Nivelamento: Banco de Dados

Qualifica SP, aula 5, nivelamento de banco de dados

O que é um banco de dados?

Banco de dados é uma coleção de dados estruturados ou informações organizadas, geralmente armazenadas em um computador.
Os bancos de dados são gerenciados por um SGBD, que permitem a criação, manutenção e utilização dos dados armazenados.

Importância de bancos de dados

De forma simples, os bancos de dados permitem o armazenamento seguro e recuperação rápida de grandes volumes de dados, o que facilita a automação de processos e tomada de decisões.
Virtualmente todas as áreas utilizam algum (ou vários) banco(s) de dados, desde empresas de saúde até o ramo da educação.

Conceitos básicos: definições e terminologias

- **Dados**: Informações brutas, que podem ser utilizadas.
- **Informação**: Dados processados, de maneira que tenham significado
- **Tabela**: Estrutura de linhas e colunas que armazena os dados
- **Registro**: Conjunto de dados relacionados em uma tabela
- **Campo**: Unidade individual de dados em um registro

Características de um Banco de Dados

Caracteristica	Definição
Persistência	O armazenamento é permanente
Confiabilidade	A integridade dos dados é mantida
Segurança	Controle de acesso e perda de dados
Eficiência	Recuperação e manipulação rápida de dados

Tipos de Bancos de dados

Relacionais

Nosso foco de estudo são os bancos relacionais, como PostgreSQL, MySQL, e Oracle Database.
Eles levam esse nome por serem construídos em cima de uma estrutura de tabelas que se relacionam.
Também são os responsáveis pela utilização e poopularização do SQL, ou Structured Query Language, que é utilizada para consultar (recuperar) dados no banco. As operações de consulta por sua vez levam o nome de Data Manipulation Language (DML).

Não Relacionais

Existem diversos tipos de bancos não relacionais, chamados de NoSQL, entre eles, nós temos:

Documentos

Bancos baseados em documentos tem uma estrutura mais flexivel, e consequentemente, mais simples de escalar. Porém eles garantem apenas consistência eventual e podem ser menos eficientes em consultas mais complexas.

Exemplo: MongoDB

Grafos

Bancos de dados de grafos visam facilitar a implementação de relações complexas entre dados, representados como grafos (nós e arestas). São ideais para aplicações que envolvem conexões e relaçoes intricadas. São muito utilizados em redes sociais para representar conexões.

Exemplo: Neo4j

NewSQL

Esses bancos foram criados para combinar a consistência e as funcionalidades de bancos Relacionais com a escalabilidade de bancos NoSQL. Um exemplo de uso é a Comcast que utiliza o CockroachDB para gerenciar os dados de seus usuários.

Exemplo: Neo4j

Modelagem de Dados

A modelagem de dados é o processo de criar uma representação visual de dados e relacionamentos de um sistema. Por meio disso, podemos melhor estruturar nossos bancos com lógica e eficiência.

Modelagem Conceitual

- Fase que não depende do SGBD que será utilizado
- Utiliza diagramas ER (Entidade-Relacionamento)

Modelagem Lógica

- Refinamento da fase conceitual
- Adiciona dados como chaves
- Normaliza dados

Modelagem Física

- Traduz as modelagens anteriores para um SGBD
- Define tipos de dados, índices e outras considerações

Ferramentas

Para criar esses diagramas, podemos utilizar ferramentas como:

- Miro/Lucidchart
- Microsoft Visio
- ER/Studio

Estrutura de um Banco Relacional

Uma tabela é composta de linhas e colunas. As linhas representam um registro, que contém dados identificados pela coluna.

Os registros em um banco podem ser dos mais váriados tipos, baseados nos tipos primitivos estudados anteriormente, e tipos personalizados, como Data ou Blob.

Além disso, podem ser criados também chaves, índices, visões, procedimentos e gatilhos.

Transações

Uma transação nada mais é do que uma série de instruções e comandos que definem uma unidade lógica de trabalho.

A vantagem de se utilizar transações é que uma vez que uma é iniciada, caso algo dentro dela falhe, todos os dados são restaurados para seus dados iniciais, evitando assim uma falha em cascata.

ACID

Letra	Significado	Descrição
A	Atomicidade	Ou tudo tem sucesso, ou tudo falha
C	Consistência	Após a execução, a integridade do banco é mantida
I	Isolamento	Uma transação não tem conhecimento ou visibilidade de outra
D	Durabilidade	Uma vez confimada, seus dados são persistidos de forma permanente

Relacionamentos

- Um para um (1:1)
- Um para muitos (1:N)
- Muitos para muitos (N:N)

Cardinalidade

Define o número de ocorrências de um relacionamento, definindo quantos de um objeto X podeem se relacionar com um ou mais objetos do tipo Y.

- (0,1) -> Uma pessoa pode possuir 0 ou 1 passaporte.
- (1,1) -> Uma pessoa pode ter apenas 1 CPF
- (0,*) -> Uma pessoa pode ter 0 ou mais empregos
- (1,*) -> Uma pessoa deve possuir ao menos um documento válido a todo momento, podendo possuir N.

Normalização

Normalização de banco de dados é o processo de garantir que um banco de dados faça sentido e não se contradiga.
Tabelas normalizadas são mais fáceis de utilizar, entender e estender, além de serem protegidas contra anomalias de inserção, atualização e deleção de registos.

Primeira forma normal

A primeira forma normal, deve garantir que

1. A ordem das linhas na tabela não deve transmitir significado
2. As colunas devem ser tipadas, evitando tipos diferentes na mesma coluna
3. Toda tabela deve ter uma chave primaria
4. Evita repetição de grupos em uma única linha

Segunda forma normal

A segunda forma normal diz apenas que cada atributo (coluna) não chave primaria, deve depender inteiramente na chave primaria.

Supondo que nós tenhamos a seguinte tabela:

Player_ID	*Item_Type*	Item_Quantity	Player_Rating
jdog21	Amulets	2	Intermediate
jdog21	Rings	4	Intermediate
gila19	Coins	18	Beginner
trev73	Shields	3	Advanced
trev73	Arrows	5	Advanced
trev73	Coins	30	Advanced
trev73	Rings	7	Advanced

Dado que a chave primaria é:
{Player_Id, Item_Type}

Temos que:

Relacionamento	É valido?
{Player_Id, Item_Type} -> {Item_Quantity}	Sim
{Player_Id} -> {Player_Rating}	Não

A solução para normalizar essa situação é quebrar essa estrutura em diferentes tabelas. Uma de jogador com Id, e Rating, e a original será "Inventory", mantendo sua estrutura, apenas com a remoção de Player_Rating

Terceira forma normal

A terceira forma normal, em resumo, diz que:

> Todo atributo não chave de uma tabela deve depender da chave, por inteiro, e de nada além da chave.

Existe no entanto uma variação chamada Boyce-Codd Normal Form, que retira o "não chave da frase acima, a tornando na seguinte:

> Todo atributo de uma tabela deve depender da chave, por inteiro, e de nada além da chave.

Quarta forma normal

Normalmente, uma tabela na 3NF já está normalizada, mas existem situações onde precisamos de mais.

A **quarta forma normal**, diz que: Dependências com multiplos valores devem ser baseadas na chave.

Em outras palavras, se temos estilos de casas e tamanhos de casas, esses valores devem ficar em tabelas diferentes que se relacionam.

Quinta forma normal

A **quinta forma normal**, diz que: não deve ser possível descrever a tabela como sendo o resultado lógico de unir diferentes tabelas.