YILDIZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

Mehmet Ahlatçıoğlu Abdussamet Haşoğlu Behiç Çağal

VERİTABANI MODELLERİ VE HİYERARŞİK VERİ TABANINDAN İLİŞKİSEL VERİ TABANINA DÖNÜŞÜM

Mat.Müh. Selma KARACABEY

F.B.E. Matematik Mühendisliği Anabilim Dalında Hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı : Prof.Dr. Behiç ÇAĞAL

İSTANBUL,1997

İÇİNDEKİLER

ŞEKİL LİSTESİ

TABLO LİSTESİ

KISALTMALAR LİSTESİ

DBMS	Database Management System
DL	Data Language
DML	Database Management Language
FK	Foreign Key
MIS	Management Information Systems
NN	Not Null
OODBMS	Object Oriented Database Management System
OOPL	Object Oriented Programming Language
PK	Primary Key
SQL	Structured Query Language
TP	Transaction Processing
U	Unique
UID	Unique Identifier

TEŞEKKÜR

Bu tezin hazırlanmasında bana yön veren tez danışmanım sayın Prof.Dr. Behiç ÇAĞAL Bey'e, konu seçiminde yardımcı olan Sümerbank A,Ş. Sistem Geliştirme Müdürü sayın Mesut SOYUMERT Bey'e, yazım ve basımında bana destek olan İlkay OĞUZ, Verda Uysal GAZOZCU, Ahmet DEMİRAĞ ve diğer arkadaşlarıma; tezi gerçekleştirmek için birçok olanaklarını kullandığım Sümerbank A.Ş'ye ve tüm öğrenim hayatım boyunca hiçbir fedakarlıktan kaçınmayan aileme teşekkür ederim.

İstanbul, 1997 Selma KARACABEY

Veritabanı Modelleri ve Hiyerarşik Veritabanı Modelinden İlişkisel Veritabanı Modeline Dönüşüm

ÖZET

Veritabanı, birçok uygulamanın erişip, düzeltme yapabileceği, yapısallaşmış ve sistematik olarak yönetilen verilerden oluşmaktadır.Veritabanına ilk yönelme, programlarında kullandıkları verileri rahat bir şekilde yönetmek isteyen programcılardan gelmiştir.Bazı uygulamalarda veriler o kadar önemli olmaktadır ki, bunları yönetmek ve düzeltme yapmak için özel programlar geliştirilmektedir.

Kullanılmakta olan başlıca üç veritabanı modeli vardır:

•Hiyerarşik
•İlişkisel
•Ağ(Network)

İlk veritabanları, çeşitli ve güvenilemeyecek veri birikimlerinden oluşan düz dosya veritabanları idi. Bu tür veritabanlarının kullanıldıklaır zamanlarda bilgisayar programcıları kendi verilerini depolama ve bakımını yapma eğilimindeydiler. Ortak olan herhangi bir veri , veri tekrarı oluşturmaktaydı ve bu da potansiyel olarak birçok probleme yol açmaktaydı. Tekrarlanan veriden dolayı , depolama alanı büyük ölçüde boşa harcanmaktaydı. Programcılar, verilerin tam olarak nerede ve nasıl saklandığını bilmek zorundaydı ve tekrarlanmış verinin bakımı da çok zordu. Tekrarlanan verinin bir yerde düzeltilmesi depolandığı her yerde de düzeltilmesini gerektirmekteydi.

Transaction tabanlı veritabanları organizasyonal aktiviteleri destekler.Banka ve finansal kurumlar gibi organizasyonlar, işlerini yürütmek için gerekli bilgiyi depolamak amacıyla bunları kullanmaktadırlar. Bu veritabanları, organizasyonların, performanslarını geliştirme veya kuvvetlendirme amacıyla yaptıkları aktiviteleri analiz etmek için gerekli bilgiyi sağlamak amacıyla da kullanılabilir.İşte tüm bu organizasyonal ihtiyaçlar, transaction veritabanlarının ortaya çıkmalarına ve gelişmelerine neden olmuştur.

Hiyerarşik veritabanı modelinde veriler, tek köke sahip ağaç yapısı şeklinde saklanır. Ağacın en üstü kök veya ana olarak adlandırılır.Ağaçtaki her düğüm bir veri nesnesini ve bağlantılar da ana-çocuk ilişkilerini temsil eder.Değişik dallar arasında ise çapraz ilişki yoktur.

Hiyerarşik veritabanının yapısı, hiyerarşideki ilişkileri takip etmek suretiyle verinin çabuk bir şekilde bulunmasını sağlar.

Bir veri birden fazla kategoriye ait olabilir, bir başka deyişle veriler arasında çoka-çok bir ilişki bulunabilir.Hiyerarşik veritabanında bu, veriyi hiyerarşinin birden fazla dalında tekrar etme şeklinde açıklanabilir.Yukarda ,düz dosyalarda anlatıldığı gibi veri tekrarı bakım problemleri yaratır.

Network veritabanı modeli, veritabanı kayıtlarının, veritabanındaki herhangi bir diğer kayıtla ilişki kurmasını sağlar.Bu model, daha çok ilişkisel modele benzer.İkisi arasındaki en önemli farklılık, ilişkileri çözmek için ilişkisel modelin veri değerlerini kullanmasına karşılık, network modelin göstergeler kullanmasıdır.Network model veriyi kayıtlar ve kümeler şeklinde yönetir.

Network veritabanı, hiyerarşik veritabanındaki veri tekrarı problemini yok eder, fakat halen veritabanı yapısını değiştirmeyle ilgili problemlere sahiptir. Değişmesi gereken veritabanı kümeleri büyük ölçüde veritabanının yeniden-yapılandırılmasına sebep olur. Bununla birlikte, yeni kümeler, yeni veri öğeleri yaratmayla ve onları önceden varolan verilerle link etme yoluyla, kolaylıkla ilave edilebilir.

1970 yılında Codd'un matematiksel küme teorisini uygulayarak kurduğu, İlişkisel veritabanı modeli, tüm veritabanı modelleri arasında en basit ve en muntazam yapıda olanıdır. Ayrıca ticari olarak en çok kullanılanıdır. Bu veritabanı modelindeki esas kavram, tablo olarak düşünülebilen ilişkilerdir.Tablonun her sütunu, bu sütuna girişler için veri-değer tanımı oluşturan özelliklerdir.Her tablonun satırları, depolanmakta olan tek veri nesnelerini temsil eden kayıtlardır.

Bir ilişkisel veritabanı, birçok sayıda tablodan oluşmaktadır.Tablolar arasındaki ilişkiler, aynı tipteki verileri içeren ve başka bir tablodaki sütunla aynı tanımda olan sütunu içeren bir tablo tarafından tanımlanmaktadır.Bu tabloda bulunan veriler, ilişkili tablolardaki verileri aramak için kullanılabilir.

İlişkisel veritabanı tekrarlanmış veri oranını azaltır ve bakımı kolaydır.Veri ilişkilerini tanımlamadaki ve verilerin kullanıcılara sunulmasındaki esneklik, transaction tabanlı veritabanları arasında ilişkisel modelin en çok tutulmasını sağlamıştır.

İlişkisel veritabanı modelinin kullanılmasındaki avantajlar, hiyerarşik modelden ilişkisel modele geçişi kaçınılmaz yapmıştır.

Database Models and Conversion from The Hierarchical Database Model to The Relational Database Model

SUMMARY

A database is structured, systematically managed set of data that a number of applications can access and update. The original motivation for databases comes from computer programmers who wanted a convenient way of handling the data that was used in their programs. In some applications, the data became so important that specialized programs were developed to manage and update the data.

There are three main database models in use:

•Hierarchical
•Relational
•Network

The earliest databases consisted of a conglomeration of various pieces of unrelated data is flat file databases. Computer programmers tended to store and maintain their own data. Any common data was duplicated. That had potentially many problems. There was a waste of storage space with duplicated data, the programmers needed to know exactly where and how the data was stored and there was a problem with the maintenance of duplicated data. When it was updated it had to be changed in each place where it was stored.

Transaction based databases, support organisational activities. Organisations such as banks and financial institutions use them to store the information they require to carry out their business. These databases also provide information that can be used to analyse the organisation's activities for the purpose of improving or streamlining its performance. It was these organisational needs that necessitated the introduction and development of transaction databases.

The hierarchical database model, represents data as a single-rooted tree. The top of the tree is called the root or parent.Each node in the tree represents a data object and connections represent a parent-child relationship. There is no cross references between the records in different branches.

The nature of an Hierarchical Database allows data to be found quickly by following relationships through the hierarchy.

It may be found that certain data may belong in more than one category, ie. a many-to-many relationship between data may be found. With the Hierarchical Database this would be implemented by duplicating the data into more than one branch of the hierarchy. As mentioned above duplication brings about maintenance problems.

The network model for databases allow records of the database to refer to any other record in the database. This model is much like the relational model, the major difference being that the network model use pointers to encode the relationships whereas the relational model use data values.The network model manipulates data in terms of records and sets.

Network Databases eliminate the data duplication problem of the Hierarchical Database but still have problems with changes to the database structure. Database sets that need to be changed will cause major re-structuring of the database. New sets, however, can be added easily by creating new data items and linking them with the already existing data.

Of all the database models the relational model, which was introduced by Codd in 1970 and used mathematics set theory to describe the relationships between data, has the simplest and most uniform structure. It is also the most commercially widespread.The primary concept in this database model is the relation which can be thought of as a table. The columns of each table are attributes which define the data-value domain for entries in that column. The rows of each table are tuples representing individual data objects that are being stored.

A Relational Database is made up of number of tables. Relationships between tables are defined by a table containing a column that contains the same type of data and in the same domain as a column in another table. The data found in these columns is used to look up data in related tables.

The Relational Database also reduces the amount of duplicated data and it is easy to maintain. Flexibility in defining the data relationships and flexibility in the presentation of the data to the user makes the relational database clearly superior to all other data storage methods for transaction based databases.

The advantages of using the Relational Database model makes the conversion from the Hierarchical Database Model to the Relational Database Model unavoidable.

GİRİŞ

Bu tezde, veritabanlarının evriminden, günümüzde kullanılmakta olan başlıca veritabanı modellerinden ve Hiyerarşik Veritabanı Modeli'nden, bu modele göre daha avantajlı olan İlişkisel Veritabanı Modeli'ne geçişten bahsedilmektedir.

Çeşitli amaçlar için bilgilerin işlenme gereksinimi, veri kavramını ortaya çıkarmıştır. Bu verilerin bir arada toplanıp, depolanması, ve depolanan verilerin bakım ve yönetim ihtiyaçları, veritabanı, veritabanı yönetim sistemi ve veritabanlarının üzerinde kurulacağı veritabanı cihazları kavramlarını ortaya çıkarmıştır.Veritabanı, veri tabloları ve diğer nesneleri kapsayan bilgilerin toplamı; sistematik olarak yönetilen veriler kümesidir. Bir veritabanındaki tablolar ve nesneler, bilgilere ulaşmaya, yani arama, sıralama ve kullanım sırasında veriyi tekrar elde etmeye yardımcı olacak şekilde düzenlenmiş ve sunulmuştur. 1970'lerin başlarında, bilinen veri modellerinin başlıcaları: Hiyerarşik Veri Modeli, Network Veri Modeli ve İlişkisel Veri Modeli idi.1970 yılında Codd'un ilişkisel cebri uygulayarak kurduğu, İlişkisel Veritabanı Modeli, tüm veritabanı modelleri arasında en basit ve en muntazam yapıda olanıdır. Ayrıca ticari olarak en çok kullanılanıdır. İlişkisel modelin, kendinden önceki modellere göre daha avantajlı olmasından dolayı, bu modelin kullanılması kullanıcı ve programcılara büyük kolaylıkar sağlamaktadır. Tüm bu sebeplerden dolayı Hiyerarşik Veritabanı Modeli'nden İlişkisel Veritabanı Modeli'ne geçiş her bakımdan daha avantajlıdır.

I. VERİTABANI MODELLERİ

1.1 VERİTABANI İŞLEME İÇİN GEREKSİNİM

Birçok sebepten dolayı, bilgi sistemleri müfredatında veritabanı işleme en önemli konulardan birisidir. İlk olarak, bu teknoloji işletmenin her seviyesinde kullanılmaktadır. Bireyler işlerini kolaylaştırmak için kendi şahsi veritabanlarını kullanmaktadırlar. Departmanlar, departmanlardaki bireylerin aktivitelerini girmek için işgrubu veritabanlarını kullanmaktadır ve şirketler birçok departmandaki işi kaynaştırmak için büyük organizasyonal veritabanlarını kullanırlar. 1970'lerin başlarında, bilinen veri modellerinin başlıcaları: Hiyerarşik Veri Modeli, Network Veri Modeli ve İlişkisel Veri Modeli idi.

1.2 VERİTABANI TANIMI

Veritabanı, veri tabloları ve diğer nesneleri kapsayan bilgilerin toplamıdır. Bir veritabanındaki tablolar ve nesneler, bilgilere ulaşmaya, yani , arama, sıralama ve kullanım sırasında veriyi tekrar elde etmeye yardımcı olacak şekilde düzenlenmiş ve sunulmuştur. Veritabanları kendileri için ayrılmış cihazlar üzerinde saklanır.

1.3 VERİTABANI MODELLERİ

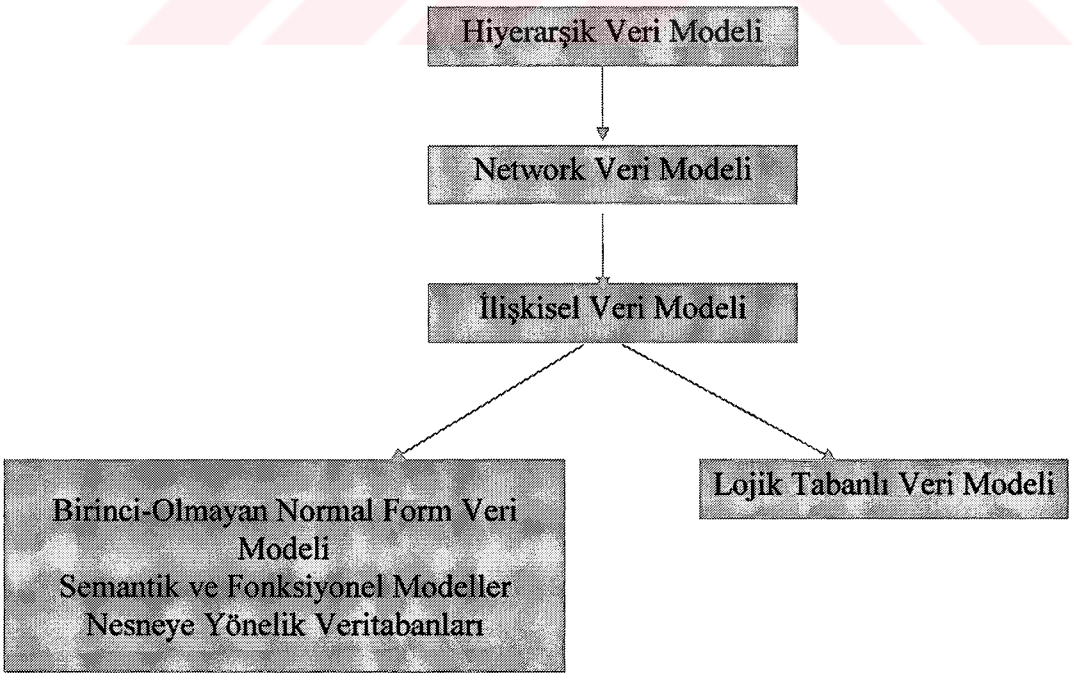
1.3.1 Amacı

Çoğumuz araba ve uçak modellerine aşinayız. Bu tür modeller gerçek dünyada var olan nesnelerin *fiziksel* gösterilimidir. Mühendisler bu tür modelleri gerçekte var olan olay ve nesneleri daha ucuz ve daha çabuk bir şekilde test etmek için kullanıyorlar. Gerçek-yaşam olaylarını test etmek için ayrıca *soyut* modeller de kullanırlar. Örneğin, bir rüzgar

tünelini tasarımlamak isteyen mühendis, tünelin performansını tanımlamak(model kurmak) amacıyla bir seri matematiksel denklem kuracaktır. Bununla birlikte, bu tür model kurma yalnızca fen ve mühendisliğe özgü değildir. Gerçekte, bilgisayar kullanıcıları da her zaman bu tür modellerle uğraşırlar.

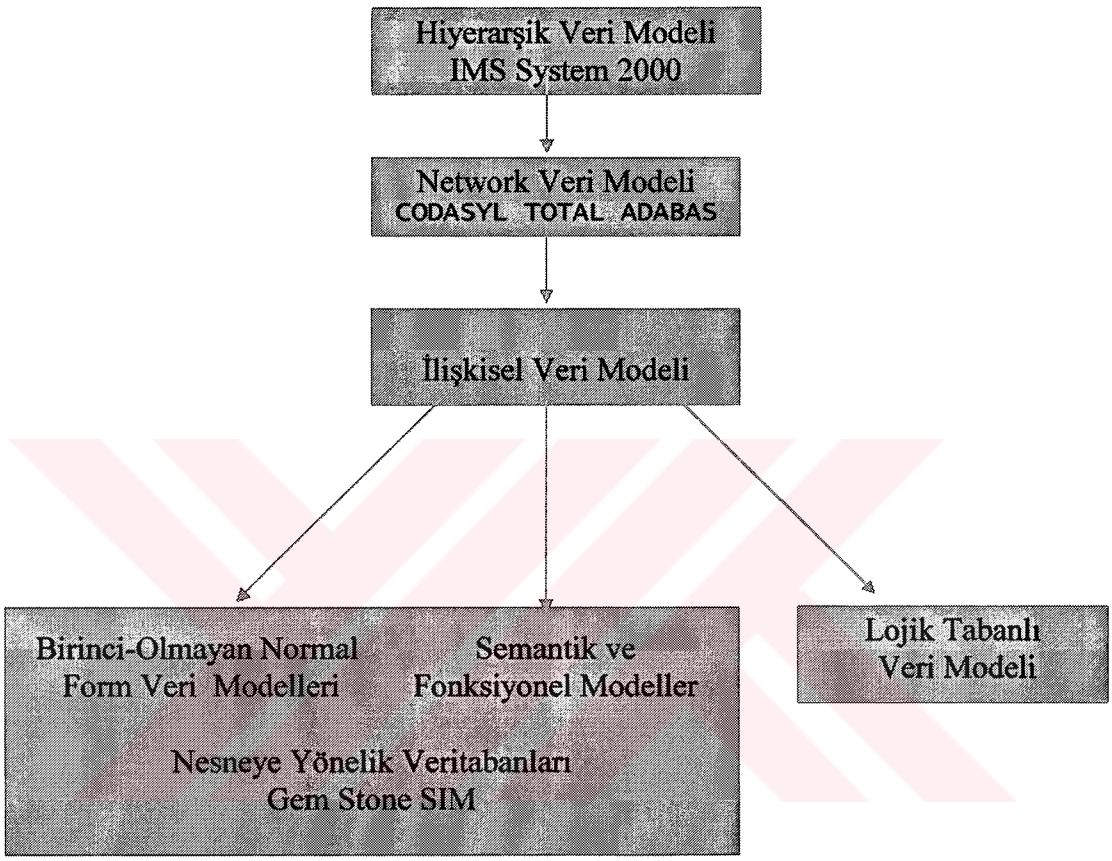
Veritabanı dünyasında, **DBMS (Data Base Management System-Veri Tabanı Yönetim Sistemi)** modelleri, verilerin nasıl saklandığını, kurulan ilişkileri, kullanılan ve yönetilen verileri temsil etmemizi olanaklı kılar. Belirli bir model üzerinde kurulu bir **DBMS** , verinin bu **DBMS** içinde organize edilme yolunu gösterir. Amaç model, tasarımcıların ve yöneticilerin bu verileri gözlemleyebilmelerine olanak sağlayacak şekilde, verileri temsil etmelidir. Daha önemlisi, veriye, performans sınırlamalarını ve bilgi gereksinimlerini karşılayacak şekilde ulaşılabilmeli ve ulaşılan veriler verimli bir şekilde yönetilebilmelidir.

1.3.2 Veritabanı Modellerinin Evrimi



Şekil 1.3.2.1 Veritabanı Modellerinin Evrimi

1.3.3 Ticari Veritabanı Yönetim Sistemlerinin Evrimi



Şekil 1.3.3.1 Ticari Veritabanı Yönetim Sistemlerinin Evrimi

1.4 DÜZ DOSYA(FLAT FILE) VERİTABANLARI

İlk veritabanları güvenilmez veriler içermekteydi.Bilgisayar programları, kendi verilerini saklama ve sürdürme eğilimindeydiler. Herhangi ortak bir veri, veri tekrarını oluşturmaktaydı. Bu da potansiyel olarak birçok probleme sebep olmaktaydı. Tekrarlanmış veri, depolama alanının boşuna harcanmasına sebep olmaktaydı, programlar tam olarak verinin nerede depolandığını bilmek zorundaydı ve tekrar eden verinin bakım ve kontrol problemi vardı. Bu veri günlendiği zaman, depolandığı her yerde de değiştirilmek zorundaydı.

1960'larda, oldukça büyük miktarlarda veri depolayan, çok-kullanıcılı, geniş mainframe (anabilgisayar) bilgisayar sistemleri verimli olduğunda, organizasyonlar kendi verilerini mainframe üzerinde, merkezileştirilmiş veritabanlarında depolamaya başladılar. Mainframe üzerindaki tüm kullanıcılar tarafından bu veritabanlarına erişilebilmekteydi, bir tek bilgi havuzu olduğu için her kullanıcı daima en son veriye ulaşmaktaydı.

Verinin bu şekilde merkezileştirilmesiyle, verinin, kayıt diye adlandırılan ayrık parçalar halinde organize edilmesi ve bu kayıtların dosyalarda biriktirilmesi fikri doğdu. Bu şekilde tanımlanan ilk dosya modelinin en basit versiyonu Düz Dosya Veritabanı (Flat File Database)' dır.

Dosyalar, kayıtların sıralı listesini içermektedir. Belirli bir kaydı bulabilmek için, dosyanın, aynı anda tek kayıt okumak suretiyle başlangıçtan istenilen kayıt bulunana kadar aranması gerekmektedir. Kayıt bulunduğu zaman, program bu kayıttaki veriyi kullanmak için onun nasıl depolandığını bilmek zorundadır. Ayrıca, depolanmış verinin çeşitli parçaları arasında hiç bir ilişki görülmemektedir. Veri ilişkileri kullanıcı tarafından veya veriye erişen program tarafından bilinmek zorundadır.

Çoğu programcılar ve kullanıcılar, Düz Dosya Veritabanındaki veriye müdahale edebildiği için, her biri veritabanına yazdığı verinin doğru ve geçerli olduğunu garanti

etmelidir. Bu, veri kontrol kodunun bir çok progam içinde tekrarlanacağı anlamına gelmektedir. Bu ise veri bütünlüğü problemini vurgular.

Bir Düz Dosya Veritabanında, kayıtlar belirli bir düzen olmaksızın ve sırasal olarak saklandığı için belirli bir kaydı çabuk bulma yolu yoktur, kayıtları sıralamak da zordur. Bu ise tüm dosyanın okunmasını, ve istenilen şekilde sıralanmasını gerektirmektedir. Büyük veritabanlarında bu, yoğun ve zaman kaybettiren bir işlemdir.

İndeksli dosyalar, Düz Dosya Veritabanlarının gelişimi olarak tanıtıldı. İndeksler, veritabanı dosyalarına çok benzeyen dosyalardır, fakat bir kaydın alanlarının bir alt kümesini içermektedir ve belirli bir şekilde sıralanmaktadır. Ayrıca, karşılık gelen kaydın nerede bulunacağını da göstermektedir.Bunun için aramada daha hızlıdır ancak veritabanına bir seviye daha fazla karmaşıklık katmaktadır.

Düz Dosya Veritabanlarıyla ilgili büyük bir problem, veritabanı yapısını değiştirmektir. Dosyaya bir alan eklemek tüm dosyayı okuyup yeni yapısına kopyalamak anlamını taşımaktadır.

1.5 TRANSACTION PROCESSING (HAREKET İŞLEME) VERİTABANLARI

Veritabanı işleme, transaction processing uygulamaları ile başlar. 1960'ların başlarında ve ortalarındaki veri biriktirilmesine dair çabaların çoğu başarısız olmuştu. Sistem tasarımcıları, dosya işlemenin yetersiz olduğunu, kayıt ilişkilerini işleyebilecek ve tanımını yapabilecek bazı veri modelleme tiplerine ihtiyaçları olduğunu biliyorlardı.

1.5.1 Transaction Tabanlı Veritabanlarının Özellikleri

Transaction tabanlı veritabanları organizasyonal aktiviteleri destekler. Bankalar ve finansal kurumlar gibi organizasyonlar, işlerini yürütebilmek için gerekli bilgileri depolamak amacıyla onları kullanırlar. Bu veritabanları ayrıca işletmenin performansını artırmak amacıyla organizasyonun aktivitelerini analiz etmek için de gerekli bilgiyi sağlar. Tüm bu gereksinimler transaction veritabanlarının tanıtılıp geliştirilmesini sağlamıştır.

Bir kayıt, kaydın sakladığı bilgiyi tanımlayan alan (field) adı verilen birçok özellikten oluşmaktadır. Her bir dosya ise, alanları içeren kayıtlardan oluşmaktadır. Bir veritabanı ise bu dosyaların toplamından oluşur.

1.5.2 Transaction Processing

Transaction processing sistemlerini anlamak için dalgalanan işletme gereksinimlerini gözönüne almak gerekmektedir. Mallar alınıp satılır, müşteriler gelip gider, para kazanılır ve harcanır, zaman geçer. Bazı noktalarda birisi (belki bir patron, bir vergi acentesi, bir işçi) sorar: " Bu işletmenin durumu nedir?" , "Geçen ay ne kadar para kazandık?", veya "Geçen ayki seyahat giderlerin ne kadardı?".

