İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

DAĞITIK VERİTABANI MANTIĞINA DAYALI SİGORTA OTOMASYONU

YÜKSEK LİSANS TEZİ

Sevil ERTÜRK

Tezin Enstitüye Verildiği Tarih : 8 Ocak 1996

Tezin Savunulduğu Tarih : 30 Ocak 1996

Tez Danışmanı : Yrd.Doç.Dr.Ali ERCENGİZ

Diğer Jüri Üyeleri : Yrd.Doç.Dr.Celal TUNCER

Yrd.Doç.Dr.Gazanfer ÜNAL

OCAK 1996

ÖNSÖZ

Tez konu seçimindeki yardımlarından dolayı Sayın Prof.Dr.Mithat Uysal, tez çalışmamdaki yardımlarından dolayı Sayın Y.Doç.Dr.Bedri Şefik, tez danışmanım Sayın Y.Doç.Dr.Ali Ercengiz ve bu çalışma sırasında manevi desteklerini esirgemeyen aileme teşekkür ederim.

İstanbul, 1996 Sevil ERTÜRK

İÇİNDEKİLER

Anabilim Dalı : Mühendislik Bilimleri
Çalışmayı Yapan : Sevil Ertürk
Danışmanı : Yrd. Doç. Dr Ali Ercengiz
Bitiriş Tarihi : 30.01.96

DAĞITIK VERİTABANI MANTIĞINA DAYALI SİGORTA OTOMASYONU

Sevil Ertürk

Anahtar Kelimeler : Dağıtık Veritabanı, ağ, istemci/sunucu,birim

Veritabanı sistemlerindeki gelişmeler beraberinde dağıtık veritabanlarında da gelişmelere sebep olmuştur. Dağıtık veritabanı bilgisayar ağı üzerinde yayılan fakat aynı sisteme ait veri koleksiyonudur ve birbirine ağ ile bağlı birimlerden oluşur.
Bu proje birden fazla coğrafi yerel yapı üzerinde hizmet veren acentaların poliçe ve reasurans işlemlerini içermektedir. Bu yapıların özerk çalışmaları ve gerektiğinde iletişim kurmaları baz alınarak dağıtık veritabanı yönetim sistemine göre tasarlanmıştır. Proje, Unix ortamında Oracle ilişkisel veritabanı ile geliştirilmiştir. Her yerel bölge kendi bilgilerini tutmaktadır ve gerektiği zaman istemci/sunucu mimarisi kullanılarak bilgi akışı gerçekleşmektedir.

INSURANCE AUTOMATİON BASES ON DİSTRİBUTED DATABASE ARCHİTECTURE

Sevil Ertürk

Keywords: Distributed database, network,client/server,side

After the development of database system has given to rise to a new field : distributed databases. A distributed database is a collection of data which belong logically to the same system but are spread over the sites of a computer network, and a distributed database system consists of a collection of sites.
In this study, an application which was built on the idea of distributed database system on the insurance sector was developed. These sites have their own computers, they work locally and they can comminicate each other when it is needed. All of the project is developed by Oracle RDMS in the context of Unix. Every machine has its own local database and data flow among this database could be realized using the client/server architecture whenever necessary.

ŞEKİL LİSTESİ

Sayfa No

Sayfa No

KISALTMALAR

VTYS	Veritabanı Yönetim Sistemi
IVTYS	İlişkisel Veritabanı Yönetim Sistemi
DVTYS	Dağıtık Veritabanı Yönetim Sistemi
VTY	Veritabanı Yöneticisi
OSI	Open Systems Interconnection
TCP	Transfer Control Protocol
IP	Internet Protocol
CPU	Central Processing Unit
SQL	Structured Query Language
DML	Data Manipulation Language
DDL	Data Defination Language
SGA	System Global Area
DTM	Distributed Transaction Manager
LTM	Local Transaction Manager
TNS	Transparent Network Substrate
API	Application Program Interface

ÖZET

Proje konusu olarak, sigortacılık sektöründe, birden fazla coğrafi yerel yapı üzerinde hizmet veren sigorta şirketi ve acentaların poliçe ve reasürans işlevlerini gerçekleştiren bir uygulama geliştirilmiştir.

Uygulama, acentaların işlemlerini merkeze bağlanmak zorunda olmadan gerçekleştirmeleri ve gerektiğinde birbirleri ile iletişim kurmaları baz alınarak dağıtık veritabanı yönetim sistemine göre tasarlanmıştır ve veritabanı uygulamaları, Oracle veritabanı yönetim sistemi üzerinde gerçekleşmektedir. Acentalarda Unix makinalar bulunmaktadır. Bir bilgisayar ağına bağlı coğrafi bir alana dağılmış olan bu makinalar, birbirine X.25 uzak iletişim ağı ile bağlıdır. Her makinanın kendi yerel veritabanı vardır ve gerektiği zaman istemci/sunucu mimarisi kullanılarak bu veritabanları arasında bilgi akışı gerçekleşmektedir.

Tez, dört ana bölümden oluşmaktadır;

Birinci bölümde, iletişim ağları, topolojiler, TCP/IP, OSI modeli ve katmanları, Internet protokolu, bağlantı şekilleri incelenmiştir.

İkinci bölümde, dağıtık veritabanı ve yönetim sistemi ile ilgili temel kavramlar, temel özellikler, hareket yönetimi, güvenirlilik, şeffaflık, problemler, avantajlar ve dezavantajlar anlatılmıştır.

Üçüncü bölümde, Oracle veritabanı yönetim sistemi hakkında bilgi verilmiş ve dağıtık mimarisi incelenmiştir.

Dördüncü bölümde ise proje konusu, iş akışı, veritabanı uygulamaları ve çeşitli programlar yer almaktadır.

SUMMARY

INSURANCE AUTOMATION BASED ON DISTRIBUTED DATABASE ARCHITECTURE

Database management has evolved from a specialized computer application to a central component of a modern computing environment. A database management system (DBMS) consists of a collection of irterrelated data and a set of programs to access that data. The primary goal of a DBMS is to provide an environment that is both convenient and efficient to use in retrieving and storing database information. The DBMS acts as an interface between the stored database and the users of the database.

Database systems are designed to manage large bodies of information. The management of data involves both the definition of structures for the storage of information and the provision of mechanisms for the manipulation of information. In addition, the database system must provide for the safety of the information stored, despite system crashes or attempts at unauthorized access. If data is to be shared amoung several users, the system must avoid possible anomalous results. The importance of information in most organizations, and hence the value of the database, has led to the development of a large body of concepts and techniques for the efficient management of data.

The most widely used types of DBMSs are the hierarchic, network and the relational. From a historical perspective, the relational data model is relatively new. The first database systems were based on either the network model or the hierarchical model. Those two order models are tied more closely to the underlying implementation of the database than is the relational model. In the years following the introduction of the relational model, a substantial theory has developed for relational databases. This theory assists in the design of relational databases and in the efficient processing of user requests for information from the database. The relational model has established itself as the primary data model for commercial data processing applications.

Relational systems offer benefits such as;

- easy access to all data
- flexibility in data modelling
- reduced data storage and redundancy
- independence of physical storage and logical data design
- a high-level data manipulation language (SQL)

After this development of database systems, computer networks have been developed, allowing the connection of different computers and the exchange of data and other resources between them. In recent years the availability of databases and computer networks has given rise to a new field: distributed databases. The data which constitute the database are stored at the different sites of the computer network, and the application programs which are run by the computer access data of different sites.

In this study, an application which is built on the idea of distributed database system on the insurance sector is developed.

Insurance is an agreement made to protect the loss of a person or a company by being paid a specific amount of money. Insurance companies are the companies whose establishment is specified with definite laws and regulations. They assure other companies and people that require to be insured. The insured people are the ones whose belongings or lives are assured by the insurance companies.

The relation between the insured people and the insurance company can be established both when one of them attempts to do this or through the insurance brokers. These brokers are called agent.

The types of insurance are called branch. Fire, transportation, car are examples for the branches. The risks which are insured in these branches are each called assurance. Schedules are the combined forms of the assurance. All this information is piled on a piece of paper which is called the insurance policy. If we show it with a simple figure, it is as follows;

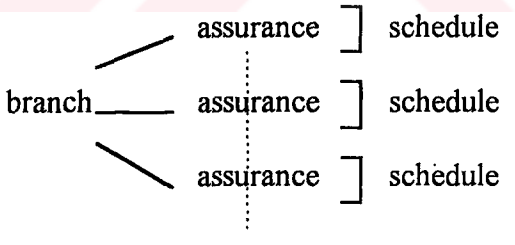


Figure 1. Relationship between branch, assurance, schedule.

The other important subject is the term of reassurance. Reassurance is the sharing of the risk amoung different companies by being paid a specific amount of money. In this way, the insurance company provides both the insured ones and its own security.

Reassurance is the international agreement designed between the insurance companies and the reassurers and with its simplest definition, it is the insurance of the insurance company.

The service in which these terms are employed is the technical service. Apart from this, in the department of damages, the damages are examined and the

necessary operations are started determining the probable damages. In the accounting department, the invoices of these damages are prepared and other necessary applications are processed.

All the structure is built upon a distributed database using the advantages of distributed processing.

Distributed Processing uses more than one processor to divide the processing for a set of related jobs. Distributed processing occurs when an application on one CPU accesses a database on another CPU. Figure 1 refers to an architecture in which a database resides on a different machine to the user process accessing it. It reduces the processing load on a single processor by allowing different processors to concentrate on a subset of related tasks, thus improving the performance and capabities of the system as a whole.

A database can easily take advantage of distributed processing by using its client-server architecture. In this architecture, the database system is divided into two parts; a front-end or a client portion and a back-end or a server portion.

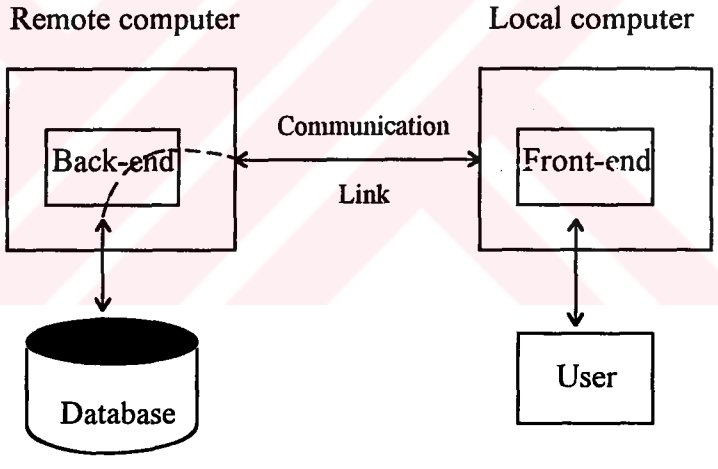


Figure 2. Distributed processing.

The client portion is the front-end database application and interacts with a user through the keyboard, display, and pointing device such as a mouse. The client portion has no data access responibilities; it concentrates on requesting, processing and presenting data managed by the server portion. The client workstation can be optimized for its job. For example, it might not need large disk capacity or it might benefit from graphic capabilities.

The server portion runs RDBMS software and handles the functions required for concurrent, shared data access. The server portion receives and process SQL and PL/SQL statements originating from client applications. The computer that manages the server portion can be optimized for its duties. For example, it can have large disk capacity and fast processors.

A distributed database system consists of a collection of sites, connected together via some kind of communications network, in which

i) Each site is a database system site in its own right, but
ii) The sites have agreed to work together (if necessary),

so that a user at any site can access data anywhere in the network exactly as if the data were all stored at the user's own site.

Each site has its own local real databases, its own local users, its own local DBMS and transaction management software and its own local data communications manager. The distributed database system can thus be regarded as a kind of partnership amoung the individual local DBMSs at the individual local sites.

The primary benefit of a distributed database is that the data of physically separete databases can be logically combined and potentially made accessible to all users on a network.

All of the application is developed by Oracle RDBMS. The Oracle RDBMS is the central Oracle product. Oracle Corporation was the first company to offer a true relational DBMS commercially, and has continually led innovations in the field of RDBMSs.

The Oracle RDBMS includes the database manager and several tools intended to assist users and DBAs in the maintenance, monitoring and use of data. It is a high-performance, fault-tolerant database management system, especially designed for online transaction processing and large database applications.

At the heart of the Oracle RDBMS is the SQL data language that is used for most database activities. SQL is simple enough to allow beginning users to access data easily and quickly, yet it is powerful enough to offer programmers all the capability and flexibility they require.

SQL was developed and defined by IBM Research, and has been refined by the American National Standards Institute (ANSI) as the standard language for relational database management systems. The SQL implemented by Oracle Corporation is a superset of the standard SQL data language.

An Oracle database can be used in a configuration as a distributed database. Oracle configured with SQL*net supports distributed queries. When data is required from remote databases, a local database server communicates with the remote database using the network, network communications software, and Oracle's SQL*net. SQL*net is the interface used to connect clients and servers that operate on different computers of a network. SQL*net also cannects database servers across networks to facilitate distributed transactions. SQL*net and Oracle's distributed database system run on most networks, the particular type of network protocol or topology does not matter.

For the distributed organization on the project, the client-server architecture and the database connection properties of Oracle were used.

As the subject of my project, I developed an application realizing policy and reassurance activies of the agenties and insurance company that serve in the sector of insurance on more than one geographical local structure.

The application has been planned according to distributed database management system. As far as this application is concerned, agent can perform their operations without necessarily having to be connected the center and applications of the database process on the Oracle database management system. There are Unix machines in the agenties. This machines, which are distributed to a geographical area connected to a network, are connected with each other by X.25. Every machine has its own local database and data flow amoung this database could be realized using the client/server architecture whenever necessary.

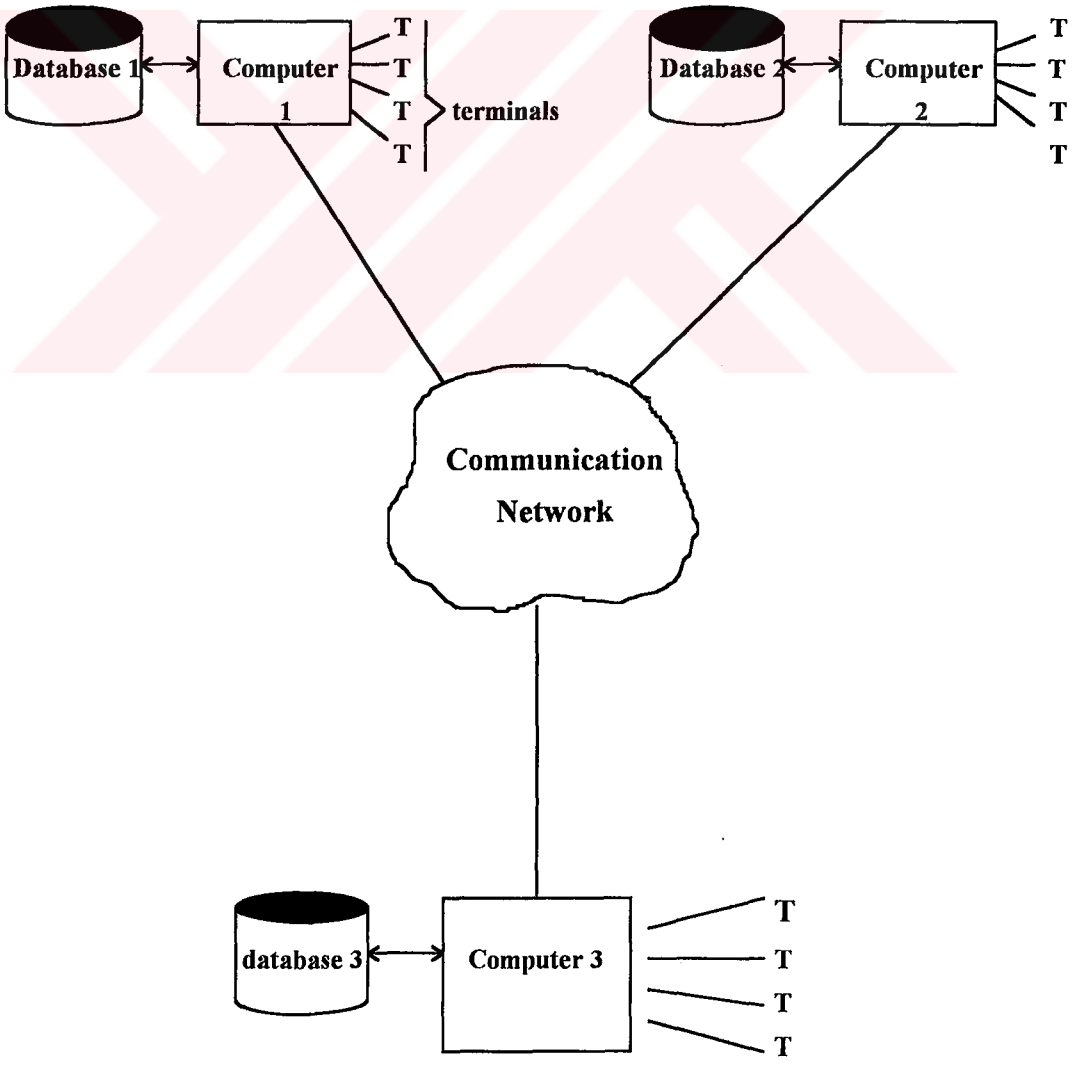


Figure 3. A distributed database on a geographically dispersed network.

My thesis consists of four main sections.

In the first section, network, topologies, TCP/IP, OSI model and layers, the Internet protocol, connection types are scrutinized.

In the second section, basic terms related with distributed database and management system, basic properties, transaction management, reliability, transparency, problems, advantages and disadvantages are revealed.

In the third section, information about the admistration system of Oracle database and its distributed architecture are mentioned.

Finally, in the last section, subject of the project, job flow, database applications and various programs are examined.

BÖLÜM 1 GİRİŞ

Veritabanı yönetim sistemi, birbiri ile ilişkili veri koleksiyonundan ve bu verilere erişim sağlayan programlar grubundan oluşan bir yazılım şeklidir ve veritabanı sistemlerindeki gelişimler yeni bir alanı da beraberinde getirmiştir: Dağıtık veritabanları.

Son yıllarda dağıtık veritabanları, bilgi işlemede önemli bir alan olmaya başlamıştır ve bu önem artışı, büyük bir hızla devam etmektedir.

Dağıtık veritabanlarının gelişmesini sağlayan iki sebep vardır. Birincisi küçük boy bilgisayarların son zamanlardaki gelişimlerinin, dağıtık sistemlerin gelişimi için gerekli donanım olanaklarını içermesi, ikincisi ise dağıtık vertibanlarının yoğun bir gelişim içinde bulunan iki teknoloji üzerine kurulmasıdır. Bu teknolojiler, bilgisayar ağları ve veritabanı teknolojisidir.

Dağıtık veritabanı, tek bir bilgisayar yerine bir bilgisayar ağı üzerine kurulan bütünleşik bir veritabanıdır [1]. Veri, iletişim ağının farklı birimlerinde saklanır ve uygulama programların farklı birimlerdeki veriye ulaşabilir.

Yeni problemlerin oluşması, dağıtık veritabanının gelişim hızını daha da arttırmıştır ve bu problemleri çözme yolunda araştırmalar yapılmaktadır. Bu araştırmaların içeriği, dağıtık veritabanı özelliklerine sahip olan yeni bir disiplin oluşturmaktır.

Dağıtıklığın kullanılır olmasıyla beraber bilgisayarlara dayalı bilgi sistemleri, özellikle coğrafi bir yapı üzerinde dağılmış organizasyonel yapılar için çok büyük avantajlar getirmektedir. Bilgi sistemlerine getirdiği avantajlar, belki de son yıllarda bilgisayar dünyasında toplumsal alandaki en büyük gelişme olmuştur.

avantajlar getirmektedir. Bilgi sistemlerine getirdiği avantajlar, belki de son yıllarda bilgisayar dünyasında toplumsal alandaki en büyük gelişme olmuştur.

Dağıtık sistemlerde, her türden durumun gereksindiği, bütün verilerin tek bir veri merkezinde toplandığı geleneksel sistemlerden tamamen farklı bir yaklaşım benimsenmiştir. Veriler, bu verilere gereksinim duyulan noktalarda bulunmaktadır. Bütün verilerin tek bir merkezi bilgisayarda tutulması, kuruluşların etkin işlemesine olanak tanımadığı için, bilgi işlem dünyasındaki gelişmeler zorunlu olarak böylesi bir noktaya gelmiştir. İletişim teknolojisindeki gelişmelerle birlikte, farklı mimarilere sahip sistemlerin haberleşebilmesi dağıtık sistemlerin teknolojik altyapısını hazırlamıştır.

Kullanıcı arabirim işlemleri, mantıksal uygulamalar ve veritabanı işlemleri olarak üç işlem bütününden oluşan tipik bir uygulama, geleneksel yaklaşım çerçevesinde tek bir sistem üzerinde çalıştırılır. Dağıtık sistemlerde ise, sunucu/istemci mimarisi, uygulama işlemlerini ve sistem kaynaklarını farklı makinalar üzerine dağıtarak, ki bu dağıtım iletişim teknolojisinin sağladığı olanaklarla donanım platformu bağımsızlığı sağlayarak gerçekleşir, performans artışı sağlar.

Bugün birçok organizasyon merkeziyetçilikten uzaklaşmaktadır. Dağıtık veritabanı yaklaşımı, bu tür organizasyonlar için en uygun tipi teşkil etmektedir. Mesela, merkezi sistemde özerk sistemlerin eklenmesi sistemin başlangıç boyutlarının değiştirilmesine neden olurken, dağıtık veritabanı minimum derecede etkilenmektedir. Özellikle coğrafi olarak yayılmış sistemlerde merkezi bir yapı düşünüldüğünde yapılan bir işlemin merkez üzerinden çalıştığı düşünülürse, bunun getireceği haberleşme maliyeti büyük rakamlara ulaşacaktır. Oysa dağıtık veritabanlarında yerel işlemler yalnız yerel makinayı meşgul ettiğinden ve sadece global işlemler için haberleşme kullanılacağından ağ trafiği daha az olacaktır ve maliyet de düşecektir.

Coğrafi bir yapı üzerinde dağıtılmış işlemlere sahip organizasyonlar, verilerini dağıtılmış olarak saklayabilmektedir. Fakat eğer bilgi sistemi dağıtık değilse, bu veri

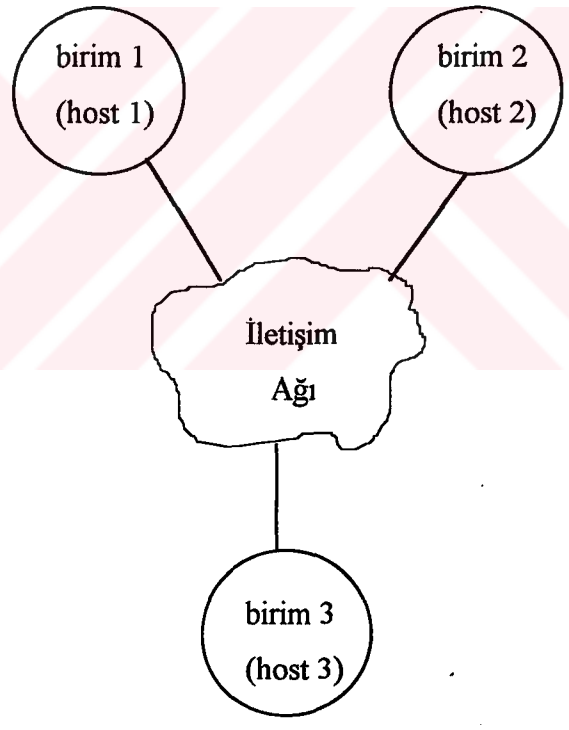
ve kaynakları paylaşmak imkansızdır. Bu nedenle dağıtık veritabanı sistemi, verilerin gerektiğinde paylaşılması ile bilgi sistemlerinin daha sağlıklı işlemesini sağlamaktadır.

Tez uygulaması olarak, coğrafi yerel yapı üzerinde sigorta hizmeti gerçekleştiren acentaların ve sigorta şirketinin ilgili işlevleri dağıtık veritabanı ortamında tasarlanmıştır. Uygulamaların yazılımı ve geliştirilmesi, Oracle veritabanı ortamında gerçekleşmiştir.

BÖLÜM 2 BİLGİSAYAR AĞLARI

2.1. Bilgisayar Ağ Tanımı

Bilgisayar ağları, bilgi alışverişi yapma yeteneği olan birbirine bağlanmış özerk bilgisayar koleksiyonudur. Ağ üzerindeki bilgisayarlara, bilgisayar ağları literatüründe host denilir [1].



Şekil 2.1. Bir bilgisayar ağ modeli.

Bir bilgisayar ağı, farklı tipteki iletişim bağlantılarından (telefon hattı, uydu bağlantıları) oluşur ve bilgisayar ağının sağladığı temel olanak şudur: bir birimde icra

edilen proses, diğer bir birimdeki prosese bir mesaj gönderir. Bu olanak, bilgisayar ağının gerçek yapısından bağımsızdır, ağın herhangi iki birimi iletişim kurabilir.

Bu temel özelliği tanımlayan parametreler şunlardır:

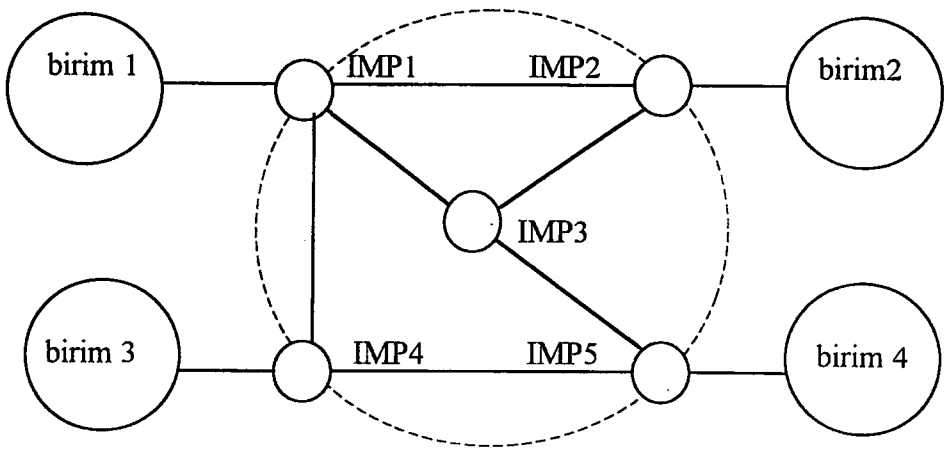
- Hedefe gönderilen mesajın gecikme durumu : Eğer ağ kullanımı düşükse, gecikme; ağı oluşturan bileşen özelliklerine bağlıdır. Ağ kullanımı yoğunsa, gecikme uzun sürecektir. Çünkü bu mesajdan önceki diğer mesajların gönderilmesi gerekmektedir.

- Gönderilen mesajların maliyeti : Bu maliyet, her bir mesaj ve mesaj uzunluğu ile orantılı ek maliyeti içerir.

- Ağ güvenirliliği : Ağ güvenirliliği, hedefe gönderilen mesajın doğruluk olasılığını içerir.

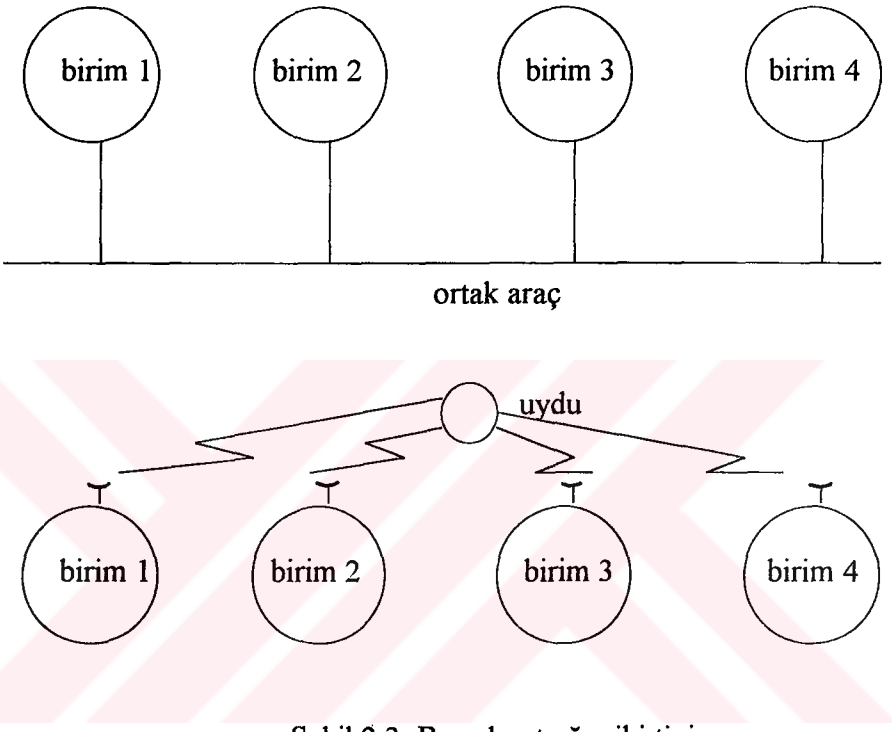
2.2. İletişim Ağ Tipleri

İletişim ağı, IMP (arayüz mesaj prosesörü) adı verilen bir grup prosesörden oluşur. Bu prosesörler, telefon hatları gibi iletişim hatları ile birbirine bağlanmıştır. Her bir birim, bir IMP'e bağlıdır. Bir birim, diğer bir birime mesaj gönderdiği zaman, mesaj bir IMP'ye gönderilir, orada saklanır. Sonra diğer IMP'e gönderilir, orada da saklanır ve bağlanılacak birimin IMP'sine ulaşana kadar bu böyle devam eder. Bu ağ tipine noktadan noktaya (point-to-point network) ağ denir.



Şekil 2.2. Noktadan noktaya ağ.

Noktadan noktaya ağda, IMP'ler, alternatifler arasında, mesajın gönderilme yolunu seçmekle sorumludur. Şekil 2.2'de görüldüğü gibi, eğer birim 1, birim 4'e bir mesaj gönderirse, mesaj, IMP1-IMP3-IMP5 yolunu takip edebilir. Diğer bir yol da IMP1-IMP2-IMP3-IMP5 olabilir. Yol seçim fonksiyona yönlendirme (routing) denir. Ağ, yönlendirme fonksiyonunu icra edebilmelidir.



Şekil 2.3. Broadcast ağın iki tipi.

Şekil 2.3'de ortak araç ve uydu üzerine kurulmuş iki ağ örneği görülmektedir. Bu iki iletişim ağının ortak özelliği; bir birimden gönderilen bütün mesajları bütün birimlerin almasıdır. Bu yüzden, diğer mesajları gözardı eden, bu mesajları kabul etmesi için her bir birime izin veren bir mekanizma olmalıdır.

Bu mekanizma, hedef birim bilgisini içeren mesajlardan oluşur.

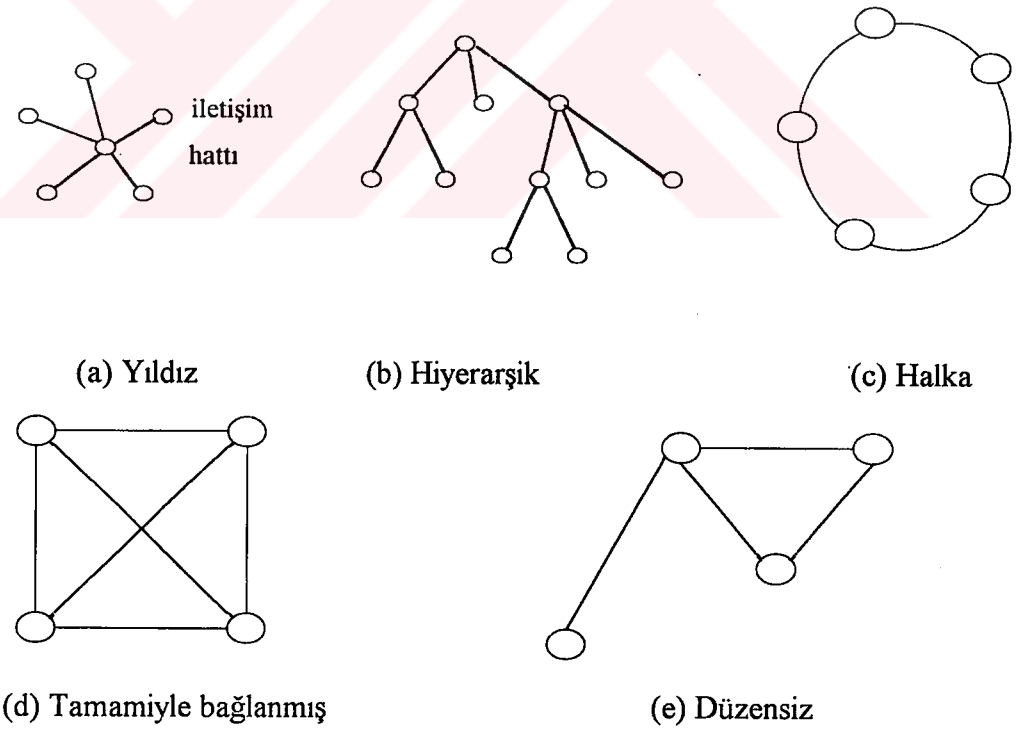
Bu iletişim ağ tipine broadcast ağ denir. Broadcast ağda, bir mesajın birden çok birime gönderilmesi, mesajın sadece bir birime gönderilmesinden daha pahalı değildir.

İletişim ağları, lokal ağ (LANs) ve coğrafi olarak dağıtık ağ (WANs) olarak da sınıflandırılabilir.

Bu fark, eğer birimler arasındaki mesafe az ise, kullanılan iletişim hatlarının tipine bağlıdır. Bir tipik lokal ağ, 1000 metre uzaklığı aşmaz. Lokal ağın birimleri, aynı bina üzerinde olabilir. İletişim hatları, yüksek frekanslı sinyalleri taşıyan kablolar ile sağlanır. Lokal ağlar, broadcast ağlardır.

Coğrafi olarak dağıtık ağlar, telefon hatları üzerine kurulmuştur. Güvenirliliği, lokal ağlardan daha azdır.

Şimdi de kısaca ağ topolojisine değinelim. Yönlendirme fonksiyonları, fiziksel bağlantıların nasıl yapılandığını içerir. Bunun için algoritmalar geliştirilmiştir. Şekil 2.4'de temel topolojiler görülmektedir.



Şekil 2.4. Temel topolojiler.

Düzensiz topolojide, birimler arasındaki bağlantı bir yolu takip etmez.

Yıldız topolojide, bütün bilgisayarlar, ağ üzerindeki iletişimi koordine eden merkezi bir bilgisayara bağlanmıştır. Eğer iki bilgisayar iletişim kurmak isterse, önce merkezi bilgisayara bağlamak durumdadırlar.

Hiyerarşik topolojide aynı seviyedeki birimler arasındaki bütün iletişimler, ortak birim bulunana kadar yukarıya kadar ilerlemek zorundadır.

Tamamıyle bağlanmış topolojide, her bir birim, diğer birimlere bağlanmış durumdadır. Bu yapı, daha iyi performans ve güvenirlilik sağlar.

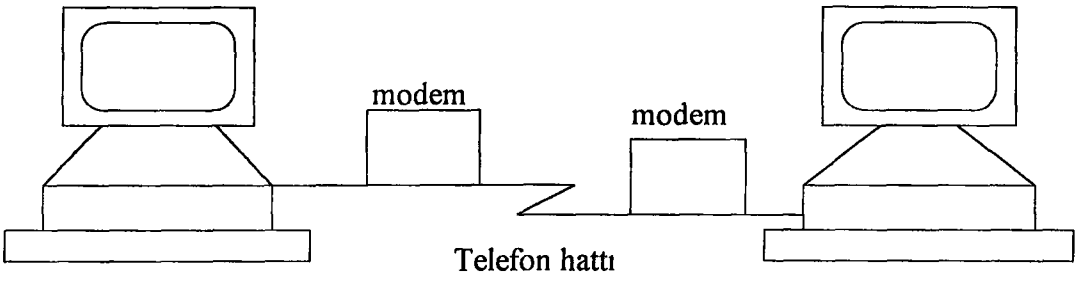
Halka topolojide bütün birimler halkaya bağlanmıştır ve halkadaki bütün mesajları alırlar. Bu topoloji, lokal ağlarda kullanılır.

2.3. Bilgisayarlararası Bağlantı Şekilleri

2.3.1. Dial-up

Bu bağlantı, telefon ile gerçekleşir. İletişim, telefon kapatılmadığı sürece devam eder ve telefon kapatıldığı zaman aynı hattan başkası yararlanır. Bu iletişim bir telefon numarası ile gerçekleşir [2].

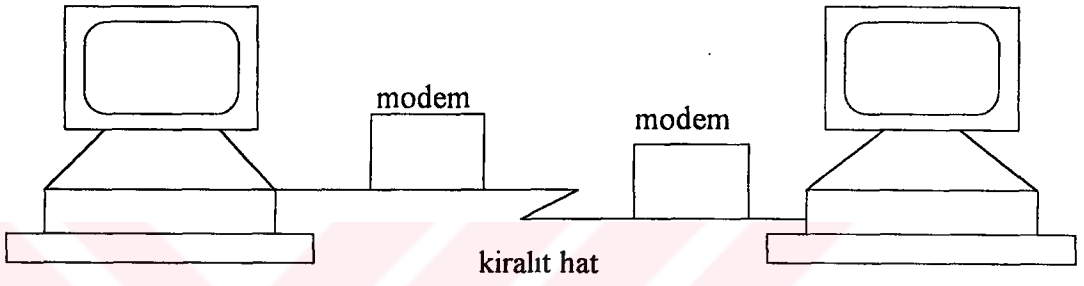
Bu bağlantıda, dijital mesajların, telefon hatlarının anlayabileceği şekilde dönüştürülmesi gerekir. Modem, bu dijital ve analog mesajlar arasındaki bir arabirim, bir dönüştürücüdür. Modemde aynı anda sadece iki bilgisayar konuşabilir ve modemlerin hızlı olması önemlidir.



Şekil 2.5. Dial-up.

2.3.2. Kiralık Hat

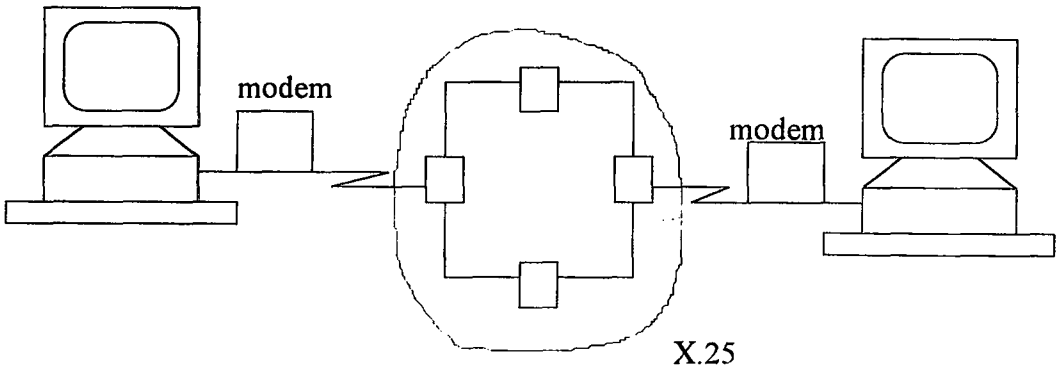
Özel PTT hattı, iki bilgisayarı birbirine bağlamak için kullanılır. Kiralık hatlar, birimler arasındaki, kalıcı bağlantıdır. Bu hat, sürekli iki nokta arasında açıktır. Aynı anda sadece iki bilgisayar haberleşebilir. Mesafe arttıkça kirası da artmaktadır. Bu yüzden mesafe yakınsa bu bağlantı tercih edilmelidir.



Şekil 2.6. Kiralık hat.

2.3.3. X.25 (Turpak)

X.25, bilgisayarlar arasında haberleşme sağlamak için kullanılan, her ülkenin PTT'sine ait bir taşıyıcı ağdır. Öncelikle X.25'e abone olmak ve bir X.25 numarası almak gerekir. Bu numara ülke kodu ve X.25 abone numarasından oluşur. X.25 ağının kendine özgü santralleri ve modemleri vardır.



Şekil 2.7. X.25.

2.4. Protokol ve Oturumlar

Bilgisayarlar arası haberleşme kurallarına protokol denir. Bir protokol, mesaj alışverişi konusunda gönderici ve alıcının nasıl bir anlaşmaya varacağını, birbirlerini nasıl tanıyacaklarını, alışveriş yapılacak mesaj sayısını ve bir mesajın cevap bekleyip beklemediği konularında saptama yapar [1].

Oturum ise iletişim kurmak isteyen iki proses arasında gerçekleşir ve mesaj alışverişi bitine kadar oturum kapanmaz. Oturum, iki proses arasındaki fiziksel bağlantı varlığına gerek duymaz. Oturum kavramı, iletişim ağı tarafından sağlanır. Aynı oturumda, prosesler arasında gidip gelen mesajlar, yönlendirme algoritmasının aldığı karara göre farklı yollar takip edebilirler. Bazı durumlarda prosesler arasında gidip gelen mesajlar düzensiz bir yola sahip olabilirler. Bu oturumun başlaması için uygun bir durum değildir. Bu durumda gönderilen her bir mesaj için protokol icra edilir. Bu yaklaşıma datagram denir.

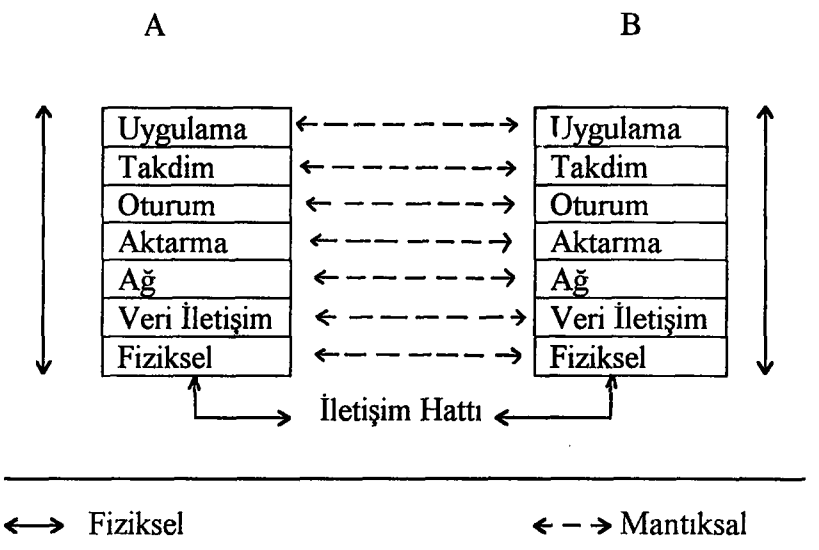
2.5. OSI Modeli ve Katmanları

Bilgisayar iletişim ağlarındaki büyük gelişmeler, bilgisayar firmalarını bazı standartlar üzerinde anlaşmalara götürmüştür. Bu standartlar, uluslararası standartlar organizasyonu tarafından gerçeklenen OSI (open systems interconnection) modeline uyum sağlamaktadır.

OSI'nın amacı; birbirine bağlanmış farklı sistemler için protokol sağlamaktır. OSI, verilen sistemler arasında ne şekilde yol alacağını belirleyen yedi katmanlı bir yapıyı ifade eder.

A sisteminden B sistemine gönderilen bir mesaj. A sisteminin yedinci katmanında birinci katmanına kadar iner, ağ ortamından geçerek B sisteminin birinci katmanından yedinci katmanına kadar çıkar.

Her bir katman, ilgili birimlerle karşılıklı birebir iletişim halindedir (Şekil 2.8).



Şekil 2.8. OSI katmanları.

Eş girişler ve fonksiyonlar birimler arasında iletişim kurar. Verinin fiziksel akışı,parametreler, her bir birimdeki her bir katmanda hareket halindedir. Her katman servisi, katmanlardaki fonksiyon özellikleri ile tanımlanır.

1. katmandan 4.katmana kadar işlemcilerin birbirleri ile bağlantısı sağlanır, beşinci katmandan yedinci katmana kadar işlemcilerin üzerinde çalışan uygulamaların birbirleri ile iletişimi gerçekleşir.

Katmanları kısaca şöyle açıklayabiliriz:

7- Uygulama Katmanı (Application layer) : Bu katmanda, ağı kullanan uygulama programları bulunur.

6- Takdim Katmanı (Presentation layer) : Sunulan verinin standardizasyonu yapılır.

5- Oturum Katmanı (Session layer) : Uygulamalar arasında oturumların yönetimi gerçekleştirilir.

4- Aktarma Katmanı (Transport layer) : Aktarılan verinin, güvenli veri akışını sağlar, hata denetimi ve düzeltmesini yapar.

3- Ağ Katmanı (Network layer) : Yönlendirme (routing) ve sıkışıklık (congestion) yönetimi ile sorumludur.

2- Veri İletişim Katmanı (Data link layer) : Noktadan noktaya bağlantı sırasında verilerin güvenilir, düşük maliyetli, etkin ve hatasız olarak gönderilmesini sağlar.

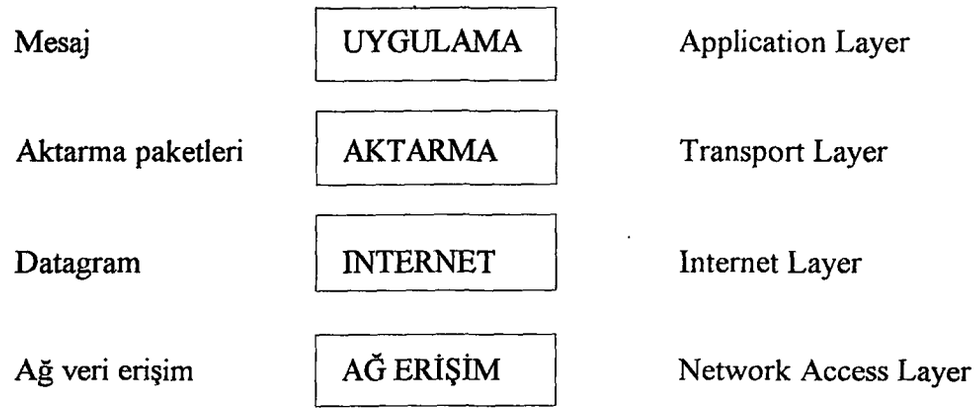
1- Fiziksel Katman (Physical layer) : İletişim yolu üzerinde verilerin aktarılmasını sağlar. Sistemin fiziksel karakteristiklerini taşır.

2.6. TCP/IP ve Katmanları

TCP/IP, karşılıklı etkileşimli bir veri transfer servisine verilen genel addır. Bir yazılım, donanım ürünün adı değildir. Bu protokol, dünya üzerinde bütün bilgisayarları birbirine bağlamak için bir çözüm olarak görülmektedir [2].

TCP/IP adı, TCP (transfer control protocol) ve IP (internet protocol) adlı iki parçanın birleşmesi ile oluşur. Bu protokol, Internet'e bağlı bütün bilgisayarlar tarafından desteklenir. Veriler bir yerden diğerine bu protokol kullanılarak aktarılır. Veri transferinin gerçekleşebilmesi için bu iki bilgisayarın bir şekilde bağlı olması yeterlidir. TCP protokolü ile bir bilgisayar, başka bir bilgisayar üzerinden diğer bir bilgisayar bağlanabilir.

IP, Internet üzerinde bilgi akışının nasıl bir şemada sağlanacağını gösteren bir protokol dizisidir. TCP/IP ağları birbirleri ile IP yönlendirme yolu (IP router) ile haberleşirler. IP yönlendiricileri bağlı oldukları her ağa üyedirler ve bu ağlar arasında bilgi alışverişini düzenlerler.



Şekil 2.9. TCP/IP katmanları.

4- Uygulama katmanı (Application Layer) : Bu katmanda, ağı kullanan uygulama programları bulunur. Aktarma katmanı ile veri alışverişinde bulunarak, bu katmana gerekli bilgiyi ve mesajı geçirirler.

3- Aktarma katmanı (Transport Layer) : Bu katmanın görevi, güvenilir veri akışını sağlamaktır. Ayrıca hata kontrolü yapar ve hata durumunda düzeltilmesini sağlar. Uygulama programları arasındaki iletişimi düzenler.

2- Internet katmanı (Internet Layer) : Yönlendirme ve trafik denetimi gibi işlevleri vardır. İki bilgisayar arasında olan iletişimi kontrol eder. Datagramların kontrolü sağlanır.

1- Ağ erişim katmanı (Network Access Layer) : Bu katman, fiziksel ağlara erişimi kontrol eder ve verilerin etkin ve hatasız bir dizi halinde gönderilmesini sağlar.

2.7. Internet ve Internet Araçları

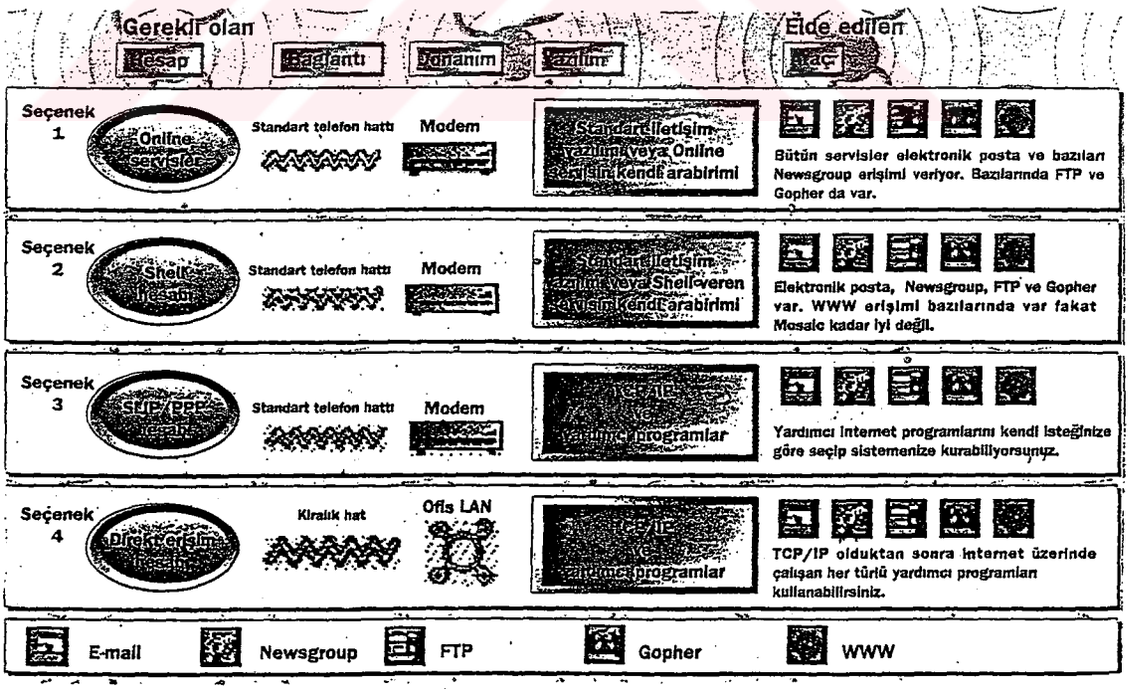
2.7.1. Internet

İlerleyen teknoloji karşısında bilgi işlem sistemlerindeki verilerin ağ üzerinden kullanıcılara açılmasının zorunluluk haline gelmesi bununla ilgili bir çok yazılıma

neden olmuştur. Uzak şirketlerin bilgisayarlarını birbirine bağlamak gerekliliği, Internet çalışma (working) kavramını, bu kavramın hayata geçmesi ile de Internet kavramını ortaya çıkarmıştır. Internet'ı birbirine bağlanmış ve dünyayı saran yüzlerce bilgisayar ağı oluşturmaktadır [3].

Internet, ilk başlarda sadece elektronik posta ve dosya alışverişi amacıyla kurulmuş bir ağ şebekesiyken, artık reklam verilebilen, üzerinde dergi ve gazete yayımlanabilen, müzik çalabilen, elektronik alışveriş yapılabilen çok büyük ve hızlı bir digital pazar olma yolunda emin adımlarla ilerlemektedir.

Internet'ın üzerinde çalışan akıllı programlar vardır. Bunların sayesinde bir yerden Internet'e girdikten sonra aranılan bilgiye, bu bilgi hangi ülkenin hangi bilgisayarında olursa olsun erişmek çok kolay ve hızlı bir hale geliyor. Sistem, farklı protokoller üzerinde çalışan tüm bilgisayar sistemleri arasında haberleşmeyi ve iletişim kurmayı amaçlıyor. Internet üzerinde sayısal forma dönüştürülmüş herhangi bir bilgi, bir diğer uçtaki bilgisayara anında aktarılabiliyor.



Şekil 2.10. Internet erişim seçenekleri.

Internet'ın herhangi bir sahibi yoktur. Ağ servisleri, herhangi bir kurum veya kuruluş tarafından da yönetilmemektedir. Önceleri sadece birkaç üniversite ve araştırma kurumunu bağlayan ağ, ticari ve resmi kurumların katılımı ile büyümüş, diğer ülkelerin ulusal ağları ile birleşerek bugünkü halini almıştır.

Bütün olanaklarına karşın, Internet'ın sınırlamaları vardır. İlk başlarda araştırma ve militer alanlarda hizmet vermesi planlanan ağ, serbest elektronik piyasası için tasarlanmamıştır. Bu yüzden Internet'ın güvenlik açısından güçlü bir koruma şeması yoktur. Kredi kart numaraları vs. kullanarak alışveriş eden kullanıcılar için çok önemli olan bilgilerin tam güvenlik altına alınması çok zordur.

2.7.2. Internet Araçları

Modemler, evlerdeki bilgisayarlardan Internet'e çıkmak için en çok başvurulan yollardan biridir. Türkiye'nin Internet bağlantısının büyük bir kısmı üniversitelerden yapılmaktadır.

FTP (File Transfer Protocol)

FTP, en çok kullanılan araçtır. FTP ile uzaktaki Internet'e bağlı bir bilgisayara erişilebilir. Oradaki dosyaları, izin verilen erişim ayrıcalıkları çerçevesinde kendi bilgisayarınıza alabilirsiniz. Dosyaların FTP ile alınmasına izin veren Internet'e bağlı bilgisayarlara "FTP birimi (site)" adı verilir. Bu birimlere bağlanıldığında kullanıcı adı ve şifresi sorulur. FTP birimleri, kullanıcılar arasında dosya alışverişi ve yazılı belgelerin Internet kullanıcılarına açılması için uzun yıllardır kullanılan bir yöntemdir. Tek olumsuz tarafı kullanımın zor ve yavaş olmasıdır.

GOPHER

Gopher, ilk çıktığından beri Internet üzerindeki bilgisayarlarda dosya arama, dosya çekme, belge görüntüleme gibi işlemleri aşırı derecede basitleştirdiği için oldukça tutulan bir uygulamadır. Gopher, FTP'den daha hızlı çalışır.

WWW Uygulamaları

Bu uygulamada, WWW servislerine bağlanan kullanıcının bilgisayarının kapasitesine ve kullandığı WWW programına göre ses, grafik ve hatta film bile aktarılabilir.

WWW uygulamaları oldukça yetenekli olduklarından, dosya görüntüleme, dosya kopyalama, araştırma yapma gibi işleri Gopher programlarına göre daha kolay bir hale getirir.

2.8. Temel IP Yönlendirmesi (Routing)

TCP/IP ağlarında, ağa bağlı PC, Mainframe gibi her üyenin dört byte (30 bit) uzunluğunda bir numarası vardır [4]. Bu numara, noktalarla ayrılmış dört desimal veya heksadesimal sayı olarak ifade edilir. IP, bu ağ numarasına bakarak paketlerin yönlendirmesini yapar. Aynı veri bağlantı ağına bağlı bütün üyelerin aynı ağ numarasına sahip olmaları gerekmektedir. Her yönlendirme (routing) tablosu şu formatta olmalıdır;

< ağ numarası,	gateway numarası,	flagler >
193.140.193.0	193.140.196.65	u

Bir yönlendirici, paketleri geçirirken, çerçevelerini (frame) çözer ve yeniden çerçevelendirir. Fakat kaynak ve hedef IP adresleri değiştirilmelidir.

Yönlendirme protokolu, ağ üzerindeki her bir yönlendirme birimindeki yönlendirme tablolarını güncelleştirmek için kullanılır.

Yönlendirme protokolü, ağ üzerinde ulaşılmak istenen hedefe en etkin biçimde bir yol çizmek için yönlendiricileri (router) aktif kılar. Yönlendirme protokolü, IP'nın bir parçası değildir.

BÖLÜM 3 DAĞITIK VERİTABANLARI

3.1. Dağıtık Veritabanı Tanımı

Son yıllarda, dağıtık veritabanları, bilgi işlemenin önemli bir alanı haline gelmiştir ve bu, dağıtık veritabanının hızla ilerlemesini sağlamıştır.

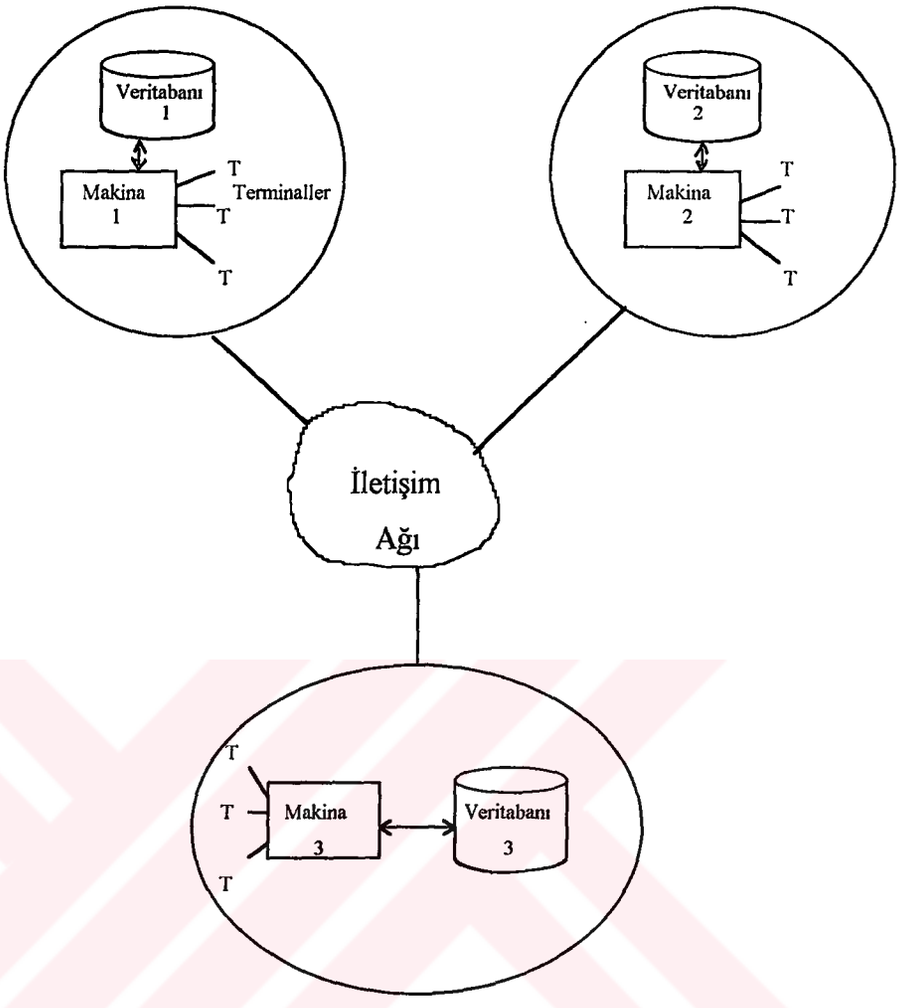
Bu hızlı ilerleyişin organizasyonel ve teknik sebebleri vardır. Dağıtık veritabanları, merkezi veritabanının kusurlarını yok etmektedir ve birçok organizasyonel yapılar için avantajlara sahiptir.

Dağıtık veritabanı, mantıksal olarak aynı sisteme ait fakat bir ağ üzerinde farklı birimlere yayılmış veri koleksiyonudur [1].

Veriler, bu verilere gereksinim duyulan birimlerde bulunur. Dağıtık bir veritabanı, birimlerde görülen işlemlerin her aşamada merkezi bilgisayara işlenmesi zorunluluğunu ortadan kaldırır ve birimlerin özerk bir organizasyon şeması içinde etkinliklerini sürdürmelerini mümkün kılar.

Her birim, kendi veritabanına sahiptir. Her birimin kendi yerel kullanıcıları, yerel veritabanı yönetim sistemi, hareket yönetim yazılımı ve veri iletişim yöneticisi vardır. Bir kullanıcı, veri üzerindeki işlemlerini, veritabanı dağıtık değilmiş gibi icra etmelidir [5].

