NOMENCLATURA & PADRÕES ABAP

Conteúdo

INTRODUÇÃO

O objetivo deste documento é estabelecer padrões de desenvolvimento ABAP para a Companhia Brasileira de Cartuchos - CBC, tanto no que tange a nomenclaturas internas de código quanto a técnicas de otimização de desempenho aceitáveis.

Por se tratar de um ambiente R/3 complexo que poderá abranger várias empresas é necessário ter sempre em mente as seguintes premissas:

Alguns desenvolvimentos locais atendem empresas com grande quantidade de informação transacional.

Tabelas com histórico de crescimento geométrico e/ou exponencial devem ser evitadas.

Há informação oficial sobre volumes de dados, portanto não pode ser assumido baixo volume de dados para nenhuma pesquisa.

Todos os objetos criados pela Equipe Técnica de ABAP deverão seguir as nomenclaturas propostas nesse documento.

Segue abaixo a relação de empresas:

Sigla	Empresa	Segmento
GL	Global	Global para no mínimo 80 % das Unidades
CB	CBC	Companhia Brasileira de Cartuchos
CD	CBC Brasil	CBC Brasil Com.Distrib.

O sistema R/3 apresenta uma classificação para as suas aplicações conforme a tabela ilustrada abaixo, essa classificação corresponde a dois caracteres, os quais serão usados na composição da nomenclatura dos objetos desenvolvidos pelos consultores.

Como auxílio ao consultor ABAP para definir a nomenclatura de objetos, seguem se os módulos SAP, caso o desenvolvimento a ser realizado não esteja contido na relação abaixo, entre em contato com a equipe responsável, para inclusão da NOVA sigla.

SIGLA	MÓDULO SAP
AM	Asset Management
BC	Basis (administração e programas técnicos)
BP	Business Planning and Simulation
BS	BCS – Business Consolidation Services
BW	Business Information Warehouse
CA	Cross Application (válido para vários módulos)
CF	CFM -Corporate Finance Management
CM	CRM – Customer Relationship Management
CO	Controlling
FI	Financial Accounting
HR	Human Resources
IS	Industry Solutions
LE	Logistic Execution
MM	Materials Management
PM	Plant Maintenance
PP	Production Planning
PW	Desenvolvimentos SONDA PROCWORK feitos pelo cliente
OS	Project System
QM	Quality Management
SD	Sales & Distribution
TR	Treasury
WF	Workflow
WM	Warehouse Management

1. NOMENCLATURA

Como convenção, adotaremos apenas a letra ‘Z’ para identificação de objetos R/3 desenvolvidos, tratando-se de produtos específicos desenvolvidos pelo projeto de forma a complementar os módulos do R/3 que não atendem as necessidades da empresa na sua totalidade.

1.1. CLASSE DE DESENVOLVIMENTO ou PACOTE

Antes de se iniciar qualquer desenvolvimento é necessária a criação de pacote, que grosso modo pode ser definido como um aglutinador de objetos. Basicamente para a CBC, foram criadas tantas classes de desenvolvimento quanto são os módulos do SAP que estão sendo utilizados. Caso seja necessária uma nova definição ela deve ser feita pela equipe responsável, a regra a ser seguida deverá ser:

O padrão a ser utilizado na nomenclatura do objeto ‘CLASSE DE DESENVOLVIMENTO’ corresponde a forma:

ZPMM_XXXXXXXXXXXX

Onde:

Z	**Representa Pacote desenvolvido pelo cliente**
P	Constante que identifica o objeto como Classe de Desenvolvimento
MM	Representa Módulo SAP (ver Tabela Módulo SAP)
“_”	Obrigatório “_”
X...X	Descrição coerente, referente utilização do pacote (sem caracteres especiais, exceto “_”)

Exemplo: ZPMM_TRANSPORTAR (Pacote para MM transportáveis para outros ambientes.)

IMPORTANTE:

Ao copiar programas standard, é obrigatória a utilização do Pacote ZP_COPIA.

Para programas de aplicação de notas SAP, é obrigatória a utilização do Pacote ZP_NOTAS.

Nos tópicos relacionados a seguir, está ilustrada a forma de definição de nomenclatura a ser empregada na identificação dos objetos.

1.2. TRANSPORTE DE REQUESTS

As change requests devem seguir a nomenclatura abaixo para descrição:

EE.MM.SS: XXXXXXXXXXXXXXXXXXX

Onde:

EE	**Representa Empresa (ver tabela de Empresas)**
“.”	Obrigatório “.”
MM	Representa Módulo SAP (ver Tabela Módulo SAP)
“.”	Caso tenha sub-modulo
SS	Sub-módulo, caso seja aplicado (Opcional)
“:”	Obrigatório “:”
XX..X	Descrição coerente, referente utilização da Request (sem caracteres especiais, exceto “_”.

No caso de aplicação de notas as change requests devem seguir a mesma nomenclatura, porém a descrição deve ser clara quanto ao número da nota que está sendo aplicada:

Exemplos:

GL.FI.AP: Relatório Contas a Pagar

A CR é GLOBAL para módulo de FI e sub-módulo AP

IDENTIFICAÇÃO DO OBJETO

A identificação de um objeto ABAP/4 deverá estar presente, sempre que o objeto ABAP/4 o permitir e na forma como será apresentado nesse tópico. Em termos gerais, pode-se considerar como parte integrante de todo código gerado, independendo do tipo do mesmo.

1.2.1. COMENTÁRIOS

Utilizar as informações abaixo, para integrar o comentário dos programas:

```
*---------------------------------------------------------------------
-
*  Empresa..: <Informar a Empresa que utilizará este programa, se
houver
* mais de uma empresa, informar GLOBAL
* Programa.: <Nome do Programa>
* Tipo.....: <Tipo de Programa>
* Módulo...: <Módulo SAP>
* Transação: <Transação(ões) utilizada(s)>
* Descrição: <Breve descrição do programa>
```

```
* Autor....: <Nome Consultor ABAP>
* Data.....: <dd/mm/aaaa>
* User Exit: <Se for include, informar a user exit e o projeto
*---------------------------------------------------------------------
```

1.2.2. ESTRUTURAÇÃO DO CÓDIGO FONTE

Todos os códigos de executáveis (REPORT) devem ser codificados na sequência definida a seguir, para ajuda na procura de elementos no ABAP.

REPORT declaration.
TABLES declaration.
DATA Workfields definition.
DATA Internal table definition from existing tables with INCLUDE STRUCTURE statements.
DATA Other internal table definition.
CONSTANTS
RANGES definition.
FIELD-GROUPS definition.
PARAMETERS and **SELECT-OPTIONS** definition (sequence depends on the screen layout).
MODULE coding.
INITIALIZATION event.
AT SELECTION SCREEN ON specific field.
AT SELECTION-SCREEN.
START-OF-SELECTION.
GET segment (sequence depend on the logical database structure. GET segment LATE comes after).
END-OF-SELECTION.
TOP-OF-PAGE.
END-OF-PAGE.
AT LINE-SELECTION.
AT USER-COMMAND.
AT PF”nn”.
FORM coding (in sequence of call in the events or other forms).

Esta seqüência de código é aplicável para os novos desenvolvimentos ABAP. Para os programas importados de outros ambientes, o desenvolvedor pode manter o estilo já existente.

1.2.3. ALTERAÇÕES EM CÓDIGO PRÉ-EXISTENTE

Caso seja necessário efetuar alterações em códigos pré-existentes, seja por melhorias ou correções, deve ser utilizada a seguinte estruturação.

