SAP DANIŞMAN EĞİTİMİ

SAP [ABAB]

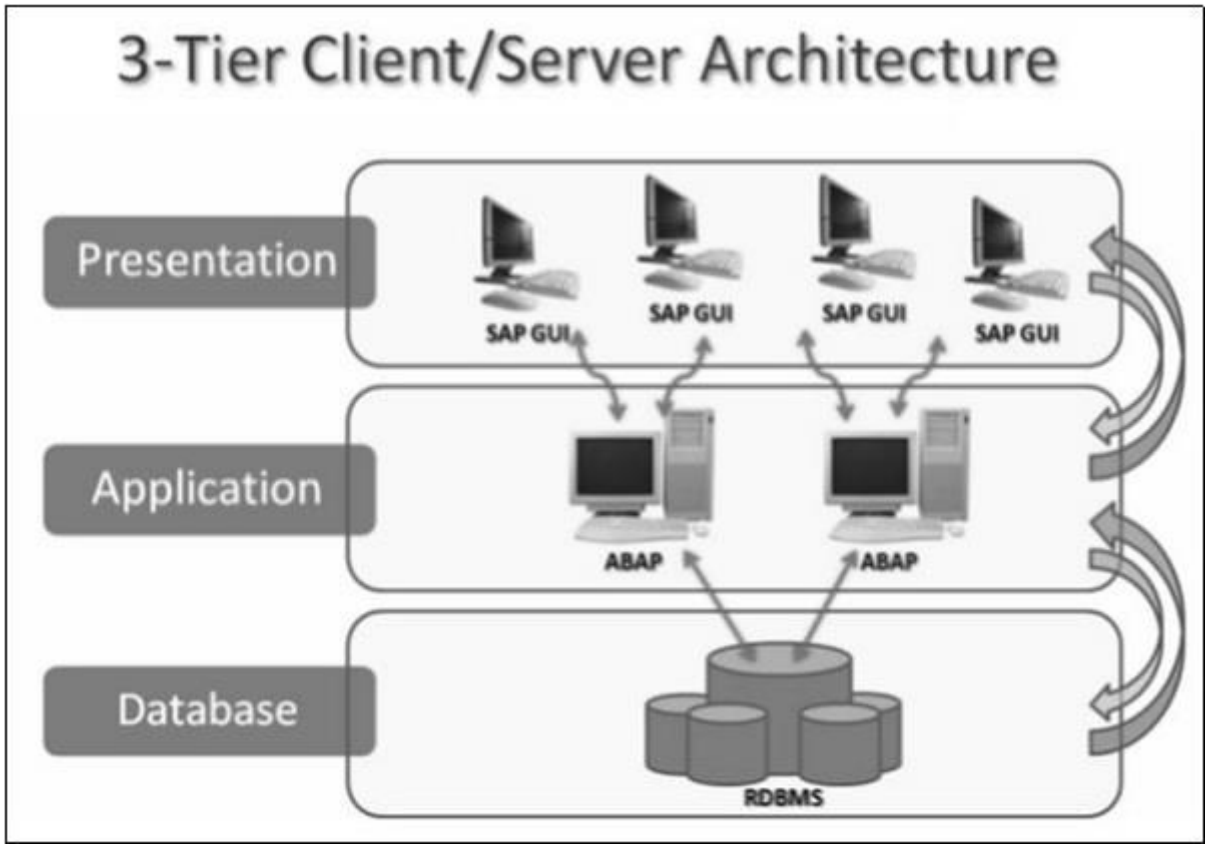
Salih KÜÇÜK - SAP Retail Consultant

SAP ABAP

Genel Bakış

ABAP, 4GL (4. nesil) bir dil olan Advanced Business Application Programming'in kısaltmasıdır. Şu anda, Java ile birlikte SAP uygulama sunucusu programlaması için ana dil olarak konumlandırılmıştır.

SAP sisteminin üst düzey mimarisiyle başlayalım. Tipik bir SAP sisteminin 3 katmanlı İstemci/Sunucu mimarisi aşağıdaki gibi gösterilmektedir.



Sunum **katmanı** , SAP sistemini kontrol etmek için kullanılabilecek herhangi bir giriş cihazından oluşur. Bu bir web tarayıcısı, bir mobil cihaz vb. olabilir. Tüm merkezi işlemler **Uygulama sunucusunda** gerçekleşir . Uygulama sunucusu kendi içinde yalnızca bir sistem değildir, ancak işleme sisteminin birden çok örneği olabilir. Sunucu, çoğunlukla performans nedenleriyle ve ayrıca güvenlik nedeniyle genellikle ayrı bir sunucuda tutulan **Veritabanı katmanıyla** iletişim kurar . Sunum katmanından Veritabanına kadar sistemin her katmanı arasında iletişim gerçekleşir ve ardından zinciri yedekler.

Not − ABAP programları, uygulama sunucusu düzeyinde çalışır. Yazılımın teknik dağıtımı, fiziksel konumundan bağımsızdır. Bu, temelde her üç seviyenin de bir bilgisayara üst üste kurulabileceği veya her seviyenin farklı bir bilgisayara veya sunucuya kurulabileceği anlamına gelir.

ABAP programları SAP veri tabanı içinde yer alır. SAP çekirdeğinin bir parçası olan çalışma zamanı sisteminin kontrolü altında yürütülürler. Çalışma zamanı sistemi, tüm ABAP ifadelerini işler, akış mantığını kontrol eder ve kullanıcı olaylarına yanıt verir.

Dolayısıyla, C++ ve Java'dan farklı olarak ABAP programları ayrı harici dosyalarda saklanmaz. Veritabanının içinde ABAP kodu iki biçimde bulunur -

- ABAP workbench araçlarıyla görüntülenebilen ve düzenlenebilen **kaynak kodu.**
- İkili bir temsil olan **oluşturulan kod .** Java'ya aşina iseniz, oluşturulan bu kod, Java bayt koduyla biraz karşılaştırılabilir.

Çalışma zamanı sistemi, tıpkı Java sanal makinesine benzer şekilde sanal bir makine olarak düşünülebilir. ABAP çalışma zamanı sisteminin önemli bir bileşeni, veritabanından bağımsız ifadeleri (Open SQL), temel alınan veritabanı (Native SQL) tarafından anlaşılan ifadelere dönüştüren veritabanı arayüzüdür. SAP çok çeşitli veritabanlarıyla çalışabilir ve aynı ABAP programı bunların hepsinde çalışabilir.

SAP ABAP - Ortam

Raporlar, genel ABAP ilkeleri ve araçlarına aşina olmak için iyi bir başlangıç noktasıdır. ABAP raporları birçok alanda kullanılmaktadır. Bu bölümde basit bir ABAP Raporu yazmanın ne kadar kolay olduğunu göreceğiz.

Merhaba ABAP

Yaygın "Merhaba Dünya" örneğiyle başlayalım.

Her ABAP ifadesi bir ABAP anahtar sözcüğü ile başlar ve bir nokta ile biter. Anahtar kelimeler en az bir boşlukla ayrılmalıdır. Bir ABAP deyimi için bir veya birkaç satır kullanıp kullanmadığınız önemli değildir.

SAP NetWeaver Uygulama Sunucusu ABAP ('AS ABAP' olarak da bilinir) ile birlikte verilen ABAP Araçlarının bir parçası olan ABAP Düzenleyicisini kullanarak kodunuzu girmeniz gerekir.

'AS ABAP', kendi veri tabanına, ABAP çalışma zamanı ortamına ve ABAP Editor gibi ABAP geliştirme araçlarına sahip bir uygulama sunucusudur. AS ABAP, donanımdan, işletim sisteminden ve veritabanından bağımsız bir geliştirme platformu sunar.

ABAP Düzenleyicisini Kullanma

Adım 1 − ABAP Editörüne gitmek için SE38 işlemini başlatın (bir sonraki bölümde ele alınacaktır). Birçok ABAP nesnesinden biri olan bir rapor oluşturmaya başlayalım.

Adım 2 − Editörün ilk ekranında, PROGRAM giriş alanında raporunuzun adını belirtin. Adı ZHELLO1 olarak belirtebilirsiniz. Öndeki Z, ad için önemlidir. Z, raporunuzun müşteri ad alanında bulunmasını sağlar.

Müşteri ad alanı, Y veya Z ön ekine sahip tüm nesneleri içerir. Müşteriler veya ortaklar nesneler oluşturduğunda (bir rapor gibi) bu nesneleri SAP nesnelerinden ayırmak ve nesnelerle ad çakışmalarını önlemek için her zaman kullanılır.

Adım 3 − Rapor adını küçük harflerle yazabilirsiniz, ancak editör büyük harfle değiştirecektir. Bu nedenle ABAP nesnelerinin adları büyük/küçük harf duyarlı değildir.

Adım 4 − Raporun adını belirledikten sonra OLUŞTUR düğmesine tıklayın. Bir açılır pencere ABAP: PROGRAM ÖZELLİKLERİ açılır ve raporunuz hakkında daha fazla bilgi sağlarsınız.

Adım 5 − Rapor türü olarak "Yürütülebilir Program"ı seçin, "İlk ABAP Raporum" başlığını girin ve devam etmek için KAYDET'i seçin. Daha sonra NESNE DİZİNİ GİRİŞİ OLUŞTUR penceresi açılacaktır. YEREL NESNE düğmesini seçin ve açılır pencere kapanacaktır.

İlk raporunuzu, REPORT ifadesinin altına WRITE ifadesini girerek tamamlayabilirsiniz, böylece tam rapor aşağıdaki gibi sadece iki satır içerir -

```
REPORT ZHELLO1.
WRITE 'Hello World'.
```

Raporu Başlatma

Raporu kaydetmek için klavyeyi (Ctrl + S) veya kaydet simgesini (komut alanının sağ tarafında) kullanabiliriz. ABAP geliştirmesi AS ABAP'ta gerçekleşir.

Raporu başlatmak, kaydetmek kadar basittir. ETKİNLEŞTİRME düğmesine tıklayın (sol taraf, başlat simgesinin yanında) ve DİREKT İŞLEME simgesini veya F8 işlev tuşunu kullanarak raporu başlatın. "Merhaba Dünya" çıktısıyla birlikte "İlk ABAP Raporum" başlığı da görüntülenir. İşte çıktı -

```
My First ABAP Report
Hello World
```

Yeni bir raporu etkinleştirmediğiniz veya mevcut bir raporda bir değişikliği etkinleştirmediğiniz sürece, bu onların kullanıcıları için geçerli değildir. Bu, diğer geliştiricilerin projelerinde kullandıkları nesneler üzerinde çalışabileceğiniz merkezi bir geliştirme ortamında önemlidir.

Mevcut Kodu Görüntüleme

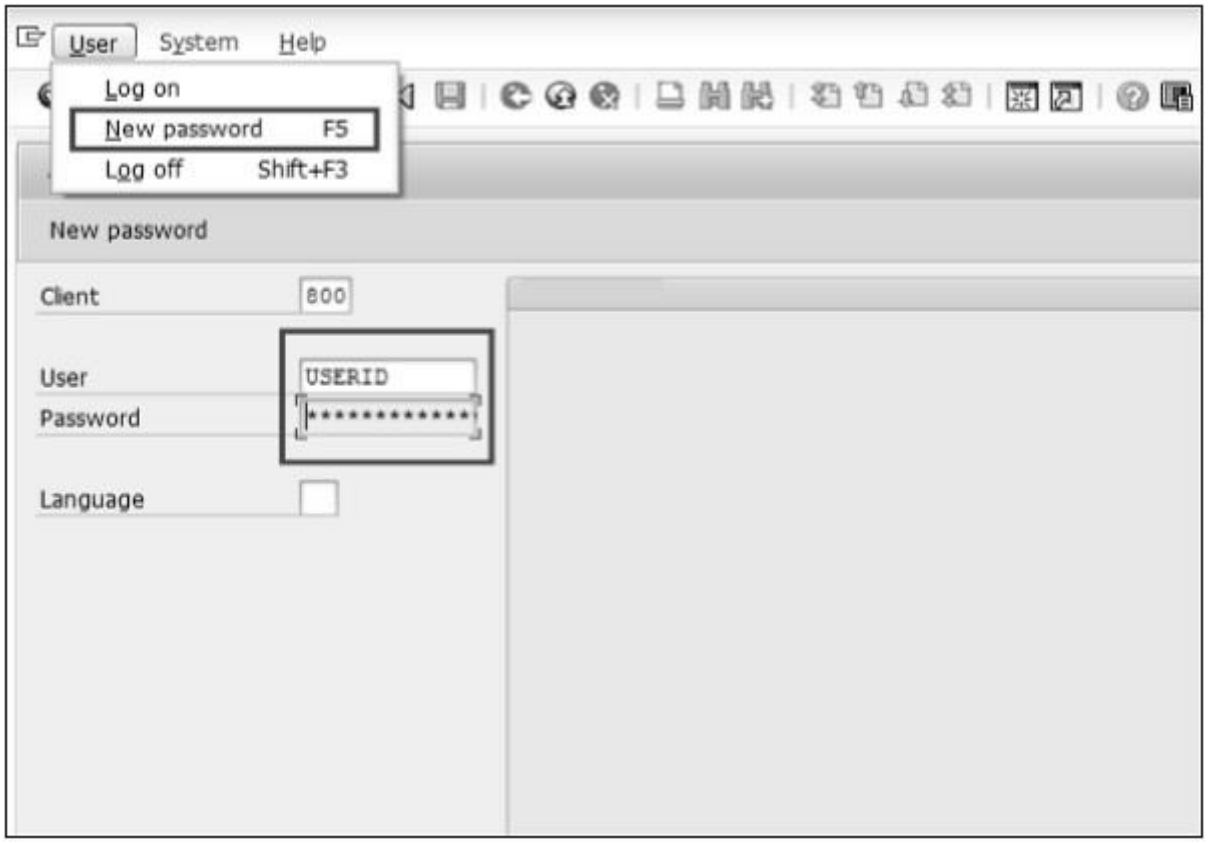
Program alanına bakarsanız ve ZHELLO1 değerine çift tıklarsanız, ABAP editörü raporunuzun kodunu gösterecektir. Buna İleri Navigasyon denir. Bir nesnenin adına çift tıklamak, o nesneyi uygun araçta açar.

SAP ABAP - Ekranda Gezinme

SAP ABAP'ı anlamak için Login, ABAP Editor, Logout vb. ekranlar hakkında temel bilgilere sahip olmanız gerekir. Bu bölüm, ekranda gezinme ve standart araç çubuğu işlevine odaklanmaktadır.

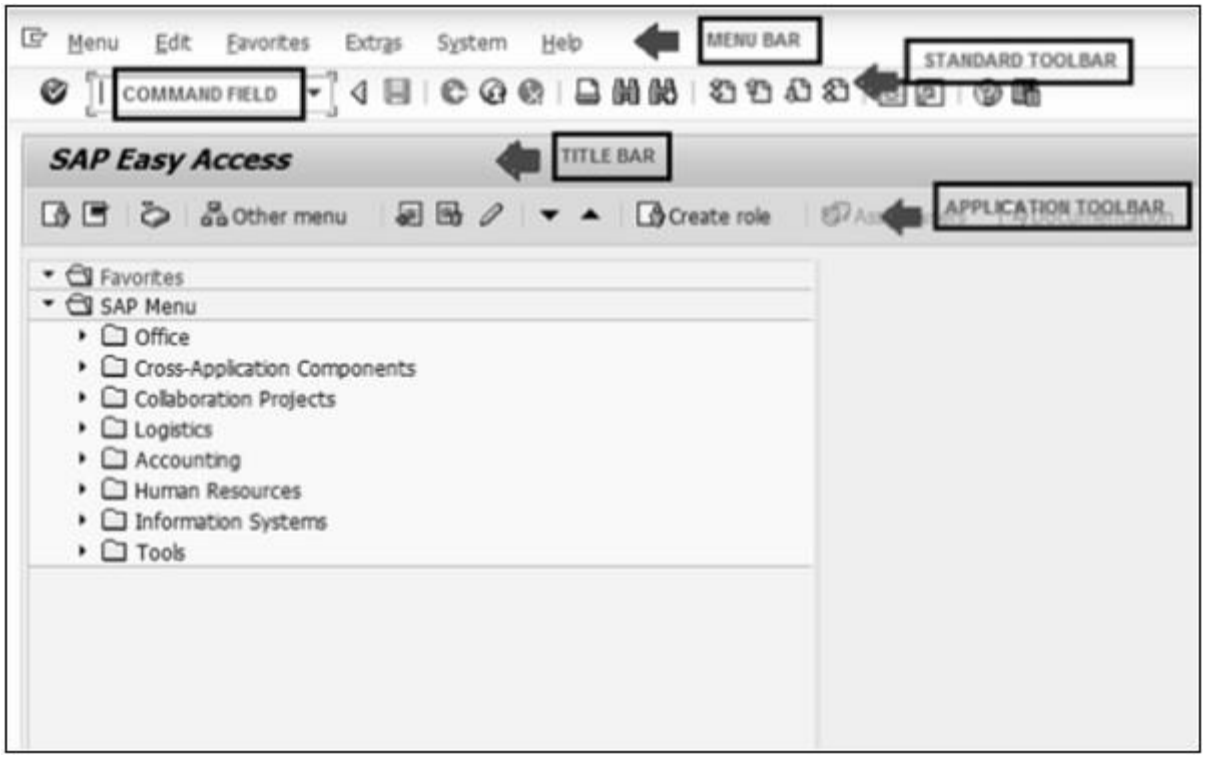
Giriş ekranı

SAP sunucusunda oturum açtıktan sonra, SAP oturum açma ekranı Kullanıcı Kimliği ve Parola isteyecektir. Geçerli bir kullanıcı kimliği ve Parola girmeniz ve Enter'a basmanız gerekir (kullanıcı kimliği ve parola sistem yöneticisi tarafından sağlanır). Giriş ekranı aşağıdadır.



Araç Çubuğu Simgesi

SAP ekranı araç çubuğu aşağıdadır.



Menü Çubuğu – Menü çubuğu, iletişim penceresinin en üst satırıdır.

Standart Araç Çubuğu - Sayfa Başı, Sayfa Sonu, Sayfa Yukarı, Sayfa Aşağı ve Kaydet gibi çoğu standart işlev bu araç çubuğunda bulunur.

Başlık Çubuğu – Başlık Çubuğu, içinde bulunduğunuz uygulamanın/iş sürecinin adını görüntüler.

Uygulama Araç Çubuğu − Uygulamaya özel menü seçenekleri burada mevcuttur.

Komut Alanı − Menü işlemleri arasında gezinmeden uygulama başlatabiliyoruz ve iş süreçlerine bazı mantıksal kodlar atanıyor. Uygulamayı doğrudan başlatmak için komut alanına işlem kodları girilir.

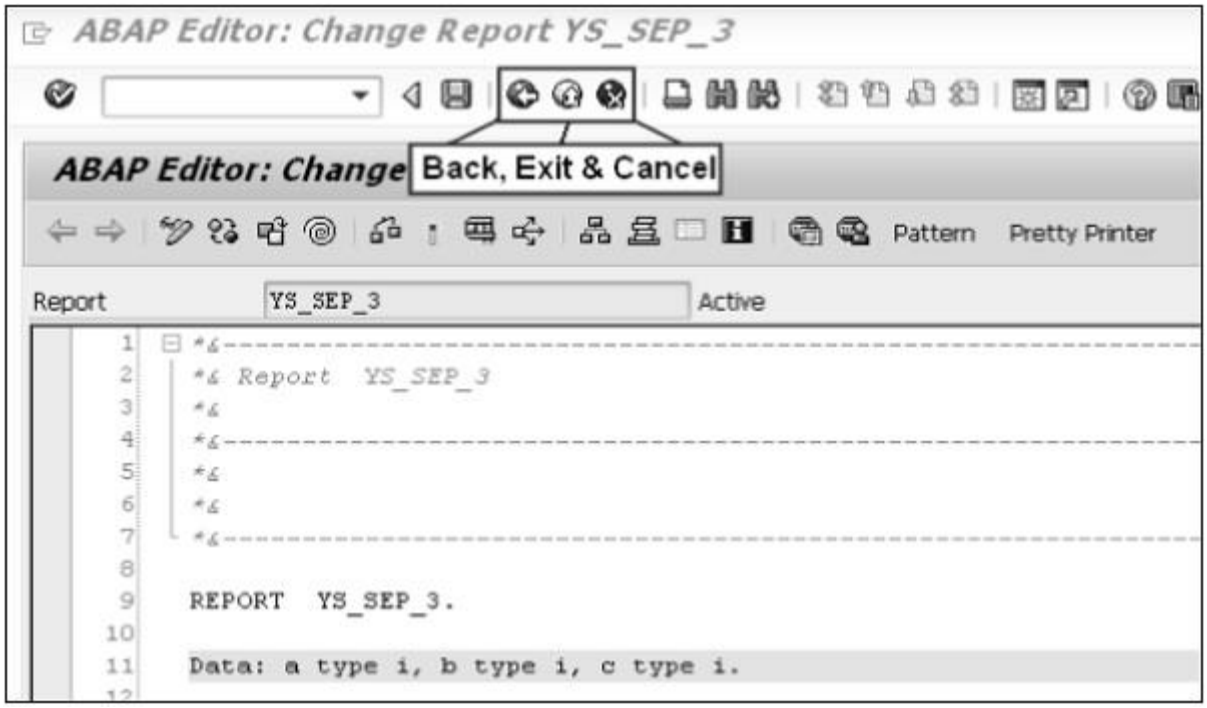
ABAP Editörü

ABAP Editörüne gitmek için SE38 işlemini (Komut Alanına SE38 girin) başlatabilirsiniz.

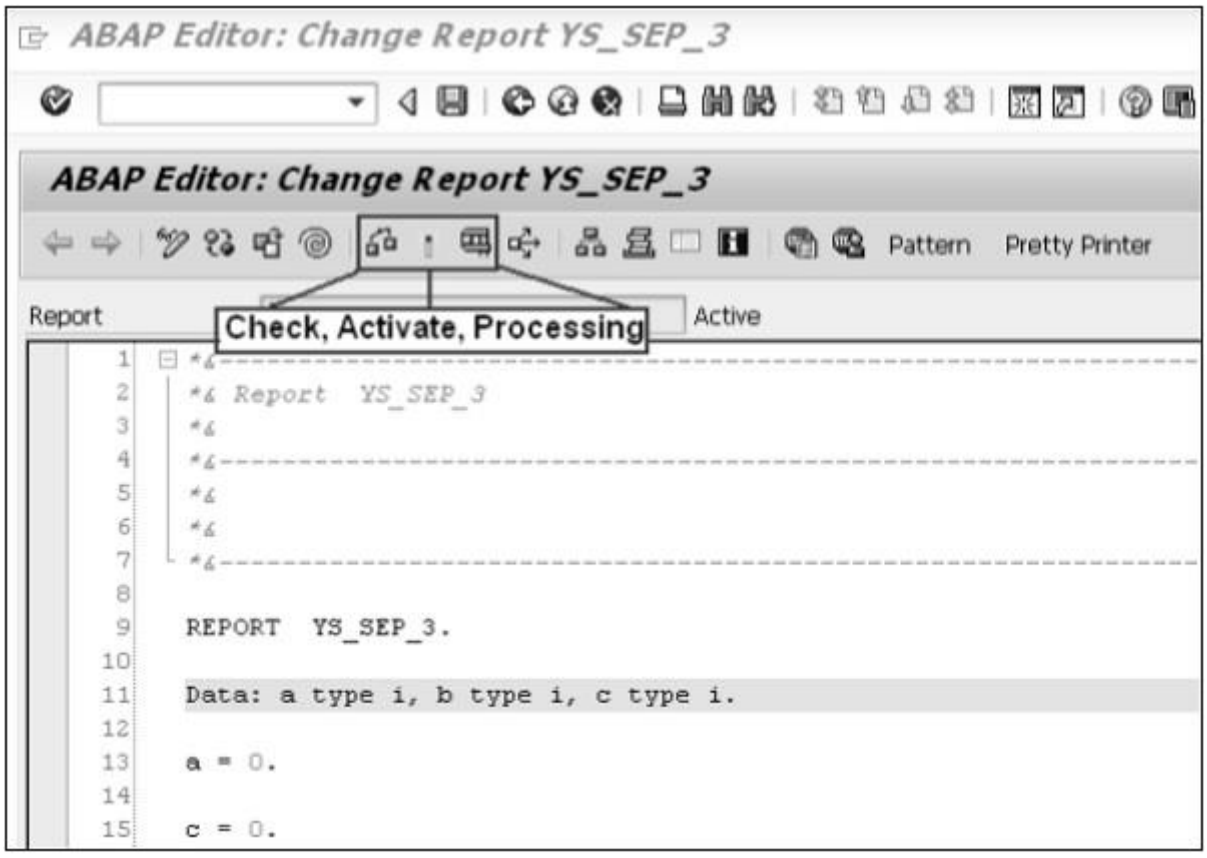
Standart Tuşlar ve Simgeler

Çıkış tuşları programdan/modülden çıkmak veya oturumu kapatmak için kullanılır. Ayrıca son erişilen ekrana geri dönmek için de kullanılırlar.

Aşağıdaki resimde gösterildiği gibi SAP'de kullanılan standart çıkış tuşlarıdır.



Raporların kontrol edilmesi, etkinleştirilmesi ve işlenmesi için seçenekler aşağıdadır.



Oturumu Kapat

ABAP Editörünüzden çıkmak ve/veya işinizi bitirdikten sonra SAP sisteminden çıkmak her zaman iyi bir uygulamadır.

Salih KÜÇÜK - SAP Retail Consultant

SAP ABAP - Temel Sözdizimi

İfadeler

ABAP kaynak programı, yorumlar ve ABAP ifadelerinden oluşur. ABAP'taki her ifade bir anahtar sözcükle başlar ve bir nokta ile biter ve ABAP büyük/küçük harfe duyarlı değildir.

Bir programdaki ilk yorumsuz satır REPORT kelimesiyle başlar. Rapor her zaman oluşturulan herhangi bir yürütülebilir programın ilk satırı olacaktır. İfadeyi daha önce oluşturulan program adı takip eder. Daha sonra hat bir nokta ile sonlandırılır.

Sözdizimi -

```
REPORT [Program_Name].

[Statements…].
```

Bu, ifadenin düzenleyicide ihtiyaç duyduğu kadar satır almasına izin verir. Örneğin, RAPOR şöyle görünebilir -

```
REPORT Z_Test123_01.
```

İfadeler, bir komut ve bir nokta ile biten herhangi bir değişken ve seçenekten oluşur. İfadenin sonunda nokta göründüğü sürece herhangi bir sorun ortaya çıkmaz. İfadenin bittiği yeri gösteren bu dönemdir.

Kodu yazalım.

REPORT ifadesinin altındaki satıra şu ifadeyi yazmanız yeterlidir: 'ABAP Eğitimi' yazın.

```
REPORT Z_Test123_01.

Write 'This is ABAP Tutorial'.
```

İfadeler yazarken dikkate alınması gereken dört şey -

- Write ifadesi, tırnak içindekileri çıktı penceresine yazar.
- ABAP düzenleyicisi, tek tırnak içine alınmış metin dizeleri dışında tüm metni büyük harfe dönüştürür.
- Bazı eski programlama dillerinden farklı olarak, ABAP bir ifadenin bir satırda nerede başladığıyla ilgilenmez. Bundan faydalanabilir ve kod bloklarını belirtmek için girinti kullanarak programınızın okunabilirliğini artırabilirsiniz.
- ABAP'ın ifadelerin düzeni üzerinde herhangi bir kısıtlaması yoktur. Yani, tek bir satıra birden çok ifade yerleştirilebilir veya tek bir ifade birden çok satıra yayılabilir.

iki nokta üst üste gösterimi

Her bir ifadenin başlangıcı aynıysa, ardışık ifadeler birbirine zincirlenebilir. Bu, tek tek ifadeleri sonlandırmak için kullanılan iki nokta üst üste (:) operatörü ve virgülle yapılır, tıpkı normal ifadeleri noktalayan noktalara benzer şekilde.

Aşağıda, bazı tuş vuruşlarını kaydedebilecek bir program örneği verilmiştir -

```
WRITE 'Hello'.
WRITE 'ABAP'.
WRITE 'World'.
```

İki nokta üst üste gösterimini kullanarak bu şekilde yeniden yazılabilir -

```
WRITE: 'Hello',
     'ABAP',
     'World'.
```

Diğer herhangi bir ABAP deyimi gibi, düzen önemli değil. Bu eşit derecede doğru bir ifadedir -

```
WRITE: 'Hello', 'ABAP', 'World'.
```

Yorumlar

Satır içi yorumlar, bir programın herhangi bir yerinde iki yöntemden biri ile bildirilebilir -

- Tam satır yorumları satırın ilk yerine yıldız işareti (*) konularak belirtilir, bu durumda satırın tamamı sistem tarafından yorum olarak değerlendirilir. Yorumların bir nokta ile sonlandırılmasına gerek yoktur çünkü birden fazla satıra yayılamazlar −

```
* This is the comment line
```

- Kısmi satır yorumları, bir ifadeden sonra çift tırnak (") girilerek belirtilir. Çift alıntıdan sonraki tüm metinler sistem tarafından yorum olarak kabul edilir. Kısmi satır açıklamalarını bir nokta ile sonlandırmanız gerekmez, çünkü bunlar daha fazla alana yayılmayabilir. birden fazla satır -

```
WRITE 'Hello'. "Here is the partial comment
```

Not − Yorumlanan kod, ABAP düzenleyicisi tarafından büyük harfle yazılmaz.

Boşlukları Bastırma

NO-ZERO komutu DATA deyimini takip eder. Boşluklar içeren bir sayı alanının baştaki tüm sıfırlarını bastırır. Çıktı genellikle kullanıcıların okuması için daha kolaydır.

Örnek

```
REPORT Z_Test123_01.

DATA: W_NUR(10) TYPE N.
    MOVE 50 TO W_NUR.
    WRITE W_NUR NO-ZERO.
```

Yukarıdaki kod aşağıdaki çıktıyı üretir -

```
50
```

Not − NO-ZERO komutu olmadan çıktı: 0000000050

Boş Çizgiler

SKIP komutu, sayfaya boş satırlar eklemeye yardımcı olur.

Örnek

Mesaj komutu aşağıdaki gibidir -

```
WRITE 'This is the 1st line'.
SKIP.
WRITE 'This is the 2nd line'.
```

Yukarıdaki mesaj komutu aşağıdaki çıktıyı üretir -

```
This is the 1st line
This is the 2nd line
```

Birden çok boş satır eklemek için SKIP komutunu kullanabiliriz.

```
SKIP number_of_lines.
```

Çıktı, satır sayısıyla tanımlanan birkaç boş satır olacaktır. SKIP komutu, imleci sayfada istenen bir satıra da konumlandırabilir.

```
SKIP TO LINE line_number.
```

Bu komut, imleci dinamik olarak sayfada yukarı ve aşağı hareket ettirmek için kullanılır. Genellikle, bu komuttan sonra çıktıyı istenen satıra koymak için bir WRITE ifadesi ortaya çıkar.

Çizgi Ekleme

ULINE komutu, çıktı boyunca otomatik olarak yatay bir çizgi ekler. Hattın konumunu ve uzunluğunu kontrol etmek de mümkündür. Sözdizimi oldukça basittir -

```
ULINE.
```

Örnek

Mesaj komutu aşağıdaki gibidir -

```
WRITE 'This is Underlined'.
ULINE.
```

Yukarıdaki kod aşağıdaki çıktıyı üretir -

This is Underlined (and a horizontal line below this).

Mesajlar

MESSAGE komutu, programın başında REPORT deyiminde belirtilen bir mesaj kimliği tarafından tanımlanan mesajları görüntüler. Mesaj kimliği, MESSAGE komutu kullanıldığında programın hangi 1.000 mesaj kümesine erişeceğini tanımlayan 2 karakterli bir koddur.

Mesajlar 000'den 999'a kadar numaralandırılmıştır. Her numara ile en fazla 80 karaktere kadar bir mesaj metni ilişkilendirilir. Mesaj numarası arandığında ilgili metin görüntülenir.

Mesaj komutuyla kullanılacak karakterler aşağıdadır -

İleti	Tip	Sonuçlar
E	Hata	Mesaj görünür ve uygulama mevcut noktasında durur. Program arka plan modunda çal edilir ve mesaj iş günlüğüne kaydedilir.
W	Uyarı	Mesaj görünür ve uygulamanın devam etmesi için kullanıcının Enter tuşuna basması ge plan modunda, mesaj iş günlüğüne kaydedilir.
ben	Bilgi	Mesaj metnini içeren bir açılır pencere açılır ve kullanıcının devam etmek için Enter tuş gerekir. Arka plan modunda, mesaj iş günlüğüne kaydedilir.
A	eğilmek	Bu ileti sınıfı, kullanıcının şu anda kullanmakta olduğu işlemi iptal eder.

S	Başarı	Bu, ekranın altında bir bilgi mesajı sağlar. Görüntülenen bilgiler doğası gereği olumlud kullanıcı geri bildirimi içindir. Mesaj, programı hiçbir şekilde engellemez.
X	iptal	Bu mesaj programı iptal eder ve bir ABAP kısa dökümü oluşturur.

Hata mesajları normalde kullanıcıların yapmamaları gereken şeyleri yapmalarını engellemek için kullanılır. Uyarı mesajları genellikle kullanıcılara eylemlerinin sonuçlarını hatırlatmak için kullanılır. Bilgi mesajları, kullanıcılara faydalı bilgiler verir.

Örnek

AB kimliği mesajı için bir mesaj oluşturduğumuzda, MESSAGE komutu - MESSAGE E011 aşağıdaki çıktıyı verir -

EAB011 This report does not support sub-number summarization.

SAP ABAP - Veri Türleri

ABAP'ta programlama yaparken, çeşitli bilgileri saklamak için çeşitli değişkenler kullanmamız gerekir. Değişkenler, değerleri depolamak için ayrılmış bellek konumlarından başka bir şey değildir. Bu, bir değişken oluşturduğunuzda bellekte biraz yer ayırdığınız anlamına gelir. Karakter, tamsayı, kayan nokta vb. gibi çeşitli veri türlerinin bilgilerini depolamak isteyebilirsiniz. Bir değişkenin veri türüne bağlı olarak, işletim sistemi bellek ayırır ve ayrılmış bellekte nelerin saklanabileceğine karar verir.

Temel Veri Türleri

ABAP, programcıya sabit uzunluklu ve değişken uzunluklu veri tiplerinden oluşan zengin bir çeşitlilik sunar. Aşağıdaki tablo ABAP temel veri türlerini listeler -

Tip	anahtar kelime
bayt alanı	X
Metin alanı	C
tamsayı	ben
Kayan nokta	F
Paketlenmiş numara	P
Metin dizesi	SİCİM

Bazı alanlar ve sayılar, aşağıdaki gibi bir veya daha fazla ad kullanılarak değiştirilebilir -

- bayt
- sayısal
- karakter benzeri

Aşağıdaki tablo, veri türünü, değeri bellekte saklamak için ne kadar bellek gerektiğini ve bu tür değişkenlerde saklanabilecek minimum ve maksimum değeri gösterir.

Tip	Tipik Uzunluk	Tipik Aralık
X	1 bayt	Herhangi bir bayt değeri (00 - FF)
C	1 karakter	1 ila 65535
N (sayısal metin dosyalandı)	1 karakter	1 ila 65535
D (karakter benzeri tarih)	8 karakter	8 karakter
T (karakter benzeri zaman)	6 karakter	6 karakter
ben	4 bayt	-2147483648 ila 2147483647
F	8 bayt	2.2250738585072014E-308 ila 1.7976931348623157E+308 pozitif v
P	8 bayt	[-10^(2len -1) +1] ila [+10^(2len -1) 1 sabit uzunluk)
SİCİM	Değişken	Herhangi bir alfanümerik karakter
XSTRING (bayt dizisi)	Değişken	Herhangi bir bayt değeri (00 - FF)

Örnek

```
REPORT YR_SEP_12.
DATA text_line TYPE C LENGTH 40.
text_line = 'A Chapter on Data Types'.
Write text_line.

DATA text_string TYPE STRING.
text_string = 'A Program in ABAP'.
```

```
Write / text_string.

DATA d_date TYPE D.
d_date = SY-DATUM.
Write / d_date.
```

Bu örnekte, önceden tanımlanmış uzunluğu 40 olan C tipi bir karakter dizgisine sahibiz. STRING, değişken uzunluktaki herhangi bir karakter dizgisi (metin dizgileri) için kullanılabilen bir veri türüdür. STRING türü veri nesneleri genellikle sabit uzunluğun önemli olmadığı karakter benzeri içerik için kullanılmalıdır.

Yukarıdaki kod aşağıdaki çıktıyı üretir -

```
A Chapter on Data Types
A Program in ABAP
12092015
```

DATE türü, tarih bilgilerinin saklanması için kullanılır ve yukarıda gösterildiği gibi sekiz hane saklayabilir.

Kompleks ve Referans Tipleri

Karmaşık türler, **Yapı türleri** ve **Tablo türleri** olarak sınıflandırılır . Yapı tiplerinde, elemanter tipler ve yapılar (yani bir yapıya gömülü yapı) birlikte gruplandırılmıştır. Yalnızca temel türlerin gruplandırılmasını düşünebilirsiniz. Ancak yapıların iç içe geçme olasılığının farkında olmalısınız.

Temel tipler birlikte gruplandırıldığında, veri öğesine gruplanmış bir veri öğesi olarak erişilebilir veya bireysel temel tip veri öğelerine (yapı alanları) erişilebilir. Tablo türleri, diğer programlama dillerinde daha çok diziler olarak bilinir. **Diziler** basit veya yapı dizileri olabilir. ABAP'ta dizilere dahili tablolar denir ve diğer programlama dilleriyle karşılaştırıldığında birçok şekilde bildirilebilir ve çalıştırılabilirler. Aşağıdaki tablo, dahili tabloların karakterize edildiği parametreleri gösterir.

S.No.	Parametre & Açıklama
1	**Satır veya satır türü** Bir iç tablonun satırı, temel, karmaşık veya referans türünde olabilir.
2	**Anahtar** Tablo satırlarını tanımlayan bir iç tablonun anahtarı olarak bir alanı veya bir alan grubunu b temel türlerin alanlarını içerir.
3	**Erişim yöntemi** ABAP programlarının bireysel tablo girişlerine nasıl eriştiğini açıklar.

Referans türleri, sınıfların, arayüzlerin ve çalışma zamanı veri öğelerinin örneklerine başvurmak için kullanılır. ABAP OOP çalışma zamanı türü hizmetleri (RTTS), çalışma zamanında veri öğelerinin bildirilmesini sağlar.

SAP ABAP - Değişkenler

Değişkenler, bir programın ayrılmış bellek alanı içinde değerleri depolamak için kullanılan veri nesneleri olarak adlandırılır. Adından da anlaşılacağı gibi, kullanıcılar ABAP deyimleri yardımıyla değişkenlerin içeriğini değiştirebilirler. ABAP'deki her değişkenin, değişkenin belleğinin boyutunu ve düzenini belirleyen belirli bir türü vardır; o bellekte saklanabilecek değer aralığı; ve değişkene uygulanabilecek işlemler kümesi.

Tüm değişkenleri kullanılmadan önce bildirmelisiniz. Değişken bildiriminin temel biçimi -

```
DATA <f> TYPE <type> VALUE <val>.
```

Burada <f> bir değişkenin adını belirtir. Değişkenin adı en fazla 30 karakter uzunluğunda olabilir. <type> değişkenin türünü belirtir. Tam olarak belirtilen teknik özelliklere sahip herhangi bir veri türü, <tür> olarak bilinir. <val>, <f> değişkeninin başlangıç değerini belirtir. Temel bir sabit uzunluklu değişken tanımlamanız durumunda, DATA deyimi, değişkenin değerini, türe özgü başlangıç değeriyle otomatik olarak doldurur. <val> için diğer olası değerler, bir hazır bilgi, sabit veya Is INITIAL gibi açık bir yan tümce olabilir.

Aşağıda, değişken bildirimlerinin geçerli örnekleri verilmiştir.

```
DATA d1(2) TYPE C.
DATA d2 LIKE d1.
DATA minimum_value TYPE I VALUE 10.
```

Yukarıdaki kod parçacığında, d1 C tipi bir değişkendir, d2 d1 tipi bir değişkendir ve minimum_değer, ABAP tamsayı tipi I bir değişkendir.

Bu bölüm, ABAP'ta bulunan çeşitli değişken türlerini açıklayacaktır. ABAP'ta üç tür değişken vardır -

- Statik Değişkenler
- Referans Değişkenler
- Sistem Değişkenleri

Statik Değişkenler

- Statik değişkenler, alt rutinlerde, fonksiyon modüllerinde ve statik yöntemlerde bildirilir.
- Ömür, beyanın bağlamıyla bağlantılıdır.
- 'CLASS-DATA' deyimi ile sınıflar içinde değişken tanımlayabilirsiniz.
- 'PARAMETERS' ifadesi, bir seçim ekranındaki giriş alanlarına bağlı olan temel veri nesnelerini bildirmek için kullanılabilir.
- 'SELECT-OPTIONS' ifadesini kullanarak bir seçim ekranındaki giriş alanlarına bağlı dahili tabloları da bildirebilirsiniz.

Bir değişkeni adlandırırken kullanılan kurallar aşağıdadır -

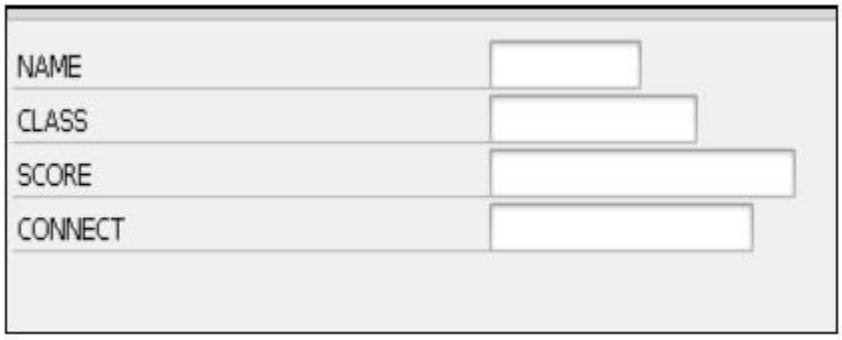
- Değişkenleri adlandırmak için "t" ve "," gibi özel karakterler kullanamazsınız.
- Önceden tanımlanmış veri nesnelerinin adı değiştirilemez.
- Değişkenin adı, herhangi bir ABAP anahtar sözcüğü veya yan tümcesi ile aynı olamaz.
- Değişkenlerin adı, daha fazla yoruma gerek kalmadan değişkenin anlamını iletmelidir.
- Tireler, yapıların bileşenlerini temsil etmek için ayrılmıştır. Bu nedenle, değişken adlarında kısa çizgilerden kaçınmanız gerekir.
- Bileşik sözcükleri ayırmak için alt çizgi karakteri kullanılabilir.

Bu program, PARAMETERS deyimini kullanarak bir değişkenin nasıl bildirileceğini gösterir -

```
REPORT ZTest123_01.
PARAMETERS: NAME(10) TYPE C,
CLASS TYPE I,
SCORE TYPE P DECIMALS 2,
CONNECT TYPE MARA-MATNR.
```

Burada NAME, 10 karakterlik bir parametreyi temsil eder, CLASS, varsayılan boyutu bayt cinsinden tamsayı türünde bir parametreyi belirtir, SCORE, iki ondalık basamağa kadar değerlere sahip bir paketlenmiş tür parametresini temsil eder ve CONNECT, ABAP Sözlüğünün MARA-MATNF tipine atıfta bulunur. .

Yukarıdaki kod aşağıdaki çıktıyı üretir -



Referans Değişkenler

Referans değişkenleri bildirmek için sözdizimi -

```
DATA <ref> TYPE REF TO <type> VALUE IS INITIAL.
```

- REF TO ekleme, bir referans değişkeni ref bildirir.
- REF TO'dan sonraki belirtim, referans değişkeninin statik türünü belirtir.
- Statik tür, <ref>'in başvurabileceği nesne kümesini kısıtlar.
- Dinamik başvuru değişkeni türü, şu anda başvurduğu veri türü veya sınıfıdır.
- Statik tip her zaman daha geneldir veya dinamik tiple aynıdır.
- TYPE eklemesi, bağlı bir referans türü oluşturmak için ve bir başlangıç değeri olarak kullanılır ve DEĞER eklemesinden sonra yalnızca IS INITIAL belirtilebilir.

Örnek

```
CLASS C1 DEFINITION.
PUBLIC SECTION.
DATA Bl TYPE I VALUE 1.
ENDCLASS. DATA: Oref TYPE REF TO C1 ,
Dref1 LIKE REF TO Oref,
Dref2 TYPE REF TO I .
CREATE OBJECT Oref.
GET REFERENCE OF Oref INTO Dref1.
CREATE DATA Dref2.
Dref2→* = Dref1→*→Bl.
```

- Yukarıdaki kod parçacığında, bir nesne referansı Oref ve iki veri referans değişkeni Dref1 ve Dref2 bildirildi.
- Her iki veri referans değişkeni de tam olarak yazılmıştır ve işlenen konumlarında referans kaldırma operatörü →* kullanılarak referansı kaldırılabilir.

Sistem Değişkenleri

- ABAP sistem değişkenlerine tüm ABAP programlarından erişilebilir.
- Bu alanlar aslında çalışma zamanı ortamı tarafından doldurulur.
- Bu alanlardaki değerler, herhangi bir zaman noktasında sistemin durumunu gösterir.
- Sistem değişkenlerinin tam listesini SAP'deki SYST tablosunda bulabilirsiniz.
- SYST yapısının bireysel alanlarına “SYST-” veya “SY-” kullanılarak erişilebilir.

Örnek

```
REPORT Z_Test123_01.

WRITE:/'SY-ABCDE', SY-ABCDE,
    /'SY-DATUM', SY-DATUM,
    /'SY-DBSYS', SY-DBSYS,
    /'SY-HOST ', SY-HOST,
    /'SY-LANGU', SY-LANGU,
    /'SY-MANDT', SY-MANDT,
    /'SY-OPSYS', SY-OPSYS,
    /'SY-SAPRL', SY-SAPRL,
    /'SY-SYSID', SY-SYSID,
    /'SY-TCODE', SY-TCODE,
    /'SY-UNAME', SY-UNAME,
    /'SY-UZEIT', SY-UZEIT.
```

Yukarıdaki kod aşağıdaki çıktıyı üretir -

```
SY-ABCDE ABCDEFGHIJKLMNOPQRSTUVWXYZ
SY-DATUM 12.09.2015
SY-DBSYS ORACLE
SY-HOST sapserver
SY-LANGU EN
SY-MANDT 800
SY-OPSYS Windows NT
SY-SAPRL 700
SY-SYSID DMO
```

SY-TCODE SE38
SY-UNAME SAPUSER
SY-UZEIT 14:25:48

SAP ABAP - Sabitler ve Değişmezler

Değişmez değerler, bir programın kaynak kodunda oluşturduğunuz adsız veri nesneleridir. Tamamen değerleriyle tanımlanırlar. Bir değişmezin değerini değiştiremezsiniz. Sabitler, bildirime dayalı ifadeler kullanılarak statik olarak oluşturulan adlandırılmış veri nesneleridir. Bir sabit, programın bellek alanında saklanan bir değer atanarak bildirilir. Bir sabite atanan değer, programın yürütülmesi sırasında değiştirilemez. Bu sabit değerler değişmez değerler olarak da kabul edilebilir. İki tür değişmez bilgi vardır - sayısal ve karakter.

Sayısal Değişmezler

Sayı değişmezleri, ön ekli bir işarete sahip olabilen basamak dizileridir. Sayı değişmezlerinde ondalık ayırıcılar ve mantis ve üslü gösterim yoktur.

Aşağıda sayısal değişmezlerin bazı örnekleri verilmiştir -

183.
-97.
+326.

Karakter Değişmezleri

Karakter değişmezleri, tek tırnak içine alınmış bir ABAP programının kaynak kodundaki alfasayısal karakter dizileridir. Tırnak işaretleri içine alınmış karakter değişmezleri, önceden tanımlanmış ABAP tip C'ye sahiptir ve metin alanı değişmezleri olarak tanımlanır. "Geri tırnak" içine alınmış değişmez değerler, STRING ABAP tipine sahiptir ve dize değişmezleri olarak tanımlanır. Alan uzunluğu karakter sayısı ile tanımlanır.

Not − Metin alanı değişmez değerlerinde, sondaki boşluklar yok sayılır, ancak dize değişmezlerinde bunlar dikkate alınır.

Aşağıda karakter değişmezlerinin bazı örnekleri verilmiştir.

Metin alanı değişmezleri

```
REPORT YR_SEP_12.
Write 'Tutorials Point'.
Write / 'ABAP Tutorial'.
```

Dize alanı değişmezleri

```
REPORT YR_SEP_12.
Write `Tutorials Point `.
Write / `ABAP Tutorial `.
```

Çıktı, yukarıdaki her iki durumda da aynıdır -

Tutorials Point
ABAP Tutorial

Not − Sabitin değerini değiştirmeye çalıştığımızda, bir sözdizimi veya çalışma zamanı hatası oluşabilir. Bir sınıfın veya arayüzün bildirim kısmında bildirdiğiniz sabitler, o sınıf veya arayüzün statik niteliklerine aittir.

SABİTLER Bildirimi

CONSTANTS deyimi yardımıyla adlandırılmış veri nesnelerini bildirebiliriz.

Sözdizimi aşağıdadır -

```
CONSTANTS <f> TYPE <type> VALUE <val>.
```

CONSTANTS ifadesi DATA ifadesine benzer.

<f> sabit için bir ad belirtir. TYPE <type>, mevcut <type> veri türüyle aynı teknik öznitelikleri devralan <f> adlı bir sabiti temsil eder. DEĞER <val>, bildirilen sabit adına <f> bir başlangıç değeri atar.

Not − CONSTANTS deyiminde VALUE yan tümcesini kullanmalıyız. 'DEĞER' yan tümcesi, bildirimi sırasında sabite bir başlangıç değeri atamak için kullanılır.

Temel, karmaşık ve referans sabitler olmak üzere 3 tür sabitimiz var. Aşağıdaki ifade, CONSTANTS ifadesini kullanarak sabitlerin nasıl tanımlanacağını gösterir -

```
REPORT YR_SEP_12.
CONSTANTS PQR TYPE P DECIMALS 4 VALUE '1.2356'.
Write: / 'The value of PQR is:', PQR.
```

çıktı -

```
The value of PQR is: 1.2356
```

Burada temel veri türünü ifade eder ve temel sabit olarak bilinir.

Aşağıda karmaşık sabitler için bir örnek verilmiştir -

```
BEGIN OF EMPLOYEE,
Name(25) TYPE C VALUE 'Management Team',
Organization(40) TYPE C VALUE 'Tutorials Point Ltd',
Place(10) TYPE C VALUE 'India',
END OF EMPLOYEE.
```

Yukarıdaki kod parçacığında EMPLOYEE, Ad, Organizasyon ve Yer alanlarından oluşan karmaşık bir sabittir.