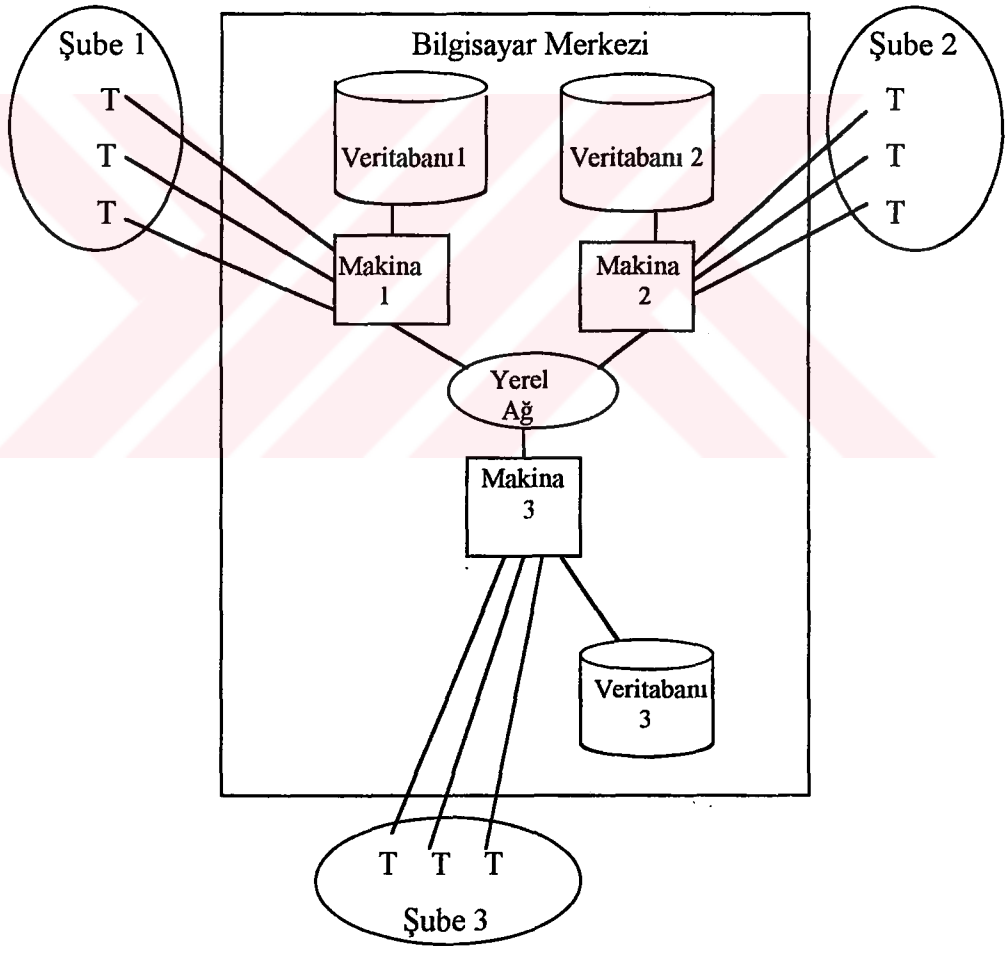
Örnek olarak bir bankanın coğrafi olarak yayılmış üç tane şubesi olsun. Her bir şubede bulunan bir bilgisayar, terminallerden gelen istekleri kontrol etmekte ve kendi veritabanında hesap verilerini tutmaktadır. Her şubedeki bilgisayar, dağıtık



Şekil 3.1. Coğrafi sistem üzerinde dağıtık veritabanı.

veritabanının bir birimidir ve birbirlerine iletişim ağı ile bağlı bilgisayarların oluşturduğu bu veritabanının bir alt sistemidir. Bir şubenin terminallerinden gelen istekler, şubenin veritabanına ulaşılarak icra edilmektedir. Bu yüzden bu uygulamalara, yerel uygulamalar denir. Eğer birden fazla şube veritabanındaki veriye erişim uygulamaları sözkonusu ise bu uygulamalara da global uygulamalar denir. Örnek vermek istersek; bankanın bir şubesinin hesabından, başka bir şubesine yapılan fon transferi global bir uygulamadır ve bu farklı birimlerdeki veritabanlarının güncellenmesi gerekmektedir. Gerçekte; burada önemli olan veritabanının güncelleme işlemi değil, iki güncellemenin yapıldığından veya yapılmadığından emin olmaktır.

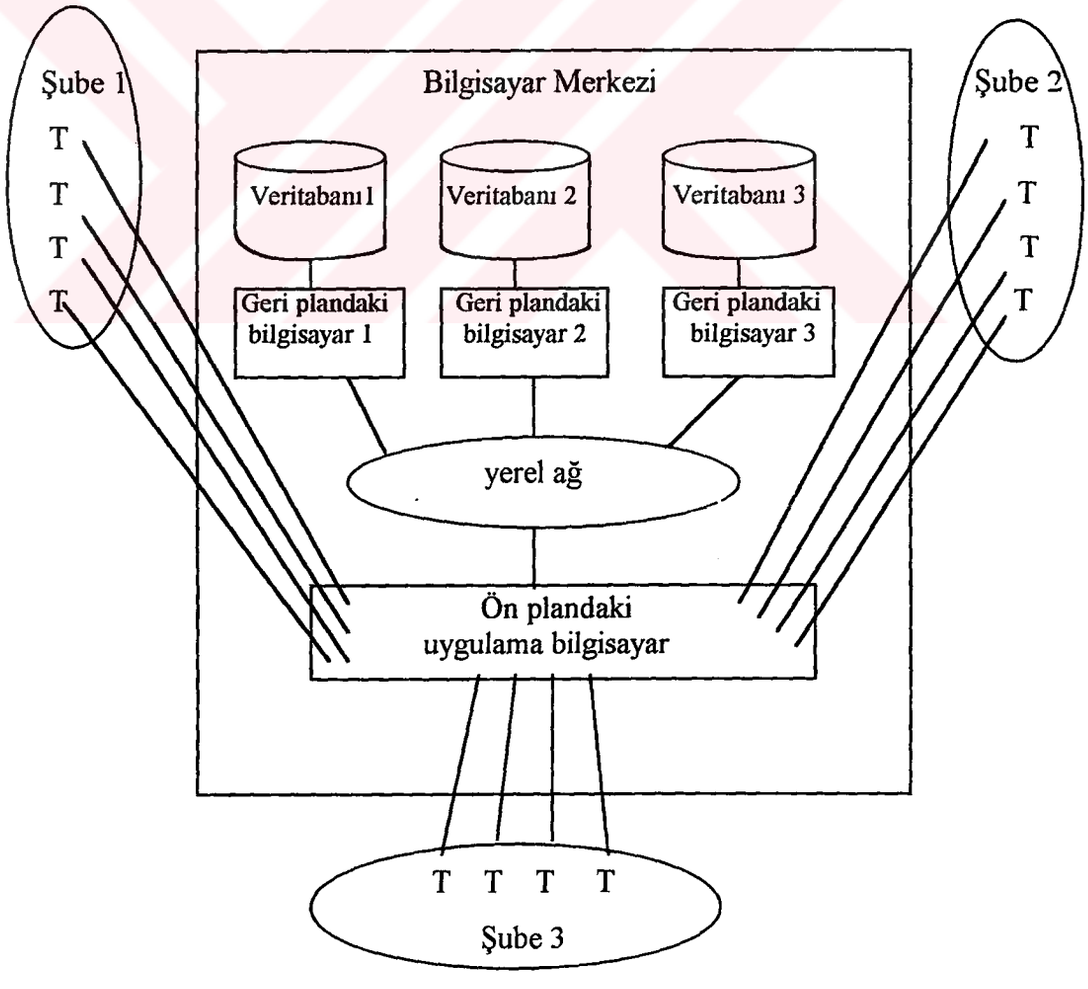
Bu kez, bu uygulamanın Şekil 3.2 yapısı üzerinde gerçekleştiğini varsayalım. Burada bilgisayarlar diğer örneğimizdeki aynı işlemcilere sahip ve aynı bina üzerinde üç farklı şubede hizmet verecek şekilde ayrılmışlardır. Bu şubeler, birbirlerine daha hızlı bir yerel ağ ile bağlıdır. Herbir işlemci ve veritabanı, yerel bilgisayar ağının bir birimini teşkil etmektedir. Bu uygulamanın diğerinden farkı coğrafi yapı üzerine yayılmamış olmasıdır. Yerel bir ağ yapısı, coğrafi ağ yapısına göre daha iyi performans göstermektedir. Coğrafi ağ yapısının on-line hareketlere cevap verecek düzeyde olması gerekir.



Şekil 3.2. Yerel ağ üzerinde dağıtık veritabanı.

Diğer bir örnek olarak, aynı banka Şekil 3.3'deki bilgisayar sistemine sahip olsun. Farklı şubelerin verileri, veritabanı yönetim fonksiyonlarını yöneten üç tane arka plandaki (back-end) bilgisayara dağılmıştır. Uygulama programları, ön plandaki (front-end) farklı bir bilgisayar ile icra edilmektedir. Bu makina, gerekli olduğunda arka plandaki bilgisayardan gelen veritabanı erişim servislerinde bulunmaktadır.

Bu sistem, dağıtık veritabanı özelliklerini içermediğinden, dağıtık bir veritabanı olarak nitelendirilmez. Ön plandaki bilgisayarda çıkabilecek bir sorun, tüm sistemi çalışamaz duruma getirir ve veri, fiziksel olarak farklı işlemciler üzerinden dağıtılmasına rağmen veri dağıtımı, uygulama ile ilgili değildir. Burada dağıtıklık için yerel uygulamanın eksikliği sözkonusudur.



Şekil 3.3. Çoklu işlemcili sistem.

Şimdi de dağıtık veritabanını istemci/sunucu yaklaşımı ile açıklayalım. Bundan önce bu kavramların ne olduğuna kısaca değinelim.

İstemci, ön plandaki (front-end) veritabanı uygulamasıdır ve kullanıcı ile klavye, ekran veya mouse arasındaki etkileşimi sağlar. İstemci kısmının veriye erişim sorumluluğu yoktur. İstemci, sunucu tarafından yönetilen işlemleri, istekleri toplar. Çok fazla disk kapasitesine ihtiyaç duymayabilir.

Sunucu kısmı, (back-end) veritabanı yönetim sistemi yazılımını çalıştırır, eş zamanlı ve paylaşımlı veri erişimi için gerekli fonksiyonları yönetir, istemci uygulamalarından kaynaklanan SQL programlarını alır ve işletir. Büyük disk kapasitesine ve hızlı işlemcilere sahip olmalıdır.

Sunucu/istemci mimarisi (dağıtık işlem), birbiri ile ilişkili işleri, işleme bölmek için birden çok işlemci kullanır. Dağıtık işlem böylece tek bir işlemciye yüklenen işlem yükünü azaltır. Farklı işlemcilere ait gruplardaki ilişkili işleri toplar ve tüm sistemin performansını artırır.

Sunucu/istemci mimarisi, uygulama işlemlerini ve sistem kaynaklarını farklı makinalar üzerine dağıtır. Bu dağıtım, iletişim teknolojisinin getirdiği olanaklarla donanım bağımsızlığı sağlayarak gerçekleşir.

Bütün bu tanımlardan sonra dağıtık veritabanını şu şekilde açıklayabiliriz; Kullanıcıya tek bir veritabanı gibi görünen, birden çok veritabanı sunucuları tarafından yönetilen veritabanı ağıdır. Dağıtık veritabanının bütün veritabanlarındaki veriye, eş zamanlı olarak erişilmeli ve değişiklik yapılabilmelidir. Fiziksel olarak ayrı veritabanları, mantıksal olarak tek bir parça gibi birleşir ve ağ üzerinden bütün kullanıcılar tarafından erişilebilir.

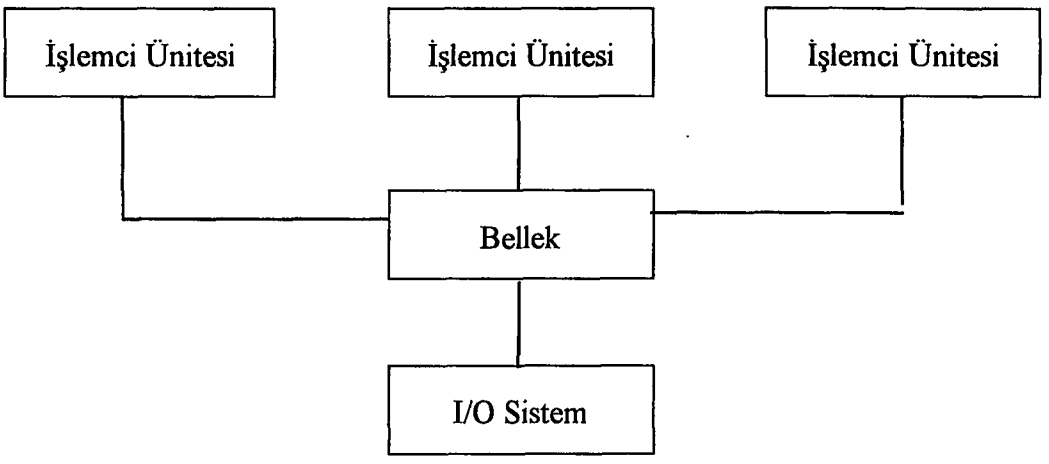
Kullanıcının direkt olarak bağlı olduğu veritabanına, yerel veritabanı, aynı kullanıcı tarafından erişilen diğer veritabanlarına da uzak (remote) veritabanı denir.

Bir veritabanını idare eden her bilgisayara bir düğüm denir. Yerel bir veritabanı bilgi için uzak veritabanına bağlandığında, yerel veritabanı uzak sunucunun istemcisidir.

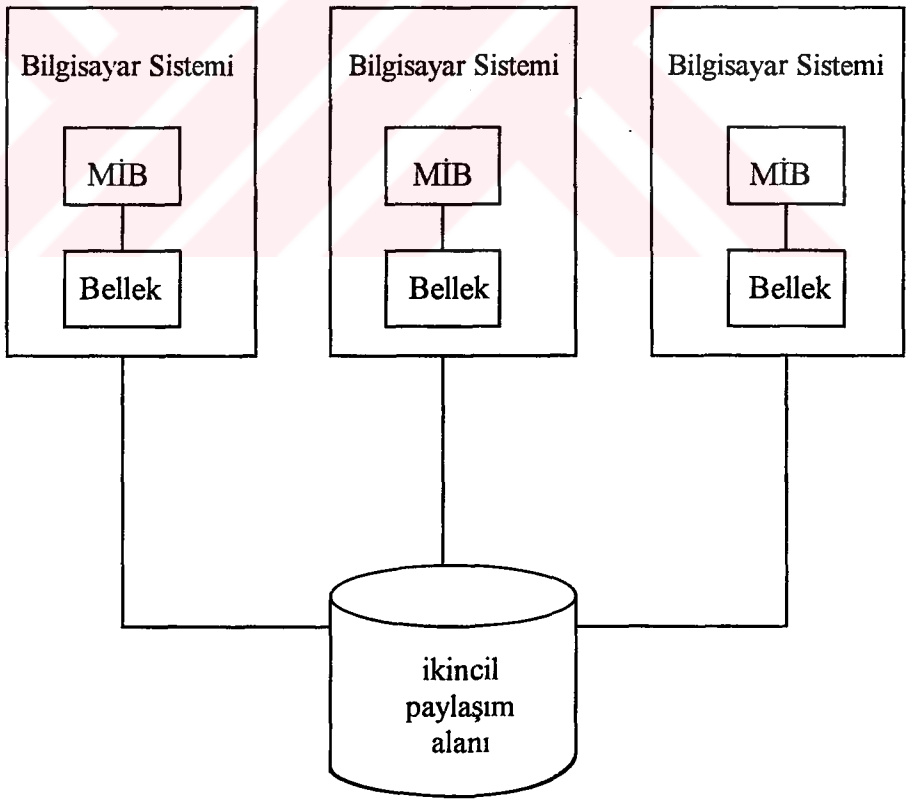
3.2. Dağıtık Veritabanı Yönetim Sistemi

Dağıtık veritabanı, bir bilgisayar ağı üzerine dağılmış, mantıksal olarak birbiri ile ilişkili birden çok veritabanı koleksiyonudur. Dağıtık veritabanı yönetim sistemi ise; dağıtıklığı kullanıcılara şeffaf yapan, veritabanını yönetmeye izin veren bir yazılım sistemidir [6]. Dağıtık veritabanı yönetim sistemi, bilgisayar ağının herbir düğümünde saklanan dosyaların koleksiyonu değildir. Dağıtık veritabanı tanımından yola çıkarsak, "mantıksal olarak ilişkili olmak" ve "bir bilgisayar ağı üzerinde dağıtılmışlık" kavramları çok önemlidir. DVTYS'nı oluşturan sadece dosyaların birbiri ile mantıksal ilişkili olması değildir. Aynı zamanda dosyalar arasında bir yapısal bağ bulunmalı ve bu dosyalara erişim herkese açık bir arayüz aracılığıyla olmalıdır. Verilerin fiziksel olarak dağıtıklığını önemseyenler, aynı bilgisayar sistemi üzerindeki birbiri ile ilişkili iki veritabanının dağıtık olduğunu iddia ediyorlar. Buradaki fiziksel dağıtıklık, bilgisayar sistemlerinin coğrafi olarak birbirlerinden çok uzak olmaları anlamına gelmiyor. Fakat verinin fiziksel dağıtıklığı önemli bir konudur. Veritabanlarının aynı bilgisayarda olması bazı problemler yaratabilir. Dikkat edilmesi gereken konu, veritabanları arasındaki haberleşmenin paylaşımlı bir bellek tarafından değil de bir ağ üzerinden sağlanmasıdır. Bütün bunlara ek olarak çok işlemcili sistemler, dağıtık veritabanı sistemi değildir. Çok işlemcili bir sistem, iki veya daha fazla işlemcinin belleği paylaştığı sistemdir. Bu birincil bellek ise buna sıkı bağlı (tighly coupled) (Şekil 3.4), ikincil bellek ise buna gevşek bağlı (loosely coupled) (Şekil 3.5) çok işlemcili denir. bellek paylaşımı, mesaj göndermeksizin işlemciler arasında iletişim sağlar.

Mikroişlemciler konusundaki gelişmeler sayesinde, bir yol verici (switch) ile birbirine bağlanan birçok mikroişlemciden meydana gelen çoklu işlemciler de ortaya çıkmıştır.

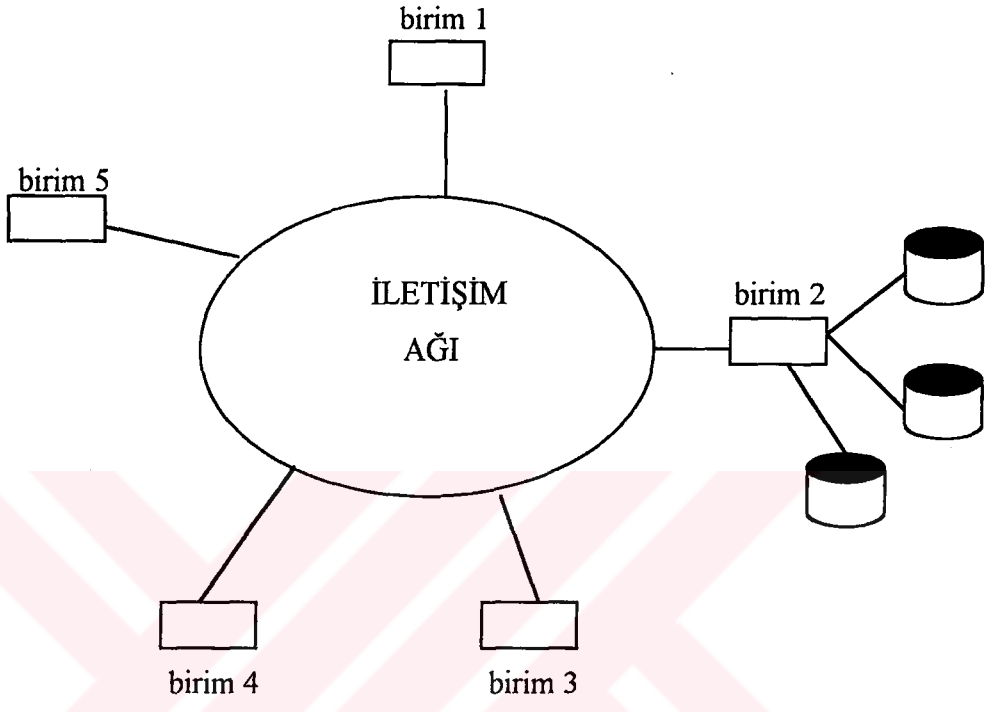


Şekil 3.4. Sıkı bağlı çok işlemcili.



Şekil 3.5. Gevşek bağlı çok işlemcili.

Buraya kadar anlatılanlara ek olarak, eğer ağ üzerinde sadece bir düğümde veritabanı varsa, bu dağıtık bir sistem değildir. Bu durumda merkezi veritabanı yönetim sisteminin problemleri yaşanır.



Şekil 3.6. Ağ üzerinde merkezi veritabanı.

Şekil 3.6'ya bakılırsa, veritabanı, bir bilgisayar sistemi tarafından yönetiliyor (birim 2) ve bütün istekler birim 2'ye gönderiliyor. Yani burada merkezi sisteme doğru bilgi akışı vardır. Sonuç olarak; bir bilgisayar ağının varlığı ve dosyaların koleksiyonu,bir dağıtık veritabanı oluşturmak için yeterli değildir.

Şimdi bir örnek verelim. Merkezi New York'ta, üretim fabrikaları Chicago ve Montreal'da, Avrupa'daki merkezi Paris'de, araştırma geliştirme bölümü San Francisco'da, stok depolarında da Köhln'de olan bir üretim firması düşünelim. Bu organizasyondaki yapılan her işe ait bilgileri tutmak gerekmektedir. Ulaşılacak veri ve bilgiler şu şekilde tutulabilir.

a) Her bir lokasyon, kendi lokasyonunda çalışanların kayıtlarını tutsun.

b) Araştırma geliştirme bölümünün aktiviteleri, bu aktivitelerin kullanılacağı yerlerde bulunsun.

c) Üretim fabrikaları, kendi mühendislik operasyonları ile ilgili verileri tutsunlar ve bu veriler araştırma grubunun ulaşabileceği şekilde düzenlensin.

d) Üretim fabrikaları kendi bilgilerini tutsun ve gerekli olduğunda stok depolarındaki bilgilere ulaşabilsinler.

e) Stok depoları kendi stok yönetimi sistemlerini kursun. Üretim yerlerine, bu stok depolarındaki bilgileri görmesi için izin verilsin.

f) Merkezi yerler, diğer bölgelerin pazarlama ve satış bilgilerini tutsun. Üretim fabrikaları ve stok depoları bu bilgilere ulaşabilsin ve veriler paylaşılabilsin.

Bu örnek, dağıtık veritabanı uygulaması için ideal bir örnektir. Bütün bilgiler, bir ağ ile bağlanmış farklı bilgisayarlarda bulunmaktadır.

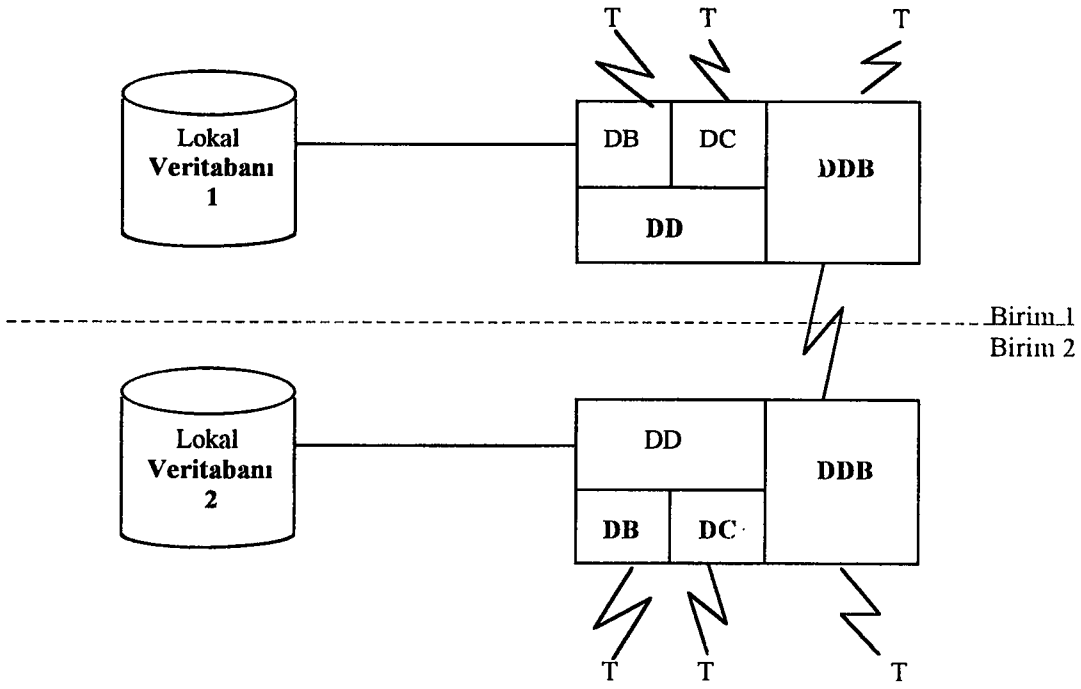
Birçok dağıtık sistem, merkezi veritabanı yönetim sisteminin satıcıları tarafından geliştirilmiştir ve ek bileşenler içerir [1]. Bu ek bileşenler, ağ üzerindeki farklı birimlere yerleştirilen VTYS'nin farklı instance'ları arasındaki iletişim ve işbirliği ile desteklenir. Bu bileşenler şunlardır;

- Veritabanı yönetim bileşeni (DB)
- Veri işletim bileşeni (DC)
- Ağ'daki dağıtık veri ile ilgili bilgi tutan veri sözlüğü (DD)
- Dağıtık veritaban bileşeni (DDB)

Bu dört bileşen (Şekil 3.7), dağıtık veritabanı yönetim sistemini oluşturur. Bu bileşenlerin sunduğu olanaklar şunlardır;

1) Uygulama programı ile uzak veritabanına ulaşım : Bu özellik çok önemlidir ve dağıtık veritabanı bileşeninin olduğu bütün sistemlerde sağlanır.

2) Dağıtık şeffaflık : Dağıtık şeffaflık ile performans açısından güçlü bir bağ olduğu için bu özellik farklı sistemlerde farklı yollarla sağlanır.



Şekil 3.7. DVTYS'nın bileşenleri.

3) Veritabanı yönetimi ve kontrolünün sağlanması. Bu özellik veritabanını gözlemlemek için gerekli araçları, veritabanı kullanım bilgilerini, farklı birimlerde varolan veri dosyalarının global bir görüntüsünü sağlar.

4) Dağıtık hareketin onarımı ve eş zamanlılık kontrolu için bazı olanaklar içerir.

3.3. Dağıtık Veritabanlarının Temel Prensipleri

3.3.1. Yerel Özerklik

Dağıtık sistemlerde her bir birim özerk olmalıdır. Her birimde yapılan operasyonlar kendi içinde kontrol edilir. Diğer bir birime bağımlı değildir, fakat bir birim diğer birimlerle çalışabilir [5].

Yerel verinin güvenliği, bütünlüğü ve saklama sorumlulukları tamamiyle yerel birimin kontrolü altındadır. Uzak (remote) birimler tarafından erişilse de, bütün veriler bir yerel veritabanına aittir.

Yerel özerkliğin şu yararları vardır;

- Yerel veri, yerel veritabanı yönetici tarafından kontrol edildiği için her bir veritabanının sorumluluğu azalır ve denetleme daha kolay olur.
- Ağ (network) ve bir veritabanı kullanılabilir olduğunda global veritabanı da kullanılabilir olacaktır. Yani bir veritabanındaki problem global veritabanını ve performansını etkilemez.
- Bir veri sözlüğü, her bir yerel veritabanında bulunur.
- Bir birim, yazılımdan bağımsız olarak güncelleştirilebilir (upgrade).

3.3.2. Merkezi Bir Sisteme Bağımsızlık

Yerel özerklik özelliklerinden anlaşılacağı gibi, bütün birimler, sistemde eşit haklara sahiptir. Birimler arasında ayrıcalık yoktur, dolayısıyla merkezi bir sisteme bağımlılık sözkonusu değildir. Bu kuralın gerekliliğini şu iki neden ile daha iyi anlayabiliriz. Birincisi, merkezi bağımlılık varsa bütün yük, merkez üzerinde olur ki bu durum darboğazı yaratacaktır. İkincisi ise merkez birimi, olabilecek bir aksaklıktan dolayı çökerse, diğer birimler çalışamaz durumda olacaktır.

3.3.3. Sürekli İşletim

Dağıtık bir sistemde, planlı sistem kapanmasına gerek duyulmamalıdır. Yeni bir birim eklemek veya versiyon değişikliği yapmak için olabilecek kapanmalar en alt düzeyde tutulmalı, işleyişi etkilememelidir.

3.3.4. Yer Bağımsızlığı

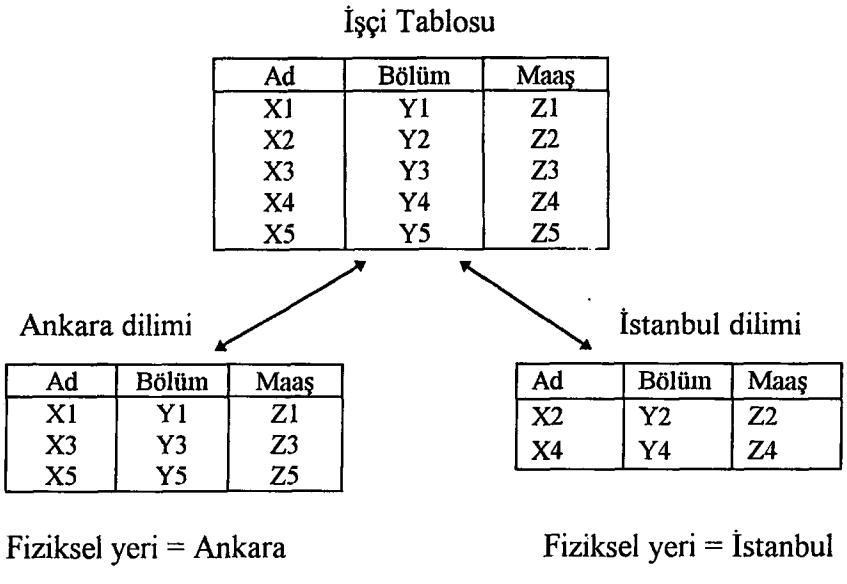
Kullanıcı, bir verinin bulunduğu yeri şu sebepten dolayı isteyebilir; veri, dağıtık veritabanının tek bir birimlerinde olabilir ve kullanıcı, bu veriye ulaşmak için bu birimi bilmek zorundadır. Fakat yer bağımsızlığı, bu problemi çözmektedir. Kullanıcı, fiziksel anlamda, verinin nerede olduğunu bilmeksizin sanki kendi yerel veritabanında bulunuyormuş gibi bu veriye ulaşabilir.

Yer bağımsızlığının diğer bir yararı ise, verilerin bir birimden diğerine taşınabilmesinin sağlanmasıdır. Bu taşınma yeteneği; verilerin gereksinimlere göre ağ içinde yer değiştirmesine imkan verir.

Yer bağımsızlığına örnek verirsek; İstanbul birimindeki bir kullanıcı, Bursa birimindeki veriye istemci/sunucu mantığıyla sanki kendi veritabanındaymış gibi ulaşabilir.

3.3.5. Dilimleme Bağımsızlığı

Dilimleme, bir tabloyu, satır alt kümeleri ya da sütun alt kümelerine bölme ve bu alt kümeleri farklı fiziki alanlara yerleştirme özelliğini ifade eder [7].



Şekil 3.8. Dilimleme örneği.

Dilimleme, performans nedeniyle gereklidir. Veri, en sık kullanılan yerde saklanabilir, böylelikle ağ trafiği azalır.

Şekil 3.8'de gördüğümüz gibi işçi tablosu, işçi bilgilerini tutsun. Bu tablodaki bilgiler, Ankara işçileri için Ankara veritabanında, İstanbul işçileri için, İstanbul veritabanında olacak şekilde dilimlenebilir.

Bir dilim, kesit alma (projection) ve sınırlama (restriction) işlemleri ile orjinal ilişkiden türetilebilir keyfi bir alt kümede olabilir. Yalnız kesit alma işlemi, orjinal kümenin ana (primary) anaktarını korumalıdır. Orjinal kümeyi yeniden oluşturmak ise birleştirme (union) ve kaynaştırma (join) işlemleriyle gerçekleşir.

Dilimlenme ve yeniden oluşturma kolaylığından dağıtık sistemlerin, neden ilişkisel olması gerektiğini daha iyi anlayabiliriz. İlişkisel model, bu görevlerin yerine gelmesi için gerekli işlemleri sağlar. Veri dilimlemeyi sağlayan bir sistem, dilimleme bağımsızlığını da sağlar. Kullanıcılar, veri hiç dilimlenmemiş gibi davranabilirler. Dilimlenme bağımsızlığı, kullanıcı programlarını basitleştirdiği için istenir bir özelliktir.

3.3.6. Kopyalama Bağımsızlığı

Kopyalama, fiziki konumdan bağımsız, veri tabanı yönetim sistemi tarafından otomatik olarak senkronize edilen farklı tablo kopyaları oluşturabilme özelliğini ifade eder. Şu iki sebepten dolayı istenir bir özelliktir:

- Daha iyi performans : Uygulamalarda, uzak birimlerle iletişim kurmak yerine yerel kopyalarla işlem yapılabilir.

- Kullanılabilirlik : Kopyalanmış bir nesne, en az bir kopya kullanılabilir kaldıkça, işlem için kullanılabilir olmaya devam eder.

Kopyalamanın dezavantajı ise kopyalanmış bir nesnenin güncelleştirilmesidir. Yapılan bir güncelleme, nesnenin bütün kopyalarına yansımalıdır.

Kopyalama, dilimleme gibi kullanıcı için şeffaf olmalıdır. Diğer bir deyişle, veri kopyalamayı destekleyen bir sistem, kopyalama bağımsızlığını da desteklemelidir. Kullanıcılar, veri hiç kopyalanmamış gibi davranabilmelidir. Bu özellik, dilimlenme özelliği gibi kullanıcı programlarını basitleştirdiği için istenir bir özelliktir.

3.3.7. Donanım Bağımsızlığı

Birden çok sistemdeki (IBM, DEC, UNISYS) veriler, birbirleri ile bütünleşmeli ve kullanıcıya tek bir sistem gibi görünmelidir. Yani, veritabanı yönetim sistemi, farklı işletim sistemlerinde çalışabilmeli, donanımdan bağımsız olmalı ve bu farklı donanım sistemleri dağıtık sisteme birer ortak olarak katılabilmelidir.

3.3.8. İşletim Sistemi Bağımsızlığı

Veritabanı yönetim sistemi, dağıtık ortamlar için işletim sisteminden bağımsız olarak çalışabilmelidir. Bu özellik, donanım bağımsızlığı ile alakalıdır. Aynı VTYS, aynı zamanda aynı donanım üzerinde farklı işletim sistemlerinde (DOS, VAX/VMS gibi) çalışabilmelidir.

3.3.9. Ağ Bağımsızlığı

VYTS, X.25, TCP/IP gibi haberleşme protokolları arasında ayırım yapmaksızın çalışabilmelidir. Sistem, hem yerel alan ağlarını hem de uzun bağlantılı hatları destekleyebilmelidir.

Sonuç olarak donanım, işletim sistemi ve ağ bağımsızlığının oluşturduğu kombinasyonu ele aldığımızda; bu öğelerden biri değiştiğinde. diğer bağlantı noktaları (node) ve çalışma birimleri (site) bundan hiç etkilenmemelidir.

3.3.10. VTYS Bağımsızlığı

Burada gerekli olan, farklı çalışma birimlerindeki VTYS'lerin aynı arabirimi (interface) desteklemesidir. Örneğin, ORACLE ve INGRES'ın her ikisi de SQL standardını desteklerse, dağıtık sistemde birbirleri ile haberleşebilirler. Sonuç olarak; birbirinden farklı VTYS'leri dağıtık sisteme bağlanabilmelidir. Diğer bir deyişle dağıtık sistem, VTYS bağımsızlığını sağlamalıdır.

3.3.11. Dağıtık Sorgu İşleme

Performans açısından, sorgu işleme, merkezi ve dağıtık sistemlerde önemli bir konu teşkil etmektedir. Dağıtık sorgunun performansını etkileyen birçok sayıdaki parametreden dolayı bu konu, dağıtık sistemde daha zordur.

Sorgu işleminin amacı; dağıtık sistemdeki yüksek seviyeli sorguyu, lokal veritabanındaki düşük seviyeli bir dille ifade edilen etkili bir icra stratejisine dönüştürmektir. Sorgu işleminin en önemli yönü, sorgu optimizeridir. Optimizer, strateji seçiminden sorumludur.

3.3.12. Dağıtık Hareket Yönetimi

Hareket yönetiminin iki temel özelliği vardır: onarım kontrolü ve eşzamanlılık kontrolü.

Eş zamanlılık kontrolü, dağıtık sistemlerde, dağıtık olmayan sistemlerde olduğu gibi kilitleme mantığına dayalıdır. Her birim, kendi yerel verilerinin kilitleme yönetiminden sorumludur. Yani, her birimde bir kilit yöneticisi vardır. Bunun sonucunda da genel bir sistem kilitlenmesi (deadlock) olasılığı sözkonusudur.

Hareketler, sistemde atomik olarak gerçekleşmelidir. Atomiklik, bir hareketin gerektirdiği işlemin ya hepsi yapılır ya da hiç biri yapılmaz şeklinde açıklanabilecek bir özelliktir.

Eğer bir hareket, bazı güncellemeler yaptıysa ve bir sonraki işlem, başarısızlıkla sonuçlanmışsa, bu güncellemelerin geri alınması gerekir. Eğer hareket, başarı ile sounçlanmış ise güncellemeler taahhüt edilir.

Taahhüt edilmemiş güncellemelerin geri alınması veritabanının önceki, doğru durumuna getirilmesidir ve buna onarım denir. Dağıtık sistemdeki bir hareket, bir çok birimde güncelleme yapabilir. Bu yüzden atomikliğin sağlanması için, yapılan işlemin ya hepsi birden taahhüt edilir ya da hepsi birden geri alınır. Bu daha sonra anlatacağımız iki aşamalı taahhüt (two phase commit) adı verilen protokol ile sağlanır.

3.4. Dağıtık Sistem ile Merkezi Sistemin Karşılaştırılması

Dağıtık sistemlerin tasarımı, merkezi sistemlere göre çok farklı özellikler göstermektedir [1]. Bir merkezi veritabanının özellikleri, merkezi kontrol, veri bağımsızlığı, tekrarın azaltılması, etkili erişim için karmaşık fiziksel yapılar, bütünlük, onarım, gizlilik, güvenlik ve eş anlılık kontrolüdür. Şimdi bu özellikleri sırasıyla dağıtık veritabanı özellikleri ile karşılaştırarak kısaca değinelim.

Merkezi Kontrol

Veritabanı yöneticisinin temel fonksiyonu, veri güvenliğini sağlamaktır. Veri, merkezi bir sorumluluk altındadır.

Dağıtık veritabanlarında, merkezi kontrol mantığı daha az işlenmektedir. Burada global veritabanı yönetici ve lokal veritabanı yönetici baz alınarak bir hiyerarşik kontrol yapısı sözkonusudur. Global veritabanı yöneticisi, bütün veritabanından, lokal veritabanı yönetici kendi lokal veritabanından sorumludur. Lokal veritabanı yöneticileri yüksek seviyede özerk olmalıdır.

Veri Bağımsızlığı

Veri bağımsızlığının anlamı, verinin uygulamacı için şeffaf olmasıdır ve en büyük avantajı programların verinin fiziksel organizasyonundaki değişikliğinden etkilenmemesidir. Dağıtık veritabanında veri bağımsızlığının önemi ise geleneksel veritabanlarındakiyle aynıdır. Burada ek olarak "dağıtık şeffaflık" kavramı ortaya çıkmaktadır. Programlar veritabanı dağıtık değilmiş gibi yazılır ve verinin birimden başka bir birime taşınmasından etkilenmez fakat bu durum, çalışma hızını etkiler.

Veri Tekrarının Azaltılması

Geleneksel veritabanlarında veri tekrarı iki sebepten dolayı aza indirgenir. Birinci sebep, aynı verinin birden çok kopyası yüzünden oluşacak veri tutarsızlığından kaçınmak için, ikinci sebep ise fazlalıklar gözardı edilerek verinin saklandığı alandan tasarruf etmek içindir. Veri tekrarının azaltılması, verinin paylaşılması ile sağlanır.

Dağıtık veritabanlarında ise veri tekrarı istenen bir olaydır. Bunun nedeni ise;

- Veri, bütün birimlerde saklanırsa, uygulamanın lokalliği artacaktır.
- Sistemin uygunluğu artacaktır. Çünkü, veri kopyalanmışsa, bir birimdeki aksaklık aynı veriye sahip diğer birimin çalışmasını etkileyemeyecek; dolayısıyla bu aksaklık sisteme yansımayacaktır.

Etkin Erişim ve Karmaşık Fiziksel Yapılar

İkincil index, ara dosya zincirleri gibi karmaşık erişim yapıları, geleneksel veritabanının en önemli özellikleridir. Bu yapıların amacı; veriye etkin bir erişim sağlamaktır.

Dağıtık veritabanlarında ise etkin erişim için bu araçlar kullanılmaz. Bu yapıların inşa edilmesi zordur. Çünkü dağıtık veritabanlarında, bu yapıların kayıt seviyesine taşınması uygun değildir.

Dağıtık bir erişim planı, programcı ya da otomatik olarak optimizer tarafından yapılmalıdır. Optimizer ise bazı problemleri çözmek durumundadır. Bu problemler global optimizasyon ve lokal optimizasyon olarak ikiye ayrılır.

Global optimizasyon: Hangi birimdeki hangi veriye ulaşılacağının ve birimler arasında transfer edilecek veri dosyalarının belirlenmesidir.

Yerel optimizasyon: Her bir birimdeki lokal veritabanına erişimin nasıl gerçekleşeceğinin belirlenmesidir.

Bütünlük, Onarım ve Eş zamanlılık Kontrolü

Bir veritabanının bütünlüğü, onarımı ve eş anlılık kontrolü birbiri ile çok bağlantılı kavramlardır. Ortaya çıkacak problemlerin çözümü, hareketler ile sağlanır. Veritabanı bütünlüğünün sağlanması hareketlerin atomik olması ile gerçekleşir. Bununla hareketin gerektirdiği işlem ya yapılır ya yapılmaz. Sistemdeki herhangi bir bozukluk veya eş anlılık, atomik hareketin iki düşmanıdır. Hareket icrası sırasında sistemde bir bozukluk, farklı hareketlerin eş anlı çalışması birbirini beklemesine veya geriye dönüşe (rollback) neden olur. Onarım, bozukluk durumlarında hareketin atomiklik problemi ile ilgilidir.

Eş zamanlılık kontrolü ise, hareketlerin eş zamanlı çalışması durumunda hareketin atomik olmasını sağlar.

Gizlilik ve Güvenlilik

Geleneksel veritabanlarında, veritabanı yöneticisi merkezi kontrole sahiptir ve veriye sadece yetkili olan kullanıcıların ulaşmasını sağlar.

Dağıtık veritabanlarındaki lokal veritabanı yöneticileri ise hemen hemen aynı problemlerle karşı karşıyadır.

Fakat dağıtık veritabanının iki özel durumu vardır;

1- Dağıtık veritabanının yüksek seviyedeki yerel özerkliği ile, verinin yerel sahipleri, veriyi daha iyi korurlar.

2- Haberleşme ağının, korumaya karşı cevap vermedeki zayıflığı yüzünden güvenlik, dağıtık veritabanında önemli bir sorun teşkil etmektedir.

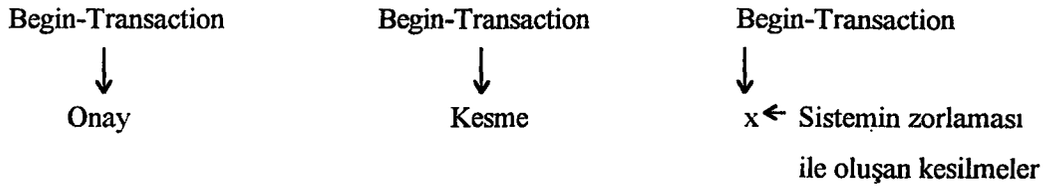
3.5. Dağıtık Hareket Yönetimi

Dağıtık hareket yönetimi güvenilirlik, sistem kaynaklarından faydalanma gibi birbirlerine bağlı pek çok problemle ilgilidir. Dağıtık hareket yönetimini anlamak için eş zamanlılık kontrolü, kurtarma mekanizması ve hepsinden önemlisi sistemin yapısı arasındaki ilişkiyi anlamak gereklidir. Literatürde önerilen tekniklerin çoğu gerçek sistemlerde geçmediğinden performansın nasıl olacağı belirsizdir. Ticari ve araştırma sistemlerinde en çok kullanılan 2 teknik vardır. Bunlar kurtarma için iki aşamalı taahhüt ile eş zamanlılık kontrolü için 2 aşamalı kilit mekanizmalarıdır [1].

3.5.1. Hareket Özellikleri

Bir hareket şu özellikleri sağlayan program ya da program parçasıdır.

Bölünmezlik : Hareket işlemlerinin ya tümü yapılır ya da hiçbiri yapılmaz. Tamamlanmamış bir hareketin 2 sebebi olabilir: Hareketin kesilmesi ve sistemin çökmesi. Hareket, hareketin ya da kullanıcının isteğiyle kesilebilir. Bunun sebebi ya girdilerin hatalı olması ya da hareketin tamamlanması için gerekli koşulların sağlanamamasıdır. Bunun yanında ölümcül kilitlenme (deadlock) ve sistemin aşırı yüklenmesi gibi sisteme bağlı sorunlar yüzünden de hareket kesilmiş olabilir. Hareketin tamamlanmasına onaylama (commitment) denir. Her hareket begin-transaction ile başlayıp onay (commit) ya da kesme (abort) ile son bulur (Şekil 3.9).



Şekil 3.9. Hareket kesilme şekilleri.

Süreklilik : Bir hareket onaylandığında, sistem bu işlemlerin sonuçlarının kaybolmamasını sağlamalıdır. Bir hareketin sistem tarafından saklanan sonuçları veritabanında saklanır ki hareketlerin sürekliliğinin sağlanması veritabanı kurtarma mekanizması olarak bilinir.

Ardışıklık : Eğer aynı anda birden çok hareket çalıştırılırsa sonuç sanki hareketler seri olarak sıralı çalıştırılmış gibi olmalıdır ki, bu da eş zamanlılık kontrolü olarak adlandırılır.

İzolasyon : Tamamlanmamış bir hareket sonuçlarını, onayı bitmeden diğer hareketlere göstermez. Bu özellik arka arkaya oluşabilecek problemlerden kaçınmak için gereklidir.

Hareket : Hareketlerin verimliliği, güvenilirliği, eş zamanlı çalışması hareket yönetimin temel amaçlarıdır.

CPU ve Bellek Kullanımı: Dağıtık ve merkezi sistemler için bu kullanım aynıdır. Büyük sistemlerde veritabanı uygulamalarında zamanın çoğunun I/O işlemleri için kullanılması ve bu tür işlemlerin eş zamanlı çalıştırılması CPU ve bellek'de darboğaza yol açar.

Eğer işletim sistemi, her bir aktif hareket için bir proses yaratmak zorundaysa, bu proseslerin çoğu taşmaya neden olacaktır (swapp) ve bu da bellek yetersizliğine yol açacaktır. Bunu azaltmak için hareket yöneticisi tarafından veritabanı uygulamalarının tipik özelliklerini koruyan belirli teknikler kullanılır.

Kontrol Mesajları : Dağıtık bir veritabanında birimler arasında gidip gelen kontrol mesajlarının sayısı verimliliği etkiler. Kontrol mesajları verileri transfer etmek için kullanılmaz, bunlar uygulamanın çalışmasını kontrol etmek için gereklidir. Bir mesajın maliyeti yalnızca iletim maliyetine değil aynı zamanda CPU'nun boş yere kullanılmasına neden olur.

Cevap Süresi: Sistemin verimli olabilmesi için her bir hareket için cevap süresi önemlidir. Farklı birimler arasında haberleşme için ek zaman gerekeceğinden dağıtık uygulamalarda cevap süresi, yerel uygulamalara göre daha önemli bir yer tutar.

Kullanılabilirlik : Dağıtık veritabanlarında hareket yönetiminin diğer bir önemli konusu, bütün sistemin kullanıbilirliğidir. Dağıtık bir sistemde, bir birimdeki bozukluk tüm sistemin işleyişini etkilememelidir.

Dağıtık veritabanındaki eş zamanlılık ve kurtarma mekanizması için kullanılan algoritmalar, bu tür birim bozuklukları durumunda tüm sistemin kullanılabilirliğini arttırmalıdır.

Dağıtık sistemdeki hareket yönetiminin amacı, hareketlerin çalışmasını kontrol etmektir. Böylece;

1) Hareketler bütünlük, süreklilik, ardışıklık ve izolasyon özelliklerine sahip olur.

2) Kullanılan CPU ve bellek miktarı, iletilen kontrol mesajlarının sayısı ve bunların cevap süresi en aza indirgenir.

3) Sistemin kullanılabilirliği maksimum noktadır.

3.5.2. Dağıtık Hareket

Bir hareket uygulamanın bir parçasıdır. Bir uygulama begin-transaction ile başlar ve bu işlemler onaylanana kadar veya kesilene kadar devam eder. Farklı

birimlerdeki işlemleri çalıştırmak için, dağıtık bir uygulama bu farklı birimlerdeki prosesleri çalıştırmak zorundadır. Bu proseslere uygulama aracı (agent) denir ve bunlar yapılan yerel işlemlerdir.

Uygulamanın ihtiyacı olan global işlemlerin icrası için, araçlar iletişim kurmak zorundadır. Bunlar farklı uçlarda bulundukları için, araçlar arasındaki iletişim mesajlarla sağlanır.

Birlikte çalışan proseslerin yapısını oluşturmak için araçlar farklı şekillerde organize edilebilir.

1) Hareketin başladığı bir ana (root) araç mevcuttur. Böylece bir kullanıcı uygulama icrası için istekte bulunduğunda ana araç başlar. Ana aracın bulundğu birim, hareketin başlangıç birimi olarak bilinir.

2) Ana aracın begin-transaction, işleminin onayı veya kesilmesi gibi temel sorumlulukları vardır.

3) Yeni bir aracın yaratılması sadece ana araç tarafından yapılır.

Dağıtık Harekete Örnek

Bir ağın farklı uçlarına dağılmış, banka şubeleri gibi, hesapları ele alalım.

Şekil 3.10 , iki aracı göstermektedir. Bunlardan biri ana araçtır. Ana aracın bulunduğu birimde paranın bulunduğu hesap, diğer birimde de paranın aktarılacağı hesap bulunur ki bu birimde $AGENT_1$ aracı icra edilir. Ana araç, $AGENT_1$ çalıştırmak istediğinde önce $AGENT_1$'ı yaratır, sonra parametre gönderir. $AGENT_1$'ın yaratılması, uzak bir prosesi yaratmak için sisteme bir istek gönderilmesi, parametre gönderimi ise iki araç arasındaki haberleşmedir.

Ana araç, SEND komutunu icra ettikten sonra eğer $AGENT_1$ aracında bir bozukluk olursa, hareket bir bütün olarak kesilmelidir. Yani ana aracın bulunduğu birimdeki hesapta yapılan güncelleştirme işlemi geri alınmalıdır.

```
ROOT-AGENT:
Read (terminal,$AMOUNT, $FROM-ACC, $TO-ACC);
Begin-transaction;
Select AMOUNT into $FROM-$AMOUNT
from ACCOUNT
where ACCOUNT-NUMBER=$FROM-ACCOUNT;
If SFROM-AMOUNT-$AMOUNT<0 then abort
else begin
        Update ACCOUNT
        set AMOUNT=AMOUNT-$AMOUNT
        where ACCOUNT=$FROM-ACC;
        Create AGENT1;
        Send to AGENT1 ($AMOUNT,$TO-ACC);
        Commit
    end
AGENT1:
Receive from ROOT-AGENT($AMOUNT, $TO-ACC);
Update ACCOUNT
set AMOUNT=AMOUNT+$AMOUNT
where ACCOUNT=$TO-ACC;
```

Şekil 3.10. İki araç tarafından işletilen FUND-TRANSFER hareketi.

3.5.3. Dağıtık Hareketteki Atomikliğin Sağlanması

Begin-transaction, onay ve işlerin kesilmesi gibi global temel işlemler, dağıtık hareketin çalıştığı birimlerdeki uzun lokal işlemler grubunu çalıştırmasıyla sağlanır. Dağıtık hareket için yukarıdaki temel işlemlerin çalıştırılmasını sağlayan dağıtık hareket yönetimini kurabilmek için, her birimde lokal işlemleri yapabilen lokal hareket yöneticisine gerek vardır.

3.5.4. Dağıtık Hareket Onarımı

Her araç, kendi lokal hareket yöneticisi tarafından begin-transaction, onay ve işlem kesilmesi özelliklerini yerine getirir. Begin-transaction başlatıldıktan sonra bir araç, lokal bir hareket özelliklerine sahip olacaktır.

Bir araç, kendini yöneten bir hareket olmadığı için, begin-transaction komutunu lokal hareket yöneticisine alt hareket olarak gönderir.

Dağıtık hareketin begin-transaction, onay ve kesilme işlevlerini lokal işlevlerden ayırt etmek için local-begin, local-commit ve local-abort kullanılır.

Bir dağıtık hareket yöneticisini kurmak için lokal hareket yöneticisi (LTM) şu özelliklere sahip olmalıdır:

1) Alt hareketin bütünlüğünün sağlanması,
2) Dağıtık hareket yöneticisinde bazı kayıtların tutulması.

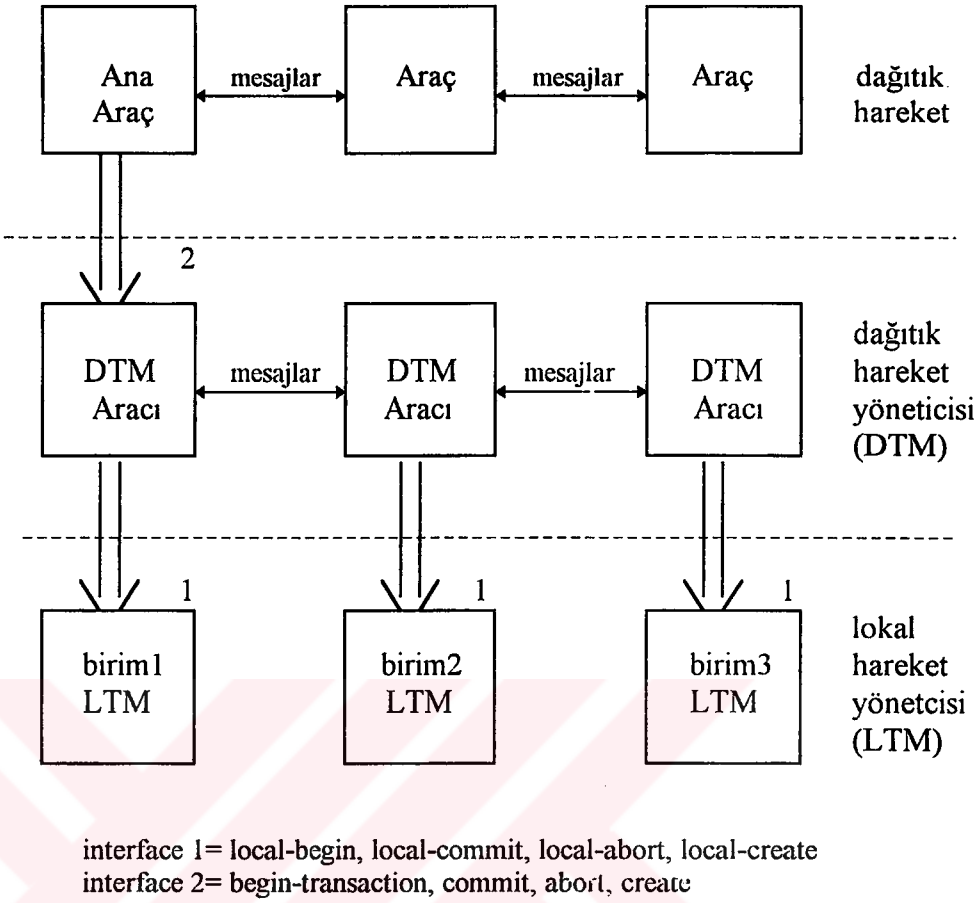
Bu özellik çok önemlidir. Bir bozukluk durumuna karşı bazı ek bilgiler tutulmalıdır.

Dağıtık bir hareketi icra eden bütün araçlar, lokal alt hareketlerdir. Bunlar dağıtık hareketin özelliklerini sağlamaya yetmez. Diğer bir deyişle, alt hareketler için bütünlüğü sağlayan lokal hareket yöneticileri tek başlarına, dağıtık seviyedeki bütünlüğü sağlamak için yeterli değildir. Bunun sebebi basittir. Dağıtık bir hareketin bütün işlemlerinin yapıldığı ya da hiçbirinin yapılmadığından emin olmak için iki şart gereklidir.

1) Her bir birimde, bütün işlemler ya yapılır ya da yapılmaz.
2) Bir althareketin onay edilmesi veya edilmemesi kararı tüm birimde aynı olmalıdır.

Şekil 3.11'de lokal hareket yönetimi ile dağıtık hareket yönetimi arasındaki ilişki gösterilmiştir. En alt seviyede aralarında iletişime gerek duyulmayan lokal hareket yöneticileri bulunur. Buna göre;

1) Lokal hareket yöneticilerinin içeriği local-begin, local-commit, local-abort arabirimleridir.



Şekil 3.11. Lokal hareket yöntemi ile dağıtık hareket yöntemi arasındaki ilişki.

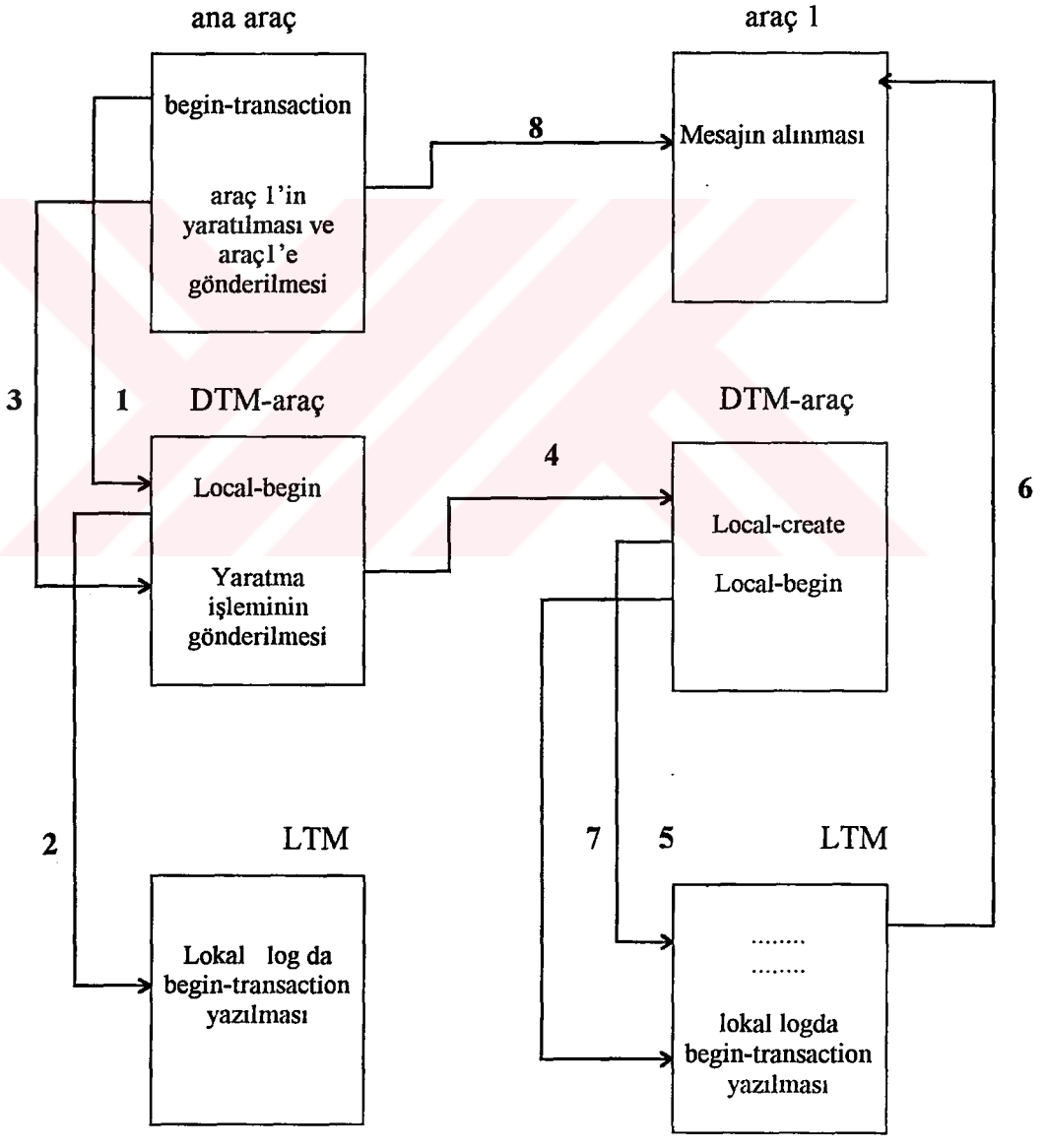
2) Lokal hareket yöneticilerinin arabirimleri local-begin, local-commit, local-abort'a ek olarak local-create'de bu arabirimin bir parçasıdır. Çünkü hareket özellikleri ile ilgili olmasa da bir proses (agent)'ın yaratılması lokal sistemlerin bir fonksiyonudur. Bu üst seviyede dağıtık hareket yöneticisi (DTM) bulunur. DTM'ı bir dağıtık katman olarak düşünürsek, dağıtık hareket yöneticisi aralarında mesajların gidip geldiği bir grup lokal dağıtık hareket yöneticilerindeki araçlar tarafından sağlanacaktır. Dağıtık hareket yöneticisi arabirimleri begin-transaction, onay (commit), kesme (abort), ve yaratma (create)' dan oluşur.

