**Veritabanları ve Özelliklerinin Kıyaslanması**

Veritabanı uygulamaları dersi kapsamında günümüzde sık kullanılan veritabanlarını çeşitli kriterlere göre detaylı bir şekilde kıyaslayalım. Aşağıda, en popüler veritabanlarını kullanım amacı, ölçeklenebilirlik, veri kapasitesi, performans, kurulum kolaylığı, kullanım kolaylığı ve diğer önemli kriterlere göre değerlendirilmiştir.

**1. Genel Karşılaştırma Tablosu**

| **Veritabanı** | **Kullanım Amacı** | **Ölçeklenebilirlik** | **Veri Kapasitesi** | **Performans** | **Kurulum Kolaylığı** | **Kullanım Kolaylığı** | **Desteklenen Veritabanı Modeli** | **Ekstra Özellikler** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **MySQL** | Web uygulamaları, küçük ve orta ölçekli projeler | Yatay ve dikey ölçeklenebilir | Büyük veri için optimize edilebilir | Yüksek performans, ancak büyük ölçekli verilerde zayıflayabilir | Kolay | Orta | İlişkisel (RDBMS) | Açık kaynak, geniş topluluk desteği |
| **PostgreSQL** | Büyük ölçekli kurumsal projeler, veri analitiği | Yatay ve dikey ölçeklenebilir | Petabayt seviyesine kadar | Çok yüksek performans ve veri tutarlılığı | Orta | Orta | İlişkisel (RDBMS) + JSON | Gelişmiş veri bütünlüğü, geniş SQL desteği |
| **MongoDB** | Büyük ölçekli NoSQL projeleri, JSON tabanlı veri | Yatay ölçeklenebilir | Büyük veri için uygun | Büyük verilerde hızlı, ancak ACID desteği sınırlı | Kolay | Kolay | Belge tabanlı (NoSQL) | JSON desteği, şemadan bağımsız esneklik |
| **Microsoft SQL Server** | Kurumsal uygulamalar, finans, ERP | Dikey ölçeklenebilir | Büyük ölçekli projeler için optimize edilmiş | Yüksek performans ve güçlü ACID desteği | Orta | Orta | İlişkisel (RDBMS) | Microsoft ekosistemi ile tam entegrasyon |
| **Oracle Database** | Büyük kurumsal uygulamalar, bankacılık | Dikey ölçeklenebilir | Büyük veri için optimize edilmiş | Çok yüksek performans, ancak lisans maliyetleri yüksek | Zor | Orta | İlişkisel (RDBMS) | Üst düzey güvenlik, güçlü transaction desteği |
| **SQLite** | Mobil ve küçük çaplı projeler | Dikey ölçeklenemez | Küçük veri setleri için uygun | Küçük projelerde yüksek, ancak büyük verilerde düşük performans | Çok kolay | Çok kolay | İlişkisel (RDBMS) | Hafif, bağımsız, dosya tabanlı |
| **Firebase (Firestore)** | Mobil uygulamalar, gerçek zamanlı veri işlemleri | Yatay ölçeklenebilir | Büyük ölçekli projelere uygun | Gerçek zamanlı hızlı okuma/yazma | Kolay | Çok kolay | NoSQL (Belge tabanlı) | Google Cloud ile tam entegre, gerçek zamanlı güncellemeler |
| **Cassandra** | Büyük ölçekli dağıtık sistemler, IoT | Yatay ölçeklenebilir | Petabayt seviyesine kadar | Yüksek hızda okuma/yazma performansı | Orta | Orta | NoSQL (Sütun tabanlı) | Dağıtık mimari, yüksek erişilebilirlik |

**2. Detaylı Kriterlere Göre Değerlendirme**

**a) Kullanım Amacı**

* **MySQL** ve **PostgreSQL**, geleneksel ilişkisel veritabanı (RDBMS) yapısına sahip olup web uygulamaları, e-ticaret siteleri ve orta ölçekli projeler için yaygın olarak kullanılır.
* **MongoDB**, esnek yapısı sayesinde büyük veri ve JSON tabanlı verileri işleyen projelerde tercih edilir.
* **Microsoft SQL Server** ve **Oracle Database**, finans, ERP, büyük ölçekli veri yönetimi ve güvenlik açısından kritik kurumsal projelerde kullanılır.
* **SQLite**, küçük çaplı uygulamalar, mobil uygulamalar ve gömülü sistemler için idealdir.
* **Firebase (Firestore)**, gerçek zamanlı veri gerektiren mobil ve web uygulamaları için uygundur.
* **Cassandra**, büyük ölçekli IoT ve dağıtık sistemler için optimize edilmiştir.

