# 1. Imperative Programming ve Declarative Programming kavramlarını kısaca açıklayıp farklarını belirtiniz

Imperative Türkçede emir, buyruk, zorunluluk anlamına geliyor. Declarative ise bildiren, açıklayan anlamındadır. Yabancılar ise imperative işlemi nasıl yapacağını anlattığın, declarative ise ne yapacağını anlattığın programlama şekli olarak tanımlıyorlar.

İmperative yöntemde açıklayıcı emirlerle işlemi detaylı bir şekilde gerçekleştiririz.

**// Diktörgen Çizme**  
kalemiSuPozisyonaGötür()  
kalemiBastır()  
kalemiSuPozisyonaSürükle()  
kalemiSuPosizyondaDurdur()  
kalemiSuPozisyonaSürükle()  
kalemiSuPosizyondaDurdur()  
kalemiSuPozisyonaSürükle()  
kalemiSuPozisyonaDurdur()  
kalemiSuPozisyonaSürükle()  
kalemiSuPozisyonaDurdur()  
kalemiKaldır()

Declarative yöntemde sadece yapacağınız şeyi anlatırsınız. diktorgen(ciz)

# 2. Veri tabanlarının sorgu optimizasyonlarında index oluşturmanın avantajı nedir ? Sık index kullanmak bir probleme yol açar mı?

Index, veri tabanı tabloları üzerinde tanımlanan ve veriye daha az işlemle daha hızlı ulaşan veri tabanı nesneleridir. Indexler hakkında klasik bir örnek olarak telefon rehberi verilebilir. Telefon rehberindeki kayıtların sıralı olmaması durumunda, yani her kaydın telefon defterinde rastgele tutulması durumunda, arayacağımız bir isim için tüm rehberi gezmemiz gerekecek. Ama rehberinizdeki kayıtlar sıralı olsaydı, aradığımız ismin rehberin ortasındaki isimden ileride mi yoksa geride mi olduğuna bakabilirdik. Bu şekilde aradığımız verileri eleyerek bir kaç adımda istediğimiz sonuca ulaşabilirdik. Bu örnekteki gibi verinin sıralı tutulmasını sağlayan nesnelere index denir.

# 3. İlişkisel veritabanları için normalizasyon kavramı neyi ifade etmektedir ? İlk 3 normal formu örnek üzerinden açıklayınız

Normalizasyon; veri tabanı tasarım aşamasında veri tekrarını, veri kaybını veya veri yetersizliğini önlemek için gerçekleştirilen işlemlerdir.

Normalizayon uygulanan veri tabanlarının performansı artar, sabit diskteki boyutu azalır ve tablolarda ki satır ve sütun sayısı azalacağından veri tekrarı önlenmiş olur. Özellikle silme, güncelleme gibi işlemler de çıkabilecek sorunlar büyük oranda azaltılmış olur.

**1NF(1. Normal Form)**

Bir veri tabanının 1NF olabilmesi için aşağıdaki özellikleri karşılayabilmesi gerekir:

Aynı tablo içinde tekrarlayan kolonlar bulunamaz.

Her kolonda yalnızca bir değer bulunabilir .

Her satır bir eşsiz anahtarla tanımlanmalıdır



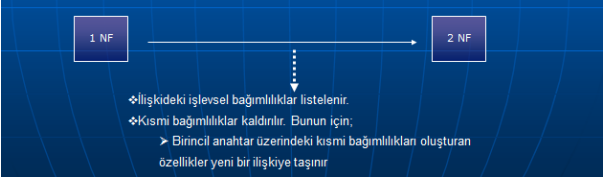
**2NF(2.Normal Form)**

Bir veri tabanının 2NF olabilmesi için aşağıdaki özellikleri karşılayabilmesi gerekir:

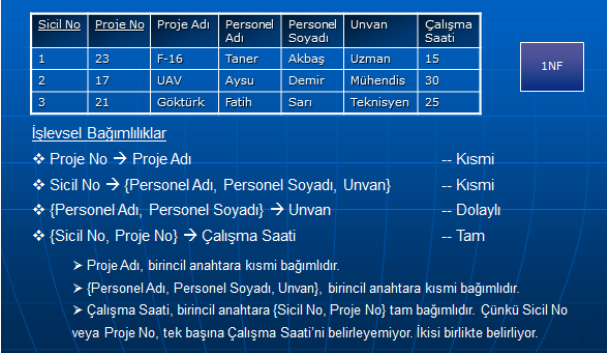
Tablo 1NF olmalıdır.

