

39276.

İSTANBUL TEKNİK ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

ÖĞRENCİ İŞLERİ OTOMASYONU

YÜKSEK LİSANS TEZİ

Müh. Hakan KAZAZ

Tezin Enstitüye Verildiği Tarih : 25 Ocak 1993

Tezin Savunulduğu Tarih : 21 Mayıs 1993

Tez Danışmanı : Doç. Dr. Füsun TUNALI

Diğer Jüri Üyeleri : Prof. Dr. Nadir YÜCEL
Prof. Dr. Emre HARMANCI

**İT. YÜKSEKÖĞRETİM KURULU
DOKÜMANTASYON MERKEZİ**

MAYIS 1993

ÖNSÖZ

Tezi gerçekleştirmemde değerli katkılarını esirgemeyen Sayın Doç. Dr. Füsun TUNALI'ya, yardımlarından dolayı Sayın Cem ARPACI'ya ve bu kitapçığın hazırlanmasında emeği geçen Bilgi İşlem Merkezi çalışanlarına teşekkür ederim.

Hakan KAZAZ, 1993.

İÇİNDEKİLER

ŞEKİL LİSTESİ	v
ÖZET	vi
SUMMARY	vii
BÖLÜM 1. GİRİŞ	1
BÖLÜM 2. VERİ TABANI SİSTEMLERİ	4
2.1. Klasik Dosya İşleme Sistemi ve Veritabanı Yaklaşımı	5
2.1.1. Klasik Dosya İşleme Sistemi	5
2.1.2. Veritabanı Yaklaşımı	7
2.2. Veritabanı Sistemleri	8
2.2.1. Veriler	8
2.2.2. Donanım	10
2.2.3. Yazılım	10
2.2.4. Kullanıcılar	11
2.3. Veritabanı Modelleri	12
2.3.1. Hiyerarşik Model	13
2.3.2. Ağ Model	14
2.3.3. İlişkisel Model	15
2.4. Veritabanı Mimarisi	16
2.4.1. Dış (Mantıksal) Seviye	17
2.4.2. Kavramsal Seviye	18
2.4.3. İç (fiziksel) Seviye	19
2.4.4. Dönüşümler	21
2.5. Veritabanına Erişim	21
2.5.1. Disk Yöneticisi	22
2.5.2. Dosya Yöneticisi	23
2.6. Erişim Performansı	24
2.6.1. Bloklama (Clustering)	24
2.6.2. Sayfa Yönetimi	25
2.6.3. Kayıt Yönetimi	26
2.6.4. İndeksleme	27
2.6.4.1. B+ Ağacı	29
2.6.4.2. Otomasyonda B+ Ağacı	31
BÖLÜM 3. ÖİÖ'YA GENEL BAKIŞ	35
3.1. ÖİÖ Kapsamı ve Amaçları	35
3.2. ÖİÖ'da Süreçler	36
3.2.1. Kayda Hazırlık Süreci	37
3.2.2. Kayıt Süreci	39

	3.2.3. Not işleme Süreci	41
	3.2.4. Karton (öğrenci izleme) Süreci	42
	3.2.5. Raporlama Süreci	43
	3.3. Süreçlerin iş Akışı	44
	3.3.1. Fakülte Geneli işlemleri	44
	3.3.1.1. Kayda Hazırlık Süreci	44
	3.3.1.2. Not işleme Süreci	46
	3.3.1.3. Karton Süreci.....	46
	3.3.1.4. Raporlama Süreci.....	47
	3.3.2. Öğrenci Bazlı işlemler	47
	3.3.2.1. Kayda Hazırlık Süreci	47
	3.3.2.2. Kayıt Süreci	47
	3.3.2.3. Not işleme Süreci	48
	3.3.2.4. Karton Süreci	48
	3.3.2.5. Raporlama Süreci	49
BÖLÜM	4. TEMEL VERİ DOSYALARI	50
	4.1. Fakülte Dosyası	50
	4.2. Özlük Dosyası	52
	4.3. Kayıt Dosyası	53
	4.4. Karton Dosyası	54
	4.5. Not Dosyaları	55
	4.6. Destek Dosyalar	56
BÖLÜM	5. VERİ TABANINI YÖNETEN TEMEL PROGRAMLAR	58
	5.1. Özlük Programı	59
	5.2. Kayıt Programı	60
	5.3. Not Programı	61
	5.4. Karton Programı	63
	5.5. Destek Programlar	64
	5.6. Hazır Altprogram Kütüphaneleri..	66
BÖLÜM	6. KARTON PROGRAMININ İNCELENMESİ..	69
	6.1. Karton Kavramı	69
	6.2. Karton Programının Yapısı	70
	6.3. Yazılımın Bölümleri ve Temel Alt Programları	72
SONUÇLAR VE ÖNERİLER		76
KAYNAKLAR		78
EKLER		79
ÖZGEÇMİŞ		115

ŞEKİL LİSTESİ

Şekil 2.1 Hiyerarşik veritabanı modeli örneği.	14
Şekil 2.2 ANSI/SPARC'ın 3 seviyeli mimarisi ..	17
Şekil 2.3 3 Seviyeli mimari örneği	20
Şekil 2.4 Veritabanına erişim	22
Şekil 2.5 Bir kaydın sayfa içindeki ötelenişi.	27
Şekil 2.6 indeksleme örneği	29
Şekil 2.7 B+ Ağacında birim eleman yapısı	32
Şekil 2.8 B+ ağacında bir sayfanın yapısı	32
Şekil 3.1 ÖİO süreçleri ve aralarındaki ilişkiler	37
Şekil 4.1 Fakülte kaydında önemli alanlar	51
Şekil 4.2 Özlük kaydında yer alan bazı bilgiler	53
Şekil 4.3 Yarıyıl kaydı özet bilgileri	54
Şekil 4.4 Karton kaydı özet bilgileri	55
Şekil 4.5 Not dosyası kaydı bilgileri	56
Şekil 5.1 ÖİO'daki entegre programlar	59
Şekil 5.2 ÖİO'yu oluşturan program, dosyalar ve aralarındaki ilişkiler .	68

ÖZET

Bu çalışmada i.T.Ü. fakülteleri öğrenci işleri bürolarında kullanılmak üzere, öğrenci kaydı ve durumunun izlenmesi amacıyla bir Öğrenci İşleri Otomasyonu tasarlanıp gerçekleştirilmiştir. Konu ele alınırken, hazırlanacak otomasyonun (tezde otomasyon kelimesi öğrenci işleri otomasyonu deyimi yerine kullanılmıştır) halen bazı öğrenci bürolarında kullanılan kayıt programı ve veritabanı ile uyumlu olması istenmiştir.

Bir veritabanında (DB) veri kümelerini ile birlikte aralarındaki ilişkiler de saklanır. Genel olarak bu veriler kalıcı, güncel ve paylaşımlı olabilirken bulundukları sistemin yapısına göre bu sıfatların anlamı değişebilir veya bazılarından yoksun kalabilirler. Bir veritabanı sistemi, veritabanı (DB) ve üzerinde çalışan yönetici yazılımdan (VTYS-DBMS) oluşan bir sistemdir.

Otomasyonda tasarlanan veritabanı dosyalar kümesinden meydana gelmektedir. Bu küme otomasyonun temel ve destek dosyalarından oluşmaktadır. Otomasyonda veri dosyalarına erişim performansını arttırmak amacıyla yaygın bir yöntem olan B+ Ağacı indeksleme yöntemi kullanılmıştır.

Otomasyon kendi içinde süreçlere ayrılmıştır. Her süreci yöneten bir veya birden fazla program bulunmakla beraber, bazı programlar da birden fazla sürece hizmet vermektedir. Bu özellik veritabanı dosyaları için de geçerlidir. Böyle bir organizasyonun nedeni, daha önce yapılan bazı çalışmaların -yeniden organize edilerek- otomasyona katılımını sağlamaktır.

Geliştirilen otomasyonunda aşağıdaki fonksiyonlar gerçekleştirilmiştir:

- Öğrenci özlük bilgilerinin saklanması ve rektörlükteki merkezi bilgisayarda SQL/DS ortamına aktarımı için bir arabirim dosyasının oluşturulması,
- Yarıyıl öğrenci kayıtlarının yapılması,
- Öğrenci kartonlarının hazırlanması ve izlenmesi ve
- Raporlamalar.

SUMMARY

DESIGN AND IMPLEMENTATION OF A STUDENT SERVICE APPLICATION ON PERSONAL COMPUTERS

In this study, a database and an application were designed and implemented on the personal computing environment for handling the student oriented services including basic functions of the faculty students of ITU such as education tracking.

While constructing these applications, previous studies and some professional applications packages including Turbo Turbo Professional from Turbo Power and Borland's Turbo Pascal Compiler and Turbo Access Database toolbox have been used. All of the programs were written in Borland's Turbo Pascal Compiler. The minimal configuration required to run these applications is an IBM PC/AT compatible computer running the MSDOS 4.0 or higher version operating system.

A Database System

A database is a set of stored data or more precisely, stored information which is interrelated. The basic interrelated information set is the stored record. A database includes data that not only the real values needed by the users, but also that some special values which the database uses for management and control. Information is the data which has meaning for the users i.e. end users or application programs.

The basic properties of the information in a database are:

- persistency,

- updatability and
- sharability.

Database Models

A database model is required to represent the entire physical database itself.

Nowadays there are several database models which are used in database systems. The primary task of the models is to organize the database and to provide an environment for accessing data.

The most widely known database models are:

- The Hierarchical Model,
- The Network Model and
- The Relational Model.

Some additional models such as The Fuzzy Relational model and The Object-Oriented model have been developed .

In a database on the hierarchical model, data is accessed in tree structures. There is a parent-child type relationship among the data. Users can access the information by scanning the data on the branch of the trees. The data is stored in a node or a segment of the tree and no data can be accessed until a path on data is defined. All related records are linked together unidirectionally within the scope of the hierarchy. The linking can be done by using pointers. In a tree, the top of the record is the physical record itself and the others are virtual records which are pointed by the records -that physical or virtual- from the upper levels. For this reason although the record accessing speed is very high, the insertion and the deletion functions are comlez and slow. Only experienced users can use this kind of database. IMS (Information Management System) is a hierarchical model database system.

The network model is the generalized version of the hierarchical model. Any two nodes can be linked together with one or more links. There are links in both directions between the segments. As a result more complex structure must be defined and more operations are needed for accessing data. This also requires experienced and skilled personnel for managing the database. It is clear that the insertion and the deletion functions are more complex than the others and the record accessing performance is low. IDMS (Integrated Database Managememt System) is a network model database system.

The relational model uses the properties of the relational algebra in the database management. All the files in a database correspond to tables in a relational model database. The whole system information is also stored in table structures. Every table defines a relation and consists of columns (fields). Each row (record) in a table corresponds to a tuple in the relational algebra. The well-defined relational algebra operations such as select, product, union, set difference and join, are used for managing tables. Users can access to the information only by querying with the columns which are specific to the tables. The relational model database system provides an interactive user environment for the terminal users. Non-professional users can access the database using only a few easy commands. DB2 and SQL (Structured Query Language) are the most known relational database systems.

The following figure illustrates a table in a relational database as it can be seen by the users:

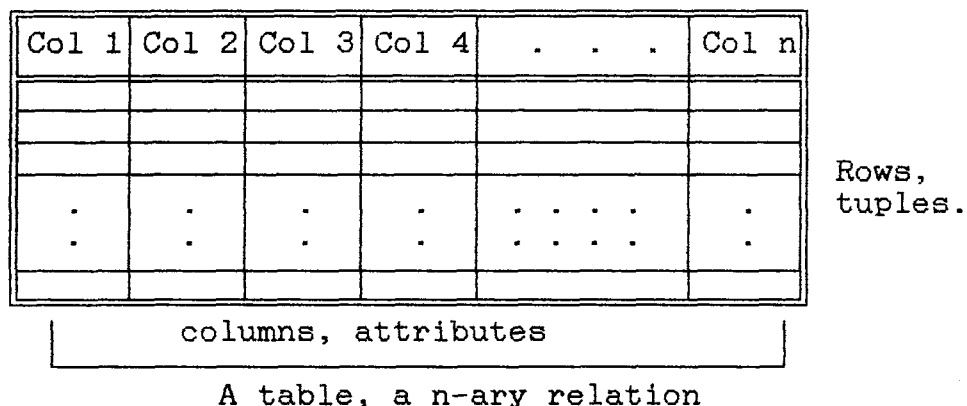


Figure 1. A table view on Relational Database.

The object-oriented model uses the concepts of abstract data, encapsulation and object inheritance for the handling the data. It is one of the newest models and it is often used in intelligent database systems. In this model, objects are classified into various files such as the data file and the message file. The operations which had been defined affect the objects.

The fuzzy relational model uses the fuzzy set theory and related concepts in database systems. Predefined membership functions provide the user accessing data with fuzzy queries.

Database Architecture

According to the ANSI/SPARC standard, a database architecture can be examined into three different levels:

- The External (logical) Level,
- The Conceptual Level and
- The Internal (physical) Level.

Each level's view is defined by that level's schema and each schema consists of definitions of the different types of its database records.

These records are defined by that level's DDL (Data Definition Language). The Insertion and the deletion operations is managed by that level's DML (Database Manipulation Language) and each data is queried by using that level's DQL (Database Query Language). All data definitions are compiled and stored to the Data Dictionary (DD). Most of the database systems provide a DDL at the external level for embedded use.

The external level is the individual user level. It is the level which is closest to the users. It has an abstract view of the conceptual level database. Different users can have different views. Each user may access any subset of the total database.

At the conceptual level, there is an abstract view of the total database. It represents all the information in the database, and for every user it is shared and common. Conceptual DDL definitions are not interested in the storage structure and the access details. There must be only the definitions of content of the information. Thus the data-independence can be provided.

The Internal level is the one closest to the data which physically stored.

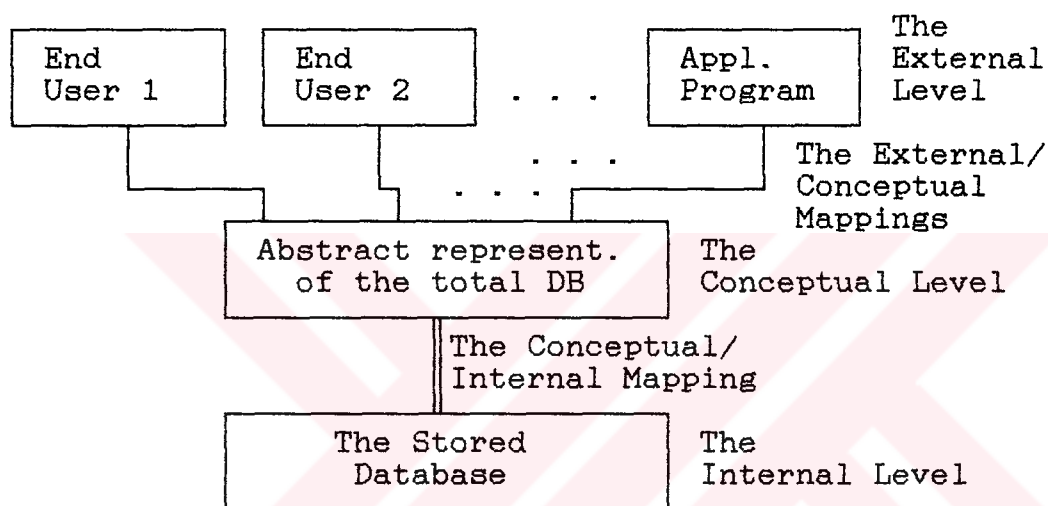


Figure 2. ANSI/SPARC's three level database architecture.

The internal level is concerned with the way the data is actually stored i.e. it is concerned with the techniques for achieving the maximum performance to access the database records (minimum disk access).

Several techniques have been used to minimize the disk access time. Some most important techniques are clustering, hashing and indexing.

By doing clustering, logically related records are stored physically adjacently. Thus accessing a record after another that logically related will require less time relatively.

Hashing is a translation from a field -hashed field- to a record address. It provides fast direct access to the data file. This translation is managed by a function called The Hash Function. The Input of the hash function is the primary key of a file which will be hashed and output is the record address which the key is related. It is obvious that for every key value there must be a different output. But in practice that is not possible and collisions were detected and overflow areas are used to solve this problem.

The indexing method is useful for both searching and accessing a data file with minimum disk I/O. Moreover, a data file can have several index files but only one hash file and the indexing method lets the users search the data file sequentially but the hashing does not.

A particularly common and important way of indexing can be made by using the B-Tree method. Nowadays, most of the database systems use the B-Tree (or B+Tree) method to access their data file records with the minimum disk I/O. The B+ tree structure consists of the dense and the nondense indexes.

The dense index is an index which there is an one to one correspondance between the entries of the index file and entries of the indexed file. In the nondense index, there is no one to one correspondance and only the specific keys of the data file records are indexed by the index file.

If the ordered entries of the dense index file are grouped, then it can be -again- indexed with a nondense index file.

The maximum number of entries of a B+ Tree segment describes the order of the B-Tree.

A B+ Tree of order of M is a tree that satisfies the following constraints:

- Excluding root, all pages have at least M entries,
- Each page has at least 2 and atmost $2 \cdot M$ items,
- Either a page is a leaf with no active node reference or it has one page reference for each entry in the segment plus one and
- All leaf nodes must be on the same level,

If a data file with N records is indexed with a B+ tree order of M, then at most order of $\text{LOG}(N)/\text{LOG}(M)$ disk I/O will be required to access any record in the data file.

Designed and Implemented Application

This thesis is intended to design and implement a student service application package to be used in the student service offices of the faculties of ITU. This package will be used for keeping the faculties' student information and tracking their successes and graduations primally.

Since the package is required to be compatible with the programs and the database implemented before such as student record keeping system, the package has been developed on some particular definitions and rules. Thus the package's programs has been written on the Turbo Pascal language and the database has been created on the B+ Tree indexing technique. Also this technique gives better access performance on the PC enviroment than the others.

In the package the consist of several programs. The most important ones are: CLASS, ENTEGRE and NOT.

All these ones, particularly the CLASS.PAS program, uses several units. These units are classified into 4 category:

The video tools written by Turbo Power Software Co: Tpcrt, Tpdos, Tppick, Tpwindow, Tpmenu, Tpentry, Tpstring, Tpmemchk, Opkey.

The definition, the creation and the accessing to the database files units are Types2, Karraccs2 (this unit is same as TACCESS unit of the Borland's but only its units and some definition files that the unit uses are different).

Support units that supports the main units: Destek1, Destek2 and Destek31.

Main units : Classeri, Class31, Class2, Class11, Classr11, Classsta, Classoz1 and Classdul.

Also, the package consist of logical processes. The first one is the 'Karton' process which is used to track the students' successes and to control the graduation. Infact the whole package is centered to this process.

Preparation for recording student process is required to able to record the studentts information,

Record process records the students to the lessons in each term and

Mark manipulation process.

Some of these processes use more than one program in the package. Yet, some programs serve more than one process.

The DBMS facilities is distributed to these programs. The DBMS interfaces in the package do not look like the database models that mentioned before. In fact they are the custom interfaces and can not be used for general queries, only the specific queries can be made.

The basic functions of the DBMS are the creation and the deletion of the files in the database system and insertion, the changing and the deletion of the records in a database file. As stated before all most accessable

files are accessed by using B+Tree indexing technique.

Conclusion

In this study the following functions were developed by using Turbo Pascal v5.5 compiler, some extended video tools and B+Tree database utility tools. All these elements and custom database queries make the package faster than any Database Management packages written for general use. But since the poor computing environment (IBM PC hardware with 640 KB main memory and particularly MSDOS operating system) leave the responsibilities of the security, the sharability and the extendibility of the database to the application programmer.

In the package the following functions are supported:

- Keeping the faculty students' family information,
- Students registration to a faculty,
- Tracking and keeping all of the students' education,
- Reports.

While developing the package it is required to be compatible with the programs and the student record keeping system mad before. For this reason, the package has been developed on some particular definitions and rules. These programs referenced by the other programs because of the 'compatibility' problem in the package. This limitation can be overcame by reanalyzing the system, understanding the customer's (office staff) requests, restructuring the database -particularly- by using an database management system.

It is recommended that this kind of applications must be developed on more reliable and multi-user operating and database management systems and central control must be provided.

BÖLÜM 1. GİRİŞ

Tezde ele alınan konu üniversitemiz fakülteleri öğrenci servislerinde kullanılmak üzere, öğrenci kaydı ve eğitim durumunu değerlendirme amacıyla bir öğrenci işleri Otomasyonu planlanması ve gerçekleşmesi problemidir. Bunun için gerekli sistem analizleri yapılmış, ihtiyaç duyulan yazılım ve veritabanı dosyaları belirlenip -daha önce yapılan çalışmalar da dikkate alınarak- üretilmiştir.

Üniversite yakın zamana kadar bu soruna ciddi bir yaklaşımda bulunmamış, fakülteler öğrenci servislerinde yürütülen işlerin bilgisayar ortamına aktarılması problemlerini kendisi çözmekle yükümlü kılınmış ve fakültelerden bazıları -özellikle Elektrik-Elektronik fakültesi- elindeki imkanlar (donanım, yazılım, yazılımcı) doğrultusunda soruna çözüm getirmeye ve çeşitli yazılımlar geliştirmeye çalışmıştır.

Ayrıca üniversite genelinde ve fakülte içlerinde birer bilgisayar ağı bulunmadığı için oluşturulan uygulamaların çoğu ilgili fakülteye özgü olmuş ve yazıldıkları yerde kalmıştır. Dolayısıyla yapılan çalışmalardan diğer fakültelerin ya haberi olmamış ya da net bir fayda sağlayamamışlardır.

Öğrenci işleri otomasyonu (ÖİO) planlanırken Elektrik-Elektronik fakültesinde bu güne kadar yapılan çalışmalardan, ÖİO'da ele alınan problemlere çözüm verebilecek durumda olanları tasarıma dahil edilmiş, hatta kullandıkları veritabanı yapısı dayanak noktası olarak alınmıştır. ilgili yazılım ve veritabanında gerekli yenilemeler yapılmıştır.

Sistem analizi sonucunda kullanılmakta olan ve gerçekleştirilmesi istenen hizmetler şu şekilde belirlenmiştir: Öğrencinin fakülteye girişte özlük bilgilerinin saklanması, yarıyıl kayıtlarının yapılması, yarıyıl içi ders notlarının işlenmesi, öğrencilerin tüm öğrenim süreleri boyunca belirtilen bilgilerinin kontrollü saklanması ve mezuniyet durumu hesapları için öğrenci kartonu işlemlerinin bilgisayar ortamına geçirilmesi ve tüm bu kayıtlı bilgilerin raporlanmasıdır.

Hizmetler böylece belirlendikten sonra hale hazırda bu hizmetlerin bazılarını karşılayabilen veritabanı ve uygulamalar gerek veritabanı gerekse yazılım açısından modernize edilerek ÖİÖ'ya dahil edilmiştir. Bu uygulamalar 1989 yılında yazılan ve en yeni versiyonu 1991 kış yarıyılında kullanılmaya başlanan Öğrenci Kayıt ve Ders Notu işleme programları ve ilgili veritabanı yapısı ve dosyalarıdır.

ÖİÖ'nun tamamlanması için katılan hizmetler ise yukarıda bahsi geçen karton işlemleri, özlük işlemleri ve ilgili raporlama olanaklarıdır. Öğrenci servislerinde kullanılmakta olan öğrenci kartonu, öğrenci ders durumunun gözlenmesi, ders kaydı ve takibi, başarı durumu takibi ve mezuniyet durumunun incelenmesi için temel teşkil etmektedir. Özlük bilgileri olarak öğrenci hakkında sosyal bilgileri saklanmaktadır. Özellikle rektörlükçe fakültelerden istenen özlük bilgilerinin sunumu için de kolaylık sağlar.

Tezin ikinci bölümünde veritabanı sistemleri hakkında genel bilgiler verilmiş, otomasyonda da kullanılan B+Ağacı yapısı ve önemi üzerinde durulmuştur.

Üçüncü bölümde otomasyon bir dizi temel sürece bölünmüş ve bu süreçler ayrı ayrı ele alınmıştır.

Dördüncü bölümde otomasyonda veritabanını oluşturan temel dosyalar ele alınıp incelenmiştir.

Beşinci bölümde otomasyon bu kez program bazında ele alınmış ve otomasyonu gerçekleyen programların işlevleri açıklanmıştır.

Altıncı bölümde ÖfO'nun en önemli bölümünü oluşturan karton kavramının bilgisayarda simülasyonu ile ilgili yazılımın tanıtılması ve incelenmesi yer almıştır.

Son bölümde otomasyon hakkında sonuç ve yorumlar dile getirilmiştir.



BÖLÜM 2. VERİ TABANI SİSTEMLERİ

Kavram olarak veritabanı, bir organizasyonun çeşitli gereksinimlerini karşılamak için tasarlanmış, birbiriyle ilişkili verilerin paylaşılır kümesidir [1].

Veri tabanları için ilk motivasyonlar, programlarında kullandıkları verileri saklı tutmak için klasik ve sürekli bir yol bulmak isteyen programcılar tarafından verilmiştir.

İlk bilgisayarların (1. kuşak) sınırlı donanım özellikleri ve dolayısıyla az gelişmiş yazılım yetenekleri yüzünden, veri işleme amaçlı programlar ve kullanılan veri yapıları çok basit kalmış ve ihtiyaca cevap verir nitelikte olmamıştır. Dolayısıyla yazılan uygulamalar, organizasyonun tümüne yönelik olmaktan çok, belirli bazı kısımlarına hizmet veren özel uygulamalardan ileriye gidememiştir.

1960'lı yıllarda büyük bilgisayarların devreye girmesi (mainframe) ile bellek ve hız kapasitesinin artışı ve kullanılmaya başlanan doğrudan erişimli saklama aygıtları (DASD) sayesinde veritabanı kavramı daha güncel olmuş ve bu kavramı gerçeklemek için ortam bulunmuştur.

Ancak tek başına veritabanı bir işe yaramaz. Onu işlemek ve kullanmak için bazı özel yazılımlara ihtiyaç vardır. Bir veritabanını yöneten yazılıma veritabanı yönetim sistemi (VTYS) adı verilir.

Veritabanı sistemi, veritabanı (Database -DB) ve bu kavramın teknolojik gelişmeler ile pratik kazanması sonucu,

veritabanı ile ilgili temel fonksiyonları yerine getiren özel bir yazılımdan (VTYS, Database Management System - DBMS) oluşan bir sistemdir.

Veritabanı sistemi kavramını açmadan önce klasik dosya işleme sistemlerinden bahsedip, veritabanı yaklaşımı ile arasındaki farkları belirtmekte fayda vardır.

2.1. Klasik Dosya İşleme Sistemi ve Veritabanı Yaklaşımı

Veritabanı yaklaşımının önemini vurgulamak amacıyla önce klasik dosya işleme sistemlerinden bahsetmek gerekir.

2.1.1. Klasik Dosya İşleme Sistemi

Klasik dosya işleme sistemlerinde -örneğin bir organizasyona- verilecek her hizmet için ayrı birer veritabanı ve bu veritabanını yöneten ayrı birer uygulama programı tasarlanır. Uygulama açısından kullanılan veri dosyaları kümesi de bir çeşit veritabanıdır. Ancak bu veritabanı kavramı ile günümüzün modern veritabanı kavramı oldukça farklıdır. Uygulamaların ve kullanacakları dosyaların belirlenmesi, geliştirilmesi ve tasarımı için genel bir plan ve yöntem yoktur. Her organizasyon kendi çözümünü gerçekler [1].

