**Veritabanları nelerdir?**

En temel tanımıyla bir veritabanı birbiriyle ilişkili veriler koleksiyonudur. Bir kağıt parçasına alışveriş listesi yazdığınızda küçük bir analog veritabanı oluşturmuş olursunuz. Peki bilgisayar biliminde veritabanı ne demektir? Bu bağlamda "veritabanı" market alışverişinizin envanteri gibi bilgisayar sisteminde veri olarak depolanan bilgilerin koleksiyonudur.

**Veritabanları ne için kullanılır?**

Veritabanları veri yönetimini ve erişimini kolaylaştırmak üzere verileri depolamak ve düzenlemek için kullanılır. Veri koleksiyonu büyüdükçe ve daha karmaşık hale geldikçe, bu verileri düzenli, erişilebilir ve güvenli tutmak da daha zor hale gelir. Bu konuda yardımcı olmak için bir veritabanı yönetim araçları katmanı içeren veritabanı yönetim sistemlerini (DBMS) kullanırsınız.

**Veri nedir?**

Veri terimi tek bir kişi, konum, öğe veya nesne (varlık olarak adlandırılır) hakkında yakalanan ve depolanan tüm bilgilere, ayrıca söz konusu varlığın özniteliklerine karşılık gelir.

Örneğin, yerel restoranlar hakkındaki bilgileri yakalıyor ve depoluyorsanız, her restoran bir varlıktır ve restoranın adı, adresi ve çalışma saatleri de özniteliklerdir. Gözde restoranlarınız hakkında topladığınız ve depoladığınız tüm bilgiler veridir.

**Veritabanı türleri**

Veritabanı türleri kabaca ilişkisel ve ilişkisel olmayan veritabanları olarak gruplandırılır. İlişkisel veritabanları son derece yapılandırılmıştır ve Yapılandırılmış Sorgu Dili (SQL) adlı bilgisayar dilini anlar. İlişkisel olmayan veritabanları çok çeşitlidir ve çeşitli veri yapılarını destekler. İlişkisel olmayan veritabanlarının birçoğu SQL kullanmadığından, bunlar genellikle NoSQL veritabanı olarak adlandırılır.

**Veri yapısı türleri**

**Tablo yapıları** , verileri satırlar ve sütunlar halinde düzenleyen ilişkisel veritabanı yapılarıdır; satırlar varlıkları, sütunlar da varlık özniteliklerini içerir. **Geniş tablolar** veya **geniş sütun tabloları**, tabloda bulunabilecek toplam sütun sayısını büyük oranda artırmak için boş özniteliklerin bulunduğu yedek sütunlar kullanır. Bazı alanlar boş olduğundan, geniş tablolar ilişkisel olmayan veritabanı yapısının bir örneğidir.

**Veritabanı örnekleri**

Veritabanları görünmez gizemler gibi geliyor olabilir ama çoğumuz onlarla her gün etkileşim kurarız. Burada bazı yaygın ilişkisel veritabanı, NoSQL veritabanı ve bellek içi veritabanı örnekleri verilmiştir:

**Finansal işlemler**

Bankalar, bakiye sorgularından hesaplar arasındaki transferlere kadar tüm müşteri işlemlerini izlemek için veritabanlarını kullanır. Bu işlemlerin neredeyse anında gerçekleştirilmesi gerekir ve çok büyük miktarlardaki işlemlerden gelen veriler her zaman güncel olmalıdır. Bu amaçlarla bankalar ilişkisel veritabanlarıyla oluşturulan ve çok fazla sayıda müşteriyi, işlemlerden gelen sık değişen verileri ve hızlı yanıt sürelerini işleyebilecek çevrimiçi işlem gerçekleştirme sistemlerini kullanır.

**E-ticaret katalogları**

Bir e-ticaret web siteniz varsa, kataloğunuzda her birinin kendi öznitelik çeşitleri olan tek tek ürünler yer alır. İlişkisel olmayan bir veritabanı örneği olan belge odaklı veritabanı, tek bir ürünün tüm özniteliklerini tanımlamak için tek tek belgelerden yararlanır. Diğer ürünlerden hiçbirine etki etmeden belgedeki öznitelikleri değiştirebilirsiniz. Bellek içi veritabanları genellikle veri alımını hızlandırmak ve veritabanı üzerindeki yükü azaltmak için envanter ve fiyatlandırma gibi sık erişilen e-ticaret verilerini önbelleğe almak için kullanılır.

