

**AZ Treinamentos**

Cursos de A a Z feitos pra você!

Academia ABAP

With

Hana & Fiori

# Sumário

1.	Apresentação do Material AZ Treinamentos .....	4
2.	Liderança pessoal em projetos SAP .....	4
3.	O que é e como funciona o SAP (ECC/Fiori/Hana) .....	7
4.	Ambientes SAP .....	8
5.	Diferenças entre o SAP ECC/Hana/Fiori .....	19
6.	Conhecendo o ABAP .....	20
7.	Como criar pacote .....	25
8.	Explicação sobre requests no SAP: .....	26
9.	Utilizando base de programas já existentes no SAP .....	28
	Utilização .....	30
10.	Versionamento de programas (Controle de versão) .....	30
11.	Entendendo a estrutura do código .....	32
12.	Explicação do conceito de programação procedural .....	33
13.	Explicação do conceito de programação orientada a objeto .....	34
14.	Objetos, Sintaxes e Lógica de programação .....	36
15.	Lógica de programação .....	49
16.	Programas de carga/Interface (Upload/Download) .....	56
17.	Formato TXT (Upload/Download) .....	56
18.	Upload de Arquivos – File Local: .....	56
19.	Download de Arquivos – File Local: .....	58
20.	Formato Excel (Upload/Download) .....	60
21.	Upload de Arquivos Excel – File Local: .....	60
22.	Diferentes tipos de ALVs e seus conceitos .....	61
23.	Tabela de configuração de Fieldcat: .....	62
24.	Seleção dos dados: .....	63
25.	Montagem do Fieldcat: .....	63
26.	ALV Múltiplo na mesma tela .....	67
27.	Aula de Performance Avançada .....	68
	• Arquitetura Client-Server .....	68
28.	Grandes vilões no que se refere à performance .....	69
29.	Comunicação entre o Application Server e Database Server - Transmissão de pacotes .....	70
	• Select e Logical Database .....	70
30.	Cursor Caching .....	71
31.	Logical database e Tables statements .....	72

Report ZAZ_PERFORMANCE. ....	72
Tables: vbak, vbkd, vbpa. ....	72
• Quando usar internal table ou Dictionary structure .....	78
32. Processamento de grandes tabelas .....	79
33. Índices secundários: .....	81
34. Regras gerais para criação e uso de índices secundários: .....	82
Select * from ztable where .....	82
➔ isto é executado usando o índice .....	82
Informações gerais: .....	83
Não crie índices se uma das seguintes condições se aplicar: .....	84
35. Conceito de atualização em update task .....	85
36. Comandos Select .....	86
37. ALGUNS TIPOS DE SELECT .....	86
38. Dicas para otimização do código .....	90
39. Ferramentas para auxiliar os desenvolvedores .....	91
40. Interfaces Batch ou Conversões: .....	92
41. AVALIAÇÃO: .....	93



## 1. Apresentação do Material AZ Treinamentos

Explicação do conteúdo do material durante o vídeo.



## 2. Liderança pessoal em projetos SAP

- Como se portar dentro de consultorias/projetos SAP/ABAP:

Os trabalhos de SAP são divididos em geralmente de duas formas, em consultorias ou em clientes. Trabalhando em consultorias, temos mais contato com outras pessoas mais técnicas e mais focadas em SAP, pois a grande parte das consultorias tem como principal sistema o ERP SAP.

Dentro de uma consultoria os trabalhos costumam ter um padrão mais alto, com documentações, estimativas mais detalhadas, desenhos técnicos, ou seja, todo trabalho é mais organizado, embora elaborar e organizar tudo também leve bastante tempo.

Os clientes costumam fechar projetos para implantar o SAP ou para fazer melhorias ou correções em seu sistema SAP, como o projeto é diretamente ligado ao cliente, todos os padrões e organizações podem variar bastante, pois cada empresa tem sua metodologia, sendo práticas de mercado ou até mesmo metodologias próprias para grandes grupos multinacionais.

Ao chegar em qualquer um desses meios de trabalho com SAP, procure sempre respeitar a risca cada regra imposta pelas empresas, pois em muitos casos o descumprimento de alguma das regras ou padrões da empresa pode resultar em desligamento do consultor envolvido, principalmente nos casos PJ. Trazer suas ideias para o projeto pode ser muito útil, mas sempre é melhor chegar e observar o comportamento de cada cliente ou consultoria.

Durante o curso, falaremos de alguns cuidados que devemos tomar dentro de clientes e consultorias, para que o início de suas carreiras não sofra com esses “pequenos” detalhes.

- Como organizar seus trabalhos dentro do SAP/ABAP:



Assim como falamos anteriormente sobre os padrões, é importante sempre estudar algum tipo de metodologia, ou até mesmo desenvolver uma que faça seu dia ser mais organizado e eficaz. Dentro da área de SAP lidamos com muitos documentos em Word, Excel, PDF, todos referentes a um chamado ou atividade que pode ser um código específico de cada empresa ou mesmo com os documentos gerados pelo SAP durante sua utilização. Se não organizarmos todas essas informações, teremos problemas muito sérios na hora de dar vida aos nossos desenvolvimentos, vamos falar durante o curso um pouco detalhado sobre este assunto.

#### ▪ O que nunca fazer no ambiente SAP/ABAP:



- Não respeitar as regras de segurança do ambiente
- Utilizar usuários não autorizados
- Utilizar ambientes não autorizados
- Forçar programas ou tabelas através de debug sem autorização

#### ▪ Envios de e-mails em projetos SAP/ABAP:



Enviar e-mail parece ser a tarefa mais simples do mundo, porém dentro de cada empresa temos padrões a serem respeitados de acordo com cada assunto, área, problema, etc, vamos abordar durante o curso um pouco sobre esses assuntos e veremos alguns exemplos de como proceder.

#### ▪ Como priorizar atividades em SAP/ABAP:



Um dos pontos mais focais na vida de um consultor, é ter o “senso de priorização”, pois muitas vezes temos muitas tarefas para realizar e acabamos por escolher aquela que não era tão prioritária quanto a um erro que está parando o faturamento de uma empresa, vamos falar mais tecnicamente durante o curso, explicando alguns exemplos usados dentro de projetos.

- **Como montar um Currículo para áreas de SAP/ABAP:**

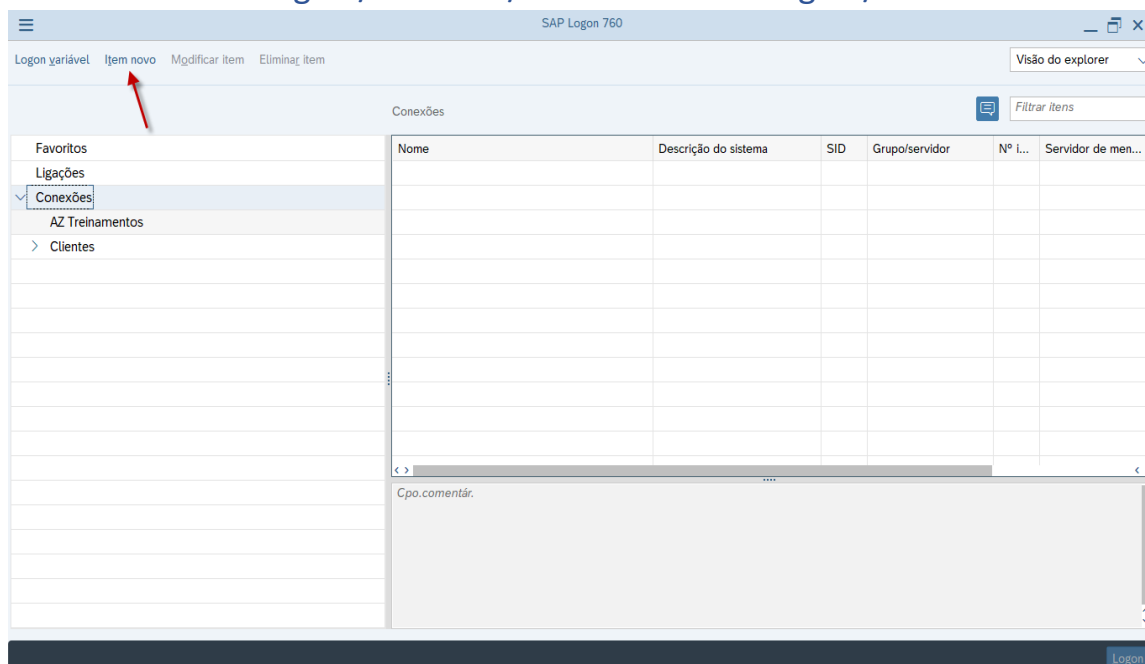


Montar um Currículo para áreas de SAP pode ser um pouco diferente do que você sempre aprendeu, vamos utilizar o modelo de um CV para explicar melhor o que devemos manter e não manter dentro de um CV.



### 3. O que é e como funciona o SAP (ECC/Fiori/Hana)

#### ■ SAP Logon / SAP Gui / Parâmetros de Logon / Client



- 1 – **Mudar:** Mudar sua senha a qualquer momento
- 2 – **Opções:** Encontrar opções adicionais do SAP Gui
- 3 – **Mandante:** É a instância gerada dentro do servidor, um servidor pode conter várias instancias dentro do mesmo servidor.
- 4 – **Usuário:** Usuário SAP
- 5 – **Texto de Entrada:** Texto usado para dar informações sobre o sistema no logon.
- 6 – **Senha:** Senha SAP
- 7 – **Idioma:** Idioma de Utilização do SAP



## 4. Ambientes SAP

### DEV (Desenvolvimento)



É o ambiente usado para o desenvolvimento dos programas e ajustes do SAP, é comum que os ambientes de DEV não contenham muitos dados nas tabelas, muitas vezes não possuem nenhum, para economizar espaço ou até mesmo por políticas de segurança. Existem também várias instâncias dentro de um mesmo ambiente, o que chamamos de Mandante ou “Client”, que são divisões onde podemos separar os dados, como por exemplo, podemos ter um mandante de número 100 sem nenhum dado em tabelas e um ambiente 110 com dados básicos para teste, sendo assim, todo código desenvolvido no ambiente 100 também será refletido no ambiente 110, mas os dados ficam apenas no 110, esse conceito é normalmente chamado de “Sandbox” e agiliza muito o processo de testes, já que não é necessário a transferência entre ambientes como DEV para QA ou DEV para PRD.

### ▪ QAS (Qualidade)



Nesse ambiente temos mais tabelas preenchidas, dados mais parecidos com os cenários do ambiente produtivo, pois aqui fazemos os testes reais tanto da parte funcional quanto da parte dos usuários que vão ter acesso aos desenvolvimentos ou ajustes no ambiente produtivo. Quase sempre o QAS é atualizado com os dados de produção, o que chamamos de “refresh”, os “refreshs” são realizados de tempo em tempo, de acordo com cada empresa, isso é necessário para evitar erros no ambiente produtivo ao transportar o programa após os testes. Vale lembrar que só testes podem ser realizados em QAS, não é possível codificar neste ambiente.



Os testes são divididos em algumas partes:

**Teste Unitário** – É um teste unitário da demanda para se certificar que programa esteja sendo executado sem DUMP.

**Teste de Integrado** – Garante que os componentes interligados funcionem conforme o esperado. Utilizado para testar rotinas de integrações.

**Teste Positivo-negativo** – Garante que a demanda vai funcionar no “caminho feliz” conforme o esperado e também vai funcionar no fluxo de exceção.

**Teste de regressão** – Testa de algo que mudado. Realiza o processo de teste novamente.

**Teste de caixa-preta** – Testa todas as entradas e saídas.

**Teste Funcional** – Testa as funcionalidades e regras de negócio presentes na documentação com objetivo de validar suas funcionalidades conforme a documentação técnica.

Cada demanda tem sua própria necessidade com isso muitas vezes não precisamos fazer todos os tipos de testes apresentados.

Cenários de testes.

Algo que devemos pensar com cuidado na hora da elaboração de um plano de teste é considerar quais os cenários (processo e rotinas) de testes devemos ter ou não.

Pensando a nível de cenário de teste, sugiro a classificação dos cenários conforme sua importância e criticidade e com isso atribuo um peso a cada cenário:

- Cenários críticos – 10p
- Cenários Importantes – 07p
- Cenários normais -04p
- Cenários com baixa prioridade -01p

Uma vez com a classificação dos cenários podemos fazer o plano de testes x os tipos de testes conforme exemplo da tabela abaixo:

Cenários	Tipo de teste	Peso	Status
Rotina Z1	Teste unitário	Importantes – 07p	
Input de fatura	Teste Funcional	Importantes – 07p	
Rotina de integração	Teste unitário	Críticos – 10p	
	Teste de Integração		
	Teste Funcional		
Cadastro de Banco	Teste unitário	Normais -04p	

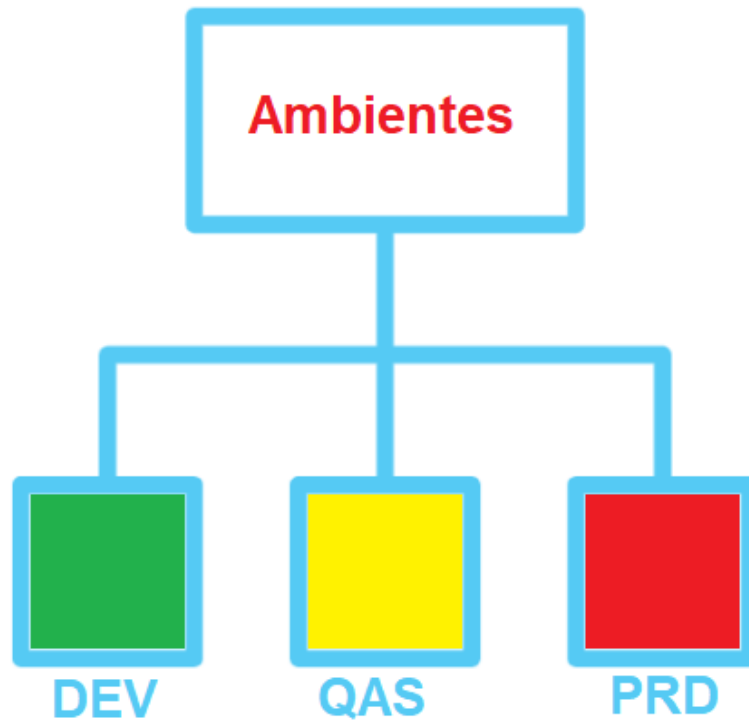
#### ▪ PRD (Produção)



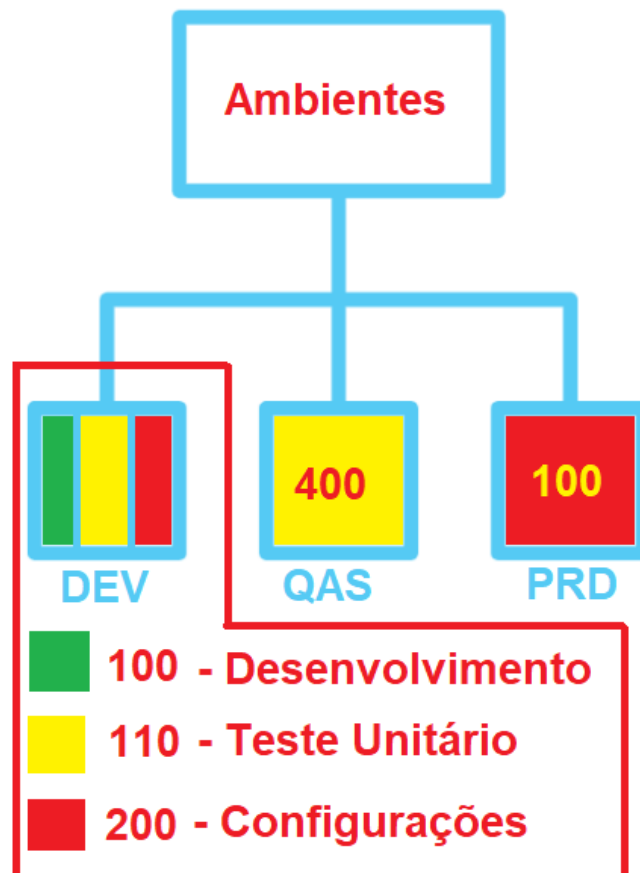
O ambiente produtivo é onde toda a “mágica” acontece, nele todos os usuários do SAP estão trabalhando com todo o SAP em operação, para cada área da empresa em que o SAP foi implantado. Neste ambiente não fazemos testes tão menos desenvolvemos, ele só é usado para o processo corporativo de cada empresa. Quando um erro ocorre em produção, ele se torna o mais emergencial entre todas as demandas, pois com a produção parada em alguma parte, alguma área da empresa também está com seus processos parados, não é muito incomum que isso ocorra em projetos de SAP.

Temos que ter muito cuidado ao analisar erros em produção, muitas vezes precisamos “debugar” algo diretamente em produção para entender um erro que só acontece “lá”, porém cada processo que analisamos pode estar sendo processado como real, ou seja, se estamos “debugando” uma ordem de venda no momento de salvar, se formos até o final do processamento essa ordem de venda será criada em ambiente produtivo, se não for estornada pode causar problemas para as áreas e isso pode ocorrer em todas as áreas do SAP.

Ambientes principais interligados:



Ambientes e Instâncias por ambiente:



## ▪ Demais ambientes

Além dos ambientes mais convencionais, muitas empresas possuem ambientes para outras áreas, como o PI/PI que faz integrações entre o SAP e outros sistemas, o GRC que cuida de toda a parte de Mensageria (NF-e, CT-E, etc.), Charm para controle de projetos e requests, dentre muitos outros, neste curso estaremos aprendendo um pouco sobre:

- GRC/NFe
- PI/PO
- Hana for ABAP
- Fiori for ABAP

## ▪ Explicação dos módulos do SAP no geral

SAP ERP (até 2003 SAP/R3, até 2007 mySAP ERP) é um sistema integrado de gestão empresarial (ERP) transacional, produto principal da SAP AG, uma empresa alemã, líder no segmento de software corporativos, tendo cerca de 86 mil clientes, segundo a própria SAP, em todo mundo, dentre a grande maioria empresas de grande porte.

O sistema procura contemplar a empresa como um todo, dividindo módulos, onde cada módulo corresponde a uma área específica, como por exemplo, o módulo SD (Sales and Distribution) que contempla a área de Vendas e Distribuição, fazendo a integração das informações para determinado processo. Cada programa é executado através de uma transação separadamente. Estes programas são desenvolvidos em ABAP, uma linguagem de programação da qual a SAP detém os direitos.

Cada módulo é responsável por mais mil processos de negócios, baseado em práticas do dia a dia de cada empresa. O sistema é configurado para atender a necessidade de cada determinado processo, onde mais de 8 mil tabelas administram em tempo real as informações que trafegam pela empresa. Seus métodos de trabalho são bastante conhecidos por disponibilizar conforto e eficiência ao relacionar programas da mais alta tecnologia e desenvolver programas capazes de solucionar os mais variados tipos de tarefas.

- SAP MM - Material Management (Gestão de Materiais)
- SAP WM - Warehouse Management (Gerenciamento de Armazenagem)
- SAP SD - Sales and Distribution (Vendas e Distribuição)
- SAP FI - Financial Accounting (Contabilidade Financeira)
- SAP PP - Production Planning and Control (Planejamento da Produção)
- SAP HCM - Human Capital Management (Gerenciamento de capital humano)
- SAP PS - Project System (sistemas de Projeto)
- SAP CO - Controlling (Controladoria)
- SAP QM - Quality Management (Administração de Qualidade)
- SAP PM - Plant Maintenance (Planejamento da Manutenção)
- SAP IS - Industry Solutions (Soluções Industriais)
- SAP BW - Business Warehousing (Armazenamento de negócios)
- SAP RE - Real Estate (Imobiliária)

### Um sistema SAP R/3 é composto por três camadas:

- Frontend
- Application
- Database

**Frontend:** é camada responsável por "exibir" as telas ao usuário.

**Application:** é onde são processadas as operações efetuadas, transferindo para o Frontend, os dados a serem exibidos. É nessa camada que os programas ABAP são executados.

A camada de Application possui diversos serviços e processos (também chamados de Work Process) disponíveis.

O desenho típico de uma instância SAP é um servidor de Banco de Dados com um ou mais servidores de Application. Isso garante a integridade dos dados, e permite uma distribuição de carga nos servidores de aplicativo entre os usuários.

### **Work Process:**

**Message:** Serviço interno responsável pela comunicação entre as instâncias.

**Dispatcher:** Serviço interno responsável pelo "despacho" das requisições para cada processo ou serviço.

**Gateway:** Garante a comunicação externa com outros sistemas.

**Enqueue:** Processo responsável pelo gerenciamento da tabela de objetos de bloqueio.

**Dialog:** Processo responsável pela execução dos processos visíveis pelo usuário.

**Background:** Processo responsável pela execução dos processos escalonados e/ou necessários de grande poder de processamento.

**Update:** Processo responsável pela atualização dos dados no banco de dados.

**Spool:** Processo que gerencia a fila de impressão.

**Database:** é a camada onde os dados são armazenados, quando a camada Application necessita de algum dado, o mesmo é requisitado a camada de Database.

## ▪ Standard e Z

Para dar manutenção ou para criar novos desenvolvimentos no SAP, nós os "ABAPs" podemos criar os objetos com as iniciais começando por "Z" ou "Y", pois a SAP reservou essas duas letras para identificar que a alteração não pertence ao pacote padrão ("standard"). As demais letras são usadas pela própria SAP para desenvolver novas soluções.

Não é possível alterar diretamente objetos "standards", existem alguns meios como Exits, Enhancements, BADIs, dentre outros para que isso aconteça.

## ▪ Explicação das ferramentas/navegação/atalhos presentes no SAP

Vamos fazer juntos o acesso ao SAP ECC da AZ Treinamento.

