EAD

EDUCAÇÃO A DISTÂNCIA

UNIDADE I

Linguagem de Programação de Banco de Dados

Profa. Dra. Vanessa Lessa

Arquiteturas de sistema de banco de dados

- A arquitetura de um sistema de banco de dados é dependente totalmente das definições de ***hardware*** e **sistema computacional**.

Considerando a história da evolução dos bancos de dados, vamos focar em quatro tipos de arquitetura:

- Centralizada;
- Cliente-servidor;
- Paralela;
- Distribuída.

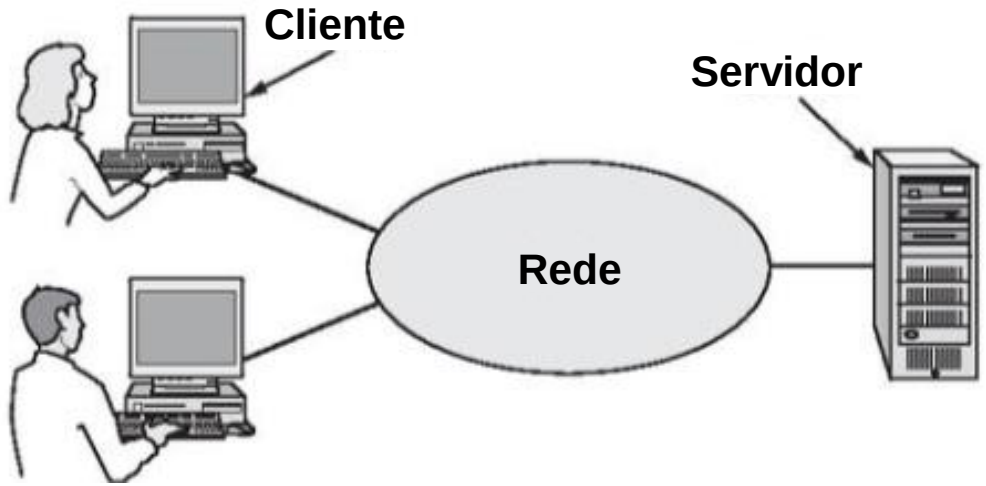
Arquiteturas centralizadas e cliente-servidor

Arquitetura Centralizada (1970)

- *Mainframes* (computador – grande porte para processamento de enormes volumes de dados).
- Usuários acessavam os dados usando terminais sem capacidade de processamento.
- Sistemas: multiusuário e monousuário.

Arquitetura Cliente-Servidor

- Evolução dos computadores pessoais;
- Máquinas mais rápidas e com um custo menor.



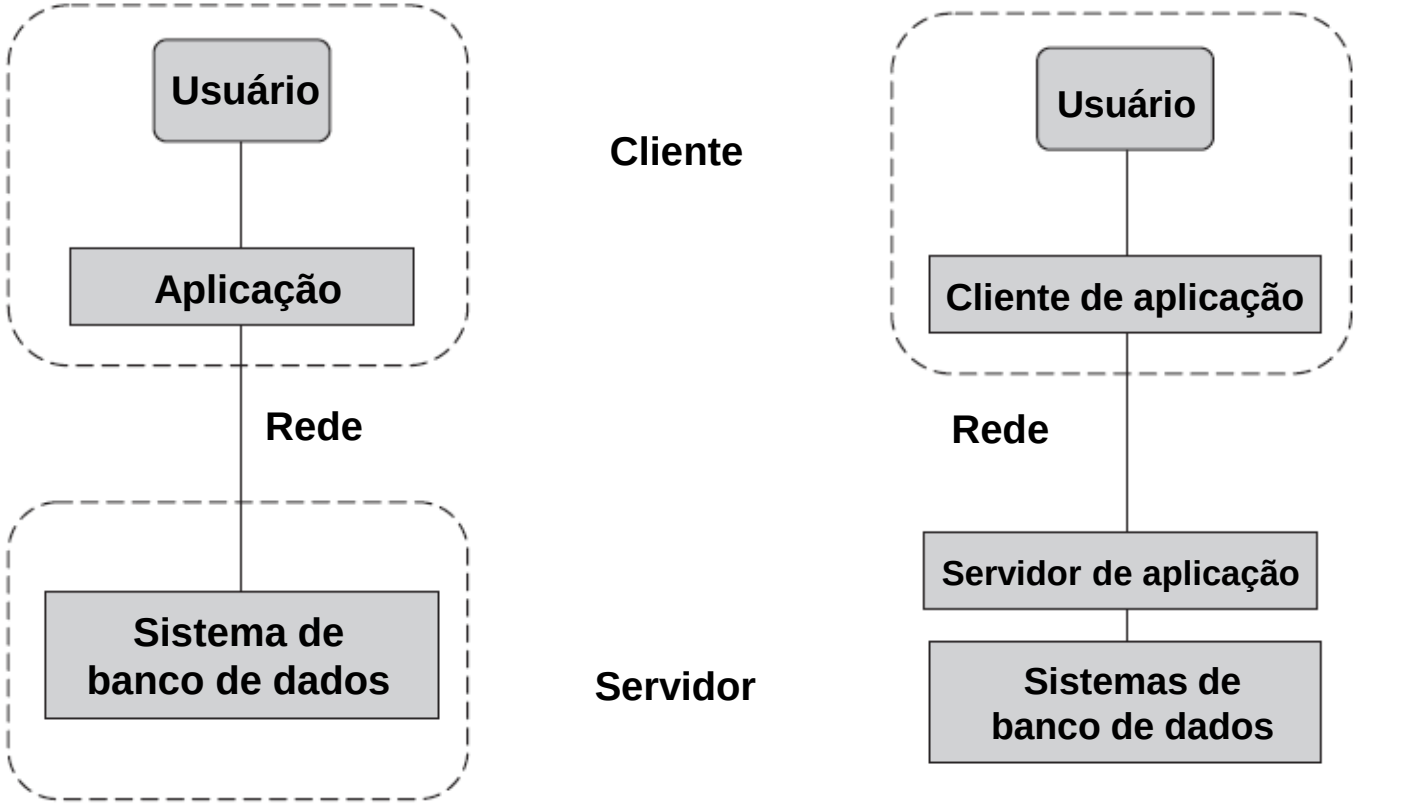
Fonte: adaptado de: Tanenbaum (2011, p. 2).

Arquitetura cliente-servidor

A funcionalidade apresentada pelos servidores de bancos de dados cliente-servidor pode ser dividida em duas partes:

- *front-end* – faz a interface com o usuário final;
- *back-end* – controla as estruturas de acesso ao dado.

Fonte: adaptado de: Silberschatz (2020, p. 13).