**Sosyal ağlar**

Bir sosyal ağa katıldığınızda, bilgileriniz söz konusu ağı kullanan herkesin ilişkisel olmayan veritabanına eklenir. Bu ağdaki diğer kişilerle bağlantı kurduğunuzda sosyal bir grafın parçası haline gelirsiniz. Bu nedenle arkadaşlarınızın veya profesyonel bağlantılarınızın filtrelenmiş bir listesini görebilir, söz konusu arkadaşların ve bağlantıların tanıdığı yeni kişileri de keşfedebilirsiniz.

**Kişiselleştirilmiş sonuçlar**

İlişkisel olmayan veritabanları, artık farkına varamayacağınız kadar yaygınlaşmış olan çevrimiçi kişiselleştirmeyi yönlendirir. Bir seyahat web sitesi üzerinden uçuş rezervasyonu yaparsanız, size otel rezervasyonu ve araç kiralama seçenekleri de gösterilir. Web sitesinin veritabanında uçuş detaylarınız, seyahat tercihleriniz, önceki araç kiralamalarınız veya otel rezervasyonlarınız gibi yapılandırılmamış zengin bilgiler bulunur ve bu bilgiler size zamandan ve paradan tasarruf etmenizi veya daha az işlem yapmanızı sağlayacak kişisel öneriler sunmak için kullanılır. Benzer şekilde bellek içi veritabanları, bir uygulamayı kullanırken arama tercihleri veya alışveriş sepeti gibi geçici kullanıcı verilerinin verimli bir şekilde barındırıldığı bir oturum deposu olarak kullanılır.

**İş analizi**

Kurumlar kendi verilerinden içgörüler elde etmek istediğinde, analizlerini yönetmek için ilişkisel veritabanlarından yararlanırlar. Örneğin bir teknoloji yardım masası müşteri sorunlarını sorun türü, sorunu çözme süresi ve müşteri memnuniyeti gibi çeşitli boyutlarda izleyebilir. Tablo yapısı kullanan bir ilişkisel veritabanı, müşteri sorunu verilerini bir kerede yalnızca iki boyut kullanarak düzenler; ama çevrimiçi analitik işlem sistemiyle, yardım masası bir kerede birden fazla tabloya bakabilir. Böylelikle çok büyük miktarlarda verinin yüksek hızlarda işlenmesi için çok boyutlu analize olanak tanınır.

**Veritabanı yönetim sistemleri**

Veritabanı yöneticileri, özellikle de **büyük verilerle**çalışırken verileri denetlemek için veritabanı yönetim sistemlerini (DBMS) kullanır. Büyük veri, genellikle sistem tarafından gerçek zamanlı ya da neredeyse gerçek zamanlı olarak alınan büyük hacimli yapılandırılmış ve yapılandırılmamış verilere karşılık gelir. DBMS, birden çok uygulama arasında kullanılan ya da birden çok konumda tutulan verileri yönetmeye de yardımcı olur.

Farklı veritabanı yönetim sistemleri farklı düzenleme, ölçeklenebilirlik ve uygulama düzeyleri sağlar. Kullandığınız DBMS, düzenlemek istediğiniz veri türüne ve buna nasıl erişmek istediğinize ek olarak verilerinizin nerede tutulduğuna, veritabanınızın kullandığı mimari türüne ve nasıl ölçeklendirmeyi planladığınıza da bağlıdır.

**Verileriniz şirket içinde mi, bulutta mı yoksa her ikisinde birden mi tutuluyor?**

Veriler **şirket içi veritabanlarında**, özel şirket içi donanımında (genellikle özel bulut olarak adlandırılır) tutulur. Veri kapasitesi eklemek için, veritabanı yöneticilerinin şirket içi sunucularında yeterli alan bulunduğundan emin olmaları veya alan oluşturmak üzere altyapılarını yeni donanımla genişletmeleri gerekir.

