
1C: İşletme 8.2

Uygulama Geliştirme Kılavuzu

İstanbul
1TÇ Şirketi
2011

1C:İşletme 8. Uygulama geliştirme kılavuzu

Basım numarası: 82.1.1.001

Basım tarihi: 05.10.2011

Orijinal baskı:

Maxim Radchenko, Elena Khrustaleva. 1C:Enterprise 8.2. A Practical Developer's Guide. Examples and Standard Techniques, Moscow, 1C-Publishing, 2010 – 838 p.
ISBN 978-5-9677-1418-4

Türkçe yayım ve çeviri 1C-Publishing'in izni ile yapılır.

© Türkçe yayım hakları ve çeviri: One C Enterprise (1TC) Ltd. Şti.
2007-2011. Bütün hakları saklıdır.

1TC, One C Enterprise Bilgi İşlem Otomasyonu Yazılım ve Danışmanlık Hizmetleri Sanayi ve Ticaret Ltd. Şti.'nin tescilli ticari markasıdır. Diğer adlar kendi sahiplerinin ticari markaları olabilir.

1C:İşletme 8 yazılım sistemini ("Program") almakla İşletme 8'in haklarını korumayı ve 1TC'nin önceden yazılı izni olmadan yazılımın ve belgelerin kopyalarını çıkarmaktan kaçınmayı kabul etmiş olursunuz.

Program (hem yazılımı hem de belgeleri kapsamaktadır) mülkiyete ilişkin bilgiler içermektedir. Bu bilgiler bir lisans sözleşmesinin yanı sıra telif hakları ve diğer fikri mülkiyet haklarına dair kanunlarla da korunmaktadır. Programın tersine mühendisliği, bileşenlerine ayrılması ve ters yönde derlenmesi yasaktır.

Bu belgede yer alan bilgiler daha önceden bildirimde bulunmaksızın değiştirilebilir. Belgede herhangi bir sorun bulduğunuz takdirde lütfen bize yazılı olarak bildiriniz. Bu belgenin hatasız olduğu hususunda garanti verilmemiştir.

Lisans sözleşmesinde tarafımızca açık olarak belirtilmiş haller haricinde, Program herhangi bir amaçla, herhangi bir biçimde veya herhangi bir elektronik veya mekanik bir yolla yeniden oluşturulamaz veya başka bir yere aktarılamaz.

One C Enterprise Bilgi İşlem Otomasyonu Yazılım ve Danışmanlık Hizmeti Sanayi ve Ticareti Ltd. Şti.

Beşiktaş P.K. 80, İstanbul, Türkiye

Telefon: +90212 327 74 90

Fax: +90212 327 74 91

e-mail: 1tc@1tc.com.tr

web: www.1tc.com.tr

DANIŞMA HATTI

“1TC” şirketi kullanıcılarına danışma hizmeti sunmaktadır.

Danışma hizmetinden sadece kayıt yapıldıktan sonra faydalananabilir. Kayıt yapmak için dağıtım paketinde var olan yazılım ürünü kullanıcısı, kayıt anketini doldurup posta ile “1TC” firma adresine göndermesi gereklidir.

Dağıtım paket fiyatına, yazılım ürünü kullanıcısının kayıt anketinde belirlenen süre boyunca, bedava danışma hizmet fiyatı dâhildir. Dağıtım paketi içinde kayıt anket formu ve posta adresini içeren zarf bulunmaktadır.

Bedava danışmanlık hizmet süresi dolduğunda, danışmanlık hizmetini almak için “1TC” şirketinin bayilerine başvurulması gerekmektedir.

Danışma hattının adresi ve telefonu:

Adres: P.K. 80 Beşiktaş 34353 İstanbul/Türkiye

Tel: 90 (212) 275-90-19

Telefon danışma hattı cumartesi ve pazar günleri hariç saat 10.00 ile 18.30 arası çalışmaktadır.

Danışma hattına başvurulurken, bilgisayarın kullanıcı yanında bulunması gereklidir. Bu kitap ve yazılım ürünü kullanıcısı kayıt anketinin ikinci parçasının da kullanıcının yanında olması gerekmektedir. Bilgisayar ve yazıcı tiplerinin ilgili destek çalışmasına önceden belirtilmesi gereklidir.

Danışma hattı arandığında, şirketinizin unvanı ve dağıtım paketinde bulunan CD'deki veya yazılım ürünü kullanıcı kayıt anketinin ikinci parçasındaki kayıt numarasının (veya istenirse diğer kayıt bilgilerinin) ilgili danışma personeline söylemesi gereklidir. Telefonda iletilen bilgiler, 1TC şirketine göndermiş olduğunuz yazılım ürünü kullanıcı kayıt anketinde bulunan bilgilere göre kontrol edilir.

Ilgili danışma hattı personeli, sorularınızı cevaplandırırken kendi bilgisayarında da mevcut durumu oluşturabilir. Duruma göre ilgili personel hemen veya geliştiricilerle konuştuktan sonra kullanıcıya çözüm sunabilir.

Danışma hattının görüşmeleri (sorun ve çözümler) kayıt altında tutulmaktadır, dolayısıyla kullanıcı daha önce karşılaştığı bir sorunla ilgili görüşmek isterse, önceki konuşmanın tarih ve zamanını söyleyerek referans verir, böylece danışma hattı personeli konuya daha hızlı hakim olabilir.

**SİZE YARDIMCI OLMAKTAN
HER ZAMAN MEMNUNİYET DUYARIZ!**

İçindekiler

Giriş	7
Ders 1 (0:40) Veritabanı oluşturma ve genel detayları	11
Ders 2 (0:45) Alt Sistemler	31
Ders 3 (2:10) Kart Listeleri	47
Ders 4 (1:30) Evraklar	97
Ders 5 (0:50) Teori	129
Ders 6 (0:50) Birikim Kayıt Tabloları	175
Ders 7 (0:25) Basit Rapor	195
Ders 8 (1:10) Şablonlar. Form ve şablon tasarıımı	207
Ders 9 (0:50) Periyodik Bilgi Kayıt Tabloları	223
Ders 10 (0:30) Sayımlar	237
Ders 11 (1:20) Evrakı Birden Fazla Kayıt Tablosuna Kaydetme	245
Ders 12 (0:40) Akışlar Birikim Kayıt Tabloları	259
Ders 13 (4:30) Raporlar	269

Önsöz

Bu kitap 1C:İşletme 8.2 platform sürümü çıkartıldıktan sonra yayınlanmıştır.

Bu kitabın hazırlanmasının en önemli iki yönü vardır:

Biri – 1C:İşletme 8.2 platform sürümünde geliştirme yapmak için gereklili olan tüm ana araçlar için örnek gösterilmiştir: yönetilen arayüzü tasarlama, yönetilen formları geliştirme, rapor seçeneklerini ve onların yeni ayarlama biçimlerini kullanma, evrak kaydetmenin yeni biçimini, masaüstü araç çubuğu ayarları, işlevsel seçenekleri kullanma vb.

İkincisi – bu kitap 1C:İşletme sistemini bilmeyenleri baz alıp hazırlanmıştır. Ayrıca daha önce belirtilmiş olan sorular, öneriler, hatalar, yorumlar da dikkate alınmıştır.

Örneğin, mevcut olan kodu analiz etmek için veya kendi kodumuzu yazmak için yardımcı olabilecek sözdizimi-yardımcısını ve farklı hata ayıklama biçimlerini hakkında önemle bilgi sunulmuştur ve bu yüzden ayrı bölüm şeklinde açıklanmıştır. Bu ise, yeni başlayan geliştiriciler için 1C:İşletme kaynak kodunu kavramada çok büyük katkı sağlayacaktır.

1C:İşletme 8.2 platform sürümünde geliştirme yapma konusunda okuyucularımıza bu kitabın çok yararlı olacağı umidi ile...

Maksim Radchenko, Elena Hrustelyova

Bu kitap kimler içindir

Kitapta, ev hizmetlerini veren bir orta boylu işletme faaliyetlerinin takibatını yapabilecek uygulama çözümü geliştirilmektedir. Bu kitabı iyi kavradığınız durumda 1C:İşletme 8 sisteminde gerçek uygulama çözümlerinin hangi mantıkla geliştirildiğini ve uygulanabilir faaliyet alanlarını öğreneceksiniz.

Niye ille ki bu örnek seçildi?

İlk olarak – hizmet verme sektörü iyi bildiğimiz sektördür. O veya farklı biçimde mutlaka hizmet verme konusu ile mutlaka karşılaşmışızdır. Ev tekniklerini tamir etme, oto servis, kuru yıkama, kuaför, kozmetik hizmetler vb.

İkinci olarak da – bu işletme faaliyetiniz örnek alınması bu işletmede farklı biçimde hizmet sunulmakta, personel maaşları hesaplanmakta, hizmet verme sürecinde harcanan malzemeler hesaplanmakta, işletme stok takibi yapılmakta. Hatta muhasebe hesabını tutma olanağı da mevcuttur. Farklı biçimlerde raporlar istenmekte. Genel olarak söylemek gerekirse, 1C:İşletme 8.2 platform sürümünde farklı işlemlerin yapılabildiğini göstermektedir.

Kitap daha çok yeni başlayan geliştiriciler düşünüülerek hazırlanmıştır.

Eğer Siz 1C:İşletme sistemi ile daha tanışmamışsanız ve bu sisteme bir şeyler yapabilmeyi çok istiyorsanız – bu kitap sırif sizin için hazırlanmış sayabilirsiniz. Kitabin asıl amacı – sizin elinizden tutarak 1C:İşletme sisteminde uygulama çözümü geliştirmenin zor olmadığını göstermektedir.

Bu kitap 1C:İşletme sisteminde belli tecrübe sahip uzmanlara faydalı olacaktır, unutukları konu varsa hatırlatacak veya bazı alanları daha da detaylı biçimde öğretecektir.

Bu kitap nasıl okunur

Bu kitap oldukça okul ders kitaplarına benzettimmiştir ve ayrı bölümler halinde hazırlanmıştır. Her bölüm ayrı küçük

Mantıksal açıdan bakıldığından her bir bölüm uygulama çözümünün hazır bölümü olarak hesaplanmaktadır. Gerçekleşme sürelerine göre bölümler birbirinden farklı olsalar bile, mümkünse bölümler başlandığından itibaren fazla büyük ara verilmeden bitirilmesi önerilir. Aksi durumda yeni başlayanlar için konunun ortasından devam etmek ve eski bilgileri hatırlamak biraz zor olacaktır.

Bölümler kolay seviyeden zora doğru ilerlemektedir. Bölümler 1C:İşletme sisteminin geliştirme araçlarının farklı alanlarını içermektedir ve ana kullanım biçimlerini göstermektedirler.

10 1C: İşletme 8. Uygulama geliştirme kılavuzu

Kitapta birçok resim ve kaynak kodu örnekleri mevcuttur. Geliştiricimiz daha kolay anlayabilsin diye bu örnekleri kitapta da gösterdik. Eğer okuyucumuz için bu örnekler gereksiz veya fazla karışık gelirse atlamasını önerilir.

Teorik bilgi olarak sunulan bilgileri okuyucumuz isterse geçebilir, çünkü kitabı geliştirdiğimiz uygulama örneğine etki etmezler.

DERS 1

Veritabanı oluşturma ve genel detayları

Süre

Dersin tahmin süresi – 40 dakika

Programlama mı geliştirme mi?	12
Sistem hakkında genel bilgi	12
Konfigürasyon ve uygulama çözümü	13
Sistem çalışma ortamı	14
Yeni veritabanı oluşturmak	15
Tasarımcı ortamında	19
Tasarımcı hakkında genel bilgi	19
Konfigürasyon nesne ağacı	20
Konfigürasyon nesneleri nedir	21
Özellik panosu	22
1C: İşletme ortamında hata ayıklamasını başlatma	25
1C: İşletme ortamında	27
Uygulama çözümünün arayüz görünümü	28

12 1C:İşletme 8. Uygulama geliştirme kılavuzu

Bizim ilk dersimiz 1C:İşletme 8 sistemi ile ve geliştirme uzmanının ana aracı olan, Tasarımcı ile bir tanışma dersi olacaktır.

Platform, konfigürasyon ve uygulama terimlerinin ne anlama geldiğini öğreneceksiniz. Ve 1C:İşletme 8 sisteminin farklı çalışma ortamları ile de tanışacaksınız.

Konfigürasyon nesnesi nedir, yeni nesne nasıl oluşturulur ve oluşturulan nesnenin parametrelerinin nasıl belirleneceğini de göreceksiniz.

Dersin sonunda yapacağımız geliştirmeler için yeni boş bir veritabanı oluşturacaksınız.

Programlama mı geliştirme mi?

Ben ne yapıyorum? 1C:İşletme üzerinde geliştirmeye ilgi gösteren neredeyse herkes zaman zaman böyle bir soru ile karşılaşır.

“Program yazıyorum”, - bu en çok rastlanan cevap. “Ne ile yazıyorsun?” – “1C’de yazıyorum”. “Hangi programda çalışıyorsunuz?” – “1C’de çalışıyoruz”. “Bu program hangi ortamda yazıldı?” – “1C’de yazıldı”. “1C bilgisi olan muhasebeci işe alınacaktır”, “yarı zamanlı 1C programcısı işe alınacaktır” vs.

Bu tür cümlelere sık rastlanabilir. Bilgisi olmayan kişi için bunlar bir şey ifade etmez. Ama 1C:İşletme sisteminde geliştirme hakkında bilgisi olan kişi için bu tür sorulara kolay cevap vermek zor olabilir. Çünkü burada “1C” olarak farklı şeyler belirlenmektedir, *program* kelimesi ise tamamen akıları karıştırabilir...

Hedefiniz “1C’de programlamayı” öğrenmek ise, bu hedef tam olarak doğru değildir. 1C:İşletme 8 sisteminde kaynak kodu mevcuttur, fakat geliştirme sürecinde bu kaynak kod ana geliştirme aracı değildir. Bu kitap herkesin anladığı anlamda “programlamayı” öğretmez. Bu kitap *1C:İşletme 8 platform üzerinde ticari uygulama çözümleri geliştirmeyi* öğretir. Bu süreçte tabii ki “programlama” mevcuttur fakat sadece geliştirme araçlarından sadece bir tanesi olarak.

Bunu 1C:İşletme’de ilk adımlarınızı atmadan önce anlamak önemlidir.

Bu kitapta ne yapacağımızı daha iyi anlamak için ilk önce 1C:İşletme 8’in tam olarak nasıl bir şey olduğunu anlatalım.

Sistem hakkında genel bilgi

1C:İşletme, bir işletmenin iktisadi ve yönetimsel faaliyetinin otomasyonu sağlayan universal bir sistemdir. Faaliyetler farklı olabileceği için, 1C:İşletme de çeşitli faaliyetlere göre uyum sağlayabilmektedir. Böyle bir özellik tanımlamak için “tasarlanabilirlik” terimi kullanılır, yani sistem belli bir işletmenin ihtiyaçlarına ve farklı iş görevlerine göre ayarlanabilmektedir.

1C: İşletme sistemi standart özellik takımı olan paket bir program olmayıp, kullanıcı ve geliştirici tarafından kullanılan çeşitli program araçlarından ibaret olduğundan böyle bir özelliğe sahip olabilmektedir. Sistem, bir biriyle sıkı çalışan iki parçadan oluşturulmaktadır; *konfigürasyon* ve konfigürasyonu işleyen *platform*.

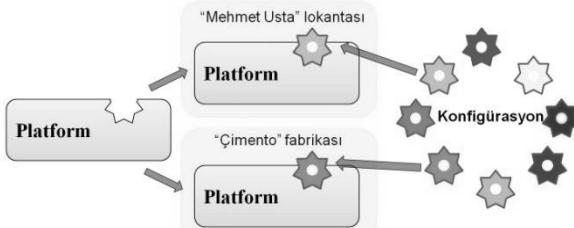
Bu sistem parçalarının etkileşimini daha açık anlamak için CD çalar sistemi ile karşılaştırma yapalım. Bildığınız gibi CD çalar müzik dinlemek için kullanılmaktadır. Farklı insanlar farklı müzik sever, dolayısıyla farklı müziği içeren birçok çeşit CD vardır.

Bir şarkıyı dinlemek için CD'yi CD çalara yerleştirmek ve oynatmak gereklidir. Bunun dışında çağdaş CD çalarlar sevdığınız şarkıları CD'ye yazma imkanı verir, yani yeni bir CD oluşturabilir.

CD çalar kendisi CD olmadan bir işe yaramaz, aynı şekilde CD kendisi CD çalarsız bize, çay fincan altlığı olarak kullanılması dışında, herhangi bir fayda getirmez.

1C: İşletme sisteminde ise platform CD çalar gibi, konfigürasyon da CD olarak düşünülebilir. Platform konfigürasyon çalışmasını sağlar ve konfigürasyonu değiştirmeye ve sıfırdan tasarlamaya imkan verir.

Tek bir platform var (1C: İşletme) ve çok sayıda konfigürasyon bulunur. Belli bir uygulama çözümünün çalışması için platform ve konfigürasyonlardan bir tanesi gereklidir.



Resim. 1.1. Çok sayıda konfigürasyon var, platform tektir.

Platform, tek başına herhangi bir bilgi işlem görevlerini yerine getiremez, çünkü platform konfigürasyon çalıştmak için tasarlanmıştır. Aynı şekilde konfigürasyon; amaçladığı görevleri yerine getirmek için onu yönetecek platforma gerek vardır.

Konfigürasyon ve uygulama çözümü

Evet, şimdi biz önceki kısımda sorulan soruya cevap verebiliriz; bu kitabı okuma ve demo örneğini geliştirme sürecinde biz *konfigürasyon* geliştireceğiz.

Burada ilerde kullanılacak olan terminolojinin iki yüzünden bahsetmek gereklidir. Aynı nesneyi ifade etmek için iki farklı terim kullanılacaktır; *konfigürasyon* ve *uygulama çözümü*.

14 1C:İşletme 8. Uygulama geliştirme kılavuzu

Bu terimler, platformun çalıştığı ve kullanıcılar tarafından gördüğü ve kullandığı 1C:İşletme kısmını belirler. Bazı kullanıcılar platformun araçları ile çalışırlar, fakat bu kullanıcılar ilerlemiş kullanıcılardır. Bu veya şu terimin kullanılması bulunduğu cümlenin manasına bağlıdır.

Geliştiricinin yaptığı hareketlerden bahsediliyorsa, o zaman konfigürasyon kullanılır, çünkü bu 1C:İşletme'nin kesin terimidir. Uygulama çözümü terimi, 1C:İşletme kullanıcılarının tarafından kullandığı ve anlaşıldığı terimdir.

Bilgi işlem otomasyon görevleri farklı olduğu için “1TC” şirketi ve bayileri belli bir faaliyet sektörüne hitap edeceği uygulama çözümleri geliştiriyorlar. Mevcut olan uygulama çözüm örneği olarak aşağıdaki standart uygulamalar verilebilir;

- 1TC:Ticari KOBİ Yönetimi
- 1TC:Muhasebe
- 1TC:Fatura

Bunun dışında diğer standart uygulamalar da var. Standart uygulamalar ile ilgili daha detaylı bilgiye <http://1tc.com.tr/produkts> adresinden ulaşabilirsiniz.

Standart uygulamalar üniversaldır ve aynı faaliyet sektöründe çalışan farklı işletmelerin ihtiyaçlarını karşılayabilir. Bu güzeldir.

Diger taraftan böyle bir universallık, bazı işletmelerde programın tüm özelliklerinin kullanılmadığına veya bazı fonksiyonunun eksik olmasına yol açar (herkes için tam uygun bir program yapılamaz).

Burada ise sistemin *tasarlanabilirliği* ön plana çıkar, çünkü platformun konfigürasyon çalışma araçları dışında, kullanılan konfigürasyona değişiklikleri ekleme olanağını sağlayan geliştirme araçlarına da sahiptir. Üstelik platform, belli bir sebeplerden mevcut olan konfigürasyon kullanılamıyorsa, sıfırdan yeni bir konfigürasyonu geliştirme olanağını sağlar.

Böylece, CD çalar ile karşılaşmasına dönersek, CD'ye yazılmış olan şarkıları isteğe göre değiştirebiliriz ve kendi şarkılarımızı içeren yeni CD oluşturabiliriz. Bu sırada herhangi bir müzik aleti kullanmamıza gerek yok, müzik oluşturmak için tüm gerekli araçlar CD çalarımızda mevcuttur.

Sistem çalışma ortamı

1C:İşletme sisteminin iki tane çalışma ortamı vardır: *1C:İşletme* ve *Tasarımcı*.

1C:İşletme ortamı kullanıcıların sistemde çalışmasını sağlar. Bu ortamda kullanıcılar veri girer, işler ve sonuç veri alırlar.

Tasarımcı ortamı veritabanı geliştiricileri ve sistem yöneticiler tarafından kullanılır. Bu ortam var olan bir konfigürasyonu değiştirme veya sıfırdan

yeniden konfigürasyonu geliştirme olanağını sağlayan geliştirme araçlarını içerir.

Kitabımızın amacı, size yeni konfigürasyonları geliştirmesi ve var olan konfigürasyonları değiştirmesi ile ilgili bilgi vermek olduğundan, kitabın sonraki sayfalarında genelde Tasarımcı ortamında çalışacağız. Ve bazen, yaptığımız fonksiyonları kontrol etmek için sistemi 1C:İşletme ortamında çalıştıracağız.

Bilgisayarınızda 1C:İşletme 8 sistemi kurulu değilse, burada kurulum yapmanız tam zamanı, çünkü artık program ile çalışmaya başlayacağız.

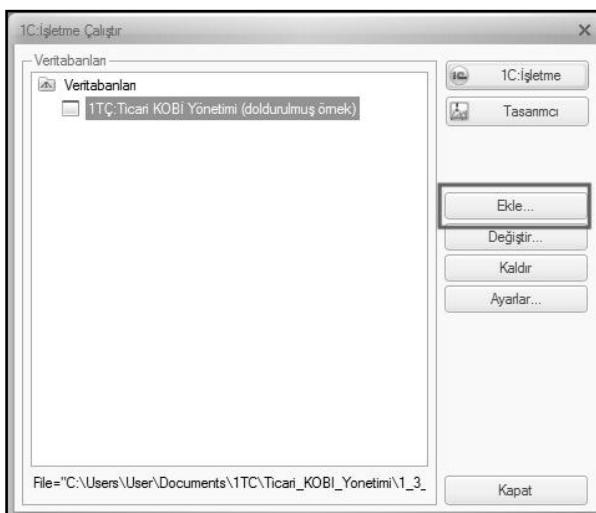
Yeni veritabanı oluşturmak

1C:İşletme 8 sistem kurulumu ile ilgili herhangi bir sorunun çekmaması gereklidir. Kurulum ile ilgili detaylı bilgiler için “1C:İşletme 8. Sistem yönetici kılavuzu” kitabına bkz.

Aynı şekilde sistemi çalıştırması ve boş konfigürasyonu içeren yeni veritabanı oluşturulması ile ilgili de sorun çekmaması gereklidir.

Dikkat! Bu kitapta anlatılan örneği geliştirmek için, şablondan oluşturulan veritabanı değil, yeni konfigürasyon geliştirmek için veritabanı gereklidir. Bunun için aşağıdaki eylemleri yerine getirmeniz gereklidir.

1C:İşletme'yi çalıştırın. Açılan pencerede çalıştığınız veritabanı listesini görürsünüz. Liste boş ise sistem otomatik olarak yeni veritabanı oluşturmayı önerir. Veritabanı listesinde herhangi bir veritabanı bulunduğu durumda yeni veritabanı oluşturmak için **Ekle** butonuna basmak gereklidir (Resim 1.2).



Resim 1.2. Yeni veritabanı oluşturma Aşama 1

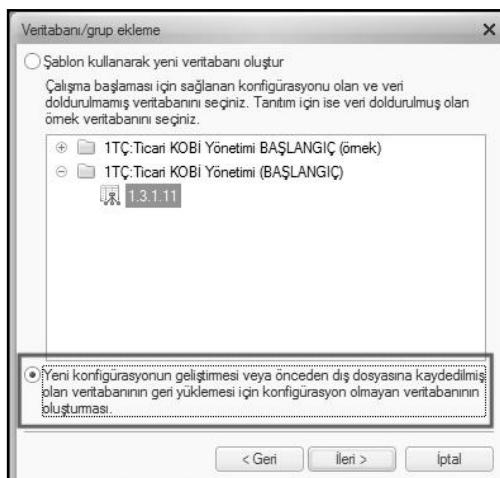
16 1C:İşletme 8. Uygulama geliştirme kılavuzu

Açılan pencerede Yeni veritabanı Oluştur maddesini seçiniz (Resim 1.3).



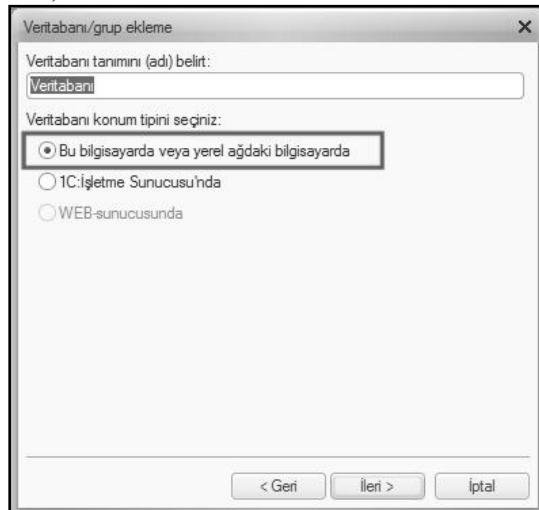
Resim 1.3. Yeni veritabanı oluşturma Aşama 2

İleri butonuna basınız. Bir sonraki aşamada Konfigürasyon olmayan veritabanı oluşturma seçeneğini seçiniz (Resim 1.4).



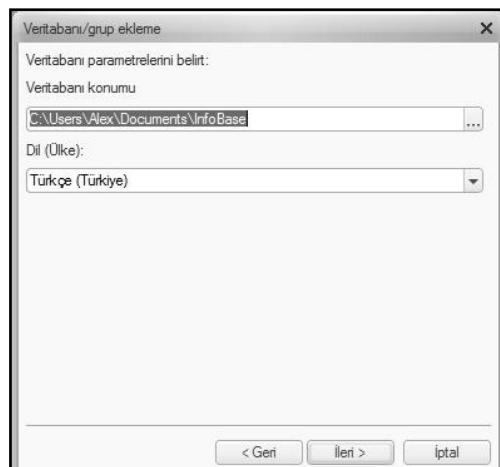
Resim 1.4. Yeni veritabanı oluşturma Aşama 3

İleri butonuna basınız. Bir sonraki aşamada veritabanı ismi giriniz ve konumu tipi olarak **Bu bilgisayarda veya yerel ağdaki bilgisayarda...** seçiniz (Resim 1.5).



Resim 1.5. Yeni veritabanı oluşturma Aşama 4

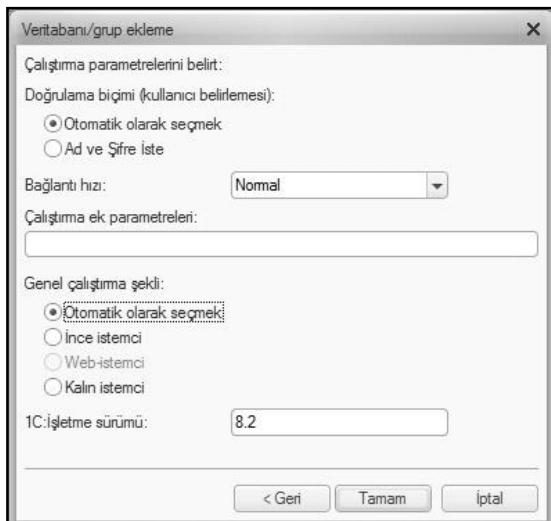
İleri butonuna basınız. Bir sonraki aşamada veritabanının konumlanacağı dizini belirtiniz. Varsayılan olarak Dil (Ülke) parametresi Türkçe (Türkiye) olarak işaretlenmiştir (Resim 1.6).



Resim 1.6. Yeni veritabanı oluşturma Aşama 5

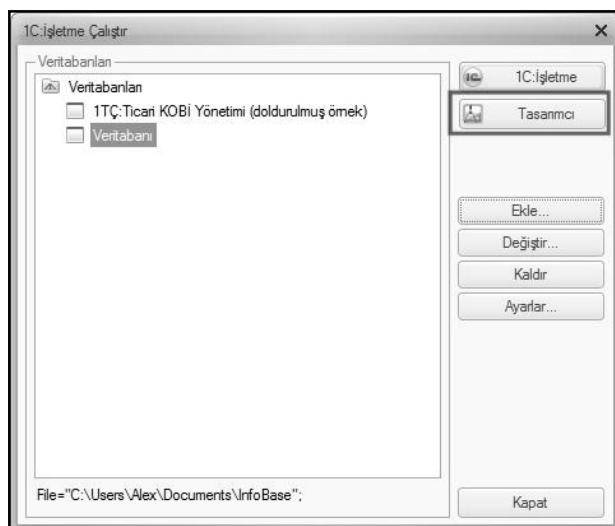
18 1C:İşletme 8. Uygulama geliştirme kılavuzu

İleri butonuna basınız. Bir sonraki aşamada **Tamam** butonuna basınız (Resim 1.7).



Resim 1.7. Yeni veritabanı oluşturma Aşama 6

1C:İşletme çalışma penceresinde, veritabanı liste bölümünde oluşturduğunuz boş veritabanı bulunacaktır (Resim 1.8).

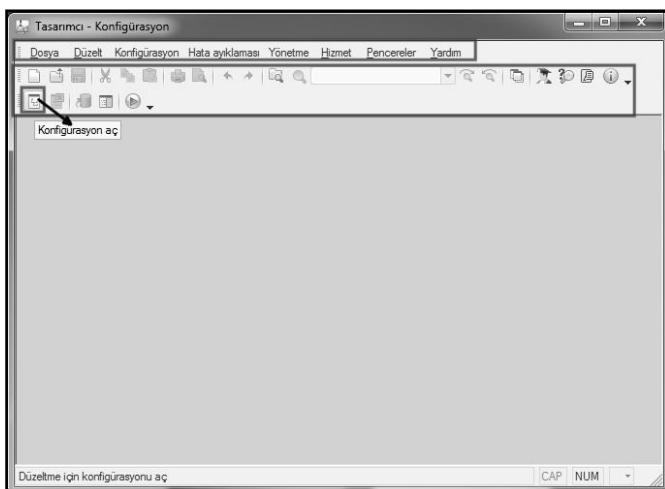


Resim 1.8 1C:İşletme sisteminin Tasarımcı ortamında çalıştırması

Tasarımcı ortamında

Tasarımcı hakkında genel bilgi

1C:İşletme sisteminin Tasarımcı ortamında çalıştırılacak. Bunu yapmak için sistem çalışma penceresinde **Tasarımcı** butonuna tıklayalım
Ekranınızda tasarımcı penceresi (Resim 1.9).



Resim 1.9. Tasarımcı penceresi

Bu aracı kullanarak konfigürasyonumuzu geliştireceğiz. Başlığın hemen altında tasarımcının genel menü bulunur. Genel menü Dosya, Düzelt, Konfigürasyon, Yönetme vs. maddeleri içerir. Her madde altında alt madde listesi bulunmaktadır. Bu alt maddeler tasarımcının farklı araçlarına ulaşmasını sağlar.

Genel menü altında tasarımcının araç çubuğu bulunur. Bu araç çubüğunda küçük resim şeklinde en sık kullanılan genel menü eylemleri yerleştirilmiştir.

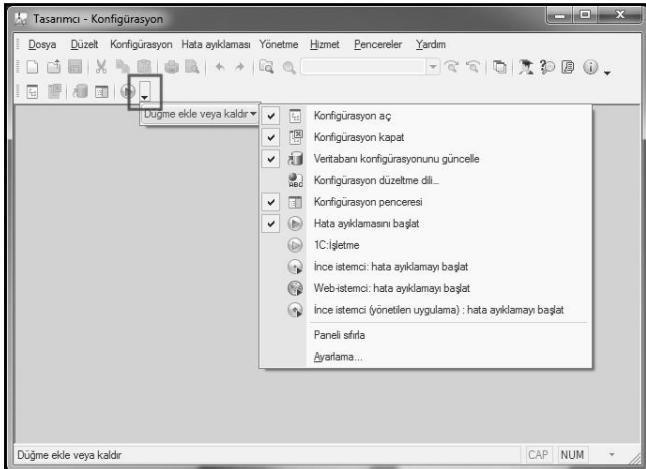
Böylece aynı işlem iki yoldan yapılabilir: menü kullanarak belli bir madde seçmek veya araç çubüğunda ilgili butona tıklamak.

Çok sayıda bulunan buton resimleri başlayan geliştiriciyi şaşırtabilir. Korkmaya gerek yok, zamanla onlara alışır, onları rahatça kullanabileceksiniz. Fare imlecini herhangi bir butona yerleştirdiğinizde, biraz bekleyiniz, böylece buton amacını açıklayan ipucu metni ekrana gelecektir (resim 1.9).

Belki ilk etapta menü kullanırsınız, fakat zamanla araç çubuğu kullanmaya başlaysınız. Çünkü araç çubuğu kullanmak daha rahattır. Zamanla, gerekli

20 1C:İşletme 8. Uygulama geliştirme kılavuzu

butonları ekleyerek ve gereksiz olanları kaldırarak araç çubuğunu kendine özel oluşturabileceksiniz (Resim 1.10).

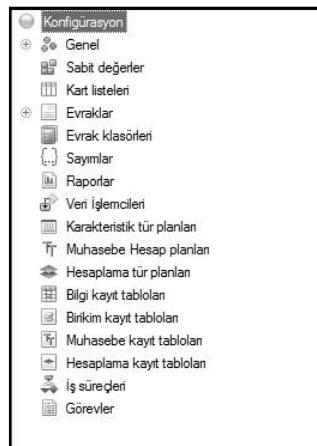


Resim 1.10. Tasarımcı araç çubuğunun ayarlaması

Konfigürasyon nesne ağaçısı

Konfigürasyon ile çalışmaya başlamak için **Konfigürasyon – Konfigürasyon aç** menüsünü veya ilgili butonu kullanarak konfigürasyonu açalım (Resim 1.9).

Ekrانınızda *konfigürasyon nesne ağaçısı* açılır (Resim 1.11).



Resim 1.11. Konfigürasyon ağaçısı

Konfigürasyon ağaçları – uygulama geliştiricinin çalıştığı genel araçlardan bir tanesidir. Konfigürasyon ağaçları, konfigürasyon yapısı ile ilgili neredeyse tüm bilgiyi içerir.

Muhtemelen aklınıza bir soru gelmiştir; biz bir şey oluşturmadık, fakat konfigürasyon ağaçında bir şeyler bulunur?

Geliştiricinin işini kolaylaştırmak için, konfigürasyonda bulunan her şey gruplandırılmış ve şu anda konfigürasyon ağaçları grupperi gösteriyor. Eğer ağaçta + simgelerine tıklarsanız, hiçbir grupta öğe olmayacağına görersünüz.

Bir şey yapmaya artık başlayabiliriz, fakat başlamadan önce bazı terimleri netleştirmek gereklidir. Konfigürasyon hakkında bahsederken herhangi bir terim kullanmadığımızı fark etmişsinizdir, biz bilerek terimleri kullanmadık. Fakat şimdi terminolojiyi belirtme ve *konfigürasyon nesneleri* hakkında anlatım zamanı geldi.

Konfigürasyon nesneleri nedir

Konfigürasyon bu bir tanımlamadır. Konfigürasyon, 1C:İşletme ortamında kullanıcıların kullanacağı veri yapısını tanımlamaktadır.

Bunun dışında konfigürasyon olabilecek veri işleme algoritmalarını tanımlar, ekranda veya çıktısında bu verinin ne şekilde gösterileceği ile ilgili bilgi içerir vs. İlerde 1C:İşletme platformu bu tanımlama bazında gerekli yapıya sahip olan veritabanını oluşturur ve kullanıcıya bu veritabanı ile çalışma olanağını sağlar.

1C:İşletme sistemini belli bir görevler için hızlı ve kolay bir şekilde ayarlayabilmek için, konfigürasyonda bulunan tanımlama belli bir birimlerden ibarettir, bu birimlere konfigürasyon nesneleri denir. Belki de, her konfigürasyon nesnesi ile ilgili kısa bilgiyi içeren, “1C:İşletme 8.2. Geliştirici Kılavuzu” kitabına göz atmışsınız.

Bu kitapta biz bu bilgiyi tekrarlamayacağız, çünkü bu kitabın amacı 1C:İşletme sistemin mantığını, nesne dayalı terminolojini kullanarak teknik iç yapı olarak anlatmak değil, daha çok 1C:İşletme imkanlarını doğru şekilde kullanmayı öğretmektedir.

Bu yüzden konfigürasyon nesnelerini biz “basit” şeklinde anlatacağız. Fakat bu, nesnelerin amacı daha net bir şekilde anlamamanızı sağlar.

Konfigürasyon nesneleri bloklu çocuk oyuncaklarına benzer. Genelde böyle tip oyuncaklarında farklı çeşitleri bulunur. Parçalar farklı olabilir; uzun, kısa, kare, dörtgen vs. Şimdi, her parçadan gerekli sayıda oluşturabildiğimizi düşünün (örneğin 5 adet uzun ve 3 adet kısa). Onları birbirile farklı biçimde bağlayabiliriz.

Aynı şey konfigürasyon nesneleri için geçerlidir. Biz sadece belli türü olan nesneleri oluşturabiliriz. Fakat her türden gerektiği kadar oluşturabiliriz. Bir tür nesneler diğer tür nesnelerden farklı özelliklere (daha doğrusu özellik

takımına) sahip oldukları için farklı olur. Nesneler bir biriyle etkileşim sağlayabilirler ve bu etkileşimi biz tanımlayabilmektediriz.

Konfigürasyon nesnelerinin, bloklu çocuk oyuncası ile diğer benzer tarafı nedir? Bloklu çocuk oyuncasında birbirile bağlanabilen parçalar mevcut, ve bir biriyle bağlanamayan parçalar mevcut, örneğin tekerlekler. Fakat tekerlekleri bağlayabilen dingil var, bağlandığında tekerlekler artık dönenbilir. Yani farklı parçalar, farklı çalışır.

Farklı konfigürasyon nesneleri farklı davranışa sahiptir. Davranış nesne türüne bağlıdır. Bazı nesneler belli bir fonksiyona sahiptir, bazıları bu fonksiyona sahip değil, fakat onları farklı bir özellik takımı mevcuttur.

Bir sonraki konfigürasyon nesne özelliğini araba örneğiyle anlatılabilir. Araba birçok parçadan oluşur. Parçalardan biri – motordur. Motor kendisi de birçok parçadan ibarettir. Ve farklı motorlarda aynı parçalar kullanılabilmektedir.

Aynı şekilde “kapsamlı” konfigürasyon nesneleri daha “basit” nesnelerden ibarettir. Ve motorda olduğu gibi aynı “basit” nesneler, farklı “kapsamlı” nesnenin parçası olabilir. Böyle bir yapı, konfigürasyon nesneleri ile çalışmayı ciddi bir ölçüde kolaylaştırır. Çünkü “basit” bir nesne ile nasıl çalışmak gerektiğini bildiğimiz takdirde, “basit” nesne “kapsamlı” bir nesnenin parçası olsa da, onun ile aynı şekilde çalışırız.

Ve sonunda konfigürasyon nesnelerinin en önemli özelliği – gerçekliktir. Konfigürasyon nesneleri, fonksiyonu tanımlamak için geliştiricinin kullandığı soyut bir yapılar değildir. Onlar, işletmenin gerçek hayatı kullandığı nesnelerin benzerleridir.

Örneğin, ticari olguları kaydetmek için her işletmede çeşitli evraklar kullanılır. Aynı şekilde konfigürasyonda **Evrak** türü olan nesneler mevcuttur.

Bunun dışında her işletmede personel, cari hesap veya malzeme listeleri mevcuttur. Konfigürasyonda, böyle listeleri bilgisayar ortamına aktarma imkanı sağlayan **Kart liste** türünde nesneler mevcuttur.

Dediğimiz gibi, konfigürasyon nesneleri bazında platform veritabanında veri depolanacağı tabloları oluşturur. Kitaplarda genelde konfigürasyon nesneleri ve tablo takımı aynı isimle adlandırılır.

Örneğin, konfigürasyonda *Personel* kart listesi bulunduğuanda, platform tarafından bu nesne bazında oluşturulan tablo takımına da *Personel kart listesi* denir.

Bu kitapta, konfigürasyon hakkında bahsedildiği zaman net bir tanım kullanacağımız –*Personel kart listesi konfigürasyon nesnesi*. Veritabanı hakkında bahsedildiğinde biz sadece *Personel kart listesi* diyeceğiz.

Konfigürasyon nesnesi nasıl eklenir

Yeni konfigürasyon nesneleri eklemeye başlamadan önce, dikkat edilmesi gereken bir şey var; işletme faaliyetinin otomasyonunu sağlayan kendi

konfigürasyonunu geliştirmek için geliştirici sadece platformda bulunan standart konfigürasyon nesne takımı kullanabilir. Kendi konfigürasyon nesnelerini oluşturma imkanı yoktur.

Çalışmaya başlamadan önce tasarımcıda bazı çalışma yöntemlerini anlatmak gereklidir.

Konfigürasyonu açmak veya kapatmak için **Konfigürasyon – Konfigürasyon aç** ve **Konfigürasyon kapat** menüleri veya araç çubuğundaki ilgili butonları kullanılabilir.

Konfigürasyon açıldığında konfigürasyon bileşenleri konfigürasyon nesne ağacında bulunur (Resim 1.11). Bu pencereyi, Windows'un tüm pencereleri gibi kapatabilirsiniz fakat konfigürasyon açık olacaktır (yani değiştirilebilecektir). Konfigürasyon nesne ağacını ekrana tekrar getirmek için **Konfigürasyon – Konfigürasyon penceresi** menüsü kullanılır.

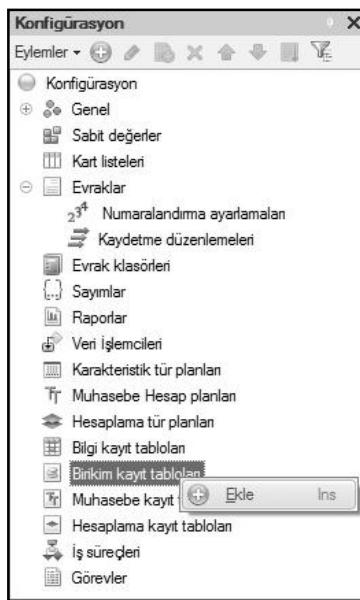
Konfigürasyon nesnesi birkaç yöntem ile eklenebilir, sizin için en rahat ve anlaşılır yöntemi kullanabilirisiniz.

Birinci yöntem. Konfigürasyon ağacında oluşturmak istenilen konfigürasyon nesne grubuna imleç yerleştirmek ve konfigürasyon penceresinin araç çubuğundaki **Eylemler – Ekle** menüsünü kullanmak gereklidir (Resim 1.12)



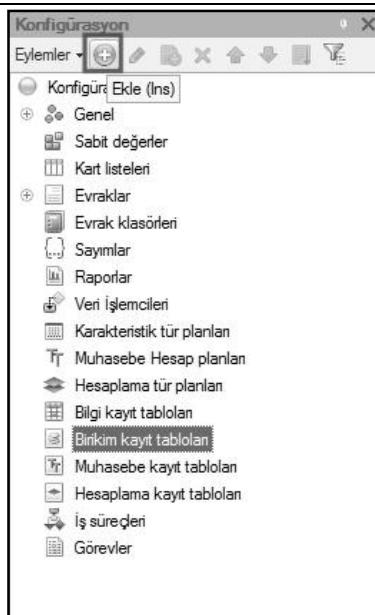
Resim 1.12. Yeni konfigürasyon nesne ekleme

İkinci yöntem. Farenin sağ butonu ile çağırılan bağlam menüsü kullanılabilir. Oluşturulmak istenilen konfigürasyon nesnesine sağ tıklayın. Ekrana gelen menüden **Ekle**'yi seçiniz (Resim 1.13).



Resim 1.13. Yeni konfigürasyon nesne ekleme

Üçüncü yöntem. Konfigürasyon ağacında eklemek istediğiniz nesneye imleci yerleştirip konfigürasyon penceresinin araç çubuğundan **Ekle** butonuna basınız (+ simgesi olan) Resim 1.14.



Resim 1.14. Yeni konfigürasyon nesne ekleme

Son yöntem daha kullanışlı geliyor, dolayısıyla biz genelde onu kullanacağız.

Özellik panosu

İşte başlıyoruz!

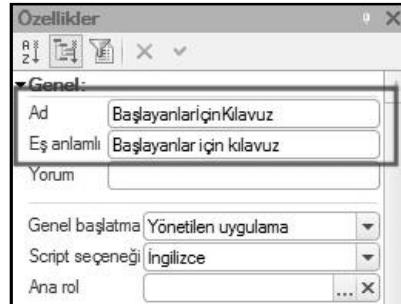
Konfigürasyon adı belirleyelim. Geliştirici, oluşturulan konfigürasyon nesne özelliklerini özellik panosunda belirleyebilir.

Özellik panosu – Konfigürasyon nesnesini tüm özelliklerini ve onun ile bağlı olan tüm veriyi tanımlama ve düzeltme imkanı sağlayan bir hizmet penceresidir. Farklı konfigürasyon nesneleri farklı özelliklere sahip oldukları için, özellik panosu da aktif olan konfigürasyon nesnesine göre (imleç hangi nesnede yerleştirildiyse) içeriğini değiştirecektir.

Konfigürasyon nesne ağacında kök **Konfigürasyon** ögesini işaretleyelim ve üzerinde çift tıklayarak özellik panosunu açalım.

Konfigürasyon adı belirleyelim; *Başlayanlar İçin Kılavuz*.

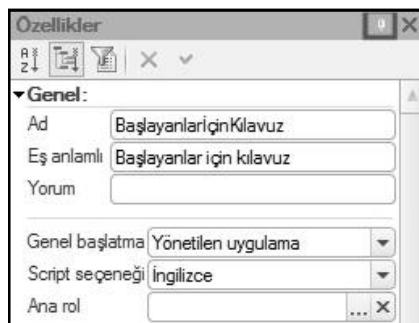
Eş anlamlı alanı otomatik olarak doldurulacaktır, fakat gerektiğinde serbest şekilde isteğe göre değiştirilebilir. 1C:İşletme ortamında eş anlamlı yansıtılacaktır (Resim 1.15),



Resim 1.15. Konfigürasyon özellik panosu

Geliştirici bazı eylemleri yaparken özellik panosu otomatik olarak açılır. Fakat geliştirici özellik panosunu her zaman sağ tuş - **Özellikler** menüsünü kullanarak özellik panosunu ekrana getirebilir.

Şu an gibi durumlarda özellik panosu açılır ve tasarımcı çalışma alanında sabitlenecektir. Yani herhangi bir konfigürasyon nesnesi işaretlendiğinde özellik panosu her zaman açık olacaktır. Fakat özellik pano başlığında bulunan butonu kullanarak, özellik panosunu “çözelim” olanağı mevcuttur. (Resim 1.16).



Resim 1.16. Özellik panosunu “çözelim”

Bu durumda fare imleci diğer pencereye yerleştiğinde özellik panosu ek çubuğu küçültecektir (Resim 1.17).



Resim 1.17. Ek çubuğundaki buton

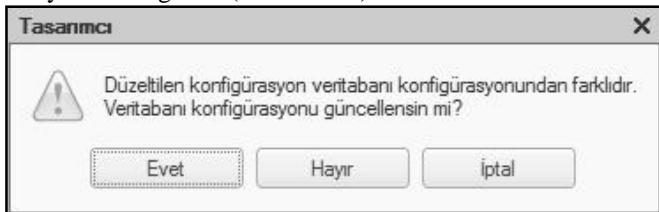
Fare imleci küçültülmüş özellik pano simgesine yerleştiğinde özellik pano penceresi açılacaktır.

Saklanabilir, sabitlenmiş vs. özelliklerine sadece özellik pano penceresi değil, tasarımcının diğer pencereleri de sahiptir. Mesela konfigürasyon penceresi.

1C: İşletme ortamında hata ayıklamasını başlatma

Şimdi 1C:İşletme ortamında yapmış olduğumuz ilk değişiklikleri kontrol edelim.

Bunu yapmak için Hata ayıklaması – Hata ayıklamasını başlat menüsü veya tasarımcı araç çubuğundan ilgili buton kullanalım. Sistem otomatik olarak değişiklik var olup olmadığını kontrol eder ve veritabanı güncellemesi ile ilgili soruyu ekrana getirir (Resim 1.18).



Resim 1.18. Konfigürasyon güncelleme sorusu

Şu anda niye böyle olduğunu açıklamayacağız, bununla ilgili daha detaylı “Genel konfigürasyon ve veritabanı konfigürasyonu” bölümünde anlatılacaktır.

Tasarımcının ekrana getirdiği pencerede Evet diyelim ve ekrana 1C:İşletme penceresi gelir (Resim 1.19).

1C:İşletme ortamında

Uygulama çözümünün arayüz görünümü

Pencere başlığında konfigürasyonumuzun adını görüyoruz. Boş alan bu uygulamanın *çalışma alanı* ve şu anda çalışma alanı boştur.



Resim 1.19. 1C:İşletme

Başlık dışında 1C:İşletme penceresinde hiç bir şey yoktur. Bu normal.

Biz daha hiçbir konfigürasyon nesnesi oluşturmadık ve konfigürasyon nesneleri bulunan alt sistemleri tanımlamadık.

1C:İşletme arayüzü temeli olan alt sistemler bir sonraki bölümde anlatılacaktır.

Sorular

- ✓ *1C:İşletme tasarılanabilirlik nedir?*
- ✓ *Sistem hangi ana parçalardan ibarettir?*
- ✓ *Platform ve konfigürasyon nedir?*
- ✓ *1C:İşletme sisteminin farkı çalışma biçimleri ne için kullanılır?*
- ✓ *Konfigürasyon nesne ağacı nedir?*
- ✓ *Konfigürasyon nesneleri nedir?*
- ✓ *Sistem, konfigürasyon nesneleribazında ne oluşturur?*
- ✓ *Yeni konfigürasyon nesnesi nasıl oluşturulur?*
- ✓ *Özellik panosu ne için kullanılır?*
- ✓ *1C:İşletme sistemini hata ayıklama biçiminde nasıl çalıştırılır?*

DERS 2

Alt Sistemler

Süre

Dersin tahmin süresi – 45 dakika

Alt sistem nedir?	32
Alt sistem ekleme	34
Konfigürasyon nesne adı ve eş anlamlısı	35
Alt sistem resmi	36
Uygulama çözümünün bölümler çubuğu	40
Bölüm sırası	41
Sorular	45

Bu bölümde biz, 1C:İşletme arayüz yapısının temeli olan alt sistemler ile tanışacağız.

Geliştirilen uygulamanın bölüm yapısını tanımlayacak olan birkaç alt sistemi oluşturacağız, alt sistem görünümünü ayarlayacağız ve uygulama arayüzünde alt sistem sırasını belirleyeceğiz.

Alt sistem nedir?

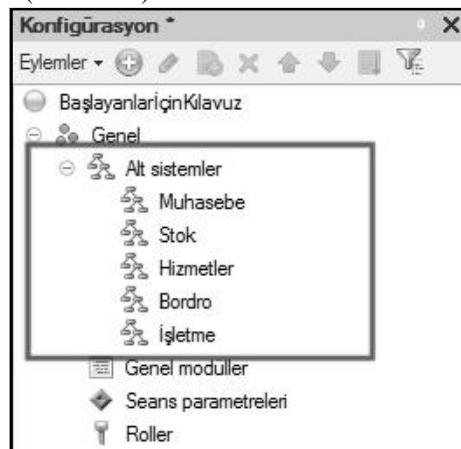
Alt sistem – 1C:İşletme arayüzü oluşturmak için ana öğelerdir. Bu yüzden uygulamayı geliştirmeye başlamadan önce alt sistem yapısını belirtmek gereklidir.

Bu sırada geliştirici için alt sistem yapısını iyi düşünmek ve sonra oluşturulan konfigürasyon nesnelerini alt sistemlere dikkatlice ve doğru bir şekilde bağlamak ciddi ölçüde önemlidir.

Basit uygulamalarda alt sistemlerin kullanılması şart değil, fakat bizim örneğimiz basit olmadığı için alt sistemler kullanılacaktır.

Alt sistem konfigürasyonda işlevsel bölümleri belirtme olanağını sağlar. Uygulama çözümünde de bu işlevsel bölmelere göre arayüz tasarılanacaktır.

Alt sistem öğeleri **Genel** grubu altında bulunur. Alt sistemler hiyerarşik şekilde oluşturulabilir; daha düşük seviye alt sistemler üst alt sistemler altında toplanabilir (Resim 2.1).



Resim 2.1. Alt sistem yapısı

Üst seviye alt sistemler arayüzün genel öğeleridir, çünkü onlar uygulama çözümünün genel bölümlerini tanımlamaktadır.



Resim 2.2. Uygulama çözümü bölümleri

Aynı konfigürasyon nesnesi yansıtılacağı birden fazla alt sisteme dahil edilebilir.

Alt sistemler ile, rollere göre görünürlüğü belirleyerek kullanıcıya gereksiz öğeleri içermeyen, kolay kullanılan, ergonomik arayüz tasarlama olanağı vardır. Örneğin, stok sorumlusu ürünleri stoka alma ve çıkartma işlemleri yapmalı ve onun için muhasebe veya verilen hizmetler ile ilgili her şey görmesine gerek yoktur.

Böylece, alt sistemler uygulama çözümünün yapısını belirler, kullanıcı arayüzü ayarlar, farklı evrakları, kullanıcının daha rahat çalışması için ve gerekli verilere daha kolay ulaşması için kart listeleri ve raporları belli bir mantığa göre düzenler. Bu sırada belli bir kullanıcı sadece onun işi ile alakalı bölüm ve nesnelere ulaşabilecektir.

Bizim uygulama çözümümüz basit olmasına rağmen, bazı işlevsel bölümler ayırilabiliriz.

Örneğin, muhasebe ile ilgili tüm fonksiyonları ayrı bir alt sistem olarak ayrılabilir.

Bunun dışında personel maaşları ile ilgili her şey ayrı bölüm olarak tanımlanabilir.

“Aspirin usta” Ltd. Şti’mizin faaliyeti iki bölüm olarak ayrılabilir: stok takibi ve hizmetler.

Bunun dışında veritabanı ile bazı yönetimsel işlemleri gerçekleştirmek için ayrı bir alt sisteme oluşturacağız ve bu alt sisteme sadece sistem yönetici izinleri olan kullanıcılar ulaşabilecekler.

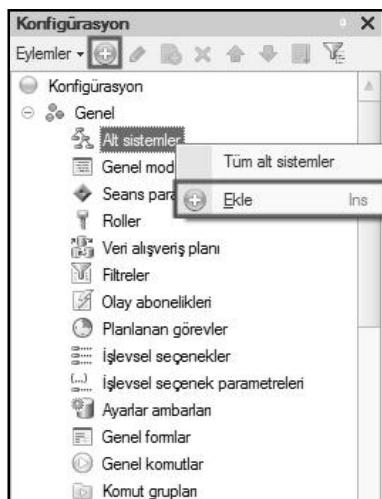
Dolayısıyla konfigürasyonumuzda biz şu anda beş tane alt sistem konfigürasyon nesneleri oluşturacağız: *Muhasebe*, *Bordro*, *Stok*, *Hizmetler* ve *İşletme*. Bunu yapmak için aşağıdaki adımları izleyelim.

Alt sistem ekleme

Tasarımcı ortamında

Uygulamayı kapatalım ve tasarımcıya geri dönelim. Alt sistemi oluşturmak için konfigürasyon nesne ağacında + simgesine tıklayarak **Genel** grubu açalım.

Alt sistemler grubunu işaretleyip sağ tuşlayalım ve açılan bağlam menüsünden **Ekle** butonu veya konfigürasyon penceresinin araç çubuğunda ilgili butona basalım (Resim 2.3).



Resim 2.3. Konfigürasyon nesne ağacına yeni alt sistem ekleme

Sistem konfigürasyon nesne düzeltme penceresini açar.

Bu pencere kapsamlı konfigürasyon nesneleri için tasarlanmıştır ve sırayla gerekli eylemleri uygulayarak hızlı bir şekilde konfigürasyon nesnesini oluşturmaya imkan sağlar.

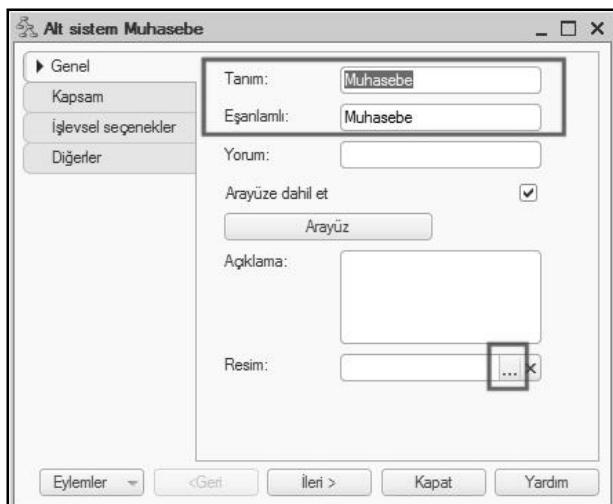
Doğru eylem sırasını takip etmek için pencerenin alt kısmında *İleri* ve *Geri* butonları bulunur. *İleri* butonu sırayla gerekli bilgileri atlamadan veri eklemeye olanak sağlar. Önceki sekmelerde girilen veri eksik veya hatalı olduğunda *Geri* butonu ile önceki sayfalara geçiş yapılmaktadır. İllerde gerekli sekmeye geçiş yaparak nesne özelliklerini tanımlayabileceksiniz.

Konfigürasyon nesne düzeltme penceresi açıldığında otomatik olarak **Genel** sekmesi açılır.

Not

Nesne özelliklerini değiştirmek için çoğu zaman konfigürasyon nesne düzeltme penceresini tekrar açmak gereklidir. Bunun için konfigürasyon ağacında gerekli nesneyi işaretleyip konfigürasyon penceresinin araç çubuğuunda *Cari öğeyi değiştir (F2)* butona basmak veya nesne üzerinde çift tıklamak gereklidir.

Alt sistem adı belirtelim – **Muhasebe**. Adı üzerinde platform otomatik olarak *Muhasebe* eş anlamlısını oluşturur.



Resim 2.4. Alt sistem ad ve eş anlamlı tanımlama

Konfigürasyon nesne adı ve eş anlamlısı

Ad, tüm konfigürasyon nesnelerinin ana özelliğidir. Yeni nesne oluşturulduğunda sistem otomatik olarak nesneye belli bir ad verir.

Sistemin verdiği ad kullanılabilir, fakat daha anlaşılır bir ad ile değiştirmek mümkündür. Herhangi bir ad tanımlanabilir, ad harf ile başlamalı ve bazı imla işaretleri ve özel semboller kullanılmamalıdır (örneğin, boşluk).

Konfigürasyon içeriğini daha rahat bir şekilde okuyabilmek için nesnelere anlaşılır net bir isimler tanımlanır ve ad birden fazla kelimedenden ibaret ise boşlukları kaldırıp her kelimeyi büyük harf ile başlayıp yazılır. Nesne adı eşsizdir ve kaynak kodunda nesnenin özellik ve metotlarına ulaşmak için kullanılmaktadır.

Aynı şekilde her konfigürasyon nesnesi *Eş anlamlı* özelliğine sahiptir. Bu özellik nesnenin ek tanımını depolamak için kullanılır. Eş anlamlı uygulamanın arayüzünde ilgili nesneyi tanımlayacak isim olduğundan eş anlamlı kullanıcının göreceği nesne adıdır. Bu yüzden eş anlamlı için herhangi bir karakter kullanma kısıtlaması yoktur ve eş anlamlı serbest şeklinde tanımlanabilir.

Alt sistem resmi

Uygulamanın arayüzüne güzelleştirmek için alt sistem için kullanıcı ortamında gösterileceği resim atanabilir.

Resim alanında seçim butona tiklayalım (Resim 2.4). Resim seçme penceresindeki **Konfigürasyondan** sekmesinde resmi listeye ekleyelim. Bunu yapmak için **Ekle** butonuna basalım (Resim 2.5).

Sistem *Genel resim* konfigürasyon nesnesini oluşturur ve onun özellik düzeltme penceresini açar.

Resim için **Muhasebe** adı belirleyelim. Resmi atamak için **Dosyadan seç** butona basalım (Resim 2.6).

Kitap ile dağıtılan CD'de *Resim* klasöründen gerekli resimi seçelim.

Resmin ön izlemesi için **İzleme** onay kutusunu işaretleyelim.

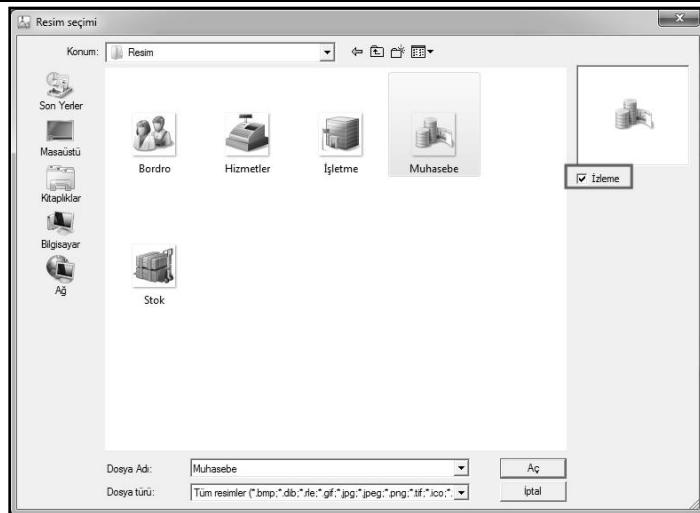
Muhasebe'yi seçi **Aç** butona basalım (Resim 2.7).



Resim 2.5. Alt sistem için resim seç



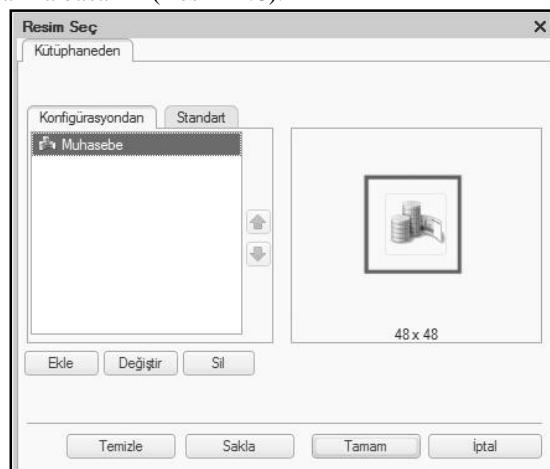
Resim 2.6. "CommonPicture" konfigürasyon nesnesini düzeltme penceresi



Resim 2.7. Alt sistem için resim seç

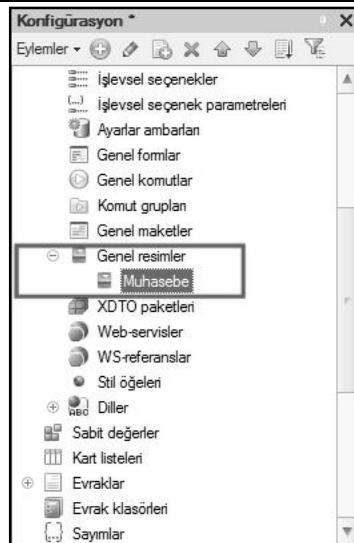
Seçtiğimiz resim genel resim düzeltme penceresinde yer alır.

Genel resim konfigürasyon nesne düzeltme penceresini kapatalım ve **Muhasebe** alt sistemi için resim seçme penceresine geri dönelim. Konfigürasyondan sekmesindeki listede eklemiş olduğumuz Muhasebe resmi yer aldı. Tamam'a basalım (Resim 2.8).



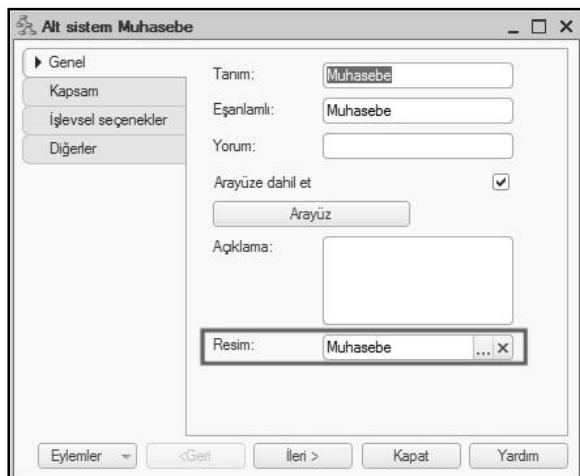
Resim 2.8. Alt sistem için resim seçme

Yapılan işlemler sonucu olarak konfigürasyon nesne ağacındaki *Genel resim* grubunda **Muhasebe** resmi oluşturuldu ve bu resmi biz konfigürasyonumuzda artık kullanabiliriz (Resim 2.9).



Resim 2.9. Konfigürasyon nesne ağacında Genel resimler

Muhasebe alt sistemi düzeltme penceresine geri geldik. Seçtiğimiz resim gerekli alanda yer aldığıni görüyoruz (Resim 2.10).



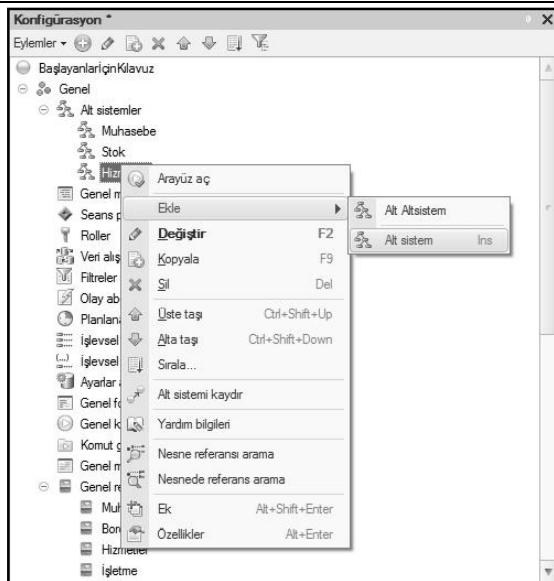
Resim 2.10. Alt sistem için resim seç

Böylece, 1C:İşletme arayüzünde bölüm tanımı olarak eş anlamlısı ve onun üstünde belirlenmiş resim gösterilecektir.

Alt sistem için herhangi bir resim tanımlanmadığı durumda, ilgili bölüm yine de arayüzde yer alır ve resim olarak varsayılan standart resim uygulanacaktır.

Yine **Alt sistemler** grubunu işaretleyip konfigürasyon penceresinin araç çubuğunda **Ekle** butonuna basarak *Stok* ve *Hizmetler* alt sistemlerini oluşturalım. Oluşturuktan sonra bu iki alt sistem için *Muhasebe* alt sistemi için uyguladığımız aynı yöntemi kullanarak *Resim* klasöründen aynı ismi olan resimleri ekleyip tanımlayalım.

Şimdi farklı bir alt sistem ekleme yöntemi kullanalım. Oluşturulmuş alt sistemlerden birisinin bağlam menüsünü çağıralım. **Ekle** maddesini seçelim. **Alt sistem** maddesini seçerek aynı seviyeye sahip olan alt sistem seçme olanağını sağlar, **Alt altsistem** maddesi ise, seçilen alt sisteminin bir alt alt sistemi oluşturma imkanını verir (Resim. 2.11).



Resim 2.11. Konfigürasyon nesne ağacına yeni alt sistem ekle

Konfigürasyonumuzda çok seviyeli kapsamlı sistem yapılmayacağı için **Alt sistem** seçeneğini seçip **Bordro** alt sistemi oluşturalım. Bordro alt sistemi için de resim belirleyelim.

Son olarak, yönetimsel işlevlerine ulaşmak için **İşletme** alt sistemi oluşturalım.

Uygulama çözümünün bölümler çubuğu

1C:İşletme ortamında

1C:İşletme'yi hata ayıklama biçiminde çalışıralım ve yapmış olduğumuz değişiklikleri görelim. Geliştirilmiş uygulama çözümünün görünümü değişmiş (Resim 2.12).

Ana menünün hemen altında, oluşturduğumuz alt sistemleri içeren, uygulamanın *bölümler çubuğu* bulunur. Tüm bölümler atanmış resimler ile gösterilmektedir.



Resim 2.12. 1C:İşletme

Bölümler köprüler şeklinde gösterilir, bölüm simgesine basıldığında ilgili alt sisteme bulunan evraklara, kart listelerine, raporlara vs. ulaşılabilir. Şu anda bölüm içerikleri boştur, çünkü konfigürasyonda orada yer alabilecek hiçbir konfigürasyon nesnesi oluşturulmamış

Not

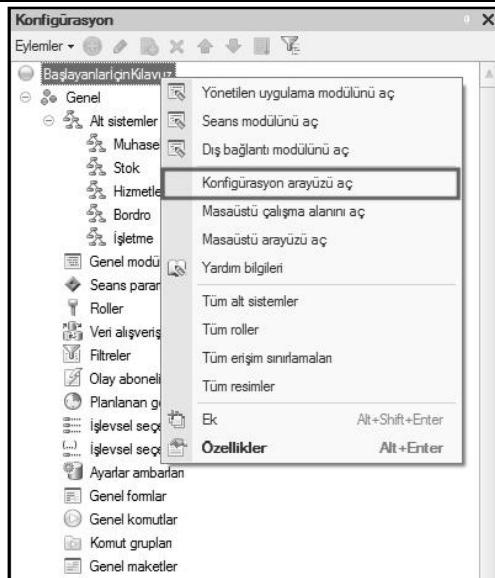
Masaüstü bölümü platform tarafından otomatik olarak oluşturulmaktadır. Kullanıcının en sık kullandığı kart liste, evrak, rapor vs. nesneleri yerleştirmek için kullanılır.

Bölüm sırası

Tasarımcı ortamında

Alt sistem yerleştirme düzeni bizim için tam olarak uygun değildir. Onu değiştirelim.

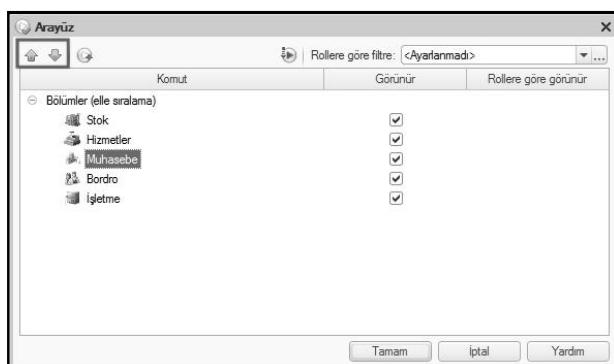
Uygulamayı kapatıp tasarımcıya geri dönelim. *Başlayanlar İçin Kılavuz* konfigürasyon nesne ağacının kökünü işaretleyip, sağ tuşla açılan bağlam menüsünden **Konfigürasyon arayüzü aç** maddesini seçelim (Resim 2.13).



Resim 2.13. Konfigürasyon arayüz ayarlama penceresini aç

Açılmış **Arayüz** penceresinde oluşturmuş olduğunuz alt sistem (uygulama bölgümleri) listesini görürsünüz. **Alta taşı** ve **Üste taşı** butonları ile bu listedeki bölgümler sırasını değiştirelim.

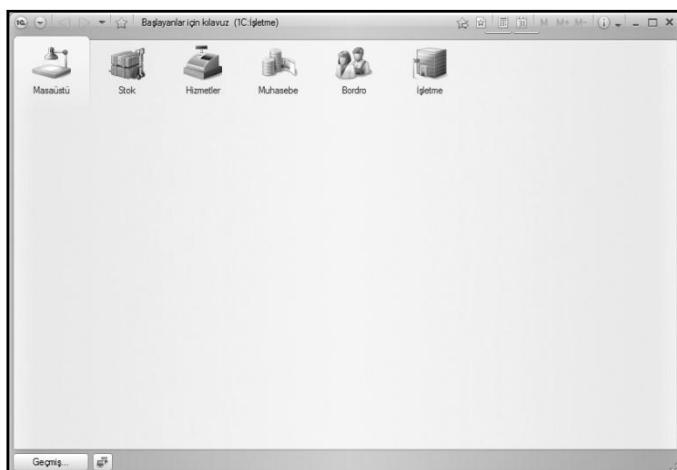
İlk sırada işletmemizin faaliyetini yansıtan alt sistemleri yerlestirelim: Stok ve Hizmetler, ondan sonra muhasebe ve personel maaşları ile ilgili bölgümler gelsin: Muhasebe ve Bordro ve en sonunda İşletme alt sistemini yerlestirelim (Resim 1.14).



Resim 2.14. Alt sistem ayarlama penceresi

1C: İşletme ortamında

1C:İşletme'yi hata ayıklama biçiminde çalışıralım ve yapmış olduğuz değişiklere göre bölümler sırasının değiştirildiğinden emin olalım (Resim 2.15).



Resim 2.15. 1C:İşletme

Uygulamayı kapatıp tasarımcıya geri dönelim. Bir sonraki derste ilk konfigürasyon nesneleri oluşturacağız, oluşturulan nesneler için farklı alt sistemleri belirteceğiz ve 1C:İşletme arayüzünde onların kullanımını göstereceğiz.

Not:

Her başarılı tamamlanmış dersin sonunda, **Konfigürasyon – Konfigürasyonu dosyaya kaydet** menüsü kullanarak, konfigürasyonu yedeklemeyi tavsiye ederiz. Bir sonraki aşamalarda kafanız karışırsa ve bir önceki başarılı dersin sürümüne konfigürasyonunuza geri yüklemek istendiğinde yapmış olduğunuz yedekler işinizi görür. Konfigürasyonu geri yüklemek için **Konfigürasyon – Konfigürasyonu geri yükle** menüsü kullanılır.

“Teori”. Konfigürasyon nesnelerini düzeltme penceresi ve özellikler paleti

İlk bakişa düzeltme penceresi ve özellikler paleti içerik olarak birbiri ile aynı gözükürler. Aslında özellikler paletinde ilgili konfigürasyon nesnesinin tüm özellikleri mevcuttur. Eğer özellikler paletini tüm özellikleri içeriyorsa, düzeltme penceresine ne lüzum vardı? Veya madem düzeltme penceresi mevcut, farklı biçimde aynı özellikleri içeren özellikler paletine ne gerek var?

Düzelme penceresi, ilk başta yeni konfigürasyon nesnelerini hızlı bir şekilde eklemek için tasarlanmış olduğunu belirtmek gereklidir. Aksi halde, hızlı bir şekilde yeni nesne oluşturmak için ilgili nesnenin özellik yapısını çok iyi bilinmesi gereklidir. Bu ise hızlı bir şekilde nesne eklemeye engel olur.

Geliştiriciler hızlı bir şekilde konfigürasyon nesnelerini ekleyebilmeleri için nesne düzeltme penceresi sekme biçiminde ve her sekmenin sıralaması da bir önceki sekmede belirtilen bilgilere bağlı olarak sonraki sekmenin değerleri belirtilir. Sekmeler arasında geçiş yapmak için *İleri* ve *Geri* düğmelerini kullanmak mümkündür.

İlgili nesne bilgilerinin eklendiğini ve farklı düzenleme eklenmesi gerektiğini veya sadece bazı sekmlerdeki özellikleri değiştirmek gerektiğini varsayıyalım, bunun için İleri ve Geri düğmelerini kullanarak bütün sekmelere göz atmak yerine sekmlerin kendilerine tıklayarak sizi ilgilendiren sekmeye ulaşabilirsiniz. Bu biçim bilmediğimiz nesneleri bile hızlı bir şekilde eklemeye ve kolay bir şekilde aranılan özelliklere ulaşmaya yardımcı olur.

Düzelme penceresinin tasarım amacı bu şekilde, peki özellikler paletinin vazgeçilmez yönü nedir? Özellikler paleti, düzeltme penceresi gibi belli bir konfigürasyon nesnesinin türüne bağlı değildir, özelliklerinin içeriği cari nesneye göre değişmektedir. İşte bunun için, hangi özellik alanını seçildiyse ve konfigürasyon nesnelerinin arasında geçişlerde ilgili alan görüntülenir. Özellikler paletinin bu yeteneği örneğin, otuz kırk tane kart listeleri arasında diğer kart listesine bağlı olanını bulmanız gerekiyor. Bu durumda kart listenin *Sahipler* özelliğine tıklarsınız ve konfigürasyon ağacında kart listeleri üzerinde ↑ ve ↓ tuşlarının yardımı ile gezinmeniz yeterlidir.

Sorular

- ✓ Alt sistem konfigürasyon nesnesi ne için kullanılır?
- ✓ Konfigürasyon mantıksal yapısı, Alt sistem nesnesi yardımıyla nasıl tanımlanabilir?
- ✓ Konfigürasyonda Alt sistem kullanma ve gösterme ayarları nasıl yapılır?
- ✓ Konfigürasyon düzeltme penceresi nedir ve özellik panosundan farklılığı nedir?

DERS 3

Kart Listeleri

Süre

Dersin tahmin süresi – 2 saat 10 dakika

Kart listesi nedir?	48
Kart liste formları	53
“Basit” kart listesi	58
Konfigürasyon nesne tanımları	59
Konfigürasyon nesneleri için alt sistemlerin atanması	60
Kart listesinin tanımı ve kodu	62
Yeni öğe ekleme komutu	62
Araç çubuğu ve bölümler çubuğu	65
Kart liste ögesini oluşturmak	66
Tablo bölümü olan kart listesi	69
Tablo bölümü	70
Tablo bölümü doldurması	75
Hiyerarşik kart listesi	77
Hiyerarşik kart listesinde öğe oluşturmazı	80
Öğeleri diğer gruplara taşıması	82
Önceden tanımlı öğeleri içeren kart listesi	84
“Hızlı seçim” özelliği	86
Önceden tanımlı öğeler	86
Sorular	96

Bu dersin konusu *Kart listesi* konfigürasyon nesnesidir. Bu nesne ne için kullanıldığını, yapısı ne olduğunu ve hangi özelliklere sahip olduğunu öğreneceksiniz.

Pratik örnekler üzerinde kart liste oluşturmaması, yapısının en önemli öğelerini tanımlamasını ve kart listelerine veri doldurmasını göreceksiniz.

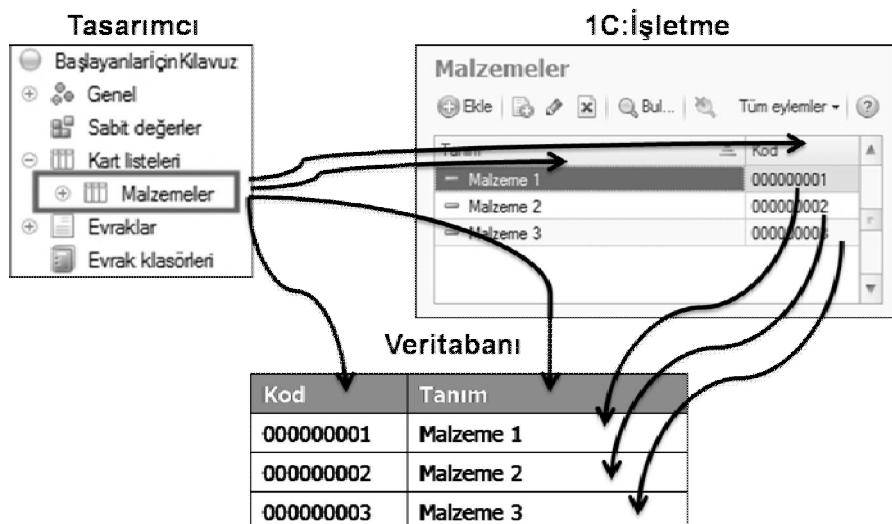
Bunun dışında, yeni konfigürasyon nesnesi hakkında bilgi edineceksiniz – *Form*. Kart liste konfigürasyon nesneleri hangi form türlerine sahip olabilecekleri ve formlar hangi durumlarda kullanılacakları ile ilgili bilgilere sahip olacaksınız.

Kart listesi nedir?

Kart listesi konfigürasyon nesnesi listeler ile çalışmak için tasarlanmıştır. Genelde her işletmede personel, ürün, müşteri, tedarikçi vs. listeler kullanılır. Listelerin yapı ve özellikleri Kart listesi konfigürasyon nesnesinde tanımlanır ve platform bu kart listelerinde veri depolamak için bu bilgilere dayanarak veritabanında gerekli tabloları oluşturur.

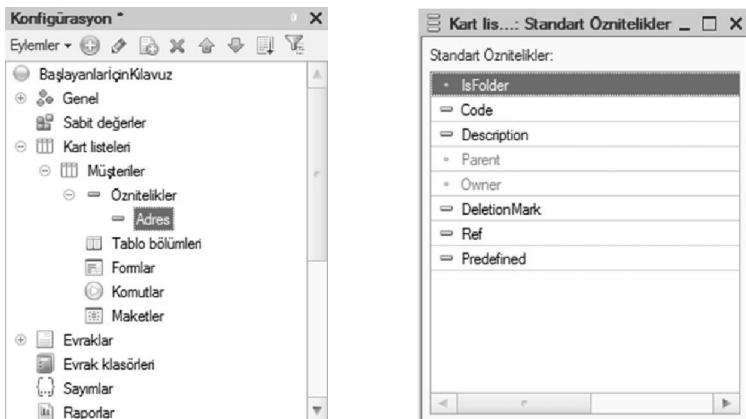
Kart listesi öğelerden ibarettir. Örneğin, personel kart listesi için personel bu bir ögedir, ürünler kart listesi için malzeme bir ögedir vs. Kullanıcı çalışırken, kart listesine yeni öğeleri ekleyebilir; örneğin, yeni personel kartı açmak, ürün eklemek veya yeni müşteri tanımlamak.

Veritabanında kart listesinin her ögesi, genel tabloda ayrı bir kayittır (Resim 3.1).



Resim 3.1. “Malzemeler” kart listesi Tasarımcı, 1C:İşletme ortamlarında ve veritabanında.

Kart listesinin her ögesi, genelde ögeyi detaylı bir şekilde tanımlanan ekstra bilgileri de içerir. Örneğin, *Malzemeler* kart listesinin her ögesi tedarikçi, raf ömrü vs. bilgileri içerebilir. Bu bilgi seti her kart liste öğeleri için aynıdır. Bu bilgi seti tanımlamak için konfigürasyon nesne *özniteligi* kullanılır. Konfigürasyon nesne özniteligi de konfigürasyonun bir nesnesidir (Resim 3.2).



Geliştirici tarafından oluşturulan öznitelikler

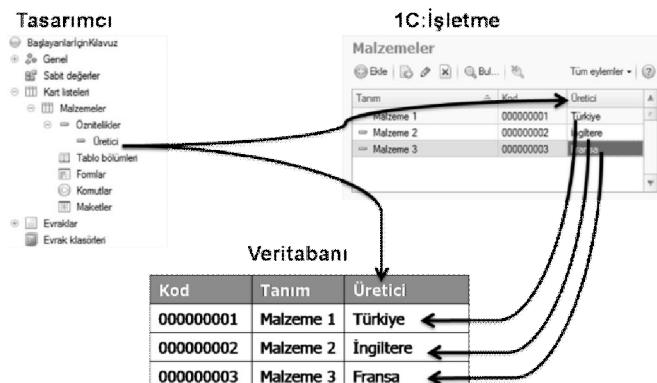
Kart liste standart öznitelikler

Resim 3.2 Kart liste standart öznitelikleri ve geliştirici tarafından oluşturulan öznitelikler

Bu konfigürasyon nesneleri Kart liste öğesine bağlı oldukları için onlara kart listesinin *alt ögesi* denir.

Özniteliklerden çoğu geliştirici tarafından oluşturulur, fakat her *Kart listesi* konfigürasyon nesnesinde standart öznitelik takımı mevcuttur; *Kod*, *Tanım* vs. (Resim 3.2). Bu sırada, standart öznitelik ulaşılabilirliği kart liste özelliklerine bağlıdır.

Örneğin, kart listesi için hiyerarşi belirlendiye **Parent (Sahip)** özniteligi aktif olur. Standart öznitelik olan *Code (Kod)* özniteligin uzunluğu 0 olarak belirlenirse, standart öznitelik listesinde **Code (Kod)** maddesi aktif olmayacağındır. Aynı şey **Description (Tanım)** alanı için geçerlidir. Fakat Tanım ve Kod özniteliklerden an azından bir tanesinin var olması gereklidir, aksi takdirde kart listesinin anlamı kalmaz.



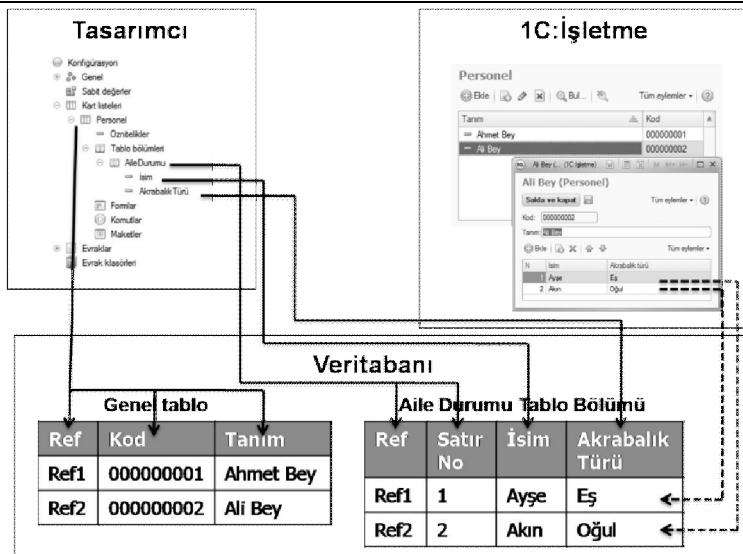
Resim 3.3. “Malzemeler” kart listesi Tasarımcı, 1C:İşletme ortamlarında ve veritabanında

Böylece, veritabanında kart listesi tablo şeklinde depolanır, bu tablonun satırlarında kart liste öğeleri bulunur ve kart listesini her özniteliği (standart veya geliştirici tarafından oluşturulan) tabloda ayrı bir sütun olarak yer almaktadır (Resim 3.3).

