infoSiS
DESENVOLVIMENTO DE SISTEMAS
Rua São Paulo, 1726, Centro
Edifício Marajó 1º andar, sala 14
Fernandópolis-SP / CEP: 15600-000
Tel: (017) 3463-2211

Sumário

Rua São Paulo, 1726, Centro
Edifício Marajó 1º andar, sala 14
Fernandópolis-SP / CEP: 15600-000
Tel: (017) 3463-2211

1. Linguagem de Banco de Dados (SQL)

1.1. Conceito/Definição

A SQL consiste na sigla de abreviação para Structured Query Language, ou traduzida para o português "Linguagem de Consulta Estruturada", essa linguagem baseia-se em uma forma de pesquisa declarativa para bancos de dados relacionais.

Possui vários grupos de comando, aqui destacaremos os principais e mais utilizados sendo eles os grupos DDL, DML, DQL e o DCL.

1.2. Banco de Dados Relacionais

Esse tipo de Banco de Dados organiza seus dados através de relações, onde, cada relação consiste em uma tabela, e cada coluna consiste em atributos e cada linha consiste em tuplas ou elementos da relação.

Um conceito importante é o de chave, pois, através dele se torna possível a identificação e diferenciação de uma tupla, além de acelerar o acesso aos elementos sendo possível a utilização de índices.

Na visão de organização de dados, esse método de organização através de tabelas fornece um conceito simples e familiar para a estruturação dos dados, dentre esses e outros motivos que o conceito de Banco de Dados Relacionais tem tido muito sucesso.

1.3. Operadores

Baseia-se em algumas formas de comparações e interações feitas com dados provindos do Banco de Dados, são três tipos de operadores mais utilizados, sendo eles, **Lógico, Relacionais** e **Aritméticos.**

1.3.1. Operadores Lógicos

Os operadores lógicos são utilizados para definição de condições para a produção de um resultado único, sendo que uma linha só poderá ser retornada caso o resultado do global da condição for verdadeiro.

Operadores:	Descrição:
AND	Retornará caso **ambas** as condições sejam verdadeiras.
OR	Retornará caso **uma** das condições seja verdadeira.
NOT	Retornará caso a condição **negada** seja verdadeira.

1.3.2. Operadores Relacionais

Operadores Relacionais são usados para descrever uma condição que faz comparação de uma expressão a outra expressão. Abaixo a tabela de Operadores Relacionais:

Operadores:	Descrição:
=	Igual a
>	Maior que

Rua São Paulo, 1726, Centro
Edifício Marajó 1º andar, sala 14
Fernandópolis-SP / CEP: 15600-000
Tel: (017) 3463-2211

>=	Maior ou igual a que
<=	Menor ou igual
<>	Diferente de

1.3.3. Operadores Aritméticos

São os operadores aritméticos comumente utilizados, os quais podem ser utilizados pelo SQL, porém somente poderão ser utilizados em colunas do tipo numérico.

Operadores:	Descrição:
+	Soma
-	Subtração
*	Multiplicação
/	Divisão

1.4. Tipologia de Comandos SQL

Os comandos de manipulações de dados SQL foram divididos em Tipologias Textuais, facilitando a forma de utilização e estudo, as quais seguem abaixo:

1.4.1. Comandos DDL

A abreviação DDL foi criada a partir da denominação de um grupo de comandos da linguagem SQL, a qual consiste em "Data Definition Language", ou traduzido "Linguagem de Definição de Dados".

Esse grupo de comandos é baseado principalmente em ações que permitem ao usuário definir novas tabelas e elementos. A maioria dos visualizadores de Banco de Dados tenta facilitar muito a vida do desenvolvedor em relação a esses comandos, pois cada dia mais esses visualizadores estão fazendo de uma forma mais dinâmica o que antigamente era estaticamente comandos.

Comando Create (Criação de Tabelas):
Comando responsável pela criação de tabelas dentro do banco de dados.