Aşağıdaki ifade sabit bir başvuru bildirir -

```
CONSTANTS null_pointer TYPE REF TO object VALUE IS INITIAL.
```

Sabit referansı karşılaştırmalarda kullanabiliriz veya prosedürlere aktarabiliriz.

SAP ABAP - Operatörler

ABAP, değişkenleri manipüle etmek için zengin bir operatör seti sağlar. Tüm ABAP operatörleri dört kategoriye ayrılmıştır -

- Aritmetik operatörler

- Karşılaştırma Operatörleri
- Bitsel Operatörler
- Karakter Dizisi Operatörleri

Aritmetik operatörler

Aritmetik operatörler, matematiksel ifadelerde cebirde kullanıldığı gibi kullanılır. Aşağıdaki liste aritmetik operatörleri açıklar. A tamsayı değişkeninin 20'yi ve B değişkeninin 40'ı tuttuğunu varsayın.

S.No.	Aritmetik Operatör & Açıklama
1	**+ (İlave)** Operatörün her iki tarafına da değerler ekler. Örnek: A + B 60 verecek.
2	**- (Çıkarma)** Sağ işleneni sol işlenenden çıkarır. Örnek: A − B -20 verecek.
3	*** (Çarpma işlemi)** Operatörün her iki tarafındaki değerleri çarpar. Örnek: A * B 800 verecek.
4	**/ (Bölüm)** Sol el işlenenini sağ işlenene böler. Örnek: B/A 2 verecek.
5	**MOD (Modül)** Sol işleneni sağ işlenene böler ve kalanı verir. Örnek: B MOD A 0 verecektir.

Örnek

```
REPORT YS_SEP_08.
DATA: A TYPE I VALUE 150,
B TYPE I VALUE 50,
Result TYPE I.
Result =  A / B.
WRITE / Result.
```

Yukarıdaki kod aşağıdaki çıktıyı üretir -

3

Karşılaştırma Operatörleri

Farklı işlenenler için çeşitli karşılaştırma operatörleri türlerini tartışalım.

S.No.	Karşılaştırma Operatörü ve Açıklama
1	**= (eşitlik testi). Alternatif form EQ'dur.** İki işlenenin değerlerinin eşit olup olmadığını kontrol eder, evet ise koşul doğru olur. Örnek değil.
2	**<> (Eşitsizlik testi). Alternatif form NE'dir.** İki işlenenin değerlerinin eşit olup olmadığını kontrol eder. Değerler eşit değilse koşul ger (A <> B) doğrudur.
3	**> (Testten büyük). Alternatif form GT'dir.** Sol işlenenin değerinin sağ işlenenin değerinden büyük olup olmadığını kontrol eder. E zaman koşul gerçekleşir. Örnek (A > B) doğru değil.
4	**< (Testten daha az). Alternatif form LT'dir.** Sol işlenenin değerinin sağ işlenenin değerinden küçük olup olmadığını kontrol eder. Eve koşul gerçek olur. Örnek (A < B) doğrudur.
5	**>= (Büyüktür veya eşittir) Alternatif biçim GE'dir.** Sol işlenenin değerinin sağ işlenenin değerinden büyük veya eşit olup olmadığını kontrol e zaman koşul gerçek olur. Örnek (A >= B) doğru değil.
6	**<= (Küçüktür veya eşittir testi). Alternatif form LE'dir.** Sol işlenenin değerinin sağ işlenenin değerinden küçük veya ona eşit olup olmadığını ko ise, o zaman koşul gerçek olur. Örnek (A <= B) doğrudur.
7	**a1 a2 VE a3 ARASINDA (Aralık testi)** a1'in a2 ve a3 (dahil) arasında olup olmadığını kontrol eder. Evet ise, o zaman koşul gerçek ARASINDA B VE C) doğrudur.
8	**İLK** Değişkenin içeriği değişmediyse ve otomatik olarak başlangıç değeri atanırsa koşul doğru IS INITIAL) doğru değil
9	**İLK DEĞİL** Değişkenin içeriği değiştiyse koşul doğru olur. Örnek (A IS NOT INITIAL) doğrudur.

Not − Değişkenlerin veri türü veya uzunluğu eşleşmiyorsa, otomatik dönüştürme gerçekleştirilir. Farklı veri türlerinin iki değeri karşılaştırılırken, değerlerden biri veya her ikisi için otomatik tip ayarlaması yapılır. Dönüştürme türüne, veri türü ve veri türünün tercih sırasına göre karar verilir.

Tercih sırası aşağıdadır -

- Bir alan I tipindeyse, diğeri I tipine dönüştürülür.
- Bir alan P türündeyse, diğeri P türüne dönüştürülür.
- Bir alan D tipi ise, diğeri D tipine dönüştürülür. Ancak C ve N tipleri dönüştürülmez ve doğrudan karşılaştırılır. T tipinde de durum benzerdir.
- Bir alan N tipinde ve diğeri C veya X tipindeyse, her iki alan da P tipine dönüştürülür.
- Bir alan C tipinde ve diğeri X tipindeyse, X tipi C tipine dönüştürülür.

örnek 1

```
REPORT YS_SEP_08.

DATA: A TYPE I VALUE 115,
    B TYPE I VALUE 119.
    IF A LT B.
    WRITE: / 'A is less than B'.
    ENDIF
```

Yukarıdaki kod aşağıdaki çıktıyı üretir -

```
A is less than B
```

Örnek 2

```
REPORT YS_SEP_08.

DATA: A TYPE I.
    IF A IS INITIAL.
    WRITE: / 'A is assigned'.
    ENDIF.
```

Yukarıdaki kod aşağıdaki çıktıyı üretir -

```
A is assigned.
```

Bitsel Operatörler

ABAP ayrıca Boole cebirsel ifadeleri oluşturmak için kullanılabilecek bir dizi bit düzeyinde mantıksal operatör sağlar. Bitsel operatörler, parantez vb. kullanılarak karmaşık ifadelerde birleştirilebilir.

S.No.	Bitsel Operatör ve Açıklama
1	**BIT-DEĞİL** Onaltılık bir sayıdaki tüm bitleri zıt değere çeviren tekli operatör. Örneğin, bu operatö (örneğin 'AA') bit düzeyi değerine sahip onaltılık bir sayıya uygulanması 01010101'i verir.

2	**BIT-VE** Bu ikili operatör, Boolean AND operatörünü kullanarak her alanı bit bit karşılaştırır.
3	**BIT-XOR** Boolean XOR (exclusive OR) operatörünü kullanarak her alanı bit bit karşılaştıran ikili opera
4	**BIT-VEYA** Boolean OR operatörünü kullanarak her alanı bit bit karşılaştıran ikili operatör.

Örneğin, Boolean AND, OR veya XOR operatörlerini A alanı ve B alanında bulunan iki bit değerine karşı uygularken oluşturulan değerleri gösteren doğruluk tablosu aşağıdadır.

Alan A	B Alanı	VE	VEYA	
0	0	0	0	
0	1	0	1	
1	0	0	1	
1	1	1	1	

Karakter Dizisi Operatörleri

Aşağıda karakter dizisi operatörlerinin bir listesi bulunmaktadır -

S.No.	Karakter Dizisi Operatörü ve Açıklama
1	**CO (Yalnızca İçerir)** A'nın yalnızca B'deki karakterlerden oluşup oluşmadığını kontrol eder.
2	**CN (YALNIZCA İçermez)** A'nın B'de olmayan karakterler içerip içermediğini kontrol eder.
3	**CA (HERHANGİ BİRİ İçerir)**

	A'nın en az bir B karakteri içerip içermediğini kontrol eder.
4	**NA (Hiçbir Şey İçermez)** A'nın B'nin herhangi bir karakterini içerip içermediğini kontrol eder.
5	**CS (Bir Dize İçerir)** A'nın B karakter dizisini içerip içermediğini kontrol eder.
6	**NS (Dize İÇERMEZ)** A'nın B karakter dizisini içerip içermediğini kontrol eder.
7	**CP (Bir Model İçerir)** A'nın B'deki kalıbı içerip içermediğini kontrol eder.
8	**NP (Bir Model İçermez)** A'nın B'deki kalıbı içerip içermediğini kontrol eder.

Örnek

```
REPORT YS_SEP_08.
DATA: P(10) TYPE C VALUE 'APPLE',
    Q(10) TYPE C VALUE 'CHAIR'.
    IF P CA Q.

    WRITE: / 'P contains at least one character of Q'.
    ENDIF.
```

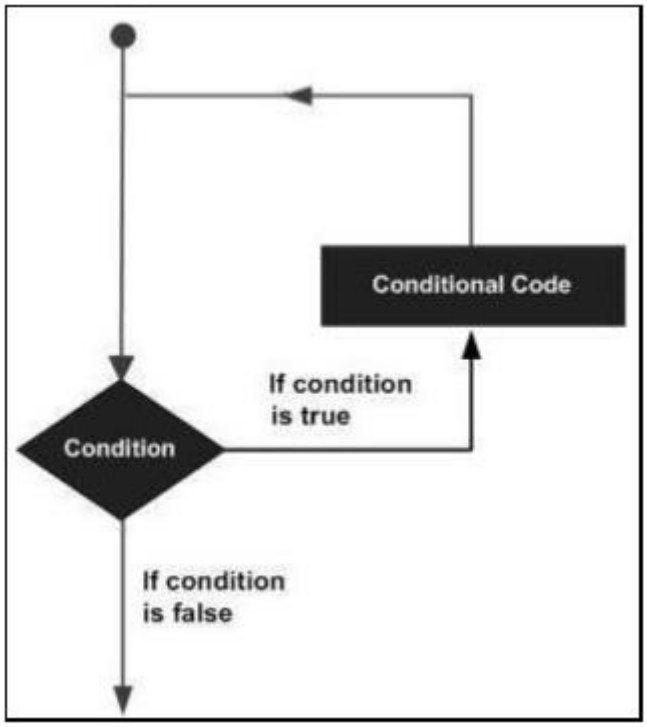
Yukarıdaki kod aşağıdaki çıktıyı üretir -

```
P contains at least one character of Q.
```

SAP ABAP - Döngü Kontrolü

Bir kod bloğunu birkaç kez çalıştırmanız gerektiğinde bir durum olabilir. Genel olarak, ifadeler sırayla yürütülür: Bir işlevdeki ilk ifade önce yürütülür, ardından ikincisi vb.

Programlama dilleri, daha karmaşık yürütme yollarına izin veren çeşitli kontrol yapıları sağlar. **Döngü deyimi** , bir deyimi veya deyimler grubunu birden çok kez yürütmemize izin verir ve aşağıdaki çoğu programlama dilinde döngü ifadesinin genel biçimidir.



ABAP programlama dili, döngü gereksinimlerini karşılamak için aşağıdaki döngü türlerini sağlar.

S.No.	Döngü Tipi ve Tanımı
1	Döngü sırasında Belirli bir koşul doğru olduğunda bir ifadeyi veya ifade grubunu tekrarlar. Döngü gövdesin önce koşulu test eder.
2	döngü yap DO ifadesi, belirli bir görevi belirli sayıda tekrarlamak için kullanışlıdır.
3	iç içe döngü Başka bir WHILE veya DO döngüsü içinde bir veya daha fazla döngü kullanabilirsiniz.

Döngü Kontrol İfadeleri

Döngü kontrol ifadeleri, yürütmeyi normal dizisinden değiştirir. ABAP, döngülerin zamanından önce sonlandırılmasına izin veren kontrol ifadeleri içerir. Aşağıdaki kontrol ifadelerini destekler.

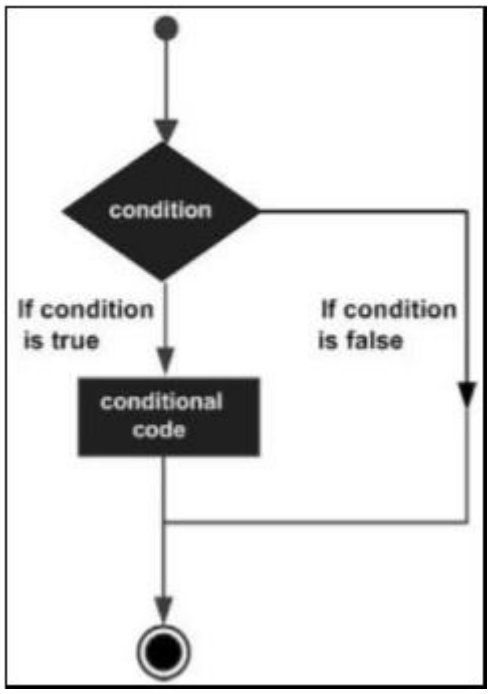
S.No.	Kontrol Bildirimi ve Açıklama

1	DEVAM ET Döngünün gövdesinin geri kalanını atlamasına neden olur ve sonraki döngü geçişini başlatı
2	KONTROL Eğer koşul yanlışsa, CHECK'den sonra kalan ifadeler yok sayılır ve sistem bir sonraki başlatır.
3	ÇIKIŞ Döngüyü tamamen sonlandırır ve yürütmeyi döngüden hemen sonraki ifadeye aktarır.

SAP ABAP - Kararlar

Karar verme yapıları, koşulun doğru olduğu belirlenirse yürütülecek bir ifade veya ifadeler ile birlikte program tarafından değerlendirilecek veya test edilecek bir veya daha fazla koşula ve isteğe bağlı olarak, koşulun gerçekleşmesi durumunda yürütülecek diğer ifadelere sahiptir. yalan olduğu belirlenir.

Programlama dillerinin çoğunda bulunan tipik bir karar verme yapısının genel şekli aşağıdadır -



ABAP programlama dili, aşağıdaki türde karar verme ifadeleri sağlar.

S.No.	Açıklama ve Açıklama
1	EĞER deyimi

	EĞER ifadesi, mantıksal bir ifadenin ardından bir veya daha fazla ifadeden oluşur.
2	EĞER.. Else İfadesi Bir IF ifadesini, ifade yanlış olduğunda yürütülen isteğe bağlı bir ELSE ifadesi izleyebilir.
3	İç İçe EĞER İfadesi Bir IF veya ELSEIF ifadesini başka bir IF veya ELSEIF ifadesinin içinde kullanabilirsiniz.
4	VAKA Kontrol Bildirimi CASE deyimi, iki veya daha fazla alanı veya değişkeni karşılaştırmamız gerektiğinde kullar

SAP ABAP - Dizeler

ABAP programlamasında yaygın olarak kullanılan karakter dizileri bir dizi karakterdir.

Alfanümerik karakterleri tutmak için minimum 1 karakter ve maksimum 65.535 karakterden oluşan veri tipi C değişkenleri kullanıyoruz. Varsayılan olarak, bunlar sola hizalanır.

Dize Oluşturma

Aşağıdaki bildirim ve başlatma, 'Merhaba' kelimesinden oluşan bir dize oluşturur. Dizenin boyutu tam olarak 'Merhaba' sözcüğündeki karakter sayısıdır.

```
Data my_Char(5) VALUE 'Hello'.
```

Aşağıdaki program, dize oluşturmaya bir örnektir.

```
REPORT YT_SEP_15.
DATA my_Char(5) VALUE 'Hello'.
Write my_Char.
```

Yukarıdaki kod aşağıdaki çıktıyı üretir -

```
Hello
```

IP uzunluğu

Karakter dizilerinin uzunluğunu bulmak için **STRLEN deyimini** kullanabiliriz . STRLEN () işlevi, dizede bulunan karakter sayısını döndürür.

Örnek

```
REPORT YT_SEP_15.
DATA: title_1(10) VALUE 'Tutorials',
    length_1 TYPE I.

length_1 = STRLEN( title_1 ).
```

```
Write: / 'The Length of the Title is:', length_1.
```

Yukarıdaki kod aşağıdaki çıktıyı üretir -

The Length of the Title is: 9

ABAP, dizeleri işleyen çok çeşitli ifadeleri destekler.

S.No.	Bildirim ve Amaç
1	**BİRLEŞTİR** Üçüncü bir dize oluşturmak için iki dize birleştirilir.
2	**YOĞUŞMA** Bu ifade boşluk karakterlerini siler.
3	**STRLEN** Bir alanın uzunluğunu bulmak için kullanılır.
4	**YER DEĞİŞTİRMEK** Karakterlerde değişiklik yapmak için kullanılır.
5	**ARAMA** Karakter dizilerinde arama yapmak için.
6	**VARDİYA** Bir dizenin içeriğini sola veya sağa taşımak için kullanılır.
7	**BÖLMEK** Bir alanın içeriğini iki veya daha fazla alana bölmek için kullanılır.

Aşağıdaki örnek, yukarıda belirtilen ifadelerden bazılarını kullanmaktadır -

Örnek

```
REPORT YT_SEP_15.
DATA: title_1(10) VALUE 'Tutorials',
    title_2(10) VALUE 'Point',
    spaced_title(30) VALUE 'Tutorials  Point  Limited',
    sep,
    dest1(30),
    dest2(30).
```

```
CONCATENATE title_1 title_2 INTO dest1.
Write: / 'Concatenation:', dest1.

CONCATENATE title_1 title_2 INTO dest2 SEPARATED BY sep.
Write: / 'Concatenation with Space:', dest2.

CONDENSE spaced_title.
Write: / 'Condense with Gaps:', spaced_title.

CONDENSE spaced_title NO-GAPS.
Write: / 'Condense with No Gaps:', spaced_title.
```

Yukarıdaki kod aşağıdaki çıktıyı üretir -

```
Concatenation: TutorialsPoint
Concatenation with Space: Tutorials Point
Condense with Gaps: Tutorials Point Limited
Condense with No Gaps: TutorialsPointLimited
```

Not -

- Birleştirme durumunda, 'sep' alanlar arasına bir boşluk ekler.
- CONDENSE ifadesi, alanlar arasındaki boşlukları kaldırır, ancak yalnızca 1 karakter boşluk bırakır.
- 'NO-GAPS', tüm boşlukları kaldıran CONDENSE ifadesine isteğe bağlı bir eklemedir.

SAP ABAP - Tarih ve Saat

ABAP, dünyanın çoğu yerinde geçerli olan Gregoryen takvimine dolaylı olarak atıfta bulunur. Çıktıyı ülkeye özel takvimlere dönüştürebiliriz. Tarih, bir takvime göre kesin bir gün, hafta veya ay olarak belirtilen bir zamandır. Zaman, bir güne göre kesin bir saniye veya dakika olarak belirtilir. ABAP her zaman 24 saat formatında zaman kazandırır. Çıktı, ülkeye özgü bir formata sahip olabilir. Tarihler ve saat genellikle geçerli saat diliminde geçerli olan yerel tarihler olarak yorumlanır.

ABAP, tarih ve saatle çalışmak için iki yerleşik tür sağlar -

- D veri türü
- T veri türü

Aşağıdaki temel formattır -

```
DATA: date TYPE D,
    time TYPE T.

DATA: year TYPE I,
month TYPE I,
day TYPE I,
hour TYPE I,
minute TYPE I,
second TYPE I.
```

Bu türlerin her ikisi de sırasıyla YYYYMMDD ve HHMMSS biçiminde olan sabit uzunluklu karakter türleridir.

zaman damgaları

Bu yerleşik türlere ek olarak, diğer iki tür **TIMESTAMP** ve **TIMESTAMPL** , UTC biçiminde bir zaman damgasını depolamak için birçok standart uygulama tablosunda kullanılmaktadır. Aşağıdaki tablo, ABAP'ta kullanılabilen temel tarih ve saat türlerini göstermektedir.

S.No.	Veri Türü ve Açıklama
1	**D** YYYYMMDD biçiminde yerleşik bir sabit uzunluklu tarih türü. Örneğin, 20100913 değeri tarihini temsil eder.
2	**T** HHMMSS formunun yerleşik bir sabit uzunluklu zaman türü. Örneğin, 102305 değeri saat 1 eder.
3	**TIMESTAMP** (Tip P – Uzunluk 8 Ondalıksız) Bu tür, kısa zaman damgalarını YYYYMMDDhhmmss biçiminde göstermek için kull 20100913102305 değeri, 13 Eylül 2010, 10:23:05 AM tarihini temsil eder.
4	**TIMESTAMPL** (Tip P - Uzunluk 11 Ondalık Sayı 7) TIMESTAMPL, YYYYMMDDhhmmss,mmmuuun biçiminde uzun zaman damgalarını tems 'mmmuuun' ek rakamları bir saniyenin kesirlerini temsil eder.

Geçerli Tarih ve Saat

Aşağıdaki kod parçacıkları, geçerli sistem tarihini ve saatini alır.

```
REPORT YR_SEP_15.
DATA: date_1 TYPE D.

date_1 = SY-DATUM.
Write: / 'Present Date is:', date_1 DD/MM/YYYY.

date_1 = date_1 + 06.
Write: / 'Date after 6 Days is:', date_1 DD/MM/YYYY.
```

Yukarıdaki kod aşağıdaki çıktıyı üretir -

```
Present Date is: 21.09.2015
```

Date after 6 Days is: 27.09.2015

date_1 değişkenine, geçerli sistem tarihi SY-DATUM'un değeri atanır. Ardından, tarih değerini 6 artırıyoruz. ABAP'ta bir tarih hesaplaması açısından bu, tarih nesnesinin gün bileşenini 6 gün artırdığımız anlamına geliyor. ABAP çalışma zamanı ortamı, bir ayın sonuna ulaştığında tarih değerini devirecek kadar akıllıdır.

Zaman hesaplamaları, tarih hesaplamalarına benzer şekilde çalışır. Aşağıdaki kod, temel zaman aritmetiğini kullanarak geçerli sistem zamanını 75 saniye artırır.

```
REPORT YR_SEP_15.
DATA: time_1 TYPE T.
    time_1 = SY-UZEIT.

Write /(60) time_1 USING EDIT MASK
'Now the Time is: __:__:__'.
time_1 = time_1 + 75.

Write /(60) time_1 USING EDIT MASK
'A Minute and a Quarter from Now, it is: __:__:__'.
```

Yukarıdaki kod aşağıdaki çıktıyı üretir -

Now the Time is 11:45:05
A Minute and a Quarter from Now, it is: 11:46:20

Zaman Damgalarıyla Çalışmak

Geçerli sistem saatini alabilir ve aşağıdaki kodda gösterildiği gibi **GET TIME STAMP kullanarak bir zaman damgası değişkeninde saklayabilirsiniz.** GET TIME STAMP deyimi, kullanılan zaman damgası veri nesnesinin türüne göre zaman damgasını uzun el veya kısa el biçiminde saklar. Zaman damgası değeri, UTC standardı kullanılarak kodlanmıştır.

```
REPORT YR_SEP_12.
DATA: stamp_1 TYPE TIMESTAMP,

stamp_2 TYPE TIMESTAMPL.
GET TIME STAMP FIELD stamp_1.
Write: / 'The short time stamp is:', stamp_1

TIME ZONE SY-ZONLO.
GET TIME STAMP FIELD stamp_2.
Write: / 'The long time stamp is:', stamp_2
TIME ZONE SY-ZONLO.
```

Yukarıdaki kod aşağıdaki çıktıyı üretir -

The short time stamp is: 18.09.2015 11:19:40
The long time stamp is: 18.09.2015 11:19:40,9370000

Yukarıdaki örnekte, WRITE ifadesinin TIME ZONE eklemesini kullanarak zaman damgasını görüntülüyoruz. Bu ekleme, zaman damgasının çıktısını belirtilen saat dilimi kurallarına göre biçimlendirir. SY-ZONLO sistem alanı, kullanıcının tercihlerinde yapılandırılan yerel saat dilimini görüntülemek için kullanılır.

SAP ABAP - Verileri Biçimlendirme

ABAP, programların çıktısını biçimlendirmek için çeşitli biçimlendirme seçenekleri sunar. Örneğin, farklı renklerde veya biçimlendirme stillerinde çeşitli öğeleri içeren bir liste oluşturabilirsiniz.

WRITE ifadesi, verileri bir ekranda görüntülemek için kullanılan bir biçimlendirme ifadesidir. WRITE deyimi için farklı biçimlendirme seçenekleri vardır. WRITE ifadesinin sözdizimi şudur:

WRITE <format> <f> <options>.

Bu sözdiziminde, <format>, yeni bir satırdan başlayarak çıktının görüntüsünü gösteren bir eğik çizgi (/) olabilen çıktı biçimi belirtimini temsil eder. Eğik çizgiye ek olarak, biçim belirtimi bir sütun numarası ve sütun uzunluğu içerir. Örneğin, WRITE/04 (6) ifadesi, yeni bir satırın 4. sütunla başladığını ve sütun uzunluğunun 6 olduğunu gösterirken, WRITE 20 ifadesi, 20 sütunlu geçerli satırı gösterir. <f> parametresi bir veri değişkenini veya numaralı metin.

Aşağıdaki tablo, biçimlendirme için kullanılan çeşitli maddeleri açıklamaktadır -

S.No.	Madde ve Açıklama
1	**SOL TARAFLI** Çıktının sola dayalı olduğunu belirtir.
2	**MERKEZLİ** Çıktının ortalandığını belirtir.
3	**DOĞRULANDI** Çıktının sağa dayalı olduğunu belirtir.
4	**<g> ALTINDA** <g> alanının altında başlar.
5	**BOŞLUK YOK** <f> alanından sonraki boşluğun reddedildiğini belirtir.
6	**DÜZENLEME MASKESİ KULLANMA <m>** <m> biçim şablonunun belirtimini belirtir. No EDIT Mask Kullanarak: Bu, ABAP Sözlüğ format şablonunun devre dışı bırakıldığını belirtir.

7	**sıfır** Bir alan yalnızca sıfır içeriyorsa, bunlar boşluklarla değiştirilir.

Sayısal Tür alanları için biçimlendirme seçenekleri aşağıdadır -

S.No.	Madde ve Açıklama
1	**SİNYAL YOK** Ekranda hiçbir öncü işaretin görüntülenmediğini belirtir.
2	**ÜS <e>** F tipinde (kayan nokta alanları) üssün <e> içinde tanımlandığını belirtir.
3	**YUVARLAK <r>** P tipi alanlar (paketlenmiş sayısal veri türleri) önce 10**(-r) ile çarpılır ve ardından bir tar yuvarlanır.
4	**PARA BİRİMİ <c>** Biçimlendirmenin TCURX veritabanı tablosunda saklanan para birimi <c> değerine göre yap
5	**BİRİM <u>** P tipi için T006 veritabanı tablosunda belirtildiği gibi, ondalık basamak sayısının <u> sabitlendiğini belirtir.
6	**ONDALIKLAR <d>** <d> basamak sayısının ondalık noktadan sonra görüntülenmesi gerektiğini belirtir.

Örneğin, aşağıdaki tabloda tarih alanları için farklı biçimlendirme seçenekleri gösterilmektedir -

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Burada DD iki haneli tarihi, AA iki haneli ayı, YY iki haneli yılı ve YYYY dört haneli yılı temsil etmektedir.

Yukarıdaki biçimlendirme seçeneklerinden bazılarını uygulayan bir ABAP kodu örneğine bakalım -

```
REPORT ZTest123_01.

DATA: n(9) TYPE C VALUE 'Tutorials',
m(5) TYPE C VALUE 'Point'.

WRITE: n, m.
WRITE: / n,
/ m UNDER n.

WRITE: / n NO-GAP, m.
DATA time TYPE T VALUE '112538'.

WRITE: / time,
/(8) time Using EDIT MASK '__:__:__'.
```

Yukarıdaki kod aşağıdaki çıktıyı üretir -

```
Tutorials Point
Tutorials
Point
TutorialsPoint
112538
11:25:38
```

SAP ABAP - İstisna İşleme

İstisna , bir programın yürütülmesi sırasında ortaya çıkan bir sorundur **.** Bir istisna oluştuğunda, programın normal akışı bozulur ve program uygulaması anormal şekilde sonlandırılır, bu tavsiye edilmez, bu nedenle bu istisnalar ele alınmalıdır.

İstisnalar, kontrolü bir programın bir bölümünden diğerine aktarmanın bir yolunu sağlar. ABAP istisna işleme, üç anahtar kelime üzerine kuruludur – RAISE, TRY, CATCH ve CLEANUP. Bir bloğun bir istisna oluşturacağını varsayarsak, bir yöntem, TRY ve CATCH anahtar kelimelerinin bir kombinasyonunu kullanarak bir istisna yakalar. Bir özel durum oluşturabilecek kodun etrafına bir TRY - CATCH bloğu yerleştirilir. TRY – CATCH – kullanımı için sözdizimi aşağıdadır.

```
TRY.
Try Block <Code that raises an exception>

CATCH
Catch Block <exception handler M>
. . .
. . .
. . .
CATCH
Catch Block <exception handler R>

CLEANUP.
Cleanup block <to restore consistent state>

ENDTRY.
```

RAISE - Bazı istisnai durumların meydana geldiğini belirtmek için istisnalar ortaya çıkar. Genellikle bir istisna işleyicisi hatayı onarmaya veya alternatif bir çözüm bulmaya çalışır.

TRY − TRY bloğu, istisnaları işlenecek olan uygulama kodlamasını içerir. Bu ifade bloğu sırayla işlenir. Daha fazla kontrol yapılarını ve prosedür çağrılarını veya diğer ABAP programlarını içerebilir. Bunu bir veya daha fazla yakalama bloğu takip eder.

CATCH - Bir program, bir programda sorunu çözmek istediğiniz yerde bir istisna işleyicisi ile bir istisna yakalar. CATCH anahtar sözcüğü, bir istisnanın yakalanmasını belirtir.

CLEANUP − CLEANUP bloğunun deyimleri, aynı TRY - ENDTRY yapısının işleyicisi tarafından yakalanmayan bir TRY bloğunda bir istisna oluştuğunda yürütülür. CLEANUP yan tümcesi içinde sistem, bir nesneyi tutarlı bir duruma geri yükleyebilir veya harici kaynakları serbest bırakabilir. Yani, TRY bloğu bağlamında temizleme çalışması yürütülebilir.

İstisnaları Artırma

İstisnalar, bir yöntemde, bir işlev modülünde, bir alt rutinde vb. herhangi bir noktada ortaya çıkabilir. Bir istisna oluşturmanın iki yolu vardır -

- ABAP çalışma zamanı sistemi tarafından oluşturulan istisnalar.
  Örneğin Y = 1 / 0. Bu, CX_SY_ZERODIVIDE türünde bir çalışma zamanı hatasıyla sonuçlanacaktır.
- Programcı tarafından oluşturulan istisnalar.
  Eşzamanlı olarak bir istisna nesnesi kaldırın ve oluşturun. İlk senaryoda zaten var olan bir istisna nesnesiyle bir istisna oluşturun. Sözdizimi şöyledir: RAISE İSTİSNAİ exep.

İstisnaları Yakalamak

İşleyiciler istisnaları yakalamak için kullanılır.

Bir kod parçacığına bir göz atalım -

```
DATA: result TYPE P LENGTH 8 DECIMALS 2,
exref TYPE REF TO CX_ROOT,
msgtxt TYPE STRING.
PARAMETERS: Num1 TYPE I, Num2 TYPE I.
TRY.
result = Num1 / Num2.
CATCH CX_SY_ZERODIVIDE INTO exref.
msgtxt = exref→GET_TEXT( ).

CATCH CX_SY_CONVERSION_NO_NUMBER INTO exref.
msgtxt = exref→GET_TEXT( ).
```

Yukarıdaki kod parçacığında, bir kayan noktalı değişkende sonucu elde etmek için Num1'i Num2'ye bölmeye çalışıyoruz.

İki tür istisna oluşturulabilir.

- Sayı dönüştürme hatası.
- Sıfır istisnaya bölün. İşleyiciler, CX_SY_CONVERSION_NO_NUMBER istisnasını ve ayrıca CX_SY_ZERODIVIDE istisnasını yakalar. Burada istisna sınıfının GET_TEXT() yöntemi, istisnanın açıklamasını almak için kullanılır.

İstisnaların Nitelikleri

İşte istisnaların beş özelliği ve yöntemi -

S.No.	Özellik ve Açıklama
1	**metin** İstisnalar için farklı metinler tanımlamak için kullanılır ve ayrıca get_text yönteminin sonucu
2	**Öncesi** Bu öznitelik, bir istisnalar zinciri oluşturmanıza izin veren orijinal istisnayı saklayabilir.
3	**get_text** Bu, istisnanın sistem diline göre metin gösterimini bir dize olarak döndürür.
4	**get_longtext** Bu, istisnanın metinsel temsilinin uzun türevini bir dize olarak döndürür.

5	**get_source_position** Özel durumun ortaya çıktığı yere ulaşılan program adını ve satır numarasını verir.

Örnek

```
REPORT ZExceptionsDemo.
PARAMETERS Num_1 TYPE I.

DATA res_1 TYPE P DECIMALS 2.
DATA orf_1 TYPE REF TO CX_ROOT.
DATA txt_1 TYPE STRING.

start-of-selection.
Write: / 'Square Root and Division with:', Num_1.
write: /.

TRY.
IF ABS( Num_1 ) > 150.
RAISE EXCEPTION TYPE CX_DEMO_ABS_TOO_LARGE.
ENDIF.

TRY.
res_1 = SQRT( Num_1 ).
Write: / 'Result of square root:', res_1.
res_1 = 1 / Num_1.

Write: / 'Result of division:', res_1.
CATCH CX_SY_ZERODIVIDE INTO orf_1.
txt_1 = orf_1→GET_TEXT( ).
CLEANUP.
CLEAR res_1.
ENDTRY.

CATCH CX_SY_ARITHMETIC_ERROR INTO orf_1.
txt_1 = orf_1→GET_TEXT( ).

CATCH CX_ROOT INTO orf_1.
txt_1 = orf_1→GET_TEXT( ).
ENDTRY.
IF NOT txt_1 IS INITIAL.
Write / txt_1.
ENDIF.
Write: / 'Final Result is:', res_1.
```

Bu örnekte, sayı 150'den büyükse, CX_DEMO_ABS_TOO_LARGE istisnası ortaya çıkar. Yukarıdaki kod, 160 sayısı için aşağıdaki çıktıyı üretir.

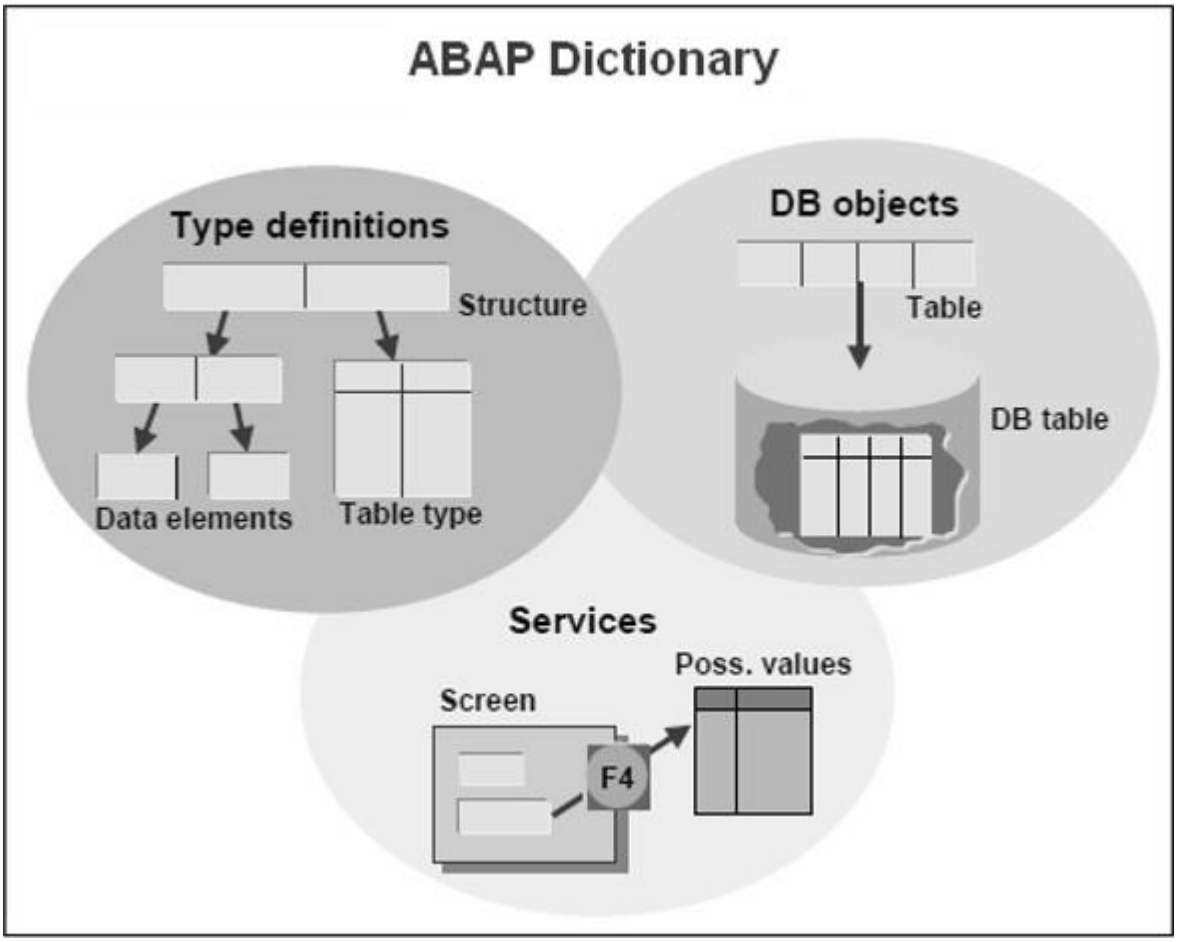
```
Square Root and Division with: 160
The absolute value of number is too high
Final Result is: 0.00
```

SAP ABAP - Sözlük

Bildiğiniz gibi, SQL iki bölüme ayrılabilir -

- DML (Veri Manipülasyon Dili)
- DDL (Veri Tanımlama Dili)

DML kısmı SELECT, INSERT, UPDATE, DELETE vb. sorgulama ve güncelleme komutlarından oluşur ve ABAP programları SQL'in DML kısmını işler. DDL bölümü CREATE TABLE, CREATE INDEX, DROP TABLE, ALTER TABLE vb. komutlardan oluşur ve ABAP Sözlüğü, SQL'in DDL bölümünü işler.



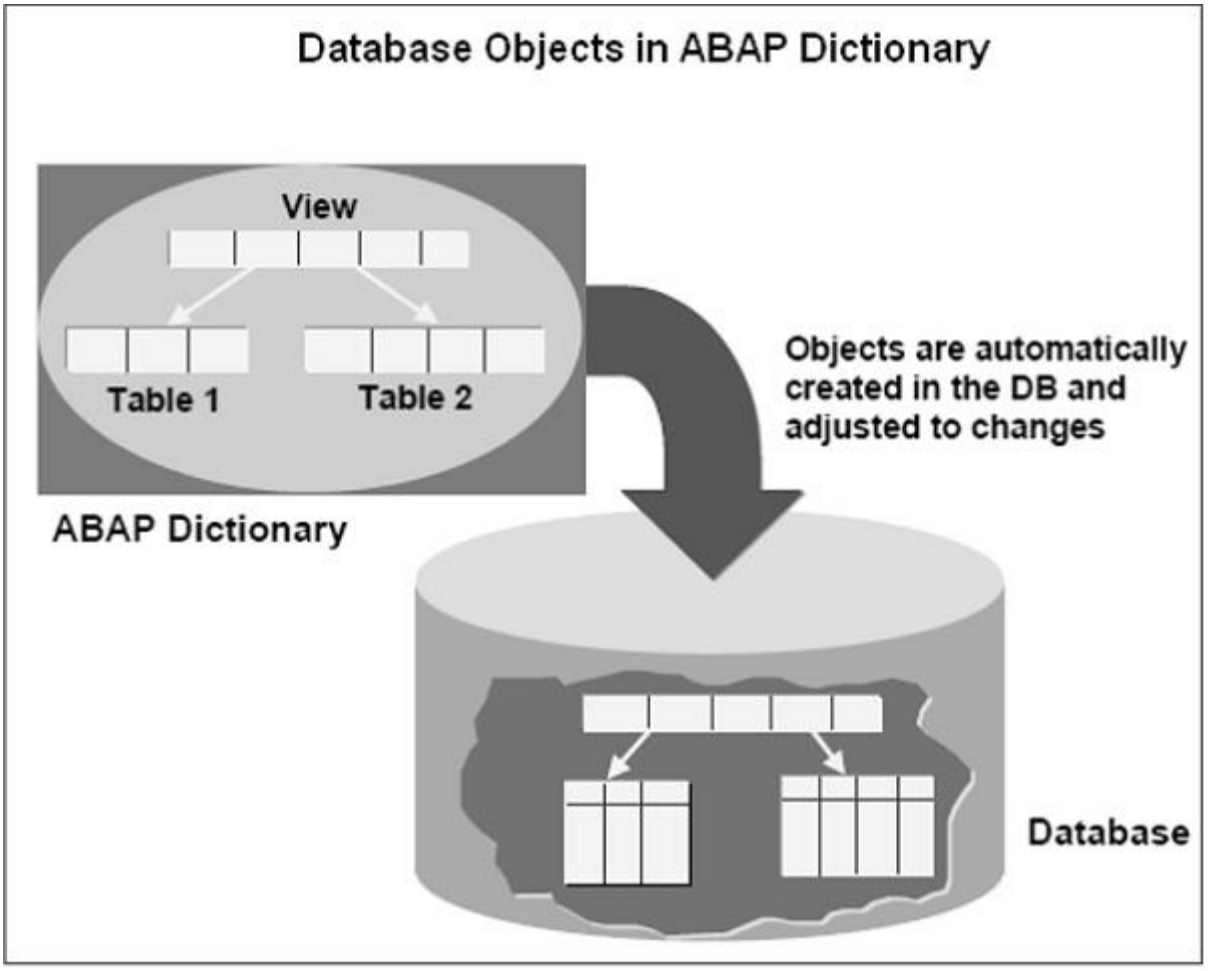
ABAP Sözlüğü, veritabanı tarafından tutulan meta verilerle birlikte SAP veritabanında bulunan meta veriler (yani veriler hakkındaki veriler) olarak görüntülenebilir. Sözlük, veri tanımları oluşturmak ve yönetmek ve Tablolar, Veri Öğeleri, Etki Alanları, Görünümler ve Türler oluşturmak için kullanılır.

ABAP Sözlüğünde Temel Türler

ABAP Sözlüğündeki temel türler aşağıdaki gibidir -

- **Veri öğeleri** , veri türünü, uzunluğunu ve muhtemelen ondalık basamakları tanımlayarak bir temel türü tanımlar.
- Herhangi bir türe sahip olabilen bileşenlere sahip **yapılar .**
- **Tablo türleri** , bir iç tablonun yapısını tanımlar.

ABAP programlarında Sözlük ortamındaki çeşitli nesnelere başvurulabilir. Sözlük, küresel alan olarak bilinir. Sözlükteki nesneler tüm ABAP programlarına geneldir ve ABAP programlarındaki veriler bu Sözlük genel nesnelerine başvurularak bildirilebilir.



Sözlük, kullanıcı tanımlı türlerin tanımını destekler ve bu türler ABAP programlarında kullanılır. Ayrıca tablolar, görünümler ve dizinler gibi veritabanı nesnelerinin yapısını da tanımlarlar. Bu nesneler, nesneler etkinleştirildiğinde, temeldeki veritabanında Sözlük tanımlarında otomatik olarak oluşturulur. Sözlük ayrıca Arama Yardımı gibi düzenleme araçları ve Nesneleri Kilitle gibi kilitleme aracı sağlar.

Sözlük Görevleri

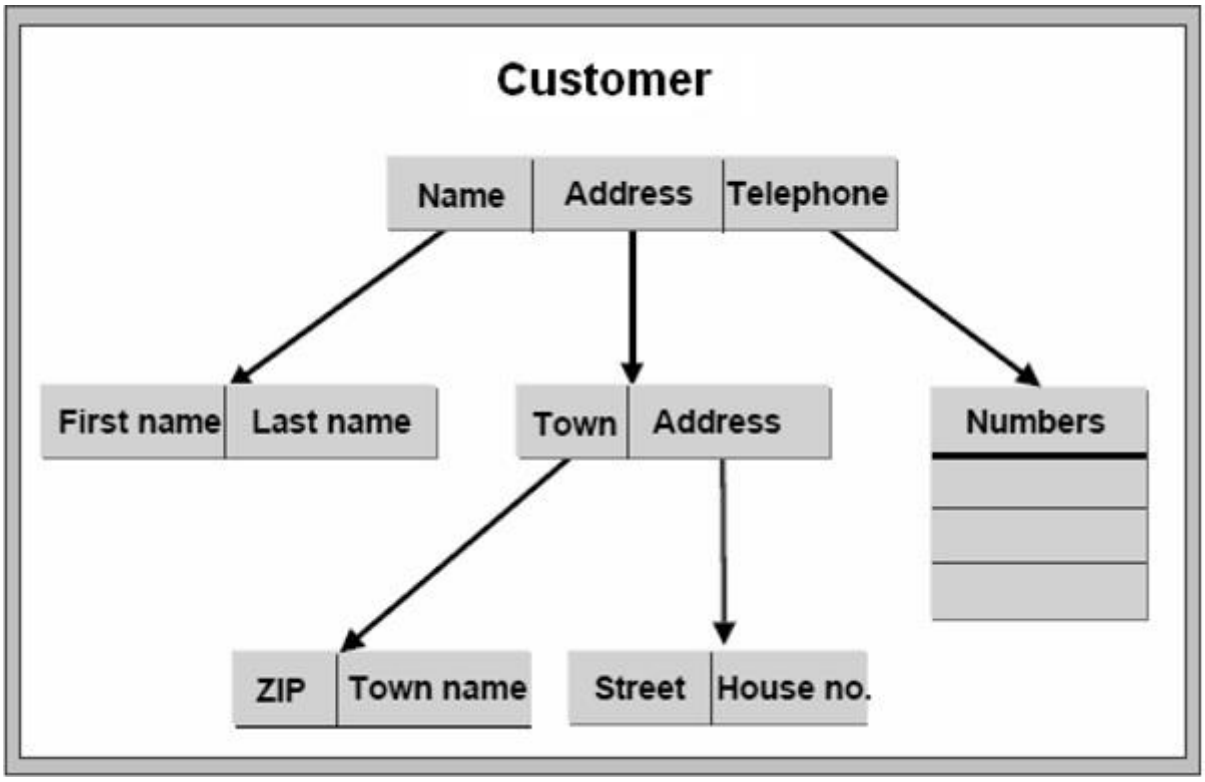
ABAP Sözlüğü aşağıdakileri başarır -

- Veri bütünlüğünü zorlar.
- Veri tanımlarını fazlalık olmadan yönetir.
- ABAP geliştirme tezgahının geri kalanıyla sıkı bir şekilde bütünleşir.

Örnek

Herhangi bir karmaşık kullanıcı tanımlı tür, Sözlükteki 3 temel türden oluşturulabilir. Müşteri verileri, aşağıdaki resimde gösterildiği gibi Ad, Adres ve Telefon bileşenleriyle bir 'Müşteri' yapısında saklanır. Ad ayrıca bileşenleri, Adı ve

Soyadı olan bir yapıdır. Bu bileşenlerin her ikisi de temeldir, çünkü türleri bir veri öğesi tarafından tanımlanır.



Bileşenin türü Adres, bileşenleri de yapı olan bir yapı tarafından tanımlanır ve Telefon bileşeni, bir müşterinin birden fazla telefon numarasına sahip olabileceği için bir tablo türü ile tanımlanır. Tipler ABAP programlarında ve ayrıca fonksiyon modüllerinin arayüz parametrelerinin tiplerini tanımlamak için kullanılır.

SAP ABAP - Etki Alanları

ABAP Sözlüğünde verileri tanımlamak için kullanılan üç temel nesne Etki Alanları, Veri öğeleri ve Tablolardır. Alan, alan türü ve uzunluğu gibi bir tablo alanının teknik tanımı için, veri öğesi ise anlamsal tanım (kısa açıklama) için kullanılır. Bir veri öğesi, belirli bir iş bağlamında bir etki alanının anlamını tanımlar. Öncelikle ekrandaki alan yardımını ve alan etiketlerini içerir.

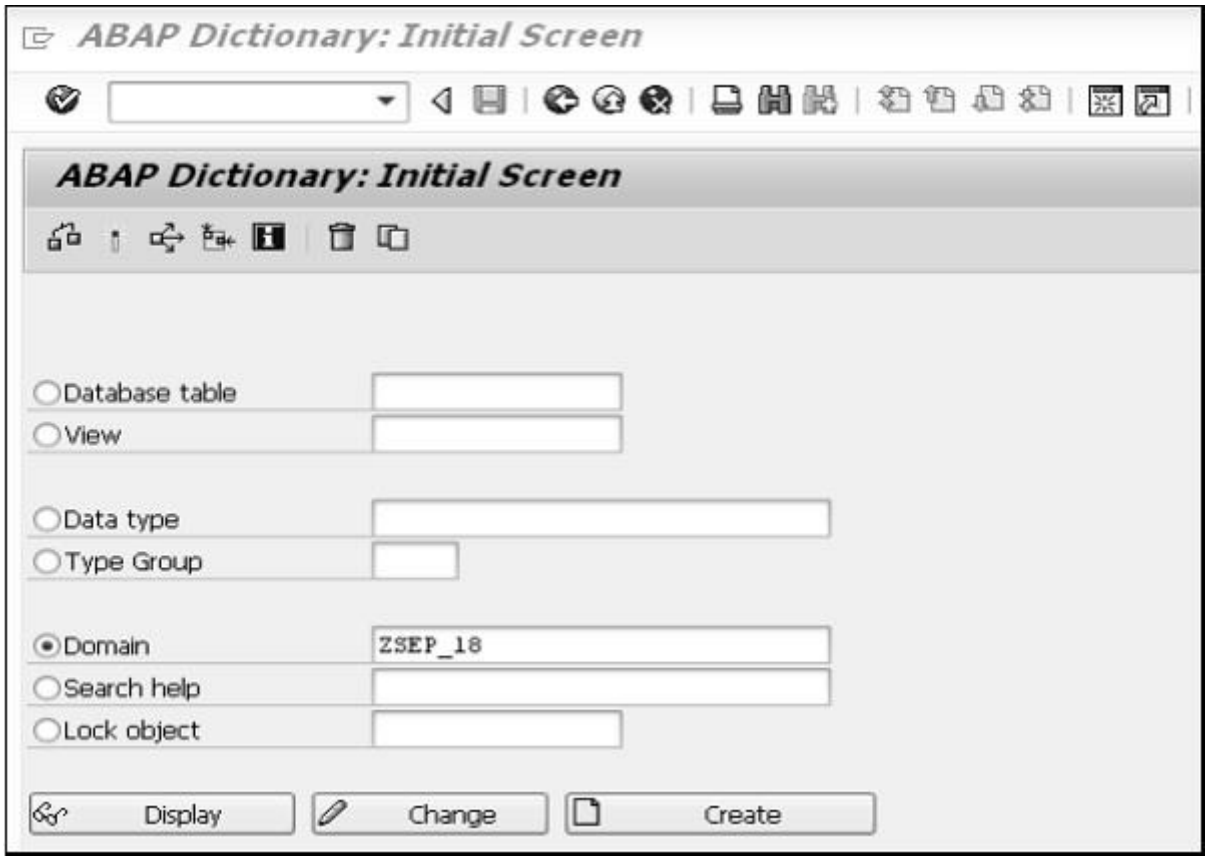
Etki alanı, sırayla tablo alanlarına veya yapı alanlarına atanan veri öğesine atanır. Örneğin, MATNR alanı (CHAR malzeme numarası), MATNR_N, MATNN ve MATNR_D gibi veri öğelerine atanır ve bunlar birçok tablo alanına ve yapı alanına atanır.