Bunun dışında, kart listesinin her öğesi, aynı yapıya sahip fakat her kart liste öğesi için farklı sayıda belirlenebilir bilgilere sahip olabilmektedir.

Örneğin, *Personel* kart listesi personel ailesi ile ilgili bilgi içerebilir. Bir elemanın ailesinde sadece bir kişi var - eş, diğer elemanın ailesinde eş, oğul ve kız olabilir.

Böyle veriyi tanımlamak için *Kart listesi* konfigürasyon nesnesinin *tablo bölümleri* kullanılabilir. Tablo bölümü Kart listesi konfigürasyon nesnesinin alt konfigürasyon nesnesidir. Tablo bölümü kullanıldığında veritabanında tablo bölüm bilgilerini depolamak için belli bir kart listesine bağlı olan ayrı tablolar oluşturulur (Resim 3.4).



Resim 3.4 Tasarımcıda, 1C:İşletme ortamında ve veritabanındaki “Personel” kart listesi

Bu sırada sistem veri depolama ile ilgili işlemin “teknik tarafını” geliştirmeye göstermemektedir; veritabanında kart listesi için birkaç tablo oluşturulur, bu tablolar eşsiz bir alana (Ref) göre ilişkilendirilir, tablo alanları belli bir türde sahiptir vs. Sistem bunu kendisi yapar. Yapmak gereken şey, sadece Kart listesi konfigürasyon nesnesine alt nesnesi olan *Tablo bölümü* nesnesini eklemektir.

Kart liste öğelerini daha rahat bir şekilde kullanabilmek için kart liste öğeleri belli bir mantığa göre gruplandırılabilir.

Örneğin, **Ev eşyaları** kart listesinde şu *gruplar* oluşturulabilir; Buz dolapları, Televizyonlar, Çamaşır makineler vs. Bunun gibi grupların belirleme olanağı, Kart listesi konfigürasyon nesnesinin *hiyerarşik* özelliği tarafından belirlenir. Hiyerarşî belirlendiğinde grup olan kart liste ögesi, içine dahil olan tüm öğeler ve alt gruplar için *Sahip (Parent)* olarak belirlenecektir. Böyle hiperarşî tipine *Öge ve grup hiperarşisi* denir (Resim 3.5).

Malzemeler		Kod
Tanım		
Buz Dolapları		000000003
AEG		000000004
Ardo		000000005
Ariston		000000006 gruplar (üst ogeler)
Çamaşır Makinaları		000000007
Bosch		000000008
Electrolux		000000009
Televizyonlar		000000010
LG		000000011
Philips		
Sony		

Resim 3.5. Öğe ve grup hiyerarşisi olan kart listesi.

Hiyerarşinin başka tipi de mevcuttur – *öğe hiyerarşisi*. Bu durumda üst öğe olarak kart listesinin öğe grubu değil, diğer öğe kullanılır. Örneğin, bir departman diğer departmanlar için üst öğe olarak kullanıldığında, öğe hiyerarşisi *Departmanlar* kart listesi oluşturulduğunda kullanılabilir.

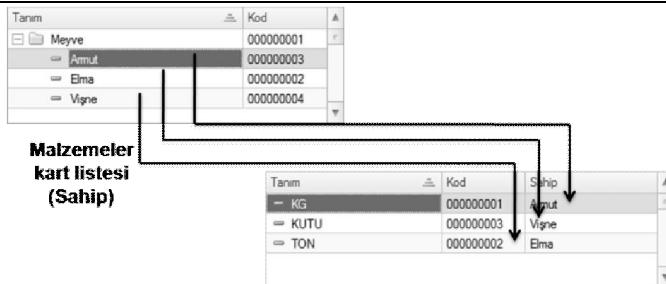
Tanım	Kod
Ankara Şubesi	000000003
İstanbul Şubesi	000000001

Tanım	Kod
İstanbul Şubesi	000000001
Akşaray Şubesi	000000007
Beşiktaş Şubesi	000000008
Şişli Şubesi	000000012

Resim 3.6. Öğe hiyerarşisi olan kart listesi

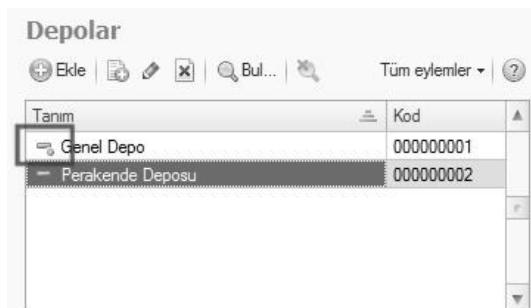
Bir kart listesinin öğeleri diğer kart liste öğelerinin ya da gruplarının alt öğeleri olarak oluşturma imkanı da vardır. Örneğin, *ÖlçümBirimleri* kart liste öğeleri *Malzemeler* kart listesinin alt ögesi olarak belirlenebilir. Böylece, *Malzemeler* kart listesinin her ögesi için stoka geliş birimi belirlenebilir.

1C:İşletme sisteminde bu işlem, her *Kart listesi* konfigürasyon nesnesi için kart listesinin sahip listesini belirleyerek yapılır. Örneğimizde *Malzemeler* kart listesi *ÖlçümBirimleri* kart listesinin sahibi olacaktır (Resim 3.7).



Resim 3.7. “Ölçüm birimleri” kart listesinin sahibi “Malzemeler” kart listesi

Bazı durumlarda, belli öğelerin kart listesinde her zaman bulunması gereklidir. Örneğin, işletme iş sürecine göre stoka gelen tüm ürünler ilk önce Genel depoya, ondan sonra diğer depolara sevk edilmektedir. Bu durumda *Depolar* kart listesinde *Genel depo*’nın her zaman bulunması gereklidir, aksi takdirde ürünlerin stoka girişi hatalı kaydedilebilir. Kart liste yapısı sınırsız sayıda böyle öğeleri oluşturma olanağını sağlar. Böyle öğelere *önceden tanımlı öğeler* denir (Resim 3.8).



Resim 3.8. “Depolar” kart listesinde önceden tanımlı “Genel Depo”

Önceden tanımlı öğelerin normal öğelere göre farklılık şudur: önceden tanımlı öğeler 1C:İşletme ortamında değil, Tasarımcı ortamında oluşturulur ve son kullanıcı bu öğeleri kaldırılamaz. Silmekten hariç diğer işlemler önceden tanımlı öğeye uygulanabilir; değiştirme, taşıma vs. Kullanıcı arayüzünde önceden tanımlı öğeler özel nokta ile işaretlenir (Resim 3.8).

Kart liste formları

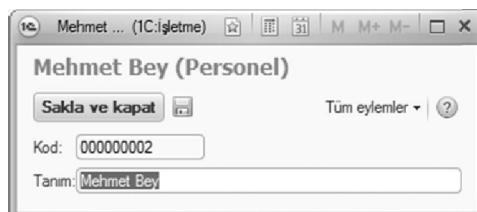
Kart listesi ile yapmak istenilen işlemeye bağlı, kart listesini farklı biçimlerde ekrana getirmek gerekmektedir. Örneğin, kart listesinden öğeyi seçmek için, kart listesini düz liste şeklinde göstermek daha kullanışlı olabilir veya

belli bir kart liste ögesini değiştirmek gereği durumda kart listesinin tüm öznitelikleri tek bir formda yansıtmak gerekebilir (Resim 3.9).

Personel kart listesinin liste formu

Tanım	Kod
Ahmet Bey	00000001
Ali Bey	00000003
Mehmet Bey	00000002

Personel kart listesinin öğe formu



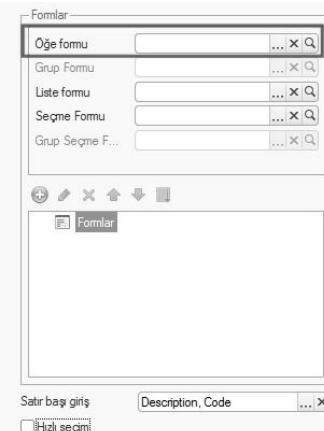
Resim 3.9. Personel kart listesinin liste ve öğe düzeltme formu

Kart listesinde bulunan veriyi görsel şeklinde göstermek için kullanılan formları sistem otomatik olarak oluşturabilir. Bu sırada sistem hangi formları hangi durumlarda kullanmak gerektiğini “bilir”.

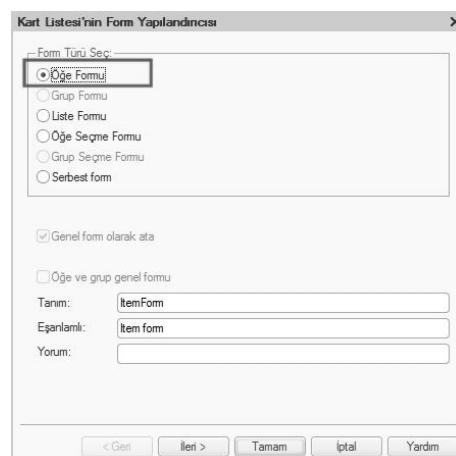
Farklı durumlarda kart listesini ekranda yansıtmak için en fazla beş form gereklidir.

Tablo 3.1 Kart liste formları

Bağlam menüsünde ve özellik panosunda (Resim 3.12).	Form yapılandırıcısında (Resim 3.11).	Form sekmesinde (Resim 3.10).
Item form	Öğe formu	Öğe formu
Group form	Grup formu	Grup formu
List form	Liste formu	Liste formu
Choice form	Öğe seçme formu	Öğe seçme formu
Group choice form	Grup seçme formu	Grup seçme formu



Resim 3.10. Formlar sekmesinde kart liste formunun tanımı

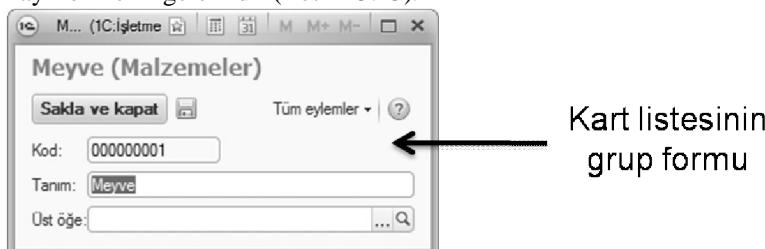


Resim 3.11. Form yapılandırıcısında kart liste formunun tanımı

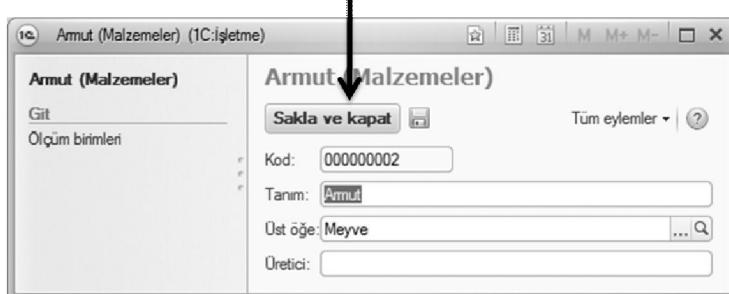
Genel öğe formu	ItemForm	... x
Genel grup formu		... x
Genel liste formu		... x
Genel seçme formu		... x
Genel grup seçme formu		... x

Resim 3.12. Özellik panosunda kart liste formunun tanımı

Öğe formu kart liste öğesini düzeltmek veya oluşturmak için kullanılır. *Grup formu* kart liste grubunu düzeltmek veya oluşturmak için kullanılır. Genelde grup kart liste öğesine göre daha az bilgiye sahiptir. Bu yüzden grup için ayrı bir form gereklidir (Resim 3.13).



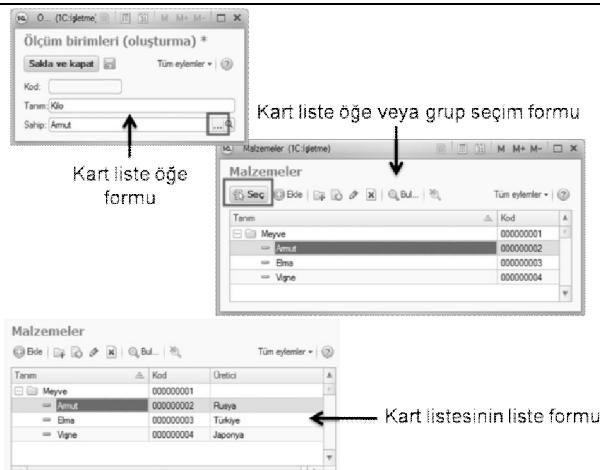
Kart listesinin öğe formu



Resim 3.13. Kart listesinin grp formu ve öğe formu

Liste formu kart liste öğelerini listelemek için kullanılan bir formdur.

Seçim formu diğer formun alanında kart liste öğesini seçmek için kullanılan bir formdur. Genelde seçim formu liste formuna göre daha az detay içerir. Örneğin, evrakta kart liste öğesini seçmek için sadece tanımı görmek yeterlidir. Bu yüzden seçim için daha basit formlar kullanılır. (Resim 3.14).



Resim 3.14. Kart listesinin seçim formu ve liste formu

Konfigürasyonun tüm referans nesneleri için (kart liste, evrak vs.) öğe formu kullanılacaktır. Burada öğe – veritabanındaki öğedir. Kart liste için – kart liste ögesi, evrak için – evrak, hesap planı için – hesap vs.

Tasarımcıda herhangi bir form oluşturulabilir. Bunu yapmak için bağlı konfigürasyon nesnesi mevcuttur – *Form* (Resim 3.15).



Resim 3.15. Tasarımcıda kart listesinin öğe formu

Form konfigürasyon nesnesinde bulunduğu detaylara göre, 1C:İşletme platformu gereği anda kullanıcı çalışacağı görsel formu oluşturur.

Böylece form, veritabanında bulunan verileri görsel şeklinde göstermek için kullanılan öğe türüdür. Form veritabanındaki veriyi görsel şeklinde gösterip içinde bulunan verileri işleme algoritmaları tanımlamaya da olanak sağlar.

"Basit" kart listesi

Kart listesi konfigürasyon nesnesi biraz bilgi aldıktan sonra, veritabanımızda kullanılacak olan birkaç kart listesi oluşturalım.

“Aspirin usta” işletmemiz ev eşyalar ile ilgili teknik hizmet vermektedir, dolayısıyla hesap tutabilmek için birkaç kart listesine ihtiyacımız vardır.

İlk önce işletmede hizmet verecek olan personel listesi gerekir.

Ondan sonra firmamızın çalışacağı müşteri listesi gerekir.

Firmamız verdiği hizmetler ve harcayaceği malzemeler ile ilgili listeler de kullanılacak.

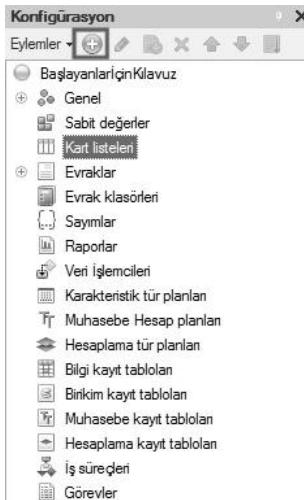
Son olarak malzemelerin bulunacağı depolar listesi lazım olacaktır.

Basit olanlardan başlayalım, personel ve müşteriler listelerini oluşturalım.

İlk önce müşteriler listesini oluşturalım.

Tasarımcı ortamında

Tasarımcı ortamında bizim eğitim konfigürasyonumuzu açalım, *Kart listeleri* grubu işaretleyip, konfigürasyon araç çubuğunda ekle butona tıklayalım (Resim 3.16).



Resim 3.16. Yeni kart listesi konfigürasyon nesnesinin oluşturulması

Ekrana gelen konfigürasyon nesne düzeltme penceresinde kart liste tanımı belirleyelim – **Müşteriler**. *Eş anlamlı* alanında programımızın arayüzünde nesnenin ifade edecek tanım girilir.

Ayrıca, geliştirici kullanma ortamında nesneyi tanımlanacak diğer özellikler de belirlenebilir. Bu özellikler zorunlu özellikler olmadıkları için onlar doldurulmayabilir, doldurulmadığı durumda onların yerine eş anlamlı kullanılır.

Konfigürasyon nesne tanımları

Nesne tanımı nesneyi tekli tanımını belirler ve standart komut oluşturmrasında kullanılır, örneğin nesne oluşturma komutu – **Müşteri: oluştur**. Konfigürasyon nesne eş anlamlısı çoklu olarak belirlendiği durumda nesne tanımını belirtmek gereklidir. Çünkü kullanıcı arayüzünde kart listesini açma veya yeni kart liste öğesini oluşturma komutu otomatik olarak oluşturmaktadır.

Eş anlamlı çoklu olarak belirlendiğinde, liste açma komutu için tanım uygundur – *Müşteriler*, yani tüm müşterilere bak. Fakat kart liste öğesini (bir müşteriyi) oluşturmak için bu seçenek uygun değildir.

Bu komut için tekli tanım belirtmek gereklidir – **Müşteri**. Nesne tanımı, yeni müşteri ekleme komutu tanımlamak için kullanılabilir. Aynı zamanda nesne tanımı form başlığında (liste form tanımı alanında değer yoksa) ve müşteriye referans tanımında kullanılacaktır.

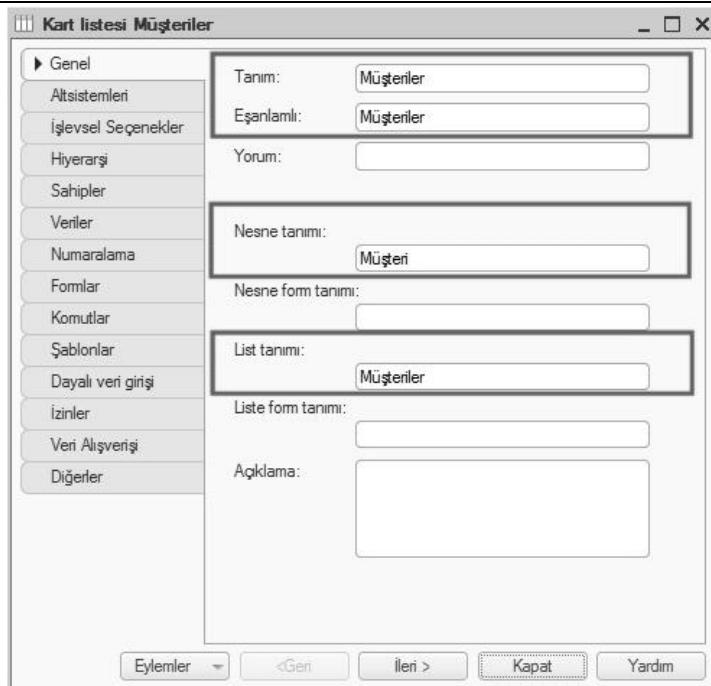
Nesne form tanımı formun başlık tanımını belirler, örneğin yeni kart liste formunun başlığı. Bu özellik belirlenmediği durumda, onun yerine *Nesne tanımı* kullanılabilir.

Liste tanımı nesne listesinin tanımı belirler ve standart komutlarında kullanılır, örneğin, nesne listesini açma komutu – **Müşteriler: aç**. Liste tanımının eş anlamlı tekli olarak belirlendiği durumda atanması gereklidir.

Örneğin, böyle bir durum evraklarda sık rastlanır (Alım ırsaliyesi). Bu durumda liste tanımında nesne tanımını çoklu olarak belirtmek gereklidir (Alım ırsaliyeleri).

Liste form tanımı liste formunun başlığını belirler, örneğin kart listesinin liste formunun başlığı. Bu özellik belirlenmediği durumda onun yerine *Liste tanımı* kullanılmalıdır.

İki özellik belirleyelim *Nesne tanımı* – **Müşteri** ve *Liste tanımı* – **Müşteriler**. Aslında liste tanımı belirlenmeyebildi, çünkü kart listesinin eş anlamlısı liste tanımı ile bir birini tutuyor (Resim 3.17).



Resim 3.17. Kart listesinin genel özelliklerini belirtme

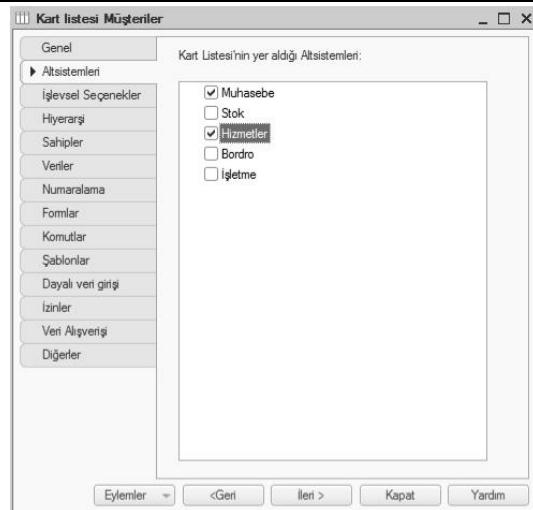
Liste tanımında **Müşteri listesi** tanımı daha doğru olacaktı, fakat ard arda gelen Personel listesi, Müşteri listesi, Depolar listesi satırları uygulamanın arayüzünde pek hoş görünmez.

Konfigürasyon nesneleri için alt sistemlerin atanması

İleri butona basıp kart listesi düzeltme penceresinin *Alt sistemleri* sekmesine geçelim. Bu sekmede kart liste nesnesi hangi alt sisteme ait olduğunu belirlenir.

Alt sistem listesinde, uygulamanın yapısını belirtme aşamasında oluşturduğumuz alt sistem listesi bulunur. Hizmetler farklı müşterilere verildiği için Müşteriler kart listesi için *Hizmetler* alt sistemi belirlenebilir. Müşteri ile ilgili muhasebe takibi *Muhasebe* bölümde yapılacaklığı için Müşteriler kart listesi için *Muhasebe* alt sistemi de belirleyelim.

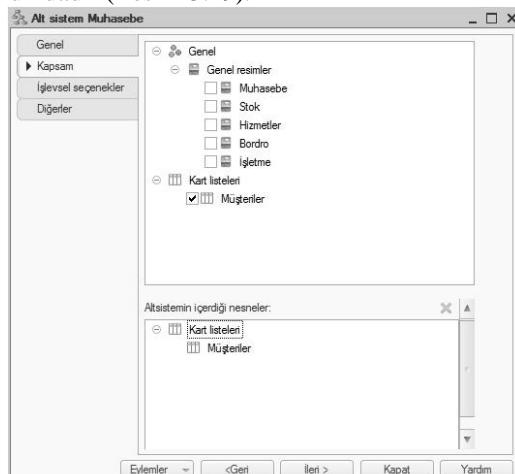
Bu yüzden alt sistem listesinde *Muhasebe* ve *Hizmetler* işaretleyelim (Resim 3.18).



Resim 3.18. Kart listesinin bulunacağı alt sistemleri belirtme

İşaretlenmiş alt sistemlerden bir tanesinin düzeltme penceresini açalım, örneğin, Muhasebe alt sistemi ve Kapsam sekmesine geçelim (Konfigürasyon ağacı – Genel grubu – Alt sistemler grubu – Muhasebe üzerinde çift tıkla – Kapsam sekmesi).

Bu alt sistemin kapsamında olduğu nesne listesinde *Müşteriler* kart listesi işaretlenmiş durumdadır (Resim 3.19).



Resim 3.19. Alt sisteme dahil olan nesne listesi

Kapsam sekmesinde de, alt sisteme dahil olan nesneler seçilebilir ve düzeltilebilir.

Kart listesinin tanımı ve kodu

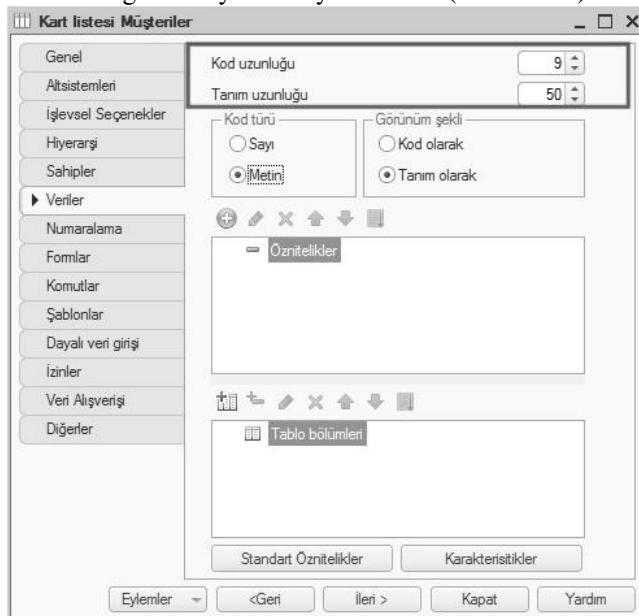
Kart listesi konfigürasyon nesnesinin düzeltme penceresine geri dönüp **Veriler** sekmesine geçelim.

Burada kod ve tanım uzunluğu ile işimiz olacak.

Kod uzunluğu – kart listesinin önemli bir özelliğidir. Genelde kart liste kodu kart liste öğelerini tanımlamak için kullanılır ve her kart liste öğesi için eşsiz değere sahiptir. Platform otomatik olarak kodun eşsiz olmasını kontrol edebilir ve kart liste öğelerinin otomatik numaralandırmasını sağlayabilir. Bu yüzden kart listesindeki öğe sayısını kod uzunluğuna bağlıdır.

Kod uzunluğu – 9 karakter. Sonuç olarak 1 – 999999999 arası kod atanabilir. Küçük ölçekli “Aspirin usta” şirketi için yeterlidir.

Tanım uzunluğuna geçelim. 25 karakter bizim için yeterli değildir, bu yüzden tanım uzunluğunu 50'ye kadar yükseltelim (Resim 3.20).



Resim 3.20. Kart listesinin kod ve tanım uzunluğunu belirtme

Yeni öğe ekleme komutu

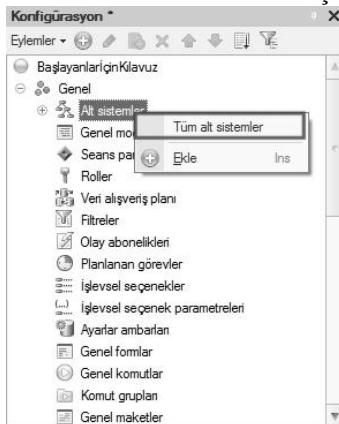
1C:İşletme ortamında programı çalıştırmadan önce, daha rahat kullanımı için programın arayüzüne ayarlayalım.

1C:İşletme arayüzünde liste açma ve yeni öğe oluşturma standart komutları yerleştirmek için genel standart algoritma vardır. Kart listesini örnek olarak anlatacağız, fakat bu algoritma evraklar, hesap planları vs. için de geçerlidir.

Kart listesini açma, kart liste öğesini oluşturma komutu da olduğu gibi, kart listesi bulunduğu bölüm (alt sistem) altında oluşturulur. Fakat yeni öğe oluşturma komutu arayüzde gösterilmemektedir.

Çünkü genelde listeleri izlemek her zaman istenir, ama yeni öğe oluşturmak her zaman gerekmeyez. Bu yüzden yeni öğe oluşturma komutu sadece gerek duyulduğu durumlarda gerekli bölümde etkinleştirilmek gereklidir.

Hizmetler bölümünde yeni *Müşteri* oluşturma komutu etkinleştirilelim. Bunu yapmak için konfigürasyon nesne ağacında *Alt sistemler* grubu işaretleyip bağlam menüsünden **Tüm alt sistemler** maddesini seçelim (Resim 3.21).

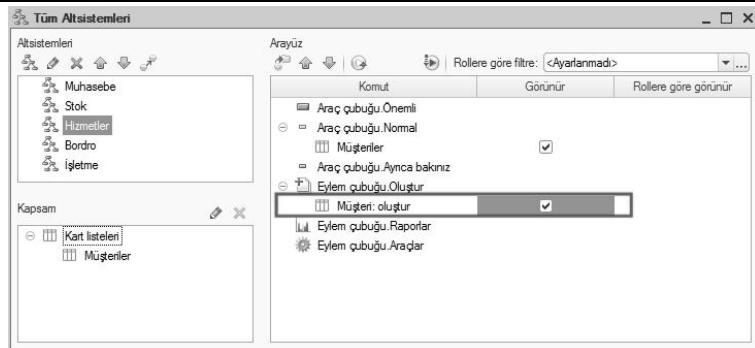


Resim 3.21. Alt sistem ayarlama penceresini açmak

Ekrana gelen **Tüm alt sistemler** penceresinin soldaki **Alt sistemler** listesinde *Hizmetler* alt sistemini işaretleyelim. Sağ taraftaki **Arayüz** alanında, seçilen alt sisteme ait tüm komutlar yansıtılır.

Kart listesi oluşturulduğunda **Araç çubuğu.Normal** grubuna *Müşteriler* komutu eklenmiştir. Varsayılan değer olarak aktif durumdadır. **Eylem çubuğu.Oluştur** grubuna *Müşteri: Oluştur* komutu eklenmiştir, fakat varsayılan değer olarak bu komut devre dışıdır.

Bu komutu etkinleştirilelim (Resim 3.22).

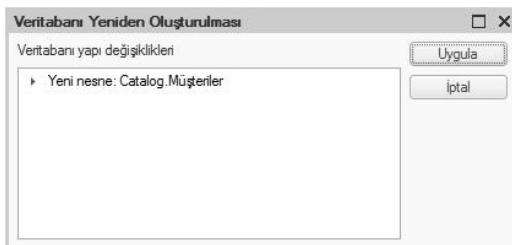


Resim 3.22. Alt sistem ayarlama penceresi

Muhasebe alt sistemi için eylem çubuğu herhangi komut belirlenmeyecektir, çünkü işletme iş sürecine göre muhasebeci için yeni müşteri kartlarını hızlı oluşturma işlemi gereksizdir. Fakat gerek duyulduğunda muhasebeci Müşteri listesini açıp yeni müşteriyi ekleyebilecektir. Nesne listesini kullanmadan, yeni öğe oluşturma komutu, izin kısıtlama gibi bir fonksiyon değil, yoğun çalışma biçimini daha rahat ve kullanışlı bir hale getirmek için kullanılan bir fonksiyondur. Bizim uygulamamızda hızlı müşteri tanımlama fonksiyondan muhasebeci değil, satıcılar yöneticiyi faydalananacaktır.

Müşteriler kart listesini kapatıp 1C:İşletme sistemini hata ayıklama biçiminde çalıştırıralım. Tasarımcının ekrana getirdiği, konfigürasyon değişikliği ile ilgili onay penceresinde Evet diyelim ve programın otomatik olarak oluşturulan konfigürasyon değişiklik listesi gösterilir. *Müşteriler* kart listesini eklemiştik.

Uygula butona basalım.

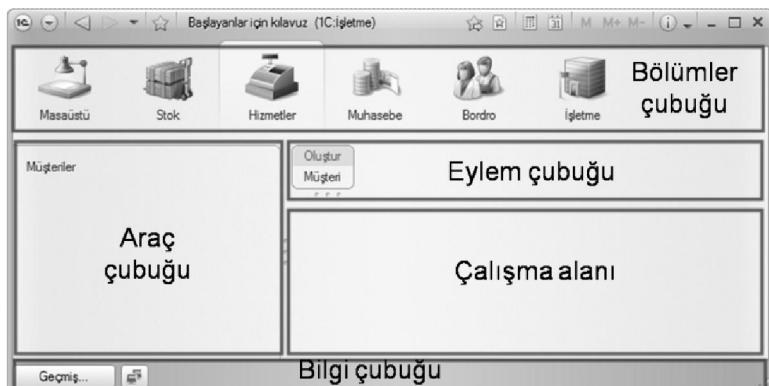


Resim 3.23. Konfigürasyon yapısındaki değişiklik listesi

1C: İşletme ortamında

Araç çubuğu ve bölümler çubuğu

Ekranımızda 1C:İşletme sistemi kullanma ortamında açılır. Hizmetler veya Muhasebe bölmelere geçtiğine sağ tarafta *araç çubuğu* ortaya çıktığını görürüz (Resim 3.24).



Resim 3.24. 1C:İşletme penceresi

Hizmetler bölümünde Eylem çubuğu da yer almıştır. *Eylem çubuğu*, işaretlenmiş bölüme (alt sisteme) ait olan komutları içerir. Komutlar standart olarak gruplandırılmaktadır; *Oluştur*, *Raporlar*, *Servis* ve geliştirici tarafından geliştirilen diğer gruplar. *Oluştur* grubu yeni veritabanı öğelerini oluşturmak için kullanılan komutları içerir, örneğin, kart liste öğelerini veya evrakları oluşturma komutları.

Şu anda *Hizmetler* bölümünde *Oluştur* grubu altında, demin oluşturduğumuz *Müşteriler* kart listesinin öğelerini oluşturmak için etkinleştirdiğimiz (arayüzde yansıtlığımız) komut bulunur. *Müşteriler* kart listesini açmadan yeni müşteri eklemek için bu komutu kullanacağız.

Yeni öğe oluşturma komutun adı, kart listesi için belirlendiğimiz *Nesne tanımı* özelliği tarafından belirlenmektedir. Bu özelliği belirlemeseydik, komut ismi olarak kart listesinin eş anlamlısı kullanılırdı – **Müşteriler**. Bu uygun değildir, çünkü öyle olsaydı kart listesini açma ve yeni öğe oluşturma komutları birbirine benzerdi ve kullanıcı onları karıştırabilirdi.

Muhasebe alt sistemi için yeni öğe oluşturma komutlarını etkinleştirmedigimiz için, eylem çubuğu bulunmamaktadır.

Bunun dışında, *Raporlar* veya *Servis* grupları altında bulunan komutlar yoktur. Dolayısıyla, grupta bulunan komutlar yoksa, grup gösterilmez, eylem çubuğunda gruplar yoksa, o zaman eylem çubuğu da gösterilmez.

Kart liste öğesini oluşturmak

Şu anda oluşturduğumuz kart listesi boştur. Kart listesine birkaç öğe ekleyelim. Bunu yapmak için *Hizmetler* bölümü altındaki *Müşteri* komutu çalıştırıralım.

Kart liste öğesini oluşturmak için ekranımıza form gelir – genel öğe formu (Resim 3.25).

Yeni müşterinin tanımı ekleyelim – **Ahmet Bey**. Kod otomatik olarak oluşturulduğu için kod belirlemeyeceğiz.

Sakla ve kapat tıklayalım.



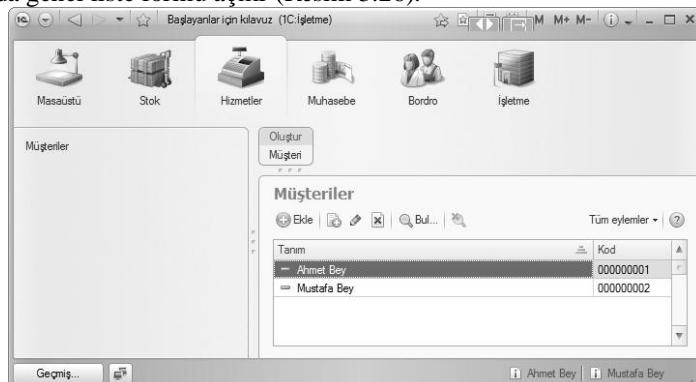
Resim 3.25. Kart listesinin yeni öğesini oluşturmak

Bu sırada sağ alt köşede hangi nesne saklandığı veya değiştirildiği ile ilgili bilgi mesajı ekrana gelecektir.

Bunun yanı sıra, bilgi çubuğundaki (sağ alt köşe) referans yazısına basıp ilgili nesne açılabilir. Bu çubukta sisteme yapılan son hareketler izlenebilir. Gerekli öğenin saklanıp saklanmadığını öğrenmek için liste kullanmak gerek yoktur.

Bir müşteri daha ekleyelim – **Mustafa Bey**.

Son müşteriyi kart listesini kullanarak ekleyelim – **Ayşe Hanım**. Bunu yapmak için *Hizmetler* bölümünde bulunan araç çubuğundaki *Müşteriler* komutu çalıştırıralım. Araç çubuğunun sağ tarafında, pencerenin çalışma alanında genel liste formu açılır (Resim 3.26).

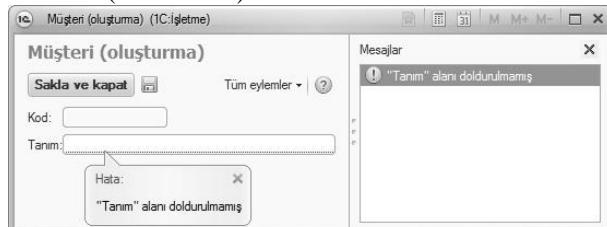


Resim 3.26. Müşterilerin genel liste formu

Formun araç çubuğundaki **Ekle** butonu veya **Insert** kısa tuş ile yeni kart liste öğesi oluşturulabilir.

Ekle butona tıklayalım.

Yeni müşteri eklendiğinde **Tanım** alanı kırmızı kesik çizgi ile işaretlenmektedir. Bu şu demektir; bu alan için veri doldurma denetimi uygulanır. Bu alan doldurmadan müşteri kartı kaydedildiğinde hata mesajı ekrana getirilecektir (Resim 3.27).

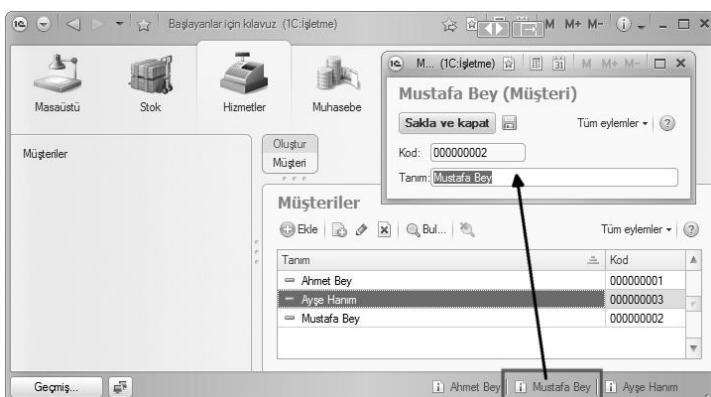


Resim 3.27. Yeni kart liste öğesini ekleme sırasında hata mesajı

Sistem bazı standart öznitelikler için otomatik olarak veri doldurma denetimini belirler. Tanım alanı kart listesinin ana ifade etme alanı olarak belirlendiği (varsayılan değer), Tanım alanı için veri doldurma denetimi geçerlidir.

Müşteri tanımı girelim – **Ayşe Hanım**.

Öğeleri ekledikten sonra kart listesi aşağıdaki gibi görünecektir (Resim 3.28).



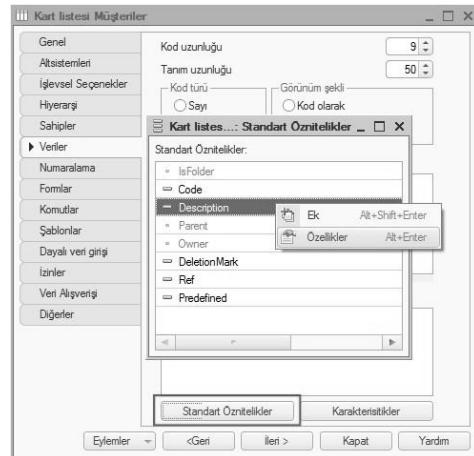
Resim 3.28 Müşteri listesi

Var olan ve kart liste öğesini değiştirmek istendiğinde üzerinde çift tıklamak yeterlidir. Son değiştirilmiş olan öğelerden birini açmak için bilgi çubuğundan ilgili isime tıklamak yeterlidir (Resim 3.28).

'Teori' standart özniteliklerini doldurma denetimi

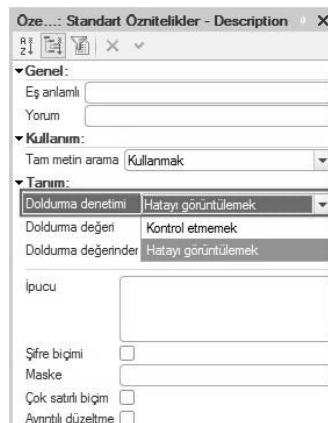
1C:İşletme ortamını kapatalım ve Tasarımcı ortamına geçelim.

Standart öznitelik içeriğini ve özelliklerini görmek için **Müşteriler** kart listesinin düzeltme penceresinde **Veriler** sekmesinde **Standart Öznitelikler** düğmesine tıklayalım ve **Description** (Tanım) öznitelığının bağlam menüsünü çağırıralım (Resim 3.29).



Resim 3.29 Kart listesi standart öznitelikler listesi

Description Standart Öznitelığının özellikler paletinde Doldurma denetimi özelliğinde varsayılan olarak Hatayı görüntülemek özelliği ayarlanmıştır (Resim 3.30).



Resim 3.30 Description Standart öznitelığının özellikler paleti

Bu şu demek, Description (Tanım) alanı doldurulmadığı durumda hata mesajı görüntülenecektir (Resim 3.27 bakınız).

Tablo bölümü olan kart listesi

Şimdi konfigürasyonumuzda kullanılacak olan ikinci kart listesini oluşturabiliriz – **Personeller**.

Bu kart listesi, *Müşteriler* kart listesinden biraz daha kapsamlı olacaktır. Bu kart listesinde sadece çalışanın ad ve soyadı bilgileri değil, geçmiş çalışma yerleri ile ilgili bilgi de kaydedilecektir.

Bu bilgi benzer yapıya sahip olacaktır (şirket, başlama tarihi, ayrılma tarihi, görev), fakat çalıştığı yer sayısı farklı olabilir. Bu yüzden bu bilgiyi kaydetmek için kart listesinin tablo bölümünü kullanılcaktır.

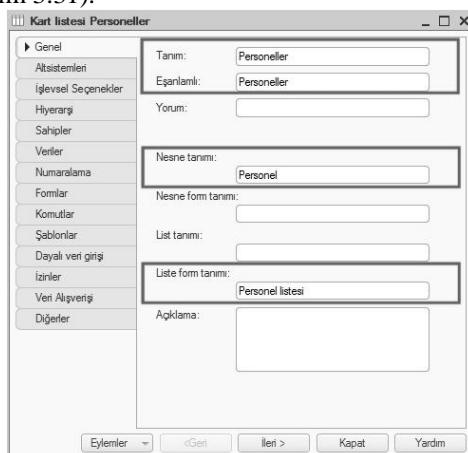
Tasarımcı ortamında

Yeni kart liste nesnesini ekleyelim – **Personeller**.

Tanıma göre platform otomatik olarak eş anlamlı alanını doldurur.

Nesne tanımını **Personel** olarak belirleyelim.

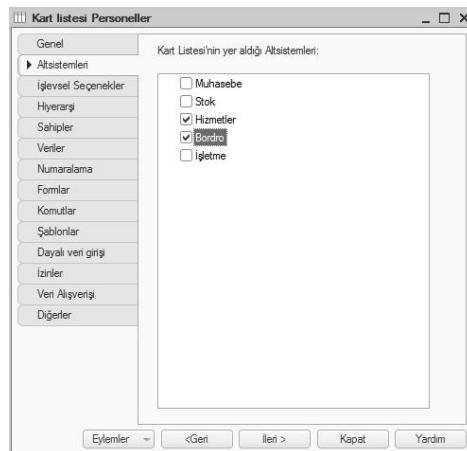
Liste tanımı belirlemeyelim, *Liste form tanımı* ise **Personel listesi** olarak belirleyelim (Resim 3.31).



Resim 3.31. Kart listesinin genel özelliklerini belirtme

İleri butona basıp *Alt sistemleri* sekmesine geçelim.

Uyguladığımız iş sürecine göre personel listesi *Hizmetler* ve *Bordro* bölümlerde yer alacaktır. Bu yüzden alt sisteme listesinde *Hizmetler* ve *Bordro* alt sistemlerini işaretleyelim (Resim 3.32).



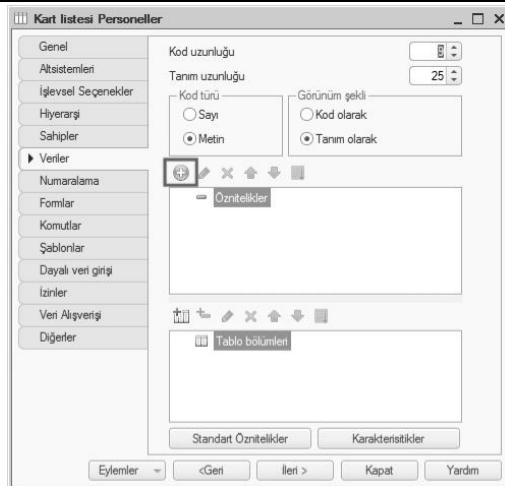
Resim 3.32. Kart liste yansıtılacak alt sistemlerini tanımlama

Veriler sekmesine geçelim. Kod tipi ve uzunluğu olduğu gibi bırakalım, kart liste tanımının uzunluğu ise 50 karakter olarak değiştirelim.

Tablo bölümü

Görevimiz, tablo bölümü olan kart listesini oluşturmak. Bu yüzden kart listesine yeni bir tablo bölümü ekleyelim – **ÇalışmaYerleri**.

Bunu yapmak için kart listesinin tablo bölüm listesinin araç çubuğunda bulunan **Tablo bölümü ekle** butona tıklayalım (Resim 3.33).

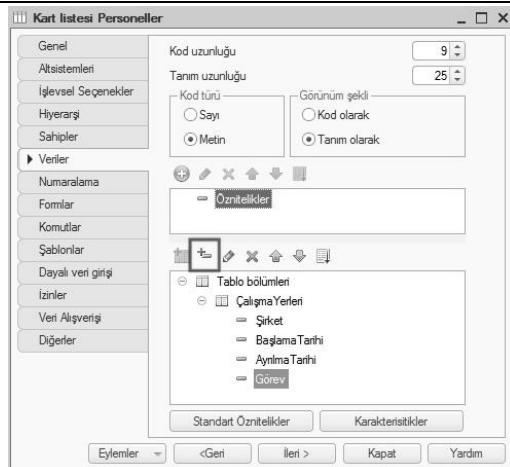


Resim 3.33. Yeni tablo bölümü ekleme

Tablo alan adı belirleyelim – ÇalışmaYerleri



Resim 3.34. Yeni tablo bölümü ekleme
ÇalışmaYerleri tablo bölümünün öznitelikleri oluşturuluyor.
 Bunu yapmak için kart listesinin tablo bölüm listesinin araç çubuğuunda bulunan **Öznitelik ekle** butonuna tıklayalım (Resim 3.35).

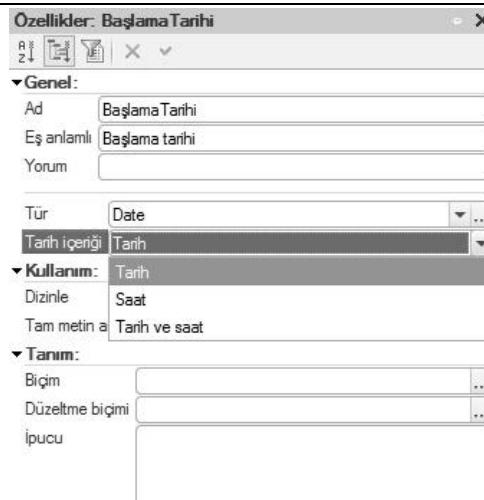


Resim 3.35. Tablo bölümüne yeni öznitelik ekleme

Aşağıdaki öznitelikleri ekleyelim;

- **Şirket** – tür **String**, uzunluk **100**,
- **BaşlamaTarihi** – tür **Date**, tarih içeriği **Tarih**,
- **AyrılmaTarihi** – tür **Date**, tarih içeriği **Tarih**,
- **Görev** – tür **String**, uzunluk **100**.

BaşlamaTarihi ve AyrılmaTarihi öznitelikler için tarih içeriği – **Tarih** seçiyoruz (Resim 3.36), çünkü 1C:İşletme 8 sisteminde **Date** tipi olan değer hem tarihi hem saatı içerebilir. Örneğimizde, işe başlama ve ayrılma saatı bizim için gereksiz.



Resim 3.36. Tablo bölüm öznitelliğinin özellikler

Son olarak, kart liste öğelerini hızlı ekleyebilmek için uygulamanın arayüzü düzeltelim. *Bordro* alt sisteminin eylem çubuğunda yeni personel oluşturma komutu etkinleştirelim.

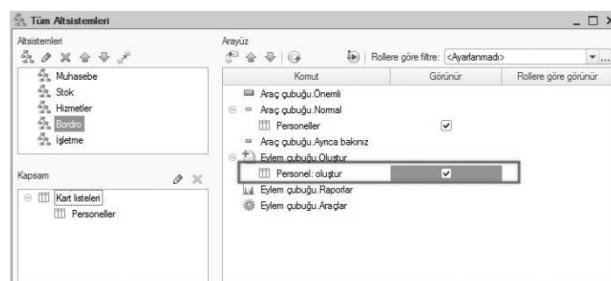
Bunu yapmak için konfigürasyon ağacında *Alt sistemleri* grubu işaretleyip bağlam menüsünden **Tüm alt sistemler** maddesini seçelim.

Açılan pencerede, sol tarafta alt sistem listesinde *Bordro* alt sistemini işaretleyelim.

Sağ taraftaki *Arayüz* alanında bu alt sistem ile ilgili tüm komutlar gösterilecektir.

Eylem çubuğu.Oluştur grubunda **Personel: oluştur** komutu işaretleyelim.

Araç çubuğu.Genel grubunda personel listesine ulaşabilmek için **Personeller** komutu eklenmişti. Varsayılan değer olarak bu komut işaretlenmiş durumdadır (Resim 3.37).

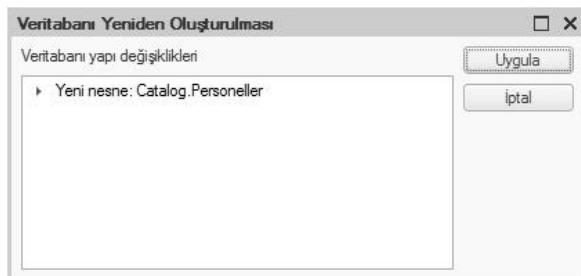


Resim 3.37. Alt sistem ayarlama penceresi

Burada Personeller kart listesi oluşturma işlemleri tamamlanmıştır.

Personeller kart listesini düzeltme penceresini kapatıp, 1C:İşletme sistemini hata ayıklama biçiminde çalışıralım. Tasarımcının ekrana getirdiği, konfigürasyon değişikliği ile ilgili onay penceresinde Evet diyelim ve programın otomatik olarak oluşturulan konfigürasyon değişiklik listesi gösterilir. *Personeller* kart listesini eklemiştik.

Uygula butona basalım.



Resim 3.38. Konfigürasyon yapısındaki değişiklik listesi

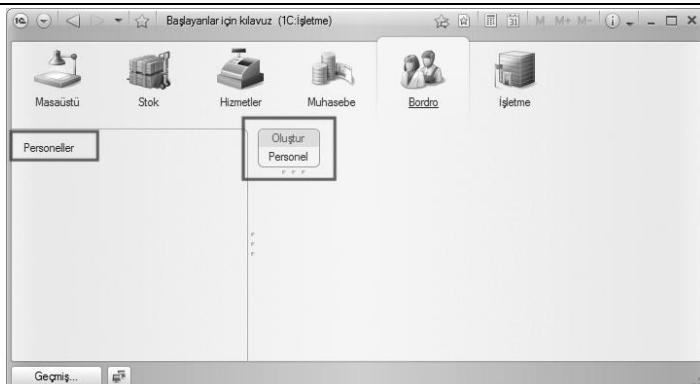
1C: İşletme ortamında

Açılan 1C:İşletme penceresinde *Hizmetler* ve *Bordro* bölümlerin araç çubuklarında personel listesini açmak için *Personeller* komutu ortaya çıktı (Resim 3.39).

Liste tanımı belirlemediğimiz için komut tanımı kart listesinin eş anlamlısı tarafından belirlenmişti.

Bordro bölümün eylem çubuğuunda yeni personel kartını açmak için Personel komutu yer aldı (Resim 3.39). Bu komutun tanımı kart liste nesnesi için belirlediğimiz *Nesne tanımı* tarafından belirlenmektedir.

Personel listesini açmadan, hızlı eklem komutu kullanarak yeni personel kartlarını ekleyeceğiz.



Resim 3.39. “Bordro” bölümü

Tablo bölümü doldurması

Personel komutuna tıklayalım

Kart liste öğesini oluşturmak için ekranımıza form gelir – genel üye formu. Bu formun başlığı nesnenin *Nesne tanımı* özelliği ile belirlenir.

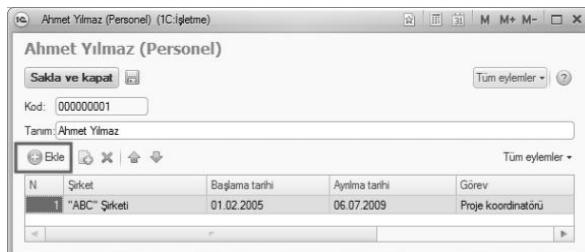
Bu formda Tasarımcı ortamında tanımladığımız tablo bölümü bulunmaktadır.

Aşağıdaki personel kartlarını açalım (Resim 3.40, 3.41, 3.42);

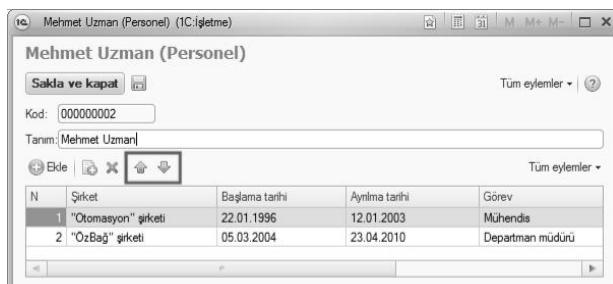
- Ahmet Yılmaz
 - Çalışma yerleri;
 - Şirket – “ABC” şirketi,
 - Başlama tarihi – 01.02.2005,
 - Ayırılma tarihi – 06.07.2009,
 - Görev – Proje koordinatörü.
- Mehmet Uzman
 - Çalışma yerleri;
 1.
 - Şirket – “Otomasyon” şirketi,
 - Başlama tarihi – 22.01.1996,
 - Ayırılma tarihi – 12.01.2003,
 - Görev – Mühendis.
 2.
 - Şirket – “ÖzBağ” şirketi,
 - Başlama tarihi – 05.03.2004,
 - Ayırılma tarihi – 23.04.2010,
 - Görev – Departman müdürü.

- Ali İşçi
 - Çalışma yerleri;
 - Şirket – “İnşaat” şirketi,
 - Başlama tarihi – 02.07.2001,
 - Ayrılma tarihi – 02.11.2010,
 - Görev – Kalite uzmanı.

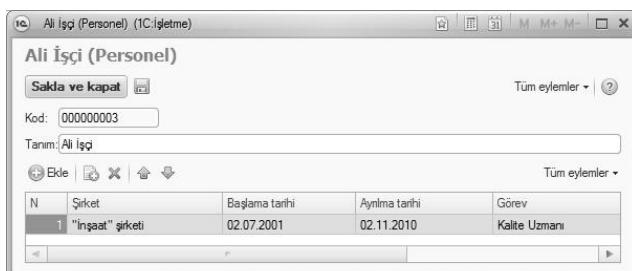
Tablo bölüm satırları Ekle butonu ile eklenebilir (Resim 3.40) ve **Üste taşı/Aşağıya taşı** butonlar yardımıyla serbest şeklinde sıralanabilir (Resim 3.41).



Resim 3.40. “Personeller” kart liste ögesinin doldurması



Resim 3.41. “Personeller” kart liste ögesinin doldurması

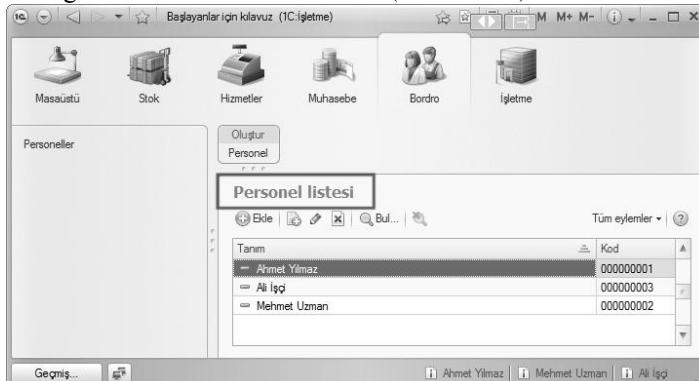


Resim 3.42. “Personeller” kart liste ögesinin doldurması

Eklendiş personel listesine bakmak için *Bordro* bölümdeki araç çubuğunda bulunan *Personeller* komutu uygulayalım.

Araç çubuğunun sağ tarafında bulunda çalışma alanında genel liste formu açılacaktır.

Bu liste formunun başlığı bu kart listesi için belirlendiğimiz *Liste form tanımı* özelliği tarafından belirlenmektedir (Resim 3.43).



Resim 3.43. Personel listesi

Şimdi bir sonraki kart listesi oluşturabiliriz – **MalzemeHizmet**.

Hiyerarşik kart listesi

Malzeme/Hizmet kart listesi “Aspirin usta” şirketi tarafından verilen hizmetler ve hizmet verme aşamasında sarf edilen malzemeler ile ilgili bilgi depolanacaktır.

Bu kart listesi kapsamlı olmayacağındır. Bir tek farklı olan tarafı hiyerarşî yapıya sahip olmasıdır. Kart listesinin daha rahat kullanımı için hizmetler bir gruba malzemeleri ayrı bir gruba yerlestireceğiz.

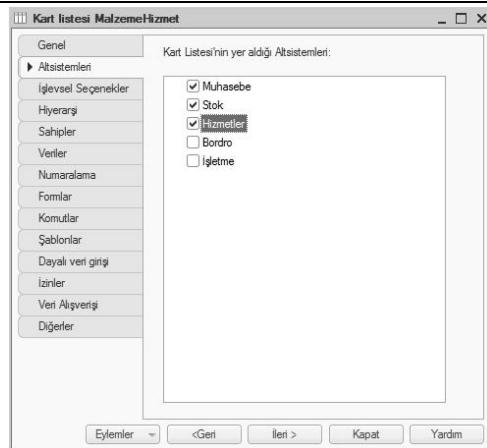
Bunun dışında, “Aspirin usta” şirketi farklı hizmetleri verdiği için, hizmetler de ayrı olarak gruplandırılacaktır. Aynı şey malzemeler için geçerlidir.

Tasarımcı ortamında

Yeni kart listesi konfigürasyon nesnesini oluşturalım ve **MalzemeHizmet** olarak adlandıralım. Eş anlamlı olarak **Malzeme/Hizmet** değeri girelim.

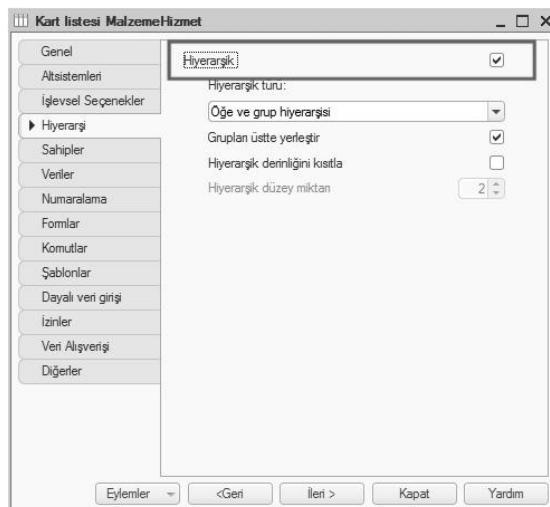
Alt sistemler sekmesine geçelim.

Şirketimizde uygulanan iş sürecine göre malzeme/hizmet listesi *Hizmetler*, *Stok* ve *Muhasebe* bölmünlere aittir. Dolayısıyla alt sistem listesinde gerekli alt sistemleri işaretleyelim (Resim 3.34).



Resim 3.44. Kart listesi ait olduğu alt sistemleri belirtme

Görevimiz hiyerarşik bir kart listesini oluşturmaktır. Dolayısıyla *Hiyerarşî* sekmesine geçip **Hiyerarşîk** onay kutusunu işaretleyelim (Resim 3.45).



Resim 3.45. Kart listesi için hiyerarşîk özelliğini belirtme

Veriler sekmesinde varsayılan kod türü ve uzunluğu bırakalım ve kart liste tanımı için uzunluk 100 karakter olarak belirtelelim.

1C:İşletme ortamında çalıştırmadan önce, kart liste öğelerini hızlı ekleme bilmek için uygulamanın arayüzü düzeltelim. *Hizmetler* ve *Stok* alt

sistemlerinin eylem çubuklarında yeni malzeme/hizmet oluşturma komutu etkinleştirelim.

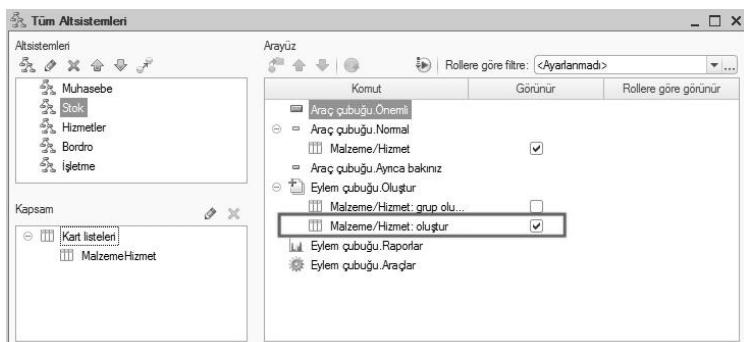
Bunu yapmak için konfigürasyon ağacında **Altsistemleri** grubu işaretleyip bağlam menüsünden **Tüm alt sistemler** maddesini seçelim.

Açılan pencerede, sol tarafta alt sistem listesinde **Stok** alt sistemini işaretleyelim.

Sağ taraftaki *Arayüz* alanında bu alt sistem ile ilgili tüm komutlar gösterilecektir.

Eylem çubuğu.Oluştur grubunda **Malzeme/Hizmet: oluştur** komutu işaretleyelim.

Araç çubuğu.Genel grubunda personel listesine ulaşabilmek için **Malzeme/Hizmet** komutu eklenmişti. Varsayılan değer olarak bu komut işaretlenmiş durumdadır (Resim 3.46).



Resim 3.37. Alt sistem ayarlama penceresi
Hizmetler alt sistemini işaretleyip aynı işlem uygulayınız.

Muhasebe alt sistemi için herhangi bir ayarlama yapılmayacaktır. Çünkü bu bölümde yeni malzeme/hizmet kart liste ögesi büyük ihtimalde oluşturulmayacaktır.

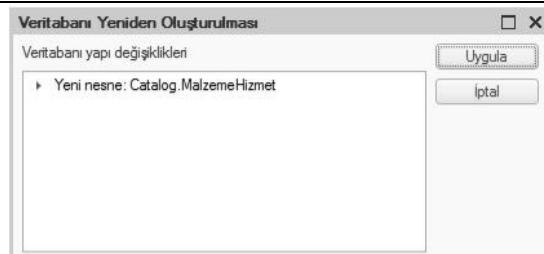
Not

Belli bir alt sistem ile alakalı arayüz ayarları alt sistem düzeltme penceresinden de yapılabilir. Alt sistemin arayüz ayarlama penceresine ulaşmak için düzeltme penceresindeki **Arayüz** butonuna tıklamak yeterlidir.

Şimdi Malzeme/Hizmet kart listesini dolduralım. Doldurma aşamasında grup oluşturma ve öğeleri gruplar arası taşıma işlemleri gösterilecektir.

Malzeme/Hizmet kart listesini düzeltme penceresini kapatıp, 1C:İşletme sistemini hata ayıklama biçiminde çalıştırıralım. Tasarımcının ekrana getirdiği, konfigürasyon değişikliği ile ilgili onay penceresinde Evet diyelim ve programın otomatik olarak oluşturulan konfigürasyon değişiklik listesi gösterilir. *Malzeme/Hizmet* kart listesini eklemiştik.

Uygula butona basalım.



Resim 3.47. Konfigürasyon yapısındaki değişiklik listesi

1C: İşletme ortamında

Açılan 1C:İşletme penceresinde *Stok*, *Hizmetler* ve *Muhasebe* bölümlerin araç çubuklarında malzeme/hizmet listesini açmak için *Malzeme/Hizmet* komutu ortaya çıktı (Resim 3.48).

Diğer tanımlar belirlemediğimiz için komut tanımı kart listesinin eş anlamlısı tarafından belirlenmiştir.

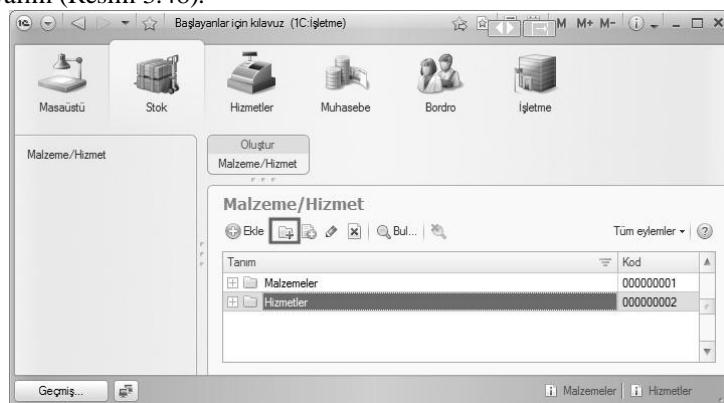
Stok bölümün araç çubوغunda *Malzeme/Hizmet* komutu çalıştırılın.

Araç çubuğuunun sağ tarafındaki çalışma alanında genel liste formu açılacaktır.

Hiyerarşik kart listesinde öğe oluşturmazı

Kart listesinde iki grup oluşturalım; *Malzemeler* ve *Hizmetler*.

Bunu yapmak için liste formunun araç çubugundaki **Grup ekle** butona tıklayalım (Resim 3.48).



Resim 3.48. “Malzeme/Hizmet” kart liste grubu oluşturma

Grup tanımlarını dolduralım; *Malzemeler* ve *Hizmetler*. **Üst öğe** ve **Kod** alanı doldurmayacağız.

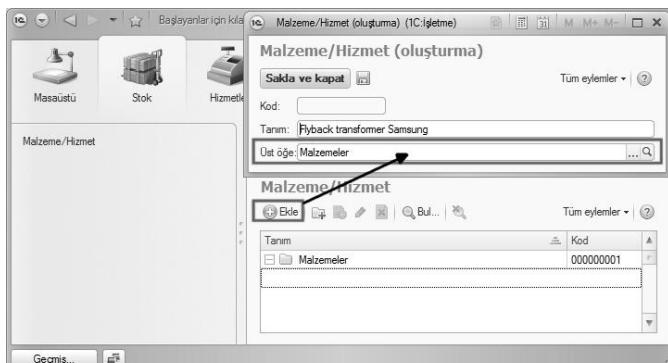
Ondan sonra *Malzemeler* grubu açıp (+ işaretine tıklayarak) beş öğe oluşturalım;

- Flyback transformer Samsung,
- Flyback transformer GoldStar,
- Transistor Philips 2N2369,
- Lastik hortum,
- Elektrik kablo.

Açık kart liste grubuna öğe eklemek için liste formunun araç çubuğunda bulunan **Ekle** butona tıklamak yeterlidir.

Kart liste öğesini oluşturmak için ekrana kart listesinin genel öğe formu gelecektir. Bu sırada, yeni öğe açık bir gruba eklendiği durumda, sistem otomatik olarak üst öğe olarak ilgili grubu dolduracaktır.

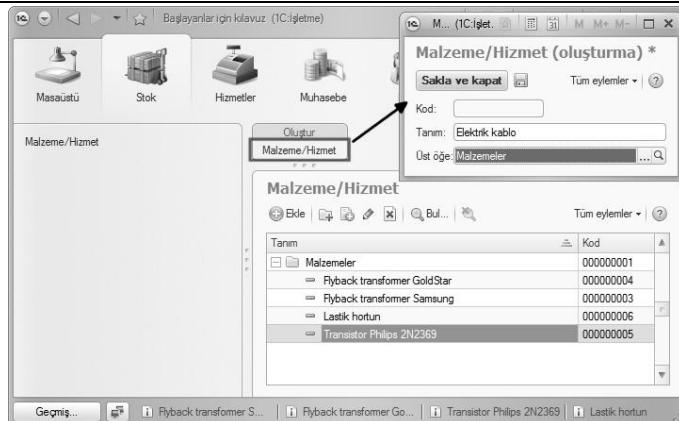
Örneğimizde üst öğe *Malzemeler* grubudur (Resim 3.49).



Resim 3.49. “Malzemeler” grubunda öğe oluşturma

Kart listesinin yeni öğesini oluşturmak için Stok ve Hizmetler bölümlerinin eylem çubuklarında yer alan Malzeme/Hizmet komutu da kullanılabilir (Resim 3.48).

Yeni kart liste öğesi eylem çubuğundan eklendiğinde, ekleme komutu malzeme/hizmet listesi ile bağlı olmadan oluşturma işlemi yapılır. Sistem öğenin hangi gruba ekleneceğini “bilmez”, dolayısıyla üst öğeyi manuel olarak belirtmek gerekmek (Resim 3.50).



Resim 3.50. "Malzemeler" grubunda öğe oluşturma

Yeni malzeme ve hizmetler oluşturulduğunda malzeme/hizmet listesini açmaya gerek yok. Kart listesine öğe ekleme olgusu uygulamanın alt köşede mesaj şeklinde bildirilir.

Hizmetler grubu açıp televizyon tamiri ile ilgili birkaç hizmet ekleyelim (Resim 3.51).

- Teşhis,
- Yerli marka televizyon tamiri,
- Yabancı marka televizyon tamiri,

Ve çamaşır makine tamiri ile ilgili hizmetler;

- Su bağlama,
- Elektrik bağlama.

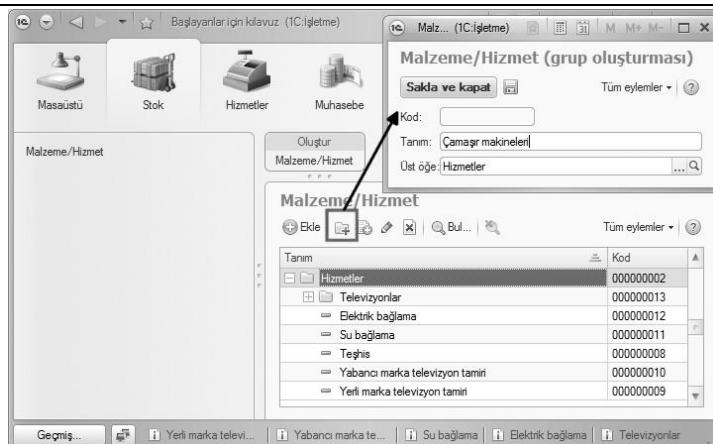
Tanım	Kod
▀ Hizmetler	00000002
▀ Elektrik bağlama	00000012
▀ Su bağlama	00000011
▀ Teşhis	00000008
▀ Yabancı marka televizyon tamiri	00000010
▀ Yerli marka televizyon tamiri	00000009

Resim 3.51. "Hizmetler" grubunda öğe oluşturma

Öğeleri farklı gruplara taşıma

Şu anda hizmetleri iki ayrı gruba ayrılalım; televizyon tamiri ile ilgili hizmetler ve çamaşır makine tamiri ile ilgili hizmetler.

Bunun için *Hizmetler* grubunda iki tane grup oluşturulmuş; *Televizyonlar* ve *Çamaşır makineleri* (Resim 3.52).

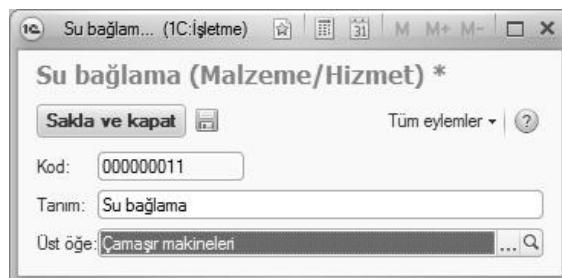


Resim 3.52. "Hizmetler" grubunda yeni grup oluşturma

Gerekli gruba hizmetleri taşımak için ilk önce taşımak istenilen öğe üzerinde imleç yerlestirelim ve **Tüm Eylemler – Gruba kaydır** komutu uygulayalım. Ekrana gelen pencerede gerekli grubu seçelim.

Listede birden fazla öğeyi işaretleyip (Ctrl basılı tutarak sağ tuşla) başka gruba taşıyabilirsiniz. Veya fare ile taşıyarak öğe kaydırılabilir.

Ya da öğeyi açıp *Üst öğe* alanındaki değer değiştirilebilir (Resim 3.53).



Resim 5.53. Kart liste öğesini diğer gruba taşıması.

Televizyon grubuna *Teşhis*, *Yerli marka televizyon tamiri* ve *Yabancı marka televizyon tamiri* hizmetleri taşıyalım.

Su bağlantı ve *Elektrik bağlantı* hizmetleri **Çamaşır makineleri** grubunda yerlestirelim.

Sonra *Malzemeler* grubunda iki grup oluşturalım; **Radyo parçaları** ve **Diger**.

Diger grubuna *Elektrik kablo* ve *Lastik hortun* taşıyalım. Diğer malzemeler **Radyo parçaları** grubunda bulunsun.

Liste görünümü ağaç şeklinde ayarlandığında (Tüm eylemler – Görünüm bicimi – Ağaç menüsü), malzeme/hizmet aşağıdaki gibi ağaç biçiminde gösterilecektir (Resim 3.54).

Tanım	Kod
Malzeme/Hizmet	000000002
Hizmetler	000000014
Çamaşır makineleri	000000012
Elektrik bağlama	000000011
Su bağlama	000000013
Televizyonlar	000000008
Tehnis	000000010
Yabancı marka televizyon tamiri	000000009
Yerli marka televizyon tamiri	
Malzemeler	000000001
Diğer	000000016
Elektrik kablo	000000007
Lastik hortun	000000006
Radyo parçaları	000000015
Flyback transformör GoldStar	000000004
Flyback transformör Samsung	000000003
Transistor Philips 2N2369	000000005

Resim 3.54. Ağaç biçiminde malzeme/hizmet listesi

Önceden tanımlı öğeleri içeren kart listesi

Sonunda, “Aspirin usta” şirketinde bulunan depolar hakkında bilgi depolayacak olan **Depolar** kart listesini oluşturacağız.

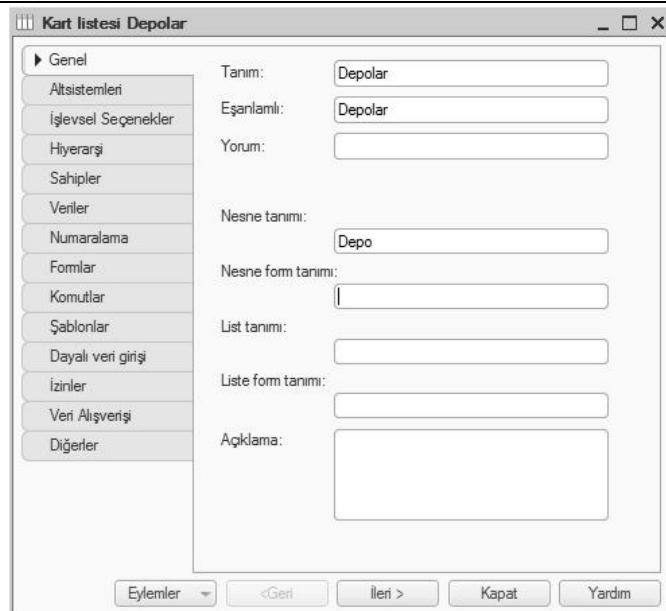
Bu kart listesi bir tane önceden tanımlı öğeyi içerecek – **Genel depo**. Bu depoya tüm malzemeler alınacaktır.

Görevimiz – önceden tanımlı öğeyi içeren kart listesini oluşturmak.

Tasarımcı ortamında

Tasarımcıyı açıp yeni kart listesi konfigürasyon nesnesini oluşturalım ve **Depolar** olarak adlandıralım. Eş anlamlı olarak **Depolar** değeri girelim.

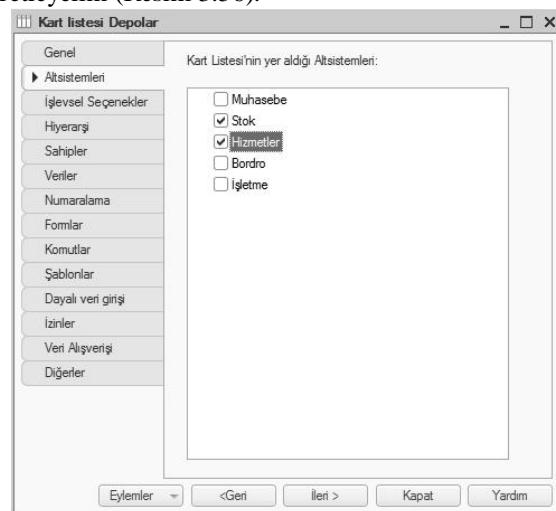
Nesne tanımı Depo olarak belirleyelim. *Liste tanımı* yerine nesne eş anlamlısını kullanacağımız – **Depolar** (Resim 3.55).



Resim 355. Kart liste özelliklerini belirtme

Alt sistemler sekmesine geçelim.

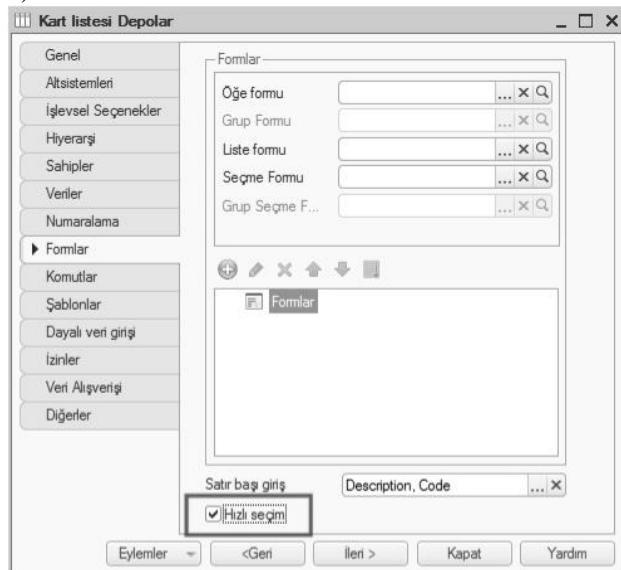
Şirketimizde uygulanan iş sürecine göre malzeme/hizmet listesi *Stok* ve *Hizmetler* bölümlere aittir. Dolayısıyla alt sistem listesinde gerekli alt sistemleri işaretleyelim (Resim 3.56).



Resim 356. Kart listesi ait olduğu alt sistemleri belirtme

“Hızlı seçim” özelliği

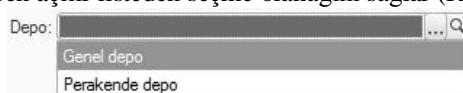
Depolar kart listesinin bir tane daha özelliği tanımlayalım – **Hızlı seçim**. Bunu yapmak için **Formlar** sekmesine geçip Hızlı seçim işaretini yerlestirelim (Resim 3.57).



Resim 3.57. “Hızlı seçim” özelliğini belirtme

Varsayılan değer olarak, kart liste öğesine referans içeren alanın seçim butonuna basıldığında, kart listesinin genel seçim formu ekrana gelir. Bu her zaman kullanışlı olmayıpabilir. Mesela kart liste hiyerarşik bir kart listesi değilse ve içinde birkaç öğe bulunuyorsa seçim formu açmaya gerek olmayıpabilir.

Hızlı seçim özelliği, kart liste öğelerini genel seçim formundan değil, kart liste öğelerini içeren açılır listeden seçme olanağını sağlar (Resim 3.58).



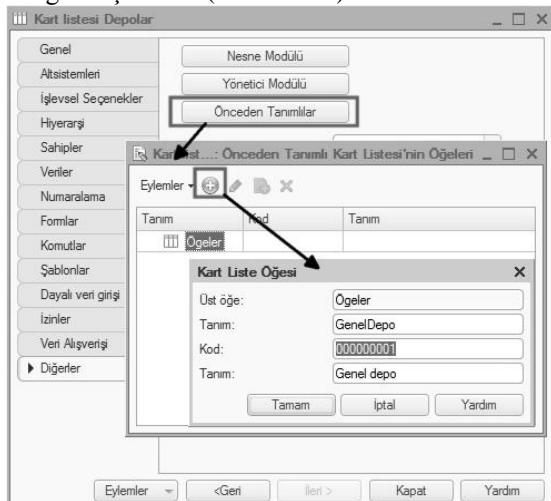
Resim 3.58. Açılan menüsünden depo seçme

Bu seçenek az sayıda olan depolar listesi için uygundur.

Önceden tanımlı öğeler

Digerler sekmesine geçip **Önceden Tanımlılar** butona tıklayalım. Sistem kart listesinin önceden tanımlı öğe listesini açar.

Şu anda liste boş, bu yüzden **Ekle** butona basalım ve **Genel depo** adı ile önceden tanımlı öğe oluşturalım (Resim 3.59).



Resim 3.59. Kart listesinin önceden tanımlı öğesini doldurma

Fark ettiğiniz gibi önceden tanımlı öğesinin iki isim alan mevcuttur. İleride, kaynak kodunu (script) kullanacağımız zaman, tasarımcıda belirlenen adı ile bu önceden tanımlı öğeyi kodlarda kullanabileceğiz. Kullanıcı, 1C:İşletme ortamında önceden tanımlı öğenin tanımı değiştirebilir, fakat tasarımcıda belirlenen adı değiştiremez.

1C:İşletme ortamında çalışmadan önce, kart liste öğelerini hızlı ekleyebilmek için uygulamanın arayüzüne düzeltelim. *Stok* alt sisteminin eylem çubuğuunda yeni depo oluşturma komutu etkinleştirelim.

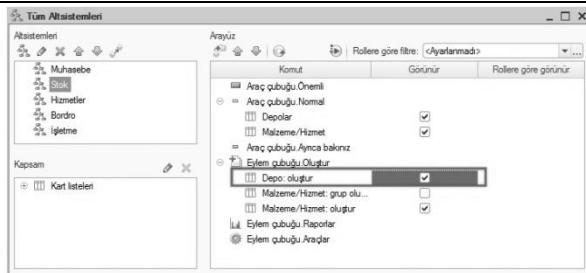
Bunu yapmak için konfigürasyon ağacında *Alt sistemleri* grubu işaretleyip bağlam menüsünden **Tüm alt sistemler** maddesini seçelim.

Açılan pencerede, sol tarafta alt sistem listesinde *Stok* alt sistemini işaretleyelim.

Sağ taraftaki *Arayüz* alanında bu alt sistem ile ilgili tüm komutlar gösterilecektir.

Eylem çubuğu.Oluştur grubunda **Depo: oluştur** komutu işaretleyelim.

Araç çubuğu.Genel grubunda personel listesine ulaşabilmek için **Depolar** komutu eklenmiştir. Varsayılan değer olarak bu komut işaretlenmiş durumdadır (Resim 3.60).



Resim 3.60. Alt sistem ayarlama penceresi

Depolar kart listesini düzeltme penceresini kapatıp, 1C:İşletme sistemini hata ayıklama biçiminde çalıştırılarım. Tasarımcının ekrana getirdiği, konfigürasyon değişikliği ile ilgili onay penceresinde Evet diyelim ve programın otomatik olarak oluşturulan konfigürasyon değişiklik listesi gösterilir. *Depolar* kart listesini eklemiştik.

Uygula butona basalım.

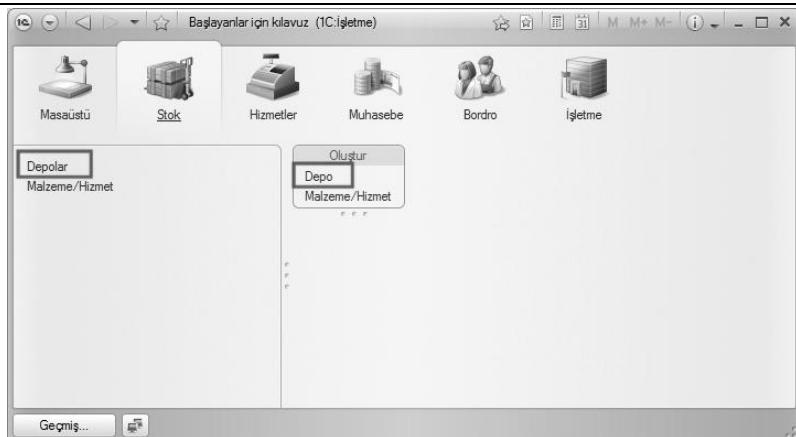


Resim 3.60. Konfigürasyon yapısındaki değişiklik listesi

1C: İşletme Ortamında

Açılan 1C:İşletme penceresinde Stok bölümündeki Eylem çubuklarında yeni Depolar kartını oluşturmak için Depo komutu ortaya çıktı.

