Projeto de Banco de Dados

Conteudista: Prof.ª M.ª Vanderson Gomes Bossi
Revisão Textual: Esp. Andressa Moreira

Objetivo da Unidade:

- Apresentar o conceito de modelagem de dados e seus níveis de abstração, bem como a importância do levantamento de requisitos e regras de negócio para o processo de desenvolvimento do projeto de banco de dados.

Material Teórico

Projeto de Banco de Dados

O projeto de banco de dados pode ser geralmente definido como uma coleção de tarefas ou processos que aprimoram o *design*, o desenvolvimento, a implementação e a manutenção do sistema de gerenciamento de dados corporativos. Projetar um banco de dados adequado reduz o custo de manutenção, melhorando assim a consistência dos dados, e as medidas econômicas são grandemente influenciadas em termos de espaço de armazenamento em disco. Portanto, o projeto de um banco de dados deve seguir restrições e decidir como os elementos se correlacionam e quais tipos de dados devem ser armazenados.

Os principais objetivos por trás do projeto de banco de dados são produzir modelos de projeto físico e lógico do sistema de banco de dados proposto. O modelo lógico concentra-se principalmente nos requisitos de dados e suas considerações devem ser feitas em termos de considerações monolíticas e, portanto, os dados físicos gravados devem ser armazenados independentemente das condições físicas. Por outro lado, o modelo físico inclui uma tradução do modelo lógico do banco de dados, mantendo o controle dos recursos de *hardware* e sistemas de *software* como Sistema de Gerenciamento de Banco de Dados (SGBD).

Vídeo

How to Design Your First Database

Etapas de um Projeto de Banco de Dados

Em primeiro lugar, o planejamento deve ser feito baseando-se nos requisitos básicos do projeto sob o qual o desenho da base de dados deve ser levado adiante. Assim, eles podem ser definidos como:

- **Planejamento:** esta etapa se preocupa em planejar todo o DDLC (ciclo de vida de desenvolvimento de banco de dados). As considerações estratégicas são levadas em consideração antes de prosseguir;

- **Definição do sistema:** esta etapa cobre os limites e escopos do banco de dados adequado após o planejamento.

A próxima etapa envolve projetar o banco de dados considerando os requisitos baseados no usuário e dividi-los em vários modelos para que não sejam impostas cargas ou dependências pesadas em um único aspecto. Portanto, uma abordagem centrada nos modelos lógicos e físicos desempenha um papel crucial.

- **Modelo físico:** o modelo físico preocupa-se com as práticas e implementações do modelo lógico;
- **Modelo lógico:** esta etapa preocupa-se principalmente em desenvolver um modelo baseado nos requisitos propostos. Todo o modelo é projetado no papel, sem qualquer implementação ou adoção de considerações de SGBD.

Introdução à Modelagem de Dados Relacional

O modelo relacional é uma abordagem no mundo dos bancos de dados que traz clareza e eficiência ao gerenciamento de informações. Nesse modelo, as estruturas lógicas de dados, como tabelas, visualizações e índices, são mantidas separadas das estruturas físicas de armazenamento. Isso significa que os administradores de banco de dados podem cuidar do armazenamento físico dos dados sem afetar a maneira como são acessados de maneira lógica.

Um exemplo simples disso é que, se você decidir renomear um arquivo de banco de dados, isso não afetará o nome das tabelas que estão armazenadas dentro dele.

Outro aspecto importante desse modelo está relacionado às operações de banco de dados. Existem operações lógicas e operações físicas: as operações lógicas permitem que um aplicativo especifique o que deseja dos dados, ou seja, o conteúdo necessário; por outro lado, as operações físicas determinam como os dados devem ser acessados e executam a tarefa em questão.

Além disso, o modelo relacional coloca um grande foco na integridade dos dados. Isso significa que são aplicadas regras específicas para garantir que os dados sejam sempre precisos e acessíveis. Por exemplo, uma regra de integridade pode proibir a existência de linhas duplicadas em uma tabela, evitando assim que informações erradas entrem no banco de dados.