Etki Alanları Oluşturma

Yeni bir alan oluşturmadan önce, mevcut alan adlarından herhangi birinin tablo alanınızda gerekli olan teknik özelliklere sahip olup olmadığını kontrol edin. Eğer öyleyse, mevcut etki alanını kullanmamız gerekiyor. Etki alanı oluşturma prosedürünü tartışalım.

Adım 1 − İşlem SE11'e gidin.

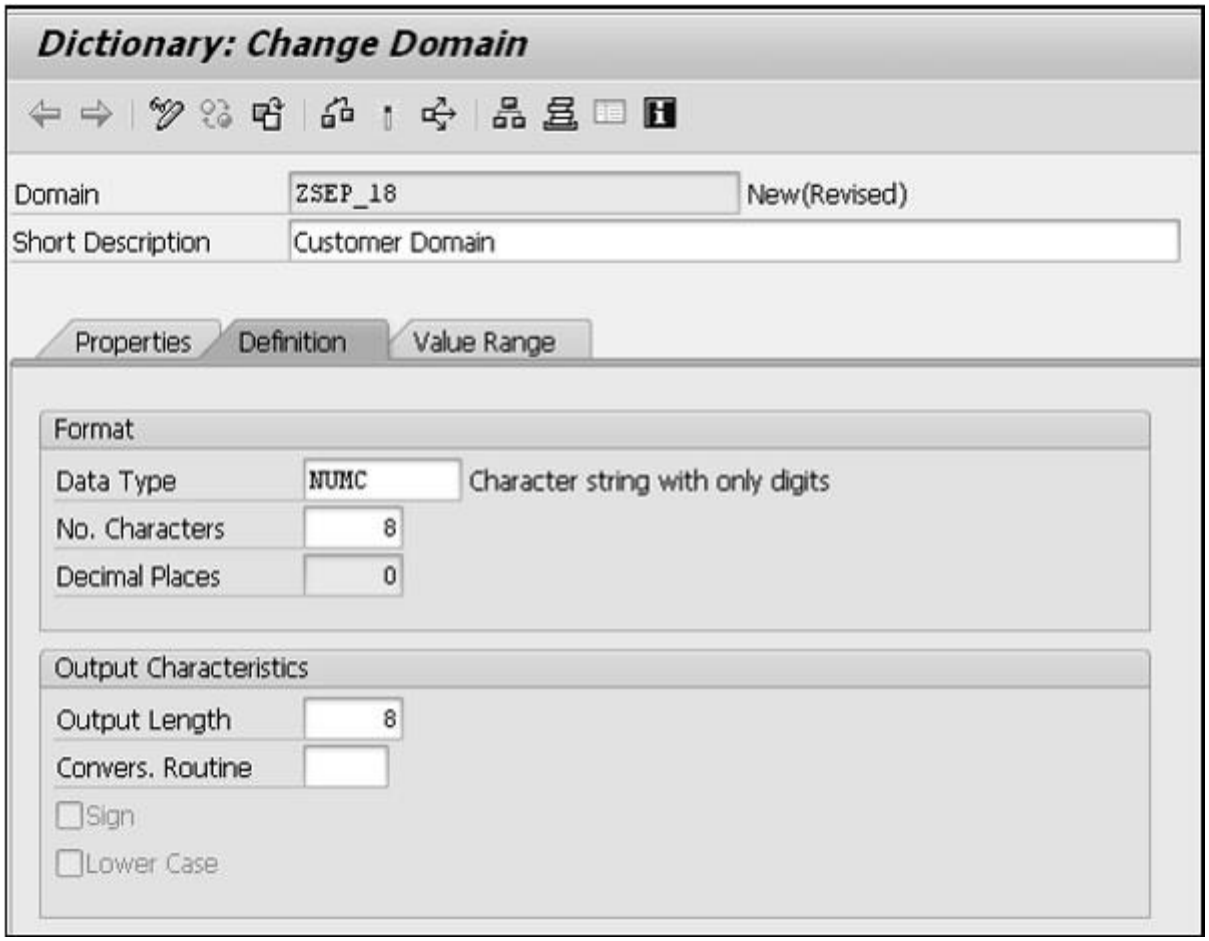
Adım 2 − ABAP Sözlüğünün ilk ekranında Alan için radyo düğmesini seçin ve aşağıdaki ekran görüntüsünde gösterildiği gibi alan adını girin. OLUŞTUR düğmesine tıklayın. Müşteri ad alanları altında etki alanları oluşturabilirsiniz ve nesnenin adı her zaman 'Z' veya 'Y' ile başlar.



Adım 3 − Alan adının bakım ekranının kısa metin alanına açıklamayı girin. Bu durumda, “Müşteri Alanı” dır. **Not** − Bu niteliği girmeden başka bir nitelik giremezsiniz.

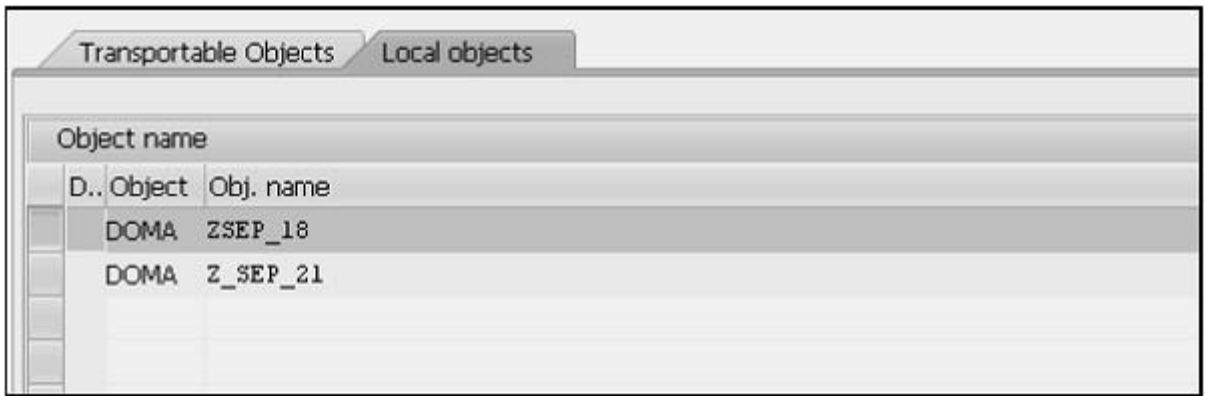
Adım 4 − Tanım sekmesinin Biçim bloğuna Veri Türü, Karakter Sayısı ve Ondalık Basamakları girin. Çıktı Uzunluğu üzerindeki tuşuna basın, çıktı uzunluğunu önerir ve görüntüler. Önerilen çıktı uzunluğunun üzerine yazarsanız, etki alanını etkinleştirirken bir uyarı görebilirsiniz. Convers'ı doldurabilirsiniz. Gerekirse Rutin, İşaret ve Küçük Harf alanları. Ancak bunlar her zaman isteğe bağlı niteliklerdir.

Adım 5 − Değer Aralığı sekmesini seçin. Etki alanı yalnızca sabit değerlerle sınırlandırılmışsa, sabit değerleri veya aralıkları girin. Bu etki alanına atıfta bulunan alanlar için bir yabancı anahtar tanımlarken sistem bu tabloyu bir kontrol tablosu olarak önermek zorundaysa değer tablosunu tanımlayın. Ancak tüm bunlar isteğe bağlı niteliklerdir.



Adım 6 – Değişikliklerinizi kaydedin. Nesne Dizini Girişi Oluştur açılır penceresi görünür ve bir paket ister. Çalıştığınız paket adını girebilirsiniz. Herhangi bir paketiniz yoksa, onu Object Navigator'da oluşturabilir veya Local Object düğmesini kullanarak etki alanınızı kaydedebilirsiniz.

Adım 7 – Alanınızı etkinleştirin. Etki alanını etkinleştirmek için Etkinleştir simgesine (kibrit çöpü simgesi) tıklayın veya CTRL + F3 tuşlarına basın. Aşağıdaki anlık görüntüde gösterildiği gibi, şu anda etkin olmayan 2 nesneyi listeleyen bir açılır pencere görünür -

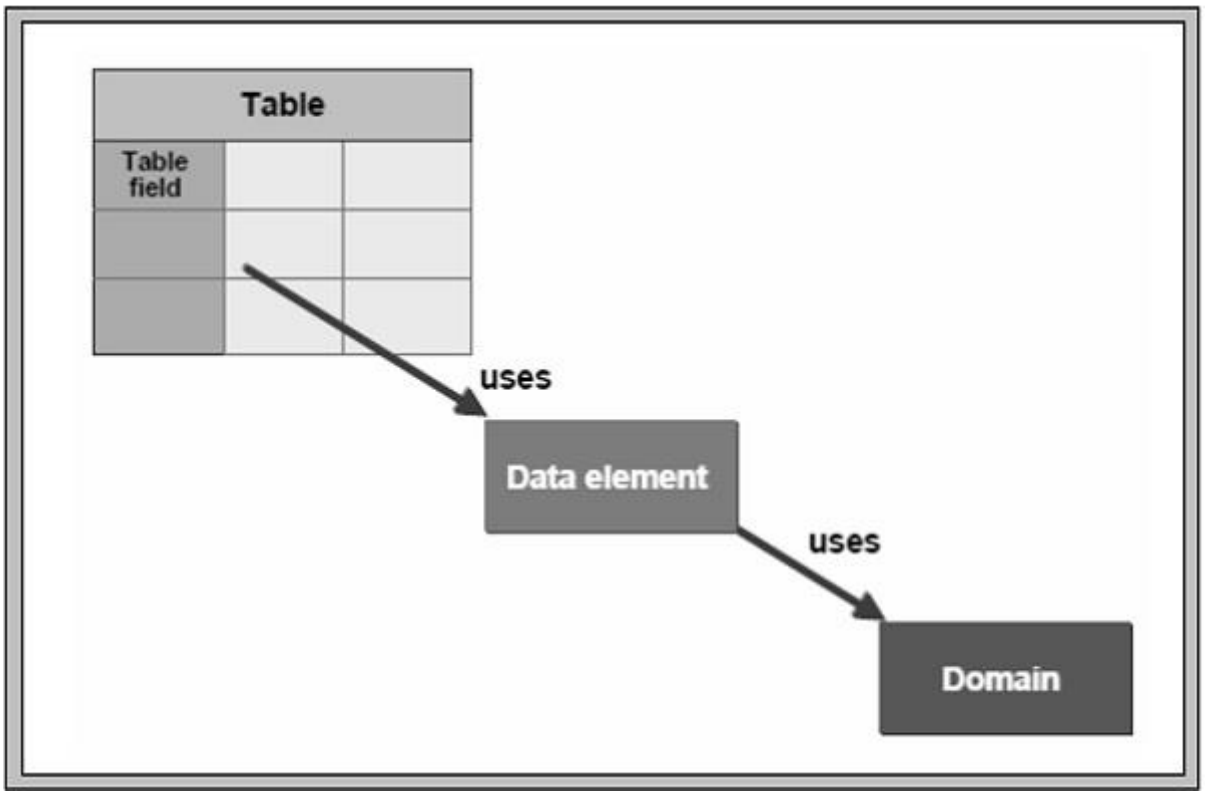


Adım 8 − Bu noktada, ZSEP_18 adıyla 'DOMA' etiketli üst giriş etkinleştirilecektir. Bu vurgulandığında, yeşil onay düğmesine tıklayın. Bu pencere kaybolur ve durum çubuğu 'Nesne etkinleştirildi' mesajını görüntüler.

Etki alanını etkinleştirdiğinizde hata mesajları veya uyarılar oluştuysa, etkinleştirme günlüğü otomatik olarak görüntülenir. Etkinleştirme günlüğü, etkinleştirme akışıyla ilgili bilgileri görüntüler. Etkinleştirme günlüğünü Utilities(M) → Etkinleştirme günlüğü ile de arayabilirsiniz.

SAP ABAP - Veri Öğeleri

Veri öğeleri, ABAP Veri Sözlüğü'ndeki tek tek alanları tanımlar. Karmaşık türlerin en küçük bölünemez birimleridir ve tablo alanının türünü, yapı bileşenini veya bir tablonun satır türünü tanımlamak için kullanılırlar. Bir tablo alanının anlamı hakkında bilgi ve ayrıca ilgili ekran alanının düzenlenmesi hakkında bilgi bir veri öğesine atanabilir. Bu bilgi, veri öğesine atıfta bulunan tüm ekran alanlarında otomatik olarak mevcuttur. Veri öğeleri, temel türleri veya referans türlerini tanımlar.



Veri Öğeleri Oluşturma

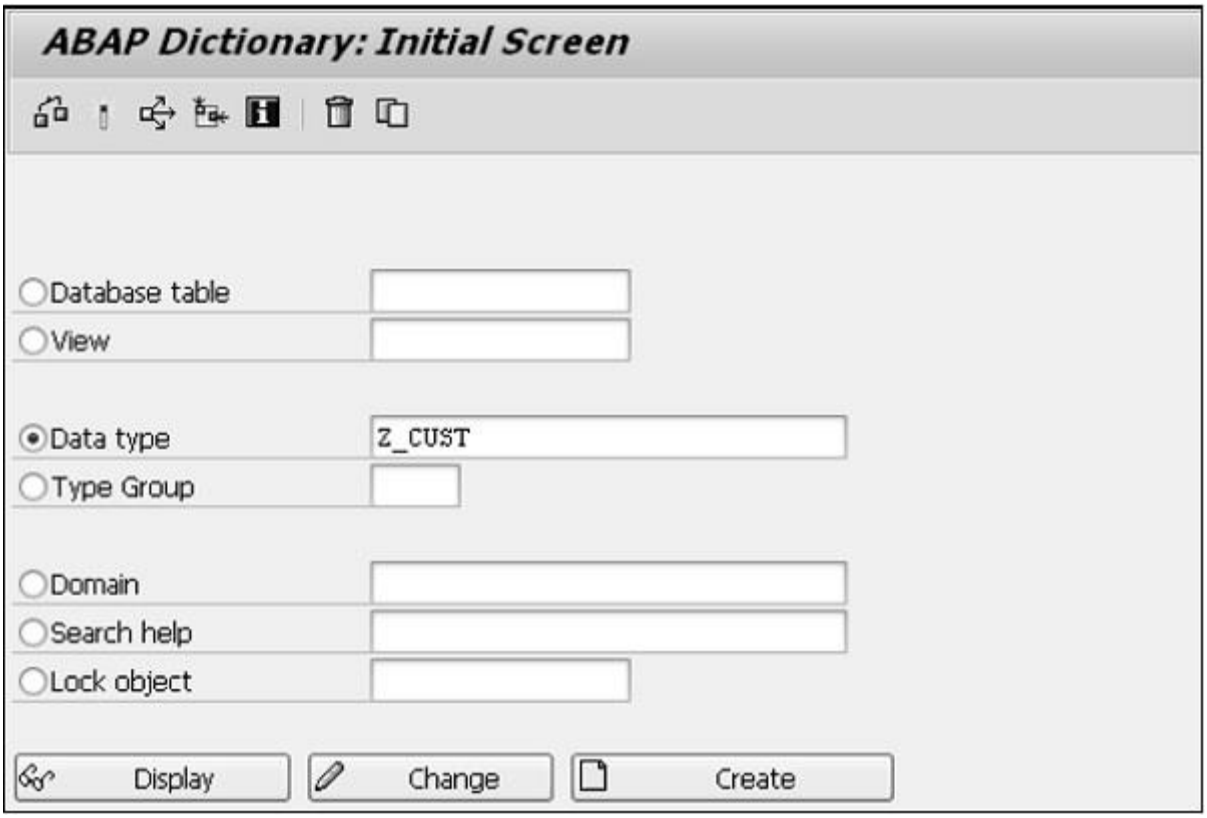
Yeni bir veri öğesi oluşturmadan önce, mevcut herhangi bir veri öğesinin tablo alanınızda gerekli olan aynı anlamsal özelliklere sahip olup olmadığını kontrol etmeniz gerekir. Eğer öyleyse, mevcut veri öğesini kullanabilirsiniz. Veri öğesine önceden tanımlanmış bir tür, etki alanı veya referans türü atayabilirsiniz.

Veri öğesini oluşturma prosedürü aşağıdadır -

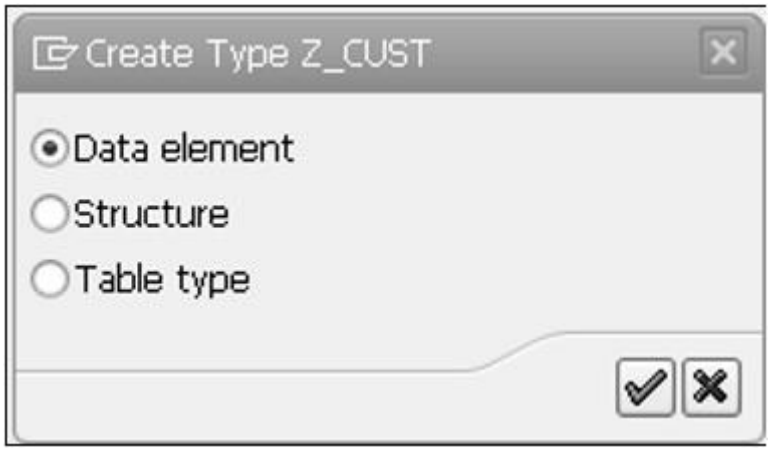
Adım 1 − İşlem SE11'e gidin.

Adım 2 − ABAP Sözlüğünün ilk ekranında Veri türü için radyo düğmesini seçin ve aşağıda gösterildiği gibi veri öğesinin adını girin.

Adım 3 − OLUŞTUR düğmesine tıklayın. Müşteri ad alanları altında veri öğeleri oluşturabilirsiniz ve nesnenin adı her zaman 'Z' veya 'Y' ile başlar.

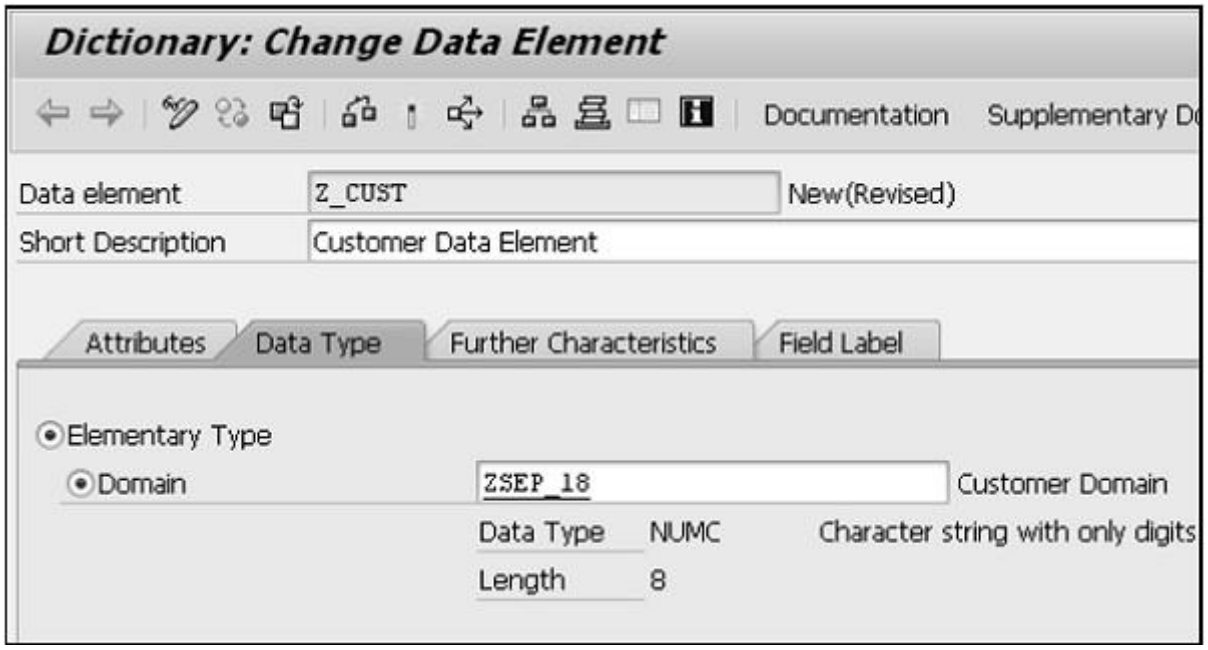


Adım 4 − Üç radyo düğmesiyle birlikte görünen CREATE TYPE açılır penceresindeki Veri öğesi radyo düğmesini kontrol edin.



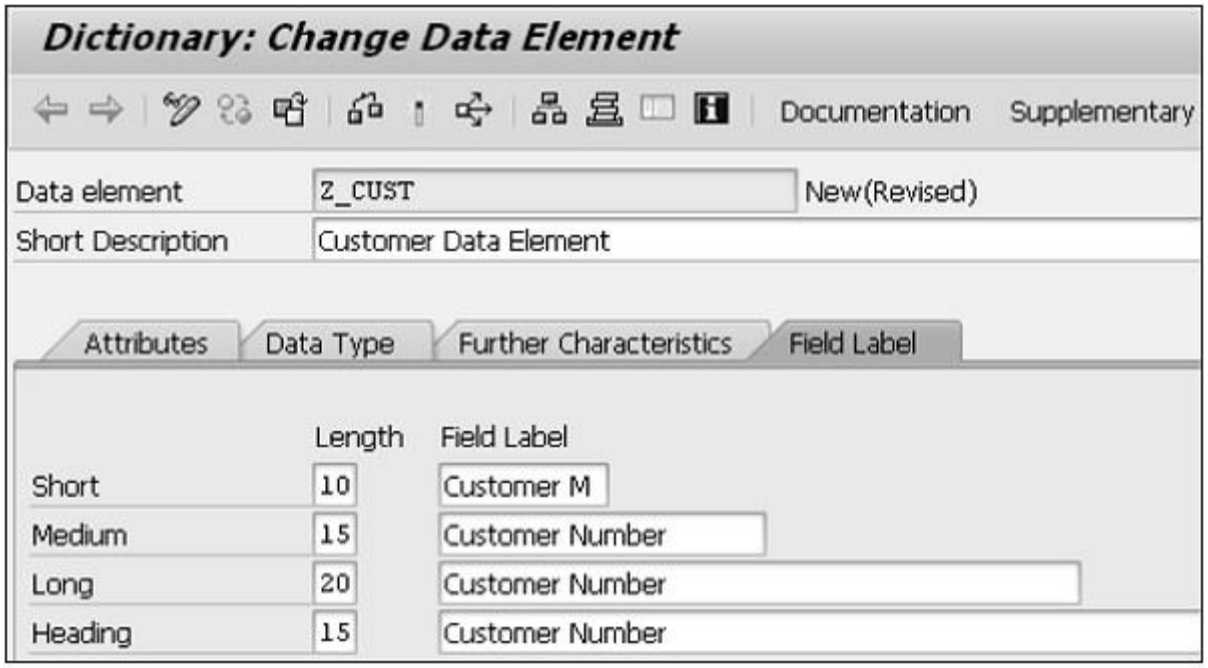
Adım 5 − Yeşil onay işareti simgesini tıklayın. Veri öğesinin bakım ekranına yönlendirilirsiniz.

Adım 6 − Veri öğesinin bakım ekranının kısa metin alanına açıklamayı girin. Bu durumda "Müşteri Veri Öğesi" dir. **Not** − Bu niteliği girmeden başka bir nitelik giremezsiniz.



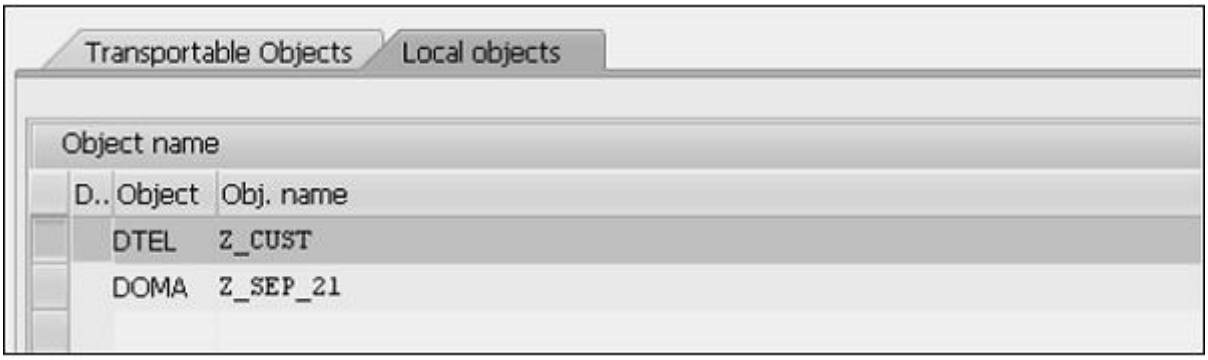
Adım 7 − Veri öğesini tiple atayın. Elementer tipi kontrol ederek bir elementer veri elemanı veya Referans tipini kontrol ederek bir referans veri elemanı oluşturabilirsiniz. Bir Veri öğesini Elementary Type içinde bir Etki Alanına veya Önceden Tanımlı Türe ve Referans Türü içinde Referans Türü Adı veya Öntanımlı Türe Referans ile atayabilirsiniz.

Adım 8 − Alan Etiketi sekmesinde kısa metin, orta metin, uzun metin ve başlık alanlarını girin. Enter tuşuna bastığınızda bu etiketler için uzunluk otomatik olarak oluşturulur.



Adım 9 – Değişikliklerinizi kaydedin. Nesne Dizini Girişi Oluştur açılır penceresi görünür ve bir paket ister. Çalıştığınız paket adını girebilirsiniz. Herhangi bir paketiniz yoksa, onu Object Navigator'da oluşturabilir veya Local Object düğmesini kullanarak veri öğenizi kaydedebilirsiniz.

Adım 10 − Veri öğenizi etkinleştirin. Veri öğesini etkinleştirmek için Etkinleştir simgesine (kibrit çöpü simgesi) tıklayın veya CTRL + F3 tuşlarına basın. Aşağıdaki ekran görüntüsünde gösterildiği gibi, şu anda etkin olmayan 2 nesneyi listeleyen bir açılır pencere belirir.



Adım 11 − Bu noktada, Z_CUST adındaki 'DTEL' etiketli üst giriş etkinleştirilecektir. Bu vurgulandığında, yeşil onay düğmesine tıklayın. Bu pencere kaybolur ve durum çubuğu 'Nesne etkinleştirildi' mesajını görüntüler.

Veri öğesini etkinleştirdiğinizde hata mesajları veya uyarılar oluşursa, etkinleştirme günlüğü otomatik olarak görüntülenir. Etkinleştirme günlüğü, etkinleştirme akışıyla ilgili bilgileri görüntüler. Etkinleştirme günlüğünü Utilities(M) → Etkinleştirme günlüğü ile de arayabilirsiniz.

SAP ABAP - Tablolar

ABAP Sözlüğü'nde tablolar veritabanından bağımsız olarak tanımlanabilir. ABAP Sözlüğü'nde bir tablo etkinleştirildiğinde, veritabanında da alanlarının benzer bir kopyası oluşturulur. ABAP Sözlüğü'nde tanımlanan tablolar, tablonun tanımı SAP sistemi tarafından kullanılan veri tabanına bağlı olduğundan, veri tabanına uygun formata otomatik olarak çevrilir.

Bir tablo, her biri kendi veri türü ve uzunluğu ile tanımlanmış bir veya daha fazla alan içerebilir. Bir tabloda depolanan büyük miktarda veri, tabloda tanımlanan çeşitli alanlara dağıtılır.

Tablo Alanı Türleri

Bir tablo birçok alandan oluşur ve her alan birçok öğe içerir. Aşağıdaki tablo, tablo alanlarının farklı öğelerini listeler -

S.No.	Öğeler ve Açıklama
1	**Alan adı** Bu, maksimum 16 karakter içerebilen bir alana verilen addır. Alan adı rakamlardan, ha çizgilerden oluşabilir. Bir harfle başlamalıdır.

2	**Anahtar bayrak** Bir alanın bir anahtar alana ait olup olmadığını belirler.
3	**Alan türü** Bir alana bir veri türü atar.
4	**alan uzunluğu** Bir alana girilebilecek karakter sayısı.
5	**Ondalık** Ondalık noktadan sonra izin verilen basamak sayısını tanımlar. Bu öğe yalnızca sayısal kullanılır.
6	**Kısa metin** İlgili alanın anlamını açıklar.

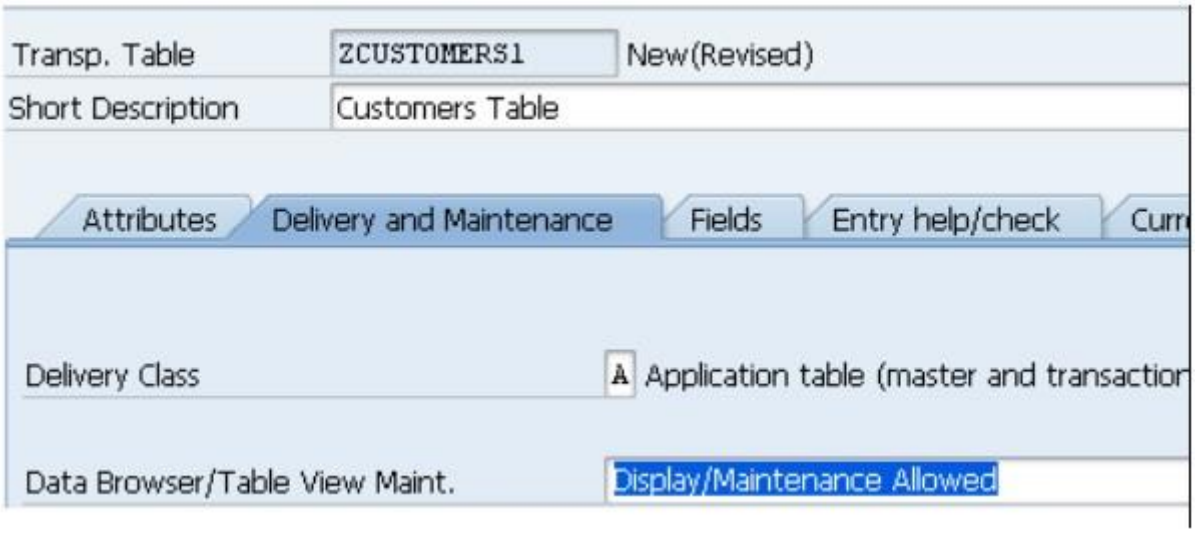
ABAP Sözlüğünde Tablo Oluşturma

Adım 1 − SE11 işlemine gidin, 'Veritabanı tablosu' radyo düğmesini seçin ve oluşturulacak tablo için bir ad girin. Bizim durumumuzda ZCUSTOMERS1 adını girdik. Oluştur düğmesini tıklayın. Sözlük: Tabloyu Koru ekranı görünür. Burada 'Teslimat ve Bakım' sekmesi varsayılan olarak seçilidir.

Adım 2 − Kısa Açıklama alanına açıklayıcı bir kısa metin girin.

Adım 3 − Teslimat Sınıfı alanının yanındaki Yardımı Ara simgesine tıklayın. 'A [Uygulama tablosu (ana ve işlem verileri)]' seçeneğini seçin.

Adım 4 − 'Veri Tarayıcı/Tablo görünümü Bakım' açılır menüsünden 'Görüntüleme/Bakım İzin Verilir' seçeneğini seçin. Sözlük: Bakım Tablosu ekranı görünür.



Adım 5 − Alanlar sekmesini seçin. Alanlar sekmesi ile ilgili seçenekleri içeren ekran görüntülenir.

Adım 6 − Alan sütununa tablo alanlarının adlarını girin. Bir alan adı harf, rakam ve alt çizgi içerebilir, ancak her zaman bir harfle başlamalı ve 16 karakterden uzun olmamalıdır.

Veri türü, uzunluk, ondalık basamaklar ve kısa metin gibi öznitelikleri tanımlanan veri öğesinden aldıkları için, oluşturulacak alanlar da veri öğelerine sahip olmalıdır.

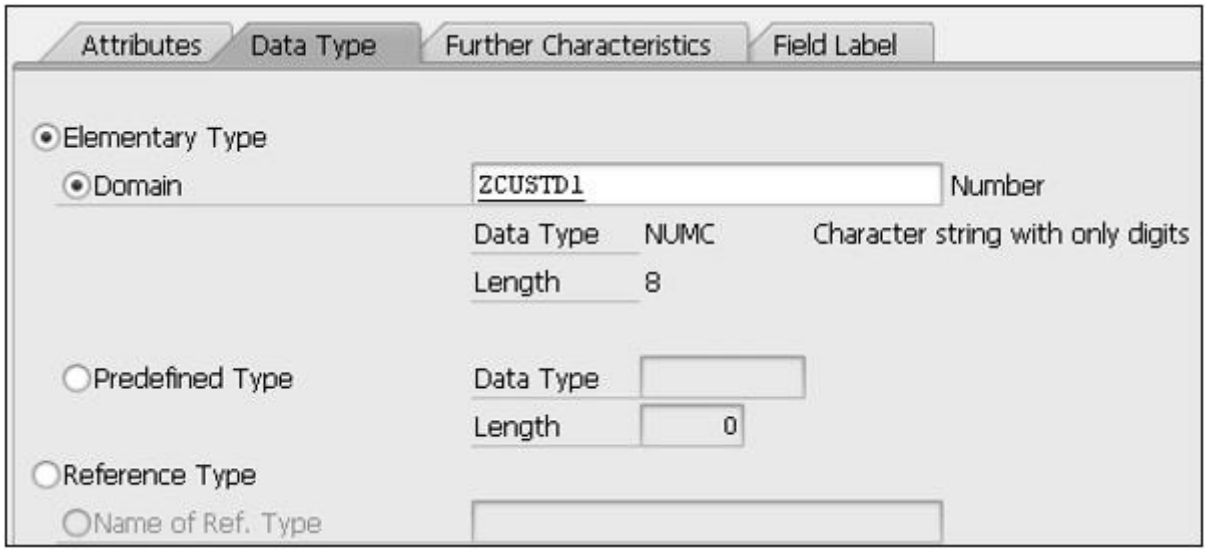
Adım 7 − Alanın tablo anahtarının bir parçası olmasını istiyorsanız Anahtar sütununu seçin. MÜŞTERİ, MÜŞTERİ, İSİM, BAŞLIK ve DOB gibi alanlar oluşturalım.

Adım 8 − İlk alan önemli bir alandır ve kayıtların ilişkilendirildiği müşteriyi tanımlar. Alan olarak 'Müşteri' ve Veri Öğesi olarak 'MANDT' girin. Sistem, Veri Türü, Uzunluk, Ondalık Sayılar ve Kısa Açıklama bilgilerini otomatik olarak doldurur. 'Müşteri' alanı, 'Anahtar' kutusu işaretlenerek bir anahtar alan haline getirilir.

Adım 9 − Bir sonraki alan 'Müşteri'dir. Anahtar alanı yapmak için kutuyu işaretleyin ve yeni 'ZCUSTNUM' Veri Öğesini girin. Kaydet düğmesini tıklayın.

Adım 10 − 'ZCUSTNUM' Veri Öğesi henüz mevcut olmadığından, yaratılması gerekir. Yeni Veri Öğesine çift tıklayın ve 'Veri Öğesi Oluştur' penceresi görünür. Buna 'Evet' yanıtı verin ve bir 'Veri Öğesini Koru' penceresi belirir.

Adım 11 − Kısa Açıklama alanına 'Müşteri Numarası' girin. Yeni Veri öğesi için 'Etki Alanı' adlı Elementary veri türü tanımlanmalıdır. Bu yüzden 'ZCUSTD1' girin, çift tıklayın ve yapılan değişiklikleri kaydetmeyi kabul edin. Etki alanını oluşturmak için "Evet"i seçin ve "Kısa Açıklama" kutusuna etki alanının açıklamasını yazın.

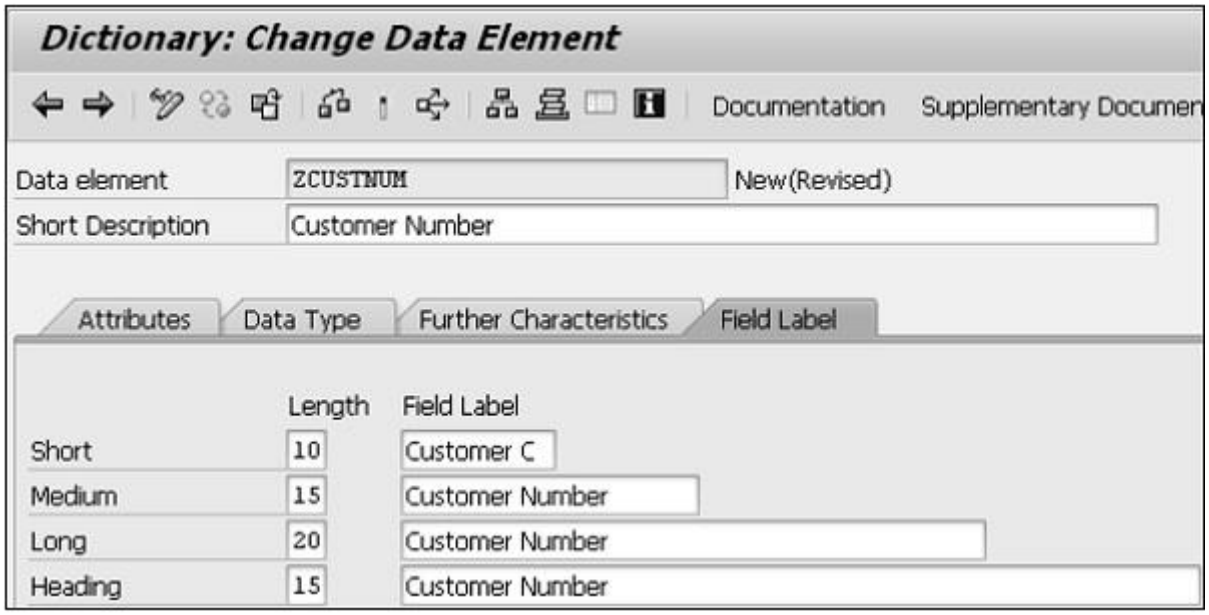


'Tanım' sekmesi otomatik olarak açılır. İlk alan 'Veri Türü'dür.

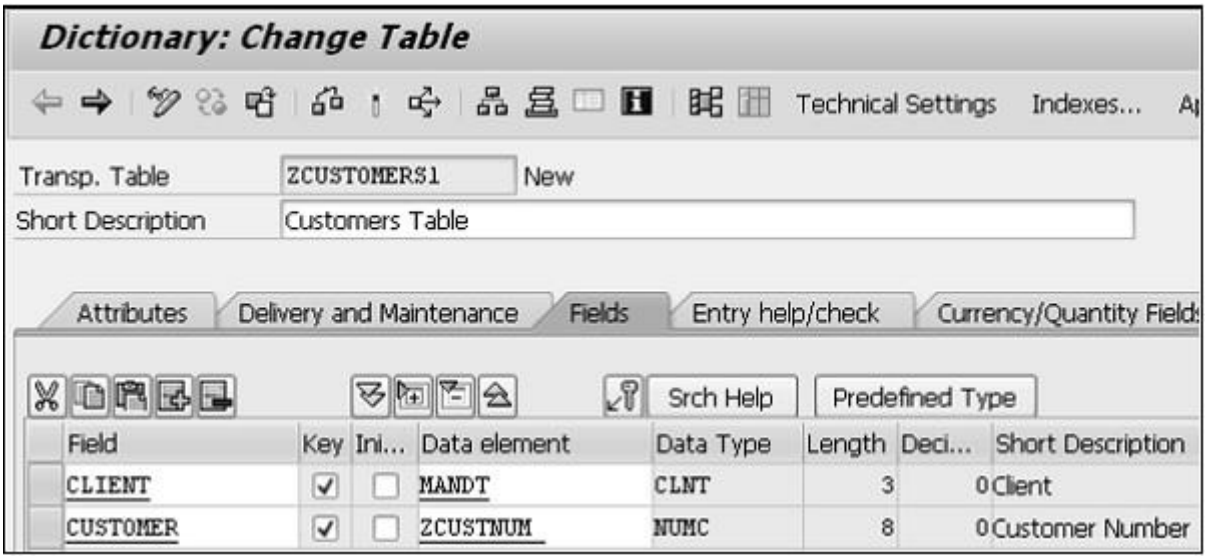
Adım 12 − Kutunun içine tıklayın ve açılır menüden 'NUMC' tipini seçin. 'Hayır' alanına 8 sayısını girin. of karakter' alanına (en fazla 8 karakter) girin ve 'Ondalık basamaklar' alanına 0 girin. Çıktı uzunluğu 8 seçilmeli ve ardından Enter tuşuna basılmalıdır. 'NUMC' alanının açıklaması, bunun geçerli bir giriş olduğunu onaylayarak yeniden görünmelidir.

Adım 13 − Kaydet düğmesine tıklayın ve nesneyi etkinleştirin.

Adım 14 − 'Veri Öğesini Koru/Değiştir' ekranına dönmek için F3'e basın. Aşağıdaki anlık görüntüde gösterildiği gibi dört Alan etiketi oluşturun. Bundan sonra, öğeyi kaydedin ve etkinleştirin.



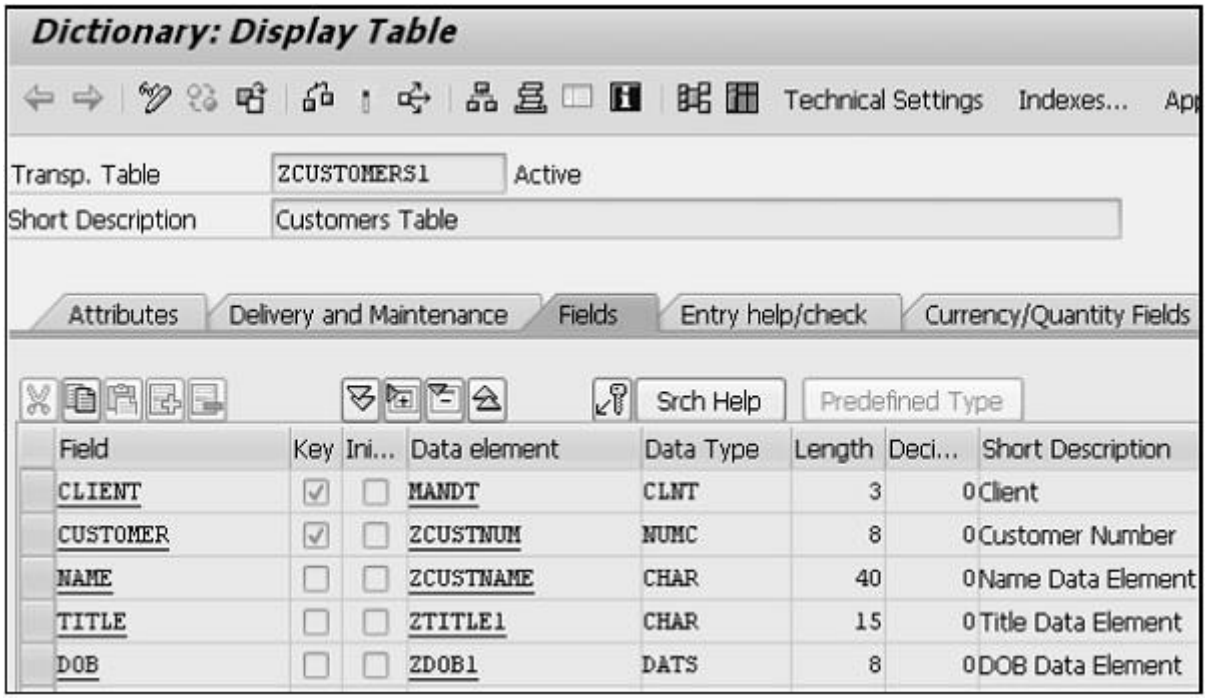
Adım 15 − Tablo bakım ekranına dönmek için geri düğmesine basın. Müşteri sütununda doğru Veri Türü, Uzunluk, Ondalık Sayılar ve Kısa Açıklama bulunur. Bu, bir Veri öğesinin ve ayrıca kullanılan Etki Alanının başarıyla oluşturulduğunu gösterir.



Benzer şekilde NAME, TITLE ve DOB gibi üç ek alan daha oluşturmamız gerekiyor.

Adım 16 − Araç çubuğundan 'Teknik ayarlar'ı seçin. 'Veri sınıfı' için APPL0'ı ve 'Boyut' kategorisi' alanı için ilk boyut kategorisi 0'ı seçin. Arabelleğe alma seçenekleri olması durumunda, 'Tamponlamaya izin verilmez' seçilmelidir.

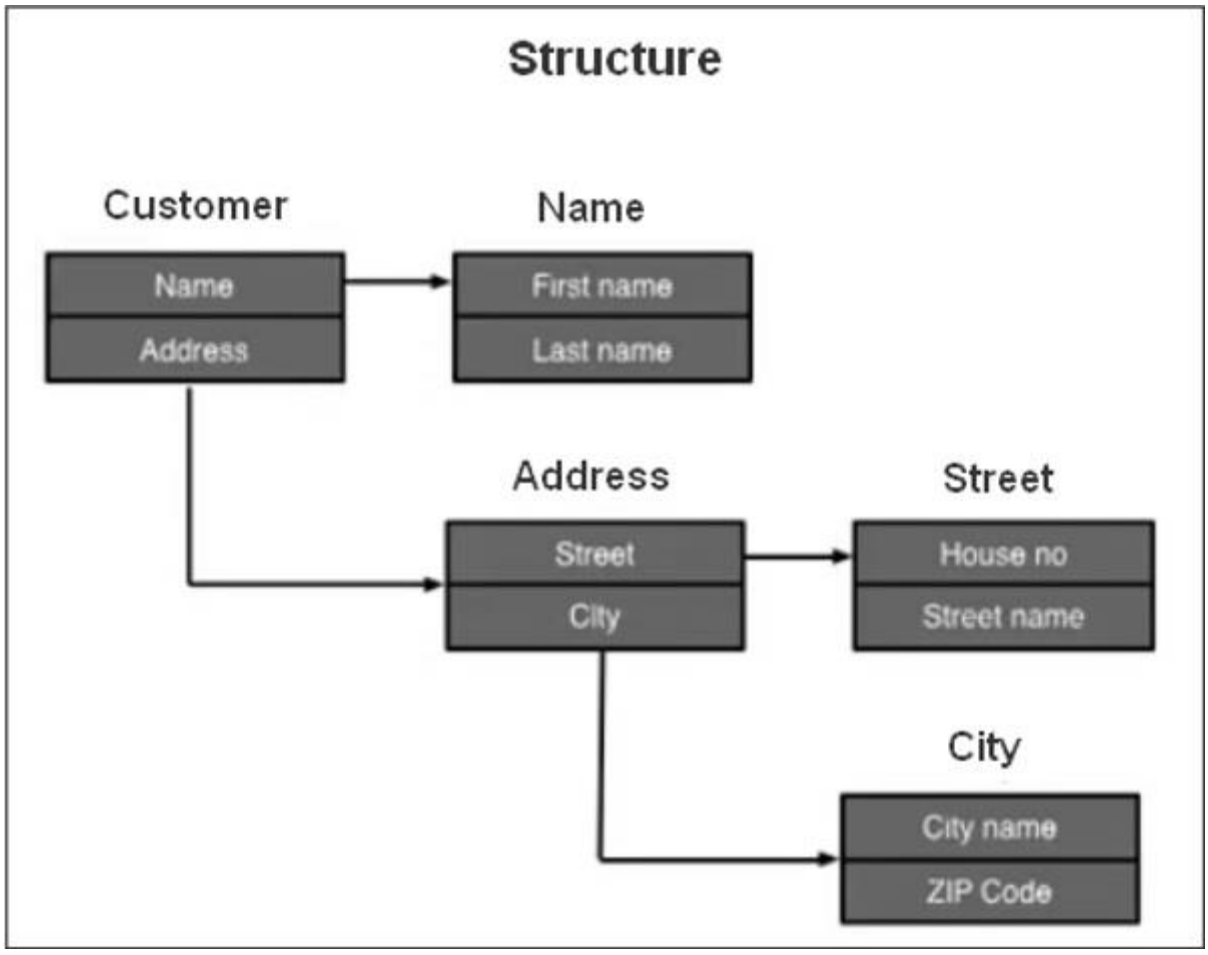
Adım 17 − Kaydet'e tıklayın. Masaya geri dönün ve etkinleştirin. Aşağıdaki ekran belirir.



'ZCUSTOMERS1' tablosu etkinleştirildi.

SAP ABAP - Yapılar

Yapı , bellekte birbiri ardına saklanan herhangi bir veri türünün bileşenlerinden oluşan bir veri nesnesidir.



Yapılar, ekran alanlarını boyamak ve belirli sayıda alan tarafından tanımlanan tutarlı bir biçime sahip verileri işlemek için kullanışlıdır.

Bir yapının çalışma zamanında yalnızca tek bir kaydı olabilir, ancak bir tablonun birçok kaydı olabilir.

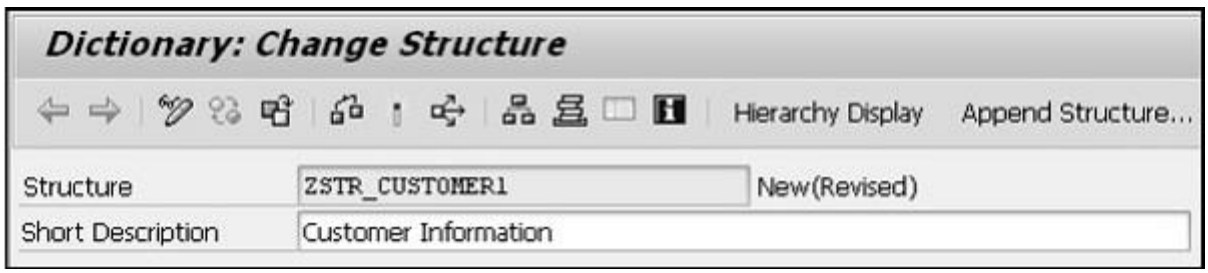
Yapı Oluşturma

Adım 1 − SE11 işlemine gidin.

Adım 2 − Ekrandaki 'Veri türü' seçeneğine tıklayın. 'ZSTR_CUSTOMER1' adını girin ve Oluştur düğmesine tıklayın.