Sonuçta, yazılan uygulamalar ancak kendi içinde bütündür ve farklı donanım ve yazılım ortamlarında gerçekleştirilmiş olabilir.

Üretilen veri dosyaları uygulamaya özgü olur. Her uygulama kendi ihtiyacı olan dosyaları kendisine özgü bir takım tanımlamalar yaparak üretir. Farklı uygulamaların aynı bilgi gereksinimi için kullandıkları -çoğu zaman farklı format ve yapıda olan- dosyalar nedeniyle gereksiz

bilgi tekrarları yapılmış ve veri saklama kaynağı (DASD) gereksiz yere işgal edilmiş olur.

Diğer bir problem ise, yine aynı hizmete destek veren çok sayıda dosya bulunması sonucu aynı tür verilerin tanımının standart olamaması ve ortak bilgilerin her dosyada güncelliği -yeterince hızlı bir şekilde- sağlayamamasıdır.

Veriler paylaşılabılır olmadığı için bazı uygulamaların kullandığı dosya sisteminde yer almayan verilere ihtiyaç duyulması halinde, verilerin bulundukları ortamlardan taşınması gerekecektir (teyp-DASD). Aynı sebep yüzünden uygulamaların raporlama imkanları kısıtlıdır ve yine bilgilerin güncelliği problemi ile karşı karşıya kalınır ve veri tutarsızlığına yol açar.

Ayrıca veritabanı sisteminde zorunluluk halinde yapılan değişikliklerin uygulamalara yansıma hızı programcılarının üretkenliğine ve becerisine bağlıdır.

Sonuç olarak klasik dosya işleme yaklaşımının başlıca dezavantajları:

- Kontrolsüz veri tekrarı,
- Veri tutarsızlığı,
- Verinin sınırlı paylaşımı veya hiç paylaşılamaması,
- Zayıf veri standartları,
- Düşük programcı üretkenliği ve programların esnek olamayışı şeklinde belirtilebilir.

2.1.2. Veritabanı Yaklaşımı

Yarıiletken teknolojisindeki gelişmeler ile kapasite ve yetenekleri artan bilgisayarlar sayesinde veritabanı sistemleri de gelişme imkanı bulmuştur.

Veritabanı bilgi kaynağını yönetmede farklı bir kavramdır. Veriler para, insan, malzeme gibi önemli ve paylaşılır bir kaynak olarak ele alınır. Kavram, verilerin paylaşılabilir olmasını ve veri kaynakları kontrolünün ayrı ayrı uygulamalara değil tek bir yönetime verilmesini öngörür. Böylece kaynak üzerinde merkezi bir kontrol sağlanmış olur [4].

Veritabanında farklı uygulamaların farklı dosyaları bulunmakla beraber, veri tanımları üzerinde bir standart vardır. Bu nedenle uygulama programları, aynı kavramı simgeleyen bilgileri aynı veri formatında görürler. Böylece veri standardı sağlanmış olur.

Veriler paylaşılabilir olduğu için bir veritabanı dosyası birden fazla uygulama programına hizmet verebilir. Bu da verilerin gereksiz tekrarını optimize eder, veri tutarsızlıklarını önler ve verilerin bütünlüğünü korur [2].

Klasik dosya işleme sisteminin yukarıda açıklanan dezavantajları, veritabanı yaklaşımında söz konusu değildir.

Bununla beraber bu yaklaşımın sakıncalı yanları da bulunmaktadır. Veritabanı sistemini kuracak, yönetecek, bakımını yapacak eğitilmiş ek personele ve kaliteli yazılımlara ihtiyaç vardır. Veritabanı organizasyonu büyük çaplı bir iş olduğundan, ilerde çıkabilecek çelişkileri önlemek açısından, tasarımdan önce isteklerin kesinlikle net olarak belirlenmesi gereklidir. Aksi halde yeniden yapılanma, klasik dosya yaklaşımı tekniklerine göre daha

ağır bir yük getirir [2].

2.2. Veritabanı Sistemleri

Veritabanı, verilerle birlikte, aralarındaki ilişkilerin de saklandığı kayıt saklama sistemidir [1][2].

Veritabanı Yönetim Sistemi, veritabanı ile ilgili kullanıcı isteklerini (dosya yaratma ve silme; dosyalara kayıt ekleme, silme ve güncelleme) yerine getiren yazılımdır.

VTYS ile çok kullanıcıli ortamlarda, verilere eş zamanlı erişmek mümkündür. Olası hata durumlarında kurtarma (recovery), veritabanında yapılan değişiklikleri kayıt etme (logging) ve veritabanını arşivleme işlemi veritabanı yönetim sisteminin tipik fonksiyonlarıdır.

Veritabanı Sistemi, yukarıda belirtilen iki unsurun oluşturduğu bir bütündür [2].

Genel olarak bir veritabanı sisteminin **dört temel elemanı** vardır. Bunlar veriler, donanım, yazılım ve kullanıcılarıdır.

2.2.1. Veriler

Günümüzde veritabanı sistemleri basit PC'lerden büyük bilgisayarlara kadar her türlü ortamda geliştirilmiştir. Genel olarak bu ortamlar tek kullanıcıli ve çok kullanıcıli olmak üzere ikiye ayrılır. Bir veritabanı sistemindeki verilerin sağlanması gereken temel iki özellik **bütünlük** ve **paylaşırlılıktır** [1][2][3].

Bütünlük özelliği her iki sınıf ortam için de

gereklidir. Bu özelliğe göre veritabanı dosyalardan, dosyalar kayıtlardan ve kayıtlar da alanlardan oluşur. Dosyalar arası veri tekrarları optimaldir. Böylece aynı cins verilerin gereksiz yere birden fazla dosyada tutulması ve veritabanının büyümesi önlenmiş olur ve veriler kendisini kullanan tüm uygulamalara aynı formda sunulmuş olur.

Paylaşılabilirlik ancak çok kullanıcıli ortamlarda anlamlıdır ve veritabanındaki farklı veya aynı veri bloklarına farklı kullanıcıların farklı amaçlarla erişmesine olanak tanır.

Daha genel bir ifade ile veritabanı sistemi tek bir dosyalar kümesi ve dosyalar da kayıtlar kümesi olduğuna göre (bütünlük özelliği) herhangi bir kullanıcı, herhangi bir anda veritabanının herhangi bir alt kümesine erişebilir ve eriştiği blok diğer kullanıcıların çalışma alanları ile çakışabilir (paylaşılabilirlik özelliği).

Veritabanındaki verilerin bir özelliği de kalıcı olmalarıdır. Kalıcı veri, veritabanı içine yerleşmiş ve onaylanmış olan veridir [2].

Veritabanı ile ilgili yapılan her işlemde veriler kullanılır. Bunların bir kısmı veritabanının kendi kontrolleri için kullandığı -anahtar kelimeler türünden- veriler, bir kısmı da özellikle güncelleme işlemlerinde kullanılan kalıcı veri adayı olan geçici verilerdir. Geçici sınıfına giren veriler, veritabanı yönetim sistemince bir hata durumu ile karşılaşılmadığı sonlanma durumunda veritabanı ortamında kalıcı veri sıfatını alır.

2.2.2. Donanım

Veritabanı sisteminin, üzerinde işleyeceği bir donanıma ihtiyacı vardır. En basit donanım, bilgilerin sürekli olarak saklanabileceği bir ikincil saklama ortamı (manyetik disk, CD), veritabanı sistemi yazılımının üzerinde çalışacağı işlemci ve ona bağlı hızlı (yarı iletken) bellek ünitelerinden oluşur.

2.2.3. Yazılım

Fiziksel veritabanı ile kullanıcıların arasında arayüz görevi işlevi bir yazılım katmanı bulunmaktadır. Bu yazılıma, veritabanı yönetim sistemi -VTYS- (database management system -DBMS) veya veritabanı yöneticisi -VTY- (database manager -DB manager) adı verilir.

Yazılım, çok kullanıcıli ortamlarda verilere eş zamanlı erişimleri sağlar. Olası hata durumlarında kurtarma (recovery), veritabanında yapılan değişiklikleri kayıt etme (logging) ve veritabanını arşivleme işlemi veritabanı yönetim sisteminin tipik fonksiyonlarıdır.

Kullanıcıların isteği doğrultusunda bir takım dosya ve kayıt işlemlerini yerine getirirken, donanım seviyesi detaylarını mümkün olduğunca yalıtır.

Veritabanı yönetim sistemi, aşağıdaki 3 alt görevi kullanarak hizmetlerini yerine getirir:

- Veritabanında yeni dosyalar yaratmak. VTYS bunu DDL (Data Definition Language) olanağı ile sağlar.

- Var olan dosyalar üzerinde kayıt ekleme, silme ve güncelleme yapmak. VTYS bu fonksiyonları DML (Database Manipulation Language) ile yerine getirir.

- Veritabanı ve dosyaları ile ilgili sorgulamalar yapmak. VTYS bu fonksiyonunu ise DQL (database query language) ile yürütür [3].

2.2.4. Kullanıcılar

Kullanıcılar veritabanından bilgi alma, ekleme, silme ve güncelleme işlemleri için yararlanırlar.

Bir veritabanı sisteminden faydalanacak olan kullanıcılar 3 sınıfta toplanır [2].

İlk sınıfta uygulama programcıları yer alır. Bu kullanıcılar, veritabanı yönetim sisteminin veriye erişim, sorgulama ve güncelleme işlemlerini başka bir programlama dili (host language) içinden (embedded) yürüterek gerçekleştirirler. Yazılan programlar -büyük sistemler için- yığın işlem (batch) veya çevrim-içi (online) olarak çalışabilir.

Uç kullanıcılar olarak adlandırılan ikinci sınıfta, veritabanı sisteminin sunduğu terminalden etkileşimli erişim olanağını kullanarak veritabanına erişen kullanıcılar yer alır. Buna örnek olarak IBM ana bilgisayarlarında çalışan SQL/DS veritabanı sisteminin ISQL olanağı verilebilir [2]. Bu tür olanaklarda ya komut süren (command driven form) ya da menü süren (menu driven form) kullanıcı arayüzleri kullanılır.

Son sınıf kullanıcıları veritabanı ile ilgili teknik ve yönetim seviyesi görevlileridir. Bunlardan ilki **Veritabanı yöneticileridir** (Database Administrators -DBA). Veritabanı yöneticileri, veri kaynaklarının yönetimi, yeni veri kaynaklarının açılması, güvenlik denetimi ve veritabanı sistemi performansının gözlenmesi ve arttırılması işlerini üstlenirler. Yine bu sınıfa dahil

olan ve **Veri yöneticisi** (Data Administrator -DA) adı verilen bir başka tür sorumlu ise, organizasyonun yönetim düzeyinde veritabanında hangi tür verilerin bulunması gerektiğine karar veren ve güvenlik politikalarını belirleyen (kullanıcı veya grup bazında veritabanına erişim yetkilerinin dağıtılması) kişi ya da kimselerden oluşur. Bu işlemlerin gerçekleşmesi ile ilgili teknik bilgilere ihtiyaçları yoktur. Tüm teknik destek, veritabanı yöneticilerine aittir.

2.3. Veritabanı Modelleri

Veritabanı yönetim sisteminin işlevliği ve performansı, üzerinde çalıştığı veri yapısının düzenleniş tekniğine -verilerin organizasyonuna- bağlıdır. Veri modelleri, veritabanındaki bilgi parçalarının nasıl ayrıştırılacağını ve erişileceğini belirler. Dolayısıyla bilginin kullanıcı seviyesine yansımaları, aranması işlemleriyle yakından ilgilidir.

Hangi veritabanı modelinin kullanılacağı uygulamaya göre değişir. Örneğin grafik uygulamalar için ilişkisel veritabanı modelinin kullanılması, uygulama performansının düşmesine neden olur. Tüm uygulamalar optimal performans veren bir model yoktur. Talebi karşılayacak en uygun model seçilmelidir [3].

Bilgi, veri yapıları ile gösterilir. Temel veri yapıları, kayıt (record) ve alan (field) veya özellik (attribute) tir.

Bir kayıt, birbirleri ile ilişkili alanların oluşturduğu kümedir. Mantıksal düzeyde (dosya) kayıtlar temel veri saklama birimleridir.

Alanlar elemanter veri birimleridir. Bir alan hem

imla hem de anlam bakımından tanımlanabilir. Sözdizim açısından alan, belirli bir tipte (karakter, sayısal) taşınan bilgidir. Semantik açıdan ise alanla belirtilen kavrama karşılık gelen bilginin taşıdığı anlamdır [2][5].

Her veri modelinin temelinde bu kayıt ve alanlar yatar. Bu alanlar sayesinde varlığı ve özellikleri olan nesnelerin veri modelleri oluşturulabilir.

Başlıca üç veritabanı modeli vardır: hiyerarşik, ağ ve ilişkisel model.

2.3.1. Hiyerarşik Model

Bu modelde veritabanında bir hiyerarşi tanımlanır. Hiyerarşide birbirleriyle ilişkili yapılar ebeveyn-çocuk bağıntısı ile bağlıdır. Dolayısıyla birbirleriyle ilişkili kayıtlar bir hiyerarşi oluşturur.

Hiyerarşide kök kayıt hariç her yapının bir ebeveyn yapısı (parent) ve her ebeveyn yapının çocuk yapısı (child) bulunur. Ebeveyn-çocuk bağlantısı 1:1 veya 1:M şeklinde tanımlanabilir, dolayısıyla bir ebeveynin birden fazla çocuk yapısı olabilir. Bu görünümüyle ağaç yapılarına benzerler. Bu nedenle hiyerarşide yer alan her bilgi yapısına düğüm veya segment denir.

Hiyerarşideki temel işlem ağaçta arama (tree search) işlemidir. Hiyerarşik veritabanına yapılan bir sorgulama ile ağaç dolaşılır ve sorgulamayı tatmin eden düğümün sonuçları iletilir.

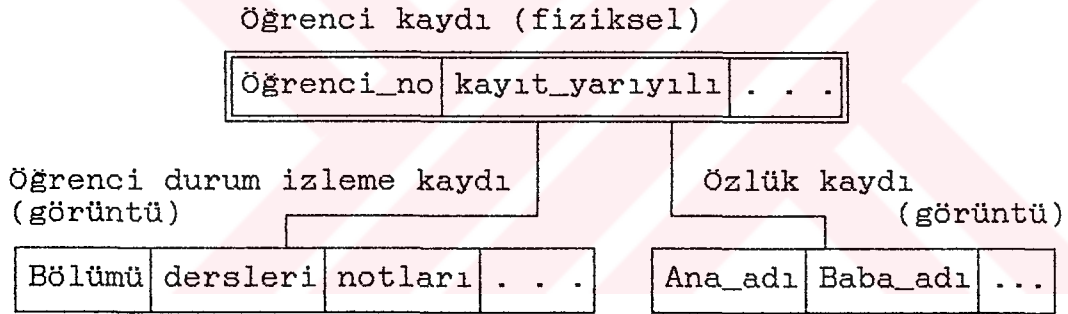
Kökteki yapı gerçek fiziksel kayıttır. Diğer yapılar ise görüntü kayıtlardır (virtual records). Bir görüntü kayıt, ilgili fiziksel kaydın kendisi yerine, o kayda bir işaretçi ile tanımlanır. Hiyerarşi bu işaretçiler

sayesinde kurulur (Şekil-2.1).

Şekildeki gibi bir veritabanında görüldüğü üzere fiziksel öğrenci kaydından diğer fiziksel kayıtlara yapılan bağlar sayesinde görüntü kayıtlar üretilmiş olur.

Hiyerarşik veritabanı modelinde veriye erişim son derece hızlı olmasına karşın bu hiyerarşiyi, kayıt ekleme/silme işlemleri nedeniyle, sürekli kılmak için esnek güncelleme algoritmalarına ihtiyaç vardır.

System 2000 ve IMS (Information Management System) veritabanı yönetim sistemleri hiyerarşik modelde tasarlanmıştır [2].



Şekil 2.1 Hiyerarşik veritabanı modeli örneği.

2.3.2. Ağ Model

Bu tip veritabanı yönetim sisteminde, hiyerarşik veritabanı yönetim sisteminin yapısına ek olarak, esnekliği arttırmak için, iki yönlü bağlantılar yapılmıştır. En genel halde herhangi iki düğüm arasında en az bir yol bulunabilir ve yapı içinde bir dolaşı tanımlanabilir (bağlantılar M:N şeklindedir) [2].

Modelde, ebeveyn/çocuk düğüm ayırımı yoktur; her düğüm hem ebeveyn hem de çocuk düğüm olabilir. Bu hiyerarşi ancak bağlantı düzeyinde vardır.

Her bağlantı (link) bir ebeveyn ve bir çocuk kaydını birbirine bağlayabilir. Belirli bir L bağlantı kümesinin A tipi bir kayıttan, B tipi bir kayda olduğunu düşünürsek; L'deki bağlantılardan, A'nın bir tek örneğine bağlı olanlar için A'daki o kayıt ebeveyn; L'deki bağlantılara en çok bir kez bağlı olan B kayıtları çocuk olabilirler (klasik ebeveyn-çocuk ilişkisi kuralı).

Dolayısıyla hiyerarşik model, ağ modelin özel bir şeklidir. ihtiyaca göre çeşitli sayıda bağ kurulabilir.

Veriye erişim çok hızlı olmakla beraber, güncelleme fonksiyonunun karmaşık ve vakit alıcı olacağı açıktır.

Ağ model veritabanı yönetim sistemine örnek olarak IDMS (Integrated Database Management System) verilebilir.

2.3.3. İlişkisel Model

Bağıntı cebrinin iyi tanımlı (well-defined) özellikleri üzerine kurulu bir modeldir [2][3]. Veritabanı kullanıcıya normalize edilmiş (veri tekrarları olmayan) tablolar kümesi halinde sunulur.

Veritabanında her tablo bir bağıntıdır. Tablo, satır ve sütunlardan oluşur. Her satır bir kayda ve her sütun kayıt içindeki bir alana karşılık gelir. N alanlı bir tablo, N'li bir bağıntıyı gösterir. Tablo 2.1 de bazı bağıntı cebri terimleri ve karşılık gelen ilişkisel model veritabanı terimleri yer almaktadır.

İlişkisel veritabanı yaklaşımında bir tabloda olması gereken özellikler:

- Tabloyu oluşturan satırlardaki alan değerleri atomiktir. Alanlar, hiçbir şekilde karmaşık veri tipi -

örneğin işaretçi- olamaz.

- Tabloyu oluşturan satır veya sütunlar tekrarlanamaz.

Tablolar (bağıntılar) arasındaki temel işlemler seçme, yansıma, küme çarpımı, küme birleşimi, küme farkı işlemleridir. Önemli diğer bir işlem de kaynaşma (join) işlemidir ve küme çarpım ile seçme işlemlerinden oluşur. Bu beş işlemten seçme ve yansıma işlemleri birli, diğerleri ikili işlemlerdir. Sözü edilen işlemler kapalılık özelliği gösterir. Bu nedenle bu işlemlerin uygulandığı tablolardan yeni tablolar üretilir.

Tablo 2.1 ilişkisel modelde veri yapısı terminolojisi [4].

<u>Formal Bağıntı Terminolojisi</u>	<u>Veritabanı Terminolojisi</u>
bağıntı	tablo
tuple	satır (kayıt)
kardinalite	satır sayısı
özellik	sütun (alan)
derece	sütun sayısı
domen	alan değerleri kümesi

2.4. Veritabanı Mimarisi

ANSI/SPARC grubunun standart haline gelmiş olan önerisine göre veritabanı mimarisi **üç seviyeye** bölünür; iç (fiziksel) seviye, **kavramsal** seviye ve dış (mantıksal) seviye [1][2][3][4].

İç seviye, fiziksel veritabanına en yakın seviyedir ve verilerin fiziksel saklanması şekilleri ve yöntemleri ile ilgilenir.

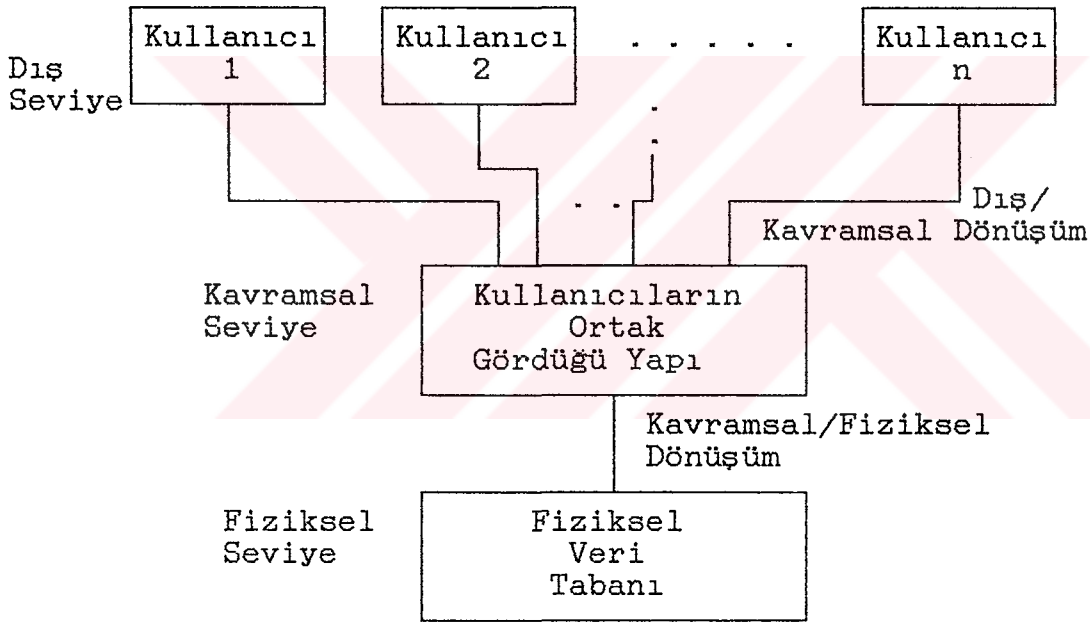
Dış seviye, kullanıcılara yakındır ve veritabanının

farklı kullanıcılara görüntülenme şekli ile ilgilidir.

Kavramsal seviye, bu iki seviye arasında ara birimdir.

iç seviye veritabanını olduğu biçimi ile görürken, kavramsal seviye bütün kullanıcılar için aynı şekilde görüntülenebilen ortak veri yapısını ve dış seviye ise kullanıcıların sorgulama şekline bağlı olan altyapıyı görür.

Mimari, aşağıdaki şema ile özetlenebilir:



Şekil 2.2 ANSI/SPARC'ın 3 seviyeli Veritabanı Mimarisi.

2.4.1. Dış (Mantıksal) Seviye

Dış seviye kullanıcı seviyesidir. Kullanıcı bir uygulama programı veya terminalden ekileşimli çalışan bir kullanıcı olabilir. Kullanıcılar veritabanına farklı şekillerde ancak VTYS'nin sunduğu diller ile (DDL, DML, DQL) erişirler. Örneğin uygulama programları (COBOL, PL/1, C), derleyicilerinin dili içinde veritabanı

yönetim sistemi sorgulama dilini kullanırken, etkileşimli kullanıcılar da terminalden sorgulama komutlarını yorumlayan dili kullanabilirler. Bazı VTYS'lerinde farklı uygulama şekilleri için (program, etkileşimli mesajlaşma) farklı diller kullanılabilir.

Önemli olan bu dillerin bir **veri alt dilini** içeriyor olmasıdır. Veri alt dili (DSL), veritabanı nesneleri ve işlemleri ile ilgili bir dildir. Bu dil bir uygulamanın (host language) içinde (embedded) kullanılabilir. Bazı sistemlerde bu iki dil kullanıcı düzeyinde ayırt edilemez. Bu durumda bu iki dil birbirine **sıkı bağlıdır** denir. Eğer bu ayırım kullanıcı düzeyinde yapılabiliyorsa iki dil **gevşek bağlıdır**.

Temelde veri alt dili iki kısımdan oluşur: **veri tanım dili (DDL)** ve **veritabanı işleme dili (DML)**. DDL veritabanı nesnelerinin (dosya yapıları) tanımlanmasını sağlar. DML, tanımlı bu nesneler üzerinde daha önce bahsedilen güncelleme işlemlerini destekler [1][2][3].

Bu seviyede ayrı ayrı kullanıcıların veritabanını gördüğü şekil bir (ANSI/SPARC terminolojisine göre) birer **dış görüntüdür** (external view). Kullanıcının veri alt dili **dış kayıtlar** (external records) ile tanımlıdır. Dış kaydın, saklı kayıtla bire bir aynı olması gerekmez ve genellikle onun bir alt kümesidir. Bir dış görüntü, o görüntüde çeşitli tiplerde dış kayıt tanımlarını içeren bir **dış şema** (external schema) ile tanımlıdır ve kullanıcının veri alt dilinin DDL kısmı ile yazılır. Bu yüzden bu seviyedeki DDL, **dış DDL** (external DDL) olarak bilinir [2].

2.4.2. Kavramsal Seviye

Kavramsal seviyede, fiziksel veritabanının kontrol bilgilerinden arınmış ve sadece kullanıcıların yarattığı

veri tiplerinden oluşan bir veritabanı görüntüsü vardır . Aynı zamanda dış seviyede farklı kullanıcılar tarafından görülen formlardan daha genel bir yapıda olup tüm veritabanının görüntüsü yer alır.

Kavramsal görüntü, tüm veritabanının sade bir görüntüsüdür ve bu görüntüyü **kavramsal kayıt** tanımlarını kullanarak **kavramsal şema** tarafından tanımlanır. Aslında bu görüntü basit kayıt tanımlarına benzer yapıların oluşturduğu bir kümedir [1][2]. Bu özelliği nedeniyle veritabanındaki verilerin güvenliği ve bütünlüğünden sorumludur [2][3].

Kavramsal şema **kavramsal DDL** ile yazılır. **Veri bağımsızlığını** sağlamak için kavramsal DDL tanımlarının yalnızca veri içeriğine bağlı olması ve veriyi saklama tekniklerinden bağımsız olması gereklidir. Bu nedenle kavramsal şemada saklı alanlara, saklı kayıt dizilerine, indeksleme tekniklerine, işaretçilere ve diğer veri saklama ve erişim detaylarına referans verilmemelidir.

2.4.3. iç Seviye

Mimarideki üçüncü düzey iç seviyedir ve fiziksel veritabanının düşük seviyeli gösterimidir. Çeşitli tiplerdeki **iç kayıtlardan** oluşur. iç kayıt, ANSI/SPARC terminolojisinde **saklı kayıt** için kullanılan bir terimdir. Bu nedenle gerçek fiziksel kayıtlardan farklı bir yapıda olup silindir, iz gibi aygıta (disk) bağlı parametrelerle ilgilenmez. Temelde bu görüntü sonsuz lineer bir adres uzayı olarak varsayılır. Fiziksel adreslere olan dönüşümler aygıta fazlasıyla bağlı olduğundan genel mimari içine alınmaz.

iç görüntü bir **iç şema** ile tanımlanır. Burada iç şema yalnızca çeşitli tiplerdeki **iç kayıtlardan** oluşmakla kalmaz

aynı zamanda hangi indekslerin ve indeks anahtarlarının var olduğu, verilerin saklı kayıttaki diziliş şekilleri gibi detayları da içine alır. iç şema bir iç DDL ile yazılır. Aslında burada sözü edilen iç görüntü saklı veritabanı, iç şema ise saklı kayıtların yapısıdır [2].

Aşağıda ANSI/SPARC mimarisi ile ilgili bir örnek gösterilmiştir (Şekil 2.3).

