

HAFTA -2- TEMEL KAVRAMLAR

- **Veritabanı** çok genel anlamda, bir kurumun ihtiyaç duyduğu ve kullandığı veriler bütününe ifade eder.
- Bu noktada öncelikle veri ve bilgi kavramları üzerinde durmak faydalı olacaktır.
 - **Veri** ham gözlemler, işlenmemiş gerçekler ya da izlenimlerdir. Bu gözlemler, gerçekler ya da izlenimler harf, rakam ya da çeşitli sembol ve işaretler yardımıyla temsil edilir.
 - **Bilgi** ise yalın tanımlı verinin işlenmiş ve karar verme sürecine destek olacak duruma dönüştürülmiş biçimidir.
- **Veritabanı**, herhangi bir konuda birbirile ilişkili olan ve amaca uygun olarak düzenlenmiş, mantıksal ve fizikselleşmiş tanımlanmış veriler bütündür.
- Veritabanı herhangi bir kurumda birden fazla uygulamada ortak olarak kullanılabilen verilerden oluşur.
- Veritabanı içerisinde sürekli niteliği olan veriler yer alır.
- **Veritabanı Yönetim Sistemi;** veritabanını tanımlamak, oluşturmak, işlem yapmak, farklı kullanıcı yetkilerini belirlemek, bakım ve yedekleme işlemlerini yapmak için geliştirilmiş programlar bütündür.

1.2. Veri Erişim Yöntemleri

- Geçmişten beri veriye erişim amacıyla farklı yaklaşımlar kullanılmıştır.
- Bu yaklaşımlardan tüyü; sıralı erişim, doğrudan erişim ve hesaba dayalı erişim biçimindedir.

1.2.1. Geleneksel Dosya Yapıları (Devam)

- Geleneksel dosya sistemlerinin sakıncaları:
 - Veri tekrarı ve tutarsızlığına yol açar.
 - Veri paylaşımına olanak vermez.
 - Uygulamalarda ihtiyaç duyulan değişikliklerin gerçekleştirilebilmesi için uzmanlık bilgisi gereklidir.
 - İstenilen veriye ulaşmada güçlüklerle karşılaşılır.
 - Verilerin güvenliği ve gizliliği konusunda sorun yaşanır.
 - Verileri yedekleme ve kurtarma konusunda güçlükler yaşanır.

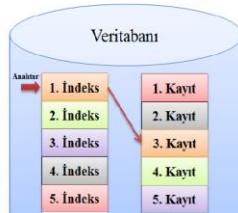
1.2.2. Sıralı Dosyalar

Sıralı dosyalar, bir başka deyişle ardışık dosyalar, içeriği kayıtlara birinci kayıtten başlamak üzere sırayla erişim yapmak üzere tasarlanmış dosyalardır.

Sıralı dosyaların her bir kayıtına ardışık olarak erişilmesi bazı durumlarda yararlı olmasına rağmen, bazı uygulamalarda sorunlar yaratır.

1.2.3. İndeksli Dosyalar

Doğrudan erişimli dosyaların en tanınmışı, indeksli dosyalar olarak bilinir. İndeksli dosyalar, veri dosyasından ayrı olarak bir indeks dosyasının oluşturulması ile birlikte hazırlanır.



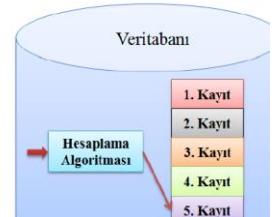
Bir dosya için oluşturulan indeks; söz konusu dosyanın anahtarları ile bu anahtarların disk üzerinde bulunduğu adresi içerir.

Anahtar alan, erişimde kullanılmak üzere seçilen alan olarak değerlendirilir.

1.2.4. Hesaba Dayalı Dosyalar

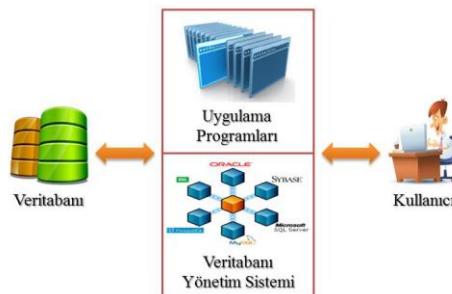
Bu tür dosyalar, indeksli dosyalar gibi ayrı bir indeksin tutulmasını gerektirmez.

Dosyanın herhangi bir kaydına doğrudan doğruya erişebilmek için bir hesaplama algoritması kullanır.



1.3. Veritabanı Yönetim Sistemleri

Geleneksel dosya sisteminin yetersiz kaldığı durumlardaki sorunu çözmek üzere, veriyi saklama ve veriye erişim konusunda yeni yazılım teknolojilerine yönelik başlamış ve Veritabanı Yönetim Sistemleri (VTYS) yaklaşımı ortaya çıkmıştır.



- Veritabanı yönetim sistemleri (VTYS) bilgisayar sistemlerinin önemli bir bileşeni olarak değerlendirilir.
- VTYS, birbirleriyle ilişkili veri ve programlar topluluğundan oluşmaktadır.
- Veri topluluğu bir veritabanı olarak değerlendirilir.
- Veritabanı bir kuruluşla ilişkin bilgilerin yer aldığı ortamdır.
- VTYS, veri kümelerinin düzenli biçimde tutulduğu ve bu verilerin çeşitli yazılımlar aracılığıyla yönetildiği ortamlardır.

Piyasa da yaygın olarak kullanılan başlıca veri tabanı yönetim sistemleri yazılımları:

- Oracle
- Microsoft SQL Server
- PostgreSQL
- Sysbase
- DB2
- MySQL
- MongoDB
- Cassandra
- CouchDB
- Paradox
- Firebird
- Filemaker
- BerkeleyDB
- Informix
- Microsoft Access
- OrientDB
- Neo4j
- Firebase Cloud Firestore

1.3. VYTS'nin Sağladığı Üstünlükler

- Veritabanı kullanımını, geleneksel dosya kullanımına göre birçok yonden üstünlük sağlamaktadır. Bunlar:
 - Verinin tekrarlamasını önerir,
 - Verinin tutarlı olmasını sağlar,
 - Aynı andaki erişimlerde tutarsızlıkların ortaya çıkmasını önerir.
 - Verinin güvenliğini sağlar.

1.3.1. VTYS'nin Sağladığı Yararlar

- VTYS kullanımının sağladığı faydaları şöyle sıralayabiliriz:
 - Veri bağımsızlığı,
 - Etkin veri erişimi,
 - Yetkisiz veri girişlerini engelleme,
 - Veri yönetimi,
 - Veriler arasında karmaşık ilişkiler tanımlama olanağı,
 - Eşzamanlı kullanım,
 - Hata durumunda sistem geri yükleme,
 - Güncel veriye anında erişim imkanı sağlama,
 - Uygulama geliştirme zamanının kısaltılması.

1.3.2. VTYS Kullanımının Dezavantajları

- VTYS kullanımının klasik dosya sistemine göre dezavantajlarını şöyle sıralayabiliriz:
 - Maliyetin yüksek olması (lisans, kurulum, programlama, bakım vs.),
 - Bir arıza durumunda veri dönüşümü daha zordur,
 - Bazı durumlarda uygulamanın yavaş çalışmasına neden olabilir,
 - Karmaşık yapısı nedeniyle kırılganlığı daha yüksektir, yani yolunda gitmeyen problemler ciddi sıkıntılarla yol açabilir.

1.4. Veritabanı Türleri

- Veritabanları;
 - kullanıcı sayısına,
 - fiziksel konumuna,
 - veri modeline göre üç şekilde sınıflandırılabilir.

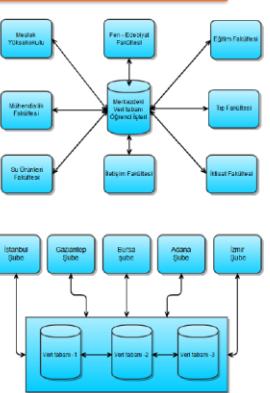
1.4.1. Kullanıcı Sayısına Göre

- Tek-kullanıcılı (single-user) veritabanları, bir seferde yalnızca bir kullanıcının desteklenmesi.
 - Masaüstü veritabanı: tek-kullanıcılı, PC üzerinde çalışır.
- Çok-kullanıcılı (multi-user) veritabanları, aynı anda birden çok kullanıcının desteklenmesi.
 - Kurumsal veritabanları...

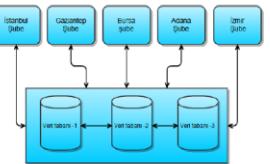


1.4.2. Fiziksel Konumuna Göre

- Merkezi Veritabanı:** Tek bir veritabanı sistemi bulunur. Fiziksel olarak da tek bir merkezde bulunur.



- Dağıtık Veritabanı:** VTYS'nin çeşitli parçaları bir çok bölgeye dağıtılmıştır. Dağıtılmış veritabanı, kullanıcıya tek bir veritabanı gibi gözükmek, fakat ayrı yerlerdeki veritabanlarından oluşur



1.4.3. Veri Modeline Göre

- Bir veritabanı yapısının temelini veri modeli kavramı oluşturmaktadır. Veriyi mantıksal düzeyde düzenlemek için; kullanılan kavramlar, yapılar ve işlemler topluluğuna **veri modeli** denir.
- Birçok veri modeli geliştirilmiştir. Başlıca veri modelleri;
 - Sıra düzensel veri modeli,
 - Ağ (network) veri modeli,
 - İlişkisel veri modeli,
 - Nesneye yönelik veri modeli.

1.4.3.1. Sıradüzensel (Hiyerarşik) Veri Modeli

- Hiyerarşik veri modeli 1960 ve 1970 yılları arasında popüler olan bir modeldir. Bu modelde çoklu ilişkileri temsil edebilmek için, varlık tiplerinin her ilişki için ayrı ayrı tanımlanması gereklidir. Bu da gereksiz veri tekrarına sebep olur.
- Bu modelde, veriler ağaç yapısına benzer bir biçimde modellenir. En üstte kök ve kökün dalları bulunur. Ayrıca her dalın alt dalı sayesinde dallanma ve çeşitlilik artar.
- Bu modelde her bir alt dalın sadece bir tane noktadan bağlanma şartı bu modelin en büyük kısıtlamalarından biridir.

1.4.3.2. Ağ (Network) Veri Modeli

- Ağ veri modeli 1970'li yılların başında geliştirilmiştir. Bir verinin doğası gereği birden çok veri ile ilişkisinin olmasından dolayı hızlıca kabul görmüştür.
- Bu modelde verilerin birbirine ağ şeklinde bağlılığı varsayılmır.
- Bir veri ağında iki kullanıcının haberleşebilmesi için bir çeşit adresleme şekli gereklidir. Genelde, iki veya üç adres gereklidir. Bazı sistemlerde bir fiziksel adres, bir veri bağlantı adresi ve bir ağ adresi kullanılır. Daha yaygın bir yaklaşım, yalnızca fiziksel ve ağ adreslerini kullanmaktadır. Bu yaklaşım, fiziksel ve veri bağlantı adresi aynıdır. Veri iletişimini bu adresler aracılığı ile sağlanmaktadır.

1.4.3.3. İlişkisel Veri Modeli

- Şu anda kullanılan veri tabanlarının çoğu ilişkisel veri modelini desteklemektedir.
- Bu modelde birbiri ile alakalı olan veriler, tablolar içinde saklanır. Ayrıca tablolar arasında değişik türlerde ilişki kurulmaktadır.
- İlişkiler kurulurken birincil anahtar (primary key) ve yabancı anahtarlar (foreign key) kullanılır.
- Anahtar alanlar sayesinde indeksleme yapma olağanı sunan ilişkisel veri tabanlarında erişim ve işlemler daha hızlı yapılmaktadır.
- İlişkiler ve onların temsilleri olan tabloların oluşan veri modelleri ilk olarak 1970 yılında Codd tarafından ortaya atılmıştır.
- Aşağıda gördüğünüz satırlar, basit bir hastane veri tabanının ilişkisel şemasını göstermektedir.

HASTANE (Hastane_Kodu, Hastane_Adı, Adres, Tel_No, Yatak_Sayısı)
POLIKLINIK (Poliklinik_No, Poliklinik_Adı, Hastane_Kodu)
DOKTOR (Hastane_Kodu, Diploma_No, Adı, Soyadı, Uzmanlık_Alımı)

1.4.3.4. Nesneye Yönelik Veri Modeli

- Günümüzde kullanılan ve gelecekte de kullanılacak pek çok uygulamada yalnızca harf, rakam ya da çeşitli karakterler kullanılarak yapılandırılmış verileri değil aynı zamanda multimedya (çeşitli çizim, fotoğraf, görüntü, ses ya da video gibi nesneleri) de içeren veritabanı yönetim sistemlerine ihtiyaç duyulmaktadır.
- Verileri satırlar ve sütunlar biçiminde düzenlemek için tasarlanmış olan veritabanı yönetim sistemleri grafik unsurları ve multimedya unsurlarını kullanmaya pek uygun değildir.
- Bu nedenle bu eksikliği gidermek amacıyla nesneye yönelik veri modelleri geliştirilmiştir.
- Nesneye-yönelik veri tabanları 1990'lı yıllarda kullanılmaya başlanmıştır ve giderek popüler hâle gelmektedir.
- Bunun nedeni, çeşitli multimedya unsurlarını ya da çeşitli kaynaklardan parça parça alınan verileri benzer biçimde birleştiren web uygulamalarında kullanılan Java uygulamalarını yönetmek için kullanılabilir olmasıdır.
- Nesneye yönelik veritabanları ilişkisel veri modellerinden farklı olarak daha karmaşık veri türleri üzerinde işlem yapmasına rağmen, çok sayıda işlemi yürütme açısından ilişkisel veritabanından göreceli olarak daha yavaştır.
- Bu nedenle günümüzde hem ilişkisel hem de nesneye yönelik veri modellerini birlikte kullanan veritabanı yönetim sistemlerinin yaygınlığı görülmektedir.

1.4.4. Diğer Veritabanı Türleri

- Veri ambarları
- NoSQL veritabanları
- Grafik veritabanları
- OLAP veritabanları

1.4.4.1. NoSQL Veritabanları

- Anahtar-Değer (Key-Value)
 - MemcacheDB, Redis, Amazon DynamoDB, Riak, Voldemort
- Doküman-Tabanlı (Document-Oriented)
 - MongoDB, CouchDB, RavenDB, MarkLogic
- Sütun-Tabanlı (Column-Oriented)
 - Hbase, Cassandra, Accumulo
- Çizelge (Graph)
 - Neo4J, OrientDB, Allegro, Virtuoso, Sones, Jena, Sesame

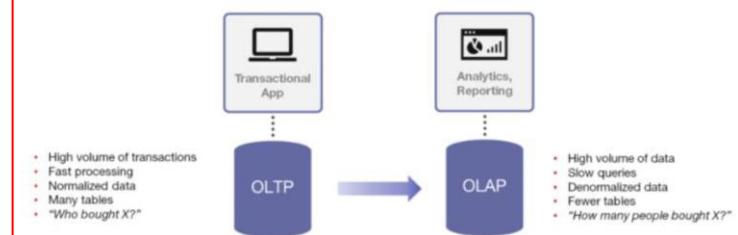


1.4.4.2. OLTP vs OLAP

OLTP: OnLine Transaction Processing

OLAP: OnLine Analytical Processing

OLTP vs OLAP



OLTP

- Günlük operasyonel işlemleri için kullanılır.
- Yapilandırılmış veriler ile çalışır.
- Bir işlem süresinin milisaniye seviyelerinde olması gereklidir.
- Verilerin ana kaynağıdır.
- Yedekleme önemlidir ve periyodik olarak yapılmalıdır.