Utilize o ícone do SAP Logon, localizado em sua área de trabalho (Desktop), conforme a imagem abaixo:



Ao clicar duas vezes sobre a entrada do ambiente, será exibida uma tela como abaixo:

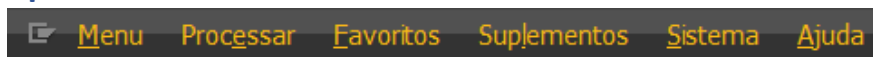
A imagem mostra a interface de login do SAP. No topo, há uma barra de menu com "Usuário", "Sistema" e "Ajuda". Abaixo, uma barra de ferramentas contém vários ícones. O corpo principal da tela é dividido em duas seções. À esquerda, sob o título "SAP", há um formulário de login com campos para "Mandante" (contendo "800"), "Usuário", "Senha" (com caracteres ocultos por pontos) e "Idioma de login" (contendo "PT"). À direita, sob o título "Informação", há um texto de boas-vindas: "Bem vindo ao ambiente ECC EHP7 da AZ Treinamentos", seguido por "Problemas de acesso: suporte@aztreinamentos.com". Abaixo disso, há informações sobre o backup: "Último Backup do ambiente: 17/08/2019" e "Senha Inicial padrão: 123456". Um agradecimento segue: "Obrigado por usar nossos serviços. Acreditamos sempre no seu sucesso!!!". Em seguida, há links para "www.aztreinamentos.com" e "contato@aztreinamentos.com". Uma observação (OBS) informa: "Não remova configurações/programas de outros usuários, o ambiente é compartilhado e pedimos, por gentileza, que usem o bom senso na hora de editar qualquer objeto." No final, há o texto "Agradecemos a compreensão!". Na barra de status na base da janela, há o logotipo do SAP e o texto "» ID7 (1) 000 | sapides | INS |".

Nessa tela acima, você poderá informar seu usuário SAP e sua senha, para ter acesso ao sistema, ao usar os ambientes da AZ Treinamentos, sempre verifique o texto de entrada ao lado direito, pois sempre colocamos informações úteis e as regras de utilização do ambiente.

Sempre verifique sua senha, pois ao errar a senha 3 vezes o acesso é bloqueado, sendo necessário o desbloqueio com o administrador de sistemas, se isso ocorrer, não se preocupe, envie um e-mail para [suporte@aztreinamentos.com](mailto:suporte@aztreinamentos.com), ou utilize nosso Whatsapp para informar o problema: 011 9 6629 0320.

Após “Logar” no sistema, uma tela como a abaixo será exibida, vamos conhecer um pouco mais das opções e ferramentas presentes no SAP.

## Menu Superior:



São o conjunto da maioria das configurações e opções de cada aplicação se encontra, opções como Salvar, Imprimir, Configurações Gerais, Características ou até mesmo processos empresariais através das aplicações específicas, vamos acompanhar em aula um pouco mais sobre o menu de entrada superior do SAP.

## Barra de botões superior:



Na barra de botões superior, encontramos as funções fixas, presentes em quase todas as telas do SAP, nela você pode encontrar opções como pesquisar, salvar, imprimir, page down, page up, nova tela e as configurações do SAP Gui que veremos logo a seguir, vamos também explorar durante o vídeo essas opções.

É nessa barra que se encontra o campo da imagem abaixo, chamado de "Campo de Comando, nele vamos acessar as transações do SAP com códigos, que são atalhos para os objetos presentes no SAP.



Existem alguns códigos que usamos para interagir com as transações do SAP, veja abaixo quais são e como funcionam:

Os seguintes comandos podem ser transferidos para este campo com **ENTER**:

- **Chamar uma transação**
  - **no mesmo modo (janela)**  
Entrar: **/nxxxx** (xxxx = código de transação).
  - **no mesmo modo (janela), a primeira tela é ignorada.**  
Entrar: **/\*xxxx** (xxxx = código de transação).
  - **em um modo adicional**  
Entrar: **/oxxxx** (xxxx = código de transação).

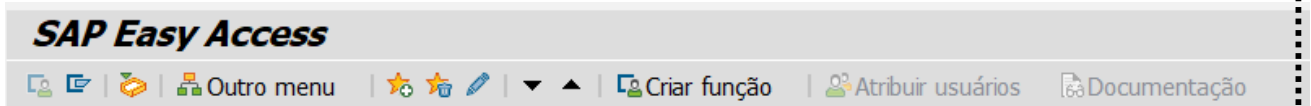


- **Encerrar a transação atual**  
Entrar: **/n.**  
Atenção: modificações não gravadas são perdidas sem aviso.
- **Eliminar o modo atual.**  
Entrar: **/i.**
- **Gerar uma lista de modos**  
Entrar: **/o.**
- **Encerrar a transação atual e voltar ao menu inicial**  
Entrar: **/ns000.**
- **Logoff do sistema**  
Entrar: **/nend.**
- **Logoff do sistema sem consulta de segurança**  
Entrar: **/nex.**

Atenção: modificações não gravadas são perdidas sem aviso.

A qualquer momento, clicando sobre qualquer objeto do SAP, você pode pressionar F1 no teclado para obter ajuda sobre este objeto, além de funcionalidades e dicas.

Barra de botões da tela:



Nesta barra, os botões variam bastante, pois em cada transação, aplicação ou função técnica ela terá os componentes que pertencem a esses objetos, vamos explorar a barra de botões da tela em algumas aplicações durante o vídeo para nos familiarizarmos.

- **Conhecendo as principais transações**

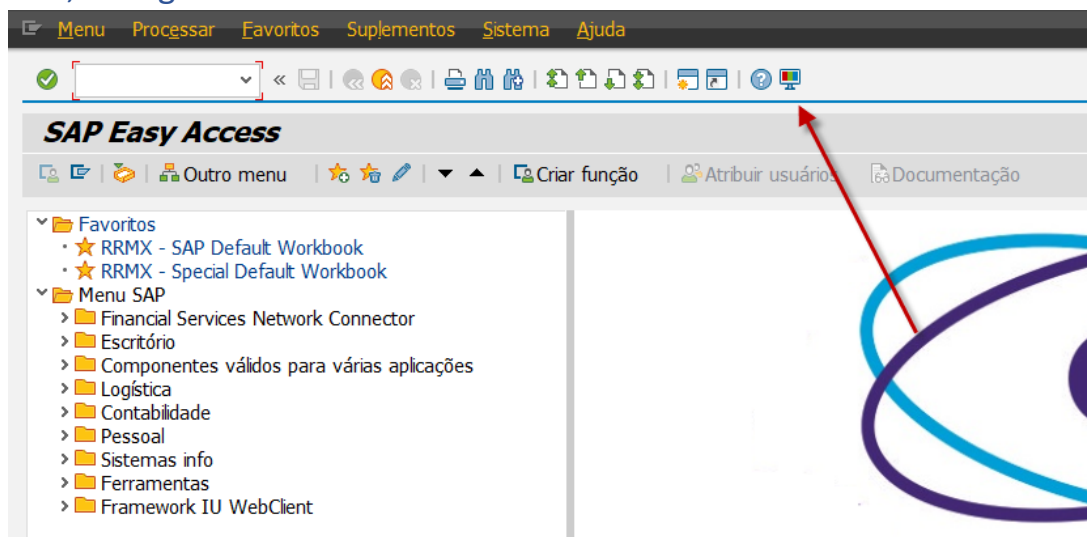
Para conhecer as principais transações do SAP, acesse o documento abaixo:



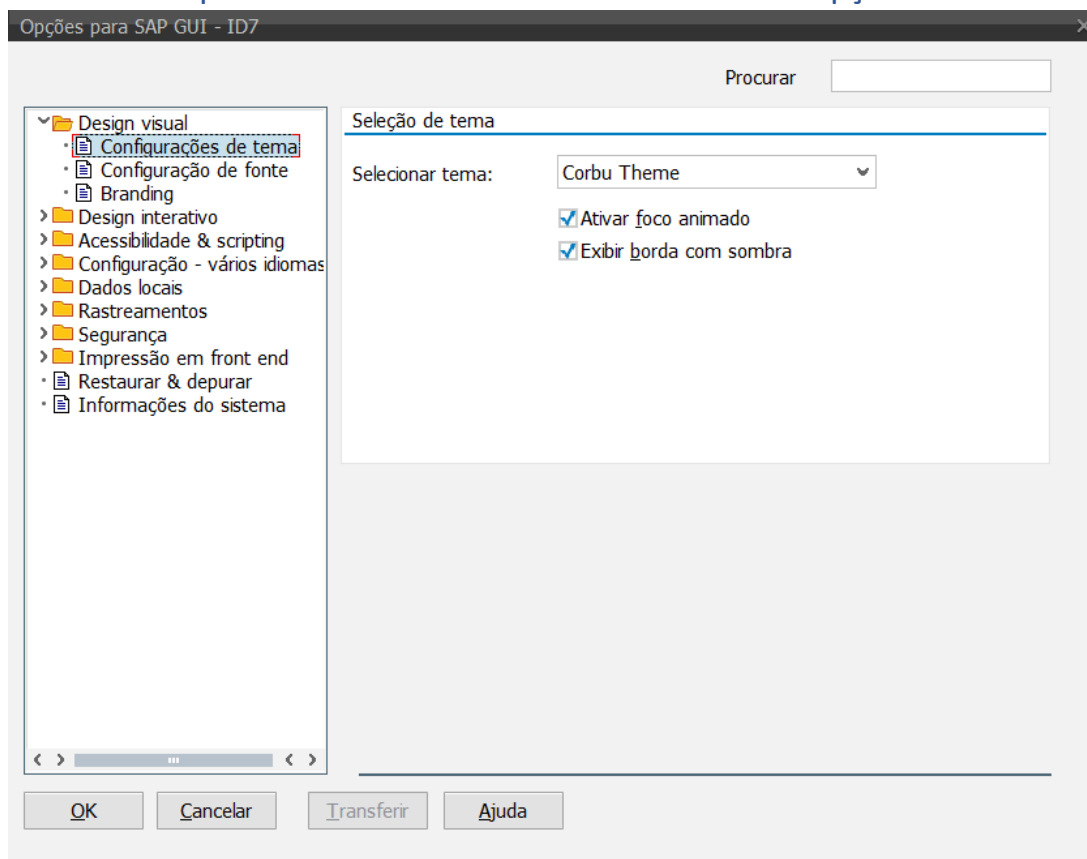
Principais  
Transações do SAP.pdf

- Dicas de como customizar seu SAP Gui de acordo com sua necessidade:

Vamos agora explorar um pouco das opções para nosso acesso ao ambiente SAP, navegue até o botão:



Ao clicar nesse botão, a tela abaixo será exibida, vamos ver alguns detalhes durante o vídeo para entender a funcionalidade de cada opção:



## 5. Diferenças entre o SAP ECC/Hana/Fiori

### ○ Frontend

É o ambiente que hospeda o Fiori para o SAP, nele você pode configurar e ajustar o Fiori de acordo com a necessidade de cada empresa, este ambiente não possui as transações convencionais do **SAP**.

### ○ Backend

É a máquina que está conectada ao ambiente de Fiori, normalmente um ambiente como o que estamos acessando nessa parte do curso, ela possui conexões entre o Fiori e o ECC – S/4Hana para enviar as informações ao Frontend (Fiori).

### ○ ECC com base de dados S/4

Existem ambientes SAP ECC que tem o banco de dados totalmente ou parcialmente em Hana, isso faz com que o ECC que é um sistema inferior ao ambiente S/4Hana tenha a mesma conexão e velocidade do S/4.

### ○ ECC com base de dados MySQL ou Oracle

Esse ECC tem base de dados como a maioria das aplicações no mercado, voltadas a MySQL, Oracle dentre outros, não possui a velocidade de processamento do Hana. Mesmo que não seja a velocidade do Hana, ainda sim o SAP processa os dados muito rapidamente, sendo hoje a líder mundial em CRM, ERP e SCM.

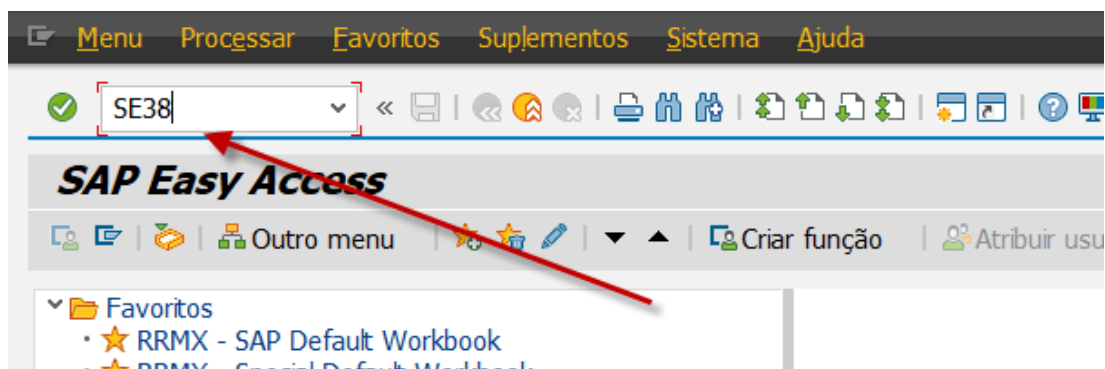
### ○ S/4 Puro

Esse é o ambiente S/4Hana com base de dados em Hana, é o que há de melhor em processamento em nuvem para o mercado atual, o banco de dados Hana também pode ser usado por outras empresas, assim como era com o MySQL e SCM.

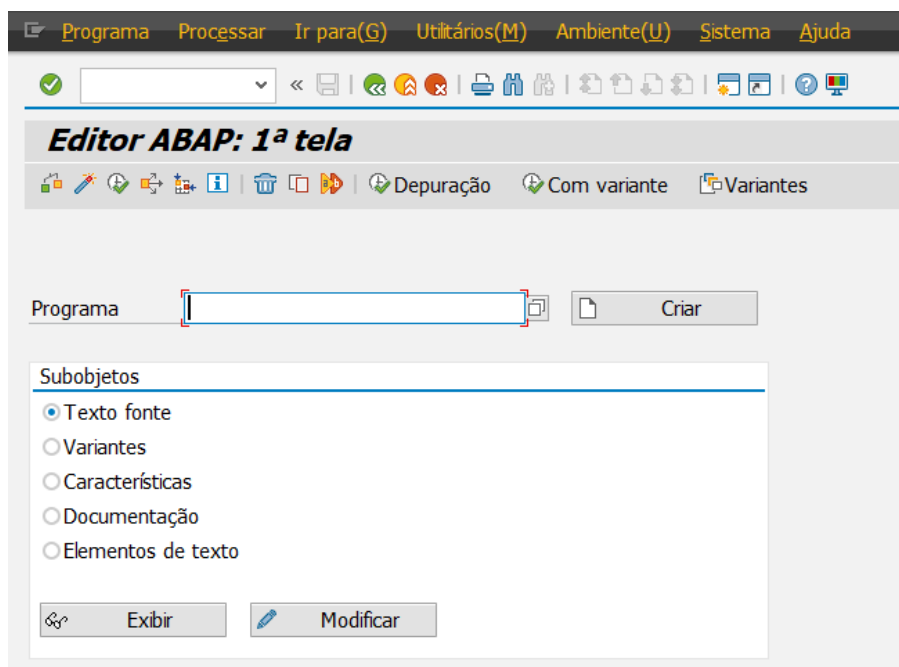
## 6. Conhecendo o ABAP

### ➤ Criando seu primeiro programa em ABAP (SE38)

Depois de toda a teoria, vamos agora começar a praticar nosso conhecimento em ABAP, para isso, vamos informar a transação SE38 no campo de comando, na parte superior esquerda da tela:



Será exibida a transação de edição de códigos do SAP:

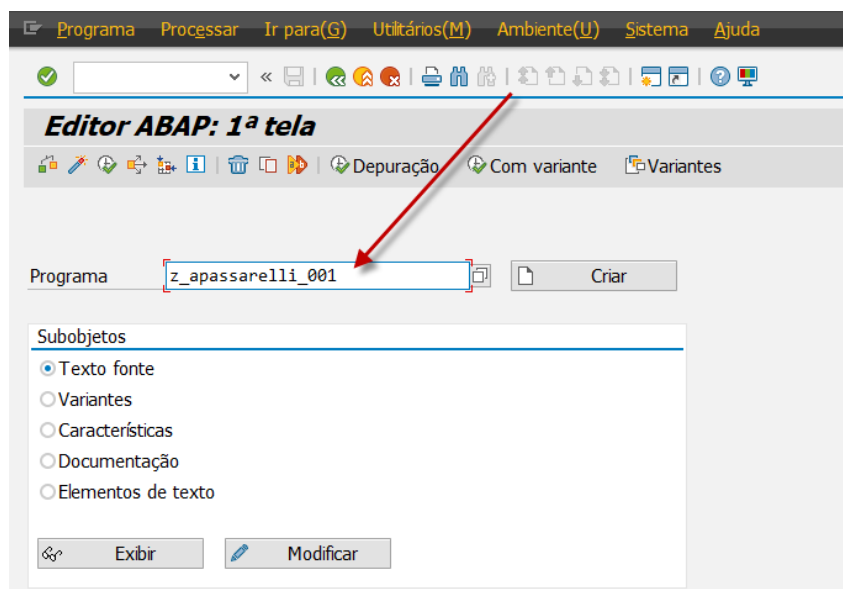


Nessa tela acima, você pode informar o nome de desenvolvimentos ABAP que já tenham sido criados, para exibir seu código fonte ou até mesmo editar o código, também é aqui que criaremos nosso primeiro desenvolvimento em ABAP, para

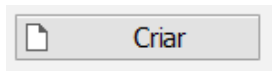
isso, vamos usar a letra Z ou Y para iniciar indicar o nome do nosso programa em seguida, informe seu usuário e o sequencial 001, por exemplo:

**Usuário:** APASSARELLI

**Programa:** ZAPASSARELLI001 ou Z\_APASSARELLI\_001



Após concluir esta etapa, clique no botão criar:



Conforme demonstrado na tela abaixo, informe a descrição do seu programa ABAP, essa descrição deve se referir ao tipo de processamento realizado no programa, como por exemplo: “Relatório de Faturas”, logo em seguida temos o tipo de programa, para começar a aula, usaremos o tipo “**1- Programa Executável**”, porém durante o curso estaremos trabalhando com os seguintes tipos:

**F** - Grupo de Funções

**I** - Programa Include

**M** - Pool de módulos ou Module Pool

Logo abaixo do tipo do programa, encontramos a opção “Status”, que serve para informar a classificação do programa, como por exemplo:

K	Programa de clientes produtivo
P	Programa standard-SAP produtivo
S	Programa do sistema
T	Programa de teste

Esse é apenas um status informativo, para programas de teste utilizamos a opção “T” e quando criarmos um programa para Clientes, informamos que o programa é do tipo “K”.

Já na última opção que vamos detalhar, você pode marcar o “Bloqueio de Editor” fazendo com que só você possa alterar ou eliminar seu programa, se alguma pessoa com usuário diferente do que foi criado o programa tentar fazer alguma dessas ações, essas serão bloqueadas.

ABAP: características do programa Z\_APASSARELLI\_001 modificar

Título

Idioma original PT Português

Criado APASSARELLI 06.09.2019

Última modific.

Status

Atributos

Tipo

Status

Aplicação

Grupo autorizações

☐ Bloqueio do editor ☒ Aritmética em ponto fixo

☒ Verifs.unicode ativas

Gravar

A tela deve ficar conforme a imagem abaixo:

ABAP: características do programa Z\_APASSARELLI\_001 modificar

Título: Meu primeiro programa

Idioma original: PT Português

Criado: APASSARELLI 06.09.2019

Última modific.:

Status:

Atributos

Tipo: 1 Programa executável

Status: T Programa de teste

Aplicação:

Grupo autorizações:

Banco de dados lógico:

Versão tela seleção:

☒ Bloqueio do editor ☒ Aritmética em ponto fixo

☒ Verifs.unicode ativas ☐ Início via variante

Gravar

Após informada todas as opções, clique em:

Será exibida uma tela como abaixo:

Criar entrada catálogo objetos

Objeto: R3TR PROG Z\_APASSARELLI\_001

Atributos

Pacote:

Responsável: APASSARELLI

Sistema de origem: ID7

Idioma original: PT Português

Data de criação:

Objeto local


Vamos entender as opções destacadas na imagem anterior:

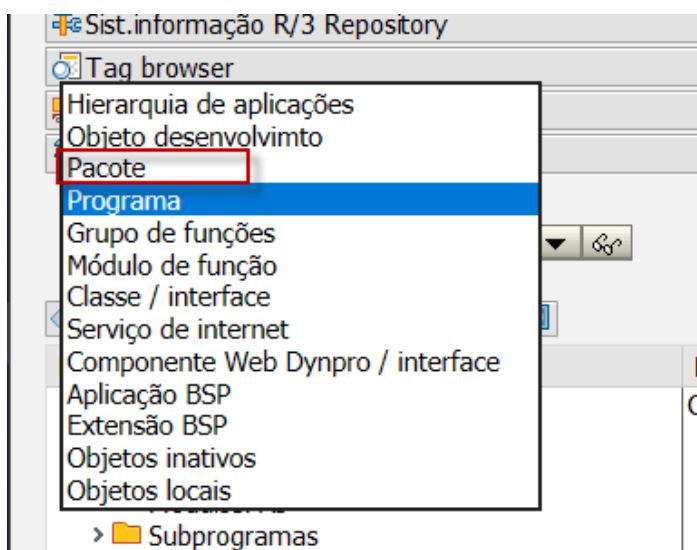
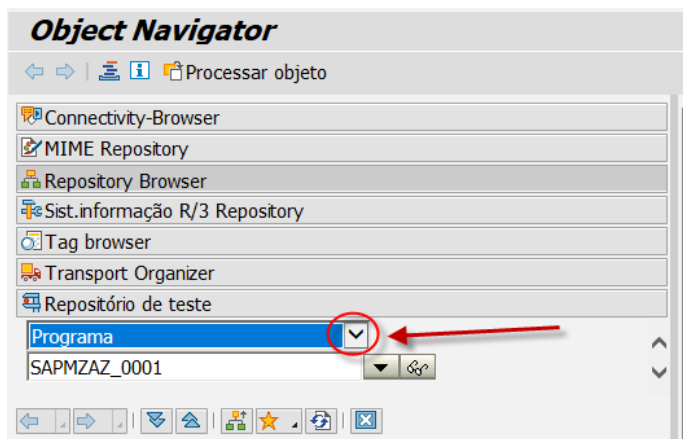
- ✓ **Objeto:** São as classificações técnicas de cada objeto do SAP, todos os objetos são atribuídos a tipos específicos para serem organizados dentro do ambiente.
- ✓ **Pacote:** Cada aplicação do SAP pode ser associada a um pacote, que agrupa todos os objetos de um projeto específico, ou de algum tipo de módulo, normalmente em clientes usamos os pacotes ZDEV para desenvolvimentos ABAP gerais, ou ZXX (XX = Módulo SAP), por exemplo ZMM para classificar todos os objetos de um mesmo módulo em um único ponto.
- ✓ **Responsável:** É o usuário que está criando o objeto, essa informação fica registrada no log de sistemas.
- ✓ **Sistema de Origem:** É o nome da máquina usada para hospedar o ambiente SAP.  
Idioma Original: É o idioma em que o programa foi criado, baseado no idioma da tela de seleção.  
  
