

Veritabanı Yönetim Sistemleri

1906003022015

Dr. Öğr. Üy. Önder EYECİOĞLU
Bilgisayar Mühendisliği

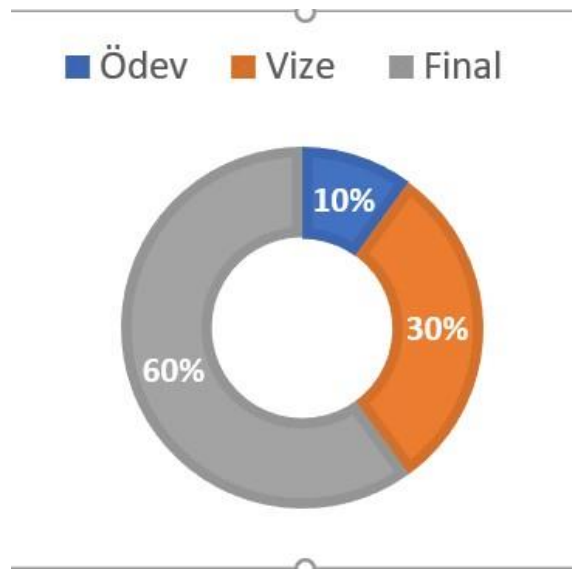


Giriş

Ders Günü ve Saati:

Pazartesi: 09:15-12:45

- Devam zorunluluğu %70
- Uygulamalar MS SQL ve MongoDB üzerinde gerçekleştirilecektir.



HAFTA	KONULAR
Hafta 1	VT ve VTYS'ne giriş
Hafta 2	ER Veri Modeli
Hafta 3	İlişkisel Modeller, İlişkisel model tasarımı
Hafta 4	İlişkisel Cebir ve Hesaplamalar
Hafta 5	İlişkisel Sorgular, SQL giriş
Hafta 6	SQL ile veri tabanı programlama
Hafta 7	SQL-Kısıtlar:Veri-tipi,birincil-anahtar,ikinci-anahtar,
Hafta 8	Vize
Hafta 9	İlişkisel Veri Tabanı Tasarımı ve Normalizasyon
Hafta 10	yarı-yapısal veri modelleri, XML
Hafta 11	JSON
Hafta 12	İlişkisel olmayan DB, NoSQL
Hafta 13	NoSQL
Hafta 14	DBMS -Eşzamanlılık (Concurrency) Kontrolü

VERİ TABANI NEDİR?

Veritabanı genellikle elektronik olarak bir bilgisayar sisteminde depolanan yapılandırılmış bilgi veya veriden oluşan düzenli bir koleksiyondur. Veritabanı genellikle bir veritabanı yönetim sistemi (DBMS) ile kontrol edilir. Veri ve DBMS ve aynı zamanda bunlarla ilişkili uygulama yazılımları bir araya getirildiğinde sıklıkla yalnızca veritabanı olarak kısaltılan veritabanı sistemi olarak ifade edilir.



VERİ TABANI NEDİR?

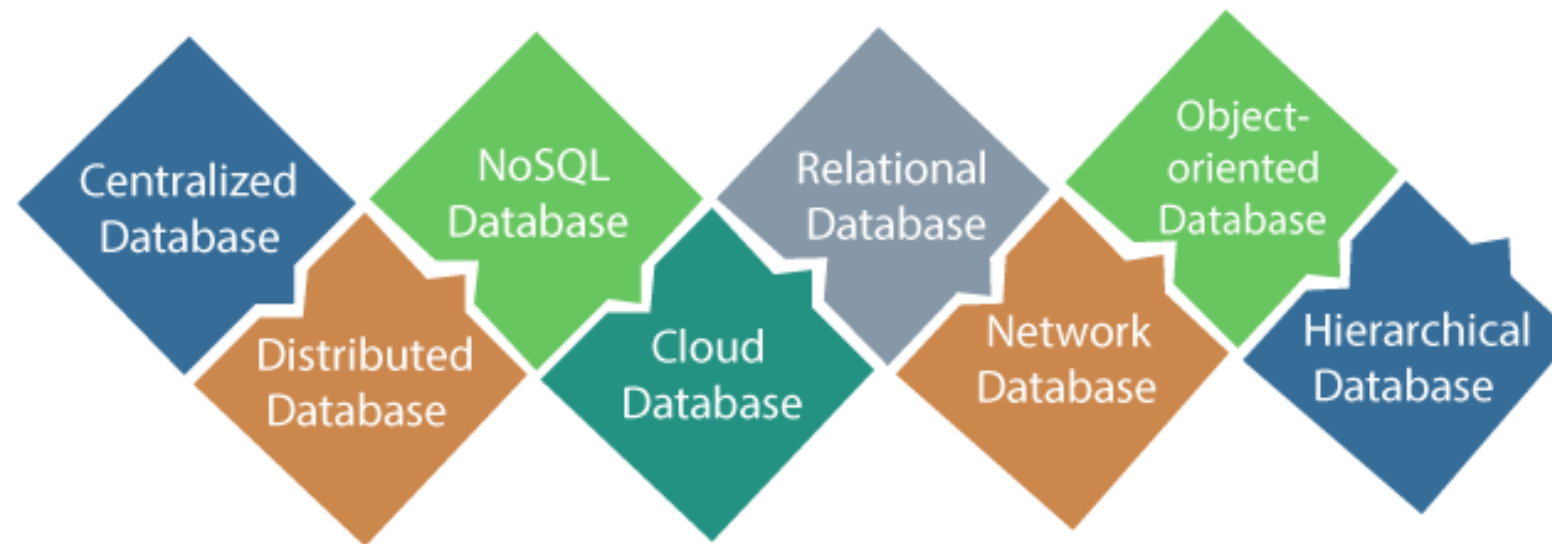
Bir veritabanı sistemi, birbiriyle ilişkili veriler topluluğu ve kullanıcıların bu verilere erişmesine ve bunları değiştirmesine izin veren bir dizi programdır. Bir veritabanı sisteminin temel amacı, kullanıcılara verilerin soyut bir görünümünü sağlamaktır. Yani sistem, verilerin nasıl saklandığına ve korunduğuna dair belirli ayrıntıları gizler.



İlişkisel Model

Görüntüleme seviyesi. En yüksek soyutlama düzeyi, tüm veritabanının yalnızca bir bölümünü tanımlar. Mantıksal düzeyde daha basit yapılar kullanılsa da, büyük bir veritabanında depolanan bilgilerin çeşitliliği nedeniyle karmaşıklık devam eder. Veritabanı sisteminin birçok kullanıcısı tüm bu bilgilere ihtiyaç duymaz; bunun yerine, veritabanının yalnızca bir kısmına erişimleri gerekir. Soyutlamanın görünüm seviyesi, sistemle etkileşimlerini basitleştirmek için mevcuttur. Sistem aynı veritabanı için birçok görünüm sağlayabilir.

Veritabanı Türleri



Merkezi Veritabanı

Verileri merkezi bir veritabanı sisteminde depolayan veritabanı türüdür. Kullanıcıların, depolanan verilere farklı konumlardan çeşitli uygulamalar aracılığıyla erişmesini kolaylaştırır. Bu uygulamalar, kullanıcıların verilere güvenli bir şekilde erişmesine izin veren kimlik doğrulama sürecini içerir. Merkezi bir veritabanına bir örnek, bir kolejdeki / üniversitedeki her bir kütüphanenin merkezi bir veritabanını taşıyan Merkez Kütüphanesi olabilir..

Merkezi Veritabanı

Merkezi Veritabanının Avantajları

- Veri yönetimi riskini azaltmıştır, yani verilerin manipüle edilmesi temel verileri etkilemeyecektir.
- Merkezi bir depodaki verileri yönetirken veri tutarlılığı korunur.
- Kuruluşların veri standartları oluşturmalarını sağlayan daha iyi veri kalitesi sağlar.
- Daha az maliyetlidir çünkü veri setlerini işlemek için daha az satıcı gerekir.

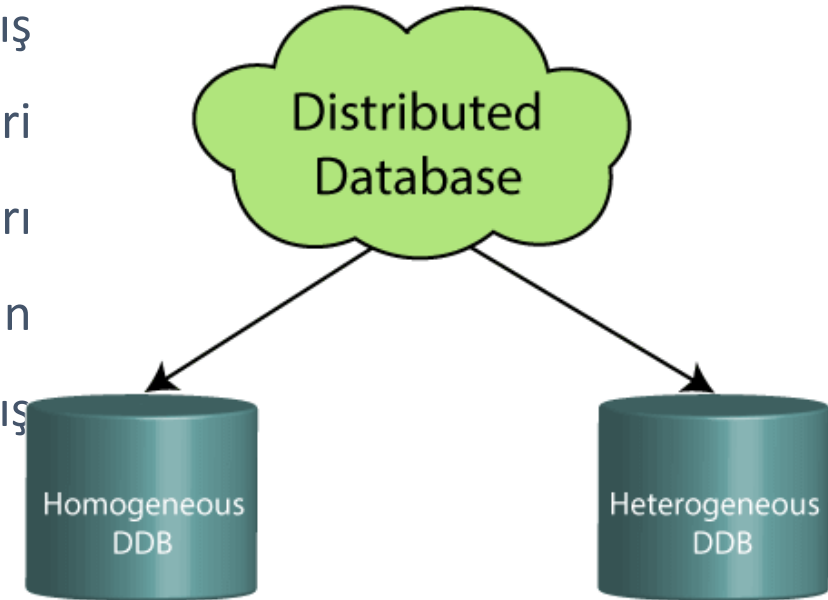
Merkezi Veritabanının Dezavantajları

- Merkezi veritabanının boyutu büyüktür, bu da verilerin alınması için yanıt süresini artırır.
- Böylesine kapsamlı bir veritabanı sistemini güncellemek kolay değil.
- Herhangi bir sunucu arızası meydana gelirse, tüm veriler kaybolur ve bu da büyük bir kayıp olabilir.



2) Dağıtılmış Veritabanı

Merkezi bir veritabanı sisteminden farklı olarak, dağıtılmış sistemlerde veriler, bir kuruluşun farklı veritabanı sistemleri arasında dağıtılır. Bu veritabanı sistemleri iletişim bağlantıları ile birbirine bağlanır. Bu tür bağlantılar, son kullanıcıların verilere kolayca erişmesine yardımcı olur. Örnekler dağıtılmış veri tabanı gibi Apache Cassandra HBase,



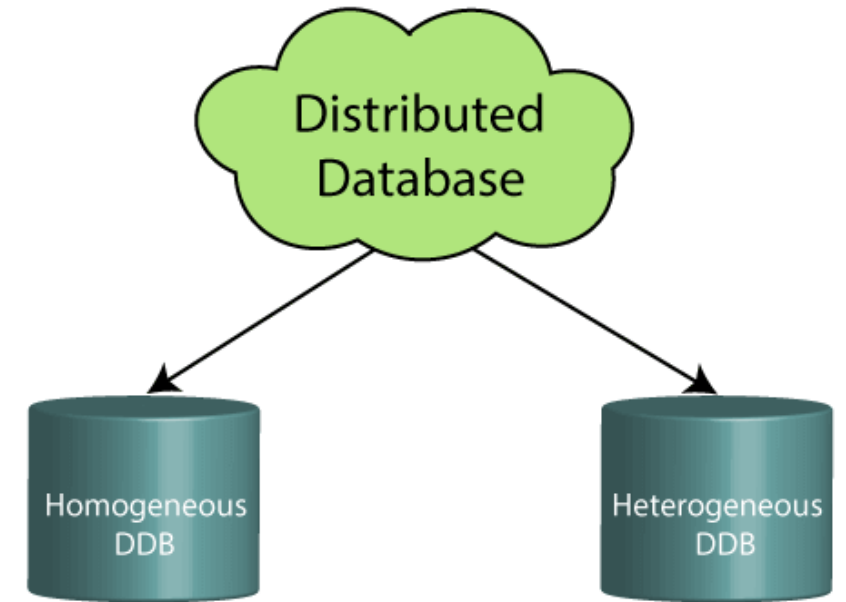
2) Dağıtılmış Veritabanı

Homojen DDB: Aynı işletim sistemi üzerinde çalışan ve aynı uygulama sürecini kullanan ve aynı donanım cihazlarını taşıyan veritabanı sistemleridir.

Heterojen DDB: Farklı uygulama prosedürleri altında farklı işletim sistemleri üzerinde çalışan ve farklı donanım aygıtları taşıyan veritabanı sistemleridir.

Dağıtık Veritabanının Avantajları

- Dağıtık bir veri tabanında modüler geliştirme mümkündür, yani yeni bilgisayarlar dahil edilerek ve dağıtık sisteme bağlanarak sistem genişletilebilir.
- Tek bir sunucu hatası, tüm veri kümesini etkilemeyecektir,



3) İlişkisel Veritabanı

Bu veritabanı, verileri satırlar (tuple) ve sütunlar (öznitelikler) biçiminde depolayan ve birlikte bir tablo (ilişki) oluşturan ilişkisel veri modeline dayanmaktadır. İlişkisel bir veritabanı, verileri depolamak, işlemek ve korumak için SQL kullanır. EF Codd, veritabanını 1970 yılında icat etti. Veritabanındaki her tablo, verileri diğerlerinden benzersiz kılan bir anahtar taşır. Örnekler ilişkisel veritabanlarının vb MySQL, Microsoft SQL Server, Oracle, vardır



3) İlişkisel Veritabanı

İlişkisel Veritabanının Özellikleri

ACID özellikleri olarak bilinen bir ilişkisel modelin yaygın olarak bilinen dört özelliği vardır; burada:

A, Atomiklik anlamına gelir: Bu, veri işleminin başarılı veya başarısız bir şekilde tamamlanmasını sağlar. "Ya hep ya hiç" stratejisini izler. Örneğin, bir işlem gerçekleştirilecek veya iptal edilecektir.