OLAP

- Karar vermeyi desteklemek, görevleri otomatikleştirmek için kullanılır.
- Yarı yapılandırılmış veya yapılandırılmamış verilerle de çalışabilir.
- Hız beklenmez, saatler sürebilir.
- Veriler çeşitli kaynaklardan toplanır.
- Yedekleme yerine veri kaynaklarından verinin yeniden çekilmesi tercih edilir.

	OLTP	OLAP
Function	Day to day operation	Decision support
Database Design	Application oriented	Subject oriented
Data	Current, up-to-date detailed, flat relational, isolated	Historical, summarized multi-dimensional, consolidated
Usage	Repetitive	Ad-hoc
Access	Read/Write	Lots of scans
Unit of Work	Short, simple transaction	Complex query
Database Size	Gigabytes	Terabytes
Metric	Transaction throughput	Query throughput, response

1.5. Veritabanı Kullanıcıları

- Veritabanı ile herhangi bir şekilde etkileşimde olan kişi ya da kişiler veritabanı kullanıcı olup aşağıdaki gibi sınıflandırılabilirler:
 - Veritabanı Sorumluları
 - + Veritabanı Yöneticisi
 - + Veritabanı Tasarımcısı
 - Son Kullanıcılar
 - Sistem Analistleri ve Uygulama Programcılar

1.5.1. Veritabanı Sorumluları

- Veritabanı sorumluları; veritabanının tasarılanması, oluşturulması ve işletim faaliyetlerinden birinci derecede sorumlu olan ve veritabanı üzerinde en fazla yetkiye sahip olan kullanıcılardır.
- Veritabanı sorumluları, veritabanı yöneticisi ve veritabanı tasarımcısı olarak iki başlık altında incelenebilir.
- Veritabanı yöneticisinin (database administrator) veritabanına erişim yetkilerini belirleme, veritabanı kullanımının düzenlenmesi ve izlenmesini sağlama, ihtiyaç duyulan yazılım ve donanım kaynaklarını edinme biçiminde sıralanan sorumlulukları vardır.
- Veritabanı tasarımcısı (database designer) veritabanında saklanacak olan verilerin tanımlanmasından ve bu verilerin depolanması ve gösterilmesi için gerekli olan uygun yapıların seçilmesinden sorumludur.
- Veritabanı sorumlularının yerine getirdikleri temel görevler:
 - Veritabanı tasarımını yapma,
 - Bütünlük kısıtlamalarını belirleyip tanımlama,
 - Veritabanı kullanım yetkilerini tanımlama,
 - Veritabanı güvenliğini sağlama,
 - Veritabanının işletimini izleme ve sürekliliğini sağlama,
 - Güncellemeye ihtiyaçlarına cevap verebilme,
 - Veritabanından beklenen performansı sağlama.

1.5.2. Son Kullanıcılar

- Son kullanıcılar (end users), yaptıkları işler gereği veritabanına sorgulama ya da güncelleme yapmak veya rapor türetmek için erişen kullanıcılardır.

1.5.3. Sistem Analistleri ve Uygulama Prog.

- Sistem analisti son kullanıcıların gereksinimlerini belirleyen ve standart işlemler yoluyla bu gereksinimleri karşılayabilecek ayrıntıları belirleyen kişi ya da kişilerdir.
- Uygulama programcıları ise sistem analisti tarafından belirlenen ayrıntıları program hâline getiren ve daha sonra test eden, hataları ayıקלayan ve belgeleyen kişilerdir.
- Yaygın olarak yazılım geliştiriciler ya da yazılım mühendisleri olarak da anılan analistler ve programcılar yukarıda sıralanan görevlerini yerine getirebilmeleri için VTYS'nin sağladığı tüm özelliklerini bilmeleri gereklidir.

1. SQL Server Temel Sürümüler

- Enterprise
 - ✓ En üst sürümüdür.
 - ✓ Veri merkezleri için geliştirilmiştir.
 - ✓ Veri merkezi için gereken bileşenleri ve kritik iş yüklerinin yönetilmesi için servisleri içerir.
- Business Intelligence
 - ✓ Şirketler için geliştirilmiş sürümüdür.
 - ✓ Görselleştirme araçları ve tarayıcı üzerinden veriye erişim gibi özellikler içerir.
- Standard
 - ✓ Küçük organizasyonların kullanımına uygundur.
 - ✓ Basit veri yönetimi, geliştirme araçları ve bulut desteği içerir.

1. SQL Server Özelleştirilmiş Sürümüler

- Web
 - ✓ Server maliyetinin minimum olduğu sürümüdür.
 - ✓ Bu sebeple web sunucuları için uygundur.
- Developer
 - ✓ Enterprise'in tüm özelliklerini içerir.
 - ✓ Yalnızca geliştirme ve test aşamaları için lisans sağlar.
- Express
 - ✓ Giriş seviyesi sürümüdür.
 - ✓ Masaüstü uygulamalar ve küçük serverlar için uygundur.

2. SQL Server Servisler: Replikasyon

- Verilerin kopyalanması ve veritabanı nesnelerinin senkronize edilmesi için kullanılır.
- İstemci – sunucu sistemi ile çalışır.
- Değişiklikler sunucudan istemcilere gönderilir.
- SQL Server üç farklı replikasyon desteği sunar.
- Transaction replication
 - ✓ Her transaction'da istemci veritabanları da senkronize edilir.
 - ✓ Veritabanları neredeyse gerçek zamanlı olarak senkronize edilir.
- Merge replication
 - ✓ Sunucu ve istemcideki değişiklikler takip edilir ve periyodik olarak iki yönlü senkronizasyon gerçekleştirilir.
 - ✓ Eğer aynı veri üzerinde her iki tarafta da değişiklik yapıldıysa çakışma olur.
 - ✓ Çakışma el ile düzeltme veya önceden belirlenmiş bir kuralın uygulanması yoluyla çözülebilir.
- Snapshot replication
 - ✓ Tüm veritabanının bir kopyası istemcilere gönderilir.
 - ✓ Son değişiklikler takip edilmez.

2. SQL Server Servisler: Analiz

- Analiz servisleri SQL Server'a OLAP ve veri madenciliği özellikleri sağlar.
- OLAP motoru MOLAP, ROLAP, HOLAP desteği sunar.
- Küp verisine ise MDX veya LINQ sorguları ile erişilebilir.
- Analiz servisleri veri madenciliği için çeşitli algoritmalar içermektedir:
 - ✓ Karar ağaçları
 - ✓ Kümeleme algoritmaları
 - ✓ Naive Bayes algoritması
 - ✓ Zaman serileri analizi
 - ✓ Sıra kümeleme algoritmaları
 - ✓ Regresyon analizi
 - ✓ Yapay sinir ağları

2. SQL Server Servisler: Raporlama

- Veritabanından elde edilen veri için rapor oluşturma araçlarıdır.
- Web arayüzünden yönetilir.
- Raporlar Business Intelligence Development Studio ve Visual Studio'nun birlikte kullanılması ile oluşturulabilir.
- Raporlar RDL dosyaları olarak oluşturulabilir. Daha sonra bu dosyalar Excel, PDF, CSV, XML, TIFF ve HTML formatına dönüştürülebilir.

2. SQL Server Servisler: Bütünlük

- Verinin bir yerden diğerine, belirli kurallar dahilinde taşınması için kullanılır.
- Grafik arayüz desteği ile
 - ✓ farklı kaynaklardan veri alımı,
 - ✓ veri sorgulama,
 - ✓ veri değiştirme,
 - ✓ veri birlleştirme,
 - ✓ tekrarlı verileri temizlemeişlemleri gerçekleştirilebilir.
- Bu işlemlerden sonra veri istenilen yere aktarılır.
- Veritabanının doğru ve tutarlı biçimde çalışması ve işlemleri yerine getirmesi gereklidir. Verinin doğru ve tutarlı olmasına "**veri bütünlüğü**" denir.
- Veri bütünlüğünün sağlanması sonucunda, veritabanının eksik, yanlış, tutarsız ve çelişkili olmaması sağlanır.
- Bir veritabanında belirli kurallar ve kısıtlar altında veri ekleme, veri silme ve veri güncelleme işlemlerinin yapılmasının ardından, o veritabanının fiziksel ve mantıksal yapısı içерdiği verilere nazaran bozulmaz ise **veri bütünlüğü (data integrity)** sağlanmış olur.
- Veritabanında, veri bütünlüğünü sağlamak için birçok yol bulunmaktadır. Bunlar:
 - ❖ Constraint'ler (kısıtlayıcılar),
 - ❖ Rule'lar (kurallar) ve
 - ❖ Default'lar (varsayılanlar) olmak üzere üç çeşidi vardır.

VERİ BÜTÜNLÜĞÜ sağlanmazsa temel SQL servislerinin çalışması bir problem olarak karşımıza çıkar !!!

4. İlişkisel Cebir (devam...)

- Bir ilişkisel veritabanında kullanılan temel ilişkisel cebir ifadeleri şunlardır.
- ✓ Seçme (select) işlemi
 - ✓ Projeksiyon / Atma (project) işlemi
 - ✓ Kartezyen çarpım (cartesian product) işlemi
 - ✓ Birleştirme (join) işlemi
 - ✓ Kesiştirme (intersect) işlemi
 - ✓ Fark (difference) işlemi

5. SQL

- SQL bir sorgulama dilidir.
- Veri tabanları ve tablolar oluşturmayı, oluşturulan tablolara kayıt ekleme, silme, güncelleme ve listeleme işlemlerinin yapılabilmesini sağlayan bir alt sorgulama dilidir.
- SQL'in kendine özgü deyimleri ve kuralları vardır.
 - Hemen hemen tüm ilişkisel veri tabanı yönetim sistemlerinin ortak dili olarak kabul edilmiştir.
- Temel veri tipleri üzerinde **SELECT, WHERE, ORDER BY, CREATE, UPDATE, DELETE** gibi deyimlerle ilişkiler oluşturur/canlandırır/ortadan kaldırır.
- Verileri sorgulamak için SQL kullanan başlica veri tabanı yönetim sistemleri şunlardır:
 - Oracle,
 - Microsoft SQL Server,
 - Sysbase,
 - DB2,
 - MySQL,
 - Microsoft Access.

6. T-SQL

- Microsoft veritabanı sorgulama dilinin gelişmiş bir versiyonudur.
- T-SQL: Transact - Structured Query Language kelimelerinin kısaltmasıdır.
- T-SQL ifadelerinde döngü ve benzeri işlemleri yapmak için herhangi bir derleyiciye ihtiyaç yoktur.
- T-SQL ile öğrendiğiniz bütün SQL komutlarını kullanabilirsiniz.
- SQL Server içerisinde bulunmayan döngü vb. işlemler T-SQL aracılığı ile yapılmaktadır.
- T-SQL, SQL Server üzerinde sorgu oluşturmak için kullanılır.
- SQL'den farklı olarak prosedürel programlamayı, yerel değişkenleri, string, tarih ve matematik fonksiyonlarını içermektedir.
- T-SQL ile;
 - ✓ Veritabanı şemaları oluşturulabilir veya düzenlenebilir.
 - ✓ Şemalar üzerinde veri girişi veya düzenlenmesi yapılabilir.
 - ✓ Veritabanı izlenebilir veya yönetilebilir.

7. İleri SQL Operasyonları

- VIEW (Görünüm) Nesnesi
- Verileri Gruplayarak Analiz Etme
- Stored Procedure
- Kullanıcı Tanımlı Fonksiyonlar
- Transactions
- Triggers

7.2. Verileri Gruplayarak Analiz Etme

- DISTINCT Komutu
- JOIN Komutu
 - INNER JOIN
 - LEFT JOIN
 - RIGHT JOIN
 - CROSS JOIN
- GROUP BY Komutu
- HAVING Deyimi

7.2.3. GROUP BY Fonksiyonu (devam...)

- GROUP BY fonksiyonu belli bir alana göre gruplama işlemi yapmak için kullanılmaktadır. Genel kullanımı şu şekildedir.

```
SELECT Alanlar  
      FROM Tablo_Adı  
      WHERE Şart_İfadeleri  
      GROUP BY Sütun/Sütunlar
```

7.2.4. HAVING Deyimi

- HAVING deyimi sorguda bir koşul belirtilmesi gerekirse kullanılmaktadır. Daha önceki slayttan hatırlayacağınız gibi gruplandırma işlemini gerçekleştirmek için **GROUP BY** fonksiyonunu kullanmıştır.
- **GROUP BY** fonksiyonunda bir koşula ihtiyaç var ise **HAVING** deyimi kullanılmaktadır.
- HAVING deyiminin genel kullanım biçimini aşağıdaki gibidir.

```
SELECT Alan_Adları  
      FROM Tablo/Tablolar  
      [WHERE Şart/Şartlar]  
      [GROUP BY Sütunlar]  
      [HAVING Grup_Kısıtlaması]
```

- HAVING deyimi, GROUP BY fonksiyonu olmadan kullanılamaz.
- Gruplama yaparken SELECT içinde yazılan alan isimleri mutlaka GROUP BY içinde geçmelidir.

7.3. Stored Procedure

- Prosedür, belli bir görevi yerine getirmek için yazılmış program parçacıklarıdır. Başka bir deyişle, herhangi bir işlevi yerine getirmek için yazılan kodların bir paket içerisinde tutulmuş halidir.
- Bir prosedür başka bir prosedür tarafından da çağrılmaktadır. Bu da, sık kullanılan işlemlerin bir defa yazılarak programın akışına göre defalarca kullanılmasını sağlamaktadır. Böylelikle kod yazımı ve programlama işlemi kolaylaştırılmış olmaktadır.

7.4. Kullanıcı Tanımlı Fonksiyonlar

- Fonksiyonlar belli bir sonucu geri döndürmek için tasarlanmış bir veya birden fazla yerde kullanılan yapılardır.
- Kullanıcı tanımlı fonksiyonlar, tíkı Stored Procedure'ler gibi dışarıdan parametre alabilirler.
- Aynı zamanda IF...ELSE gibi T-SQL'in diğer ifadelerinin de kullanılmasına imkan tanırlar.

7.5. Transactions

- Transaction veritabanındaki verilere erişen ve çoğunlukla bu veriler üzerinde değişiklikler yapan bir program kesimidir.
- Birçok kullanıcının eşzamanlı olarak işlem yaptığı büyük veritabanı sistemlerinde daha çok kullanılmaktadır.
- Bir işlem büyük bir bütününe parçası olabilir. Bu işlemlerden herhangi bir tanesinin gerçekleşmemesi bütün işlemleri anlamsız kılmaktadır.
- Böyle bir durumda bütün işlemler tek bir işlem gibi ele alınmalıdır. Bu parçalanamaz işlemlerin oluşturduğu yeni tek işleme **«transaction»** adı verilmektedir.

- Bir veritabanı veriler üzerinde değişiklik yaparken dört kuralı sağlamak zorundadır. Sağlanması gereken bu dört kural aşağıdaki gibidir.

