



BANCOS DE DADOS

AULA 2



Prof. Ricardo Sonaglio Albano
Profª. Silvie Guedes Albano



CONVERSA INICIAL

Modelagem de Bancos de Dados

Aqui, vamos abordar a modelagem relacional, também conhecida como modelagem lógica, convertendo o Modelo Entidade-Relacionamento (MER) para o modelo relacional e estudando os aspectos de refinamento no processo de normalização e conversão para o modelo físico. Vamos apresentar também outros modelos que podem ser aplicados em Bancos de Dados relacionais.

Daremos início ao estudo da linguagem *Structured Query Language* (SQL), apresentando conceitos importantes para que você possa fortalecer o seu conhecimento teórico e aplicá-lo na prática. Também vamos dar prosseguimento ao nosso projeto de Bancos de Dados, que iniciamos na etapa anterior, desenvolvendo um passo a passo do modelo relacional e convertendo-o para o projeto físico.

Bons estudos e sucesso!

TEMA 1 – MODELO LÓGICO OU RELACIONAL

O modelo relacional é um modelo de dados adequado para um Sistema Gerenciador de Bancos de Dados (SGBD) específico, com base no princípio de que todos os dados estão armazenados em tabelas.

Como já mencionado, o pesquisador Edgar Codd detectou que seria possível aplicar a teoria dos conjuntos (álgebra relacional) nas estruturas de dados, possibilitando a realização de operações de seleção, projeção, atribuição, entre outras.

O modelo relacional foi aprimorado, por Chris Date e Hugh Darwen, como um modelo geral de dados. No “Terceiro Manifesto”, de 1995, os dois autores da atualização mostraram como o modelo relacional pode ser estendido com características de orientação a objeto, sem comprometimento de seus princípios fundamentais.

A modelagem relacional permite ao projetista criar um modelo lógico consistente da informação que será armazenada. Esse modelo lógico pode ser refinado através de um processo de normalização. Dessa forma, o Bancos de Dados construído, baseado no modelo relacional, apresenta-se inteiramente normalizado.



Em Bancos de Dados relacionais, a linguagem padrão utilizada é a *Structured Query Language* (Linguagem de Consulta Estruturada) – ou, como é comumente conhecida, SQL.

1.1 Modelo de dados relacional

Representa os dados contidos em um Bancos de Dados através de relações. Tais relações contêm as informações sobre as entidades representadas e seus relacionamentos. O modelo relacional se baseia no conceito de matrizes, em que as chamadas linhas (presentes nas matrizes) recebem a designação de tuplas, e em que as colunas são chamadas de atributos. Os nomes das relações e dos atributos são de fundamental importância para a nossa compreensão entre o que está sendo armazenado, onde está sendo armazenado e qual a relação existente entre os dados armazenados.

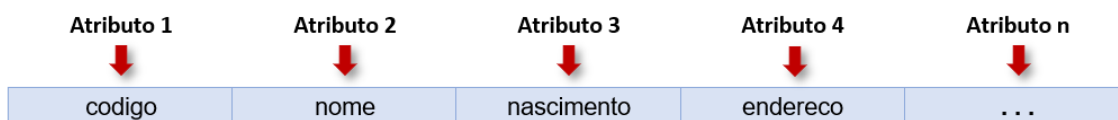
Um Bancos de Dados relacional nada mais é do que um repositório de arquivos de dados computadorizados, sendo composto de:

- Atributos;
- Domínio;
- Tuplas; e
- Relações.

1.1.1 Atributo

É o nome que identifica o dado que está armazenado em um Bancos de Dados, ou seja, o nome de cada coluna presente na tabela.

Figura 1 – Representação de atributo em uma relação



Fonte: Albano, 2022.

1.1.2 Domínio

Refere-se ao tipo de dado (inteiro, caractere, decimal, data, entre outros) que define o tamanho e os valores que cada atributo poderá receber. O domínio







também consiste na definição de padrões de validação que fazem parte da restrição de atributos e padrões, como veremos na sequência deste material.

Exemplos:

- Atributo “nome”, domínio cadeia de caracteres de comprimento 100:
`nome varchar(100);`
- Atributo “nascimento”, domínio data: `nascimento date.`

Figura 2 – Representação de domínio em uma relação



20001001	João da Silva	01/01/1980	Sete de setembro, 1000
			
Inteiro	Cad. caracteres	Data	Cadeia de caracteres

Fonte: Albano, 2022.

1.1.3 Tupla

A tupla é o conjunto horizontal de dados presentes nos atributos, ou seja, refere-se ao registro ou linha presente em uma tabela. Perceba que na representação a seguir existem duas tuplas (registros) assinaladas.

Figura 3 – Representação de tupla em uma relação

Tupla 1 	20001001	João da Silva	01/01/1980	Sete de setembro, 1000
Tupla 2 	20002100	Maria de Souza	30/01/1985	XV novembro, 10

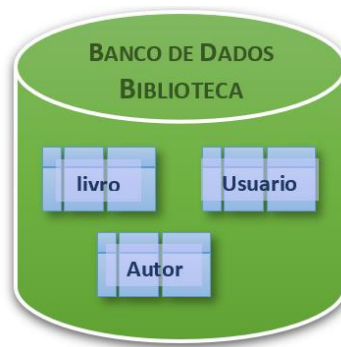
Fonte: Albano, 2022.

1.1.4 Relação

Conjunto de tuplas que formam uma relação (tabela). Cada relação está vinculada a um objeto específico dentro do Bancos de Dados. Por exemplo, no Bancos de Dados de uma biblioteca, encontramos diversas relações, tais como livro, usuário, empréstimo e autor.



Figura 4 – Representação de um Bancos de Dados



Fonte: Albano, 2022.

Também podemos dizer que uma **relação é composta de atributos (colunas) e tuplas (linhas)**. Logo, as tabelas são uma divisão lógica do Bancos de Dados, pois, fisicamente, o Bancos de Dados é um arquivo único.

Figura 5 – Representação da relação



Figura 6 – Conceitos fundamentais do modelo relacional

CLIENTE			
codigo	nome	nascimento	endereço
20001001	João da Silva	01/01/1980	Sete de setembro, 1000
20002100	Maria de Souza	30/01/1985	XV novembro, 10
19992009	Paulo de Gil	01/01/1980	General Osório, 102
20039564	Maria de Souza	12/10/1990	Getúlio Vargas, 200
20093212	Ana de Jesus	26/09/1992	Sete de setembro, 1000

Domínios (tipo de dados)

inteiro	cad. caracteres	data	cadeia de caracteres
---------	-----------------	------	----------------------

Fonte: Albano, 2022.

Perceba que a terminologia na modelagem relacional é diferente da terminologia usada na modelagem conceitual, tema que já estudamos. **Antes, trabalhamos com os termos entidade, campo e registro. Já no modelo relacional, utilizamos a terminologia relação, atributo e tupla.**

1.2 Álgebra relacional



A principal proposição do modelo relacional é que todos os dados são representados como relações matemáticas, as quais são baseadas na teoria dos conjuntos. No modelo matemático, a análise dos dados é feita em uma lógica de predicados de dois valores (sem o valor nulo), o que significa que existem dois possíveis valores para uma proposição, isto é, verdadeiro ou falso. O tratamento dos dados é feito pelo cálculo relacional ou pela álgebra relacional.

A álgebra relacional é uma linguagem de consulta formal, ou seja, o usuário informa as instruções ao SGBD para que ele realize a sequência de operações na base de dados.

Apesar de serem conceitos matemáticos, eles correspondem basicamente aos conceitos tradicionais de Bancos de Dados. Dessa forma, uma relação é similar ao conceito de tabela e uma tupla é similar ao conceito de linha.

1.2.1 Conhecendo os Operadores Relacionais


A álgebra relacional define operadores para atuar nas relações, com o objetivo de manipular uma ou mais relações como entrada e produzir uma nova relação como saída (resultado).

Os operadores relacionais são classificados da seguinte forma:

- Operadores unários – seleção (*where*) e projeção (*select*);
- Operadores binários – união (*union*), diferença (*difference*) e produto cartesiano (*cross join*);
- Operadores derivados – intersecção (*intersect*), junção (*join*) e divisão (*divide*);
- Operadores especiais – renomeação e atribuição (*assignment*), operadores criados especificamente para serem aplicados ao modelo de dados relacional.

Neste material, vamos estudar a aplicação dos seguintes operadores: atribuição, seleção e projeção.

1.2.1.1 Operador de atribuição (*Assignment*)

Esse operador tem o objetivo de redirecionar o resultado da operação para uma nova relação. Representamos o operador de atribuição com o símbolo .

