**Kavramsal, mantıksal ve fiziksel veri modelleri**

Veri modelleme, dokümantasyon uğruna diyagramlar oluşturmakla ilgili değildir. İşletme ve veri ekipleri arasında ortak bir anlayış oluşturmak, güven oluşturmak ve verilerle değer sunmakla ilgilidir. Aynı zamanda bir yatırımdır. Veri sistemlerinizin istikrarına, güvenilirliğine ve gelecekteki uyarlanabilirliğine yapılan bir yatırımdır. Tüm değerli girişimler gibi, önceden biraz ek çaba gerektirecektir. Yine de uzun vadede, iletişim, verimlilik, veri kalitesi ve ölçeklenebilirlik açısından yatırımların getirileri önemlidir.

[Bu makalede, kavramsal, mantıksal ve fiziksel modellerin veri modellemesinin](https://www.thoughtspot.com/data-trends/data-modeling) bu önemli yönlerini nasıl etkilediğini anlayarak bunların işinizi nasıl olumlu etkileyebileceğini göreceksiniz :

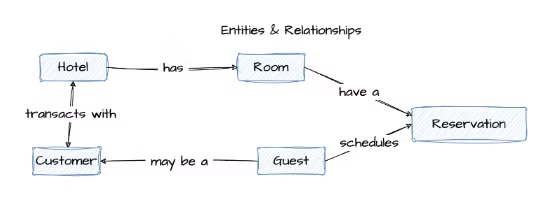
* Paylaşılan anlayış ve iletişim
* Veri ilişkileri, yapısı ve organizasyonu
* Performans, ölçeklenebilirlik ve verimlilik
* Veri kalitesi ve tutarlılığı
* Uyarlanabilirlik, ölçeklenebilirlik ve geleceğe hazırlık

Ayrıca, dünya standartlarında bir veri modeli oluşturmak ve paydaşlarınızın desteğini almak için ihtiyaç duyacağınız bilgiyi de edineceksiniz.

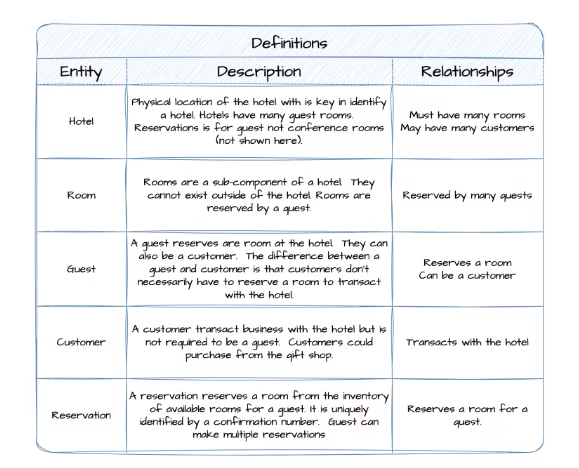
Kavramsal veri modeli nedir?

Kavramsal veri modeli (CDM), kuruluşun veri ihtiyaçları hakkında kapsamlı bir bakış açısı sağlayarak yüksek düzeyde çalışır. Bir işletmenin günlük operasyonlarında kullandığı veya kullanmayı planladığı verilerin geniş ve basitleştirilmiş bir görünümünü tanımlar. [Kavramsal veri modelleme,](https://www.thoughtspot.com/data-trends/data-modeling/conceptual-data-model-examples) bir iş sürecinin temel kavramlarını yakalayarak işletme hakkında ortak bir anlayış yaratmayı amaçlar. Bu temel kavramlar genellikle bir Varlık İlişki Diyagramı'nda (ERD) ve eşlik eden varlık tanımlarında yakalanır.

Kavramsal bir veri modeli geliştirmek, ekibinizin ve paydaşlarınızın temelleri ve genel resmi anlamalarına yardımcı olur; ne tür verilerle çalıştığınızı ve farklı veri varlıklarının nasıl ilişkili olduğunu. Ayrıca, iş süreci hakkında paylaşılan bir anlayış ve tüm teknik ve teknik olmayan üyeler için ortak bir dil oluşturur. Birçok veri ekibi paydaşlarıyla iletişim ve güven konusunda zorluk çekerken, CDM etkili iletişimi teşvik ederek yardımcı olabilir.



Şekil 1 - Örnek kavramsal veri modeli—otel rezervasyonları



Şekil 2 - Örnek kavramsal veri modeli varlık tanımları

Kavramsal veri modelinin faydaları

Yani, artık kavramsal veri modelinin net bir tanımına sahip olduğunuza göre, muhtemelen faydaları nelerdir diye düşünüyorsunuzdur. Kavramsal veri modelinin faydaları iki alana ayrılır: paylaşılan anlayış yaratmak ve iletişimi ve iş birliğini geliştirmek. Her ikisi de daha büyük bir paydaş güveni duygusuna yol açar. Kavramsal model ayrıca fiziksel bir modelin yalnızca doğru sonuçlar üretmekle kalmayıp aynı zamanda işletmenin geçerli sorular sormasını sağlayacak şekilde tasarlanmasını sağlar. Örneğin, bir müşterinin birden fazla rezervasyonu olabilir mi? Yoksa müşteri ve misafir her zaman aynı mıdır?

Paylaşılan anlayış

Kavramsal veri modelleri, veri nesnelerini ve bunların ilişkilerini tanımlamak ve tanımlamak için basit, iş odaklı terminoloji kullanır ve hem teknik hem de teknik olmayan paydaşların anlayabileceği evrensel bir dil oluşturur. Ayrıca, özellikle karmaşık ilişkiler için yazılı belgelerden daha kolay anlaşılan görsel temsiller sağlarlar. İş sürecinin bu ortak anlayışı, paydaş güveni ve inancı oluşturmak için çok önemlidir.

İletişim ve işbirliği

Kavramsal modeller, paydaşlar, veri ekipleri ve farklı iş kolları arasındaki tartışmalar için bir odak noktası görevi görebilir, iş birliğini ve güveni teşvik edebilir. Ayrıca, gereksinimleri netleştirmeye, boşlukları veya fazlalıkları belirlemeye ve iş kuralları ve politikaları üzerinde uyumu teşvik etmeye yardımcı olabilirler. Kısacası, iletişim ve iş birliği, maliyet, teslimat ve işlevsellik konusunda önemli kararlar alırken ihtiyaç duyacağınız güvene yol açar.