Antes do surgimento desse modelo, cada aplicativo costumava armazenar dados em sua própria estrutura exclusiva, o que dificultava a integração entre aplicativos e exigia um conhecimento profundo da estrutura de dados específica para encontrar informações. Essas estruturas de dados eram ineficientes, difíceis de manter e não otimizadas para um bom desempenho.

O modelo relacional veio para solucionar esses problemas, oferecendo uma forma padronizada de representar e consultar dados que poderia ser usada por qualquer aplicativo. Ele introduziu o uso de tabelas, que são uma maneira intuitiva, eficiente e flexível de armazenar e acessar informações estruturadas. Isso tornou mais fácil para os desenvolvedores representarem os dados de forma clara e consistente, facilitando o acesso e a consulta.

Com o tempo, o uso da linguagem *SQL* (*Structured Query Language*) tornou-se outra vantagem do modelo relacional. O SQL, baseado na álgebra relacional, tornou-se a linguagem padrão para consulta e manipulação de dados em sistemas de bancos de dados relacionais. Sua base matemática sólida garante a consistência interna das consultas ao banco de dados e simplifica a otimização de desempenho. Em comparação, outras abordagens muitas vezes exigem a definição de consultas individuais, o que pode ser mais complexo e suscetível a erros.

Em resumo, o modelo relacional trouxe uma abordagem mais organizada e eficaz para o gerenciamento de dados, com a separação entre lógica e física, operações claras de banco de dados, ênfase na integridade dos dados e a eficiência das tabelas e da linguagem SQL. Isso revolucionou a forma como os sistemas de bancos de dados são projetados e continua sendo uma base sólida para aplicações de gerenciamento de dados modernas.

Vídeo

Modelagem de Dados – O Modelo Relacional – Introdução

Benefícios do Sistema de Gerenciamento de Banco de Dados Relacional

O modelo relacional é uma abordagem fundamental e amplamente adotada no gerenciamento de informações por organizações de todos os portes. Sua simplicidade e poder o tornam a escolha ideal para diversas necessidades de informação. Bancos de dados relacionais desempenham um papel crucial em várias aplicações, desde rastrear estoques até processar transações de comércio eletrônico e gerenciar informações vitais do cliente.

Imagine um banco de dados relacional como um guarda-roupa organizado, em que cada peça de roupa é cuidadosamente disposta em categorias específicas. Isso permite encontrar qualquer peça rapidamente e garantir que elas estejam sempre em ordem.

Esses bancos de dados são especialmente úteis quando os pontos de dados estão interconectados e precisam ser gerenciados de forma segura, obedecendo a regras e mantendo consistência. Por exemplo, em um sistema de vendas *on-line*, é fundamental manter a

correlação entre os produtos, os pedidos e os clientes. O modelo relacional oferece a estrutura perfeita para fazer isso.

Os bancos de dados relacionais não são uma novidade; eles existem desde a década de 1970. Mesmo com o avanço da tecnologia, o modelo relacional continua sendo a abordagem mais aceita e amplamente utilizada em aplicações comerciais. Isso se deve à sua capacidade de fornecer organização, segurança e confiabilidade aos dados.

Modelo Relacional e Consistência de Dados

O modelo relacional é amplamente reconhecido por sua habilidade excepcional em manter a consistência dos dados em diferentes aplicativos e cópias de bancos de dados, conhecidas como instâncias. Isso é essencial em situações em que a precisão e a atualização imediata das informações são cruciais. Vamos explorar porque o modelo relacional se destaca nesse aspecto e como ele se compara a outros tipos de bancos de dados, como os NoSQL.

Imagine a seguinte situação: um cliente realiza um depósito em um caixa eletrônico e, em seguida, verifica o saldo da conta em seu celular. Nesse cenário, o cliente espera ver o saldo atualizado imediatamente após o depósito. O modelo relacional é particularmente eficaz em garantir essa consistência de dados em tempo real. Isso significa que, em várias instâncias dos bancos de dados, os dados permanecem consistentes e refletem as mudanças quase que instantaneamente.

