Conceitos Primordiais de Banco de Dados

Banco de Dados

Um banco de dados é uma coleção de dados persistentes e organizada usada pelos sitemas de informação. Através deles podemos salvar dados, resgatar os dados e também manipulá-los.
Os bancos de dados relacionais são formados por tabelas e colunas.
Existem diversos tipos de Bancos de Dados.
"O melhor banco de dados para uma organização específica depende de como a organização pretende usar os dados" (Oracle)

> **Bancos de dados relacionais**
> Bancos de dados relacionais se tornaram dominantes na década de 1980. Os itens em um banco de dados relacional são organizados como um conjunto de tabelas com colunas e linhas. A tecnologia de banco de dados relacional fornece a maneira mais eficiente e flexível de acessar informações estruturadas.

> **Bancos de dados orientados a objetos**
> As informações em um banco de dados orientado a objetos são representadas na forma de objetos, como na programação orientada a objetos.

> **Bancos de dados distribuídos**
> Um banco de dados distribuído consiste em dois ou mais arquivos localizados em sites diferentes. O banco de dados pode ser armazenado em vários computadores, localizados no mesmo local físico ou espalhados por diferentes redes.

> **Data warehouses**
> Um repositório central de dados, um data warehouse é um tipo de banco de dados projetado especificamente para consultas e análises rápidas.

> **Bancos de dados NoSQL**
> Um NoSQL, ou banco de dados não relacional, permite que dados não estruturados e semiestruturados sejam armazenados e manipulados (em contraste com um banco de dados relacional, que define como todos os dados inseridos no banco de dados devem ser

> compostos). Os bancos de dados NoSQL se tornaram populares à medida que os aplicativos web se tornaram mais comuns e mais complexos.

> **Bancos de dados gráficos**
> Um banco de dados gráfico armazena dados em termos de entidades e os relacionamentos entre entidades.

> **Bancos de dados OLTP**
> Um banco de dados OLTP é um banco de dados rápido e analítico projetado para um grande número de transações realizadas por vários usuários.

Um Banco de Dados envolve principalmente 4 componentes: **Dados, hardware, software e usuários**

- Dados pode ser definido aqui como tudo que está armazenado no banco de dados. Eles estão integrados e compartilhados. Também devem ser ***persistentes***, isto é, uma vez inseridos só podem ser apagados com um comando explícito.
- Hardware é o host em que o Banco de Dados irá funcionar. Ao conceito de hardware é imprescindível pesar os aspectos: ***volumas de armazenamento, dispositivos de entrada e saída, processador e memória***.
- Software é a camada que liga o Banco de Dados físico (isto é, aquele que está no hardware) ao usuário. A tal software damos o nome de ***Sistema Gerenciador de Banco de Dados***
- Usuários podem ser os programadores, os usuários comuns, os administradores de banco de dados (DBA - sigla em inglês) ou um Administrador de Dados.

Sistema Gerenciador de Banco de Dados

Através dos SGBDs (ou DBMS - sigla em inglês), os dados podem ser facilmente acessados, gerenciados, modificados, atualizados, controlados e organizados. A maioria dos bancos de dados usa a linguagem de consulta estruturada (SQL) para escrever e consultar dados.

> *O software de banco de dados simplifica o gerenciamento de dados, permitindo que os usuários armazenem dados em um formulário estruturado e depois os acessem. Ele normalmente tem uma interface gráfica para ajudar a criar e gerenciar os dados e, em alguns casos, os usuários podem usando o software do banco de dados.* (Oracle)

SQL (Structured Query Language)

Esta é a linguagem padrão utilizada para trabalhar com bases de dados *relacionais*. Ela possue as seguintes subdivisões:

1. **DDL (Definition Data Language)**: Define a criação de dados e colunas (**create table, alter table, create database**)
2. **DML (Data Manipulation Language)**: Serve para manipulação de dados (**insert, update, alter column, delete**)

> O SQL foi desenvolvido pela primeira vez na IBM nos anos 1970, com a Oracle como principal contribuinte, o que levou à implementação do padrão SQL ANSI; o SQL estimulou muitas extensões de empresas como IBM, Oracle e Microsoft. Embora o SQL ainda seja amplamente usado hoje em dia, novas linguagens de programação estão começando a aparecer.