**Bulut tabanlı veritabanlarında,** yapılandırılmış veya yapılandırılmamış veriler özel, genel veya karma bulut bilişim platformunda (özel ve genel bulut depolamasını birleştiren platform) tutulur. Bulut veritabanları sanallaştırılmış bir ortam için tasarlandığından, bunlar hem üst düzeyde ölçeklenebilir hem de üst düzeyde kullanılabilir veritabanlarıdır. Ayrıca çok kadar fazla donanım satın almanız gerekmediğinden ve yalnızca kullandığınız depolama alanı için ödeme yaptığınızdan, bunlar maliyeti düşürmeye de yardımcı olur.

**Veritabanı mimariniz merkezi, dağıtılmış veya birleştirilmiş mi?**

**Merkezi bir veri tabanında**, verilerin tümü tek sistemde, tek bir yerde tutulur. Bu tek sistem tüm kullanıcılar için erişim noktasıdır.

**Dağıtılmış bir veritabanı** hem ilişkisel hem de ilişkisel olmayan veritabanı türlerine yayılabilir. Dağıtılmış veritabanlarında veriler birden çok fiziksel konumda depolanır. Bu konumlar birden çok şirket içi bilgisayarında yer alır veya bağlantılı bilgisayarlardan oluşturulmuş bir ağa dağılmış olur.

**Birleştirilmiş veritabanında**, bağımsız sunucular üzerinde çalıştırılan birkaç ayrı veritabanı tek bir büyük nesnede birleştirilir. Blok zinciri, finansal kayıt defterlerini ve diğer işlem kayıtlarını güvenli bir şekilde yönetmek için kullanılan bir birleştirilmiş veritabanı türüdür.

**İlişkisel veritabanı nedir?**

İlişkisel veritabanları, hızlı erişim için tanımlı ilişkilerle veri noktalarını depolayan ve düzenleyen bir veritabanı türüdür. İlişkisel veritabanıyla, veriler her varlıkla ilgili bilgilerin barındırıldığı ve satırlarla sütunlar aracılığıyla önceden tanımlanmış kategorilerin gösterildiği tablolarda düzenlenir. Verilerin bu şekilde yapılandırılması, erişilmesini verimli ve esnek hale getirir. Bu nedenle ilişkisel veritabanları en yaygın olarak kullanılır. İlişkisel veritabanları; verileri depolamak, işlemek ve almak için kullanılan standartlaştırılmış bir programlama dili olan Yapılandırılmış Sorgu Dili’ni (SQL) anlamak için de oluşturulmuştur. SQL'de, tablolar oluşturmak için Veri Tanımlama Dili (DDL) adlı yerleşik bir dil ve verileri işlemek için Veri İşleme Dili (DML) adlı bir dil vardır.

İlişkisel ne anlama gelir? İlişkisel, ilişki belirten veya oluşturan anlamına gelir. Veritabanları bağlamında, ilişkisel kelimesini tanımlama şeklimiz öncelikli olarak verilerin kendisi için geçerlidir. İlişkisel olan veri kümelerinin arasında önceden tanımlanmış ilişkiler vardır. Örneğin, bir şirketin müşteri bilgilerini içeren bir veritabanı, her bir hesapta ekli olan bireysel işlem verilerini de içerebilir. İlişkisel veritabanları, depolanan veri öğeleri arasındaki ilişkiye odaklanır.

**İlişkisel veritabanının özellikleri:**

* İlişkisel veritabanları birden çok varlıktan oluşur
* Standart Sorgu Dili (SQL), ilişkisel veritabanları için standart arabirimdir
* İlişkisel veritabanları yüksek oranda yapılandırılmıştır ve şema (mantıksal ve fiziksel) kullanılarak temsil edilir
* İlişkisel veritabanları veri fazlalığı ve tutarsızlığını azaltır

**İlişkisel veritabanları nasıl yapılandırılır?**

İlişkisel model, mantıksal veri yapılarının (veri tabloları, görünümleri ve dizinleri) fiziksel depo yapılarından ayrı olduğu anlamına gelir. Bu ayrım, veritabanı yöneticilerinin, mantıksal bir yapı olarak bu verilere erişimi etkilemeden fiziksel veri depolamayı yönetebilmeleri anlamına gelir. Örneğin, bir veritabanı dosyasını yeniden adlandırdığınızda, içinde depolanan tablolar yeniden adlandırılmaz.