C Tutarlılık demektir: Veriler üzerinde herhangi bir işlem yaparsak, işlem öncesi ve sonrası değeri korunmalıdır. Örneğin, işlem öncesi ve sonrası hesap bakiyesi doğru olmalı yani korunmuş kalmalıdır.

3) İlişkisel Veritabanı

İlişkisel Veritabanının Özellikleri

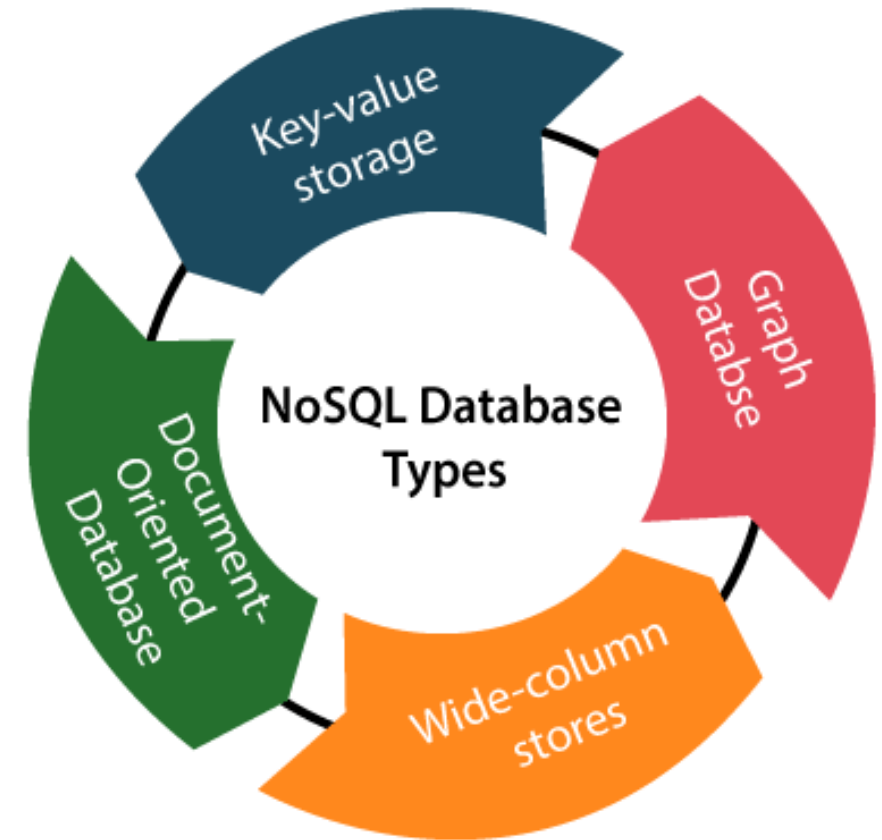
ACID özellikleri olarak bilinen bir ilişkisel modelin yaygın olarak bilinen dört özelliği vardır; burada:

I İzolasyon demek : Veri tabanından aynı anda veri erişimi için eşzamanlı kullanıcılar olabilir. Bu nedenle, veriler arasındaki izolasyon izole kalmalıdır. Örneğin, aynı anda birden fazla işlem gerçekleştiğinde, bir işlemin etkisi veri tabanındaki diğer işlemlere görünmemelidir.

D Dayanıklılık anlamına gelir: İşlemi tamamlayıp verileri işlediğinde veri değişikliklerinin kalıcı kalmasını sağlar.

4) NoSQL Database

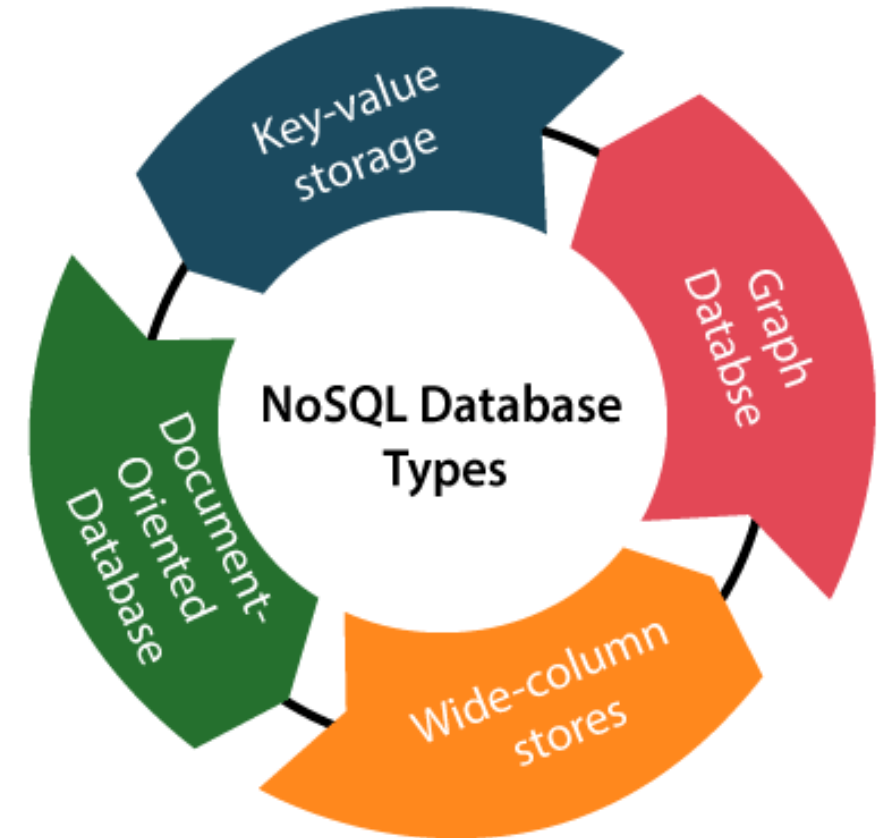
Non-SQL / Not Only SQL, çok çeşitli veri kümelerini depolamak için kullanılan bir veritabanı türüdür. Verileri yalnızca tablo biçiminde değil, birkaç farklı şekilde depoladığı için ilişkisel bir veritabanı değildir. Modern uygulamalar inşa etme talebi arttığında ortaya çıktı. Bu nedenle, NoSQL, taleplere yanıt olarak çok çeşitli veritabanı teknolojileri sundu. Bir NoSQL veritabanını aşağıdaki dört türe ayırabiliriz:



4) NoSQL Database

Anahtar-değer depolaması: Her bir öğeyi değerini bir arada tutan bir anahtar (veya öznitelik adı) olarak depoladığı en basit veritabanı depolama türüdür.

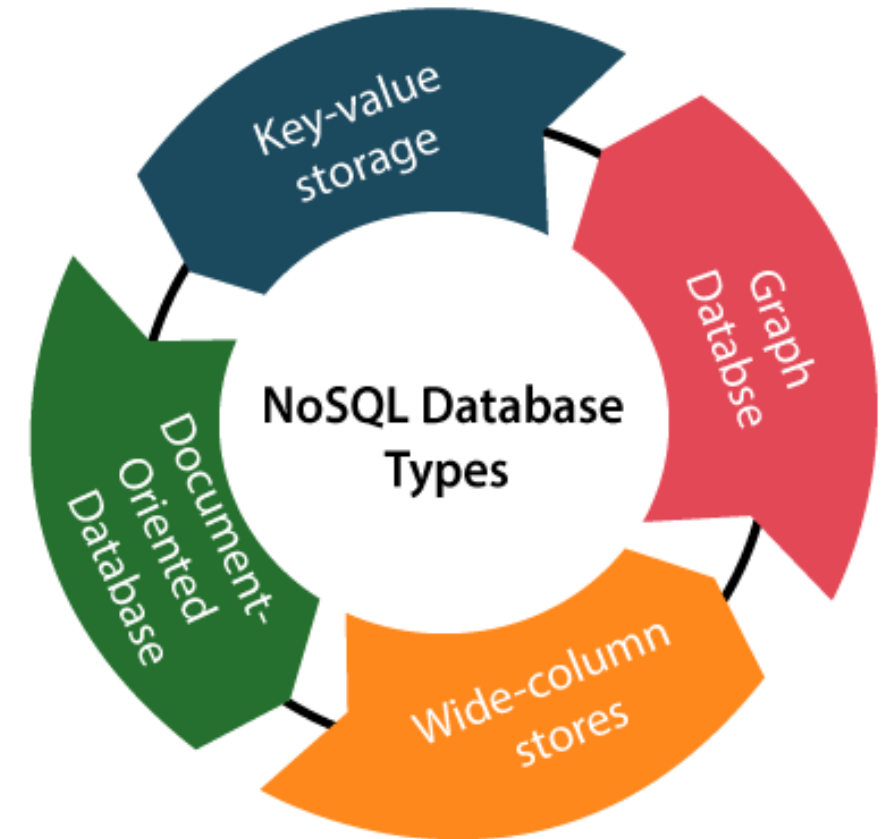
Belge odaklı Veritabanı: Verileri JSON benzeri belge olarak depolamak için kullanılan bir veritabanı türüdür. Geliştiricilerin, uygulama kodunda kullanılan aynı belge modeli biçimini kullanarak verileri depolamasına yardımcı olur.



4) NoSQL Database

Grafik Veritabanları: Büyük miktarda veriyi grafik benzeri bir yapıda saklamak için kullanılır. En yaygın olarak, sosyal ağ siteleri grafik veritabanını kullanır.

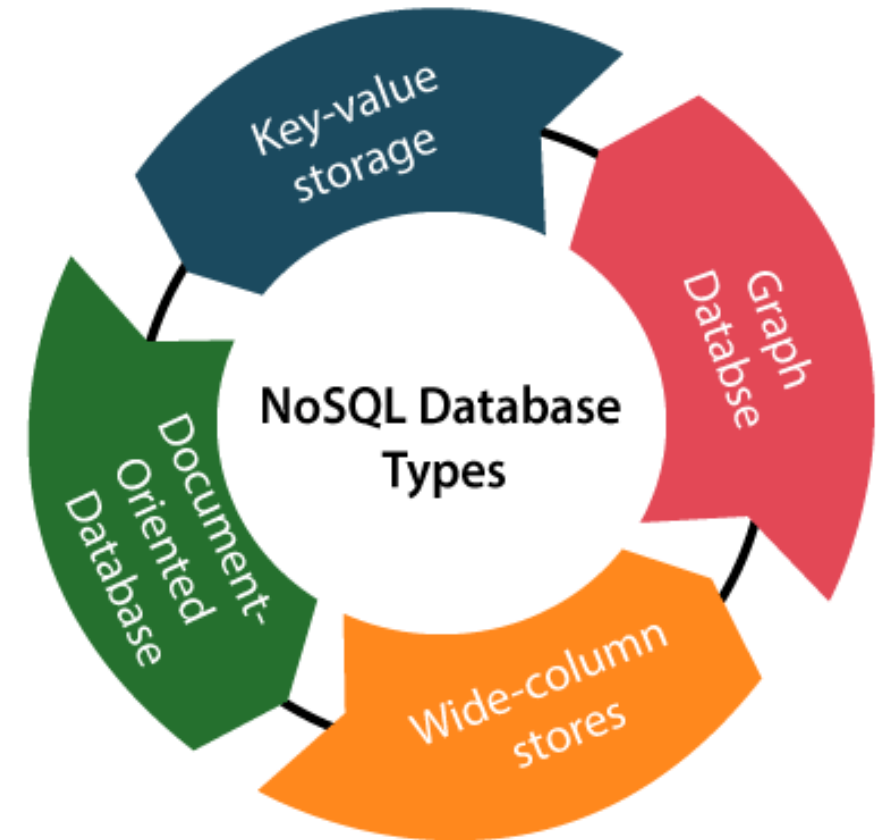
Geniş sütun depoları: İlişkisel veritabanlarında temsil edilen verilere benzer. Burada veriler, satırlar halinde depolanmak yerine büyük sütunlarda birlikte depolanır.



4) NoSQL Database

NoSQL Veritabanının Avantajları

- Verilerin yapılandırılmış bir biçimde depolanması gerekmediğinden, uygulama geliştirmede iyi bir üretkenlik sağlar.
- Büyük veri kümelerini yönetmek ve işlemek için daha iyi bir seçenektir.
- Yüksek ölçeklenebilirlik sağlar.
- Kullanıcılar, anahtar / değer çifti aracılığıyla veritabanındaki verilere hızlı bir şekilde erişebilir.



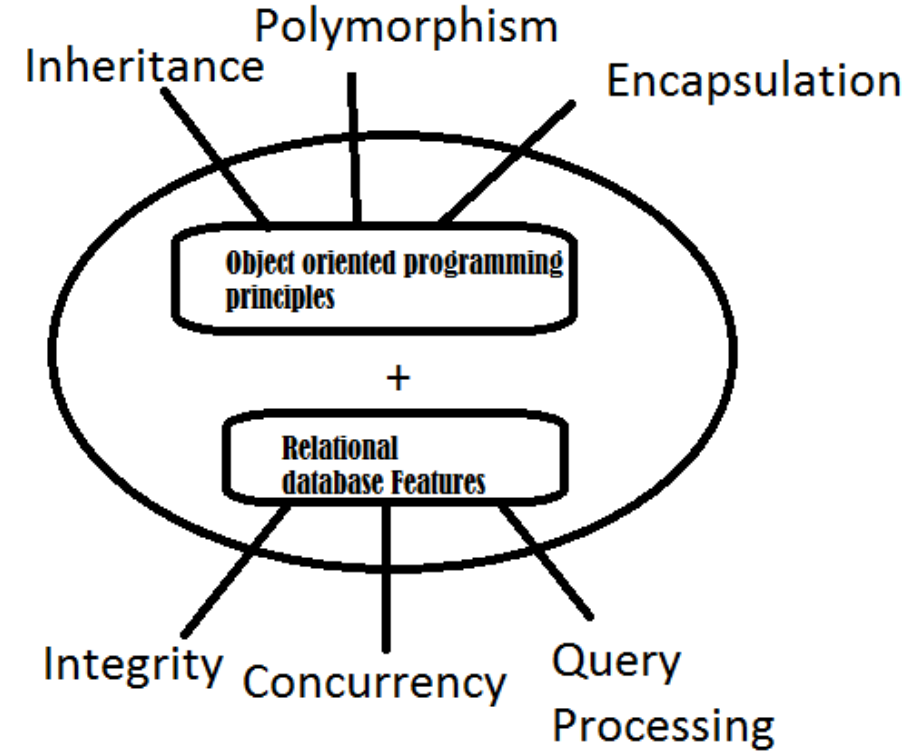
5) Bulut Veritabanı

Verilerin sanal bir ortamda depolandığı ve bulut bilişim platformu üzerinden yürütüldüğü bir veritabanı türü. Kullanıcılara veritabanına erişim için çeşitli bulut bilişim hizmetleri (SaaS, PaaS, IaaS, vb.) Sağlar. Çok sayıda bulut platformu var, ancak en iyi seçenekler:

- Amazon Web Hizmetleri (AWS)
- Microsoft Azure
- Kamatera
- PhonixNAP
- ScienceSoft
- Google Cloud SQL vb.