No entanto, outros tipos de bancos de dados, como os NoSQL, enfrentam desafios ao lidar com a mesma consistência em situações semelhantes. Eles podem oferecer apenas o que é chamado de “consistência eventual”. Isso significa que, quando um banco de dados é escalado para lidar com um grande volume de dados, ou quando vários usuários acessam os mesmos dados simultaneamente, pode haver um pequeno intervalo de tempo até que todos os dados estejam completamente atualizados e consistentes em todas as instâncias.

A consistência eventual pode ser aceitável em determinados cenários, como manter listagens em um catálogo de produtos, em que pequenos atrasos não são críticos. No entanto, em operações comerciais essenciais, como transações de carrinho de compras, em que a precisão

dos dados é fundamental para evitar erros financeiros, o banco de dados relacional ainda é considerado o padrão ouro.

Em suma, o modelo relacional é altamente valorizado por sua capacidade de manter a consistência dos dados em tempo real entre diferentes instâncias de banco de dados. Isso é especialmente importante em operações comerciais críticas, em que a precisão e a atualização imediata dos dados são fundamentais. Embora existam outras abordagens, como os bancos de dados NoSQL, que podem oferecer consistência eventual, o modelo relacional continua sendo a escolha preferencial quando se trata de garantir a integridade e a precisão dos dados em aplicações empresariais vitais.

Leitura

Teorema CAP

Clique no botão para conferir o conteúdo.

ACESSE

Compromisso e Atomicidade

Os bancos de dados relacionais têm um papel fundamental no gerenciamento de regras e políticas de negócios de forma minuciosa e precisa. Eles operam em um nível detalhado, aplicando políticas rígidas relacionadas ao compromisso dos dados, ou seja, tornar permanentes as alterações feitas no banco de dados. Para ilustrar essa capacidade,

consideremos um exemplo em que um banco de dados de controle de inventário acompanha três peças que sempre são usadas juntas.

Quando uma dessas peças é retirada do estoque, as outras duas também precisam ser retiradas. No entanto, existe uma regra estrita: se uma das três partes não estiver disponível, nenhuma delas deverá ser retirada. O banco de dados relacional opera com a precaução de que todas as três partes precisam estar disponíveis antes que qualquer compromisso seja feito. Esse tipo de compromisso multifacetado é chamado de atomicidade.

A atomicidade é essencial para manter a precisão e a integridade dos dados no banco de dados, bem como para garantir que esses dados estejam em conformidade com as regras, regulamentos e políticas de negócios. Em outras palavras, antes de que qualquer alteração ser efetivada no banco de dados, todas as condições e requisitos especificados pelas políticas de negócios devem ser atendidos. Isso evita que situações inconsistentes ocorram, como a retirada de uma parte sem as outras duas, o que poderia levar a erros graves no controle de inventário.

Assim, o modelo relacional oferece uma abordagem rigorosa e confiável para a gestão de dados em conformidade com as políticas de negócios. Isso é de suma importância para garantir que os dados no banco de dados sejam confiáveis e sigam os procedimentos estabelecidos pela organização. A capacidade de garantir que todas as condições sejam satisfeitas antes de efetivar qualquer alteração é uma característica crucial do modelo relacional, conhecida como atomicidade, que contribui para a consistência e a precisão dos dados.

Propriedades ACID

Quatro propriedades cruciais definem as transações de banco de dados relacional: atomicidade, consistência, isolamento e durabilidade – normalmente chamadas de ACID.

- **Atomicidade:** define todos os elementos que constituem uma transação completa de banco de dados;
- **Consistência:** define as regras para manter os pontos de dados em um estado correto após uma transação;

- **Isolamento:** mantém o efeito de uma transação invisível para os outros até que ela seja confirmada, para evitar confusão;

- **Durabilidade:** garante que as alterações nos dados se tornem permanentes assim que a transação for confirmada.

Vídeo

O que é ACID?

