SISTEMA GERENCIADOR DE BANCO DE DADOS

AULA 1

Prof. Leonel da Rocha

CONVERSA INICIAL

Com o avanço da tecnologia da informação, os dados se tornaram um bem valioso a todas as organizações. Nosso objetivo neste estudo é aprender sobre os Sistemas Gerenciadores de Banco de Dados (SGBD), que são *softwares* especialistas na administração de uma base de dados e permitem manipular e extrair informações.

Nesta etapa, estudaremos a arquitetura de um SGBD, seus aspectos gerenciais e os tipos de banco de dados que podemos manipular, além dos principais sistemas encontrados no mercado. Veremos como configurar alguns parâmetros na criação de um banco de dados, a criação de usuários. As operações de administração de um banco de dados são realizadas por um profissional especialista que também vamos conhecer, acompanhando suas atribuições. Bons estudos.

TEMA 1 – ARQUITETURA DE UM SGBD E SEUS ASPECTOS OPERACIONAIS

Um Sistema Gerenciador de Banco de Dados, popularmente chamado pela sigla SGBD, é um *software* que gerencia a estrutura de um banco de dados. Ele favorece o controle do acesso dos usuários por meio de permissões, gerencia cópias dos dados, sua disponibilidade e distribuição. Também controla a integridade da base de dados, permitindo que os dados armazenados reflitam da maneira mais fiel possível a realidade.

Figura 1 – SGBD como ferramenta de gerenciamento de um banco de dados

Créditos: FGC / Shutterstock.

Destacamos cinco funções de um SGBD que agilizam a administração e o controle de acesso aos dados:

- gerenciar acesso compartilhado aos dados;
- atribuir permissões de acesso aos usuários;
- realizar cópias do banco de dados;
- adicionar regras e padrões nas estruturas de armazenamento;
- permitir a recuperação dos dados armazenados.

Para entender como esse sistema opera, é importante conhecer o funcionamento de seus componentes que são a estrutura lógica básica para que gerencie um banco de dados. Vamos ver o que os componentes fazem para o funcionamento de toda a estrutura:

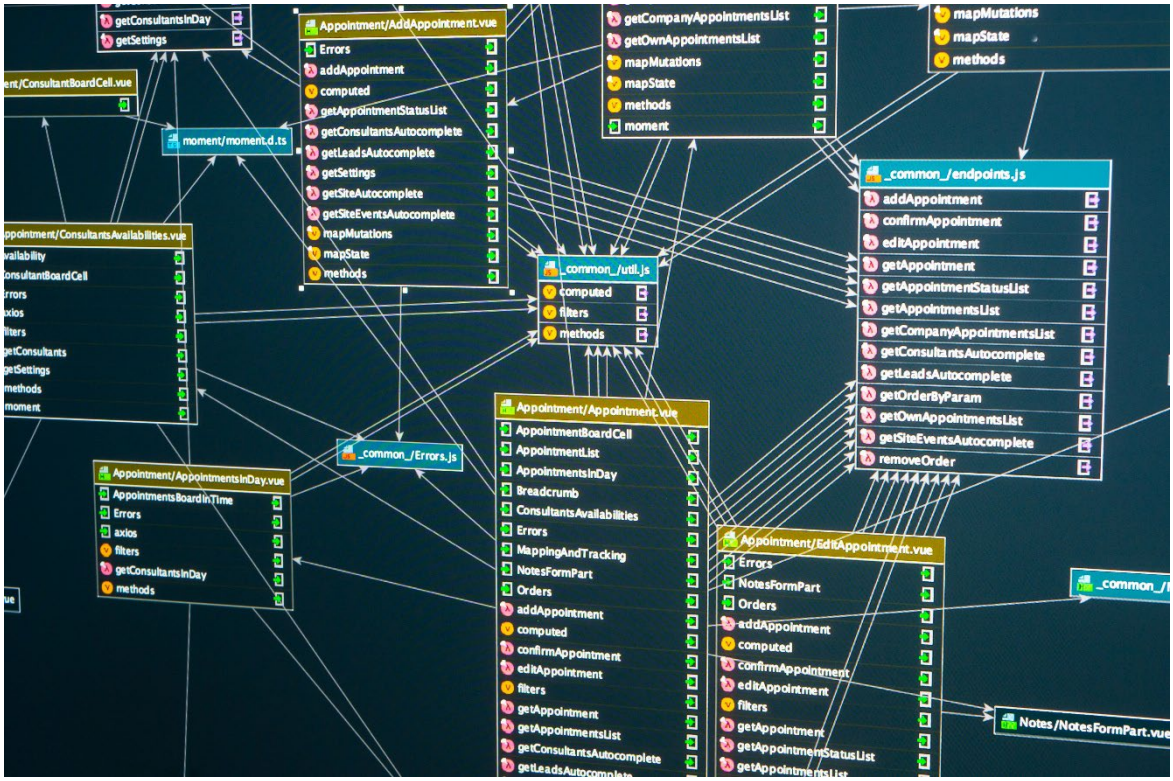
- Pré-compilador DML: permite ao SGBD identificar a sintaxe do *Data Manipulation Language* (DML ou linguagem de manipulação de dados), que faz as chamadas proceduralmente;
- Compilador DML: traduz a DML para uma linguagem de baixo nível, realizando as operações de manipulação dos dados;
- Interpretador de *Data Definition Language* (DDL ou linguagem de definição de dados): apesar do nome, não interage com os dados, e sim com os objetos do banco, ou seja, gerencia esses objetos;
- Gerenciador de permissões *Data Control Language* (DCL ou linguagem de controle de dados): atribui permissões de acesso aos objetos do SGBD;
- Gerenciador de integridade: permite ao SGBD controlar a integridade dos dados quando uma operação é realizada, não permitindo que fiquem inconsistentes;
- Para controlar as transições que acontecem no SGBD, temos o gerenciador de transição, que controla o acesso simultâneo dos usuários, assim como a integridade das operações;
- Gerenciador de arquivos: controla o espaço de disco do banco de dados e a sua estrutura de arquivos;
- Gerenciador de *buffer*: que controla a memória em cache e gerencia a transferência de dados da estrutura principal para a secundária.

