Fundamentos de SQL

Curso: Técnico em Redes de Computadores

Disciplina: Tecnologias e Linguagem de Banco de dados

Professor: Rodrigo da Rocha

SQL - Introdução

- SQL-Structured Query Language (Linguagem de Consulta Estruturada)
  - Apesar do QUERY, não é apenas de consulta (inclusão, alteração,...)
- É fundamentada na álgebra relacional, inclui comandos para:
  - Definição, Consulta e Atualização de dados
- Histórico:
  - Definição da1ª versão em 1974 – IBM – chamada SEQUEL
  - 1975 implementado o 1º protótipo
  - Revisada e ampliada entre 1976/77.
    - Teve seu nome alterado para SQL por razões Jurídicas
  - Publicada como padrão para SGBDR em 1986 pela ANSI (American National Standar Institute)
    - ANSI eqüivale a nossa ABNT
    - Mesmo padronizada, existem variações
  - Versões posteriores a de 86 → SQL2 e SQL3

SQL BÁSICO - Histórico

- Primeira versão em 1974, na IBM - SEQUEL
- Protótipo implementado em 1975
- Revisada e ampliada entre 1976 e 1977 - SQL
- Padrão oficial ANSI em 1986 – SQL1
- Revisão - SQL 2 – 1992
- Revisão - SQL 3 - 1999

