SQL

INSTITUTO FEDERAL
Piauí

A LINGUAGEM SQL

- SQL - **Structured Query Language.**

- Foi definida nos laboratórios de pesquisa da IBM em San Jose, California, em 1974.

- Teve seus fundamentos no **modelo relacional**.

- Sua primeira versão recebeu o nome de SEQUEL - Structured English Query Language

A LINGUAGEM SQL

- Órgãos como **ANSI** e **ISO adotaram** a **SQL** como o **padrão oficial** de linguagem em ambiente relacional.

- O ANSI publicou as padronizações SQL ANSI-89 e ANSI-92.

- Revisões da SQL: SQL99 (SQL 3)

A LINGUAGEM SQL

- **É uma linguagem usada em SGBDs para:**
  - Definir estruturas de dados (Ex: criar tabelas)
  - Modificar dados no BD (Ex: inserir e alterar dados)
  - Especificar restrições de segurança (Ex: privilégios de acesso)
  - Realizar consultas

- **Não é uma linguagem *case-sensitive***

A LINGUAGEM SQL

- **Definição de Dados**: através da DDL possibilita a definição da estrutura e organização dos dados

- **Manipulação de Dados**: através da DML possibilita a manipulação dos dados armazenados, compreendendo inclusão, consulta, alteração e eliminação..

- **Controle de Acesso**: protege os dados de manipulações não autorizadas, através de comandos de autorização de acesso.

- **Integridade dos Dados**: define as regras de integridade dos dados contra corrupções, inconsistências e falhas do sistema.

- **Controle de Transações**: inclui comandos que controlam a especificação do início e do fim das transações.

A LINGUAGEM SQL

- **Independência de fabricante**: está incorporada em quase todos os SGBDs em seu padrão ANSI, com as extensões proprietárias de cada fabricante.

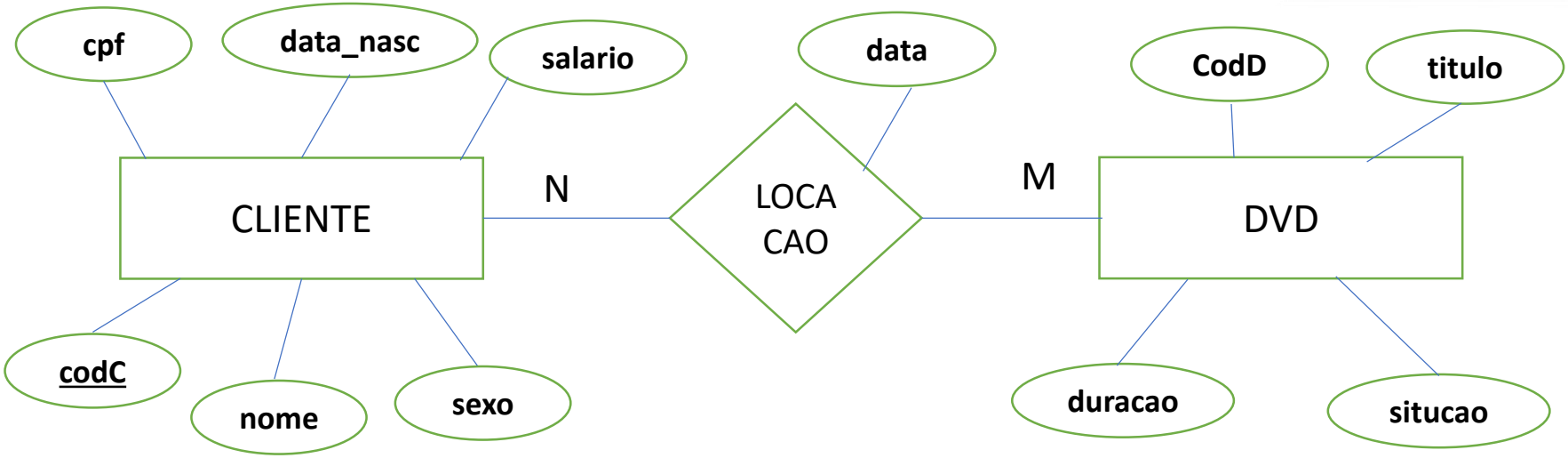
- **Portabilidade entre computadores**: pode ser usada desde um PC até um mainframe.