Sintaxe:

```
CREATE TABLE nometabela(
  campotabela SERIAL NOT NULL,
  campotabela2 VARCHAR(45) NOT NULL,
  campotabela3 VARCHAR(45) NOT NULL,
  campotablea4 INTEGER NOT NULL DEFAULT 0,
  CONSTRAINT pk_tabela PRIMARY KEY (`campotabela`)
);
```

Comando Alter (Alteração de Tabelas):
Comando responsável pela alteração de componentes de uma tabela dentro do banco de dados.

Rua São Paulo, 1726, Centro
Edifício Marajó 1º andar, sala 14
Fernandópolis-SP / CEP: 15600-000
Tel: (017) 3463-2211

Sintaxe:
ALTER TABLE nometabela **ADD** colunanova int;

Comando Drop (Exclusão de Tabelas):
Comando responsável pela exclusão de tabelas dentro do bando de dados.

Sintaxe:
DROP TABLE nometabela;

Comando Truncate (Exclusão de Identidade Tabela):
Comando responsável por excluir todo o conteúdo de uma tabela e redefinir sua identidade para o valor inicial. Essa exclusão envolve também os espaços destinados aos registros. Trata-se de uma operação que não pode ser revertida.

Sintaxe:
TRUNCATE nometabela;

Comando Comment (Comentários)
Comando responsável por criar um comentário explicativo ou impedir a execução de uma linha de SQL pelo sistema.

Sintaxe:
COMMENT ON TABLE nometabela **IS** 'Tabela Responsavel por Armazenar as Informações dos Usuários';

1.4.2. Comandos DML

A abreviação DML foi criada a partir da denominação de um grupo de comandos da linguagem SQL, a qual consiste em "Data Manipulation Language".

Esse grupo de comandos é baseado principalmente em ações de manipulação de registros de uma tabela, esse grupo é formado por:

Comando Insert (Inserção):
Comando responsável pela inserção de dados em uma tabela.

Sintaxe:
INSERT INTO nometabela(campo1, campo2, campo3) **VALUES** ('valor1', 'valor2', valor3');

Regras:
- valores numéricos não utilizam as aspas simples, em valores de texto e data é obrigatório a utilização das aspas simples.

- valores de data além de utilizar as aspas, devem ser inseridas na forma **inglesa curta.** Ex: 2012-05-31.

- valores numéricos com decimais devem ser inseridos sem separação de milhares e com uma virgula na separação decimal (método inglês). Ex: 1878.90.

Rua São Paulo, 1726, Centro
Edifício Marajó 1º andar, sala 14
Fernandópolis-SP / CEP: 15600-000
Tel: (017) 3463-2211

Comando Delete (Remoção)
Comando responsável pela remoção de dados de uma tabela.

Sintaxe:
DELETE FROM nometabela **WHERE** comparacao=1;

Regras:
- quanto aos dados utilizam-se as mesmas normas do insert.

Recomendações:
- esse comando por ser um comando de remoção de dados, se torna necessário um maior cuidado, pois, remover dados de um banco é uma ação de alto risco para os dados, recomenda-se um backup sempre quando for ser utilizado.

- utilização mais comum desse comando sempre é acompanhado da instrução **WHERE**.

Comando Update (Atualização):
Comando responsável pela atualização de dados de uma tabela.

Sintaxe:
UPDATE nometabela **SET** nomecampo='valorcamponovo', nomecampo2='valor camponovo2' **WHERE** comparação=1;

Regras:
- quanto aos dados utilizam-se as mesmas normas do insert.

Recomendações:
- esse comando por ser um comando de remoção de dados, se torna necessário um maior cuidado, pois, remover dados de um banco é uma ação de alto risco para os dados, recomenda-se um backup sempre quando for ser utilizado.

- utilização mais comum desse comando sempre é acompanhada da instrução **WHERE**.

1.4.3. Comandos DQL

A abreviação DQL foi criada a partir da denominação de um grupo de comandos da linguagem SQL, a qual consiste em "Data Query Language".

