* **Veritabnının Tarihçesi:** 1960'lar: Veritabanları, büyük bilgisayar sistemlerinin geliştirilmesiyle ortaya çıkmıştır. Bu dönemde, ilişkisel olmayan ve hiyerarşik veritabanı modelleri kullanılmıştır.
* 1970'ler: IBM'den Edgar F. Codd, ilişkisel veritabanı modelini önerdi ve ilişkisel veritabanı sistemlerinin temellerini attı. Bu dönemde, SQL gibi standart sorgu dilleri ve ilişkisel veritabanı yönetim sistemleri (RDBMS) geliştirildi.
* 1980'ler: İlişkisel veritabanı sistemleri popüler hale geldi ve ticari olarak kullanılmaya başlandı. Oracle, IBM DB2 ve Microsoft Access gibi büyük veritabanı şirketleri ve ürünleri ortaya çıktı.
* 1990'lar: İnternetin gelişimi ve e-ticaretin yükselişiyle birlikte, veritabanları daha da önem kazandı. Büyük ölçekli veritabanları ve veri ambarları oluşturuldu.
* 2000'ler: Bulut bilişim ve büyük veri analitiği gibi teknolojilerin yükselişi, veritabanlarının dağıtık ve ölçeklenebilir hale gelmesine yol açtı. NoSQL ve NewSQL gibi alternatif veritabanı modelleri ortaya çıktı.
* 2010'lar: Büyük veri analitiği ve yapay zeka gibi alanlardaki gelişmeler, veritabanlarının daha karmaşık ve çok yönlü hale gelmesine neden oldu. Ayrıca, veri güvenliği ve uyumluluk konuları da daha fazla önem kazandı.
* Günümüzde, veritabanları işletmelerin ve kuruluşların temel altyapılarından biri olarak kabul edilmekte olup, hızla gelişen teknolojiye ve veri ihtiyaçlarına uyum sağlamak için sürekli olarak yenilikler yapılmaktadır.

**VERİ TABANINA İLİŞKİN TEMEL KAVRAMLAR**

* **Veritabanı**: Verilerin düzenli bir şekilde depolandığı ve yönetildiği bir elektronik ortamdır.
* **Tablo**: Veritabanında verilerin organize edildiği yapılardır. Sütunlar (alanlar) ve satırlardan (kayıtlar) oluşurlar.
* **Alan (Sütun)**: Bir tablodaki belirli bir veri türünü temsil eden sütunlardır. Örneğin, bir kullanıcının adını, soyadını veya yaşını içerebilir.
* **Kayıt (Satır)**: Bir tablodaki tek bir veri öğesini temsil eden satırlardır. Her satır, tablodaki alanların bir kombinasyonunu içerir.
* **Anahtar (Primary Key)**: Bir tablodaki her kaydı benzersiz bir şekilde tanımlayan bir alan veya alan grubudur. Genellikle bu, birincil anahtar olarak adlandırılan bir tek bir alan olabilir.
* **Yabancı Anahtar (Foreign Key)**: Bir tablodaki veriler arasındaki ilişkiyi tanımlayan bir alan veya alan grubudur. Başka bir tablodaki birincil anahtara atıfta bulunabilir.
* **İndeks**: Veritabanı performansını artırmak için kullanılan yapısal bir öğedir. Belirli bir sütuna veya sütun kombinasyonuna hızlı erişim sağlar.
* **SQL (Structured Query Language)**: İlişkisel veritabanı yönetim sistemlerinde veri manipülasyonu, tanımlama ve sorgulama için kullanılan standart bir dil.
* **Normalizasyon**: Veritabanı tasarımının bir parçası olan ve verilerin tekrarlanmasını en aza indirerek veritabanı yapısını optimize etmeyi amaçlayan bir süreçtir.
* **Yedekleme ve Kurtarma**: Veritabanıyla ilişkili verilerin kaybını önlemek için düzenli olarak yedeklenmesi ve gerektiğinde geri yüklenmesi işlemidir.

**GELENEKSEL DOSYA SİSTEMLERİ VE VERİTABANI YÖNETİM SİSTEMLERİ**

Veritabanı yönetim sistemleri, veri yönetiminde daha yapılandırılmış bir yaklaşım sunar ve veri bütünlüğü, paylaşılabilirlik ve güvenlik gibi önemli avantajlar sağlar. Bununla birlikte, geleneksel dosya sistemleri, basit uygulamalarda hızlı ve doğrudan veri erişimi sağlama avantajına sahiptir.