Bu komut tanımı, kart listesi için tanımladığımız *Nesne tanımı* özelliği tarafından belirlenir (Resim 3.62).



Resim 3.62. "Stok" bölümü

Hizmetler ve *Stok* bölümlerinin araç çubuklarında *Depolar* kart listesini açmak için **Depolar** komutu yer almıştır.

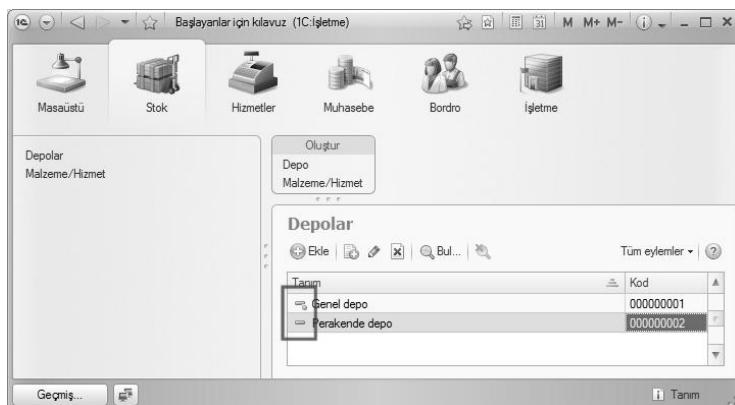
Bu komutun tanımı nesnenin eş anlamlısı tarafından belirlenir, çünkü bu kart listesi için Liste tanımı belirlememiştir (Resim 3.62)..

Stok bölümün araç çubuğundaki *Depolar* komutu çalıştırılınır.

Araç çubuğuının sağ tarafındaki uygulamanın çalışma alanında genel liste formu açılır.

Depolar listesinde *Genel depo* ismi ile bir tane öğe mevcuttur. Bu Tasarımcıda oluşturduğumuz önceden tanımlı öğe.

Eylem çubuğundaki **Depo** komutu uygulayarak kart listesine bir depo daha ekleyelim. Adı **Perakende depo** olarak belirleyelim (Resim 3.63).



Resim 3.63. "Depolar" kart listesinin öğeleri

Burada kart liste oluşturma işlemlerimiz tamamlanmıştır!

'Teori' Önceden tanımlı öğeler

Dikkat ederseniz sistem önceden tanımlı nesneleri farklı resimle göstermektedir.

Normal ve önceden tanımlı kart listelerde kod ve tanım alanı değiştirilebildiğine bilmeksızın, önceden tanımlı kart liste silmek için işaretlendiği zaman sistem otomatik olarak uyarı mesajını görüntüleyecektir (Resim 3.64).



Resim 3.64 Önceden tanımlı kart listesini silmeye çalışıldığında görüntülenen uyarı mesajı

Böylece, önceden tanımlı öğelerin iki özelliklerini tanımlamak mümkündür:

- Önceden tanımlı öğelere konfigürasyon algoritmasından başvurmak mümkün değildir (kaynak kodundan ilgili nesnenin adını kullanmak mümkün);
- Önceden tanımlı öğeler 1C:İşletme ortamında silinemeyen veritabanı nesneleridirler.

Buradan da anlamak mümkün ki, normal nesneler konfigürasyon için daimi olmayan nesnelerdir. Kullanma sürecinde var olabilir veya kaybolabilirler. Konfigürasyon daimi ve daimi olmayan nesneleri birbirinden ayırt edebildiğine rağmen daimi olmayan nesnelere bu özelliklerinden dolayı direkt kaynak kodunda başvuru yapamamaktadır.

Önceden tanımlı öğeler ise tam aksine daimi oldukları için, kullanım sürecinde kaybolma gibi durumları olmadığı için konfigürasyon onlarla güvenli bir şekilde çalışır ve farklı algoritmalar işlendiği süreçte doğrudan olarak onlara başvurabilmektedir. Bunun için her önceden tanımlı nesne benzersiz ada sahiptir.

Ana konfigürasyon ve veritabanı konfigürasyonu

Şuana kadar 1C:İşletme 8 sisteminin yapısı hakkında detaylı açıklama getirmemiştik. Artık ilgili detaylı açıklamayı gösterme zamanı gelmiştir.

Hatırladığınız gibi, kullanıcının gözü ile bakıldığında “1C programı” konfigürasyon ve platform denilen iki tane terimden ibaretti. Belli otomasyonda belli konfigürasyonun kullanıldığı hakkında bilgi vermiştık. Artık bunun tam olarak böyle olmadığını söyleme zamanı gelmiştir.

Bu niye öyle değil? Çünkü her veritabanında en az iki adet konfigürasyon mevcuttur. Çünkü kullanıcı her zaman gerçekten de bir tane konfigürasyona çalışmaktadır. İkincisi ise, geliştirici veya konfigürasyonda değişiklik yapabilen biri (örneğin, veritabanı yöneticisi) için tanımlanan konfigürasyondur. Kullanıcı için bu ikinci konfigürasyon görünmezdir.

Geliştirici için tanımlanan konfigürasyon **Ana konfigürasyon** olarak adlandırılır (veya sadece tasarımcıda düzenleme yaptığımız **Konfigürasyon**).

Kullanıcının çalıştığı konfigürasyon ise **Veritabanı konfigürasyonudur**.

Ana konfigürasyonu düzeltmek mümkündür. Veritabanı konfigürasyonu ise değiştirilmez, sadece ana konfigürasyona bağlı olarak veritabanı konfigürasyonunu güncellemek mümkündür.

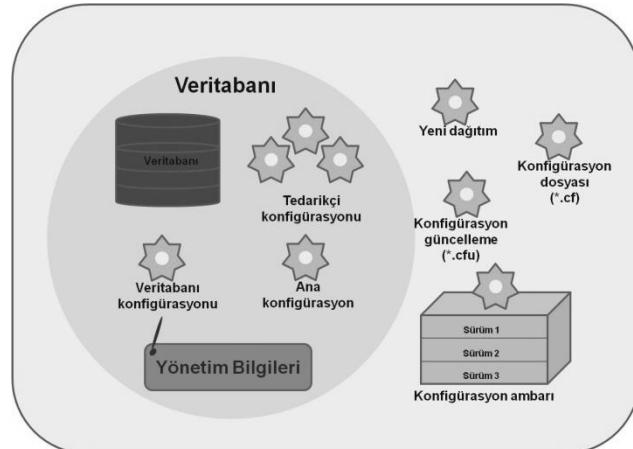
Belki de şu gibi soru aklınıza gelebilir: madem iki tane konfigürasyonumuz var, birini düzeltmek mümkün diğeri ile ise “son kullanıcı” çalışmakta, niye o zaman düzeltilebilen konfigürasyon ana konfigürasyon olarak adlandırılır. Halbuki son kullanıcı tarafından bakıldığından ana konfigürasyon kullanmakta olduğu konfigürasyon olacaktır.

Ana konfigürasyon terimi geliştirici tarafına bakılarak tanımlanmıştır ve bunun derin anımları mevcuttur.

Genel olarak açıklama getirmek gerekirse, 1C:İşletme sistemi ikiden fazla konfigürasyonu içerebilmektedir: ana konfigürasyon, veritabanı konfigürasyonu ve birkaç tane daha tedarikçi konfigürasyonu.

Bundan başka da veritabanının dışında veritabanı ambarı mevcut olabilir. Burada grup halinde geliştirme işlemleri için kullanılan konfigürasyon mevcuttur.

Veritabanının dışında bundan başka da birkaç tane konfigürasyon dosyaları ve yeni konfigürasyon dağıtım dosyaları mevcut olabilmektedir (Resim 3.65).



Resim 3.65 Konfigürasyon yapısı

Veritabanında bulunan tedarikçi dağıtımları konfigürasyonu bir önceki dağıtım durumunu içermektedir. Konfigürasyon bazen birkaç tedarikçi konfigürasyonlarının desteğinde bulunabilir. Her tedarikçi konfigürasyonu ise sadece kendine ait kısımları içerir ve ayrı konfigürasyon şekilde saklanır. Böylece, veritabanı birkaç adet konfigürasyonu daha içerir.

Yeni dağıtım dosyaları konfigürasyon dosyaları şeklinde (**tam dağıtım dosyası**) ve güncelleme dosyaları (**güncelleme dağıtım dosyası**) şeklinde mevcut olabilirler.

Konfigürasyon ambarı grup halinde geliştirme işlemleri için tanımlanmıştır. Bu konfigürasyon tek konfigürasyon halinde değil de, konfigürasyon sürümünde ayrı nesneler halinde saklanırlar. Böylece konfigürasyon ambarından istenilen sürümü elde etmek mümkündür – bunun o için gerekli sürümlerden derlenirler.

Şimdi ise bu konfigürasyonlar arasında karşılaştırma ve güncelleme imkanlarının bulunduğu düşünün. Bu durumda bahsi geçenler arasında karışıklığı önlemek için geliştiricinin kullandığı konfigürasyona **Ana konfigürasyon** denilmiştir.

Şimdi ise ana konfigürasyon ve veritabanı konfigürasyonuna geri dönerek şunu belirtmek de gereklidir, konfigürasyonun iki yapıdan ibaret olması, kullanıcının işini durdurmadan konfigürasyonda değişiklik gerçekleştirmeye açısından faydalıdır.

Onun yanında ise geliştirici gerçekleştirdiği değişikliklerden doğruluğundan emin olduğu zaman ana konfigürasyonu kullanarak veritabanını güncelleyebilir.

Eğer yapılan değişiklikler veritabanı yapısını etkilemiyorsa (örneğin, tablolar değişmediyse, sadece belli modülde kaynak kodu değiştiyse),

kullanıcı çalışmasını bölmeden konfigürasyonu güncellemek mümkündür. Buna **dinamik olarak güncelleme** denir.

Kullanıcılar bu değişiklikleri sadece uygulamayı tekrar çalıştırdıklarında görebilirler. **Kaynak kodunda**

DataBaseConfigurationChangedDynamically() metodu ile uygulamanın yeniden başlatılması gerektiğini belirtmek mümkündür.

Ama eğer değişiklikler veritabanı yapısı ile ilgili ise, örneğin, kart listesine yeni bir öznitelik eklendiye veya mevcut olan özniteliğin türü değiştiye, tüm kullanıcıların çalışmalarını tamamlandırmak gereklidir. Geliştirici her zaman ana konfigürasyon ile veritabanı konfigürasyonunu karşılaştırabilir, veritabanı konfigürasyonunu kullanarak ana konfigürasyonun eski haline geri gelebilir (örneğin, yaptığı değişiklikleri kariştırıya iyice).

Böylece iki konfigürasyon arasındaki etkileşimi aşağıdaki resim olduğu gibi açıklamak mümkündür (Resim 3.66).



Resim 3.66. İki konfigürasyon arasındaki etkileşim

Geliştirici ana konfigürasyon ile çalıştığı zaman sistem, çalışmakta olduğu konfigürasyon kaydedilmiş konfigürasyondan farklı olduğu hakkında ve kaydedilen konfigürasyonun veritabanı konfigürasyonundan farklı olduğu hakkında bilgi verir.

Eğer geliştirici ana konfigürasyon ile çalışmaktadır ise ve çalıştığı ana konfigürasyon daha önce kaydedilmiş olan ana konfigürasyondan farklı ise konfigürasyon ağacı penceresinin başlığında konfigürasyonda değişiklik yapıldı hakkında belirti (*) görüntülenir (Resim 3.67).



Resim 3.67. Konfigürasyon ağacı penceresi başlığı

Eğer kaydedilen konfigürasyon veritabanı konfigürasyonundan farklı ise konfigürasyon ağacının pencere başlığında konfigürasyonun farklılığı hakkında belirti (<!>) görüntülenir (Resim 3.68).



Resim 3.68. Konfigürasyon ağacı penceresi başlığı

Konfigürasyonu kaydetmek için Konfigürasyon > Konfigürasyon kaydet komutunu kullanmak gereklidir, konfigürasyonu güncellemek için ise

Konfigürasyon > Veritabanı konfigürasyonunu güncelle komutunu kullanmak gerekir.

Hata ayıklaması > Hata ayıklamasını başlat komutu kullanıldığı zaman sistem otomatik olarak konfigürasyonu kaydeder ve ardından veritabanı konfigürasyonu ile karşılaşır. Konfigürasyon değişik olduğu durumda konfigürasyonun güncellenip güncellenmeyeceği hakkında soru görüntülenir.

Hata ayıklaması > Hata ayıklamasına devam et komutu kullanıldığı zaman ise sistem yukarıda sıralanınların yanında bir de hata ayıklaması biçiminde çalıştırılmış olan uygulamayı yeniden başlatıp başlatılmayacağı hakkında uyarı verir.

Burada önemli bilgiden bahsetmekte fayda vardır, veritabanı konfigürasyonu güncellendiği sırada sistem, konfigürasyon nesnesi olarak tanımladığımız şekilde veri saklama yapısını veritabanında oluşturur (düzeltir).

Böylece, Kart listesi konfigürasyon nesnesi olarak tanımlanan veritabanı yapısında kart listesinin normal öğelerini kullanıcı ekler. Önceden tanımlı nesneler ise sistemin kendisi eklemektedir.

Daha önce bahsettiğimiz konuya bir daha tekrar edelim, eğer eklenmiş olan veriler (öğeler) konfigürasyon için önemli değilse, önceden tanımlı nesneler önemlidir, çünkü bu nesneler konfigürasyonun çalışma algoritmasında kullanılabilmektedir.

Sorular

- ✓ Kart listesi konfigürasyon nesnesi ne için kullanılır?
- ✓ Kart listesinin özellikleri nedir?
- ✓ Kart listes öznitelikleri ve tablo bölümleri ne için kullanılır?
- ✓ Hiyerarşik kart listeleri ne için kullanılır ve üst öğe nedir?
- ✓ Alt kart listeleri ne için kullanılır ve sahip nedir?
- ✓ Kart listesinin genel formları nelerdir?
- ✓ Önceden tanımlı nesne nedir?
- ✓ Konfigürasyon açısından normal kart liste öğelerinin önceden tanımlı öğelerden farklı nedir?
- ✓ Kullanıcı önceden tanımlı öğeleri nasıl fark edebilir?
- ✓ Yeni Kart listesi konfigürasyon nesnesi nasıl oluşturulur ve oluşturulan nesnenin özellikleri nasıl tanımlanabilir?
- ✓ Kart listesine yeni öğe nasıl eklenir?
- ✓ Yeni kart liste grubu (klasörü) nasıl eklenir?
- ✓ Kart liste öğeleri bir gruptan diğer gruba nasıl taşınabilir?
- ✓ Kart liste öznitelikleri için doldurma denetimi ne

DERS 4

Evraklar

Süre

Dersin tahmin süresi – 1 saat 30 dakika

Evrak nedir?	98
Evrak formları	100
“Teori”. Veri türleri Tür oluşturan konfigürasyon nesneleri	100
Alım İrsaliyesi Evrakı	101
Evrak ekleme	102
Referans türünde öznitelikler	104
Konfigürasyon nesne özniteliğinin “Doldurma değeri”	104
özelliği	104
Tablo bölümü doldurma denetimi	106
Alım ırsaliyeleri ekleme	109
Evrak satırında tutarın otomatik olarak hesaplanması	112
Evrak formu	113
Olay işleyicisi	116
Birden fazla olayın işlenmesi için tek prosedür	119
Genel modül	119
“Hizmet faturası” evrakı	122
Soru	128

Bu derste *Evrak* konfigürasyon nesnesi ile tanışacağız. Bu nesne ne iş yaptığı, yapısı ne olduğunu ve genel özelliklerini ile ilgili bilgiye sahip olacaksınız.

Ondan sonra birkaç evrak oluşturacağız ve bazı eylemlerin gerçekleştirilmesi için geliştirici tarafından algoritmalar nasıl belirlenebildiğini göstereceğiz.

Bunun dışında evrak formu nasıl oluşturacağımı, kaynak kodunun bazı yapılar ve tip oluşturan konfigürasyon nesneler hakkında bilgi edinmiş olursunuz.

Evrak nedir?

Evrak konfigürasyon nesnesi işletmede gerçekleştirilen ekonomik, muhasebesel veya diğer olguları programa kaydetmek için kullanılan öğe türüdür. Genelde her işletmede ırsaliye, fatura, fiş, makbuz gibi evraklar kullanılır. Evrakların yapısı ve özellikleri *Evrak* konfigürasyon nesnesinde tanımlanır, platform bu detaylara göre evrakların bilgilerini depolamak için veritabanında gerekli tabloları oluşturur.

Evrakların çalışma mantığı diğer konfigürasyon nesnelerin çalışma mantıklarından farklıdır. *Evrak kaydetme (posting)* özelliğe sahip. *Evrak* kaydedildiğinde, yansımış olduğu olgu programın hesaplamalarında ve raporlarında yer alır.

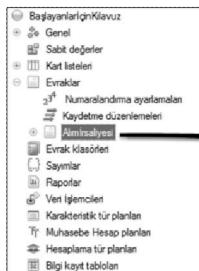
Evrak kaydetmediği sürece, programına hesapları değişmez. Bu durumda evrak sadece taslak veya kalıp olarak kullanılabilir. *Evrak* kaydedildiği anda içinde bulunan bilgiler gerekli hesaplarda yansıtılmış olur ve ilgili borç/alacak durumlar değişir.

Programın hesaplarına değişiklik yaptığı için *evrak* her zaman belli bir tarihe bağlıdır. Böylece veritabanında gerçekleşmiş olgularına sırası takip edilebilmektedir.

Yukarıdaki özellikler sayesinde, 1C:İşletme sistemi hesapların işlemesi doğru olup olmadığını kontrol etme araçlarına sahiptir. Örneğin, önceden kaydedilmiş evrakı düzeltməz ve tarihi değişmeden (geçmiş tarihi ile) evrakı kaydetmiş olduğumuzu var sayalım. 1C:İşletme sistemi kaydetmiş olduğumuz evrak sonraki evrakları etkileyip etkilemediğini takip edebilir ve gerek duyulduğunda ilgili evrakları tekrar kaydedebilir.

Çalışma sürecinde kullanıcı yeni evrakları oluşturabilir – alış ve satış ırsaliyeler, faturalar vs.

Veritabanındaki her evrak, evrakin yapı bilgilerini içeren genel tabloda bir kayittır (Resim 4.1).

Tasarımcı**1C:İşletme**

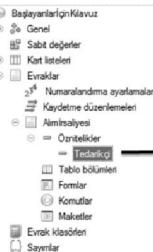
Alım ırsaliyesi		
Tarih	Numara	Tüm eylerler
27.02.2011 18:48:03	000000001	
27.02.2011 18:48:06	000000002	
27.02.2011 18:48:07	000000003	

Veritabanı

Tarih	Numara
27.02.2011 18:48:03	000000001
27.02.2011 18:48:06	000000002
27.02.2011 18:48:07	000000003

Resim 4.1. Tasarımcı ortamında, 1C:İşletme ortamında ve veritabanında “Alım ırsaliyesi” evrakinin standart öznitelikleri.

Genelde evrak, türüne göre, daha detaylı bilgilere sahiptir. Örneğin, her alım ırsaliyesi ürün tedarikçisi, depo ve ürün ile ilgili bilgi içerebilir. Bu tür bilgiler aynı türde sahip evraklar için aynıdır. Evrakın içeriğini tanımlamak için konfigürasyon nesne öznitelikleri kullanılmaktadır. Coğu öznitelikler geliştirici tarafından geliştirilir, fakat her evrakin standart öznitelikler de vardır. Onlardan en önemli iki öznitelik: *Tarih* (Date) ve *Numara* (Number). *Tarih* veri türü saniye içeren tarih olduğu için bu öznitelik zaman çizgisinde konumunu belirler. (Resim 4.2).

Tasarımcı**1C:İşletme**

Alım ırsaliyesi		
Ekle	Düzenle	Tüm eylerler
27.02.2011 19:23:18	000000001 "Standart" LTD	
27.02.2011 19:23:36	000000002 "Otomasyon" Şti	
27.02.2011 19:23:49	000000003 "Altant" Şirketi	

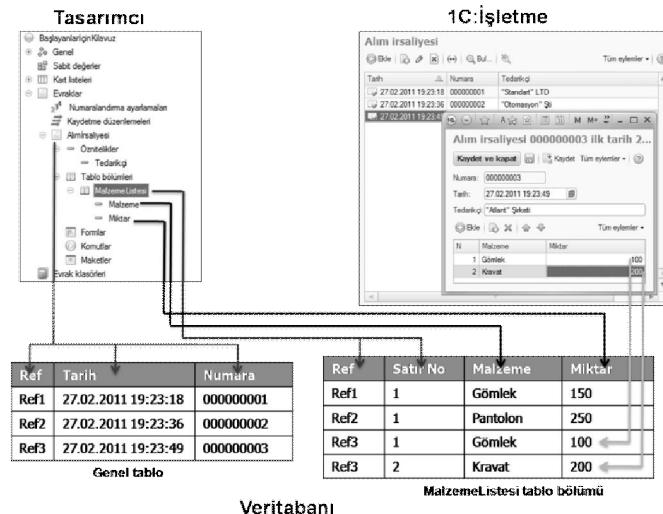
Veritabanı

Tarih	Numara	Tedarikçi
27.02.2011 19:23:18	000000001	"Standart" LTD
27.02.2011 19:23:36	000000002	"Otomasyon" Şti
27.02.2011 19:23:49	000000003	"Altant" Şirketi

Resim 4.2. Tasarımcı ortamında 1C:İşletme ortamında ve veritabanında “Alım ırsaliyesi” evrakinin “Tedarikçi” özniteligi.

Bunun dışında her evrak genelde, aynı yapıya sahip ve birden fazla sayıda belirlenen bir birine benzer bilgiye sahip. Örneğin, Alım ırsaliyesi stoka alınan malzemeler listesini içerebilir.

Böyle bir bilgiyi tanımlamak için *Evrak* konfigürasyon nesnesinin tablo bölgümleri kullanılmaktadır. Tablo bölgümleri tanımlandığı durumda, belli bir evraka bağlı olan tablolar oluşturulacaktır ve bu tablolarda tablo bölümdeki bilgiler kaydedilecektir (Resim 4.3).



Resim 4.3. Tasarımcı ortamında 1C:İşletme ortamında ve veritabanında “Alım İrsaliyesi” evrakinin “Malzeme listesi” tablo bölümü.

Evrak formları

Evraklı görsel şeklinde ekranda yansıtmak için birkaç genel form mevcuttur.

Bağlam menüsünde ve özellik panosunda	Form yapılandırıcısında	Formlar sekmesinde
Öğe formu	Evrak formu	Evrak formu
Liste formu	Evrak liste formu	Liste formu
Seçme formu	Evrak seçme formu	Seçme formu

“Teori”. Veri türleri Tür oluşturan konfigürasyon nesneleri

Evrakları oluşturmadan önce, 1C:İşletme sisteminde hangi veri türleri kullanılabilirliğinden bahsetmek gereklidir.

Geçen derste kart liste öznitelikleri türü belirlerdik. Belirlediğimiz veri türü *basit* veri türleri idi; *Number*, *String*, *Date* ve *Boolean*. Basit veri türleri programda standart olarak belirlenmişti ve basit veri türleri değiştirilemez.

Böyle basit veri türleri yanı sıra her konfigürasyonda kendine özel veri türleri olabilir. Yani, konfigürasyonda sürekli bulunmayan veri türleri, belli bir konfigürasyon nesnesi oluşturulduğunda ortaya çıkan veri türleri.

Örneğin, *Depolar* kart listesi konfigürasyon nesnesi oluşturulduktan sonra bu kart listesine bağlı olan birkaç yeni veri türü oluşturmuştu. Mesela, **CatalogRef.Depolar**. Ve şimdi yeni oluşturulan öznitelik için CatalogRef.Depolar veri türü belirlediğimizde bu öznitelikte *Depolar* kart listesindeki belli bir öğeye referans depolanabilir.

Yeni veri türü oluşturabilen konfigürasyon nesnelerine *tür oluştururan* konfigürasyon nesneleri denir.

Örneğin, *MalzemeHizmet* kart listesi oluşturulduktan sonra aşağıdaki veri türleri oluşturuluyor;

- CatalogsManager.MalzemeHizmet.
- CatalogRef.MalzemeHizmet
- CatalogObject.MalzemeHizmet
- CatalogSelection.MalzemeHizmet

Bu veri türleri platformda her zaman bulunmaz, sadece belli bir konfigürasyonda mevcuttur (*MalzemeHizmet* kart listesi içeren konfigürasyon).

Birinci evraki oluşturacağımızda *Depolar* ve *MalzemeHizmet* kart listesi oluşturulduktan sonra konfigürasyonumuzda meydana gelen *CatalogRef.Depolar* ve *CatalogRef.MalzemeHizmet* veri türleri kullanacağız.

Alım İrsaliyesi Evraki

Evrak konfigürasyon nesnesi hakkında biraz bilgi aldıktan sonra “Aspirin usta” işletmedeki olguları yansıtmak için konfigürasyonumuzda birkaç evrak oluştururalım.

İşletmemizde en çok talep edilen hizmet – televizyon tamiri ve çamaşır makine kurulumudur. Her iki durumda sarf edilecek malzemeler gerekir. Bu yüzden işletmemizin iki önemli olgu türü mevcuttur; malzeme alımı ve hizmet verme.

Bu olguları yansıtmak için veritabanında iki evrak oluşturacağız: *Alım Irsaliyesi* ve *Hizmet faturası*.

Alım Irsaliyesi stokumuza gerekli malzemelerin gelişini kaydedecek, Hizmet faturası ise verilen hizmet detaylarını ve hizmet verme sırasında sarf edilen malzemeleri kaydedecek.

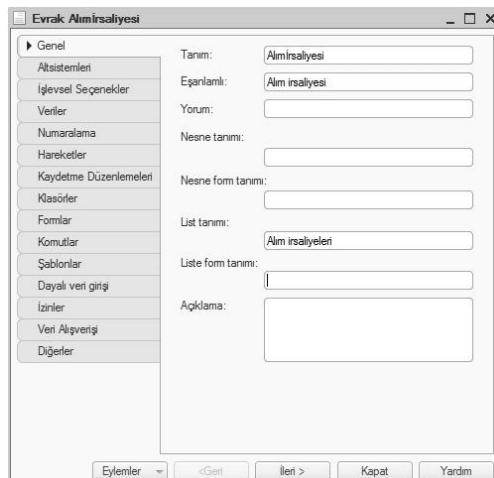
Tasarımcı ortamında

Evrak ekleme

Tasarımcı açıp yeni *Evrak* konfigürasyon nesnesini ekleyelim. Genel sekmesinde evrakin tanımı belirleyelim – **Alım İrsaliyesi**. Tanım üzerinde platform eş anlamlı alanı otomatik olarak doldurur.

Aynı sekmede 1C:İşletme arayüzünde evrak nasıl yansıtılacağını belirleyelim. *Nesne tanımı* belirlemeyeceğiz, çünkü eş anlamlı zaten tekli olarak girilmiştir.

Liste tanımı çoklu olarak belirleyelim; *Alım İrsaliyeleri* (Resim 4.4).

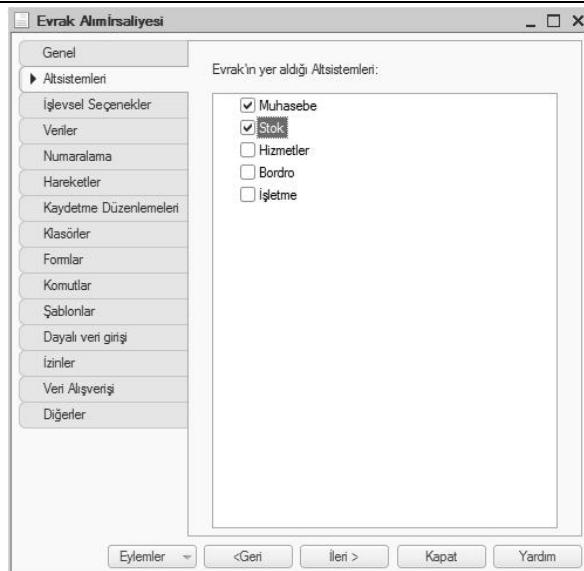


Resim 4.4 Evrakin genel özelliklerini belirtme

İleri butona tıklayalım ve Altistemleri sekmesine geçelim.

Konfigürasyonumuzun mantığına göre alım İrsaliye listesi *Stok* ve *Muhasebe* bölümlerde gösterilmelidir.

Dolayısıyla, gerekli alt sistemleri işaretleyelim (Resim 4.5).



Resim 4.5. Evrap yansıtılacak alt sistemleri belirtme

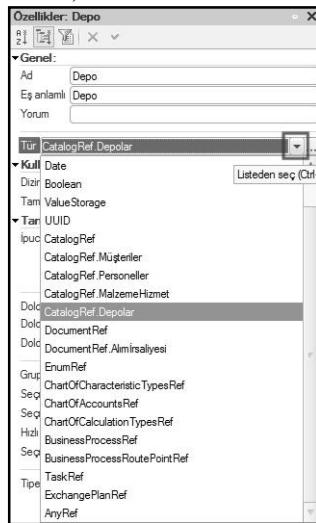
Veriler sekmesine geçip evrakın yeni özniteligi oluşturalım – **Depo**. Bunu yapmak için evrap öznitelik listesi üzerinde bulunan **Ekle** butona basmak yeterlidir.



Resim 4.6. Evrap özniteligi oluşturma

Referans türünde öznitelikler

Öznitelik için referans veri türünü seçelim **CatalogRef.Depolar**. Bu veri türü, *Depolar* kart listesi konfigürasyon nesnesini oluşturuktan sonra ulaşılabilir olmuştu (Resim 4.7).



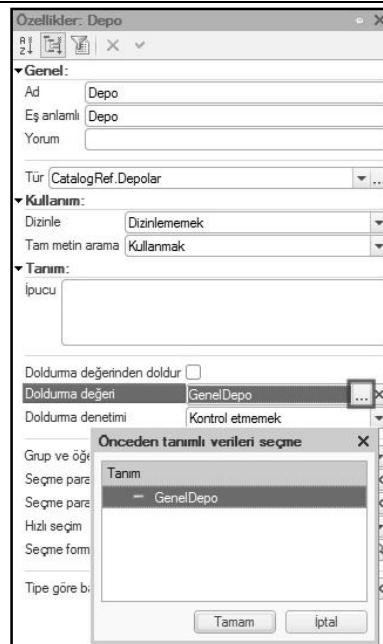
Resim 4.7. Evrak özniteliği oluşturma

Konfigürasyon nesne özniteliğinin “Doldurma değeri” özelliği

Kullanıcıların işi nasıl kolaylaştırılabileceğini gösterelim. Otomasyon sağladığımız şirkette tüm alınan mallar genelde Genel depoya alınır.

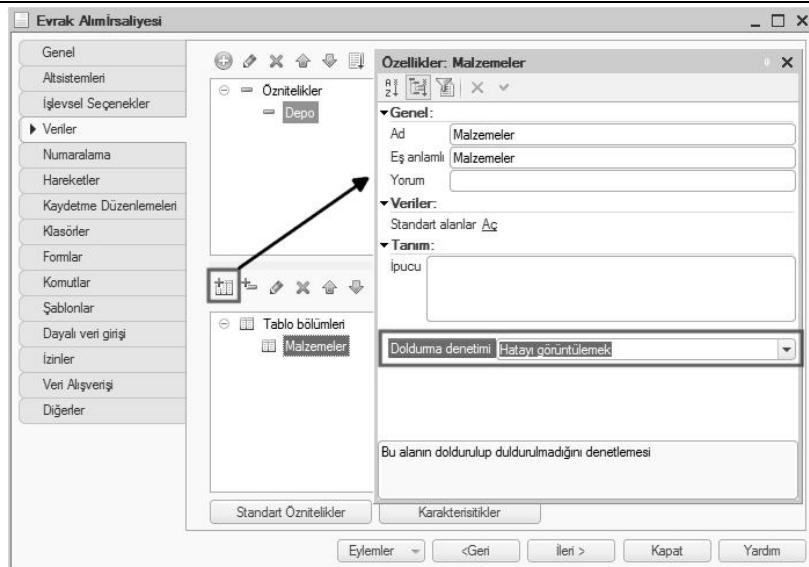
Bu yüzden oluşturduğumuz *Depo* özniteliğin özellik panosunda *Doldurma değeri* özelliği bulalım.

Özellik değeri olarak önceden tanımlı *Depolar* kart liste öğesini seçelim – **Genel depo**.



Resim 4.8. “Depo” özniteligi için varsayılan doldurma değerini seçme
Böylece, yeni evrak oluşturulduğunda depo alanına otomatik olarak *Genel depo* gelir ve kullanıcının bunu manüel olarak yapmasına gerek kalmaz.

Bundan sonra evraka tablo alanı ekleyelim, tanım – **Malzemeler**. Bunu yapmak için evrakin tablo bölüm listesinin üstünde bulunan **Tablo bölümü ekle** butonuna basalım (Resim 4.9).



Resim 4.9. Evrakın yeni tablo bölümünü ekleme

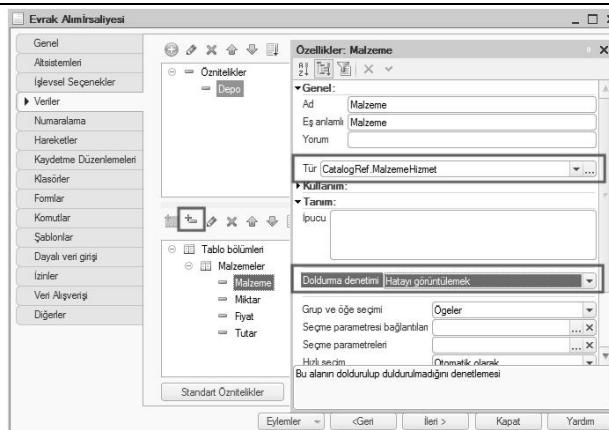
Tablo bölümü doldurma denetimi

Tablo bölüm tanımı dışında **Doldurma denetimi** özelliğini *Hatayı görüntüleme* olarak belirleyelim. Böylece, alım ırsaliyesinde kesinlikle *Malzemeler* tablo bölümünü doldurmak gerektiğini belirtiyoruz, aksi halde, *Malzemeler* tablo bölümünde hiçbir kayıt olmasa, stoka alının anlamı kalmaz. Tablo bölümünde kayıt bulunmadığı durumda, ekrana hata mesajı gelir ve evrak kaydedilmez.

Malzemeler tablo bölümünün öznitelikleri oluşturalım. Bunu yapmak için tablo bölüm listesinin araç çubuğu'nda *Öznitelik ekle* butona tıklamak yeterlidir (Resim 4.10).

- **Malzeme**, tür *CatalogRef.MalzemeHizmet*,
- **Miktar**, tür *Number*, Uzunluk 15, Ondalık 2, *Negatif olamaz*,
- **Fiyat**, tür *Number*, Uzunluk 15, Ondalık 2, *Negatif olamaz*,
- **Tutar**, tür *Number*, Uzunluk 15, Ondalık 2, *Negatif olamaz*.

Tablo bölümünün her özniteligi için **Doldurma denetimi** özelliğini **Hatayı görüntüleme** olarak belirleyelim. Böylece, evrak kaydedildiği anda, sadece tablo bölüm değil, tablo bölümündeki her özniteligine veri doldurma denetimi yapılacaktır.



Resim 4.10. Evrak tablo bölümünün özniteliklerini ekleme

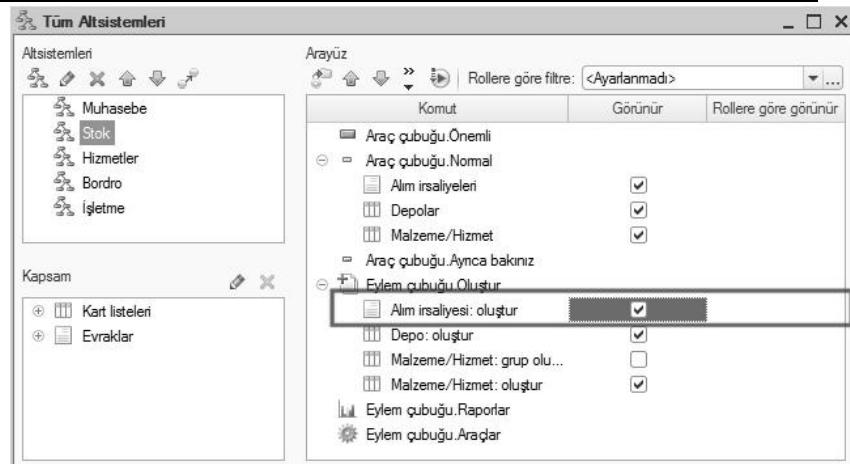
Numaralama sekmesine geçelim ve **Otomatik numaralama** özelliği işaretlenmiş olduğundan emin olalım. Bu özellik, oluşturulan evraklara otomatik olarak eşsiz numara tanımlanmasını sağlar.

Son olarak uygulama arayüzüni düzeltelim. Stok alt sisteminde, yeni evrak oluşturma komutu göstermek gereklidir.

Bunu yapmak için konfigürasyon nesne ağaçında *Alt sistemler* grubu işaretleyip sağ tuş ile çağırılan menüden **Tüm alt sistemler** maddesini seçelim. Ekrana gelen pencerede soldaki listeden *Stok* alt sistemi işaretleyelim.

Sağ tarafta bulunan *Arayüz* listesinde seçilen alt sistemini tüm komutlar gösterilir.

Eylem çubuğu. Oluştur grubunda **Alım ırsaliyesi: oluştur** komutu işaretleyelim (Resim 4.11).



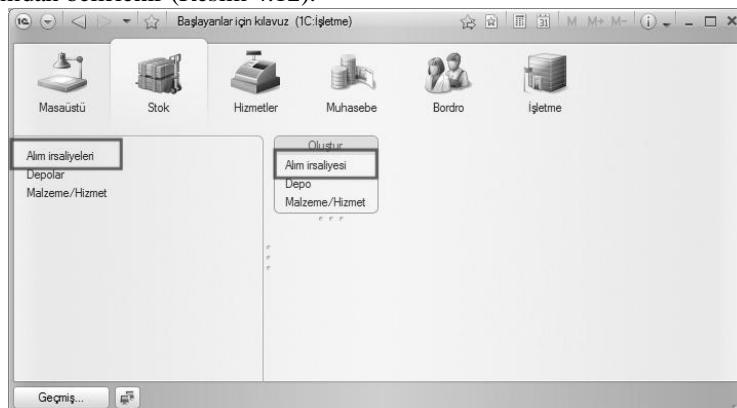
Resim 4.11. Arayüz ayarlama

1C:İşletme ortamında

1C:İşletme sisteminin hata ayıklama biçiminde çalıştırılacak.

Açılan 1C:İşletme penceresinde Stok ve Muhasebe bölümlerin araç çubuklarında Alım ırsaliye listesini görüntülemek için *Alım İrsaliyeleri* komutu ortaya çıktı.

Bu komut tanımı, kart listesi için tanımladığımız *Liste tanımı* özelliği tarafından belirlenir (Resim 4.12).



Resim 4.12. "Stok" bölümü

Aynı şekilde Stok bölümün eylem çubuğunda yeni alım ırsaliye evrakını oluşturmak için **Alım ırsaliyesi** komutu yer aldı. Bu komutun tanımı eş

anlamlı tarafından belirlenir, çünkü bu konfigürasyon nesnesi için *Nesne tanımı* belirlenmemiştir.

Alım ırsaliyeleri ekleme

Şu anda veritabanımızda alım ırsaliyeler yok, bu yüzden *Stok* bölümünün eylem çubuğuunda bulunan *Alım İrsaliyesi* komutu uygulayarak yeni alım ırsaliyesi oluşturalım.

Ekranımıza evrakin formu gelir – genel öğe formu. Formun başlığı nesnenin eş anlamlısı ile aynıdır.

Sistem otomatik olarak geçerli tarihini doldurur, ancak saat sıfır olarak belirlenir, çünkü evrak daha kaydedilmemiş. Evrak gerçek zamanlı kaydedildiğinde evraka o anlık saatı doldurulur.

Numara alanı boştur, fakat sistem otomatik olarak eşsiz numarayı oluşturur, çünkü bu evrak için **Otomatik numaralama** özelliği varsayılan değer olarak geçerlidir. Yeni numara kaydedildiğinde otomatik olarak doldurulacaktır.

Öznitelik özelliklerinde belirlediğimiz gibi *Depo* alanına *Genel depo* otomatik olarak doldurulmuştur.

Resimde gösterildiği gibi evrakin tablo bölümünü dolduralım (Resim 4.13).

N	Malzeme	Miktar	Fiyat	Tutar
1	Flyback transformer GoldStar	10,00	11,00	110,00
2	Flyback transformer Samsung	10,00	25,00	250,00
3	Transistor Philips 2N2369	10,00	0,50	5,00

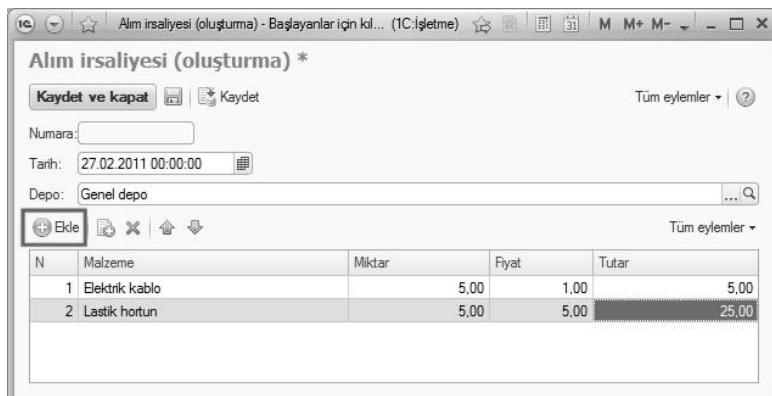
Resim 4.13. “Alım ırsaliyesi No 1” yeni evraki oluşturma

Tablo bölümünde bulunan *Malzeme* alanının seçme butonuna basıldığında, Malzeme/Hizmet kart listesinin seçim formu ekrana gelir. Çünkü tablo bölümündeki *Malzeme* özniteligin veri türü *CatalogRef.MalzemeHizmet* belirtmiştir.

Kaydet ve kapat tıklayalım.

Evrak saklanmış ve kaydedilmiştir. Evrak için eşsiz numara oluşturulmuş ve evrak tarihi olarak geçerli tarih ve saat otomatik olarak belirlenmiştir.

Aynı şekilde çamaşır makineler ile ilgili malzemeleri almak için ikinci evrak oluşturalım. Fakat bu kez *Malzeme* alanında seçim butonu kullanmayıp, alana direkt malzemenin tanımını girmeyi çalışalım. Platform otomatik olarak, tanımın ilk harflerine göre, malzemeleri bulur ve seçmek için seçenekleri sunar.

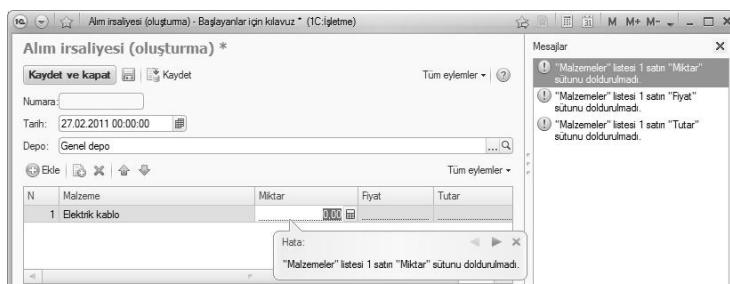


Resim 4.14. Alım İrsaliyesi N o 2” evrakını ekleme

Kaydet ve kapat tıklayalım.

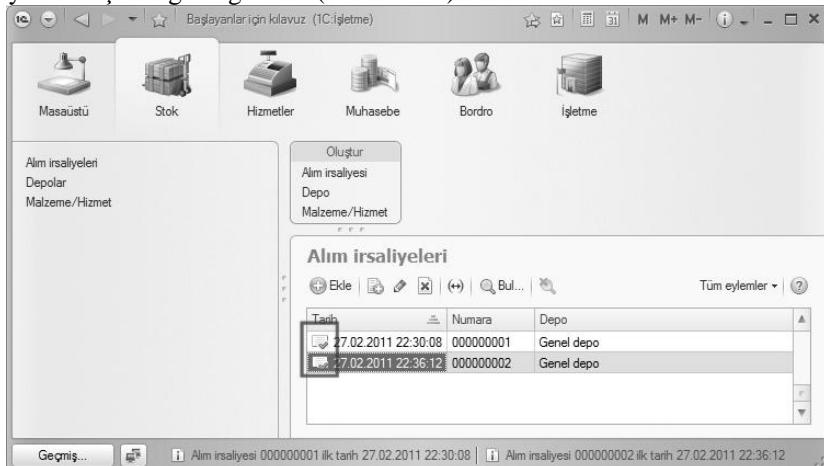
Evrak saklanacak ve kaydedilecektir. Evrak için eşsiz numara oluşturulacak ve evrak tarihi olarak geçerli tarih ve saat otomatik olarak belirlenecektir.

Yeni evrak eklendiğinde, tablo bölümü ve tablo bölümündeki her sütünün veri giriş alanı kırmızı noktalı çizgi ile işaretlenmiştir. Kırmızı noktalı çizgi ile işaretleme öğeler için doldurma denetimi belirlendiğini ifade eder. Tablo bölümüne hiç satır girmeyip veya eklenen satırda herhangi bir alanı doldurmayıp evrak kaydetmeye çalıştığında ekrana hata mesajları gelir ve evrak kaydedilmez (Resim 4.15).



Resim 4.15. Yeni evrak ögesi eklendiğinde veri doldurma denetim hataları
Eklenmiş evrak listesine ulaşmak için araç çubuğuunda *Alım İrsaliyeleri* komutu uygulayalım.

Uygulamanın çalışma alanında açılan liste forumda eklemiş olduğumuz iki evrak görüyoruz. Soldaki yeşil tık (listedeki kayıt satırında), evrakların kaydedilmiş olduğunu gösterir (Resim 4.16).



Resim 4.16. Alım İrsaliye listesi

“Teori”. Kart listeleri ve evraklar

Kart listesi ve **evrakları** isimlendirme yöntemi hakkında açıklama yapalım, faydalı olacaktır.

Eğer standart uygulamanın konfigürasyon ağacını açıp bakarsanız göreceksiniz, bütün Kart listeleri çoğul biçimde, bütün Evraklar ise tekil biçimdedir.

Tasarımcı ortamında yeni kart listesi eklendiği zaman bu kart listesi tanımına çoğul eki eklenir (Malzemeler). Tanımı çoğul eki olsa bile, bu kart liste belli bir malzemeyi tanımlayan (tekil ekli) öğelerden ibaret olduğu kastedilmektedir.

Tasarımcı ortamında yeni Evrak eklendiğinde bu evraka tekil ekli tanım (Alımİrsaliyesi) belirtiyoruz. Aslına bakılırsa, evraklar aynen kart listeleri gibi bir nevi «ambar» anlamındadır. Bu ambarın her bir nesnesi bir evrak, bir alım ırsaliyesi anlamına gelir (tekil ekli). Olaya kavramsal açıdan bakıldığından aynen kart listelerinde olduğu gibi evraklara da çoğul eki eklenmesi gerekecekti (örneğin, Alımİrsaliyeleri).

Ama insanın genel psikolojisi konfigürasyon ağacının Evraklar dalını açtığı zaman çoğul değil de, tekil biçimli bir liste görmek ister. Bunun sebebi ise, gerçek hayatı evrakların seti için belli bir terim bulmak çok zordur (tek türden kayıtları içeren set için terim bulmak çok daha kolaydır – kart listesi, plan vb.). Bunun için konfigürasyon ağacının Evraklar dalı çoğul şekildedir, onun altındaki öğeler / nesneler ise tekil biçimde tanımlanır ve bu nesnelerin her biri tek türden evrakların listesi anlamına gelir.

Evrak satırında tutarın otomatik olarak hesaplanması

Evrakları eklerken tutarın el ile girmek gerektiğini fark etmişsinizdir. Bu pek rahat değil. Evrak işleme sürecini otomasyonlaştırmak gereklidir, yani tablo bölümündeki miktar veya fiyat değeri değiştirildiğinde tutarın otomatik olarak hesaplanmasılığını sağlamak gereklidir.

Bu hiç zor değil. Bunu yapmak için ilk önce evrak için form oluşturmak ve kaynak kodun sağladığı olanakları kullanmak gereklidir.

Şu ana kadar, 1C:İşletme platformunun otomatik olarak oluşturmuş olduğu, standart nesne formları kullanıyorduk. Şimdi formun standart çalışma mantığını biraz değiştirmek gereklidir, dolayısıyla *Alımİrsaliyesi* için bir form oluşturmak lazımdır. Oluşturulan formda kaynak kodu kullanarak gereklili algoritma tanımlayabileceğiz. Ve sistem oluşturduğumuz formu varsayılan form olarak kullanacaktır.

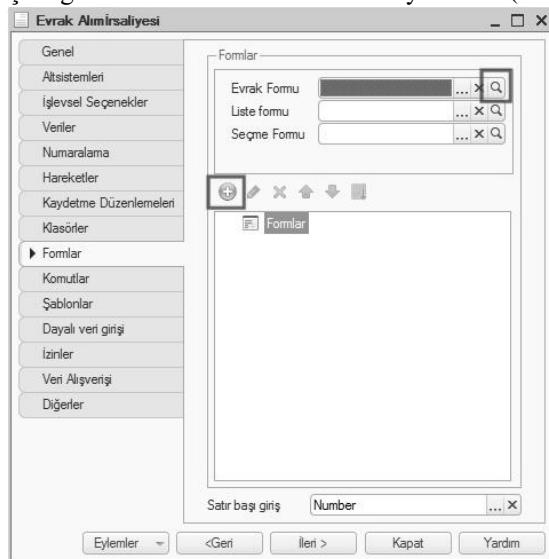
Tasarımcı ortamında

Evrak formu

Tasarımcıya geri dönelim ve *Alımİrsaliyesi* evrakinin düzeltme penceresini açalım.

Formlar sekmesine geçelim.

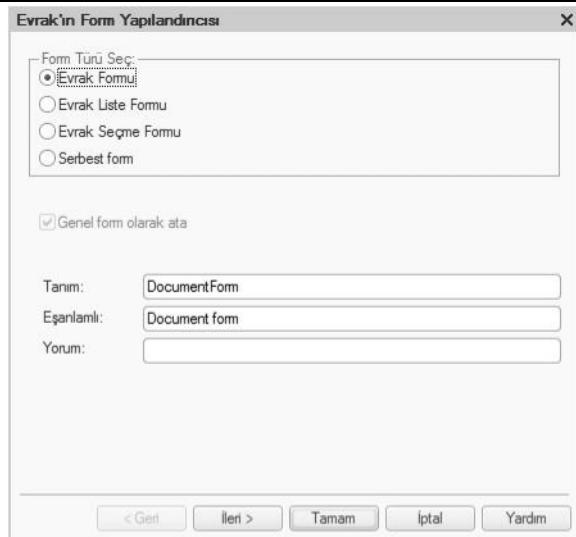
Gördüğümüz gibi evrakin genel formları belirlenmemiş. Evrak formunu oluşturmak için veri giriş alandan açma butonuna (büyüteç) veya form listesinin araç çubuğundaki **Ekle** butonuna basmak yeterlidir (Resim 4.17).



Resim 4.17. Evrak form oluşturmazı

Sistem, geliştirici için çok faydalı olan bir araç ekrana getirecek – *Form yapılandırıcı* (Resim 4.18).

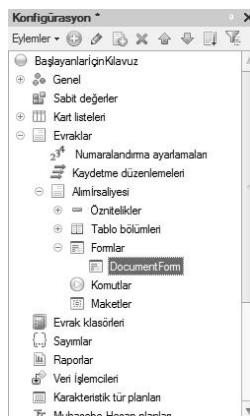
Form yapılandırıcı “sihirbaz” mantığı ile geliştirilmiştir; *İleri* ve *Geri* butonları kullanarak belli bir sırayla form özellikleri tanımlamak gereklidir.



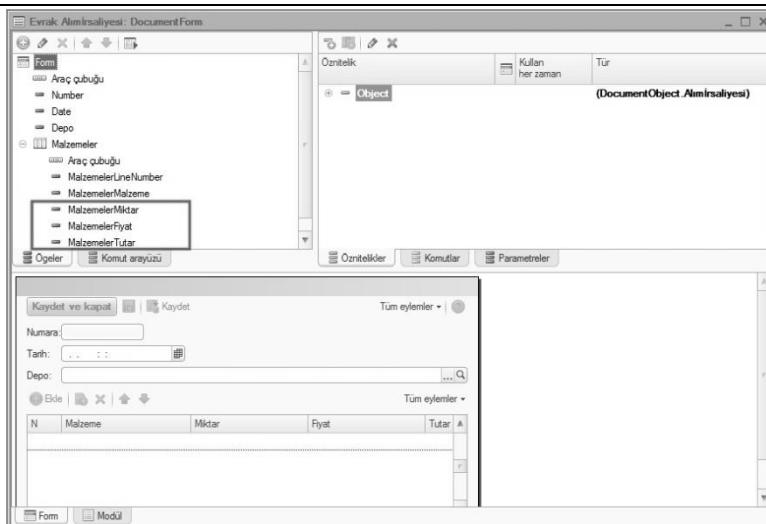
Resim 4.18. Form yapılandırıcı

Form tipi olarak **Evrak formu** seçelim ve sistemin sunduğu standart özellikleri kabul etmek için **Tamam** butona tıklayalım.

Konfigürasyon ağacında, *Alımİrsaliyesi* evrakı konfigürasyon nesnesi için *DocumentForm* formu eklenmiştir (Resim 4.19), ekranda ise oluşturulan formu içeren form tasarım penceresi gelmiştir (Resim 4.20).



Resim 4.19. Tasarımcıda evrakin yeni formu



Resim 4.20. Form tasarım penceresi

Form tasarımcısı, birbiriyile ilişkili olan birkaç tasarım pencerelerini birleştirir. Form tasarımcısı ile çalışma detaylı olarak burada anlatmayacağız, sadece şu anda geliştirilen fonksiyon ile ilgili kısımları kullanacağız.

Şu anda alt kısmında bulunan formun görsel şeklini ve form tasarımcısının üst solda bulunan form öğeleri düzeltme kısmını inceleyeceğiz.

İlk önce anlamak gereken şey şudur; geliştirici istege göre formu “çizemez”. Geliştirici sadece formda bulunması gereken öğeleri belirler, sistem ise, otomatik olarak bu öğeleri formda yerleştirir.

Form tasarımcısının üst sol tarafında bulunan form öğeleri hiyerarşik şeklinde gösterilir. Formda öğeleri genel yerleştirme mantığı şudur: hiyerarşide öge ne kadar çok yüksek seviyede bulunuyorsa, formda o kadar çok üstte ve solda yer almaktadır (yani hiyerarşide en üstte bulunan öğeler, formda üst ve sol tarafta yer alır, hiyerarşide daha alta bulunan öğeler, üst öğelere göre daha alt ve sağda konumlandırılır).

Bu hiyerarşik yapı *Öğeler* sekmesinde düzeltilebilir ve formda veri yansıtılmasını ve düzeltmesini ayarlar.

Konfigürasyonda tanımlanmış olduğu *Alımİrsaliyesi* evrakına göre sistem otomatik olarak, form şeklini belirleyen öge yapısını oluşturmuştu.

Bu öğeler farklı amaç ve davranışlara sahiptir. Fakat tüm öğeler veritabanında bulunan verinin görsel şeklinde yansıtılmasını ve bu veri ile çalışmasını sağlamak için geliştirilmiştir.

Öge ağacında taşıyarak öğe yerleri değiştirebilirsınız (mesela tablo bölümündeki öznitelikler), aşağıdaki ekranda yapılan değişikliklere göre formun görsel şekli de değişir. Bunu yaparken geliştiricinin, formda ögenin

yerleştirmesini, boyutunu, bağlamasını vs. pixellere kadar düşünmesine gerek yok, platform bunu otomatik olarak ayarlar.

Geliştirici özellik panosu araçları ile ögenin özelliğini değiştirek formda yansıtma şeklini değiştirebilir. Veya form öğe yapısını değiştirebilir – yeni öğe, alan grubu veya yeni tablo bölümü ekleyip form verileri ile bağlayabilir.

Fakat şu anda buna ihtiyacımız yok. Biz tablo bölümünde bulunan üç öğe ile çalışacağız; *MalzemelerMiktar*, *MalzemelerFiyat* ve *MalzemelerTutar* (Resim4.20).

Miktar veya *Fiyat* alanında değer değiştirildiğinde otomatik olarak *Tutar* alanı değeri *Miktar*Fiyat* olarak hesaplanı istiyoruz.

Bunu gerçekleştirmek için kaynak kodu ile *Tutar = Miktar * Fiyat* komuta benzer bir komut yazmamız gereklidir. Ve bu komut *Miktar* veya *Fiyat* alanı değiştirildiğinde çalıştırılacaktır. Fakat bu değer değiştirme anı nasıl “yakalayacağız”?

Olay işleyicisi

Gördüğümüz gibi, sistem, konfigürasyon ağacında tanımlanmış nesneler ile çalışmayı “biliyor”; nesne verileri göstermek, yeni öğe eklemek vs. Yani, sistemin, her şeyin nasıl çalışmak gerektiği ile ilgili “standart bilgisi” vardır.

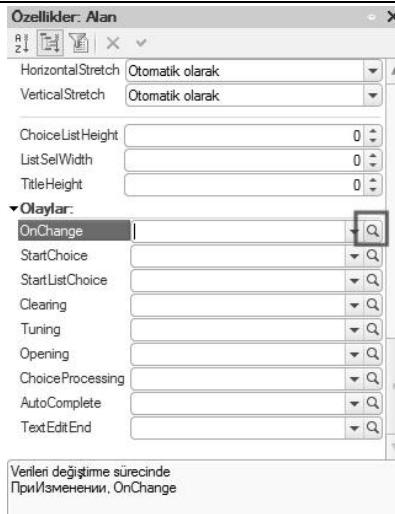
Fakat genelde geliştirici için bu “standart bilgiler” sadece çok basit durumlarda yeterlidir. Gerçek görevler daha çok kapsamlıdır. Bu yüzden sistemde, standart işleyişin farklı durumları ile bağlı olan *olaylar* vardır. Bahsedilen farklı durumlara, form işleyışı ve formdaki öğelerin işleyışı durumları da dahildir.

Kaynak kodu kullanarak, geliştirici bu standart işleyişlere müdahale edip, ilgili olay gerçekleştirliğinde, kendi algoritmaları tanımlama olanağına sahiptir. İşte şimdi yapacağımız şey budur.

MalzemelerMiktar form öğesi üzerinde çift tıklayarak veya sağ tuş ile çağırılan menüden bağlam menüsünden *Özellikler* maddesini seçerek *Özellik panosunu* açalım.

Özellik panosunun en alt kısmında bu alan ile ilgili var olabilecek olaylar listesine ulaşırız.

Belli ki, şu anda ihtiyacımız olan olay *OnChange* (Değiştirildiğinde) olaydır. Bu olay alanda veri değişikliğinden sonra çalıştırılır. Olay listesinde onu bulup büyütüp resmi olan açma butona basalım (Resim 4.21).



Resim 4.21. “Miktar” tablo bölüm alanının
“OnChange” olay işleyicisini oluşturma

Sistem, formun modülünde *işleme* prosedürünün şablonu oluşturup **Modül** sekmesini açar (Resim 4.22)

```

&AtClient
Procedure MalzemelerMiktarOnChange(Item)
    // İşleyici içeriğini ekle.
EndProcedure

```

Resim 4.22. “Miktar” tablo bölüm alanının
“OnChange” olay işleyicinin şablonu

Modül – kaynak kodundaki program metni için bir “depodur”. Bu form modülüdür, çünkü form denetimler ile ilgili tüm olay işleyicileri form modülünde yerleştirilmektedir.

Form modülündeki, **MalzemelerMiktarOnChange()** prosedürüne aşağıdaki metin ekleyelim (list 4.1).

List 4.1 “MalzemelerMiktarOnChange()” prosedürü

```
TabloBölümSatırı = Items.Malzemeler.CurrentData;
TabloBölümSatırı.Tutar      =      TabloBölümSatırı.Miktar      *
TabloBölümSatırı.Fiyat;
```

Bu satırların anlamı açıklayalım.

Birinci satırda ilk önce bir değişken oluşturuyoruz: **TabloBölümSatırı**. Bu değişken için, hesaplanmak gereken satırın bilgilerini içeren nesne atanacaktır.

Kaynak kodunun özelliklerinden biri, oluşturulan değişkeni ve değişkenin veri türü önceden tanımlamadan, değişken doğrudan oluşturulabilir. Değişkeni direkt ekliyoruz, değişkenin veri türü ise içerdiği değer türü tarafından belirlenir.

Form modülünde bulunduğuumuz için, **ManagedForm** kaynak kod nesnesinin tüm özellikler ve metotları doğrudan ulaşılabilir. Bizim örneğimizde, eşit simgesinden sonra formun denetim koleksiyonuna başvuruyoruz. Bunu yapmak için *ManagedForm* kaynak kod nesnesinin **Items** özelliğini kullanacağız.

Form denetim koleksiyonu, formun tüm denetimlerini içeren **FormAllItems** kaynak kod nesnesidir. Yani form denetimlerini hiyerarşik şeklinde ifade ettiğimizde, form denetim koleksiyonu bu ağaçın programsal köküdür.

Formun her denetimi bu nesnenin özelliği olarak belirlenebilir. Yani belirlemek için nesne sonunda noktayı yazmak gereklidir. Bizim örneğimizde **Malzemeler** tablo bölümüne başvuruyoruz (*Items.Malzemeler*).

Evrakin tablo bölümü bir **FormTable** kaynak kod nesnesidir. Şu anda düzeltmekte olan (aktif olan) satırı almak için *FormTable* kaynak kod nesnesinin **CurrentData** özelliğini kullanmak gereklidir (*Items.Malzemeler.CurrentData*)

Böylece işleyici prosedürünün ilk satırı işleme sonucunda, *TabloBölümSatırı* değişkeni **FormDataStructure** nesnesin içerecek. Bu nesne evrak tablo bölümünün geçerli (aktif) satırında bulunan bilgileri içerir (*Items.Malzemeler.CurrentData*).

Bu nesneyi almaktan sonra, sütun adını nesnenin özelliği olarak belirlendiğinde, belli bir sütunun verilerine ulaşabiliriz. Örneğin, *TabloBölümSatırı.Miktar* ifadesini kullanarak, aktif satırındaki *Miktar* sütun değerine ulaşabiliriz.

Böylece, işleyici prosedürünün ikinci satırında aktif satırındaki *Tutar* sütun değeri *Miktar* ve *Fiyat* değerleri çarparak hesaplanmaktadır.

1C: İşletme ortamında

Şimdi bu nasıl çalıştığını bakalım. Sistemi 1C:İşletme ortamında çalıştırılarım ve önceden oluşturmuş olduğumuz evraklardan bir tanesini açalım. Evrakına tablo bölümünde *Miktar* değiştirildiğine tutarın otomatik olarak hesaplandığını göreceğiz.

Birden fazla olayın işlenmesi için tek prosedür

Gördüğümüz gibi evrakta *Miktar* alanında değer değiştirildiğinde *Tutar* değeri otomatik olarak hesaplanmaktadır.

Çok güzel. Fakat şimdi aynı fonksiyon *Fiyat* alanı için uygulamak gerekir. Biraz ileri de baktığımızda, böyle bir fonksiyon diğer evraklarda da gerekli olabilir.

Bu yüzden, benzer tablo bölüm alanlarına sahip olan evrakların bu algoritmayı kullanabilmeleri için, tutar hesaplamasını “herkesin ulaşabildiği” yere konumlandırmak gereklidir.

“Herkesin ulaşabildiği” yerleri tanımlamak için **Genel modül** konfigürasyon nesnesi kullanılır. *Genel modüller* konfigürasyon ağacının *Genel – Genel modüller* grubunda bulunur. Bu modüllerde bulunan prosedürler ve fonksiyonlara diğer konfigürasyon nesnelerinden ulaşılabilir.

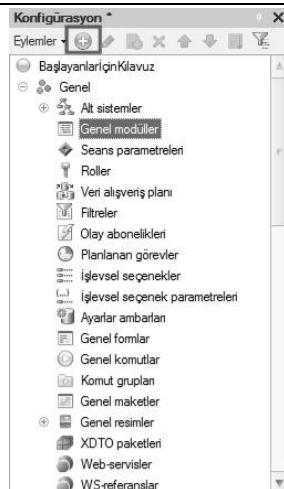
Dolayısıyla bir genel modül ekleyip bizim tutar hesaplama prosedürünü oraya taşıyalım. Evrakta da, genel modülde bu prosedürü çağırmasını sağlayalım.

Tasarımcı ortamında

Genel modül

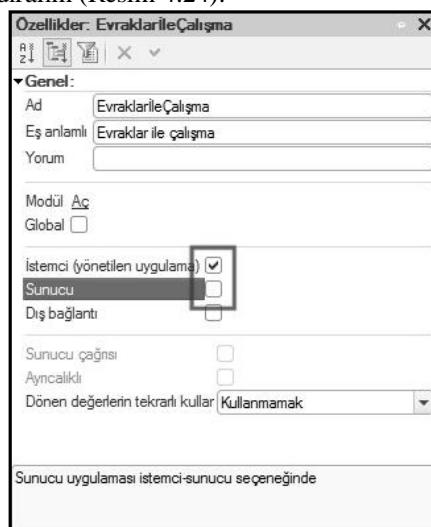
Genel modül konfigürasyon nesnesini ekleyelim.

Bunu yapmak için konfigürasyon ağacında *Genel grubu* açıp *Genel modüller* üzerinde imleç yerleştirerek konfigürasyon nesne ağacının araç çubuğundan **Ekle** butonuna tıklayalım (Resim 4.23).



Resim 4.23. Konfigürasyon nesne ağacında Genel modül eklem

Oluşturulan genel modülüli **Evraklar ile Çalışma** olarak adlandıralım, özelliklerinde **İstemci (yönetilen uygulama)** özelliğini işaretleyelim ve **Sunucu** işaretini kaldırıralım (Resim 4.24).



Resim 4.24. Genel modül özellikleri

Modüle aşağıdaki metin ekleyelim (liste 4.2).

Liste 4.2. “TutarHesaplama()” prosedürü

```
Procedure TutarHesaplama(TableName) Export
```

```
    TabloBölümSatırı.Tutar = TabloBölümSatırı.Miktar  
        *
```

```
    TabloBölümSatırı.Fiyat;
```

```
EndProcedure
```

Kodu açıklayalım. **TutarHesaplama()** prosedürüne, *Miktar* alanın *OnChange* olay işleyicisinde tanımladığımı **TabloBölümSatırı** değişkeni aktarıyoruz. Bu değişken AlımIrsaliyesinin düzeltmekte olan tablo bölüm satır bilgilerini içerir.

Şimdi bu değişkeni kullanarak, tablo bölüm sütunlarına ulaşıp tutarı hesaplayabiliriz (Tutar = Miktar * Fiyat).

Prosedürün başlığında bulunan **Export** anahtar kelimesi, bu prosedüre diğer modüllerden ulaşabileceğini tanımlar.

Şimdi formun modülünde işleyici metni değiştirelim (liste 4.3).

Liste 4.3. “MalzemelerMiktarOnChange()” prosedürü

&AtClient

Procedure MalzemelerMiktarOnChange(Item)

```
    TabloBölümSatırı = Items.Malzemeler.CurrentData;  
    EvraklarİleÇalışma.TutarHesaplama(TabloBölümSatırı);
```

EndProcedure

Gördüğümüz gibi prosedürün ilk satırı değişmemiş. İkinci satırda ise *EvraklarİleÇalışma* genel modülünden *TutarHesaplama()* prosedürü çağrııp içine parametre olarak tablo bölümünün aktif satırını aktarıyoruz.

Şimdi Fiyat alanı için aynı işleyici belirtmek gereklidir. Form modülünde gerekli prosedürü yazdığımız için, bu prosedürü diğer form denetiminin olay için atayabilirdik. Fakat “1TC” şirketinin yazılım geliştirme standartları böyle çözüme izin vermez.

Bu yüzden, *MalzemelerMiktar* alanı yaptığımız gibi, *MalzemelerFiyat* form ögesi için OnChange olay işleyicisini oluşturup genel modülden tutar hesaplama prosedürünü çağıracağız (Liste 4.4).

Liste 4.4. “MalzemelerFiyatOnChange()” prosedürü

&AtClient

Procedure MalzemelerFiyatOnChange(Item)

```
    TabloBölümSatırı = Items.Malzemeler.CurrentData;  
    EvraklarİleÇalışma.TutarHesaplama(TabloBölümSatırı);
```

EndProcedure

1C: İşletme Ortamında

1C:İşletme sistemi hata ayıklama biçiminde çalışıralım ve *Alımİrsaliyesi* evrakinin tablo bölümündeki tutar hesaplama işlemi *Miktar* veya *Fiyat* alanı değiştirildiğinde doğru çalıştığından emin olalım.

“ Hizmet faturası” evrakı

Şimdi aynı şekilde ihtiyaç duyduğumuz diğer evrakı oluşturalım – **Hizmet faturası**. Bunu yapmak için Alım irsaliyesi evrakını tanımlarken yaptığımız ile aynı şey yapmak gereklidir.

Tasarımcı ortamında

Yeni evrak konfigürasyon nesnesini ekleyelim ve **HizmetFaturası** olarak adlandıralım.

Genel sekmesinde nesnenin 1C:İşletme ortamındaki arayüzünde nasıl gösterileceğini belirleyelim.

Nesne tanımını belirlemeyeceğiz, onun yerine nesne eş anlamlısı kullanılacak.

Liste tanımını Hizmet faturaları olarak belirleyelim.

Alt sistemler sekmesinde evrakin *Hizmetler* ve *Muhasebe* alt sistemlere ait olduğunu belirleyelim.

Veriler sekmesinde evrakin özniteliklerini oluşturalım;

- **Depo**, tür **CatalogRef.Depolar**. *Doldurma değer* özelliğinde *Depolar* kart listesinde önceki tanımlı olan *Genel depo*'yu tanımlayalım.
- **Müşteri**, tür **CatalogRef.Müşteriler**. *Doldurma denetimi* özelliğini **Hatayı görüntülemek** olarak belirleyelim.
- **Usta**, tür **CatalogRef.Personeller**. *Doldurma denetimi* özelliğini **Hatayı görüntülemek** olarak belirleyelim.

Bu evrakin **MalzemeHizmetListesi** tablo bölümünü oluşturalım ve tablo bölüme öznitelikleri ekleyelim;

- **Malzeme**, tür *CatalogRef.MalzemeHizmet*,
- **Miktar**, tür *Number*, Uzunluk 15, Ondalık 2, Negatif olamaz,
- **Fiyat**, tür *Number*, Uzunluk 15, Ondalık 2, Negatif olamaz,
- **Tutar**, tür *Number*, Uzunluk 15, Ondalık 2, Negatif olamaz.

Tablo bölümü ve tüm tablo bölüm öznitelikleri için *Doldurma denetimi* özelliğini **Hatayı görüntülemek** olarak belirleyelim.

Formlar sekmesinde evrakin genel formu oluşturalım.

MalzemeHizmetListesiMiktar alanı için *EvraklarİleÇalışma* genel modülünden tutar hesaplama prosedürünü çağırın, *OnChange* olay işleyicisini oluşturalım.

Olay işleyicisi oluşturulduğunda, *MalzemeHizmetListesiMiktarOnChange* olay işlemci şablonunu içeren form modülü açılacak. Herhangi bir değişiklik yapmadan formun *Form* sekmesine geri gelip *MalzemeHizmetListesiFiyat* alanı için *MalzemeHizmetListesiFiyatOnChange* olay işleyicisini oluşturalım.

Bunu yaptıktan sonra form modülünü aşağıdaki listede gösterildiği gibi dolduralım (Liste 4.5).

Liste 4.5. “HizmetFatura” evrakin form modülü

&AtClient

Procedure MalzemeHizmetListesiMiktarOnChange(Item)

TabloBölümSatırı = Items.MalzemeHizmetListesi.CurrentData;
EvraklarİleÇalışma.TutarHesaplama(TabloBölümSatırı);

EndProcedure

&AtClient

Procedure MalzemeHizmetListesiFiyatOnChange(Item)

TabloBölümSatırı = Items.MalzemeHizmetListesi.CurrentData;
EvraklarİleÇalışma.TutarHesaplama(TabloBölümSatırı);

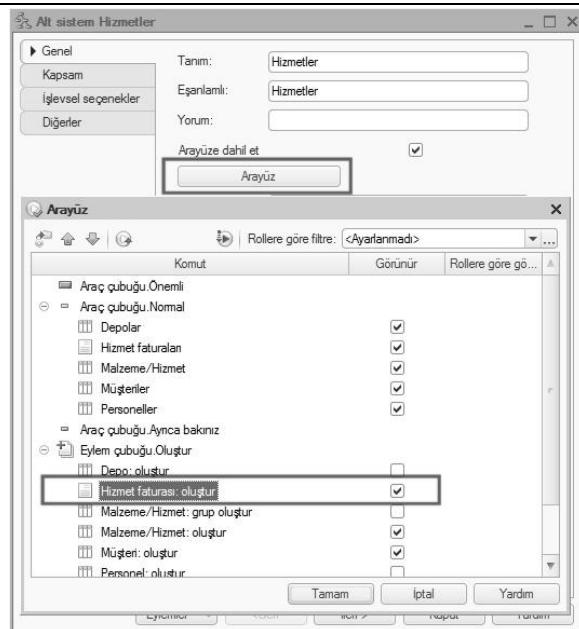
EndProcedure

Son olarak uygulama arayüzü düzeltelim. *Hizmetler* bölümünün eylem çubuğu yeni evrak oluşturma komutu gösterilmesini istiyoruz.

Farklılık olsun diye, bu kez biraz farklı bir yol izleyeceğiz. *Hizmetler* alt sisteminin düzeltme penceresini açıp **Arayüz** butonuna tıklayalım.

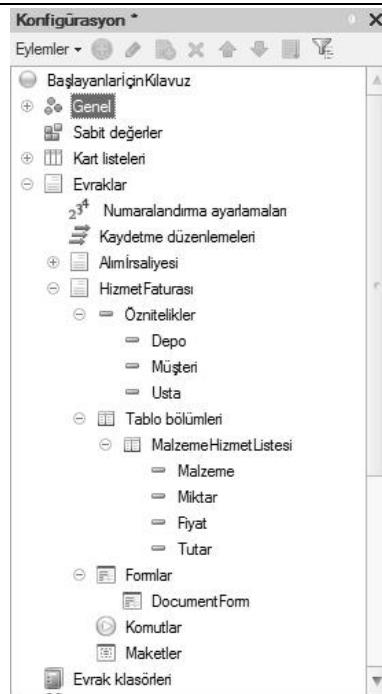
Açılan pencerede bu alt sisteme ait tüm komutlar gösterilecektir.

Eylem çubuğu.Oluştur grubunda *Hizmet faturası: oluştur* işaretleyelim (Resim 4.25).



Resim 4.25. Alt sistemini arayüz ayarları

Yapmış olduğumuz eylemlerin sonunda konfigürasyon nesne ağacında *HizmetFatura* aşağıdaki gibi görünmelidir (Resim 4.26).



Resim 4.26. Konfigürasyon nesne ağacında “HizmetFurası”

1C:İşletme Ortamında

1C:İşletme sistemini hata ayıklama biçiminde çalışıralım.

Hizmetler bölümünün eylem çubuğu yeni Hizmet fatura evrakını oluşturma komutunu uygulayalım ve resimde gösterildiği gibi dolduralım (Resim 4.27).

Hizmet faturası (oluşturma) *

Kaydet ve kapat | Kaydet | Tüm eylemler | ?

Numara:

Tarih: 28.02.2011 00:00:00 |

Depo: Genel depo | ... |

Müşteri: Ahmet Bey | ... |

Usta: Mehmet Uzman | ... |

+ Ekle | | | | | Tüm eylemler

N	Malzeme	Miktar	Fiyat	Tutar
1	Transistor Philips 2N2369	1	3,00	3,00

Resim 4.27. "Hizmet fatura No 1" evrakı oluşturma

Genel depo otomatik olarak geldiğini ve Usta ve Müşteri alanları için veri doldurma denetimi uyguladığını görüyoruz.

Tablo bölümünde de *Miktar* ve *Fiyat* alanlarına veri girildiğinde oluşturduğumuz algoritmaya göre *Tutar*'ın otomatik olarak hesaplanması fark edeceğiz.

Sorular

- ✓ *Evrak özellikler nelerdir?*
- ✓ *Evrak öznitelikler ve tablo bölümleri ne için kullanılır?*
- ✓ *Evrakin genel formları nelerdir?*
- ✓ *Evrak kaydetme nedir?*
- ✓ *Yeni Evrak konfigürasyon nesnesi nasıl oluşturulur ve özellikleri nasıl tanımlanır?*
- ✓ *Yeni evrak ögesi nasıl eklenir ve evraka veri nasıl doldurulur?*
- ✓ *Evrakin kendi formu nasıl oluşturulabilir?*
- ✓ *Form yapılandırıcısı nedir?*
- ✓ *Form tasarımcı nedir?*
- ✓ *Form denetimi nedir?*
- ✓ *Olaylar nedir ve ne ile bağlıdır?*

DERS 5

Teori

Süre

Dersin tahmin süresi – 2 saat

Genel form mekanizması?	130
Olay işleyicileri?	131
Modüler	131
Modül türleri	131
Form modülü bağılamı	136
Formlar – program nesneleri	145
Form modülünde olay işleyicileri – prosedürleri	146
Form modülünde yazılmış olanları nasıl anlamak gereklidir	148
Kaynak kodunda belli bir kodun çalışmasını nasıl anlamak gereklidir?	149
Sözdizimi-yardımcısı ile kodları inceleme	150
Birinci yöntem	150
İkinci yöntem	158
Hata ayıklama biçimleri ile kod analizi	161
Nesneler	167
Sunucu ve istemciler	168
Genel modüllerini derleme	171
Derleme direktifleri	172
Kaynak kodunu sunucuda veya istemcide işlemek	172

Diger derslerde belirtirmesi gereken teknik detayları bu bölüme yerleştirmemizin asıl amacı, okuyucularımızın diğer konuları daha iyi kavramaları ve teknik detaylara ihtiyaç duyduklarında kolaylıkla bu bölüme ulaşabilmeleridir. Diğer bölümlerde bahsini etmeyi ertelediğimiz detayları bu bölümde yerleştirdik.

Bu teorik bilgiler 1C:İşletme sisteminde daha iyi, kaliteli ve doğru geliştirmek için ve daha önce yapmış olduğumuz hakkında daha detaylı bilgi edinmek için yazılmıştır.

Ama eğer şuanda bu bilgileri kabul edecek durumda değilseniz, bu bölümü geçebilirsiniz ve daha sonra geri dönebilirsiniz. Bu durum gerçekleştirmekte olduğumuz örnek konfigürasyonu geliştirmek için zararlı değildir.

Genel form mekanizması?

Daha önceki bölümde biz Alım irsaliyesi için evrak formu oluşturmuştu ve onu genel form olarak atamıştık.

Bütün konfigürasyon nesnelerinde birkaç türlü formlar mevcuttur. Bu formların ilgili nesnelerin verilerini farklı biçimlerde görüntülemek için tasarlanmıştır.

Eğer geliştirici genel form olarak kendi formlarını tanımlamazsa, ilgili formlara başvurulduğunda sistem otomatik olarak o formu oluşturacaktır. Bu durumda geliştiricinin ekstra fonksiyon içermeyen formları oluşturmak ve tasarlamak için zaman kaybetmesine gerek yoktur.

Burada formun ne biçimde çağrılması önem ifade etmemektedir. İster etkileşimli, ister program üzerinden çağrıma olsun hiç fark etmez, ilgili form sistem tarafından çağrı sürecinde dinamik olarak oluşturulacaktır. Mesela, Müşteriler kart listesinin liste formu otomatik olarak hem Eylemler > Kart listeleri > Müşteriler komutu ile hem de programda yazılmış olan global metotla (“GetForm()”) açılacaktır (list 5.1).

List 5.1 “GetForm()” metodu çağrısı

```
ListeFormu = GetForm("Catalog.CariHesaplar.ListForm");
```

Burada belirtmek gereken küçük bir özellik daha vardır. Bazı konfigürasyon nesnelerinin genel form listesi ile yapılandırıcı yardımı ile oluşturulabilen formların listesi arasında fark vardır.

Örneğin, bazı kayıt tablolarında genel form olarak sadece liste formu atanabilir, ama yapılandırıcı yardımı ile oluşturulabilen formların listesine baktığınızda kayıt formunu da göreceksiniz ve aynı kayıt formu ise genel formların listesinde mevcut değildir.

Bu form listelerinin farklı olmasının sebebi şu şekildedir, genel formlar kısmında sadece kullanıcı açısından bakıldığından kullanılacak olan formların listesi. Yapılandırıcı listesinde ise daha çok konfigürasyon geliştiricisinin kullanacağı formların listesi gelmektedir.

Olay işleyicileri?

Daha önceki konularda formun bazı denetimleri için OnChange olayını oluşturmuştu. Bu nedir?

1C:İşletme 8 sisteminde olay işleyicilerini iki türlü tanımlamak mümkündür: form ve form denetimleri ile ilgili olaylar ve diğerleri.

Bu olayların arasındaki tek fark şudur: form veya form denetimleri için geçerli olan olaylar atanabilir olaylardır. Diğerleri diye nitelendirdiğimiz olaylar ise, sabit olaylardır. Bu sabit olayların ayrıcalığı ilgili prosedür aynı ad altında olması gereklidir.

Atanabilir olaylarda ise belli form denetiminin ilgili özelliğine ilgili prosedür ismini yerleştirmek mümkündür.

İşleyicileri atama işlemi tasarımcıda etkileşimli olarak formla çalışma sürecinde gerçekleştirilebildiği gibi, program tarafından form veya form denetimlerinin metotlarını kullanarak – SetAction().

Modüller

Daha önceki konularda formun modül kısmına geçerek yazılmış olan kodların burada saklandığını belirtmiştık. Modül – programın kaynak kodu şeklinde metinlerinin saklandığı bir nevi ambardır.

Modül türleri

Konfigürasyonda farklı türden modüller mevcuttur. Onlar belli bir konfigürasyon nesnelerine (örneğin, formlara) ait olabilirler veya serbest bir şekilde konfigürasyonda mevcut olabilirler (konfigürasyonun tamamına ait olabilirler).

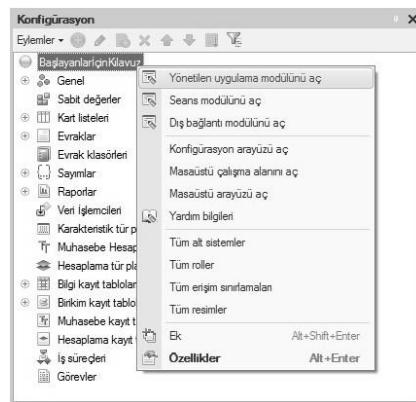
Modüllerde tanımlanan kaynak kodu metinleri 1C:İşletme sistemi tarafından önceden tanımlanmış çalışma anları - olaylar tarafından kullanılırlar.

1C:İşletme 8 sisteminde aşağıdaki modül türleri mevcuttur.

Yönetilen uygulama modülü. Yönetilen uygulama modülü 1C:İşletme sistemi ince istemcide veya web-istemci şeklinde çalıştırıldığında durumlarda kullanılacaktır.

Bu modülde değişkenleri tanımlamak ve konfigürasyonun herhangi ulaşılabilir modülünde mevcut prosedür ve fonksiyonlara (dış bağlantı modülü hariç) başvuru yapmak mümkündür. Bu modülden global olmayan ve İstemci (yönetilen uygulama) özellikli modülün prosedür ve fonksiyonlarını çağırma mümkün. Yönetilen uygulama modülünün bağlamından genel modüllerin export özellikli tüm fonksiyon ve prosedürleri ulaşılabilirdir.

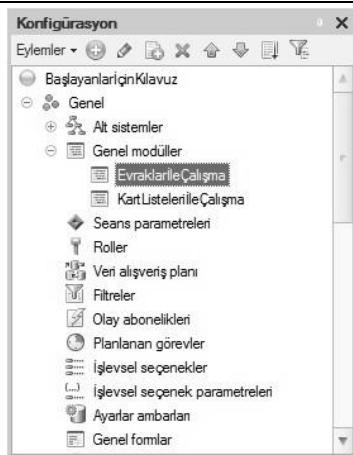
Yönetilen uygulama modülünü açmak için konfigürasyon ağacının kökünde (**Başlayanlar İçin Kılavuz** satırı üzerinde) fare üzerinde sağ tuşu ve bağlam menüsünde **Yönetilen uygulama modülünü aç** komutu kullanılabilir (Resim 5.1).



Resim 5.1. Yönetilen uygulama modülünü açma biçimini

Genel modüller. Genel modüllerde sistemin diğer modüllerinden çağrılabilecek olan prosedür ve fonksiyonlar saklanırlar. Bu fonksiyon ve prosedürler kendi kendilerine uygulanmaz. Bu fonksiyon ve prosedürleri uygulamak için diğer modüllerden çağrılmak gereklidir.

Genel modülleri açmak için konfigürasyon ağacında Genel dalını açmak gereklidir, sonra ise **Genel modüller** dalını açmak ve ilgili satır üzerine çift tıklamak yeterlidir (Resim 5.2).