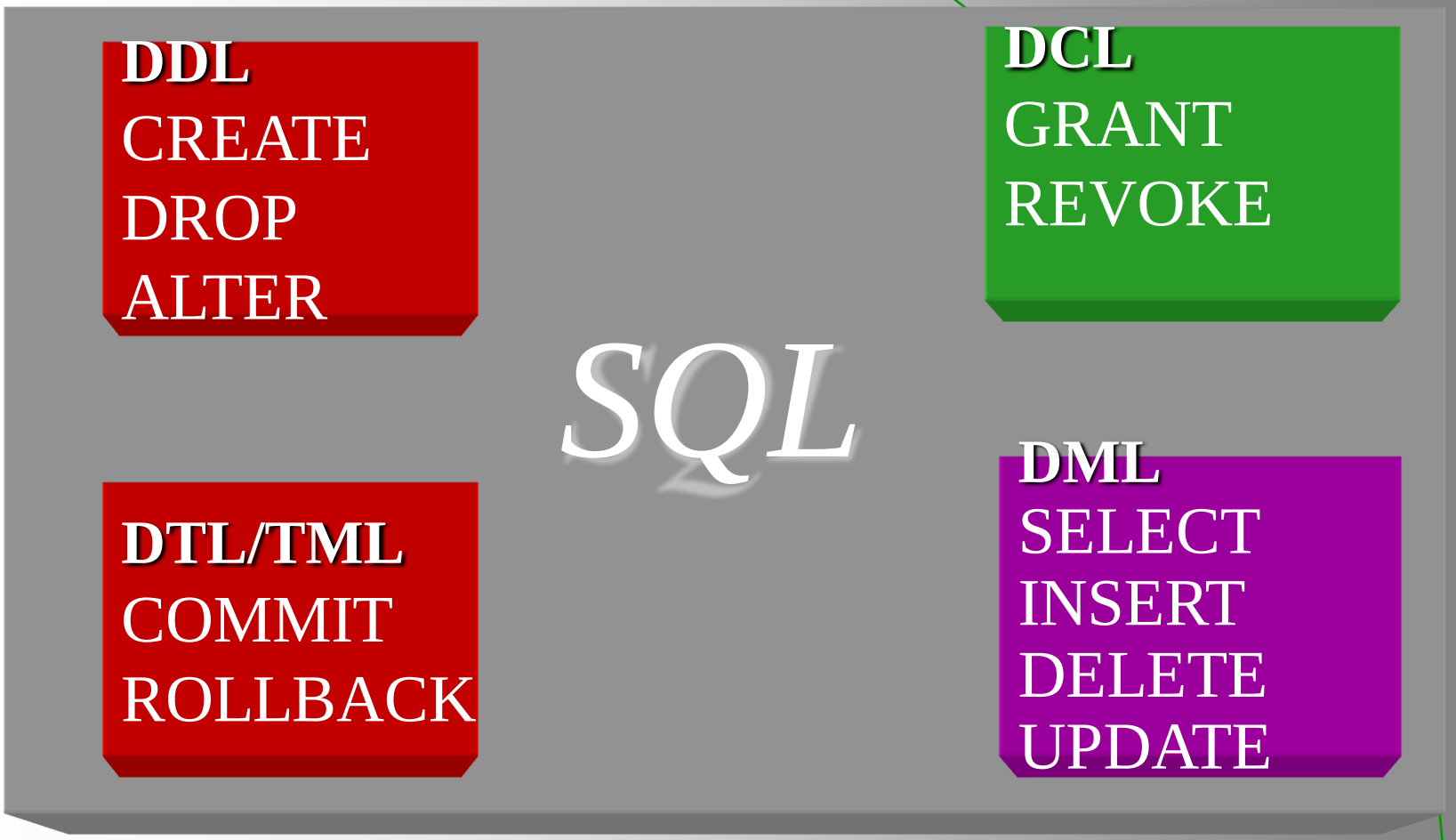
SQL - Propriedades

- Permitir consultas interativas (*query AdHoc*)
  - Usuários podem definir consultas poderosas sem a necessidade da criação de programas.
- Permite acesso e compartilhamento de dados em SGBDR
  - Pode ser embutida em programas de aplicação.
  - Pode ser usada para compartilhar dados Cliente/Servidor
- Possui comandos para administração do BD
  - O responsável pela administração do banco de dados (*BDA*) pode utilizar SQL para realizar suas tarefas.
- Maximiza a interoperabilidade entre SGBDR Heterogêneos
  - A padronização de SQL aumenta a portabilidade entre diferentes SGBDR.

SQL - Funções

- SQL provê suporte a várias funções de um SGBD :
  - **DDL** (linguagem de definição de dados)
    - Define as tabelas (virtuais ou não) onde os dados serão armazenados.
  - **DML** (linguagem de manipulação de dados)
    - Permite a inclusão, remoção, atualização e seleção dos dados;
  - **DCL** (linguagem de controle de dados)
    - Controla o acesso e os privilégios dos usuários, protegendo os dados de manipulações não autorizadas;
  - **DTL/TML** (linguagem de manipulação de transações)
    - Especifica as transações, garantindo o compartilhamento e a integridade dos dados.

SQL - Funções


DDL
CREATE
DROP
ALTER
DCL
GRANT
REVOKE
SQL
DTL/TML
COMMIT
ROLLBACK
DML
SELECT
INSERT
DELETE
UPDATE

SQL - Atenção !

- Cada implementação de SQL possui algumas adaptações para resolver certas particularidades, portanto, qualquer comando mostrado neste curso pode ser usado de forma diferente em um determinado SGBD.
  - Recomenda-se a leitura do manual do fabricante para maiores informações sobre o uso da linguagem SQL em SGBDs comerciais.
- O SQL usado neste curso será o baseado no Padrão ANSI e nenhuma característica específica de SGBD será abordada
- A maioria dos SGBDR baseiam-se no SQL ANSI

SQL BÁSICO

- SQL - Structured Query Language
- Fundamentada no modelo relacional
- Comandos para definição de dados, consulta e atualização