Yukardaki gibi sorulara daha gerçekçi cevap verebilmek için, iş, hesaplama ve operasyonal kayıtları gerektirmektedir. İsimler toplanır, mallar araştırılır, para sayılır vs. ve daha sonra ölçümler biriktirilir. Bazı işletmelerde bu ölçümlerin çoğu **operasyonel veritabanı** 'nda saklanırlar.Böyle bir veritabanı, işin operasyonel yönünün kullanıcı modelinin bir **model** 'idir. Bu model, organizasyondaki durumları temsil eder. Bununla birlikte işletme, dinamiktir ve kondisyonlar değişmektedir. Böyle olduğu takdirde modelin de değişmesi gerekmektedir. Aksi takdirde, model, işletmenin yükünü taşıyamayacaktır.

Bir **transaction (hareket)**, bir olayın işletmedeki gösterilimidir. Örneğin, siparişler, faturalar, transferler ve bunun gibi diğer işlemler.Veritabanına hareket işleme genel anlamda Transaction Processing (TP) olarak bilinir. Bir transaction başlatıldıktan sonra veritabanında kullanılan objeler üzerinde istenilen işlemler yapılabilir. Örneğin, bir tabloya kayıt ekleme, silme ve günleme gibi veritabanı işlemlerininin biri veya birkaçı birlikte kullanılarak çeşitli işlemler yapılabilir. SQL veritabanı kullanılarak yapılan transaction processing'ler COMMIT edilene kadar veritabanına işlenmez ve bu ana kadar olan tüm değişiklikler ROLLBACK yapılarak geri alınabilir (Kroenke,1994).

1.5.3 Transaction Processing'in Karakteristikleri

- *Kısa zamanda hızlı performans sağlamalıdır.*
- *Güvenilir* olmalıdır. Etkili olması için bir transaction işlemenin başarısız olma oranı çok düşük olmalıdır. Dahası, eğer sistemde arıza meydana gelirse, çabuk ve doğru recovery(önceden yapılan çalışmaları kurtarma) şarttır.
- Diğer veritabanı uygulamalarının aksine, *esneklik* , transaction processing uygulamaları için daha az önemlidir. Gerçekte, esneklik bazen istenmemektedir. İşletmeler, her transaction'ın aynı şekilde işletilmesini istemektedir. Eğer transaction uygulamaları esnek olsalardı, standart olmayan operasyonlar için çok fazla fırsat olacaktı.

- Transaction processing, işletme operasyonlarını sağladığı için, *kontrol,* önemlidir.
- Transaction Processing uygulamaları, veritabanlarının kısıtlanmış bir görünüşüne sahiptir.
- Operasyonel bir veritabanı, bir işletme veya organizasyonun modelidir. Böyle olduğu için de oldukça değerlidir ve korunması gerekmektedir. Sadece yetki verilmiş kullanıcılar veri tabanına ulaşabilmeli ve sadece yetki verilmiş işlemleri yerine getirmelidirler. DBMS ürünlerine yönelik transaction process'lerin çoğu güvenlik gereksinimlerini sağlar.
- Transaction Processing uygulamalarının bulunduğu çevre, tek ve belirli olmasına, veya veritabanı uygulamalarına karar vermesine göre önemli ölçüde değişir.

1.5.4 Transaction Veritabanlarının Özellikleri

- Veriye erişim hızlıdır.
- Veri tekrarı yoktur.
- Veriler arasındaki ilişkiler veritabanının kendisinde de tanımlanır.
- Verinin saklanma metodu ve özellikleri veritabanının içine yöneliktir.

Birinci özelliğin çok istenilir olduğu açıktır. Programlar çok sayıda kayıtlarla işlem yapmaya başladığında, veritabanına erişim hızlı olmalıdır, böylece kullanıcının veritabanı sorgulamaya verdiği cevap kabul edilebilir olur.

Fazla yer kaplamasının yanısıra, tekrar edilmiş verinin yönetimi de zordur. Bu veri değiştirilmek istendiğinde, bu verinin kullanıldığı her yerde değiştirilmek zorundadır.

Son iki özellik, bir programın veritabanına erişmesi için gerekli bilginin içe dönüklüğünü azaltmayı amaçlar. Birden fazla programla aynı veritabanına erişildiğinde aynı veri birden fazla şekilde ifade edilebilir.

1.5.5 Önceden Tanımlanmış İlişkiler

Hiyeraraşik ve Network Modelin her ikisi de ilişkilerin önceden tanımlanmasını gerektirmektedir. Yani, veritabanı kullanılmadan önce, tüm ilişkiler tahmin edilmeli ve tanımlanmalıdır. Bu modellerde, varolan bir veritabanına ilişkiler eklemek, ilişkisel modele göre oldukça zordur. Bu endişelerin nedeni, ilişkileri temsil etmek için kullanılan metoddur.

İlişkilerin veri değerleriyle kurulmuş olduğu İlişkisel Modelin aksine, Hiyerarşik ve Network modeller ilişkileri, ayrı veri yapılarıyla, örneğin indeksli ve linkli listeler gibi, tutarlar. İlişkiler temsil edilmeden önce DBMS ile bu veri yapıları kurulmalıdır.

Transaction processing uygulamaları için, veri yapıları arasındaki ilişkilerin gösterilimi, veri ilişkilerini saklamaya göre iki avantaja sahiptir:

Birinci avantaj, performans daha iyi olabilir (en azından var olan teknolojilerle). İlişkiler önceden tanımlandığında, veri yapıları seçilebilir ve iş yüküne göre ayarlanabilir. İlişkisel DBMS ürünlerinin çoğu iyi-ayarlanmış Hiyerarşik ve Network DBMS ürünlerinin performansını seçmeye başlayamaz.

İkinci avantajı ise kontrol' dür. Transaction processing uygulamaları için, ilişkilerin önceden tanımlanmasının çok cazip olabilmesi bir kısıtlama olabilir. Transaction processing 'in standartlaştırılması gerekebilir.

1.6 HİYERARŞİK(HIERARCHICAL) VERİTABANI MODELİ

1.6.1 Hiyerarşik Veri Yapıları

Hiyerarşik veritabanı modeli, yukarda bahsedilen Düz Dosya Veritabanı problemlerinin üstesinden gelmek için geliştirildi. Ek olarak, Hiyerarşik veritabanı çeşitli veri parçaları arasındaki ilişkileri tanımlar. Bu ilişkiler, aynı özellikteki kayıt kümelerini aynı kategori altında toplayabileceğimiz bire-çok ilişki şeklindedir.Düz dosyaya her giriş, şemada gösterilen öğe adlarından her biri için en fazla bir değer içerir. Eğer herhangi bir veri öğesi veya veri öğelerinin bir grubu verilen bir giriş için birden fazla değer içereriyorsa, dosya hiyerarşik veri yapısına dönüşür. Hiyerarşik veri yapısının en göze çarpan özelliği, dosya kaydındaki verilerin kopyalanmasıdır. Daha ötesi, kopyalanan veri, içerilen kaydın manasının dışında hiçbir anlam ifade etmez. Hiyerarşik veri modeli, hiyerarşiler veya ağaçları kullanarak veriler arasındaki ilişkileri göstermektedir. Veriler arasındaki bütün ilişkiler, veri tabanı içerisinde saklanmadan önce hiyerarşilere dönüştürülmelidir. Hiyerarşik veri modeli üzerine kurulmuş en popüler ürün, IBM tarafından pazarlanan Data Language/I veya DL/I ' dir.

1.6.2 Hiyerarşik Veritabanı Modeli

Hiyerarşik Model ve Network Model öncelikli olarak Transaction processing'de kullanılır.

Günümüzde, Hiyerarşik ve Network modellerine göre yapısallaştırılmış veritabanları çok daha az yaygındır. Bununla birlikte halen, bu tür veritabanlarından binlerce vardır ve çoğu, geniş, organizasyonal veritabanı işlemede kullanılmaktadır. Bu kurulmuş tabandan dolayı Hiyerarşik ve Network modelleri çok önemlidir.

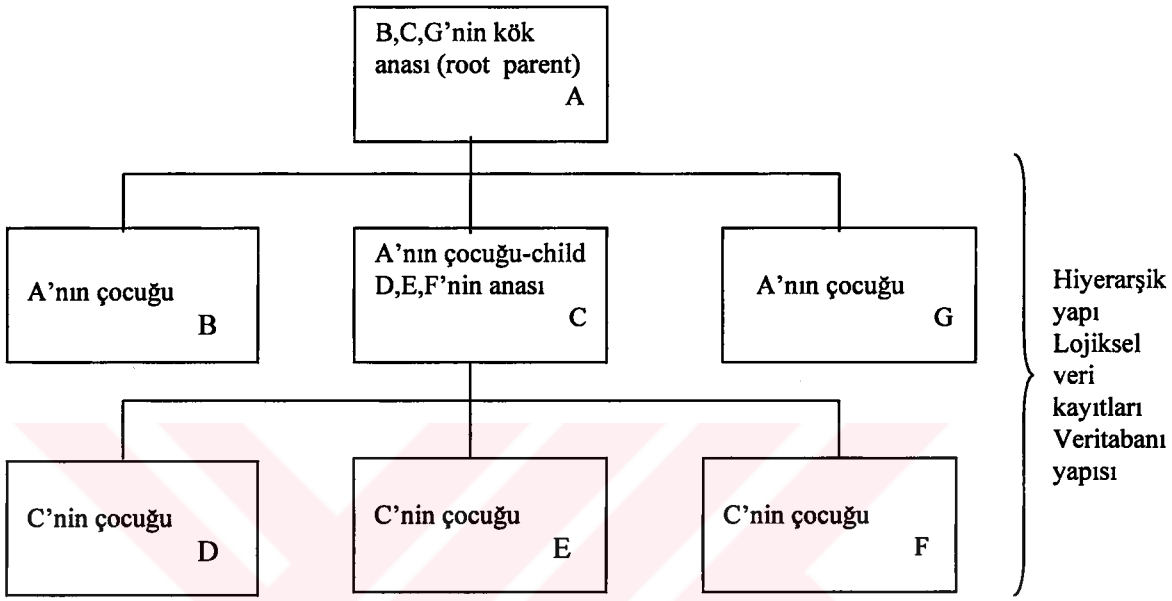
1.6.3 Hiyerarşik Modelin Veri Modeli Görünüşü

Aşağıda gösterilen hiyerarşik diagram (Şekil 1.6.3.1), ters çevrilmiş ağaç yapısı içerisindeki veri parçacıklarını(fragments) temsil eder. Bu ters çevrilmiş ağaç yapısı veri segmentasyonunu, segment bağlantılarını ve bu segmentler arasındaki içsel fonksiyonları gösterir. Her ağaç yapısı bir tip kayıtla ilgili veri birikimini gösterir ve lojiksel veri kaydı olarak da adlandırılır. Her veri tabanı başına sadece bir hiyerarşik yapı olabilir. Ağaç yapısının özel bir tanımlaması, çoklu ağaç yapılarının, mantıksal veri kaydı olarak da adlandırılan daha geniş ağaç yapılarıyla birleşmesini sağlar. Bununla birlikte, her ağaç bileşeni yine de ayrı bir veri tabanı olarak adlandırılır (Modell , 1995).

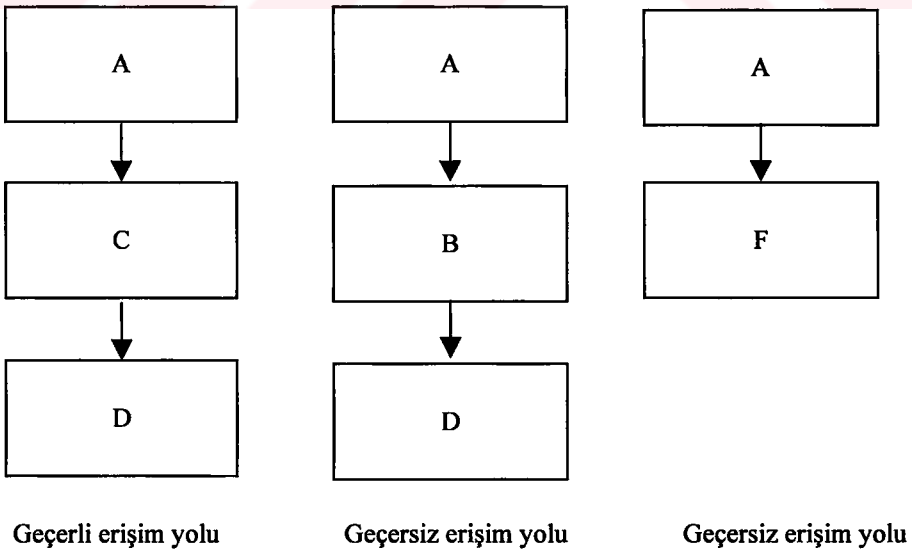
Lojiksel veri kaydına tüm erişimler veri tabanı boyunca veya kök (root) segment vasıtasıyla olmaktadır. Kayıt oluşumu için tanımlanan belirteçleri veya anahtarları içeren bu segmenttir. Bir veri tabanı, her biri belirli kayıt oluşumuna ait olan çoklu ağaç oluşumlarını içerir. Verilen bir veri tabanında herbir tek kayıtta, oluşumların kendi konfügürasyonları ya da genel kayıt hiyerarşisi için tanımlanan her segment tipinin olumsuzlukları vardır.

Ağaç yapısı diagramı her segment tipini sadece bir kez gösterir. Bununla birlikte, her yapının kök segmenti yanı sıra, yapıdaki herhangi verilmiş bağımlı segment tipinin çoklu oluşumları olabilir. Kök segment altındaki her veri segmenti taban kaydının bazı durumlarını tanımlar. Bu bağımlı segmentler, içeriklerine, kullanımlarına ve içeriklerinin oluşum sayılarına bağlı olarak anahtarlanmış veya anahtarlanmamış olabilir. Bağımlı segment anahtarları, oluşumların içinde ve karşısında tek(unique), çoğaltılmış(duplicate) veya her ikisi de olabilir.

Hiyerarşide, kök seviyesindeki segmentler sadece direk olarak kendilerine bağımlı olan segmentlerle ilişkilidir. Hiyeraşik yapıda tanımlanmış olduğu gibi, kök segment altındaki herhangi bir segmente erişim yolu, kök üzerindeki direk bir yol üzerindeki, onun tüm direk atalarını (predecessors) veya analarını (parents) içermelidir.



Şekil 1.6.3.1 Genel Hiyerarşik Model



Şekil 1.6.3.2 Geçerli-Geçersiz Erişim Yolu

1.6.4 Bir Hiyerarşide Görülen Özellikler

- Kökten aynı seviye aşağıda olan segmentlerin birbirleriyle ilişkisi yoktur.
- Çocuk(child) segmentler, hiyerarşi içinde bir sonraki daha yüksek seviyedeki sadece bir ana segmentle veya kökün kendisiyle ilişkilendirilmiştir.
- Her bir çocuk segment, hiyerarşide kendinden daha aşağıdaki herhangi sayıdaki çocuk segmente ana olarak bağlıdır.
- Bu tür seviye-seviye, ana-çocuk bağımlılıkları, düşük-seviye segmentlerin (çocukların) hiç bir anlamı olmadığını ve aslında, hiyerarşideki pozisyon bakımından daha yüksek seviyedeki segmentler (analar) olmaksızın var olamayacaklarını gösterir.

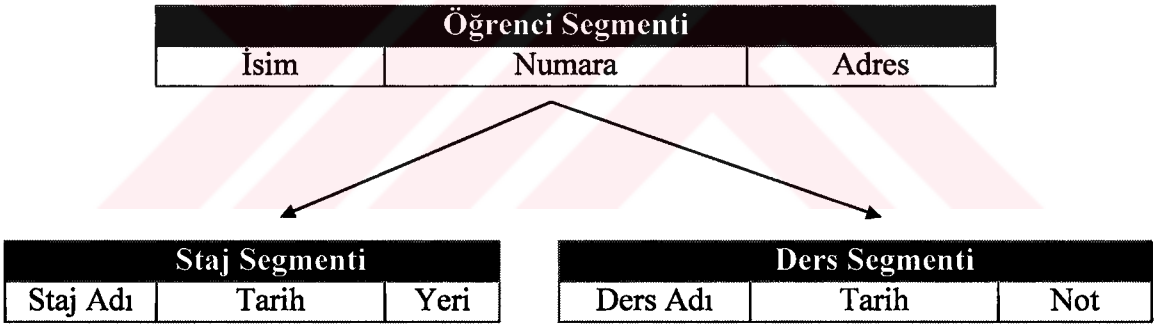
Hiyerarşik modelin en etkili olduğu durum, aşağıdaki özellikler sağlandığında olur:

- Her bir hiyerarşik yapı tek bir kayıtla ilgili veri içerirse, her bir kayıt relatif olarak homojense ve çok az farklı alttiplere sahipse.
- Her hiyerarşiye ilk erişim kaydın belirteci vasıtasıyla oluşuyorsa.
- Tanımlanmakta olan kayıt, tanımlayıcı alanlar bakımından zenginse ve bu alanların oluşumu birden çoksa veya hiç yoksa.
- Kayıt oluşumları aynı zamanda bir kez işlenmişse.

1.7 DATA LANGUAGE/I

Hiyerarşik veritabanı modelinin en önemli elemanı Data Language/I 'dir. DL/I veritabanını işlemek için kullanılan bir dildir. DL/I, ilişkileri temsil etmek için hiyeraraşileri(ağaçları) kullanır. Bunun anlamı, kullanıcı nesneleri DL/I tarafından temsil edilmeden önce ağaç gösterilimine çevrilmelidir.

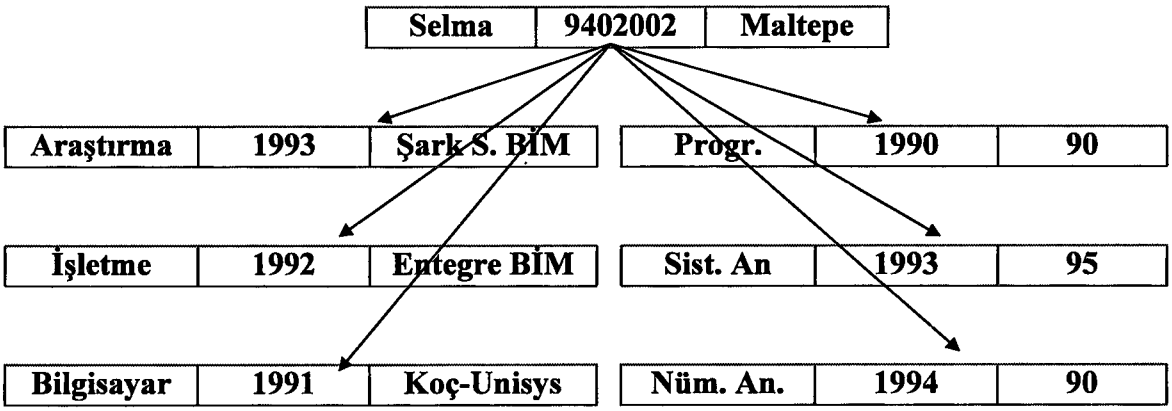
DL/I teriminde, alanlar **segmentler(kayıtlar)** içerisinde gruplandırılır ve segmentler ağaç yapısının düğümleridir. Bir **ağaç**, kayıtların ve ilişkilerinin toplamıdır, öyle ki her kayıt en fazla bir ana(parent)ya sahiptir ve anadan çocuğa tüm ilişkiler bire-çoktur. DL/I , bir veri tabanı* kaydı olarak, belirli ağaç yapısını (ilişkili dosyaların toplamını) temsil eder. Aşağıda bir ÖĞRENCİ veri tabanı kaydı gösterilmiştir.



Şekil 1.7.1 Öğrenci Segmenti

Her ÖĞRENCİ segmentinde İsim, Numara, Adres alanları vardır. Ayrıca, her ÖĞRENCİ segmentinin altında çeşitli sayıda STAJ ve DERS segmenti vardır.Böyle bir dosyanın görünümü Şekil 1.7.2.'de görüldüğü gibidir:

* DL/I ' da veri ve taban iki ayrı sözcük gibi kullanıldığı için ayrı yazılmaktadır



Şekil 1.7.2 Öğrenci Segmenti-Örnek

DL/I 'da, bir **veri tabanı,** veri tabanı kayıtlarının birleşimidir. Bu kayıtlar, aynı kayıt tipinin veya birkaç değişik kayıt tiplerinin olaylarıdır. Örneğin yukarıdaki örnekte bir veri tabanı, ÖĞRENCİ veri tabanının olaylarını içermektedir.

Veri tabanı kayıtları **veri tabanı betimlemesi** yoluyla tanımlanmaktadır. DBMS ürünü IMS'te, assembly-dili makro tanımlamalarının kümesi, her veri tabanı kaydının yapısını işaret eder (bu, ilişkisel dil SQL 'den farklıdır. SQL, hem veri tanımı hem de veri işleme cümleciklerini içermektedir.DL/I ise sadece veri işlemeyi içermektedir. Bu ise DL/I 'ın, veri tanımının öneminin farkedilmesinden önce ortaya çıktığını göstermektedir.)

Veri Yapısı	Tanımı
Alan	Verinin en küçük birimi
Segment	Alanların grubu;segmentler hiyerarşik yapıyla ilişkilendirilmelidir; her segment mantıksal sıralama için kullanılan ardışık bir alana sahiptir
Veri tabanı kaydı	Segmentlerin hiyerarşik olarak yapısallaşması
Veri tabanı	Bir veya daha fazla veri tabanı kayıt tiplerinin veri tiplerinin veri tabanı kayıt olaylarının toplamıdır

Tablo 1.7.1 DL/I veri yapılarının özeti

1.8 NETWORK(AĞ) VERİ MODELİ

DBMS ürünlerinin iki önemli sınıfı vardır. Bu ürünler organizasyonal operasyonları sağlayan hareket işleme sistemlerinde; nadir olarak da karar verme ve diğer bazı spesifik işlemlerde kullanılmakta olup, kişisel veritabanı uygulamalarında hiç kullanılmamaktadır.

Daha önceden de bahsedildiği gibi, bu sınıflardan birisi, hiyerarşik veritabanı modeli, veriyi hiyerarşiler veya ağaçlar şeklinde gösterir. İkinci sınıf ise network modeldir.

Network veritabanı, herhangi bir düğümün (node'un), diğer bir düğümü göstermesine ve böylece veriler arasında çoka-çok ilişkinin tanımlanmasına olanak sağlayarak, hiyerarşik veritabanında görülen birçok problemin üstesinden gelmektedir. Bu ise, hemen hemen tüm karmaşık ilişkilerin tanımlanmasını sağlar.

Hiyerarşik veritabanında kullanılan ana-çocuk (parent-child) kavramından çok, network veritabanı, bir veya daha çok kümeye ait olan veri öğeleri kümesi kavramını tanıtır. Bir veritabanı kurulduğunda, veritabanı boyunca herhangi bir aramada başlangıç noktası olarak kullanılan, birçok sayıda küme önceden tanımlanır (pre-defined). İstenilen veri öğesi bulunana kadar, çeşitli veri öğeleri arasındaki linkler kullanılır.

Network veritabanı, hiyerarşik veritabanındaki veri tekrarı (data duplication) problemini yok eder, fakat halen veritabanı yapısını değiştirmeyle ilgili problemlere sahiptir. Değişmesi gereken veritabanı kümeleri büyük ölçüde veritabanının yeniden-yapılandırılmasına (re-structuring) sebep olur. Bununla birlikte, yeni kümeler, yeni veri öğeleri yaratma ve onları önceden varolan verilerle link etme yoluyla, kolaylıkla ilave edilebilir.

Bire-çoklu ilişkileri temsil eden network modelin en iyi örneği, CODASYL DBTG modeldir.

1.9 İLİŞKİSEL (RELATIONAL) MODEL

İlişkisel model 1970 yılında E.F. Codd isimli bir bilim adamı tarafından, matematiğin bir dalı olan ilişkisel cebir kavramlarının, büyük miktarlardaki verilerin toplanması probleminin çözümüne uygulanması sonucu ortaya çıkmıştır. Bu model, veri tabanı işleme ve yapılandırmanın özel bir yoludur. 1970'lerde tanımlanmasına rağmen, 1980' lerin başlarında ilişkisel DBMS ürünleri sağlanana kadar pratikte çok az öneme sahipti.

İlişkisel veritabanı modeli, tüm veritabanı modelleri arasında en basit ve en muntazam yapıda olanıdır. Ayrıca ticari olarak en çok kullanılanıdır.

1.9.1 İlişkisel Modelin Yararları

İlişkisel modelin avantajı, en azından kavramsal olarak, verilerin kullanıcıların hemen anlayabileceği şekilde saklanmasıdır. Veri, tablolar halinde saklanır ve tablonun satırları arasındaki ilişkiler veride görülebilir.

Bu veritabanını, sadece veriyi değil, veriler arasındaki ilişkileri de saklıyor şeklinde adlandırabiliriz.

Örneğin bir öğrenci transcriptinin oluşturulmasını gözönüne alalım. Aşağıdaki tabloda (Tablo 1.9.1.1), bir öğrenci transcripti için gerekli temel verileri göstermektedir: Öğrencilerle ilgili veriler ve öğrencilerin tamamlamış oldukları derslerle ilgili veriler bulunmaktadır. Fakat transcripti oluşturmak için sadece verileri değil, veri değerleri arasındaki ilişkileri de bilmek zorundayız, bu durumda ilişki, belirli öğrenciler ve dersler arasındaki ilişkilerdir.

DBMS ürünleri böylesi veri ilişkilerini gösterme şekillerine göre değişiklik gösterirler. Önceleri DBMS ürünleri ilişkileri, tepeüstü (overhead) veri yapıları hakkında bilgisi olmayanlardan saklayan üstüste yığılmış verinin içinde saklardı. Programcıların ve

diğer MIS(Managing Informatıon Systems) profesyonellerinin bu yapıyı ve veritabanını yönlendirmek için onları nasıl kullanacaklarını öğrenmelerine rağmen , tipik bir son-kullanıcı onların ne olduğunu bilmiyordu.

Öğrenci No	Ad
9402002	Selma
9402003	Elif
9402004	Tülay
9502004	Jale

Öğrenci Verileri

Ders Kodu	Ders Adı	Vize	Final	Öğrenci No
32	İleri programlama	85	95	9402002
33	Sistem Planlama	85	90	9402003
32	İleri Programlama	85	85	9502004
34	Kodlama Teorisi	80	85	9402004
35	Çok Amaçlı Prog.	90	90	9402002
36	Kantitatif K.V.T	90	85	9402004
37	Veri Yapıları	90	95	9402002
33	Sistem Planlama	85	95	9402002
37	Veri Yapıları	90	90	9502004
38	Çok Boyutlu İst.	80	95	9402003

Ders Verileri

İlişkileri göstermek için değerler seçme

Tablo 1.9.1.1 Öğrenci-Ders Verileri

Yukardaki sebeplerden dolayı, kullanıcılar, program yazmada gerekli bilgileri almak için bilgi sistemleri profesyonellerine bağlıydılar. Bu, çoğunlukla fazla vakit almaktaydı ve oldukça bıkkınlık vermekteydi.