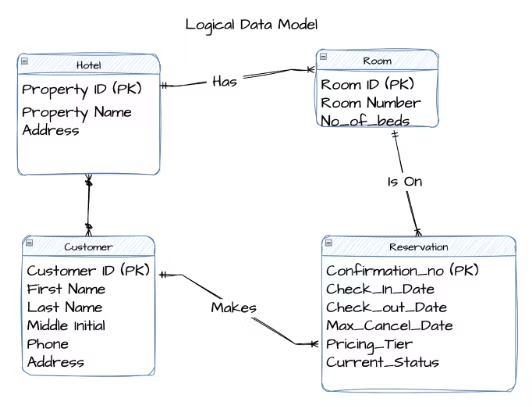
Mantıksal veri modeli nedir?

Mantıksal bir veri modeli (LDM), verilerdeki ilişkileri tam olarak tanımlayan, temel varlıkların ayrıntılarını ve yapısını ekleyen temsiller içerir. LDM'nin iş ihtiyaçlarına, esnekliğe ve taşınabilirliğe odaklandığı için veri platformundan bağımsız kaldığını belirtmek önemlidir.

LDM, her varlığın belirli niteliklerini, varlıklar arasındaki ilişkileri ve bu ilişkilerin kardinalitesini içerir. Sistemlerinizi oluştururken ekibinize takip edebileceğiniz sağlam bir çerçeve sunar ve [bulut veri ambarlarını](https://www.thoughtspot.com/data-trends/data-storage/what-is-a-cloud-data-warehouse) , veri depolarını, uygulama veritabanlarını veya veri analizi veri kümelerini etkili ve verimli bir şekilde planlamak ve uygulamak için kullanılabilir.

Kavramsal veri modelinde, varlıklar ve ilişkiler tanımlandı. Bir sonraki adım, kavramsal bir veri modelinden mantıksal bir veri modeline geçmek için [veri modelleme en iyi uygulamalarını](https://www.thoughtspot.com/data-trends/data-modeling/data-modeling-best-practices-for-analytics-and-data-engineers) kullanmaktır ; genellikle aşağıdaki adımların gerçekleştirilmesi gerekir:

1. Varlıkların ve ilişkilerin CDM'de doğru olduğunu doğrulayarak **CMD'yi doğrulayın .**
2. Modeldeki her varlık için **ek nitelikleri belirleyin , varlığı tamamlamak için gereken diğer nitelikleri belirleyin.**
3. Varlıktaki tüm öznitelikleri inceleyerek **aday anahtarları belirleyin ve hangi özniteliklerin varlığı benzersiz şekilde tanımladığını belirleyin. Birden fazla olabilir. Amaç, birincil ve benzersiz anahtarlar için hangi özniteliklerin en iyi adaylar olduğunu belirlemektir.**
4. **Paydaşlarınızla işbirliği içinde birincil anahtarları seçin** aday anahtarlar listesinden aday özniteliklerden birincil anahtarı seçin. En iyi uygulama birincil anahtar olan tek bir özniteliğe sahip olmaktır, ancak bu her zaman mümkün değildir.
5. **Belirli uygulamanız için veri modeline göre normalleştirin** . Bu durumda, varlığımızdan tekrar eden öğeleri kaldırarak ve tüm özniteliklerin yalnızca birincil anahtara bağlı olmasını sağlayarak yedekliliği en aza indiren 3. Normal Formu (3NF) kullanıyoruz. Analitik modelleme için yaygın bir yaklaşım, boyutlu modelleme kullanmaktır. Amaç yedekliliği ortadan kaldırmak olan 3NF'nin aksine, boyutlu modelleme, birleştirmeleri en aza indirmek ve sorgu performansını iyileştirmek için denormalizasyonu içerir. **Profesyonel İpucu:** Varlıklar 3NF gereksinimlerini karşılayamadığında, çözüm genellikle yeni bir varlık ve ilişki oluşturmaktır.
6. Normalleştirme yeni varlıklar, öznitelikler, ilişkiler veya özniteliklerin yeni varlıklara yeniden dağıtılmasıyla sonuçlanabileceğinden   **ERD varlıklarını oluşturun .**
7. **Varlık ilişkilerini tanımlayın** ve paydaşlarla doğrulayarak ilişkilerin doğru olduğundan emin olun. **Profesyonel İpucu:** Varlıklar genellikle yalnızca birkaç başka varlıkla ilişkilidir, bu nedenle birçok ilişki bir tasarım sorununa işaret edebilir. Örneğin, Şekil 3'te Otel varlığı, "Sahip" ilişkisi aracılığıyla bir Oda varlığına bağlıdır.
8. **İlişki kardinalitesini,** bir varlığın kaç örneğinin ilişkili varlıktaki örneklerle ilişkili olduğunu belirleyerek tanımlayın. En yaygın kardinaliteler bire bir, bire çok ve çoktan çoğadır. Kardinalite isteğe bağlı da olabilir. Örneğin Şekil 3'te, Otel ve Oda varlıkları arasındaki bire çok ilişkiyi görebilirsiniz. Bu genellikle "bir Otelin birçok Odası vardır" şeklinde okunur.
9. **Mantıksal veri modelini iş gereksinimlerine göre doğrulayarak modeli doğrulayın** . Bu, modelin gerekli verileri ve ilişkilerini doğru bir şekilde temsil ettiğinden emin olmak için paydaşlar, son kullanıcılar ve geliştiricilerle kontrol etmeyi içerir.
10. **En iyi iş değeri için yineleyin ve iyileştirin** . Modelleme genellikle yinelemeli bir süreçtir, bu nedenle geri bildirime göre mantıksal veri modelini iyileştirmeniz veya güncellemeniz gerekebilir.



Şekil 3 - Mantıksal veri modeli örneği

Mantıksal veri modelinin faydaları

Kavramsal veri modelinde yaptığımız gibi, mantıksal veri modelinin birincil faydalarını gözden geçirelim. Mantıksal veri modellemenin birincil faydaları iki kategoriye ayrılır: çevik veri modelleme ve veri organizasyonu. Her ikisi de girişimin kapsamını korur ve verimliliği artırır.

Çevik veri modelleme

Mantıksal veri modellemesi, çevik olabilen yinelemeli bir süreçtir. Mantıksal veri modelleri teknolojiye bağımlı olmadığından, paydaşlarla iş mantığı ve iş süreci üzerinde yineleme yapmak için bir kanal sağlar. Bu esneklik, gelecekteki iş ihtiyaçları için uyarlanabilirlik ve ölçeklenebilirlik sağlar.