**Data de Criação:** É preenchida após o programa ser criado, para ficar registrada no log de sistemas
- ✓ **Objeto Local:** Usamos para não associar o objeto a qualquer pacote, esses programas ficam gravados apenas dentro do ambiente e não geram request para o programa criado, vamos entender melhor o conceito de requests informando o pacote ZDEV em nosso primeiro programa.



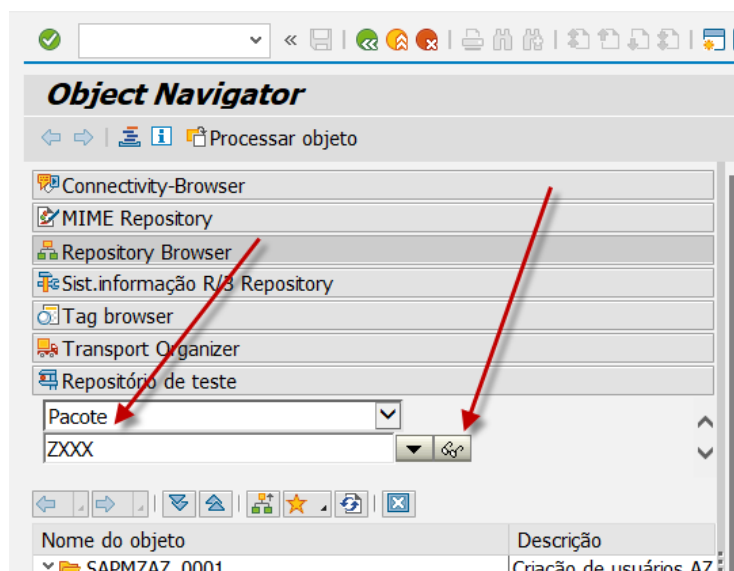
## 7. Como criar pacote

Para criar um pacote, podemos ir até a transação SE80, selecionar o objeto pacote e seguir os passos descritos abaixo:

Transação:  SE80



Substitua o XXX pelas iniciais de seu nome e clique em **Exibir:** 



## 8. Explicação sobre requests no SAP:

As requests são os pacotes que transportam os objetos criados ou configurações entre os ambientes, como falamos na primeira parte da aula, existem vários ambientes SAP e cada um pode ser um computador diferente e para que os objetos sigam de um computador para outro, uma rota é criada através de algumas configurações, essa parte é feita pela equipe de BASIS que cuida de toda infraestrutura do SAP. Temos uma transação específica para gerenciar e criar “Requests” a transação **SE09** que veremos como funciona logo adiante, abaixo vamos conhecer os tipos de requests mais usados que existem no SAP.

### ○ Workbench

A request “Workbench” é a mais comum dentre as requests, pois todos objetos criados como: dicionário de dados, códigos abap, formulários, etc., é a request que será mais usada por nós ABAPs, embora como veremos a seguir, existam ainda outros tipos de request envolvendo a área técnica, como veremos a seguir.

### ○ Transporte de cópias

A request de “Transporte de Cópia” serve para tirarmos uma cópia de uma Request Workbench e transportar os objetos sem que seja necessário que a

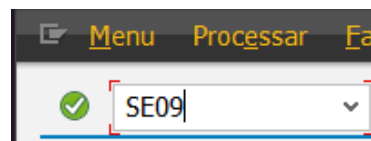
requests principal seja transportada, isso é usado para diminuir a margem de erros no transporte de requests para o ambiente “PRD” e garante que todos os objetos fiquem em apenas uma request no final.

- **Customizing**

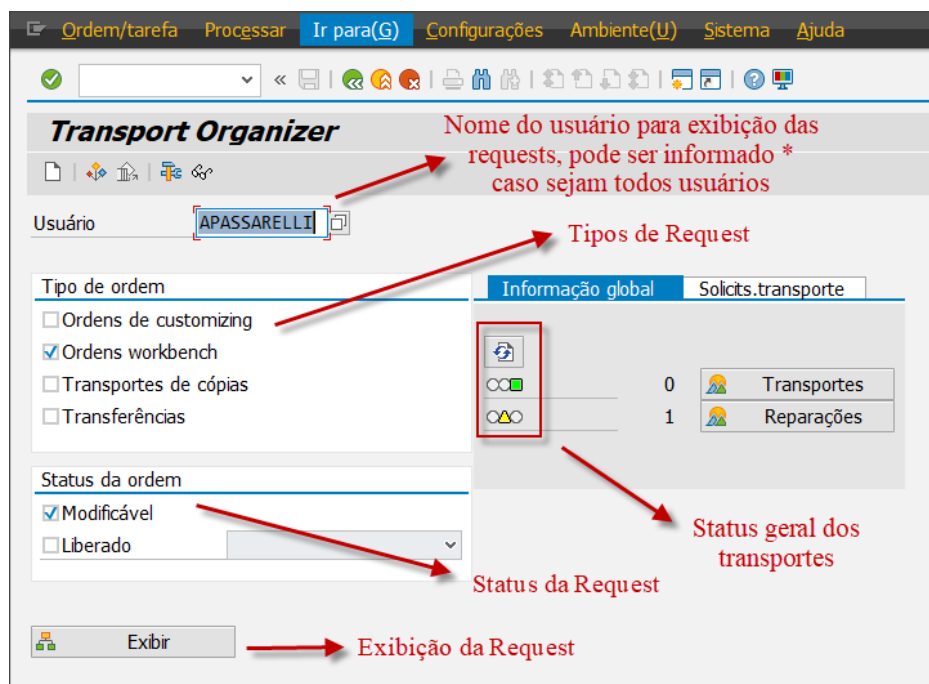
Essa request é usada para gravar as configurações realizadas pela equipe Funcional e também pode ser usada para diversas outras configurações de todos os módulos, inclusive os que envolvem ABAP, nela não existem códigos, ou objetos que pertençam a códigos, apenas dados de tabelas são transportados nesse tipo de request.

- **Transações relacionadas a Request (SE03/SE09)**

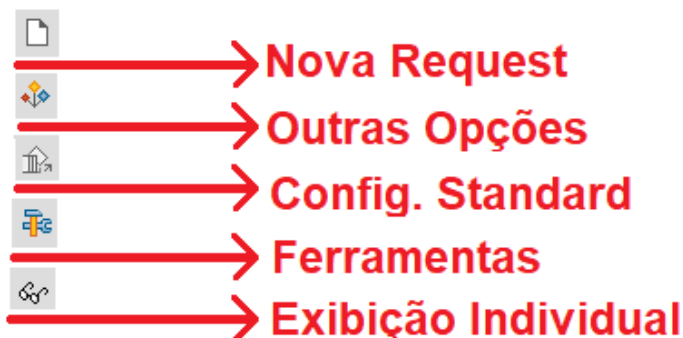
Vamos agora aprender como gerenciar as requests no SAP, para isso, informe a transação SE09 no “campo de comando”, conforme a imagem:



Será exibida a tela abaixo:



E também temos a “Barra de botões da tela”, com os seguintes campos:



Veremos durante o vídeo cada opção separadamente, assim como os detalhes e como funcionam.

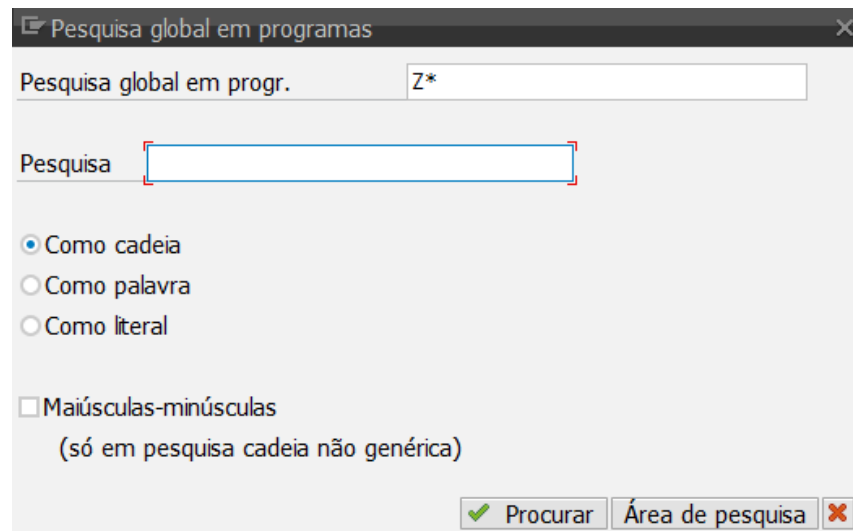
## 9. Utilizando base de programas já existentes no SAP

Sempre que um ABAP vai desenvolver um novo código, ou fazer uma lógica para ajustar os códigos já existentes é preciso pensar no que será feito, construir todas as partes do programa e como muitas lógicas podem ser reutilizadas, é muito bacana que busquemos códigos que já atendem ao que precisamos fazer, isso pode otimizar o nosso tempo durante o desenvolvimento de uma demanda além de nos apresentar outras ideias de outros programadores para o que estamos tentando fazer especificamente.

Para pesquisarmos o que precisamos em outras lógicas de outros programas, vamos conhecer duas formas três das diversas formas que você mesmo pode desenvolver para conseguir manter uma base de dados, como por exemplo, gravar todos os seus programas realizados em arquivos “TXT”, mantendo esses arquivos sempre com você, assim você poderá reutilizar no futuro, caso passe pelo mesmo cenário.

### ○ **EWK1**

A transação EWK1 faz uma busca em todos os objetos que envolvam um código, procurando pela instrução específica informada pelo usuário, veja abaixo como funciona:



○ **Cadeia de pesquisa:**

Na pesquisa global (ou seja, pesquisa no include geral ou programa) é possível uma entrada genérica com + e \*. Símbolo de escape é #, ou seja #+, #\* e ## para a pesquisa do caractere +, \* ou #. Não é permitida qualquer cadeia de pesquisa genérica para substituição e para pesquisa local. Caso a cadeia de pesquisa tenha espaços em branco precedentes ou subsequentes, tem de estar incluída em caracteres especiais ( /-.,;? ).

O tipo da pesquisa depende do método de pesquisa selecionado:

- **como cadeia** = cadeia de pesquisa é procurada no texto como uma sequência parcial qualquer
- **como palavra** = cadeia de pesquisa é procurada como palavra no sentido do léxico ABAP.

Caso as maiúsculas/minúsculas estejam marcadas, a cadeia de pesquisa é procurada na ortografia entrada. De outro modo, é procurada independentemente das maiúsculas/minúsculas. Na cadeia de pesquisa genérica, a procura é sempre efetuada independentemente da ortografia. No caso da pesquisa por palavras, não se pode distinguir entre maiúsculas/minúsculas.

○ **Área de pesquisa:**

A área de pesquisa pode ainda ser processada. O ícone "Seleção múltipla" no popup de pesquisa junto ao nome do programa/classe conduz a um outro popup com os componentes da área de pesquisa. Os componentes serão retirados da

área de pesquisa, quando a sua marcação for anulada, possibilitando a inserção de novos componentes.

Na pesquisa em classes, o botão "Objetos herdados" no popup da área de pesquisa permite exibir todos os componentes herdados e transferi-los para a área de pesquisa. São apenas oferecidos os componentes válidos na classe a ser pesquisada, ou seja, componentes em excesso após a redefinição serão ocultados.

### Utilização

Entrada genérica com + e \*, mas não para a substituição. O símbolo de escape é #, portanto #+ e #\* para a pesquisa do caractere + ou \*.

#### O tipo de pesquisa depende do método de pesquisa selecionado:

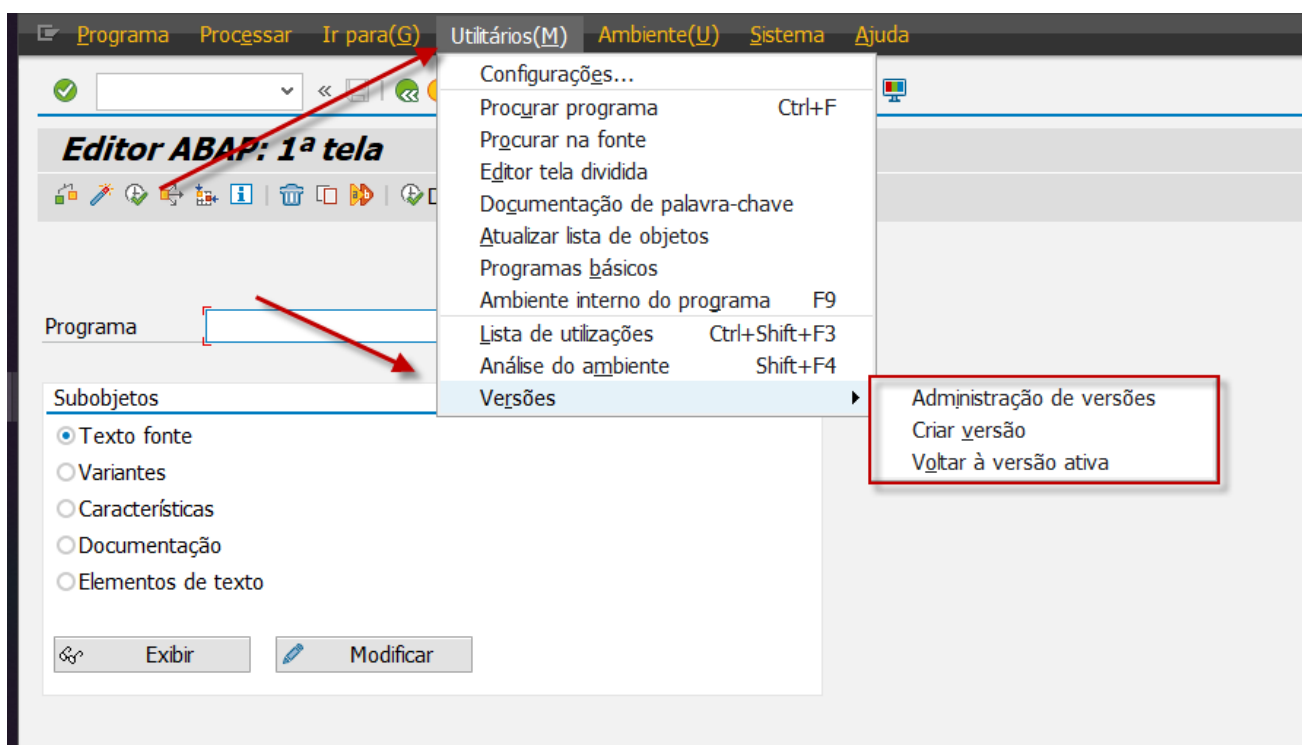
- **como cadeia** = cadeia de pesquisa, é procurada como sequência parcial no texto
- **como palavra**= cadeia de pesquisa, é procurada no sentido do léxico ABAP/4/

Se estiver marcado **Maiúsculas/Minúsculas**, a cadeia de pesquisa é procurada na ortografia. Caso contrário, a pesquisa é efetuada independentemente das maiúsculas e das minúsculas. Com uma cadeia de pesquisa genérica, a pesquisa é sempre efetuada independentemente da ortografia.

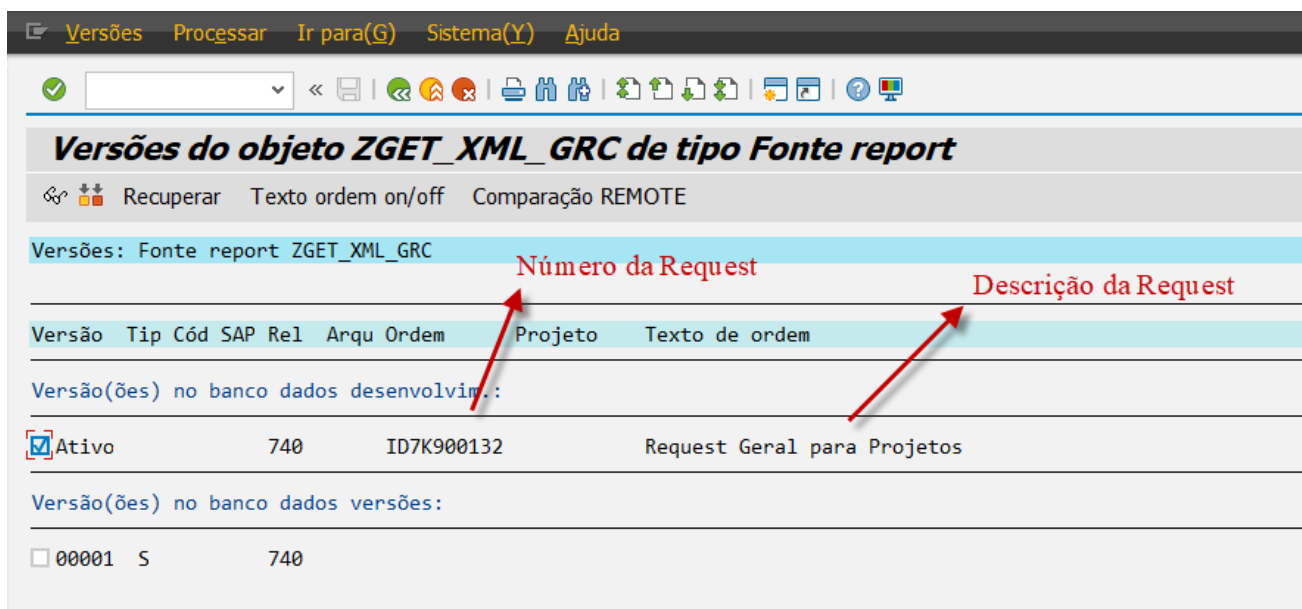
## 10. Versionamento de programas (Controle de versão)

A cada request gerada e transportada para um objeto do SAP, uma nova versão daquele objeto é criada, assim, caso alguma coisa aconteça de errado, você pode voltar a versão anterior para ajustar o que deu errado.

O versionamento de programas fica no “Menu Superior” em cada objeto, navegando até a opção “Versão”, conforme a imagem abaixo:



Nessa tela podemos verificar todas as versões transportadas ou ativas de um objeto SAP, durante o curso vamos explorar essas opções mais detalhadamente, além da barra de botões da tela, com a explicação de cada campo:

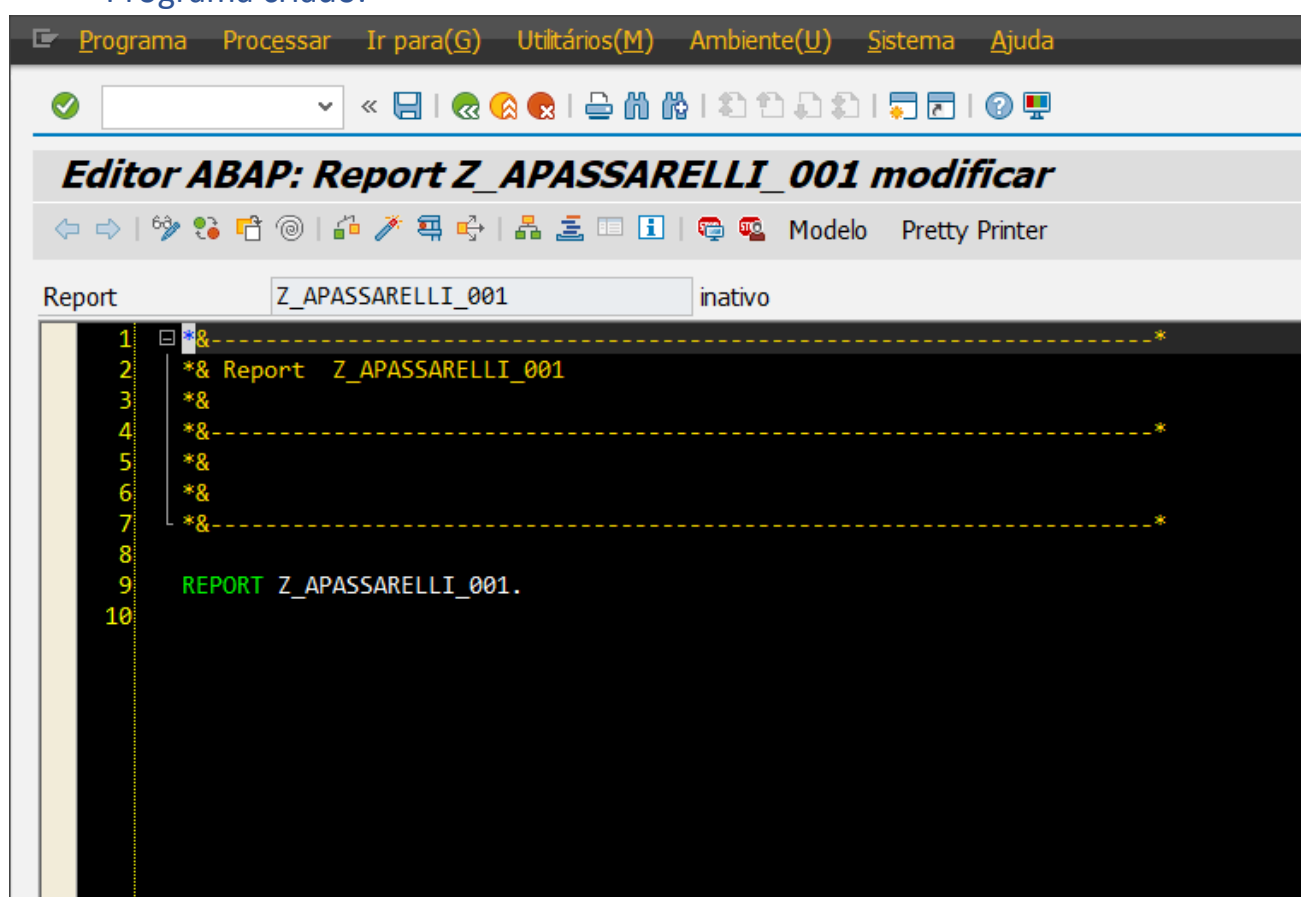


## 11. Entendendo a estrutura do código

Para codificar em ABAP, assim como em outras linguagens de programação, temos que respeitar algumas regras durante o desenvolvimento dos nossos programas, pois todo código tem sua estrutura genérica que não pode ser alterada, pois a ordem de cada lógica tem que ser encaixada perfeitamente em seu lugar, assim como veremos nos exemplos abaixo.

Ao criarmos um programa em ABAP, nenhum código além do nome do programa é informado no editor, todo o código é de responsabilidade do programador ABAP que ao conhecer as “Sintaxes” presentes na linguagem, consegue converter as solicitações dos clientes em códigos, para atender todas as áreas do SAP.

Programa criado:



Como vemos na imagem acima, somente o cabeçalho do programa é informado ao programa ser criado, precisamos entender que todos os objetos que farão parte de todo desenvolvimento devem e ficar na parte superior, pois o conceito



de um objeto criado é que sua utilização seja a partir de sua declaração, por exemplo:

**Declaração de Campo de Valor do tipo Valor.**

Campo de valor = 10,00.

