

VERİTABANI SİSTEMLERİ

- **1960'lı yılların başında** Charles Bachman tarafından **IDS** (Integrated Data Store-Bütünleştirilmiş Veri Depolama) adıyla **ilk genel amaçlı veritabanı yönetim sistemi** geliştirilmiştir.
- **1960'ların sonunda** ise **IBM** tarafından **IMS** (Information Management System-Bilgi Yönetim Sistemi) adıyla **ilk ticari veritabanı yönetim sistemi** geliştirilmiş ve bu yapı hiyerarşik veri modeline temel teşkil etmiştir.
- **İlişkisel veritabanı modeli**, **1969'da Edgar F. Codd** tarafından geliştirildi.
- **Veri:** Ham gözlemler, işlenmemiş gerçekler ya da izlenimlerdir. Birbirleriyle ilişkilendirilip yorumlanmadıkları sürece tek başlarına bir anlam ifade etmezler ve bu halleriyle karar verme konusunda da karar vericilere bir katkı sağlayamazlar.
- **Bilgi:** Verinin işlenmiş ve karar vermeye destek olacak duruma getirilmiş halidir. Söz konusu işleme ve dönüştürme süreci; veri üzerinde kaydetme, sınıflama, sıralama, hesaplama, özetleme, çoğaltma, analiz ve raporlama işlemlerinin uygulanması ile gerçekleştirilir. Bu işlemler sonucunda veri anlam kazanarak bilgiye dönüşmüş olur.
- **Veritabanı (Database):** Herhangi bir konuda birbiriyle ilişkili olan ve amaca uygun olarak düzenlenmiş, mantıksal ve fiziksel olarak tanımlanmış veriler bütünüdür.
- **Veritabanı Yönetim Sistemi (Database Management System):** Veritabanı tanımlamak, oluşturmak, veritabanında işlem yapmak, veritabanının farklı kullanıcı yetkilerini belirlemek, veritabanının bakımını ve yedeklemesini yapmak için geliştirilmiş programlardır.
- **Sıralı Erişim:** İstenilen veriye ulaşılncaya kadar ilgili dosyadaki tüm verilerin sırayla okunması gerekir. Geçmişte kullanılan müzik kasetleri bu tür bir erişime örnektir.
- **Doğrudan Erişim:** Verilerin yer aldığı fiziksel adresler indeks numarası ile tanımlanıp bu indeks numaraları da ayrı bir dosya olarak saklanır. Bir veriye erişilmek istendiğinde, önce indeks dosyasından verinin yer aldığı adresi gösteren indeks numarası bulunur, daha sonra bu numaraya karşılık gelen fiziksel adrese doğrudan erişim sağlanır. Müzik CD'lerindeki erişim, bu erişim türüne örnektir.
- VTYS'ler, sağladıkları birçok avantaja rağmen malîyet açısından geleneksel dosya sistemlerine göre **dezavantajlıdır**. Ayrıca basit ve iyi tanımlanmış veritabanı uygulamaları söz konusu olduğunda geleneksel dosya sistemlerini kullanmak daha avantajlıdır.
- **Veritabanı Kullanıcıları:** Veritabanı ile herhangi bir şekilde etkileşimde olan kişi ya da kişilerdir.
 - **Veritabanı Sorumluları:** Veritabanının tasarlanması, oluşturulması, işletim faaliyetlerinden birinci derecede sorumlu olan ve en fazla yetkiye sahip olan kullanıcılarıdır.
 - **Veritabanı Yöneticisi:** Veritabanına erişim yetkilerini belirleme, kullanımın düzenlenmesi ve izlenmesini sağlama, ihtiyaç duyulan yazılım ve donanım kaynaklarını edinme sorumluluğu vardır.
 - **Veritabanı Tasarımcısı:** Veritabanında saklanacak olan verilerin tanımlanmasından ve bu verilerin depolanması ve gösterilmesi için gerekli olan uygun yapıların seçilmesinden sorumludur.
 - **Son Kullanıcılar:** Veritabanına sorgulama, güncelleme yapmak veya rapor üretmek için erişen kullanıcılarıdır.
 - **Standart:** Veritabanına nadiren erişim yapan, her seferinde farklı bilgi ihtiyacı olabilen kullanıcılarıdır.
 - **Sıradan ya da Parametrik:** Bu kullanıcıların işleri, veritabanı üzerinde sürekli bir sorgulama ve güncelleme yapmalarını gerektirir. (Rezervasyon yapan acentalar)
 - **Gelişmiş:** Ayrıntılı olarak belirledikleri karmaşık gereksinimlerini karşılamak amacıyla veritabanını kullanan gruptur. (Mühendisler, Bilim adamları vb.)
 - **Bağımsız:** Bu kullanıcılar menü kullanımı ya da araç çubukları gibi grafiksel öğeler yardımıyla kullanım kolaylığı sağlayan hazır paket programlarını kullanarak kişisel veritabanlarının sürekliliğini sağlar.
 - **Sistem Analistleri ve Uygulama Programcıları:** **Sistem analisti;** özellikle sıradan son kullanıcıların gereksinimlerini belirleyen ve standart işlemler yoluyla bu gereksinimleri karşılayabilecek ayrıntıları belirleyen kişi ya da kişilerdir. **Uygulama programcıları** ise; sistem analisti tarafından belirlenen ayrıntıları program hâline getiren ve daha sonra test eden, hataları ayıklayan, belgeleyen ve kaydedilmiş işlemler olarak sürekliliğini sağlayan kişilerdir.
- **Veri soyutlama (data abstraction):** Verilerin düzenlenmesi ve depolanmasına ilişkin ayrıntıların gizlenmesi ve verinin daha iyi anlaşılmasını sağlamak için veriye ilişkin temel özelliklerin vurgulanması anlamına gelir.
- **Veri modeli (data model):** Bir veritabanının mantıksal yapısını tanımlamada kullanılacak kavramlar, işlemler ve kurallar bütünüdür. Temel veri modeli işlemleri; veritabanı üzerinde ekleme, silme, değiştirme, veriyi geri çağırma gibi genel işlemleri içerir.
- **Fiziksel veri modelleri:** Kayıt biçimi, sırası ve erişim yolu bilgileri ile dosya olarak verilerin bilgisayarda nasıl saklandığını belirler.
- **Üç Şema Mimarisi:** **1) İçsel (fiziksel) düzey:** Veritabanının fiziksel depolama yapısını tanımlayan içsel şemayı içerir. **2) Kavramsal düzey:** Fiziksel depolama yapısının ayrıntılarını gizler ve veritabanında yer alan verilerin tipine, veriler arası ilişkilere, kullanıcı işlemlerine ve kısıtlara ilişkin tanımlara yoğunlaşır. **3) Dışsal (görünüm) düzey:** Her dışsal şema bir grup kullanıcının ilgilendiği bazı veritabanı bölümlerini tanımlar. Böylece veritabanının diğer kısmı bu kullanıcı grubundan gizlenir.
- **Veri bağımsızlığı:** Herhangi bir düzeydeki şema değiştiğinde bir üst düzeydeki şemanın değişmeden kalmasını bununla birlikte iki düzey arasındaki eşleştirmelerin değişmesini sağlar. Bu eşleştirme sonucunda bir üst düzeydeki şemaya başvuran uygulama programları değişiklik ihtiyacı göstermez.
- **Mantıksal veri bağımsızlığı** (logical data independence): Kavramsal şemanın dışsal şemalarda ya da uygulama programlarında değişiklik yapılmaksızın değiştirilebilmesi anlamına gelir. **Fiziksel veri bağımsızlığı** (physical data independence): Kavramsal şemada bir değişiklik yapılmaksızın içsel şemada değişiklik yapma kapasitesidir.
- **Veritabanı Türleri:**
 - **1) Hiyerarşik veritabanı:** Kayıtlar, ilişkileri temsil eden ve ağaca benzeyen dallar biçiminde hiyerarşik bir yapıda oluşturulur.
 - **2) Ağ veritabanı:** Her bağlantı noktası düğüm olarak ifade edilirse, hiyerarşik yapıdan farklı olarak, her düğümün birden fazla ebeveyn ve birden fazla çocuk düğümü ile bağlantısı olabilir.
 - **3) İlişkisel veritabanı:** Bu yapıda ilk iki veri modelinden farklı olarak birden çok ilişki biçimi kullanılabilir. Günümüzde kullanılan veritabanı yönetim sistemlerinin hemen hemen hepsinde tercih edilen model ilişkisel veri modelidir.
 - **4) Nesneye yönelik veritabanı:** Yalnızca harf, rakam ya da çeşitli karakterler kullanılarak yapılandırılmış verileri değil, aynı zamanda multimedya (çeşitli çizim, fotoğraf, görüntü, ses ya da video gibi nesneleri) de içeren veritabanıdır.

- **Veritabanı Yönetim Sistemi Yazılımları:** MS SQL, Oracle, MySQL, Sybase, PostgreSQL, MS Access, DB2.
- **Veritabanı Tasarım Aşamaları:** Üç aşamadan oluşmaktadır.
 - **1) Kavramsal Veri Modeli:** Veritabanında saklanacak verilerin, kısıtlarının ve ilişkilerinin bir gösterimidir.
 - **2) Mantıksal Veri Modeli:** Kavramsal modelin veritabanı şemasına dönüştürülmesidir. Veri depolama, veri kısıtları ve ilişkiler tarafından tamamiyle bağlanmış veri modelidir. Bir mantıksal veri modeli insanları, yerleri, nesneleri, kuralları ve ilişkiler ile onlar arasındaki olayları standartlaştırır.
 - **3) Fiziksel Veri Modeli:** Veri süreklilik teknolojisinin belirli bir sürümü ile bağımlı, tamamiyle bağlanmış veri modelidir. Veritabanının nasıl inşa edileceğini belirtir. Sütun adı, sütun veri türü, sütun kısıtlamaları, birincil anahtar, yabancı anahtar ve tablolar arasındaki ilişkiler de dâhil olmak üzere tüm tablo yapılarını göstermektedir.
- **Varlık ilişki modelleme:** Bir işletme veya iş alanında kullanılan verilerin detaylı ve mantıksal gösterimidir. Gerçek hayattaki varlıklar, aralarındaki ilişkiler ve varlıklar ile ilişkilerin özelliklerini içerir.
- **Varlık ilişki diyagramı (ER-diagram):** Varlık ilişki modelinin grafiksel gösterimidir.
- **Varlık ilişki modelinde veri yapısı** grafiksel olarak 3 şekilde ifade edilir:
 - **1) Varlıklar (Entity):** İşletmenin verisini tutmak istediği kullanıcı ortamındaki kişi, yer, nesne, olay veya kavramı temsil eder.
 - **Varlık kümesi (Entity type):** Ortak özellikleri paylaşan varlıkların koleksiyonudur. Dikdörtgen ile gösterilir.
 - **Varlık örneği (Entity instance):** Varlık kümesinin özelliklerini taşıyan bir varlığın oluşudur.
 - **Güçlü varlık kümesi:** Diğer varlık kümelerinden bağımsız tanımlanabilen kümelerdir.
 - **Zayıf varlık kümesi:** Ancak bir güçlü varlık kümesine bağlı olarak var olabilir. Çift çizgili dikdörtgen ile gösterilir.
 - **2) Öznitelikler (Attribute):** Veri modelinde tanımlanan varlık veya ilişkilerin bir özellik ya da niteliğidir. Oval ile gösterilir ve içine adı yazılır. Bazı varlık kümeleri ve bu kümelerin sahip olduğu öznitelikler:
 - **Öğrenci:** Öğrenci_no, Öğrenci_adı, Ev_adresi, Telefon, Doğum_tarihi
 - **Araç:** Şasi_No, Renk, Ağırlık, Motor_gücü
 - **Zorunlu ve Seçimli Öznitelikler:** Varlık kümesi içindeki her bir örnek için değer girilmesi zorunlu olan özniteliklere zorunlu, zorunlu olmayan özniteliklere seçimli öznitelikler denir.
 - **Basit ve Birleşik Öznitelikler:** Basit öznitelik anlamlı daha küçük parçalara bölünemez. Ancak bazı öznitelikler daha detay bilgileri içeren anlamlı alt parçalara bölünebilir. Bunlara birleşik öznitelikler (çok parçalı) denir. Adres birleşik özniteliği; mahalle, cadde, kapı no, postaKodu ve şehir olarak anlamlı alt özniteliklere bölünebilir.
 - **Türetilen Öznitelik:** Değeri ilgili olduğu diğer özniteliklerden hesaplanabilen özniteliktir. Varlık ilişki diyagramında kesik çizgili oval şekilde gösterilir.
 - **Tek ve Çok Değerli Öznitelikler:** Bir varlığın özniteliği sadece tek değer alıyorsa buna tek değerli öznitelik denir. Birden çok değer alıyorsa buna çok değerli öznitelik denir. Örneğin Çalışan tablosunda çalışanların telefon bilgilerini saklamak için kullanılan Telefon özniteliğinde çalışana ait birden fazla telefon bilgisi tutulursa çok değerli öznitelik olur. Birleşik öznitelik ile karıştırılmamalıdır. Çok değerli öznitelikte bir varlık örneğine ait birden fazla veri tutulur. Birleşik öznitelikte ise bir varlığa ait bir değer tutulur.
 - **Anahtar Öznitelik:** Varlık kümesindeki her bir örnek için farklı değer alan öznitelige denir. İki örneğin değeri aynı olamaz! Tek bir öznitelikten oluşabileceği gibi birden fazla öznitelğin birleşimi ile de tanımlanabilir. Bu tip öznitelige birleşik anahtar denir.
 - **Etki Alanı:** Bir öznitelğin alabileceği değerlerin tümünün oluşturduğu kümeye denir. Etki alanı varlık ilişki diyagramı içinde gösterilmez.
 - **3) İlişkiler:** En az iki varlık kümesi arasındaki etkileşimi gösteren bağlantıya denir. İki varlık kümesi arasında olabileceği gibi çoklu şekilde de olabilir. Varlık ilişki diyagramında baklava dilimi şekliyle gösterilir.
 - **İlişkisel Varlıklar:** Bir ilişkinin özniteliği birden fazla ise bu ilişki bir varlık olarak gösterilebilir.
 - **İlişki Derecesi:** İlişkide yer alan varlık kümesinin sayısı ilişkinin derecesini gösterir.
 - **İlişki Türleri:** Varlık kümeleri arasında üç tür ilişki vardır:
 - **Bire Bir İlişki:** Bir varlık kümesinin elemanı, diğer bir kümenin elemanlarından sadece biri ile ilişki kurabilir.
 - **Bire Çok İlişki veya Çoka Bir İlişki:** Bir varlık kümesinin elemanı, diğer bir varlık kümesinin elemanlarıyla birden çok ilişki kurabilir.
 - **Çoka Çok İlişki:** Bir varlık kümesinin birden fazla elemanı, diğer bir varlık kümesinin elemanlarıyla birden fazla ilişki kurabilir.
- **Varlık İlişki Diyagramlarının Tablolara Dönüştürülmesi:** Varlık ilişki modelleri tablolara dönüştürülürken varlık kümeleri tablolara, öznitelikler ise alanlara (tablo sütunları) dönüştürülür. Anahtar öznitelikler ise tabloda birincil anahtara dönüştürülür.
- **Yabancı anahtar:** İlişkisel veri tabanlarında bir tablonun birincil anahtarının diğer bir tabloya daha kolay bağlanmak amacıyla ikinci tablo üzerinde bir sütun olarak yer almasıdır.
- **Veritabanı Yaşam Döngüsü Fazları:**
 - **1) Gereksinim Analizi:** Tüm veri gereksinimleri ve veri işleme süreçleri ve benzerlerinin analizi yapılır.
 - **2) Veritabanı Tasarımı:** Bu fazda kavramsal, mantıksal ve fiziksel veri modelleri oluşturulur.
 - **3) Uygulama ve Yükleme:** Veritabanı uygulama ekiplerince geliştirilir ve çalışacağı hedef sisteme yüklenir.
 - **4) Operasyon:** Bu fazda bilgi sistemi ve geliştirilen veritabanı uygulamaya alınmış ve kullanımdadır.
 - **5) Bakım:** Sistem kullanıma alındıktan sonra kullanıcılar, yeni isteklerin gerçekleştirilmesi ve iyileştirmeler talep edebilir.
- **İlişkisel Cebir:** Aşağıdaki ilişkisel cebir türlerini A ve B şeklindeki iki ayrı tabloya uyguladığımızı düşüneceğiz.
 - **Birleşim (U):** Bu işlem A ve B tablolarına uygulanırsa (AUB), tüm kayıtlar bir araya getirilir, tekrar eden kayıtlar elenir.
 - **Kesişim (∩):** İki tablonun sadece ortak olan kayıtlarını döndürür. (A∩B)
 - **Fark (-):** İki tablodan birincisinde olan, ancak ikincisinde olmayan kayıtları döndürür. (A - B)
 - **Yansıtma (π):** Bir tablonun sadece seçilen alanlarından oluşan yeni bir tablo oluşturur. $\pi_{\text{İsim},\text{Adres}}(\mathbf{A})$, A tablosunun sadece İsim ve Adres sütunlarından oluşan yeni bir tabloyu ifade eder.