Adım 3 − Bir sonraki ekranda 'Yapı' seçeneğini seçin ve Enter'a basın. 'Yapıyı Koru / Değiştir' sihirbazını görebilirsiniz.

Adım 4 − Aşağıdaki anlık görüntüde gösterildiği gibi Kısa Açıklamayı girin.

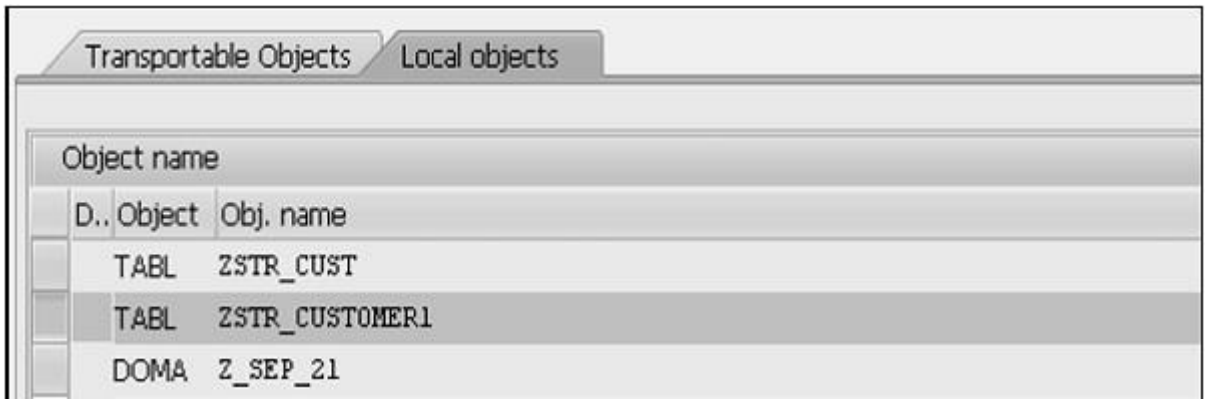


Adım 5 − Bileşen (Alan Adı) ve Bileşen Türünü (Veri Öğesi) girin.

Not : Burada bileşen adları SAP tavsiyesine göre Z ile başlar. Veritabanı tablosunda önceden oluşturduğumuz veri öğelerini kullanalım.

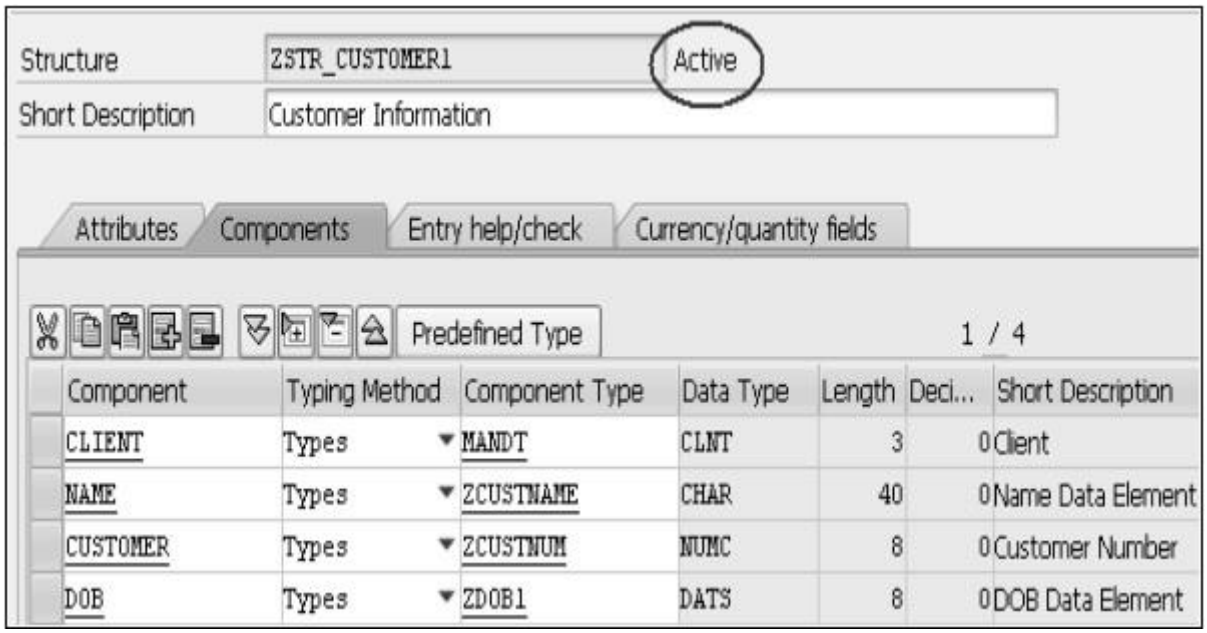
Adım 6 − Tüm bileşenleri ve bileşen türlerini sağladıktan sonra Kaydet, Kontrol Et ve Etkinleştir.

Aşağıdaki ekran belirir -



Adım 7 − Bu 'ZSTR_CUSTOMER1' vurgulandığından, yeşil onay düğmesine tıklayın. Bu pencere kaybolur ve durum çubuğu 'Etkin' mesajını görüntüler.

Yapı şimdi aşağıdaki anlık görüntüde gösterildiği gibi etkinleştirilmiştir -



SAP ABAP - Görünümler

Bir Görünüm yalnızca bir veritabanı tablosu gibi davranır. Ancak depolama alanı işgal etmeyecektir. Görünüm, sanal bir tabloya benzer şekilde davranır - herhangi bir fiziksel varlığı olmayan bir tablo. Bir uygulama nesnesi hakkında bilgi içeren bir veya daha fazla tablonun verilerinin birleştirilmesiyle bir görünüm oluşturulur. Görünümleri kullanarak, bir tabloda bulunan verilerin bir alt kümesini temsil edebilir veya birden çok tabloyu tek bir sanal tabloda birleştirebilirsiniz.

Bir uygulama nesnesiyle ilgili veriler, veritabanı görünümleri kullanılarak birden çok tablo arasında dağıtılır. Farklı tabloların verilerini birleştirmek için iç birleştirme koşulunu kullanırlar. Bir uygulama nesnesinde depolanan verileri görüntülemek ve değiştirmek için bir bakım görünümü kullanılır. Her bakım görünümünün kendisiyle ilişkilendirilmiş bir bakım durumu vardır.

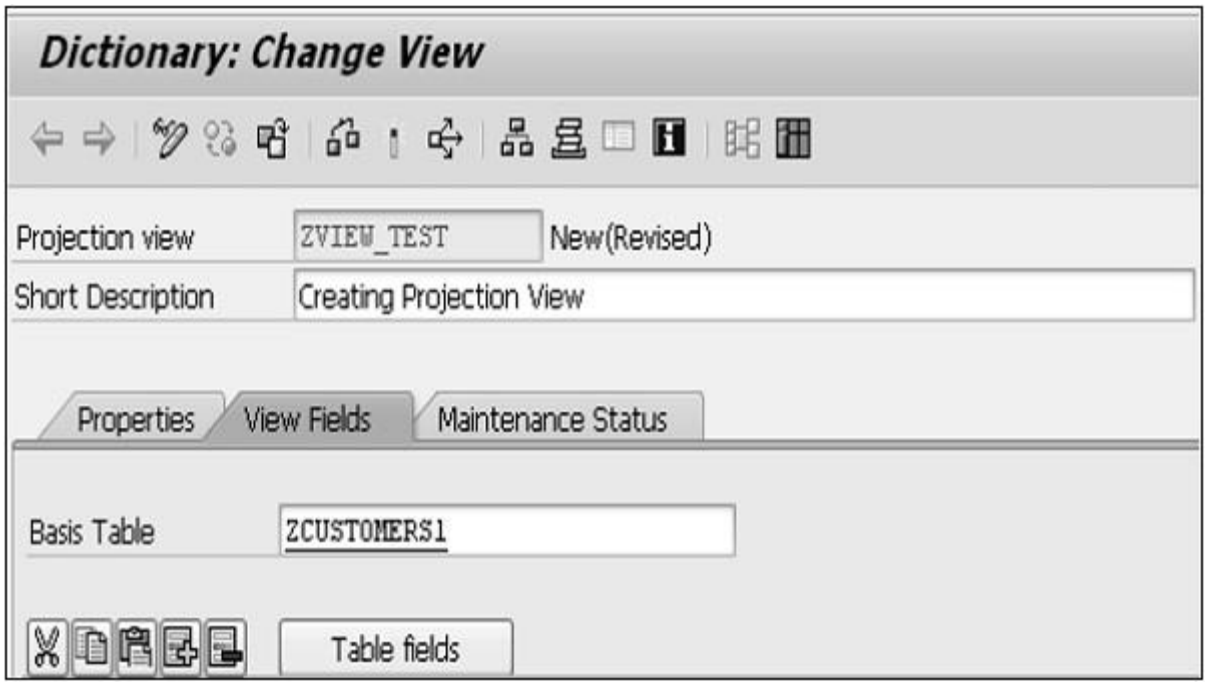
İstenmeyen alanları maskelemek ve bir tabloda yalnızca ilgili alanları görüntülemek için projeksiyon görünümünü kullanırız. Projeksiyon görünümleri tek bir şeffaf tablo üzerinden tanımlanmalıdır. Bir projeksiyon görünümü tam olarak bir tablo içerir. Projeksiyon görünümleri için seçim koşullarını tanımlayamıyoruz.

Görünüm Oluşturma

Adım 1 − ABAP Sözlüğü'nün ilk ekranında Görüntüle radyo düğmesini seçin. Oluşturulacak görünümün adını girin ve ardından Oluştur düğmesine tıklayın. Görünümün adını ZVIEW_TEST olarak girdik.

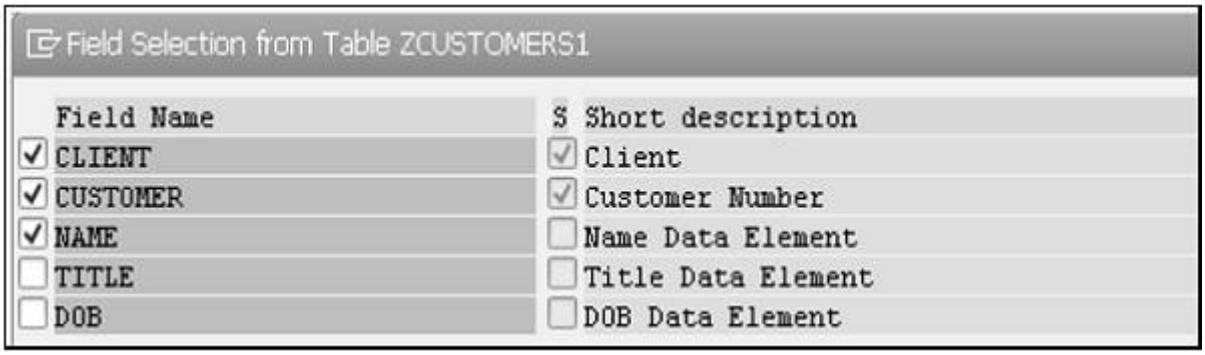
Adım 2 − Görünüm tipini seçerken projeksiyon görünümü radyo düğmesini seçin ve Kopyala düğmesine tıklayın. 'Sözlük: Görünümü Değiştir' ekranı görünür.

Adım 3 − Kısa Açıklama alanına kısa bir açıklama girin ve Temel Tablo alanında kullanılacak tablonun adını aşağıdaki anlık görüntüde gösterildiği gibi girin.

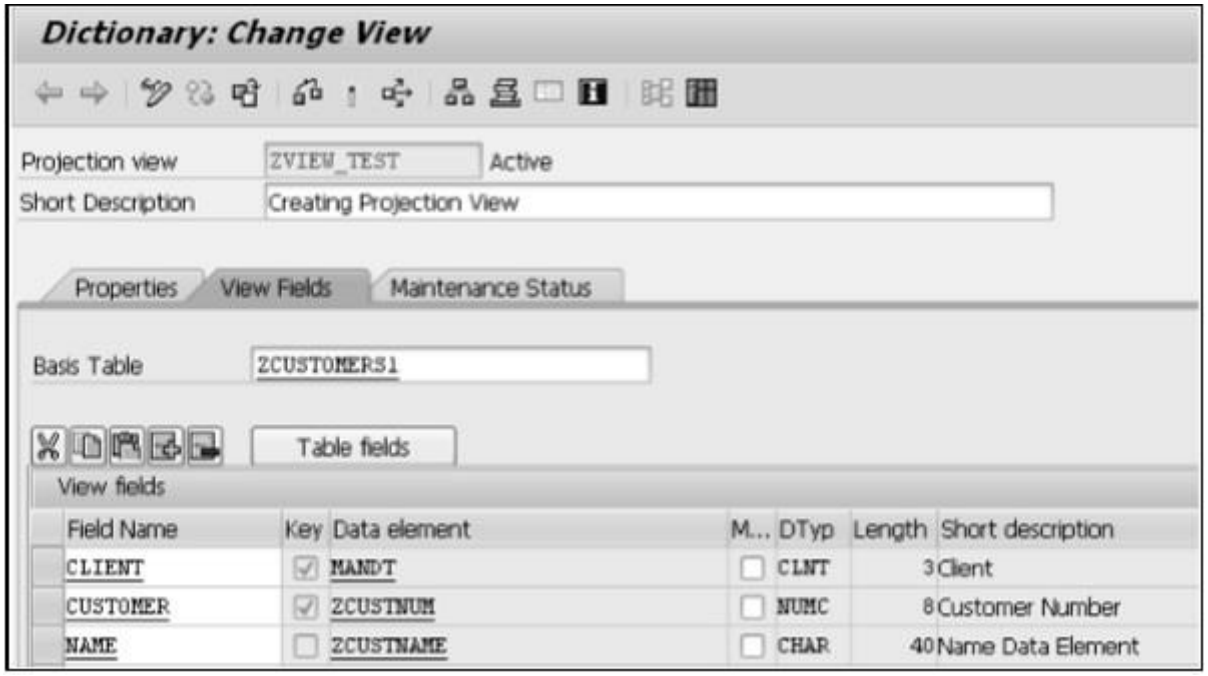


Adım 4 − ZCUSTOMERS1 tablosunun alanlarını projeksiyon görünümüne dahil etmek için 'Tablo alanları' düğmesini tıklayın.

Adım 5 − Tablo ZCUSTOMERS1'den Alan Seçimi ekranı görünür. Aşağıdaki anlık görüntüde gösterildiği gibi projeksiyon görünümüne dahil etmek istediğiniz alanları seçin.



Adım 6 − Kopyala düğmesine tıkladıktan sonra, projeksiyon görünümü için seçilen tüm alanlar 'Sözlük: Görünümü Değiştir' ekranında görüntülenir.



Adım 7 − Bir erişim yöntemi tanımlamak için Bakım Durumu sekmesini seçin. 'Veri Tarayıcı/Tablo Görünümü Bakımı' açılır menüsünden salt okunur radyo düğmesini ve 'Kısıtlamalarla İzin Verilen Görüntüleme/Bakım' seçeneğini seçin.

Adım 8 - Kaydedin ve Etkinleştirin. 'Sözlük: Görünümü Değiştir' ekranında, ZVIEW_TEST seçim ekranını görüntülemek için Utilities(M) > Contents'i seçin.

Adım 9 − Yürüt simgesine tıklayın. Projeksiyon görünümünün çıktısı, aşağıdaki ekran görüntüsünde gösterildiği gibi görünür.

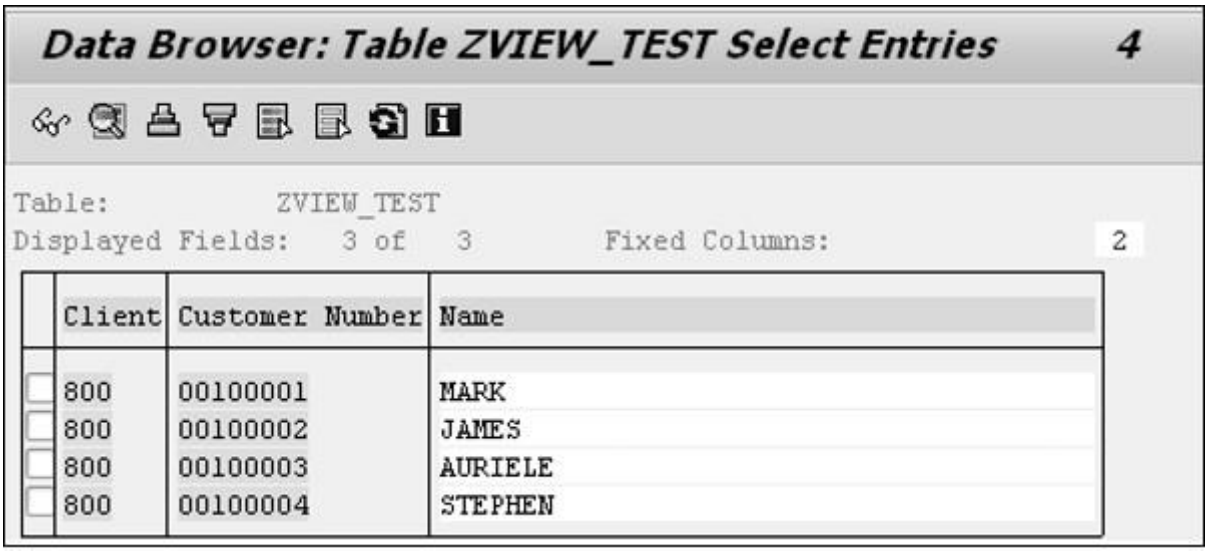

Data Browser: Table ZVIEW_TEST Select Entries 4

Table: ZVIEW_TEST
Displayed Fields: 3 of 3 Fixed Columns: 2

Client	Customer Number	Name
800	00100001	MARK
800	00100002	JAMES
800	00100003	AURIELE
800	00100004	STEPHEN

ZCUSTOMERS1 tablosu 5 alandan oluşur. Burada görüntülenen alanlar 4 girişli 3 (Müşteri, Müşteri Numarası ve Adı)'dir. Müşteri numaraları uygun isimlerle 100001'den 100004'e kadardır.

SAP ABAP - Arama Yardımı

ABAP Sözlüğü'nün diğer bir veri havuzu nesnesi olan Arama Yardımı, bir alan için olası tüm değerleri bir liste biçiminde görüntülemek için kullanılır. **Bu liste aynı zamanda bir isabet listesi** olarak da bilinir . Sıkıcı ve hataya açık olan değeri manuel olarak girmek yerine bu isabet listesinden alanlara girilecek değerleri seçebilirsiniz.

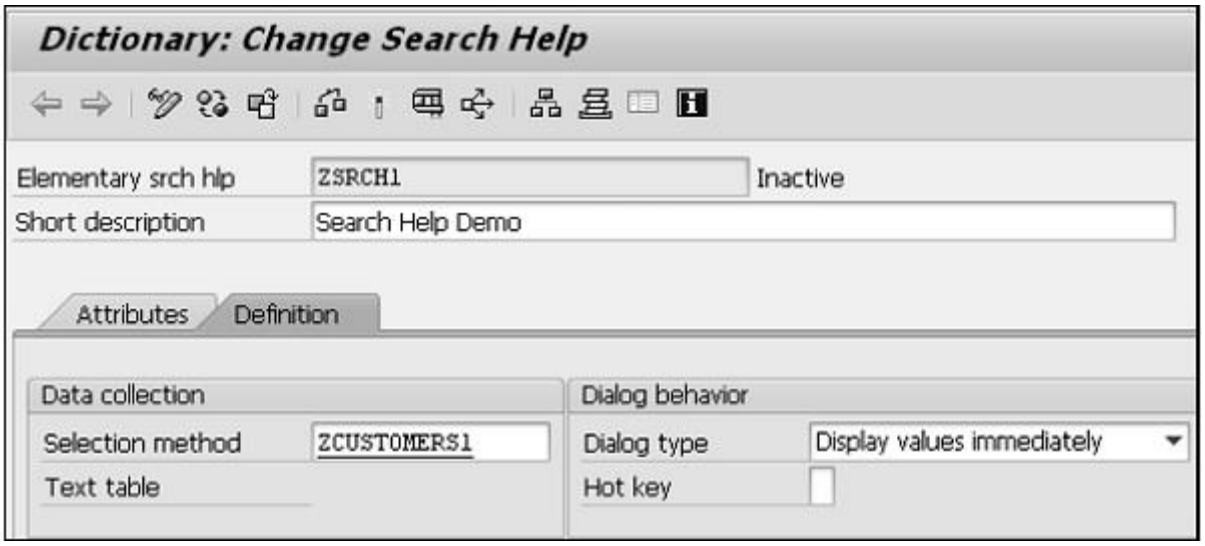
Arama Yardımı Oluşturma

Adım 1 − SE11 işlemine gidin. Arama yardımı için radyo düğmesini seçin. Oluşturulacak arama yardımının adını girin. ZSRCH1 adını girelim. Oluştur düğmesine tıklayın.

Adım 2 − Sistem, oluşturulacak arama yardım türünü soracaktır. Varsayılan olan Temel arama yardımını seçin. Aşağıdaki ekran görüntüsünde gösterildiği gibi temel arama yardımı oluşturma ekranı görüntülenir.

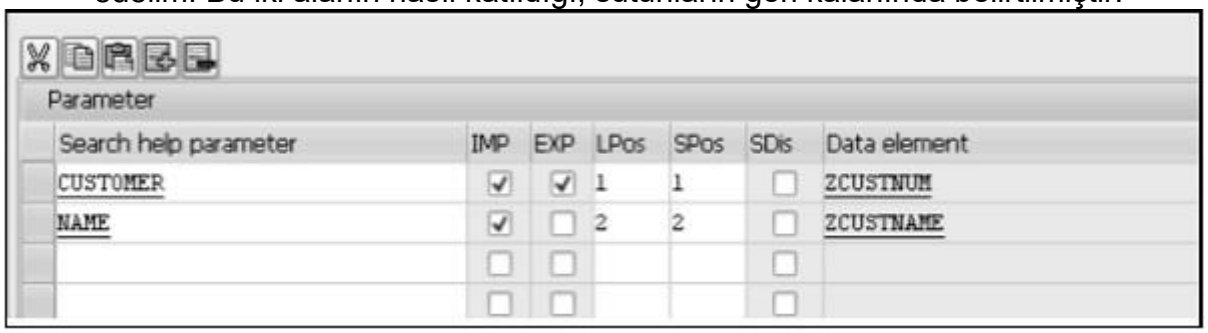
Adım 3 − Seçim yönteminde veri kaynağımızın tablo mu yoksa view mi olduğunu belirtmemiz gerekiyor. Bizim durumumuzda bir tablo olur. Tablo ZCUSTOMERS1'dir. Bir seçim listesinden seçilir.

Adım 4 − Seçim yöntemi girildikten sonra sonraki alan Diyalog tipidir. Bu, kısıtlayıcı iletişim kutusunun görünümünü kontrol eder. Üç seçenekli bir açılır liste var. 'Değerleri hemen göster' seçeneğini seçelim.



Adım 5 − Sonraki parametre alanıdır. Her Arama yardım parametresi veya alanı için, gereksinimlere göre bu sütun alanları girilmelidir.

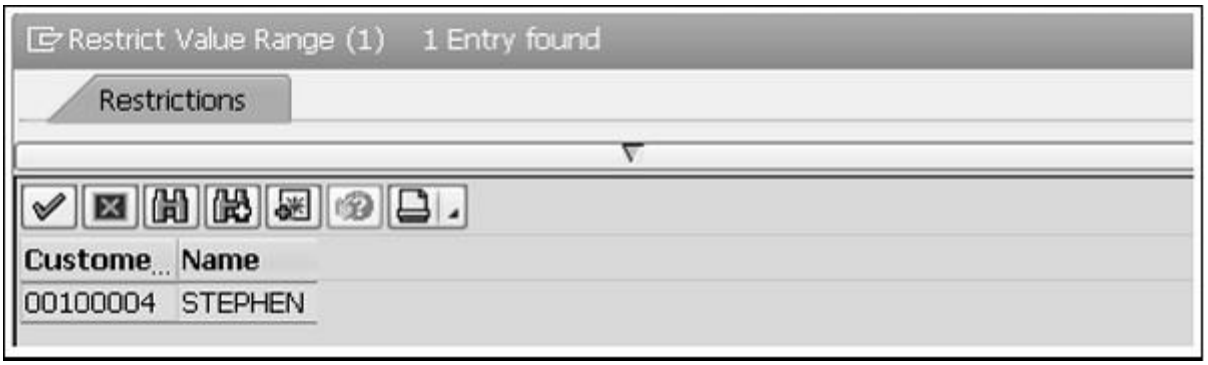
- **Arama yardım parametresi** − Bu, veri kaynağından bir alandır. Tablodaki alanlar seçim listesinde listelenir. Arama yardımına katılan alanlar, her satırda bir alan olacak şekilde girilecektir. CUSTOMER ve NAME alanlarını dahil edelim. Bu iki alanın nasıl katıldığı, sütunların geri kalanında belirtilmiştir.



- **İçe Aktar** − Bu alan, Arama yardım parametresinin bir içe aktarma parametresi olup olmadığını gösteren bir onay kutusudur. Dışa aktarma veya içe aktarma, arama yardımına atıfta bulunur.
- **Dışa Aktar** − Bu alan, Arama yardımı parametresinin bir dışa aktarma parametresi olup olmadığını gösteren bir onay kutusudur. Dışa aktarma, seçim listesindeki alan değerlerinin ekran alanlarına aktarılması olacaktır.
- **LPos** − Değeri, seçim listesindeki Arama yardım parametresinin veya alanının fiziksel konumunu kontrol eder. 1 değeri girerseniz, alan seçim listesinde ilk konumda görünür ve bu şekilde devam eder.
- **SPos** − Kısıtlayıcı iletişim kutusunda Arama Yardımı parametresinin veya alanının fiziksel konumunu kontrol eder. 1 değerini girerseniz, alan kısıtlayıcı iletişim kutusunda ilk konumda görünür ve bu şekilde devam eder.
- **Veri öğesi** − Varsayılan olarak her Arama Yardımı parametresine veya alanına, veri kaynağında (Tablo veya Görünüm) atanan bir veri öğesi atanır. Bu veri öğesi adı, görüntüleme modunda görünür.

Adım 6 − Bir tutarlılık kontrolü yapın ve arama yardımını etkinleştirin. Yürütmek için F8 tuşuna basın. Aşağıdaki ekran görüntüsünde gösterildiği gibi 'Test Arama Yardımı ZSRCH1' ekranı görüntülenir.

Adım 7 − MÜŞTERİ'nin 'Giriş için hazır' ekran alanına 100004 sayısını girelim. Enter tuşuna basın.



100004 müşteri numarası ve 'STEPHEN' adı görüntülenir.

SAP ABAP - Nesneleri Kilitle

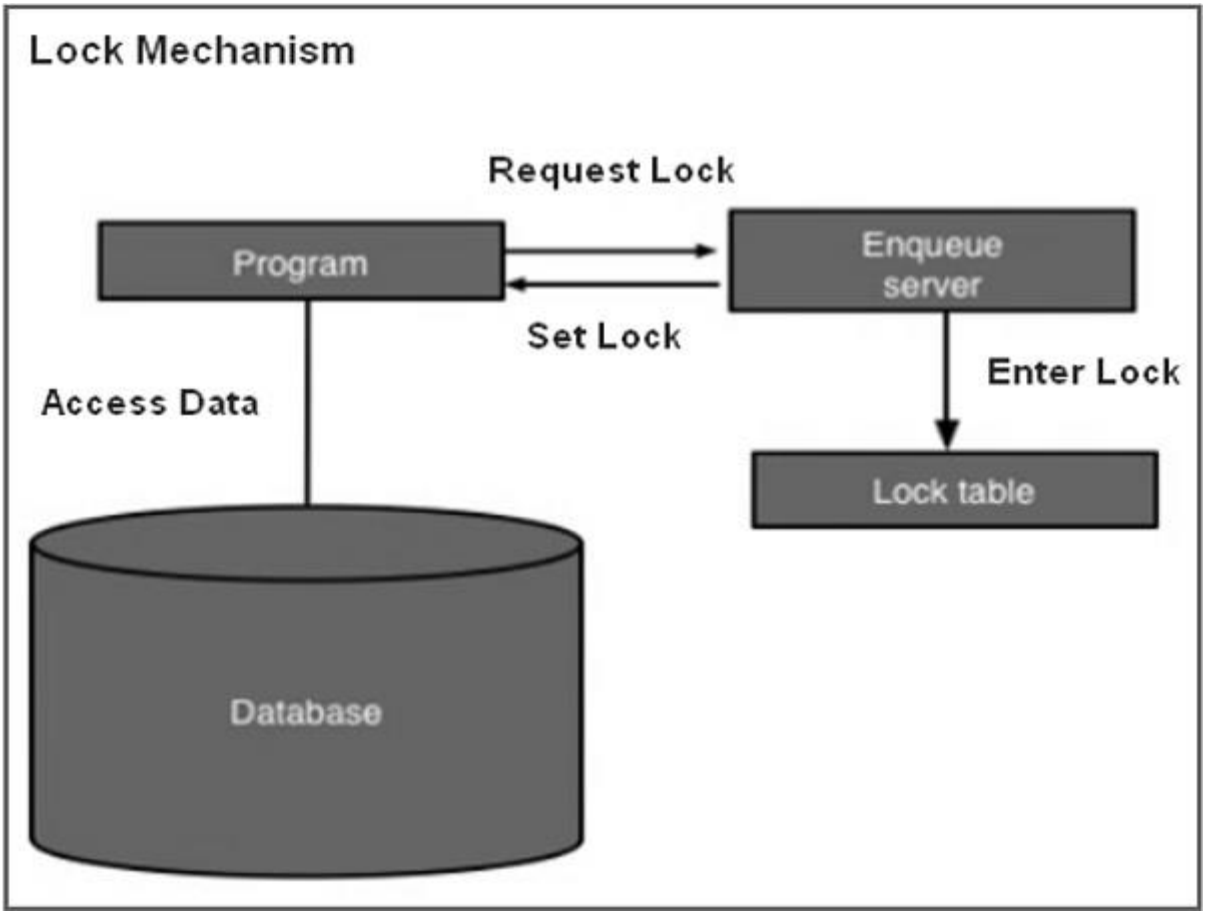
Nesneyi Kilitle, ABAP Sözlüğü'nün sunduğu ve aynı verilere birden fazla programın erişimini senkronize etmek için kullanılan bir özelliktir. Veri kayıtlarına belirli programlar yardımıyla ulaşılır. SAP'de, veriler veritabanına eklenirken veya veritabanında değiştirilirken tutarsızlığı önlemek için kilit nesneleri kullanılır. Veri kayıtları kilitlenecek olan tablolar, anahtar alanlarıyla birlikte bir Kilit Nesnesinde tanımlanmalıdır.

Kilit Mekanizması

Kilit mekanizması ile gerçekleştirilen iki ana işlev aşağıdadır -

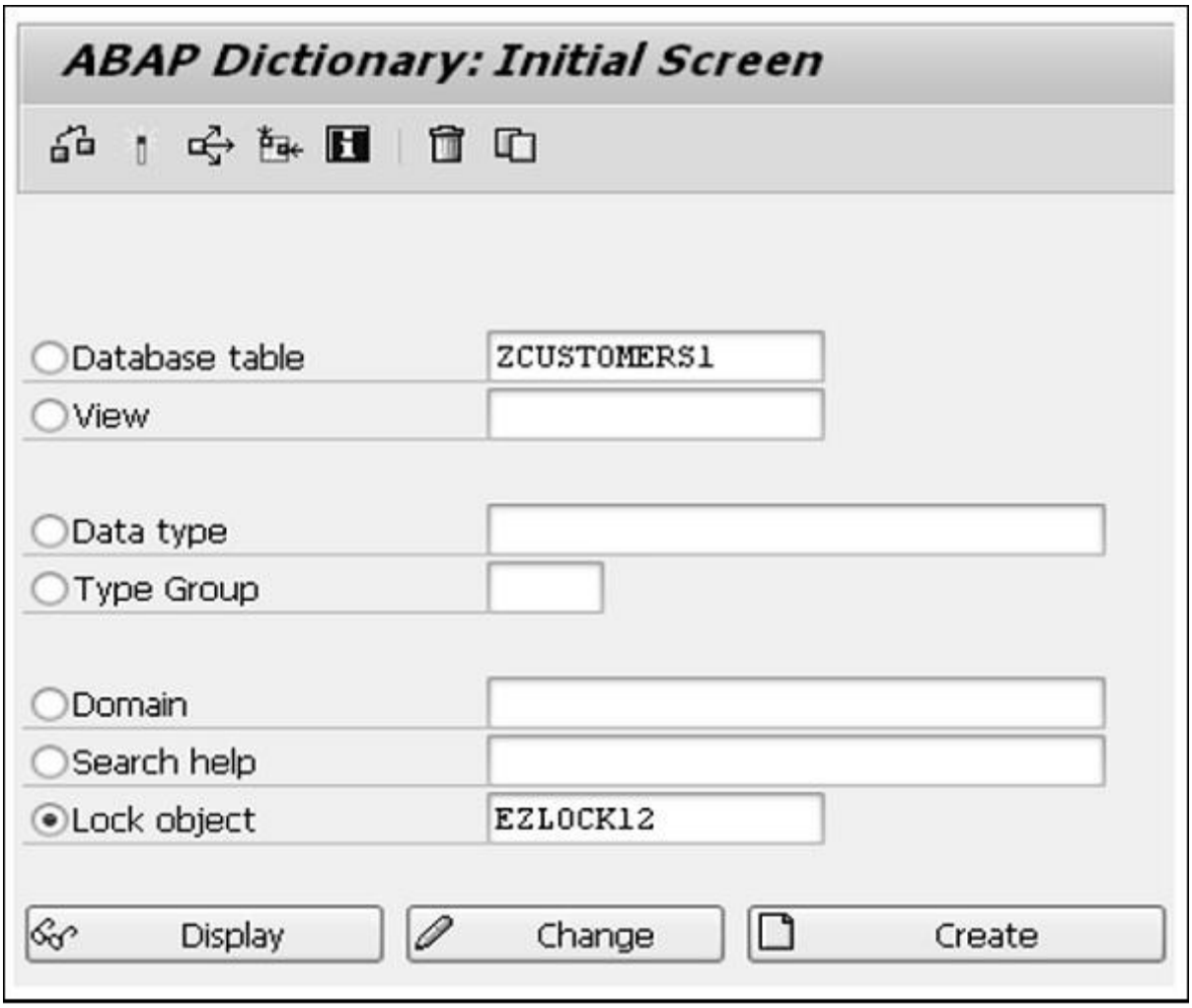
- Bir program, sadece okuduğu veya değiştirdiği veri kayıtları hakkında diğer programlarla iletişim kurabilir.
- Bir program, başka bir program tarafından yeni değiştirilen verileri okumasını engelleyebilir.

İlk olarak program tarafından bir **kilit talebi** oluşturulur. Daha sonra bu istek Enqueue sunucusuna gider ve kilit tablosunda kilit oluşturulur. Enqueue sunucusu kilidi ayarlar ve program nihayet verilere erişmeye hazırdır.



Kilit Nesneleri Oluşturma

Adım 1 − SE11 işlemine gidin. Aşağıdaki ekran açılır.

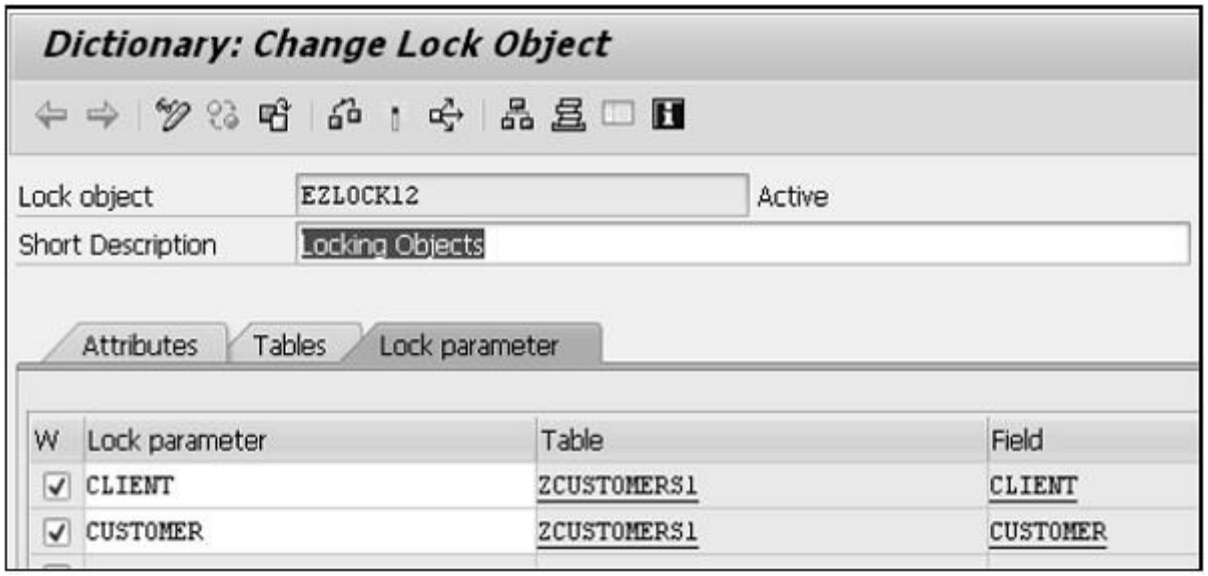


Adım 2 − 'Nesneyi Kilitle' radyo düğmesine tıklayın. E ile başlayan kilit nesnesinin adını girin ve Oluştur düğmesine tıklayın. Burada EZLOCK12 kullanıyoruz.

Adım 3 − Kısa açıklama alanına girin ve Tablolar sekmesine tıklayın.

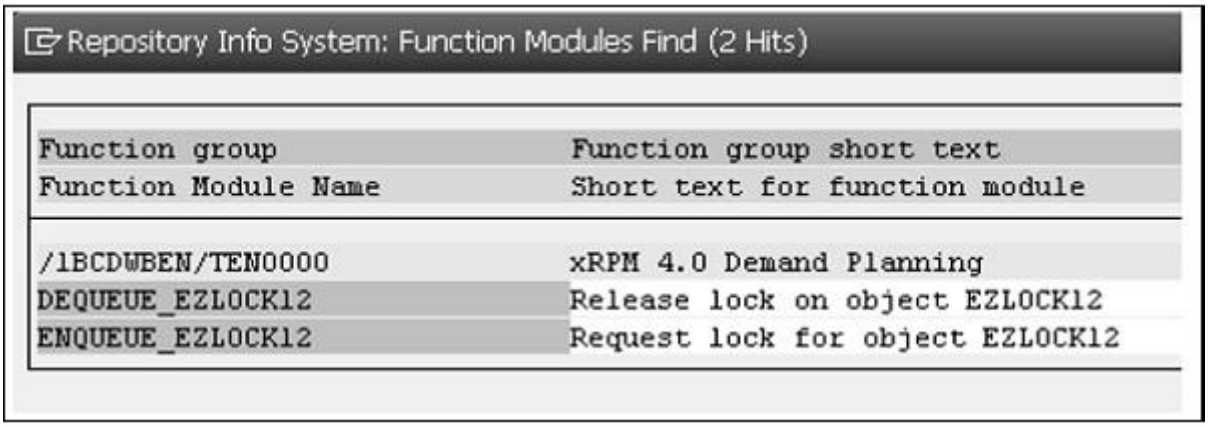
Adım 4 − Ad alanına tablo adını girin ve Kilit modunu Yazma Kilidi olarak seçin.

Adım 5 − Parametreyi kilitle sekmesine tıklayın, aşağıdaki ekran görünecektir.



Adım 6 − Kaydedin ve etkinleştirin. Otomatik olarak 2 fonksiyon modülü üretecektir. Fonksiyon modüllerini kontrol etmek için Git → Modülleri Kilitle'yi kullanabiliriz.

Adım 7 − Modülleri Kilitle'ye tıklayın, aşağıdaki ekran açılacaktır.



Kilit nesnesi başarıyla oluşturuldu.

Bir Kilit Nesnesinde bulunan bir tablonun anahtar alanlarına kilit argümanları denir ve bunlar fonksiyon modüllerinde giriş parametreleri olarak kullanılır. Bu bağımsız değişkenler, Kilit Nesnesi tanımı tarafından oluşturulan kilitleri ayarlamak ve kaldırmak için kullanılır.

SAP ABAP - Modülerleştirme

Programlarınızı mümkün olduğunca bağımsız ve okunması kolay tutmak iyi bir uygulamadır. Her bir görevi, geliştiricinin başka dikkat dağıtıcı olmadan konsantre olabileceği kendi modülüne yerleştirerek, büyük ve karmaşık görevleri daha küçük ve daha basit olanlara bölmeye çalışın.

SAP ABAP ortamında modülerleştirme, programların mantıksal bloklar olarak da bilinen modüler birimler halinde düzenlenmesini içerir . Yedeklemeyi azaltır ve siz onu oluştururken ve daha sonra bakım döngüsü sırasında bile program okunabilirliğini artırır. Modülerleştirme, aynı kodun tekrar kullanılabilirliğini de

sağlar. ABAP, geliştiricilerin modülerleştirmesini, yani programları nispeten daha yerleşik modüler özelliklere sahip OOPS tabanlı dillere göre nispeten daha fazla organize etmesini gerekli kılmıştır. Küçük, modülerleştirilmiş bir kod bölümü tamamlandıktan, hata ayıklandıktan ve benzerlerinden sonra, daha sonra geri döndürülmesi gerekmez ve geliştiriciler daha sonra devam edebilir ve diğer konulara odaklanabilir.

ABAP programları, modülerleştirme işlem blokları olarak bilinen işlem bloklarından oluşur. onlar -

- Programın dışından ve ABAP çalışma zamanı ortamından çağrılan işleme blokları (yani, olay blokları ve diyalog modülleri).
- ABAP programlarından çağrılan blokları işleme.

İşlem blokları ile modülerleştirmenin yanı sıra, kaynak kodunuzu makrolar ve dahil etme programları aracılığıyla modülerleştirmek için kaynak kod modülleri kullanılır.

Kaynak kodu düzeyinde modülerleştirme -

- Yerel Makrolar
- Küresel Dahil Etme programları

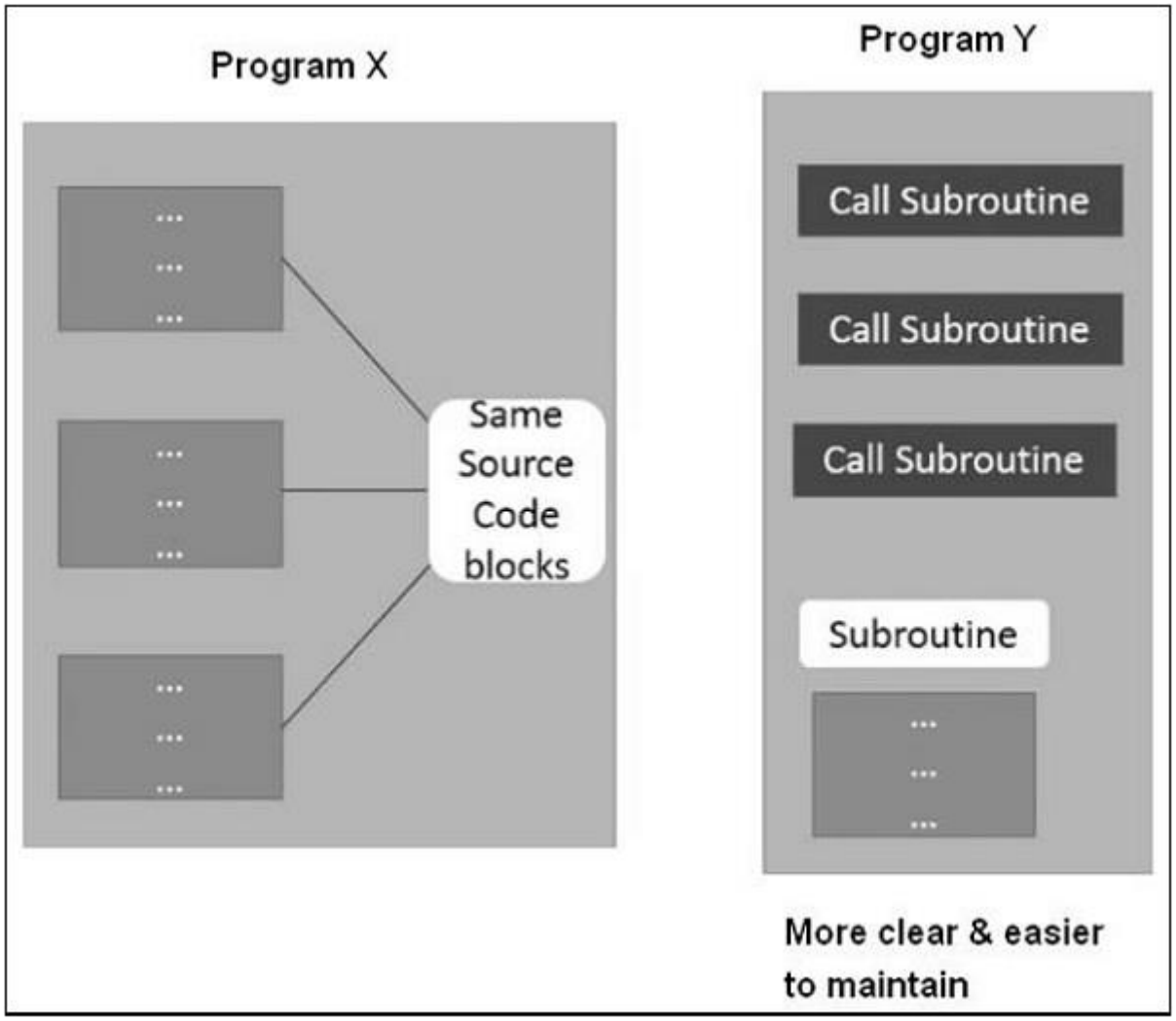
ABAP programlarından çağrılan işleme blokları aracılığıyla modülerleştirme -

- alt programlar
- Fonksiyon modülleri

Bir kaynak kodu modülerleştirmek, bir modüle bir dizi ABAP deyimi yerleştirmek anlamına gelir. Modülerleştirilmiş kaynak kodu, kullanıcının ihtiyacına göre bir programda çağrılabilir. Kaynak kodu modülleri, ABAP programlarının okunabilirliğini ve anlaşılırlığını artırır. Bireysel kaynak kodu modülleri oluşturmak aynı zamanda aynı ifadeleri tekrar tekrar yazmak zorunda kalmaktan da kaçınır ve bu da kodu ilk kez kullananlar için kodun anlaşılmasını kolaylaştırır.

SAP ABAP - Altyordamlar

Bir alt program, kodun yeniden kullanılabilir bir bölümüdür. Bir fonksiyonun kaynak kodu biçiminde kapsüllendiği program içindeki bir modülerleştirme birimidir. Ana programa daha iyi bir genel bakış elde etmek ve aşağıdaki şemada gösterildiği gibi karşılık gelen ifade dizisini birçok kez kullanmak için bir programın bir bölümünü bir alt programa aktarırsınız.



3 farklı **kaynak kod bloğuna** sahip X programımız var . Her blok aynı ABAP ifadelerine sahiptir. Temel olarak, bunlar aynı kod bloklarıdır. Bu kodun bakımını kolaylaştırmak için kodu bir alt programa yerleştirebiliriz. Bu alt programı istediğimiz kadar programlarımızda çağırabiliriz. Form ve EndForm deyimleri kullanılarak bir alt program tanımlanabilir.

Aşağıda bir altyordam tanımının genel sözdizimi verilmiştir.

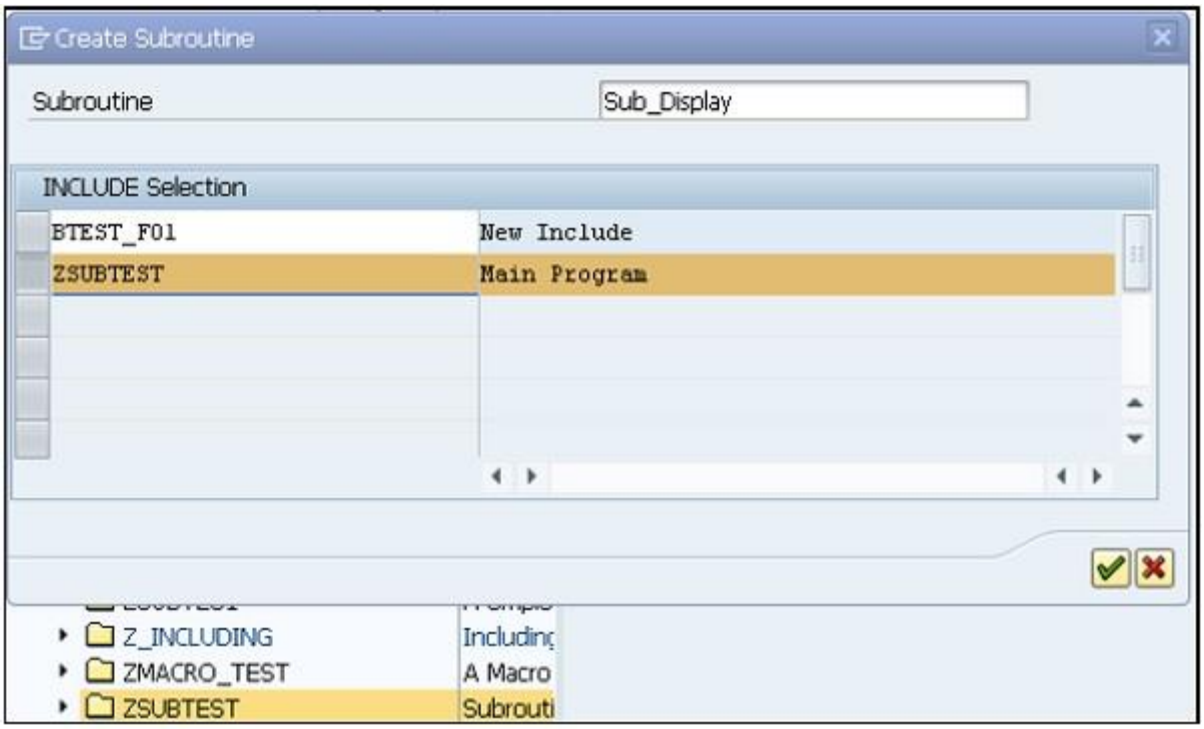
```
FORM <subroutine_name>.

<statements>

ENDFORM.
```

PERFORM deyimini kullanarak bir alt program çağırabiliriz. Denetim, <altyordam_adı> altyordamındaki ilk yürütülebilir ifadeye atlar. ENDFORM ile karşılaşıldığında, kontrol, PERFORM deyimini izleyen deyime geri döner.