Se usarmos a lógica da seguinte forma, não será possível ativar o código, pois o objeto ainda não foi criado onde está sendo “**alimentado com o valor de 10,00**”.

Campo de valor = 10,00

**Declaração de Campo de Valor do tipo Valor.** (Criação do objeto usado acima)

A lógica deve ser respeitada, ou seja, para colocarmos uma informação em algum lugar, esse lugar primeiramente precisa existir e só então poderemos trabalhar com ele, entendendo esse conceito, vemos que a maioria dos objetos de um programa será declarada no topo, logo abaixo do nome do programa, pois eles serão úteis em todo o código, os tornando “**Globais**”.

Logo abaixo, estaremos conhecendo os objetos existentes no SAP, assim como sua utilização em exemplos de códigos realizados durante a aula, mas antes, vamos entender o porque a lógica acima deve ser respeitada.

## 12. Explicação do conceito de programação procedural



Um programa ABAP sempre está lendo todas as linhas codificadas e seguindo com o fluxo desenvolvido, em programas procedurais eles partem de um ponto e avançam até o final, ou até alguma instrução que o faça parar, esse conceito de seguir o fluxo sempre para “baixo” é o que caracteriza uma programação procedural.

Existe um indicador que vai interpretando cada linha do código e as convertendo em informações



```

40 IF sy-subrc EQ 0.
41   SELECT *
42   FROM j_1bnfstx
43   INTO TABLE t_stx
44   WHERE docnum EQ <nf_item>-docnum AND
45         itmnum EQ <nf_item>-itmnum.
46
47   READ TABLE t_stx INTO w_stx WITH KEY taxtyp = 'ICM2'.
48
49 IF sy-subrc EQ 0.
50   CLEAR: cs_c100-vl_icms,
51         cs_c100-vl_bc_icms.
52   ENDIF.

```

### 13. Explicação do conceito de programação orientada a objeto



Quando falamos de orientação a objetos, fugimos totalmente do conceito de começo, meio e fim, pois toda aplicação depende de uma série de métodos e pode englobar diversos processos, como por exemplo, existe uma classe no SAP criada para calcular os impostos, ela se chama “CL\_TAX\_CALC\_BR”, nela cada método corresponde, por exemplo, a um tipo de imposto, onde o cálculo é feito apenas nesse método e todas as lógicas necessárias, isso facilita na hora de construir aplicações complexas, que dependam desses tipos de processos. Como veremos diversos métodos, abaixo, existem ICMS, outro IPI o um para calcular o outro o ISS e assim por diante, dessa forma, é como se vários mini programas fossem construídos a cada método, os campos possuem importação e campos de exportação para enviar ou receber dados de dentro dos métodos, essa configuração permite que todos os métodos que precisem se comunicar diretamente tenham a possibilidade de transferir dados através desse campos, uma outra vantagem é ter todos os objetos dentro de um só local, assim se for necessário usar alguns desses objetos em diversos locais essas opções de parâmetros podem ajudar com essa questão.



## Classe: CL\_TAX\_CALC\_BR

### Class Builder: exibir classe CL\_TAX\_CALC\_BR

Definições locais/implementações				Documentação de classes
Classe/interface CL_TAX_CALC_BR realizado / Ativo				
Caracts.	Interfaces	Friends	Atributos	Métodos
<div> <div>Parâmetro</div> <div>Exceções</div> <div>Txt. fonte</div> <div>Filtro</div> </div>				
Método	Tipo	Visibilidade	T...	Descrição
SET_CTE_IS_ACTIVE	Instance Method	Public		Determine CT-e active flag
GET_TAX_DATA_FINAL	Instance Method	Public		Get tax rate and final tax base
GET_TAX_ACTIVE_COND	Static Method	Public		Returns condition type for active tax group
CALCULATE	Instance Method	Public		Calculate taxes: Call from outside
CONSTRUCTOR	Instance Method	Public		Constructor
GET_CONDITION_CODE	Instance Method	Public		Get internal code assigned to condition type
GET_RESULT	Instance Method	Public		Returns calculated tax values
GET_ROUNDING_CONFIGURATION	Instance Method	Public		Get Rounding Configuration for Unit Price(Abstract)
CALCULATE_ICMS_FOR_CTE	Instance Method	Protected		Calculate ICMS for CTe
CALCULATE_ICMS_ST_FOR_CTE	Instance Method	Protected		Calculate ICMS ST for CTe
CALCULATE_PIS_COF_BASE_CONFIG	Instance Method	Protected		Calculate configurable PIS / COFINS base
ADD_TAX_DATA	Instance Method	Protected		Derive additional tax data
CHECK_CTE_ACTIVE	Instance Method	Protected		Returns 'X' if document is Conhecimento and switch is on
CALCULATE_ICMS	Instance Method	Protected		Calculate ICMS
CALCULATE_ICMS_FREIGHT	Instance Method	Protected		Calculate ICMS on freight
CALCULATE_ICMS_IPI_BASES	Instance Method	Protected		Calculate ICMS and IPI base amounts
CALC_ICMS_CTE_VALUES	Instance Method	Protected		Calculate ICMS for Conhecimento
CALCULATE_IPI	Instance Method	Protected		Calculate IPI
CALCULATE_ISS	Instance Method	Protected		Calculate ISS
CALCULATE_SUBTRIB	Instance Method	Protected		Calculate Substituição Tributária
CALCULATE_SUBTRIB_FREIGHT	Instance Method	Protected		Calculate sub.trib. on freight

## Método para calcular o ICMS: CALCULATE\_ICMS

### Class Builder: Classe CL\_TAX\_CALC\_BR exibir

Modelo Pretty Printer Assinatura Seção pública Seção protegida Seção privada			
Tp.	Parâmetro	Atrib. tipo	Descrição
	EV_VAL_INCL_ICMS	TYPE MTY_TAXAMOUNT	Nettowert + ICMS
	EV_DIS_INCL_ICMS	TYPE MTY_TAXAMOUNT	Rabatt + ICMS

Método	CALCULATE_ICMS	ativo
--------	----------------	-------

```

1 METHOD calculate_icms .
2
3 * Check if ICMS is to be calculated
4 IF check_icms_active( ) <> 'X'.
5 * ICMS is not active
6
7     ev_val_incl_icms = mv_adjusted_net.
8     ev_dis_incl_icms = ms_tax_data-discount.
9     RETURN.
10
11 ENDIF.
12
13 * Calculation is different depending on usage
14 * (Industrialization / Consumption)
15 CASE ms_tax_control-usage.
16
17     WHEN mc_indust.
18 * ICMS Industrialization
19     CALL METHOD calc_icms_indus
20     IMPORTING
21         ev_val_incl_icms = ev_val_incl_icms
22         ev_dis_incl_icms = ev_dis_incl_icms.
23

```

Parâmetros para Import ou Export

## 14. Objetos, Sintaxes e Lógica de programação

### ➤ Objetos ABAP

#### ○ Variáveis

Tipo	Função
ANY	Qualquer tipo de dados
ANY TABLE	Tabela interna com qualquer tipo de tabela
C	Campo de texto com um comprimento genérico
CLIKE	Estruturas semelhantes a caracteres (c, d, n, t, sequência e estruturas planas semelhantes a caracteres); em programas não-Unicode também x, xstring e qualquer estrutura plana
CSEQUENCE	Tipo de texto (c, string)
DATA	Qualquer tipo de dados
DECFLOAT	Número decimal de ponto flutuante
HASHED TABLE	Tabela do tipo Hashed
INDEX TABLE	Tabela de Índice
N	Texto numérico com comprimento genérico
NUMERIC	Numérico (i (b, s), p, decfloat16, decfloat34, f)
OBJECT	Qualquer tipo de objeto (classe raiz da hierarquia de herança)
P	Campo de valor com comprimento genérico e número genérico de casas decimais
SIMPLE	Tipo de dados elementar, incluindo tipos estruturados com componentes planos exclusivamente para caracteres
SORTED TABLE	Tabela Ordenada
STANDARD TABLE	Tabela Standard
TABLE	Tabela Standard
X	Campo de bytes com comprimento genérico
XSEQUENCE	Campo de bytes (x, xstring)

**DATA:** v\_variavel\_1 TYPE c,  
v\_variavel\_2(10) TYPE c,  
v\_variavel\_3 TYPE mara-matnr,  
v\_variavel\_4 TYPE n,  
v\_variavel\_5 TYPE p DECIMALS 2,  
v\_variavel\_6 TYPE d,  
v\_variavel\_7 TYPE t,  
v\_variavel\_8 TYPE float,  
v\_variavel\_9 TYPE i,  
v\_variavel\_10 TYPE string.

- Estruturas

Estruturas são um conjunto de variáveis, ou seja, ao invés de podermos armazenar apenas uma informação por vez no objeto, na estrutura podemos inserir várias informações limitadas a uma linha, conforme o exemplo do no Excel abaixo:

A	B	C	D	E	F	G
	Variável					
	Estrutura					

**DATA:** e\_estrutura **TYPE** mara.

Conforme podemos notar, a estrutura tem várias colunas, mas contém apenas uma linha, esse tipo de objeto é ideal na hora de selecionar valores únicos em tabelas, ou seja, apenas uma linha da tabela, ou para armazenar as informações durante a execução do código ABAP, ela pode ser criada com base em dados de tabelas ou até mesmo criada internamente no programa ABAP.

- Tabelas internas / Tabelas Transparentes

Diferente da variável e da estrutura, uma tabela interna pode ter várias colunas e também várias linhas, conforme o exemplo no Excel apresentado abaixo:

A	B	C	D	E	F	G
	Variável					
	Estrutura					
	Tabela					

**DATA:** t\_tabela **TYPE** TABLE **OF** mara.

Conforme vemos no modelo acima, uma tabela é a soma de várias estruturas, podemos então entender que a sequência lógica dos objetos primários do ABAP é: “Variável, estrutura e depois tabela interna”, conforme também o desenho acima.

- Constantes

Com a mesma configuração da variável, podendo ser de todos os tipos, tamanhos, porém ao usar constantes, nós estamos definindo textos fixos dentro de um código o que são chamados de “literais”, uma literal sem constante é considerada uma má prática de programação, pois a utilização de constantes facilita na hora de dar manutenção ao programa de forma geral.

Abaixo segue um exemplo de um código usando literais sem constantes, os chamados “Hard Codes”:

```
MESSAGE 'Mensagem Literal' TYPE 'E'.
```

MESSAGE 'Mensagem Literal' TYPE 'E'.

Exemplo com constante:

```
39  CONSTANTS: c_mensagem(22) TYPE c VALUE 'Mensagem com constante'.  
40  
41  MESSAGE c_mensagem TYPE 'E'.
```

CONSTANTS: c\_mensagem(22) TYPE c VALUE 'Mensagem com constante'.

MESSAGE c\_mensagem TYPE 'E'.

Conforme o exemplo acima, usamos uma constante para criar a mensagem, especificamos exatamente o tamanho da mensagem e inserimos os tipos, exatamente como uma variável, porém, essa de valor fixo e imutável.

- Ranges

Diferente de valores fixos únicos como variáveis, os ranges podem determinar um intervalo “de/até” de informações a serem usadas internamente, por exemplo.

Podemos buscar uma data de: 01/01/2019 até 31/08/2019, o “de” seria a primeira data e o “para” seria a segunda, vejamos na aula mais detalhes de como funcionam os ranges.

- **Field Symbols**

Embora sejam um pouco complicados para esta parte do curso, vamos já nos ambientar com os “ponteiros”. Através de um ponteiro é possível transformar a informação de um objeto para essa ser usada em diversos locais, como esse conceito é um pouco complicado, vamos entender melhor durante a explicação no vídeo, pois voltaremos a usar esse conceito um pouco mais tarde.

- Parameter

Para possibilitar a entrada de dados de usuários em programas, podemos usar o comando PARAMETER, nele criamos uma entrada personalizada de acordo com suas configurações técnicas, também igual as variáveis em tipos, tamanhos, etc. Esses campos só podem aceitar um valor de cada vez, respeitando sua limitação e tipo associado.

```
12 ▶ PARAMETERS: p_campo1(10) TYPE c.
```

PARAMETERS: p\_campo1(10) TYPE c.

- Select Options

Ao contrário do PARAMETER, o SELECT-OPTIONS é usado para mais de um valor por vez em uma entrada de usuário, ou seja, ele permite que o usuário informe vários dados na entrada além de ranges como vimos acima, durante a aula veremos detalhado como esse comando funciona.

```
346 ▶  
347 ▶ TABLES: mara.  
348 ▶  
349 ▶ SELECT-OPTIONS: s_matnr FOR mara-matnr.
```

TABLES: MARA.

**SELECT-OPTIONS:** s\_matnr **FOR** mara-matnr.

### ➤ Telas de seleção

Para um usuário interagir com o programa, é necessário que sejam criados parâmetros de entrada, para que os dados possam ser digitados ou até mesmo selecionados para a execução parcial ou completa do programa.

Existem diversos tipos de tela de seleção, pois cada processo terá uma definição diferente e assim, a tela também será criada de acordo com cada necessidade.

Exemplo de telas de seleção: (Programa: **Z\_APASSARELLI\_004\_1**)

The image displays two examples of SAP selection screens. The first screen, titled 'Tela de seleção 1', features a single input field labeled 'Entrada'. The second screen, titled 'Tela de Seleção 2', includes a section header 'Informação de seleção' and a single input field labeled 'Entrada 2'.

#### ○ Opções da tela de seleção

Para tornar a tela de seleção mais completa e personalizada de acordo com o solicitado, podemos alterar diversas opções no comando para a chamada de tela de seleção: **“SELECTION-SCREEN”**.

Dentro de uma tela de seleção, além de poder personalizar como a tela será exibida, como foi dito anteriormente a tela de seleção pode conter objetos de entrada para o usuário, e para isso usamos os dois mais comuns para uma tela de seleção:

PARAMETER

SELECT-OPTIONS



Um PARAMETER é um parâmetro de entradas únicas, ou seja, é possível inserir apenas um registro de cada vez, se precisarmos de uma nova informação a anterior deverá ser apagada para que o novo valor seja aceito, exemplos de PARAMETERS:

#### Programa: Z\_APASSARELLI\_004\_6

**Programa de testes**

P\_MATNR

P\_MTART

P\_DATA

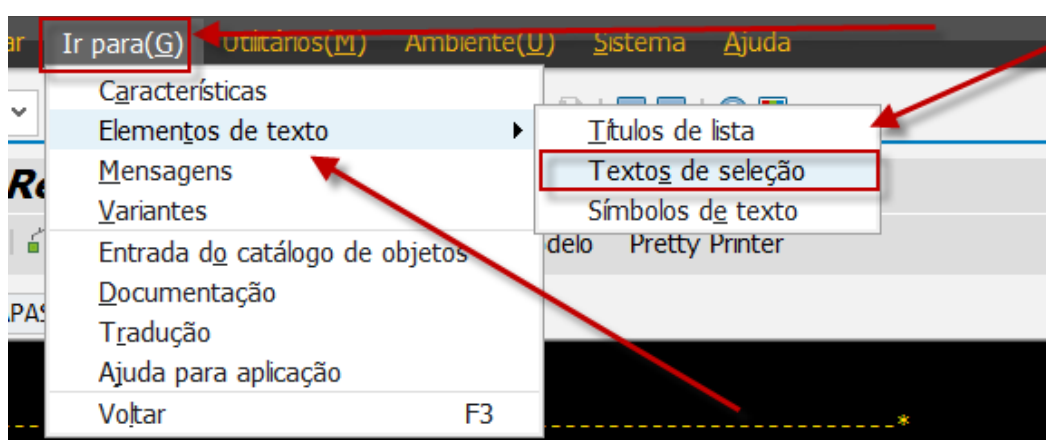
☒ P\_R1  
☐ P\_R2

☐ P\_CHECK1  
☐ P\_CHECK2

P\_LIST1

P\_LIST2

Acima vemos os parâmetros criados exibindo no programa seus nomes técnicos, pois ainda não definimos qual será o texto de exibição de cada um, para mudarmos o nome técnico para um texto personalizado, usamos o seguinte menu dentro da transação SE38:



Podemos inserir o nome que desejarmos para cada um dos campos, bastando substituir o “ ?... ” pelo nome que você deseja, mas lembre-se sempre de deixar

um nome bastante informativo, pois o usuário muitas vezes não tem qualquer conhecimento avançado sobre SAP, precisando que tudo seja muito informativo na hora que for descrito.

**Elementos de texto ABAP: modificar textos seleção idioma Português**

Programa  ativo

1 / 9

Nome	Texto	Refer.Dict...
P_CHECK1	?...	<input type="checkbox"/>
P_CHECK2	?...	<input type="checkbox"/>
P_DATA	?...	<input type="checkbox"/>
P_LIST1	?...	<input type="checkbox"/>
P_LIST2	?...	<input type="checkbox"/>
P_MATNR	?...	<input type="checkbox"/>
P_MTART	?...	<input type="checkbox"/>
P_R1	?...	<input type="checkbox"/>
P_R2	?...	<input type="checkbox"/>

Após informar os nomes de cada campo e ativar, os textos serão exibidos no lugar das informações técnicas do campo.

**Programa de testes**

☐

Material

Tipo de Material

Data de entrada

☒ Radio Button 1  
☐ Radio Button 2

☐ Check 1  
☐ Check 2

Lista 1


Lista 2




Ao contrário de um PARAMETER o SELECT-OPTIONS pode aceitar mais de um valor de uma vez, além de permitir que sejam definidos ranges

(de/até) que podem auxiliar muito na hora de pesquisar uma informação, vejamos abaixo alguns exemplos de SELECT-OPTIONS em telas de seleção:

Programa: **Z\_APASSARELLI\_004\_7**


**Programa de testes**






S_DATA	<input type="text"/>	até	<input type="text"/>	
S_MATNR	<input type="text"/>	até	<input type="text"/>	
S_WERKS	<input type="text"/>	até	<input type="text"/>	
S_EBELN	<input type="text"/>	até	<input type="text"/>	
S_EBELP	<input type="text"/>			

Após mudarmos os elementos de texto:

**Programa de testes**



Data	<input type="text"/>	até	<input type="text"/>	
Material	<input type="text"/>	até	<input type="text"/>	
Centro do Material	<input type="text"/>	até	<input type="text"/>	
Pedido	<input type="text"/>	até	<input type="text"/>	
Item do Pedido	<input type="text"/>			

### ○ Tela de seleção dinâmica

Uma tela de seleção dinâmica pode variar em sua exibição de acordo com opções selecionadas, no exemplo abaixo, temos uma tela de entrada para o usuário selecionar um caminho do **Windows** ou do **Servidor SAP**, onde está hospedado o sistema.

Ao selecionar a opção local, serão exibidas as opções para a seleção de Arquivo e ao selecionar a opção de servidor, serão exibidas só as opções para o servidor (imagens 1 e 2)

### Exemplo de tela dinâmica: (Programa: Z\_APASSARELLI\_004\_5)

**Programa de testes**

☒ Local  
☐ Servidor

Arquivo Local: C:\  
Arquivo do Servidor:

Cliente: até  
País: até

**Programa de testes**

☐ Local  
☒ Servidor

Arquivo do Servidor: /default

País: até

- **Eventos da tela de seleção**

Para entender a estrutura de um código, precisamos entender que existem diversos eventos na execução de um programa, isso define a estrutura lógica obrigatória que sempre vamos respeitar na construção de nossas demandas.

Por exemplo quando falamos acima da tela de seleção, ela marca o início de um programa, pois o usuário partirá dessa tela para as lógicas programadas para serem executadas após a ação do usuário.

Exemplos – Programa: **Z\_APASSARELLI\_004\_9**

- AT SELECTION-SCREEN ON
- START-OF-SELECTION
- INITIALIZATION

Vamos entender o conceito de eventos durante a explicação da aula de sintaxes em vídeo, para melhor experiência com essa ferramenta no código.

➤ **Sintaxes ABAP:**

➤ **Conhecendo o Help de Sintaxes (F1) e as sintaxes abaixo detalhadas com exemplos práticos:**

Exemplos das sintaxes no programa: **Z\_APASSARELLI\_004\_1**

- ADD
- APPEND
- AT FIRST
- AT LAST
- AT NEW
- AT SELECTION-SCREEN ON
- AT USER-COMMAND
- AUTHORITY-CHECK OBJECT
- ASSING f1 TO <f>
- CALL TRANSACTION
- CASE .. WHEN .. ENDCASE

- CHECK
- CLEAR
- COLLECT
- COMMIT WORK
- CONCATENATE
- CONTINUE
- CONDENSE
- DELETE
- DESCRIBE TABLE
- DO .. ENDDO
- EXIT
- FORM
- FREE
- IF .. ELSE .. ENDIF.
- IF NOT .. IS INITIAL
- INITIALIZATION
- INSERT
- INSERT LINES
- LEAVE PROGRAM
- LOOP AT
- MESSAGE
- MESSAGE-ID
- MODIFY
- MOVE
- MOVE-CORRESPONDING
- PARAMETERS
- PERFORM
- READ TABLE
- READ\_TEXT
- REFRESH
- REPLACE
- SAPGUI\_PROGRESS\_INDICATOR
- SEARCH
- SELECT
- SKIP
- SORT
- SPLIT .. AT .. INTO
- START-OF-SELECTION

- STRLEN
- SUM
- SY-BATCH
- SY-DATUM
- SY-LANGU
- SY-LINNO
- SY-LISEL
- SY-MANDT
- SY-PAGNO
- SY-SUBRC
- SY-TABIX
- SY-TVAR0 .. SY-TVAR9
- SY-UCOMM
- SY-UNAME
- SY-UZEIT
- SY-VLINE
- SY-ULINE
- SY-CPROG
- SY-TCODE
- SY-DBCNT
- TABLES
- TRANSLATE
- TYPES
- ULINE
- UNPACK
- UPDATE
- WHILE .. ENDWHILE
- WRITE

## Blocos/Performs:

Sempre que passamos a tela de seleção, definimos o programa em blocos, o que facilita a organização dos códigos dentro de um código fonte, esses blocos são chamados de PERFORMS.