Veri ilişkileri, yapısı ve organizasyonu

İyi bir LDM, verinin yapısı ve karşılıklı ilişkilerinin net bir resmini sunarak sistemin anlaşılmasını kolaylaştırır. Model, yedekliliği önlemek ve veri bütünlüğünü korumak için tutarlılığı ve standardizasyonu teşvik eder. Mantıksal bir veri modeli, nihayetinde fiziksel veri modeli için bir taslak görevi görür ve veritabanı tasarımını kolaylaştırır. Veritabanı yöneticilerinin ve yazılım mühendislerinin fiziksel veritabanını verimli ve etkili bir şekilde tasarlamalarını sağlar.

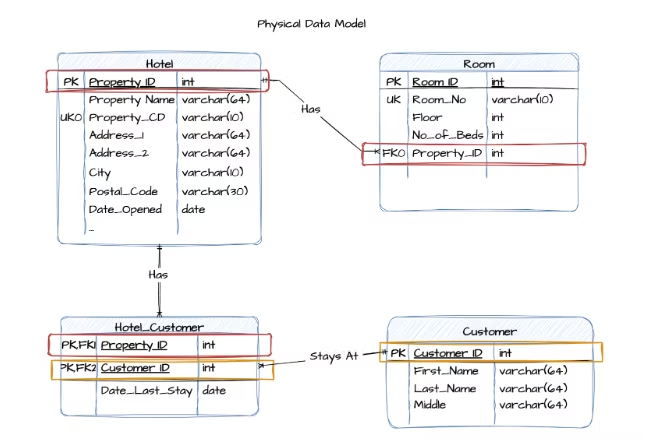
Kalite güvencesi

Sistem geliştirmenin erken aşamalarında hataları veya tutarsızlıkları belirleyerek mantıksal veri modelleri daha yüksek veri kalitesine, güvenilirliğe ve daha düşük maliyete katkıda bulunur. LDM'ler ayrıca veri yapısını iş ihtiyaçlarıyla uyumlu hale getirerek iş gereksinimlerini doğrular.

Elbette, kuruluşunuz için başka faydalar da geçerli olabilir, ancak mantıksal veri modellemesinin, verileri bir iş perspektifinden daha iyi anlamak, paydaşlar arasında daha iyi iletişim kurmak ve fiziksel veritabanı tasarımı için sağlam bir temel oluşturmak açısından çok önemli olduğunu unutmayın.

Fiziksel veri modeli nedir?

Fiziksel veri modeli (PDM), ilişkisel veri nesnelerini temsil eden bir veri modelidir. Veri modelinin teknolojiye özgü ve veritabanına özgü uygulamasını tanımlar ve mantıksal bir veri modelinden çalışan bir veritabanına dönüştürmenin son adımıdır. Fiziksel veri modeli, bir veritabanı oluşturmak için gereken tüm fiziksel ayrıntıları içerir.



Şekil 4 - Fiziksel veri modeli

Mantıksal bir veri modelinden fiziksel bir veri modeline geçiş yinelemeli bir süreçtir. İstenilen veritabanı tasarımını elde etmek için veri modelinin daha da iyileştirilmesini içerir. İyi fiziksel veri tasarımı genellikle veri platformları ve modelleme konusunda derinlemesine bir anlayış gerektirir. Bu süreçte yer alan genel adımlar şunlardır:

1. **Veri modelinin bulunacağı veri platformunu seçin** çünkü bu gelecekteki fiziksel veri tasarım kararlarını etkileyecektir. Bu aynı zamanda önemlidir çünkü model platforma özgü yeteneklerden yararlanacaktır.
2. **Mantıksal varlıkları** fiziksel tablolara dönüştürün. Mantıksal modelinizdeki her varlığın bir tabloya veya hatta birden fazla tabloya dönüştürülmesi gerekir. Mantıksal veri modelleme süreci sırasında aday anahtarların belirlendiğini hatırlayın. Aynı aday anahtarlardan tablo için birincil bir anahtar seçin.
3. Mantıksal modeldeki her özniteliği karşılık gelen tablodaki bir sütuna dönüştürerek **sütunları tanımlayın . Ek olarak, her sütun için veri türünü (integer, varchar, date, vb.) tanımlayın.**
4. **Üst ve alt tablolar arasındaki ilişkileri tanımlayın** . Bu, üst tablonun birincil anahtar (PK) niteliğine başvuran alt tabloda bir yabancı anahtar (FK) niteliği oluşturarak elde edilir.
5. **Tabloların, ortadan kaldırılmış veri fazlalığını ve veri bütünlüğünü garanti eden 3NF'de olduğunu doğrulayın** . Bu, tabloların 3NF normalizasyon prensiplerine ve veri platformunun belirli yeteneklerine göre bölünmesini veya birleştirilmesini içerebilir. Şekil 4'te, Hotel ve Customer arasındaki çoktan çoğa ilişkiyi hesaba katmak için Hotel\_Customer adlı bir tablo eklendi.
6. **Sıralama, filtreleme ve tabloları birleştirme için en sık kullanılan öznitelikleri belirleyerek dizinleri ve bölümleri tanımlayın** . Dizinleri ve bölümleri optimize etmek, veri platformuna ve kullanım durumuna göre büyük ölçüde değişir, bu nedenle genellikle çok yinelemeli bir işlemdir. Dizinleri ve bölümleri optimize etmeye başlarken KISS (basit tutmak akıllıca) yapmak en iyisidir.
7. **Tablo kısıtlamalarını,** birincil anahtarları, benzersiz anahtarları, null/noll olmayan kontrollerini ve tabloda diğer mantıksal kısıtlamaları tanımlayarak uygulayın. LDM'de tanımlanan mantıksal kısıtlamalar, veri platformu bu kısıtlamaları uygulamasa bile fiziksel veri tasarımında da ifade edilmelidir. **Profesyonel İpucu** : Modern [bulut veri platformları,](https://www.thoughtspot.com/data-trends/cloud/cloud-data-platform) kısıtlamalar tanımlanabilse bile genellikle tüm kısıtlamaları uygulamaz. Kısıtlamalar uygulanmasa da, veri bütünlüğünü uygulamak, karmaşık veri analizleri oluşturmak ve performansı iyileştirmek için alt akış uygulamaları ve analiz araçları tarafından kullanıldıkları için yine de değerlidirler.
8. Görünümler, saklı yordamlar, tetikleyiciler, akışlar ve görevler oluşturarak veri modelinizin **programlanabilirlik yönlerini uygulayın . Veri platformunuzun karmaşıklığına bağlı olarak, görünümler (bir sorguya dayalı sanal tablolar), saklı yordamlar (önceden derlenmiş**[SQL analitik](https://www.thoughtspot.com/data-trends/data-science/sql-analytics-functions) ifadeleri grupları), tetikleyiciler (olaylara yanıt olarak otomatik eylemler), Akışlar (olayların neredeyse gerçek zamanlı alımı) ve Görevler (otomatik [veri hatları](https://www.thoughtspot.com/data-trends/data-integration/what-is-a-data-pipeline-and-how-do-you-build-one) ) uygulamanız gerekebilir.
9. **Model doğrulaması ve standartları,** PDM'nin mantıksal modelde tanımlanan iş gereksinimlerini karşıladığından emin olmak için paydaşlarla doğrulanmalıdır. Doğrulama testi ayrıca iyi tanımlanmış test verileri üzerinde gerçekleştirilmelidir. Fiziksel veri modelini, işletme tarafından tanımlanan soruları yanıtlayan sorgular oluşturarak doğrulamak ve doğrulamak yaygındır.