(a) arquitetura de duas camadas **(b) arquitetura de três camadas**

Arquiteturas de sistema servidor

Podemos classificar os sistemas servidores em dois tipos:

- **Sistemas servidores de transação ou servidores de consulta**: disponibilizam uma interface para o cliente que pode enviar solicitações de ação. O servidor por sua vez executa a ação solicitada pelo cliente e devolve o resultado.

- **Sistemas servidores de dados**: possibilitam a interação do cliente com o servidor executando ações como leitura ou atualização de dados, em unidades como arquivos ou páginas.

Sistemas paralelos

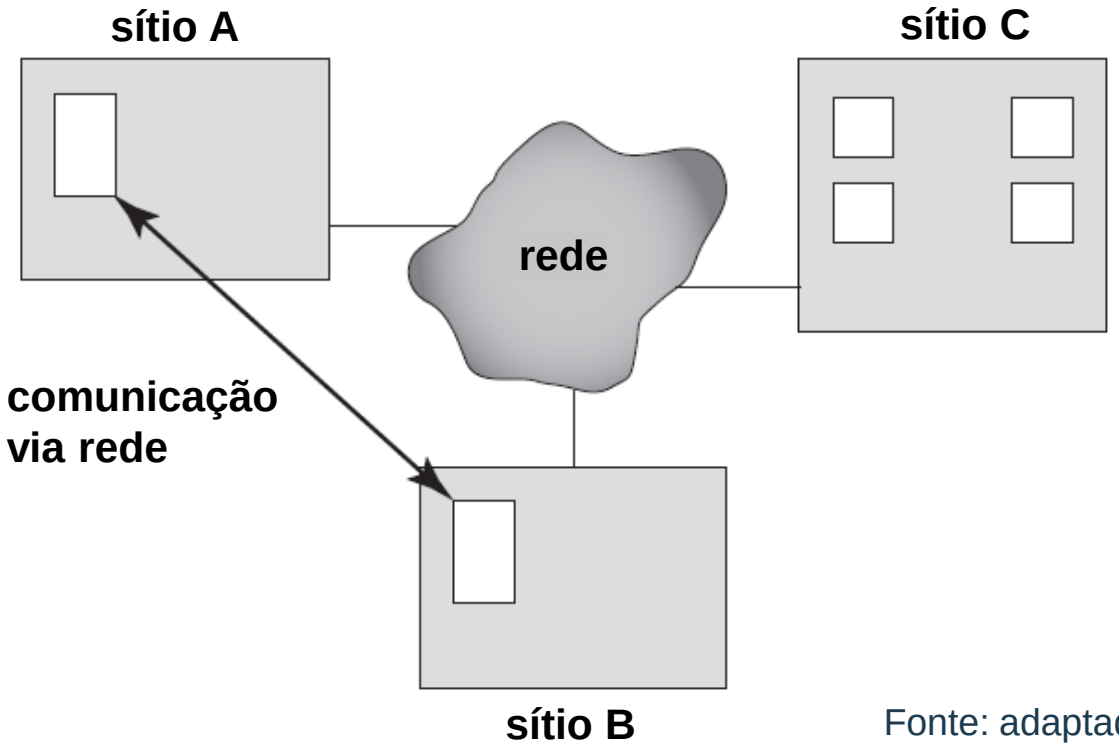
- Compartilhamento de recursos para melhorar o processamento de dados.
- No processamento serial, as operações são realizadas de forma sequenciada, de maneira oposta, o processamento paralelo realiza diversas operações ao mesmo tempo.

Podemos analisar o desempenho de um sistema de banco de dados utilizando basicamente duas medidas:

1. *throughput* – a quantidade de tarefas que o sistema de banco de dados pode executar completamente em um determinado intervalo de tempo;
2. tempo de resposta – quanto tempo é necessário para completar uma determinada tarefa a partir do instante em que ela foi submetida.

Sistemas distribuídos

- Os sistemas distribuídos caracterizam-se pela utilização de diferentes máquinas que compartilham recursos conectadas entre si por meio de uma rede de comunicação para atingir um objetivo comum ou compartilhado. A diferença entre os sistemas distribuídos e os sistemas paralelos está na localização das máquinas. Nos sistemas paralelos, as máquinas devem estar próximas. Para os sistemas distribuídos, a localização das máquinas é indiferente, podendo estar geograficamente distantes.



Fonte: adaptado de: Silberschatz (2020, p. 557).

Bancos de dados paralelos e dados distribuídos

- Na década de 1980 surgiu o **Sistema de Gerência de Banco de Dados Paralelos (SGBDP)** com o objetivo de resolver o problema dos bancos de dados convencionais de “gargalo de E/S”, devido ao alto custo de acesso aos mecanismos de armazenamento secundário (discos) se comparado com memória principal. Uma solução foi particionar os dados utilizando vários discos, podendo acessá-los paralelamente durante uma consulta. Dessa forma teríamos uma melhora do *throughput* para cada disco utilizado na estrutura.

- Um **Sistema de Gerência de Banco de Dados Distribuído (SGBDD)** constitui-se de uma relação de nós (pontos de conexão, como por exemplo os computadores), em que cada um pode participar na execução de transações que acessam dados em um ou mais nós. Em um SGBDD podemos armazenar os dados em vários computadores que se comunicam entre si, por meios de comunicação como redes sem fio, redes de alta velocidade e memória principal.

Paralelismo de E/S

O objetivo de diminuir o tempo de processamento de um programa, particionando as relações sobre múltiplos discos. Podemos listar três tipos de particionamento de dados:

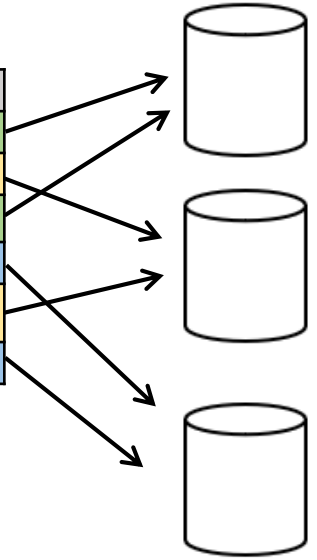
- Horizontal.
- Vertical.
- Misto.

Particionamento horizontal

- O particionamento horizontal é o tipo mais utilizado, em que cada tupla (uma linha) de uma relação é distribuída entre os discos. Dessa forma, cada tupla estará em um disco diferente.

codigo_cliente	nome_cliente	endereco	dt_nasc	sexo	credito
C-001	Monteiro Lobato	Praça Picapau Amarelo	18/04/1882	M	5.000,00
C-002	José de Alencar	Rua Iracema	01/05/1829	M	4.000,00
C-101	Cecília Meireles	Av Espectros	07/11/1901	F	9.000,00
C-102	Carlos Drummond	Av Poemas	31/10/1902	M	7.000,00
C-103	Machado de Assis	Rua Dom Casmurro	21/06/1839	M	4.000,00
C-202	Clarice Lispector	Rua Todos os Contos	10/12/1920	F	8.000,00

Fonte: autoria própria.



