Técnico em Desenvolvimento de Sistemas

Modelagem conceitual do banco de dados

Neste material, você aprenderá sobre a modelagem conceitual de um banco de dados e as técnicas de levantamento de dados e requisitos funcionais e não funcionais.

Além disso, você estudará os modelos relacionais embasados na teoria dos conjuntos, que fazem o relacionamento entre as tabelas e deixam os dados acessíveis e possíveis de serem consultados de diferentes formas, considerando seus tipos e tamanhos.

Modelagem conceitual

O modelo conceitual do banco de dados é uma representação simples. Trata-se de uma representação gráfica das estruturas (tabelas) do banco de dados, em que é possível identificar todas as relações e restrições e suas demais características.

O desenvolvimento desse modelo é considerado uma das primeiras e principais etapas no projeto de desenvolvimento do banco de dados. Os projetos de banco de dados mantêm seu foco na maneira como sua estrutura será utilizada para armazenar e gerenciar os dados do usuário final.

A modelagem de dados começa com a compreensão do ambiente real, com escopo e objetivos bem-definidos. Se realizado adequadamente, o modelo de dados final torna-se o equivalente a uma “planta baixa”, com todas as instruções para a construção de um banco de dados que atenda às necessidades dos usuários finais.

O modelo conceitual de dados **mais utilizado atualmente**, devido principalmente à simplicidade e à eficiência, é o modelo entidade-relacionamento, também chamado de **diagrama entidade-relacionamento (ER)**.

Para construir um banco de dados sem erros, é preciso aprender a utilizar técnicas pertinentes a sua construção. Confira, a partir de agora, as principais técnicas, estratégias e definições para a construção do banco de dados.

Técnicas de levantamento de dados

O levantamento de dados é uma forma de mapeamento e o principal objetivo é permitir conhecer as principais e reais necessidades do cliente. Em grande parte dos casos, o cliente pode não saber expressar claramente as reais necessidades que precisam ser atendidas com a construção e a utilização de *software* com banco de dados.

Para realizar esses levantamentos adequadamente, deve-se aplicar as seguintes técnicas:

Observação direta ou pessoal

Essa técnica permite vivenciar o dia a dia da empresa. Por meio dela, obtém-se a confirmação de informações recebidas, como erros de procedimentos. É um elemento importante quando outras técnicas já tiverem sido aplicadas, como a entrevista, por exemplo.

Questionário

Esse é um instrumento normalmente preparado em formulário para levantar determinadas informações. O questionário pode ser aberto ou fechado: o fechado apresenta questões e suas possíveis respostas; já o aberto permite que cada participante responda às questões de acordo com o seu entendimento. O questionário deve ser elaborado com cuidado, principalmente quando se tratar do tipo aberto, para que as respostas sejam mais detalhadas, evitando simplesmente “sim” e “não”.

Entrevista

Assim como o questionário, a entrevista também apresenta algumas questões. A grande diferença é que os questionamentos, nesse caso, podem ser modificados durante a aplicação do instrumento.

Análise da documentação

Essa técnica consiste na análise dos documentos já existentes que a empresa utiliza para organizar a sua rotina, como o cadastro de clientes, por exemplo. As informações obtidas a partir da aplicação dessas técnicas serão norteadoras para o desenvolvimento dos requisitos e das demais características do projeto de banco de dados.

É muito comum que precise ser realizada mais de uma técnica para compreender o que deve ser modelado. Não existe uma fórmula pronta e cada levantamento de dados poderá englobar várias técnicas.

Levantamento e especificação de requisitos

A partir do levantamento de dados, você estará apto para realizar especificações e requisitos aos quais o *software* deve atender.

Todo projeto de *software* tem um conjunto de requisitos, os quais são determinados pelas necessidades e pelas expectativas dos usuários que efetivamente o utilizarão. Além disso, os requisitos devem estar relacionados ao atendimento dos objetivos dos negócios da empresa.