İlişkisel model, bu durumu değiştirdi. İlişkisel modelin en önemli özelliği, ilişkileri kullanıcı-görebilir veri şeklinde saklamasıdır. Yukardaki tabloda (Tablo 1.9.1.1.), öğrenciler ve dersler arasındaki ilişkiler, **Ders Verileri** kaydında bulunan **Öğrenci Numarası** alanında saklanmaktadır. Bir ders kaydını yeniden elde edebiliriz ve bileşenlerini incelemek suretiyle hangi öğrencinin bu dersi aldığını belirleyebiliriz. Ayrıca veriyi tam aksi şekilde de işleyebiliriz; bir öğrencinin numarasını verdiğimiz takdirde, ilişkisel bir DBMS bu öğrencinin hangi dersleri aldığını belirleyebilir.

İlişkisel modelde, kullanıcının işlem yapmak için sadece nasıl bir kayıt istediğini belirlemesi gereklidir.

1.9.2 Kayıtlar ve Özellikeri

İlişkisel tasarım gerekli kayıtları tanımlamakla başlar. Bir kayıt(entity) oldukça basit şekilde , bir şahıs, yer , olay veya verilerini saklamak istediğimiz herhangi birşey olabilir. Bir üniversite ortamında ilgilenilen kayıtlar, yukardaki örnekte de gösterildiği gibi öğrenciler, dersler olabileceği gibi fakülte elemanları, kurslar vb. olabilir.

Her kayıt, özellik, alan(attribute, field) olarak bilinen belirli karakteristiklere sahiptir. Örneğin bir öğrenci kaydı, öğrencinin adı soyadı, numarası, bitirme notu, ortalaması gibi özelliklere sahiptir. Her özellik bakıldığında neyi anlatmak istediği anlaşılacak şekilde adlandırılmalıdır. Örneğin Öğrenci Numarası, OGRENCI-NO şeklinde adlandırılabilir.

İlişkilendirilmiş kayıtların gruplandırılması kayıt kümesini oluşturur. Kayıt kümesinin ismi, temsil ettiği kayıtları çağrıştıracak şekilde verilmelidir. Örneğin öğrenci kayıtlarının bulunduğu entity kümesi OGRENCI şeklinde adlandırılabilir.

1.9.3 Tablolar ve Karakteristikleri

İlişkisel veritabanının mantıksal görünüşü, *tablo* olarak bilinen bir yapının üzerende veriler arasında ilişkiler yaratılmasıyla faydalı hale getirilmiştir. Bir tablo ilişkilendirilmiş kayıtlardan oluşur. Bir tablo ayrıca *ilişki* olarak da adlandırılır. Çünkü ilişkisel modelin babası sayılan E.F. Codd *tablo* ile eş anlamlı olarak *ilişki* ' yi kullanmıştır.

Veritabanı kavramı çok açıktır. Fakat , dosya sistem terminolojisi zaman zaman veritabanı çevresinde görünmeye başlamıştır. Bunun için tablonun satırları(row), *kayıt*

olarak, sütunları(column), *field* olarak ve tablonun kendisi de *dosya* olarak adlandırılmaktadır. Teknik olarak belirtmek gerekirse, terimlerin birbirlerinin yerini alması her zaman uygun değildir. Veritabanı fiziksel bir kavramdan daha çok mantıksal bir kavramdır ve dosya, kayıt, alan terimleri fiziklel kavramları tanımlar.

Bir tablo birçok satır ve sütun içerebilir. Tablonun büyüklüğü kullanılan ilişkisel yazılımla sınırlıdır. Örneğin, bazı yazılımlarda satır sınırlaması olmamamasına rağmen bazı yazılımlar satırların sayısını iki milyarla sınırlamaktadır. Bununla birlikte hiçbir durumda da satırların kapasitesi dolmamaktadır.

1.9.4 Veri Tipleri

Nümerik(numeric-sayısal): Aritmetik işlemleri anlamlı bir şekilde yapabilmek için kullanılan veri tipidir. Nümerik veri tipine sahip veriler nümerik alan adını alırlar.

Karakter(character): Karakter veri tipi, matematiksel işlemlerde kullanılmayacak olan herhangi sembol veya karakterlerden oluşur. Bu veri tipine string veri tipi de denilir.

Tarih(date): Tarih alanı, *Julian tarih formatı* olarak bilinen takvim tarihlerini içerir. Julian tarih formatı, *Julian tarih aritmetiği* olarak bilinen özel aritmetik işlemlerin yapılabilmesini sağlar. Örneğin 01/06/1973 ile 01/06/1997 tarihleri arasında kaç gün olduğunu bir tarihten diğerini çıkararak bulabiliriz. Burada tarih formatı Amerikan tarih formatına göredir, yani aa/gg/yy (mm/dd/yy) olarak alınmaktadır. İlişkisel veritabanlarının çoğu, ancak hepsi değil Julian tarih formatını desteklemektedir.

Lojik(logical): Lojik veri tipi sadece Doğru/Yanlış (True/False) değeri döndürür. İlişkisel veritabanı yazılım paketlerinin hepsi değil ama çoğu lojik veri formatını destekler.

1.9.5 İlişkisel Veritabanı Anahtarları

Süperanahtar(Superkey): Bir tablodaki her kaydı tek(unique) olarak tanımlayan bir alandan veya birkaç alanın toplamından oluşabilir.

Aday anahtar(Candidate Key): Minimal bir süperanahtardır. Kendisi süperanahtar olan alanların alt kümesini içermeyen bir süperanahtardır.

Birincil anahtar(Primary Key): Herhangi bir verilen satırda(kayıtta) tüm diğer alanların değerlerini unique(tek) olarak tanımlamak için seçilmiş bir aday anahtardır. Boş girişleri kabul edemez.

İkincil Anahtar(Secondary Key) : Bir alan(veya alanların toplamı) veri tekrarı gibi amaçlar için kullanılır.

Yabancı Anahtar(Foreign Key): Bir tabloda, başka bir tablodaki birincil anahtarın değerini gösteren veya değeri sıfır olan bir alan veya alanların toplamıdır.

1.9.6 Veri Modelini İlişkisel Veri Yapısına Dönüştürme

İlişkisel modelde, sadece bazı karakteristiklere sahip, bazılarına sahip olmayan kayıtlar ilişkisel tablolara dönüşebilir. Kayıtların alanları, her alanın oluşum sayısı, kayıtlar arasındaki ilişkinin kompleksliği ve her kayıt ailesinin içinde dağılacağı grupların sayısı bu dönüşümü etkileyen faktörlerdir.

Her farklı kayıt ailesi veya grubu ilişkisel bir tablo olur. Tablonun kolonları(attributes) , kaydın belirteçlerini ve tablonun tekrar etmeyen özelliklerinin veri elemanlarını içerir. Kayıt özelliklerinin birden fazla oluştuğu durumlarda, her farklı alan tipi, tekrar eden olaylar arasındaki farklılığı sezebilmeyi gerektiren herhangi karakteristik veya veri elemanlarına ilaveten kayıt ailesinin belirteçlerinden oluşmuş bir anahtara sahip olarak ayrı bir tablo kurulmalıdır.

Network ve Hiyerarşik modelde, basit ve karmaşık ilişkiler değişik şekillerde ele alınır. Eğer ilişki basitse, yani eğer hiçbiri tekrarlanmayan az sayıda alana sahipse bunun için tek bir tablonun yaratılması yeterlidir. Bu tablonun her satırı iki kayıt arasında belirli bir ilişki oluşumunu anlatır. Tablo, her kolonun ilişkili kayıtların eşleştirilmiş(paired) anahtarlarından birini içeren bir çift sütun içermelidir. Bu eşleştirilmiş anahtarlar, ilişkilerin oluşumunu belirleyecek şekilde hareket ederler. Tablodaki kolonların geri kalanları, ilişkileri sınıflandıran ve tanımlayan alanların veri elemanlarını içerir. Her kayıt çifti arasındaki, her ayrı ilişki tipi ayrı bir tabloya dönüştürülmelidir.Karmaşık ilişkileri, çoklu veya çok olabilirli alanları içeren kayıtlara, tekrar eden alanlı kayıtlarmış gibi davranılmalıdır. Bu ilişki kaydının taban tablosu, tüm tekrar etmeyen alanların veri elemanlarını içermelidir ve ilişkinin tekrar eden her alan tipi için ayrı bir tablo kurulmalıdır. İlişki kümesinin her tablosu, herbiri ilişkili kayıtların eşleştirilmiş anahtarlarından birini içeren sütunların bir kümesini içermelidir.

Eğer seçilecek kayıt aşağıdaki özelliklere sahipse, ilişkisel uygulaması uygundur:

- Kayıt, tekrar etme olasılığı birden fazla olan az sayıda alana sahiptir.
- Kayda erişim, kaydın birincil anahtarı ile veya kaydın elmanlarından kurulmuş tabloda tanımlı veri elemanlarıyla olmaktadır.
- Kayıtlar arasında birden fazla ilişki vardır ve ilişkiler iyi tanımlanmış, veya onları tanımlayacak ve niteleyecek alanları gerektirecek şekilde kötü ve basit tanımlanmış olabilir.
- Kayıtlar arasındaki ilişkiler her kayıtta yerleşmiş veri elemanlarıyla sürdürülebilir.
- Kaydın işlenmesi genellikle tek olarak veya diğer kayıt tiplerinin küçük gruplarının kombinasyonu şeklinde oluşur. Yani, referans için bir uygulama tarafından veya herhangi bir zamanda kayıtları işleme amaçları için sadece az sayıda kayda (kayıtların oluşlarına değil) ihtiyaç vardır.
- Kayıt işleme için kayıtların oluş sırası önemsizdir.

- Kayıt işleme herhangi bir zamanda kurulabilir.
- Kayıt işleme, aynı zamanda elde edilebilir olan kaydın tüm elemanlarını gerektirmez.
- Kayıtlarla ilgili veriler durağandır ve kayıt işlemenin bir sonucu olarak değişmez.
- Kayıtlar , birçok ortak alanı paylaşmayan birden fazla gruba bölünmüştür.
- Herbiri değişen alan karakteristiklerine sahip çeşitli kayıt grupları vardır ve uygulama işleme, kayıt kümesinin bir bütün olarak işleneceğini vurgulamaz.
- Bir kayıtla ilgili veriler bir veya iki tabloda içerilebilir ve verilen bir kaydın kayıt veri alanlarının güncellenmesi herhangi bir diğer tablodaki veriye bağlı değildir.
- Bir tablodaki veriyi güncelleme, herhangi bir diğer tablolayla olan ilişkisini etkilemez.

1.9.7 Kayıt bütünlüğü, referans bütünlüğü kısıtları

Referans bütünlüğü kısıtları, yabancı anahtarla tanımlanan herhangi kaydın referans edilen tabloda da bulunması gerektiği anlamına gelir.

Kayıt bütünlüğü kısıtları, hiç bir birincil anahtarın değerinin boş(null) olamayacağını ifade eder. Bu, birincil anahtar değerinin bir ilişkide tek kayıtlar tanımladığından dolayıdır. Birincil anahtar için boş değerlerin olması, bizim bazı kayıtları tanımlayamayacağımız anlamına gelir. Örneğin, iki veya daha fazla kaydın birincil anahtarı boş ise biz bu iki kaydı ayırdedemeyiz.

Kayıt kısıtları ve kayıt bütünlüğü kısıtları tek ilişki üzerinde tanımlanır. Referans bütünlüğü kısıtı iki ilişki arasında tanımlanır ve ilişkinin kayıtları arasındaki tutarlılığı sürdürmek için kullanılır.

Çoklu ilişkilerin olduğu bir veritabanında çok fazla referans bütünlüğü kısıtları vardır. Referans bütünlüğü kısıtı, tipik olarak ilişki şemalarındaki kayıtlar arasındaki ilişkiden doğar.

1.10 NESNE YÖNELİMLİ (OBJECT ORIENTED) VERİTABANI MODELİ

İlişkisel Model son 10-15 sene boyunca veritabanları dünyasının hükümdarıydı. Bu süreç zarfında geliştirilen ve bir veritabanı gerektiren projelerden çok azı ilişkisel sistemlerden daha farklı bir model kullanmıştır. Nesne Yönelimli Veritabanı Yönetim Sistemi (OODBMS-Object Oriented Database Management System) ürünleri, karmaşık verilerin kullanılmasında, İlişkisel Modele dayalı ürünlerin başaramadıkları alanlarda gösterdikleri yetenekleriyle daha geniş bir kitle tarafından benimsenmişlerdir.

Nesne Yönelimli Model, işlemsel ve kural-tabanlı programlamadan sonra bilgisayar programcılığı için en son modeli temsil etmektedir. Nesnetabanlı diller, karmaşık veri yapılarını ve bu verileri kullanmak için metodları biraraya getiren nesneleri kullanırlar. Bir metodu çalıştırabilmek için bu metodun bulunduğu nesneye bir mesaj gönderilir. Her nesne kendi metodlarını içerdiği için çoğu prosedürel kod elenebilir. Nesne Yönelimli Model şu sıralar gündemde bir konu olmasına rağmen aslında pek de yeni bir model değildir. İlk nesne yönelimli programlama dili (OOPL-Object Oriented Programme Language) olan Simula-67 1967 senesinde ortaya çıkmıştır.

1.10.1 Nesne Yönelimli Veritabanı Yönetim Sistemleri

OOPL ' lerin oluşturdukları ve üzerinde işlem yaptıkları verileri kontrol edebilmek ve nesne yönelimli uygulamaların DBMS'lerin tüm yararlarından faydalanabilmek için OODBMS' ler öne sürülmüştür. OODBMS'lerin tipik avantajları verilerin sürekliliği, veri paylaşımı, paralel veri erişimi olarak sayılabilir.

OODBMS ürünleri nesne yönelimli bir geliştirme ortamındaki DBMS ürünleri yerine kullanılmak üzere tasarlanırlar. Bu ürünlerin mimarisi, karmaşık nesneler, soyut veri

tipleri ve "kalıcı" gibi nesne yönelimli teknikleri algılayıp kullanabilecek şekilde dizayn edilmektedir.

OODBMS hareketlerinin bütünlüğü esastır. Burada hareketin anlamı komutlar zincirindeki hiçbir komutun atlanmadan uygulanmasıdır; yani hareketi oluşturan işlemlerin ya hepsi tamamlanır ya da hiçbiri. Bir OODBMS hareketi, kayıt aracına bir nesne işlendiği ve bu kaydın onaylandığı anlamına gelir.

II.VERİTABANI DÖNÜŞTÜRME (CONVERSION)

Veri modeli tasarımcılarının karşılaştığı en büyük problemlerden birisi, var olan bir veritabanı ürününün yapısının değiştirilmesi gerektiğinde ne yapılması gerektiğidir. Veritabanındaki bir bölümün değişikliği, o bölümü kullanan her yerde de değişiklik yapılmasını gerekli kılmaktadır. Bu değişikliğin derecesi ufaktan büyüğe doğru değişir. Genelde, yeni bir veritabanı, var olan eski dosya (file)- tabanlı uygulamanın (genellikle COBOL etrafında dönen) veya diğer veritabanının (organizasyonun işlem ihtiyaçlarını karşılamayacağı ispatlanmış veritabanının) yerini alacak şekilde tasarlanıp, gerçekleştirilir . Bir DBMS'i diğerine dönüştürürken kullanıcılar ve yöneticiler görevlerini çok çabuk bir şekilde başarıyla tamamlamak için aşırı bir endişe duyarlar. Çünkü görülen değişiklik sadece basit bir upgrade (seviye yükseltme)'dir, bunun için eski sistemi, yeni sistemin üzerine mümkün olduğu kadar çok az bir tasarım eforuyla eşleştirme (map etme) eğilimi vardır.

Var olan veritabanı işlemcisini, veritabanı yönetimine doğrudan eşleştirme kötü bir uygulamadır. Nadir olarak eldeki DBMS tool' unun (aletinin) avantajlarını kullanan bir uygulamayla sonuçlanır. Sonuç genellikle orjinal sistemden daha iyidir. Bir DBMS'ten diğerine dönüşüm(conversion) yapılırken, sadece iki DBMS aynı ise ve orjinal veritabanı iyi tasarımlanmışsa eşleştirme yönteminin uygulaması tavsiye edilmektedir. Eğer DBMS'ler farklıysa, eşleştirme işlemi yeni sistemin güçlü olmasını kolaylaştırmayacaktır. Dahası, eski sistemin güçlü olduğu yönleri yeni sistemde zayıf olabilecektir bu yüzden sonuçtaki uygulama eski sistem kadar iyi icra edilemeyecektir. Bu, özellikle, eski ve yeni sistemler birbirlerinden önemli ölçüde farklıysa doğru olacaktır. Genelde, doğrudan bir veritabanı dönüşümü iyi bir uygulamayla sonuçlanmaz. Doğrudan dönüştürme, bir tasarım

metodu değildir, fakat bir tasarımdan kaçma metodudur. Mümkün olduğu kadar bundan sakınılmalı , kullanılması kaçınılmaz olan durumlarda ise büyük dikkatle kullanılmalıdır.

2.1 VERİ DÖNÜŞTÜRME İŞLEMİ

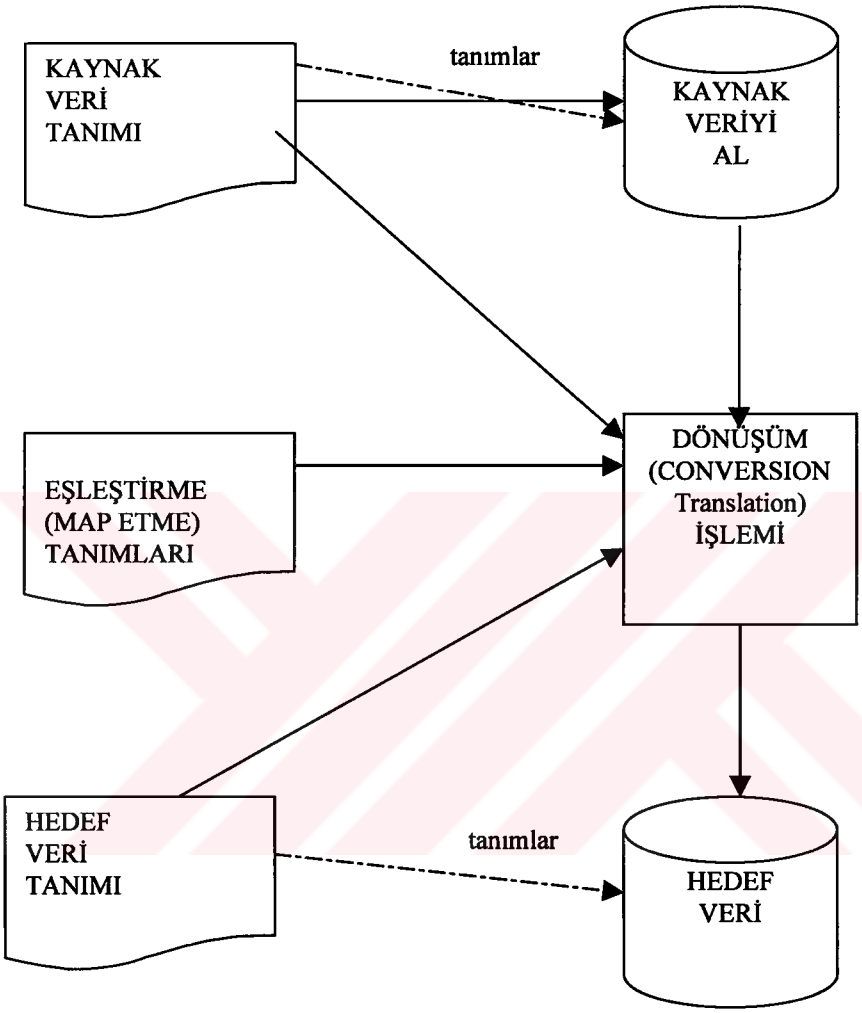
Bir veri dönüştürme işlemi verilerin makinedeki okunabilir halini alır ve diğerine dönüştürür. Veri dönüştürme, veritabanı yaratma, günleme, şema-kullanıcı şeması eşleştirme işlemlerinde kullanılır.

Input(giriş) olarak veri dönüştürme süreci

- Var olan mekanize edilmiş veri (bir veri tabanında external bir dosya veya program buffer'ı) , dönüşüm işlemi için *kaynak(source)* veri olarak adlandırılır.
- Verinin daha açık ve ayırt edilebilir tam tanımı, dönüşüm programının içine gömülebilir, veya var olan verinin anlaşılır şekil ve yapıda olmasını belirleyen genel dönüşüm programında olduğu varsayılabilir.
- Yeni *hedef(target)* verinin tanımı dönüşüm işlemi tarafından üretilecektir. Hedef veri tanımı tam olabilir veya kaynak veri tanımına bağlı olarak değişebilir.

Output(çıkış) olarak veri dönüştürme süreci

- Bir veritabanında veya external dosyada, veya program buffer'ında hedef veri tanımına dönüşmüş verinin yeni bir birikimi.



Şekil 2.1.1 Genel Veri Dönüşümü

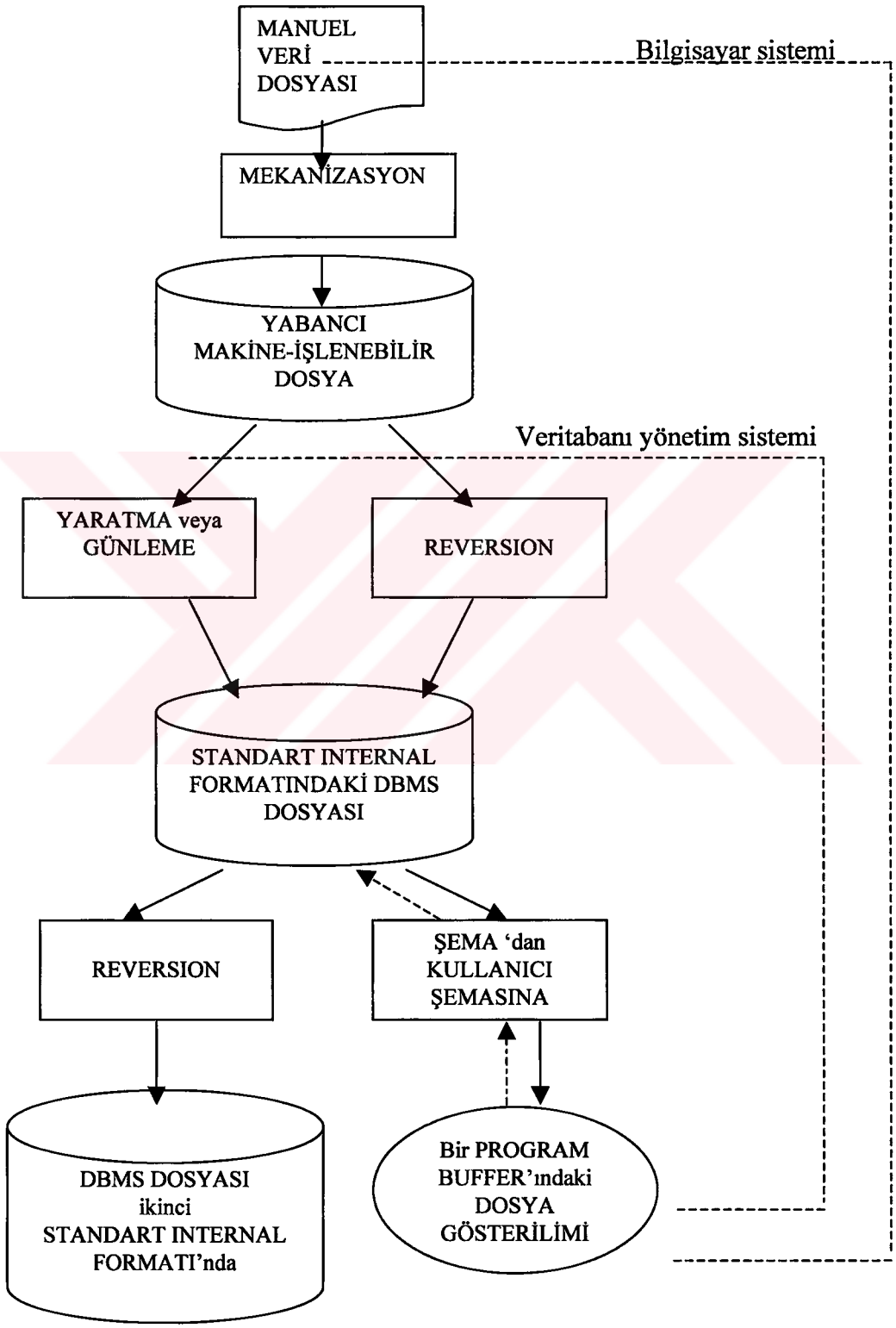
2.2 VERİ DÖNÜŞÜM İŞLEMİNİN AİLESİ

Veri dönüşüm işlemlerinin ailesi mekanizasyon, yaratma, günleme, sorgulama, reversion(dosyaya yazma veya dosya generate etme olarak da adlandırılır) ve şema-kullanıcı şema dönüşümünü içerir. Şekil 2.2.1'de daha açık olması bakımından, her dönüşüm sürecinde gerekli olan hedef ve kaynak veri tanımları gösterilmemiştir.

Bir DBMS'de veri tabanı dönüştürme işleminin veritabanı kontrol sisteminde önemi vurgulanan modüllerle yapılması tercih edilir. Dönüşüm işlemi ailesinin her elemanında zengin veri dönüştürme kolaylıklarının bulunması, sistemde sergilenen tüm verilerin bağımsızlık derecesini belirleyen zengin veri dönüşüm kolaylıkları, kaynak ve hedef veri arasındaki eşleştirmenin çeşitli seviyelerini ölçecektir.