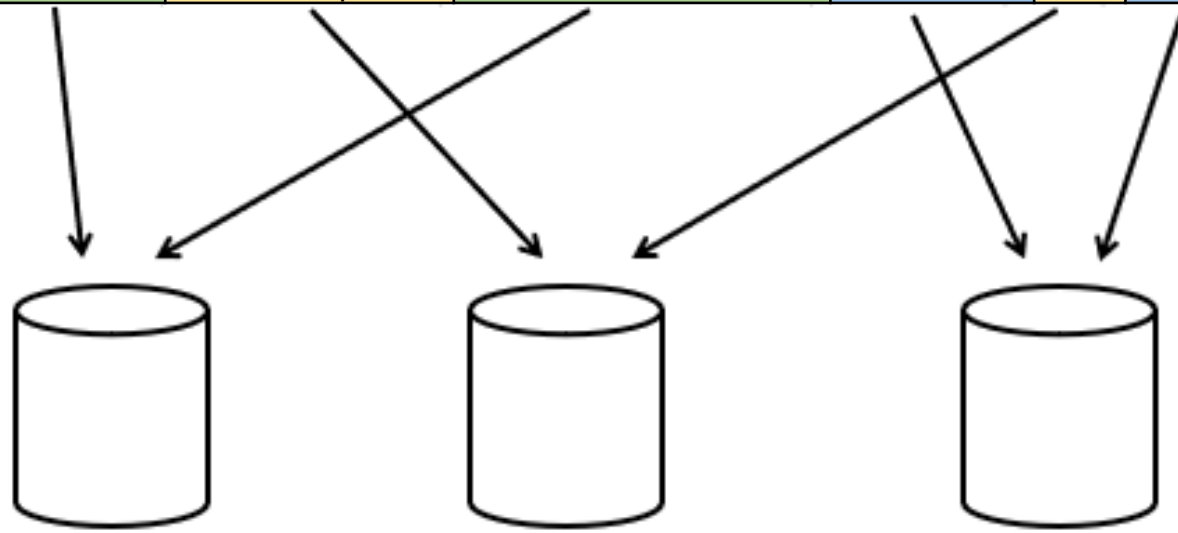
Paralelismo de E/S

Particionamento vertical

- Os campos de uma relação são separados entre os discos, cada campo reside em um ou mais discos. Esse tipo de particionamento divide a relação original em um conjunto de relações menores, com o objetivo de diminuir o tempo de execução da aplicação que utiliza esses fragmentos.

codigo_cliente	nome_cliente	endereco	dt_nasc	sexo	credito
C-001	Monteiro Lobato	Praça Picapau Amarelo	18/04/1882	M	5.000,00
C-002	José de Alencar	Rua Iracema	01/05/1829	M	4.000,00
C-101	Cecília Meireles	Av Espectros	07/11/1901	F	9.000,00
C-102	Carlos Drummond	Av Poemas	31/10/1902	M	7.000,00
C-103	Machado de Assis	Rua Dom Casmurro	21/06/1839	M	4.000,00
C-202	Clarice Lispector	Rua Todos os Contos	10/12/1920	F	8.000,00

Fonte: autoria própria.



Paralelismo de E/S

Particionamento misto

Em algumas situações, a utilização de apenas um tipo de particionamento horizontal ou vertical não satisfaz os acessos da aplicação. Para essas situações existem dois tipos de particionamento misto:

- HV – primeiro executamos uma partição horizontal na relação. Em seguida, nos dados fragmentos executamos uma partição vertical.

- VH – primeiro executamos uma partição vertical na relação. Em seguida, nos dados fragmentos executamos uma partição horizontal.

Interatividade

Analise as afirmações:

I. A arquitetura de um banco de dados é dependente totalmente das definições de *hardware*.

II. Na arquitetura centralizada, o *mainframe* era responsável por todo o processamento necessário.

III. Na arquitetura cliente-servidor, os computadores pessoais utilizados pelos usuários assumiram o gerenciamento da interface.

IV. Os sistemas distribuídos caracterizam-se pela utilização de diferentes máquinas conectadas entre si por meio de uma rede de comunicação, mas que não compartilham recursos.

V. No particionamento horizontal, os campos de uma relação são separados entre os discos.

Estão corretas:

a) I, II e III.
b) III e IV.
c) I, II e V.
d) III, IV e V.
e) Todas as afirmações.

Resposta

Analise as afirmações:

I. A arquitetura de um banco de dados é dependente totalmente das definições de *hardware*.

II. Na arquitetura centralizada, o *mainframe* era responsável por todo o processamento necessário.

III. Na arquitetura cliente-servidor, os computadores pessoais utilizados pelos usuários assumiram o gerenciamento da interface.

IV. Os sistemas distribuídos caracterizam-se pela utilização de diferentes máquinas conectadas entre si por meio de uma rede de comunicação, mas que não compartilham recursos.

V. No particionamento horizontal, os campos de uma relação são separados entre os discos.

Estão corretas:

a) I, II e III.
b) III e IV.
c) I, II e V.
d) III, IV e V.
e) Todas as afirmações.

Bancos de dados relacionais

- Os bancos relacionais são os bancos mais utilizados no mundo. Em 1970, Edgard Frank Codd definiu o modelo relacional, que continua dominante no mercado. Codd propôs uma nova forma de pensamento que desconectou a estrutura lógica do banco de dados do método de armazenamento físico. Dessa maneira, a organização dos dados poderia ser tratada considerando um conjunto de relações.

- O modelo relacional criado por Codd (1970) é baseado no modelo de dados para aplicações de processamento de dados. Utilizamos um conjunto de tabelas para caracterizar os dados e as relações entre eles, em que cada tabela pode conter diversas colunas e cada coluna receber um nome único.

    - O modelo relacional é baseado na estrutura de registros de formato fixo de vários tipos. As tabelas possuem registros (tipos próprios) e cada registro possui um número fixo de atributos (campos). Os atributos correspondem às colunas das tabelas.

Estrutura dos bancos de dados relacionais

A tabela clientes possui três cabeçalhos de coluna: codigo_cliente, nome_cliente e credito. Considerando a terminologia do modelo relacional, denominamos esses cabeçalhos de atributos, em que cada atributo possui um conjunto de valores possíveis, chamado de domínio do atributo. Por exemplo, para o atributo crédito, o domínio é o conjunto dos números positivos. Dessa forma, temos:

- D1 – o conjunto de todos os códigos de clientes.
- D2 – o conjunto de todos os nomes de clientes.
- D3 – o conjunto de todos os créditos.

codigo_cliente	nome_cliente	credito
C-001	Monteiro Lobato	5.000,00
C-002	José de Alencar	4.000,00
C-101	Cecília Meireles	9.000,00
C-102	Carlos Drummond	7.000,00
C-103	Machado de Assis	4.500,00
C-202	Clarice Lispector	8.000,00

Fonte: autoria própria.