Ayrıca, mantıksal ve fiziksel depolama alanı arasındaki ayrım, uygulama yazılımlarının veritabanına ait veri ve yapıları [değiştirmesini](https://www.oracle.com/tr/database/what-is-data-management/) sağlayan, açıkça tanımlanmış işlemler olan veritabanı işlemleri için de geçerlidir. Mantıksal operasyonlar bir uygulama yazılımının ihtiyaç duyduğu içeriği belirlemesine olanak tanır, fiziksel operasyonlar bu verilere nasıl erişilmesi gerektiğini belirler ve ardından görevi yerine getirir.

Verilerin her zaman doğru ve erişilebilir olmasını sağlamak için ilişkisel veritabanları belirli bütünlük kurallarına uyar. Örneğin bir bütünlük kuralı, veritabanına hatalı bilgi girme olasılığını ortadan kaldırmak için bir tabloda yinelenen satırlara izin verilmediğini belirtebilir.

**İlişkisel model**

Veritabanlarının ilk yıllarında, her uygulama yazılımı kendi benzersiz yapısında veri depolardı. Geliştiriciler, bu verileri kullanmak için uygulama yazılımları oluşturmak istediklerinde ihtiyaç duydukları verileri bulmak için söz konusu veri yapıları hakkında çok şey bilmek zorundaydılar. Bu veri yapıları verimsiz, bakımı zor ve iyi uygulama yazılımı performansı sağlamak için optimize etmesi zor yapılardı. İlişkisel veritabanı modeli çoklu rastgele veri yapıları problemini çözmek için tasarlandı.

İlişkisel veri modeli, herhangi bir uygulama yazılımı tarafından kullanılabilecek verileri temsil etmek ve sorgulamak için standart bir yöntem sağlamıştır. Geliştiriciler, en baştan itibaren ilişkisel veritabanı modelinin temel gücünün, yapılandırılmış bilgileri depolamak ve bunlara erişmek için sezgisel, verimli ve esnek bir yol olan tabloları kullanmakta olduğunu fark etti.

Zamanla, geliştiriciler bir veritabanında veri yazmak ve sorgulamak için yapılandırılmış sorgu dilini (SQL) kullanmaya başladıkça ilişkisel modelin bir diğer gücü ortaya çıktı. SQL uzun yıllardır veritabanı sorgulama dili olarak yaygın şekilde kullanılmaktadır. İlişkisel cebire dayanan SQL, tüm veritabanı sorgularının performansını iyileştirmeyi kolaylaştıran, dâhilî tutarlılığa sahip bir matematik dili sağlar. Buna karşılık, diğer yaklaşımlar tekil sorgular tanımlamalıdır.

**İlişkisel veritabanı yönetim sisteminin avantajları**

Kolay ama güçlü bir yapıdaki ilişkisel model, her tür ve büyüklükteki kurumlar tarafından çok çeşitli bilgi ihtiyaçları için kullanılmaktadır. İlişkisel veritabanları, stokları takip etmek, e-ticaret işlemlerini gerçekleştirmek, çok büyük miktarlarda misyon açısından kritik müşteri bilgisini yönetmek ve çok daha fazlasını yapmak için kullanılıyor. Veri noktalarının birbiriyle ilgili olduğu; güvenli, kurallara dayalı ve tutarlı bir şekilde yönetilmesi gereken tüm bilgi ihtiyaçları için ilişkisel veritabanı kullanmayı düşünebilirsiniz.

İlişkisel veritabanları 1970'lerden bu yana kullanılıyor. İlişkisel model, avantajları sayesinde günümüzde de veritabanları için en yaygın kabul gören model olmayı sürdürüyor.

**İlişkisel model ve veri tutarlılığı**

İlişkisel model, uygulama yazılımları ve veritabanı kopyaları (bunlara örnekler adı verilir) arasında veri tutarlılığını korumakta en iyi çözümdür. Örneğin, bir müşteri bir ATM'ye para yatırıp cep telefonunda hesap bakiyesine baktığında, yatırdığı paranın güncel hesap bakiyesine derhal yansıdığını görmeyi bekler. İlişkisel veritabanları, bu tür veri tutarlılığında üstün performans gösterir ve bir veritabanının birden fazla örneğinin her zaman aynı verilere sahip olmasını sağlar.