> Um banco de dados geralmente é gerenciado por um SGBD (Sistema Gerenciador de Banco de Dados - ou DBMS, sigla em inglês). Eles servem para gerenciar, prover mecanismos para o banco de dados e esquemas, e servem como uma interface entre usuários e bancos de dados. Estes sistemas de gerenciamento organizam os dados armazenados por meio de uma técnica de esquema de DB chamada de ***Normalização***.

Anomalias e Normalização

Normalização é um processo de organização dos dados armazenados em um Banco de Dados de maneira lógica e eficaz. O objetivo principal da normalização é reduzir a redundância de dados, isto é, excesso de dados repetidos e também manter uma integridade dos dados. Mediante a decomposição das relações presentes no banco de dados, a *Normalização* busca ***Anomalias***, isto é, repetições e redundâncias entre os dados. Ao identificar anomalias, o conjunto de regras da normalização tentará eliminá-las e redefinir as relações afetadas, para que tudo se encaixe em seu devido lugar depois das alterações.

As anomalias nos bancos de dados são problemas que ocorrem quando os bancos de dados são mal planejados e não-normalizados. Em última análise, são mudanças em dados que podem gerar inconsistências dentro do Banco de Dados, como por exemplo, um mercado que disponibiliza um sistema de consulta de valores de produtos e ao passar um determinado produto o valor que aparece no caixa é diferente do valor mostrado no sistema de consulta.

As anomalias são divididas em 3 categorias:

1. Anomalias de **Inserção**
   Esse é um problema muito comum, que é quando temos que inserir muitas informações de forma manual, então é muito mais provável que ocorra um erro de digitação. Então pode gerar retrabalho e até informações erradas de preços, nomes, ou qualquer outra informação relevante.
   A melhor forma de evitar é criando relações que garantem a necessidade de um dado para inserção de um novo.

> **Exemplo**: Uma tabela *livros* precisa de um *autor*. Ao invés de permitir que sejam inseridos nomes de maneira manual, o ideal é criar uma tabela *autores* que irá conter identificadores (*autor_id*, por exemplo) e inserir um registro da tabela *livros* com o identificador do autor desejado.

2. Anomalias de **Exclusão**
   Acontece ao excluir um dado ou uma linha de informações. Isso pode gerar uma inconsistência no banco de dados, pois podemos excluir informações relevantes para outros bancos.

> **Exemplo**: Ainda usando o exemplo das tabelas *livros* e *autores*, podemos excluir um autor da tabela *autores*. Precisamos assim garantir que haja um efeito cascata, que garanta a exclusão de todos os livros que contém o identificador do autor excluído.

3. Anomalias de **Modificação**
   Acontece ao atualizar uma informação, pois é possível atualizar um dado específico e não atualizar no restante do banco de dados. Então teremos uma inconsistência ao pegar essas informações, o que pode gerar erro ou até problemas mais graves dependendo da área de atuação.

> **Exemplo**: Se um identificador de determinado autor for alterado, é necessário que todos os registros de livros que possuem aquele identificador tenham a coluna *autor_id* alterada também para o valor atualizado, garantindo a persistência e integridade do relacionamento.