Operações fundamentais da álgebra relacional

Operação de seleção

- Utilizamos a operação de seleção quando precisamos apresentar tuplas que satisfaçam um determinado predicado. A letra grega sigma (σ) é usada para denotar seleção.

codigo_cliente	nome_cliente	credito
C-001	Monteiro Lobato	5.000,00
C-002	José de Alencar	4.000,00
C-101	Cecília Meireles	9.000,00
C-102	Carlos Drummond	7.000,00
C-103	Machado de Assis	4.500,00
C-202	Clarice Lispector	8.000,00

Fonte: autoria própria.

$$\sigma_{\text{credito} > 7500}(\text{cliente})$$

codigo_cliente	nome_cliente	credito
C-101	Cecília Meireles	9.000,00
C-202	Clarice Lispector	8.000,00

- Para comparações, podemos usar =, ≠, <, ≤, >, ≥ no predicado da seleção. Podemos conjugar diversos predicados em um predicado maior usando os conectivos e (∧), ou (∨) e não ().

Operações fundamentais da álgebra relacional

Operação de projeção

- Utilizamos a operação de projeção quando precisamos selecionar colunas específicas de uma relação. A letra grega pi (π) é utilizada para projeção.

codigo_cliente	nome_cliente	credito
C-001	Monteiro Lobato	5.000,00
C-002	José de Alencar	4.000,00
C-101	Cecília Meireles	9.000,00
C-102	Carlos Drummond	7.000,00
C-103	Machado de Assis	4.500,00
C-202	Clarice Lispector	8.000,00

Fonte: autoria própria.

π codigo_cliente, credito (cliente)

codigo_cliente	credito
C-001	5.000,00
C-002	4.000,00
C-101	9.000,00
C-102	7.000,00
C-103	4.500,00
C-202	8.000,00

Operações fundamentais da álgebra relacional

Operação de união

Relação de Pedido

codigo_cliente	nome_cliente	valor_pedido
C-001	Monteiro Lobato	1.000,00
C-101	Cecília Meireles	800,00
C-202	Clarice Lispector	2.000,00

Fonte: autoria própria.

Relação de Cotação

codigo_cliente	nome_cliente	valor_cotacao
C-001	Monteiro Lobato	600,00
C-002	José de Alencar	1.500,00
C-101	Cecília Meireles	1.000,00
C-102	Carlos Drummond	700,00

Fonte: autoria própria.

π nome_cliente (pedido) u
π nome_cliente (cotação)

nome_cliente
Monteiro Lobato
José de Alencar
Cecília Meireles
Carlos Drummond
Clarice Lispector

Fonte: autoria própria.

Operações fundamentais da álgebra relacional

Operação de diferença de conjunto

Relação de Cotação

codigo_cliente	nome_cliente	valor_cotacao
C-001	Monteiro Lobato	600,00
C-002	José de Alencar	1.500,00
C-101	Cecília Meireles	1.000,00
C-102	Carlos Drummond	700,00

Fonte: autoria própria.

Relação de Pedido

codigo_cliente	nome_cliente	valor_pedido
C-001	Monteiro Lobato	1.000,00
C-101	Cecília Meireles	800,00
C-202	Clarice Lispector	2.000,00

Fonte: autoria própria.

π nome_cliente (cotação) -
π nome_cliente (pedido)

nome_cliente
José de Alencar
Carlos Drummond

Fonte: autoria própria.

Operações fundamentais da álgebra relacional

Operação de produto cartesiano

- Para combinar duas relações, utilizamos a operação produto cartesiano, representada pelo sinal de vezes (x), temos r1 x r2.

cliente x pedido

cliente.codigo_cliente	cliente.nome_cliente	credito	pedido.codigo_cliente	pedido.nome_cliente	valor_pedido
C-001	Monteiro Lobato	5.000,00	C-001	Monteiro Lobato	1.000,00
C-001	Monteiro Lobato	5.000,00	C-101	Cecília Meireles	800,00
C-001	Monteiro Lobato	5.000,00	C-202	Clarice Lispector	2.000,00
C-002	José de Alencar	4.000,00	C-001	Monteiro Lobato	1.000,00
C-002	José de Alencar	4.000,00	C-101	Cecília Meireles	800,00
C-002	José de Alencar	4.000,00	C-202	Clarice Lispector	2.000,00
C-101	Cecília Meireles	9.000,00	C-001	Monteiro Lobato	1.000,00
C-101	Cecília Meireles	9.000,00	C-101	Cecília Meireles	800,00
C-101	Cecília Meireles	9.000,00	C-202	Clarice Lispector	2.000,00
C-102	Carlos Drummond	7.000,00	C-001	Monteiro Lobato	1.000,00
C-102	Carlos Drummond	7.000,00	C-101	Cecília Meireles	800,00
C-102	Carlos Drummond	7.000,00	C-202	Clarice Lispector	2.000,00
C-103	Machado de Assis	4.500,00	C-001	Monteiro Lobato	1.000,00
C-103	Machado de Assis	4.500,00	C-101	Cecília Meireles	800,00
C-103	Machado de Assis	4.500,00	C-202	Clarice Lispector	2.000,00
C-202	Clarice Lispector	8.000,00	C-001	Monteiro Lobato	1.000,00
C-202	Clarice Lispector	8.000,00	C-101	Cecília Meireles	800,00
C-202	Clarice Lispector	8.000,00	C-202	Clarice Lispector	2.000,00

Fonte: autoria própria.

Projeto de banco de dados relacional

- Um projeto de banco de dados relacional consiste em uma coleção de esquemas de relação que possibilita a recuperação dos dados sem dificuldade e o armazenamento de dados sem redundância. Quando o projeto respeita uma forma normal adequada, conseguimos alcançar esses objetivos, consequentemente necessitamos de informações da empresa que estamos modelando. Para entender a empresa, podemos utilizar o diagrama entidade-relacionamento e informações adicionais fornecidas pela empresa.
- Para o sucesso na implantação de um banco de dados em uma empresa é essencial um bom projeto relacional. Quando o projeto é para uma pequena empresa, muitas vezes, o responsável ignora algumas etapas importantes e começa a criação do banco de dados físico, criando tabelas, colunas e índices.
    - Quando o projeto de banco de dados é para uma empresa grande, deve-se respeitar todas as etapas do projeto para que se possa garantir ao usuário todos os requisitos de informações necessárias, com qualidade na disponibilidade desses dados, desempenho e confiabilidade.