1.2.1.2 Operador de seleção (*Where*)



Seleciona um subconjunto em uma relação, ou seja, **seleciona os dados de acordo com uma condição pré-estabelecida, conhecida como predicado.** Resumindo, **é a aplicação de um filtro.** Representamos a seleção através do símbolo da letra grega sigma, isto é, **σ .**

Os operadores de comparação e lógicos são:

- Operadores de comparação: \neq , $=$, $<$, $=>$, $>$, $=$
- Operadores lógicos: \wedge (E, *and*), \vee (OU, *or*), \neg (não, *not*)

Exemplo: na relação Cliente, estão armazenados os dados referentes aos clientes de um restaurante. Suponha que precisamos somente dos dados dos clientes do gênero masculino.

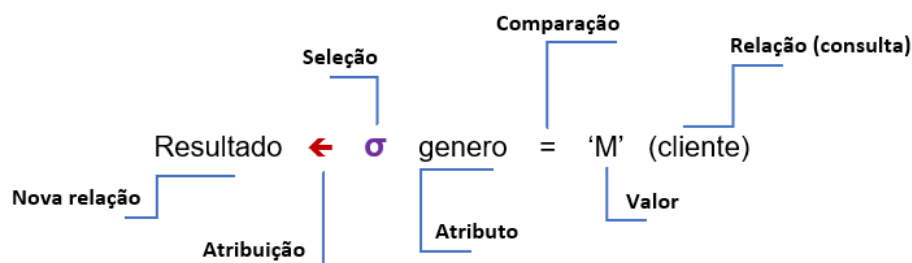
Instrução de seleção:

Resultado $\leftarrow \sigma$ genero = 'M' (cliente)

Onde:

- Resultado: nome da nova relação que conterà o resultado da consulta
- \leftarrow : símbolo de atribuição
- σ : símbolo da seleção
- genero: nome do atributo que contém a informação
- =: operador de comparação
- 'M': valor a ser comparado
- (cliente): nome da relação onde será realizada a consulta

Figura 7 – Desmembrando o exemplo da fórmula de seleção



Fonte: Albano, 2022.



Figura 8 – Exemplo da relação Cliente

CLIENTE				
codigo	nome	nascimento	endereço	genero
20001001	João da Silva	01/01/1980	Sete de setembro, 1000	M
20002100	Maria de Souza	30/01/1985	XV novembro, 10	F
19992009	Paulo de Gil	01/01/1980	General Osório, 102	M
20039564	Maria de Souza	12/10/1990	Getúlio Vargas, 200	F
20093212	Ana de Jesus	26/09/1992	Sete de setembro, 1000	F

Fonte: Albano, 2022.

Figura 9 – Tabela Resultado (operador seleção)

RESULTADO				
codigo	nome	nascimento	endereço	genero
20001001	João da Silva	01/01/1980	Sete de setembro, 1000	M
19992009	Paulo de Gil	01/01/1980	General Osório, 102	M

Fonte: Albano, 2022.

A nova relação, chamada Resultado, contém somente os clientes do gênero 'M' (masculino), conforme a condição definida na instrução descrita anteriormente.

1.2.1.3 Operador de Projeção (*Select*)

Retorna determinadas colunas de uma relação, sendo representado através do símbolo da letra grega pi, isto é, π .

Exemplo: com base na mesma relação Cliente, utilizada no exemplo anterior, suponha que desejamos obter somente o nome dos clientes.

Instrução de seleção:

π nome (cliente)

Onde:

- π : símbolo da projeção
- nome: nome do atributo que contém a informação
- (cliente): nome da relação em que ocorrerá a consulta



Figura 10 – Tabela resultante (operador projeção)

nome
João da Silva
Maria de Souza
Paulo de Gil
Maria de Souza
Ana de Jesus

Fonte: Albano, 2022.

Analisando o resultado obtido, notamos que a nova relação contém somente a coluna “nome”, conforme solicitação da instrução definida na fórmula de projeção.

Vale salientar também que é possível ainda realizar uma projeção dentro de uma seleção. Por exemplo, selecionar o nome de todos os clientes do gênero feminino.

Instrução: π nome (σ sexo = ‘F’ (cliente))

Por ordem de precedência (primeiro o conteúdo presente dentro dos parênteses), a seleção é executada, retornando as tuplas correspondentes aos clientes do gênero feminino (‘F’). Depois, é realizada a projeção solicitada, isto é, a apresentação apenas do conteúdo do atributo nome.

Figura 11 – Tabela resultante (projeção e seleção)

nome
Maria de Souza
Maria de Souza
Ana de Jesus

Fonte: Albano, 2022.

1.2.1.4 Comparação entre os Operadores Projeção e Seleção

O operador de projeção aplica filtros sempre na vertical, ou seja, apenas filtra o conteúdo dos atributos. Dessa forma, podemos definir:

- O resultado será uma relação com quantidade igual ou menor de atributos presentes na relação;
- Todas as tuplas estarão relacionadas.

O operador de seleção, por sua vez, executa filtros horizontais, ou seja, filtra o conteúdo presente nas tuplas. Assim, podemos afirmar:



- O resultado é uma relação com quantidade igual ou menor ao total de tuplas presentes na relação;
- Os atributos permanecem idênticos.

Observe que o *select* é uma projeção do resultado que pode ter passado ou não por uma seleção (*where*).

É importante ressaltar que todos os conceitos apresentados serão utilizados posteriormente no estudo dos comandos SQL.

1.3 Restrições de integridade

O princípio básico do modelo relacional é o princípio da informação, isto é, toda informação é representada por valores contidos nas relações (tabelas). Se um atributo é referenciado em mais de uma relação (chave estrangeira), ele deve apresentar o mesmo domínio (tipo de dado), o que torna o atributo dependente de outro.

As restrições de integridade são divididas da seguinte forma.

- Integridade referencial: presença de chave estrangeira, isto é, um atributo que faz relação com a chave primária de outra relação (tabela).
- Restrição de unicidade (*unique*): para utilização, no caso de chaves candidatas, garantindo a integridade, ou seja, a ausência de valores repetidos, como o atributo CPF.
- Restrições de atributos e padrões:
 - Nenhuma chave primária ou estrangeira poderá conter valor nulo (*not null*);
 - Valor padrão (*default*): definição de um valor pré-definido quando não for informado (por exemplo, se não for informado o estado, o UF será PR);
 - Validação (*check*): aplicação de regras de validação. Por exemplo, a nota do aluno não poderá ser inferior a 0 (zero) ou superior a 10; o atributo gênero não poderá ser diferente de 'M' ou 'F'.
- Integridade de chave: baseia-se no princípio de que toda tupla apresenta um conjunto de atributos que a identifica de maneira única na relação, isto é, com a presença de chave primária.

1.3.1 Chaves



Uma relação (tabela) deve ter a capacidade de identificar cada uma das tuplas separadamente, ou seja, cada tupla deve ter um ou mais atributos que sejam diferentes de outra linha. Para realizar essa diferenciação, utilizamos o conceito de chaves.

As relações (tabelas) interagem umas com as outras por meio de chaves. Tal relacionamento é realizado por meio de chaves estrangeiras.

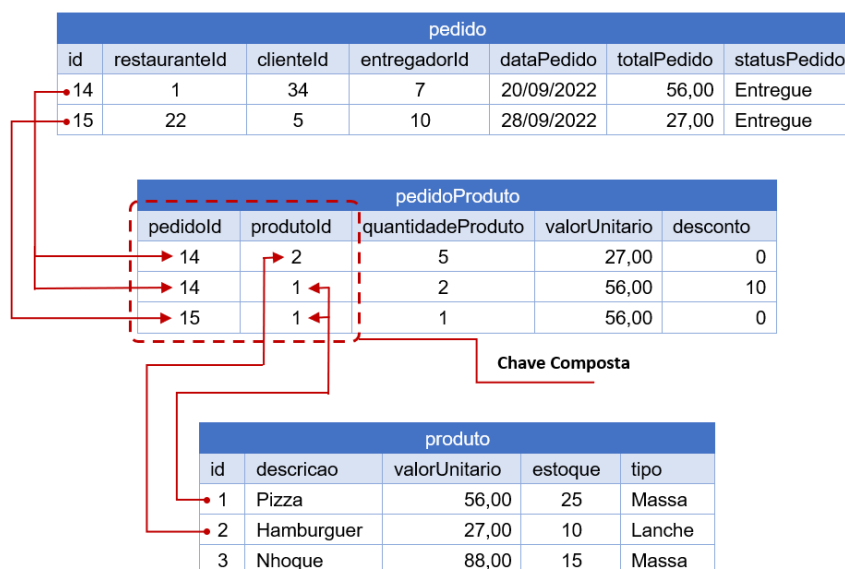
Já estudamos a importância da utilização de chaves primárias e estrangeiras. Agora, vamos apresentar um novo tipo de chave: a chave composta.