A. Inclusão:

*** Inicio de Inclusão -<login R/3> -<data> -<Descrição Breve>
MOVE: 3 TO V_COMECO.
*** Final de Inclusão -<login R/3> -<data> -<Descrição Breve>

B. Modificação:

*** Inicio de Alteracao -<login R/3> -<data> -<Descrição Breve>
* MOVE: 1 TO V_COMECO.
MOVE: 3 TO V_COMECO.
*** Final de Alteração -<login R/3> -<data> -<Descrição Breve>

C. Exclusão:

*** Inicio de Exclusão -<login R/3> -<data> -<Descrição Breve>
* MOVE: 1 TO V_COMECO.
*** Final de Exclusão -<login R/3> -<data> -<Descrição Breve>

Onde,

<login R/3>: login do usuário que está efetuando a alteração.
<data>: data em que se está efetuando a modificação.
<Descrição Breve>: Descrição da alteração e/ou número do Chamado.

Vale salientar que um mesmo ajuste de programa pode conter todas as operações citadas acima. Bem como, um desenvolvimento novo nunca vai conter nenhuma das situações acima.

Esta documentação é particularmente importante para melhorias e enhancements, onde qualquer modificação pode parar o sistema Produtivo, sendo assim deve estar presente em todas essas situações e as demais cabíveis.

1.3. OBJETOS DE PROGRAMA

Os objetos ABAP/4 estão relacionados, com as dimensões, conforme a tabela abaixo:

OBJETO	RELEASE 4.X
Programa	30 caracteres
Include	30 caracteres
Variante	14 caracteres
Campo Global	30 caracteres
Evento	30 caracteres

Sub-Rotinas (Forms)	30 caracteres
Macro	30 caracteres
Tela	04 caracteres
Status GUI	20 caracteres
Título GUI	20 caracteres
Transação	20 caracteres
Módulo de Diálogo	30 caracteres
Classe de Desenvolvimento	30 caracteres
Módulo de Função	30 caracteres
Banco de Dados Lógico	20 caracteres
ID Parâmetro SET/GET	20 caracteres
Menu de Área	20 caracteres
Classe de Mensagem	20 caracteres
Número da Mensagem	20 caracteres

1.3.1. PROGRAMA

O padrão a ser utilizado na nomenclatura de objetos do tipo 'PROGRAMA' corresponde à forma:

ZEETMMNNNN_XXXXXXXXXXXXXXXXX

Onde:

Z	Representa programa desenvolvido pelo Cliente
EE	Representa Empresa (ver tabela de Empresas)
T	Representa Tipo de Programa (ver Tabela Tipo de Programa)
MM	Representa Módulo SAP (ver Tabela Módulo SAP)
NNNN	Número Sequencial – 0001 a 9999, seguindo a seqüência dos programas já criados. Exemplo: 0001, 0002, 0003, etc, procurar sempre por ZEETMM* e verificar a próxima numeração.
"_"	Obrigatório "_"
X..X	Descrição coerente, referente utilização do programa (sem caracteres especiais, exceto "_".

Exemplo: ZGLCMM0001_CARGA_MATERIAL (Programa Global para Carga de Materiais).

SIGLA	TIPO DE PROGRAMA
C	Conversão/Migração de Dados
E	Extensão (Aplicações do Cliente)
F	Programas auxiliares e cópias de SAPscript
I	Interfaces
R	Relatórios
X	Cross-Aplication (IDOC, ALE, Etc...)

1.3.1.1. PROGRAMAS CE – COMÉRCIO EXTERIOR

O padrão a ser utilizado na nomenclatura de objetos do tipo 'PROGRAMAS CE' corresponde à forma:

ZEETMMMNNNN_XXXXXXXXXXXXXXXXX

Onde:

Z	Representa programa desenvolvido pelo Cliente
EE	Representa Empresa (ver tabela de Empresas)
T	Representa Tipo de Programa (ver Tabela Tipo de Programa)
MMM	Representa Módulo de Comércio Exterior (ver Tabela Modulo CE)
NNNN	Número Sequencial – 0001 a 9999, seguindo a seqüência dos programas já criados. Exemplo: 0001, 0002, 0003, etc, procurar sempre por ZEETMM* e verificar a próxima numeração.
"_"	Obrigatório "_"
X..X	Descrição coerente, referente utilização do programa (sem caracteres especiais, exceto "_".

Exemplo : ZGLCCEI0001_CARGA_MATERIAL (Programa Global para Carga de Materiais de Comercio Exterior de Importação).

SIGLA	MÓDULO CE
CEE	Comercio Exterior – Exportação
CEI	Comercio Exterior – Importação
CEC	Comercio Exterior – Câmbio
CED	Comercio Exterior – Draw Back

OBS.: TODOS OS OBJETOS DE **COMÉRCIO** EXTERIOR DEVEM SER CRIADOS NA CLASSE DE DESENVOLVIMENTO (PACOTE) ZPCE.

1.3.2. MÓDULO DE DIÁLOGO

O padrão a ser utilizado na nomenclatura de objetos do tipo 'MÓDULO DE DIÁLOGO' corresponde a forma:

ZEEDMMNNNN_XXXXXXXXXXXXXXXXX

Onde:

Z	Representa módulo de diálogo desenvolvido pelo Cliente
EE	Representa Empresa (ver tabela de Empresas)
D	Constante que identifica o objeto como sendo um módulo de Diálogo.

MM	Representa Módulo SAP (ver Tabela Módulo SAP)
NNNN	Número Seqüencial – 0001 a 9999, seguindo a seqüência dos programas já criados. Exemplo: 0001, 0002, 0003, etc, procurar sempre por ZEEDMM* e verificar a próxima numeração.
“_”	Obrigatório “_”
X...X	Descrição coerente, referente utilização do módulo de diálogo (sem caracteres especiais, exceto “_”)

Exemplo: ZGLDMM0001_SOLICITA_DADOS

1.3.2.1. MÓDULO DE DIÁLOGO – CE – COMÉRCIO EXTERIOR

O padrão a ser utilizado na nomenclatura de objetos do tipo ‘MÓDULO DE DIÁLOGO CE’ corresponde a forma:

ZEEDMMMNNNN_XXXXXXXXXXXXXXXXXX

Onde:

Z	Representa módulo de diálogo desenvolvido pelo Cliente
EE	Representa Empresa (ver tabela de Empresas)
D	Constante que identifica o objeto como sendo um diálogo módulo de diálogo
MMM	Representa Módulo de Comércio Exterior (ver Tabela Módulo CE)
NNNN	Número Sequencial – 0001 a 9999, seguindo a seqüência dos programas já criados. Exemplo: 0001, 0002, 0003, etc, procurar sempre por ZEEDMM* e verificar a próxima numeração.
“_”	Obrigatório “_”
X...X	Descrição coerente, referente utilização do módulo de diálogo (sem caracteres especiais, exceto “_”)

Exemplo: ZGLDCEE0001_SOLICITA_DADOS

1.3.3. INCLUDE

Os objetos do tipo ‘INCLUDE’ apresentarão sua nomenclatura como os objetos do tipo ‘PROGRAMA’, a menos da constante de identificação e a empresa, que nesse caso será ‘I’ ao invés de ‘P’.

ZIMMNNNN_XXXXXXXXXXXXXXXXXX

Onde:

Z	Representa include desenvolvido pelo Cliente
I	Constante que identifica o objeto como sendo um Include
MM	Representa Módulo SAP (ver Tabela Módulo SAP)
NNNN	Número Sequencial – 0001 a 9999, seguindo a sequência dos programas já criados. Exemplo: 0001, 0002, 0003, etc, procurar sempre por ZIMM* e verificar a próxima numeração.
"_"	Obrigatório "_"
X..X	Descrição coerente, referente utilização da include (sem caracteres especiais, exceto "_")

1.3.3.1. MÓDULO DE DIÁLOGO – CE – COMÉRCIO EXTERIOR

Os objetos do tipo 'INCLUDE CE' apresentarão sua nomenclatura como os objetos do tipo 'PROGRAMA CE', a menos da constante de identificação e a empresa, que nesse caso será 'I' ao invés de 'P'.