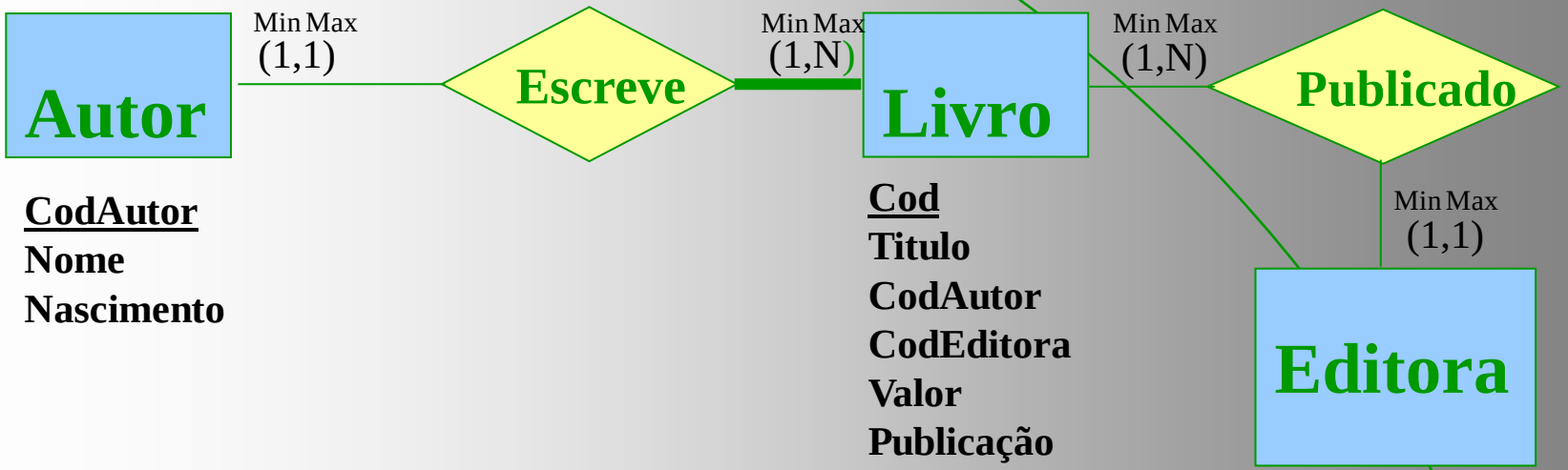
SQL BÁSICO - Vantagens

- Independência de fabricante
- Portabilidade entre sistemas
- Redução de custos com treinamento
- Consultas em Inglês
- Consulta Interativa
- Múltiplas visões de dados
- Definição dinâmica de dados

SQL BÁSICO - Desvantagens

- A padronização inibe a criatividade
- Não é ideal como linguagem relacional
- Falta ortogonalidade nas expressões

Modelo Conceitual



AUTOR(CodAutor, Nome, Nascimento)

LIVRO(Cod,Titulo,CodAutor,CodEditora,Valor,Publicacao,Volume,Idioma) CodAutor referencia AUTOR

CodEditora referencia Editora

EDITORA (CodEditora, Razao, Endereco, Cidade)

Tabelas correspondentes ao Modelo Conceitual Anterior

Tabela Autor

CodAutor	Nome	Nascimento
1	Rodrigo	17/04/1999
2	Danielle	20/07/1995
3	Claudia	30/01/2000

Tabela Editora

Cod	Razao	Cod Endereco
1	Cultura	2
2	Imperatriz	1

Tabela Endereço

Cod	Rua	Bairro	Cidade
1	Rua1	Centro	Recife
2	Rua2	Maranguape	Paulista

Tabela Livro

Cod	Titulo	Autor	Editora	Valor	Publicacao	Volume	Idioma
1	BD	1	1	200.00	10/10/2013	3	Inglês
2	Lógica	1	2	150.00	11/05/2010	3	Inglês
3	Redes	3	2	223.00	20/09/2013	2	Português

Requisitos do Sistema (SQL Server 2008)