Os *Schemas* (ou esquemas) são estruturas importantes em um SGBD, lembrando que têm conceitos distintos entre os fabricantes, havendo diferenças em suas implementações e atribuições. Conceitualmente, é a estrutura-base que

representa uma visualização geral do banco de dados, estabelecendo como os dados são e sua interligação, além de permitir visualizar as regras implementadas em cada campo.

O esquema físico ou interno é definido em um nível mais baixo e permite visualizar informações referentes ao *hardware*, e o esquema lógico ou conceitual é definido em nível abstrato e permite alterar os dados. Já o esquema de visão possibilita visualizar os dados conforme as permissões atribuídas ao usuário.

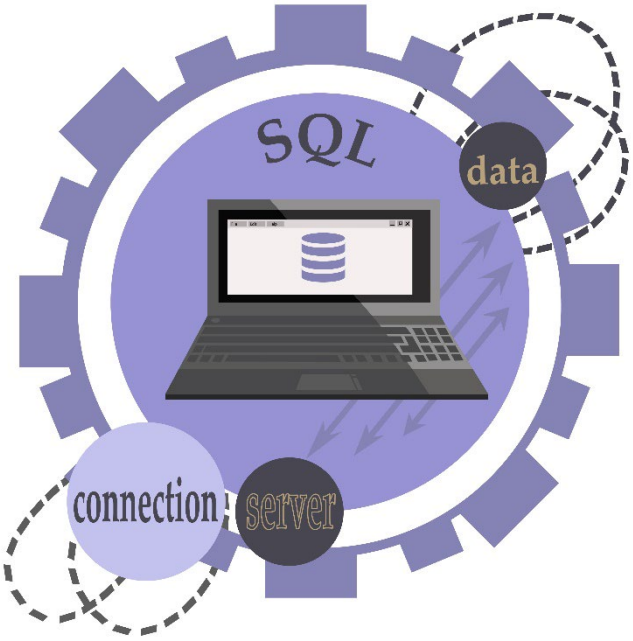
Figura 2 – Dados



Créditos: Yurich / Shutterstock.

As instâncias representam um conjunto de informações definidas por um esquema. Em um SGBD, são criadas quando acontece uma operação de inserção, alteração ou exclusão de dados em um conjunto de objetos.

Figura 3 – Informações



Créditos: Bobicova Valeria / Shutterstock.

Outro ponto muito importante suportado por um SGBD é a linguagem de manipulação de dados, já citada anteriormente. Mostraremos os principais comandos dos três principais tipos, que são:

- *Data Manipulation Language* (DML ou linguagem de manipulação de dados): usada para inserir, modificar ou excluir dados em uma tabela. Seus principais comandos são: *insert*, *delete*, *update* e *select*.
- *Data Definition Language* (DDL ou linguagem de definição de dados): utilizada para adicionar ou modificar os objetos em um banco de dados. Os principais comandos são: *CREATE*, *ALTER* e *DROP*.
- *Data Control Language* (DCL ou linguagem de controle de dados): usada com objetivo de atribuir ou revogar direitos de acesso para os usuários se conectarem ao um banco de dados e acessarem os seus objetos. Seus principais comandos são: *CREATE USER*, *GRANT* e *REVOKE*.

Um SGBD deve possuir algumas funções que garantam seu funcionamento e principalmente o gerenciamento dos dados. Vamos ver quais são elas. A primeira a ser citada é a da segurança, que deve implementar regras customizadas para as tabelas e seus campos e contempla a realização de cópias de segurança dos dados e seu retorno.