Diğer veritabanı türlerinin büyük miktarda veriyle zamanında tutarlılık seviyesini sürdürmesi zordur. Günümüzde NoSQL gibi bazı veritabanları, yalnızca nihai tutarlılığı destekleyebilmektedir. Bu ilkeye göre, veritabanı ölçeklendiğinde veya birden fazla kullanıcı aynı verilere aynı anda eriştiğinde, verilerin "yakalanması" biraz sürer. Nihai tutarlılık bir ürün kataloğundaki girişleri tutmak gibi bazı kullanımlar için kabul edilebilir, ancak alışveriş sepeti işlemleri gibi kritik iş operasyonları için ilişkisel veritabanı hâlâ altın standarttır.

**Taahhüt ve atomisite**

İlişkisel veritabanları, iş kuralları ve politikalarını kesin kalıcılık politikalarıyla (yani, veritabanı değişikliklerini kalıcı hâle getirmek) çok ayrıntılı düzeyde hayata geçirir. Örneğin, her zaman birlikte kullanılan üç parçayı izleyen bir envanter veritabanını düşünün. Bir parça envanterden çekildiğinde diğer ikisi de çekilmelidir. Üç bölümden biri mevcut değilse, parçaların hiçbiri çekilmemelidir - veritabanı herhangi bir taahhütte bulunmadan önce üç parçanın tümü mevcut olmalıdır. İlişkisel veritabanı, üç parçanın tümü için taahhütte bulunabileceğini bilene kadar bir parça için taahhütte bulunmaz. Bu çok yönlü kalıcılık becerisine atomisite denir. Atomisite, verileri veritabanında doğru olarak tutmanın ve işletmenin kuralları, mevzuatı ve politikaları ile uyumlu olmasını sağlamanın anahtarıdır.

**ACID nitelikleri ve RDBMS**

İlişkisel veritabanı işlemlerini tanımlayan dört önemli özellik vardır: atomisite, tutarlılık, yalıtım ve dayanıklılık (genellikle ACID olarak da adlandırılır).

* **Atomisite** tam bir veritabanı işlemini oluşturan tüm unsurları tanımlar.
* **Tutarlılık** veri noktalarını bir işlemden sonra doğru durumda tutmaya ilişkin kuralları tanımlar.
* **İzolasyon** karışıklığı önlemek için, kalıcı hâle gelene kadar, bir işlemin etkisini diğer işlemlerden görünmez kılar.
* **Dayanıklılık** işlem gerçekleştiğinde veri değişikliklerinin kalıcı olmasını sağlar.

**Depolanan prosedürler ve ilişkisel veritabanları**

Veri erişimi tekrar eden birçok eylemden oluşur. Örneğin, bir veri tablosundan bilgi alırken, istenen sonucu elde etmek için kolay bir sorgunun yüzlerce veya binlerce kez tekrarlanması gerekebilir. Bu veri erişim işlevleri, veritabanına erişmek için bir tür kod gerektirir. Uygulama yazılımı geliştiricileri, her yeni uygulama yazılımında bu işlevler için yeni kod yazmak istemezler. Neyse ki ilişkisel veritabanları kayıtlı prosedürlere, yani tek uygulama yazılımı çağrısı ile erişilebilen kod bloklarına olanak tanır. Örneğin, tek bir kayıtlı prosedür, birden çok uygulama yazılımının kullanıcıları için tutarlı bir kayıt etiketleme işlevi sağlayabilir. Kayıtlı prosedürler, geliştiricilerin, uygulama yazılımındaki belirli veri işlevlerinin belirli bir şekilde uygulanmasını sağlamalarına da yardımcı olabilir.

**Veritabanı kilitleme ve eşzamanlılık**

Birden fazla kullanıcı veya uygulama yazılımı, aynı verileri aynı anda değiştirmeye çalıştığında, veritabanında çakışma yaşanabilir. Kilitleme ve eşzamanlılık teknikleri, verilerin bütünlüğünü korurken çelişki ihtimalini azaltır.

Kilitleme diğer kullanıcıların ve uygulama yazılımlarının güncellenmekte olan verilere erişmesini önler. Bazı veritabanlarında kilitleme tüm tabloya uygulanır ve bu, uygulama yazılımı performansını olumsuz etkiler. Oracle ilişkisel veritabanları gibi diğer veritabanları, kilitleri kayıt düzeyinde uygulayarak, tablodaki diğer kayıtları erişilebilir hâlde bırakır ve daha iyi uygulama yazılımı performansı elde edilmesine yardımcı olur.