Örnek

Adım 1 − SE80 işlemine gidin. Mevcut programı açın ve ardından programa sağ tıklayın. Bu durumda, 'ZSUBTEST' olur.

Adım 2 − Oluştur'u seçin ve ardından Altyordam'ı seçin. Alt program adını alana yazın ve ardından devam düğmesine tıklayın. Alt program adı, aşağıdaki ekran görüntüsünde gösterildiği gibi 'Sub_Display'dir.



Adım 3 − Kodu FORM ve ENDFORM deyim bloğuna yazın. Alt program başarıyla oluşturuldu.

Altyordamı çağırmak için PERFORM deyimini eklememiz gerekiyor. Şimdi koda bir göz atalım -

```
REPORT ZSUBTEST.
PERFORM Sub_Display.

* Form Sub_Display
* -->  p1 text
* <--  p2 text

FORM Sub_Display.
Write: 'This is Subroutine'.
Write: / 'Subroutine created successfully'.
ENDFORM.                    " Sub_Display
```

Adım 4 − Programı kaydedin, etkinleştirin ve çalıştırın. Yukarıdaki kod aşağıdaki çıktıyı üretir -

```
Subroutine Test:

This is Subroutine

Subroutine created successfully
```

Bu nedenle, alt programların kullanılması, programınızı daha işlev odaklı hale getirir. Programın görevini alt fonksiyonlara böler, böylece her bir alt program bir alt

fonksiyondan sorumludur. İşlevlerdeki değişikliklerin genellikle yalnızca alt programda uygulanması gerektiğinden programınızın bakımı daha kolay hale gelir.

SAP ABAP - Makrolar

Aynı ifade setini bir programda birden fazla kullanmak istiyorsak, bunları bir makroya dahil etmemiz gerekir. Örneğin, bir makro, uzun hesaplamalar veya karmaşık WRITE ifadeleri yazmak için yararlı olabilir. Bir makroyu yalnızca tanımlandığı bir program içinde kullanabiliriz. Makro tanımı, programda makro kullanılmadan önce yapılmalıdır.

Makrolar, yer tutuculara dayalı olarak tasarlanmıştır. Yer tutucu, C dilinde işaretçiler gibi çalışır. DEFINE...END-OF-DEFINITION deyimi içinde bir makro tanımlayabilirsiniz.

Bir makro tanımının temel sözdizimi aşağıdadır -

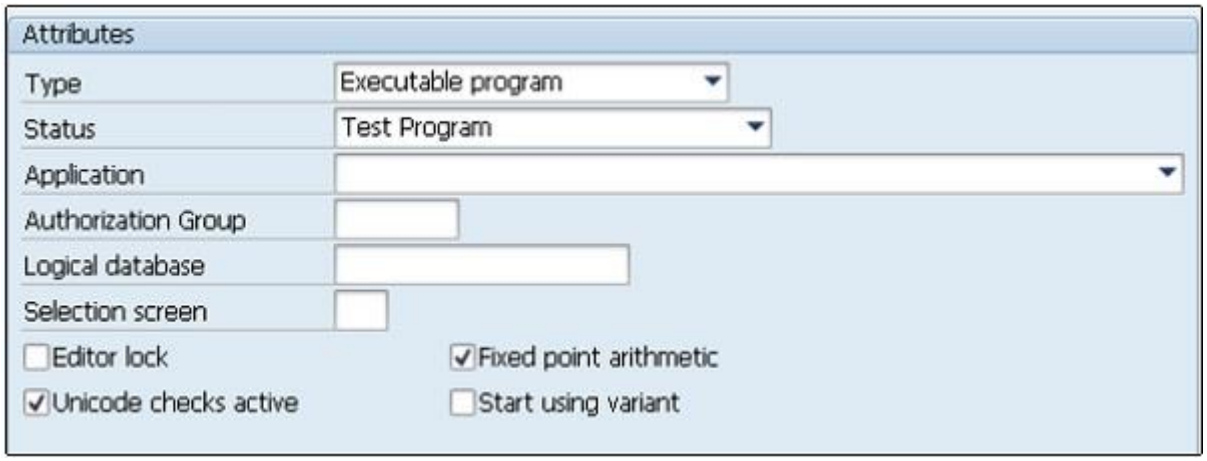
```
DEFINE <macro_name>. <statements>
END-OF-DEFINITION.
.....

<macro_name> [<param1> <param2>....].
```

Bir makroyu çağırmadan önce tanımlamanız gerekir. <param1>…. makro tanımında yer alan ABAP deyimlerindeki &1... yer tutucularının yerini alır.

Bir makro tanımındaki maksimum yer tutucu sayısı dokuzdur. Yani, bir program yürütüldüğünde, SAP sistemi makroyu uygun ifadelerle değiştirir ve &1, &2,….&9 yer tutucuları param1, param2,....param9 parametreleriyle değiştirilir. Bir makroyu başka bir makro içinde çağırabiliriz, ancak aynı makroyu değil.

Örnek

SE38 işlemine gidin. Kısa metin alanındaki açıklama ve ayrıca aşağıdaki ekran görüntüsünde gösterildiği gibi Tür ve Durum gibi uygun niteliklerle birlikte yeni bir ZMACRO_TEST programı oluşturun -



Kod aşağıdadır -

```
REPORT ZMACRO_TEST.
DEFINE mac_test.
```

```
WRITE: 'This is Macro &1'.
END-OF-DEFINITION.

PARAMETERS: s1 type C as checkbox.
PARAMETERS: s2 type C as checkbox.
PARAMETERS: s3 type C as checkbox default 'X'.

START-OF-SELECTION.
IF s1 = 'X'.
  mac_test 1. ENDIF.
IF s2 = 'X'.
  mac_test 2.
ENDIF.

IF s3 = 'X'.
  mac_test 3.
ENDIF.
```

3 onay kutumuz var. Programı çalıştırırken S2 onay kutusunu seçelim.

Yukarıdaki kod aşağıdaki çıktıyı üretir -

A Macro Program

This is Macro 2

Tüm onay kutuları seçiliyse, kod aşağıdaki çıktıyı üretir -

A Macro Program

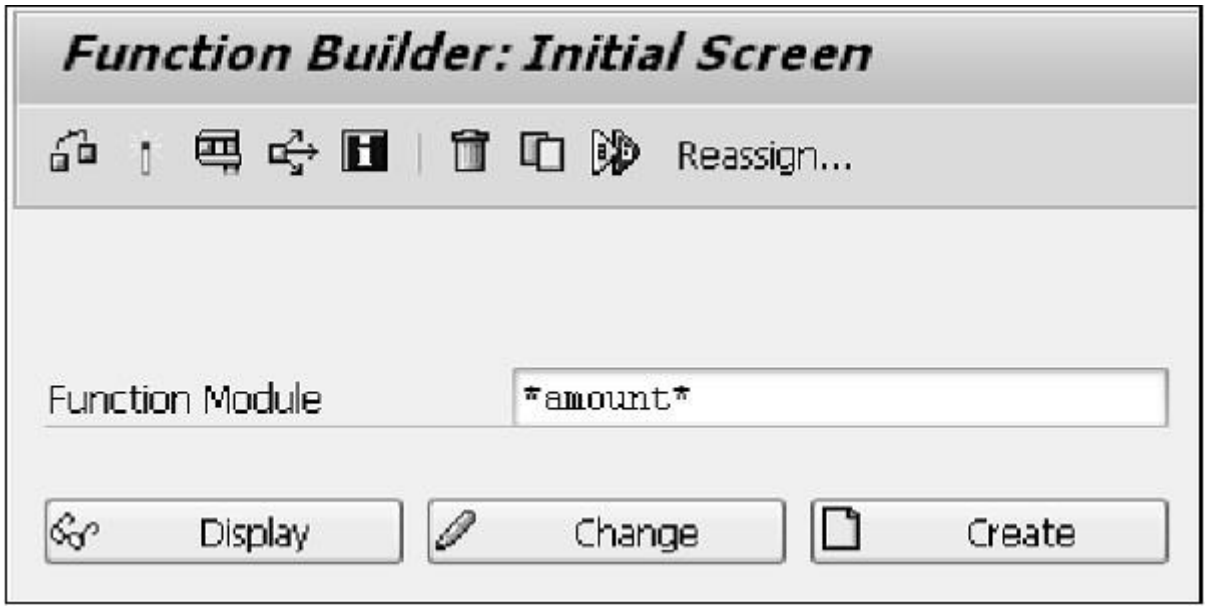
This is Macro 1 This is Macro 2 This is Macro 3

SAP ABAP - İşlev Modülleri

İşlev modülleri, SAP sisteminin önemli bir bölümünü oluşturur, çünkü SAP yıllardır fonksiyon modüllerini kullanarak kodu modüler hale getirerek kodun kendileri, geliştiricileri ve ayrıca müşterileri tarafından yeniden kullanılmasına izin verir.

İşlev modülleri, parametreleri içe ve dışa aktaran bir dizi yeniden kullanılabilir ifade içeren alt programlardır. Include programlarının aksine, fonksiyon modülleri bağımsız olarak yürütülebilir. SAP sistemi, herhangi bir ABAP programından çağrılabilen önceden tanımlanmış birkaç fonksiyon modülü içerir. İşlev grubu, mantıksal olarak birbirine ait olacak bir dizi işlev modülü için bir tür kap görevi görür. Örneğin, bir İK bordro sistemi için fonksiyon modülleri bir fonksiyon grubunda bir araya getirilecektir.

İşlev modüllerinin nasıl oluşturulacağına bakmak için işlev oluşturucu araştırılmalıdır. İşlev oluşturucuyu SE37 işlem koduyla bulabilirsiniz. İşlev modüllerinin aranma şeklini göstermek için bir işlev modülü adının bir bölümünü joker karakterle yazmanız yeterlidir. *tutar* yazın ve ardından F4 tuşuna basın.



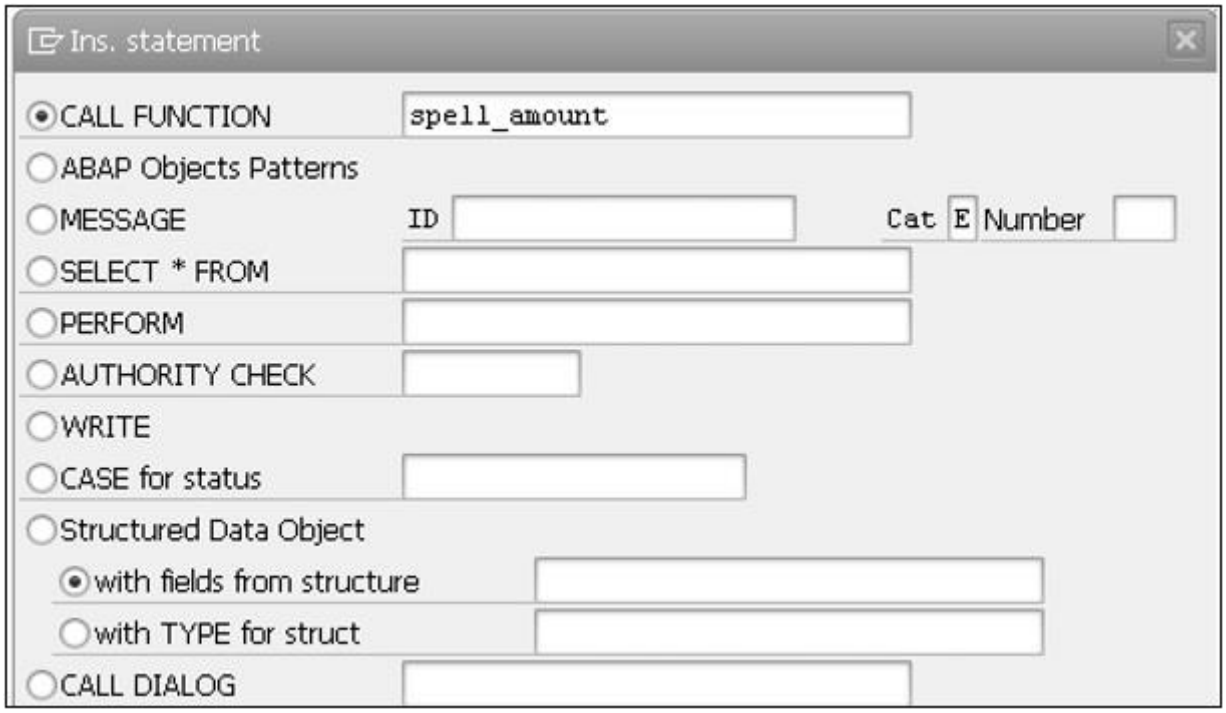
Aramanın sonuçları yeni bir pencerede görüntülenecektir. Fonksiyon modülleri mavi arka planlı satırlarda ve fonksiyon grupları pembe satırlarda gösterilir. Nesne Gezgini ekranını (İşlem SE80) kullanarak ISOC fonksiyon grubuna daha fazla bakabilirsiniz. İşlev modüllerinin bir listesini ve ayrıca işlev grubunda tutulan diğer nesneleri görebilirsiniz. SPELL_AMOUNT fonksiyon modülünü ele alalım. Bu fonksiyon modülü, sayısal rakamları kelimelere dönüştürür.

Yeni Program Oluşturma

Adım 1 − SE38 işlemine gidin ve Z_SPELLAMOUNT adlı yeni bir program oluşturun.

Adım 2 − Bir değerin girilebileceği ve fonksiyon modülüne aktarılabileceği bir parametrenin ayarlanabilmesi için bir kod girin. Buradaki text-001 metin öğesinde 'Bir Değer Girin' yazıyor.

Adım 3 − Bunun kodunu yazmak için CTRL+F6 tuşlarını kullanın. Bundan sonra, 'ÇAĞRI FONKSİYONU'nun listedeki ilk seçenek olduğu bir pencere belirir. Metin kutusuna 'spell_amount' yazın ve devam düğmesini tıklayın.



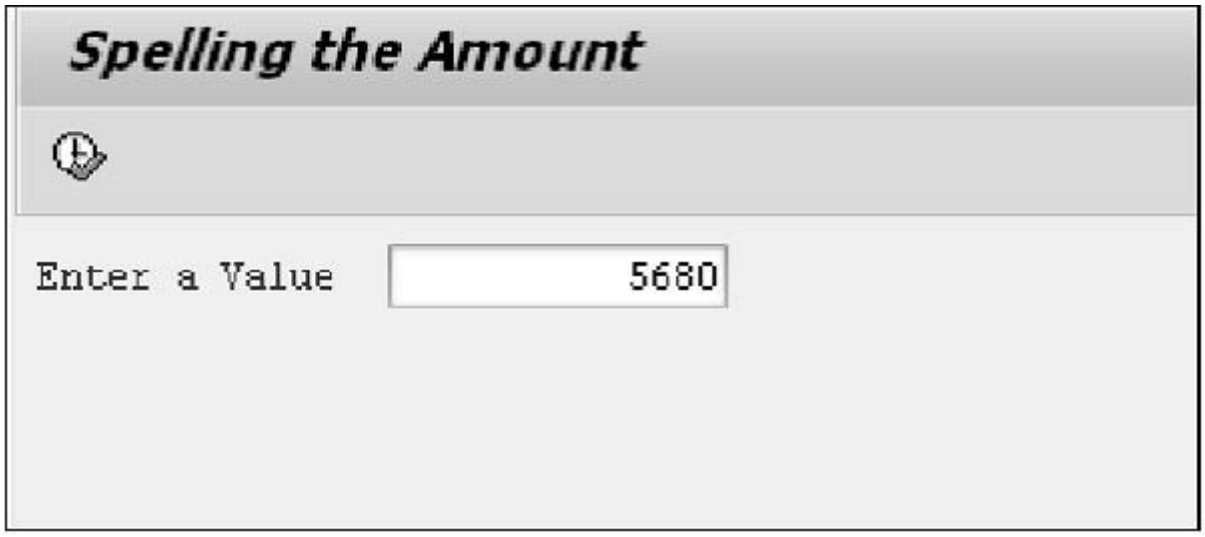
Adım 4 − Bazı kodlar otomatik olarak oluşturulur. Ancak IF ifadesini, "İşlev modülü bir değer döndürdü: sy-subrc" diyen ekrana bir mesaj YAZMAK için bir kod içerecek şekilde geliştirmemiz ve doğru sonucu yazmak için ELSE ifadesini eklememiz gerekiyor. modül başarılı.Burada fonksiyon modülünden dönen değeri tutacak yeni bir değişken ayarlanmalıdır.Buna 'sonuç' diyelim.

Kod aşağıdadır -

```
REPORT Z_SPELLAMOUNT.
data result like SPELL.

selection-screen begin of line.
selection-screen comment 1(15) text-001.

parameter num_1 Type I.
selection-screen end of line.
CALL FUNCTION 'SPELL_AMOUNT'
EXPORTING
AMOUNT = num_1
IMPORTING
IN_WORDS = result.

IF SY-SUBRC <> 0.
  Write: 'Value returned is:', SY-SUBRC.
else.
  Write: 'Amount in words is:', result-word.
ENDIF.
```

Adım 5 − İşlev modülünün döndürdüğü değişkene IN_WORDS adı verilir. 'sonuç' adlı programda karşılık gelen değişkeni ayarlayın. SPELL adlı bir yapıya başvurmak için LIKE deyimini kullanarak IN_WORDS tanımlayın.

Adım 6 − Programı kaydedin, etkinleştirin ve çalıştırın. Aşağıdaki ekran görüntüsünde gösterildiği gibi bir değer girin ve F8 tuşuna basın.



Yukarıdaki kod aşağıdaki çıktıyı üretir -

```
Spelling the Amount
Amount in words is:
FIVE THOUSAND SIX HUNDRED EIGHTY
```

SAP ABAP - Programları Dahil Et

Dahil etme programları, kaynak kodunu modüler hale getirmek için kullanılan genel depo nesneleridir. Aynı kaynak kodunu farklı programlarda kullanmanıza izin verirler. Include programları, karmaşık programları düzenli bir şekilde yönetmenize de olanak tanır. Bir include programını başka bir programda kullanmak için aşağıdaki sözdizimini kullanırız -

```
INCLUDE <program_name>.
```

INCLUDE deyimi, <program_name> include programının kaynak kodunu başka bir programa kopyalamakla aynı etkiye sahiptir. Dahil etme programı bağımsız olarak çalışamayacağından, diğer programların içine yerleştirilmelidir. Dahil etme programlarını da yuvalayabilirsiniz.

Aşağıdakiler, Include programları için kod yazarken birkaç kısıtlamadır -

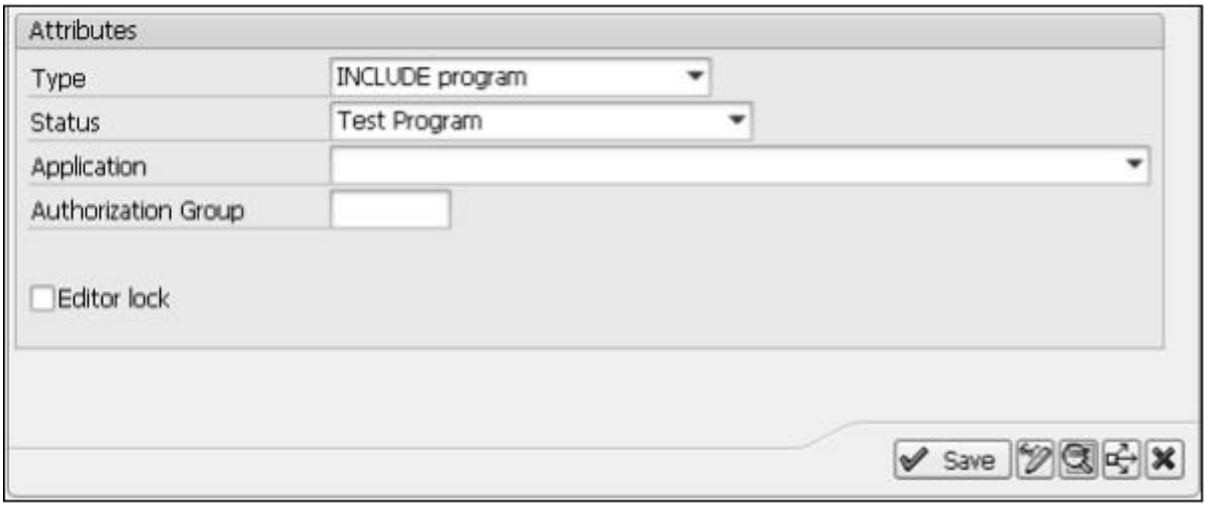
- Dahil etme programları kendilerini arayamaz.
- Dahil etme programları eksiksiz ifadeler içermelidir.

Bir Include programı oluşturma ve kullanma adımları aşağıdadır -

Adım 1 − ABAP Editor'a dahil edilecek programı (Z_TOBEINCLUDED) oluşturun. ABAP Editor'a dahil edilecek kod:

```
PROGRAM Z_TOBEINCLUDED.
Write: / 'This program is started by:', SY-UNAME,
       / 'The Date is:', SY-DATUM,
       / 'Time is', SY-UZEIT.
```

Adım 2 − Programın Türünü aşağıdaki ekran görüntüsünde gösterildiği gibi INCLUDE programı olarak ayarlayın.



Adım 3 − 'Kaydet' düğmesine tıklayın ve programı ZINCL_PCKG adlı bir pakete kaydedin.

Adım 4 − Z_TOBEINCLUDED programının kullanılması gereken başka bir program oluşturun. Burada Z_INCLUDINGTEST adında başka bir program oluşturduk ve programın tipini Executable program olarak atadık.

Adım 5 − Z_INCLUDINGTEST programı için kodlama, aşağıdaki kodda gösterildiği gibi INCLUDE ifadesi yardımıyla Z_TOBEINCLUDED programını içerir.

```
REPORT Z_INCLUDINGTEST.
INCLUDE Z_TOBEINCLUDED.
```

Adım 6 − Programı kaydedin, etkinleştirin ve çalıştırın.

Yukarıdaki kod aşağıdaki çıktıyı üretir -

```
This program is started by: SAPUSER
The Date is: 06.10.2015
Time is 13:25:11
```

SAP ABAP - SQL'e Genel Bakışı Aç

Open SQL, mevcut AS ABAP'ın merkezi veritabanındaki verilere doğrudan erişim sağlayan ABAP ifadelerinin alt kümesini belirtir. Açık SQL ifadeleri, SQL'in tüm veritabanı sistemleri tarafından desteklenen ABAP'taki Veri İşleme Dili işlevselliğini eşler.

Open SQL'in deyimleri, veritabanı arayüzünün Open SQL arayüzünde veritabanına özel SQL'e dönüştürülür. Daha sonra veritabanı sistemine aktarılır ve yürütülür. ABAP Sözlüğünde bildirilen veritabanı tablolarına erişmek için Open SQL deyimleri kullanılabilir. AS ABAP'ın merkezi veritabanına varsayılan olarak erişilir ve ayrıca ikincil veritabanı bağlantıları aracılığıyla diğer veritabanlarına erişim mümkündür.

Bu ifadelerden herhangi biri bir ABAP programında kullanıldığında, yürütülen eylemin başarılı olup olmadığını kontrol etmek önemlidir. Bir veritabanı tablosuna bir kayıt girmeye çalışırsa ve doğru şekilde eklenmemişse, programda uygun eylemin

yapılabilmesi için bilmek çok önemlidir. Bu, daha önce kullanılmış olan bir sistem alanı, yani SY-SUBRC kullanılarak yapılabilir. Bir ifade başarılı bir şekilde yürütüldüğünde, SY-SUBRC alanı 0 değerini içerecektir, bu nedenle bu kontrol edilebilir ve belirirse programa devam edilebilir.

DATA ifadesi, bir çalışma alanını bildirmek için kullanılır. Buna 'wa_customers1' adını verelim. Bunun için tek bir veri tipi bildirmek yerine, tabloyu oluşturan birkaç alan bildirilebilir. Bunu yapmanın en kolay yolu LIKE deyimini kullanmaktır.

INSERT İfadesi

wa_customers1 çalışma alanı burada ZCUSTOMERS1 tablosu GİBİ bildirilir ve bir tablo haline gelmeden aynı yapıyı alır. Bu çalışma alanı yalnızca bir kayıt saklayabilir. Bir kez bildirildiğinde, çalışma alanını ve tuttuğu kaydı tabloya eklemek için INSERT ifadesi kullanılabilir. Buradaki kod 'wa_customers1 FROM ZCUSTOMERS1' EKLE' olarak okunacaktır.

Çalışma alanı bazı verilerle doldurulmalıdır. ZCUSTOMERS1 tablosundaki alan adlarını kullanın. Bu, ileriye doğru gezinme, koddaki tablo adına çift tıklama veya yeni bir oturum açma ve SE11 işlemini kullanarak yapılabilir. Tablonun alanları daha sonra kopyalanıp ABAP düzenleyicisine yapıştırılabilir.

Kod parçacığı aşağıdadır -

```
DATA wa_customers1 LIKE ZCUSTOMERS1.
wa_customers1-customer = '100006'.
wa_customers1-name = 'DAVE'.
wa_customers1-title = 'MR'.
wa_customers1-dob = '19931017'.
INSERT ZCUSTOMERS1 FROM wa_customers1.
```

CHECK deyimi daha sonra aşağıdaki gibi kullanılabilir. Yani kayıt doğru girilmişse sistem bunu belirtecektir. Değilse, sıfıra eşit olmayacak SY-SUBRC kodu görüntülenecektir. Kod parçacığı aşağıdadır -

```
IF SY-SUBRC = 0.
  WRITE 'Record Inserted Successfully'.
ELSE.
  WRITE: 'The return code is ', SY-SUBRC.
ENDIF.
```

Programı kontrol edin, kaydedin, kodu etkinleştirin ve ardından test edin. Çıktı penceresi 'Kayıt Başarıyla Eklendi' olarak görüntülenmelidir.

Anlaşılır durum

CLEAR deyimi, bir alanın veya değişkenin yerine yeni verilerin eklenmesi için temizlenmesine ve yeniden kullanılmasına izin verir. CLEAR deyimi genellikle programlarda kullanılır ve mevcut alanların defalarca kullanılmasına izin verir.

Önceki kod parçacığında, ZCUSTOMERS1 tablosuna eklenecek yeni bir kayıt oluşturmak için çalışma alanı yapısı verilerle dolduruldu ve ardından bir doğrulama

kontrolü yapıldı. Yeni bir kayıt eklemek istiyorsak, daha sonra yeni verilerle tekrar doldurulabilmesi için CLEAR ifadesi kullanılmalıdır.

GÜNCELLEME Bildirimi

Bir tablodaki bir veya daha fazla mevcut kaydı aynı anda güncellemek istiyorsanız UPDATE deyimini kullanın. INSERT deyimine benzer şekilde, program yürütülürken kayda eklenen yeni verilerle doldurulan bir çalışma alanı bildirilir. INSERT deyimi ile daha önce oluşturulan kayıt burada güncellenecektir. NAME ve TITLE alanlarında saklanan metni düzenlemeniz yeterlidir. Daha sonra yeni bir satırda, INSERT deyimiyle aynı yapı kullanılır ve bu sefer aşağıdaki kod parçacığında gösterildiği gibi UPDATE deyimi kullanılır -

```
DATA wa_customers1 LIKE ZCUSTOMERS1.
wa_customers1-customer = '100006'.
wa_customers1-name = 'RICHARD'.
wa_customers1-title = 'MR'.
wa_customers1-dob = '19931017'.
UPDATE ZCUSTOMERS1 FROM wa_customers1.
```

UPDATE deyimi yürütülürken, kaydın başarıyla güncellendiğini görmek için ABAP Sözlüğündeki Veri Tarayıcısını görüntüleyebilirsiniz.

DEĞİŞTİR Bildirimi

MODIFY ifadesi, INSERT ve UPDATE ifadelerinin bir kombinasyonu olarak düşünülebilir. Yeni bir kayıt eklemek veya mevcut bir kaydı değiştirmek için kullanılabilir. Bir çalışma alanına girilen verilerden kaydın değiştirilmesinde önceki iki ifadeye benzer bir sözdizimi izler.

Bu ifade yürütüldüğünde, ilgili anahtar alanlar tablodakilere karşı kontrol edilecektir. Bu anahtar alan değerlerine sahip bir kayıt zaten mevcutsa, güncellenecektir. Değilse, yeni bir kayıt oluşturulacaktır.

Yeni bir kayıt oluşturmak için kod parçacığı aşağıdadır -

```
CLEAR wa_customers1.

DATA wa_customers1 LIKE ZCUSTOMERS1.
wa_customers1-customer = '100007'.
wa_customers1-name = 'RALPH'.
wa_customers1-title = 'MR'.
wa_customers1-dob = '19910921'.
MODIFY ZCUSTOMERS1 FROM wa_customers1.
```

Bu örnekte çalışma alanına yeni bir giriş yapılabilmesi için CLEAR deyimi kullanılmış ve ardından müşteri (numara) 100007 eklenmiştir. Bu yeni, benzersiz bir anahtar alan değeri olduğundan, yeni bir kayıt eklenecek ve başka bir doğrulama denetimi yürütülecektir.

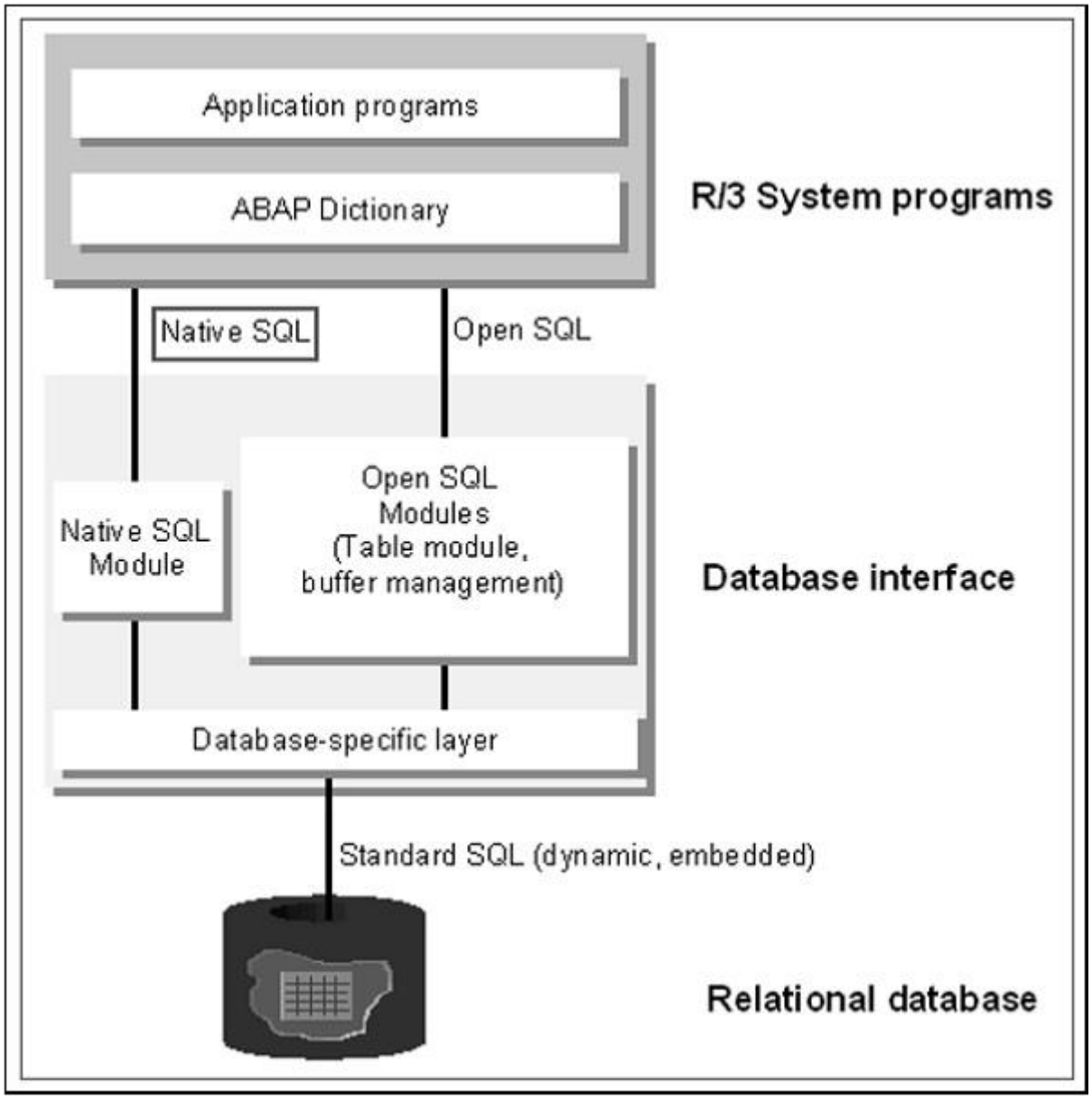
Bu yapıldığında ve Veri Tarayıcıda veriler görüntülendiğinde, 100007 (RALPH) müşteri numarası için yeni bir kayıt oluşturulmuş olacaktır.

Yukarıdaki kod aşağıdaki çıktıyı üretir (tablo içeriği) -

Client	Customer Number	Name	Title	DOB
800	00100001	MARK	MR	21.05.1981
800	00100002	JAMES	MR	14.08.1977
800	00100003	AURIELE	MS	19.06.1990
800	00100004	STEPHEN	MR	22.07.1985
800	00100005	MARGARET	MS	02.11.1994
800	00100006	RICHARD	MR	17.10.1993
800	00100007	RALPH	MR	21.09.1991

SAP ABAP - Yerel SQL'e Genel Bakış

'Yerel SQL' terimi, veritabanı arabiriminin Yerel SQL arabirimine statik olarak aktarılabilen tüm ifadeleri ifade eder. Yerel SQL ifadeleri ABAP'ın dil kapsamına girmez ve ABAP sözdizimini izlemez. ABAP, yalnızca Yerel SQL ifadelerinin listelenebileceği program bölümlerini izole etmek için ifadeler içerir.



Yerel SQL'de, esas olarak veritabanına özgü SQL ifadeleri kullanılabilir. Bunlar, yerel SQL arabiriminden bir veritabanı sistemine değişmeden aktarılır ve yürütülür. İlgili veritabanının tam SQL dil kapsamı kullanılabilir ve adreslenen veritabanı tablolarının ABAP Sözlüğü'nde bildirilmesi gerekmez. Ayrıca, yerel SQL arabirimi tarafından belirli bir şekilde işlenen küçük bir SAP'ye özgü Yerel SQL ifadesi seti de vardır.

Native SQL deyimi kullanmak için, EXEC SQL deyimi ile önüne geçmeniz ve ENDEXEC deyimi ile bitirmeniz gerekir.

Sözdizimi aşağıdadır -

```
EXEC SQL PERFORMING <form>.
<Native SQL statement>
ENDEXEC.
```

Bu ifadeler, bir ABAP programında bir veya daha fazla Yerel SQL ifadesinin listelenebileceği bir alanı tanımlar. Girilen ifadeler Yerel SQL arayüzüne iletilir ve ardından aşağıdaki gibi işlenir:

- Adreslenen veritabanı sisteminin program arayüzü için geçerli olan tüm SQL ifadeleri, özellikle DDL (veri tanımlama dili) ifadeleri olmak üzere EXEC ve ENDEXEC arasında listelenebilir.
- Bu SQL ifadeleri, Yerel SQL arabiriminden veritabanı sistemine büyük ölçüde değişmeden geçirilir. Sözdizimi kuralları, özellikle veritabanı nesneleri için büyük/küçük harf duyarlılığı kuralları olmak üzere, veritabanı sistemi tarafından belirlenir.
- Sözdizimi, bireysel ifadeler arasında bir ayırıcıya izin veriyorsa, EXEC ve ENDEXEC arasına birçok Yerel SQL ifadesi dahil edebilirsiniz.
- SAP'ye özel Yerel SQL dil öğeleri, EXEC ve ENDEXEC arasında belirtilebilir. Bu ifadeler doğrudan Yerel SQL arabiriminden veritabanına iletilmez, ancak uygun şekilde dönüştürülür.

Örnek

SPFLI, Uçuş planı bilgilerini depolamak için kullanılan standart bir SAP Tablosudur. Bu, sürüme ve sürüm düzeyine bağlı olarak R/3 SAP sistemlerinde mevcuttur. SE11 veya SE80 gibi ilgili SAP işlemine SPFLI Tablo adını girdiğinizde bu bilgileri görüntüleyebilirsiniz. Bu iki işlemi kullanarak da bu veritabanı tablosunda yer alan verileri görüntüleyebilirsiniz.

```
REPORT ZDEMONATIVE_SQL.
DATA: BEGIN OF wa,
    connid  TYPE SPFLI-connid,
    cityfrom TYPE SPFLI-cityfrom,
    cityto  TYPE SPFLI-cityto,
    END OF wa.

DATA c1 TYPE SPFLI-carrid VALUE 'LH'.
EXEC SQL PERFORMING loop_output.
  SELECT connid, cityfrom, cityto
  INTO :wa
  FROM SPFLI
  WHERE carrid = :c1
ENDEXEC.

FORM loop_output.
  WRITE: / wa-connid, wa-cityfrom, wa-cityto.
ENDFORM.
```

Yukarıdaki kod aşağıdaki çıktıyı üretir -

```
0400 FRANKFURT NEW YORK
2402 FRANKFURT BERLIN
0402 FRANKFURT NEW YORK
```

SAP ABAP - Dahili Tablolar

Dahili tablo aslında yürütülmekte olan bir ABAP programının kayıtlarını içeren geçici bir tablodur. Bir iç tablo, yalnızca bir SAP programının çalışma zamanı sırasında bulunur. ABAP dilini kullanarak büyük hacimli verileri işlemek için kullanılırlar. Veritabanı tablolarından veri almanız gerektiğinde, bir ABAP programında dahili bir tablo bildirmemiz gerekir.

Dahili bir tablodaki veriler, satırlar ve sütunlar halinde depolanır. Her satıra **satır** ve her sütuna alan adı **verilir** . Dahili bir tabloda tüm kayıtlar aynı yapıya ve anahtara sahiptir. Dahili bir tablonun bireysel kayıtlarına bir indeks veya bir anahtar ile erişilir. İlişkili program yürütülene kadar dahili tablo var olduğundan, programın yürütülmesi sonlandırıldığında dahili tablonun kayıtları atılır. Böylece dahili tablolar, verilerin gerektiği gibi değiştirilebileceği geçici depolama alanları veya geçici arabellekler olarak kullanılabilir. Bu tablolar, bildirimleri sırasında değil, yalnızca çalışma zamanında belleği işgal eder.

Dahili tablolar yalnızca bir program çalışırken mevcuttur, bu nedenle kod yazıldığında, dahili tablo, programın onu kullanabileceği şekilde yapılandırılmalıdır. Dahili tabloların yapılarla aynı şekilde çalıştığını göreceksiniz. Ana fark, yapıların yalnızca bir satıra sahip olması, dahili bir tablonun ise gerektiği kadar satıra sahip olabilmesidir.

Bir iç tablo, bir tablonun sütunlarına karşılık gelen bir dizi alandan oluşabilir, tıpkı ABAP sözlüğünde bir tablonun birkaç alan kullanılarak oluşturulması gibi. Anahtar alanlar dahili tablolarla da kullanılabilir ve bu dahili tabloları oluştururken biraz daha fazla esneklik sunarlar. Dahili tablolarla, benzersiz olmayan bir anahtar belirtilebilir, böylece herhangi bir sayıda benzersiz olmayan kaydın saklanmasına izin verilir ve gerekirse yinelenen kayıtların saklanmasına izin verilir.

Dahili bir tablonun boyutu veya içerdiği satır sayısı sabit değildir. Dahili tablonun boyutu, dahili tabloyla ilişkili programın gereksinimine göre değişir. Ancak dahili tabloların mümkün olduğunca küçük tutulması önerilir. Bu, muazzam miktarda veriyi işlemek için mücadele ederken sistemin yavaş çalışmasını önlemek içindir.

Dahili tablolar birçok amaç için kullanılır -

- Programda daha sonra kullanılabilecek hesaplama sonuçlarını tutmak için kullanılabilirler.
- Dahili bir tablo ayrıca kayıtları ve verileri tutabilir, böylece bu verilere veritabanı tablolarından erişmek zorunda kalmadan hızlı bir şekilde erişilebilir.
- Çok yönlüdürler. Herhangi bir sayıda başka tanımlanmış yapı kullanılarak tanımlanabilirler.

Örnek

Bir kullanıcının bir veya birkaç büyük tablodan çeşitli müşterilerin irtibat numaralarının bir listesini oluşturmak istediğini varsayalım. Kullanıcı önce dahili bir tablo oluşturur, müşteri tablolarından ilgili verileri seçer ve ardından verileri dahili tabloya yerleştirir. Diğer kullanıcılar, programın çalışma süresi boyunca her işlemi gerçekleştirmek için veritabanı sorguları yazmak yerine, istenen bilgileri almak için bu dahili tabloya doğrudan erişebilir ve kullanabilir.

SAP ABAP - Dahili Tablolar Oluşturma

DATA deyimi, dahili bir tabloyu bildirmek için kullanılır. Programa tablonun nerede başlayıp nerede bittiği söylenmelidir. Bu yüzden BEGIN OF deyimini kullanın ve ardından tablo adını bildirin. Bundan sonra, OCCURS eklemesi kullanılır ve ardından burada 0 olan bir sayı gelir. OCCURS, SAP'ye dahili bir tablonun oluşturulmakta

olduğunu söyler ve 0, başlangıçta herhangi bir kayıt içermeyeceğini belirtir. Daha sonra verilerle doldukça genişleyecektir.

Sözdizimi aşağıdadır -

```
DATA: BEGIN OF <internal_tab> Occurs 0,
```

Alanları yeni bir satırda oluşturalım. Örneğin, LIKE ZCUSTOMERS1-name olarak bildirilen 'name' oluşturun. ZCUSTOMERS1-dob GİBİ 'dob' adında başka bir alan oluşturun. Başlangıçta, dahili tablolardaki alan adlarına, başka bir yerde oluşturulmuş diğer alanlarla aynı adları vermek yararlıdır. Son olarak, "END OF <internal_tab>" ile dahili tablonun sonunu bildirin. aşağıdaki kodda gösterildiği gibi -

```
DATA: BEGIN OF itab01 Occurs 0,
  name LIKE ZCUSTOMERS1-name,
  dob LIKE ZCUSTOMERS1-dob,
END OF itab01.
```

Burada 'itab01', SAP'de geçici tablolar oluştururken yaygın olarak kullanılan stenodur. OCCURS yan tümcesi, tablonun alanlarını bildirerek bir iç tablonun gövdesini tanımlamak için kullanılır. OCCURS yan tümcesi kullanıldığında, gerekirse ek varsayılan belleği belirlemek için sayısal bir sabit 'n' belirtebilirsiniz. OCCUR 0 yan tümcesi tarafından kullanılan varsayılan bellek boyutu 8 KB'dir. Dahili tablonun yapısı şimdi oluşturulmuştur ve kod, onu kayıtlarla doldurmak için yazılabilir.

Başlık satırı kullanılarak veya kullanılmadan dahili bir tablo oluşturulabilir. Başlık satırı olan bir dahili tablo oluşturmak için, dahili tablonun tanımındaki OCCURS yan tümcesinden önce BEGIN OF yan tümcesini veya OCCURS yan tümcesinden sonra WITH HEADER LINE yan tümcesini kullanın. Başlık satırı olmayan bir iç tablo oluşturmak için BEGIN OF yan tümcesi olmadan OCCURS yan tümcesini kullanın.

TYPES deyimini kullanarak yerel veri türü (yalnızca geçerli program bağlamında kullanılan bir veri türü) olarak bir iç tablo da oluşturabilirsiniz. Bu ifade, mevcut bir tabloya atıfta bulunmak için TYPE veya LIKE yan tümcesini kullanır.

Yerel veri türü olarak bir dahili tablo oluşturmak için kullanılan sözdizimi -

```
TYPES <internal_tab> TYPE|LIKE <internal_tab_type> OF
<line_type_itab> WITH <key> INITIAL SIZE <size_number>.
```

Burada <internal_tab_type>, <internal_tab> dahili tablosu için bir tablo türünü belirtir ve <line_type_itab>, bir dahili tablo satırının türünü belirtir. TYPES deyiminde, veri türü olarak bir iç tablonun satır türünü belirtmek için TYPE yan tümcesini ve bir veri nesnesi olarak satır türünü belirtmek için LIKE yan tümcesini kullanabilirsiniz. Dahili tablo için bir anahtar belirtmek isteğe bağlıdır ve kullanıcı bir anahtar belirtmezse, SAP sistemi isteğe bağlı bir anahtarla bir tablo türü tanımlar.

INITIAL SIZE <size_number>, kendisine bir başlangıç bellek miktarı ayırarak dahili bir tablo nesnesi oluşturur. Önceki sözdiziminde, INITIAL SIZE yan tümcesi size_number tablo satırları için bir bellek alanı ayırır. Bir iç tablo nesnesi bildirildiğinde, tablonun boyutu tablonun veri tipine ait değildir.

Not − Bir dahili tablo ilk kez doldurulduğunda çok daha az bellek tüketilir.

Örnek

Adım 1 − SE38 işlem kodunu çalıştırarak ABAP Editörünü açın. ABAP Editörünün başlangıç ekranı belirir.

Adım 2 − Başlangıç ekranında, program için bir ad girin, Kaynak kodu radyo düğmesini seçin ve yeni bir program oluşturmak için Oluştur düğmesine tıklayın.

Adım 3 − 'ABAP: Program Nitelikleri' iletişim kutusunda, Başlık alanına program için kısa bir açıklama girin, Nitelikler grup kutusundaki Tür açılır menüsünden 'Yürütülebilir program' seçeneğini seçin. Kaydet düğmesini tıklayın.

Adım 4 − ABAP düzenleyicisine aşağıdaki kodu yazın.

```
REPORT ZINTERNAL_DEMO.
TYPES: BEGIN OF CustomerLine,
Cust_ID TYPE C,
Cust_Name(20) TYPE C,
END OF CustomerLine.

TYPES mytable TYPE SORTED TABLE OF CustomerLine
WITH UNIQUE KEY Cust_ID.
WRITE:/'The mytable is an Internal Table'.
```

Adım 5 − Programı her zamanki gibi kaydedin, etkinleştirin ve çalıştırın.

Bu örnekte, mytable dahili bir tablodur ve Cust_ID alanında benzersiz bir anahtar tanımlanmıştır.

Yukarıdaki kod aşağıdaki çıktıyı üretir -

The mytable is an Internal Table.

SAP ABAP - Dahili Tabloları Doldurma

Dahili tablolarda doldurma, seçme, ekleme ve ekleme gibi özellikleri içerir. Bu bölüm INSERT ve APPEND ifadelerine odaklanmaktadır.

INSERT İfadesi

INSERT deyimi, dahili bir tabloya tek bir satır veya bir grup satır eklemek için kullanılır.

Dahili bir tabloya tek bir satır eklemek için kullanılan sözdizimi aşağıdadır -

INSERT <work_area_itab> INTO <internal_tab> INDEX <index_num>.

Bu sözdiziminde, INSERT ifadesi internal_tab iç tablosuna yeni bir satır ekler. internal_tab parametresinden önce work_area_itab INTO ifadesi kullanılarak yeni bir satır eklenebilir. work_area_itab INTO ifadesi kullanıldığında, work_area_itab çalışma alanından yeni satır alınır ve internal_tab tablosuna eklenir. Ancak, satır eklemek için work_area_itab INTO ifadesi kullanılmadığında, satır internal_tab tablosunun başlık satırından alınır.

INDEX yan tümcesi kullanılarak bir iç tabloya yeni bir satır eklendiğinde, eklenen satırdan sonraki satırların dizin numarası 1 artırılır. masanın sonu. SAP sistemi bir dahili tabloya başarıyla bir satır eklediğinde, SY-SUBRC değişkeni 0'a ayarlanır.

Örnek

Aşağıda, insert deyimini kullanan örnek bir program verilmiştir.

```
REPORT  ZCUSLIST1.
DATA: BEGIN OF itable1 OCCURS 4,
     F1 LIKE SY-INDEX,
     END OF itable1.

  DO 4 TIMES.
     itable1-F1 = sy-index.
     APPEND itable1.
  ENDDO.

itable1-F1 = -96.
INSERT itable1 INDEX 2.

LOOP AT itable1.
   Write / itable1-F1.
ENDLOOP.

LOOP AT itable1 Where F1 ≥ 3.
   itable1-F1 = -78.
   INSERT itable1.
ENDLOOP.

Skip.
LOOP AT itable1.
   Write / itable1-F1.
ENDLOOP.
```

Yukarıdaki kod aşağıdaki çıktıyı üretir -

```
1
96-
2
3
4
1
96-
2
78-
3
78-
4
```

Yukarıdaki örnekte, DO döngüsü kendisine 1'den 4'e kadar sayıları içeren 4 satır ekler. Başlık satırı bileşeni itable1-F1'e -96 değeri atanmıştır. Insert ifadesi, başlık satırını 3. satırdan önce gövdeye yeni satır olarak ekler. Var olan 3. satır, eklemeden sonra 4. satır olur. LOOP AT deyimi, dahili tablodan F1 değeri 3'e eşit veya daha büyük olan satırları alır. Insert deyimi, her satırdan önce başlık satırından yeni bir satır ekler. Eklemeden önce, F1 bileşeni -78 içerecek şekilde değiştirildi.

Her ekleme ifadesi yürütüldükten sonra, sistem eklenen satırın altındaki tüm satırları yeniden indeksler. Bu, büyük bir dahili tablonun üst kısmına yakın satırlar

eklediğinizde ek yük getirir. Büyük bir dahili tabloya bir satır bloğu eklemeniz gerekiyorsa, eklenecek satırlarla başka bir tablo hazırlayın ve bunun yerine satır ekleme kullanın.

itable1'deki bir döngünün içine itable1'in içine yeni bir satır eklerken, dahili tabloyu anında etkilemez. Aslında bir sonraki döngü geçişinde etkili olur. Geçerli satırdan sonra bir satır eklerken, tablo ENDLOOP'ta yeniden indekslenir. Sy-tabix artırılır ve sonraki döngü, sy-tabix ile gösterilen satırı işler. Örneğin, ikinci döngü geçişindeyseniz ve 3. satırdan önce bir kayıt eklerseniz, endloop yürütüldüğünde, yeni satır 3. satır ve eski 3. satır 4. satır olur ve bu böyle devam eder. Sy-tabix 1 artırılır ve sonraki döngü geçişi yeni eklenen kaydı işler.

EK Bildirimi

APPEND ifadesi, mevcut bir dahili tabloya tek bir satır veya satır eklemek için kullanılır. Bu ifade, bir çalışma alanından tek bir satırı kopyalar ve bunu dahili bir tabloda var olan son satırın arkasına ekler. Çalışma alanı, bir başlık satırı veya bir iç tablo satırıyla aynı yapıya sahip başka bir alan dizesi olabilir. Dahili bir tabloda tek bir satır eklemek için kullanılan APPEND ifadesinin sözdizimi aşağıdadır -

APPEND <record_for_itab> TO <internal_tab>.