Bloqueio e Simultaneidade de Banco de Dados

Podem surgir conflitos em um banco de dados quando vários usuários ou aplicativos tentam alterar os mesmos dados ao mesmo tempo. As técnicas de bloqueio e simultaneidade reduzem o potencial de conflitos, ao mesmo tempo que mantêm a integridade dos dados.

O bloqueio impede que outros usuários e aplicativos acessem os dados enquanto eles estão sendo atualizados. Em alguns bancos de dados, o bloqueio se aplica a toda a tabela, o que cria um impacto negativo no desempenho do aplicativo. Outros bancos de dados, como os bancos de dados relacionais Oracle, aplicam bloqueios no nível do registro, deixando os demais registros da tabela disponíveis e ajudando a garantir melhor desempenho do aplicativo.

A simultaneidade gerencia a atividade quando vários usuários ou aplicativos invocam consultas ao mesmo tempo no mesmo banco de dados. Esse recurso fornece acesso correto a usuários e aplicativos de acordo com políticas definidas para controle de dados.

Principais Bancos de Dados Relacionais

O armazenamento de dados é essencial para *softwares*. Seja para armazenar dados de clientes, empresas, transações, entre outros itens importantes, é necessário ter um banco de dados para esta importante missão.

Embora existam muitos modelos de banco de dados hoje, o banco de dados relacional ainda se destaca por uma série de fatores. Seja pela sua estrutura de funcionamento, seja pela integridade de armazenamento e recuperação de dados, os bancos de dados relacionais trazem características importantes para o desenvolvedor e para o usuário.

Assim, podemos relacionar os bancos de dados relacionais mais utilizados:

- **MySQL:** é um sistema de gerenciamento de banco de dados SQL de código aberto mais popular que é desenvolvido, distribuído e suportado pela *Oracle Corporation*. O *software* de banco de dados MySQL é um sistema cliente/servidor que consiste em um servidor SQL *multithread* que suporta diferentes linguagens de programação e

bibliotecas diferentes, ferramentas administrativas e uma ampla gama de interfaces de programação de aplicativos.

Vídeo

What is MySQL? | Beginners Guide

- ***Oracle Database*:** é o primeiro banco de dados projetado para computação em grade empresarial, a maneira mais flexível e econômica de gerenciar informações e aplicativos. A computação em grade corporativa cria grandes conjuntos de armazenamento e servidores modulares padrão do setor. Com esta arquitetura, cada novo sistema pode ser rapidamente provisionado a partir do conjunto de

componentes. Não há necessidade de picos de carga de trabalho, pois a capacidade pode ser facilmente adicionada ou realocada a partir dos pools de recursos, conforme necessário.

A arquitetura *grid* da *Oracle* reúne um grande número de servidores, armazenamento e redes em um recurso de computação flexível e sob demanda para as necessidades de computação empresarial. A infraestrutura de computação em grade analisa continuamente a demanda por recursos e ajusta a oferta de acordo.

Por exemplo, você poderia executar diferentes aplicativos em uma grade de vários servidores de banco de dados vinculados. Quando os relatórios vencem no fim do mês, o administrador do banco de dados pode provisionar automaticamente mais servidores para esse aplicativo para lidar com o aumento da demanda.

Vídeo

What is Oracle Database and What db Can do

- ***Microsoft SQL Server:*** é um sistema de gerenciamento de banco de dados relacional (RDBMS) que oferece suporte a uma ampla variedade de aplicativos de processamento de transações, inteligência de negócios e análise em ambientes de TI corporativos.

Como outros *softwares* gerenciadores de banco de dados, o *Microsoft SQL Server* é construído com base na linguagem SQL, sendo uma linguagem de programação padronizada que administradores de banco de dados e outros profissionais de TI usam para gerenciar bancos de dados e consultar os dados que eles contêm. O SQL Server está vinculado ao ***Transact-SQL*** (T-SQL), uma implementação da linguagem SQL reformulada pela *Microsoft* que adiciona um conjunto de extensões de programação proprietárias à linguagem padrão.

Vídeo

What is Microsoft SQL Server?