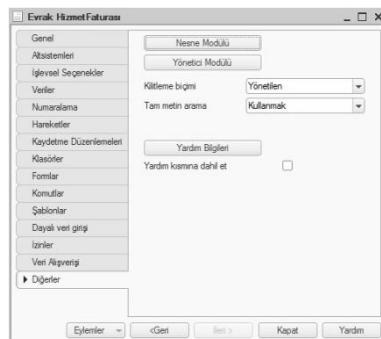
Resim 5.2. Genel modülü açma biçimi

Nesne modülleri. Nesne modülleri – bu örneğin, kart liste nesne modülleri veya evrak modülleri.

Program tarafından evrak ("CreateDocument()" metodu) ve kart listelerin ("CreateItem()" metodu) kaynak kodu metodları ile yeni nesne oluşturulduğunda veya kullanıcılar tarafında ilgili nesne eklendiğinde bu modüller çalışmaya başlayacaktır.

Nesnelerin verileri kaydedildiğinde veritabanında nesne modülünde yerleşen farklı olaylar gerçekleşmektedir.

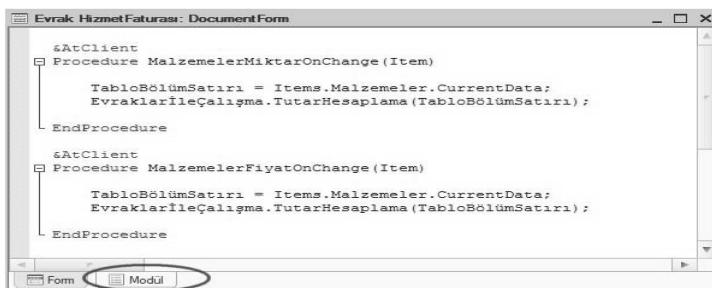
Nesne modülünü açmak için ilgili nesnenin düzeltme penceresinde **Diğerler** sekmesinde **Nesne Modülü** düğmesine tıklamak gereklidir. Veya konfigürasyon ağacında ilgili nesne üzerinden bağlam menüsünü çağırarak **Nesne modülünü aç** kısmına tıklamak yeterlidir.



Resim 5.3. Nesne modülünü açma biçimi

Form modülü. Konfigürasyonda tanımlanan her bir formun kendine ait modülü vardır. Kaynak kodunun ManagedForm nesnesi oluşturulduğunda bu form modülü çalışmaya başlar. Bu nesne, kullanıcı ortamında belli bir nesnenin formunu açtığımızda veya program (kaynak kodunda "GetForm()" veya "OpenForm()") metodları kullanıldığında 1C:İşletme sistemi tarafından otomatik olarak oluşturulur.

Form modülünü açmak için ilgili **Form** konfigürasyon nesnesini açmak gereklidir ve form editörü penceresinde **Modül** sekmesine geçmek gerekir

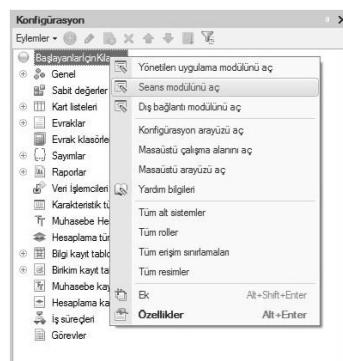


(Resim 5.4).

Resim 5.4. Form modülünü açma biçimi

Seans modülü. 1C:İşletme sistemi çalıştırıldığında konfigürasyonun yükleme sürecinde uygulanacak kaynak kodları bu modülde belirtilir. Seans modülü seans parametrelerini başlatmak için ve seans ile ilgili eylemleri gerçekleştirmek için tasarlanmıştır. Seans modülü export özellikli prosedür ve fonksiyonları içermemektedir ve genel module ait fonksiyon ve prosedürleri kullanabilmektedir.

Bu modülü çağrılmak için konfigürasyon ağacının kökünde ("Başlayanlar İçin Kılavuz" satırında) bağlam menüsünü çağırarak **Seans modülünü aç** komutunu uygulamak gereklidir (Resim 5.5).



Resim 5.5. Seans modülünü açma biçimi

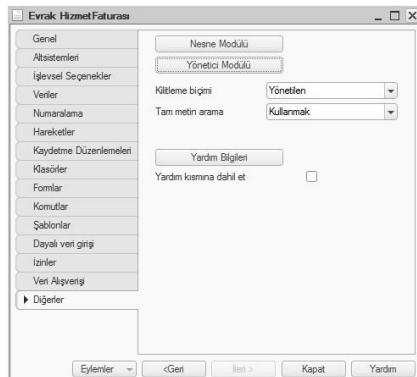
Dış bağlantı modülü. Dış bağlantı modülü dış bağlantı oturumundan çağrılabilen fonksiyonları ve prosedürleri içerir.

Dış bağlantı modülünü açmak için konfigürasyon ağacının kökünde ("Başlayanlar İçin Kılavuz" satırında) bağlam menüsünü çağırarak **Dış bağlantı modülünü aç** komutunu kullanmak gereklidir (Resim 5.5).

Yönetici modülü. Her bir uygulama nesnesinde onu konfigürasyon nesnesiymiş gibi yönetecek yönetici bulunmaktadır. Bu yöneticinin yardımcı ile yeni nesneler oluşturmak, formlar ve maketlerle çalışmak mümkündür. Yönetici modülü, sistem tarafından belirlenen yönetici imkanlarını daha da genişletmek için tasarlanmıştır (tabii ki yazılmış olan kaynak kodlarının yardımcı ile).

Bu ise, belli bir konfigürasyon nesnesi örneği için değil de ilgili konfigürasyon nesnesi için kendine ait metotları tanımlamada faydalı olacaktır (örneğin, kart listesi).

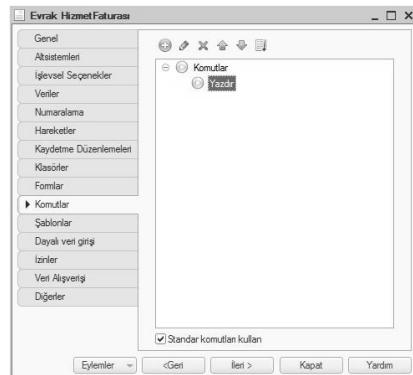
Yönetici modülü, ilgili nesnenin düzeltme penceresinde **Digerler** sekmesinde **Yönetici Modülü** komutu ile (Resim 5.6) veya ilgili nesne üzerinden bağlam menüsü çağrılarak **Yönetici modülünü aç** komutu uygulanarak açılabilir.



Resim 5.6. Yönetici modülünü açma biçimi

Komutlar modülü. Konfigürasyonun kendisinde olduğu gibi, bazı konfigürasyon nesnelerinde de bağlı (alt) konfigürasyon nesneleri mevcuttur – Komutlar. Her bir komutta komut modülü mevcuttur. Bu modülde bu komutun uygulanması için önceden tanımlanmış prosedürü ("CommandProcessing()") tanımlamak mümkündür.

Belli bir konfigürasyon nesnesine ait olan komutu açmak için ilgili nesnenin düzeltme penceresinde Komutlar sekmesinde ilgili komut üzerine çift tıklamak gereklidir (Resim 5.7). Bundan başka konfigürasyon ağacında ilgili komut üzerinden bağlam menüsünü çağırarak Komut modülünü aç komutunu kullanmak yeterlidir.



Resim 5.7. Konfigürasyon nesnesinin komut modülünü açma biçimi

Form modülü bağılamlı

Her bir modül konfigürasyonun diğer kısımları ile bağlıdır. Bu bağlantıya modül bağılamlı denir.

Bu modül bağılamlı, modülün uygulanma sürecinde ulaşılabilcek değişkenleri, nesneleri, prosedürleri ve fonksiyonları tanımlar.

Formda göstermiş olduğumuz olay işleyicisine geri dönemlim ve daha detaylı inceleyelim ve formun modül bağılamlı neyden ibaret olduğunu gözlemleyelim.

Form modülü bağılamlı içerir:

- form modülünün kendi bağılamlı;
- form öznitelikler;
- ManagedForm nesnesi için kaynak kodunda tanımlanan özellik ve metodlar;
- ana öznitelığın türüne göre tanımlanan form uzantisının özellik ve metodları;

- global bağlam, global olmayan genel modüller ve global özellikli genel modüllerin export özellikli fonksiyon ve prosedürleri;
- yönetilen uygulama modülünün export özellikli değişkenleri, fonksiyonları ve prosedürleri;

Şimdi ise bu listelenenleri inceleyelim hep birlikte.

1. Form modülünün kendi bağlamı

Form modülünün kendi bağlamı şu demek – bu modülde belirlenen değişkenler, prosedürler ve fonksiyonlar.

Örneğin, bu modülde tanımlanan **TutarHesapla()** prosedürüne doğrudan başvurmak mümkündür (list 5.2).

List 5.2 Form modülü

```
&AtClient
Procedure Komut1()
    TutarHesapla();
EndProcedure

&AtServerNoContext
Procedure TutarHesapla()
    ...
EndProcedure
```

Veya bu form modülünde tanımlanan değişkene aynı modülde farklı prosedürden çağrılmak mümkündür, örneğin, **SeçimNotu** (list. 5.3).

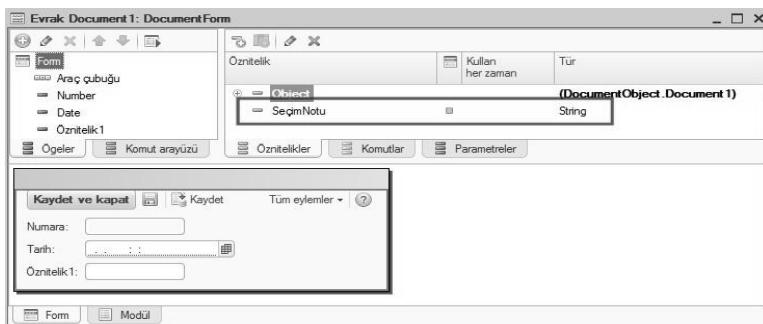
List 5.3 Form modülü

```
&AtClient
Var SeçimNotu

&AtClient
Procedure Komut1()
    SeçimNotu = 3;
EndProcedure
```

2. Form öznitelikleri

Eğer formda **SeçimNotu** özniteligi mevcut ise (Resim 5.8), form modülünde ilgili öznitelige doğrudan olarak başvurmak mümkündür (List. 5.4).



Resim 5.8. Form özniteligi – "SeçimNotu"

List 5.4 Form modülü

```
&AtClient
Procedure Komut1()
    SeçimNotu = 3;
EndProcedure
```

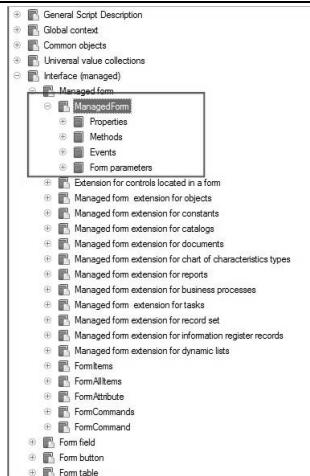
3. ManagedForm nesnesinin özellik ve metodları

Kaynak kodunun ManagedForm nesnesinin özellik metodları sözdizimi-yardımcıda açıklanmıştır: **Interface (managed)** – **Managed form** – **ManagedForm** (Resim 5.9).

Bu özellik ve metodlara doğrudan isimlerini yazarak başvurmak mümkündür. Örneğin, form başlığını tanımlamak mümkündür (List. 5.5).

List 5.5 Form modülü

```
&AtClient
Procedure Komut1()
    Title = "Yeni form başlığı";
EndProcedure
```



Resim 5.9. Sözdizimi-yardımcıda yönetilen formun özellikleri

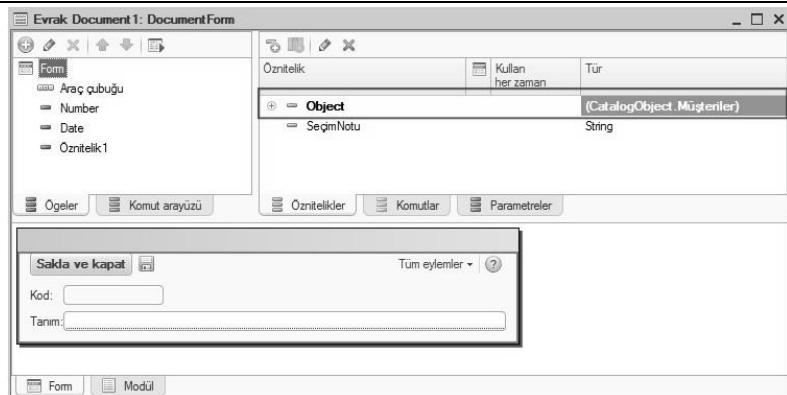
Veya formu kapatmak mümkündür (List. 5.6).

List 5.6 Form modülü

```
&AtClient
Procedure Komut1()
    Close();
EndProcedure
```

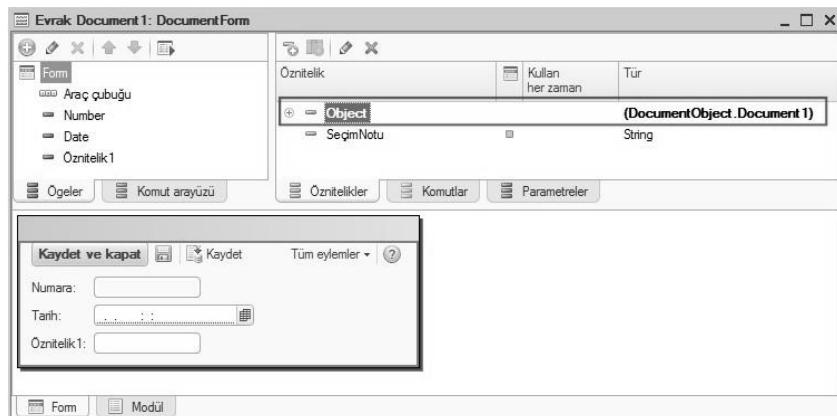
4. Ana öznitelığın türüne göre tanımlanan form uzantısının özellik ve metodları

Form özniteliklerinden birisi ana öznitelik (alan) olarak belirtilebilmektedir ve öznitelikler listesinde kalın harflerle seçilecektir. Genellikle ana alan (öznitelik) formda görüntülenen nesnenin verilerini içerir. Örneğin, eğer form kart listesi formu ise, ana öznitelik **CatalogObject.<adı>** nesnesinin verilerini içerecektir (Resim 5.10).



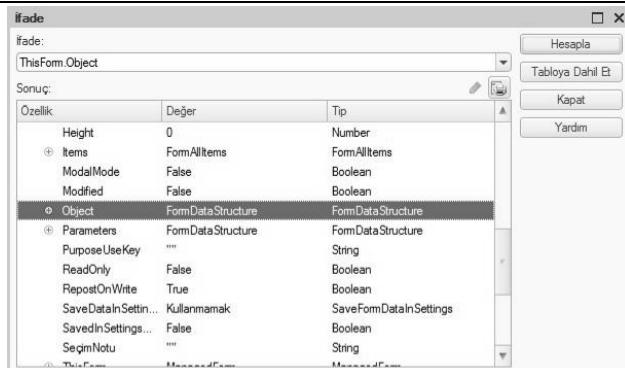
Resim 5.10. Formun ana özniteligi

Eğer bu form evrak formu ise, formun ana öznitelığının türü **DocumentObject.<adı>** olacaktır (Resim 5.11).



Resim 5.11. Formun ana özniteligi

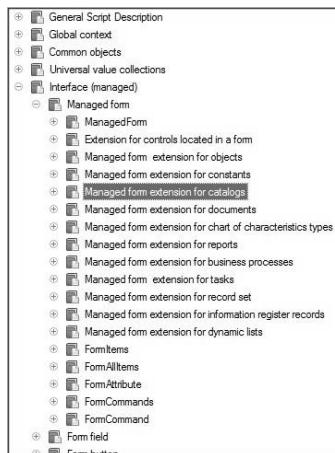
Burada şunu açıklamakta fayda vardır: ana öznitelik (alan) türü parantez içinde - "**(DocumentObject.Document1)**". Aslında bu öznitelik türü gerçek tür değildir. Aslında bunun türü **FormDataStructure** olacaktr (Resim 5.12).



Resim 5.12. Formun ana öznitelliğinin türü

FormDataStructure – farklı türden nesnelerin verilerini içerebilecek universal türdür. Bu yüzden form editöründe daha kolay bir şekilde odaklanmak için Tür sütununda **FormDataStructure** değil de, bu öznitelikte bulunan nesnenin türü görüntülenmektedir. Bu ise gerçek tür olmadığından dolayı parantez içinde yerleştirilmiştir.

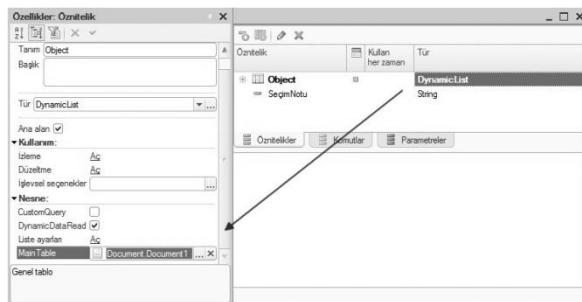
Ana öznitelik türüne bağlı olarak, form modülünde modül bağlamında ulaşılabilir metod ve özellikler listesine ilgili ana öznitelik türünün yönetilen form uzantıları eklenecektir: **Sözdizimi-yardımcı – Interface (managed)** – **Managed form – Managed form extension for catalogs** (Resim 5.13).



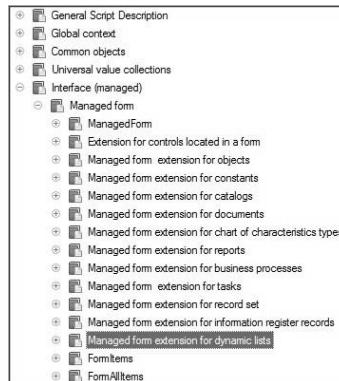
Resim 5.13. Sözdizimi-yardımcıda nesnelerin açıklaması

Eğer ana öznitelik **DynamicList** ise (Resim 5.14), form modülünde modül bağlamında ulaşılabilir metodlar ve özellikler listesine *dinamik liste için form*

metotları ve özellikleri eklenecektir: **Sözdizimi-yardımcısı - Interface (managed) – Managed form – Managed form extension for dynamic lists** (Resim 5.15).

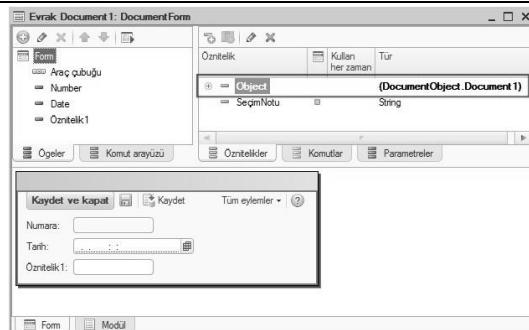


Resim 5.14. Formun ana özniteligi



Resim 5.15. Sözdizimi-yardımcıda nesnelerin açıklaması

Böylece ana öznitelik türünde evrak seçilmişse (Resim 5.16), evrak için yönetilen form uzantılarına başvurmak mümkündür, örneğin, **AutoTime** (List. 5.7).



Resim 5.16. Formun ana özniteliği

List 5.7 Form modülü

```
&AtClient
Procedure Komut1()
    AutoTime = AutoTimeMode. First;
EndProcedure
```

Veya yönetilen form uzantısının **Write()** metodunu yardımcı ile evraki kaydetmek mümkündür (List. 5.8).

List 5.8 Form modülü

```
&AtClient
Procedure Komut1()
    Write(DocumentWriteMode. Posting);
EndProcedure
```

5. Global bağlam, global olmayan genel modüller ve global genel modüllerin export özellikli prosedür ve fonksiyonları

Form modülünde **CurrentDate()** metoduna başvurarak sistem tarihini elde etmek mümkündür (List. 5.9).

List 5.9 Form modülü

```
&AtClient
Procedure Komut1()
    Message(CurentDate());
EndProcedure
```

Veya global bağlamın özelliğine başvurarak kullanıcının çalışma geçmişini elde etmek mümkündür (List. 5.10).

List 5.10 Form modülü

```
&AtClient  
Procedure Komut1()  
    Geçmiş = UserWorkHistory.Get();  
EndProcedure
```

Eğer global özellikli genel modülde (örneğin, **VeriAlışVerisi**) export özellikli **NumaraÖnekiniEldeEtme()** (List. 5.11) prosedürü tanımlandıysa, form modülünde bu prosedüre doğrudan başvurmak mümkündür (List. 5.12).

List 5.11 Global özellikli genel modül

```
Function NumaraÖnekiniEldeEtme() Export  
    Return Constants.DağıtımlıVeritabanıDüğümÖneği.Get();  
EndFunction
```

List 5.12 Form modülü

```
&AtClient  
Procedure Komut1(Önek)  
    Önek = NumaraÖnekiniEldeEtme();  
EndProcedure
```

Eğer bu gibi genel modül – global özellikli değil ise (örneğin, **EvraklarİleÇalışma**), onun prosedürüne başvurmak için ilk önce ilgili modülün adı, sonra nokta işareteti ve sonunda da ilgili prosedür adı belirtilir (List. 5.13).

List 5.13 Form modülü

```
&AtClient  
Procedure MiktarOnChange(Item)  
    EvraklarİleÇalışma.TutarHesaplama (TabloBölümSatırı);  
EndProcedure
```

Gösterdiğimiz ikinci yöntem daha kullanışlıdır, çünkü global olmayan genel modüller sadece kendilerine başvuru olduğu zaman derlenmektedirler. Global olan genel modüllerse sistem çalışlığında derlenmektedirler.

Tabii ki bundan başka form modülünde ilgili prosedürün nasıl tanımlanmış olması (örneğin, &AtClient &AtServer vb.) ve ilgili genel modül için hangi onay kutuları (örneğin, İstemci (yönetilen uygulama), Sunucu vb.) işaretlenmiş olmaları önemlidir.

6. Yönetilen uygulama modülünün export özellikli değişkenleri, prosedürleri ve fonksiyonları

Eğer uygulama modülünde export özellikli TestMesajı() prosedürü tanımlanmış ise (List. 5.14), form modülünden direkt olarak o prosedüre başvurmak mümkündür (List. 5.15).

List 5.14 Uygulama modülü

```
Procedure TestMesajı()
    Message("Test mesajı");
EndProcedure
```

List 5.15 Form modülü

```
&AtClient
Procedure MiktarOnChange(Item)
    TestMesajı();
EndProcedure
```

Formlar – program nesneleri

Formun içerisindeki modülünden konfigürasyonun farklı kısımlarına başvurabildiği gibi, konfigürasyonun farklı kısımlarından ilgili formun kendisi de program nesnesi şeklinde ulaşılabilirdir.

Bundan başka da ManagedForm kaynak kodunun standart özellik ve metodlarından başka, geliştiricinin bu nesneye atadığı özellik ve metodları mevcut olabilir.

Eğer Alımİrsaliyesinin DocumentForm’unda MalzemelerMiktarOnChange() prosedürü export özellikli ise (List. 5.16), ilgili prosedüre farklı yerlerden de ulaşmak mümkündür (List. 5.17).

List 5.16 Form modülü

```
&AtClient
Procedure MalzemelerMiktarOnChange(Item)
    TabloBölümSatırı = Items.Malzemeler.CurrentData;
    EvraklarİleÇalışma.TutarHesaplama(TabloBölümSatırı);
EndProcedure
```

List 5.17 Form modülü

```
Form = GetForm("Document.Alımİrsaliyesi.Form.DocumentForm");
Form.MalzemelerMiktarOnChange();
```

Form modülünde olay işleyicileri - prosedürleri

Tanımlanan değişken ve sabit özellik ve metodlardan başka form modülleri, form ile bağlantılı olay işleyicilerini içerebilir. Form modülünde işlenen ana olay işleyicilerinden biri de form penceresini açma ve kapatma olay işleyicidir (List. 5.18).

List 5.18 Form modülü

```
&AtServer
```

```
Procedure OnCreateAtServer(Cancel, StandardProcessing)
```

```
    //İşleyici içeriğini ekle
```

```
EndProcedure
```

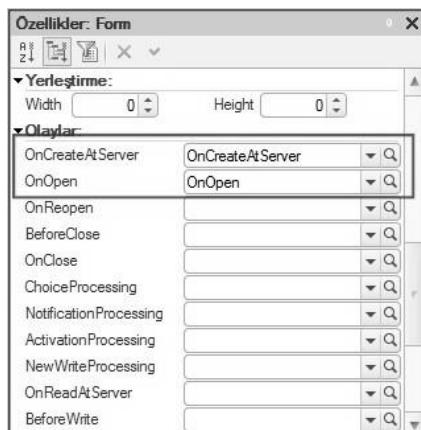
```
&AtClient
```

```
Procedure OnOpen(Cancel)
```

```
    //İşleyici içeriğini ekle
```

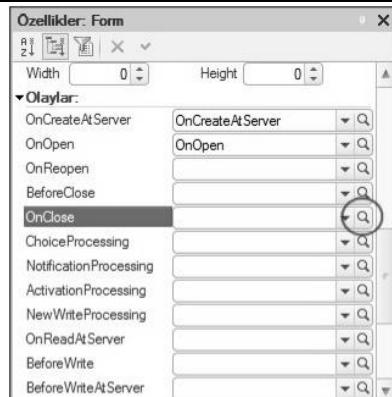
```
EndProcedure
```

Olay işleyici prosedürlerin adları sabit degildirler, gerekirse farklı isim altında belirtmek mümkündür. Burada önemli bir noktaya dikkatinizi çekelim, form modülünde ilgili isim altında prosedür (AtCreateOnServer, OnOpen) yazmanız yeteli değildir, ondan başka da formun ilgili olayında ilgili prosedürü tanımlamak (olay ve işleyicisi arasını bağlamak) gerekmektedir (Resim 5.17).



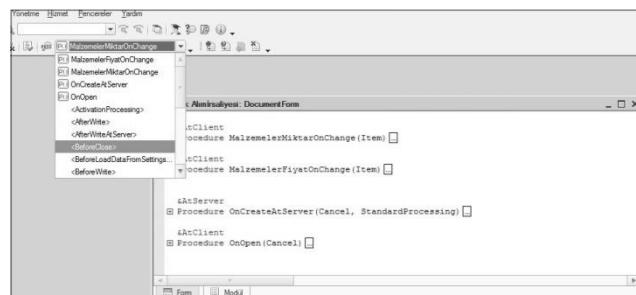
Resim 5.17. Form olayları

Yeni başlayan geliştiriciler genellikle bu küçük ayrıntıyı unutmaktadırlar. Bu gibi karışıklıkları önlemek için form olayları alanında ilgili olayı açma düğmesine (büyüteç resimli) tıklamak daha mantıklıdır (Resim 5.18).



Resim 5.18. Form olayları

Form modülünde bulunduğuuz zaman açılır penceresinden de seçmek mümkündür (5.19).



Resim 5.19. Form olayları listesi

Bu durumda sistem otomatik olarak ilgili prosedürü oluşturur ve ilgili olay arasında bağlantı kurar.

Form modülünde yazılmış olanları nasıl anlamak gereklidir

Form modülünde yazılmış prosedürü inceleyerek algoritmayı açıklayalım. Örnek olarak hazır prosedürümüzü kullanalım (**MalzemelerMiktarOnChange()** prosedürü).

Örneğe geldiğimizde şu satır ile karşılaşırız: **TabloBölümSatırı = Items.Malzemeler.CurrentData**.

Gelin hep birlikte bu satırı açıklayalım. **TabloBölümSatırı** nedir? Bunun için form bağlamının neyden ibaret olduğunu hatırlamamız gerekecek.

- form modülünün kendi bağlamı;
- form öznitelikler;
- ManagedForm nesnesi için kaynak kodunda tanımlanan özellik ve metotlar;
- ana özniteliğin türüne göre tanımlanan form uzantısının özellik ve metotları;
- global bağlam, global olmayan genel modüller ve global özellikli genel modüllerin export özellikli fonksiyon ve prosedürleri;
- yönetilen uygulama modülünün export özellikli değişkenleri, fonksiyonları ve prosedürleri.

Şimdi ise bunları **TabloBölümSatırı** için kontrol edelim:

1. Form modülünde ilgili satır tanımlanmış mı? Hayır.
2. Formda bu öznitelik mevcut mudur? Hayır.
3. ManagedForm nesnesinde bu özellik var mıdır? Hayır.
4. Form uzantısında bu özellik var mıdır? Hayır.
5. Global bağlamda bu özellik var mıdır? Hayır.
6. Yönetilen uygulama modülünde bu ad altında export özellikli değişken var mıdır? Hayır.

Demek, **TabloBölümSatırı** – form modülünün bu atama işleminde tanımlanan ve bu forma ait yeni bir değişkendir.

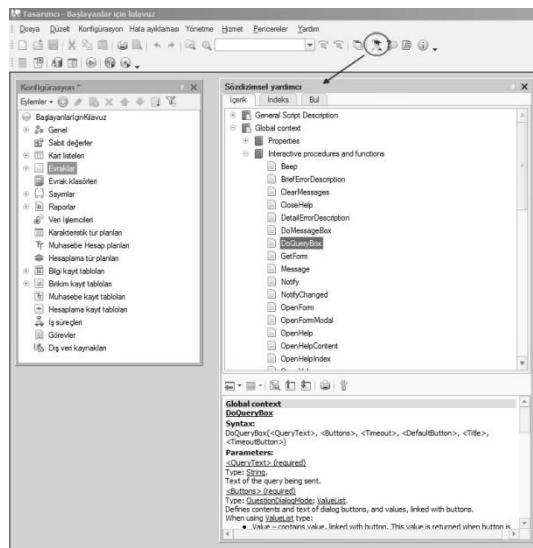
Kaynak kodunda belli bir kodun çalışmasını nasıl anlamak gereklidir?

Daha önce MalzemelerMiktarOnChange() olay işleyicisinin açıklamasını kısa bir şekilde yapmıştık.

Şimdi ise, geliştiricilerimiz için iki tane yöntem göstereceğiz. Böylece geliştiricimiz karşısına çıkan tüm metod ve özellikleri kendi başına çözebilecek ve belki de kendisinin yazacağı prosedürlerinde bu yöntemler işine yarayacaktır.

Sözdizimi-yardımcı – geliştiriciler için eklenmiş yardımcı olması için geliştirilen araçtır. Bu araç, sistemin kullandığı tüm nesneler, onların özellikleri, metodları, olayları vb. hakkında bilgi içerir.

Sözdizimi-yardımcısını açmak için tasarımcının araç çubuğuunda ilgili resimli düğme üzerine tıklamak veya ana menüden Yardım – Sözdizimi-yardımcı komutunu uygulamak gereklidir (Resim 5.20).



Resim 5.20. Sözdizimi-yardımcısı çağrıma

Her bir yardım sistemi gibi, sözdizimi-yardımcısı da ağaç şeklinde tasarlanmıştır. Burada bölümler ve alt nesneleri ve nesnelerin metodları, özellikleri, olayları vb. hiyerarşik biçimde tanımlanmıştır. Sözdizimi-yardımcısını kullanmak ilk daha uygundur. Çünkü ulaşması çok kolay ve bağamlar ile ilgili yardımı da içermektedir (Ctrl + F1 kısayol tuşu).

Sözdizimi-yardımcısı ile kodları inceleme

Daha önce yazılmış ve anlaşılmayan kodu anlamada sözdizimi-yardımcısı yardımcı olacaktır. **MalzemelerMiktarOnChange()** prosedürüümüz üzerine sözdizimi-yardımcısının nasıl kullanılacağı hakkında bilgi verelim.

Birinci yöntem

Birinci yöntem – sözdizimi-yardımcısının içerisinde bölgeler ağacında ilgili satırın ait olduğu sekmelere kadar inerek incelemek.

Evet tekrar ilgili olay işleyicisini ele alalım (List. 5.19).

List 5.19 “MalzemelerMiktarOnChange” prosedürüünün ilk satırı

```
TabloBölümSatırı = Items.Malzemeler.CurrentData;
```

Bu kodu anlamak için ilk önce hangi bağlamda bulunduğuunu bilmek gerekir. Kaynak kodu bağlamı ilgili kodun hangi modülde bulunduğuna bağlıdır. Yukarıda belirtilen kod form modülünde yerleşmektedir, dolayısıyla form modülü bağılamında yerleşmektedir.

Yazılan kodu soldan-sağá inceleyeceğiz. TabloBölümSatırı nedir? Atama operatörünün (=) sol tarafında bulunan kod, bu bağlamda doğrudan olarak başvurabileceğimiz özellik ya-da bir değişken olması gereklidir.

Gelin hep birlikte bunları inceleyelim:

- Bu form modülünde ilgili isim altında belli bir değişken tanımlanmış mı? Form modülünü açalım. Form modülünde tanımlanan herhangi bir form değişkeni (Var TabloBölümSatırı;) mevcut değildir. O zaman TabloBölümSatırı bir form değişkeni değildir.
- Formda TabloBölümSatırı adı altında bir öznitelik mevcut mudur? Alımİrsaliyesi evrakinin evrak formunu açalım (DocumentForm) ve öznitelikler penceresine geçelim (Resim 5.21).



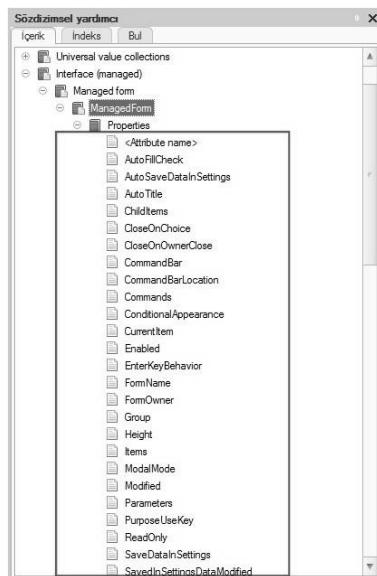
Resim 5.21. Alımİrsaliyesi evrak formunun öznitelikleri listesi

Gördüğünüz gibi, formda sadece tek Object adlı öznitelik mevcut. O zaman TabloBölümSatırı kodu bir öznitelik değil.

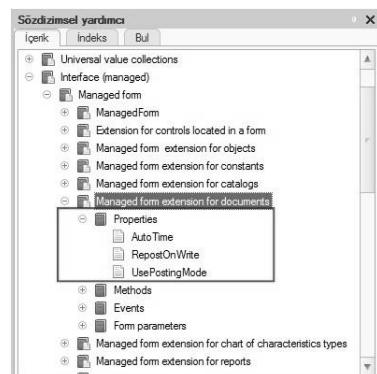
- ManagedForm nesnesinde TabloBölümSatırı özelliği mevcut mudur? Sözdizimi-yardımcıda yönetilen form özelliklerine bakalım. Sözdizimi-yardımcıda İçerik sekmesini açalım. Yönetilen form – yönetilen uygulama arayüzünün bir nesnesidir, bunun için bölgeler ağacında **Interface (managed)** – **Managed form** dallarını açalım. Sonra ise **ManagedForm** ve **Properties** dallarını da açalım (Resim 5.22).

Özellikler alfabetik sıraya göre sıralanmış durumdadır. Sıralanan özelliklerin arasında TabloBölümSatırı satırının olmadığını görüyoruz.

- Acaba form uzantısında TabloBölümSatırı özelliği mevcut mudur? Form ana özniteligi **DocumentObject.Alımİrsaliyesi** türünde nesnenin verilerini içerdiğini biliyoruz (daha önceki derslerde bahsetmiştik). Bu durumda form modülünde evrak için yönetilen form uzantısının (Managed form extension for document) özellikleri ulaşılabilir olacaktır (**Interface (managed) – Managed form – Managed form extension for document** dalları). Bu özelliklere bakalım.



Resim 5.22. Sözdizimi-yardımcıda ManagedForm (yönetilen biçimli form) nesnesinin özellikler listesi



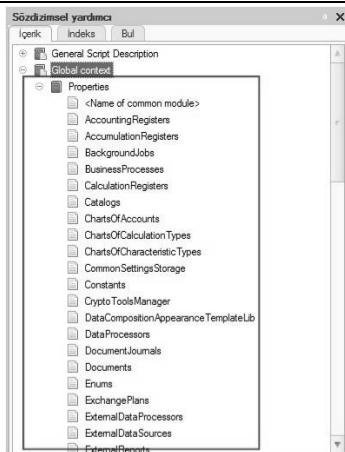
Resim 5.23. Sözdizimi-yardımcıda Evrak için yönetilen form uzantısı
nesnesinin özellikler listesi

Bu özelliklerin arasında da TabloBölümSatırı özelliğinin mevcut olmadığını görmekteyiz.

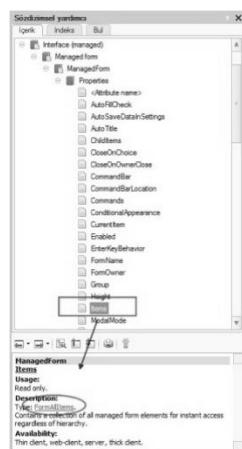
- Global bağlam özellikleri arasında TabloBölümSatırı adı altında özellik var mıdır? Sözdizimi-yardımcısında global bağlam özelliklerini açalım (Resim 5.24). Global bağlam özellikleri arasında **TabloBölümSatırı** adlı özelliğin mevcut olmadığını görüyoruz. Demek, TabloBölümSatırı bir global bağlam özelliği değildir. Global olmayan modül adı değildir, çünkü global olmayan modüllerden sonra nokta işaretleri olur (**TabloBölümSatırı()**). Export özellikli prosedür ve fonksiyonlardan da değildir, çünkü fonksiyon veya prosedür adından sonra açma-kapatma parantezi gelir (**TabloBölümSatırı()**).

Acaba yönetilen uygulama modülünde ilgili satır-değişken var mıdır? Bunun için yönetilen uygulama modülünü açalım. Yönetilen uygulama modülünde de tanımlanan değişken mevcut değil (**Var TabloBölümSatırı**). O zaman TabloBölümSatırı yönetilen uygulama modülü değildir. Böylece, **TabloBölümSatırı** ilgili **MalzemelerMiktarOnChange()** prosedürü ait bir değişken olduğunu anlamışsınızdır. Uygulama çalışma sürecinde bu değişene herhangi bir değer atanıyor. Değişkenler belli bir tipe sahip olmadıklarından dolayı onlara istenildiği zaman istenilen türden değeri atamak mümkündür. Prosedür içindeki değişkeni daha önceden değişken olarak tanımlamaya gerek yoktur, çünkü değişken ilgili prosedür işlendiğinde otomatik olarak tanımlanmaktadır. Atama operatörünün sağ tarafında **Items.Malzemeler.CurrentData** ifadesi bulunmaktadır. Items ifadesinin ne olduğunu anlamak için **TabloBölümSatırı** için geçtiğimiz işlemin aynısını tekrarlamamız gerekecektir.

- Items adı altında form modülünde bir değişken tanımlanmış mı (**Var Items**)? Hayır.
- Formda Items adlı öznitelik mevcut mu? Hayır.
- ManagedForm nesnesinde Items adlı özellik mevcut mu? ManagedForm nesnesinin özellikler listesini açalım ve Items adlı özelliği arayalım. Evet var. O zaman Items MangedForm nesnesinin özelliklerinden biridir. Items özelliğinin ne işe yaradığını bilmek için ilgili satır üzerine çift tıklamak gerektir (Resim 5.25).

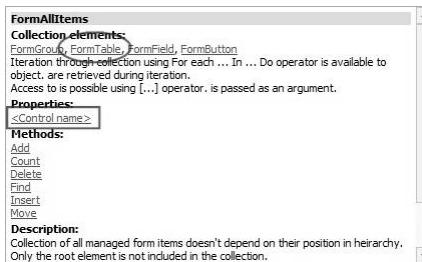


Resim 5.24. Sözdizimi-yardımcıda global bağlam özelliklerinin listesi



Resim 5.25. Sözdizimi-yardımcıda ManagedForm nesnesinin Items özelliğinin açıklaması

Sözdizimi-yardımcısının alt kısmında seçmiş olduğumuz özellik hakkında bilgi görüntülenmektedir. Sunu anlamak gereklidir, Items özelliğini kullandığımızda FormAllItems (formun tüm denetimleri) nesnesini elde ederiz. Bu nesne ise, formda mevcut olan tüm denetimleri içerir. Bu nesnenin ne işe yarlığını anlamak için ilgili köprü üzerine tıklamak yeterlidir. Sözdizimi-yardımcının alt kısmında ilgili nesne (FormAllItems) hakkında bilgi görüntülenecektir (Resim 5.26).



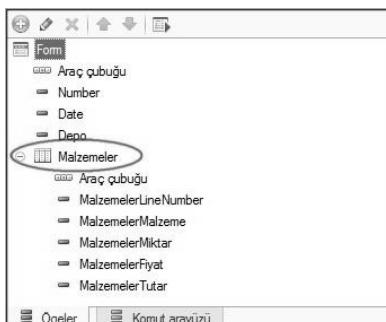
Resim 5.26. Sözdizimi-yardımcıda FormAllItems koleksiyonunun açıklaması

NOT:

Fark ettiyseniz, sözdizimi-yardımcısının üst kısmı hala değişmemiştir. Açıklama kısmında belirtilenleri sözdizimi-yardımcısında aramak için açıklama bölümünün hemen üstündeki araç çubuğuunda bulunan **Cari ögeyi ağaçta ara** düğmesine () tıklamak yeterlidir.

Bu koleksiyon formda eklenen tüm yönetilen form öğelerini içermektedir. Gerekli öğeye sadece adını belirterek başvurulur.

Artık **Items** kodunun ne işe yaradığını biliyoruz. Şimdi ise geri kalanını araştıralım: **Items.Malzemeler**, nokta işaretinden sonra **Malzemeler** ismi gelmekte. Biz ise, **FormAllItems** koleksiyonunda **<Control name>** (denetim adı) adlı özelliğini olduğunu biliyoruz. O zaman "Malzemeler" bir denetim adı olmaktadır. Formun denetimler listesine baktamız gerekecektir. Bunun için **Alımİrsaliyesi** evrakinin evrak formunu açalım. Formun üst kısmında denetimlerin yapısı bulunmaktadır.



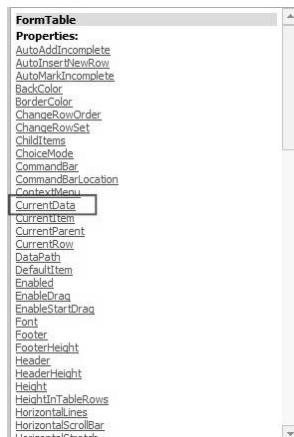
Resim 5.27. Alımİrsaliyesi evrak formunun denetimler yapısı

Form denetimleri yapısında **Malzemeler** satırını görmekteyiz. Bu satırın özellikler paletini açtığımızda paletin başlığında – **Özellikler: Tablo** tanımını göreceksiniz (Resim 5.28).



Resim 5.28. Tablo – form denetiminin özellikler paleti

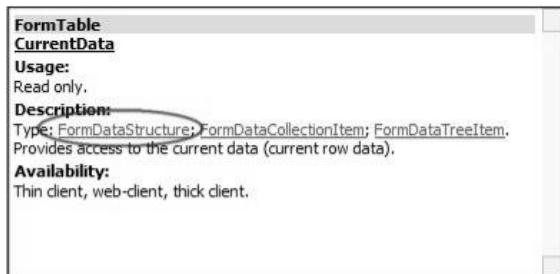
Demek, bu denetimin türü tabloymuş. Bunun için **FormTable** koleksiyonuna bakmak gereklidir. Ne olduğu hakkında daha fazla bilgi edinmek için ilgili köprü üzerine tıklamak gereklidir (bkz resim 5.26). Açılan sekmede FormTable nesnesinin özelliklerini göreceksiniz (Resim 5.29).



Resim 5.29. Sözdizimi-yardımcıda FormTable nesnesinin özelliklerinin listesi

Artık Items.Malzemeler kodunun da ne işe yaradığını biliyoruz. Şimdi ise noktadan sonraki satırı bakalım: Items.Malzemeler.CurrentData. Yönetilen

form tablosunun özellikler listesinde CurrentData satırını görmekteyiz. Demek CurrentData satırı FormTable nesnesine ait bir özellikmiş. İlgili satırı tıklayarak açıklama kısmına görelim (Resim 5.30).



Resim 5.30. Sözdizimi-yardımcıda FormTable nesnesinin CurrentData özelliğinin açıklaması

Sözdizimi-yardımcının alt kısmında seçmiş olduğumuz satırın açıklaması görüntülenir. Ve buradan anlaşılır ki, CurrentData özelliğini kullanarak FormDataStructure nesnesini elde ederiz. Bu nesne ise, cari satırda mevcut olan verileri içerir.

Demek, kaynak kodunun ilk satırında (**TabloBölümSatırı = Items.Malzemeler.CurrentData;**) bizim **TabloBölümSatırı** değişkenimize **FormDataStructure** türlü değer atanmakta.

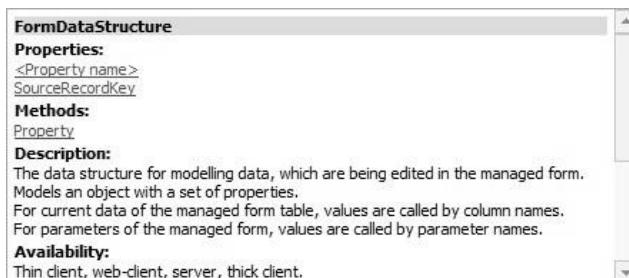
Olay işleyicisinin bir sonraki satırına geçelim (List. 5.20).

List 5.20 “MalzemelerMiktarOnChange” prosedürüne ikinci satırı

```
TabloBölümSatırı.Tutar = TabloBölümSatırı.Miktar *
```

```
TabloBölümSatırı.Fiyat;
```

Mantık olarak baktığında Tutat, Miktar ve Fiyat kodları sanki **FormDataStructure** nesnesinin birer özellikleri gibidir. Bunun ne olduğunu anlamak için ilgili köprü üzerine tıklayalım (Resim 5.30, 5.31).



Resim 5.31. Sözdizimi-yardımcıda FormDataStructure nesnesinin açıklaması

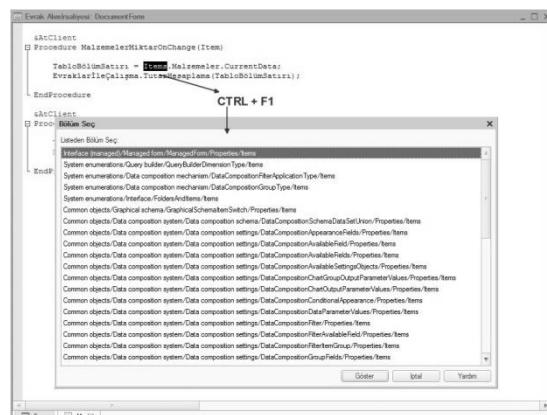
Burada anlatılanlara göre, FormDataStructure nesnesini kullanarak, tablo bölümündeki belli bir sütunun değerini almak mümkündür. Bunun için o sütunların isimlerini belirmek gerekmektedir. Örneğin, TabloBölümSatırı.Tutar = tutar sütunun değeri. Sonuç olarak şunlar elde edilir: Miktar sütunda belirtilen değer ile Fiyat sütundaki değer çarpılır ve sonucu da Tutar sütuna atanır.

İkinci yöntem

İkinci yöntem – sözdizimi-yardımcısının bağlam yardımcısını kullanmaktadır. Bunun için program modülünü açmak gereklidir. Modülde bizi ilgilendiren kod üzerine fare işaretçisini yerleştirerek **Control+F1** kısayol tuşuna basmak gerekmektedir. Bunu için Alımİrsaliyesi evrak formunu açalım ve modül kısmına geçelim, **MalzemelerMiktarOnChange()** prosedürüne gelelim.

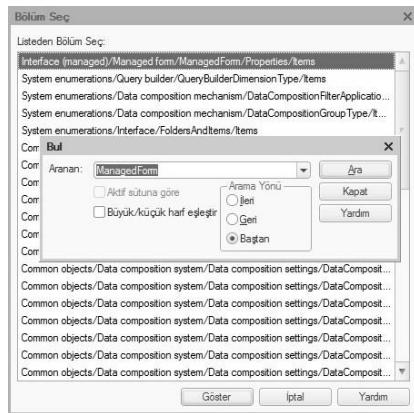
TabloBölümSatırı koduna biz daha önce o prosedürün değişkeni demiştik. Şimdi ise atama operatörünün sağ tarafındaki kodlara gelelim. Fare işaretçisini Items kodunun üzerine yerleştirirelim ve Ctrl+F1 kısayol tuş takımını tıklayalım. Sözdizimi yardımcısı İndeks sekmesinde açılır ve alfabeyle göre sıralanan tüm özelliklerin listesinde **Items** tanımına göre arama başlatılmış olur.

Ifade bulunmuştur. Ve ayrı bir pencerede bu ifadenin kullanılmış olduğu başlıkların tümü sıralanır. Açılan pencerede gerekli satır üzerine çift tıklayarak satırın kaynağuna ulaşmak mümkündür (Resim 5.32).



Resim 5.32. Sözdizimi-yardımcıda bağlam yardımını çağrıma

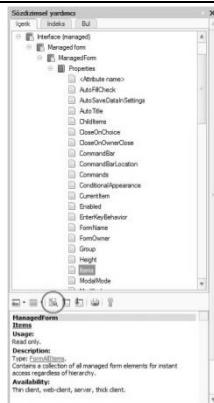
Ama Items kelimesinin kullanıldığı bölümler çok fazla, gerekirse bu bölümler arasında arama yapmak mümkün değildir. Bunun için Ctrl+F tuş takımını tıklamak yeterlidir. Bölümler listesinde iken yeni bir pencere açılacaktır. Açılan arama penceresinde aranacak ifade alanına «ManagedForm» yazalım ve ara düğmesine tıklayalım. Arama sonucunda bulunan satır seçilmiş olacaktır (Resim 5.33).



Resim 5.33. Bağlam aramasının bölgümler listesinde arama

Göster düğmesine tıklandığında ise, ilgili bölümün açıklaması sözdizimi-yardımcısının alt kısmında açılacaktır. Bölümler ağacı ise olduğu gibi kalacaktır. İlgili nesneyi bölgümler ağacında bulmak için **Cari öğeyi ağaçta ara** komutunu uygulamak yeterlidir.

İçerik sekmesinde ilgili satırın ait olduğu bölüm açılacaktır. Görüüğünüz gibi, Items satırı ManagedForm nesnesine ait bir özelliktir. İlgili köprüleri açtığımız zaman TabloBölümSatırı değişkenine neler atandığını anlarız ve sütunundaki verilerle nasıl çalışıldığını anlayabiliriz.



Resim 5.34. ManagedForm nesnesinin

Items özelliğinin açıklaması

Bundan başka da, sözdizimi-yardımcısı fazla karışık gösteriyorsa içeriğine belli bir sınırlama getirmek mümkündür. Örneğin, bizi ilgilendiren sadece Web-istemci ve İnce-istemci kaynak kodları ise, ilgili yönlendirmeye göre sınırlama getirmek mümkündür. Bunun için **Hizmet – Parametreler...** komutunu uygulamak veya sözdizimi-yardımcı penceresinde **Parametre ayarları biçimini aç** düğmesine tıklamak gereklidir. Açılan pencerede **Yardım** sekmesinde **Sözdizimi** grubunda ilgili onay kutularını onaylamak yeterlidir (Resim 5.35).



Resim 5.35. Sözdizimi-yardımcı görüntüülenen nesnelere filtre uygulamak

NOT:

Örneğin, sunucuda kullanılacak prosedür yazmanız gerekiği durumda ve sözdizimi-yardımcısını kullanmanız gerekiyorsa, parametreler penceresinden ilgili onay kutusunu işaretlemeyi unutmayın.

Hata ayıklama biçimi ile kod analizi

Hata ayıklama biçimi daha çok kendi kodumuzu yazdığımızda yararlı olacaktır. Sözdizimi-yardımcısından farklı olarak, nesnelerin yapısını düşünerek yapılacak kodun kullanımını tamamıyla kafanızda canlandırmak için çaba sarf etmeye gerek yok.

Yapılması gereken sadece programın belli bir yerinde durmak ve burada hangi eylemler ulaşılabilir veya hangi nesneler kullanılabilir olduğuna bakmaktadır.

Hata ayıklayııcı – 1C:İşletme sisteminin program modüllerini geliştirmede ve hata ayıklamasında kullanılan yardımcı bir araçtır. Hata ayıklama biçimini aşağıdaki imkanları sağlamaktadır:

- modülleri adım adım gerçekleştireme,
- durak noktalarını yerleştirme,
- modül gerçekleştirme işlemini durdurma ve devam ettirme,
- aynı zaman birkaç modül için hata ayıklama imkanı,
- değişken durumlarını analiz edebilmek için ifadeleri hesaplayabilme imkanı,
- prosedür ve fonksiyonların çağrı katmanı,
- hata çıktıığı durumda duraklama imkanı,
- hata ayıklama biçiminde iken modül düzeltme imkanı.

Eğer Tasarımcı ortamında modül metni düzeltiliyorsa, ana menüde hata ayıklama menüsü görüntülenir. Bu menü ise durak noktalarını yerleştirme ve kaldırma imkanını sağlar. Durak noktası – modülün neresinde yerleştirildiyse, işlem sırası o satra gelindiğinde duraklama gerçekleşir. Böylece geliştirici değişkenlerin o zamanki değerlerini, ifade tiplerini ve işlem sırasında değerleri takip edebilir, bir sonraki durak noktasına kadar devam edebilir vb.

Evet Alımlısalıyesi evrak formunu açalım ve modül sekmesine geçelim. MalzemelerMiktarOnChange() prosedürüne gelelim. Hata ayıklama menüsünde ve araç çubuğuunda durak noktasını yerleştirme komutlarının etkinleştirilmiş olduğunu göreceksiniz (Resim 5.36).



Resim 5.36. Durak noktaları araç çubuğu

Bundan başka da ilgili satırında çalışma alanının sol tarafında fare ile çift tıklayarak durak noktası yerleştirmek mümkündür. Bu durumda çalışma alanının sol kısmında durak noktası görüntülenecektir.

NOT:

İşlemenin durdurulacağı satır fare işaretçisi yerleştirilir, sonra **Hata ayıklaması – Durak noktası** komutunu uygulamak gereklidir veya

Durak noktası araç çubuğunda **Durak noktası**  düğmesine tıklamak gereklidir.

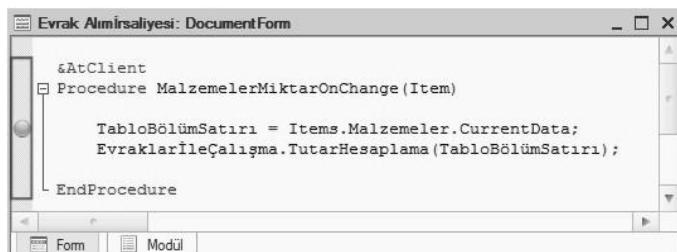
Durak noktasını kaldırma için aynı satırın solunda ilgili durak noktası işaretini üzerine çift tıklamak yeterlidir.

NOT:

Durak noktası yerleştirilmiş olan satır üzerine fare işaretçisini yerleştirerek menüden **Hata ayıklaması – Durak noktasını devre dışı bırak** komutunu uygulamak veya **Durak noktasını devre dışı bırak**  düğmesine tıklamak gereklidir.

Tüm durak noktalarını kaldırma için **Hata ayıklaması – Tüm durak noktaları kaldır** komutunu uygulamak veya **Durak noktası** araç çubuğundaki ilgili düğmeyi kullanmak gereklidir.

Evet şimdi **MalzemelerMiktarOnChange()** prosedürüne ilk satırının sol kısmı üzerine fare ile çift tıklayalım (Resim 5.37).

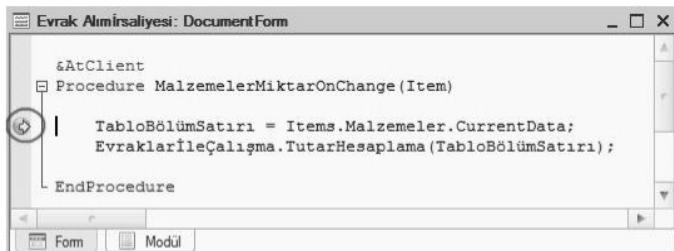


Resim 5.37. **MalzemelerMiktarOnChange()** prosedüründe
durak noktasını yerleştirme

Yazılan kodları birer birer hayata geçirmek için hata açıklama biçiminde ilgili işlemleri yapmak gereklidir. Bunun için ilk başta **Hata ayıklaması – Hata**

ayıklamasını başlat komutunu veya araç çubuğundaki ilgili komutu uygulayalım ve hata ayıklama biçiminde kullanıcı ortamına geçelim.

Alım ırsaliyeleri listesini açalım ve oluşturmuş olduğumuz herhangi bir evrakı açalım. Herhangi bir satırın miktar sütununu değiştirelim. İşlem durdurularak tasarımcı ortamında bırakmış olduğumuz durak noktası görüntülenir. Durak noktasının yanında ok işaretti belirecektir. Bu işaret gerçekleşmekte olan satırı görüntüler (Resim 5.38).



Resim 5.38. Sisteminde durak noktasında işlemi durdurması

Bundan başka da Hata ayıklaması menüsünde ve araç çubuğunda konfigürasyon ile çalışma komutları belirecektir (Resim 5.39).



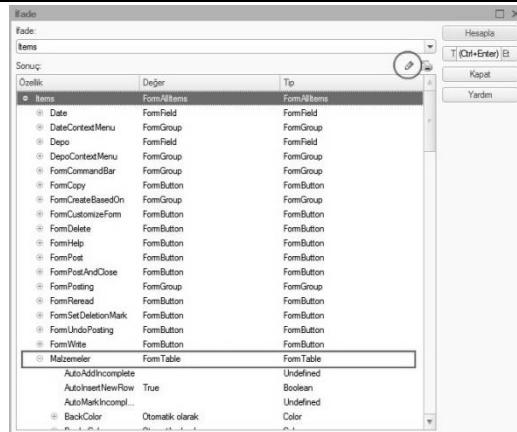
Resim 5.39. Konfigürasyon hata ayıklama araç çubuğu

Bu komut ve düğmelerin yardımcı ile yapılacak olan işlemi adım adım takip etmek veya gerekiyorsa bir sonraki durak noktasına kadar hata ayıklamasını devam ettirmek mümkündür. İfade hesapla ve Tablo komutlarının yardımcı ile istediğiniz zaman belli bir ifadenin değerini alabilirsiniz. Çağrı katmanı düğmesi yardımcı ile prosedür ve fonksiyonların işleyiş sıralamasını almak mümkündür.

Bizi ilgilendiren satırda durakladık. Satır daha işlenmediğinden dolayı değişkenimiz hiçbir şey içermeyecektir. Onların ne içerdiğini öğrenmek için Geçiş yap komutunu uygulamak gerektir.

Evet biz Items.Malzemeler.CurrentData satırının ne işe yaradığını ve ne olduğunu bilmek istiyoruz. Bunun için Items kodu üzerine çift tıklayalım ve İfade hesapla düğmesine tıklayalım (Shift+F9 tuş takımı).

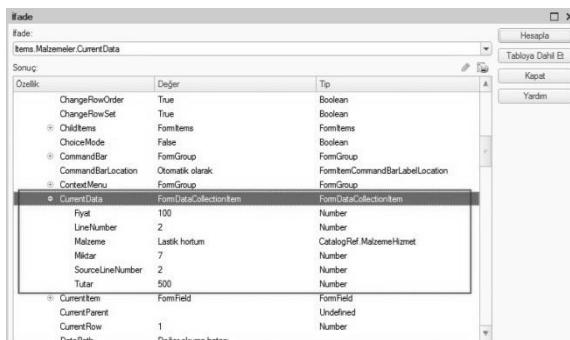
Seçmiş olduğumuz Items kodu ifade alanına aktarılacaktır. Değer ve tip alanlarında ilgili satırın değeri ve tipi tanımlandığını görmektesiniz. Örneğimizde Items kodu formda mevcut olan tüm denetimleri içermektedir. Bu nesneyi açalım (Resim 5.40).



Resim 5.40. Items nesnesi

Sonuç alanının üstünde Değeri ayrı pencerede göster düğmesine tıklandığında Items nesnesinin içерdiği bütün koleksiyonu ayrı pencerede açar.

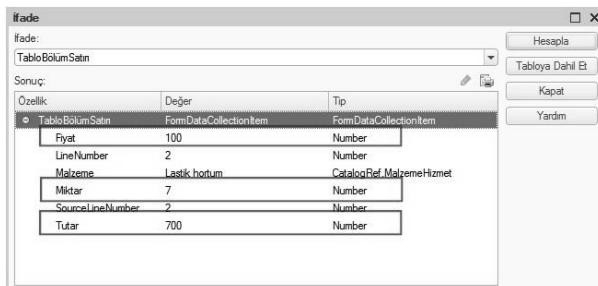
Daha sonra bizi ilgilendiren Malzemeler denetimi. Tabloda onu bulalım. Malzemeler satırını da açtığımızda özelliklerini göreceğiz. Özellikleri arasında ise CurrentData özelliğini aramamız gerekecek. CurrentData özelliğinin altında tablo bölümünün verilerini, değerlerini ve tiplerini göreceksiniz. Miktar alanının değerine ise bizim yeni değiştirmiş olduğumuz alan gelmiş oldu (Resim 5.41).



Resim 5.41. Items.Malzemeler.CurrentData nesnesi

Şuanda TabloBölümSatırı değişken belirsizdir. Çünkü modül işleme oku daha onu geçmedi. Geçiş yap komutunu kullanalım. Modül işleme oku bir sonraki kod satırına gelecektir. TabloBölümSatırı kodu üzerine çift tıklayarak

İfade hesapla düğmesine tıklayalım (Resim 5.42). Az önce Items.Malzemeler.CurrentData kod satırı için baktığımızın aynısı TabloBölümSatırı için atanmış oldu (TabloBölümSatırı = Items.Malzemeler.CurrentData). Şimdi ise İçe geçiş yap komutunu uygulamayalım. Bu komut, satırda belirtilen EvraklarİleÇalışma genel modülünde bulunan TutarHesaplama prosedürüne geçiş yapacaktır. Tutar hesaplama algoritması burada tanımlanmıştır. Geçiş yapılan prosedürde de Geçiş yap komutunu kullanalım. Sistem otomatik olarak belirtilen algoritmeye göre hesaplama yapar. Yapılan hesaplamanın sonucunu izleyebilmek için Tutar kodunun üzerine fare işaretçisini yerleştirelim. Sistem açılır ipucu olarak hesaplanan tutarı gösterir (Resim 5.43).



Resim 5.42. TabloBölümSatırı nesnesi



Resim 5.43. TabloBölümSatırı nesnesi

Değişken ve ifadeler analiz edildikten sonra **Hata ayıklamaya devam et** düğmesi ile hata ayıklama biçiminde kullanıcı ortamında çalışmalara devam etmek mümkündür. Eğer gerekirse hata ayıklamasını sonlandırmak da mümkünkündür (**Hata ayıklamasını sonlandır** komutu).

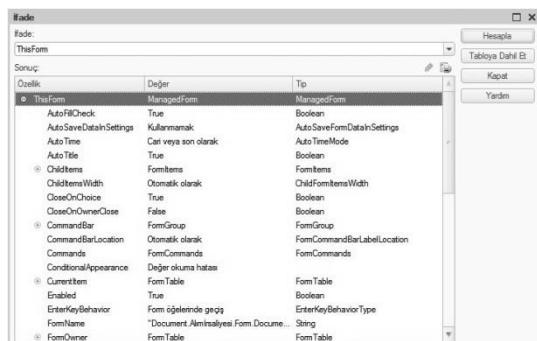
NOT:

Hata ayıklama biçiminde iken, ayıklanan modüllerin değiştirilmek mümkündür, ama değiştirilen kodu derlememektedir – veritabanı konfigürasyonu kullanılmaktadır. Bu yüzden, yapılan değişiklikleri kullanıcı ortamında işleyebilmek için konfigürasyonu kaydetmek gereklidir.

Sonuç olarak, hata ayıklama biçimini kullanmada son faydalı bilgimizi de aktaralım.

Varsayılm ki siz form modülünde bulunuyorsunuz ve herhangi bir olay işleyici yazmanız gerekiyor. Bu formla ilgili olan tüm bağlamları izleyebilmek istiyoruz. Bunun için, durak noktası ile herhangi bir yerde

durdugunuz zaman **İfade hesapla (Shift+F)** düğmesine tıkmanız gereklidir. Açılan pencerede ifade alanında ThisForm yazmanız ve Hesapla düğmesine tıkmanız yeterlidir (Resim 5.44).



Resim 5.44. ThisForm nesnesi

Bu nesneyi açtığınızda duraklama alanında kullanılan kaynak kodu nesnelerinin tip ve özelliklerini göreceksiniz.

Aynen bu şekilde de nesne modülünde veya kayıt modülünde iken, ThisObject anahtar kelime ile ilgili nesnenin veya kaydın modül bağlam özelliklerini izlemek mümkündür.

Nesneler

1C: İşletme terimleri arasında "nesne" terimi ne anlama gelmekte? Bu soru sadece yeni başlayan geliştiricileri değil, bu sistemde belli bir yol kat etmiş, belli bir tecrübe sahip geliştiricileri de bazen çıkmaz sokaga sokmaktadır.

Bunu anlama konusunda bir tane zorluk vardır – o da terimin hangi bağlamda kullanıldığını iyi bilmektir.

Bu terim genel olarak üç bağlamda kullanılmaktadır:

- konfigürasyon,
- veritabanı,
- kaynak kodu.

Hep birlikte bu üç sekme hakkında açıklama getirelim.

Konfigürasyon. **Konfigürasyon nesnesi** terimi kullanıldığı zaman, verileri yapılandırma seti ve bu verilerle çalışmak için tanımlanan algoritmaların seti anlatılmaktadır. Örneğin, konfigürasyonda Personeller Kart listesi nesnesi mevcut olabilir.

Bu konfigürasyon nesnelerine bağlı olarak veritabanında bilgi (veri) yapısı oluşturulmaktadır. Bu yapının içinde ise veriler saklanacaktır.

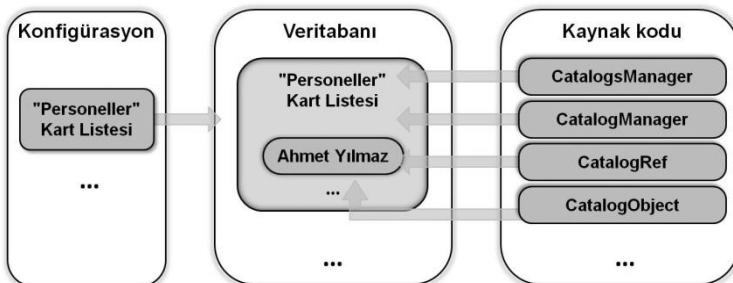
Veritabanı. **Veritabanı nesnesi** olarak biz sadece bu bilgi yapısının herhangi bir öğesini tanımlamaktayız. Bu öğelerin asıl karakteristik özelliklerinden biri de, farklı bilgi yapılarından kendilerine referans mevcut olmasıdır ve o referansın ilgili bilgi yapısının bir değeri olmasıdır.

Örneğin, veritabanında Personeller adlı kart listesi mevcut ve bu kart listesinde ise Ahmet Yılmaz adlı bir personel mevcut. Örneğin, Alımlırsalıyesi evrakında Sorumlu adlı öznitelik mevcut ve onun da türü Personeller kart listesine referans ise, bu durumda bu personel hakkında bilgileri içeren kart liste öğesine veritabanı nesnesi denir.

Kaynak kodu nesnesi – eğer söz konusu kaynak kodu ve kaynak kodunu kullanarak kart listelerle çalışma hakkında bahsediliyorsa, verilere ulaşabilmeyi sağlayan ve özellik ve metot setlerini içeren veri türlerine kaynak kodu nesnesi deriz.

1C: İşletme sisteminde kart listelerle çalışmayı sağlayan birçok kaynak kodu nesnesi vardır (CatalogsManager, CatalogManager.<Catalog name>,

CatalogRef.<Catalog name> vb.). Veritabanında kart listesi nesnesine erişim sağlayan tek bir nesne mevcuttur, o da – CatalogObject.<Catalog name>.



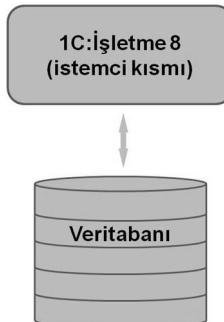
Resim 5.45. Konfigürasyon nesni, veritabanı nesni, kaynak kodu nesni

Sunucu ve istemciler

Daha önce birkaç olay işleyicileri için genel modülde prosedür eklemiştir (Evrakları ile Çalışma genel modülü). Bu modülü oluşturduğumuzda bu modülün özelliklerinde **İstemci** ve **Sunucu** onay kutularını işaretlemiştik. Bu terimler hakkında biraz daha bilgi verelim.

1C:İşletme sistemi iki şekilde çalışma biçimine sahiptir: dosya biçimli çalışma ve istemci-sunucu biçimli çalışma.

Dosya biçimli çalışma şekli, tek kullanıcı veya ağ üzerinde az miktarda kullanıcı sayısının veritabanı çalışması için tasarlanmıştır. Bu biçimli çalışmada veritabanının bütün dosyaları tek dosya içinde tutulmaktadır.

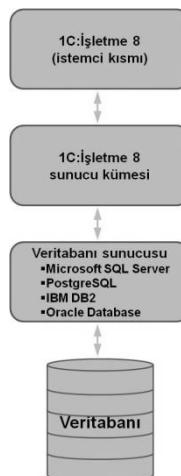


Resim 5.46. Dosya biçimli çalışma

Dosya biçimli çalışmanın asıl amacı – hızlı ve kolay bir şekilde sistemin kurulumunu yapmak ve onunla çalışmaya başlamak. Dosya biçimli çalışmada

gerçek hesap yürütme mümkün. Ama şunu da unutmamak gereklidir ki, verileri koruma ve ölçeklendirme konusunda dosya biçimci istemci-sunucu biçimli çalışmadan daha düşüktür. Bunun için siz eğer kendi başınıza muhasebe hesabını yürütüyorsanız veya işletme aç miktarda personel mevcut ise, kullandığınız verilerin hacmi pek fazla değil ise, dosya biçimli çalışma gerçekleştirmek mümkün. Aksi halde istemci-sunucu biçimli çalışma gerekmektedir.

İstemci-sunucu biçimli çalışma gruplu çalışmalarında veya büyük hacimli verileri kullanan işletmeler için tasarlanmıştır. İstemci-sunucu biçimli çalışma üç seviyeli mimari yapı üzerine gerçekleştirilmiştir (Resim 5.47).



Resim 5.47. 1C:İşletme sisteminin üç seviyeli mimarisi

İstemci-sunucu biçimli çalışması – büyük hacimli verilerle çalışma ortamında kullanıcıların çoklu olarak çalışması için tasarlanmış bir biçimdir. Verileri koruma, yönetme ve ölçeklendirme konularında bu biçim yeteri kadar imkanları sağlamaktadır. Yalnız bu biçim kurulumu ve yönetimi dosya biçimci gibi kolay değildir.

Ayrıca 1C:İşletme 8 istemci ve sunucu kısımları maddesel olarak birbirinden farklı yerlerde olabileceği gibi, aynı yerde de olabilirler. Burada önemli olan, kullanıcılar doğrudan olarak verilere ulaşamamaktadır, bu ise verilerin korumasını sağlamaktadır. Dosya biçimli çalışma ise böyle değildir, veritabanı dosyası ağ üzerindeki bütün kullanıcılarla açıktr.

Şuanda geliştirdiğimiz uygulama dosya biçimli uygulama. Bu uygulamayı istemci-sunucu biçimli de çalışıtmak mümkün.

Uygulama çözümleri bir defa ve iki biçimdedir aynı şekilde çalışmaktadır. Yani, bir biçimden diğer biçimde geçiş için konfigürasyon yapısında değişiklik yapmaya gerek yoktur.

Bunun sebebi, uygulama geliştirilirken bile istemci-sunucu biçimine göre geliştirilmektedir. 1C:İşletme sisteminde hiçbir şekilde başka türlü geliştirilmemektedir. Dosya biçimli çalışma kullanıldığı zaman ise, sistem otomatik olarak istemci bilgisayarında sunucu varmış gibi davranışmaktadır.

İstemci-sunucu mimarisi, çalışmakta olan tüm sistemi kendi aralarında belli çalışma prensiplerine göre çalışan 3 ayrı bölüme böler – **İstemci, 1C:İşletme Sunucusu ve Veritabanı Sunucusu**.

İstemci uygulaması – program, 1C:İşletme sisteminin bir kısmıdır. Asıl amacı – kullanıcı arayüzüni organize etmek, verileri görüntüleme ve değiştirebilme. Bundan başka da istemci uygulaması kaynak kodundaki bazı kodları (geliştircinin tanımlamış olduğu bazı algoritmaları) kullanabilmektedir. İstemci uygulamasının işleyebildiği kaynak kodu türleri sınırlanmıştır. Bu gibi yaklaşım, istemci uygulamasının çok daha hafif, fazla kaynak gerektirmeyen, İnternet üzerinde gezinmek, hatta Web-tarayıcılarla birlikte çalışabilmesi sağlamaktadır.

İstemci uygulaması 1C:İşletme Sunucusu ile etkileşim içerisindeidir. 1C:İşletme Sunucusu – bu da bir program ve 1C:İşletme sisteminin bir parçasıdır.

Bu sunucunun asıl amacı, istemci uygulamasından almış olduğu sorguları veritabanı sunucusuna yönlendirmek ve sunucudan alınan sonucu istemciye aktarmaktır.

Sunucunun diğer görevi – kaynak kodundaki çoğu algoritmaları gerçekleştirmek, form, rapor görüntülemek için verileri hazırlama vb. Verilerle doğrudan çalışarak zor hesaplamalar sunucuda yapılmaktadır. Bundan başka da sunucu için kaynak kodundaki çoğu algoritmalar sunucuda ulaşılabilirdir (tamamen arayüzle ilgili algoritmalar hariçtir, çünkü sunucuda arayüz ile ilgili kullanım yanı yoktur, sadece istemci uygulaması ve veritabanı sunucusu ile etkileşimde olduğundan dolayı kullanıcı ile hiç irtibatı yoktur).

Veritabanı sunucusu – bu bir programdır. Bu artık 1C:İşletme sisteminin bir parçası değildir. Bu farklı amaçlara özel yapılmış ayrı programdır. Bunun asıl amacı – fiziksel veya sanal sistemlerin karakteristiklerini açıklayan yapısal veri setlerinin veritabanını yürütmek ve organize etmektir.

Bu günkü gündə 1C:İşletme sistemi aşağıda sıralanan veritabanı sunucuları ile çalışabilmektedir:

- Microsoft SQL Server,
- PostgreSQL,
- IBM DB2,

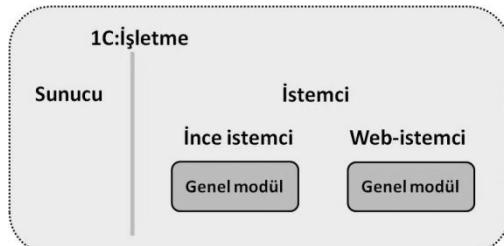
- Oracle Database.

Genel modülleri derleme

Daha önce bir genel modül oluşturmuştu (EvraklarİleÇalışma). Bu genel modülün özelliklerinde olduğu gibi tüm genel modüllerin özellikleri İstemci (yönetilen uygulama), Sunucu ve Dış bağlantı. Bu alanların anlamları nedir – bu modüllerin örnekleri hangi modülde derleneceği anlamına gelir. Gelin hep birlikte bu konuyu açalım.

İlk önce derleme olayına neden gerek duyuldu, onu anlamak gereklidir. Asıl anlatılması gereken, konfigürasyonda geliştirdiğimiz ve yazmış olduğumuz her şey sadece bir hazır paket (şablon) şeklinde dir. Uygulamayı 1C:İşletme ortamında açtığımızda platform bu paketi bilgisayar ortamında kullanabilecek programa dönüştür – derleme işlemi gerçekleştirir. Ayrıca daha önce de belirttiğimiz gibi, kaynak kodların sistemin farklı yerlerinde işlenirler – sunucu, istemci uygulamaları. Bunun için genel modüllerin özelliklerinde ilgili modülün hangi kısımda derleneceğini belirtmek gereklidir.

EvraklarİleÇalışma genel modülü için İstemci (yönetilen uygulama) onay kutusu işaretlenmiştir (bakınız resim 4.24). Bu şu anlama gelir: bu modülün kopyası sadece İnce ve Web-istemci bağlamında istemci tarafında derlenir (Resim 5.48).



Resim 5.48. Sunucu veya istemci tarafında genel modülü derleme şeması

Eğer genel modülde sadece Sunucu onay kutusu işaretlenmiş ise, modül sadece sunucu tarafında derlenecektir.

İstemci uygulaması – sadece iki tanedir: İnce istemci ve Web-istemci.

İnce istemci – 1cv8c.exe dosyasıdır. Hata ayıklamasını başlattığımızda platform bu dosyayı çalıştırır aslında.

Web-istemci – dosya biçiminde çalıştırılmaz, çünkü bu biçim işletim sistemi ortamında çalışmaz, internet-tarayıcıları ortamında çalıştırılır. Kullanıcı sadece tarayıcısını çalıştırması gereklidir, sonra veritabanının yayımlanmış

olduğu Web-sunucu adresini girmesi gereklidir. Girdiğinde Web-istemci kullanıcı bilgisayarına yüklenir ve çalışmaya başlar.

Derleme direktifleri

Daha önce biz **OnChange** olay işleyicisini oluşturmuştu. Sitem ilgili prosedürü otomatik oluştururken, prosedürün önüne birer derleme direktiflerini yerleştirmiştir (**&AtClient**). Hep birlikte konuyu biraz daha açalım.

Form hem sunucuda, hem de istemcide bulunmaktadır. Bunun için form modülündeki her bir prosedür için ilgili prosedürün sistemin hangi kısmında – istemcide veya sunucuda uygulanacağını belirtmek gereklidir. **&AtServer** veya **&AtClient** veya **&AtServerNoContext** derleme direktiflerinin mevcut olmasının sebebi, kaynak kodunu kullanmadan istemci prosedürü net bir şekilde sunucu prosedüründen farklı olması gerekmektedir. Prosedür ve fonksiyonların önlerine direktifleri koyarak geliştirici açık bir şekilde sunucu ve istemci işlemlerini kontrol edebilmektedir.

Form modülünde aynı zaman farklı işlemler için kullanılan direktiflere sahip prosedürleri tanımlamak ve bazı algoritmaların işlenmesi için istemciden sunucuya görev aktarmak mümkündür. Ayrıca form modülündeki istemci prosedüründen genel modülde hem istemcide hem sunucuda işlenebilecek prosedürü çağrılmak mümkündür (ilgili modülün özelliklerine bakılması gereklidir – İnce istemci, Sunucu).

Kaynak kodunu sunucuda veya istemcide işlemek

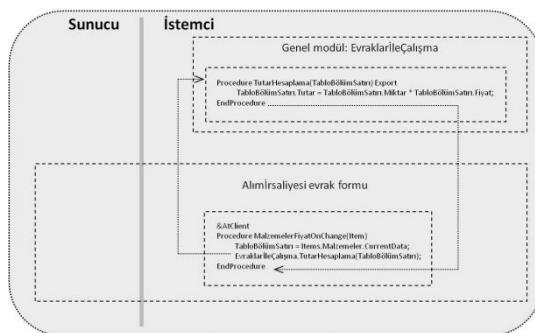
Daha önce sizinle birlikte birkaç olay işleyicisinin kullandığı bir prosedürü yazdık ve Evrakları ile Çalışma genel modülünde yerleştirdik. Bu modülün özelliklerinde ise İstemci (yönetilen uygulama) onay kutusu işaretlendi ve Sunucu onay kutusunun ise işaretri kaldırılmıştı. Bu prosedür form modülünde **&AtClient** direktifli prosedürden çağrılmıştır. Bu işlemin tam olarak nasıl gerçekleştiğini inceleyelim.

Uygulama çözümü başlatıldıktan sonra, bütün işlemler ilk önce istemcide başlatılır. Algoritma sürecinde kodu işlemi görevi Sunucuda gerçekleştirilebilmesi için Sunucu onay kutusu işaretlenen genel modüllerdeki prosedürleri çağrılmak gereklidir.

Aslında prosedür çağrısı gerçekleşende ilk önce istemcide arama yapılır. Eğer istemcide derlenen prosedür bulunmuyorsa, sunucuda arama yapılmaktadır. Eğer çağrılan prosedür bulunursa, ilgili işlemin gerçekleşmesi

sunucuda gerçekleşir. Sunucuda işlem bittiği zaman işlem istemcide devam eder.

Örneğin, tutar hesaplama işlemini kontrol edelim. Alım ırsaliyesinin form modülünde FiyatOnChange prosedürü çağrılmaktır. Bu prosedür istemci tarafında ince istemci biçiminde gerçekleşmektedir. Olay işleyicisinin ikinci satırında TutarHesaplama() prosedürü çağrılmaktadır. Bu prosedür istemcide derlenen EvraklarİleÇalışma modülünde yerleştiği için çağrılan işlem ince istemci biçiminde gerçekleşmektedir. Bu prosedür tamamlandıktan sonra algoritma form modülü yönetimine aktarılır (Resim 5.49).



Resim 5.49. İstemcide modüller arasında algoritma gerçekleştirmeye şeması

DERS 6

Birikim Kayıt Tabloları

Süre

Dersin tahmin süresi – 50 dakika

Birikim kayıt tablosu ne için kullanılır	176
Birikim kayıt tablosu nedir	177
Birikim kayıt tablosunu ekleme	178
Evrap kayıt hareketleri	181
Evrap formunda evrap hareketlerine geçiş komudu	188
Hizmet faturası evrakının hareketleri	189

Bu derste *Birikim kayıt tablosu* konfigürasyon nesnesi ile tanışacağız. Bu nesne ne iş yaptığını, yapısı ne olduğunu ve genel özelliklerini ile ilgili bilgiye sahip olacaksınız.

Ondan sonra önceden oluşturmuş olduğumuz evrakların hareketlerini kaydedecek olan bir birikim kayıt tablosu oluşturacağız.

Birikim kayıt tablosu ne için kullanılır

Her geliştirilen konfigürasyonun en önemli kısmına geldik; veri birikimi hesaplama yapısını oluşturacağız.

Gereken her şey oluşturmuş gibiyiz; ne almak ve satmak gerektiğini mevcut (kart listeleri), ne ile almak ve satmak da mevcut (evraklar). “Aspirin usta” şirketinin otomasyonunu tamamlamak için geri kalan şey, sadece birkaç rapor oluşturmak.

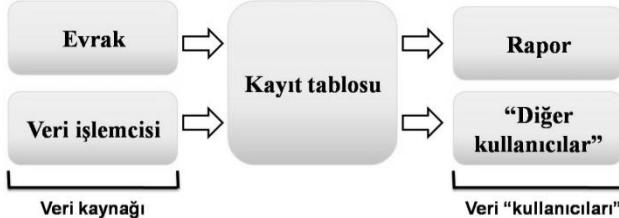
Fakat bu doğru değil.

Birinci evrakları analiz ederek gerekli veri elde edebiliriz. Fakat varsayıyalı ki, öbür gün “Aspirin usta” şirketi iş sürecini değiştirmeye kararını verir ve konfigürasyona daha bir tane (ya ya birkaç tane) evrak girmemizi gereklidir.

Örneğin, şu anda malzemeler ilk önce şirkete gelir sonra kullanımı sırasında şirketten sıkar. Yöneticiler stok hareketleri daha kontrollü bir şekilde getirmek istediler; şirkete gelen mallar ilk önce Genel depoya alınır, ondan sonra sorumlu kişiye verilir. Bu durumda, depo ve sorumlu kişi arasında malzeme hareketini kaydedecek bir tane daha evrak eklememiz gereklidir. Büyük ihtimalde, yeni evrak hareketlerinin genel hesaplamalarda yansıtılması için önceden oluşturulmuş olduğu stok durumları yansıt tüm raporları tekrar düzeltmek gereklidir. Peki, konfigürasyonumuzda iki değil yirmi evrak ise?

İkinci evraklardan veri çeken evraklar çok yavaş çalışır bu da kullanıcıların kızmalarına ve yöneticilerin memnun kalmalarına sebep olabilir.

Bu yüzden 1C:İşletme sisteminde veri analizi için uygun bir şekilde veritabanında veri birikim yapılarını oluşturmayı sağlayan birkaç konfigürasyon nesneleri mevcuttur. Böyle veri “depolarının” kullanımını, bize bir taraftan farklı evraklardan gelen verileri tek yerde toplamaya olanak sağlar, ikinci taraftan gerekli raporları kolayca oluşturma ve bu verileri konfigürasyon çalışma algoritmalarına kullanmaya imkan verir.



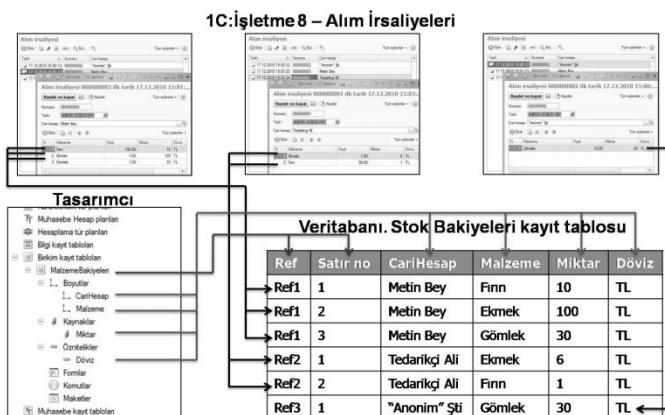
Resim 5.1. Konfigürasyon çalışma yapısı

Böyle depoları tanımlama için konfigürasyonda *kayıt tablosu* denen birkaç nesne vardır.