Um PERFORM quando codificado gera um FORM, que é a sua sub sequência, a chamada PERFORM pode se fazer referência a diferentes FORMS, já que o código PERFORM é apenas o código que dá função a sequência de blocos, vejamos um exemplo abaixo:

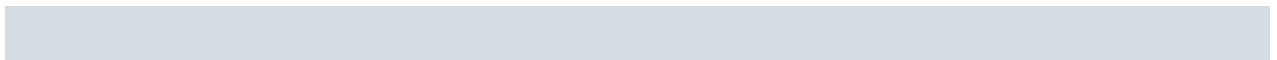
```
PERFORM: f_form1,  
        f_form2.
```

```
FORM f_form1.
```

```
ENDFORM.
```

```
FORM f_form2.
```

```
ENDFORM.
```





## 15. Lógica de programação

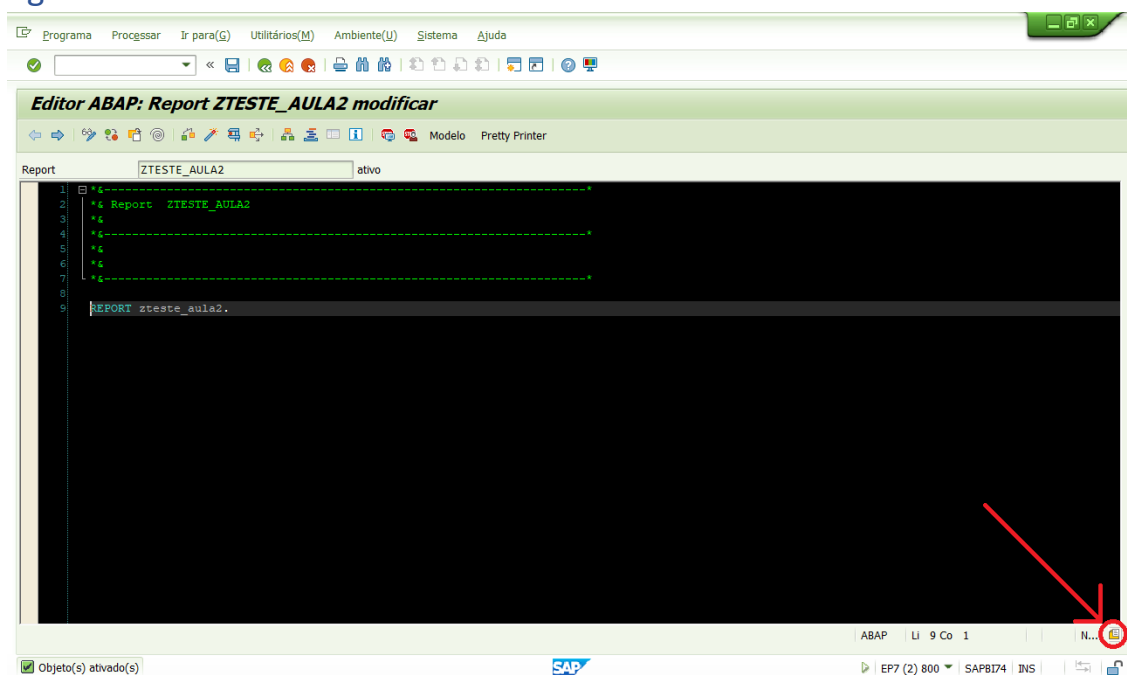
- Boas práticas ABAP
- Tornando o seu trabalho mais fácil (configuração de sintaxes pré-definidas)

Uma das dicas pouco conhecidas no SAP é configuração de sintaxes pré-definidas, todas as sintaxes do ABAP podem ser completadas com o uso da tecla TAB do teclado, conforme o exemplo abaixo:

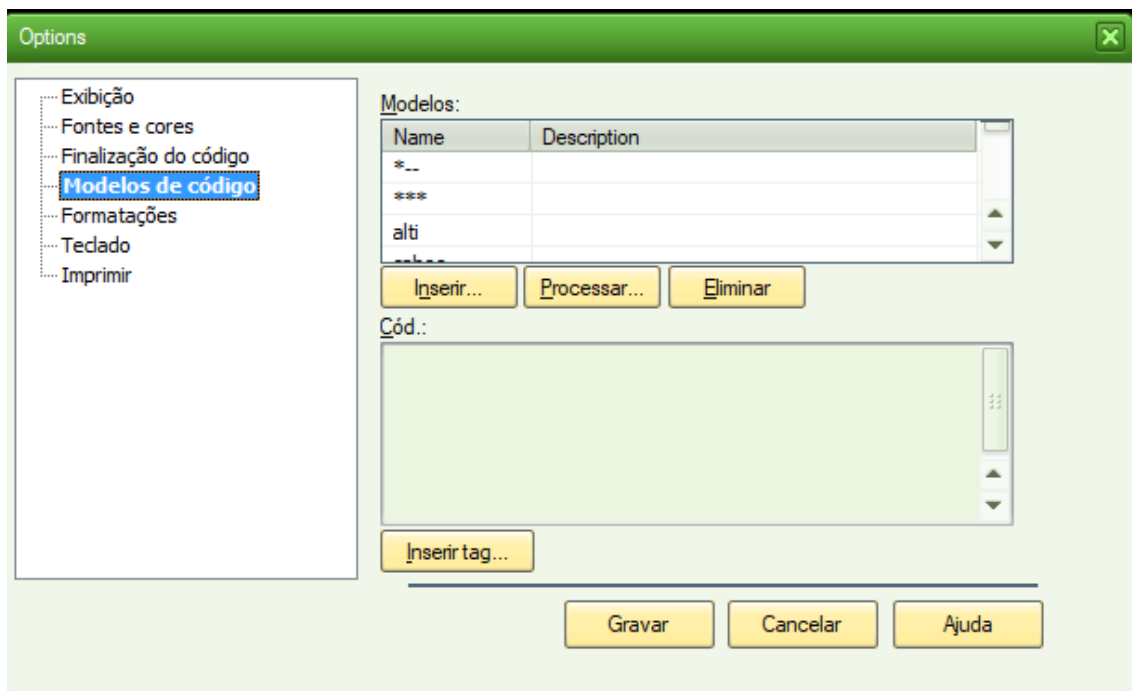
```
9      REPORT zt
10      Loop (5)
11      LOOP
```

Sempre que o símbolo entre ( ) for exibido, ao pressionar a tecla o comando será preenchido automaticamente, isso existe em diversas linguagens de programação e é muito útil para facilitar o trabalho do programador. O SAP permite que sejam criadas novas configurações pré-definidas como, por exemplo, para indicar alterações no código ABAP, vejamos abaixo como criar uma configuração pré-definida de sintaxe.

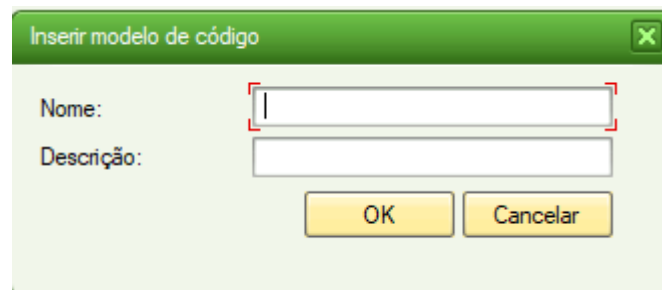
No editor de código, clique no botão que está destacado em vermelho, conforme a imagem:



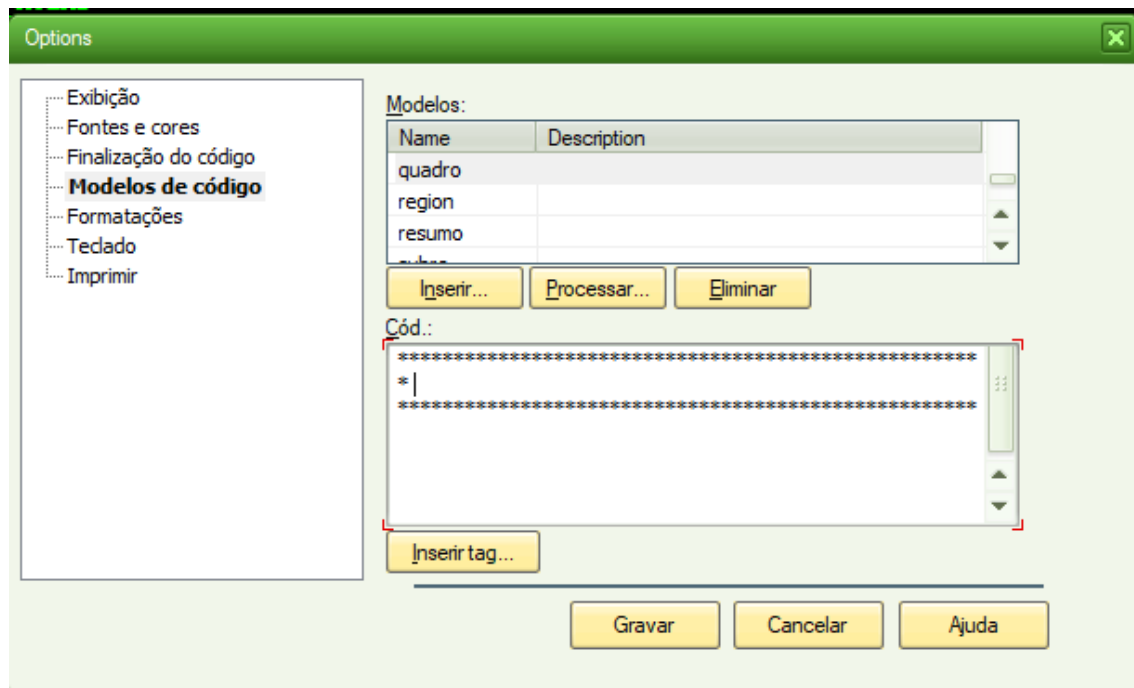
Vá até a opção “Modelo de código”:



Aqui ficam definidas todas as sintaxes que podem ser completadas automaticamente com a tecla TAB, inclusive as já definidas pela SAP, não é recomendado que sejam removidas as já existentes, para inserir uma nova, clique no botão “Inserir”.

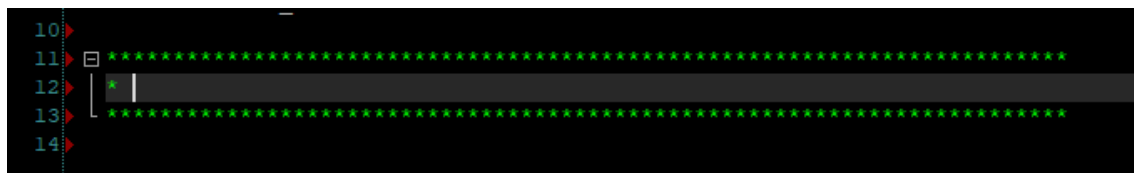


O campo nome é usado para indicar qual será o comando que será chamado para ser auto completado pela tecla TAB, utilize algo que não tenha conflitos com alguma outra sintaxe já existente, como por exemplo, “quadro”, que serve para fazer cabeçalhos e vamos fazer nesse exemplo, a descrição não necessita de nenhuma configuração específica, serve apenas para informar para que serve o comando auto completar criado.

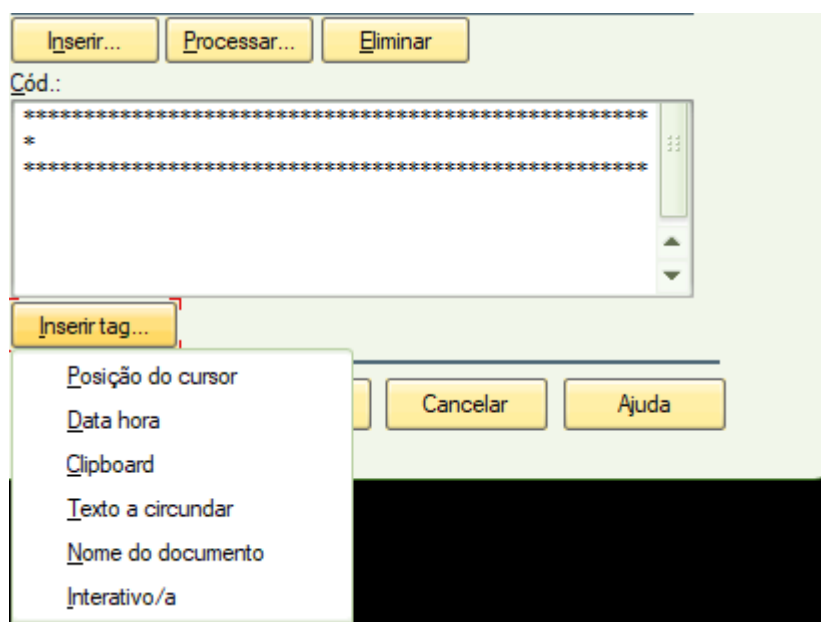


Tudo o que for inserido dentro da parte de codificação do modelo de código, será preenchido automaticamente pela chamada do “quadro” dentro do editor ABAP, neste exemplo criamos uma linha de asteriscos para formar um cabeçalho.

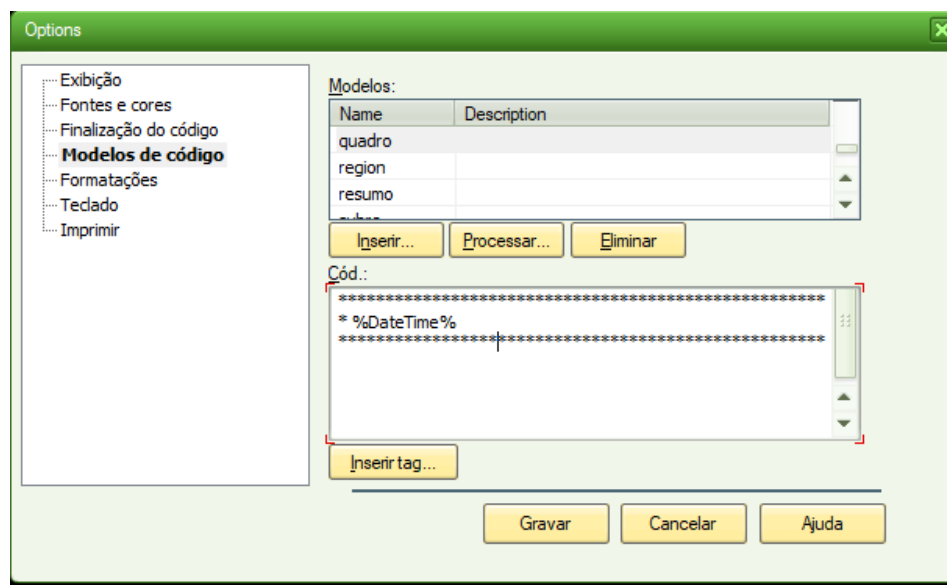
Ao chamar a palavra quadro no editor ABAP e pressionar o TAB no teclado, automaticamente será preenchido o quadro criado, já formatado e agilizando o trabalho que muitas vezes tem que ser feito repetidamente:



Também é possível inserir informações precisas, como horas, posição do cursor, dentre outros, para isso, basta durante a edição do modelo de código, clicar na opção “Inserir Tag”:



Vamos inserir, por exemplo, a data e hora dentro do quadro:

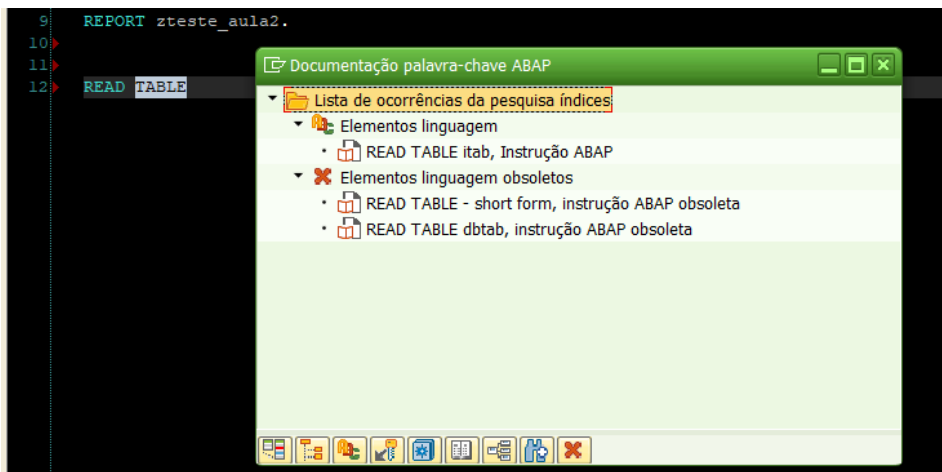


Ao ser chamado em um programa ABAP:

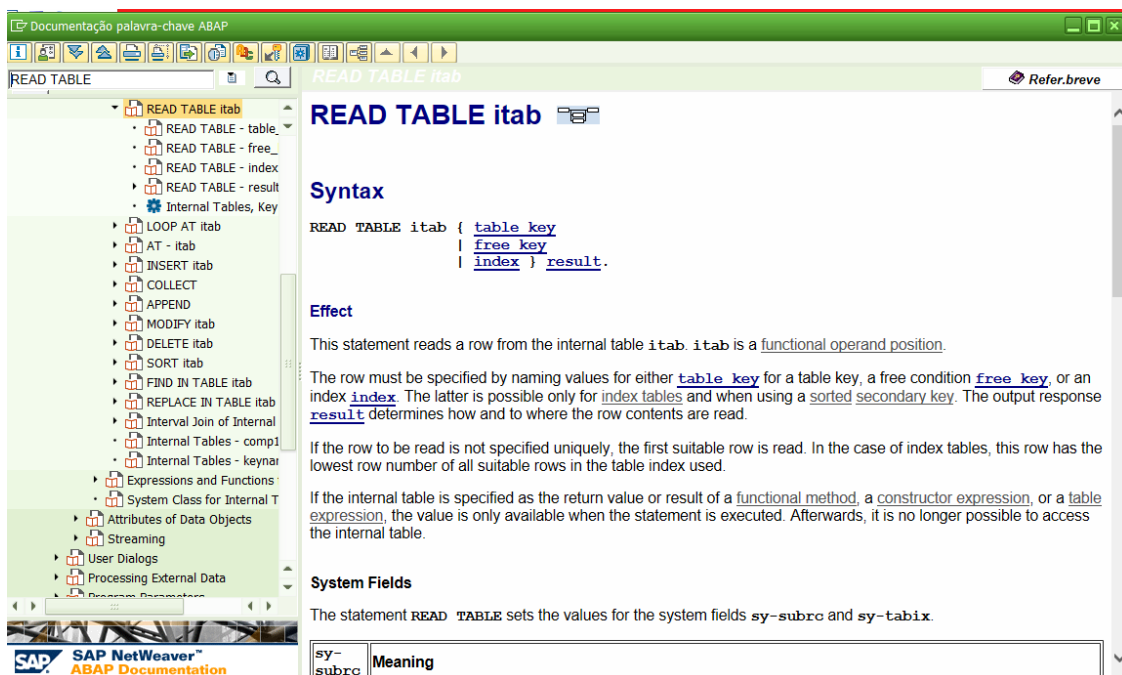


- Entendendo o Help de Sintaxe ABAP (Documentação e Transação de ajuda)

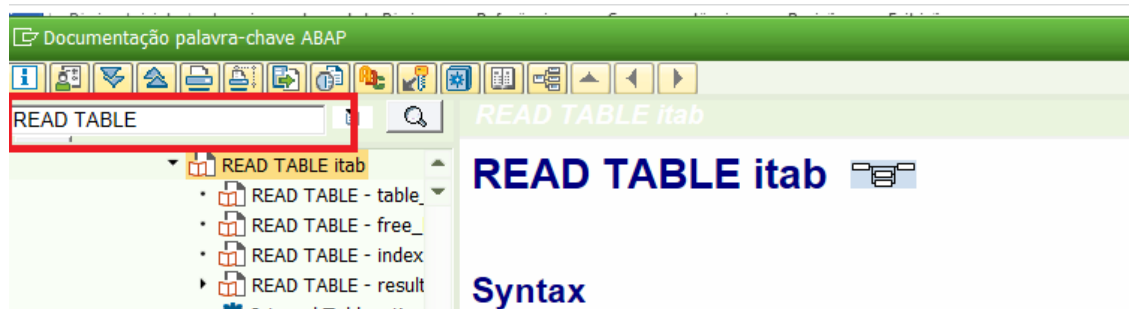
Todas as sintaxes do ABAP podem ser exploradas de forma mais completas através do help SAP (F1 no Teclado), para isso, em qualquer sintaxe dentro do editor, basta clicar sobre o comando desejado e pressionar a tecla F1 no teclado e o Help será exibido, demonstrando o comando de forma completa, seguido de um exemplo de sua utilização, conforme demonstrado abaixo:



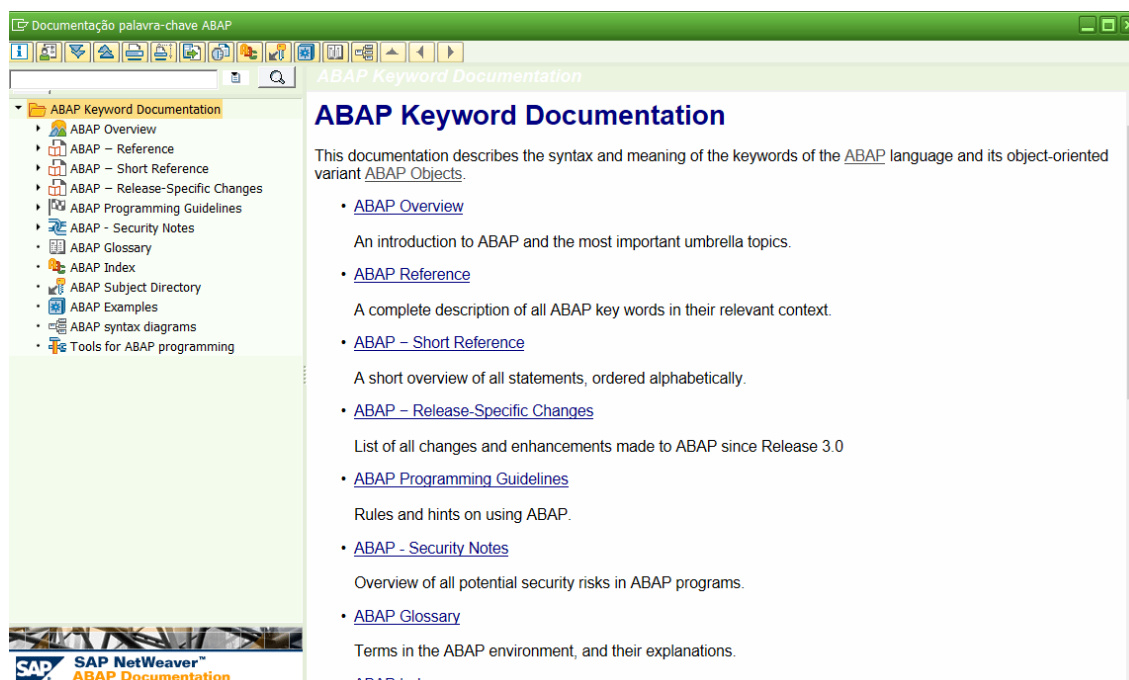
Quando exibida a ajuda, basta clicar duas vezes sobre o comando desejado e a ajuda completa será exibida:



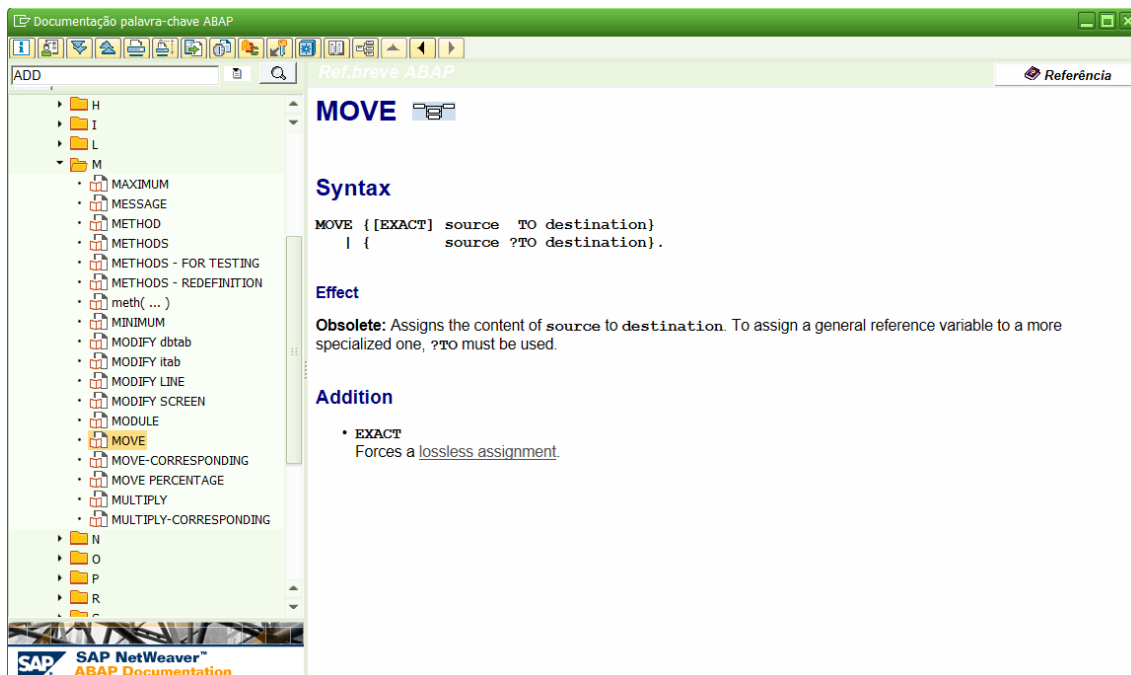
Ainda dentro do Help, outros comandos podem ser pesquisados diretamente da barra de pesquisas, localizada no canto superior esquerdo:



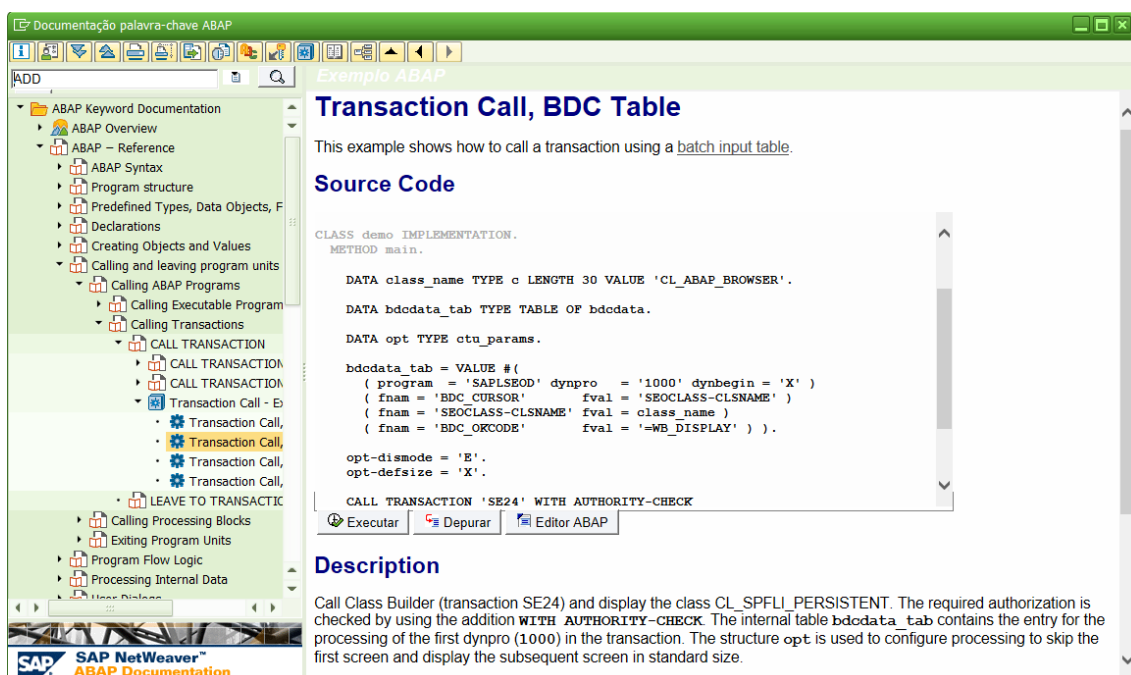
Ainda se tratando do Help SAP, temos a transação ABAPDOCU, que contém diversos exemplos de comandos e sintaxes, que podem ajudar no entendimento de um comando:



Você pode buscar o comando por ordem alfabética:



E pode até mesmo executar e testar comandos através dos SAMPLES ABAP:



## 16. Programas de carga/Interface (Upload/Download)

Muitas das interfaces de comunicação no SAP, tem como sua principal base a troca de arquivo entre sistemas, para isso, os arquivos podem ser recebidos no SAP através de um TXT, CSV, Excel, dentre outros formatos conhecidos e mais utilizados, o SAP também pode gerar arquivos nesse formato, para fornecer para sistemas terceiros (legados) as informações que forem necessárias para essa transferência de dados.

## 17. Formato TXT (Upload/Download)

Um dos formatos mais conhecidos para geração de arquivos é o TXT, por ser mais leve e de fácil comunicação entre vários sistemas, vejamos abaixo como funciona um programa de upload e download e informações no SAP:

## 18. Upload de Arquivos – File Local:

Podemos subir arquivos diretamente de um caminho local no computador, para isso, criamos um PARAMETER associado a uma ajuda de pesquisa para encontrar esse arquivo, o tipo de objeto padrão associado a esse PARAMETER é o RLGRAP-FILENAME, pois este tipo de objeto já está associado a um tamanho de 128 caracteres padrão aos caminhos de arquivo, sua declaração ficaria como na imagem abaixo:

```
PARAMETER: p_file TYPE RLGRAP-FILENAME. "Caminho do Arquivo Local
```

Ao criar o objeto acima, o campo na tela será criado com o tamanho indicado, porém, a caixa de pesquisa para o caminho no Windows não será exibida ao menos que associarmos um evento para que essa pesquisa seja adicionada, veremos mais adiante no curso como os eventos funcionam, mas para não perdemos esse ponto nessa aula, vamos criar o evento abaixo para associar uma ação ao PARAMETER P\_FILE criado, conforme na imagem abaixo:

```
AT SELECTION-SCREEN ON VALUE-REQUEST FOR p_file.
```

O evento acima indica que uma ação será adicionada ao campo P\_FILE, no caso, chamaremos uma função do SAP que atribui automaticamente a caixa de diálogo do Windows, permitindo a pesquisa pelo arquivo, para isso, basta informar após esse evento o código, conforme abaixo:

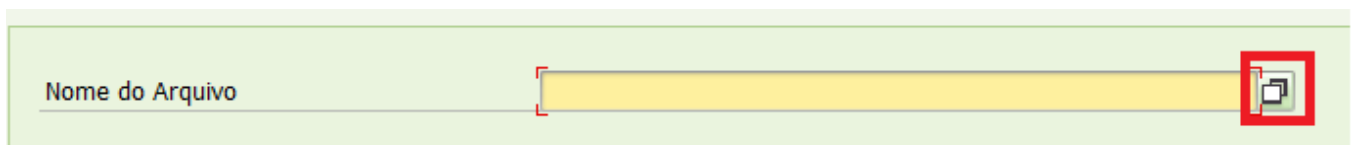


```

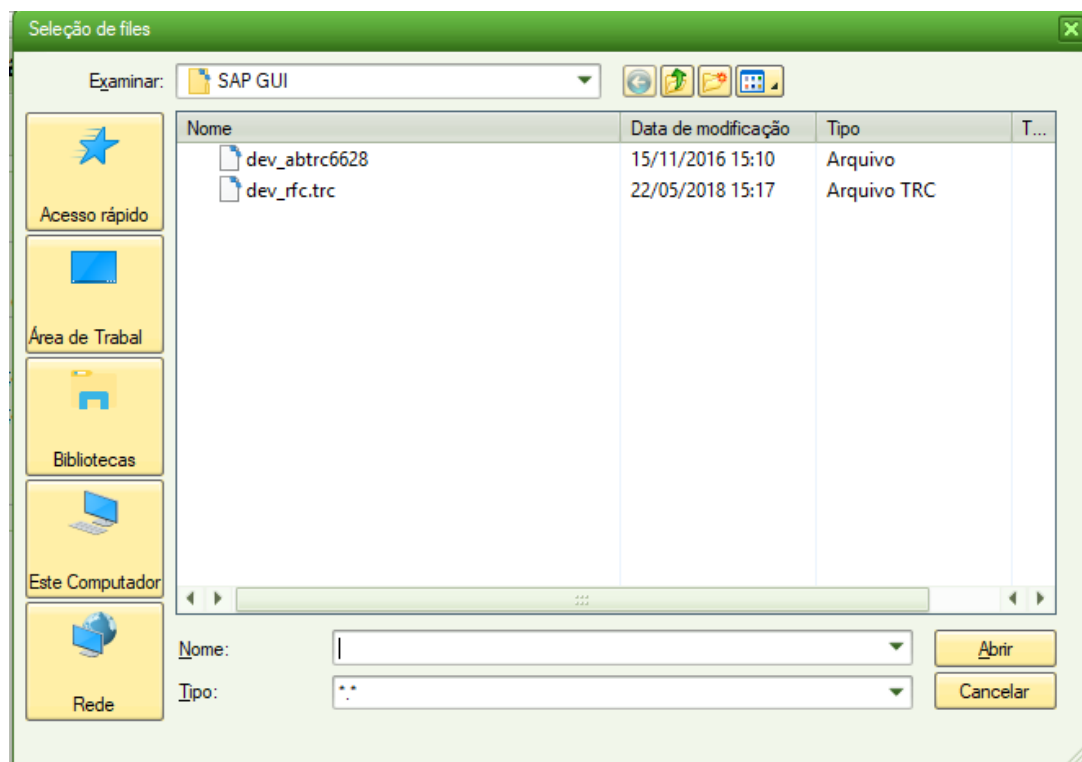
" Função que cria o Match Code para abrir caminho do arquivo no parâmetro de tela
CALL FUNCTION 'F4_FILENAME'
EXPORTING
  program_name = syst-cprog "Nome do programa principal (variável de sistema)
  dynpro_number = syst-dynnr "Número da tela do programa principal (variável de sistema)
IMPORTING
  file_name     = p_file.    "Nome do Parameter que será associada a caixa de pesquisa

```

Ao criar a função com os parâmetros corretos, o PARAMETER terá uma opção de pesquisa associado, permitindo ao usuário buscar o caminho do arquivo, conforme as imagens abaixo:



Caixa de diálogo exibida:



Para que o arquivo seja importado para o SAP, precisamos usar uma outra função do SAP, chamada GUI\_UPLOAD, essa função tem diversas utilizações, utilizaremos nesse exemplo o seu conceito mais simples, onde ela lê o nome do arquivo e o converte para uma tabela interna, para isso, antes de mais nada, vamos criar uma tabela interna que possa importar os dados do arquivo de forma completa, nesse exemplo declaramos nossa tabela interna com o tipo abaixo:

```

4 |-----*
5 | TYPES: BEGIN OF ty_arquivo,
6 |   line(500) TYPE c,
7 |   END OF ty_arquivo.
8 | *****

```

O tamanho 500 foi utilizado de modo genérico, em programas de ambientes produtivos, teremos o tamanho exato do que esses arquivos podem conter, então, sempre vamos definir o seu tipo com o tamanho máximo da linha, para que nenhum dado seja perdido durante a importação, após a criação da tabela interna, vamos chamar a função GUI\_UPLOAD, passando os parâmetros obrigatórios para a conversão do arquivo em tabela interna, conforme demonstrado abaixo:

```

DATA: lv_file TYPE string. "Variável Local criada com o tamanho do campo da função
"Convertendo campo da tela tamanho 128 em um campo tamanho string
lv_file = p_file.

"Função usada para carregar arquivos do SAP para uma tabela interna
CALL FUNCTION 'GUI_UPLOAD'
  EXPORTING
    filename = lv_file      "Variável que contém o caminho do arquivo
    filetype = 'ASC'        "Formato padrão para importação de arquivo TXT
  TABLES
    data_tab = t_arquivo.  "Tabela interna criada para receber os dados

```

Todas as funções do SAP só podem receber campos definidos exatamente com o seu tipo correto, no caso o campo P\_FILE foi definido com tamanho de 128 para a tela, e o tamanho do campo da função é de formato STRING, devido a isso, fizemos a passagem do campo para uma variável criada localmente, caso contrário a função não interpretará o campo correto causando DUMP durante a importação das informações necessárias.

## 19. Download de Arquivos – File Local:

Seguindo o mesmo conceito do Upload, precisamos criar um PARAMETER com o tipo RLGRAP-FILENAME para que o usuário possa informar o caminho e o arquivo a ser criado, também é possível associar um evento para encontrar um arquivo, mas a função é um pouco diferente da que usamos para encontrar um arquivo local, já que na criação do arquivo o ideal é que o usuário sugira o local já de modo Default e o arquivo seja gerado via programa, ou seja, dentro da lógica o nome do arquivo seja gerado através de algumas concatenações de data, siglas etc, por isso, no exemplo abaixo, criamos apenas um PARAMETER simples associado ao tipo mencionado:

```

PARAMETER: p_file_d TYPE rlgrap-filename. "Caminho do Arquivo Local Baixado.

```

Após informado na tela o caminho e o nome do arquivo, vamos chamar a função GUI\_DOWNLOAD, que se parece muito com a função GUI\_UPLOAD, conforme demonstrada abaixo:

```
"Variável Local criada com o tamanho do campo da função
DATA lv_filename TYPE string.

"Convertendo campo da tela tamanho 128 em um campo tamanho string
lv_filename = p_file_d.

"Função usada para fazer download de informações do SAP para o arquivo Local
CALL FUNCTION 'GUI_DOWNLOAD'
  EXPORTING
    filename = lv_filename "Variável que contém o caminho do arquivo que será criado
    filetype = 'ASC'       "Formato padrão para criação do arquivo TXT
  TABLES
    data_tab = t_arquivo.  "Tabela interna que contém os dados a serem baixados
```

## 20. Formato Excel (Upload/Download)

Além do formato TXT, também muitas vezes usamos o formato Excel para carregar os dados para o SAP, para isso, seguimos com a mesma lógica para a criação do PARAMETER de importação.

## 21. Upload de Arquivos Excel – File Local:

Para importar um arquivo Excel para o SAP, criamos um PARAMETER da mesma forma que criamos para os formatos TXT, sem nenhuma alteração a única diferença para esse tipo de importação está na declaração da tabela interna que receberá as informações e na função que é chamada para fazer a conversão, vejamos abaixo em um exemplo como isso é realizado.

Foi criada uma tabela interna com 3 campos, esses 3 campos representam as 3 colunas em um Excel, ou seja, caso o Excel tenha 15 colunas, vamos precisar criar 15 campos dentro do nosso TYPES, usamos um exemplo de declaração de tabelas internas mais antigo, embora obsoleto ele fica como um exemplo diferente de como uma tabela interna pode ser criada, segue abaixo:

```
6  
7 DATA: BEGIN OF t_saida_excel OCCURS 0, "Uma outra forma de fazer tabela interna  
8     material    TYPE bapimathead-material,  
9     maktx       TYPE maktx,  
10    lote(4)     TYPE c,  
11 END OF t_saida_excel.  
12
```

Após a criação da tabela interna e o número de campos equivalente as colunas do Excel, usaremos a função TEXT\_CONVERT\_XLS\_TO\_SAP para converter os dados do Excel para uma tabela interna do SAP, vejamos abaixo como o preenchimento dessa função deve ocorrer:

```
CALL FUNCTION 'TEXT_CONVERT_XLS_TO_SAP'  
EXPORTING  
  i_line_header      = 'X'           "Indica se irá ignorar a primeira linha ou não (X = Pula ' ' = Mantém)  
  i_filename         = p_file        "Caminho do Excel (No caso esse contém também 128 como a tela)  
TABLES  
  i_tab_converted_data = t_saida_excel. "Tabela Interna Criada para Receber as 3 colunas do Excel
```

Existe uma diferença em relação a função GUI\_UPLOAD que recebe o caminho do arquivo apenas por um campo do tipo STRING, a função TEXT\_CONVERT\_XLS\_TO\_SAP pode receber diretamente do campo da tela de seleção, desde que o mesmo tenha o

formato de 128 caracteres, pois a função contém o mesmo parâmetro, não causando problemas. O parâmetro `i_line_header` indica se será importada a primeira linha ou apenas a partir da segunda, isso para o caso de arquivos Excel que possam conter linhas de cabeçalho.

- **Download de Arquivos Excel – File Local:**

Para gerar um arquivo em CSV, o formato padrão em arquivos Excel usados em comunicação entre sistemas, usamos a mesma função `GUI_DOWNLOAD`, porém, com a regra que ao invés de informar um arquivo com extensão TXT usamos o formato CSV, e a função fará a conversão automaticamente, no exemplo abaixo, usamos os mesmos dados que foram importados do Excel no exemplo acima para gerar um download em CSV das mesmas informações, segue abaixo o exemplo:

```
lv_filename = 'C:\temp\arquivo_01.CSV'.  
  
CALL FUNCTION 'GUI_DOWNLOAD'  
EXPORTING  
  filename = lv_filename  
  filetype = 'ASC'  
TABLES  
  data_tab = t_saida_excel  
EXCEPTIONS  
  OTHERS   = 1.
```

## 22. Diferentes tipos de ALVs e seus conceitos

Os ALVs são os relatórios gerados para exibir as informações diversas do SAP, muito requisitados em todos os clientes, pois podem fornecer quaisquer tipo de informações programadas além de carregar automaticamente uma série de ferramentas para manuseio desse relatório, abaixo veremos como montar um ALV simples passo a passo e em seguida alguns conceitos de diferentes tipos de ALVs que o SAP pode conter.

No exemplo abaixo, temos um ALV que não possui tela de seleção, quando executado, o programa faz uma busca dos dados de uma tabela e apresenta diretamente o relatório dos dados encontrados nesta tabela.

A função do ALV, como as demais funções do SAP, precisa de algumas configurações pré-definidas para que possa ser executada, embora a função `REUSE_ALV_GRID_DISPLAY` tenha muitas ferramentas que possam ser utilizadas a que

veremos logo abaixo são as essenciais para que o relatório seja exibido de forma simples:

## 23. Tabela de configuração de Fieldcat:

O Fieldcat pode ser comparado o TYPES de uma tabela interna, o ALV precisa saber quais campos e qual o tipo dos campos que serão exibidos em seu relatório, bem como sua descrição, tamanho, etc. Essa estrutura é obrigatória e sem ela não é possível exibir o relatório, para definir essa estrutura, sempre devemos declarar uma tabela interna com o tipo SLIS\_T\_FIELDCAT\_ALV, que pertence a um grupo de tipos do ALV. O grupo de tipos é um conjunto de TYPES já definidos dentro do SAP, ele pode ser acessado pela sintaxe TYPE-POOLS, dentro desse grupo de tipos definimos diversos TYPES que podem ser reutilizados em um programa ABAP, é possível criar um grupo de tipos Z, veremos isso mais adiante no curso, mas é essencial lembrar que para todos os ALVs, vamos precisar definir logo no início do programa a chamada da sintaxe conforme abaixo:

```
*-----*  
*Definições para ALV  
TYPE-POOLS: slis.
```

O grupo de tipo SLIS contém diversos TYPES do ALV, inclusive o SLIS\_T\_FIELDCAT\_ALV que vamos usar para referenciar nossa tabela interna do fieldcat, para isso, vamos criar uma tabela interna, conforme a imagem abaixo:

```
14 TYPE-POOLS: slis.  
15 *-----*  
16 * Tabelas Internas Exclusivas do ALV  
17 *-----*  
18 DATA: t_fieldcat TYPE slis_t_fieldcat_alv. "Definição da Estrutura do ALV  
19
```

Ainda se tratando da configuração do ALV, podemos definir algumas opções para a exibição do relatório, como colunas otimizadas automaticamente, campos de valores já somados de modo default, o zebra, que faz as linhas ficarem tracejadas para melhor visualização, e para isso usamos uma outra referencia associada ao grupo de tipos SLIS que é SLIS\_LAYOUT\_ALV, nesse caso esse objeto não precisa e nem deve ser criado como tabela, pois possui apenas uma linha, sendo assim, criamos apenas como uma work área, conforme a imagem abaixo:

```

24 *-----*
25 * Work Áreas
26 *-----*
27 DATA: w_layout TYPE slis_layout_alv, "Estrutura de Configuração do ALV
28

```

Em nosso programa de exemplo, criamos 3 rotinas de modularização para agrupar os passos do programa, uma para a seleção dos dados que serão exibidos, a outra para montar o fieldcat e a última para exibir o ALV, conforme a imagem abaixo:

```

4 *-----*
5 * Início da lógica do programa
6 *-----*
7 START-OF-SELECTION.
8
9     PERFORM f_seleciona_dados.
10    PERFORM f_monta_fieldcat.
11    PERFORM f_exibe_alv.
12

```

## 24. Seleção dos dados:

Foi utilizada uma seleção simples em uma tabela Z já existente, para pegar todo o seu conteúdo e jogar em uma tabela interna que foi criada com o nome de t\_itab, ela faz referência a tabela ZSOFTWARE.

```

SELECT *
  FROM zsoftware
 INTO TABLE t_itab
UP TO 100 ROWS.