Um bom requisito deve endereçar uma necessidade direta ou indireta dos futuros usuários do sistema a ser desenvolvido.

A prática recomendada é sempre documentar, organizar e disponibilizar os requisitos a todos os envolvidos no projeto, garantindo que seu entendimento seja compartilhado entre todos, ou seja, clientes e equipe responsável pelo projeto.

Existe uma variedade de tipos de requisitos, que deverá estar relacionada a funcionalidades esperadas para o sistema e também ao desempenho, à segurança e à confiabilidade de informações. Quando se trata de projeto de desenvolvimento de *softwares*, os requisitos dividem-se em requisitos funcionais e não funcionais.

Há várias maneiras de documentar e disponibilizar esses requisitos. Uma das ferramentas para essa documentação é o DER, que monta as entidades e os relacionamentos entre elas. Outra ferramenta possível são os diagramas UML (*unified modeling language*).

Dicionário de dados

O dicionário de dados fornece uma descrição detalhada entre todas as tabelas encontradas no banco de dados criado pelo usuário ou pelo projetista. Pode-se dizer que o dicionário de dados é um documento de texto ou uma planilha que centraliza as informações sobre todo o conjunto de dados (*dataset*).

Ele contém no mínimo todos os nomes e as características dos atributos de cada uma das tabelas do sistema.

Importância da dicionarização

Para contextualizar e transmitir informações, muitas vezes são utilizados elementos que nem são percebidos, pois estão no cotidiano das pessoas, como entonação, postura, velocidade e até gestos.

Quando ocorre a transformação da informação para dados, os quais serão armazenados em uma base, em planilhas etc., esses dados podem ser reutilizados em outro momento por outras pessoas ou sistemas, mas muitas dessas informações contextuais, essenciais para a compreensão de todos os envolvidos, são perdidas.

Portanto, a qualidade da comunicação influencia diretamente na qualidade dos *insights* das análises.

O ponto de partida é o dicionário de dados, que deve ser objetivo e não conter ambiguidades, logo, é possível se reconstruir todo o contexto em que a informação foi extraída, melhorando muito a qualidade das análises de dados construídas a partir dos dados coletados.

Problema típico da dicionarização

Existem algumas situações nas quais se tem uma planilha de dados, mas não se sabe fazer a interpretação desses dados, não compreendendo o que significam as linhas e ou colunas. Isso ocorre devido a problemas de comunicação e à falta de compartilhamento de significados.

Assim como um dicionário de idiomas, que explica a origem, os significados e os sinônimos das palavras, um dicionário de dados organiza o conhecimento necessário a respeito dos dados, como onde são coletados, suas características e funções etc.

Exemplo de dicionário de dados

Será utilizada uma hipotética entidade “consumidor”, com seus atributos, como exemplo de um dicionário de dados. Observe na tabela a seguir:

Nome da tabela	Nome do atributo	Conteúdo	Tipo	Formato	Faixa	Necessá-rio	PK/FK ou UK	Tabela referen-ciada por FK
Consumidor	Nome	Nome do consumidor	Varchar (200)	xxxxxxxxxx		S		
	Endereço	Endereço do consumidor	Varchar (200)	xxxxxxxxxxx				
	CPF	CPF do consumidor	CHAR (14)	999.999.999-99		S	pk	

Arquitetura de arquivos de dados

A arquitetura de dados define um conjunto padrão de produtos e ferramentas que uma organização usa para gerenciar dados.

Um fator importante no desempenho do banco de dados é o mecanismo de armazenamento utilizado pelo banco de dados e, mais especificamente, suas tabelas. Diferentes mecanismos de armazenamento podem fornecer maior desempenho em uma situação do que em outra.

Para uso geral, há dois concorrentes a serem considerados: o MyISAM, que é o mecanismo de armazenamento padrão do MySQL, ou InnoDB, que é um mecanismo alternativo integrado ao MySQL para bancos de dados de alto desempenho.

O que é bloqueio no MySQL?