ZIMMMNNNN_XXXXXXXXXXXXXXXXXXXX

Onde:

Z	Representa include desenvolvido pelo Cliente
I	Constante que identifica o objeto como sendo um Include
MMM	Representa Módulo de Comércio Exterior (ver Tabela Módulo CE)
NNNN	Número Sequencial – 0001 a 9999, seguindo a sequência dos programas já criados. Exemplo: 0001, 0002, 0003, etc, procurar sempre por ZIMM* e verificar a próxima numeração.
"_"	Obrigatório "_"
X..X	Descrição coerente, referente utilização da include (sem caracteres especiais, exceto "_")

1.3.4. VARIÁVEIS

Todas as variáveis devem ser tecnicamente definidas, ou seja, mesmo que sejam variáveis tipo caractere onde a definição no comando DATA, é implícito, é importante sua declaração de tipo, bem como tamanho.

O consultor ABAP deverá nomear todas as variáveis conforme ilustração, seguida do caractere 'underscore' ('_'), mais o nome, ou nomes que melhor identifiquem o conteúdo da variável com o caractere 'underscore' entre os nomes.

TIPO	PREFIXO	EXEMPLO
Parameters	P_	PARAMETERS: P_BUKRS LIKE T001-BUKRS.
Select-options	S_	SELECTION-OPTIONS: S_BELNR FOR BKPF-BELNR.
Ranges	R_	RANGES: R_WERKS FOR T001W-WERKS.
Variável Global Local	V_ L_	DATA: V_CNT TYPE P. DATA: L_CVV TYPE P.
Field groups/ field symbols	F_	FIELD-GROUPS: HEADER, F_LINE
Tabelas internas	T_	DATA: BEGIN OF T_T001 OCCURS 0. INCLUDE STRUCTURE T001.' DATA: END OF T_T001. DATA: BEGIN OF T_MATNR OCCURS 0, MATNR(8) TYPE C, END OF T_MATNR.
Field-String ou Estruturas de Dados/Workareas	W_	DATA: BEGIN OF W_PERSON, NAME(20) TYPE C, AGE TYPE I, END OF W_PERSON.
Types	Y_	TYPES: BEGIN OF Y_PERSON, FIELD1 LIKE XFIELD1, FIELD2 LIKE XFIELD2, END OF Y_PERSON. OR TYPE S: Y_FIELD1 TYPE C.
Constants	C_	CONSTANTS: C_NBDAYS VALUE 7.

Ainda em declaração de variáveis é importante destacar a diferença entre a definição Y_ e W_, onde o primeiro refere-se à definição de uma estrutura de campos, mas nesse ponto ainda não é nem uma tabela interna tão pouco uma field string ou estrutura de dados.

1.3.5. MACRO

Os objetos do tipo 'MACRO' definidos pelo consultor ABAP, deve apresentar a forma:

ZMMM_XXXXXXXXXXXXXXXXXXXXXXXXX

Considere no ‘corpo’ do objeto que conterá os macros, uma seção de declaração devidamente comentada para as mesmas.

Onde:

Z	Representa macro desenvolvido pelo Cliente
M	Constante que identifica o objeto como sendo um macro
MM	Representa Módulo SAP (ver Tabela Módulo SAP)
“_”	Obrigatório “_”
X...X	Descrição coerente, referente utilização do Macro (sem caracteres especiais, exceto “_”)

Exemplo : ZMBC_EXIBE_DATA (Macro para exibição de data).

1.3.6. SUB-ROTINAS (FORMS E MÓDULOS PBO E PAI)

O padrão a ser utilizado na nomenclatura de objetos do tipo ‘SUB-ROTINA’ corresponde a forma:

ZR_XXXXXXXXXXXXXXXXXXXXXXXXXX

Onde:

Z	Representa Sub-rotina
R	Constante que identifica o objeto como sendo sub-rotina conforme tabela abaixo.
“_”	Obrigatório “_”
X...X	Descrição coerente, referente utilização da rotina (sem caracteres especiais, exceto “_”)

SIGLA	SUB ROTINAS
F	Forms
M	MODULOS PBO E PAI

Exemplo: ZM_UPDATE_FLAG (Modulo atualizar variável)

1.3.7. TELA

A identificação dos objetos do tipo ‘TELA’ deverá apresentar apenas caracteres numéricos. A tela inicial deve ser necessariamente a 9000.

1.3.8. STATUS GUI

O padrão a ser utilizado na nomenclatura de objetos do tipo 'STATUS GUI' corresponde a forma:

ZSMM_XXXXXXXXXXXXXXX

Onde:

Z	Representa status GUI desenvolvido pelo Cliente
S	Constante que identifica o objeto como sendo um status GUI
MM	Representa Módulo SAP (ver Tabela Módulo SAP)
"_"	Obrigatório "_"
X...X	Descrição coerente, referente utilização do Status GUI (sem caracteres especiais, exceto "_")

Exemplo: ZSMM_EXIBE_ITEM

1.3.9. TÍTULO GUI

O padrão a ser utilizado na nomenclatura de objetos do tipo 'STATUS GUI' corresponde a forma:

ZUMM_XXXXXXXXXXXXXXX

Onde:

Z	Representa título GUI desenvolvido pelo Cliente
U	Constante que identifica o objeto como sendo um título GUI
MM	Representa Módulo SAP (ver Tabela Módulo SAP)
"_"	Obrigatório "_"
X...X	Descrição coerente, referente utilização do Título GUI (sem caracteres especiais, exceto "_")

Exemplo: ZUMM_EXIBE_ITEM

1.3.10. TRANSAÇÃO

Todos os desenvolvimentos de projetos, independentemente de não terem suas chamadas explicitas pelo sistema, e sim eventualmente serem chamados por outros sistemas devem ter definidas transações.

O padrão a ser utilizado na nomenclatura de objetos do tipo 'TRANSAÇÃO' corresponde à forma:

ZXEEMMNNN

Z	Representa transação desenvolvida pelo Cliente
X	T – Transação ou R – Relatório
EE	Representa Empresa (ver tabela de Empresas)
MM	Representa Módulo SAP (ver Tabela Módulo SAP)
NNN	Número Sequencial 3 dígitos – 001 a 999, seguindo a seqüência das transações já criadas. Exemplo: 001, 002, 003, etc, procurar sempre por ZXEEMMNNN e verificar a próxima numeração.

Exemplo:

ZTGLSD001 (Transação Global do módulo SD).

ZRGLSD001 (Relatório Global do módulo SD).

*** IMPORTANTE:**

Não deve ser enviado programa com nomenclatura GL para criação de transação de desenvolvimento de unidades específicas.