- **Seçim (σ):** Verilen belirli bir şarta göre, bir tablonun kayıtlarının alt kümesi olarak, yeni bir tablo üretir. $\sigma_{\text{Cinsiyet=F}}(A)$, A tablosundaki verilerde, Cinsiyet alanı F olan verileri yeni bir tablo olarak döndürür.
- **Kartezyen Çarpım (\times):** İki tablonun alanlarının tüm olası eşleşmelerini getirir. ($A \times B$)
- **Şartlı Bitişme (\bowtie):** Sonuç şeması kartezyen çarpımdaki gibidir. İki tablonun kartezyen çarpım sonucundan verilen şarta uygun olanlar getirilir. $A \bowtie_{A.yas < B.yas} B$, ifadesi A ve B tablolarındaki yaş alanlarından hareketle, $A \times B$ kartezyen çarpımının sonuçlarında, B'nin A'dan büyük olan yaş alanlarını döndürür. Bu ifade aynı zamanda, $\sigma_{A.yas < B.yas}(A \times B)$ ifadesine eşittir.
- **Doğal Bitişme (\bowtie):** Sonuç, iki tablonun eşit bitişmede tüm ortak alanlarının kullanılmasıyla bulunur. İki tablo arasında en az bir alan ortak ise uygulanabilir. ($A \bowtie B$)
- **Bölme (\div):** $M(x,y)$ ve $N(y)$ iki tablo olmak üzere, $M \div N$ işlemi alan değeri y'ye eşit olan M tablosu içindeki (x) alan değerlerini verir. **M isim** ve **ayakkabı** alanları içeren bir tablo olsun. **N** de sadece **ayakkabı** alanı içeren bir tablo olsun. $M \div N$ işlemi, M tablosundaki hangi ismin, tüm ayakkabı markalarından aldığı bilgisini döndürür.
- **Veri Tipleri üçe ayrılır:** Basit, Karmaşık ve Özelleştirilmiş.
- **Basit Veri Tipleri:** Bu veri tiplerinde tek bir değer üzerinde bir örüntü veya değer kısıtlaması yapılır.
 - **Karakter:** Bu veri tipi sabit veya değişken boydaki karakter dizilerinde karakterleri depolamakta kullanılır.
 - Sabit boyutlu karakter veri tipi **CHARACTER** ya da **CHAR**;
 - Değişken boydaki karakter veri tipi **CHARACTER VARYING**, **CHAR VARYING** veya **VARCHAR**;
 - Sabit boydaki ve yabancı dildeki karakterleri depolamak için **NATIONAL CHARACTER**, **NATIONAL CHAR** ve **NCHAR**;
 - Değişken boydaki ve yabancı dildeki karakter dizilerini depolamak için ise **NATIONAL CHARACTER VARYING**, **NATIONAL CHAR VARYING** ve **NCHAR VARYING** kullanılır.
 - **Bit:** Bu veri tipi ikili sayı sistemindeki sayılardan (binary number) oluşan dizileri depolamakta kullanılır. Bu veri tipi **BIT**, **BIT VARYING**, **BINARY** veya **VARBINARY** olarak kullanılır.
 - **Tam Sayısal:** Tüm sayıları ve ondalık sayıları depolar. **NUMERIC**, **DECIMAL (DEC)**, **INTEGER (INT)** ve **SMALLINT** kullanır.
 - **Yaklaşık Sayısal:** Ondalık sayılar ve üslü sayılar depolanır. Veri tipi olarak; **FLOAT**, **REAL** ve **DOUBLE PRECISION** kullanılır.
 - **Tarih ve Zaman:** Bu veri tipi genelde **TIMESTAMP** olarak bilinir. Tarih ve saat bilgisini depolamakta kullanılır.
- **Karmaşık Veri Tipleri:** Nesne veri tiplerini kapsar. Nesnelerin kullanımı ve ilişkisel veritabanları arasında bir köprü görevi görür.
 - **İkili (binary) nesneler:** İkili nesneler olağan ilişkisel veritabanı kayıt yapılarından ikili veriyi ayırmak için oluşturulmuştur. Resim, ses benzeri ikili veriler çok büyük olabileceğinden, normal veritabanı kayıtlarına sığmayacak ve veritabanının fiziksel veri depolama özelliğini zorlayarak verimsiz kullanıma sebep olacaktır. Bu nedenle ikili nesneler geleneksel tablo kayıt değerlerinden farklı olarak oluşturulmuştur. Büyük karakter dizileri, video, XML veri, ses vb. ikili nesneler olarak tutulur.
 - **Referans işaretçileri (pointers):** Bazı ilişkisel veritabanları veritabanı dışına depolanmış nesne veya dosyaları işaretlemek (point) için bu veri tipini kullanır.
 - **Koleksiyon diziler:** Bu veri tipinde tablo alanında dizi tutulur. Koleksiyon diziler sabit veya dinamik uzunlukta olabilir.
 - **Kullanıcı tanımlı:** Bazı İVTYS yazılımları kullanıcının kendi tanımladığı veri tiplerini kullanmasına izin verir.
- **Özelleştirilmiş Veri Tipleri:** Bu veri tipleri daha gelişmiş ilişkili veritabanı sistemlerinde görünür.
 - **XML:** Hiyerarşik düzende verilerin saklanması sağlayan bir veri saklama sistematiğidir.
 - **Geography:** Coğrafi verilerin saklanması için özelleştirilmiş veri türleridir.
 - **Geometry:** Geometrik şekillerin standart bir tanımı yapılarak yeniden çizilmesine yönelik verilerin saklanması sağlar.
 - **Hierarchyid:** Birbirleri ile ilişkili nesneleri hiyerarşik bir ağaç yapısında saklayan veri türüdür. Bu yapı sadece veri değil verilerin birbiri olan ilişkilerini saklayan bir veri türüdür.
- **Null Değerinin Kullanımı:** NULL bilinmeyen veya olmayan değeri gösterir. NULL sayısal değerlerdeki sıfır ya da karakter dizilerindeki boş karakterlere karşılık gelmez. NULL değeri içeren bir işlemin sonucu yine NULL olur.
Örneğin; $(25 \times 3) + 4 = 79$ iken; $(\text{Null} \times 3) + 4 = \text{Null}$, $(25 \times \text{Null}) + 4 = \text{Null}$, $(25 \times 3) + \text{Null} = \text{Null}$ değerini alır.
- **Kısıtlar (Constraint):** Kısıtlar tablodaki alanlar üzerine kuralların uygulanmasını mümkün kılar.
 - **NOT NULL Kısıtı:** Eğer bir alanın özelliği NOT NULL olarak belirlenmiş ise tabloya yeni bir kayıt eklenirken bu alana ait değerin boş bırakılmasına izin verilmez.
 - **DEFAULT Kısıtı:** Tabloya bir kayıt eklendiği veya değiştirildiği zaman DEFAULT kısıtlı alana değer belirtilmemiş ise ilgili alana varsayılan bir değer atanmasını sağlar.
 - **Anahtar Kısıtı:** Anahtar kısıtları farklı tablolardaki alanlar arasındaki değerlerin kontrol ve doğrulanmasına izin verir.
 - **Birincil Anahtar (Primary Key):** Tablo içindeki bir kaydı benzersiz olarak belirlemek için kullanılır. NULL değeri alamaz. Tabloda toplam bir adet Birincil anahtar tanımlanabilir. Fakat birden fazla alana ortak bir şekilde birincil anahtar tanımlanabilir. Birden fazla alan ile oluşturulan birincil anahtara **kompozit birincil anahtar** denir.
 - **Benzersiz Anahtar (Unique Key):** Bu anahtar bir alanın değerlerinin benzersizliğini belirler. Bir tabloda birden fazla benzersiz anahtar olabilir ve NULL değeri atanabilir.
 - **Yabancı Anahtar (Foreign Key):** Yabancı anahtarlar ana tablodaki birincil anahtarın alt tablodaki kopyasıdır. Yabancı anahtar her iki tablonun uygun bir şekilde ilişkilendirilmesini ve veri tutarlılığının korunmasına yardımcı olurlar.
 - **CHECK Kısıtı:** Bu kısıt ile bir veya daha fazla alana girilebilecek değerlerin sınırlandırılması sağlanır. Örneğin, bir alana girilebilecek sayısal değer, CHECK kısıtı ile 0-100 arasında tanımlanırsa, o alana 100+ değerler, mesela 101 girilemez.
- **Görünümler (Views):** Veritabanından bir veya daha fazla tablonun alanlarından oluşturulan sanal tablodur. Görünümler veritabanındaki bilgileri farklı yönlerden görmenizi sağlar.
- **İndeksler (Indexes):** İndeks bir tablonun küçük bir kısmının (tablonun bir alanı gibi) kopyasıdır. İndeksler bir kitabın başındaki içindekiler bölümü gibi davranır. Bir kitabın indeks bölümünden ilgili sayfa no bulunup konuya erişilmesi gibi, aynı şekilde tablolardaki indeks ile de istenilen değer indeksten bulunur ve tablodaki kayda hızlı olarak erişilir.
- **Saklı Yordamlar (Stored Procedures):** Veri üzerinde ön işlem yaparak veritabanı dışına gönderilecek verinin miktarını düşürmekte kullanılır. Ayrıca güvenliği arttırmak için sistemler tasarlanırken kullanıcıların sadece saklı yordamlara erişimi açılır ve tablolara erişim izni verilmez.
- Masaüstü veritabanı yönetim sistemi yazılımlarında hiç program kodu kullanmadan veritabanı hazırlamak mümkündür.

- **MS Access**, ilk masaüstü ilişkisel veritabanı yönetim sistemidir ve oluşturulan dosya uzantısını **.accdb** olarak kaydeder.
- **MS Access Veri Tipleri:**
 - **Ek:** Dijital fotoğraflar gibi dosyalar.
 - **Otomatik Sayı:** Her kayıt için otomatik olarak üretilen numaralar.
 - **Para Birimi:** Para değerleri.
 - **Tarih/Saat:** Tarihler ve saatler.
 - **Köprü:** E-posta adresleri, web sayfaları gibi bağlantılar.
 - **Not:** Uzun metin blokları ve metin biçimlendirilmesinin kullanıldığı metinler.
 - **Sayı:** Sayısal değerler. Para birimleri için ayrı bir veri türü olduğunu unutmayın.
 - **OLE Nesnesi:** Word belgeleri gibi OLE (Object Link and Embedding–Bağlı ve gömülü nesneler) nesnesi.
 - **Metin:** Ad-Soyad ya da posta adresi gibi kısa alfasayısal değerler.
 - **Evet/Hayır:** 0 ya da 1 değeri olarak saklanan boolean değeri.
- MS Access'te **tablo isimleri en fazla 64 karakter** uzunluğu ile sınırlıdır.
- **MS Access'te Formlar:** Veritabanında depolanan tablolara veri eklemek, düzenlemek ya da görüntülemek amacı ile kullanılırlar.
- **MS Access'te Raporlar:** Veritabanında yer alan verilerin, yazıcıdan çıktı alınabilecek bir görünümde düzenlenmiş hâlidir. Genellikle çıktısı alınabilecek şekilde biçimlendirilir, ekranda görüntülenebilir, başka bir programa verilebilir ya da e-posta iletilisi olarak gönderilebilir.
- **MS Access'te Sorgular:** Tablolarda depolanmış verileri özel bir görünümle sunan nesnelerdir. Sorgu sonuçları genellikle formlara ve raporlara yönelik kayıt kaynağı olarak hizmet verir.
- **MS Access'te Sorgu Türleri:** 1) Seçme Sorgusu, 2) Eylem Sorgusu, 3) Çapraz Sorgu, 4) Parametre Sorgusu, 5) SQL Sorgu.
- **Seçme sorgusu**, veriyi bulup alır ve kullanıma hazır hâle getirir. **Eylem sorgusu**, verilerle ilgili bir görev gerçekleştirir. Eylem sorguları yeni tablolar oluşturmak, var olan tablolara veri eklemek, verileri güncelleştirmek veya silmek amacıyla kullanılabilir.
- **MS Access'te Eylem Sorguları:** 1) Tablo oluşturma sorgusu, 2) Ekleme sorgusu, 3) Güncelleştirme sorgusu, 4) Silme sorgusu.