3) Bir üst seviyede, ana araç ve diğer araçlardan oluşan dağıtık hareket vardır. Yalnızca ana araç, begin-transaction, onay ve kesme işlemlerini yapabildiğinden interface (2), sadece ana araç tarafından kullanılır.

Bu örnek, dağıtık bir hareketin gerçekleşme (run-time) organizasyonu olarak düşünülmemelidir. Birçok sistemler, bu katmanların çalışma sırasını sağlamazlar.

BEGIN-TRANSACTION

Ana araç tarafından bir begin-transaction başlatıldığında DTM hem orjinal birimindeki LTM'ye hem de aynı uygulamanın diğer aktif araçlarının bulunduğu tüm birimlere local-begin göndermek zorundadır.



Şekil 3.12. FUND-TRANSFER hareketinin mesaj ve işlem yapısı.

Aynı dağıtık hareket içinde yeni bir aracın başlatılması ile tüm araçların alt hareketlere dönüşmesi, aracın, aktif olduğu LTM'ye local-begin göndermesini gerektirir. Böylece yeni araç, alt hareket olarak yaratılır.

Daha önce anlattığımız Şekil 3.10'deki FUND-TRANSFER hareketi ana araçtan araç 1'e parametre gönderiyordu. Şekil 3.12'de gönderilen mesajların sayısı yapılan işlem sırasını göstermektedir.

ABORT

Ana araç tarafından bir kesme işlemi yapıldığında, var olan bütün alt hareketler kesilmelidir. Bu işlem, alt hareketin aktif olduğu bütün birimlerdeki LTM'lere local-abort gönderilerek sağlanır.

COMMIT

Onay işlemi en zor ve pahalı olanıdır. Dağıtık bir hareketin doğru olarak onaylanması, lokal bozukluk durumunda bile tüm alt hareketin onaylanmasını gerektirir. Bu yüzden de diğer alt hareketlerin onaylanması mümkün değildir. Dağıtık hareketler için 2 aşamalı taahhüt geliştirilmiştir.

3.5.5. İki Aşamalı Taahhüt

Bu protokol'un temelinde, özel bir rolü olan bir araç vardır ki bizim örneğimiz de DTM-araçtır. Bu araca koordinatör denir ve onaylama işlemini yapan diğer bütün araçlara katılımcı (participant) denir. Koordinatörün görevi, sonuçtaki onay veya kesme kararını vermektir. Her katılımcının görevi, kendi lokal veritabanına yazma işlemi yapmaktır. Her bir katılımcının farklı bir birimde olduğu varsayılmıştır. Bir hareket, koordinatörün bulunduğu uçta bir yazma işlemi yaptığında (koordinatör ve bir katılımcı aynı birimde) ağ kullanarak iletişime gerek duymadan sanki farklı birimdeymiş gibi protokolu takip ederler.

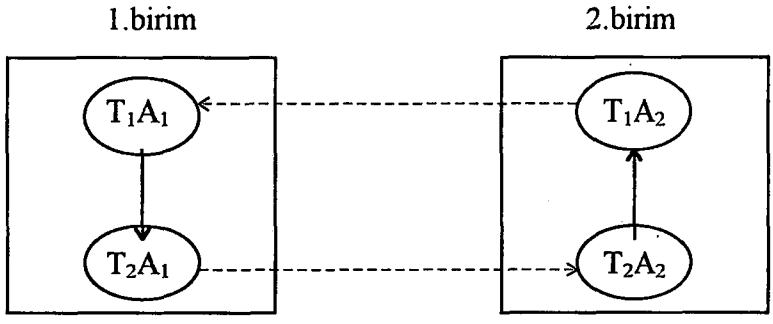
2 aşamalı taahhütün temel fikri, bütün katılımcılar için, tüm lokal alt hareketlerin onaylanması veya kesilmesi gibi tek bir karar vermektir. Eğer bir katılımcı, lokal olarak alt hareketlerini onaylayamıyorsa, tüm katılımcılar lokal olarak onaylanmaz.

Bu protokol, iki aşamadan oluşur. Birinci aşamasının amacı; ortak bir karara varmaktır. İkinci aşaması, bu kararın uygulanmasıdır.

3.5.6. Dağıtık Ölümcül Kilitlenme

Ölümcül kilitlenme (deadlock), merkezi veritabanlarına göre dağıtık veritabanlarında daha büyük problem teşkil eder. Çünkü ölümcül kilitlenmeyi belirleyen sonsuz bekleme durumu, sadece bir birimde değil, birden çok birimde meydana gelir.

Şekil 3.13, ölümcül kilitlenme durumunu ve bir döngü içeren dağıtık bekleme (wait-for) grafiğini gösteriyor. Buradaki A_j, araca, T_i'de harekete karşılık gelmektedir. Bu şekilde iki birim ve herbiri iki araçtan oluşan iki hareket vardır. Yani her bir hareketin çalıştıkları birimde bir aracı vardır.



Şekil 3.13. Dağıtık ölümcül kilitlenmeyi gösteren dağıtık bekleme grafiği.

T_1A_1'den T_2A_1' ye giden yönün anlamı, T_1A_1'ın bloke olması ve T_2A_1'ın beklemede olmasıdır.

Bir aracın diğerini beklemesinin iki sebebi vardır:

1) Araç T_1A_1 gereksinim duyduğu kaynağı serbest bırakması için T_2A_1 'yı bekler. T_1 , T_2 'den farklı bir harekettir.

2) Gerekli fonksiyonları çalıştırmak için T_1A_2, T_1A_1'yı bekler. Bu iki araç aynı harekete aittir. Fakat farklı birimlerde bulunurlar.

Ölümcül kilitlenmenin çözülmesi, kesilecek ve yeniden başlatılacak bir veya daha fazla hareket seçiminden ibarettir. Bir hareket kesildiğinde, ilgili kaynağı serbest bırakır ve böylece diğer hareket buna ulaşır. Kesilecek hareketin seçiminde kullanılan kriter, bu işlemin maliyetini minimize etmelidir. Bu kriteri şöyle açıklayabiliriz; en son hareket ve en az kaynağı olan hareket, kesme maliyeti en az olan hareket, tamamlanması uzun sürecek hareket kesilir.

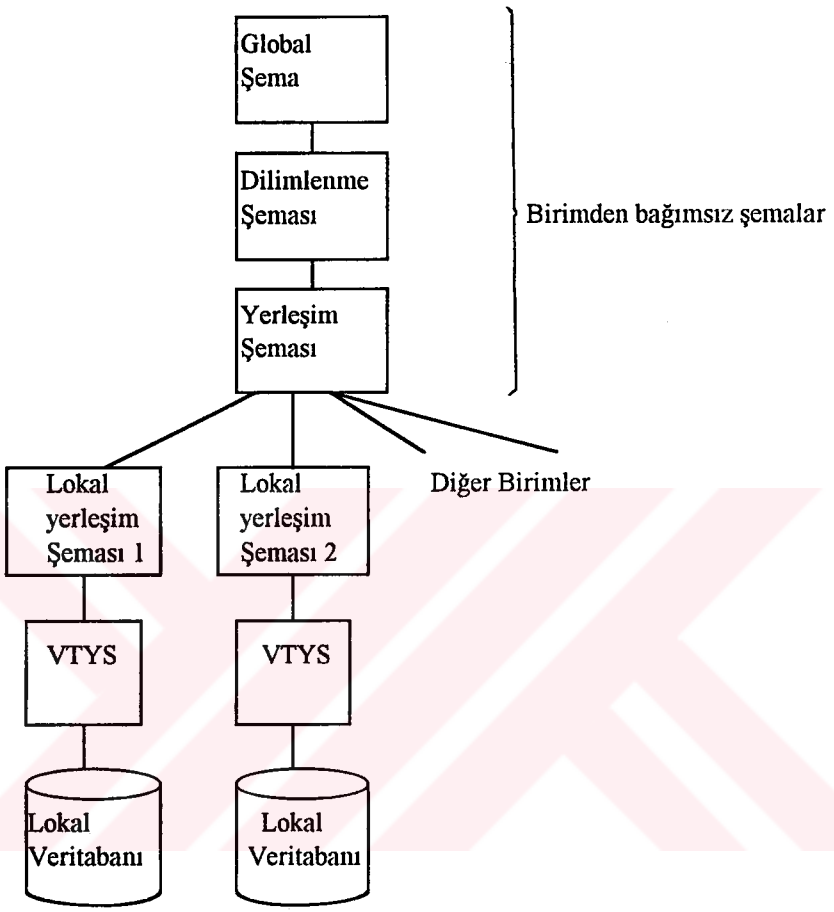
Dağıtık veritabanındaki gereksiz işlemler, ölümcül kilitlenmeye düşme olasılığını arttırır. Örneğin T_1 ve T_2 iki hareket olsun ve her ikisi de aynı x verisini kilitlemiş olsun. Eğer x'in bir kopyası yoksa, bir hareket veriyi kilitler, işlevini yapar ve bu sırada diğeri bekler. Eğer x'in farklı uçlarda kopyaları varsa, aşağıdaki sırada 1 ve 2. birimlerde ölümcül kilitlenme oluşur.

1. birim = T_1, x_1'i kilitler, T_2 bekler.
2. birim = T_2, x_2'yi kilitler, T_1 bekler.

Şimdide ölümcül kilitlenmeyi önleme yollarından birini inceleyelim. Eğer ölümcül kilitlenmenin meydana gelme riski varsa bir hareket kesilir ve yeniden başlatılır.

Bu işlem şu şekilde yapılır: Eğer T_1 hareketi, T_2 tarafından tutulan bir kayda ulaşmak istiyorsa, "prevention test" uygulanır. Bu test sonucunda bir ölümcül kilitlenme riski varsa ortaya çıkar ve o zaman, T_1'in beklemesine izin verilmez. Bunun yerine ya T_1 kesilir ve yeniden başlatılır ya da T_2 kesilir ve yeniden başlatılır.

3.6. Dağıtık Veritabanında Şeffaflık



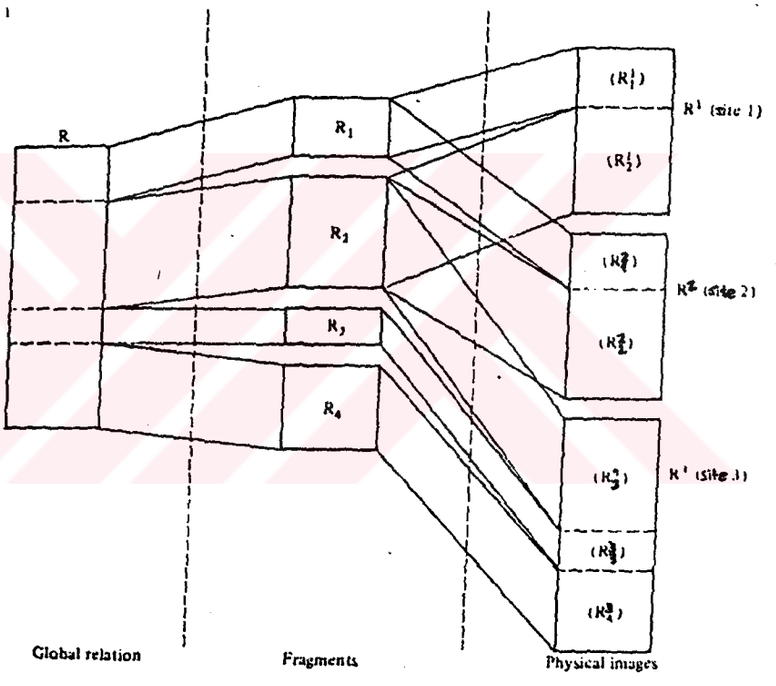
Şekil 3.14. Dağıtık veritabanı için referans mimarisi.

Şekil 3.14 dağıtık vertabanı için referans bir mimari göstermektedir. Bu mimari bütün dağıtık veritabanları tarafından sağlanmaz fakat herhangi bir dağıtık veritabanı organizasyonunu anlamak için gereklidir. Dağıtık veritabanı şeffaflığı bu mimari üzerinde anlatılacaktır.

Global şema, dağıtık veritabanındaki bütün verileri tanımlar. Bir global şema, dağıtık olmayan veritabanında da benzer yolla tanımlanır. Global şema, bir grup global ilişki tanımından oluşur.

Her bir global şema, dilim (fragment) denilen bir kaç kesişen parçalara ayrılabilir. Global ilişkiler ile dilimler arasındaki yapı, dilimlenme (fragmentation) şemasında tanımlanır. Birçok dilim, bir global ilişkiye aittir, fakat bir global ilişki, sadece bir dilime aittir.

Dilimler, ağ üzerindeki bir veya daha fazla birimde fiziksel olarak yerleştirilmiş global ilişkilerin mantıksal parçalarıdır. Yerleşim şeması, dilimin bulunduğu birimleri tanımlar.



Şekil 3.15. Global ilişkinin dilimleri ve fiziksel görüntüleri.

Şekil 3.15 'ye göre R global ilişkisi, R_1, R_2, R_3, R_4 olmak üzere 4 tane dilime ayrılmıştır. Bu dilimlerin fiziksel görüntüleri (image) R^1, R^2, R^3, ağ üzerindeki üç birimde bulunmaktadır. R_2^3, 3.birimde bulunan R_2 diliminin kopyasını göstermektedir. Sonuçta, bir fiziksel görüntü, diğerinin kopyasıdır. R^1, R^2'nın kopyasıdır.

Global şema, dilimlenme şeması ve yerleşim şeması, birimden bağımsızdır. Lokal veritabanı yönetim sisteminin veri modeline bağlı değildirler.

Daha aşağıdaki seviyede, lokal veritabanı yönetim sistemi tarafından işletilen nesnelerin fiziksel görüntülerinin planını yapmak gereklidir.

Bu plana, lokal yerleşim şeması denir ve lokal veritabanı yönetim sisteminin tipine bağlıdır. Heterojen bir sistemde, farklı birimlerde farklı lokal yerleşim tipleri bulunur.

Bu mimarinin özelliklerine ait üç tane önemli konu vardır:

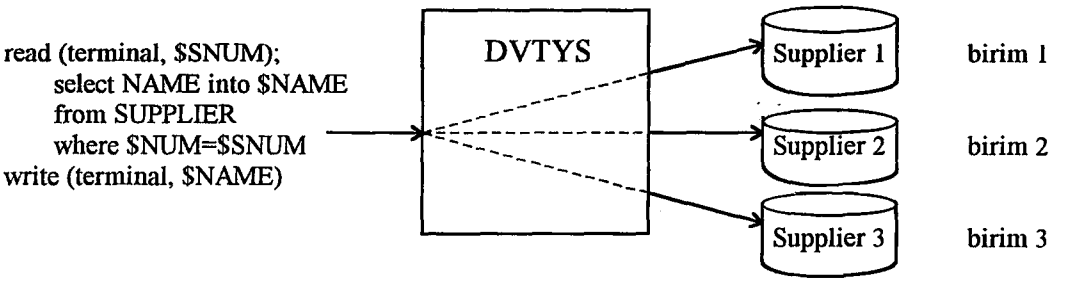
1) Veri dilimlenmesinin veri yerleşiminden ayrılması.

Bu ayrım, bize dağıtık şeffaflığın iki farklı seviyesini anlamamızı sağlar. Bu şeffaflıklar, dilimleme ve yer şeffaflıklarıdır.

- Dilimleme Şeffaflığı

Her bir veritabanı ilişkisini küçük parçalara bölmek ve her bir parçayı ayrı bir veritabanı nesnesi gibi görmek istenir bir özelliktir. Dilimleme, kopyalamanın negatif etkilerini azaltır.

Eğer bir dağıtık veritabanı yönetim sistemi dilimleme şeffaflığını sağlarsa, veritabanına erişim Şekil 3.16'da gösterildiği gibi gerçekleşir.



Şekil 3.16. Dilimleme şeffaflığı.

Kullanıcı, terminalden bir satıcı (supplier) numarası girerek veritabanına ulaşmaktadır. Bu SQL komutu, basit dağıtık veritabanı erişim ilkesini temsil

etmektedir. $SNUM girişi parametresini, $NAME çıkış parametresini göstermektedir. Dağıtık veritabanı yönetim sistemi, kendisi tarafından karar verilen bir yolla, bu üç birimden birinin veritabanına ulaşarak komutu yorumlamaktadır.

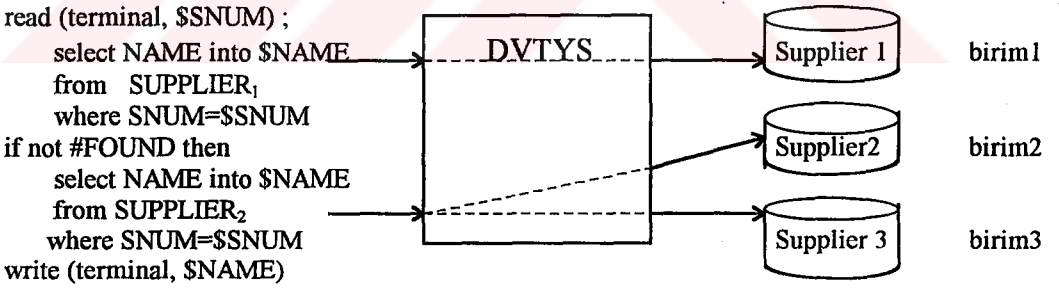
Dağıtık şeffaflık bakış açısıyla, uygulama veritabanı dağıtık değilmiş gibi davranan global ilişki ismi, Supplier1'e karşılık gelmektedir.

Bu yolla, uygulama, referans mimarinin bütün şemalarda uygulanacak değişikliklere karşı tamamıyle muaftır.

- Yer Şeffaflığı

Yer şeffaflığında komut, verinin bulunduğu yerden ve işlemin yapıldığı sistemden bağımsızdır.

Eğer bir dağıtık veritabanı yönetim sistemi, yer şeffaflığını sağlıyorsa ve dilimleme şeffaflığını sağlamıyorsa Şekil 3.17' deki gibi gösterilir.



Şekil 3.17 Yer şeffaflığı.

Terminalden girilen satıcı (supplier) numarası, önce $SUPPLIER_1$ diliminde aranır. Eğer dağıtık veritabanı yönetim sistemi negatif bir cevap verirse, aynı istek $SUPPLIER_2$ dilimine uygulanır. Bu uygulama, yerleşim şemasından bağımsızdır. Fakat dilimlenme şemasından bağımsız değildir. Çünkü dilimlenme yapısı uygulamada bağımsızdır. Yer şeffaflığı, varolan dilim kopyalarının önemsenmediği uygulamalara

izin verdiği için çok yararlıdır. Bu sayede kopyaların bir birimden diğerine taşınmasına ve uygulamaları etkilemeden yeni kopyaların yaratılmasına izin verilir.

Dilimlenme şeffaflığı olmadan yer şeffaflığı sağlanırsa, uygulama programı yazmak daha etkilidir. Buna bir örnek şu şekilde olabilir;

```
SUBINQUIRY;
   read (terminal, $SNUM);
   read (terminal, $CITY);
   case $CITY of
     "SF"  : SELECT name INTO $NAME
             FROM SUPPLIER1
             WHERE SNUM=$SNUM
     "LA"  : SELECT name INTO $NAME
             FROM SUPPLIER2
             WHERE SNUM=$SNUM
   end.
   write (terminal, $NAME).
```

Bu yolla yazılan uygulama, Şekil 3.17'deki uygulama gibi, dağıtık veritabanı yönetim sistemi tarafından sağlanan aynı dağıtık şeffaflığın seviyelerine dayandırılır. Fakat bu uygulama, dilimlenme şemasına daha bağımlıdır. Çünkü dilimlenme şemasında kullanılan "SF" ve "LA" sabitleri bağımsızdır.

Bu yüzden, bu değerlerin herhangi bir değişikliği uygulamayı etkiler fakat Şekil 3.17'deki uygulamayı etkilemez.

2) Fazlalık kontrolu.

Bu referans yapı, dilim seviyesinde fazlalık kontrolu yapar. Örneğin Şekil 3.15'deki R^2 ve R^3 fiziksel görüntüleri kesişirler ve ortak verileri içerirler.

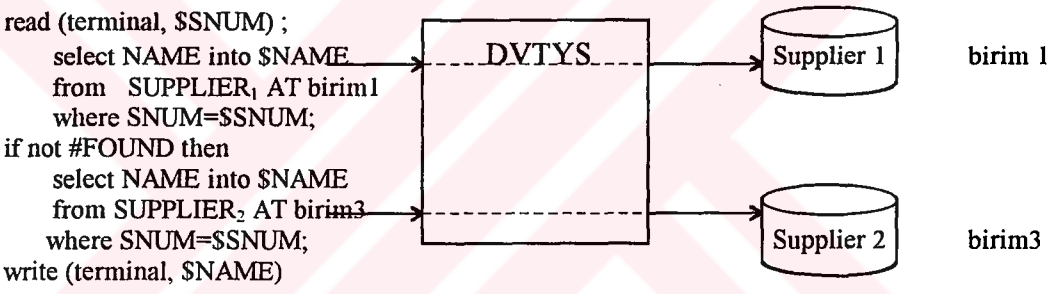
Fazlalık kontrolü, dağıtık veritabanı yönetimi için çok yararlıdır.

3) Lokal veritabanından bağımsızlık.

Bu özelliğe lokal yerleşim şeffaflığı denir ve lokal veritabanının veri modellerini hesaba katmadan, dağıtık veritabanı yönetiminin problemlerini inceleme imkanı sağlar.

- Lokal Yerleşim Şeffaflığı

Bu seviyedeki uygulama, lokal sistemlerden bağımsız, isim kullanan nesnelere karşılık gelir ve nesnenin bulunduğu birimi belirtmek zorundadır.



Şekil 3.18. Lokal yerleşim şeffaflığı.

Şekil 3.18'de de görüldüğü gibi SQL komutunda nesnenin bulunduğu birim belirtilmiştir. Bu durumda, dağıtık veritabanı yönetim sistemi tarafından herbir veritabanı erişimi, bu belirtilmiş birimlerde gerçekleşir. Eğer bu yerleşim sağlanmasaydı, uygulama lokal sistemler tarafından dosya isimlerini arayacaktı.

Lokal yerleşim şeffaflığı, dağıtık veritabanı yönetim sistemleri için önemli bir özelliktir.

Örneğin, birim 1'deki lokal veritabanı yönetim sistemi IMS, birim2'deki lokal veritabanı yönetim sistemi Codasyl olsun. Lokal yerleşim şeffaflığı sağlandığı için ,

dağıtık veritabanı yönetim sistemi, uygulamanın veritabanı erişimlerini, ilgili IMS ve Codasyl programlarına dönüştürmek zorundadır.

Lokal veritabanı özellikleri ile global seviyede kullanılan özellikler arasındaki fark yüzünden aynı problem homojen dağıtık veritabanları için de geçerlidir.

Kopyalama Şeffaflığı

Performans, güvenlik sebeplerinden dolayı, ağ üzerindeki birimlerde kopyalanmış dağıtık verinin olması istenen bir özelliktir. Farklı kullanıcıları uzlaştırmak için kopyalama, performansı artırır. Bir kullanıcı kendi lokal veritabanındaki veriye ulaştığı sırada, başka bir kullanıcı da aynı veriye ulaşmak isteyebilir. Eğer ulaşımda bir problem olursa, ağ üzerindeki diğer bir veritabanındaki verinin kopyası ulaşıma açıktır. Tartışılacak konu, bir nesnenin kaç tane kopyasının olacağıdır. Birden çok kopyası olan verinin, her bir değişiklikte, bunun bütün kopyalara yansıtılması gereklidir.

Veri Bağımsızlığı

Veri bağımsızlığı, bir DVTYS için temel şeffaflık biçimidir. Bunun anlamı; verinin organizasyonunda veya tanımındaki yapılan değişikliklerin kullanıcı uygulamasına yansımamasıdır veya tam tersidir.

Veri tanımlama; iki aşamada gerçekleşir. Birinci aşamada verinin mantıksal yapısı, ikinci aşamada verinin fiziksel yapısı tanımlanır. Bu durumda, veri bağımsızlığını mantıksal veri bağımsızlığı ve fiziksel veri bağımsızlığı olarak iki ayrı kısımda inceleyebiliriz.

Mantıksal veri bağımsızlığında, veritabanının mantıksal yapısında yapılan değişiklikler kullanıcı uygulamasına yansımaz. Eğer bir kullanıcı uygulaması, bir ilişkinin özelliklerinden bazılarını değiştiriyorsa, aynı ilişkiye yeni özelliklerin de eklenmesi, hiç bir uygulamayı etkilemez.

Fiziksel veri bağımsızlığı, kullanıcı uygulamaları için saklama (storage) yapı detaylarının saklanması ile ilgilidir.

Bir kullanıcı uygulaması yazıldığında, fiziksel veri organizasyonunun detayları ile ilgilenilmez. Veri, farklı disklerde hatta farklı saklama hiyerarşileri arasında dağıtılmış da olabilir. Uygulama bu tip durumlarla hiç ilgilenmez. Çünkü uygulamanın icrası için bu durumların önemi yoktur. Bu yüzden, veri organizasyonundaki değişiklikler yüzünden kullanıcı uygulamasının değiştirilmesine gerek duyulmaz.

3.7. Güvenirlilik

Güvenilir bir dağıtık veritabanı yönetim sistemi dağıtık ortamın bileşenlerinde bozukluk olursa bile, veritabanı tutarlılığını bozmadan, işletilen kullanıcı isteklerini devam ettirebilmelidir [6]. Bu güvenirlilik, dağıtık ortamı oluşturan yazılım ve donanım bileşenlerinin güvenirliliğine bağlıdır.

Güvenirlilik ile ilgili iki kavram vardır. Bunlar doğruluk ve kullanılabilirliktir. Bu kavramlar arasında kuvvetli bir ilişki vardır. Çünkü olabilecek yanlış sonuçların riskini artıran şartlar altında işlemin devam ettiğini kabul ederek sistemin kullanılabilirliği artırılabilir.

Sistemin yanlış sonuçlar vermesine neden olan durumlarda kullanılabilirlik azalır. Şimdi bir örnek verelim. İki birimi içeren bir dağıtık veritabanı düşünelim. x_1 ve x_2, 1. ve 2. birimde saklanan aynı x kaydının kopyası olsun. T'de x'ı güncelleyen bir hareket olsun. Ardaşıklığı sağlamak için iki aşamalı kilit, bütünlüğü sağlamak için iki aşamalı taahhüt kullanılsın. Bu durumda T hareketi x_1 ve x_2'ı kilitleyip güncelleme için gerekli hazırlığı yaparak iki aşamalı taahhütü gerçekleştirir.

Her iki birimde, onay işlemine karar verdikten sonra ağ'da bir aksaklık olsun. Ancak bozukluk, onay komutu birinci birimden verilmeden önce oluşsun. Bu durumda sistem ne yapar?

1. birim hareketi onaylayabilir ancak 2. birim, 1.birimin ne yaptığını bilemez. Bu durumda iki olasılık vardır. İlki, bozukluk giderilene kadar x_2'nın kilitli kalmasıdır. İletişim yeniden sağlandığında, hareket sona erecektir. Bu çözümle kullanılabilirlik sağlanamaz, çünkü 2. birimde icra edilen hareket için x'in x_2 kopyası kullanılabilir değildir. Veritabanındaki tutarsızlık riskine karşı kullanılabilirliği maksimize eden ikinci strateji, x_2'nın serbest bırakılması ve diğer hareketlerin bu değeri kullanmasına izin vermektir.

3.7.1. Güvenirlilik Problemleri

Güvenilir dağıtık veritabanı sistemini tasarlarken şu problemler ortaya çıkabilir:

- **Hareket onayı** : Bozukluk durumlarında bile hareketin doğru olarak bitirilmesini sağlayan protokoller tasarlanabilir. Bu protokollere, bitirme protokolleri denir. Bu protokoller, bir onay işlemi bir aksaklık yüzünden kesildiğinde, birimlerdeki bütün hareketlerin doğru olarak bitirilmesini sağlar. Bitme olasılığı, bu protokolun bir aksaklık yüzünden kesilen hareketi bırakmasına bağlıdır.

- **Verinin birden çok kopyası ve eş zamanlılığın önemi**: x kaydına ulaşan yeni hareketler icra edilmek istendiğinde üç farklı alternatif olabilir.

- her bir uçta uygun lokal kopyaların kullanılması.
- x'ın uygun bir kopyasının seçilmesi.
- uygun olmayan x kaydının seçilmesi.

Veritabanının kullanılabilirliği, bu üç yaklaşım için birbirinden farklıdır. Hangi çözümlerin uygulanacağı eş zamanlı kontrol mekanizmasına bağlıdır.

Daha önce anlattığımız gibi, x verisi üstünde işlem yapan bir T hareketi, bozukluk giderilmeden önce çalıştırılacak ve bu durumda T hareketi problemsiz bir şekilde 1. veya 2. birimde icra edilebilecektir. Eğer T hareketi x'ın x_1 kopyasını güncelliyorsa, işleme yeniden başlamadan önce, onarım işlemi x_2 içinde yapılır. x'i

güncelleyen iki hareketin 1. ve 2. birimlerde çalışmasına izin verilirse, çalışma esnasında ardaşıklık sağlanmaz.

- **Ağ'ın belirlenmesi** : Güvenilirlik algoritmaları, ağ üzerindeki bütün aktif olan veya olmayan birimlerde tutarlılık sağlamalıdır. Bu tutarlılığın sağlanması basit bir olay değildir. Bunun için özel bir takım algoritmalara gereksinim duyulur.

- **Nesnel başlangıçlar (Cold restart) :** Nesnel başlangıçlara, ağ üzerindeki bir birimde log bilgisi kaybolduğunda gereksinim duyulur. Bu durumdan etkilenen birimin en son kaldığı durum yapma yetkisi yoktur. Bu yüzden bir önceki duruma dönülür ve bazı alt hareketlerin etkisi kaybolur.

Dağıtık bir veritabanında nesnel başlangıç, bütün dağıtık veritabanında, önceki durumun global tutarlığının sağlanması gerekliliğinden, merkezi sistemlere göre daha zordur.

- **İşlem (commission) hataları** : Bir bileşen, bazı gerekli işlemleri yapamadığında oluşan bozukluklarla karşılaşılabilir. Örneğin, bir birimin bir mesaja cevap vermemesi gibi. Bu hatalar, önemsenmeyen hatalardır. Dağıtık veritabanları için güvenilirlik mekanizması sadece bu hata ile ilgilenir. Ancak, bazı bileşenler, işlemi sona erdirmek yerine, yanlış icra edebilir. Bu hatalar işlem hatası olarak olabilir ve çözümü daha zordur.

3.8. Dağıtık Veritabanının Avantajları ve Dezavantajları

3.8.1. Avantajlar

Performans

Dağıtık sistemlerin gereği olan paralellik, veritabanına erişim performansını arttırır [6]. Çünkü uygulamalar işlemcilere paralel olarak dağılmaktadır. Diğer yandan, her bir birim, veritabanının sadece bir bölümünü idare ettiği için, performansı

düşüren CPU ve I/O servis uyumsuzluğu (darboğaz), merkezi sistemlerde olduğu kadar ciddi bir sorun teşkil etmez. Çünkü işlem yükü, farklı birimlere yayılmaktadır.

Yerel Özerklik

Dağıtık sistemde veri dağıtık olduğu için, bir veriyi paylaşan kullanıcılar, bu veriyi kendi birimlerine taşıyabilirler ve bu sadece birimlerin yerel kontrolü ile sağlanır. Bilgi yönetim sorumluluğunu ve yetki ayrışımlarını içeren çalışmalar, dağıtık sistemleri içeren iş organizasyonlarının bir sonucudur. Bir merkezden idare edilemeyen bu organizasyonlar için yerel özerklik kavramı çok önemlidir.

Güvenirlilik ve Kullanılabilirlik

Kopyası birden fazla birimde bulunan bir veri düşünelim. Birimlerin birinde gerçekleşen bir çökme veya bu birimlerin bazıları arasındaki haberleşme bağlantılarındaki bir aksaklık veriye erişimi imkansızlaştırmaz. Ayrıca sistemin çökmesi veya bağlantı aksaklıkları, tüm sistemin aksamasına neden olmaz. Veriye erişilemediği halde VTYS sınırlı olanaklar sağlar.

Ekonomik Yönü

Veritabanları coğrafi olarak dağılmış ve uygulamalarda birbirlerini etkileyecek dağıtık verileri gerektiriyorsa işlemler, her bir birimde lokal olarak yapılır ve birimlere ayrılan uygulama daha ekonomik olur.

Başka bir deyişle, yerel işlemler yalnız yerel makinayı meşgul ettiğinden ve sadece global işlemler için haberleşme kullanıldığından ağ trafiği daha az olur ve maliyeti de düşürür.

Paylaşılabilirlik

Coğrafi bir yapı üzerinde dağılmış işlemlere sahip organizasyonlarda veri, dağıtık bir yapıda saklanır. Eğer bilgi sistemi dağıtık değilse, veri ve kaynakları paylaşmak imkansızdır. Dağıtık veritabanı sistemi ise bu paylaşımı sağlar.

3.8.2. Dezavantajlar

Güvenlik

Merkezi sistemlerde, veri erişimindeki kontrol, büyük bir avantajdır. Güvenlik, VTYS kuralları ile merkezi bir lokasyondan kontrol edilebilir. Dağıtık bir sistemde ise kendi güvenlik mekanizmasına sahip bir ağ sözkonusudur. Fakat ağ üzerinde yeterli güvenliğin sağlanması ile ilgili ciddi problemlerin olduğu da bilinen bir gerçektir.

Bu yüzden, dağıtık sistemlerin güvenlik problemleri merkezi sistemlere göre daha karmaşıktır.

Kontrol Dağılımı

Dağıtım, koordinasyon ve eş zamanlılık problemleri yaratır. Bunun kontrolü ise büyük bir sorumluluk gerektirir.

Deneyim Eksikliği

Dağıtık veritabanı sistemleri, henüz yaygın olarak kullanıma geçmedi. Karşılaşılabilecek çeşitli problemlerin çözümü, gerçek ortamlarda test edilmediği için, deneyim eksikliği ciddi bir problemdir.

Güçlük

DVTYS problemleri, merkezi veritabanı yönetim sistemi problemlerinden daha karmaşıktır. Bu problemler, sadece merkezi sistem problemlerini içermez, aynı zamanda çözüme ulaşmamış problemleri de kapsar.

Maliyet

Dağıtık sistemler, haberleşme mekanizması gibi ek donanımlara ihtiyaç duyduğunda, donanım maliyeti artar. Teknik problemlerin bazılarını çözmeye gerekli olan karmaşık yazılım ve iletişim, maliyetin önemli bir kısmını oluşturur.

Değişim Zorluğu

Bugün birçok firma, dağıtık olmayan veritabanı sistemlerine yatırım yapmaktadır. Halen merkezi yapıyı, dağıtık yapıya çevirmeye yardımcı olacak araç ve metodlar mevcut değildir. Yapılan araştırmalar, bu zorlukları çözme yolunda ilerlemektedir.

3.9. Dağıtık Veritabanı Problemleri

Bir dağıtık veritabanının en önemli problemlerinden biri iletişim ağlarıdır [5]. Bu yüzden dağıtık sistemlerde hedef, gidip gelen mesajların sayısını ve hacmini en aza indirgemektir. Bu konu baz alınarak bazı problemlerle karşılaşılır. Bunlar aşağıda açıklanmıştır.

3.9.1. Sorgu İşleme

Ağ trafiğini en aza indirgemek demek, sorgu optimizasyon prosesinin dağıtık olmasıdır. Proses, ilgili birimlerdeki lokal optimizasyon adımlarını takip ederek global optimizasyon adımlarını oluşturacaktır.

Örneğin, Q, X birimi tarafından yaratılan bir sorgu olsun. Bu sorgu, Y birimindeki Y_i olarak ifade edebileceğimiz yüzlerce kayıtla, Z birimindeki yüzlerce Z_i kayıtlarının birleşimi olsun. Q sorgusunun icra edilmesi için, X birimindeki optimizer global bir strateji seçmek durumundadır. Bu stratejinin Y_i 'den Z'ye doğru hareketi sözkonusudur. Bu belirlendikten sonra, Z birimindeki birleşme stratejisi, Z'deki lokal optimizer tarafından karar verilmelidir.

3.9.2. Katalog İdaresi

Dağıtık bir sistemde, sistem kataloğu sadece kullanıcılar, index'ler, ilişkiler gibi verileri tutmaz, aynı zamanda kopyalama, yer , dilimleme bağımsızlığı sağlayan gerekli

kontrol bilgilerini içerir. Bu kataloğun nerede ve nasıl saklanacağı ise bir problem teşkil etmektedir. Şu ihtimaller sözkonusudur:

1) Merkezi : Bütün katalog, tek bir merkezi sistemde saklanır.

2) Full kopya : Tüm katalog, her bir birimde saklanır.

3) Parçalanmış : Her bir birimin kendi kataloğu, o birimde saklanır.

4) 1 ve 3'un kombinasyonu : Her bir birim, kendi kataloğuna sahiptir ve ek olarak, bütün bu lokal katalogların kopyası tek bir merkezi birimde saklanır.

1. ihtimal, dağıtık sistemin "tek bir merkeze bağımlılığı yoktur" ilkesine ters düşmektedir. 2. ihtimal, özerklilik kuralından ödün vermektedir. Bir katalogdaki değişim, bunun bütün sistemde yapılmasına neden olmaktadır. 3. ihtimal, yerel olmayan operasyonlar için pahalıya gelebilir. Çünkü bir nesneyi bulmak için birimlerin yarısından fazlasını dolaşmak gerekebilir. 4. ihtimal, daha verimlidir. Bir nesneyi bulmak tek bir erişime indirgenir. Fakat "tek merkeze bağımlılık yoktur" ilkesi ile çelişmektedir.

Katalog yapısına göre, nesnenin tek olan yerel ismi sistem için global olmalıdır. Bu sistem ismi dört kısımdan oluşur:

1) Nesneyi yaratan yaratıcı ismi

2) Bu komutun girildiği birimin ismi

3) Yerel isim

4) Nesnenin saklandığı birimin ismi

SEVIL@NEWYORK.STATS@LONDON şeklindeki bir sistem ismini inceleyelim. Lokal ismi STATS olan bu nesne, NEWYORK birimindeki SEVIL isimli kullanıcı tarafından yaratılmış ve ilk LONDON biriminde saklanmıştır.

CREATE synonym mstats FOR SEVIL@NEWYORK.STATS@LONDON komutu kullanılırsa kullanıcı;

```
select * from  stats
```

veya

```
select * from mstats
```

program parçaları ile aynı verilere ulaşır.

Birinci durumda, lokal isim kullanarak, nesnenin bu kullanıcı tarafından ve bu birimde yaratıldığı ve bu birimde saklandığı varsayılıyor. İkinci durumda, sistem synonym tablosundan sistem ismini belirliyor ve hangi sistemde olduğunu anlayarak ona göre sorgulama yapıyor. Synonym tabloları, bir kataloğun bileşeni olarak düşünülebilir. Her bir birimde olan bu tablolar sayesinde, kullanıcı nesnenin nerede olduğunu bilmeden işlemlerini yapıyor.

Bunlara ek olarak, her bir birimde

1) O birimde doğan nesne için bir katalog girdisi
2) O birimde saklanan her bir nesne için bir katalog girdisi tutulur.

Eğer bir kullanıcı, MSTATS synonym'ı kullanarak sorgulama yapmak istiyorsa; sistem, sistem ismini arar ki bu tamamıyla bir lokal aramadır. Bunun sonucunda, doğduğu birimi bulur, yani bizim örneğimizde London'dı. Böylece uzak bir erişimle London'daki kataloğa ulaşır. Bu katalogda bu nesne için ise yukarda anlatılan bir girdi olmalıdır. Bu nesnenin bulunması ile tek bir erişimle nesneye ulaşılır. Eğer nesne London'dan Los Angeles'a taşınmışsa, bu London kataloğunda tutulduğundan Los Angeles'a ikinci bir erişim gerçekleşir. Nesne en fazla iki erişim ile bulunmaktadır.

Eğer, nesne yeniden San Francisco'ya taşınırsa bu durumda sistem;

1) Bir San Francisco katalog girdisi ekler
2) Los Angeles katalog girdisi siler
3) London'daki, Los Angeles'i gösteren katalog girdisi San Francisco olarak değiştirilir.

Bütün bu anlatılanlar, dağıtıklığı destekler. Merkezi birim sözkonusu değildir ve tek bir birimdeki bir aksaklık tüm sistemi etkileyemeyecektir.

3.9.3. Güncelleme Tekrarı

Veri kopyalarındaki temel sorun, bir nesnede yapılan değişikliğin nesnenin tüm kopyalarına yansıtılmasıdır. Değişikliğin yapıldığı sırada, kopyanın bulunduğu birimde veya ağda bir bozukluğun olması önemli bir sorun teşkil etmektedir. Bu durumda güncelleme başarısızlıkla sonuçlanır, yapılan değişiklik tüm kopyalara yansımamış olur. Bu problemi çözmek için şu yöntemler kullanılabilir:

1) Kopyalanmış her bir nesnenin bir kopyası ana kopya diğer kopyalar ikincil kopya olarak tasarlanır.

2) Farklı nesnelerin ana kopyaları farklı birimlerde tutulur.

3) Ana kopyada güncelleştirme yapıldığında, güncelleştirme operasyonu tamamlanmış sayılır. Kopyanın tutulduğu birim, daha sonra değişikliği ikincil kopyalara yansıtmakla görevlidir.

Bu çözümlerin de kendi içlerinde birtakım problemleri vardır. Dağıtık yapının yerel özerklik ilkesine ters düşmektedir. İkincil kopyalara erişim mümkün olsa bile, ana kopyaya erişim olmadığında, değişiklik hiçbir yere yansımayacaktır. Bunun sonucunda da tek bir sisteme bağlılık sözkonusu olacaktır.

3.9.4. Onarım (Recovery)

Dağıtık bir sistemde onarım kontrolü, iki aşamalı taahhüt (two phase commit) protokolu baz alınarak gerçekleşir. İki aşamalı taahhüt, tek bir hareketin birden çok özerk kaynak yöneticileri ile karşılıklı ilişkide bulunduğu herhangi bir ortamda

gereksinim duyulur. Dağıtık sistemde bu konu çok önemlidir. Çünkü, kaynak yöneticileri (lokal VTYS'ler) farklı birimlerde bulunmakta ve bu yüzden de özerktirler.

Şu konulara dikkat etmekte yarar vardır;

1) "Merkezi sisteme bağımlılık yoktur" kuralı, koordinatör fonksiyonunun, ağ üzerindeki seçilmiş tek bir birimde olmamasını ifade eder. Aynı zamanda da farklı hareketlerin farklı birimlerde çalıştırılmasını ifade eder. Tipik olarak bir hareket, başlatıldığı sistem tarafından yönetilir.

2) İki aşamalı taahhüt prosesi koordinatöre gereksinim duyar. Koordinatör, diğer birimlerle ileşitimi sağlar, bu da fazla mesaj yüzünden fazla maliyete neden olur.

3) Eğer Y birimi, X birimi tarafından koordine edilen iki aşamalı taahhüt prosesine bir katılımcı olarak etki ediyorsa, Y birimi, X biriminin geri dönüş ve onay kararına uymak zorundadır. Bu da yerel özerkliğin bir kaybıdır.

4) Birim veya ağ üzerindeki bir bozukluk durumunda bile iki aşamalı taahhüt prosesinin çalışması istenebilir. Fakat sözkonusu durumlarda bu prosesin sağlıklı çalışması garantili değildir. Bütün araçların (agent), bir hareketi başarı olarak onaylamasını ya da geriye dönmesini garantileyecek bir protokol mevcut değildir. Böyle bir protokol için N, gereksinim duyulan minimum mesaj sayısı olsun ve bazı bozukluklardan dolayı bu mesajlardan bazılarının kaybolduğunu varsayalım. Bu durumda ya kaybolan mesajlar gereksizdir ki o zaman gereksinim duyulan minimum N mesaj sayısı ile çelişmektedir ya da protokol çalışmamaktadır. Burada böyle bir protokolün olmadığını anlıyoruz.

3.9.5. Eş Zamanlılık

Dağıtık sistemlerde eş zamanlılık kontrolü, diğer sistemlerde olduğu gibi kilitleme mantığına göre sağlanır.

Kilitlemek veya kilidi çözmek mesajlar ile sağlanmaktadır. Fakat mesajlar maliyetin artmasına neden olmaktadır. Örneğin, T hareketinin, n tane uzak birimde kopyası olan bir nesneyi güncellemek istediğini varsayalım.

Eğer her bir birim kendi birimlerinde bulunan nesneyi kilitlemekten sorumlu ise ki bu yerel özerklik özelliği ile gerçekleşecektir, böyle bir uygulama için en az 5*n mesaja ihtiyaç vardır:

n tane kilit isteği
n tane kilit teslimi
n tane güncelleme mesajı
n tane doğrulama
n tane kilit kaldırma isteği

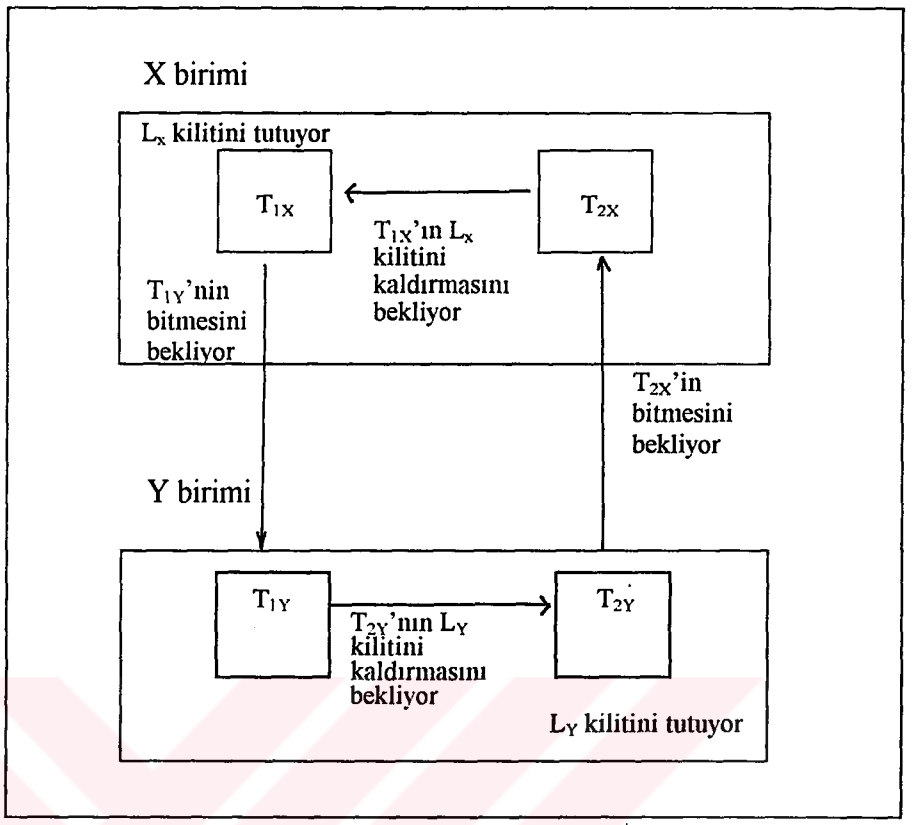
Bu güncellemenin yapılması için geçen süre, merkezi bir sisteme göre daha fazladır. Bu problemi çözmek için daha önce anlattığımız ana kopya mantığı kullanılabilir. Bu sayede toplam mesaj sayısı bir kilit isteği, bir kilit teslimi, n güncelleme, n doğrulama ve bir kilit kaldırma mesajı ile 5*n'den 2*n+3'e düşürülebilir.

Dağıtık sistemlerde, kilitleme ile ilgili diğer bir problem de ölümcül kilitleme (deadlock) olayıdır. Global ölümcül kilitleme, iki veya daha fazla birimde meydana gelen bir kilitlemedir.

Şekil 3.19'da görüleceği gibi X, Y birimlerinde T_1 ve T_2 hareketleri icra edilmektedir.

Buna göre;

1) X birimindeki T_2 hareketinin aracı, kilidin serbest bırakılması için X birimindeki T_1 hareketinin aracını beklemektedir.



Şekil 3.19. Ölümcül kilitlenme.

2) X birimindeki T_1 hareketinin aracı, bitmek için Y birimindeki T_1 hareketinin aracını beklemektedir.

3) Y birimindeki T_1 hareketinin aracı, kilidin serbest bırakılması için Y birimindeki T_2 hareketinin aracını beklemektedir.

4) Y birimindeki T_2 hareketinin aracı, bitmek için X birimindeki T_2 hareketinin aracını beklemektedir.

Bunun sonucunda da ölümcül kilit oluşmaktadır.

BÖLÜM 4 ORACLE İLİŞKİSEL VERİTABANI YÖNETİM SİSTEMİ

Veritabanı yöntem sistemi, veritabanı ile ilişkili tüm işlemlerin kontrol edildiği yazılım şeklidir [8]. Bugün birçok veritabanı modeli ilişkiseldir. İlişkisel veritabanı yönetim sistemi bize:

- Mantıksal veritabanı yapısı ile fiziksel veri saklama bağımsızlığı
- Veriye ulaşım kolaylığı
- Veritabanı tasarımında esneklik
- Veri tekrarını indirgemede kolaylık sağlar.

Veritabanı yönetim yazılımları, bilgi yönetiminin problemlerini çözmede bir anahtar niteliğindedir. Bir veritabanı yönetim sistemi, güvenliği, eş zamanlılığı, bozukluk durumunda etkili çözümleri sağlamalı, veritabanı kullanıcıları için yüksek performanslı bir ortam yaratmalıdır.

İlişkisel bir veritabanı olan Oracle'ın şu veritabanı özellikleri vardır:

- Büyük veritabanları

Oracle, büyüklüğü yüzlerce gigabyte olabilecek büyük veritabanlarını destekler.

- Eşzamanlılık

Oracle, farklı uygulamalarda aynı veriyi çalıştıran kullanıcıların eşzamanlılığını destekler. Veriye ulaşımdaki problemleri en aza indirger.

- Performans

Oracle, özelliklerini yüksek bir performansla gerçekleştirir. Veritabanı kullanıcıları, düşük bir performansla çalışmaz.

- Kullanılabilirlik

Kullanılabilirlik maksimum noktadadır. Veritabanı yedekleme veya sistemdeki kısmi bozukluklar veritabanı kullanımını kesmez.

- Güvenlik

Yetkisiz veritabanı verişimini ve kullanımını engellemek için kullandığı yöntemlerle veri erişim tasarımını kolaylaştırır.

- Uyumluluk

Oracle yazılımı, farklı işletim sistemlerinde çalışabilir. Uygulamalar, değişikliğe gerek kalmadan veya ufak bir takım değişikliklerle herhangi bir işletim sisteminde çalıştırılabilirler.

- Sunucu/İstemci (dağıtık işleme) Ortamı

Oracle, ağ avantajlarını kullanarak, veritabanı sunucusu (server) ile istemci (client) uygulama programları arasındaki işlemin dağıtılmasına izin verir. Paylaşılan veri yönteminin sorumlulukları, veritabanı yönetim sisteminin bulunduğu makina tarafından sağlanır.

- Dağıtık Veritabanı Sistemi

Birbirine ağ ile bağlı ortamlar için, Oracle fiziksel olarak yerleştirilmiş farklı birimlerdeki veriyi, bütün ağ kullanıcılarının erişebileceği mantıksal bir veritabanına dönüştürebilir.

Dağıtık sistemde kullanıcılar, dağıtık olmayan sistemdeki gibi aynı veri tutarlılığına ve kullanıcı şeffaflığına sahiptir.

4.1. Oracle Veritabanı Yapısı

Oracle veritabanı, mantıksal ve fiziksel yapılardan oluşur. Fiziksel olarak bir veritabanı işletim sistemi dosyalarını içerir. Her Oracle veritabanında olması gereken üç tip dosya vardır.

1) bir veya daha fazla veri dosyası
2) iki veya daha fazla redo log dosyası
3) bir veya daha fazla kontrol dosyası

Mantıksal yapısını ise;

1) bir veya daha fazla tablo iş alanı (tablespace)
2) veritabanı şema nesneleri (tablo, görüntü, indeks)
3) veri blokları, extent ve segmentler oluşturur.

4.1.1. Mantıksal Yapısı

Tablo iş alanı (tablespace)

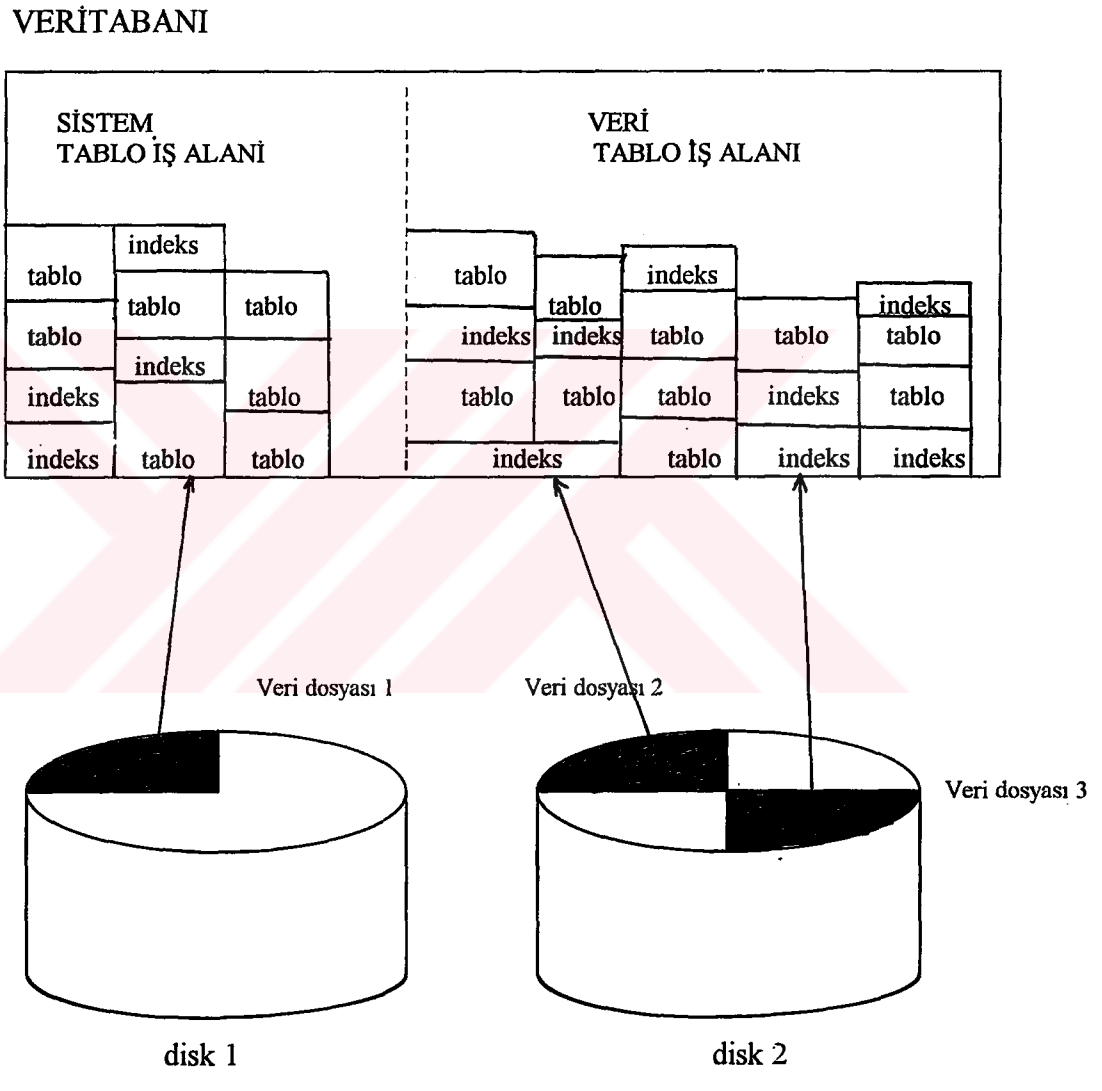
Bir veritabanı, tablo iş alanı adı verilen mantıksal ünitelere bölünür [8] Tablo iş alanı, birbiri ile mantıksal ilişkisi olan yapıları gruplandırmak için kullanılır. SYSTEM tablo iş alanı, veritabanı yaratıldığında otomatik olarak yaratılır, içeriğinde veri sözlüğü, bütün veritabanı nesnelerinin yerleri vardır.

VTY, şu amaçlar için tablo iş alanını kullanır;

- veritabanı nesneleri için boş yer kontrolü sağlamak
- veritabanı kullanıcılarına boş yer açabilmek
- veritabanının kullanılabilirliğini kontrol etmek
- yedekleme ve onarım için

Şema ve Şema Nesneleri

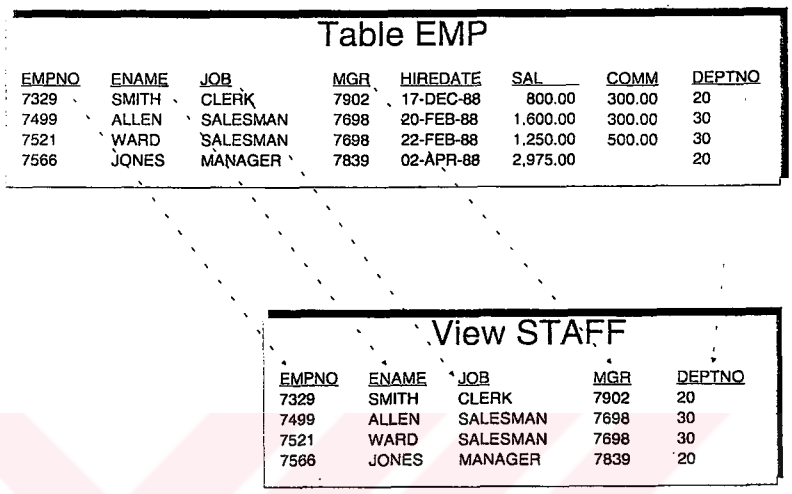
Bir şema, nesneler topluluğudur ve tabloları, görüntüleri (view), indeksleri (index), işlemleri (procedure) içerir. Bir şema nesnesi mantıksal olarak tablo iş alanında saklanır. Nesnenin verileri, fiziksel olarak tablo iş alanının veri dosyalarında tutulur.



Şekil 4.1. Şema nesneleri, tablo iş alanı ve veri dosyaları.

Tablo : Oracle veritabanında, verinin saklandığı en basit ünitedir. Veritabanı tabloları, kullanıcının erişebileceği verileri tutar.

Görüntü : Görüntüler gerçekte veri içermez, tabloların birer türevidir. Sorgulanabilir, güncellenir, kayıt girilebilir veya silinebilir. Görüntü üzerindeki icra edilen herşey, gerçekte görüntünün temel tablosuna yansır. Bellekte yaratılır ve makina bağlandığında silinir. Bu daha çok kompleks sorgular için kullanılır (Şekil 4.2).