- Bölünmezlik (Atomicity)
- Tutarlılık (Consistency)
- İzolasyon (Isolation)
- Dayanıklılık (Durability)

ACID

7.6. Triggers

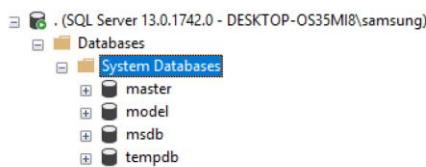
- Trigger yani tetikleyici, ilişkisel veritabanı yönetim sistemlerinde bir tabloda belirli olaylar meydana geldiği zaman yani ekleme, güncelleme, silme işlemlerinden biri gerçekleşmeden önce veya sonra çalışan ve belirli işlemleri kodlandığı şekilde yerine getiren yordamdır.
- Tetikleyiciler veritabanında yapılan değişikliklerle birlikte otomatik olarak çalışan prosedürel program parçacıklarıdır.
- Trigger'lar, veri değişiminin hemen ardından log dosyalar üzerinden otomatik olarak devreye giren özel bir Stored Procedure'dür.

HAFTA -4-

MsSQL System Databases ve Analiz Servisleri

1. System Databases

- MsSQL Server'da kullanıcı tanımla veri tabanları dışında System Databases adında 4 adet veri tabanı yer alır:
 - master
 - model
 - msdb
 - Tempdb
- MsSQL Server'da yapılan her işlem veri tabanı sunucusu üstünde tutulur.



1.1. System Databases: master

- Sistem veri tabanlarının en önemli master veri tabanıdır.
 - Sistem üzerinde kullanıcıların oluşturduğu veri tabanı listesi
 - `select * from sys.master_files`
 - Kullanıcı veya sistem tarafından belirlenen login bilgileri
 - `select * from sys.sql_logins`
 - Bu kullanıcıların roller ve yetkileri
 - `select * from sys.server_principals`

gibi bilgileri tutar.

1.2. System Databases: model

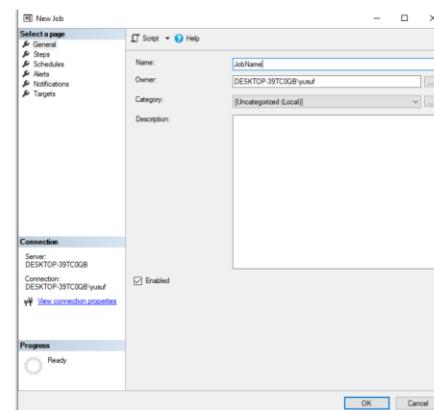
- model veri tabanı sunucuda oluşturulan her veri tabanı için bir temel model (taslak) oluşturur.
- SQL server üzerinden oluşturulan yeni bir veri tabanı, model veri tabanının birebir kopyasıdır.
 - `CREATE DATABASE` deyimi kullanıldığında ilk olarak model veri tabanının içeriği kopyalanarak veri tabanının ilk kısmı oluşturulur.
- Bu nedenle, ortak bazı fonksiyonlar ya da işlevsellikler model veri tabanı üzerinde tanımlanarak sunucuda kullanıcı tanımlı oluşturulan her veri tabanına kolayca aktarılabilir.
 - Gelen kelimenin tamamını büyük harfe çeviren bir fonksiyon tanımlanarak kullanıcının oluşturduğu tüm veri tabanlarında kullanılmak istendiğini varsayıyalım. Her veri tabanında aynı fonksiyonu yazmak yerine bunu model veri tabanında bir kere yazmanız yeterli olacaktır

1.3. System Databases: msdb

- SQL Server Agent için hizmet eder.
 - SQL Server Agent içerisinde job tanımlanabilen bir yapıdır.
 - Job belirtilen zamanlarda tek sefer veya tekrarlı olarak çalışan ve kendisine atanmış görevleri yerine getiren bir mekanizmadır.
 - Her gece saat 02:00'de veri tabanı backup'ının alınması ve işlem sorunsuz bir şekilde tamamlandıysa veri tabanı yöneticisine mail atılması gibi
 - Eğer kullanıcı yeni bir Job açarsa veya var olan bir Job'ın işleyişini değiştirirse bunların kayıtları burada saklanır.

1.3.1. Job Oluşturma Adımları

- General: Job adı, sahibi, tanımı gibi genel bilgiler tanımlamaya yarar.



1.3.2. Job Tetikleme

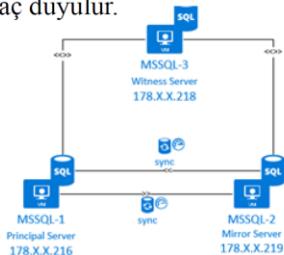
- Zamanlanmış job objeleri belirtilen zamanda tek bir sefer ya da tekrarlı olarak çalışır.
- Elle Job tetikleme için Job objesini sağ tıklayarak «Start Job at Step...» seçeneği tıklanır.
- Ayrıca, Properties penceresi üzerinde görünen «View Job History» seçeneği ile, ilgili Job'ın çalıştırılma geçmişine ve başarıyla sonuçlanıp sonuçlanmadığına bakılabilir.

1.4. System Databases: tempdb

- Geçici bir veri tabanıdır
- Üzerinde sistem ya da kullanıcı objeleri geçici olarak saklanabilir.
 - Tablo, stored procedure, vd.
- Örneğin yoğun bir raporlama sorgusu yapılacağını ve bu sorgu esnasında geçici bir tabloda bazı verilerin toplanıp okuması gerektiğini düşünelim. Okunacak veriler işlem bittikten sonra gerekli değilse geçici bir tabloda tutmak daha uygun olur.
- tempdb üzerinde bir tablo açmak için
 - `create table #tabloadi`
 - TempDB üzerinde işlem yapmak için tablo isminin başına '#' karakterini yerleştirilir.
 - Temp tabloları kullanıcı bazlı değil oturum bazlı çalışır. Oturum sonlandığında silinirler.
- Aynı kullanıcı iki farklı oturumda aynı tablo üzerinde işlem yapamaz.
- Eğer bir geçici tabloya farklı oturumların her biri tarafından erişilmek isteniyorsa başına ## eklenmelidir.
 - `create table ##tabloadi`
 - «geçici tablo oluşturma ve erişme demo»

2. Aynalama

- Bu yapı felaket (disaster) diye adlandırılan ve sunucuda problem olması durumunu ifade eden zamanlarda, sunucunun çok kısa sürede (15 – 60 sn arası) kaldığı yerden devam etmesi için önerilmektedir.
- MS SQL ceri tabanı aynalama işlemi için 3 adet SQL server sunucusuna ihtiyaç duyulur.



- Principal:** Uygulamaların bağlandığı sunucu
- Mirror:** Veri kopyasının tutulduğu sunucu
- Witness:** Felaket durumunda sunucular arası geçiş sağlayıcı sunucu

3. Maskleme

- Dynamic Data Masking
 - SQL Server 2016 ile gelen bir özellikir.
 - View oluşturmadan maskleme yapmaya olanak sağlar.
 - default, email, random ve partial olmak üzere 4 adet fonksiyonel parametre ile çalışır.
 - Sadece select sorgusu ile yetkilendirilmiş kullanıcıların verilerini maskeler.
 - Yeni tablo oluştururken CREATE sırasında ya da mevcut tablolara ALTER komutu ile maskleme yapılabilir.
 - Bu işlemi yapan 3rd party araçlar da mevcuttur.

```
CREATE TABLE OGRENCELER
```

```
(  
OgrenciID INT PRIMARY KEY IDENTITY,  
Adi NVARCHAR(10) MASKED WITH (FUNCTION = 'default()') NULL,  
SoyAdi NVARCHAR(10) MASKED WITH (FUNCTION = 'default()') NULL,  
TCNo int MASKED WITH (FUNCTION = 'default()') NULL  
)
```

```
INSERT INTO OGRENCELER VALUES('Yusuf', 'Özçevik', '1234567')
```

```
INSERT INTO OGRENCELER VALUES('Müge', 'Özçevik', '7654123')
```

```
Select * from [dbo].[OGRENCELER2]
```

```
CREATE USER YETKILIUSER WITHOUT LOGIN  
GO  
GRANT SELECT ON OGRENCELER TO YETKILIUSER
```

```
EXECUTE AS USER = 'YETKILIUSER'
```

```
SELECT * FROM OGRENCELER
```

4. Veri Aktarımı

- SQL Server Import and Export Wizard
(SQL Server İçe Aktarma ve Dışa Aktarma Sihirbazı)



5. Sorgu Performansı

- Sunucuların performansını hızlandırmak için SQL sorgularını iyileştirmek mümkündür.
- Genel amaç, bir kullanıcının bir sorgu yürütmesiyle birlikte sonuç alması için geçen süreyi ve bir sorguyu işlemek için kullanılan kaynak miktarını azaltmaktadır.
- Veritabanı, bilgisayarda çalışan bir yazılım olarak tüm yazılımlarla aynı sınırlamalara tabidir ve yalnızca donanımının işleyebileceği kadar bilgiyi işleyebilir.
- Bir sorguyu daha hızlı çalıştırmanın yolu, yazılımın (ve dolayısıyla donanımın) gerçekleştirmesi gereken hesaplama sayısını azaltmaktır.
- Bunu yapmak için, SQL'in gerçekte nasıl hesaplama yaptığıńı biraz bilmek gerekir.
- Tablo boyutu:** Bir sorgu milyonlarca satır içeren bir veya daha fazla tabloya ulaşırsa, bu durum performansı kötü etkileyebilir.
- Join işlemleri:** Bir sorgu iki tabloyu, sonuç kümesinin satır sayısını önemli ölçüde artıracak şekilde birleştiriyorsa, büyük olasılıkla yavaş olacaktır.
- Aggregation işlemleri:** Bir sonuç üretmek için birden çok satır birleştirerek, bu satırları basitçe almaktan daha fazla hesaplama gerektirir.
- Sorgu çalışma zamanı, veritabanının kendisiyle ilgili olarak geliştiricinin kontrol edemediği bazı şeylere de bağlıdır:
 - Sorgu çalıştırın diğer kullanıcılar:** Bir veritabanında aynı anda ne kadar çok sorgu çalıştırılırsa, veritabanı belirli bir zamanda o kadar fazla işlem yapmak zorunda kalır ve her şey o kadar yavaş çalışır. Bazı son kullanıcı uygulamalarının, yukarıdaki kriterlerden bazlarına göre, özellikle yoğun kaynak kullanan sorgular çalıştırması performansı kötü etkileyebilir.
 - Veritabanı yazılımı ve optimizasyonu:** Bu muhtemelen kontrol edemeyeceğiniz bir şemdir, ancak kullandığınız sistemi biliyorsanız, sorgularınızı daha verimli hale getirmek için sınırlar dahilinde çalıştırılabilirsiniz.

5.1. Tablo Boyutunu Küçültme

- Tablo sütunlarında **null** alan değerler varsa, sütunda tutulan değişken türü bellek (kaynak) kullanımını etkiler.
- Int** değişken türünde değer tutan bir sütunda **null** değerler bellek alanı tüketirken **varchar(100)** değerler tüketmez.
- 0-1, **Doğru-Yanlış** gibi ikili ifadeler içeren bir sütun türü **int** ya da **text** seçilirse, **bit** seçilmesine göre daha fazla bellek alanı tüketir.
- Dolayısıyla, sütunda tutulan verilere ilişkin değişken türü seçimi verİYE göre belirlenerek tablonun bellekte kapladığı alan küçültülebilir.
- Bellekte depolanan miktar küçültülebileceği gibi **select** sorguları sırasında uygulanan filtrelemeler ile sorguların hızlandırılması mümkündür.
 - Verileri yalnızca ihtiyaç duyulan filtreleri kullanarak sorgulamak, sorgu hızını önemli ölçüde artırabilir.
 - Bunun nasıl yapılacağı tamamen çözmeye çalışılan soruna bağlı olacaktır. Örneğin, zaman serisi verileriniz varsa, küçük bir zaman aralığıyla sınırlamak, sorgularınızın çok daha hızlı çalışmasını sağlayabilir:

```
SELECT * FROM Siparisler  
WHERE siparisTarihi >= '2014-03-01' AND teslimTarihi < '2014-04-01'
```

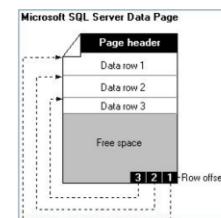
5.2. Join İşlemlerini Basitleştirme

- Join işlemleri iki veri kümесini birleştirerek tüm varlık kümesi üzerinde işlemler yapmaya yarar.
- Birleştirme işlemi için farklı join ifadeleri yazmak mümkündür.
- Farklı join ifadeleri farklı sayıda veri içeren veri kümeleri oluşturabilir.
- Bu durumda, sorgu gereksinimlerini karşılayacak şekilde, veri kümescini boyutunun daha küçük olduğu join ifadeleri tercih edilmelidir.
- Bu amaçla, çalışma planı (Execution Plan) tahmini ve gerçek çalışma planı sonuçları incelenerek sorgu iyileştirmesi yapılabilir.

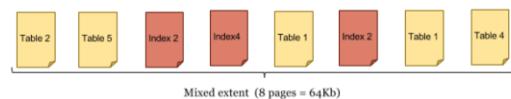
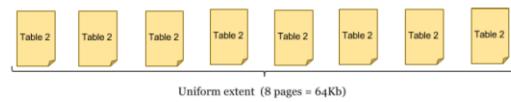
5.3. Index Oluşturma

- Index, veri tabanı tabloları üzerinde tanımlanan ve veriye daha az işlemle daha hızlı ulaşan veri tabanı nesneleridir.
- Indexler hakkında klasik bir örnek olarak telefon rehberi verilebilir.
- Telefon rehberindeki kayıtların sıralı olmaması durumunda, yani her kaydın telefon defterinde rastgele tutulması durumunda, aranacak bir isim için tüm rehberi gezmemiz gereklidir.
- Ama rehberdeki kayıtlar sıralı olsaydı, aranan ismin rehberin ortasındaki isimden ileride mi yoksa geride mi olduğuna bakılabilirdi. Bu şekilde aranan verileri eleyerek bir kaç adımda istenilen sonuca ulaşılabilirdi.
- Bu örnekteki gibi verinin sıralı tutulmasını sağlayan nesnelere index denir.

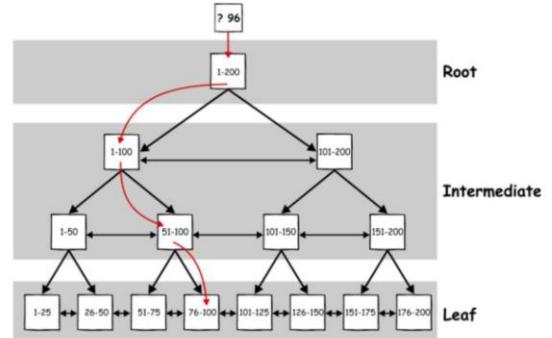
- SQL Server yeni bir veri tabanı oluşturulduğunda dosyaları mantıksal olarak 8 KB'lık bloklara böler. Bu bloklara **page** adı verilmektedir.
 - Bir dosyanın ilk 8 KB'ı page0, bir sonraki 8 KB'ı page1 olur ve bu şekilde devam eder.
- Page'lerin içinde ise tablolardaki satırlara benzeyen ve adına **row** denilen yapılar bulunur.
- Sql Server page'ler üzerinde başka bir mantıksal gruplama daha yapar.
 - Art arda 8 tane page'in bir araya gelmesiyle oluşan 64 KB büyütüğündeki veri yapısına **extent** denir.
- Her page içinde bulunan satır sayısı aynı değildir.
- Page'ler, veri büyütüğünne göre değişen satırlara sahiptir ve bir satır sadece bir page içinde olabilir.
- Sql Server aslında satırları okumaz bunun yerine page'leri okuyarak verilere ulaşır.