A função de controle de acesso permite gerenciar as atividades dos usuários no SGBD, atribuindo permissões somente de leitura, por exemplo, a determinado grupo de usuários e permissões de administradores para um número reduzido e especializado de usuários, que geralmente fazem parte do departamento de tecnologia da informação.

A funcionalidade de controle de redundâncias monitora a aplicação de regras definidas pelos desenvolvedores ao banco de dados, garantindo que os dados, que geralmente são manipulados por vários usuários, se mantenham íntegros sem distorcer o que representam.

Figura 4 – Data

Créditos: daddy.icon / Shutterstock.

A utilização de um SGBD se mostra importante e fundamental para as aplicações, permitindo o gerenciamento do banco de dados de modo flexível, com fácil acesso aos dados e sua visualização. Apresenta como vantagem a possibilidade de gerenciar a segurança de acesso, disponibilizando ferramentas que criam e atribuem acesso específico para cada usuário, implementando mais segurança para as informações. Além disso, permite minimizar a inconsistência dos dados mediante mecanismos de controle que implementam a consistência nas transações de inserção, alteração e exclusão de dados. Outro benefício é a facilidade de integração de dados por gerenciar diretamente os objetos de um banco de dados, podendo ainda disponibilizar tal gerenciamento por outros meios.

Figura 5 – Gerenciamento

Créditos: eamesBot / Shutterstock.

Como em todas as áreas, não existem apenas flores, e os SGBD trazem algumas desvantagens que devem ser analisadas para definir sua utilização ou não. Vamos ver algumas e avaliar se são impeditivas ou não. Alguns sistemas possuem licença de uso e apresentam alto custo para ser adquiridos, além do fato de que, dependendo do tamanho da organização, são necessários equipamentos de alta performance que também podem exigir um investimento muito elevado. Apresentam uma complexidade operacional relativamente alta, necessitando de técnicos especializados que, por sua vez, demandam um valor de contratação elevado, inviabilizando sua utilização. A performance, que é a capacidade de atender às solicitações dos usuários, está ligada diretamente aos recursos relacionados ao *hardware* disponível, podendo comprometer a utilização de um SGBD, razão por que deve ser estudado com muito cuidado para não inviabilizar todo o processo.

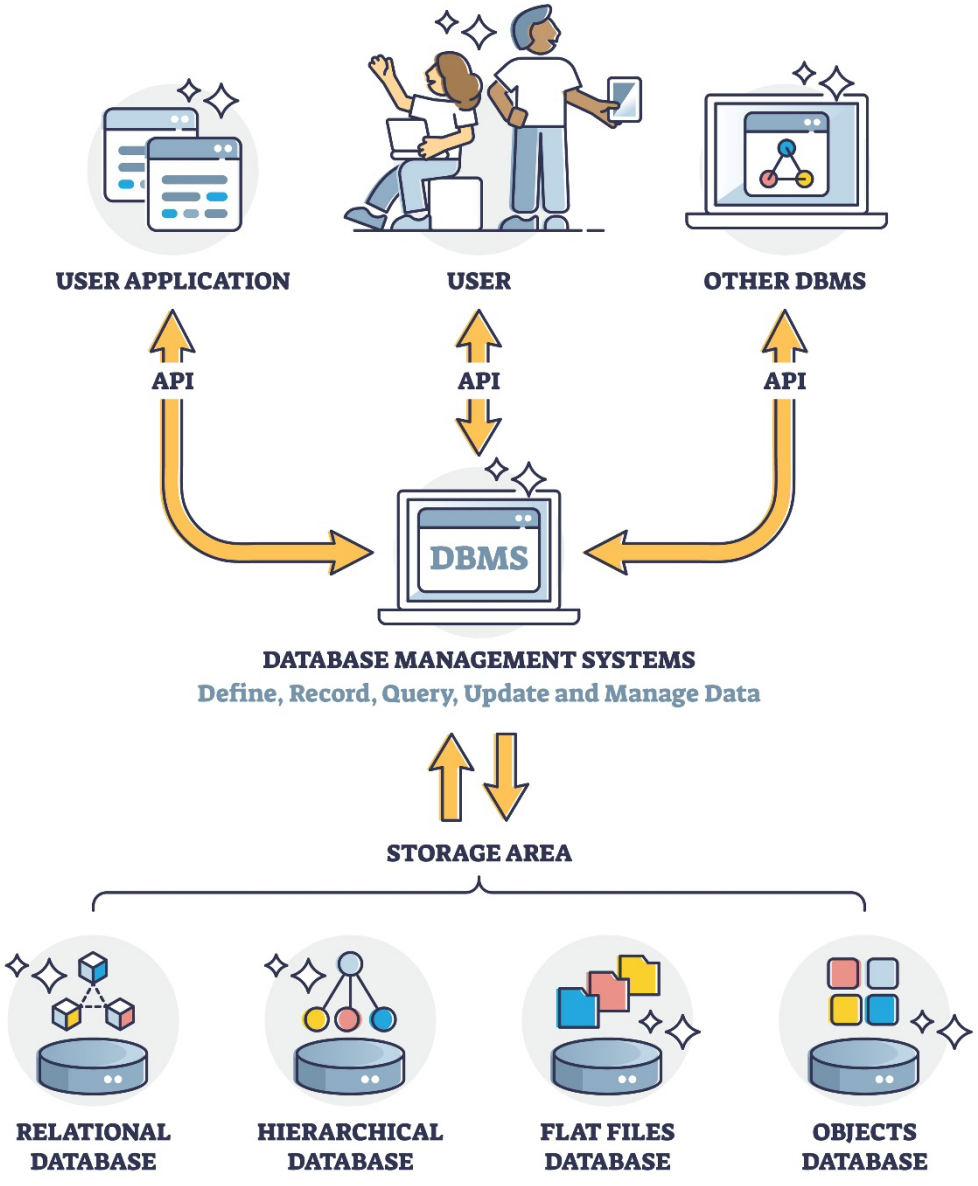
Outra funcionalidade importante em um SGBD é o controle de concorrência, procedimento de segurança cuja finalidade é evitar mudanças simultâneas em um mesmo objeto por mais do que um usuário ao mesmo tempo, o que pode ocasionar problemas com os dados manipulados. Esse controle é feito para que as operações de manipulações sejam concluídas pelo processo que acessou o objeto por primeiro, liberando o acesso para o próximo usuário quando for finalizado o anterior. Vale lembrar que essa concorrência existe apenas quando os usuários tentam manipular os dados de uma tabela, e não para a leitura deles.