Veri bağımsızlığının derecesini tahmin ederken, ilk adım, muhtemelen kaynak ve hedef arasında değişen veri karakteristiklerini tanımlamaktır. Sadece aradaki farklılıklar tanımlanmakla kalmamalı, ayrıca hedef , ile kaynak arasındaki veri transfer etme dönüşüm işlemi tarafından işlenmelidir. Böyle bir durumun gözardı edilmesi durumunda tüm değişiklerin işlenebileceğinin düşünülmemesi gerekmektedir. Genelde, kaynak tanımın, hedef tanımının günleyemeyeceği belirli yapısal karakteristikleri vardır.

Bazı veri dönüştürme işlemleri veri azaltıcı(reducing) iken diğerleri veri koruyucudur (data preserving). Bir veri koruyucu dönüşüm, hedef veriyi üretirken kaynak verideki tüm bilgiyi elinde tutar. Veritabanı yaratma, günleme ve yazma, genelde veri koruyucu dönüşümlerdir. Bir veri azaltıcı dönüşüm hedef veriyi üretirken kaynaktaki verinin sadece bir kısmını kullanır. Şema-kullanıcı şeması ve reversion (bir iç kayıda erişme) genelde veri azaltıcı dönüşümlerdir.



Şekil 2.2.1 Veri Dönüşüm İşleminin Ailesi

2.3 VERİ DÖNÜŞTÜRMEDE EŞLEŞTİRME SEVİYELERİ

Hedef ve kaynak veri tanımları arasındaki farklılıklar dört farklı seviyede ortaya çıkmaktadır:

- Tek veri öğelerinde
- Kayıt-içi'nde, yani, kayıt veya öğelerin gruplarında
- Kayıtlar-arası(inter-record) ilişkilerde
- Dosyalarda

Bazı durumlarda, sistem, sadece hedef ve kaynak veri tanımlarını bilmek suretiyle gerekli olan veri dönüşümünü gerçekleştirebilir. Diğer durumlarda, sistemin gerekli olan veri dönüşümünü gerçekleştirebilmesi için eşleştirme özelliklerinin anlaşılır olması gerekmektedir.

2.3.1 Veri öğesi eşleştirme

Öğe seviyesi eşleşleştirmeler, kaynak veri tanımındaki her veri öğesini, hedef veri tanımındaki bir veri öğesiyle ilişkilendirir.

- **İsim(Name):**Hedef veri tanımındaki bir veri madddesinin yeniden isimlendirilmesi, onu kaynak veri tanımındaki bir öğeyle ilişkilendirmek için anlaşılır bir eşleştirme gerekmektedir.
- **Tip(Type) :**Bazı tipleri dönüştürme hiçbir anlam ifade etmez, ve bunlar arasındaki dönüşüm genelde iki veri tanımından çıkarılabilir.Çoğu sistemler hangi dönüşümlerin yapılabilir olduğunu ve nasıl icra edileceğini gösteren tipten-tipe dönüşüm tablosu kurarlar. Bir fiziksel gösterilimden diğerine nümerik veri tipleri arasındaki dönüşümler

genelde, tamdeğere dönüştürme veya ondalıktan tamsayıya dönüştürme adımlarına izin vererek doğrudan yapılır.

- **Boyut(Size):**Tamdeğere dönüştürme boyutun kısalmasına, doldurma (soldan sağa doğru veri ayarlamasından dolayı) boyutun genişlemesine neden olacağından, bir veri öğesi herhangi belirli bir eşleştirme açıklamasına gerek kalmadan sıklıkla değişebilir. Açık bir eşleştirme direktifleri yerleşik değerlerin (default değerler) eskisini geçersiz kılmak için kullanılabilir. Eğer veri öğesi değerleri daima değişken uzunluğunda saklanırsa, hedef ve kaynaktaki alan boyutları arasındaki farklılıklar, çok az zorluk çıkaracaktır.
- **Ölçü(Scale):**Ölçü birmini veya ölçütü değiştirme, dönüşüm işleminin ilişkileri bilmemesi halinde açık bir eşleştirme tanımı gerektirmektedir. Dönüşüm işlemi için belirli genel veri öğeleri arasındaki ilişkilerin önceden belirlenmesi gerekmektedir, örneğin Fahrenheit ve Celcius arsındaki ilişki gibi; doğru, alan ve hacmin çeşitli gösterilimleri arasındaki ilişkiler veya ağırlık ölçüleri arasındaki ve SI ölçü birimiyle değişik İngiliz ve Amerikan birimleri arasındaki ilişkiler gibi.
- **Değer(Value):**Yukardaki dönüşümlerin çoğunda değerlerin gösterilimindeki değişiklikten bahsedildi, fakat ek olarak, içinde kaynak veride var olan her değer için yeni bir değerin listelendiği, açık ve anlaşılır bir kodlama-kod çözme tablosunda belirlenmelidir. Değer dönüşümleri çoktan-bire(N:1) olabilir, örneğin maaş oranlarının gerçek değerlerin yerini alacak şekilde ayarlanması gibi.

Bir şema-kullanıcı şeması eşleştirmede sadece bir veri öğesini değiştirme programda hiçbir değişiklik gerektirmeyecektir. Eşleştirme tanımına işlenemeyen diğer değişiklikler programda değişiklik gerektirecektir.

Çok az sayıdaki sistem, öğe-seviye bağımsızlığının bir ölçüsünü sunar. Veri öğe değerlerinin transferindeki tekrarlanan dönüşüm, işletim performansı bakımından pahalı olmasına rağmen; gelecekte, var olan gruplardaki öğelerin veritabanında tekrar

tanımlanması gerektiğinde, öğe-seviye bağımsızlığının az olması çok daha pahalıya mal olacaktır. Yeniden programlamanın sonuçtaki tutarı çok fazla olabilecektir.

2.3.2 Kayıt-İçi(Intra-Record) Eşleştirme

Kayıt-seviye eşleştirmeleri kayıtların veya grupların, içerikleriyle ve tekrar eden grup veya kayıt tiplerinin her örneğindeki veri öğeleri arasındaki ilişkilerle ilişkilidir.Veri öğeleri yeniden düzenlenebilir, eklenebilir veya silinebilir. Yeniden düzenleme veri öğelerinin açık şekilde ardışık olarak yeniden adlandırılmasını gerektirir.Silme, veri öğesinin isminin atlanması durumunda örtülü veya eşleştirme tanımında açık olabilir.Ekleme yapma, bir türetme kuralının belirlenmesini gerektirmektedir. Kaydın fiziksel boyutunu değiştiren bir dönüşüm açık olabilir veya içerilen veri öğelerindeki boyut değişikliklerinden türetilebilir.

Aynı şekilde, veritabanındaki bir öğeyi silme, öğeye bağlı olan her programda istenmeyen sonuçlara sebep olacaktır.Böylesi programlar silinen öğeyle ilgili tüm referansları kaldırmak için yeniden yazılmak zorunda kalacaktır.

Genel olarak, kayıt-seviye eşleştirmelerini icra ederken, sistemler, kaynak ve hedef tanımındaki veri değerlerinin isimlerinin aynı olmasını gerektirmektedir. Bu yolla dönüşümün başarılı olması için açıklayıcı eşleştirme cümleciklerine gerek kalmamaktadır. Farklı şekilde bir yeniden tanımlama(redefinition) yoluyla veri öğelerinin isimlerinin değiştirilmesi mümkündür.

Bugün çoğu sistem, seviye öğe bağımsızlığına ek olarak, şema ve kullanıcı şeması arasındaki grupiçi-seviye(intragroup-level) bağımsızlığının herhangi önemli bir seviyesini sunmamaktadır. Bu, sistemlerin, sadece anahtar veri öğesinin haricinde, içeriklerin tanımının ve grupların formatının farkında olmaması gerçeğinden dolayıdır. Tüm bu koşullar altında bir program, gruptaki veri öğelerini tam olarak veritabanında saklandığı şekilde gözlemlemelidir. Dahası, eğer sistem bir grup tanımının farkında değilse, grupların

genelde anlaşılan tanımlarının yapılması sorumluluğu yalnızca programcı ve kullanıcıya aittir. Bugün çoğu kuruluşlarda, bu, veritabanı birleştirmeyi ciddi şekilde etkileyen müsamaha edilemez bir durumdur. Gruplar içi veya seviye bağımsızlığı olmadan , var olan bir gruptaki herhangi bir değişiklik bu gruba bağlı tüm programların yeniden derlenmesine ve uzatmalı programlamaya sebep olacaktır.

Eğer bir programcının kullanıcı şeması. veritabanında saklanan bazı veri maddelerini gruptan gözardı edebilirse, veritabanı yöneticisi için, var olan herhangi bir programı etkilemeden veri öğelerini var olan gruplara veya kayıtlara ekleme mümkündür. Veritabanında açıkça görülmemesine rağmen, yeni veri öğeleri kullanıcı şemasında içerilebilir. Bu tür öğeler sanal öğe olarak adlandırılırlar ve veritabanında var olan veri öğe değerlerine bir tanım kuralı uygulanarak türetilirler.

Yukarda tartışılan benzer grup tiplerinin iki örneğini bire-bir ilişkilendiren grup-seviye eşleştirmelerine ek olarak, diğer grup-seviye eşleştirmeleri birden fazla tekrar eden grup ve kayıt tiplerini içerebilirler. Diğer bir örnek, bir veri öğesi, birkaç örnek boyunca veri maddelerinin değerlerinin toplamından veya diğer bir gruptan, belki ikincil bir tekrar eden gruptan türetilebilir. Açık olarak görülüyor ki, bu tür dönüşümlerin gerçekleştirilmesi çok daha zordur.

Bir program, hepsine değil ancak sadece gruptaki bazı maddelere bağlıysa ve bu program veritabanına bu grubun yeni örneklerini eklerse ilginç bir durum ortaya çıkar.Kayıp veri öğe değeri sistem tarafından açık bir şekilde kaydedilmelidir. Sadece bu yolla, bir alt program aynı veri öğelerinin değerlerini çağırdığı zaman veritabanı birleştirme sürdürülebilir. Dahası, sistem bağlantılı olmayan veri öğesinin zorunlu öğe, yani her grup örneğinde bir değeri olması gereken öğe, olup olmadığını kontrol etmek zorundadır. Bu durumda, bir program, grubun sadece bir bölümüne bağlı olan, var olan bir grup örneğini okuduğunda, grubu günlediğinde, ve onu yeniden veritabanına yazdığında daha da karmaşık olmaktadır. Bu işlem sırasında sistem, bağlantılı olmayan veri öğelerinin değerlerini korumak zorundadır.

2.3.3 Kayıtlar-Arası (Inter-Record) İlişki Eşleştirme

Kayıtlar-arası seviye eşleştirmeler, kayıt tiplerini gözardı etmek, kayıtlar-arası ilişkileri ters çevirmek, kayıt tiplerinin yapısını bölmek veya birleştirmek gibi sebeplerden dolayı birden fazla kayıt tipi arasındaki ilişkileri değiştirir.

Kayıtlar-arası ilişki seviyesinde program-veri bağımsızlığı, tüm kaydın bölünemez yapısal eleman olduğunu farzeder. Eğer sistem, bir program veritabanındaki, hepsine bağlı olandan daha çok, bazı kayıtları(veya grupları) gözlemlemesine izin verirse, kayıtlar-arası bağımsızlığın bir ölçüsü elde edilmiş olur. Bu durumda, yeni kayıtlar(veya gruplar), var olan programları etkilemeksizin veritabanına tanıtılabilir. Benzer şekilde, programların bağlı olmadığı kayıtların veya grupların yeniden tanımlanması programı etkilemeyecektir. Elbetteki bir kayıt veya grubun silinmesi, bu kayıt tipini referans eden tüm programları etkileyecektir.

Program-veri bağımsızlığının bir ölçüsü, grupların birbiriyle olan ilişkisini, gerçekte veritabanında varolduğundan farklı bir şekilde gözlemleyebilirse, sağlanır.Bu, örneğin, bir programcı kullanıcı-şemasının, bir veritabanındaki gruplar arasında üste-bağımlı(parent-dependent) ilişkiyi değiştirebileceği durumları içerir.

Bu, program-veri bağımsızlığının elde edilebileceği en zor seviyelerden birisidir. Var olan veri sistemerinin çoğu, kullanıcılara ağaç yapılı dosyalar veya biraz daha genel olarak verinin Network Modellerini sağlar. Çalışması için bu sistemlerle geliştirilen uygulama programları, eğer ağaçların veya programların yapısı değişirse bozulmaya eğilimlidir.

2.3.4 Dosya-Seviye Eşleştirmeler

Bir dosyanın kayıt örnekleri üzerinde hareket eden dosya-seviye eşleştirmeler birbirinden bağımsız olmaktan daha çok bir birikim şeklindedir. Dosya-seviye eşleştirmeler bir Bool seçme tanımına dayalı dosyadan kayıtlar seçmeyi, bir dosyadan tanımlanmış istem doğrultusunda kayıtlar istemeyi ve daha fiziksel seviyede dosyaya indeksler(anahtarlar) yaratmayı veya dosyadan indeksler silmeyi , taşmış bir dosya alanından esas dosya alanına kayıtlar birleştirmeyi içerir.

Dosya-seviye bağımsızlığı hareketlerin, dosyayla bir bütün şeklinde uğraşmasını sağlar. Eğer veritabanına yeni bir dosya eklenirse bir program etkilenmemektedir. Veritabanından bir dosyanın silinmesinin, bunu referans eden herhangi bir programın çalıştırılmasını imkansız kılacağı açıktır. Böyle bir harekete öncelik verecek veritabanı yöneticisi, silinecek dosyanın varlığından etkilenen hiçbir aktif programın olmadığından emin olmalıdır. Lojiksel dosyaları birleştirme veya bölme, dosya-seviye değişikliğidir. Bir programın, değişik aşamalarda değişik dosyaları referans ettiği durum dosya-seviye bağımsızlığı olarak adlandırılır.

2.4 VERİTABANI REVİZYONU

Veritabanı ve veritabanı yönetim sistemlerinin işlenmesiyle, kullanımıyla ilgili olarak revizyon (gözden geçirme), yeniden tanımlama (redefinition), yeniden yapılandırma (restructure), yeniden düzenleme (reorganization) gibi kelimeler kullanılmıştır.

Revizyon (yeniden gözden geçirme) işlemi daha önceden tanımlanan veya yaratılan veritabanının lojiksel yapısının veya fiziksel depolama yapısının değiştirilmesidir. Revizyon, bir şemayı ve veritabanı kurulduktan sonra onun veritabanını değiştirme işlemidir (genelde belli bir zaman dilimi için kullanılır).

Revizyon, günleme yapma (kurulu bir veritabanının içeriğinin değiştirilmesi işlemi) ile karıştırılmamalıdır. Günleme, içerik gelişimidir, bir veritabanı yönetim sistemine sağlanan, hemen hemen evrensel olarak iyi-kabul görmüş, normal bir fonksiyondur. Revizyon, bir veritabanının lojiksel veya fiziksel yapısının değiştirilmesi, *yapının gelişimidir.* Yapısal değişiklikler sistemini ve onun kullanıcı çevresini bozucu olmasından dolayı, revizyon fonksiyonunun kullanımı , veri tabanı yöneticisinin dışardan kontrolü altında olmalıdır.

Bir organizasyon, kurulmuş bir veritabanını gözden geçirmek isterse (veya zorunda kalırsa), sonuçta oluşacak bir kaç aktivite hesaba alınmalıdır.Depolanmış veritabanı tanımını günlemek için ilk ve en basit adım, *yeniden tanımlama'* dır. Bu, hemen, depolanmış veritabanını, gözden geçirilen tanıma uygun hale getirmek için bir dönüşüm gerektirmektedir. Eğer sadece fiziksel depolama yapısı değişmişse, veritabanını *yeniden düzenleme* gerekmektedir.

Yeniden yapılandırma dönüşüm işlemi, veritabanının lojiksel yapısı değişmişse gereklidir. Bundan sonra, gözden geçirilen veritabanında, içerilen kullanıcı uygulama programlarında , kataloglaştırılan sorgulamalarda, depolanan rapor tanımlarında, günleme işlemi tanımlarıyla birlikte depolanan transaction tanımlarında, yetki verilen kullanıcıların profillerinde, ve herhangi kullanıcı şemasını onun eşleştirme tanımıyla birleştirmede rolü olan her işlemin incelenmesi ve belki de değiştirilmesi gerekmektedir.Son olarak, gözden geçirilen veritabanını kullanan kişilerin kafasındaki veritabanı imajının değiştirilmesi gerekmektedir. Bu, kullanıcı el kitaplarının ve insana-yönelik dökümantasyonların gözden geçirilmesini, değişiklik haberlerinin yayınlanmasını, eğitim oturumlarının yönetilmesini içerir. Çoğu organizasyonların, dış baskılar haricinde kurulu veritabanlarına revizyon yapmayacakları açıktır.

Yeniden tanımlama (redefinition), basit anlamda veritabanı tanımını değiştirmektir, kavramsal olarak doğrudan ileriye yönelik yapılan bir işlemdir.Problem, yeniden tanımlama tarafından ima edilen diğer her şeyin, veritabanı birleştirmenin varlığını

sürdürecek şekilde yapılmasıdır. Daha önceden kurulmuş veritabanının tanımını değiştirme geçersiz veriye sebep olacaktır, yani, veritabanında önceden saklanmış verinin, değiştirilmiş tanım altında geçersiz olmasını sağlayacaktır.Saklanmış veritabanının tanımına uyması uzun sürmeyebilir.

Yeniden yapılandırma(restructuring), saklanmış veritabanının, yeni tanımıyla uyum sağlayacak şekilde dönüştürülmesi işlemidir.Yeniden tanımlama, içeriklerde(yeni veya silinmiş), var olan içeriklerin yapısında, veya var olan içeriğin fiziksel gösteriliminde bir değişikliği içerebilir. Revizyon seviyelerin herhangi birisinde yer alabilir ve eklemeyi, günlemeyi, veya silmeyi içerebilir.

Veritabanının lojiksel tanımı ve fiziksel depolama yapısı arasındaki bağımlılığın derecesine bağlı olarak,yeniden tanımlamanın bazı durumları herhangi bir yeniden yapılandırılmayı gerektirmeyebilir. Bu, yeni bir kayıt tipi ekleme veya bir gruba yeni bir veri öğesi ekleme gibi ciddi ekleme değişikliklerinin durumu olabilir.

Yeniden düzenleme(reorganization), yeniden yapılandırmanın bir özel durumu olarak adlandırılır.Fiziksel depolama yapısı değişirken, lojiksel yapı ve içerik değişmezliğini sürdürür. Çok ağır bir aktiviteden sonra, bir mağazanın raflarındaki mallarını yeniden ayarlaması gibi, bir süre çeşitli ekleme ve silmeleri içeren kullanımdan sonra veritabanının yeniden düzenlenmesi gerekmektedir. Yeniden düzenlemenin tek amacı veritabanı işlemenin performansını artırmak veya depolama yapılacak boş yer kullanımını artırmaktır. Yeniden düzenleme(bazı sistemlerde yapmasına rağmen) insanlar üzerinde ve performans yönünün ötesindeki kullanıcı çevresindeki işlemlere hiçbir etki yapmamalıdır.

Yeniden düzenleme, lojiksel olarak silinmiş kaydın depolandığı boşluğu silme (purge etme) taşan alanlardaki veriyi, esas dosya ile birleştirme, fiziksel gösterge (pointer) mekanizmalarını günleme, indeksler yaratma veya silme ve depolanmış veriyi sıralamayı içerir.

2.5 VERİ MODELLERİ VE SİSTEMLERİNİN KARŞILAŞTIRILMASI

2.5.1 Veri modeli gösterilişlerinin karşılaştırılması

Veri modeli gösterilişleri arasındaki en önemli ayrım, ilişkilerin nasıl gösterildiğine bağlıdır. İlişkisel modelde, iki ilişki arasındaki bağlantı bir ilişkide diğer ilişkinin birincil anahtarını (primary key'ini) gösteren yabancı anahtar (foreign key) alanlarıyla temsil edilir. Birincil ve yabancı anahtarlardan değerler seçen tek kayıtlar, fiziksel olarak bağlı olmasalar bile mantıksal olarak ilişkilidirler.

Network modelde 1:N bağlantılar açıkça küme tipi yapısıyla temsil edilir ve DBMS 'in fiziksel olarak ilişkili kayıtları bir küme örneğinde bir araya gelirler (Elmasri, Navathe 1994).

İlişkisel modeller lojiksel veya sembolik değerleri kullanırlar diyebiliriz. Üye kayıtlara sahip olan kaydın anahtar alanını çoğaltmak suretiyle, fiziksel referanslara ek olarak lojiksel referansları bir Network veritabanında tutabiliriz. Eğer bu çoğaltma tüm küme tipleri için yapılırsa Network şemadaki kayıt tipleri, İlişkisel şemada karşılık gelen ilişkilere benzer. Diğer yandan eğer üye kayıtlara sahip olan kaydın anahtar alanını çoğaltmazsak, sistemin geçerli küme üyeliğini kontrol etmenin hiçbir anlamı olmaz.

Örnek kurmak için, CONNECT(bağlan), DISCONNECT(bağlantıyı kes), RECONNECT(yeniden bağlan) veya STORE(depola) komutlarını kullanarak üye kayıtları fiziksel olarak link etmek kullanıcının sorumluluğundadır. STORE komutu AUTOMATIC küme üyeliği belirlendiği zaman link etmeyi sağlar.

Hiyerarşik model de ilişkileri açıkça gösterir fakat, Network Modelle karşılaştırıldığında çok ciddi sınırlamalara sahiptir. Network Modeldeki kayıt tipinin, herhangi sayıda küme tipi elemanı olabilirken, hiyerarşik modeldeki kayıt tipinin bir gerçek ve bir sanal parent'ı(anası) vardır. Bu, M:N ve n'li ilişki tiplerini modellemede sorun

yaratır. Eğer bir şema başlıca 1:N 'li aynı yönde ilişki içeriyorsa, doğal olarak hiyerarşik olarak modellenir. Fakat birçok ilişki tipinde mevcut pointer kullanmayarak ya da bazı kayıtları tekrarlamayarak iyi bir hiyerarşiksel gösterim sağlamak zordur. Sonuç olarak, Hiyerarşik Modelin genellikle modelleme yeterliliği bakımından İlişkisel ve Network Modelden daha güçlü olduğu görülür.

Nesneye Yönelik (Object-Oriented) modellerde, ilişkiler tipik olarak nesnelerin tanımlayıcıları yoluyla elde edilen referanslarla gösterilir. Kullanıcı tanımlı alanlardan daha çok dahili (internal) sistemlerin kullanılmasının dışında, bazı yönlerden yabancı anahtarlara benzer. Nesneye Yönelik modeller, kayıt (tuple), küme, liste ve diğer yapıları kullanmak suretiyle karmaşık nesne yapılarını destekler. Bunlara ilaveten, metodların belirlenmesini ve varolan sınıf tanımlarında yenilerini yaratmaya izin veren kalıtım mekanizmalarını destekler.

İç İçe İlişkisel Model (Nested Relational Model), düz ilişkilerin yanısıra hiyerarşik olarak yapısallaşmış ilişkilerin yaratılmasını da destekler. Bazı bakımlardan iç içe ilişkiler, iç içe biçimindeki kayıt ve küme kurucuların kullanımına benzer. Bu yüzden iç içe ilşkilerin yapılandırma yeterliliklerini, kurucuların Nesneye Yönelik Modelinin bir alt kümesi olarak nitelendirebiliriz.

ER Model Kavramlarını İlişkisel,Network,Hiyerarşik Ve Nesneye Yönelik Modellerle Eşleştirme

ER Model Kavramı	İlişkisel Model	Hiyerarşik Model	Network Model	Nesneye yönelik Model
Kayıt (entity) tipi	İlişki(relation) olarak	Kayıt(record) tipi olarak	Kayıt(record) tipi olarak	Sınıf(Class) olarak
Zayıf kayıt tipi	İlişki olarak , fakat belirleyici ilişkinin birincil anahtarını içerir	Belirleyici kayıt tipinin çocuğu olan kayıt tipi olarak	Kayıt tipi olarak bir küme tipindeki belirleyici kayıt tipiyle birlikte tablo yaratıcı (veya tekrar eden grup) gibi olan elemandır	Sınıf olarak, veya bir sahip sınıfındaki kayıtların kümesi olarak
1:1 İlişki tipi	Bir ilişkinin birincil anahtarını diğer bir ilişkinin yabancıl anahtarı olarak içerir veya tek ilişkide toplar	Örnekleri tek çocuk kaydına sahip olacak şekilde sınırlanan **PCR*** tipini kullanır veya tek kayıt tipine birleştirir	Örnekleri tek eleman kaydına sahip olacak şekilde sınırlanan küme tipini kullanır veya tek kayıt tipine birleştirir	Birbirinin tersi ve ikisi de basit olan iki referans alanıdır
1:N İlişki tipi	İlişkinin 1'li tarafının birincil anahtarını, N'li tarafın yabancıl anahtarı olarak içerir.	Ana-çocuk ilşki tipini kullanır	Küme tipini kullanır	Ters ve birinin değeri verilmiş olan iki referans alanıdır
M:N İlişki tipi	İçerilen ilişkilerin birincil anahtarı olan yabancıl anahtarı içeren yeni bir ilşki kurar	(a)Tek hiyerarşi ve çoğaltılmış kayıtları kullanır (b)Çoklu hiyerarşileri ve **VPCR**** tiplerini kullanır	Link edilmiş kayıt tipini kurar ve onu katılan kayıt tiplerinin sahip olduğu kümenin bir elemanı yapar	İki referans alanının da değeri verilmiştir veya aşağıdaki gibidir
n'li İlşki tipi	M:N 'deki gibidir	(a)M:N 'deki gibidir (b)ilişkileri ana gibi, ve katılan kayıt tiplerini tek hiyerarşideki çocuklar gibi yapar.	M:N 'deki gibidir	n referanslı yeni bir sınıf yaratır

Tablo 2.5.1.1 Veri modeli gösterilişlerinin karşılaştırılması

* Parent-Child Relationship
** Virtual Parent-Child Relationship

2.5.2 Veri yönetme dillerinin karşılaştırılması

İki tip veri tabanı dili ayırt edilmiştir: Kayıt kümelerini işleyecek yüksek-seviye dilleri ve aynı zamanda tek kayıt işleyen düşük-seviye dilleri. Birçok yüksek seviye veritabanı dili İlişkisel modellerle ilgili olduğu halde, Network ve Hiyerarşik modeller zamanda-bir-kayıt düşük-seviye dilleriyle ilgilidir. Nesneye yönelik sistemler, topluluklardan nesne altkümelerinin seçimini destekleyen yüksek seviye sorgulama dillerine sahip olmaya ek olarak , bir nesneye yönelik programlama dilinde tipik olarak zamanda-bir-nesne programlamasını destekler.