(Dış görüntü 1)	(Dış görüntü 2)
PL/1 DCL 1 KAYITP, 2 DIZIP CHAR(6), 2 SAYI FIXED BIN (31), 2 EK CHAR(10),	COBOL 01 KAYITC. 02 DIZIC PIC X(6). 02 SAYIC PIC 9(4).
(Kavramsal görüntü) KAYITTIP DIZITIP SAYITIP EKTIP CHARACTER (6) SHORTINT CHARACTER (10)	
(iç görüntü) SAKLI_KAYIT PREFIX DIZIS SAYIS EKS LENGTH=23 TYPE=Sekizli(3), OFFSET=0 TYPE=Sekizli(6), OFFSET=3, TYPE=WORD, OFFSET=9, TYPE=Sekizli(10), OFFSET=13	

Şekil 2.3 ANSI/SPARC'ın 3 seviyeli mimari örneği.

Diskte verinin düzenleniş şekline **saklama yapısı** denir. Saklama yapısı veritabanına erişim performansı ile yakından ilgilidir. Veritabanı sistemlerinde kullanılabilen çok çeşitli saklama yapıları vardır ve bunların her biri farklı performans özelliğine sahip olup, bazıları bir grup uygulama için elverişli iken diğer bir kısmı için aynı performans söz konusu olmayabilir.

Bütün uygulamalar için optimal performans veren bir yapı yoktur. Bu yüzden iyi bir veritabanı sistemi birden fazla model destekleyebilmelidir. Bir veritabanı için en uygun saklama yapısını seçme işlemine fiziksel veritabanı tasarımı adı verilir [2].

2.4.4. Dönüşümler

Mimaride biri dış ve kavramsal seviyeler ve diğeri de kavramsal ve iç seviyeler arasında olmak üzere iki farklı dönüşüm bulunmaktadır.

Kavramsal-iç seviyeleri dönüşümü, kavramsal görüntü ve saklı veritabanı arasında bir bağlantı kurar; kavramsal kayıt ve alanların iç seviyede nasıl tanımlanacağını belirler. Saklı veritabanı tanımında bir değişiklik yapıldığında (saklı kayıtların yapısı değiştirildiğinde) kavramsal şemanın olduğu gibi kalması için, bu dönüşümde de uygun bir değişiklik gerekir.

Dış-kavramsal dönüşüm, belirli bir dış görüntü ve kavramsal görüntü arasında tanımlandığından her bir dış görüntü için ayrı ayrı vardır. Bu dönüşümlerden herbiri bir dış seviyedeki görüntünün iç seviyeden nasıl indirgeneceği ile ilgilidir.

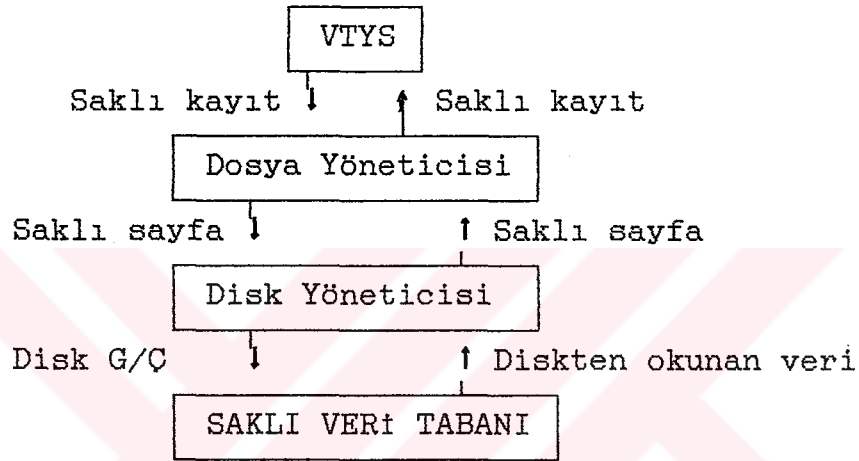
2.5. Veritabanına Erişim

Bir veritabanı sisteminde veriye erişim yolu katmanlı bir yapıya sahiptir. Katman yapıları detayda sistemden sisteme değişebilir. Ancak fonksiyonel olarak her sistemde benzerdir. Bu katmanlar sırasıyla VTYS (DBMS), Dosya yöneticisi ve Disk yöneticisidir [2].

VTYS, dosya yöneticisi sayesinde veri tabanını

saklı kayıtların bir kümesi, dosya yöneticisi ise disk yöneticisi sayesinde veritabanını sayfalar kümesi olarak görür. Disk yöneticisi ise fiziksel veritabanını olduğu gibi görmektedir.

Aşağıdaki şekilde ilgili katmanlar ve aralarındaki ilişkiler gösterilmiştir (Şekil 2.4)



Şekil 2.4 Veritabanına erişim.

2.5.1. Disk Yöneticisi

Disk yöneticisi disk üzerinde istenilen sayfayı saptar ve gerekli giriş-çıkış işlemlerini yürütür. Bu nedenle işletim sisteminin bir parçasıdır. Fiziksel disk adresleri ile çalışır. Dosya yöneticisi belirli bir p sayfası için sorgulama yaptığında, disk yöneticisinin p sayfasının diskteki tam adresini bilmesi gerekir. Ancak disk yöneticisinin kullanıcısı olan dosya yöneticisinin bu tür bilgilere ihtiyacı yoktur, sadece sayfaların mantıksal kümesi ile ilgilidir. Her sayfanın boyu sabit olup (1K, 2K, 4K..) sayfa kümeleri içinde bir **sayfa küme kimliği** ile tanımlıdır ve küme içindeki her sayfanın, benzer şekilde bir sayfa kimliği (sayfa numarası) vardır.

Sayfa numaraları ve fiziksel disk adresleri arasındaki dönüşüm disk yöneticisi tarafından yönetilir. Bu tür bir düzenlemenin temel avantajı, tüm aygıta bağlı işlemlerin tek bir katman içine alınmış olması ve üst katmanlara ağıttan bağımsız olma özelliği vermesidir. Bu nedenle dosya yöneticisi diski sayfalar kümesi olarak görür.

Tüm veritabanı iki tür sayfa kümesine ayrılır: boş sayfalar kümesi ve dolu sayfalar kümesi. Boş sayfaların tahsisi ve geri iadesi disk yöneticisi tarafından sağlanır.

Disk yöneticisinin sayfa kümeleri üzerinde yürüttüğü işlemler şöyle sıralanabilir:

- s sayfa kümesinden p sayfasına erişmek,
- s sayfa kümesindeki p sayfasını değiştirmek,
- s kümesine, boş sayfalar kümesinden, yeni sayfa eklemek ve sayfaya kimlik (p) vermek,
- s kümesinden p sayfasını silmek ve boş sayfalar kümesine eklemek.

2.5.2. Dosya Yöneticisi

Dosya yöneticisi, disk yöneticisinin yeteneklerini kullanarak, disk ortamını kendi kullanıcısına (VTYS-DBMS) saklı dosyalar kümesi olarak gösterir. Her saklı dosya, saklı kayıtlardan oluşur. Her sayfa kümesi bir ya da daha fazla saklı dosyadan oluşabilir. Dosya yöneticisi sayesinde VTYS, sayfa kümeleri ile ilgilenmez ancak farklı dosyaların paylaştığı sayfa kümelerini ve farklı cinsten kayıtlar tarafından paylaşılan sayfaları bilmesi gerekir.

Her saklı dosya, bir dosya kimliği (dosya adı) ile

tanımlıdır. Aynı sayfa kümesi içinde, aynı kimliğe sahip birden fazla dosya bulunamaz. Her saklı kayıt bir **kayıt kimliği** (kayıt numarası) ile tanımlıdır ve bu kimlik en azından ilgili sayfa içinde tektir.

Bazı sistemlerde dosya yöneticisi işletim sisteminin, bazılarında ise VTYS'nin bir parçasıdır.

Dosya yöneticisinin saklı dosyalar üzerinde uygulayabileceği işlemler:

- f dosyasından r kaydını al,
- f dosyasındaki r kaydını değiştir,
- f dosyasına yeni kayıt ekle, r kayıt kimliğini üret,
- f dosyasından r kaydını çıkar,
- yeni f dosyası yarat,
- var olan bir f dosyasını sil.

2.6. Erişim Performansı

Veritabanına erişim performansını arttırmak için çeşitli yöntemler vardır. Bunlardan bazıları bloklama (clustering), sayfa ve kayıt yönetimi ve indekslemedir.

2.6.1 Bloklama (Clustering)

Birbirleriyle mantıksal ilişkili olan kayıtların fiziksel olarak ardışıl alanlara yerleştirilmesi, veritabanı G/Ç işlemlerinin sayısını azaltan faktörlerden birisidir. Örneğin ardışıl olarak erişeceğimizi

varsaydığımız r1 ve r2 kayıtları sırasıyla p1 ve p2 sayfalarında bulunsun ve r1 ve r2 kayıtları sırayla okunmak istensin. P1 ve p2 aynı sayfa ise r2 kaydını okumak için ayrıca ek bir disk erişimi gerekmeyecektir (pratikte diskten okuma ve diske yazma işlemleri sayfa kümeleri bazında yapılır). Çünkü aranan kaydın (r2) bulunduğu sayfa zaten bellekte bulunmaktadır. P1 ve p2 ayrı sayfalar ise, sayfaların birbirine olan yakınlık derecesine göre (en iyi ihtimalle bitişik olabilirler) r2 kaydını okumak için gereken zaman azalacaktır.

iki tür bloklama vardır: **dosya içi** ve **dosya dışı**.

Dosya içi (inter file) bloklamada belirli bir dosyaya ait kayıtlar fiziksel olarak birbirlerine yakın yerleştirilirler. Mantıksal olarak ilişkili kayıtlar farklı dosyalardan ise (veri ve indeks dosyaları gibi), bu dosyalara ait kayıtlar için yapılan bloklama dosyalar arası (intra file) bloklama adını alır. Ne zaman ve hangi tür bloklama yapılacağına VTYS karar verir [1][2].

2.6.2. Sayfa Yönetimi

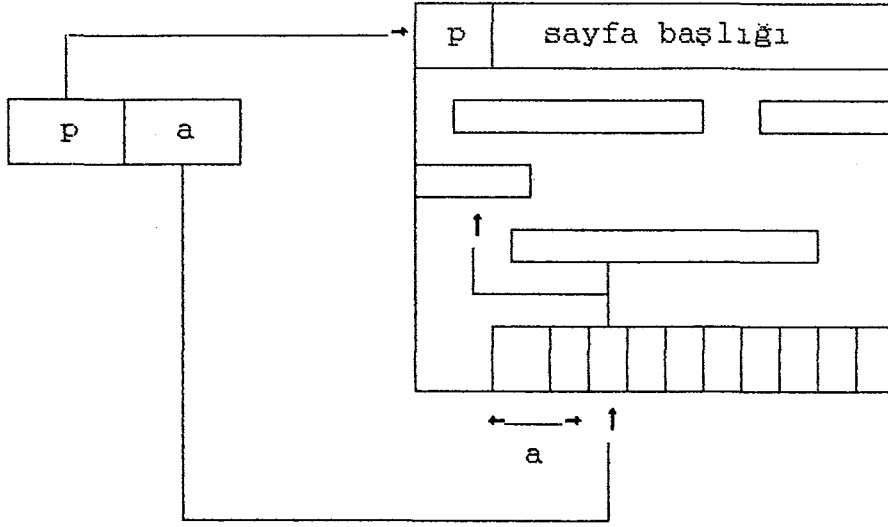
Daha önce açıklandığı gibi disk yöneticisinin ana görevi dosya yöneticisini fiziksel disk G/Ç detaylarından arındırmak ve bu olayı sayfa G/Ç olarak düşünmesini sağlamaktır. Disk yöneticisinin üstlendiği bu fonksiyona sayfa yönetimi (page management) denir.

Pratikte kullanılmış bir veritabanı ortamına baktığımızda, sayısız sayfa tahsis/iade işlemleri ile bir süre sonra diskteki birbiriyle alakalı çoğu sayfanın diskte ayrı yerlerde bulunduğu ve mantıksal ilişkinin korunamadığı görülür. Bu sebeple mantıksal ilişkiyi korumak için sayfalar işaretçiler ile birbirine bağlanır. Her sayfada bir sayfa başlığı vardır. Burada sayfa numarası, sayfanın

ait olduđu sayfa kümesi kimliğı gibi bir takım kontrol bilgilerinin yanında, mantıksal olarak kendisini izleyen sayfanın diskteki fiziksel adresine bir işaretçi yer alır. Böylece her sayfa kümesi belirlenmiş olur. Ayrıca veritabanında hangi sayfa kümelerinin bulunduđu ve ilk sayfalarının fiziksel adresleri diskte ilk iz/silindirdeki ilk sayfada yer alır.

2.6.3. Kayıt Yönetimi

Dosya yöneticisinin fonksiyonu kendi üst seviyesi (VTYS) ile alt seviyesini (disk yöneticisi) birbirinden yalıtmak ve VTYS'ye disk ortamını sayfalardan bağımsız olarak saklı dosya ve kayıt kümesi biçiminde sunmaktır. Dosya yöneticisinin bu fonksiyonuna kayıt yönetimi (record management) denir. Aynı sayfa üzerindeki mantıksal olarak ilişkili kayıtları fiziksel olarak bitişik adreslere koymak için sayfa içi öteleme özelliğı kullanılır. Gerekli ötelemeler ile kaydın sayfa içindeki uygun yeri bulunur. Kayıt numaraları kaydın içinde bulunduđu sayfa numarası ile sayfa içindeki offset değerinden oluşur ve iyi bir veritabanı sisteminde bu kayıt numaralarının sıkca (idealde hiçbir zaman) değişmemesi istenir. Bunu sağlamanın bir yolu kayıttan sayfa içindeki adresine bir tablo yardımı ile erişmektir. Bu durumda kaydın offset alan değeri doğrudan sayfa içi offseti yerine, sayfa içindeki bir tablonun offsetini gösterir. Sayfa içi ötelemeler bu tablonun ilgili içeriklerinin değiştirilmesi ile sağlanır (Şekil 2.5). Böylelikle kayıt adresi değiştirilmeden bloklama yapılmış olur.



Şekil 2.5 Bir kaydın sayfa içinde ötelenişi.

2.6.4. indeksleme

Veritabanı sistemlerinde en önemli amaçlardan birisi verilere mümkün olduğunca kısa sürede erişmektir. Bloklama olayı erişim performansını (özellikle ardışıl erişimler için) arttırıyor olsa da özellikle büyük çapta veri depolayan sistemlerde saklanan dosya ve kayıt bloklarının sayısı arttığından, herhangi bir anda istenilen bir kayda erişim için yapılan disk G/Ç sayısını arttıracığından yavaşlayacaktır.

Veri dosyalarına erişimi hızlandırmak için veri dosyası kayıtlarını belirli bir düzene göre doğrudan işaretleyebilen bir ek dosya kullanılır. Bu dosyaya **indeks dosyası** denir. Basitçe bir indeks dosyası, en az iki temel alandan oluşur: Anahtar alan ve Kayıt numarası.

Anahtar alan, veri dosyası kayıtlarında bulunan alanlardan birisidir, kullanıcı tarafından belirlenir ve **indeksli alan** veya **indeks anahtarı** adını alır. **Kayıt numarası** ise veri dosyasındaki ilgili kaydın kimliğidir.

Üzerinde indeksleme yapılan veri dosyasına **indeksli dosya**, indeksli dosyanın her bir kaydına **indeksli kayıt** ve indeksli kayıt üzerindeki indeksleme yapılan alanlara da **indeksli alan** denir. Bir indeksli dosya birden fazla indeksli alana sahip olabilir. Ancak bunlardan birisi **birincil indeks alanı** diğerleri **ikincil indeks alanıdır**. Bu alanlardan oluşturulan indeks dosyaları da benzer şekilde birincil indeks ve ikincil indeks olurlar.

indeks dosyalarını kullanarak veri dosyalarına başlıca iki şekilde erişmek mümkündür: **Sıralı erişim** ve **doğrudan erişim**.

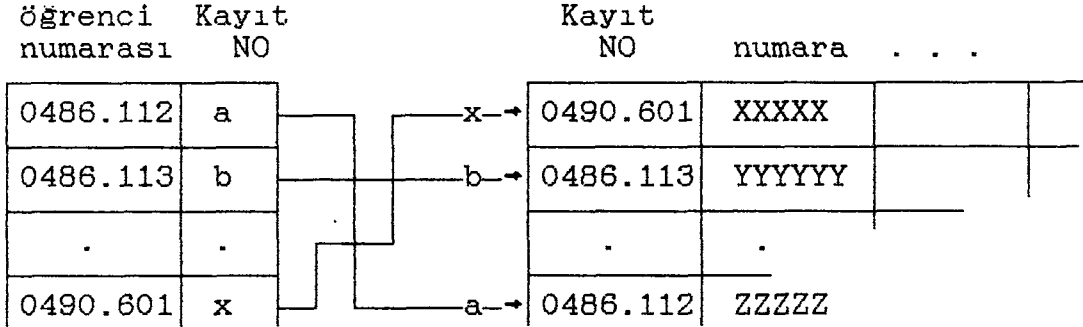
Sıralı erişim, veri dosyasına indeks alanındaki değerler dizisine göre sırayla erişmektir. Özellikle sınır sorgulamalarında (indeks alanının iki değeri arasındaki kayıtların dökümü) kolaylık sağlar. Doğrudan erişim, veri dosyasındaki istenilen özellikleri (indeks alanına göre) taşıyan kaydına doğrudan erişmek için kullanılır. Böylece kayıtların varlık testleri, veri dosyasına erişmeksizin, yalnızca indeks dosyası taranarak yapılabilir. Bir veri dosyası için birden fazla indeks oluşturulabildiği gibi, birden fazla kayıt alanının kombinasyonunu kullanarak (bir tek anahtarmış gibi) indeks oluşturulabilir (Şekil 2.6).

indeksleme yöntemi veriye erişim süresini azaltırken, güncelleme işlemlerinin yükünü arttırmış olur.

Veri dosyasındaki her kayıt için indekste bir kayıt varsa, bu indekse **yoğun indeks** (dense) adı verilir.

Numara alanına göre
indeks dosyası

Veri dosyası



Şekil 2.6 indeksleme örneği.

Eğer veri dosyasındaki bilgiler bloklanmış ise (dosyadaki kayıtların belirli bir disipline göre fiziksel dizilişi mantıksal dizilişine denk ise) indeksleme yaparken her blokun ilk veya son kayıt bilgisi dikkate alınabilir. Bu durumda indeks dosyasının boyu küçülür ancak blok içi sıralı arama yapılır. Böylece oluşturulan indeks yapısına **seyrek indeks** (nondense, sparse) denir [2].

Büyük çaplı dosyalarda yoğun indeks, indeks dosyasının boyunu büyüteceğinden, beklenen performansı veremez. Bu yoğun indeksi işaretleyen çok seviyeli sayrek indeks yapısı kurulur. indeksli bir veri dosyasında en fazla bir tane yoğun, en az bir tane de seyrek indeks bulunur. Pratikte en çok 3 seviyeli indekslemeler yer almaktadır.

Günümüzde en çok kullanılan indeksleme tekniği B-Ağacı (B-tree) yöntemidir [2][5][6].

2.6.4.1. B+ Ağacı

En çok bilinen indeksleme yöntemi B+ ağacı yöntemidir. B+ ağacı çok seviyeli ve ağaç yapılı bir indekstir. R.Bayer ve E.McCreight tarafından 60'lı yılların sonunda keşfedilmiştir [6].

indeks iki kısımdan oluşmaktadır: Sıralı küme ve indeks kümesi.

Sıralı küme, gerçek verilere tek düzeyli indeksten oluşur. Normalde yoğun indekstir, ancak veri dosyası bloklanmış durumda ise (clustering) seyrek indeks de olabilir. İndeks aynı büyüklükte gruplara ayrılır ve her grup fiziksel sırayı bozmayacak şekilde birbirlerine işaretçilerle bağlanır. Bu gruplamalar çoğunlukla sayfa bazında olur. Böylelikle sıralı küme, veri dosyasına hızlı sıralı erişim sağlar.

indeks kümesi sıralı kümeye doğrudan erişimi sağlamak içindir. İndeks kümesi ağaç yapılı bir indekstir ve asıl B-ağacı bu kısımdır. Sıralı küme ile birlikte bu yapıya genellikle B+ ağacı adı verilir.

B+ ağacının en üst düğümü (sayfa) kök, en alt düğümü ise yaprak ve aradaki düğümler de iç düğüm adını alır. Her düğümde birden fazla indeks alan değeri yer alabilir.

İç düğümlerde yer alacak maksimum indeks alanı sayısı, -tanıma bağlı olarak- ya B+ ağacının mertebesi ile doğru orantılıdır ya da mertebesine eşittir.

N mertebeli bir B+ ağaçta aşağıdaki koşullar sağlanmalıdır:

- Kök dışında bütün sayfaların (düğüm) en az n elemanı, Kök düğümün en az bir eleman olmalıdır.

- Her sayfada en fazla $2n$ eleman bulunabilir.

- Bir sayfa ya yapraktır ve başka hiçbir sayfaya referans vermez, ya da içindeki veri sayısının 1 fazlası kadar bir alt düzeyden sayfaya işaret eder.

- Bütün yapraklar aynı seviyededirler (dengeli yapı).

2.6.4.2. Otomasyonda B+ Ağacı

Öğrenci işleri otomasyonunda B+ Ağacı yapısı Borland'ın Turbo Access veritabanı tasarımı program kütüphanesi kullanılarak oluşturuldu.

Yapının temel elemanı anahtar alan, sayfa referans işaretçisi ve kayıt işaretçisinden oluşmaktadır. Anahtar alan, veri dosyasında karakter katarı olarak tanımlanan ve indeksleme yapılacak alana karşılık gelen alandır. Sayfa referansı, ağacın bellekte tutulan sayfalarına işaretçidir. Nümerik tipte tanımlanan işaretçi, işaret ettiği sayfanın numarasını içerir. Kayıt işaretçisi alanı ise anahtar alan ile ilgili kaydın veri dosyasındaki kayıt numarasını taşıyan nümerik tipte bir alandır.

Bu üç elemandan oluşan yapıya birim yapı (item) denir (Şekil 2.7). Her sayfa belirli ve aynı sayıda birim elemanlardan ve sayfa kontrol bilgilerinden oluşur (Şekil 2.8). Kontrol bilgileri olarak, sayfadaki maksimum birim eleman sayısı ve ek sayfa işaretçisi yer alır. Birim elemanlarda yer alan sayfa işaretçisi, ilgili anahtar değerinden daha büyük anahtarları içeren alt düzey sayfalara işaret ettiği için tüm anahtarlardan küçük olan bir anahtar değerinde, sayfa yapısındaki ek sayfa işaretçisinin işaret ettiği sayfa kullanılır.

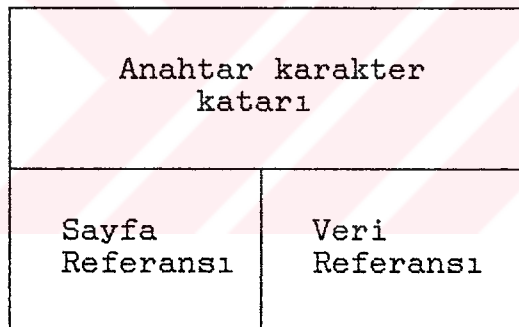
Bu yapıyı simgeleyen örnek tip tanımlamaları aşağıdaki gibi verilebilir:

```
TaKeyStr = string[MaxKeyLen];
```

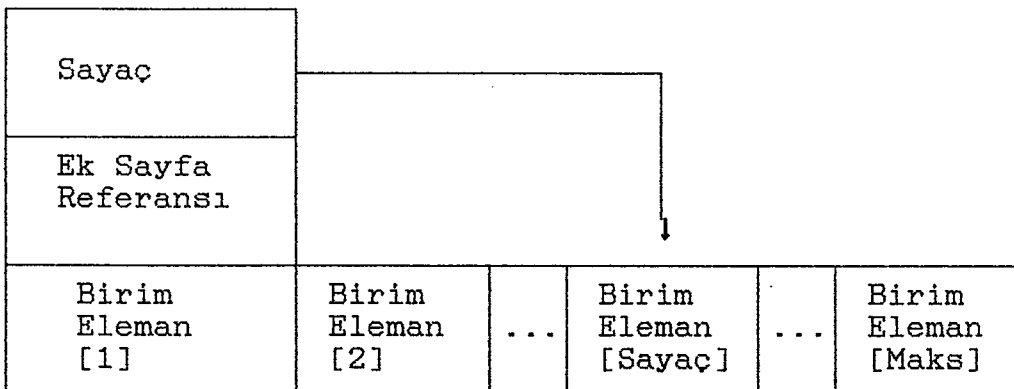
```
TaItem = record
    DataRef, PageRef : LongInt;
    Key : TaKeyStr;
end; (* Şekil 2.7 *)
```

```
TaPage = record
    ItemsOnPage : 0..PageSize;
    BckwPageRef : LongInt;
    ItemArray : array[1..PageSize] of
        TaItem;
end; (* Şekil 2.8 *)
```

```
TaPagePtr = ^TaPage;
```



Şekil 2.7 B+ Ağacında birim eleman yapısı.



Şekil 2.8 B+ Ağacında bir sayfanın yapısı.

Birim elemanlardaki anahtar alan ve sayfa referansları, altprogram kütüphanesinin AddKey alt programı ile, veri kaydı referansı ise AddRec altprogramı ile belirlenir [5].

Bir veri dosyası veya dosyaları topluluğu için gerekli optimal (en az disk erişimini sağlayan) ağaç yapısını belirlemek için aşağıdaki sabitlere ve hesaplanmış değerlere ihtiyaç vardır.

MaksVeriKaydıBoy (MaxDataRecSize): Veritabanında tanımlanan kayıt tiplerinden en büyüğünün kayıt boyu olup 18 ile 32767 arasında değişen bir tamsayıdır [5].

MaksAnahtarUzunluğu (MaxKeyLen): Tanımlanan kayıt tiplerinde indeksleme için kullanılacak olan ve karakter katarı (string) tipinde tanımlanmış alanlardan en uzun olanı [5].

Bir sınırlama da bellekte ağaç için ayrılan alanın en fazla 64 KB olmasıdır.

Tanımlanan bu sabitler ile ağaç için gerekli olan hesaplanacak parametreler ise; Sayfa Büyüklüğü (4 ile 254 sekizli arasında bir tamsayı), MaksimumAğaçYüksekliği (tamsayı), AğacınMertebesi (2 ile 127 arasında bir tamsayı) ve bellekte tutulacak sayfa sayısı (3 ile 254 arasında tamsayı) dır [5].

Maksimum kayıt boyu 7089 Sekizli ve maksimum anahtar alanı boyu 30 Sekizli olan 1000 kayıtlık bir dosyanın her 100 kayıt arama işleminde yalnız 1 kez disk erişimi yapabilmesi için aşağıdaki sonuçlar elde edilmiştir:

- Bir sayfada yer alacak anahtar sayısı = 56,
- Bellekte saklanacak sayfa sayısı = 29,

- Ağacın mertebesi = 28,
- Ağacın maksimum yüksekliği = 3.



BÖLÜM 3. ÖĞRENCİ İŞLERİ OTOMASYONUNA (ÖİO) GENEL BAKIŞ

3.1. ÖİO Kapsamı ve Amaçları

ÖİO, fakülte öğrenci bürolarında verilen temel hizmetlerin daha hızlı ve güvenilir yürütülebilmesi amacıyla hazırlanmıştır. Otomasyon kapsamında fakülte öğrencilerinin özlük bilgilerinin tutulması, yarıyıl ve ders kayıtlarının kontrollü yapılması, ders durum bilgilerinin (devam, notlar, ödevler) işlenmesi, tüm öğrenimi boyunca bu bilgilerin rafine olarak kartonda saklanması ile mezuniyet durumunun kontrolü ve tüm bu bilgilerden raporlar alınması olanakları yer almaktadır.

