VERİ TABANI TÜRLERİ: KATEGORİLERİ, KRİTERLERİ VE YAYGIN ÖRNEKLERİ

Veri tabanları, farklı kullanım senaryolarına, teknolojik gereksinimlere ve sistem mimarilerine göre çeşitli türlere ayrılır. Bu çeşitlilik, verinin **nerede saklandığı**, **nasıl sorgulandığı**, **hangi kaynak kod altyapısına sahip olduğu** gibi birçok kritere göre sınıflandırılmasını sağlar. Aşağıda, veri tabanı türleri beş ana başlık altında incelenmektedir:

1. Kaynak Koduna (Source Code) Göre Veri Tabanı Türleri

Veri tabanları, sahip oldukları kaynak kodun açık ya da kapalı olmasına göre sınıflandırılır.

Tür	Açıklama	Örnekler
Açık Kaynak Kodlu	Kaynak kodları herkese açıktır. Ücretsiz kullanılabilir ve geliştirilebilir.	PostgreSQL, MySQL, MariaDB, SQLite
Ticari / Kapalı Kaynak Kodlu	Lisans gerektirir, kaynak kodları gizlidir. Kurumsal destek sunar.	Oracle Database, Microsoft SQL Server, IBM Db2

2. Lokasyona Göre Veri Tabanı Türleri

Verinin nerede barındırıldığına ve sistemin nerede çalıştığına göre sınıflandırılır.

Tür	Açıklama	Örnekler
Bulut (Cloud) Veritabanı	Veri uzaktaki bir sunucuda tutulur. İnternet üzerinden erişilir.	Amazon Aurora, Google Cloud Spanner, Firebase
Yerel (On-Premise)	Veriler, firmanın kendi fiziksel sunucularında tutulur.	MS SQL Server (kendi sunucuda kurulu), Oracle
Hibrit Veri Tabanı	Hem bulut hem yerel sistemleri birleştirir.	Oracle Cloud Hybrid, Azure SQL Hybrid

3. Sorgulama Diline Göre Veri Tabanı Türleri

Verilerin nasıl sorgulandığına göre (yapısal ya da esnek) türlere ayrılır.

Tür	Açıklama	Örnekler
SQL (Yapısal)	Veriler tablo yapısındadır. Sorgular SQL diliyle yapılır.	MySQL, PostgreSQL, Oracle, MS SQL Server
NoSQL (Yapısal Olmayan)	Esnek yapılar, belge, anahtar-değer ya da grafik biçimli veriler.	MongoDB, Cassandra, Redis, Neo4j

4. Mimarisine Göre Veri Tabanı Türleri

Veri tabanı sisteminin tasarımı ve yönetim mantığına göre ayrılır.

Tür	Açıklama	Örnekler
DBMS (Veri Tabanı Yönetim Sistemi)	Temel veri yönetim sistemleridir, ilişkisel olmayan da olabilir.	dBase, Microsoft Access
RDBMS (İlişkisel VTYS)	Veriler ilişkisel tablolarda saklanır, veri bütünlüğü sağlanır.	Oracle, PostgreSQL, MySQL, MS SQL Server
OODBMS (Nesne Yönelimli VTYS)	Nesne tabanlı veri yapıları kullanılır.	db4o, ObjectDB

5. Yaygın Kullanıma Göre Veri Tabanlarının Gruplaması

Kullanım alanlarına ve sektörel ihtiyaçlara göre yaygın veri tabanı sınıflamaları aşağıdaki gibidir:

Tür	Kullanım Alanı	Örnekler
Kurumsal Veritabanları	Büyük şirketler, finans, sağlık	Oracle, MS SQL Server, SAP HANA
Web Tabanlı Veritabanları	Web siteleri, CMS sistemleri	MySQL, Firebase, MongoDB
Mobil Uygulama Veritabanları	Mobil cihazlarda offline ya da online veri tutma	SQLite, Realm, Firebase Realtime DB
Bilimsel / Akademik Amaçlı	Büyük veri, araştırma analizi	PostgreSQL, Apache Hive
Gerçek Zamanlı Veritabanları	IoT, sensör verileri, hızlı yanıt gereken sistemler	Redis, InfluxDB

Sonuç

Veri tabanlarının bu şekilde kategorilere ayrılması, doğru veri tabanı teknolojisinin seçilmesinde büyük önem taşır. Her sistemin gereksinimi farklı olduğundan, uygun veri tabanı türünü belirlemek sistem mimarisi, veri hacmi, erişim modeli ve güvenlik gereksinimleri açısından stratejik bir karardır. Gelişen dijital dünyada, hem **bulut tabanlı** hem de **açık kaynaklı** veri tabanları giderek daha yaygın hale gelmektedir.

İLİŞKİSEL (SQL) VE İLİŞKİSEL OLMAYAN (NoSQL) VERİ TABANI SİSTEMLERİ

Giriş

Veri, günümüzün en stratejik kaynaklarından biri haline gelmiştir. Bu verilerin güvenli, tutarlı ve verimli bir şekilde saklanması, işlenmesi ve analiz edilmesi için farklı veritabanı sistemleri geliştirilmiştir. Bu sistemlerin başında gelen **İlişkisel Veritabanı Yönetim Sistemleri (SQL)** ve **İlişkisel Olmayan Veritabanı Sistemleri (NoSQL)**, kullanım amacına ve veri tipine göre farklı avantajlar sunmaktadır. Bu bölümde SQL ve NoSQL veritabanı sistemleri, teknik, yapısal ve işlevsel açılardan karşılaştırılacak; ayrıca her iki sistemde kullanılan temel işlemler akademik bir yaklaşımla ele alınacaktır.