Um SGBD possui uma arquitetura que têm como objetivo separar as

aplicações dos dados e que é dividida nos esquemas assim descritos:

- Esquema ou nível interno: utiliza um modelo de dados que mostra a estrutura de armazenamento físico do banco de dados, os detalhes dos dados armazenados e os caminhos de acesso;
- Esquema ou nível conceitual: descreve toda a estrutura do banco de dados sem mostrar detalhes dos dados armazenados;
- Esquema de visão ou nível externo: possibilita ao administrador visualizar quais usuários têm acesso ao banco de dados.

Figura 6 – Aplicações



Créditos: VectorMine / Shutterstock.

Os SGBD podem ser classificados quanto ao número de usuários, localização e ambiente onde estão inseridos. A seguir, apresentamos uma descrição desses tipos de classificação:

- Usuário: monousuários (utilizados em computadores de menor porte e acessados localmente), e multiusuários (instalados em máquinas maiores, chamadas de servidores, e podem ser acessados por vários usuários);
- Ambiente: homogêneo (o ambiente é constituído por apenas um SGBD) e heterogêneo (o ambiente é composto por diferentes SGBD);
- Localização: localizado (todos os dados estão em um único disco) e distribuído (os dados estão em vários servidores).

1.1 Estrutura de memórias no MySQL

O MySQL implementa dois tipos de gerenciamento de memória: o global e por sessão. Conheça a descrição deles a seguir:

- Global (instância): a memória é compartilhada com todas as sessões do MySQL; sua alocação é feita quando o servidor é *startado*, e liberada apenas quando é desligado;
- Sessão: a memória é alocada dinamicamente por sessão, podendo ser liberada após sua utilização ou quando não for necessária.

A alocação da memória no MYSQL pode ser realizada de três maneiras: estática, automática e dinâmica. A estática acontece quando usamos variáveis globais (alocadas fora de funções) ou variáveis locais (internas a uma função) que são alocadas com a utilização do modificador "static". Esse tipo de variável mantém seu valor indefinidamente, a não ser que seja explicitamente modificada.

Na alocação automática, por *default*, as variáveis definidas dentro de uma função (variáveis locais e parâmetros) serão alocadas automaticamente na pilha de execução do programa (*stack*) em cada chamada da função, sendo descartadas quando a função for encerrada.

Na alocação dinâmica, o sistema solicita de forma explícita áreas de memória ao sistema operacional, utilizando-as até o encerramento. As requisições de memória dinâmica são alocadas na área de memória denominada *heap*.

Em relação à utilização da memória no servidor, quando ela é compartilhada com todas as sessões, que é a conexão que cada usuário faz com o banco de dados, temos um exemplo da utilização do *query cache*. Ele salva os resultados

dos SELECTs executados na memória, e isso faz com que o tempo de resposta de consultas que são repetidas muitas vezes seja muito mais rápido, devido ao fato de o MySQL não necessitar buscar os dados solicitados no disco de novo, pois eles estão armazenados no *cache*.