Projeto de banco de dados relacional

Consideramos três fases em um projeto de banco de dados:

1. **Modelo conceitual –** usado para representar as regras e os conceitos do negócio e como são associados. Seu maior objetivo é fornecer uma visão geral do negócio envolvendo os desenvolvedores e os usuários que não precisam possuir conhecimentos técnicos. Nessa primeira fase não existe dependência de tecnologia para a implementação de banco de dados, apenas os conceitos de entidades, relacionamentos e atributos são necessários para a modelagem conceitual do negócio.

Nessa fase geramos dois produtos:

- Diagrama de entidade-relacionamento.
- Lista de regras de restrição de integridade.

Projeto de banco de dados relacional

2. **Modelo Lógico** – derivado do modelo conceitual, representa as características das estruturas de dados que serão desenvolvidas levando em conta os limites colocados pelo modelo de dados utilizado para o banco de dados (banco de dados hierárquico, banco de dados de rede, banco de dados relacionais etc.). Nessa fase geramos o projeto lógico observando algumas características:

- Entidades associativas e não relacionamento n:m.
- Chaves primárias.
- Chaves estrangeiras.
- Normalização.
    - Consideração da nomenclatura da empresa.
    - Dicionário de dados (entidades e atributos).

Projeto de banco de dados relacional

3. **Modelo Físico –** é a implementação do modelo lógico utilizando algum banco de dados e considerando os requisitos não funcionais de segurança, desempenho e disponibilidade que foram levantados pelo analista de requisitos. Nessa fase criamos o modelo físico a partir do modelo lógico, utilizando a linguagem SQL (*Structured Query Language*) para definição da estrutura, manipulação e controle dos dados.

Interatividade

Analise as afirmações:

I. O Modelo Relacional de Codd propôs uma nova forma de pensamento que desconectou a estrutura lógica do banco de dados do método de armazenamento físico.

II. O Modelo Relacional utiliza um conjunto de tabelas para caracterizar os dados e as relações entre eles, em que cada tabela pode conter diversas colunas.

III. O Modelo Conceitual é usado para representar as regras e os conceitos do negócio e como são associados.

IV. O Modelo Lógico é derivado do Modelo Conceitual.

V. O Modelo Físico é a implementação do Modelo Lógico utilizando um banco de dados.

Estão corretas:

a) I, II e III.
b) I, II e IV.
c) I, II e V.
d) I, III, IV e V.
e) Todas as afirmações.

Resposta

Analise as afirmações:

I. O Modelo Relacional de Codd propôs uma nova forma de pensamento que desconectou a estrutura lógica do banco de dados do método de armazenamento físico.

II. O Modelo Relacional utiliza um conjunto de tabelas para caracterizar os dados e as relações entre eles, em que cada tabela pode conter diversas colunas.

III. O Modelo Conceitual é usado para representar as regras e os conceitos do negócio e como são associados.

IV. O Modelo Lógico é derivado do Modelo Conceitual.

V. O Modelo Físico é a implementação do Modelo Lógico utilizando um banco de dados.

Estão corretas:

a) I, II e III.
b) I, II e IV.
c) I, II e V.
d) I, III, IV e V.
e) Todas as afirmações.

Linguagem SQL

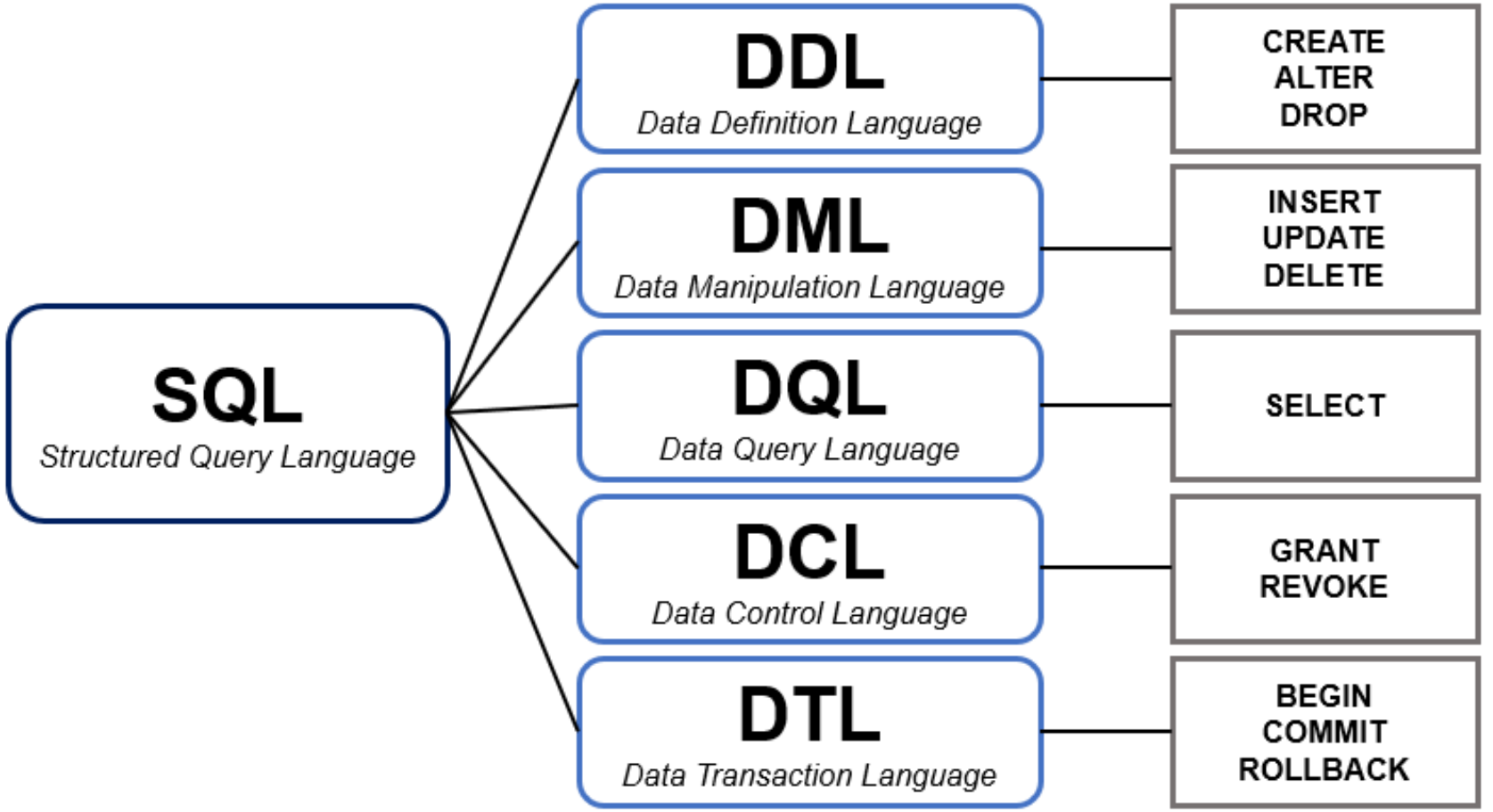
- A SQL (*Structured Query Language*), ou linguagem de consulta estruturada, é um tipo de linguagem de consulta muito utilizada para executar operações em BD relacionais.