6) Nesneye Yönelik Veritabanları

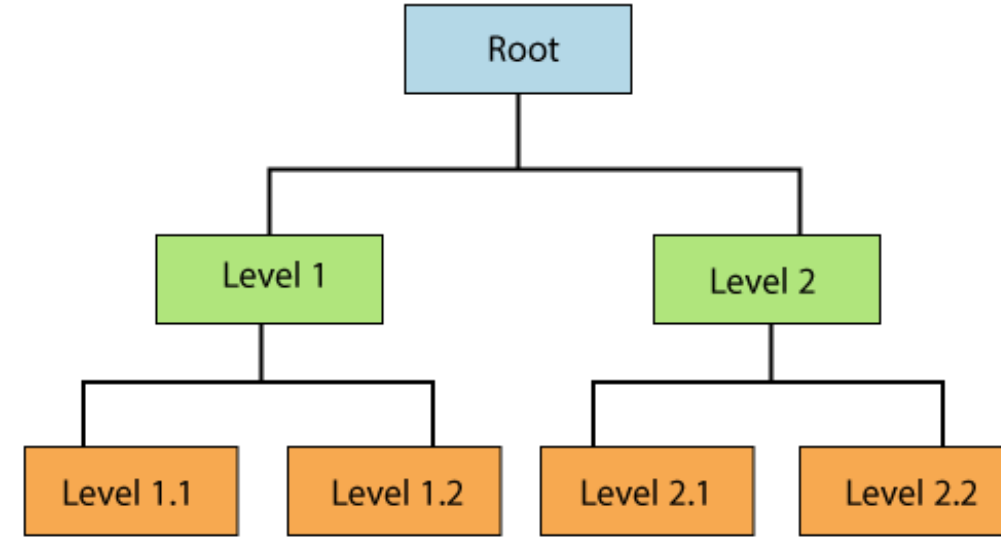
Veri tabanı sisteminde veri depolamak için nesne tabanlı veri modeli yaklaşımını kullanan veri tabanı türü. Veriler, nesne yönelimli programlama dili kullanılan nesnelere benzer nesneler olarak temsil edilir ve saklanır.



7) Hiyerarşik Veritabanları

Verileri ebeveyn-çocuk ilişki düğümleri şeklinde depolayan veritabanı türüdür. Burada, verileri ağaç benzeri bir yapıda düzenler.

Veriler, bağlantılarla bağlanan kayıtlar biçiminde depolanır. Ağaçtaki her alt kayıt yalnızca bir ebeveyn içerecektir. Öte yandan, her ana kaydın birden çok alt kaydı olabilir.



Hierarchical Database

Veritabanı Yönetim Sistemi (DBMS)

Bir DBMS kullanıcılarının depolamak işlem ve kolay bir veri analiz sağlayarak, oluşturma, tanımı ve veritabanı işleme olanak sağlayan bir yazılım. DBMS, veritabanı oluşturma, içinde veri depolama, verileri güncelleme, veritabanında tablo oluşturma ve çok daha fazlası gibi çeşitli işlemleri gerçekleştirmek için bize bir arayüz veya bir araç sağlar.

DBMS ayrıca veritabanlarına koruma ve güvenlik sağlar. Ayrıca birden fazla kullanıcı olması durumunda veri tutarlılığını korur.

İşte bu günlerde kullanılan popüler DBMS'lerin bazı örnekleri:

MySQL

IBM DB2

Oracle

PostgreSQL

SQL Server

Amazon SimpleDB (bulut tabanlı) vb.



Veritabanı Yönetim Sistemi (DBMS)

Özellikler

Modern bir DBMS aşağıdaki özelliklere sahiptir -

Gerçek dünya varlığı - Modern bir DBMS daha gerçekçidir ve mimarisini tasarlamak için gerçek dünyadaki varlıkları kullanır. Davranışı ve nitelikleri de kullanır. Örneğin, bir okul veritabanı öğrencileri bir varlık olarak ve yaşlarını bir öznitelik olarak kullanabilir.

İlişkiye dayalı tablolar - DBMS, varlıkların ve aralarındaki ilişkilerin tablolar oluşturmaya izin verir. Bir kullanıcı bir veritabanının mimarisini sadece tablo adlarına bakarak anlayabilir.

Veri izolasyonu ve uygulama - Bir veritabanı sistemi, verilerinden tamamen farklıdır. Bir veritabanı aktif bir varlıktır, oysa verilerin üzerinde çalıştığı ve düzenlediği pasif olduğu söylenir. DBMS ayrıca kendi sürecini kolaylaştırmak için verilerle ilgili veriler olan meta verileri de depolar.

Veritabanı Yönetim Sistemi (DBMS)

Özellikler

Daha az fazlalık - DBMS, niteliklerinden herhangi biri değerlerde fazlalık olduğunda bir ilişkiyi bölen normalleştirme kurallarına uyar. Normalleştirme, veri fazlalığını azaltan matematiksel açıdan zengin ve bilimsel bir süreçtir.

Tutarlılık - Tutarlılık, bir veritabanındaki her ilişkinin tutarlı kaldığı bir durumdur. Veritabanını tutarsız durumda bırakma girişimini tespit edebilen yöntemler ve teknikler vardır. Bir DBMS, dosya işleme sistemleri gibi önceki veri depolama uygulamaları formlarına kıyasla daha fazla tutarlılık sağlayabilir.

Sorgu Dili - DBMS, verileri almayı ve değiştirmeyi daha verimli hale getiren sorgu diliyle donatılmıştır. Bir kullanıcı, bir veri kümesini almak için gerektiği kadar çok ve farklı filtreleme seçeneği uygulayabilir. Geleneksel olarak, dosya işleme sisteminin kullanıldığı yerlerde bu mümkün değildi.

Veritabanı Yönetim Sistemi (DBMS)

Özellikler

ACID Özellikleri - DBMS kavramlarını aşağıdaki bir Atomicity, Consistency, Isolation ve Durability (normal olarak, asit olarak kısaltılmıştır). Bu kavramlar, bir veritabanındaki verileri işleyen işlemlere uygulanır. ACID özellikleri, veritabanının çok işlemleri ortamlarda ve hata durumunda sağlıklı kalmasına yardımcı olur.

Çok Kullanıcı ve Eşzamanlı Erişim - DBMS, çok kullanıcı ortamı destekler ve verilere paralel olarak erişmelerine ve bunları değiştirmelerine olanak tanır. Kullanıcılar aynı veri ögesini işlemeye çalıştığında işlemlerde kısıtlamalar olsa da, kullanıcılar her zaman bunların farkında değildir.

Veritabanı Yönetim Sistemi (DBMS)

Özellikler

Birden çok görünüm - DBMS, farklı kullanıcılar için birden çok görünüm sunar. Satış departmanındaki bir kullanıcı, Üretim departmanında çalışan bir kişiden farklı bir veritabanı görünümüne sahip olacaktır. Bu özellik, kullanıcıların ihtiyaçlarına göre veritabanının konsantre bir görünümüne sahip olmalarını sağlar.

Güvenlik - Çoklu görünüm gibi özellikler, kullanıcıların diğer kullanıcıların ve departmanların verilerine erişemediği durumlarda bir dereceye kadar güvenlik sağlar.

Veritabanı Yönetim Sistemi (DBMS)

Özellikler

Birden çok görünüm - DBMS, farklı kullanıcılar için birden çok görünüm sunar. Satış departmanındaki bir kullanıcı, Üretim departmanında çalışan bir kişiden farklı bir veritabanı görünümüne sahip olacaktır. Bu özellik, kullanıcıların ihtiyaçlarına göre veritabanının konsantre bir görünümüne sahip olmalarını sağlar.

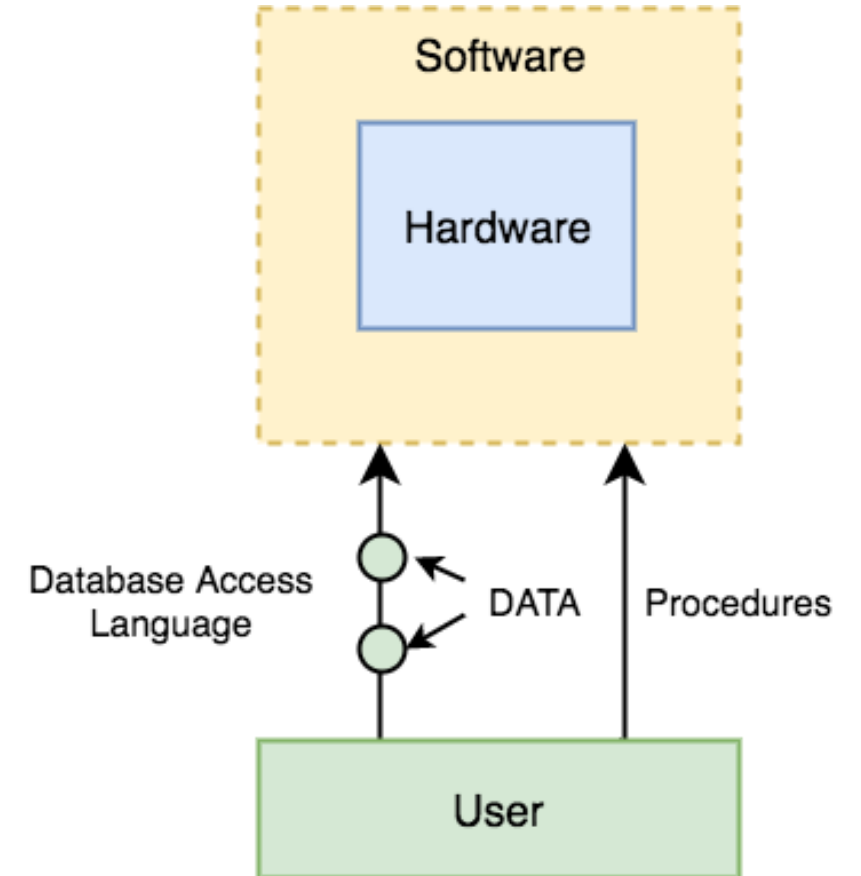
Güvenlik - Çoklu görünüm gibi özellikler, kullanıcıların diğer kullanıcıların ve departmanların verilerine erişemediği durumlarda bir dereceye kadar güvenlik sağlar.

DBMS Bileşenleri

Özellikler

Veritabanı yönetim sistemi beş ana bileşene ayrılabilir, bunlar:

- Donanım
- Yazılım
- Veri
- Prosedürler
- Veritabanı Erişim Dili



DBMS - Mimari

Bir DBMS'nin mimarisi, tek katmanlı veya çok katmanlı olarak görülebilir. Bir n katmanlı mimari, tüm sistemi , bağımsız olarak değiştirilebilen, değiştirilebilen, değiştirilebilen veya değiştirilebilen ilgili ancak bağımsız n modüle böler.

1-katmanlı DBMS mimarisi de mevcuttur, bu, veritabanının verileri depolamak için kullanması için doğrudan kullanıcıya sunulduğu zamandır. Genellikle böyle bir kurulum, programcılarının hızlı yanıt için doğrudan veritabanı ile iletişim kurduğu yerel uygulama geliştirme için kullanılır.

Veritabanı Mimarisi mantıksal olarak iki türdendir:

1.2 katmanlı DBMS mimarisi

2.3 katmanlı DBMS mimarisi



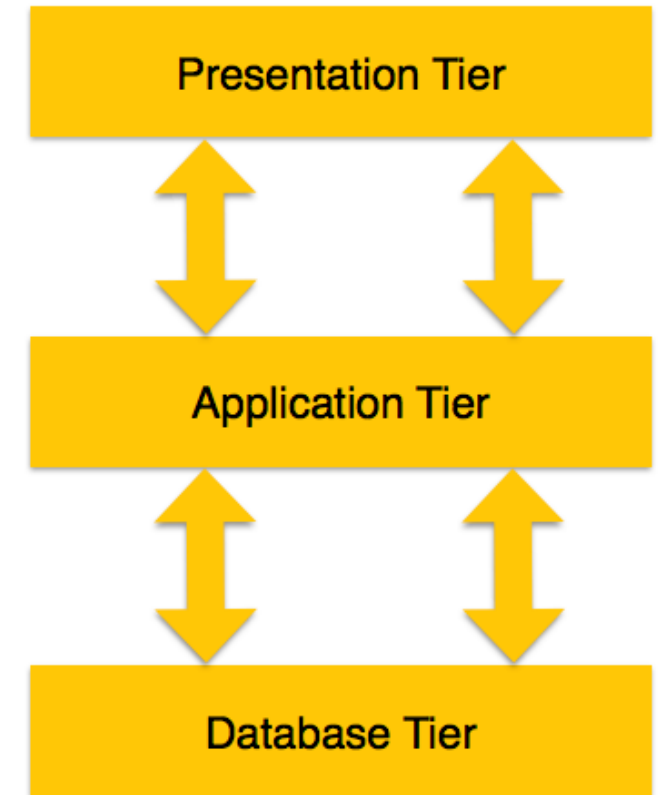
DBMS - Mimari

3 katmanlı DBMS mimarisi

2 katmanlı mimarinin bir uzantısıdır. 2 katmanlı mimaride, DBMS üzerinde çeşitli işlemleri gerçekleştirmek için programlı olarak erişilebilen bir uygulama katmanımız vardır. Uygulama genellikle Veritabanı Erişim Dilini anlar ve son kullanıcıların DBMS'ye yönelik isteklerini işler.