Esse grupo de comandos é baseado principalmente em ações que permitem ao usuário definir criar consultas de tabelas no banco de dados, esse grupo consiste em um único comando o **SELECT**, esse comando em algumas bibliografias também é constatado como integrante do grupo DML, pelo fato de ser um manipulador de dados também, sendo assim levando em consideração essas bibliografias pode se presumir que o SELECT seja um comando DQL e DML.

Comando Select (Consulta de Dados):
Comando responsável pela consulta de dados de uma ou varias tabelas no banco de dados.

Rua São Paulo, 1726, Centro
Edifício Marajó 1º andar, sala 14
Fernandópolis-SP / CEP: 15600-000
Tel: (017) 3463-2211

Sintaxe:
SELECT coluna, coluna1, coluna2 **FROM** nometabela <CLAUSULAS>.

Principais Cláusulas:
As cláusulas complementam o SELECT adicionando condições e outros recursos a sua consulta, as cláusulas disponíveis são:

WHERE: responsável por determinar a condição que será aplicada na consulta, está expressão poderá conter operadores lógicos, aritméticos e relacionais.

ORDER BY: responsável por ordenar os dados retornados na consulta criada através do SELECT, onde pode se definir uma ou mais coluna como referência de ordem.

GROUP BY: responsável por agrupar os dados contidos na consulta

1.4.4. Comandos DCL

A abreviação DCL foi criada a partir da denominação de um grupo de comandos da linguagem SQL, a qual consiste em "Data Control Language". Esse grupo é responsável pelo controle de aspectos de autorização de manipulação do banco e licenças de usuários para controlar o acesso aos dados.

Comando Grant (Garantir Privilégios):
Comando responsável por definir os privilégios de um usuário criado no Banco de dados.

Sintaxe:
GRANT ALL ON *.* **TO** user@localhost **IDENTIFIED BY** 'senha';

Recomendação:
- Utilizar somente quando houver criação de usuários novos ou alterações de privilégios.

Comando Revoke (Revogar Privilégios)
Comando responsável por revogar ou barrar privilégios de um usuário no Banco de dados.

Sintaxe:
REVOKE <comando> **ON** <tabela> **TO** <usuário>;

Recomendação:
- Utilizar somente quando houver a necessidade de retirar privilégios de algum usuário ou restringir permissão de acesso aos dados armazenados.

1.4.5. Comandos DTL

```
variaveis a e b do tipo inteiro.
a recebe o valor 2 e b recebe o valor 3.
se a for maior que b, executa o "if" e exibe a mensagem
se b for maior que a, executa o "else if" e exibe a mensagem
caso for igual executa o "else", e exibe a mensagem
```

Rua São Paulo, 1726, Centro
Edifício Marajó 1º andar, sala 14
Fernandópolis-SP / CEP: 15600-000
Tel: (017) 3463-2211

A abreviação DTL foi criada a partir da denominação de um grupo de comandos da linguagem SQL, a qual consiste em "Data Transaction Language". Esse grupo é responsável pelo controle e gerenciamento das Transações.

Comando Begin Transaction (Iniciar Transação)
Comando responsável por iniciar uma transação no Banco de Dados.

Sintaxe:
BEGIN TRANSACTION;
<comando>
<CLAUSULAS>

Clausulas que podem ser utilizadas:
COMMIT; - aparece no final daquela transação em específico, fechando o que foi aberto pelo BEGIN TRANSACTION;

ROLLBACK; - reverte uma transação, na prática, restaura o banco de dados desde a última vez que o comando COMMIT foi aplicado, garantindo apenas até onde as alterações já foram salvas.

SAVEPOINT; - define um ponto de salvamento dentro de uma transação, funciona como um "ponto de segurança", tudo que é anterior a ele não pode ser descartado com o comando ROLLBACK, apenas o que vem após, porém com o comando **ROLLBACK TO SAVEPOINT** é possível desfazer até o ponto de salvamento e com comando **RELEASE SAVEPOINT** é possível destruir o ponto de salvamento, mantendo os efeitos dos comandos executados após ter sido estabelecido.