İlişkisel model birkaç yüksek-seviye diline sahiptir. İlişkisel cebrin formal operasyonları kayıt kümelerine uygulanır. Bir sorgu, ilişkilerdeki *bir dizi* işlemler olarak tanımlanır. İlişkisel analizde, tek bir anlatım (bir dizi işlemlerden daha çok) bir sorgu tanımlar; nasıl bulacağımızla ilgili hiçbir komut vermeden sadece ne istediğimizi tanımlarız. Bundan dolayı ilişkisel analiz, ilişkisel cebre oranla daha yüksek tanımlama özelliğine sahiptir. İlişkisel model için olan ticari dillerin, örneğin SQL, QUEL, QBE gibi, temeli İşkisel analize dayanmaktadır.Bunlar ayrıca, temel ilişkisel cebir veya analizin dışında kalan birleştirme fonksiyonları, gruplandırma, sıralama, kayıtları çoğaltma ve aritmetik işlemler için birçok kolaylık sağlarlar. Çoğu ilişkisel sistemler kullanıcıların doğrudan ve interaktif olarak sorgulama girmelerine veya sorgulamayı programlama diline gömmelerine izin verirler.

Network ve Hiyerarşik veri yönetme dili (DML) 'nin tartışılan konuları düşük-seviyededir, çünkü bunlar tek kayıtları arayıp bulmak içindir.Bundan dolayı genel-amaçlı programlama dili kullanmak ve veritabanı komutlarını programa gömmek zorunluluğu ortaya çıkar. Bu dillerin her ikisinde de, o andaki kayıt (current record) kavramı, DML komutlarının manasını açıklamak için oldukça önemlidir; çünkü komutun etkisi o andaki kayda bağlıdır. Network model DML, DML komutlarının sonuçlarını etkileyen, örneğin o andaki küme tipi ve o andaki kayıt tipi gibi, ek göstergeler kullanır. Bu, anlık kavramlar,

zamanda-bir kayıt erişimini kolaylaştırmalarına rağmen, programcı doğru programlar yazabilmek için değişik komutların anlık göstergeleri nasıl etkileyeceğini tam olarak bilmek zorundadır. Bu zamanda-bir kayıt komutlarının orijinali geleneksel dosya-işleme komutlarından gelmektedir.

İlişkisel modelin gözle görülür bir avantajı vardır. İlişkisel modelle ilgili yapısal ve ticari dillerin her ikisi de oldukça güçlü ve yüksek-seviyelidir. Bir SQL standardının kendisi de devamlı gelişmekte olmasına rağmen var olması büyük bir avantajdır.

2.5.3 Saklama yapılarının karşılaştırılması

Günümüzde veri depolama tekniklerinin en önemlisi indekslemedir (anahtar kurma) ve hızlandırma tekniklerinin kullanımında da artmaktadır.

İlişkisel model için, genel teknik, her taban ilişkiyi ayrı bir dosya olarak tutmaktır. Eğer kullanıcı hiçbir depolama yapısını belirlemezse, çoğu İlişkisel DBMS'ler kayıtları tabloda düzensiz bir şekilde saklayacaklardır. Birçok ilişkisel DBMS kullanıcının her tabloda dinamik olarak bir primary(birincil) anahtar veya kümeleme(toplama) indeksi ve herhangi sayıda ikincil indeksi vardır. İndekslerin kurulacağı alanları seçme sorumluluğu kullanıcıya aittir. DBMS' lerin artan sayısı, hem kayıtları saklamak için asal organizasyon olarak hem de indeksleme yapıları olarak parçalama tekniklerinin belirlenmesine izin verir. Genelde, hem seçme hem de durumları birleştirmede sıklıkla kullanılacak olan tüm alanlarda indeksler veya karma anahtarlar yaratma tavsiye edilmektedir. Tek(unique) indeksler, anahtar kısıtlarını kuvvetlendirmek için verimli bir teknik sağlarlar. Çoğu ilişkisel DBMS'ler kullanıcıların ilişkileri dinamik olarak silmelerine olanak sağlar.

Bazı DBMS'ler, kullanıcıların, bazı kayıtların temel ilişkilerini birbirine karıştırmasına izin verir.Bu, birden fazla ilişkiye sahip bazı ilişkili kayıtlara sık sık beraber erişilmesi gerektiğinde faydalıdır. Kayıtları bu şekilde biraraya getirme, bir kaydı ilişkili kayıtlar tarafından takip edilen bir ilişkiden alıp, başka bir ilişkiye yerleştirir. Böylece

ilşkili kayıtlara mümkün olan en verimli şekilde ulaşılabilir. Bu işlem, bu yapıyı gösteren Network ve Hiyerarşik sistemlerde de sıklıkla kullanılır.

Hiyerarşik model genellikle, veritabanının hiyerarşik yapısını koruyan hiyerarşik dosyaları kullanmak yoluyla gerçekleşir.

Bunlara ek olarak, indekslemeyi ve pointer (gösterge)'leri içeren birçok seçenek vardır. Bunun için, tek kayıtlara ve ilişkili kayıtlara verimli şekilde erişebiliriz. Çoğu hiyerarşik sistem, veritabanını ayarlama (tuning) için böylesi seçenekler sağlar.

Aşağıda bir veritabanını dönüştürmeye başlamadan önce yapılması gerekenler listelenmiştir.

2.6 SİSTEM DÖNÜŞTÜRME PLANINI GÖZDEN GEÇİRME

I-Tanım

- İhtiyaçları, prosedürleri ve aşağıdakileri içeren özel programlamayı kurmaktır:
    - Yeni master kayıtlar
    - Yeni dosya-oluşturan (file-build) programlar
    - Yeni formları üretme
    - Yeni depolama aygıtları
    - Veriyi saflaştırma eforunu gözden geçirme

II-Amaçlar

- Aşağıdaki maddeleri içeren sistem araçlarını hazırlama.
    - Kavramsal Yaklaşım
    - Stratejiler
    - Taktikler
- Yeni sisteme dönüşüm için aşağıdaki maddeleri içeren detayların dökümanını yapma.

- Dönüşüm aktivitesinin kontrolü
- Ek programları tahmin etme

III-Organizasyon

- Organizasyon ve sorumluluk şemasındaki, geliştirme müdürüne , program yöneticisine, sistem programcılarına, uygulama programcılarına, teknik destek müdürüne, konfigürasyon yönetim uzmanlarına, kalite-güvenlik uzmanlarına, testçilere, destek müdürüne, analiz denetimcisine, sistem analistlerine, metod analistlerine ve proje yöneticisine başvurmak gerekmektedir.

2.7 SİSTEM DÖNÜŞTÜRME PLANI

I-Aşağıdakileri içeren baş sayfayı hazırlama:

- Döküman başlığı
- Döküman numarası
- Orjinal yayım tarihi
- En son yayım tarihi
- En son gözden geçirme tarihi
- Uygun imzalar ve tarih

II-Aşağıdakileri içeren günleme yapraklarını hazırlama:

- Sıralı olarak numaralandırılmış değişikliklerin listesi
- Değişikliklerin açıklaması
- Değişikliklerin sayfa numarası

- Uygun imzalar ve tarih

III-Aşağıdakilerin içerildiği, içerikler tablosunu hazırlama:

- Ana bölüm başlıkları
- Bölüm başlıkları
- Bölüm altbaşlıkları
- İlgili sayfa numaraları

IV-Aşağıdakileri içeren özeti hazırlama:

- Gözden geçirme
- Başlıca özellikler

V-Aşağıdakileri içeren sunuşu hazırlama:

- Amaçlar
- Kapsam
- Aşağıdaki gibi amaçlar
  - Teknik
  - İş
- Arka plan

VI-Aşağıdakileri içeren program kimlik ve dökümantasyonunu hazırlama:

- Gerekli değişiklik ve düzenlemeler
- Gerekli olmayan değişiklik ve düzenlemeler
- Prosedür ve taktikler
- Standart ve alışılan kurallar
- Erişim güvenliği

- Seçim sağlayıcı

VII-Aşağıdakileri içeren operasyonel ihtiyaçları hazırlama:

- Giderler
- Programlar
- Personel
- Donanım
- Yazılım
- Destek
- Olanak

VIII-Terimler sözlüğü hazırlama

IX-Ekleri ilave etme

2.8 SİSTEM DÖNÜŞTÜRME KONTROL LİSTESİ

I- Sistem dönüşümü için aşağıdaki gereksinimleri tanımlama ve döküman hazırlama:

- İşin durumu, örneğin:
  - İmza ve onaylanma tarihi gibi
- Kaynaklar, örneğin:
  - Zaman
  - Malzemeler
  - İşgücü
  - Tutar
- Dönüşüm kriterleri
- Standartlar
- Organizasyonal plan, örneğin:
  - İmza ve onaylanma tarihi gibi
- İmkanlar planı, örneğin:
  - İmza ve onaylanma tarihi gibi

II- Aşağıdakileri içeren, sistem dönüşümü için değişiklikleri adresleme:

- İşlem(ler)
- Prosedür
- İşletim sistemi
- Yazılım
- Donanım
- Programlama dili
- Network mimarisi
- Dosya araçları

III- Aşağıdakileri içeren, çeşitli alanları tanımlama ve dökümante etme:

- Üstyönetim ve kullanıcı-topluluğu desteği
- Hedef çevre
- Dönüşüm esnasındaki işlem(ler)in etkisi
- Dağıtım programı tarihleri
- Karmaşıklık faktörleri
- Riskler ve bağlı etkileri

2.9 VERİ DÖNÜŞTÜRME PLANINI GÖZDEN GEÇİRME

I-Tanım

- Veri dönüştürme için aşağıdakileri içeren esnek bir çevre oluşturmak:
  - Taktikler, prosedürler, işlemler, metodolojiler ve standartlar
  - Basitleştirme ve geliştirme alanları
  - Etki alanları
  - İndirgenmiş veri havuzları

II-Amaçlar

- Veri dönüştürme yönetim ve ayarlamasını sağlama

III-Organizasyon

- Organizasyon ve sorumluluk şemasındaki, teknik destek müdürüne, veritabanı yöneticilerine, konfigürasyon yönetim uzmanlarına, kalite-güvenlik uzmanlarına, destek müdürüne, analiz denetimcisine, sistem analistlerine, proje yöneticisine başvurmak gerekmektedir.

2.10 VERİ DÖNÜŞTÜRME PLANI

I-Aşağıdakileri içeren baş sayfayı hazırlama:

- Döküman başlığı
- Döküman numarası
- Orjinal yayım tarihi
- En son yayım tarihi
- En son gözden geçirme tarihi
- Uygun imzalar ve tarih

II-Aşağıdakileri içeren günleme yapraklarını hazırlama:

- Sıralı olarak numaralandırılmış değişikliklerin listesi
- Değişikliklerin açıklaması
- Değişikliklerin sayfa numarası
- Uygun imzalar ve tarih

III-Aşağıdakileri içeren içerdekiler tablosunu hazırlama:

- Ana bölüm başlıkları
- Bölüm başlıkları
- Bölüm altbaşlıkları
- İlgili sayfa numaraları

IV-Aşağıdakileri içeren özeti hazırlama:

- Gözden geçirme
- Başlıca özellikler

V-Aşağıdakileri içeren sunuşu hazırlama:

- Amaçlar
- Kapsam
- Aşağıdaki gibi amaçlar
  - Teknik
  - İş
- Arka plan

VI-Aşağıdakileri içeren iz ve kütük farklılıkları raporlarını hazırlama:

- Problem raporları
- Arayüz gözden geçirmeler

VII-Aşağıdakileri içeren güvenlik ve dökümantasyon kurma:

- Test veri üretme
- Test çıkış ve veri dosyaları
- Gerçekleri beklenenlerle karşılaştırma

VIII-Aşağıdakileri içeren veri/veritabanı arayüzlerini birleştirme:

- Veri tanımları

- Veri sözlüğü
- Veritabanı kontrolü
- Veritabanı bakımı

IX-Aşağıdakileri içeren kütüphane fonksiyonlarını birleştirme:

- Kod değişiklikleri
- Yedekleme kopyaları
- Sistem jenerasyonları
- Kayıt koruma

X-Terimler sözlüğü hazırlama

XI-Ekleri ilave etme

2.11 VERİ DÖNÜŞTÜRME KONTROL LİSTESİ

I- Aşağıdakileri içeren gereksinimleri kurma, tanımlama ve yönetme:

- Veritabanı çevresi
- İş kuralları ve metodları
- Sistem arayüzleri
- Fazlalıkları indirgeme ve yok etme
- Güvenlik seviyelerine erişim

II- Veri dönüştürme tanım ve standartlarını yaratma ve sürdürme

- İşlem standartlarını isimlendirme
- Veri elemanları stendartlarını isimlendirme
- Toolset tanımları
- Veri sözlüğü elemanları
- Veri ilişkileri
- Veritabanı ilişkileri

III- Aşağıdakileri içeren veri dönüştürme performansını belirleme:

- Veritabanı performansını raporlama
- Veritabanı kontrollerini izlemek

SONUÇLAR

Veritabanları ve veritabanı teknolojisi, bilgisayar kullanımının artışında önemli bir etkiye sahiptir. Veritabanları, bilgisayarların kullanıldığı alanlarda, örneğin iş, mühendislik, tıp, hukuk, eğitim, kütüphanecilik gibi çok büyük rol oynamaktadır.Veritabanı, ilişkili verilerden; veritabanı yönetim sistemi ise kullanıcıların, veritabanı yaratmalarını ve yönetmelerini sağlayan programlardan oluşmaktadır.

Hiyerarşik modelde veriler, tek köke sahip ağaç yapısı şeklinde, hiyerarşik olarak saklanır ve veriler arasında ana-çocuk ilişkisi vardır. Bu veriler arsındaki bir ilişkiyi değiştirmek çok zordur, örneğin bir çocuk düğümün yerini değiştirmek, bir seviye yükselterek ana düğüm seviyesine kaydırmak veya hepten kaldırmak ona bağlı diğer çocuk düğümleri de etkileyecektir. Her değişiklik sonucunda veritabanının yeniden yapılandırılması gerekmektedir. Hiyerarşik modelde bir kaydın bulunması için hiyerarşiler arasındaki ilişkiler kullanılır.

Hiyerarşik veritabanında kullanılan ana-çocuk (parent-child) kavramından çok, network veritabanı, bir veya daha çok kümeye ait olan veri öğeleri kümesi kavramını tanıtır. Network modelde kayıtlar arasındaki ilişkiyi çözmek için gösterge(pointer)'lerden yararlanılır.

İlişkisel modelin avantajı, en azından kavramsal olarak, verilerin, kullanıcıların hemen anlayabileceği şekilde saklanmasıdır. Veri, tablolar halinde saklanır ve tablonun satırları arasındaki ilişkiler veride görülebilir. Bu model, veriler arsındaki ilişkileri tanımlamak suretiyle çok büyük esneklik sağlamıştır.Veritabanına bir tablo ekleme veya veritabanından bir tablo silme, veya bir tabloda değişiklik yapma diğer tabloları etkilemez. Daha önceden var olan diğer tablolarla olan ilişkisi belirlenen tablo kolayca veritabanına eklenebilir. Diğer tablolarla ilişki kurmayı sağlayan verinin saklandığı kolonlar sayesinde aranılan kayıt, ilişkili tablolarda kolayca bulunur. Bu modelde, tekrarlanan veri miktarı da azaltılmıştır. İlişkisel veritabanı modelinin yönetimi de oldukça kolaydır.

İlişkisel Veritabanı Modelinin tezde tartışılan üstünlükleri, ondan önceki diğer veritabanı modellerinden bu modele geçişi gerekli kılmıştır. Ancak veritabanı dönüştürmeden önce, sistem ve veri dönüşümü için planlar ve kontrol listeleri yapılmalı, bunların sonucunda dönüşüm için nelerin sağlanması gerektiği ortaya çıkarılmalı, elde olanlar değerlendirilmelidir.

Var olan veritabanı işlemcisini, veritabanı yönetimine doğrudan eşleştirme kötü bir uygulamadır. Nadir olarak eldeki DBMS tool' unun (aletinin) avantajlarını kullanan bir uygulamayla sonuçlanır. Sonuç genellikle orjinal sistemden daha iyidir. Bir DBMS'ten diğerine dönüşüm(conversion) yapılırken, sadece iki DBMS aynı ise ve orjinal veritabanı iyi tasarımlanmışsa eşleştirme yönteminin uygulaması tavsiye edilmektedir. Eğer DBMS'ler farklıysa, eşleştirme işlemi yeni sistemin güçlü olmasını kolaylaştırmayacaktır. Dahası, eski sistemin güçlü olduğu yönleri yeni sistemde zayıf olabilecektir bu yüzden sonuçtaki uygulama eski sistem kadar iyi icra edilemeyecektir. Bu, özellikle, eski ve yeni sistemler birbirlerinden önemli ölçüde farklıysa doğru olacaktır. Genelde, doğrudan bir veritabanı dönüşümü iyi bir uygulamayla sonuçlanmaz. Doğrudan dönüştürme, bir tasarım metodu değildir, fakat bir tasarımdan kaçma metodudur. Mümkün olduğu kadar bundan sakınılmalı , kullanılması kaçınılmaz olan durumlarda ise büyük dikkatle kullanılmalıdır.

KAYNAKLAR

Connoly T.M. , Begg C.E., Strachan A.D.,1995.
"Database systems, A Practical Approach to Design, Implementation and Management"

Elmasri R., Navathe S.B.,1994. "Fundamentals of Database Systems"

Everest G.C.,1993. "Database Management"

Klein R.L., Ludin I.S.,1993. "Data processing manager's Model reports & formats"

Kramer D., 1996 "Introduction to Oracle:SQL and PL/SQL Using Procedure Builder"

Kroenke D.,Dolan K. 1994, "Database Processing"

Modell M.E. , 1995."Data Analysys,Data Modelling, and Classification"

Parsaye K., Chignell M., Khoshafian S., Wong,H.,1993.
"Intelligent Databases Object Oriented, Deducting Hypermedia Technologies"

Rob P., Coronel C., 1993."Database Systems Design, Implementation and Management"

Stamper D.,Price W.C.,1990 "Database Design and Management"