Eşzamanlılık birden çok kullanıcı veya uygulama yazılımı aynı anda aynı veritabanında sorgular gerçekleştirdiğinde faaliyetleri yönetir. Bu beceri, veri kontrolü için tanımlanan politikalara göre kullanıcılara ve uygulama yazılımlarına doğru erişimi sağlar.

**İlişkisel veritabanı seçerken nelere dikkat edilmelidir?**

İlişkisel veritabanında depolanan verileri saklamak, yönetmek, sorgulamak ve almak için kullanılan yazılıma ilişkisel veritabanı yönetim sistemi (RDBMSf). denir RDBMS, kullanıcılar ve uygulama yazılımları ile veritabanı arasında bir arayüzün yanı sıra veri depolama, erişim ve performans yönetimi için yönetim işlevleri sağlar.

Veritabanı türleri ve ilişkisel veritabanı ürünleri arasında seçim yaparken çeşitli faktörler kararınıza yön verebilir. Seçtiğiniz RDBMS işletmenizin gereksinimlerine bağlı olacaktır. Kendinize aşağıdaki soruları sorun:

* Veri doğruluğu gereksinimlerimiz neler? Veri depolama ve doğruluk iş mantığına dayanıyor mu? Verimizin katı doğruluk gereksinimleri var mı (örneğin, finansal veri ve devlet raporları)?
* Ölçeklenebilirliğe ihtiyacımız var mı? Yönetilecek verilerin ölçeği ve beklenen büyüme nedir? Veritabanı modelinin, ölçeklenebilirlik için yansıtılmış veritabanı kopyalarını (ayrı örnekler olarak) desteklemesi gerekir mi? Gerekiyorsa, bu örnekler arasında veri tutarlılığını koruyabilir mi?
* Eşzamanlılık ne kadar önemli? Birden fazla kullanıcı ve uygulama yazılımının eşzamanlı veri erişimine ihtiyacı olacak mı? Veritabanı yazılımı verileri korurken eşzamanlılığı destekliyor mu?
* Performans ve güvenilirlik ihtiyaçlarımız neler? Yüksek performanslı, güvenilir bir ürüne ihtiyacımız var mı? Sorgu-yanıt performansı için gereksinimler neler? Satıcının hizmet düzeyi anlaşmaları (SLA'lar) veya planlanmamış kesinti süreleri için taahhütleri nelerdir?

**Geleceğin ilişkisel veritabanı: Kendi kendini yöneten veritabanı**

İlişkisel veritabanları yıllar içinde giderek daha iyi, daha hızlı, daha güçlü ve daha kolay kullanılır hâle geldi. Ancak aynı zamanda daha karmaşık bir hale de geldiler ve veritabanı yönetmek, uzun süredir tam zamanlı bir iş durumunda. Geliştiriciler uzmanlıklarını işletmeye değer katacak yenilikçi uygulama yazılımları geliştirmeye odaklanmak için kullanamıyor. Zamanlarının çoğunu veritabanı performansını optimize etmek için gereken yönetim faaliyetlerine harcamak zorunda kalıyorlar.

Bugün, [kendi kendini yöneten teknoloji](https://www.oracle.com/tr/autonomous-database/) yeni bir ilişkisel veritabanı türü sunmak için [bulut veritabanı](https://www.oracle.com/tr/database/what-is-a-cloud-database/) teknolojisi ve [makine öğreniminin](https://www.oracle.com/tr/artificial-intelligence/machine-learning/what-is-machine-learning/?source=:so:tw:or:awr:ocl::cloud) güçlü yönleri üzerinde yükselmektedir. Kendi kendini yöneten veritabanı (diğer bir deyişle otonom veritabanı) ilişkisel modelin güç ve avantajlarını korurken sorgu performansını artırmak ve yönetim görevlerini izlemek için yapay zekâ (AI), makine öğrenimi ve otomasyon da kullanıyor. Örneğin, sorgu performansını artırmak için kendi kendini yöneten veritabanı, sorguları daha hızlı hale getirmek için dizinleri hipotezleyebilir ve test edebilir ve ardından en iyilerini kendi başına canlı kullanıma aktarabilir. Kendi kendini yöneten veritabanı, bu geliştirmeleri insan müdahalesine gerek kalmadan sürekli yapar.