Bu sözdiziminde, <record_for_itab> ifadesi, bir satır türüne dönüştürülebilen <work_area_itab> çalışma alanı veya INITIAL LINE yan tümcesi ile temsil edilebilir. Kullanıcı bir <work_area_itab> çalışma alanı kullanıyorsa, SAP sistemi <internal_tab> dahili tablosuna yeni bir satır ekler ve onu çalışma alanının içeriğiyle doldurur. INITIAL LINE yan tümcesi, tablo yapısının her alanı için başlangıç değerini içeren boş bir satır ekler. Her APPEND ifadesinden sonra, SY-TABIX değişkeni, eklenen satırın indeks numarasını içerir.

Benzersiz olmayan bir anahtarla standart ve sıralanmış tablolara satırlar eklemek, aynı anahtara sahip satırların zaten tabloda bulunup bulunmadığından bağımsız olarak çalışır. Başka bir deyişle, yinelenen girişler oluşabilir. Ancak, kullanıcı benzersiz bir anahtarla sıralanmış bir tabloya yinelenen bir giriş eklemeye çalışırsa veya kullanıcı, sıralanmış bir tablonun sıralama düzenini ona satırlar ekleyerek ihlal ederse, bir çalışma zamanı hatası oluşur.

Örnek

```
REPORT  ZCUSLIST1.
DATA: BEGIN OF linv Occurs 0,
     Name(20) TYPE C,
     ID_Number TYPE I,
END OF linv.

DATA table1 LIKE TABLE OF linv.
linv-Name = 'Melissa'.
linv-ID_Number = 105467.
APPEND linv TO table1.
LOOP AT table1 INTO linv.

Write: / linv-name, linv-ID_Number.
ENDLOOP.
```

Yukarıdaki kod aşağıdaki çıktıyı üretir -

Melissa 105467

SAP ABAP - Dahili Tabloları Kopyalama

Başlık satırı olan dahili bir tablodan bir kayıt okuduğumuzda, bu kayıt tablonun kendisinden başlık satırına taşınır. Daha sonra programımızın çalıştığı başlık satırıdır. Aynısı yeni bir kayıt oluştururken de geçerlidir. Birlikte çalıştığınız ve yeni kaydın tablo gövdesinin kendisine gönderildiği başlık satırıdır.

Kayıtları kopyalamak için, tablodan tüm kayıtları seçmek için bir SELECT deyimi kullanabilir ve ardından kayıtları orijinal tablodan yeni dahili tabloya isimlerin karşılık geldiği alanlara taşıyacak olan MOVE deyimini kullanabiliriz.

MOVE deyiminin sözdizimi aşağıdadır -

MOVE <table_field> TO <internal_tab_field>.

Örnek

```
REPORT  ZCUSLIST1.
TABLES: ZCUSTOMERS1.
DATA: BEGIN OF itab01 Occurs 0,
    name LIKE ZCUSTOMERS1-name,
    dob LIKE ZCUSTOMERS1-dob,
END OF itab01.

Select * FROM ZCUSTOMERS1.
MOVE ZCUSTOMERS1-name TO itab01-name.
MOVE ZCUSTOMERS1-dob TO itab01-dob.
ENDSELECT.

Write: / itab01-name, itab01-dob.
```

Yukarıdaki kod aşağıdaki çıktıyı üretir -

MARGARET 02.11.1994

Seçim döngüsü, verileri bir tablonun alanından diğerine taşımak için MOVE deyimini kullanarak her bir alanı birer birer doldurur. Yukarıdaki örnekte, ZCUSTOMERS1 tablosunun içeriğini dahili tablodaki ilgili alanlara taşımak için MOVE deyimleri kullanılmıştır. Bu eylemi sadece bir kod satırı ile gerçekleştirebilirsiniz. MOVECORRESPONDING ifadesini kullanabilirsiniz.

MOVE-CORRESPONDING ifadesinin sözdizimi aşağıdadır -

MOVE-CORRESPONDING <table_name> TO <internal_tab>.

Sisteme verileri ZCUSTOMERS1 alanlarından itab01'deki karşılık gelen alanlarına taşımasını söyler.

Örnek

```
REPORT  ZCUSTOMERLIST.
```

```
TABLES: ZCUSTOMERS1.
DATA: Begin of itab01 occurs 0,
    customer LIKE ZCUSTOMERS1-customer,
    name LIKE ZCUSTOMERS1-name,
    title LIKE ZCUSTOMERS1-title,
    dob LIKE ZCUSTOMERS1-dob,
END OF itab01.

SELECT * from ZCUSTOMERS1.
MOVE-Corresponding ZCUSTOMERS1 TO itab01.
APPEND itab01.
ENDSELECT.
LOOP AT itab01.
Write: / itab01-name, itab01-dob.
ENDLOOP.
```

Yukarıdaki kod aşağıdaki çıktıyı üretir -

```
MARK 21.05.1981
JAMES 14.08.1977
AURIELE 19.06.1990
STEPHEN 22.07.1985
MARGARET 02.11.1994
```

Bu, her ikisinin de eşleşen alan adlarına sahip olmasıyla mümkün olur. Bu ifadeyi kullanırken, her iki alanın da eşleşen veri türleri ve uzunluklarına sahip olduğundan emin olmanız gerekir. Burada daha önce LIKE ifadesi ile yapılmıştır.

SAP ABAP - Dahili Tabloları Okuma

READ TABLE ifadesinin aşağıdaki sözdizimini kullanarak bir tablonun satırlarını okuyabiliriz -

```
READ TABLE <internal_table> FROM <work_area_itab>.
```

Bu söz diziminde, <work_area_itab> ifadesi, <internal_table> tablosunun satır türüyle uyumlu bir çalışma alanını temsil eder. Aşağıdaki sözdiziminde gösterildiği gibi, WITH KEY yan tümcesini kullanarak READ deyimi içinde bir arama anahtarı belirtebiliriz, ancak bir tablo anahtarı belirtemeyiz -

```
READ TABLE <internal_table> WITH KEY = <internal_tab_field>.
```

Burada dahili tablonun tüm satırı bir **arama anahtarı** olarak kullanılır . Tablonun tüm satırının içeriği, <internal_tab_field> alanının içeriğiyle karşılaştırılır. <internal_tab_field> alanının değerleri tablonun satır tipi ile uyumlu değilse bu değerler tablonun satır tipine göre dönüştürülür. Arama tuşu, yapılandırılmış bir satır türüne sahip olmayan, yani satırın tek bir alan veya bir dahili tablo türü olduğu dahili tablolardaki girdileri bulmanızı sağlar.

READ ifadesinin aşağıdaki sözdizimi, COMPARING yan tümcesi kullanılarak bir çalışma alanı veya alan sembolü belirtmek için kullanılır -

```
READ TABLE <internal_table> <key> INTO <work_area_itab>
[COMPARING <F1> <F2>...<Fn>].
```

COMPARING yan tümcesi kullanıldığında, yapılandırılmış hat türünün belirtilen <F1>, <F2>...<Fn> tablo alanları, taşınmadan önce çalışma alanının karşılık gelen alanlarıyla karşılaştırılır. ALL FIELDS maddesi belirtilirse, SAP sistemi tüm bileşenleri karşılaştırır. SAP sistemi bir anahtar bazında bir giriş bulduğunda, SY-SUBRC değişkeninin değeri 0 olarak ayarlanır. Ayrıca, karşılaştırılan içeriğin içeriği ise SY-SUBRC değişkeninin değeri 2 veya 4 olarak ayarlanır. alanlar aynı değilse veya SAP sistemi bir giriş bulamıyorsa. Ancak SAP sistemi, karşılaştırmanın sonucundan bağımsız olarak, bir giriş bulduğunda girişi hedef çalışma alanına kopyalar.

Örnek

```
REPORT  ZREAD_DEMO.
*/Creating an internal table
DATA: BEGIN OF Record1,
ColP TYPE I,
ColQ TYPE I,
END OF Record1.

DATA mytable LIKE HASHED TABLE OF Record1 WITH UNIQUE KEY ColP.
DO 6 Times.
Record1-ColP = SY-INDEX.
Record1-ColQ = SY-INDEX + 5.
INSERT Record1 INTO TABLE mytable.
ENDDO.

Record1-ColP = 4.
Record1-ColQ = 12.
READ TABLE mytable FROM Record1 INTO Record1 COMPARING ColQ.

WRITE: 'SY-SUBRC =', SY-SUBRC.
SKIP.
WRITE: / Record1-ColP, Record1-ColQ.
```

Yukarıdaki kod aşağıdaki çıktıyı üretir -

```
SY-SUBRC = 2

4 9
```

Yukarıdaki örnekte, mytable, çalışma alanı olarak Record1 ve benzersiz anahtar olarak ColP ile, karma tablo türünün dahili bir tablosudur. Başlangıçta, mytable altı satırla doldurulur, burada ColP alanı SY-INDEX değişkeninin değerlerini içerir ve ColQ alanı (SY-INDEX + 5) değerlerini içerir.

Record1 çalışma alanı, sırasıyla ColP ve ColQ alanları için değerler olarak 4 ve 12 ile doldurulur. READ ifadesi, COMPARING yan tümcesini kullanarak ColP anahtar alanının değerini Record1 çalışma alanındaki değerle karşılaştırdıktan sonra tablonun satırını okur ve ardından okuma satırının içeriğini çalışma alanına kopyalar. SY-SUBRC değişkeninin değeri 2 olarak görüntülenir çünkü ColP alanındaki değer 4 olduğunda ColQ'daki değer 12 değil 9'dur.

SAP ABAP - Dahili Tabloları Silme

DELETE ifadesi, dahili bir tablodan bir veya daha fazla kaydı silmek için kullanılır. Bir iç tablonun kayıtları, ya bir tablo anahtarı ya da koşulu belirtilerek ya da mükerrer girişler bulunarak silinir. Bir iç tablo benzersiz olmayan bir anahtara sahipse ve yinelenen girişler içeriyorsa, tablodaki ilk giriş silinir.

Dahili bir tablodan bir kaydı veya satırı silmek için DELETE deyimini kullanmak için kullanılan sözdizimi aşağıdadır -

DELETE TABLE <internal_table> FROM <work_area_itab>.

Yukarıdaki sözdiziminde, <work_area_itab> ifadesi bir çalışma alanıdır ve <internal_table> dahili tablonun türüyle uyumlu olmalıdır. Silme işlemi, çalışma alanı bileşenlerinden alınabilecek varsayılan bir anahtar temelinde gerçekleştirilir.

Ayrıca aşağıdaki sözdizimini kullanarak DELETE TABLE ifadesinde bir tablo anahtarı belirtebilirsiniz -

DELETE TABLE <internal_table> WITH TABLE KEY <K1> = <F1>…… <Kn> = <Fn>.

Bu sözdiziminde, <F1>, <F2>....<Fn> bir iç tablonun alanlarıdır ve <K1>, <K2>....<Kn> tablonun anahtar alanlarıdır. DELETE ifadesi, <K1> = <F1>, <K2> = <F2>...<Kn> = <Fn> ifadelerine dayalı olarak <internal_table> tablosunun kayıtlarını veya satırlarını silmek için kullanılır.

Not − <F1>, <F2>....<Fn> alanlarının veri türleri <K1>, <K2>...<Kn> anahtar alanları ile uyumlu değilse, SAP sistemi bunları otomatik olarak dönüştürür. uyumlu formata dönüştürün.

Örnek

```
REPORT  ZDELETE_DEMO.
DATA: BEGIN OF Line1,
ColP TYPE I,
ColQ TYPE I,
END OF Line1.
DATA mytable LIKE HASHED TABLE OF Line1
WITH UNIQUE KEY ColP.
DO 8 TIMES.

Line1-ColP = SY-INDEX.
Line1-ColQ = SY-INDEX + 4.
INSERT Line1 INTO TABLE mytable.
ENDDO.

Line1-ColP = 1.
DELETE TABLE mytable: FROM Line1,
WITH TABLE KEY ColP = 3.
LOOP AT mytable INTO Line1.

WRITE: / Line1-ColP, Line1-ColQ.
ENDLOOP.
```

Yukarıdaki kod aşağıdaki çıktıyı üretir -

```
2 6
```

4 8
5 9
6 10
7 11
8 12

Bu örnekte, mytable'ın ColP ve ColQ olmak üzere iki alanı vardır. Başlangıçta, mytable sekiz satırla doldurulur, burada ColP 1, 2, 3, 4, 5, 6, 7 ve 8 değerlerini içerir. ColQ 5, 6, 7, 8, 9, 10, 11 ve 12 çünkü ColP değerleri her seferinde 4 artırılır.

DELETE ifadesi, ColP anahtar alanının değeri 1 veya 3 olan satırları mytable'dan silmek için kullanılır. Silme işleminden sonra, mytable'ın ColP alanı gösterildiği gibi 2, 4, 5, 6, 7 ve 8 değerlerini içerir. çıktıda. ColQ alanı 6, 8, 9, 10, 11 ve 12 değerlerini içerir.

SAP ABAP - Nesne Yönelimi

Nesne yönelimi, yazılım tasarımını, anlaşılmasını, bakımını ve yeniden kullanımını kolaylaştırmak için basitleştirir. **Nesneye Yönelik Programlama** (OOP), yazılım yazarken farklı bir düşünme biçimini temsil eder. OOP'nin güzelliği basitliğinde yatmaktadır. OOP'nin etkileyiciliği, kaliteli yazılım bileşenlerinin zamanında teslim edilmesini kolaylaştırır.

Çözümler gerçek dünyadaki nesneler açısından tasarlandığından, programcıların ve iş analistlerinin ortak bir etki alanı dili kullanan bir tasarım hakkında fikir ve bilgi alışverişinde bulunmaları çok daha kolay hale gelir. İletişimdeki bu gelişmeler, gizli gereksinimlerin ortaya çıkarılmasına, risklerin belirlenmesine ve geliştirilmekte olan yazılımın kalitesinin artırılmasına yardımcı olur. Nesne yönelimli yaklaşım, gerçek dünyanın soyut veya somut şeylerini temsil eden nesnelere odaklanır. Bu nesneler, iç yapıları ve nitelikleri (verileri) ile temsil edilen karakterleri ve özellikleri ile tanımlanır. Bu nesnelerin davranışı, yöntemlerle (yani işlevsellik) tanımlanır.

Prosedürel ve nesne yönelimli programlamayı karşılaştıralım -

Özellikler	Prosedür Odaklı yaklaşım	Nes
Vurgu	Vurgu görevler üzerindedir.	Vurgu yapar üzerir
modülerleştirme	Programlar, işlevler olarak bilinen daha küçük programlara bölünebilir.	Progr ve ne düzer bir sı yönte gömü

Veri güvenliği	İşlevlerin çoğu genel verileri paylaşır.	Verile harici tarafı erişile
genişletilebilirlik	Bu, mevcut işlevselliği değiştirmek ve genişletmek için daha fazla zaman alır.	Yeni işlevl zahm eklen

ABAP başlangıçta bir prosedür dili olarak geliştirildi (sadece COBOL gibi önceki prosedürel programlama diline benzer). Ancak ABAP, artık ABAP Nesnelerinin tanıtılmasıyla nesne yönelimli paradigmaların ilkelerini uyarlamıştır. ABAP'taki sınıf, nesne, kalıtım ve polimorfizm gibi nesne yönelimli kavramlar, esasen Java veya C++ gibi diğer modern nesne yönelimli dillerinkilerle aynıdır.

Nesne yönelimi şekillenmeye başladığında, her sınıf belirli rol atamalarını üstlenir. Bu iş bölümü, genel programlama modelini basitleştirmeye yardımcı olur ve her sınıfın eldeki sorunun belirli bir parçasını çözmede uzmanlaşmasını sağlar. Bu tür sınıfların yüksek bir uyumu vardır ve her sınıfın işlemleri sezgisel bir şekilde yakından ilişkilidir.

Nesne yöneliminin temel özellikleri şunlardır:

- Etkili programlama yapısı.
- Gerçek dünya varlıkları çok iyi modellenebilir.
- Veri güvenliği ve erişimi üzerinde stres.
- Kod fazlalığını en aza indirir.
- Veri soyutlama ve kapsülleme.

SAP ABAP - Nesneler

Bir nesne, farklı özelliklere ve davranışlara sahip özel bir değişken türüdür. Bir nesnenin özellikleri veya nitelikleri, bir nesnenin durumunu tanımlamak için kullanılır ve davranışlar veya yöntemler, bir nesne tarafından gerçekleştirilen eylemleri temsil eder.

Bir nesne, bir sınıfın modeli veya örneğidir. Bir kişi gibi gerçek dünyadaki bir varlığı veya değişkenler ve sabitler gibi bir programlama varlığını temsil eder. Örneğin, hesaplar ve öğrenciler gerçek dünyadaki varlıkların örnekleridir. Ancak bir bilgisayarın donanım ve yazılım bileşenleri, programlama varlıklarının örnekleridir.

Bir nesne aşağıdaki üç ana özelliğe sahiptir -

- devleti vardır.
- Eşsiz bir kimliğe sahiptir.
- Davranışı gösterebilir veya göstermeyebilir.

Bir nesnenin durumu, bir dizi nitelik ve bunların değerleri olarak tanımlanabilir. Örneğin, bir banka hesabının Hesap Numarası, Adı, Hesap Türü,

Bakiye gibi bir dizi özelliği ve tüm bu niteliklerin değerleri vardır. Bir nesnenin davranışı, belirli bir süre boyunca özniteliklerinde meydana gelen değişiklikleri ifade eder.

Her nesnenin, onu diğer nesnelerden ayırt etmek için kullanılabilecek benzersiz bir kimliği vardır. İki nesne aynı davranışı sergileyebilir ve aynı duruma sahip olabilir veya olmayabilir, ancak asla aynı kimliğe sahip olmazlar. İki kişi aynı ada, yaşa ve cinsiyete sahip olabilir, ancak bunlar aynı değildir. Benzer şekilde, bir nesnenin kimliği de ömrü boyunca asla değişmez.

Nesneler mesaj göndererek birbirleriyle etkileşime girebilir. Nesneler, verileri işlemek için veri ve kod içerir. Bir nesne, bir sınıf yardımıyla kullanıcı tanımlı bir veri türü olarak da kullanılabilir. Nesnelere, tür sınıfının değişkenleri de denir. Bir sınıf tanımladıktan sonra o sınıfa ait istediğiniz sayıda nesne oluşturabilirsiniz. Her nesne, oluşturulduğu tür sınıfının verileriyle ilişkilendirilir.

Nesne Oluşturma

Nesne oluşturma genellikle aşağıdaki adımları içerir -

- Sınıfa referansla bir referans değişkeni oluşturma. Bunun sözdizimi -

DATA: <object_name> TYPE REF TO <class_name>.

- Referans değişkeninden bir nesne oluşturma. Bunun sözdizimi -

CREATE Object: <object_name>.

Örnek

```
REPORT ZDEMO_OBJECT.
CLASS Class1 Definition.
Public Section.
DATA: text1(45) VALUE 'ABAP Objects.'.
METHODS: Display1.
ENDCLASS.

CLASS Class1 Implementation.
METHOD Display1.
Write:/ 'This is the Display method.'.
ENDMETHOD.
ENDCLASS.

START-OF-SELECTION.
DATA: Class1 TYPE REF TO Class1.
CREATE Object: Class1.
Write:/ Class1->text1.
CALL METHOD: Class1->Display1.
```

Yukarıdaki kod aşağıdaki çıktıyı üretir -

```
ABAP Objects.
This is the Display method.
```

SAP ABAP - Sınıflar

Bir nesnenin biçimini belirtmek için bir sınıf kullanılır ve veri gösterimini ve bu verileri tek bir düzgün pakette işlemek için yöntemleri birleştirir. Bir sınıf içindeki veriler ve işlevler, sınıfın **üyeleri** olarak adlandırılır .

Sınıf Tanımı ve Uygulaması

Bir sınıf tanımladığınızda, bir veri türü için bir plan tanımlarsınız. Bu aslında herhangi bir veriyi tanımlamaz, ancak sınıf adının ne anlama geldiğini, sınıfın bir nesnesinin nelerden oluşacağını ve böyle bir nesne üzerinde hangi işlemlerin gerçekleştirilebileceğini tanımlar. Yani öznitelikler, alanlar ve özellikler gibi bir nesnenin soyut özelliklerini tanımlar.

Aşağıdaki sözdizimi bir sınıfın nasıl tanımlanacağını gösterir -

```
CLASS <class_name> DEFINITION.
.....
.....
ENDCLASS.
```

Bir sınıf tanımı, CLASS anahtar sözcüğüyle başlar, ardından sınıf adı, DEFINITION ve sınıf gövdesi gelir. Bir sınıfın tanımı, nitelikler, yöntemler ve olaylar gibi sınıfın çeşitli bileşenlerini içerebilir. Sınıf bildiriminde bir yöntem bildirdiğimizde, yöntem uygulaması sınıf uygulamasına dahil edilmelidir. Aşağıdaki sözdizimi bir sınıfın nasıl uygulanacağını gösterir -

```
CLASS <class_name> IMPLEMENTATION.
.....
.....
ENDCLASS.
```

Not − Bir sınıfın uygulanması, tüm yöntemlerinin uygulanmasını içerir. ABAP Nesnelerinde, bir sınıfın yapısı öznitelikler, yöntemler, olaylar, türler ve sabitler gibi bileşenler içerir.

Öznitellikler

Öznitelikler, C, I, F ve N gibi herhangi bir veri türüne sahip olabilen bir sınıfın veri alanlarıdır. Sınıf bildiriminde bildirilirler. Bu nitelikler 2 kategoriye ayrılabilir: örnek ve statik nitelikler. Bir **örnek niteliği** , bir nesnenin örneğe özgü durumunu tanımlar. Durumlar farklı nesneler için farklıdır. DATA deyimi kullanılarak bir örnek özniteliği bildirilir.

Statik nitelikler , sınıfın tüm örnekleri tarafından paylaşılan bir sınıfın ortak durumunu tanımlar. Yani, bir sınıfın bir nesnesindeki statik bir niteliği değiştirirseniz, değişiklik, sınıfın diğer tüm nesneleri tarafından da görülebilir. Statik bir öznitelik, CLASS-DATA deyimi kullanılarak bildirilir.

yöntemler

Yöntem, sınıftaki bir nesnenin davranışını temsil eden bir işlev veya prosedürdür. Sınıfın yöntemleri, sınıfın herhangi bir özelliğine erişebilir. Bir yöntemin tanımı ayrıca parametreler içerebilir, böylece yöntemler çağrıldığında bu

parametrelere değerler sağlayabilirsiniz. Bir yöntemin tanımı, sınıf bildiriminde bildirilir ve bir sınıfın uygulama kısmında uygulanır. METHOD ve ENDMETHOD ifadeleri, bir metodun uygulama kısmını tanımlamak için kullanılır. Aşağıdaki sözdizimi, bir yöntemin nasıl uygulanacağını gösterir -

```
METHOD <m_name>.
.....
.....
ENDMETHOD.
```

Bu söz diziminde <m_name>, bir yöntemin adını temsil eder. **Not** − CALL METHOD deyimini kullanarak bir yöntemi çağırabilirsiniz.

Niteliklere ve Yöntemlere Erişme

Sınıf bileşenleri, bu bileşenlere nasıl erişilebileceğini kontrol eden genel, özel veya korumalı görünürlük bölümlerinde tanımlanabilir. Özel görünürlük bölümü, bileşenlere sınıfın dışından erişimi engellemek için kullanılır. Bu tür bileşenlere yalnızca bir yöntem gibi sınıfın içinden erişilebilir.

Genel görünürlük bölümünde tanımlanan bileşenlere herhangi bir bağlamdan erişilebilir. Varsayılan olarak, bir sınıfın tüm üyeleri özel olacaktır. Pratik olarak, aşağıdaki programda gösterildiği gibi sınıf dışından çağrılabilmeleri için verileri özel bölümde ve ilgili yöntemleri de genel bölümde tanımlarız.

- Bir sınıfta Public bölümünde tanımlanan özniteliklere ve yöntemlere, o sınıf ve programın herhangi bir alt sınıfı, alt sınıfı tarafından erişilebilir.
- Bir sınıfın Korumalı bölümünde öznitelikler ve yöntemler bildirildiğinde, bunlara yalnızca o sınıf ve alt sınıflar (türetilmiş sınıflar) tarafından erişilebilir.
- Öznitelikler ve yöntemler bir sınıfın Özel bölümünde bildirildiğinde, bunlara başka bir sınıf tarafından değil, yalnızca o sınıf tarafından erişilebilir.

Örnek

```
Report ZAccess1.
CLASS class1 Definition.
  PUBLIC Section.
    Data: text1 Type char25 Value 'Public Data'.
    Methods meth1.

  PROTECTED Section.
    Data: text2 Type char25 Value 'Protected Data'.

  PRIVATE Section.
    Data: text3 Type char25 Value 'Private Data'.
ENDCLASS.

CLASS class1 Implementation.
  Method meth1.
    Write: / 'Public Method:',
           / text1,
           / text2,
           / text3.
    Skip.
```

```
  EndMethod.
ENDCLASS.

Start-Of-Selection.
  Data: Objectx Type Ref To class1.
  Create Object: Objectx.
  CALL Method: Objectx→meth1.
  Write: / Objectx→text1.
```

Yukarıdaki kod aşağıdaki çıktıyı üretir -

```
Public Method:
Public Data
Protected Data
Private Data

Public Data
```

Statik Özellikler

Statik bir öznitelik, CLASS-DATA deyimiyle bildirilir. Tüm nesneler veya örnekler, sınıfın statik niteliğini kullanabilir. Statik özniteliklere, class_name⇒name_1 = 'Some Text' gibi sınıf adı yardımıyla doğrudan erişilir.

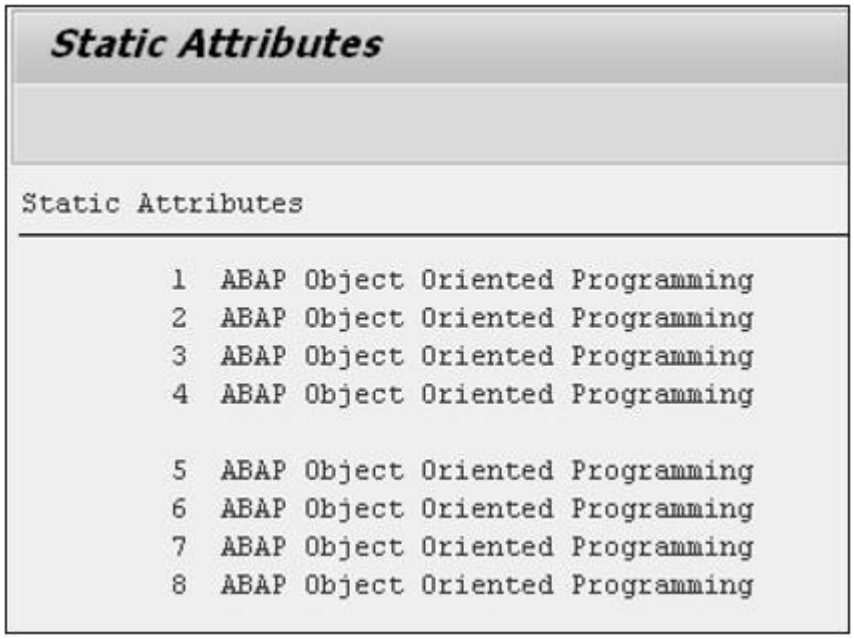
Örnek

Aşağıda satır numarası 4 ila 8 kez olan bir metni yazdırmak istediğimiz bir program var. Bir class1 sınıfı tanımlarız ve public bölümünde CLASS-DATA (statik özellik) ve bir metot ilan ederiz. Sınıfı ve yöntemi uyguladıktan sonra, Start-Of-Selection olayında static özelliğine doğrudan erişiriz. Sonra sadece sınıfın örneğini yaratırız ve metodu çağırırız.

```
Report ZStatic1.
CLASS class1 Definition.
  PUBLIC Section.
    CLASS-DATA: name1 Type char45,
          data1 Type I.
  Methods: meth1.
ENDCLASS.

CLASS class1 Implementation.
  Method meth1.
    Do 4 Times.
      data1 = 1 + data1.
      Write: / data1, name1.
    EndDo.
    Skip.
  EndMethod.
ENDCLASS.

Start-Of-Selection.
  class1⇒name1 = 'ABAP Object Oriented Programming'.
  class1⇒data1 = 0.
```

```
Data: Object1 Type Ref To class1,
      Object2 Type Ref To class1.

Create Object: Object1, Object2.
CALL Method: Object1→meth1,
             Object2→meth1.
```

Yukarıdaki kod aşağıdaki çıktıyı üretir -



yapıcılar

Yapıcılar, bir nesne oluştururken veya bir sınıfın bileşenlerine erişirken otomatik olarak çağrılan özel yöntemlerdir. Yapıcı, bir nesne oluşturulduğunda tetiklenir, ancak genel yöntemi tetiklemek için bir yöntem çağırmamız gerekir. Aşağıdaki örnekte, method1 ve yapıcı olmak üzere iki genel yöntem bildirdik. Bu yöntemlerin her ikisi de farklı işlemlere sahiptir. Sınıfın bir nesnesini oluştururken, yapıcı yöntemi onun çalışmasını tetikler.

Örnek

```
Report ZConstructor1.
CLASS class1 Definition.
  PUBLIC Section.
    Methods: method1, constructor.
ENDCLASS.

CLASS class1 Implementation.
  Method method1.
    Write: / 'This is Method1'.
  EndMethod.

  Method constructor.
    Write: / 'Constructor Triggered'.
  EndMethod.
```

```
ENDCLASS.

Start-Of-Selection.
  Data Object1 Type Ref To class1.
  Create Object Object1.
```

Yukarıdaki kod aşağıdaki çıktıyı üretir -

```
Constructor Triggered
```

Yöntemlerde ME Operatörü

Bir sınıfın genel bölümünde herhangi bir türden bir değişken bildirdiğinizde, onu başka herhangi bir uygulamada kullanabilirsiniz. Bir değişken, genel bölümde bir başlangıç değeri ile bildirilebilir. Değişkeni farklı bir değere sahip bir metot içinde yeniden tanımlayabiliriz. Değişkeni metodun içine yazdığımızda sistem değiştirilen değeri yazdıracaktır. Değişkenin önceki değerini yansıtmak için 'ME' operatörünü kullanmalıyız.

Bu programda bir public değişkeni text1 tanımladık ve bir değer ile başlattık. Aynı değişkeni tekrar bildirdik, ancak farklı bir değerle somutlaştırdık. Yöntemin içinde, daha önce başlatılan değeri almak için o değişkeni 'ME' operatörüyle yazıyoruz. Değiştirilen değeri doğrudan bildirerek elde ederiz.

Örnek

```
Report ZMEOperator1.
CLASS class1 Definition.
  PUBLIC Section.

Data text1 Type char25 Value 'This is CLASS Attribute'.
  Methods method1.
ENDCLASS.

CLASS class1 Implementation.
  Method method1.

Data text1 Type char25 Value 'This is METHOD Attribute'.
  Write: / ME→text1,
         / text1.
  ENDMethod.
ENDCLASS.

Start-Of-Selection.
  Data objectx Type Ref To class1.
  Create Object objectx.
  CALL Method objectx→method1.
```

Yukarıdaki kod aşağıdaki çıktıyı üretir -

```
This is CLASS Attribute
This is METHOD Attribute
```

SAP ABAP - Kalıtım

Nesne yönelimli programlamadaki en önemli kavramlardan biri kalıtımdır. Kalıtım, bir sınıfı başka bir sınıf açısından tanımlamamızı sağlar, bu da bir uygulamanın oluşturulmasını ve bakımını kolaylaştırır. Bu aynı zamanda kod işlevselliğini ve hızlı uygulama süresini yeniden kullanma fırsatı sağlar.

Bir sınıf oluştururken, tamamen yeni veri üyeleri ve yöntemler yazmak yerine, programcı yeni sınıfın mevcut bir sınıfın üyelerini devralması gerektiğini belirleyebilir. Bu mevcut sınıfa **temel sınıf** veya **süper sınıf** denir ve yeni sınıf, **türetilmiş sınıf** veya **alt sınıf** olarak adlandırılır .

- Bir sınıfın nesnesi, başka bir sınıfın özelliklerini alabilir.
- Türetilmiş sınıf, bir süper sınıfın verilerini ve yöntemlerini devralır. Ancak, yöntemlerin üzerine yazabilir ve ayrıca yeni yöntemler ekleyebilirler.
- Kalıtımın ana avantajı yeniden kullanılabilirliktir.

Miras ilişkisi, sınıf tanımı deyimine 'INHERITING FROM' eki kullanılarak belirtilir.

Sözdizimi aşağıdadır -

CLASS <subclass> DEFINITION INHERITING FROM <superclass>.

Örnek

```
Report ZINHERITAN_1.
CLASS Parent Definition.
PUBLIC Section.
Data: w_public(25) Value 'This is public data'.
Methods: ParentM.
ENDCLASS.

CLASS Child Definition Inheriting From Parent.
PUBLIC Section.
Methods: ChildM.
ENDCLASS.

CLASS Parent Implementation.
Method ParentM.
Write /: w_public.
EndMethod. ENDCLASS.

CLASS Child Implementation.
Method ChildM.
Skip.
Write /: 'Method in child class', w_public.
EndMethod.
ENDCLASS.

Start-of-selection.
Data: Parent Type Ref To Parent,
Child Type Ref To Child.
Create Object: Parent, Child.
Call Method: Parent→ParentM,
child→ChildM.
```

Yukarıdaki kod aşağıdaki çıktıyı üretir -

```
This is public data
Method in child class
This is public data
```

Erişim Kontrolü ve Kalıtım

Türetilmiş bir sınıf, temel sınıfının özel olmayan tüm üyelerine erişebilir. Bu nedenle, alt sınıfların üye işlevlerine erişilmemesi gereken süper sınıf üyeleri, süper sınıfta özel olarak bildirilmelidir. Kimlerin erişebildiğine göre farklı erişim türlerini şu şekilde özetleyebiliriz:

Erişim	Halk	Korumalı	
aynı cals	Evet	Evet	
Türetilmiş sınıf	Evet	Evet	
sınıf dışı	Evet	Numara	

Bir süper sınıftan bir sınıf türetildiğinde, genel, korumalı veya özel miras yoluyla miras alınabilir. Kalıtımın türü, yukarıda açıklandığı gibi erişim belirteci tarafından belirlenir. Korumalı veya özel kalıtımı pek kullanmıyoruz, ancak genel kalıtım yaygın olarak kullanılıyor. Farklı kalıtım türleri kullanılırken aşağıdaki kurallar uygulanır.

- **Genel Kalıtım** − Bir genel üst sınıftan bir sınıf türetirken, üst sınıfın genel üyeleri alt sınıfın genel üyeleri olur ve üst sınıfın korumalı üyeleri alt sınıfın korumalı üyeleri olur. Süper sınıfın özel üyelerine hiçbir zaman doğrudan bir alt sınıftan erişilemez, ancak üst sınıfın genel ve korunan üyelerine yapılan çağrılar yoluyla erişilebilir.
- **Korumalı Kalıtım** - Korumalı bir üst sınıftan türetildiğinde, üst sınıfın genel ve korunan üyeleri, alt sınıfın korumalı üyeleri haline gelir.
- **Özel Kalıtım** - Özel bir üst sınıftan türetildiğinde, üst sınıfın genel ve korunan üyeleri, alt sınıfın özel üyeleri haline gelir.

Alt Sınıfta Yöntemleri Yeniden Tanımlama

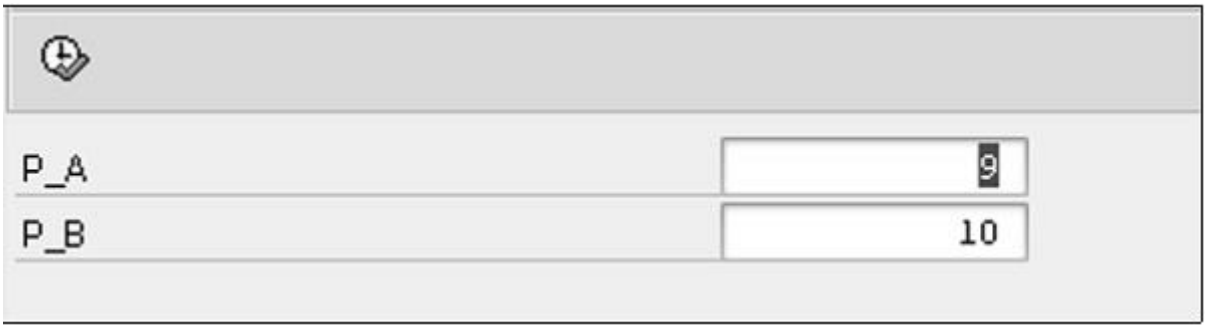
Süper sınıfın yöntemleri alt sınıfta yeniden uygulanabilir. Yöntemleri yeniden tanımlamanın birkaç kuralı -

- Devralınan yöntemin yeniden tanımlama ifadesi, orijinal yöntemin tanımıyla aynı bölümde olmalıdır.
- Bir yöntemi yeniden tanımlarsanız, alt sınıfa onun arabirimini yeniden girmeniz gerekmez, yalnızca yöntemin adını girmeniz gerekir.
- Yeniden tanımlanmış yöntem içinde, süper referansı kullanarak doğrudan süper sınıfın bileşenlerine erişebilirsiniz.
- Sözde referans süper yalnızca yeniden tanımlanmış yöntemlerde kullanılabilir.

Örnek

```
Report Zinheri_Redefine.
CLASS super_class Definition.
Public Section.
Methods: Addition1 importing g_a TYPE I
                        g_b TYPE I
                            exporting g_c TYPE I.
ENDCLASS.

CLASS super_class Implementation.
Method Addition1.
g_c = g_a + g_b.
EndMethod.
ENDCLASS.

CLASS sub_class Definition Inheriting From super_class.
Public Section.
METHODS: Addition1 Redefinition.
ENDCLASS.

CLASS sub_class Implementation.
Method Addition1.
g_c = g_a + g_b + 10.
EndMethod.
ENDCLASS.

Start-Of-Selection.
Parameters: P_a Type I, P_b TYPE I.
Data: H_Addition1 TYPE I.
Data: H_Sub TYPE I.
Data: Ref1 TYPE Ref TO sub_class.
Create Object Ref1.
Call Method Ref1→Addition1 exporting g_a = P_a
                                g_b = P_b
                                    Importing g_c = H_Addition1.
Write:/ H_Addition1.
```

F8'i çalıştırdıktan sonra, 9 ve 10 değerlerini girersek, yukarıdaki kod aşağıdaki çıktıyı verir -



Redefinition Demo
29

SAP ABAP - Polimorfizm

Polimorfizm terimi, kelimenin tam anlamıyla 'birçok form' anlamına gelir. Nesne yönelimli bir perspektiften bakıldığında, polimorfizm, bir kalıtım ağacındaki çeşitli türlerin birbirinin yerine kullanılmasını mümkün kılmak için kalıtımla birlikte çalışır. Yani, polimorfizm, bir sınıflar hiyerarşisi olduğunda ortaya çıkar ve bunlar kalıtım yoluyla ilişkilidir. ABAP polimorfizmi, bir yönteme yapılan çağrının, yöntemi çağıran nesnenin türüne bağlı olarak farklı bir yöntemin yürütülmesine neden olacağı anlamına gelir.

Aşağıdaki program bir soyut sınıf 'class_prgm', 2 alt sınıf (class_procedural ve class_OO) ve bir test sürücüsü sınıfı 'class_type_approach' içerir. Bu uygulamada, 'start' sınıf yöntemi, programlama türünü ve yaklaşımını görüntülememize izin verir. 'start' yönteminin imzasına yakından bakarsanız, class_prgm türünde bir içe aktarma parametresi aldığını göreceksiniz. Ancak, Start-Of-Selection olayında, bu yöntem çalışma zamanında class_procedural ve class_OO türündeki nesnelerle çağrıldı.

Örnek

```
Report ZPolymorphism1.
CLASS class_prgm Definition Abstract.
PUBLIC Section.
Methods: prgm_type Abstract,
approach1 Abstract.
ENDCLASS.

CLASS class_procedural Definition
Inheriting From class_prgm.
PUBLIC Section.
Methods: prgm_type Redefinition,
approach1 Redefinition.
ENDCLASS.

CLASS class_procedural Implementation.
Method prgm_type.
Write: 'Procedural programming'.

EndMethod. Method approach1.
Write: 'top-down approach'.

EndMethod. ENDCLASS.
CLASS class_OO Definition
Inheriting From class_prgm.
PUBLIC Section.
Methods: prgm_type Redefinition,
approach1 Redefinition.
ENDCLASS.

CLASS class_OO Implementation.
Method prgm_type.
```

```
Write: 'Object oriented programming'.
EndMethod.

Method approach1.
Write: 'bottom-up approach'.
EndMethod.
ENDCLASS.

CLASS class_type_approach Definition.
PUBLIC Section.
CLASS-METHODS:
start Importing class1_prgm
Type Ref To class_prgm.
ENDCLASS.

CLASS class_type_approach IMPLEMENTATION.
Method start.
CALL Method class1_prgm→prgm_type.
Write: 'follows'.

CALL Method class1_prgm→approach1.
EndMethod.
ENDCLASS.

Start-Of-Selection.
Data: class_1 Type Ref To class_procedural,
class_2 Type Ref To class_OO.

Create Object class_1.
Create Object class_2.
CALL Method class_type_approach⇒start
Exporting

class1_prgm = class_1.
New-Line.
CALL Method class_type_approach⇒start
Exporting
class1_prgm = class_2.
```

Yukarıdaki kod aşağıdaki çıktıyı üretir -

```
Procedural programming follows top-down approach
Object oriented programming follows bottom-up approach
```

ABAP çalışma zamanı ortamı, sınıf1_prgm içe aktarma parametresinin atanması sırasında örtük bir daraltma ataması gerçekleştirir. Bu özellik, 'start' yönteminin genel olarak uygulanmasına yardımcı olur. Bir nesne referans değişkeni ile ilişkili dinamik tip bilgisi, ABAP çalışma zamanı ortamının, nesne referans değişkeni tarafından işaret edilen nesnede tanımlanan uygulama ile bir yöntem çağrısını dinamik olarak bağlamasına izin verir. Örneğin, 'class_type_approach' sınıfındaki 'start' yöntemi için 'class1_prgm' içe aktarma parametresi, kendi başına hiçbir zaman somutlaştırılamayan soyut bir türe atıfta bulunur.

Yöntem, class_procedural veya class_OO gibi somut bir alt sınıf uygulamasıyla her çağrıldığında, class1_prgm referans parametresinin dinamik türü, bu somut türlerden birine bağlanır. Bu nedenle, 'prgm_type' ve 'yaklaşım1' yöntemlerine yapılan çağrılar, 'class_prgm' sınıfında sağlanan tanımsız soyut uygulamalar yerine class_procedural veya class_OO alt sınıflarında sağlanan uygulamalara atıfta bulunur.

SAP ABAP - Kapsülleme

Kapsülleme, verileri ve verileri işleyen işlevleri birbirine bağlayan ve hem dış müdahalelerden hem de kötüye kullanımdan koruyan bir Nesne Yönelimli Programlama (OOP) konseptidir. Veri kapsülleme, önemli OOP veri gizleme kavramına yol açtı. Kapsülleme, verileri ve bunları kullanan işlevleri gruplandırma mekanizmasıdır ve veri soyutlama, yalnızca arabirimleri açığa çıkarma ve uygulama ayrıntılarını kullanıcıdan gizleme mekanizmasıdır.

ABAP, sınıf adı verilen kullanıcı tanımlı türlerin oluşturulması yoluyla kapsülleme ve veri gizleme özelliklerini destekler. Daha önce tartışıldığı gibi, bir sınıf özel, korumalı ve genel üyeler içerebilir. Varsayılan olarak, bir sınıfta tanımlanan tüm öğeler özeldir.

Arayüz ile Kapsülleme

Kapsülleme aslında bir öznitelik ve yöntemin farklı sınıflarda değiştirilebileceği anlamına gelir. Bu nedenle veri ve yöntem, ayrı bir sınıfa gizlenebilecek farklı biçim ve mantığa sahip olabilir.

Arayüze göre kapsüllemeyi düşünelim. Arayüz, farklı sınıflarda farklı işlevlere sahip bir yöntem oluşturmamız gerektiğinde kullanılır. Burada yöntemin adının değiştirilmesi gerekmez. Aynı yöntemin farklı sınıf uygulamalarında uygulanması gerekecektir.

Örnek

Aşağıdaki program bir Arayüz inter_1 içerir. Özniteliği ve bir yöntem yöntemi1 ilan ettik. Ayrıca Class1 ve Class2 gibi iki sınıf tanımladık. Bu nedenle, her iki sınıf uygulamasında da 'method1' yöntemini uygulamamız gerekiyor. 'method1' yöntemini farklı sınıflarda farklı şekilde uyguladık. Seçim başlangıcında, iki sınıf için iki nesne Object1 ve Object2 yaratıyoruz. Ardından, ayrı sınıflarda bildirilen işlevi elde etmek için yöntemi farklı nesnelerle çağırırız.

```
Report ZEncap1.
Interface inter_1.
  Data text1 Type char35.
  Methods method1.
EndInterface.

CLASS Class1 Definition.
  PUBLIC Section.
    Interfaces inter_1.
ENDCLASS.

CLASS Class2 Definition.
  PUBLIC Section.
    Interfaces inter_1.
```

```
ENDCLASS.

CLASS Class1 Implementation.
  Method inter_1~method1.
    inter_1~text1 = 'Class 1 Interface method'.
    Write / inter_1~text1.
  EndMethod.
ENDCLASS.

CLASS Class2 Implementation.
  Method inter_1~method1.
    inter_1~text1 = 'Class 2 Interface method'.
    Write / inter_1~text1.
  EndMethod.
ENDCLASS.

Start-Of-Selection.
  Data: Object1 Type Ref To Class1,
    Object2 Type Ref To Class2.

  Create Object: Object1, Object2.
  CALL Method: Object1→inter_1~method1,
           Object2→inter_1~method1.
```

Yukarıdaki kod aşağıdaki çıktıyı üretir -

```
Class 1 Interface method
Class 2 Interface method
```

Kapsüllenmiş sınıfların dış dünyaya çok fazla bağımlılığı yoktur. Ayrıca, harici müşterilerle olan etkileşimleri, sabit bir genel arayüz aracılığıyla kontrol edilir. Yani, kapsüllenmiş bir sınıf ve istemcileri gevşek bir şekilde bağlanmıştır. Çoğunlukla, iyi tanımlanmış arayüzlere sahip sınıflar başka bir bağlama eklenebilir. Doğru tasarlandıklarında, kapsüllenmiş sınıflar yeniden kullanılabilir yazılım varlıkları haline gelir.

Strateji Tasarlama

Çoğumuz, gerçekten ifşa etmemiz gerekmedikçe, sınıf üyelerini varsayılan olarak özel yapmayı acı deneyimler yoluyla öğrendik. Bu sadece iyi bir kapsülleme. Bu bilgelik en sık veri üyelerine uygulanır ve aynı zamanda tüm üyelere eşit olarak uygulanır.

SAP ABAP - Arayüzler

ABAP'deki sınıflara benzer şekilde, arabirimler nesneler için veri türleri olarak işlev görür. Arayüzlerin bileşenleri, sınıfların bileşenleri ile aynıdır. Sınıfların bildiriminden farklı olarak, bir arabirimin bildirimi görünürlük bölümlerini içermez. Bunun nedeni, bir arabirimin bildiriminde tanımlanan bileşenlerin her zaman sınıfların genel görünürlük bölümüne entegre edilmiş olmasıdır.

Arabirimler, iki benzer sınıfın aynı ada sahip bir yöntemi olduğunda, ancak işlevleri birbirinden farklı olduğunda kullanılır. Arayüzler sınıflara benzer görünebilir, ancak bir

arayüzde tanımlanan işlevler, o sınıfın kapsamını genişletmek için bir sınıfta uygulanır. Kalıtım özelliği ile birlikte arabirimler, polimorfizm için bir temel sağlar. Bunun nedeni, bir arabirimde tanımlanan bir yöntemin farklı sınıflarda farklı davranabilmesidir.

Bir arayüz oluşturmak için genel format aşağıdadır -

```
INTERFACE <intf_name>.
DATA.....
CLASS-DATA.....
METHODS.....
CLASS-METHODS.....
ENDINTERFACE.
```

Bu sözdiziminde <intf_name>, bir arabirimin adını temsil eder. DATA ve CLASSDATA deyimleri, sırasıyla arayüzün örneğini ve statik özniteliklerini tanımlamak için kullanılabilir. YÖNTEMLER ve SINIF YÖNTEMLERİ deyimleri, sırasıyla arabirimin örneğini ve statik yöntemlerini tanımlamak için kullanılabilir. Bir arabirimin tanımı, uygulama sınıfını içermediğinden, bir arabirimin bildiriminde DEFINITION yan tümcesinin eklenmesi gerekli değildir.

Not − Bir arabirimin tüm yöntemleri soyuttur. Parametre arabirimleri de dahil olmak üzere tamamen bildirilirler, ancak arabirimde uygulanmazlar. Bir arabirim kullanmak isteyen tüm sınıflar, arabirimin tüm yöntemlerini uygulamalıdır. Aksi takdirde, sınıf soyut bir sınıf olur.

Sınıfın uygulama bölümünde aşağıdaki sözdizimini kullanıyoruz -

```
INTERFACE <intf_name>.
```

Bu sözdiziminde <intf_name>, bir arabirimin adını temsil eder. Bu sözdiziminin sınıfın genel bölümünde kullanılması gerektiğini unutmayın.

Aşağıdaki sözdizimi, bir sınıfın uygulanması içinde bir arabirimin yöntemlerini uygulamak için kullanılır -

```
METHOD <intf_name~method_m>.
<statements>.
ENDMETHOD.
```

Bu sözdiziminde, <intf_name~method_m>, <intf_name> arabiriminin bir yönteminin tam olarak bildirilen adını temsil eder.

Örnek

```
Report ZINTERFACE1.
INTERFACE my_interface1.
Methods msg.
ENDINTERFACE.

CLASS num_counter Definition.
PUBLIC Section.
INTERFACES my_interface1.
Methods add_number.
PRIVATE Section.
```

```
Data num Type I.
ENDCLASS.

CLASS num_counter Implementation.
Method my_interface1~msg.
Write: / 'The number is', num.
EndMethod.

Method add_number.
ADD 7 TO num.
EndMethod.
ENDCLASS.

CLASS drive1 Definition.
PUBLIC Section.
INTERFACES my_interface1.
Methods speed1.
PRIVATE Section.
Data wheel1 Type I.
ENDCLASS.

CLASS drive1 Implementation.
Method my_interface1~msg.
Write: / 'Total number of wheels is', wheel1.
EndMethod.

Method speed1.
Add 4 To wheel1.
EndMethod.
ENDCLASS.

Start-Of-Selection.
Data object1 Type Ref To num_counter.
Create Object object1.

CALL Method object1→add_number.
CALL Method object1→my_interface1~msg.

Data object2 Type Ref To drive1.
Create Object object2.

CALL Method object2→speed1.
CALL Method object2→my_interface1~msg.
```

Yukarıdaki kod aşağıdaki çıktıyı üretir -

```
The number is 7
Total number of wheels is 4
```

Yukarıdaki örnekte, my_interface1 'msg' yöntemini içeren bir arabirimin adıdır. Daha sonra iki sınıf, num_counter ve drive1 tanımlanır ve uygulanır. Bu sınıfların her ikisi de 'msg' yöntemini ve ayrıca add_number ve speed1 yöntemleri gibi ilgili örneklerinin davranışını tanımlayan belirli yöntemleri uygular.