Fiziksel veri modellemede birden fazla adım ve yineleme olsa da, iyi tanımlanmış bir LDM'ye sahip olmak fiziksel modelleme sürecinin verimli ve etkili olmasını sağlar. PDM, iş ve veri gereksinimlerini doğru bir şekilde temsil eden ve seçilen platformda verimli bir şekilde çalışan bir model oluşturmayı hedefler.

Fiziksel veri modelinin faydaları

Çoğu veritabanı karmaşıktır, düzenli olarak değişir ve gelişir, peki fiziksel veri modellemenin ve fiziksel veri modeline (PDM) sahip olmanın faydaları nelerdir? Veri ekibi için birincil faydalar hız iken iş ekipleri ölçeklenebilirlik ve geleceğe hazırlıktan faydalanır.

Hız ve verimlilik

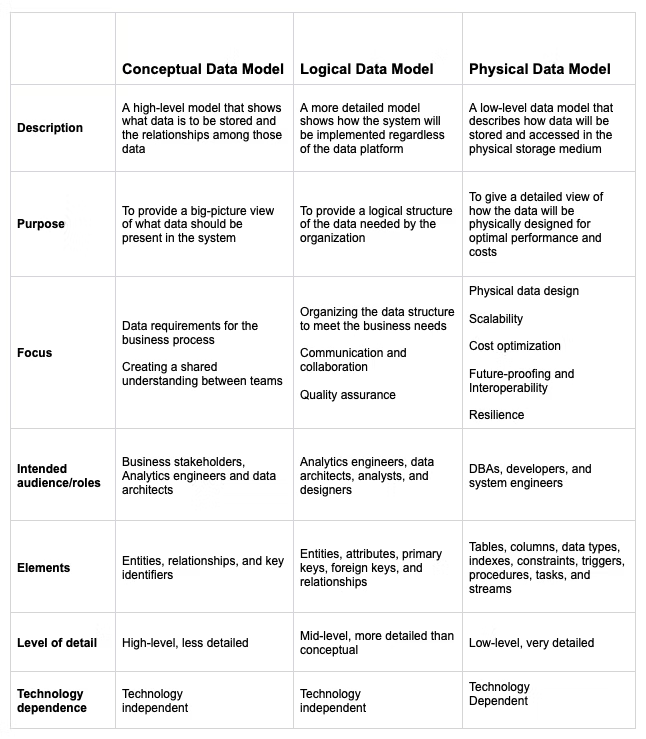
Fiziksel veri modeli (PDM), verilerinizin fiziksel olarak nasıl depolanacağı, yapılandırılacağı ve güvenli bir şekilde nasıl erişileceği ile ilgilidir ve belirli bulut veri platformu veya şirket içi veritabanı için optimize edilmiştir. PDM'ler maliyet, ölçek, depolama ve performans için en uygun yapılandırmayı sağlar. Tüketime dayalı veri platformlarının olduğu bir dünyada, iyi modellenmiş bir fiziksel veri modeline sahip olmak nihayetinde zamandan, paradan ve kaynaklardan tasarruf sağlar.

Ölçeklenebilirlik ve geleceğe hazırlık

İyi modellenmiş bir fiziksel veri modeli, işiniz büyüdükçe ve geliştikçe kaçınılmaz değişiklikleri yönetmeyi çok daha kolay hale getirir. PDM'ler, yeni veri kaynaklarına, iş kurallarına veya sistem entegrasyonlarına uyum sağlayabilen bir plan ve ölçeklenebilir çerçeve sunarak daha düşük sistem karmaşıklığı, azaltılmış eskime riski ve daha hızlı pazar hızı sağlar.

Kavramsal, mantıksal ve fiziksel veri modellerinin karşılaştırılması

Tablo 1 yukarıda açıklanan kavramsal, mantıksal ve fiziksel veri modellerini özetlemektedir.



Tablo 1 - Model tiplerine genel bakış

Her şeyi toparlamak

Verileri kavramsal, mantıksal ve fiziksel modellerle modellemeli misiniz? Bunu yapmanızı şiddetle tavsiye ederim. Doğru [veri modelleme araçlarına](https://www.thoughtspot.com/data-trends/data-modeling/data-modeling-tools) yatırım yapmak , gelecekte size pazara sunma hızının artması, daha yüksek kalite, daha düşük maliyet, azaltılmış ürün riskleri ve paydaşlarınızla büyük güven ve iş birliği dahil olmak üzere önemli getiriler sağlayacaktır.

Ne yazık ki, bazı veri ekipleri [modern veri yığınında veri modelleme](https://www.linkedin.com/video/live/urn:li:ugcPost:6966031689688981504/) için en iyi uygulamaları göz ardı ediyor ve One Big Table (OBT) yaklaşımını kullanarak modellemeye devam ediyor. OBT'ler genellikle bir ızgarada veya Excel'de verilerle çalışmanın algılanan basitliği ve benzerliği için seçilse de, yatırım getirinizi en üst düzeye çıkarma fırsatını kaçırıyorlar.

Veri modellemesi çevik olmalı ve artımlı sürümler halinde sunulmalı, böylece pazara sunma süresi ve maliyetleri azaltılmalıdır. Kuruluşunuzun mevcut ve gelecekteki başarısını garantileyen değerli bir iş yatırımıdır. Veride, "şimdi öde ya da sonra öde" şeklindeki eski söz aslında "şimdi öde ya da sonra 10 katı öde"dir.

Bir sonraki veri projenizde kendinize birkaç soru sorun.