----- Vize buraya kadar -----

- **Veri Tanımlama Dilinde (DDL);** tabloların oluşturulması, silinmesi ve bazı temel özelliklerinin düzenlenmesini sağlamak üzere sırası ile **CREATE**, **DROP** ve **ALTER** komutları kullanılır. Aşağıdaki tüm işlemler MSSQL diline göre anlatılmıştır.
- **Veri İşleme Dilinde (DML);** **SELECT**, **INSERT**, **DELETE** ve **UPDATE** komutları kullanılır.
- **Veri Kontrol Dilinde (DCL);** kullanıcıya yetki tanımlama için **GRANT**, kullanıcı yetkilerini engellemek için **DENY** ve daha önce yapılmış olan yetki ve izinleri kaldırmak için **REVOKE** komutu kullanılır.
- **Şema Oluşturma:** Tablolar, görünüm, alanlar ve yetkilendirmeler şemayı tanımlamaktadır. SQL'de ayrıca katalog kavramı da bulunmaktadır. Katalog ise belli şemaların bir araya gelmesi ile oluşturulmaktadır.
- **İndeks Oluşturma:** Eğer işletmelerde belli tip sorgular daha yaygın kullanılmakta ve sorguların cevaplarında gecikmeler olmakta ise yapılacak indeks tanımlamaları ile hızlandırma mümkün olabilir.
- **Görünüm Oluşturma:** Bazı veriler belirli kullanıcıların ortak erişimine açık iken bazı verilerin ise tüm kullanıcılara açık olmaması gerekir. Farklı kullanıcıların erişimi için **CREATE VIEW** ile sanal veri kümeleri oluşturulmaktadır.
- **Veritabanı Oluşturma:** **CREATE DATABASE** Dbismi
- **Veritabanının İsmi Değiştirme:** **ALTER DATABASE** Dbismi **MODIFY NAME=**YeniIsim;
- **Veritabanı Silme:** **DROP DATABASE** YeniIsim
- **Sayısal Veri Tipleri:**
 - **Tam Sayılar:** **tinyint** (1 bayt), **smallint** (2 bayt), **int** (4 bayt), **bigint** (8 bayt)
 - **Ondalıklı Sayılar:** **float** (4 bayt), **real** (4 bayt) veya basamak sayıları tanımlanabilen **decimal** ve **numeric**.
 - **İkili Sayılar (Binary):** Sabit uzunlukta değerler için **binary**, değişken uzunlukta değerler için **varbinary**.
- **Metin-Karakter Veri Tipleri:** ASCII karakter seti için, sabit uzunluklu veri kümesi saklayan **char**, değişken uzunluklu veri kümesi saklayan **varchar** veri tipleri vardır. Unicode karakter seti için, sabit uzunluklu veri kümesi saklayan **nchar**, değişken uzunluklu veri kümesi saklayan **nvarchar** veri tipi vardır.
- **Tarih ve Zaman Veri Tipleri:** **sene-ay-gün** olarak saklamak için **date**, Zamanı **saat:dakika:saniye** olarak saklamak için **time**, her ikisini beraber saklayabilen **datetime**, **smalldatetime** vb. veri tipleri bulunur.
- **Öznitelik Kısıtları:** Eğer bir öznitelik boş değer alabiliyorsa, **NULL** olarak atanabilir. Eğer boş olmaması gerekiyorsa, kesinlikle doldurulması için **NOT NULL** olarak atanmalıdır. Buna rağmen boş bırakılırsa, değer NULL olarak atanır. Eğer bir öznitelik için herhangi bir değer girilmemişse, varsayılan bir değer atanabilir. Bunun için **DEFAULT** <değer> kullanılır. Özniteliklerde aralık tanımlamak için **CHECK** komutu kullanılır. Belirli bir sayıdan başlayıp, belirlediğimiz aralığa göre artan veya azalan bir şekilde sayısal değer üretilmesi isteniyorsa, **IDENTITY** komutu kullanılır.
- **Anahtar Kısıtı:** Birincil anahtarlar, ilgili tabloda benzersiz değer alırlar. Diğer bir deyişle aynı değerın farklı satırda yer almasını garanti altına alırlar. **PRIMARY KEY** komutu ile tanımlanabilir. Birden fazla alanın, beraber anahtar olması durumunda **UNIQUE** komutu da kullanılabilir.
- **Bütünlük (integrity) Kısıtı:** Bu kısıt, tüm veritabanı tabloları arasındaki özniteliklerin birbirleri ile olan ilişkisinin bütünlüğünün sağlanması için önemlidir. **FOREIGN KEY** komutu kullanılır.
- MSSQL **tablo**, **alan** ve **benzeri kullanıcı tanımlı isimleri** verirken **köşeli parantez ([])** ya da **çift tırnak (" ")** kullanılabilir. Özellikle iki kelimeden ya da Türkçe karakterlerden oluşan ifadeler tanımlanırken adların bu semboller içine alınması derleyicinin ifadeleri doğru yorumlamasını sağlar.
- **Tablolarda değişiklik yapılması:** Bir tablonun adı, öznitelikleri veya alanlar, kısıtlar vb. tüm tablo özellikleri değiştirilebilir veya ek özellikler eklenebilir. Bunun için **ALTER** komutu kullanılır. Tablo üzerinde ALTER ve/veya DROP işlemi yaparken diğer tablolardan bu tabloya yabancı anahtar ile bağlı bir alan varsa işlem gerçekleşmeyecektir.
- **Tabloyu silme işlemi:** **DROP TABLE** Tabloismi
- Veritabanlarında **indeks** veriyi erişim hızını arttıran yapılardır.

- **Tablolarda İndeks İşlemleri:** İndeks oluşturma için **CREATE INDEX**, oluşturulan indeksin silinmesi için **DROP INDEX** komutu kullanılır.
- **Görünüm Oluşturma ve Silme İşlemleri:** Bu yapılar veriyi değil veriyi elde edecek sorgu komutlarını saklarlar.
CREATE VIEW Görünüm_Adı **AS** <SQL Sorgusu>
SQL sorgusu başına **BEGIN**, sonuna **END** komutu koyularak veya sadece sonuna **GO** komutu koyularak çalıştırılabilir.
Görünüm silmek için **DROP VIEW** Görünüm_Adı; komutu kullanılır.
- **Saklı Yordam Oluşturma ve Silme İşlemleri:** Sunucu üzerinde tutulan belirli bir görevi yerine getirmek için birden fazla tablo üzerinde işlem yapabilen, program içinden farklı parametreler ile çağrılarak kullanılabilen SQL tabanlı komut kümesidir. Görünümler sadece bir SQL sorgusunu çalıştırırken saklı yordamlarda T-SQL programlama dili kullanılabilir. Yordam oluşturma işlemi aşağıdaki gibi yapılır.
CREATE PROCEDURE Procedureismi
AS BEGIN
 <SQL Sorgusu>
END
Yordam silme işlemi aşağıdaki gibi yapılır.
DROP PROCEDURE Procedureismi;
- SQL dilinin, diğer programlama dillerinden farklı olarak, işlemleri değil sonuçları tarif eden bir yapısı vardır.
- **SELECT** en fazla kullanılan komuttur. Depolanan verilerin, kullanıcıların ihtiyacına cevap verecek şekilde elde edilmesini sağlar.
- **Genel SELECT komutu kullanımı:** SELECT ve diğer örnekler için, **Northwind** tablosunu indirip, kullanabiliriz. (<https://goo.gl/K839eI>)
SELECT alan_listesi **INTO** yeni_tablo **FROM** tablo_kaynağı **WHERE** seçme_koşulları **GROUP BY** gruptama ifadeleri **HAVING** Gruplanan_alan_kısıtlamaları **ORDER BY** sıralama düzeni **ASC DESC**
 - **FROM** ile başlayan kısım seçme işleminin hangi veri kümesinden yapılacağını belirtir.
 - **WHERE** seçme işleminde filtreleme işlevi görür.
 - **GROUP BY** verileri özetleme ya da gruptama işlemleri için kullanılır.
 - **HAVING** Gruplanan ya da hesaplanan alanların sınırlandırılması için kullanılır. WHERE ile karıştırılmamalıdır.
 - **ORDER BY** sorgunun en sonunda yer alan sıralama işlemidir.
- **SELECT * FROM Tablo_Adı;** sorgusu Tablo_adi tablosundaki tüm alan ve satırları listeler.
- **SELECT ifadesinin mantıksal işleme sırası:**
 - FROM > ON > JOIN > WHERE > GROUP BY > WITH CUBE or WITH ROLLUP > HAVING > SELECT > DISTINCT > ORDER BY > TOP
- **SELECT ile yeni bir tablo oluşturmak:** Aşağıdaki sorgu, Çalışanlar tablosunun üç kolonunu kullanarak yeni bir Eposta tablosu oluşturur.
SELECT Ad, Soyadı, [E-posta Adresi] **INTO** Eposta **FROM** [Çalışanlar]
- **SELECT ile benzersiz değerler** elde etmek: Aşağıdaki sorgu, kaç farklı [İş Unvanı] olduğu ve bu unvanların listesini getirir.
SELECT DISTINCT [İş Unvanı] **FROM** [Çalışanlar]
- **SELECT ile sorgu sonuçlarının sıralanması:** Sonuçlar [Şehir] kolonundaki verilerde Z'den A'ya doğru sıralanır.
SELECT [Şehir], [Soyadı], Ad **FROM** [Çalışanlar] **ORDER BY** [Şehir] **DESC**
- **SELECT TOP ile en üstte yer alan sonuçların getirilmesi:** Aşağıda liste fiyatı en fazla olan üç ürünü getirme kodu yer alır.
SELECT TOP (3) [Ürün Adı], [Liste Fiyatı] **FROM** [Ürünler] **ORDER BY** [Liste Fiyatı] **DESC**
- **WHERE ile sonuçları sınırlandırma:** [Ürünler] tablosundan üç kolon seçip, [Ürün Adı]'nın 'Ceviz'e eşit olduğu satırları getiren kod.
SELECT [Ürün Kodu], [Ürün Adı], [Liste Fiyatı] **FROM** Ürünler **WHERE** ([Ürün Adı] = 'Ceviz')
- **MSSQL'de kullanılan diğer koşul ifadeleri tablosu:** Bazı ifadelerin yanında köşeli parantez içerisindeki [NOT] ifadesi ilgili koşulun değilini ifade etmek için kullanılır.

Operatör	Tanımı	Where sonrası yazım şekli
=	Eşitir	[Ürün Adı] = 'Ceviz'
<>	Eşit Değildir (!=)	[Kategori] <> 'Çerezler'
>	Büyük	[Liste Fiyatı]>50
<	Küçük	[Liste Fiyatı]<50
>=	Büyük veya eşit	[Ürün Adı] >= 'H'
<=	Küçük veya eşit	[Liste Fiyatı]<=10
[NOT] BETWEEN	Belirtilen değerler arasında. (Sınır değerleri listeye dahil edilir)	[Liste Fiyatı] BETWEEN 10 AND 20
[NOT] LIKE	Metin içerisindeki desene göre kısıtlama	[Ürün Adı] LIKE 'C%' (C ile başlayan ürünler)
[NOT] IN	Bir alanın birden fazla değer ile sınırlandırılması	[Ürün Adı] IN ('Ceviz', 'Hint Çayı')
EXISTS	Alt sorgular ile mevcut sorgunun sınırlandırılması sağlanır	EXISTS (Select * From Where)
IS [NOT] NULL	Boş satırların bulunması	[Liste FİYATI] IS NOT NULL
CONTAINS	Metin içerisinde karakter bazlı aranacak ifadeler	CONTAINS (ŞikayetMetni, 'Son Kullanım')
FREE TEXT	Metin içerisinde doğal dil özelliklerine göre ifade arama	FREETEXT (Belge, 'go fast')

- Listede yer alan **CONTAINS** ve **FREETEXT** ifadeleri uzun metin verisi içeren alanlarda ilgili metin ifadelerinin bulunması için kullanılır. Arama yapılacak alanların daha önce **Full-Text Indeks** olarak tanımlanması gerekmektedir.
- [Ürünler] tablosunda [Liste fiyatı] 40-60 arasında olan ve [Ürün Adı] içerisinde "A" harfi olan ürünleri getiren kod:
SELECT [Ürün Kodu], [Ürün Adı], [Liste Fiyatı] **FROM** [Ürünler] **WHERE** ([Liste Fiyatı] **BETWEEN** 20 **AND** 40) **AND** ([Ürün Adı] **LIKE** '%A%')
- LIKE** komutu ile kullanılacak ifadeler:
 - %** : Bir karakter dizisi ya da hiçbir metin için kullanılır. 'A%' ifadesi "A" ile başlayan ya da sadece 'A' içeren alanları sınırlandırabilir.
 - _** : Herhangi tek bir karakter. '_eri' ifadesi ilk harfi farklı olabilecek "Geri", "Seri" gibi ifadeleri bulabilmektedir.
 - []** : Köşeli parantez aralığında tanımlanan harfler ile sınırlandırılmaktadır. '[a-f]' veya '[abcdef]' şeklinde yazılabilir. '[C-T]apa' ifadesi ile "Çapa", "Sapa", "Tapa" gibi metinler sınırlandırılabilir.
 - [^]** : Belirtilen aralıkta karakter içermeyen metinler bulunur. 'Araba[^1]%' ifadesi içerisinde 1 olamayan ancak 'Araba' ile başlayan metinleri bulabilecektir.
- AND** birbirine bağlanan ifadelerin her ikisinin de gerçekleşmesini, **OR** ifadesi ise koşullardan en az birinin gerçekleşmesini ifade eder.
- Birden fazla tablodan veri seçmek**: Yabancı anahtar ile birbirine bağlı iki tablodan veri çekmek için ilgili anahtarların WHERE koşulunda bir araya getirilmesi gerekir. Aşağıdaki SQL kodu, Müşteri ile Siparişler tablosundaki ortak alanlardan hareketle iki tabloya da bağlanıp bilgi getirmektedir. Müşteriler tablosundaki "No" alanı ile Siparişler tablosundaki [Müşteri No] alanı birbirine eşittir.
SELECT TOP (3) Müşteriler.Şirket, Müşteriler.Soyadı, Müşteriler.Ad, Müşteriler.[No], Siparişler.[Müşteri No]
FROM Siparişler, Müşteriler **WHERE** Siparişler.[Müşteri No] = Müşteriler.[No]
- AS** komutu, tablo isimlerini kısaltmak için kullanılabilen bir komuttur. Üstteki kod AS ile aşağıdaki gibi kısa şekilde yazılabilir.
SELECT TOP (3) M.Şirket, M.Soyadı, M.Ad, M.[No], S.[Müşteri No]
FROM Siparişler **AS** S, Müşteriler **AS** M **WHERE** S.[Müşteri No] = M.[No]
- JOIN** Komutu ile tablolara bağlanmak: JOIN komutu ile sadece iki değil çok sayıda tabloya birden bağlanılabilir. JOIN komutu ile bağlanmada **INNER**, **OUTER** ve **CROSS** olmak üzere farklı seçenekler vardır. Tüm JOIN örneklerini aşağıdaki iki tablo üzerinde gösterelim.

