O que é um Sistemas de Gerenciamento de Banco de Dados

Objetivos de aprendizagem:

* Identificar os sistemas de gerenciamento de banco de dados (SGBD).
* Criar dicionário de dados.
* Diferenciar os sistemas de gerenciamento de banco de dados.

1. Introdução

Neste capítulo, você vai aprender a identificar e reconhecer os diversos tipos de sistemas de gerenciamento de bancos de dados. Assim, será capaz de discernir os padrões de cada um deles. Além disso, você verá para que serve um dicionário de dados e como criá-lo. Por fim, você vai saber diferenciar os sistemas de gerenciamento de banco de dados de forma natural e rápida.

### 2. Os sistemas de gerenciamento de banco de dados (SGBDs)

Para realizar o acesso a um banco de dados, é imprescindível utilizar um sistema de gerenciamento de banco de dados (SGBD). Aí surge a grande dúvida de qual é a diferença entre um banco de dados e um SGBD. Para facilitar o seu entendimento, vamos relembrar o conceito de bancos de dados, mencionado por Korth, Silberschatz e Sudarshan (2012), que definem que banco de dados é uma coleção de dados inter-relacionados, representando informações sobre um domínio específico. O SGDB, por sua vez, é definido por Macêdo (2012, documento on-line) como “[...] uma coleção de programas que permitem ao usuário definir, construir e manipular bases de dados para as mais diversas finalidades”.

De forma mais simples e para resumir esses conceitos, você deve considerar que bancos de dados são a porção que representa os dados efetivamente salvos, e o SGBD é a ferramenta que fica encarregada de salvar, editar, deletar ou ainda garantir que esses dados estejam disponíveis quando a aplicação ou o usuário requisitar.

Para visualizar essa discussão, veja, a seguir, a Figura que mostra um fluxo de trabalho baseado em um relatório solicitado pelo usuário em um sistema qualquer e as etapas necessárias para que o resultado seja exibido na tela do sistema:



Ao observar a Figura, vemos que o fluxo de trabalho é dividido entre alguns degraus até chegar à informação: sistema, SGBD e o dado necessário dentro do banco de dados. Entenda que o sistema não irá acessar diretamente o valor desejado, e sim pedir que SGBD o faça.

Novamente surge outra grande dúvida: “mas por que colocar mais complexidade ao pedir que o SGBD execute a busca e retorne com os valores desejados e não o sistema final?”.

Antigamente, quando não existiam os SGBDs, a própria aplicação final (ou sistema final) deveria fazer essa busca e esse tratamento. O problema disso é o aumento da complexidade do código por parte do desenvolvedor, além do fato de que a manipulação de dados é algo muito específico. O resultado eram sistemas com erros diversos e genéricos, com grande carga de trabalho para a correção; quando chegavam a ela, percebiam que era uma exceção que o banco de dados havia retornado e que não havia sido tratada.

Com o surgimento dos SGBDs, esses problemas acabaram. A aplicação final deve se preocupar em chamar o SGBD conforme sua necessidade, passar os valores de busca ou edição desejados e aguardar o retorno.

Esses SGBDs foram e são desenvolvidos com o único propósito de gerenciar e manipular os dados e as variáveis dos bancos de dados, maximizando o foco dos desenvolvedores de aplicações naquilo que elas se propõem a fazer e entregar valor, em vez de manipular dados.

Um exemplo dessa complexidade maior ao não utilizar um SGBD é a concorrência. Imagine que um usuário solicita a edição de um cadastro e salva os novos valores no mesmo segundo que outro usuário também salva a informação desse mesmo cadastro. Como deve proceder a validação? Quem deve ter prioridade de tarefa? Qual será a forma de resolver esse impasse? Tudo isso deveria estar dentro do código da aplicação final, caso não utilize um SGBD.

Para garantir que um SGBD esteja devidamente cumprindo seu papel, Macêdo (2012) menciona que é importante que ele atenda a algumas características básicas de trabalho. Estas características elementares são:

* controle de redundâncias: Os valores de um banco de dados estão armazenados única e exclusivamente em um local, evitando problemas com inconsistência devido a valores diferentes. Caso o SGBD tenha replicação de dados, ela ocorrerá após o banco de dados máster (principal) ter salvo totalmente os dados, garantindo a integridade;
* compartilhamento de dados: O SGBD deve garantir que a concorrência por um mesmo valor de dados ocorra sem problemas;
* controle de acesso: Deve ser assegurado o controle de acesso ao banco de dados pelo SGBD. Para controlar este tipo de atividade, o SGBD utiliza a relação de usuários e permissões;
* interfaceamento: Como o SGBD é o único responsável pelo acesso aos dados efetivamente guardados, deverá disponibilizar interface de acesso aos dados presentes no banco de dados, não podendo ser uma “caixa-preta” de informações;
* esquematização: A existência de uma forma de relacionamento dos dados, presentes nas mais diversas tabelas, deve ser clara e precisa;
* controle de integridade: Um SGBD deve impedir a todo custo o comprometimento do banco de dados. Uma atividade de leitura, escrita, edição, deleção ou qualquer outra manipulação ou é concluída 100% dentro das regras ou é descartada totalmente. Não existe manipulação de dados incompleta ou quase completa;
* backups: O SGBD deve ser autônomo e conseguir recuperar-se de falhas de software e hardware sem a intervenção do pessoal técnico ou ser minimamente de-pendente deste pessoal.