EKLER

UYGULAMA

Aşağıdaki COBOL programlama diliyle yazılmış Sipariş programında kullanılan dosyalar, Visual BASIC'te yazılmış olan bir **conversion(dönüştürme)** programı kullanılarak, ilişkisel tablolara dönüştürülecektir:

```
IDENTIFICATION DIVISION.
PROGRAM-ID. SIPARIS.
AUTHOR. SELMA KARACABEY.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
       SELECT DOSYA ASSIGN TO DISK
       ORGANIZATION IS INDEXED
       ACCESS MODE IS DYNAMIC
       RECORD KEY IS SNO
             ALTERNATE RECORD KEY IS MKOD WITH DUPLICATES
             ALTERNATE RECORD KEY IS DKOD WITH DUPLICATES
             ALTERNATE RECORD KEY IS TAR WITH DUPLICATES
             FILE STATUS IS DURUM.
       SELECT DOSYA1 ASSIGN TO DISK
       ORGANIZATION IS INDEXED
       ACCESS MODE IS DYNAMIC
       RECORD KEY IS SKOD
             ALTERNATE RECORD KEY IS SAD WITH DUPLICATES
             ALTERNATE RECORD KEY IS BIRIM WITH DUPLICATES
             FILE STATUS IS DURUM1.
       SELECT DOSYA2 ASSIGN TO DISK
       ORGANIZATION IS INDEXED
       ACCESS MODE IS DYNAMIC
       RECORD KEY IS DKOD1
             ALTERNATE RECORD KEY IS DAD1 WITH DUPLICATES
             FILE STATUS IS DURUM2.
       SELECT DOSYA3 ASSIGN TO DISK
       ORGANIZATION IS INDEXED
       ACCESS MODE IS DYNAMIC
       RECORD KEY IS MKOD1
             ALTERNATE RECORD KEY IS FIRMA1 WITH DUPLICATES
             ALTERNATE RECORD KEY IS ILGILI1 WITH DUPLICATES
             ALTERNATE RECORD KEY IS ADRES1 WITH DUPLICATES
             FILE STATUS IS DURUM3.
       SELECT YAZICI ASSIGN TO PRINTER.
DATA DIVISION.
FILE SECTION.
FD YAZICI LABEL RECORD OMITTED
LINAGE IS 58 LINES TOP 3 BOTTOM 4.
01 YAZICI-KAYDI PIC X(80).
FD DOSYA LABEL RECORD IS STANDARD
    VALUE OF FILE-ID IS "SIPAR.DAT".
01 KAYIT.
```

```
    02 SNO         PIC 9(7).
    02 DKOD        PIC X(7).
    02 MKOD        PIC X(7).
    02 SKOD1       PIC X(7)  OCCURS 12 TIMES.
    02 SAD1        PIC X(10) OCCURS 12 TIMES.
    02 BIRIM1      PIC X(6)  OCCURS 12 TIMES.
    02 BIR-FI1     PIC 9(7)  OCCURS 12 TIMES.
    02 BIR-MI1     PIC 9(5)  OCCURS 12 TIMES.
    02 DAD         PIC X(20).
    02 FIRMA       PIC X(20).
    02 ILGILI      PIC X(18).
    02 ADRES       PIC X(20).
    02 TAR.
       03 GUN      PIC 99.
       03 AY       PIC 99.
       03 YIL      PIC 9999.
    02 ACIK        PIC X(20).
    02 AL          PIC X.
    02 ART         PIC 99.
    02 TUTAR       PIC 9(10).
    02 TUT11       PIC 9(9) OCCURS 12 TIMES.
FD DOSYA1 LABEL RECORD IS STANDARD
    VALUE OF FILE-ID IS "STOK.DAT".
01 KAYIT1.
    02 SKOD       PIC X(7).
    02 SAD        PIC X(10).
    02 BIRIM      PIC X(6).
    02 BIR-FI     PIC 9(7).
    02 BIR-MI     PIC 9(5).
    02 TUT        PIC 9(9).
    02 MEV-MIK    PIC S9(9).
    02 MEV-TUT    PIC S9(10).
    02 ALS        PIC 9.
FD DOSYA2 LABEL RECORD IS STANDARD
    VALUE OF FILE-ID IS "DEPT.DAT".
01 KAYIT2.
    02 DKOD1      PIC X(7).
    02 DAD1       PIC X(20).
FD DOSYA3 LABEL RECORD IS STANDARD
    VALUE OF FILE-ID IS "MUST.DAT".
01 KAYIT3.
    02 MKOD1      PIC X(7).
    02 FIRMA1     PIC X(20).
    02 ILGILI1    PIC X(18).
    02 ADRES1     PIC X(20).
WORKING-STORAGE SECTION.
77 DURUM  PIC XX.
77 DURUM1 PIC XX.
77 DURUM2 PIC XX.
77 DURUM3 PIC XX.
77 I PIC 99.
77 J PIC 99.
77 L PIC 99.
77 K PIC 99.
77 SAYAC      PIC 9(3).
77 SAYAC1     PIC 9(3).
```

```
77 D1KOD       PIC X(7).
77 M1KOD       PIC X(7).
77 S1KOD       PIC X(7).
77 SEC         PIC X.
77 KNO         PIC 9(7).
77 MEV-MIK1    PIC  +++,+++,+++.
77 MEV-TUT1    PIC  +,+++,+++,+++.
77 TOPTUT      PIC 9(11).
77 TOPTUT1     PIC ZZ,ZZZ,ZZZ,ZZZ.
77 TUT1        PIC ZZZ,ZZZ,ZZZ.
77 TUTAR1      PIC Z,ZZZ,ZZZ,ZZZ.
77 BIR-MI11    PIC ZZ,ZZZ.
77 BIR-FI11    PIC Z,ZZZ,ZZZ.
77 SNO-1       PIC 9(7).
01 W-TAR.
    02 IGUN1   PIC 99.
    02 IAY1    PIC 99.
    02 IYIL1   PIC 9999.
01 S-TAR.
    02 SGUN1       PIC 99.
    02 SAY1        PIC 99.
    02 SYIL1       PIC 9999.
77 STK-1        PIC X(7).
77 DEP-1        PIC X(7).
77 DEP-2        PIC X(7).
77 MUS-1        PIC X(7).
77 MUS-2        PIC X(7).
01 F PIC XX.
    88 ESC VALUE "01".
    88 F1  VALUE "02".
    88 F2  VALUE "03".
    88 F3  VALUE "04".
    88 F4  VALUE "05".
    88 F5  VALUE "06".
    88 F6  VALUE "07".
    88 F7  VALUE "08".
    88 F8  VALUE "09".
    88 F9  VALUE "10".
    88 F10 VALUE "11".
01 CEVAP PIC X.
    88 CEV VALUE "E" "e".
01 YAZ-555.
    02 FILLER PIC X(11) VALUE SPACES.
    02 FILLER PIC X(15) VALUE "SIPARIS NO'SU  ".
    02 K-SNO PIC X(7).
    02 FILLER PIC X(1) VALUE SPACES.
    02 FILLER PIC X(3) VALUE "ILE".
    02 FILLER PIC X(1) VALUE SPACES.
    02 K-SNO1 PIC X(7).
    02 FILLER PIC X(2) VALUE SPACES.
    02 FILLER PIC X(28) VALUE "ARASINDAKI KAYITLARIN DOKUMU".
    02 FILLER  PIC X(17) VALUE SPACES.
01 YAZ-333.
    02 FILLER  PIC X(10) VALUE "Siparis No".
    02 FILLER  PIC X(11)  VALUE " Tarih     ".
    02 FILLER  PIC X(21) VALUE " Departman          ".
```

```
    02 FILLER  PIC X(20) VALUE " Firma                ".
    02 FILLER  PIC X(12) VALUE "    Toplam  ".
    02 FILLER  PIC X(6)  VALUE "   A/S".
01 KAYIT-YAZ.
    02 K-SNO2 PIC X(7).
    02 FILLER PIC X(4) VALUE SPACES.
    02 K-GUN PIC 99.
    02 FILLER PIC X VALUE "/".
    02 K-AY PIC 99.
    02 FILLER  PIC X VALUE "/".
    02 K-YIL PIC 9999.
    02 FILLER PIC X VALUE SPACES.
    02 K-DAD PIC X(20).
    02 FILLER PIC X VALUE SPACES.
    02 K-FIRMA PIC X(20).
    02 FILLER PIC X VALUE SPACES.
    02 K-TUTAR PIC Z,ZZZ,ZZZ,ZZZ.
    02 FILLER PIC X VALUE SPACES.
    02 K-AL PIC X.
01 YAZ-444.
    02 FILLER PIC X(46)  VALUE SPACES.
    02 FILLER PIC X(17) VALUE "GENEL TOPLAM ==> ".
    02 K-GENEL PIC ZZ,ZZZ,ZZZ,ZZZ.
    02 FILLER PIC X(7) VALUE SPACES.
01 YAZ-BOS.
   02 FILLER PIC X(80) VALUE ALL SPACES.
01 YAZ-CIZ.
    02 FILLER PIC X(80) VALUE ALL "=".
SCREEN SECTION.
01 ANA-MENU.
    02 LINE  8 COLUMN 25 HIGHLIGHT  VALUE=
    "╔═══════════════════════════╗"
    02 LINE  9 COLUMN 25 HIGHLIGHT  VALUE
    "║     A N A    M E N U      ║".
    02 LINE 10 COLUMN 25 HIGHLIGHT  VALUE
    "╠ F1==>".
    02 LINE 10 COLUMN 32 VALUE
    "SIPARIS GIRISI".
    02 LINE 10 COLUMN 49 HIGHLIGHT  VALUE
    "    ╣".
    02 LINE 11 COLUMN 25 HIGHLIGHT  VALUE
    "╠ F2==>".
    02 LINE 11 COLUMN 32 VALUE
    "SIPARIS SILME".
    02 LINE 11 COLUMN 49 HIGHLIGHT  VALUE
    "    ╣".
    02 LINE 12 COLUMN 25 HIGHLIGHT  VALUE
    "╠ F3==>".
    02 LINE 12 COLUMN 32 VALUE
    "SIPARIS DUZELTME".
    02 LINE 12 COLUMN 49 HIGHLIGHT  VALUE
    "    ╣".
    02 LINE 13 COLUMN 25 HIGHLIGHT  VALUE
    "╠ F4==>".
    02 LINE 13 COLUMN 32 VALUE
    "LISTELEME".
```

```
    02 LINE 13 COLUMN 49 HIGHLIGHT  VALUE
    "    ‖".
    02 LINE 14 COLUMN 25 HIGHLIGHT  VALUE
    "‖ F5==>".
    02 LINE 14 COLUMN 32 VALUE
    "CIKIS".
    02 LINE 14 COLUMN 49 HIGHLIGHT  VALUE
    "    ‖".
    02 LINE 15 COLUMN 25 HIGHLIGHT  VALUE
    "‖           SECIMINIZ".
    02 LINE 15 COLUMN 45 HIGHLIGHT  BLINK  VALUE
    "==>[  ]".
    02 LINE 15 COLUMN 48 HIGHLIGHT  VALUE
    "    ‖".
    02 LINE 16 COLUMN 25 HIGHLIGHT  VALUE
    "╚═══════════════════════════════╝".
01 EKRAN1.
    02 LINE  3 COLUMN  7 VALUE
    "╔═══════════════════════════════════════════════════════════════
      -    "═══════╗".
    02 LINE  4 COLUMN  7 VALUE
    "‖".
    02 LINE  4 COLUMN 23 HIGHLIGHT  VALUE
    "S I P A R I S     G I R I S I".
    02 LINE  4 COLUMN 52 VALUE
    "                   ‖".
    02 LINE  5 COLUMN  7 VALUE
    "‖".
    02 LINE  5 COLUMN  9 HIGHLIGHT  VALUE
    "Siparis No :                    Tutar :".
    02 LINE  5 COLUMN 51 VALUE
    "                   ‖".
    02 LINE  6 COLUMN  7 VALUE
    "‖                 ┌──────────────────────────────────────────────
      -    "────────┐ ‖".
    02 LINE  7 COLUMN  7 VALUE
    "‖ DEPARTMAN  │Kod".
    02 LINE  7 COLUMN 27 HIGHLIGHT  VALUE
    ":".
    02 LINE  7 COLUMN 28 VALUE
    "              Ad".
    02 LINE  7 COLUMN 47 HIGHLIGHT  VALUE
    ":".
    02 LINE  7 COLUMN 48 VALUE
    "                   │ ‖".
    02 LINE  8 COLUMN  7 VALUE
    "‖                 ├──────────────────────────────────────────────
      -    "────────┐ ‖".
    02 LINE  9 COLUMN  7 VALUE
    "‖ MUSTERI    │Kod".
    02 LINE  9 COLUMN 27 HIGHLIGHT  VALUE
    ":".
    02 LINE  9 COLUMN 28 VALUE
    "             Firma ".
    02 LINE  9 COLUMN 47 HIGHLIGHT VALUE
    ":".
```

```
02 LINE  9 COLUMN 48 VALUE
"                              ¦ ‖".
02 LINE 10 COLUMN  7 VALUE
"‖             ¦Ilgili".
02 LINE 10 COLUMN 27 HIGHLIGHT  VALUE
":".
02 LINE 10 COLUMN 28 VALUE
"                                          ¦ ‖".
02 LINE 11 COLUMN  7 VALUE
"‖             ¦Adres".
02 LINE 11 COLUMN 27 HIGHLIGHT  VALUE
":".
02 LINE 11 COLUMN 28 VALUE
"                                          ¦ ‖".
02 LINE 12 COLUMN  7 VALUE
"‖ ┌──────────┴──────────────────────────────────────────────────────
  _    "──────┐ ‖".
02 LINE 13 COLUMN  7 VALUE
"‖ ¦Tarih".
02 LINE 13 COLUMN 17 HIGHLIGHT  VALUE
":".
02 LINE 13 COLUMN 21 HIGHLIGHT  VALUE
"/".
02 LINE 13 COLUMN 25 HIGHLIGHT  VALUE
"/".
02 LINE 13 COLUMN 26 VALUE
"    Aciklama".
02 LINE 13 COLUMN 40 HIGHLIGHT  VALUE
":".
02 LINE 13 COLUMN 64 HIGHLIGHT  VALUE
"A".
02 LINE 13 COLUMN 65 VALUE
"1/".
02 LINE 13 COLUMN 67 HIGHLIGHT  VALUE
"S".
02 LINE 13 COLUMN 68 VALUE
"a".
02 LINE 13 COLUMN 70 HIGHLIGHT  VALUE
":".
02 LINE 13 COLUMN 71 VALUE
"  ¦ ‖".
02 LINE 14 COLUMN  7 VALUE
"‖ └──────────┬──────────────────────────────────────────────────────
  _    "──────┘ ‖".
02 LINE 15 COLUMN  7 VALUE
"‖".
02 LINE 15 COLUMN 25 HIGHLIGHT  VALUE
"                ".
02 LINE 15 COLUMN 45 HIGHLIGHT  BLINK  VALUE
"   ".
02 LINE 15 COLUMN 48 HIGHLIGHT  VALUE
"      ".
02 LINE 15 COLUMN 54 VALUE
"                      ‖".
02 LINE 16 COLUMN  7 VALUE
"‖".
```

```
    02 LINE 16 COLUMN 25 HIGHLIGHT  VALUE
    "                              ".
    02 LINE 16 COLUMN 54 VALUE
    "                               ".
    02 LINE 17 COLUMN  7 VALUE
    "
      -    "          ".
    02 LINE 18 COLUMN  7 VALUE
    "
      -    "          ".
    02 LINE 19 COLUMN  7 VALUE
    "
      -    "          ".
    02 LINE 20 COLUMN  7 VALUE
    "
      -    "          ".
    02 LINE 21 COLUMN  7 VALUE
    "
      -    "          ".
    02 LINE 22 COLUMN  7 VALUE
    "
      -    "          ".
    02 LINE 23 COLUMN  7 VALUE
    "
      -    "          ".
    02 LINE 2 COLUMN 8 VALUE "ESC==>ANA MENU".
    02 LINE 2 COLUMN 30 VALUE "F9==>AYNI BOLUMDEKI BIR ONCEKI BIL
      -    "GI".
01 EKRAN2.
    02 LINE 15 COLUMN  9 VALUE
    "
      -    "      ".
    02 LINE 16 COLUMN  9 VALUE
    "|Stok  Kodu|Stok Adi     |Birim |Miktar |B.Fiyat  |Tutar
      -    "     ".
    02 LINE 17 COLUMN  9 VALUE
    "
      -    "      ".
    02 LINE 18 COLUMN  9 VALUE
    "|          |            |      |       |         |
      -    "     ".
    02 LINE 19 COLUMN  9 VALUE
    "|          |            |      |       |         |
      -    "     ".
    02 LINE 20 COLUMN  9 VALUE
    "|          |            |      |       |         |
      -    "     ".
    02 LINE 21 COLUMN  9 VALUE
    "|          |            |      |       |         |
      -    "     ".
    02 LINE 22 COLUMN  9 VALUE
    "
      -    "      ".
01 EKRAN3.
    02 LINE 15 COLUMN 13 VALUE
    "                                                  ".
```

```
    02 LINE 16 COLUMN 13 VALUE
    "|                |                                                    |".
    02 LINE 17 COLUMN 13 VALUE
    "|----------------|----------------------------------------------------|".
    02 LINE 18 COLUMN 13 VALUE
    "|                |                                                    |".
    02 LINE 19 COLUMN 13 VALUE
    "|                |                                                    |".
    02 LINE 20 COLUMN 13 VALUE
    "|                |                                                    |".
    02 LINE 21 COLUMN 13 VALUE
    "|                |                                                    |".
    02 LINE 22 COLUMN 13 VALUE
    "|----------------|----------------------------------------------------|".
01 EKRAN4.
    02 LINE  8 COLUMN 22 HIGHLIGHT  VALUE
    "╔═══════════════════════════════════════╗    ".
    02 LINE  9 COLUMN 22 HIGHLIGHT  VALUE
    "║           LISTELEME MENUSU            ║    ".
    02 LINE 10 COLUMN 22 HIGHLIGHT  VALUE
    "║                                       ║    ".
    02 LINE 11 COLUMN 22 HIGHLIGHT  VALUE
    "║ F1==>".
    02 LINE 11 COLUMN 30 VALUE
    "TARIH BAZINDA MUSTERI DEPARTMAN".
    02 LINE 11 COLUMN 61 HIGHLIGHT  VALUE
    "║".
    02 LINE 11 COLUMN 63 HIGHLIGHT  VALUE
    " ".
    02 LINE 12 COLUMN 22 HIGHLIGHT  VALUE
    "║ F2==>".
    02 LINE 12 COLUMN 30 VALUE
    "MUSTERI BAZINDA DEPARTMAN TARIH".
    02 LINE 12 COLUMN 61 HIGHLIGHT  VALUE
    "║".
    02 LINE 12 COLUMN 63 HIGHLIGHT  VALUE
    "  ".
    02 LINE 13 COLUMN 22 HIGHLIGHT  VALUE
    "║ F3==>".
    02 LINE 13 COLUMN 30 VALUE
    "DEPARTMAN BAZINDA TARIH MUSTERI".
    02 LINE 13 COLUMN 61 HIGHLIGHT  VALUE
    "║".
    02 LINE 13 COLUMN 63 HIGHLIGHT  VALUE
    "  ".
    02 LINE 14 COLUMN 22 HIGHLIGHT  VALUE
    "║ F4==>".
    02 LINE 14 COLUMN 30 VALUE
    "MEVCUT KAYITLAR".
    02 LINE 14 COLUMN 61 HIGHLIGHT  VALUE
    "║".
    02 LINE 14 COLUMN 63 HIGHLIGHT  VALUE
    "  ".
    02 LINE 15 COLUMN 22 HIGHLIGHT  VALUE
    "║ F5==>".
    02 LINE 15 COLUMN 30 VALUE
```

```
    "STOK DURUMU".
    02 LINE 15 COLUMN 61 HIGHLIGHT  VALUE
    "╢".
    02 LINE 15 COLUMN 63 HIGHLIGHT  VALUE
    "  ".
    02 LINE 16 COLUMN 22 HIGHLIGHT  VALUE
    "╢ F6==>".
    02 LINE 16 COLUMN 30 VALUE
    "ANA MENU".
    02 LINE 16 COLUMN 39 HIGHLIGHT  VALUE
    "              ".
    02 LINE 16 COLUMN 51 HIGHLIGHT  BLINK  VALUE
    "   ".
    02 LINE 16 COLUMN 54 HIGHLIGHT  VALUE
    "       ╢   ".
    02 LINE 17 COLUMN 22 HIGHLIGHT  VALUE
    "╢                  SECIMINIZ".
    02 LINE 17 COLUMN 51 HIGHLIGHT  BLINK  VALUE
    "==>".
    02 LINE 17 COLUMN 54 HIGHLIGHT  VALUE
    "       ╢   ".
    02 LINE 18 COLUMN 22 HIGHLIGHT  VALUE
    "╚═══════════════════════════════════════╝    ".

01 EKRAN5.
    02 LINE  9 COLUMN 32 REVERSE-VIDEO  VALUE
    " LISTELEME YAPMAK ICIN ".
    02 LINE 10 COLUMN 20 VALUE
    "╔═══════════════════════════════════════════╗".
    02 LINE 11 COLUMN 20 VALUE
    "╢                                           ╢".
    02 LINE 12 COLUMN 20 VALUE
    "╢ILK            KODUNU GIRINIZ==>          ╢".
    02 LINE 13 COLUMN 20 VALUE
    "╢                                           ╢".
    02 LINE 14 COLUMN 20 VALUE
    "╢SON            KODUNU GIRINIZ==>          ╢".
    02 LINE 15 COLUMN 20 VALUE
    "╢                                           ╢".
    02 LINE 16 COLUMN 20 VALUE
    "╚═══════════════════════════════════════════╝".
01 EKRAN6.
     02 LINE  4 COLUMN 31 VALUE
    "GIREN/CIKAN".
    02 LINE  5 COLUMN  2 VALUE
    "╔═══════╦═══════════╦═══════╦═══════════╦═══════════
      -     "╔═══════════╗".
    02 LINE  6 COLUMN  2 VALUE
    "╢STOK KODU ╢STOK ADI     ╢MIKTAR ╢  TUTAR     ╢MEVCUT MIKTAR
      -     "╢MEVCUT TUTAR   ╢".
    02 LINE  7 COLUMN  2 VALUE
    "╠═══════╬═══════════╬═══════╬═══════════╬═══════════
      -     "╠═══════════╣".
    02 LINE  8 COLUMN  2 VALUE
    "╢       ╢           ╢       ╢           ╢
      -     "╢           ╢".
```

```
    02 LINE  9 COLUMN  2 VALUE
    "‖          ‖                ‖        ‖            ‖
      -     "‖                   ‖".
    02 LINE 10 COLUMN  2 VALUE
    "‖          ‖                ‖        ‖            ‖
      -     "‖                   ‖".
    02 LINE 11 COLUMN  2 VALUE
    "‖          ‖                ‖        ‖            ‖
      -     "‖                   ‖".
    02 LINE 12 COLUMN  2 VALUE
    "‖          ‖                ‖        ‖            ‖
      -     "‖                   ‖".
    02 LINE 13 COLUMN  2 VALUE
    "‖          ‖                ‖        ‖            ‖
      -     "‖                   ‖".
    02 LINE 14 COLUMN  2 VALUE
    "‖          ‖                ‖        ‖            ‖
      -     "‖                   ‖".
    02 LINE 15 COLUMN  2 VALUE
    "‖          ‖                ‖        ‖            ‖
      -     "‖                   ‖".
    02 LINE 16 COLUMN  2 VALUE
    "‖          ‖                ‖        ‖            ‖
      -     "‖                   ‖".
    02 LINE 17 COLUMN  2 VALUE
    "‖          ‖                ‖        ‖            ‖
      -     "‖                   ‖".
    02 LINE 18 COLUMN  2 VALUE
    "‖          ‖                ‖        ‖            ‖
      -     "‖                   ‖".
    02 LINE 19 COLUMN  2 VALUE
    "╚══════════╩════════════════╩════════╩════════════╩
      -     "╩═══════════════════╝".
    02 LINE 20 COLUMN  3 VALUE
    "(+ )GIREN MIKTAR (-) CIKAN MIKTAR    (+)SATIMDA (-)ALIMDA ELD
      -     "E EDILEN TUTAR".
01 EKRAN7.
    02 LINE 11 COLUMN 18 VALUE
    "╔═══════════════════════════════════════════╗".
    02 LINE 12 COLUMN 18 VALUE
    "‖      HANGI TARIHLER ARASINDAKI KAYITLAR   ‖".
    02 LINE 13 COLUMN 18 VALUE
    "‖ ILK TARIH ==>  /  /                       ‖".
    02 LINE 14 COLUMN 18 VALUE
    "‖ SON TARIH ==>  /  /                       ‖".
    02 LINE 15 COLUMN 18 VALUE
    "‖                                           ‖".
    02 LINE 16 COLUMN 18 VALUE
    "╚═══════════════════════════════════════════╝".

01 EKRAN8.
     02 LINE  5 COLUMN  2 VALUE
    "╔══════════╦════════════════╦═══════════════════╦
      -     "═══════════╗".
    02 LINE  6 COLUMN  2 VALUE
    "‖Siparis No‖ Tarih    ‖MUSTERI (FIRMA)      ‖ DEPARTMAN
```

```
     -     "      ║TUTAR       ║".
    02 LINE  7 COLUMN  2 VALUE
    "╟──────────╫──────────╫────────────────────╫────────────────
     -     "═════╫══════════╢".
    02 LINE  8 COLUMN  2 VALUE
    "║          ║          ║                    ║
     -     "     ║          ║".
    02 LINE  9 COLUMN  2 VALUE
    "║          ║          ║                    ║
     -     "     ║          ║".
    02 LINE 10 COLUMN  2 VALUE
    "║          ║          ║                    ║
     -     "     ║          ║".
    02 LINE 11 COLUMN  2 VALUE
    "║          ║          ║                    ║
     -     "     ║          ║".
    02 LINE 12 COLUMN  2 VALUE
    "║          ║          ║                    ║
     -     "     ║          ║".
    02 LINE 13 COLUMN  2 VALUE
    "║          ║          ║                    ║
     -     "     ║          ║".
    02 LINE 14 COLUMN  2 VALUE
    "║          ║          ║                    ║
     -     "     ║          ║".
    02 LINE 15 COLUMN  2 VALUE
    "║          ║          ║                    ║
     -     "     ║          ║".
    02 LINE 16 COLUMN  2 VALUE
    "║          ║          ║                    ║
     -     "     ║          ║".
    02 LINE 17 COLUMN  2 VALUE
    "║          ║          ║                    ║
     -     "     ║          ║".
    02 LINE 18 COLUMN  2 VALUE
    "║          ║          ║                    ║
     -     "     ║          ║".
    02 LINE 19 COLUMN  2 VALUE
    "╙──────────╨──────────╨────────────────────╨────────────────
     -     "═════╨══════════╜".
    02 LINE 20 COLUMN 55 VALUE
    "TOPLAM==>".
01 EKRAN9.
     02 LINE 10 COLUMN 27 HIGHLIGHT  VALUE
    "╓───────────────────────────╖".
    02 LINE 11 COLUMN 27 HIGHLIGHT  VALUE
    "║ F1==>".
    02 LINE 11 COLUMN 34 VALUE
    "EKRANA LISTELEME".
    02 LINE 11 COLUMN 50 HIGHLIGHT  VALUE
    "    ║".
    02 LINE 12 COLUMN 27 HIGHLIGHT  VALUE
    "║ F2==>".
    02 LINE 12 COLUMN 34 VALUE
    "KAGIDA LISTELEME".
    02 LINE 12 COLUMN 50 HIGHLIGHT  VALUE
```

```
    "   ‖".
    02 LINE 13 COLUMN 27 HIGHLIGHT  VALUE
    "‖         SECIMINIZ".
    02 LINE 13 COLUMN 46 HIGHLIGHT  BLINK  VALUE
    "==>".
    02 LINE 13 COLUMN 49 HIGHLIGHT  VALUE
    "    ‖".
    02 LINE 14 COLUMN 27 HIGHLIGHT  VALUE
    "╚═══════════════════════════╝".


PROCEDURE DIVISION.
BASLAMA.
    OPEN I-O DOSYA.
    IF DURUM = "30"
    OPEN OUTPUT DOSYA
    CLOSE DOSYA OPEN I-O DOSYA.
    OPEN I-O DOSYA2.
    IF DURUM2 = "30"  OPEN OUTPUT DOSYA2
    CLOSE DOSYA2 OPEN I-O DOSYA2.
    OPEN I-O DOSYA1.
    IF DURUM1 = "30"  OPEN OUTPUT DOSYA1
    CLOSE DOSYA1 OPEN I-O DOSYA1.
    OPEN I-O DOSYA3.
    IF DURUM3 = "30"  OPEN OUTPUT DOSYA3
    CLOSE DOSYA3 OPEN I-O DOSYA3.
BASLA.
    DISPLAY (1 1) ERASE.
    DISPLAY ANA-MENU.
    ACCEPT (15 49) SEC WITH BEEP AUTO-SKIP.
    ACCEPT F FROM ESCAPE KEY.
    IF F1 GO KYARAT.
    IF F2 GO KIPTAL.
    IF F3 GO DEG1.
    IF F4 GO LISTE.
    IF F5 GO KYSON.
    GO BASLA.
      *************   KAYIT GIRISI YAPILIYOR  ************************
KYARAT.
    MOVE 0 TO J L K TUT TUTAR SAYAC SAYAC1 K MEV-TUT