- Bir extent dolduğunda; bir sonraki kayıt, kayıt boyutu büyütüğünde yeni bir extent'e yazılır.
- Bir extent tamamı aynı tabloya ait sayfalar içerebileceği gibi; farklı tablolara ait sayfaları içerebilir.
- Sql Server'da bir tabloya index tanımlandığı zaman o tablodaki verileri bir ağaç yapısına göre organize eder.



- ÖRNEK:



- Resimde görüldüğü gibi aradığımız veriyi üç adımda bulabiliriz.
- Index kullanılmamasıydı, veriler sayfalara rastgele dağıtılacığı için tüm kayıtları gezmek gerekecektir.

- Sql Server'da indexler temelde clustered ve non-clustered index olmak üzere ikiye ayrılır.
 - Yaprak düğümlerde tutulan verinin kendisi ise clustured
 - Verinin hangi sayfada tutulduğunu gösteren işaretçi ise non-clustered
- Clustered index'ler tablodaki veriyi fiziksəl olarak sıralar.
- Bir tablo fiziksəl olarak sıralandığından tablo üzerinde sadece bir tane clustered index tanımlanabilir.
- Clustered index için seçilecek sütun veya sütunlar sorgulardaki en fazla kullanılan sütunlar olmalıdır.
 - Veriler, bu sütunlara göre fiziksəl olarak sıralanacağından çok hızlı erişilir. Ayrıca seçilen sütunun çok değiştirilmeyen bir alan olması gereklidir. Çünkü index'e ait sütunun değişmesi demek tüm index'in yeniden organize olması yani fiziksəl olarak yeniden sıralanması anlamına gelir. INSERT, UPDATE ve DELETE işlemleri maliyetlidir.
- Non-Clustered Index veriyi fiziksəl değil mantıksal olarak sıralar.
- Yaprak düğümlerde verinin kendisi değil nerede olduğu bilgisi tutulur.
- Tablo üzerinde en fazla 999 tane non-clustered index tanımlanabilir.
- Non-clustered index'ler veriye doğrudan erişemez.
- Bu index'i oluştururken sorguların koşul kısmında sık kullanılan sütunlar dikkate alınmalıdır.
- Bir tabloda index oluşturulmaya başlandığında Sql Server tabloyu kitler ve erişimi engeller.
- Index oluşturma işlemi tablodaki veri sayısına göre kısa veya uzun sürebilir.
- Clustered ve non-clustered index'ler farklı özellikler içerecek şekilde tanımlanabilirler.
 - Unique Index:** Veri tekilliği sağlar.
 - Filtered Index:** Belirli koşula uygun veriye index tanımlar.
 - Composite Index:** Birden fazla sütun üzerinde index tanımlar.
 - Covered Index:** Sorgularda index dışında kalan alanları ama sorguda yer alan sütunları eklemeye yarar.
 - Full-text Index:** Büyük boyutlu veri içeren alanlarda (varchar(max), nvarchar(max), xml, text) hızlı arama yapabilmek için kullanılır.
- Özetlemek gerekirse,
 - Yoğun şekilde veri güncelleme işlemi olan tablolarda, index tamında mümkün olduğunda az sütun seçilmelidir.
 - Veri güncellenenin az olduğu tablolarda daha çok index tanımlanabilir.
 - Clustered index mümkün olduğunda az sütuna tanımlanmalıdır.
 - Mممكئنse clustered index'ümüz unique olan sütunda tanımlanmalı ve null değeri içermemeli.
 - Index tanımlanan sütunda ne kadar tekrarlı veri varsa index performansınız düşük olacaktır.
 - Index tamında sütün sayısı, yapılacak INSERT, UPDATE ve DELETE işlemlerinde performansı direkt olarak etkileyecektir.

HAFTA -5-

MsSQL İlişkisel Veritabanı Alternatifleri

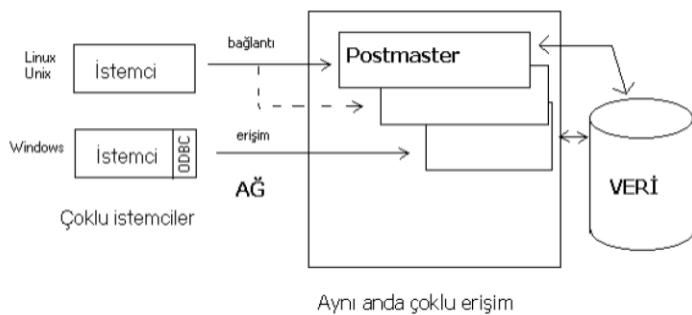
1. MySQL Database

- Genel olarak bir Linux dağıtımında üzerinde Apache Web Sunucusu veya PHP ile birlikte kullanılır.
 - LAMP (Linux, Apache, MySQL, PHP)
 - WINS (Windows, IIS, .Net, MsSQL)
 - MsSQL için Linux dağıtımında 2019 yılı itibarıyla mevcuttur.
- Veri depolama yöntemi MsSQL'den farklıdır.
 - Çok sayıda depolama motoru mevcuttur.
 - InnoDB, MyISAM, Memory, CSV, Merge, Archive, Federated, Blackhole
 - Hız, güvenilirlik veya başka bir sebepten dolayı farklı tablolar için farklı motorlar kullanılabilir.
 - Geliştiricilere esneklik sağlar.
- Yönetim aracı olarak Oracle firması tarafından geliştirilen MySQL Workbench yaygın olarak kullanılmaktadır.
- MsSQL ile birçok benzer özelliği vardır.
 - Clustered Yapı**
 - Views**
 - Stored Procedures,**
 - Triggers,**
 - SQL Fonksiyonları**
 - Kullanıcı tanımlı fonksiyonlar**
- MsSQL ile performans farklılıklarını akademik olarak bazı çalışmalarında araştırılmaktadır.
- Hangisinin tercih edileceğini genellikle geliştirme ekosisteminin diğer bileşenleri belirler.
 - LAMP vs WINS**

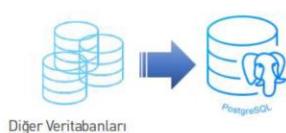
MS SQL Server	MySQL
Developed by Microsoft.	Developed by Oracle.
It supports programming languages like C++, JAVA, Ruby, Visual Basic, Delphi, R etc.	MySQL offers extended running support for languages like Perl, Tcl, Haskey etc.
Expects a large amount of operational storage space.	Expects less amount of operational storage space.
It enables for stopping query execution.	It doesn't allow query cancellation mid-way in the process.
Doesn't block the database while backing up the data.	Blocks the database while backing up the data.
It is not free.	It is open source. It is freely available.
It is a highly secured and doesn't allow any kind of database file manipulation while running.	It allows database file manipulation while running.
It is available in multiple editions, such as Enterprise, Standard, Web, Workgroup, or Express.	It is available in MySQL Standard Edition, MySQL Enterprise Edition, and MySQL Cluster Grade Edition.

2. PostgreSQL Database

- PostgreSQL, veri tabanları için ilişkisel veri modeli kullanan ve SQL standart sorgu dilini destekleyen bir veri tabanı yönetim sistemidir.
- Hemen hemen tüm UNIX ya da Unix türevi (Linux, FreeBSD gibi) işletim sistemlerinde çalışır.
 - Ücretsiz ve açık kaynak kodludur.
- Ticari ya da açık kodlu veri tabanlarında bulabileceğiniz özelliklerin hemen hemen hepsini kapsar.
 - Transactions, iç içe SELECT sorguları, view nesneleri, foreign key ile veri bütünlüğü, fonksiyonlar, rule nesneleri
- PostgreSQL gücünü mimarı yapısından alır.



- Postmaster: Veri tabanı sunucu işlemi (process)
- Türkçe'ye yerelleştirilmiştir ve Türkçe desteği vardır.
- Platform bağımsızdır.
 - PostgreSQL'i kullanmak için geliştirme ortamımızı veya sistemlerimizi değiştirmenize gerek yoktur.
 - Tüm modern işletim sistemleri (Linux, Unix, Windows, Mac OS, vb.) ve işlemciler (x86, x86_64, IA64, vb.) üzerinde çalışır
- ACID prensipleri ile tam uyumludur.
- SQL Standartlarına (ANSI-SQL 2008/2011) en çok uyum gösteren veri tabanlarından biridir.
 - Yüksek güvenlidir.
 - Yüksek erişilebilirliğidir.
 - Genişleyebilir mimariye sahiptir.
 - Her işlem ve veri bütünlüğüne göre ölçeklenebilir, esnek, genişleyebilir veya daraltılabilir.
- PostgreSQL'in mimarisini, hem ilişkisel hem de ilişkisel olmayan modellerden herhangi bir mevcut veri tabanı sisteminde geçiş destekleyeceğinden esnekdir.



- Yapısal veri modellerinin yanında, yarı yapısal ve yapısal olmayan (NoSQL) veri yapılarını da destekler.
- JSON (JavaScript Simple Object Notation) ve JSONB (JSON Binary) veri türlerini destekler.
- Anahtar/değer (key/value) çiftleri PostgreSQL hstore uzantısı tarafından desteklenir.

- C#, C/C++, Java, Python, JavaScript (Node.js), Ruby vb. dahil tüm popüler programlama dillerini destekler.
- MsSQL ve MySQL'de yer alan birçok benzer yapı barındırır.
 - Primary Keys
 - Foreign Keys
 - Constraints
 - Stored procedures
 - Functions
- PgAdmin adlı verilen yönetim aracı yönetilmektedir.

	PostgreSQL	MySQL
Known as	The most advanced open-source database.	The most popular open-source database.
Development	An open-source project .	An open-source product .
GUI tool	PgAdmin	MySQL Workbench
ACID	✓	✓
Storage engine	Single	Multiple
Data types	Support advanced types such as array, hstore.	SQL-standard types
Auto increment Column	SERIAL	AUTO_INCREMENT

3. Oracle Database

- Oracle firması tarafından geliştirilen ve Oracle RDBMS olarak bilinen gelişmiş bir ilişkisel veri tabanı yönetim sistemidir.
 - Oracle firması en büyük teknoloji devlerinden biridir.
 - Java programlama dilini geliştiren bir şirkettir.
 - Java, Sun Microsystem tarafından yaratılmış olan nesne tabanlı programlama dilidir.
 - Açık kodlu ve bağımsız oluşunun yanı sıra yüksek verimli, çok işlevli, adım adım geliştirilebilen bir yapıya sahip olması yönünden çok elverişlidir.
- Oracle veri tabanı yönetim sistemi, geliştirme ekosisteminde bulunan Java programlama dili ve ilgili unsurları barındıran diğer araçlar için yaygın olarak tercih edilmektedir.
- Kurumsal alanda kullanılan yaygın bir veri tabanı yönetim sistemidir.
 - Çok sayıda araçtan oluşur ve uygulama geliştiricilerinin kolay ve esnek uygulamalar geliştirmesini sağlar.
- Farklı ölçeklerdeki kurumsal çözümler için farklı sürümleri mevcuttur.
 - MsSQL'de yer alan sürümler gibi.
 - Oracle firması tarafından lisanslanmaktadır.
- PL/SQL adı verilen dil ile kullanılır.
 - Procedural Language extensions to SQL temel olarak Oracle veritabanı içerisinde programlama dillerinde yer alan if, loop, while, fonksiyon, prosedür gibi yapıları kullanmaya imkan veren bir programlama dilidir.
 - Birçok veritabanı SQL olarak adlandırılan veri sorgulama dilini kullanır.
 - Anahtar Oracle, SQL Server gibi gelişmiş veritabanı yönetim sistemleri SQL dilini esnek olarak kullanmak için çeşitli programlama dillerinde yer alan özellikleri ek dillerle karşılar.
 - SQL Server T-SQL dilinin benzeridir.

	Oracle	MsSQL
Geliştirici	Oracle	Microsoft
İşletim Sistemi	AIX, HP-UX, Linux, OS X, Solaris, Windows, z/OS	Windows, Linux
Desteklenen Programlama Teknolojileri	C, C#, C++, Clojure, Cobol, Eiffel, Erlang, Fortran, Groovy, Haskell, Java, JavaScript, Lisp, Objective C, Ocaml, Perl, PHP, Python, R, Ruby, Scala, Tcl, Visual Basic	.Net, Java, PHP, Python, Ruby, Visual Basic
Sunucu tarafındaki scriptler	PL/SQL	Transact-SQL
ACID	✓	✓

4. SQLite

- C ve C++ programlama dilleriyle yazılmış açık kaynaklı bir veri tabanı yönetim sistemidir.
- SQLite'ı diğer veri tabanı motorlarından ayıran başlıca özelliği **basit** olmasıdır.
 - Lite → Hafif
 - Basit Yönetim
 - Basit İşletim
 - Daha büyük programların içine basit entegrasyon
 - Basit sürdürülebilirlik
 - Basit donanım üzerinde
- Projelere bir modül olarak dahil edilebilir.
- Ayrı bir yazılım veya sunucu kurulumu gerektirmez.
- Veri tabanını diskte saklanır ve kolaylıkla transfer edilebilir.
- Söz dizimi oldukça basittir, rahat öğrenilebilir.
- Her ne kadar basit bir veritabanı gibi gözükse de oldukça güçlü bir veritabanıdır ve kullanımı yaygındır.
 - Mobil uygulama geliştirirken
 - Android işletim sisteminde
 - iOS işletim sisteminde
- Her veri tabanı bir dosyadır, bu nedenle veri tabanları çoğunlukla birbirinden yalıtılmıştır.
- ACID prensiplerini destekler.
- Replikasyon, bölümleme, analiz gibi ileri seviye servisler sağlamaz.
- Sunucu tarafı script dili yoktur.
 - T-SQL ya da PL/SQL gibi bir script dili mevcut değildir.
- Veri hacmi arttığında performansı düşer.
 - Bu nedenle yalnızca küçük bir veri kümесini işleme sırasında performanslıdır.

- Eş zamanlı olarak bir bağlantıya olanak sağlar.
- Kullanımının uygun olduğu durumlar aşağıdaki gibi sıralanabilir:
 - Küçük bağımsız uygulamalar geliştirirken
 - Fazla ölçeklenebilirlik gerektirmeyen küçük projeler geliştirirken
 - Doğrudan diskten okuma ve yazma gereksiniminiz olduğunda
- Sqlitebrowser, Sqlite veri tabanlarının içeriğini grafik bir arayüz aracılığıyla görüntüleyebilmemizi sağlayan bir programdır.
- Tümleşik geliştirme ortamlarının bazılarında (Android Studio gibi) Sqlite veri tabanı yönetimine olanak sağlayan eklentiler bulunur.

5. Özet

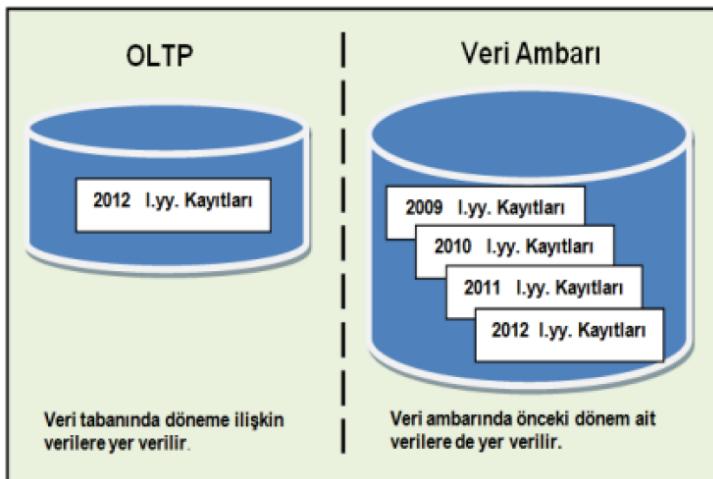
- İlişkisel veri modelini destekleyen ve bu modele uygun veri depolama sağlayan farklı veri tabanı yönetim sistemleri mevcuttur.
 - Bu sistemlerin bazıları firmalar tarafından üretilip pazarlanırken (MsSQL, Oracle, etc.) bazıları ise açık kaynak kodlu olarak geliştirilmekte ve/veya kullanılmaktadır (MySQL, PostgreSQL, SQLite, etc.).
- Her bir VTYS teknolojisinin daha uyumlu çalıştığı yazılım geliştirme ekosistemleri mevcuttur.
 - LAMP vs WINS gibi
- Bir yazılım projesinde kullanılacak ilişkisel VTYS teknolojisini belirlemeye ekibin kullandığı diğer teknolojiler, bilgi birikimi, lisanslama maliyeti vb. gibi kurumsal gereksinimler önemli rol oynar.

