ANEXO XI - TERMO DE REFERÊNCIA

PADRÃO DE OBJETOS E ESTRUTURA DE BANCO DE DADOS

1. Introdução

A definição de padrões para criação de objetos e estruturas de banco no âmbito da Enap faz parte da iniciativa de padronização da área de tecnologia da informação, buscando garantir uma melhor organização das bases de dados, contribuir de forma considerável para uma plena administração de dados, permitir o planejamento global a fim de evitar aplicações isoladas e a proliferação de dados incompatíveis e redundantes, obter maior disponibilidade e facilidade de interpretação, aumentar a utilização compartilhada e consequentemente a diminuição do seu custo, contribuir para a rapidez na resolução de problemas, facilidade de entendimento e conhecimento dos aplicativos e finalmente a melhoria da qualidade dos produtos e serviços prestados.

Este documento estabelece nomenclaturas, regras e procedimentos que devem ser seguidos para a criação de objetos no banco de dados, objetivando a padronização das definições das estruturas físicas e lógicas relacionadas com o armazenamento dos dados.

Entenda-se aqui por definição tanto a questão da nomenclatura, quanto a questão de como devem ser constituídos os objetos de banco de dados e estruturas de armazenamento, abrangendo, inclusive, regras sobre os scripts de criação e manipulação de dados.

Todos os servidores ou colaboradores deverão seguir as regras desta norma quanto se tratar de bases de dados dos sistemas de informação desenvolvidos ou mantidos no âmbito da Enap.

2. Nomenclatura

2.1. *TABLE*

O nome da tabela, deverá expressar o que será armazenado em seu conteúdo, para facilitar a leitura e entendimento do modelo de dados (MER) e também a criação dos outros objetos que a utilizarem. Deve ser escrita em caixa baixa e não deve conter acentos ou plural.

Ex: pessoa, funcionario, usuario.

2.2. COLUMN

O nome de uma coluna de uma tabela deve ser alusivo à sua finalidade dentro da tabela; no caso de um nome muito extenso, deve-se abreviá-lo mantendo, no entanto, a finalidade da Coluna. Deve, obrigatoriamente, começar com um grupo das três letras iniciais do nome da tabela seguido da finalidade da coluna, conforme exemplo a seguir:

Tabela	Prefixo	Exemplo	Tipo de dado utilizado
funcionario	fun	funid	integer
funcionario	fun	funnome	varchar
curso	cur	curid	integer
curso	cur	curnome	varchar
turma	tur	turdatainicio	date
turma	tur	turnome	varchar

Obs.: Para as colunas que são utilizadas como chave estrangeira, deve-se manter o nome original da mesma (herdado da tabela pai). Exemplo: tabela turma, chave estrangeira curid (chave id vindo da tabela curso).

2.3. DATABASE

A denominação da database deve seguir o padrão (prefixo) db + nome do sistema. Exemplo db_avalienap.

2.4. OWNER

A denominação do *owner* de uma database deve seguir de prefixo usr + nome da database (sem o prefixo db). Exemplo: usr_avalienap.

Obs.: Por questões de segurança e padronização de arquitetura, não utilizamos usuário postgres como owner. Cada database deverá ter seu próprio owner, individualmente.

3. Utilização dos Tipos de Dados

3.1. *Varchar* - Utilizado para dados que representam cadeias de caracteres com um tamanho variável, mas com possibilidade de se estimar o tamanho máximo. Cadeias de caracteres deste tipo podem variar de tamanho, mas sem ultrapassar o tamanho máximo.

3.2. *Char* - Utilizado para dados que representam cadeias de caracteres com tamanho fixo e invariável, ou seja, todas as cadeias de caracteres representadas por este tipo possuem um tamanho único.

3.3. *Numeric* - Utilizado para dados que representam números de uma forma genérica.

3.4. *Date* - Utilizado para dados que representam datas cronológicas informando obrigatoriamente dia, mês e ano. O formato a ser utilizado será o padrão brasileiro dd/mm/aaaa.