SQL ve NoSQL Veri Tabanlarının Karşılaştırılması

Aşağıda yer alan tablo, SQL ve NoSQL sistemlerinin farklı açılardan karşılaştırmasını içermektedir:

Özellik	SQL (İlişkisel Veritabanı)	NoSQL (İlişkisel Olmayan Veritabanı)
Veri Yapısı	Satır ve sütunlardan oluşan tablolar	Belge, anahtar-değer, sütun odaklı, grafik temelli yapı
Şema Yapısı	Katı ve önceden tanımlı	Dinamik ve esnek
Sorgulama Dili	SQL (Structured Query Language)	API, JSON-tabanlı diller, özel sorgular (Mongo Query vb.)

Ölçeklenebilirlik	Dikey (donanım iyileştirmesi)	Yatay (sunucu çoğaltma)
Veri Bütünlüğü	ACID (Atomiklik, Tutarlılık, İzolasyon, Kalıcılık)	BASE (Temelde Ulaşılabilirlik, Gevşek Durum, Sonuçta Tutarlılık)
Uygulama Alanı	Finans, banka, muhasebe, kurumsal uygulamalar	Büyük veri, içerik yönetimi, IoT, sosyal medya
Örnek Sistemler	MySQL, PostgreSQL, Oracle, SQL Server	MongoDB, Cassandra, Redis, Neo4j
Performans	Karmaşık sorgularda güçlü	Gerçek zamanlı veri işleme ve dağıtık sistemlerde etkili
Yedekleme ve Kurtarma	Entegre ve güvenli yöntemlerle	Genellikle manuel ya da harici çözümlerle

SQL ve NoSQL Sistemlerinde Temel Veri İşlemleri

Veritabanı sistemleri, verilerin tanımlanması, düzenlenmesi, sorgulanması ve silinmesi gibi temel işlemler üzerine kuruludur. Aşağıda, hem SQL hem de NoSQL veritabanlarında kullanılan başlıca işlemler detaylı bir biçimde açıklanmıştır.

1. CREATE (Tablo/Koleksiyon Oluşturma)

Verinin saklanacağı yapının tanımlandığı işlemdir.

Sistem	Komut Örneği	Açıklama
SQL	`CREATE TABLE ogrenci (id INT, ad VARCHAR(50));`	“ogrenci” adında bir tablo oluşturur.
NoSQL	`db.createCollection("ogrenci")` (MongoDB)	Koleksiyon oluşturulur, şema tanımı gerekmez.

2. INSERT (Veri Ekleme)

Veritabanına yeni kayıtların eklenmesini sağlar.

Sistem	Komut Örneği	Açıklama
SQL	`INSERT INTO ogrenci (id, ad) VALUES (1, 'Ahmet');`	Yeni bir kayıt tabloya eklenir.

NoSQL	`db.ogrenci.insertOne({id: 1, ad: "Ahmet"})`	Yeni belge koleksiyona eklenir.

3. SELECT / FIND (Veri Okuma)

Veritabanından istenen verilerin sorgulanması işlemidir.

Sistem	Komut Örneği	Açıklama
SQL	`SELECT * FROM ogrenci WHERE id = 1;`	id'si 1 olan kayıtları getirir.
NoSQL	`db.ogrenci.find({id: 1})`	Koşulu sağlayan belgeleri döndürür.

4. UPDATE (Veri Güncelleme)

Var olan kayıtlar üzerinde değişiklik yapmayı sağlar.

Sistem	Komut Örneği	Açıklama
SQL	`UPDATE ogrenci SET ad = 'Mehmet' WHERE id = 1;`	Adı 'Mehmet' olarak güncellenir.
NoSQL	`db.ogrenci.updateOne({id: 1}, {$set: {ad: "Mehmet"}})`	Belge güncellenir.

5. DELETE (Veri Silme)

Belirli verilerin kalıcı olarak silinmesini sağlar.

Sistem	Komut Örneği	Açıklama
SQL	`DELETE FROM ogrenci WHERE id = 1;`	Koşulu sağlayan kayıt silinir.
NoSQL	`db.ogrenci.deleteOne({id: 1})`	İlgili belge silinir.

6. DROP (Tablo/Koleksiyon Silme)

Verinin yapısal olarak tutulduğu tablo veya koleksiyonun tamamen silinmesini sağlar.

Sistem	Komut Örneği	Açıklama

SQL	`DROP TABLE ogrenci;`	“ogrenci” tablosu silinir.
NoSQL	`db.ogrenci.drop();`	Koleksiyon silinir.

Sonuç

SQL ve NoSQL veri tabanı sistemleri, farklı kullanım senaryoları ve veri yapıları için geliştirilmiş iki temel paradigma olarak karşımıza çıkmaktadır. SQL sistemleri veri tutarlılığı, karmaşık sorgular ve güvenlik açısından avantaj sağlarken; NoSQL sistemleri yüksek ölçeklenebilirlik, esneklik ve performans yönünden öne çıkar. Temel veri işlemleri açısından benzer işlevleri yerine getiren bu iki sistemin seçimi, uygulamanın gereksinimlerine göre yapılmalıdır. Bu nedenle modern yazılım projelerinde her iki sistemin de hibrit yapılar içinde kullanımı yaygınlaşmaktadır.

Bu çalışma BTK Akademi kaynakları ve çeşitli akademik içeriklerden derlenerek Arda Karadağ tarafından hazırlanmıştır.