HAFTA -6-

MsSQL System Databases ve Analiz Servisleri

1. Veri Ambarı

- Veri ambarı, çeşitli kaynaklardan gelen anlamlı içeriklerin tutulduğu ve yönetildiği ortamdır.
 - Verilerin bir veya daha fazla veri kaynağından geldiği merkezi bir depo olarak düşünülebilir.
- Karmaşık ve bağımsız sistemlerden gelen verilere kolayca erişim ve analiz imkanı sağlar.
- BI (Business Intelligence | iş zekası | iş istihbaratı) ekiplerinin merkezde konumlandığı veri ambarı analitik ve raporlama işleri için kullanılır.
- Veri ambarı için tanımlanan 4 temel özellik vardır:
 - Özne odaklıdır:** Belirli bir konu veya işlev alanı hakkındaki verileri analiz edebilirler (örneğin satış verileri).
 - Bütünleşiktir:** Veri ambarları farklı kaynaklardan gelen farklı veri tipleri arasında tutarlılık sağlar.
 - Kalıcıdır:** Veri bir veri ambarına girdiğinde, kararlıdır ve değişmez.
 - Zamana dayalıdır:** Veri ambarı analizi, zaman içindeki değişime bakar.



1.2. Veri Ambarının Avantajları

- Farklı kaynaklardan gelen veriler için standardizasyon, kalite ve tutarlılık sağlar.
- Veri ambarı, kullanıcıların farklı kaynaklardaki kritik verilere tek bir ortamdan hızla erişmesine olanak tanır.
- Mevcut durumu bütünsel bir bakış açısıyla görme imkanı verir.
- Fırsatları ve riskleri değerlendirmeye yardımcı olarak şirketlere rekabet avantajı sağlar.
- Analitik olarak geçmiş veriye bakılarak öngörücü tahminlerde bulunulması sağlanır.

1.3. Dikkat Edilmesi Gerekenler

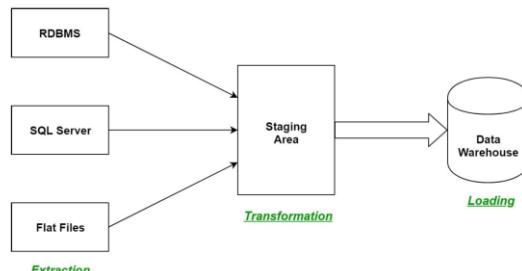
- Verilerin tutarlığını, doğruluğunu ve bütünlüğünü test etmek için bir plan belirlenmesi.
- Veri ambarının iyi tanımlanmış entegrasyon süreçlerinden geçtiğinden emin olunması.
- Verileri çıkarmak, temizlemek ve yüklemek için çok fazla zaman harcanmamalıdır.
- Veri ambarını tasarlarken doğru aracın kullanıldığından, yaşam döngüsüne bağlı kalındığından, veri uyuşmazlıklarına dikkat edildiğinden ve hatalardan ders çıkarıldığından emin olunmalıdır.
- Son kullanıcılar için bir eğitim planı hazırlanmalıdır.

1.4. Veri Ambarı Kullanan Sistemler

- Büyük miktarda veriye ihtiyaç duyan karar vericiler
- Birden çok veri kaynağından bilgi almak için özelleştirilmiş, karmaşık süreçler kullananlar
- Verilere erişmek için basit teknoloji isteyenler
- Karar vermek için sistematik bir yaklaşım isteyenler
- Raporlar, analizler veya grafikler için bir gereklilik olan büyük miktarda veri üzerinde hızlı performans isteyenler
- Veri ambarı, veri akışlarının ve gruplamaların 'gizli kalıplarını' keşfetmek isteyenler

1.5. Veri Entegrasyonu

- ETL: 4 temel DWH özelliğini sağlamak için kullanılan aşamaların kısaltmasıdır.



• Extraction (Çıkarma):

- Veri çıkışma, çeşitli veri kaynaklarından ilgili bilgilerin seçimi içeri.
- Bu, bir gönderme veya alma stratejisi olarak yürütülebilir.
- Veri çıkışma, bir alma stratejisinin parçası olarak gerçekleşse, veri kaynaklarına periyodik olarak DWH'a aktarmaları talimatı verilir.
- Bir çekme stratejisi ile DWH, veri çıkarmayı kendi başına tetikleyebilir.

• Transformation (Dönüşürme):

- Çıkarılan dosyalar bir dönüşümün parçası olarak ayarlanır ve hedef veri tabanı formatına uygun olarak çevrilir.

• Loading (Yükleme):

- Yükleme aşaması, dönüştürülen dosyaların DWH'in ilgili hedef veri tabanlarına kaydedilmesini içerir.

2. Entegrasyon Servisleri – SSIS

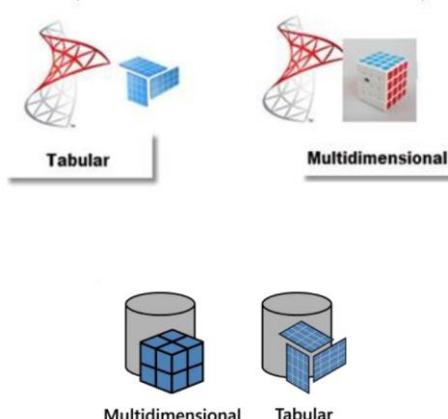
- MsSQL üzerinde veri ambarı oluşturmak için entegrasyon servisleri kullanılabilir:
 - **SSIS → SQL Server's Integration Services**
- SSIS servisleri veri ambarına veri aktarmak için gerekli adımları otomatikleştirir ve yönetimini kolaylaştırır.
- SSIS için kullanılan SSDT (SQL Server Data Tools) araçları Visual Studio üzerinden çalıştırılmaktadır.
- «**SSIS kurulum ve kullanımı demo»**

3. Analiz Servisleri – SSAS

- Veri ambarları uygulamalar arıcılığıyla doğrudan erişilerek BI ürünleri ortaya konabileceği gibi, veriler bir OLAP ortamına taşınarak onun üzerinde gerekli çözümlemeler yapılabilir.
- Böylece, çok boyutlu çözümlemelere daha etkin bir şekilde olanak sağlanır.
- MsSQL üzerinde OLAP Küpleri oluşturmak için analiz servisleri kullanılabilir:
 - **SSAS → SQL Server's Analysis Services**
- «**SSAS kurulum ve kullanımı demo»**

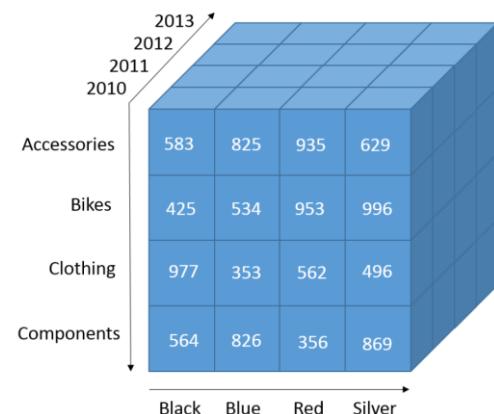
4. OLAP Modelleri

- Tipik olarak bir OLAP sisteminin OLTP sistemine avantajları
 - Hızlı cevap verebilme
 - Metadata tanımlı sorgulama
 - Çalışma sayfası (spreadsheet) formül kullanımı
- OLAP modeline göre verilerin sorgulanması için farklı sorgulama dilleri kullanılır.
 - **DAX**
 - **MDX**
- Tabular Model vs Multidimensional Model



4.1. Multidimensional Model

- SSAS ilk tanıtıldığında, onunla birlikte Çok Boyutlu Model de tanıtılmıştır.
- OLAP küpü olarak da bilinir.
- Verileri çok boyutlu yapılar halinde düzenler ve bu yapıda yer alan küp hücrelerinde kümeleme değerleri depolanır.
- Çok Boyutlu Model tarafından kullanılan teknoloji daha olgundur, kurumsal BI'nin geleneksel ihtiyaçlarını karşılar ve birçok BI yazılımı satıcısı tarafından benimsenir, ancak uygulanması oldukça zor olabilir.
- MDX (Multi-dimensional Expressions) sorgulama dilini kullanır.

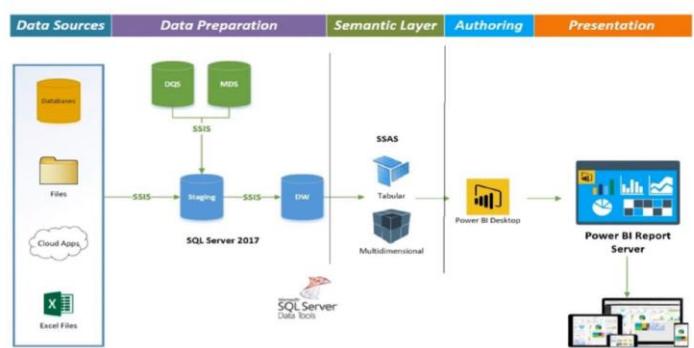


4.2. Tabular Model

- Tablo Modeli 2012'de tanıtılmıştır.
- Bellek İçi küp olarak da bilinir.
- Tablo Modeli, daha iyi veri sıkıştırması için sütunlu depolama kullanır ve sütunlara dayalı sorgular için çok daha hızlıdır.
- Bu modellerin geliştirilmesi ve yönetilmesi daha kolaydır.
- Bu modeli uygularken, tüm veriler bellekte depolandığı için çok fazla bellek ve çok hızlı CPU'lar gereklidir.
- DAX (Data Analysis Expressions) sorgulama dilini kullanır.

5. OLTP-SSIS-DWH-SSAS-OLAP

- BI Mimarisi

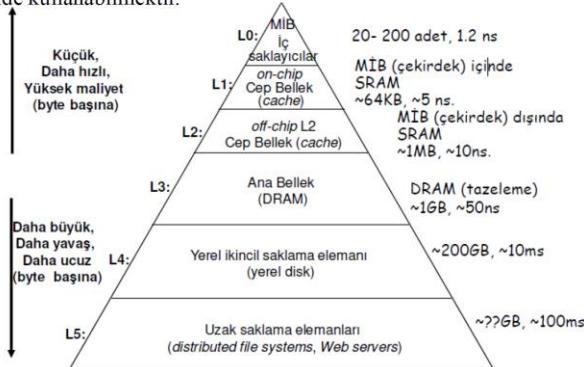


HAFTA -7-

Bellek Organizasyonu

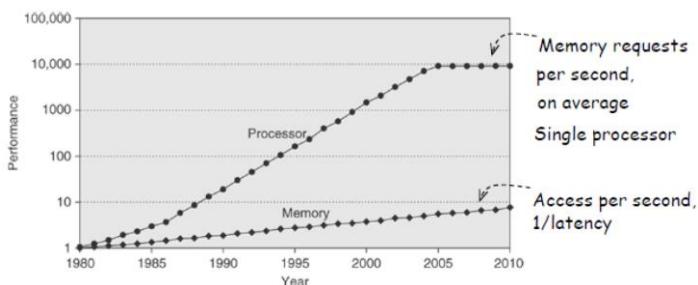
1. Bellek Organizasyonu

- Farklı hız, boyut ve fiyatlarında bellekler mevcuttur.
- Amaç, bellekleri toplam maliyeti düşük, performansı ise yüksek tutacak şekilde kullanabilmektir.



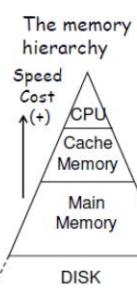
1.1. Bellek Teknolojileri

- Ana bellekler DRAM (dinamik RAM) teknolojisine dayalı olarak geliştirilmektedir.
- İşlemcilerin hızı, ana belleklerin hızından çok daha yüksektir.
- Bu nedenle ana bellek, MİB'in bant genişliği gereksinimlerini karşılayamaz.
- Çok çekirdekli işlemciler, bant genişliği gereksinimlerini artırır.
- 4 çekirdekli ve 3,2 GHz saat hızına sahip Intel i7, yaklaşık 409,6 GB / sn'lik toplam bant genişliğine ulaşabilir.
- DRAM ana belleğe giden en yüksek bant genişliği bunun yalnızca % 6'sıdır (25 GB / s).



1.2. Bellek Hiyerarşisi

- Farklı hız, boyut ve fiyatlarında bellekler mevcuttur.
 - Hızlı bellekler daha pahalıdır.
 - Ortalama hızı artırmak ve (aynı zamanda) bellek sisteminin maliyetini düşürmek için, çok düzeyli bellek hiyerarşisi şeklinde farklı bellek türleri kullanılır.
- Daha hızlı bir bellek (SRAM), yani önbellek (Cep bellek - cache memory), ana bellek ile MİB arasına yerleştirilir.



- Amaç:** Farklı belleklerden uygun miktarlarda kullanarak ve sık erişilen verileri daha hızlı olan belleğe yerleştirerek maliyeti düşük ve ortalama erişim süresi kısa bir bellek sistemi oluşturmaktır.
- Bellek hiperarşisindeki cep bellek ve ana bellek MİB'in program ve veriler için doğrudan erişebildiği sistem içi belleklerdir.
 - Ana bellek:** DRAM (Dynamic Memory) olarak üretilir. Kapasitesi daha büyük, birim maliyeti daha düşük, ancak daha yavaştır.
 - Cep bellek:** SRAM (Static Memory) olarak üretilir. İşlemci ile aynı devrede olabilir. Kapasitesi daha küçük, birim maliyeti daha yüksek, ancak daha hızlıdır.
- Ancak diskteki veriler önce ana belleğe (veya önbelleğe) alınmalıdır.
 - Veriler belleklere yerleştirilirken programlardaki başvuru yöreselliği (locality of reference) özelliğinden yararlanılır.

1.3. Başvuru Yöreselliği Prensibi

- Programlar, son zamanlarda kullandıkları komutları ve verileri yeniden kullanma eğilimindedir. (döngüler, diziler, vb.)
- Gözlem:** Çoğu program, yürütme süresinin % 90'ını kodun yalnızca % 10'unda geçirmektedir.
- Bir programın yakın gelecekte hangi komutları ve verileri kullanacağını, yakın geçmişteki erişimine bakarak, tahmin edebiliriz.
- 2 tip yöresellik mevcuttur:
 - Zamansal yöresellik (temporal locality):** Bir adres'e başvurulduktan sonra büyük olasılıkla bir süre sonra aynı adres'e tekrar başvurulur.
 - Uzaysal yöresellik (spatial locality):** Bir adres'e başvurulduktan sonra büyük olasılıkla bir süre sonra yakın adreslere tekrar başvurulur.