A Tablosu		B Tablosu	
A_Id	Ad	B_Id	Name
1	Bir	1	One
2	İki	2	Two
4	Dört	3	Three
8	Sekiz	7	Seven

- INNER JOIN** ile her iki tablonun da eşleşen alanlarının seçilmesi sağlanır. Aşağıdaki koda göre, A tablosundaki A_Id'lerden, B tablosundaki B_Id'ler ile eşit olan tüm satırlar sonuç olarak döndürülür.

Inner Join				
<pre>SELECT A.A_Id, A.Ad, B.B_Id, B.Name FROM A INNER JOIN B ON A.A_Id=B.B_Id</pre>	Sonuç			
	A_Id	Ad	B_Id	Name
	1	Bir	1	One
	2	İki	2	Two

- LEFT OUTER JOIN**, solda bulunan tablonun (örneğimizde A tablosu) tüm verisinin sonuçta yer alması sağlar. Aynı SQL kodunu LEFT OUTER JOIN ile uyguladığımızda, A tablosundaki tüm A_Id değerlerinin, B tablosundaki B_Id kolonunda olup olmadığına bakılır ve bulunamayan eşleşmeler NULL olarak atanır.

Left Outer Join				
<pre>SELECT A.A_Id, A.Ad, B.B_Id, B.Name FROM A LEFT OUTER JOIN B ON A.A_Id=B.B_Id</pre>	Sonuç			
	A_Id	Ad	B_Id	Name
	1	Bir	1	One
	2	İki	2	Two
	4	Dört	NULL	NULL
	8	Sekiz	NULL	NULL

- RIGHT OUTER JOIN** ise sağdaki tabloyu ölçü alır ve tüm verisinin döndürülen sonuçta yer almasını sağlar. B tablosundaki tüm B_Id değerlerinin, A tablosundaki A_Id kolonunda olup olmadığına bakılır ve bulunamayan eşleşmeler NULL olarak atanır.

Right Outer Join				
<pre>SELECT A.A_Id, A.Ad, B.B_Id, B.Name FROM A RIGHT OUTER JOIN B ON A.A_Id=B.B_Id</pre>	Sonuç			
	A_Id	Ad	B_Id	Name
	1	Bir	1	One
	2	İki	2	Two
	NULL	NULL	3	Three
	NULL	NULL	7	Seven


- **FULL OUTER JOIN** komutu, LEFT OUTER JOIN ve RIGHT OUTER JOIN komutlarının birleşimini döndürür. Tekrar eden değerler, sonuç içerisinde bir kez yer alır.

Full Outer Join				
<pre>SELECT A.A_Id, A.Ad, B.B_Id, B.Name FROM A FULL OUTER JOIN B ON A.A_Id=B.B_Id</pre>		Sonuç		
		A_Id	Ad	B_Id
		1	Bir	1
		2	İki	2
		4	Dört	NULL
		8	Sekiz	NULL
		NULL	NULL	3
		NULL	NULL	7
				Name
				One
				Two
				NULL
				NULL
				Three
				Seven

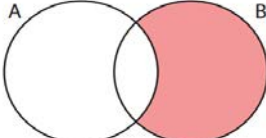
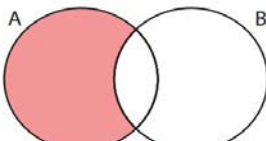
- **CROSS JOIN** sorgusunda, A tablosundaki her satır için B tablosunun tüm satırları getirilir. Böylece her iki tablonun satır sayılarının çarpımı kadar toplam sonuç satırı elde edilir.

Cross Join				
<pre>SELECT A.A_Id, A.Ad, B.B_Id, B.Name FROM A CROSS JOIN B</pre>		A_Id	Ad	B_Id
		1	Bir	1
		2	İki	1
		4	Dört	1
		8	Sekiz	1
		1	Bir	2
		2	İki	2
		4	Dört	2
		8	Sekiz	2
		1	Bir	3
		2	İki	3
		4	Dört	3
		8	Sekiz	3
		1	Bir	7
		2	İki	7
		4	Dört	7
		8	Sekiz	7
				Name
				One
				Two
				Two
				Two
				Three
				Three
				Three
				Seven
				Seven
				Seven
				Seven

- **Veri Kümelerinin Birleştirilmesi:** İki veya daha fazla veri kümesinin birleştirilmesi için veri kümelerinin aynı sayıda ve türde alanlarının olması (yani aynı tipte, farklı değerlere sahip tablolar) gerekir. Örneğimizdeki A ile B tabloları, farklı değerler içerir ama öznitelikleri bakımından aynı tiptir. Veri kümeleri **INTERSECT**, **EXCEPT**, **UNION**, **UNION ALL** komutları ile birleştirilebilir.
- **INTERSECT:** İki veya daha çok veri kümesindeki satırların kesişiminin bulunmasında kullanılır. Sonuç veri kümesinin alan adının ilk sorgudaki alan adı olduğu görülmektedir.

<pre>SELECT A_Id FROM A INTERSECT SELECT B_Id FROM B</pre>	<table><tr><th>A_Id</th></tr><tr><td>1</td></tr><tr><td>2</td></tr></table>	A_Id	1	2	 <p>A Venn diagram with two overlapping circles labeled A and B. The intersection of the two circles is shaded in red, representing the set intersection.</p>
A_Id					
1					
2					

- **EXCEPT:** Veri kümeleri arasındaki farkı listelemek için kullanılır. Birinci şekil B-A, diğeri de A-B'dir.

<pre>SELECT B_Id FROM B EXCEPT SELECT A_Id FROM A</pre>	<table><tr><th>B_Id</th></tr><tr><td>3</td></tr><tr><td>7</td></tr></table>	B_Id	3	7	
B_Id					
3					
7					
<pre>SELECT A_Id FROM A EXCEPT SELECT B_Id FROM B</pre>	<table><tr><th>A_Id</th></tr><tr><td>4</td></tr><tr><td>8</td></tr></table>	A_Id	4	8	
A_Id					
4					
8					

- **UNION ve UNION ALL:** Veri kümelerinin birleşim kümelerini elde etmek için kullanılır. UNION, tekrar eden verileri bir kez içerirken; UNION ALL tüm tekrar eden verileri de içerir.

<pre>SELECT A_Id FROM A UNION SELECT B_Id FROM B</pre>	<table><tr><th>A_Id</th></tr><tr><td>1</td></tr><tr><td>2</td></tr><tr><td>3</td></tr><tr><td>4</td></tr><tr><td>7</td></tr><tr><td>8</td></tr></table>	A_Id	1	2	3	4	7	8	<p>A Venn diagram with two overlapping circles labeled A and B. Both circles are filled with a light red color. The intersection of the two circles is also shaded red.</p>		
A_Id											
1											
2											
3											
4											
7											
8											
<pre>SELECT A_Id FROM A UNION ALL SELECT B_Id FROM B</pre>	<table><tr><th>A_Id</th></tr><tr><td>1</td></tr><tr><td>2</td></tr><tr><td>4</td></tr><tr><td>8</td></tr><tr><td>1</td></tr><tr><td>2</td></tr><tr><td>3</td></tr><tr><td>7</td></tr></table>	A_Id	1	2	4	8	1	2	3	7	<p>A Venn diagram with two separate circles labeled A and B. Both circles are filled with a light red color. The circles do not overlap.</p>
A_Id											
1											
2											
4											
8											
1											
2											
3											
7											

- Veritabanı tablolarında özetleme ve gruplama işlemlerinde **GROUP BY** komutu kullanılır. GROUP BY komutu bir veri kümesinde belirlenen alanların içerdiği verinin tekrarsız hâle getirerek özetlenmesidir. Altaki SQL sorgusu çalıştırıldığında 47 adet siparişe ait, o siparişi alan çalışanların numaraları tekrarlı bir şekilde döndürülür.

```
SELECT [Çalışan No] FROM [Siparişler]
```

Bu çalışanlar listesini tekrarsız olarak edinebilmek için, GROUP BY kullanabiliriz. Böylece tekrarsız 8 satır sonuç döndürülür.

```
SELECT [Çalışan No] FROM [Siparişler] GROUP BY [Çalışan No]
```

- **Özetleme işlevleri** gruplama işlemlerinde sayma, toplama, ortalama ve benzeri işlemleri yerine getirirler. Özetleme işlevleri tablosu.

İşlev	Açıklaması
COUNT, COUNT_BIG	Bir gruptaki kayıt sayısını hesaplarlar. COUNT Int (tamsayı) veri tipi ile veri döndürürken, COUNT_BIG BigInt(büyük tamsayı) veri tipinde sonuç döndürür.
SUM	Sayısal veriler için kullanılır. NULL olan verileri dikkate almaz. Distinct komutu ile benzersiz değerlerin toplanması sağlanabilir.
AVG	Sayısal değerli alanların ortalamasını hesaplar.
MIN	Belirtilen alanın en küçük değerini getirir. NULL değerleri dikkate alınmaz. Metin değerler için de alfabetik sıraya göre en küçüğünü getirir.
MAX	Belirtilen alanın en büyük değerini getirir. NULL değerleri dikkate alınmaz. Metin değerler için de alfabetik sıraya göre en küçüğünü getirir.
STDEV	Belirtilen alan için standart sapma hesaplar.
STDEVP	Belirtilen alanın tüm değerleri için ana kütle standart sapmasını hesaplar.
VAR	Belirtilen alan için varyans hesaplar.
VARP	Belirtilen alanın tüm değerleri için ana kütle varyansını hesaplar.

- Bu sorgu, [Sipariş Ayrıntıları] tablosundaki toplam ürün sayısını verir. Northwind veritabanına göre bu sayı 56 çıkacaktır.

```
SELECT COUNT([Ürün No]) FROM [Sipariş Ayrıntıları]
```
- Bu sorgu ise, [Sipariş Ayrıntıları] tablosunda kaç farklı ürün olduğu bilgisini verir. Bunun sonucu da 24 çıkacaktır.

```
SELECT COUNT(DISTINCT [Ürün No]) FROM [Sipariş Ayrıntıları]
```
- **Cast ve Convert** işlevleri görüntülen verilerin, veri türü değiştirmek için kullanılır. Kalıcı olarak değiştirmek için veri ALTER komutu kullanılır. Veri türünün dönüşümü için kullanılan Cast komutunun yazım kuralı **Cast(AlanAdı AS Veritürü)** şeklindedir.
- **HAVING:** Gruplama sorgularında özetlenen değerlerin belirli koşullara göre sınırlanması için kullanılır.
- **MSSQL Mantıksal İşlevler:**
 - **CHOOSE:** İndeks değeri içeren bir değerın tanımlanan sıralı listede eşleştirerek karşı gelen elemanı görüntüler. Altaki sorguda sonuçlar görüntülenirken, Durum adında bir kolon oluşur. Durum, aslında Siparişler tablosunda yoktur. Siparişler tablosundan alınan [Durum No] değerlerine göre, CHOOSE() işlevi ile oluşturulur.

```
SELECT [Durum No], CHOOSE([Durum No]+1, 'Yeni', 'Faturalandı', 'Sevk Edildi', 'Kapatıldı') AS Durum FROM Siparişler
```
 - **IIF:** Mantıksal bir karşılaştırma sonucu doğru ve yanlış durumları için iki ayrı değer görüntüler. Bu mantıksal işlev **IIF(Mantıksal_ifade, Doğru_değer, Yanlış_Değer)** şeklinde oluşturulur.
 - **CASE-END:** Birden çok koşulun mantıksal sorgulanabilmesine imkân tanır. Programlamadaki Switch-Case yapısı gibi çalışır.

```
CASE ifade
WHEN Değer1 THEN Değer1 WHEN Değer2 THEN Değer2
ELSE 'diğer'
END
```


- MSSQL Sayısal İşlevler:

İşlev	Açıklaması	Örnek
ABS	Belirtilen değerin mutlak değerini veren matematik işlevidir.	ABS (-5) Sonuç : 5
SIGN	Sayısal değerin pozitif, negatif veya işaretsiz olduğunu döndürür	SIGN(-550) Sonuç : -1
ASIN, ACOS, ATAN, ATN2	Belirtilen trigonometrik değerlerin radyan cinsinden açılış değerini hesaplar.	4*ATAN(1) Sonuç : 3,14
SIN, COS, TAN, COT	Radyan cinsinden belirtilen değerlerin trigonometrik değerlerini hesaplar.	TAN(3.14/4) Sonuç : 1
RADIANS	Derece cinsinden verilen açılış değerini radyan cinsine çevirir.	RADIANS(180.0) Sonuç : 3.14
DEGREES	Radyan cinsinden verilen açılış değerini derece cinsine çevirir.	DEGREES(PI()) Sonuç : 180
PI	Pi (π) katsayısını verir.	PI() Sonuç : 3,14159265358979
CEILING	Kesirli sayıları bir üst tamsayıya yuvarlar.	CEILING(-5.5) Sonuç : -5
FLOOR	Kesirli sayıları bir üst tamsayıya yuvarlar.	FLOOR(-5.5) Sonuç : -6
ROUND	Belirtilen sayı ya da alandaki değeri istenilen basamağa yuvarlar. İlk değeri yuvarlanacak değeri, ikinci değeri yuvarlanacak basamak.	ROUND(56.55, -1) Sonuç : 60.00
EXP, LOG, LOG10, POWER	Üssel ve logaritmik işlem yapan işlevlerdir.	POWER(2,5) Sonuç : 32
RAND	0 ile 1 aralığında düzgün dağılmış rassal sayı üretir.	RAND() Sonuç : 0,5299
SQRT	Belirtilen değerin karekökünü hesaplar.	SQRT(25) Sonuç : 5
SQUARE	Belirtilen değerin karesini hesaplar.	SQRT(25) Sonuç : 625

