Veritabanı Yönetim Sistemleri

Bu metin, veri tabanı yönetim sistemleri ve sorgulama dilleri üzerine kapsamlı bir genel bakış sunmaktadır. **Veri tabanının tanımından ve gelişiminden** başlayarak, geleneksel dosyalama sistemlerinin yetersizliklerinden **Veri Tabanı Yönetim Sistemlerinin (VTYS)** ortaya çıkışına kadar uzanan süreci anlatır. Metinler, **ilişkisel (SQL)** ve **ilişkisel olmayan (NoSQL)** veri tabanı modellerinin temel özelliklerini, avantajlarını, dezavantajlarını ve kullanım alanlarını detaylıca karşılaştırmaktadır. Ayrıca, **farklı sorgulama dilleri** ve **veri tabanı seçiminde dikkate alınması gereken kritik faktörler** açıklanmaktadır.

İncelenen kaynaklar, **Veritabanı Sistemleri ve Sorgulama Dilleri** hakkında dijital çağdaki önemleri, evrimleri, türleri, sağladıkları faydaları ve temel farklılıkları üzerine detaylı bilgiler sunmaktadır.

Karşılaştırmalı Veritabanı Yönetim Sistemleri

İşte kaynaklarda bu konular hakkında söylenenler:

1.1 Veritabanı ve Veritabanı Yönetim Sistemi (VTYS) Kavramı

Veritabanı Sistemleri, dijital çağda **bilgi yönetiminin temelini** oluşturur. Bu sistemler, verilerin **düzenli, güvenli ve erişilebilir** bir şekilde depolanmasını, yönetilmesini ve analiz edilmesini sağlar. Geleneksel dosyalama sistemlerinin artan veri hacmi ve işleme hızı ihtiyaçlarını karşılamakta yetersiz kalması, **Veritabanı Yönetim Sistemlerinin (VTYS)** ortaya çıkışına yol açmıştır.

- **Veritabanı (Database)**: Düzenli bir şekilde organize edilmiş ve depolanmış veri koleksiyonudur. Bu veriler **metin, sayılar, resimler, videolar veya diğer dijital formatlarda** olabilir. Veritabanları, verilerin **kolayca erişilmesini, güncellenmesini, sorgulanmasını ve analiz edilmesini** mümkün kılar. Bilgisayar terminolojisinde, veritabanı sistematik erişim imkanı olan, yönetilebilir, güncellenebilir, taşınabilir ve birbirleri arasında tanımlı ilişkiler bulunabilen bilgiler kümesidir.
- **Veritabanı Yönetim Sistemi (VTYS / DBMS)**: Veritabanlarını **oluşturmak, yönetmek, sorgulamak ve korumak** için kullanılan yazılım araçlarıdır. VTYS, kullanıcıların verilere erişmesini, veri eklemesini, güncellemesini veya silmesini kolaylaştırır. Ayrıca performans izleme, ince ayar, yedekleme ve kurtarma gibi yönetim operasyonlarını da sağlar.

1.2 VTYS'nin Sağladığı Yararlar

VTYS'nin Sağladığı Yararlar şunlardır:

- **Veri tekrarını önler (Data Redundancy)**: Aynı verinin farklı yerlerde tekrar edilmesini engeller, bellek israfını ve sistemin hantallaşmasını azaltır.
- **Veri tutarlılığını sağlar (Data Consistency)**: Verinin doğru ve tutarlı kalmasını garanti eder, çelişkili raporların önüne geçer.
- **Aynı andaki erişimlerde tutarsızlıkları önler (Eşzamanlılık – Concurrency)**: Birden fazla kullanıcının aynı anda verilere erişmesine ve işlem yapmasına olanak tanır.
- **Veri bütünlüğünü sağlar (Data Integrity)**: Verinin doğruluğunu ve tutarlılığını ifade eder; örneğin, bir kaydın silinmesi durumunda ilgili tüm tablolardan silinmesini garanti eder.
- **Verilerin güvenliğini sağlar (Data Security)**: Yetkisiz erişimi önlemek için sıkı mekanizmalar sunar; her kullanıcının erişeceği veriler ayrı ayrı tanımlanabilir.
- **Veri bağımsızlığı (Data Independence)**: Programcıların verilerin yapısı ve organizasyonu ile ilgilenmesine gerek kalmaz.