2. RAM (Random Access Memory)

- Ayırt edici özellikleri:
 - Hem veri okumak hem de yeni verileri kolay ve hızlı bir şekilde yazmak mümkündür.
 - Bu işlemler, elektrik sinyalleri kullanılarak gerçekleştirilir.
 - RAM'deki veriler uçucudur. Güç kesilirse veriler kaybolur.
 - Bu nedenle, RAM yalnızca geçici depolama olarak kullanılabilir.
- Bellek gecikme ölçümleri:
 - Erişim süresi:** Okuma talebi ile istenen kelimenin gelmesi arasındaki süre
 - Çevrim süresi:** İki farklı istek arasındaki minimum süre
 - DRAM ve SRAM olmak üzere 2 türü vardır:

2.1. DRAM (Dynamic RAM)

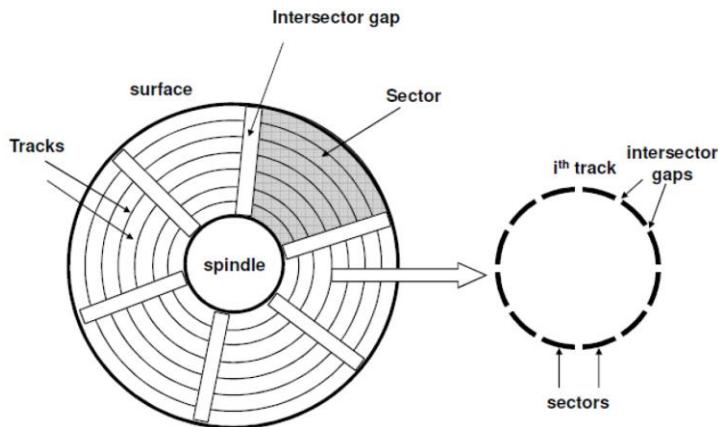
- Ana bellekler için kullanılır.
- Veriyi kapasitörlerde tutan hücrelerden oluşur.
- Kapasitörlerin boşalma eğiliminden dolayı periyodik olarak yeniden şarj olması gereklidir. (ortalama 8 ms'de bir)
- Dinamik olarak adlandırılır, çünkü depolanan yük, sürekli uygulanan güçle bile sızar.
- Hafıza yenileme sırasında kullanılamaz (gecikmeye sebep olur).
- Okunduktan sonra yeniden yazılmalıdır. Okumak bilgiyi yok eder (gecikmeye sebep olur).
- Ucuz ve yoğundur: bir transistör ile bir bit saklanabilir.
- SDRAM (Synchronous DRAM) ve DDRAM (Double data rate DRAM) gibi geliştirilmiş versiyonları mevcuttur.

2.2. SRAM (Static RAM)

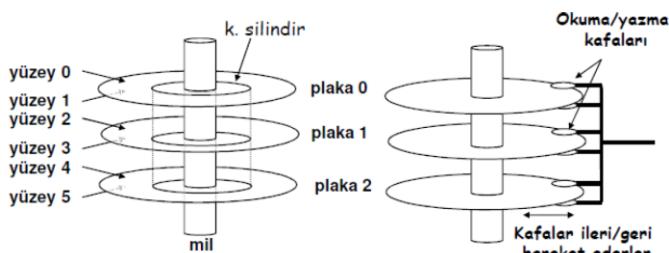
- Cep bellekler için kullanılır.
- SRAM, bitleri depolamak için flip-flops (veya latches) kullanır.
- 1 biti saklamak için 6 transistör gerekir.
- Hafıza yenilemeye gereklidir.
- DRAM'e göre maliyetlidir.
- DRAM'e göre gecikmesi çok daha azdır.

3. Manyetik Disk

- Harici Bellek - External Memory
- Her plakada iki yüzey bulunur.
- Her yüzeyde eş merkezli halkalar (izler-tracks) bulunur.
 - Veri izlere yazılır/okunur.
- Her izde boşluklarla (gaps) ayrılan sektörler bulunur.
 - Veri sektörlerden/sektörlerle aktarılır.
 - Her izde yüzlerce sektör bulunur.
 - Güncel sistemlerin çoğunda, 512 bayt ile sabit uzunlukta sektörler kullanılır.
 - Bitişik sektörler, sektörler arası boşluklarla ayrılır.



- Aynı hızadaki izler bir silindir oluşturur.



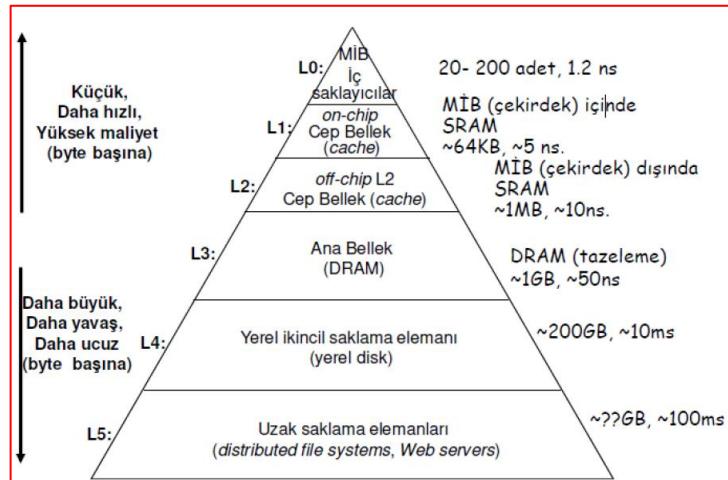
- 512 byte/sektör
 - 300 sektör/iz (ortalama)
 - 20,000 iz/yüzey
 - 2 yüzey/plaka
 - 5 plaka/disk
- Kapasite = $512 \times 300 \times 20000 \times 2 \times 5$
= 30,720,000,000 byte
~ 28.6 GB

3.1. Manyetik Disk Erişim Süresi

- Bir diskin ortalama erişim süresi (T_a) üç bileşenden oluşur:
 - Konumlandırma Süresi (T_s)
 - Dönüş Gecikmesi (T_r)
 - Aktarım Süresi (T_t)
$$(T_a) = (T_s) + (T_r) + (T_t)$$
- **Ortalama konumlanma süresi (Seek time) T_s :**
 - Okuma/yazma kafasının ilgili ize konumlanması için geçen süredir.
 - Yaklaşık 9ms (3-15ms) sürer.
- **Ortalama dönüş gecikmesi (Rotational latency) T_r :**
 - Okuma/yazma kafasının, iz içinde ilgili sektörün başına konumlanması için geçen süredir.
 - Kafa diskte ilgili izin üstüne konumlandıktan sonra gerekli olan sektörün gelmesi için plakanın dönmesini bekler.
 - Bu bekleme süresi ortalama olarak diskin bir turunu tamamlaması için gerekli olan sürenin yarısı kadardır:
 - $T_r = \frac{1}{2} r$, r: Diskin bir tur dönüş süresi (saniye)
- **Aktarım Süresi (Transfer time) T_t :**
 - **İki farklı şekilde ifade edilebilir:**
 - Bir sektörü aktarmak için geçen süre
 - Belli miktarda byte aktarmak için geçen süre
 - **Bir sektörü okumak için geçen süre T_{ts}**
 - $T_{ts} = \frac{1}{\text{ort.sektör/iz RPM}} \frac{60}{\text{saniye}}$ (saniye)
 - Bir manyetik disk için:
 - Disk dönüş hızı = 7200 RPM
 - Ortalama konumlanma süresi = 9 ms,
 - Bir izdeki ortalama sektör sayısı = 400
 - Buna göre:
 - Dönüş gecikmesi = $1/2 \times (60 \text{ s}/7200 \text{ RPM}) \times 1000 \text{ ms/s} = 4 \text{ ms}$
 - Aktarım süresi = $60/7200 \text{ RPM} \times 1/400 \text{ sektör/iz} \times 1000 \text{ ms/sec} = 0.02 \text{ ms}$
 - Erişim süresi = 9 ms + 4 ms + 0.02 ms
 - Erişim süresinin belirleyici bileşenleri konumlanma süresi ve dönüş gecikmesidir.
 - Sektörün ilk bitine ulaşmak zaman kaybetir.
 - Disk SRAM'dan 40,000 kat daha yavaştır.
 - Disk DRAM'dan 2,500 kat daha yavaştır.
 - Veri aktarım hızları: Standart masaüstü bir bilgisayardaki 7200 RPM SATA disk
 - Arabirim tampon belleği – fiziksel plakalar arası: 1030 Mbit/s
 - Bilgisayar belleği – arabirim tampon belleği arası: 300 Mbyte/s

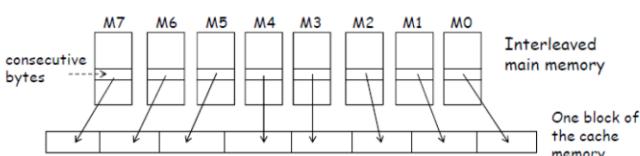
4. Cep Bellek

- Cep Bellek – Ana Bellek Etkileşimleri
- Cep Bellek – Ana Bellek Bağlantıları
- Ana Bellek - İkincil Bellek Etkileşimi
- Ana Bellek - İkincil Bellek Bağlantıları



4.1. Çağrışızlı Bellek

- **Çalışma mantığı:**
 - Sık başvurulan adreslerdeki verilerin cep bellekte tutulması amaçlanır.
 - **Vuru (Hit):** Başvurulan adresdeki verinin cep bellekte bulunması.
 - **Iska (Miss):** Başvurulan adresdeki verinin cep bellekte bulunmaması.
- **Örnek:**
 - Cep bellek erişim süresi: 20 ns.
 - Ana bellek erişim süresi: 100 ns.
 - Vuru oranı: H=0.9, %90 vuru var.
- **Ortalama bellek erişim süresi:**
 - $T_a = 0.9 \cdot 20 + 0.1 \cdot 100 = 28$ ns
- **Blok transfer**
 - Önbellek bir iska oluştugunda (istenen öğe önbelekte yoksa), istenen öğeyi içeren bir öğe bloğu ana bellekten önbellege getirilir.
 - Sebebi uzaysal yöresellik ilkesidir.
- **Blok transfer gecikmeyi uzatmaz mı?**
 - Ana bellek ve ön bellek bellekleri arasındaki blok aktarım süresini azaltmak için bellek serpiştirme (memory interleaving) yöntemi kullanılır.
 - Veriler, ardışık bellek adresleri ardışık bellek modüllerinde olacak şekilde, farklı bellek modüllerinde saklanır.



- Yer değiştirme yöntemleri
- Önbellek dolduguunda, ana bellekten yeni gelen blokla değiştirilecek olan önbellek bloğunu seçmek için bir değiştirme algoritması uygulanmalıdır.
- Bunun için farklı değiştirme teknikleri vardır. En yaygın teknikler:
 - **FIFO (First In First Out):** Önbelekte en uzun zamandır bulunan blok değiştirilir..
 - **LRU (Least Recently Used):** Ön bellekte en az kullanılan (en uzun süredir kullanılmayan) blok değiştirilir.
 - Ön bellekteki her blok için kullanım geçmişi tutulur.
 - Yaşlandırma sayacı adı verilen yapı eklenir.
- Önbellek işlemleri, Önbellek Bellek Denetleyici veya Önbellek Bellek Yönetim Birimi adı verilen bir donanım birimi tarafından gerçekleştirilir.

4.2. Cep Bellek Erişim Yöntemleri

- Aşağıdaki soruları cep bellek erişim yöntemleri ile yanıtlamak mümkündür.
 - Önbellette ana belleğin hangi içerikleri var?
 - Stk erişilen veriler, önbellegin hangi konumunda yer almaktadır?
- Cep bellek erişim yöntemleri:
 - Çağrışızlı (Full Associative) Erişim Yöntemi
 - Doğrudan Erişim (Direct Mapping)
 - Kümeli Çağrışızlı (Set Associative) Erişim

4.3. CB – AB Etkileşimleri

1. **Okuma - Vuru:** Veri cep bellekten okunur.
2. **Okuma - Iska:**
 - **Doğrudan Okuma (Read Through-RT):** Veri ana bellekten MİB'e okunurken aynı anda cebe de getirilir. Cep belleğe ve ana belleğe paralel erişilir.
 - **Dolaylı Okuma (No Read Through-NRT):** Veri ana bellekten önce cep belleğe getirilir, sonra MİB cep belleği okur.
3. **Yazma - Iska:**
 - **Cebe Yükleyerek Yazma (Write Allocate-WA):** Ana bellekte değiştirilen blok aynı zamanda cep belleğe de getirilir.
 - **Cebe Yüklemeden Yazma (NWA):** Veri sadece ana belleğe yazılır. Daha sonra eğer bu veri okunmak istenirse iska olacağından ilgili blok cep belleğe getirilir.

4. **Yazma - Vuru:** Veri cep bellekten okunur.

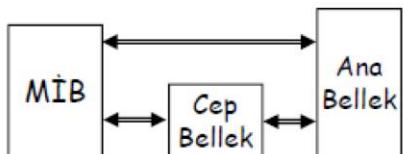
- **Doğrudan Yazma (Write Through-WT):** Her yazma çevriminde veri hem cep belleğe hem de ana belleğe yazılır. Erişim süresi artar, ancak önbellek ile ana arasında tutarlılık sağlanır.
- **Sonradan Yazma (Write Back-WB):** Veriler sadece cep belleğe yazılır. Değişiklikle uğrayan blok ancak cep bellekten kaldırılırken ana belleğe yazılır.
 - Basit sonradan yazma (Simple Write Back – SWB): Cep bellekten çıkartılan her blok ana belleğe yazılır. Zaman kaybettiir.
 - Bayraklı sonradan yazma (Flagged Write Back – FWB): Sadece değişiklikle uğramış olan bloklar cep bellekten çıkartılırken ana belleğe yazılır. Değişen blokları belirleyebilmek için takı belleğinde "geçerlilik" bitleri ile birlikte bir de "kirleme" (dirty) biti bulunur.

4.3. CB – AB Bağlantıları

- MİB-cep bellek-ana bellek bağlantıları iki farklı şekilde yapılabilir:
 1. Paralel Bağlantı
 2. Seri Bağlantı

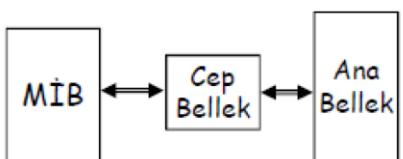
4.3.1. Paralel CB – AB Bağlantısı

- Read Through (RT): Blok ana bellekten cep belleğe aktarılırken gerekli veri aynı zamanda MİB tarafından paralel olarak okunabilir.
- Load Through (LT): MİB cep belleğe veri yazarken veri aynı zamanda paralel olarak ana belleğe de yazılır.
- Paralel yapı özellikle Doğrudan Yazma (Write Through-WT) yöntemi için uygundur.
- Sonradan Yazma (Write Back-WB) yöntemi ile de kullanılabilir.



4.3.2. Seri CB – AB Bağlantısı

- No Read Through (NRT): Blok önce ana bellekten cep belleğe aktarılır ardından gerekli veri MİB tarafından cep bellekten okunur.
- No Load Through (NLT): MİB veriyi cep belleğe yazar, ardından veri cep bellekten ana belleğe aktarılır.
- Seri yapı Sonradan Yazma (Write Back-WB) yöntemi için uygundur.