Birikim kayıt tablosu nedir

Birikim kayıt tablosu veri birikme yapısını tanımlamak için kullanılan konfigürasyon nesne türüdür. Birikim kayıt tablosu üzerinde platform veritabanında tabloları oluşturur, bu tablolar farklı veritabanı nesnelerinden “gelen” veriyi biriktirmek için kullanılmaktadır.

Bu veriler tablolarda ayrı bir kayıt olarak depolanacaktır. Her kayıt, tasarımcıda oluşturulmuş, aynı yapıya sahiptir (resim 6.2).



Resim 6.2. Tasarımcıda, 1C:İşletme ortamında ve veritabanında “StokBakiyeleri” kayıt tablosu

Birikim kayıt tablosunun hareket tablosu üzerinden sistem kayıt tablosunun bakiye tablosunu hesaplar. Bu tabloda son hareket tarihi için bakiye bilgileri içerir (güncel bakiye bilgileri).

Birikim kayıt tablosunun ayırcılıklarda biri şudur; birikim kayıt tablosu kullanıcı tarafından doğrudan düzeltilmek için tasarılanmamış.

Geliştirici gerektiğiinde kullanıcıya birikim kayıt tablo kayıtlarının düzeltme hakkı verebilir. Fakat birikim kayıt tablosunun amacı, kullanıcıların direkt

veri girmesi değil, konfigürasyonun diğer nesnelerin oluşturduğu hareket bilgilerini depolamaktır.

Birikim kayıt tablosunun genel amacı, birden fazla **boyuta** göre sayısal bilgilerin birikmesidir. Boyutlar birikim kayıt tablosu konfigürasyon nesnesinin alt konfigürasyon nesnesidir.

Birikmiş olan sayısal bilgilere **kaynaklar** denir. Kaynaklar da birikim kayıt tablosu konfigürasyon nesnesinin alt konfigürasyon nesnesidir.

Örneğin birikim kayıt tablosu stokta bulunan malzeme miktarı ve tutarı ile ilgili bilgi depolayacak. Bu durumda *Malzeme* ve *Depo* iki boyut ve *Miktar* ve *Tutar* iki kaynağı sahip olacaktır.

Birikim kayıt tablosunun durumu genelde evrak kaydedildiği zaman değişir. Evrak kaydedildiğinde kayıt tablosuna bu evrak ile ilgili bilgi eklenir. Birikim kayıt tablosunun her kaydı boyut bilgilerini, eklenen kaynak bilgilerini, değişiklik yapan evraka (kayıt eden) referans bilgilerini ve kaynak bilgi eklemeye “yönü” (Giren veya Çıkan) içerir. Böyle bir kayıt setine *kayıt tablo hareketleri* denir. Kayıt tablosunun her hareketi için belli bir *kayıt eden* atanmalıdır – hareketi oluşturan veritabanı nesnesi (genelde evrak).

Bunun dışında birikim kayıt tablosu her hareketi ile ilgili ek bilgi depolayabilir. Ek bilgi özellikleri, birikim kayıt tablosu konfigürasyon nesnesinin alt konfigürasyon nesnesi olan **öznitelikler** tarafından belirlenir.

Birikim kayıt tablosunu ekleme

Birikim kayıt tabloları ne için tasarlardığını öğrenmişistik. Artık birim örneğimizde birikim kayıt tablolarını nasıl kullanabileceğimize bakalım.

İlk önce hangi malzemeler stokumuzda bulunduğu bilmek isteriz. Bu bilgiyi birikmek için **StokBakiyesi** kayıt tablosunun oluşturulalım.

Tasarımcı ortamında

Tasarımcıda geliştirdiğimiz konfigürasyonumuzu açalım ve yeni Birikim kayıt tablosu konfigürasyon nesnesini ekleylelim.

Bunu yapmak için konfigürasyon ağacında *Birikim kayıt tabloları* grubu işaretleyerek konfigürasyon ağacının araç çubuğundan **Ekle** butona tıklayalım.

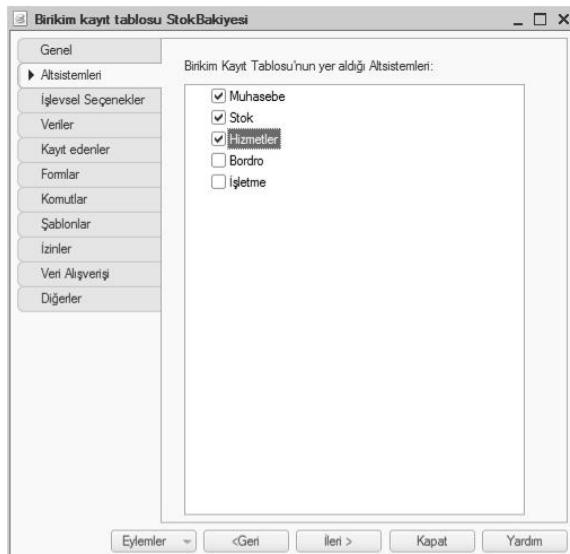
Ekrana gelen konfigürasyon nesne düzeltme penceresinde *Genel* sekmesinde nesne adı tanımlayalım – **StokBakiyesi**.

Aynı sekmede Liste form tanımı belirleyelim – **Stok bakiye kayıt tablosunun hareketleri**. Bu başlık birikim kayıt tablosunun liste formunda gösterilecektir.

İleri butona tıklayıp *Alt sistemler* sekmesine geçelim.

Konfigürasyon yapısına göre bu birikim kayıt tablosunun **Stok**, **Hizmetler** ve **Muhasebe** bölümlerde yansıtılması gereklidir.

Bu yüzden alt sistem listesinde ilgili alt sistemleri işaretleyelim (Resim 6.3).

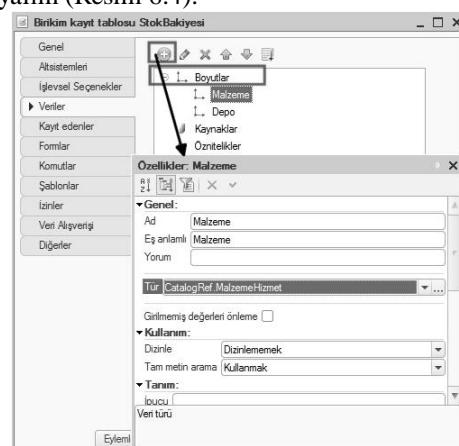


Resim 6.3. Kayıt tablosu yansıtılacağı alt sistemleri belirtme

Veriler sekmesine geçip kayıt tablo yapısını oluşturmak gereklidir. Kayıt tablo boyutları oluşturalım;

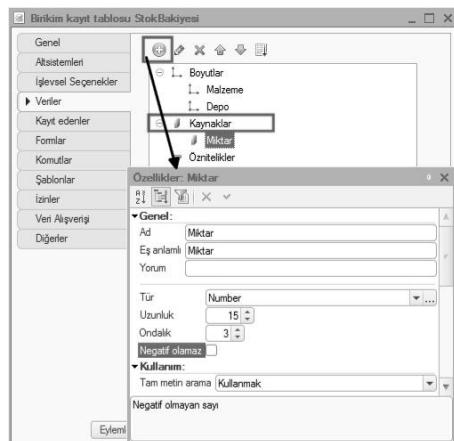
- **Malzeme**, tip *CatalogRef.MalzemeHizmet*,
- **Depo**, tip *CatalogRef.Depolar*.

Bunu yapmak için Boyutlar grubu işaretleyip pencerenin araç çubuğunda **Ekle** butonuna tıklayalım (Resim 6.4).



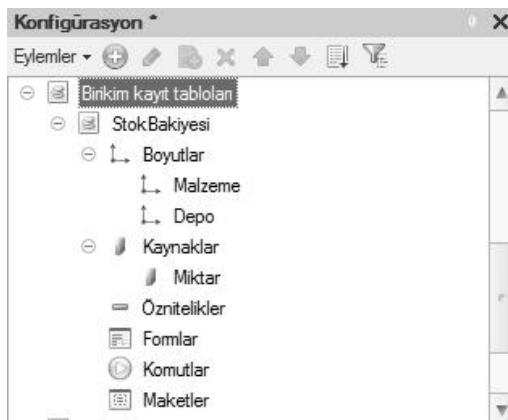
Resim 6.4. Kayıt tablo boyutunu oluşturma

Ondan sonra uzunluk 15 ve ondalık 3 özelliklerine sahip olan bir **Miktar** kaynak oluşturalım. Bunu yapmak için *Kaynaklar* grubu işaretleyip pencerenin araç çubuğuunda **Ekle** butona tıklayalım (Resim 6.5).



Resim 6.5 Kayıt tablo kaynağını oluşturma

Sonuç olarak StokBakiyesi birikim kayıt tablosu aşağıdaki gibi görünmelidir (resim 6.5).



Resim 6.6 "StokBakiyesi" kayıt tablosu

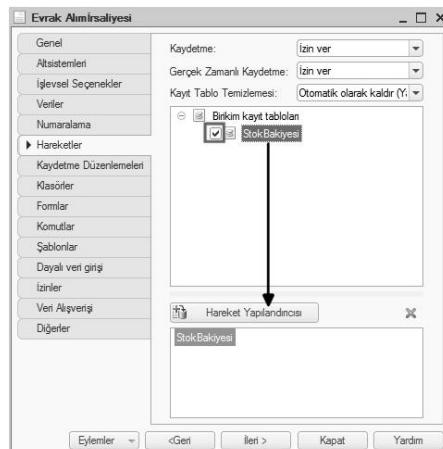
Şimdi 1C:İşletme sistemini hata ayıklama biçiminde çalıştırmayı denerseniz, sistem hatayı verir; “AccumulationRegister.StokBakiyesi: Evraklardan hiç biri kayıt tablosunda kayıt yapmamaktadır”. Bu mesaj tekrar birikim kayıt tablosunun amacını hatırlar bize; farklı evraklar tarafından sağlanan veriyi kaydetmek.

Bu yüzden oluşturduğumuz *Alımİrsaliyesi* ve *HizmetFaturası* için *StokBakiyesi* birikim kayıt tablosuna kaydetme prosedürleri oluşturalım.

Evrak kayıt hareketleri

Evrak kayıt hareketleri - Evrak kaydetme sırasında kayıt tablolarında oluşturduğu kaytlara evrak kayıt hareketleri denir. Evrak kayıt harekletleri evrakin oluşturduğu değişiklikleri yansıtır.

Alımİrsaliyesi evraki konfigürasyon nesnesinin düzeltme penceresini açalım. *Hareketler* sekmesine geçtikten sonra *Birikim kayıt tablolari* grubu açıp *StokBakiyesi* kayıt tablosunun işaretleyelim (Resim 6.7).



Resim 6.7. “Alımİrsaliyesi” evrak hareketlerinin
“StokBakiyesi” kayıt tablosunda oluşturma

Kayıt tablosunu seçtikten sonra **Hareket yapılandırıcı** butonu ulaşılabilir hale gelmiştir. Bu butona basıp yapılandırıcıyı kullanalım.

Yapilandırıcı yapısı basittir. *Kayıt tablolari* listesinde evrakin hareket yapılacak kayıt tabloları bulunur. Bizim örneğimizde sadece bir kayıt tablosu mevcuttur - *StokBakiyesi*.



Resim .68. Hareket yapılandırıcısı

Evrak öznitelikleri lisesinde hareketi oluşturmak için kaynak veri bulunur – Alımİrsaliyesi evrakın öznitelikleri.

Alan ve İfade tablosunda, evrak kaydetme sürecinde gerçekleşecek olan kayıt tablosunun boyut, kaynak ve özniteliklere kaydetme algoritmaları girilmelidir (Resim 6.8).

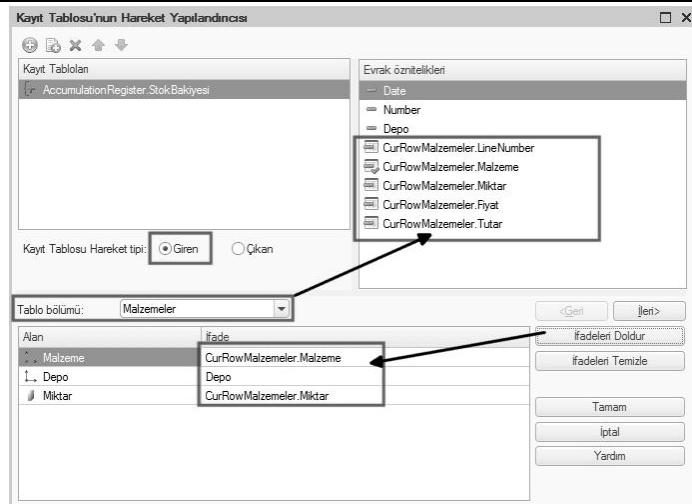
Varsayılan değer olarak hareket yapılandırıcısı *StokBakiyesi* kayıt tablosu için giriş hareketleri oluşturma seçeneğini işaretledi (Kayıt tablosu Hareket tipi - Giren). Biçim için bu uygun bir seçenek, çünkü Alımİrsaliyesi stoka ürünlerin girmesini sağlayan bir evraktır.

Tablo bölümü seçme alanında evrakımızın tablo bölümü seçelim – *Malzemeler*.

Evrakın öznitelik listesine, tablo bölümündeki bulunan öznitelikler otomatik olarak eklenir.

İfadeleri Doldur butona basalım.

Pencerenin alt kısmında kayıt tablo alanlarının hesaplama ifadeleri otomatik olarak oluşturulur (Resim 6.9).



Resim 6.9. Evrakin tablo bölümünü seçme ve kayıt tablo hareketlerini hesaplama için ifade doldurma

Gördüğünüz gibi, hareket yapılandırıcı ilişkiler aşağıdaki gibi kurdur; malzeme olarak kayıt tablosuna evrakin tablo bölümünde bulunan malzeme kaydedilecek, depo olarak evrakin başlıkta bulunan depo kaydedilecek ve miktar olarak evrakin tablo bölümünde bulunan miktar kaydedilecektir.

Tamam butonuna basalım ve hareket yapılandırıcısının Alımlısalıyesi evraki için oluşturduğu koda bakalım (liste 6.1).

Liste 6.1. "Posting" prosedürü

```
Procedure Posting(Cancel, Mode)
    //{{__REGISTER_REGISTERRECORDS_WIZARD
    // Bu parça yapılandırıcı tarafından oluşturuldu.
    // Yapılandırıcıyı tekrar kullanma sürecinde elle kaydedilen değişiklikler
    // kaybedilecektir!!!

    // kayıt tablosu StokBakiyesi Receipt
    RegisterRecords.StokBakiyesi.Write = True;
    For Each CurRowMalzemeler In Malzemeler Do
        Record = RegisterRecords.StokBakiyesi.Add();
        Record.RecordType = AccumulationRecordType.Receipt;
        Record.Period = Date;
        Record.Malzeme = CurRowMalzemeler.Malzeme;
        Record.Depo = Depo;
        Record.Miktar = CurRowMalzemeler.Miktar;
    EndDo;
```

```
//}}__REGISTER_REGISTERRECORDS_WIZARD
EndProcedure
```

Yapılandırıcı *AlımSaliyesi* evrakı konfigürasyon nesnesinin *Posting* olay isleyicisini oluşturdu, prosedürü nesne modülüne yerleştirdi ve modül metnini ekrana getirdi.

Posting prosedürü evrakin en önemli olaylardan biridir. Bu olay evrak kaydedildiğinde çalıştırılır. Bu prosedürün genel amacı kaydetme sırasında evrak hareketlerini oluşturmaktır. Prosedürdeki işlemler genel hesap borç/alacak durumlarıdır. Bu yüzden geliştirici, evrak kaydedildiği zaman oluşturulmuş veri hareket algoritmalarını bu prosedürde tanımlamalıdır.

Prosedürün metnini açıklayalım.

DocumentObject kaynak kod nesnesi *RegisterRecords* özelliğine sahip. Bu özellik, evrakin hareket yapabildiği kayıt tablosu kayıt seti koleksiyonunu içeren *RegisterRecordsCollection* nesnesini iade eder.

Bu koleksiyonun somut bir kayıt setine ulaşmak için nokta ile kayıt tablo adını belirterek ulaşılabilir. Örneğin; *RegisterRecords.StokBakiyesi*.

Sonra nokta ile kayıt tablosu kayıt setinin farklı metodlar kullanılabilir, örneğin; *RegisterRecords.StokBakiyesi.Add()*. *Add()* metodu kayıt setine yeni kayıt ekler.

Prosedürün birinci satırında *Write* özelliği için *True* değeri tanımlıyoruz. Yani direkt, kaydetme işlemi sonucunda platformun kayıt setini veritabanına kaydetmek gerektiğini belirtiyoruz.

Döngüde evrakin tablo bölümüne ulaşması tablo bölüm adı ile yapılır (*Malzemeler*). *CurRowMalzemeler* değişkeni evrakin tablo bölümünün geçerli satır bilgisini içerir. Bu değişken döngü başında oluşturulur ve döngü işleme sürecinde değişir.

Döngü gövdesindeki birinci satırda *Add()* metodu kullanarak, evrakin kayıt tablosunda oluşturan kayıt setine yeni bir satır eklenmektedir.

Böylece biz *AccumulationRegisterRecord* nesnesini oluşturup *Record* değişkene onu kaydediyoruz.

Bu nesneyi kullanarak, alan adını noktayla belirleyerek kaydın alanlarına ulaşabiliriz (Örneğin, *Record.Miktar*).

Record.Malzeme ve *Record.Depo* - kayıt tablosunun boyutları, *Record.Miktar* - kayıt tablosunun kaynağı, *Record.RecordType* ve *Record.Period* - otomatik olarak oluşturulmuş kayıt tablosunun standart öznitelikleridir.

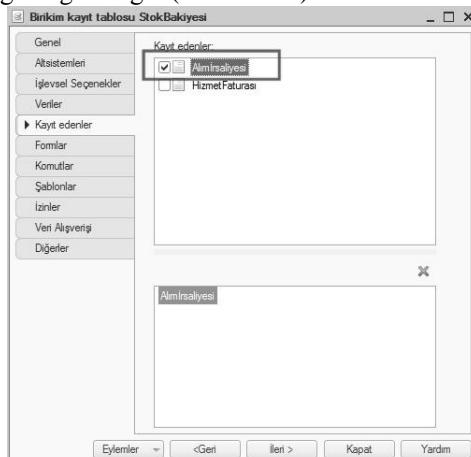
Kayıt tablosunun yeni kayıt alanları için ilgili evrak alan değerlerini atamak için, *CurRowMalzemeler* değişkeni kullanarak, alan adını nokta ile belirleyerek evrakin tablo bölüm alanlarına ulaşabiliriz (*CurRowMalzemeler.Malzeme*).

Depo - evrak başlık özniteliği ve Date - otomatik olarak oluşturulmuş evrakin standart özniteliğidir. Döngüde sadece evrakin tablo bölüm değerleri değişir - *CurRowMalzemeler.Malzeme* ve *CurRowMalzemeler.Miktar*. Diğer alanlar değişmez, çünkü diğer alanlar evrakin standart alanları ve tablo bölüm satırına göre değişmez.

`AccumulationRecordType.Receipt` – birikim kayıt tablosunun hareketini Giren olarak tanımlayan bir sistem sayımlarıdır.

Böylece biz yeni kaydın tüm alanları için gerekli değer tanımlıyoruz. Evraktaki tüm tablo bölümleri geçtikten sonra (döngü bittiğinde), kayıt setinde (*Record.StokBakiyesi*) kayıt edilen evrakında tablo bölümündeki satır sayısına kadar satır bulunur.

Artık, *StokBakiyesi* birikim kayıt tablosu konfigürasyon nesnesinin düzeltme penceresini açtığımızda *Kayıt edenler* sekmesine geçersek, konfigürasyonda var olan evraklar listesinde Alımlısalıyesi evrakının işaretlenmiş olduğunu göreceğiz (Resim 6.10).



Resim 6.10. “StokBakiyesi” kayıt tablosunun kayıt edenleri

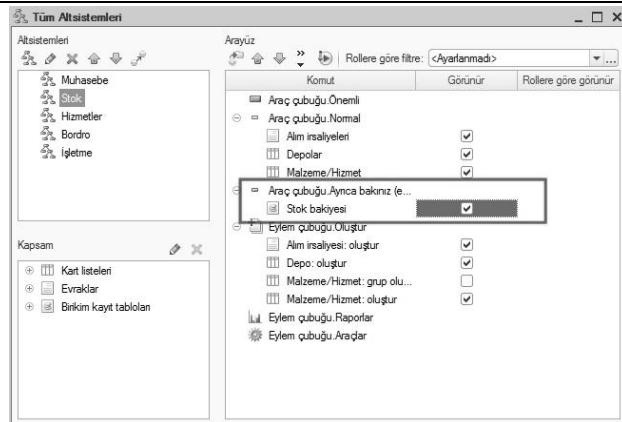
Son olarak uygulama arayüzüne düzeltelim. *Stok*, *Hizmetler* ve *Muhasebe* alt sistemlerinde, birikim kayıt tablosunun kayıtlarını izleyebilmek için komutu yerleştirmek gereklidir.

Kayıt tabloları açma komutları alt sistemin araç çubuğunda eklenir, fakat evraklardan farklı olarak varsayılan değer olarak kayıt tablosu açma komutu devre dışıdır.

Konfigürasyon nesne ağacında *Alt sistemler* grubu işaretleyip sağ tuş ile çağrılan menüden **Tüm alt sistemler** maddesini seçelim. Ekrana gelen pencerede soldaki listeden *Stok* alt sistemi işaretleyelim.

Sağ tarafta bulunan *Arayız* listesinde seçilen alt sistemini tüm komutlar gösterilir.

Eylem çubuğu. Normal grubunda **Stok bakiyesi** komutu etkinleştirelim ve **Araç çubuğu.** Ayrıca **bakınız** grubuna taşıyalım (Resim 6.11).



Resim 6.11. Alt sistemin arayüz ayarlaması

Birim kayıt tabloları açma komutu az kullanılır, bu onu yüzden arayüz araç çubuğunun *Ayrıca bakınız* grubuna yerleştirmek daha iyi olur.

Aynı şekilde kayıt tablosunun arayüzde gösterme komutunu Hizmetler ve Muhasebe alt sisteminde araç çubuğunun “*ayrıca bakınız*” bölümüne yerlestirelim.

1C:İşletme ortamında

1C:İşletme sistemini hata ayıklama biçiminde çalıştırıralım ve yapmış olduğumuz değişiklikleri kontrol edelim.

Açılan 1C:İşletme penceresinde Stok, Hizmetler ve Muhasebe bölümlerin araç çubuklarında Stok bakiyesi kayıt tablosunu görüntülemek için *Stok Bakiyesi* komutu ortaya çıktı (Resim 6.12).

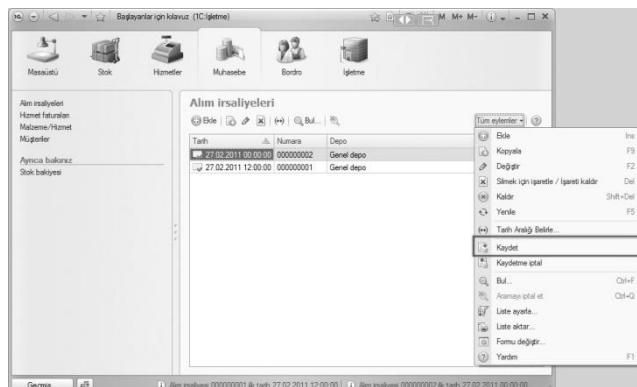


Resim 6.12. “StokBakiyesi” kayıt tablosunun listesi

Evrak kaydetme ve kayıt tablosundaki veri birikmesi arasında bağlantıya görebilmek için *Muhasebe* bölümünde bulunan *Alım ırsaliyeleri* komutu kullanarak alım ırsaliye listesini açalım.

1 nolu alım ırsaliyesini açalım ve **Kaydet ve kapat** butona basalım (yani evrakı tekrar kaydedelim). Aynı şekilde 2 nolu alım ırsaliyesini tekrar kaydedelim.

Evrakların yeniden kaydetmesi evrakı açmadan da yapılabilir. Bunu yapmak için listede yeniden kaydetmek istenilen evrakı işaretlemek (veya Ctrl tuşu basılı tutup birden fazla evrakı seçmek) ve **Eylemler** menüden **Kaydet** maddesini seçmek yeterlidir (Resim 6.13).



Resim 6.13. Evrak kaydetmesi

Şimdi Stok bakiyesi komutu kullanarak kayıt tablo listesini açalım (Resim 6.14).

Stok bakiye kayıt tablosunun hareketleri					
Doneşen	Kayıt eden	Sat..	Malzeme	Depo	Miktar
+ 27.02.2011 00:00:00	Alım ırsaliye 00000002 ik. tarih 27.02.2011 00:00:00	1	Elektrik kablo	Genel depo	5,000
+ 27.02.2011 00:00:00	Alım ırsaliye 00000002 ik. tarih 27.02.2011 00:00:00	2	Lastik hortum	Genel depo	5,000
+ 27.02.2011 12:00:00	Alım ırsaliye 00000001 ik. tarih 27.02.2011 12:00:00	1	Flyback transformör GoldStar	Genel depo	10,000
+ 27.02.2011 12:00:00	Alım ırsaliye 00000001 ik. tarih 27.02.2011 12:00:00	2	Flyback transformör Samusung	Genel depo	10,000
27.02.2011 12:00:00	Alım ırsaliye 00000001 ik. tarih 27.02.2011 12:00:00	3	Transistor Philips 2N2369	Genel depo	10,000

Resim 6.14. "StokBakiyesi" kayıt tablo listesi

Alım ırsaliyeleri kaydettiğimiz zaman Stok bakiyesi kayıt tablosunda kayıtların eklendiğini görüyoruz.

Kayıt tablosunun tüm alanları, *Alım İrsaliyesi* evrakının *Posting* prosedüründe belirlediğimiz algoritma göre doldurulmuştur. Her kaydın sol tarafında bulunan + simgesi hareket tipi gösterir – Giren.

Evrap formunda evrap hareketlerine geçiş komutu

Tasarımcı ortamında

Gerçek çalışmalarında StokBakiyesi kayıt tablosunda çok sayıda kayıt olacak ve hangi hayıtlar hangi evraka ait olduğunu bulmak zor olur.

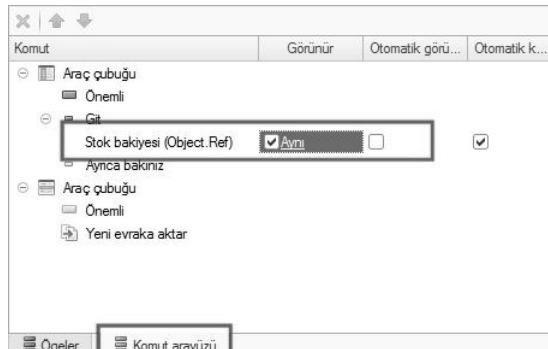
Bu yüzden kayıt tablosunun yanı sıra evrap formundan sadece o evrap ile ilgili kayıt tablo kayıtlarını görebilmek iyi olur.

Bu fonksiyonu tanımlamak için tasarımcıya geri dönelim ve **Alımİrsaliyesinin** evrap formunu (DocumentForm) açalım.

Pencerenin sol üst kısmında *Komut arayüzü* sekmesine geçelim.

Araç çubuğu bölümünde **Git** grubu açalım. Git grubu içinde StokBakiyesi kayıt tablosunu açmak için bir komutu göreceğiz. Bu komut, evrakin formuna otomatik olarak yerleştirilmiş, çünkü bu kayıt tablosu için Alımİrsaliyesi evraklı kayıt eden bir evraktır.

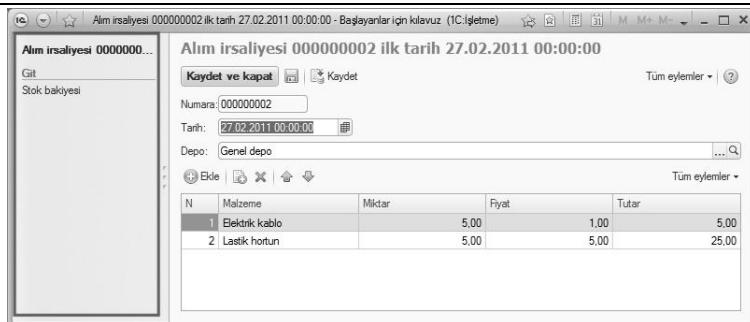
Bu komut için **Görünür** özelliğini işaretleyelim.



Resim 6.15. Evrap formunun arayüzü ayarlama

1C: İşletme ortamında

1C:İşletme sistemini hata ayıklama biçiminde çalıştıralım ve 2 nolu alım ırsaliyesini açalım (Resim 6.16).



Resim 6.16. “Alım irsaliyesi” evrakinin araç çubuğu

Evrak formunda araç çubuğu görüyoruz. Araç çubuğu kullanarak evrakin yaptığı kayıt tablo kayıtlarına geçebiliriz, bunu yapmak için *Stok bakiyesi* yazısına tıklamak yeterlidir (Resim 6.17) ve evrakin formuna geri dönmek için Alım irsaliyesi yazısı kullanılabilir.



Resim 6.17. Evrak formundan kayıt tablo kayıtlarına geçiş

Önceden, alım irsaliyesi evrak formunda araç çubuğu görünmüyordu. Araç çubuğu sadece içinde komut var olduğunda görünür.

Hizmet faturası evrakinin hareketleri

Aynı şekilde Hizmet faturasının hareketleri tanımlayalım.

Tasarımcı ortamında

HizmetFatura evrakı konfigürasyon nesnesinin düzeltme penceresini açalım.

Hareketler sekmesine geçip kayıt tablosu listesinde *StokBakiyesi* kayıt tablosunu işaretleyelim.

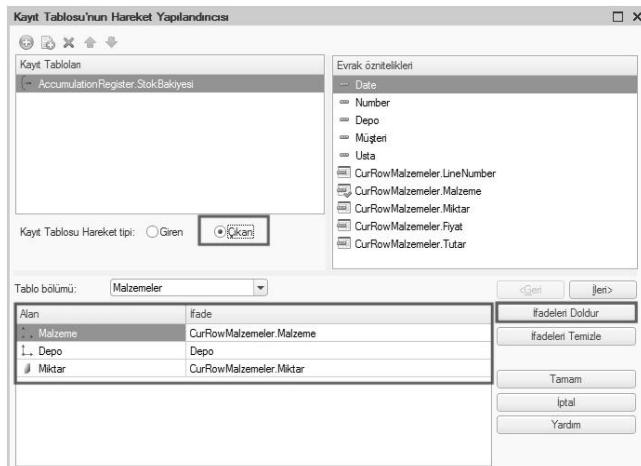
Hareket yapılandırıcı butona tıklayalım.

Ekrana gelen hareket yapılandırıcı penceresinde *Kayıt tablo hareket tipi* **Çıkan** olarak işaretleyelim. Çünkü hizmet faturası stoktan malzeme çıkışlarını kaydedecek. Kayıt tablo adının sol tarafındaki simge “—“ olarak değişir.

Tablo bölümü alanında evrakin tablo bölümünü seçelim – *Malzemeler*.

İfadeleri doldur butonuna tıklayalım.

Sistem gerekli alanları otomatik olarak doldurur (resim 6.18).



Resim 6.18. Hareket yapılandırıcı penceresi

Tamam butonuna basalım.

Hareket yapılandırıcısının *HizmetFaturası* evrakı için oluşturduğu koda bakalım (liste 6.2).

Liste 6.2. “Posting” prosedürü

```

Procedure Posting(Cancel, Mode)
    //{{_REGISTER_REGISTERRECORDS_WIZARD
    // Bu parça yapılandırıcı tarafından oluşturuldu.
    // Yapılandırıcıyı tekrar kullanma sürecinde elle kaydedilen değişiklikler
    // kaybedilecektir!!!

    // kayıt tablosu StokBakiyesi Expense
    RegisterRecords.StokBakiyesi.Write = True;
    For Each CurRowMalzemeler In Malzemeler Do
        Record = RegisterRecords.StokBakiyesi.Add();
        Record.RecordType = AccumulationRecordType.Expense;
        Record.Period = Date;
        Record.Malzeme = CurRowMalzemeler.Malzeme;
        Record.Depo = Depo;
        Record.Miktar = CurRowMalzemeler.Miktar;
    EndDo;

    //}}_REGISTER_REGISTERRECORDS_WIZARD
EndProcedure

```

Bu prosedür *Alımışsalıyesi* için oluşturulmuş prosedür ile neredeyse aynıdır (Liste 6.1). Bir tek farklı olan satır şudur:

Record.RecordType = AccumulationRecordType.Expense. Yani HizmetFaturası evrakı StokBakiyesi kayıt tablosuna depodan malzemelerin çıkışını kaydedecektr.

Son olarak, *HizmetFaturası* evrak formundan yapmış olduğu *StokBakiyesi* kayıt tablosu hareketlerine ulaşabilmek için evrakin form arayüzü düzeltelim.

Bunu yapmak için HizmetFaturası evrak formunu açalım.

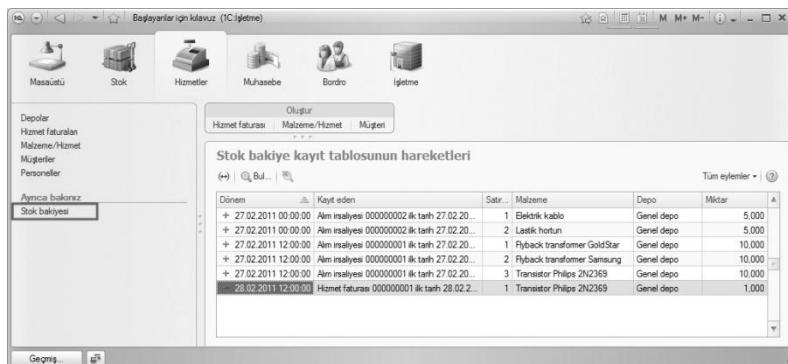
Pencerenin sol üst tarafında *Komut arayüz* sekmesine geçelim.

Araç çubuğu bölümde **Geç** grubu açalım ve *Stok bakiyesi* kayıt tablosunu açma komutunu görür olarak işaretleyelim.

1C: İşletme ortamında

1C: İşletme sistemini hata ayıklama biçiminde çalıştırıyalım ve *Hizmetler* bölümünde 1 nolu hizmet faturasını açıp Kaydet ve kapat butona tıklayalım, yani evraki tekrar kaydedelim.

Şimdi *Stok bakiyesi* komutu uygulayıp kayıt tablo listesini açalım (Resim 6.19).



Resim 6.19. "StokBakiyesi" kayıt tablo listesi

Stok bakiyesi kayıt tablosunda bir tane ekstra satır eklendi. Eklenen satır sayısı kaydedilmiş evraktaki tablo bölümünde satır sayısı ile aynıdır.

Kayıt tablodaki tüm alanlar *HizmetFaturasi* kaydetme prosedürdeki belirlendiğimiz algoritma göre doldurulmuştur.

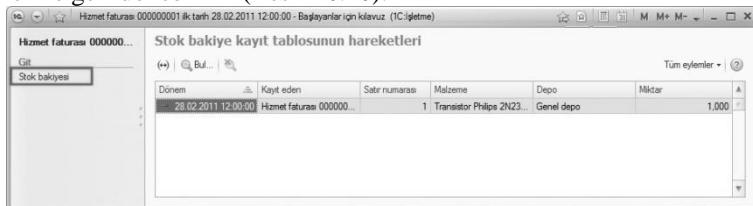
Satır başındaki “-“ simgesi kaydın hareketi Çikan olduğunu gösteriyor.

Şu anda listedeki tüm kayıtları görüyoruz. Bu listeyi evrak formundan açıp listeyi kayıt yapan evraka göre filtreleyebiliriz.

1 nolu hizmet faturasını açalım.

192 1C:İşletme 8. Uygulama geliştirme kılavuzu

Evrak formunda araç çubuğu var oldu. Araç çubuğundaki komutları kullanarak evrakin yapmış olduğu kayıt tablo hareketlerine geçip evrakin bilgilerine geri dönebiliriz (Resim 6.20).



Resim 6.20. Evrak formundan kayıt tablo kayıtlarına geçme

Böyle oluşturulmuş hizmet faturasının hareketleri tam olarak doğru değil.

Hizmet faturasında, Alım irsaliyesinden farklı olarak sadece sarf edilen mallar değil, verilen hizmetler de kaydedilir. Ve Stok bakiyesi kayıt tablosuna sarf edilen hizmetler ile ilgili bilgi kaydedilecek ki bu doğru değil.

Şu anda hiçbir şey yapmadan olduğu gibi bırakalım. Sayımlar konfigürasyon nesnelerini öğreneceğimiz zaman evrak kaydetme prosedürüne gerekli değişiklikleri ekleyeceğiz.

"Teori". Koleksiyonla çalışma yöntemleri

Alımİrsaliyesi ve HizmetFaturası evrakları için kayıt hareketleri oluşturduğumuzda tablo bölmeleri için döngü oluşturmuştuk. Bu döngüde bir kaynak kodu nesnesi – koleksiyon ile karşılaşmıştır.

Kaynak kodunun çoğu nesneleri koleksiyon olarak hesaplanmaktadır. Koleksiyonun asıl tanımı – kaynak kodu nesnelerinin setidir. Bütün koleksiyonlar için genel çalışma prensipleri mevcuttur.

İlk olarak şunu belirtmek gerekmek: koleksiyonun her bir nesnesine erişebilmek için koleksiyonun öğelerini döngü içerisinde tanımlamak gerekmektedir. Bunun için For Each ... In ... Do ... yapısı kullanılır (List. 6.3).

List 6.3 Koleksiyon öğelerini döngüde toplama

```
For Each TabloBölümüSatırı In TabloBölümü Do;
    Message(TabloBölümüSatırı.Hizmet);
EndDo;
```

Bu örnekte TabloBölümü – konfigürasyon nesnesinin tablo bölümünün satırlarının koleksiyondur. Döngü her tekrarlandığında TabloBölümüSatırı değişkeni ilgili koleksiyonunu satırının verilerini içerecektir.

1. Kaynak kodunda belli bir ad altında koleksiyonlar mevcuttur. Bu koleksiyonların öğeleri kendi aralarında benzersiz ad altındadırlar. Yani ilgili öğenin adına göre çağrı gerçekleştirmek mümkündür (List. 6.4).

List 6.4 Koleksiyon öğesini çağrıma işlemi

```
Catalogs. Personeller;
Catalogs["Personeller"];
```

Burada Catalogs – konfigürasyonda tüm kart listeleri yöneticilerinin koleksiyonu demektir. Her bir kart liste benzersiz bir tanıma sahip olduğundan dolayı, ilgili kart listenin adını belirterek ilgili kart listeye çağrı yapmak mümkündür: Catalogs["Personeller"].

2. Koleksiyon öğelerini kişiselleştirmeye gerek yoksa (koleksiyon adı mevcut değilse), o zaman ilgili öğeye dizin numarasını kullanarak başvurmak mümkündür (koleksiyonun birinci öğesinin dizini – sıfır) List.6.5.

List 6.5 Koleksiyon öğesini dizin kullanarak çağrıma işlemi

```
TabloBölümü[0];
```

Bu örnekte **TabloBölümü** – konfigürasyon nesnesinin tablo bölümü satırlarının koleksiyonu. Dizini – 0 göstererek ilk satırına başvuruyoruz.

Şunu da belirtmek gerekiyor ki, hem dizinine göre, hem de adına göre çağrı gerçekleştirilebilen koleksiyonlar mevcuttur. Örneğin, sütunlar koleksiyonunda adına göre çağrı gerçekleştirilebildiği gibi, dizinine göre de çağrı gerçekleştirilebilir.

Sorular

- ✓ *Birikim kayıt tablosu konfigürasyon nesnesi ne için kullanılır?*
- ✓ *Gerekli bilgiler diğer konfigürasyon nesnelerinde bulunmasına rağmen birikim kayıt tabloları niye kullanmak gerekiyor?*
- ✓ *Kayıt tablosunun boyutları, kaynaklar ve öznitelikler ne için kullanılır?*
- ✓ *Kayıt tablo hareketleri nedir ve kayıt eden evrak ne için kullanılır?*
- ✓ *Yeni birikim kayıt tablosu nasıl oluşturulabilir ve özellikleri nasıl tanımlanabilir?*
- ✓ *Hareket yapılandırıcısını kullanarak evrakin kayıt hareketleri nasıl oluşturulabilir?*

DERS 7

Basit Rapor

Süre

Dersin tahmin süresi – 25 dakika

Rapor nedir	196
Rapor ekleme	196
Maket	198
Veri inşa edilme şeması	198
Veri seti	198
Sorgu metni	199
Rapor ayarları	202
Sorular	206

Bu derste *Rapor* konfigürasyon nesnesi ile tanışacağız. Rapor ne için kullandığını öğreneceksiniz, işletmemizin stok durumu ve malzeme hareketlerini yansıtacak bir rapor oluşturacaksınız.

Bu bölüm sadece rapor oluşturma mekanizması hakkında genel bilgi vermek için amaçlanmaktadır. Bu yüzden bu bölümde, her rapor tasarımda kullanılan veri inşa edilme yapısının detaylarına girmeyeceğiz. Daha detaylı bu soru “Raporlar” bölümünde anlatılacaktır.

Rapor nedir

Rapor, kullanıcının istediği bilgileri görsel şeklinde elde etme algoritmalarını tanımlamak için kullanılan bir konfigürasyon nesnesidir. Kullanıcıya sunmak için verileri hazırlama algoritmalar görsel geliştirme araçları veya kaynak kodunu kullanarak tanımlanabilmektedir. Gerçek hayatta, Rapor konfigürasyon nesnesini karşılığı çeşitli raporlar, ekstreler, sonuç tabloları, grafikler vs.

Rapor ekleme

Tasarımcı ortamında

Sonuç veriyi elde edebilmek için her şeyimiz hazırır. Bu yüzden stoka malzemelerin giriş, çıkış ve güncel bakiye bilgilerini gösterecek rapor oluşturmaya başlayalım (Resim 7.1).

Veri parametreleri:					
Begin of period: 28.02.2011 00:00:00					
End of period: 24.04.2011 00:00:00					
Depo	Malzeme	Miktar Dönem başı bakiye	Miktar Giren	Miktar Çıkan	Miktar Dönem sonu bakiye
Genel depo	Flyback transformer Samsung	10.000			10.000
Genel depo	Flyback transformer GoldStar	10.000			10.000
Genel depo	Transistor Philips 2N2369	10.000			10.000
Genel depo	Lastik hortum	5.000	1.000		4.000
Genel depo	Elektrik kablo	5.000			5.000

Resim 7.1. Rapor sonucu

Bu örnekle, bir satır kod yazmadan sadece görsel geliştirme araçlarını kullanarak raporu hızlı ve kolay bir şekilde nasıl geliştirilebildiğini göstereceğiz.

Konfigürasyonumuzu açıp yeni *Rapor* konfigürasyon nesnesini ekleyelim.

Bunu yapmak için konfigürasyon ağacında *Rapor* grubunda imleci yerleştirerek konfigürasyon ağaç penceresinin araç çubuğu bulunan **Ekle** butona basmak yeterlidir (Resim 7.2).

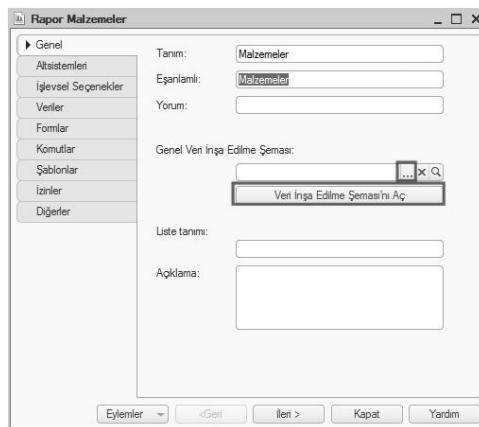


Resim 7.2. Yeni Rapor konfigürasyon nesnesini oluşturma

Ekrana gelen konfigürasyon nesne düzeltme penceresindeki *Genel* sekmesinde raporun adı tanımlayalım – **Malzemeler**.

Uygulama arayüzünde raporu ifade edecek başka özelligi belirlemeyelim. Her yerde konfigürasyon nesnesini eş anlamlı kullanılcaktır.

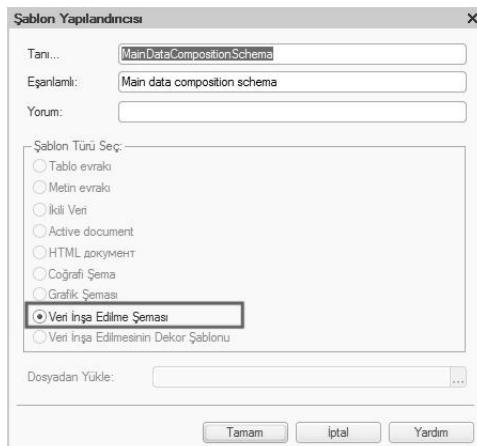
Her rapor oluşturma temelini tasarlayalım – **Veri inşa edilme şeması**. Bunu yapmak için *Veri inşa edilme şemasını aç* butonuna veya büyütme simgesini olan açma butonuna basmak gereklidir (Resim 7.4).



Resim 7.3. Raporun veri inşa edilme şemasını oluşturma

Maket

Oluşturduğumuz raporun daha veri inşa edilme şeması bulunmadığından, platform yeni veri inşa edilme şemasını oluşturmayı önerir. Konfigürasyon açısından veri inşa edilme şeması bir makettir, bu yüzden bir maket yapılandırıcı ekrana gelecek ve maket yapılandırıcısında sadece **Veri inşa edilme şeması** seçeneği ulaşılabilir olacaktır (Resim 7.4).



Resim 7.4. Raporun veri inşa edilme şemasını oluşturma

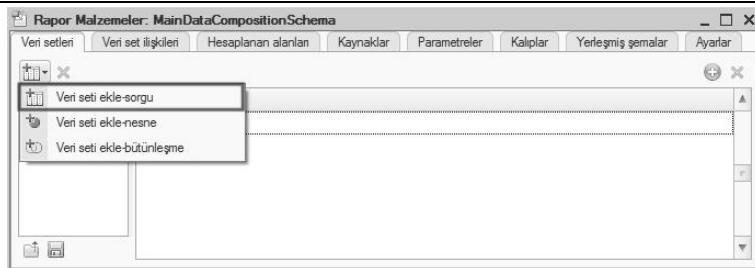
Veri inşa edilme şeması

Platform, veri inşa edilme şemasını içeren yeni maket oluşturacak ve veri inşa edilme şema yapılandırıcısını ekrana getirir.

Yapılandırıcı raporları görsel şeklinde tasarlamak için birçok araçlara sahiptir, fakat biz sadece en basit olanaklarından faydalananacağız.

Veri seti

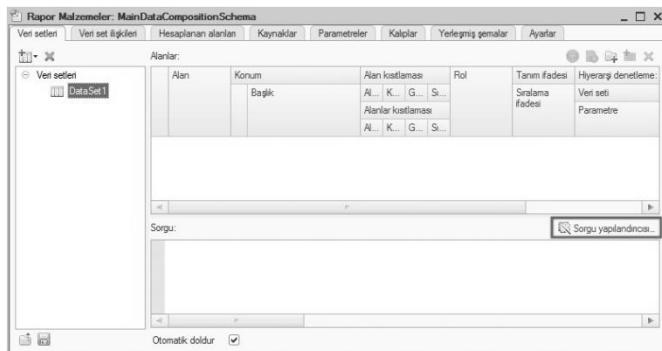
Yeni *veri seti – sorğu* ekleyelim. Bunu yapmak için **Ekle** butonuna basıp Veri seti açılır menüsünü çağıralım (Resim 7.5).



Resim 7.5. Veri inşa edilme şema yapılandırıcısında yeni veri seti ekleme

Sorgu metni

Sorgu metni oluşturmak için sorgu yapılandırıcısını çalıştırıralım – **Sorgu yapılandırıcı** butonuna tıklayalım (Resim 7.6).

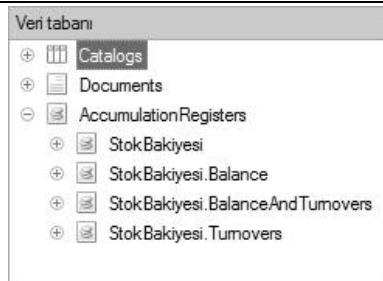


Resim 7.6. Sorgu yapılandırıcısını çağrıma

Sorgu yapılandırıcı – Geliştiricilerin, sorguları görsel şeklinde oluşturabilmeleri için geliştirilmiş olduğu bir araçtır. Sorgu yapılandırıcısını kullanarak, sorgular konusunda bilgisi olmayan kullanıcı bile, söz dizim olarak doğru olan bir sorguyu oluşturabilir.

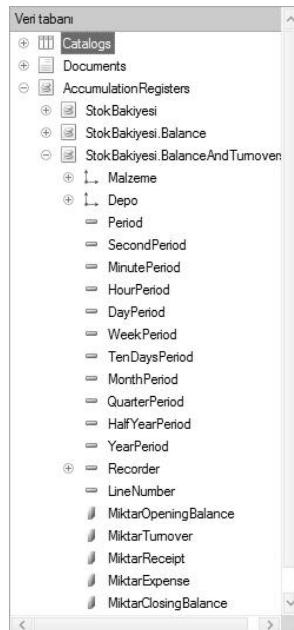
Veritabanı listesinde sorgu oluşturmak için tablolar bulunur. Bu tabloların bilgilerini kullanarak sorgu oluşturabiliriz.

AccumulationRegisters grubu açıldığında StokBakiyesi birikim tablosu hariç sistem tarafından otomatik olarak oluşturulan daha birkaç tane “*sanal tablo*” bulunmaktadır (Resim 7.7).



Resim 7.7. Sorgu oluşturma tablosu

Bunlar platform tarafından, kayıt tablosu için oluşturulan ve farklı raporları oluşturmak için kullanılan sanal tablolardır. Biz raporumuzda güncel bakiye hariç malzemelerin giriş ve çıkış bilgilerini görmek istediğimiz için **StokBakiyesi.BalanceAndTurnovers** sanal tablo gereklidir. Açılmış onu (Resim 7.8).



Resim 7.8. "StokBakiyesi.BalanceAndTurnovers" tablosu

Gördüğümüz gibi bu tablo *StokBakiyesi* kayıt tablosunun boyutları – *Malzeme*, *Depo*, ilk tarih ve son tarih bakiye bilgileri ve bunun yanı sıra *StokBakiyesi* kayıt tablosunun her boyutu için giriş, çıkış ve akış bilgilerini içerir.

Çift tıklayarak sırayla bize gerekli olan alanları seçelim.

İlk önce *Depo* ve *Malzeme* seçelim.

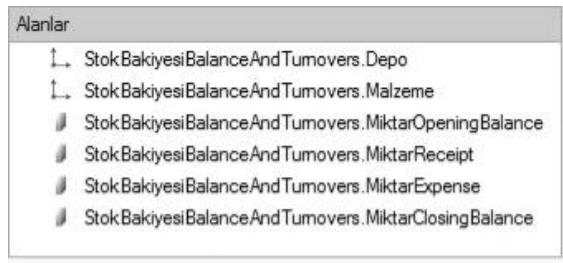
Ondan sonra *MiktarOpeningBalance*, *MiktarReceipt*, *MiktarExpense*.

Son olarak *MiktarClosingBalance* seçelim.

Not

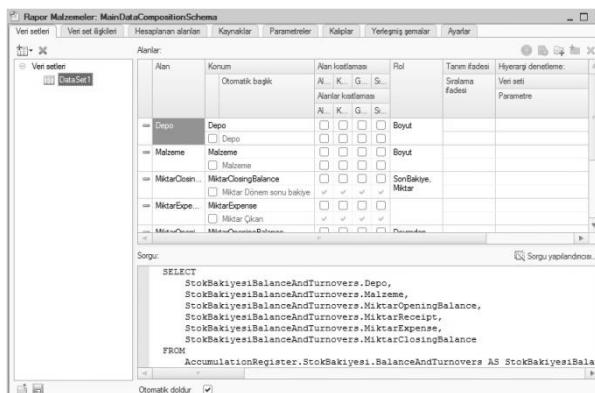
Seçilen öğeleri bir listeden diğer listeye çift tıklayarak veya fare ile sürükleyerek taşınabilir. Ya da , , , butonlar kullanılabilir.

Sonuç olarak Alanlar penceresi aşağıdaki gibi doldurulmalıdır (Resim 7.9).



Resim 7.9. Seçilen alanlar

Tamam butonuna basalım ve veri inşa edilme şema yapılandırıcısına geri dönelim (Resim 7.10).



Resim 7.10. Veri inşa edilme şeması

Sorgu yapılandırıcısı yardımıyla oluşturulan sorgu metni *Sorgu* alanında yer aldı.

Bu alan metin düzeltme alanıdır, dolayısıyla gerektiğinde oluşturulan sorgu el ile de düzeltilebilir. Bunun dışında sorgu yapılandırıcısı tekrar çağırılabilir ve var olan sorgu düzeltilebilir.

Bilgisi daha kolay anlayabilmek için biz şimdi sorgu metin detaylarına girmeyeceğiz. Bu rapor için olduğu gibi çoğu rapor için sorgu metnini

incelemek ve el ile düzeltmek gerekmeyez. İllerdeki “Raporlar” bölümünde detaylı olarak sorğu metni anlatılacaktır.

Yapilandırıcı penceresinin üst kısmında olan veri inşa edilme alan listesini platform otomatik olarak doldurdu. Bu liste geçerli veri setinde ulaşabilir alanları içerir. Bu alanı sistemi otomatik doldurma parametreleri bizim için uygun olduğu için herhangi bir düzeltme yapmamıza gerek yok.

Rapor için veri nasıl alacağını tanımladık. Fakat rapor için standart ayarları oluşturmadığımız sürece raporda hiçbir şey göremeyeceğiz. Bu yüzden veritabanı bilgilerini detaylı görebilmek için en basit ayarları tanımlayalım.

Bizim örneğimizde, satır şeklinde veritabanına kaydetme sırasıyla *StokBakiyesi* sanal tablosunun kayıtları.

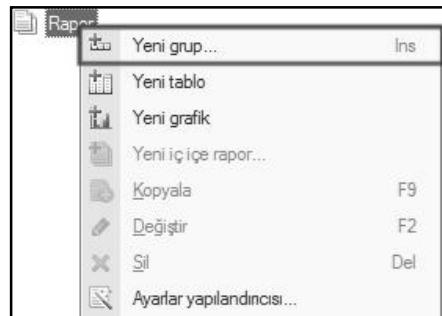
Rapor ayarları

Ayarlar sekmesine geçelim. Sağ üst pencerede raporumuzun hiyerarşik yapı bulunmaktadır.

Yeni öğe eklemek için yapı ağacının kökünde *Rapor* öğesini seçip sağ tuş ile bağlam menüsünü çağıralım. Araç çubuğundaki **Ekle** butonu ve Ins tuşu da kullanılabilir.

Rapora grup ekleyelim (bağlam menü – *Yeni grup*). Grup alanı belirtmeden Tamam butona basalım.

Böylece raporda *detaylı kayıtlar* (sorgu ile elde edilen ve toplamları olmayan bilgi) gösterileceğini belirttik (Resim 7.11).



Resim 7.11. Rapora yeni grup eklemesi

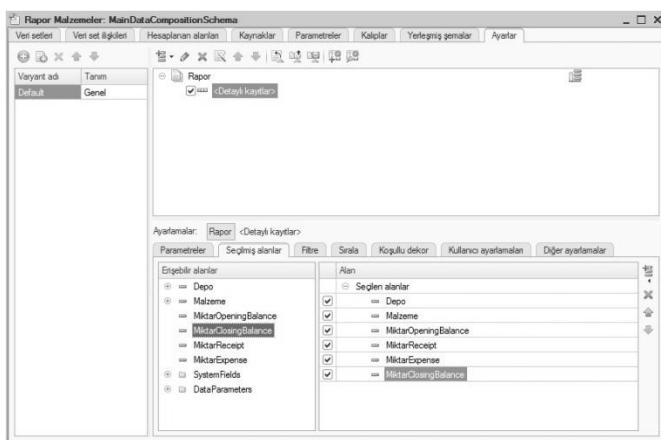
Rapor yapısında *Detaylı kayıtlar* grubu eklenmiştir.

Şimdi raporun sonu alanlarını ayarlayalım.

Bunun yapmak için pencerenin alt kısmında bulunan *Seçilmiş veri* sekmesine geçip *erişebilir alanlar* listesinden fare ile aşağıdaki alanları sağıdaki pencereye sürükleylelim sürükleylelim.

- Depo,
- Malzeme,
- MiktarOpeningBalance
- MiktarReceipt
- MiktarExpense
- MiktarClosingBalance

Sonuç olarak rapor ayarlama penceresi aşağıdaki resimde gibi görünmelidir.

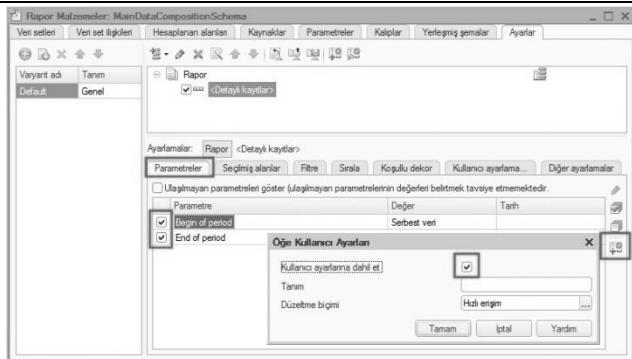


Resim 7.12. Rapor ayarlama penceresi

Ondan sonra Parametreler sekmesine geçip Begin of period ve End of period rapor parametreleri kullanıcı ayarlarına dahil olacağını belirleyelim ve bu ayarlar direkt formunda bulunacağını belirleyelim (yani bu parametreler “hızlı” ayarlar olarak kullanılacak).

İki parametrenin raporda kullanılacağını belirleyelim – birinci sütunda tık atalım.

Ondan sonra her parametre aktif edip **Kullanıcı ayarları öğe özelliklerini** butonuna basalım ve **Kullanıcı ayarlarına dahil et** alanı işaretleyelim (Resim 7.13).



Resim 7.13. Raporun parametrelerini belirtme

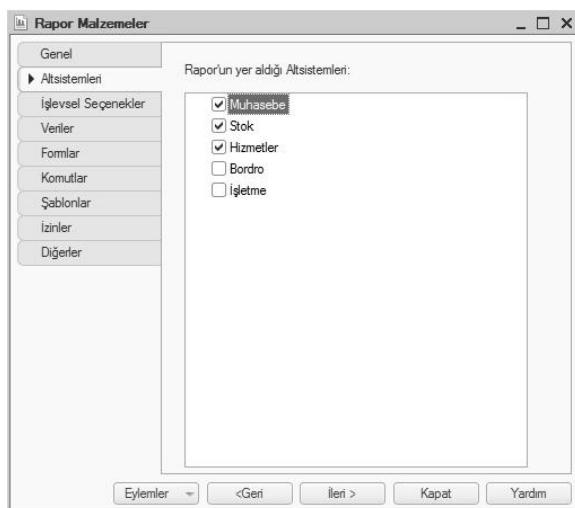
Böylece rapor oluşturmadan önce kullanıcı raporlanacak dönemi belirleyebilir. Parametreler ve kullanıcı ayarları hakkında daha detaylı bilgiler "Raporlar" bölümünde verilecektir.

Son olarak raporun yansıtılacağı alt sistemlerini belirleyelim.

Veri inşa edilme şema yapılandırıcısını kapatıp *Malzemeler* raporu konfigürasyon nesnesinin düzeltme penceresinde *Alt sistemler* sekmesine geçelim.

Alt sistem listesinde *Stok*, *Hizmetler* ve *Muhasebe* alt sistemlerini işaretleyelim.

Böylece rapor açma komutu belirlediğimiz üç bölümde gösterilir (Resim 7.14).

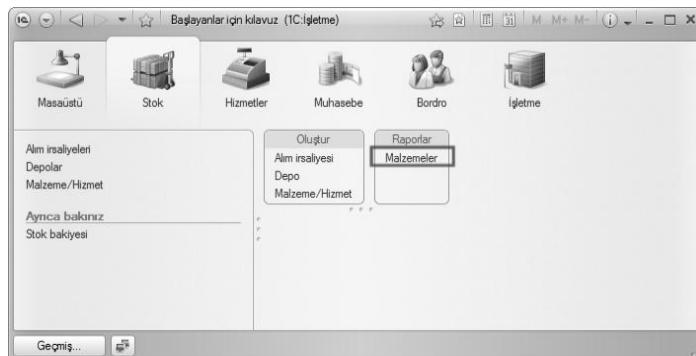


Resim 7.14. Raporun gösterileceği alt sistemlerini belirletme

1C: İşletme ortamında

1C:İşletme sistemini hata ayıklama biçiminde çalışıralım.

Açılan 1C:İşletme penceresinde *Stok*, *Hizmetler* ve *Muhasebe* bölümünün eylem çubuklarında *Malzemeler* raporu görüntülemek için *Malzemeler* komutu ortaya çıktı. Onu uygulayalım.



Resim 7.15. "Malzemeler" raporu oluşturma komutu

Ekranımız sistemin otomatik olarak oluşturulmuş olan rapor formu gelir.

Begin of period (İlk tarih) ve End of period (Son tarih) değerlerini doldurduktan sonra *Oluştur* butonuna basalım (Resim 7.16).

Malzemeler					
Rapor bigimi:	Genel				
<input type="button" value="Oluştur"/>		<input type="button" value="Ayarlar..."/>			
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Begin of period 01.02.2011 00:00:00			
<input checked="" type="checkbox"/>	<input type="checkbox"/>	End of period 07.03.2011 00:00:00			
Veri parametreleri		Begin of period = 01.02.2011 00:00:00 End of period = 07.03.2011 00:00:00			
Depo	Malzeme	Miktar Dönem başı bakiye	Miktar Giren	Miktar Çıkan	Miktar Dönem sonu bakiye
Genel depo	Flyback transformer Samsung	10,000			10,000
Genel depo	Flyback transformer GoldStar	10,000			10,000
Genel depo	Transistor Philips 2N2369	10,000		1,000	9,000
Genel depo	Lastikhortun		5,000		5,000
Genel depo	Elektrik kablo		5,000		5,000

Resim 7.16. "Malzemeler" raporu

Gördüğünüz gibi rapor malzemelerin giriş/çıkışları ile ilgili oldukça detaylı bir bilgi sunar.

Sorular

- ✓ *Rapor konfigürasyon nesnesi ne için kullanılır?*
- ✓ *Veri inşa edilme şemasının kullanarak rapor nasıl oluşturulabilir?*
- ✓ *Uygulama çözüm arayüzünde rapor nasıl gösterilir?*

DERS 8

Şablonlar. Form ve şablon tasarımlı

Süre

Dersin tahmin süresi – 1 saat 10 dakika

Şablon nedir	208
Matbu form şablonu	208
Şablon düzeltme	213
Form düzeltme	218
Sorular	222

Bu derste *Şablon* konfigürasyon nesnesi ile tanışacağız. Bu nesnenin yapısını öğrenip evrakin çıktısını alabilmek için bir şablon geliştireceğiz.

Şablon nedir

Şablon, verilerin farklı görsel formların tanımlamak için kullanılan konfigürasyon nesnesidir. Şablon'daki veriler belli bir konfigürasyon nesnesi için uygulanabildiği gibi komple uygulama çözümü için de uygulanabilir.

Şablon tablo veya metin evrakını, ikili veriyi, HTML-evrakını veya Active Document, grafik veya coğrafik şemasını, veri inşa edilme şemasını veya veri inşa edilme şemasının dekor şablonunu içerebilir.

Şablonlar ayrı olarak oluşturulabildiği gibi (genel şablonlar) belli bir konfigürasyon nesnesine ait olabilirler.

Belli bir konfigürasyon nesnesine ait olan ve tablo evrakını içeren şablonun genel amacı, bu nesnenin matbu formunu tanımlamaktır. Matbu formu tasarlamak için adlandırılmalı matbu form bölümlerini oluşturmak gereklidir. Sonra bu bölümleri bir araya getirerek matbu form tasarılanır.

Böülümlere veri doldurması ve nihai form oluşturulması kaynak kod ile yapılır. Matbu form farklı grafik nesnelerini içerebilir; resimler, OLE-objeleri, grafikler vs.

Tasarımcıda, şablonu “manüel” olarak geliştirme araçları dışında, şablonu geliştirmesini kolaylaştıracak özel tasarlama aracı da mevcuttur – *Yazdırma yapılandırıcısı*. Yazdırma yapılandırıcısı “manüel” olarak yapılan işlemlerin çoğu kendisi yapar.

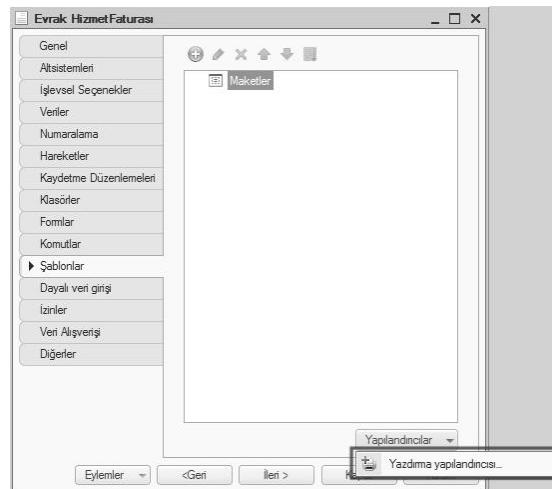
Matbu form şablonu

Tasarımcı ortamında

Hizmet faturası için matbu formu oluşturmak gereklidir.

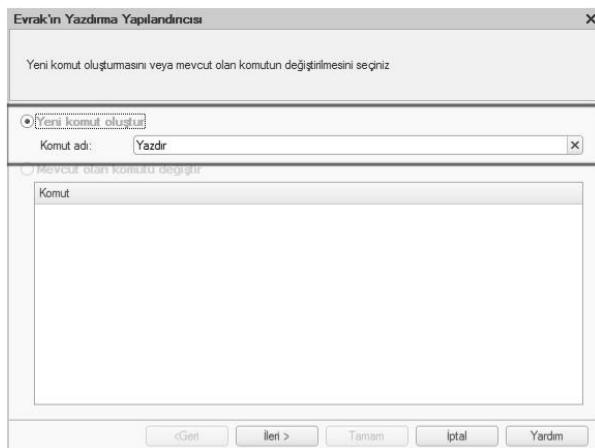
Tasarımcıda *Hizmet faturası* evrakı konfigürasyon nesnesinin düzeltme penceresini açalım.

Şablonlar sekmesine geçelim ve yazdırma yapılandırıcısını çalıştırıralım (Resim 8.1).



Resim 8.1. Yazdırma yapılandırıcısını çalıştırma

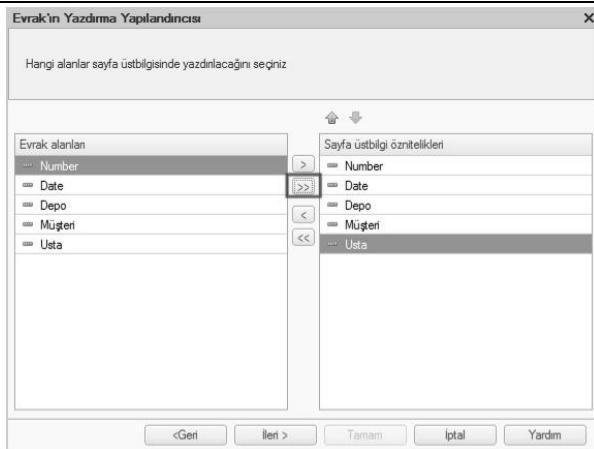
Ekrana gelen yazdırma yapılandırıcısının ilk aşamasında, evrakin matbu formunu oluşturmak için yeni *Yazdır* komutu oluşturacağımı belirleyelim (Resim 8.2).



Resim 8.2. Yazdırma yapılandırıcısı. Aşama 1

İleri tıklayalım.

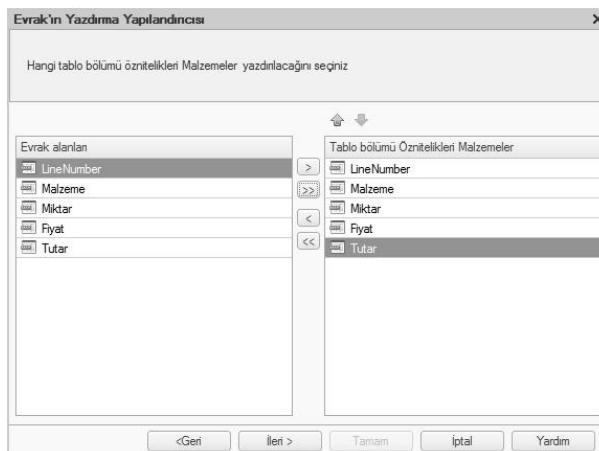
İkinci aşamada **>>** butonu tıklayarak evrakin tüm öznitelikleri matbu formun başlığında bulunacağını belirleyelim (Resim 8.3).



Resim 8.3. Yazdırma yapılandırıcısı. Aşama 2

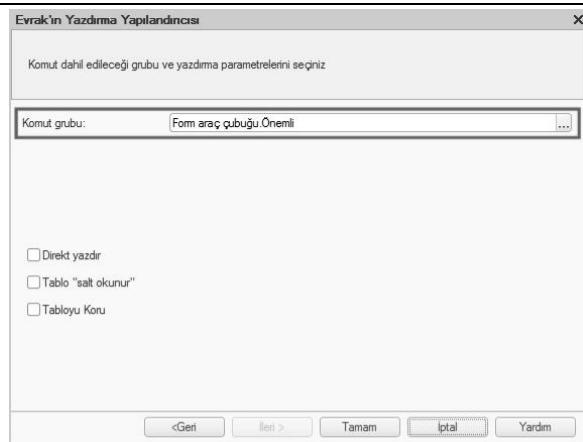
İleri tıklayalım.

Üçüncü aşamada aynı şekilde, evrakın tablo bölümünde bulunan tüm öznitelikler matbu formunda gösterileceğini belirtelim (Resim 8.4).



Resim 8.4. Yazdırma yapılandırıcısı. Aşama 3.

Dördüncü aşamada yazdırma yapılandırıcısı matbu formunun alt bilgisi için alan seçme penceresini açar. Herhangi bir değişiklik yapmadan **İleri** butona basarak bir sonraki aşamaya geçiyoruz (matbu formumuzda alt bilgisi olmayacağı) (Resim 8.5).



Resim 8.5. Yazdırma yapılandırıcısı.Aşama 5

Burada da herhangi bir değişiklik yapmayacağız. Böylece, matbu form oluşturma prosedürü araç çubuğundaki *Önemli* bölümünde konumlandırılacağını kabul edelim.

Tamam tıklayalım.

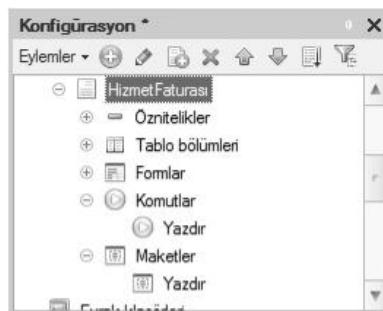
Tasarımcıda *Yazdır* komutunun modülü, *HizmetFaturası* evrakının yönetici modülü ve evrakın şablonu açılacaktır (Resim 8.6).



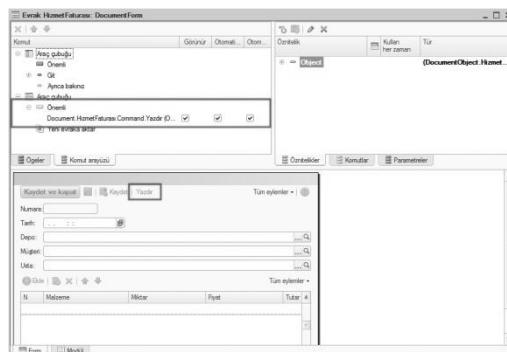
Resim 8.6. "Hizmet faturası" evrakının şablonu

Geliştirici sıfırdan da şablonu oluşturup yazdırılması için komutu ve evrak formundaki butonu oluşturabilir, fakat bizim örneğimizde bu tüm işlemler yazdırma yapılandırıcısı ile gerçekleşmiştir.

- Yazdır adı ile HizmetFaturası evrakının matbu formu oluşturulmuştur (Resime 8.6).
- *HizmetFaturası* evrakını *Yazdır* komutu oluşturulmuştur. Bu komut modülüne serverde gerçekleştirilecek olan evrak yazdırma prosedürü çağrıran işleyici yerleştirilmiştir. Evrak yazdırma prosedürü kendisi, *HizmetFaturası* evrakının yönetici modülüne konumlandırılmıştır (Resim 8.7).
- *HizmetFaturası* evrak formunun araç çubuğunda evrakin matbu formunu oluşturmak için *Yazdır* komutu yerleştirilmiştir (Resim 8.8).



Resim 8.7. Konfigürasyon ağacında “Hizmet faturası” evrakını yapısı



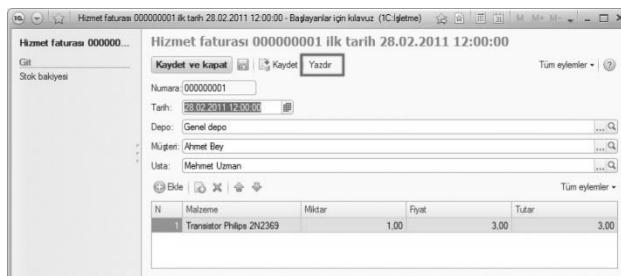
Resim 8.8. “Hizmet faturası” evrakının formu

Bu sırada Yazdır komutu belli bir forma değil genel olarak *HizmetFaturası* evraka ait olduğu için bu komut evrak için oluşturululan herhangi bir formda yerleştirilebilir.

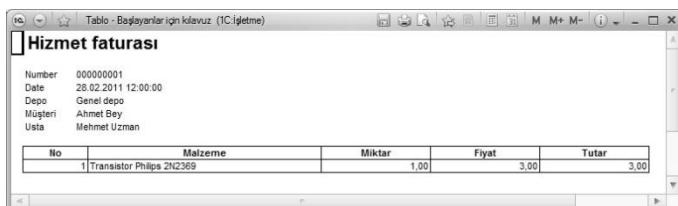
1C: İşletme ortamında

1C: İşletme sistemini hata ayıklama biçiminde oluşturalım ve 1 nolu Hizmet fatura evrakını açalım.

Evrak formunu araç çubuğunda Yazdır butonu oluşturulmuştu (Resim 8.9).



Resim 8.9. Yeni “Yazdır” komutu içeren “Hizmet fatura” evrakının formu
Komuta basıp evrakımızın matbu formu görelim (Resim 8.10).



Resim 8.10. “Hizmet fatura” evrakının matbu formu

Gördüğünüz gibi yazdırma yapılandırıcısı gayet güzel bir form oluşturmuştur. Bir tek eksik olan şey – evrakın toplam tutarıdır.

Bir sonraki paragrafta evrakın toplam tutarının ekleme örneği ile evrak şablonlarını ve konfigürasyon nesne formlarını nasıl değiştireceğini göstereceğiz.

Şablon düzeltme

Tasarımcı ortamında

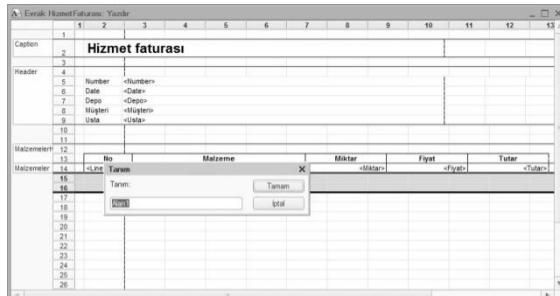
İlk önce *HizmetFatura* matbu formuna toplam tutarını ekleyelim.

Tasarımcı açalım, konfigürasyon ağacında *HizmetFatura* grubu açıp *Yazdır* şablonu üzerinde çift tıklayalım.

Gördüğünüz gibi şablon, belli bir sırayla sonuç formuna eklenen adlandırılmış bölümlerden ibarettir.

Yani, sol tarafta gördüğümüz adlandırılmış bölümler yazdırma yapılandırıcısı tarafından oluşturulmuştur. Fakat geliştirici bölgeler ile ilgili ekleme, silme veya yeniden adlandırma vs. işlemleri uygulayabilir.

Evrakin toplam tutarını göstermek için yeni bölüm ekleyelim. Tablo bölümü altında iki tane boş satır seçelim ve pencerenin genel menüsünden **Tablo – Adlar – Ad gir** menüsü kullanalım (Resim 8.11).



Resim 8.11. Evrak toplamı girmek için yeni bölüm oluşturmacı

Yeni bölümü **Toplam** olarak adlandıralım.

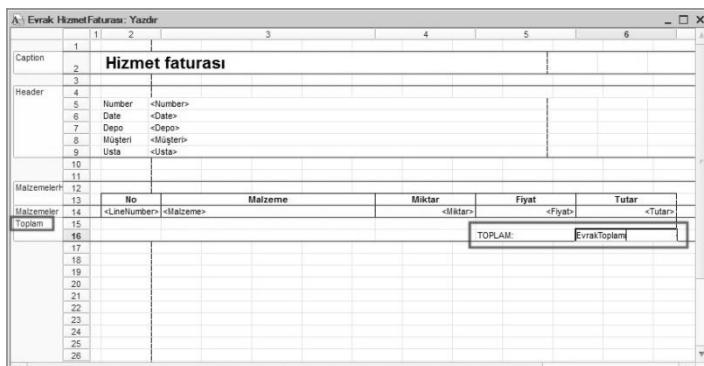
Oluşturduğumuz satırların formatının evrakin başlık ve tablo bölümü formatı ile uyumlu olması sütun genişliğini değiştirelim.

Bunu yapmak için sol tarafta oluşturduğumuz bölümü tek tıklayalım ve ikinci sütunun denişliğini tablo *MalzemelerHeader* bölümündeki “No” hücresinin genişliği ile aynı yapalım. Fare bırakalım.

Platform yeni satır biçimini oluşturmasını önerir. Kabul edelim.

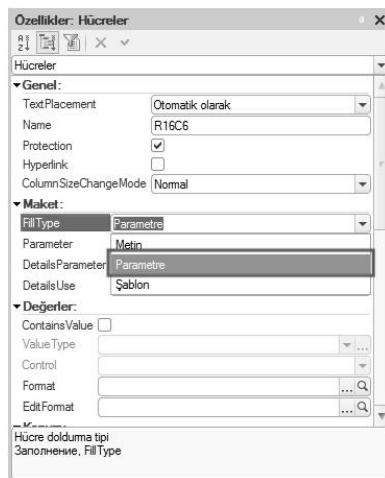
Benzer eylemleri oluşturulan *Toplam* bölümünün 3,4,5 ve 6 sütunları için uygulayalım.

Oluşturulan bölümünün *Fiyat* sütunda **TOPLAM:** yazalım ve *Tutar* sütunda **EvrakToplamı** yazalım (Resim 8.12).



Resim 8.12. Evrak toplamını göstermek için hücre oluşturmacı

EvrakToplamı hücresinin özelliklerine gelip (sağ tuş – Özellikler) **FillType** özelliğini **Parametre** olarak değiştirerek bu hücrede metin değil parametre konumlandırılacağını belirtelim (Resim 8.13).



Resim 8.13. “EvrakToplamı” hüresinin özellikleri

Tasarlanmış olduğumuz tablo evrakının her hücresi metin, parametre veya şablonu içerebilir.

Hücredeki **Metin** ekranda gösterilecektir.

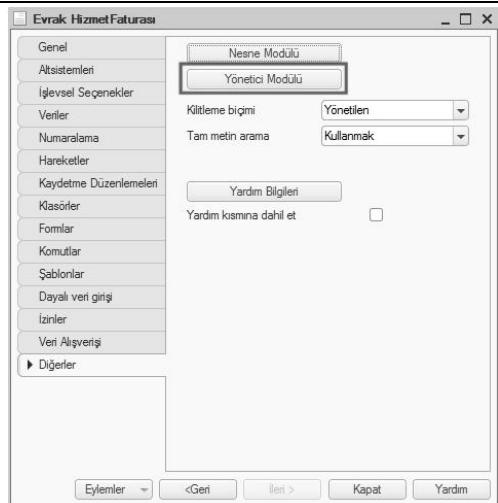
Parametre, kaynak kod ile hesaplanabilir bir değer ile doldurulacaktır. Hücredeki metin bu parametrenin adıdır.

Şablon bir satırdır ve bu satır içinde metin yanı sıra parametre değerleri eklenebilir.

Yani *EvrakToplamı* hüresinde kaynak kod ile hesaplanacak bir değer gelir.

HizmetFatura evrakının **Yönetici modülünü** açalım.

Bunu yapmak için *SatışFatura* evrakı konfigürasyon nesnesinin düzeltme penceresinin *Digerler* sekmesine geçelim ve *Yönetici Modülü* butona basalım (Resim 8.14).



Resim 8.14. “HizmetFatura” evrakının yönetici modülünü açma
Yazdır prosedürü bulup aşağıdaki gibi onu düzeltelim (eklenen satırlar
kalın olarak işaretlendi) Liste 8.1.

Liste 8.1 Evrak yazdırma (parça)

```

AreaCaption = Template.GetArea("Caption");
Header = Template.GetArea("Header");
AreaMalzemelerHeader = Template.GetArea("MalzemelerHeader");
AreaMalzemeler = Template.GetArea("Malzemeler");
AreaToplam = Template.GetArea("Toplam");
Spreadsheet.Clear();

InsertPageBreak = False;
While Selection.Next() Do
    If InsertPageBreak Then
        Spreadsheet.PutHorizontalPageBreak();
    EndIf;

    Spreadsheet.Put(AreaCaption);

    Header.Parameters.Fill(Selection);
    Spreadsheet.Put(Header, Selection.Level());

    Spreadsheet.Put(AreaMalzemelerHeader);
    SelectionMalzemeler = Selection.Malzemeler.Choose();
    ToplamTutari = 0;
    While SelectionMalzemeler.Next() Do

        AreaMalzemeler.Parameters.Fill(SelectionMalzemeler);
        Spreadsheet.Put(AreaMalzemeler,
        SelectionMalzemeler.Level());
        ToplamTutari = ToplamTutari +
SelectionMalzemeler.Tutar;
        EndDo;

        AreaToplam.Parameters.EvrakToplam =
ToplamTutari;
        Spreadsheet.Put (AreaToplam);

        InsertPageBreak = True;
    EndDo;

```

Eklenen parçanın mantığı basit. HizmetFaturası evrakinin şablonuna adı ile başvuruyoruz – **Şablon**.

Onun **GetArea()** metodunu kullanarak önceden eklediğimiz **Toplam** bölümü alıyoruz ve **AreaToplam** değişkene kaydediyoruz.

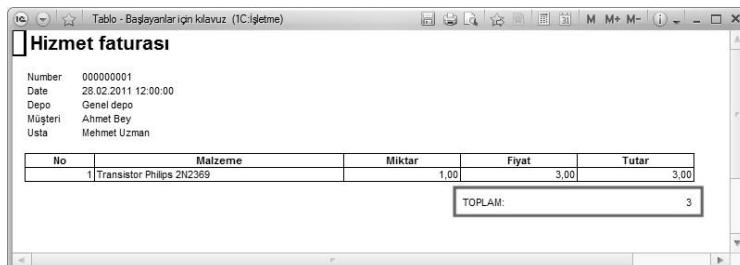
Evrakin tablo bölüm işleme döngüsünde **Tutar** sütunlarda bulunan miktarı **ToplamTutarı** değişkende biriktiriyoruz.

Ondan sonra **Toplam** bölümünde bulunan **EvrakToplamı** parametresine başvuruyoruz (*AreaToplam.Parameters.EvrakToplamı*) ve bu parametreye **ToplamTutarı** değerini belirliyoruz.

Son olarak biz sonuç bölümü ekranda gösterilecek ve yazdırılacak tablo evrakına aktarıyoruz - **Spreadsheet.Put (AreaToplam)**;

1C: İşletme ortamında

1C: İşletmeyi hata ayıklama biçiminde çalıştıralım ve yapmış olduğu değişikliklere bakalım (Resim 8.15).



Resim 8.15. "Hizmet faturası" evrakının matbu formu

Böylece, adlandırılmış bölgümleri ve şablon hücreleri oluşturarak, özelliklerini kullanarak ve kaynak kodu ile onları gösterme mantığını yöneterek her türlü matbu formlar tasarlanabilir.

Şimdi de hizmet faturası evrak formunun tam bitmiş olması için kullanıcının evrakı doldururken genel toplamını görebilmesi için evrak toplamını ekran formuna ekleyelim.

Form düzeltme

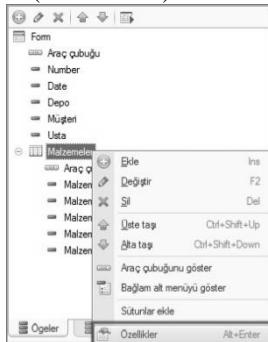
Tasarımcı ortamında

Evrakin matbu formuna evrakin toplamını aktardıktan sonra evrakin ekran formunda da evrakin genel toplamını görme gereği duyduk. Böylece evrakı doldururken hemen genel toplamını görebileceğiz.

Bunun yapmak için **HizmetFatura** evrakının evrak formuna bazı değişiklikler uygulayalım.