3 katmanlı mimari, kullanıcıların karmaşıklığına ve veritabanında bulunan verileri nasıl kullandıklarına bağlı olarak katmanlarını birbirinden ayırır. Bir DBMS tasarlamak için en yaygın kullanılan mimaridir.

PHPMyAdmin 3 katmanlı bir DBMS mimarisinin en iyi örneğidir.

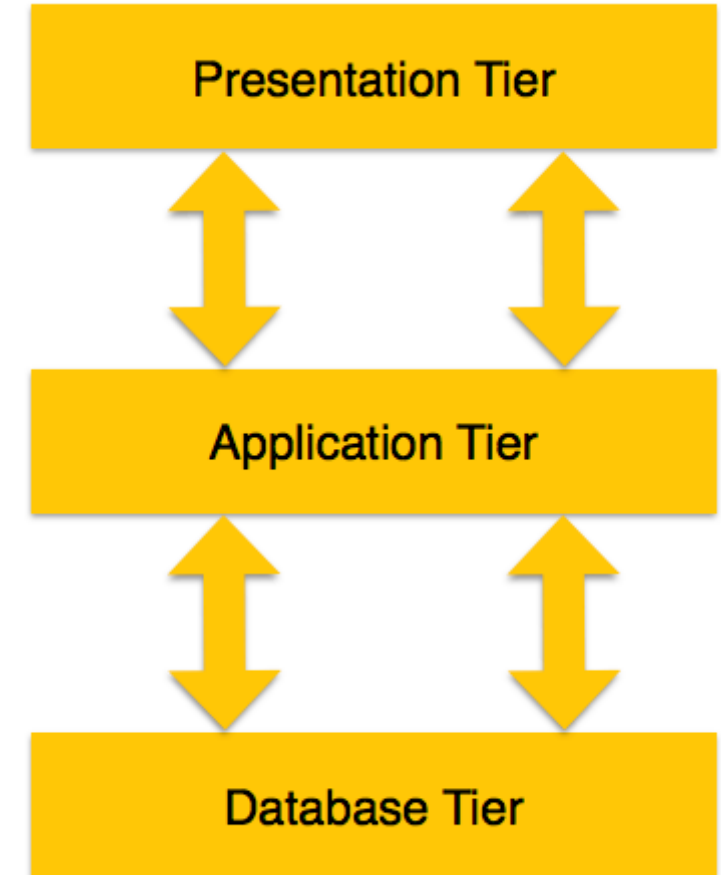


DBMS - Mimari

3 katmanlı DBMS mimarisi

Veritabanı (Veri) Katmanı - Bu katmanda veritabanı, sorgu işleme dilleriyle birlikte bulunur. Ayrıca bu düzeyde verileri ve kısıtlamalarını tanımlayan ilişkilere sahibiz.

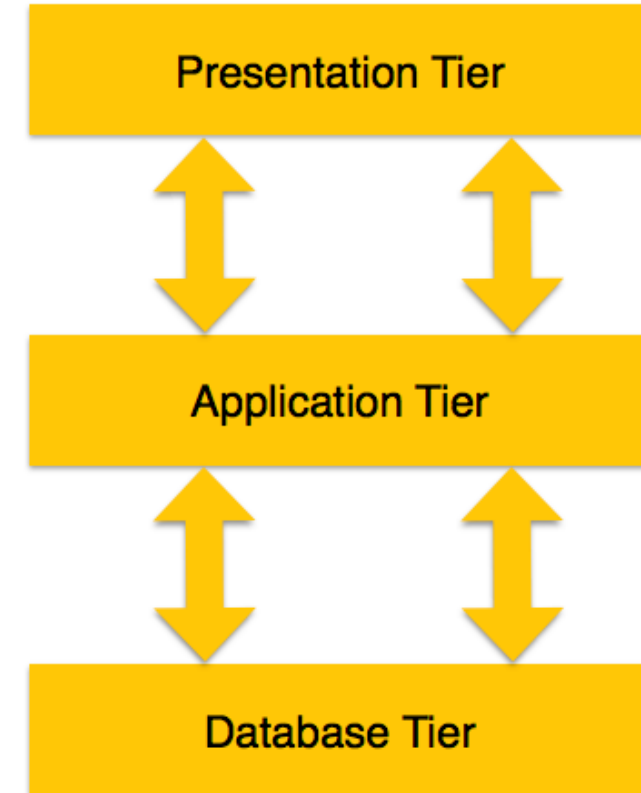
Uygulama (Orta) Katmanı - Bu katmanda, uygulama sunucusu ve veritabanına erişen programlar bulunur. Bir kullanıcı için bu uygulama katmanı, veritabanının soyutlanmış bir görünümünü sunar. Son kullanıcılar, uygulama dışında veritabanının varlığından habersizdir. Diğer uçta, veritabanı katmanı, uygulama katmanının dışındaki herhangi bir kullanıcının farkında değildir. Bu nedenle, uygulama katmanı ortada oturur ve son kullanıcı ile veritabanı arasında bir aracı görevi görür.



DBMS - Mimari

3 katmanlı DBMS mimarisi

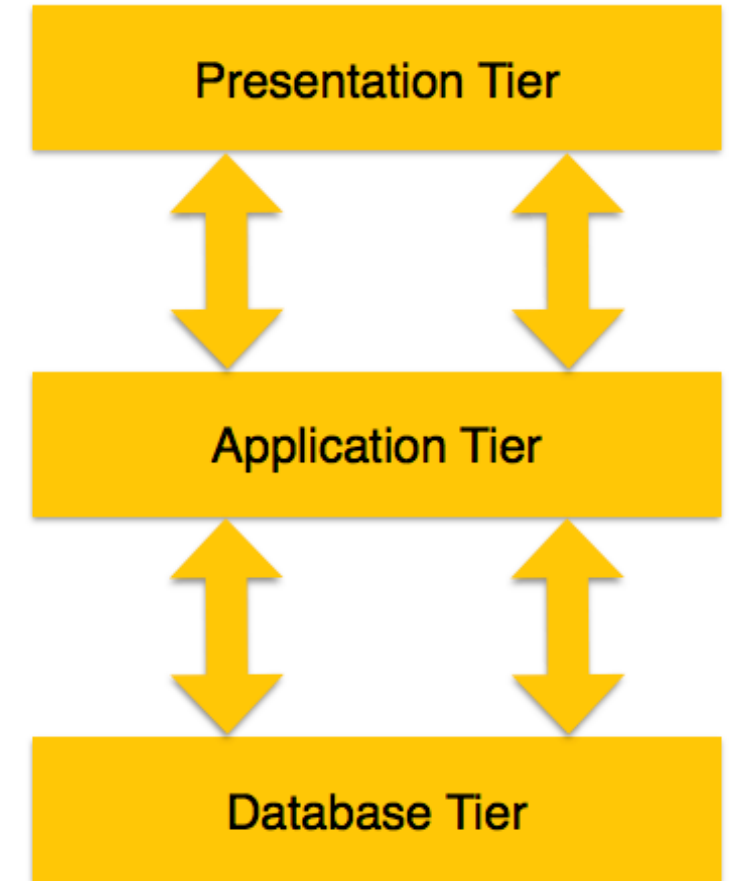
Kullanıcı (Sunum) Katmanı - Son kullanıcılar bu katmanda çalışır ve bu katmanın ötesinde veritabanının varlığıyla ilgili hiçbir şey bilmiyorlar. Bu katmanda, veritabanının birden çok görünümü uygulama tarafından sağlanabilir. Tüm görünümler, uygulama katmanında bulunan uygulamalar tarafından oluşturulur.



DBMS - Mimari

3 katmanlı DBMS mimarisi

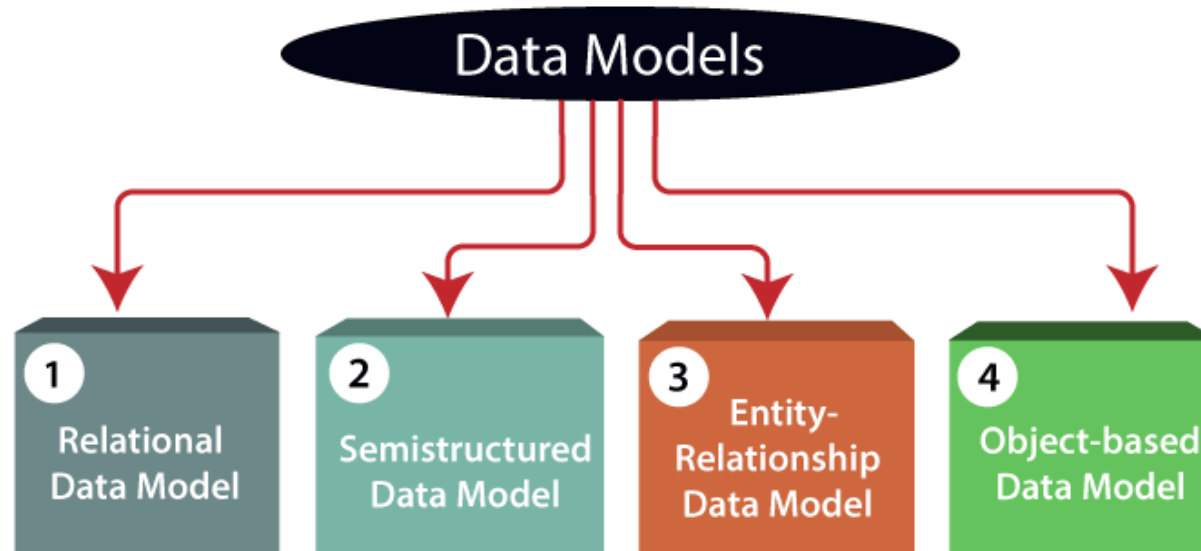
Kullanıcı (Sunum) Katmanı - Son kullanıcılar bu katmanda çalışır ve bu katmanın ötesinde veritabanının varlığıyla ilgili hiçbir şey bilmiyorlar. Bu katmanda, veritabanının birden çok görünümü uygulama tarafından sağlanabilir. Tüm görünümler, uygulama katmanında bulunan uygulamalar tarafından oluşturulur.



DBMS Veritabanı Modelleri

DBMS Veritabanı Modelleri

Veri Modeli, veri tanımının, veri anlamının ve verilerin tutarlılık kısıtlamalarının modellenmesidir. Her veri soyutlama düzeyinde bir veritabanı tasarımını açıklamak için kavramsal araçlar sağlar. Bu nedenle, veritabanının yapısını anlamak için kullanılan aşağıdaki dört veri modeli vardır:



DBMS Veritabanı Modelleri

1) İlişkisel Veri Modeli: Bu model türü, verileri bir tablo içindeki satırlar ve sütunlar şeklinde tasarlar. Bu nedenle, ilişkisel bir model, verileri ve aradaki ilişkileri temsil etmek için tabloları kullanır. Tablolara ilişkiler de denir. Bu model ilk olarak 1969'da Edgar F. Codd tarafından tanımlanmıştır. İlişkisel veri modeli, esas olarak ticari veri işleme uygulamaları tarafından kullanılan ve yaygın olarak kullanılan modeldir.

DBMS Veritabanı Modelleri

2) Varlık-İlişki Veri Modeli: Bir ER modeli, verilerin nesneler ve bunlar arasındaki ilişkiler olarak mantıksal temsilidir. Bu nesneler varlıklar olarak bilinir ve ilişki bu varlıklar arasındaki bir ilişkidir. Bu model Peter Chen tarafından tasarlandı ve 1976 makalelerinde yayınlandı. Veritabanı tasarımında yaygın olarak kullanılmıştır. Bir dizi nitelik varlıkları tanımlar. Örneğin, öğrenci_adı, öğrenci_kimliği 'öğrenci' varlığını tanımlar. Aynı tür varlıklardan oluşan bir dizi 'Varlık kümesi' olarak bilinir ve aynı türden ilişkiler kümesi 'ilişki kümesi' olarak bilinir.

DBMS Veritabanı Modelleri

3) Nesne Tabanlı Veri Modeli: ER modelinin işlevler, kapsülleme ve nesne kimliği kavramlarını içeren bir uzantısıdır. Bu model, yapılandırılmış ve koleksiyon türlerini içeren zengin bir tür sistemi destekler. Böylece 1980'lerde nesne yönelimli yaklaşımı izleyen çeşitli veri tabanı sistemleri geliştirilmiştir. Burada nesneler, özelliklerini taşıyan verilerden başka bir şey değildir.

DBMS Veritabanı Modelleri

4) Yarı Yapılandırılmış Veri Modeli: Bu tür veri modeli, diğer üç veri modelinden farklıdır (yukarıda açıklanmıştır). Yarı yapılandırılmış veri modeli, aynı tipteki münferit veri öğelerinin farklı öznitelik kümelerine sahip olabileceği yerlerde veri spesifikasyonlarına izin verir. XML olarak da bilinen Genişletilebilir İşaretleme Dili, yarı yapılandırılmış verileri temsil etmek için yaygın olarak kullanılır. XML başlangıçta biçimlendirme bilgisini metin belgesine dahil etmek için tasarlanmış olsa da, veri alışverişinde uygulaması nedeniyle önem kazanmaktadır.