1.3 Veritabanının Dönüşümü ve Türleri

Veritabanları, 1960'lı yılların başlarından itibaren **önemli bir evrim süreci** geçirmiştir. Bu evrim, veri hacminin artması, işleme hızına duyulan ihtiyaç ve farklı veri türlerinin yönetimi gibi gereksinimlerle şekillenmiştir.

Erken Veri Modelleri (1960'lar):

- **Hiyerarşik Veritabanları**: Verileri ağaç benzeri bir yapıda saklar ve sadece bir kaynaktan çoklu alıcıya doğru ilişkiye izin verir. Her varlığın tek bir varlığa bağlı olması ve kökten veri silindiğinde dalların da silinmesi olumsuz yanıdır.
- **Ağ Veritabanları**: Hiyerarşik modelin eksikliklerini gidermek için ortaya çıkmış, daha esnek bir modelle çoklu ilişkilere olanak tanımıştır. Ancak bu modeller karmaşık bir yapıya sahipti.

Modern Veritabanı Türleri:

- **Nesne Yönelimli Veritabanları**: Bilgiyi nesneler ve sınıflar şeklinde depolar ve nesne yönelimli programlama dilleriyle entegre çalışabilirler.
- **Dağıtık Veritabanları**: Verileri birden fazla fiziksel konumda depolar ve yönetir, yüksek erişilebilirlik ve veri yedekliliği sağlar.
- **Veri Ambarları**: Hızlı sorgulama ve analiz amaçları için tasarlanmış merkezi veri havuzlarıdır.
- **Bulut Veritabanları**: Verileri özel, genel veya hibrit bulut bilişim platformlarında depolar, esnek ölçeklendirme ve yönetim kolaylığı sağlar. Örnekler: Amazon RDS, Google Cloud SQL, Azure SQL Database.

- **Çoklu Model Veritabanları**: Farklı veritabanı türlerini tek bir entegre arka uçta birleştirir ve çeşitli veri türlerini içerebilir.
- **Belge/JSON Veritabanları**: Verileri satırlar ve sütunlar yerine JSON formatında depolar.
- **Kendi Kendini Yöneten Veritabanları (Otonom Veritabanları)**: Makine öğreniminden yararlanarak veritabanı yöneticileri tarafından gerçekleştirilen ince ayar, güvenlik, yedekleme, güncelleme gibi rutin yönetim görevlerini otomatikleştirir. Oracle Autonomous Database, 2017 yılının sonlarında tanıtılmıştır.
- **Zaman Serisi Veritabanları**: Zaman damgasına dayalı verilerin depolanması için optimize edilmiştir (örneğin IoT ve finansal veriler için).
- **Açık Kaynak Veritabanları**: Kaynak kodu açık olan ve SQL veya NoSQL olabilen sistemlerdir.
- **OLTP Veritabanları**: Birden fazla kullanıcı tarafından çok sayıda işlemin gerçekleştirilmesi için tasarlanmış hızlı ve analitik veritabanlarıdır.

1.4 İlişkisel Veritabanları (RDBMS)

İlişkisel Veritabanları, 1980'li yıllarda popülerlik kazanmış ve **günümüzde de en yaygın kullanılan** veritabanı modelidir.

- **Temelleri**, Edgar Frank Codd'un 1970'lerde yayımladığı "A Relational Model of Data for Large Shared Data Banks" adlı makalesiyle atılmıştır.
- **Yapı**: Verileri **tablolar (satırlar ve sütunlar)** halinde düzenler ve tablolar arasındaki **ilişkileri** kullanarak veri bütünlüğünü sağlar.
- **Sorgu Dili**: İlişkisel veritabanlarını yönetmek için **SQL (Structured Query Language)** temel bir dildir. SQL, veri yapılarını tanımlama, sorgulama, kayıtları güncelleme ve izinleri yönetme gibi işlemleri sağlar. **Bildirimsel** bir dildir, yani "nasıl"dan ziyade "ne" istendiğine odaklanır. CRUD (Create, Read, Update, Delete) operasyonlarını destekler. İlk olarak 1970'lerde IBM'de Oracle'ın katkılarıyla geliştirilmiş ve 1986'da ANSI, 1987'de ISO tarafından resmi standart haline getirilmiştir. Kesin bir önceden tanımlanmış şemaya sıkı sıkıya bağlı kalmayı gerektirir.
- **ACID Uyumluluğu**: İşlemlerin güvenilirliğini ve veri bütünlüğünü garanti eden **ACID (Atomicity, Consistency, Isolation, Durability)** uyumluluğu ile tanınır.
  - **Atomik**: Bir işlemin ya tamamen tamamlanması ya da hiç tamamlanmaması gerekir.
  - **Tutarlı**: Veriler tüm kural ve kısıtlamalara göre geçerli olmalıdır.
  - **Yalıtılmış**: Eş zamanlı işlemlerin birbirini etkilememesi gerekir.
  - **Dayanıklı**: Başarılı bir işlem gerçekleştikten sonra geri alınamaz.
- **Ölçeklenebilirlik**: **Dikey olarak ölçeklenirler**, yani performansı artırmak için tek bir sunucuya daha fazla CPU, RAM veya SSD kapasitesi eklenir. Bu yaklaşımın sınırları ve maliyetleri olabilir. Yatay ölçekleme de mümkündür ancak karmaşık yapılandırma gerektirebilir.
- **Performans**: Genellikle karmaşık, çoklu birleşimli sorgularda (OLAP) iyi performans