Para proteger a integridade dos dados armazenados nos bancos de dados, o MySQL emprega o bloqueio. Bloquear significa simplesmente proteger os dados para que não sejam acessados. Quando um bloqueio é aplicado, os dados não podem ser modificados, exceto pela consulta que iniciou o bloqueio, ou seja, o bloqueio é um componente necessário para garantir a precisão das informações armazenadas. Cada mecanismo de armazenamento utiliza um método diferente de bloqueio. Dependendo de seus dados e de suas práticas de consulta, um mecanismo pode superar outro.

Existem dois tipos mais comuns de travamento empregados nos dois principais mecanismos de armazenamento.

O **bloqueio de tabela** é a técnica de bloquear uma tabela inteira quando uma ou mais células da tabela precisam ser atualizadas ou excluídas. O bloqueio de tabela é o método padrão empregado pelo mecanismo de armazenamento padrão MyISAM.

O **bloqueio em nível de linha** é o ato de bloquear um intervalo efetivo de linhas em uma tabela enquanto uma ou mais células dentro do intervalo são modificadas ou excluídas. O bloqueio de nível de linha é o método usado pelo mecanismo de armazenamento InnoDB e destina-se a bancos de dados de alto desempenho.

MyISAM *vs*. InnoDB

Comparando-se os dois mecanismos de armazenamento, chega-se ao ponto crucial do argumento sobre utilizar o InnoDB em vez de o MyISAM.

Um aplicativo ou *site* que contém uma tabela utilizada com frequência funciona excepcionalmente bem empregando-se o mecanismo de armazenamento InnoDB, resolvendo gargalos de bloqueio de tabela. No entanto, a questão de usar um sobre o outro é subjetiva, pois nenhum deles é perfeito em todas as situações. Existem pontos fortes e limitações para ambos os mecanismos de armazenamento.

O conhecimento íntimo da estrutura do banco de dados e das práticas de consulta é fundamental para selecionar o melhor mecanismo de armazenamento para suas tabelas.

MyISAM

Supera o InnoDB em tabelas grandes que requerem muito mais atividade de leitura do que atividade de gravação. As legibilidades do MyISAM superam o InnoDB porque bloquear a tabela inteira é mais rápido do que descobrir quais linhas estão bloqueadas na tabela. Quanto mais informações na tabela, mais tempo o InnoDB leva para descobrir quais não estão acessíveis. Se sua aplicação depende de tabelas grandes que não mudam os dados com frequência, então o MyISAM superará o InnoDB.

InnoDB

Por outro lado, o InnoDB supera o MyISAM quando os dados da tabela mudam com frequência. As alterações na tabela gravam mais dados do que a leitura de dados por segundo. Nessas situações, o InnoDB pode acompanhar grandes quantidades de solicitações mais facilmente do que bloquear a tabela inteira para cada solicitação.

Teoria dos conjuntos

A teoria dos conjuntos é o ramo da matemática que estuda os conjuntos, os quais, por sua vez, tratam do comportamento de uma coleção de elementos. Logo, toda coleção de elementos (números, objetos, figuras, pessoas, animais e tudo o que precisar ser ordenado), pode ser chamada de conjunto.

Segundo José Roberto Lessa, em artigo publicado no *site* InfoEscola,

A relação básica entre um conjunto e o elemento que o compõe é chamada de relação de pertinência, ou seja, definimos um conjunto quando existe uma regra que permite decidir se um elemento pertence ou não a ele. Se um elemento x pertence a um conjunto (ou coleção) A, dizemos que x pertence a. Formalmente escrevemos:

$$x \in A$$

E quando x não é um elemento deste conjunto, dizemos que x não pertence a A:

$$x \notin A$$

Com relação a bancos de dados, essa teoria é aplicada para extrair informações. As principais operações, chamadas de operações de conjuntos, aplicadas para a extração de dados são a união, a diferença e a intersecção.

União

Essa operação permite que todos os elementos dos conjuntos sejam unidos, ou seja, agrupados. No caso de um banco de dados, todos os dados das tabelas participantes da operação seriam agrupados.