Bahsi geçtiği gibi bunlar temel hizmetler olduğu için ÖİO sayesinde ofis çalışanlarının işgücü kayıpları azaltılmış, insan doğasından kaynaklanan olası hataların önüne geçilmiş ve adı geçen hizmetlerin daha sağlıklı verilmesi sağlanmış olacaktır.

Belirtildiği gibi otomasyon öğrencinin fakülteye ilk girdiği günden çıkışına kadar tüm durumunu izlemeye imkan verir. Bu işlemin başarıya ulaşması için öğrenciler hakkında özlük bilgileri, yarıyıl ve ders kayıtları, ders başarı durumları, fakülteye ait kimlik, yarıyıl dersleri ve müfredat programı gibi temel bilgiler, özellikle ileride ortaya çıkabilecek çelişkili durumların önlemesi açısından, eksiksiz verilmelidir.

ÖİO amaçlanan her hizmet, bilgisayar ve insan unsurlarının birbirini tamamladığı işlemler dizisinden oluşan birer süreç olarak tanımlanmıştır.

3.2. ÖfO'da Süreçler

Süreçten kasıt belirli bir hizmeti gerçekleyen mekanizmaların tümüdür. Bu mekanizmaya -yazılım dışında- etki eden, yürümesini sağlayan diğer faktörler (insan) de dahildir.

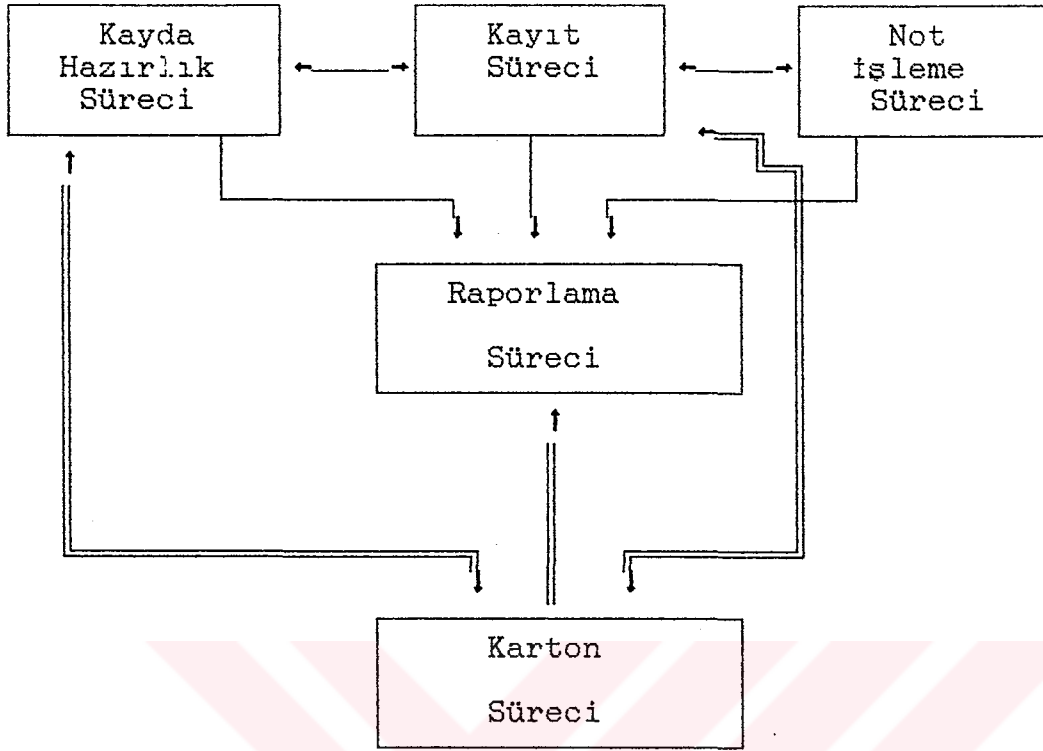
ÖfO'nu beş ana süreçten oluşmaktadır. Süreçlerin bazıları kendi başlarına birer uygulama programı iken, bazıları birden fazla yazılımın belirli kısımlarından oluşan bir küme olarak düşünülmüştür. Süreçler kendi başlarına varlık göstermelerine rağmen, ancak birbirleriyle iletişim kurarak bir bütünlük sağlar.

Devam eden bölümlerde süreçlerin işlevleri anlatılmıştır. Ancak unutulmamalıdır ki her süreçte insan faktörü de yer almaktadır ve sürecin başarıyla tamamlanmasında bu faktörün payı göz ardı edilmemelidir.

ÖfO süreçleri sırasıyla:

- Kayda hazırlık,
- Kayıt,
- Not işleme,
- Karton (Öğrenci izleme) ve
- Raporlama

olup aralarındaki ilişkiler şekil 3.1 de gösterilmiştir.



Şekil 3.1 ÖİÖ Süreçleri ve aralarındaki ilişkiler.

3.2.1. Kayda Hazırlık Süreci

Kayda hazırlık süreci, kayıt sürecine destek verir ve kayıt sürecinden önce yapılması gerekir. Başlıca iki kısımdan oluşmaktadır. İlk kısımda fakülteye ait bilgiler toparlanırken ikinci kısımda ise fakülte öğrencilerine ait düzenlemeler yapılır.

Fakülteye ait kimlik bilgisinin düzenlenmesi, fakültede o yarıyıl açılan derslerin işlenmesi, kayıтта cakişma kontrolünde rol oynayacak olan haftalık ders çizelgesinin bölüm ve şubeler bazında oluşturulması ve varsa bölümlere ait paket ve seçmeli ders müfredatının kaydı bu sürecin ilk kısmına aittir ve yarıyıl bazında yapılması gerekir.

Paket ve seçmeli ders müfredatı yalnızca paket ders

seeneğini kullanan fakltelerce doldurulmalıdır. Burada paket derslerle ilgili kod, kredi ve isim bilgileri blm bazında oluřturulur. Ama blm bazında farklı olan paket derslerinin kayıt srecinde kolay takibini saėlamaktır.

Faklte kimlik bilgisi olarak fO'nun alıřtırılacağı fakltenin kodu, blm sayısı; blmlerin kod, paket yarıyılı ve paket sayısı gibi bilgiler tutulur. fO'yu oluřturan yazılımlarca ve veritabanı dosyalarının isimlendirilmsi amacı ile kullanılır.

Bahsi geen bilgiler ayrı ayrı dosyalarda saklanır; faklte kimlik bilgisi faklte dosyasına (rnek dosya: F92K.V15), dersler ders dosyasına (rnek dosya: D0492K.V15), haftalık izelge hafta dosyasına (rnek dosya: H0492K.V15), paket ve semeli dersler mfredat dosyasına (rnek dosya: M0492K.V15) aktarılır (fO'unda veritabanında bulunan dosyaların isimlendirilmesi ile ilgili ayrıntılı bilgileri EKLER blmnn SİSTEM DOSYALARI alt blmnde bulabilirsiniz). Sre iřlevini karton programı ve kayıt entegre programında yer alan ders, hafta ve mfredat programları sayesinde gerekler.

İkinci kısımda, ėrencilere ait kayda hazırlık bilgilerinin oluřturulması gerekir. Bunlardan ilki faklteye yeni kayıt olacak ėrencilerin zlk bilgilerinin girilmesidir. zlk bilgileri ėrencinin tm ėrenim sresi boyunca saklı kalır. Bu bilgilerin fO'da saklandığı dosya zlk dosyasıdır (rnek dosya: 004.V15).

Fakltenin eski ėrencileri iin ise her yarıyıl ėrenci karton dosyasından (rnek dosya: K04.V15) ėrencilerin kayıt olacakları derslerin ve varsa alt yarıyıllardan kalan bařarısız derslerin bir tampon dosyaya atılması hazırlığı (rnek dosya: T0491k.V15) yapılır. Bu fonksiyonun amacı, kayıt esnasında ėrencilerin zellikle alt yarıyıldan bařarısız oldukları derslerin kayıt yapacak

operatore sunulması ve kontrollu bir şekilde ders kayıtlarının yapılmasını sağlamaktır.

Bir diğer işlem de kartona kayıtlı öğrencilerin numara, isim-soyisim bilgilerinin yer aldığı bir liste dosyası oluşturmak, bu sayede kayıt işlemlerinde aynı öğrenciye ait farklı isimlerin girilmesini önlemektir.

Kayda hazırlık sürecinin bu kısmını gerçeklemek için özlük ve karton programlarından yararlanılır.

Kayda hazırlık süreci tamamlanınca, bir sonraki aşamaya geçilir. Bu da Kayıt Süreci'dir.

3.2.2. Kayıt Süreci

Kayıt süreci öğrencinin ilgili yarıyıla ve yarıyıl derslerine kayıt olmasını sağlayan süreçtir.

Bilindiği gibi üniversitemiz fakültelerinde her yarıyıl başında öğrencilerin yarıyıl ve ders kayıtları yenilenmektedir. Öğrenciler kayıt olmak istedikleri dersleri basılı formda öğrenci servisine verirler. Ancak yarıyıl harcını ödeyenlerin yarıyıl ve ders kayıtları yapılır.

Bir öğrencinin ders kaydının kontrollu yapılabilmesi için operatore sunulması gereken ders bilgileri üç grupta toplanabilir: Öğrencinin almak istediği (formda yazılı) dersler, öğrencinin alt yarıyıldan kalan başarısız dersleri ve o yarıyıl bölümde açılan ders bilgileri.

Öğrenci almak istediği dersleri yarıyıl ders programından seçerek forma yazar. Öğrenci, yukarıda sözü geçen bilgiler ışığında öncelikle alt yarıyıllardan kalan ve almakla zorunlu olduğu derslere kayıt edilmelidir.

Bunun için kayda hazırlık sürecinden gelen ve fakültedeki tüm öğrencilerin alt yarıyıllardan kalan başarısız ders bilgilerinin yer aldığı tampon dosyasından yararlanılır.

Belirlenen dersler iki tür filtreden geçirilir. Bunlardan ilki, seçilen derslerin o yarıyıl açılmış olması gerektiğinden, ders dosyasından faydalanarak yapılır. Ders dosyasında yer almayan dersler için kayıt yaptırılamaz. Diğer, alttan alınabilen dersler ile öğrencinin aktif (kayıt olacağı) yarıyılından aldığı derslerin çakışma kontrolundan geçirilmesi ile yapılır. Yönetmelik gereğince derslere devam zorunluluğu olduğundan, öğrencinin alt ve aktif yarıyıldan aldığı derslerin saatlerinin çakışmaması gerekir. Bunu sağlamak için kayda hazırlık sürecinde hazırlanan bölüm ve varsa bölüm şubeleri bazında yaratılan haftalık ders çizelgelerinden yararlanılır. Seçilen ve saatleri çakışan dersler operatöre bildirilir. Bu sayede öğrencinin bir şekilde büro çalışanlarını atlatması bir ölçü azaltılmış olur.

Ancak yönetmeliğin değişmesi veya fakülte kurulu kararları gibi bazı kural dışı durumlar yüzünden bu sınırlamalar operatör müdahalesi ile aşılabılır.

Özellikle kayıt zamanlarında ofislerin büyük derdi haline gelen kayıt değiştirme, ders ekleme ve değiştirme gibi işlemler bu süreç içinde yapılarak son durum karton dosyasına aktarılır. Böylelikle öğrenci kartonu sıklıkla yapılan yaz-boz türü işlemlerden yalıtılmış olur.

Başarıyla kayıt edilen öğrenciye belge verilir.

Kayıt sürecinin iletişimde bulunduğu süreçler arasında karton ve not işleme süreçleri bulunur.

Kayıtların tamamlanması ile birlikte not sürecinde süreçte kullanılmak üzere ders dosyaları oluşturulur. Bu

dosyalarda dersi alan Öğrencilerin listesi yer alır. Dosyalar ilgili Öğretim Üyelerine disketler içinde dağıtılır.

Sınav dönemleri sonunda, not sürecinden geri alınan ders not dosyalarındaki (örnek dosya: KBZ200.V15) not bilgileri, öğrenci kayıt dosyasına disketler geldikçe aktarılır.

Not aktarım işlemleri tümüyle bittikten sonra, yine son durum not işleme süreci tarafından kayıt dosyasından karton dosyasına alınır.

Kayıt süreci kayıt programını kullanır.

3.2.3. Not İşleme Süreci

Not işleme süreci kayıttan gelen ve disketler halinde ilgili Öğretim Üyelerine dağıtılan bilgilerin, her sınav döneminde işlenmesi ve yarıyıl sonunda kayıt dosyasına aktarılmasından ibarettir.

Sürecin kullandığı not işleme programı, dersi alan öğrencilerin yarıyıl içi, sonu ve bütünleme sınav sonuçları ile derse devam durumlarını, ilgili not dosyasına Öğretim üyesi tarafından işlenmesini sağlar.

Ayrıca süreç içinde ilgili sınav sonuçlarının çıkışları alınabilir ve ders başarı çizelgeleri oluşturulabilir.

Öğretim üyelerinin sınav değerlendirme kriterleri farklı olduğundan, sınav sonuçları üzerinde kullanıcının (Öğretim üyesi) genel değişiklikler yapabilmesi için olanak sağlanmıştır. İstenildiğinde tüm sonuçlar, Öğretim üyesinin değerlendirme şekline göre, bir katsayı ile

çarpılır veya toplanabilir veya her iki işlem birden gerçekleştirilebilir.

Sınav dönemleri tümüyle sona erdikten ve not işleme işlemleri öğretim üyeleri tarafından tamamlandıktan sonra ilgili not dosyaları yine disketler halinde geriye, öğrenci işlerine alınır. Bilgiler kayıt programındaki bir altprogram ile öğrenci kayıt dosyasına aktarılır. Ders notu aktarımı işlemleri bittikten sonra son not durumu karton süreci tarafından alınır.

3.2.4. Karton (Öğrenci izleme) Süreci

Karton süreci, öğrencinin fakülteye ilk girişinden çıkışına kadar, tüm eğitim durumunun izlenmesini sağlayan temel süreçtir. Diğer süreçlerle iletişim kurarak fonksiyonunu yerine getirir.

Öğrenci durumu izleme süreci, öğrencinin fakülteye ilk kayıt oluşundan başlar ve fakülteden ayrılınca kadar, bir çevrim (daha özel olarak ders izleme çevrimi) içinde durumunu izlemeye devam eder.

Fakülteye yeni kaydı yapılan her öğrenciye bir karton kaydı açılır ve boş kartona öğrencinin tüm öğrenimi boyunca alması gereken zorunlu dersleri, ders dosyalarından (ilgili öğretim yılının yaz ve kış ders doyaları) yarıyıllar bazında aktarılır. Bu durumda derslere henüz kayıt yapılmadığı için ilgili derslerin durum hanesinde ALINMADI bayrağı bulunacaktır.

Asıl süreç, kayıt sürecinden gelen yarıyıl kayıt ve ders kayıt bilgilerinin, kayıt dosyasından karton dosyasına aktarımı ile başlar. Kayıt prosesi ile yarıyıllar boyunca alınan dersler kartona geçirilir. Sınav dönemleri sonunda alınan bu derslere ait not bilgileri de aktarılır.

Kartonda derslerin durum bayrağı başarısız dersler için TEKRAR, başarılı dersler için GEÇTİ olarak konumlanacaktır. Tekrara kalınan zorunlu dersler, kayda hazırlık süreci tarafından tampon dosyasına (alttan alınacak zorunlu derslerin dosyası) öğrenci bazında kayıt edilecektir.

Süreçte öğrencinin yalnız ders bilgileri değil, izin, staj, bitirme ödevi, konusu ve durumu, yarıyıllar bazında zorunlu -varsa paket ve seçmeli- türden derslerin kredileri, izinli olup olmadığı gibi bilgiler de tutulmaktadır. Tüm bu bilgiler ile mezuniyet durumuna gelen öğrenciler saptanabilir. Mezun olan öğrenciler mezunlar dosyasına transfer edilir (örnek dosya: M0492K.V15).

Karton süreci kayda hazırlık ve kayıt süreci ile birlikte bir kapalı çevrim oluşturmaktadır. Süreç karton programı tarafından yürütülür.

3.2.5. Raporlama Süreci

Raporlama süreci, özlük, kayıt ve karton bilgilerinin değerlendirilmesi, istatistiksel bilgilerin alınması için gerekli olup, kullanıcı için oldukça geniş rapor oluşturabilme imkanı sağlar.

Fakülte genelini ve öğrenciyi baz alan raporlama imkanları vardır. Kayıtlı öğrencilerin derslere göre listesi, not listeleri, harç listeleri, karne , karton bilgilerinin dökümü, özlük bilgilerinin dökümü, transkript, ara transkript, kural dışı durumlarda yapılan operatör müdahalelerini saklayan jurnaller bu sürecin içinde üretilirler. Süreç ÖİÖ'daki her temel program tarafından desteklenir.

3.3. Süreçlerin İş Akışı

ÖİÖ'daki iş akışını iki sınıfta incelemek mümkündür. Bunlar fakülte geneli için ve öğrenciler bazında yapılanlardır. Burada iş akışları bir yarıyıl verilecektir.

3.3.1. Fakülte Geneli İşlemleri

Süreçlerin tek tek öğrencilere bağlı olmayan, fakültenin genelini ilgilendiren kısımlarıdır.

3.3.1.1. Kayda Hazırlık

Fakülte geneli için geçerli olan ve bu sürece ait olan işlemler fakülte kimliğinin oluşturulması, karton durum dosyasının oluşturulması, paket ders müfredatının hazırlanması, ders dosyasının hazırlanması, haftalık ders çizelgelerinin hazırlanması ve alt yarıyıllardan kalan başarısız zorunlu derslerin belirlenmesi işlemleridir.

Bu süreç her yarıyıl başında fakülteye özgü kimlik bilgisinin oluşturulması ile başlar. ÖİÖ, herhangi bir fakülteden bağımsız, üniversite geneli için düşünüldüğünden kullanmaya geçmeden fakülte dosyasına girilmesi gereken bilgiler vardır. Bu bilgilerin oluşturduğu yapı fakülte kimlik bilgisidir. Fakülte kimliğinde kitapçığın ilerleyen bölümlerinde detaylı olarak açıklanacak olan fakülte adı, numarası, bölüm sayısı, bölümleri, bölümlerle ilgili grup sayıları, bölüm kodları ve varsa paket sayıları, paket yarıyılları gibi fakültenin tanımını yapan bilgiler yer alır ve bu dosya her yarıyıl yenilenir. Sürecin bu kısmı operatör tarafından koşul programı ile fakülte dosyası oluşturularak gerçekleştirir.

Karton durum dosyasının ve paket müfredatının hazırlanması işlemleri karton programı tarafından sırasıyla karton durum ve müfredat dosyalarının işlenmesi ile olur. Paket müfredat programı ile bölümlere ait paketlerin dersleri ayrı bilgisayar ortamına geçirilmiş olur. Böylelikle paket seçmeye hak kazanan bir öğrencinin, seçtiği paketin zorunlu derslerini eksiksiz vermesi izlenebilecektir. Karton durum dosyası ile bölümler bazında staj koşulları ve yarıyıl bazında verilmesi gereken toplam ders kredileri girilir. Bu bilgiler açılan her kartona aktarılarak öğrencinin yarıyıllara ve ders türüne göre (zorunlu ve varsa paket ve seçmeli) vermek zorunda olduğu kredi hesaplamaları için kullanılır ve öğrencinin ilgili yarıyılı için kredi kontrolleri yapılmış olur.

Ders dosyasında fakülte geneli için açılan tüm derslerin listesi bulunmaktadır. Bu bilgiler ders programı tarafından girilir. İlgili dosya özellikle kayıt programında ders kayıt işleminde seçilen dersin varlık testinin yapılması için ve açılan kartonlara sekiz yarıyıllık ders bilgilerinin -öğrencinin tüm öğrenimi boyunca alması gerekli zorunlu derslerin belirlenmesi amacıyla- aktarılması için kullanılır.

Yönetmelik gereğince öğrencilerin ders devam zorunluluğu vardır. Bu nedenle alt yarıyıllardan dersi kalan öğrencilerin farklı yarıyıllardan aldıkları derslerin saatlerinin çakışmaması gereklidir. Bu nedenle, öğrenci derslere kayıt edilirken bu kontrol, ilgili haftalık ders çizelgesi ile sağlanır. Bu çizelgeler bölümler ve şubeleri bazında hafta programı tarafından hazırlanır ve hafta dosyasına yerleştirilir.

Yapılması gereken diğer bir işlem de eski öğrencilerin alt yarıyıllardan kalan başarısız zorunlu derslerinin belirlenip veritabanında ayrı bir dosya içine yerleştirilmesidir. Ders çevrimini sağlayacak olan bu

işlem karton programının bir opsiyonu ile karton dosyasındaki bilgilerden yararlanılarak oluşturulur. Böylelikle ders kayıtlarında öğrencilerin alt yarıyıllardan başarısız oldukları dersleri almalarının kontrolü yapılmış olur.

3.3.1.2. Not İşleme Süreci

Yarıyıl sınav dönemi tamamlanıp, tüm derslere ilişkin not bilgilerinin kayıt ortamına aktarılmasını izleyen ve not durumlarının kayıt ortamından karton ortamına aktarılmasını gerçekleyen süreç işlemlerini kapsar. Böylece kartona öğrencilerin ders başarı bilgileri yerleştirilmiş olur. Not işleme sürecinin bu kısmı, tek tek öğrenci bazında da yapılabilir. Ancak bu işlemler diğer kısma aittir. Bu kısım karton programı kayıt ve karton dosyalarının kullanımı ile gerçekleşir.

3.3.1.3. Karton Süreci

Hem fakülte geneli hem de öğrenciye bağlı işlemlerin yapıldığı süreç olması itibarıyla bu kategoriye dahil edilmiştir.

Öğrencilerin tüm öğrenimleri boyunca almak zorunda oldukları dersler, kendileri için ilk karton açıldığında aktarılmaktadır. Bu ders bilgilerinden giderek, her yarıyıl başında alınan derslerin ve her yarıyıl sonunda da alınan derslere ilişkin notların, fakülte genelinde öğrenci kayıt dosyasından aktarılması işlemi yürütülür.

Bu işlem, karton programı tarafından kayıt ve karton dosyalarının kullanılması ile gerçekleşir.

3.3.1.4. Raporlama Süreci

Bu sürecin fakülte genelini ilgilendiren durum ve istatistiksel raporlama işlemleri, bu kısma dahildir.

Fakülte geneli ve bölüm genelini ilgilendiren raporlamalar not istatistikleri, transkriptler, bazı özlük bilgilerine göre dökümler, fakülte geneli rafine bilgilerin rektörlüğe sunulmak üzere SQL/DS veritabanı yönetim sisteminin formatında alınmasıdır. Bu işlemleri gerçeklemek üzere karton, kayıt, özlük yazılımları ve ilgili dosyaları kullanılır.

3.3.2. Öğrenci Bazlı İşlemler

Burada tek tek öğrenciler için yapılan işlemleri içeren süreçlerin ilgili kısımları anlatılacaktır.

3.3.2.1. Kayda Hazırlık Süreci

Sürecin fakülteye ilk kez gelen öğrencinin özlük bilgilerinin girilmesi işlemini kapsayan kısmı bu kategoriye dahildir. Özlük bilgileri öğrencinin tüm eğitimi boyunca saklı kalır.

3.3.2.2. Kayıt Süreci

Yarıyıl başında fakültenin her öğrencisi için ayrı ayrı kayıt yapıldığından kayıt sürecinin tümü bu kısma dahildir. Süreç, öğrencilerin harçlarını yatırmaları ve kayıt fişlerine alacakları dersleri yazmaları ile başlar, kayıt yazılımı ile alttan ders alma, alınan dersin var olması ve geçerli olması, farklı yarıyıllardan alınan derslerin çakışmaması kontrollerinin yapılması ile derslere

kaydı içerir.

Ders silme, deęiřtirme gibi pratikte srekli ortaya çıkan iřlemler de srecin bu kısmına dahildir. Bu iřlemleri gerekleřtirmek iin kayıt programı, kayıt dosyası, kartondan gelen tampon dosya, haftalık ders izelgesi ve ders dosyasından yararlanılır.

3.3.2.3. Not iřleme Sreci

Srecin sınav dnemi iindeki kısmı bu kategoriye girer. Not disketlerinin hazırlanması, daęıtımı, iadesi ve kayıt dosyasına transferi iřlemlerini ierir. Kayıt ortamına giren ęrencilerin derslere gre daęılımı ıkartılıp bu bilgiler ders bazında birer not dosyası řeklinde, ęretim yelerine gnderilmek zere, disketler halinde paketlenir ve daęıtılır. Yarıyıl ii, sonu sınavlarının deęerlendirmeleri ilgili not dosyalarına iřlenir. Dosyalardaki not ve bilgileri tek tek, kayıt ortamındaki yerlerine aktarılır.

Daha nce aıklanan not bilgilerinin kartona aktarım iřleminin ęrenciyi baz alarak gerekleyen kısmı da bu kategoriye dahildir.

3.3.2.4. Karton Sreci

Karton bilgilerinin ęrenci bazında incelenmesi, deęiřtirilmesi, ęrenci kartonu aılması iřlemleri srecin bu kategorisine aittir.

3.3.2.5. Raporlama Süreci

Raporlama sürecinin öğrenciye ait olan, not durumu çıkışları (karne), transkript, özlük raporu, ders not listeleri gibi işlemler bu gruba dahildir. Sürecin bu kısmını not programı ve not dosyası, karne programı, karton programı rapor işleme opsiyonu ve karton dosyası gerçekler.



BÖLÜM 4. TEMEL VERİ DOSYALARI

ÖİO'nun veritabanını oluşturan dosyalar, erişim yöntemi açısından iki gruba ayrılırlar. İlk gruptaki veri dosyaları oldukça büyük boyutlarda olabilmektedir. İlgili dosyalara erişim süresini kısaltmak (en az disk G/Ç işlemi yapmak) için bu tür dosyalara bir indeks dosyası aracılığı ile erişmek gerekir. Diğer türdeki dosyalar ise, indekse gerek duyulmaksızın yaratılan rastgele erişimli ve ASCII dosyalardır.

İndeksleme işlemine gerek duyulan dosyalar hiç şüphesiz çok sayıda kayıt içeren, sıklıkla erişilen ve dolayısıyla erişimi süratli olması istenen dosyalardır.

Veritabanındaki dosyaları, kullanımı açısından temel ve destek dosyalar iki sınıfa ayırabiliriz. Temel dosyalar, ÖİO'nun vazgeçilmez elemanlarıdır ve ÖİO'nun bütün yükü bunların üzerlerine dağılmış durumdadır. Bunlar fakülte dosyası, özlük dosyası, kayıt dosyası ve karton dosyasıdır. Özlük, kayıt ve karton dosyaları öğrenci numarasına göre indekslidir.

Dosyaların isimlendirilmesi ile ilgili ayrıntılı bilgi için EKLER bölümünde SİSTEM DOSYALARI kısmına bakınız.

4.1. Fakülte Dosyası

ÖİO üniversite geneli için düşünüldüğünden, her fakülte, otomasyonun temel programlarınca gerek duyulacak olan fakülte parametrelerini ve fakülte kimliğini saklayan bir rastgele erişimli dosya oluşturur. Dosyanın indeksi yoktur ve tek kayıttan oluşmaktadır. Dosya ismi her dönem,

yarıyıl ve zaman parametrelerine göre yenilenir. Örneğin 1992 yılının yaz yarıyılı için düzenlenen fakülte dosyası F92Y.V15 tir.