Şekil 4.2. Görüntü örneği.

Veri Blokları

Oracle'da veri, veri bloklarında saklanır. Bir veri bloğu, disk üzerindeki fiziksel veritabanı alanıdır. Bu bloğun büyüklüğü veritabanı yaratılırken belirtilir ve maksimum limiti, işletim sistemi blok büyüklüğünün birkaç katıdır. Bütün bloklar byte olarak saklanır.

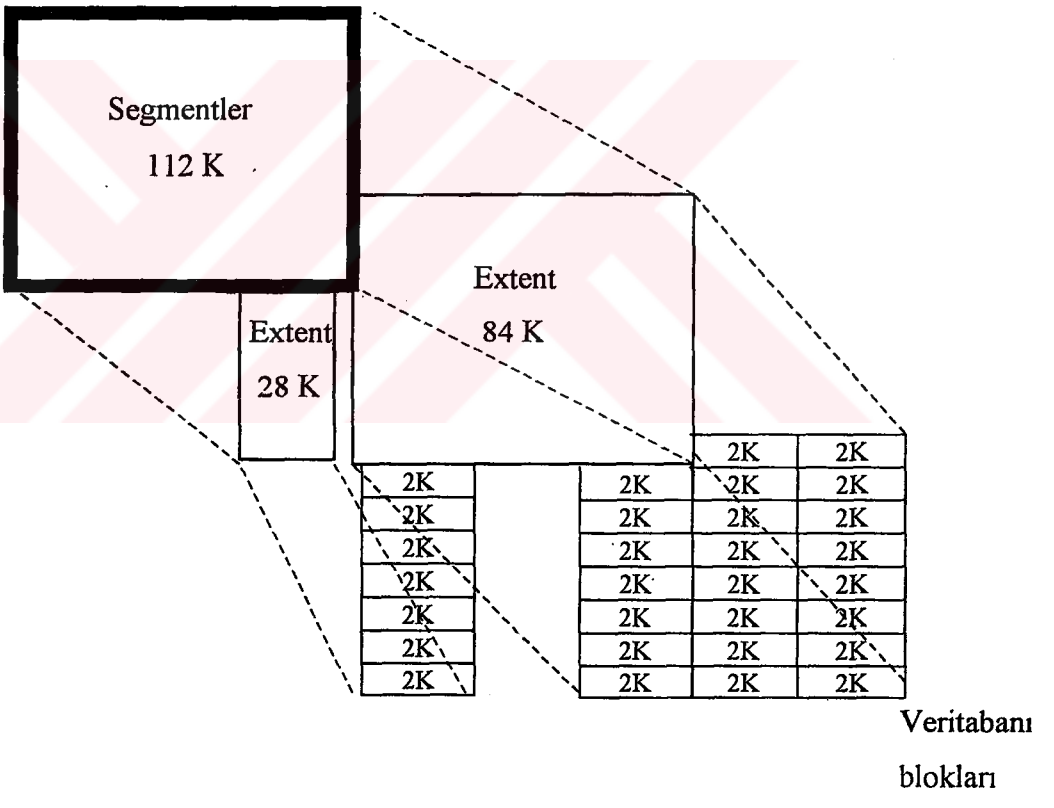
Extent

Extent, bilgileri saklamak için kullanılan veri bloklarının sayısıdır. Extent, blok grubudur. Her segment tipi, bir veya birden çok extent'den oluşur. Bir segmentteki var olan bütün alanlar kullanıldığında, Oracle, segment için yeni bir extent ayırır.

Segment

Segment, tablo iş alanındaki verilerin tutulduğu birden çok veritabanı bloklarından oluşur. Bu veriler, bir tablonun ya da indeksin verileri olabilir. Her bir tablonun verileri kendi veri segmentinde, indeks verileri kendi indeks segmentinde tutulur. Bir geri dönüş (rollback) segmenti için en az iki extent ayrılır. Bir segment, birden çok tablo iş alanına ayrılamaz, ancak bir tablo iş alanı içinde tutulur.

Veri blokları, extentler ve segmentler arasındaki ilişki Şekil 4.3'de görülmektedir.



Şekil 4.3. Veri blokları, extentler ve segmentler arasındaki ilişki.

Bir veritabanının şu segmentlere ihtiyacı vardır:

1) veri
2) index
3) geri dönüş
4) geçici

Bütün segmentlerin yapıları aynıdır ve extentlerden oluşur. Her segment, STORAGE parametresi ile tanımlanır. Bu parametrelerin amacı; tablo ya da görüntüyü kontrol etmektir. Yani tablonun kullanacağı maksimum limit alanı belirlemektir. Örneğin bir tablo yaratıldığında, bu tablonun veri segmenti INITIAL extent içerir. İçinde hiçbir veri olmasa da, ORACLE blokları, bu tablo için ayrılır. Zamanla ayrılan yerler dolacağından, yeni veriler için NEXT parametresi kullanılır. INITIAL extent'ın uzunluğu 5 ORACLE blok uzunluğu kadardır.

Veri Segment; bir tablodaki bütün verileri içerir, bir tablo daima tek bir veri segmente sahiptir.

Indeks Segment; yaratılan tablonun indeksi için bütün indeks verileri tutar.

Geri Dönüş Segment; her veritabanının bir veya daha fazla geri dönüş segmenti vardır. Geri dönüş segmenti veritabanının bir parçasıdır ve şu amaçlar için kullanılır;

- hareket kesilmesi (transaction rollback)
- tutarlılık
- onarım

Geri dönüş segmenti, veritabanındaki herhangi bir tablo iş alanındaki veritabanı nesnelerini etkileyebilecek geri dönüş bilgileri tutar. VTY, SQL komutları ile geri dönüş segmenti yaratabilir, silebilir veya büyüklüğünü artırabilir. Birden çok geri dönüş segmenti, performansı artırdığı için tercih edilir. Geri dönüş segmentleri birçok geri dönüş girişlerini (entry) içerirler. Bir geri dönüş girişi, blok bilgisini, hareket ID, geri dönüş öncesindeki veri bilgisini tutar. Sistemde bir bozukluk olduğunda, aktif hareketin geri dönüş bilgileri olduğu için, geri dönüş segmentinteki bilgilere geri dönülür. Geri dönüş segmentleri, veritabanı kullanıcıları tarafından okunamaz, ulaşılamazlar. Sadece IVTYS tarafından okunur ve yazılırlar.

Geçici Segment; işlemler kuyruğa girdiğinde, Oracle, işlemleri yerine getirmek için geçici bir çalışma alanına (work space) ihtiyaç duyar. Bu disk alanı Oracle tarafından otomatik ayrılır. Bu segment, sıralama için gereklidir. Eğer sıralama işlemi, bellekte yapılırsa veya index kullanarak yapılırsa bu segment yaratılmaz. Geçici segment, komut icrası bittiğinde silinir.

4.1.2. Fiziksel Yapı

Veri Dosyaları

Her Oracle veritabanı bir veya daha fazla fiziksel veri dosyasına sahiptir [8]. Bunlar fiziksel olarak veritabanı veri dosyaları için ayrılmış yerlerde saklanır. Yapılan bir değişiklik veya yeni bir kayıt anında veri dosyasına yazılmaz. Performansı artırmak ve disk I/O'sunu azaltmak için, veri bellekte saklanır ve hepsi bir kerede yazılır. Bir veritabanı yaratıldıktan sonra, ayrılan disk boşluğu herhangi bir veri içermez, bu yerde, program verisi saklanmaz, yaratılacak segmentlerin verileri saklanır. Bir segment yaratıldığında ve büyüdüğünde, Oracle segmentin extenti için yer ayırmak amacıyla veri dosyasındaki boş yerleri kullanır. Tablo iş alanındaki nesnenin segmentindeki veri, tablo iş alanı ile ilgili bir veya birden çok veri dosyasında saklanır. Bir nesne, bir veri dosyasına karşılık gelmez. Veri dosyası, nesnenin verilerinin saklandığı yerdir. Segmentin extenti, bir veya daha fazla veri dosyasında tutulur. Bu yüzden bir nesne, bir veya birden çok veri dosyasına parçalanabilir.

Redo Log Dosyaları

Hareket onaylandığı anda, veritabanında yapılan bütün değişiklikler redo log dosyalarına yazılır. Her Oracle veritabanı, iki veya daha fazla redo log dosyasına sahiptir. Bu dosyalar, onarım operasyonlarında ve veritabanı dosyaları zarar gördüğünde kullanılır.

Kullanıcılar tarafından değişiklik yapılan kayıtlar LGWR prosesi tarafından bir tane redo log dosyasına yazılır. Güç kesintisi olduğu zaman, veritabanı operasyonu

kesildiği için bellekteki veri, veri dosyasına yazılmaz. Sistem açıldığında arada kaybolan veri onarılır. Oracle otomatik olarak, en son redo log dosyasındaki bilgiyi, veri dosyasına yazar. Online redo log dosyası, veritabanı yaratıldığında yaratılır, kaç tane olacağı, uzunluğu ve nerede saklanacağı belirtilir. Bir tane log dosyası dolduğunda LGWR prosesi diğer bir redo log dosyasına yazar. Bu da dolduğunda, ilk log dosyasına döner ve ona yazmaya başlar. Bir dairesel hareket sözkonusudur (log switch). Bir online redo log, en azından iki tane redo log dosyasından oluşur. Oracle ARHIVELOG modda ise, bir dosyaya yazılırken diğeri arşivlenir. Offline redo log dosyası, bir online redo log dosyası kopyasıdır.

Kontrol Dosyaları

Oracle veritabanı her açıldığında, kontrol dosyaları kontrol edilir [9]. Kontrol dosyaları, küçük binary dosyalarıdır, veritabanı ile ilgili bilgileri tutarlar ve veritabanı yaratılırken yaratılırlar. Bu dosyaların içinde veritabanı adı, veri dosyalarının ve redo log dosyalarının adları ve yerleri vardır. Kontrol dosyaları, Oracle tarafından otomatik olarak değiştirilirler. Mesela yeni bir veri dosyası veya redo log dosyası yaratıldığında güncellenirler.

Veri Sözlüğü

Oracle'ın en önemli bölümleriden biri, veri sözlüğüdür. Veri sözlüğü, veritabanı bilgilerini tutan, sadece okuma amaçlı kullanılan tablolar topluluğudur.

Bir veri sözlüğünde şu bilgiler vardır:

- Oracle kullanıcılarının kullanıcı kodu
- nesne isimleri
- ana anaktar bilgileri
- kolonlar için default değerler
- bir tablo'a uygulanacak sınırlamalar
- nesneler için ne kadar yer ayrıldığı
- diğer veritabanı genel bilgileri

Veri sözlüğü, bir tablo'dir ve ulaşmak için SQL komutları kullanılır, kullanıcılar ulaşabilir. Bir veritabanı yaratıldığında veri sözlüğü de yaratılır. Veritabanının çalışması sırasında, Oracle IVTYS tarafından güncelleştirilir. Bir DDL komutu çalıştığında, Oracle önce veri sözlüğüne ulaşır. Veritabanının performansı bu sözlükteki bilgilerle ölçülür.

Veri sözlüğü, şu yapılarda oluşur:

- Temel tablolar : Bu tablolar sadece Oracle tarafından yazılır. Kullanıcılar bu tablolara ulaşabilir. Tablolar normalize edilmiştir ve birçok veri sifreli formatta saklanır.

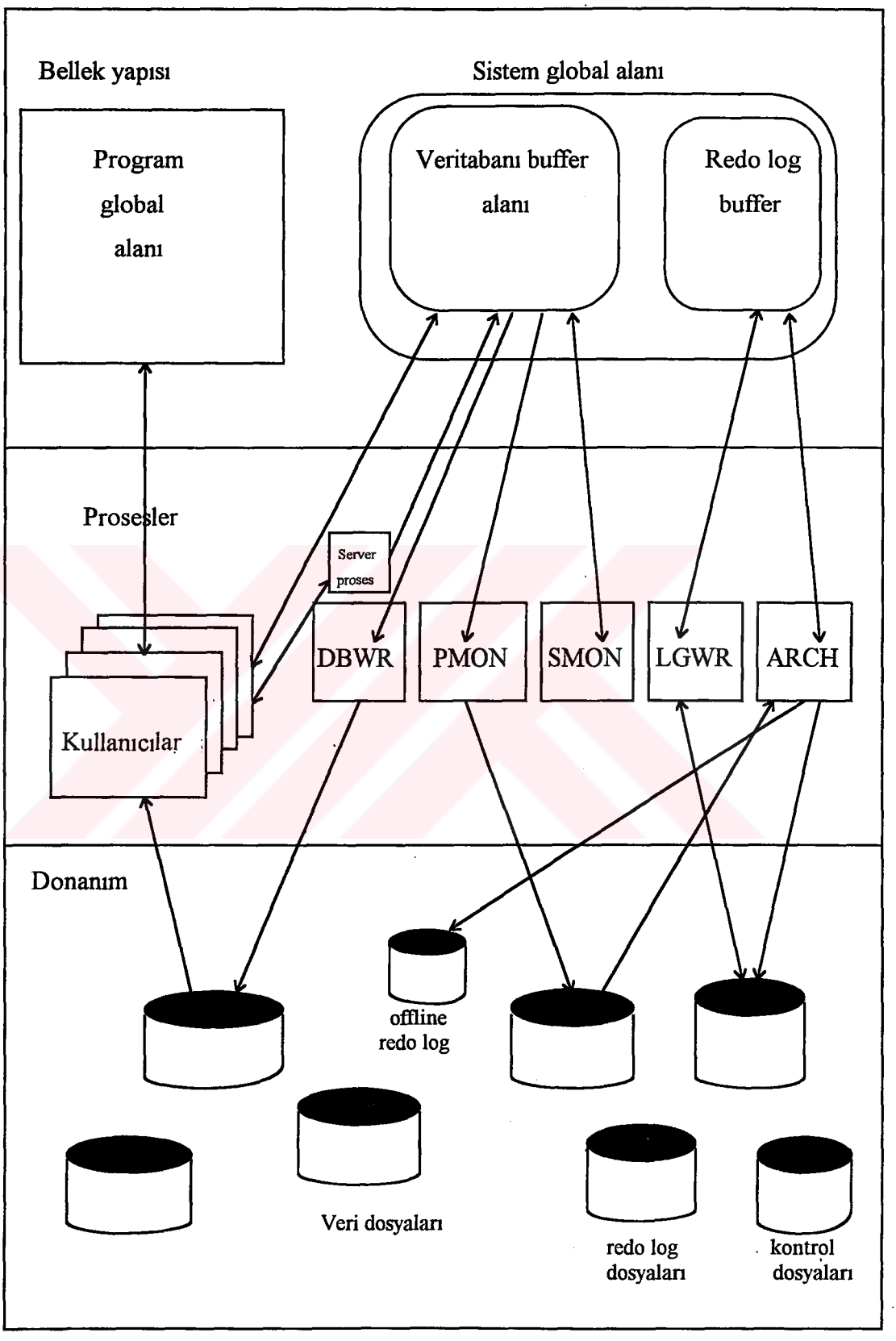
- Görüntüler : Veri sözlüğü, sözlükteki temel tabloların bilgilerini görüntüleyen ve özetleyen görüntülerden oluşur. Veri sözlüğü, görüntü grubundan oluşur. Her bir görüntü, mantıksal bir bilgidir.

4.2. Oracle Sistem Mimarisi

4.2.1. Bellek Yapısı

Oracle'ın işleyişi, bellek yapısı ve proses mantığı ile gerçekleşir [8]. Bütün bellek yapıları, bilgisayarın ana belleğinde mevcuttur. Proses, bilgisayarların belleğinde çalışan iş (job) veya task'lardır.

Bunlar kullanıcı proseslerini ve 5 tane geri (background) prosesleri içerir. Bu 5 geri proses şunlardır: veritabanı yazıcısı (DBWR), log yazıcısı (LGWR), sistem monitor (SMON), proses monitor (PMON) ve arşivleyici (ARCH). Oracle, birden çok işi bitirmek için, bellek yapısını kullanır. Örneğin bellek, işletilecek program kodunu saklamak için kullanılır ve veri, birden çok kullanıcı tarafından kullanılır. Oracle, şu amaçlar için belleği kullanır:



Şekil 4.4 Oracle IVTYS mimarisi.

1) Program kodunu saklamak için

2) Program kodunu işleteceği sırada gereksinim duyduğu veriyi saklamak için

3) Oracle prosesleri arasındaki ilişkiyi ve paylaşma bilgisini tutmak için (mesela kilit bilgisi)

4) Prosesler ve bellek arasındaki transfer bilgilerini saklamak için (örneğin, bir veritabanı bloğunun diskten programa taşınması).

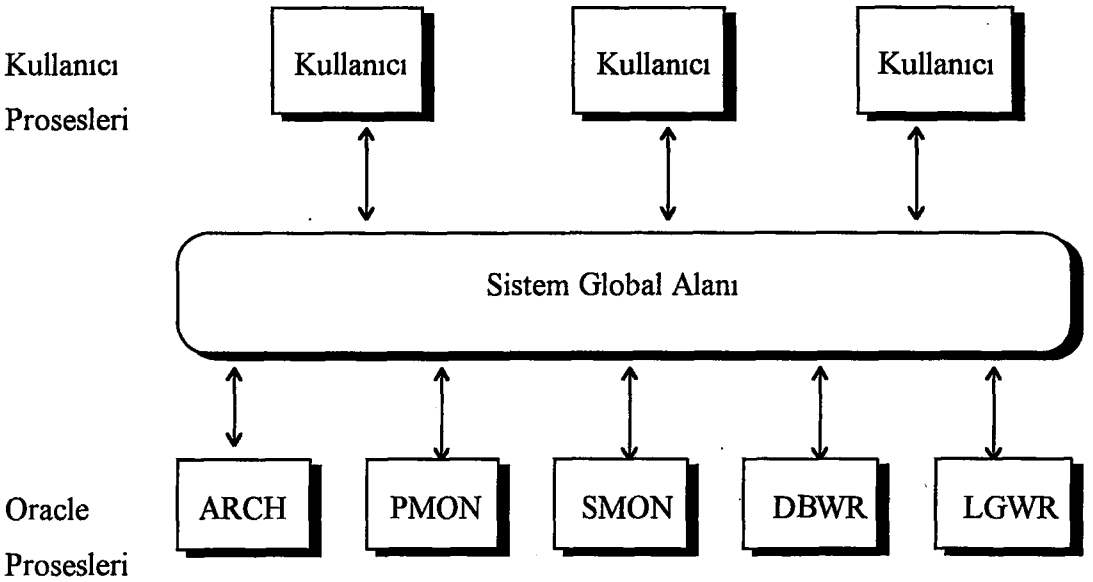
Oracle Instance

Bellek yapısına başlamadan önce, INSTANCE kavramını açıklayalım. Instanse, veritabanına ulaşım ve kontrolu sağlayan bir mekanizmadır. Oracle veritabanındaki bütün işlemler Oracle instance ile ilgilidir ve şunlardan oluşur:

1) Prosesler arasında iletişimi sağlayan paylaşımlı bellek alanı

2) Kullanıcılar tarafından paylaşılan 5 geri proses

Sunucu üzerinden Oracle veritabanı başlatıldığı zaman sistem global alanı (SGA) ayrılır ve bir veya birden çok proses başlatılır. Bu proseslerin ve SGA'nın kombinasyonuna Oracle instance denir (Şekil 4.5).



Şekil 4.5. Oracle instance'ı.

Bellek ve instance'ın prosesleri, veriye etkili bir şekilde ulaşımı sağlar ve kullanıcıların isteklerini yerine getirir. Bir instance başladığında, veritabanı, instance tarafından başlatılır (mount). Aynı makina üzerinde, birden çok instance icra edilebilir, bunlar kendi fiziksel veritabanına ulaşır. Bir CPU üzerinde bütün instance'lar tek bir isme aittir.

Eğer bir veritabanı birden çok instance'a ulaşıyorsa, buna paylaşımlı disk sistemi denir. Instance'ın bir veritabanına bağlanması, veritabanının açılmasıdır. Bir instance durursa, veritabanı kullanıma açık olmaz.

Şimdi Oracle bellek yapısını kısaca inceleyelim.

Sistem Global Alanı

Bir Oracle instance için tutulan verilerin ortak bellek bölgesidir. Instance başladığında Sistem Global Alanı (SGA) ayrılır, bittiğinde boşaltılır. Her bir instance, kendi SGA'sında başlatılır. SGA'daki veri, veritabanına bağlanan kullanıcılar arasında paylaşılır. Çok veri saklamak, disk I/O'sunu en aza indirgemek için SGA büyük olmalıdır.

Veritabanı buffer (cache)

Bu buffer, değiştirilmiş ama henüz diske yazılmamış verileri tutar. Veriler, veritabanı dosyasından okunur ve veritabanı buffer alanında saklanır. Yapılan değişiklikler onaylanana kadar veriler orada kalır. Bu alanın olması disk I/O'sını azaltır.

Redo log buffer

Veritabanında yapılan değişiklikler burada tutulur. Bu buffer, hareket sırasında, geri dönüş segment bloklarında değişiklik yapılmış veriler hakkında bilgi tutar. Hareket onaylandığı zaman, bu buffer alandaki bilgi aktif redo log dosyasına yazılır.

Paylaşımlı havuz (shared pool)

SGA'nın bu bölümü, paylaşımlı bellek yapılarını içerir. Her SQL komutunu çalıştırmak için bu alana gerek vardır. Bu alan, büyük uygulamalarda, aynı komutu

her defasında saklamak (parse) yerine bir kere saklanarak kullanılır. Bu alanın büyük olması, çok kullanıcılı sistemlerde performansı arttırır. Alanın az olması, daha az bellek demektir. Limiti, SGA'nın büyüklüğüne göre belirlenir.

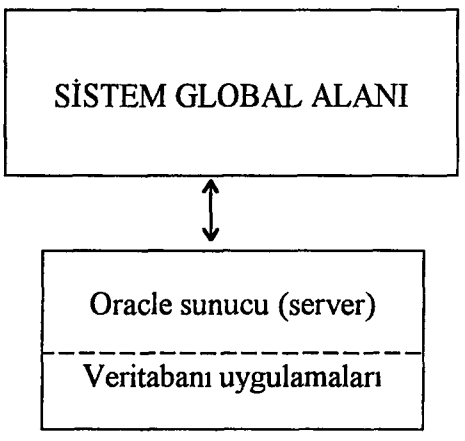
Program global alanı

Program Global Alanı (PGA), kullanıcı prosesleri için verileri tutar. Her bir kullanıcı prosesi bir PGA'ya sahiptir. Bir kullanıcı, Oracle'a bağlandığı zaman, Oracle IVTYS bir PGA yaratır. İçinde kullanıcının bağlantı bilgileri vardır. Bellek yetersiz ise kullanıcı Oracle'a bağlanamaz.

4.2.2. Prosesler

4.2.2.1. Tek Proses (Single Process)

Tek proses Oracle'da veritabanına sadece bir kullanıcı ulaşabilir [8]. PC'ler üzerindeki Oracle, tek kullanıcılıdır. Çünkü MS-DOS gibi işletim sistemleri, çok kullanıcılı ortamlar için yeterli kapasiteye sahip değildir. Tek proses aynı zamanda tek instance'dır. Veritabanı diğer instance'lar tarafından açılamaz. Tek proses Oracle, 5 tane geri prosesleri kullanmaz.



Şekil 4.6. Tek proses.

4.2.2.2. Çoklu Proses (Multiple Process)

Çoklu proses Oracle, Oracle'ın farklı kısımlarını çalıştırmak için birçok proses kullanır ve her bir proses için Oracle instance, bir iş icra eder. Çoklu proses ortamında birden çok kullanıcı ve uygulama kullanılır ve her bir kullanıcı için 5 geri prosese ve bir kullanıcı prosesine gereksinim duyulur. Burada amaç, aynı zamanda, birden çok kullanıcının veriye ulaşmasıdır. Veritabanı uygulamalarının ve Oracle işleyişinin birçok prosese bölünmesiyle, birden çok uygulama ve kullanıcı, tek bir veritabanı instance'sına bağlanabilir.

Çok kullanıcılı sistemde prosesler ikiye ayrılır:

-Kullanıcı prosesi
- Oracle prosesi

Kullanıcı Prosesi

Her veritabanında bulunur. Bir kullanıcı prosesi, uygulama programına ait yazılım kodu çalıştırıldığında kullanılır. Sunucu prosesler ile iletişimi sağlar. Normal olarak bir kullanıcı prosesi bir kullanıcı, Oracle'a bağlandığında yaratılır.

Oracle Prosesi

Çok kullanıcılı sistemlerde, Oracle, iki tip Oracle prosesi ile kontrol edilir.

1) 5 geri proses

Kullanıcıların düzenli çalışmasını ve performansını artırmak için, çoklu proses Oracle bazı ek prosesler kullanılır. Bu prosesler, eş zamanlı değildir, farklı zamanlarda veritabanına yazma ve okuma işlemlerini gerçekleştirirler. Aynı zamanda Oracle proseslerinin düzenini sağlarlar ve veritabanının performansı ve güvenirliliği için paralel olarak çalışırlar.

Her bir Oracle instance, 5 geri prosesten oluşur.

DBWR (veritabanı yazıcısı) : Bunun görevi, veritabanı buffer alanındaki, değiştirilmiş blokları veritabanı yazmaktır. Performans açısından, hareket onaylanana kadar yazma işlemini yapmaz.

LGWR (log yazıcısı) : Bu proses, redo log girişlerini diske yazar. Redo log bilgisi, SGA'daki buffer'da oluşur. Hareket onaylandığında, LGWR, redo log girişlerini redo log dosyasına yazar. Bazı durumlarda, eğer daha fazla buffer alanına ihtiyaç duyulursa, LGWR, hareket onaylanmadan redo log girişlerini diske yazar. Hareket daha sonra onaylanırsa, bir girişler kalıcı hale gelir.

SMON (Sistem Monitor) : Bunun amacı, instance onarımını sağlamaktır. SMON, kullanılmayan geçici segmentleri temizlemek, sistem ve instance onarımı sırasında biten hareketleri onarmakla sorumludur. Tablo iş alanı ve dosya geri döndüğünde, bu hareketler SMON tarafından onarılır.

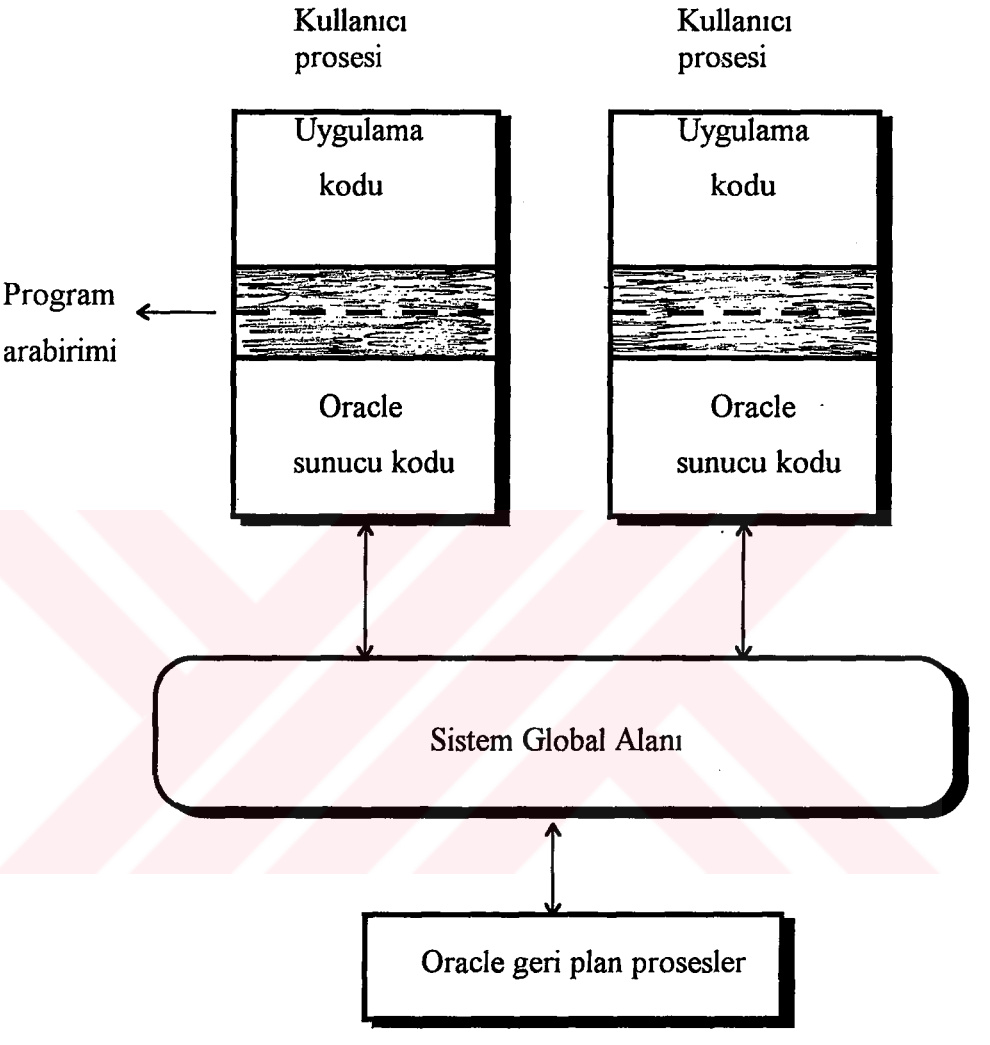
PMON (Proses Monitor) : Bir kullanıcı prosesi, başarısızlıkla sonuçlandığında, PMON onarımı başlatır, proseslerin kullandığı buffer alanlarını temizler, kilitleri açar.

ARCH (Arşivleyici) : Bir redo log dosyası dolduğunda ARCH prosesi bunu teype kopyalar. Yalnız redo log dosyası, ARCHIELOG modunda ise ARCH prosesi aktiftir.

2) Sunucu proses

Sunucu proses, Oracle'a bağlanmış kullanıcı proseslerinden gelen istekleri karşılamak için yaratılır. Sunucu proses, kullanıcı prosesi ile kullanıcı prosesinden gelen istekleri icra etmek için Oracle arasında iletişimi sağlamakla yükümlüdür. Mesela, SGA'nın veritabanı buffer'ında bulunmayan bir veri sorgulanmak istenirse, sunucu proses, SGA'daki veri dosyalarındaki uygun veri bloklarını okur. Oracle, her

sunucu proses için kullanıcı proses sayısını değiştirmeyi olanak sağlayan konfigürasyona sahiptir.



Şekil 4.7. Kullanıcı/sunucu proseslerin birleşimi.

4.3. Hareket Yönetimi

Hareket, bir veya daha fazla SQL komutundan oluşan mantıksal bir iş ünitesidir [8]. Hareketteki SQL komutu ya onaylanır (commit) ya da geri dönülür (rollback).

Şekil 4.8'de de görüldüğü gibi 3 tane SQL komutunu içeren bir hareket sözkonusudur. Eğer bu üç SQL komutu başarılı bir şekilde çalışırsa, hareketin sonucu veritabanına yazılır. Eğer herhangi bir sebepten dolayı birinde hata olursa yapılan bütün değişikler geri alınır.

Bir SQL komutunun başarıyla tamamlanması, onaylanmasından farklıdır. Komut, onaylanana kadar geçici olarak yapılan değişiklikler değiştirilebilir. Onaylanması, bütün değişikliklerin kalıcı olmasını sağlar. Bir hareket, onaylanana kadar, hareketin etkileri kullanıcıya görünmez.

Oracle'da veri tutarlılığı, hareket yöntemi ile garanti altındadır. Çünkü, işletim sistemi ya da kullanıcı programı herhangi bir sebeple hareketin ortasında kesilirse, veritabanı otomatik olarak hareket başlamadan önceki haline geri döner. Bununla çalışılan veriyi kontrol etmede esneklik ve tutarlılık sağlanmış olur.

```
UPDATE savings_accounts
  SET balance = balance - 500
  WHERE account = 3209;

UPDATE checking_accounts
  SET balance = balance + 500
  WHERE account = 3208;

INSERT INTO journal VALUES
(journal_seq.NEXTAL, '1B',
 3209, 3208, 500);

End Transaction
COMMIT WORK;
```

Şekil 4.8. Hareket örneği.

Hareket sırasında kilitlenen kayıtlar, onay ve geri alma durumlarında serbest bırakılır. Eğer hareket ilgili kayıtları kilitlemişse, hareket kiliti kalkana kadar kullanıcılar beklemek zorundadır.

4.3.1. Geri Dönüş Segmenti ve Hareket

Daha önce anlattığımız gibi geri dönüş segmentlerinde değişen veri blokları tutulur. Sistemde bir aksaklık olduğu zaman, aktif hareketin geri dönüş girişleri olduğu için geri dönüş segmentlerdeki bilgilerden geriye dönülür. Oracle, veritabanında var olan her geri dönüş segmenti için bir hareket tablosu tutar. Bu tablo bütün hareketleri içerir. Bu hareketler, icra edilen herbir değişiklik için geri dönüş segmentlerini kullanan hareketlerdir. Geri dönüş segmentlerinde verinin değişmeden önceki hali tutulur. Eğer bir hareket, geri dönüşü gerektiriyorsa, verinin bir önceki hali sağlanır.

Bir hareket başladığında, hareket için, Oracle tarafından otomatik olarak uygun geri dönüş segmenti belirlenir. Güncelleme işlemi yapan hareketler için geri dönüş segmenti ayrılmaz. Hareket sırasında, ilgili kullanıcı proses, bu segmente geri dönüş bilgilerini yazar. Hareket onaylandığında, geri dönüş bilgisi serbest bırakılır. Bir hareket için extent dolarsa, Oracle uygun bir extent bulmak durumundadır. Bu ya, geri dönüş segmenti için ayrılan extentler yeniden kullanılarak ya da yeni bir extent için yer ayrılarak sağlanır.

4.3.2. Hareket Kontrolü

Hareketin onaylanması veya geri alınması durumunu kontrol etmek için üç tane SQL komutu kullanılır:

1) savepoint, 2)commit work, 3) rollback work

1) Savepoint

Kaydetme noktaları (Savepoint), hareketleri küçük parçalara bölmek için kullanılır. Bununla, herhangi bir zamanda yapılan işlemler saklanabilir ki bu işlemler daha sonra onaylanabilir veya geri alınabilir. Böylece uzun bir hareket için işlem devam ederken, hareketin parçaları saklanabilir ve sonuçta onay veya geri işlem yapılabilir.

İkinci bir kaydetme noktası yaratıldıysa ve öncekiyle ismi aynı ise daha önceki silinir. Bir kaydetme noktası yaratıldıktan sonra ya COMMIT WORK ya da kaydetme noktasına kadar ROLLBACK WORK yapılır. Default olarak, her işlem için maksimum aktif kaydetme nokta sayısı 5 tanedir. Aktif kaydetme noktası, son onay veya geri alma işleminden itibaren belirlenmiş olan kaydetme noktasıdır.

Kaydetme noktaları, birbirini etkileyen programlar için çok yararlıdır. Çünkü bir işlemin adımlarını yaratıp isimlendirebiliriz. Böylece karmaşık işlemlerin kontrolü sağlanmış olur. Örneğin, uzun karmaşık güncellemelerin olduğu bir programda kaydetme noktası kullanılıyorsa herhangi bir yerde hata olduğu zaman en başa dönme zorunluluğu olmaz.

2) Commit work

Bu komut ile şu işlemler gerçekleşir:

- Aktif hareketteki bütün değişiklikler kalıcı hale gelir.
- Hareketteki bütün kaydetme noktaları silinir.
- Hareket sonlandırılır.
- Hareketteki kilitleri kaldırır.

Bir hareket onaylanmadan önce şu işlemler yapılır;

- IVTYS, bellekte ilgili kaydın güncelleştirilmeden önceki halini oluşturur.
- IVTYS, bellekte redo log kayıtlarını oluşturur.
- Yapılan değişiklikler, veritabanı buffer alanına yazılır.

Onaylandıktan sonra ise;

- Hareket, "tamamlandı" diye işaretlenir.
- Redo log girişleri diske yazılır.
- İlgili kayıtlar serbest bırakılır.
- Veritabanı buffer alanı veritabanına yazılır.

3) Rollback Work

Bir SQL komutunun çalışması sırasında, mesela tek (unique) tanımlanmış bir indekse, çift kayıt girilmek istenirse; bu komut geri döner. SQL komutunun yazılış hatası varsa, komut saklanırken hata verir, geri dönüşe neden olmaz.

Oracle, SQL komutu çalışmadan önce, bir kaydetme noktası yaratarak komut bazında geri dönüş gerçekleştirir. Kullanıcı direkt olarak bu kaydetme noktasına ulaşamaz. Eğer ölümcül kilitlenme sözkonusu ise SQL komutundan geri dönebilir.

4.4. Kurtarma Mekanizması (Recovery)

Her veritabanında olabilecek sistem ve donanım aksaklıklarına karşı, veritabanı yönetim sistemi, veritabanını sorundan önceki haline getirecek olan kurtarma mekanizmasına sahip olmalıdır.

Oracle veritabanında şu kurtarma özellikleri vardır:

1) donanım, yazılım veya sistem aksaklıklarına karşı koruma
2) online, offline yedekleme kararı
3) veritabanı açık veya kapalı iken onarım kararı
4) veritabanı icrası sırasında, bir veya birden çok tablo iş alanı ve tablo iş alanındaki birden çok dosya onarımı
5) export, import kullanımı
6) redo log dosyalarına birden çok yazılım.

Bir bozukluk durumu gerçekleştiğinde ilk adım, hangi durumdan kaynaklandığını belirtmektir. Daha sonra uygun kurtarma adımları uygulanarak veritabanı yeniden başlatılır. Biraz sonra anlatılacak altı çeşit aksaklık olabilir. Bunlardan instance ve disk aksaklığı gerçekleştiğinde bazı onarım işlemleri VTY tarafından çalıştırılır. Kullanıcı komut hatalarında ve proses aksaklığında Oracle, otomatik olarak onarımı gerçekleştirir.

Olabilecek aksaklıklar kısaca şöyledir;

1) Kullanıcı Hatası : Eğer her akşam export alınıyorsa ve kullanıcı bir tabloyu kaza ile silmişse önceki günün tablo versiyonu yeniden yüklenebilir [9].

2) Komut Hatası : Buna, uygulama sırasında karşılaşılan mantıksal bir hata sebeb olur. Bu hata, yanlış bir komut veya geçersiz bir veri olabilir, ve bu durumda komut otomatik olarak geri dönüş olur; Oracle veya işletim sistemine hata mesajı gönderilir.

3) Proses Aksaklığı (failure) : Bu aksaklık, Oracle'a erişen kullanıcı proseslerinin aksaklığını ifade eder ve meydana geldiğinde veritabanı prosesleri çalışmasına devam eder. Oracle geri prosesi (PMON) bu prosesi keser. PMON, prosesin kullandığı kaynakları serbest bırakır, hareketi keserek problemi çözer. PMON, bunu otomatik olarak yapar. VTY buna hiç bir müdahalede bulunmaz.

4) İnstance Aksaklığı : Bu aksaklık, instance'ın çalışmasına engel olan donanım, yazılım veya sistem aksaklığıdır. Şunlara neden olur:

- CPU kaybına
- Güç kaybına
- Oracle geri proseslerden birinin başarısızlığına
- Dosya kaybolmasa da, dosyaya ulaşım başarısızlığına

Bu durumda onarım otomatik olarak çalışır.

5) Disk Bozukluğu : Dosyaları okuma ve yazma işlemlerini engelleyen donanım, yazılım ve sistem aksaklığıdır. Veritabanının kaza ile silinmesi, birçok

dosyanın kaybolmasına neden olan sistem çökmesi disk bozukluğuna örnektir. Bu durumda veritabanının yedeği ve bozukluk olmadan önceki redo log dosyaları indirilir. Bu bozukluk veri dosyalarını okuma hataları ve yazma hataları olarak ikiye ayırır.

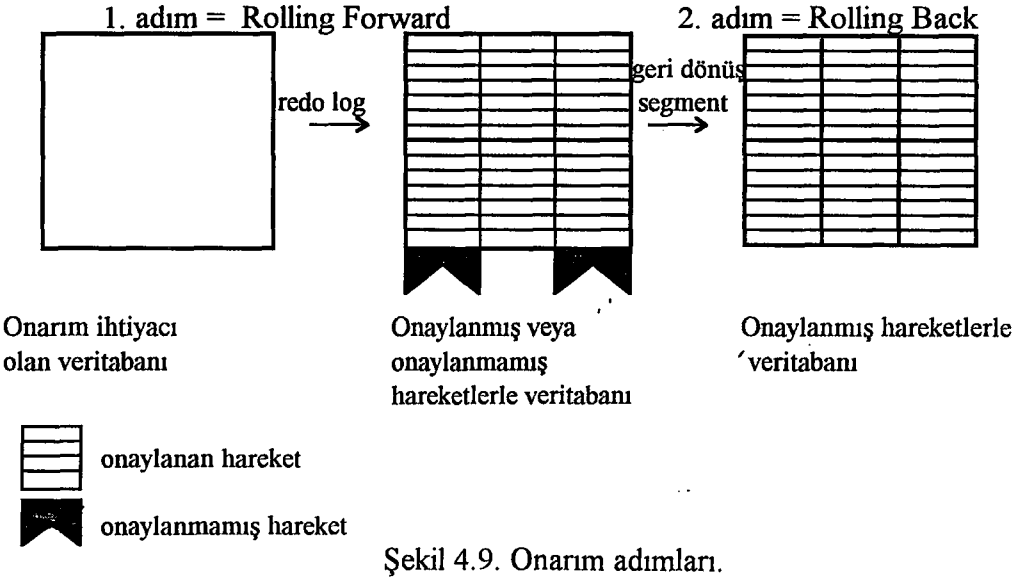
6) Ağ Bozukluğu : Ağ bozukluğu, veritabanının normal akışını yarıda keser. Mesela istemci uygulamalarının icrasını kesebilir ve bu da proses bozukluğuna neden olur. Bu durumda Oracle geri prosesi PMON problemi araştırır.

Sistem bozuklukları durumunda Oracle'ın kurtarma mekanizması iki adımda gerçekleşir. Orjinal adlarıyla;

1) Rolling forward
2) Rolling back

4.4.1. Rolling Forward ve Redo Log

Redo log, veritabanı buffer alanında, değişiklik yapılan bütün kayıtları tutan işletim sistemi dosyasıdır. Bir bozukluk durumunda ilk kurtarma adımı, redo log dosyalarındaki bütün değiştirilen kayıtları veri dosyalarına geçirmektir. Fakat geri dönülen veride redo log'da tutulduğu için, rolling forward, geri dönüş segmentini yeniden derler. Rolling forward, online redo log dosyalarını içerir. Roll forward'dan sonra, veri blokları bütün onaylanan değişiklikleri ve redo log'da tutulan onaylanmamış değişiklikleri içerir.



Şekil 4.9. Onarım adımları.

4.4.2. Rolling Back ve Geri Dönüş Segmenti

Veritabanı kurtarma işleminde, geri dönüş segmentleri, rolling forward aşamasında uygulanan onaylanmamış hareketlerin etkilerini yok etmek için kullanılır. Roll forward'dan sonra, onaylanmamış değişiklikler ihmal edilir. Bu işleme rolling back denir.

Instance Bozukluğunu Kurtarma

Bu kurtarma işlemi, instance bozukluğu olmadan önceki hareket tutarlılığını sağlamak için veritabanını onarır. Oracle bunu otomatik olarak yapar ve şu aşamalardan geçer;

1) Rolling forward, redo log dosyasına ve geri dönüş segmentlerine yazılmış, veri dosyalarına yazılmamış kayıtları onarır.

2) Onaylanmamış hareketlerden geri dönülür.

3) Bozukluk anında, işlemdeki hareketlerin tuttuğu kilitler açılır.

Disk Bozukluğunu Kurtarma

Bu kurtarma operasyonu için şu iki işlem ile gerçekleşebilir.

1) Eğer veritabanının redo log dosyaları arşivlenmiyorsa buradaki onarım, tüm yedeğin indirilmesidir.

2) Eğer redo log dosyaları arşivlenmiyorsa, zarar görmüş veritabanını inşa etmek için bir onarım işlemi sözkonusudur.

Herhangi bir disk bozukluğu, tüm veritabanının onarımını gerektirir. Veritabanının sadece bir parçası onarılmaz, bu tutarsızlığa neden olur.

4.5. Tutarlılık ve Eşzamanlılık

Tek kullanıcılı bir veritabanında, bir kullanıcı aynı zamanda aynı veriyi değiştiren diğer bir kullanıcı ile ilgilenmeksizin bir hareket ile değişiklik yapabilir. Çok kullanıcılı bir veritabanında, eş zamanlı birden çok hareket ile aynı veri güncelleşebilir. Hareket eş zamanlılığı, çalışan aynı hareketin aynı etkiyi sağlayacağını garanti etmelidir. Bu yüzden, çok kullanıcılı bir veritabanında en önemli konulardan biri, tutarlılığın ve eşzamanlılığın nasıl sağlanacağıdır.

Birden çok kullanıcı tarafından kullanılan veriye erişim koordine edilmelidir ve çok kullanıcılı bir veritabanında şu durumlara engel olunmalıdır:

1) Tutarsız okuma : Bir hareketteki basit bir sorgulama sırasında, diğer bir hareket tarafından onaylanan veya onaylanmayan değişiklik görülür. Bu yüzden, sorgulamadan gelen veri tutarsızdır.

2) Tekrar edilmeyen okuma : Basit bir hareketteki aynı kaydı bulmaya çalışan iki sorgu, farklı kayıtları getirebilir. Çünkü, diğer onaylanan hareketteki değişiklikler, bu sorgulamaya yansımayabilir.

3) Kirli okuma : Bir hareket, başka bir kullanıcının onaylanmamış hareketindeki kaydını okuyabilir. Fakat birinci hareketin okuduğu kayıt daha oluşmamıştır.

4) Kayıp değişiklikler : Bir hareketteki bir güncelleme, başka bir hareket onaylanmadan önce, bu hareketin yaptığı güncelleme üzerine yazılabilir.

5) Yıkıcı DDL operasyonları : Bir kullanıcı, bir tabloyu silerken ya da değiştirirken, başka bir kullanıcı tablodaki kaydı değiştirebilir. Kursor ile kayıt seçildiğinde kilitler serbest bırakılır. Bu yüzden sorgulama aynı kaydı içeriyorsa ve diğer kullanıcılar kursor ile okuma arasında kaydı değiştirdiyse okunan kayıt farklı olur.

4.5.1. Kilitleme (Locking)

Çok kullanıcılı veritabanı, veri tutarlılığını, bütünlüğünü ve eş zamanlılığı problemlerini çözmek için bazı veri kilitleme yapıları kullanır. Kilitleme, aynı yapıya ulaşan kullanıcılar arasında karşılıklı olumsuz etkiyi önlemek için kullanılan bir mekanizmadır.

Oracle, değişik lock çeşitleri kullanır. Bu da Oracle'a büyük ölçüde veri eşzamanlılığını sağlar.

Oracle'ın sağladığı olanaklar şunlardır;

- Kaydı okuyanlar, aynı kayıt bloğuna yazanları beklemez
- Kaydı yazanlar, aynı kayıt bloğunu okuyanları beklemez
- Kaydı yazanlar, sadece aynı zamanda aynı satırı güncelliyorsa diğer yazanları bekler.

Bütün bu durumlarda, bir SQL komutu çalıştırılırken, gerekli kilit, Oracle tarafından otomatik olarak belirlenir. Oracle, istenirse, elle kaydı kilitleme seçeneği de verir. Sorgulama için okuma kilitleri kullanmaz.

Kilit sınırlamaları : Çok kullanıcılı bir veritabanında iki türlü kilit kullanılır:

1) Exclusive kilit : Bu kilit, aralarında ilişkili kaynak paylaşımına engel olur. Bir hareket bir kaynağı kilitlemişse, exclusive kilit serbest kalana kadar, sadece bu hareket kaynağı değiştirebilir.

2) Share kilit : Yapılan operasyona bağlı olarak, bir share kilit, aralarında ilişkili kaynakların paylaşımına izin verir. Birden çok kullanıcının, aynı anda aynı kayda ulaşması, share kilit ile gerçekleşir.

Share kilit, exclusive kilitten daha fazla veri eşzamanlılığına izin verir.

Ölümcül Kilitlenme: Birden çok hareketin birlikte çalışması sonucunda ortaya çıkan durumdur. Bir ölümcül kilitlenme, ikiden fazla kullanıcının, birbirlerini kilitleyen kayıtları beklemesi sonucunda meydana gelir. Yani bir kullanıcı bir kaynağı kilitledikten sonra, iki veya daha fazla kullanıcı, bu kaynağa ulaşmak için bekliyor durumundadır. Oracle, otomatik olarak ölümcül kilitlenme durumunu arar. Bulduğunda, buna sebep olan komutlardan birini keserek ölümcül kiliti ortadan kaldırır.

Tablo 4.1. Ölümcül kilitleme.

Hareket 1	Zaman sırası	Hareket 2
UPDATE emp SET sal=sal*1.1 WHERE empno=1000;	1	UPDATE emp SET mgr=1342 WHERE empno=2000;
UPDATE emp SET sal=sal*1.1 WHERE empno=2000;	2	UPDATE emp SET mgr=1342 WHERE empno=1000;

Birinci durumda bir problem yoktur. Her bir hareket, güncelleme yapmak için bir satır kilitlemiş durumdadır. İkinci durumda, bir hareket, diğer bir hareket tarafından kilitlenen satırı güncelleştirmeye çalışmakta ve ölümcül kilitlenme oluşmaktadır.

4.5.2. Hareket ve Eşzamanlılık

Oracle, kayıt eşzamanlılığını ve bütünlüğünü, hareket ve kilit mekanizması ile sağlar. Bir hareketteki komutların neden olduğu bütün kilitler, hareket boyunca tutulur ve kirli okumalara, kayıt değişikliklerine engel olunur. Bir hareketteki SQL komutlarının yaptığı değişiklikler onaylandıktan sonra diğer hareketler tarafından görülebilir. Bir hareket, onaylandıktan veya geri dönüşten sonra bütün kilitler serbest bırakılır.

4.5.3. Okuma Tutarlılığı

Okuma tutarlılığı, Oracle'ın sağladığı tutarlılık modelidir. Bu modele göre;

1) Bir komut ile erişilen kayıt, komut icra edilirken değişmeyecektir.
2) Veritabanını okuyanlar, yazanları beklemezler.

Oracle, okuma tutarlılığını, iki değişik seviyede sağlar.

1) Komut seviyeli okuma tutarlılığı

Bu, basit bir sorgulama sonucunda gelen kaydın tutarlılığını garantiler. Bu yüzden, sorgulamanın yapıldığı anda, onaylanan bir hareket tarafından değişiklikler asla görülmez.

2) Hareket seviyeli okuma tutarlılığı

Bu, aynı hareketteki bütün sorgulamalardan gelen kaydın tutarlılığını garantiler. Bu yüzden, hareket seviyeli okuma tutarlılığı, tekrarlanabilir okumalara neden olur.

4.6. Güvenlik

Veritabanı güvenliği, veritabanı nesnelerini istenmeyen kullanıcılara karşı koruyan, denetimi veritabanı yöneticisi konumundaki kişilere belli ölçülerde veren bir mekanizmadır.

Veritabanı güvenliğini, sistem güvenliği ve veri güvenliği olarak ikiye ayırabiliriz:

Sistem güvenliği;

1) Uygun kullanıcı kodu ve şifre kombinasyonunu
2) Kullanıcı için kaynak limitlerini
3) Kullanıcının hangi sistem operasyonlarını icra edeceğini belirler.

Veri güvenliği;

1) Herbir şema nesnesi için veritabanı güncellemelerini

2) Hangi kullanıcıların, hangi şema nesnesine ulaşacağını ve ne gibi yetkileri olacağını (mesela bir kullanıcı yeni kayıt ekleme yetkisine sahip olabilir, fakat kayıt silme yetkisine sahip olmayabilir) belirler.

Oracle'ın bilgileri koruma stratejisi; kapsamlı erişim kontrolü sağlamasıdır. Oracle veritabanı güvenliğini şunlarla sağlar:

1) veritabanı kullanıcı kod ve şifresi
2) haklar
3) roller
4) kaynak limitleri
5) veritabanındaki güncellemelerin dosyaya yazılması

1) Veritabanı Kullanıcı kod ve şifresi

Bir kullanıcının Oracle veritabanına ulaşmak için geçerli kullanıcı koduna ve şifresine sahip olması gerekir. Veri sözlüğü, her kullanıcı koduna ait bilgileri tutar. Kullanıcı, her veritabanına ulaşmak istediğinde, kod ve şifrenin doğruluğu veri sözlüğünden kontrol edilir. Bir kere değiştiren şifrenin bir daha kullanılmasına izin verilmez.

Yetkisi olmayan veritabanı kullanıcılarına engel olmak için diğer bir yol da, kullanıcının, işletim sistemi tarafından doğrulanmasıdır. Oracle, kullanıcının doğrulaması için işletim sistemi tarafından saklanan bilgileri kullanabilir. Bunun şu yararlı olabilir;

- Kullanıcı, şifresiz ve kodsuz Oracle'a bağlanamaz
- Kullanıcı girişleri veritabanında tutulur ve işletim sistemi bunu bir dosyaya yazar.

Kullanıcının doğru erişimi, güvenlik domain'leri ile de kontrol edilebilir. Güvenlik domain'i, kullanıcı için sistem kaynak limitine, uygun disk boşluğuna karar verir. Her bir kullanıcı için güvenlik domain'leri şu bilgileri içerir:

- Kullanıcının default tablo iş alanı
- Kullanıcının geçici tablo iş alanı
- Tablo iş alanının boşluk miktarı

2) Haklar

Nesnelere erişen kullanıcılar, kendilerine verilen haklar (privilege) ile kontrol edilebilirler. Hak, nesnelere erişim iznidir.

Haklar şunları içerir;

- Veritabanına erişim hakkı
- Nesne yaratma hakkı
- Başkasının nesnesine erişim hakkı

Nesne yaratıcısı, nesnedeki verilere diğer kullanıcıların da erişmesine izin verebilir. Bu iznin verilmesi GRANT, iznin geri alınması ise REVOKE komutudur. Verilen izin sadece sorgu izni veya sadece güncelleme izni olabilir.

3) Roller

Oracle, rol kullanımı ile hak yönetiminin kontrolünü kolaylıkla sağlayabilir. Rol, kullanıcılara veya diğer rollere verilmiş haklar grubunun ismidir. Varolan Rol bilgileri veri sözlüğünde tutulur.

Rol'un şu özellikleri, veritabanında kolay bir hak yönetimi sağlar:

- Birden çok kullanıcıya aynı hakkı vermektense, birbiri ile ilişkili kullanıcı gruplarına rol verilir. Bir rol, sadece grubun üyelerine verilir.

- Bir grubun hakkı değişirse, sadece rol'un hakkı değiştirilir.

4) Kaynak limitleri

Her bir kullanıcı için kaynak limitinin belirlenmesi ile güvenlik yönetici kontrolsüz kaynak tüketimini engellemiş olur (CPU zamanı).

Oracle'ın bu özelliği, çok kullanıcılı sistemler için çok yararlıdır.

5) Veritabanındaki güncellemelerin dosyaya yazılması

Oracle'da bu durum, veritabanındaki kullanıcıların aktivitelerini gözlemek açısından bir güvenlik özelliğidir. Veritabanı yöneticisi, bunu aktif veya pasif yapabilir.

4.7. Bütünlük

Bütünlük, veritabanına geçersiz bilgi girişini engellemeyi sağlar. Bir ilişkisel veritabanında uygulanacak veri bütünlüğü tipleri şunlardır:

- not null sınırlaması
- tek (unique) kolon değerleri
- ana (primary) anaktar değerleri

Eğer bir DML komutunun icrası sırasında bütünlük bozulursa, bu komut kesilir ya da hata verir. Bütünlük sınırı, tablolara uygulanır. Örneğin EMP tablosundaki SAL kolonuna bir bütünlük sabiti (contraint) uygulansın. Buna göre bu kolon 10.000'den büyük bir değer içermesin. Eğer bir INSERT veya UPDATE komutu bunu bozuyorsa, kesilir veya hata verir. Böylelikle veriye ulaşım kontrol altına alınmış olur.

NOT NULL sınırlaması : Default olarak bir tablodaki kolonların null olmasına izin verilir. NOT NULL sınırlaması ile buna izin verilmez. Bir tablodaki bütün kolonlara bu sınırlama verilebilir.

TEK anaktar sınırlaması : Bu sınırlamaya göre, bir kolondaki değerlerden ikisi aynı olamaz. Bu kolona tek anaktar denir. Tek anaktar, birden fazla kolonu da ihtiva edebilir. Buna göre bu kolanların kombinasyonu bir tane olmalıdır.

ANA anaktar sınırlaması : Ana anaktar, tablonun ana anaktar sınırlamasının tanımını içeren kolondur. Veritabanındaki her bir table, bir tane ana anaktara sahip olabilir. Tablonun satırları ana anaktar sınırlaması ile belirlenen değerleri içerebilir. Bu sınırlama şunları garanti eder;

1) Belirtilen kolonlarda, tablonun satırları çift değer almaz.
2) Ana anaktar kolonu, null değeri almaz.

Bu tip sınırlamalar, index kullanımı ile uygulanır.

4.8. Oracle Dağıtık Sistem Mimarisi

4.8.1. Oracle İstemci/Sunucu Mimarisi

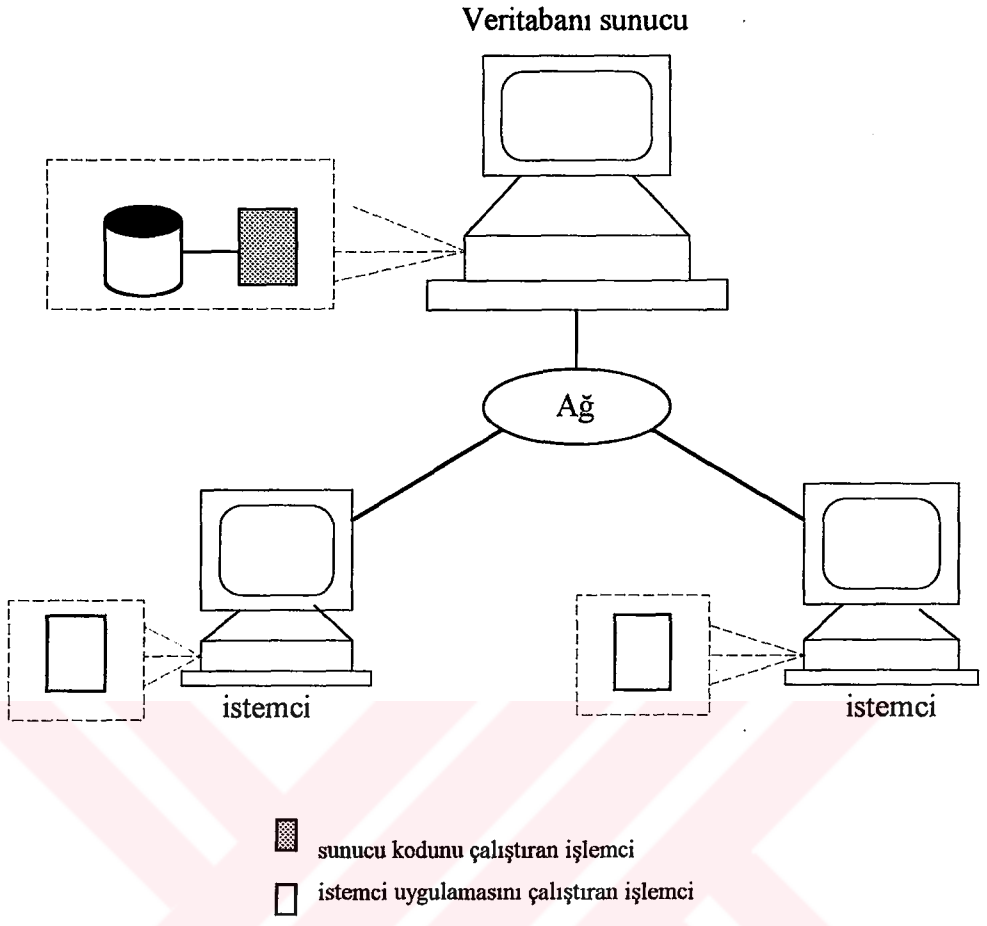
Oracle istemci/sunucu mimarisinde veritabanı ve uygulamaları iki kısma ayrılır:

- İstemci kısmı (front-end)
- Sunucu kısmı (back-end)

İstemci, veritabanına ulaşan veritabanı uygulamalarını çalıştırır, kullanıcı ile ekran, klavye gibi araçların bağlantısını sağlar [8].