**b) Büyük Çaplı vs Küçük Çaplı Projeler İçin Tercih**

* **Büyük çaplı projeler için**: PostgreSQL, Microsoft SQL Server, Oracle, Cassandra
* **Orta çaplı projeler için**: MySQL, MongoDB
* **Küçük çaplı projeler için**: SQLite, Firebase

**c) Performans**

* **Yüksek performanslı seçenekler**: PostgreSQL, Microsoft SQL Server, Oracle, Cassandra
* **Orta düzey performanslı seçenekler**: MySQL, MongoDB, Firebase
* **Daha düşük performanslı seçenekler**: SQLite

**d) Veri Kapasitesi**

* **Petabayt seviyesine kadar**: PostgreSQL, Oracle, Cassandra
* **Terabayt seviyesinde**: MySQL, Microsoft SQL Server, MongoDB
* **Gigabayt seviyesinde**: SQLite, Firebase

**e) Kurulum ve Kullanım Kolaylığı**

* **Kolay kurulum ve kullanım**: SQLite, Firebase, MySQL, MongoDB
* **Orta seviyede zorluk**: PostgreSQL, Microsoft SQL Server
* **Daha zor kurulum gerektirenler**: Oracle, Cassandra

**f) Güvenlik ve Veri Bütünlüğü**

* **Yüksek güvenlik gerektiren projeler**: Oracle, Microsoft SQL Server, PostgreSQL
* **Orta düzey güvenlik**: MySQL, MongoDB
* **Daha hafif güvenlik**: SQLite, Firebase

**g) Fiyatlandırma ve Lisans Durumu**

* **Ücretsiz/Açık Kaynak**: MySQL, PostgreSQL, SQLite, MongoDB, Cassandra
* **Ücretli (Kurumsal)**: Microsoft SQL Server, Oracle
* **Bulut tabanlı fiyatlandırma**: Firebase

**Sonuç ve Öneriler**

* **Web ve küçük ölçekli projeler için**: **MySQL** veya **SQLite**
* **Kurumsal projeler ve veri analitiği için**: **PostgreSQL** veya **Microsoft SQL Server**
* **Gerçek zamanlı veri ve mobil uygulamalar için**: **Firebase**
* **Büyük ölçekli dağıtık sistemler için**: **Cassandra**
* **Bankacılık ve yüksek güvenlik gerektiren projeler için**: **Oracle Database**

Seçim yaparken projenin ihtiyaçlarına göre en uygun veritabanını seçmek kritik öneme sahiptir. Eğer ölçeklenebilirlik, güvenlik ve veri tutarlılığı önemliyse PostgreSQL, Microsoft SQL Server veya Oracle tercih edilmelidir. Ancak, NoSQL tabanlı çözümler arıyorsanız MongoDB veya Firebase gibi çözümler daha esnek bir yapı sunabilir.

**1. Yatay ve Dikey Ölçeklenebilirlik Nedir?**

Ölçeklenebilirlik, bir veritabanının artan veri ve işlem yüküne nasıl yanıt verdiğini ifade eder. İki temel ölçeklenebilirlik yöntemi vardır:

**a) Yatay Ölçeklenebilirlik (Horizontal Scaling - Scale Out)**

* Sistem performansını artırmak için **daha fazla sunucu ekleyerek** genişleme yapılır.
* Veriler birden fazla sunucuya bölünerek dağıtılır.
* Büyük ölçekli sistemlerde tercih edilir (örneğin, **MongoDB, Cassandra, Firebase** gibi NoSQL veritabanları genellikle yatay ölçeklenebilir).
* **Avantajı**: Sonsuza kadar genişletilebilir; büyük veri yönetimi için idealdir.
* **Dezavantajı**: Veri tutarlılığı (ACID) sağlamak zorlaşabilir, karmaşık yönetim gerektirir.

**b) Dikey Ölçeklenebilirlik (Vertical Scaling - Scale Up)**

* Daha güçlü donanımlara geçerek (**CPU, RAM, disk artırarak**) sistemin kapasitesi artırılır.
* Veriler tek bir sunucuda kalır.
* **PostgreSQL, MySQL, Microsoft SQL Server, Oracle** gibi ilişkisel (RDBMS) veritabanları genellikle dikey ölçeklenebilir.
* **Avantajı**: Yönetimi daha kolaydır, veri tutarlılığı korunur.
* **Dezavantajı**: Donanımın sınırlarına ulaşılınca daha fazla büyütülemez ve maliyetli olabilir.

**Özetle:**

* **Yatay ölçekleme** = **Daha fazla sunucu ekleyerek genişletme**
* **Dikey ölçekleme** = **Daha güçlü donanım ekleyerek genişletme**

**2. İlişkisel (RDBMS) + JSON Nedir?**

**a) İlişkisel Veritabanı (RDBMS - Relational Database Management System)**

* **Tablolardan oluşur** ve her tablo belirli sütunlara (kolonlara) sahiptir.
* **SQL (Structured Query Language)** kullanarak veri sorgulama işlemleri yapılır.
* **Veriler, tablolar arasındaki ilişkilerle yönetilir** (örneğin: Öğrenciler ve Notlar tablosu arasında ilişki olabilir).
* Örnekler: **MySQL, PostgreSQL, Microsoft SQL Server, Oracle**

**b) JSON Desteği (JavaScript Object Notation)**

* JSON, veriyi **anahtar-değer** çiftleriyle saklayan hafif bir veri formatıdır.
* Geleneksel RDBMS'ler (PostgreSQL, MySQL, SQL Server) **JSON formatında veri saklayabilir ve işleyebilir.**
* Bu sayede **yarı-yapılandırılmış (semi-structured) veri** işlenebilir.
* **Örnek kullanım**:

INSERT INTO urunler (id, ad, ozellikler) VALUES (1, 'Telefon', '{"renk": "siyah", "bellek": "128GB"}');

Burada **ozellikler** sütununda JSON formatında veri saklanmıştır.