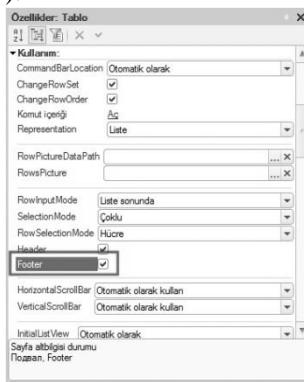
Formu düzeltmek için konfigürasyon ağacında **HizmetFatura** evrak grubundan **DocumentForm** üzerinde çift tıklayalım.

Malzemeler öğesi üzerinde çift tıklayalım veya sağ tuş menüsünden Özellikler maddesini seçelim (Resim 8.16).



Resim 8.16. Tablo bölümünün özelliklerine geçiş

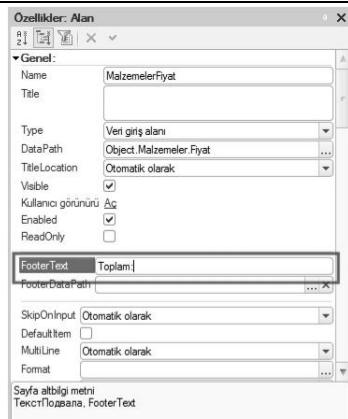
Tablo bölümünün alt bilgisinin varlığını yöneten **Footer** özelliğini işaretleyelim (Resim 8.17).



Resim 8.17. Tablo bölümünün alt bilgisini belirleme

Ondan sonra MalzemelerFiyat form öğesinin özelliklerine geçip aşağıdaki parametrelerini belirleyelim;

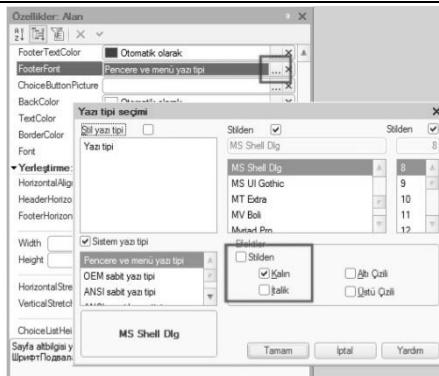
- **FooterText – Toplam:** (Resim 8.18).
- **FooterHorizontalAlign – Sağ** (Resim 8.19).
- **FooterFont** özelliğinde yazı tipini **Kalın** yapalım (Resim 8.20).



Resim 8.18. Sütunun alt bilgi özelliği



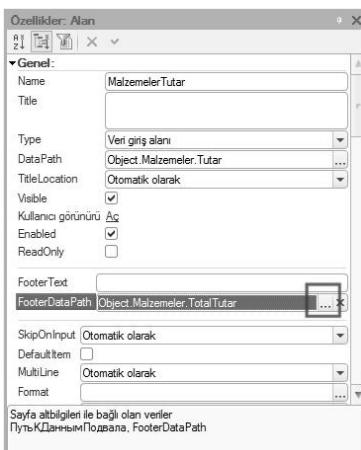
Resim 8.19. Sütunun alt bilgi özelliği



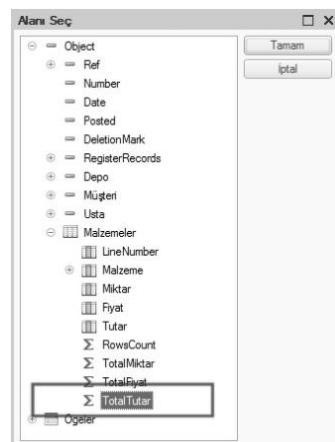
Resim 8.20. Sütunun alt bilgi özelliği

Tutar sütunun alt bilgisinde sütunun toplamının yansıtılması için sütunun özellik panosunda **FooterDataPath** özelliğinde seçim butona tıklayalım (Resim 8.21).

Öznitelik ağacının açıp **TotalTutar** öğesini seçelim (Resim 8.22).



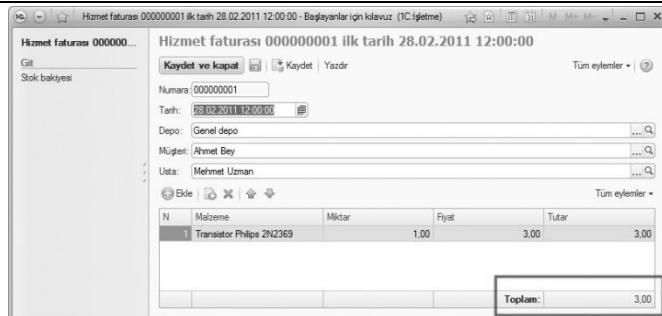
Resim 8.21. Tutar sütunun için alt bilgi değer seçimi



Resim 8.22. Tutar sütunun için alt bilgi değer seçimi

1C: İşletme ortamında

1C:İşletme’yi hata ayıklama biçiminde çalıştıralım ve 1 nolu Hizmet faturası evrakını açalım (Resim 8.23).



Resim 8.23. “Hizmet faturası” evrakı formu

Tutar sütuna göre evrakin tablo bölümünde evrakin genel toplamı hesaplandığını görüyoruz.

Formda yaptığımız bu küçük değişiklik sayesinde kullanıcı arayüz daha kullanışlı biçimde gelmiştir. Böylece, form öğe özelliklerini kullanarak, özellik değerlerini doldurarak ve formuna arayüzü değiştirerek geliştirici, formun gerekli davranışını ve dizaynı belirleyebilir.

Sorular

- ✓ *Maket konfigürasyon nesnesi ne için kullanılır?*
- ✓ *Yazdırma yapılandırıcısı nedir?*
- ✓ *Yazdırma yapılandırıcısını kullanarak rapor nasıl oluşturulabilir?*
- ✓ *Tablo evrakı nasıl değiştirilir?*
- ✓ *Tablo evrak hücrelerinin metin, parametre ve şablon tipleri arasındaki fark nedir?*
- ✓ *Kaynak kodu kullanarak tablo evrakına yeni bir alan nasıl doldurulur?*
- ✓ *Form denetimin dekoru ve oluşturduğu hareketi nasıl değiştirilebilir?*
- ✓ *Tablo sütunun toplamı nasıl gösterilir?*

DERS 9

Periyodik Bilgi Kayıt Tabloları

Süre

Dersin tahmin süresi – 50 dakika

Periyodik kayıt tablosu ne için kullanılır	224
Bilgi kayıt tablosu nedir	224
Periyodik bilgi kayıt tablosu ekleme	226
Boyutlar ve kaynaklar	227
Bilgi kayıt tablosunda kayıt ekleme	230
Evrakta malzeme/hizmet seçildiğinde otomatik olarak fiyat doldurma	231
Malzeme/hizmet fiyatını veren fonksiyon	231
Evrakta malzeme veya hizmet seçildiğinde fonksiyonu çağırma ve fiyatı doldurma	233
Sorular	236

Bu derste periyodik türü olan *Bilgi kayıt tablosu* ile tanışacağız. Bu nesne için tasarlandığını ve yapısı ne olduğunu öğreneceksiniz.

Konfigürasyonumuzda kullanacağımız bir tane periyodik bilgi kayıt tablosu oluşturacağız ve kaynak kodu kullanarak kayıt tablo bilgilerine nasıl ulaşabileceğini göstereceğiz.

Periyodik kayıt tablosu ne için kullanılır

İlk önce *Hizmet faturası* evrakımıza bakalım, bu evrakta verilen hizmet seçilir ve fiyat belirlenir.

Büyük ihtimalde, “Aspirin usta” şirketinin verilen hizmetlerinin fiyatlarını içeren bir fiyat listesi vardır. İlk bakıştan hizmet fiyatı hizmetin ayrılmaz bir parça olduğunu ve bu yüzden Malzeme/Hizmet kart listesinin özniteliği olarak eklenebileceğini düşünülebilir.

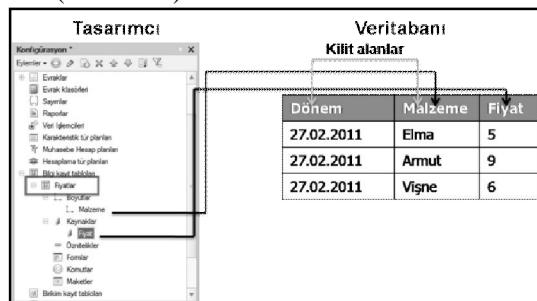
Fakat hizmet fiyat zamanla değişen bir değerdir. Önceden eklenmiş *Hizmet faturası* evrakına bir değişiklik yapmamız gerekebilir. Bu durumda doğru bir fiyat değeri alamayacağız, çünkü Malzeme/Hizmet kart listesinin özniteliğinde sadece son fiyat değeri bulunacaktır.

Bunun dışında “Aspirin usta” şirketin yöneticileri işletmenin karlılığını fiyatattaki değişikliklere göre takip etmek isterler. Bu durumda fiyatların zaman çerçevesinde takip edilmesi şarttır.

Bu yüzden hizmetleri fiyatlarını takip etmek için yeni bir konfigürasyon nesnesini kullanacağız – **Bilgi kayıt tablosu**.

Bilgi kayıt tablosu nedir

Bilgi kayıt tablosu konfigürasyon nesnesi birden fazla boyut detayları ile veri depolama yapısını tanımlamak için kullanılan bir konfigürasyon nesnesidir. Bilgi kayıt tablosu konfigürasyon nesnesine göre platform veritabanında bir tablo oluşturur ve bu tabloda boyutlar detayları ile veri depolanabilecektir (Resim 9.1).



Resim 9.1. Tasarımcıda ve veritabanında “Fiyatlar” bağımsız periyodik bilgi kayıt tablosu

Bilgi kayıt tablosunun birikim kayıt tablosuna göre farkı şudur; bilgi kayıt tablosunun her hareketi yeni kaynak değeri belirlenir, birikim kayıt tablosunun hareketi ise var olan kaynak değeri değiştirir. Bu yüzden bilgi kayıt tablosu sadece miktarsal değerleri değil, herhangi türü olan değerleri içerebilir.

Bilgi kayıt tablosunun diğer önemli özelliği, gerektiğinde verileri tarihe göre bağlayabilme olanağıdır. Böylece bilgi kayıt tablosu sadece güncel bilgileri değil, zamanla değişen verilerin geçmişini de kaydedebilir. Tarih detayı ile veri kaydetme olanağına sahip olan bilgi kayıt tablosuna ***periyodik bilgi kayıt tablosu*** denir.

Kayıt tablosunun periyodikliği aşağıdaki değerlere göre belirlenebilir;

- Saniye olarak,
- Gün olarak,
- Ay olarak,
- Üç aylık olarak,
- Yıl olarak,
- Kayıt eden olarak (kayıt eden bağılılığı kayıt türü belirlendiğinde).

Periyodik bilgi kayıt tablosu sistemin otomatik olarak eklendiği **Dönem** alanına her zaman sahiptir. Bu alanın türü *Date* ve bu alan yapılan kaydın hangi döneme ait olduğunu belirlemek için kullanılır. Kayıt tablosuna veri kaydedildiği anda platform bu alanın değerini, içinde bulunduğu dönemin başına eşitliyor.

Örneğin, ay olarak periyodikliği olan kayıt tablosuna veri kaydedildiği zaman tarih 08.04.2011 olduğunda, kayıt tablosu bu kaydın Dönemi 01.04.2011 olarak kaydeder.

Diğer kayıt tabloları olduğu gibi sistem bilgi kayıt tablosu için kayıt bensersizlik kontrolünü yapar. Diğer kayıt tabloları için kaydın anahtarı satır numarası ve kayıt yapan evraktır, fakat bilgi kayıt tablosu için biraz farklı bir mekanizma kullanılmaktadır.

Kayıdı tanımlayan *Kayıt anahtarı*, kayıt tablo kaynaklarının değerleri ve dönemidir (kayıt tablosu periyodik ise). Bilgi kayıt tablosu aynı kayıt anahtarı olan birden fazla kayıt içeremez.

Birikim kayıt tabloları ile karşılaşmaya devam edersek, bilgi kayıt tablosu depolanan veriyi düzeltme konusunda daha çok esnekliğe sahiptir. Kayıt eden bağımlılığı biçiminde kullanılabildiği gibi (kayıt eden evraka bağlı olarak), bilgi kayıt tablosu bağımsız olarak da kullanılabilir. Yani kullanıcı serbest bir şekilde bilgi kayıt tablo bilgilerine müdahale edebilir. Kayıt yapan evraka bağımlılığı kullanmayan bilgi kayıt tablolarına ***bağımsız bilgi kayıt tablosu*** denir.

Periyodik bilgi kayıt tablosu ekleme

“Aspirin usta” firmamın verdiği hizmetlerin fiyatlarını içeren periyodik bilgi kayıt tablosunu oluşturalım.

Tasarımcı ortamında

Tasarımcıda bizim eğitim konfigürasyonumuzu açıp yeni *Bilgi kayıt tablosu* konfigürasyon nesnesini ekleyelim.

Bunu yapmak için konfigürasyon nesne ağacında Bilgi kayıt tablosu grubu işaretledikten sonra konfigürasyon nesne ağaç penceresinin ağaç çubuğunda **Ekle** butonuna basalım.

Ekrana gelen konfigürasyon nesne düzeltme penceresinin *Genel* sekmesinde kayıt tablo adını belirleyelim – **Fiyatlar**.

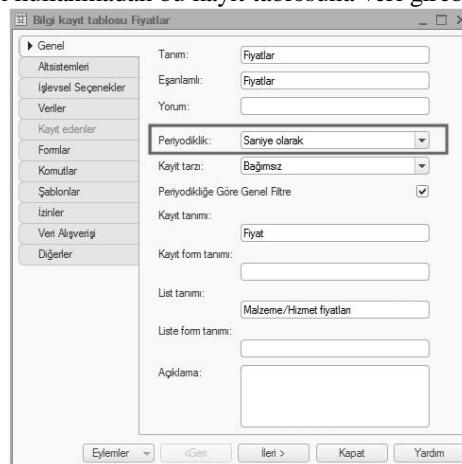
Bu kayıt tablosunun **Periyodiklik** özelliğini *Saniye olarak* işaretleyelim.

Günde birkaç defa değişen fiyatları takip edebilmek için saniye olarak periyodikliğini işaretledik. Fiyatlar bu kadar sık değiştirilmediği durumda periyodiklik *gün içerisinde* de belirlenebilir.

Aynı sekmede nesnenin kullanıcı arayüzündeki tanımları belirleyelim.

Kayıt tanımı özelliğini **Fiyat** olarak ve *Liste tanımı* özelliğini **Malzeme/Hizmet fiyatları** olarak belirleyelim (Resim 9.2).

Kayıt tarzı alanına bakalım. Varsayılan değer olarak değeri **Bağımsız** olarak işaretlendi. Biz bağımsız bir periyodik bilgi kayıt tablosu oluşturduk ve ilerde evrakları kullanmadan bu kayıt tablosuna veri girebileceğiz.

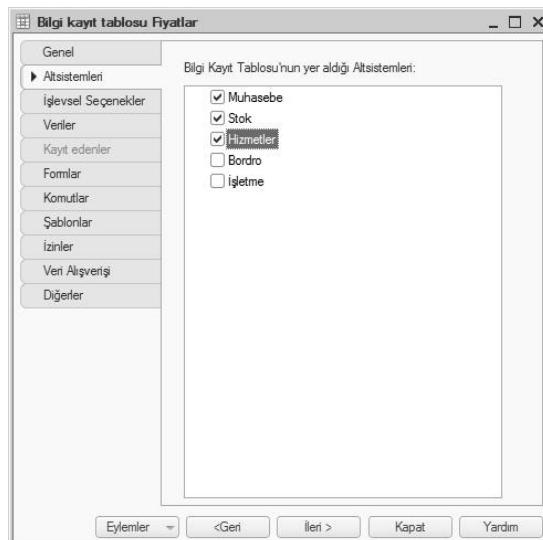


Resim 9.2. “Fiyatlar” bilgi kayıt tablosunun genel özellikler

İleri butona basarak *Alt sistemler* sekmesine geçelim.

Konfigürasyonumuzda uygulanan yapıya göre *Fiyatlar* bilgi kayıt tablosu *Stok*, *Hizmetler* ve *Muhasebe* alt sistemler altında gözükmelidir.

Bu yüzden alt sistem listesinde bu alt sistemlerini işaretleyelim (Resim 9.3).



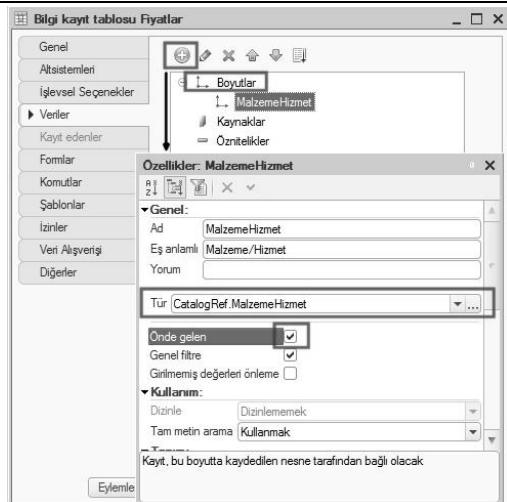
Resim 9.3. Kayıt tablosu yansıtılacağı alt sistemlerini belirtme

Boyutlar ve kaynaklar

Veriler sekmesine geçelim ve **CatalogRef.MalzemeHizmet** türü olan **MalzemeHizmet** boyutu oluşturalım.

Bunu yapmak için *Boyutlar* grubu işaretleyip araç çubuğuunda bulunan **Ekle** butonuna basalım.

Bu boyutun **Önde gelen** özelliğini işaretleyelim (Resim 9.4).



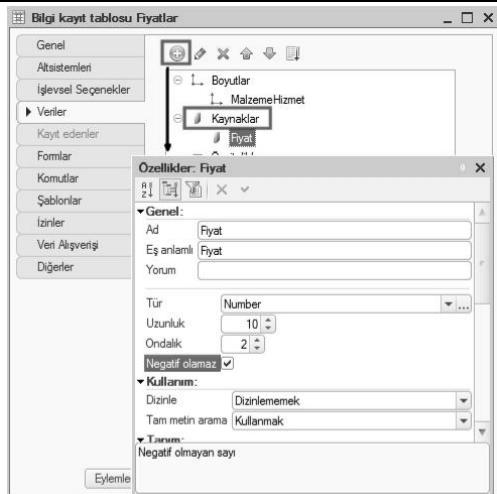
Resim 9.4. Bilgi kayıt tablo boyut ekleme

Önde gelen özelliğini genelde boyutun tipi diğer konfigürasyon nesnesine referans olarak belirlendiğinde kullanılır. *Önde gelen* özelliğin işaretlenmesi şu anlama gelir; kayıt tablo kaydı sadece referansı atılan nesne var olduğunda gereklidir. İlişki atılan nesne kaldırıldığında bu nesneye ait bilgi kayıt tablo kayıtları da silinir.

Kayıt tablo boyutunun *Önde gelen* özelliği işaretlendiğinden *Malzeme/Hizmet* kart listesinin öğe formunun araç çubuğundaki **Git** grubu altında bir komut oluşturur. Bu komuta tıklayarak ilgili malzeme/hizmet ile ilgili bilgi kayıt tablo kayıtlarına geçiş yapılabilir.

Fiyat kaynağı oluşturalım, Tip: **Miktar**, Uzunluk: **15**, Ondalık: **2**, **Negatif olamaz**.

Bunu yapmak için *Kaynaklar* grubu işaretledikten sonra **Ekle** butona basalım (Resim 9.5).



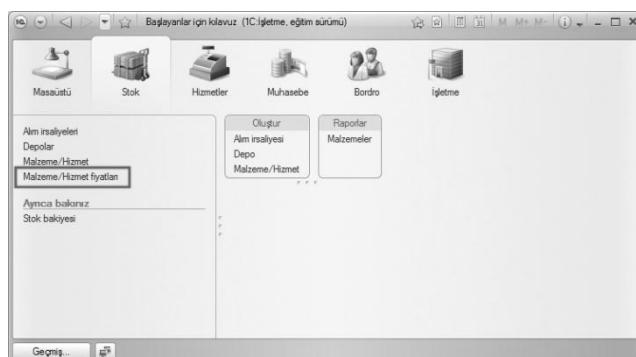
Resim 9.5. Bilgi kayıt tablosunun Kaynağını ekleme

1C: İşletme ortamında

1C: İşletme sistemini hata ayıklama biçiminde çalışıralım ve oluşturduğumuz *Fiyatlar* bilgi kayıt tablosunun nasıl çalıştığını bakalım.

Açılan 1C: İşletme penceresinde *Muhasebe*, *Stok* ve *Hizmetler* bölümlerinde **Malzeme/Hizmet fiyatları** bilgi kayıt tablosunu açmak için komut oluşturuldu (Resim 9.6).

Varsayılan değer olarak bilgi kayıt tablosu açmak için komutlar belirlenmiş olduğu alt sistemler altında yer alır, çünkü birikim kayıt tablosuna göre farklı olarak kullanıcı bu bilgi kayıt tablosuna veriyi manuel olarak girecektir.

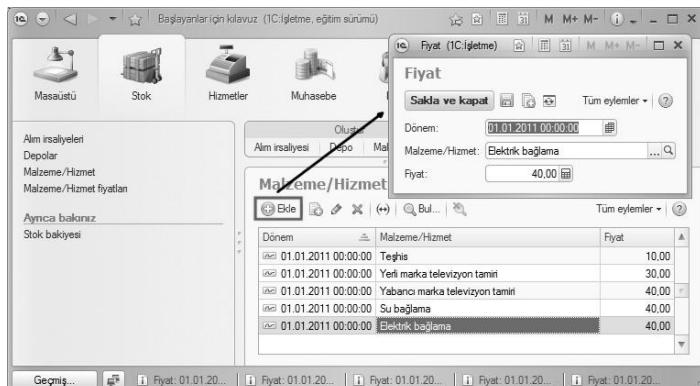


Resim 9.6. "Fiyatlar" periyodik kayıt tablosunu açmak için komut

Bilgi kayıt tablosunda kayıt ekleme

Bilgi kayıt tablosuna yeni kayıt eklemek için **Ekle** butonuna tıklayalım. “Aspirin usta” firmamın hizmet fiyatları aşağıdaki gibi belirleyelim (Resim 9.7).

Fiyatları eklerken geçmiş bir tarih belirleyelim. Bu tarih oluşturduğumuz evraklarına tarihinden daha küçük veya eşit olmalı – 01.01.2011



Resim 9.7. “Fiyatlar” bilgi kayıt tablosunda hizmet fiyatları
Bundan sonra malzemelerin perakende fiyatlarını girelim (Resim 9.8).

Dönem	Malzeme/Hizmet	Fiyat
01.01.2011 00:00:00	Flyback transformer Samsung	45,00
01.01.2011 00:00:00	Flyback transformer GoldStar	20,00
01.01.2011 00:00:00	Transistor Philips 2N2369	0,25
01.01.2011 00:00:00	Lastik hortun	7,50
01.01.2011 00:00:00	Elektrik kablo	1,50

Resim 9.8. “Fiyatlar” bilgi kayıt tablosunda malzeme fiyatları

Programımızda çok kullanışlı bir fonksiyona sahip olduk – malzeme ve hizmetler için fiyat belirleme. Fiyatlar tarihe bağlı olarak kaydedildiğinden önceden fiyatları belirleyebiliriz ve bu fiyatları sadece zaman gelince geçerli olacaktır.

Evrakta malzeme/hizmet seçildiğinde otomatik olarak fiyat doldurma

Malzeme/hizmet fiyatı ayrı bir bilgi kayıt tablosunda bulunur. Hizmet faturası oluşturduğumuz veya değiştirdiğimizde tablo bölümüne bir malzeme veya hizmet eklediğimiz zaman fiyat alanına Fiyatlar bilgi kayıt tablosundan evrak tarihine göre ilgili malzeme veya hizmetin güncel fiyatın gelmesini istiyoruz.

Bunun için iki şey yapmamız gereklidir.

İlk önce güncel fiyatını veren fonksiyonu yazmak ve evrakta malzeme veya hizmet eklendiğinde bu fonksiyonu çağrııp elde edilen fiyatın ilgili alana doldurmak gereklidir.

Böyle bir fonksiyon sadece hizmet faturasında değil, diğer evraklarda da gerekliliği olacak için her nesnenin ulaşabileceği bir yerde yerleştirelim onu – *genel modülde*.

Tasarımcı ortamında

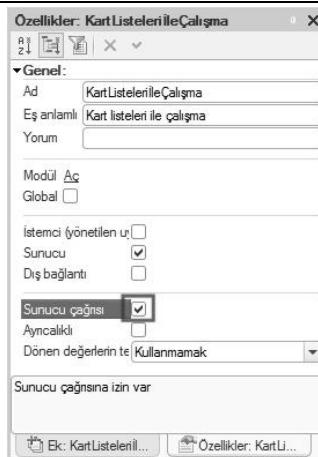
Malzeme/hizmet fiyatını veren fonksiyon

İlk önce biz malzeme veya hizmetin güncel fiyatını bulan **PerakendeFiyat()** fonksiyonu yazalım ve konfigürasyonun genel modülüne yerleştirelim.

Tasarımcı ortamında, Genel – Genel modüller yeni Modül konfigürasyon nesnesini ekleyelim – **KartListeleriİleÇalışma**.

Modül için varsayılan değer olarak Sunucu özelliği işaretlenmiş durumda olduğunu görüyoruz. Bu modül sadece server tarafında oluşturulacağını ifade ediyor.

İstemciden bu modülü çağırabilmek için **Sunucu çağrısı** özelliğini işaretleyelim (Resim 9.9).



Resim 9.9. Genel modül özellikleri
İçine aşağıdaki metin yazalım (Liste 9.1).

Liste 9.1. “PerakendeFiyat” fonksiyonu

```
Function PerakendeFiyat(GüncelTarih, MalzemeHizmetÖgesi) Export
```

```
//Filtre ögesini oluşturmak
Filtre = New Structure("MalzemeHizmet", MalzemeHizmetÖgesi);

//Kayıt tablosunun güncel bilgilerini almak
KaynakDeğerleri      =      InformationRegisters.Fiyatlar.      GetLast
(GüncelTarih, Filtre);
Return KaynakDeğerleri.Fiyat;

EndFunction
```

Fonksiyonu açıklayalım.

Fiyatı almak için fonksiyona iki tane parametre aktaracağız;

- **GüncelFiyat** – Date türü olan parametre. Zaman doğrusunda fiyat almak istediğimiz notayı belirler.
- **MalzemeHizmetÖgesi** – Fiyat almak istenilen MalzemeHizmet ögesine referans.

Fonksiyon gövdesinde biz ilk olarak ek ödeme oluşturuyoruz – *Filtre*.

Bu bir Structure nesnesi, kayıt tablo boyutlarına göre filtre içerir. Bu nesne vasıtasiyla kayıt tablosunun kayıtlarından sadece bize gereken kayıtları seçiyoruz; kayıtların *MalzemeHizmet* boyutunda ki değerler

MalzemeHizmetÖgesi ile aktarılan değere eşit olacaktır (yani belli bir malzeme veya hizmetin fiyatları alıyoruz).

Structure'un anahtarı ("*MalzemeHizmet*"), tasarımcıda belirttiğimiz kayıt tablo boyutunun adı ile aynı olmalıdır. Structure'un değeri ise bu boyutun filtreleme değerini belirler (*MalzemeHizmetÖgesi*).

İkinci satırda *Fiyatlar* bilgi kayıt yöneticisine başvuruyoruz (*InformationRegisters.Fiyatlar*) ve *GetLast()* metodu uyguluyoruz. Sonuçta fonksiyona aktarılan *GüncelTarih* ve boyut değerlerine (*Filtre*) göre bilgi kayıt tablosunun son değerini elde ediyoruz.

GetLast() metodunu structure tipinde kaynaklar değerlerini verir, bu structure *KaynakDeğerleri* değişkene kaydedilir. Bilgi kayıt tablosunun birden fazla kaynak olabilir. Bizim kayıt tablomuzda tek bir kaynak var, fakat buna rağmen tek öğe olan structure oluşturur.

Bu yüzden sonraki satırda bize gerekli olan perakende fiyat değerini almak için, noktayı kullanarak kayıt tablo kaynağının adını belirtmek gereklidir (*KaynakDeğerleri.Fiyat*). Bu değeri fonksiyon gerçekleştirdiğinde aktarıyoruz.

Şimdi bu fonksiyonu evrakta malzeme veya hizmeti seçtiğinde çağırılmak gereklidir.

Evrakta malzeme veya hizmet seçildiğinde fonksiyonu çağrıma ve fiyatı doldurma

Kullanıcı *HizmetFaturası* evrakında hizmeti seçtiği zaman, fiyatın otomatik olarak *Fiyat* alanına aktarılmasını sağlamak gereklidir. Bir de hizmet fiyatı evrakin tarihine göre alınmalıdır.

Tasarımcıda *HizmetFaturası* evrakını bulalım ve evrakın *DocumentForm*'unu açalım.

MalzemelerMalzeme form öğesinde çift tıklayarak veya sağ tuş menüsünden *Özellikler* maddesini seçerek öğe özellik panosunu açalım. Listenin en aşağısında *OnChange* olayı bulun bu olayın büyütme butonuna basalım.

Sistem form modülünde bu olayın işleyici prosedürü oluşturur ve form yapılandırıcısının Modül sekmesini açar.

Açılan prosedüre aşağıdaki metin yazalım (liste 9.2).

Liste 9.2. “*MalzemelerMalzemeOnChange*” prosedürü

```
// Tablo bölümünün geçerli satırı almak
TabloBölümSatırı = Items.Malzemeler.CurrentData;

//Fiyat belirleyelim
TabloBölümSatırı.Fiyat = KartListeleriİleÇalışma.PerakendeFiyat(Object.Date,
TabloBölümSatırı.Malzeme);
```

```
//Satır toplamını hesaplayalım
EvvaklarlaÇalışma.TutarHesaplama(TabloBölümSatırı);
```

Prosedür içeriğini açıklayalım.

Prosedürün ilk satırı *MalzemelerMiktarOnChange* veya *MalzemelerFiyatOnChange* prosedürlerde kullandığımız için biliyoruz. Evrakin tablo bölümünün geçerli satırı alıp *TabloBölümSatırı* değişkene kaydediyoruz.

Ondan sonra *KartListeleriİleÇalışma* genel modülünden *PerakendeFiyat()* fonksiyonu çağrıyoruz.

İlk parametre olarak evrakin tarihini aktarıyoruz. Evrakin tarihi formun genel özniteligidenden alıyoruz – *Object.Date*.

İkinci parametre olarak evrakin geçerli tablo bölüm satırında bulunan *MalzemeHizmet* kart listesine referansı aktarıyoruz (*TabloBölümSatırı.Malzeme*).

Fonksiyon fiyatın son değerini veriyor ve biz bu değeri geçerli tablo bölüm satırındaki Fiyat alanına dolduruyoruz (*TabloBölümSatırı.Fiyat*).

Ondan sonra *EvraklarlaÇalışma* genel modülünden *TutarHesaplama()* prosedürü çağrıyoruz. Önceki derslerde bu prosedürü tutarın otomatik olarak hesaplanması için oluşturmuştu.

MalzemelerMalzemeOnChange prosedür kendisi form modülünde istemci tarafında çalışmaya başlıyor çünkü bu prosedür formun enteratif olay işleyicisidir. Bu prosedürü oluştururken platform otomatik olarak *&AtClient* komutu yerleştirmiştir.

Sonra *PerakendeFiyat()* fonksiyonu çağrırlar. Bu fonksiyon istemci tarafında bulunamayacağı için fonksiyon uygulanması, sunucu tarafında uygulanacak olan, *KartListeleriİleÇalışma* genel modülüne aktarılacaktır. Fonksiyon tamamlandığında kaynak kodu istemci tarafında uygulanmaya devam eder.

Niye böyle bir “kurnazlık” kullanılmış? Kodun uygulanması niye sunucuya aktamasına ihtiyaç duyuldu?

Veritabanı ile herhangi bir çalışma (veri okuma, kaydetme) sadece sunucuda yapılabilir. Bizim durumuzda belli bir malzeme/hizmet için bilgi kayıt tablosundan son veriyi okumak gerekiydi.

Bu gibi eylemler sadece sunucuda yapılabilir. Sözdizim sihirbazına bakarsak bilgi kayıt tablosunun *GetLast()* metot açıklamasına bakarsak, bu metodun sadece sunucuda, kalın istemcide ve dış bağlantısında ulaşılabilir olduğunu görüyoruz.

Kalın istemci ve dış bağlantı – bu bir önceki platform sürümünün istemci biçimleridir. Onlar önceki uygulama çözümleri ile uyumluluk sağlamak için kullanılır.

Biz şu anda tamamen yeni uygulama çözümü geliştiriyoruz. Bu uygulama çözümü ince veya web istemci olarak çalışacaktır. Bu yüzden bizim

örneğimizde veritabanından herhangi bir bilgi almak için kodun uygulanmasını sunucuya aktarmak ve hesaplanmış veriyi istemciye geri aktarmak gereklidir. Yaptığımız şey buydu zaten.

1C: İşletme ortamında

Evrakin nasıl çalıştığını kontrol edelim.

1C:İşletme sistemini hata ayıklama biçiminde çalıştırılarım ve *Fiyatlar* bilgi kayıt tablosunu açalıım.

Philips transistorü için yeni fiyat belirleyelim. Tarihi 17.03.2011 olsun. (Resim 9.10).

Dönem	Malzeme/Hizmet	Fiyat
01.01.2011 00:00:00	Transistor Philips 2N2369	0,25
17.03.2011 00:00:00	Transistor Philips 2N2369	0,35

Resim 9.10. "Fiyatlar" kayıt tablosu

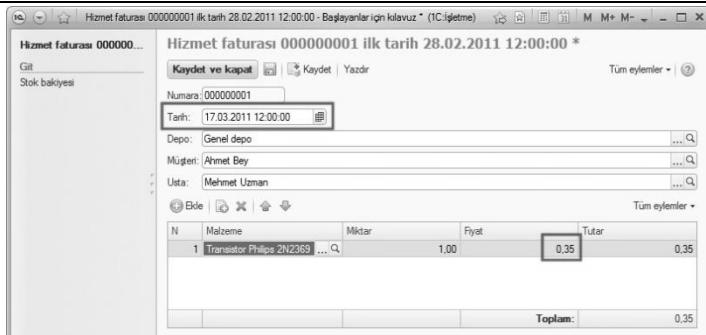
Şimdi 1 nolu hizmet faturası açalıım. Bu evrak ile 1 tane transistor sarf etmiştim.

Evrakin tarihi değiştirmeden evrakin tablo bölümünde Philips transistorü tekrar seçelim. Otomatik olarak 01.01.2011 tarihinde belirlenmiş fiyat ilgili alanına gelecektir. Bu değer evrak tarihi için son değerdir (Resim 9.11).

N	Malzeme	Miktar	Fiyat	Tutar
1	Transistor Philips 2N2369	1,00	0,25	0,25

Resim 9.11. "Hizmet faturası" evrakinin doldurması

Şimdi evrakin tarihini 17.03.2011 olarak değiştirelim ve transistorü tekrar seçelim. Bu tarih için son fiyat değeri belirlenecektir (Resim 9.12).



Resim 9.12. "Hizmet faturaası" evrakının doldurması

Sorular

- ✓ Bilgi kayıt tablosu konfigürasyon nesnesi ne için kullanılır?
- ✓ Bilgi kayıt tablosu özellikleri nedir?
- ✓ Bilgi kayıt tablosunun birikim kayıt tablosundan ana farkları nedir?
- ✓ Bilgi kayıt tablo kayıtların eşsizliği hangi alanlar tanımlar?
- ✓ Peryodik bilgi kayıt tablosu ve bağımsız bilgi kayıt tablosu nedir?
- ✓ Peryodik bilgi kayıt tablosu nasıl oluşturulur?
- ✓ Önde gelen bilgi kayıt tablo boyutu nedir?
- ✓ Kaynak kodu kullanarak en son bilgi kayıt tablo kayıtları nasıl elde edilir?

DERS 10

Sayımlar

Süre

Dersin tahmin süresi – 30 dakika

Sayım nedir	238
Sayım ekleme	238
Malzeme/Hizmet öğelerin “Malzeme hizmet türleri” sayımla bağlama	239
Sadece Malzeme türü olan MalzemeHizmet öğesinin depodan çıkışını kaydetme	241
Sorular	244

Şu ana kadar Malzeme/Hizmet kart listesinin ögesi malzeme mi hizmet mi olduğunu tanımlamamışız. Kart liste öğeler belli bir grplara yerleştirilmişti, fakat bu durum güvenilir bir sınıflandırma sayılmaz; gruplar kaldırılabilir, yeniden adlandırılabilir veya öğeler başka bir mantığa göre gruplandırılabilir.

Bu yüzden hiyerarşi yapısına bağımsız olarak her kart liste ögesi için malzeme veya hizmet belirtisi tanımlamak gereklidir.

Bu derste Malzeme/Hizmet kart listesi için özel öznitelik oluşturacağız, bu özniteliğin tipi bizim için daha yeni konfigürasyon nesnesi olacaktır – **Sayımlar**. İlerde bu özniteliği ile malzeme/hizmet kart liste ögesinin malzeme veya hizmet olduğunu belirleyebileceğiz.

Bunun dışında HizmetFaturası evrakının kaydetme prosedürü düzelticeğiz ve kaynak kod kullanarak sayıml ile nasıl çalışılacağını göstereceğiz.

Sayımlar nedir

Sayımlar, konfigürasyon çalışma aşamasında değişmez değer takımlarının yapısını tanımlayan bir konfigürasyon nesnesidir. Sayımlar konfigürasyon nesnesi için platform veritabanında bir tablo oluşturur, bu tabloda bu değişmez değerler depolanacaktır.

Gerçek hayatı sayımlar olarak fiyat tipi kullanılabilir (“KDV Dahil”, “KDV Hariç”). Sayımlar için var olabilecek tüm değerler geliştirici tarafından Tasarımcı ortamında belirlenir ve kullanıcı kullanma ortamında bu değerleri değiştiremez, ekleyemez ve kaldırılamaz.

Buradan sayımların önemli özelliği ortaya çıkarıyor: sayımlar sadece tasarımcı ortamında düzeltilebildiği için programın algoritmalarında sayımların değerleri kullanılabilir.

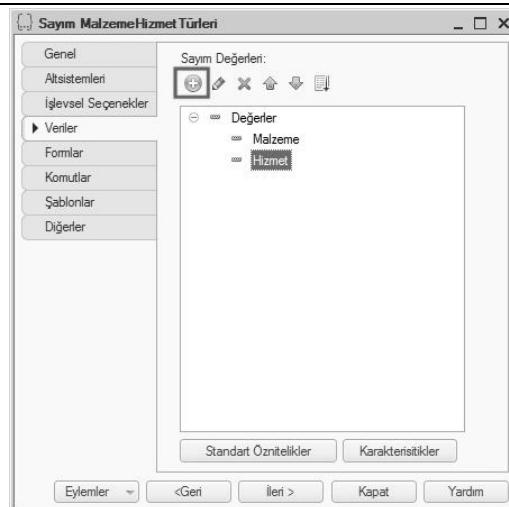
Sayımlar ekleme

Tasarımcı ortamında

Tasarımcı açıp ilk önce yeni bir Sayımlar konfigürasyon nesnesini ekleyelim – **MalzemeHizmetTürleri**.

Veriler sekmesinde iki tane sayımlar ekleyelim; **Malzeme** ve **Hizmet**.

Bunu yapmak için sayımların üst bölümünde bulunan **Ekle** butona tıklayalım (Resim 10.1).



Resim 10.1. “MalzemeHizmetTürleri” sayım değerleri

Malzeme/Hizmet öğelerin “Malzeme hizmet türleri” sayım ile bağlama

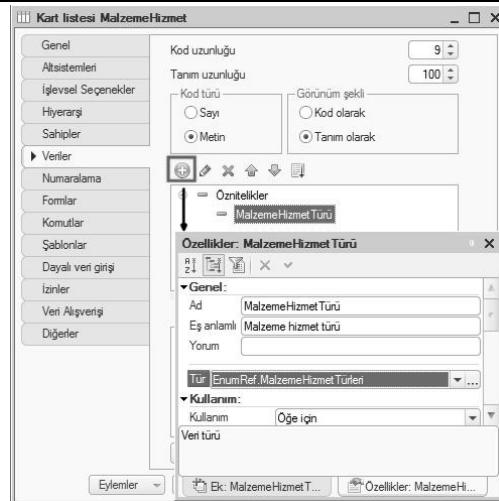
Malzeme/Hizmet öğelerini sayıma bağlamak için aşağıdaki iki adımı izleyelim;

- Tasarımcı ortamında Malzeme/Hizmet kart listesi için bir öznitelik oluşturalım, bu öznitelik sayım değerini içerir.
- 1C:İşletme ortamında Malzeme/Hizmet kart listesinin her ögesi için bu özniteligin değerleri seçip tanımlayalım.

Tasarımcı ortamında

MalzemeHizmet kart listesine yeni **MalzemeHizmetTürü** özniteligi ekleyelim, türü **EnumRef.MalzemeHizmetTürleri**.

Bunu yapmak için MalzemeHizmet konfigürasyon nesnesinin düzeltme penceresini açıp *Veriler* sekmesinde öznitelik listesinin araç çubuğunda **Ekle** butona tıklayalım (Resim 10.2).



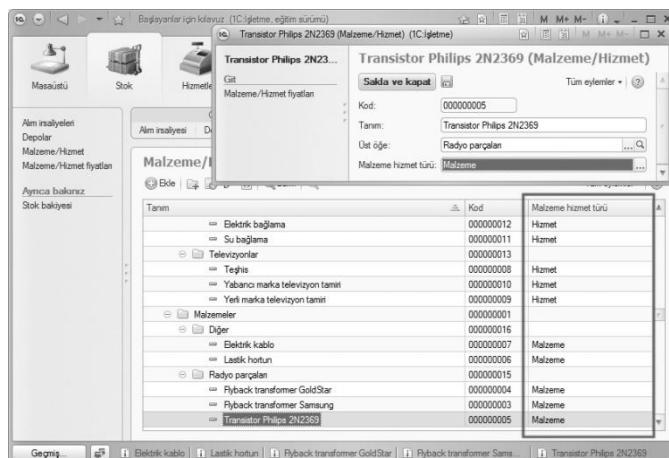
Resim 10.2. "MalzemeHizmet" kart listesinin verileri

1C: İşletme ortamında

Bundan sonra 1C:İşletme'yi hata ayıklama biçiminde çalıştırıralım.

Platform sayının hiçbir alt sisteme ait olmadığına dair bir uyarı mesaj ekrana getirir. Kabul edip konfigürasyon değişiklikleri uygulayalım.

1C:İşletme ortamında *Malzeme/Hizmet* kart listesinin her öğesi için *Malzeme hizmet türü* öznitelinin gerekli değeri seçelim (Resim 10.3).



10.3. "MalzemeHizmet" kart liste verileri

Şimdi MalzemeHizmetTürleri sayının kullanımı sayesinde ortaya çıkan yeni olanakları nasıl kullanılabileceğine bakalım.

Sadece Malzeme türü olan MalzemeHizmet ögesinin depodan çıkışını kaydetme

Altıncı derste, HizmetFaturası hareketleri oluşturulduğunda, evrakin StokBakiyesi birikim kayıt tablosuna yapılan kayıtlar tam olarak doğru olmadığını söylemişik, çünkü kayıt tablosuna sadece sarf edilen malzemeler değil verilen hizmet bilgileri de kaydedilir (Liste 10.1).

Liste 10.1. HizmetFaturası evrakinin “Posting” prosedürü

```
Procedure Posting(Cancel, Mode)
```

```
//{{{{REGISTER_REGISTERRECORDS_WIZARD
```

```
// Bu parça yapılandırıcı tarafından oluşturuldu.
```

```
// Yapılandırıcıyı tekrar kullanma sürecinde elle kaydedilen değişiklikler
// kaybedilecektir!!!
```

```
// kayıt tablosu StokBakiyesi Expense
```

```
RegisterRecords.StokBakiyesi.Write = True;
```

```
For Each CurRowMalzemeler In Malzemeler Do
```

```
    Record = RegisterRecords.StokBakiyesi.Add();
```

```
    Record.RecordType = AccumulationRecordType.Expense;
```

```
    Record.Period = Date;
```

```
    Record.Malzeme = CurRowMalzemeler.Malzeme;
```

```
    Record.Depo = Depo;
```

```
    Record.Miktar = CurRowMalzemeler.Miktar;
```

```
EndDo;
```

```
//}}}_REGISTER_REGISTERRECORDS_WIZARD
```

```
EndProcedure
```

Şimdi biz bu evraki biraz değiştirip kayıt tablosuna sadece malzemeler ile ilgili kayıtların yapıldığını sağlayacağız.

Bunu yapmak için ilk önce Tasarımcı ortamında HizmetFaturası'nın kaydetme prosedürü değiştirelim ve 1C:İşletme ortamında Hizmet faturaları tekrar kaydedelim.

Yapacağımız düzeltme, performans açısından pek verimli değil, fakat bu aşamada StokBakiyesi kayıt tablosuna doğru veri kaydetmesini sağlayacak.

Tasarımcı ortamında

Evrakin yaptığı hareketleri düzeltelim. Prosedür döngüsünden *hizmet* malzeme hizmet türü olan tablo bölüm satırlarını çıkartalım.

Bunu yapmak için tasarımcıda *HizmetFatura* evrakinin nesne modülünü açıp (*HizmetFatura* sağ tuş – Nesne modülüni aç) *Posting* olayın işleyici prosedürüne bu koşulu ekleyelim.

Sonuç olarak *Posting* prosedürü aşağıdaki gibi görünmelidir (Liste 10.2).

Liste 10.2. HizmetFatura evrakinin “Posting” prosedürü

```
Procedure Posting(Cancel, Mode)
```

```
//{{__REGISTER_REGISTERRECORDS_WIZARD
// Bu parça yapılandırıcı tarafından oluşturuldu.
// Yapılandırıcıyı tekrar kullanma sürecinde elle kaydedilen değişiklikler
// kaybedilecektir!!!

// kayıt tablosu StokBakiyesi Expense
RegisterRecords.StokBakiyesi.Write = True;
For Each CurRowMalzemeler In Malzemeler Do
    If CurRowMalzemeler.Malzeme.MalzemeHizmetTürü =
Enums.MalzemeHizmetTürleri.Malzeme Then
        Record = RegisterRecords.StokBakiyesi.Add();
        Record.RecordType =
AccumulationRecordType.Expense;
        Record.Period = Date;
        Record.Malzeme = CurRowMalzemeler.Malzeme;
        Record.Depo = Depo;
        Record.Miktar = CurRowMalzemeler.Miktar;
    EndIf;
EndDo;
//}}__REGISTER_REGISTERRECORDS_WIZARD
```

```
EndProcedure
```

Eklenen metin şunu yapar: evrakin tablo bölümündeki her satır için kontrol yapar, malzeme/hizmetin türü malzeme ise kayıt tablosuna kayıt yapar, değilse yapmaz.

CurRowMalzemeler değişkeni döngünün her aşamasında *Malzemeler* tablo bölümünün geçerli satır bilgisini içerir.

Nokta ile sütunun adını belirleyerek tablo bölüm satırında bulunan *Malzeme/Hizmet* referansına başvuruyoruz (*CurRowMalzemeler.Malzeme*).

Sonra nokta ile *MalzemeHizmetTürü* belirleyerek (*CurRowMalzemeler.Malzeme.MalzemeHizmetTürü*) bu *MalzemeHizmet* kart liste ögesinin *MalzemeHizmetTürü* özniteligi ulaşiyoruz.

Elde edilen değer eşitleme operatörü (=) kullanarak *MalzemeHizmetTürleri* sayiminin *Malzeme* değeri ile karşılaştırıyoruz (*Enums.MalzemeHizmetTürleri.Malzeme*).

Değerler bir birini tutuyorsa, döngü içeriği uygulanır. Tutmuyorsa, döngünün bir sonraki aşamaya yani bir sonraki satır geçiyoruz.

1C: İşletme ortamında

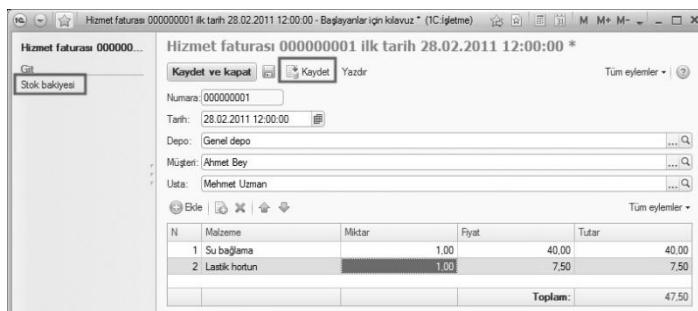
1C: İşletme sistemini hata ayıklama biçiminde çalıştırıp Hizmet faturası evrakinin nasıl çalıştığını bakalım.

Hizmetler bölümünde *Hizmet faturaları* komutu uygulayarak Hizmet fatura listesini açalım.

1 nolu hizmet faturasını açıp aşağıdaki değişiklikler uygulayalım;

- Philips transistorü içeren satırı kaldırıyalım,
- Hizmet ekleyelim – Su bağlaması,
- Malzeme ekleyelim – Lastik hortum (Resim 10.4).

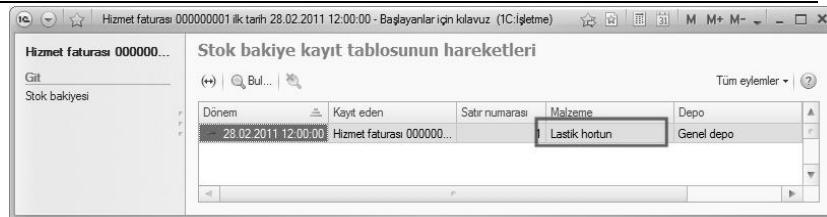
Fiyatların Fiyatlar bilgi kayıt tablosundan otomatik olarak geldiğine dikkat edelim.



Resim 10.4. Değiştirilmiş “1 nolu hizmet faturası”

Formun araç çubuğuunda **Kaydet** butonuna basalım.

Evrakin yaptığı stok bakiye hareketlerine geçmek için, formun sol tarafında bulunan **Stok bakiyesi** komutunu uygulayalım (Resim 10.5).



Resim 10.5. "Stok bakiyesi" kayıt tablo kayıtları

Gördüğümüz gibi, Stok bakiyesi kayıt tablosunun kayıtları sadece malzeme olan satırları içerir. Su bağlaması ile ilgili kayıt tabloda yoktur.

Sorular

nesnesi ne için kullanılır?

- ✓ *Yeni sayım nasıl oluşturulur?*
- ✓ *Sayım kullanarak kart liste öğelerini belli bir mantıksal gruba nasıl bölebiliriz?*
- ✓ *Kaynak kodu kullanarak sayım değerlerine nasıl ulaşabiliriz?*

DERS 11

Evrakı Birden Fazla Kayıt Tablosuna Kaydetme

Süre

Dersin tahmin süresi – 1 saat 20 dakika

Evrakı birden fazla kayıt tablosuna kaydetmek ne için gerekir	246
Birikim kayıt tablosunu ekleme	246
Alım ırsaliyesinin iki kayıt tablosuna veri kaydetmesi	248
Kaydetme prosedürünü değiştirme	248
Kayıt tablo kayıtlarına geçiş komutu	250
Hizmet faturasının iki kayıt tablosuna veri kaydetmesi	252
Yeni evrak özniteligi	252
Kaydetme prosedüründe değişiklik	254
Sorular	258

Bu derste tek evrakin birden fazla konfigürasyon kayıt tablosuna veri kaydetme olanağını inceleyeceğiz. Birden fazla kayıt tablosuna kaydetme nasıl tanımlandığını ve ne için gerektiğini öğreneceğiz.

Bu derste bir tane daha birikim kayıt tablosu oluşturacağız ve iki kayıt tablosuna veri kaydedebilmek için evrakların kaydetme prosedürlerini değiştireceğiz.

Evrakı birden fazla kayıt tablosuna kaydetmek ne için gerekir

Şu ana kadar “Aspirin usta” işletmemizdeki sadece miktarsal stok hareketlerini takip ediyorduk. Bunu yapmak için *StokBakiyesi* kayıt tablosunun kullanmıştık.

Fakat gördüğünüz gibi sadece miktarsal stok takibi işletmenin ihtiyaçlarını karşılamıyor. Ürünleri almak için ne kadar para harcadığını ve şu anda stokta tutar olarak ne kadar malzeme bulunduğu da bilinmek isteniyor.

“Aspirin usta” firmanın yöneticileri stokun tutarsal durumunun ortalama maliyet olarak hesaplanması istemeleri.

Yani malzeme alındığı zaman, tutar alım fiyatına göre hesaplansın istiyorlar. Satış aşamasında ise satılan malzemenin değeri stokta bulunan malzeme miktarına ve ortalama alış fiyatına göre hesaplanması talep etmişler.

Bu bilgi miktarsal takibe göre farklı yapıya sahip olduğu için, malzemelerin maliyetlerini tutmak için bir tane daha birikim kayıt tablosunu kullanacağız – **StokMaliyeti**.

Böylece *Alımİrsaliyesi* ve *HizmetFaturası* evrakları sadece *StokBakiyesi* kayıt tablosunda değil, *StokMaliyeti* kayıt tablosunda da kayıt yapacaklar.

Birikim kayıt tablosunu ekleme

Tasarımcı ortamında

StokMaliyeti kayıt tablosu karmaşık bir öğe olmadığından derin detaylarına girmeyeceğiz.

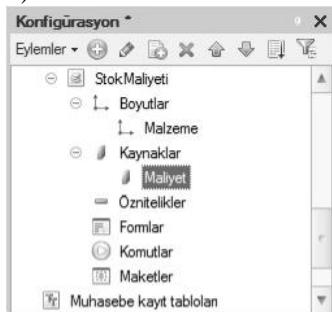
Yeni Birikim kayıt tablosu konfigürasyon nesnesini ekleyelim – *StokMaliyeti*.

Liste tanımını **Stok maliyet kayıt tablo kayıtları** olarak belirleyelim. Bu yazı kayıt tablo listesinde başlık olarak gösterilecektir.

Alt sistemler sekmesinde kayıt tablosunun *Muhasebe*, *Stok* ve *Hizmetler* alt sistemlerinde gösterileceğini belirleyelim.

Veriler sekmesinde bir tane boyut – **Malzeme**, tür CatalogRef.MalzemeHizmet ve bir tane kaynak – **Maliyet**, uzunluk: 15, ondalık: 2 oluşturalım.

Oluşturduktan sonra kayıt tablosu konfigürasyon ağacında aşağıdaki gibi görünmelidir (Resim 11.1).



Resim 11.1. StokMaliyeti birikim kayıt tablosu

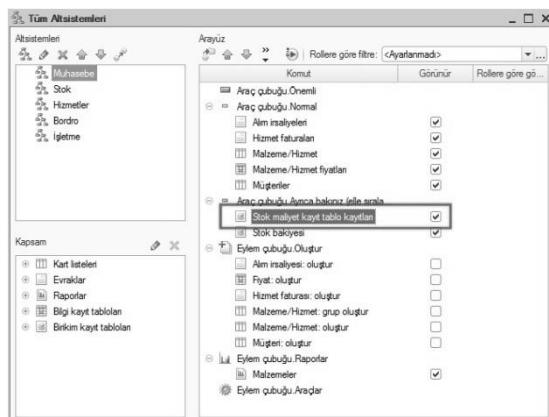
Şimdi kullanıcı arayüzümüzün Muhasebe, Stok ve Hizmetler bölümleri altında kayıt tablo listesini izleme komutlarının gösterilmesi için arayüz ayarlarını düzeltelim.

Konfigürasyon nesne ağacında *Alt sistemleri* grubu işaretleyerek sağ tuş menüsünden *Tüm altsistemler* maddesini seçelim.

Ekrana gelen pencerenin sol tarafında *Alt sistemler* listesinde *Muhasebe* alt sistemini seçelim.

Sağ tarafında bulunan Arayüz listesinde seçilen alt sistemi ile ilgili tüm komutlar listelenir.

Araç çubuğu. Normal grubunda *StokMaliyeti* komut için görünürüğünü işaretleyelim ve fare ile **Araç çubuğu. Ayrıca bakınız** grubuna taşıyalım (Resim 11.2).



Resim 11.2. Alt sistem arayüz ayarları

Aynı şekilde Hizmetler ve Stok alt sistemler için **Araç çubuğu**. Normal grubunda Stok maliyeti komutunun görünürüğünü işaretleyip **Araç çubuğu**. **Ayrıca bakınız** grubuna taşıyalım.

Şimdi evraklarınızın kaydetme prosedürleri değiştirme işlemine geçelim. Basitten başlayalım – *Alım ırsaliyesi* evrakı.

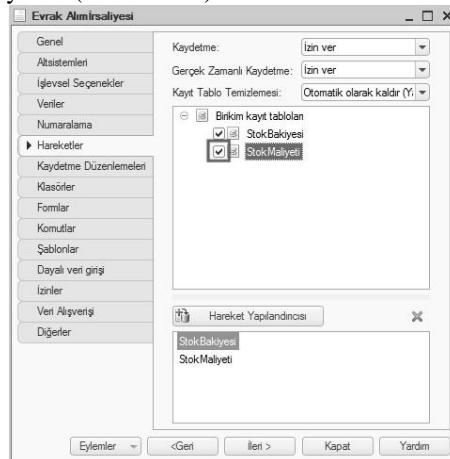
Alım ırsaliyesinin iki kayıt tablosuna veri kaydetmesi

Tasarımcı ortamında

Kaydetme prosedürüni değiştirme

Tasarımcıda *Alım ırsaliyesi* evrakinin düzeltme penceresini açalım ve **Hareketler** sekmesine geçelim.

Kayıt tablo listesinde evrakin StokMaliyeti kayıt tablosuna da kayıt yapacağını işaretleyelim (Resim 11.3).



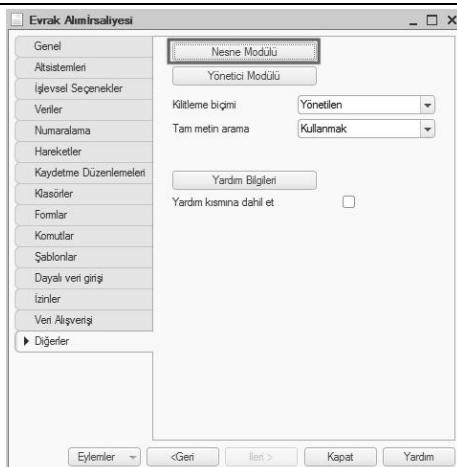
Resim 11.3. Evrak hareketleri

Bu kez evrakına hareket yapılandırıcısını kullanmayacağız. Gerekli olan değişiklikler direkt *Alım İrsaliyesi* evrakinin *Posting* prosedüründe yapacağız.

Hareket yapılandırıcısını kullanarak da birden fazla kayıt tablosuna kaydetme prosedürü oluşturulabilir. Fakat mevcut kaydetme prosedürü varken, hareket yapılandırıcısı kullanımı var prosedürün silinmesine sebep olur ve her iki kayıt tablosu için yeniden hareket tanımlamak gereklidir. Bu yüzden var olan prosedüre manüel olarak değişiklik tanımlamak daha kolay olur.

Digerler sekmesine geçip nesne modülü açalım.

Bunu yapmak için Nesne modülü butona tıklamak yeterlidir (Resim 11.4).



Resim 11.4. Nesne modülünü açma

Posting prosedürü açalım.

Döngünün en alta, EndDo satırından önce StokMaliyet kayıt tablosuna kayıt yapacak kod satırlarını ekleyelim (Liste 11.1).

Liste 11.1. “Alımlısalıyesi” kaydetme prosedürü (parça)

```
// kayıt tablosu StokMaliyeti Receipt
Record = RegisterRecords.StokMaliyeti.Add();
Record.RecordType = AccumulationRecordType.Receipt;
Record.Period = Date;
Record.Malzeme = CurRowMalzemeler.Malzeme;
Record.Maliyet = CurRowMalzemeler.Tutar;
```

Döngü başlamadan önce kayıt tablosunun kayıt seti için *Write* özelliği içi *True* değeri belirleyelim. Ve hareket yapılandırıcısının oluşturduğu yorumlar kaldırıralım.

Sonuç olarak Posting prosedürü aşağıdaki gibi görünür (Liste 11.2).

Liste 11.1. “Alımİrsaliyesi” kaydetme prosedürü (parça)

```
Procedure Posting(Cancel, Mode)
```

```
    RegisterRecords.StokBakiyesi.Write = True;  
    RegisterRecords.StokMaliyeti.Write = True;  
  
    For Each CurRowMalzemeler In Malzemeler Do  
        // kayıt tablosu StokBakiyesi Receipt  
        Record = RegisterRecords.StokBakiyesi.Add();  
        Record.RecordType = AccumulationRecordType.Receipt;  
        Record.Period = Date;  
        Record.Malzeme = CurRowMalzemeler.Malzeme;  
        Record.Depo = Depo;  
        Record.Miktar = CurRowMalzemeler.Miktar;  
  
        // kayıt tablosu StokMaliyeti Receipt  
        Record = RegisterRecords.StokMaliyeti.Add();  
        Record.RecordType = AccumulationRecordType.Receipt;  
        Record.Period = Date;  
        Record.Malzeme = CurRowMalzemeler.Malzeme;  
        Record.Maliyet = CurRowMalzemeler.Tutar;  
  
    EndDo;  
  
EndProcedure
```

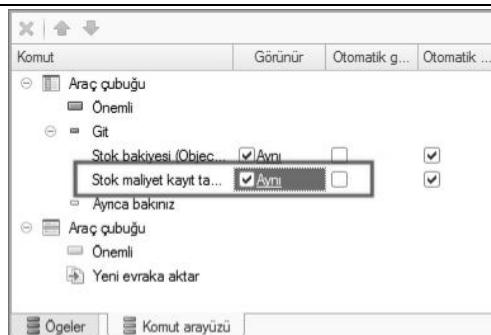
Kayıt tablo kayıtlarına geçiş komutu

Son olarak, evraktan direkt yaptığı kayıt tablo kayıtlarına ulaşmak için evrakin document formunu düzeltelim.

Bunu yapmak için Alımİrsaliyesi evrakinin document formu açalım.

Sol üst köşede *Komut arayüz* sekmesine geçelim.

Araç çubuğu bölümünde **Git** grubu açıp Stok maliyeti kayıt tablosunu açmak için bir komut göreceğiz. Komutun görünürüğünü işaretleyelim (Resim 11.5).



Resim 11.5. Alımİrsaliyesi evrak formunun arayüz ayarları

1C: İşletme ortamında

1C:İşletme ortamında tüm alım ırsaliyeleri tekrar kaydetmek gereklidir. Bu işlem evrakların yeni kaydetme düzeneğine göre kayıt yapmaları için gereklidir.

1C:İşletme sistemini hata ayıklama biçiminde çalıştırılarım. *Stok* bölümündeki bulunan *Alım ırsaliyeleri* komutu uygulayarak alım ırsaliye listesini açalım.

Ctrl tuşu kullanarak tüm alım ırsaliyeleri seçelim ve **Tüm eylemler – Kaydet** menüsü kullanarak tüm evrakları yeniden kaydedelim.

Ondan sonra birinci evrakı açıp (11.6) *StokBakiyesi* ve *StokMaliyeti* kayıt tablolarına geçiş butonlarını kullanarak evrakin gerekli kayıtları hem *StokBakiyesi* kayıt tablosunda (Resim 11.7) hem de *StokMaliyeti* kayıt tablosunda (Resim 11.8) yaptığından emin olalım.

N	Malzeme	Miktar	Fiyat	Tutar
1	Flyback transformer GoldStar	10,00	11,00	110,00
2	Flyback transformer Samsung	10,00	25,00	250,00
3	Transistor Philips 2N2369	10,00	0,50	5,00

Resim 11.6. 1 nolu Alım İrsaliyesi

İsim isimleyici	İlk tarih	Bitiş tarih	Toplam miktar		
Aynı isimleyici 000000001	27.02.2011 12:00:00				
Dönem	Kayıt eden	Satır numarası	Malzeme	Depo	Miktar
+ 27.02.2011 12:00:00	Aynı isimleyici 000000001	1	Ryback transformer Go...	Genel depo	10.000
+ 27.02.2011 12:00:00	Aynı isimleyici 000000001	2	Ryback transformer Sa...	Genel depo	10.000
+ 27.02.2011 12:00:00	Aynı isimleyici 000000001	3	Transistor Philips 2N2369	Genel depo	10.000

Resim 11.7. “Stok bakiyesi” kayıt tablosunun kayıtları

İsim isimleyici	İlk tarih	Bitiş tarih	Toplam maliyet	
Aynı isimleyici 000000001	27.02.2011 12:00:00			
Dönem	Kayıt eden	Satır numarası	Malzeme	Maliyet
+ 27.02.2011 12:00:00	Aynı isimleyici 000000001 İk. tarih 27...	1	Ryback transformer GoldStar	110.00
+ 27.02.2011 12:00:00	Aynı isimleyici 000000001 İk. tarih 27...	2	Ryback transformer Samsung	250.00
+ 27.02.2011 12:00:00	Aynı isimleyici 000000001 İk. tarih 27...	3	Transistor Philips 2N2369	5.00

Resim 11.8. “Stok maliyeti” kayıt tablosunun kayıtları

Hizmet faturasının iki kayıt tablosuna veri kaydetmesi

Hizmet faturası evrakının kaydetme prosedürüne de bazı değişiklik ekleyeceğiz.

Bunu yaparken “Aspirin ustası” işletme yöneticilerin isteklerini dikkate alacağız. Hizmet verildiğinde sarf edilen malzemeler için, geçerli piyasa durumuna göre yöneticilerin uygun gördüğü, aynı malzeme için farklı maliyet girilebiliridir.

Hizmet faturasında sadece malzeme fiyatı bulunduğu için aşağıdaki maddeleri uygulamak gereklidir;

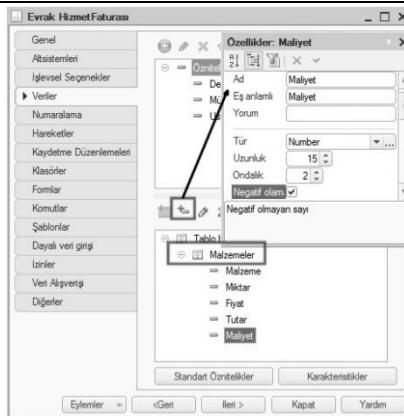
1. Evrakin tablo bölümünde bir tane daha öznitelik eklemektir. Bu öznitelijke malzemenin maliyetini gireceğiz.
2. Ondan sonra *Hizmet faturası* evrakının kaydetme prosedürünü değiştirmek gerekmektedir.
3. Son olarak 1C:İşletme ortamında oluşturduğumuz kaydetme düzenine göre tüm hizmet faturaları yeniden kaydetmek gerekmektedir.

Tasarımcı ortamında

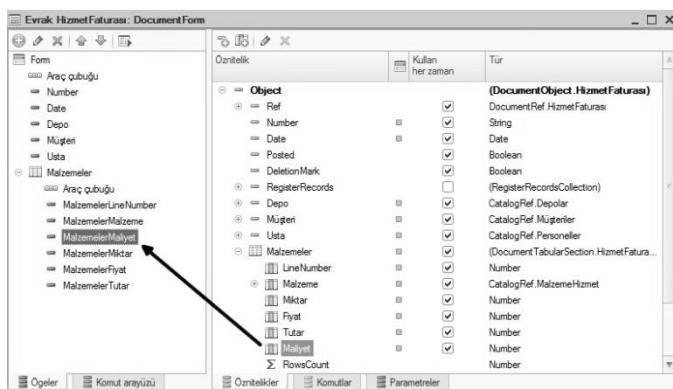
Yeni evrak özniteligi

Tasarımcı ortamında *Hizmet faturası* evrakı konfigürasyon nesnesinin düzeltme penceresini açıp *Veriler* sekmesine geçelim.

Evrakin tablo bölümünde yeni öznitelik oluşturalım – **Maliyet**, tip: **Number**, Uzunluk: **15**, Ondalık: **2**, **Negatif olamaz** (Resim 11.9)



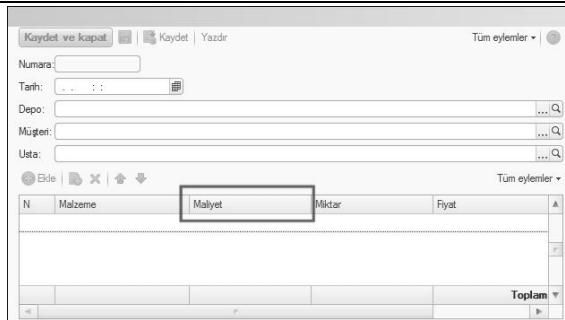
Resim 11.9. Hizmet faturası evrakındaki değişiklikler
*HizmetFatura*sı evrakının *DocumentForm* açalım ve *Malzemeler* tablo bölümünde *Maliyet* alanının gösterilmesini sağlayalım.
Bunu yapmak için form düzeltme penceresinin sağ üst bölümünde *Öznitelikler* sekmesinde **Object** grubu açalım (Resim 11.10).



Resim 11.10. Hizmet faturası evrak formunu düzeltme
Object grubu *HizmetFaturası*'nın tüm özniteliklerini içerdiğini görüyoruz.
Tablo bölümünde *Maliyet* özniteliğini bulup sağ tarafında bulunan form öğe listesine fare ile taşıyalım.

Yeni öğeyi Malzeme alanının altına yerleştirelim. Form öğesinin özellikleri varsayılan olarak bırakalım.

Ekleken öznitelik form düzeltme penceresinin alt bölümünde gösterilecektir (Resim 11.11).



Resim 11.11. HizmetFaturası evrakı formundaki değişiklik

Kaydetme prosedüründe değişiklik

Alımışsaliyesi için yaptığımız gibi HizmetFurası için evrak hareketlerini oluşturalım.

HizmetFurası evrakı konfigürasyon nesnesinin düzeltme penceresinde *Hareketler* sekmesine geçelim.

Kayıt tablo listesinde evrakin *StokMaliyeti* kayıt tablosuna da kayıt yapacağını belirtelim.

Digerler sekmesine geçip nesne modülü açalım. Bunu yapmak için *Nesne modülü* butonuna basmak yeterlidir.

Posting prosedürü açalım.

Döngünün en alta, EndDo satırından önce StokMaliyet kayıt tablosuna kayıt yapacak kod satırlarını ekleyelim (Liste 11.3).

Liste 11.3. "HizmetFurası" kaydetme prosedürü (parça)

```
// kayıt tablosu StokMaliyeti Expense
Record = RegisterRecords.StokMaliyeti.Add();
Record.RecordType = AccumulationRecordType.Expense;
Record.Period = Date;
Record.Malzeme = CurRowMalzemeler.Malzeme;
Record.Maliyet = CurRowMalzemeler.Miktar * CurRowMalzemeler.Maliyet;
```

Döngü başlamadan önce kayıt tablosunun kayıt seti için *Write* özelliği içi *True* değeri belirleyelim. Ve hareket yapılandırıcısının oluşturduğu yorumlar kaldırıralım.

Sonuç olarak Posting prosedürü aşağıdaki gibi görünür (Liste 11.4).

Liste 11.3. “HizmetFaturası” kaydetme prosedürü

```
Procedure Posting(Cancel, Mode)
```

```
    RegisterRecords.StokBakiyesi.Write = True;
RegisterRecords.StokMaliyeti.Write = True;
```

```
    For Each CurRowMalzemeler In Malzemeler Do
        If CurRowMalzemeler.Malzeme.MalzemeHizmetTürü      =
            Enums.MalzemeHizmetTürleri.Malzeme Then
                // kayıt tablosu StokBakiyesi Expense
                Record = RegisterRecords.StokBakiyesi.Add();
                Record.RecordType      =
                AccumulationRecordType.Expense;
                Record.Period = Date;
                Record.Malzeme = CurRowMalzemeler.Malzeme;
                Record.Depo = Depo;
                Record.Miktar = CurRowMalzemeler.Miktar;

                // kayıt tablosu StokMaliyeti Expense
                Record = RegisterRecords.StokMaliyeti.Add();
                Record.RecordType      =
                AccumulationRecordType.Expense;
                Record.Period = Date;
                Record.Malzeme      =
                CurRowMalzemeler.Malzeme;
                Record.Maliyet = CurRowMalzemeler.Miktar *
                CurRowMalzemeler.Maliyet;

            EndIf;
        EndDo;
```

```
    EndProcedure
```

Kayıt tablosunun *Maliyet* kaynağı tablo bölümündeki *Miktar* çarpı *Maliyet* olarak kaydedildiğine dikkat ediniz.

Son olarak evrak formundan evrakinin yaptığı kayıt tablo hareketlerine direkt ulaşabilmek için evrak formunun arayüzü düzeltelim.

Bunu yapmak için *HizmetFaturası* evrakinin *DocumentForm*'u açalım.

Sol üst bölümünde *Komut arayüz* sekmesine geçelim.

Araç çubuğu bölümünde Git grubunda Stok maliyeti kayıt tablosunun açmak için bir komut görüyoruz. Bu komutun görünürüğünü işaretleyelim.

1C: İşletme ortamında

1C:İşletme ortamında tüm hizmet faturaları tekrar kaydetmek gerekir. Bu işlem evrakların yeni kaydetme düzeneğine göre kayıt yapmaları için gereklidir.

1C:İşletme sistemini hata ayıklama biçiminde çalıştıralım. *Hizmetler* bölümü altında bulunan *Hizmet faturaları* komutu uygulayarak hizmet fatura listesini açalım.

1 nolu hizmet faturasını açıp lastik hortumun maliyetini 5 olarak tanımlayalım (Resim 11.12).

N	Malzeme	Maliyet	Fiyat	Tutar
1	Su bağlama	1.00	40.00	40.00
2	Lastik hortun	5.00	7.50	7.50
			Toplam:	47.50

Resim 11.12. 1 nolu hizmet faturası

1 nolu hizmet furasını kaydedip *Stok maliyeti* kayıt tablosunda yapmış olduğu kayıtlarına bakalım.

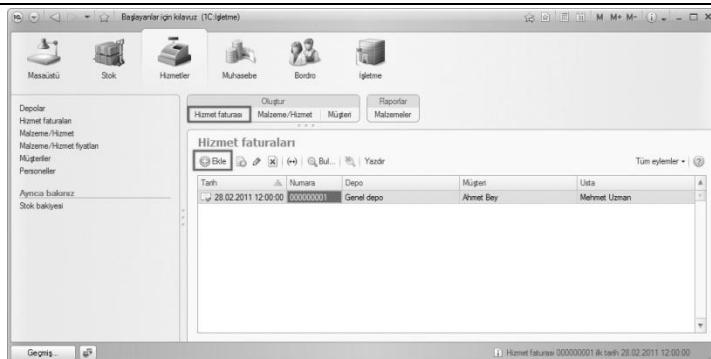
Bunu yapmak için **Kaydet** butona basıp *Stok maliyeti* kayıt tablosuna geçme komutunu uygulayalım (Resim 11.13).

Dönem	Kayıt eden	Satır numarası	Malzeme	Maliyet
28.02.2011 12:00:00	Hizmet fatura 00000001	1	Lastik hortun	5,00

Resim 11.13. Stok maliyeti kayıt tablo kayıtları

Şimdi iki tane daha Hizmet fatura evrakını oluşturup kaydedelim.

Bunu yapmak için hizmet fatura listesinde **Ekle** butonuna basalım veya eylem çubuğundaki **Hizmet faturası** komutunu uygulayalım (Resim 11.14).



Resim 11.14. Yeni evrak oluşturma

Hizmet faturası 00000002 ilk tarih 24.04.2011 19:21:06

Kaydet ve kapat		Kaydet Yazdır	Tüm eylemler		
Numara:	00000002				
Tarih:	24.04.2011 19:21:06				
Depo:	Genel depo	...			
Müşteri:	Ahmet Bey	...			
Usta:	Ali İşçi	...			
<input type="button" value="Ekle"/> <input type="button" value="Sil"/> <input type="button" value="Düzenle"/> <input type="button" value="Ara"/> <input type="button" value="Tüm eylemler"/>					
N	Malzeme	Maliyet	Miktar	Fiyat	Tutar
1	Yabancı marka televizyon ta...			1,00	40,00
2	Ryback transformer Samsung	30,00		1,00	45,00
					Toplam: 85,00

Resim 11.15 2 nolu hizmet faturası

Hizmet faturası 00000003 ilk tarih 24.04.2011 19:23:21

Kaydet ve kapat		Kaydet Yazdır	Tüm eylemler		
Numara:	00000003				
Tarih:	24.04.2011 19:23:21				
Depo:	Genel depo	...			
Müşteri:	Mustafa Bey	...			
Usta:	Ali İşçi	...			
<input type="button" value="Ekle"/> <input type="button" value="Sil"/> <input type="button" value="Düzenle"/> <input type="button" value="Ara"/> <input type="button" value="Tüm eylemler"/>					
N	Malzeme	Maliyet	Miktar	Fiyat	Tutar
1	Elektrik bağılane			1,00	40,00
2	Lاستik hortun	5,00	2,00	7,50	15,00
3	Elektrik kablo	1,00	1,00	1,50	1,50
4	Yerli marka televizyon tamiri			1,00	30,00
5	Ryback transformör GoldStar	13,50	1,00	20,00	20,00
6	Transistor Philips 2N2369	0,15	2,00	0,35	0,70
				Toplam:	107,20

Resim 11.16. 3 nolu hizmet faturası

2 ve 3 nolu hizmet faturaların StokMaliyeti kayıt tablo hareketleri aşağıdaki gibi olmalıdır (Resim 11.17, 11.18).

Stok maliyet kayıt tablo kayıtları				
Dönem	Kayıt eden	Satır numarası	Malzeme	Maliyet
24.04.2011 19:21:06	Hizmet faturaşı 00000002 İlk tarih 2...	1	Flyback transformér Samsung	30,00

Resim 11.17. 2 nolu hizmet furasının hareketleri

Stok maliyet kayıt tablo kayıtları				
Dönem	Kayıt eden	Satır numarası	Malzeme	Maliyet
- 24.04.2011 19:23:21	Hizmet faturaşı 00000003 İlk tarih 2...	1	Lastikhortun	10,00
- 24.04.2011 19:23:21	Hizmet faturaşı 00000003 İlk tarih 2...	2	Elektrik kablosu	1,00
- 24.04.2011 19:23:21	Hizmet faturaşı 00000003 İlk tarih 2...	3	Flyback transformér GoldStar	13,50
24.04.2011 19:23:21	Hizmet faturaşı 00000003 İlk tarih 2...	4	Transistor Philips 2N2369	0,30

Resim 11.18. 3 nolu hizmet furasının hareketleri

Sorular

- ✓ *Evrakı birden fazla kayıt tablosuna kaydetmeye ne gerek var?*
- ✓ *Evrak kaydetme işleyicisinde birden fazla kayıt tabluna kaydetme nasıl sağlanabilir?*
- ✓ *Hareket yapılandırıcısı kullanmadan evrak kayıt hareketleri nasıl oluşturulabilir?*
- ✓ *Kaynak kodu kullanarak evrak hareketleri nasıl oluşturulabilir ve kayıt tablosuna nasıl kaydedilebilir?*
- ✓ *Evrak formuna yeni öznitelik nasıl eklenebilir?*

DERS 12

Akışlar Birikim Kayıt Tabloları

Süre

Dersin tahmin süresi – 40 dakika

Bir tane daha kayıt tablosu ne için oluşturuyoruz	260
Akışlar birikim kayıt tablosu nedir	260
Akışlar birikim kayıt tablosunu ekleme	261
Hizmet faturası evrakının üç kayıt tablosuna veri kaydetme	263
Sorular	268

Bu derste diğer birikim kayıt tablosunun türü ile tanışacağız – akışlar birikim kayıt tablosu.

Birikim kayıt tablosunun kaynak ve özniteliği seçerken bazı önemli prensiplerden bahsedilecektir.

Bir tane akışlar birikim kayıt tablosunu oluşturuktan sonra onu evrakin hareketlerine bağlarız.

Bir tane daha kayıt tablosu ne için oluşturuyoruz

HizmetFaturası evrakımızın çalışma biçimini inceleyelim.

Şu ana kadar sadece malzemeleri içeren evrak tablo bölüm satırları için kayıt tablo hareketlerini oluşturmuştuk. Evrakta bulunan hizmetleri ile ilgili herhangi bir hesap tutulmuyordu.

Hizmetlerin ne kadar olduğunu ve ne kadar kaldığını hesaplamak mantıksızdır. Hizmetler ile ilgili sadece belli bir dönem içinde kaç kere hizmet verildiğini ve tutar ne olduğunu hesaplamak önemlidir.

Bunun dışında aşağıdaki bilgiler bizim için önemlidir;

- Verilen hizmetler nedir (hizmet raporu oluşturmak için),
- Hizmetler hangi müşteriye verilmiş (önceden alınan hizmet hacmine göre indirim hesaplamak için),
- Hizmeti hangi usta vermiş (maası hesaplamak için).

Belli ki mevcut olan kayıt tabloları bu görevleri yerine getirmek için uygun değildir.

B yüzden bu bilgiyi elde edebilmek için bir tane daha kayıt tablosunu oluşturacağız – **Satışlar** akışlar birikim kayıt tablosu.

Akışlar birikim kayıt tablosu nedir

StokBakiyesi ve StokMaliyeti kayıt tabloları oluştururken, 1C:İşletme var olan birikim kayıt tablo türlerinden bahsetmemiştik. Şimdi türleri hakkında biraz bahsedelim.

Birikim kayıt tabloları *bakiyeler kayıt tablosu* veya *akışlar kayıt tablosu* olabilir.

Konfigürasyonumuzda var olan StokBakiyesi ve StokMaliyeti birikim kayıt tabloları bakiyeler kayıt tablolarıdır.

Hatırlıyorsanız, *Malzemeler* raporu oluşturduğumuzda soru yapılandırıcısında böyle kayıt tabloları için sistem 3 tane sanal tablo oluşturulmaktadır; bakiyeler tablosu, akışlar tablosu ve birleşik bakiye ve akışlar tablosu.

Akışlar birikim kayıt tablosu bildiğimiz bakiyeler birikim kayıt tablosuna çok benzer, fakat akışlar kayıt tablosu için “bakiye” kavramı anlamsızdır. Akışlar birikim kayıt tablosu sadece akışları biriktirir, bakiyeler hesabını

tutmaz. Bu yüzden sistem sadece bir tane sanal tablo oluşturacaktır – akış bilgilerini içeren bir tablo.

Birikim kayıt tabloları oluşturmanın önemli bir özelliğinden bahsetmek gereklidir. Akışlar birikim kayıt tabloları oluştururken, kayıt tablo boyutları ne olmak gerektiği ile ilgili pek problem yoktur. Boyut olarak gerekli olan herhangi bir şey tanımlayabiliriz.

Bakiyeleri hesaplayan birikim kayıt tablo ile ilgiliambaşka bir durum vardır. Bakiyeler birikim kayıt tablosu için boyut tanımlarken, bu boyut ile ilgili hareketlerin çift yönlü olmaları şarttır – giren ve çıkan. Böylece boyut olarak sadece iki tarafa hareketi oluşturan bilgi atanabilir.

Örneğin, malzeme ve depo detayları ile stok takibi yapılmaktadır. Belli ki malzeme ve depo boyut olarak atanabilir, çünkü giriş ve çıkış hareketleri somut malzeme ve somut depo belirleyerek girilir. Bu durumda eğer stok takibinde tedarikçi bilgileri takip etmek istediğiinde artık işletmenin hesap tutma şemasına göre hareket etmek gereklidir.

Büyük ihtimalde, ürün alım aşamasında tedarikçi belirlenecek, fakat ürün çıktığında tedarikçi belirlenme olasılığı çok düşük.

Bu yüzden tedarikçi boyut olarak değil, öznitelik olarak tanımlamak gereklidir.

Ürün çıkışında tedarikçi kesin belirlendiği durumda, tedarikçi boyut olarak tanımlamak doğru olabilir.

Bakiyeler birikim kayıt tablolarının tüm boyutlarına göre kaynaklar çift yönlü hareketleri yapmalıdır – giriş ve çıkış. Sadece giriş veya sadece çıkış hareketlerinde yer alan boyutlar tanımlanmamalıdır.

Bu birikim kayıt tablo oluşturma prensibini uymaması sistem kaynaklarının verimli kullanılmamasına ve dolayısıyla sistemin daha yavaş çalışmasına ve performans seviyesinin düşmesine sebep olur.

Kayıt tablo öznitelikleri için bu prensip geçerli değildir. Özniteliklere göre kaynaklar sadece giriş veya sadece çıkış hareketlerini oluşturabilir.

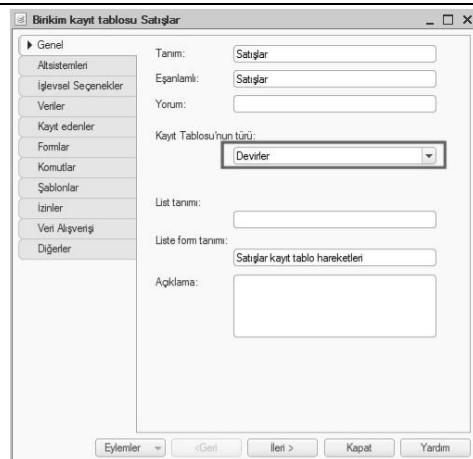
Akışlar birikim kayıt tablosunu ekleme

Tasarımcı ortamında

Tasarımcı ortamını açıp yeni birikim kayıt tablosu konfigürasyon nesnesini oluşturalım.