3.5. *Text* - Utilizado para dados que representam cadeias de caracteres com um tamanho variável e sem possibilidade de se estimar o tamanho máximo. Cadeias de caracteres deste tipo constituem textos que podem ser longos ou curtos e devem ser utilizados quando o conteúdo puder exceder 4.000 caracteres, caso contrário utilizar o tipo varchar.

3.6. *Timestamp* - utilizado para dados que representam data e hora. O formato a ser utilizado será o padrão brasileiro dd/mm/aaaa hh:mm:ss

4. SCRIPTS

Os *scripts* para criação de Tabelas devem ter as Colunas ordenadas segundo os critérios a seguir:

a) Primeiro devem vir as Colunas cujo preenchimento é obrigatório (NOT NULL) na seguinte ordem:

1. Chave Primária;
2. Chave(s) Estrangeira(s);
3. Demais colunas obrigatórias.

b) Segundo devem vir as Colunas cujo preenchimento não é obrigatório (NULL) mas que tem maior probabilidade de serem preenchidas futuramente.

c) Terceiro devem vir as Colunas cujo preenchimento não é obrigatório (NULL) e que tem maior probabilidade de não serem preenchidas.

5. TRIGGER

Para a criação do nome de um *trigger*, deve ser utilizado o prefixo tg_ + nome da tabela + tipo de gatilho (inc, alt ou exc) caso a tabela tenha mais de um *trigger*, seguindo a mesma regra de abreviação utilizada para a criação do nome da tabela quando este ultrapassar o limite de 30 caracteres.

Ex: tg_municipio_inc, tg_uf_inc, tg_uf_alt

6. SEQUENCE

Para a criação do nome de uma *sequence*, deve ser utilizado o prefixo sq_ + nome da tabela, seguindo a mesma regra de abreviação utilizada para a criação do nome da tabela quando este ultrapassar o limite de 30 caracteres.

Ex: sg_municipio, sq_uf, sq_tipo.

7. VIEW

Para a criação do nome de uma visão, deve ser utilizado o prefixo vw_ + nome da tabela ou nome curto da tabela + complemento se for necessário, seguindo a mesma regra de abreviação utilizada para a criação do nome da tabela quando este ultrapassar o limite de 30 caracteres.

O nome curto da tabela pode ser criado obedecendo qualquer critério, desde que seja único entre todas as tabelas do esquema

Ex: vw_municipio, vw_mun_regiao, vw_uf, vw_pessoa_endereco.

8. MATERIALIZED VIEW

Para a criação do nome de uma visão, deve ser utilizado o prefixo mv_ + nome da tabela ou nome curto da tabela + complemento se for necessário, seguindo a mesma regra de abreviação utilizada para a criação do nome da tabela quando este ultrapassar o limite de 30 caracteres.

Ex: mv_municipio, mv_mun_regiao, mv_uf, mv_pessoa_endereco.

9. ÍNDICE

Uma tabela pode ter um ou mais índices, sendo o nome do índice formado pelo nome curto da tabela + nome da coluna(s) + "_idx", seguindo a mesma regra de abreviação utilizada para a criação do nome da tabela, quando este ultrapassar o limite de 30 caracteres. O nome curto da tabela pode ser criado obedecendo qualquer critério, desde que seja único entre todas as tabelas do esquema.

Para a criação de um índice na tabela de município para a coluna nome, poderíamos utilizar o nome curto mncp (retirado as vogais da palavra MUNICIPIO) e criar o nome do índice como: mncp_nome_idx.

Para os indexes compostos, a nomenclatura deve seguir o seguinte padrão:

Nome ou nome curto da tabela + Nome ou nome curto da Coluna 1 + "_" + Nome ou Nome curto da coluna 2 + "_idx".

Ex: func_nome_func_datnasc_idx (índice referente às colunas nome e dtnascimento do funcionario).