1.3.1.1 Chave composta

A sua principal característica é conter mais de um atributo em sua composição. Normalmente, a chave composta deriva de uma relação n:n (muitos para muitos), em que é necessário gerar uma entidade associativa (modelo conceitual). Essa nova relação produz agregação das chaves primárias de cada relação envolvida.

Para exemplificar a aplicação de uma chave composta, vamos utilizar as entidades obtidas através do modelo conceitual de nosso estudo de caso.

Figura 12 – Exemplo de chave composta



Fonte: Albano, 2022.

É bastante comum usar a chave Surrogate Key (chave artificial). Isso é devido a característica de ser um elemento único, o que simplifica a

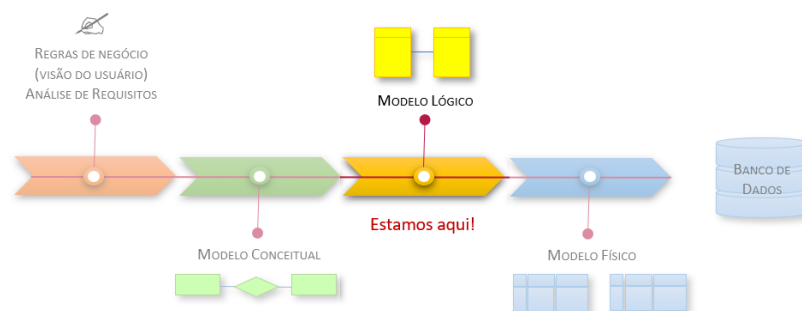


implementação das chaves no modelo físico. Estamos falando do famoso “id” (identificador).

1.4 Estudo de caso

Concluídos os estudos sobre o modelo lógico ou relacional, vamos para a próxima etapa do projeto de Bancos de Dados, que é conversão do modelo conceitual para o modelo relacional.

Figura 13 – Localização no projeto de Bancos de Dados



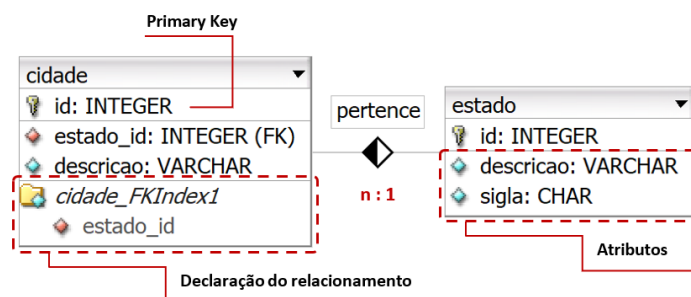
Fonte: Albano, 2022.

Para desenvolver a modelagem do modelo relacional, escolhemos a ferramenta DBDesigner. Vale salientar que existem diversas ferramentas disponíveis para a mesma finalidade.

O DBDesigner apresenta alguns padrões de notação que talvez provoquem uma pequena diferença em alguns detalhes do modelo. Ressaltamos as seguintes diferenças.

- A atribuição do nome para as chaves estrangeiras difere um pouco. A ferramenta inclui automaticamente o nome da relação, um *underline* (_) e o nome do atributo (por exemplo: restaurante_id).
- Em relacionamentos que apresentem cardinalidade n:n (entidades associativas), a ferramenta gera automaticamente as chaves compostas, o que acaba dificultando o uso da chave auxiliar (Surrogate Key).
- Sobre a notação de cardinalidade, a ferramenta fornece algumas variações na notação.

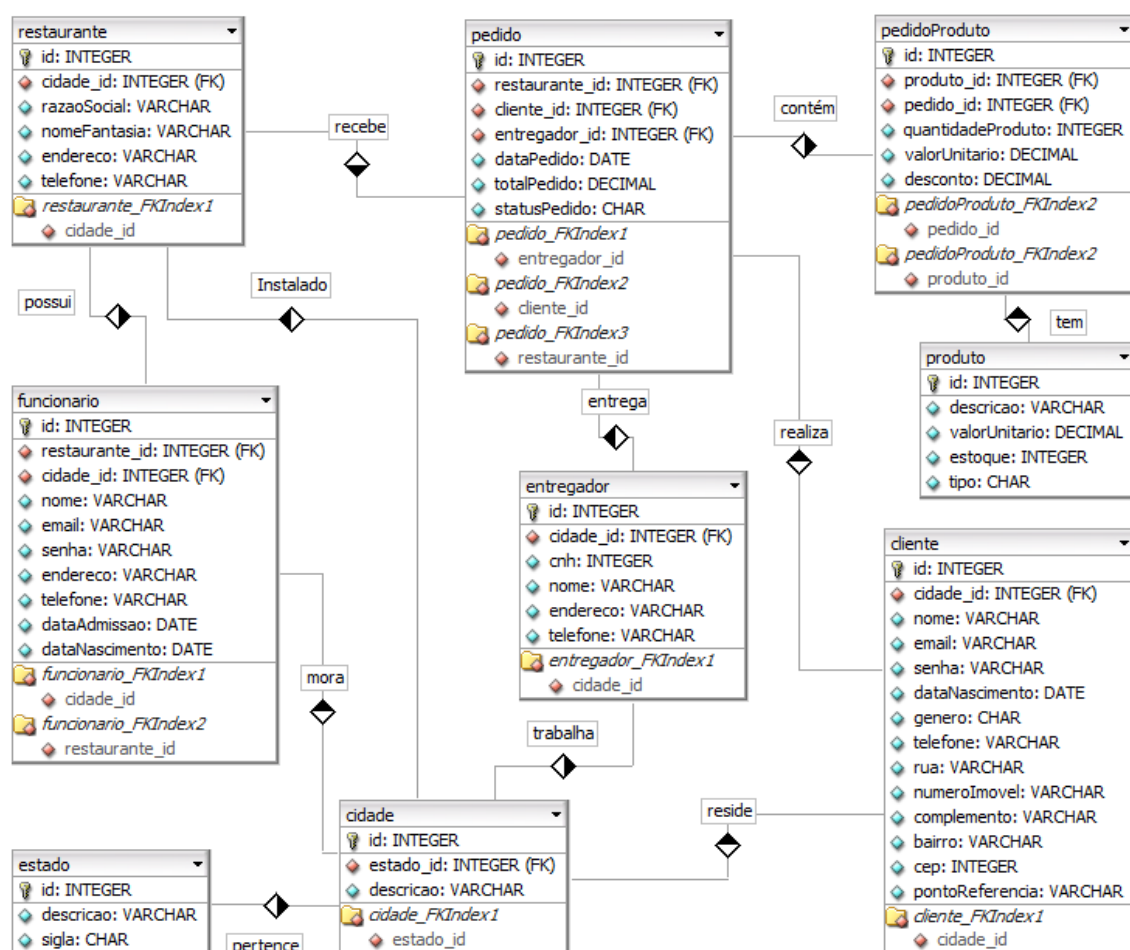
Figura 14 – Notação de relacionamento no DBDesigner



Fonte: Albano, 2022.

Com base no modelo conceitual, construímos o modelo lógico apresentado na figura a seguir.

Figura 15 – Modelo lógico do estudo de caso (ferramenta DBDesigner)



Fonte: Albano, 2022.

Observação: os tipos de dados utilizados serão explicados em um próximo tópico, quando veremos a diferença entre os atributos *char* e *varchar*.

TEMA 2 – NORMALIZAÇÃO



A normalização consiste em definir o formato lógico adequado para as relações presentes no modelo. O objetivo é minimizar o espaço utilizado pelos dados, garantindo a integridade e a confiabilidade das informações.

Nessa fase, é realizada uma varredura em todos as relações do modelo, buscando por inconsistências, por meio da aplicação de Formas Normais (FN).

As Formas Normais abarcam um conjunto de regras e restrições que devem ser aplicadas sobre os dados. A normalização estará completa quando o modelo tiver passado por todas as Formas Normais.

Vale salientar que a passagem de uma FN para outra é feita com base no resultado obtido na etapa anterior – ou seja, em FN2, necessariamente passamos por FN1.

A normalização é composta de cinco Formais Normais. Contudo, na prática, aplicamos somente as três primeiras Formas Normais, uma vez que a sua aplicação é suficiente para resolver quase que a totalidade dos problemas de modelagem.

Podemos citar como benefícios da aplicação do processo de normalização: minimização de redundâncias e de inconsistências nos dados, bem como a melhoria na manipulação do Bancos de Dados.