* Bir sonraki veri varlığını mı oluşturuyorum yoksa sadece bir sonraki teknoloji borcunu mu?
* Paydaşlar ve ben iş süreci ve modeli konusunda ortak bir anlayışa sahip miyiz?
* Modelimiz daha fazla değer yaratmak için [self servis analitiği destekliyor mu?](https://www.thoughtspot.com/data-trends/business-intelligence/self-service-bi)
* [Veri kalitesinin](https://www.thoughtspot.com/data-trends/data-science/data-quality) iyileştirilmesi bana veya ekibime olan güveni ve inancı artıracak mı?
* Karmaşıklığı azaltıp veri varlıklarımızı geleceğe hazırlayabilir miyiz?

O halde hemen harekete geçin ve paydaşlarınızı memnun edecek, iş değeri yaratacak dünya standartlarında bir veri ürünü oluşturmaya başlayın!

**İş Kuralları, Veri Modelleri, Varlık Bağıntı Modeli**

**Konular**

* Veritabanı Geliştirme Yaşam Döngüsü
* İş Kuralları (Business Rules)
* Veri Modeli Temel Bileşenleri
* İş Kurallarını Veri Modeline Dönüştürme
* Veri Modellerinin Gelişimi
  + Dosya Sistemi
  + Hiyerarşik Model
  + Ağ Modeli
  + İlişkisel Model
  + Varlık Bağıntı Modeli
  + Nesne Yönelimli Model
  + Yeni Veri Modelleri
* Veri Soyutlama
* Varlık Bağıntı Modeli
  + Temel Kavramlar
  + Varlıklar Arası Bağıntılar
  + Var Olma Bağımlılığı (Existence Dependency)
  + Zayıf/Güçlü Bağıntılar
  + Bağıntı Dereceleri
  + Varlık Bağıntı Modeli (VBM/ERM) ile Tasarım
* Kaynaklar

**Veritabanı Geliştirme Yaşam Döngüsü**

[metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu](https://github.com/celalceken/DatabaseManagementSystems/blob/master/Sekiller/02/YasamDongusu.png)

**İş Kuralları (Business Rules)**

* Veritabanı (varlık, nitelik, bağıntı ve kısıtlar) oluşturulurken iş kurallarına bakılır.
* **İş kuralı:** Veritabanı tasarımı yapılacak organizasyon ile ilgili işleyiş, kural ya da yönetmeliğin özetlenmiş şekline iş kuralları denilebilir. İş kuralları ihtiyaç listesine benzer.
* Örnek iş kuralları:
  + Bir müşteri çok sayıda sipariş verebilir.
  + Her müşterinin adı, soyadı, telefon numarası vs. istenir.
  + Öğrenciler bir ara sınav ve bir yarıyıl sonu sınavına girerler.
* İş kurallarının kaynağı, son kullanıcılar, yöneticiler, kural koyucular ve yazılı dokümanlar (standart, yönetmelik vs.) olabilir.
* İş kurallarını oluşturmak için doğrudan son kullanıcılarla görüşmek oldukça etkili bir çözümdür.
* Veritabanı tasarımı açısından iş kurallarının önemi;
  + Kullanıcılar ile tasarımcılar arasındaki iletişimi sağlar.
  + Tasarımcının verinin doğasını, önemini ve kapsamını anlamasını sağlar.
  + Tasarımcının iş süreçlerini anlamasını sağlar.
  + Tasarımcının doğru bir veri modeli geliştirmesine yardım eder (veriler arası ilişkiler ve kısıtların kolayca belirlenmesini sağlar).
  + Kuruluşun veriye bakışını standart haline getirir.
* İş kuralları oluşturulduktan sonra, gerçekleştirilecek veritabanının modellenmesi aşamasına geçilir.

**Veri Modeli**

* **Veri modeli:** Karmaşık gerçek dünya veri yapılarının basit olarak gösterilmesi (genellikle grafiksel) için kullanılan araca veri modeli ismi verilir.
* Veri modeli, veritabanı tasarımcıları, uygulama programcıları ve son kullanıcılar arasındaki iletişimi kolaylaştırır.
* Veri modelleri sayesinde veritabanı tasarımını gerçekleştirmek daha kolay olur.
* Veri modelleme yinelemeli (iterative) bir işlemdir. Önce basit model oluşturulur. Daha sonra ayrıntılar eklenir. En sonunda veritabanı tasarımında kullanılan şablon (blueprint) elde edilir.

**Veri Modelinin Temel Bileşenleri**

* **Varlık (Entity):** Hakkında veri toplanan ve saklanan her şey (öğrenci, ders, personel vb.). Gerçek dünyadaki nesneleri ifade eder. Var olan ve benzerlerinden ayırt edilebilen her şey.
* **Varlık kümesi (Entity set):** Aynı türden varlıkların oluşturduğu kümeye denir (Öğrenciler, Dersler vb.).
* **Nitelik (Attribute):** Varlığın sahip olduğu özellikler.
* **Bağıntı (Relationship):** Varlıklar arasındaki ilişkiyi ifade eder.
  + Bir-Çok (One to Many 1:M)
    - Bir müşteri çok sayıda sipariş verebilir.
    - Her sipariş yalnızca bir müşteri tarafından verilir.
  + Çok-Çok (Many to Many M:N)
    - Bir öğrenci çok sayıda ders alabilir.
    - Her ders çok sayıda öğrenci tarafından alınabilir.
  + Bir-Bir (One to One 1:1)
    - Bir mağaza bir personel tarafından yönetilir.
    - Bir personel bir mağazayı yönetir.
* **Kısıtlar (Constraints):** Veri üzerindeki sınırlamalardır. Veri bütünlüğünün sağlanması açısından önemlidir. Örneğin;
  + Öğrenci notunun 0-100 arasında olması
  + T.C. kimlik numarasının 11 karakter olması
  + Aynı ürünün birden fazla kayıt edilememesi

**Veri Modellerinin Gelişimi**

* Dosya Sistemi
* Hiyerarşik Model
* Ağ Modeli
* İlişkisel model
* Varlık Bağıntı modeli
* Nesne Yönelimli Model
* Yeni Veri Modelleri

**Dosya Sistemi**

* 1960-1970 lerde çoğunlukla IBM ana çatı (mainframe) sistemlerde kullanılmıştır
* Dosyalar arasında ilişki yoktur.
* Veri tekrarı ve sıralı tarama gibi sorunları bulunmaktadır.
* Örnek bir dosya yapısı aşağıda görünmektedir.

[metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu](https://github.com/celalceken/DatabaseManagementSystems/blob/master/Sekiller/02/DosyaYapisi.png)

**Hiyerarşik Model**

* 1960’larda büyük miktardaki verileri yönetebilmek için geliştirilmiştir.
* Veriler ağaç yapısı şeklinde organize edilir.
* Ana-çocuk (parent-child) arasında 1:M ilişkisi vardır. Kayıtların sadece 1 ana (parent) kaydı vardır.