Componentes	Requisitos mínimos
Sistemas Operacionais Compatíveis	Windows 7; Windows Server 2003; Windows Server 2008; Windows Server 2008 R2; Windows Vista
Processador	PC com processador Intel Pentium III 600 MHz ou equivalente mais rápido (1 GHz ou mais rápido é recomendável)
Memória	Mínimo de 512 MB de RAM (recomendável 1 GB ou mais) Mínimo de 256 MB de RAM (recomendável 1 GB ou mais)
Espaço em disco	675 MB de espaço livre no disco rígido

Tipos de dados numéricos Inteiros

Tipo de dados	Intervalo	Storage
bigint	-2^63 (-9,223,372,036,854,775,808) a 2^63-1 (9,223,372,036,854,775,807)	8 bytes
int	-2^31 (-2.147.483.648) a 2^31-1 (2.147.483.647)	4 bytes
smallint	-2^15 (-32,768) a 2^15-1 (32,767)	2 bytes
tinyint	0 a 255	1 byte

Tipos de Dados Números Aproximados

Tipo	Intervalo	Storage
float	- 1,79E+308 a -2,23E-308, 0 e 2,23E-308 a 1,79E+308	Depende do valor de n
real	- 3,40E + 38 a -1,18E - 38, 0 e 1,18E - 38 a 3,40E + 38	4 bytes

Tipos de Dados de Cadeias de Caracteres

Cadeias de caracteres	Descrição
char [(n)]	Dados de cadeia de caracteres não Unicode de comprimento fixo. n define o comprimento da cadeia de caracteres e deve ser um valor de 1 a 8.000.
varchar [(n | max)]	Dados de cadeia de caracteres não Unicode de comprimento variável. n define o comprimento da cadeia de caracteres e pode ser um valor de 1 a 8.000. max indica que o tamanho de armazenamento máximo é 2^31-1 bytes (2 GB).
ntext	Dados Unicode de comprimento variável com um comprimento máximo de cadeia de caracteres de 2^30 - 1 (1.073.741.823) bytes.
text	Dados não Unicode de comprimento variável na página de código do servidor e com um comprimento máximo de cadeia de caracteres de 2^31-1 (2.147.483.647).

DDL - Criando Tabelas

- **CREATE TABLE** - Cria uma nova tabela com seus campos e define as restrições de campo.

CREATE TABLE ***Tabela*** **(**
Coluna1 Tipo **[(*Tamanho*)] [NOT NULL] [DEFAULT] [...],**
[,*Coluna2 Tipo* [(T*amanho*)] [NOT NULL] [DEFAULT] [...],
[PRIMARY KEY (*Primária1*[, *Primária2* [, ...]])] [identity(1,1)**]**
[UNIQUE (*Candidata1*[, *Candidata2*[, ...]])]
[FOREIGN KEY (*Estrangeira1*[, *Estrangeira2* [, ...]]) REFERENCES
TabelaExterna **[(*ColunaExterna1* [, *ColunaExterna2* [, ...]])]**
[CHECK (*condição*)]
)

Onde : ( ) Indica parte da sintaxe do comando,
[ ] Indica opcionalidade do comando,

DDL – Criando Índices e Visões

- CREATE INDEX <Nome_Indice> ON <Nome_Tabela>.<(atributo)>

- CREATE VIEW <Nome_Tabela_Visão> AS SELECT * FROM AUTOR;

DDL - Criando Tabelas (Autor)

- Exemplo:

```
/* Cria tabela autor */
CREATE TABLE AUTOR(
        CodAutor INTEGER NOT NULL,
        Nome CHAR(30) NOT NULL,
        Nascimento DATE NOT NULL,
        PRIMARY KEY (CodAutor),
        UNIQUE (Nome,  Nascimento)
);
```

DDL - Criando Tabelas (Endereço)

```
/* Cria tabela Endereço*/
CREATE TABLE ENDERECO (
        CodEndereco INTEGER NOT NULL,
        RUA CHAR(30),
        BAIRRO CHAR(30),
        CIDADE CHAR(30),
        PRIMARY KEY(CodEndereco )
);
```