Not − add_number ve speed1 yöntemleri, ilgili sınıflara özeldir.

SAP ABAP - Nesne Olayları

Olay , diğer sınıflardaki olay işleyicilerini tetiklemek için bir sınıfta tanımlanan bir dizi sonuçtur. Bir olay tetiklendiğinde, herhangi bir sayıda olay işleyici yöntemini çağırabiliriz. Bir tetikleyici ile onun işleyici yöntemi arasındaki bağlantıya aslında çalışma zamanında dinamik olarak karar verilir.

Normal bir yöntem çağrısında, çağıran bir program, bir nesnenin veya sınıfın hangi yönteminin çağrılması gerektiğini belirler. Sabit işleyici yöntemi her olay için kaydedilmediğinden, olay işleme durumunda tetiklenmesi gereken olayı işleyici yöntemi belirler.

Bir sınıfın olayı, RAISE EVENT deyimini kullanarak aynı sınıfın olay işleyici yöntemini tetikleyebilir. Bir olay için olay işleyici yöntemi, aşağıdaki sözdiziminde gösterildiği gibi FOR EVENT yan tümcesi kullanılarak aynı veya farklı sınıfta tanımlanabilir -

FOR EVENT <event_name> OF <class_name>.

Bir sınıfın yöntemlerine benzer şekilde, bir olay parametre arabirimine sahip olabilir, ancak yalnızca çıktı parametrelerine sahiptir. Çıkış parametreleri, onları giriş parametreleri olarak alan RAISE EVENT deyimi tarafından olay işleyici yöntemine iletilir. Bir olay, SET HANDLER deyimi kullanılarak bir programda dinamik olarak işleyici yöntemine bağlanır.

Bir olay tetiklendiğinde, tüm işleme sınıflarında uygun olay işleyici yöntemlerinin yürütülmesi gerekir.

Örnek

```
REPORT ZEVENT1.
CLASS CL_main DEFINITION.
PUBLIC SECTION.
DATA: num1 TYPE I.
METHODS: PRO IMPORTING num2 TYPE I.
EVENTS: CUTOFF.
ENDCLASS.

CLASS CL_eventhandler DEFINITION.
PUBLIC SECTION.
METHODS: handling_CUTOFF FOR EVENT CUTOFF OF CL_main.
ENDCLASS.

START-OF-SELECTION.
DATA: main1 TYPE REF TO CL_main.
DATA: eventhandler1 TYPE REF TO CL_eventhandler.

CREATE OBJECT main1.
CREATE OBJECT eventhandler1.

SET HANDLER eventhandler1→handling_CUTOFF FOR main1.
main1→PRO( 4 ).
```

```
CLASS CL_main IMPLEMENTATION.
METHOD PRO.
num1 = num2.
IF num2 ≥ 2.
RAISE EVENT CUTOFF.
ENDIF.
ENDMETHOD.
ENDCLASS.

CLASS CL_eventhandler IMPLEMENTATION.
METHOD handling_CUTOFF.
WRITE: 'Handling the CutOff'.
WRITE: / 'Event has been processed'.
ENDMETHOD. ENDCLASS.
```

Yukarıdaki kod aşağıdaki çıktıyı üretir -

```
Handling the CutOff
Event has been processed
```

SAP ABAP - Rapor Programlama

Rapor , verilerin organize bir yapıda sunulmasıdır. Birçok veritabanı yönetim sistemi, raporlar tasarlamanıza ve oluşturmanıza olanak tanıyan bir rapor yazıcısı içerir. SAP uygulamaları rapor oluşturmayı destekler.

WRITE deyimindeki çıktı verileri bir döngü içinde kullanılarak klasik bir rapor oluşturulur. Herhangi bir alt rapor içermezler. SAP, istemciler arasında tabloları kopyalamak için kullanılan RSCLTCOP ve örnek parametrelerini görüntülemek için kullanılan RSPARAM gibi bazı standart raporlar da sağlar.

Bu raporlar çıktı olarak tek ekrandan oluşmaktadır. Klasik bir rapor oluşturmak için BAŞLANGIÇ & SAYFA BAŞI gibi çeşitli olayları kullanabiliriz ve klasik bir raporun oluşturulması sırasında her olayın kendi önemi vardır. Bu olayların her biri belirli bir kullanıcı eylemiyle ilişkilendirilir ve yalnızca kullanıcı bu eylemi gerçekleştirdiğinde tetiklenir.

Olayları ve açıklamaları açıklayan bir tablo aşağıdadır -

S.No.	Etkinlik Açıklaması
1	**BAŞLATMA** Seçim ekranı görüntülenmeden önce tetiklendi.
2	**SEÇİM EKRANINDA** Seçim ekranında kullanıcı girişinin işlenmesinden sonra tetiklenir. Bu olay, bir programın y önce kullanıcı girişini doğrular. Kullanıcı girişi işlendikten sonra seçim ekranı aktif modda ka
3	**SEÇİM BAŞLANGICI**

	Sadece seçim ekranının işlenmesi bittikten sonra tetiklenir; yani, kullanıcı seçim ekra simgesine tıkladığında.
4	**SEÇİM SONU** START-OF-SELECTON olayındaki son ifade yürütüldükten sonra tetiklenir.
5	**SAYFA BAŞI** Verileri yeni bir sayfada görüntülemek için ilk WRITE ifadesi tarafından tetiklenir.
6	**SAYFA SONU** Metni bir raporda bir sayfanın sonunda görüntülemek için tetiklendi. Bu olayın bir rapor ol olay olduğunu ve REPORT ifadesinin LINE-COUNT yan tümcesi ile birleştirilmesi gerektiğir

Örnek

Klasik bir rapor oluşturalım. ABAP düzenleyicide bir dizi ifadeyi kullanarak standart MARA veritabanında (genel malzeme verilerini içerir) depolanan bilgileri görüntüleyeceğiz.

```
REPORT ZREPORT2
LINE-SIZE 75
LINE-COUNT 30(3)
NO STANDARD PAGE HEADING.
Tables: MARA.
TYPES: Begin of itab,

MATNR TYPE MARA-MATNR,
MBRSH TYPE MARA-MBRSH,
MEINS TYPE MARA-MEINS,
MTART TYPE MARA-MTART,

End of itab.

DATA: wa_ma TYPE itab,
    it_ma TYPE STANDARD TABLE OF itab.

SELECT-OPTIONS: MATS FOR MARA-MATNR OBLIGATORY.
INITIALIZATION.
MATS-LOW = '1'.
MATS-HIGH = '500'.

APPEND MATS.
AT SELECTION-SCREEN. .
IF MATS-LOW = ' '.
MESSAGE I000(ZKMESSAGE).
```

```
ELSEIF MATS-HIGH = ' '.
MESSAGE I001(ZKMESSAGE).
ENDIF.

TOP-OF-PAGE.
WRITE:/ 'CLASSICAL REPORT CONTAINING GENERAL MATERIAL DATA
FROM THE TABLE MARA' COLOR 7.
ULINE.
WRITE:/ 'MATERIAL' COLOR 1,

24 'INDUSTRY' COLOR 2,
38 'UNITS' COLOR 3,
53 'MATERIAL TYPE' COLOR 4.
ULINE.
END-OF-PAGE.

START-OF-SELECTION.
SELECT MATNR MBRSH MEINS MTART FROM MARA
INTO TABLE it_ma WHERE MATNR IN MATS.
LOOP AT it_ma into wa_ma.
WRITE:/  wa_ma-MATNR,

25 wa_ma-MBRSH,
40 wa_ma-MEINS,
55 wa_ma-MTART.
ENDLOOP.
END-OF-SELECTION.

ULINE.
WRITE:/ 'CLASSICAL REPORT HAS BEEN CREATED' COLOR 7.
ULINE.
SKIP.
```

Yukarıdaki kod, MARA standart tablosundan genel malzeme verilerini içeren aşağıdaki çıktıyı üretir -

CLASSICAL REPORT CONTAINING GENERAL MATERIAL DATA FROM THE TABLE MARA

MATERIAL	INDUSTRY	UNITS	MATERIAL TYPE
23	1	EA	ROH
38	M	PC	HALB
43	1	HR	HAWA
58	M	PC	HIBE
59	M	PC	HIBE
68	M	PC	FHMI
78	M	PC	DIEN
88	M	PC	FERT
89	M	PC	FERT
98	M	PC	HALB
170	M	PC	NLAG
178	M	PC	NLAG
188	M	PC	NLAG
288	M	PC	HALB
358	M	PC	HAWA
359	M	PC	HAWA

CLASSICAL REPORT HAS BEEN CREATED

SAP ABAP - Diyalog Programlama

Diyalog programlama, çoklu nesnelerin geliştirilmesi ile ilgilenir. Tüm bu nesneler hiyerarşik olarak ana programa bağlıdır ve bir sırayla yürütülürler. Diyalog programı geliştirme, ABAP tezgahındaki araçlardan yararlanır. Bunlar, standart SAP uygulama geliştirmede kullanılan araçlarla aynıdır.

İşte diyalog programlarının ana bileşenleri:

- Ekranlar
- Modül havuzları
- alt programlar
- Menüler
- işlemler

Araç Seti

Central Component	Tool	Transaction
All Components	Object Browser	SE80 Tools > ABAP Workbench > Object Browser
Screen	Screen Painter	SE51 Tools > ABAP Workbench > Screen Painter
ABAP/4 Module Pool	ABAP/4 Editor	SE38 Tools > ABAP Workbench > ABAP/4 Editor
Dictionary Objects (tables, fields, etc.)	ABAP/4 Dictionary	SE11 Tools > ABAP Workbench > ABAP/4 Dictionary
Menu	Menu Painter	SE41 Tools > ABAP Workbench > Menu Painter
Transaction	Maintain Transaction	SE93 Tools > ABAP Workbench > Development > Other Tools > Transactions

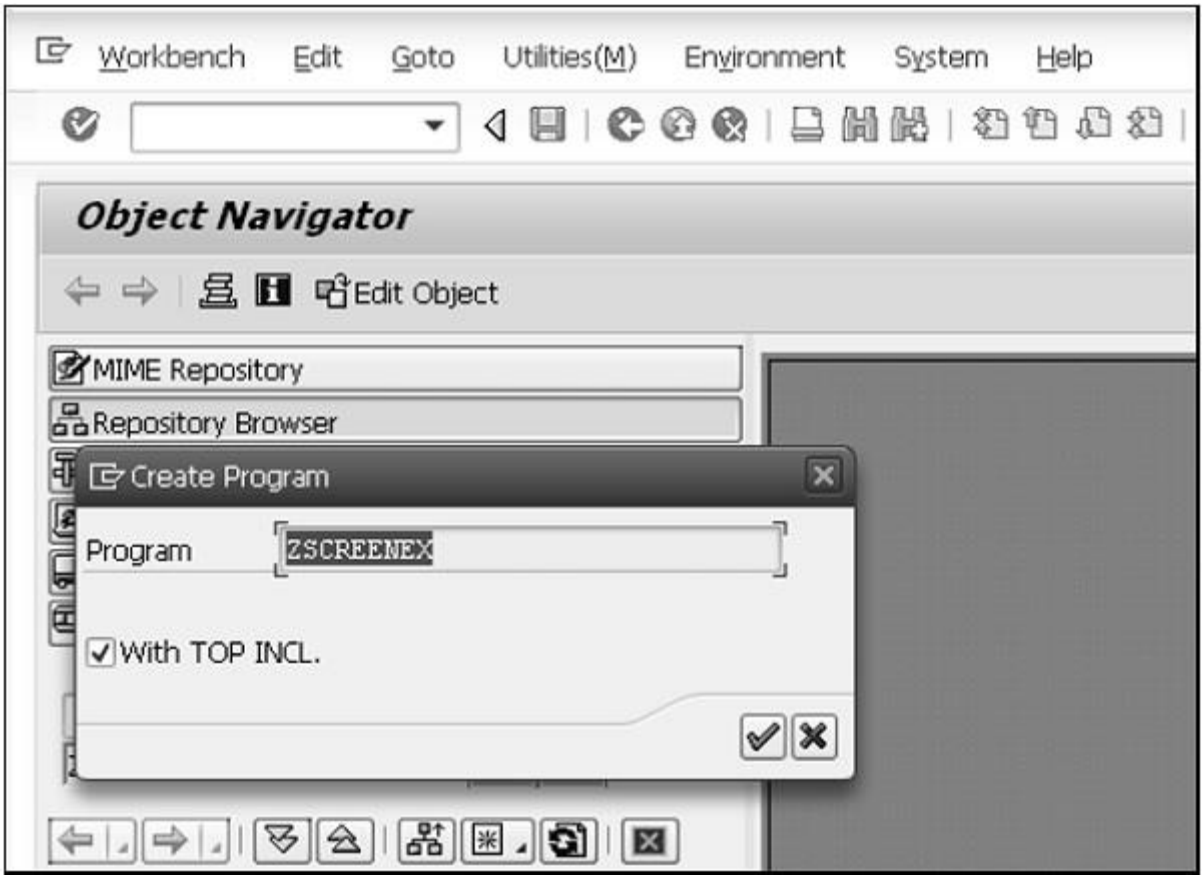
İletişim programları, nesne tarayıcısı (işlem: SE80) tarafından geliştirilmelidir, böylece tüm nesneler, her bir nesneyi açıkça işaret etmek zorunda kalmadan ana programa bağlanır. Gelişmiş navigasyon teknikleri, bir nesneden diğerine geçme sürecini geliştirir.

Ekranlar, ekran nitelikleri, ekran düzeni, alanlar ve akış mantığından oluşur. Modül havuzu, diyalog programının içerdiği programların içine yerleştirilmiş modülerleştirilmiş söz diziminden oluşur. Bu modüller, diyalog işlemcisi tarafından işlenen akış mantığı tarafından çağrılabilir.

Yeni Bir Diyalog Programı Oluşturma

Adım 1 − SE80 işleminde, açılır menüden 'Program'ı seçin ve özel SAP programınız için 'ZSCREENEX' olarak bir Z adı girin.

Adım 2 − Enter'a basın, 'ÜST DAHİLDE' seçeneğini seçin ve 'Evet' düğmesini tıklayın.



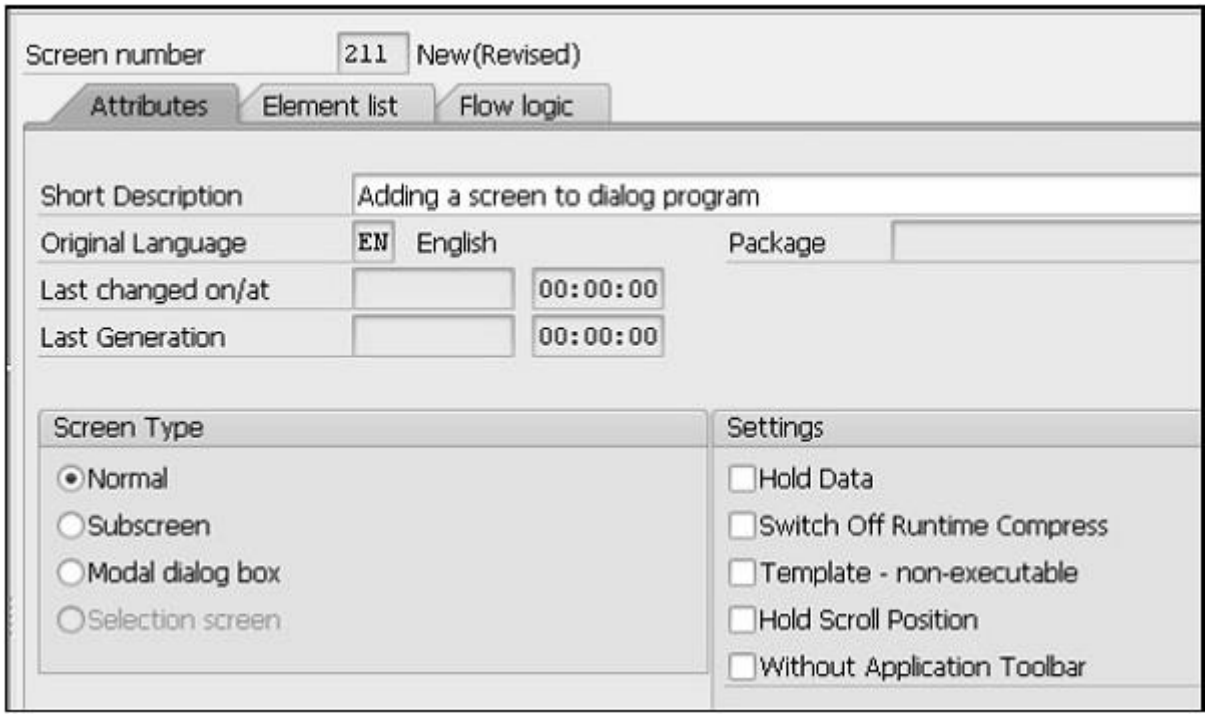
Adım 3 − En üstünüz için 'ZSCRTOP' olarak bir ad girin ve yeşil onay işaretine tıklayın.

Adım 4 − Nitelikler ekranında bir başlık girin ve kaydet düğmesine tıklayın.

Diyalog Programına Ekran Ekleme

Adım 1 − Programa bir ekran eklemek için program adına sağ tıklayın ve Oluştur → Ekran seçeneklerini seçin.

Adım 2 − '0211' olarak bir ekran numarası girin ve yeşil onay işaretine tıklayın.

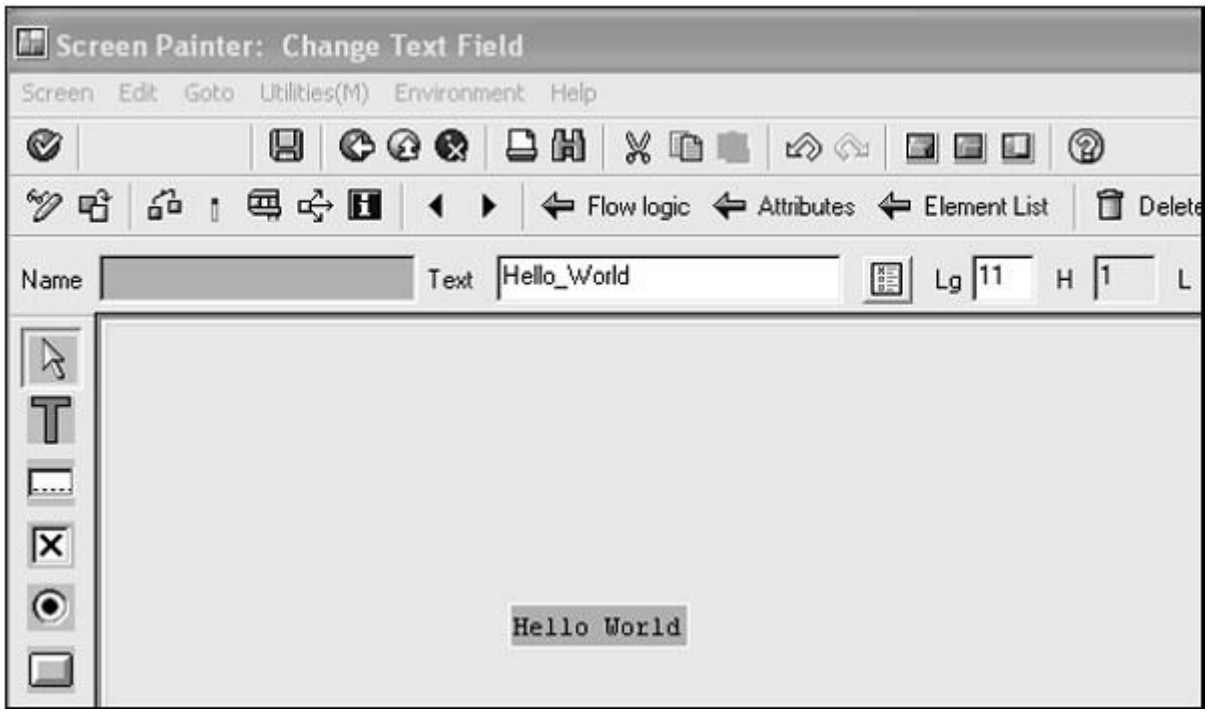


Adım 3 − Bir sonraki ekranda kısa bir başlık girin, normal ekran tipine ayarlayın ve üst uygulama araç çubuğundaki kaydet düğmesine tıklayın.

Ekran Düzeni ve 'Merhaba Dünya' Metni Ekleme

Adım 1 − Uygulama araç çubuğundaki düzen düğmesine tıklayın ve Ekran Boyacısı penceresi belirir.

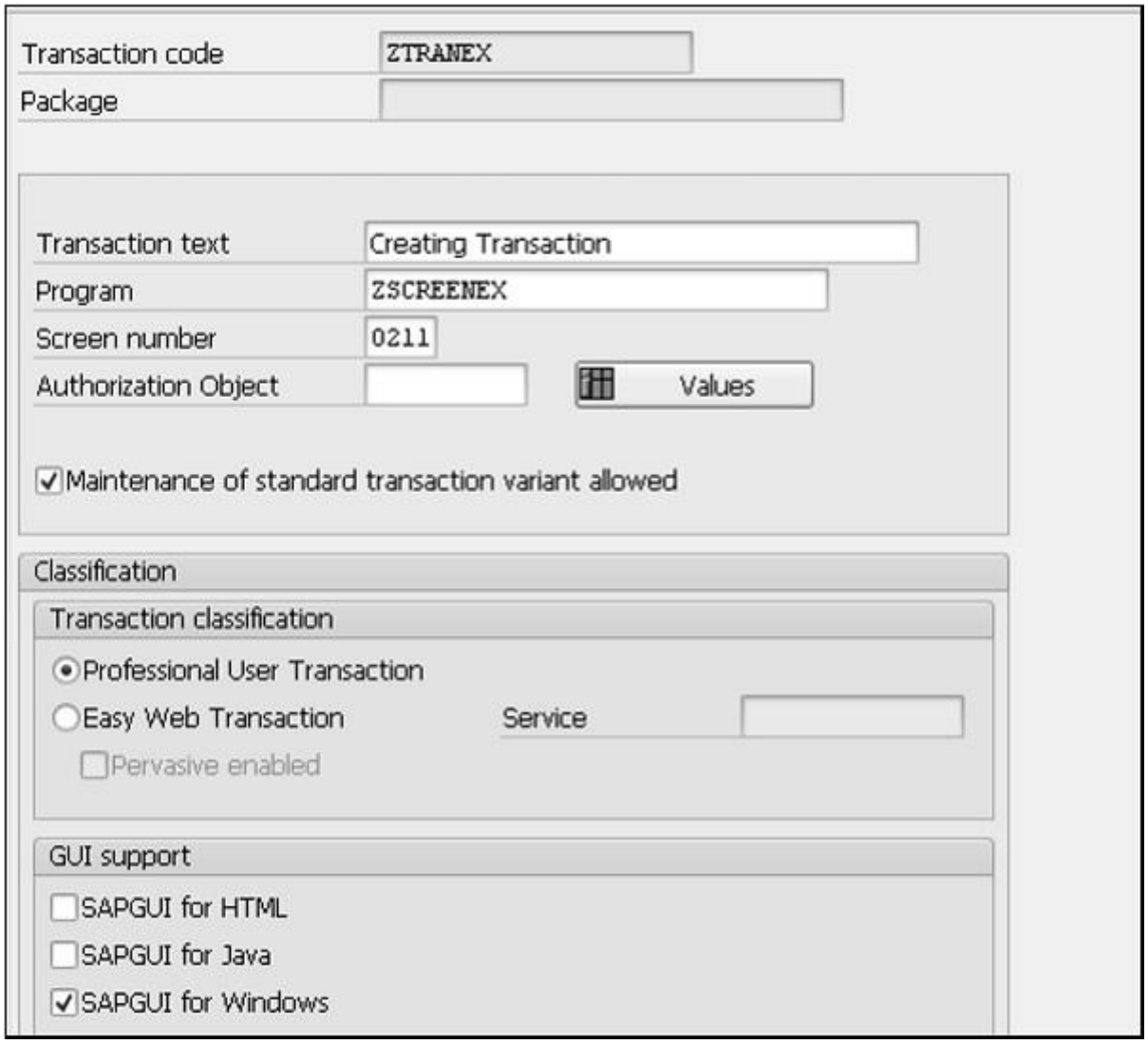
Adım 2 − Bir Metin Alanı ekleyin ve "Merhaba Dünya" gibi bir metin girin.



Adım 3 − Ekranı kaydedin ve etkinleştirin.

İşlem Oluşturma

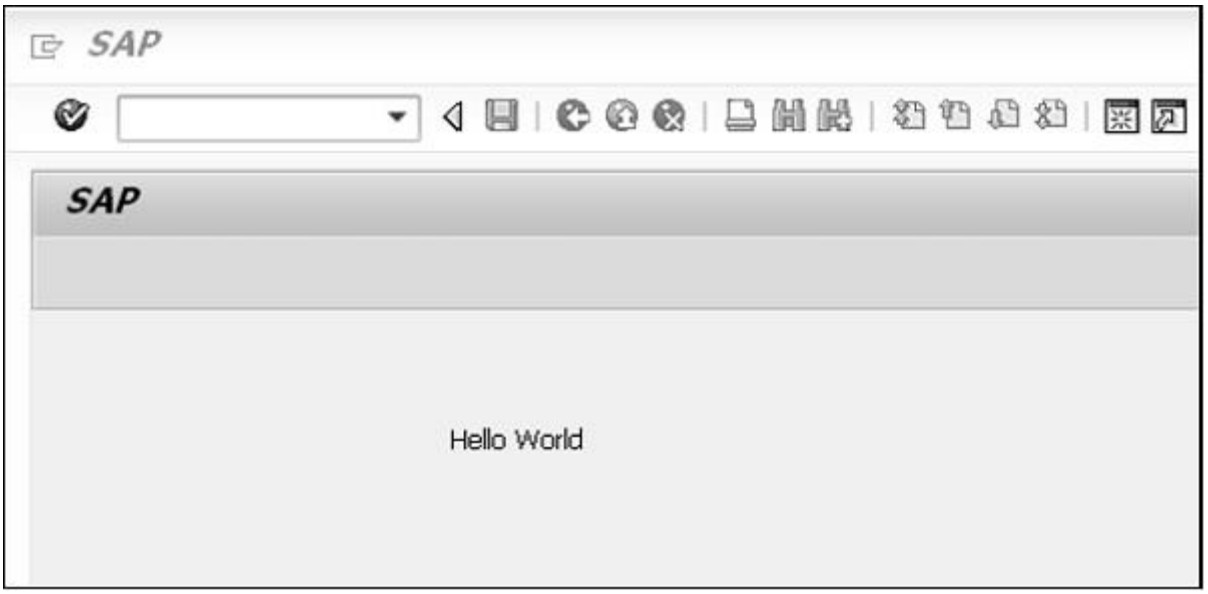
Adım 1 − Programınız için bir işlem kodu oluşturmak için, program adına sağ tıklayın ve Oluştur → İşlem seçeneğini seçin ve 'ZTRANEX' olarak bir işlem kodu girin.



Adım 2 − Yeni oluşturduğunuz işlem metnini, programı ve ekranı (ZSCREENEX & 0211) girin ve 'GUI desteği' bölümünde 'Windows için SAPGUI' onay kutusunu işaretleyin.

Programın Yürütülmesi

Her şeyi kaydedin ve etkinleştirin. Programı çalıştırabilirsiniz. Program çalışırken girdiğiniz metin aşağıdaki ekran görüntüsünde gösterildiği gibi ekrana gelir.



SAP ABAP - Akıllı Formlar

SAP Smart Forms aracı, belgeleri yazdırmak ve göndermek için kullanılabilir. Bu araç, İnternet için formlar, PDF dosyaları, e-postalar ve belgeler geliştirmek için kullanışlıdır. Araç, bir formun düzenini ve mantığını oluşturmak ve sürdürmek için bir arabirim sağlar. SAP ayrıca Müşteri İlişkileri Yönetimi (CRM), Satış ve Dağıtım (SD), Finansal Muhasebe (FI) ve İnsan Kaynakları'nda (İK) kullanılanlar gibi iş süreçleri için çeşitli formlar sunar.

Araç, herhangi bir programlama aracı kullanmak yerine basit grafik araçları kullanarak formları değiştirmenize olanak tanır. Bu, programlama bilgisi olmayan bir kullanıcının bu formları bir iş süreci için verilerle zahmetsizce yapılandırabileceği anlamına gelir.

Akıllı Formda, veriler statik ve dinamik tablolardan alınır. Tablo başlığı ve alt toplam, tetiklenen olaylar tarafından belirlenir ve veriler daha sonra nihai çıktıdan önce sıralanır. Akıllı Form, formun bir parçası veya arka plan olarak görüntülenebilen grafikleri birleştirmenize olanak tanır. Bir formun çıktısını alırken gerekirse bir arka plan grafiğini de bastırabilirsiniz.

SAP sisteminde bulunan bazı standart Akıllı Form örnekleri aşağıdaki gibidir:

- SF_EXAMPLE_01, bir müşteri için uçuş rezervasyonu için tablo çıktısı olan bir faturayı temsil eder.
- SF_EXAMPLE_02, SF_EXAMPLE_01'e benzer, ancak ara toplamları olan bir faturayı temsil eder.
- SF_EXAMPLE_03, SF_EXAMPLE_02'ye benzer, ancak bir uygulama programında birkaç müşterinin seçilebileceği bir fatura belirtir.

Form Oluşturma

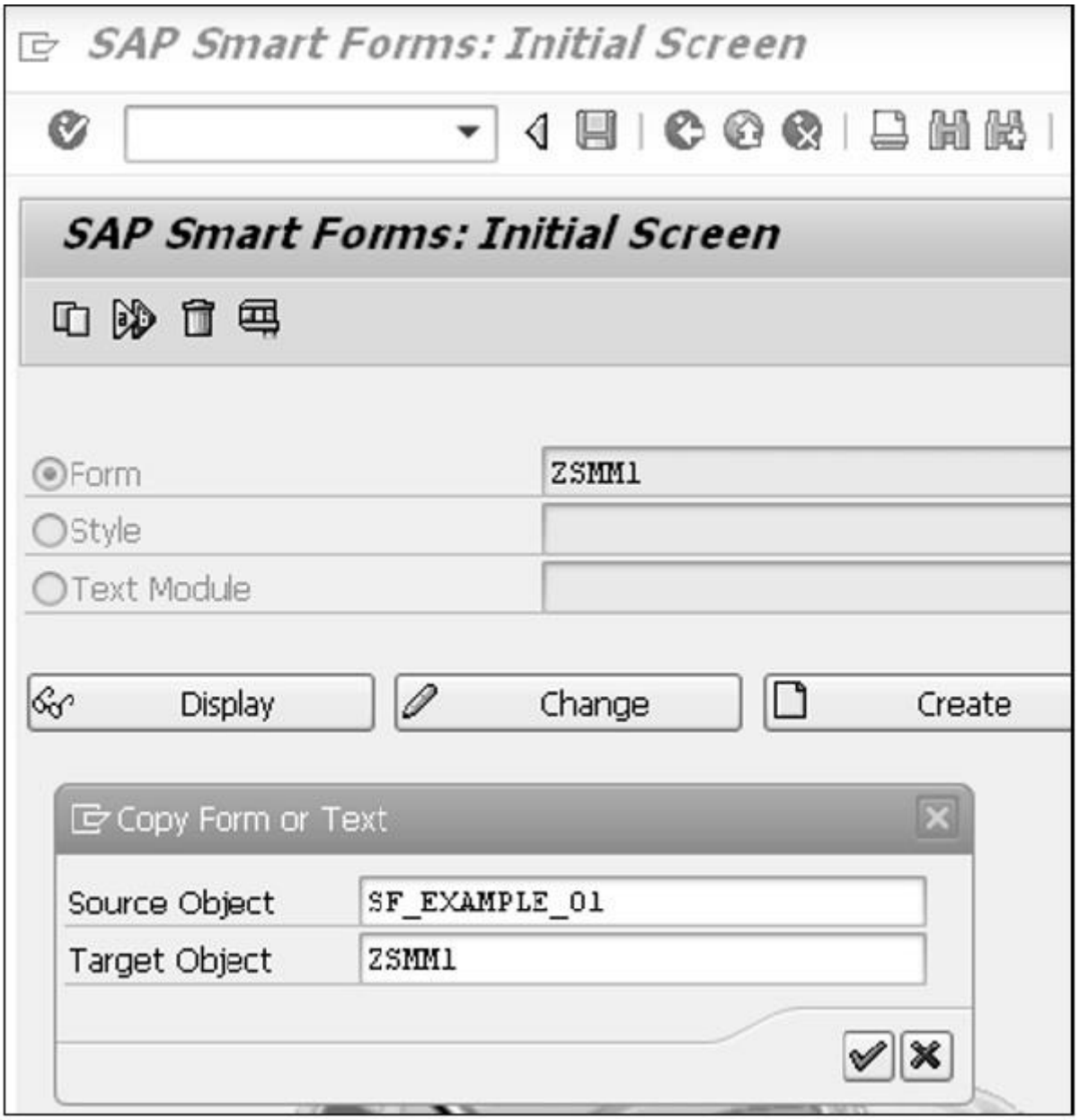
SAP Smart Forms aracını kullanarak bir form oluşturalım. Ayrıca bu eğitimde Akıllı Form'a nasıl düğüm ekleyeceğinizi ve formu nasıl test edeceğinizi öğreneceksiniz. Burada SF_EXAMPLE_01 formunun bir kopyasını oluşturmaya

başlıyoruz. SF_EXAMPLE_01 formu, SAP sisteminde bulunan standart bir Akıllı Formdur.

Adım 1 − Akıllı Form Oluşturucu, Akıllı Form oluşturmak için kullanılan ana arayüzdür. SAP Smart Forms'un ilk ekranında bulunur. SAP Smart Forms'un ilk ekranını açmak için Komut alanına 'SMARTFORMS' işlem kodunu yazmamız gerekiyor. Bu ekranda, Form alanına SF_EXAMPLE_01 form adını girin.

Adım 2 − Akıllı Formlar → Kopyala'yı seçin veya Formu veya Metni Kopyala iletişim kutusunu açmak için Kopyala simgesini tıklayın.

Adım 3 − Hedef Nesne alanına yeni form için bir ad girin. İsim Y veya Z harfi ile başlamalıdır. Bu durumda formun adı 'ZSMM1'dir.

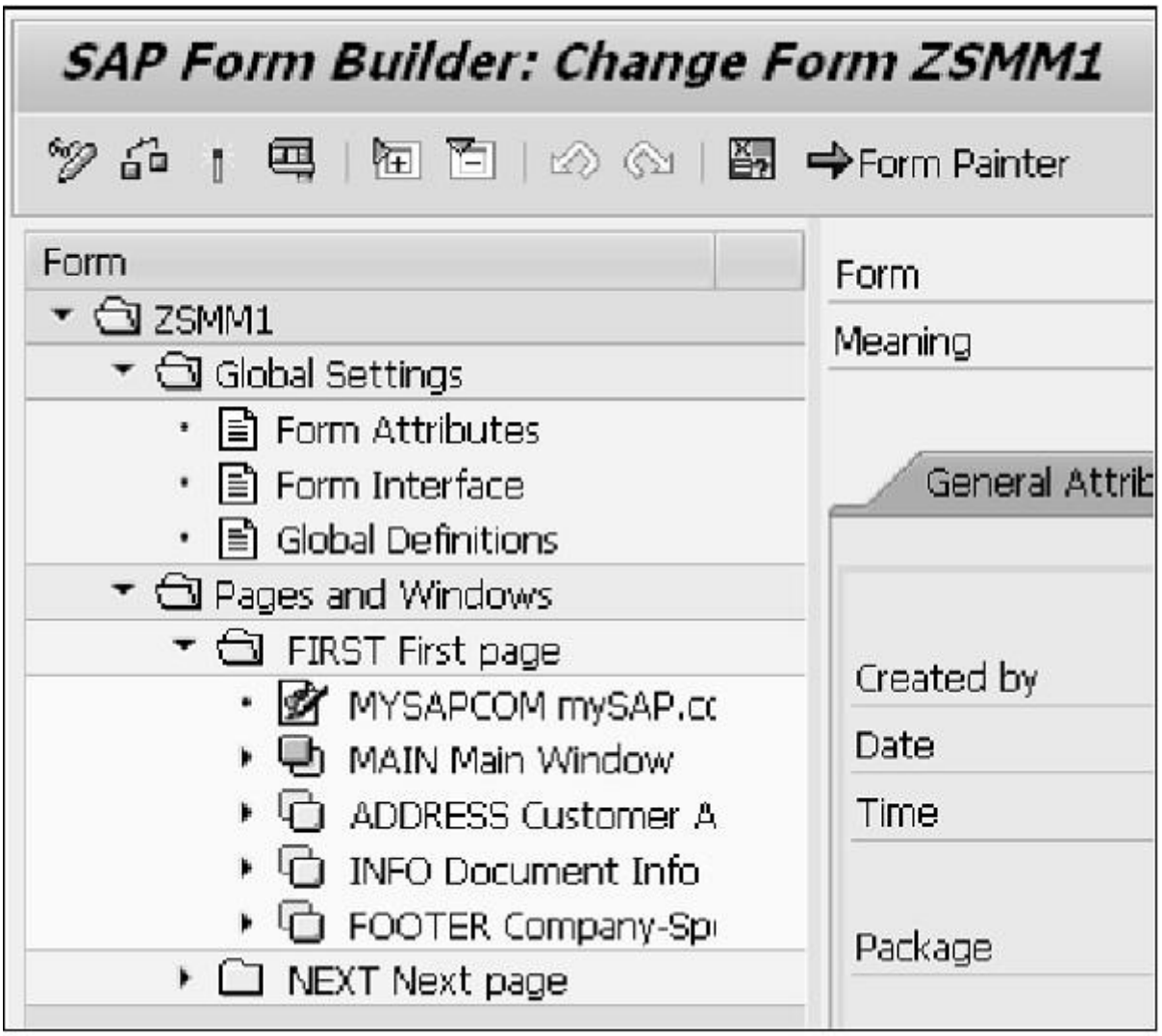


Adım 4 − ZSMM1 formunun önceden tanımlanmış SF_EXAMPLE_01 formunun bir kopyası olarak oluşturulması için Devam simgesine tıklayın veya Form veya Metin Kopyala iletişim kutusunda ENTER tuşuna basın.

Adım 5 − Kaydet simgesine tıklayın. Formun adı, SAP Smart Forms'un ilk ekranındaki Form alanında görüntülenir.

Adım 6 − SAP Smart Forms'un ilk ekranında Oluştur düğmesine tıklayın. ZSMM1 formu, Form Oluşturucu'da görünür.

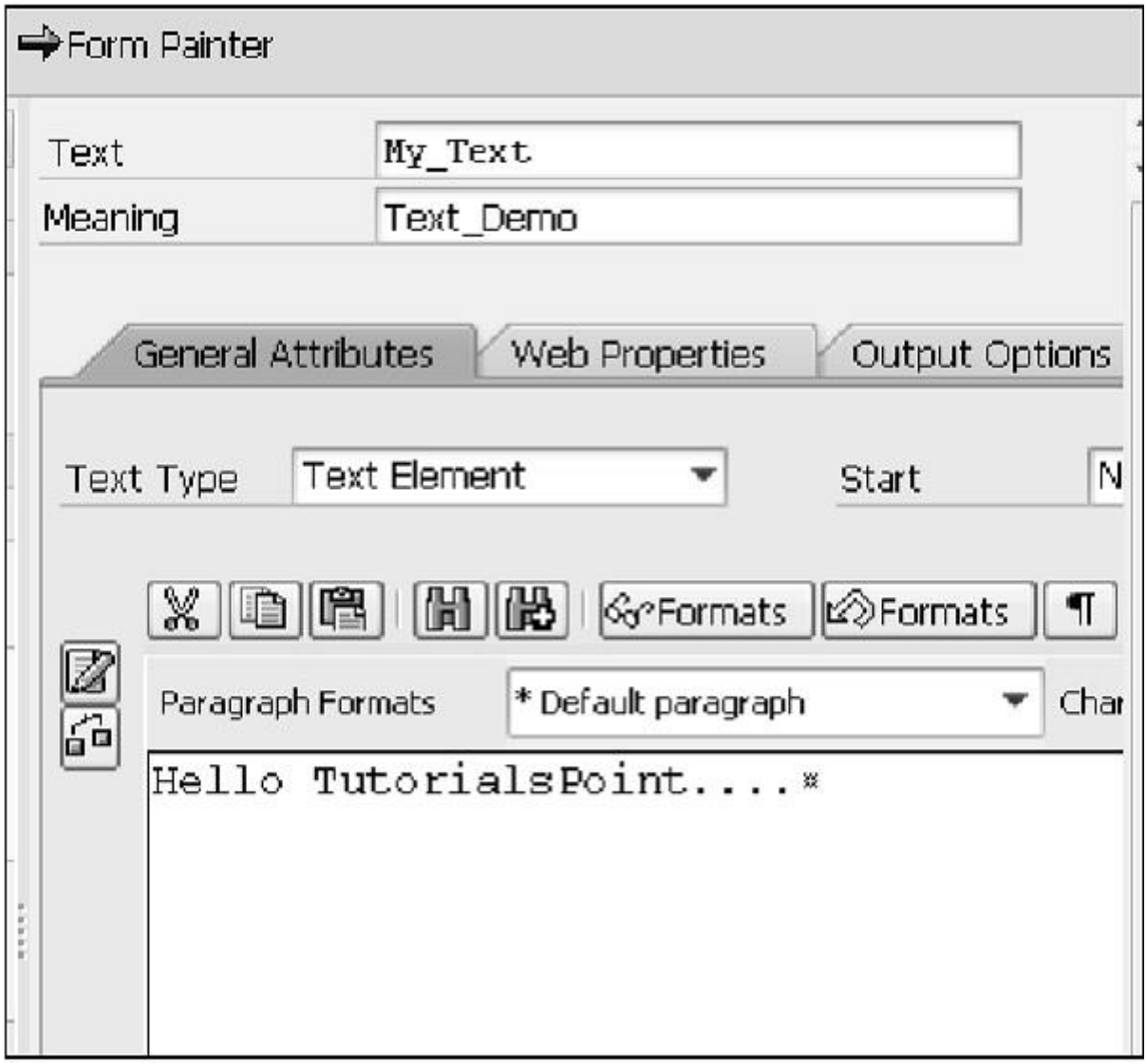
Adım 7 − İlk taslak sayfası bir ANA pencere ile oluşturulur. Yeni formun tüm bileşenleri, SF_EXAMPLE_01 önceden tanımlanmış formu temel alır. İçeriğini görüntülemek için Gezinme menüsündeki bir düğümü tıklamanız yeterlidir.



Formda Metin Düğümü Oluşturma

Adım 1 − SAP Form Builder ekranının değişiklik modunda bir form açın ve İlk Sayfa düğümündeki Ana Pencere seçeneğine sağ tıklayın ve içerik menüsünden Oluştur → Metin'i seçin.

Adım 2 − Metin alanındaki metni 'My_Text' ve Anlam alanındaki metni 'Text_Demo' olarak değiştirin. Aşağıdaki anlık görüntüde gösterildiği gibi Form Builder'ın orta çerçevesindeki metin düzenleme kutusuna 'Hello TutorialsPoint.....' metnini girin -



Adım 3 − Düğümü kaydetmek için Kaydet düğmesine tıklayın.

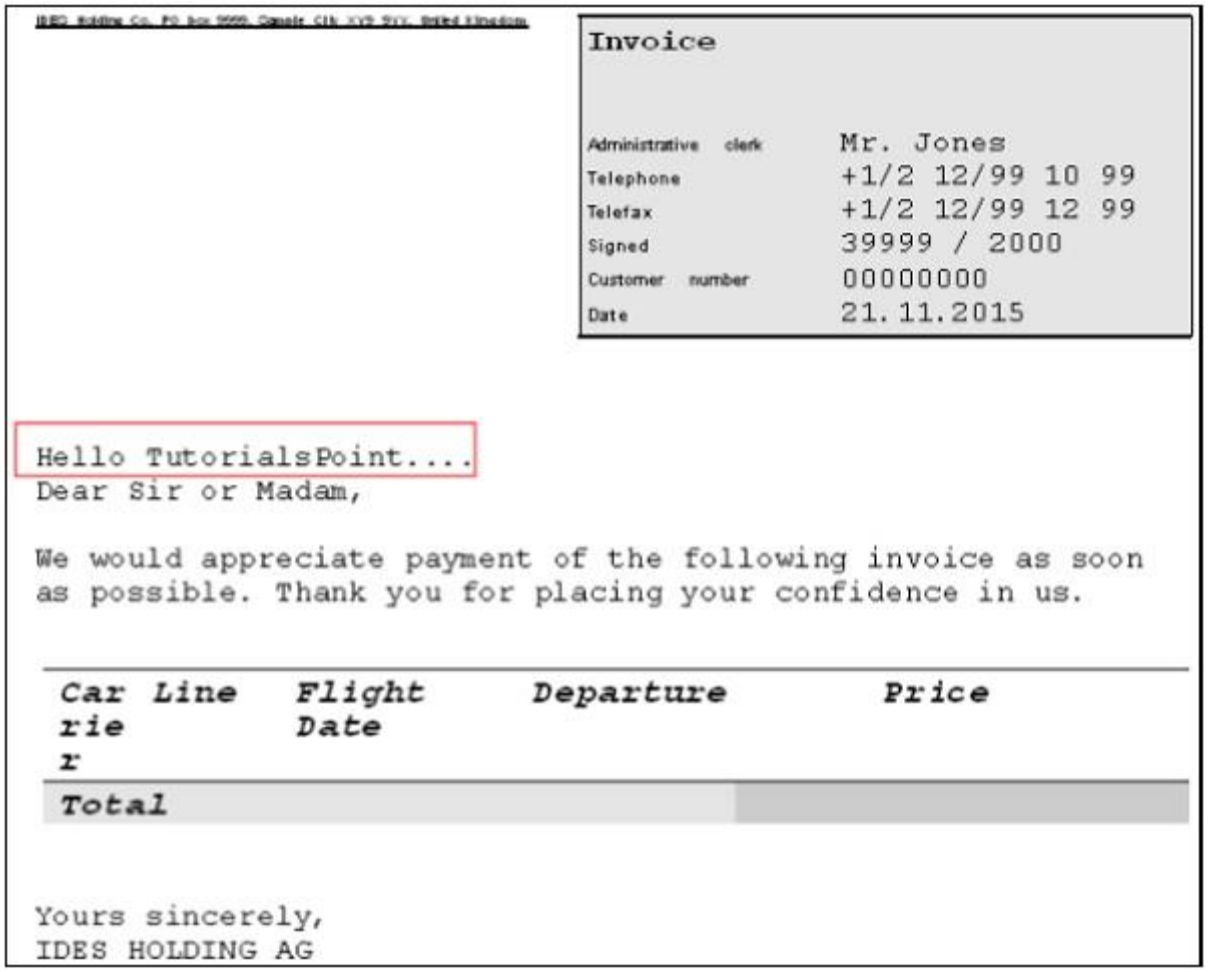
Adım 4 − Sırasıyla Etkinleştir ve Test Et simgelerine tıklayarak düğümü etkinleştirin ve test edin. Function Builder'ın başlangıç ekranı belirir.

Adım 5 − Etkinleştir ve Yürüt simgelerine tıklayarak işlev modülünü etkinleştirin ve test edin. İşlev modülünün parametreleri, Function Builder'ın ilk ekranında görüntülenir.

Adım 6 − Çalıştır simgesine tıklayarak fonksiyon modülünü çalıştırın. Yazdır iletişim kutusu görünür.

Adım 7 − Çıkış aygıtını 'LP01' olarak belirleyin ve Baskı önizleme düğmesine tıklayın.

Yukarıdaki adımlar aşağıdaki çıktıyı üretecektir -



SAP ABAP - SAPscript'ler

SAP sisteminin SAPscript aracı, faturalar ve satınalma siparişleri gibi iş formlarını oluşturmak ve yönetmek için kullanılabilir. SAPscript aracı, bir iş formunun tasarımını büyük ölçüde basitleştiren çok sayıda şablon sağlar.

SAP sistemi, SAP standart istemcisi (genellikle istemci 000 olarak) ile birlikte verilen standart SAPscript formlarıyla birlikte gelir. Aşağıda, müşteri 000 ile teslim edilen standart SAPscript formlarının birkaç örneği verilmiştir -

S.No.	Form Adı ve Açıklaması
1	**RVORDER01** Satış Siparişi Onay Formu
2	**RVDELNOT** Paket listesi

3	**RVINVOICE01** Fatura
4	**MEDRUCK** Satın alma emri
5	**F110_PRENUM_CHCK** Ön Numaralı Çek

Bir SAPscript formunun yapısı 2 ana bileşenden oluşur -

İçerik − Bu, metin (iş verileri) veya grafik (şirket logosu) olabilir.