1.3.11. OBJETOS PRIVADOS LOCAIS

A forma para esse tipo de objeto está definida como:

ZOMM_XXXXXXXXXXXXXXX

Onde:

Z	Representa objeto privado Local desenvolvido pelo Cliente
O	Constante que identifica o objeto privado Local
MM	Representa Módulo SAP (ver Tabela Módulo SAP)
"_"	Obrigatório "_"
X..X	Descrição coerente, referente utilização do objeto privado Local (sem caracteres especiais, exceto "_")

1.4. OBJETOS DO DICIONÁRIO

Os objetos ABAP/4 desse grupo estão relacionados, com as respectivas dimensões, conforme a tabela abaixo:

OBJETO	RELEASE 4.X
Tabela	30 caracteres
Estrutura	30 caracteres
Visão	30 caracteres
Elemento de Dados	30 caracteres
Domínio	30 caracteres
Objeto de Bloqueio	30 caracteres
Ajuda para Pesquisa	30 caracteres
Grupo de Tipos	05 caracteres

O padrão a ser utilizado na nomenclatura de objetos do 'Dicionário' corresponde à forma:

ZXXMM_XXXXXXXXXXXXXXXXXXXXXXXX

Onde:

Z	Representa objeto desenvolvido pelo Cliente
XX	Constante que identifica o objeto conforme a tabela abaixo.
MM	Representa Módulo SAP (ver Tabela Módulo SAP)
"_"	Obrigatório "_"
X...X	Descrição coerente, referente utilização do objeto do dicionário de dados (sem caracteres especiais, exceto "_")

O fator que proverá a diferença entre os objetos será o valor da constante 'XX', que assumirá um dos seguintes valores possíveis, conforme a tabela abaixo:

Objetos do Dicionário	Valor da constante 'XX'
Tabela	TB
Estrutura	ST
Visão	VW
Elemento de Dados	DE
Domínio	DO
Objeto de Bloqueio	BO
Ajuda para Pesquisa	SH
Grupo de Tipos	TY

Para os objetos 'Objeto de Bloqueio', deve ser inserida a letra 'E' no início do nome do objeto, conforme abaixo:

ZEBOMM_XXXXXXXXXXXXXXXXXXXXXXXX,

Onde:

Z	Representa objeto desenvolvido pelo Cliente
E	É uma constante obrigatória para criação de objetos de bloqueio
BO	É a Identificação de objetos de bloqueio
MM	É o Módulo correspondente
‘_’	É obrigatório ‘_’
X..X	é uma descrição coerente.

Para os objetos ‘Ajuda de Pesquisa’, podem ser definidos IDs. Um ID descreve os possíveis caminhos de busca para a procura do termo. Os campos ou combinações dos campos de busca são definidos no ID. Seguindo a forma ‘**?**’ onde a variável ‘**?**’ assumirá o valor entre 0 – Z.

Para os objetos ‘GRUPO DE TIPOS’ apresenta-se com dimensão 5 e a sua nomenclatura seguirá a forma **‘ZXXMMK**’ onde a variável **‘XX’** assumirá o valor **‘TY’** e a variável **‘K’**, que se apresenta mais flexível, poderá assumir qualquer valor alfanumérico.

1.5. OBJETOS DE GRUPO DE FUNÇÃO

Os objetos ABAP desse grupo estão relacionados, com as respectivas dimensões, conforme a tabela abaixo:

OBJETO	RELEASE 4.X
Grupo	26 caracteres
Módulo de Função	30 caracteres
Campo Global	30 caracteres
Evento	30 caracteres
Módulo PBO	30 caracteres
Módulo PAI	30 caracteres
Sub-Programa	30 caracteres
Macro	30 caracteres
Tela	04 caracteres
Status GUI	20 caracteres
Título GUI	20 caracteres
Include	30 caracteres
Transação *	20 caracteres
Módulo de Diálogo	30 caracteres

*** Ver tópico 1.3.10 TRANSAÇÃO**

A forma a ser adotada para objetos do tipo ‘GRUPO’ é:

ZGMM_XXXXXXXXXXXXXXX

E para objetos do tipo ‘MÓDULO DE FUNÇÃO’ (exceto USER-EXITS) é:

ZFMM_XXXXXXXXXXXXXXX

Onde:

Z	Representa Grupo ou Módulo de Função desenvolvido pelo Cliente
G	Constante que identifica o objeto como grupo
F	Constante que identifica o objeto como módulo de função
MM	Representa Módulo SAP (ver Tabela Módulo SAP)
“_”	Obrigatório “_”
X..X	Descrição coerente, referente utilização do módulo de função (sem caracteres especiais, exceto “_”)

Exemplo: ZFMM_CALCULA_IMPOSTO

Quando da utilização dos módulos de função, a declaração do mesmo objeto deve ser feita na sua totalidade dentro do código do programa. Isso é particularmente importante nos seguintes aspectos:

A. Declarações incompletas de módulos de função omitindo o parâmetro EXCEPTIONS, causam um processamento não integro por parte do SAP, ou seja, se uma função omitir essa informação, mesmo que o processamento não tenha sido bem sucedido a informação que será enviada pelo código de retorno é sempre Zero (0), o que é de conhecimento, o SAP considera sucesso. Ao mesmo tempo, não é valido identificar parâmetros de exceção que não estejam definidos no objeto, o sistema não é capaz de enviar um código de retorno eficiente.

B. Após cada declaração de função no R/3, que de alguma forma retorna status de processamento seja um código de retorno ou uma tabela, essa informação deve necessariamente ser validade, caso contrário o próprio sistema acusará essa não conformidade na ferramenta de verificação estendida, e o desenvolvimento será retornado para ajuste pela FSW.

1.6. ENHANCEMENTS – EXITS

Os objetos como USER-EXITS, MENU-EXITS, SCREEN-EXITS e BADIs deverão seguir o padrão definido pela SAP.

Caso se faça necessário a implementação de códigos específicos para campos de tela, as FIELD-EXITS deverão ser substituídos por BADIs ou User-Exits. Não podendo de maneira nenhuma serem implementadas sem a devida autorização do

projeto, uma vez que não é mais recomendada pela SAP e a mesma empresa não fornece suporte para esse objeto.

Os PROJETOS e as IMPLEMENTAÇÕES para BADIs devem seguir a seguinte forma:

ZJMMXXXXXX

Onde:

Z	Representa Projeto desenvolvido pelo Cliente
J	Constante que identifica o Projeto
MM	Representa Módulo SAP (ver Tabela Módulo SAP)
X..X	Breve descrição coerente, referente utilização do Projeto (sem caracteres especiais, exceto "_") no caso das Implementações de BADIs, a sugestão é incluir na descrição a definição que está sendo utilizada.

É estritamente proibido "COMMIT WORK" e "CHECK" dentro de qualquer USER-EXIT ou IMPLEMENTAÇÕES de BADIs.

1.6.1. MANUTENÇÃO DA MV45AFZZ OU EXITS DA MESMA CATEGORIA

Em caso de manutenção em exits como a MV45AFZZ ou qualquer outra da mesma categoria, é importante ter em mente que uma modificação mal feita pode parar o sistema produtivo.

A. Cada rotina deve conter um include master, que receberá o mesmo nome da rotina como segue:

```
*---------------------------------------------------------------------*
* FORM USEREXIT_FIELD_MODIFICATION *
*---------------------------------------------------------------------*
* This userexit can be used to modify the attributes of *
* screen fields. *
* This form is processed for each field in the screen. *
**
* The use of the fields screen-group1 to screen-group4 is: *
**
* Screen-group1: Automatic modification contolles by transaction *
* MFAW. *
* Screen-group2: Contents 'LOO' for steploop-fields. *
* Screen-group3: Used for modifiaction, which are dependent on *
* control tables or other fix information. *
* Screen-group4: Unused *
**
```

```
* For field mofications, which are dependent on the document *
* status, you can use the status field in the workareas *


* XVBAP for item status and XVBUK for header status. *
**
* This form is called from module FELDAUSWAHL. *
**
*---------------------------------------------------------------------*
FORM userexit_field_modification.
************************* PROCEDIMENTO CBC ************************

* Não incluir nenhum código nos FORMS DO INCLUDE MV45AFZZ.
* Todo GAP deverá ser desenvolvido dentro de um INCLUDE PRÓPRIO.
* Todo INCLUDE NOVO deverá ser declarado dentro do INCLUDE PAI
(abaixo).
INCLUDE zisd035_field_modification.
ENDFORM. "USEREXIT_FIELD_MODIFICATION
```

B. Dentro do include zisd035_field_modification, haverá sim a distinção de processo, dependente do campo empresa. Ou seja, cada empresa vai ganhar uma ou vários INCLUDES para que seus processos sejam eficientemente atendidos. Se duas empresas tiverem exatamente o mesmo processo, podem ser consideradas na mesma condição decisória, caso haja uma ou mais linhas de diferença, um novo item no comando decisório será criado para esta empresa e uma copia ajustada será criada como seu INCLUDE para atendê-la.