Por sua vez, quando é feita alguma alteração dos dados na tabela de consulta, o *hash* desse *cache* será quebrado; dessa maneira, evita-se que o retorno dos dados seja incorreto, pois estaria desatualizado.

O query_cache oferece três níveis de configuração que podem ser encontrados e alterados no my.cnf ou my.ini:

- Para configurar o query cache como desligado, setar o query_cache_type como 0: # query_cache_type = 0
- Para configurar o query cache como ligado para todas as consultas, setar query_cache_type como 1: # query_cache_type = 1
- Para configurar o query cache como ligado sob demanda, setar query_cache_type como 2: # query_cache_type = 2

Quando for utilizar a opção em demanda, é preciso definir na query que for utilizar o *cache* com o SQL_CACHE: SELECT SQL_CACHE * FROM <Tabela>...

No MySQL existe um mecanismo de armazenamento de memória que a compartilha com todas as sessões, sendo possível por utilizar os *Storages Engines*. Para esse exemplo, podemos citar a utilização da variável *buffer pool* do *innoDB*, que mantém uma área de armazenamento na memória para trabalhar com os dados e índices, permitindo um ganho de performance em todas as atividades.

A partir dessa variável, é definido um valor exclusivo na memória para as operações do *innoDB*. Assim, toda vez que o MySQL for iniciado, o valor será reservado na memória RAM do ambiente. Aqui vale uma observação sobre o valor de memória a ser utilizado para que não prejudique outros aplicativos e o próprio servidor onde está instalado o MySQL. Para um servidor dedicado ao MySQL, que significa que apenas o MySQL roda nessa máquina, é recomendada a utilização de 50% a 80% da memória.

Figura 7 – MySQL

Créditos: monticello / Shutterstock.

TEMA 2 – TIPOS DE BANCO DE DADOS E PRINCIPAIS SGBD

Na tecnologia da informação, quando o assunto é dados, é automático falarmos de sistema gerenciadores de bancos de dados, pois são eles que administram os vários tipos de banco de dados e permitem que as informações sejam extraídas de uma forma que os sistemas possam intuitivamente atender às necessidades dos usuários, tanto em sistemas administrativos quanto em *sites* de busca ou de compras, entretenimento e várias outras aplicações.

Para dar conta de um número muito grande de aplicações, há vários SGBD no mercado, e eles administram diversos tipos de de banco de dados, cada um voltado a demandas específicas de armazenamento de dados em suas variadas formas (informações pessoais e de produtos, imagens, sons), além de estar armazenados em vários locais e se interligarem utilizando tecnologias diferentes.

Vamos conhecer os diversos tipos de banco de dados disponíveis no mercado. Vale lembrar que essa lista pode variar conforme a necessidade e tamanho deles: desde pequenos bancos pessoais, nos quais se utiliza uma planilha de cálculo para fazer o gerenciamento de um número pequeno de informações, até megabancos de dados, em que são empregados equipamentos e diversas tecnologias para gerenciar banco de dados gigantescos que podem ser acessados por um número muito grande de usuários espalhados por todos os lugares atendidos por algum tipo de comunicação. Nesse caso, podemos citar um

site de busca como o Google, que está presente em todos os dispositivos que tenham acesso à internet e que armazena enorme quantidade de informações que são acessadas diariamente por um número assustador de usuários.

Então vamos acompanhar a descrição desses tipos de banco de dados, tarefa importante para construir o conhecimento sobre essa tecnologia.

- **Banco de dados relacional**: são os mais comuns e utilizados no mercado porque possuem alta confiabilidade de informações e facilidade no armazenamento. Seu funcionamento é feito mediante o armazenamento de dados em colunas e linhas, o que o torna bem intuitivo, já que é muito semelhante a várias aplicações manuais controladas por meio de uma relação de coisas, tal como o contato de pessoas, em que as colunas representam atributos (nome, telefone, endereço, *e-mail*), e as linhas, os dados respectivos (João, 99999999, Rua xv, joao@gmail.com). O Quadro 1 ilustra um exemplo de uma tabela com a relação dos contatos que representa a forma como um banco de dados relacional armazena os dados.

Quadro 1 – Exemplo de como um banco de dados relacional armazena informações

Nome	Telefone	Endereço	*E-mail*
João	99999999	rua XV	joao@gmail.com
Maria	88888888	Rua A	maria@uol.com
Pedro	98997866	Rua Z	pedro@bol.com
Ana	97689089	Rua D	ana@cit.com

Fonte: Rocha, 2023.

Existe outro fator muito importante para os bancos de dados relacionais: a linguagem de manipulação dos seus objetos e dados, que é a *Structured Query Language* (SQL). Comum a todos os SGBD de sucesso no mercado, permite uma utilização fácil dessa categoria de banco de dados, pois, além de ser de fácil aprendizado, também é simples de ser implementada, possibilitando uma interação muito grande com a base de dados, sua manutenção e manipulação.