DBMS Veritabanı Modelleri

Veritabanı modeli, bir veritabanının mantıksal tasarımını ve yapısını tanımlar ve verilerin bir veritabanı yönetim sisteminde nasıl depolanacağını, erişileceğini ve güncelleneceğini tanımlar. İken ilişkisel Model en yaygın kullanılan veritabanı modeli, diğer modeller de vardır:

- Hiyerarşik Model
- Ağ Modeli
- Varlık-iliski Modeli
- İlişkisel Model

DBMS Veritabanı Modelleri

Hiyerarşik Model

Bu veritabanı modeli, verileri, diğer tüm verilerin bağlı olduğu tek bir kök ile ağaç benzeri bir yapı halinde düzenler. Hiyerarşi, Kök verilerden başlar ve bir ağaç gibi genişleyerek ana düğümlere alt düğümler ekler.

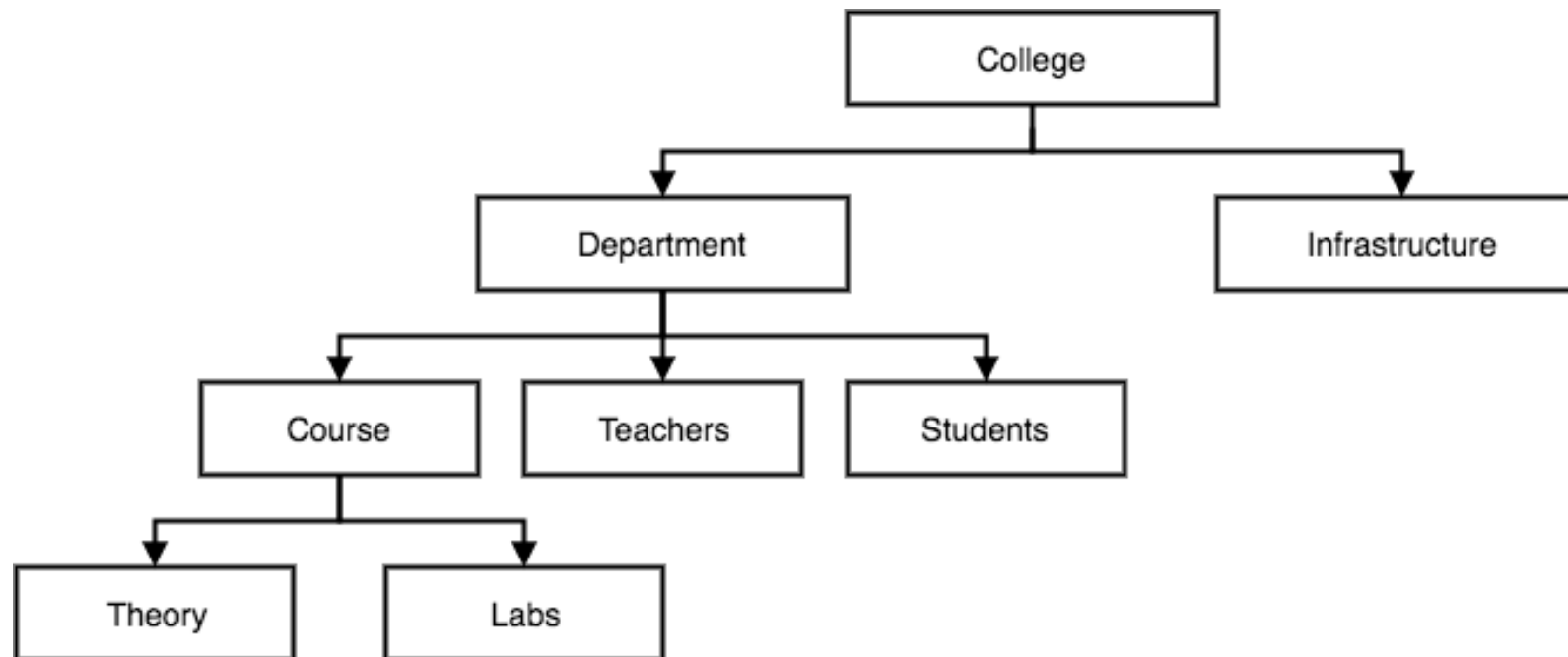
Bu modelde, bir alt düğüm yalnızca tek bir ana düğüme sahip olacaktır.

Bu model, bir kitabın dizini, tarifler vb. Gibi birçok gerçek dünya ilişkisini verimli bir şekilde tanımlar.

Hiyerarşik modelde veriler, iki farklı veri türü arasında bire çok ilişki ile ağaç benzeri bir yapı halinde düzenlenir; örneğin, bir bölümün birçok dersi olabilir, birçok profesör ve tabii ki birçok öğrenci olabilir.

DBMS Veritabanı Modelleri

Hiyerarşik Model



DBMS Veritabanı Modelleri

Ağ Modeli

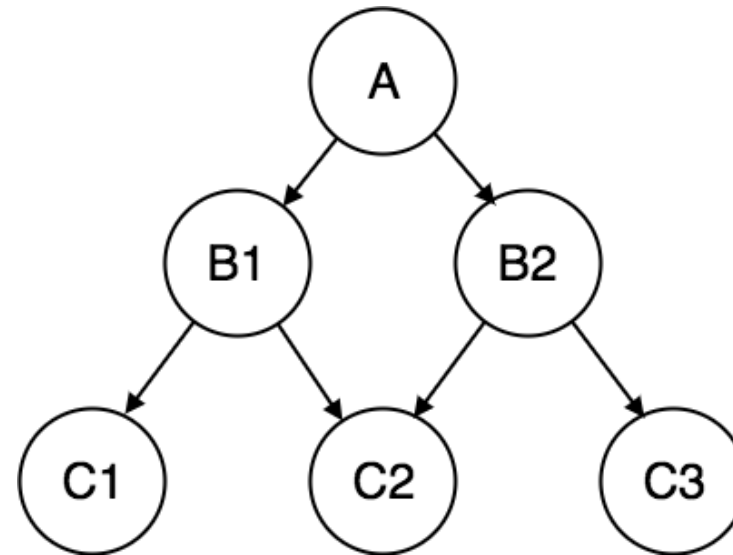
Bu, Hiyerarşik modelin bir uzantısıdır. Bu modelde veriler daha çok bir grafik gibi düzenlenir ve birden fazla ana düğüme sahip olmasına izin verilir.

Bu veritabanı modelinde, bu veritabanı modelinde daha fazla ilişki kurulduğu için veriler daha fazla ilişkilidir. Ayrıca, veriler daha ilişkili olduğundan, verilere erişim de daha kolay ve hızlıdır. Bu veritabanı modeli, çoktan çoğa veri ilişkilerini eşlemek için kullanıldı.

Bu, İlişkisel Model tanıtılmadan önce en yaygın kullanılan veritabanı modeliydi.

DBMS Veritabanı Modelleri

Ağ Modeli



DBMS Veritabanı Modelleri

Varlık-ilişki Modeli

Bu veritabanı modelinde ilişkiler, ilgilenilen nesnenin varlığa ve özelliklerinin özniteliklere bölünmesiyle oluşturulur.

Farklı varlıklar ilişkiler kullanılarak ilişkilendirilir.

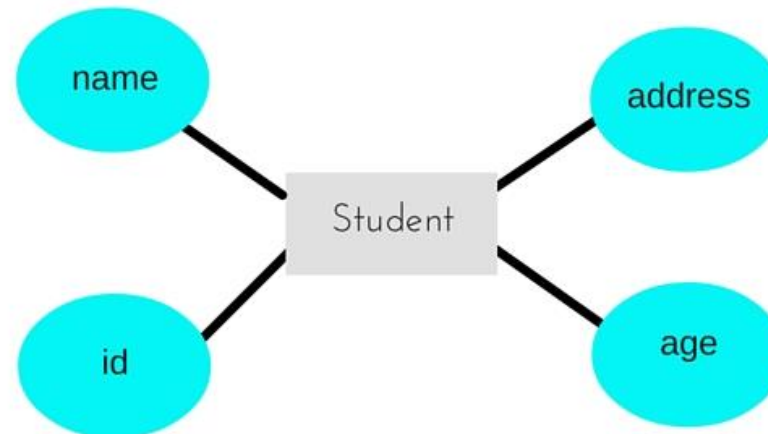
ER Modelleri, farklı paydaşların anlamasını kolaylaştırmak için ilişkileri resimli biçimde temsil edecek şekilde tanımlanmıştır.

Bu model, daha sonra ilişkisel modelde tablolara dönüştürülebilecek bir veritabanı tasarlamak için iyidir.

DBMS Veritabanı Modelleri

Varlık-ilişki Modeli

Bir örnek alalım, Bir Okul Veritabanı tasarlamamız gerekiyorsa, Öğrenci adı, yaşı, adresi vb. Niteliklere sahip bir varlık olacaktır . Adres genel olarak karmaşık olduğundan, sokak adı, pin kodu, şehir vb. Niteliklere sahip başka bir varlık olabilir. ve aralarında bir ilişki olacak.



DBMS Veritabanı Modelleri

İlişkisel Model

Bu modelde veriler iki boyutlu tablolar halinde düzenlenir ve ortak bir alan saklanarak ilişki sürdürülür.

Bu model 1970 yılında EF Codd tarafından tanıtıldı ve o zamandan beri en yaygın kullanılan veritabanı modeli oldu, infact, dünya çapında kullanılan tek veritabanı modeli diyebiliriz.


İlişkisel modeldeki verilerin temel yapısı tablolardır. Belirli bir türle ilgili tüm bilgiler, o tablonun satırlarında saklanır.

DBMS Veritabanı Modelleri

İlişkisel Model

student_id	name	age
1	Akon	17
2	Bkon	18
3	Ckon	17
4	Dkon	18

subject_id	name	teacher
1	Java	Mr. J
2	C++	Miss C
3	C#	Mr. C Hash
4	Php	Mr. P H P



student_id	subject_id	marks
1	1	98
1	2	78
2	1	76
3	2	88

Veritabanı Şeması

Bir veritabanı şeması, tüm veritabanının mantıksal görünümünü temsil eden iskelet yapısıdır. Verilerin nasıl organize edildiğini ve aralarındaki ilişkilerin nasıl ilişkilendirildiğini tanımlar. Verilere uygulanacak tüm kısıtlamaları formüle eder.

Bir veritabanı şeması, varlıklarını ve aralarındaki ilişkiyi tanımlar. Şema diyagramları ile gösterilebilen veritabanının açıklayıcı bir detayını içerir. Programcıların veritabanını anlamalarına ve kullanışlı hale getirmelerine yardımcı olmak için şemayı tasarlayanlar veritabanı tasarımcılarıdır.

İlişkisel modelde, bir ilişkinin şeması, adını, her alanın (veya nitelik veya sütunun) adını ve her alanın türünü belirtir. Örnek olarak, bir üniversite veri tabanındaki öğrenci bilgileri aşağıdaki şema ile ilişkili olarak saklanabilir:



Veritabanı Şeması

Students(*sid*: string, *name*: string, *login*: string, *age*: integer, *gpa*: real)

<i>sid</i>	<i>name</i>	<i>login</i>	<i>age</i>	<i>gpa</i>
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Data Abstraction

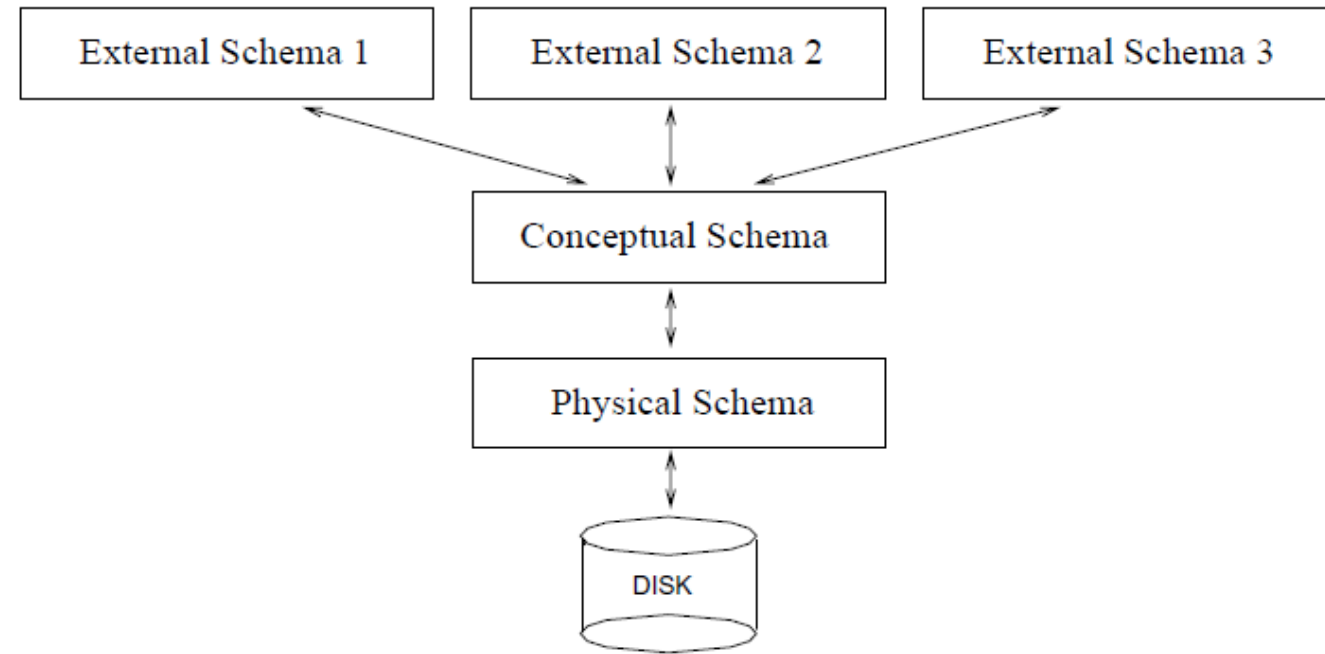
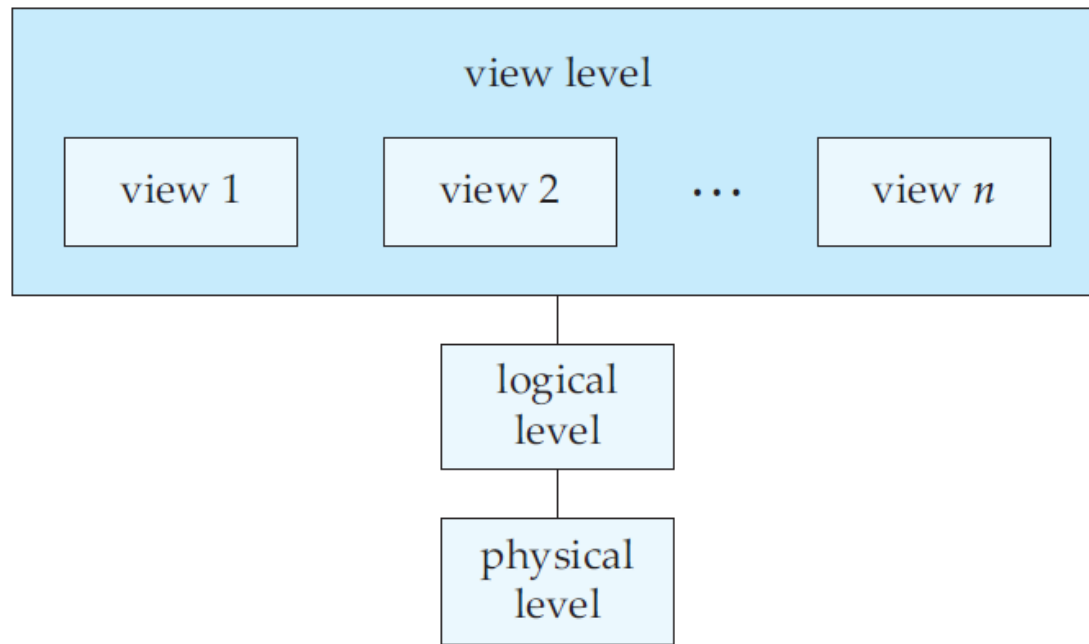
Fiziksel seviye. En düşük soyutlama seviyesi, verilerin gerçekte nasıl saklandığını açıklar. Fiziksel seviye, karmaşık düşük seviyeli veri yapılarını ayrıntılı olarak tanımlar.