```

## 25. Montagem do Fieldcat:

A tabela do fieldcat contém inúmeras configurações que podem ser levadas ao ALV, usamos apenas 3 dos diversos campos que ela pode receber, esses 3 campos informam para o ALV qual é o campo, de qual tabela interna e qual sua descrição a ser exibida no ALV:

	Campo	Tabela Interna	Descrição
* Configuração do ALV			
PERFORM monta_fieldcat USING:	'CODIGO'	'T_ITAB'	'Código',
	'FABRICANTE'	'T_ITAB'	'Fabricante',
	'DESCRICAO'	'T_ITAB'	'Descrição',
	'METRICA'	'T_ITAB'	'Métrica',
	'TIPO'	'T_ITAB'	'Tipo'.

Cada um dos campos exibidos em literais acima, são convertidos em variáveis criadas dentro do form, na mesma ordem em que foram criadas no comando USING, ou seja, P\_CODIGO recebe o valor do nome do campo declarado na chamada do PERFORM, P\_TAB recebe o nome da tabela interna e P\_DESC recebe o nome da descrição do campo que foi informado anteriormente.

Os dados são passados para a work área da estrutura do fieldcat em seguida são inseridos na tabela interna do fieldcat, que neste exemplo terá 3 linhas, contendo a configuração dos campos que serão exibidos no ALV:

```
FORM monta_fieldcat USING p_codigo p_tab p_desc.  
  
  w_fieldcat-fieldname      = p_codigo. "Nome do campo  
  w_fieldcat-tabname       = p_tab.    "Tabela Interna que contém esse campo  
  w_fieldcat-reptext_ddic  = p_desc.   "Descrição do Campo  
  
  APPEND w_fieldcat TO t_fieldcat.  
  CLEAR w_fieldcat.
```

- **Exibição do ALV:**

Antes de chamar a função do ALV, definimos algumas configurações para sua exibição, essa estrutura abaixo contém diversas configurações que podem ser personalizadas, usamos apenas o ZEBRA mencionado acima e a otimização das colunas, para que sejam redimensionadas de acordo com o tamanho do texto e do campo:

```
*-----*  
* Configuração Visual do ALV  
*-----*  
w_layout-zebra          = 'X'. "Insere linhas com traçados diferentes  
w_layout-colwidth_optimize = 'X'. "Redimensiona o campo para exibição completa  
*-----*
```

Após feita a configuração de exibição, passamos os parâmetros para a função do alv REUSE\_ALV\_GRID\_DISPLAY, informando os parâmetros básicos essenciais para que o relatório seja exibido.

O parâmetro i\_callback\_program deve sempre receber a variável de sistema com o nome do programa, assim quando o usuário clicar em voltar, o SAP irá retornar para a tela de seleção do programa que chamou a função.

Os parâmetros i\_default e i\_save normalmente são preenchidos como estão abaixo, salvo raras exceções para alvs específicos.




```

102 *-----*
103 * Exibe o ALV
104 *-----*
105 CALL FUNCTION 'REUSE_ALV_GRID_DISPLAY'
106   EXPORTING
107     i_callback_program = sy-repid      "Nome do programa que chamou o ALV (variável de sistema)
108     is_layout          = w_layout      "Configuração Visual do ALV
109     it_fieldcat        = t_fieldcat[]  "Tabela de configuração do ALV
110     i_default          = 'X'           "Default X
111     i_save             = 'A'           "Default A
112   TABLES
113     t_outtab           = t_itab.       "Tabela com os dados de saída da seleção
114

```

## Resultado do ALV:

Relatório ALV de Exemplo					
					
Código	Fabricante	Descrição	Métrica	Tipo	
00000001	ORACLE	ORACLE 12	CORE	BANCO DE DADOS	
00000002	IBM	WEBSHERE APPLICATION SERVER	PVU	WEB	
00000003	IBM	MESSAGE BROKER	PVU	WEB	
00000004	MICROSOFT	OFFICE 365	INSTALL	OFFICE	
00000005	ORACLE	EXADATA	CORE	APPLIANCE	
00000006	MICROSOFT	OFFICE 365	INSTALL	OFFICE	
00000007	MICROSOFT	WINDOWS 2012	INSTALL	SIST OPERAC	
00000008	MICROSOFT	WINDOWS 7	INSTALL	SIST OPERAC	
00000009	MICROSOFT	WINDOWS 8	INSTALL	SIST OPERAC	
00000010	MICROSOFT	WINDOWS 10	INSTALL	SIST OPERAC	

As ferramentas exibidas acima do relatório já são configuradas automaticamente pela função, é possível editar essas ferramentas, seja para adicionar ou remover botões, porém as que já vem configuradas não precisam de nenhuma programação ABAP para funcionar, pois a função já tem sua programação realizada.

- **ALV Tree**

O SAP possui diversos tipos de ALVs mais complexos, a imagem abaixo exibe um ALV com pastas para seus itens, com um conceito de cabeçalho, item e sub itens de registros, embora sua codificação seja um pouco mais ampla, a SAP disponibiliza exemplos desses ALVs, são os chamados BCALV\*, esse abaixo se chama: BCALV\_TREE\_DND\_MULTIPLE

## ALV Tree Control: arrastar e soltar dentro árvore hierárq.

Arrastar um nó para o voo na pasta de favoritos




Total/Mês/Companhia aérea/Data	...	Preço	M...	TpAvião	Capac	Ocup...	Total	Capac	Ocup...	Capac	.
• Favoritos					0	0		0	0	0	
▼ Vões					0	0		0	0	0	
▼ 2009->Fevereiro	17	422,94 USD	747-400		385	370	192.057,12	31	31	21	
▼ AA	17	422,94 USD	747-400		385	370	192.057,12	31	31	21	
• 11.02.2009	17	422,94 USD	747-400		385	370	192.057,12	31	31	21	
• 13.02.2009	64	422,94 USD	A310-300		280	265	131.166,47	22	21	10	
▶ AZ	...0	1.014,00 EUR	747-400		385	367	459.200,04	31	31	21	
▶ JL	...7	106.136 JPY	DC-10-10		380	361	48.109.288	41	38	18	
▶ LH	...1	666,00 EUR	A319		220	214	173.632,86	22	21	10	
▶ QF	5	788,64 AUD	A310-300		280	265	246.513,26	22	22	10	
▶ 2009->Março	17	422,94 USD	747-400		385	365	189.938,20	31	29	21	
▶ 2009->Abril	17	422,94 USD	747-400		385	374	195.009,38	31	29	21	

## 26. ALV Múltiplo na mesma tela

Também é possível exibir mais de um ALV na mesma tela, como o BCALV\_TEST\_BLOCK\_LIST, esse conceito é chamado de ALV SPLIT, não é muito utilizado, mas é importante que seja conhecido, pois em alguma etapa de algum processo pode ser necessária a criação de algum ALV desse tipo, mesmo que os exemplos do SAP não ajudem, na internet ao usar o termo ALV SPLIT várias referências serão demonstradas sobre esse ALV.

My WinTitlebar



ID	Nº	Data voo	LIST	Companhia aérea	Companhia aérea	Preço voo	Preço voo-2	Moeda	Moeda	Tipo avião	Capacid.	Ocupados
Lista não contém dados												

ID	Nº	Data voo	LIST	Companhia aérea	Companhia aérea	Preço voo	Preço voo-2	Moeda	Moeda	Tipo avião	Capacid.	Ocupados
Lista não contém dados												

ID	Nº	Data voo	LIST	Companhia aérea	Companhia aérea	Preço voo	Preço voo-2	Moeda	Moeda	Tipo avião	Capacid.	Ocupados
Lista não contém dados												

- Instruções para códigos mais performáticos

## 27. Aula de Performance Avançada

### Objetivo

- Troca de informações sobre os pontos mais comuns no mercado de programação ABAP que tem causado problemas graves de performance (clientes e parceiros)
  - Recomendações SAP
  - Demonstrar alguns casos verídicos
  - Demonstrar estatísticas sobre o uso de algumas técnicas de conversão (Batch-Input, Call Transaction, Direct Input)
- **Arquitetura Client-Server**

O R/3 trabalha com uma filosofia de cliente/servidor de 3 níveis:

Database server : responsável pelo acesso e pela atualização dos dados

Application server : responsável pelo processamento da aplicação (tempo despendido com a interpretação de comandos ABAP)

Frontend : responsável pelo processamento dos gráficos (tempo despendido pelo sistema R/3, ou seja, o middleware e Kernel). Kernel = conjunto de programas utilitários que existe dentro do R/3, servindo ao cumprimento de inúmeras tarefas

A alocação dos servidores de aplicação é definida automaticamente no momento de logon (é checado qual servidor está menos sobrecarregado).

Dica: Para se obter uma otimização da performance em programas ABAP, deve-se minimizar o tempo de acesso ao database.

## 28. Grandes vilões no que se refere à performance

- Ninhos de select
- Select .... Endselect ao invés de comandos que recuperem do banco de dados todos os registros de uma única vez
- Select \* ao invés de select com as colunas necessárias ao processamento
- Select single sem chave completa especificada ao invés de Select up to 1 row
- Selects genéricos, ou seja, onde a cláusula where não foi fortemente especificada, com várias condições, visando restringir a seleção
- Índices não utilizados
- Falta de índices
- Select em tabelas com alto número de registros utilizando cláusula where baseada em tela de seleção onde o preenchimento não é obrigatório
- Definição funcional falha
- Grandes tabelas do sistema: BKPF, BSEG, EKKO, EKPO, VBAK, VBAB, MKPF, MSEG, J\_1BNFDOC, J\_1BNFLIN
- Funções genéricas, como por exemplo CLAF\_CLASSIFICATION\_OF\_OBJECTS

## 29. Comunicação entre o Application Server e Database Server - Transmissão de pacotes

### Premissa:

VBAK contém 1000 registros, sendo que cada registro contém 575 bytes.

### Fato:

A comunicação entre o DB server e o Application server acontece em pacotes de 32.000 bytes, dependendo da rede e equipamentos de comunicação utilizados.

➔ Select \* from VBAK.

### Portanto:

1000 registros x 575 bytes = 575.000 bytes

575.000 bytes / 32.000 = 20 pacotes

20 pacotes: na verdade este número é um pouco mais alto (mais ou menos 24), uma vez que informações administrativas são transferidas em cada transmissão, juntamente com os dados.

O exemplo acima tenta ilustrar que, se forem selecionadas apenas as colunas necessárias, a transferência de dados será menor e, conseqüentemente, haverá uma redução significativa no tempo de resposta.

- **Select e Logical Database**

➔ Get VBAK

➔ Get VBAK fields vbeln auart bname kunnr

O segundo comando é muito melhor que o primeiro. Porém, nem todos os logical database suportam select em colunas. Para verificar se isso é possível, acesse a transação SE36, coloque o nome do database no qual se deseja efetuar uma pesquisa e escolha a opção: extras -> field selection.

## 30. Cursor Caching

Os seguintes comandos SQL produzem o mesmo resultado, mas cada um deles requer seu próprio cursor uma vez que os comandos não são idênticos (verifique a ordem das colunas após a palavra Select e na cláusula WHERE):

```
Select vbeln auart into (vbak-vbeln, vbak-auart) from vbak  
where vbeln = nnn and auart = yyyy
```

```
Select vbeln auart into (vbak-vbeln, vbak-auart) from vbak  
where auart = yyyy and vbeln = nnn
```

```
Select auart vbeln into (vbak-auart, vbak- vbeln) from vbak  
where vbeln = nnn and auart = yyyy
```

```
Select  auart vbeln  into (vbak-auart, vbak- vbeln) from vbak  
where auart = yyyy and vbeln = nnn
```

Desnecessários comandos Declare e Prepare podem ser evitados se os comandos SQL são mantidos consistentes nos programas que estão sendo desenvolvidos (quando se usar select com colunas ou declarando condições WHERE sempre use a sequência de campos conforme definido no Data Dictionary)

## 31. Logical database e Tables statements

Report ZAZ\_PERFORMANCE.

Tables: vbak, vbkd, vbpa.

Get vbak fields vbeln auart bname kunnr.

Write: vbak-vbeln, vbak-auart, .... .

Get vbpa fields parvw kunnr.

Write: vbpa-parvw, vbpa-kunnr, vbkd-zterm.

Ao invés use:

Get vbak fields vbeln auart bname kunnr.

Write: vbak-vbeln, vbak-auart, .... .

Get vbkd field zterm.

Get vbpa fields parvw kunnr.

Write: vbpa-parvw, vbpa-kunnr, vbkd-zterm.

OBS.: A estrutura do logical database VAV (utilizado anteriormente) é VBAK – VBUK – VBKD – VBPA.

Se uma tabela não é especificada via um cartão TABLES então o sistema automaticamente recupera somente os campos chave para aquela tabela (aqui VBUK não está presente no cartão tables, assim somente os campos chaves são recuperados pelo sistema).

Se uma tabela é especificada via TABLES, todas as colunas são recuperadas, mesmo se nenhum campo daquela tabela é usado no programa (veja a tabela VBKD – somente o campo ZTERM deveria ser impresso).

Para evitar ter todas as colunas selecionadas da tabela, insira um comando GET para aquela tabela, especificando os campos que serão utilizados pelo programa posteriormente.



- **Select ... exit versus Select .... Up to 1 rows.**

#### **Vantagem em performance:**

Select ... exit. Endselect. = 150.000 ms  
 Select ... up to 1 rows = 1.500 ms

- **Check inside Select ...endselect.**

Comparação de Runtime é baseada numa varredura de 57.000 registros na VBAK

Select & Check = 27.958.000 ms  
 Select com cláusula WHERE = 3.065.000 ms

Em caso de condições de checagens muito complexas poderia ser mais rápido obter dados do banco de dados primeiramente. SQL Trace precisa ser usado para verificar qual método é o mais rápido.

Isso também é válido para select single.

- **Selects em campos sem índices:**

Select-options: bname\_so for vbak-bname.

Select vbeln bname .... Into (vbak-vbeln ...) from vbak  
 Where bname in bname\_so order by <field> descending.

Endselect.

O sistema standard é distribuído sem um índice para o campo bname. A seleção acima irá resultar numa varredura sequencial da tabela vbak. Isto não é aceitável num sistema em produção. Criando um índice na tabela vbak para o campo bname irá resolver o problema neste caso. Exceção: tabelas cuja buferização é full ou generic (neste caso não tem sentido pois o primeiro acesso irá trazer todos os dados para a memória. Das próximas vezes, o sistema busca do buffer e não do database – até que o dado mude novamente).

## Select sum, avg, min, max versus ABAP calculations

```
Select matnr kwmeng meins into <internal table>
  From vbap where ....
  Collect <internal table>.
Endselect.
```

- **Ao invés use:**

```
Select matnr sum( kwmeng ) meins into table <internal table>
  From vbap where ....
  Group by matnr meins.
```

O ganho em performance para o processamento de 10.000 registros:

Select + collect = 2.370.000 ms

Select sum .... Into table = 1.574.000 ms

OBS.: o group by não pode ser usado para tabelas pool e cluster.

- **Select sem cláusula where**

```
Select col1 col2 .... Into .... from vbak.
  Perform calculate_stuff.
Endselect.
```

Um select sem cláusula WHERE indica um erro no design do programa, especialmente se o select é usado em tabelas SAP com grande crescimento/tamanho em pouco tempo, em geral, BKPF, BSEG, COBK, COEP, LIPK, LIPS, MKPF, MSEG, VBAK, VBAP, VBPA, VBFA.

Programas irão rodar de forma correta e bem na entrada em produção, todavia a performance irá decrescer à medida que dados são adicionados diariamente.

- **Table buffering**

Para analisar a buferização de tabelas use função de chamadas estatísticas: Transaction ST02 -> details analysis menu -> call statistics (tables). Esta função ajuda você a determinar se tabelas desenvolvidas pelo cliente são buferizadas corretamente.

Para analisar a qualidade da buferização de tabelas use a transação ST02, dê duplo click em TABLES (generic key) e clique no pushbutton “buffered objects”.  
Por que buferizar ?

Usando buferização pode-se reduzir consideravelmente o tempo que se leva para recuperar um registro.

- **Tipos de buferização:**

- a) full: resident buffer (100 %): o conteúdo de toda a tabela é carregado no buffer no primeiro acesso realizado na tabela
- b) generic: uma chave genérica (primeiros “n” campo chaves) é especificado quando da criação da tabela, quando se mantém technical settings. Esta chave genérica divide o conteúdo da tabela em chamadas áreas genéricas. Quando acessando qualquer dado com uma determinada/especificada chave genérica, toda a área genérica é carregada no buffer.
- c) Partial (single record): somente registros únicos são lidos do banco de dados e armazenados no buffer.

- **Quando bufferizar uma tabela ?**

Uma tabela deveria ser buferizada quando:

- Menor em tamanho (poucas linhas ou poucas colunas com tamanho de campos pequenos)
- acessada muito mais para leitura
- alterações não ocorrem frequentemente

- **Tabelas que são boas candidatas a buferização:**

- tabelas de controle/tabelas de customização
- “small” master data tables (ex.: 100 material master records -> few changes)

- **Como buferizar uma tabela ?**

- a) Como ativar/desativar uma buferização:
  - dictionary maintenance (SE11 -> technical settings)
  - technical settings for a table (transaction SE13)
- a) Decidindo sobre o tipo de buferização:
  - full (resident = 100 %)
  - generic (with number of key fields for the generic key)
  - single records (partial buffering)

- **Comandos SQL “bypassing” a buferização:**

- select ... bypassing buffer
- select from database views (projection views are ok)
- select ... distinct
- select ... count, sum, avg, min, max
- select ... order by (outros além da chave primária)
- select ... for update
- Cláusula where contém o comando IS NULL
- Comandos nativos do SQL (EXEC SQL ... ENDEXEC) – Evite esses comandos quando estiver trabalhando com tabelas buferizadas!!!

- **Outros responsáveis pela degradação da performance**

- Move corresponding é interessante para pequenas tabelas ou quando alguns campos (mas não todos) precisam ser movimentados.
- Não use sort - appends. Ele consome muito tempo de CPU e os resultados são frequentemente imprevisíveis.
- Sempre especifique ASCENDING ou DESCENDING em processamentos com SORT. Qualificar todos os comandos SORT através da cláusula BY <data fields> melhora a performance e a leitura do código, evitando assim que todos os campos do registro sofram classificação
- 
- Use COMPUTE ao invés de ADD, SUB, MULTI, DIV desde que performance não seja um grande problema. O comando COMPUTE é mais fácil de implementar e com ele a leitura do programa se torna mais fácil. A diferença em performance é minuto, todavia se o objetivo é conseguir tempo de resposta de subsegundos para transações complexas, então deve-se usar comandos separados
- Quando da definição de uma tabela no dicionário de dados do R/3 os campos chave sempre devem ser colocados nas primeiras posições do registro da tabela (primeiras colunas). O banco de dados comprime os dados para todos os campos da tabela, mas a compressão não ocorre para campos chave. A compressão também só é possível para as colunas após o último campo chave: isso explica o porque de agrupar todos os campos chaves nas primeiras posições
- Transparent e Pool Tables: sempre que for feito um select para buscar dados de tabelas deste tipo pode-se qualificar a cláusula WHERE com campos chave e não

chave, ou seja, quanto mais condições puderem ser definidas, melhor para a obtenção de performance. Isto permite que o banco de dados avalie os registros e retorne somente aqueles que combinem com o critério de seleção definido

- Cluster table: ocorre exatamente o inverso: devido a forma de armazenamento destas tabelas, na cláusula WHERE deve constar apenas os campos chaves e os demais campos devem ser checados através do comando CHECK. O banco de dados não consegue processar cluster tables como ele processa transparent tables. Forçando o banco de dados a descompactar e checar campos (em caso de Select com campos não chaves na cláusula WHERE) é menos eficiente, na maioria dos casos, que qualificando somente com campos chaves e deixando o CHECK para os campos não chave após eles terem sido retornados. Na verdade ambos modos irão funcionar, porém um deles será ineficiente pois irá requerer grande parte da memória disponível, espaço em buffer e tempo do banco de dados para descompactar campos não chave, o que certamente irá refletir em performance. Em algumas tabelas Cluster, mesmo a seleção com restrições através das chaves primárias torna-se lenta, compensando, nestes casos, a seleção da tabela inteira e jogando-a para uma tabela interna e a partir desta fazer as restrições.
- Campos que serão comparados na cláusula WHERE devem ter atributos similares. Se isso não ocorre, o sistema tem que converter o dado toda vez que a comparação é feita. Quando isso não é possível, mova o dado a ser comparado para uma área

auxiliar, antes de efetuar a comparação (esta área auxiliar deve ser definida com o mesmo tipo e tamanho da coluna da tabela)

- Evitar classificações em várias tabelas e, posteriormente, a realização de loops. Ou seja, evitar construções do tipo SORT tab1, SORT tab2, ..., e depois o LOOP tab1, LOOP tab2 .... . Isso funciona, porém irá requerer mais memória e recursos para possuir todos os dados resultantes da classificação até que eles sejam usados. Ao contrário, quando se faz SORT seguido de LOOP, o espaço é liberado ao término do loop

- **Quando usar internal table ou Dictionary structure**

Para tabelas que são usadas repetidamente ou por vários programas, considere a possibilidade de criar estruturas de dados no Data Dictionary (e abrí-las através do comando TABLES) ao invés de definí-las dentro de programas individuais (através do comando DATA .... ENDDATA).

Sobre estas circunstâncias, o uso de tabela interna através do Data Dictionary é preferível:

- a tabela interna é grande
- a tabela é acessada da mesma forma toda vez
- a mesma tabela interna é usada em vários programas
- o programa irá processar várias tabelas internas ao mesmo tempo (e então precisa de muita memória)
- o uso de memória é um ponto chave para otimização de um programa em particular

Use o comando FREE para liberar memória alocada a tabelas internas. Este comando deve seguir o último comando para processamento do dado na tabela.

- **Use o comando FREE sob as seguintes condições:**

- a tabela interna é grande
- a tabela interna é classificada e reprocessada várias vezes
- o programa está processando várias tabelas internas (e portanto necessita de grande quantidade de memória)
- o uso de memória é importante para um programa em particular ser otimizado: um LOOP AT .... WHERE é preferível a um LOOP AT ... CHECK ... ENDLOOP porque estará sendo reduzido o número de comandos a serem interpretados

## 32. Processamento de grandes tabelas

Quando estiverem sendo manipuladas grandes tabelas é muito importante que seja processada a maior quantidade possível de informação num primeiro momento e, se possível, eliminar qualquer outra passagem nesta mesma tabela. Muitas vezes isso requer o uso de tabelas internas. Cada situação deve ser analisada e avaliada, para ver qual o procedimento correto a ser tomado: se armazenar o dado ou fazer novo acesso à tabela do Data Dictionary. Considere as seguintes questões:

qual o tamanho da tabela original comparado ao subset que seria armazenado em uma tabela interna ?

quanto espaço para armazenamento seria necessário para armazenar o dado em uma tabela interna ?