10. FOREING KEY

Uma tabela pode ter uma ou mais chaves estrangeiras, sendo o nome da chave estrangeira formada pelo nome curto da tabela + nome da coluna + '_fk', seguindo a mesma regra de abreviação utilizada para a criação do nome da tabela quando este ultrapassar o limite de 30 caracteres.

O nome curto da tabela pode ser criado obedecendo qualquer critério, desde que seja único entre todas as tabelas do esquema.

Para a criação de uma chave estrangeira na tabela de município para a coluna cod_uf, poderíamos utilizar o nome curto mun e criar o nome da chave estrangeira como: fk_mun_cod_uf.

Índices de chaves estrangeiras: o nome do índice referente à chave estrangeira deve ser formado pelo nome ou nome curto da tabela de referência + nome da coluna estrangeira + '_fk_idx'.

11. UNIQUE KEY

Uma tabela pode ter uma ou mais chaves únicas, sendo que o nome da chave única deve ser formado pelo nome curto da tabela + nome da(s) coluna(s) + '_uk', seguindo a mesma regra de abreviação utilizada para a criação do nome da tabela quando este ultrapassar o limite de 30 caracteres.

O nome curto da tabela pode ser criado obedecendo qualquer critério, desde que seja único entre todas as tabelas do esquema.

Para a criação de uma chave única na tabela de município para as colunas cod_uf e cod_municipio, poderíamos utilizar o nome curto MUN e criar o nome da chave estrangeira como: mun_cod_uf_cod_mun_uk

12. RULE

O nome da *RULE* deverá ser em minúsculo, no singular, sem caracteres especiais, de acordo com o seguinte padrão: rul_ + Nome significativo para a RULE.

Ex.: rul_sexo, rul_sim_nao

13. TYPE

O nome de um *Type* deverá ser em minúsculo, no singular, sem caracteres especiais, de acordo com o seguinte padrão: tp_ + nome do Type

Ex: tp_funcionario

14. SINÔNIMO

O nome de um Sinônimo deve ser exatamente igual ao nome dado ao objeto do banco de dados sem a parte do *Owner* do *Schema*. Apenas a Equipe de DBAs deve poder criar, alterar ou excluir Sinônimos.

15. PROCEDURE

O nome de uma *Procedure* deverá ser em minúsculo, no singular, sem caracteres especiais, de acordo com o seguinte padrão: proc_ + nome da procedure

Ex: proc_calcular_juro

16. FUNCTION

O nome de uma *Function* deverá ser em minúsculo, no singular, sem caracteres especiais, de acordo com o seguinte padrão: func_ + nome da *function*

Ex: func_converter_temperatura

17. PARÂMETROS DE ENTRADA DAS FUNÇÕES

Os nomes dos parâmetros utilizados na criação das *procedures*/*functions* devem ser em minúsculo, sem caracteres especiais, de acordo com o seguinte padrão:

p_ + nome do parâmetro

Sempre que possível deve ser utilizado o mesmo nome da coluna do banco de dados ao qual o parâmetro fizer referência, seja para consulta, alteração ou inclusão de dados**.**

Ex: p_cod_uf, p_num_pessoa

18. CONSTRAINTS TIPO CHECK

O nome de uma *Constraint* tipo *Check*, deverá ser em minúsculo, no singular, sem caracteres especiais, de acordo com o seguinte padrão: ck_ + nome da constraint

Ex: ck_deptartamento_deptno

19. SCHEMA

O nome de um *schema* deve ser em minúsculo, no singular, sem caracteres especiais, e, quando possível, igual à sigla do sistema ao qual pertencerá.

db_siglasistema, onde:

- db = Indicação que se trata de uma base de banco de dados.
- siglasistema = Nome (sigla) do sistema em letras minúsculas.