Esta seqüência de código é aplicável para os novos desenvolvimentos ABAP. Para os programas importados de outros ambientes, o desenvolvedor pode manter o estilo já existente.

1.7. SAPSCRIPT – FORMULÁRIOS

O padrão a ser utilizado na nomenclatura de objetos do tipo ‘SAPscript’ corresponde a forma:

ZEEFMMNNNN_XXXXXXXXXX

Onde:

Z	Representa SAPscript desenvolvido pelo Cliente

EE	Representa Empresa (ver tabela de Empresas)
F	Constante Obrigatória
MM	Representa Módulo SAP (ver Tabela Módulo SAP)
NNNN	Número Sequencial – 0001 a 9999, seguindo a seqüência dos programas já criados. Exemplo: 0001, 0002, 0003, etc, procurar sempre por ZEEFMM* e verificar a próxima numeração.
“_”	Obrigatório “_”
X..X	Descrição coerente, referente utilização do SAPscript (sem caracteres especiais, exceto “_”)

Exemplo : ZVCFSD0001_NOTA_FISCAL

Esta nomenclatura pode deve ser utilizada tanto para os formulários quando para os programas eventualmente necessários.

1.8. OUTROS OBJETOS

Esta seqüência de código é aplicável para os novos desenvolvimentos ABAP. Para os programas importados de outros ambientes, o desenvolvedor pode manter o estilo já existente.

1.8.1. BANCO DE DADOS LÓGICO

O padrão a ser utilizado na nomenclatura do objeto ‘BANCO DE DADOS LÓGICO’ corresponde à forma:

ZYMM_XXXXXXXXXXXXXXX

Onde:

Z	Representa Banco de Dados Lógico desenvolvido pelo Cliente
Y	Constante que identifica o Banco de Dados Lógico
MM	Representa Módulo SAP (ver Tabela Módulo SAP)
“_”	Obrigatório “_”
X..X	Descrição coerente, referente utilização do Banco de Dados Lógico (sem caracteres especiais, exceto “_”)

Exemplo : ZYMM_CARACTERISTICA

1.8.2. ID PARÂMETRO SET/GET

O objeto ‘ID’ usado como parâmetro junto aos comandos ‘SET’ e ‘GET’ devera ser identificados pelo consultor ABAP através do formato:

ID_XXXXXXX

Onde:

ID	Constante que identifica o objeto como um parâmetro ID
“_”	Obrigatório “_”
X..X	Descrição coerente, referente utilização do ID (sem caracteres especiais, exceto “_”)

É importante destacar, que caso sejam utilizados objetos clientes para armazenamento de dados em memória, deve-se:

A. No programa que carrega a variável de memória: Informar o programa onde o dado está sendo utilizado.

B. No programa que recebe o conteúdo da variável de memória: Informar o programa que carregou a variável.

1.8.3. MENU ÁREA

O padrão a ser utilizado na nomenclatura do objeto do tipo ‘MENU ÁREA’ corresponde a forma:

ZHMM_XXXXXXXXXXXXXXX

Onde:

Z	Representa Menu Área desenvolvido pelo Cliente
H	Constante que identifica o Menu Área
MM	Representa Módulo SAP (ver Tabela Módulo SAP)
“_”	Obrigatório “_”
X..X	Descrição coerente, referente utilização do Menu Área (sem caracteres especiais, exceto “_”)

Exemplo: ZHCF_REL_ESPECIAIS (Menu de Relatórios Especiais CFM)

1.8.4. CLASSE DE MENSAGEM

O padrão a ser utilizado na nomenclatura do objeto do tipo ‘CLASSE DE MENSAGEM’ corresponde a forma:

ZMMNN

Onde:

Z	Representa Classe de mensagem desenvolvido pelo Cliente
MM	Representa Módulo SAP (ver Tabela Módulo SAP)
NN	Número sequencial – 01 a 99

Exemplo: ZMM00 (Classe de Mensagem genérica de MM)

Recomenda-se utilizar sempre uma classe de mensagens já definida, exceto desenvolvimentos muito complexos, considerados sistemas.

1.8.5. NÚMERO DE MENSAGEM

O padrão a ser utilizado na nomenclatura do objeto do tipo ‘NUMERO DE MENSAGEM’ corresponde a forma:

NNN

Onde:

NNN	Seqüencial – 000 a ZZZ

1.8.6. OBJETOS DE AUTORIZAÇÃO

Todos os desenvolvimentos devem conter objetos de autorização por Empresa, ao menos.

O responsável pela identificação da atividade permitida para um objeto desenvolvido é a equipe funcional de projeto. Essa informação deve estar explícita na documentação Z400.

Por recomendação, objetos de autorização devem ser reutilizados sempre que possível de forma a minimizar a manutenção. Ainda sob recomendação da mesma equipe, é importante salientar que o parâmetro atividade ou ACTVT nunca deve receber o conteúdo asterisco, caso isso ocorra o desenvolvimento será devolvido para ajustes.

Para identificar os descritivos do parâmetro ACTVT disponíveis, ver Anexo 3.

Além disso, deve se estar atento a forma de declaração do parâmetro, o objeto não aceita conteúdos como: 1,2,3, por exemplo e sim, 01, 02, 03.

Ficam excluídos de declaração de objetos de autorização os seguintes desenvolvimentos: Fórmulas e Módulos de Função RFC com interfaceamento com o BizTalk. Onde o primeiro é uma convenção do projeto, a segunda e terceira se justificam pelo fato se estarem inerentes ao um código padrão SAP.

A criação de objetos de autorização é feita pela equipe de BASIS.

O padrão a ser utilizado na nomenclatura do objeto do tipo 'AUTHORITYCHECK' corresponde a forma:

ZO:XXXXXXX

Onde:

ZO:	Representa objeto Authority-Check (obrigatório)
X...X	Nome do programa que utiliza o objeto

1.8.7. MEMORY-ID

O padrão a ser utilizado na nomenclatura do objeto do tipo 'MEMORY-ID' corresponde a forma:

ID_XXXXXXX

Onde:

ID	Constante que identifica o objeto como um parâmetro ID
"_"	Obrigatório "_"
X..X	Descrição coerente, referente utilização do ID (sem caracteres especiais, exceto "_")

É importante destacar, que caso sejam utilizados objetos clientes para armazenamento de dados em memória, deve-se:

A. No programa que carrega a variável de memória: Informar o programa onde o dado está sendo utilizado.

B. No programa que recebe o conteúdo da variável de memória: Informar o programa que carregou a variável.

1.8.8. VARIANTE

O padrão a ser utilizado na nomenclatura de objetos do tipo 'VARIANTE' corresponde à forma:

ZEEVMM_XXXXXXX

Onde:

Z	Representa variável desenvolvida pelo Cliente
EE	Representa Empresa (ver tabela de Empresas)
V	Identifica o objeto como sendo uma variante
MM	Representa Módulo SAP (ver Tabela Módulo SAP)
“_”	Obrigatório “_”
X..X	Descrição coerente, referente utilização da variante (sem caracteres especiais, exceto “_”)

Exemplo: ZGLVMM_MATERIAL_FERT (Variante Global para Carga de Materiais do tipo FERT)

2. LAYOUT

Layout de telas e/ou relatórios, devem sempre seguir as recomendações feitas pela especificação funcional Z300, caso alguma complementação seja necessária para melhora de apresentação.