[diyagram, metin, ekran görüntüsü, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu](https://github.com/celalceken/DatabaseManagementSystems/blob/master/Sekiller/02/HiyerarsikModel.png)

**Ağ Modeli**

* 1970’lerde geliştirilmiştir. Veritabanı başarımını artırmak üzere daha karmaşık ilişkilere izin verilir.
* Hiyerarşik modelden farklı olarak kayıtların birden fazla ana (parent) kayıtları olabilir.
* Ağ modeliyle birlikte ortaya çıkan ve hala kullanılan bazı kavramlar aşağıdadır.
  + **Şema:** Tüm veritabanının, veritabanı yöneticisi tarafından görünen kavramsal organizasyonu.
  + **Alt şema:** Veritabanının istenen bilgiyi üreten uygulama programı tarafından görünen kısmı.
  + **Veri işleme dili (data manipulation language, DML):** Veritabanında bulunan verilerin, sorgulama işlemleri yapılarak güncellenmesi, yeni verilerin eklenmesi ve olan verilerin silinme işlemlerinin yapılmasını sağlayan dil.
  + **Veri tanımlama dili (data definition language, DDL):** Veritabanında bulunan verilerin tip, yapı ve kısıtlamalarının tanımlanmasını sağlayan dil.
  + **Anlık sorgu (ad hoc query):** Yazılımlarla birlikte gelmeyen kullanıcının kendi oluşturduğu sorgulara verilen isimdir.
* Ağ modelinin dezavantajı, çok basit sorgular için bile karmaşık program kodlarının kullanımını gerektirmesidir.

[metin, çizgi, diyagram, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu](https://github.com/celalceken/DatabaseManagementSystems/blob/master/Sekiller/02/AgModeli.png)

**İlişkisel Model**

* 1970’de E. F. Codd tarafından ortaya atılmıştır (A Relational Model of Data for Large Shared Databanks, Communications of the ACM, June 1970, pp. 377−387).
* İlişkisel Veritabanı Yönetim Sistemleri (Relational Database Management Systems, RDBMS) tarafından kullanılır.
* RDBMS’nin en önemli özelliklerinden birisi ilişkisel modelin karmaşık yapısını kullanıcıdan gizlemesidir.
* Kullanıcı, ilişkisel modeli, verileri içeren tablolardan oluşan bir yapı gibi görür.
* Tablolar birbirlerine ortak alanlarla bağlanırlar.
* İlişkisel şema, varlıklar, varlıkların nitelikleri ve aralarındaki bağlantıların gösteriminden oluşur.

[metin, diyagram, çizgi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu](https://github.com/celalceken/DatabaseManagementSystems/blob/master/Sekiller/02/IliskiselSema.png)

* İlişkisel veritabanı modelinin en güçlü yanlarından birisi, verileri yönetmek için yapısal sorgulama dili (structured query language, SQL) dilinin kullanılıyor olmasıdır.
* SQL dili nasıl yapılması gerektiğini anlatmak yerine ne yapılması gerektiğinin ifade edildiği basit bir dildir.
* Bu nedenle, SQL kullanılarak veri tabanlarının tasarımı ve yönetimi daha kolaydır.
* İlişkisel bir veritabanı yönetim sistemi 3 temel bileşenden oluşur.
  + Verilerin saklandığı veritabanı
  + SQL komutlarını derleyerek istenenleri gerçekleştiren SQL Motoru (SQL Engine)
  + Kullanıcılarla iletişimi sağlayan arayüzler.

**Varlık Bağıntı Modeli**

* İlişkisel model daha önceki modellere göre çok daha kullanışlı olmasına rağmen veritabanı tasarımı için ilişkisel modelin grafiksel gösterimi olan varlık bağıntı modeli (VBM) (entity relationship model, ERM) daha sık kullanılır. 1976’da Peter Chen tarafından önerilmiştir.
* İlişkisel modelinin tamamlayıcısı olduğu için kullanımı oldukça yaygınlaşmıştır.
* i) Chen gösterimi ve ii) Crow’s Foot gösterimi sıkça kullanılan gösterim şekillerindendir.
* Ders kapsamında Crow’s Foot gösterimi kullanılacaktır.

**Chen Gösterimi**

[diyagram, taslak, çizim, origami içeren bir resim

Açıklama otomatik olarak oluşturuldu](https://github.com/celalceken/DatabaseManagementSystems/blob/master/Sekiller/02/ChenGosterimi.png)

**Crow’s Foot Gösterimi**

[metin, diyagram, plan, teknik çizim içeren bir resim

Açıklama otomatik olarak oluşturuldu](https://github.com/celalceken/DatabaseManagementSystems/blob/master/Sekiller/02/ETicaret.png)

**Nesne Yönelimli Model**

* Nesne yönelimli programlama paradigmasından esinlenerek geliştirilen modeldir.
* Varlık bağıntı (VB - ER) modelindeki varlık (entity) bu modelde nesne olarak adlandırılır.
* Nesne hakkındaki bilgi, VB modelindeki niteliklere karşılık gelir.
* Varlık kümesi sınıf olarak adlandırılır.
* VB modelinden farklı olarak sınıflar üye fonksiyonlara da sahiptirler. Kisi ara, Ad listele vb.

[metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu](https://github.com/celalceken/DatabaseManagementSystems/blob/master/Sekiller/02/ModelTablosu.png)

**Yeni Veri Modelleri**

* Nesne İlişkisel Model (Object/Relational Model)
  + İlişkisel modelle nesne yönelimli modelin birleştirilmesi sonucu ortaya çıkmıştır.
* Genişletilebilir İşaretleme Dili (Extensible Markup Language, XML)
  + Çoğunlukla, farklı platformlar arası veri değişimi için kullanılan veri tanımlama standardıdır. Yapısal olmayan verileri tanımlamak için de kullanılır.
* JavaScript Nesne Gösterimi (JavaScript Object Notation, JSON)
  + Çoğunlukla, farklı platformlar arası veri değişimi için kullanılan veri tanımlama standardıdır. Yapısal olmayan verileri tanımlamak için de kullanılır.
* NoSQL
  + İlişkisel modelin yetersiz kaldığı büyük hacimli verilerin yönetimi için tercih edilir.