- **Mantıksal seviye (Logical/Conseptional).** Bir sonraki daha yüksek soyutlama düzeyi, veritabanında hangi verilerin depolandığını ve bu veriler arasında hangi ilişkilerin bulunduğunu açıklar. Bu nedenle mantıksal düzey, tüm veritabanını az sayıda nispeten basit yapılar açısından tanımlar. Basit yapıların mantıksal düzeyde uygulanması karmaşık fiziksel düzeydeki yapıları içerebilse de, mantıksal düzeydeki kullanıcının bu karmaşıklığın farkında olması gerekmez. Buna fiziksel veri bağımsızlığı denir. Veritabanında hangi bilgilerin tutulacağına karar vermesi gereken veritabanı yöneticileri, mantıksal soyutlama düzeyini kullanır.

Data Abstraction

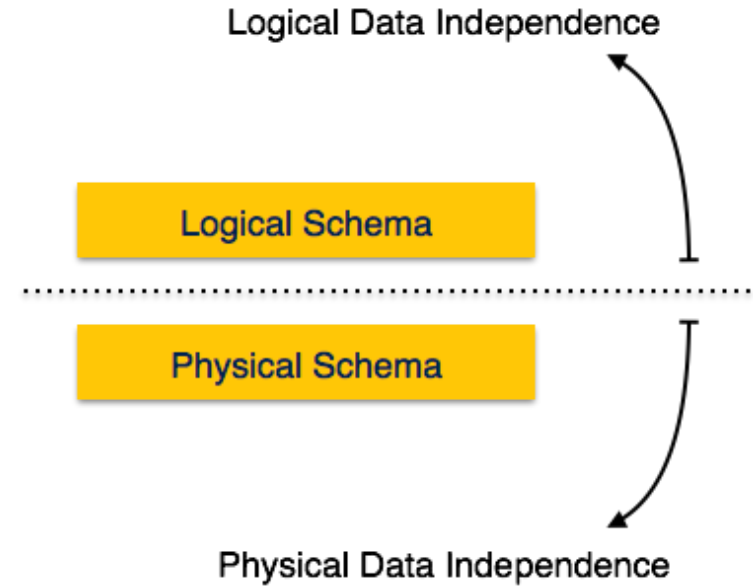
Görüntüleme seviyesi (View/External). En yüksek soyutlama düzeyi, tüm veritabanının yalnızca bir bölümünü tanımlar. Mantıksal düzeyde daha basit yapılar kullanılsa da, büyük bir veritabanında depolanan bilgilerin çeşitliliği nedeniyle karmaşıklık devam eder. Veritabanı sisteminin birçok kullanıcısı tüm bu bilgilere ihtiyaç duymaz; bunun yerine, veritabanının yalnızca bir kısmına erişimleri gerekir. Soyutlamanın görünüm seviyesi, sistemle etkileşimlerini basitleştirmek için mevcuttur. Sistem aynı veritabanı için birçok görünüm sağlayabilir.

Data Abstraction



Veri Bağımsızlığı

Bir veritabanı sistemi normalde kullanıcıların verilerine ek olarak çok fazla veri içerir. Örneğin, verileri kolayca bulmak ve almak için meta veri olarak bilinen verilerle ilgili verileri depolar. Veritabanında depolandıktan sonra bir dizi meta veriyi değiştirmek veya güncellemek oldukça zordur. Ancak bir DBMS genişledikçe, kullanıcıların gereksinimlerini karşılamak için zamanla değişmesi gerekir. Tüm veriler bağımlıysa, sıkıcı ve oldukça karmaşık bir iş haline gelir.

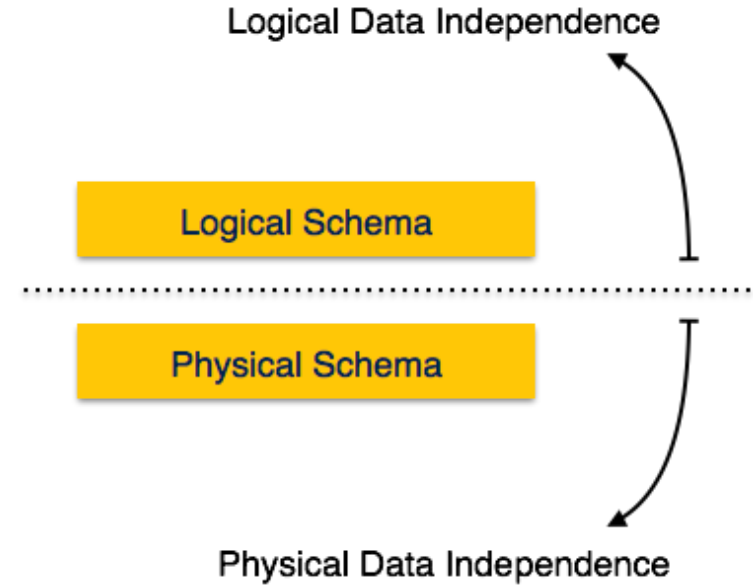


Veri Bağımsızlığı

Mantıksal Veri Bağımsızlığı

Mantıksal veriler, veritabanıyla ilgili verilerdir, yani verilerin içinde nasıl yönetildiği hakkında bilgi depolar. Örneğin, veritabanında depolanan bir tablo (ilişki) ve bu ilişkiye uygulanan tüm kısıtlamaları.

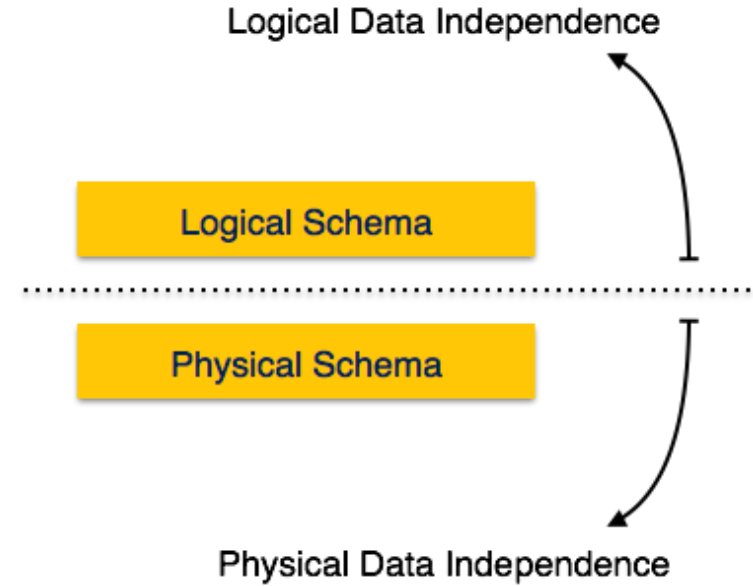
Mantıksal veri bağımsızlığı, kendisini diskte depolanan gerçek verilerden özgürleştiren bir tür mekanizmadır. Tablo formatında bazı değişiklikler yaparsak, diskte bulunan verileri değiştirmemelidir.



Veri Bağımsızlığı

Mantıksal Veri Bağımsızlığı

- Mantıksal veri bağımsızlığı, dış şemayı değiştirmek zorunda kalmadan kavramsal şemayı değiştirebilme özelliğini ifade eder.
- Mantıksal veri bağımsızlığı, dış düzeyi kavramsal görünümünden ayırmak için kullanılır.
- Verilerin kavramsal görünümünde herhangi bir değişiklik yaparsak, verilerin kullanıcı görünümü etkilenmeyecektir.
- Mantıksal veri bağımsızlığı, kullanıcı arabirimi düzeyinde gerçekleşir.

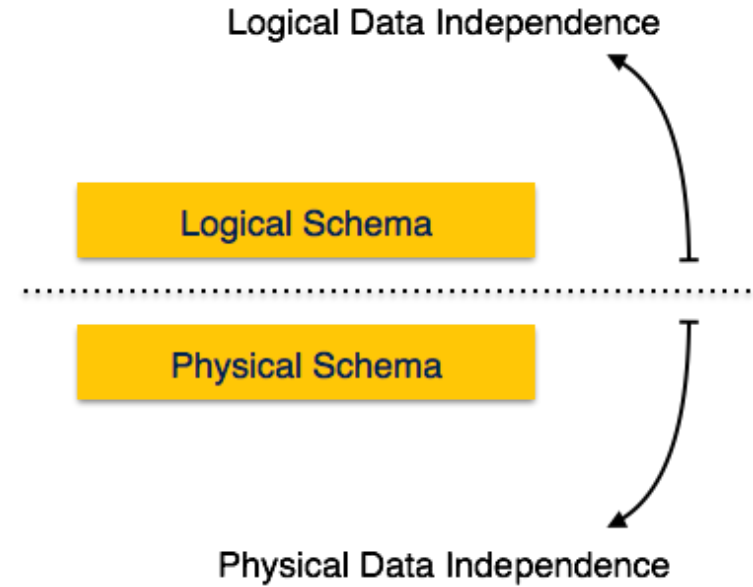


Veri Bağımsızlığı

Fiziksel Veri Bağımsızlığı

Tüm şemalar mantıksaldır ve gerçek veriler diskte bit formatında saklanır. Fiziksel veri bağımsızlığı, şemayı veya mantıksal verileri etkilemeden fiziksel verileri değiştirme gücüdür.

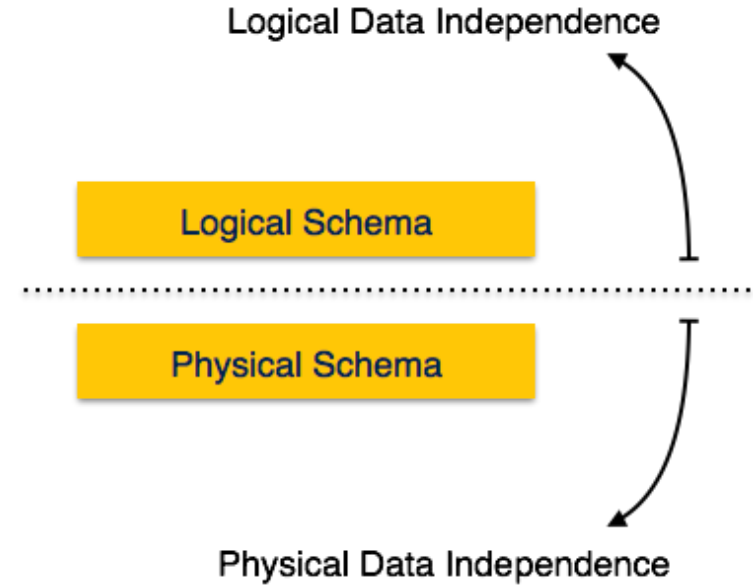
Örneğin, depolama sisteminin kendisini değiştirmek veya yükseltmek istememiz durumunda - sabit diskleri SSD ile değiştirmek istediğimizi varsayalım - mantıksal veriler veya şemalar üzerinde herhangi bir etkisi olmamalıdır.



Veri Bağımsızlığı

Fiziksel Veri Bağımsızlığı

- Fiziksel veri bağımsızlığı, kavramsal şemayı değiştirmek zorunda kalmadan dahili şemayı değiştirme kapasitesi olarak tanımlanabilir.
- Veritabanı sistem sunucusunun depolama boyutunda herhangi bir değişiklik yaparsak, veritabanının Kavramsal yapısı etkilenmeyecektir.
- Fiziksel veri bağımsızlığı, kavramsal düzeyleri iç düzeylerden ayırmak için kullanılır.
- Fiziksel veri bağımsızlığı mantıksal arayüz seviyesinde gerçekleşir.



DBMS Sorgular

Bir veri tabanından bilginin elde edilme kolaylığı, çoğu zaman bir kullanıcı için değerini belirler.