**Geleneksel dosya sistemlerinin sakıncaları**

* **Veri Tekrarı ve Tutarsızlığı**: Farklı uygulamaların farklı dosya formatları kullanması, veri tekrarına ve tutarsızlıklara neden olabilir. Aynı veri birden fazla dosyada depolanabilir ve bu da veri bütünlüğünü zayıflatabilir.
* **Kısıtlı Paylaşılabilirlik**: Dosyalar genellikle belirli bir uygulama tarafından kullanılır ve diğer uygulamalarla paylaşılamaz. Bu, veriye erişimde ve kullanımda kısıtlamalara yol açabilir.
* **Zor Veri Erişimi ve Manipülasyonu**: Dosyaların yapısal olarak organize edilmemesi, veriye erişimi ve manipülasyonu zorlaştırabilir. Belirli bir veriye erişmek veya değiştirmek için dosyalar arasında dolaşmak gerekebilir.
* **Zayıf Veri Güvenliği**: Dosyalar genellikle düşük düzeyde güvenlik önlemleriyle korunur. Verilere yetkisiz erişim riski yüksektir ve veri güvenliği sağlanamaz.
* **Performans Sorunları**: Büyük veri hacimleriyle çalışırken, dosya sistemleri performans sorunlarına yol açabilir. Verilere erişim ve manipülasyon süreçleri yavaş olabilir.
* **Yedekleme ve Kurtarma Zorlukları**: Dosyaların yedeklenmesi ve kurtarılması karmaşık olabilir. Veri kaybı riski yüksektir ve veri kurtarma işlemleri zorlu olabilir.

**VERİTABANI YÖNETİM SİSTEMLERİNİN AVANTAJLARI**

* **Veri Bütünlüğü ve Tutarlılığı**: Veritabanı yönetim sistemleri, veri bütünlüğünü korumak için çeşitli kısıtlamalar ve ilişkiler sağlar. Bu, veri tutarlılığını artırır ve veri bütünlüğü sorunlarını önler.
* **Paylaşılabilirlik ve Erişilebilirlik**: Veritabanları, birden fazla kullanıcı ve uygulama arasında veri paylaşımını kolaylaştırır. Aynı veriye birden fazla kullanıcı aynı anda erişebilir.
* **Veri Güvenliği**: Veritabanı yönetim sistemleri, veri güvenliği için çeşitli güvenlik önlemleri sunar. Kullanıcı yetkilendirme, şifreleme ve izleme gibi mekanizmalarla veri güvenliği sağlanır.
* **Veri Yönetiminde Esneklik**: Veritabanları, farklı veri tiplerini ve yapılarını destekler. Bu, çeşitli veri tipleriyle çalışmayı ve veri yönetimini esnek hale getirir.
* **Yüksek Performans**: İyi tasarlanmış bir veritabanı, veri erişim ve manipülasyon işlemlerini hızlı bir şekilde gerçekleştirir. İndeksleme, sorgu optimizasyonu ve veri parçalama gibi tekniklerle performans artırılır.
* **Veri Yedekleme ve Kurtarma**: Veritabanı yönetim sistemleri, veri yedekleme ve kurtarma süreçlerini kolaylaştırır. Veri kaybını önlemek için düzenli yedeklemeler yapılarak, veri kurtarma işlemleri hızlı ve etkili bir şekilde gerçekleştirilir.
* **Veri Modelleme ve Analiz**: İlişkisel veritabanı yönetim sistemleri, verilerin mantıksal yapılarını temsil etmek için ilişkisel modellemeyi kullanır. Bu da veri analizi ve raporlama işlemlerini kolaylaştırır.

**VERİTABANI KULLANICILARI**

* **Veritabanı Yöneticileri** :
  + Veritabanı yöneticileri, veritabanı sistemlerini planlar, uygular, bakımını yapar ve günceller.
  + Veritabanı güvenliğini sağlar, yedekleme ve kurtarma süreçlerini yönetir.
  + Performans izleme ve iyileştirme çalışmaları yapar.
* **Uygulama Geliştiricileri** :
  + Uygulama geliştiricileri, veritabanı üzerinde çalışan uygulamaları tasarlar, geliştirir ve bakımını yapar.
  + SQL ve diğer sorgu dillerini kullanarak veritabanı sorgularını oluşturur ve yönetir.
  + Veritabanı şemalarını tasarlar ve uygulamalar için veri modellemesi yapar.
* **Son Kullanıcılar:**
  + Son kullanıcılar, genellikle işletmenin farklı departmanlarından gelen çalışanlardır.
  + Veritabanı üzerinde sorgular çalıştırır, raporlar oluşturur ve verilere erişir.
  + Kullanıcı arayüzleri veya raporlama araçları kullanarak veritabanı verilerini görüntüler ve analiz eder.
* **Veritabanı Tasarımcıları** :
  + Veritabanı tasarımcıları, veritabanı şemalarını ve yapılarını tasarlar.
  + İhtiyaçları analiz eder, veri modellemesi yapar ve ilişkisel şemalar oluşturur.
  + Veritabanı performansını artırmak için endeksleme ve diğer optimizasyon tekniklerini uygular.
* **Veritabanı İş Analistleri** :
  + Veritabanı iş analistleri, veritabanı verilerini analiz eder ve yorumlar.
  + İş gereksinimlerini anlar, veri analizi yapar ve raporlar oluşturur.
  + Veritabanı verilerinin doğruluğunu ve bütünlüğünü sağlar.