Em casos de relatórios, deve ser dada prioridade para apresentação ALV, desenvolvimento GRID, uma vez que facilita a visualização por parte do usuário, bem como fornece a ele, algumas ferramentas amigáveis de ajuste de layout.

Para relatórios ALV é obrigatória a inclusão de um campo variante para se escolher o layout de saída dos campos ALV.

3. DOCUMENTAÇÃO

Segue abaixo os documentos que serão sendo utilizados:

. Z300: Especificação funcional

. Z400: Especificação técnica

A. Histórico de Requests, para que o desenvolvimento seja de alguma forma rastreável, é necessário que os históricos de change requests sejam mantidos, essa informação aparecerá tanto na documentação quanto no cabeçalho do desenvolvimento.

B. Objetos de autorização, nesse ponto devem ser identificados todos os objetos de autorização que estão sendo utilizados pelo desenvolvimento bem com suas atividades permitidas. Caso não seja necessário objeto de autorização para o desenvolvimento, deve ser também justificado na documentação.

C. Código de transação, esse item é um dos que identifica a conformidade entre o código desenvolvido no ambiente e a documentação.

D. Apresentação geral do documento, ou seja, o documento deve ser descrito em cor preta, pois caso haja melhorias e modificações, os mesmos itens podem ser destacados com outras cores. Se do desenvolvimento inicial do documento se fizer necessários que sejam destacados partes de textos em cores diferenciadas, deve ser incluída uma legenda que justifique o destaque

4. QA

Algumas das regras importantes consideradas pelo QA de desenvolvimento:

A. Padrão de nomenclatura. Em concordância com este guia.

B. Objetos de autorização. Exceto módulos de função com comunicação para o BizTalk e fórmulas.

Verificação da existência do comando AUTHORITY-CHECK, os mesmos objetos não devem estar como comentário no programa.

Verificação pela utilização de objetos já existentes como padrão no sistema.

Não informar asterisco (*) para o campo atividade (ACTVT), pois está em discordância com as recomendações da Equipe de Segurança de Informação. Caso haja dúvidas, uma boa transação para entendimento de objetos de autorização é a SUIM, que está disponível para visualização e o descritivo de cada de cada valor do parâmetro ACTVT, pode ser verificado no Anexo 3.

C. Transação. Todos os objetos devem estar atrelados a transações e a transação deve ser cadastrada na SU24 com o objeto de autorização.

D. Comandos de seleção.

SELECT, a prioridade na seguinte seqüência: chave primária e índice.

Índices são autorizados até que todas as possibilidades de acesso a tabelas sejam testadas.

SELECT...FOR ALL ENTRIES, verificar o conteúdo da tabela interna referenciada antes da execução do comando.

SELECT SINGLE com chave primária completa.

E. HardCode. Não são autorizados, exceto Batch-Input, outros casos podem ser discutidos e acordados.

F. Verificação extendida de sintaxe. Para transporte sem QA, os desenvolvimentos NÃO podem apresentar ERROS de verificação extendida, somente mensagem, para o QA, a premissa é eliminar todas as mensagens possíveis apresentadas na SLIN. Acordos só acontecem se a solução não for identificada.

Códigos comentados. Devem ser excluídos, exceto melhorias em desenvolvimentos produtivos desse projeto.

H. Elementos de Texto. Não deixar elementos que sejam desnecessários.

I. Módulos de Função. Todos os objetos são verificados para checar o tratamento de código de retorno. Se a função apresentar na definição o parâmetro EXCEPTIONS, o programa não será aprovado até que o devido tratamento seja efetuado.

J. Modularização. Todo o programa deve ser modular. Ou seja, não exigimos que cada rotina tenha necessariamente uma tela de comprimento, no entanto, processos devem ser claramente separados de forma a facilitar a manutenção futura a qualquer desenvolvedor.

K. Otimização de código. Por exemplo, tabelas internas. Várias vezes as tabelas internas têm a mesma composição de campos, ao invés de serem citadas n vezes, deve se criar um estrutura e referenciá-la para todas as tabelas relevantes. A mesma observação se da para rotinas repetitivas.

L. Comparação de Igualdade. Sempre que possível deve ser comparado na primeira condição o IF, a igualdade ao invés da desigualdade.

M. Documentação. Programas sem documentação não são verificados, ficam em "standby" até todos os requisitos estarem disponíveis.

Deve estar no Solution Manager sob o item identificado no QA.

Informação mínima verificada: Existência de transação, validade do nome do programa, estruturação lógica, para esse item, não aceitamos cópia do programa na documentação, deve estar no mínimo em português estruturado.

Informação sobre o objeto de autorização utilizado, bem como os campos validados.

Conciliação entre a informação da documentação com objetos desenvolvidos.

N. Índices. Estatisticamente tabelas com mais de 5 índices não têm boa performance, consequentemente não autorizamos criação de novos índices para tabelas que tenham atingido o número máximo. A criação do índice é feita pela equipe de Basis.

O. Parâmetros IMPORT/EXPORT. Sempre que for utilizar parâmetros de importação e exportação, comentar acima do código para onde está enviando ou de onde está recebendo o parâmetro, a fim de facilitar para quem for efetuar uma manutenção no programa.

Exemplo:

```
* Importa o registro mestre do grupo de função ZGCM_AGENDA_CARREGAMENTO
import v_rmestre from memory id c_idregme.
```

Em especial, analisa-se também a estruturação do programa, o que significa que o QA está disponível a arrumar todo o programa, se esse ajuste melhorar o desempenho de processamento. Ou seja, alterar a seqüência de seleção de tabelas para melhorar acesso a registros, incluindo ou excluindo tabelas. Um bom exemplo prático são os acessos efetuados na VBFA, normalmente, utilizada com chave incompleta. Se for necessário, incluir outras tabelas de Vendas com acessos em chaves primárias para melhorar as informações na chave da VBFA.

Outro ponto relevante que vale a pena ser citado, são projetos muito complexos de desenvolvimento. Ou seja, não adianta desenvolver um programa com INNER JOIN para 15 tabelas se somente uma pessoa conseguir entender o assunto o suficiente para dar manutenção. Nesse caso, a prioridade é desmembrar o programa no nível do compreensível comum nem que isso custe um pouco de perda de desempenho.

ANEXO 2 - DICAS DE PERFORMANCE PARA DESENVOLVER OS PROGRAMAS.

A primeira e mais importante das dicas é que as ferramentas de gerenciamento e performance (SE30) e rastreabilidade de dados (ST05) do R/3 sejam utilizadas.

SELECT + CHECK X SELECT...WHERE

Sempre especifique as condições na cláusula WHERE ao invés de checá-las você mesmo com o comando CHECK.

O BD poderá utilizar um índice, caso exista algum que satisfaça a condição, e a carga sobre a rede é consideravelmente menor, em vista do menor número de registros trafegados.

SELECT USANDO ÍNDICE

Tente utilizar um índice para os comandos SELECT mais utilizados.

Você sempre usa um índice se especifica os campos de um índice (todos ou os primeiros, sempre na ordem definida no índice) concatenados por ANDs na cláusula WHERE do comando SELECT.

Observar que a utilização dos índices é realizada pelo otimizador de cada BD, portanto, em alguns casos, onde a cláusula WHERE possua uma complexidade muito alta, este pode não servir-se do índice esperado.

SELECT...ENDSELECT X SELECT SINGLE

Se você deseja apenas uma linha de uma tabela ou de uma view, utilize o comando SELECT SINGLE ao invés do laço SELECT-ENDSELECT.