gösterir, ancak veri hacmi arttıkça düşebilir.
- **Kullanım Alanları**: Yapılandırılmış veriler, tanımlanmış ilişkiler, karmaşık sorgular ve güvenilir işlem işleme gerektiren sistemler (ERP, CRM, finans, e-ticaret) için uygundur.
- **Popüler Örnekler**: MySQL, Microsoft SQL Server, Oracle Database, PostgreSQL, SQLite, IBM DB2, MariaDB ve Microsoft Access.

1.5 NoSQL Veritabanları

Geleneksel ilişkisel veritabanlarının ölçeklenebilirlik ve esneklik gibi sınırlamalarına bir yanıt olarak **2000'li yılların sonlarında** NoSQL veritabanları geliştirilmiştir.

- **Tanım**: **"Not Only SQL"** (Sadece SQL değil) anlamına gelir. Yapılandırılmamış veya yarı yapılandırılmış verileri yönetmek için tasarlanmıştır ve önceden tanımlanmış bir şema gerektirmez.
- **Veri Modelleri**: Esnek veri modelleri sunarlar: belge tabanlı, anahtar-değer tabanlı, sütun tabanlı, grafik tabanlı gibi.
  - **Belge Tabanlı**: Verileri JSON veya BSON formatında saklar. Örnek: MongoDB.
  - **Anahtar-Değer Tabanlı**: Her öğe bir öznitelik adı (anahtar) ve değer olarak saklanır. Örnek: Redis.
  - **Geniş Sütunlu (Wide-column stores)**: Verileri satırlar yerine sütunlarda depolar ve büyük veri kümeleri üzerindeki sorgular için optimize edilmiştir. Örnek: Apache Cassandra.
  - **Grafik Tabanlı**: Verileri düğümler ve ilişkiler şeklinde depolar. Örnek: Neo4j.
- **Ölçeklenebilirlik**: **Yatay ölçeklenebilirlik** için tasarlanmışlardır, yani daha fazla sunucu ekleyerek artan veri yüklerini kaldırabilirler. Bulut ortamları için idealdir.
- **Tutarlılık**: Genellikle **BASE (Basically Available, Soft state, Eventually consistent)** yaklaşımını benimserler. Bu, sistemin hemen tutarlı olması garanti edilmese de, belirli bir süre sonra tutarlı bir duruma geleceği anlamına gelir.
- **Performans**: Yüksek hızlı yazma ve okuma işlemleri için tasarlanmıştır. Özellikle basit işlemlerde (OLTP) okuma/yazma hızları yüksektir.
- **Sorgu Dili**: Tek bir sorgu diline bağlı kalmaz; bunun yerine veri modelleriyle uyumlu çeşitli API'ler ve sorgu arayüzleri sunarlar. Örneğin, MongoDB'nin MQL (MongoDB Query Language) ve Apache Cassandra'nın Sorgu Dili (CQL).
- **Kullanım Alanları**: Esneklik ve ölçeklenebilirliğin kritik olduğu senaryolar (içerik yönetim sistemleri, sosyal ağlar, gerçek zamanlı analiz, IoT, mobil uygulamalar, büyük veri) için uygundur.
- **Örnekler**: MongoDB, Cassandra, Couchbase, Redis, Neo4j, Apache HBase gibi sistemler bulunur.

1.6 Sorgulama Dilleri (Query Languages)

Sorgulama dili, veritabanlarından veri almak ve manipüle etmek için kullanılan bir bilgisayar programlama dilidir. Kullanıcıların veritabanı yönetim sistemi (DBMS) ile iletişim

kurarak veri ekleme, güncelleme, silme ve alma gibi işlemleri gerçekleştirmesini sağlar.