Sunucu, Oracle yazılımını icra eder, eş zamanlılık ve veri erişimi için gerekli fonksiyonları çalıştırır.

Oracle ve istemci uygulamaları aynı makinada çalıştırılabilir. Ayrıca bir ağ ile birbirine bağlanmış farklı makinalar üzerinde çalışan sunucu ve istemci kısımları daha verimli çalışabilir.



Şekil 4.10. İstemci/sunucu mimarisi ve dağıtık ortam.

4.8.2. SQL*Net Mimarisi

SQL*net, birbirinden farklı ağları birbirine bağlayan, istemci/sunucu hareketinin şeffaf olarak gerçekleşmesini sağlayan Oracle ağ arabirimidir. Veritabanının TCP/IP, uygulamanın DECNET üzerinde olmasının önemi yoktur. Ağ iletişimi içinde, SQL*net, program arabiriminin bir parçasıdır. Program arabirimi ise bir veritabanı uygulaması ile Oracle arasındaki yazılım katmanıdır. SQL*net, dağıtık veritabanı için ağ üzerinden ağ protokollerini ve uygulama arabirimlerini kullanır. Ağ protokolleri, ağ üzerinde veri transferini sağlayan standartlardır. Ağ ortamlarında, Oracle sunucu, SQL*net kullanarak diğer Oracle sunucuları ve istemci iş istasyonları ile iletişim kurar. Mainframe'ler için PC Lan gerekir. SQL*net kullanarak, veritabanı uygulamalarında ağ iletişim problemi olmaz. Ağ, işletim sisteminden bağımsızdır.

SQL*net ve Oracle, dağıtık sorgulamaları sağlar. SQL*net ile bir kullanıcı, Oracle'daki veriye ulaştığı zaman, verinin lokal makinada ya da uzak makinada olmasının önemi yoktur.

SQL*net şu olanakları sağlar:

- Bulunan birim dışındaki birimlerin verilerine ulaşır.
- Bir veya çoklu birimde bulunan veriyi sorgu olanağı sağlar.
- Farklı birimlerdeki tablolar üzerinde, görüntü tanımlama imkanı verir.

SQL*net ve ağ bağımsızlığı

Veri, uzak veritabanından istendiği zaman, lokal veritabanı sunucusu, ağ, ağ haberleşme yazılımı ve SQL*net'ı kullanarak uzak veritabanı ile iletişim kurar. SQL*net, ağ üzerindeki farklı makinalardaki istemci ve sunuculara ulaşmak için arabirim olarak kullanıldığı gibi dağıtık hareketi gerçekleştirmek için ağ üzerinden veritabanına bağlanır.

SQL*net ve Oracle dağıtık veritabanı sistemi birçok ağ üzerinde çalıştırılabilir.

4.8.2.1. SQL*net Bileşenleri

SQL*net şu parçalardan oluşur:

- SQL*net arabirimi
- TNS (transparent network substrate)
- Oracle Protokol Adaptörleri

SQL*net arabirimi

Bu arabirim, TNS tarafından gönderilen veya gelen mesajları toplar. Arabirim kodu, SQL*net'ı kullanan bütün birimlerde bulunur. İstemci tarafında (uygulama programı) arabirim, uygulamadan gelen mesajları toplar ve dağıtım için TNS'e

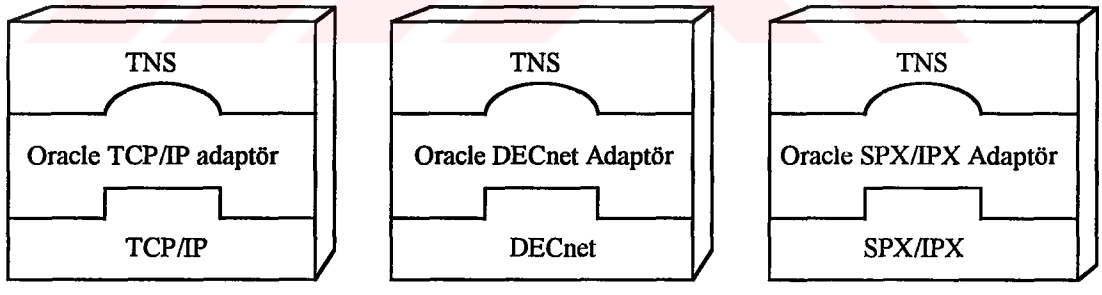
gönderir. Sunucu tarafında arabirim, TNS'den gelen mesajları alır ve Oracle'a gönderir.

TNS

TNS, makina bağlantısının olmadığı yerde birebir bağlantıya izin verir, farklı protokoller içeren heterojen bir ağ üzerinde kullanıcı-şeffaf bir katman sağlar. Farklı protokollerle birbirine bağlı farklı ağlar üzerinde bir şeffaf katman ihtiva eder ve bu ağlar üzerinde ağ uygulamalarına imkan verir.

Oracle Protokol Adaptörleri

Oracle protokol adaptörleri, TNS'e izin vererek, varolan ağ iletişim protokolleri üzerinden iletişim sağlar. Bu protokol adaptörleri, TNS'ın fonksiyonlarına eşdeğer protokol fonksiyonlarının planını yapar. Bir Oracle protokol adaptör, her bir uygulama program arayüzü (API) için mevcuttur.



Şekil 4.11. TNS, adaptörler ve protokoller.

Şekil 4.11, varolan ağ protokolü ile TNS ve Oracle protokol adaptörlerinin nasıl birleştiğini göstermektedir. Standart protokol üzerinde çalışan herhangi bir TNS istemci için, Oracle protokol adaptörü, protokolün tek API'sı ve Oracle'ın arabirimi TNS arasında arayüz olarak kullanılır.

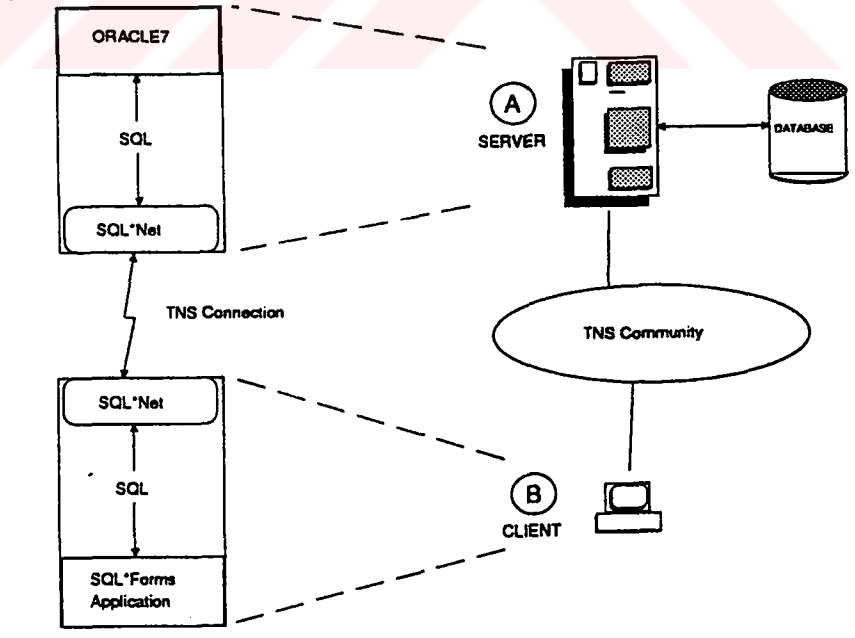
Çoklu protokolu ve protokol adaptörlerini destekleyen bir birim, çoklu TNS topluluğunun bir üyesidir. Bu gruba ait bir istemci TNS, şu instancelar için ortaktır;

- Birden fazla ağ üzerinde diğer uygulamalara ulaşmak için gerekli bir istemci uygulaması. Çoklu protokol ve protokol adaptör ile istemci herhangi bir sunucu uygulamasına bağlanabilir.

- Çoklu TNS'den istemciler tarafından erişilecek bir sunucu uygulaması. İki protokol ve protokol adaptör ile bütün istemciler makina üzerindeki bir sunucu uygulamasına erişebilir.

4.8.2.2. SQL*net Kullanımı

Bir kullanıcı, veriye ulaşmak istediğinde, SQL*net kullanımı ile verinin lokal veritabanında veya uzak veritabanında olmasının bir önemi yoktur.



Şekil 4.12.SQL*net kullanarak istemci/sunucu konfigürasyonu.

Buna göre, bir hareket şu aşamalardan geçer;

- İstemci, veri isteğinde bulunur.
- SQL*net bunu karşılar ve TNS'e gönderir.
- TNS bunu sunucuya gönderir.
- Sunucu tarafindaki SQL*net bu isteği alır ve Oracle'a gönderir.
- Oracle, isteği yerine getirir ve ilgili veriyi SQL*net'e gönderir.
- SQL*net, veri alır ve TNS'e gönderir.
- TNS, veriyi istemciye gönderir.
- İstemci tarafindaki SQL*net, veriyi kullanıcıya gönderir.

SQL*net, TNS ile farklı protokol kullanan ağların bağlantısını sağlar.

4.8.3. Ağ Domain

Ağ domain, birçok işletim sistemi tarafından kullanılan ağaç yapısına benzer ve ağ yapısı içinde veritabanının yerini belirlemek için kullanılır.

Ancak, dosya sistemlerinin tersine, ağ domainler, ağ üzerindeki diğer yapıların ve veritabanının fiziksel düzenine bağlı olabilirler veya olmayabilirler.

Tavsiye edilen, birbiri ile alakalı veritabanlarını aynı ağ domain içersine koymaktır. Bu ağ domain organizasyonunu sağlar ve karmaşayı önler.

Bir ağ domain servisi kullanılabilirse, ağ domaini yaratmak, silmek, ağ domainlerine erişimi kontrol etmek, ağ üzerindeki yapıları (veritabanı gibi) yaratmak, silmek ve ağ üzerindeki yapıların tek isimli olmasını sağlamak için kullanılır. Eğer kullanılmazsa, bütün konrol ağ yöneticisinindir.

Oracle mimarisi, X.25 gibi ağ domain servisinin kullanılması için tasarlanmıştır.

4.8.4. Nesne İsimleri

Oracle, global nesne isimleri kullanarak, dağıtık veritabanındaki SQL komutlarına referans olabilecek şema nesnelerine olanak sağlar. Bir nesnenin global ismi, nesne ismi, @ işareti ve veritabanı isminden oluşur.

SELECT * FROM scott.emp @sales.division3.acme.com

Bu komut ile sales veritabanındaki scott.emp isimli tablodan sorgulama yapılabilir.

Oracle, bir nesne yaratıldığında, nesnenin global nesne isminin tek olup olmadığını kontrol etmez. Fakat bir nesne adı, kendi yerel veritabanında tektir ve Oracle'da her bir veritabanının tek bir ismi vardır.

Bir global veritabanı ismi, DB_NAME ve DB_DOMAIN parametreleri kullanılarak oluşturulur. DB_NAME parametresi, veritabanının global nesne isminin, isim bileşenlerini, aynı zamanda en fazla sekiz karakterden oluşan lokal veritabanı ismini belirler [10].

DB_DOMAIN parametresi, ağdaki veritabanının mantıksal yerini (ağ domaini) tutar. Örnek verirsek;

DB_NAME = TEST
DB_DOMAIN = US.ACME.COM

Bu tanımlara göre global veritabanının adı;

TEST.US.ACME.COM'dur.

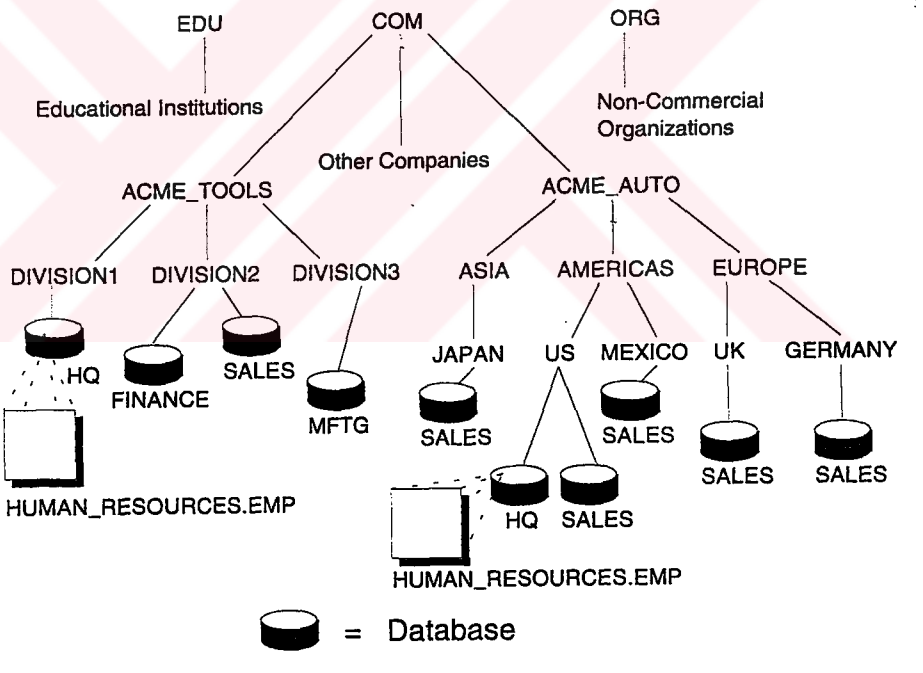
Veritabanı yaratılırken; DB_NAME'de belirtilen isim, veri dosyalarında, redo log dosyalarında ve kontrol dosyalarında tutulur. Veritabanı açılırken, veritabanı ismi

ve DB_NAME parametresinin değeri, kontrol dosyalarında bulunamazsa, veritabanı açılmaz.

Veritabanı yaratırken ağ domainini belirten DB_DOMAIN parametresi 119 karakteri aşamaz. Domain isimlerinin seviyesi (.) ile ayrılır.

Global veritabanı isminin ağ domain parçası Internet standartlarına uygulamalıdır.

Şekil 4.13 incelenirse, ağ üzerindeki birçok veritabanının adı aynı fakat ağ domain yapısının yerinden dolayı, her bir veritabanı, tek bri global veritabanı ismine sahiptir [8].



Şekil 4.13. Ağ ağaç yapısı ve global veritabanı isimleri.

hq.division1.acme_tools.com
sales.division2.acme_tools.com
hq.us.americas.acme_auto.com
sales.mexico.americas.acme_auto.com
sales.germany.europe.acme_auto.com

Bu şekildeki yapı ile, her bir veritabanı ve nesneleri tek olarak ifade edebilir. Oracle dağıtık sistemde, herbir lokal veri sözlüğü, global nesne isimlerini saklamaz, kendi nesne ve şema isimlerini saklar. Global isimler GLOBAL_NAME veri sözlüğünde bulunur. Şu komutla, bağlanılmak istenen veritabanlarının global veritabanı isimleri listelenir:

SELECT * FROM GLOBAL_NAME

Bazı durumlarda global veritabanı ismi değiştirilmek istenebilir. Bu değişikliğin yapılmasının şu etkileri olur;

- Uzak veritabanındaki varolan veritabanı bağlantıları bu veritabanını içeriyorsa, veritabanı bağlantılarının isimleri otomatik olarak değiştirilmez.

- Aynı şekilde synonymler, bu veritabanının nesnelerini içeriyorsa, synonym isimleri güncelleştirilmez.

- Uzak veritabanındaki SQL program ünitesi, bu veritabanındaki verileri kullanıyorsa, otomatik olarak güncelleştirilmez.

Bu durumda, veritabanının ismini değiştirmek için, ALTER DATABASE komutunun RENAME GLOBAL_NAME parametresi kullanılır.

4.8.5. Veritabanı Bağlantıları

Dağıtık sistemde, ağ domain servisi kullanılmazsa veritabanı bağlantıları kullanılır [8]. Bir veritabanı bağlantısı, uzak veritabanı için bir yol tanımlar. Bu bağlantı, dağıtık veritabanının kullanıcıları için şeffaftır. Çünkü bir veritabanı bağlantısının adı, global veritabanı ismi ile aynıdır.

Şu komut, lokal veritabanında bir veritabanı bağlantısı yaratır:

CREATE PUBLIC DATABASE LINK sales.division3.acme.com.

Örnekte, sales.division3.acme.com isimli veritabanı bağlantısı, aynı isimli veritabanına ulaşmak için bir yol tanımladı. Böylece lokal veritabanına bağlanan bir kullanıcı ve uygulama, global nesne isimlerini kullanarak sales veritabanındaki verilere ulaşabilir. Sales veritabanı bağlantısı, sales veritabanına bağlantıyı gerçekleştirmek için kullanıldı.

Bir SQL komutu, global nesne ismi içeriyorsa Oracle, global nesne isminde tanımlanan veritabanına karşılık gelen veritabanı bağlantısını arar. Eğer bu bağlantı bulunursa, lokal Oracle, SQL komutunun icrasını tamamlamak ve uzak veritabanına bağlantıyı sağlamak için bu veritabanı bağlantısını kullanır. Bulunmazsa, bağlantı kurulamaz ve SQL komutu icra edilemez.

Veritabanı bağlantı şekillerini üç kısma ayırabiliriz [9]:

1- Özel Veritabanı Bağlantısı: Özel bir kullanıcı adına yaratılır. Bu veritabanı bağlantısı sadece SQL komutundaki global nesne isminin sahibi tarafından kullanılır.

2- Genel Veritabanı Bağlantısı : Genel kullanıcı grubu için yaratılır. Genel veritabanı bağlantısı, bir SQL komutundaki global nesne ismine ait veritabanının herhangi bir kullanıcısı tarafından kullanılır. Lokal veritabanındaki bütün kullanıcılar, sistemin uzak veritabanındaki veriye ulaşabilirler.

3- Ağ Veritabanı Bağlantısı: Bir ağ domain servisi tarafından yaratılır ve yönetilir. SQL komutundaki global nesne isminin tanımlandığı ağdaki herhangi bir veritabanının herhangi bir kullanıcı tarafından kullanılır. Oracle bütün bu bilgileri veri sözlüğünde tutar.

Bir veritabanı bağlantısının tanımladığı yolun iki bileşeni vardır [8]:

- Veritabanı dizisi (string)
- Uzak yol (Remote account)

```
CREATE PUBLIC DATABASE LINK sales.division3.acme.com
        CONNECT TO guest IDENTIFIED BY password
        USING 'dbstring'
```

Bu örnek, SALES veritabanı bağlantı yolunun nasıl tanımlandığını gösteriyor. Bu tanımlamadaki dbstring uzak bağlantıları sağlamak için kullanılır, bununla bağlantı özellikleri (ağ protokolleri gibi) belirtilir ve işletim sistemi ağ bağımsızlığı sağlanır. Bir dizi, veritabanına erişmek için SQL*net bağlantısını ve işletim sistemi bilgisini içerir. Fakat, kullanıcı başka bir bağlantıda tanımlandığına güvenerek veritabanı dizisini gözardı edebilir.

Seçimli olan CONNECT TO komutu, uzak veritabanında bir oturum yaratıldığı zaman kullanılmak üzere kullanıcı kodu ve şifresi tanımlar. Tanımlanmazsa yerel kullanıcı kodu ve şifresi kullanılır.

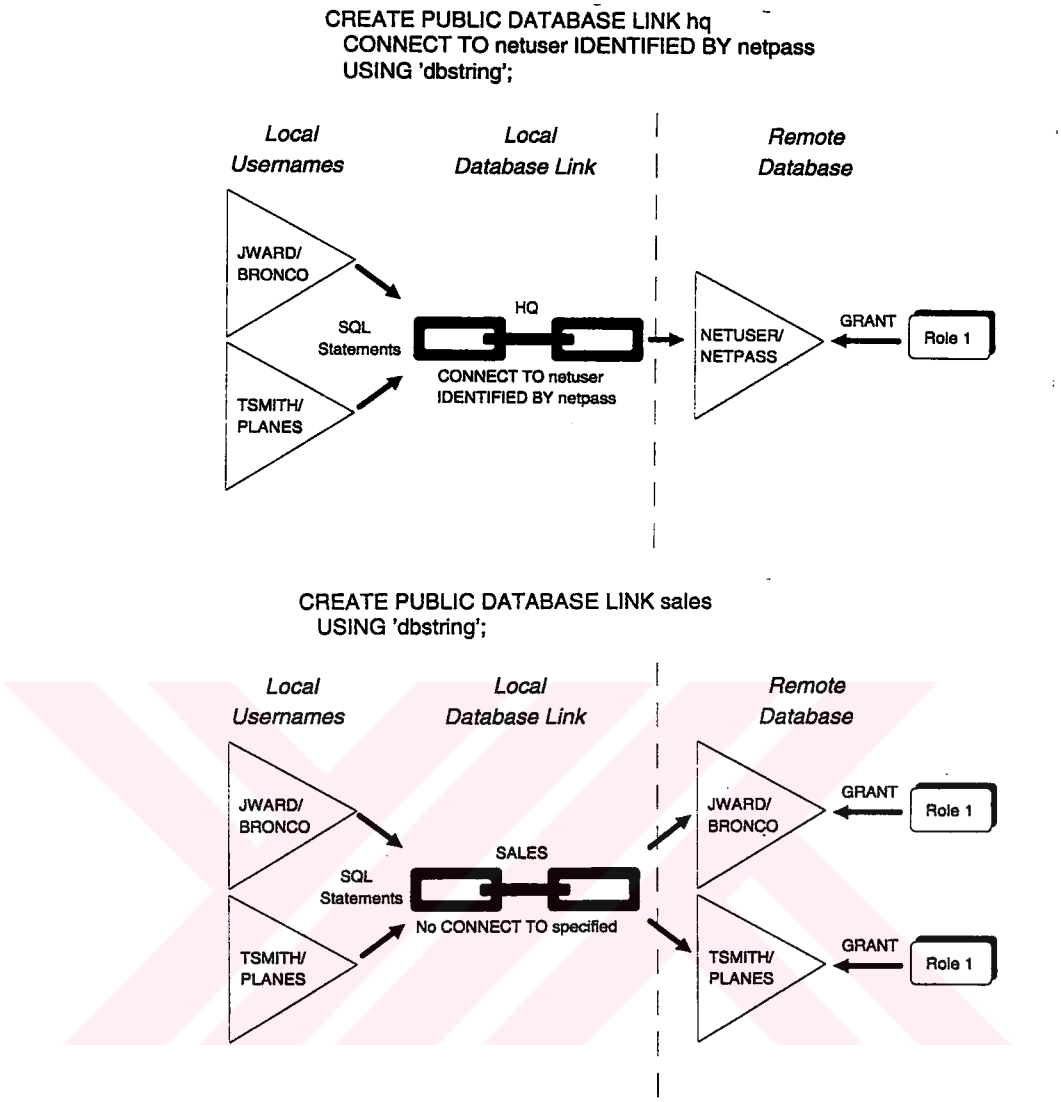
Örneğin, satış bilgilerini tutan uzak veritabanının adı sales.acme.com olsun.

```
CREATE PUBLIC DATABASE LINK sales.acme.com
USING 'dbstring'
```

CONNECT TO cümleciği olmadığı için, veritabanı bağlantısı kullanılarak, Oracle uzak veritabanına ulaşmak için yerel kod ve şifreyi kullanır.

Veritabanı bağlantısı ya merkezi (central) uzak yol ya da bireysel (individual) uzak yol kullanılarak tanımlanır.

Bir merkezi uzak yol kullanan genel veritabanı bağlantısı yerel veritabanının herhangi bir kullanıcısına, uzak veritabanına ulaşma izini verir. Bir merkezi yol kullanımının avantajlarından biri de hak ve rollerin bir kere verilmesidir.



Şekil 4.14. Merkezi ve bireysel uzak yol bileşeni.

Bireysel uzak yol ise, özel veritabanı bağlantı tanımındaki uzak kod ve şifre saklanmadığı için daha az güvenlidir.

Ayrıca uzak güvenlik yöneticisi, bağlantı kurmak isteyen kullanıcılara hak ve rol yaratmak zorundadır.

Bazı durumlarda, kullanıcılar farklı haberleşme yolları (pathway) kullanarak uzak veritabanlarına bağlantı isteyebilirler. Örneğin, uzak veritabanı ORACLE paralel sunucusu kullanıyorsa, kullanıcı uzaktaki veritabanının instance'ına bağlantıyı

sağlamak için lokal birimde bazı genel veritabanı bağlantıları tanımlamak isteyebilir. Bu istekleri karşılamak için, Oracle, veritabanı bağlantı ismindeki seçimlik "connection qualifier" ile veritabanı bağlantısı yaratmaya izin verir.

"Connection-qualifer" kullanımı ile aynı veritabanı üzerinde çoklu veritabanı bağlantısı gerçekleşebilir. Örneğin aynı veritabanına ulaşan Oracle paralel sunucunun iki farklı instance'ı için çoklu veritabanı bağlantısı bu yolla sağlanabilir.

Örnek olarak, HQ.ACME.COM uzak veritabanı, Oracle paralel sunucusu kullansın ve veritabanı, hq-1 ve hq-2 isimli iki instance'a sahip olsun. Lokal veritabanı, HQ veritabanının iki uzak instance'ına ulaşım yolunu tanımlayan şu genel veritabanı bağlantılarını içerebilir.

```
CREATE PUBLIC DATABASE LINK hq.acme.com @ hq-1
   USING 'string-to-hq-1';
CREATE PUBLIC DATABASE LINK hq.acme.com @ hq-2
   USING 'string-to-hq-2';
CREATE PUBLIC DATABASE LINK hq.acme.com
   USING 'string-to-hq';
```

Bir SQL komutu, bir veritabanı bağlantısı içerecekse, önce isminin kısmi mi, tam mı olacağı belirlenir [11].

- Tam olması : Eğer veritabanı bağlantısı, veri sözlüğünde tutulan veritabanı, domain ve seçimlik "connection-qualifier" bileşenlerini içeriyorsa tamdır.

- Kısmi olması : Veritabanı ve "connection-qualifier" bileşenlerini içeriyor fakat domain bileşinini içermiyorsa kısmidir.

Oracle, uzak veritabanına bağlanmadan önce şu işleri icra eder;

- Komutta belirtilen veritabanı bağlantı ismi kısmi ise, Oracle bu ismi, lokal veritabanının domain'ını içerecek şekilde genişletir.

- Oracle, komuttaki veritabanı bağlantısı ile aynı isimde olan kendi şemasında özel veritabanı bağlantısını arar.

a) Oracle, veritabanı bağlantısında kullanıcı kodu ve şifresi belirtilmişse bunu kullanır, eğer kullanılmamışsa aktif olan kullanıcı kodu ve şifresini kullanır.

b) Oracle, veritabanı bağlantısı dizi içeriyorsa bunu kullanır. Eğer yoksa, sonraki genel veritabanı bağlantısını arar. Böylece bir bağlantı yoksa veya dizi belirtilmemişse hata verir.

- Oracle, uzak veritabanına ulaşmak için dizi kullanır ve uzak veritabanına ulaştıktan sonra şu şartları kontrol eder;

a) Uzak veritabanı ismi ile bağlantıda belirtilen veritabanı ismini karşılaştırır.
b) Uzak veritabanının domaini ile bağlantının domainini karşılaştırır.

Eğer bunlar doğrulanırsa, belirtilen kullanıcı kodu ve şifresi ile bağlantı tamamlanır. Doğrulanmazsa Oracle hata verir. Şimdi bunları örneklerle anlatalım.

Örnek 1 : Bu örnek, özel ve genel veritabanı bağlantısı kullanarak, uzak veritabanına bağlanma yollarını gösteriyor. EMP uzak tablosu, TSMITH şemasında bulunuyor ve şu komutlar yerel veritabanında icra ediliyor [8].

```
CREATE PUBLIC DATABASE LINK sales.division3.acme.com
    CONNECT TO guest IDENTIFIED BY network
    USING 'dbstring';
CONNECT jward/bronco;
CREATE DATABASE LINK sales.division3.acme.com
    CONNECT TO tsmith IDENTIFIED By radio;
UPDATE tsmith.emp @ sales.division3.acme.com
  SET deptno=40
  WHERE deptno=10;
```

Önce Oracle, UPDATE komutunda belirtilen global nesne ismini fark eder. Sonra lokal veritabanında bu isimle veritabanı bağlantısını arar. Bulduğu JWARD. SALES.DIVISION3.ACME.COM özel veritabanı bağlantısı, uzak veritabanı SALES'a tam bir yol belirtmemektedir. Sadece uzak yoldur. Bu yüzden Oracle genel veritabanı bağlantısı arar. Uzak yol ile bu genel veritabanı bağlantısının dizisini birleştirerek TSMITH/RADIO kullanıcı ile SALES veritabanına bağlantıyı gerçekleştirir. Uzak Oracle, EMP tablosundaki nesne referansını çözümler. TSMITH şemasını ve EMP tablosunu bulur, komutu işletir ve sonuçlar lokal veritabanına geri döner.

Örnek 2 : Örnek 1'de olduğu gibi uzak tablo EMP, TSMITH şemasında olsun ve genel synonym ismi EMP olarak tanımlansın. Genel veritabanı bağlantıları Örnek 1'deki bağlantıların aynısı olsun.

Şu komutlar lokal veritabanında gerçekleşiyor;

```
CONNECT scott/tiger;
CREATE DATABASE LINK sales.division3.acme.com;
DELETE FROM emp @ sales
WHERE empno=4299;
```

Oracle, DELETE komutundaki global nesne ismini farkeder. Sonra, lokal veritabanının ağ domainini kullanarak global nesne ismini tamamlar.

```
DELETE FROM emp @ sales.division3.acme.com
   WHERE empno=4299;
```

Daha sonra lokal veritabanında bu isimle veritabanı bağlantısını arar. Bulduğu bağlantı, özel veritabanı bağlantısıdır ve yol belirlememektedir. Oracle lokal kullanıcı kodu ve şifresini kullanarak genel veritabanı bağlantısı arar. Tam yol genel veritabanı bağlantısının dizisi ile sağlanır ve SCOTT/TIGER olarak uzak veritabanına bağlantı sağlanır. Uzak Oracle, SCOTT şemasında EMP nesnesini arar ve bulamaz. Bunun

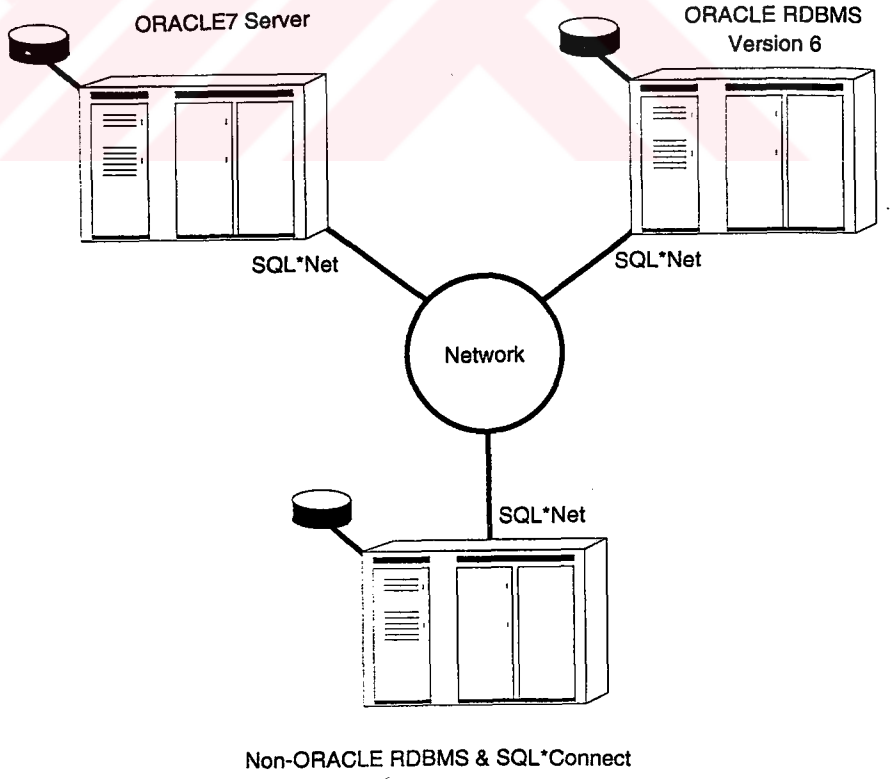
üzerine genel synynm olarak tanımlanan EMP tablosunu bulur, komutu icra eder ve sonuçları lokal Oracle'a gönderir.

4.8.6. Oracle Dağıtık Veritabanı Mekanizması

Dağıtık bir sistemde, Oracle veritabanına bağlanan bir kullanıcı, uzaktaki bir veriye, SQL komutu ile şu şekillerde ulaşır:

1) Veri başka bir Oracle veritabanında olsun. Her Oracle veritabanı bir ağ ile birbirine bağlıdır ve SQL*net ile haberleşme sağlanır.

2) Veri, Oracle olmayan bir veritabanında olsun. Bu farklı veritabanı Oracle'ın gateway mimarisini (SQL*connect) desteklesin. Oracle ve Oracle olmayan veritabanları bir ağ üzerinden birbirine bağlanır ve SQL*net ile haberleşme sağlanır.



Şekil 4.15. Heterojen dağıtık veritabanı sistemleri.

Bir Oracle biriminden uzak bir birime (Oracle veya değil) bağlantı sağlandığında, bağlanılan Oracle birimi, ilgili gateway'lerin ve her bir uzak sistemin kapasitelerini tutar. SQL komutu çalıştırılır. Dağıtık sistemde, SQL komutu Oracle veritabanı tarafından icra edilir. Çünkü uzak veritabanı ilgili sunucusu, gateway ve uzak veritabanı sunucusunun kapasitesi tarafından sınırlıdır. Örneğin, eğer dağıtık bir sorgu, ilave (extented) SQL fonksiyonları içeriyorsa, bu fonksiyon, lokal Oracle veritabanı tarafından çalıştırılır. Uzak güncellemelerin olduğu ilave SQL fonksiyonları bütün gateway'ler tarafından desteklenmez.

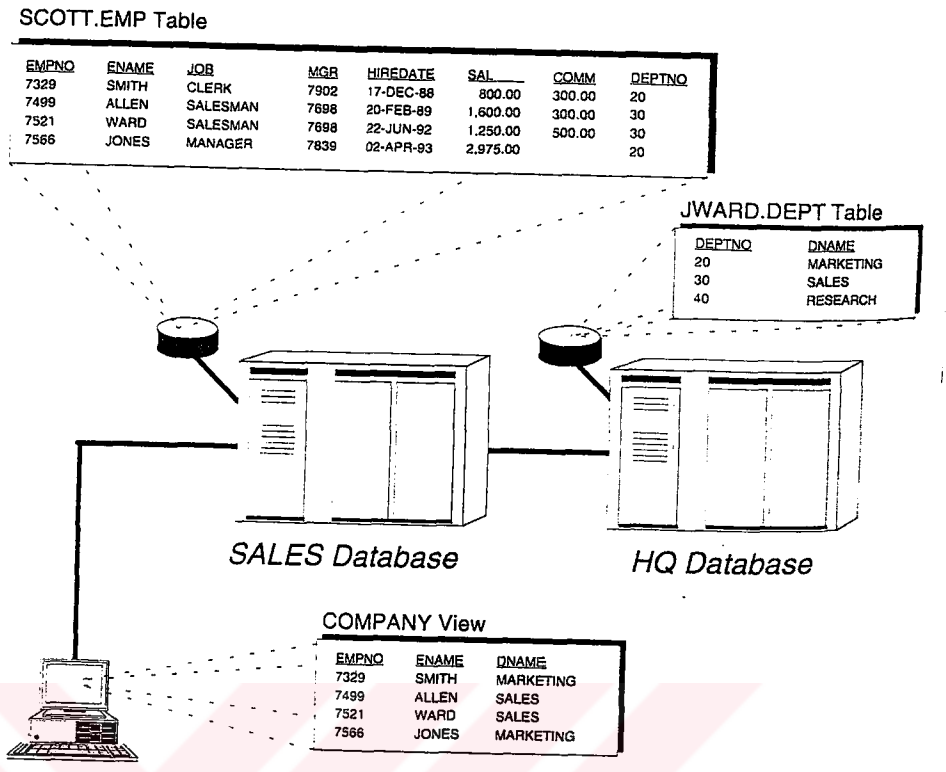
4.8.7. Oracle Dağıtık Veritabanı Şeffaflığı

Görüntü ve Yer Şeffaflığı

Lokal görüntüler, lokal ve uzak tablolar için yer saydamlığını sağlamak için kullanılır. Mesela, EMP tablosu, lokal veritabanında saklansın. DEP tablosu da uzak bir veritabanda saklansın. Lokal veritabanında, bu veritabanlarındaki veriyi birleştiren bir görüntü yaratılsın. Kullanıcı bir görüntüye ulaştığında, bunun fiziksel olarak nerede tutulduğunu bilmez (Şekil 4.16).

```
CREATE VIEW company AS
        SELECT empno, ename, dname
        FROM scott.emp a, jward.dept @ hq.acme.com b
        WHERE a.deptno=b.deptno;

SELECT * FROM company;
```



Şekil 4.16. Görüntü ve yer şeffaflığı.

Synonym ve yer şeffaflığı

Synonym'ler dağıtık ve dağıtık olmayan sistemler için çok faydalıdır. Böylece mevcut nesnelerin dağıtım sistemdeki yeri ve tanımı saklanmış olur. Eğer nesnenin adı değişecekse veya yeri değişecekse sadece synonym yeniden tanımlar. Örneğin; HQ veritabanında saklanan işçi bilgilerini tutan SCOTT.EMP tablosu için, dağıtık sistemin her bir veritabanında, genel bir synonym yaratılsın.

CREATE PUBLIC SYNONYM emp for SCOTT.EMP @ hp.acme.com

Genel Synonym ile EMP tablosunun yeri saklandığı için, uygulamanın nerede olacağını bilmeksizin bir işçi uygulaması tasarlanabilir.

Uygulamanın SQL komutu, bu EMP genel synonym'ı ile SCOTT.EMP @ HQ.ACME.COM tablosuna ulaşabilir. Eğer tablonun yeri değiştirilirse, genel synonym, sistemin birimi üzerinde değiştirilmelidir.

Sorgu ve güncelleme şeffaflığı

Oracle, uzak bir veritabanındaki veriye ulaşmak için INSERT, UPDATE, DELETE, SELECT gibi standart DML komutlarına imkan sağlar. Bir sorgu, birçok uzak veya lokal tablo ve görüntüleri içerebilir.

```
SELECT empno, ename, dname,
  FROM scott.emp @ sales.acme.com a,
       jward.dept @.acme.com b
WHERE a.dept no= b.deptno;
```

Bir uzak veritabanındaki veriyi güncellemek için herhangi bir program kodu yazmaya gerek yoktur.

```
INSERT INTO scott.emp @ sales.division3.acme.com
  SELECT * FROM jward.emp;
```

Bu komut ile, lokal veritabanındaki jward.emp tablosunun kayıtları, SALES isimli uzak veritabanındaki Scott.emp tablosuna eklenir.

Fakat CREATE, ALTER, DROP gibi DDL komutları uzak bir veritabanı için çalıştırılamaz. Oracle buna izin vermez.

Hareket şeffaflığı

Oracle veritabanında icra edilen bütün hareketler, veritabanı dağıtık olsun olmasın ya onaylanır ya da onaylanmaz. Oracle dağıtık mimarisi, aynı zamanda SAVEPOINT, ROLLBACK TO SAVEPOINT komutlarını da destekler. Dağıtık bir hareket, onay ile biterse, Oracle'ın iki aşamalı taahhüt mekanizması başlatılır. İki aşamalı taahhüt mekanizması, hareket icra edilirken ağda bir aksaklık olsa bile, onay veya kesilme ile biten hareketin ilgili birimlerini garanti altına alır.

BÖLÜM 5 UYGULAMA HAKKINDA BİLGİ

Bu bölümde, tez konusu olan uygulama ile ilgili açıklayıcı bilgiler verilmektedir. Sigorta sektörü hakkında temel bilgiler anlatılmakta ve dağıtık yapı içinde nasıl bir organizasyonun oluşturulduğu hakkında temel program örneklerine yer verilmektedir.

5.1. Proje Konusu ve İş Akışı

5.1.1. Proje Konusu

Proje, UNIX ortamında ORACLE veritabanı ile yazılmış sigorta paket programı dahilinde poliçe ve reasurans işlemlerini içermektedir. Bu paket program tamamıyle parametrik bir altyapısı olan bütünleşik bir yapıya sahip bir yazılımdır. İçeriğinde teknik servis, muhasebe servis ve hasar servis işlemleri gerçekleşmektedir.

Bu yapı içinde Genel Müdürlükçe, önce bu servislere referans olacak parametrik bilgiler girilir. Bu parametrik bilgiler, sigortacılığın temel taşlarını oluşturur. Bu temel taşlardan en önemlisi poliçedir. Poliçe, sigorta ile ilgili tüm bilgilerin dökümüdür. Çeşitli rizikolara karşı hazırlanan poliçelerin kapsadığı şartlar sigorta şirketleri arasında farklılık göstermez. Eğer bir şirket daha önce uygulama alanı bulmamış bir sigorta dalında çalışmak isterse, uygulayacağı genel şartları hazırlayarak Bakanlığın ilgili organına tastik ettirir. Bu şartlar tasdik edilmedikçe poliçe düzenlenemez. Sigorta akdi, poliçenin bitim tarihinde sona erer. Ancak yangın, kasko, trafik gibi sürekli nitelikteki poliçelerde, sürenin bitmesinden bir ay önce şirket yeni bir poliçe veya tecditname hazırlayarak acentelerine gönderir.

Sigortalı ve sigortacı arasındaki bağlantı taraflardan birinin doğrudan teşebbüsü sonucu sağlanacağı gibi, genellikle de sigorta aracıları vasıtaları ile temin olunur. Bu sigorta aracıları, acentalardır.

Acentalar ikiye ayrılır. B acentaları, sigorta şirketi tarafından yetkili kılındıkları branşlarda poliçe düzenlerler. A acentaları, poliçe düzenleyip vergileri ödeyip, poliçe primlerini tahsil ederler. Acentalık mukavelesinde, sigorta ettirenden tahsil edilen primlerin acenta tarafından şirkete hangi tarihte ödeneceği belirlenir. Bu tarihte acenta, primleri, sigorta şirketine nakden ödemek veya şirket adına bankaya yatırmak zorundadır. A acentaleri ayrıca rejistro (kayıt) defteri tutmak zorundadır. Acentalar, bunun karşılığında komisyon alırlar.

Poliçe primi ise sigortacının hasar halinde ödeyeceği tazminata karşılık olarak, sigorta ettiren tarafından peşinen veya taksitle ödenen ücrettir. Bu ücret, teknik olarak hesaplanan prime vergi, komisyon gibi unsurların eklenmesiyle bulunur.

Prim, poliçe başlangıcından örneğin 20 gün sonra ödenecekse, sigortacının sorumluluğunun başlangıcı, sigorta başlangıç tarihinden 20 gün sonra başlayacaktır. Sigorta ücretinin tamamı veya taksitle ödenmesi kararlaştırılmış ise ilk taksit en geç poliçenin ibrazında, diğer taksitler de poliçede belirtilen tarihlerde ödenir.

Sigorta bedeli, sigortalının beyan ettiği ve poliçede yazılı olan meblağdır. Sigorta tazminatı, hasarın gerçekleşmesi halinde sigortacının ödeyeceği miktardır.

Sigorta türlerine branş denir. Yangın, nakliyat, oto, otodışı birer branş örneğidir. Bu branşlar altında sigortalanan risklerin her birine teminat denir. Tarifeler, teminatların konbine edilmiş halidir.

Poliçe düzenleniş sebebine göre isimlere ayrılabilir. Mesela, bir depo içindeki mallar sigortalanıyorsa buna fulotal poliçe denir. Burada depodaki malın sabit bir değeri olmadığı için sigorta ettiren kişi her ay, ne kadar mal kaldığını bildirmek durumundadır. Emtea poliçesi kamyon, tren, gemi, uçak gibi nakil aracına yüklenerek, bir yerden başka bir yere nakledilen mala teminat veren poliçedir. Grup

poliçesi, birden çok kişiye ait poliçedir. Burada tek bir poliçe hazırlanır. Bu grup poliçesi, bir handaki bürolar içinde hazırlanabilir.

Poliçe üzerinde bir değişiklik yapılacaksa zeylname adı altında yeni bir poliçe kesilir.

Diğer bir önemli konu reasurans kavramıdır. Reasurans, sigortacının üzerinde tuttuğu bir rizikoyu, prim karşılığında, meydana gelebilecek hasarlara da hisse oranında katılacak diğer şirket ve şirketlerle paylaşmasıdır. Bu yolla, sigorta şirketi hem kendi hem de sigortalının emniyetini sağlar. Reasurans, sigorta şirketleri ile reasurörler ile yapılan uluslararası bir alışveriştir, en kısa tanımı ile sigortacının sigortalanmasıdır. Şirketlere yapılacak dağılımlar teknik parametrelerde girilen oranlar bazında yapılır. Bu oranlar ise reasurans şirketleriyle yapılan anlaşmalarla Genel Müdürlük tarafından belirlenir. Bu anlaşmalara trete anlaşmaları denir.

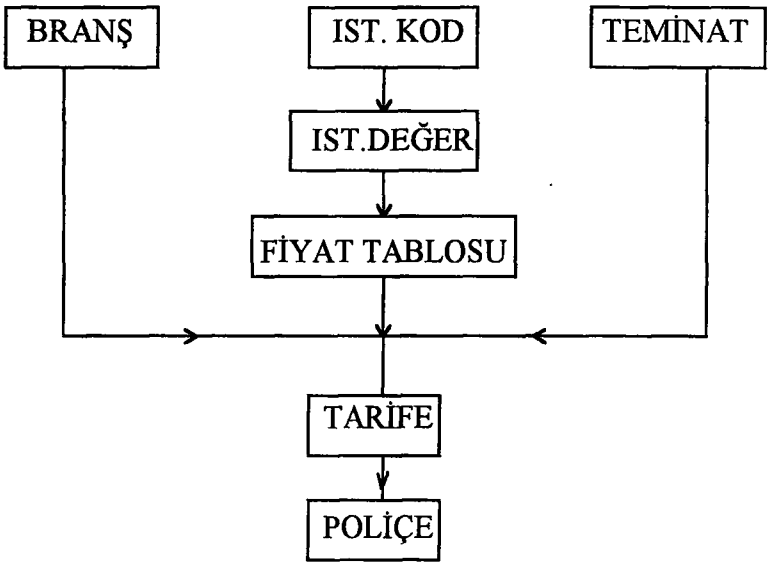
Genel Müdürlük poliçeleri ve acentalardan gelen poliçeler doğrultusunda riskler reasurörlere dağıtılır.

Anlatılan bütün bu kavramların parametrik hale dönüştürülmesi sonucunda servis işlemleri gerçekleştirilir. Proje konusu olarak Genel Müdürlük ve acentaların poliçe ve reasurans işlemleri detaylıca tasarlanmıştır. Ama genel yapı itibariyle paket program hakkında da kısaca bilgi verilmektedir.

5.1.2. İş Akışı

Yapılan işler, teknik, muhasebe ve hasar servisleri tarafından yürütülür.

Her birim için, Genel Müdürlük tarafından girilen parametreler baz alınır. Bunlar teknik ve muhasebe parametreleridir. Teknik parametreleri poliçe ve reasurans işlemleri için baz alan daha önce anlattığımız branş, teminat, tarife, fiyat tablosu ve reasurans dağılım oranları oluşturur.



Şekil 5.1. Poliçe bileşenleri.

Tarife üzerinde belirlenen fiyat tablosu poliçe priminin hesaplanmasında kullanılır.

Her teminatın bir fiyatı vardır. Bunu belirleyen faktörler istatistik kod ve istatistik değerlerdir. Deprem için örnek verirsek; binanın bulunduğu yerin deprem derecesi, binanın yapı tarzı, binanın kat adedi birer ist. koddur. Binanın adedinin belirlenmesi ise istatistik değerdir. Bunların yanyana gelmesiyle fiyat tablosu oluşur. Teminattan teminata değişir ve prim hesaplamasında kullanılır.

Örnek olarak yangın sigortasındaki bir binanın temel fiyatlarını inceleyelim:

1) Yapı tarzların göre
   a) tam kagir yapı
   b) adi kagir yapı
   c) kagir olmayan yapı
2) Kullanılmış tarzına göre,
   a) Sivil
   b) Ticari

Bu istatistik kod ve istatistik değerler ışığında fiyat tablosu şöyle oluşturulabilir.

Tablo 5.1. Fiyat tablosu.

İnşa Tarzları	Sivil Fiyatlar %	Ticari Fiyatlar %
Tam kagir	0.30	0.62
Adi kagir	0.60	1.24
Kagir olmayan	2.10	4.85

Tam kagir bir binanın bir sene için 30.000.000 TL üzerinden yangına karşı sigortası istensin. Bu durumda prim şöyle hesaplanır.

Fiyat = %0.30 (fiyat tablosundan)

30.000.000 * % 0.30 = 9.000 → net prim

\+ 900 → YSV (%10)

9.900

\+ 495 → Gider vergisi (%5)

10.395 brüt prim

Eğer surprim tanımlanmış ise belirtilen oran üzerinden brüt prime eklenir. Şekil 5.2'de anlattığımız kavramları üzerinde taşıyan bir poliçe örneği görülmektedir.

Poliçenin vade tarihleri poliçe üzerinde tanımlanan tahsilat kodu bilgisine göre bulunur. Mesela girilen bir tahsilat koduna göre %25 peşin, kalanı 4 eşit taksit yapılabilir. Bu şekilde tahsilat kodları tanımlanarak poliçe üzerine taşınır. Eğer tahsilat kodu tanımlanmazsa başlangıç tarihinden bitiş tarihi çıkarılır ve tek vade olarak poliçe basılır. Bu tarihlere göre prim ise; gün sayısı/toplam gün sayısı * prim formülü ile hesaplanır.

KASKO POLİÇESİ (.)

ACENTA : 1-3-8883 TARİFE : 420 POLİÇE NO :6669-0

SİGORTA ETTİREN :	RİZİKO TARİHLERİ :	SİGORTALI
ADI : Sevil Ertürk	YÜKLEME TARİHİ :	ADI : Sevil Ertürk
ADRES : İstanbul	BAŞLAMA TARİHİ : 01/01/1995	ADRES : İstanbul
	BİTİŞ TARİHİ : 01/01/1996	
	TANZİM TARİHİ : 01/01/1995	

POSTA KODU	: 80840	POSTA KODU : 80840
İLÇE / İL	: Ortaköy/İstanbul	İLÇE / İL : Ortaköy/İstanbul
ÜLKE	: TÜRKİYE	ÜLKE : TÜRKİYE

TEMİNATLAR :	SİGORTA BEDELİ :	MUAFİYET :	FİYAT :	NET PRİM :
KASKO	2,500,000,000.0	0.00	45.00	112,500,000.00
		TOPLAM NET PRİM	:	112,500,000.00
		GİDER VERGİSİ		5,625,000.00
		BRÜT PRİM	:	118,125,000.00

BİLGİ : AÇIKLAMA :

POLİÇENİN ÖDEME PLANI :

VADE :	TUTAR :
01/02/1995	118,125,000.00

DÜZENLEME TARİHİ : 01/01/1995 DÜZENLEME YER : İSTANBUL

EGE SİGORTA

Şekil 5.2. Poliçe örneği.

Sigorta şirketleri teminat verdikleri rizikoların önceden belirlenmiş kısmını üzerinde tutarlar. Buna konservasyon denir. Konservasyonun Sigorta Mürakaba Kurumuna bildirilmesi gerekmektedir. Diğer kısımlarda reasurans şirketleri arasında paylaşılır. Bu reasurans şirketlerinin girilmesi ve oranların belirlenmesi bu kısma girer. Daha sonra muhasebe parametreleri girilir.

Muhasebe parametrelerini ekno, kaynak, müşteri tanımlamaları, vergi tanımlama, müşteri hesabı açma gibi bilgiler oluşturur. Ekno tanımlamadaki amaç, acentanın cari hesabını takip etmek içindir. Ekno ile cari hesabının prim borcu mu,

komisyon mu, vergi mi olduğu anlaşılır. Kaynak kod, müşterinin veya acentanın türünü belirler. A acentası mı, B acentası mı bunu kaynak kod ile anlayabiliriz.

Acentalar girdikleri poliçenin bedel karşılığını senet veya çek olarak verirler. Bunun karşılığı bankadan alınır. Bu bankaların tanımı ise müşteri tanımlama başlığı altında gerçekleşir ve bunlara ait bir hesap açılır. Hesap tanımlama sabittir. Bu bölümde aynı zamanda çek ve senet girişi gerçekleşir.

Vergi bilgileri ise üç tanedir. Bunlar, gider vergisi, yangın sigorta vergisi (YSV) ve garanti fonudur. Yangın sigorta vergisi, belediye sınırları içinde bulunan rizikoların sigortalarında sadece yangın primine %10 oranında uygulanır. Ek teminatlar primine uygulamaz. Gider vergisi, saptanan toplam net prim ile yangın sigorta vergisi toplamı üzerinde %5 oranında uygulanır.

Garanti fonu, her yıl sigorta şirketlerince trafik sigortası için tahsil edilen primlerin %1 ile sigorta yaptıranların primlerin %2'si oranında sigorta şirketine ayrıca ödeyecekleri katılma paylarından oluşur. Yalnız trafik primlerinde uygulanır.

Bütün bilgiler ışığında teknik, muhasebe, hasar servis işlemleri gerçekleşir. Teknik servis, poliçe girişi, poliçe güncelleme ve zeyl girişi, reasurans dağılımından sorumludur.

Reasurans işlemlerinde temel kavramlar şunlardır;

Reasürör, reasurans işi kabul eden sigortacı veya şirket, sedan, reasurans işi veren sigortacı veya şirkettir. Bu şirket veya sigortacılara trete denir. Konservasyon tretesi daha önce anlattığımız gibi sigorta şirketinin kendisidir, yani sedandır. Buna göre konservasyon miktarı, sigorta şirketinin kendi üzerinde tuttuğu miktardır. Sedan ile reasürörler anlaşmasına göre girilen oranlarda primler, hasar olma durumunda hasar bedeli paylaşılır. Dağılım yapılırken poliçenin üzerinde tanımlanan poliçe cinsi önemlidir. Çünkü bu bilgi üzerinde prim mi, bedel mi dağıtılacak ayrımı vardır. Eğer

bedel dağıtılmayacaksa tüm prim konservasyona verilir. Her ikiside dağıtılacaksa, treteler arasında paylaştırılır. Bunları örnek ile açıklayalım;

Sedanın saklama payı her poliçede %40 olarak belirlensin. Kalan bakiye %60, reasürör A %40, reasurör B %20 olmak üzere iki reasürör arasında paylaştırılsın. Policenin bedeli 60.000.000 TL, primi 120.000 TL, poliçenin yıl içinde 750.00 TL hasarı mevcut olsun. Buna göre dağılım şu şekilde gerçekleşir;

a) Sorumluluğun paylaşımı

Sedan	%40	=	24.000.000 TL
Reasürör A	%40	=	24.000.000 TL
Reasürör B	+%20	=	+ 12.000.000 TL
	%100		60.000.000 TL

b) Primin paylaşımı

Sedan	%40	=	48.000 TL
Reasürör A	%40	=	48.000 TL
Reasürör B	+%20	=	+ 24.000 TL
	%100		120.000 TL

c) Hasar paylaşımı

Sedan	%40	=	300.000 TL
Reasürör A	%40	=	300.000 TL
Reasürör B	+%20	=	+ 150.000 TL
	%100		750.000 TL

Eğer hala bir risk sözkonusu ise kalan kısım ihtiyari tretesine devredilir. İhtiyari, "seçimlik" anlamında kullanılır. Yani sedan, ortaklığı istediği reasürör'e önermekte serbesttir. Reasürör'un kendisine önerilen ihtiyari reasurans işini kabul etmesi, sedan'ın hazırlayacağı bir döküman üzerine imza atması ile geçerlilik kazanır. Bu dökümana "bülten" denir ve Genel Müdürlük tarafından bir rapor ile hazırlanır. Üzerinde sigortalı ile ilgili tüm özellikler ve hisseler belirtilir. Böylece ihtiyari oranı da

belirlenerek dağılım gerçekleşir. Bu işlemlerin yapılması Genel Müdürlük bazında olur ve reasurans kapatılır. Reasuran oranları her bir trete için girilrken komisyon oranı ve komisyon matrahı da tanımlanır. Reasurans kapatırken;

(komisyon oranı*prim)/matrah

işlemi ile her bir tretenin komisyonu hesaplanır. Eğer trete, ihtiyacı ise bu, acentanın komisyon bilgisinden hesaplanır. Çünkü ilgili ihtiyari şirket, sisteme müşteri bazında girilir ve trete üzerinde tanımlanan kaynak kodu ile müşteri kodu yakalanarak komisyon bilgisine ulaşılır.

Bu komisyon, reasürörün elde ettiği primler karşılığında sedan'a ödediği komisyondur.

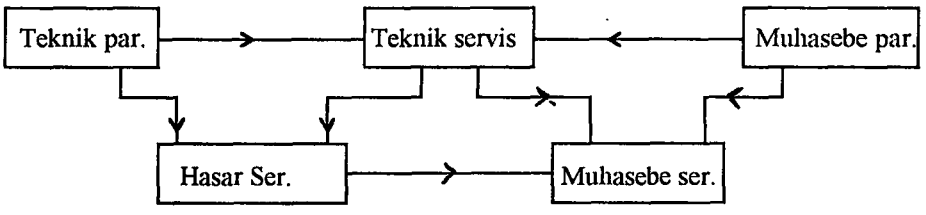
Bütün bilgilerin dökümü, istenen ay ve yılda reasurans bordrosu ve trete bordrosu çekilerek gerçekleşir. Bu raporlar poliçe poliçe bilgi vermektedir ve sonunda prim, bedel ve komisyon toplamlarını göstermektedir. Daha sonra reasurans mahsubu çekilerek muhasebeleşir.

Bu servis branş branş rejistro (kayıt) defteri tutmak zorundadır. Aynı zamanda acentalarda bu defteri tutmakla yükümlüdür. Rejistro defteri poliçe poliçe alınan yekün aylık net prim istihsalini, gider vergisi miktarını ve tahsil olunan brüt primin miktarını gösterir.

Ayrıca teknik servis işlemleri için tecditname hazırlama gerçekleşir. Tecditname yangın, kasko, trafik gibi süreli nitelikteki poliçeler için sürenin bitmesinde hazırlanır. Acentanın tecditname hazırlama yetkisi vardır.

Daha sonra çekilen vergi icmalleri ile belediyelere YSV ve gider vergisi ödenir ve o ay için çalışılan dağılımdan hemen sonra ay kapatılır (prodüksiyon kapama). Borç ve alacaklar eşitlenerek hareket dökümüne başlanır, resmi defterler tutulur ve

artık o aya ait bir daha işlem yapılmaz. Servisler arası ilişki Şekil 5.3'de gösterilmektedir.



Şekil 5.3. Servisler arası ilişki.

Muhasebe servisinin diğer bir ana işlevi ise acenta hesap özeti çekmesidir. Bu acentaların ilgili ay içinde muhasebeleşen tüm hesap bakiyelerini gösterir. Hesap özetinde acenta ne kadar prim üretmiş, ne kadarını ödemiş, şirketten ne kadar komisyon alacak belirlenir. Komisyon, poliçenin üzerinde yazılı net primin o branşa ait komisyon oranı ile çarpılmasından bulunur. Bu hesap özeti ilgili acentalara gönderilir.

Hasar servisi tarafından müşteri veya acenta tarafından yapılan hasar ihbarları değerlendirilir. Hasarın gerçekleştiği yere eksper gönderilir ve tahmini hasar (muallak hasar) belirlenir. Hasar, zeyl bazında sisteme girilir. Bir zeyl'ın cinsi, üzerinde tanımlanan zeyl türü bilgisi ile ayırtedilir. Daha sonra çekilen raporlarla hergün ihbar edilen hasar dökümü teslim edilir. Hasar servis tarafından girilen bu bilgiler ışığında muhasebe servisi hasar mahsubu çekerek ödenecek gün belirlenir. Hasardan muafiyet kısmı ve konservasyon çıkarılır. Reasurans şirketlerinden yüzdeleri alınır. Müşteriye ödeme yapılır. Hasar kapatılır.

Eksperin, sigortalı ile sigortacı arasındaki hasar olaylarında tarafsızlığını koruyarak en iyi hizmeti vermesi esastır. Olay yangın ise; itfaiye raporu, fırtınadan dolayı ortaya çıkan bir hasar ise; o gün itibariyle Meteoroloji Müdürlüğünden alınacak fırtınanın şiddeti hakkında hava raporu gereklidir.

Yıl sonu işlemlerinde ise; bilanço çekilir. Yıllık kar-zarar belirlenir, yeni fiyatlar saptanır.