Kayıt tablosunun **Satışlar** olarak adlandırılmış ve kayıt tablo türü olarak **Akışlar** seçelim.

Bunun dışında Liste tanımı alanına *Satışlar kayıt tablo hareketleri* dolduralım. Bu başlık kayıt tablo liste formunda yansıtılacaktır (Resim 12.1).



Resim 12.1. Akışlar birikim kayıt tablosunu ekleme

Alt sistemleri sekmesinde bu kayıt tablosunun Muhasebe, Stok ve Hizmetler alt sistemlerinde yansıtılacağını tanımlayalım.

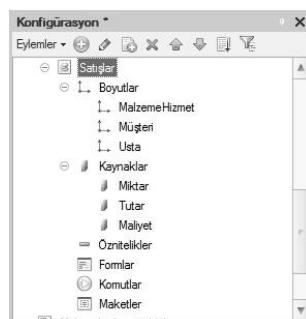
Veriler sekmesinde kayıt tablo boyutlarını oluşturalım;

- **MalzemeHizmet**, tip: **CatalogRef.MalzemeHizmet**,
- **Müşteri**, tür **CatalogRef.Müşteriler**,
- **Usta**, tür: **CatalogRef.Personeller**.

Kayıt tablosunun üç tane kaynak olacak;

- **Miktar**, tür: **Number**, Uzunluk: **15**, Ondalık: **3**,
- **Tutar**, tür: **Number**, Uzunluk: **15**, Ondalık: **2**,
- **Maliyet**, tür: **Number**, Uzunluk: **15**, Ondalık: **2**.

Sonuç olarak Satışlar kayıt tablosu aşağıdaki gibi görünmelidir (Resim 12.2).



Resim 12.2. "Satışlar" akışlar birikim kayıt tablosu

Şimdi kullanıcı arayüzümüzün Muhasebe, Stok ve Hizmetler bölümleri altında akışlar birikim kayıt tablo listesini izleme komutlarının gösterilmesi için arayüz ayarlarını düzeltelim.

Konfigürasyon nesne ağacında *Alt sistemleri* grubu işaretleyerek sağ tuş menüsünden *Tüm altsistemler* maddesini seçelim.

Ekrana gelen pencerenin sol tarafında *Alt sistemler* listesinde *Muhasebe* alt sistemini seçelim.

Sağ tarafında bulunan Arayüz listesinde seçilen alt sistemi ile ilgili tüm komutlar listelenir.

Araç çubuğu. Normal grubunda *Satışlar* komut için görünürlüğünü işaretleyelim ve fare ile **Araç çubuğu. Ayrıca bakınız** grubuna taşıyalım.

Aynı şekilde Hizmetler ve Stok alt sistemler için **Araç çubuğu. Normal** grubunda *Satışlar* komutunun görünürlüğünü işaretleyip **Araç çubuğu. Ayrıca bakınız** grubuna taşıyalım.

Hizmet faturası evrakının üç kayıt tablosuna veri kaydetme

Bu bölümde biz ilk önce HizmetFaturası evrakının kaydetme prosedürüne değiştirip 1C:İşletme ortamında tüm hizmet faturaları tekrar kaydedeceğiz. Böylece hizmet faturalarındaki veri tanımladığımız algoritmaya göre üçüncü kayıt tablomuza da kaydedilecektir.

Tasarımcı ortamında

HizmetFatura evrakı konfigürasyon nesnesinin düzeltme penceresinde *Hareketler* sekmesine geçelim.

Kayıt tablo listesinde evrakin *Satışlar* kayıt tablosuna da kayıt yapacağını belirtelim.

Digerler sekmesine geçip nesne modülü açalım. Bunu yapmak için *Nesne modülü* butonuna basmak yeterlidir.

Posting prosedürü açalım.

Döngünün en altta, EndDo satırından önce *Satışlar* kayıt tablosuna kayıt yapacak kod satırlarını ekleyelim (Liste 12.1).

Liste 12.1. “HizmetFaturası” kaydetme prosedürü (parça)

```
// kayıt tablosu Satışlar
Record = RegisterRecords.Satışlar.Add();
Record.Period = Date;
Record.Malzeme = CurRowMalzemeler.Malzeme;
Record.Müşteri = Müşteri;
Record.Usta = Usta;
Record.Miktar = CurRowMalzemeler.Miktar;
```

```
Record.Tutar = CurRowMalzemeler.Tutar;
```

```
Record.Maliyet = CurRowMalzemeler.Miktar * CurRowMalzemeler.Maliyet;
```

Döngü başlamadan önce kayıt tablosunun kayıt seti için *Write* özelliği için *True* değeri belirleyelim.

Sonuç olarak Posting prosedürü aşağıdaki gibi görünür (Liste 12.2).

Liste 12.2. “HizmetFaturası” kaydetme prosedürü

```
Procedure Posting(Cancel, Mode)
```

```
RegisterRecords.StokBakiyesi.Write = True;
```

```
RegisterRecords.StokMaliyeti.Write = True;
```

```
RegisterRecords.Satışlar.Write = True;
```

```
For Each CurRowMalzemeler In Malzemeler Do
```

```
    If CurRowMalzemeler.Malzeme.HizmetTürü =
```

```
Enums.MalzemeHizmetTürleri.Malzeme Then
```

```
// kayıt tablosu StokBakiyesi Expense
```

```
Record = RegisterRecords.StokBakiyesi.Add();
```

```
Record.RecordType =
```

```
AccumulationRecordType.Expense;
```

```
Record.Period = Date;
```

```
Record.Malzeme = CurRowMalzemeler.Malzeme;
```

```
Record.Depo = Depo;
```

```
Record.Miktar = CurRowMalzemeler.Miktar;
```

```
// kayıt tablosu StokMaliyeti Expense
```

```
Record = RegisterRecords.StokMaliyeti.Add();
```

```
Record.RecordType =
```

```
AccumulationRecordType.Expense;
```

```
Record.Period = Date;
```

```
Record.Malzeme = CurRowMalzemeler.Malzeme;
```

```
Record.Maliyet = CurRowMalzemeler.Miktar *
```

```
CurRowMalzemeler.Maliyet;
```

```
EndIf;
```

```
// kayıt tablosu Satışlar
```

```
Record = RegisterRecords.Satışlar.Add();
```

```
Record.Period = Date;
```

```
Record.MalzemeHizmet = CurRowMalzemeler.Malzeme;
```

```
Record.Müşteri = Müşteri;
```

```
Record.Usta = Usta;
```

```
Record.Miktar = CurRowMalzemeler.Miktar;
```

```
Record.Tutar = CurRowMalzemeler.Tutar;
```

```
Record.Maliyet      =  CurRowMalzemeler.Miktar  *
CurRowMalzemeler.Maliyet;
```

```
EndDo;
```

```
EndProcedure
```

Eklenen kod parçaları önceden açıklamıştık.

Eklediğimiz metinde bir tek fark var; akışlar birikim kayıt tablosunda RecordType özelliği eksiktir, çünkü dediğimiz gibi hareket yönü (giren veya çıkan) sadece bakiye hesaplanmasında gereklidir. Akışlar kayıt tablosu durumunda bizi sadece kayıt tablo kaynağına kaydedilecek değer önemlidir.

Birde fark ettiyseñiz, Satışlar kayıt tablosunda hareketi oluşturan kod sadece malzemeler için uygulanacak koşul dışında bulunur. Çünkü satışlar bilgileri sadece malzemeler için değil, hem malzeme ve hizmetler için kaydedilecektir.

Son olarak evrak formundan evrakin yaptığı kayıt tablo hareketlerine direkt ulaşabilmek için evrak formunun arayüzü düzeltelim.

Bunu yapmak için *HizmetFaturası* evrakinin *DocumentForm*'u açalım.

Sol üst bölümünde *Komut arayüz* sekmesine geçelim.

Araç çubuğu bölümünde Git grubunda Satışlar kayıt tablosunun açmak içim bir komut görüyoruz. Bu komutun görünürüğünü işaretleyelim.

1C: İşletme ortamında

1C:İşletme ortamında tüm hizmet faturaları yeniden kaydetmek ve Satışlar kayıt tablosuna doğru kayıt yapıldığını kontrol etmek gereklidir.

1C:İşletme sistemini hata ayıklama biçiminde çalıştırıalım ve sıra ile tüm *Hizmet fatura* evraklarını açalım.

Kaydet tıklayalım ve evrakin Satışlar kayıt tablo hareketlerine geçelim. Hareketler aşağıdaki gibi görülmeliidir (Resim 12.3a, 12.3b, 12.4a, 12.4b, 12.5a, 12.5b).

Satışlar kayıt tablo hareketleri				
Dönem	Kayıt eden	Satır numarası	Malzeme hizmet	Müşteri
• 28.02.2011 12:00:00	Hizmet faturası 000000...	1	Su bağlama	Ahmet Bey
• 28.02.2011 12:00:00	Hizmet faturası 000000...	2	Lastik hortun	Ahmet Bey

Resim 12.3a. "1 nolu Hizmet faturası" evrak hareketleri

Satışlar kayıt tablo hareketleri				
Usta	Miktar	Tutar	Maliyet	
Mehmet Uzman	1.000	40.00		
Mehmet Uzman	1.000	7.50	5.00	

Resim 12.3b. “1 nolu Hizmet faturası” evrak hareketleri

Satışlar kayıt tablo hareketleri				
Dönem	Kayıt eden	Satır numarası	Malzeme hizmet	Müşteri
24.04.2011 21:28:40	Hizmet faturası 000000...	1	Yabancı marka televiz...	Ahmet Bey
24.04.2011 21:28:40	Hizmet faturası 000000...	2	Flyback transforme... Sa...	Ahmet Bey

Resim 12.4a. “1 nolu Hizmet furası” evrak hareketleri

Satışlar kayıt tablo hareketleri				
Usta	Miktar	Tutar	Maliyet	
Ali İşçi	1.000	40,00		
Ali İşçi	1.000	45,00	30,00	

Resim 12.4b. “1 nolu Hizmet furası” evrak hareketleri

Satışlar kayıt tablo hareketleri				
Dönem	Kayıt eden	Satır numarası	Malzeme hizmet	Müşteri
24.04.2011 21:29:52	Hizmet furası 000000...	1	Elektrik bağlama	Mustafa Bey
24.04.2011 21:29:52	Hizmet furası 000000...	2	Lastikhortun	Mustafa Bey
24.04.2011 21:29:52	Hizmet furası 000000...	3	Elektrik kablo	Mustafa Bey
24.04.2011 21:29:52	Hizmet furası 000000...	4	Yerli marka televizyon t...	Mustafa Bey
24.04.2011 21:29:52	Hizmet furası 000000...	5	Flyback transforme... G...	Mustafa Bey
24.04.2011 21:29:52	Hizmet furası 000000...	6	Transistor Philips 2N23...	Mustafa Bey

Resim 12.5a. “1 nolu Hizmet furası” evrak hareketleri

Satışlar kayıt tablo hareketleri			
(+)	Bul...	Yönlendir	Tüm eylemler ▾ ?
Usta	Miktar	Tutar	Maliyet
Ali İşçi	1.000	40,00	
Ali İşçi	2.000	15,00	10,00
Ali İşçi	1.000	1,50	1,00
Ali İşçi	1.000	30,00	
Ali İşçi	1.000	20,00	13,50
Ali İşçi	2.000	0,70	0,30

Resim 12.5b. “1 nolu Hizmet faturası” evrak hareketleri

Şimdi “Aspirin usta” işletmenin analizi yapabilmek için tüm gerekli bilgi mevcuttur.

Bir sonraki derste işletme durumunu yansitan birkaç rapor oluşturacağız.

Sorular

- ✓ *Akışlar birikim kayıt tablosu nedir?*
- ✓ *Bakiye ve akışlar birikim kayıt tabloları arasındaki fark nedir?*
- ✓ *Birikim kayıt tabloları oluştururken boyutlar ve öznitelikler nasıl seçilir?*

DERS 13

Raporlar

Süre

Dersin tahmin süresi – 4 saat 30 dakika

“Teorik bilgi”	270
Veriye erişim biçimleri	270
Sorgular ile çalışma	271
Sorgu veri kaynakları	271
Sorgu dili	272
Veri inşa edilme şeması	273
Bir tablodan veri alma	276
İki tablodan veri alma	284
Seçilen dönem içinde gün detayı ile veri dökümü	307
Periyodik bilgi kayıt tablosundan güncel veri alımı	322
Raporda hesaplanan alanı kullanma	329
Verileri tabloda gösterilmesi	333
Sorular	339

1C:İşletme platformunun en önemli araçlarından biri olan, **Veri inşa edilme şeması** ile tanıma zamanı geldi. Bu derste, konfigürasyonumuzda kullanılacak olan birkaç rapor oluşturacağız. Geliştirme aşamasında veri inşa edilme mekanizmasının sağladığı genel imkanları inceleyeceğiz.

Her rapor, belli mantığa göre gruplandırılmış ve sıralanmış veri setidir. Veri inşa edilme sistemi, farklı kaynaklardan veri alınmasından başlayarak, kullanıcı için rahat bir şekilde veriyi sunmak için gereken tüm eylemleri gerçekleştirebilir ve esnek mekanizmadır.

Genelde rapor için gerekli kaynak veriler veritabanında bulunur. Veri inşa edilme şemasından hangi veriyi nereden almak gerektiğini belirtmek için 1C:İşletme sisteminin sorgu dili kullanılmaktadır.

Rapor geliştirme aşamasında, kullanıcı raporu hemen çalıştırabilmesi için, raporun standart ayarları belirlenebilir. Bunun dışında kullanıcı kendisi raporun ayarları değiştirebilir ve raporu oluşturabilir durumdadır. Bu durumda veri inşa edilme şeması, kullanıcı belirlediği ayarlara göre, farklı bir sorgu oluşturup, farklı şekilde sonuç veriyi ekrana getirir.

İlk olarak bir 1C:İşletme sisteminin sorgu dili ve veri inşa edilme şeması ile ilgili genel bilgilerinden bahsedeceğiz.

Ondan sonra basit örnekleri uygulayarak veri inşa edilme şemasını kullanmaya başlarız.

“Teorik bilgi”

Veriye erişim biçimleri

1C:İşletme sistemi veritabanında bulunan veriye iki erişme biçimini destekler;

- Nesnesel (okumak ve kayıt etmek için),
- Tablosal (okumak için).

Nesnesel erişim biçimini kaynak dil nesneleri ile gerçekleştirilir.

Önceki derslerde onlardan birkaç tanesini kullanmıştır.

Nesnesel erişim biçiminin önemli özelliği şudur; herhangi bir kaynak dil nesnesine başvurduğumuzda, veritabanında bulunan belli bir veri setine, tek parça olarak başvuruyoruz.

Örneğin, *DocumentObject.HizmetFaturası* nesnesi *Hizmet faturası* evrakinin tüm öznitelik ve tablo bölüm değerlerini içerir.

Nesnesel teknik, nesne bütünlüğünün korunmasını, nesne önbelleklemesini, gerekli olay işleyicilerinin çağırmasını vs. sağlar.

Tablosal erişim biçimini, 1C:İşletme sisteminde, sorgu dili ile oluşturulan veritabanı sorguları ile gerçekleştirilir.

Bu özelliği kullanarak geliştirme uzmanı, gerekli veriyi içeren veritabanında bulunan tablolarının belli sütunları ile çalışma olanağına sahip olur.

Tablosal erişim biçimini belli koşuları uygulayıp (filtreleme, gruplandırma, sıralama, birden fazla seçimlerin birleştirme, toplam hesaplama vs.) veritabanından veri almak için kullanılır. Tablosal erişim biçimini veritabanında bulunan büyük veri hacimleri ile çalışmak ve belli kriterlere göre gerekli veriyi elde etmek için optimize edilmiştir.

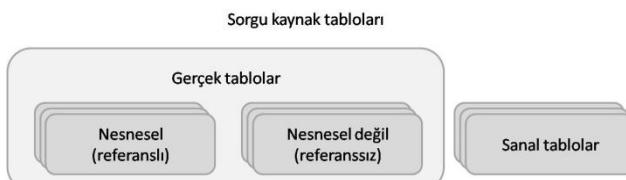
Sorgular ile çalışma

Sorgular ile çalışmak için **Query** kaynak kod nesnesi kullanılmaktadır. Bu kaynak kod nesnesi veritabanı alanlarında bulunan verileri belli kriterlere göre alma olanağını sağlar.

Sorgu veri kaynakları

Sorgu kaynak verileri tablolardan alır. Bu tablolar, kullanıcıya analiz için rahat bir şekilde, veritabanında bulunan gerçek tabloları sunar.

Sorgunun çalıştığı tüm tabloları iki grup olarak bölünebilir; gerçek tablolar ve sanal tablolardır (Resim 13.1).



Resim 13.1. Sorgu tabloları

Sorgu için ulaşılabilen tabloları ve özelliklerini sözdizim sihirbazında bakılabilir (*Working with queries – Query tables*).

Gerçek tabloların farklı tarafı şudur; gerçek tablolar veritabanında bulunan gerçek tabloları temsil eder.

Örneğin, *Catalogs.Müşteriler* tablosu gerçek bir tablodur. Bu tablo Müşteriler kart listesini tanımlar. *AccumulationRegisters.StokBakiyesi* tablosu da bir gerçek tablodur, StokBakiyesi birikim kayıt tablosunun gerçek tablosunun içerir.

Sanal tablolar genelde veritabanında bulunan birden fazla gerçek tablosundan oluşturulur.

Örneği *AccumulationRegisters.StokBakiyesi.BalanceAndTurnovers* tablosu sanal bir tablodur. *Stok Bakiyesi* birikim kayıt tablosunun birden fazla tablosundan oluşur.

Bazen sanal tablolar bir tane tablodan da oluşturulabilir (örneğin, *Fiyatlar.SliceLast* sanal tablosu, *Fiyatlar* bilgi kayıt tablosundan oluşturulur).

Fakat tüm sanal tablolar için, tablolardan bulunacak veriler ile ilgili parametreler tanımlanabilir. Parametreler her tablo için farklı olabilir, parametreler sanal tabloda bulunan veri türlerine göre değişir.

Gerçek tablolar *nesnesel (referanslı)* ve *nesnesel değil (referanssız)* olarak iki gruba ayrılır.

Nesnesel (referanslı) tablarda referanslı veri türleri olan veri bulunur (kart listeleri, evrakları, karakteristik tür planları vs.). *Nesnesel olmayan* tablarda – diğer veri türleri olan veriler (sabit değerler, kayıt tabloları vs.).

Nesnesel tabloların farkı şudur; geçerli kayda referansı olan **Ref** alanına sahiptir. Bunun dışında bu tablolar için kullanıcı ekranı çağrırlabilen. Bu tablolar hiyerarşik olabilir ve iç içe tabloları (tablo bölümleri) içerebilir.

Sorgu dili

Kaynak tablolarından veri alma algoritması özel dil ile tanımlanmaktadır – *Sorgu dili*.

Sorgu metni birden fazla parçadan oluşabilir;

- Sorgu tanımı,
- Sorgu bireştirmesi,
- Sonuç sıralaması,
- Otomatik sıralama,
- Sonuç tanımı.

Sorgu tanımı – sorgunun zorunlu olan kısımdir. Diğer kısımlar gerektiğinde eklenebilir.

Sorgu tanımı, veri kaynağını, seçim alanlarını, grupları vs. tanımlar.

Sorgu bireştirmesi, birden fazla sorgunun sonuçları nasıl birleştirileceğini tanımlar.

Sonuç sıralaması, sorgu sonuç satırlarının sıralama düzenini belirler.

Otomatik sıralama, sorgu sonuç satırlarının otomatik olarak sıralama özelliğini tanımlar.

Sonuç tanımı, sorguda hangi sonuçlarını hesaplamak gerektiğini ve sonucu nasıl gruplandırmak gerektiğini tanımlar.

Veri inşa edilme sisteminde, veri kaynaklarını tanımlamak için sorgu dili kullanıldığında, sorgu sonuç tanımlama sekmesi kullanılmıyor. Çünkü veri inşa edilme şeması, kullanıcının veya geliştiricinin tanımladığı ayarlara göre sonuç tanımlamasını kendisi yapar.

Sorgu dilinin çeşitli sözdizim yapılarının kullanımı, Tasarımcıda bulunan Sözdizim sihirbazında detaylı bir şekilde tanımlanmıştır; *Yardım – Yardım içeriği – 1C:İşletme – Script - Working with Queries*.

Sorgu dili ile daha detaylı bir şekilde rapor oluşturma aşamasında açıklanacaktır.

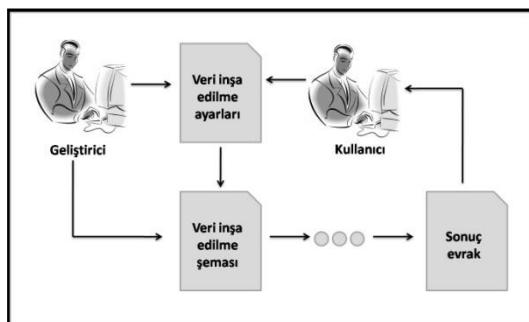
Sorguları “manüel” olarak yazmayacağız. Veri inşa edilme sistemi ile geliştirilen raporlar için çoğu durumda sorgular sorgu yapılandırıcı ile oluşturulur.

Bu dersteki amacımız – ilerde sorguları değiştirebilmek için bu sorguların metinlerini anlamaktır.

Veri inşa edilme şeması

Veri inşa edilme sistemi 1C: İşletme sisteminde serbest bir şekilde raporları oluşturmak için kullanılır. Veri inşa edilme sistemi birkaç ana parçadan oluşur.

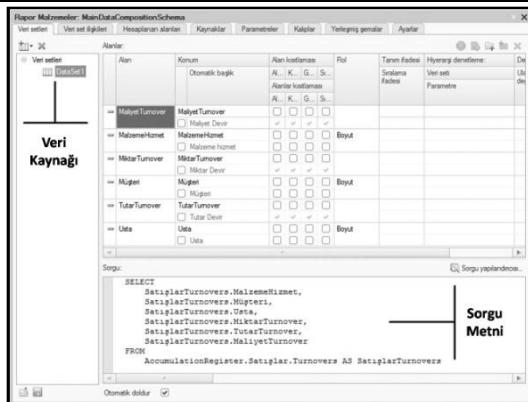
Rapor oluşturmak için gerekli olan kaynak verileri *veri inşa edilme şeması* içerir. Bu veri setleri ve veri setleri ile çalışma metotları (Resim 12.3).



Resim 13.2. Veri inşa edilme sistemi ile genel çalışma şeması

Geliştirici veri inşa edilme şemasını oluşturur. Veri inşa edilme şemasında sorgu metnini, veri setlerini, aralarındaki ilişkileri, ulaşılabilir alanları, veri alma parametreleri tanımlar ve veri inşa edilme şemasının görünüm ayarları tanımlar – rapor yapısı, dekor şablonu vs.

Örneğin, veri inşa edilme şeması aşağıdaki veri setini içerebilir (Resim 13.3).



Resim 13.3. Veri inşa edilme şema örneği (veri seti ve veri setini kullanan sorgu)

Resimde, veri kaynağını, sorgu metnini ve sorgu ile seçilen alanları içeren veri inşa edilme şemasının yapılandırıcısının penceresi gösterilmiştir.

Kullanıcının aldığı veri inşa edile sistem raporu bu sadece sorgu kriterlerine uygun olan kayıtlar tablosu değil.

Veri inşa edilme sistem raporu kapsamlı hiyerarşik yapıya sahiptir, gruplar, tablolar ve grafikler farklı öğelerden oluşabilir.

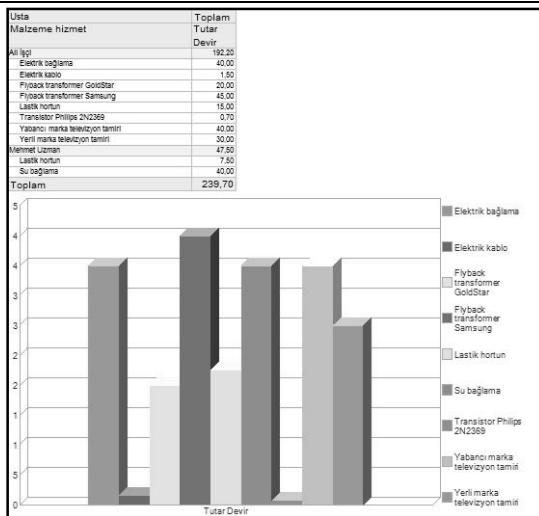
Kullanıcı var olan raporu yapısını değiştirebildiği gibi, sıfırdan da rapor yapısını oluşturabilen durumdadır. Kullanıcı gerekli olan filtreyi, rapor yapısının öğe dekorunu ayarlayabilir, raporun her öğesi için ayrıntılı rapor alabilir vs.

Örneğin, bir tablo ve bir grafikten oluşan rapor yapısı tanımlanabilir.



Resim 13.4. Var olabilir rapor yapısı

Bu durumda oluşturulan rapor aşağıdaki gibi görünecektir.



Resim 13.5. Rapor örneği

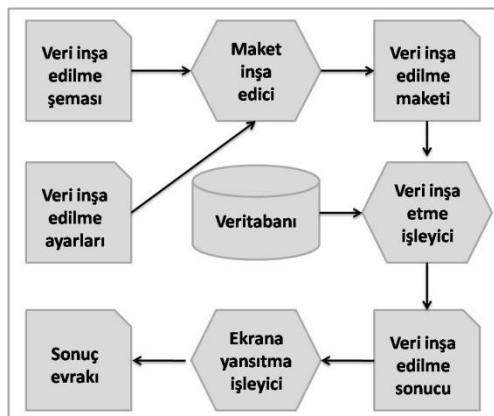
Bu raporda, *SatışlarTurnovers* birikim kayıt tablosundan alınan müşteri ve müşterilere verilen hizmetler ile ilgili kayıtlar yansıtılır. Bu kayıtlar hizmet veren ustalara göre gruplandırılmış.

Bölüm başında dediğimiz gibi veri inşa edilme sistemi birden fazla nesneden oluşturulmaktadır. Rapor oluşturma ve uygulanma aşamasında, veriler sırayla bir veri inşa edilme sistem nesnesinden diğer veri inşa edilme nesnesine aktarılmaktadır. Sonuç olarak kullanıcıya görsel bir şekilde rapor gösterilir.

Bu nesne çalışma algoritması aşağıdaki gibidir;

- Geliştirici veri *inşa edilme şemasını* ve *varsayılan ayaları* oluşturur. Genelde tek bir veri inşa edilme şeması üzerinde çok sayıda rapor oluşturulabilir. Geliştiricinin tanımladığı veya kullanıcının değiştirdiği veri inşa edilme ayarları nasıl bir rapor elde edileceğini tanımlar.
- Veri inşa edilme şeması ve var olan ayarlara göre *şablon inşa edici* bir *şablon* oluşturur. Bu rapor uygulanmasının hazırlama aşaması. Veri inşa edilme şablonu veri inşa edilme işleyicisi için hazır uygulama görevidir. O gerekli sorguları, rapor alan şablonlarını vs. içerir.
- *Veri inşa edilme işleyicisi* veri inşa edilme şablonuna göre veritabanından veriyi toplar, işler ve dekore eder.
- *Veri inşa edilme sonucu* ekrana yansıtma işleyicisi tarafından işlenir ve sonuç olara kullanıcı ekranına rapor bilgilerini içeren bir tablo belgesi gelir.

Bu çalışma düzenini aşağıdaki şema olarak tanımlanabilir (Resim 13.6).



Resim 13.6. Veri inşa edilme sisteminin çalışma şeması

Bir tablodan veri alma

Veri inşa edilme sistemini kullanarak “*Hizmet fatura kayıt defteri*” raporu oluşturalım.

Bu rapor örneği ile bir tane veritabanı tablosundan veriyi nasıl alındığını ve sıralama nasıl yapıldığını göstereceğiz. Bir de var olan bir raporda detaylı bilgi nasıl alındığını öğreneceğiz.

Bu rapor veritabanında var olan *HizmetFaturası* evrakları gösterecek. Sıralama ise evrak tarihine ve numarasına göre yapılacaktır (Resim 13.7).

Evrap	Depo	Usta	Müşteri
Hizmet faturası 00000001 ilk tarih 28.02.2011 12:00:00	Genel depo	Mehmet Uzman	Ahmet Bey
Hizmet faturası 00000002 ilk tarih 24.04.2011 21:28:40	Genel depo	Ali İşçi	Ahmet Bey
Hizmet faturası 00000003 ilk tarih 24.04.2011 21:29:52	Genel depo	Ali İşçi	Mustafa Bey

Resim 13.7. Rapor sonucu

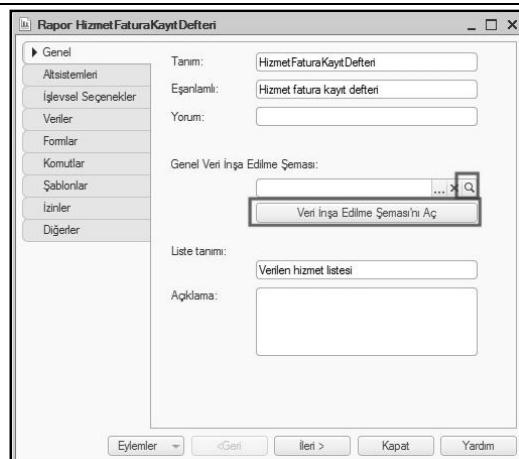
Tasarımcı ortamında

Tasarımcıda **Rapor** konfigürasyon nesnesini ekleyelim.

Genel sekmesinde raporun adı tanımlayalım – **Hizmet fatura kayıt defteri**.

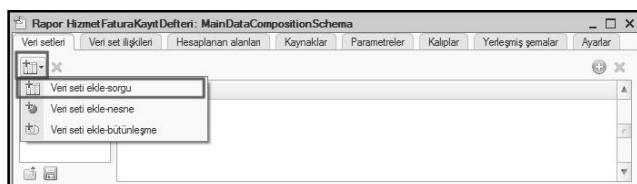
Raporu program arayüzünde tanımlamak için oluşturulan öğenin *Liste tanımı*'nı *Verilen hizmet listesi* olarak belirleyelim.

Rapor için veri inşa edilme şemasını oluşturalım. Bunu yapmak için *Veri inşa edilme şemasını aç* butonuna veya butonun sağ tarafında bulunan büyütçe simgesine tıklayalım (Resim 13.7).



Resim 13.8. Raporun genel özellikleri

Ekrana gelen şablon inşa etme sihirbazında **Tamam** butona tıklayalım. Veri inşa edilme yapılandırıcısında **Veri seti – sorgu** oluşturulmuş (Resim 13.9).



Resim 13.9. Veri seti – sorgu oluşturmazı

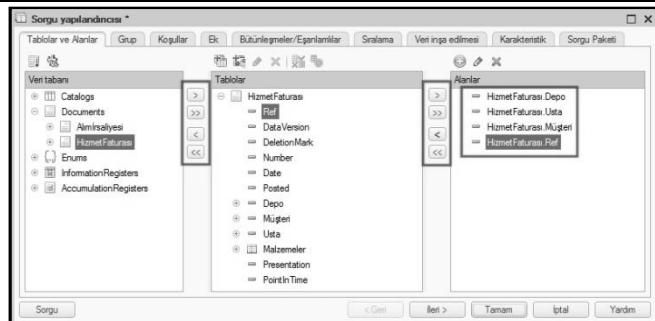
Veri seti için sorgu

Sorgu yapılandırıcı... butonuna basarak, sorgu yapılandırıcısını açalım.

Sorgunun veri kaynağı olarak *HizmetFaturası* evrakinin nesnesel (referanslı) tablosunu seçelim.

Bu tablodan aşağıdaki alanları seçelim (Resim 13.10);

- Depo,
- Usta,
- Müşteri,
- Ref.



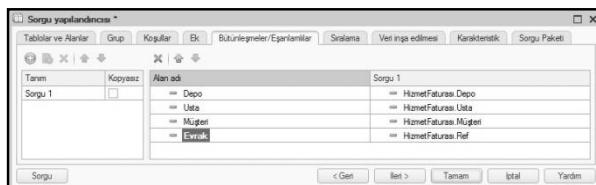
Resim 13.10. Sorgu için seçilen alanlar.

Not

Seçilen öğeleri çift tıklayarak veya sürükleyerek diğer listeye taşınamılır. Veya **>**, **>>**, **<**, **<<** butonları kullanılabilir.

Alan eşanımlıları

Bütünleşmeler/Eşanımlılar sekmesine geçip Ref alanı için Evrak eşanımlısını belirleyelim (Resim 13.11).



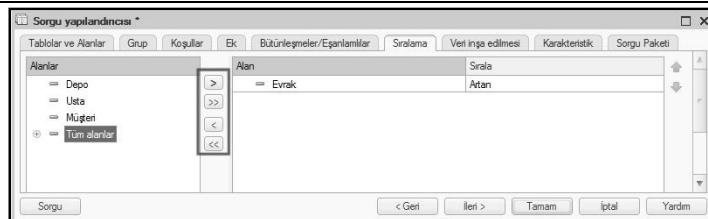
Resim 13.11. Sorgu alanları için eşanımlısını tanımlama

Tavsiye

Alan adları sorguda değiştirmek daha iyidir, çünkü alan adı sorguda değiştirilirse veri inşa edilme şemasında üç sütunda değiştirilmiş olur: Alan, Yol ve Başlık.

Kayıt sıralaması

Bundan sonra Sıralama sekmesine geçip sorgu sonucu *Evrak* alan değerine göre sıralanacağını belirleyelim (resim 13.12).



Resim 13.12. Sorgudaki kayıt sırası

Sorgu metni analizi

Tamam butonuna basalım ve sorgu yapılandırıcısı nasıl bir sorgu oluşturacağına bakalım (Liste 13.1).

Liste 13.1. Sorgu metni

```

SELECT
    HizmetFaturası.Depo,
    HizmetFaturası.Usta,
    HizmetFaturası.Müşteri,
    HizmetFaturası.Ref AS Evrap
FROM
    Document.HizmetFaturası AS HizmetFaturası
ORDER BY
    Evrap
  
```

Bahsettiğimiz gibi sorgu metni *sorgu tanımlama* kısmı ile başlar (Liste 13.2).

Liste 13.2. Sorgu tanımı

```

SELECT
    HizmetFaturası.Depo,
    HizmetFaturası.Usta,
    HizmetFaturası.Müşteri,
    HizmetFaturası.Ref AS Evrap
FROM
    Document.HizmetFaturası AS HizmetFaturası
  
```

Sorgu tanımı her zaman SELECT anahtar kelimesi ile başlar.

Ondan sonra *seçilen alanlar listesi* gelir. Bu listede sorgu sonucunda yer alacak alanlar yer alır. Bu liste kendi alanlarını içerebildiği gibi, alan değerleri üzerinde hesaplanan ifadeleri de içerebilir.

FROM anahtar kelimesinden sonra *veri kaynağı* belirlenir – sorguda işlenen kaynak tabloları.

Örneğimizde bu *Document.HizmetFaturası* nesnesel (referanslı) tablosu.

AS anahtar kelimesinden sonra *veri kaynak eşanlamlısı* belirlenir.

Örneğimizde bu *HizmetFaturası*'dır. İlerde bu kaynağı eşanlamlısını kullanarak başvurulabilir.

Seçilen alanları tanımlama listesinde bunu görebiliriz (Liste 13.3).

Liste 13.3. Seçilen alanlarını tanımlama

```
SELECT
```

```
    HizmetFaturası.Depo,  
    HizmetFaturası.Usta,  
    HizmetFaturası.Müşteri,  
    HizmetFaturası.Ref AS Evrak
```

Seçilen alanlar da eşanlamlısına sahip olabilir. Örneğimizde bu *Ref* alanının *Evrak* eşanlamlısıdır.

Sorgu tanımlama kısımdan sonra örneğimizde sonuçlar sıralama kısmı geliyor (Liste 13.4).

Liste 13.3. Sorgu sonuç sıralaması

```
ORDER BY
```

```
    Evrak
```

ORDER BY ifadesi sorgu sonucundaki satırları sıralama imkanı veriyor. Bu ifadesinden sonra sıralama ifadeler yazılır, genelde sorgu sonucu hangi alanlara göre sıralayacağını ve sıralama biçimini tanımlanır.

Örneğimizde sıralama *Evrak* alanına göre yapılacaktır. Sıralama biçimini *Artan* olacak (sıralama biçimini belirlenmediği durumda, varsayılan değer olarak sıralama *Artan* olarak yapılmaktadır).

Sorgu metnini inceledikten sonra veri inşa edilme şemasının ayarlarına geçelim.

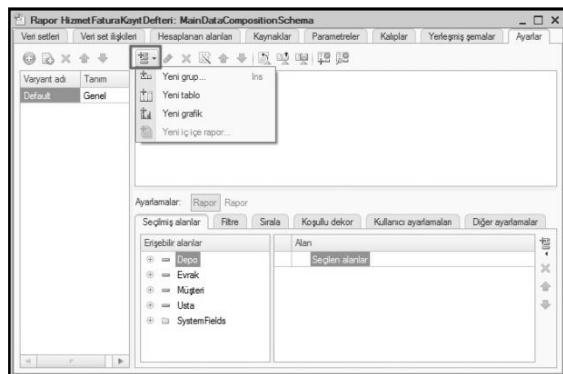
Ayarlar

Ayarlar sekmesine geçelim ve raporun görünüş şeklini tanımlayacak olan standart ayarları oluşturalım.

Raporun hiyerarşik yapısı üç ana öğelerin takımları olabilir.

- *Grup* – normal satır bazlı rapor görsüntülemek için kullanılır.
- *Tablo* – veriyi tablo şeklinde görsüntülemek için kullanılır.
- *Grafik* – veriyi grafik şeklinde görsüntülemek için kullanılır.

Yeni öğeyi (bizim örneğimizde grubu) eklemek için raporun hiyerarşik yapısında Rapor grubunu işaretleyip sağ tuş menüsünü çağırıralım veya Ins klavye tuşu kullanalım (Resim 13.13).



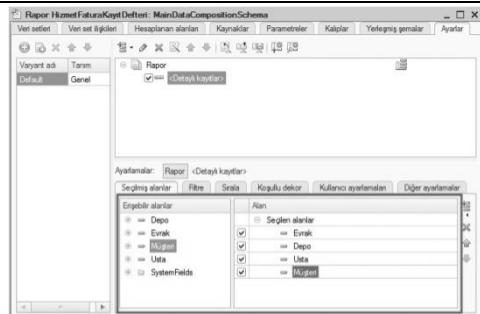
Resim 13.13. Rapor'a yeni grup ekleme

Grup alanını seçme penceresinde herhangi bir alan seçmeden **Tamam** butonuna basalım (böylece bu grupta veritabanından alınan detaylı kayıtların geleceğini tanımlıyoruz).

Rapor yapısında *Detaylı kayıtlar* grubu eklenir.

Seçilmiş alanlar sekmesinde ulaşılabilir alan listesinden raporda gösterilecek olan alanları taşıyalım;

- Evrak,
- Depo,
- Usta,
- Müşteri.



Resim 13.14. Rapor ayarları penceresi

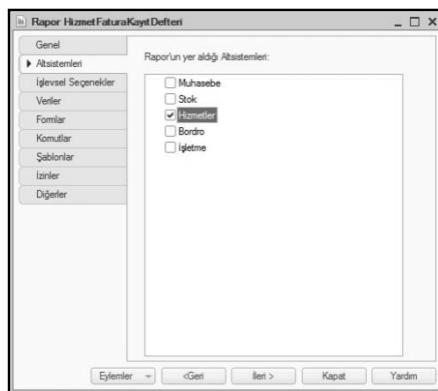
Not

Ulaşılabilir alanları seçilmiş alanlar listesine seçimi çift tuş ile, sürükleyerek veya Ekle butonuna basarak yapılabilir. Seçilmiş alan sırasını Üste taşı, Aşağıya taşı butonları ile yapılabilmektedir.

Sonuçta rapor ayarlama penceresi resimde gibi görünmelidir (Resim 13.14). Veri inşa edilme şemasının yapılandırıcı penceresini kapatıp *HizmetFaturaKayıtDefteri* konfigürasyon nesnesinin düzeltme penceresinin Alt sistemleri sekmesine geçip raporun ait olduğu alt sistemini tanımlayalım.

Listeden Hizmetler alt sistemini işaretleyelim.

Böylece raporu açma butonu Hizmetler bölümü altında bulunacaktır (Resim 13.15).



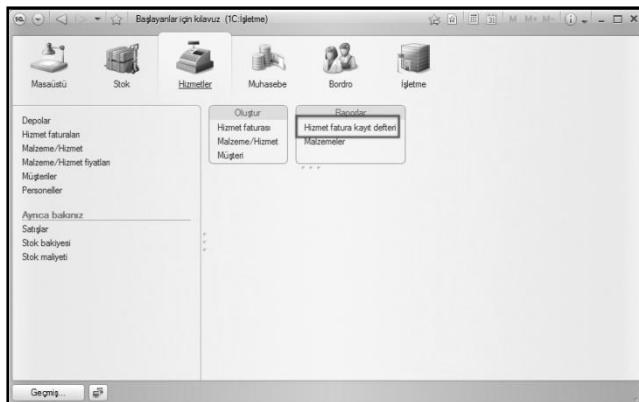
Resim 13.15. Raporun gösterileceği alt sistemini tanımlama

1C: İşletme ortamında

1C:İşletme sistemini hata ayıklama biçiminde çalıştıralım ve raporun nasıl çalışmasına bakalım.

Ekrana gelen 1C:İşletme penceresinin Hizmetler bölümünün eylem çubuğuunda *Hizmet fatura kayıt defteri* raporunu açmak için komutun var olduğunu göreceğiz.

Bu komutu uygulayalım (Resim 13.16).

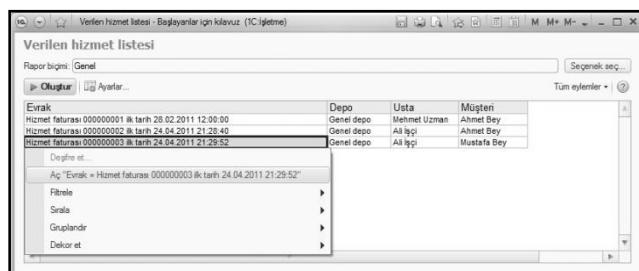


Resim 13.16. Rapor oluşturma komutu

Ekranımıza sistem tarafından otomatik olarak oluşturulan raporu formu gelir.

Formun başlığı raporun *Liste tanımı* özelliğinden alındığına dikkat edelim.

Oluştur butonuna tıklayalım (Resim 13.17).



Resim 13.17. Rapor sonucu

Gördüğümüz gibi raporu hizmet fatura listesini içerir.

Bu sırada Evrak alanında çift tıklayarak kaynak evrakını açabiliriz veya veri inşa edilme şemasının sunduğu diğer detaylardan faydalabiliriz.

Böylece, bu rapor örneğiyle, veri inşa edilme yapılandırıcısının nasıl kullanıldığını ve sorgu dilinin genel yapılarını görmüş olduk.

İki tablodan veri alma

“Verilen hizmet verimliliği” raporunda belli bir dönem içinde hangi verilen hizmetler “Aspirin usta” şirketine en fazla para getirdiğini gösterilecek (Resim 13.18).

Verilen Hizmet Verimliliği		
Veri parametreleri		
İlk tarih = 01.04.2011		
Son tarih = 30.04.2011		
Hizmet		Tutar
Yabancı marka televizyon tamiri		40,00
Elektrik bakımı		40,00
Yerini marka televizyon tamiri		30,00
Tepsiye		
Su başlama		
Toplam		110,00

Resim 13.18. Rapor sonucu

Verilen hizmet verimliliği raporu örneğiyle, belli bir dönemde verileri nasıl filtrelediğini, sorgu parametrelerini nasıl belirlendiğini, sorguda birden fazla tabloların verilerini nasıl kullanıldığını ve kaynaklardan bir tanesinin verilerini sorgu sonucuna nasıl dahil edildiğini gösterilecektir.

Ayrıca, veri inşa edilme sisteminin parametreleri ile nasıl çalışmak gerektiğini, standart tarihleri nasıl kullanmak gerektiğini öğreneceğiz ve raporun hızlı kullanıcı ayarları ile tanışacağız.

Bunun dışında raporda filtrelemeyi ve koşullu dekoru daha detaylı bir şekilde ayarlamaya öğreneceğiz.

Tasarımcı ortamında

Yeni *Rapor* konfigürasyon nesnesini ekleyelim.

Raporun adı *VerilenHizmetVerimliliği* olarak belirleyip veri inşa edilme şemasını çalıştırıralım.

Yeni *Veri seti – sorgu* ekleyelim ve sorgu yapılandırıcısını çağırıralım.

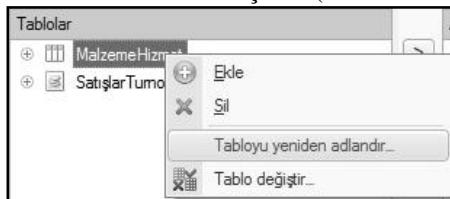
Veri seti için sorgu

İki tabloyu soldan birleştirme

Sorgu için veri kaynağı olarak nesnesel (referanslı) *MalzemeHizmet* tablosunu ve birikim kayıt tablosunun sanal tablosu olan *Satışlar.Turnovers* tablosunu seçelim.

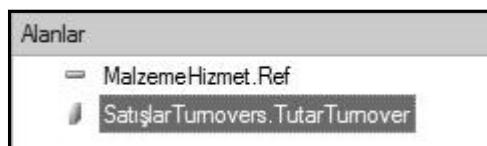
Sorguda isim çakışmasını önlemek için *MalzemeHizmet* tablosunu *krtMalzemHizmet* olarak yeniden adlandıralım.

Bunu yapmak için Tablolar listesinde tabloyu seçip sağ tuş menüsünden **Tabloyu yeniden adlandır** maddesini seçelim (Resim 13.19).



Resim 13.19. Sorguda tabloyu yeniden adlandırma

Alanlar listesine *MalzemeHizmet.Ref* ve *SatışlarTurnovers.TutarTurnover* alanlarını taşıyalım (Resim 13.20).



Resim 13.20. Seçilen alanlar

İlişkiler sekmesine geçelim.

Sorguda birden fazla tablo var olduğu için, tablolar arasında ilişkileri kurmamız gereklidir.

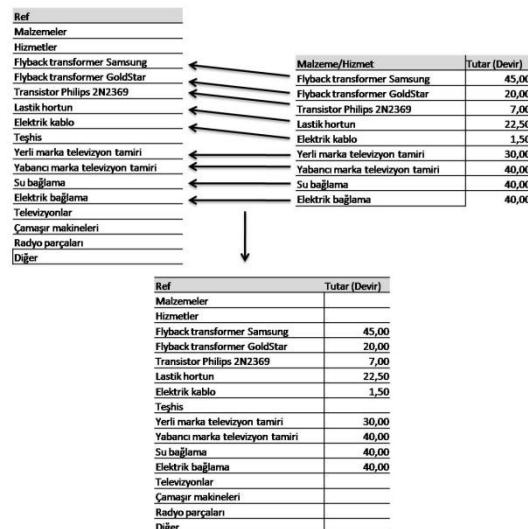
Varsayılan değer olarak platform tarafından *MalzemeHizmet* alanına göre ilişki kurulmuştur. Yani *Satışlar* birikim kayıt tablosunun *MalzemeHizmet* kaynağının değeri *MalzemeHizmet* kart listesinin öğe referansına eşit olmalıdır.

SatışlarTurnovers tablosunun **Tüm** işaretini kaldırırmak ve *krtMalzemeHizmet* tablosu için **Tüm** işaretini yerleştirmek gereklidir.

Böylece biz ilişkilendirme tipini *Soldan bağlantı* olarak işaretliyoruz. Yani sorgu sonucunda *MalzemeHizmet* kart listesinin tüm kayıtları ve bu kayıtlara *MalzemeHizmet* alanına göre uygun olan *Satışlar* kayıt tablosunun kayıtları yer alır.

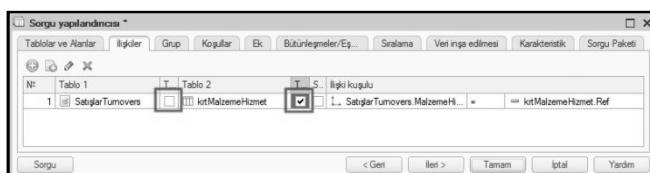
Böylece, sorgu sonucunda tüm hizmetler yer alır ve hizmetlerden bazıları için tutar değeri olacaktır. Belli dönem içinde müşteriye verilmeyen hizmetler için hiçbir değer belirlenmeyecektir.

Şema olarak tanımlanmış tablo ilişkisi aşağıdaki gibi gösterilebilir (Resim 13.21).



Resim 13.21. Sorguda tablo ilişkileri

Sonuç olarak İlişkiler sekmesi aşağıdaki gibi görünecektir (Resim 13.21).



Resim 13.22. Tablolar arasında ilişkiyi tanımlama

Kayıt filtreleme koşulu

Koşular sekmesine geçelim ve *MalzemeHizmet* kart liste klasörlerinin sorguda yer almamasını sağlayalım.

Bunu yapmak için *krtMalzemeHizmet* tablosunu açıp *IsFolder* alanını koşular listesine taşıyalım. *Serbest* alanında işaret yerleştirip *Koşul* alanında aşağıdaki metin yazalım (Liste 13.5).

Liste 13.5. Sorgu koşulu

```
krtMalzemeHizmet.IsFolder = FALSE
```

Böylece, veritabanından sadece klasör olmayan *MalzemeHizmet* kart listesinin öğelerini almak istediğimizi tanımladık.

Bu koşulun çalışmasını aşağıdaki resimle açıklayabiliriz. Sol tarafta – MalzemeHizmet kart listesinin kaynak tablosu, sağ tarafta – bu tablodan alınacak değer tablosu (Resim 13.23).

Ref	Tutar (Devir)
Hizmetler	
Flyback transformer Samsung	45,00
Flyback transformer GoldStar	20,00
Transistor Philips 2N2369	7,00
Lastik hortun	22,50
Elektrik kablo	1,50
Teghis	
Yerli marka televizyon tamiri	30,00
Yabancı marka televizyon tamiri	40,00
Su bağlama	40,00
Elektrik bağlama	40,00
Televizyonlar	
Çamaşır makineleri	
Radyo parçaları	
Diger	

Ref	Tutar (Devir)
Flyback transformer Samsung	45,00
Flyback transformer GoldStar	20,00
Transistor Philips 2N2369	7,00
Lastik hortun	22,50
Elektrik kablo	1,50
Teghis	
Yerli marka televizyon tamiri	30,00
Yabancı marka televizyon tamiri	40,00
Su bağlama	40,00
Elektrik bağlama	40,00

Resim 13.23. MalzemeHizmet kayıtlarını sorguda filtreleme

İkinci koşul şudur; seçilen ögenin hizmet olması gereklidir. Bu – *Basit koşul*. Onu oluşturmak için *MalzemeHizmetTürü* alanını koşul listesine taşıyalım.

Platform otomatik olarak koşulu oluşturur. Bu koşula göre malzeme/hizmet türü *MalzemeHizmetTürü* parametre değerine eşit olması gereklidir.

İllerde sorgu gerçekleştirmeden önce *MalzemeHizmetTürü* parametresine sayımlı değerini aktaracağı – **Hizmet**.

Bu koşulun çalışmasını da resim ile anlatalım. Solda birinci koşula göre filtrelenmiş *MalzemeHizmet* kart listesinin kayıtları. Sol tarafta sadece hizmet olan kayıtlar bulunur (Resim 13.24).

Ref	Tutar (Devir)
Hizmetler	
Flyback transformer Samsung	45,00
Flyback transformer GoldStar	20,00
Transistor Philips 2N2369	7,00
Lastik hortun	22,50
Elektrik kablo	1,50
Teghis	
Yerli marka televizyon tamiri	30,00
Yabancı marka televizyon tamiri	40,00
Su bağlama	40,00
Elektrik bağlama	40,00
Televizyonlar	
Çamaşır makineleri	
Radyo parçaları	
Diger	

Ref	Tutar (Devir)
Teghis	
Yerli marka televizyon tamiri	30,00
Yabancı marka televizyon tamiri	40,00
Su bağlama	40,00
Elektrik bağlama	40,00

Resim 13.24. Sorguda MalzemeHizmet kayıtlarını filtreleme

Sonuç olarak Koşullar sekmesi aşağıdaki gibi görünecek (Resim 13.25).



Resim 13.25. Sorgu koşullarını tanımlama

Tavsiye

Filtre sorguda tanımlandığı gibi rapor ayarlarında da tanımlanabilir. Aynı şey gruplandırma ve sıralama için geçerlidir. Sorgu koşuluna uygun olmayan kayıtlar raporda kesin kullanılmayacak ise, filtreleme ayarları sorguda yapmak daha iyidir. Sıralama ve gruplandırmayı rapor ayarlarında yapmak daha iyi, böylece rapor daha esnek olur.

Alan eşanımlıları

Bütünleşme/Eşanımlılar sekmesine geçip, kart liste ifadesi (*Ref* alanı) **Hizmet** eşanımlıya sahip olacağını ve kayıt tablo alanı **Tutar** eşanımlıya sahip olacağını belirleyelim (Resim 13.26).



Resim 13.26. Sorgu alanları için eşanımlıları tanımlama

Kayıt sıralaması

Sıralama sekmesine geçip Tutar alanına göre sıralanacağını belirleyelim, sıralama değeri Azalan olacaktır.



Resim 13.27. Sorgu kayıt sıralaması

Sorgu oluşturma işlemi tamamlanmıştır, *Tamam* butona tıklayalım. Veri inşa edilme şeması yapılandırıcısına geri dönelim.

Sorgu metni analizi

Platform tarafından oluşturulan sorgu metni aşağıdaki gibidir (Liste 13.6).

Liste 13.6. Sorgu koşulu

```

SELECT
    krtMalzemeHizmet.Ref AS Hizmet,
    SatışlarTurnovers.TutarTurnover AS Tutar
FROM
    Catalog.MalzemeHizmet AS krtMalzemeHizmet
        LEFT JOIN AccumulationRegister.Satışlar.Turnovers AS
SatışlarTurnovers
        ON SatışlarTurnovers.MalzemeHizmet = =
krtMalzemeHizmet.Ref
WHERE
    krtMalzemeHizmet.IsFolder = FALSE
    AND krtMalzemeHizmet.MalzemeHizmetTürü = =
&MalzemeHizmetTürü

ORDER BY
    Tutar DESC

```

İlk önce her zaman gibi sorgu tanımlama kısmı geliyor ve orada bizim için yeni kısımlar var.

Sorgu kaynağını tanımlandığında (FROM anahtar kelimesinden sonra) birden fazla kaynak kullanma olanağı kullanılmıştır (Liste 13.7).

Liste 13.7. Birden fazla sorgu kaynağını tanımlama

```

FROM
    Catalog.MalzemeHizmet AS krtMalzemeHizmet
        LEFT JOIN AccumulationRegister.Satışlar.Turnovers AS
SatışlarTurnovers
        ON SatışlarTurnovers.MalzemeHizmet = =
krtMalzemeHizmet.Ref

```

Örneğimizde kayıtlar iki kaynağından alınır; *krtMalzemeHizmet* ve *SatışlarTurnovers*. LEFT JOIN ... ON anahtar cümlesi ile iki kaynağın birleştirme yöntemi tanımlanmıştır.

LEFT JOIN şu anlamına gelir; sorgu sonucunda, ON koşuldan sonra tanımlanan koşula göre, iki tablonun kayıtlarını birleştirmek isteniyor. Bunun dışında sorgu sonucuna, ikinci kaynağında karşılığı bulunmayan birinci kaynağının (JOIN kelimesinden sol tarafında bulunan) kayıtlarını da eklemek gereklidir.

Sorgu metnine devam edelim. Sorgu tanımlama kısmında daha bir tane yeni yapı görüyoruz – kaynak tablosundan veri filtreleme koşulları (Liste 13.8).

Liste 13.8. Filtreleme koşulları

WHERE

```
krtMalzemeHizmet.IsFolder = FALSE
AND          krtMalzemeHizmet.MalzemeHizmetTürü      =
&MalzemeHizmetTürü
```

Filtreleme koşulları başında her zaman WHERE kelimesi durur. Bu kelimededen sonra koşul kendisi yer alır. Filtreleme uygulanan alanlar seçilen alan listesinde bulunmayabilir (örneğimizde olduğu gibi). Bunun dışında tanımladığımız koşulda sorgu parametresi kullanılmıştır – *MalzemeHizmetTürü* (parametre önce & simgesi yer alır).

Sorgu metni ile işimiz bitmiştir. Veri inşa edilme şemamızı oluşturmaya devam edebiliriz.

Kaynaklar

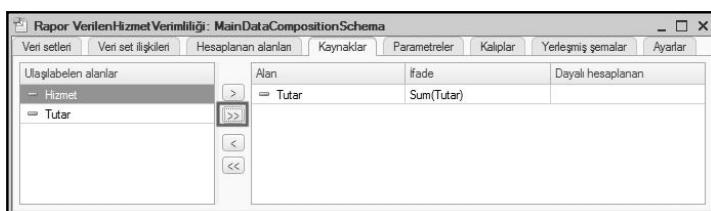
Raporumuzda her hizmet için toplam tutarını görmek istiyoruz. Bunun için raporun kaynak alanlarını tanımlamak gereklidir.

Veri inşa edilme sisteminde kaynaklar, gruplarda bulunan detaylı kayıtlar değerleri üzerinde hesaplanan alanlardır. Yani kaynaklar bu grupların ve total toplamlardır.

Toplam değerler *Kaynaklar* sekmesinde oluşturulur.

Kaynaklar sekmesine geçip butonuna tıklayalım. Veri inşa edilme şema yapılandırıcısı toplam hesaplanabilir tüm kaynakları seçer. Örneğimizde bu sadece *Tutar* kaynağı.

Platform otomatik olarak bu alanın değer toplamını hesaplanması ayarlar. Bu bizim için uygundur (Resim 13.28).



Resim 13.28. Veri inşa edilme şemasının kaynakları

Parametreler

Genelde kullanıcılar belli bir dönem için durumunu görmek isterler. Bu yüzden neredeyse her raporda raporlama döneminin başlangıcını ve sonu belirleyen parametreler kullanılır.

Raporun *parametreleri* rapordaki kayıtların filtreleme koşullarını tanımlar. Veri inşa edilme şemasında rapor parametreleri Parametreler sekmesinde tanımlanır (Resim 13.29).

Tanım	Bağık	Tip	Ulaşılabili...	U...	Değer	İfade	İşlevsel ...	U...	U...	Değiştir...
BeginOfPeriod		Begin of period	Date	<input type="checkbox"/>			<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
EndOfPeriod		End of period	Date	<input type="checkbox"/>			<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
MalzemeHizmet...	Malzeme hizmet t...	EnumRef.Malzeme...		<input type="checkbox"/>	Enum Malzeme...			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

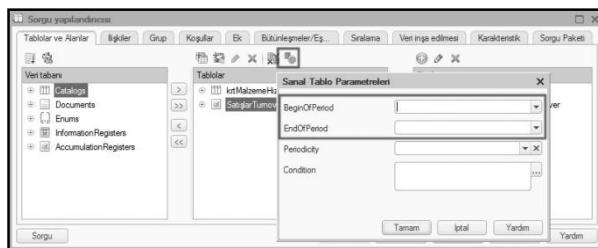
Resim 13.29. Veri inşa edilme parametreleri

Bu sekmede üç tane parametre görüyoruz; *BeginOfPeriod*, *EndOfPeriod* ve *MalzemeHizmetTürü*. Ortaya bir soru çıkıyor; sorguda bir tane parametre (*MalzemeHizmetTürü*) tanımladığımıza rağmen burada üç tane parametre yer alıyor, neden?

Veri inşa edilme şeması otomatik olarak sorgu metnini analiz eder ve geliştircisinin tanımladığı kesin parametreler dışında sorguda yer alan sanal tabloların parametrelerini de ayarlama imkanı veriyor.

BeginOfPeriod ve *EndOfPeriod* bu sorguda soldan birleşik (LEFT JOIN) olarak kullandığımız *AccumulationRegisters.Satışlar.Turnovers* sanal tablosunun ilk iki parametredir.

Sorgu yapılandırıcısında tablo listesinde bu tabloyu işaretledikten sonra Sanal tablo parametreleri butonuna basarsanız, *BeginOfPeriod* ve *EndOfPeriod* parametreleri görebileceğimiz pencere ekrana gelir (Resim 13.30).



Resim 13.30. Sanal tablo parametreleri

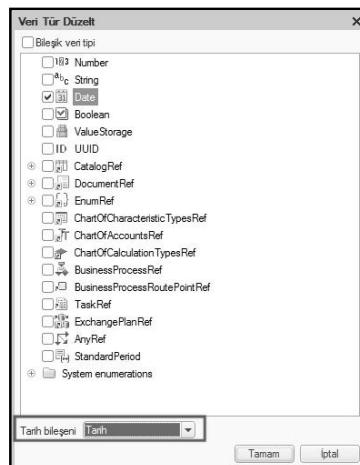
İlk parametre ile toplam hesaplamalarının ilk tarihi aktarılır, ikinci parametre ile ise son tarih. Sonuç olarak kaynak tablosu sadece belirlenmiş dönem için hesaplanmış toplamları içerir.

Burada hatırlamak gereken şey şudur; bu parametre olarak tarihi aktardığımızda (örneğimizde öyle olacak), tarih saniye ayrıntılı saatе de sahiptir.

Kullanıcı için saniye ayrıntıları ile raporu istemeyeceğini önceden bilmış olduğumuzu varsayıyalım. Bu durumda iki özelliği dikkate almak gereklidir.

Birincisi, kullanıcı tarih girdiği zaman saniye ayrıntılı saatе girmemelidir. Bunun için *BeginOfPeriod* parametresi için var olan tür tanımını değiştirelim.

Veri inşa edilme şemasının Parametreler sekmesine dönelim ve *BeginOfPeriod* parametresinin Tür alanındaki seçme butonuna tıklayalım. Ekrana gelene pencerenin alt bölümündeki *Tarih bileşeni* alanındaki değeri **Tarih** olarak değiştirelim. *Tamam* tıklayalım (Resim 13.31).



Resim 13.31. Tarih tipini değiştirme

İkinci özellik şudur; varsayılan değer olarak tarihteki saat 00:00:00 olarak belirlenir. Bu yüzden eğer kullanıcı 01.05.2011 – 14.05.2011 tarih aralığını belirlerse, kayıt tablo toplamları 01.05.2011 00:00:00 tarihinden 14.05.2011 00:00:00 tarihine kadar hesaplanacaktır. Böylece ayın 14’ünde ki veriler hesaplamada yer almaz, kullanıcıya bunu görünce biraz şaşırabilir.

Bu durumu ortadan kaldırmak için bir tane daha parametre ekleyelim. Kullanıcı bu parametre son tarihini girecek. *EndOfPeriod* parametre değerini değiştireceğiz. Parametre değeri olarak, kullanıcının girdiği tarihin gün sonu otomatik olarak tanımlanacaktır.

EndOfPeriod parametresi için *Ulaşılabilirlik sınırlaması* özelliğini işaretleyelim (Resim 13.32).

The screenshot shows a software interface for managing data composition schemas. The main title is 'Rapor Verilendirme Verimliliği - MainDataCompositionSchema'. Below it is a toolbar with icons for file operations like Open, Save, and Close. A tab bar includes 'Veri setleri', 'Veri set ilişkileri', 'Hesaplanan alanları', 'Kaynaklar', 'Parametreler', 'Kılçalar', 'Yerleşmiş şemalar', and 'Ayarlar'. The 'Parametreler' tab is selected. A table lists parameters with columns for Name, Bağlık (Type), Ulaşılabilirlik (Accessibility), Değer (Value), Ifade (Expression), and several checkboxes for 'Ulaşılabilirlik' (Accessibility) and 'Değer' (Value). The 'SonTarih' parameter is being added, with its 'Tip' (Type) set to Date and its 'Ulaşılabilirlik' (Accessibility) checkbox checked.

Rapor Verilendirme Verimliliği - MainDataCompositionSchema							
Veri setleri	Veri set ilişkileri	Hesaplanan alanları	Kaynaklar	Parametreler	Kılçalar	Yerleşmiş şemalar	Ayarlar
Tanım	Bağlık	Tip	Ulaşılabilirlik	Değer	Ifade	Ulaşılabilirlik	D...
BeginOfPeriod	İlk tarih	Date	<input type="checkbox"/>		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
EndOfPeriod	End of period	Date	<input type="checkbox"/>		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
MalzemeHizmetTuru	Malzeme hizmet türü	EnumRef MalzemeHizmet	<input type="checkbox"/>	Enum MalzemeHiz...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
SonTarih	Son tarih	Date	<input type="checkbox"/>		<input checked="" type="checkbox"/>	<input type="checkbox"/>	

Resim 13.32. "SonTarih" parametre ekleme

Bu işaret yerleştirilmezse, bu parametre kullanıcı tarafından değiştirilecektir. İşaret var olduğunda, kullanıcı bu parametreyi göremez.

Ondan sonra Ekle butonu kullanarak yeni parametre ekleyelim, adı *SonTarih* olsun (Resim 13.32). Bu parametre için platform otomatik olarak başlık oluşturacaktır – *Son tarih*.

Yeni oluşturululan parametre için tür tanımlayalım – *Date*. *BeginOfPeriod* parametresi için belirlediğimiz gibi, *SonTarih* parametresi için tarih bileşeni *Tarih* olarak tanımlayalım. *BeginOfPeriod* parametresinin Başlık özelliğini de değiştirelim – *İlk tarih* olsun.

Varsayılan değer olarak eklediğimiz parametre kullanıcı için ulaşılabilir durumdadır (Ulaşılabilirlik sınırlaması özelliği işaretlenmemiş durumda). Bizim için bu uygundur.

EndOfPeriod parametresine geçelim. Bu parametre için *Ulaşılabilirlik sınırlaması* işaretini yerleştirmıştık. Çünkü bu parametre değerini, *SonTarih* parametresine girilen değeri üzerinde hesaplanacaktır.

EndOfPeriod parametre değerini hesaplanacak formülü tanımlamak için veri inşa edilme şemasının ifade dilini kullanacağız.

Veri inşa edilme dilinde **EndOfPeriod()** fonksiyonu vardır. Bu fonksiyon ile herhangi bir dönemin son tarihi elde edinebilir, örneğin belirlenen günün son tarihi.

İfade hücresında *EndOfPeriod* parametresi için aşağıdaki ifade tanımlayalım (Liste 13.9).

Liste 13.9. "EndOfPeriod" parametresini hesapla ifadesi

```
EndOfPeriod(&SonTarih, "Day")
```

Not

Veri inşa edilme sistemini ifade dili ile ilgili detaylı açıklamalar 1C:İşletme sistemini yardım kısmında bulunabilir; *Yardım – Yardım içeriği – 1C:İşletme – Script - Common objects - Data Composition System - Data Composition System Expression Language.*

Yapmış olduğu eylemlerin sonucu olarak Parametreler sekmesi aşağıdaki gibi görünmelidir (Resim 13.33).

Tanım	Bağlık	Tip	Ula...	Ula...	Değer	İfade	İgloşel se...	U...	Ulaşabil...	D...
BeginOfPeriod	İlk tarih	Date	<input type="checkbox"/>					<input checked="" type="checkbox"/>	<input type="checkbox"/>	
EndOfPeriod	End of period	Date	<input type="checkbox"/>			EndOfPeriod(&SonTarih, "Day")		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
MalzemeHizmetTuru	Malzeme hizmet türü	Enum...	<input type="checkbox"/>			Enum.Malzeme...		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
SonTarih	Son tarih	Date	<input type="checkbox"/>					<input checked="" type="checkbox"/>	<input type="checkbox"/>	

Resim 13.33. Veri inşa edilme sisteminin parametreleri
Son olarak *MalzemeHizmetTürü* parametresini ayarlayalım.

Raporda sadece hizmetler ile ilgili tutarların bulunması gereklidir ve kullanıcı bu seçenekini değiştirmemeli, dolayısıyla *MalzemeHizmetTürü* parametrenin değerini veri inşa edilme şemasında tanımlamak gereklidir. *MalzemeHizmetTürü* parametresinin değeri *Enum.MalzemeHizmetTürleri.Hizmet* olarak belirleyelim.

MalzemeHizmetTürü parametresinin Ulaşabilirlik sınırlama özelliğini platform otomatik olarak tanımlamıştır. Oluduğu gibi bırakalım. Değer hücresinde seçim butonu kullanarak **Hizmet** değerini seçelim (Resim 13.34).

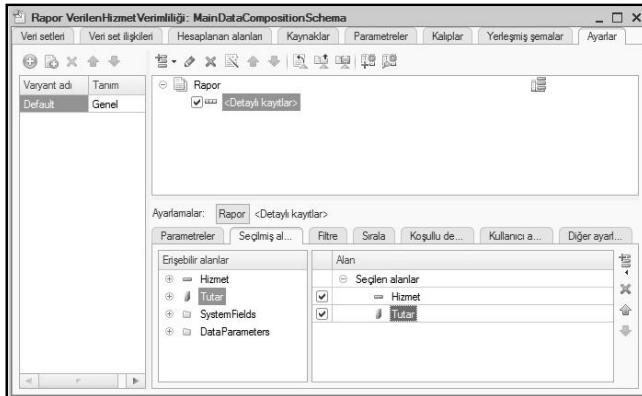
Tanım	Bağlık	Tip	Ula...	Ula...	Değer	İfade	İgloşel se...	U...	Ulaşabil...	D...
BeginOfPeriod	İlk tarih	Date	<input type="checkbox"/>					<input checked="" type="checkbox"/>	<input type="checkbox"/>	
EndOfPeriod	End of period	Date	<input type="checkbox"/>			EndOfPeriod(&SonTarih, "Day")		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
MalzemeHizmetTuru	Malzeme hizmet türü	Enum...	<input type="checkbox"/>			Enum.MalzemeHizmetTürleri.Hizmet	...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
SonTarih	Son tarih	Date	<input type="checkbox"/>					<input checked="" type="checkbox"/>	<input type="checkbox"/>	

Resim 13.34. *MalzemeHizmetTürü* parametre değerini tanımlama

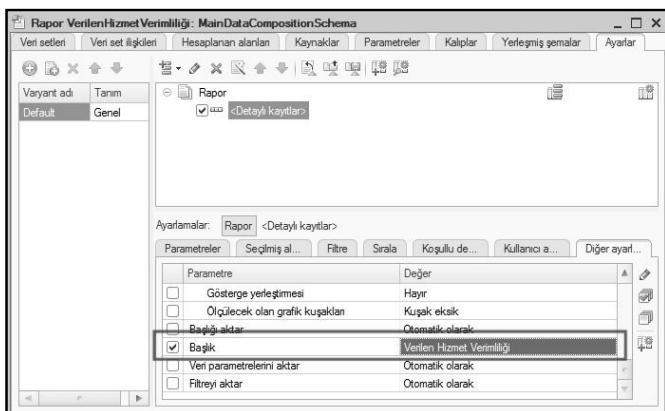
Ayarlar

Rapor yapısını oluşturmaya geçelim.

Ayarlar sekmesinde bir grup ekleyelim ve yine grup alanını belirlemeyelim. Seçilmiş alanlar sekmesinde Hizmet ve Tutar alanlarını belirleyelim (Resim 13.35).



Resim 13.35. “VerilenHizmetVerimliliği” raporunun yapısı
Ondan sonra Diğer ayarlar sekmesine geçip rapor başlığını belirleyelim – **Verilen Hizmet Verimliliği** (Resim 13.36).



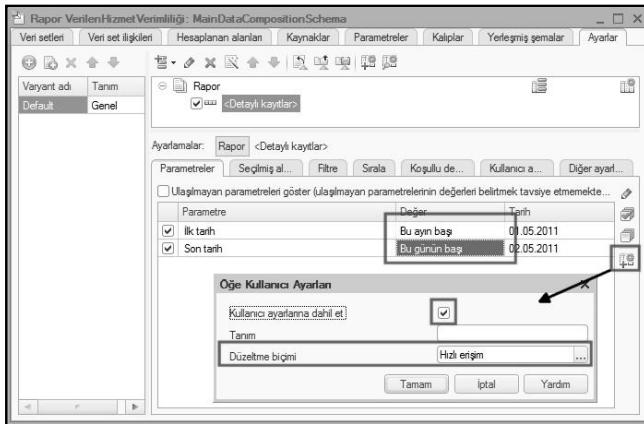
Resim 13.36. Rapor başlık tanımlama

Hızlı kullanıcı ayarları

Son olarak, kullanıcı için rapor dönemini belirleme olanağını tanımlamak gereklidir. Yani İlkTarih ve SonTarih parametreleri kullanıcı ayarlarına dahil etmek gereklidir.

Rapor dönemi her rapor oluşturulduğunda belirleneceği için, dönem belirleme alanları direkt rapor formunda bulunmalıdır.

Parametreler sekmesinde kullanıcı tarafından düzeltilebilen parametreleri görebiliriz (bu parametreler için *Ulaşılabilirlik sınırlaması* işaretli kaldırılmış durumdadır) (Resim 13.37).

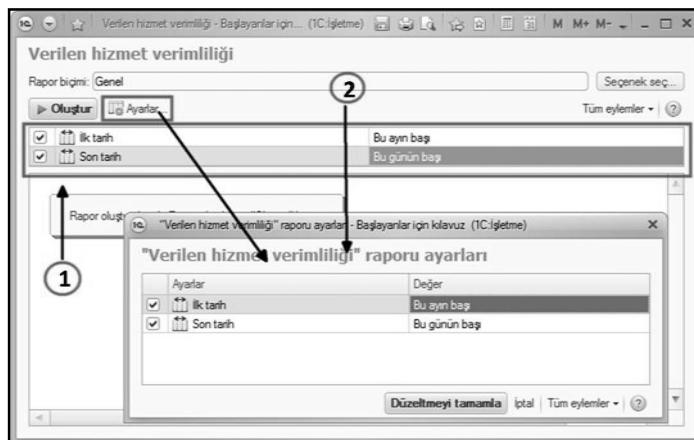


Resim 13.37. Kullanıcı ayarlarını tanımlama

Sırayla parametre seçelim ve sağ tarafından bulunan *Kullanıcı ayarları öğe özellikleri* butonuna tıklayalım.

Kullanıcı ayarlarına dahil et işaretini yerleştirelim ve varsayılan olarak gelen *Düzelteme biçimini – Hızlı erişim* olduğu gibi bırakalım.

Kullanıcı ayarlarına dahil et işaretini şu anlama gelir; rapor formundan Ayarlar butonu basıldığında bu parametre kullanıcı için ayrı bir pencerede (2) ulaşılabilir olur (sık kullanılmayan ayarlar için uygulanabilir, Resim 13.38).



Resim 13.38. Hızlı (1) ve normal (2) kullanıcı ayarları

Düzelteme biçimi – Hızlı erişim olarak belirlendiğinde, parametrenin direkt rapor formunda (1) bulunacağını belirler. Bu *hızlı kullanıcı ayarı* – her zaman kullanıcı için gerekli olan ayardır.

Kullanıcı arayüzüünü daha kullanılabılır duruma getirmek için *İlkTarih* ve *SonTarih* parametreler için değer olarak *Bu ayın başı* ve *Bu günün başı* belirleyelim (Resim 13.37).

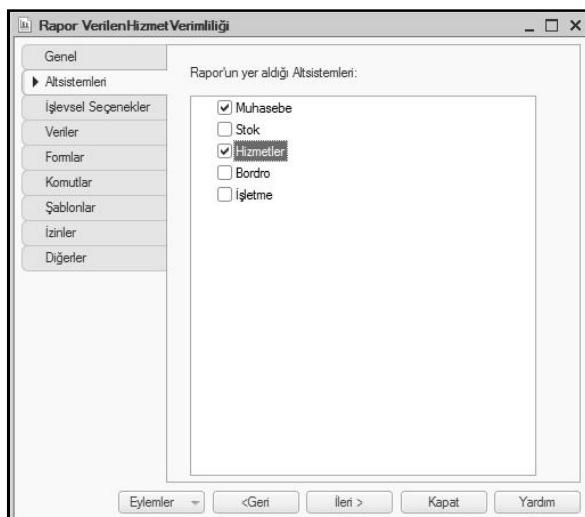
Böylece rapor oluşturulduğunda, raporun ilk tarihi ve son tarihi dinamik olarak değişir. Geçerli ayın başından bu güne kadar tarih aralığı otomatik olarak belirlenir.

Son olarak raporun hangi alt sistemlerinde bulunacağı belirleyelim.

Veri inşa edilme şemasının yapılandırıcı penceresini kapatıp *VerilenHizmetVerimliliği* konfigürasyon nesnesinin düzeltme penceresinin Alt sistemleri sekmesine geçip raporun ait olduğu alt sistemini tanımlayalım.

Listeden *Hizmetler* ve *Muhasebe* alt sistemini işaretleyelim.

Böylece raporu açma butonu *Hizmetler* ve *Muhasebe* bölümleri altında bulunacaktır (Resim 13.39).



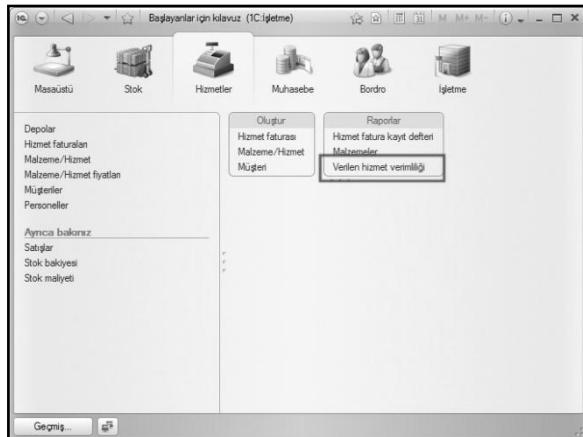
Resim 13.39. Raporun yer aldığı alt sistemler

1C: İşletme ortamında

1C:İşletme sistemini hata ayıklama biçiminde çalıştırıralım ve raporun nasıl çalışmasına bakalım.

Ekrana gelen 1C:İşletme penceresinin Hizmetler ve Muhasebe bölümlerinin eylem çubuklarında *Verilen hizmet verimliliği* raporunu açmak için komutun var olduğunu göreceğiz.

Bu komutu uygulayalım (Resim 13.40).



Resim 13.40. Rapor oluşturma komutu

Ekranımıza sistemin otomatik olarak oluşturulan rapor formu gelir.

Rapor penceresinde, raporlama dönemini belirleyen parametreleri görüyoruz. Varsayılan değer olarak raporlama dönemi bu ayın başından itibaren bugüne kadar tanımlanır. Dönemi değiştirmek istediginde ilgili alanda tarih değerini değiştirmek gerekir.

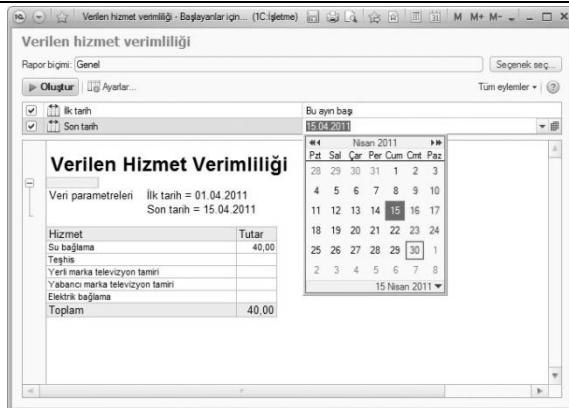
Oluştur butonuna tıklayalım. Sonuç aşağıdaki gibi görünür (Resim 13.41).

Verilen hizmet verimliliği	
Rapor tipi:	Genel
<input type="button" value="Olustur"/>	<input type="button" value="Ayarlar..."/>
<input checked="" type="checkbox"/> İlk tarih	Bu ayın baş
<input checked="" type="checkbox"/> Son tarih	Bu günün başı
Verilen Hizmet Verimliliği	
Veri parametreleri	İlk tarih = 01.04.2011 Son tarih = 30.04.2011
Hizmet	Tutar
Yabancı marka televizyon tamiri	40,00
Elektrik bağıma	40,00
Yerli marka televizyon tamiri	30,00
Tehhis	
Su bağıma	
Toplam	110,00

Resim 13.41. Rapor uygulama sonucu

Rapor sonucu ekranının başında tanımladığımız başlık ve raporlama dönemi yer alır.

Şimdi son tarihi 15.04.2011 olarak tanımlayalım. Sadece 1 nolu hizmet faturasının bilgileri raporda yer alır (Resim 13.42).



Resim 13.42. Rapor uygulama sonucu

Rapor sorgusunda malzeme/hizmet tablosu satışlar tablosu ile soldan birleştiğinden, satışı yapılmayan hizmetler raporda yine de yer alır.

Tasarımcı ve 1C:İşletme ortamındaki ayarlar.

Bu raporu örnek olarak raporun diğer ayarlarının oluşturmasını ve kullanılmasını göstereceğiz – *Koşullu dekor* ve *Filtre*.

Ayarları oluşturma sürecinde eylemleri tasarımcı ortamında oluşturup, ne elde ettiğimizi kontrol etmek için 1C:İşletme ortamında raporu çalıştıracağız.

Aslında, tasarımcı ortamında yapacağımız ayarlama işlemleri, kullanıcı ortamından da, **Tüm eylemler – Seçeneki değiştir** menüsünden ayarlanabilir. Fark şudur;

Tasarımcı ortamında oluşturacağımız ayarlara *standart ayarlar* denir ve bu ayarlar veri inşa edilme şemasında kaydedilir, yani bu ayarlar konfigürasyonu parçası olacaktır. Tüm konfigürasyon kullanıcıları, raporu tasarımcıdaki standart ayarlara göre ayarlanmış şeklinde görecekler.

Aynı şekilde rapor 1C:İşletme ortamından ayarlanabilir. Fakat bu ayarlar konfigürasyon parçası olmadan, sadece geçerli kullanıcı için uyarlanacaktır.

Not

Konfigürasyonda, kullanıcılar arası rapor ayarları aktarma mekanizması geliştirilebilir. Fakat bu kolay bir şey değil. Bu kitapta bu fonksiyon anlatılmayacaktır.

1C:İşletme ortamından rapor ayarlarını tasarlama, normal kullanıcı için değil, daha çok entegrasyonu uygulayan geliştirici veya sistem yöneticisi için düşünülmüştür.

1C:İşletme ortamındaki ayarlar rapor oluşturduğunda standart ayarlardan daha yüksek öncelik seviyesine sahiptir. Fakat kullanıcı standart ayarlara geri

dönmek istediğiinde **Tüm eylemler – Standart ayarları yükle** menüsü kullanılabilir.

Şu anda biz raporu tüm kullanıcılar için ayarlamak istiyoruz, dolayısıyla yapacağımızı değişikleri tasarım ortamında gerçekleştirelim.

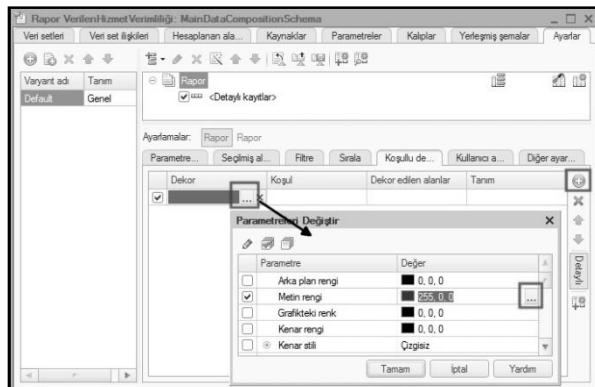
Koşullu dekor

Verilen hizmet verimliliği gibi raporlarda en fazla veya en az tutarı (veya başka bir kriterre göre) olan kayıtları farklı renk ile işaretlemek iyi olurdu.

Tasarımcı ortamında

Tasarımcı ortamında *Verilen hizmet verimliliği* raporunun veri inşa edilme şemasını açıp, Ayarlar sekmesine geçelim.

Pencereden alt kısmında **Koşullu dekor** sekmesine geçip, sağ üst köşede bulunan **Ekle** butonuna tıklayalım (Resim 13.43).



Resim 13.34. Koşullu dekor ayarları

İlk önce *Dekor* belirleyelim. Yani gerekli olan alanlar nasıl görüneceğini tanıyalayalım.

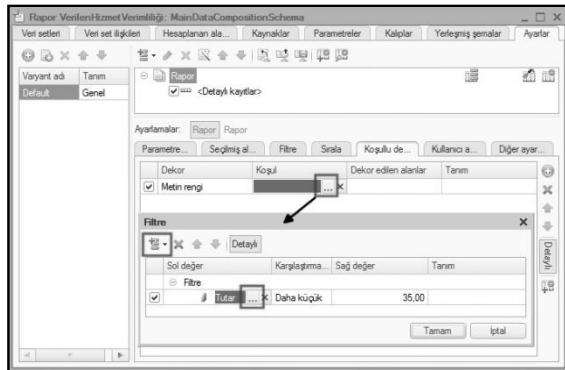
Dekor alanında seçim butonuna tıklayalım ve kırmızı metin rengini tanıyalayalım.

Tamam butonuna tıklayalım.

Ondan sonra *Koşul* alanında dekor uygulanacağı durumu tanımlamak gereklidir (örneğimizde alan kırmızı olacağı durumu belirtmek gereklidir).

Koşul alanında seçim butona tıklayalım ve ekrana gelen pencerede *Yeni filtreleme öğesini* oluşturmak gereklidir.

Her öğe bir tane koşul tanımlayacak. Birden fazla koşul olabilir (Resim 13.44).