- MSSQL Metinsel İşlevler:

İşlev	Açıklaması	Örnek
SUBSTRING	Metin türündeki bir değerin içerisinde belirli bir bölümün alınmasını sağlayan işlevdir.	SUBSTRING('YBS206U',4,3) Sonuç : 206
PATINDEX	Bir metin içerisinde belirli bir metin ifadesinin arayarak ilk bulunduğu konum bilgisini döndürür.	PATINDEX('%YBS%',Belge) Sonuç : 50
LEN	Metindeki karakter sayısını döndürür	LEN('YBS206U') Sonuç : 7
LEFT, RIGHT	Metin türündeki bir alan ya da değerin soldan(LEFT) veya sağdan(RIGHT) istenilen sayıda karakterin seçilmesini sağlar.	LEFT('YBS206U',3) Sonuç : 'YBS'
REPLACE	Bir metin içerisinde belirli bir metni bularak o istenilen metin ile değiştirir.	REPLACE('YBS206U','206','304') Sonuç : 'YBS304U'
CONCAT	Bir veya daha fazla metin değeri birleştirir.	CONCAT('Veri','tabanı','I') Sonuç : 'Veritabanı I'
REPLICATE	Bir metnin değişkenin istenildiği kadar tekrarlanması için kullanılır.	REPLICATE('0',5) Sonuç : '00000'
SPACE	İstenilen sayıda boşluk karakteri üretir.	'A'+SPACE(3)+'B' Sonuç : 'A A '
NCHAR, CHAR	NCHAR Unicode karakter tablosunun, CHAR ise ASCII kod tablosunun belirtilen rakama karşı gelen karakterini getirir.	NCHAR(65) Sonuç : 'A'
UNICODE, ASCII	UNICODE bir karakterin Unicode, ASCII ise ASCII kod tablosundaki kod karşılığını getirir.	ASCII('A') Sonuç : 65
STR	Sayısal bir değeri istenilen hassasiyette metne çevirir.	STR(123.45, 6, 1); Sonuç : 123.5

- **Alt sorgu (Sub Query):** bir seçme sorgusunun bir parçasını oluşturan ve parantez içinde yazılmış sql seçme sorgusuna denilir.
- **EXISTS** ile yapılan sınırlamalarda alt sorgunun tüm alanlarının "*" sembolü ile seçildiğine dikkat edilmelidir. "*" ile tüm alanları seçilmeyen bir alt sorgu EXISTS ile kullanılamaz.
- **Veri İşlemleri Dili (DML, Data Manipulation Language):** Veritabanı tablosuna kayıt ekleme, silme ve güncelleme işlemlerinde kullanılır.
- Veritabanı tablolarına bir veya daha fazla satır (kayıt) eklemek için **INSERT INTO** komutu kullanılır. Herhangi bir kayıt satırında ilgili sütuna hiç bir veri girilmez ise alana **NULL** değer atanır.
- **INSERT INTO ile tabloya doğrudan veri ekleme:** VALUES(...) içindeki metinsel değerler tek tırnakla eklenir.
INSERT INTO TabloAdı [(alan1, alan2, ...)] **VALUES** (değer1, değer2, ...)
- **INSERT INTO ile bir sorgu sonucunun tabloya eklenmesi:**
INSERT INTO TabloAdı [(alan1, alan2, ...)] **SELECT** değer1, değer2,.. **FROM** Tablo
- **Veri Ekleme ve Düzenleme İşlevlerinde Kullanılabilecek İşlevler:**

İşlev	Açıklaması	Kullanım Şekli
CAST	Sorgu esnasında veri türünün dönüştürülmesini sağlar. (Veritabanı yapısını değiştirmez)	Cast ([Tutar] as nvarchar (20)) + 'TL dir.' Sonuç: 20.00TL dir.
CONVERT	Cast ile aynı işlevi gerçekleştirir.	Convert (nvarchar (20), [Tutar]) + 'TL dir.' Sonuç: 20.00TL dir.
PARSE	Bir ifadedeki bilgili ilgili dile göre metin içerisinde çıkarır.	Parse ('19 Haziran 1982' AS datetime USING 'Tr-TR') Sonuç: 1982-06-19 00:00:00 dir.
LOWER	Metinleri küçük harfe çevirir.	Lower ('KÜÇÜK HARF') Sonuç: küçük harf
UPPER	Metinleri büyük harfe çevirir.	Upper ('büyük harf') Sonuç: BÜYÜK HARF
ROUND	Ondalık bir sayının istenen basamağa kadar yuvarlanmasını sağlar	ROUND (123.4545, 2); Sonuç: 123.4500
GETDATE	SQL Server yazılımının çalıştığı bilgisayarın sistem tarihini döndürür.	GetDate () Sonuç: 2017-06-01 11:00:12.98
DAY, MONTH, YEAR	Tarih veri türündeki değişkenlerin sırasıyla gün, ay ve yıl verisine dönüştüren işlevlerdir.	Day ('2018/5/17') Sonuç: 17 Month (FaturaTarihi) Sonuç: 05 Year (GetDate()) Sonuç: 2017
LTRIM, RTRIM	Karakter türündeki değişkenlerin başındaki (LTRIM) ya da sonundaki (RTRIM) boşluk karakterlerini kaldırır.	LTRIM (' TÜRKİYE') Sonuç: 'TÜRKİYE'

- **DELETE ile tablodan veri silme:**
DELETE FROM TabloAdı **WHERE** Koşul
- Aşağıdaki kod, Müşteriler tablosunda No alanındaki değeri 29 olan satırı tablodan siler:
DELETE FROM [Müşteriler] **WHERE** No=29
- **WHERE** ifadesi olmadan **DELETE** komut çalıştırılırsa, tablodaki tüm veriler silinir: **DELETE** [Müşteriler];
- Tüm kayıtları silmek için **TRUNCATE** komutu da kullanılabilir: **TRUNCATE TABLE** TabloAdı;
- İşlem günlüğüne yazılmadığından dolayı **TRUNCATE** komutu ile yapılan silme işlemi geri alınamaz. Bu yüzden tüm kayıtları silmek için TRUNCATE komutu, DELETE komutundan daha hızlı çalışır.
- Silme işlemi yapılırken tablolar arasında bulunan ilişkilerin bozulmamasına dikkat edilmesi gerekir. Birden fazla tablo birbirine ilişkisel olarak bağlıysa, bir tabloda yapacağınız silme işlemi diğer tablodaki düzeni de değiştirebilir. Örneğin aşağıdaki kod çalışırsa, hata meydana gelir. Çünkü [Siparişler] tablosunda 28. müşteriye ait veriler bulunmaktadır. 28. müşteri [Müşteriler] tablosundan silinirse, [Siparişler] tablosundaki o müşteriye ait verilerin hiçbir anlamı kalmaz. Bu yüzden aşağıdaki kod hata üretir.
DELETE FROM Müşteriler **WHERE** [No]=28
Bu hata veri bütünlüğünün korunması için çok önemlidir. O zaman silme işlemi yapılırken, o silme işlemi ile alakalı diğer tablolardaki verilerin de otomatik olarak silinmesi için **CASCADE DELETE** yapısı kullanılmalıdır. Bu yapı, veritabanının ilk oluşturulduğu anda tabloya **ON DELETE CASCADE** şeklinde bir özellik olarak verilebilir. Bu yapıldığında, yukarıdaki kod, hata üretmeyecek ve siz Müşteriler tablosundan 28. müşteriyi sildiğinizde, ilişkili tüm tablolardaki veriler de otomatik olarak silinmiş olacaktır.
- **UPDATE ile tablodaki verileri düzenleme:** Genel kullanımı aşağıdaki gibidir.
UPDATE TabloAdı **SET** alan1 = değer1, alan2 = değer2, ... **WHERE** Koşul
Aşağıdaki kod, Müşteriler tablosunda No'su 25 olan müşterinin Ev Telefonu bilgisini kodda verilen telefon numarası ile değiştirir. **WHERE** komutu ile koşul ifadesi yazılmazsa, tüm müşterilerin numarası kodda verilen numara olacak şekilde değiştirilir.
UPDATE [Müşteriler] **SET** [Ev Telefonu] ='(222) 222 33 44' **WHERE** No=25
- **DELETE** komutu için **ON DELETE CASCADE** kullanılabildiği gibi, **UPDATE** komutu için aynı mantıkla **ON UPDATE CASCADE** kullanılabilir.
- Bir veya birden fazla SQL ifadesi arka arkaya tek bir işlem gibi çalıştırılmak istenildiği zaman **TRANSACTION** yapısı kullanılır.

- **TRANSACTION** ile SQL komutlarının ya tamamı gerçekleştirilir veya hiçbiri gerçekleştirilmez. TRANSACTION işlemleri tek bir işlem olarak ele alacağı için herhangi birisi gerçekleşmediği zaman diğer gerçekleşen işlemleri de yok sayacaktır. Yani gerçekleşen işlemi geri alacaktır (**ROLLBACK**). Eğer işlemlerin tamamı sorunsuz bir şekilde gerçekleşirse, tüm işlemleri kalıcı (**COMMIT**) hâle gelecektir.
- **Fonksiyonel Bağımlılık:** Bir tabloda iki öznelik A ve B olsun. Eğer A özneliğinin değeri B özneliğinin değerini belirliyorsa B özneliğinin A özneliğine bağımlı olduğu söylenir. B'nin A'ya fonksiyonel bağımlılığı ok işareti ile **A→B** şeklinde gösterilir.
- **Birleşik Anahtar:** Bir tabloda birincil anahtar bir veya daha fazla öznelikten oluşabilir. Eğer birincil anahtar iki veya daha fazla öznelikten oluşuyorsa bu tür birincil anahtarlara birleşik anahtar (composite key) adı verilir.
- **Kısmi Bağımlılık (Partial Dependence):** Bir tabloda; A, B, C ve D gibi toplam dört öznelik olsun ve bu tablonun birleşik anahtarı da (A, B) olsun. Bu durumda **AB→CD** fonksiyonel bağımlılığı yazılabilir. Bu tabloda **A→C** fonksiyonel bağımlılığını da varsayalım. Bu durumda C özneliği birleşik anahtarın sadece bir kısmı olan A özneliğine bağımlıdır. Yani C özneliği A'ya kısmi bağımlıdır denir.
- **Geçişli Bağımlılık (Transitive Dependence):** Bir tabloda; A, B, C ve D gibi toplam dört öznelik olsun ve bu tabloda A birincil anahtar olsun. **A→B** ve **B→C** fonksiyonel bağımlılıklarının olduğunu varsayalım. C'nin B'ye bağımlı ve B'nin de A'ya bağımlı olduğu görülür. Bu durumda, C aynı zamanda A'ya da bağımlıdır. Bu durum geçişli bağımlılık olup **A→C** fonksiyonel bağımlılığı yazılabilir.
- **Tam Fonksiyonel Bağımlılık (Full Functional Dependence):** Bir özneliğin değeri bağılı olduğu anahtar ile benzersiz olarak belirleniyorsa bu ilişki tam bağımlılık olarak adlandırılır.
- **Çok Değerli Bağımlılık (Multiple Valued Dependence):** Tabloda bir alandaki değerler virgülle ayrılarak oluşturulan liste veya dizi değerlerinden oluşuyorsa çok değerli bağımlılık vardır.
- **Döngüsel Bağımlılık (Cyclic Dependency):** Döngüsel bağımlılıkta döngü kapalı halka, tekrar etme anlamında kullanılmaktadır. Veritabanında bir öznelik **A→B** iken aynı zamanda **B→A** ise döngüsel bağımlılık vardır.
- **Aykırlık (Anomali):** İlişkisel veritabanında bir satırdaki verinin (veya satırlardaki verilerin) hatalı olarak değiştirilmesine denir.
- Bir veritabanında görülebilecek üç aykırılık vardır. Bunlar **ekleme**, **silme** ve **güncelleme** aykırılıklarıdır. Eğer bir tabloda bu aykırılıklardan biri veya birkaçı varsa veritabanının iyi tasarlanmadığı söylenir. Bu nedenle veritabanı bu aykırılıklar giderilecek şekilde iyileştirilmelidir.
- **Ekleme Aykırılığı:** Birincil ve yabancı anahtar ile ilişkilendirilmiş iki tablo olsun. Birinci tabloda kayıtların özet bilgileri, ikincisinde ise ilgili kayıtların detay bilgileri var olsun. İkinci tabloya veri eklenirken, birinci tabloya ilgili veri eklenmez ise ekleme aykırılığı oluşur.
- **Silme Aykırılığı:** Yine üstteki mantıkla, birinci tablodaki veri silinip, ona bağılı olan ikinci tablodaki veriler silinmezse, silme aykırılığı oluşur. Bunu engellemenin yolu ON DELETE CASCADE kullanmaktır. Böylece ilişkili diğer tablolardaki tüm veriler de silinmiş olur.
- **Güncelleme Aykırılığı:** Silme aykırılığında olduğu gibi, bir tablodaki değişiklik diğer ilişkili tablolara da yansıtılmazsa, aykırılık oluşur. Bu aykırılığı gidermek için de yine ON UPDATE CASCADE komutu kullanılabilir.
- **Normalleştirme:** Tablolar ve aralarındaki ilişkilerin aykırılıkları azaltmak üzere aşamalı olarak daha küçük ve iyi yapılandırılmış tablolar ve ilişkilere dönüştürülmesidir.
- Normalleştirmenin her aşamasında uygulanan kurala **Normal Form (NF)** denir. İlk kural uygulanıyorsa veritabanı tasarımının birinci normal formda (**1NF**) olduğu, ikinci kural uygulanıyorsa ikinci normal formda (**2NF**) olduğu şeklinde ifade edilir.
- **Normalleştirmenin Amaçları:** 1) Veri artıklığını (redundancy) minimum yapmak. 2) Veri bütünlüğünü sağlayan kısıtların uygulamasını basitleştirmek. 3) Veri işlemeyi (ekleme, güncelleme ve silme) daha basit hâle getirmek. 4) Gerçek varlık ve ilişkileri daha iyi temsil edecek veritabanı modeli tasarımını gerçekleştirmek.
- **Normalleştirme Aşamaları:**
 - **1. Normal Form (1st Normal Form - 1NF):** Çok değerli alanlar tek değerli hâle dönüştürülür. Her bir alandaki değer atomik olur. Birden fazla bilgi tek bir sütunda tutulmaz. Her tabloda birincil anahtar tanımlanır. Tablolar ilişkisel yapıya dönüştürülür. Yeni tablo oluşturulmaz.
 - **2. Normal Form (2nd Normal Form - 2NF):** Kısmi bağımlılıklar kaldırılır.
 - **3. Normal Form (3th Normal Form - 3NF):** Geçişli fonksiyonel bağımlılıklar kaldırılır. Geçiş tabloları oluşturulabilir.
 - **Boyce-Codd Normal Form (BCNF):** Birincil anahtar dışındaki diğer alanlarda bire çok ilişkisi olanlar kaldırılır. Fonksiyonel bağımlılıktan geri kalan tüm aykırılıklar kaldırılır. Bir tablo 3NF'de ise ve her belirleyici anahtar olarak tanımlandıysa ilgili tablo BCNF formunda denir.
 - **4. Normal Form (4th Normal Form - 4NF):** Çok değerli bağımlılıklar kaldırılır. 4NF için tüm bağımsız bire çok ilişki için ayrı tablo oluşturulur.
 - **5. Normal Form (5th Normal Form - 5NF):** Kalan aykırılıklar kaldırılır. Tekrarlı alanları önlemek için her tablo mümkün olduğunca küçük tablolara bölünür.
- Günümüzde modern ilişkisel veritabanı modellerinde genelde 3. normal formdan sonrası genelde uygulanmaz. 3NF üzeri normalleştirmeler veritabanında çok fazla tablo ve ilişki oluşturur. Sorgu tamamlanma süreleri uzar, performans düşüşleri yaşanır.