- ***PostgreSQL:*** é um sistema de banco de dados relacional de objeto de código aberto com mais de 35 anos de desenvolvimento ativo que lhe rendeu uma forte reputação de confiabilidade, robustez de recursos e desempenho.

O *PostgreSQL* conquistou uma forte reputação por sua arquitetura comprovada, confiabilidade, integridade de dados, conjunto robusto de recursos, extensibilidade e dedicação da

comunidade de código aberto por trás do *software* para fornecer soluções inovadoras e de desempenho consistente. O *PostgreSQL* é executado em todos os principais sistemas operacionais, é compatível com ACID desde 2001 e possui complementos poderosos, como o popular extensor de banco de dados geoespacial ***PostGIS***.

Vídeo

O Que é *PostgreSQL*?

- ***Firebird:*** é um banco de dados relacional que oferece muitos recursos do padrão ANSI SQL que roda em *Linux*, *Windows* e uma variedade de plataformas *Unix*. O *Firebird* oferece excelente simultaneidade, alto desempenho e poderoso suporte de linguagem para procedimentos armazenados e gatilhos. Ele tem sido usado em sistemas de produção, sob vários nomes, desde 1981.

O Projeto *Firebird* é comercialmente independente de programadores C e C++, consultores técnicos e apoiadores que desenvolvem e aprimoram um sistema de gerenciamento de banco de dados relacional multiplataforma baseado no código-fonte lançado pela *Inprise Corp* (agora conhecida como *Borland Software Corp*) em 25 de julho de 2000.

Vídeo

How to Install and Use Firebird Database in Windows

Modelo Conceitual

O modelo de dados conceitual permite uma visão geral do sistema no qual está trabalhando sem ter que se preocupar com as minúcias da implementação. A modelagem conceitual é o momento de fazer perguntas, interagir com as partes interessadas e pensar nos requisitos e processos de negócios que o banco de dados irá abordar. Embora os modelos de dados conceituais sejam rápidos de construir, eles também podem revelar rapidamente suposições incorretas e possíveis problemas. O modelo conceitual é uma representação simplificada do banco de dados final, com as especificidades removidas para focar no quadro geral.

Principais Recursos da Modelagem Conceitual de Banco de Dados

- **Defina as necessidade de negócios de maneira nítida e precisa:** defina as necessidades de negócios de maneira nítida e precisa: do ponto de vista do gerenciamento de recursos e de

tempo, um modelo conceitual pode ajudar as principais partes interessadas a compreender melhor o que é necessário para atingir os objetivos de negócios pretendidos;

- **Dê à organização um vocabulário empresarial:** ao apresentar uma imagem de conceitos, entidades e seus relacionamentos, as lacunas na linguagem podem ser identificadas e possíveis falhas eliminadas;

- **Promover uma forma de comunicação orientada para a análise:** é necessária a contribuição das partes interessadas, tanto de dentro como de fora da bolha tecnológica, para garantir uma adoção bem-sucedida. Modelos de dados conceituais auxiliam nessa colaboração;

- **Crie um plano de negócios para obter resultados favoráveis:** a modelagem de dados conceituais pode ser considerada o primeiro passo em direção a uma modelagem de dados mais complexa.

Características de um Modelo de Dados Conceitual

Este tipo de modelo de dados é projetado e desenvolvido para um público empresarial. O modelo conceitual é desenvolvido independentemente das especificações de *hardware*, como capacidade de armazenamento de dados, localização ou especificações de *software*, como fornecedor e tecnologia de SGBD. O foco é representar os dados como o usuário os verá no "mundo real".

Modelos de dados conceituais conhecidos como modelos de domínio criam um vocabulário comum para todas as partes interessadas, estabelecendo conceitos básicos e escopo.

Vídeo

Modelagem de Dados – Modelos Conceitual, Lógico e Físico

Normalização, Modelo Lógico e Físico

A modelagem lógica é a ponte entre o modelo conceitual do negócio e a estrutura física do banco de dados. A modelagem lógica usa convenções que contextualizam a natureza dos dados além do que os modelos conceituais ou físicos podem expressar.