O SELECT SINGLE necessita apenas de uma comunicação com o BD, enquanto o SELECTENDSELECT necessita de duas.

Se você não tiver a chave completa, não é aconselhável utilizar SELECT SINGLE, nesse caso utilize SELECT... UP TO 1 ROWS... ENDSELECT.

SELECT + APPEND X SELECT INTO TABLE

É sempre mais rápido utilizar a cláusula INTO TABLE no comando SELECT que utilizar o comando APPEND dentro de um laço SELECT...ENDSELECT., pois:

·A cláusula INTO TABLE realiza apenas uma interação com o BD e a rede, obtendo todos os dados, e os move para a memória numa operação única.

Portanto, 1 operação de I/O e 1 operação de memória;

O laço SELECT...ENDSELECT + APPEND realiza (n + 1) interações com o BD e a rede, e mais n movimentações para a memória, onde n é o número de linhas da tabela que satisfazem as condições. Portanto, em (n + 1) operações de I/O e n operações de memória.

SELECT...ENDSELECT X LOOP

É sempre mais rápido utilizar a cláusula INTO TABLE no comando SELECT e o comando LOOP que utilizar um laço SELECT...ENDSELECT., pois:

A cláusula INTO TABLE realiza apenas uma interação com o BD e a rede, obtendo todos os dados, e os move para a memória de numa operação única, e o laço do LOOP realiza n operações de memória, onde n é o número de linhas da tabela que satisfazem as condições. Portanto em 1 operação de I/O e (n + 1) operações de memória;

O laço SELECT...ENDSELECT realiza (n + 1) interações com o BD e com a rede, onde n é o número de linhas da tabela que satisfazem as condições.

Portanto em (n + 1) operações de I/O.

OPERAÇÕES AUTOMÁTICAS DO BD

É sempre mais rápido utilizar uma operação automática do BD (max, min, sum ou avg) que utilizar um laço SELECT...ENDSELECT., pois:

Os operadores automáticos são resolvidos de forma otimizada pelo BD, numa única operação com o BD e a rede, obtendo o dado, e a movimentação para a memória é realizada de uma única vez. Portanto em 1 operação de I/O e 1 operações de memória;

O laço SELECT..ENDSELECT realiza (n + 1) interações com o BD e a rede, onde n é o número de linhas da tabela que satisfazem as condições, e mais (2n + 1) operações na memória. Portanto em (n + 1) operações de I/O e (2n + 1) operações na memória.

SELECT UTILIZANDO VISÕES

Utilize visões para o acesso de tabelas que formem joins sempre que estas visões existam ou possam ser criadas.

O número de operações de I/O é, no mínimo, metade das necessárias ao acesso de todas as tabelas que contém os dados desejados.

A carga na rede é, da mesma forma, reduzida a, no pior dos casos, à metade.

SELECT DE COLUNAS ESPECÍFICAS

Sempre que possível especifique as colunas que devem ser lidas de uma tabela quando da utilização do comando SELECT.

A carga na rede é consideravelmente menor, em vista da menor quantidade de dados trafegando.

SELECT DE TABELAS BUFFERIZADAS

Para acessar tabelas que possuem pouca atualização e muitas leituras, procure utilizar o recurso do buffer do R/3.

O número de operações no BD é sensivelmente menor, pois as informações, após o primeiro acesso, não são buscadas no BD, mas diretamente no servidor de aplicações.

Da mesma forma, a carga da rede é bastante reduzida.

ATUALIZAÇÃO POR LINHA X ATUALIZAÇÃO POR BLOCO

Sempre que possível utilize comandos de atualização de tabelas por blocos de linhas, ao invés de utilizar uma linha de cada vez.

A comunicação freqüente entre o aplicativo e o BD provoca “overhead” considerável, tanto pelo número de operações a serem executadas pelo BD, quanto pela carga na rede.

UPDATE POR LINHA X UPDATE POR BLOCO

Sempre que possível utilize o comando UPDATE por coluna de tabela, ao invés de utilizar uma linha de cada vez, pois desta forma estará executando por bloco de linhas, com menos dados e de forma otimizada internamente pelo BD.

A comunicação freqüente entre o aplicativo e o BD provoca “overhead” considerável, tanto pelo número de operações a serem executadas pelo BD, quanto pela carga na rede.

OPERADORES PARA STRINGS

Ao invés de criar seus algoritmos, utilize os operadores específicos de verificação de conteúdo de “strings” do ABAP/4.

Estes operadores já são otimizados ao máximo, utilizando funções internas que diminuem consideravelmente a carga de trabalho da CPU.

Os operadores de verificação de string são:

CO (contains only);

CN (contains not only);

CA (contains any);

NA (contains not any);

CS (contains string);

NS (contains not string);

CP (contains pattern);

NP (contains not pattern).

FUNÇÕES X COMANDOS PARA STRINGS

Algumas funções para manipulação de strings estão obsoletas, tendo sido substituídas por comandos, o que torna sua execução muito mais eficiente.

Sempre que for executar alguma função de manipulação de strings, verifique se não existe algum comando, cláusula ou operador, que desempenhe a mesma função.

Alguns exemplos são:
CONCATENATE por STRING_CONCATENATE;
SPLIT por STRING_SPLIT;
STRLEN() por STRING_LENGTH;
WRITE...TO...CENTERED por STRING_CENTER;
WRITE...TO...RIGTH-JUSTIFIED por STRING_MOVE_RIGTH.

ALGORITMOS X COMANDOS PARA STRINGS

Ao invés de desenvolver algoritmo próprio para manipulação de strings, utilize os comandos já disponíveis, quando possível. Estes comandos são extremamente eficientes e deixam o código muito mais limpo e de fácil entendimento e manutenção.

Alguns destes comandos são:
CONCATENATE;
SPLIT;
SHIFT;
STRLEN();
CLEAR [WITH];

TABELAS CONDENSADAS

Ao invés de desenvolver algoritmo próprio para condensação de tabelas, utilize os comandos já disponíveis, quando possível. Estes comandos são extremamente eficientes e deixam o código muito mais limpo e de fácil entendimento e manutenção.

TABELAS SEM DUPLICIDADES

Ao invés de desenvolver algoritmo próprio para eliminação de duplicidades em tabelas, utilize os comandos já disponíveis, quando possível. Estes comandos são extremamente eficientes e deixam o código muito mais limpo e de fácil entendimento e manutenção.

BUSCA LINEAR X BUSCA BINÁRIA

A busca binária em tabelas internas com mais de 20 linhas é sensivelmente melhor que a busca linear. Pois a busca linear demanda F(n) segundos, em quanto a busca binária demanda F(log2(n)) segundos.

Portanto, tente manter sua tabela interna ordenada e utilize a busca binária.

CHAVE IMPLÍCITA X CHAVE EXPLÍCITA

Especifique, sempre que possível, a chave de acesso de uma tabela interna.

De outra forma esta terá que ser definida dinamicamente pelo sistema em tempo de execução.

BUSCA LINEAR X ÍNDICE SECUNDÁRIO

Se for necessário acessar uma tabela interna por uma chave que não a de ordenação repetidamente, crie seu próprio índice secundário.

Com um índice secundário você troca um acesso linear (F(n)) por dois acessos, um binário (F(log2(n))) e um direto (F(1)), cuja soma , na maioria dos casos, é menor que o acesso linear.

LOOP + CHECK X LOOP...WHERE

Sempre especifique as condições na cláusula WHERE ao invés de checá-las você mesmo com o comando CHECK, pois a execução da condição internamente é sempre melhor.

A performance pode ser melhorada se o comando LOOP...WHERE puder ser adicionado das cláusulas FROM I1 e TO I2.