2.1 Formas Normais na prática

Para entendermos na prática o processo de normalização, vamos analisar a relação (tabela) Funcionário.

Figura 16 – Exemplo da relação Funcionário

FUNCIONARIO			
nome	cargo	departamento	habilidades
João da Silva	Gerente	RH	Gestão, comunicação
Maria de Souza	Desenvolvedor Python	TI	Python avançado, proatividade
Paulo de Gil	Suporte TI	Suporte	Proatividade, comunicação
Maria de Souza	Analista de sistema	TI	Gestão de projeto, agilidade
Ana de Jesus	Suporte TI	Suporte	Comunicação
Pedro da Silva	Vendedor	Vendas	Comunicação
Luísa de Souza	Vendedora	Vendas	Comunicação

Fonte: Albano, 2022.

Por meio da análise da relação Funcionário, podemos identificar os seguintes problemas (anomalias).



- O que aconteceria com o departamento de RH se você excluísse o funcionário João da Silva? O departamento deixaria de existir. Esse tipo de evento é chamado de anomalia de exclusão.
- E se o nome do departamento de vendas fosse alterado? Você seria obrigado a alterar todas as tuplas onde constam o departamento. E se você esquecesse de alguma? Esse tipo de situação é uma anomalia de alteração.
- Se um novo funcionário for incluído, o que aconteceria se o cargo fosse descrito simplesmente como 'Desenvolvedor'? Teríamos um cargo diferente de 'Desenvolvedor Python', mas os dois têm a mesma função: desenvolvedor. Esse tipo de ocorrência é conhecido como anomalia de inclusão.

2.1.1 Primeira Forma Normal – 1FN

Uma relação (tabela) está na 1FN apenas quando todos os domínios básicos apresentam valores atômicos, isto é, não armazenam grupos repetitivos. Para atingir essa Forma Normal, devemos eliminar os grupos de repetição.

Regras:

- Cada tupla (linha) de dados deve ter um identificador único;
- Cada tupla de dados deve conter dados atômicos (não repetitivos).

Ao aplicar a 1FN na relação Funcionário, podemos fazer as seguintes considerações.

- Não existe um identificador único capaz de representar cada tupla, de modo que o atributo 'id' será incluído como chave primária da relação (tabela).
- O conteúdo armazenado no atributo 'departamento' apresenta dados repetitivos. Dessa forma, esse atributo deve ser transformado em uma nova relação (tabela). Tal procedimento, além de evitar a ocorrência de informações inconsistentes (exemplo do desenvolvedor e desenvolvedor Python), também reduz espaço, pois o dado será armazenado somente uma vez na nova relação Departamento, sendo referenciado apenas na relação Funcionário. É importante destacar que a cardinalidade entre a relação Departamento e a relação Funcionário resulta em 1:n, ou seja, um



funcionário trabalha em apenas um departamento, mas em um departamento podem trabalhar um ou vários funcionários.

- O mesmo procedimento será aplicado no atributo 'cargo', que se tornará uma nova relação, pelos motivos já explicados.
- Analise o atributo 'habilidades'. Repare que ele apresenta diversas informações na mesma tupla, de modo que se trata de um atributo multivalorado. Avaliando esse atributo, podemos declarar que um funcionário pode apresentar uma ou várias habilidades. Além disso, uma habilidade pode pertencer a nenhum ou diversos funcionários, resultando em uma cardinalidade n:n. Como sabemos, toda cardinalidade n:n gera uma nova relação – no caso, a relação Cargo.

Após a aplicação da 1FN, nosso exemplo leva à representação da figura a seguir.

Figura 17 – Resultado após a aplicação da Primeira Forma Normal (1FN)

FUNCIONARIO				DEPARTAMENTO		CARGO	
id	nome	departamento	cargo	id	descricao	id	nome
1	João da Silva	1	1	1	RH	1	Gerente
2	Maria de Souza	2	2	2	TI	2	Desenvolvedor python
3	Paulo de Gil	3	3	3	Suporte	3	Suporte TI
4	Maria de Souza	2	4	4	Vendas	4	Analista de sistema
5	Ana de Jesus	3	3			5	Vendedor(a)
6	Pedro da Silva	4	5				
7	Luísa de Souza	4	5				

HABILIDADE		FUNCIONARIOHABILIDADE		
id	descricao	id	funcionario	habilidade
1	Gestão	1	1	1
2	comunicação	2	1	2
3	Python avançado	3	2	3
4	proatividade	4	2	4
5	Gestão de projeto	5	3	4
6	agilidade	6	3	2
		7	4	5
		8	4	6
		9	5	2
		10	6	2
		11	7	2

Fonte: Albano, 2022.

Perceba que, após a aplicação de 1FN, a partir da relação de origem (Funcionário), surgiram mais quatro relações (Cargo, Departamento, Habilidades



e FuncionarioHabilidade). Com essas novas relações, o modelo se torna mais íntegro, com baixa redundância de dados.

Lembre-se de que estamos trabalhando com as restrições de integridade. Para tanto, fazemos uso da chave estrangeira. Inclusive, estamos excluindo atributos repetidos e multivalorados das relações (redução de redundância).

2.1.2 Segunda Forma Normal – 2FN

Uma relação (tabela) é considerada normalizada em sua 2FN se, e somente se, já houver sido refinada pela 1FN, e se todos os atributos não-chaves apresentarem total dependência da chave primária (dependente de toda a chave e não apenas parte dela).

Regras:

- Estar em conformidade com a 1FN, ou seja, o modelo já deve ter passado pelo primeiro processo de refinamento;
- Não apresentar nenhum atributo dependente de parte da chave primária.

Para esclarecer a 2FN de forma prática, vamos analisar a relação Venda, fazendo as adequações e os refinamentos necessários.

Figura 18 – Exemplo da relação Venda

VENDA					
pedidoId	produtoId	descricao	quantidade	valorUnitario	total
1	1	HD externo 16g	2	500,00	1.000,00
1	2	Monitor 32 pol	1	1.200,00	1.200,00
2	3	Notebook i9	2	4.000,00	8.000,00
2	1	HD externo 16g	1	500,00	500,00
3	3	Notebook i9	1	4.000,00	4.000,00

Chave Primária Composta

Fonte: Albano, 2022.

Considerações:

- A relação Venda apresenta uma chave primária composta que envolve os atributos 'pedidoId' e 'produtoId', pertencentes a outras relações.
- Perceba que o atributo nome do produto (descrição) depende do código do produto (produtoId). Porém, não depende do número do pedido (pedidoId), que também faz parte da chave primária. Portanto, isso indica que não está em conformidade com a 2FN. Essa anomalia gera problemas futuros de manutenção dos dados. Quando ocorrer alguma



alteração no nome do produto, será necessário realizar uma alteração idêntica em todos os registros que pertencem a relação Venda.

- A solução é remover o atributo identificado da relação Venda, criando uma nova relação com a função de armazená-lo de forma adequada, ou seja, iremos remover o atributo 'descricao' da relação Venda, criando uma relação chamada de Produto.

Após a aplicação da 2FN, o nosso exemplo resultará na representação da figura a seguir.

Figura 19 – Resultado após a aplicação da Segunda Forma Normal (2FN)

VENDA					PRODUTO	
pedidoId	produtoId	quantidade	valorUnitario	total	produtoId	descricao
1	1	2	500,00	1.000,00	1	HD externo 16g
1	2	1	1.200,00	1.200,00	2	Monitor 32 pol
2	3	2	4.000,00	8.000,00	3	Notebook i9
2	1	1	500,00	500,00		
3	3	1	4.000,00	4.000,00		

Fonte: Albano, 2022.

Perceba que a 2FN refere-se às relações com chave primária composta. Caso o modelo relacional não tenha nenhuma relação com a chave composta, automaticamente estará na 2FN.

2.1.3 Terceira Forma Normal – 3FN

Um modelo é normalizado pela 3FN se já tiver passado pelo refinamento da 2FN, e se todos os atributos não-chaves forem dependentes não transitivos da chave primária, ou seja, se cada atributo for funcionalmente dependente apenas dos atributos componentes da chave primária, ou se todos os seus atributos não-chaves forem independentes entre si.

Regras:

- Conformidade com a 2FN; e
- Não possuir nenhum atributo dependente de outros atributos que não sejam chaves da relação.

Para exemplificar, vamos prosseguir com a análise da relação Venda.



Figura 20 – Exemplo da relação Venda (2FN)

VENDA				
pedidoId	produtoId	quantidade	valorUnitario	total
1	1	2	500,00	1.000,00
1	2	1	1.200,00	1.200,00
2	3	2	4.000,00	8.000,00
2	1	1	500,00	500,00
3	3	1	4.000,00	4.000,00

Fonte: Albano, 2022.