### 3. Dicionários de dados

Agora que você já tem conhecimento sobre as articulações dos bancos de dados e seus sistemas de gerenciamento, como podemos padronizar as informações? Isso é importante para garantir a integridade dos dados e também sua consistência, fatores importantes e totalmente determinantes para a saúde do banco de dados.

O dicionário de dados vem para realizar esta padronização básica e extremamente importante, pois, à medida que o banco de dados cresce e mais desenvolvedores são escalados para trabalhar no projeto, mais fora de padrão pode ficar o banco de dados como um todo.

De exemplo inicial, imagine que o campo celular, por definição do escopo inicial do projeto, deverá ser do tipo varchar e comportar a máscara “(DDD) xxxxx-xxxx”.

Bem, tomando por base essa definição de escopo inicial, foi criada a tabela “clientes”, na qual um dos atributos é o campo “celular”, que obedece a essas regras. Todo o sistema da aplicação foi construído e finalizado, a implantação no cliente final realizada e, por isso, o banco de dados também foi finalizado com sucesso e aderente às regras.

Então, surge uma necessidade de upgrade de funcionalidade exigida pelo cliente, o qual deseja que, no cadastro de fornecedores, coloque-se um campo de celular de contato com esses fornecedores. Como passou muito tempo desde o início do projeto, esqueceu-se que aquela regra do campo de celular foi elaborada e é feito este segundo campo sem a opção de DDD ou, ainda, em vez de utilizar o separador “-” como pede a regra, foi utilizado o separador “.”.

Veja que o banco de dados começa a ficar confuso e sem padrão, fazendo com que a função de chamada de cadastro apresente erro ou, ainda, que precise ser feita de forma diferente para quando chamar o cadastro de cliente em relação a quando chamar o cadastro de fornecedor.

Vamos além? Imagine que exista mais uma tabela que contenha o campo celular: a tabela de cadastro de colaboradores. Ela pode ter sido desenvolvida por outro programador e, assim, recebeu outra validação de tipo e máscara. Extrapolando esse exemplo, pense em um sistema de aplicação gigante, como um sistema bancário. Como garantir que todos os desenvolvedores respeitem as mesmas regras de escopo de campos durante todo o projeto?!

A resposta para essas situações é o dicionário de dados!

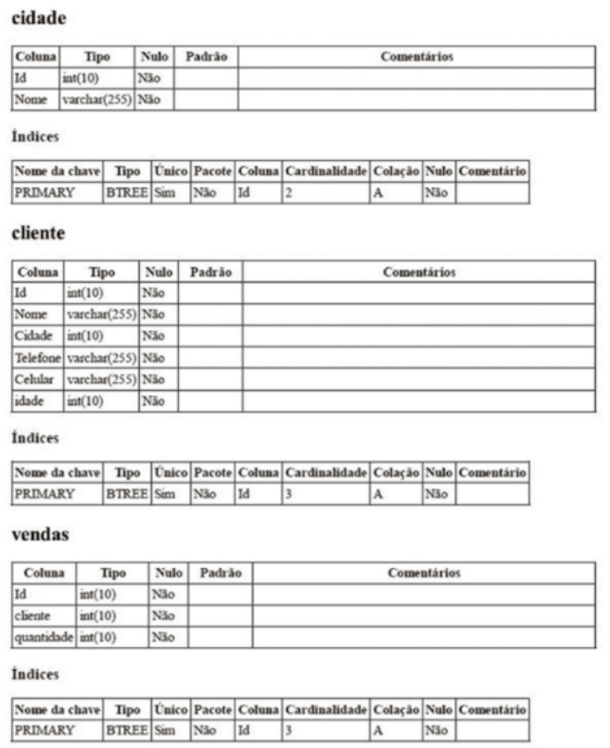
No dicionário de dados, também estará a relação de usuários e permissões, a estrutura geral e a alocação de espaço. Assim, definindo a regra do número de celular no dicionário de dados, todas as tabelas do sistema devem respeitar o dicionário de dados, criando um padrão fácil de ser entendido e respeitado por todos que manipulam o sistema, incluindo, aqui, desenvolvedores diferentes.

O dicionário de dados deve ter regras e valores para cada campo das tabelas, de modo que é natural que ocorram situações de opcionalidade de um caractere, por exemplo.

Para sanar essa situação, você verá, no Quadro, um exemplo de tabela de símbolo utilizada em um dicionário de dados qualquer:



Observe, na Figura, um exemplo simples de um dicionário de dados para a construção das tabelas cidade, cliente e vendas.



### 4. Os diferentes Sistemas de Gerenciamento de Bancos de Dados

Cada SGBD, seja ele MySQL, MariaDB, Oracle, Microsoft SQL Server ou PostgreSQL, garante a base de administração e manipulação de dados estudada e, além disso, apresenta características que o tornam ímpar.

A escolha pelo melhor SGBD se dará em virtude de qual é a aplicação a ser desenvolvida, o fluxo de acesso, o tipo de dados a serem guardados e a necessidade de profissionais no mercado.

Caso sua aplicação seja web, com acessos moderados e tamanho de pequeno a médio, poderá optar pelo MySQL, MariaDB ou, ainda, pelo PostgreSQL. Se a necessidade for a de suportar uma aplicação grande, com vários acessos simultâneos, cruzamento de dados intenso e grande granularidade de perfis, prefira o Oracle ou o Microsoft SQL Server.

Por fim, se a necessidade de performance estiver acima de qualquer outra frente, será necessária a utilização de bancos de dados NoSQL, como o MongoDB, por exemplo.