Bir DBMS, sorguların oluşturulabileceği, sorgu dili adı verilen özel bir dil sağlar.

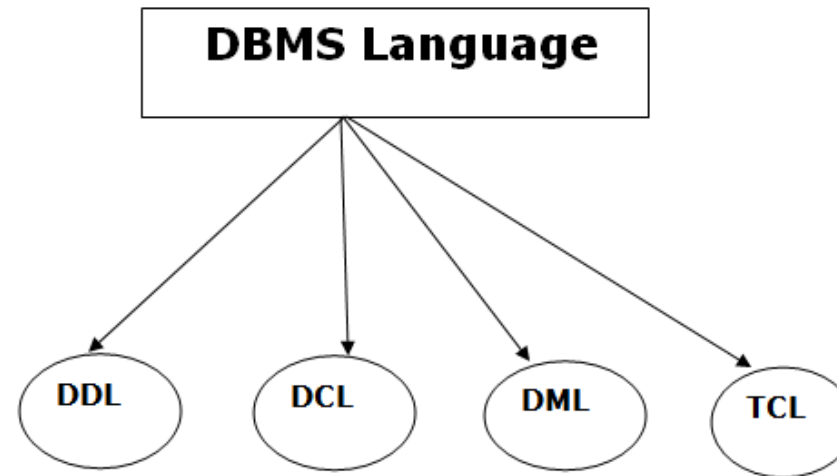
İlişkisel modelin çok çekici bir özelliği, güçlü sorgu dillerini desteklemesidir.

İlişkisel analiz, matematiksel mantığa dayalı resmi bir sorgulama dilidir ve bu dildeki sorguların sezgisel, kesin bir anlamı vardır.

İlişkisel cebir, hesabın gücüne eşdeğer olan ilişkileri işlemek için bir operatörler koleksiyonuna dayanan başka bir resmi sorgulama dilidir.

Veritabanı Dili

- Bir DBMS, veritabanı sorgularını ve güncellemelerini ifade etmek için uygun dillere ve arayüzlere sahiptir.
- Veri tabanı dilleri, veri tabanındaki verileri okumak, saklamak ve güncellemek için kullanılabilir.



Veritabanı Dili

1. Veri Tanımlama Dili (DDL)

- DDL , D ata D efinition L anguage anlamına gelir . Veritabanı yapısını veya modelini tanımlamak için kullanılır.
- Harici ve kavramsal şemaları tanımlamak için bir "veri tanımlama dili" (DDL) kullanılır.
- Veritabanında şema, tablolar, indeksler, kısıtlamalar vb. Oluşturmak için kullanılır.
- DDL ifadelerini kullanarak veritabanının iskeletini oluşturabilirsiniz.
- Veri tanımlama dili, tablo ve şema sayısı, adları, indeksleri, her tablodaki sütunlar, kısıtlamalar vb. Gibi meta verilerin bilgilerini depolamak için kullanılır.

Veritabanı Dili

1. Veri Tanımlama Dili (DDL)

- create: Veritabanında nesne oluşturmak için kullanılır.
- alter: Veritabanının yapısını değiştirmek için kullanılır.
- drop: Veritabanından **nesnelerin** silinmesi için kullanılır.
- delete: Bir tablodaki **kayıtları** kaldırmak için kullanılır.
- Yeniden adlandır: Bir nesneyi yeniden adlandırmak için kullanılır.
- Yorum: Veri sözlüğüne yorum yapmak için kullanılır.

Veritabanı Dili

2. Veri İşleme Dili (DML)

DML , (Data Manipülasyon Language). Bir veritabanındaki verilere erişmek ve bunları işlemek için kullanılır. Kullanıcı isteklerini ele alır.

DML kapsamında gelen bazı görevler şunlardır:

- Seç: Bir veri tabanından veri almak için kullanılır.
- Ekle: Bir tabloya veri eklemek için kullanılır.
- Güncelleme: Bir tablo içindeki mevcut verileri güncellemek için kullanılır.
- Sil: Bir tablodaki tüm kayıtları silmek için kullanılır.
- Birleştirme: UPSERT işlemini, yani ekleme veya güncelleme işlemlerini gerçekleştirir.
- Çağrı: Yapılandırılmış bir sorgu dilini veya bir Java alt programını çağırmak için kullanılır.
- Planı Açıkla: Verileri açıklama parametresine sahiptir.
- Lock Table: Eşzamanlılığı kontrol eder.

Veritabanı Dili

3. Veri Kontrol Dili (DCL)

- DCL D ata C ontrol L anguage anlamına gelir . Saklanan veya kaydedilen verileri almak için kullanılır.
- DCL yürütmesi işlemseldir. Ayrıca geri alma parametrelerine de sahiptir.
- (Ancak Oracle veritabanında, veri kontrol dilinin yürütülmesi geri alma özelliğine sahip değildir.)
- DCL kapsamında gelen bazı görevler şunlardır:
- Grant: Kullanıcıya bir veritabanına erişim yetkileri vermek için kullanılır.
- İptal Et: Kullanıcıdan izinleri geri almak için kullanılır.

Veritabanı Dili

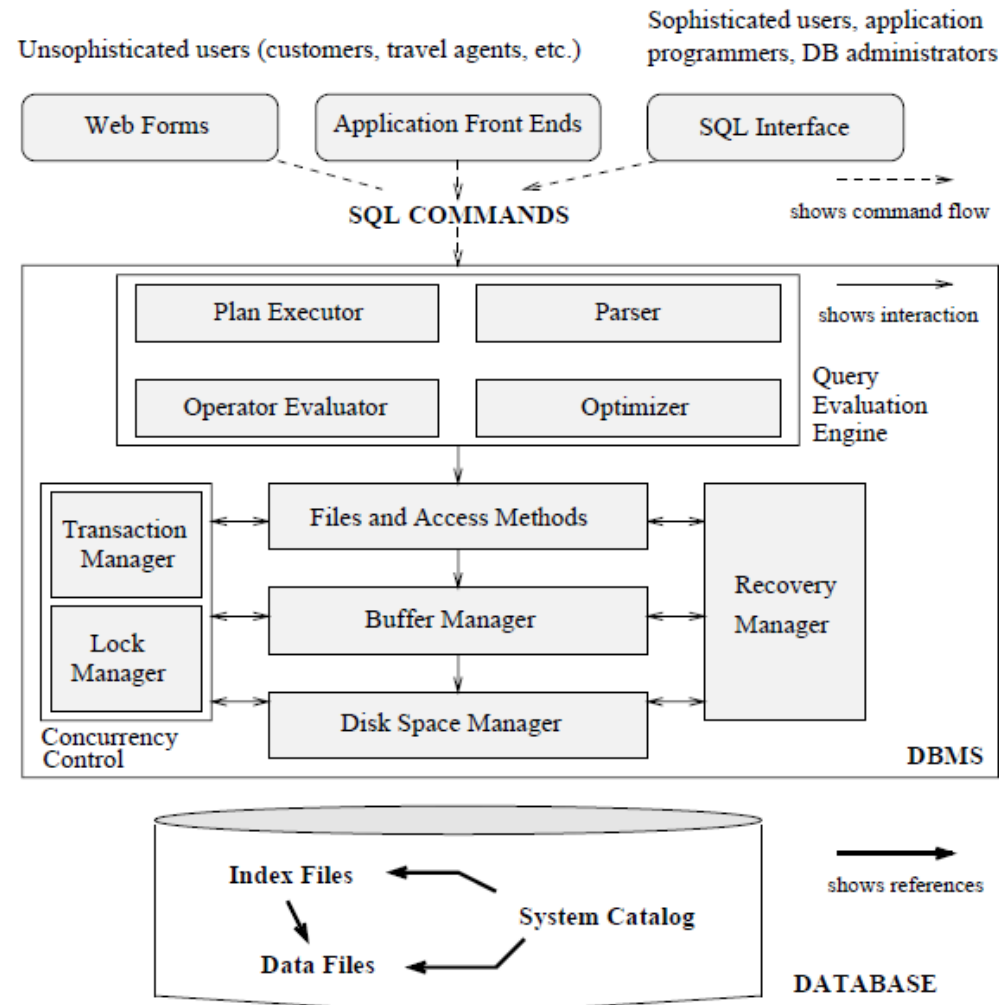
4. İşlem Kontrol Dili (TCL)

- TCL, DML ifadesi tarafından yapılan değişiklikleri çalıştırmak için kullanılır. TCL, mantıksal bir işlem olarak gruplandırılabilir.
- TCL kapsamına giren bazı görevler şunlardır:
- Commit: İşlemin veri tabanına kaydedilmesi için kullanılır.
- Geri Alma: Veritabanını son Commit'ten itibaren orijinaline geri yüklemek için kullanılır.

TRANSACTION MANAGEMENT

İşlem, bir veritabanı uygulamasında tek bir mantıksal işlevi gerçekleştiren bir işlemler topluluğudur. Her işlem, hem atomite hem de tutarlılık birimidir. Bu nedenle, işlemlerin herhangi bir veritabanı tutarlılık kısıtlamasını ihlal etmemesini şart koşuyoruz. Diğer bir deyişle, bir işlem başlatıldığında veritabanı tutarlıysa, işlem başarıyla sona erdiğinde veritabanı tutarlı olmalıdır. Ancak, bir işlemin yürütülmesi sırasında, A'nın borcunun veya B'nin kredisinin diğerinden önce yapılması gerektiğinden, geçici olarak tutarsızlığa izin vermek gerekebilir. Bu geçici tutarsızlık, gerekli olmakla birlikte, bir arıza meydana geldiğinde zorluklara yol açabilmektedir.

VTYS'NİN YAPISI



VTYS'NİN YAPISI

İşlem, bir veritabanı uygulamasında tek bir mantıksal işlevi gerçekleştiren bir işlemler topluluğudur. Her işlem, hem atomite hem de tutarlılık birimidir. Bu nedenle, işlemlerin herhangi bir veritabanı tutarlılık kısıtlamasını ihlal etmemesini şart koşuyoruz. Diğer bir deyişle, bir işlem başlatıldığında veritabanı tutarlıysa, işlem başarıyla sona erdiğinde veritabanı tutarlı olmalıdır. Ancak, bir işlemin yürütülmesi sırasında, A'nın borcunun veya B'nin kredisinin diğerinden önce yapılması gerektiğinden, geçici olarak tutarsızlığa izin vermek gerekebilir. Bu geçici tutarsızlık, gerekli olmakla birlikte, bir arıza meydana geldiğinde zorluklara yol açabilmektedir.

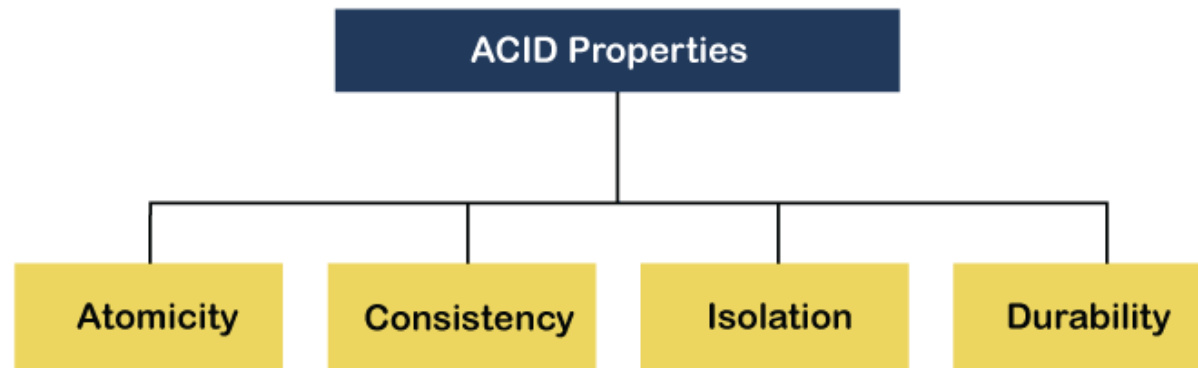
DBMS'de ACID Özellikleri

DBMS, üzerinde herhangi bir değişiklik yapıldığında entegre kalması gereken verilerin yönetimidir. Bunun nedeni, verilerin bütünlüğünün etkilenmesi durumunda tüm verilerin bozulacak ve bozulacak olmasıdır. Bu nedenle, verilerin bütünlüğünü korumak için, veritabanı yönetim sisteminde ACID özellikleri olarak bilinen dört özellik vardır . ACID özellikleri, farklı bir görev grubundan geçen işlem içindir ve burada ACID özelliklerinin rolünü görmeye geldik.

Bu bölümde, ACID özelliklerini öğrenip anlayacağız. Bu özelliklerin ne anlama geldiğini ve her bir özelliğin ne için kullanıldığını öğreneceğiz. ACID özelliklerini de bazı örnekler yardımıyla anlayacağız.

ACID Özellikleri

ACID teriminin genişlemesi şunları tanımlar:



ACID Özellikleri

ACID teriminin genişlemesi şunları tanımlar:

1) Atomiklik: Atomiklik terimi, verilerin atomik kaldığını ifade eder. Bu, veriler üzerinde herhangi bir işlem yapılıyorsa, ya gerçekleştirilmesi ya da tamamen yürütülmesi ya da hiç yürütülmemesi gerektiği anlamına gelir. Ayrıca, işlemin arada kesintiye uğramaması veya kısmen yürütülmemesi gerektiği anlamına gelir. İşlem üzerinde işlem yapılması durumunda, işlem kısmen değil, tamamen yürütülmelidir.