Resim 13.44. Koşullu dekor ayarları

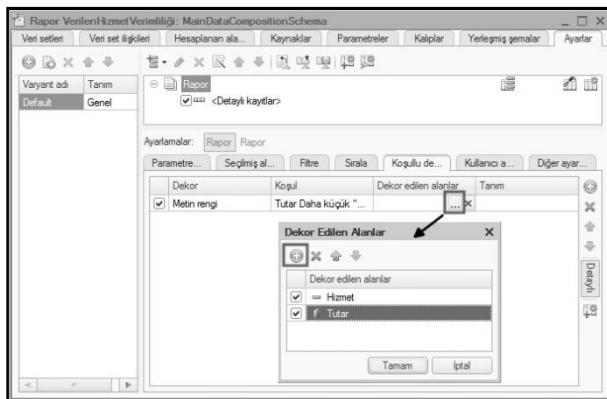
Bunu yapmak için Ekle butonuna tıklayalım ve *Sol değer* alanında – **Tutar** alanı, *Karşılaştırma değer* alanında – **Daha küçük** ve *Sağ değer* alanında – **700** tanımlayalım.

Tamam butona tıklayalım.

Tutar alanında 700'den daha küçük değer olduğunda “bir şey” kırmızı renk ile işaretlenecektir.

Şimdi bu “bir şeyi” tanımlayalım. Dekor edilen alan listesini belirtelim.

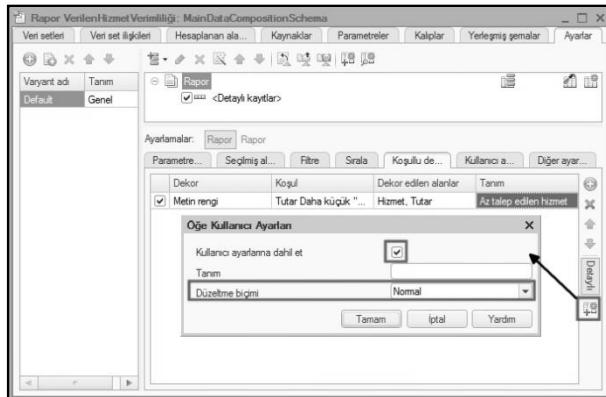
Raporun tam satırını işaretlemek istediğimizde bu liste boş bırakılabilir. Veya seçim butonunu tıkladıktan sonra, *Dekor edilen alanlar* alanında *Hizmet* ve *Tutar* alanları seçilebilir (Resim 13.45).



Resim 13.45. Koşullu dekor ayarları

Tamam butona tıklayalım.

Son olarak koşullu dekorunun tanımı *Az talep edilen hizmet* olarak belirleyelim (Resim 13.46).



Resim 13.46. Koşullu dekor ayarları

Az talep dilen hizmet – kullanıcı ayarlarında gösterileceği tanımdır. Yani kullanıcı “Tutar Daha küçük 700” yerinde daha anlamlı bir yazı görecektir. Şimdi bu koşulu kullanıcı ayarlarına ekleyelim.

Ayarlama lisesinin sağ alt köşede bulunan *Kullanıcı ayarları öge özelliklerini* butonuna tıklayalım (Resim 13.46). *Kullanıcı ayarların dahi et* işaretini yerleştirelim ve düzeltme biçimini *Normal* olarak seçelim.

Böylece oluşturduğumuz koşullu dekor ayarı, kullanıcı ortamından normal ayar olarak ulaşılabilir olmuştur. Hızlı ayarlardan farklı olarak normal ayarı direkt rapor formunda değil, *Ayarlar* butonu ile çağırılan özel pencerede bulunur.

1C: İşletme ortamında

1C: İşletme ortamında geçelim. Rapor açalım.

Raporlama dönemini son tarihini *Bugünün başı* olarak belirtip *Oluştur* butona tıklayalım (Resim 13.47).

Verilen hizmet verimliliği	
Rapor başlığı: Genel	<input type="button" value="Seçenek seç..."/>
<input type="button" value="Oluştur"/>	<input type="button" value="Ayarlar..."/>
<input type="checkbox"/> İlk tarih	Bu ayın başı
<input checked="" type="checkbox"/> Son tarih	Bu günün başı
Verilen Hizmet Verimliliği	
Veri parametreleri:	Son tarih: 08.05.2011
Hizmet	Tutar
Yabancı marka televizyon tamiri	40,00
Su bağlama	40,00
Elektrik怕alaması	40,00
Yerli marka televizyon tamiri	30,00
İşlemler	
Toplam	150,00

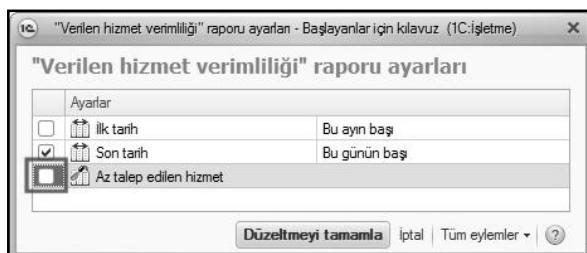
Resim 13.47. Rapor oluşturma sonucu

35 TL'den daha düşük hizmet tutarı kırmızı renk ile işaretlenmiş durumdadır.

Ayarlar butonuna tıklayalım.

Ekrana, rapor dönemleri ve koşullu dekor ayarı içeren kullanıcı ayarlar lisesi gelir.

Koşullu dekor ayarının işaretini kaldırıp **Düzelteymi tamamla** butona bastıktan sonra (Resim 13.48) raporu yeniden oluşturalım.



Resim 13.48. Kullanıcı ayarlar penceresi

Renk ile işaretleme uygulanmayacaktır.

Az talep edilen hizmet ayarı rapor formunda görünmüyör, çünkü bu ayar için düzeltme biçimini olarak *Hızlı* değil *Normal* erişimi tanımlamıştık.

Bu ayar kesin olarak belirtildi, yani kullanıcı sadece kullanıp kullanılmadığını seçebilir.

Daha deneyimli kullanıcılar için daha geniş ayarlama seçenekleri tanımlayabiliriz. Örneğin, onlara özel filtreleme, sıralama, koşullu dekor ayarları tanımlama olanağı sağlayabiliriz.

Bir sonraki örnekle bu özelliğe inceleyelim.

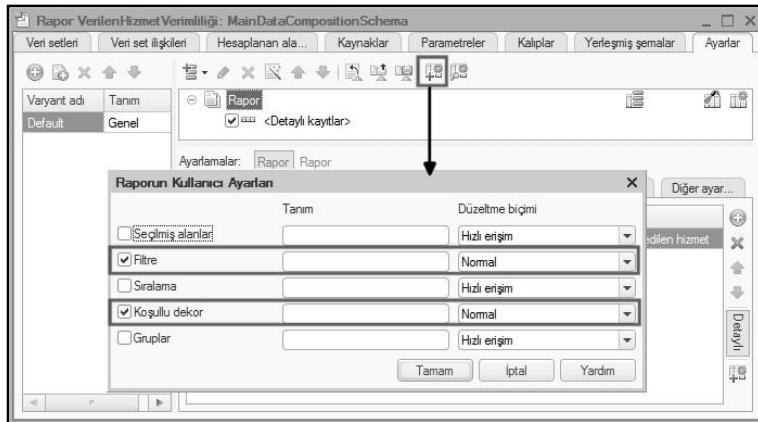
Kullanıcı ayarları

Tasarımcı ortamında

Tasarımcı ortamında geri dönelim.

Veri inşa edilme şemasının Ayarlar sekmesinde geliştirici tarafından tanımlandığı tüm ayarlar bulunur. Ayarlardan bazıları kullanıcı ortamında, serbest filtre ve koşullu dekor tanımlamak için, yansıtılmış olabilir.

Bunu yapmak için ayarlar penceresinin komut panelinde bulunan *Kullanıcı ayarlar öğe özellikleri* butonuna tıklayalım (Resim 13.49)



Resim 13.49. Kullanıcı ayarlar bileşenleri

Ekrana gelen pencerede raporun kullanıcı ayarları düzeltme işlemleri yapabiliriz.

Filtre ve *Koşullu dekor* alanları için işaret yerleştirip *Düzelteme biçimi* özelliklerini *Normal* olarak tanımlayalım.

Böylece, filtre ve koşullu dekor ayarları *Ayarlar* butonu ile çağrılan kullanıcı ayarlar listesine dahil ettik.

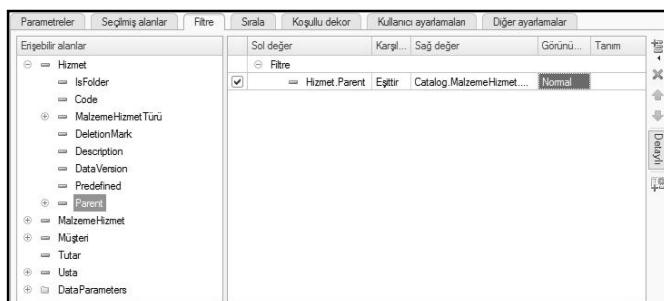
Filtre

Tasarımcı ortamında

Şimdi raporun filtre ayarı oluşturalım. Bunun için ayarlar penceresinin alt kısmında *Filtre* sekmesine geçelim.

Sol tarafta raporun ulaşılabilir alanları görüyoruz.

Hizmet alanı açıp **Parent** alanı üzerinde çift tıklayarak onu sağ tarafta bulunan filtreleme listesine ekleyelim (Resim 13.50).



Resim 13.50. Filtre ayarlama

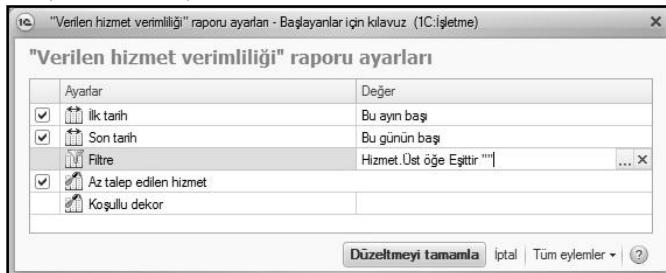
Görümüm biçimini Normal olarak işaretleyelim.

Böylece 1C:İşletme ortamından hizmetleri ait olduğu gruplara göre filtreleme tanımlama olanağını ayarladık.

1C: İşletme ortamında

1C:İşletme ortamında rapor açıp Ayarlar butonuna tıklayalım.

Raporun kullanıcı ayarları penceresinde *Filtre* ve *Koşullu dekor* ayarları ortaya çıktı (Resim 13.51).



Resim 13.51. Kullanıcı ayarları penceresi

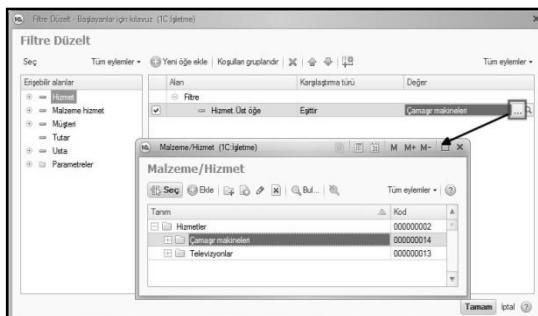
Aslında bura iki tane koşullu dekor ayarı bulunur.

Az talep edilen hizmet ayarı önceden tasarımcı ortamında tanımlamıştık. Bunun yanı sıra, son kullanıcıya koşullu dekor ayarları tam müdahale etme olanağını tanımlamıştık. Kullanıcı artık isteklerine göre kendine özel koşullu dekor ayarları tanımlayabilir.

Şimdi, raporumuzda sadece çamaşır makineler ile ilgili hizmetler yer alacak şeklinde, filtre ayarları tanımlayalım. Bunu yapmak için kullanıcı ayarlar listesinde *Filtre* satırının değer hücresinde seçme butonuna tıklayalım (Resim 13.51).

Ekrana gelen filtre düzeltme penceresinde tasarımcıda önceden oluşturduğumuz filtre koşulu görüyoruz.

Değer alanında seçim butonuna basıp, hangi değere göre filtreleme yapılacağını belirtmek gereklidir. Malzeme/Hizmet kart listesinden Hizmetler grubu altında *Çamaşır makineleri* grubu seçmek gereklidir (Resim 13.52).

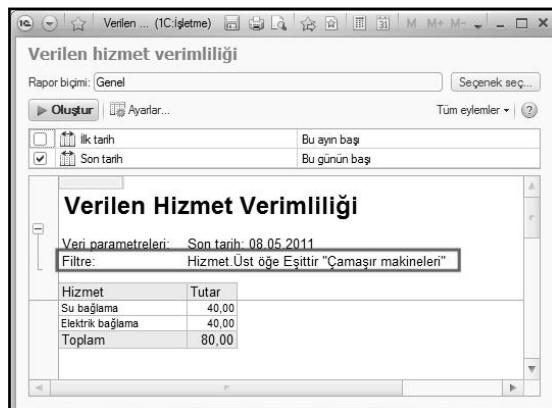


Resim 13.52. Filtre ayarları

Tamam butona tıklayalım.

Tanımladığımız filtreye göre raporda sadece Çamaşır makineleri üst öge olan Malzeme/Hizmet kartları ile ilgili satış bilgileri yer alır.

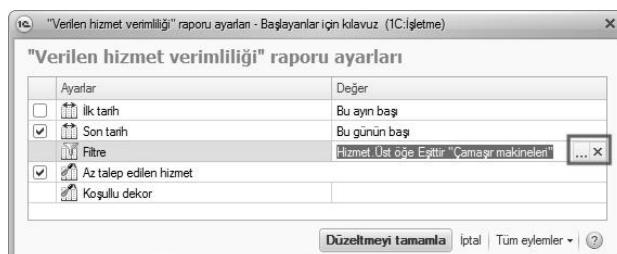
Raporun kullanıcı ayarlar penceresinde *Düzelteyi tamamla* butona basıp *Oluştur* butona tıklayarak raporu oluşturalım (Resim 13.53).



Resim 13.53. Rapor oluşturma sonucu

Gördüğümüz gibi raporda sadece çamaşır makineleri ile ilgili hizmetler yer aldı ve raporun başlığında filtre ile ilgili bilgi bulunur.

Ayarlar penceresini çağırıp, *Filtre* satırında temizleme butonu basarakfiltreleme ayarları temizlenebilir veya butonuna basarak başka bir kriter'e göre ayarlanabilir (Resim 13.54).



Resim 13.54. Kullanıcı ayarlar penceresi

Böylece, kullanıcı belli bir bilgiye sahip olmadan sonra farklı ayarları kendine özel yapabılır.

Şimdi tasarımcıya geri dönüp filtre ayarının kullanma işaretini kaldırıralım. Sonraki örneklerde bu bize lazım olacaktır.

Seçilen dönemde gün detayı ile veri dökümü

Eklenecek olan bir sonraki rapor – *Ustalar satış raporu*.

Rapor, her ustanın belli bir dönemde kazanmış olduğu tutar gösterilecek. Usta satışları gün bazlı verilecek ve gün içerisinde hangi müşteriye hizmet verildiğini de listelenecektir (Resim 13.55).

Ustalar Satış Raporu			
Usta	Dönem	Müşteri	Tutar
Toplam			239,70
Ahmet Yılmaz			85,00
	19.04.2011 00:00:00		
	20.04.2011 00:00:00		
	21.04.2011 00:00:00		
	22.04.2011 00:00:00		
	23.04.2011 00:00:00		
	24.04.2011 00:00:00		85,00
	25.04.2011 00:00:00	Ayşe Hanım	85,00
	26.04.2011 00:00:00		
Ali İşçi			107,20
	19.04.2011 00:00:00		
	20.04.2011 00:00:00		
	21.04.2011 00:00:00		
	22.04.2011 00:00:00		
	23.04.2011 00:00:00		
	24.04.2011 00:00:00		107,20
	25.04.2011 00:00:00	Mustafa Bey	107,20
	26.04.2011 00:00:00		
Mehmet Uzman			47,50
	19.04.2011 00:00:00		
	20.04.2011 00:00:00		
	21.04.2011 00:00:00		47,50

Resim 13.55. Rapor sonucu

Bu rapor örneği ile birden fazla grup olan raporlar nasıl tasarlanabildiğini ve dönemde tüm tarihleri nasıl gösterildiğini açıklanacaktır.

Bunun dışında, rapor yapısının belli öğelerin ayarlanması, verileri grafik olarak gösterilmesi ve birden fazla rapor seçeneklerini oluşturması ile ilgili bilgiye sahip olacağız.

Tasarımcı ortamında

Yeni rapor konfigürasyon nesnesini ekleyelim.

Raporun adı olarak **UstalarSatışRaporu** tanımlayalım ve veri inşa edilme şemasını açalım.

Yeni *Veri seti – sorgu* ekleyip sorgu yapılandırıcısını açalım.

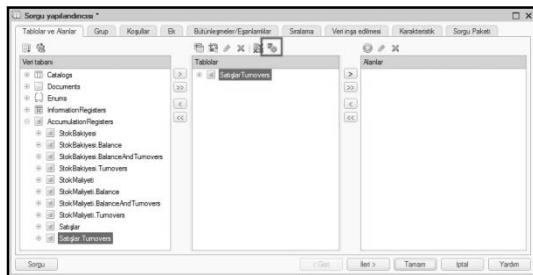
Sorgu için kaynak tablo olarak birikin kayıt tablosunun *Satışlar.Turnovers* sanal tablosunu seçelim.

Veri seti için sorgu

Sanal tablo parametreleri

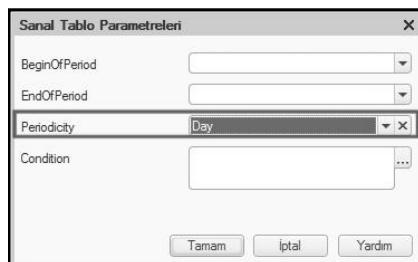
Sanal tablo parametrelerinden birini tanımlayalım – **Periodicity**.

Bunu yapmak için *Tablolar* alanında tabloyu seçip *Sanal tablo parametreleri* butonuna tıklayalım (Resim 13.56).



Resim 13.56. Sanal tablo parametrelerini değiştirme

Ekrana gelen parametre ekranında *Periodicity* belirleyelim – **Day** (Resim 13.57).



Resim 13.57. Sanal tablo parametreleri

Tamam butonuna tıklayalım.

Ondan sonra tablodan aşağıdaki alanları seçelim (Resim 13.58);

- SatışlarTurnovers.Usta
- SatışlarTurnovers.Period
- SatışlarTurnovers.Müşteri
- SatışlarTurnovers.TutarTurnover



Resim 13.58. Seçilen alanlar

Bütünleşmeler/Eşanamlıları sekmesine geçip *SatışlarTurnovers.TutarTurnover* alanı için **Tutar** eşanamlılığını tanımlayalım (Resim 13.59).

Alan adı	Sorgu 1
↳ Usta	↳ SatışlarTurnovers.Usta
⇒ Period	⇒ SatışlarTurnovers.Period
↳ Müşteri	↳ SatışlarTurnovers.Müşteri
☒ Tutar	☒ SatışlarTurnovers.TutarTurnover

Resim 13.59. Bütünleşmeler/Eşanamlılar

Sorgu metin analizi

Tamam butonuna basıp sorgu yapılandırıcısı tarafından oluşturulan sorgu metnini inceleyelim (Liste 13.10).

Liste 13.10. Sorgu metni

```

SELECT
    SatışlarTurnovers.Usta,
    SatışlarTurnovers.Period,
    SatışlarTurnovers.Müşteri,
    SatışlarTurnovers.TutarTurnover AS Tutar
FROM
    AccumulationRegister.Satışlar.Turnovers(, , Day, ) AS SatışlarTurnovers

```

Sorgu açıklama kısmında veri kaynağının periyodikliği *Gün (Day)* olarak belirlenmiştir (Liste 13.11).

Liste 13.11. Sanal tablo periyodikliği tanımlama

```
AccumulationRegister.Satışlar.Turnovers(, , Day, ) AS SatışlarTurnovers
```

Bunun sayesinde seçilmiş alan olarak Period alanını seçme olanağımız oldu.

Kaynaklar

Veri inşa edilme şemasını düzeltmeye devam edelim.

Kaynaklar sekmesinde butona basarak, veri inşa edilme yapılandırıcısı **Tutar** alanını seçtiğinden emin olalım.

Parametreler

Parametreler sekmesine bir önceki rapor için uyguladığımız işlemleri yapalım.

BeginOfPeriod parametresi için başlık İlk tarih olarak belirleyelim. *Tip* alanında tarih bileşenini **Tarih** olarak değiştirelim.

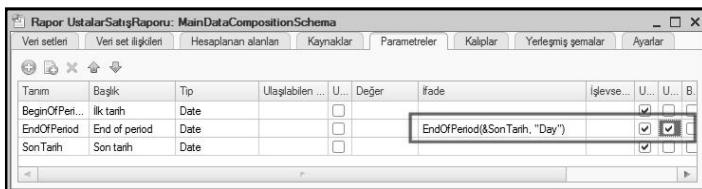
Ondan sonra bir tane daha parametre ekleyelim – **SonTarih**. Tipi *Date* olarak belirleyerek tarih bileşenini *Tarih* olarak seçelim.

EndOfPeriod parametresi için ifade tanımlayalım (Liste 13.12) ve *Ulaşılabilitlik sınırlaması* işaretini yerleştirelim.

Liste 13.12. EndOfPeriod parametre ifadesi

EndOfPeriod(&SonTarih, "Day")

Sonuç olarak parametreler sekmesi aşağıdaki gibi görünmelidir (Resim 13.60).



Resim 13.60 Veri inşa edilme parametreleri

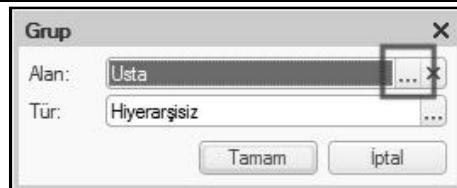
Ayarlar

Şimdi rapor yapısını oluşturalım.

Ayarlar sekmesinde iç içe iki grubu oluşturalım;

- Üst seviye – **Usta** alanına göre
- Alt seviye – **Period** alanına göre.

Bunu yapmak için rapor yapısında *Rapor* kök öğesini işaretleyip, ayarlar penceresinin eylem çubuğundaki **Ekle** butonuna tıklayarak yeni grup oluşturalım. Grup alanı olarak **Usta** alanını seçelim (Resim 13.61).



Resim 13.61. Grup alanları

Ondan sonra *Usta* grubu altında bir grup daha ekleyelim – *Period*.

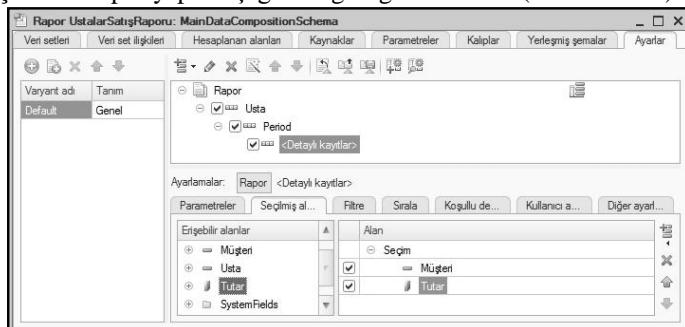
Bunu yapmak için *Usta* grubu işaretleyip **Ekle** butonuna basarak yeni grup ekleyelim ve grup alanı olarak *Period* alanını seçelim.

Bir tane daha grup ekleyelim. *Period* alanına göre grubu seçerek altında *Detaylı kayıtlar* grubu (herhangi bir grup alanı seçmeden) tanımlayalım.

Bundan sonra *Seçilmiş alanlar* sekmesine geçip seçilmiş alanlar listesine *Müşteri* ve *Tutar* alanları taşıyalım.

Usta ve *Dönem* alanları seçilmiş alanlar listesine eklemiyoruz, çünkü biz onları zaten grup olarak tanımlamıştık ve onların değerleri otomatik olarak gösterilecektir.

Sonuç olarak rapor yapısı aşağıdaki gibi görünecektir (Resim 13.62).



Resim 13.62. Rapor yapı ve alanları

Son olarak *Diğer ayarlamalar* sekmesine geçip aşağıdaki parametreleri değiştirelim.

Grup alanlarının yerleştirimi parametresi için **Ayrı ve sadece toplamlarda** değeri tanımlayalım.

Varsayılan değer olarak grup alanları raporda dikey olarak yerleştirilir (Resim 13.64).

Usta	Tutar
Dönem	
Müşteri	
Ali İşçi	192,20
24.04.2011 00:00:00	192,20
Ahmet Bey	85,00
Mustafa Bey	107,20
Mehmet Uzman	47,50
05.04.2011 00:00:00	47,50
Ahmet Bey	47,50
Toplam	239,70

Resim 13.63. Varsayılan değer olarak grup alanlarının ve toplamlarının yerleştirilmesi

Bu özelliğini Ayrı ve sadece toplamlarda olarak belirlenmesi, her grup ayrı bir sütunda yerleştirileceğini ve onun tanımı sadece bu grup içinde gösterileceğini ifade eder (Resim 13.64).

Usta	Dönem	Müşteri	Tutar
Ahmet Yılmaz			85,00
	24.04.2011 00:00:00		85,00
		Ayşe Hanım	85,00
Ali İşçi			107,20
	24.04.2011 00:00:00		107,20
		Mustafa Bey	107,20
Mehmet Uzman			47,50
	05.04.2011 00:00:00		47,50
		Ahmet Bey	47,50
Toplam			239,70

Resim 13.64. Grup alanlarının “Ayrı ve sadece toplamlarda” olarak yerleştirilmesi

Genel toplamların dikeye göre yerleştirilmesi parametresi için **Baş** değeri belirleyelim.

Varsayılan değer olarak dikey olarak toplamlar raporun sonunda yer alır (Resim 13.63). Bu özellik Baş olarak tanımlandığında rapor toplamları raporun başında yer alır (Resim 13.65).

Usta	Dönem	Müşteri	Tutar
Toplam			239,70
Ahmet Yılmaz			85,00
	24.04.2011 00:00:00		85,00
		Ayşe Hanım	85,00
Ali İşçi			107,20
	24.04.2011 00:00:00		107,20
		Mustafa Bey	107,20
Mehmet Uzman			47,50
	05.04.2011 00:00:00		47,50
		Ahmet Bey	47,50

Resim 13.65. Dikey toplamların başta yerleştirme

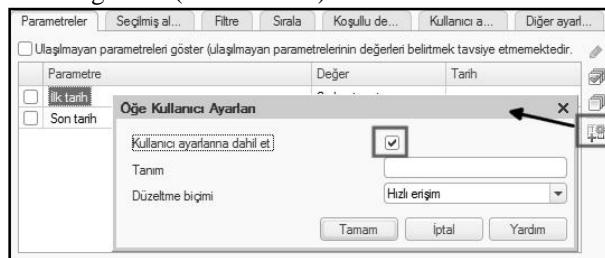
Sonuç olarak raporun diğer ayarlamalar aşağıdaki gibi görünecekler (Resim 13.66).

Parametreler	Seçilmiş al...	Filtre	Sırala	Koşullu de...	Kullanıcı a...	Diğer ayar...
Parametre	Değer					
<input type="checkbox"/> Dekor şablonu	Genel					
<input type="checkbox"/> Toplam yerlestirmesi	Otomatik olarak					
<input checked="" type="checkbox"/> Grup alanlarının yerlestirmesi	Aynı ve sadece toplamlarda					
<input type="checkbox"/> Grup yerlestirmesi	Baş					
<input type="checkbox"/> öznitelik yerlestirmesi	Sahiple birlikte					
<input type="checkbox"/> Kaynaklar yerlestirmesi	Yatay					
<input type="checkbox"/> Genel toplamlann yataya göre yerlestirmesi	Otomatik olarak					
<input checked="" type="checkbox"/> Genel toplamlann dikkate göre yerlestirmesi	Baş					
<input type="checkbox"/> Alan başlığı türü	Otomatik olarak					

Resim 13.66. Rapor gösterme parametreleri

Son olarak ayarlamalar sekmesinde *Başlık* parametresini **Ustalar Satış Raporu** olarak tanımlayalım.

İlk tarih ve Son tarih parametrelerin kullanıcı ayarlarında direkt rapor formunda yer alacağını tanımlayalım. Yani parametreler için Kullanıcı ayarlarına dahil et işaretini yerleştirmek ve Düzeltme biçimini özelliğini Hızlı olarak tanımlamak gereklidir (Resim 13.67).



Resim 13.67. Rapor dönemin hızlı ayarlarını oluşturma

En sonunda raporun hangi alt sistem altında gösterileceğini belirtmek gereklidir.

Veri inşa edilme şema yapılandırıcısını kapatalım ve *UstalarSatışRaporu* konfigürasyon nesnesinin düzeltme penceresinin *Alt sistemler* sekmesine geçelim.

Alt sistem listesinden *Hizmetler* ve *Bordro* alt sistemlerini işaretleyelim.

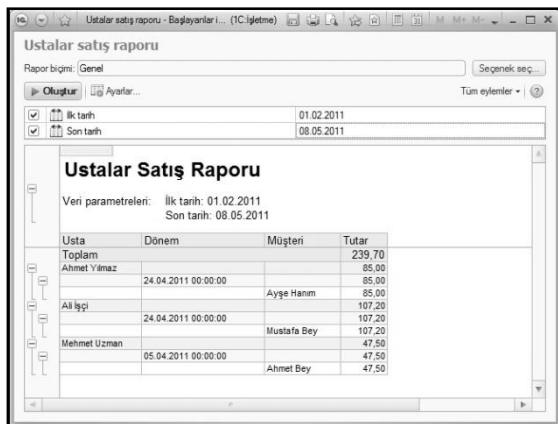
1C: İşletme ortamında

1C: İşletme sistemini hata ayıklama biçiminde çalıştırılarım ve raporun nasıl çalıştuğunu bakalım.

Ekrana gelen 1C: İşletme penceresinde *Hizmetler* ve *Bordro* bölümlerinin eylem çubuklarında *Ustalar satış raporu* oluşturma komutu yer aldığı görüyorum.

Bu komutu uygulayalım.

Raporlama dönemini 01.02.2011 – 08.05.2011 olarak belirleyelim ve raporu oluşturalım (Resim 13.68).



Resim 13.68 Rapor oluşturma sonucu

Seçilen dönemde tüm tarihlerin yansıtılması

Hatırlıyorsanız, bölümün başında bu raporda veriler, seçilen dönem içinde her gün için gösterileceğini söylemiştık.

Şu anda raporumuzda sadece *Satışlar* birikim kayıt tablosunda kaydı olan günler yansıtılmaktadır.

Verileri daha detaylı bir şekilde sunmak için veri inşa edilme sistemi gruplar için dönem tamamlama imkanı sunmaktadır.

Bu yüzden şu anda raporumuzu biraz değiştirelim. Belirlenen dönem içinde tüm tarihlerin (günler) raporumuza aktarılmasını sağlayalım.

Tasarımcı ortamında

Tasarımcı ortamına geri dönelim ve daha hassas rapor ayarını yapalım.

Veri inşa edilme şemasını *Ayarlar* sekmesinde açalım.

Şu ana kadar yaptığımız ayarlar raporun tümüne uygulanmıştır. Fakat veri inşa edilme sisteminde raporun tamamı için ayarlar belirlenebildiği gibi belli bir Höhe için de belirlenebilmektedir.

DİKKAT!

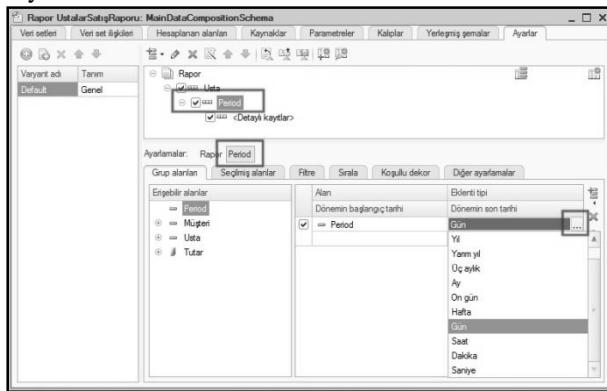
Rapor ayarları belirlendiğinde, pencerenin orta kısmında, rapor yapı aғacı altında, ayarlama biçimine uygun olan butonun aktif olması gereklidir. Raporun tamamı için ayarlar belirlendiğinde - **Rapor** butonu veya örneğin, sadece *Detaylı kayıtlar* grubu öğesi için ayarlar belirlendiğinde - **Detaylı kayıtlar** butonu aktifleştirmek gereklidir.

Bizim örneğimizde *Period* grubunun ayarları değiştirmek gereklidir.

Bu grubun ayarlarına geçmek için rapor yapı ağacında bu grubu işaretlemek gerekir ve pencerenin orta kısmında **Period** butona tıklamak gereklidir.

Pencerenin alt kısmında sadece bu gruba ait ayarlar görünür.

Grup alanları sekmesine geçelim. *Period* alanı için *Eklenti tipi – Gün* olarak belirleyelim.



Resim 13.69. Dönemin tamamlama tipini belirleme

Bunu yaptıktan sonra, raporda sıfır kayıt olan tarihler de gösterilecektir.

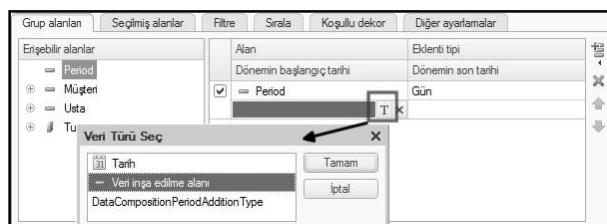
Bundan sonra bu tamamlamak işlemi hangi dönem içinde yapılacağını belirtmek gereklidir.

Bir alt satırda bulunanlara bu dönemin ilk tarihini ve son tarihini girebiliriz. Fakat kesin tarihler bizim için uygun değildir çünkü kullanıcı serbest bir şekilde rapor dönemini belirleyebilir. Ve biz sadece kullanıcının belirdiği dönem için tarih tamamlaması yapılacağını istiyoruz.

Bunu ayarlamak için Dönemin ilk tarihi alanında çift tıklayarak temizleme butonuna tıklayalım.

Ondan sonra veri tipi seçme butonuna basarak bu alanda yansıtılacak veri türünü seçebileceğiz.

Veri inşa edilme alanını seçelim (Resim 13.70).

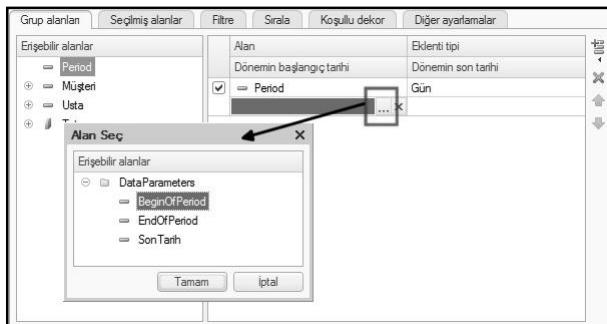


Resim 13.70. Veri türü seçme

Tamam tıklayalım.

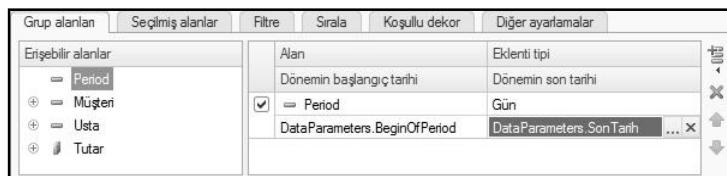
Şimdi veri giriş alanında seçme butonuna tıklayalım ve ekrana gelen pencereden *BeginOfPeriod* parametresini seçelim (Resim 13.71).

Tamam butona tıklayalım.



Resim 13.71. Alan seçme

İkinci alan için benzer şeklinde tarih *SonTarih* parametresinden alınacağını belirleyelim (Resim 13.72).



Resim 13.72. “Period” grubun ayarları

1C: İşletme ortamında

1C:İşletme sistemini hata ayıklama biçiminde çalıştırıyalım ve *Ustalar Satış Raporunu* 19.04.2011 – 26.04.2011 dönemi için oluşturalım (Resim 13.73).

Ustalar satış raporu			
Rapor bigemi: Genel	Seçenekler		
Olustur		Ayarlar...	
<input checked="" type="checkbox"/> İlk tarih	19.04.2011	Tüm eylemler	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/> Son tarih	26.04.2011		
Ustalar Satış Raporu			
Veri parametreleri:	İlk tarih: 19.04.2011	Son tarih: 26.04.2011	
Usta	Dönen	Müşteri	Tutar
Toplam			239,70
Ahmet Yemaz			85,00
	19.04.2011 00:00:00		
	20.04.2011 00:00:00		
	21.04.2011 00:00:00		
	22.04.2011 00:00:00		
	23.04.2011 00:00:00		
	24.04.2011 00:00:00		85,00
	25.04.2011 00:00:00	Ayşe Hanım	85,00
	26.04.2011 00:00:00		
Ali İşçi			107,20
	19.04.2011 00:00:00		
	20.04.2011 00:00:00		
	21.04.2011 00:00:00		
	22.04.2011 00:00:00		
	23.04.2011 00:00:00		
	24.04.2011 00:00:00		107,20
	25.04.2011 00:00:00	Mustafa Bey	107,20
	26.04.2011 00:00:00		
Mehmet Uzman			47,50
	19.04.2011 00:00:00		
	20.04.2011 00:00:00		
	21.04.2011 00:00:00		

Resim 13.73. Rapor oluşturma sonucu

Yeni rapor seçeneği

Belli bir dönemde ustaların çalışmalarını analiz etmek için aynı veri daha hoş görünür bir şekilde yansıtmak gerekebilir.

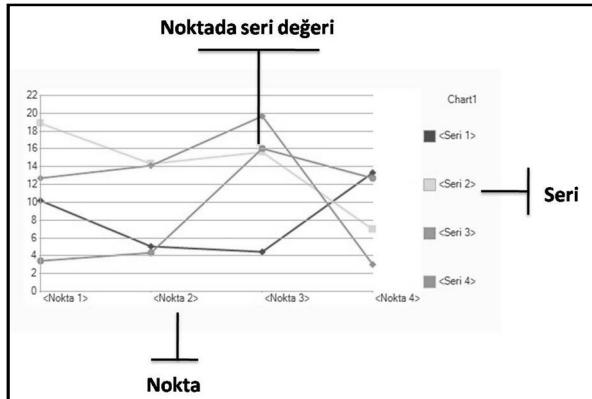
Örneğin, müdürlük maaş hesaplarını yaptığından, personelin çalışma performansı ile ilgili bir grafiğe ihtiyaç olabilir.

Bu yüzden, verileri grafik şeklinde yansıtacak olan raporun ikinci seçenekini geliştireceğiz. Bu grafikte personellerin firmaya kazandıkları tutarlar yansıtılacaktır. Seçenek adı – **UstalarSatışTutarları**.

Grafik

Tablolarda ve formlarda farklı diyagram ve grafikleri yerleştirmek için grafik öğeleri kullanılır.

Diyagram noktalarda, serilerden ve noktalardaki seri değerlerinden ibarettir (Resim 13.74).

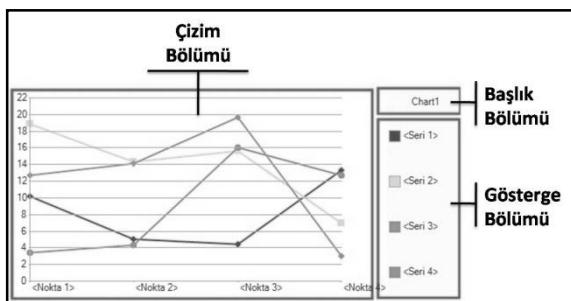


Resim 13.74. Grafik örneği

Genelde, *nokta* olarak dönem veya nesneler kullanılır, onlar için karakteristik değerlerini alıyoruz. *Seri* olarak değerleri lazım olan karakteristikler kullanılır. *Seri* ve *nokta* kesiştiğinde grafik *değeri* bulunur.

Örneğin, aylık malzeme/hizmet satış raporu böyle bir yapıya sahip olur; aylar – noktalar, malzeme/hizmet – seriler ve satış tutarları – değerler.

Kaynak dili olarak grafik üç bölüme sahiptir, bu bölümler sayesinde grafik dekor tanımlamaları yapılmaktadır; *Grafik çizim bölümü*, *Grafik başlık bölümü* ve *Grafik gösterge bölümü* (Resim 13.75).



Resim 13.75. Grafik bölümleri

Grafik rapor yapısına ayrı bir öğe olarak dahil edilebilir. Ustalar Satış Raporu raporunun bir sonraki ayarlarında grafiği veri inşa edilme şemasında kullanacağımız.

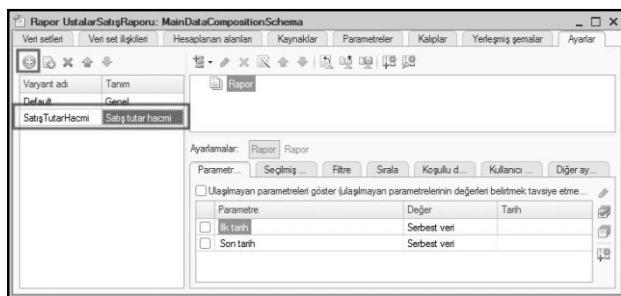
Tasarımcı ortamında

Tasarımcıya geri dönüp veri inşa edilme şemasını **Ayarlar** sekmesinde açalım.

Pencererin sol tarafında rapor seçenekler listesi bulunur.

Rapor ayarlarını ilk defa oluşturduğumuz zaman veri inşa edilme sistemi otomatik olarak *Genel* seçeneğini oluşturmuştu. Bu seçenek şu anda seçenek listesinde bulunur.

Yeni seçeneği eklemek için bu listenin araç çubuğundaki **Ekle** butona tıklayalım. Seçen adı – **SatışTutarHacmi** (Resim 13.76).



Resim 13.76. Yeni ayarlar seçeneğini ekleme

Raporun yapısı ve ayarlar sıfırlanmış olduğunu görüyoruz.

Tüm ayarlar silinmedi, görünmez oldular çünkü tüm ayarlar Genel seçeneğine aittir.

Raporun birden fazla seçenek varsa, sadece seçilmiş seçenek ile geliştirme ve değiştirme işlemi uygulanabilir. Bu sırada veri inşa edilme şemasındaki diğer veriler olduğu gibi kaldır (kaynaklar, parametreler, veri setleri). Rapor için veriler aynı sorgudan alınacaktır. Sadece raporun görünüş şeklini tanımlayan ayarlar değişir.

Rapor yapı ağaçına grafik ekleyelim. Bunu yapmak için rapor yapı ağaçının *Rapor* kökünü işaretleyerek sağ tuş menüsünden grafik ekleyelim (Resim 13.77).



Resim 13.77. Rapor yapısına grafik ekleme

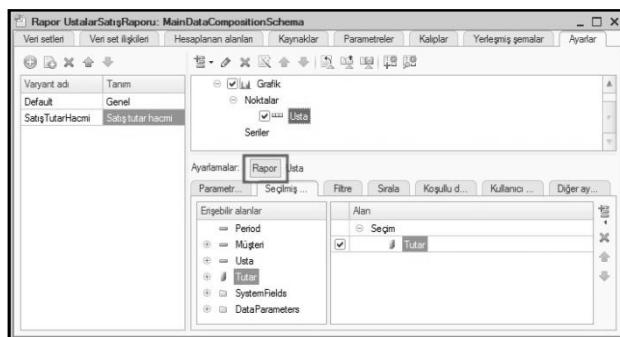
Noktalar grubu seçip **Usta** alanına göre bir grup ekleyelim. Grafik serilerini değişiksiz bırakalım.

Toplam satış tutarında ustaların getirdiği payları göstermek için kullanacağımız grafik türü için serileri belirtmek gerek yok. Sadece noktaları tanımlamak yeterlidir.

Grafik değeri olarak her zaman raporun kaynaklarından biri kullanılabilir. Bizde sadece bir tane kaynak vardır – **Tutar** (kaynak alanı özel yeşil simge ile işaretlenir).

Seçilen alanlar sekmesine geçip, raporun genel ayarlama seviyesine geçip (penceremim orta bölümünde *Rapor* butonun tıklayarak) **Tutar** alanın seçilen alan listesine ekleyelim.

Rapor yapısı aşağıdaki gibi şecline sahip olmalıdır (Resim 13.78).



Resim 13.78. Rapor yapısı ve grafik ayarları

DİKKAT!

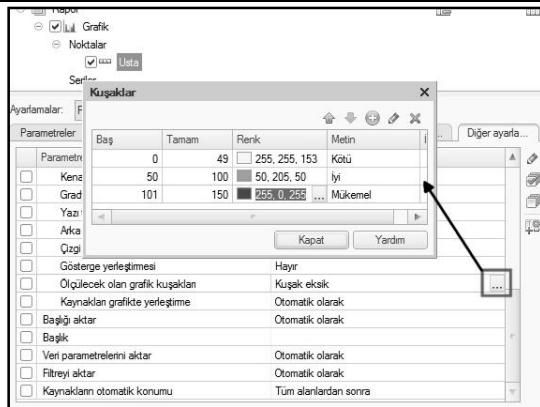
Grafikte kesinlikle raporun kaynağı yansıtılmalıdır, aksi halde program hata verir.

Diğer ayarlamalar sekmesinde grafik türü seçelim - **Ölçülecek olan** (Resim 13.79).

Parametreler	Seçilmiş ala...	Filtre	Sırala	Koşullu dekor	Kullanıcı aya...	Diğer ayarla...
Parametre	Değer					
<input type="checkbox"/> Alan bağlı türü	Otomatik olarak					
<input checked="" type="checkbox"/> Grafik türü	<input checked="" type="radio"/> Ölçülecek olan	...				
<input type="checkbox"/> Temel değer						
<input type="checkbox"/> Temel değerleri atla	True					
<input type="checkbox"/> İmza bileşimi	Seri%Yüze					
<input type="checkbox"/> Ağırma bileşimi	Hayır					
<input type="checkbox"/> Veri tablosunu göster	False					

Resim 13.79. Grafik tür ayarları

Grafik özellikler listesinin *Ölçülecek olan* grafik kuşakları parametresinde grafik seviyelerini tanımlayalım – Kötü, İyi ve Mükemmel (Resim 13.80).



Resim 13.80. Grafik ayarları

Son olarak *İlkTarih* ve *SonTarih* parametrelerini kullanıcı ayarlarına dahil edelim ve düzeltme biçimini *Hızlı* olarak belirtelim.

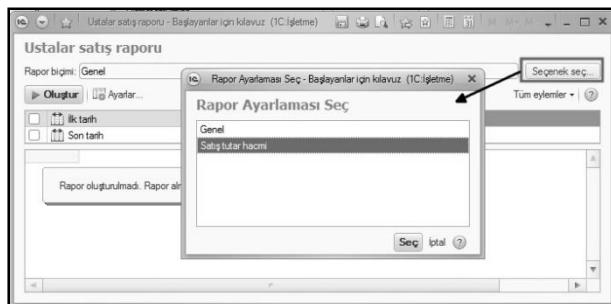
DİKKAT!

Her rapor seçenekleri için kullanıcı ayarları sıfırdan tanımlamak gereklidir, çünkü her raporu seçeneklerinin kullanıcı ayarları ona özeldir.

1C: İşletme ortamında

1C: İşletme sistemini hata ayıklama biçiminde çalıştıralım ve Hizmetler bölümünün eylem çubuğuunda *Ustalar satış raporu* komutu uygulayalım.

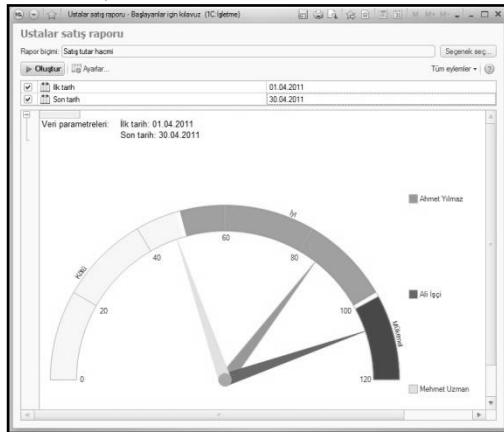
Ekrana gelen raporu penceresinde *Seçeneği seç* butona tıklayalım (Resim 13.81).



Resim 13.81. Rapor seçeneklerini seçme

Rapor seçenek penceresinde iki seçenek görüyoruz – Genel ve demin oluşturduğumuz Satış tutar hacmi seçeneği. Onu seçip *Seç* butonuna tıklayalım.

Raporlama dönemi 01.04.2011 – 30.04.2011 olarak belirleyelim ve raporu oluşturalım (Resim 13.82).



Resim 13.82. Rapor oluşturma sonucu

Periyodik bilgi kayıt tablosundan güncel veri alımı

Bir sonraki rapor – Hizmet listesi. Bu raporda “Aspirin usta” firması hangi hizmetleri verdiği ve hizmetlerin fiyatları ne olsulu gösterilecektir. (Resim 13.83).

Hizmet listesi		
Rapor başlığı	Genel	Seçenekler
<input checked="" type="checkbox"/> Olustar	<input type="checkbox"/> Ayarlar...	Tüm eylüler
<input checked="" type="checkbox"/> Rapor tarihi		
07.04.2011		
Hizmet Listesi		
Veri parametreleri: Rapor tarihi: 07.04.2011		
Hizmet grubu	Hizmet	Fiyat
Çamaşır makineleri	Su bağlama	40,00
	Elektrik bağlama	40,00
Televizyonlar	Tehnis	10,00
	Yerli marka televizyon tamiri	30,00
	Yabancı marka televizyon tamiri	40,00

Resim 13.83. Rapor sonucu

Bu rapor örneği ile periyodik bilgi kayıt tablosundan güncel veri nasıl alınıldığı ve hiyerarşik kart listelerinin rapora nasıl aktarılıldığı ile ilgili bilgiye sahip olacağız.

Tasarımcı ortamında

Yeni *Rapor* konfigürasyon nesnesini ekleyelim.

Raporu **HizmetListesi** olarak adlandırip veri inşa edilme şema yapılandırıcısını çalışıralım.

Yeni Veri seti – sorgu ekleyelim ve sorgu yapılandırıcısını çağırıralım.

Veri seti için sorgu

Sorgu için veri kaynağı olarak *MalzemeHizmet* kart listesinin nesnel (referanslı) tablosunu ve *Fiyatlar.SliceLast* bilgi kayıt tablosunun sanal tablosunu seçelim.

Sorguda isim çalışmasını önlemek için *MalzemeHizmet* tablosunu *krtMalzemeHizmet* olarak yeniden adlandıralım.

Bunu yapmak için onu Tablolar bölümünde aktif edip sağ tuş menüsünden *Tabloyu yeniden adlandır* maddesini seçelim

Sanal tablo parametreleri

FiyatlarSliceLast sanal tablosunun parametrelerini girme penceresini açıp dönem *RaporDönemi* parametresinden aktarılacağını belirleyelim.

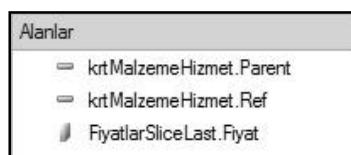
Bunu yapmak için *Tablolar* bölümünde tabloyu işaretleyip *Sanal tablo parametreleri* butonuna tıklayalım (Resim 13.84).



Resim 13.84. Sanal tablo parametreleri

Ondan sonra listeden aşağıdaki alanları seçelim (Resim 13.85).

- *krtMalzemeHizmet.Parent*
- *krtMalzemeHizmet.Ref*
- *FiyatlarSliceLast.Fiyat*

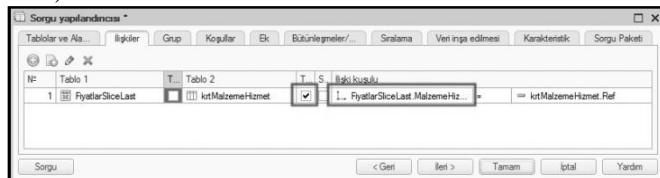


Resim 13.85. Seçilen alanlar

Tabloları soldan birleştirme

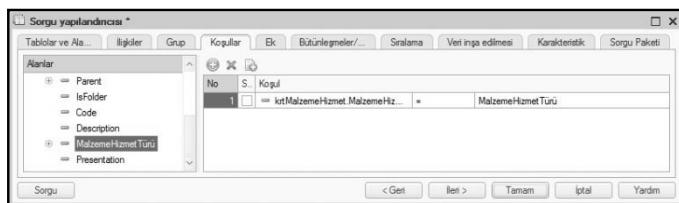
İlişkiler sekmesine geçip İlişki koşulu alanında bilgi kayıt tablosunun *MalzemeHizme* boyutunun *MalzemeHizmet* kart listesinin referansına eşit olduğunu belirleyelim.

Kayıt tablosunun Tüm işaretini kaldırıp, kart listesi için işaretleyelim (Resim 13.86).



Resim 13.86. Sorguda tablo ilişkileri

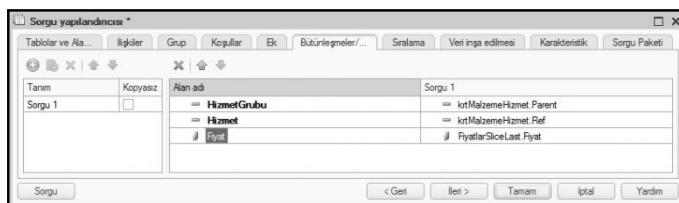
Koşullar sekmesinde MalzemeHizmet kart liste öğelerini seçme koşulu tanımlayalım – seçilen öğeler **MalzemeHizmetTürü** parametresinde aktarılan malzeme hizmet türünden olacaktır (Resim 13.87).



Resim 13.87. Öğe seçme koşulları

Alan eşanlamlıları

Bütünleşmeler/Eşanlamlılar sekmesinde *Parent* alanı için **HizmetGrubu** ve *Ref* alanı için **Hizmet** eşanlamlılarını tanımlayalım (Resim 13.88).



Resim 13.88. Bütünleşmeler/Eşanlamlılar

Bu aşamada sorgu oluşturma işlemi sona erdi. *Tamam* butonuna tıklayalım.

Sorgu metni analizi

Sorgu yapılandırıcısı tarafından oluşturulan sorgu metnine bakalım (Liste 13.13).

Liste 13.13. Sorgu metni

```

SELECT
    krtMalzemeHizmet.Parent AS HizmetGrubu,
    krtMalzemeHizmet.Ref AS Hizmet,
    FiyatlarSliceLast.Fiyat
FROM
    Catalog.MalzemeHizmet AS krtMalzemeHizmet
    LEFT JOIN
    InformationRegister.Fiyatlar.SliceLast(&RaporTarihi, ) AS FiyatlarSliceLast
        ON FiyatlarSliceLast.MalzemeHizmet = krtMalzemeHizmet.Ref
    WHERE
        krtMalzemeHizmet.MalzemeHizmetTürü = &MalzemeHizmetTürü

```

Bu sorguda kullanılan tüm yapıları biliyoruz.

Veri inşa edilme şemasını düzeltme işlemlerine geçelim.

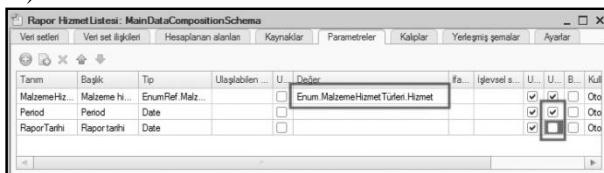
Kaynaklar sekmesinde butona basarak tek ulaşılabilir kaynağını seçelim – *Fiyat*.

Parametreler

Parametreler sekmesinde *MalzemeHizmetTürü* parametresinin değeri **Enum.MalzemeHizmetTürleri.Hizmet** olarak belirleyelim.

RaporTarihi parametresinin ulaşılabilirlik sınırlama işaretini kaldırıralım. Bu parametrenin tip alanında tarih bileşenini belirleyelim – *Tarih*.

Period parametresi için ulaşılabilirlik sınırlama işaretini yerleştirelim (Resim 13.89).



Resim 13.89. Veri inşa edilme parametreleri

Ayarlar

Rapor yapısını oluşturmaya başlayalım. *Ayarlar* sekmesine geçip *HizmetGrubu* alanına göre grup ekleyelim. Gruplama türü **Hiyerarşi** olarak belirleyelim (Resim 13.90).



Resim 13.90. Alan ve türü seçme

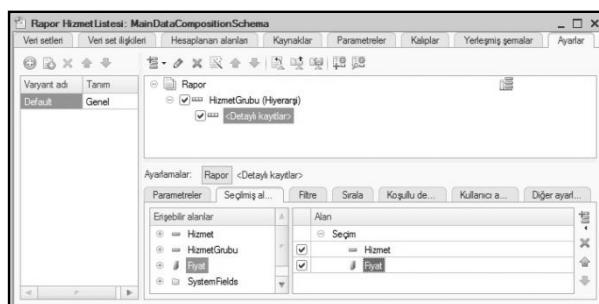
NOT

Şu ana kadar varsayılan hiyerarşî türü kullanmıştık – Hiyerarşisiz. Rapor grupları için aşağıdaki hiyerarşî türleri mevcuttur;

- Hiyerarşisiz – grupta sadece hiyerarşik olmayan kayıtlar gösterilir.
- Hiyerarşî – grupta hem hiyerarşik ve hiyerarşik olmayan kayıtlar gösterilir.
- Yalnız hiyerarşî – grupta sadece hiyerarşik kayıtlar gösterilir.

Bu grup içinde daha bir tane grup oluşturalım. Alt grup için alan eklemeyelim. Grup detaylı kayıtları içerecektir.

Seçilmiş alanlar sekmesine geçip, raporda Hizmet ve Fiyat alanlar gösterileceğini tanımlayalım (Resim 13.91).

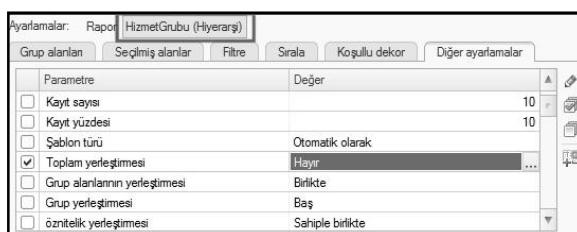


Resim 13.91. Rapor yapısı ve alanları

Şimdi Diğer ayarlamalar sekmesinde raporun görünüş şeklini ayarlayalım.

Oluşturduğumuz rapor sadece verilen hizmetlerin fiyatları ile ilgili bilgi vereceği için, Fiyat kaynağını toplamları her grup için ve rapor toplamında çıkartmak mantıklı değildir. Genel toplamları devre dışı bırakmak için *Genel toplamların yataya göre yerleştirilmesi* parametresinin değeri **Hayır** olarak tanımlayalım (Resim 13.93).

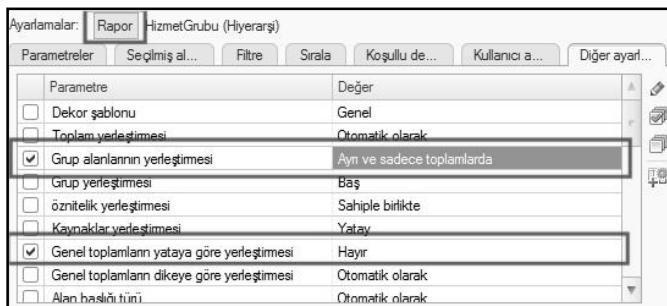
HizmetGrubu grubunun ayarlarına geçelim. Bu grubun *Toplam yerleştirilmesi* parametresini **Hayır** olarak belirleyelim (Resim 13.92).



Resim 13.92. “HizmetGrubu” grubu için genel toplam göstereme ayarları

Raporun genel ayarlarına geri dönelim.

Grup alanlarının yerlestirmesi parametresi için **Ayrı ve sadece toplamlarda** değeri belirleyelim (böylece raporumuz daha kolay okunabilir olacak). Raporun başlığını tanımlayalım – **Hizmet Listesi**.



Resim 13.93. Rapor ayarlarında toplamları gösterme ayarları

Son olarak Rapor tarihi parametresini kullanıcı ayarlarına dahil edelim ve düzeltme biçimini *Hızlı* olarak tanımlayalım.

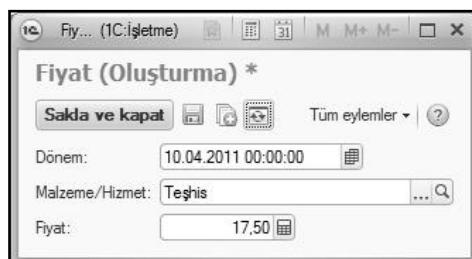
Raporun hangi alt sistemlerde bulunacağını işaretleyelim.

Veri inşa edilme şema yapılandırıcısını kapatıp *HizmetListesi* rapor konfigürasyon nesnesini düzeltme penceresinin *Alt sistemler* sekmesine geçip *Hizmetler* ve *Muhasebe* alt sistemlerini işaretleyelim.

1C: İşletme ortamında

1C:İşletme sisteminin hata ayıklama biçiminde çalışıralım ve ilk önce Fiyatlar bilgi kayıt tablosunu açalım.

Teşhis hizmeti için bir tane fiyat daha ekleyelim – 10.04.2011 tarihindeki hizmetin yeni fiyatı 17,50 TL'dir (Resim 12.94). Bu raporu test etme olanağını sağlar.



Resim 13.94. Teşhis hizmeti için yeni fiyat tanımlama

Şimdi Hizmet listesi raporunu 07.04.2011 tarihi için oluşturalım (Resim 13.95).

Hizmet listesi

Rapor türü: Genel

Oluştur | Ayarlar... | Seçenek seç... | Tüm eylemler | ?

Rapor tarihi: 07.04.2011

Hizmet Listesi

Veri parametreleri: Rapor tarihi: 07.04.2011

Hizmet grubu	Hizmet	Fiyat
Hizmetler	Çamaşır makineleri	40,00
	Su başlama Elektrik başlama	40,00
Televizyonlar	Tehsil	10,00
	Yerli marka televizyon tamiri	30,00
	Yabancı marka televizyon tamiri	40,00

Resim 13.95. Rapor oluşturma sonucu

Raporumuz Teşhis hizmetinin doğru fiyatı yansıttı – 10 TL.

Raporu tekrar oluşturalım. Fakat bu kez Rapor tarihi 10.04.2011 olarak tanımlayalım (Resim 13.96).

Hizmet listesi

Rapor türü: Genel

Oluştur | Ayarlar... | Seçenek seç... | Tüm eylemler | ?

Rapor tarihi: 10.04.2011

Hizmet Listesi

Veri parametreleri: Rapor tarihi: 10.04.2011

Hizmet grubu	Hizmet	Fiyat
Hizmetler	Çamaşır makineleri	40,00
	Su başlama Elektrik başlama	40,00
Televizyonlar	Tehsil	17,50
	Yerli marka televizyon tamiri	30,00
	Yabancı marka televizyon tamiri	40,00

Resim 13.96. Rapor oluşturma sonucu

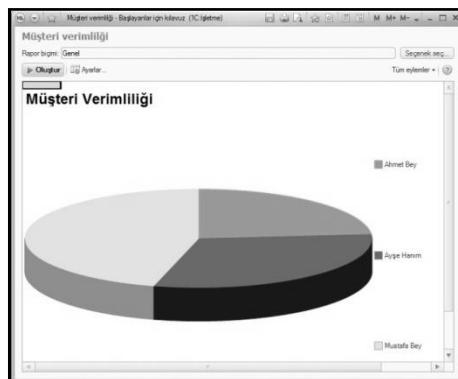
Gördüğünüz gibi Teşhis hizmetin fiyatı 17.50 TL olarak değişti.

Bu rapor örneği ile periyodik bilgi kayıt tablosundan son değerleri nasıl almak gerektiğini ve kart listesini hiyerarşisini nasıl gösterilebileceğini açıklamıştık.

Raporda hesaplanan alanı kullanma

Sıradaki rapor – **Müşteri verimliliği**. Bu rapor grafik olarak “Aspirin usta” şirketi çalışma sürecinde müşteriden elde edilen karı gösterecek.

Bu rapor örneği ile hesaplanan alan kullanımını ve sonucu yuvarlak diyagram olarak gösterme olanağını anlatacağız.



Tasarımcı ortamında

Yeni *Rapor* konfigürasyon nesnesini ekleyelim.

Raporu **MüşteriVerimliliği** olarak adlandırap veri inşa edilme şema yapılandırıcısını çalıştıralım.

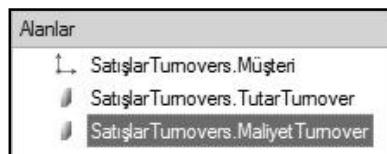
Yeni *Veri seti – sorgu* ekleyelim ve sorgu yapılandırıcısını çağırıralım.

Veri seti için sorgu

Sorgu için veri kaynağı olarak *Satışlar.Turnovers* sanal birikim kayıt tablosunu seçelim.

Bu tablodan aşağıdaki alanları seçelim (Resim 13.98);

- SatışlarTurnovers.Müşteri
- SatışlarTurnovers.TutarTurnover
- SatışlarTurnovers.MaliyetTurnover



Resim 13.98. Seçilen alanlar

Bütünleşmeler/Eşanlamlılar sekmesinde *Tutar Turnover* alanı için **Tutar**, *Maliyet Turnover* alanı için **Maliyet** eşanlamlıları tanımlayalım

Bu aşamada soru oluşturma işlemi tamamlanmıştır. *Tamam* butona tıklayalım.

Sorgu metninde yeni ve anlaşılmaz bir şey olmadığından veri inşa edilme şemasını tasarlamaya devam edelim.

Hesaplanan alanlar

Bu aşamada veri setinde bulunmayan bir alanı göstermemiz gereklidir. Önceden biz sadece veri setinde bulunan alanları kullanıyorduk. Şimdi ise müşteriden alınan kar tutarını gösterebilmemiz için yeni bir alanı oluşturmayı gereklidir. Kar tutarı hesaplama mantığı budur; Kart tutarı = Tutar – Maliyet.

Bunu yapmak için veri inşa edilme sisteminde hesaplanan alanları tanımlama olanağı mevcuttur.

Hesaplanan alanlar, değerleri belli bir formüle göre hesaplanan ekstra alanlardır.

Hesaplanan alanlar sekmesine geçip **Ekle** butona tıklayarak yeni alan ekleyelim.

Bu alan için ad tanımlayalım – Kar ve İfade sütuna hesaplanan alanını hesaplama ifadesini yazalım (Liste 13.14).

Liste 13.14. “Kar” hesaplanan alanının hesaplama ifadesi

Tutar - Maliyet

Raporun başlık satırlarda bulunacak hesaplanan alanının başlığı otomatik olarak oluşturulacak (gerektiğinde değiştirilebilir). (Resim 13.99).

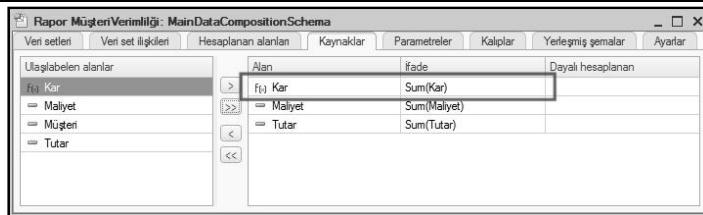


Resim 13.99. Hesaplanan alanı oluşturma

Hesaplanan alanı grup veya genel toplamlarını hesaplayabilmek için rapor kaynaklarına eklenebilir.

Kaynaklar

Kaynaklar sekmesinde butona tıklayarak raporun tüm ulaşılabilir kaynaklarını seçelim. Gördüğümüz gibi Kar alanı da kaynak listesine eklenmiştir (Resim 13.100).



Resim 13.100. Veri inşa edilme kaynakları

Ayarlar

Ayarlar sekmesinde rapor yapı ağacına grafik ekleyelim.

Bunu yapmak için araç çubuğuında **Ekle** butonuna tıklayıp grafik ekleyelim (Resim 13.101).



Resim 13.101. Rapor yapısına grafik ekleme

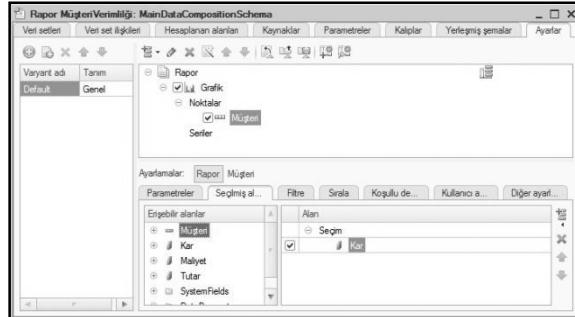
Noktalar grubu seçip Müşteri alanına göre yeni bir grup ekleyelim.

Grafik serileri değişikiksiz bırakalım.

Müşteri verimliliği göstermek için yuvarlak grafik iyi görünür. Bu grafik için sadece noktaları tanımlamak yeterlidir, bu yüzden bir serileri tanımlamıyoruz.

Grafik değerleri olarak her zaman rapor kaynaklarından biri belirlenir. Seçilen alanlar sekmesine geçip raporda göstermek için **Kar** alanını seçelim.

Raporun yapısı aşağıdaki gibi görülmeli (Resim 13.102).



Resim 13.102. Rapor yapısı ve grafik ayarları

Diger ayarlamalar sekmesinde grafik tipini seçelim - **3-B Pasta** ve rapor başlığını belirleyelim – **Müşteri Verimliliği**.

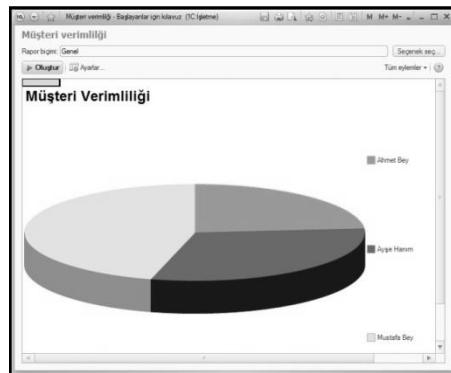
Son olarak raporun hangi alt sistemlerde bulunacağını işaretleyelim.

Veri inşa edilme şema yapılandırıcısını kapatıp *Müşteri Verimliliği* rapor konfigürasyon nesnesini düzeltme penceresinin *Alt sistemler* sekmesine geçip *Hizmetler* ve *Muhasebe* alt sistemlerini işaretleyelim.

1C: İşletme ortamında

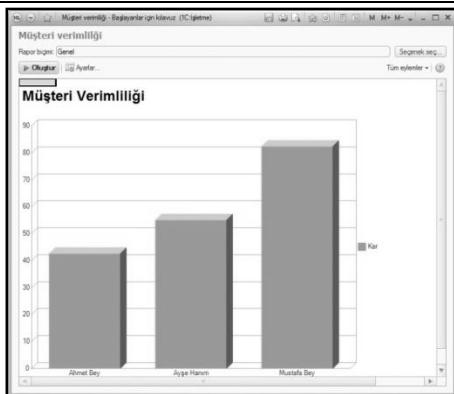
1C:İşletme sisteminde hata ayıklama biçiminde çalışıralım ve *Muhasebe* bölümünün eylem çubuğuunda *Müşteri verimliliği* komutu uygulayalım. *Oluştur* tıklayalım.

Verilen hizmetlerden elde edilen müşteri bazlı kar bilgileri pasta grafiği şeklinde görebiliyoruz (Resim 3.103).



Resim 13.103. Pasta grafiği

Tasarımcıya geri dönüp grafik türü **Yığılmış 3-B Sütun** olarak değiştirelim. Raporu yeniden oluşturalım (Resim 13.104).



Resim 13.104. Sütun grafiği

Böylece rapor verileri görsel şeklinde göstermek için grafikler nasıl kullanılabildiğini göstermiş olduk.

Verileri tabloda gösterme

Üniversal raporu örneği ile verileri tabloda gösterilmesini öğreneceğiz (Resim 13.105).

Üniversal rapor				
Rapor başlığı: Genel	Tutar Devir			
Seçilen alanlar		Malzeme hizmet (Hyerarg)		
Üts				
Malzeme hizmet	Ahmet Yılmaz Tutar Devir	Ali İşçi Tutar Devir	Mehmet Uzman Tutar Devir	Toplam Tutar Devir
Hizmetler	40,00	70,00	40,00	150,00
Çamaşır makineleri		40,00	40,00	80,00
Elektrik bağlama		40,00		40,00
Su bağlama			40,00	40,00
Teknolojiler	40,00	30,00		70,00
Yabancı marka televizyon tamiri	40,00			40,00
Yerli marka televizyon tamiri		30,00		30,00
Malzemeler	45,00	37,20	7,50	89,70
Dijital		16,50	7,50	24,00
Elektrik kablo		1,50		1,50
Lastik hortum		15,00	7,50	22,50
Radyo parçaları	45,00	20,70		65,70
Flyback transformer GoldStar		20,00		20,00
Flyback transformer Samsung	45,00			45,00
Transistor Philips 2N2369			0,70	0,70
Toplam	85,00	107,20	47,50	239,70

Resim 13.105. Rapor sonucu

Kullanıcı işlerini kolaylaştmak için raporu mümkün olduğu kadarıyla üniversal yapmaya çalışırız. Yani kullanıcı seçenek içeriğini değiştirmeden (Tüm eylemler – Seçeneki değiştir menüsü) raporu değiştirme olanağına sahip olacaktır. Örneğin, satır ve sütun yerlerini değiştirmek, tablo hücrelerinde gösterilen verileri değiştirmek vs.

Tasarımcı ortamında

Yeni *Rapor* konfigürasyon nesnesini ekleyelim. Adı **UniversalRapor** belirleyelim ve veri inşa edilme şema yapılandırıcısın çalıştırıyalım.

Yeni *Veri seti – sorgu* oluşturup sorgu yapılandırıcısı çağırıyalım.

Veri seti için sorgu

Sorgu için veri kaynağı olarak *Satışlar.Turnovers* sanal birikim kayıt tablosunu seçelim.

Bu tablonun tüm alanları seçelim (Resim 13.106).



Resim 13.106. Seçilen alanlar

Sorgu metni analizi

Tamam butona tıklayalım ve yapılandırıcı tarafından oluşturulan sorguya bakalım (Liste 13.15).

Liste 13.15. Sorgu metni

```
SELECT
    SatışlarTurnovers.MalzemeHizmet,
    SatışlarTurnovers.Müşteri,
    SatışlarTurnovers.Usta,
    SatışlarTurnovers.MiktarTurnover,
    SatışlarTurnovers.TutarTurnover,
    SatışlarTurnovers.MaliyetTurnover
FROM
    AccumulationRegister.Satışlar.Turnovers AS SatışlarTurnovers
```

Kaynaklar

sekmesinde butonuna tıklayarak tüm ulaşılabilir rapor kaynaklarını seçelim.

Ayarlar

Ayarlar sekmesinde rapor yapı ağacına tablo ekleyelim.

Bunu yapmak için ayarlar penceresinin araç çubuğunda Ekle butona basıp tabloyu ekleyelim (Resim 13.107).



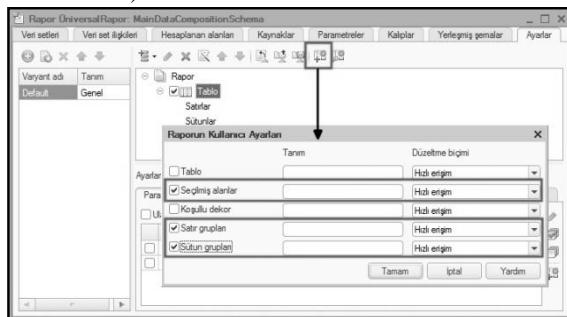
Resim 13.107. Rapor yapısına tablo ekleme

Burada bu tablonun satırları ve sütunları belirlemeyeceğiz, çünkü kullanıcı kendi isteklerine göre raporu oluşturabileceğine karar verildi.

Bunu sağlamak için rapor öğe ağacında *Tablo* öğesini secerék araç çubuğunda bulunan *Kullanıcı ayarları öğe özellikleri* butonuna tıklayalım.

Ekrana gelen pencerede tablonun kullanıcı ayarlarını tanımlayabiliriz.

Seçilmiş alanlar, *Satır grupları* ve *Sütun grupları* özellikleri için kullanma işaretini yerleştirelim ve düzeltme biçimini özelliğini *Hızlı erişim* olarak bırakalım (Resim 13.108).



Resim 13.108. Kullanıcı ayarları

Böylece, kullanıcı rapor oluşturmadan önce direkt rapor formundan görmek istediği satır gruplarını, sütun gruplarını ve seçilmiş alanlarını tanımlayabilecektir.

Son olarak raporun hangi alt sistemlerde bulunacağını işaretleyelim.

Veri inşa edilme şema yapılandırıcısını kapatıp *UniversalRapor* rapor konfigürasyon nesnesini düzeltme penceresinin *Alt sistemler* sekmesine geçip *Hizmetler* alt sistemini işaretleyelim.

1C: İşletme ortamında

1C: İşletme sistemini hata ayıklama biçiminde çalıştırıp *Hizmetler* bölümünün eylem çubuğundaki *Universal rapor* komutunu uygulayalım.

Şimdi Oluştur butonuna tıklaysak, rapor hiçbir veri görüntülemez, çünkü daha ne satır grupları, ne sütun grupları ne seçilmiş alanlar seçilmiş durumdadır. Bu hızlı kullanıcı ayarlarını dolduralım.

Seçilmiş alanlar satırında seçim butonuna tıklayıp ulaşılabilir alanlardan **Tutar Akış** seçelim.

Satırlar satırında seçim butonuna tıklayıp **MalzemeHizmet** alanına göre grup ekleyelim tipi **Hiyerarşî** olacaktır.

Sütunlar satırında seçim butonuna tıklayıp **Usta** alanına göre grup ekleyelim.

Oluştur butonuna tıklayalım.

Rapor aşağıdaki gibi görünecek (Resim 13.109).

Malzeme hizmet	Ahmet Yılmaz Tutar Devir	Ali İşçi Tutar Devir	Mehmet Uzman Tutar Devir	Toplam Tutar Devir
Hizmetler	40.00	70.00	40.00	150.00
Çamış makineleri	40.00	40.00	40.00	80.00
Elektrik bağlama	40.00	40.00	40.00	80.00
Su bağlama	40.00	40.00	40.00	80.00
Televizyonlar	40.00	30.00	40.00	70.00
Yerine marka televizyon tamiri	40.00	30.00	30.00	60.00
Yerine marka televizyon tamiri	40.00	30.00	30.00	60.00
Malzemeler	45.00	37.20	7.50	89.70
Diger	16.50	7.50	24.00	48.00
Elektrik kablo	1.50	1.50	1.50	4.50
Lastik hortun	15.00	20.70	22.50	58.20
Radyo parçları	45.00	20.00	20.00	65.00
Flyback transformer GoldStar	45.00	0.70	45.00	45.00
Flyback transformer Samsung	45.00	0.70	45.00	45.00
Transistor Philips 2N2369				
Toplam	85.00	107.20	47.50	239.70

Resim 13.109. Rapor sonucu

Şimdi Seçilen alanlar listesine **Maliyet Akış** ekleyelim.

Tablo satırlarına **MalzemeHizmet** alanı yerine **Müşteri** alanına göre grup ekleyelim.

Rapor oluşturulduğunda aşağıdaki gibi görünecektir (Resim 13.110).

Müşteri	Ahmet Yılmaz		Ali İşçi		Mehmet Uzman		Toplam	
	Tutar Devrir	Maliyet Devrir	Tutar Devrir	Maliyet Devrir	Tutar Devrir	Maliyet Devrir	Tutar Devrir	Maliyet Devrir
Ahmet Bey					47,50	5,00	47,50	5,00
Ayşe Hanım	85,00	30,00					85,00	30,00
Mustafa Bey			107,20	24,80			107,20	24,80
Toplam	85,00	30,00	107,20	24,80	47,50	5,00	239,70	59,80

Resim 13.110. Rapor sonuçu

Şimdi seçilen alanlar listesinden Maliyet Akış alanını kaldırıralım. Tablo satırlarında önceki gruptaki alanı **MalzemeHizmet** alanı ile değiştirelim türü **Yalnız hiyerarşi** olacaktır.

Tablo sütunlarında **Müşteri** alanına göre yeni grup ekleyelim ve bu grubu grup listesinde birinci sıraya yerlestirelim.

Sonuç olarak rapor aşağıdaki gibi görünecektir (Resim 13.111).

Malzeme hizmet	Ahmet Bey		Mehmet Uzman		Ayşe Hanım		Ahmet Yılmaz		Mustafa Bey		Ali İşçi		Toplam	
	Tutar Devrir	Tutar Devrir	Tutar Devrir	Tutar Devrir	Tutar Devrir	Tutar Devrir	Tutar Devrir	Tutar Devrir	Tutar Devrir	Tutar Devrir	Tutar Devrir	Tutar Devrir	Tutar Devrir	Tutar Devrir
Hizmetler	40,00		40,00		40,00		40,00		70,00		70,00		150,00	
Çamaşır makineleri	40,00		40,00						40,00		40,00		80,00	
Televizyonlar									30,00		30,00		70,00	
Malzemeler	7,50		7,50		45,00		45,00		37,20		37,20		69,70	
Diger	7,50		7,50						16,50		16,50		24,00	
Radyo parçaları					45,00		45,00		20,70		20,70		65,70	
Toplam	47,50		47,50		85,00		85,00		107,20		107,20		239,70	

Resim 13.111. Rapor sonuçu

Böylece, kullanıcı serbest bir şekilde *Satışlar* birikim kayıt tablosuna göre rapor oluşturabilecektir.

“Teori”. Sorgunun sanal tabloları

Bildiğiniz üzere, sorgu oluşturulduğunda platform bize veri kaynağı olarak bazı sanal tabloları sunmaktadır. İşlem gerçekleşme sırası ilgili satırı geldiğinde bu tablolar da sorgu sonucu olarak algılanırlar.

Genel olarak alınırsa, sistemin sanal tablo olarak sunduklarını geliştiricinin kendisi de elde edebilmektedir. Ancak bu verileri alma algoritması pek optimize edilmiş sayılmayacaktır. Bunun iki sebebi vardır:

İlk olarak, tüm sanal tablolar parametrelenmiş durumdadırlar. Yani geliştirici her bir tablo için parametre belirtebilmektedir. Ve bu parametreler sistem tarafından sanal tablo oluşturulduğunda kullanılacaktır.

Geliştiricinin sorgu metninde belirttiği sanal tablo parametreleri her zaman basit çözüm haline gelmeyebilir. Sanal tablo için belirtilen parametreye bağlı olarak, belirtilen parametreler için en uygun işlem gerçekleştirmek maksadı ile sistem **Distinct** anahtar kelimeli sorgular da oluşturabilmektedir.

İkinci olarak da, sistem tarafından sunulan verilerin tümü geliştiriciler için ulaşılabilir olmayıpabilir. Örneğin, sistem tarafından sunulan bilgi kayıt tablolarının sanal tablo bilgilerinin tamamı geliştiricilere sunulmuştur.

Birikim kayıt tablolarında ise durum biraz değişktir. Sistem otomatik olarak, sadece sanal tablo parametreleri için sorgu oluşturmuyor, bunun arasında belli dönem aralığı için hesaplanan toplamlar da dahildir. Bu ise sorgu oluşturma sürecinde geliştiriciler için ulaşılmazdır.

Tabii ki, geliştirici belli dönemlere ait birikimleri hesaplayabilir ve buna göre sistemin sunduğu sanal tabloları sorgu halinde de oluşturabilir, fakat bu oluşturulan sorgu az etkili olacaktır ve çok daha fazla zahmet gerektirecektir.

Sorular

- ✓ *Query kaynak kod nesnesi ne için kullanılır?*
- ✓ *Veri inşa edilme sistemi ne için kullanılır?*
- ✓ *Veri inşa edilme şeması ne için kullanılır?*
- ✓ *Veri inşa edilme ayarları ne için kullanılır?*
- ✓ *Gerçek ve sanal tablolar arasındaki fark nedir?*
- ✓ *Sorgu hangi parçalardan ibarettir? Hangi sorgu parçaları zorunlu parlalardır?*
- ✓ *Sorguy dilin genel sözdizim yapıları nelerdir?*
- ✓ *Sorgunun veri kaynakları nelerdir?*
- ✓ *Sorgu dilinde eş anlamlıları nedir?*
- ✓ *Sorgu parametreleri nedir?*
- ✓ *Sanal tablo parametreleri nedir?*
- ✓ *Soldan bağlantı nedir?*
- ✓ *Sorgu yapılandırıcısı nasıl kullanılır?*
- ✓ *Rapor için belli dönem içinde veriler nasıl alınabilir?*
- ✓ *Raporda veriler nasıl düzenlenlenebilir?*
- ✓ *Raporda birden fazla tablonun verileri nasıl kullanılabilir?*
- ✓ *Rapor yapısında gruplar nasıl kullanılır?*
- ✓ *Bilgi kayıt tablosunun en son veriyi nasıl elde alınabilir?*
- ✓ *Raporda hiyerarşik veri nasıl gösterilebilir?*
- ✓ *Gruplara göre toplamları ve genel toplamları nasıl tanımlanabilir?*
- ✓ *Grafiği içeren rapor nasıl oluşturulabilir?*
- ✓ *Veri inşa edilme sisteminde parametreler nasıl kullanılır?*
- ✓ *Veri inşa edilme sisteminde kaynaklar nedir?*
- ✓ *Veri inşa edilme hesaplanan alanlar nedir?*
- ✓ *Dönem grubunda, tüm tarihler nasıl gösterilir?*
- ✓ *Raporun kullanıcı ayarları nasıl tanımlanabilir?*

- ✓ “Hızlı” kullanıcı ayarları ve normal kullanıcı ayarları arasındaki fark nedir?
- ✓ Raporun kullanıcı ayarları takımı nasıl tanımlanabilir?
- ✓ Universal rapor nasıl oluşturulabilir?