Rua São Paulo, 1726, Centro
Edifício Marajó 1º andar, sala 14
Fernandópolis-SP / CEP: 15600-000
Tel: (017) 3463-2211

2. Modelagem Banco Dados (Diagramas)

2.1. Conceito/Definição

Diagramas são representações gráficas que auxiliam na análise e desenvolvimento de um sistema, existem muitos tipos de diagramas, mas durante a programação utilizaremos os diagramas estruturais, dentre esse tipo será utilizado o "Diagrama de Classe", para a criação de um rascunho de como seria modelado o Banco de Dados,

2.2. Conceitos Principais Utilizados

Apesar de existirem vários tipos de ligações de classe as mais utilizadas são as **Associações** e **Generalizações**.

2.2.1. Associação

A associação e a forma mais simples de ligação de Classe, porém é necessário se definir a multiplicidade dessa associação a qual pode variar, esse tipo de ligação possui também algumas regras para auxiliar o desenvolvimento do Banco de Dados, as quais estão expressas na tabela abaixo.

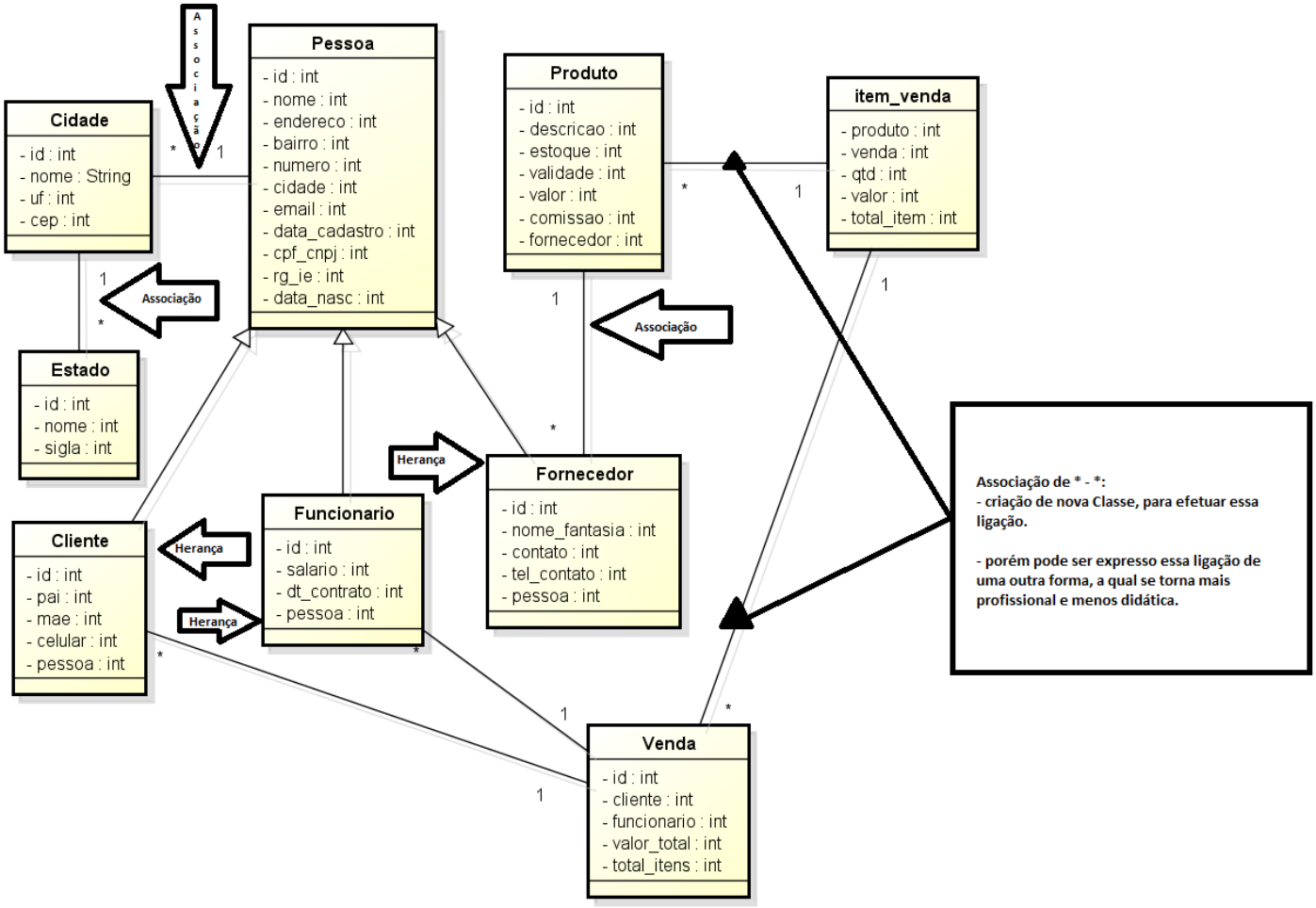
Multiplicidade	Regra Aplicada
1 – 1	O campo de ligação fica na Classe com a multiplicidade **1,** ou seja, qualquer uma delas.
*** – 1**	O campo de ligação fica na Classe com a multiplicidade **1.**
1 – *	O campo de ligação fica na Classe com a multiplicidade **1.**
*** – ***	Ocorre a divisão da ligação, e a criação de uma nova classe para gerenciar esse tipo de ligação, sendo assim, a divisão resultante será equivalente a duas **1 – *.**