VERİTABANI PROGRAMLAMA

- **SQL kısaltması ile kullanılan Yapısal Sorgu Dili (Structured Query Language)** verinin yönetilmesi ve tasarımı için geliştirilmiş bir dildir. SQL, veri ile ilgili yapılacak işlemi VTYS'ye tarif eden bir bildirim dilidir (Declarative Language). Bu işlemin nasıl yapılacağı ile ilgilenmez bu süreç VTYS programları tarafından yürütülür.
- SQL Komutları kullanım amaçlarına göre üç ayrı kategoriye ayrılmaktadır. 1) **Veri Tanımlama Dili (DDL -Data Definition Language)**, 2) **Veri İşleme Dili (DML - Data Manipulation Language)**, 3) **Veri Kontrol Dilidir (DCL Data Control Language)**.
- **Veri Tanımlama Dilinde (DDL);** tabloların oluşturulması, silinmesi ve bazı temel özelliklerinin düzenlenmesini sağlamak üzere sırası ile **CREATE**, **DROP** ve **ALTER** komutları kullanılır. Aşağıdaki tüm işlemler MSSQL diline göre anlatılmıştır.
- **Veri İşleme Dilinde (DML);** **SELECT**, **INSERT**, **DELETE** ve **UPDATE** komutları kullanılır.
- **Veri Kontrol Dilinde (DCL);** kullanıcıya yetki tanımlama için **GRANT**, kullanıcı yetkilerini engellemek için **DENY** ve daha önce yapılmış olan yetki ve izinleri kaldırmak için **REVOKE** komutu kullanılır.
- Farklı yazılım platformları için oluşmuş standart kütüphaneler (**ODBC**, **JDBC**, **ADO.NET** vb) bulunmaktadır. Bu kütüphaneler sadece veritabanı programlama değil, günümüzde yaygın olarak kullanılan web tabanlı ve mobil kullanıcı arayüzlerinin VTYS'ne erişimi için tercih edilmektedir.
- **Görünüm (View):** VTYS'de kullanıcıların ihtiyaçlarına göre verinin farklı şekillerde görüntülenmesini sağlayan yapılardır.
- **SQL Sorgu penceresi:** Ana menüde **"New Query"** ile açılan SQL Sorgu penceresi sorguların çalıştırılması için kullanılmaktadır. Sorgu bir veritabanı üzerinde işlem yapacaksa ana menü veritabanı açılır listesinden ilgili veritabanının da seçili olması gerekmektedir. Bu pencerenin altında **"Command(s) completed successful"** çıktısı yazdığımız komutun düzgün çalıştığını göstermektedir. Komutun diğer çıktıları da burada görülmektedir.
- **Veritabanı Oluşturma:** "Bilisim" adlı veritabanını oluşturmak için, SQL Sorgu penceresine **"CREATE DATABASE Bilisim"** komut satırı yazılıp ana menüde **"Execute"** tıklanır veya **"F5"** tuşuna basılır.
- **Birincil Anahtar (Primary Key)** olarak tanımlanan alan ya da alanlar, ilgili tabloda benzersiz değer alırlar. Diğer bir deyişle aynı değerın farklı satırda yer almamasını garanti altına alırlar.
- Eğer herhangi bir alan için mutlaka değer olması isteniyorsa alan tanımlama devamında **"NOT NULL"** un olması gerekir.
- **SQL gerçek bir programlama dilinin özelliklerine sahip değildir.** İşletmeler ile ilgili bazı veritabanı projelerinde akış kontrolü, döngü vb. yordamsal dil özelliklerinin raporlama, analiz vb. işlemler için kullanılması gerekir. Bu durumlar için bazı yordamsal dil özelliklerini barındıran **T-SQL (Transact-SQL)** dili geliştirilmiştir.
- **@:** T-SQL dilinde yerel değişken tanımlamak için kullanılan ön ektir.
- **@@:** Sistem tarafından bilgi vermek amaçlı oluşturmuş evrensel değişkendir.
- Değişken tanımlama **DECLARE** komutu ile yapılır. **DECLARE @degiskenadi <veri tipi> [(boyut)]**
DECLARE @Bolum_No INT;
DECLARE @Bolum_Adi VARCHAR(50)
- **Değişken isimlendirme kuralları:**
 - Değişkenler **Türkçe karakter ve boşluk** içermez.
 - Değişken isimleri **ilk karakteri harf ile başlayıp** harf, rakam ve alt çizgi (_) ile devam edebilir.
 - SQL veya T-SQL için kullanılan **komutlar ve ayrılmış sözcükler** (SELECT, INSERT, UPDATE, NOT vb.) kullanılmaz.
 - Değişken ismi, SQL'de **özel anlamı olan sembollerle** (@, @@, #, ##, \$) başlamamalıdır.
 - Değişken isimlerinde **küçük veya büyük harf kullanımı fark etmez.**
- Tanımlanan değişkenlere **SET** ya da **SELECT** ifadeleri kullanılarak değer ataması yapabiliriz.
SET @degiskenadi = deger
SELECT @degiskenadi = deger
- Değişken değerlerini ekranda görüntülemek için **PRINT** (veya **SELECT**) komutu kullanılabilir.
- **CASE-END:** Birden çok koşulun mantıksal sorgulanabilmesine imkân tanır. Programlamadaki Switch-Case yapısı gibi çalışır.
CASE ifade
WHEN Değer1 **THEN** Değer1 **WHEN** Değer2 **THEN** Değer2
ELSE 'diğer'
END
- **WHILE:** Verilen bir koşulun sağlanması durumunda belirlenen komut blokunu tekrar eden bir yapıdır.
WHILE koşul_ifadesi
BEGIN
Tekrarlanacak işlemler
END
- **GOTO:** Bu komut ile kod içerisinde belirlenen bir etikete direkt geçiş yapıp bu etiketten sonra devam edilir.
Etiket:
..... Komutlar
GOTO Etiket
- **Dinamik SQL Sorguları:** Bu yöntemde sorgu değişkenleri içerisine eklenen SQL komutları "EXECUTE" fonksiyonu ile çalıştırılabilirler.
DECLARE @tabloAdi VARCHAR(50); **DECLARE @Sorgu** VARCHAR(50)
SET @tabloAdi='Bolumler'; SET @Sorgu='SELECT * FROM '+@tabloAdi
EXECUTE(@Sorgu)

- **İMLEÇ (Cursor):** Veri kümelerinin satırları arasında birer birer ilerlemeyi sağlayan yapılardır. **FETCH** komutu kayıtlar arasında gezinmeyi sağlar, **FETCH NEXT** bir sonraki, **FETCH PRIOR** bir önceki, **FETCH LAST** son kayda ve **FETCH FIRST** ilk kayda ilerlemeyi sağlar.
- T-SQL'de kodlara yorum eklemek için -- veya /* */ işaretleri kullanılır. İlki tek satır, diğeri çoklu satırda açıklama yapmayı sağlar.
- **Yığın İşlemi:** **GO** komutu **GO [sayı]** şeklinde yazılarak en son yığının yazılan sayı kadar çalıştırılması sağlanır. Böylece aynı komut tekrar tekrar çalıştırılarak bir döngü elde edilir.
- T-SQL'de bir takım işlemler sonucunda, değişiklik olan kayıtları listelemek için **OUTPUT** komutu kullanılır. **OUTPUT** **OP**.Alan INTO **Tablo** Burada **OP** yerine **INSERTED** veya **DELETED** kullanılıp noktadan sonra alan isimlerinin yazılması gerekir. Tablo, etkilenen verilerin tutulduğu tabloyu veya tablo değişkenini tutar. Böylece yeni girilen veya silinen veriler gösterilebilir.
- T-SQL'de oluşan hata mesajları ve kodları **"sys.messages"** adlı sistem tablosunda tutulmaktadır. **"SELECT * FROM sys.messages"** komutu ile hepsi listelenebilir.
- **"@@ERROR"** sistem fonksiyonunda ise en son hata mesajının kodu tutulmaktadır.
- **TRY ... CATCH** ile de hata denetimi aşağıdaki gibi yapılabilir:

BEGIN TRY

Test edilen SQL_Komutları

END TRY

BEGIN CATCH

Hata meydana gelirse çalışacak olan SQL_Komutları

END CATCH

- **Saklı Yordamlar (Stored Procedures):** Veritabanlarında tekrarlı işlemler için oluşturulan komut kümeleridir. Belirli bir görevi yerine getirmek için tasarlanmış, sunucu üzerinde tutulan, birden fazla tablo üzerinde işlem yapabilen, program içinden farklı parametreler ile çağrılarak kullanılabilen SQL tabanlı komut kümesidir. Veritabanı yöneticileri, kullanıcılarına saklı yordam bazında kullanıcı hakları tanımlayabildiğinden aynı zamanda güvenliğe katkı sağlarlar. İstemci tarafından birçok satıra (yüzlerce satır olabilir) sahip SQL komutunun sunucuya gitmesinden, sadece saklı yordamın adının sunucuya gitmesi, ağı daha az meşgul eder. Saklı yordamlar **EXEC** komutu ile çalıştırılır.
- **Saklı Yordam Oluşturma ve Silme İşlemleri:**
CREATE PROCEDURE Procedureismi
AS BEGIN
 <SQL Sorgusu>
END

Yordam silme işlemi aşağıdaki gibi yapılır.

DROP PROCEDURE Procedureismi;

- Saklı yordam için gerekli sorguyu yardımcı kullanarak oluşturma için **"Design Query in Editor"** seçilir veya **Ctrl+Shift+Q** tuşları kullanılır.
- **Sistem Fonksiyonları**

Fonksiyon Grup Adı	İşlevi
Kümeleme Fonksiyonları (Aggregate Functions)	Belli bir veri kümesinde işlem yapıp tek değer döndüren; Avg (ortalama alır), Count (veri kümesi satır sayısını döndürür) vb. fonksiyonlardır.
Yapılandırma Fonksiyonları (Configuration Functions)	Sunucunun mevcut yapılandırılması hakkında bilgi veren; @@Servername (sunucu adı), @@version (SQL sunucu sürümü) vb. fonksiyonlardır.
İmleç Fonksiyonları (Cursor Functions)	İmleçler ile ilgili veri döndüren Bölüm 2 de örnek olarak kullanılan fonksiyonlardır.
Tarih ve zaman fonksiyonlar (Date and Time Functions)	Tarih ve zaman üzerinde işlemler yapan ve karakter, nümerik değer veya tarih bilgisi döndüren; Getdate (tarih al), Month (ay) vb. fonksiyonlardır.
Matematiksel Fonksiyonlar (Mathematical Functions)	Girdi değerlerine bağlı nümerik değer döndüren; Log (logaritma), Abs (mutlak değer) vb. fonksiyonlardır.
Metaveri Fonksiyonları (Metadata Functions)	Veritabanı ve veritabanı nesneleri üzerinde bilgi döndüren; Object_Id (Nesne numarası), Object_Name (Nesne Adı) vb. fonksiyonlardır.
Güvenlik Fonksiyonları (Security Functions)	Kullanıcılar ve rolleri hakkında bilgi döndüren; User_Id (kullanıcı numarası), User_Name (kullanıcı adı) vb. fonksiyonlardır.
Dizgi Fonksiyonları (String Functions)	Dizgiler üzerinde işlemler yapan; Len (dizgi uzunluğu), Reverse (dizgiyi tersine çeviren) vb. fonksiyonlardır.
Sistem ile ilgili statiksel Fonksiyonlar (System Statistical Functions)	İstatiski bilgi sağlayan; @@Total_Errors (toplam hata sayısı), @@Total_Read (toplam okuma sayısı) vb. fonksiyonlardır.

- **Kullanıcı Tanımlı Fonksiyonlar:**
CREATE FUNCTION Fonksiyon oluşturmak için kullanılır.
ALTER FUNCTION Fonksiyonda değişiklik yapmak için kullanılır.
DROP FUNCTION Mevcut olan fonksiyonu silmek için kullanılır.