Formalmente, seria esta a definição:

$$A \cup B = \{x : x \in A \text{ ou } x \in B\}$$

Na equação literal, tem-se que “A” união “B” é igual a (x), que pode ser definido como o conjunto universo, e “A” pertence ao conjunto universo juntamente com “B”, ou seja, na união, apenas os dados diferentes de “A” ou “B” pertencem ao conjunto universo.

Observe um exemplo no qual “A” é a tabela de produtos 1 e “B” é a tabela de produtos 2.

Tabela de produtos 1	
Produto	**Peso**
Melão	800 g
Morango	150 g
Maçã	120 g
Limão	200 g

Tabela de produtos 2	
Produto	**Peso**
Pinhão	100 g
Caqui	350 g
Melão	800 g
Morango	150 g

Observe agora o resultado da aplicação da união entre as tabelas:

Resultado da união	
Produto	**Peso**
Melão	800 g
Morango	150 g
Maçã	120 g
Limão	200 g
Pinhão	100 g
Caqui	350 g

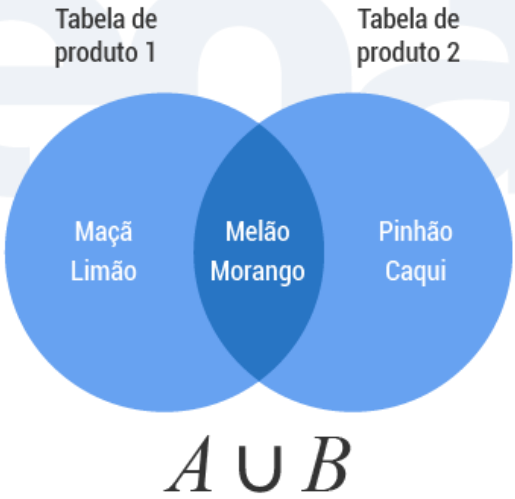

Figura 1 – Conjunto união

Intersecção

Essa operação permite que todos os elementos iguais dos conjuntos sejam selecionados.

Uma operação de intersecção pode extrair todos os dados da primeira tabela que estejam incluídos na segunda. Observe na figura a aplicação da operação de intersecção entre os dados dos conjuntos e também entre as tabelas.

Formalmente, seria esta a definição:

$$A \cap B = \{x : x \in A \text{ e } x \in B\}$$

Na equação literal, tem-se que “A” intersecção “B” é igual a (x), que pode ser definido como o conjunto universo, e “A” pertence ao conjunto universo juntamente com “B”, ou seja, na intersecção, todos os dados pertencem ao conjunto universo.

Veja outro exemplo, no qual “A” é a tabela de produtos 1 e “B” é a tabela de produtos 2.

Tabela de produtos 1	
Produto	**Peso**
Melão	800 g
Morango	150 g
Maçã	120 g
Limão	200 g

Tabela de produtos 2	
Produto	**Peso**
Pinhão	100 g
Caqui	350 g
Melão	800 g
Morango	150 g

Confira o resultado da aplicação da intersecção entre as tabelas:

Resultado da intersecção	
Produto	**Peso**
Melão	800 g
Morango	150 g

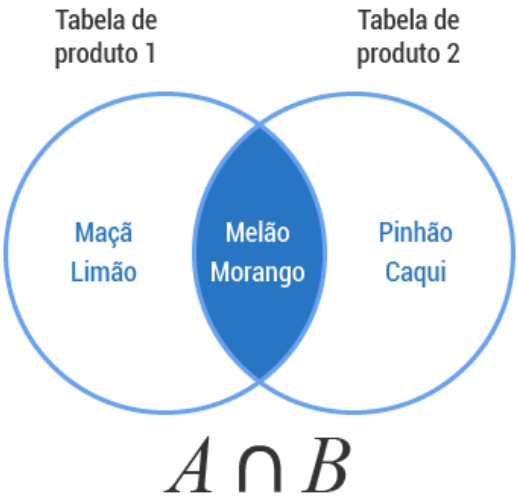

Figura 2 – Conjunto intersecção

Diferença (complemento)

Essa operação permite que todos os elementos diferentes dos conjuntos sejam selecionados. Uma operação de diferença pode extrair todos os dados da primeira tabela que não estejam incluídos na segunda. Nas figuras, observe a aplicação da operação da diferença entre os dados dos conjuntos e também entre as tabelas.