2.2.2. Generalização

Mais conhecida como Herança, essa forma de relacionamento pode ser facilmente implementada, pois como o próprio nome diz ela ocorre quando uma classe herda os atributos de uma classe mais genérica, além disso a Herança ou Generalização não possui multiplicidade, fazendo assim com que os filhos automaticamente recebam o campo de ligação, pois, por se tratar de filhos eles receberam as características da classe pai.

Rua São Paulo, 1726, Centro
Edifício Marajó 1º andar, sala 14
Fernandópolis-SP / CEP: 15600-000
Tel: (017) 3463-2211

2.3. Modelando Diagrama da Fase de Projeto



Rua São Paulo, 1726, Centro
Edifício Marajó 1º andar, sala 14
Fernandópolis-SP / CEP: 15600-000
Tel: (017) 3463-2211

3. Linguagem Pascal (Delphi)

3.1. Conceito/Definição

https://blog.betrybe.com/linguagem-de-programacao/delphi-tudo-sobre/

3.2. Estruturas de Controles

São lógicas de comandos criados, afim de que o Sistema faça uma determinada ação através de algumas orientações.

3.2.1. Estruturas de Condições

Esse tipo de Estrutura é amplamente utilizado, pois ela aplica uma condição e dependendo do retorno dela torna-se possível a criação de dois possíveis caminhos, no ambiente de programação ela e conhecida como **Se**, ou em forma de comando mesmo o **If**, abaixo alguns exemplos de utilizações.

Exemplo 1: if... else..

```
var
a, b: Integer;

begin
  a:= 2;
  b:= 3;

  if a > b then
  begin
    ShowMessage('numero a é maior');
  end
  else
  begin
    ShowMessage('numero b é maior');
  end;
end;
```

variaveis a e b do tipo inteiro.
a recebe o valor 2 e b recebe o valor 3.
se a for maior que b, executa o "if" e exibe a mensagem
se b for maior que a, executa o "else" e exibe a mensagem

Exemplo 2: if... else if... else..

```
var
a, b: Integer;

begin
  a:= 2;
  b:= 3;

  if a > b then
  begin
```

variaveis a e b do tipo inteiro.
a recebe o valor 2 e b recebe o valor 3.
se a for maior que b, executa o "if" e exibe a mensagem
se b for maior que a, executa o "else if" e exibe a mensagem
se a e b for igual, executa o "else" e exibe a mensagem