Dosyanın içinde fakülte adı, fakülte kodu, bölüm sayısı, her bölüm için bölüm adı, grup sayısı, bölüm kodu, paket sayısı ve paket isimleri yer almaktadır. (Şekil 4.1).

Kayıt yapısı, fakülte kaydı ve bölüm kayıtlarından oluşup aşağıda sembolik gösterimi sunulmuştur.

Fakülte Kaydı = 2448 Sekizli	
Fakülte Adı	Karakter katarı
Fakülte Kodu	Karakter katarı
Bölüm Sayısı	Sayısal
Bölüm Kaydı	
Bölüm Kaydı*8 = 299 Sekizli	
Bölüm Adı	Karakter katarı
Grup Sayısı	Sayısal
No Başlangıç	Sayısal
No Son	Sayısal
Bölüm Kodu	Karakter katarı
Paket Sayısı	Sayısal
Paket Yarıyılı	Sayısal
Paket isimleri	
Paket isimleri*8 = 248 Sekizli	
Paket Adı	Karakter katarı

Şekil 4.1 Fakülte kaydında önemli alanlar.

Dosya, koşul programı tarafından oluşturulur ve kayda hazırlık ve kayıt sürecine katkıda bulunur.

Dosyanın kayıt uzunluęu 2448 Sekizli dir.

4.2. Özlük Dosyası

Fakülteye kayıtlı öğrenciler hakkındaki özlük bilgileri bu dosyada tutulur. Bilgiler bir kez girilir ve her öğrenci için saklanır. Dosyada verilere indeksli erişim söz konusudur.

Dosya ismi fakülte parametresine göre değişmektedir. Elektrik-Elektronik fakültesi için 004.V15 veri dosyası ve 004.NXN indeks dosyası kullanılır.

Dosyanın kayıt tipinde yer alan bilgiler: öğrenci numarası, adı, soyadı, kayıt tarihi, fakültesi ,bölümü, baba adı, doğum tarihi, doğum yeri, nüfusa kayıtlı olduğu yer, cinsiyeti, medeni durumu, askerlik durumu, varsa bedensel özü, bitirdiğı son öğretim kurumu, bitirdiğı lise ve bitirme yılı, lise derecesi, ÖSYM de kaçınıcı tercihi olduğu, giriş puanı, ebeveynlerin -hayatta iseler- işleri, ailesinin birey sayısı, ailesinin ortalama aylık geliri, varsa kendisinin aldığı ortalama aylık geliri, ailesinin ikametgah adresi, öğrencinin haberleşme adresi, yurttaki kalıyorsa yurt adresi, bildiğı yabancı diller (Şekil 4.2).

Dosya karton programı özlük alt programı tarafından yönetilir ve kayda hazırlık sürecine hizmet verir.

Aşağıda şekilde, özlük kaydında yer alan bilgiler ve gösterildikleri veri tipleri hakkında özet bilgiler yer almaktadır.

Özlük Kaydı.	
Öğrenci no	Karakter katarı
Adı	" "
Soyadı	" "
Fakültesi	" "
Bölümü	Nümerik
Şubesi	Karakter
Baba adı	Karakter katarı
Nüfusa kayıt yeri	" "
Doğum yeri	" "
Doğum tarihi	Nümerik
Cinsiyeti	Lojik
Haberleşme Adresi	Karakter katarı
ikametgahı	" "
Bitirdiği Lise	" "
Lise Derecesi	Nümerik

Şekil 4.2 Özlük kaydında yer alan bazı bilgiler.

4.3. Kayıt Dosyası

Bu dosyada, yarıyıl başlarında yapılan öğrenci kayıtları saklanır ve her yarıyıl yenilenir. Dosyaya indeksle erişilir.

Dosya ismi zaman ve fakülte parametrelerine göre değişir. Elektrik-Elektronik fakültesi için 1992 yılı yaz yarıyılına ait öğrenci kayıt dosyası 00492Y.V15 ve indeks dosyası 00492Y.NXN olacaktır.

Bir dosya kaydında yer alan bilgiler: öğrenci adı, soyadı, numarası, kayıt yarıyılı, aldığı derslerin kodları, derslerle ilgili devam, not ve ders kodu alanları, bitirme ödevi alıp-almadığı, varsa paketi, bölümü, hazırlık öğrencisi olup olmadığı, yarıyıl izni alıp almadığı ve harç durumu (Şekil 4.3).

Bu dosya kayıt programı tarafından yönetilir ve kayıt sürecine hizmet verir.

Dosyaya eklenen her kayıt 390 Sekizli yer tutmaktadır.

Öğrenci Yarıyıl kaydı = 390 Sekizli	
Öğrenci No	Karakter katarı
Adı	" "
Soyadı	" "
Bölümü	Nümerik
Kayıt Yarıyılı	Karakter
Ders Kodları	Karakter dizisi
Ders Notları	Nümerik dizi
Ders Hakları	Karakter dizisi
Paketi	Nümerik
İzinli	Lojik
Harcı	Nümerik

Şekil 4.3 Yarıyıl kaydı özet bilgileri.

4.4. Öğrenci Durum izleme (Karton) Dosyası

Karton dosyası öğrencinin tüm öğrenimi boyunca eğitiminin yarıyıl bazında izlenmesini sağlamak amacıyla veritabanında yer almaktadır.

Kayıt yapısında öğrenci numarası, adı-soyadı, bölümü, şubesi, paketi, bitirme ödevi bilgileri, yaptığı stajlatın hafta sayısı, yarıyıllara göre aldığı derslerin kodları, kredileri, kaçınıcı hakta aldığı, başarı durumu bilgileri yer almaktadır (Şekil 4.4).

Her kaydı 7089 Sekizli uzunluğunda olan dosya, ismini fakülte parametresinden alır. Elektrik-Elektronik fakültesi için K04.V15 veri dosyası ve K04.NXN ilgili indeks dosyası isimidir.

Dosya, öğrenci durumu izleme sürecine hizmet verir.

Karton Kaydı = 7089 Sekizli	
Öğrenci No	Karakter Katarı
Adı	" "
Soyadı	" "
Bölümü	Nümerik
Şube	Karakter
Bitirme Ödevi	Karakter Katarı
Bitirme Hocası	" "
Bitirme Notu	Nümerik
Bitirme Kredi	"
Yarıyıl Harcı	"
" Dersleri	Karakter Katarı
Ders Notları	Nümerik
" Kredileri	"

Şekil 4.4 Karton kaydı özet bilgileri.

4.5. Not Dosyaları

Not dosyaları, kayıt programı tarafından oluşturulur ve not programı tarafından işlenir. Dersi alan öğrencilerin listesini barındırır. Not işleme sürecine hizmet verir.

Kayıt tiplerinde öğrenci numarası, adı - soyadı, dersi kaçınıcı kez aldığı, ara ve yıl sonu sınav notları, devam durumu ve ödev notu bilgileri yer almaktadır (Şekil 4.5).

Dosyalar ismini, ilgili dersin kodunu alır.

Her kayıt 50 Sekizli yer almaktadır.

Not kaydı = 50 Sekizli	
Öğrenci No	Karakter Kat.
Adı	" "
Soyadı	" "
Hak	Karakter
Vize	Nümerik
Final	"
Bütünleme	"
Başarı	"
Ödev	"

Şekil 4.5 Not dosyası kaydı bilgileri.

4.6. Destek Dosyalar

Destek dosyalar süreçlere hizmet veren programlara, kullandıkları temel dosyaların dışında yararlandıkları ek bilgilerin sunulduğu dosyalardır.

Bunlar ders dosyası, karton tampon dosyası, jurnal dosyalar, rapor dosyalarıdır.

Ders dosyası, ilgili yarıyılta açılan derslerin listesinin bulunduğu dosyadır. Dosya ismi fakülte ve zaman parametrelerine göre konulur. Örneğin Elektrik-Eletronik fakültesi için 1992 yaz yarıyılında D0492Y.V15 ismini alır. Kayıt ve karton süreçlerine hizmet verir. Kayıt bilgisi olarak, ders kodu, kredisi, haftada kaç saat açıldığı, Türkçe ve İngilizce isimleri, dersi veren öğretim üyelerinin isimleri yer almaktadır. Bir ders kaydı 4164 Sekizlidir.

DULP-Kredi dosyası transkript çıkışlarında ders kredilerinin D(ders), U(uygulama), L(laboratuar) ve P(proje) olarak dağılımını sağlayan dosyadır. Ders koduna göre indekslidir. Kayıt boyu 43 Sekizlidir.

Karton tampon dosyası kayıt süreci için gerekli olan öğrencilerin alt yarıyıllardan kalan derslerinin listesinin yer aldığı dosyadır. Benzer şekilde dosya ismi T0492Y.V15 tir ve indeksli bir dosyadır.

Her kaydında öğrenci numarası ve ders kodu yer almaktadır.

Kayıt uzunluğu 18 Sekizlidir.

Jurnal ve rapor dosya çıkışları düz formatta ASCII dosyalardır.



BÖLÜM 5. VERİ TABANINI YÖNETEN TEMEL PROGRAMLAR

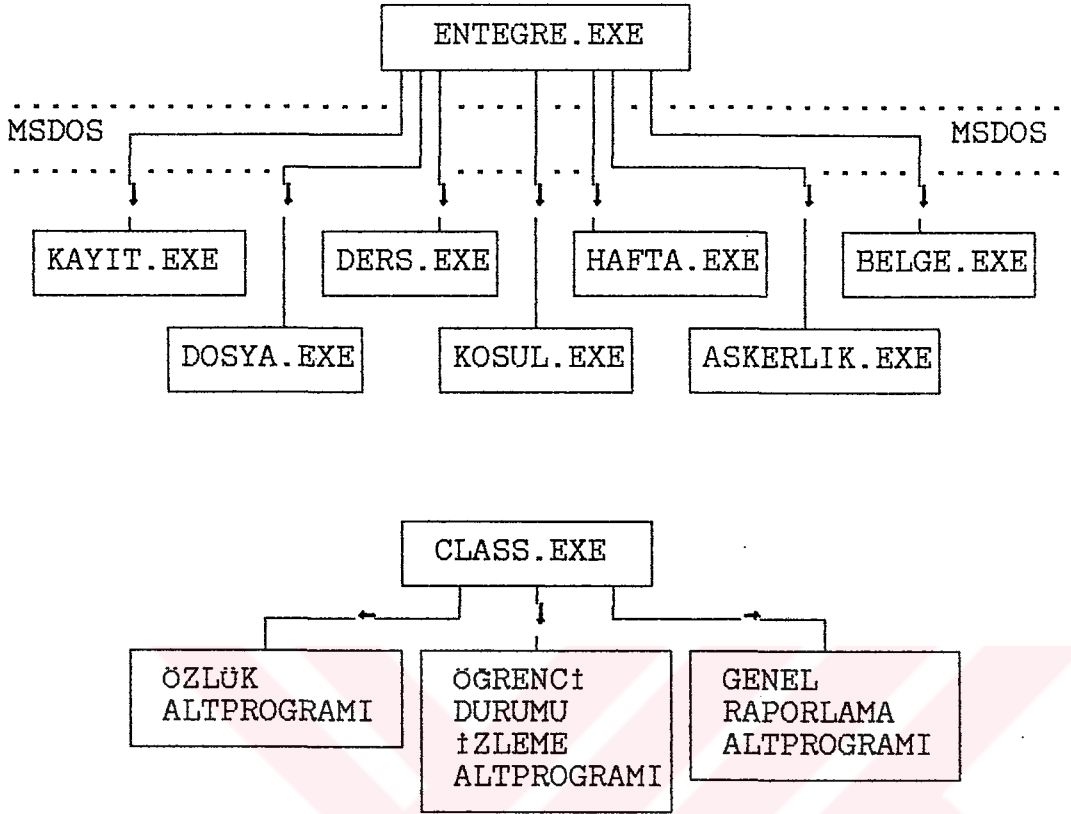
Burada öğrenci işleri otomasyonu veritabanını yöneten başlıca programlara değinilecek, programların işlevleri ve özellikleri belirtilecektir.

Sistemdeki veritabanını yöneten yazılımlar dört temel program ve bu programların işlevlerini yerine getirmek için gerekli ek veri dosyalarının yönetilmesini sağlayan destek programlardan oluşmaktadır.

Temel programlar özlük, kayıt, not, karton programlarıdır. Temel programlara destek olunması ve dolayısıyla belirtilen süreçlerin yerine getirilmesi için, koşul, ders, karne gibi programlar bulunmaktadır (Şekil 5.1).

ÖİO'daki tüm programların yazımı ile ilgili klavye/ekran giriş-çıkışları ve bazı karakter katarı fonksiyonları ile temel veritabanı fonksiyonlarını yerine getiren altprogram kütüphaneleri son bölümde sunulmaktadır. ÖİO için yazılan uygulama programlarının tümünde Turbo Pascal v5.5 derleyicisi kullanılmıştır.

ÖİO'da yer alan programların bazıları bir programın altprogramı şeklinde; bazıları da diğer bir program tarafından işletim sistemi üzerinden çağrılarak, daha entegre bir şekilde oluşturulmaya çalışılmıştır (Şekil 5.1). Aşağıdaki şemada, programlar arasındaki söz edilen bağlantılar gösterilmiştir.



Şekil 5.1 ÖİÖ'daki entegre programlar.

5.1. Özlük Programı

Özlük programı, veritabanında fakülte öğrencileri hakkında temel özlük bilgilerinin saklandığı veri dosyasını (Özlük Dosyası) yöneten programdır.

Program, kullanıcıya öğrenci özlük bilgilerinin ekran girişi için iki sayfalık bir giriş ortamı sunar.

Fakülteye kayıt olan her yeni öğrenci için bu program aracılığı ile kayda hazırlık sürecinin özlük verilerinin alınması ile ilgili kısmı tamamlanmış olur. Özlük bilgi girişi her öğrenci için bir kereye mahsus yapılır.

Program ilgili fakülteye ilişkin özlük dosyası oluşturur veya daha önceden yaratılmış dosyayı kullanır. Özlük dosyası üzerinde kayıt ekleme, kayıt düzeltme, kayıt silme ve arama gibi temel kayıt işlemlerini yerine getirir.

Ayrıca Özlük programı öğrenciler hakkında rafine özlük bilgilerinin rektörlükte IBM 4381'de çalışan SQL/DS v2.2 Veritabanı Yönetim Sistemine aktarılması için bir ara dosya oluşturabilme imkanına sahiptir.

Özlük programı Kayda Hazırlık ve Raporlama süreçlerine hizmet verir ve ÖİÖ'nün belkemiğini oluşturan üç programdan biridir.

Program Karton programına (KARTON.EXE) entegre durumdadır. Yaklaşık kod büyüklüğü 150 KB dir.

5.2. Kayıt Programı

Kayıt programı, ÖİÖ veritabanındaki öğrencilerin yarıyıl kayıtlarının saklandığı veri dosyasını yöneten programdır. Öğrenci kayıt dosyasının yönetimi ve yarıyıl kayıtları gerçekleştirir.

Bu program ile öğrenci kaydının ekran girişi için olanak sunulur. Her yarıyıl yinelenen kayıt süreci bu program tarafından desteklenir.

Kayıt programı, her yarıyıl kayıt döneminde kullanılmak üzere tasarlanmıştır. Öğrenci o yarıyıl almak istediği derslerin listesini öğrenci bürosuna getirdiğinde, bu program derslerin alınabilirlik ve çakışma kontrolü yapılmakta, engel çıkmadığı takdirde dersler kayıt dosyasına işlenmekte ve öğrenciye kayıt belgesi verilmektedir.

Program ilgili zaman ve fakülte parametrelerine göre bir kayıt dosyası oluşturur veya daha önceden açılmış olan kayıt dosyasını kullanır. Ancak her yeni yarıyıl için ayrı birer dosya açılmaktadır.

Yeni öğrenci kaydı ekleme, var olan öğrenci kayıtlarını değiştirme ve silme; öğrenci kaydındaki ders bilgileri ile ilgili olarak ders ekleme, silme ve değiştirme işlemleri kayıt programı ile yapılabilir.

Program, kayıt dosyasını öğrenci numarasına göre indeksli olarak saklar. Yönettiği veri dosyasına ait indeks dosyası zarar görmüş ise ilgili veri dosyasından indeks dosyasını yeniden üretme özelliği vardır.

Program ayrıca kayıtlı öğrencileri tarayarak, not işleme süreci için gerekli olan, boş not dosyalarını ders bazında üretir.

Ayrıca program ile kayıt dosyasından çeşitli rapor çıktıları alabilmek amacı ile rapor kaynak dosyası hazırlanabilir.

Program kayıt ve raporlama süreçlerine hizmet verir ve ÖİO'nun belkemiğini oluşturan üç programdan biridir.

Kendisi KAYIT.EXE isimli yürütülebilir bir dosyadır ve diğer bazı programlar gibi ENTEGRE.EXE yürütülebilir dosyası içinden bir DOS çağırısı ile kullanılır.

5.3. Not Programı

Not programı veritabanındaki not dosyalarını işleyen programdır. Ders notlarını not dosyalarına işlemeyi sağlayan bir yazılımdır. Bu bakımdan not dosyaları üzerinde kayıt ekleme, silme fonksiyonları yerine kayıt

güncelleme veritabanı fonksiyonu yer almaktadır.

Program, derse ait yoklama, vize, mazeret, final ve bütünleme listelerini ekranda izleme ve giriş yapma olanaklarını sunar.

Fakültelerde öğrenci sayılarının yüksek olması ve sınav sayıları göz önüne alındığında, bütün bölümlere ait ders notlarının tek tek işlenmesinde operatöre düşecek yük ve bunun sonucu yapacağı hata oranının yüksek olacağı açıktır.

Örneğin, Elektrik-Elektronik fakültesinde şubeleri de göz önüne alınırsa, 200 civarında ders açılmaktadır. Her ders ortalama 100 öğrenci tarafından alınmaktadır. Toplam üç sınav notu işleneceği düşünülürse, operatörün hata yapma olasılığının ne denli yüksek olduğu görülebilir.

Ancak bu işlem, ilgili öğretim üyesine bilgisayar ortamında sunulduğunda toplam yük dağıtılarak öğrenci başına düşen hata riski büyük ölçüde azaltılır ve işlem süresi azalırken güvenilirlik artar.

Not işleme programı bu amaçla hazırlanmıştır. Kayıt işlemleri sonunda operatör tarafından ders bazında oluşturulan ve ilgili öğretim üyelerine gönderilen, not dosyalarına öğretim üyesi tarafından öğrencilerin devam, vize, final ve bütünleme notları işlenir. Dönem sonunda da bu dosyalar geri alınır.

Program, dersi alan öğrencilerle ilgili boş liste, devam listesi; vize, final ve bütünleme sınav sonuç listelerini verebilmektedir.

Not işleme ve raporlama süreçlerine hizmet veren not programının kendisi NOT.EXE isminde bir yürütülebilir dosyadır.

5.4. Karton (Öğrenci Durum izleme) Programı

Karton programı ÖİO veritabanındaki öğrenci karton (durum izleme) dosyasını yöneten programdır.

Öğrenci kartonları, bir öğrencinin fakülteye ilk girişinden çıkışına kadar tüm eğitim durumunun saklandığı ortamlardır.

Fakülteye ilk kez kayıt olan öğrenciye boş bir karton açılır. Karton üzerinde öğrencinin tüm yarıyıllar boyunca alması gereken zorunlu dersler kayıtlıdır. Her yarıyıl öğrencinin aldığı dersler ve başarı durumları bu kartona aktarılır. Öğrencinin hangi yarıyıllardan ne gibi eksiklerinin kaldığı kartonun incelenmesi ile anlaşılır.

Ancak bu işlemin ortalama birkaç bin öğrenci içeren bir fakültede yapıldığı (Elektrik-Elektronik Fakültesi'nde yaklaşık 5000 öğrenci öğrenim görmektedir) düşünülürse hata olasılığı ve dolayısıyla sonuçların büyüklüğü tartışılmaz.

Karton programının hazırlanmasındaki temel amaç, sözü edilen işlemleri, daha hızlı ve güvenilir olarak gerçekleştirmektir. Bu sebebi ile karton programı ÖİO'nun belkemiğini oluşturan programdır.

Program karton bilgilerinin doğrudan müdahale edilmeksizin işlenmesine olanak sağlayacak şekilde tasarlanmıştır. Kayıttan gelen ders alındı ve not bilgisinin işlenmesi, aktif yarıyılın belirlenmesi, başarısız derslerin belirlenip kayda hazırlık sürecinde kullanılmak üzere bir tampon dosyaya aktarılması ve öğrencinin mezun olabilmesi için gerekli bilgilerin işlenmesi, karton programı tarafından yapılır. Ancak, yönetim kurulu kararları gibi istisnalar için, yapılan her tür değişik gerekçesini bir jurnal dosyasına eklemek koşulu ile, operatöre kartona müdahale etme imkanı sunulmuştur.

Karton programında bu müdahaleler ancak operator modunda gerçekleştirilebilir. Programda bir öğrencinin tüm yarıyıllar boyunca alması gereken, aldığı, tekrara kaldığı ve geçtiği dersleri ayrı ayrı inceleyebilme imkanı verilmektedir.

Yarıyıllara göre toplam ve geçilen derslerin, ders türüne göre (zorunlu, paket ve seçmeli) kredi dağılımı karton programı tarafından sürekli kontrol edilmektedir.

Karton dosyası üzerinde kayıt ekleme, silme ve değiştirme gibi temel dosya fonksiyonları da karton yazılımı tarafından yerine getirilmektedir.

Program, kayıt programıyla iletişim kurarak kayıtlı öğrencilerin kartonlarına yarıyıl kayıtlarında alınan dersleri ve yarıyıl sonunda derslerin başarı durumunu aktarır.

Kayıt programına ise, o yarıyılın alt yarıyıllarından kalan başarısız derslerin listesini, bir tampon dosya aracılığı ile sunar. Böylece her öğrenci için gerekli ders çevrimleri yapılmış olur.

Karton programı, öğrenci durumu izleme ve kayda hazırlık süreçlerine hizmet vermektedir.

Program CLASS.EXE isimli yürütülebilir bir dosyadır ve yürütülebilir kod uzunluğu yaklaşık 250 KB, kullandığı veri segmenti yaklaşık 30 KB'dir.

5.5. Destek Programlar

ÖİO veritabanı yönetim sistemini oluşturan temel programlar dışında, bu programların kullandığı bazı veri dosyalarını yöneten veya temel programların ihtiyaç duyduğu birtakım işlemleri yerine getirip ÖİO'nun tamamlanmasına

yardımcı olan programlar bu sınıfa girer.

Ana programlara destek sağlayan veri dosyalarını (ders dosyası, karton tampon dosyası) yöneten ve temel programlarda bulunmayan bazı görevleri yerine getiren (karne, öğrenci belgesi, askerlik belgesi verme gibi) yazılımlardan başlıcaları aşağıda anlatılmaktadır.

Bunlardan en önemlisi Koşul programıdır. Programın kendisi KOSUL.EXE isminde yürütülebilir bir dosyadır. ÖİO'nun temel dosyalarından biridir. Fakülte kimliğinin saklı olduğu fakülte dosyasını yönetir (oluşturur, değiştirir). Bu sayede diğer programların koşullanması için gerekli bilgiler fakülte bazında yerleştirilmiş ve ÖİO'nun üniversite içinde genelliği sağlanmış olur.

Tüm bilgiler, kullanıcı arayüzü sayesinde kolaylıkla girilebilir. Normalde her yarıyılta bir kez kullanılan program, kayda hazırlık sürecine hizmet verir.

Ders programı, yarıyılta açılan derslerin bilgisini taşıyan ders dosyasını yöneten ve dosyaya işlemede kullanıcı arayüzü sunan bir programdır. Ders dosyası, kayıt esnasında alınmak istenen dersin varlığını test etmek için kullanılır. Ders programı, bu dosyaları yaz ve kıış yarıyılları için ayrı ayrı oluşturur veya var olan dosyalar üzerinde işlem yapmaya olanak sağlar. Tipik bir veritabanı işleme programı olan Ders programı, ders bilgisi ekleme, silme, değiştirme, listeleme gibi olanaklar sunar.

Programın kendisi DERS.EXE isimli yürütülebilir bir dosyadır. Kayda hazırlık ve raporlama süreçlerine hizmet verir.

Ders kayıtlarında önemli noktalardan birisi, öğrencilerin alt yarıyıllardan aldıkları derslerle, aktif yarıyıldan (kayıtlı oldukları son arıyıl) aldıkları

derslerin (yönetmelik gereğince derse devam zorunluluğunun bulunması nedeniyle) çakışmasının önlenmesidir.

Bu yüzden bölümlerin şube bazında haftalık ders çizelgesine ihtiyaç vardır. Haftalık ders çizelgeleri dosyasını oluşturmak ve işlemek için Hafta programı kullanılır. Kullanıcı arayüzü sayesinde haftalık çizelgeler kolaylıkla oluşturulabilmekte ve haftalık programlardan herhangi birinin kopyaları alınarak diğer şubelerin çizelgelerinin kolayca hazırlanmasında kullanılabilir. Programın kendisi HAFTA.EXE isimli yürütülebilir bir dosyadır ve kayda hazırlık sürecine hizmet verir.

Karne programı, öğrenci kayıt dosyasından, yarıyıllara ait derslerle ilgili başarı durumlarını "yarıyıl karnesi" adı altında çıkartan programdır. Program, istenirse fakültedeki tüm öğrencilere veya kullanıcı arayüzü sayesinde seçilen öğrencilere karne verebilir. Kendisi KARNE.EXE isimli yürütülebilir bir dosyadır ve raporlama sürecine hizmet vermektedir.