**Veri Soyutlama**

Veri modellerinin daha iyi anlaşılabilmesini sağlamak amacıyla ANSI-SPARC, 1970’lerin başında, veri soyutlamanın 3 düzeyini tanımlamıştır. (ANSI-SPARC: American National Standards Institute, Standards Planning and Requirements Committee.)

* Harici Model (External Model)
* Kavramsal Model (Conceptual Model)
* Dahili Model (Internal Model)

**Harici Model (External Model)**

* Veritabanının son kullanıcılar açısından görünen kısmı. Veritabanının sadece kullanıcıyla ilgili alt bölümlerini ifade eder.

[metin, ekran görüntüsü, sayı, numara, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu](https://github.com/celalceken/DatabaseManagementSystems/blob/master/Sekiller/02/HariciModel1.png)

[metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu](https://github.com/celalceken/DatabaseManagementSystems/blob/master/Sekiller/02/HariciModel2.png)

**Kavramsal Model (Conceptual Model)**

* Veritabanının veritabanı tasarımcısı açısından görünen kısmı. Veritabanının tüm alt bölümlerini birleştirerek global olarak görünmesini sağlar.
* Varlık Bağıntı Diyagramı (VBD – ERD) ile gösterilir. Kullanılan yazılım (DBMS) ve donanımdan bağımsızdır. Donanım ya da yazılım değişikliği kavramsal model tasarımını etkilemez.
* Kavramsal model mantıksal görünüş olarak da kullanılır.

[metin, ekran görüntüsü, sayı, numara, ekran, görüntüleme içeren bir resim

Açıklama otomatik olarak oluşturuldu](https://github.com/celalceken/DatabaseManagementSystems/blob/master/Sekiller/02/KavramsalModel.png)

**Dahili Model (Internal Model)**

* Veritabanının, Veritabanı Yönetim Sistemi tarafından görünen kısmı.
* Dahili model = ilişkisel model
* Donanım bağımsız, yazılım bağımlı.

[metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu](https://github.com/celalceken/DatabaseManagementSystems/blob/master/Sekiller/02/DahiliModel.png)

**Varlık Bağıntı Modeli**

Bu bölümde varlık bağıntı modeli detaylı bir şekilde anlatılmaktadır.

**Giriş**

* Varlık Bağıntı Diyagramı (VBD, Entity Relationship Diagram (ERD)) veritabanının kavramsal olarak modellenmesini sağlayan yazılım ve donanımlardan tamamen bağımsız gösterim şekilleridir.
* Farklı kullanıcılar arasında haberleşmeyi sağlamak için kullanılır.
* VB modeline bakarak veri tabanının tasarımını gerçekleştirmek daha kolaydır.
* VBD, veritabanının temel bileşenleri olan varlık, nitelik (özellik), varlıklar arası bağıntılar (relationship) ve kısıtlardan meydana gelen çizelgelerdir.
* Chen notasyonu kavramsal modellemeyi ön plana çıkartır.
* Crow's Foot notasyonu ise daha çok uygulama (implementasyon) yönelimli yaklaşımı ön plana çıkartır.
* UML notasyonu, hem kavramsal hem de uygulama (implementasyon) modelleme yöntemleri için kullanılabilir.

**İş Kurallarını Varlık Bağıntı Modeline Dönüştürme**

* Genel olarak iş kurallarındaki isimler varlık, fiiller ise varlıklar arasındaki bağıntı olma adayıdır.
* Hakkında bilgi bulunan isim ya da isim tamlamaları varlık adayı iken, bilgi bulunmayanlar varlığa ait nitelik adayıdır.
  + *Müşterinin* **ad, soyad, numara, adres** bilgileri saklanır.
  + Bir *müşteri* çok sayıda *fatura* ***üretir***.
* Bağıntılar iki yönlüdür.
  + **1 öğretim üyesi çok** sayıda (4) **ders** verebilir
  + **1 ders** sadece **1 öğretim üyesi** tarafından verilebilir.
  + **1 kişi 1 bölüme** yönetici olabilir.
  + **1 bölüm** sadece **1 kişi** tarafından yönetilebilir.
  + **1 öğrenci çok** sayıda **derse** kayıt yaptırabilir
  + **1 ders çok** sayıda **öğrenci** tarafından alınabilir.
* İsimlendirme kuralları
* Kodlama türleri
  + Linux Coding Style, Linus Torvalds
  + Hungarian Notation,
  + GNU Coding Standards
  + Java Coding Style Guide
* Birincil anahtar (primary key): Bir varlık kümesindeki herbir varlığı eşsiz bir şekilde tanımlamıza olanak tanıyan nitelik(ler).
  + Tek bir nitelik (alan) olabileceği gibi birden fazla niteliğin birleşiminden de oluşabilir.
  + urunKodu
  + dersKodu + ogrenciNo
* Birleşik nitelikler:
  + adres: cadde, şehir, ülke, posta kodu vb.
  + Detaylı sorgular için basit özellikli alanlara dönüştürülmeliler.
* Basit nitelikler
  + yaş, ad, soyad, cinsiyet vb.
* Türetilmiş nitelikler
  + Değeri diğer özellikler kullanılarak oluşturulan özellikler.

SELECT AVG(AGE(dogumTarihi)) FROM Kisiler

* + Saklanmalı mı, hesaplanmalı mı?
* Saklanması Durumunda:
  + Avantaj: Az işlemci gücü gerekir, veriye daha hızlı erişim, geçmiş bilgisi için kullanılabilir
  + Dezavantaj: Güncel değer için sürekli denetlenmelidir, fazladan yer kaplar
* Hesaplanması Durumunda:
  + Avantaj: Yer tasarrufu sağlar. Her an güncel değer olur.
  + Dezavantaj: Çok işlemci gücü gerekir, veriye daha yavaş erişim olur, sorgular daha karmaşık olur.

**Varlıklar Arası Bağıntılar**

**Bir - Çok Bağıntısı**

* **1 derslik** sadece **1 binada** bulunabilir.
* **1 binada çok** sayıda **derslik** bulunabilir.

[metin, diyagram, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu](https://github.com/celalceken/DatabaseManagementSystems/blob/master/Sekiller/02/BirCokBagintisi.png)

* Normal şartlarda tablo içerisindeki kayıt sayısı sınırlanamaz. Bunun için uygulama yazılımları ya da tetikleyiciler (trigger) kullanılabilir.
* Kayıt sayılarının gösterilmesi uygulama yazılımı geliştirilirken çok faydalıdır. (Sınıfın açılabilmesi için en az 10 kayıt, en fazla 30 kayıt gereklidir. Bir doktor bir günde en fazla 30 reçete yazabilir...)
* Sayılar iş kurallarına bakılarak belirlenir.