5.2. Görev Dağılımı ve Dağıtık Organizasyonel

Genel Müdürlük

Sigorta şirketleri, sigortalılardan devraldıkları rizikoların altında kalmamak ve taahhütlerini yerine getirebilmek için kendi mali yapılarını güçlendirmekten başka bir takım sigorta tekniklerinden yararlanırlar. Bu sigorta tekniklerinden biri de reasurans anlaşmalarıdır. Sigorta şirketi üzerinde biriken toplam rizikoların bir bölümünü kendi üzerinde tutarken kalanını ise başka şirketler sigortalar. Bu şirketler diğer sigorta şirketleri olabileceği gibi, genellikle yurtiçi ve yurtdışı reasurans şirketleridir. Böylece sigortalılarına teminat veren bir sigorta şirketi reasurans yoluyla üzerindeki rizikoları devrederek teminat alır. Şirketlere dağılan rizikoların oranları yapılan reasurans anlaşmaları ile belirlenir ve bu oranların girilmesi Genel Müdürlükçe gerçekleşir ve değişmez. Bu bilgilerin muhasebeleşmesi ise yine Genel Müdürlük tarafından yapılır.

Sigorta ve Reasurans şirketleri, sigorta ve reasurans şirketlerine ait değişiklikleri, belli dönemlerde Sigorta Murakabe Kuruluna bildirmek zorundadır.

Daha önce anlattığımız branş tanımlama, teminat ve tarife oluşturma, poliçe üzerinde yapılacak indirimlerin, muafiyetin ve fiyat tablosunun belirlenmesi, acenta bilgi girişi, muhasebe parametrelerinin girişi, acenta cari hesapları dökümü, Genel Müdürlük tarafından gerçekleşir. Hasar gerçekleştiğinde ilgili yere eksper gönderilmesi ve tahmini hasarın belirlenmesi yine bu birim tarafından yapılır. Bütün işlemlerin raporlanması Genel Müdürlüğe aittir.

Müşteri veya acenta tarafından yapılan hasar ihbarları değerlendirilir. Eksper gönderilir ve tahmini hasar belirlenir.

Genel Müdürlük, aylık cari hesabı oluşturan muhasebe işlemlerini içeren verileri, hesap özetini acentalara iletir.

Genel Müdürlükçe girilen bu bilgiler diğer işlemler için baz olmaktadır ve bilgilerin dışına çıkılması söz konusu değildir.

Bir poliçeyi oluşturan parametrik bilgiler, her acenta veritabanında olması gerekir. Fakat bu bilgilerin girilmesi ve değiştirilmesi anlattığımız gibi Genel Müdürlükçe yapılır.

Acentalar

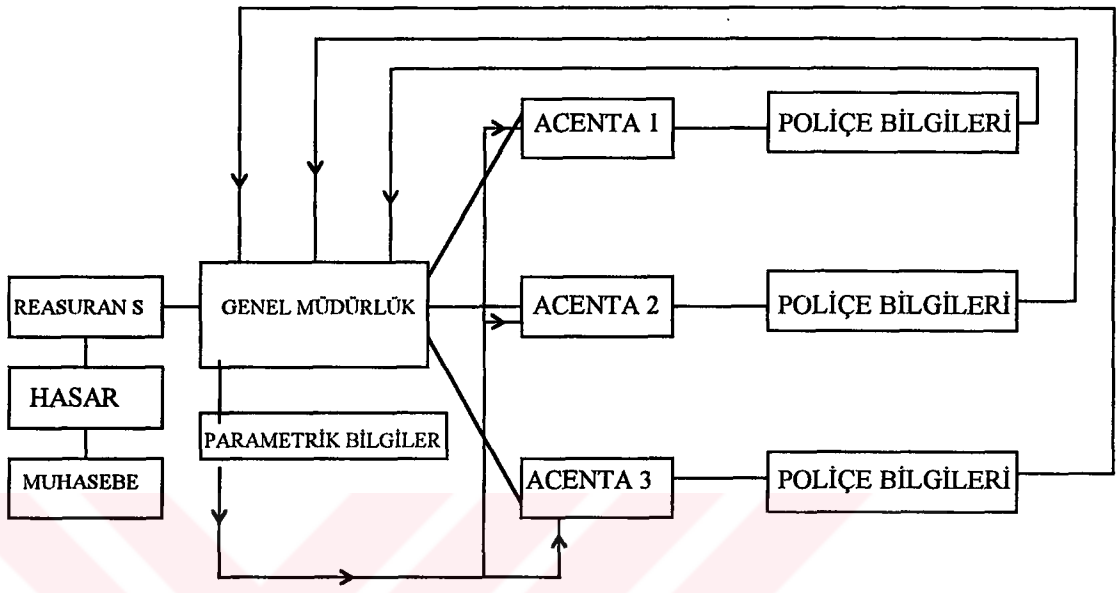
Her acenta kendi yerel yapısı içinde aynı işlemleri yapmaktadır ve bu birimlerin kendine ait poliçe bilgileri vardır. Acenta bağlı olduğu şubeye veya merkeze 10 gün içersinde poliçe bilgilerini bildirmek zorundadır.

Sigorta Murakabe Kanunu gereği acentalar, sigortalılardan tamamı veya belirlenmiş taksitlerde yaptıkları tahsilatı en geç poliçe tanzim tarihinden sonra gelen ayın belirlenmiş gününe kadar şirkete göndermekle yükümlüdür. Eğer A tipi acenta ise rejistro (poliçe kayıt) defteri B tipi acenta ise teklif defteri tutmak zorundadır. Ayırca A tipi acenta, aylık üretimleriyle ilgili gider verilerini maliyeye, yangın sigorta vergilerini tanımlanmış belediyelere, garanti fonlarını da fon hesabına yatırmakla yükümlüdür.

Acenta sözleşmeyle bağlı bulunduğu sigorta şirketinin ürünlerini iyi tanımalıdır ve tanıtmalıdır. Sigorta şirketinin yetki verdiği branşlarda poliçe düzenler ve yetkisi dışında kalan branşlarla ilgili teklifname hazırlayarak poliçenin yapılmasına aracılık yapar. Poliçe primlerini tahsil eder, hasar tesbiti yaparak şirketin kendisine vermiş olduğu yetki sınırları içersinde hasarları öder.

Acentalara ödecek komisyon oranı en az %30'dur. Komisyonlar tahakkuk eden net prim üzerinden hesaplanır. Acentaların tahsil ettiği net prim tahsilatı,

yapıldığı ayın belirlenmiş tarihine kadar sigorta şirketine ödenmek zorundadır. Bu bilgiler Genel Müdürlük tarafından daha sonra muhasebeleştirilir.



Şekil 5.4. Yapılan işlere göre dağılım.

Şekil 5.4'de de görüldüğü gibi yapılan işler Genel Müdürlük, acentalar bazında farklılık göstermektedir. Genel Müdürlük master bilgileri girmekte, acentalar bu bilgiler altında poliçe girmekte, tahsil etmekte, rejistro defteri tutmaktadır. Buna göre, Genel Müdürlük, kendisi ve acentalar tarafından girilen poliçeler doğrultusunda ay sonu, reasurans ve muhasebe işlemlerini yapmaktadır.

Bu işlemler farklı birimlerde gerçekleşmesine rağmen hepsi birbiriyle ilişki içersindedir. Örneğin acentalar poliçe keserken Genel Müdürlük tarafından belirlenen teminat, tarife ve fiyat tablosu gibi ana bilgileri baz almaktadır ve keza primlerin hesaplanması parametrik bilgilerle gerçekleşmektedir. Dolayısıyla bu bilgilerin her bir birimde bulunması gerekir.

Daha sonra Genel Müdürlük, ay sonunda bütün bu poliçe bilgileri ışığında reasurans dağılımı gerçekleştirmekte, muhasebesini yapmakta ve acentaların haber verdiği hasarları inceleyerek ilgili ödemeleri gerçekleştirmektedir.

Her bir birim yeri geldiği zaman özerk olarak çalışmakta, yeri geldiği zaman diğer birimlerle bağlantılı çalışmaktadır. Böyle bir organizasyon için en uygun yapı dağıtık organizasyon yapısıdır. Çünkü her birim özerk çalıştığı için, o sistem üzerindeki her hareket merkezi sisteme gitmek zorunda kalmamakta ve dolayısıyla olası ağ maliyeti azalmaktadır. Yerel bir işlem yalnız kendi sistemine bağlı olmakta ve ağ yavaşlığı veya ağ problemlerinden dolayı bir işlemin geç gerçekleşmesi veya gerçekleşememesi ihtimali azalmaktadır.

Mümkün olduğu kadar hiç bir sistem kendi işini yapmak için diğerine bağımlı olmamaktadır. Mesela Genel Müdürlük'teki sistemin durması acentanın hizmet vermesini etkilememektedir. Her sistemin kapasite ve performansı kendi yerel işlemleri doğrultusunda tesbit edilir. İşlem yükü dağıtıldığından dolayı merkezi bir sisteme aşırı bir yük sözkonusu değildir.

5.3. Proje İçersinde Kullanılan Master Dosyalar

Proje içersinde kullanılan master dosyalar Genel Müdürlükçe girilir. Birimler bu master bilgilere müdahalede bulunmazlar ve birimlere kopyalama yöntemiyle aktarılırlar. Bu master bilgiler diğer bütün işlemler için baz olmaktadır, servislerin işleyişini gerçekleştirmektedir ve sistem içinde değişmeden sürekli kullanılmaktadır. Bu master dosyalar branş, teminat, istatistik değer, istatistik kod, tarife, müşteri bilgileri, reasurans oranları, trete kod, fiyat tablosu, tarife ve teminat, tarife ve istatistik kod ilişkisidir.

Branş

Yapılan işlemler sigorta türlerine göre farklılık göstermektedir. Yani bir yangın branşının poliçeleri ile nakliyat branşının poliçeleri farklı işlemlere tabi tutulur. Her bir branş için bir kod tanımlanır. Bu bilgi yapılan tüm işlemlere taşınır.

Teminat

Olabilecek risklerin kodlandığı dosyadır. Ana teminat ve alt teminat olmak üzere ikiye ayrlır. Ana teminat zorunludur, alt teminat seçimlidir. Mesela ana teminat hırsızlık ise radyo-teyp çalışması bir alt teminattır.

İstatistik kod, İstatistik Değer ve Fiyat Tablosu

Bu bilgiler fiyat tablosunu oluşturur, sigorta primine etken olan parametrelerdir. İstatistik kod her bir teminat için olabilecek riskleri içerir. İstatistik değer ise bu riskin detaylandırılmasıdır. Bunu bir örnek ile açıklayalım.

Yangın tarifesindeki deprem teminatını etkileyecek fiyatlar şu bazda yapılabilir; binanın bulunduğu yerin deprem derecesi, binanın yapı tarzı, binanın kat adedi. Bunlar birer istatistik koddur. Binanın yapı tarzını detaylandırmak istersek; tam kagir olması, yarı kagir olması ya da ahşap olmasıdır. Bu bilgiler istatistik değer dosyasında tutulur. Bunlar üzerinde muafiyet, indirim, ekprim gibi tanımlar yapılır. Mesela, ekprim, 4 sene üstüste kaza yapmış birine uygulanır. Eğer 4 sene hiç kaza yapmamış ise indirim uygulanır.

Bu istatistik kod ve istatistik değer tanımlamaları bir tablo oluşturur ve her tablonun numarası vardır.

Tarife Tanımlama

Birbiri ile ilişkili teminatlar yani riskler bir tarife oluşturur. Tarife üzerindeki her bir teminat için poliçe üzerinde etken olacak parametreler tanımlanır. Bunlar reasurans dağılıma dahil olup olmayacağı, tablo kodu, tahsilat kodu, branşı, istatistik kodu ve istatistik kodun zorunlu veya seçimli kararı, vergi bilgisi ve muafiyet kodu gibi temel bilgilerdir. Burada seçilen istatistik kod zorunlu ise bu bilgi poliçe üzerinde değiştirilemez, seçimli ise değiştirilebilir. Ayrıca bu istatistik kodunun amacı belirtilir. Mesela bu istatistik kod ile ekprim, indirim veya muafiyet uygulanabilir. Poliçe üzerinde çıkacak bilgi ve açıklamalarda tarife bazında sisteme girilir.

Trete Tanımlama

Trete kod tanımlama, reasurans dağılımda hisselerin paylaştırılacağı reasürörlerin kodunu içerir. Bu tanımlama içinde trete tipi de vardır. Trete tipi, tretelere bir sıra vermektir amacıyla kullanılır. Mesela konservasyonun sıra no'su 1'dır ve programlar içersinde bu numaralara göre kodlama yapılır.

Ayrıca ihtiyari trete kodu için kaynak kod bilgisi bu dosya içinde tutulur. Çünkü trete'nin ihtiyari şirket kodu sisteme müşteri bazında girilir ve bu şirket kodlarına ait bir kaynak kodu tanımlanır.

Reasurans Master Bilgileri

Burada, dağılıma girecek olan her bir trete kodu için gerekli tanımlamalar yapılır. Bu tanımlamalar tarifeyi, prim mi, hasar mı olduğunu, branşını içerir. Daha sonra, bu treteler için oran, bedel, plenden biri tanımlanır. Bu tanımlar dağılımın hangi bazda olacağını belirler. Saklama payı bazen plen ile belirlenir. Bunun anlamı mesela 2 plendan daha fazla ödeme yapılmayacağıdır. Plen olarak belirtilmeyen bir dağılım şekli bir oran veya bir bedel dahilinde gerçekleşecektir. Bu tanımlamaların asgari ve azami değerleri bu bazda belirtilir. Ayrıca her bir tretenin (ihtiyari hariç) komisyon oranı ve matrahı, trete tanımlanırken girilir.

Vergi Tanımlama

Daha önce anlattığımız gibi 3 tip vergi vardır. Bunlar gider vergisi, yangın sigorta vergisi ve garanti fonudur. Yangın sigorta vergisi sadece yangın poliçeleri için uygulanır. Bu vergiler, saptanan primler üzerinden belirli oranlar dahilinde uygulanır.

Ekno ve Kaynak Tanımlama

Ekno, acentanın veya müşterinin cari hesabını takip etmek için kullanılır. Kaynak, müşterinin veya acentanın türünü belirler, acentanın yetkileri burada tanımlanır. Örneğin burada tanımlanan istihsal organı "E" ise bu, acentanın poliçe üretme yetkisi olduğunu gösterir. Ayrıca ödeme takibi ve prim takibi yetkisi de burada

verilir. Eğer bir acenta komisyon alacaksa, bu tanımlama kaynak kod üzerinden gerçekleşir. Bir acenta aynı zamanda komisyonda verebilir. Bu acentalara bağlı tali acentalar için geçerlidir. Bir acentanın işlemlerini yapan şahıs veya kuruluşlar da olabilir. Bunlar tali acentalardır. Bunu komisyon verir sahasından anlayabiliriz.

Ekno ve kaynak ilişkisi ile müşteri veya acentanın cari hesabının vergi mi, prim borcu mu, komisyon mu olduğunu saptayabiliriz.

Polise Cinsi Tanımlama

Poliçe üzerinde tanımlanan bu master bilgi ile o polıçenin cinsi ayırt edilir. Mesela poliçe daha önce anlattığımız gibi fulotal poliçe olabilir. Ayrıca poliçenin, reasurans dağılıma ne ölçüde katılacağı burada belirtilir. Sadece primi dağıtılabilir ya da prim ve bedelin ikisi birden dağıtılabilir.

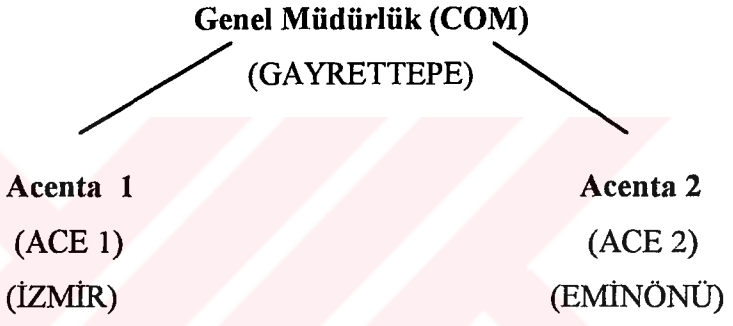
Zeyl Türü Tanımlama

Sistemden girilen hasarlar ve intaller police bazında gerçekleşir. Bu [illegible]

5.4. Veritabanı ve Uygulamalar

Veritabanı uygulamaları daha önce anlattığımız Oracle veritabanının SQL*net, veritabanı bağlantıları ve istemci/sunucu mimarisi özelliğinden yararlanarak gerçekleşmektedir. SQL*net, dağıtık veritabanı için ağ üzerinden TCP/IP protokolunu kullanmaktadır ve dağıtık hareketi gerçekleştirmek için ağ üzerinden veritabanına bağlanmaktadır.

Buna göre her bir veritabanının tek bir global nesne ismi belirlenmelidir.



ACE1 veritabanındaki tarife_table'a ulaşmak için; tarife_table@ace1.com şeklinde mantıksal bir ifade kullanılmaktadır.

Oracle, yerel bir veritabanından uzak veritabanına bağlanmak için veritabanı bağlantıları kullanır. Veritabanı bağlantısının adı, global veritabanı adı ile aynıdır. Örneğin, Genel Müdürlükteki bir kullanıcı acentaya bağlanıp brans_table'daki bir güncellemeyi şu program parçası ile yapar.

```
CREATE PUBLIC DATABASE LINK ace1.com
UPDATE brans_table@ace1.com
   SET brans_adı="YANGIN BRANŞI"
   WHERE brans_adı="YANGIN"
```

Yerel yapılar arasındaki bilgi akışı şu aşamalardan geçmektedir;

- Sabit bilgilerin, Genel Müdürlükten acentalara veri kopyalaması yolu kullanılarak aktarılması,

- Acentalarda kesilen poliçe bilgilerinin Genel Müdürlük tarafina aktarılıp reasurans çalışmalarının ve muhasebe işlemlerinin yapılması.

Daha önce anlattığımız gibi acentalarda, bir poliçeyi oluşturan tüm parametrik bilgiler tutulmaktadır. Yerel özerklik ve merkezi bir sisteme bağımlılık yoktur dağıtıklık kurallarına göre verilerin tek bir merkezi sistemde yer alması sakıncalıdır. Bu parametrik bilgilerin yerel veritabanlarında tutulması performansı artıracaktır. Master bilgilerin sabitliğini koruduğunu da düşünerek dağıtıklık kuralı olan veri kopyalama bağımsızlığını kullanarak merkezde (Genel Müdürlük) yeri girilen veya güncellenen veriler diğer acentalara toplu olarak aktarılmaktadır.

Önce bu veritabanlarına bağlanmak için veritabanı bağlantıları yaratılır;

```
CREATE PUBLIC DATABASE LINK ace1.com
CREATE PUBLIC DATABASE LINK ace2.com
```

Yeni girilen bir teminat bilgisi acentalara şu program parçası ile aktarılır;

```
DEFINE TRIGGER
NAME=KEY-NXTFLD
TRIGGER-TYPE=V3
TEXT=<<<
if : tem-kod is null then message ("aktarılacak teminat kodunu giriniz");
                    bell;
                    raise_form_trigger_failure;
end if;
if  :  tem-kod is not null then
     AKTAR_1;
     AKTAR_2;
```

```
            commit;
            if form_success then message ("aktarma tamamlandı");
                                else message ("bağlantı kurulamadı");
            end if;
        end if;
    >>>
    ENDDEFINE TRIGGER
    DEFINE PROCEDURE
    NAME = AKTAR_1;
    DEFINITION=<<<
    PROCEDURE aktar_1 is
    begin
        begin
        select * from teminat_table@ace1.com
        where teminat_kod=:tem_kod
        when no_data_found then
            insert into teminat_table@ace1.com
            (select * from teminat_table
            where teminat_kod=:tem_kod);
        end;
    end;
>>>
ENDDEFINE PROCEDURE
DEFINE PROCEDURE
NAME = AKTAR_2
DEFINATION=<<<
PROCEDURE aktar_2 is
begin
    begin
      select  * from teminat_table@ace2.com
      where teminat_kod=:tem_kod
      when no_data_found then
```

```
        insert into teminat_table@ace2.com
        (select * from teminat_table
        where teminat_kod=:tem_kod);
    end;
end;
>>>
ENDDEFINE PROCEDURE
```

Bir değişikliğin aktarılması sözkonusu ise bu aktarım şu şekilde yapılır;

```
DEFINE PROCEDURE
NAME = değiş_1
DEFINATION=<<<
PROCEDURE değiş_1 is
uzun teminat_table.uzun_ad%type;
kısa teminat_table.kısa_ad%type;
baş teminat_table.baş_tar%type;
bit teminat_table.bit_tar%type;
begin
    begin
    select uzun_ad, kısa_ad, baş_tar, bit_tar into
            uzun, kısa, baş, bit
    from teminat_table
    where teminat_kod=:tem_kod
    update teminat_table@ace1.com
    set uzun_ad=uzun,
        kısa_ad=kısa,
        baş_tar=baş,
        bit_tar=bit
    where teminat_kod: tem_kod
   end;
end;
>>>
ENDDEFINE PROCEDURE
```

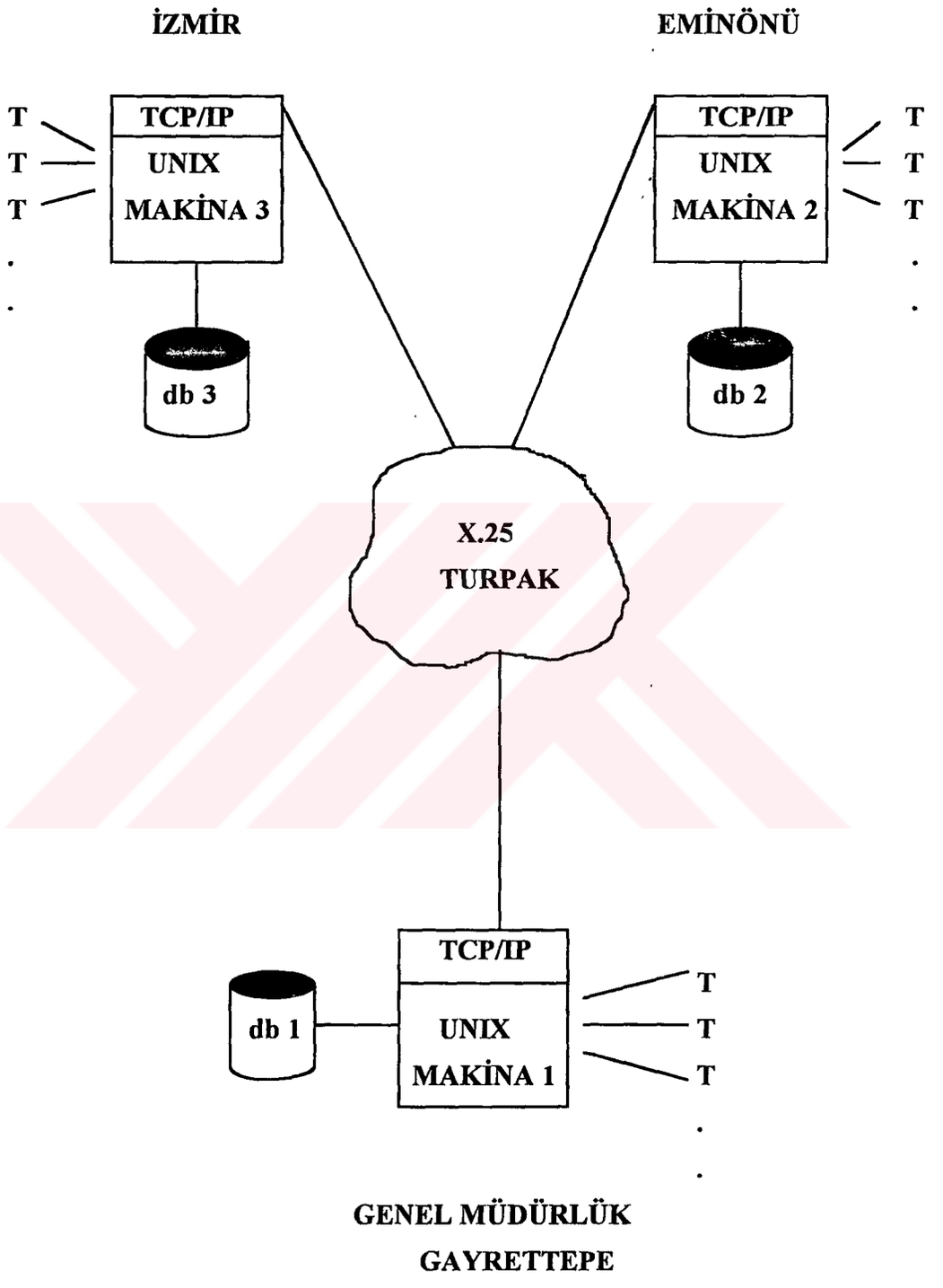
Diğer taraftan acentaların girdiği poliçeler dahilinde gerçekleşecek reasurans dağılım için acenta poliçelerinin Genel Müdürlüğe aktarılması gerekmektedir. Çünkü verilerin direkt acentalardan sorgulanarak işlem görmesi, yapılacak dağılımı oldukça yavaşlatacaktır.

Acenta poliçe bilgilerinin Genel Müdürlüğe aktarılması ise ara dosya kullanılarak yapılır ve şu program parçası ile gerçekleştirilir;

```
DEFINE TRIGGER
NAME = KEY_NXTFLD
TRIGGER-TYPE=V3
TEXT=<<<
if : başlangıç_tarih is not null and
if : bitiş_tarih is not null then
   insert into poliçe_geçici_table
   (select * from poliçe_table@ace1.com
   where tarih in (:başlangıç_tarih, : bitiş_tarih))
   insert into poliçe_teminat_geçici_table
   (select * from poliçe_teminat_table@ace1.com
   where tarih in (:başlangıç_tarih, : bitiş_tarih))
   insert into poliçe_ist_geçici_table
   (select * from poliçe_ist_table@ace1.com
   where tarih in (:başlangıç_tarih, : bitiş_tarih))
   insert into poliçe_tahsilat_geçici_table
   (select * from poliçe_tahsilat_table
   where tarih in (:başlangıç_tarih, :bitiş_tarih))
   insert into poliçe_sigortalı_geçici_table
   (select * from poliçe_sigortalı_table
   where tarih in (:başlangıç_tarih, : bitiş_tarih))
   commit;
   if form_success then message ("transfer tamamlandı");
                        else message ("transfer tamamlanamadı");
                             bell;
                             raise_form_trigger_failure;
```

```
  end if;
 AKTAR;
end if;
>>>
ENDDEFINE TRIGGER

DEFINE PROCEDURE
NAME = AKTAR
DEFINATION = <<<
PROCEDURE aktar is
begin
    insert into poliçe_table
    (select * from poliçe_geçici_table)
    insert into poliçe_teminat_table
    (select * from poliçe_teminat_geçici_table)
    insert into poliçe_ist_table
    (select * from poliçe_ist_geçici_table)
    insert into poliçe_tahsilat_table
    (select * from poliçe_tahsilat_geçici_table)
    insert into poliçe_sigortalı_table
    (select * from poliçe_sigortalı_geçici_table)
    delete from poliçe_geçici_table
    delete from poliçe_teminat_geçici_table
    delete from poliçe_tahsilat_geçici_table
    delete from poliçe_sigortalı_geçici_table
    commit;
    if form_success then message ("işlem tamamlandı");
                    else  message ("işlem tamamlanamadı");
    end if;
end;
>>>
ENDDEFINE PROCEDURE
```



Şekil 5.5. Ağ üzerinde veritabanı dağılımı.

SONUÇLAR VE ÖNERİLER

Bugün veritabanı yönetim sistemleri diğer bilgi sistem teknolojileri gibi bir değişim içindedir. Şirketler, istemci/sunucu bilgi işleme ve kullanıcıların daha fazla veriye erişim yönündeki talepleriyle gelen bu değişim sürecine adapte olmaya çalışmaktadırlar.

Bütün verilerin tek bir merkezi bilgisayarda tutulması, değişen ekonomik koşullara ayak uydurmak zorunda olan kuruluşların etkin işlemesine olanak tanımadığı için, bilgi işlem dünyasındaki gelişmeler zorunlu olarak dağıtık veritabanı kavramını ortaya çıkardı. İletişim teknolojisindeki gelişmelerle birlikte, farklı mimarilere sahip sistemlerin haberleşmesi, dağıtık sistemlerin altyapısını hazırladı.

Dağıtık veritabanı yönetim sistemleri, içerdiği olanaklar sayesinde birçok uygulama için gerçekten büyük faydalar getiriyor. Bu faydalardan yararlanmak için şu faktörlerin iyice hesaplanması gerekir; hangi verilerin hangi yerel veritabanında yer alacağı, bunlar arasında kurulacak mantıksal ilişki, hatalara karşı gösterilecek gözardı payı.

Bütün bunların yanında bilgisayar ağ hızının çok iyi olması gerekmektedir. Çünkü dağıtık yapı coğrafi olarak yayılmış bir ağ üzerinde gerçekleştiğinden, bu ağın hızlı olması, sistemin performansını o kadar artıracaktır. Mesela on-line hareketlerin performansı ağın hızıyla alakalıdır. Bu hareketlerin zamanında çalışmaması performansı etkilemekten öte veri tutarsızlığına neden olmaktadır.

Bu yüzden, farklı veritabanlarındaki hareketler, iletişim hızı ve hareketin önemi gözönündebulundurularak düzenlenmelidir. Eğer iletişim hızı yetersiz kalıyorsa sabit

bilgilerin bütün veritabanlarında olması faydalı olacaktır. Ayrıca verinin, en çok kullanılan birimlerde saklanılması ağ trafiğini azaltacaktır.

Sonuçta, bütün bu uğraşların sebebi, kullanıcıların taleplerine daha hızlı karşılık vermek ve veri erişimini kolay hale getirmektir. Dağıtık veritabanı yönetim sisteminin özelliklerini sağlamak görevi ise veritabanı yöneticilerine ve yazılımcılara düşüyor.

KAYNAKLAR

[1] CERI, S., PELAGATTI, G., Distributed Databases Principles and Systems, Politecnico di Milano, McGraw-Hill Book Company, (1984).

[2] UYLESS, D. B., Data Communications and Distributed Networks, Prentice Hall International Editions.

[3] Internet, PC World, Ocak (1995).

[4] ÇAĞLAYAN, M.U., ERSOY, C., Department of Computer Engineering, TRUUG, TCP/IP Network Administration, Açık Sistem'95 Sempozyumu, 22-24 Şubat (1995).

[5] DATE, C.J., An Introduction to Database Systems, Volume I-Fifth Edition The Systems Programming Series, (1990).

[6] ÖZSU, M., VALDURIEZ, P., Principles of Distributed Database Systems, Prentice Hall, (1991).

[7] Dağıtık RDBMS'ın Temelleri, Computer World Monitör, (1994).

[8] Oracle 7 Server Concepts Manual, (1992).

[9] Oracle RDBMS Database Administrator's Guide, Version 6.0, (1990).

[10] Oracle 7 Server Administrator's Guide, (1992).

[11] Oracle 7 Server SQL Language Reference Manual, (1992).

EK PROGRAM CIKTILARI

SQL> desc brans_table

Name	Null?	Type
SIRKET_KOD		NUMBER(3)
BRANS_KOD		VARCHAR2(3)
BRANS_ING_ADI		VARCHAR2(15)
BRANS_ADI		VARCHAR2(15)
TARIH		DATE
OP_ID		VARCHAR2(12)

SQL> desc ist_kod_table

Name	Null?	Type
SIRKET_KOD		NUMBER(3)
IST_KOD		VARCHAR2(3)
IST_ADI_KISA		VARCHAR2(20)
IST_ADI_UZUN		VARCHAR2(30)
TARIH		DATE
OP_ID		VARCHAR2(12)

SQL> desc trete_kod_table

Name	Null?	Type
SIRKET_KOD		NUMBER(3)
TRETE_KOD		VARCHAR2(2)
TURKCE_ADI		VARCHAR2(15)
ING_ADI		VARCHAR2(15)
TRETE_TIP		VARCHAR2(15)
KAYNAK_KOD		NUMBER(3)
TRETE_TIP		NUMBER(2)
TARIH		DATE
OP_ID		VARCHAR2(12)

SQL> desc ist_deger_table

Name	Null?	Type
SIRKET_KOD		NUMBER(3)
IST_KOD		VARCHAR2(3)
DEGER_KOD		VARCHAR2(3)
DEGER_AD		VARCHAR2(20)
MUAF_ORAN		NUMBER(7,4)
MUAF_MATRAH		NUMBER(5)
MUAF_ASGARI		NUMBER(14,2)
MUAF_AZAMI		NUMBER(14,2)
IND_ORAN		NUMBER(7,4)
IND_MATRAH		NUMBER(5)
SURP_ORAN		NUMBER(7,4)
SURP_MATRAH		NUMBER(5)
TARIH		DATE

```
OP_ID                          VARCHAR2(12)

SQL> desc tarife_table
 Name                      Null?  Type
 ------------------------- -------- ----
 SIRKET_KOD                       NUMBER(3)
 TARIFE_KOD                       VARCHAR2(3)
 VK                               NUMBER(2)
 ADI                              VARCHAR2(25)
 ING_ADI                          VARCHAR2(25)
 KISA_ADI                         VARCHAR2(15)
 ING_KISA_ADI                     VARCHAR2(15)
 BASTAR                           DATE
 TARIH                            DATE
 OP_ID                            VARCHAR2(12)
 BITTAR                           DATE

SQL> desc tarife_teminat_table
 Name                      Null?  Type
 ------------------------- -------- ----
 SIRKET_KOD                       NUMBER(3)
 TARIFE_KOD                       VARCHAR2(3)
 VK                               NUMBER(2)
 TEMINAT_KOD                      VARCHAR2(3)
 TEMINAT_VK                       NUMBER(2)
 Z_S                              VARCHAR2(1)
 BRN                              VARCHAR2(3)
 REA                              VARCHAR2(1)
 POL                              VARCHAR2(1)
 TAB_KOD                          VARCHAR2(5)
 TAHSIL                           VARCHAR2(5)
 TARIH                            DATE
 OP_ID                            VARCHAR2(12)
 S_NO                             NUMBER(2)
 ORAN                             NUMBER(7,4)

SQL> desc tarife_ist_table
 Name                      Null?  Type
 ------------------------- -------- ----
 SIRKET_KOD                       NUMBER(3)
 TARIFE_KOD                       VARCHAR2(3)
 VK                               NUMBER(2)
 TEMINAT_KOD                      VARCHAR2(3)
 TEMINAT_VK                       NUMBER(2)
 IST_KOD                          VARCHAR2(3)
 BRN                              VARCHAR2(3)
 AMAC                             VARCHAR2(1)
 Z_S                              VARCHAR2(1)
 MUAFIYET                         VARCHAR2(5)
 OT_DEGER                         VARCHAR2(3)
 IST_ACIK                         VARCHAR2(1)
 UST_TEMINAT_KOD                  VARCHAR2(3)
 UST_TEMINAT_VK                   NUMBER(2)
 TARIH                            DATE
 OP_ID                            VARCHAR2(12)
 S_NO                             NUMBER(2)
```

SQL> desc teminat_table

Name	Null?	Type
SIRKET_KOD		NUMBER(3)
TEM_KOD		VARCHAR2(3)
VK		NUMBER(2)
UZUN_AD		VARCHAR2(30)
KISA_AD		VARCHAR2(20)
BAS_TAR		DATE
BIT_TAR		DATE
ALT_TEM		VARCHAR2(1)
TARIH		DATE
OP_ID		VARCHAR2(12)

SQL> desc reas_dagilim_table

Name	Null?	Type
SIRKET_KOD		NUMBER(3)
SUBE_KOD		NUMBER(4)
KAYNAK		NUMBER(3)
ACENTA_NO		VARCHAR2(7)
CARI_POL_NO		NUMBER(7)
ZEYL_TURU		VARCHAR2(3)
ZEYL_SIRA_NO		NUMBER(3)
T_I		VARCHAR2(1)
BRANS_KODU		VARCHAR2(3)
TANZIM_YILI		VARCHAR2(4)
BILGI_TURU		VARCHAR2(1)
YI_YD		VARCHAR2(2)
POL_BAS_TARIH		DATE
POL_BIT_TARIH		DATE
TRETE_KODU		VARCHAR2(2)
TRETE_SIRKET_KODU		VARCHAR2(7)
TRETE_SIRKET_MEBLAG		NUMBER(14)
TRETE_SIRKET_PRIM		NUMBER(14,2)
ACIKLAMA		VARCHAR2(40)
AY		VARCHAR2(2)
YIL		VARCHAR2(4)
TARIH		DATE
OP_ID		VARCHAR2(12)
TARIFE		VARCHAR2(3)
ORAN		NUMBER(10,7)
TANZIM_TARIHI		DATE
YUKLEME_TARIHI		DATE
TRETE_SIRKET_KOM		NUMBER(14,2)

SQL> desc reas_otomatik_table

Name	Null?	Type
SIRKET_KOD		NUMBER(3)
AY		VARCHAR2(2)
YIL		VARCHAR2(4)
BRANS		VARCHAR2(3)
KOD		VARCHAR2(1)
TARIH		DATE
OP_ID		VARCHAR2(12)

SQL> desc reasurans_header_table

Name	Null?	Type
SIRKET_KOD		NUMBER(3)
TANZIM_YILI		VARCHAR2(4)
PRIM_HASAR		VARCHAR2(1)
BRANS		VARCHAR2(3)
YIL_KODU		VARCHAR2(1)
YILI		VARCHAR2(4)
TARIFE		VARCHAR2(3)
TRETE_KODU		VARCHAR2(2)
SIRA_NO		NUMBER(1)
IST_KOD		VARCHAR2(3)
IST_SIRA		NUMBER(2)
TARIH		DATE
OP_ID		VARCHAR2(12)

SQL> desc reasurans_bedel_table

Name	Null?	Type
SIRKET_KOD		NUMBER(3)
TANZIM_YILI		VARCHAR2(4)
PRIM_HASAR		VARCHAR2(1)
BRANS		VARCHAR2(3)
YIL_KODU		VARCHAR2(1)
YILI		VARCHAR2(4)
TARIFE		VARCHAR2(3)
TRETE_KODU		VARCHAR2(2)
SIRA_NO		NUMBER(1)
REAS_KEY		VARCHAR2(90)
ORAN		NUMBER(7,4)
MATRAH		NUMBER(5)
R_E		VARCHAR2(1)
B_K		VARCHAR2(1)
PLEN		NUMBER(3)
BEDEL		NUMBER(16,2)
BEDEL_DOVIZ		VARCHAR2(3)
MAX_BEDEL		NUMBER(16,2)
MAX_DOVIZ		VARCHAR2(3)
ASG_BEDEL		NUMBER(16,3)
ASG_DOVIZ		VARCHAR2(3)
TARIH		DATE
OP_ID		VARCHAR2(12)

SQL> desc acenta_komisyon_table

Name	Null?	Type
SIRKET_KOD		NUMBER(3)
MUST_KOD		VARCHAR2(7)
KAYNAK_KOD		NUMBER(3)
BAS_TARIH		DATE
BRANS_KOD		VARCHAR2(3)
KOM_ORANI		NUMBER(7,4)
MATRAH		NUMBER(5)
TARIH		DATE
OP_ID		VARCHAR2(12)

SQL> desc must_tanim_table

Name	Null?	Type
SIRKET_KOD		NUMBER(3)
MUST_KOD		VARCHAR2(7)
KAYNAK_KODU		NUMBER(3)
SF		VARCHAR2(1)
MUST_ADI		VARCHAR2(25)
MUST_SOYADI		VARCHAR2(25)
DOGUM_IL		VARCHAR2(3)
DOGUM_ILCE		VARCHAR2(3)
DOGUM_TARIH		DATE
FAAL_KOD		NUMBER(2)
UNVAN1		VARCHAR2(25)
UNVAN2		VARCHAR2(25)
MAHALLE		VARCHAR2(15)
CADDE		VARCHAR2(15)
SOKAK		VARCHAR2(15)
HAN_APT		VARCHAR2(15)
NO		NUMBER(3)
DAIRE		NUMBER(3)
POSTA_KODU		VARCHAR2(6)
BEL_IL		VARCHAR2(3)
BEL_ILCE		VARCHAR2(3)
VERGI_KODU		NUMBER(2)
YETKI_KODU		VARCHAR2(1)
TEL		VARCHAR2(15)
FAKS		VARCHAR2(15)
TELEKS		VARCHAR2(15)
ULKE_KOD		NUMBER(2)
VERGI_DAIRE		VARCHAR2(15)
VERGI_NO		VARCHAR2(15)
TGS_TARIH		DATE
YEREL_TARIH		DATE
ACILIS		DATE
KAPANIS		DATE
TARIH		DATE
OP_ID		VARCHAR2(12)

SQL> desc police_cins_table

Name	Null?	Type
SIRKET_KOD		NUMBER(3)
POL_CINSI		VARCHAR2(1)
CINS_ADI		VARCHAR2(25)
BAS_TARIH		DATE
REAS_BEDEL		VARCHAR2(1)
REAS_PRIM		VARCHAR2(1)
TARIH		DATE
OP_ID		VARCHAR2(12)

SQL> desc zeyl_header_table

Name	Null?	Type
SIRKET_KOD		NUMBER(3)
ZEYL_KODU		VARCHAR2(3)
SIRA_NO		NUMBER(3)
TURKCE_ADI		VARCHAR2(25)
ING_ADI		VARCHAR2(25)
T_I		VARCHAR2(1)
ZEYL_GIRIS		VARCHAR2(1)
REAS_GIRIS		VARCHAR2(1)
POL_MEVCUT_DURUM		VARCHAR2(1)
POL_DEVAM_DURUM		VARCHAR2(1)
VADE_DEGISTIR		VARCHAR2(1)
TEMINAT_ILAVE		VARCHAR2(1)
TEMINAT_EKSILT		VARCHAR2(1)
TEMINAT_MEV_DURUM		VARCHAR2(1)
TEMINAT_DEV_DURUM		VARCHAR2(1)
MEBLAG		VARCHAR2(1)
ABONMAN		VARCHAR2(1)
TARIH		DATE
OP_ID		VARCHAR2(12)

SQL> desc police_table

Name	Null?	Type
SIRKET_KOD		NUMBER(3)
SUBE_KOD		NUMBER(4)
KAYNAK_KODU		NUMBER(3)
ACENTA_NO		VARCHAR2(7)
TALI_ACENTA_NO		VARCHAR2(7)
POL_CINSI		VARCHAR2(1)
CARI_POL_NO		NUMBER(7)
ESKI_POL_NO		NUMBER(7)
TARIFE_KOD		VARCHAR2(3)
VK		NUMBER(2)
TEKLIF_TARIH		DATE
YUKLEME_TARIH		DATE
BASLAMA_TARIH		DATE
BITIS_TARIH		DATE
TANZIM_TARIH		DATE
TANZIM_YERI		VARCHAR2(3)
TEKLIF_NO		NUMBER(7)
TAHSILAT_KODU		VARCHAR2(5)
GENEL_SURP_ORAN		NUMBER(7,4)
GENEL_SURP_MATRAH		NUMBER(5)
GENEL_IND_ORAN		NUMBER(7,4)
GENEL_IND_MATRAH		NUMBER(5)
TEKLIF_ONAY		NUMBER(2)
ZEYL_TURU		VARCHAR2(3)
ZEYL_SIRA_NO		NUMBER(3)
T_I		VARCHAR2(1)
SON_DURUM		VARCHAR2(1)
TARIH		DATE
OP_ID		VARCHAR2(12)
ADET		NUMBER(4)

```
IPTAL_ZEYL_SIRA                NUMBER(3)

SQL> desc police_teminat_table
Name                           Null?  Type
------------------------------ -------- ----
SIRKET_KOD                     NUMBER(3)
SUBE_KOD                       NUMBER(4)
KAYNAK_KODU                    NUMBER(3)
ACENTA_NO                      VARCHAR2(7)
CARI_POL_NO                    NUMBER(7)
 TARIFE                        VARCHAR2(3)
TEMINAT_KODU                   VARCHAR2(3)
BEDEL                          NUMBER(16,2)
BEDEL_DOVIZ                    VARCHAR2(3)
BEDEL_KUR                      NUMBER(9,4)
TL_BEDEL                       NUMBER(14)
FIYAT                          NUMBER(7,4)
FIYAT_MATRAH                   NUMBER(5)
MUAFIYET                       NUMBER(14,2)
TL_MUAF                        NUMBER(14)
PRIM                           NUMBER(14,2)
TL_PRIM                        NUMBER(14)
SURPRIM                        NUMBER(14,2)
TL_SURPRIM                     NUMBER(14)
INDIRIM                        NUMBER(14,2)
TL_INDIRIM                     NUMBER(14)
NET_PRIM                       NUMBER(14,2)
TL_NET_PRIM                    NUMBER(14)
KOM_TUTARI                     NUMBER(14)
KOM_ORANI                      NUMBER(7,4)
TANZIM_TARIHI                  DATE
BRANS                          VARCHAR2(3)
REAS_BEDEL                     NUMBER(14)
ZEYL_TURU                      VARCHAR2(3)
POLICE_CINSI                   VARCHAR2(1)
TAN_AY                         VARCHAR2(2)
TAN_YIL                        VARCHAR2(4)
ZEYL_SIRA_NO                   NUMBER(3)
T_I                            VARCHAR2(1)
TARIH                          DATE
OP_ID                          VARCHAR2(12)
KOM_MATRAH                     NUMBER(5)
UST_TEMINAT_KOD                VARCHAR2(3)
ADET                           NUMBER(4)
TEMINAT_VK                     NUMBER(2)
TAHSILAT_KODU                  VARCHAR2(5)
Z_S                            VARCHAR2(1)
S_NO                           NUMBER(3)

SQL> desc police_sigortali_table
Name                           Null?  Type
------------------------------ -------- ----
SIRKET_KOD                     NUMBER(3)
CARI_POL_NO                    NUMBER(7)
ZEYL_SIRA_NO                   NUMBER(3)
AD1                            VARCHAR2(25)
```

AD2	VARCHAR2(25)
MAH_KOY	VARCHAR2(25)
CADDE	VARCHAR2(20)
NO	NUMBER(4)
ILCE	VARCHAR2(3)
IL	VARCHAR2(3)
TARIH	DATE
OP_ID	VARCHAR2(12)

SQL> desc reasbor_rapor_table

Name	Null?	Type
TRETE_ADI		VARCHAR2(25)
CARI_POL_NO		NUMBER(7)
ZEYL_SIRA_NO		NUMBER(3)
BASLIK		VARCHAR2(12)
T_I		VARCHAR2(1)
TARIFE		VARCHAR2(3)
POLICE		VARCHAR2(12)
BAS_TAR		DATE
BIT_TAR		DATE
TAN_TAR		DATE
SIGORTA_BEDEL		NUMBER(14)
SIGORTA_PRIM		NUMBER(14,2)
TRETE_BEDEL		NUMBER(16)
TRETE_PRIM		NUMBER(16,2)
TRETE_YUZDE		NUMBER(7,4)
NET_BEDEL		NUMBER(14)
NET_PRIM		NUMBER(14,2)
TRETE_SIRKET_KODU		VARCHAR2(7)
TRETE_KODU		VARCHAR2(2)
OP_ID		VARCHAR2(12)
BRANS		VARCHAR2(3)
ADRES		VARCHAR2(90)
SIRA		NUMBER(5)
KAYNAK		NUMBER(3)
ACENTA_NO		VARCHAR2(12)

SQL> desc bordro_rapor_table

Name	Null?	Type
CARI_POL_NO		NUMBER(7)
ZEYL_SIRA_NO		NUMBER(3)
BASLIK		VARCHAR2(28)
T_I		VARCHAR2(1)
TARIFE		VARCHAR2(3)
POLICE		VARCHAR2(14)
ZEYL		VARCHAR2(6)
BAS_TAR		DATE
BIT_TAR		DATE
TAN_TAR		DATE
SIGORTA_BEDEL		NUMBER(14)
SIGORTA_PRIM		NUMBER(14,2)
SIGORTA_FIYAT		NUMBER(7,4)
KON_BEDEL		NUMBER(14)
KON_PRIM		NUMBER(14,2)

KON_FIYAT	NUMBER(7,4)
TRETE_BEDEL	NUMBER(14)
TRETE_PRIM	NUMBER(14,2)
TRETE_FIYAT	NUMBER(7,4)
KOM	NUMBER(14,2)
OP_ID	VARCHAR2(12)

SQL> desc bulten_table

Name	Null?	Type
BULTEN_NO		NUMBER(7)
POLICE		NUMBER(7)
TARIFE		VARCHAR2(3)
BRANS		VARCHAR2(3)
TRETE		VARCHAR2(2)
SIRKET		VARCHAR2(7)
TRETE_MEBLAG		NUMBER(14)
AY		VARCHAR2(2)
YIL		VARCHAR2(4)

SQL> desc belediye_tanim_table

Name	Null?	Type
SIRKET_KOD		NUMBER(3)
IL_KOD		VARCHAR2(3)
ILCE_KOD		VARCHAR2(3)
ILCE_AD		VARCHAR2(25)
MAH_KOY		VARCHAR2(20)
CADDE		VARCHAR2(20)
SOKAK		VARCHAR2(20)
BINA_ADI		VARCHAR2(20)
NO		NUMBER(4)
DAIRE		NUMBER(3)
SEMT		VARCHAR2(20)
POSTA_KODU		VARCHAR2(6)
TARIH		DATE
OP_ID		VARCHAR2(12)

SQL> desc ek_no_table

Name	Null?	Type
SIRKET_KOD		NUMBER(3)
EKNO		VARCHAR2(4)
HESAP_NO		VARCHAR2(16)
BAS_TARIH		DATE
EKNO_ADI		VARCHAR2(25)
TARIH		DATE
OP_ID		VARCHAR2(12)

SQL> desc kaynak_table

Name	Null?	Type
SIRKET_KOD		NUMBER(3)
KAYNAK_KOD		NUMBER(3)
KISA_AD		VARCHAR2(15)
UZUN_AD		VARCHAR2(25)
IST_ORG		VARCHAR2(1)

MUST_TAKIP	VARCHAR2(1)
KOM_ALIR	VARCHAR2(1)
KOM_VERIR	VARCHAR2(1)
YETKI_KOD	NUMBER(1)
TARIH	DATE
OP_ID	VARCHAR2(12)
SIRA_NO	NUMBER(2)

SQL> desc tarife_vergi_table

Name	Null?	Type
SIRKET_KOD		NUMBER(3)
TARIFE_KOD		VARCHAR2(3)
VK		NUMBER(2)
VERGI		VARCHAR2(3)
TARIH		DATE
OP_ID		VARCHAR2(12)
TEMINAT_KOD		VARCHAR2(3)
TEMINAT_VK		NUMBER(2)

SQL> desc police_tahsilat_table

Name	Null?	Type
SIRKET_KOD		NUMBER(3)
CARI_POL_NO		NUMBER (7)
ZEYL_SIRA_ NO		NUMBER(3)
TAH_KOD		VARCHAR2(5)
VADE_TAR		DATE
TUTAR		NUMBER(14)
TARIH		DATE
OP_ID		VARCHAR2(12)