Considerações: avaliando o atributo 'total', percebemos que ele é derivado dos atributos 'quantidade' e 'valorUnitario', ou seja, o atributo 'total' origina-se de um cálculo, o que o torna dependente de dois atributos que não são definidos como chaves na relação Venda. Dessa forma, a regra da 3FN acaba sendo ferida. Logo, a solução é eliminar o atributo 'total' da relação Venda.

É importante ressaltar que, como boa prática, se evite criar atributos derivados nas relações do modelo relacional.

Figura 21 – Resultado após a aplicação da Terceira Forma Normal (3FN)

VENDA			
pedidoId	produtoId	quantidade	valorUnitario
1	1	2	500,00
1	2	1	1.200,00
2	3	2	4.000,00
2	1	1	500,00
3	3	1	4.000,00

Fonte: Albano, 2022.

Como comentamos anteriormente, apesar da existência da quarta e da quinta Forma Normal, apenas as três primeiras são realmente aplicadas.

As Formas Normais são importantes instrumentos para resolver, antecipadamente, problemas na estrutura do Bancos de Dados. Após a normalização, as estruturas dos dados são projetadas para eliminar, de forma automatizada, qualquer inconsistência e redundância dos dados, evitando assim problemas na manutenção e operacionalização do sistema.

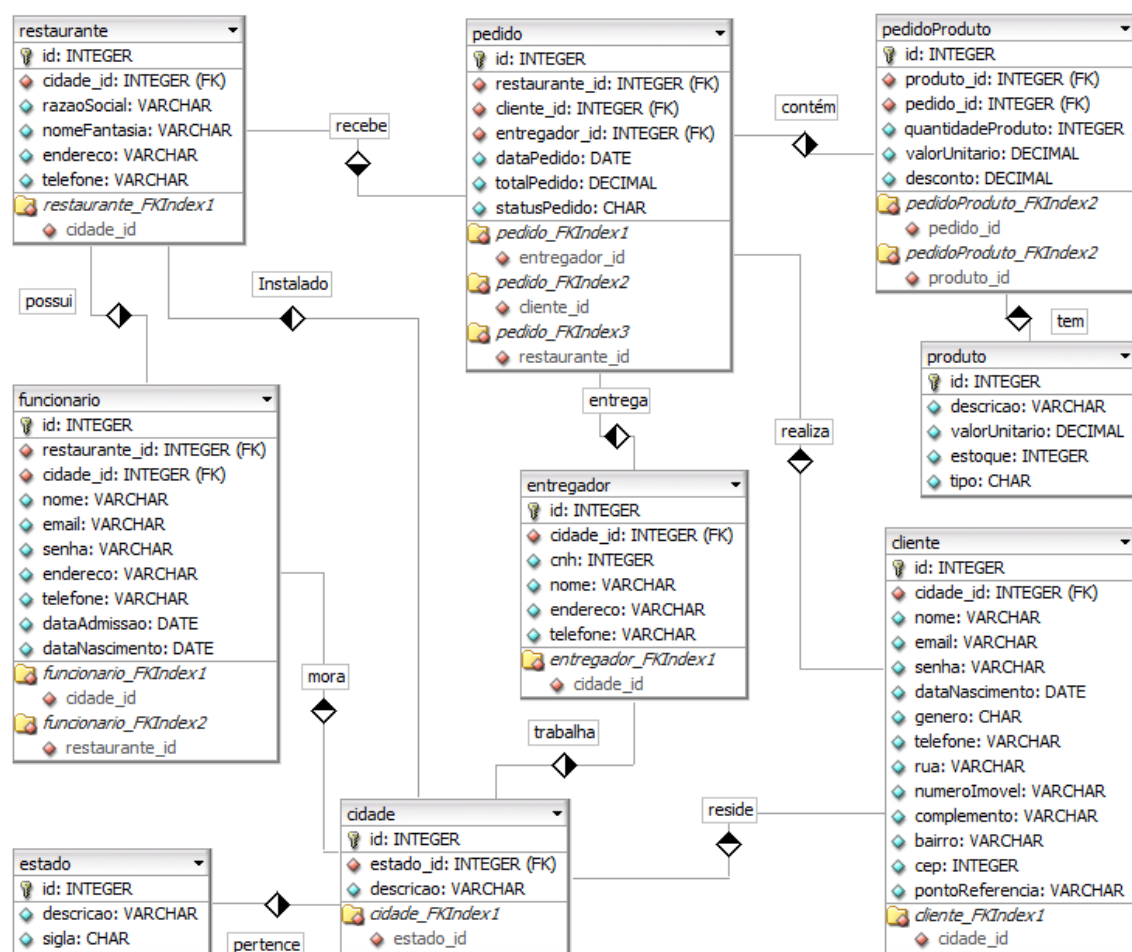
2.2 Estudo de caso



Você está pronto para prosseguir em nosso estudo de caso? Essa é a última etapa para a conclusão de nosso modelo relacional. Assim, munidos do conhecimento apresentado sobre a normalização, vamos praticar!

A ideia é retornar ao modelo relacional que desenvolvemos inicialmente, analisando cuidadosamente cada uma das relações, procurando por irregularidades e realizando um processo de normalização (refinamento), conforme estudamos neste tema.

Figura 22 – Modelo lógico do estudo de caso (ferramenta DBDesigner)



Fonte: Albano, 2022.

Considerações:

- Analisando as relações Funcionário e Cliente, detectamos os atributos 'email' e 'senha', que apresentam a mesma função nas duas relações. Dessa forma, criamos uma relação denominada Usuário, com a função de armazenar o e-mail (pela regra de negócio, equivalente ao login) e a senha de todos os usuários que acessam o sistema, o que engloba



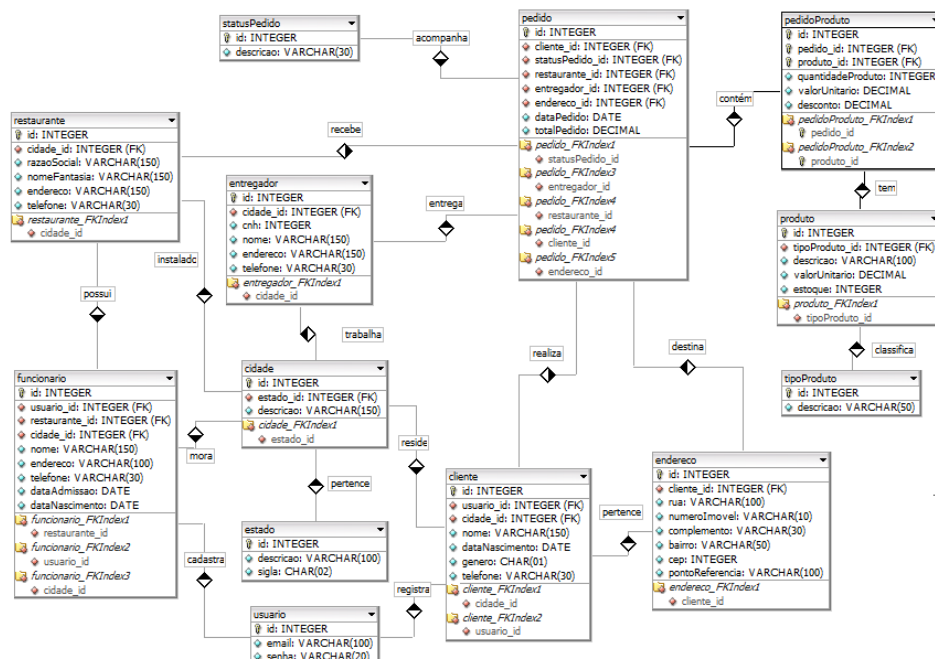
clientes e funcionários. Logo, a nova relação Usuário se relaciona com as duas relações citadas.

- Ainda na relação Cliente, encontramos um conjunto de atributos que compõem o endereço. Em busca de uma melhor organização dos dados, vamos gerar uma nova relação, chamada Endereço.
- Na relação Pedido, o atributo 'statusPedido' (em preparação, entregue, cancelado, entre outros) caracteriza-se como um atributo repetitivo, ou seja, um atributo que recebe dados iguais. Recorde que, pelas regras da normalização, todo atributo repetitivo gera uma nova relação – nesse caso, chamada de statusPedido.
- O mesmo ocorre na relação Produto. O atributo 'tipo' (massa, lanche, bebida, entre outros) gerará uma nova relação, chamada tipoProduto.

Vale salientar que em todas as relações presentes no modelo lógico a cardinalidade resultante foi 1:n ou n:1.

Após a aplicação das Formas Normais, apresentamos na figura o novo modelo relacional.