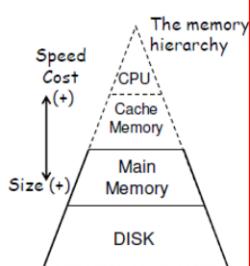


5. Ana Bellek - İkincil Bellek Etkileşimi

- Kullanıcılar/programlara fiziksel belleğin (ana bellek) boyutundan bağımsız olarak büyük boyutta ve lineer (sürekli) bellek alanı sağlamayı amaçlar.
- Bellek blokları üzerinde erişim denetimi (güvenlik/koruma) sağlamak üzere kullanılır.
- Çok kullanıcılı, çok programlı sistemlerde ortak programların/verilerin kullanılmasını sağlamayı hedefler.

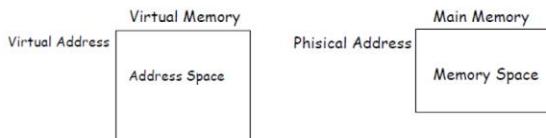
Cep bellek ile ana bellek arasındaki benzer bir ilişki ana bellek ile disk arasında kurulur. (yöresellik)

- Sanal belleği kullanarak, bilgisayarınız gerçekte sahip olduğundan daha fazla belleği adresler ve fazlalığı tutmak için sabit diski kullanır.
- Programlar ve veriler diskte saklanır; gerekli veriler ana belleğe getirilir (sayfalama).

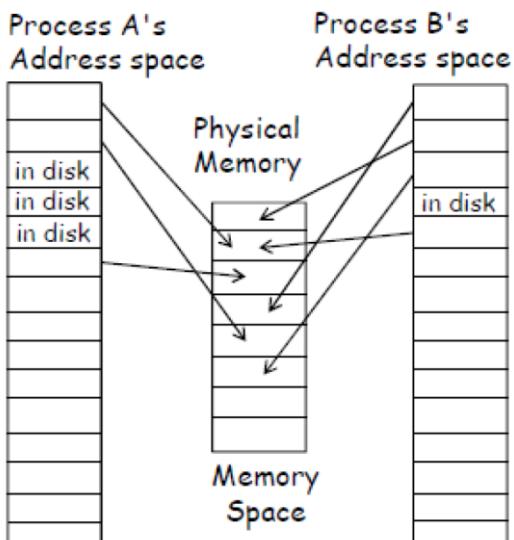


- Programlar sanal belleğe göre (doğrusal ve büyük) yazılır ve adresler sanal belleğe göre oluşturulur.
- Programların ürettiği adrese görüntü (sanal) adres veya mantıksal adres denir. Görüntü adreslerin kümesine (yani görüntü belleğin adres alanına) adres uzayı (address space) denir.

- Ana bellek adreslerine fiziksel adres denir. Bu kümeye de (ana belleğin adres alına) bellek uzayı (memory space) denir.



- Programların adresleri fiziksel belleğe dağılmıştır ve büyük olasılıkla ardışık değildir (kullanılmayan veriler disktedir).
- İşletim sistemi, gerçek adres - sanal adres eşlemesi için bir tablo tutar; diskte veya fiziksel bellekte (hangi adreste) olabilir.
- Prosesler bu durumun farkında değildir.
- Her process sanki tüm makineye (büyük ve doğrusal bir bellek) sahipmiş gibi adresler üretir.
- Programcı ve derleyicinin işi basitleştirilmiştir, çünkü bir program oluşturmak için donanım veya bellek organizasyonunun ayrıntıları gerekmektedir.
- Program, fiziksel belleğin özelliklerinden bağımsız olarak derlenebilir.



5.1. Sayfalı Dönüşüm

Demand Paging:

- Bir program yürütülmeye başladığında, işletim sistemi sayfaların küçük bir bölümünü diskiten ana belleğe kopyalar.
- Bu genellikle programdaki ilk komutlara ilişiği sayfayı ve programın başlangıçta ihtiyaç duyduğu küçük miktarda veriyi içerir.
- Daha sonra, daha fazla komut veya veriye ihtiyaç duyulduğça, işletim sistemi talep üzerine sayfaları diksten getirir.

Paylaşılan Bellek (Shared Memory):

- Normal şartlarda iki processin adres alanları birbirinden korunur.
- Paylaşılan bellek mekanizması ile, processler arası iletişim sağlamak için iki adres alan bazı noktalarda kesişebilir.
- Paylaşılan sayfalar aynı fiziksel sayfaya eşleşir.
- Aynı sayfa çerçeve numarası, bir sayfayı paylaşan iki processin sayfa tablolarına yerleştirilir.

Sayfa Değiştirme (Page replacement):

- Ana bellek doluyken ana bellekte olmayan yeni bir sayfaya başvurulursa bir yer değiştirme algoritmasına göre ana bellekteki bir bloğun boşaltılarak yerine yeni sayfanın yerleştirilmesi gereklidir.
- LRU (Least Recently Used) yönteminde ana bellekte olan sayflara (bloklara) yerleştirme sayacı atanır.

• Sayfa değiştirme (Page replacement):

- Örnek: Başvuru zinciri sayfa numaraları: 0, 1, 2, 3, 0, 3, 4, 5
- Ana bellekte 4 blok yer var. Bu durumda 2 bitlik sayaçlar yeterli olur.

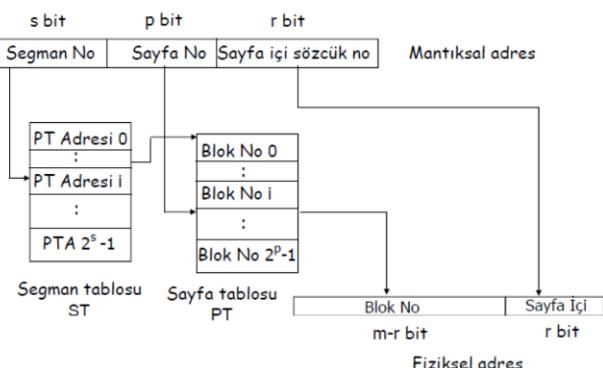
Başvurulan sayfa		1	2	3	0	3	4	5	
Sayaç	0	00 0	00 1	10 0	11 0	00 0	01 0	10 0	11 0
		00 1	01 1	10 1	11 1	01 1	11 1	00 4	01 4
		00 2	01 2	10 2	11 2	00 2	10 2	11 2	00 5
		00 3	01 3	10 3	11 3	00 3	01 3	10 3	

• Kırıntılanma (Fragmentation):

- Tüm sayfalar aynı boyutta olduğundan programlar ya da veriler belli sayıda sayfayı tam olarak dolduramaz.
- Örneğin sayfalar 1K sözcük boyutundaysa $5K+1$ sözcüklük bir program 6 adet sayfa yer kaplayacaktır ancak son sayfanın sadece 1 sözcüklük kısmı anlamlı olarak kullanılacaktır.
- Son sayfanın belli bir yeri kullanılmaz ancak yer değiştirme işlemlerinde sayfanın tamamı aktarılır.
- Sayfalar dönüşümde sayfalar daha küçük parçalara bölünmez; her zaman bir bütün olarak aktarılırlar.
- Bir sayfadaki bazı alanların kullanılamaması sorununa dahili kırıntılama (inner fragmentation) denir.
- Diğer taraftan bazı programlar ve veriler görüntü belleğin tamamına gerek duymazlar. Bu nedenle 2^p satırlı sayfa tablosunun bazı satırları hiç kullanılmaz.

5.2. Segmanlı Sayfalı Dönüşüm

- Mantıksal adres uzayı segmanlara bölünür.
- Her segman değişik sayıda sayfadan oluşabilir.
- Bir segman, mantıksal bir program parçası veya veridir.
 - Bütün bir program, bir alt program, dizi, matris bir segman oluşturabilir.
- Bu yöntem ile mantıksal bir bütün oluşturan program parçaları ve veriler (segmanlar) üzerinde erişim ve paylaşım denetimi oluşturulabilir.
- Mantıksal adres üç alandan oluşur:
- Segman tablosunda her segmanla ilgili erişim denetimi, uzunluk gibi bilgiler de bulunur.



6. Paylaşılan Bellek Sistemleri

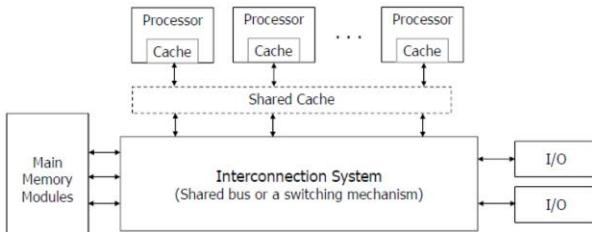
- Sıkı bağlı paylaşılan bellek sistemleri
 - Uniform memory access (UMA)
 - Nonuniform memory access (NUMA)
- Zayıf bağlı dağıtılmış bellek sistemleri
- Veri tutarlılığı için çözümler
 - Yazılım tabanlı çözümler
 - Donanım tabanlı çözümler

6.1. Sıkı Bağlı Sistemler

- Paylaşılan bellek sistemleri ikiye ayrılır:
 1. Uniform memory access (UMA) multiprocessors:
 - Symmetric multiprocessor systems-SMP olarak da adlandırılır.
 - Bellek erişim süresi, hangi işlemcinin hangi bellek sözcüğüne eriştiğine bakılmaksızın her işlemci için yaklaşık olarak aynıdır (simetrik).
 2. Nonuniform memory access (NUMA) multiprocessors:
 - Bir işlemci, bir belleğe fizikal olarak daha yakınsa, belleğe daha hızlı erişebilir.
- **UMA: Özellikleri**
 - Tek adres uzayı (paylaşılan bellek), ve tek işletim sistemi kullanılır.
 - Benzer (aynı) yeteneklere sahip iki veya daha fazla işlemci vardır.
 - Bu işlemciler aynı ana belleği ve G/C olanaklarını paylaşır ve bir veriyolu veya çapraz anahtar bağlantısı ile birbirine bağlanırlar.
 - Bellek erişim süresi her işlemci için yaklaşık olarak aynıdır (simetrik).
 - Tüm işlemciler aynı işlevleri (simetrik) gerçekleştirebilir.
- **UMA: Potansiyel faydaları**
 - Performans, tek işlemcili sistemlere göre, daha yüksektir.
 - Erişilebilirlik daha fazladır. Bu işlemciler aynı ana belleği ve G/C olanaklarını paylaşır ve bir veri yolu veya çapraz anahtar bağlantısı ile birbirine bağlanırlar.
 - Tüm işlemciler aynı işlevleri yerine getirebildiğinden, tek bir işlemcinin arızası makineyi durdurmaz.
 - Bir kullanıcı, ek bir işlemci ekleyerek bir sistemin performansını artırabilir.
 - Fakat, çok fazla işlemci eklendikçe, veri yoluna erişim rekabeti performansa düşüre neden olur.

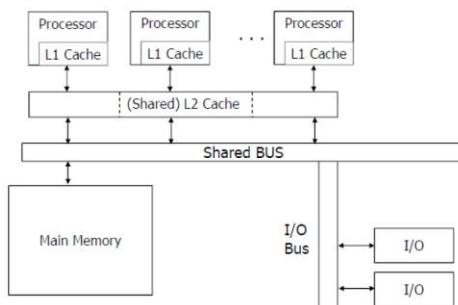
• UMA: Şematik Organizasyon-1

- Her işlemci bir kontrol birimi, ALU, saklayıcı ve tipik olarak bir veya daha fazla önbellek seviyesi içerir.
- Bellek, ayrı bellek bloklarına birden çok eşzamanlı erişim için uygun şekilde organize edilir (interleaved or multiport).
- Ara bağlantı sistemi farklı şekillerde (paylaşımlı veri yolu, çapraz anahtar bağlantısı) tasarlanabilir.



• UMA: Şematik Organizasyon-2 (ortak veri yolu ile)

- Zaman paylaşımı gereklidir.
- Bir işlemci veri yolunu kontrol ederken, diğer işlemciler kilitlenir ve gerekirse, veri yolu erişimi elde edilene kadar çalışmayı askıya almalıdır.



• UMA: Avantajları

• Basitlik (Simplicity):

- Her işlemcinin fiziksel arabirimini, adreslemesi, ve zaman paylaşımı mantığı tek işlemcili bir sistemdekiyle aynı kalır.

• Esneklik (Flexibility):

- Sistemi daha fazla işlemci ekleyerek genişletmek genellikle kolaydır. (Ancak veri yolu sınırları vardır)

• Güvenilirlik (Reliability):

- Veriyolu esasen pasif bir ortamdır ve herhangi bir takılı aygıtın arızası tüm sistemin arızalanmasına neden olmamalıdır.

• UMA: Dezavantajları

• Performans (Performance):

- Tüm veri erişimi işlemleri için ortak veri yolu kullanılmaktadır. Bu da zaman paylaşımı çalışma gerektirir ve performansı düşürür.

• Çözüm olarak her işlemciye yerel bir cep bellek kullanımı önerilebilir.

- Bu durumda sistem performansı artacaktır. Ancak bu kez, veri tutarlılığına ilişkin başka problemler oluşacaktır.

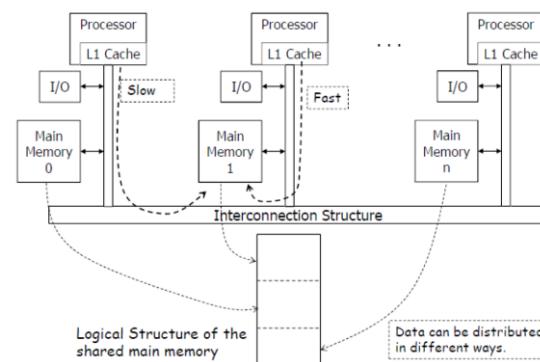
• NUMA:

- UMA sistemlerinde ortak veri yolu performans için bir darboğazdır. (bottleneck)
 - Bu nedenle sisteme eklenebilecek işlemci sayısı sınırlıdır.
 - Dağıtılmış bellek sistemleri bu duruma bir çözüm olabilir, ancak bu sistemlerde işlemciler ortak bir bellek kullanmaz.
- NUMA sistemleri, paylaşılan belleğin avantajlarını korurken büyük ölçekli çoklu işlemler için tasarlanmıştır.

• NUMA: Özellikleri

- Tek adres uzayı (paylaşılan bellek), ve tek işletim sistemi kullanılır.
- Paylaşılan bellek fizikselt olarak CPU'lara dağıtıltır.
 - Bu sistemlere ayrıca dağıtılmış paylaşımı bellek sistemleri denir.
- Bir CPU, kendi bellek modülüne diğer modüllerden daha hızlı erişebilir.
- İşlemler ve veriler sistemde dağıtılabiliyorsa ve CPU'lar çoğunlukla ana bellek modüllerine (veya yerel cep belleklerine) daha fazla erişiyorlarsa, sistemin performansı artar.
- Programların (verilerin) mekansal/uzaysal yerelligi performansı etkiler.