Fonksiyon oluşturma örneği:

CREATE FUNCTION *Fonksiyon_Adi* (Parametreler)

RETURNS *geriDonusTipi*

AS BEGIN

-- SQL Sorguları

RETURN *geriDonusDegeri*

END

- **Tablo Değerli Fonksiyonlar:** Çalıştıktan sonra geriye bir tablo ile dönen fonksiyonlardır.
- **Sayı Değerli Fonksiyonlar:** Tek bir sayısal değer döndüren fonksiyonlardır.
- **Kümeleme Fonksiyonları:** Sistem fonksiyonları altında bulunan **MIN()**, **MAX()**, **AVG()**, **SUM()** gibi fonksiyonlardır.
- **Saklı yordamlar ile fonksiyonlar arasındaki önemli farklar:**
 - **WHERE/HAVING/SELECT** sadece fonksiyonlarda kullanılabilir.
 - Fonksiyon içinde saklı yordam çağrılmaz. (Tersi mümkün)
 - Fonksiyon mutlaka bir tablo veya sayılı değer döndürmelidir. Saklı yordam mecbur değildir.
 - **INSERT/UPDATE/DELETE** komutları kullanıcı tanımlı fonksiyonlarda yapılamaz.
 - **TRY ... CATCH** yapısı ile hata ayıklama fonksiyonlar içinde kullanılmaz.
 - Hareket yönetimi fonksiyon içinde kullanılmaz.
 - Saklı yordamlar derlenmiş olarak veritabanında tutulurken, fonksiyonlar çalışma zamanında derlenir ve çalıştırılır.
- **Aktif Veritabanı:** Veritabanı içinden veya dışından gelen olaylara otomatik olarak tepki üreten veritabanına denir.
- **Veritabanı bütünlüğü sağlama teknikleri:**
 - 1) Tanımlanabilir Veri Bütünlüğü
 - a. **Kısıtlar**
 - i. **Birincil anahtar kısıtlayıcı (PRIMARY KEY):** Tablodaki her satırın diğer satırlardan farklı olmasını sağlar. NULL olamaz!
 - ii. **Tekil alan kısıtlayıcısı (UNIQUE):** Birincil anahtar kısıtlayıcısının aksine tablodaki birden fazla sütuna tekil kısıtlayıcı tanımlanabilir ve sütundaki değer NULL olabilir.
 - iii. **Kontrol kısıtlayıcı (CHECK):** Tablodaki herhangi bir sütunun hangi gruptan verileri alabileceğini ve girilebilecek verileri bir koşul ile kısıtlayarak istenilen verilerin girilmesini sağlayan bütünlüktür.
 - iv. **Yabancı anahtar kısıtlayıcı (FOREIGN KEY):** Bir tablodaki bir sütundaki değer diğer tabloardaki değerlerle denetlenmesini sağlayan kısıtlayıcıdır.
 - v. **Varsayılan kısıtlayıcı (DEFAULT):** Bir tabloya veri girişi esnasında verinin girildiği alan için alacağı varsayılan bir değer tanımlanması için kullanılan kısıtlayıcıdır.
 - b. **Kurallar**
 - c. **Varsayılanlar**
 - 2) Prosedürel (Programsal) Veri Bütünlüğü
 - a. **Tetikleyiciler:** Veritabanı bir eylem gerçekleştirdiğinde otomatik olarak başka eylem veya eylemler gerçeklemesi amacıyla kullanılan özel saklı yordamlardır. İşlemler RAM’de gerçekleşir. Tetikleyiciyi oluşturmak için **CREATE TRIGGER** kullanılır. Daha önce oluşturulmuş bir tetikleyici **ALTER TRIGGER** komutu ile düzenlenebilir.
 - i. **Ekleme (INSERT):** Yeni kayıt veya kayıtlar girildikten sonra otomatik olarak bir eylem veya eylemler yapılması istenildiği zaman kullanılır.
 - ii. **Silme (DELETE):** Kayıt silme işleminden sonra otomatik olarak devreye girer. Silinen kayıtlar DELETED sahte tablosuna kaydedilir. Asıl tablodan silinen kayıt artık DELETED sahte tablosunda yer almaktadır. Böylece silme işleminden vazgeçilmesi durumunda kayıt kaybedilmemiş olur.
 - iii. **Güncelleme (UPDATE):** Kayıtlarda güncelleme olduğunda devreye girer.
 - b. **Saklı yordamlar**
 - c. **Program kodları**
- **Kısıt:** Veritabanında bütünlüğü sağlamak için veri üzerinde oluşturulmuş mantıksal sınırlamalara denir. Kısıtların kullanılmasının en temel sebebi veritabanına hatalı giriş yapılmasını engellemektir.
- **Eş zamanlılık (concurrency):** Aynı veya farklı istemciler tarafından gerçekleştirilen veritabanı hareketlerinin, bir veri veya veri grubu üzerinde aynı anda işlem gerçekleştirmeye çalışmasıdır.
- **Hareket (transaction):** Bir veritabanı üzerinde gerçekleştirilen bir dizi SQL işlemini kapsayan, daha küçük parçalara bölünemeyen en küçük işlem grubudur. (SELECT, INSERT, UPDATE ve DELETE). Kalıcı hâle gelebilmeleri, hareket bloğu içindeki tüm SQL ifadelerin başarılı bir şekilde gerçekleştirilmesine bağlıdır. SQL işlemleri belirlenen bazı şartlar sağlanmış ise veritabanına uygulanır ve kalıcı hâle getirilir. Hareket bloğu içinde bulunan tüm işlemler bazı sebeplerden dolayı başarılı bir şekilde tamamlanamadan hata oluşabilir. Bu durumda; 1) Veritabanında hatanın meydana geldiği zamana kadar gerçekleşen tüm değişiklikler iptal edilmeli, 2) Hareket başlamadan önceki başlangıç durumuna geri dönmelidir. Bunun için veri tabanlarında hareket blokları **ACID** olarak kısaltılmış olan dört özelliğe sahip olmalıdır. **Bölünmezlik, Tutarlılık, İzolasyon ve Devamlılık**.

- **ACID:** İngilizce Atomicity (bölünmezlik), Consistency (tutarlılık), Isolation (izolasyon) ve Durability (devamlılık) kelimelerinin baş harflerinden oluşur.
- **Bölünmezlik:** Hareket bloğu içinde yer alan SQL ifadelerin tamamının başarılı olarak çalıştırılması ya da hiçbirinin çalıştırılmadan hareket bloğu başlangıç durumuna geri dönülmesidir.
- **Tutarlılık:** Veritabanında gerçekleştirilen işlemler sonucunda oluşan yeni veriler arasındaki tutarlılığı ifade eder.
- **Hareket Günlüğü:** Transaction log
- **İzolasyon:** Veri erişim kontrolü amacıyla kullanılan bir mekanizmadır. Örneğin bir hareket bloğunun veri üzerinde değişiklik yaptığı sırada başka bir hareket bloğunun bu veriyi okumak istediğini düşünelim. Bu durumda veriyi okumak isteyen hareket bloğu veri tutarlı bir duruma gelinceye kadar engellenir ve çalışmasına izin verilmez.
- **Devamlılık:** Tamamlanmış olan ya da devam eden hareketlerin gerçekleştirdiği veri değişikliklerinin VTYS'de bir arıza meydana gelse bile kalıcı olarak kaydedildiğinden emin olunması beklenmektedir. MS SQL Server VTYS'nin en önemli özelliklerinden biri, bu amaçla hareket günlüğü dosyası tutmasıdır. Hareket tamamlandığında bellekte gerçekleşen veri değişiklikleri fiziksel disk üzerindeki veri dosyaları üzerine kaydedilir. Değişiklikler önce **hareket günlüklerine yazılır**. Sonra veri dosyalarında kalıcı hâle getirilir. Böylelikle bir çökme veya bozulma anında en güncel veri **hareket günlüklerinde** bulunmuş olur.
- **Veri Dosyaları:** MS SQL Server VTYS'de, veriler üzerindeki kalıcı değişiklikler MDF (master data file), hareket günlükleri LDF (log data file) uzantılı dosyalarda kayıt altına alınmaktadır.
- MS SQL Server VTYS, **otokayıt** (autocommit), **açık** (explicit) ve **örtük** (implicit) olmak üzere üç farklı modda hareket desteği sağlamaktadır.
- **Otokayıt Hareket:** Hareket içindeki her bir SQL komutu tamamlandıktan sonra otomatik olarak veritabanına kaydedilir. İki adet insert komutunun olduğu, birinin hatalı olduğunu düşünelim. Hatasız olan, tüm veriler doğru olarak girildiğinden bu ifade başarı ile sonlanır. Çünkü her bir insert komutu kendi hareketi içinde tamamlandığında otomatik kayıt gerçekleşecektir.
- **Örtük Hareket:** MS SQL Server VTYS tarafından; **Alter Table, Grant, Truncate Table, Fetch, Select, Drop, Revoke, Delete, Open, Create, Insert, Update** ifadelerden herhangi biri ilk kez yürütüldüğünde yeni bir hareket otomatik olarak başlatılmaktadır. Bu hareket bloğu **Rollback** veya **Commit** ifadelerinden herhangi biri yürütülene kadar açık kalmaktadır.
- **Rollback, commit (geri al, kalıcı yap):** Bir hareket bloğunun veritabanı üzerinde gerçekleştirdiği değişikliklerin **kalıcı hâle getirilmesi için commit; değişikliklerin geri alınması için ise rollback** ifadesi kullanılır.
- **Açık Hareket:** Kullanıcıların kendilerinin başlatıp sonlandırdığı hareket bloklarıdır. Veritabanı uygulamalarında veri değişiklikleri yapılırken tavsiye edilen hareket modudur. Bir hareket bloğunun başarılı ya da başarısız olması sonucunda hangi işlemlerin yapılacağı uygulama geliştirici tarafından açık bir şekilde kontrol edilebilir. Bu yüzden daha çok tercih edilen bir hareket yönetim modudur.
- **Oturum:** Bir veritabanına farklı kullanıcılar tarafından gerçekleştirilen erişimdir.
- VTYS'lerde aynı anda birden fazla oturum üzerinden aynı kaynaklara erişilmeye çalışılması eş zamanlılık problemlerini ortaya çıkarmaktadır. Bu problemler genel olarak **kayıp güncelleme, kirli okuma, tekrarlanamayan okuma** ve **hayalet okuma** olmak üzere dört ana grupta toplanmaktadır.
- **Kayıp Güncelleme:** İki ayrı hareket, **iki farklı zamanda aynı kaynak üzerinde güncelleme/kayıt yaptığına, birinin yaptığı kayıtlı diğerinden önce olacağı için, sonra yapılan kayıt geçersiz** olur. Buna kayıp güncelleme denir.
- **Kirli Okuma:** Bir hareket tarafından değiştirilmiş fakat **kalıcı olarak kaydedilmemiş bir verinin başka bir hareket tarafından geçerli bir veri gibi okunması** kirli okuma olarak ifade edilmektedir.
- **Tekrarlanamayan Okuma:** Bir veriye bir hareket içinde **birden fazla kez erişim** gerçekleştirilebilir. Bu erişimler arasında eş zamanlı olarak **başka bir hareket güncelleme veya silme** işlemi gerçekleştirebilmektedir. Bu durumda tekrarlanamayan okuma problemi ile karşı karşıya kalınmış olunmaktadır.
- **Hayalet Okuma:** Aynı hareket içinde ve aynı tablo üzerinde aynı WHERE ifadesi ile **iki farklı sorgulama yapıldığını varsayalım**. Bu iki sorgulama arasında eş zamanlı olarak **başka bir hareket, okunan aralık içindeki veri kümesine yeni bir kayıt eklemiş olsun**. Bu durumda **ikinci sorgulamada birinciye göre daha fazla kayıt** okunmuş olur.
- MS SQL Server VTYS eş zamanlı hareketlerin rekabet etkinliğini kontrol etmek için **kilit mekanizması** kullanılmaktadır.
- **İzolasyon düzeyleri**, eş zamanlı olarak çalışan hareketlerin birbirlerini nasıl etkilemesi gerektiğinin belirtilmesinde kullanılmaktadır.
- **Kaydedilmiş okuma**, varsayılan MS SQL Server VTYS izolasyon düzeyidir. Kaydedilmemiş veri değişiklikleri okunamaz. Sorgu esnasında paylaşılan kilitler kullanılır ve bir hareket okuma gerçekleştirirken eş zamanlı olarak başka bir hareket aynı veri üzerinde değişiklik gerçekleştiremez.
- **Kaydedilmemiş okuma**, en düşük kısıtlamaya sahip izolasyon düzeyidir. Hareket tarafından seçilmiş olan veri üzerine herhangi bir kilit uygulanmaz. Yüksek eş zamanlılık sağlamakla birlikte veri tutarlılığı sağlama düzeyi düşüktür.
- **Tekrarlanabilir okuma**, aktif edildiğinde kirli okuma ve tekrarlanamayan okuma eş zamanlılık problemleri ortadan kaldırılabilir. Okunan tüm kaynaklara paylaşılan (S) kilit uygulanır.
- **Serileştirilebilir** izolasyon düzeyi **en kısıtlayıcı izolasyon** düzeyidir. Bir hareket bir kayıt üzerinde okuma işlemi gerçekleştirdiği sırada başka bir hareketin okunan kayıtlar üzerinde değişiklik yapmasına izin verilmez. Aynı şekilde ilgili veri aralığına yeni bir kayıt eklemesine de izin verilmemektedir.
- **Anlık görüntü** izolasyon düzeyi hareketin başladığı sırada okunan verinin hareket içinde tutarlı bir değerinin kullanılmasına imkan vermektedir. Veri okuma işlemi esnasında diğer hareketlerin değişiklikleri bloke edilmez. Bununla birlikte, anlık görüntü izolasyon düzeyindeki bir hareket, yapılan değişiklikleri tespit edemez.
- **Kilitleme:** Veritabanı kaynakları ya da kayıtlara başka oturum ya da hareketler tarafından yapılacak eş zamanlı erişimin engellenmesi işlemidir.