Formalmente, seria esta a definição:

$$A - B = \{x : x \in A \text{ e } x \notin B\}$$

Na equação literal, tem-se que “A” diferença “B” é igual a (x), que pode ser definido como o conjunto universo, e apenas “A” pertence ao conjunto universo, ou seja, na diferença, é apresentado o complemento de “A” em relação a “B”.

Observe um exemplo no qual “A” é a tabela de produtos 1 e “B” é a tabela de produtos 2.

Tabela de produtos 1	
Produto	**Peso**
Melão	800 g
Morango	150 g
Maçã	120 g
Limão	200 g

Tabela de produtos 2	
Produto	**Peso**
Pinhão	100 g
Caqui	350 g
Melão	800 g
Morango	150 g

Confira o resultado da aplicação da diferença entre as tabelas:

Resultado de produto 1	
Produto	**Peso**
Maçã	120 g
Limão	200 g

Resultado de produto 2	
Produto	**Peso**
Pinhão	100 g
Caqui	350 g

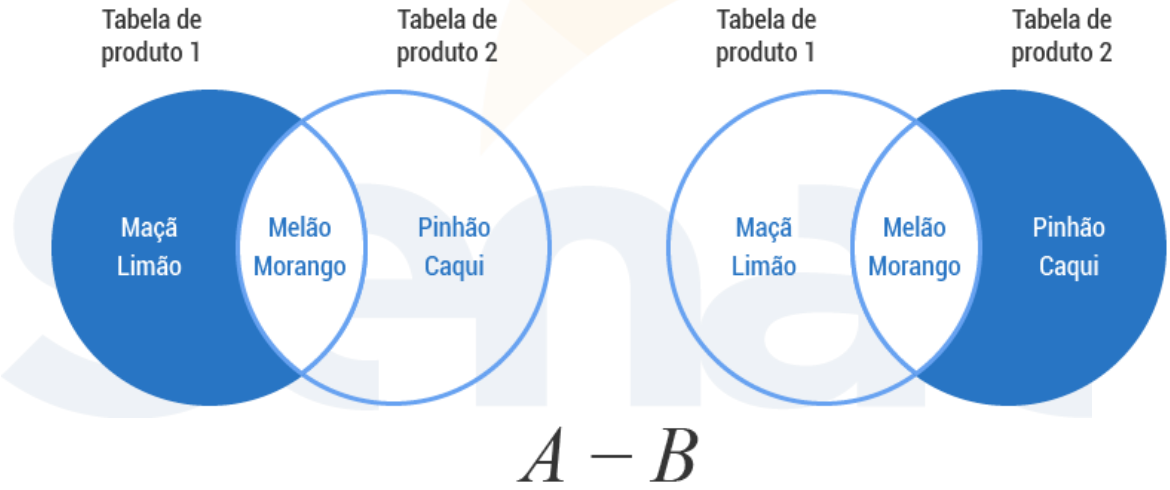

Figura 3 – Conjunto complemento

A SQL (*structured query language*, ou “linguagem de consulta estruturada”, em português) não trata diretamente das operações de diferença em *scripts*, logo, é preciso utilizar *joins* (junções) para fazer o tratamento.

Tipos de dados

Quando se fala em linguagens de programação e SQL, é preciso definir tipos de dados para as variáveis e operações, seguindo a mesma regra das línguas idiomáticas.

Em línguas, sabe-se que palavras são representadas por caracteres que vão de “A” a “Z”, e quando se fala em números, refere-se de “0” a “9”.