Figura 23 – Normalização do estudo de caso (ferramenta DBDesigner)



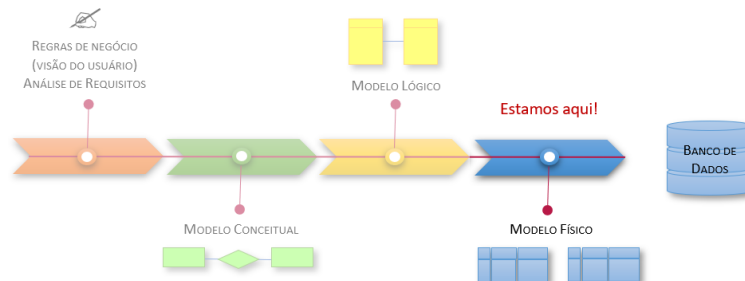
Fonte: Albano, 2022.

TEMA 3 – CONVERSÃO PARA O MODELO FÍSICO



Ingressamos na última fase do projeto de Bancos de Dados, a qual possui o objetivo de converter o modelo relacional para o modelo físico. Nesse modelo é onde ocorre a implementação propriamente dita.

Figura 24 – Localização no projeto de Bancos de Dados



Fonte: Albano, 2022.

Neste modelo, deve-se levar em consideração as limitações do Sistema Gerenciador de Bancos de Dados (SGBD) escolhido para o armazenamento dos dados. Exemplos de SGBD: MySQL, Oracle, SQL Server, DB2, Firebird e PostgreSQL.

No modelo físico, os objetos (relações) podem ser definidos e agrupados em esquemas. Tais esquemas são um agrupamento das relações, de acordo com finalidade, assunto, entre outras características.

As tabelas e as colunas são criadas de acordo com as definições do modelo lógico.

Nessa etapa, também são criadas restrições (chave primárias e estrangeiras), visualizações, índices, gatilhos e procedimentos.

Um modelo físico bem estruturado resulta em dados com melhor qualidade, além de atualizações e manutenções menos complexas.

3.1 Arquivo Físico

Como já mencionamos, as tabelas de um Bancos de Dados implicam um agrupamento lógico dos dados. Fisicamente, no disco, o Bancos de Dados é normalmente um arquivo único, onde estão armazenados todos os dados. No entanto, o arquivo físico contém mais do que apenas os dados propriamente ditos, pois armazena também o dicionário de dados, que é uma descrição da forma como os dados devem ser apresentados e manipulados. Além do arquivo com os dados, existem SGBDs que armazenam outros tipos de arquivos, como o arquivo de *log* das transações, que contém informações referentes às



operações realizadas no Bancos de Dados. Inclusive, um SGBD pode gerenciar vários bancos de dados em um mesmo servidor.

É importante ressaltar que o gerenciamento do Bancos de Dados fica a cargo do DBA, profissional responsável por gerenciar o espaço em disco, considerando índices, alocação dos arquivos no disco, entre outras atividades.

3.2 Terminologias e Equivalências

Você pode estar um pouco confuso com o uso de tantas terminologias diferentes para objetos similares. Para auxiliá-lo na compreensão e na aplicação da terminologia correta, apresentamos a seguir uma tabela com os termos e suas equivalências em um Bancos de Dados.

Figura 25 – Terminologias e equivalências

Modelo Conceitual	Modelo Relacional	Modelo Físico
Entidade	Relação	Tabela
Campo	Atributo	Coluna
Registro	Tupla	Linha

Fonte: Albano, 2022.

Figura 26 – Identificação da terminologia de Bancos de Dados

Entidade / Relação / Tabela			
CLIENTE			
codigo	nome	nascimento	endereco
20001001	João da Silva	01/01/1980	Sete de setembro, 1000
20002100	Maria de Souza	30/01/1985	XV novembro, 10
19992009	Paulo de Gil	01/01/1980	General Osório, 102
20039564	Maria de Souza	12/10/1990	Getúlio Vargas, 200
20093212	Ana de Jesus	26/09/1992	Sete de setembro, 1000

Fonte: Albano, 2022.

3.3 Estudo de caso

Chegamos na última fase do projeto de Bancos de Dados. Após a conversão do modelo relacional para o modelo físico, resta apenas executar o *script* e o nosso Bancos de Dados estará pronto.

A transformação do modelo relacional para o modelo físico pode ser facilitada pelo uso das ferramentas de modelagem relacional. Normalmente, tais



ferramentas possibilitam a geração automática do modelo físico, ou seja, o *script* em SQL para a criação do Bancos de Dados.

Para criar o Bancos de Dados no SGBD, será necessário executar o *script* gerado no servidor.

Modelo físico (gerado pela ferramenta DBDesigner): caso você tenha optado pela ferramenta DBDesigner, que é a utilizada pelos autores, ressaltamos que ela fornece o recurso de geração automática de *script* do modelo físico. Por esse motivo, incluímos um passo a passo no DBDesigner para gerar o *script*, sendo:

- Menu File;
- Submenu Export; e
- Opção SQL Create Script.

Vale salientar que o DBDesigner gera, de forma automática, o *script* no padrão MySQL, que será o SGBD adotado nesta nossa caminhada.

Com o *script* pronto, o próximo passo é executá-lo na interface do SGBD MySQL. Existem diversas interfaces gráficas que fazem a interação com o SGBD. Neste material, vamos usar a interface gráfica MySQL Workbench.

```
CREATE DATABASE delivery
USE delivery;
```

```
CREATE TABLE tipoProduto (
  id INTEGER NOT NULL AUTO_INCREMENT,
  descricao VARCHAR(50) NULL,
  PRIMARY KEY(id) );
```

```
CREATE TABLE estado (
  id INTEGER NOT NULL AUTO_INCREMENT,
  descricao VARCHAR(100) NULL,
  sigla CHAR(02) NULL,
  PRIMARY KEY(id) );
```

```
CREATE TABLE statusPedido (
  id INTEGER NOT NULL AUTO_INCREMENT,
  descricao VARCHAR(30) NULL,
  PRIMARY KEY(id) );
```

```
CREATE TABLE usuario (
  id INTEGER NOT NULL AUTO_INCREMENT,
  email VARCHAR(100) NULL,
```




```
        senha VARCHAR(20) NULL,
        PRIMARY KEY(id) );

CREATE TABLE produto (
    id INTEGER NOT NULL AUTO_INCREMENT,
    tipoProduto_id INTEGER NOT NULL,
    descricao VARCHAR(100) NULL,
    valorUnitario DECIMAL NULL,
    estoque INTEGER NULL,
    PRIMARY KEY(id),
    FOREIGN KEY(tipoProduto_id) REFERENCES tipoProduto(id)
        ON DELETE NO ACTION ON UPDATE NO ACTION );

CREATE TABLE cidade (
    id INTEGER NOT NULL AUTO_INCREMENT,
    estado_id INTEGER NOT NULL,
    descricao VARCHAR(150) NULL,
    PRIMARY KEY(id),
    FOREIGN KEY(estado_id) REFERENCES estado(id)
        ON DELETE NO ACTION ON UPDATE NO ACTION );

CREATE TABLE entregador (
    id INTEGER NOT NULL AUTO_INCREMENT,
    cidade_id INTEGER NOT NULL,
    cnh INTEGER NULL,
    nome VARCHAR(150) NULL,
    endereco VARCHAR(150) NULL,
    telefone VARCHAR(30) NULL,
    PRIMARY KEY(id),
    FOREIGN KEY(cidade_id) REFERENCES cidade(id)
        ON DELETE NO ACTION ON UPDATE NO ACTION );

CREATE TABLE cliente (
    id INTEGER NOT NULL AUTO_INCREMENT,
    usuario_id INTEGER NOT NULL,
    cidade_id INTEGER NOT NULL,
    nome VARCHAR(150) NULL,
    dataNascimento DATE NULL,
    genero CHAR(01) NULL,
    telefone VARCHAR(30) NULL,
    PRIMARY KEY(id),
    FOREIGN KEY(cidade_id) REFERENCES cidade(id)
        ON DELETE NO ACTION ON UPDATE NO ACTION,
    FOREIGN KEY(usuario_id) REFERENCES usuario(id)
        ON DELETE NO ACTION ON UPDATE NO ACTION );

CREATE TABLE restaurante (
    id INTEGER NOT NULL AUTO_INCREMENT,
```



```
cidade_id INTEGER NOT NULL,
razaoSocial VARCHAR(150) NULL,
nomeFantasia VARCHAR(150) NULL,
endereco VARCHAR(150) NULL,
telefone VARCHAR(30) NULL,
PRIMARY KEY(id),
FOREIGN KEY(cidade_id) REFERENCES cidade(id)
    ON DELETE NO ACTION ON UPDATE NO ACTION );

CREATE TABLE endereco (
    id INTEGER NOT NULL AUTO_INCREMENT,
    cliente_id INTEGER NOT NULL,
    rua VARCHAR(100) NULL,
    numeroImovel VARCHAR(10) NULL,
    complemento VARCHAR(30) NULL,
    bairro VARCHAR(50) NULL,
    cep INTEGER NULL,
    pontoReferencia VARCHAR(100) NULL,
    PRIMARY KEY(id),
    FOREIGN KEY(cliente_id) REFERENCES cliente(id)
        ON DELETE NO ACTION ON UPDATE NO ACTION );

CREATE TABLE funcionario (
    id INTEGER NOT NULL AUTO_INCREMENT,
    usuario_id INTEGER NOT NULL,
    restaurante_id INTEGER NOT NULL,
    cidade_id INTEGER NOT NULL,
    nome VARCHAR(150) NULL,
    endereco VARCHAR(100) NULL,
    telefone VARCHAR(30) NULL,
    dataAdmissao DATE NULL,
    dataNascimento DATE NULL,
    PRIMARY KEY(id),
    FOREIGN KEY(cidade_id) REFERENCES cidade(id)
        ON DELETE NO ACTION ON UPDATE NO ACTION,
    FOREIGN KEY(restaurante_id) REFERENCES restaurante(id)
        ON DELETE NO ACTION ON UPDATE NO ACTION,
    FOREIGN KEY(usuario_id) REFERENCES usuario(id)
        ON DELETE NO ACTION ON UPDATE NO ACTION );

CREATE TABLE pedido (
    id INTEGER NOT NULL AUTO_INCREMENT,
    statusPedido_id INTEGER NOT NULL,
    restaurante_id INTEGER NOT NULL,
    endereco_id INTEGER NOT NULL,
    entregador_id INTEGER NOT NULL,
    dataPedido DATE NULL,
    totalPedido DECIMAL NULL,
```



```
PRIMARY KEY(id),
FOREIGN KEY(entregador_id) REFERENCES entregador(id)
    ON DELETE NO ACTION ON UPDATE NO ACTION,
FOREIGN KEY(endereco_id) REFERENCES cliente(id)
    ON DELETE NO ACTION ON UPDATE NO ACTION,
FOREIGN KEY(restaurante_id) REFERENCES restaurante(id)
    ON DELETE NO ACTION ON UPDATE NO ACTION,
FOREIGN KEY(statusPedido_id) REFERENCES statusPedido(id)
    ON DELETE NO ACTION ON UPDATE NO ACTION );

CREATE TABLE pedidoProduto (
    id INTEGER ZEROFILL NOT NULL,
    pedido_id INTEGER NOT NULL,
    produto_id INTEGER NOT NULL,
    quantidadeProduto INTEGER NULL,
    valorUnitario DECIMAL NULL,
    desconto DECIMAL NULL,
    PRIMARY KEY(id),
    FOREIGN KEY(pedido_id) REFERENCES pedido(id)
        ON DELETE NO ACTION ON UPDATE NO ACTION,
    FOREIGN KEY(produto_id) REFERENCES produto(id)
        ON DELETE NO ACTION ON UPDATE NO ACTION );
```