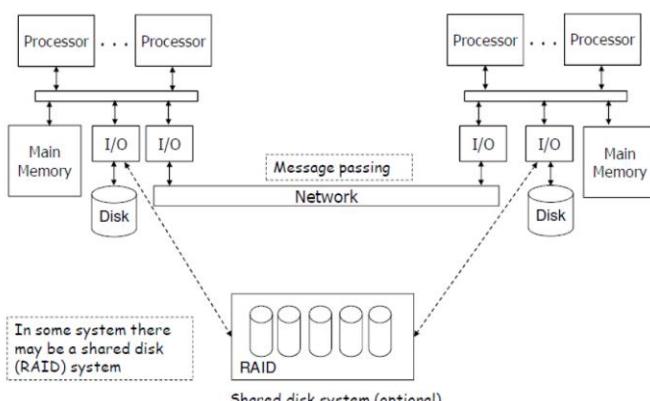
• NUMA: Şematik Organizasyon



6.2. Zayıf Bağlı Sistemler

- Her işlemcinin kendi fiziksel adres alanı vardır.
- İşlemciler mesaj aktarımı ile birbirleriyle haberleşirler.
- Kümeler (Clusters) bu sistemlere örnek olarak verilebilir.
- Kümeler, standart ağ ekipmanları üzerinden birbirine bağlanan bilgisayar sistemleridir.
- Daha büyük ölçeklerde bulut sistemleri için kullanılabilir.
- Avantajları:
 - Ölçülebilirlik (Scalability): Bir kümeye, her biri çok işlemcili olarca, yüzlerce veya binlerce makine olabilir.
 - Erişilebilirlik (High availability): Bir kümeye her düğüm bağımsız bir bilgisayardır, bu nedenle bir düğümün başarısızlığı hizmet kaybına yol açmaz.
 - Fiyat/Performans Oranı (Superior price/performance): Uygun maliyetlere büyük bir bilgi işlem gücüne sahip kümeler oluşturmak mümkündür.

- Dezavantajları:
 - İşlemciler arası haberleşme yükü fazladır.
- Şematik Organizasyon



7. Veri Tutarlılığı için Stratejiler

- Ortalama erişim süresini (ve gerekli bant genişliğini) azaltmak için cep bellekler kullanılır.
- Paylaşılan verileri cep belleklere almak veri tutarsızlığına sebebiyet vermektedir.
- Aynı verinin birden çok kopyası aynı anda farklı işlemcilere ait cep belleklerde bulunabilir ve işlemcilerin kendi kopyalarını serbestçe güncellemelerine izin verilirse, tutarsız bir bellek görünümü ortaya çıkar.
- Çözüm için önerilen yaklaşımlar 2 temel başlık altında ele alınabilir:

1. Yazılım Çözümleri
2. Donanım Çözümleri

7.1. Yazılım Çözümleri

- Ek bir donanıma ihtiyaç duyulmaz.
- Derleyici ve işletim sistemi sorunu derleme sırasında çözer.
- Ancak bu çözüm tutucu kararlar içerir ve bu da verimsiz önbellek kullanımına yol açar.
 - Derleyici tabanlı mekanizmalar, hangi verilerin önbelleğe alma için (ne zaman) güvenli olmayıabileceğini belirlemek için kod üzerinde bir analiz gerçekleştirir ve bu öğeleri işaretler.
 - İşletim sistemi (veya donanım) daha sonra bu öğelerin önbelleğe alınmasını engeller.
- En basit yaklaşım, paylaşılan veri değişkenlerinin önbelleğe alınmasını önlemektir.
 - Etkinliği azaltır.

7.2. Donanım Çözümleri

- Donanım üzerinde belirli bir protokole göre denetim yapan birimler vardır.
 - Directory Protocols
 - Snoopy Protocols
 - MESI Protocol

7.2.1 Directory Protocols

- Ana bellekte bir sözlük tutan merkezi bir denetleyici vardır.
- Bu sözlük, hangi işlemcinin özel önbelleğinde hangi satırların (çerçevelerin) bir kopyasına sahip olduğu hakkında bilgi içerir.
- Avantajları:
 - Birden çok veri yolu veya diğer bazı karmaşık ara bağlantı şemalarını içeren büyük ölçekli sistemlerde etkilidir.
- Dezavantajları:
 - Merkezi denetleyici bir darboğazdır. Tüm istekler tek bir denetleyiciye yönlendirilir.
 - Yerel önbellek denetleyicileri ile merkezi denetleyici arasındaki iletişim ek yük getirmektedir.

7.2.2 Snoopy Protocols

- Önbellek tutarlığını sürdürme sorumluluğu, çok işlemcili sistemdeki tüm önbellek denetleyicileri arasında dağıtılr.
- Paylaşılan bir önbellek çerçevesi güncellendiğinde, yerel denetleyici bu işlemi bir yayın mekanizması tarafından diğer tüm önbelleklere duyurur.
- Her bir önbellek denetleyicisi, bu yayınlanan bildirimleri gözlemelemek için ağ üzerinde "gözleme" yapabilir ve buna göre tepki verebilir (örneğin, kopyayı geçersiz kılmaya).
- Snoopy protokolleri, veri yolu tabanlı çoklu işlemciler için uygundur, çünkü paylaşılan veri yolu, yayın ve gözetleme için basit bir mekanizma sağlar.
- Yerel önbelleklerin, paylaşılan veri yolundaki trafiği azaltmak için kullanıldığı unutulmamalıdır. Bu nedenle, paylaşılan veri yolu üzerindeki trafiğin yayın ve gözetleme yoluyla artırılmamasına özen gösterilmelidir.

1. Write-invalidate protocol (geçersiz kıılma):
 - İşlemcilerden biri kendi önbelleğindeki bir satırı yazmak istediği, bir "geçersiz kıılma" mesajı gönderir.
 - Tüm önbellek denetleyicileri, ilgili önbellek satırı kopyalarını geçersiz kılar.
 - Veriyi yazacak işlemci kendi kopyasına yazabilir.
 - Bu yöntemde veri ana belleğe de yazılır.
 - Başka bir işlemci bu işlemler sırasında veriyi okumaya çalışırsa iska oluşur ve veriyi ana bellekten okur.
2. Write-update protocol (güncelleme):
 - İşlemcilerden biri paylaşılan bir veriyi güncellemek istediği, yeni veriyi diğer tüm işlemcilere yayınlar, böylece her işlemci özel önbelleklerini güncelleyebilir.
 - Aynı zamanda merkezi CPU, önbellekte kendi kopyasını günceller.

7.2.3 MESI Protocol

- M (Modified):
 - Bu önbellekteki çerçeve değiştirilmiştir ve ana bellekten farklıdır. Bu çerçeve yalnızca bu önbellekte geçerlidir.
- E (Exclusive):
 - Önbellekteki çerçeve, ana bellektekiyle aynıdır ve başka herhangi bir önbellekte mevcut değildir.
- S (Shared):
 - Önbellekteki çerçeve, ana bellektekiyle aynıdır ve başka bir önbellekte bulunmaktadır.
- I (Invalid):
 - Önbellekteki çerçeve geçerli değildir.

8. RAID Sistemlerine Giriş

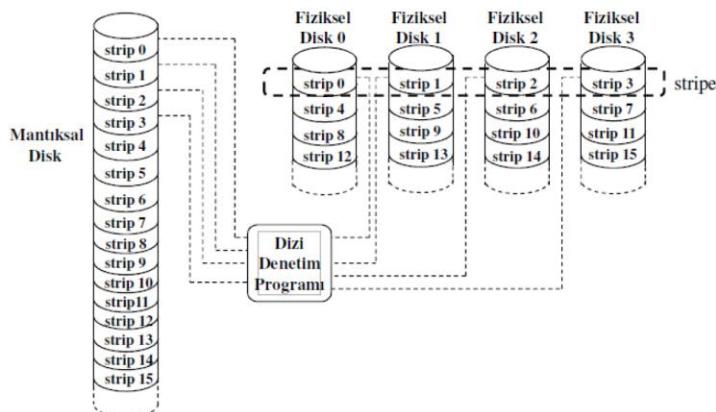
- RAID (Redundant Array of Independent/Inexpensive Disks)
- Veriler paralel çalışan birden çok diske dağıtılr.
- **Amaç:** Performansı ve güvenilirliği artırmak.
 - Paralel ve bağımsız diskler performansı artırır.
 - Fazlalık bilgi, hataları sezmek ve düzeltmek için kullanılır.
- **Düzyeler:** RAID 0 – RAID 6
 - 7 farklı düzey ve bunların bileşiminden oluşan bileşik düzeyler vardır.
 - Ayrıca tam bir RAID düzeyi olmamakla birlikte konuyu anlamak için kullanılan RAID 0 vardır.
 - En çok RAID 3 ve 5 kullanılır.

Ortak özellikler:

- Birden fazla fiziksel disk vardır. İşletim sistemleri bunları bir bütün, tek bir mantıksal disk olarak görür/gösterir.
- Mantıksal olarak peş peşe gelen veriler belli büyülükteki bloklar (şerit – strip) halinde farklı fiziksel disklere paralel olarak yerleştirilirler.
- Fazlalık olarak eşlik bilgileri (parity) yerleştirilir. Disklerden biri fiziksel olarak bozulduğunda bu diskteki bilgi tekrar oluşturulabilir.

8.1. RAID 0

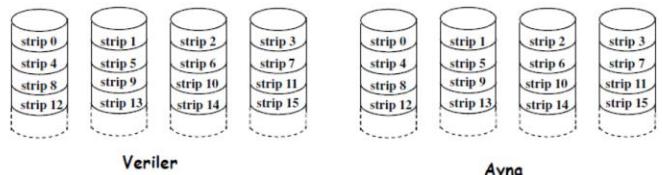
- Fazlalık ve eşlik biti yoktur.
 - Hatalar düzeltilemez.
 - Tam bir RAID sistemi değildir.
- Veriler paralel erişilebilen fiziksel disklere dağıtılr, performans artımı sağlanır.
- Strip (şerit): Belli miktarda veriyi ifade eder. Bir sektörden bir kaç MB'a kadar değişebilir.
- Stripe (bant): Farklı disklerdeki mantıksal olarak birbirini izleyen şeritlerin oluşturduğu yapıdır.



- Şerit (strip) boylarınının performansa etkisi:
- Eğer mantıksal olarak peş peşe gelen büyük bloklara erişiliyorsa paralel erişim sağlayabilmek için mantıksal olarak birbirini izleyen verilerin mümkün olduğu kadar farklı fiziksel disklere dağıtılması istenir.
 - Bu durumda küçük şeritler performansı artırır.
- Eğer sık G/Ç istekleri varsa (transaction-oriented environment) ve küçük bloklara erişiliyorsa (örneğin kısa ve sık veri tabanı sorguları);
 - Erişimlerin farklı disklere olması için büyük şeritler tercih edilir.

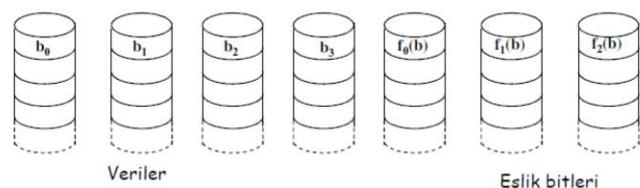
8.2. RAID 1

- Veriler aynaların (mirroring).
 - Her veri iki ayrı diske yazılr.
- Yazarken veri iki diske de paralel yazılr.
 - Bu durumda daha yavaş olan disk beklenir.
- Bir disk bozulduğunda veri diğerinden alınır.
 - Fiziksel olarak hangi diskin bozulduğu bellidir.
- Fazla disk kullandığı için pahali bir yöntemdir.



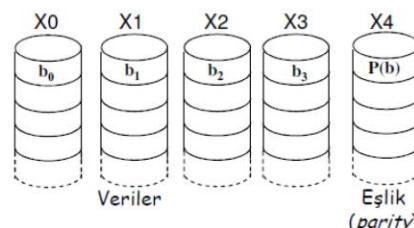
8.3. RAID 2

- Hata sezme/düzelme için eşlik bitleri eklenir.
- Hata sezme için Hamming kodu kullanılır.
 - Hamming kodlaması (sonraki bölümde açıklanacak) belleklerde ve iletişimde hata sezme/düzelme için kullanılır.
- RAID 2'de diskler senkron çalışır, tüm disklerdeki kafalar aynı yere konumlanır.
- Küçük stripler (1 sözcük) kullanılır.
 - Sürekli ve büyük blok erişimlerinde avantajlıdır.
- Aşağıdaki şekilde 4 bitlik veriye hata sezme için 3 bitlik eşlik eklenmiştir.
- Disklerin fiziksel olarak bozulduğunu anlamak mümkün olduğu için disklerde Hamming kodlaması gereksiz yere karmaşıklığı artırmakta ve maliyeti yükseltmektedir.



8.4. RAID 3

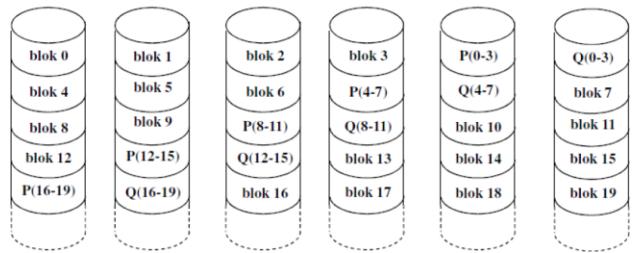
- RAID 2'de olduğu gibi diskler senkron çalışır, tüm disklerdeki kafalar aynı yere konumlanır.
- Küçük stripler kullanılır.
 - Sürekli ve büyük blok erişimlerinde avantajlıdır.
- Hata sezme/düzelme için daha basit bir kodlama kullanılır ve tek eşlik biti eklenir.



- Her yazmada eşlik diskine de erişmek gerekir.
- Diskler senkron çalıştığı için bağımsız olarak disklerin farklı bölgelerine erişim mümkün değildir.
- Sık, bağımsız erişimlerde performans düşer.

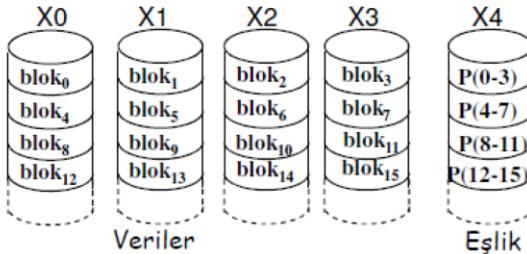
8.7. RAID 6

- İki eşlik bilgisi kullanılır böylece hata sezme miktarı artırılmış olur.
- Kullanıcı verisini N diskte saklamak için N+2 disk kullanmak gerekir.



8.5. RAID 4

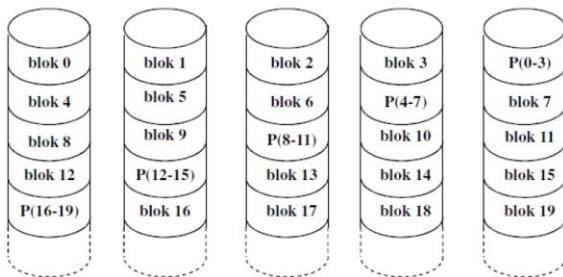
- Diskler senkron değildir ve bağımsız çalışıyor.
- Büyük stripler (blok) kullanılır.
 - Sık ve bağımsız okuma erişimlerinde avantajlıdır.
 - Yüksek G/C istek oranları gerektiren uygulamalar için uygundur.



- Hata sezme/düzelme için eşlik biti eklenir.
- Tek bir eşlik disk var.
 - Okumada eşlik disk okunmaz ancak yazma işlemlerinde disklerin farklı yerlerine aynı anda yazmak mümkün olmaz çünkü eşlik disk tekdir, beklemek gerekir.
 - Eşlik disk bir darboğaz(bottleneck) oluşturabilir.
- Her bir yazma için, kullanıcı verileri değil aynı zamanda karşılık gelen eşlik bitleri güncellenebilir. Buna yazma cezası denir.

8.6. RAID 5

- RAID 4'e benzerdir.
 - Diskler senkron değildir ve bağımsız çalışıyor.
 - Büyük stripler (blok) kullanılır.
- Hata sezme/düzelme için eşlik biti eklenir.



RAID 4'ten farklı olarak eşlik bilgileri disklere dağıtılmıştır.

- Böylece her yazma işleminde aynı eşlik diskinin beklenmesi önlenmiş olur ve olası bir darboğaz önlenebilir.