    MEV-MIK TOPTUT.
GIRIS.
    DISPLAY (1 1) ERASE.
    DISPLAY EKRAN1.
    ACCEPT (5 22) SNO WITH BEEP AUTO-SKIP.
    IF SNO = 0
    GO BASLA.
    READ DOSYA KEY IS SNO INVALID KEY GO OK1.
    ACCEPT F FROM ESCAPE  KEY.
    IF ESC GO BASLA.
    IF F9 GO BASLA.
    GO DEG11.
OK1.
    DISPLAY EKRAN3.
    DISPLAY (16 16) "DEP. Kodu".
    DISPLAY (16 37) "DEPARTMAN  ADI".
```

```
    MOVE 18 TO LIN.
OKU1.
    READ DOSYA2 NEXT RECORD AT END
    ACCEPT (24 50) SEC WITH BEEP AUTO-SKIP GO PR1.
    DISPLAY (LIN, 16) DKOD1.
    DISPLAY (LIN, 28) DAD1.
    ADD 1 TO LIN
    IF LIN > 21 DISPLAY (24 10) "GORDUKTEN SONRA ENTER'A BASINIZ"
    ACCEPT ( , ) SEC WITH BEEP AUTO-SKIP
    MOVE 18 TO LIN.
    GO OKU1.
PR1.
    ACCEPT (7 28) DKOD1 WITH BEEP AUTO-SKIP.
    IF DKOD1 = 0 GO PR1.
    READ DOSYA2 KEY IS DKOD1 INVALID KEY GO PR2.
    DISPLAY (7 28) DKOD1.
    DISPLAY (7 48) DAD1.
    ACCEPT F FROM ESCAPE  KEY.
    IF ESC GO BASLA.
    IF F9 GO GIRIS.
    GO DONG2.
PR2.
    ACCEPT (7 48) DAD1 WITH BEEP AUTO-SKIP.
    ACCEPT F FROM ESCAPE  KEY.
    IF ESC GO  BASLA.
    IF F9 GO PR1.
DONG2.
    WRITE KAYIT2 INVALID KEY REWRITE KAYIT2.
    MOVE DKOD1 TO DKOD.
    MOVE DAD1 TO DAD.
DONG3.
    DISPLAY EKRAN3.
    DISPLAY (16 14) "MUSTERI Kodu".
    DISPLAY (16 37) "FIRMA   ADI".
    MOVE 18 TO LIN.
YAZ2.
    READ DOSYA3 NEXT RECORD AT END
    ACCEPT (24 50) SEC WITH BEEP AUTO-SKIP GO PR3.
    DISPLAY (LIN, 16) MKOD1.
    DISPLAY (LIN, 28) FIRMA1.
    ADD 1 TO LIN
    IF LIN > 21 DISPLAY (24 10) "GORDUKTEN SONRA ENTER'A BASINIZ"
    ACCEPT ( , ) SEC WITH BEEP AUTO-SKIP

    MOVE 18 TO LIN.
    GO YAZ2.
PR3.
    ACCEPT (9 28) MKOD1 WITH BEEP AUTO-SKIP.
    IF MKOD1 = 0 GO PR3.
    READ DOSYA3 KEY IS MKOD1 INVALID KEY GO PR4.
    DISPLAY (9 48) FIRMA1.
    ACCEPT F FROM ESCAPE  KEY.
    IF ESC GO BASLA.
    IF F9 GO PR1.
    GO PR5.
PR4.
```

```
    ACCEPT (9 48) FIRMA1 WITH BEEP AUTO-SKIP.
    ACCEPT F FROM ESCAPE  KEY.
    IF ESC GO BASLA.
    IF F9 GO PR3.
PR5.
    ACCEPT (10 28) ILGILI1 WITH BEEP AUTO-SKIP.
    ACCEPT F FROM ESCAPE  KEY.
    IF ESC GO BASLA.
    IF F9 GO PR4.
PR6.
    ACCEPT (11 28) ADRES1 WITH BEEP AUTO-SKIP.
    ACCEPT F FROM ESCAPE  KEY.
    IF ESC GO BASLA.
    IF F9 GO PR5.
DONG4.
    WRITE KAYIT3 INVALID KEY REWRITE KAYIT3.
    MOVE MKOD1 TO MKOD.
    MOVE ADRES1 TO ADRES.
    MOVE FIRMA1 TO FIRMA.
    MOVE ILGILI1 TO ILGILI.
PR7.
    ACCEPT (13 18) GUN WITH BEEP AUTO-SKIP.
    IF GUN > 31 OR = 0 GO PR7.
    ACCEPT F FROM ESCAPE  KEY.
    IF ESC GO BASLA.
    IF F9 GO PR6.
PR8.
    ACCEPT (13 22) AY WITH BEEP AUTO-SKIP.
    IF AY > 12 OR = 0 GO PR8.
    ACCEPT F FROM ESCAPE  KEY.
    IF ESC GO BASLA.
    IF F9 GO PR7.
PR9.
    ACCEPT (13 26) YIL WITH BEEP AUTO-SKIP.
    IF YIL = 0 GO PR9.
    ACCEPT F FROM ESCAPE  KEY.
    IF ESC GO BASLA.
    IF F9 GO PR8.
PR10.
    ACCEPT (13 41) ACIK WITH BEEP AUTO-SKIP.
    ACCEPT F FROM ESCAPE KEY.
    IF ESC GO BASLA.
    IF F9 GO PR9.
PR11.
    ACCEPT (13 71) AL WITH BEEP AUTO-SKIP.
    IF AL = "A" GO D11.
    IF AL = "S" GO D11.
    GO PR11.
D11.
    ACCEPT F FROM ESCAPE KEY.
    IF ESC GO BASLA.
    IF F9 GO PR10.
KYARAT2.
    DISPLAY EKRAN3.
    DISPLAY (16 14) "STOK Kodu".
    DISPLAY (16 40) "STOK ADI".
```

```
    MOVE 18 TO LIN.
OK2.
    READ DOSYA1 NEXT RECORD AT END
    ACCEPT (24 50) SEC WITH BEEP AUTO-SKIP GO STGIR.
    DISPLAY (LIN, 16) SKOD.
    DISPLAY (LIN, 28) SAD.
    ADD 1 TO LIN
    IF LIN > 21 DISPLAY (24 10) "GORDUKTEN SONRA ENTER'A BASINIZ"
    ACCEPT ( , ) SEC WITH BEEP AUTO-SKIP
    MOVE 18 TO LIN.
    GO OK2.
STGIR.
    DISPLAY EKRAN2.
    MOVE 18 TO LIN.
SR1.
    ACCEPT (LIN, 10) SKOD WITH BEEP AUTO-SKIP.
    IF SKOD = 0 GO SR1.
    READ DOSYA1 KEY IS SKOD INVALID KEY GO SR2.
    DISPLAY (LIN, 21) SAD.
    DISPLAY (LIN, 34) BIRIM.
    ACCEPT F FROM ESCAPE  KEY.
    IF ESC GO BASLA.
    IF F9 GO PR11.
    GO SR4.
SR2.
    ACCEPT (LIN, 21) SAD WITH BEEP AUTO-SKIP.
    DISPLAY (24 3) "MEVCUT STOK MIKTARI==>              ".
    ACCEPT (24 26) MEV-MIK WITH BEEP AUTO-SKIP.
    DISPLAY (24 3) ERASE.
    ACCEPT F FROM ESCAPE  KEY.
    IF ESC GO BASLA.
    IF F9 GO SR1.
SR3.
    ACCEPT (LIN, 34) BIRIM WITH BEEP AUTO-SKIP.
    ACCEPT F FROM ESCAPE  KEY.
    IF ESC GO BASLA.
    IF F9 GO SR2.
SR4.
    ACCEPT (LIN, 42) BIR-MI WITH BEEP AUTO-SKIP.
    ACCEPT F FROM ESCAPE  KEY.
    IF ESC GO BASLA.
    IF F9 GO SR3.
SR5.
    ACCEPT (LIN, 50) BIR-FI WITH BEEP AUTO-SKIP.
    ACCEPT F FROM ESCAPE  KEY.
    IF ESC GO BASLA.
    IF F9 GO SR4.
    COMPUTE TUT = BIR-MI * BIR-FI.
    MOVE TUT TO TUT1.
    DISPLAY (LIN, 60) TUT1.
    COMPUTE TUTAR = TUTAR + TUT.
    MOVE TUTAR TO TUTAR1.
    DISPLAY (5 52) TUTAR1.
HESAP.
    IF AL = "A" GO TOPLA.
    IF AL = "S" GO CIKAR.
```

```
    GO HESAP.
TOPLA.
    COMPUTE MEV-TUT = MEV-TUT - TUT.
    COMPUTE MEV-MIK = MEV-MIK + BIR-MI.
    MOVE 0 TO ALS.
    GO KYAP2.
CIKAR.
    COMPUTE MEV-TUT = MEV-TUT + TUT.
    COMPUTE MEV-MIK = MEV-MIK - BIR-MI.
    MOVE 1 TO ALS.
KYAP2.
    ADD 1 TO J.
    WRITE KAYIT1 INVALID KEY REWRITE KAYIT1.
                MOVE SKOD TO SKOD1(J).
                MOVE SAD TO SAD1(J).
                MOVE BIRIM TO BIRIM1(J).
                MOVE BIR-FI TO BIR-FI1(J).
                MOVE BIR-MI TO BIR-MI1(J).
                MOVE TUT TO TUT11(J).
    MOVE J TO ART.
    DISPLAY (24 8) "BASKA SIPARIS VAR MI? [E/H]==>[ ]"
    ACCEPT (24 39) CEVAP WITH BEEP AUTO-SKIP.
    IF NOT CEV WRITE KAYIT
    GO BASLA.
    IF LIN > 20 GO STGIR.
    ADD 1 TO LIN GO SR1.
KIPTAL.
    DISPLAY (1 1) ERASE.
    DISPLAY (12 10) "SILINECEK KAYIT ICIN KAYIT NO GIRINIZ==>"
    ACCEPT ( , ) SNO WITH BEEP AUTO-SKIP.
    IF SNO = 0 GO BASLA.
    READ DOSYA KEY IS SNO INVALID KEY GO HATA111.
    PERFORM DEG11.
    DISPLAY (24 8) "SILMEK ISTEDIGINIZ KAYIT BU MU?"
    " [E/H]  [ ]"
    ACCEPT (24 48) CEVAP WITH BEEP AUTO-SKIP.
    IF NOT CEV
    GO BASLA.
    DELETE DOSYA RECORD INVALID KEY GO HATA111.
    MOVE ZEROS TO SNO AY YIL GUN.
    MOVE SPACES TO DKOD DAD MKOD ACIK FIRMA AL ILGILI ADRES.
    PERFORM SIL VARYING I FROM 1 BY 1 UNTIL I > ART.
    MOVE ZEROS TO TUTAR ART.
    GO SIL1.
SIL.
    MOVE SPACES TO SKOD1(I) SAD1(I) BIRIM1(I).
    MOVE ZEROS TO BIR-MI1(I) BIR-FI1(I) TUT11(I).
SIL1.
    DISPLAY (24 8) "KAYIT SILINDI!!!.. SILMEK ISTEDIGINIZ BASKA"
    " KAYIT VAR MI? [E/H] [ ]"
    ACCEPT (24 73) CEVAP WITH BEEP AUTO-SKIP IF NOT CEV
    GO BASLA.
    GO KIPTAL.
DEG1.
    DISPLAY (1 1) ERASE.
    DISPLAY (12 10) "DUZELTILECEK KAYIT ICIN KAYIT NO GIRINIZ==>"
```

```
    ACCEPT ( , ) SNO WITH BEEP AUTO-SKIP.
    READ DOSYA KEY IS SNO INVALID KEY GO HATA1.
DEG11.
    DISPLAY EKRAN1.
    DISPLAY (2 7) "                                                    "
    "                    ".
    DISPLAY (5 22) SNO
            (7 28) DKOD
            (7 48) DAD
            (9 28) MKOD
            (10 28) ILGILI
            (11 28) ADRES
            (9 48) FIRMA
            (13 18) GUN
            (13 22) AY
            (13 26) YIL
            (13 41) ACIK
            (13 71) AL.
    DISPLAY EKRAN2.
    MOVE 18 TO LIN.
    PERFORM SYAZ VARYING I FROM 1 BY 1 UNTIL I > ART.
    MOVE TUTAR TO TUTAR1
    DISPLAY (5 52) TUTAR1.
DEG22.
    GO DEG2.
SYAZ.
    DISPLAY (LIN, 10) SKOD1(I)
    DISPLAY (LIN, 21) SAD1(I).
    DISPLAY (LIN, 34) BIRIM1(I).
    MOVE BIR-MI1(I) TO BIR-MI11.
    DISPLAY (LIN, 42) BIR-MI11.
    MOVE BIR-FI1(I) TO BIR-FI11.
    DISPLAY (LIN, 49) BIR-FI11.
    MOVE TUT11(I) TO TUT1.
    DISPLAY (LIN, 60) TUT1.
    IF LIN > 21 DISPLAY (24 2) "          DEVAM ICIN ENTER
           "
    ACCEPT ( , ) SEC WITH BEEP AUTO-SKIP DISPLAY EKRAN2
    MOVE 17 TO LIN.
    ADD 1 TO LIN.
DEG2.
    DISPLAY (24 8) "DEGISTIRMEK ISTEDIGINIZ KAYIT BU MU?"
    " [E/H]  [ ]"
    ACCEPT (24 53) CEVAP WITH BEEP AUTO-SKIP.
    IF NOT CEV
    GO BASLA.
    DISPLAY (24 8) "DEGISIKLIK YAPILMAYACAK BILGI ICIN"
    " SADECE ENTER".
DEG3.
    ACCEPT (7 28) DKOD WITH UPDATE.
    ACCEPT (7 48) DAD WITH UPDATE.
    MOVE DKOD TO DKOD1.
    MOVE DAD TO DAD1.
    WRITE KAYIT2 INVALID KEY REWRITE KAYIT2.
    ACCEPT (9 28) MKOD WITH UPDATE.
    ACCEPT  (9 48) FIRMA WITH UPDATE.
```

```
    ACCEPT (10 28) ILGILI WITH UPDATE.
    ACCEPT  (11 28) ADRES WITH UPDATE.
    MOVE MKOD TO MKOD1.
    MOVE FIRMA TO FIRMA1.
    MOVE ILGILI TO ILGILI1.
    MOVE ADRES TO ADRES1.
    WRITE KAYIT3 INVALID KEY REWRITE KAYIT3.
    ACCEPT  (13 18) GUN  WITH UPDATE.
    ACCEPT  (13 22) AY  WITH UPDATE.
    ACCEPT (13 26) YIL WITH UPDATE.
    ACCEPT (13 41) ACIK WITH UPDATE.
    ACCEPT (13 71) AL WITH UPDATE.
    MOVE DKOD1 TO DKOD.
    MOVE DAD1 TO DAD.
    MOVE MKOD1 TO MKOD.
    MOVE FIRMA1 TO FIRMA.
    MOVE ILGILI1 TO ILGILI.
    MOVE ADRES1 TO ADRES.
    DISPLAY EKRAN2.
    MOVE 18 TO LIN.
    MOVE 0 TO TUTAR.
    PERFORM SYAZ1 VARYING I FROM 1 BY 1 UNTIL I > ART.
    MOVE TUTAR TO TUTAR1.
    DISPLAY (5 52) TUTAR1 GO DEG4.
SYAZ1.
    ACCEPT (LIN, 10) SKOD1(I) WITH UPDATE.
    ACCEPT (LIN, 21) SAD1(I) WITH UPDATE.
    ACCEPT (LIN, 34) BIRIM1(I) WITH UPDATE.
    ACCEPT (LIN, 42) BIR-MI1(I) WITH UPDATE.
    ACCEPT (LIN, 51) BIR-FI1(I) WITH UPDATE.
    COMPUTE TUT11(I) = BIR-MI1(I) * BIR-FI1(I).
    MOVE TUT11(I) TO TUT1
    DISPLAY (LIN, 60) TUT1.
    COMPUTE TUTAR = TUTAR + TUT11(I).
    MOVE SKOD1(I) TO SKOD.
    MOVE SAD1(I) TO SAD.
    MOVE BIRIM1(I) TO BIRIM.
    MOVE BIR-MI1(I) TO BIR-MI.
    MOVE BIR-FI1(I) TO BIR-FI.
    IF AL = "A"
    COMPUTE MEV-TUT = MEV-TUT - TUT
    COMPUTE MEV-MIK = MEV-MIK + BIR-MI
    WRITE KAYIT1 INVALID KEY REWRITE KAYIT1
    MOVE SKOD TO SKOD1(I)
    MOVE SAD TO SAD1(I)
    MOVE BIRIM TO BIRIM1(I)
    MOVE BIR-MI TO BIR-MI1(I)
    MOVE BIR-FI TO BIR-FI1(I)
          ELSE COMPUTE MEV-TUT = MEV-TUT + TUT
          COMPUTE MEV-MIK = MEV-MIK - BIR-MI
          WRITE KAYIT1 INVALID KEY REWRITE KAYIT1
          MOVE SKOD TO SKOD1(I)
          MOVE SAD TO SAD1(I)
          MOVE BIRIM TO BIRIM1(I)
          MOVE BIR-MI TO BIR-MI1(I)
          MOVE BIR-FI TO BIR-FI1(I).
```

```
    IF LIN > 21
    DISPLAY (24 10) "           DEVAM ICIN ENTER           "
    "       "
    ACCEPT ( , ) SEC WITH BEEP AUTO-SKIP DISPLAY EKRAN2
    MOVE 17 TO LIN.
    ADD 1 TO LIN.
DEG4.
    REWRITE KAYIT.
    DISPLAY (24 8) "    DEGISTIRMEK   ISTEDIGINIZ  BASKA KAYIT"
    " VAR MI? [E/H] [ ]"
    ACCEPT (24 66) CEVAP WITH BEEP AUTO-SKIP
    IF NOT CEV
    GO BASLA.
    GO DEG1.
LISTE.
    DISPLAY (1 1) ERASE.
    DISPLAY EKRAN9.
    ACCEPT (13 49) SEC WITH BEEP AUTO-SKIP.
    ACCEPT F FROM ESCAPE KEY.
    IF F1 GO LIST.
    IF F2 GO KLIST.
    GO LISTE.
LIST.
    DISPLAY (1 1) ERASE.
    DISPLAY EKRAN4.
    ACCEPT (17 55) SEC WITH BEEP AUTO-SKIP.
    ACCEPT F FROM ESCAPE KEY.
    IF F1 GO LIST1.
    IF F2 GO LIST2.
    IF F3 GO LIST3.
    IF F4 GO LIST4.
    IF F5 GO LIST5.
    IF F6 GO BASLA.
    GO LIST.
LIST1.
    DISPLAY (1 1) ERASE.
    DISPLAY EKRAN7.
LL1.
    ACCEPT (13 33) IGUN1.
    IF IGUN1 > 31 OR = 0 GO LL1.
LL2.
    ACCEPT (13 36) IAY1.
    IF IAY1 > 12 OR = 0 GO LL2.
LL3.
    ACCEPT (13 39) IYIL1.
    IF IYIL1 = 0 GO LL3.
LL4.
    ACCEPT (14 33) SGUN1.
    IF SGUN1 > 31 OR = 0 GO LL4.
LL5.
    ACCEPT (14 36) SAY1.
    IF SAY1 > 12 OR = 0 GO LL5.
LL6.
    ACCEPT (14 39) SYIL1.
    IF SYIL1 = 0 GO LL6.
    IF SYIL1 < IYIL1 GO LL4.
```

```
    IF SYIL1 = IYIL1 AND SAY1 < IAY1 GO LL4.
    IF SYIL1 = IYIL1 AND SAY1 = IAY1 AND SGUN1 < IGUN1 GO LL4.
TRLIST1.
    MOVE ZEROS TO TOPTUT.
    MOVE ZEROS TO TAR.
    MOVE W-TAR TO TAR
    START DOSYA KEY IS NOT LESS TAR
    INVALID KEY GO HATA-1.
    READ DOSYA NEXT RECORD AT END GO TOPLAM1.
    IF IGUN1 = GUN AND
    IAY1 = AY   AND
    IYIL1 = YIL GO TR1.
    GO HATA-1.
TR1.
    PERFORM TRH
    PERFORM TRH2
    PERFORM TRH1
    PERFORM TRLIST3
    MOVE TAR TO W-TAR.
    GO TRLIST2.
    GO HATA-1.
TRH2.
    DISPLAY (3 10) "ILK TARIH==>" IGUN1 "/" IAY1
     "/" IYIL1 (3 48) "SON TARIH==>" SGUN1 "/" SAY1 "/" SYIL1.
TRH3.
    DISPLAY (3 10) "ILK KOD==>" MUS-1
            (3 48) "SON KOD==>" MUS-2.
TRH4.
    DISPLAY (3 10) "ILK KOD==>" DEP-1
            (3 48) "SON KOD==>" DEP-2.
TRH5.
    DISPLAY (3 10) "ILK SIPARIS NUMARASI==>" SNO
            (3 48) "SON SIPARIS NUMARASI==>" SNO-1.
TRH1.
    MOVE 7 TO LIN.
TRLIST2.
    READ DOSYA NEXT RECORD AT END GO TOPLAM1.
    IF IGUN1 = GUN  AND
    IAY1 = AY  AND
    IYIL1 = YIL
    OR YIL LESS THAN SYIL1 OR = SYIL1 GO TRL1.
    GO TRLIST2.
TRL1.
    IF AY LESS THAN SAY1 OR = SAY1 GO TRL2.
    GO TRLIST2.
TRL2.
    IF GUN LESS THAN SGUN1 OR = SGUN1
    PERFORM TRLIST3.
    GO TRLIST2.
TRH.
    DISPLAY (1 1) ERASE.
    DISPLAY EKRAN8.
TRLIST3.
    ADD  1 TO LIN.
    DISPLAY (LIN, 3) SNO
            (LIN, 14) GUN "/" AY "/" YIL
```

```
            (LIN, 25) FIRMA
            (LIN, 47) DAD
            (LIN, 68) TUTAR.
    COMPUTE TOPTUT = TOPTUT + TUTAR.
    IF LIN > 18 DISPLAY (20 10) "DEVAM ICIN ENTER"
    ACCEPT ( , ) SEC WITH BEEP AUTO-SKIP PERFORM TRH PERFORM TRH2
    PERFORM TRH1.
TOPLAM1.
    MOVE TOPTUT TO TOPTUT1
    DISPLAY (20 65) TOPTUT1.
    ACCEPT ( , ) SEC WITH BEEP AUTO-SKIP.
DEV.
    DISPLAY (22 10) "BASKA TARIHLER ARASINI LISTELEMEK ISTER MI"
    "SINIZ [E/H]  [ ]"
    ACCEPT (22 66) CEVAP WITH BEEP AUTO-SKIP IF NOT CEV
    GO LIST.
    GO LIST1.
LIST2.
    DISPLAY (1 1) ERASE.
    DISPLAY EKRAN5.
    DISPLAY (12 25) "MUSTERI".
    DISPLAY (14 25) "MUSTERI".
    ACCEPT (12 55) MUS-1.
LIST22.
    ACCEPT (14 55) MUS-2.
    IF MUS-2 < MUS-1 GO LIST22.
MSLIST1.
    MOVE SPACES TO MKOD.
    MOVE MUS-1 TO MKOD.
    MOVE ZEROS TO TOPTUT.
    START DOSYA KEY IS NOT LESS MKOD
    INVALID KEY GO HATA-2.
    READ DOSYA NEXT RECORD AT END PERFORM TOPLAM1 GO
    DEV1.
    MOVE MKOD TO MUS-1
    IF MUS-1 = MKOD OR MKOD LESS THAN MUS-2 OR = MUS-2
    PERFORM TRH
    PERFORM TRH3
    PERFORM TRH1
    PERFORM TRLIST3
    GO MSLIST2.
    GO HATA-2.
MSLIST2.
    READ DOSYA NEXT RECORD AT END PERFORM TOPLAM1 GO DEV1.
    IF MUS-1 = MKOD OR MKOD LESS THAN MUS-2 OR = MUS-2
    PERFORM TRLIST3.
    GO MSLIST2.
DEV1.
    DISPLAY (22 10) "MUSTERI BAZINDA BASKA LISTELEME YAPMAK IST"
    "ER MISINIZ [E/H]  [ ]"
    ACCEPT (22 71) CEVAP WITH BEEP AUTO-SKIP IF NOT CEV
    GO LIST.
    GO LIST2.
LIST3.
    DISPLAY (1 1) ERASE.
    DISPLAY EKRAN5.
```

```
    DISPLAY (12 25) "DEPARTMAN".
    DISPLAY (14 25) "DEPARTMAN".
    ACCEPT (12 55) DEP-1.
LIST33.
    ACCEPT (14 55) DEP-2.
    IF DEP-2 < DEP-1 GO LIST33.
DPLIST1.
    MOVE SPACES TO DKOD.
    MOVE DEP-1 TO DKOD.
    MOVE ZEROS TO TOPTUT.
    START DOSYA KEY IS NOT LESS DKOD
    INVALID KEY GO HATA-3.
    READ DOSYA NEXT RECORD AT END PERFORM TOPLAM1 GO
    DEV2.
    MOVE DKOD TO DEP-1
    IF DEP-1 = DKOD OR DKOD LESS THAN DEP-2 OR = DEP-2
    PERFORM TRH
    PERFORM TRH4
    PERFORM TRH1
    PERFORM TRLIST3
    GO DPLIST2.
    GO HATA-3.
DPLIST2.
    READ DOSYA NEXT RECORD AT END PERFORM TOPLAM1 GO DEV2.
    IF DEP-1 = DKOD OR DKOD LESS THAN DEP-2 OR = DEP-2
    PERFORM TRLIST3.
    GO DPLIST2.
DEV2.
    DISPLAY (22 10) "DEPARTMAN BAZINDA BASKA LISTELEME YAPMAK I"
    "STER MISINIZ [E/H]   [ ]"
    ACCEPT (22 73) CEVAP WITH BEEP AUTO-SKIP IF NOT CEV
    GO LIST.
    GO LIST3.
LIST5.
    DISPLAY (1 1) ERASE.
    DISPLAY EKRAN5.
    DISPLAY (12 25) "STOK".
    DISPLAY (14 25) "STOK".
    ACCEPT (12 55) SKOD.
    IF SKOD = " " GO LIST5.
LIST55.
    ACCEPT (14 55) STK-1.
    IF STK-1 < SKOD GO LIST55.
STKLIST1.
    START DOSYA1 KEY IS NOT LESS SKOD
    INVALID KEY GO HATA-11.
    DISPLAY EKRAN6.
    IF SKOD = STK-1 DISPLAY (3 10) SKOD "   KODLU KAYIT"
    MOVE 7 TO LIN GO STKLIST2.
    DISPLAY (3 10) SKOD "   ILE     " STK-1
    "   ARASINDAKI KAYITLAR"
    MOVE 7 TO LIN.
STKLIST2.
    READ DOSYA1 NEXT RECORD AT END GO DEV4.
    IF SKOD > STK-1 GO DEV4.
    MOVE MEV-TUT TO MEV-TUT1.
```

```
    MOVE MEV-MIK TO MEV-MIK1.
    MOVE BIR-MI TO BIR-MI11.
    MOVE TUT TO TUT1.
    ADD  1 TO LIN.
    DISPLAY (LIN, 3) SKOD
            (LIN, 14) SAD.
    IF ALS = 0 DISPLAY
            (LIN, 28) "+"  BIR-MI11
            (LIN, 36) "-"  TUT1
    ELSE DISPLAY (LIN, 28) "-" BIR-MI11
            (LIN, 36) "+" TUT1.
    DISPLAY (LIN, 51) MEV-MIK1
            (LIN, 63) MEV-TUT1.
    IF LIN > 18 DISPLAY (20 10) "DEVAM ICIN ENTER"
    ACCEPT ( , ) SEC WITH BEEP DISPLAY (1 1) ERASE
    DISPLAY EKRAN6 MOVE 7 TO LIN.
    GO STKLIST2.
DEV4.
    DISPLAY (22 10) "STOK BAZINDA BASKA LISTELEME YAPMAK ISTER"
    " MISINIZ [E/H]  [ ]"
    ACCEPT (22 68) CEVAP WITH BEEP AUTO-SKIP IF NOT CEV
    GO LIST.
    GO LIST5.
LIST4.
    DISPLAY (1 1) ERASE.
    DISPLAY EKRAN5.
    DISPLAY (12 25) " SIPARIS NO'SUNU GIRINIZ ".
    DISPLAY (14 25) " SIPARIS NO'SUNU GIRINIZ ".
    ACCEPT (12 55) SNO.
    IF SNO = 0 GO LIST4.
LIST44.
    ACCEPT (14 55) SNO-1.
    IF SNO-1 < SNO GO LIST44.
SNOLIST1.
    MOVE ZEROS TO TOPTUT.
    START DOSYA KEY IS NOT LESS SNO
    INVALID KEY PERFORM HATA1 GO HAT1.
    PERFORM TRH
    PERFORM TRH5
    PERFORM TRH1.
SNOLIST2.
    READ DOSYA NEXT RECORD AT END PERFORM TOPLAM1
    GO BASLA.
    IF SNO > SNO-1 PERFORM TOPLAM1
    GO BASLA.
    PERFORM TRLIST3.
    GO SNOLIST2.
KLIST.
    DISPLAY (1 1) ERASE.
    DISPLAY (12 10) "PRINTERINIZ ACIK MI? [E/H] [ ]"
    ACCEPT (12 38) CEVAP WITH BEEP AUTO-SKIP IF NOT CEV GO BASLA.
KLISTE.
    OPEN OUTPUT YAZICI.
KLIST1.
    DISPLAY (1 1) ERASE.
    DISPLAY EKRAN5.
```

```
    DISPLAY (12 25) " SIPARIS NO'SUNU GIRINIZ ".
    DISPLAY (14 25) " SIPARIS NO'SUNU GIRINIZ ".
    ACCEPT (12 55) SNO.
    IF SNO = " " GO KLIST1.
KLIST2.
    ACCEPT (14 55) SNO-1.
    IF SNO-1 = " " GO KLIST2.
KLIST3.
    MOVE ZEROS TO TOPTUT.
    START DOSYA KEY IS NOT LESS SNO
    INVALID KEY GO KHATA11.
    MOVE SNO TO K-SNO.
    MOVE SNO-1 TO K-SNO1.
    WRITE YAZICI-KAYDI FROM YAZ-555.
    WRITE YAZICI-KAYDI FROM YAZ-BOS.
    WRITE YAZICI-KAYDI FROM YAZ-333.
    WRITE YAZICI-KAYDI FROM YAZ-CIZ.
KLISTE2.
    READ DOSYA NEXT RECORD AT END GO KTOPLAM1.
    IF SNO > SNO-1 GO KTOPLAM1.
    MOVE SNO TO K-SNO2.
    MOVE DAD TO K-DAD
    MOVE FIRMA TO K-FIRMA
    MOVE GUN TO K-GUN
    MOVE AY TO K-AY
    MOVE YIL TO K-YIL
    MOVE AL TO K-AL
    COMPUTE TOPTUT = TOPTUT + TUTAR.
    MOVE TUTAR TO K-TUTAR
    WRITE YAZICI-KAYDI FROM KAYIT-YAZ.
    GO KLISTE2.
KTOPLAM1.
    MOVE TOPTUT TO K-GENEL.
    WRITE YAZICI-KAYDI FROM YAZ-CIZ.
    WRITE YAZICI-KAYDI FROM YAZ-444.
    WRITE YAZICI-KAYDI FROM YAZ-CIZ.
KTAMAM.
    DISPLAY (1 1) ERASE.
    DISPLAY (12 10) "KAYITLAR BITTI !!!"
    ACCEPT ( , ) CEVAP WITH BEEP AUTO-SKIP
    CLOSE YAZICI GO BASLA.
KYSON.
    DISPLAY (1 1) ERASE.
    CLOSE DOSYA DOSYA1 DOSYA2 DOSYA3.
    STOP  RUN.
HATA.
    DISPLAY (1 1) ERASE.
    DISPLAY (12 10) "DOSYAYA KAYIT YAZILAMIYOR"
    ACCEPT ( , ) SEC WITH BEEP AUTO-SKIP.
    GO BASLA.
HATA1.
    DISPLAY (1 1) ERASE.
    DISPLAY (12 10) "ARADIGINIZ NUMARAYLA ILGILI KAYIT"
    " MEVCUT DEGIL!!!"
    DISPLAY  (13 10) "YENIDEN DENEMEK ISTER MISINIZ?[E/H]  [ ]"
    ACCEPT (13 48) CEVAP WITH BEEP AUTO-SKIP.
```

```
HAT.
    IF NOT CEV GO BASLA.
    GO DEG1.
HAT1.
    IF NOT CEV GO LIST.
    GO LIST4.
HATA111.
    DISPLAY (1 1) ERASE.
    DISPLAY (12 10) "ARADIGINIZ NUMARAYLA ILGILI KAYIT"
    " MEVCUT DEGIL!!!"
    DISPLAY  (13 10) "YENIDEN DENEMEK ISTER MISINIZ?[E/H]  [ ]"
    ACCEPT (13 48) CEVAP WITH BEEP AUTO-SKIP.
    IF NOT CEV GO BASLA.
    GO KIPTAL.
HATA-1.
    DISPLAY (1 1) ERASE.
    DISPLAY (12 5) "GIRMIS OLDUGUNUZ BASLANGIC TARIHIYLE ILGILI"
    " KAYITLARIMIZ MEVCUT DEGIL!!!"
    DISPLAY  (13 15) "YENIDEN DENEMEK ISTER MISINIZ?[E/H]  [ ]"
    ACCEPT (13 53) CEVAP WITH BEEP AUTO-SKIP.
    IF NOT CEV GO LIST.
    GO LIST1.
HATA-2.
    DISPLAY (1 1) ERASE.
    DISPLAY (12 10) "ARADIGINIZ KODLA  ILGILI KAYIT MEVCUT"
    " DEGIL!!!"
    DISPLAY  (13 10) "YENIDEN DENEMEK ISTER MISINIZ?[E/H]  [ ]"
    ACCEPT (13 48) CEVAP WITH BEEP AUTO-SKIP.
    IF NOT CEV GO LIST.
    GO LIST2.
HATA-11.
    DISPLAY (1 1) ERASE.
    DISPLAY (12 10) "ARADIGINIZ KODLA  ILGILI KAYIT MEVCUT"
    " DEGIL!!!"
    DISPLAY  (13 10) "YENIDEN DENEMEK ISTER MISINIZ?[E/H]  [ ]"
    ACCEPT (13 48) CEVAP WITH BEEP AUTO-SKIP.
    IF NOT CEV GO LIST.
    GO LIST.
    GO LIST5.
KHATA11.
    DISPLAY (1 1) ERASE.
    DISPLAY (12 10) "ARADIGINIZ KODLA  ILGILI KAYIT MEVCUT"
    " DEGIL!!!"
    DISPLAY  (13 10) "YENIDEN DENEMEK ISTER MISINIZ?[E/H]  [ ]"
    ACCEPT (13 48) CEVAP WITH BEEP AUTO-SKIP.
    IF NOT CEV CLOSE YAZICI GO BASLA.
    CLOSE YAZICI GO KLIST.
HATA-3.
    DISPLAY (1 1) ERASE.
    DISPLAY (12 10) "ARADIGINIZ KODLA  ILGILI KAYIT MEVCUT"
    " DEGIL!!!"
    DISPLAY  (13 10) "YENIDEN DENEMEK ISTER MISINIZ?[E/H]  [ ]"
    ACCEPT (13 48) CEVAP WITH BEEP AUTO-SKIP.
    IF NOT CEV GO LIST.
    GO LIST3.
```