No *script* descrito, existem vários detalhes que com certeza podem motivar dúvidas para você que, pela primeira vez, está tendo contato com o SQL. Não se preocupe, vamos começar a estudar esse assunto de forma mais profunda em outra ocasião.

TEMA 4 – SCHEMAS

Além do modelo relacional, que estudamos nessa etapa, também existe outro tipo de modelagem que pode ser utilizada nos SGBDs relacionais, conhecida como modelagem dimensional, usada na criação de Data Warehouse e Data Mart, que são a base para o Business Intelligence (BI).

Um Data Warehouse (armazém de dados) consiste em um Bancos de Dados que reúne informações de várias fontes, como comentários de redes sociais, planilhas, documentos, outros Bancos de Dados, entre outros, as quais são reunidas e padronizadas em um único Bancos de Dados.

Já um Data Mart é uma divisão lógica de um Data Warehouse, que atende a uma área específica de negócio (marketing, vendas, entre outras). Uma



mesma empresa pode construir diversos Data Marts ao longo do tempo, vinculando-os a um Data Warehouse da empresa.

4.1 Modelagem Dimensional

Enquanto os modelos relacionais são projetados para eliminar a redundância de dados e executar rapidamente as operações de inserção, atualização, exclusão e consulta de dados, os modelos dimensionais abarcam um conjunto de dados desnormalizados, projetados para a recuperação de dados de um *Data Warehouse*. Essa técnica de modelagem foi proposta por Ralph Kimball.

O modelo dimensional permite a visualização das mesmas informações a partir de ângulos distintos, como um cubo, em que cada face representa uma análise da informação.

A modelagem multidimensional é definida por dois pilares: dimensões e fatos.

4.1.1 Tabela Dimensão

Conjunto de objetos que descrevem e classificam os fatos por meio de seus atributos. Por exemplo, os atributos de uma dimensão restaurante podem incluir uma hierarquia de bairro, cidade e estado, além de atributos descritivos, como o nome do restaurante.

Normalmente, as dimensões apresentam atributos com dados textuais. Essas tabelas contêm a descrição dos fatos que serão analisados. Por exemplo, se analisamos as receitas (vendas) de uma rede de restaurante, as dimensões para a avaliação podem ser produtos vendidos, os dias com maiores vendas, entre outros aspectos.

4.1.2 Tabela Fato

Os fatos são as medidas do negócio. Geralmente englobam dados numéricos e aditivos, ou seja, dados que podem ser reunidos por soma, média ou pela aplicação de outras funções. Por exemplo, o valor total das vendas do restaurante e a quantidade de produtos vendidos.

As tabelas fatos são as principais tabelas do modelo dimensional, onde as métricas (indicativos) que interessam à empresa serão armazenadas.



4.3 Modelos

A modelagem dimensional tem três tipos de modelos:

- Estrela (Star Schema);
- Floco de neve (Snow Flake); e
- Constelação (Fact Constellation).

4.3.1 Modelo estrela – Star Schema

Ocorre quando todas as dimensões se relacionam diretamente com a tabela fato. Assim, as dimensões devem conter todas as descrições necessárias para definir os fatos que vão ser analisados. Por exemplo, a descrição do produto, o nome das cidades e o nome das categorias dos produtos.

Uma característica desse modelo é que as dimensões não são normalizadas, ou seja, podem conter dados redundantes, nulos, entre outros.

Figura 27 – Exemplo do modelo *Star Schema*



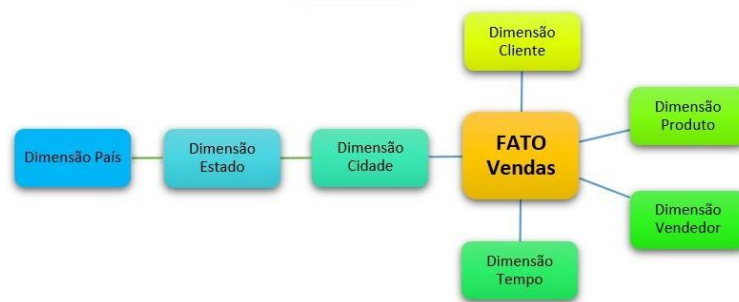
Fonte: Albano, 2022.

4.3.2 Modelo Floco de Neve – Snow Flake

Neste modelo, as dimensões não se relacionam necessariamente com a tabela fato, pois podem se relacionar com outras dimensões. É o modelo que mais se assemelha ao modelo relacional, em que os dados se encontram mais normalizados. Consequentemente, ocupam menos espaço.



Figura 28 – Exemplo do modelo *Snow Flake*



Fonte: Albano, 2022.

4.3.3 Modelo constelação – Fact Constellation

O modelo constelação é uma variação dos outros dois modelos, mas apresenta mais de uma tabela fato. Por conta disso, recebe o nome de constelação. É um modelo que contém várias “estrelas”, formando uma constelação. Geralmente, esse tipo de modelo é utilizado para representar os fatos com um nível de detalhamento (granularidade) diferente. Por exemplo, temos duas consultas: uma apresenta o total diário vendido no mês e outra o total mensal. Nessa situação, seriam implementadas duas tabelas fato, uma com os totais diários e outra com os totais mensais, o que reduz o tempo de resposta da consulta.

Figura 29 – Exemplo do modelo constelação



Fonte: Albano, 2022.

TEMA 5 – *STRUCTURED QUERY LANGUAGE (SQL)*

Sempre que desejamos acessar uma informação que está armazenada em um Bancos de Dados, o acesso será realizado através da linguagem SQL. Ou seja, independentemente da linguagem de programação (Python, Java, PHP, entre outras), a comunicação com o Bancos de Dados será realizada com o SQL.