Ex: db_suap

20. GLOSSÁRIO

Banco de Dados Corporativo: é um banco de dados relacional contendo objetos (tabelas, visões, procedimentos, etc.) que são compartilhados e usados por diversas aplicações ou sistemas ao mesmo tempo, evitando assim redundância de dados e garantindo integridade das informações, além de outras vantagens. Ele é composto por outros três bancos de dados independentes, cada qual constituído por uma instância e uma base de dados. São assim denominados: Banco de Dados de Desenvolvimento, Banco de Dados de Homologação e Banco de Dados de Produção.

Banco de Dados de Desenvolvimento: corresponde a uma instância e a uma base de dados com o objetivo de armazenar objetos de banco de dados e dados para desenvolvimento dos novos sistemas ou aplicações, sendo um banco de dados relacional utilizado unicamente pelas equipes de desenvolvimento, devidamente autorizadas. Todo novo sistema ou aplicação deve ser desenvolvido utilizando este banco de dados, antes de ser colocado em homologação.

Banco de Dados de Homologação: corresponde a uma instância e a uma base de dados com o objetivo de armazenar objetos de banco de dados e dados para testes dos novos sistemas ou aplicações, sendo um banco de dados relacional. É utilizado pelos usuários finais e as equipes de desenvolvimento, devidamente autorizadas. Todo novo sistema ou aplicação deve, antes de ser colocado em produção, ser colocado neste ambiente para testes e validação por parte do usuário final.

Banco de Dados de Produção: corresponde a uma instância e a uma base de dados com o objetivo de armazenar objetos de banco de dados e dados de sistemas ou aplicações já em produção, constituindo assim dados válidos e que são acessados por diversos usuários do banco de dados, devidamente autorizados. Também é um banco de dados relacional.

Column (coluna): é uma parte de uma Tabela que contém dados do mesmo tipo e que, semanticamente, são de mesma natureza.

Constraint (regra de integridade): correspondem a uma regra de integridade aplicada sobre uma determinada tabela, mantida no banco de dados. Em linhas gerais, *constraints*, são utilizadas para prevenir entrada inválida de dados, implementando regras a nível de tabela que são verificadas toda vez que uma linha é inserida, alterada ou excluída, se a regra não for respeitada, a operação não é concluída no banco de dados.

Datafile (arquivo de dados): é uma estrutura física (um arquivo do sistema operacional) onde fisicamente o conteúdo de uma t*ablespace* é armazenado. Uma t*ablespace* é constituída por um ou mais *datafiles*.

DBA: O termo DBA é uma sigla de origem inglesa para *Database Administrator*. Como no jargão técnico, no comércio, em empresas, enfim, em tudo que se relaciona com administração de banco de dados no mundo (inclusive no Brasil) se utiliza o termo DBA, ao invés da tradução para a língua local, resolvemos adotá-lo para referenciar aquele que é responsável por gerenciar e administrar o banco de dados.

Equipe de DBAs: é um grupo formado por dois ou mais técnicos e/ou analistas responsáveis pela administração do Banco de Dados Corporativo, devidamente treinados para tal tarefa de administração. A necessidade de uma equipe deve-se ao fato de que a administração de um banco de dados não deve ficar centralizada em uma única pessoa.

Equipe de Desenvolvimento: é qualquer grupo de dois ou mais técnicos, analistas e/ou programadores responsáveis por definir, projetar, desenvolver e implantar sistemas, sejam de uso local e restrito a Enap (intranet), seja de âmbito mundial (internet).

Function (função): possui a mesma definição de uma *procedure*, diferenciando apenas na obrigatoriedade de retornar um resultado no final de sua execução.

Index (índice): é um objeto de banco de dados pertencente a um *schema*, utilizado para acelerar o acesso e a recuperação de linhas em uma tabela através do uso de ponteiros, provendo assim um acesso direto e rápido às suas linhas.

Line (linha): corresponde a uma instância ou registro de uma Tabela.

Owner (proprietário): um *owner*, é um usuário do banco de dados, que tem poder total de ação sobre os objetos dos *schema* a que está associado, sem precisar de algum direito ou privilégio especial para isso, podendo executar comandos DML e DDL naturalmente.