**Çok - Çok Bağıntısı**

* **1 öğrenci çok** sayıda **derse** kayıt yaptırabilir
* **1 ders çok** sayıda **öğrenci** tarafından alınabilir.

[metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu](https://github.com/celalceken/DatabaseManagementSystems/blob/master/Sekiller/02/CokCokBagintisi.png)

[metin, ekran görüntüsü, diyagram, kare içeren bir resim

Açıklama otomatik olarak oluşturuldu](https://github.com/celalceken/DatabaseManagementSystems/blob/master/Sekiller/02/CokCokBagintisi2.png)

**Bir - Bir Bağıntısı**

* **1 öğretim üyesi 1 bölüm** yönetebilir
* **1 bölüm 1 öğretim üyesi** tarafından yönetilebilir.

[metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu](https://github.com/celalceken/DatabaseManagementSystems/blob/master/Sekiller/02/BirBirBagintisi.png)

**Var Olma Bağımlılığı (Existence Dependency)**

* Bir site yönetim sisteminde Apartman ve Daire varlık kümeleri olsun.
* Böyle bir sistemde «**bir apartmana bağlı olmayan daire olamaz**» kuralı mevcuttur.
* Örneğin Daire varlık kümesine hiçbir apartmana ait olmayan bir dairenin kaydını yapamayız, yapmamalıyız.
* Bu örnekte, Apartman ve Daire arasında var olma bağımlılığı vardır denir.
* Bu durumda Apartman üstün varlık, Daire ise bağımlı varlıktır.

[metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu](https://github.com/celalceken/DatabaseManagementSystems/blob/master/Sekiller/02/VarOlmaBagimliligi.png)

Bir personel bilgi sisteminde Personel ve Bagimli (Personele bagimli olan kişiler. Örneğin çocuk, eş vb.) varlık kümelerini düşünelim. Böyle bir sistemde «**bir personele bağlı olmayan bagimli varlığı olamaz**» kuralı mevcuttur. Örneğin bagimli varlık kümesine hiçbir personele ait olmayan bir çocuğun kaydını yapamayız, yapmamalıyız. Bu örnekte, bagimli ve personel arasında var olma bağımlılığı vardır denir. Bu durumda personel **üstün varlık**, bagimli ise **bağımlı varlıktır**.

[metin, yazı tipi, ekran görüntüsü, diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu](https://github.com/celalceken/DatabaseManagementSystems/blob/master/Sekiller/02/VarOlmaBagimliligi2.png)

* Var olma Bağımlılığı, Tanımlama Bağıntısı

[metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu](https://github.com/celalceken/DatabaseManagementSystems/blob/master/Sekiller/02/VarolmaBagimliligiTanimlamaBagintisi.png)

**Zayıf/Güçlü Bağıntılar**

* Bağıntı kurulan varlığın birincil anahtarı içerisinde, bağıntı kuran varlığın birincil anahtar bilgisi yer almıyorsa “iki varlık arasında zayıf bağıntı vardır” denir.
* Bağıntı kurulan varlığın birincil anahtarı içerisinde, bağıntı kuran varlığın birincil anahtarı yer alıyorsa “iki varlık arasında güçlü bağıntı vardır” denir.

[metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu](https://github.com/celalceken/DatabaseManagementSystems/blob/master/Sekiller/02/ZayifGucluBagintilar.png)

**Bağıntı Dereceleri**

* Tekli (Unary) Bağıntı
* İkili (Binary) Bağıntı
* Üçlü (Ternary) Bağıntı

**Tekli Bağıntı**

* **Tekli (Unary) Bağıntı:** Bir varlık kendisi ile bağıntılı (ilişkili) ise bu tür bir bağıntıya tekli bağıntı adı verilir.
* Örneğin, Personel tablosu içerisindeki bir personel, sıfır veya daha fazla personelin aynı zamanda yöneticisidir. Bir personelin sıfır ya da bir yöneticisi olmalıdır.

[metin, ekran görüntüsü, ay, daire içeren bir resim

Açıklama otomatik olarak oluşturuldu](https://github.com/celalceken/DatabaseManagementSystems/blob/master/Sekiller/02/TekliBaginti.png)

**İkili Bağıntı**

* **İkili (Binary) Bağıntı:** İki varlığın bağıntısına (ilişkisine) ikili bağıntı denir.

[metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu](https://github.com/celalceken/DatabaseManagementSystems/blob/master/Sekiller/02/IkiliBaginti.png)

**Üçlü Bağıntı**

* **Üçlü (Ternary) Bağıntı:** Aynı anda 3 varlık birbirine bağlanıyorsa, bu tür bağıntıya üçlü bağıntı adı verilir.
* Kavramsal tasarımda her ne kadar 3 varlık mevcut ise de bunu gerçekleştirebilmek için 4. bir varlığa gereksinim duyulur.

[ekran görüntüsü, çizgi, kare, dikdörtgen içeren bir resim

Açıklama otomatik olarak oluşturuldu](https://github.com/celalceken/DatabaseManagementSystems/blob/master/Sekiller/02/UcluBaginti.png)

**Varlık Bağıntı Modeli (VBM/ERM) ile Tasarım**

* Veritabanı tasarımı, lineer ya da ardışıl (yazılım geliştirmedeki şelale modeli gibi) olmaktan çok tekrarlı (yazılım geliştirmedeki spiral model ya da iteratif model gibi) bir süreçtir.
* Tekrar fiili, “tekrar tekrar yap” anlamındadır. Tekrarlı bir süreç, süreçlerin ve prosedürlerin tekrarlanması temeline dayanır.
* Bir varlık bağıntı diyagramının geliştirilmesi genellikle aşağıdaki adımları içermektedir.
  + Organizasyonun (kurumun) işlerinin (operasyonlarının) tanımını içeren detaylı bir senaryo (hikaye) oluşturulur. Senaryo özellikle organizasyon içerisindeki rol temsilcilerine danışılarak oluşturulursa çok daha gerçekçi ve etkili olur.
  + Senaryoda geçen işlerin tanımları baz alınarak iş kuralları oluşturulur.
  + İş kuralları baz alınarak ana varlıklar ve varlıklar arasındaki bağıntılar oluşturulur.
  + İlk varlık bağıntı diyagramı geliştirilir.
  + Varlıkları net bir şekilde tanımlayan nitelikleri ve birincil anahtarları oluşturulur.
  + Varlık Bağıntı Diyagramı gözden geçirilerek gerekirse yukarıdaki adımlar, istenilen duruma gelininceye kadar tekrarlanır.