- **Banco de dados não relacional**: popularmente chamados de banco de dados NoSQL, sua principal utilização se dá quando é necessário manipular dados não estruturados, como vídeos, imagens e gráficos, pois eles não podem ser dispostos em tabelas, como as descritas nos bancos relacionais.

Em relação à manipulação de dados, na linguagem dos bancos de dados relacionais utilizamos a SQL, e no caso dos bancos de dados não relacionais, a Not Only SQL (NoSQL), que significa “não apenas SQL”. Uma dica importante é conhecer e entender os diferentes tipos de banco de dados disponíveis no mercado para definir qual atenderá da melhor forma o projeto que está sendo desenvolvido.

Como o tipo de banco de dados não relacional, ou NoSQL, não é nosso principal objetivo neste estudo, descrevemos um pouco mais sobre essa estrutura para completar o conhecimento a seu respeito. O termo *NoSQL* foi utilizado pela primeira vez em 1998 por Carlo Strozzi quando falou acerca de um banco de dados não relacional de código aberto. Com a popularização do uso da Internet e das soluções digitais, o mercado viu a necessidade de descobrir cada vez mais formas inteligentes de gerenciar os bancos de dados visando atender de modo mais eficaz a demanda crescente pela informação.

Os principais bancos de dados NoSQL disponíveis são:

- **Redis**: é o banco de dados NoSQL de chave-valor mais famoso. Ele vincula um valor a uma chave em sua estrutura, o que facilita o armazenamento e a extração dos dados. Por essa característica, sua utilização é muito popular entre os desenvolvedores.

Figura 8 – Redis

Fonte: Redis, [s.d.].

- **Cassandra**: desenvolvido pelo *Facebook*, utiliza um banco de dados descentralizado, em que os dados são armazenados em diversos *datacenters*. É otimizado para cluster e fornece baixa latência nas atualizações.

Figura 9 – Cassandra

Créditos: monticello / Shutterstock.

- **MongoDB**: um banco de dados NoSQL de código aberto de alta performance. Portável em diferentes sistemas operacionais, tem como principal característica ser orientado a documentos, e armazena todas as informações relevantes, utiliza sistemas avançados de agrupamento e filtragem. Várias plataformas e linguagens possuem suporte ao MongoDB, entre elas Java, JavaScript, PHP, Python e Ruby.

Figura 10 – MongoDB

Créditos: Vitalii Vodolazskyi / Shutterstock.

- **Memcached**: tem como característica o armazenamento com chave-valor e usa um *cache* de memória distribuída. É utilizado com frequência para criar *sites* dinâmicos, pois agiliza a abertura das páginas e ainda minimiza as buscas de dados de fontes externas.
- **Neo4j**: é baseado em grafos, que são arestas que se relacionam aos nós. Trata-se de uma implementação de código aberto e pode ser utilizado para casos de mineração de dados e reconhecimento de padrões.
- **Amazon DynamoDB**: é um banco de dados NoSQL em nuvem, disponibilizado pela Amazon Web Service. Possui baixa latência, é rápido e flexível e muito utilizado para aplicações móveis, jogos e soluções com *Internet of Things* (IoT). Tem alto desempenho e escalabilidade automática, características imprescindíveis para negócios que necessitam crescer com eficiência.

Figura 11 – Amazon Web Service

Créditos: Michael Vi / Shutterstock.

- **Hbase**: o Apache HBase provê acesso aleatório e em tempo real aos seus dados no Hadoop. Criado para hospedar tabelas muito grandes, se tornou ótima opção para armazenar dados multiestruturados ou esparsos. Está presente em diferentes plataformas, como *LinkedIn*, *Facebook* e *Spotify*.

Figura 12 – Apache HBase

Fonte: Apache Hbase, [s.d.].

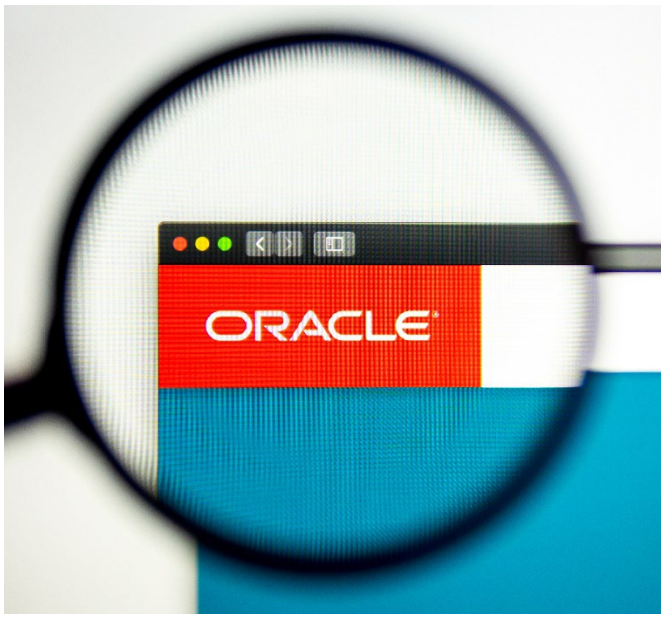
2.1 Principais SGBD relacionais no mercado

Em qualquer área, sempre é complicado falarmos em principais produtos ou coisas, pois podemos cometer erros ao apontar os mais usados ou os melhores. Tudo depende da situação abordada. Com SGBD não é diferente, mas mesmo assim vamos citar os quatro principais SGBD relacionais do mercado.