Anahtar olmayan değerler ile kompozit (bileşik) anahtarlar arasında kısmi (partial) bağımlılık durumu oluşmamalıdır. Kısmi bağımlılık durumu, anahtar olmayan herhangi bir değer kompozit bir anahtarın yalnızca bir kısmına bağıl ise oluşur. Herhangi bir veri alt kümesi birden çok satırda tekrarlanmamalıdır. Bu tür veri alt kümeleri için yeni tablolar oluşturulmalıdır.

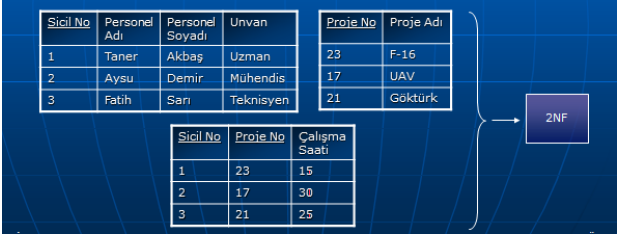
Ana tablo ile yeni tablolar arasında, dış anahtarlar (foreign key) kullanılarak ilişkiler tanımlanmalıdır.



Aşağıda 1NF halindeki tablo bulunmaktadır.



Bu 1 NF halindeki tablonun 2NF uygulanmış hali aşağıdaki gibidir.



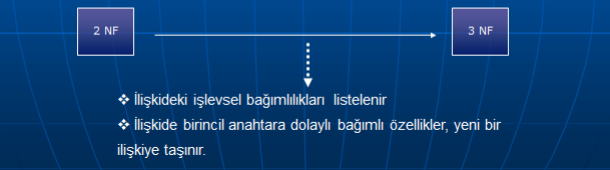
3NF(3.Normal Form)

Bir veri tabanının 3NF olabilmesi için aşağıdaki özellikleri karşılayabilmesi gerekir:

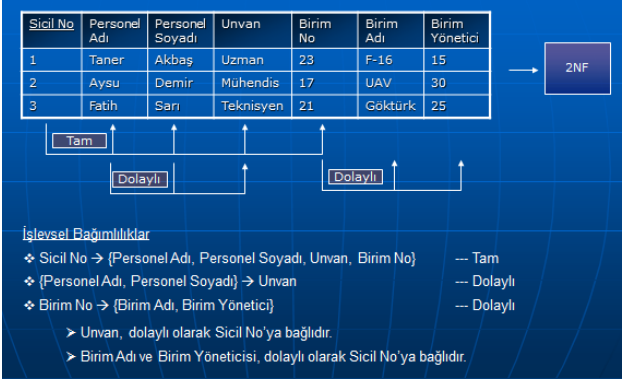
Veri tabanı 2NF olmalıdır,

Anahtar olmayan hiç bir kolon bir diğerine (anahtar olmayan başka bir kolona) bağıl olmamalı ya da geçişken fonksiyonel bir bağımlılığı (transitional functional dependency) olmamalıdır. Başka bir deyişle her kolon eşsiz anahtara tam bağımlı olmak zorundadır.

Veri tabanımızı 3NF şartlarına uydurabilmek için anahtar olmayan ve eşsiz anahtara tam bağımlı olmayan tüm kolonları kaldırmalıyız.



Aşağıda 2NF halindeki tablo bulunmaktadır.



Bu 2NF halindeki tablonun 3NF uygulanmış hali aşağıdaki gibidir



# 4. ORM kütüphaneleri kullanmak her zaman avantajlı mıdır ? ORM kütüphanelerinin ne gibi dezavantajları olabilir ?

**ORM kullanımı çok zaman kazandırır çünkü:**

DRY : Veri modelinizi yalnızca bir yere yazarsınız ve kodu güncellemek, sürdürmek ve yeniden kullanmak kolaydır.