A linguagem SQL é dividida em subconjuntos baseando-se nas operações efetuadas sobre um banco de dados, tais como:

- *Data Definition Language* (DDL) – Linguagem de Definição de Dados.
- *Data Manipulation Language* (DML) – Linguagem de Manipulação de Dados.
- *Data Query Language* (DQL) – Linguagem de Consulta de Dados.
    - *Data Control Language* (DCL) – Linguagem de Controle de Dados.
    - *Data Transaction Language* (DTL) – Linguagem de Transação de Dados.

Tipos de linguagem SQL

- Os subconjuntos da SQL e seus principais comandos.



Fonte: autoria própria.

Estruturas básicas do SQL

- **Criando tabelas**

```
USE BD_LOJA
GO

CREATE TABLE Funcionario (
      id_Funcionario integer NOT NULL,
      Nome CHAR(40),
      Logradouro CHAR (40),
      Bairro CHAR(15),
      Salario numeric (11,2)
)
GO

CREATE TABLE Departamento (
      id_Departamento integer NOT NULL,
      Nome VARCHAR(14),
      Localizacao VARCHAR(13))
GO
```

Fonte: autoria própria.

Estruturas básicas do SQL

Alterando tabelas

- Adicionando a coluna idade na tabela funcionario

```
ALTER TABLE Funcionario
ADD idade int
GO
```

- Modificando o tamanho da coluna Localizacao da tabela Departamento de VARCHAR(13) para VARCHAR(20)

```
ALTER TABLE Departamento
ALTER COLUMN Localizacao VARCHAR(20)
GO
```

- Excluindo a tabela Departamento do Banco de Dados

```
DROP TABLE Departamento
GO
```

Estruturas básicas do SQL

Consultando os dados

- Para listar todos os atributos de todos os funcionários.

```
SELECT * FROM Funcionario
GO
```

- Para listar os nomes e os salários de todos os funcionários.

```
SELECT Nome, Salario FROM Funcionario
GO
```

  - Para forçar a eliminação de nomes duplicados na tabela funcionario.

```
SELECT distinct NOME FROM Funcionario
GO
```

Estruturas básicas do SQL

Cláusula WHERE

```sql
-- Nome do Funcionario igual a Tadeu
SELECT Nome FROM Funcionario WHERE Nome = 'Tadeu'

-- Nome do Funcionario diferente a Tadeu
SELECT Nome FROM Funcionario WHERE Nome <> 'Tadeu'

-- Nome do Funcionario maior a Tadeu
SELECT Nome FROM Funcionario WHERE Nome > 'Tadeu'

-- Nome do Funcionario maior igual a Tadeu
SELECT Nome FROM Funcionario WHERE Nome >= 'Tadeu'

-- Nome do Funcionario menor a Tadeu
SELECT Nome FROM Funcionario WHERE Nome < 'Tadeu'

-- Nome do Funcionario menor igual a Tadeu
SELECT Nome FROM Funcionario WHERE Nome <= 'Tadeu'
```

Fonte: autoria própria.

Estruturas básicas do SQL

Cláusula WHERE

```sql
-- Funcionários com nome entre João e Tadeu (inclusivo)
SELECT Nome FROM Funcionario WHERE Nome between 'João' and 'Tadeu'

-- Funcionários com nome entre João e Tadeu (não inclusivo)
SELECT Nome FROM Funcionario WHERE Nome not between 'João' and 'Tadeu'


-- Funcionários com os nomes iguais João e Tadeu
SELECT Nome FROM Funcionario WHERE Nome in ('João','Tadeu')

-- Funcionários com os nomes diferentes João e Tadeu
SELECT Nome FROM Funcionario WHERE Nome not in ('João','Tadeu')
```

```sql
-- Funcionários com nomes iniciados pela letra 'J'
SELECT Nome FROM Funcionario WHERE Nome like 'J%'

-- Funcionários com Nome igual a Tadeu ou Vania
SELECT Nome FROM Funcionario WHERE Nome = 'Tadeu' or Nome = 'Vania'

-- Funcionários com nomes iniciados pela letra 'J' e Salário maior que 1000
SELECT Nome FROM Funcionario WHERE Nome like 'J%' and Salario > 1000
```

Estruturas básicas do SQL

Operações de conjunto

- Para encontrar o nome de todos os Funcionários e o nome de todos os Clientes eliminando os nomes duplicados.

```
(select Nome
from Funcionario)
union
(select Nome
from Cliente)
```

- Para encontrar o nome de todos os Funcionários e o nome de todos os Clientes mantendo os nomes duplicados utilizamos o UNION ALL.

```
(select Nome
from Funcionario)
union all
(select Nome
from Cliente)
```

Estruturas básicas do SQL

Operações de conjunto

- Para encontrar o nome de todos os Funcionários que também possuem nome na tabela Clientes eliminando os nomes duplicados.

```
(select Nome
from Funcionario)
intersect
(select Nome
from Cliente)
```

Estruturas básicas do SQL

Operações de conjunto

- Para encontrar o nome de todos os Funcionários que não possuem o nome na tabela Clientes eliminando os nomes duplicados.

```
(select Nome
from Funcionario)
except
(select Nome
from Cliente)
```

Estruturas básicas do SQL

Funções agregadas

```sql
-- Apresenta a quantidade de funcionários
SELECT count(id_Funcionario) as QtTotal
FROM Funcionario

-- Apresenta o maior Salário
SELECT max(Salario) as Maior_Salario
FROM Funcionario

-- Apresenta o menor Salário
SELECT min(Salario) as Menor_Salario
FROM Funcionario
```

```sql
-- Apresenta a soma dos Salários
SELECT sum(Salario) as Total
FROM Funcionario

-- Apresenta a média de Salário dos Funcionários
SELECT avg(Salario) as Media
FROM Funcionario
```

Estruturas básicas do SQL

Valores nulos

- Para localizar os funcionários que não possuem Setor

```
SELECT Nome
FROM Funcionario
WHERE Setor is null
```

- Para localizar os funcionários que possuem Setor

```
SELECT Nome
FROM Funcionario
WHERE Setor is not null
```

Estruturas básicas do SQL

Views

- Criação

```
CREATE VIEW VW_Analise AS
(SELECT CPF, Nome, Vencimento, Valor
FROM Fatura)
GO
```