O modelo lógico aumenta o conceitual adicionando notação que descreve metadados de relacionamento e incorpora elementos do projeto de banco de dados. É nesta fase de modelagem lógica que são introduzidos os detalhes técnicos necessários para a implementação do banco de dados, como tipos de dados e chaves, ainda que de forma simplificada. O que separa a

modelagem lógica da física é que os detalhes técnicos não são específicos de nenhum tipo de banco de dados.

Ficar entre os domínios físico e conceitual permite que a modelagem lógica faça a ponte entre os dois mundos e expresse nuances que, de outra forma, seriam perdidas na rigidez dos objetos de bancos de dados puros.

Assim, podemos dizer que um modelo de dados lógico geralmente descreve os dados com o máximo de detalhes possível, sem a necessidade de se preocupar com as implementações físicas no banco de dados. Os recursos do modelo de dados lógicos podem incluir:

- Todas as entidades e relacionamentos entre elas;
- Cada entidade possui atributos bem especificados;
- A chave primária para cada entidade é especificada;
- São especificadas chaves estrangeiras que são usadas para identificar um relacionamento entre diferentes entidades;
- A normalização ocorre neste nível.

A normalização do banco de dados é o processo de organização de um banco de dados de forma a reduzir a redundância e a dependência em suas tabelas. Isso é conseguido dividindo uma tabela grande em tabelas menores e vinculando-as por meio de relacionamentos de chave estrangeira. Isso leva a menos inconsistências de dados e melhora a integridade dos dados. Um banco de dados normalizado resulta em um *design* modular fácil de dimensionar e modificar.

A normalização ocorre através de estágios crescentes de regras formais chamadas formas normais, que vão desde a primeira forma normal (1FN) até a sexta forma normal (6FN) – embora da primeira à terceira sejam mais comumente usadas e sejam suficientes para a maioria dos casos de uso.

Cada forma normal baseia-se nos requisitos de seu antecessor e adiciona critérios adicionais que toda tabela de banco de dados deve satisfazer. Uma forma normal é considerada satisfeita quando cada tabela do banco de dados atende aos critérios estabelecidos para ela (e, por extensão, para suas antecessoras).

Negligenciar as regras de normalização pode resultar em dados errôneos e autocontraditórios que são difíceis de gerenciar. A Tabela 1 a seguir apresenta uma tabela que viola quase todos os princípios de consistência existentes:

Tabela 1 – Exemplo de Dados que não Seguem Nenhuma Regra de Normalização

Álbum	Artista	País	Etiqueta	Gênero	Tempo	Formato	Preço	Cert	RIAA Cert	Riaa Cert ID
Aquimi	Solo	Brasil	*LaF*	*Country*	74:10	CD	23,00	XX	Ouro	2
Aquimi	Solo	USA	*LaFace*	*Country*	74:10	CD	21,00	2XX	Prata	2

Alguns problemas de dados que conseguimos identificar em apenas duas linhas:

- Nenhum identificador exclusivo torna difícil saber o que uma única linha representa;
- As duas linhas parecem estar duplicadas, mas não há como saber sem uma chave primária;
- EUA e Brasil são dois valores descrevem a mesma entidade;
- A entrada de dados duplicada na coluna etiqueta resulta em valores com erros ortográficos;

- Se a loja em questão quiser parar de vender o formato CD, a exclusão de ambas as linhas também removeria as informações do álbum.

A normalização ajuda a evitar anomalias de dados, como as anteriores, por meio do projeto do esquema e regras de banco de dados. À medida que mais a normalização é aplicada, a possibilidade de informações anômalas ou duplicadas diminui enquanto o número geral de tabelas aumenta. As proteções de qualidade dos dados obtidas por meio da normalização são compensadas pela complexidade de consultar informações em diversas tabelas, em vez de tê-las em uma única fonte.

Antes de analisarmos a progressão das formas normais, precisamos compreender melhor as anomalias de dados das quais eles ajudam a proteger.