Veritabanı işlemeden I18N 'ye kadar pek çok işlem otomatik olarak yapılır.

Sizi yazmaya zorlar MVC kod, sonunda kodunuzu biraz temizler.

Kötü biçimlendirilmiş bir SQL yazmak zorunda değilsiniz

Hijyenik; Hazırlanan ifadeleri veya işlemleri kullanmak, bir yöntemi çağırmak kadar kolaydır.

**ORM kitaplığı kullanmak daha esnektir, çünkü:**

Doğal kodlama biçiminize uyuyor

DB sistemini soyutlar, böylece istediğiniz zaman değiştirebilirsiniz.

Model uygulamanın geri kalan kısmına zayıf şekilde bağlı olduğundan, onu değiştirebilir veya başka bir yerde kullanabilirsiniz.

Baş ağrısız veri kalıtımı gibi OOP iyiliğini kullanmanıza izin verir.

**Dezavantajı:**

Bunu öğrenmek zorundasınız ve ORM kütüphaneleri hafif araçlar değildir;

Ayarlamak gerek.

Performans her zamanki sorgular için uygundur, ancak bir SQL master büyük projeler için kendi SQL'i ile her zaman daha iyisini yapar.

DB'yi soyutlar. Sahnenin arkasında neler olup bittiğini biliyorsanız, tamamıyla, for döngüsünde ağır bir hit gibi çok açgözlü ifadeler yazabilen yeni programcılar için bir tuzak.

# 5. Domain Specific Language (DSL) kavramını açıklayınız.

Programlama dilleri, özel(Domain Specific Language) ve genel (General Purpose Language) kapsamda kullanılan diller olarak ikiye ayrılabilir. DSL (Domain Specific Language), özel bir uygulama alanı için kullanılan dildir. DSL Örneği olarak web uygulaması geliştireceksiniz ve uygulamanın özel bir kapsamı olan tasarım konusunda CSS'i verebiliriz.

Örneğin java ile uygulama geliştiriyorsunuz ve veritabanı işlerini SQL ile yaparken, build ve deploy işlemleri için ANT kullanıyorsunuz.

# 6. Long lived transaction kavramı hangi tip transactionları ifade etmektedir ? Dezavantajları var mıdır ? Varsa nelerdir ?

Choreographed Sagas tip trans ıfade edıyor. Güzellikleri olduğu kadar bazı dezavantajlarıda var mesela parça parça hale getirdiğiniz iş mantığınızı yönetmek daha zor olabilir çünkü orchestration yok eventler havada uçuşuyor, tümüyle sistemin nasıl çalıştığını anlamak zorlaşacak bu da maintain etmenizi zorlaştıracaktır. Bu yüzden küçük boyutlu sagalar ve synchronous gerekmeyen uygulamalar için tercih etmek mantıklı olabilir.

# 7. Thread Pool nedir ? Nerelerde kullanılır ? Thread Pool’un amacı sistemdeki Thread sayısını kontrol altında tutmak ve yaratılan belirli sayıdaki Thread’leri işi bitince sonlandırmak yerine gelecekteki olası işler için bekletmektir. Bir Thread Pool yaratıldığında belirli sayıda Thread’i yaratır ve yapılacak işler gelene kadar bu Thread’leri bekletir. Yapılacak işler sıraya girdikçe boşta bulunan Thread’lerden birisi uyanıp işi yapar ve tekrar bekleme durumuna geçer.

# 8. Scalability nedir ? Horizontal ve Vertical Scalability kavramlarını açıklayınız.

Performans limitlerinin ötesine geçmek için mevcut yapıya daha fazla kaynak eklenmesine “Scaling” (Ölçekleme) denilmektedir. Facebook gibi bazı firmalarda ise bu iş “Capacity Planning” olarak isimlendirilmektedir

**Vertical Scaling (Single Node üzerinde kaynak arttırımı):**

Mecutta var olan bir sunucuya daha fazla kaynak aktarılması ile gerçekleştirilmektedir. Bu yaklaşım genellikle sanallaştırılmış ortamlarda daha yaygın olarak kullanılabilmektedir. Sanallaştırma çözümleri üzerinde (Vmware, Hyper-V vb.) CPU, Memory gibi kaynakların arttırılması ile mümkün olmaktadır. En büyük avantajı, mevcut uygulama mimarisi üzerinde değişiklik gerektirmemesi iken, var olan baz donanım her zaman kısıtlayıcı bir unsur olmaktadır.