Com as palavras, podem ser feitas algumas atividades, tais como separação, acentuação e tonificação. Já com os números, é possível realizar cálculos matemáticos, equações e operações exponencias.

Com as palavras, podem ser feitas algumas atividades, tais como separação, acentuação e tonificação. Já com os números, é possível realizar cálculos matemáticos, equações e operações exponencias.

É por isso que se afirma que as linguagens são **tipadas**, ou **fortemente tipadas**, referindo-se aos tipos que cada dado pode receber.

Principais tipos de dados

A escolha do tipo de dado é um ponto extremamente importante no que se refere ao desenvolvimento e à construção das tabelas do banco de dados. Quando se escolhe o tipo de dado, está-se determinando como a informação deve ser inserida e quanto espaço ela poderá ocupar. Essa preocupação, que inicialmente pode parecer irrelevante, é uma iniciativa de prevenção de falhas do banco de dados.

Dados numéricos

- **Smallint** – É um número inteiro de 16 *bits*, que permite representar até 65.535 números, entre positivos e negativos.
- **Integer** ou **int** – É um número inteiro de 32 *bits*, que permite representar até 4.294.967.295 números, entre positivos e negativos.
- **BigInt** – É um número inteiro de 64 *bits*, que permite representar até 18.446.744.073.709.551.615 números, entre positivos e negativos.
- **Real** – É um número fracionário de precisão simples, ou seja, ocupa o tamanho máximo de 32 *bits*.
- **Decimal** – É um número fracionário com casas decimais exatas, ou seja, pode-se controlar exatamente a quantidade de casas inteiras e decimais de um número. A declaração decimal (10,2), por exemplo, indica que há dez dígitos de representação numérica, porém dois deles servirão para a parte fracionária do número.

Dados alfanuméricos

- **Char** – Faz alocação fixa da quantidade de caracteres a serem armazenados. Por exemplo, se for declarado um **char (200)** e, dentro desse campo, for escrita a palavra “João”, ela ocupará quatro posições de caracteres para ser armazenada. Entretanto, como foram alocadas 200 posições, será reservado em disco espaço para 200 caracteres, independentemente da quantidade que for ocupada.
- **Varchar** – Faz alocação variável da quantidade de caracteres a ser armazenada. Se for declarado um **varchar (200)**, por exemplo, e dentro dele for escrita a palavra “João”, das 200 posições reservadas para armazenamento, apenas quatro serão ocupadas. As outras 196 posições que estiverem sem conteúdo serão compactadas, reduzindo-se assim o espaço de armazenamento em disco.

Dados de tempo

- **Date** – É utilizado para armazenar datas (padrão americano: AAAA-MM-DD, por exemplo: 2008-01-11).
- **Time** – É utilizado para armazenar horas (padrão americano: HH:MM:SS, por exemplo: 22:54:32).

Mais informações sobre tipos de dados aplicados à SQL podem ser consultadas no conteúdo sobre a definição de dados, desta unidade curricular.

Integridade referencial

A integridade referencial está ligada a toda garantia de que um valor que aparece em uma relação para determinado conjunto de atributos também apareça para certo conjunto de atributos em outra relação.

Integridade referencial é um recurso padrão de *design* de banco de dados relacional que evita que usuários ou aplicativos insiram dados inconsistentes. A maioria dos bancos de dados relacionais tem regras de integridade referencial que você pode aplicar ao criar um relacionamento entre tabelas.

As regras de chaves mantêm a integridade referencial das tabelas por meio da ligação entre as colunas que as receberam e a validação de valores que poderão ser inseridos.