**VERİTABANI YÖNETİM SİSTEMLERİNİN MİMARİSİ**

* **Kullanıcı Arayüzü**: Kullanıcıların veritabanıyla etkileşimde bulunmalarını sağlayan bir arayüzdür. Bu, SQL sorgularını çalıştırma, veri girme ve raporlama gibi işlevleri içerebilir. Grafik arayüzler, komut satırı arabirimleri veya web tabanlı arayüzler olabilir.
* **Sorgu Dili İşleyicisi**: Kullanıcıların gönderdiği SQL veya diğer sorgu dillerini işleyen bir bileşendir. Bu bileşen, sorguyu analiz eder, sorgunun en uygun yürütme planını oluşturur ve veritabanı yönetim sistemine yürütme talebini iletir.
* **Sorgu Yürütme Motoru**: Sorguların gerçekleştirilmesinden ve sonuçların üretilmesinden sorumlu olan bileşendir. Bu bileşen, veri erişimini, işlemleri ve gerektiğinde veri manipülasyonunu gerçekleştirir.
* **Veri Depolama Motoru**: Veritabanında saklanan verilerin fiziksel olarak nasıl depolanacağından sorumlu olan bileşendir. Bu, veri yapılarını, dosyaları, veri bloklarını ve endeksleri yönetir.
* **İşlem Yöneticisi**: Eşzamanlılık kontrolünü ve işlem yönetimini sağlar. Birden çok kullanıcının aynı anda veritabanına erişmesini ve işlem yapmasını sağlar, ancak bu işlemlerin birbirini engellememesi ve veri bütünlüğünü sağlaması gerekir.
* **Veri Güvenlik ve Yetkilendirme**: Veritabanının güvenliğinden ve erişim yetkilendirmesinden sorumlu olan bileşendir. Kullanıcıların veriye erişim haklarını yönetir, kimlik doğrulama süreçlerini gerçekleştirir ve veri güvenliğini sağlar.
* **Yedekleme ve Kurtarma**: Veri kaybını önlemek için düzenli yedeklemelerin yapılmasını ve veri kurtarma işlemlerinin yönetilmesini sağlar.
* **Veritabanı Yöneticisi (DBA) Arayüzü**: Veritabanı yöneticilerinin veritabanını yapılandırmasını, izlemesini, bakımını yapmasını ve performansını optimize etmesini sağlayan bir arayüzdür.

**ÜÇ ŞEMA MİMARİSİ**

Üç şema mimarisi, veritabanı yönetim sistemlerinde veri bağımsızlığını sağlamak için kullanılan bir tasarım prensibidir. Bu mimari, veri modelleme ve veritabanı yönetiminde genellikle kullanılır ve şu üç ana bölümü içerir:

* **Harici Şema (External Schema veya View Schema)**:
  + Harici şema, son kullanıcıların veya uygulamaların gördüğü ve eriştiği veritabanının dışında yer alan bir yapıdır.
  + Bu düzeydeki şema, kullanıcıların ihtiyaçlarına uygun olarak özelleştirilmiş ve belirli bir amaç için optimize edilmiş veri görünümlerini içerir.
  + Harici şema, genellikle veritabanının mantıksal modelini temsil eder. Kullanıcılar, bu seviyede veriye erişmek için sorgular oluşturur ve raporlar üretir.
* **Kavramsal Şema (Conceptual Schema veya Conceptual Model)**:
  + Kavramsal şema, veritabanının mantıksal yapısını ve ilişkilerini tanımlayan orta seviyedeki bir yapıdır.
  + Bu düzeydeki şema, veritabanının genel yapısını ve veriler arasındaki ilişkileri temsil eder. Farklı harici şemaları desteklemek için kullanılır.
  + Veritabanı tasarımcıları ve veritabanı yöneticileri tarafından oluşturulan ve yönetilen bir düzeydir.
* **Dahili Şema (Internal Schema veya Physical Schema)**:
  + Dahili şema, veritabanının fiziksel depolama yapısını ve verilerin fiziksel düzeyde nasıl saklandığını tanımlar.
  + Bu düzeydeki şema, veritabanının altında yatan fiziksel yapıyı temsil eder. Disk üzerindeki veri dosyalarını, veri bloklarını ve endeksleri içerir.
  + Veritabanı yöneticileri ve veri tabanı mimarları tarafından tasarlanır ve yönetilir.