5.6. Hazır Altprogram Kütüphaneleri

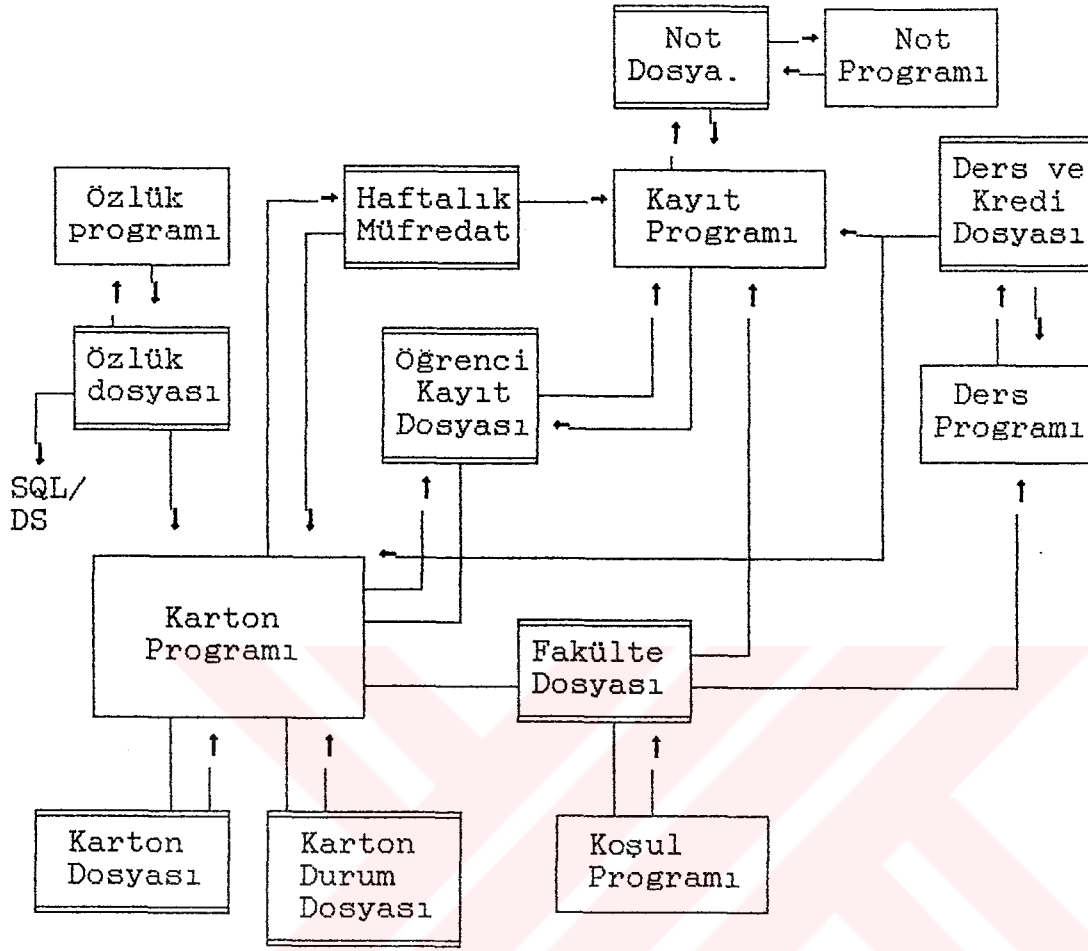
ÖfO'da temel klavye-ekran giriş-çıkışı ve bazı karakter dizisi işlemleri için Turbo Power şirketinin piyasaya sunduğu Turbo Professional V5.0 genel amaçlı giriş-çıkış altprogram kütüphanesi ile veritabanı yönetim sistemi programlarında B+ Ağacı indeksleme yöntemini kullanmak için Borland Int.'in sunduğu Turbo Access (TACCESS) V4.0 veritabanı yönetimi altprogram kütüphanesi kullanılmıştır [5].

Turbo Professional altprogram kütüphanesindeki altprogramlardan özellikle klavye-ekran giriş-çıkış işlemlerini yapanlar (getstring, fastwrite...) ve karakter

katarlarını işleyenler (str2word, real2str, trim...) -gerektiğinde kaynak kodlarında değişiklikler yapılarak kullanılmıştır. Bu altprogram kütüphaneleri Tpcrt, Tpdos, Tpstring, Tpentry (tüm tuş kontrolunu programcıya vermesi için değiştirildi: Tpentry2), Tpmemchk, Tpwindow, Tppick, Tpmenu ve Opkey birimeri (pascal unit) içindedir.

Veritabanındaki dosyalara optimal sürede (en az disk başvurusu ile) ile erişmek için halen en çok sözü geçen indeksleme yöntemlerinden biri olan B+ Ağacı yöntemi kullanılmıştır. B+ Ağacı indeksleme metodunu kullanmak için Borland şirketinin hazırlamış olduğu altprogramlar kullanılmıştır. Bunlar, kullanıcı tarafından tanımlanabilen tip bildirimleri doğrultusunda veri dosyaları ve ilgili indeks dosyalarını yaratır. Bu dosyalar üzerinde (dosya açma-kapama, kayıt ekleme, düzeltme, silme gibi) temel veritabanı işlemlerini yürütür.

Yukarıda iki bölüm halinde anlatılan veritabanını oluşturan başlıca dosyalar ve veritabanı yönetim sistemini meydana getiren başlıca yazılımlar ile aralarındaki ilişkiler şemada gösterilmiştir (Şekil 5.2).



Sekil 5.2 ÖİO'yu oluşturan program, dosyalar ve aralarındaki ilişkiler.

BÖLÜM 6. KARTON PROGRAMININ İNCELENMESİ

Karton programı fakültelerin öğrenci servislerinde kullanılmakta olan ve öğrencinin tüm eğitim durumunun saklandığı çizelgelerin bilgisayar ortamına geçirilmesi amacıyla yazılmıştır.

Program altprogramlardan ve altprogramlar da daha küçük alt programlardan oluşmaktadır.

Öğrenci durumu izleme ve raporlama süreçlerine hizmet eden karton programı, istenirse elle diğer programlardan bağımsız olarak kullanılabilir veya diğer programlarla -ara dosyalar üzerinden- iletişim kurarak ÖİÖ içinde de yer alabilir.

6.1. Karton Kavramı

Fakülte öğrencilerinin tüm eğitim-öğretim süreleri içinde ders başarı bilgileri kartonlarda saklanır. Karton, üzerinde öğrenci numarası, adı-soyadını, bölümünü, tüm öğrenimi boyunca alması gereken dersleri ve aldığı derslere ilişkin başarı durumunu yarıyıllar bazında saklamak için kullanılan basılı formlardır.

Yıllardır karton üzerinde, fakülte öğrenci işleri büro çalışanları tarafından elle sırasıyla şu işlemler yapılmaktadır: Fakülteye kayıt olan her yeni öğrenci için bir karton açılır. Açılan kartonda öğrencinin kayıt olacağı yarıyıl ile ilgili alması gereken zorunlu dersler yer almaktadır. Her dönem başında buradaki bilgilerden gidilerek öğrencinin ders kayıt kontrolleri yapılır ve

dönem sonunda bu derslere ilişkin not bilgileri yazılır. Öğrencilerin mezuniyet hesapları burada gerçekleştirilir. Öğrenci fakülteden ayrılma koşullarından birini sağlayıncaya kadar bu çevrime devam edilir.

Yeni kayıt olan her öğrenciye 8 yarıyıl için bir karton tahsis edilir. Üzerlerinde ilgili yarıyıllarda almaları gereken derslerle ilgili ders kodları, sınav notları, başarı durumu ve fakülte kurulu kararları için yer mevcuttur.

Karton bilgileri sayesinde öğrencinin tüm yarıyıl derslerini eksiksiz tamamlaması için gereken kontroller de yapılmış olur.

Bilgisayar ortamında benzer şekilde her öğrenci için karton dosyasında bir kayıt açılır.

Karton kavramını bilgisayarda temsil ederken bu bilgiler dışında öğrencinin mezun olması için gerekli kontrollerin yapılmasını sağlayan bazı ek bilgiler de dahil edilmiştir.

6.2. Karton Programının Yapısı

Karton programının başlıca görevleri öğrenci kartonunu saklayan karton dosyasını yönetmek, karton işlemlerini kontrollu olarak yerine getirmek ve ÖİÖ'nu tamamlamaktır.

Karton yazılımı 8 adet ana (classeri, class31, class2, class11, classr11, classsta, classoz1, classdul), 3 adet program destek (destek1, destek2, destek31), 2 adet veritabanı destek (karaccs2 ve types2), 8 adet hazır kütüphane (tpcrt, tpdos, tpwindow, tppick, tpmemchk, tpmenu, tpentry, opkey) ve 2 adet standart derleyici (printer ve dos) biriminden (pascal unit) oluşmaktadır.

Bunlardan ana, destek ve 1 adet vertitabanı destek birimi (types2.tpu) bu öiO gereklemesi esnasında yazılmış ve 1 adet hazır kütüphaneden bazı sınır deęerleri deęiştirilmiştir (tpentry2).

Her ana bölümün belirli bir görevi vardır ve kendisini ilgilendiren altprogram paralarından oluşmaktadır. Bölümlerin birden fazlasını ilgilendiren ortak alt programlar ise destek birimlerince sağlanmaktadır.

Ana programda yer alan bazı önemli deęişkenler ile alt programlar ve işlevleri aşağıda belirtilmiştir gibidir.

(* Kartonrec tipi, karton dosyası kaydının tipidir. 7089 sekizli uzunluęundaki bu tampon alan "mkarton" deęişkeni ile gösterilmiştir. Bu ve benzeri tampon alanlara, programın veri segmentinden yer ayırmamak için, dinamik deęişkenler karşılık getirilmeye çalışılmıştır *)

MKARTON: ^KARTONREC;

(* Kartondaki yarıyıl bilgilerinin ayrı ayrı inclenebilmesi için "iyyilbilgi" dinamik alanı kullanılmaktadır. Uzunluęu 871 sekizlidir *)

IYYILBILGI: ^YYILTYPE;

(* Özlük dosyasının dinamik tampon alanı için kullanılan işaretçi *)

MOZLUK: ^OZLUKREC;

(* 3 aşamalı karton ekranına doğrudan müdahaleleri kontrol eden deęişkenler, karton ekranları EKLERin Veri Giriş Ekranları bölümünde yer almaktadır *)

READONLY1, READONLY2, READONLY3: Boolean;

(* 3 aşamalı karton kullanıcı ara yüzünden ilki, edit_karton prosedüründe yer alan çevrim ile gereklenir *)

Procedure EDIT_KARTON:

(* 3 aşamalı karton kullanıcı ara yüzünden ikincisi, edit2 prcsedürü içindeki çevrimde yer alır. Benzer şekilde son aşama da edit3 prosedürü tanımlanmıştır *)

Procedure EDIT_2;

Procedure EDIT_3;

(* Tüm bu aşamalarla ilgili kullanıcı ara yüzlerin tanımlı sırasıyla aşağıdaki prosedürlerde yer alır *)

Procedure INIT_EDIT1;

Procedure INIT_EDIT2;

Procedure INIT_EDIT3;

(* Her üç aşama için kullanıcı arayüzü aşağıdaki fonksiyonlar ile sağlanmakta olup Turbo Profesional yazılımının TPEDIT alt kütüphanesinin bir fonksiyonudur. Tanımda yer alan ESR, giriş ekranının alanlarının saklı bulunduğu kayıt bloklarını temsil eder. ID, giriş ekranında imleç pozisyonunun hangi elemanda konumlanacağını berliirler. ID, 0'dan 65535'e kadar değer alabilir. Readonly, giriş ekranına yazma koruması verir. EsType, 2 sekizli büyüklüğünde sayısal tiptir ve ekran giriş ortamından hangi tuş ile çıkıldığını verir *)

Function EDITSCREEN (Var Esr: ESRECORD; ID: word;
READONLY: Boolean): ESTYPE;

6.3. Yazılımın Bölümleri ve Temel Altprogramları

CLASSERI.PAS ana birimi ile yazılımın kullandığı temel dosyalar ile ilgili açma ve kapama işlemleri yapılmaktadır. Dosyada DOSYA_AC ve DOSYA_KAPA isimli iki prosedür bu işlemleri gerçeklemekle yükümlüdürler. Prosedürler fakülte, karton ,karton durum, özlük dosyaları dosya açma ve kapama işlemlerini yaparlar. Dosya açma işlemlerinden birisi başarısızlıkla sonuçlanırsa, bu durum çağırılan programa aktarılır.

Aşağıda ilgili prosedürlerin kaynak kodlarından alıntılar yer almaktadır.

```
(** veritabanlarına erişimler **)
function dosya_ac: byte;
begin
    dosya_ac := 0;
    (** fakülte dosyası **)
    assign(fakultef,fakultefname);
    {$i-}
    reset(fakultef);
    {$i+}
    (*** karton dosyası ***)
    openfile(kartonf,kartonfname,sizeof(kartonrec));
    if not ok then begin
        makefile(kartonf, kartonfname,
            sizeof(kartonrec));
        makeindex(kartoni
            forceextension(kartonfname,'NXN'),
            sizeof(kartonvar.no)-1,
            noduplicates);
    end
    else begin
        openindex(kartoni,
            forceextension(kartonfname,'NXN'),
            sizeof(kartonvar.no)-1,
            noduplicates);
        if not ok then begin
            (* indeksi yeniden oluşturan program bloğu *)
            end;
        clearkey(kartoni);
    end;
end;

procedure dosya_kapat;
begin
    (** fakülte dosyası **)
    close(fakultef);
```

```

(***) karton dosyası (***)
closefile (kartonf);
closeindex(kartoni);
end;

```

CLASS11.PAS ana biriminde yer alan ve karton ekranlarının ilki ile ilgili kullanıcı ara yüzünü sunan başlıca prosedür ve fonksiyonlar aşağıda sunulmuştur.

```

Function EditMsg1;   Ekran yardım bilgisini taşır.
Procedure IncXXXXX;  "inc" ile başlayan prosedürler,
ilgili giriş alanlarında çubuk tuşu ile seçme yapılmasını
sağlarlar.

```

```

Function NoKontrol (oncekino: str8): Boolean;
Girilen öğrenci numarasına test uygular.

```

```

Procedure Ilk_kayit;
Karton dosyasında, öğrenci numarasına göre, ilk kayda
gidilmesini sağlar. Benzer şekilde dosyanın son kaydına ve
bulunulan kayıttan itibaren bir sonra ve bir önceki
kayıtlara konumlanmayı ve herhangi bir kaydı silmeyi
sağlayan prosedürler vardır.

```

```

Procedure Init_editX;
ilgili kullanıcı ekranını düzenleyen prosedürdür.

```

```

Procedure Transfer_sta;
ilk defa karton açılan öğrenci için karton durum
dosyasından bazı sabitler yüklenir.

```

CLASS2.PAS ve CLASS31.PAS dosyalarında kartonun yarıyıl bilgileri için kullanıcı ekranı sunan, editsta.pas karton durum dosyası için ve CLASSOZL.PAS özlük dosyası için kullanıcı ekranları açan ve dosyaların yönetimini sağlayan prosedür ve fonksiyonlar yer almaktadır. Bunlar edit1.pas dosyasında anlatılan alt programlarla -makro

seviyede- benzer işlemleri yaparlar.

DESTEK2.PAS ve DESTEK31.PAS dosyasında yukarıda anlatılar bölümlerden bazıları veya hepsi için ortak olan alt programlar yer alır. Bunlardan en önemlileri,

Function Modified_Karton(var kar1, kar2: kartonrec):
boolean; iki karton dosyası tamponunu karşılaştırır.

Procedure Init_Karton;
Karton değişkenini sıfırlar.

Procedure Operator (Onoff:boolean); Operatör modu
denilen kartona doğrudan operatör müdahalesini sağlayan
konuma izin verir veya izni kaldırır.

SONUÇLAR ve ÖNERİLER

Bu çalışma ile İTÜ fakülteleri öğrenci ofislerinde kullanılmak amacıyla ve daha önceden yapılan çalışmalar doğrultusunda bir ÖİO (Öğrenci İşleri Otomasyonu) paketi oluşturulmaya çalışılmıştır.

Konunun yeni olmaması sebebiyle hale hazırda var olan yazılımların kullanılması bir takım sınırlamaları da beraberinde getirmiştir. Örneğin öğrenci kayıtları için kullanılmakta olan kayıt programının Turbo Pascal'da yazılmış olması sebebiyle, tüm otomasyonun bütünlüğü açısından diğer yazılımlar da hazır bir veritabanı yönetim sistemi dili veya farklı bir derleyici dili yerine yine Pascal'da yazılmıştır.

Yazılımların uyum içinde çalışmasını sağlayan bir veritabanı tasarlanmıştır. Ancak üzerindeki yazılım, bir veri tabanı yönetim sisteminin gereklerini tam olarak yerine getirememektedir. Özellikle güvenlik sorunu ön plandadır ve yazılımın üzerinde çalıştığı işletim sisteminin de açık noktaları vardır. Verinin paylaşırlılığı söz konusu değildir. ÖİO veritabanında verinin bütünlük özelliğinin korunması ve veri standardının sağlanması tamamen bu otomasyonu tasarlayan kişinin sorumluluğundadır. Ayrıca kullanılan veritabanını yöneten yazılımlar tamamen otomasyona özgüdür.

ÖİO yazılımlarının hepsi Turbo Pascal v5.5 derleyicisinde yazılmış olup, 640 KB ana bellek ve renkli ekran kartlı IBM PC veya uyumlu bir donanım üzerinde yüklü MSDOS 4.0 veya yukarı bir versiyon işletim sisteminde kullanılmaya uygun olarak hazırlanmıştır (ancak MSDOS

6.0'ın problemlili oluşu sebebiyle bu versiyona sahip bir makinada çalıştırılmaması tavsiye edilir). Programlar genişlemeye imkan verecek şekilde yazılmıştır ve istenildiğinde kolayca ek yapılabilir.

Veritabanında sıkca kullanılan büyük boyutlu dosyalara erişim performansını arttırmak (en az disk erişimini sağlamak) için popüler yöntemlerden birisi olan B+ Ağacı tekniğini kullanan bir indeksleme kutuphanesi (TACCESS.PAS) ile indeks dosyaları oluşturulmuştur.

Bu yapı sayesinde N kayıtlı veri dosyası üzerinde kurulu M mertebeli bir B+ ağacında herhangi bir kaydı bulma işlemi en fazla $\log(N)/\log(M)$ mertebesi adımda yapılabilmektedir.

ÖİÖ kullnımı, yazılımlardaki kullanıcı arayüzleri sayesinde kullanımı oldukça basittir.

Sistemin güvenlik sorunu -kayba uğrayan bilgilerin yeniden elde edilmesi- ancak dışarıdan yapılacak periyodik yedeklemeler ile çözümlenebilir. Ayrıca raporlamalardan alınan çıkışlar uygun bir süre ilgili bölümlerde birer kopya şeklinde saklanmalıdır.

Bu tür bir çalışma için daha güvenli bir hesap ortamı (un*x gibi) ve kendisini kanıtlamış bir VTYS paketi (oracle gibi) kullanılması daha uygundur. Böylece yazılan uygulamaların programcı tekelinde kalması da bir ölçüde azaltılmış olur.

Gerçeklenen ÖİÖ halen İTÜ Elektrik-Elektronik fakültesi öğrenci servisinde olumlu eleştiriler almış olup halen denenmektedir.

KAYNAKLAR

- [1] McFADDEN, Fred, and HOFFER, J., Data Base Management
Second Edition, Benjamin-Cummings
Company, (1988).
- [2] DATE, C.J., An Introduction to Database Systems
Vol.1, Addison-Wesley Pub., (1990).
- [3] STOCKER, P.M., GRAY P.M.D., ATKINSON, M.P.,
Database-Role and Structure An Advanced
Course, (1984).
- [4] KAMRAN, P., CHIGNELL, M., KHOSHAFIAN, S., WONG H.,
Intelligent Databases, John Wiley & Sons
Inc, (1989).
- [5] BORLAND INC., Turbo Pascal Database Toolbox, Borland
International, (1987).
- [6] KNUTH, D.E., The Art of Computer Programming Vol.3,
Addison-Wesley, (1973).

EK.1 KARTON PROGRAMI KULLANICI REHBERİ

Yazı içinde geçen otomasvon kelimesi Öğrenci işleri Otomasvonu devimine karşılık olarak kullanılmıştır.

KARTON PROGRAMININ ÇALIŞTIRILMASI

Öncelikle bu vazılım IBM XT,AT veva uyumlu, renkli ekran kart ve monitörlü ve en az 640 KB ana bellekli bir kişisel bilgisayar üzerinde çalışabilmektedir.

Programı çalıştırmak için CLASS.EXE isimli yürütülebilir dosya ve sadece fakülte kosul dosyası yeterlidir. Bu konfigürasyonda program, karton inceleme için çalıştırılabilir. Ancak karton deşisiklikleri ve raporlamalar için fakülte ders dosyaları ve dulp-kredi dosyası gerekmektedir. Program, eğer yoksa fakülte için yeni karton dosyası varadır. Otomasvonda kullanılan diğer programlarda olduğu gibi karton programında da kullanılan dosyaların isimlendirilmesinde, komut satırında verilen parametrelerin önemi vardır. Dosya isimlendirmeleri ile ilgili bilgi için SİSTEM DOSYALARI kısmına bakınız. Karton programını çalıştırmak için en az üç adet parametre gereklidir. Bununla beraber program parametresiz çalıştırılırsa konuyla ilgili yardım bilgisi ekrana gelir.

Program Parametreleri

Programı çalıştırmak için CLASS komutu ile beraber üçü zorunlu biri secimlik olmak üzere en az üç parametre gerekmektedir. Bu parametreler sayesinde program içinde veri dosyalarının isimlendirilmesi yapılır ve dosyaların hangi dizinden alınacağı saptanır.

Zorunlu parametreler sırası ile fakülte numarası, yıl bilgisinin son iki rakamı ve sôemistir (yaz/kis) bilgisinin ilk harfidir. Böylece programın kullanacağı dosyaların isimleri için destek verilmiş olur.

Seçimlik parametre `./V` avıracı ile ve bu avıraca bitişik olarak verilir. Bu, veri dosyalarının bulunduğu dizini gösteren yoldur.

Programı çalıştırmak için gerekli format aşağıdaki gibidir:

CLASS <fk_no> <yıl> <sömestir> [/V<dizin_yolu>]

Örnek olarak, program Elektrik-Elektronik fakültesinde 1992-1993 öğretim yılının yaz sömestiri için kullanılacaksa `'CLASS 04 93 Y'` komutunun verilmesi gereklidir. Yol bilgisi verilmediğinden dosyalar komutun verildiği dizinde aranacaktır. Bir başka örnekte program, Fen-Edebiyat fakültesinde 1990 yılının kış sömestiri için çağırılmalı ve `C:\1990\KIS` dizinindeki dosyalar kullanılmak istensin. Bu durumda da `'CLASS 03 90 K /VC:\1990\KIS'` komutunun girilmesi gerekmektedir. Program parametresiz çağırılmayacağı ve sürekli parametre yazmanın da zaman kaybı olacağı düşünülerek bir batch dosyasının oluşturulması önerilir. Bunun için aşağıdaki işlemleri sırasıyla yapınız:

1. COPY CON KAR.BAT komutunu girin,
2. CLASS program ismini ilgili parametreleri ile yazın,
3. CTRL ve Z tuşlarına aynı anda basarak batch dosyayı saklayın.

Artık programı sadece KAR komutunu vererek de çalıştırabilirsiniz. Hatta AUTOEXEC.BAT sistem dosyasında tanımlanan yollardan birine ait bir dizinde ürettiğiniz batch dosyasını saklayarak, programın her yerden çağırılması sağlanabilir.

Programı İlk Kez Çalıştırıyorsunuz

Karton programının kullanıcı ara yüzü, kullanım kolaylığı sağlayacak şekilde düzenlenmiş olup kullanıcıya o andaki olanaklar hakkında bilgi vermektedir. Program menu

yapılı olup ekranın en alt satırlarında kullanıcıya yön veren yardımlar bulmak mümkündür. Programı ilk kez kullanıyorsanız bu dökümanın tüm bölümlerini okumanız tavsiye edilir.

MENÜLER

Programın çalıştırılmasıyla ekrana, programın temel olanaklarıyla ilgili bir menü gelir. Ana menü olarak adlandırabileceğimiz bu menüde 7 seçenek vardır: **Kayda Hazırlık, Kartona hazırlık, Not/Ders Yükleme, Karton, Raporlar, Mezunlar ve Son - programı sonlandırma - seçeneği.**

Kayda Hazırlık

Kayıt programına destek vermek amacıyla kartondaki tüm öğrencilerin listesinin alınması ve sonraki yarıyıldaki almaları gereken ders listesinin oluşturulması amacıyla kullanılır. Seçildiğinde Liste al ve Alınacak dersleri aktar seçeneklerinin yer aldığı bir alt menü ekrana gelir.

LISTE AL seçildiğinde liste çıkışının yazılacağı dosyanın ismi sorgulanır. Kullanıcı dilerse ESC tuşuna basarak çıkış almaktan vazgeçebilir. Onaylandığı takdirde liste dosyasına kartondaki tüm öğrencilerin numara, ad ve soyad bilgileri aktarılır. Bu listenin oluşturulması ile kayıt esnasında eski öğrencilerin isimlerinin tekrar tekrar veya hatalı girilmesi önlenmiş olur.

ALINACAK DERSLERİ AKTAR seçeneği, kartondaki öğrencilerin kayıt olacakları yarıyıla ait ve varsa alt yarıyıllardan almaları gereken derslerin bir tampon dosyaya aktarılmasını sağlar. Seçildiğinde aktarımın yapılacağı dosya ismi sorgulanır. Kullanıcı yine ESC ile aktarımdan vazgeçebilir veya gerekli düzeltmeleri yapıp ENTER ile onaylayabilir. Aktarım yapılan tampon dosyasında öğrenci numarası ve ders kodundan oluşan liste yer alır.

Aynı alt menüde liste ve tampon dosyalarının içeriğini görmek için seçenekler de mevcuttur.

Kartona Hazırlık

Bu seçenek ile öğrenci kartonu işlemleri için gerekli ön hazırlıkların yapılmasına olanak sağlanır. Seçildiğinde üç seçenekli bir alt menü ekrana gelir: **Tüm ders yükleme, DULP-Kredi düzenleme, Karton durum dosyası ve Özlük bilgileri.**

TÜM DERS YÜKLEME ile o yılın yaz ve kış yarıyıllarına ait ders dosyaları okutulup, karton kaydında ve ders değişikliklerinde kullanılacak tüm ders listesinin hazırlanması sağlanır. Her yeni kayıta bölüme ait dersler listeden, öğrenci karton kaydına aktarılır. Bunlar, o öğrencinin tüm öğrenimi boyunca alacağı zorunlu derslerdir. Daha sonra öğrenci kartonunda yapılacak ders ekleme/değiştirme işlemlerinde (bakınız Karton seçeneği 3. pencere olanakları) bu seçenekten üretilen ders listelerinden yararlanılır.

Seçildiğinde o yılın yaz yarıyılı ders dosyası sorgulanır. Kullanıcı onayı ile dosya okunup liste hazırlanır. Ardından kış yarıyılı ders dosyası için de benzer işlemler yürütülür. Dosyaların herhangi birinin eksik olması veya herhangi birinin başarıyla okutulamaması, yeni karton kaydına tüm yarıyıl derslerinin aktarılamamasına yol açar.

DULP-KREDİLER seçeneği: Ders kredilerinin transkriptte D(=ders), U(=uygulama), L(=lab.) ve P(=proje) olarak çıkması gerekir. Bu seçenek ile ders dosyasındaki salt kredi bilgilerinden de yararlanarak, kredilerin D,U,L ve P olarak dağılımlarını sağlamak amacıyla gerekli veri giriş ve inceleme ekranı sağlar.

KARTON DURUM DOSYASI seçeneği: Karton kaydında yer alan staj, bitirme ödevi yarıyılı ve yarıyıllar bazında zorunlu, seçmeli ve paket ders kredilerinin yeni kayıt anında öğrenci kartonuna aktarılması için karton durum dosyasından yararlanılır. Bu seçenek bahsedilen durum dosyasının hazırlanması, incelenmesi veya değiştirilmesi için kullanılır. Her yeni öğrenci kartonu açıldığında ilgili bölüme ait sabitler bu dosyadan kartona aktarılır. Daha ayrıntılı bilgi için **VERİ GİRİŞ EKRANLARI** kısmına bakınız.

ÖZLÜK BİLGİLERİ seçeneği, kartona kayıt edilecek öğrencilerle ilgili özlük bilgilerinin girilebileceği bir

veri giriş-inceleme ekranı sunar. Öğrencinin numarası, adı-soyadı, bölümü, fakülteye kayıt tarihi, ÖSYM bilgileri, nüfus kütüğü bilgileri, orta öğretim başarı bilgileri, ikamet bilgileri ve ailevi bilgileri için 2 sayfalık veri giriş-inceleme ekranıdır. Ayrıca bu kısmın diğer bir özelliği rektörlükçe fakültelerden istenen öğrenci özlük bilgilerinin, IBM 4381 üzerinde kullanılan SQL/DS ortamına transfer etmek üzere bir ASCII dosyaya aktarılabilmesidir.