- **Redução de custos com treinamento**: as aplicações podem migrar de ambiente com custo reduzido em treinamento.

- **Facilidade no entendimento**: oferece um rápido entendimento, com comandos escritos em um inglês estruturado de alto nível.

- **Múltiplas visões de dados**: possibilita levar diferentes visões dos dados a diferentes usuários.

A LINGUAGEM SQL



- **Tabela Clientes (CodC, nome, cpf, data_nasc, sexo, salario)**
- **Tabela Dvd (CodD, titulo, genero, duracao, situacao)**
- **Tabela Locacoes (CodC, CodD, data)**

A LINGUAGEM SQL

- Em todos os exemplos a seguir, será utilizado o exemplo de uma locadora de DVD

  - **Tabela Clientes (CodC, nome, cpf, data_nasc, sexo, salario)**
  - **Tabela Dvd (CodD, titulo, genero, duracao, situacao)**
  - **Tabela Locacoes (CodC, CodD, data)**

- Observação: os nomes das tabelas e dos atributos não podem conter acentos ou espaços em branco

CRIANDO TABELAS

Bancos

Codigo	Nome
001	Banco do Brasil
033	Santander
237	Bradesco
341	Itaú

Tabela 1: Bancos Comerciais.

Pessoas

CPF	Nome
86277635697	José da Silva
88208811874	Manoel da Silva
66516764743	Maria dos Santos

Tabela 2: Pessoas.

Conta_Corrente

Banco	Pessoa	Numero
033	86277635697	98876788
237	86277635697	96645727
341	66516764743	9102947
001	88208811874	8120938

- O **nome** de uma **tabela** em um banco de dados deverá ser **único** para cada proprietário;

- Cada **coluna** deverá ser criada através da especificação do seu **nome**, **tipo** e **tamanho** do **dado** que irá armazenar

- O **nome** de uma **coluna** deverá ser **único** dentro de cada **tabela** podendo, entretanto, existir **colunas** com o **mesmo nome** em **tabelas diferentes**.

- Uma **tabela** representa uma **entidade** do **banco** de **dados** onde **cada linha equivale** a uma **ocorrência** e cada **coluna equivale** a um **atributo** dessa **entidade**.

CRIANDO TABELAS - SINTAXE

```
CREATE TABLE tabela(
        atributo1 tipo1,
        atributo2 tipo2,
        ...,
        restrições de integridade
  )
```

CRIANDO TABELAS

```
CREATE TABLE cliente(
        CodC int not null identity,
        nome varchar(80) not null,
        cpf char(11) not null,
        data_nasc date,
        sexo char(1),
        salario numeric (9,2),
        PRIMARY KEY (CodC),
        UNIQUE (cpf),
        CHECK (sexo in ('M','F')),
        CHECK (salario > 0)
)
```

CRIANDO TABELAS

- **Not null** – indica que o **atributo** deverá ser **obrigatoriamente informado** pelo usuário na hora de inserir dados na tabela. (Todos os campos, por default, aceitam valores nulos)

- **Identity** – indica que o **atributo** será **preenchido automaticamente** com valores auto-incrementados (não é possível definir um valor para esse atributo na hora de inserir dados)

- **Varchar** (tamanho) – tipo de **String** com **tamanho** de **armazenamento variável**, de acordo com os valores inseridos pelo usuário (o tamanho indicado na criação da tabela é a quantidade máxima de caracteres que poderão ser armazenados no campo)

CRIANDO TABELAS

- **Char** (tamanho) – tipo de **String** com **tamanho** de **armazenamento fixo**, de acordo com a quantidade de caracteres definidos na criação da tabela.