2.1.1 Oracle Database

O Oracle Database surgiu no final dos anos 1970 e, de acordo com a DB-Engines ([s.d.]), é o SGBD mais utilizado do mundo. Ele possui alta disponibilidade e tem como principais características: recuperação ágil; transparência de alterações que ocorrem no sistema, nos dados ou nas aplicações; e alta tolerância a falhas que permite ao processamento um valor baixo de interrupções; além de contar com medidas preventivas próprias.

Figura 13 – Oracle Database



Créditos: Anton Garin / Shutterstock.

2.1.2 MYSQL

O MySQL é um sistema de gerenciamento de banco de dados, *open source* e gratuito para uso de aplicações *open source*. Apresenta diversas funcionalidades e ampla gama de interfaces de usuários disponíveis para implementação. Possui alta performance e disponibilidade, além de um suporte transacional robusto. É compatível com outros bancos de dados, como Oracle e DB2, que é outro exemplo de SGBD relacional de propriedade da IBM.

Figura 14 – MySQL

Créditos: dennizn / Shutterstock.

2.1.3 SQL SERVER

O Microsoft SQL Server é um SGBD relacional desenvolvido pela Sybase em parceria com a Microsoft e lançado em 1994. Como vantagens possui maior facilidade na análise de dados devido a sua interface amigável, implementa uma flexibilidade na SQL, tem recursos de processamento de consulta inteligente, alta segurança e criptografia, armazenamento em nuvem e possibilita a emissão de relatórios com alta gestão.

Figura 15 – SQL Server

Créditos: monticello / Shutterstock.

2.1.4 POSTGRESQL

PostgreSQL, lançado em 1996, é um SGBD com projeto de código aberto, sendo essa uma de suas vantagens. É robusto por natureza, com alto desempenho e multitarefa. Apresenta vários recursos e é uma boa opção de escolha dentre os SGBD relacionais citados neste material.

TEMA 3 – CONFIGURAÇÕES UTILIZADAS NA CRIAÇÃO DE UM *DATABASE*

Para a criação de um banco de dados, ou *database* (DB), podemos configurar alguns itens, levando em consideração vários fatores, como espaço em disco, equipamento, número de usuários que o acessarão, segurança, disponibilidade e outros de acordo principalmente com a utilização e o ambiente em que ele estará exposto.

Para fazer a criação de um banco de dados no MySQL, é possível utilizar o comando CREATE DATABASE com a seguinte sintaxe:

```
CREATE DATABASE [IF NOT EXISTS] nome_banco_dados [CHARACTER SET
nome_charset]
[COLLATE nome_collation] [ENCRYPTION] [=] {'Y' | 'N'}
```

A primeira situação é atribuir um nome para o banco de dados após o comando CREATE DATABASE. O nome do banco de dados precisa ser único em uma instância do servidor MySQL. Se houver a tentativa de criar um banco de

dados com um nome já existente, o MySQL não permitirá, causando um erro.

Podemos utilizar a cláusula IF NOT EXISTS para criar um banco de dados se ele não existir na instância do MySQL que está sendo utilizada.

Depois, devemos especificar o conjunto de caracteres e a coleção de caracteres para o novo banco de dados. Se as cláusulas CHARACTER SET e COLLATE não forem informadas, o MySQL utilizará valores-padra para o novo banco de dados.

A opção ENCRYPTION definirá a criptografia-padrão do banco de dados, que será herdada pelas tabelas. Possui dois valores possíveis: 'Y' para criptografia habilitada e 'N' para criptografia desabilitada. Vale lembrar que as opções entre colchetes ([opção]) são facultativas no comando mostrado nesta seção. Eles não precisam estar obrigatoriamente destacados, e quando isso acontecer, valores-padrão serão determinados pelo MySQL.

TEMA 4 – CRIAÇÃO DE USUÁRIOS

Para fazer a utilização de um SGBD, é necessário se conectar a ele por meio de usuário e senha, e com o MySQL não é diferente. Para isso, vamos ver agora como criar um usuário para ele se conectar e ter acesso aos objetos e aos dados. É possível dar controle total quando criamos um usuário, mas isso não deve ser feito, pois a segurança de acesso aos dados é uma condição bastante importante dentro das organizações e precisa ser levada muito a sério.