ACID Özellikleri

ACID teriminin genişlemesi şunları tanımlar:

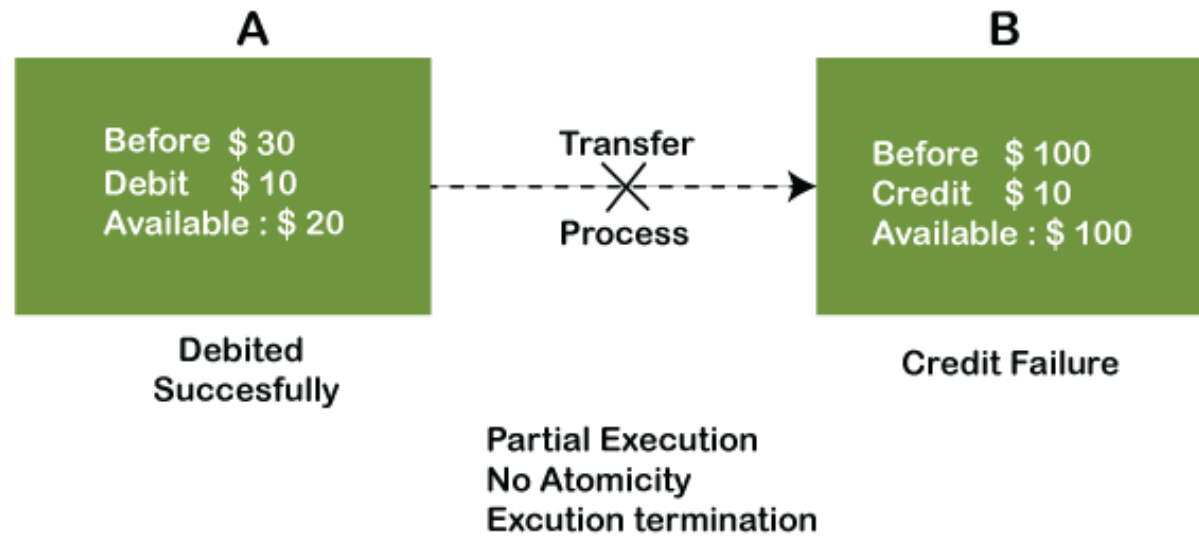
1) Atomiklik:

Örnek: Eğer Remo, Sheero'nun B hesabına 10 \$ göndermek istediği hesabında 30 \$ olan bir A hesabına sahipse, B hesabında zaten 100 \$ var. B hesabına 10 ABD doları transfer edildiğinde, toplam 110 ABD doları olacaktır. Şimdi gerçekleşecek iki operasyon olacak. Bunlardan biri, Remo'nun transfer etmek istediği 10 \$ 'lık tutar A hesabından borçlandırılacak ve aynı miktar B hesabına, yani Sheero'nun hesabına yatırılacak. Şimdi, ne olur - ilk borçlandırma işlemi başarılı bir şekilde yürütülür, ancak kredi işlemi başarısız olur. Böylece, Remo'nun A hesabında değer 20 dolar olur ve Sheero'nun hesabına göre daha önce olduğu gibi 100 dolar kalır.

ACID Özellikleri

ACID teriminin genişlemesi şunları tanımlar:

1) Atomiklik:

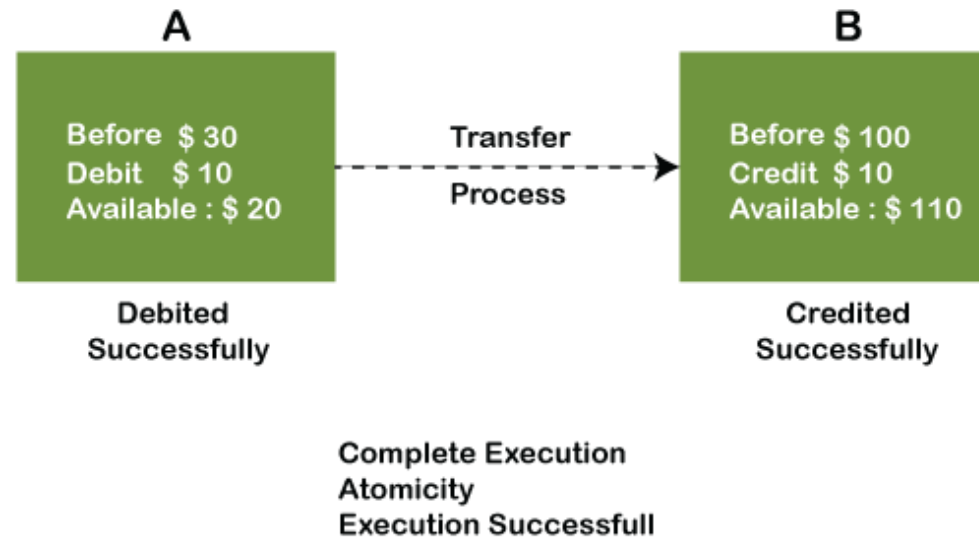


ACID Özellikleri

1) Atomiklik:

Yukarıdaki diyagramda, 10 \$ kredilendirdikten sonra, miktarın B hesabında hala 100 \$ olduğu görülebilir. Yani bu atomik bir işlem değil.

Aşağıdaki resim hem borç hem de kredi işlemlerinin başarılı bir şekilde yapıldığını göstermektedir. Böylece işlem atomiktir.



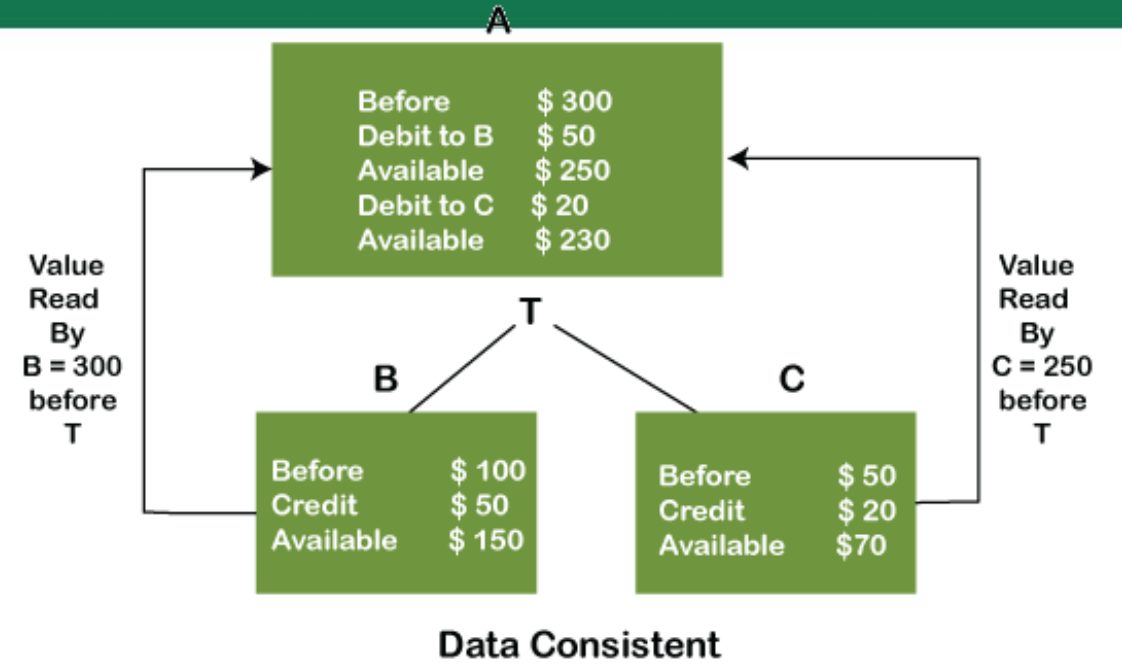
ACID Özellikleri

2) Tutarlılık: kelime tutarlılık değeri her zaman muhafaza edilmiş halde kalır gerektiği anlamına gelir. Gelen DBMS veri bütünlüğü veritabanındaki bir değişiklik yapılması halinde, her zaman korunmuş kalması gereken araçlar, muhafaza edilmelidir. İşlemler söz konusu olduğunda, veri tabanının işlemde önce ve sonra tutarlı kalması için verilerin bütünlüğü çok önemlidir. Veriler her zaman doğru olmalıdır.

ACID Özellikleri

2) Tutarlılık:

A, B ve C olmak üzere üç hesap vardır; burada A, her iki B & C'ye birer birer T işlemi yapmaktadır. Gerçekleşen iki işlem vardır, yani Borç ve Alacak. A Hesabı öncelikle B hesabına 50 \$ borçlandırır ve A hesabındaki tutar işlemden önce B tarafından 300 \$ okunur.



Başarılı T işleminden sonra, B'deki mevcut miktar 150 \$ olur. Şimdi, A, C hesabına 20 \$ borçludur ve o zaman, C tarafından okunan değer 250 \$ 'dır (bu, B'ye 50 \$'lık bir borç başarıyla gerçekleştirildiği için doğrudur). A hesabından C'ye borç ve alacak işlemi başarıyla gerçekleştirildi. İşlemin başarılı bir şekilde yapıldığını ve değerlerin de doğru okunduğunu görebiliyoruz. Böylece veriler tutarlıdır. B ve C tarafından okunan değerlerin 300 \$ olması durumunda, bu, verilerin tutarsız olduğu anlamına gelir, çünkü borçlandırma işlemi yürütüldüğünde,

ACID Özellikleri

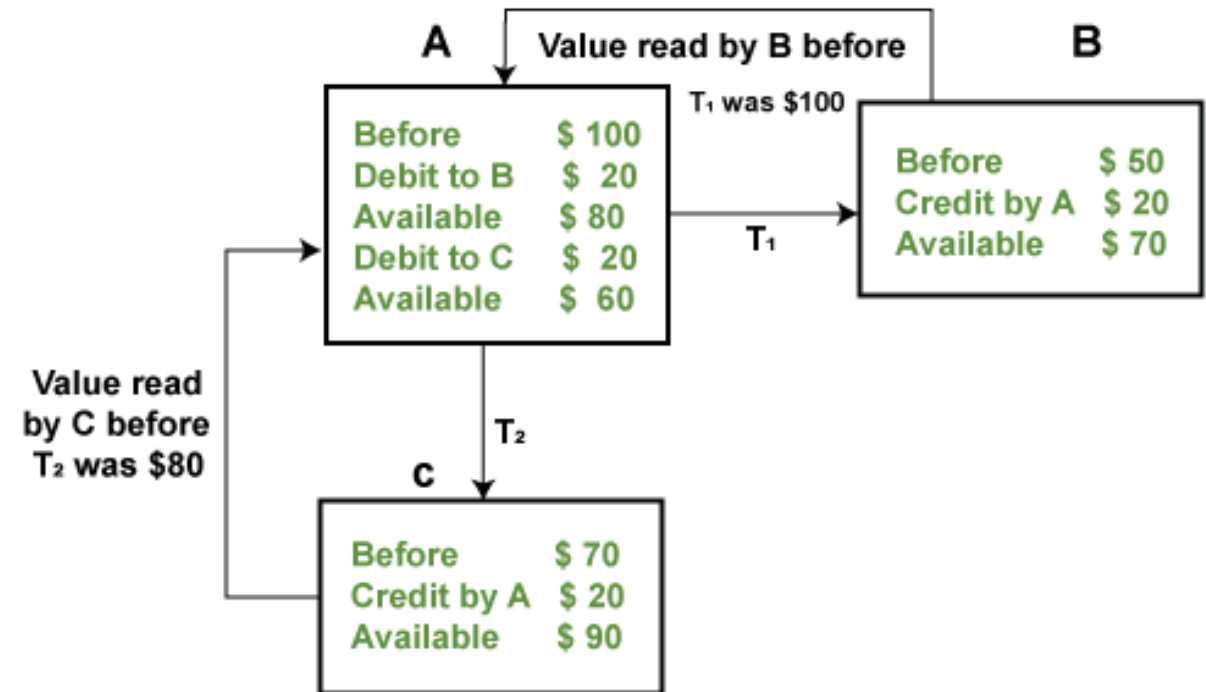
3) 'İzolasyon:

'İzolasyon' terimi, ayırma anlamına gelir. DBMS'de İzolasyon, hiçbir verinin diğerini etkilememesi gereken ve eşzamanlı olarak meydana gelebilecek bir veritabanının özelliğidir. Kısaca ilk veritabanı üzerindeki işlem tamamlandığında bir veritabanı üzerindeki işlem başlamalıdır. Bu, iki farklı veri tabanında iki işlem yapılıyorsa, birbirlerinin değerini etkilemeyebilecekleri anlamına gelir. İşlemler söz konusu olduğunda, iki veya daha fazla işlem aynı anda gerçekleştiğinde, tutarlılık korunmalıdır. Herhangi bir işlemde meydana gelen herhangi bir değişiklik, değişiklik hafızaya kaydedilmediği sürece diğer işlemler tarafından görülmeyecektir.

ACID Özellikleri

3) 'İzolasyon:

Örnek: İki işlem aynı anda iki farklı hesapta çalışıyorsa, her iki hesabın değeri etkilenmemelidir. Değer kalıcı kalmalıdır. Aşağıdaki diyagramda görebileceğiniz gibi, A hesabı B ve C hesabına T1 ve T2 işlemleri yapıyor, ancak her ikisi de birbirini etkilemeden bağımsız olarak yürütüyor. İzolasyon olarak bilinir.



Isolation - Independent execution of T₁ & T₂ by A

ACID Özellikleri

4) Dayanıklılık:

Dayanıklılık, bir şeyin kalıcılığını sağlar. DBMS'de dayanıklılık terimi, işlemin başarılı bir şekilde yürütülmesinden sonra verilerin veritabanında kalıcı olmasını sağlar. Verilerin dayanıklılığı o kadar mükemmel olmalıdır ki, sistem arızalansa veya bir çökmeye yol açsa bile, veritabanı hala hayatta kalır. Bununla birlikte, kaybolursa, veritabanının dayanıklılığını sağlamak kurtarma yöneticisinin sorumluluğunda olur. Değerleri işlemek için, her değişiklik yaptığımızda COMMIT komutu kullanılmalıdır.

Bu nedenle, DBMS'nin ACID özelliği, veritabanındaki verilerin tutarlılığını ve kullanılabilirliğini sürdürmede hayati bir rol oynar.