Package (pacote): é um conjunto de procedimentos na linguagem PL/SQL, agrupados numa mesma estrutura armazenada no banco de dados. Um pacote tem duas partes: a especificação do pacote e o corpo do pacote.

Package Specification (especificação do pacote): também é conhecido como cabeçalho do pacote; e a parte do pacote que contém informações sobre o conteúdo do pacote (assinatura das funções), não contendo nenhum código PL/SQL.

Package Body (corpo do pacote): é a parte do Pacote que contém o código PL/SQL de todas as funções definidas na Especificação do Pacote.

Procedure (procedimento): *Procedures* são trechos de código na linguagem PL/SQL que são separados do fluxo principal de execução e podem ser chamadas uma ou mais vezes, desviam o fluxo do processamento para o seu corpo,

permitem manipular diretamente os dados do banco de dados e ao final de sua execução retornam para o lugar de onde foi chamada sem retornar resultado.

Role (papel/cargo): é um grupo de privilégios (de sistema e/ou sobre objetos do banco de dados) que podem ser concedidos a um ou mais usuários.

Rule (regra): é uma regra que restringe o conjunto de valores (domínio) que podem ser inseridos em campos do banco de dados.

Schema (esquema): um *schema* na terminologia Oracle é um conjunto de vários objetos de bancos de dados que estão associados a um usuário específico do banco de dados (também chamado de *owner* ou proprietário).

Sequence (sequência): é um objeto de banco de dados usado para gerar números únicos e sequenciais. É geralmente utilizada para implementar chave primária em uma Tabela. Ela é gerada e incrementada (ou decrementada) internamente por uma rotina do próprio banco de dados Oracle.

Synonyms (sinônimo): é um nome alternativo para um objeto do banco de dados.

Table (tabela): é a unidade básica de armazenamento de dados sendo composta por linhas e colunas.

Tablespace: corresponde a uma área lógica para armazenamento. Dentro de uma *tablespace* são armazenados objetos do banco de dados além dos próprios dados, inclusive os dados sobre o banco de dados (dicionário de dados).

Terceiros: toda e qualquer entidade (empresa, microempresa, conjunto de um ou mais técnicos em informática, tais como analistas, programadores, etc. que atuem como profissional liberal, consultores independentes ou ligados a alguma corporação, etc.) que porventura presta ou venha a prestar serviços para a Enap, contratada temporariamente ou não, independente da atividade que desempenhe ou venha a desempenhar relacionada ao desenvolvimento de sistemas e banco de dados.

Trigger (gatilho): é um conjunto de comandos PL/SQL que são executados quando se realiza uma inserção, alteração e/ou exclusão em uma Tabela. A *trigger* é executada (disparada) antes da consumação da operação sobre a tabela. Se, por algum motivo, a execução da *trigger* não for completada, a operação sobre a tabela também não será. Geralmente são utilizadas para implementar regras de integridade complexas e que por isso não podem ser implementadas por *Constraints*; também são utilizadas para execução de ações diversas desencadeadas pela operação de inserção, alteração ou exclusão sobre a tabela.

Type (tipo): são tipos de dados de *alias* ou um tipo definido pelo usuário no banco de dados atual, baseado em um tipo nativo do banco de dados.

Username (nome do usuário): corresponde a um nome pelo qual o usuário é identificado e reconhecido no banco de dados, não sendo necessariamente parte do nome da pessoa que é o usuário do banco de dados. Por padrão, deve ser utilizado o número do CPF ou CNPJ da pessoa sem a formatação, ou seja, somente os caracteres numéricos.

View (visão): representam logicamente subconjuntos de uma ou mais tabelas e se comportam como tabelas quanto a inserção, alteração, exclusão e consulta de dados; ou seja, pode-se inserir, alterar, excluir ou consultar dados em uma *view*, respeitando sempre as regras de integridade e de segurança que estiverem definidas sobre as Tabelas envolvidas na constituição da *view*.