Vamos ver como criar um usuário utilizando a linha de comandos do MySQL:

```
CREATE USER 'nome_usuário'@'localhost' IDENTIFIED BY 'senha';
```

Vale lembrar que nome_usuário é o nome que será dado à nova conta de usuário e que a seção IDENTIFIED BY ‘senha’ define uma senha de acesso para o novo usuário. Os valores devem ser substituídos sempre que for criado um usuário, porém só os valores que estão dentro das aspas.

Para garantir os privilégios de acesso ao banco de dados a um novo usuário, o seguinte comando deve ser executado:

```
GRANT ALL PRIVILEGES ON * . * TO 'nome_usuario'@'localhost';
```

Para que as mudanças sejam aplicadas, executar um *flush* dos privilégios com o seguinte comando:

```
FLUSH PRIVILEGES;
```

TEMA 5 – ADMINISTRADOR DE BANCO DE DADOS

O *DataBase Administrator* (DBA ou administrador de banco de dados) é um profissional da área de tecnologia da informação com responsabilidade para instalação de um SGBD, criação, monitoramento, manutenções e análise das estruturas e objetos de um banco de dados.

O SGBD e o banco de dados devem ser geridos de forma constante pelo DBA, focando para que não haja sobrecargas e paralisações do sistema e que os dados sejam incluídos e armazenados de forma correta nos servidores. Outras funções relevantes incluem analisar o espaço em disco, buscar melhorias e atualizações para os sistemas, realizar *backups* e implementar segurança aos dados. Aumentar a eficiência das consultas para extração das informações também constitui tarefa importantíssima.

Como o desenvolvimento de sistemas e aplicações está cada vez mais intenso, o DBA encontra muitas oportunidades no mercado de trabalho, além de uma boa remuneração. De acordo com dados do Cadastro Geral de Empregados e Desempregados (Caged), do Ministério do Trabalho, o salário médio de um DBA sênior no início de 2023 pode chegar a R$ 10.800,00 (Brasil, 2023).

Esse profissional pode estar presente na área de tecnologia de informação de qualquer empresa, seja de saúde, educação, marketing, setor público em geral, indústrias, bibliotecas etc. Ele pode ser contratado diretamente por esses locais ou prestar serviço por meio de uma empresa de tecnologia da informação. Além disso, pode atuar de modo independente como consultor para assuntos voltados à área de banco de dados, inteligência de negócios, ciência de dados, entre outras que utilizam os SGBD como fonte para os dados.

FINALIZANDO

Nesta etapa, vimos a arquitetura de um SGBD e seus aspectos gerenciais, os tipos de banco de dados e o que eles manipulam, assim como os principais SGBD encontrados no mercado, sejam os relacionais, sejam os NoSQL, que apresentam um crescimento grande devido a sua especialização em dados não estruturados, como imagens, vídeos e sons.

Tivemos a oportunidade também de conhecer como se configuram alguns parâmetros na criação de um banco de dados e a criação de usuários. Por fim, mas não menos importante, verificamos as principais características de

administração de um banco de dados realizadas por um profissional especialista. Seu papel é destacado na criação dos bancos de dados, no gerenciamento dos dados e na extração das informações, podendo ser uma alternativa de atuação profissional muito interessante dentro da área de tecnologia da informação.

REFERÊNCIAS

APACHE HBASE. Disponível em: <https://hbase.apache.org/downloads.html>. Acesso em: 10 maio 2023.

BARRIE, H. **Dominando Firebird**: uma referência para desenvolvedores de bancos de dados. Rio de Janeiro: Ciência Moderna, 2006.

BRASIL. Ministério do Trabalho. Programa de Disseminação das Estatísticas de Trabalho. **Novo Caged**, 2023. Disponível em: <http://pdet.mte.gov.br/novo-caged>. Acesso em: 10 maio 2023.

CARLOS, S. **Comparativo de desempenho de bancos de dados de código aberto**. Recife: Universidade Federal de Pernambuco, 2011.

CARNEIRO, A. P.; MOREIRA, J. L.; FREITAS, A. L. C. **Tuning**: Técnicas de otimização de bancos de dados – um estudo comparativo: MySQL e PostegreSQL. 10 f. Trabalho de Graduação (Ciências Computacionais) – Universidade Federal do Rio Grande, Rio Grande, 2011.

DB-Engines. Disponível em: <https://db-engines.com/en/>. Acesso em: 10 maio 2023.

KORTH, H.; SILBERSCHATZ, A.; SUDARSHAM, S. **Sistemas de bancos de dados**. 5. ed. São Paulo: Makron Books, 2006.

LEITE, M. **Acessando bancos de dados com ferramentas RAD**. Rio de Janeiro: Brasport, 2007.

REDIS. Disponível em: <https://redis.io/>. Acesso em: 10 maio 2023.