- **Date** – tipo data no **formato ano-mes-dia** (Obs: no SQL-Server, o tipo é *datetime*)

- **Numeric** (n,d) – tipo **numérico** que aceita **valores reais** (n indica o total de números e d indica a quantidade de decimais)

- **Primary Key** – restrição de **integridade** que define a **chave primária** da tabela (se a chave for composta, os nomes devem ser separados por vírgulas)

CRIANDO TABELAS

- **Unique** – restrição de integridade que indica que um campo **não** poderá **receber valores repetidos** na tabela (ou seja, dois registros não podem ter o mesmo valor para esse campo)

- **Check** – restrição de integridade que indica **condições** para o **preenchimento** de um **campo**

- **In** (conjunto de valores) – indica que o **valor** de um determinado **atributo** deve estar **presente** no **conjunto** de **valores definido**, para que os dados possam ser inseridos na tabela

CRIANDO TABELAS

```
CREATE TABLE dvd (
        CodD int not null identity,
        titulo varchar(40) not null,
        genero varchar(15),
        duracao time,
        situacao varchar(12) default (‘Disponível’),
        PRIMARY KEY (CodD),
        CHECK (situacao in (‘Alugada’,‘Disponível’))
)
```

CRIANDO TABELAS

- **Time** – **tipo tempo** no formato hora:minuto:segundo (Obs: no SQL-Server, o tipo é *datetime*)

- **Default** – indica um **valor** que será **armazenado** no **atributo caso não seja informado** outro valor **pelo usuário**

CRIANDO TABELAS

```
CREATE TABLE locacoes (
        CodC int not null,
        CodD int not null,
        data date,
        FOREIGN KEY (CodC) REFERENCES clientes,
        FOREIGN KEY (CodD) REFERENCES dvd,
         PRIMARY KEY (CodC, CodD)
)
```

CRIANDO TABELAS

- **FOREIGN KEY** – restrição de integridade que **define** uma **chave estrangeira** para a tabela

- Obs:
  - Para que um atributo seja **chave estrangeira** de uma tabela, é **necessário** que ele seja **chave primária** da **tabela referenciada**
  - O **nome** do **campo** na tabela que terá a **chave estrangeira não precisa** ser o **mesmo** do **campo** na **tabela referenciada** (Exemplo: o campo CodD em locacoes poderia ser substituído por CodDvd), mas deve ser do mesmo tipo

COMANDOS SQL - INSERÇÃO

Inserção de dados nas tabelas

INSERT INTO tabela(atributo1,atributo2,...)

VALUES(valor1,valor2,...)

Ou

INSERT INTO tabela

VALUES(valor1,valor2,...)

COMANDOS SQL - INSERÇÃO

```
INSERT INTO clientes (nome,cpf,data_nasc,sexo,salario)
VALUES ('Ana Moura', '8245738', '1979-10-02', 'F' , 650.39)

INSERT INTO dvd (titulo,genero,duracao)
VALUES ('Matrix', 'Ficção', '02:30:00')

INSERT INTO locacoes
VALUES (1, 1, '2003-11-11')
```

COMANDOS SQL - INSERÇÃO

- A **lista** de **atributos** é usada para indicar **que campos** da **tabela** devem ser **preenchidos**, e com que **valores**.

- **Se não for incluída**, o **BD tentará preencher todos** os **campos** da tabela na **sequência** em que **foram criados**.

- Portanto, a **lista** é **obrigatória quando** alguns **campos não forem preenchidos**, **ou** quando a **ordem** dos **valores** estiver **alterada**.

COMANDOS SQL - INSERÇÃO

Ex: INSERT INTO dvd VALUES (‘X-Men’, ‘Ação’)

Seria um comando incorreto pois não há como saber a que atributos se referem os valores.