Rua São Paulo, 1726, Centro
Edifício Marajó 1º andar, sala 14
Fernandópolis-SP / CEP: 15600-000
Tel: (017) 3463-2211

```
      ShowMessage('numero a é maior');
    end
    else if b > a then
    begin
      ShowMessage('numero b é maior');
    end
    else
    begin
      ShowMessage('são iguais');
    end;
end;
```

3.2.2. Estruturas de Caso

Esse tipo de Estrutura é utilizado em condições onde temos muitas possibilidades e essa possibilidade pode ser tratada de uma forma mais simples utilizando a Estrutura de Caso, ou mais conhecida em programação como **switch case**, abaixo um exemplo de utilização:

```
var
a: Integer;

begin
  a:= 6;

  case a of
     0: begin
       ShowMessage('Caso 0');
     end;

     1: begin
       ShowMessage('Caso 1');
     end;

     2: begin
       ShowMessage('Caso 2');
     end;

     else
     begin
       ShowMessage('Caso padrao');
     end;
  end;
end;
```

Rua São Paulo, 1726, Centro
Edifício Marajó 1º andar, sala 14
Fernandópolis-SP / CEP: 15600-000
Tel: (017) 3463-2211

3.2.3. Estruturas de Repetições

Estrutura muito utilizada, pois ela possibilita a repetição de um comando ou uma coleção de comandos de forma **finita** ou **baseado em uma condição**, abaixo estão listadas as estruturas atuais:

3.2.3.1. Laço de Repetição Para (For)

Consiste em uma estrutura, onde se constrói um laço **finito** onde será expresso um início e se possui um fim previsto, a estrutura principal desse tipo de laço baseia-se em:

for(valor inicial; comparação finita; modificação de valor){
comando a ser executado
}

Utilização no código ("to" - Crescente):

```
var
  i: Integer;

begin
  for i:= 0 to 10 do
  begin
    Showmessage('O valor do I: ' + IntToStr(i));
  end;
end;
```

Utilização no código ("downto" - Decrescente):

```
var
  i: Integer;

begin
  for i:= 10 downto 0 do
  begin
    Showmessage('O valor do I: ' + IntToStr(i));
  end;
end;
```

Rua São Paulo, 1726, Centro
Edifício Marajó 1º andar, sala 14
Fernandópolis-SP / CEP: 15600-000
Tel: (017) 3463-2211

3.2.3.2. Laço de Repetição Enquanto (While)

Consiste em uma repetição baseada em uma condição, sendo assim, **enquanto** essa condição for satisfeita, ou seja, enquanto ser verdadeira a condição o bloco de comando será repetido, a estrutura principal desse laço baseia-se em:

```
while (condição){
  Comando a ser executado
}
```

Utilização no código:

```
var
  n: Integer;

begin
  n:= 1;

  while n <= 10 do
  begin
    ShowMessage('numero: ' + IntToStr(n));
    n:= n + 1;
  end;
end;
```

Observações:

Nesse tipo de laço pode ser utilizado essas duas instruções para a implementação de novas funcionalidade.

Continue: Este comando serve para voltarmos ao início do Laço, mas cuidado quando for utilizar o mesmo, pois você pode acabar caindo em um Loop infinito, travando assim sua aplicação.

Break: Este comando serve para interrompermos um Loop, sairmos do mesmo.

Rua São Paulo, 1726, Centro
Edifício Marajó 1º andar, sala 14
Fernandópolis-SP / CEP: 15600-000
Tel: (017) 3463-2211

3.2.3.3. Laço de Repetição Faça Enquanto (Repeat .. Until)

Seu funcionamento e baseado no laço **while**, porém com uma individualidade, nesse laço **não** importa se a condição seja verdadeira pelo menos uma vez o comando dentro dele será executado, pois a verificação da condição e feita após a primeira execução do comando, sua estrutura principal baseia-se em:

```
repeat{
   comando a ser executado
}until(condição);
```

```
var
  n: Integer;

begin
  n:= 1;

  repeat
    ShowMessage('numero: ' + IntToStr(n));
    Inc(n);
  until (n >= 10);

end;
```