Üç şema mimarisi, veritabanı sistemlerindeki veri bağımsızlığını sağlar. Bu, bir düzeydeki değişikliklerin diğer düzeyleri etkilememesini sağlar. Örneğin, harici şemadaki bir değişiklik, kavramsal veya dahili şemayı etkilemez. Bu sayede, veritabanı tasarımı daha esnek hale gelir ve veri modellemesi sürecinde daha fazla kontrol sağlanır.

**Veritabanı Yönetim Sistemlerinde Kullanılan Diller**

* **SQL (Structured Query Language)**:
  + SQL, ilişkisel veritabanı yönetim sistemlerinde veri manipülasyonu, tanımlama ve sorgulama için standart bir dil olarak kullanılır.
  + Veri eklemek, güncellemek, silmek ve sorgulamak için SQL sorguları kullanılır.
  + Tablo ve veritabanı yapılarını tanımlamak için DDL (Data Definition Language), veri sorgulamak ve manipüle etmek için DML (Data Manipulation Language), veritabanı işlemlerini yönetmek için DCL (Data Control Language) ve veritabanı yönetimini yönetmek için DCL (Data Control Language) gibi farklı SQL türleri bulunmaktadır.
* **PL/SQL (Procedural Language/Structured Query Language)**:
  + PL/SQL, Oracle veritabanı yönetim sistemi için Oracle tarafından geliştirilen bir programlama dilidir.
  + SQL sorgularını içeren prosedürler, işlevler ve paketler oluşturmak ve yürütmek için kullanılır.
  + Veri manipülasyonu ve iş mantığı işlemlerinin yürütülmesi için kullanılır.
* **Transact-SQL (T-SQL)**:
  + Transact-SQL, Microsoft SQL Server'da kullanılan bir SQL dilidir.
  + Standart SQL'e ek olarak, T-SQL, yerleşik depolama prosedürleri, fonksiyonlar, tetikleyiciler ve diğer gelişmiş özellikler içerir.
* **PL/pgSQL**:
  + PL/pgSQL, PostgreSQL veritabanı yönetim sistemi için PostgreSQL topluluğu tarafından geliştirilen bir programlama dilidir.
  + SQL sorgularını içeren depolama prosedürleri, fonksiyonlar ve tetikleyiciler oluşturmak için kullanılır.
* **Java, Python, Ruby vb. Programlama Dilleri**:
  + Bazı veritabanı yönetim sistemleri, harici uygulamaları entegre etmek veya özel işlevsellik sağlamak için genel programlama dilleri kullanılmasına izin verir. Örneğin, JDBC (Java Database Connectivity) Java uygulamalarının veritabanlarına erişmesini sağlar.

**VERİTABANI TÜRLERİ**

* **İlişkisel Veritabanları**:
  + İlişkisel veritabanları, tablo ve ilişkilerle verilerin yapılandırıldığı bir türdür.
  + Verilerin tablolar halinde düzenlenmesi ve bu tablolar arasında ilişkilerin tanımlanması esas alınır.
  + Örnek olarak, MySQL, PostgreSQL ve Microsoft SQL Server gibi ilişkisel veritabanı yönetim sistemleri bulunmaktadır.
* **Belge Tabanlı Veritabanları**:
  + Belge tabanlı veritabanları, JSON veya XML gibi belge formatlarını kullanarak verileri depolar.
  + Her belge, bir belge türüne veya kayıt türüne karşılık gelir ve belgeler genellikle belge tabanlı veritabanı yönetim sistemlerinde koleksiyonlar içinde gruplanır.
  + MongoDB, Couchbase ve CouchDB gibi belge tabanlı veritabanları bu kategoriye örnektir.
* **Graf Tabanlı Veritabanları**:
  + Graf tabanlı veritabanları, graf yapısıyla verileri temsil eder.
  + Noktalar (düğümler) ve kenarlar (kenarlar) arasındaki ilişkilerle verileri modelleyen graf tabanlı veritabanları, ağ, sosyal medya analitiği ve bilgi keşfi gibi alanlarda kullanılır.
  + Neo4j, Amazon Neptune ve ArangoDB gibi graf tabanlı veritabanları bu kategoriye örnektir.
* **Anahtar-Değer Mağazaları (Key-Value Stores)**:
  + Anahtar-değer mağazaları, basit bir anahtar-değer çiftleri koleksiyonu olarak verileri saklar.
  + Her bir veri kaydı, bir anahtar ve ona karşılık gelen bir değerden oluşur.
  + Redis, Apache Cassandra ve Amazon DynamoDB gibi anahtar-değer mağazaları bu kategoriye örnektir.
* **Sütun Tabanlı Veritabanları**:
  + Sütun tabanlı veritabanları, sütunlara göre depolama ve erişim sağlayarak büyük veri setlerinin etkin bir şekilde işlenmesini sağlar.
  + Veri sütunlarına göre paralel işlem yapılmasını sağlayarak yüksek performans sağlarlar.
  + Apache HBase ve Google Bigtable gibi sütun tabanlı veritabanları bu kategoriye örnektir.