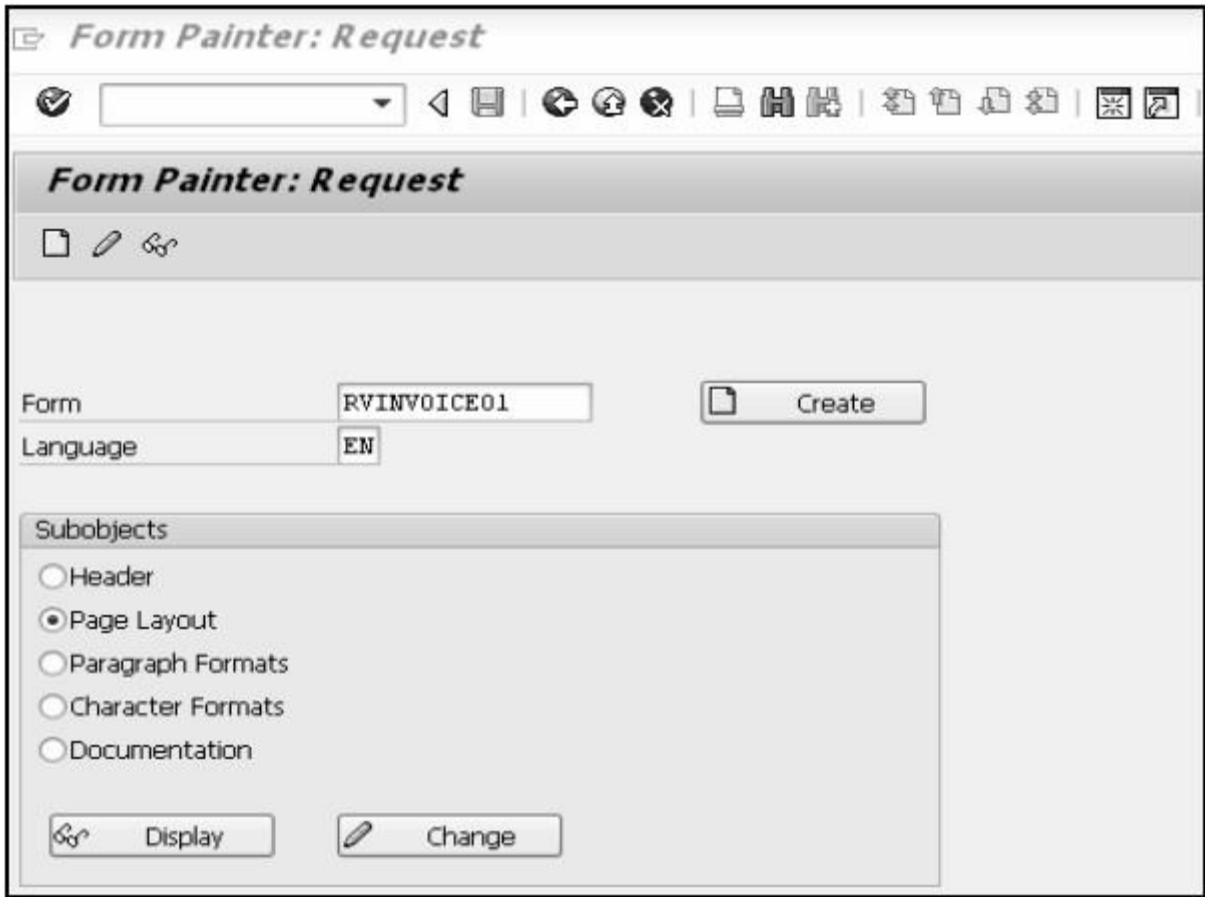
Düzen - Bu, form içeriğinin göründüğü bir dizi pencere tarafından tanımlanır.

SAPscript – Form Boyacısı Aracı

Form Boyacısı aracı, bir SAPscript formunun grafik düzenini ve formu işlemek için çeşitli işlevleri sağlar. Aşağıdaki örnekte, standart bir SAPscript formu RVINVOICE01'den pafta yapısını kopyaladıktan sonra bir fatura formu oluşturacağız ve Form Painter aracına erişerek mizanpajını görüntüleyeceğiz.

Adım 1 − Form Boyacısı'nı açın. Ekranı ister SAP menüsünden dolaşarak isterseniz SE71 işlem kodunu kullanarak talep edebilirsiniz.

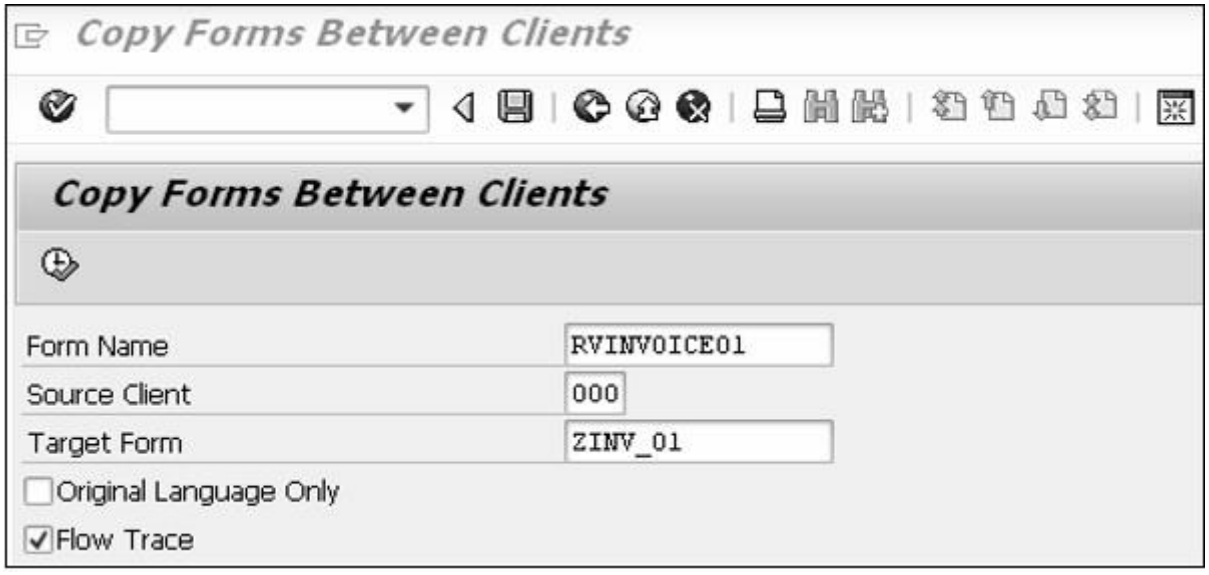
Adım 2 − Form Boyacısı, istek ekranında, sırasıyla Form ve Dil alanlarına bir SAPscript formu için bir ad ve dil girin. Bu alanlara sırasıyla 'RVINVOICE01' ve 'EN' girelim.



Adım 3 − Alt nesneler grup kutusunda Sayfa Düzeni radyo düğmesini seçin.

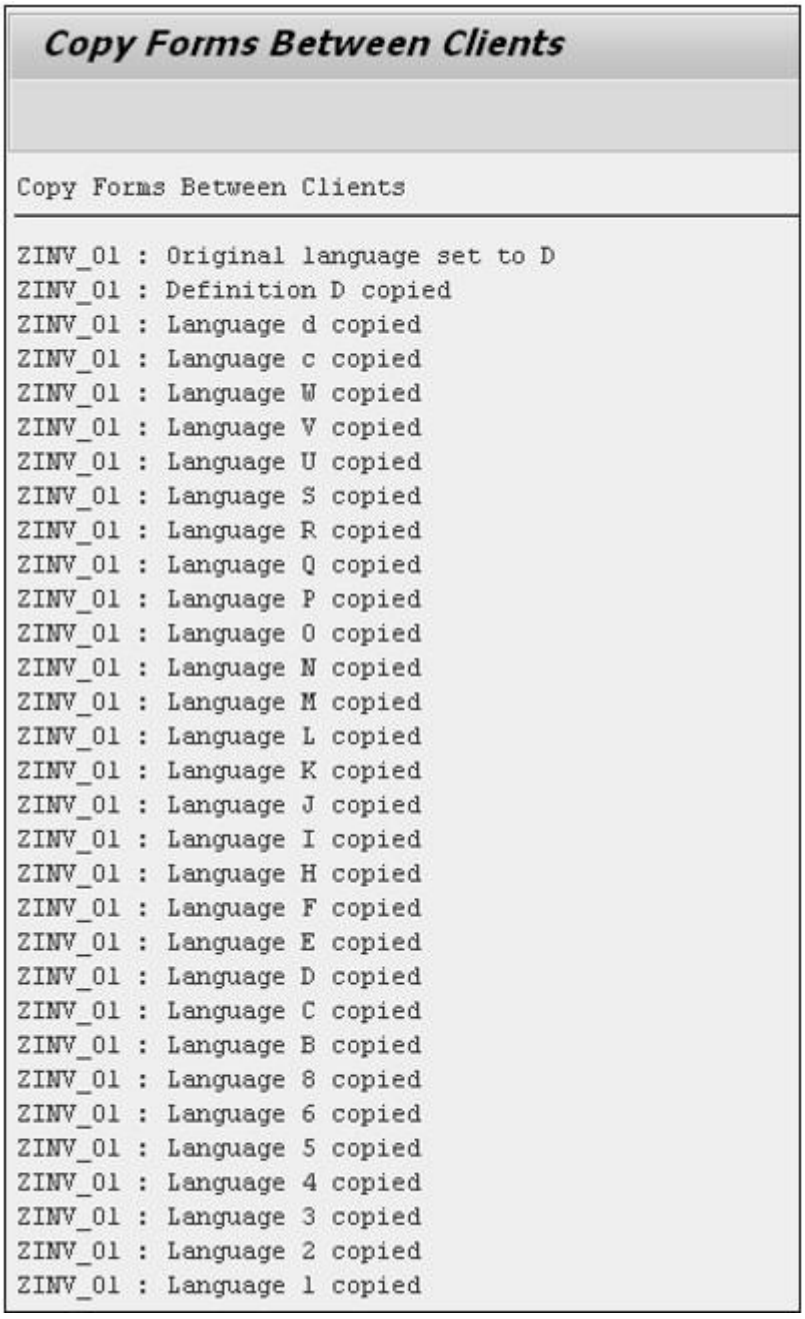
Adım 4 − RVINVOICE01 formunun bir kopyasını oluşturmak için Yardımcı Programlar → İstemciden Kopyala'yı seçin. 'İstemciler Arasında Formları Kopyala' ekranı görünür.

Adım 5 − 'Müşteriler Arasında Formları Kopyala' ekranında, Form Adı alanına formun orijinal adı 'RVINVOICE01', Kaynak İstemci alanına kaynak istemcinin '000' numarasını ve Hedef Formu alanında 'ZINV_01' olarak hedef formu. Diğer ayarların değişmeden kaldığından emin olun.



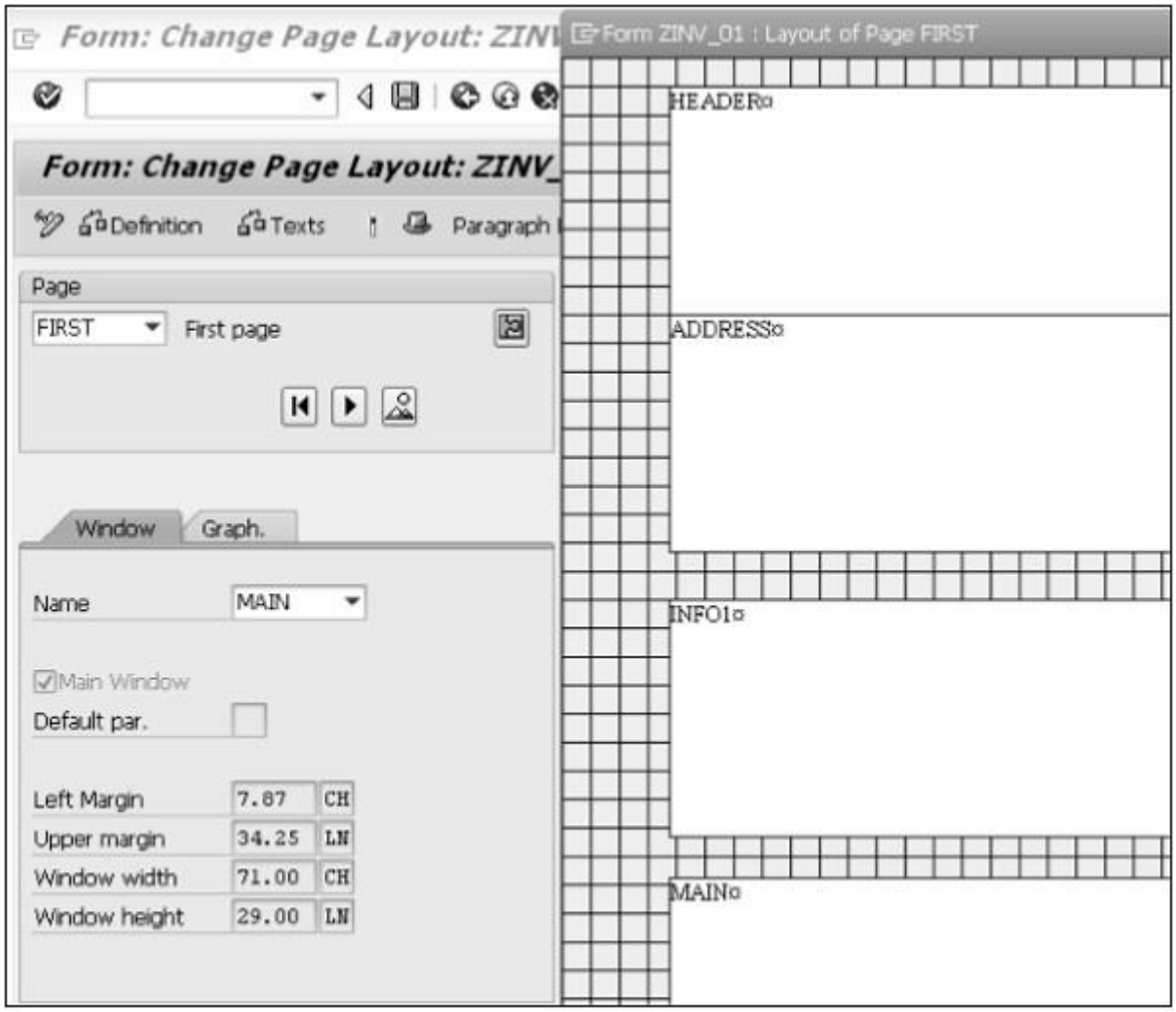
Adım 6 − Ardından, 'İstemciler Arasında Formları Kopyala' ekranındaki Yürüt simgesine tıklayın. 'Nesne Dizini Girişi Oluştur' iletişim kutusu görünür. Kaydet simgesine tıklayın.

ZINV_01 formu, RVINVOICE01 formundan kopyalanır ve aşağıdaki anlık görüntüde gösterildiği gibi 'Müşteriler Arasında Formları Kopyala ekranında' görüntülenir −



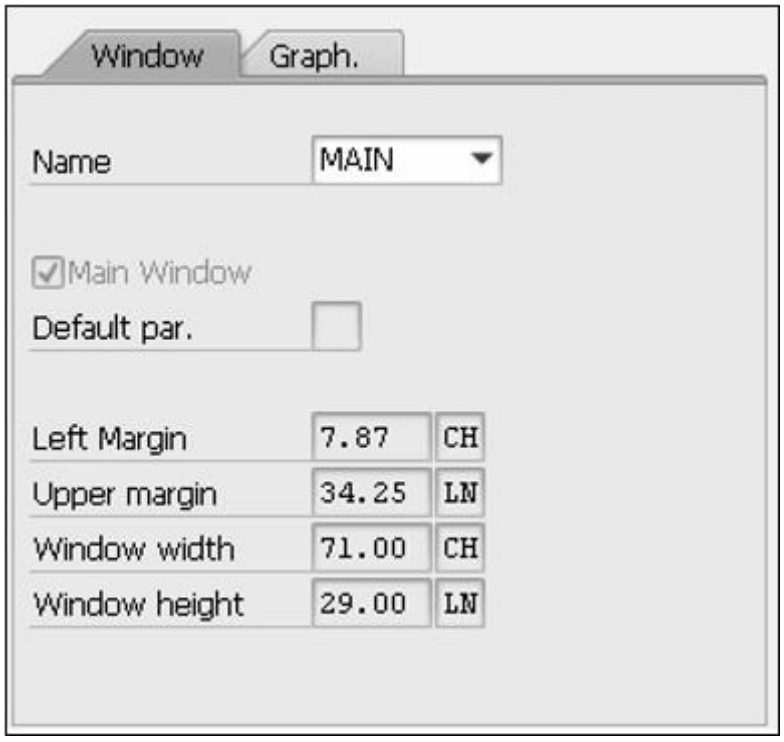
Adım 7 − Geri simgesine iki kez tıklayın ve kopyalanan ZINV_01 formunun adını içeren Form Boyacısı: İstek ekranına geri dönün.

Adım 8 − Görüntüle düğmesine tıkladıktan sonra, aşağıdaki ekran görüntüsünde gösterildiği gibi 'Form ZINV_01: Layout of Page FIRST' penceresi ve 'Form: Change Page Layout: ZINV_01' ekranı belirir.



Adım 9 − 'Form ZINV_01: Sayfa İLK Düzeni' penceresi, formun ilk düzenini gösterir. Formun düzeni beş pencere içerir: HEADER, ADDRESS, INFO, INFO1 ve MAIN. Bu pencerelerin açıklamasına PC Editor'dan erişilebilir.

Örneğin, sadece ANA pencereyi seçip 'Form: Sayfa Düzenini Değiştir: ZINV_01' ekranındaki Metin simgesine tıklayarak, aşağıdaki ekran görüntüsünde gösterildiği gibi tüm kenar boşluğu değerlerini görüntüleyebilirsiniz −

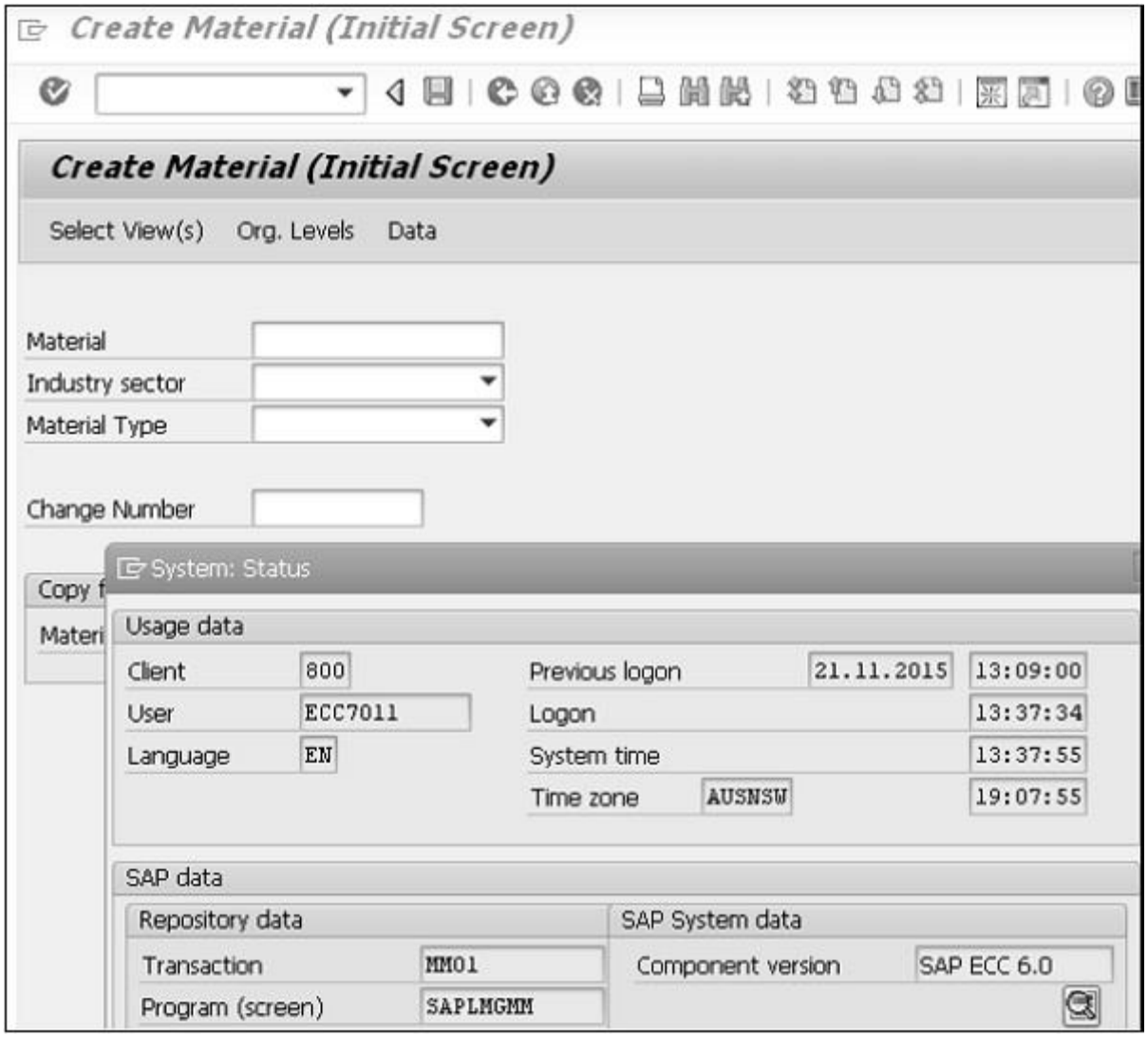


SAP ABAP - Müşteri Çıkışları

Müşteri çıkışları, SAP standart programlarına kanca olarak kabul edilebilir. Kodu yazmak için bir erişim anahtarına ihtiyacımız yok ve SAP standart programını değiştirmeye gerek yok. Bu çıkışların herhangi bir işlevi yoktur ve boştur. Çeşitli müşteri gereksinimlerini karşılamak için iş mantığı eklenebilir. Ancak, Müşteri Çıkışları tüm programlar için mevcut değildir.

Standart İşlemler İçin Müşteri Çıkışları

Standart işlemlerle ilgili olarak müşteri çıkışlarını bulma adımları aşağıdadır. MM01'de (Material Master Creation) bulunan müşteri çıkışlarını tanımlayalım.

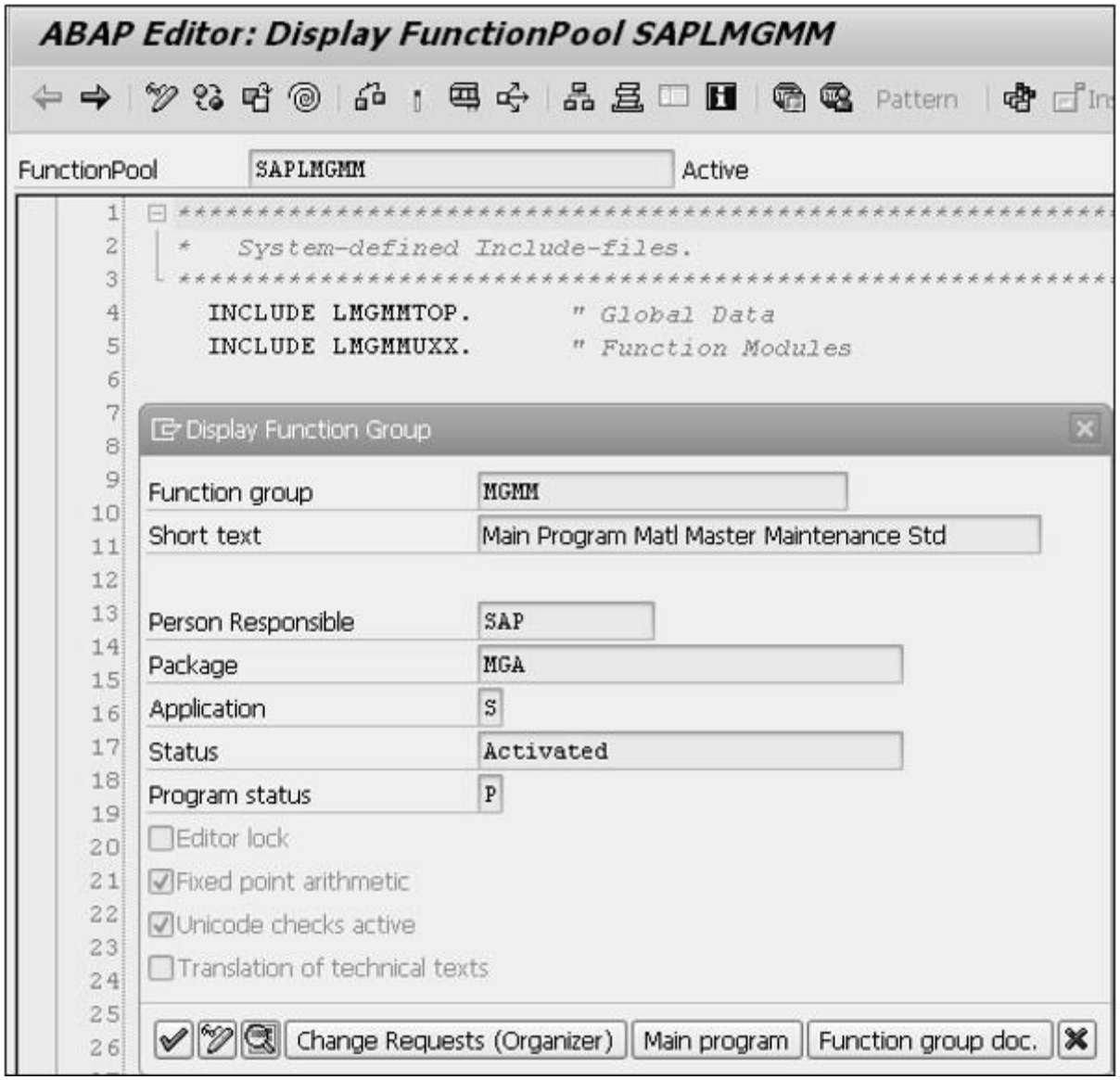


Adım 1 − MM01 işlemine gidin ve yukarıdaki ekran görüntüsünde gösterildiği gibi Menü çubuğu → Sistem → Durum'a giderek MM01 program adını belirleyin.

Adım 2 − Açılır ekrandan program adını alın. Programın adı 'SAPLMGMM'dir.

Adım 3 − SE38 işlemine gidin, program adını girin ve Görüntüle'ye tıklayın.

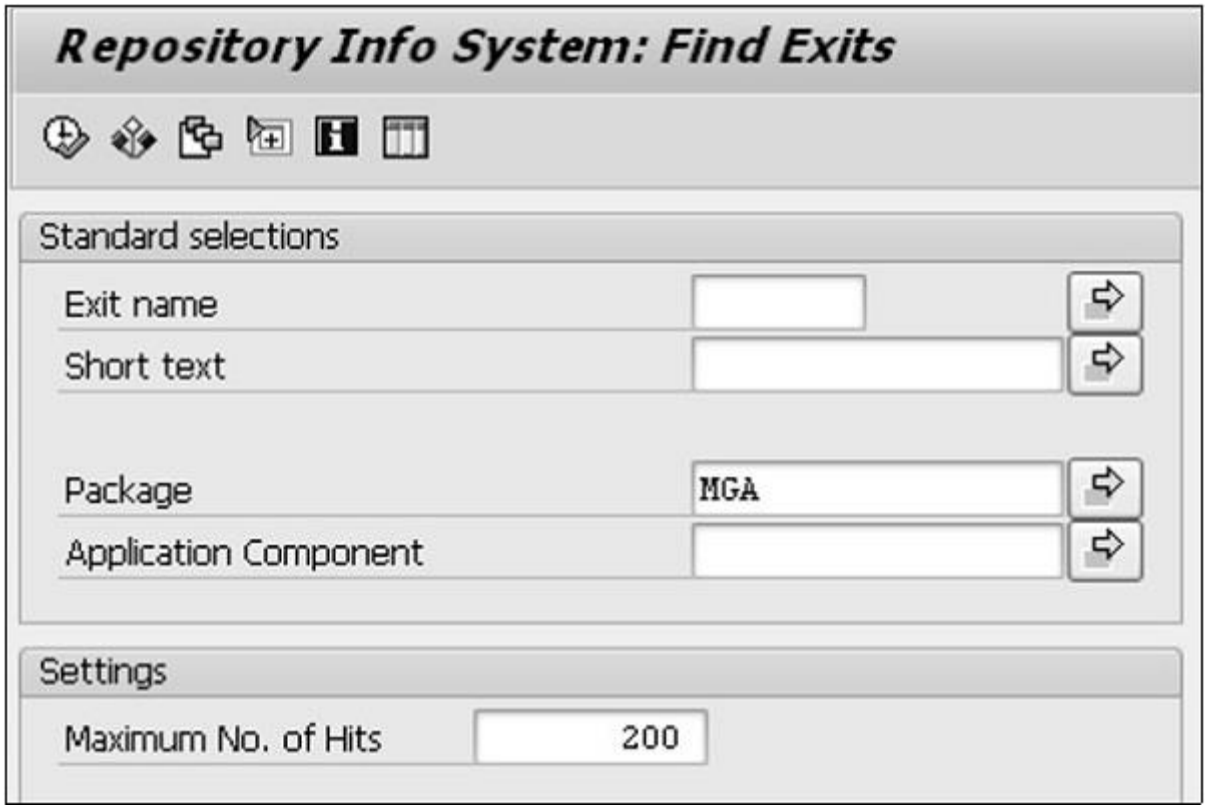
Adım 4 − → Özellikler'e gidin ve bu program adının paketini bulun.



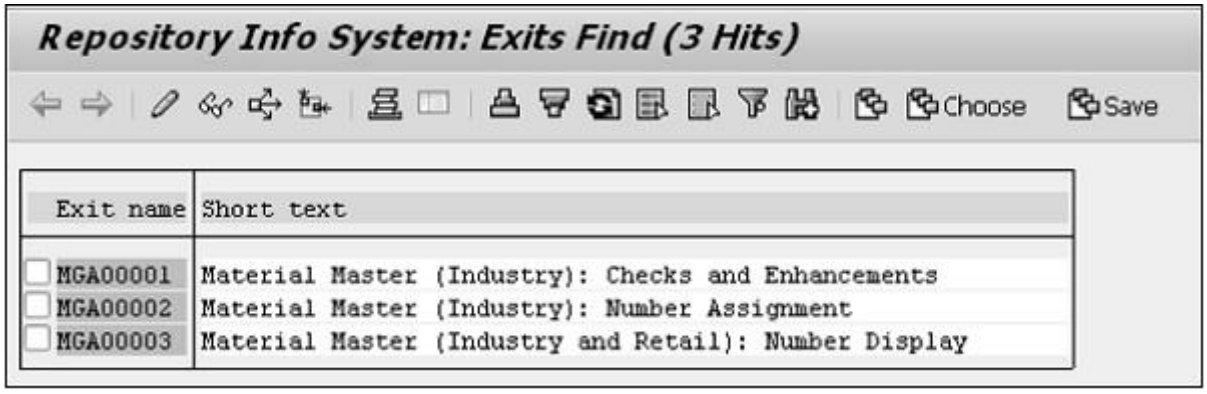
Paket adı 'MGA'dır.

Adım 5 − Genellikle müşteri çıkışlarını tanımlamak için kullanılan SMOD işlem koduna gidin. Yardımcı Programlar → Bul'a gidin (veya) SMOD işlem kodunda doğrudan Ctrl + F tuşlarına basabilirsiniz.

Adım 6 − 'Çıkışları Bul' ekranına gittikten sonra, daha önce aldığımız paket adını girin ve F8 (Yürüt) düğmesine basın.



Yukarıdaki adımlar, Malzeme Ana Oluşturma'da bulunan çıkışların listesiyle birlikte aşağıdaki çıktıyı üretir.



SAP ABAP - Kullanıcı Çıkışları

Standart SAP çıkarıcılar, örneğin yetkiler veya zaman kontrollerinde beklenen verileri veya gerekli işlevselliği sağlamazsa, kullanıcı çıkışları bir ayıklamada kullanılır. Kullanıcı çıkışları, Satış ve Dağıtım (SD) modüllerinde yaygın olarak kullanılır. SAP'nin satış, nakliye, sevkiyat ve faturalama alanlarında sağladığı birçok çıkış var. Bir kullanıcı çıkışı, standart SAP tüm gereksinimleri karşılayamadığında bazı değişiklikler yapmak üzere tasarlanmıştır.

Her satış alanında hangi çıkışların mevcut olduğuna erişebilmek için şu yolu kullanarak IMG'ye gidin: IMG → Satış ve Dağıtım → Sistem Değişiklikleri → Kullanıcı Çıkışları. SD alanlarındaki her bir çıkış için belgeler ayrıntılı olarak açıklanmıştır.

Örneğin, Satış Belgesi İşleme'de (sözleşme, teklif veya satış siparişi) kullanıcı çıkışlarını bulmak istiyorsanız, yukarıda belirtilen yolu takip edin ve Satış → Kullanıcı

Çıkışları'ndaki Kullanıcı Çıkışları düğümünü genişletmeye devam edin. Satış Belgesi İşleme'de bulunan tüm kullanıcı çıkışlarını görmek için simge belgelerine tıklayın.

S.No.	Kullanıcı Çıkışı ve Açıklama
1	**USEREXIT_FIELD_MODIFICATION** Ekran özelliklerini değiştirmek için kullanılır.
2	**USEREXIT_SAVE_DOCUMENT** Kullanıcı Kaydet'e bastığında işlemleri gerçekleştirmeye yardımcı olur.
3	**USEREXIT_SAVE_DOCUMENT_PREPARE** Giriş alanlarını kontrol etmek, alana herhangi bir değer koymak veya kullanıcılara bir göstermek ve belgeyi onaylamak için çok kullanışlıdır.
4	**USEREXIT_MOVE_FIELD_TO_VBAK** Kullanıcı başlık değişiklikleri başlık çalışma alanına taşındığında kullanılır.
5	**USEREXIT_MOVE_FIELD_TO_VBAP** Kullanıcı öğesi değişiklikleri SAP öğesi çalışma alanına taşındığında kullanılır.

Bir Kullanıcı Çıkışı, Müşteri Çıkışları ile aynı amaca hizmet eder, ancak bunlar yalnızca SD modülü için kullanılabilir. Çıkış, bir İşlev Modülüne çağrı olarak uygulanır. Kullanıcı Çıkışları, SAP standart programlarında yapılan değişikliklerdir.

Örnek

```
REPORT ZUSEREXIT1.
TABLES:
  TSTC, TSTCT,
  TADIR, TRDIR, TFDIR, ENLFDIR,
  MODSAPT, MODACT.

DATA:
  JTAB LIKE TADIR OCCURS 0 WITH HEADER LINE,
  field1(30),
  v_devclass LIKE TADIR-devclass.

PARAMETERS:
  P_TCODE LIKE TSTC-tcode OBLIGATORY.

SELECT SINGLE *
```

```
  FROM TSTC
  WHERE tcode EQ P_TCODE.

IF SY-SUBRC EQ 0.
  SELECT SINGLE *
  FROM TADIR

  WHERE pgmid = 'R3TR' AND
      object = 'PROG' AND
      obj_name = TSTC-pgmna.

  MOVE TADIR-devclass TO v_devclass.

  IF SY-SUBRC NE 0.
  SELECT SINGLE *
    FROM TRDIR
    WHERE name = TSTC-pgmna.

  IF TRDIR-subc EQ 'F'.
    SELECT SINGLE *
      FROM TFDIR
      WHERE pname = TSTC-pgmna.

    SELECT SINGLE *
      FROM ENLFDIR
      WHERE funcname = TFDIR-funcname.

    SELECT SINGLE *
      FROM TADIR
      WHERE pgmid = 'R3TR' AND
          object = 'FUGR' AND
          obj_name EQ ENLFDIR-area.
      MOVE TADIR-devclass TO v_devclass.
    ENDIF.
  ENDIF.

  SELECT *
    FROM TADIR
    INTO TABLE JTAB

    WHERE pgmid = 'R3TR' AND
        object = 'SMOD' AND
        devclass = v_devclass.

  SELECT SINGLE *
    FROM TSTCT
    WHERE sprsl EQ SY-LANGU AND
        tcode EQ P_TCODE.

  FORMAT COLOR COL_POSITIVE INTENSIFIED OFF.
  WRITE:/(19) 'Transaction Code - ',
    20(20) P_TCODE,
```

```
    45(50) TSTCT-ttext.
  SKIP.

  IF NOT JTAB[] IS INITIAL.
    WRITE:/(95) SY-ULINE.
    FORMAT COLOR COL_HEADING INTENSIFIED ON.

    WRITE:/1 SY-VLINE,
         2 'Exit Name',
         21 SY-VLINE ,
         22 'Description',
         95 SY-VLINE.

    WRITE:/(95) SY-ULINE.
    LOOP AT JTAB.
      SELECT SINGLE * FROM MODSAPT
      WHERE sprsl = SY-LANGU AND
           name = JTAB-obj_name.

      FORMAT COLOR COL_NORMAL INTENSIFIED OFF.
      WRITE:/1 SY-VLINE,
           2 JTAB-obj_name HOTSPOT ON,
           21 SY-VLINE ,
           22 MODSAPT-modtext,
           95 SY-VLINE.
    ENDLOOP.

    WRITE:/(95) SY-ULINE.
    DESCRIBE TABLE JTAB.
    SKIP.
    FORMAT COLOR COL_TOTAL INTENSIFIED ON.
    WRITE:/ 'No of Exits:' , SY-TFILL.

  ELSE.
    FORMAT COLOR COL_NEGATIVE INTENSIFIED ON.
    WRITE:/(95) 'User Exit doesn't exist'.
  ENDIF.
ELSE.

  FORMAT COLOR COL_NEGATIVE INTENSIFIED ON.
  WRITE:/(95) 'Transaction Code Does Not Exist'.
ENDIF.

AT LINE-SELECTION.
  GET CURSOR FIELD field1.
  CHECK field1(4) EQ 'JTAB'.
  SET PARAMETER ID 'MON' FIELD sy-lisel+1(10).
  CALL TRANSACTION 'SMOD' AND SKIP FIRST SCREEN.
```

İşlem sırasında 'ME01' işlem kodunu girin ve F8 (Yürüt) düğmesine basın. Yukarıdaki kod aşağıdaki çıktıyı üretir -

Transaction Code - ME01 Maintain Source List

Exit Name	Description
AMPL0001	User subscreen for additional data on AMPL
LMEDR001	Enhancements to print program
LMELA002	Adopt batch no. from shipping notification when posting a GR
LMELA010	Inbound shipping notification: Transfer item data from IDOC
LMEQR001	User exit for source determination
LMEXF001	Conditions in Purchasing Documents Without Invoice Receipt
LWSUS001	Customer-Specific Source Determination in Retail
MO6B0001	Role determination for purchase requisition release
MO6B0002	Changes to comm. structure for purchase requisition release
MO6B0003	Number range and document number
MO6B0004	Number range and document number
MO6B0005	Changes to comm. structure for overall release of requisn.
MO6E0004	Changes to communication structure for release purch. doc.
MO6E0005	Role determination for release of purchasing documents
ME590001	Grouping of requsitions for PO split in ME59
MEETA001	Define schedule line type (backlog, immed. req., preview)
MEFLD004	Determine earliest delivery date f. check w. GR (only PO)
MELAB001	Gen. forecast delivery schedules: Transfer schedule implem.
MEQUERY1	Enhancement to Document Overview ME21N/ME51N
MEVME001	WE default quantity calc. and over/ underdelivery tolerance
MM06E001	User exits for EDI inbound and outbound purchasing documents
MM06E003	Number range and document number
MM06E004	Control import data screens in purchase order
MM06E005	Customer fields in purchasing document

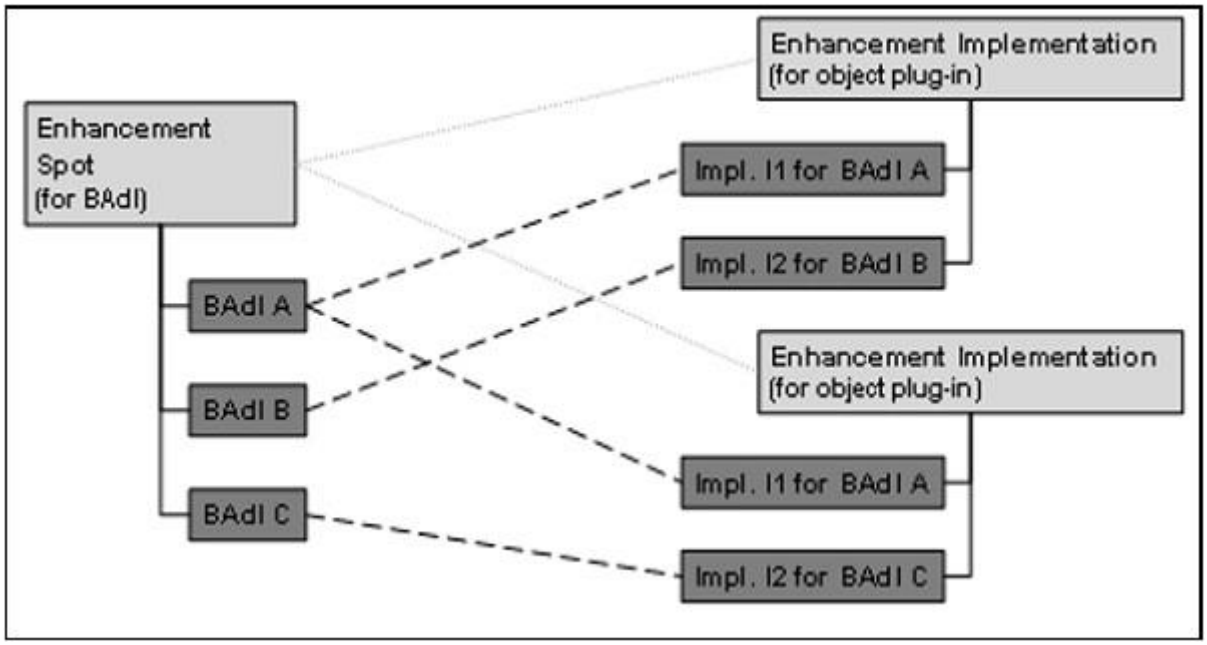
SAP ABAP - İş Eklentileri

Bazı durumlarda, çeşitli uygulamaların işlevselliğini geliştirmek için bir yazılım uygulamasında özel işlevlerin önceden tanımlanması gerekir. MS Excel'in işlevselliğini geliştirmek için birçok Microsoft Excel eklentisi vardır. Benzer şekilde SAP , BADI olarak bilinen **İş Eklentileri** sağlayarak önceden tanımlanmış bazı işlevleri kolaylaştırır .

BADI, bir SAP programcısının, bir kullanıcının veya belirli bir endüstrinin, SAP sistemindeki mevcut programa bazı ek kodlar eklemesini kolaylaştıran bir geliştirme tekniğidir. SAP sistemini geliştirmek için standart veya özelleştirilmiş mantığı kullanabiliriz. SAP uygulamasını geliştirmek için önce bir BADI tanımlanmalı ve ardından uygulanmalıdır. BADI tanımlanırken bir arayüz oluşturulur. BADI, sırayla bir veya daha fazla bağdaştırıcı sınıfı tarafından uygulanan bu arabirim tarafından uygulanır.

BADI tekniği, diğer geliştirme tekniklerinden iki yönden farklıdır -

- Geliştirme tekniği yalnızca bir kez uygulanabilir.
- Bu geliştirme tekniği birçok müşteri tarafından aynı anda kullanılabilir.



Ayrıca filtre BADI'leri de oluşturabilirsiniz; bu, BADI'lerin geliştirme teknikleriyle mümkün olmayan filtrelenmiş veriler temelinde tanımlandığı anlamına gelir. BADI kavramı, SAP Sürüm 7.0'da aşağıdaki hedeflerle yeniden tanımlanmıştır:

- ABAP dilinde 'GET BADI' ve 'CALL BADI' olmak üzere iki yeni öğe ekleyerek bir SAP sistemindeki standart uygulamaları geliştirmek.
- Bir SAP sistemindeki standart uygulamaların geliştirilmesi için bağlamlar ve filtreler gibi daha fazla esneklik özelliği sunar.

Bir BADI oluşturulduğunda, bir arayüz ve menü geliştirmeleri ve ekran geliştirmeleri için fonksiyon kodları gibi diğer ek bileşenleri içerir. BADI oluşturma, müşterilerin kendi geliştirmelerini standart SAP uygulamasına dahil etmelerine olanak tanır. Geliştirme, arabirim ve oluşturulan sınıflar, uygun bir uygulama geliştirme ad alanında bulunur.

Bu nedenle, bir BADI, SAP bileşenlerinde 'önceden tanımlanmış noktalar' oluşturmak için ABAP nesnelerini kullanan bir geliştirme tekniği olarak düşünülebilir. Bu önceden tanımlanmış noktalar daha sonra özel gereksinimlerine uyacak şekilde bireysel endüstri çözümleri, ülke çeşitleri, ortaklar ve müşteriler tarafından uygulanır. SAP, BADI geliştirme tekniğini Sürüm 4.6A ile birlikte tanıttı ve teknik, Sürüm 7.0'da yeniden uygulandı.

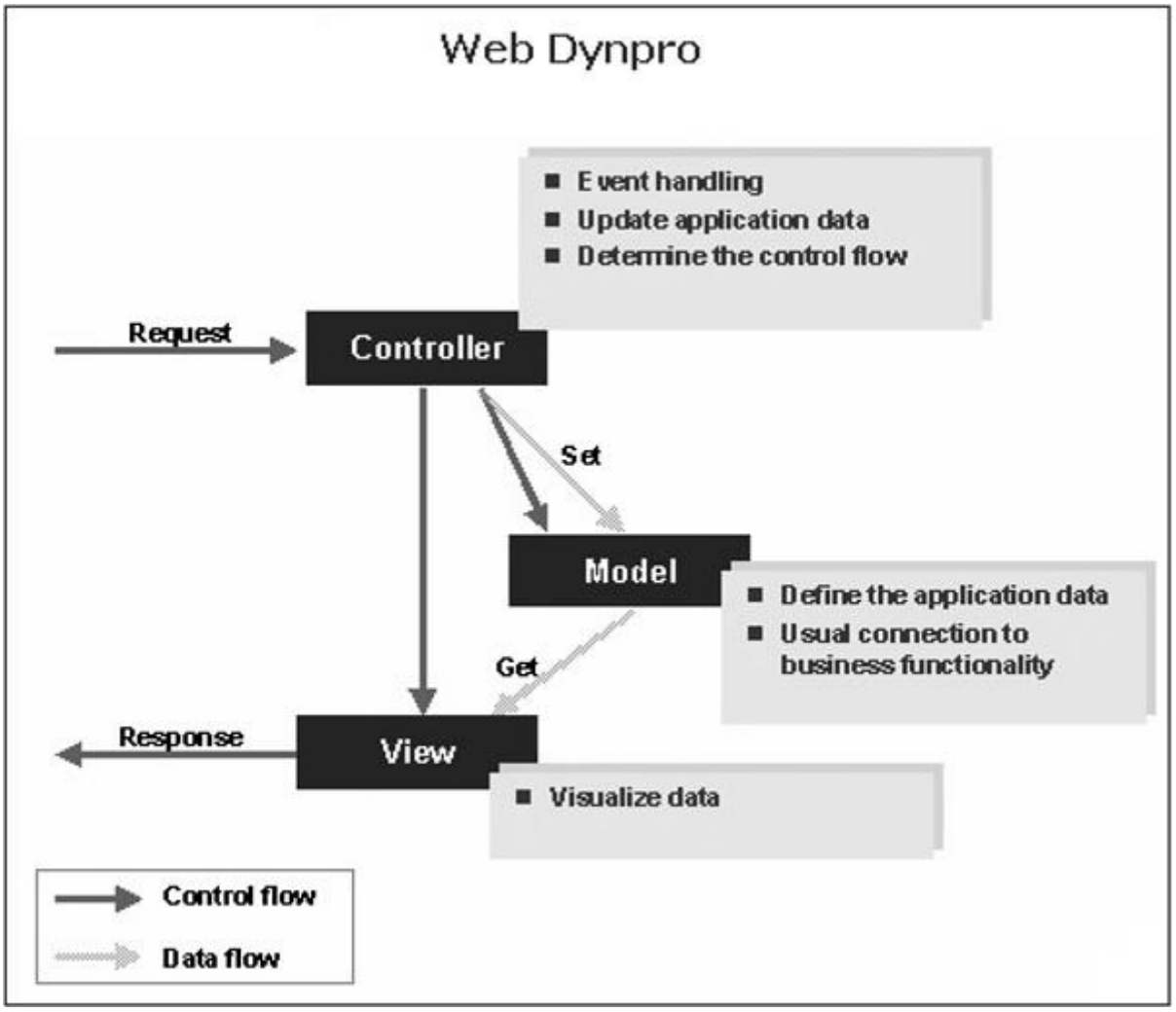
SAP ABAP - Web Dynpro

ABAP için Web Dynpro (WD), SAP AG tarafından geliştirilen SAP standart kullanıcı arayüzü teknolojisidir. SAP geliştirme araçlarını ve kavramlarını kullanan SAP ABAP ortamında web tabanlı uygulamaların geliştirilmesinde kullanılabilir. Raporlama için verilere ve işlevlere erişmek için doğrudan arka uç SAP R/3 sistemlerine bağlanmak için bir ön uç web kullanıcı arayüzü sağlar.

ABAP için Web Dynpro, ABAP Workbench'e (işlem: SE80) entegre edilmiş belirli geliştirme araçlarına sahip bir çalışma zamanı ortamından ve bir grafik geliştirme ortamından oluşur.

Web Dynpro'nun Mimarisi

Aşağıdaki çizim, Web Dynpro'nun genel mimarisini göstermektedir -



Aşağıda Web Dynpro ile ilgili akılda tutulması gereken birkaç nokta bulunmaktadır -

- Web Dynpro, kullanıcı arayüzleri için SAP NetWeaver programlama modelidir.
- Tüm Web Dynpro uygulamaları, Model View Controller (MVC) programlama modeline göre yapılandırılmıştır.
- Model, ana sisteme bir arayüz tanımlar ve Web Dynpro uygulamasının sistem verilerine erişimi olabilir.
- Görünüm, verileri web tarayıcısında göstermekten sorumludur.
- Denetleyici, görünüm ve model arasında bulunur. Kontrolör, görünümde görüntülenecek model verilerini biçimlendirir. Kullanıcı tarafından yapılan kullanıcı girişlerini işleyerek modele geri döndürür.

Avantajlar

Web Dynpro, uygulama geliştiricileri için aşağıdaki avantajları sunar -

- Grafiksel araçların kullanımı, uygulama çabasını önemli ölçüde azaltır.
- Bileşenleri kullanarak yeniden kullanım ve daha iyi bakım.
- Düzen ve gezinme, Web Dynpro araçları kullanılarak kolayca değiştirilebilir.
- Kullanıcı arabirimi erişilebilirliği desteklenir.

Salih KÜÇÜK - SAP Retail Consultant

- ABAP geliştirme ortamında tam entegrasyon.

Web Dynpro Bileşeni ve Penceresi

Bileşen, Web Dynpro uygulama projesinin global birimidir. Bir Web Dynpro bileşeni oluşturmak, yeni bir Web Dynpro uygulaması geliştirmenin ilk adımıdır. Bileşen oluşturulduktan sonra Web Dynpro nesne listesinde bir düğüm görevi görür. Bir bileşende istediğiniz sayıda bileşen görünümü oluşturabilir ve bunları istediğiniz sayıda ilgili Web Dynpro penceresinde birleştirebilirsiniz.

Her Web Dynpro bileşeninde en az bir Web Dynpro penceresi bulunur. Web Dynpro penceresi, ön uç web uygulamasında görüntülenen tüm görünümleri gömer. Pencere, ABAP Workbench'in pencere düzenleyicilerinde işlenir.

Not

- Bileşen görünümü, açıklama, onu oluşturan kişinin adı, oluşturma tarihi ve atanan geliştirme paketi dahil olmak üzere uygulamaya ilişkin tüm yönetim ayrıntılarını görüntüler.
- Web Dynpro uygulaması, ABAP Workbench'in nesne listesindeki bağımsız nesnedir. Pencere ve uygulama arasındaki etkileşim, belirli bir pencerenin arayüz görünümü tarafından oluşturulur.