SQL> desc police_ist_table

Name	Null?	Type
SIRKET_KOD		NUMBER(3)
CARI_POL_NO		NUMBER(7)
ZEYL_SIRA_NO		NUMBER(3)
IST_KOD		VARCHAR2(3)
IST_DEGER		VARCHAR2(3)
TARIH		DATE
OP_ID		VARCHAR2(12)

```
INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- -------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- -----------------
STATUS
----------
KAYNAK_EKNO_NDX                COM

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- -------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- -----------------
STATUS
----------
KAYNAK_EKNO_TABLE              TABLE      UNIQUE

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- -------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- -----------------
STATUS
----------
SIG_NDX                         2        255         960          96

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- -------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
```

```
---------- ---------- ---------- --------- --------- ----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
                         0       0

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE   PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------


INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE   PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
VALID

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE   PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
```

```
INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- -------------- -----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE   PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------- ---------------------- ------------------------ -----------------
STATUS
-----------
MUST_TANIM_NDX                 COM

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- -------------- -----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE   PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------- ---------------------- ------------------------ -----------------
STATUS
-----------
MUST_TANIM_TABLE               TABLE      UNIQUE

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- -------------- -----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE   PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------- ---------------------- ------------------------ -----------------
STATUS
-----------
SIG_NDX                         2      255        2500       1000

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- -------------- -----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE   PCT_FREE    BLEVEL LEAF_BLOCKS
```

```
---------- ---------- ---------- --------- --------- ----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
                         0       0

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
---------- ---------- ------------ ---------- ---------- ----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------


INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
---------- ---------- ------------ ---------- ---------- ----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
VALID

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
---------- ---------- ------------ ---------- ---------- ----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
```

```
INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- -------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE   PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ ---------------------- ---------------------- -----------------
STATUS
-----------
POLICE_NDX                     COM
INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- -------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE   PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ ---------------------- ---------------------- -----------------
STATUS
-----------
POLICE_TABLE                   TABLE      UNIQUE

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- -------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE   PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ ---------------------- ---------------------- -----------------
STATUS
-----------
SIG_NDX                          2        255        2050        1022

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- -------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE   PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
```

```
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
                     0      0

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- --------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------


INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- --------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
VALID

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- --------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------


INDEX_NAME                     TABLE_OWNER
```

```
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- --------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- -----------------
STATUS
-----------
POLICE_TEMINAT_NDX             COM

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- --------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- -----------------
STATUS
-----------
POLICE_TEMINAT_TABLE           TABLE      UNIQUE

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- --------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- -----------------
STATUS
-----------
SIG_NDX                         2        255        5120        1024

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- --------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
```

```
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
                         0       0

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- --------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------


INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- --------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
VALID

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- --------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------


INDEX_NAME                     TABLE_OWNER
```

```
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- -----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE   PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ ---------------------- ---------------------- -----------------
STATUS
-----------
REASURANS_BEDEL_NDX            COM

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- -----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE   PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ ---------------------- ---------------------- -----------------
STATUS
-----------
REAURANS_BEDEL_TABLE           TABLE     UNIQUE
INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- -----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE   PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ ---------------------- ---------------------- -----------------
STATUS
-----------
SIG_NDX                          2        255         610         62

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- -----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE   PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
```

```
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
                   0                    0

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------


INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
VALID

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------


INDEX_NAME                     TABLE_OWNER
```

```
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- -----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE   PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ ---------------------- ---------------------- -----------------
STATUS
-----------
REASURANS_HEADER_NDX           COM

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- -----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE   PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ ---------------------- ---------------------- -----------------
STATUS
-----------
REASURANS_HEADER_TABLE         TABLE      UNIQUE

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- -----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE   PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ ---------------------- ---------------------- -----------------
STATUS
-----------
SIG_NDX                          2        255          130          14

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- -----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE   PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
```

```
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
                     0                    0

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------


INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
VALID

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------


INDEX_NAME                     TABLE_OWNER
```

```
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- -----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ --------------------- ---------------------- -----------------
STATUS
----------
REAS_KOM_NDX                   COM

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- -----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ --------------------- ---------------------- -----------------
STATUS
----------
REAS_KOM_TABLE                 TABLE      NONUNIQUE

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- -----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ --------------------- ---------------------- -----------------
STATUS
----------
SIG_NDX                          2        255         66          6
INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- -----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
```

```
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
---------
                     0       0

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE   PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
---------


INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE   PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
---------
VALID

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE   PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
---------


INDEX_NAME                     TABLE_OWNER
```

```
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- --------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
TARIFE_TEMINAT_NDX             COM

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- --------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
TARIFE_TEMINAT_TABLE           TABLE      UNIQUE

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- --------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
SIG_NDX                         2        255        220         22

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- --------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
```

```
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ---------------
STATUS
-----------
                         0       0

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- --------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- --------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ---------------
STATUS
-----------


INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- --------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- --------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ---------------
STATUS
-----------
VALID

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- --------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- --------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ---------------
STATUS
-----------


INDEX_NAME                     TABLE_OWNER
```

```
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ --------------------- --------------------- ----------------
STATUS
-----------
TARIFE_NDX                     COM

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ --------------------- --------------------- ----------------
STATUS
-----------
TARIFE_TABLE                   TABLE      UNIQUE

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ --------------------- --------------------- ----------------
STATUS
-----------
SIG_NDX                          2        255         20          2

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
```

```
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
                   0                    0
INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------


INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
VALID

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------


INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
```

```
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE   PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- -----------------
STATUS
-----------
TARIFE_VERGI_NDX               SYS

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE   PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- -----------------
STATUS
-----------
TARIFE_VERGI_TABLE             TABLE      UNIQUE

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE   PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- -----------------
STATUS
-----------
SIG_NDX                         2        255        1200         120

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE   PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
```

```
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
                         0                    0

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- -------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------


INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- -------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
VALID

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- -------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------


INDEX_NAME                     TABLE_OWNER
```

```
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ --------------------- --------------------- ----------------
STATUS
-----------
TEMINAT_NDX                    COM

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ --------------------- --------------------- ----------------
STATUS
-----------
TEMINAT_TABLE                  TABLE      UNIQUE

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ --------------------- --------------------- ----------------
STATUS
-----------
SIG_NDX                         2        255          50          5

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
```

```
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
                   0       0

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
VALID

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------


INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
```

```
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- --------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
TRETE_KOD_NDX                  COM

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- --------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
TRETE_KOD_TABLE                TABLE      UNIQUE

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- --------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
SIG_NDX                         2        255          4          2

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- --------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
```

```
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
                   0                    0

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- -------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------


INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- -------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
VALID

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- -------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------


INDEX_NAME                     TABLE_OWNER
```

```
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ---------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
ZEYL_HEADER_NDX                COM

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ---------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
ZEYL_HEADER_TABLE              TABLE      UNIQUE

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ---------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
SIG_NDX                         2        255         26          4

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ---------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
```

```
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
---------
                     0       0

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
---------

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
---------
VALID
INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
---------

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
```

```
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ --------------------- --------------------- -----------------
STATUS
-----------
BRANS_NDX                      COM

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ --------------------- --------------------- -----------------
STATUS
-----------
BRANS_TABLE                    TABLE      UNIQUE

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ --------------------- --------------------- -----------------
STATUS
-----------
SIG_NDX                         2        255          4          2

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
```

```
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
---------
                         0       0

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
---------


INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
---------
VALID

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
---------


INDEX_NAME                     TABLE_OWNER
```

```
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- -----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ ---------------------- ---------------------- -----------------
STATUS
-----------
EK_NO_NDX                      COM

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- -----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ ---------------------- ---------------------- -----------------
STATUS
-----------
EK_NO_TABLE                    TABLE      UNIQUE

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- -----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ ---------------------- ---------------------- -----------------
STATUS
-----------
SIG_NDX                          2        255          80          8

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- -----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
```

```
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
---------
                         0       0

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- --------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
---------


INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- --------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
---------
VALID

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- --------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
---------

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
```

```
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- -----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ ---------------------- ---------------------- -----------------
STATUS
-----------
KAY_NDX                        COM

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- -----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ ---------------------- ---------------------- -----------------
STATUS
-----------
KAYNAK_TABLE                   TABLE      UNIQUE

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- -----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ ---------------------- ---------------------- -----------------
STATUS
-----------
SYSTEM                          2        255          4          2

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- -----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
```

```
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- -----------------
STATUS
-----------
                     0       0

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE   PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- -----------------
STATUS
-----------


INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE   PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- -----------------
STATUS
-----------
VALID

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------- ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE   PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- -----------------
STATUS
-----------


INDEX_NAME                     TABLE_OWNER
```

```
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ ---------------------- ---------------------- -----------------
STATUS
-----------
IST_KOD_NDX                    COM

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ ---------------------- ---------------------- -----------------
STATUS
-----------
IST_KOD_TABLE                  TABLE      UNIQUE

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ ---------------------- ---------------------- -----------------
STATUS
-----------
SIG_NDX                          2        255         86          8

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
```

```
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
                         0       0

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- -------------- -----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------


INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- -------------- -----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
VALID

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- -------------- -----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------


INDEX_NAME                     TABLE_OWNER
```

```
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ---------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
----------
IST_DEGER_NDX                  COM
INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ---------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
----------
IST_DEGER_TABLE                TABLE      UNIQUE

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ---------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
----------
SIG_NDX                          2        255         90         10

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ---------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- -----------
```

```
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
                         0       0

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE   PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- ----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------


INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME                     TABLE_TYPE  UNIQUENES
------------------------------ ----------- ---------
TABLESPACE_NAME                INI_TRANS  MAX_TRANS INITIAL_EXTENT
NEXT_EXTENT
------------------------------ ---------- ---------- ------------ ----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE   PCT_FREE    BLEVEL LEAF_BLOCKS
----------- ----------- ------------ ---------- ---------- ----------
DISTINCT_KEYS AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY
CLUSTERING_FACTOR
------------ -------------------- -------------------- ----------------
STATUS
-----------
VALID

INDEX_NAME                     TABLE_OWNER
------------------------------ ------------------------------
TABLE_NAME
```

```
/* Copyright (c) 1988 by the Oracle Corporation */


SQL*FORMS_VERSION = 03.00.16.12.02
TERSE = ON

DEFINE FORM

  NAME = BRANS
  TITLE = BRANS
  DEFAULT_MENU_APPLICATION = DEFAULT

  DEFINE PROCEDURE

    NAME = basla

    DEFINE REFERENCE

      OWNER = sg
      APPLICATION = genproc
      PROCEDURE = basla

    ENDDEFINE REFERENCE

  ENDDEFINE PROCEDURE

  DEFINE PROCEDURE

    NAME = brans_kont
    DEFINITION = <<<
    procedure brans_kont is
    begin
    declare sayi number;
    begin
    if :giris.brans is not null then
    begin
    select 1 into sayi from brans_table
         where sirket_kod=:logo.sirket_kod and =:giris.brans;
    exception
        when no_data_found then msg_no(4,'BRANS KODU');
                                bell;
                                raise form_trigger_failure;
    end;
    select brans_adi into :giris.brans_adi from brans_table
         where sirket_kod=:logo.sirket_kod and =:giris.brans;
    end if;
    end;
    end;
    >>>
  DEFINE TRIGGER

    NAME = PRE-FORM
    TRIGGER_TYPE = V3
    TEXT = <<<
    basla(:logo.sirket_kod,:logo.sirket_adi,:logo.sube_kod,:logo.sube_ad,
        :logo.bugun);
```

```
  >>>

ENDDEFINE TRIGGER

DEFINE TRIGGER

  NAME = KEY-COMMIT
  TRIGGER_TYPE = V3
  TEXT = <<<
  brans_kont;
  commit_form;
  if form_success then msg_no(11,null);
                  bell;
                  go_field('giris.gds');
                  raise form_trigger_failure;
  else msg_no(12,null);
      bell;
      go_field('giris.gds');
      raise form_trigger_failure;
  end if;
  >>>

ENDDEFINE TRIGGER

DEFINE TRIGGER

  NAME = KEY-EXIT
  TRIGGER_TYPE = V3
  TEXT = <<<
  clear_form(no_validate,full_rollback);
  execute_trigger('pre-form');
  go_field('gds');
  >>>

ENDDEFINE TRIGGER

DEFINE TRIGGER

  NAME = KEY-HELP
  TRIGGER_TYPE = V3
  TEXT = <<<
  help;
  >>>

ENDDEFINE TRIGGER

DEFINE TRIGGER

  NAME = KEY-LISTVAL
  TRIGGER_TYPE = V3
  TEXT = <<<
  list_values;
  >>>

ENDDEFINE TRIGGER
```

```
DEFINE TRIGGER

  NAME = KEY-OTHERS
  TRIGGER_TYPE = V3
  TEXT = <<<
  bell;
  >>>

ENDDEFINE TRIGGER

DEFINE TRIGGER

  NAME = KEY-PRINT
  TRIGGER_TYPE = V3
  TEXT = <<<
  print;
  >>>

ENDDEFINE TRIGGER

DEFINE BLOCK

  NAME = logo
  ROWS_DISPLAYED = 1
  ROWS_BUFFERED = 1
  BASE_LINE = 1
  LINES_PER_ROW = 0
  ARRAY_SIZE = 0

  DEFINE FIELD

    NAME = sirket_adi
    DATATYPE = CHAR
    LENGTH = 20
    DISPLAY_LENGTH = 20
    QUERY_LENGTH = 20
    BASE_TABLE = OFF
    PAGE = 1
    LINE = 1
    COLUMN = 1
    INPUT = OFF
    UPDATE = OFF
    QUERY = OFF

  ENDDEFINE FIELD

  DEFINE FIELD

    NAME = bugun
    DATATYPE = CHAR
    LENGTH = 10
    DISPLAY_LENGTH = 10
    QUERY_LENGTH = 10
    BASE_TABLE = OFF
    PAGE = 1
    LINE = 1
```

```
  COLUMN = 71
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = sirket_kod
  DATATYPE = NUMBER
  LENGTH = 3
  DISPLAY_LENGTH = 3
  QUERY_LENGTH = 3
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = sube_kod
  DATATYPE = NUMBER
  LENGTH = 4
  DISPLAY_LENGTH = 4
  QUERY_LENGTH = 4
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = sube_ad
  DATATYPE = CHAR
  LENGTH = 20
  DISPLAY_LENGTH = 20
  QUERY_LENGTH = 20
  BASE_TABLE = OFF
  PAGE = 1
  LINE = 2
  COLUMN = 1
```

```
    INPUT = OFF
    UPDATE = OFF
    QUERY = OFF

  ENDDEFINE FIELD

ENDDEFINE BLOCK

DEFINE BLOCK

  NAME = giris
  ROWS_DISPLAYED = 1
  ROWS_BUFFERED = 1
  BASE_LINE = 1
  LINES_PER_ROW = 0
  ARRAY_SIZE = 0

ENDDEFINE BLOCK

 DEFINE TRIGGER

    NAME = KEY-COMMIT
    TRIGGER_TYPE = V3
    TEXT = <<<
    bell;
    >>>

 ENDDEFINE TRIGGER

 DEFINE TRIGGER

    NAME = KEY-DELREC
    TRIGGER_TYPE = V3
    TEXT = <<<
    bell;
    >>>

 ENDDEFINE TRIGGER

 DEFINE FIELD

    NAME = gds
    DATATYPE = CHAR
    LENGTH = 1
    DISPLAY_LENGTH = 1
    QUERY_LENGTH = 1
    BASE_TABLE = OFF
    PAGE = 1
    LINE = 22
    COLUMN = 8
    QUERY = OFF
    UPPERCASE = ON

ENDDEFINE FIELD

 DEFINE TRIGGER
```

```
    NAME = KEY-EXIT
    TRIGGER_TYPE = V3
    TEXT = <<<
    exit_form;
    >>>

  ENDDEFINE TRIGGER

  DEFINE TRIGGER

    NAME = KEY-NXTFLD
    TRIGGER_TYPE = V3
    TEXT = <<<
    if :gds is null then msg_no(10,null);
                    bell;
                    raise form_trigger_failure;
    end if;
    if :gds not in ('G','D','S') then msg_no(36,null);
                                      bell;
                                      raise form_trigger_failure;
    end if;
    declare sakla_gds char(1);
    begin
    sakla_gds := :giris.gds;
    clear_block(no_commit);
    go_block('blok1');
    clear_block(no_commit);
    :giris.gds := sakla_gds;
    go_block('giris');
    end;
    go_field('giris.');
    >>>

  ENDDEFINE TRIGGER

ENDDEFINE FIELD

ENDDEFINE FIELD

DEFINE FIELD

  NAME = adi
  DATATYPE = CHAR
  LENGTH = 5
  DISPLAY_LENGTH = 5
  QUERY_LENGTH = 5
  BASE_TABLE = OFF
  PAGE = 1
  LINE = 5
  COLUMN = 19
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF
  UPPERCASE = ON

ENDDEFINE FIELD
```

```
DEFINE FIELD

  NAME =
  DATATYPE = CHAR
  LENGTH = 3
  DISPLAY_LENGTH = 3
  QUERY_LENGTH = 3
  BASE_TABLE = OFF
  PAGE = 1
  LINE = 6
  COLUMN = 14
  LOV_TEXT = <<<
  select ,brans_adi into :giris. from brans_table
  where sirket_kod=:logo.sirket_kod
   and brans_adi like :giris.||'%'
  >>>
  LOV_TITLE = BRANS KOD VE ADLARI
  QUERY = OFF
  UPPERCASE = ON

  DEFINE TRIGGER

    NAME = KEY-NXTFLD
    TRIGGER_TYPE = V3
    TEXT = <<<
    if :giris. is null then msg_no(10,null);
                       bell;
                       raise form_trigger_failure;
   end if;
   begin
   select 1 into sayi from brans_table
        where sirket_kod=:logo.sirket_kod and =:giris.;
        sayi:=1
   exception
    when no_data_found then sayi:=0
   end;
   if :giris.gds ='G`  and  sayi = 1  then
      msg_no(13,null);
      bell;
      raise_form_trigger_failure;
   end;
   if :giris.gds = `D` and sayi = 0  then
      msg_no(14,null);
      bell;
      raise_form_trigger_failure;
   end;
   go_block(`blok1`);
   execute_query;

  DEFINE TRIGGER

    NAME = KEY-PRVFLD
    TRIGGER_TYPE = V3
    TEXT = <<<
    previous_field;
    >>>
```

```
  ENDDEFINE TRIGGER

ENDDEFINE FIELD

  DEFINE TRIGGER

    NAME = KEY-PRVFLD
    TRIGGER_TYPE = V3
    TEXT = <<<
    previous_field;
    >>>

  ENDDEFINE TRIGGER

ENDDEFINE FIELD

DEFINE FIELD

  NAME = sil
  DATATYPE = CHAR
  LENGTH = 1
  DISPLAY_LENGTH = 1
  QUERY_LENGTH = 1
  BASE_TABLE = OFF
  PAGE = 1
  LINE = 22
  COLUMN = 80
  QUERY = OFF
  UPPERCASE = ON

  DEFINE TRIGGER

    NAME = KEY-LISTVAL
    TRIGGER_TYPE = V3
    TEXT = <<<
    bell;
    >>>

  ENDDEFINE TRIGGER

  DEFINE TRIGGER

    NAME = KEY-NXTFLD
    TRIGGER_TYPE = V3
    TEXT = <<<
    declare sayac number;
    begin
    if :giris.sil is null then msg_no(10,null);
                        bell;
                        raise form_trigger_failure;
    end if;
    if :giris.sil  not in ('H','E') then msg_no(9,null);
                                    bell;
                                    raise form_trigger_failure;
    end if;
    if :giris.sil ='E' then
```

```
        commit_form;
        if form_success then msg_no(11,null);
                        bell;
                        go_field('giris.gds');
                        raise form_trigger_failure;
        else msg_no(12,null);
            bell;
            go_field('giris.gds');
            raise form_trigger_failure;
        end if;
      end;
      >>>

    ENDDEFINE TRIGGER

  ENDDEFINE FIELD

  DEFINE TRIGGER

    NAME = KEY-DELREC
    TRIGGER_TYPE = V3
    TEXT = <<<
    bell;
    >>>

  ENDDEFINE TRIGGER

DEFINE BLOCK

  NAME = blok1
  TABLE = brans_table
  ROWS_DISPLAYED = 1
  ROWS_BUFFERED = 1
  BASE_LINE = 1
  LINES_PER_ROW = 0
  ARRAY_SIZE = 0

  DEFINE FIELD

    NAME = sirket_kod
    DATATYPE = NUMBER
    LENGTH = 3
    DISPLAY_LENGTH = 3
    QUERY_LENGTH = 3
    DISPLAYED = OFF
    PAGE = 0
    LINE = 243
    COLUMN = 0
    EDIT_WORD_WRAP = OFF
    ENFORCE_KEY_FROM = logo.sirket_kod
    ECHO = OFF
    INPUT = OFF
    UPDATE = OFF
    QUERY = OFF

  ENDDEFINE FIELD
```

```
DEFINE FIELD

  NAME =brans_kod
  DATATYPE = CHAR
  LENGTH = 3
  DISPLAY_LENGTH = 3
  QUERY_LENGTH = 3
  DISPLAYED = OFF
  PAGE = 0
  LINE = 243
  COLUMN = 0
  ENFORCE_KEY_FROM = giris.brans_kod
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = tarih
  DATATYPE = DATE
  LENGTH = 10
  DISPLAY_LENGTH = 10
  QUERY_LENGTH = 10
  DISPLAYED = OFF
  PAGE = 0
  LINE = 243
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = op_id
  DATATYPE = CHAR
  LENGTH = 12
  DISPLAY_LENGTH = 12
  QUERY_LENGTH = 12
  DISPLAYED = OFF
  PAGE = 0
  LINE = 243
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE TRIGGER
```

```
    NAME = PRE-INSERT
    TRIGGER_TYPE = V3
    TEXT = <<<
    :blok1.tarih:=trunc(sysdate);
    :blok1.op_id:=substr(user,5,12);
    >>>

  ENDDEFINE TRIGGER

  DEFINE TRIGGER

    NAME = PRE-UPDATE
    TRIGGER_TYPE = V3
    TEXT = <<<
    :blok1.tarih:=trunc(sysdate);
    :blok1.op_id:=substr(user,5,12);
    >>>

  ENDDEFINE TRIGGER

  DEFINE FIELD

    NAME = brans_kod_adi
    DATATYPE = CHAR
    LENGTH = 15
    DISPLAY_LENGTH = 15
    QUERY_LENGTH = 15
    PAGE = 1
    LINE = 6
    COLUMN = 19
    INPUT = OFF
    UPDATE = OFF
    QUERY = OFF
    UPPERCASE = ON

  ENDDEFINE FIELD

  DEFINE FIELD

    NAME = brans_ing-adi
    DATATYPE = CHAR
    LENGTH = 15
    DISPLAY_LENGTH = 15
    QUERY_LENGTH = 15
    PAGE = 1
    LINE = 6
    COLUMN = 35
    INPUT = OFF
    UPDATE = OFF
    QUERY = OFF
    UPPERCASE = ON

  ENDDEFINE FIELD

ENDDEFINE BLOCK
```

```
DEFINE SCREEN

  DEFINE PAGE

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

                              BRANS TANIMLAMA
    >>>
    LINE = 4
    BOILER = <<<
    BRANS KODU  :
    BRANS ADI   :                BRANS ING. ADI:
    >>>
    LINE = 22
    BOILER = <<<
    G/D/S :                           E/H:
    >>>

  ENDDEFINE PAGE


ENDDEFINE SCREEN

ENDDEFINE FORM
```

```
/* Copyright (c) 1988 by the Oracle Corporation */


SQL*FORMS_VERSION = 03.00.16.12.02
TERSE = ON

DEFINE FORM

  NAME = DAGILIM
  TITLE = DAGILIM
  DEFAULT_MENU_APPLICATION = DEFAULT

  DEFINE PROCEDURE

    NAME = basla

    DEFINE REFERENCE

      OWNER = sg
      APPLICATION = genproc
      PROCEDURE = basla

    ENDDEFINE REFERENCE

  ENDDEFINE PROCEDURE

  DEFINE PROCEDURE

    NAME = brans_kont
    DEFINITION = <<<
    procedure brans_kont is
    begin
    declare sayi number;
    begin
    if :brans_kodu is not null then
    begin
    select /*+ (brans_table brans_ndx) */
          1 into sayi from brans_table
          where sirket_kod=:logo.sirket_kod and brans_kod=:brans_kodu;
    exception
         when no_data_found then msg_no(4,'BRANS KODU');
                                 bell;
                                 raise form_trigger_failure;
    end;
    select /*+ (brans_table brans_ndx) */
          brans_adi into :brans_adi from brans_table
          where sirket_kod=:logo.sirket_kod and brans_kod=:brans_kodu;
    end if;
    end;
    end;
    >>>

  ENDDEFINE PROCEDURE

  DEFINE PROCEDURE
    NAME = genel
```

```
DEFINITION = <<<
procedure genel(tar in tarife_table.tarife_kod%type,
                pol in police_teminat_table.cari_pol_no%type,
                total1 in number,
                total2 in number,
                a_dov in doviz_table.doviz%type,
                m_dov in doviz_table.doviz%type,
                b_dov in doviz_table.doviz%type) is
tutar        number;
sirket_meblag number;
kur          char;
trete        char(2);
t_tip        char(15);
kon          number;
xkur         number(9);
begin
begin
  if a_dov != :dov and a_dov is not null then
    bedel_bul(a_dov,xkur);
    :asg_bedel := :asg_bedel * xkur;
  end if;
  if m_dov != :dov and m_dov is not null then
    bedel_bul(m_dov,xkur);
    :max_bedel := :max_bedel * xkur;
  end if;
  if b_dov != :dov and b_dov is not null then
    bedel_bul(b_dov,xkur);
    :bedel:=nvl(:bedel,0) * xkur;
  end if;
-- ORAN DOLU ISE
if :oran is not null then
if :b_k ='B' then tutar:=total2*:oran/:matrah;end if;
if :b_k ='K' then tutar:=:kalan*:oran/:matrah;end if;
 if nvl(:max_bedel,0)=0 then :trete_sirket_meblag:=tutar;
  elsif nvl(:max_bedel,0) <= nvl(tutar,0) then
        :trete_sirket_meblag:=:max_bedel;
  elsif  nvl(:max_bedel,0) > nvl(tutar,0) then
        :trete_sirket_meblag:=tutar;
 end if;
 if nvl(:asg_bedel,0) > 0 then
   if nvl(:asg_bedel,0) >= nvl(:trete_sirket_meblag,0) and
     nvl(:asg_bedel,0) >=nvl(:kalan,0) then
     :trete_sirket_meblag:=:kalan;
   end if;
   if nvl(:asg_bedel,0) >= nvl(:trete_sirket_meblag,0) and
     nvl(:asg_bedel,0) < nvl(:kalan,0) then
     :trete_sirket_meblag:=:asg_bedel;
   end if;
 end if;
 :trete_sirket_prim:=(:trete_sirket_meblag/total2) * total1;
 procel(tar,total1,total2,pol);
 :plen:=null;
 :bedel:=null;
end if;
--PLEN DOLU ISE
if :plen is not null then
```

```
 select trete_tip into t_tip from trete_tip_table
     where sirket_kod=:logo.sirket_kod and sira_no=1;
 select trete_kod into trete from trete_kod_table
     where sirket_kod=:logo.sirket_kod and trete_tip=t_tip;
if :zeyl_sira_no=0 then
 begin
 select /*+ index(reas_dagilim_table reas_dagilim_ndx) */
       trete_sirket_meblag into sirket_meblag
      from reas_dagilim_table
      where sirket_kod=:logo.sirket_kod and cari_pol_no=:cari_pol_no and
           zeyl_sira_no=:zeyl_sira_no and trete_kodu=trete and
           brans_kodu=:brans_kodu and ay=:ay and yil=:yil;
   kon := sirket_meblag;
   tutar:=kon * :plen;
   exception when no_data_found then kon:=0;
                              tutar:=kon * :plen;
           when too_many_rows then message('too1');pause;
end;
else
 begin
 select trete_sirket_meblag into sirket_meblag
      from reas_gecici_table
      where sirket_kod=:logo.sirket_kod and cari_pol_no=:cari_pol_no and
           zeyl_sira_no=:zeyl_sira_no and trete_kodu=trete;
   kon := sirket_meblag;
   tutar:=kon * :plen;
   exception when no_data_found then kon:=0;
                              tutar:=kon * :plen;
end;
end if;
  if nvl(:max_bedel,0)=0 then
    :trete_sirket_meblag:=tutar;
end if;
if nvl(:max_bedel,0) > 0 then
  if nvl(:max_bedel,0) >= nvl(tutar,0) then :trete_sirket_meblag:=tutar;
   else :trete_sirket_meblag:=:max_bedel;
  end if;
end if;
  if nvl(:trete_sirket_meblag,0) >= nvl(:kalan,0) then
    :trete_sirket_meblag:=:kalan;
  end if;
 proce1(tar,total1,total2,pol);
:bedel:=null;
:oran:=null;
end if;
--BEDEL DOLU ISE
if :bedel is not null then
  if nvl(:bedel,0) >= nvl(:kalan,0) then :trete_sirket_meblag:=:kalan;
   else
     :trete_sirket_meblag:=:bedel;
  end if;
  if nvl(:max_bedel,0) > 0 and
    nvl(:max_bedel,0) <= nvl(:trete_sirket_meblag,0) then
    :trete_sirket_meblag:=:max_bedel;
  end if;
  if nvl(:asg_bedel,0) > 0 then
```

```
      if nvl(:asg_bedel,0) > nvl(:trete_sirket_meblag,0) and
        nvl(:asg_bedel,0) < nvl(:kalan,0) then
        :trete_sirket_meblag:=:asg_bedel;
      end if;
    end if;
   proce1(tar,total1,total2,pol);
   :plen:=null;
   :oran:=null;
  end if;
  end;
  end;
  >>>

ENDDEFINE PROCEDURE

DEFINE PROCEDURE

  NAME = genel_kont
  DEFINITION = <<<
  procedure genel_kont is
  begin
  if :brans_kodu is null then msg_no(3,'BRANS KODU');
                bell;
                raise form_trigger_failure;
  end if;
  brans_kont;
  if :ay is null then msg_no(3,'AY');
                bell;
                go_field('ay');
                raise form_trigger_failure;
  end if;
  declare month date;
  begin
   month:=to_date(:ay,'mm');
   exception
    when others then msg_no(26,null);
                bell;
                raise form_trigger_failure;
  end;
  :ay :=lpad(rtrim(ltrim(:ay,' '),' '),2,'0');
  if :yil is null then msg_no(3,'YIL');
                bell;
                raise form_trigger_failure;
  end if;
  declare year date;
  begin
   year:=to_date(:yil,'yyyy');
   exception
    when others then msg_no(26,null);
                bell;
                raise form_trigger_failure;
  end;
  /*
  declare s_yil number;
        ilk_yil number;
  begin
```

```
s_yil:=to_number(to_char(sysdate,'yyyy') ) + 1 ;
ilk_yil :=s_yil-2;
if to_number(:yil)>s_yil or to_number(:yil)<ilk_yil
  then msg_no(26,' YIL '||to_char(ilk_yil)||' ILE '||
            to_char(s_yil)|| ' ARASINDA OLMALIDIR');
  bell;
  raise form_trigger_failure;
end if;
end;
*/
end;
>>>
```

ENDDEFINE PROCEDURE

DEFINE PROCEDURE

```
NAME = islem
DEFINITION = <<<
procedure islem(tar in tarife_table.tarife_kod%type,
          top1 in number,
          top2 in number,
          pol in police_teminat_table.cari_pol_no%type) is
ara_yil char(4);
yil_k char(1);
t_tip char(15);
trete char(3);
ist char(3);
begin
   begin
   select /*+ index(reasurans_header_table reasurans_header_ndx) */
       yil_kodu into yil_k from reasurans_header_table
     where sirket_kod=:logo.sirket_kod and
          tanzim_yili=:tanzim_yili and
          prim_hasar='P' and
          brans=:brans_kodu and
          tarife=tar and
          sira_no=:sno and
          rownum=1;
     :sayi:=2;
     exception when no_data_found then :sayi:=1;
           when too_many_rows then msg_no(13,'REAS.HEADER TABLE');
                        bell;
                        raise form_trigger_failure;
   end;
   if :sayi=1 then
     begin
     select /*+ index(reasurans_header_table reasurans_header_ndx) */
         yil_kodu into yil_k from reasurans_header_table
     where sirket_kod=:logo.sirket_kod and
          tanzim_yili=:tanzim_yili and
          prim_hasar='P' and
          brans=:brans_kodu and
          tarife is null and
          sira_no=:sno and
          rownum=1;
```

```
    exception
          when no_data_found then :sno:=:sno;
          when too_many_rows then msg_no(13,'reas_header_table');
                                  bell;
                                  raise form_trigger_failure;
    end;
    if yil_k='R' then
      ara_yil:=:ryil;
    elsif yil_k='U' then
      ara_yil:=:uyil;
    end if;
  begin
    select /*+ index(reasurans_header_table reasurans_header_ndx) */
        trete_kodu into :trete_kodu
        from reasurans_header_table
    where sirket_kod=:logo.sirket_kod and
        tanzim_yili=:tanzim_yili and
        prim_hasar='P' and
        brans=:brans_kodu and
        yil_kodu=yil_k and
        yili=ara_yil and
        tarife is null and
        sira_no=:sno and
        rownum=1;
    exception when NO_DATA_FOUND  then
--KALAN SIFIRDAN BUYUKSE VE SIRA NO YOKSA KALAN, IHTIYARI TRETE KODUNA
--YEDIRILIR
  if :kalan > 0 then
      select trete_kod into trete from trete_kod_table
       where sirket_kod=:logo.sirket_kod and trete_tip=t_tip;
    if :zeyl_sira_no=0 then
    begin
     insert into reas_dagilim_table values (:logo.sirket_kod,
     :sube,:kaynak,:acenta_no,:cari_pol_no,
     :zeyl_turu,:zeyl_sira_no,:t_i,:brans_kodu,
     :tanzim_yili,'P',null,:pol_bas_tarih,
     :pol_bit_tarih,trete,null,:kalan,top1,null,
     :ay,:yil,trunc(sysdate),
     substr(user,5,12),:tarife,null,:tanzim_tarihi,:yukleme_tarihi,0);
     proce2(top1 in number,top2 in number);
      exception when others then
         insert into cari_dagilim_hata_table values(:cari_pol_no,
              :zeyl_sira_no);
     end;
    else
     insert into reas_gecici_table values (:logo.sirket_kod,
     :sube,:kaynak,:acenta_no,:cari_pol_no,
     :zeyl_turu,:zeyl_sira_no,:t_i,:brans_kodu,
     :tanzim_yili,'P',null,:pol_bas_tarih,
     :pol_bit_tarih,trete,null,:kalan,top1,null,
     :ay,:yil,trunc(sysdate),
     substr(user,5,12),:tarife,null,:tanzim_tarihi,:yukleme_tarihi,0);
     :global.top1:=top1;
     :global.top2:=top2;
    end if;
    else :trete_kodu:=null;
```

```
  end if;
 end;
   begin
   select /*+ index(reasurans_header_table reasurans_header_ndx) */
        ist_kod into ist from reasurans_header_table
    where sirket_kod=:logo.sirket_kod and
         tanzim_yili=:tanzim_yili and
         prim_hasar='P' and
         brans=:brans_kodu and
         yil_kodu=yil_k and
         yili=ara_yil and
         tarife is null  and
         trete_kodu=:trete_kodu and sira_no=:sno and rownum=1;
   if ist is not null then
    pro2(yil_k,ara_yil,tar,pol,top1,top2);
   else
    pro1(yil_k,ara_yil,tar,top1,top2,pol);
   end if;
    exception when no_data_found then :trete_kodu:=null;
   end;
--TARIFE DOLU ISE
  elsif :sayi=2 then
     begin
     select /*+ index(reasurans_header_table reasurans_header_ndx) */
          yil_kodu into yil_k from reasurans_header_table
     where sirket_kod=:logo.sirket_kod and
          tanzim_yili=:tanzim_yili and
          prim_hasar='P' and
          brans=:brans_kodu and
          tarife=tar and sira_no=:sno and rownum=1;
     exception when no_data_found then   :sno:=:sno;
               when too_many_rows then msg_no(13,'YIL KODU');
                                       bell;
                                       raise form_trigger_failure;
    end;
     if yil_k='R' then
       ara_yil:=:ryil;
     elsif yil_k='U' then
       ara_yil:=:uyil;
     end if;
  begin
   select /*+ index(reasurans_header_table reasurans_header_ndx) */
        trete_kodu into :trete_kodu
        from reasurans_header_table
   where sirket_kod=:logo.sirket_kod and
        tanzim_yili=:tanzim_yili and
        prim_hasar='P' and
        brans=:brans_kodu and
        yil_kodu=yil_k and
        yili=ara_yil and
        tarife=tar and
        sira_no=:sno and
        rownum=1;
   exception when no_data_found then
--KALAN SIFIRDAN BUYUKSE KALAN, IHTIYARI TRETE KODUNA YEDIRILIR
   if :kalan > 0 then
```

```
      select trete_kod into trete from trete_kod_table
       where sirket_kod=:logo.sirket_kod and trete_tip=t_tip;
    if :zeyl_sira_no=0 then
    begin
    insert into reas_dagilim_table values (:logo.sirket_kod,
     :sube,:kaynak,:acenta_no,:cari_pol_no,
     :zeyl_turu,:zeyl_sira_no,:t_i,:brans_kodu,
     :tanzim_yili,'P',null,:pol_bas_tarih,
     :pol_bit_tarih,trete,null,:kalan,top1,null,
     :ay,:yil,trunc(sysdate),
     substr(user,5,12),:tarife,null,:tanzim_tarihi,:yukleme_tarihi,0);
     proce2(top1 in number,top2 in number);
      exception when others then
         insert into cari_dagilim_hata_table values (:cari_pol_no,
              :zeyl_sira_no);
     end;
    else
     insert into reas_gecici_table values (:logo.sirket_kod,
     :sube,:kaynak,:acenta_no,:cari_pol_no,
     :zeyl_turu,:zeyl_sira_no,:t_i,:brans_kodu,
     :tanzim_yili,'P',null,:pol_bas_tarih,
     :pol_bit_tarih,trete,null,:kalan,top1,null,
     :ay,:yil,trunc(sysdate),
     substr(user,5,12),:tarife,null,:tanzim_tarihi,:yukleme_tarihi,0);
    end if;
    else :trete_kodu:=null;
   end if;
  end;
    begin
    select /*+ index(reasurans_header_ndx) */
         ist_kod into ist from reasurans_header_table
     where sirket_kod=:logo.sirket_kod and
         tanzim_yili=:tanzim_yili and
         prim_hasar='P' and
         brans=:brans_kodu and
         yil_kodu=yil_k and
         yili=ara_yil and
         tarife=tar  and
         trete_kodu=:trete_kodu and
         sira_no=:sno and rownum=1;
    if ist is not null then
     pro2(yil_k,ara_yil,tar,pol,top1,top2);
    else
     pro1(yil_k,ara_yil,tar,top1,top2,pol);
    end if;
     exception when no_data_found then :trete_kodu:=0;
     end;
 end if;
end;
>>>
```

ENDDEFINE PROCEDURE

DEFINE PROCEDURE

NAME = msg_no

```
  DEFINE REFERENCE

    OWNER = sg
    APPLICATION = genproc
    PROCEDURE = msg_no

  ENDDEFINE REFERENCE

ENDDEFINE PROCEDURE

DEFINE PROCEDURE

  NAME = pol_varlik
  DEFINITION = <<<
  procedure pol_varlik is
  sayi number;
  begin
  if :pol_no is not null then
     begin
     select /*+ index(police_teminat_table police_teminat_ndx) */
          1 into sayi from police_teminat_table
       where sirket_kod=:logo.sirket_kod and cari_pol_no=:pol_no;
     exception when no_data_found then msg_no(4,'POLICE NO');
                                bell;
                                raise form_trigger_failure;
               when too_many_rows then sayi:=0;
     end;
  end if;
  end;
  >>>

ENDDEFINE PROCEDURE

DEFINE PROCEDURE

  NAME = pro1
  DEFINITION = <<<
  procedure pro1 (y_k in reasurans_header_table.yil_kodu%type,
                  yl in reasurans_header_table.yili%type,
                  tr in tarife_table.tarife_kod%type,
                  total1 in number,
                  total2 in number,
                  pol in police_teminat_table.cari_pol_no%type) is
  tutar number;
  kon number;
  a_dov char(3);
  m_dov char(3);
  b_dov char(3);
  t_k char;
  begin
  if :sayi=1 then
  begin
  select /*+ index(reasurans_bedel_table reasurans_bedel_ndx) */
       oran,plen,bedel,b_k,max_bedel,max_doviz,matrah,
       asg_bedel,asg_doviz,bedel_doviz into
       :oran,:plen,:bedel,:b_k,:max_bedel,m_dov,:matrah,:asg_bedel,a_dov,
```

```
      b_dov
      from reasurans_bedel_table
       where sirket_kod=:logo.sirket_kod and
            tanzim_yili=:tanzim_yili and
            prim_hasar='P' and
            brans=:brans_kodu and
            yil_kodu=y_k and
            yili=yl and
            tarife is null and
            trete_kodu=:trete_kodu and
            sira_no=:sno;
  genel(tr,pol,total1,total2,a_dov,m_dov,b_dov);
  exception when no_data_found then msg_no(4,'ORAN PLEN BEDELDEN BIRI');
                              bell;
                              raise form_trigger_failure;
          when too_many_rows then msg_no(13,'REASURANS_HEADER_TABLEDA');
                              bell;
                              raise form_trigger_failure;
  end;
  elsif :sayi=2 then
  begin
   select /*+ index(reasurans_bedel_table reasurans_bedel_ndx) */
          oran,plen,bedel,b_k,max_bedel,max_doviz,matrah,
       asg_bedel,asg_doviz,bedel_doviz into
       :oran,:plen,:bedel,:b_k,:max_bedel,m_dov,:matrah,:asg_bedel,a_dov,
       b_dov
    from reasurans_bedel_table
         where sirket_kod=:logo.sirket_kod and
            tanzim_yili=:tanzim_yili and
            prim_hasar='P' and
            brans=:brans_kodu and
            yil_kodu=y_k and
            yili=yl and
            tarife=tr and
            trete_kodu=:trete_kodu and
            sira_no=:sno;
  genel(tr,pol,total1,total2,a_dov,m_dov,b_dov);
  exception when no_data_found then msg_no(4,'ORAN PLEN BEDELDEN BIRI');
                              bell;
                              raise form_trigger_failure;
  end;
  end if;
  end;
  >>>

ENDDEFINE PROCEDURE

DEFINE PROCEDURE

  NAME = pro2
  DEFINITION = <<<
  procedure pro2 (y_k in reasurans_header_table.yil_kodu%type,
              yl in reasurans_header_table.yili%type,
              tr in tarife_table.tarife_kod%type,
              pol in police_table.cari_pol_no%type,
              total1 in number,
```

```
total2 in number) is
a_dov char(3);
m_dov char(3);
b_dov char(3);
begin
:t_deger:=null;
if :sayi=1 then
declare cursor cr is
 select /*+ index(reasurans_header_table reasurans_header_ndx) */
      ist_kod from reasurans_header_table
 where sirket_kod=:logo.sirket_kod and
      tanzim_yili=:tanzim_yili and
      prim_hasar= 'P' and
      brans=:brans_kodu and
      yil_kodu=y_k and
      yili=yl and
      tarife is null and
      trete_kodu =:trete_kodu and
      sira_no=:sno;
  deger char(3);
  begin
  for kayit in cr
  loop
   begin
    select /*+ index(police_ist_table police_ist_ndx) */
         ist_degeri into deger from police_ist_table
     where sirket_kod=:logo.sirket_kod and cari_pol_no=pol and
          zeyl_sira_no=:zeyl_sira_no and
          ist_kodu=kayit.ist_kod;
    sub(deger);
   end;
  end loop;
  :t_deger:=translate(:t_deger,'!',' ');
  select /*+ index(reasurans_bedel_table reasurans_bedel_ndx) */
       oran,plen,bedel,max_bedel,max_doviz,asg_bedel,asg_doviz,
       matrah,b_k,bedel_doviz
    into :oran,:plen,:bedel,:max_bedel,m_dov,:asg_bedel,a_dov,
       :matrah,:b_k,b_dov
  from reasurans_bedel_table
  where sirket_kod=:logo.sirket_kod and tanzim_yili=:tanzim_yili and
      prim_hasar= 'P' and brans=:brans_kodu and yil_kodu=y_k and
      yili=yl and tarife is null and trete_kodu =:trete_kodu and
      sira_no=:sno and reas_key=:t_deger;
  exception
     when no_data_found then msg_no(4,'REAS.BILGILERI EKSIK');
                    bell;
                    raise form_trigger_failure;
  end;
genel(tr,pol,total1,total2,a_dov,m_dov,b_dov);
elsif :sayi=2 then
declare cursor cr is
 select /*+ index(reasurans_header_table reasurans_header_ndx) */
      ist_kod from reasurans_header_table
 where sirket_kod=:logo.sirket_kod and tanzim_yili=:tanzim_yili and
      prim_hasar= 'P' and brans=:brans_kodu and yil_kodu=y_k and
      yili=yl and tarife=tr and trete_kodu =:trete_kodu and
```

```
 sira_no=:sno;
deger char(3);
begin
for kayit in cr
loop
 begin
  select /*+ index(police_ist_table police_ist_ndx) */
       ist_degeri into deger from police_ist_table
   where sirket_kod=:logo.sirket_kod and cari_pol_no=pol and
        zeyl_sira_no=:zeyl_sira_no and
        ist_kodu=kayit.ist_kod;
  sub(deger);
 end;
end loop;
 :t_deger:=translate(:t_deger,'!',' ');
 select /*+ index(reasurans_bedel_table reasurans_bedel_ndx) */
      oran,plen,bedel,max_bedel,max_doviz,asg_bedel,asg_doviz,
      matrah,b_k,bedel_doviz
     into :oran,:plen,:bedel,:max_bedel,m_dov,:asg_bedel,a_dov,
        :matrah,:b_k,b_dov
 from reasurans_bedel_table
 where sirket_kod=:logo.sirket_kod and tanzim_yili=:tanzim_yili and
     prim_hasar= 'P' and brans=:brans_kodu and yil_kodu=y_k and
     yili=yl and tarife=tr and trete_kodu =:trete_kodu and
     sira_no=:sno and reas_key=:t_deger;
 exception
   when no_data_found then msg_no(4,'REAS.BILGILERI EKSIK');
                bell;
                raise form_trigger_failure;
 end;
genel(tr,pol,total1,total2,a_dov,m_dov,b_dov);
end if;
end;
>>>
```

ENDDEFINE PROCEDURE

DEFINE PROCEDURE

```
NAME = pro_kont
DEFINITION = <<<
procedure pro_kont is
sayi number;
begin
declare cursor cr is
  select /*+ (tarife_teminat_table tarife_tem_ndx) */
       tarife_kod from tarife_teminat_table
       where sirket_kod=:logo.sirket_kod and brn=:brans_kodu;
  begin
  for kayit in cr
  loop
   begin
   select /*+ (tarife_kapali_table tarife_kapali_ndx) */
        1 into sayi from tarife_kapali_table
   where sirket_kod=:logo.sirket_kod and yil=:yil and ay=:ay
       and tarife_kod=kayit.tarife_kod;
```

```
      exception
        when no_data_found then msg_no(95,null);
                                bell;
                                raise form_trigger_failure;
      end;
     end loop;
   end;
  end;
  >>>
```

ENDDEFINE PROCEDURE

DEFINE PROCEDURE

```
  NAME = proce1
  DEFINITION = <<<
  procedure proce1(tar in tarife_table.tarife_kod%type,
                   top1 in number,
                   top2 in number,
                   pol in police_teminat_table.cari_pol_no%type) is
  meblag    number;
  tip1      char(15);
  trete_meb number;
  meb       number;
  begin
  if :zeyl_sira_no=0 then
   begin
   insert into reas_dagilim_table values (:logo.sirket_kod,
   :sube,:kaynak,:acenta_no,pol,
   :zeyl_turu,:zeyl_sira_no,:t_i,:brans_kodu,
   :tanzim_yili,'P',null,:pol_bas_tarih,
   :pol_bit_tarih,:trete_kodu,null,:trete_sirket_meblag,:trete_sirket_prim
   ,null,:ay,:yil,trunc(sysdate),
   substr(user,5,12),:tarife,null,:tanzim_tarihi,:yukleme_tarihi,0);
    exception when others then
     insert into cari_dagilim_hata_table values(pol,
                 :zeyl_sira_no);
    end;
  else
   insert into reas_gecici_table values (:logo.sirket_kod,
   :sube,:kaynak,:acenta_no,pol,
   :zeyl_turu,:zeyl_sira_no,:t_i,:brans_kodu,
   :tanzim_yili,'P',null,:pol_bas_tarih,
   :pol_bit_tarih,:trete_kodu,null,:trete_sirket_meblag,0,
   null,:ay,:yil,trunc(sysdate),
   substr(user,5,12),:tarife,null,:tanzim_tarihi,:yukleme_tarihi,0);
  end if;
   :t_meb := nvl(:t_meb,0) + nvl(:trete_sirket_meblag,0);
  --1.IF 'IN BASI
  if nvl(top2,0) > nvl(:t_meb,0) then
  :kont1:='E';
   if :kont2 ='H' or :kont3 ='S' then :t_meb:=:t_meb;
    else
     :kalan :=nvl(top2,0) - nvl(:t_meb,0);
     :sno:=nvl(:sno,0)+1;
     islem(tar,top1,top2,pol);
```

```
 end if;
end if;
--2.IF'IN BASI
if nvl(top2,0) < nvl(:t_meb,0) then
 :kont2:='H';
if :kont1='E' then :t_meb:=:t_meb;
 else
 if :buldu='E' then :t_meb:=:t_meb;
 else
--BU MEBLAG KONSERVASYONA YEDIRILIR
 meblag := nvl(:t_meb,0) - nvl(top2,0);
  select trete_kod into :trete_kodu from trete_kod_table
     where sirket_kod=:logo.sirket_kod and trete_tip=tip1;
-- ZEYL ISE GECICIYI UPDATE EDER DEGILSE DAGILIMI
 if :zeyl_sira_no = 0 then
  begin
  select /*+ index(reas_dagilim_table reas_dagilim_ndx) */
       trete_sirket_meblag into meb from reas_dagilim_table
    where sirket_kod=:logo.sirket_kod and
       cari_pol_no=:cari_pol_no and
       zeyl_sira_no=:zeyl_sira_no and
       trete_kodu=:trete_kodu and
       brans_kodu=:brans_kodu;
  meb := nvl(meb,0) + nvl(meblag,0);
  exception when no_data_found then msg_no(4,'TRETE KODU PROCE1');
                          bell;
                          raise form_trigger_failure;
  when too_many_rows then msg_no(13,'TRETE KODU PROCE1');
                  bell;
                  raise form_trigger_failure;
  end;
  update reas_dagilim_table set trete_sirket_meblag = meb
   where sirket_kod=:logo.sirket_kod and trete_kodu=:trete_kodu
       and cari_pol_no=:cari_pol_no and zeyl_sira_no=:zeyl_sira_no
       and brans_kodu=:brans_kodu;
 else
  begin
  select trete_sirket_meblag into meb from reas_gecici_table
    where sirket_kod=:logo.sirket_kod and trete_kodu=:trete_kodu
       and cari_pol_no=:cari_pol_no and zeyl_sira_no=:zeyl_sira_no
       and brans_kodu=:brans_kodu;
  meb := nvl(meb,0) + nvl(meblag,0);
  exception when no_data_found then msg_no(4,'trete kodu gecici de');
                          bell;
                          raise form_trigger_failure;
  end;
  update reas_gecici_table set trete_sirket_meblag = meb
   where sirket_kod=:logo.sirket_kod and trete_kodu=:trete_kodu
       and cari_pol_no=:cari_pol_no and zeyl_sira_no=:zeyl_sira_no
       and brans_kodu=:brans_kodu;
end if;
   :global.top1:=top1;
   :global.top2:=top2;
 if :zeyl_sira_no =0 then
   proce2(top1 in number,top2 in number);
   :buldu:='E';
```

```
 end if;
 end if;
 end if;
end if;
--3.IF 'IN BASI
if nvl(top2,0) = nvl(:t_meb,0) then
:kont3:='S';
 if :buldu='E' then :t_meb:=:t_meb;
 else
   :global.top1:=top1;
   :global.top2:=top2;
  if :zeyl_sira_no =0 then
   proce2(top1 in number,top2 in number);
   :buldu:='E';
  end if;
 end if;
end if;
end;
>>>
```

ENDDEFINE PROCEDURE

DEFINE PROCEDURE

```
NAME = proce2
DEFINITION = <<<
procedure proce2(total1 in number,total2 in number) is
sonoran  number;
sayi1    number;
toplam   number;
son_top  number;
tip3     char(15);
tipi     char(15);
s_oran   number;
kodu     char(2);
prim     number;
t_oran   number;
fark     number;
begin
toplam :=0;
t_oran :=0;
declare cursor cr is
 select /*+ index(reas_dagilim_table reas_dagilim_ndx) */
      trete_kodu,trete_sirket_prim,trete_sirket_meblag
  from reas_dagilim_table
  where sirket_kod=:logo.sirket_kod and cari_pol_no =:cari_pol_no
       and zeyl_sira_no=:zeyl_sira_no and
       trete_kodu is not null and
       brans_kodu=:brans_kodu;
 begin
  for kayit in cr
  loop
    if nvl(:aded,0) > 1 then
      :trete_sirket_meblag := kayit.trete_sirket_meblag * :aded;
    else
      :trete_sirket_meblag := kayit.trete_sirket_meblag;
```

```
  end if;
 if nvl(:zeyl_sira_no,0)!=0 then
    sonoran := :trete_sirket_meblag / total2;
    sayi1:= sonoran * total1;
    sayi1:=round(sayi1,:kesir);
 else
    if nvl(:aded,0)>1 then
      sonoran := :trete_sirket_meblag / (total2*:aded);
    else
      sonoran := :trete_sirket_meblag / total2;
    end if;
    sayi1:= sonoran * total1;
    sayi1:=round(sayi1,:kesir);
 end if;
  update reas_dagilim_table
              set trete_sirket_prim = sayi1,
                 trete_sirket_meblag = :trete_sirket_meblag,
                 oran = sonoran
   where sirket_kod=:logo.sirket_kod and trete_kodu=kayit.trete_kodu
      and cari_pol_no=:cari_pol_no and zeyl_sira_no=:zeyl_sira_no
      and brans_kodu=:brans_kodu;
  toplam := nvl(toplam,0) + nvl(sayi1,0);
end loop;
end;
 son_top := nvl(total1,0) - nvl(toplam,0);
   select trete_kod into kodu from trete_kod_table
   where sirket_kod=:logo.sirket_kod and trete_tip=tip3;
   begin
   select /*+ index(reas_dagilim_table reas_dagilim_ndx) */
        nvl(trete_sirket_prim,0) into prim from reas_dagilim_table
   where sirket_kod=:logo.sirket_kod and
        cari_pol_no=:cari_pol_no and
        zeyl_sira_no=:zeyl_sira_no and
        trete_kodu=kodu and
        brans_kodu=:brans_kodu and
        ay=:ay and yil=:yil;
   exception when no_data_found then null;
           when too_many_rows then null;
   end;
   prim :=nvl(prim,0) + nvl(son_top,0);
   prim:=round(prim,:kesir);
   update reas_dagilim_table set trete_sirket_prim = prim
   where sirket_kod=:logo.sirket_kod and trete_kodu=kodu and
        cari_pol_no=:cari_pol_no and zeyl_sira_no=:zeyl_sira_no
        and brans_kodu=:brans_kodu and ay=:ay and yil=:yil;
declare cursor cr is
  select /*+ index(reas_dagilim_table reas_dagilim_ndx) */
        oran from reas_dagilim_table
    where sirket_kod=:logo.sirket_kod and
         cari_pol_no =:cari_pol_no
         and zeyl_sira_no=:zeyl_sira_no and
         trete_kodu is not null and
         brans_kodu=:brans_kodu;
begin
for kayit in cr
 loop
```

```
    t_oran := nvl(t_oran,0) + nvl(kayit.oran,0);
   end loop;
  end;
    fark :=1-nvl(t_oran,0);
    select trete_kod into kodu from trete_kod_table
    where sirket_kod=:logo.sirket_kod and trete_tip=tipi;
    begin
    select /*+ index(reas_dagilim_table reas_dagilim_ndx) */
          oran into s_oran from reas_dagilim_table
    where sirket_kod=:logo.sirket_kod and
         cari_pol_no=:cari_pol_no and
         zeyl_sira_no=:zeyl_sira_no and
         trete_kodu=kodu and
         brans_kodu=:brans_kodu and ay=:ay and yil=:yil;
   exception when no_data_found then null;
           when too_many_rows then null;
   end;
   s_oran :=nvl(s_oran,0) + nvl(fark,0);
   update reas_dagilim_table set oran = s_oran
   where sirket_kod=:logo.sirket_kod and trete_kodu=kodu and
         cari_pol_no=:cari_pol_no and zeyl_sira_no=:zeyl_sira_no
         and brans_kodu=:brans_kodu and ay=:ay and yil=:yil;
  declare cursor son is
   select /*+ index(reas_dagilim_table reas_dagilim_ndx) */
         trete_sirket_meblag,trete_sirket_prim,trete_kodu
    from reas_dagilim_table
 where sirket_kod=:logo.sirket_kod and
      cari_pol_no =:cari_pol_no
      and zeyl_sira_no=:zeyl_sira_no and
      trete_kodu is not null and
      brans_kodu=:brans_kodu;
  begin
  for kayitlar in son
   loop
    if nvl(kayitlar.trete_sirket_meblag,0)=0 and
      nvl(kayitlar.trete_sirket_prim,0)=0  then
    delete reas_dagilim_table
    where sirket_kod=:logo.sirket_kod and
         trete_kodu=kayitlar.trete_kodu and
         cari_pol_no=:cari_pol_no and
         zeyl_sira_no=:zeyl_sira_no and
         brans_kodu=:brans_kodu and
         ay=:ay and yil=:yil;
    end if;
   end loop;
  end;
end;
>>>

ENDDEFINE PROCEDURE

DEFINE PROCEDURE

  NAME = reas_kapali
  DEFINITION = <<<
  procedure reas_kapali is
```

```
kod1 char;
begin
select /*+ index(reas_otomatik_table reas_otomatik_ndx) */
     kod into kod1 from reas_otomatik_table
     where sirket_kod=:logo.sirket_kod and ay=:ay and yil=:yil
         and brans=:brans_kodu and kod='O';
     msg_no(36,'OTOMATIK DAGILIM YAPILMISTIR');
     bell;
     raise form_trigger_failure;
     exception
       when no_data_found then kod1:=kod1;
end;
>>>
```

ENDDEFINE PROCEDURE

DEFINE PROCEDURE

```
NAME = reas_prim
DEFINITION = <<<
procedure reas_prim is
ara_yil  char;
tip      char(15);
bedel    char(1);
prim     char(1);
yil_k    char;
sayi     number;
ist      char;
gon2     number(14);
teklif   number(2);
begin
declare cursor cr is
 select /*+ index(police_teminat_table police_bulten_ndx) */
      sum(tl_net_prim) t1,sum(reas_bedel) t2,sube_kod,cari_pol_no,
      zeyl_sira_no,t_i,tarife,tan_ay,tan_yil,tanzim_tarihi,
      zeyl_turu,bedel_kur,police_cinsi,acenta_no
      from police_teminat_table
      where sirket_kod=:logo.sirket_kod and
          tan_ay=:ay and tan_yil=:yil and
          brans=:brans_kodu
      group by sube_kod,cari_pol_no,zeyl_sira_no,t_i,tarife,
            tan_ay,tan_yil,tanzim_tarihi,
            zeyl_turu,bedel_kur,police_cinsi,
            acenta_no
      order by cari_pol_no,zeyl_sira_no,tanzim_tarihi;
begin
   :son:=1;
   for kayitlar in cr
   loop
    :son:=2;
   :cari_pol_no:=kayitlar.cari_pol_no;
   :acenta_no:=kayitlar.acenta_no;
   :zeyl_turu:=kayitlar.zeyl_turu;
   :zeyl_sira_no:=kayitlar.zeyl_sira_no;
   :t_i:=kayitlar.t_i;
   :tarife:=kayitlar.tarife;
```

```
    :sube:=kayitlar.sube_kod;
    :t_meb:=0;
    :kalan:=0;      ,
    :tanzim_yili:=kayitlar.tan_yil;
    :tanzim_tarihi:=kayitlar.tanzim_tarihi;
    :cins:=kayitlar.police_cinsi;
    :bedel_kur:=kayitlar.bedel_kur;
if pol_kont then
    begin
   select /*+ index(police_table police_ndx) */
        baslama_tarih,bitis_tarih,yukleme_tarih,adet,teklif_onay
        into :pol_bas_tarih,:pol_bit_tarih,:yukleme_tarihi,
        :aded,teklif from police_table
        where sirket_kod=:logo.sirket_kod and
            cari_pol_no=:cari_pol_no and zeyl_sira_no=:zeyl_sira_no;
   exception when no_data_found then msg_no(4,null);
                              bell;
                              raise form_trigger_failure;
   end;
   begin
   select /*+ index(police_table police_ndx) */
         kaynak_kodu into :kaynak from police_table
          where sirket_kod=:logo.sirket_kod and
              cari_pol_no=:cari_pol_no
              and zeyl_sira_no=:zeyl_sira_no and
              acenta_no=:acenta_no ;
   exception when no_data_found then msg_no(4,'KAYNAK KODU');
                              bell;
                              raise form_trigger_failure;
   end;
 if :zeyl_sira_no =0 then
  if nvl(teklif,0)=1 then
   begin
   select reas_bedel,reas_prim into bedel,prim
       from police_cins_table
       where sirket_kod=:logo.sirket_kod and
           pol_cinsi=:cins;
   exception when no_data_found then msg_no(4,'REAS BEDEL,REAS PRIM');
                              bell;
                              raise form_trigger_failure;
   end;
   if prim='E' and bedel='H' then
      select trete_kod into :trete_kodu from trete_kod_table
       where sirket_kod=:logo.sirket_kod and trete_tip=tip;
       :trete_sirket_prim:=round(kayitlar.t1,:kesir);
    if nvl(:trete_sirket_prim,0)!=0 then
    begin
    insert into reas_dagilim_table values (:logo.sirket_kod,
    :sube,:kaynak,:acenta_no,:cari_pol_no,
    :zeyl_turu,:zeyl_sira_no,:t_i,:brans_kodu,
    :tanzim_yili,'P',null,:pol_bas_tarih,
    :pol_bit_tarih,:trete_kodu,null,0,:trete_sirket_prim,null,
    :ay,:yil,trunc(sysdate),
    substr(user,5,12),:tarife,100,:tanzim_tarihi,:yukleme_tarihi,0);
       exception when others then
          insert into cari_dagilim_hata_table values(:cari_pol_no,
```

```
                :zeyl_sira_no);
      end;
    end if;
  end if;
  if prim='E' and bedel='E' then
    :kalan:=kayitlar.t2;
    :sno:=1;
     if nvl(:aded,0) > 1 then
        gon2:=kayitlar.t2/:aded;
        islem(:tarife,kayitlar.t1,gon2,:cari_pol_no);
     else
        islem(:tarife,kayitlar.t1,kayitlar.t2,:cari_pol_no);
     end if;
  end if;
 end if;
else
    if police_dagilim(:cari_pol_no) then
       zeyl(:tarife,kayitlar.t1,kayitlar.t2,:cari_pol_no);
    else
   :kalan:=kayitlar.t2;
   :sno:=1;
     if nvl(kayitlar.t2,0)=0 then
       select trete_kod into :trete_kodu from trete_kod_table
        where sirket_kod=:logo.sirket_kod and trete_tip=tip;
        :trete_sirket_prim:=round(kayitlar.t1,:kesir);
      if nvl(:trete_sirket_prim,0)!=0 then
      begin
       insert into reas_dagilim_table values (:logo.sirket_kod,
       :sube,:kaynak,:acenta_no,:cari_pol_no,
       :zeyl_turu,:zeyl_sira_no,:t_i,:brans_kodu,
       :tanzim_yili,'P',null,:pol_bas_tarih,
       :pol_bit_tarih,:trete_kodu,null,0,:trete_sirket_prim,null,
       :ay,:yil,trunc(sysdate),
       substr(user,5,12),:tarife,100,:tanzim_tarihi,:yukleme_tarihi,0);
        exception when others then
           insert into cari_dagilim_hata_table values(:cari_pol_no,
                :zeyl_sira_no);
      end;
    end if;
      else
          if nvl(:aded,0) > 1 then
            gon2:=kayitlar.t2/:aded;
            islem(:tarife,kayitlar.t1,gon2,:cari_pol_no);
          else
            islem(:tarife,kayitlar.t1,kayitlar.t2,:cari_pol_no);
          end if;
          reas_insert;
      end if;
    end if;
 end if;
commit_form;
:buldu:=null;
:kont1:=null;
:kont2:=null;
:kont3:=null;
else prim:=prim;
```

```
end if;
end loop;
if :brans_kodu is not null then
 if :son=1 then msg_no(4,'POLICE TEMINAT TABLEDA');
           bell;
 end if;
end if;
end;
end;
>>>
```