Se o processamento precisar ocorrer somente uma vez (por exemplo, armazenar alguns dados para posterior comparação), esteja certo que isso acontecerá for a de uma estrutura em forma de looping (por exemplo, Select, Loop, Do), de forma que o processamento não seja repetido desnecessariamente.

Dica: o operador IN consome muito tempo de máquina e não deveria ser usado em lugar do operador EQ .

Expressões lógicas são avaliadas da esquerda para a direita. A avaliação termina quando o resultado final foi estabelecido (eliminação ou inclusão completa). Todavia, quando se utiliza os operadores AND ou OR (por exemplo, em IF, WHERE) o critério de eliminação mais comum deve ocorrer primeiramente.

Exemplo:

A seguinte tabela deve ser lida, imprimindo-se os empregados da companhia ABC na Georgia

EMPLOYEE	NAME	COMPANY	STATE
001	Doe, Jr.	ABC	TX
002	Doe, M.	ABC	OK
003	Jones, A.	XYZ	TX
004	Jones, B.	ABC	GA
005	Jones, C.	ABC	TX
006	Jones, D.	XYZ	GA
007	Jones, E.	ABC	TX
008	Smith, A.	ABC	GA
009	Smith, B.	ABC	TX
010	Smith, C.	ABC	OK

```
if company = 'ABC' and
  state      = 'GA'
  write .....
endif.
```

➔ isto funciona, mas tanto os campos company quanto state precisam ser avaliados para oito de dez campos

```
if state      = 'GA' and
  company = 'ABC'
  write .....
endif.
```

➔ isto necessita menos tempo para ser processado, porque ele elimina todos os registros sem state = 'GA' e portanto tanto o campo company quanto state precisam ser avaliados para somente três registros.



### 33. Índices secundários:

- **Sempre esteja certo que o seu índice está sendo usado:**

“ A well-defined and properly implemented index is one of the best performance and tuning tool available. Because of the diversity of various database systems, however, and in particular various database optimizers, it is not possible to establish any hard-and-fast rules for creating and using database index. Additionally, it is impossible to guarantee that the database optimizers will use your index “.

Para saber se o índice definido está sendo usado:

- \* use o SQL Trace (transação ST05) para ativar e desativar o trace
- \* clique em trace on
- \* execute a transação em questão em uma outra sessão
- \* retorne a ST05 e clique em trace off
- \* clique em list trace para visualizar os resultados de seu trace
- \* clique em prepare, open ou reopen para selecionar o comando SQL que será avaliado
- \* clique em EXPLAIN para obter o resultado do comando SQL

## 34. Regras gerais para criação e uso de índices secundários:

Um índice suporta pesquisa a dados no banco de dados. Todas as tabelas standard SAP possuem um índice primário, o qual consiste de campos chaves que o usuário define quando da criação de uma tabela. Para os selects aonde a chave primária não pode ser utilizada na cláusula WHERE, ou quando selects não são qualificados, o banco de dados pesquisa a tabela inteira (executa uma varredura sequencial).

O Dicionário de Dados da SAP suporta até 16 índices para cada tabela. Geralmente, crie índices com menos que 5 campos.

Geralmente, se uma condição inclui OR, o otimizador pára o processamento (e chama uma varredura sequencial) tão logo o primeiro OR seja encontrado. A exceção possível é um OR que propõe uma condição separada e única para avaliação.

### Exemplo:

ZTABLE é definida com um índice secundário:

FIELD NAME	TYPE	LENGTH
FieldC	Char	3
FieldF	Char	2

Select \* from ztable where

fieldc = 'ABC' and  
(fieldf = '12' or '13')

→ isto é executado, mas não usa o índice conforme esperado

Select \* from ztable where

(fieldc = 'ABC' and fieldf = '12') or  
(fieldc = 'ABC' and fieldf = '13')

→ isto é executado usando o índice

### Informações gerais:

- Cláusulas IN são frequentemente interpretadas como condições OR e podem gerar alguns problemas
- Índices não são usados para condições IS (NOT) NULL
- A maior parte dos otimizadores tem problemas com condições OR, NEQ e LIKE

### Exemplo:

ZTABLE é definida com um índice secundário

FIELD NAME	TYPE	LENGTH
FieldA	Char	3
FieldB	Char	3
FieldC	Char	2
FieldD	Char	4

Select \* from ztable

Where fieldA = 'ABC' and  
field B = 'XYZ' and  
fieldC = '12'.

➔ isto funciona bem.

Select \* from ztable

Where fieldA = 'ABC' and  
field B = 'XYZ' and  
fieldD = 'DEFG'.

➔ isto não usa o índice conforme esperado e provavelmente invoca uma varredura sequencial da tabela baseada na chave primária

### **Considere a criação de um índice se um ou mais condições se aplicarem:**

- campos não chave são repetidamente usados para fazer seleções
- somente uma pequena parte de uma grande tabela é selecionada (< 5%)
- a cláusula WHERE de um comando SELECT é simples
- os campos que compõe o índice reduzem significativamente o conjunto de registros em uma cláusula WHERE

### **Não crie índices se uma das seguintes condições se aplicar:**

- a redundância de dados de armazenamento do índice cria problemas devido ao tamanho da tabela ou índice
- atualizações constantes criam excessivo overhead ou perda de performance durante a atualização do índice
- manutenção de um índice (como a reorganização) causa perda dos benefícios
- estão sendo usados campos cujos valores para a maioria dos registros na tabela é um valor inicial ou tem um valor que é idêntico para a maioria dos registros

## 35. Conceito de atualização em update task

Para aproveitar ao máximo os recursos computacionais de cada nível, a SAP criou um conceito de atualização em update task. Se for feita uma pesquisa em todos os programas "on-line" da SAP, será verificado que a atualização é sempre em funções que só fazem o update, e que não retornam valores nem mesmo exceptions se estas ocorrerem. (CALL FUNCTION .... IN UPDATE TASK).

Logo todas as aplicações fazem todas as checagens possíveis e imagináveis antes de chamarem estas funções, ou seja, as exceptions que podem ocorrer são somente as de BD (por exemplo table space, ou algo não previsto). Se uma aplicação faz "n" atualizações, esta irá chamar "n" funções (uma para cada update de um conjunto de tabelas) e, ao final, dará um commit work. O que o R/3 faz é simplesmente ir criando todos estes dados em uma espécie de log. No momento do commit work ele cria uma outra LUW (Logical Unit of Work) no servidor de BD e passa esta tarefa para este. Note que o programa (que esta sendo processado no servidor de aplicação), não fica esperando a resposta se o update ocorreu corretamente ou não, ele continua o processamento e termina a transação muitas vezes antes do servidor de DB ter terminado o processamento.

A vantagem óbvia deste processo é a diminuição de gargalos via este "tratamento de mensagens" entre as duas camadas (DB e aplicação).

Quando se utiliza o comando SET UPDATE TASK LOCAL o servidor de aplicação passa a executar o update (ou seja, não é criada uma outra LUW no servidor de DB) e este fica esperando a resposta se tudo ocorreu OK.

Um SET UPDATE TASK LOCAL é suficiente para todo o processamento do commit ser efetuado como local. Por isso, todos os updates da SAP procuram não utilizar o comando Set update task local. O default é o SET UPDATE (em funções). Isso evita que a performance seja degradada.

## 36. Comandos Select

Visando garantir a performance e evitar problemas futuros, cuidado com os comandos empregados. Existem alguns tipos de select mais eficientes que outros. Analise a aplicação que será desenvolvida e faça a melhor escolha.

## 37. ALGUNS TIPOS DE SELECT

`SELECT ...FROM <table> INTO TABLE <INTERNAL TABLE> .`

A estrutura da tabela interna deve corresponder à estrutura da tabela que está sendo acessada. O sistema lê os registros em conjunto, não individualmente, e os coloca dentro de uma internal table. Este processo é mais rápido que ler individualmente através de um LOOP e ir gravando os registros, um a um.

`SELECT * FROM <table> APPENDING TABLE <internal table>.`

Lê os registros e os inclui - não sobrepõe - em uma internal table.

`SELECT .... INTO CORRESPONDING FIELDS OF TABLE <itab>.`

Neste caso a estrutura da tabela interna não precisa corresponder à estrutura da tabela que está sendo acessada. <itab> é o nome da internal table. Movimentará os registros para as colunas definidas na internal table que possuam nome igual ao da tabela acessada).

Obs.: corresponding ou appending corresponding não exigem o endselect.

`SELECT ..... APPENDING CORRESPONDING FIELDS OF TABLE <itab>.`

Lê e grava (não sobrepõe) os dados em uma internal table que possua nomes idênticos aos nomes da tabela que está sendo lida.

`SELECT SINGLE * FROM SPFLI WHERE .....<campo>..... EQ ... <conteúdo>`

Toda vez que se usa select single \* a chave primária completa deve ser especificada. Se a chave especificada não é qualificada, você receberá uma mensagem de warning e a performance ficará prejudicada.

No caso de haver a necessidade de acessar um único registro via select, as opções são: select \* ..... seguido de comando exit OU select \* ... up to 1 rows. Neste caso não é necessário especificar a chave completa.

SELECT \* FROM ...<tabela>

Quando não se impõe nenhum tipo de restrição, ocorre uma varredura sequencial dos registros da tabela. Quando se utiliza grandes tabelas, isso obviamente afeta o runtime.

Select \* seleciona todas as colunas de uma tabela. É melhor sempre especificar as colunas, pois em caso de tabelas com muitas colunas, prejudicará performance.

SELECT \* FROM <tabela> WHERE <campo> eq <conteúdo>.

Lê todos os registros da tabela especificada onde o campo é igual ao conteúdo especificado. O ideal é que se qualifique a cláusula WHERE tanto mais quanto seja possível. Atentar que isso é válido para tabelas do tipo Pool e Transparent. Para Cluster, seguir as recomendações dadas anteriormente.

SELECT <a1> <a2> ... INTO (<f1>, <f2>, ... ) FROM ....<tabela>  
WHERE ..... .

Lê as colunas especificadas (a1, a2). Após INTO deverão ser especificadas as áreas de trabalho auxiliares (f1, f2). O número de colunas lidas deverá ser igual ao número de work-areas especificadas.

SELECT \* FROM <table> WHERE <table field> BETWEEN <field1> and <field2>.

Ex.: field1 = 100 e field2 = 500. Pega inclusive 100 e 500. Você trabalha com o range.

SELECT \* FROM <table> WHERE <table field> LIKE ....'\_R%'.

\_ = a primeira letra não importa o que virá

a segunda deverá ser R (eu defini)

% = não importa a sequência de caracteres que virá.

(Varredura para que possa realizar a comparação)

SELECT MAX(campo)  
MIN(campo)  
AVG(campo)  
COUNT(\*) FROM <table> INTO (.....,.....,.....,.....)  
WHERE ..... .

AVG e SUM: somente para campos numéricos.

Não se usa endselect.

Mais rápido fazer uma rotina “à mão” que utilizar este comando.

SELECT \* FROM <table> WHERE <table field> IN (.....,.....).

Exemplo: select \* from <table> where campo1 in (123,1000) - podem ser valores ou literais. É igual a perguntar se campo1 é 123 ou 1000.

SELECT \* FROM <table> WHERE <table field> IN <internal table>.

Exemplo:

DATA : begin of ITAB occurs 10,  
          sign(1), option(2), low like sflight-price, high like sflight-price,  
          end of ITAB.

\* RANGES: ITAB for sflight-table

Move: 'l' to itab-sign, 'bt' to itab-option, '500' to itab-low, '1000' to itab-high.

Append itab. Move: 'l' to itab-sign, 'bt' to itab-option, '440' to itab-low.

Append itab.

SELECT \* FROM (<table>) INTO <work area>.

Exemplo: data: begin of WA,  
          line(100),  
          end of WA.

Select \* from (tablename) into WA

Write ....

Endselect.

Parameters: tablename(10) default 'SPFLI'.

Obs.: especificando o nome da tabela dinamicamente no select statement sempre consome mais tempo de CPU que especificando estaticamente no programa.

SELECT \* FROM <table> FOR ALL ENTRIES IN <internal table> WHERE  
          campo1 = <conteúdo> and  
          campo2 = <conteúdo>

Defino uma tabela interna. Alimento os campos desta tabela interna. (move e append).  
No meu select campo1 e campo2 serão os campos definidos e alimentados na tabela interna.

Esta é uma excelente solução quando se trabalha com grandes tabelas.  
O select for all entries simula a funcionalidade join.



SELECT \* FROM <table> ORDER BY <field1> <field2> ... PRIMARY KEY.

Obs.: Classifica a tabela interna numa área auxiliar, sem afetar a tabela original. Evitar o uso de sorts dentro de um select. Consome mais tempo que descarregar os dados em uma tabela interna e classificá-los.

SELECT carrid MIN( price ) max( price ) INTO (carrid, minimum, maximum) FROM sflight GROUP BY carrid.

(Todos os campos que eu quero que apareçam na minha lista eu preciso especificar após a cláusula GROUP BY. Carrid, maximum e minimum são campos auxiliares. Se o nome do database não é conhecido até runtime não se pode especificar a cláusula GROUP BY).

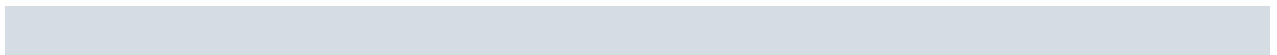
Performance não é tão boa (funções MIN, MAX, etc. geram uma varredura sequencial na tabela).

SELECT \* FROM <table> BYPASSING BUFFER.

(Usado para ler diretamente da tabela original, e não do buffer).

OBS.: Select single \* sempre com chave completa especificada. Particularidade do Abap/4.

Select \* - procurar evitar. Informar as colunas que serão necessárias, apenas.



## 38. Dicas para otimização do código

- Use o comando FREE para liberar espaço em internal tables; Sempre usar os comandos Clear / Refresh após o fim de um LOOP;
- Evite comparações num SELECT com campos numéricos versus campos alfanuméricos; o sistema perde tempo para conversão;
- Testar SY-SUBRC após cada acesso ao banco de dados;
- O comando MOVE-CORRESPONDING é bom para tabelas pequenas. É interessante que a tabela interna contenha os campos na sequência em que serão movimentados;
- Ao utilizar o comando CASE, codificar sempre a cláusula WHEN OTHERS;
- Sempre identifique se um SORT é ascending ou descending e especifique a cláusula BY <fields>. Caso contrário, todos os campos serão classificados.
- Evitar lógicas do tipo IF not CPOA = CPOB. É mais claro codificar IF CPOA ne CPOB.
- Evitar construções do tipo:

SORT tabela1, SORT tabela2, SORT tabela3.

LOOP tabela1, LOOP tabela2, LOOP tabela3

Para cada SORT fazer o LOOP correspondente. Aí então iniciar novo SORT e LOOP, e assim por diante.

- Campos chave devem ser sempre os primeiros campos da tabela. Assim, todos os demais campos serão comprimidos;
- 
- SELECT (para Transparent e Pool Tables): a cláusula WHERE deve conter, preferencialmente, os campos chaves e demais campos que possam restringir a pesquisa;
- 
- SELECT (para Cluster Tables): só os campos chaves devem ser especificados na cláusula WHERE. Os demais devem ser checados através do comando CHECK;
- O conhecimento do conteúdo dos dados de uma tabela pode auxiliar no momento da codificação do comando select. O campo que ocorrer em número menor de vezes deverá constar na cláusula where antes daquele que ocorre um número maior de vezes, caso seja necessário satisfazer a ambas condições. Isso faz com que o processamento seja mais ágil.
- Manuseio de tabelas: Estudar a possibilidade de manuseio em tabelas internas para agilizar o processo. Analisar também o uso de comando select, sendo o que melhor se adapte a situação em questão.

## 39. Ferramentas para auxiliar os desenvolvedores

Existem algumas ferramentas que auxiliam os desenvolvedores a descobrir erros (errors, warnings, etc), bem como avaliar a performance de seus programas.

Path: Tools – Abap/4 Workbench – Test - Runtime Analysis

Existe um pushbutton chamado Tips and Tricks – Clicando-se nele poderá ser feita comparação entre comandos Select (diferentes tipos). Dando double-click sobre um dos exemplos você passará para outra tela, onde você visualizará a medida do tempo em microsegundos de ambos, servindo de base de comparação.

Neste mesmo path, pode-se utilizar as facilidades do Runtime Analysis para verificar a performance de seu programa. Ao final da execução você poderá acessar informações sobre o seu programa (gráficos, acesso a tabelas, etc). Para tanto clique o pushbutton ANALYSE, que somente aparecerá depois que for informado o nome do programa e/ou transação e clicado execute.

O SQL Trace (transação ST05) é outra facilidade que pode ser utilizada para trilhar a lógica de sua aplicação e verificar possíveis pontos de correção. Você poderá visualizar os comandos de acesso a banco de dados utilizados, obter informações sobre um comando específico, visualizar os índices que estão sendo utilizados (ver tópico 5.3 sobre como utilizar o SQL Trace).

O Extended Program Check é outra facilidade que deve ser empregada visando manter o seu código o mais correto possível. Nesta opção você poderá selecionar os itens que você deseja que sejam checados e o sistema apontará o seu parecer. Para tanto basta clicar os itens e o pushbutton PERFORM CHECK. Selecionando uma linha você poderá ver os detalhes, bem como posicionando o cursor em uma linha e clicando o pushbutton DISPLAY ALL CHANGES.

## 40. Interfaces Batch ou Conversões:

Segundo a documentação da SAP, existem três métodos para conversão: Direct input, Call transaction e o Batch Input.

- **Direct Input:** é um dos métodos para transferência de dados do sistema legado para o sistema R/3. É considerado o método mais rápido. Um arquivo sequencial com dados é gerado como um arquivo texto para processamento por alguns function modules especiais. Estas funções executam todas as checagens normais para garantir a integridade de dados. Quando os registros são processados com sucesso, eles são gravados diretamente nas correspondentes tabelas do banco de dados da aplicação. Na ocorrência de erros, os dados errados são passados para uma rotina de manuseio de exceção. O gargalo associado com o processamento do dialog e update são eliminados neste caso. Para todas as necessidades de transferência de dados, especialmente com transações de alto volume, o direct input é o método mais indicado. Nota: este método deveria ser usado em todas as situações nas quais funções de aplicação utilizando esta tecnologia existam.
- **Call Transaction:** é o próximo método mais rápido para processamento de dados do sistema legado para o sistema R/3. Neste caso, o programa de transferência de dados processa os dados do arquivo sequencial e chama a transação desejada usando um comando Abap. Dados de um arquivo sequencial são processados via telas de aplicação para uma única transação. A lógica de aplicação executa todos os checks e a validação dos dados. Este é um processamento síncrono. Quando o processamento do dialog é realizado com sucesso, o processamento do update é chamado pela aplicação correspondente para executar todas as solicitações do banco de dados. A escolha pode ser feita entre update síncrono ou assíncrono. Nenhum protocolo de erros ou saída para lidar com os erros é fornecida por esta técnica. É responsabilidade do programador da aplicação incorporar rotinas de execução e funções dos protocolos desejados como parte do programa de transferência de dados. A nossa recomendação é usar o método Call Transaction em todos os casos em que não existe um programa Direct Input. Somente em situações de lidar com erros nós recomendamos que o batch-input tradicional seja realizado para posterior processamento. Resumo: em contraste com o Batch-Input, o Call Transaction permite que sejam passados dados diretamente ao dialog interface sem usar uma fila. Para armazenar estes dados temporariamente, você usa uma internal table (uma tabela BDC, a qual tem a mesma estrutura daquela utilizada no batch-input). Antes de entrar dados na BDC table é necessário fazer uma verificação nos dados. Call transaction: pode ser executado imediatamente.

- **Batch-Input:** o batch-input tem sido tradicionalmente selecionado como um método de implementação de programas de transferência de dados. Um benefício que este método traz sobre o Call Transaction é que o batch-input tem um utilitário responsável pela administração e gerenciamento das funções do batch-input. Além do mais, utilitários existem para monitorar e dar manutenção em tarefas associadas com este método. Assim, não há necessidade de nenhuma programação adicional para análise de exceções e funções de protocolo. Dados vindos de um arquivo sequencial são processados via telas de aplicação e armazenados numa sessão batch-input. Por definição, uma sessão é uma coleção de dados de transação para uma ou mais transações. As sessões batch-input são fisicamente armazenadas pelo sistema num banco de dados como uma fila. Estas sessões podem conter tanto registros de dados corretos quanto incorretos. O batch-input trata tanto o update síncrono como o assíncrono. O método batch-input, em contraste com o método Call Transaction, pode transferir dados do sistema legado para o sistema R/3 para múltiplas transações da aplicação. Todavia, nenhuma nova transação é iniciada até que a transação anterior tenha sido gravada no correspondente banco de dados durante o processamento das sessões de batch-input. Também, sessões de batch-input não podem ser geradas em paralelo. Ele pode ser processado de três modos: foreground, display errors only ou background.

Este método oferece excelente capacidade para gerenciamento de erros e é detalhado para análise do protocolo da transação.

Resumo: é gerado um log e pode ser programada a execução. Em caso de erro, o registro pode ser corrigido e inserido novamente. Através deste recurso, todos os dados são entrados na transação original SAP, ou seja, seguindo todas as consistências necessárias, evitando assim que erros de integridade sejam cometidos.

## 41. AVALIAÇÃO:

Sempre execute alguns testes dentro do processo de transferência de dados e avalie qual é o método mais apropriado para uso, dado o volume de dados, tamanho, recursos e performance do sistema.

Todos os fatores sendo iguais, o método Direct Input deve ser usado em todos os casos em que tais funções estejam disponíveis ou novas funções para transferência de dados precisem ser desenvolvidas.

Para escolher entre o Call Transaction e o Batch-Input, sempre use o Call Transaction primeiro e manuseie as exceções então através do método de batch-input. Esta

estratégia é altamente recomendada em implementação de soluções de programação para transferência de dados com sistemas legados.

- Exercícios para evoluir a lógica de programação com todos os conteúdos aplicados até agora (18/09/2019).

➤ **Debug ABAP**

- Manuseando o Debug
- Configurações do Debug
- Debug Velho/Debug Novo
- Pontos de Parada
  - Sintaxe
  - Função
  - Linha
  - Watch Point
  - Mensagem
- **Abas**
  - Desktop 1
  - Desktop 2
  - Desktop 3
  - Standard
  - Estruturas
  - Tabelas
  - Objetos
- **Ferramentas Ocultas no Debug**