MOVIMENTAÇÃO EXPLÍCITA X AUTOMÁTICA

Sempre que possível, utilize comandos de manipulação de tabelas internas que realizem as movimentações de campos automaticamente.

Os comandos de manipulação de tabelas internas que executam movimentações automaticamente são:

APPEND wa TO tab;

INSERT wa INTO tab;

COLLECT wa INTO tab;

MODIFY tab FROM wa;

READ TABLE tab INTO wa.

LOOP AT tab INTO wa.

COMPARAÇÃO DE TABELAS INTERNAS

Tabelas internas podem ser comparadas através como qualquer outro objeto de dados.

Duas tabelas internas são iguais se:

. Possuem o mesmo número de linhas;

. Cada par de linhas correspondentes possuem o mesmo conteúdo.

LOOPS ANINHADOS DE TABELAS INTERNAS

Se o número de linhas de TAB1 é n1, e o de TAB2 é n2, o tempo necessário para realizar os loops aninhados das tabelas é dado pelas funções:

F(n1 * n2), para o acesso por chave;

F(n1 + n2), para o algoritmo paralelo.

O algoritmo paralelo acima assume que a tabela TAB2 é uma tabela secundária que possui apenas entradas contidas na tabela TAB1. Caso essa tese não se mantenha, o algoritmo paralelo deverá ter sua complexidade aumentada, mas as suas características de performance permanecem as mesmas.

MODIFICAÇÃO DE TABELAS INTERNAS

Com o uso da cláusula TRANSPORTING no comando MODIFY a tarefa de alteração de uma linha de uma tabela interna pode ser acelerada.

A velocidade de alteração é proporcional ao tamanho das colunas a serem alteradas.

INCLUSÃO AO FINAL DE TABELAS INTERNAS

Com o uso da cláusula LINES OF no comando APPEND a tarefa de inclusão de uma tabela interna em outra tabela interna pode ser acelerada, pois a execução do comando pode ser transferida para o kernel.

INCLUSÃO NO MEIO DE TABELAS INTERNAS

Com o uso da cláusula LINES OF no comando INSERT a tarefa de inclusão de uma tabela interna no meio de outra tabela interna pode ser acelerada, pois a execução do comando pode ser transferida para o kernel.

ELIMINAÇÃO DE LINHAS DUPLICADAS

Com o uso da cláusula ADJACENTS DUPLICATES FROM ...COMPARING no comando DELETE a tarefa de eliminação das linhas duplicadas em uma tabela ordenada pode ser acelerada, pois a execução do comando pode ser transferida para o kernel.

ELIMINAÇÃO DE LINHAS EM SEQUÊNCIA

Com o uso da cláusula FROM ... TO no comando DELETE a tarefa de eliminação das linhas em seqüência em uma tabela pode ser acelerada, pois a execução do comando pode ser transferida para o kernel.

PROGRAMAS – SELEÇÃO POR DATAS

Para evitarmos problemas de performance em tabelas acessadas por campos que são datas, em todos os programas desenvolvidos é obrigatória a restrição por datas dentro de um período fechado. O período sempre deve ser definido com o analista funcional.

ANEXO 3 – DESCRITIVOS DOS PARÂMETROS DE ATIVIDADE

Atividade	Texto	Atividade	Texto
1	Anexar ou criar	49	Solicitar
2	Modificar	50	Deslocar
3	Exibir	51	Inicializar
4	Imprimir,processar mensagem	52	Modificar início aplicação
5	Bloquear	53	Exibir início aplicação
6	Eliminar	54	Exibir arquivo aplicação
7	Ativar,gerar	55	Modificar arquivo aplicação
8	Exibir documentos modificação	56	Exibir arquivo
9	Exibição de preço	57	Arquivar arquivo
10	Gravar	58	Exibir transferência
11	Modif.status intervalo numer.	59	Distribuir
12	Atualizar/gerar docs. modific.	60	Importar
13	Inicializar posições de número	61	Exportar
14	Seleção de campo: Gerar tela	62	Criar ledger
15	Seleção de campo: Atrib.tabela	63	Ativar
16	Executar	64	Gerar
17	Atualizar objeto inter.numer.	65	Reorganizar
18	Remessas de processam.coletivo	66	Atualizar
19	Faturas de processam.coletivo	67	Traduzir
20	Transportar sem tradução	68	modelagem
21	Transportar	69	Rejeitar
22	Entrar, incluir, atribuir	70	Administrar
23	Atualizar	71	Analisar
24	Arquivar	72	Planejar
25	Recarregar	73	Executar assinatura digital
26	Modif. Cliente grupo contas	74	Anular autorização
27	Exibir registros de totais	75	Diminuir
28	Exibir partidas individuais	76	Entrar
29	Exibir dados gravados	77	Pré-editar
30	Determinar	78	Atribuir
31	Confirmar	79	Atrib.função à função composta
32	Gravar	80	Imprimir
33	Ler	81	Programar
34	Escrever	82	Completar
35	Sair	83	Reconfirmar
36	Atualização ampliada	84	Liquidar
37	Aceitar	85	Estornar

38	Exercer	86	Transferir
39	Verificar	87	Devolver
40	Criar no banco de dados	88	Exercer
41	Eliminar no banco de dados	89	Forçar lançamento
42	Converter no banco de dados	90	Transferir
43	Liberar	91	Reativar
44	Marcar	93	Calcular
45	Autorizar	95	Desbloquear
47	Emprestar	97	Definir
48	Simular	98	marcar para liberação
99	Gerar listas de faturas	GL	Síntese total
A1	Reter	H1	Desativar
A2	Pagar	H2	Ativar registro em log
A3	Modificar status	H3	Desativar registro em log
A4	Reapresentar	KA	Ativar rescisão
A5	Exibir relatórios	KI	Knock in
A6	Ler com filtro	KO	Knock out
A7	Escrever com filtro	KS	Estornar rescisão
A8	Processar dados em massa	KU	Rescindir
A9	Enviar	L0	Todas as funções
AA	Imprimir novamente	L1	Volume de função nível 1
AB	Liquidar	L2	Volume de função nível 2
B1	Exibir valores permitidos	LM	Modificar mapeamento LDAP
B2	Encerrar tecnicamente	LS	Modificar comutador sinc. LDAP
B3	Derivar	MA	Desativar assistente modific.
B9	lançar obj.c/entr.preliminar	P0	Aceitar dados CCMS CSM
BD	Atual.objs.no sist.não propr.	P1	Processar dados CCMS CSM
BE	Projeção IMG	P2	Atualizar métodos CCMS CSM
C1	Atualização cartões pagamento	P3	Efet.logon Sist Remoto CCMS CSM
C2	Exibições de cartões pagamento	PA	Abrir período
C3	Atualizaç.autorizaçs.manuais	PB	Encerrar período
C4		PC	Abrir processamento ACON
C8	Confirmar modificação	PD	Encerrar procmto.unid.cons.
D1	Copiar	S1	Processar formulário
DL	Download	S2	Processar especificação
DP	Eliminar planejamento	U2	Executar ajuste volume vendas
E0	Gravar extrato	U3	Modificar ajuste volume vendas
E6	Eliminar extratos próprios	U4	Anexar ajuste de volume vendas
E7	Eliminar extratos externos	UL	Upload
EP	Dar prioridade a extrato	V1	Criar versão
FP	Modificar seleç.campo cliente	V2	Modificar versão
G1	Atualizar orçamento	V3	Exibir versão
G2	Faturar	V4	Eliminar versão

G3	Atualizar custos indiretos	V5	Transportar versão
G4	Atualizar reavaliação	V6	Eliminar cabeçalho de versão
G5	Pré-editar	VE	Criar um ID de ampliação
G6	Transf.orçamento	VF	Vencido
G7	Estornar		