Comportamento dos campos compostos por chaves

Observe o comportamento dos campos que são compostos por chaves:

- **Chave primária (primary key)**: uma chave primária é uma regra implementada em uma coluna ou em um conjunto de colunas a fim de garantir que os valores contidos nelas sejam únicos, ou seja, que esses valores nunca se repitam.
- **Chave alternativa ou candidata (unique key)**: algumas colunas, que naturalmente apresentam característica de informação única, como CPF ou CNPJ, podem ficar de fora da regra de chave primária. Para garantir que os valores inseridos nessas colunas sejam únicos, pode-se implementar a regra de chave alternativa, que também garante a unicidade das informações na coluna (ou colunas) que recebem essa regra.
- **Chave estrangeira (foreign key)**: uma chave estrangeira é uma regra que pode determinar o comportamento de uma ou mais colunas, fazendo com que ela(s) referencie(m) as informações existentes em uma chave primária. Portanto, toda coluna que recebe regra de chave estrangeira deve ter recebido a regra de chave primária em sua tabela de origem, pois isso permite estabelecer relacionamentos entre tabelas do banco de dados.

Observe o exemplo da utilização de chaves:

Departamento	
4620	Fundamentos da computação
4622	Computação aplicada

Agora, veja nova tabela com a utilização do código da tabela departamento:

Funcionário			
ID	Nome	Código	Superior
1	Ana Vargas		
2	Avelino	4620	1
3	Demétrius	4622	2

A tabela departamento tem como chave primária (PK) o campo código. Na tabela empregado, o campo ID recebe regra de chave primária (PK). O campo código da tabela empregado recebe chave estrangeira (FK).

Há uma ligação entre o campo código da tabela departamento e o campo código da tabela empregado, a fim de apresentar a integridade referencial entre os campos de ambas as tabelas.

A integridade referencial evita que ocorrências sejam adicionadas a uma entidade se não houver nenhuma ocorrência relacionada de uma entidade e garante que, se uma ocorrência de uma entidade for excluída, as ocorrências relacionadas na entidade também serão excluídas.

Consequências da falta de integridade referencial

A falta de integridade referencial em um banco de dados pode levar ao retorno de dados incompletos, geralmente sem indicação de erro. Isso pode resultar na perda dos registros do banco de dados, porque eles nunca são retornados em consultas ou relatórios.

A falta de integridade referencial também pode originar resultados estranhos que aparecem em relatórios, como produtos sem uma empresa associada, ou gerar um problema maior, que é fazer com que os clientes não recebam os produtos pelos quais pagaram e aguardam a entrega.

A integridade referencial, portanto, é um item muito importante, que requer bastante atenção no estudo das tabelas e de suas respectivas referências.

Mecanismos para integridade referencial

Na SQL, uma chave estrangeira pode estar relacionada a qualquer combinação única de colunas na tabela referenciada. Se a lista de colunas referenciadas for omitida, a chave estrangeira estará relacionada à chave primária.

O padrão SQL fornece as seguintes ações de integridade referencial para exclusões.

- **Cascade**: a exclusão de um registro pode causar a exclusão de registros de chave estrangeira correspondentes. Por exemplo, se você exclui uma empresa, você também pode querer excluir o histórico de endereços da empresa.
- **Nenhuma ação (no action)**: alternativamente, você pode proibir a exclusão de um registro se houver registros de chave estrangeira dependentes. Por exemplo, se você vendeu produtos para uma empresa, você pode querer evitar a exclusão do registro da empresa.
- **Defina nulo (set null)**: A exclusão de um registro pode fazer com que as chaves estrangeiras correspondentes sejam definidas como nulas. Por exemplo, se há uma substituição de aeronave em um voo, você pode querer anular algumas atribuições de assento (esses passageiros devem então solicitar outras designações de assento).
- **Conjunto padrão (set default)**: você pode definir uma chave estrangeira com um valor padrão em vez de nulo na exclusão de um registro.

Encerramento

Neste material, você aprendeu sobre umas das mais importantes técnicas para construir um banco de dados relacional.

A compreensão e o estudo deste material são de suma importância para que a construção do banco de dados seja realizada o mais corretamente possível.

É bom lembrar que um banco de dados nunca estará finalizado em sua totalidade, pois novos recursos, novas leis e funcionalidades podem ser incorporados ao projeto existente, devendo-se refazer todos os processos para a inserção desses novos requisitos.