**VERİTABANI YÖNETİM SİSTEMİ YAZILIMLARI**

* **Oracle Database**:
  + Oracle Corporation tarafından geliştirilen Oracle Database, dünyanın en popüler ilişkisel veritabanı yönetim sistemlerinden biridir.
  + Büyük işletmeler, kurumsal uygulamalar ve bulut tabanlı hizmetler için geniş bir yelpazede kullanılır.
* **MySQL**:
  + Oracle Corporation tarafından geliştirilen ve açık kaynaklı olarak sunulan MySQL, web uygulamaları ve küçük ve orta ölçekli işletmeler için popüler bir ilişkisel veritabanı yönetim sistemidir.
  + Basit ve hızlı, aynı zamanda geniş bir topluluğa ve destek ekosistemine sahiptir.
* **Microsoft SQL Server**:
  + Microsoft Corporation tarafından geliştirilen Microsoft SQL Server, Windows platformu için güçlü bir ilişkisel veritabanı yönetim sistemidir.
  + Kurumsal uygulamalar, iş zekası ve büyük veri işleme için yaygın olarak kullanılmaktadır.
* **PostgreSQL**:
  + PostgreSQL, açık kaynaklı bir ilişkisel veritabanı yönetim sistemidir.
  + Yüksek derecede uygunluk, gelişmiş özellikler ve geniş bir topluluk tarafından desteklenir.
  + Web uygulamaları, veri analizi ve coğrafi bilgi sistemleri (GIS) gibi birçok alanda kullanılmaktadır.
* **SQLite**:
  + SQLite, hafif ve yerel bir ilişkisel veritabanı yönetim sistemidir.
  + Gömülü sistemler, mobil uygulamalar ve basit veri depolama gereksinimleri için idealdir.
* **MongoDB**:
  + MongoDB, belge tabanlı ve NoSQL bir veritabanı yönetim sistemidir.
  + Esnek veri modeli, yüksek ölçeklenebilirlik ve hızlı geliştirme süreçleri sağlar.
* **Cassandra**:
  + Apache Cassandra, dağıtık ve yüksek ölçeklenebilir bir sütun tabanlı veritabanı yönetim sistemidir.
  + Büyük veri uygulamaları, IoT (Nesnelerin İnterneti) ve bulut tabanlı sistemler için popülerdir.
* **Redis**:
  + Redis, anahtar-değer mağazası olarak bilinen bir veritabanı yönetim sistemidir.
  + Yüksek performanslı veri yapıları ve önbellek çözümleri için sıkça kullanılır.

**Bire bir ilişki**

Öğrenci

+------------------------+

| ÖğrenciID |

| Ad |

| Soyad |

| ... |

+------------------------+

TC Kimlik )

+------------------------+

| ÖğrenciID |

| TC Kimlik Numarası |

| ... |

+------------------------+

**Bire çok ilişki**

Öğrenci

+------------------------+

| ÖğrenciID |

| Ad |

| Soyad |

| ... |

+------------------------+

Ders

+------------------------+

| DersID |

| Ders Adı |

| ... |

+------------------------+

ÖğrenciDers

+------------------------+

| ÖğrenciID |

| DersID |

| ... |

+------------------------+

**Çoka çok ilişki**

Müşteri

+------------------------+

| MüşteriID |

| Ad |

| Soyad |

| ... |

+------------------------+

Sipariş

+------------------------+

| SiparişID |

| Tarih |

| ... |

+------------------------+

Ürün

+------------------------+

| ÜrünID |

| Ad |

| Fiyat |

| ... |

+------------------------+

SiparişDetay

+------------------------+

| SiparişID |

| ÜrünID |

| Adet |

| ... |

+------------------------+