Dönüştürme programını kullanmadan önce yaratılacak veritabanının tasarımını yapalım:

VERİTABANI TASARIMLAMA

Kayıt İlişkileri Modelini Tabloyla Eşleştirme

- Basit kayıtları tablolara eşleştirme
- Alanları kolonlara eşleştirme ve örnek veriyi dökümanlaştırma. Kolonları anlaşılır şekilde isimlendirme ve veri tiplerini tanımlama.
- Teklik belirteçlerini (Unique Identifiers), birincil anahtarla (primary key) eşleştirme.
- İlişkileri yabancı anahtarlarla eşleştirme.

Ek Gereksinimler

- Tablonun satırlarına (rows) doğrudan veya çabuk erişimi sağlayan veritabanı objesi olan indeksleri tasarımlama.
- Bir veya birden fazla tablo üzerinde kurulu lojiksel tablo olan view'ları kurma.
- Bir tablonun veritabanında kaplayacağı yerin miktarı olan fiziksel depolama boşluğunu planlama.
- Bütünlük kısıtlarını (integrity constraint) yeniden tanımlama.

Adım 1:

Kayıtları Tablolara Eşleştirme

- Bir tablo örnek kartı yaratılmalı
  - Tablo ismi
  - Kolon ismi
  - Anahtar tipleri
  - Boş(null) ve teklik(unique) referansları
  - Yabancı anahtar bilgisi
  - Kolon veritipi ve maksimum uzunluğu
  - Örnek veri
- Tablo adı kaydedilmeli

Adım 2:

Alanları Kolonlara Eşleştirme

- Kayıt İlişki modelindeki her alan tablodaki bir kolon adıyla eşleştirilmeli.
- Zorunlu alanlar NN (not null- boş olmamalı) referansıyla belirtilmeli.

Adım 3:

Teklik Belirteçlerini Birincil Anahtarla Eşleştirme

- Kayıt İlişki modelinde (#) sembolüyle belirtilen teklik belirteçleri (UIDs), birincil anahtar kolonuyla eşleştirilip, anahtar tipine PK (primary key) denilmeli.

- Not null ve unique belirteçleri etiketlenmeli.

- Birçok alan içeren bir UID'i bir karma PK'ye eşlenip bu kolonlar NN (not null) ve U (unique) olarak etiketlenmeli. Eğer birden fazla alternatif birincil anahtar (PK) varsa, NN ve U olarak etiketlenmeli ancak bir tek PK seçilmeli.

Adım 4:
İlişkileri Yabancı Anahtarla Eşleştirme

Son adım ilişkileri yabancı anahtarla eşleştirmedir. Gözönüne alınacak iki tür ilişki vardır: *Çoka-bir* ve *Bire-bir ilişki*. Eğer ilişki birincil anahtarın bir parçasıysa, bu Adım 3'te eşleştirilmiştir. zorunlu alanların NN olmasına dikkat edilmelidir.Yabancı anahtar için tek bir isim verilmesine dikkat edilmelidir.

Çoka-Bir (Many-to-One) İlişki

Bir(li) uçtaki birincil anahtar alınarak çok(lu) uçtaki tablonun olduğu tarafta bunun için bir yabancı anahtar (FK-foreign key) kolonu yaratılır. Bu teknik, yinelenen ilişkilerde de kullanılır.

Bire-Bir Seçmeli (One-to-One Optional) İlişki

Yabancı anahtar ilişkinin her iki ucundaki tablolardan birine konulabilir. U referansının işaretlenmesi unutulmamalıdır.

Bire-Bir Zorunlu (One-to-One Mandatory) İlişki

Teklik belirten yabancı anahtar ilişkinin zorunlu tarafına yerleştirilmeli ve zorunluluk gerektirmesini kuvvetlendirmek için NN(not null) olarak, bire-bir ilişkiyi kuvvetlendirmek için de U (unique) olarak etiketlendirilmelidir.

Programda kullanılan dosyalar

DOSYA	*Sipariş Bilgilerinin Tutulduğu Dosya*		
Anahtarlar	**Tanımı**	**COBOL anahtar tipi**	**Yeni Kolon Adı**
SNO	Sipariş Numarası	Record Key	NO
MKOD	Müşteri Kodu	Alternate Key	MUSTERI_KOD
DKOD	Departman Kodu	Alternate Key	DEP_KOD
TAR	Sipariş Tarihi	Alternate Key	TARIH

DOSYA	*Dosyada Kullanılan Alanlar*	
Alan Adı	**Tanımı**	**Yeni Kolon Adı**
SNO	Sipariş Numarası	NO
DKOD	Departman Kodu	DEP_KOD
MKOD	Müşteri Kodu	MUSTERI_KOD
SKOD1	Stok Kodu	STOK_KOD
SAD1	Stok Adı	STOK_AD
BIRIM1	Stok Birimi	STOK_BIRIM
BIR-FI1	Stok Birim Fiyatı	STOK_BIRIM_FIYAT
BIR-MI1	Stok Birim Miktarı	STOK_BIRIM_MIKTAR
DAD	Departman Adı	DEP_AD
FIRMA	Firma Adı	MUSTERI_FIRMA
ILGILI	İlgili Kişi	MUSTERI_ILGILI
ADRES	Adres	MUSTERI_ADRES
TAR	Sipariş Tarihi	TARIH
ACIK	Açıklama	ACIKLAMA
AL	Alım/Satım Flag i	ALIM_SATIM
ART	Bir siparişe ait stok sayısı	STOK_SAYISI
TUTAR	Toplam sipariş tutarı	TOPLAM_TUTAR
TUT1	Siparişteki bir stok miktarına ait tutar	STOK_TUTARI

DOSYA1	*Stok Bilgilerinin Tutulduğu Dosya*		
Anahtarlar	**Tanımı**	**COBOL anahtar tipi**	**Yeni Kolon Adı**
SKOD	Stok Kodu	Record Key	KOD
SAD	Stok Adı	Alternate Key	AD
BIRIM	Stok Birimi	Alternate Key	BIRIM

DOSYA1	*Dosyada Kullanılan Alanlar*	
Alan Adı	**Tanımı**	**Yeni Kolon Adı**
SKOD	Stok Kodu	KOD
SAD	Stok Adı	AD
BIRIM	Stok Birimi	BIRIM
BIR-FI	Stok Birim Fiyatı	BIRIM_FIYAT
BIR-MI	Stok Birim Miktarı	BIRIM_MIKTAR
TUT	Stok Tutarı	TUTAR
MEV-MIK	Mevcut Stok Miktarı	MEVCUT_MIKTAR
MEV-TUT	Mevcut Stok Tutarı	MEVCUT_TUTAR
ALS	Alım/Satım Flag I	ALIM_SATIM

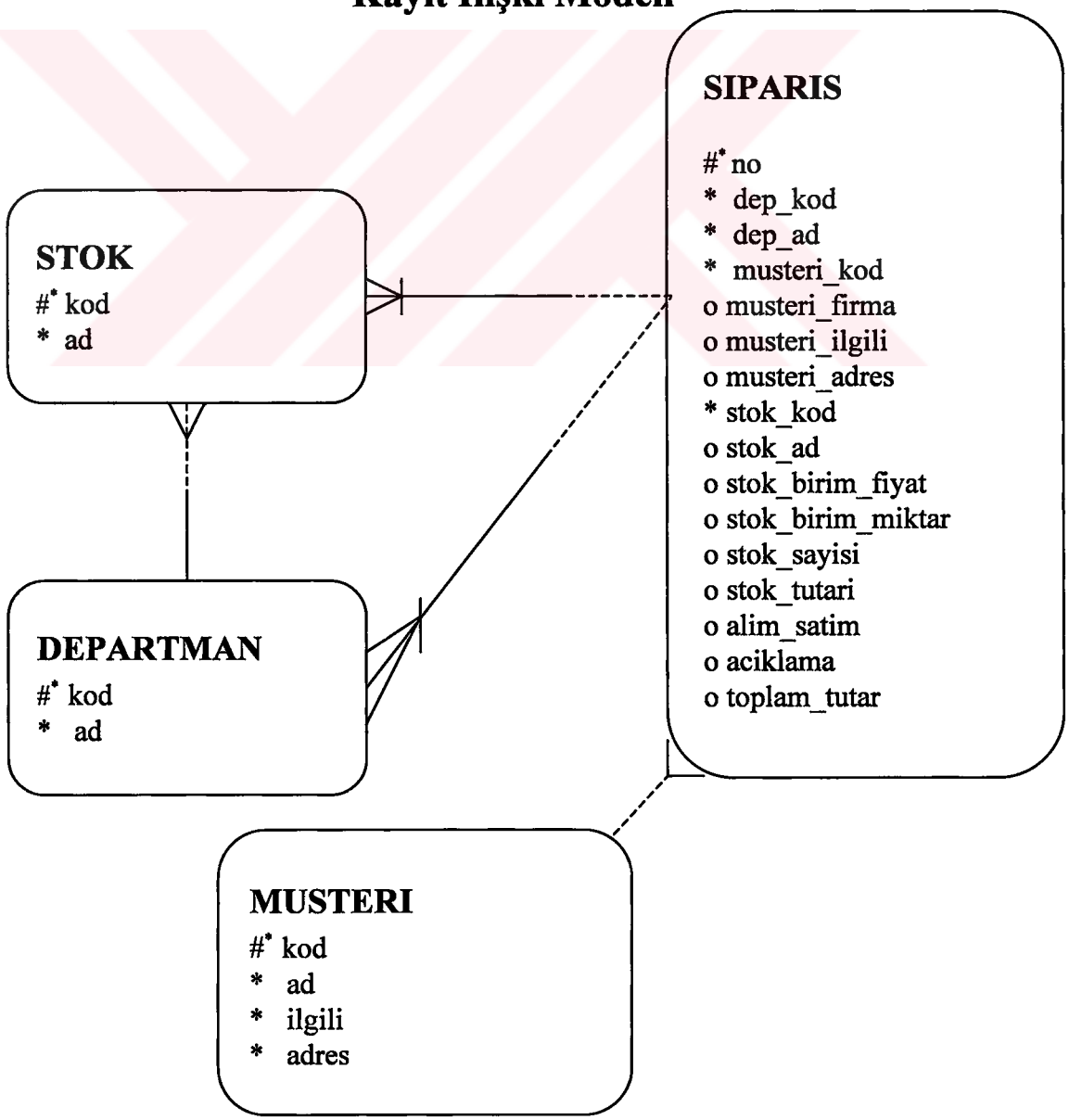
DOSYA2	*Departman Bilgilerinin Tutulduğu Dosya*		
Anahtarlar	**Tanımı**	**COBOL anahtar tipi**	**Yeni Kolon Adı**
DKOD1	Departman Kodu	Record Key	KOD
DAD1	Departman Adı	Alternate Key	AD

DOSYA3	*Müşteri Bilgilerinin Tutulduğu Dosya*		
Anahtarlar	**Tanımı**	**COBOL anahtar tipi**	**Yeni Kolon Adı**
MKOD1	Müşteri Kodu	Record Key	KOD
FIRMA1	Firma Adı	Alternate Key	AD
ILGILI1	İlgili Kişi	Alternate Key	ILGILI
ADRES1	Adres	Alternate Key	ADRES

Tabloları İsimlendirme

ESKİ COBOL DOSYA ADI	*YENİ İLİŞKİSEL TABLO ADI*
DOSYA	SIPARIS
DOSYA1	STOK
DOSYA2	DEPARTMAN
DOSYA3	MUSTERI

Kayıt İlişki Modeli

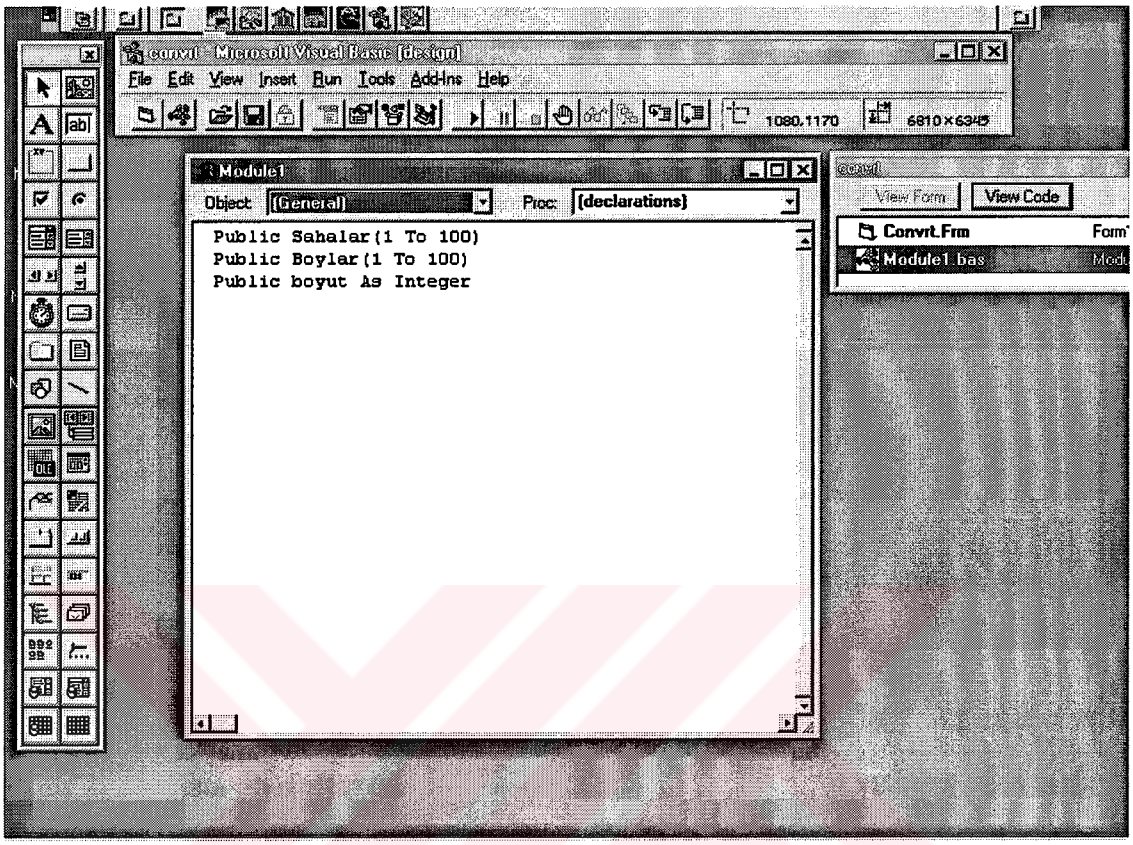


KULLANILAN SEMBOLLER		
▭	Entity	kayıt
#	Primary key	birincil anahtar
(#)	Secondary key	ikincil anahtar
*	Mandatory attribute	zorunlu alan
O	optional attribute	zorunlu olmayan alan
-<	one-to-many relationship	bire-çok ilişki
——	one-to-one mandatory relationship	bire-bir zorunlu ilişki
-----	one-to-one optional relationship	bire-bir zorunlu olmayan ilişki

VERİ DÖNÜŞTÜRME

Verileri dönüştürmek için aşağıdaki Visual Basic programı kullanılmıştır.Verileri dönüştürmek için kaynak ve hedef dosyaların yeri belirtilmeli. Bundan sonra başlangıçta ve sonuçta atlanacak karakter varsa bunların sayısı verilmeli. Yeni oluşturulacak tablo kolonlarını ayırmada kolaylık olması amacıyla bir separatör tanımlanabilir. Daha sonra field(alan) adlarıyla (burada kolonlara yeni ve daha anlaşılır bir isim vermek de mümkündür), uzunlukları verilmelidir. Her field tanımından sonra Ekle Button'una basarak field'ın aktarılması yapılır. Tüm field'lar eklendikten sonra CONVERT et Button'una basılarak tablo haline dönüştürülecek düzenli text dosya elde edilmiş olur.

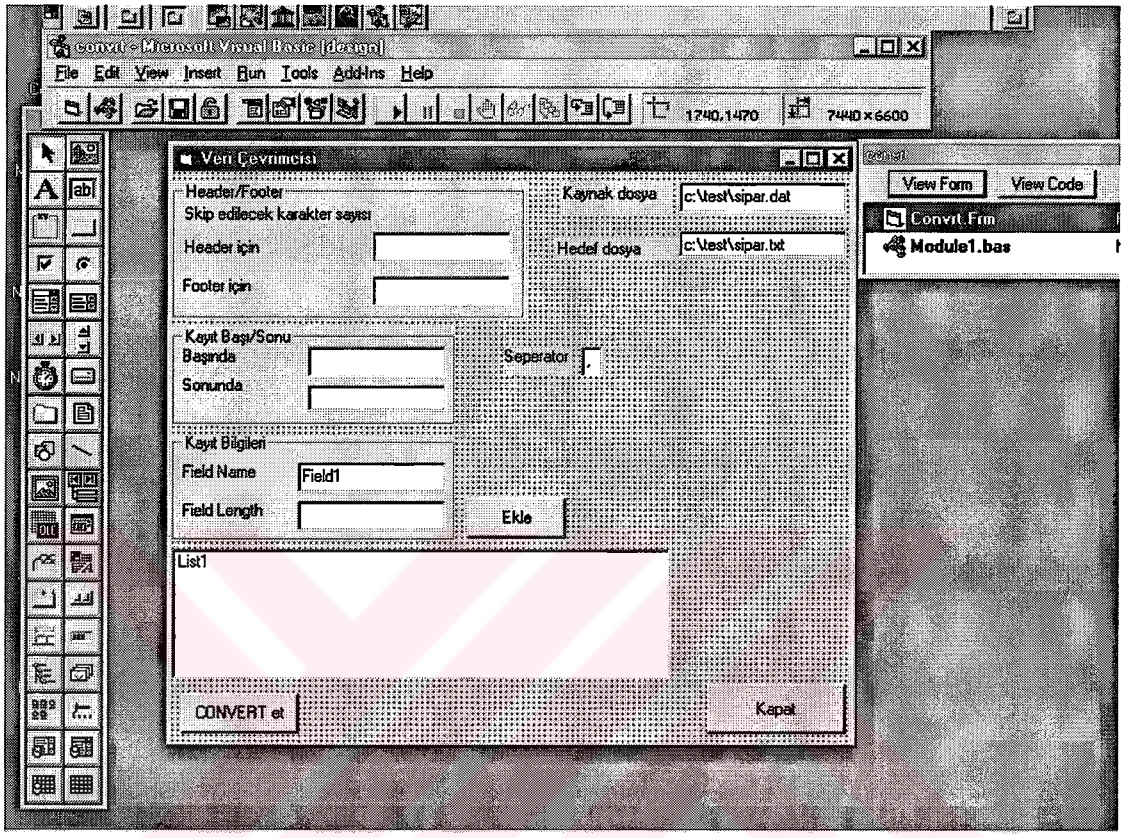
COBOL 'da yapılmış olan sipariş programında kullanılan tüm dosyalar, aşağıdaki program tarafından tablo şekline dönüştürülür.



Programın yapısı aşağıdaki gibidir:

```
Private Sub EkleCmd_Click()
    boyut = boyut + 1
    Sahalar(boyut) = Text2
    Boylar(boyut) = Text3
  List1.AddItem Sahalar(boyut) + "-" + Str$(Boylar(boyut))
  Text3.SetFocus

End Sub
```



Private Sub ConvertCmd_Click()

Open Text8 For Binary Access Read Lock Read As #1
Open Text9 For Output As #2

If Text1.Text = "" Then
Else
  bos$ = Input$(Text1, #1) 'boş oku
End If

While Not EOF(1)

hepsi = False

On Error GoTo ErrorHandler

```
abc$ = ""

If  Text4.Text = "" Then
Else
  bos$ = Input$(Text4, #1) 'başı
End If

For i = 1 To boyut

  a$ = Input$(Boylar(i), #1)
  abc$ = abc$ + a$
  If i <> boyut Then abc$ = abc$ + Text7

Next i

If Text5.Text = "" Then
Else
  bos$ = Input$(Text5, #1) 'başı
End If

hepsi = True

If (hepsi = True) And (abc$ <> "") Then
    Print #2, abc$
End If
Wend

Close #2
Close #1

ErrorHandler:
If Not hepsi Then abc$ = ""
Resume Next
End Sub


Private Sub KapatCmd_Click()
  Unload Me
End Sub
```

```
Private Sub Form_Load()
  boyut = 0
End Sub

Private Sub Frame1_DragDrop(Source As Control, X As Single, Y As Single)
End Sub

Private Sub Frame2_DragDrop(Source As Control, X As Single, Y As Single)
End Sub
```

ÖZGEÇMİŞ

Doğum Tarihi	6 Ocak 1973
Doğum Yeri	Divriği
Eğitim	1987-1990 Şişli Kuştepe Lisesi (Lise Birinciliğiyle Mezuniyet)
	1990-1994 Y.T.Ü Matematik Mühendisliği Bölümü
	(Bölüm Birinciliği ve Fakülte Üçüncülüğüyle Mezuniyet)
	1994-1997 Y.T.Ü Matematik Mühendisliği Yüksek Lisans
	(1 yıl İngilizce hazırlık programına katılım)
Stajlar	1991 Koç-Unisys(Bilgisayar Programlama)
	1992 Entegre Bilgisayar Sistemleri (İşletme)
	1993 Şark Sigorta B.İ.M (Araştırma)
İş Tecrübesi	Sümerbank A.Ş. Sistem Geliştirme Müdürlüğü'nde Programcı olarak bir yıldan fazla çalışma (halen burada çalışmaktadır).