Sorgulama Dili Türleri:

- **Prosedürel Diller**: Bilgisayara görevleri nasıl yürüteceğini adım adım talimatlarla belirler. Örneğin, PL/SQL.
- **Bildirimsel Diller**: "Nasıl"dan ziyade "ne" istendiğine odaklanır, yani yalnızca istenen sonucu tanımlar ancak veri alma yöntemini dikte etmez. SQL, bildirimsel bir sorgu dilidir.

Sorgulama Dilleri ve Programlama Dilleri Arasındaki Fark:

- Programlama dilleri (örn. Python, Java), yazılım uygulamaları oluşturmak ve algoritmik işlemeden kullanıcı arayüzü tasarımına kadar her şeyi ele almak için tasarlanmıştır.
- Sorgulama dilleri ise veritabanlarıyla etkileşim kurmak için özeldir; yazılım derlemezler, veritabanı içindeki verileri etkileşimli olarak yönetmek için kullanılırlar.

Yaygın Sorgulama Dilleri:

- **SQL (Structured Query Language)**: İlişkisel veritabanlarını yönetmek için temel bir dildir. CRUD operasyonlarını destekler. Farklı SQL türleri, genellikle kullanıldığı veritabanı yönetim sistemi tarafından belirlenir ve sözdiziminde küçük farklılıklar gösterebilir.
- **GraphQL**: Dinamik istemciye özgü veri alımı için optimize edilmiş, API'lar için yenilikçi bir sorgu dilidir. Tek bir istek, iç içe geçmiş kaynakları getirerek sunucu gezilerini önemli ölçüde azaltır.
- **SPARQL (SPARQL Protocol and RDF Query Language)**: Semantik Web'deki RDF (Resource Description Framework) verilerini sorgulamak için kullanılır. Grafikleri seçme, bağlama ve düğümleri filtreleme yeteneği aracılığıyla ifade edici arama sağlar.
- **Oracle PL/SQL**: Oracle Database ortamı için tasarlanmış, SQL'in prosedürel yeteneklerini artıran güçlü bir uzantıdır.
- **Prisma**: Tür güvenli veritabanı erişimi için otomatik olarak oluşturulan bir sorgu oluşturucuya sahip açık kaynaklı bir veritabanı araç kitidir. Birleşik bir API aracılığıyla birden fazla veritabanını destekler.
- **NoSQL Sorgu Dilleri**: NoSQL veritabanları tek bir sorgu diline bağlı kalmaz; bunun yerine veri modelleriyle uyumlu çeşitli API'ler ve sorgu arayüzleri sunarlar. Örneğin, MongoDB'nin MQL ve Apache Cassandra'nın CQL.

1.7 SQL ve NoSQL Arasındaki Temel Farklar

SQL ve NoSQL veritabanları, dijital verileri depolamak için iki ana veritabanı türüdür ve yapıları, ölçeklenebilirlikleri, ilişkileri, dilleri ve destekleri açısından farklılık gösterir.

Özellik	SQL Veritabanları (İlişkisel)	NoSQL Veritabanları (İlişkisel Olmayan)