- Utilização

```
SELECT Valor
FROM VW_Analise
WHERE CPF = '11111111111'
GO
```

- Eliminar

```
DROP VIEW VW_Analise
GO
```

Estruturas básicas do SQL

Modificação de dados

- EXCLUSÃO

```
DELETE FROM Funcionario

DELETE FROM Funcionario
WHERE id_Funcionario = 3
```

- INSERÇÃO

```
INSERT INTO Funcionario
VALUES (1,'Tadeu', 'Av. Paulista', 'Moema', 1500, 32);
GO
```

- ATUALIZAÇÃO

```
UPDATE Funcionario
SET Bairro = 'Ibira'
WHERE id_Funcionario = 3;
GO
UPDATE Funcionario
SET Salario = Salario + 100
WHERE Salario <= 1000;
GO
```

Interatividade

Analise as afirmações:

I. O subconjunto DDL nos permite criar, atualizar e remover tabelas e elementos associados.

II. O subconjunto DML é utilizado para realizar inclusões, consultas, alterações e exclusões de dados existentes em registros.

III. O subconjunto DQL permite ao usuário executar uma consulta nas tabelas do banco de dados.

IV. O subconjunto DCL gerencia aspectos de autorização de dados e liberação de usuários para acessos de visualização ou manipulação de dados dentro do banco de dados.

V. O subconjunto DTL controla a execução de transações em um banco de dados.

Estão corretas:

a) I, II e III.
b) I, II e IV.
c) II, III e V.
d) III, IV e V.
e) Todas as afirmações.

Resposta

Analise as afirmações:

I. O subconjunto DDL nos permite criar, atualizar e remover tabelas e elementos associados.

II. O subconjunto DML é utilizado para realizar inclusões, consultas, alterações e exclusões de dados existentes em registros.

III. O subconjunto DQL permite ao usuário executar uma consulta nas tabelas do banco de dados.

IV. O subconjunto DCL gerencia aspectos de autorização de dados e liberação de usuários para acessos de visualização ou manipulação de dados dentro do banco de dados.

V. O subconjunto DTL controla a execução de transações em um banco de dados.

Estão corretas:

a) I, II e III.
b) I, II e IV.
c) II, III e V.
d) III, IV e V.
e) Todas as afirmações.

SQL Avançada

Restrições de integridade

- Para assegurar a integridade referencial entre as tabelas do banco de dados, precisamos criar dois objetos e associá-los à tabela: uma chave primária e uma chave de estrangeira (referência). Ambos são restrições que garantem a qualidade dos dados no banco de dados.

- A chave primária (*primary key*) assegura que não haverá duplicidade de linhas com os valores idênticos para as colunas da chave primária.

- A chave estrangeira (*foreign key*) assegura que uma referência a um dado em outra tabela será sempre válida. A chave estrangeira sempre aponta para uma chave primária de outra tabela com um valor que existe na outra tabela.

Criar chave primária para uma tabela (PRIMARY KEY)

A tabela filial possui uma chave primária composta de dois campos id_empresa e id_filial. Essa chave assegura que não teremos duplicidades de empresa e filial no cadastro. O código de criação ficaria assim:

```
CREATE TABLE Filial(
      id_filial integer NOT NULL,
      id_empresa integer NOT NULL,
      cod_tipo varchar(20),
      dt_cadastro date,
      CONSTRAINT PK_FILIAL PRIMARY KEY(id_empresa,id_filial))
GO
```

SQL Avançada

Criar chave estrangeira para uma tabela (FOREIGN KEY)

A chave estrangeira na tabela Filial apresenta a referência das tabelas Empresa e Filial, isto é, Empresa.id_empresa = Filial.id_empresa. O código de criação ficaria assim:

```
CREATE TABLE Empresa(
      id_empresa integer NOT NULL,
      nome varchar(80) NOT NULL,
CONSTRAINT PK_EMPRESA PRIMARY KEY(id_empresa))
GO

CREATE TABLE Filial(
      id_filial int NOT NULL,
      id_empresa int NOT NULL,
      cod_tipo varchar(20),
      dt_cadastro date,
CONSTRAINT PK_FILIAL PRIMARY KEY (id_empresa, id_filial),
CONSTRAINT FK_EMPRESA FOREIGN KEY(id_empresa) REFERENCES Empresa(id_empresa))
GO
```

SQL Avançada

Restrição CHECK

- A restrição CHECK é usada para limitar o intervalo de valores que podem ser colocados em uma coluna. Se você definir uma restrição CHECK em uma coluna, ela permite apenas determinados valores para aquela coluna. Quando você especifica uma restrição CHECK em uma tabela, ela pode limitar os valores de algumas colunas com base nos valores de algumas outras colunas.
- A restrição CHECK garante a qualidade dos dados adicionados/atualizados na tabela. Recomenda-se sempre usar a restrição CHECK principalmente para colunas que representam dados com um determinado conjunto de valores.

SQL Avançada

- Criar uma restrição CHECK no comando de criação de uma tabela

```
CREATE TABLE Funcionario (
      id_Funcionario int NOT NULL,
      Nome CHAR(40),
      Logradouro CHAR (40),
      Bairro CHAR(15),
      Salario numeric (11,2),
      idade int CHECK (idade>=18)
)
GO
```

- Criar uma restrição CHECK para mais de uma coluna

```
CREATE TABLE Funcionario (
      id_Funcionario int NOT NULL,
      Nome CHAR(40),
      Logradouro CHAR (40),
      Bairro CHAR(15),
      Salario numeric (11,2),
      idade int,
      CONSTRAINT CHK_Funcionario CHECK (idade>=18 AND salario > 10)
)
GO
```

SQL Avançada

- Adicionar uma restrição CHECK para uma tabela que já existe no banco de dados.

```
ALTER TABLE Funcionario
ADD CHECK (idade>=18);
```

- Adicionar uma restrição CHECK para mais de uma coluna em uma tabela que já existe no banco de dados.

```
ALTER TABLE Funcionario
ADD CONSTRAINT CHK_Funcionario CHECK (idade>=18 AND salario > 10);
```

  - Remover uma restrição CHECK de uma tabela

```
ALTER TABLE Funcionario
DROP CONSTRAINT CHK_Funcionario;
```

SQL Avançada

Restrição UNIQUE

- Uma restrição UNIQUE garante que todos os valores em uma coluna sejam únicos. As restrições UNIQUE e PRIMARY KEY garantem a exclusividade de uma coluna ou conjunto de colunas.

- Uma restrição PRIMARY KEY tem automaticamente uma restrição UNIQUE. No entanto, pode haver várias restrições UNIQUE por tabela, mas apenas uma restrição PRIMARY KEY por tabela.