O **correto** seria:

INSERT INTO dvd (titulo,genero) VALUES (‘X-Men’,‘Ação’)

Os **campos não informados** seriam **preenchidos** com ***Null*** (se não tiverem sido definidos como *not null*), **ou** com **valores *default*** definidos na criação da tabela.

COMANDOS SQL - INSERÇÃO

- Campos definidos como **IDENTITY não podem assumir valores** informados pelo usuário, e não precisam ser incluídos na lista de atributos da tabela.

- Os valores desse campo vão sendo **incrementados automaticamente** e não são reaproveitados (Ex: mesmo que o DVD de código 5 seja excluído do banco, nenhum outro DVD receberá esse código)

- No SQL Server, o valor inicial de campos IDENTITY é 1 e o incremento tem valor 1.

COMANDOS SQL - INSERÇÃO

- Valores do tipo **char**, **varchar**, **date** e **time** (ou *datetime* no SQL-Server) devem ser **representados entre apóstrofos** (aspas simples ‘ ’).

- No SQL-Server, o formato **padrão** para **datas** é **YYYY-MM-DD** (ano-mes-dia) e para **horas** é **HH:MM:SS** (hora-minuto-segundo)

- As **casas decimais** dos números devem ser **separadas** por **pontos**, em vez de vírgulas

- Valores do tipo **varchar** podem conter **acentos** e **espaços** em **branco**

COMANDOS SQL - ATUALIZAÇÃO

Atualização de dados nas tabelas

UPDATE tabela
SET atributo = valor
WHERE condicao

COMANDOS SQL - ATUALIZAÇÃO

- Mudar o salário do cliente com código 1 para 1400

```
UPDATE clientes
SET salario = 1400
WHERE CodC = 1
```

- Mudar a situação do DVD de código 1 para alugada

```
UPDATE dvd
SET situacao = 'alugada'
WHERE CodD = 1
```

COMANDOS SQL - ATUALIZAÇÃO

- A cláusula **WHERE** é **opcional** no comando **UPDATE**. **Se não** for **informada**, a **atualização** será realizada em **toda** a **tabela**

- **Ex**: O comando abaixo muda o preço de todos os DVDs cadastrados para 2.20

UPDATE dvd

SET preco = 2.20

COMANDOS SQL – EXCLUSÃO

Exclusão de dados das tabelas

DELETE FROM tabela
WHERE condição

COMANDOS SQL - EXCLUSÃO

- Apagar cadastros de todos os clientes do sexo masculino

```
DELETE FROM clientes
WHERE sexo = 'M'
```

- Apagar cadastros de todos os DVDs de terror

```
DELETE FROM dvd
WHERE genero = 'terror'
```

COMANDOS SQL - EXCLUSÃO

- A cláusula **WHERE** é opcional no comando **DELETE**. **Se não** for **informada**, a **exclusão** será realizada em **toda** a **tabela**

  Ex: O comando abaixo exclui todas as locacoes cadastradas

  DELETE FROM locacoes

- O comando **DELETE exclui** os **dados**, mas não exclui a tabela do BD.
  - **Para excluir** a **tabela inteira** (dados e estrutura), o comando é:

    DROP TABLE tabela

    DROP TABLE locacoes

COMANDOS SQL – ALTERAÇÃO DE TABELAS

- Alteração de tabelas para inclusão ou exclusão de campos

  - **Inclusão**

    ALTER TABLE tabela

    ADD atributo tipo restrição_integridade

  - **Exclusão**

    ALTER TABLE tabela

    DROP COLUMN atributo

COMANDOS SQL – ALTERAÇÃO DE TABELAS

- Inclusão do campo ano na tabela DVD

  ALTER TABLE dvd

  ADD ano int

- Exclusão do campo sexo da tabela Clientes

  ALTER TABLE clientes

  DROP COLUMN sexo