Anomalias de Dados

Desde erros ortográficos até codificação inadequada, alguns problemas de qualidade dos dados não podem ser evitados. No entanto, os *designs* desnormalizados tornam possível cair de cabeça em vários erros bem conhecidos e evitáveis.

Para compreender como a normalização evita anomalias nos dados, precisamos desvendar os perigos duplos que ela mitiga: redundância e dependência:

- **Redundância:** dados repetidos, seja em uma ou em várias tabelas. Quando os valores dos dados são duplicados, sincronizar tudo por meio de operações DML fica mais difícil;

- **Dependência:** quando o valor de um atributo depende do valor de outro. As dependências podem ser funcionais (como o atributo idade de uma pessoa, dependendo de seu nome) ou multivaloradas (como nome, idade e *hobby* armazenados em uma única tabela tornariam impossível excluir um *hobby* sem excluir todas as pessoas que o praticam).

Normalização de Banco de Dados Através de Exemplos

A normalização foi proposta pela primeira vez por Edgar F. Codd, o inventor do modelo relacional para gerenciamento de banco de dados. Codd introduziu a normalização na forma 1FN (Primeira Forma Normal) e posteriormente estendeu-a para 2FN e 3FN. Mais tarde, Codd, trabalhando com Raymond F. Boyce, desenvolveu a Forma Normal de Boyce-Codd (BCNF), também chamada de 3,5FN.

À medida que a teoria do banco de dados continua a se desenvolver, foram propostas formas normais subsequentes até **6FN**, seguindo a progressão da menos para a mais restritiva. Embora seja importante compreender a normalização em todas as suas formas, também é essencial reconhecer que os cenários empresariais típicos não exigem que se ultrapasse a 3FN.

Vídeo

Modelagem de Dados – Normalização – Primeira Forma Normal

Primeira Forma Normal (1FN) é satisfeito quando o seguinte estiver em vigor:

- Cada registro é único;
- Cada célula contém um único valor;

Segunda Forma Normal (2FN) é satisfeito quando o seguinte está em vigor:

- As regras 1NF são satisfeitas;
- Cada atributo de chave não candidata deve depender de toda a chave candidata (sem dependências parciais).

Terceira Forma Normal (3FN) é satisfeito quando o seguinte estiver em vigor:

- As regras 2NF são satisfeitas;
- Não há dependências funcionais transitivas.

Modelo Entidade-Relacionamento

O Modelo Entidade-Relacionamento (Modelo ER) é uma técnica amplamente usada na área de bancos de dados para representar e organizar os dados de um sistema. Esta técnica oferece uma maneira intuitiva e eficaz de visualizar como os dados estão estruturados e como eles se relacionam entre si. Para isso, precisamos entender alguns conceitos:

Entidade

São objetos do mundo real que podem ser identificados de forma única e que desempenham um papel importante no contexto do sistema de banco de dados. Elas representam categorias específicas de informações. Por exemplo, em um sistema de gerenciamento de uma loja de varejo, as entidades podem incluir **"clientes"**, **"produtos"** e **"pedidos"**, onde:

- **Clientes:** representam as pessoas ou organizações que realizam compras na loja;
- **Produtos:** referem-se aos itens disponíveis para compra, como roupas, eletrônicos, alimentos etc;
- **Pedidos:** são registros das transações realizadas pelos clientes, que incluem informações sobre quais produtos foram comprados, a data da compra, o valor total, entre outros.

Relacionamentos

São as conexões estabelecidas entre as entidades. Eles descrevem como as entidades interagem entre si ou estão associadas dentro do sistema. No exemplo anterior da loja de varejo:

- Um **cliente** pode estar relacionado a **vários pedidos**. Isso significa que um cliente pode realizar múltiplas compras ao longo do tempo;
- Um **pedido** pode estar relacionado a **vários produtos**. Isso quer dizer que um pedido pode incluir diferentes itens.

Esses relacionamentos são essenciais para entender como as diferentes partes do sistema de banco de dados se conectam e interagem entre si.