Através do processo de normalização, empregamos diversas técnicas que garantem a resolução das anomalias citadas.
O conceito de **Normalização** foi apresentado originalmente em um artigo científico publicado pela IBM de autoria do matemático Edgar Codd, intitulado *"Um modelo de dados relacionais para grandes bancos de dados compartilhados"*. Codd se centra nos valores dos elementos relacionados no banco de dados, não em ligações ou agrupamentos específicos.
Ao identificar anomalias, o conjunto de regras da normalização tentará eliminá-las e redefinir

as relações afetadas, para que tudo se encaixe em seu devido lugar depois das alterações. O nome que se dá a uma regra da Normalização é **Forma Normal**. Existem 3 Formas Normais: ***Primeira Forma Normal (1FN), Segunda Forma Normal (2FN) e a Terceira Forma Normal (3FN) e a Quarta Forma Normal (4FN - opcional)*** .

Formas Normais

1. **Primeira Forma Normal (1FN)**
   A 1FN serve para assegurar que não haja informações repetidas em uma tabela, organizando dados em um grupos lógicos. Dessa forma, nos certificamos que os atributos estão sendo armazenados de forma única, isto é, não há nenhum outro atributo com os valores da mesma linha na tabela.

   > - Cada tabela em 1FN tem uma chave primária única que identifica cada fila da tabela.
   > - A chave primária pode ser composta por uma ou mais colunas da tabela.
   > - Cada coluna pode ter um só valor.

2. **Segunda Forma Normal (2FN)**
   A segunda forma trabalha focada nas possíveis redundâncias nas tabelas, em especial, define se os atributos da tabela dependem inteiramente da chave primária. Os atributos que não dependem ou dependem parcialmente da chave são associados a uma outra tabela, agora com uma relação clara com a chave primária da tabela original. Em outras palavras, a chave primária é convertida em chave estrangeira (ou externa) na nova tabela.
   A segunda forma normal se aplica somente depois que a primeira tiver sido realizada. Se uma tabela já cumpre com os requisitos de 1FN, pode-se verificar se ela também está alinhada com os de 2FN.

3. **Terceira Forma Normal (3FN)**
   Na terceira forma normal trabalhamos precisamente com a organização dos atributos que dependem uns dos outros, porém que não são atributos chaves (primárias ou estrangeiras). Caso necessário, é criada uma tabela secundária para reestruturar a relação de dependência entre os atributos. Essas tabelas devem ter a chave primária ou estrangeira.
   Consiste em verificar em uma tabela um conjunto de colunas que podem ser alocadas em uma segunda tabela e apontar através do uso de chaves primárias e estrangeiras, o registro na segunda tabela que indicará os dados combinados.

4. **Quarta Formal Normal (4FN)**
   A quarta e última forma normal é focada em eliminar dependências multivariadas entre os atributos chave, isto é, se há mais atributos (além das chaves primárias ou estrangeiras) que se repetem na tabela. Se isso ocorrer, geramos novas tabelas para eliminar tal redundância e manter as relações entre os atributos.

Com os conceitos de Anomalias e Normalização e como utilizar as Formas Normais para correção e prevenção de anomalias, podemos partir para outros conceitos importantes... Um banco de dados é formado por tabelas e colunas, e é necessário entender o conceito por trás desses fundamentos.

Tabelas

As tabelas são usadas como banco de dados para armazenar dados, as tabelas podem armazenar várias linhas de dados e cada linha possui um identificador exclusivo chamado chave primária.
Uma tabela é composta basicamente por conjuntos de colunas e linhas que contêm os dados contidos na tabela, coluna (ou campo) é um lugar onde você pode armazenar dados e linhas (ou registros) são os valores reais que são armazenados.
As tabelas no MySQL também são usadas para consultas que retornam conjuntos de linhas, as consultas podem ser simples, como, por exemplo, selecionar nome e sobrenome dos funcionários ou consultas mais complexas que recuperam grupos de linhas com a mesma chave primária.

Colunas

As colunas são os dados que estão gravados na tabela. Por exemplo, uma tabela *autores* pode conter a coluna *nome_autor* onde podemos inserir e modificar o nome do autor relacionado a linha desta tabela.