SQL Avançada

- Criar uma restrição UNIQUE no comando de criação de uma tabela.

```
CREATE TABLE Funcionario (
      id_Funcionario int NOT NULL,
      CPF CHAR(11) UNIQUE,
      Nome CHAR(40),
      Logradouro CHAR (40),
      Bairro CHAR(15),
      Salario numeric (11,2),
      idade int
)
GO
```

- Criar uma restrição UNIQUE para mais de uma coluna

```
CREATE TABLE Funcionario (
      id_Funcionario int NOT NULL,
      CPF CHAR(11),
      Nome CHAR(40),
      Logradouro CHAR (40),
      Bairro CHAR(15),
      Salario numeric (11,2),
      idade int
      CONSTRAINT UC_Funcionario UNIQUE (id_funcionario, cpf)
)
GO
```

SQL Avançada

- Adicionar uma restrição UNIQUE a uma tabela já existente no banco de dados

```
ALTER TABLE Funcionario
ADD UNIQUE (cpf);
```

- Adicionar uma restrição UNIQUE para mais de uma coluna em uma tabela já existente no banco de dados

```
ALTER TABLE Funcionario
CONSTRAINT UC_Funcionario UNIQUE (id_funcionario, cpf);
```

- Remover uma restrição UNIQUE de uma tabela

```
ALTER TABLE Funcionario
DROP CONSTRAINT UC_Funcionario;
```

SQL Avançada

SQL embutida

- A linguagem SQL possui uma linguagem de consulta declarativa muito poderosa. Escrever consultas em linguagem de programação é muito mais complexo que escrever consultas em SQL.
- A SQL embutida ocorre sempre que colocamos código SQL em uma linguagem de programação. A linguagem está embutida no programa e é responsável por gerar o SQL e enviá-lo ao banco de dados para fazer o trabalho. As instruções são executadas pelo programa de computador e todas as decisões necessárias para a consulta ao banco de dados. O banco de dados receberá da aplicação apenas uma SQL Estática na maioria dos casos e, por sua vez, a aplicação ganha independência do banco de dados.

Após o programa ser compilado pelo compilador da linguagem, usamos a instrução EXEC SQL para identificar requisições de SQL embutidas para o pré-processamento. A sintaxe do comando é:

EXEC SQL <instruções SQL embutidas> END-EXEC

SQL Avançada

SQL dinâmica

- SQL Dinâmica é um recurso bastante usado por programadores, no qual é possível construir e submeter comandos SQL em tempo de execução. Dessa forma, o programador pode adequar as necessidades do seu negócio ou do seu usuário ao montar a SQL. Por exemplo: depois de executar o comando enviado ao banco de dados, o banco retorna as informações nas variáveis do programa e o programador poderá manipular em tempo de execução.
- A partir de 1999, a linguagem SQL aceita funções e procedimentos, que podem ser escritos na própria SQL ou em uma linguagem de programação externa. As funções são muito úteis com tipos de dados especializados, como por exemplo: imagens (funções para verificar semelhança entre imagens) e objetos geométricos (funções para verificação se existe sobreposição de polígonos). Alguns sistemas de banco de dados aceitam funções com valor de tabela e que podem retornar uma outra tabela como resultado. Um grande conjunto de instrução também é aceito: loops, if-the-else e atribuições.

SQL Avançada

- Podemos definir uma função que, dado o ID de um cliente, retorne a contagem do número de pedidos pertencentes a esse cliente.

```
CREATE FUNCTION dbo.conta_pedido (@id_cliente int)
      RETURNS integer
      AS
      BEGIN
            DECLARE @contagem_c integer;
                  SELECT @contagem_c = count(*)
                  FROM PEDIDO
                  WHERE Pedido.id_cliente = @id_cliente
            RETURN @contagem_c;
      END
GO
```

Podemos encontrar os nomes dos clientes e a quantidade de pedidos feitos por cada um

```
SELECT Nome, dbo.conta_pedido(id_cliente) as 'Qtde_Pedido'
FROM CLIENTE
```

SQL Avançada

A função conta_pedido poderia ser escrita como um procedimento:

```
CREATE PROCEDURE dbo.proc_conta_pedido (@id_cliente integer, @contagem_c integer OUTPUT)
      AS
      BEGIN
            SELECT @contagem_c = count(*)
            FROM PEDIDO
            WHERE Pedido.id_cliente = @id_cliente
      END
```

- Para chamar um procedimento no SQL ou no SQL embutido, usamos a instrução CALL

```
DECLARE @contagem int;
EXECUTE dbo.proc_conta_pedido 2, @contagem OUTPUT;
PRINT  @contagem
```

- A linguagem SQL permite mais de um procedimento/função com o mesmo nome (*overloading*) desde que o número de argumentos ou os tipos sejam diferentes.

Interatividade

Analise as afirmações:

I. A chave primária (*primary key*) assegura que não haverá duplicidade de linhas com os valores idênticos para as colunas da chave primária.

II. A chave estrangeira (*foreign key*) assegura que uma referência a um dado em outra tabela será sempre válida.

III. Usamos a restrição CHECK para limitar o intervalo de valores que colocados em uma coluna.

IV. A SQL dinâmica ocorre sempre que colocamos código SQL em uma linguagem de programação.

V. A SQL dinâmica é um recurso bastante usado por programadores, mas não é possível construir e submeter comandos SQL em tempo de execução.

Estão corretas:

a) I, II e III.
b) I, II e IV.
c) I, II e V.
d) III, IV e V.
e) Todas as afirmações.

Resposta

Analise as afirmações:

I. A chave primária (*primary key*) assegura que não haverá duplicidade de linhas com os valores idênticos para as colunas da chave primária.

II. A chave estrangeira (*foreign key*) assegura que uma referência a um dado em outra tabela será sempre válida.

III. Usamos a restrição CHECK para limitar o intervalo de valores que colocados em uma coluna.

IV. A SQL dinâmica ocorre sempre que colocamos código SQL em uma linguagem de programação.

V. A SQL dinâmica é um recurso bastante usado por programadores, mas não é possível construir e submeter comandos SQL em tempo de execução.

Estão corretas:

a) I, II e III.
b) I, II e IV.
c) I, II e V.
d) III, IV e V.
e) Todas as afirmações.

Referências

- CODD, E. F. A relational model of data for large shared data banks. *Communications of the ACM*. Vol. 6, June 1970, p. 377-387. Disponível em: https://doi.org/10.1145/362384.362685. Acesso em: 31 out. 2022.

- SILBERSCHATZ, A.; SUDARSHAN, H. F. *Sistema de Banco de dados*. Rio de Janeiro: Elsevier, 2020.

- TANENBAUM, A. S.; WETHERALL, D. J. *Redes de computadores*. Tradução Daniel Vieira. 5. ed. Rio de Janeiro: Pearson Prentice Hall, 2011.

ATÉ A PRÓXIMA!