ENDDEFINE PROCEDURE

DEFINE PROCEDURE

```
NAME = reas_prim2
DEFINITION = <<<
procedure reas_prim2 is
teklif   number(2);
ara_yil  char(4);
tip      char(15);
bedel    char(1);
prim     char(1);
yil_k    char;
sayi     number;
ist      char;
t1       number(14);
t2       number(14);
ay       char(2);
yil      char(4);
gon1     number(15);
gon2     number(15);
begin
 begin
 select /*+ index(police_teminat_table police_bulten_ndx) */
       sum(tl_net_prim),sum(reas_bedel),sube_kod,t_i,tarife,tan_yil,
       tanzim_tarihi,zeyl_turu,bedel_kur,police_cinsi,acenta_no into
       t1,t2,:sube,:t_i,:tarife,:yil,
       :tanzim_tarihi,:zeyl_turu,:bedel_kur,:cins,:acenta_no
       from police_teminat_table
       where sirket_kod=:logo.sirket_kod and
            tan_ay=:ay and tan_yil=:yil and
            brans=:brans_kodu and
            cari_pol_no=:pol_no and
            zeyl_sira_no=:zeyl_sira
       group by sube_kod,cari_pol_no,zeyl_sira_no,t_i,tarife,
             tan_yil,tanzim_tarihi,
             zeyl_turu,bedel_kur,police_cinsi,
             acenta_no;
 exception when no_data_found then message('POLİCE TEM.DE KAYİT YOK');
                           bell;
                           raise form_trigger_failure;
          when too_many_rows then message('POLİCE TEM:DE CİFT KAYİT');
                           bell;
                           raise form_trigger_failure;
 end;
```

```
  :cari_pol_no:=:pol_no;
  :zeyl_sira_no:=:zeyl_sira;
  :t_meb:=null;
  :tanzim_yili:=:yil;
 begin
 select /*+ index(police_table police_ndx) */
      baslama_tarih,bitis_tarih,yukleme_tarih,adet,teklif_onay,
      zeyl_turu
      into :pol_bas_tarih,:pol_bit_tarih,
      :yukleme_tarihi,:aded,teklif,:zeyl_turu
      from police_table
      where sirket_kod=:logo.sirket_kod and
          cari_pol_no=:pol_no and zeyl_sira_no=:zeyl_sira;
  exception when no_data_found then msg_no(4,null);
                        bell;
                        raise form_trigger_failure;
          when too_many_rows then msg_no(13,' CİFT KAYİT
                                -POLICE TABLE');
                        bell;
                        raise form_trigger_failure;
 end;
 begin
 select /*+ index(police_table police_ndx) */
       kaynak_kodu into :kaynak from police_table
        where sirket_kod=:logo.sirket_kod and
            cari_pol_no=:pol_no and
            zeyl_sira_no=:zeyl_sira and
            acenta_no=:acenta_no ;
  exception when no_data_found then msg_no(4,'KAYNAK KODU');
                        bell;
                        raise form_trigger_failure;
          when too_many_rows then msg_no(13,'KAYNAK CİFT KAYİT
                                -POLICE TABLE');
                        bell;
                        raise form_trigger_failure;
 end;
if :zeyl_sira =0 then
-----------------------------------------
--bedel sifirsa tum primi konservasyona verir
-----------------------------------------
 if nvl(t2,0) > 0 then
  if nvl(teklif,0)=1 then
  begin
  select reas_bedel,reas_prim into bedel,prim
     from police_cins_table
     where sirket_kod=:logo.sirket_kod and
         pol_cinsi=:cins;
  exception when no_data_found then msg_no(4,'REAS BEDEL,REAS PRIM');
                        bell;
                        raise form_trigger_failure;
  end;
  if prim='E' and bedel='H' then
     select trete_kod into :trete_kodu from trete_kod_table
      where sirket_kod=:logo.sirket_kod and trete_tip=tip;
      :trete_sirket_prim:=t1;
   if nvl(:trete_sirket_prim,0)!=0 then
```

```
    insert into reas_dagilim_table values (:logo.sirket_kod,
    :sube,null,:acenta_no,:pol_no,
    :zeyl_turu,:zeyl_sira,:t_i,:brans_kodu,
    :tanzim_yili,'P',null,:pol_bas_tarih,
    :pol_bit_tarih,:trete_kodu,null,null,:trete_sirket_prim,null,
    :ay,:yil,trunc(sysdate),
    substr(user,5,12),:tarife,100,:tanzim_tarihi,:yukleme_tarihi,0);
    end if;
   end if;
   if prim='E' and bedel='E' then
    :kalan:=t2;
    :sno:=1;
     if nvl(:aded,0) > 1 then
       gon1:=t1/:aded;
       gon2:=t2/:aded;
       islem(:tarife,gon1,gon2,:cari_pol_no);
     else
       islem(:tarife,t1,t2,:cari_pol_no);
     end if;
   end if;
  end if;
 elsif nvl(t2,0) = 0 then
    select trete_kod into :trete_kodu from trete_kod_table
       where sirket_kod=:logo.sirket_kod and trete_tip=tip;
       :trete_sirket_prim:=t1;
    if nvl(:trete_sirket_prim,0)!=0 then
    insert into reas_dagilim_table values (:logo.sirket_kod,
    :sube,null,:acenta_no,:pol_no,
    :zeyl_turu,:zeyl_sira,:t_i,:brans_kodu,
    :tanzim_yili,'P',null,:pol_bas_tarih,
    :pol_bit_tarih,:trete_kodu,null,null,:trete_sirket_prim,null,
    :ay,:yil,trunc(sysdate),
    substr(user,5,12),:tarife,100,:tanzim_tarihi,:yukleme_tarihi,0);
    end if;
   end if;
  else zeyl(:tarife,t1,t2,:cari_pol_no);
 end if;
 end;
 >>>

ENDDEFINE PROCEDURE

DEFINE PROCEDURE

 NAME = son_kont
 DEFINITION = <<<
 procedure son_kont is
 sayi number;
 begin
 if :pol_no is not null and :zeyl_sira is not null then
 begin
 select /*+ index(reas_dagilim_table reas1_ndx) */
      1 into sayi from reas_dagilim_table
      where sirket_kod=:logo.sirket_kod and
          brans_kodu=:brans_kodu
          and ay=:ay and yil=:yil and
```

```
        cari_pol_no=:pol_no
        and zeyl_sira_no=:zeyl_sira;
      msg_no(13,'CARI POL NO VE ZEYL SIRA NO');
      bell;
      raise form_trigger_failure;
      exception when too_many_rows then msg_no(13,'CARI POL NO VE
                                        ZEYL SIRA NO');
                                bell;
                                raise form_trigger_failure;
      when no_data_found then sayi:=sayi;
  end;
  end if;
  end;
  >>>

ENDDEFINE PROCEDURE

DEFINE PROCEDURE

  NAME = sub
  DEFINITION = <<<
  procedure sub(deger in ist_deger_table.deger_kod%type) is
  begin
  :t_deger:=:t_deger||nvl(substr(deger,1,1),'!');
  :t_deger:=:t_deger||nvl(substr(deger,2,1),'!');
  :t_deger:=:t_deger||nvl(substr(deger,3,1),'!');
  end;
  >>>

ENDDEFINE PROCEDURE

DEFINE PROCEDURE

  NAME = ZEYL
  DEFINITION = <<<
  procedure ZEYL (tar in tarife_table.tarife_kod%type,
              top1 in number,
              top2 in number,
              pol in police_teminat_table.cari_pol_no%type) is
  flag    number;
  say     number;
  zk      char(3);
  bul     number;
  mebl    number;
  t2      number;
  cari    number;
  son_top number;
  bk      char(3);
  fark_bedel number;
  begin
   flag:=0;
   begin
     select /*+ index(zeyl_header_table zeyl_header1_ndx) */
          zeyl_kodu  into zk from zeyl_header_table
       where sirket_kod=:logo.sirket_kod
            and t_i='B' and reas_giris='E';
```

```
  exception when no_data_found then msg_no(4,null);
                                bell;
                                raise form_trigger_failure;
end;
begin
select /*+ index(reas_dagilim_table reas_dagilim_ndx) */
      cari_pol_no,brans_kodu,
sum(decode(t_i,'T', trete_sirket_meblag,-1*trete_sirket_meblag))
   into cari,bk,t2
     from reas_dagilim_table
         where sirket_kod=:logo.sirket_kod and
              cari_pol_no=:cari_pol_no and
              zeyl_sira_no is not null and
              trete_kodu is not null and
              brans_kodu=:brans_kodu
              and tanzim_tarihi <= :tanzim_tarihi
    group by cari_pol_no,brans_kodu;
 exception when no_data_found then t2:=0 ;
 end;
 :t_meb:=null;
if top2 > 0 then
  if zeyl_var(zk) then
       if :t_i ='T' then
          son_top := nvl(top2,0) + nvl(t2,0);
          else
          son_top := nvl(t2,0) - nvl(top2,0);
       end if;
  if nvl(:aded,0) > 1 then
   son_top:=son_top/:aded;
  end if;
--BU SON TOP DAGITILIR,GECICI TABLEDA SAKLANIR
    if nvl(son_top,0) !=0 then
       sonislem(tar,top1,son_top,pol);
    end if;
    declare cursor cr1 is
     select /*+ index(reas_dagilim_table reas_dagilim_ndx) */
          trete_kodu,
      sum(decode(t_i,'T', trete_sirket_meblag,-1*trete_sirket_meblag))
       tt2 from reas_dagilim_table
          where sirket_kod=:logo.sirket_kod and
              cari_pol_no=:cari_pol_no and
              zeyl_sira_no is not null and
              trete_kodu is not null and
              brans_kodu=:brans_kodu
              and tanzim_tarihi <= :tanzim_tarihi
    group by trete_kodu;
    begin
      for kayit in cr1
      loop
      begin
      select trete_sirket_meblag into mebl from reas_gecici_table
       where sirket_kod=:logo.sirket_kod and cari_pol_no=:cari_pol_no
        and brans_kodu=:brans_kodu and trete_kodu=kayit.trete_kodu
        and zeyl_sira_no=:zeyl_sira_no;
       exception when no_data_found then mebl:=0;
       end;
```

```
if flag=0 then
    if nvl(:aded,0) > 1 then
      kayit.tt2 :=kayit.tt2/:aded;
    end if;
    if :t_i = 'T' then
     fark_bedel := nvl(mebl,0) - nvl(kayit.tt2,0);
     else
     fark_bedel := nvl(kayit.tt2,0) - nvl(mebl,0);
    end if;
  if nvl(fark_bedel,0) > 0 then
    delete reas_gecici_table
    where sirket_kod=:logo.sirket_kod and cari_pol_no=:cari_pol_no
      and brans_kodu=:brans_kodu and trete_kodu=kayit.trete_kodu
      and zeyl_sira_no=:zeyl_sira_no;
    begin
    insert into reas_dagilim_table values (:logo.sirket_kod,
              :sube,:kaynak,:acenta_no,pol,
              :zeyl_turu,:zeyl_sira_no,:t_i,:brans_kodu,
              :tanzim_yili,'P',
              null,:pol_bas_tarih,
              :pol_bit_tarih,kayit.trete_kodu,null,
              fark_bedel,null,
              null,:ay,:yil,trunc(sysdate),
              substr(user,5,12),:tarife,null,
              :tanzim_tarihi,:yukleme_tarihi,0);
    exception when others then
      insert into cari_dagilim_hata_table values(pol,
          :zeyl_sira_no);
   end;
   elsif nvl(fark_bedel,0) < 0 then
    delete reas_dagilim_table where sirket_kod=:logo.sirket_kod and
        cari_pol_no=pol and zeyl_sira_no=:zeyl_sira_no and
        brans_kodu=:brans_kodu and tanzim_tarihi=:tanzim_tarihi;
    begin
    insert into reas_dagilim_table values (:logo.sirket_kod,
              :sube,:kaynak,:acenta_no,pol,
              :zeyl_turu,:zeyl_sira_no,:t_i,:brans_kodu,
              :tanzim_yili,'P',
              null,:pol_bas_tarih,
              :pol_bit_tarih,'01',null,
              top2,null,
              null,:ay,:yil,trunc(sysdate),
              substr(user,5,12),:tarife,null,
              :tanzim_tarihi,:yukleme_tarihi,0);
    flag:=1;
    delete reas_gecici_table;
    exception when others then null;
   end;
  end if;
 end if;
  end loop;
 end;
else
  re_zeyl(top1,top2,zk,pol);
end if;
end if;
```

```
if nvl(top2,0) = 0 then
 if zeyl_var(zk) then
    declare cursor cr1 is
      select /*+ index(reas_dagilim_table reas_dagilim_ndx) */
           trete_kodu,sum(DECODE(T_I,'T',trete_sirket_meblag,
                    -TRETE_SIRKET_MEBLAG)) tt1
        from reas_dagilim_table
        where sirket_kod=:logo.sirket_kod and
            cari_pol_no=:cari_pol_no and
            zeyl_sira_no is not null and
            trete_kodu is not null and
            brans_kodu=:brans_kodu
            and tanzim_tarihi <= :tanzim_tarihi
       group by trete_kodu;
    begin
      for kayit in cr1
      loop
         :oran:=kayit.tt1/t2;
         :trete_sirket_prim := :oran * top1;
         :trete_sirket_prim:=round(:trete_sirket_prim,:kesir);
       if :trete_sirket_prim is not null then
       begin
       insert into reas_dagilim_table values (:logo.sirket_kod,
            :sube,:kaynak,:acenta_no,pol,
            :zeyl_turu,:zeyl_sira_no,:t_i,:brans_kodu,
            :tanzim_yili,'P',null,
            :pol_bas_tarih,:pol_bit_tarih,kayit.trete_kodu,null,
            null,:trete_sirket_prim,
            null,:ay,:yil,trunc(sysdate),
            substr(user,5,12),:tarife,:oran,:tanzim_tarihi,
            :yukleme_tarihi,0);
       exception when others then
          insert into cari_dagilim_hata_table values(pol,
             :zeyl_sira_no);
     end;
       end if;
      end loop;
      end;
 else
  re_zeyl(top1,top2,zk,pol);
 end if;
end if;
if nvl(top2,0)>0 then
  begin
  select count(*) into say from reas_gecici_table
     where sirket_kod=:logo.sirket_kod and cari_pol_no=:cari_pol_no
        and brans_kodu=:brans_kodu;
  exception when no_data_found then say:=0;
  end;
  if nvl(say,0)>0 then
  begin
  insert into reas_dagilim_table
  (select * from reas_gecici_table);
   exception when others then
         insert into cari_dagilim_hata_table values(0,0);
  end;
```

```
        delete reas_gecici_table
        where sirket_kod=:logo.sirket_kod and cari_pol_no=:cari_pol_no
          and brans_kodu=:brans_kodu;
    end if;
    bedel_kont(top2);
    proce2(top1,top2);
  end if;
   commit_form;
  end;
  >>>

ENDDEFINE PROCEDURE

DEFINE PROCEDURE

  NAME = zeyl_varlik
  DEFINITION = <<<
  procedure zeyl_varlik is
  sayi number;
  begin
  if :zeyl_sira is not null then
     begin
     select /*+ index(police_teminat_table police_teminat_ndx) */
          1 into sayi from police_teminat_table
       where sirket_kod=:logo.sirket_kod and cari_pol_no=:pol_no
           and zeyl_sira_no=:zeyl_sira;
     exception when no_data_found then msg_no(4,'ZEYL SIRA NO');
                                bell;
                                raise form_trigger_failure;
             when too_many_rows then sayi:=0;
     end;
  end if;
  end;
  >>>

ENDDEFINE PROCEDURE

DEFINE TRIGGER

  NAME = PRE-FORM
  TRIGGER_TYPE = V3
  TEXT = <<<
  basla(:logo.sirket_kod,:logo.sirket_adi,:logo.sube_kod,:logo.sube_adi,
       :logo.bugun);
  :s1:='/';
  select doviz into :dov from sirket_tanim_table
    where sirket_kod=:logo.sirket_kod;
  select kesir into :kesir from doviz_tanim_table
     where doviz=:dov;
  >>>

ENDDEFINE TRIGGER

DEFINE TRIGGER

  NAME = KEY-EXIT
```

```
  TRIGGER_TYPE = V3
  TEXT = <<<
  clear_form(no_validate,full_rollback);
  execute_trigger('pre-form');
  go_field('gds');
  >>>

ENDDEFINE TRIGGER

DEFINE TRIGGER

  NAME = KEY-OTHERS
  TRIGGER_TYPE = V3
  TEXT = <<<
  bell;
  msg_no(6,null);
  >>>

ENDDEFINE TRIGGER

DEFINE BLOCK

  NAME = logo
  ROWS_DISPLAYED = 1
  ROWS_BUFFERED = 1
  BASE_LINE = 1
  LINES_PER_ROW = 1
  ARRAY_SIZE = 0

  DEFINE FIELD

    NAME = sirket_adi
    DATATYPE = CHAR
    LENGTH = 20
    DISPLAY_LENGTH = 20
    QUERY_LENGTH = 20
    BASE_TABLE = OFF
    PAGE = 1
    LINE = 1
    COLUMN = 1
    INPUT = OFF
    UPDATE = OFF
    QUERY = OFF
    UPPERCASE = ON

  ENDDEFINE FIELD

  DEFINE FIELD

    NAME = bugun
    DATATYPE = CHAR
    LENGTH = 10
    DISPLAY_LENGTH = 10
    QUERY_LENGTH = 10
    BASE_TABLE = OFF
    PAGE = 1
```

```
    LINE = 1
    COLUMN = 71
    INPUT = OFF
    UPDATE = OFF
    QUERY = OFF

  ENDDEFINE FIELD

  DEFINE FIELD

    NAME = sirket_kod
    DATATYPE = NUMBER
    LENGTH = 3
    DISPLAY_LENGTH = 3
    QUERY_LENGTH = 3
    BASE_TABLE = OFF
    DISPLAYED = OFF
    PAGE = 0
    LINE = 0
    COLUMN = 0
    ECHO = OFF
    INPUT = OFF
    UPDATE = OFF
    QUERY = OFF

  ENDDEFINE FIELD

  DEFINE FIELD

    NAME = sube_kod
    DATATYPE = NUMBER
    LENGTH = 4
    DISPLAY_LENGTH = 4
    QUERY_LENGTH = 4
    BASE_TABLE = OFF
    DISPLAYED = OFF
    PAGE = 0
    LINE = 0
    COLUMN = 0
    ECHO = OFF
    INPUT = OFF
    UPDATE = OFF
    QUERY = OFF

  ENDDEFINE FIELD

  DEFINE FIELD

    NAME = gds
    DATATYPE = CHAR
    LENGTH = 1
    DISPLAY_LENGTH = 1
    QUERY_LENGTH = 1
    BASE_TABLE = OFF
    PAGE = 1
    LINE = 18
```

```
COLUMN = 10
UPPERCASE = ON

DEFINE TRIGGER

  NAME = KEY-EXIT
  TRIGGER_TYPE = V3
  TEXT = <<<
  exit_form;
  >>>

ENDDEFINE TRIGGER

DEFINE TRIGGER

  NAME = KEY-NXTFLD
  TRIGGER_TYPE = V3
  TEXT = <<<
  if :gds is null then msg_no(10,null);
                bell;
                raise form_trigger_failure;
  end if;
  if :gds not in ('G','S') then msg_no(36,'G VEYA S GIRINIZ');
                          bell;
                          raise form_trigger_failure;
  end if;
  :sil:=null;
  go_block('blok1');
  clear_block(no_commit);
  :s1:='/';
  go_field('brans_kodu');
  >>>

ENDDEFINE TRIGGER

ENDDEFINE FIELD

DEFINE FIELD

  NAME = sil
  DATATYPE = CHAR
  LENGTH = 1
  DISPLAY_LENGTH = 1
  QUERY_LENGTH = 1
  BASE_TABLE = OFF
  PAGE = 1
  LINE = 18
  COLUMN = 79
  QUERY = OFF
  UPPERCASE = ON

  DEFINE TRIGGER

    NAME = KEY-NXTFLD
    TRIGGER_TYPE = V3
    TEXT = <<<
```

```
if :sil is null then msg_no(10,null);
                bell;
                raise form_trigger_failure;
end if;
if :sil not in ('E','H') then msg_no(36,'E VEYA H GIRINIZ');
                        bell;
                        raise form_trigger_failure;
end if;
if :sil='E' then
 if :pol_no is null and :zeyl_sira is null then
  delete reas_dagilim_table
     where sirket_kod=:logo.sirket_kod and
         brans_kodu=:brans_kodu and
         ay=:ay and yil=:yil;
  delete reas_otomatik_table
     where sirket_kod=:logo.sirket_kod and
         brans=:brans_kodu and
         ay=:ay and yil=:yil and kod='O';
  else
    delete reas_dagilim_table
      where sirket_kod=:logo.sirket_kod and
          brans_kodu=:brans_kodu and
          ay=:ay and yil=:yil and
          cari_pol_no=:pol_no and
          zeyl_sira_no=:zeyl_sira;
 end if;
 commit_form;
 if form_success then msg_no(11,null);
            else msg_no(12,null);
 end if;
 bell;
end if;
go_block('blok1');
clear_block(no_commit);
:sil:=null;
go_field('gds');
raise form_trigger_failure;
>>>

ENDDEFINE TRIGGER

ENDDEFINE FIELD

DEFINE FIELD

NAME = buldu
DATATYPE = CHAR
LENGTH = 1
DISPLAY_LENGTH = 1
QUERY_LENGTH = 1
BASE_TABLE = OFF
DISPLAYED = OFF
PAGE = 0
LINE = 0
COLUMN = 0
ECHO = OFF
```

```
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = var
  DATATYPE = NUMBER
  LENGTH = 1
  DISPLAY_LENGTH = 1
  QUERY_LENGTH = 1
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = kont1
  DATATYPE = CHAR
  LENGTH = 1
  DISPLAY_LENGTH = 1
  QUERY_LENGTH = 1
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = kont2
  DATATYPE = CHAR
  LENGTH = 1
  DISPLAY_LENGTH = 1
  QUERY_LENGTH = 1
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
```

```
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = kont3
  DATATYPE = CHAR
  LENGTH = 1
  DISPLAY_LENGTH = 1
  QUERY_LENGTH = 1
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = son
  DATATYPE = NUMBER
  LENGTH = 14
  DISPLAY_LENGTH = 14
  QUERY_LENGTH = 14
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = kontrol
  DATATYPE = NUMBER
  LENGTH = 1
  DISPLAY_LENGTH = 1
  QUERY_LENGTH = 1
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
```

```
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = kesir
  DATATYPE = NUMBER
  LENGTH = 2
  DISPLAY_LENGTH = 2
  QUERY_LENGTH = 2
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = bedel_kur
  DATATYPE = NUMBER
  LENGTH = 9
  DISPLAY_LENGTH = 9
  QUERY_LENGTH = 9
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = sube_adi
  DATATYPE = CHAR
  LENGTH = 20
  DISPLAY_LENGTH = 20
  QUERY_LENGTH = 20
  BASE_TABLE = OFF
  PAGE = 1
  LINE = 2
```

```
    COLUMN = 1
    INPUT = OFF
    UPDATE = OFF
    QUERY = OFF
    UPPERCASE = ON

  ENDDEFINE FIELD

  DEFINE FIELD

    NAME = aded
    DATATYPE = NUMBER
    LENGTH = 4
    DISPLAY_LENGTH = 4
    QUERY_LENGTH = 4
    BASE_TABLE = OFF
    DISPLAYED = OFF
    PAGE = 0
    LINE = 0
    COLUMN = 0
    ECHO = OFF
    INPUT = OFF
    UPDATE = OFF
    QUERY = OFF

  ENDDEFINE FIELD

ENDDEFINE BLOCK

DEFINE BLOCK

  NAME = blok1
  ROWS_DISPLAYED = 1
  ROWS_BUFFERED = 1
  BASE_LINE = 1
  LINES_PER_ROW = 1
  ARRAY_SIZE = 0

  DEFINE TRIGGER

    NAME = KEY-COMMIT
    TRIGGER_TYPE = V3
    TEXT = <<<
    if :gds='S' then
            msg_no(6,null);
            bell;
            raise form_trigger_failure;
    end if;
    genel_kont;
    reas_kapali;
    if :pol_no is null and :zeyl_sira is null then
      pro_kont;
      if :gds = 'G' then
        reas_prim;
        insert into reas_otomatik_table values(1,:ay,:yil,:brans_kodu,'O',
                              trunc(sysdate),substr(user,5,12));
```

```
    end if;
  else
   if :pol_no is not null and :zeyl_sira is null then
     msg_no(3,'ZEYL SIRA NO');
     bell;
     go_field('zeyl_sira');
     raise form_trigger_failure;
   end if;
   if :gds = 'G' then
     pol_varlik;
     zeyl_varlik;
     son_kont;
     reas_prim2;
   end if;
  end if;
  commit_form;
  if form_success then msg_no(11,null);
              else msg_no(12,null);
  end if;
  bell;
  go_field('gds');
  raise form_trigger_failure;
  >>>

ENDDEFINE TRIGGER

DEFINE TRIGGER

  NAME = KEY-LISTVAL
  TRIGGER_TYPE = V3
  TEXT = <<<
  list_values;
  >>>

ENDDEFINE TRIGGER

DEFINE FIELD

  NAME = brans_kodu
  DATATYPE = CHAR
  LENGTH = 3
  DISPLAY_LENGTH = 3
  QUERY_LENGTH = 3
  BASE_TABLE = OFF
  PAGE = 1
  LINE = 7
  COLUMN = 26
  LOV_TEXT = <<<
  select brans_kod,brans_adi into :brans_kodu,:brans_adi
  from brans_table where sirket_kod=:logo.sirket_kod
  >>>
  LOV_TITLE = BRANS KOD VE ADLARI
  UPPERCASE = ON

  DEFINE TRIGGER
```

```
    NAME = KEY-NXTFLD
    TRIGGER_TYPE = V3
    TEXT = <<<
    if :brans_kodu is null then msg_no(10,null);
                    bell;
                    raise form_trigger_failure;
    end if;
    brans_kont;
    go_field('ay');
    >>>

  ENDDEFINE TRIGGER

ENDDEFINE FIELD

DEFINE FIELD

  NAME = brans_adi
  DATATYPE = CHAR
  LENGTH = 15
  DISPLAY_LENGTH = 15
  QUERY_LENGTH = 15
  BASE_TABLE = OFF
  PAGE = 1
  LINE = 7
  COLUMN = 33
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF
  UPPERCASE = ON

ENDDEFINE FIELD

DEFINE FIELD

  NAME = ay
  DATATYPE = CHAR
  LENGTH = 2
  DISPLAY_LENGTH = 2
  QUERY_LENGTH = 2
  BASE_TABLE = OFF
  PAGE = 1
  LINE = 9
  COLUMN = 26
  QUERY = OFF
  AUTOSKIP = ON
  UPPERCASE = ON

  DEFINE TRIGGER

    NAME = KEY-NXTFLD
    TRIGGER_TYPE = V3
    TEXT = <<<
    if :ay is null then msg_no(10,null);
                    bell;
                    raise form_trigger_failure;
```

```
  end if;
  declare month date;
  begin
   month:=to_date(:ay,'mm');
   exception
    when others then msg_no(26,null);
                     bell;
                     raise form_trigger_failure;
  end;
  :ay :=lpad(rtrim(ltrim(:ay,' '),' '),2,'0');
  next_field;
  >>>

 ENDDEFINE TRIGGER

 DEFINE TRIGGER

  NAME = KEY-PRVFLD
  TRIGGER_TYPE = V3
  TEXT = <<<
  go_field('brans_kodu');
  >>>

 ENDDEFINE TRIGGER

ENDDEFINE FIELD

DEFINE FIELD

 NAME = s1
 DATATYPE = CHAR
 LENGTH = 1
 DISPLAY_LENGTH = 1
 QUERY_LENGTH = 1
 BASE_TABLE = OFF
 PAGE = 1
 LINE = 9
 COLUMN = 28
 INPUT = OFF
 UPDATE = OFF
 QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

 NAME = yil
 DATATYPE = CHAR
 LENGTH = 4
 DISPLAY_LENGTH = 4
 QUERY_LENGTH = 4
 BASE_TABLE = OFF
 PAGE = 1
 LINE = 9
 COLUMN = 29
 QUERY = OFF
```

```
DEFINE TRIGGER

  NAME = KEY-NXTFLD
  TRIGGER_TYPE = V3
  TEXT = <<<
  if :yil is null then msg_no(10,null);
                  bell;
                  raise form_trigger_failure;
  end if;
  declare year date;
  begin
   year:=to_date(:yil,'yyyy');
   exception
     when others then msg_no(26,null);
                  bell;
                  raise form_trigger_failure;
  end;
  :ay :=lpad(rtrim(ltrim(:ay,' '),' '),2,'0');
  /*
  declare s_yil number;
        ilk_yil number;
  begin
  s_yil:=to_number(to_char(sysdate,'yyyy')) + 1;
  ilk_yil :=s_yil-2;
  if to_number(:yil)>s_yil or to_number(:yil)<ilk_yil
    then msg_no(26,' YIL '||to_char(ilk_yil)||' ILE '||
                to_char(s_yil)|| ' ARASINDA OLMALIDIR');
    bell;
    raise form_trigger_failure;
  end if;
  end;
  */
  if :gds='G' then
    reas_kapali;
  end if;
  next_field;
  >>>

ENDDEFINE TRIGGER

DEFINE TRIGGER

  NAME = KEY-PRVFLD
  TRIGGER_TYPE = V3
  TEXT = <<<
  previous_field;
  >>>

ENDDEFINE TRIGGER

ENDDEFINE FIELD

DEFINE FIELD

  NAME = pol_no
  DATATYPE = NUMBER
```

```
LENGTH = 7
DISPLAY_LENGTH = 7
QUERY_LENGTH = 7
BASE_TABLE = OFF
PAGE = 1
LINE = 11
COLUMN = 26
QUERY = OFF

DEFINE TRIGGER

  NAME = KEY-NXTFLD
  TRIGGER_TYPE = V3
  TEXT = <<<
  if :gds ='S' and :pol_no is null then
    declare sayi number;
    begin
    select 1 into sayi from reas_dagilim_table
     where sirket_kod=:logo.sirket_kod and brans_kodu=:brans_kodu
         and ay=:ay and yil=:yil;
      exception when no_data_found then msg_no(4,null);
                                        bell;
                                        raise form_trigger_failure;
                when too_many_rows then sayi:=0;
    end;
    msg_no(8,null);
    go_field('sil');
  end if;
  pol_varlik;
  if :gds='G' then
   reas_kapali;
  end if;
  if :pol_no is not null then
    next_field;
  end if;
  >>>

ENDDEFINE TRIGGER

DEFINE TRIGGER

  NAME = KEY-PRVFLD
  TRIGGER_TYPE = V3
  TEXT = <<<
  pol_varlik;
  previous_field;
  >>>

ENDDEFINE TRIGGER

ENDDEFINE FIELD

DEFINE FIELD

  NAME = zeyl_sira
  DATATYPE = NUMBER
```

```
    LENGTH = 3
    DISPLAY_LENGTH = 3
    QUERY_LENGTH = 3
    BASE_TABLE = OFF
    PAGE = 1
    LINE = 11
    COLUMN = 50
    QUERY = OFF

    DEFINE TRIGGER

      NAME = KEY-NXTFLD
      TRIGGER_TYPE = V3
      TEXT = <<<
      zeyl_varlik;
      if :gds='G' then
        son_kont;
      end if;
      if :gds ='S' then
        declare sayi number;
        begin
        select 1 into sayi from reas_dagilim_table
         where sirket_kod=:logo.sirket_kod and brans_kodu=:brans_kodu
             and ay=:ay and yil=:yil and cari_pol_no=:pol_no
             and zeyl_sira_no=:zeyl_sira;
           exception when no_data_found then msg_no(4,null);
                                        bell;
                                        go_field('brans_kodu');
                                    raise form_trigger_failure;
                    when too_many_rows then sayi:=1;
        end;
        msg_no(8,null);
        go_field('sil');
      end if;
      >>>

    ENDDEFINE TRIGGER

    DEFINE TRIGGER

      NAME = KEY-PRVFLD
      TRIGGER_TYPE = V3
      TEXT = <<<
      zeyl_varlik;
      previous_field;
      >>>

    ENDDEFINE TRIGGER

  ENDDEFINE FIELD

ENDDEFINE BLOCK

DEFINE BLOCK

  NAME = blok2
```

```
ROWS_DISPLAYED = 1
ROWS_BUFFERED = 1
BASE_LINE = 1
LINES_PER_ROW = 1
ARRAY_SIZE = 0

DEFINE FIELD

  NAME = kaynak
  DATATYPE = NUMBER
  LENGTH = 3
  DISPLAY_LENGTH = 3
  QUERY_LENGTH = 3
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  EDIT_WORD_WRAP = OFF
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = sube
  DATATYPE = NUMBER
  LENGTH = 4
  DISPLAY_LENGTH = 4
  QUERY_LENGTH = 4
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = acenta_no
  DATATYPE = CHAR
  LENGTH = 7
  DISPLAY_LENGTH = 7
  QUERY_LENGTH = 7
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
```

```
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = cari_pol_no
  DATATYPE = NUMBER
  LENGTH = 7
  DISPLAY_LENGTH = 7
  QUERY_LENGTH = 7
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = zeyl_turu
  DATATYPE = CHAR
  LENGTH = 3
  DISPLAY_LENGTH = 3
  QUERY_LENGTH = 3
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = zeyl_sira_no
  DATATYPE = NUMBER
  LENGTH = 3
  DISPLAY_LENGTH = 3
  QUERY_LENGTH = 3
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
```

```
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = t_i
  DATATYPE = CHAR
  LENGTH = 1
  DISPLAY_LENGTH = 1
  QUERY_LENGTH = 1
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = tanzim_yili
  DATATYPE = CHAR
  LENGTH = 4
  DISPLAY_LENGTH = 4
  QUERY_LENGTH = 4
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = bilgi_turu
  DATATYPE = CHAR
  LENGTH = 1
  DISPLAY_LENGTH = 1
  QUERY_LENGTH = 1
  BASE_TABLE = OFF
  DISPLAYED = OFF
```

```
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = yi_yd
  DATATYPE = CHAR
  LENGTH = 2
  DISPLAY_LENGTH = 2
  QUERY_LENGTH = 2
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = pol_bas_tarih
  DATATYPE = DATE
  LENGTH = 10
  DISPLAY_LENGTH = 10
  QUERY_LENGTH = 10
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = pol_bit_tarih
  DATATYPE = DATE
  LENGTH = 10
  DISPLAY_LENGTH = 10
  QUERY_LENGTH = 10
  BASE_TABLE = OFF
```

```
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = trete_kodu
  DATATYPE = CHAR
  LENGTH = 2
  DISPLAY_LENGTH = 2
  QUERY_LENGTH = 2
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = trete_sirket_kodu
  DATATYPE = NUMBER
  LENGTH = 3
  DISPLAY_LENGTH = 3
  QUERY_LENGTH = 3
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = trete_sirket_meblag
  DATATYPE = NUMBER
  LENGTH = 14
  DISPLAY_LENGTH = 14
  QUERY_LENGTH = 14
```

```
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = trete_sirket_prim
  DATATYPE = NUMBER
  LENGTH = 14
  DISPLAY_LENGTH = 14
  QUERY_LENGTH = 14
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = aciklama
  DATATYPE = CHAR
  LENGTH = 40
  DISPLAY_LENGTH = 40
  QUERY_LENGTH = 40
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = tarife
  DATATYPE = CHAR
  LENGTH = 3
  DISPLAY_LENGTH = 3
```

```
  QUERY_LENGTH = 3
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = sno
  DATATYPE = NUMBER
  LENGTH = 1
  DISPLAY_LENGTH = 1
  QUERY_LENGTH = 1
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = tanzim_tarihi
  DATATYPE = DATE
  LENGTH = 10
  DISPLAY_LENGTH = 10
  QUERY_LENGTH = 10
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = yukleme_tarihi
  DATATYPE = DATE
  LENGTH = 10
```

```
  DISPLAY_LENGTH = 10
  QUERY_LENGTH = 10
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = kalan
  DATATYPE = NUMBER
  LENGTH = 14
  DISPLAY_LENGTH = 14
  QUERY_LENGTH = 14
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = oran
  DATATYPE = NUMBER
  LENGTH = 10
  DISPLAY_LENGTH = 10
  QUERY_LENGTH = 10
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = plen
  DATATYPE = NUMBER
```

```
  LENGTH = 3
  DISPLAY_LENGTH = 3
  QUERY_LENGTH = 3
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = bedel
  DATATYPE = NUMBER
  LENGTH = 16
  DISPLAY_LENGTH = 16
  QUERY_LENGTH = 16
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = max_bedel
  DATATYPE = NUMBER
  LENGTH = 16
  DISPLAY_LENGTH = 16
  QUERY_LENGTH = 16
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = b_k
```

```
  DATATYPE = CHAR
  LENGTH = 1
  DISPLAY_LENGTH = 1
  QUERY_LENGTH = 1
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = matrah
  DATATYPE = NUMBER
  LENGTH = 5
  DISPLAY_LENGTH = 5
  QUERY_LENGTH = 5
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = asg_bedel
  DATATYPE = NUMBER
  LENGTH = 16
  DISPLAY_LENGTH = 16
  QUERY_LENGTH = 16
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD
```

```
  NAME = t_deger
  DATATYPE = CHAR
  LENGTH = 90
  DISPLAY_LENGTH = 90
  QUERY_LENGTH = 90
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = t_meb
  DATATYPE = NUMBER
  LENGTH = 14
  DISPLAY_LENGTH = 14
  QUERY_LENGTH = 14
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = dov
  DATATYPE = CHAR
  LENGTH = 3
  DISPLAY_LENGTH = 3
  QUERY_LENGTH = 3
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD
```

```
  NAME = kur
  DATATYPE = DATE
  LENGTH = 10
  DISPLAY_LENGTH = 10
  QUERY_LENGTH = 10
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = son_meblag
  DATATYPE = NUMBER
  LENGTH = 14
  DISPLAY_LENGTH = 14
  QUERY_LENGTH = 14
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD

  NAME = cins
  DATATYPE = CHAR
  LENGTH = 1
  DISPLAY_LENGTH = 1
  QUERY_LENGTH = 1
  BASE_TABLE = OFF
  DISPLAYED = OFF
  PAGE = 0
  LINE = 0
  COLUMN = 0
  ECHO = OFF
  INPUT = OFF
  UPDATE = OFF
  QUERY = OFF

ENDDEFINE FIELD

DEFINE FIELD
```

```
    NAME = sayi
    DATATYPE = NUMBER
    LENGTH = 1
    DISPLAY_LENGTH = 1
    QUERY_LENGTH = 1
    BASE_TABLE = OFF
    DISPLAYED = OFF
    PAGE = 0
    LINE = 0
    COLUMN = 0
    ECHO = OFF
    INPUT = OFF
    UPDATE = OFF
    QUERY = OFF

  ENDDEFINE FIELD

ENDDEFINE BLOCK


DEFINE SCREEN

  DEFINE PAGE

    PAGE = 1
    PAGE_XS = 0
    PAGE_YS = 0
    PAGE_PX0 = 0
    PAGE_PY0 = 0
    PAGE_PXS = 0
    PAGE_PYS = 0
    PAGE_SX0 = 0
    PAGE_SY0 = 0
    MODE = TEXT
    LINE = 5
    BOILER = <<<
                  REASURANS OTOMATIK DAGILIM
    >>>
    LINE = 7
    BOILER = <<<
            BRANS KODU :
    >>>
    LINE = 9
    BOILER = <<<
            AY - YIL   :
    >>>
    LINE = 11
    BOILER = <<<
            POLICE NO  :        ZEYL SIRA NO :
    >>>
    LINE = 18
    BOILER = <<<
     G / S :
    >>>

  ENDDEFINE PAGE
```

```
ENDDEFINE SCREEN

ENDDEFINE FORM
```

```
/* Copyright (c) 1988 by the Oracle Corporation */

SQL*FORMS_VERSION = 03.00.16.12.02
TERSE = ON

DEFINE FORM

  NAME = rekapama
  TITLE = rekapama
  DEFAULT_MENU_APPLICATION = DEFAULT

  DEFINE PROCEDURE

    NAME = basla

    DEFINE REFERENCE

      OWNER = sg
      APPLICATION = genproc
      PROCEDURE = basla

    ENDDEFINE REFERENCE

  ENDDEFINE PROCEDURE

  DEFINE PROCEDURE

    NAME = brans_kont
    DEFINITION = <<<
    procedure brans_kont is
    begin
    declare sayi number;
    begin
    if :brans is not null then
    begin
    select /*+ index(brans_table brans_ndx) */
          1 into sayi from brans_table
          where sirket_kod=:logo.sirket_kod and brans_kod=:brans;
    exception
        when no_data_found then msg_no(4,'BRANS KODU');
                          bell;
                          raise form_trigger_failure;
    end;
    select /*+ index(brans_table brans_ndx) */
          brans_adi into :brans_adi from brans_table
          where sirket_kod=:logo.sirket_kod and brans_kod=:brans;
    end if;
    end;
    end;
    >>>

  ENDDEFINE PROCEDURE

  DEFINE PROCEDURE
```

```
NAME = genel_kont
DEFINITION = <<<
procedure genel_kont is
begin
if :brans is null then msg_no(3,'BRANS KODU');
                  bell;
                  raise form_trigger_failure;
end if;
brans_kont;
if :ay is null then msg_no(3,'AY');
                bell;
                raise form_trigger_failure;
end if;
declare month date;
begin
 month:=to_date(:ay,'mm');
 exception
  when others then msg_no(26,null);
                bell;
                raise form_trigger_failure;
end;
:ay :=lpad(rtrim(ltrim(:ay,' '),' '),2,'0');
if :yil is null then msg_no(3,'YIL');
                bell;
                raise form_trigger_failure;
end if;
declare year date;
begin
 year:=to_date(:yil,'yyyy');
 exception
  when others then msg_no(26,null);
                bell;
                raise form_trigger_failure;
end;
end;
>>>
```

ENDDEFINE PROCEDURE

DEFINE PROCEDURE

```
NAME = insert_et
DEFINITION = <<<
procedure insert_et is
d_i        char(1);
cursor cr is
select /*+ index(reas_dagilim_table reas1_ndx) */
     sube,kaynak,acenta_no,t_i,bilgi_turu,
     trete_kodu,trete_sirket_kodu,ay,yil,
     brans_kodu,sum(trete_sirket_prim) t1,
     sum(trete_sirket_kom) t2
  from reas_dagilim_table
  where sirket_kod=:logo.sirket_kod and
       brans_kodu=:brans and
       ay=:ay and yil=:yil and
       bilgi_turu='P'
```

```
   group by sube,kaynak,acenta_no,t_i,
         bilgi_turu,trete_kodu,trete_sirket_kodu,
         ay,yil,brans_kodu;
begin
 kom_update;
 begin
  for kayit in cr
  loop
  select /*+ index(kaynak_table kay_ndx) */
       direct into d_i from kaynak_table
       where sirket_kod=:logo.sirket_kod and kaynak_kod=kayit.kaynak;
  begin
  insert into reas_ozet_table values (:logo.sirket_kod,kayit.sube,
       kayit.kaynak,kayit.acenta_no,kayit.t_i,:brans,
       :ay,:yil,'P',null,kayit.trete_kodu,kayit.t1,kayit.trete_sirket_kodu,null,null,
       null,null,null,null,1,trunc(sysdate),substr(user,5,12));
  insert into reas_ozet_table values (:logo.sirket_kod,kayit.sube,
       kayit.kaynak,kayit.acenta_no,kayit.t_i,:brans,
       :ay,:yil,'K',null,kayit.trete_kodu,kayit.t2,kayit.trete_sirket_kodu,null,null,
       null,null,null,null,1,trunc(sysdate),substr(user,5,12));
  end;
  end loop;
 end;
end;
>>>
```

ENDDEFINE PROCEDURE

DEFINE PROCEDURE

```
NAME = kom_update
DEFINITION = <<<
procedure kom_update is
yil_k       char(1);
ara_yil     char(4);
sira        number(2);
trete_kom   number(14,2);
trete_kom1  number(14,2) := 0 ;
oran        number(7,4);
matrah      number(7);
kay         number(3);
mat         number(7);
kom         number(7,4);
d_i         char(1);
cursor cr1 is
 select /*+ index(reas_dagilim_table reas1_ndx) */
      cari_pol_no,trete_kodu,trete_sirket_prim,zeyl_sira_no,t_i
      tarife  from reas_dagilim_table
  where sirket_kod=:sirket_kod and
       brans_kodu=:brans and
       ay=:ay and yil=:yil and
       cari_pol_no is not null and
       zeyl_sira_no is not null and
       tarife is not null and
       (trete_kodu!='01' or trete_kodu!='04') and
       trete_sirket_kodu is null ;
```

```
begin
 begin
  for kayit1 in cr1
  loop
   begin
   select /*+ index(reasurans_header_table reasurans_header_ndx) */
        yil_kodu into yil_k from reasurans_header_table
     where sirket_kod=:logo.sirket_kod and
          tanzim_yili=:yil and
          prim_hasar='P' and
          brans=:brans and
          tarife=kayit1.tarife and
          trete_kodu=kayit1.trete_kodu and
          rownum=1;
   exception when no_data_found then
     begin
     select /*+ index(reasurans_header_table reasurans_header_ndx) */
            yil_kodu into yil_k from reasurans_header_table
           where sirket_kod=:logo.sirket_kod and
                tanzim_yili=:yil and
                prim_hasar='P' and
                brans=:brans and
                tarife is null and
                trete_kodu=kayit1.trete_kodu and
                rownum=1;
           exception when no_data_found then msg_no(4,'REASURANS
                                    HEADER TABLE'||
           'trete'||kayit1.trete_kodu);
                                bell;
                                raise form_trigger_failure;
      end;
   end;
    if yil_k='R' then
      ara_yil:=kayit1.ryil;
    elsif yil_k='U' then
      ara_yil:=kayit1.uwy;
    end if;
  begin
  select /*+ index(reasurans_header_table reasurans_header_ndx) */
       sira_no into sira from reasurans_header_table
    where sirket_kod=:logo.sirket_kod and
         tanzim_yili=:yil and
         prim_hasar='P' and
         brans=:brans and
         yil_kodu=yil_k and
         yili=ara_yil and
         tarife=kayit1.tarife and
         trete_kodu=kayit1.trete_kodu and
         rownum=1;
    exception when no_data_found then
    begin
    select /*+ index(reasurans_header_table reasurans_header_ndx) */
           sira_no into sira from reasurans_header_table
           where sirket_kod=:logo.sirket_kod and
                tanzim_yili=:yil and
                prim_hasar='P' and
```

```
                    brans=:brans and
                    yil_kodu=yil_k and
                    yili=ara_yil and
                    tarife is null and
                    trete_kodu=kayit1.trete_kodu and
                    rownum=1;
               exception when no_data_found then msg_no(4,'REASURANS
                                HEADER TABLE'||'trete'||
               kayit1.trete_kodu||'yil k'||yil_k||'yili'||ara_yil);
                                bell;
                                raise form_trigger_failure;
     end;
   end;
   begin
   select /*+ index(reas_kom_table reas_kom_ndx) */
        komisyon_oran,komisyon_matrah into oran,matrah
     from reas_kom_table
     where sirket_kod=:logo.sirket_kod and
         tanzim_yili=:yil and
         prim_hasar='P' and
         brans=:brans and
         yil_kodu=yil_k and
         yili=ara_yil and
         tarife=kayit1.tarife and
         trete_kodu=kayit1.trete_kodu and
         sira_no=sira and
         e_s='S' and
         k_g='K' ;
   trete_kom:=(oran*kayit1.trete_sirket_prim)/matrah;
   yeni_oran_bul(kayit1.trete_kodu, :brans, kayit1.ryil, oran, matrah,
            kayit1.trete_sirket_prim, trete_kom1) ;
   if trete_kom1 is not null then
     trete_kom := trete_kom1 ;
   end if ;
exception when NO_DATA_FOUND then
        begin
        select /*+ index(reas_kom_table reas_kom_ndx) */
             komisyon_oran,komisyon_matrah into oran,matrah
             from reas_kom_table
             where sirket_kod=:logo.sirket_kod and
                 tanzim_yili=:yil and
                 prim_hasar='P' and
                 brans=:brans and
                 yil_kodu=yil_k and
                 yili=ara_yil and
                 tarife is null and
                 trete_kodu=kayit1.trete_kodu and
                 sira_no=sira and
                 e_s='S' and
                 k_g='K' ;
             trete_kom:=(oran*kayit1.trete_sirket_prim)/matrah;
             yeni_oran_bul(kayit1.trete_kodu, :brans, kayit1.ryil,
                      oran, matrah, kayit1.trete_sirket_prim,
                      trete_kom1) ;
             if trete_kom1 is not null then
               trete_kom := trete_kom1 ;
```

```
            end if ;
            exception when no_data_found  then trete_kom:=0;
          end;
      end;
      update reas_dagilim_table set trete_sirket_kom=trete_kom
          where sirket_kod=:logo.sirket_kod and
              cari_pol_no=kayit1.cari_pol_no and
              zeyl_sira_no=kayit1.zeyl_sira_no and
              ay=:ay and yil=:yil and brans_kodu=:brans and
              trete_kodu=kayit1.trete_kodu;
    end loop;
   end;
  end;
  >>>

ENDDEFINE PROCEDURE

DEFINE PROCEDURE

  NAME = msg_no

  DEFINE REFERENCE

    OWNER = sg
    APPLICATION = genproc
    PROCEDURE = msg_no

  ENDDEFINE REFERENCE

ENDDEFINE PROCEDURE

DEFINE PROCEDURE

  NAME = reas_kapali
  DEFINITION = <<<
  procedure reas_kapali is
  kod1 char;
  begin
  select /*+ index(reas_otomatik_table reas_otomatik_ndx) */
       kod into kod1 from reas_otomatik_table
       where sirket_kod=:logo.sirket_kod and ay=:ay and yil=:yil
           and brans=:brans and kod='O';
       exception
         when no_data_found then
                   msg_no(36,'OTOMATIK DAGILIM YAPILMAMISTIR');
                   bell;
                   raise form_trigger_failure;
  end;
  >>>

ENDDEFINE PROCEDURE

DEFINE PROCEDURE

  NAME = trete_kont
  DEFINITION = <<<
```

```
procedure trete_kont is
begin
declare cursor kontrol is
select /*+ index(reas_dagilim_table reas1_ndx) */
     trete_sirket_kodu from reas_dagilim_table
  where sirket_kod=:logo.sirket_kod and
       brans_kodu=:brans and
       ay=:ay and yil=:yil and
       cari_pol_no is not null and
       zeyl_sira_no is not null and
       tarife is not null and
       trete_kodu='04';
begin
for kayit in kontrol
loop
if kayit.trete_sirket_kodu is null then msg_no(36,'IHTIYARININ TRETE
                                        SIRKET KODU YOK');
                                bell;
                                raise form_trigger_failure;
end if;
end loop;
end;
end;
>>>
```

ENDDEFINE PROCEDURE

DEFINE PROCEDURE

```
NAME = varlik
DEFINITION = <<<
procedure varlik is
sayi number;
begin
select /*+ index(reas_otomatik_table reas_otomatik_ndx) */
     1 into sayi from reas_otomatik_table
 where sirket_kod=:logo.sirket_kod and ay=:ay and yil=:yil and
     brans=:brans and kod='Q';
msg_no(13,'REASURANS CALISMASI KAPANMISTIR');
bell;
raise form_trigger_failure;
exception when no_data_found then sayi:=sayi;
end;
>>>
```

ENDDEFINE PROCEDURE

DEFINE PROCEDURE

```
NAME = yeni_oran_bul
DEFINITION = <<<
procedure yeni_oran_bul (trete in char, brans in char, ryili in char,
                  oran in number, matrah in number,
                  trt_sir_prim in number,
                  trete_komisyonu out number) is
cursor kont is select sirket, hisse_orani from hisse_table
```

```
            where sirket_kod = :logo.sirket_kod and
                calisma_yili = :yil and
                brans_kodu = brans and
                trete_kodu = trete and
                ryil = ryili ;
kk        number(3) ;
koran     number ;
kmatrah   number ;
yeni_kom  number := 0 ;
yeni_prim number := 0 ;
rec       kont%rowtype ;
begin
   select kaynak_kod into kk from trete_kod_table
   where sirket_kod = :logo.sirket_kod and
         trete_kod = trete ;
  open kont ;
  loop
   fetch kont into rec ;
   exit when kont%notfound ;
     yeni_prim := round(rec.hisse_orani * trt_sir_prim / 100) ;
     begin
       select kom_orani, matrah into koran, kmatrah
        from acenta_komisyon_table
       where sirket_kod = :logo.sirket_kod and
             must_kod = rec.sirket and
            kaynak_kod = kk and
             brans_kod = :brans and
         to_char(bas_tarih,'yyyy') = :yil and
             rownum = 1 ;
       yeni_kom := round(koran * yeni_prim / kmatrah) ;
     exception
      when no_data_found then
        yeni_kom := round(oran * yeni_prim / matrah) ;
     end ;
     trete_komisyonu :=round( nvl(trete_komisyonu,0) +
                        nvl(yeni_kom,0)) ;
   end loop ;
   close kont ;
end ;
>>>
```

ENDDEFINE PROCEDURE

DEFINE TRIGGER

```
NAME = PRE-FORM
TRIGGER_TYPE = V3
TEXT = <<<
basla(:logo.sirket_kod,:logo.sirket_adi,:logo.sube_kod,:logo.sube_adi,
    :logo.bugun);
>>>
```

ENDDEFINE TRIGGER

DEFINE TRIGGER

```
  NAME = KEY-OTHERS
  TRIGGER_TYPE = V3
  TEXT = <<<
  bell;
  >>>

ENDDEFINE TRIGGER

DEFINE BLOCK

  NAME = logo
  ROWS_DISPLAYED = 1
  BASE_LINE = 1
  LINES_PER_ROW = 1
  ARRAY_SIZE = 0

  DEFINE FIELD

    NAME = sirket_adi
    DATATYPE = CHAR
    LENGTH = 20
    DISPLAY_LENGTH = 20
    QUERY_LENGTH = 20
    BASE_TABLE = OFF
    PAGE = 1
    LINE = 1
    COLUMN = 1
    INPUT = OFF
    UPDATE = OFF
    QUERY = OFF
    UPPERCASE = ON

  ENDDEFINE FIELD

  DEFINE FIELD

    NAME = bugun
    DATATYPE = CHAR
    LENGTH = 10
    DISPLAY_LENGTH = 10
    QUERY_LENGTH = 10
    BASE_TABLE = OFF
    PAGE = 1
    LINE = 1
    COLUMN = 70
    INPUT = OFF
    UPDATE = OFF
    QUERY = OFF

  ENDDEFINE FIELD

  DEFINE FIELD

    NAME = sirket_kod
    DATATYPE = NUMBER
    LENGTH = 3
```

```
    DISPLAY_LENGTH = 3
    QUERY_LENGTH = 3
    BASE_TABLE = OFF
    DISPLAYED = OFF
    PAGE = 0
    LINE = 0
    COLUMN = 0
    ECHO = OFF
    INPUT = OFF
    UPDATE = OFF
    QUERY = OFF

  ENDDEFINE FIELD

  DEFINE FIELD

    NAME = sube_adi
    DATATYPE = CHAR
    LENGTH = 20
    DISPLAY_LENGTH = 20
    QUERY_LENGTH = 20
    BASE_TABLE = OFF
    PAGE = 1
    LINE = 2
    COLUMN = 1
    INPUT = OFF
    UPDATE = OFF
    QUERY = OFF
    UPPERCASE = ON

  ENDDEFINE FIELD

  DEFINE FIELD

    NAME = sube_kod
    DATATYPE = NUMBER
    LENGTH = 4
    DISPLAY_LENGTH = 4
    QUERY_LENGTH = 4
    BASE_TABLE = OFF
    DISPLAYED = OFF
    PAGE = 0
    LINE = 0
    COLUMN = 0
    ECHO = OFF
    INPUT = OFF
    UPDATE = OFF
    QUERY = OFF

  ENDDEFINE FIELD

ENDDEFINE BLOCK

DEFINE BLOCK

  NAME = blok1
```

```
ROWS_DISPLAYED = 1
ROWS_BUFFERED = 1
BASE_LINE = 1
LINES_PER_ROW = 1
ARRAY_SIZE = 0

DEFINE TRIGGER

  NAME = KEY-COMMIT
  TRIGGER_TYPE = V3
  TEXT = <<<
  genel_kont;
  reas_kapali;
  trete_kont;
  esit_kont;
  varlik;
  insert_et;
  insert into reas_otomatik_table values(:logo.sirket_kod,:ay,:yil,:brans,
                              'Q',trunc(sysdate),substr(user,5,12));
  commit_form;
  if form_success then msg_no(11,null);
            else msg_no(12,null);
  end if;
  bell;
  go_field('brans');
  >>>

ENDDEFINE TRIGGER

DEFINE TRIGGER

  NAME = KEY-EXIT
  TRIGGER_TYPE = V3
  TEXT = <<<
  exit_form;
  >>>

ENDDEFINE TRIGGER

DEFINE TRIGGER

  NAME = KEY-LISTVAL
  TRIGGER_TYPE = V3
  TEXT = <<<
  list_values;
  >>>

ENDDEFINE TRIGGER

DEFINE TRIGGER

  NAME = KEY-PRVFLD
  TRIGGER_TYPE = V3
  TEXT = <<<
  previous_field;
  >>>
```

```
ENDDEFINE TRIGGER

DEFINE FIELD

  NAME = brans
  DATATYPE = CHAR
  LENGTH = 3
  DISPLAY_LENGTH = 3
  QUERY_LENGTH = 3
  BASE_TABLE = OFF
  PAGE = 1
  LINE = 8
  COLUMN = 34
  LOV_TEXT = <<<
  select distinct a.brans_kodu,b.brans_adi into :brans,:brans_adi
  from reas_dagilim_table a,brans_table b
  where a.sirket_kod=:logo.sirket_kod and a.brans_kodu=b.brans_kod and
      b.sirket_kod=:logo.sirket_kod
  >>>
  LOV_TITLE = BRANS VE ADLARI
  QUERY = OFF
  UPPERCASE = ON

  DEFINE TRIGGER

    NAME = KEY-NXTFLD
    TRIGGER_TYPE = V3
    TEXT = <<<
    if :brans is null then msg_no(10,null);
                    bell;
                    raise form_trigger_failure;
    end if;
    brans_kont;
    next_field;
    >>>

  ENDDEFINE TRIGGER

ENDDEFINE FIELD

DEFINE FIELD

  NAME = ay
  DATATYPE = CHAR
  LENGTH = 2
  DISPLAY_LENGTH = 2
  QUERY_LENGTH = 2
  BASE_TABLE = OFF
  PAGE = 1
  LINE = 10
  COLUMN = 26
  QUERY = OFF

  DEFINE TRIGGER

    NAME = KEY-NXTFLD
```

```
  TRIGGER_TYPE = V3
  TEXT = <<<
  if :ay is null then msg_no(10,null);
                  bell;
                  raise form_trigger_failure;
  end if;
  declare month date;
  begin
   month:=to_date(:ay,'mm');
   exception
    when others then msg_no(26,null);
                  bell;
                  raise form_trigger_failure;
  end;
  :ay :=lpad(rtrim(ltrim(:ay,' '),' '),2,'0');
  next_field;
  >>>

 ENDDEFINE TRIGGER

ENDDEFINE FIELD

DEFINE FIELD

 NAME = yil
 DATATYPE = CHAR
 LENGTH = 4
 DISPLAY_LENGTH = 4
 QUERY_LENGTH = 4
 BASE_TABLE = OFF
 PAGE = 1
 LINE = 10
 COLUMN = 34
 QUERY = OFF

 DEFINE TRIGGER

  NAME = KEY-NXTFLD
  TRIGGER_TYPE = V3
  TEXT = <<<
  if :yil is null then msg_no(10,null);
                  bell;
                  raise form_trigger_failure;
  end if;
  declare year date;
  begin
   year:=to_date(:yil,'yyyy');
   exception
    when others then msg_no(26,null);
                  bell;
                  raise form_trigger_failure;
  end;
  >>>

 ENDDEFINE TRIGGER
```

```
    ENDDEFINE FIELD

    DEFINE FIELD

      NAME = brans_adi
      DATATYPE = CHAR
      LENGTH = 15
      DISPLAY_LENGTH = 15
      QUERY_LENGTH = 15
      BASE_TABLE = OFF
      PAGE = 1
      LINE = 8
      COLUMN = 38
      INPUT = OFF
      UPDATE = OFF
      QUERY = OFF
      UPPERCASE = ON

    ENDDEFINE FIELD

  ENDDEFINE BLOCK

  DEFINE SCREEN

    DEFINE PAGE

      PAGE = 1
      PAGE_XS = 0
      PAGE_YS = 0
      PAGE_PX0 = 0
      PAGE_PY0 = 0
      PAGE_PXS = 0
      PAGE_PYS = 0
      PAGE_SX0 = 0
      PAGE_SY0 = 0
      MODE = TEXT
      LINE = 6
      BOILER = <<<
                        REASURANS  KAPAMA
      >>>
      LINE = 8
      BOILER = <<<
                    BRANS KODU :
      >>>
      LINE = 10
      BOILER = <<<
                    AY :   YIL :
      >>>

    ENDDEFINE PAGE

  ENDDEFINE SCREEN

ENDDEFINE FORM
```

EGE_SIGORTA_____ _____ANA_MENU_____
01/01/1995
GENEL_MUDURLUK_____

U Y G U L A M A L A R

1_____TEKNIK_PARAMETRELERI_____
2_____MUHASEBE_PARAMETRELERI_____
3_____HASAR_SERVIS_ISLEMLERI_____
4_____TEKNIK_SERVIS_ISLEMLERI_____
5_____SISTEM_MENUSU_____
6_____MUHASEBE_SERVIS_ISLEMLERI_____
7_____DONUSUM_MENUSU_____
8_____YIL_SONU_ISLEMLERI_____
9_____OS_COMMAND_(UNIX_KOMUT_GIRISI)_____

U&G_DANISMANLIK SAGBIM_YAZILIM_EVI

Count: *0 <List><Replace>

EGE_SIGORTA_____ _____TEKNIK_PARAMETRELERI_____
01/01/1995
GENEL_MUDURLUK_____

U Y G U L A M A L A R

1_____TEKNIK_PARAMETRE_GIRISI_____
2_____ TEKNIK_PARAMETRE_RAPORLARI_____

U&G_DANISMANLIK SAGBIM_YAZILIM_EVI

Count: *0 <List><Replace>

EGE_SIGORTA_____ _____TEKNIK_PARAMETRE_GIRISI_____
01/01/1995
GENEL_MUDURLUK_____

U Y G U L A M A L A R

1_____BRANS_TANIMLAMA_____
2_____BELEDIYE_TANIMLAMA_____
3_____BRANS_YETKI_VERME_____
4_____ISTATISTIK_DEGER_GIRISLERI_____
5_____ISTATISTIK_KODLARI_GIRISI_____
6_____MEBLAG_TABLO_ISIMLERI_GIRISI_____
7_____MEBLAG_FIYAT_PRIM_GIRISI_____
8_____MUAFIYET_PARAMETRELERI_____
9_____POLICE_CINSI_TANIMLAMA_____
10_____REASURANS_BILGILERI_TANIMLAMA_____
11_____TARIFE_TANIMLAMA_____
12_____TEMINAT_OLUSTURMA_____
13_____TRETE_TIPLERI_TANIMLAMA_____
14_____TRETE_KODLARI_TANIMLAMA_____
15_____TARIFE_YETKI_VERME_____
16_____ZEYL_TURU_TANIMLAMA_____

EGE_SIGORTA_____ _____TEKNIK_SERVIS_BILGI_GIRISLERI_____
01/01/1995
GENEL_MUDURLUK_____

U Y G U L A M A L A R

1_____POLICE_GIRIS_____
2_____POLICE_GUNCELLESTIRME_____
3_____POLICE_GORUNTULEME_____
4_____ZEYL_GIRISI(ESKI)_____
5_____ZEYL_GUNCELLESTIRME_____
6_____PRODUKSIYON_KAPAMA_____
7_____REASURANS_DAGILIM_____
8_____REASURANS_GUNCELLESTIRME_____
9_____REASURANS_KAPAMA_____
10_____BULTEN_BILGILERI_____
11_____ZEYL_GIRISI(YENI)_____
12_____POLICE_GIRIS_(_KISA_VADELI)_____
13_____PRODUKSIYON_KAPAMA_____
14_____REASURANS_DAGILIM_____
15_____REASURANS_GUNCELLESTIRME_____
16_____REASURANS_KAPAMA_____
17_____BULTEN_BILGILERI_____
18_____ZEYL_GIRISI(YENI)_____
19_____TECDIT_OLUSTURMA_____
20_____ISIMDEN_POLICE_NO_BULMA_____
21_____TECDIT_GUNCELLESTIRME_____
22_____TARIFE_ACMA_____
23_____HASAR_POLICE_BILGILERI_____
24_____YENI_ACENTA_NO_DAN_POLICE_BULMA_____

U&G_DANISMANLIK SAGBIM_YAZILIM_EVI

Count: *0 ^ <List><Replace>
□

ÖZGEÇMİŞ

11.06.1969 tarihinde İstanbul'da doğmuştur. Beşiktaş Kız Lisesi'nden mezun olmuştur. 1987 yılında Yıldız Teknik Üniversitesi Matematik Bölümü'ne girmiştir. 1991 yılında bölüm birincisi olarak bu bölümden mezun olmuştur. Yine 1991 yılında İ.T.Ü. Fen Bilimleri Enstitüsü Mühendislik Bilimleri Anabilim Dalı Sistem Analizi Programında Yüksek Lisans Eğitimi'ne başlamıştır.

1.5 yıl U&G Danışmanlık A.Ş.'de bilgisayar programcısı olarak çalışmıştır. Bir yıldır KoçBank'ta analist programcı olarak görev almaktadır.