Constraints, Chaves Primárias e Chaves Estrangeiras

1. Uma constraint é uma restrição que atribuímos a uma tabela e um campo dessa tabela. Por exemplo, se quero dizer que o campo *cpf* da tabela *cidadão* é um campo restritivo de caráter único (ou seja, cada cidadão deve ter somente 1 cpf e o cpf não deve ser

repetido - aplicando assim o 1FN) posso declarar uma constraint de nome *unique_cpf_cidadao*.
2. Uma chave primária (em inglês *primary key*) é uma coluna que identificará um registro na tabela, por exemplo *id_cidadao* da tabela *cidadão*. Geralmente as chaves primárias contém um atributo **AUTO_INCREMENT** (*no caso do SGBD MySQL*) ou **SERIAL** (*no caso do SGBD postgreSQL*). Basicamente esses atributos entregam ao SGBD a responsabilidade de gerar chaves primárias únicas a cada registro criado no banco e também buscar o registro através desta chave.
3. Uma chave estrangeira é uma coluna que recebe um identificador (chave primária) de um registro localizado em outra tabela. Por exemplo, a tabela *carros* possui a coluna *id_cidadao_proprietario* que por sua vez irá receber o valor de uma chave primária relacionada a um registro da tabela *cidadãos*. Dessa forma, posso relacionar (*daí o conceito de Banco de Dados Relacional*) um carro a um cidadão específico.

Com o conceito de chaves primárias e estrangeiras, admitos então que há relações entre dados de diferentes tabelas. Portanto, é necessário verificar que existem diferentes tipos de relações no mundo dos bancos de dados relacionais.

Relacionamentos em Banco de Dados

Os relacionamentos de banco de dados são associações entre tabelas que são criadas usando instruções de junção para recuperar dados. Existem os seguintes relacionamentos em um banco de dados:

1. **Relacionamento um-para-um (1:1)**
   Ambas tabelas podem ter somente um registro de cada lado do relacionamento.
   Cada valor da chave primária se relaciona a nenhum ou a apenas um registro na tabela relacionada.
   A maioria dos relacionamentos de um para um são forçados por regras de negócios e não fluem naturalmente dos dados. Sem tal regra, geralmente você pode combinar as duas tabelas sem quebrar nenhuma regra de normalização.

2. **Relacionamento um-para-muitos (1:N)**
   A tabela de chave primária contém somente um registro relacionado a nenhum, a um ou a muitos registros da tabela relacionada.

3. **Relacionamento muitos-para-muitos (N:N)**
   Cada registro em ambas as tabelas pode se relacionar a nenhum ou a qualquer número

de registros na outra tabela. Esses relacionamentos requerem uma terceira tabela, chamada de tabela associada ou de associação, pois os sistemas relacionais não podem acomodar diretamente o relacionamento.

Referências

https://www.oracle.com/br/database/what-is-database/ (O que é um Banco de Dados?)
https://www.hostinger.com.br/tutoriais/dbms-o-que-e? (Entenda os Sistemas de Gerenciamento de Banco de Dados)
https://www.dio.me/articles/organizando-um-banco-de-dados-usando-as-formas-normais (Organizando um banco de dados usando as formas normais)
https://www.hashtagtreinamentos.com/anomalias-em-bancos-de-dados-sql (Anomalias em banco dedados - o que são? E quais problemas causam?)
https://ebaconline.com.br/blog/normalizacao-de-bases-de-dados (O que é a normalização de bases de dados e como fazê-la?)
https://www.alura.com.br/artigos/normalizacao-banco-de-dados-estrutura (Normalização em Banco de Dados - Estrutura)
https://metodoprogramar.com.br/banco-de-dados-mysql-o-que-e-o-que-sao-tabelas-campos-e-chave-primaria/ (Banco de Dados MySQL: o Que é, o Que São Tabelas, Campos e Chave Primária)
https://www.ibm.com/docs/pt-br/control-desk/7.6.1.2?topic=structure-database-relationships (# Relacionamentos do Banco de Dados)

Apostila feita por:
Victor Lima Reboredo
Meu Github
Meu Linkedin