Yapı/Şema	Tablolar (satırlar ve sütunlar) ile yapılandırılmış, önceden tanımlanmış, katı şemaya sahiptir.	Dinamik şemalara veya şemasız kurulumlara sahiptir, yapılandırılmamış veya yarı yapılandırılmış veriye adapte olur.
Performans	Karmaşık, çoklu birleşimli sorgularda (OLAP) iyi performans gösterir, ancak veri hacmi arttıkça düşebilir.	Yatay ölçekleme ile büyük veri hacimlerini verimli bir şekilde işler, özellikle basit işlemlerde (OLTP) okuma/yazma hızları yüksektir.
Ölçeklenebilirlik	Dikey olarak ölçeklenir (mevcut donanımı yükseltme). Yatay ölçekleme karmaşık yapılandırma gerektirebilir.	Yatay olarak ölçeklenir (ağa daha fazla sunucu ekleme), bulut ortamları için tasarlanmıştır.
Sorgu Dili	Standartlaştırılmış Yapılandırılmış Sorgu Dili (SQL) kullanır.	Tek bir sorgu diline bağlı değildir, farklı API'ler ve sorgu arayüzleri kullanır.
Tutarlılık	ACID (Atomicity, Consistency, Isolation, Durability) uyumluluğu ile tutarlılığı vurgular, veri bütünlüğünü garantiler.	Genellikle BASE (Basically Available, Soft state, Eventually consistent) yaklaşımını benimser; kullanılabilirlik ve bölüm toleransına öncelik verir.
İşlem Bütünlüğü	ACID uyumluluğu ile kesin işlem bütünlüğü sağlar.	Genellikle katı ACID uyumluluğunu ölçeklenebilirlik ve hız için feda eder, ancak bazıları sınırlı işlem özellikleri sunabilir.
Veri Yapıları	Veri satırları ve sütunları içeren tablo şeklinde bir yapı kullanır.	Belge odaklı veya anahtar-değer çiftleri gibi daha esnek bir yapı kullanır.
İndeksleme	Verileri depolamak için hiyerarşik bir yapı kullanan B-ağacı dizinini kullanır.	Verileri bir anahtar-değer deposunda saklayan bir hash dizini kullanır.
Kullanım Alanları	Yapılandırılmış veriler, tanımlanmış ilişkiler, karmaşık sorgular ve güvenilir işlem işleme gerektiren sistemler (ERP, CRM, finans, e-ticaret) için uygundur.	Esneklik ve ölçeklenebilirliğin kritik olduğu senaryolar (içerik yönetim sistemleri, sosyal ağlar, gerçek zamanlı analiz, IoT, mobil uygulamalar, büyük veri) için uygundur.

1.8 Veritabanı Seçimi İçin Kritik Faktörler

SQL ve NoSQL veritabanları arasında seçim yaparken, uygulamanın ihtiyaçları ve uzun vadeli hedeflerle uyumlu bir dizi faktör göz önünde bulundurulmalıdır.

- **Veri Yapısı ve Şema Esnekliği**: Verilerinizin yapılandırılmış, yarı yapılandırılmış veya yapılandırılmamış olması. SQL katı şema gerektirirken, NoSQL dinamik veya şemasız yapı sunar.

- **Ölçeklenebilirlik Gereksinimleri**: Dikey mi yoksa yatay ölçeklendirmeye mi ihtiyacınız olduğu.
- **Performans**: Uygulamanızın yüksek hızlı yazma/okuma veya karmaşık sorgulara öncelik verip vermediği.
- **Tutarlılık ve İşlemsel Bütünlük (ACID Uyumluluğu)**: Uygulamanızın finansal sistemler gibi yüksek veri bütünlüğü ve işlem garantisi gerektirip gerektirmediği.
- **Sorgu Dili**: SQL'in standartlaştırılmış yapısına mı, yoksa NoSQL'in farklı API ve sorgu arayüzlerine mi ihtiyaç duyduğunuz.
- **Kullanım Alanları**: Uygulamanızın türü ve temel gereksinimleri (örneğin, ERP/CRM için SQL, sosyal ağlar/IoT için NoSQL).
- **Maliyet Etkinliği**: Dikey ölçeklemenin (SQL) mi yoksa yatay ölçeklemenin (NoSQL) uzun vadede daha maliyet etkin olacağı.
- **Ekosistem ve Araçlar**: Mevcut araçların, çerçevelerin ve topluluk desteğinin kullanılabilirliği.
- **Operasyonel Karmaşıklık**: Şema tasarımı ve sürekli bakım gerektiren SQL'e karşın, NoSQL'in sharding ve çoğaltma gibi yerleşik otomatikleştirilmiş operasyonları.
- **Veri Hacmi ve Hızı**: Yüksek hızlı veya büyük veri kümeleri için NoSQL daha uygunken, orta hacimli veriler için SQL tercih edilebilir.
- **Veri Yaşam Döngüsü ve Erişim Modelleri**: Stabil, uzun ömürlü ve karmaşık sorgular gerektiren veriler için SQL; dinamik erişim, sık güncellemeler ve gerçek zamanlı işleme gerektiren veriler için NoSQL tercih edilir.
- **Satıcı Bağımlılığı ve Esneklik**: SQL veritabanları genellikle belirli satıcılara bağlı olabilirken, NoSQL daha fazla esneklik sunabilir.

Modern uygulamalar genellikle her iki veritabanı türünün güçlü yönlerinden yararlanmak için **hibrit veritabanı mimarilerini** kullanmaktan fayda sağlar. Örneğin, tutarlılık gerektiren işlemsel veriler için SQL, analitik, önbelleğe alma veya oturum depolama için NoSQL kullanılabilir. Doğru veritabanı seçimi, projenin başarısı için kritik öneme sahiptir.