Kayıttan Ders/Not Yükleme

Bu seçenek kayıt dosyasından öğrencilerin aldığı ders ve derse ilişkin not bilgilerinin kartona aktarılmasına yardımcı olur. Bunun için kullanıcıya iki seçenekli bir alt menü sunar: **Kayıttan Ders Yükle** ve **Kayıttan Not Yükle**.

KAYITTAN DERS YÜKLE, öğrencinin kayıt edilen derslerinin kayıt dosyasından okunmasını ve öğrenci kartonunda ilgili yarıyıllara ait ders listelerinde **ALINDI** olarak işaretlenmesini sağlar. Zira tüm ders yükle seçeneği ile kartona aktarılan derslerde **ALINMADI** işareti vardır. Seçildiğinde ilgili kayıt dosyası kullanıcıya onaylatılır. Kullanıcı ESC ile bu işlemi vazgeçebilir yada gereken düzeltmeleri yapıp ENTER ile onayı verebilir. Bu sayede kayıttan alınan dersler kartona işlenmiş olur.

KAYITTAN NOT YÜKLE seçeneği benzer şekilde not bilgilerinin kayıt dosyasından aktarımı için gereklidir. Not bilgileri not disketlerinden öğrenci kaydına aktarıldıktan sonra bu seçenek kullanılabilir. Seçildiğinde öğrenci kaydından, alınan derslere ilişkin vize, final, bütünleme ve başarı notları kartona aktarılır.

Karton

Bu seçenekte öğrenci kartonuna ait karton kaydı, ders izleme-değiştirme, başarı durumu inceleme gibi karton işlemlerinin yapılması sağlanır. Öğrenci kartonu için geniş olanaklı veri giriş-inceleme ekranı, kullanıcıya üç ayrı pencere ile sunulur: 1. pencere - genel karton bilgileri, 2. pencere - derslerden bağımsız yarıyıl bilgileri ve 3. pencere - yarıyıl dersleri.

Kullanılması en yoğun seçenek olarak düşünüldüğünden, menüde sadece boşluk tuşuna basarak seçmek mümkün kılınmıştır.

Karton Listeleme

Karton dosyasındaki öğrencilerin numara ve isim bilgilerini veren listeyi ekrana çıkarır. Liste üzerinde ok tuşları ile gezinilebilir. P tuşuna basarak, daha önceden saptanmış çıkış ortamına da yollanabilir. Esc ile listeden menüye dönlür.

Mezunlar

Karton kaydı bulunan öğrencilerden o yarıyıl mezun olanların belirlenmesi, mezunların mezunlar dosyasına aktarılması ve bu dosyanın listelenmesi işlemleri bu seçenekte sağlanır.

Seçildiğinde kullanıcıya 4 seçenekli bir alt menü sunulur: Kesin mezunların belirlenmesi, mezun olabileceklerin belirlenmesi, mezun dosyasına aktarma ve mezun listesi.

KESİN MEZUNLARIN BELİRLENMESİ seçeneği ile öğrenci kartonu tamamlanmış olanlar, yani tüm yarıyıl derslerini başarmış, harçlarını ödemiş, stajlarını tamamlamış ve bitirme projesini başarmış olan öğrenciler için karton kaydında tarama yapılır.

MEZUN OLABİLECEKLERİN BELİRLENMESİ seçeneği ile sadece ilk 6 yarıyılını tamamlamış olan öğrenciler taranıp listelenir.

MEZUN DOSYASINA AKTARMA seçeneği, mezuniyet durumuna gelmiş öğrencilerin, kartondan mezunlar dosyasına aktarılması için kullanılır. Seçildiğinde aktarım işleminin gerçekleştirilmesi için 2 alternatif sunulur: **Hepsi** veya **Seçerek**.

Hepsi seçeneği ile mezuniyeti kesin tüm öğrenciler mezunlar dosyasına aktarılırken karton kayıtları silinir.

Seçerek seçeneği ile de mezun olanlar listelenir ve ekrana gelen menüden ENTER ile seçilen öğrenciler mezunlar dosyasına aktarılıp karton kayıtları silinir. Her iki seçenekte aktarım işleminden hemen önce mezunlar dosyası sorgulanıp kullanıcıdan onaylanmsı istenir. Kullanıcı dilerse ESC tuşu ile aktarımdan vazgeçebilir.

MEZUNLAR DOSYASINI LİSTELEME seçeneği ile mezunlar dosyasında bulunan öğrencilerin listesi ekrana çıkarılır. Listedenden ENTER ile seçilen öğrenciyi tekrar kartona geri almak mümkündür.

Raporlar

Bu seçenekte karton dosyasından alınan rapor çıkışları ile ilgili seçeneklerin sunulduğu bir alt menü ekrana gelir. Alt menüde **Transkript**, **Yarıyıl karne**, **Karton çıkışı**, **çıkış ortamlarını ayarlama** ve **ASCII dosya görme** olanakları vardır.

TRANSKRİPT seçildiğinde içinde transkript dili, dekan ismi belirleme ve transkript çıkışının alınması için bir alt menü sunulur. Transkript dili seçeneğinde alınacak transkriptin Türkçe veya İngilizce olacağı, seçildiğinde ekrana gelen veri girişinde boşluk tuşuna basarak belirlenir. Dekan seçeneği seçildiğinde dekan ismi sorgulanır. Transkript al seçeneği ile de ekrana gelen öğrenci listesinden seçim yapılarak belirlenen çıkış ortamına transkript basılır. Yalnız başarılı derslerin çıkışı alınabildiğinden ara transkript çıkışları almak için de bu seçenek kullanılabilir.

YARIYIL KARNE seçeneği ile bir öğrencinin bulunduğu yarıyıldan itibaren geriye doğru herhangi bir yarıyla ait karne çıkışı alınabilir. Seçildiğinde öğrenci numarası ve yarıyıl bilgisi sorgulanır.

KARTON ÇIKIŞI seçeneği ile bir öğrencinin karton kaydı bilgilerinin dökümü alınabilir. Seçildiğinde **Hepsi** ve **Seçerek** seçeneklerinin bulunduğu bir alt menü ekrana gelir. Bu alt menünün işlevi daha önce anlatıldığı gibidir.

ÇIKIŞ ORTAMI seçeneğinde rapor çıkışları için ortam seçme (LPT1/LPT2/ASCII_dosya) ve kağıt ayarları ile ilgili

bilgilerin yer aldığı bir giriş ekranı sunulur.

ASCII DOSYA GÖRME seçeneği ile dosya çıkış ortamına alınan raporların içeriği görüntülenir. Seçildiğinde görüntülenecek dosyanın ismi sorgulanır. Bu seçeneğin görevini yürütebilmesi için VBROWSE.EXE programının bulunması ve programın çalışabilmesi için bellekte yeterli yerin olması gerekir. Aksi halde dosya görüntülenemez.

VERİ GİRİŞ EKRANLARI

Veri giriş ekranları kullanıcıların bilgi girişine izin veren ekranlardır. Her bilgi girişi ENTER ile onaylanmalıdır. Ekranların alttan 2 satırı kullanıcıya yardım bilgilerini içerir. Aşağıda karton programındaki temel veri giriş ekranları anlatılmaktadır.

Karton Durum Dosyası Veri Giriş Ekranı

Karton durum dosyası kayıtlarını oluşturmak, incelemek ve değiştirmek için kullanılır. Fakültenin her bölümüne ait minimum staj hafta sayısı, bitirme ödevinin alınacağı yarıyıl ve bölümde yarıyıllar bazında verilmesi gereken zorunlu, paket ve seçmeli ders kredi sabitleri girişi için bir veri girişi, inceleme ve değiştirme ekranıdır.

Ekranı veri alanları arasında düşey ok tuşları veya enter ile dolaşılabilir. Yalnız ekranın alt kısmında yer alan yarıyıl kredi bilgileri arasında sağa gitmek için CTRL ve sağ ok tuşuna, sola gitmek için ise CTRL ve sol ok tuşuna basılmalıdır. Alanlara yapılan her bilgi girişi ENTER ile onaylanmalıdır.

Kayıt işlemi F2 tuşu ile sağlanır. F9 tuşu kayıt siler. Bir sonraki bölümün karton durum bilgilerini görmek için PGDN tuşuna basılmalıdır. Benzer şekilde PGUP tuşu ile bir önceki bölümün karton durum bilgileri ekrana yansıtılabilir. İlk bölüme ulaşmak için CTRL-PGUP (^PGUP) tuşuna, son bölüme ulaşmak için ise CTRL-PGDN (^PGDN) tuşuna basılmalıdır.

Özlük Bilgileri Veri Giriş Ekranı

Özlük veri giriş ekranında öğrenci özlük bilgilerinin geniş sorgulaması yer almaktadır. Ekranda bazı veri alanları doğrudan bilgi girişi isterken, bazıları da program tarafından verilen alternatiflerin boşluk tuşuna basılarak seçilmesini ister. Okutulan bilgiler arasında öğrencinin numarası, adı-soyadı, bölümü, fakülteye kayıt tarihi, ÖSYM bilgileri, nüfus kütüğü bilgileri, orta öğretim başarı bilgileri, ikamet bilgileri ve ailevi bilgiler bulunmaktadır. Tüm bunları yaklaşık 2 sayfalık veri-giriş ekranı ile sorgular.

Daha önce belirtildiği gibi veri girişi 2 türlü olmaktadır: normal bilgi girişi ve ENTER ile onaylama veya BOŞLUK tuşu ile istenen alternatifi arama ve yine ENTER ile onaylama. Veri alanları arasında geçiş düşey ok tuşları ile sağlanır. Özlük bilgileri F2 tuşu ile kayıt edilir. Kayıtlı bilgi F9 ile silinebilir. Daha önce kayıt edilmiş verilere ulaşmak için PGUP (numara sırasına göre bir önceki öğrenci), PGDN (numara sırasına göre bir sonraki öğrenci), ^PGUP (CTRL-PGUP, numara sırasına göre ilk öğrenci), ^PGDN (CTRL-PGDN, numara sırasına göre son öğrenci) tuşları kullanılabilir. Eğer öğrenci numarası biliniyorsa, ^PGDN ve sonra PGDN (veya ^PGUP ve sonra PGUP) tuşları ile 'Yeni kayıt' konumuna gelip ilgili numara yazılıp ENTER'a basılır. Program, özlük dosyasında bu numaraya ait kayıt varsa ekrana yansıtır aksi halde yeni bir kayıt girişi olarak kabul eder.

Ekranın yeni kayıt konumunda olup olmadığı ise sağ üst köşedeki 'Yeni kayıt' işaretinden anlaşılır. Aksi durumda aynı yerde özlük dosyasındaki kayıt sayısı yer alır.

Özlük veri girişi ekranında kullanıcıya sunulan diğer bir olanak da her yıl rektörlüğün fakültelerden istediği özlük bilgilerinin, bu ekran da yer alan ve TAB tuşu ile sağlanan, ASCII çıkışı olanağıdır. Dosya formatının ASCII olması rektörlükte IBM 4381 ortamında kullanılan SQL/DS (Structured Query Language / Data System) e aktarılabilmesi için kolaylık sağlar. TAB tuşuna basıldığında Çıkış alınacak dosyanın ismi kullanıcıya sorgulanıp onaylatılır ve çıkış alınır. Daha sonra bu dosya bir transfer makinası ile 'Mainframe' sistemine aktarılır. Yazılacak küçük bir JOB dosyası ile SQLDBSU olanağı kullanılarak açılmış veri tabanına yollanabilir. Aktarım için diğer bir yöntem ise SQL kullanan bir uygulama programı yazmaktır.

Karton Veri Giriş Ekranı

Bu ekranda karton kayıt bilgileri ile ilgili veri alanları yer alır. Bunlar öğrencinin fakülte numarası, ismi, bölümü ve şubesi, hazırlıkta yer alıp almadığı, öğrencinin son kayıtlı olduğu yarıyıl, yaptığı staj hafta sayısı, varsa aldığı paket ve bitirme ödevi bilgileri; yarıyıllar bazında harçları, izin durumu ve ders durumu bilgileridir.

Bu veri giriş alanlarıyla beraber karton işlemlerinin daha kontrollü yapılmasını sağlayan ek alanlar da bulunmaktadır: karton durum dosyasından aktarılan yarıyıl kredi sabitleri ve staj sabitleri ile yarıyıl tamam bayrağı.

Karton veri giriş ekranı karmaşık kontrollerin yapıldığı bir ekran olup kullanıcıya 3 adet pencere halinde sunulur: yarıyıllardan bağımsız bilgi - 1.pencere, derslerden bağımsız yarıyıl bilgileri -2.pencere ve yarıyıl dersleri -3.pencere. Pencere arası geçiş ise TAB (ileriye) ve Sh-tab (geriye) tuşları ile sağlanır.

Ekranın sağ üst köşesinde özlük ekranında olduğu gibi 'Yeni kayıt' veya kartondaki öğrenci sayısını gösteren işaret vardır.

Ana menüden karton seçeneği ilk defa seçildiğinde karton ekranı Yeni kayıt durumundadır. Aksi halde kartonu veya özlük bilgisi en son incelenen öğrencinin kartonu ekrana yansır.

Tüm karton ekranı için geçerli diğer bir özellik ise Yeni kayıt durumu haricinde (var olan bir öğrencinin kartonu incelenirken) kartona müdahalenin engellenmesidir. Bu durumda karton ekranı 'Normal Mod'dadır. Ancak gerektiğinde bir takım değişikliklerin yapılması da istenir. Bu da ancak 'Operatör Modu' ile mümkündür.

OPERATÖR MODU, var olan karton kaydı ekranında değişiklik yapılmak istenince geçilmesi gereken bir durumdur. Operatör moduna ALT-O tuşu ile geçilir. Kullanıcıyı kısıtlamaz ve kontrollü olarak değişiklik yapabilmesini sağlar. Yapılan her değişikliğin hemen sonrasında ilgili açıklama bilgisinin girilmesi istenir ve açıklama bir ASCII dosya olan 'jurnal dosyası'na aktarılır. Eğer açıklamadan

vazgeçilirse ESC tuşuna basılmalıdır. Bu durumda yapılan değişikliklerin iptali sorgulanır. iptali istenmezse tekrar açıklama metnini okutmaya dönlür. Sonuçta ya açıklama girilmelidir ya da değişiklikler iptal edilmelidir. Yapılan ve onaylanan her değişiklik ekranda da fark edilebilecek şekilde bir renk değişimine uğrar ve aynı fon üzerine kırmızı veya kırmızı fon üzerine siyah renk ile farkedilir.

1. pencerede öğrencinin yarıyıllardan bağımsız karton bilgileri yer alır. Bunlar; fakülte numarası, ismi, bölümü, şubesi, hazırlık bilgisi, aktif yarıyılı (kayıtlı olduğu yarıyıl), yaptığı ve yapması gerektiği staj haftası sayısı, paketi, bitirme ödevi durumu, bitirme konusu, bitirmeyi veren, bitirme ödevi kredisi ve notu ve ekranın alt satırlarında tüm yarıyıllarda vermesi gerekli ders kredi değerleridir.

Bölüm, hazırlık, paket ve bitirme ödevi alanlarına veri girişi, boşluk tuşuna basılarak alternatif seçme ile yapılır. Diğer alanlar için doğrudan veri girişi uygulanır.

Ekranın üst-orta kısmında öğrencinin fakülteye giriş tarihi özlükten alınarak yazılır. Aksi halde günün tarihi belirir. Hemen altında mezuniyet durumu bulunmaktadır.

Yeni kayıt durumunda numara alanına girilen veri, fakültesi için test edilir ve test başarılı ise karton dosyasında aranır. Dosyada bulunmazsa yeni kayıt için sayfa açılır, aksi halde daha önce girilmiş olan kayıt ekrana yansıtılır.

Bu pencerede kullanılabilen tuşlar: ESC menüye dönme, F2 kayıt, F9 kayıt silme, TAB -2.pencereye geçiş, ALT-O operatör moduna geçiş veya operatör modundan çıkış, F10 rapor seçenekleri.

PGUP ile bir önceki kayda, PGDN ile bir sonraki kayda geçilebilir. ^PGUP ile ilk kayda, ^PGDN ile son kayda ulaşılır. Yeni kayıt ortamına gelmek için ilk kayda gelip PGUP tuşuna veya son kayda gelip PGDN tuşuna basılmalıdır. Yeni kayıt girişinde veya var olan bir kayıtta değişiklik yapıldığında kayıt yapılmaksızın ESC, PGUP veya PGDN tuşlarından birisine basılırsa kullanıcıyı kayıt yapması için sorgular.

TAB tuşuna basıldığında 2. ve 3. pencereler ekranda belirir ve 2. pencere aktif duruma geçer. 2. pencerenin hemen üzerinde bilgilerin hangi yarıyıla ait olduğuna dair yarıyıl bilgisi bulunur.

2. pencerede öğrenci harcı ve izin bilgisi yer alır. Harç alanına doğrudan bilgi girilebilir, izin alanında ise boşluk tuşu ile alternatif seçilebilir. PGDN tuşu ile bir sonraki yarıyıla, PGUP tuşu ile bir önceki yarıyıla, ^PGDN ile 8. yarıyıla ve ^PGUP ile 1. yarıyıla geçilir. Sh-tab veya ESC tuşuna basarak 1. penreceye; Tab tuşu veya son alanda iken ENTER yada aşağı ok tuşu ile 3. pencereye geçilebilir.

3. pencerede o yarıyıldaki alınması gereken, alınan, başarılan veya tekrara kalınan derslerin listesi yer alır. Pencerenin hemen altında her ders türü ile ilgili sayı ve kredi olarak başarılan ve verilmesi gereken miktarlar bulunur. Başarılan derslerin ağırlıklı ortalamaları da altta sağda yazılıdır. Tüm dersler başarıldı ise yarıyıl ortalaması adını alır. Normal modda 3. pencere içinde yalnız 1. sütun kullanılarak aşağı ve yukarı ok tuşları ve (PGDN, PGUP) sayfa tuşları ile dersler üzerinde gezinilebilir. Burada da operator müdahaleleri ancak operatör modunda geçerlidir.

Ders satırları, dersin durumuna göre renklendirilmiştir: Alınması gereken ve alınan dersler pencere renginde, başarılan dersler yeşil fon üzerinde ve tekrara kalınan dersler kırmızı fon üzerinde görüntülenir. Alınan derslerin not bilgileri ise mavi fon üzerinedir. Tüm operator müdahaleleri bulunduğu fon üzerine kırmızı renk ile ayırt edilebilir. Yalnız tekrara kalan dersler için kırmızı fon kullanıldığından, bu fon üzerinde siyah renk ile operatör müdahalesi uyarısı bildirilmiştir. Pencere altındaki bilgilerde kırmızı fon kullanıcıya uyarı için kullanılmıştır.

3. pencere içinde sütunlar arası yatay geçişler Tab ve Sh-Tab tuşu ile sağlanır. ESC ile veya 1. satırın 1. kolonunda iken Sh-Tab tuşu ile 2.pencereye geçilir. Kullanıcı operator modunda iken F7 ders seçme ve F5 paket seçme fonksiyonlarını kullanarak ders ekleme işlemi de yapabilir. Bu tuşlar ile ilgili dersler liste halinde kullanıcıya sunulur. Kullanıcı istediği dersin ilk karakterlerini veya tümünü klavyeden girerek ya da düşey ok veya sayfa tuşları ile aradığı dersin üzerine gelebilir. ENTER tuşu ile dersi yarıyıl ders listesine ekleyebilir.

Paket listesinden seçme olanağı ancak bölümün paket ders olanağının var olmasına ve bulunulan yarıyılın paket yarıyılına eşit veya büyük olmasına bağlıdır.

Normal otomasyon süreci dışında F3 (-yükle) tuşu ile kayıt dosyasından sadece ilgili öğrenci için alınan derslerin işaretlenmesi veya alınmış derslere not bilgilerinin aktarılması ve F6 tuşu ile yine sadece ilgili öğrenci için kayıta alması gerekli dersleri tampon dosyaya aktarılması olanağı da kullanıcıya sunulmuştur.

Dulp-Krediler Veri Giriş Ekranı

Bu ekrana Kartona hazırlık menüsünün 2. seçeneği ile girilir. Ekran sütun sütun ayrılmış bilgi bloklarından oluşmaktadır. Bunlar soldan sağa doğru sırası ile ders kodu, ders adı, DULP kredileri, DULP kredileri için ağırlık katsayıları, 3. ve 4. sütundan hesap ile elde edilen kredi değerleri ve ders dosyasından okunan kredi değerleri.

Veri girişi, 2. ve son iki sütunda engellenmiş olup diğer sütunlar için serbesttir.

Sütunlar arası geçiş için TAB ve ShTAB tuşları kullanılmalıdır. Sütun içinde ise ENTER veya CTRL ve yatay ok tuşlarına aynı anda basarak hareket etmek mümkündür. Satırlar arasında ise düşey ok tuşları ve sayfa tuşları ile dolaşılabilir.

SİSTEM DOSYALARI

Öğrenci işleri otomasyonunda kullanılan dosyalar ve dosyaların nasıl isimlendirildikleri bu bölümün konusudur.

Tüm dosyalar fakülte kodu, yıl ve sömestir bilgisine göre isimlendirilir. Bu bilgiler de programı çalıştırmak için komut satırında verilen parametrelerdir.

Dosya isimlerini formülize etmek için ff (fakülte kodunu temsilen), tt (yıl bilgisinin son iki rakamını temsilen) ve

s (yaz yarıyılı için Y, kış yarıyılı için K harfini temsilen) kodları kullanılacaktır. Formüllerde yer alan gerçek sabitler büyük harf ile gösterilmiştir.

Sistemde yer alan dosyaları sırasıyla aşağıdaki gibi açıklayabiliriz:

FAKÜLTE DOSYASI.....: Bu dosyada fakülte ve fakülte bölümleri ile ilgili bilgiler yer alır. Kayıt programı tarafından oluşturulur. Dosya ismi formatı:

Ftts.V15

şeklindedir. Örnek olarak Elektrik-Elektronik fakültesinin 1992- 1993 öğretim yılı kış yarıyılı için kullanması gerekli dosya **F93K.V15**, Fen-Edebiyat fakültesinin 1995-1996 öğretim yılı yaz yarıyılı için kullanması gerekli dosya **F96Y.V15** isminde olacaktır.

DERS DOSYALARI.....: Bu dosyalar fakültenin tüm derslerini ve derslerle ilgili isim, kod, kredi gibi bilgilerini içerirler. Yaz ve kış semestirleri için ayrı ayrı oluşturulurlar. Dosya ismi formatı:

Dfftts.V15

şeklindedir. Örnek olarak Elektrik-Elektronik fakültesinin 1992-1993 öğretim yılı için kullanması gerekli dosyalar **D0493K.V15** ve **D0493Y.V15**, Fen-Edebiyat fakültesinin 1995-1996 yılı için kullanması gerekli dosyalar **D0396Y.V15** ve **D0496K.V15** isminde olacaktır.

KAYIT DOSYASI.....: Kayıt dosyası kayıt programı tarafından yaratılan bir dosyadır ve ilgili öğretim yılı ve semestirde fakülteye kayıt olan öğrencilerin kayıt bilgisini saklar. Dosya ismi formatı:

Offtts.V15

şeklindedir. Örnek olarak Elektrik-Elektronik fakültesinin 1992-1993 öğretim yılı kış ve yaz semestirleri için kullanması gerekli dosyalar **00493K.V15** ve **00493Y.V15**, Fen-Edebiyat fakültesinin 1995-1996 yılı kış ve yaz semestirleri için kullanması gerekli dosyalar **00396K.V15** ve **00396Y.V15** isminde olacaktır.

KARTON DOSYASI.....: Karton programının yarattığı bir dosyadır. Öğrenci kartonu ile ilgili bilgileri içerir. Dosya ismi formatı:

Kff.V15

şeklindedir. Örnek olarak Elektrik-Elektronik fakültesinin kullanması gerekli dosya **K04.V15**, Fen-Edebiyat fakültesinin kullanması gerekli dosya **K03.V15** isminde olacaktır.

JURNAL DOSYASI.....: Karton programı tarafından oluşturulan ve operatör modu konumunda kartonda yapılan değişikliklerin aktarıldığı ASCII dosyadır. Dosya ismi formatı:

Jfffts.V15

şeklindedir. Örnek olarak Elektrik-Elektronik fakültesinin 1992-1993 öğretim yılı kış ve yaz sömestirlerinde oluşturulan dosyalar **J0493K.V15** ve **J0493Y.V15**, Fen-Edebiyat fakültesinin 1995-1996 yılı kış ve yaz sömestirlerinde oluşturulan dosyalar **J0396K.V15** ve **J0396Y.V15** isminde olacaktır.

LiSTE DOSYASI.....: Kartona kayıtlı öğrencilerin listesinin kayıt edildiği dosyadır. Bu dosya kayıt programınca kullanılır ve öğrenci numarası ile isim bilgilerini taşır. Dosya ismi formatı:

Lfffts.V15

şeklindedir. Örnek olarak Elektrik-Elektronik fakültesinin 1992-1993 öğretim yılı kış ve yaz sömestirlerinde oluşturulan dosyalar **L0494K.V15** ve **L0494Y.V15**, Fen-Edebiyat fakültesinin 1995-1996 yılı kış ve yaz sömestirlerinde oluşturulan dosyalar **L0397K.V15** ve **L0397Y.V15** isminde olacaktır. İsimlerinden de anlaşılabileceği gibi bu dosyalar bir sonraki öğretim yılına hizmet verecektir.

TAMPON DOSYA.....: Kartona kayıtlı öğrencilerin bir sonraki öğretim yılına kalan ve almaları gerekli derslerin kayıt programında kullanılmak üzere aktarıldığı dosyadır. Dosyada öğrenci numaraları ve her öğrenci için ders listesi yer alır. Dosya ismi formatı:

Tfffts.V15

şeklindedir. Örnek olarak Elektrik-Elektronik fakültesinin 1992-1993 öğretim yılı kış ve yaz sömestirlerinde oluşturulan

dosyalar T0494K.V15 ve T0494Y.V15, Fen-Edebiyat fakültesinin 1995-1996 yılı kış ve yaz smestirlerinde oluřturulan dosyalar T0397K.V15 ve T0397Y.V15 isminde olacaktır. isimlerinden de anlařılacađı gibi bu dosyalar bir sonraki đretim yılına hizmet verecektir.

MEZUNLAR DOSYASI.....: Faklte mezunlarının yer aldıđı dosyadır. Karton dosyası ile aynı bilgi alanlarına sahiptir. Dosya ismi formatı:

Mfffts.V15

řeklindedir. rnek olarak Elektrik-Elektronik fakltesinin 1992-1993 đretim yılı gz ve kış smestiri mezunları M0493K.V15 dosyasına, Fen-Edebiyat fakltesinin 1995-1996 yılı bahar ve yaz smestiri mezunları ise M0396Y.V15 dosyasına aktarılacaklardır.

ZLK DOSYASI.....: đrenci zlk bilgilerinin tutulduđu dosyadır. Karton dosyası gibi yıl ve smestire bađlı olmaksızın faklte bazında aılır. Dosya ismi formatı:

Off.V15

řeklindedir. rnek olarak Elektrik-Elektronik fakltesinin kullanması gerekli dosya M04.V15, Fen-Edebiyat fakltesinin kullanması gerekli dosya M03.V15 isminde olacaktır.

KARTON DURUM DOSYASI: Faklte ile ilgili blmlerin ders kredilerinin yarıyıllara dađılımı, bitirme devi yarıyılları ve yapılması gerekli staj hafta sayılarının saklandıđı dosyadır. Dosya ismi formatı:

Kff.STA

řeklindedir. rnek olarak Elektrik-Elektronik fakltesinin kullanması gerekli dosya K04.STA, Fen-Edebiyat fakltesinin kullanması gerekli dosya K03.STA isminde olacaktır.