A IBM foi a primeira empresa a desenvolver um modelo de Bancos de Dados relacional, consequentemente, o SQL, na década de 1970. Vale salientar que Edgar Codd, que propôs o modelo, era funcionário da empresa.

Após a IBM, outras empresas, como Oracle e Microsoft, criaram Bancos de Dados relacionais – consequentemente, os seus SQLs. Apesar do conceito e da estrutura serem idênticos, cada empresa criou a sua própria distribuição, o que acabou gerando falta de padronização, o que por sua vez dificulta a migração de um Bancos de Dados de uma distribuição para outra.

Assim, o *American National Standard Institute* (ANSI) reuniu as empresas de tecnologia que desenvolviam Bancos de Dados para criar um padrão que deveria ser adotado por todas. Como consequência, em 1986 a ANSI lançou a primeira versão do SQL ANSI. Tal padrão era uma recomendação, ou seja, as empresas não eram obrigadas a adotá-lo, mas com o passar do tempo o SQL ANSI virou de fato um padrão reconhecido.

É importante ressaltar que, apesar da existência do padrão SQL ANSI, as empresas continuam tendo liberdade de adicionar, em seus SQLs, novos recursos, comandos, tipos de dados, entre outros fatores. Porém, elas devem manter a base (*core*) do SQL ANSI.

Uma curiosidade interessante é que a Organização Internacional de Normalização (International Organization for Standardization – ISO) juntou-se à ANSI na padronização do SQL, dando mais credibilidade a linguagem.

Vale salientar ainda que a linguagem SQL está em constante evolução. De tempos em tempos, é lançada uma nova atualização. Por exemplo, em 1999 surgiu o tipo de dado *boolean* (verdadeiro/falso), bem como o comando *savepoint*. Em 2003, foram lançados o comando *merge*, o tipo de dado *bigint*, o *sequence* e o *identity* (os dois últimos são formas de autoincremento). De 2003 a 2006, foram implementados vários recursos para suporte ao XML. Em 2016, foi introduzida a extensão JSON.

Outro aspecto importante é a interface de comunicação entre o SGBD e o Bancos de Dados propriamente dito. Essa interface normalmente é feita por meio de *prompt* de comando, ou seja, o usuário digita o comando SQL em uma linha de comando. Contudo, por comodidade, todos os SGBDs fornecem uma interface gráfica, com visual e recursos que facilitam as atividades do usuário no dia a dia. Além das ferramentas dos próprios fabricantes, encontramos uma variedade de outras interfaces que podem ser conectadas ao SGBD, para o



acesso ao Bancos de Dados. Com o auxílio dessas ferramentas, a realização de tarefas em um Bancos de Dados se tornou mais prática.

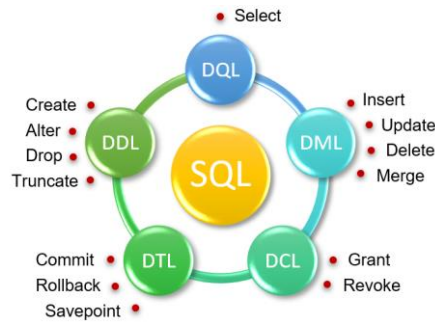
5.1 Divisão do SQL

O SQL ANSI, ou simplesmente SQL, apresenta uma série de comandos, classificados nas seguintes categorias.

- Data Definition Language (DDL ou Linguagem de Definição de Dados): definição dos dados, com comandos utilizados nas definições dos objetos do Bancos de Dados. Ou seja, comandos para criação, alteração ou exclusão dos objetos. Tais comandos são: *create*, *drop*, *alter* e *truncate*.
- Data Manipulation Language (DML ou Linguagem de Manipulação de Dados): atua na manipulação de tabelas, através de inserções, exclusões e alterações em linhas (registros) armazenadas em tabelas. Os comandos pertencentes a essa subdivisão são: *insert*, *update*, *delete* e *merge*.
- Data Query Language (DQL ou Linguagem de Consulta de Dados): comando que possibilita a consulta dos dados. O comando é o *select*. Provavelmente, esse é o comando mais utilizado, pois possibilita a realização de todas as consultas no Bancos de Dados.
- Data Control Language (DCL ou Linguagem de Controle dos Dados): age no controle de acesso dos usuários aos objetos e dados, através dos comandos *grant* e *revoke*.
- Data Transaction Language (DTL ou Linguagem de Transação de Dados): os comandos *commit*, *rollback* e *savepoint* permitem, respectivamente, a confirmação ou o cancelamento de uma transação, podendo conter um ou mais comandos SQL. Dessa forma, garantimos que toda a transação realizada em um Bancos de Dados seja finalizada.



Figura 30 – Subdivisões da linguagem SQL



Fonte: Albano, 2022.

5.2 Query (Script)

Você irá ouvir muito sobre *query* e *script* na aplicação da linguagem SQL, em expressões como “fiz uma *query* que...”, “executei a *query* para...”. Porém, o que seria uma *query* (*script*)? Basicamente, uma *query* é uma instrução SQL que pode ser composta por apenas uma linha de comando ou um conjunto de comandos sequenciais que serão executados por um SGBD.

Cada instrução SQL deverá ser finalizada com ponto e vírgula (;). É através desse caractere que o SGBD identifica o final da instrução. Aqui, vale uma observação: nas interfaces gráficas, o ponto e vírgula pode ser opcional; porém, na interface de linha, o seu uso é obrigatório.

Figura 31 – Exemplo de *query* (ferramenta MySQL Workbench)

```
1 • use exemplo;
2
3 • insert into estado (estID, estNome, EstSigla, EstRegiao) values (1, 'Paraná', 'PR', 'Sul');
4 • insert into estado (estID, estNome, EstSigla, EstRegiao) values (2, 'Rio Grande do Sul', 'RS', 'Sul');
5 • insert into estado (estID, estNome, EstSigla, EstRegiao) values (3, 'Santa Catarina', 'SC', 'Sul');
6
```

Fonte: Albano, 2022.

Observe a *query* anterior, em que executamos quatro comandos: o *use* e três comandos *insert*. Veja que todos os casos foram finalizados com ponto e vírgula.

Outro detalhe é que você pode inserir comentários em uma *query*, englobando apenas uma linha (--) ou várias linhas (/* ... */).

Exemplo:

```
/* Barra asterisco inicia o comentário para várias linhas
e asterisco barra finaliza o bloco de comentário */
-- Dois hifens, comentário de linha única.
```



5.3 Dialetos

Como vimos, cada empresa que desenvolve SGBDs (Oracle, Microsoft, IBM, entre outras), além de usar o padrão do SQL ANSI, também utiliza uma linguagem adicional para o desenvolvimento de códigos mais avançados no Bancos de Dados, o que chamamos de dialeto. Normalmente, os dialetos são utilizados no desenvolvimento de códigos de criação de procedimento (*procedure*), gatilhos (*trigger*) e cursores, os quais exigem a aplicação de uma lógica de programação mais elaborada.

Como os dialetos apresentam variações de um SGBD para outro, é necessário estudar o dialeto referente ao SGBD escolhido. Os dialetos existentes são:

- PL/SQL é a linguagem de programação utilizada pelo Oracle e MySQL;
- Transact-SQL (T-SQL) é propriedade da Microsoft e Sybase;
- SQL/PL, DB2 da IBM;
- PL/pgSQL, PostgreSQL (variação do PL/SQL); e
- PSQL, Firebird.

FINALIZANDO

Nesta etapa, estudamos o modelo relacional, o processo de normalização e a sua importância para a criação de um modelo eficiente, enxuto e sem a presença de dados redundantes. Realizamos a transformação do nosso Modelo Entidade-Relacionamento (MER) para o modelo lógico. Concluímos assim o nosso projeto de Bancos de Dados, convertendo o modelo lógico para o modelo físico. Também iniciamos o estudo da linguagem SQL, que nos permite criar e manipular os objetos e os dados presentes em um Bancos de Dados.

Mais uma vez, propomos que você revise os assuntos tratados nesta etapa, pratique e crie os seus próprios exemplos de Bancos de Dados, refazendo todas as etapas do projeto (MER com cardinalidade, modelo relacional, sua normalização e, por fim, modelo físico). Assim, você estará preparado para outras discussões, durante as quais vamos trabalhar com SQL, compreendendo, na prática, os conceitos estudados até o momento.



REFERÊNCIAS

DATE, C. J. **Introdução à Sistemas de Bancos de Dados**. Rio de Janeiro: Campus, 2000.