**Özetle:**

* **İlişkisel (RDBMS) veritabanları** tablo yapısında çalışırken, JSON desteği sayesinde **esneklik kazandırır**.
* **PostgreSQL ve MySQL** gibi bazı ilişkisel veritabanları **JSON formatında veri saklayabilir ve sorgulayabilir**.

**3. Yüksek Performans ve Güçlü ACID Desteği Nedir?**

**a) Yüksek Performans**

* Veritabanının **hızlı okuma, yazma, işlem yapma yeteneği** olarak tanımlanır.
* Verinin hızlı çağrılması, indeksleme, önbellekleme gibi teknikler kullanılarak artırılır.
* **PostgreSQL, Oracle ve SQL Server** genellikle yüksek performanslı veritabanlarıdır.

**b) ACID Desteği (Atomicity, Consistency, Isolation, Durability)**

ACID, **veri bütünlüğünü ve güvenilirliğini** sağlamak için kullanılan dört temel prensibi ifade eder:

1. **Atomicity (Atomiklik)**:
   * Bir işlem ya **tamamen gerçekleşir ya da hiç gerçekleşmez**.
   * Örneğin, **banka transferi sırasında** para bir hesaptan çekilip diğerine yatırılmadıysa, işlem iptal edilir.
2. **Consistency (Tutarlılık)**:
   * Veri her zaman **geçerli ve tutarlı** olmalıdır.
   * Örneğin, bir ürünün stoktan düşmesi gerekirken hata olursa eski veriye dönülerek tutarlılık sağlanır.
3. **Isolation (İzolasyon)**:
   * Birden fazla işlem **birbirini etkilememelidir**.
   * Örneğin, iki müşteri aynı anda bir bilet alırken, biri bileti satın alana kadar diğeri bekletilir.
4. **Durability (Dayanıklılık)**:
   * İşlem tamamlandıktan sonra veri **kesin olarak saklanır**, sistem çökmese bile kaybolmaz.

**Özetle:**

* **ACID desteği olan veritabanları (PostgreSQL, SQL Server, Oracle)** yüksek veri güvenliği sunar.
* **MongoDB gibi bazı NoSQL veritabanlarında ACID desteği kısıtlı olabilir**.

**4. NoSQL Nedir?**

**NoSQL (Not Only SQL)**, ilişkisel veritabanlarından farklı olarak **tablo yapısı kullanmayan** bir veritabanı modelidir.

* **Esneklik sunar**, JSON veya belge tabanlı veri formatlarını destekler.
* **Büyük veri ve yatay ölçeklenebilirlik için uygundur**.
* **Hızlı okuma/yazma işlemleri sağlar**, ancak ACID uyumluluğu zayıf olabilir.

**NoSQL Türleri:**

1. **Belge Tabanlı (Document-Based)** → **MongoDB, Firebase**
   * JSON formatında veri saklar.
2. **Anahtar-Değer (Key-Value)** → **Redis, DynamoDB**
   * Veriler anahtar-değer çiftleri olarak saklanır.
3. **Sütun Tabanlı (Column-Family)** → **Cassandra, HBase**
   * Büyük ölçekli dağıtık sistemlerde kullanılır.
4. **Graf Tabanlı (Graph Database)** → **Neo4j**
   * Sosyal ağ, öneri sistemleri gibi bağlantılı veri yapılarında kullanılır.

**5. ERP Nedir?**

**ERP (Enterprise Resource Planning - Kurumsal Kaynak Planlama)**, bir işletmenin **finans, muhasebe, insan kaynakları, stok yönetimi, üretim, müşteri ilişkileri (CRM) gibi süreçlerini tek bir merkezi sistemde yöneten yazılım** sistemleridir.

* **Örnek ERP Sistemleri**: SAP, Oracle ERP, Microsoft Dynamics
* **Avantajları**:
  + **Bütünleşik sistem**: Tüm iş süreçleri tek bir sistemde yönetilir.
  + **Gerçek zamanlı veri**: Hataları ve gecikmeleri önler.
  + **Verimlilik artışı**: İş süreçleri otomatikleştirilir.

**Özetle:**  
ERP, büyük şirketlerin **tüm iş süreçlerini tek bir merkezden yönetmesini sağlayan** bir yazılımdır.