DULP-KREDİ DOSYASI..: Ders kredilerinin D(=ders), U(=uygulama), L(=lab.) ve P(=proje) olarak dađılımı ve her dađılım için ađırlık katsayısı bilgisinin yer aldıđı dosyadır. Transkript ıkışında gerekli olmaktadır. Dosya ismi formatı:

KRfffts.V15

şeklindedir. Örnek olarak Elektrik-Elektronik fakültesinin 1992-1993 öğretim yılı kış ve yaz s6mestirleri i7in kullanması gerekli dosyalar KR0493K.V15 ve KR0493Y.V15, Fen-Edebiyat fak6ltesinin 1995-1996 yılı kış ve yaz s6mestirleri i7in kullanması gerekli dosyalar KR0396K.V15 ve KR0396Y.V15 isminde olacaktır.

SQL 7ıkış DOSYASI...: 6zl6k bilgilerinin SQL ortamına aktarılması i7in gerekli bir ara dosyadır ve ASCII dir. Dosya ismi formatı

Sfffts.V15

şeklindedir. Örnek olarak Elektrik-Elektronik fak6ltesinin 1992-1993 6ğretim yılı kış ve yaz s6mestirleri i7in 7ıkış dosyaları S0493K.V15 ve S0493Y.V15, Fen-Edebiyat fak6ltesinin 1995-1996 yılı kış ve yaz s6mestirleri i7in 7ıkış dosyaları S0396K.V15 ve S0396Y.V15 isminde olacaktır.

7IKIŞ DOSYASI.....: 7ıkış ortamının y6nlendirilmesi ile 7ıkışlar yazıcı yerine dosya ortamına da verilebilir. Programın bu dosya i7in koyduėu isim 7IKIS.DAT olup 7ıkış ortamı men6s6nde kullanıcı tarafından deėiştirilebilir.

EK.2 KARTON PROGRAMI PROGRAMCI REHBERİ

Ana program CLASS.PAS olup aşağıdaki destek birimlerini kullanmaktadır. Kullanılan birimleri (unitleri) şöyle sınıflandırabiliriz:

1- Standart Derleyici Birimleri:

Dos : işletim sistemi ara birimi.
Printer : Yazıcıyı ara birimi.

2- Turbo Power Toolları:

Bunlar çoğunluğunu ekran G/Ç için kullanılan birimlerin oluşturduğu toollardır. Programlarda kullanılanlar,

TpCrt : Standart Crt biriminin gelişmiş.
TpDos : Standart Dos biriminin gelişmiş.
TpPick : Pencerede liste oluşturma ve seçme birimi.
TpWindow: Pencere oluşturma birimi.
TpMenu : Menü oluşturma birimi.
TpEntry : Ekranda veri girişi sağlayan birim (bu birimin bazı özellikleri değiştirilerek edit anında tüm kontrol tuşları kontrollerinin kullanıcı müdahalesine verilmiştir (TpEntry2).
TpString: String işlemleri için birim,
TpMemChk: Heap kontrol birimi,
Opkey : Tuş kodları.

3- Veri tabanı oluşturmak ve temel veri tabanı fonksiyonlarını yürütmek için birimler:

Types2 : Veri tabanındaki temel tip tanımlamalarının yer aldığı birimdir.

Karaccs2 : Bu toolun orijinali TACCESS dir. Yalnız içinde kullanılan 'include' tip dosyası ve unitler farklıdır.

4- Destek Birimler:

Programın ana birimlerine destek veren birimlerdir:

Destek1 : Programdaki temel sabit, tip ve değişkenlerin tanımlandığı birimdir.

Destek2 : Ana birimlere destek veren prosedürleri içeren birimlerden ilkidir.

Destek31: Ana birimlere destek veren prosedürleri içeren birimlerden ikincisidir.

5- Ana Birimler:

Bunlar ana programı fonksiyonel olarak oluşturan birimlerdir:

Classeri: Programın kullanacağı temel dosyaları açan ve kapatan rutinlerin yer aldığı birim.

Class31 : Üç aşamalı karton ekranının 3. penceresi ile ilgili birim.

Class2 : Üç aşamalı karton ekranının 2. penceresi ile ilgili birim.

Class11 : Üç aşamalı karton ekranının 1. penceresi ile ilgili birim.

Classr11: Raporlama birimi.

Classsta: Karton durum dosyası ile ilgili birim.

Classoz1: Özlük dosyası ile ilgili birim.

Classdul: Dulp-kredi işlemleri ile ilgili birim.

Tez sırasında yazılan birimler takip eden sayfalarda anlatılmıştır.

DESTEK BİRİMLER

PROGRAM ADI.....: DESTEK1.PAS

KULLANDIĞI BİRİMLER:

```
(** turbo power toolları **)
TpCrt,
TpPick,
TpWindow,
TpMenu,
TpEntry2, (** "TPENTRY" programı, edit olanağında bazı
            tuşların kontrolunu almak için değiştirildi **)
```

```
(** tuş sabitleri tanımı **)
Opkey,
```

```
(** TEMEL veri tabanı tip dosyaları **)
Types2,
```

```
(** veri tabanı fonksiyonları **)
Karaccs2.
```

PROGRAM ADI.....: DESTEK2.PAS

KULLANDIĞI BİRİMLER:

```
(** turbo power toolları **)
TpCrt,
TpWindow,
TpString,
TpDos, Dos,
TpMemChk,
TpEntry2, (** "TPENTRY" programı, edit olanağında bazı
            tuşların kontrolunu almak için değiştirildi **)
```

```
(** TEMEL program parametreleri ve veri yapıları **)
Destek1,
```

```
(** TEMEL veri tabanı tip dosyaları **)
Types2,
```

```
(** veri tabanı fonksiyonları **)
Karaccs2.
```

SUNDUGU RUTINLER...:

```
(* s karakter katarını len uzunluğuna göre ayarlama 1 *)
Function Pad ( s : string;
               len : byte) : string;
```

```
(* s karakter katarını len uzunluğuna göre ayarlama 2 *)
Function Padch ( s : string;
                 ch : char;
                 len : byte) : string;
```

```
(* ekranı FILLCHR karakteri ile doldurur *)
Procedure fillscr;
```

```
(* video konfigürasyonunun saklanması *)
Procedure InitConfig ( S : string);
```

```
(* video konfigürasyonunun kontrolü *)
Procedure CheckConfig;
```

```
(* n (1..maxbölüm) olmak üzere fakültenin ilgili
   bölümünün adını verir *)
Function GetBolumAdi ( n : word) : String;
```

```
(* Evet/Hayır cevabı isteyen kontrol prosedürü
   Evet-TRUE, hayır-FALSE *)
Function MyYesOrNo2 ( Prompt : string;
                     defchar: Char) : Boolean;
```

```
(* ekranın belirli satırını temizleme *)
Procedure Clear ( n,
                  at : Byte);
```

```
(* S'deki yazıyı ekranın mesaj satırına yazıp uyarı sinyali
   verir ve tuş bekler *)
Procedure uyari ( S : String;
                 y : byte);
```

```
(* pencere açar *)
Procedure myWindow (Var W : WindowPtr;
                    X1, Y1,
```

100

```
        X2, Y2 : byte;
        attr1,
        attr2,
        attr3,
        attr4 : byte;
        Header : String;
    Var D      : WinDrawRec);
```

(* pencere kapat *)

```
Procedure killmywindow (Var W      : WindowPtr;
                        Var D      : WinDrawRec);
```

(* ekrandan karakter katarı okutan prosedür *)

```
Procedure GetString    (    P      : Str80;
                        Var S      : String; Y, X, Len
                        :Byte);
```

(* 'S' numarasının, 'f' fakültesine ait olup olmadığının kontrolü *)

```
Function JustifyNO      (var f      : fakulterec;
                        S      : string) : boolean;
```

(* 'S' ders kodunun 'f' fakültesine ait olup olmadığının kontrolü *)

```
Function JustifyKOD      (var f      : fakulterec;
                        S      : string) : boolean;
```

(* fakülte numarasından bölüm numarasını veren fonksiyon *)

```
Function No2bolum      (var f      : fakulterec;
                        S      : String) : Word;
```

(* yazıcının açık/online kontrolü *)

```
Function PrinterOK      : Boolean;
```

(* çıkış ortamına yazma prosedürleri*)

```
Procedure Println (s: string);
```

```
Procedure Print (s: string);
```

(* klavye tampon alanını temizler *)

```
Procedure FlushKeys;
```

(* Ekranı, SCREENP isimli değişkende adresi bulunan alana saklar *)

```
Procedure savescr;
```

```
(* SCREENP isimli deðiřkende adresi bulunan alandan
   ekranı alır *)
Procedure restorescr;
```

```
(* index dosyasını listeler ve listeden seçme
   işlemini sağlar *)
Procedure GetIndexFile(      InFile : IndexFile;
                             dtfile : DataFile;
                             Var   _s;
                             Var   Choice : LongInt;
                             Var   exitfl : Boolean;
                             X1, Y1,
                             X2, Y2 : Integer;
                             H      : string;
                             p, p1  : ProcType;
                             Var   rec;
                             Len    : Byte;
                             Msg    : Boolean);
```

```
(* program parametrelerinin kontrolu ve
   standart dosya isimlerinin atanması *)
Function InitParam: boolean;
```

```
(* ekran temizler ve İTÜ logosunu çıkartır *)
Procedure Temizle;
```

```
(* uyarı ses sinyali verir *)
Procedure WarningSound;
```

```
(* 'S' karakter katarının sağdan ilk 'l' adedini verir *)
Function Right (s: string; l:byte): string;
```

```
(* 'S' deki Türkçe harfleri kaldırır *)
Function Ing(s: string): string;
```

PROGRAM ADI.....: DESTEK31.PAS

KULLANDIĞI BİRİMLER:

```
(** turbo power toolları **)
TpCrt,
TpMenu,
TpDate,
TpWindow,
TpPick,
```

**T.C. YÜKSEKÖĞRETİM KURULU
DOKÜMANTASYON MERKEZİ**

```

TpString,
TpDos, Dos,
TpEntry2, (** "TPENTRY" programı, edit olanağında bazı
            tuşların kontrolunu almak için değiştirildi
            **)

```

```

(** TEMEL veri tabanı tip yapıları **)
Types2,

```

```

(** tuş kodları **)
Opkey,

```

```

(** veri tabanı fonksiyonları **)
Karaccs2,

```

```

(** diğer destekler **)
Destek1,
Destek2;

```

SUNDUGU RUTINLER...:

```

(* karton ders bilgileri arasında karşılaştırma yapar *)
Function modified_ogr (Var kar1,kar2 : ogrderstype):
Boolean;

```

```

(* karton kayıtları arasında karşılaştırma yapar *)
Function Modified_karton (Var kar1,kar2 : kartonrec):
Boolean;

```

```

(* yarıyıl kayıtları arasında karşılaştırma yapar *)
Function Modified_yyil (Var kar1,kar2 : yyiltype):
Boolean;

```

```

(* karton durum kayıtları arasında karşılaştırma yapar *)
Function Modified_kartonsta (Var kar1,kar2 :
kartonstarec): Boolean;

```

```

(* karton kaydına ilk değerleri atar *)
Procedure init_karton (Var karton: kartonrec);

```

```

(* özlük kaydına ilk değerleri atar *)
Procedure init_özlük(var özlük: özlükrec);

```

```
(* kontrollu kayıt etme ve kayıt silme *)
Procedure kayıt1;
Procedure kayıtsil1;
```

```
(* asıl kayıt etme ve silme prosedürleri *)
Procedure kayıt;
Function kayıtsil: Boolean;
```

```
(* ders ve dulp listesinden ilgili dersin kredisini alma *)
Function kredi_from_derslist (kod: str8; derslist:
derslistptr; yyil: byte; var sıra: word):real;
```

```
Function kredi_from_dulplist (kod: str8; var sıra:
word):real;
```

```
(* ders listesinden ilgili dersin adını alma *)
Function dadi_from_derslist (kod: str8; derslist:
derslistptr; yyil: byte; var sıra: word):str30;
```

```
(* ders listesinden ilgili dersin dulp kredilerini alma *)
Function dulp_from_dulplist (kod: str8; var sıra:
word):str15;
```

```
(* karton ekranı 3.penceresinde bir satırın tamamen
dolu olup olmadığının kontrolü *)
Function Bossatır(dersno: byte): boolean;
```

```
(* yarıyıla ait ders sayısını bulma *)
Procedure findmaxdersno;
```

```
(* toplam kredileri yazdırma *)
Procedure totkrediyaz;
```

```
(* toplam ve verilen kredilerin hesabı *)
Procedure topla_kredi;
```

```
(* yarıyıl ortalamasını hesapla *)
Procedure totyyilkredi;
```

```
(* yarıyıl tamam fonksiyonu *)
Procedure iyyiltamam;
```

```
(* genel yarıyıl tamam fonksiyonu *)
Function yyiltamam(i: byte): boolean;
```

```
(* karton tamam fonksiyonu *)
Function kartontamam (yeter: boolean; var tamamlanan:
byte): Boolean;
```

```
(* karton ekranı 3.penceresinde
notların byte->string dönüşümü *)
Procedure notlar2str;
```

```
(* karton ekranı 3.penceresinde
notların string->byte dönüşümü *)
Procedure str2notlar;
```

```
(* operator değişkenini günceller *)
Procedure operatorONoff3 (OnOff: Boolean);
```

```
(* F3 tuşu ile yükleme ders/not/tüm derslerin yüklenmesi *)
Procedure Yukle3;
```

```
(* değişiklik jurnalleri *)
Procedure jurnal(var k1,k2: kartonrec);
Procedure jurnal2(var k1,k2: yyiltype);
```

```
(* kayıttan alınan ders/notların aktarılması *)
Procedure kayıttan_yukle;
```

```
(* kış yarıyılı ders listesinin oluşturulması *)
Function getderslistkis: boolean;
```

```
(* yaz yarıyılı ders listesinin oluşturulması *)
Function getderslistyaz: boolean;
```

```
(* dulp-kredi ders listesinin oluşturulması *)
Function Getdulplist: boolean;
```

```
(* tüm derslerin (YAZ/KIŞ) kartona aktarılması *)
Procedure TumDersaktar;
```

```
(* mezunlar ile ilgili işlemler *)
Procedure mezunlar;
```

```
(* karton ekranı 3. penceresinde iken ders seçme *)
Procedure derssec3;
```

```
(* karton ekranı 3.penceresinde iken paket ders seçme *)
Procedure PaketSec1;
```

```
(* tüm kartona ait kalan derslerin aktarılması *)
Procedure Kalanlar;
```

```
(* karton ekranında 3. pencerede iken kalan
derslerin aktarılması *)
Procedure Kalanlar3;
```

```
(* özet öğrenci bilgisinin kayıt programına
hazırlık için aktarılması *)
Procedure ListeVer;
```

```
(* operatörce yapılan değişikliği status2 bayraklarına
kaydetmek için bit işlemlerini gerçekleyen prosedürler *)
Procedure operatorbayragi;
Procedure SetBit (var v; mainbitpos:byte); (* ilgili biti
birleme *)
Function GetBit (var v; mainbitpos:byte): byte; (* ilgili
biti öğrenme *)
```

```
(* kayda hazırlık1: kalan derslerin dosyadan izlenmesi *)
Procedure KalanGor;
```

```
(* kayda hazırlık2 : öğrenci listesinin izlenmesi *)
Procedure ListeGor;
```

```
(* kartona kayıtlı öğrenciler *)
Procedure KartonGor;
```

```
(* kayıt dosyasını indeksler *)
Procedure RebuiltOgr (var daf: datafile; var inf:indexfile;
name: filename;
var buf: öğrencirec2);
```

```
(* ders dosyasını indeksler *)
Procedure RebuiltDers (name: filename);
```

ANA BİRİMLER

PROGRAM ADI.....: CLASSER1.PAS

KULLANDIĞI BİRİMLER:

```
(** turbo power tool **)
TpCrt,
TpString,
```

```
(** veri tabanı tip yapıları **)
Types2,
```

```
(** veri tabanına erişim metodları **)
Karaccs2,
```

```
(** destek birimler **)
Destek1,
Destek2.
```

SUNDUGU RUTİNLER....:

```
Function dosya_ac      : byte;
Procedure dosya_kapat;
```

PROGRAM ADI.....: CLASS11.PAS

KULLANDIĞI BİRİMLER:

```
(** turbo power tooları **)
TpCrt,
TpDate,
TpWindow,
TpDos,
TpString,
TpEntry2,  (* değiştirilen kısım *)
```

```
(** tuş kodları **)
Opkey,
```

```
(** veri tabanı tip yapıları **)
Types2,
```

```
(** veri tabanına erişim metodları **)
Karaccs2,
```

```
(** destek birimler **)
Destek1,
Destek2,
Destek31,
```

```
(** ana birimler **)
Classoz1,
Classr11,
Class31,
Class2,
```

```
(* standart yazıcı birimi *)
Printer.
```

SUNDUGU RUTİNLER...:

```
(* Karton ilk ekranını görüntüler *)
Procedure edit_karton;
```

PROGRAM ADI.....: CLASS2.PAS

KULLANDIĞI BİRİMLER:

```
(** turbo power toolları **)
TpCrt,
TpDate,
TpWindow,
TpDos,
TpString,
TpEntry2, (* değiştirilen kısım *)
```

```
(** tuş kodları **)
Opkey,
```

```
(** veri tabanı tip yapıları *)
Types2,
```

```
(* veri tabanına erişim metodları *)
Karaccs2,
```

```
(* destek birimler **)
Destek1,
Destek2,
Destek31,
```

```
(* ana birimler *)
Classr11,
Class31.
```

SUNDUGU RUTINLER....:

```
(* karton 2. ekranına 'edit' olanağı *)
Procedure edit_2;
```

```
(* edit2 yi oluşturan program *)
Procedure init_edit2;
```

PROGRAM ADI.....: CLASS31.PAS

KULLANDIGI BİRİMLER:

```
(** turbo power tooları **)
TpCrt,
TpMenu,
TpDate,
TpWindow,
TpDos,
TpString,
TpEntry2, (* değiştirilen tool *)
```

```
(** tuş kodları **)
Opkey,
```

```
(** veri tabanı tip yapıları *)
Types2,
```

```
(** veri tabanına erişim metodları **)
Karaccs2,
```

```
(** destek birimler **)
Destek1,
Destek2,
Destek31,
```

```
(** ana birimler **)
Classr11.
```

SUNDUGU RUTİNLER...:

```
(* karton 3. ekranının 'edit' edilmesi *)
Procedure edit_3;
```

```
(* karton ekranı 3. penceresinin yerleştirilmesi *)
Procedure init_edit3;
```

```
(* 3. ekranda cursor kontrolu *)
Procedure check_loc3;
```

```
(* font düzenleme *)
Procedure check_attrib3;
```

```
(* ekranı düzenleme *)
Procedure Prompt;
```

PROGRAM ADI.....: CLASSSTA.PAS

KULLANDIĞI BİRİMLER:

```
(** turbo power toolları **)
TpCrt,
TpString,
TpEtry2, (* değiştirilen kısım *)
```

```
(** veri tabanı tip yapıları **)
Types2,
```

```
(** veri tabanına erişim metodları **)
Karaccs2,
```

```
(** destek birimleri **)
```

```
Destek1,
Destek2,
Destek31.
```

SUNDUGU RUTINLER...:

```
(* karton durum dosyasının görüntülenmesi *)
Procedure Edit_Sta;
```

PROGRAM ADI.....: CLASSOZL.PAS

KULLANDIĞI BİRİMLER:

```
(** turbo power tooları **)
TpCrt,
Tpmenu,
TpDate,
TpWindow,
TpDos,
TpString,
TpEntry2, (* değiştirilen kısım *)
```

```
(** tuş kodları **)
Opkey,
```

```
(** veri tabanı tip yapıları **)
Types2,
```

```
(** veri tabanına erişim metodları **)
Karaccs2,
```

```
(** destek birimler **)
Destek1,
Destek2,
Destek31.
```

SUNDUGU RUTINLER...:

```
(* özlük dosyası ile kullanıcı arasında arabirimi sağlayan
ve SQL için ASCII dosya oluşturan temel program **)
Procedure edit_Ozluk;
```

```
(* karton'a ait özlük kaydının bulunması *)
Procedure özlük_getir (no: str8);
```

```
(* özlük'e ait karton'un bulunması *)
Procedure karton_getir (no: str8);
```

PROGRAM ADI.....: CLASSDUL.PAS

KULLANDIĞI BİRİMLER:

```
(** turbo power toolları **)
TpCrt,
TpString,
TpEntry2, (** değiştirilen birim **)
```

```
(** veri tabanı fonksiyonları **)
Karaccs2,
```

```
(** destek birimler **)
Destek1,
Destek2,
Destek31.
```

SUNDUĞU RUTİNLER...:

```
(** DULP ve kredi değerlerini inceleyen ana prosedür **)
Procedure DULPincele;
```

PROGRAM ADI.....: CLASSR11.PAS

KULLANDIĞI BİRİMLER:

```
(* standart yazıcı birimi *)
Printer,
```

```
(** turbo power toolları *)
TpCrt,
TpDate,
TpWindow,
TpDos,
TpString,
```

```
TpMenu,
TpEntry2, (* değiştirilen kısım *)
```

```
(** tuş kodları **)
Opkey,
```

```
(** veri tabanı tip yapıları **)
Types2,
```

```
(** veri tabanına erişim metodları **)
Karaccs2,
```

```
(** destek birimler **)
Destek1,
Destek2,
Destek31;
```

SUNDUGU RUTİNLER....:

```
(* çıkış ortamı kontrolü *)
Procedure CikOrtKontrol;
```

```
(* transkript dilini değiştirme *)
Procedure IncTransDil(Var Value; FieldID : Word; Factor :
                      Integer; Var St : String);
```

```
(* transkript çıkışı alma *)
Procedure TransCikis;
```

```
(* yarıyıl karnesini alma *)
Procedure YyilKarne2;
```

```
(* yukarıdaki işlemleri de içeren raporlama ana prosedürü*)
Procedure Raporlama;
```

PROGRAM ADI.....: CLASS.PAS (Ana program)

KULLANDIĞI BİRİMLER:

(** turbo power toolları **)

Tpcrt,
Tpmenu,
Tpwindow,
Tpstring,
Tpentry2, (* değiştirilen tool *)
Dos,

(** veri tabanı tanımı ve erişimi **)

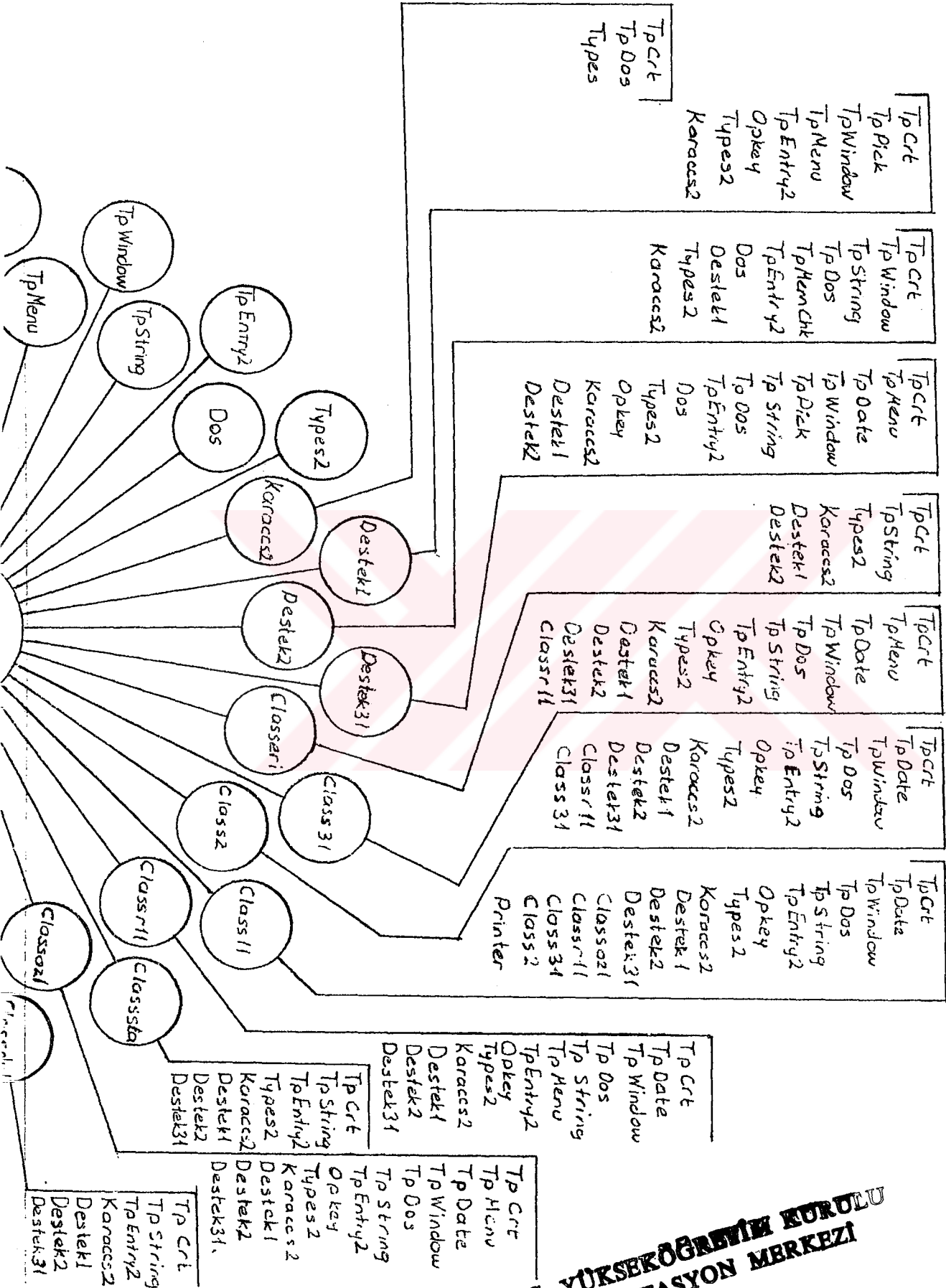
Types2, (* veri tabanı tip tanımlamaları *)
Karaccs2, (* veri tabanına erişim metodları *)

(** destek birimler **)

Destek1, (* tüm sabitler, tipler ve değişkenler *)
Destek2, (* destek prosedürler -1- *)
Destek31, (* destek prosedürler -2- *)

(** ana birimler **)

Classeri, (* veri tabanı dosyalarını açar *)
Class31, (* karton ekranı 3. pencere olanakları *)
Class2, (* karton ekranı 2. pencere olanakları *)
Class11, (* karton ekranı 1. pencere olanakları *)
Classr11, (* raporlama birimi *)
Classsta, (* karton durum ekranı olanağı birimi *)
Classozl, (* özlük ekranı olanağı birimi *)
Classdul{p}; (* DULP-kredi edit işlemleri birimi *)



ÖZGEÇMİŞ

1968 yılında Batı Almanya'da doğan Hakan KAZAZ, ilk öğrenimini A. Merter ilk okulunda, orta öğrenimini A. Merter orta okulu ve Pertevniyal Lisesi'nde tamamlamıştır. 1986 yılında İ.T.Ü. Elektrik-Elektronik Fakültesi Kontrol ve Bilgisayar bölümünde öğrenime başlamış ve 1990 yılında mezun olmuştur. Aynı yıl aynı bölümde yüksek lisans eğitimine başlayan Hakan KAZAZ, halen İ.T.Ü. Bilgi İşlem Merkezinde çalışmaktadır.