O modelo entidade relacionamento pode ser considerado um modelo conceitual, o que significa que ele se concentra na representação de dados de uma forma independente de qualquer sistema de banco de dados específico. Isso o torna uma ferramenta valiosa para projetar sistemas de banco de dados antes mesmo de sua implementação.

Vantagens do Modelo ER

- **Facilidade de compreensão:** é uma ferramenta intuitiva que pode ser compreendida mesmo por pessoas que não são especialistas em bancos de dados. Isso facilita a comunicação e a colaboração entre equipes de desenvolvimento e usuários finais;
- **Flexibilidade:** o modelo entidade relacionamento é altamente flexível e pode ser aplicado a uma ampla variedade de cenários e tipos de dados. Ele não está vinculado a um sistema de gerenciamento de banco de dados específico, o que o torna versátil;
- **Eficiência:** ajuda a identificar possíveis problemas de *design* de banco de dados antes da implementação, economizando tempo e recursos. Também contribui para o aprimoramento do desempenho do banco de dados, garantindo que as estruturas de dados sejam otimizadas.

Em resumo, o modelo entidade relacionamento é uma ferramenta poderosa e versátil para modelagem de dados em sistemas de banco de dados. Ele ajuda a representar, de maneira clara e lógica, as entidades, seus relacionamentos e as características essenciais dos dados, independentemente de detalhes de implementação. Profissionais de TI o utilizam extensivamente para projetar e implementar bancos de dados eficientes e bem estruturados.

Para ilustrar como o modelo entidade relacionamento é aplicado, consideremos um exemplo prático de um banco de dados de uma loja de varejo. Neste sistema:

- **Entidades:** temos três entidades principais: clientes, produtos e pedidos;
- **Relacionamentos:** os relacionamentos incluem a capacidade de um cliente fazer vários pedidos e um pedido incluir vários produtos.

A representação visual desse modelo pode ser feita da seguinte forma:

Figura 1 – Sistema entidade-relacionamento

#ParaTodosVerem: imagem de um esquema visual com 3 caixas de texto interligadas. Na primeira caixa, Clientes; na segunda, Pedidos e na terceira, Produtos. Fim da descrição.

Neste diagrama, os retângulos representam as entidades (clientes e produtos) e as linhas ligadas por um losango representam os relacionamentos (clientes fazem pedidos, pedidos incluem produtos). Isso oferece uma visão clara de como as entidades estão conectadas e como

os dados fluem dentro do sistema de banco de dados. É uma ferramenta essencial para o projeto e a compreensão de sistemas de banco de dados complexos.

Material Complementar

Indicações para saber mais sobre os assuntos abordados nesta Unidade:

Vídeo

Modelagem de Dados – Modelo Entidade – Relacionamento e Diagrama ER

Leitura

Mapeamento do Modelo Entidade Relacionamento (ER) para o Modelo Relacional

Clique no botão para conferir o conteúdo.

ACESSE

O que é um Diagrama Entidade Relacionamento? | Lucidchart

Clique no botão para conferir o conteúdo.

ACESSE

Chapter 5 Data Modelling

Clique no botão para conferir o conteúdo.

ACESSE

Referências

DATABASE design basics. ***Microsoft Support***, s.d. Disponível em: <https://support.microsoft.com/en-gb/office/database-design-basics-eb2159cf-1e30-401a-8084-bd4f9c9ca1f5>. Acesso em: 07/10/2023.

SEBASTIAN,C. L. *Data Management*, *Models*, *and Metadata*. *In*: SEBASTIAN-COLEMAN, L. (Ed.). ***Measuring Data Quality for Ongoing Improvement***. Oxford, England: Elsevier, 2013. p. 27-37.

TRAINA JR., C.; FERREIRA, J. E.; BIAJIZ, M. *Use of a semantically grained database system for distribution and control within design environments*. ***Lecture Notes in Computer Science*** (*including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics*), v. 1300, LNCS, p. 1130-1134, 1997.

TUPPER, C. D. *Model constructs and model types*. *In*: TUPPER, C. D. (Ed.). ***Data Architecture. Oxford***, *England*: Elsevier, 2011. p. 207-221.