**Horizontal Scaling (Node sayısının arttırımı):**

Kullanıcılardan gelen isteklerin, farklı uygulama sunucularına yönlendirilmesinin var olan sunucuların kapasitesini arttırmaktan daha kolay olduğu durumlarda tercih edilmektedir. Özellikle cloud ortamlarda kolaylıkla gerçekleştirilebilmektedir. Ayrıca vertical scaling ile erişilebilecek maksimum bir kapasite olduğu için, bir noktadan sonra horizontal scaling yapılması zorunlu hale gelmektedir. Ayrıca, düşük maliyetli sunucular ile kapasitenin arttırımı ve availability açısından da avantaj sağlamaktadır.

Ancak bu tarz bir yaklaşım için, uygulamanın uygun bir mimari ile geliştirilmiş olması gerekmektedir.

# 9. Data replication ve sharding nedir ? Aralarında nasıl bir fark bulunmaktadır ?

Veri Replikasyonu, işlem sırasında ortamda biriken verilerin birden fazla konumda tutundurma işlemine verilen Türkçe addır. Tüm kullanıcılar için, yapılan işlemler adına herhangi bir tutarsızlık veya sapma gibi bir durum oluşmadan veritabanında biriken verileri kaynaklardan alıp, diğer birden farklı kaynaklara taşıma olayıdır. Dağıtılmış veri tabanı olarak oluşturulan bu sistemde, veriler kişinin isteğine göre hareket eder ve iki farklı kullanıcı birbirlerinin çalışmalarına müdahale etmeden görevlerini yerine getirebilirler. Kısacası veri replikasyonu, internet ortamında bir karışıklık oluşmadan verilerin rahatça hareket edebilmesidir.

Bu işlemler sırasında tüm biriken veriler replike edilebilir. Veri Replikasyonu, yapılan işlemlerin çoğalmasının farkına vararak saptar. Böylece çoğalan veriler düzgün bir şekilde güncelleme alır ve senkronize işlemi kusursuz şekilde tamamlanır.

**Replication Nedir?**

Replication, sunucu–istemci mimarisine sahip olan tüm yapılarda, aboneler ve sunucular arasında hızlı ve düzenli bir şekilde veritabanı nesnelerini eşitleme imkânı sunan bir yönteme verilen addır.

Bir replikasyon işlemi için bir yayıncı (Publisher) veritabanı seçilir. Bu yayıncı veritabanından yayınlanacak her türlü nesneler tamamı ile belirlenir. Daha sonra yapılacak işlem, Publisher veritabanının bilgilerini ve güncellenen verileri tutması için yayıncı veritabanına aboneler olarak tanımlanır. Bu işlem sayesinde sitemde ki kaynak veritabanı, sürekli güncel halde hızlı bir şekilde başka bilgisayarda da yedeklenmiş ve depolanmış olur.

Bilişim sektöründe kendine has farklı kelimeler bulunmaktadır. Bilişim terimlerinde Aboneler (Subscribers) olarak kayda geçmiştir.

**Database Sharding Nedir**

Shard’ın sözlük anlamı “a small piece or part”, yani küçük bir parça ya da bir şeyin parçası olmak anlamına gelmektedir. Buradan yola çıkarak database sharding, en basit ifadeyle veritabanı içerisinde tutulan verilerin benzersiz küçük parçalara (shard’lara ya da chunk’lara) ayrılmasına denir.

nam-ı diğer horizontal partitioning, vertical partioning’in aksine entityleri bir bütün olarak farklı veritabanları üzerinde tutabilmeyi sağlıyor.

Daha farklı bir ifadeyle büyük tek bir veritabanı kullanmak yerine, verileri vertical partitioning’in aksine aralarında hiçbir ilişki olmayacak\* şekilde küçük parçalara bölerek farklı ve nispeten daha ucuz sunuculardaki veritabanları üzerinde tutabilmeyi sağlıyor.