- **Kilit Modu:** Eş zamanlı hareketlerin veritabanı sunucusu kaynaklarına ne şekilde erişilebileceğinin belirlenmesidir.
- **İzolasyon Düzeyi:** Eş zamanlı olarak çalışan hareketlerin birbirlerini nasıl etkilemeleri gerektiğini belirtmek amacıyla kullanılır.
- **Paylaşılan Kilit Modu:** Bu kilit modunda okuma yapılan sorgu ve hareketlerin verilere erişimine izin verilir. Bununla birlikte kilit kaldırılana kadar eş zamanlı olarak farklı bir hareketin güncelleme yapmasına izin verilmez.
- **Ayrıcalıklı Kilit Modu:** Bir veri kaynağı ayrıcalıklı kilit modunda ise eş zamanlı hareketler bu veri kaynağı üzerine başka hiçbir kilit modu uygulayamaz. Bunun anlamı, veri kaynağı üzerinde değişiklik gerçekleştiren hareket, Commit veya Rollback işlemini gerçekleştirmeden diğer hareketler bu veri kaynağına erişemeyecektir.
- **Güncelleştirme Kilit Modu:** Paylaşılan ve ayrıcalıklı kilit modlarının birleşimidir.
- **Kilitlenme (deadlock),** iki ya da daha fazla hareketin aynı kaynaklara erişim gerçekleştirmek istediği durumlarda meydana gelmektedir.
- **Kilitlenmeyi önlemek amacıyla** kilitlenme izleyicisi adındaki sistem görev izleyicisi her beş saniyede bir yürütülür. Böylece sistemde bir kilitlenme olup olmadığı kontrol edilmiş olur. Bir kilitlenme tespit edildiğinde kilitlenmeye sebep olan hareketlerden birisi üzerinde rollback işlemi gerçekleştirilir.
- **1205:** Kilitlenme hata numarasıdır.
- **Koklayıcı (Sniffer):** Paket çözümleyicisi ya da koklayıcı, bir sayısal ağ üzerinden geçen trafiği izlemeye yarayan yazılımların genel adıdır.
- Türk Standartları Enstitüsü (TSE) tarafından Türkçeye çevrilerek yayınlanan TS ISO/IEC 27001:2005 **Bilgi Güvenliği Yönetim Sistemi Standardı**, bilgi güvenliğini üç başlık altında inceler: **Gizlilik, Bütünlük ve Kullanılabilirlik.**
- **Gizlilik:** Kuruma veya bireye özel ve gizliliği olan bilgilere, sadece yetkisi olan kişilerin erişim hakkına sahip olmasının gerekliliğidir. Örneğin, bir öğrenciye diğer öğrencilerin notlarını incelemek için izin verilmemelidir.
- **Bütünlük:** Kısaca bilinçli veya bilinçsiz olarak verinin bozulmaması gerekliliğidir. Örneğin, öğrencilere notlarını görmek için izin verilebilir, ancak notların öğrenciler tarafından değiştirilmesi için izin verilmemesi gerekmektedir.
- **Kullanılabilirlik:** Bilgi ve kaynakların ihtiyaç duyan kişilerce sürekli erişilebilir durumda olması olarak özetlenebilir. Örneğin, dersi veren kişinin, öğrencinin notlarına erişebilmesi ve onları düzeltebilmesi gerekmektedir.
- Veritabanı yönetim sistemlerindeki erişimi düzenleme yöntemleri:
 - **Kısıtlı erişim denetim mekanizması:** Kullanıcı veya kullanıcı grupları için veri dosyalarına, kayıtlara veya veritabanının bazı alanlarına seçme, ekleme, silme veya güncelleme gibi erişim hakkı veya ayrıcalıklar vermektir. **GRANT** komutu ile kullanıcılara erişim hakkı verilirken **REVOKE** komutu ile verilmiş olan haklar geri alınmaktadır.
 - **Sistemsel erişim denetim mekanizması:** Roller olarak isimlendirilen ve bireysel kullanıcılar tarafından değiştirilemeyen ve kuruluşun güvenlik politikalarını gerçekleştirmek için kullanılan bu yöntemde veriler ve kullanıcılar için belirli güvenlik etiketleri tahsis edilir.
- Verilen erişim hakkından bazılarını kısıtlama getirilmek isteniyor ise **DENY** komutu, tüm kısıtlama ve izinlerin iptal edilmesi isteniyor ise **REVOKE** komutu kullanılmaktadır.
- MS SQL Server'de oluşturulan bir kullanıcı veritabanında ön tanımlı olarak aşağıdaki dosya türleri bulunur:
 - **Veri dosyaları (.mdf ve .ndf):** Tabloları ve indeksleri barındırır. Veri dosyaları (ön tanımlı 8 kb) bloklara ayrılmıştır ve sql veritabanı tüm işlemleri; blokları yüklemek, bellekte tutmak, değiştirmek ve bir bloğu diske kaydetmek şeklinde gerçekleştirir.
 - **İşlem kayıtlarını tutan hareket dosyaları ("*.ldf"):** Kayıt dosyası, bir açıdan veri dosyasında yapılacak işlerin komut dizisidir ve geriye doğru da çalıştırılabilir. Bu sayede bir işlemin geri alınabilmesi ya da sistemin çökmesi durumunda veritabanının kararlı bir noktaya geri döndürülebilmesi mümkün olur.
- **Oturum açma:** Login, **Replikasyon:** Replication, **İkizleme:** Mirroring, **Geri alma:** Rollback.
- MS SQL Server, çalışmak için **dört sistem veritabanı**na ihtiyaç duyar:
 - **Master:** MS SQL Server'ın tüm yapılandırma bilgileri bu veritabanında yer alır. Bu yapılandırma bilgilerinde; genel sistem yapılandırması, oturum açma bilgileri, veritabanlarının konumları, diğer bağlı sunucular, replikasyon ve ikizleme gibi işlemlerin bilgileri yer alır.
 - **Model:** Yeni oluşturulan veritabanları için ön tanımlı örnektir. Yeni bir veritabanı oluşturulduğunda Model veritabanının kopyası oluşturulur.
 - **MsdB:** MS SQL Server SQL Agent servisinin yapılandırma bilgilerini içerir. Veritabanı uyarı ve işleri ile ilgili bilgileri tutar.
 - **Tempdb:** MS SQL Server'da aşağıdaki işlemler Tempdb veritabanı üzerinde gerçekleştirilir:
 - **Geçici tablolar,** her türlü yordam değişkenleri ve imleç verileri Tempdb'ye yazılır ve okunur.
 - **Verilerin sıralama işlemleri** Tempdb üzerine yazılarak gerçekleştirilir (order by).
 - **Sürüm yönetimi bulunan tablolarda verilerdeki değişiklikler** Tempdb'de depolanır. Bazı hareket türlerinde (isolation level transactions) verinin değişmeden önceki hâli, diğer kullanıcıların erişimine sunulabilmesi için Tempdb'de saklanır.
 - **Çevrimiçi oluşturulan indeksler,** Tempdb'ye yazılarak oluşturulur ve tamamlandıktan sonra Tempdb'den okunarak sahibi olan veritabanına yazılır.
- **Yedekleme;** verinin, veritabanı sunucusunun dışına çıkartılması ve güvenlik altına alınmasıdır.
- **Yedekleme Çeşitleri:** tam yedekleme, fark yedeklemesi, hareket günlük yedeklemeleri.
 - **Tam yedekleme:** Sadece veritabanı dosyalarında içinde veri olan blokları kaydeder.
 - **Fark yedeklemeleri:** Artımlı yedekleme olarak da anılır. Son tam yedekten itibaren verinin sadece değişen blokları alınarak gerçekleştirilir. Daha az veri yedeklendiği için daha kısa sürede tamamlanır ve az yer kaplar. Ancak bir fark yedeği, kendisinden önce alınmış tam yedek ile bir arada kullanılabilir.
- **Hareket günlük yedeklemeleri:** Verinin kendisini değil, veride yapılan değişiklik işlemlerinin kayıtlarını elde eder ve saklar.

- **Yedek seti (Backup set):** Bir ortam setine kaydedilen yedekler, aksi söylenmedikçe yedek seti adında bir indekse eklenir. Yedek setleri yedeklerin bir arada yönetilebileceği kümeler oluşturur.
- **Replikasyon:** Ana veritabanında gerçekleşen işlemlerin hareket kayıtları kısa periyotlarla çalışan bir işlem aracılığı ile ayrı bir disk alanına dosya blokları şeklinde bırakılır. Kopya veritabanları, bu hareket dosyalarını okur ve işlemleri taklit ederek kendilerini ana veritabanı ile eşitlerler.
- **İkizleme:** Beklenmedik bir hata, felaket ya da istenmeyen durumlarda felaketten dönme ve yüksek erişilebilirliği sağlamak için geliştirmiş olan bir çözümdür. İkizleme, şahit veritabanı sunucusu kullanıldığında asıl veritabanının devre dışı kaldığı hâllerde otomatik olarak kopya veritabanının devreye girmesine izin verir.
- **Yedekleme / Geri yükleme maliyeti:** Yedekleme işlemlerinin kurgu ve hazırlığı için çalışanların emek maliyeti, alınması gereken donanım ya da hizmetlerin maliyeti, yedekleme / geri yükleme işlemlerinin gerektirdiği sistem kaynakları (işlemci ve disk yükleri), yedekleme / geri yükleme işlemlerinin gerektirdiği süre ve yedekleme bütçesi maliyet olarak ele alınır.
- **Yedeklerin rotasyonu:** Yedeklerin kullandığı disk alanının artmaması için bir kısmının (genellikle eski tarihli) otomatik olarak silinmesi ve yenilerine yer açılması.
- **Yedeklerinizi veritabanı ile aynı sistemde tutmamanız,** ilgili sistemden uzaklaştırmanız gerekir. Aksi durumda ilgili sistemde ortaya çıkacak felaketlerde yedekleriniz de zarar görecektir.
- **Veritabanı bakım planındaki görevler;** Yedekleme görevlerini planlamak, Tutarlılık kontrolü, İndeks bakımı, İstatistik bakımı, Bakım sonrası temizlik, Veritabanı sıkıştırma oluşturma.
- **Bulut bilişim,** tek bir yerel sunucunun ya da kişisel bir cihazın kaynağına bağlı olarak yapılan hesaplama yerine birden fazla cihazın kaynaklarını beraber kullanarak hesaplama yapma işi olarak tanımlanabilir.
- **Anaçatı bilgisayar,** (mainframe) yüzlerce kullanıcıya aynı zamanda farklı hizmetler verebilen içerisinde çok sayıda işlemci ve sabit disk barındıran güçlü ve pahalı bilgisayarlardır.
- Bulut bilişim fikri ilk olarak 1969'da Leonard Kleinrock tarafından ortaya atılmıştır.
- Bulut bilişim sistemleri erişim modellerine göre dörde ayrılırlar: genel bulut, topluluk bulut, özel bulut ve karma bulut.
 - **Genel bulut** yapılarında depolama, yazılım ve diğer kaynaklar hizmet sağlayan şirket tarafından genel son kullanıcılara sunulur. Sunulan hizmet kullandığın kadar öde modeli ile ya da ücretsiz olarak kullanıcılara servis edilir. Amazon EC2, Google AppEngine, Dropbox
 - **Topluluk bulut** hizmeti ortak ihtiyaçları olan ve bu ihtiyaçları beraber karşılama amaçları olan çeşitli organizasyonların altyapılarını birbirleri ile paylaşarak kaynakları daha etkin bir biçimde kullandıkları bulut yapılarıdır.
 - **Özel bulut** yapıları sadece tek bir organizasyon için işletilen yapılardır. Bu tür bulutların sadece şirket içi kullanımına izin verilmektedir, dış dünya/son kullanıcı ile herhangi bir bağlantıları yoktur.
 - **Karma bulut** yapıları en az iki veya daha fazla özel, topluluk veya genel bulut yapılarının birleşerek oluşturduğu yapılardır.
- **Bulut Bilişim Hizmetleri Modelleri:** Hizmet olarak altyapı, hizmet olarak platform, hizmet olarak yazılım.
 - **Hizmet olarak altyapı (IaaS),** teknoloji işletmeleri veya geliştiriciler için dosya saklama alanı ya da hesap yapabilme gibi kaynakları hizmet olarak sunulduğu yapılardır. IaaS İngilizce "Infrastructure as a Service" kelimelerinin ilk harflerinin birleşmesi ile oluşmuştur. Kullanıcılar nasıl elektrik su gibi hizmetler için kullandığı kadar ödeme yapabiliyorsa aynı durum burada da geçerlidir. Amazon EC2, Amazon Web Services, BlueLock, PrimaCloud.
 - **Hizmet olarak platform (PaaS),** teknoloji işletmeleri ve geliştiriciler için bulut bilişime hazır çözümler üretmesini sağlayan yapıların sunulduğu modeldir. PaaS İngilizce "Platform as a Service" kelimelerinin ilk harflerinin birleşmesi ile oluşmuştur. ActiveState Stackato, Apprenda, Amazon Elastic Beanstalk, Microsoft Azure, Centurylink Appfog, CloudControl dotCloud, Engine Yard, Google App Engine, IBM Bluemix, Pivotal Cloud Foundry, Red Hat OpenShift, Heroku.
 - **Hizmet olarak yazılım (SaaS),** belirli bir amaç için geliştirilmiş yazılımları son kullanıcıya kiralama yöntemi ile sunmamızı sağlayan bulut yapılarıdır. Yazılım sağlama hizmeti ile kullanıcıların ihtiyaç duyduğu CRM, ERP, finans ve muhasebe, ofis uygulamaları, e-posta yazılımları gibi programları bulut üzerinde dağıtılır. SaaS İngilizce "Software as a Service" kelimelerinin ilk harflerinin birleşmesi ile oluşmuştur. Salesforce, Workday, Netsuite, ServiceNow, Athenahealth, Microsoft Office Online, Google Apps, Concur, Citrix GoToMeeting, Cisco WebEx, TTNet Bulutu.
- **ACID** kelimesi Atomicity, Consistency, Isolation, Durability kelimelerinin ilk harflerinin birleşmesi ile oluşmuştur.
- **Hareket** veritabanında yapılan işlemler olarak adlandırılır. İngilizce "**transaction**" teriminin Türkçedeki karşılığıdır.
- **Eric Brewer** tarafından **1998** yılında ortaya konan CAP teoremi ya da bir diğer adı ile Brewer teoremi dağıtık sistemlerin aynı anda tutarlılık, ulaşılabilirlik ve bölünebilirlik toleransı koşullarına sahip olamayacağını söyler.
- **CAP** kelimesi İngilizce "Consistency", "Availability" ve "Partition Tolerance" kelimelerinin ilk harflerinden oluşturulmuştur. Türkçede sırası ile tutarlılık, ulaşılabilirlik ve bölünebilirlik toleransı kelimelerine karşılık gelmektedir.
- İlişkisel veritabanı yapılarında **tablolar üzerinde birleştirme "join" komutu ile** yapılabilir. İlişkisel veritabanı yapılarında **tablolar üzerinde grupta "group by" komutu ile** yapılır.