SQLCA in a place where it would be
automatic AND your C compiler/OS doesn't allow this style
initialization, then SQLCA_INIT should be left undefined --
l others can define SQLCA_INIT if they wish.

w rules for defining SQLCA_INIT, SQLCA_STORAGE_CLASS, and DLL in OS/2:
ers should not define SQLCA_STORAGE_CLASS if defining DLL.
LCA_STORAGE_CLASS is primarily used for single-threaded programs
d for internal development.

FIED

amura	08/15/89	- OS/2: users must define SQLMT for multi-threaded case
amura	06/23/89	- OS/2: modify for multi-threaded case
are	12/06/84	- Ch SQLCA to not be an extern.
are	10/21/85	- Add initialization.
adbury	01/05/86	- Only initialize when SQLCA_INIT set
are	06/12/86	- Add SQLCA_STORAGE_CLASS option.

```
def SQLCA
```



```

/* b4 */ long      sqlabc;
/* b4 */ long      sqlcode;
struct
{
    /* ub2 */ unsigned short sqlerrml;
    /* ub1 */ char          sqlerrmc[70];
} sqlerrm;
/* ub1 */ char          sqlerrp[8];
/* b4 */ long          sqlerrd[6];
/* ub1 */ char          sqlwarn[8];
/* ub1 */ char          sqlext[8];
};

#ifdef SQLMT
extern struct sqlca *sqlcamt();
#define sqlca (*sqlcamt())
/* SQLMT */

#ifdef SQLCA_STORAGE_CLASS
SQLCA_STORAGE_CLASS struct sqlca sqlca
#define SQLCA_INIT
= {
    {'S', 'Q', 'L', 'C', 'A', ' ', ' ', ' ', ' '},
    sizeof(struct sqlca),
    0,
    { 0, {0}},
    {'N', 'O', 'T', ' ', 'S', 'E', 'T', ' ', ' '},
    {0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0}
}
#ifdef SQLCA_INIT */
;

/* SQLCA_STORAGE_CLASS */
struct sqlca sqlca
/* For single-threaded version :

#define SQLCA_INIT
= {
    {'S', 'Q', 'L', 'C', 'A', ' ', ' ', ' ', ' '},
    sizeof(struct sqlca),
    0,
    { 0, {0}},
    {'N', 'O', 'T', ' ', 'S', 'E', 'T', ' ', ' '},
    {0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0}
}
#ifdef SQLCA_INIT */
;

/* SQLCA_STORAGE_CLASS */

/* SQLMT */

#ifdef SQLCA */
/* SQLCA */
far *scr_buf;
renk(void)

t loc;int x;int y; char *str;
(peekb(0x0040,0x0049)==7) scr_buf=MK_FP(0xB000,0);

```



```

ic int SQLTM[8];
ic int sqlusi[1] = {

ic int SQLBT0 = 1;
ic int SQLBT1 = 2;
ic int SQLBT2 = 4;
ic int SQLBT3 = 9;
ic unsigned long sqlvsn = 10317;
ic int IAPSUCCL = 0;
ic int IAPFAIL = 1403;
ic int IAPFTL = 535;
def SQLPROTO
rn      sqliem();
if /* SQLPROTO */

*****
user_exit cagirmak icin hazirlanan tablen assembler program
YE EMIROGLU
***** */
e IAPXTB - IAP EXIT VECTOR TABLE

a segment para public 'data'
  assume ds:_data
ic _iapxtb
xtb label byte
11      ; pointer to name string
_renk    ; pointer to user-exit function
2        ; exit type/language is C
a ends

n _renk:far      ; declare outside all segments

a segment para public 'data'
d 0      ; end of table marker
b "renk", 0      ; ASCII name of exit
a ends
end

```


KAYNAKLAR

- OZKARAHAN Esen Database Managment Concepts,Design and Practice 1990
- McFADDEN Fred R. , HOFFER A.Jeffrey Database Management 1988
- DATE C.J. An Introduction to Database Systems 1981
- ALPAY Meral Kutuphanecilik Terimleri
- BAYSAL Jale Kütüphanecilik Alanında Yeni Kavramlar, Araçlar, Yöntemler
- GODDARD David The Support of Library Systems (Article)
- ORACLE ORACLE RDBMS Database Administrator Guide 1990
- ORACLE SQL Language Referance Manual 1990
- ASHTON-TATE Learning dBASE III PLUS 1985
- PRESSMAN Roger S. Software Engineering A Practitioner's Approach 1987
- PARSAYE K.- CHIGNELL M. Intelligent Databases 1989
- KHOSHAFIAN S. - WONG H.

ÖZGEÇMİŞ

Zekiye Emiroğlu; 1967 yılında İstanbul'da doğdu. Liseyi 1984 yılında Üsküdar Kız Lisesi'nde bitirdi. 1988 yılında Yıldız Üniversitesi Bilgisayar Bilimleri ve Mühendisliği Bölümünden mezun oldu. Aynı yıl Yıldız Üniversitesi , Fen Bilimleri Enstitüsü, Bilgisayar Bilimleri Anabilim dalında Yüksek Lisans eğitimine başladı. 1988 yılından beri aynı üniversitenin Bilgisayar Bilimleri ve Mühendisliği Bölümünde Araştırma Görevlisi olarak görev yapmaktadır.