DDL - Criando Tabelas (Editora)

```
/* Cria tabela editora */
CREATE TABLE EDITORA (
    CodEditora INTEGER NOT NULL,
    Razao CHAR(30),
    Endereco INTEGER,
    PRIMARY KEY(CodEditora),
    FOREIGN KEY (Endereco) REFERENCES
    ENDERECO(CodEndereco)
);
```

DDL - Criando Tabelas (Livro)

- Exemplo: */* *Cria tabela livro* */*

```
CREATE TABLE LIVRO(
    Cod INTEGER,
    Titulo CHAR(30) NOT NULL,
    Autor INTEGER NOT NULL,
    Editora INTEGER NOT NULL,
    Valor float,
    Publicacao DATE,
    Volume INTEGER,
    Idioma CHAR(20) DEFAULT 'Português',
    PRIMARY KEY (Cod),
    FOREIGN KEY (Autor) REFERENCES AUTOR(CodAutor),
    FOREIGN KEY (Editora) REFERENCES EDITORA(CodEditora)
);
```

DDL - Alterando Tabelas

- **ALTER TABLE** - permite inserir/eliminar/modificar colunas nas tabelas já existentes

ALTER TABLE *Tabela*

{ADD (*NovaColuna NovoTipo* [BEFORE *Coluna*] [, ...]) |

DROP (coluna [, ...]) |

MODIFY (*Coluna NovoTipo* [NOT NULL] [, ...]) }

Onde : | Indica escolha de várias opções

{ } Indica obrigatoriedade de escolha de uma opção entre várias

Propriedades do ALTER

- **OBS:**
  - A cláusula DROP não remove atributos da chave primária
  - Não se usa NOT NULL juntamente com ADD, quando a tabela já contém registros (a nova coluna é carregada com NULL's)
  - Quando se altera o tipo de dados de uma coluna, os dados são convertidos para o novo tipo.
  - Se diminuir o tamanho de colunas do tipo CHAR, os dados são truncados

DDL - Alterando Tabelas

- **Exemplo:**

```
/* Adicionar o campo E-MAIL na tabela Autor */
ALTER TABLE AUTOR
ADD  EMAIL CHAR(30);

/* Modificar a quantidade de caracteres do campo E-MAIL na tabela Autor */
ALTER TABLE AUTOR
ALTER COLUMN EMAIL CHAR(25);

/* ELIMINAR o campo E-MAIL na tabela Autor */
ALTER TABLE AUTOR
DROP COLUMN EMAIL
```

SQL BÁSICO – DML - INSERT

- Incluindo linhas na tabela
  - INSERT INTO <tabela> [(<campos>)] VALUES (<valores>)

- Os dados são inseridos pela ordem especificada.
- Valores para campos CHAR, VARCHAR ou DATE são inseridos entre aspas simples.
- Se omitir a lista de colunas, serão selecionadas todas as colunas da tabela, pela sua ordem de criação

SQL BÁSICO – DML - INSERT

- Exemplos

```
/* Insere um registro na tabela autor */
INSERT INTO AUTOR(CodAutor, Nome, Nascimento ) VALUES
(1,'C.J.Date', '03/12/1941'), (2,'C.J.Date', '03/12/1941');
/* Insere um registro na tabela endereço */
INSERT INTO ENDERECO(codEndereco,rua,bairro,cidade)
values (4,'rua1,15', 'Centro', 'Recife');
/* Insere um registro na tabela editora */
INSERT INTO EDITORA(CodEditora, Razao, Endereco) VALUES
(1,'Cultura',1);
/* Insere um registro na tabela livro */
INSERT INTO LIVRO (Cod, titulo, autor,editora,valor,publicacao,volume)
VALUES(1,‘BD', 1, 1, 200.0, '10/10/2013', 3);
```

SQL BÁSICO – DML - SELECT

- Selecionar dados de uma tabela
  - SELECT * | <campos> FROM <tabela>
  - SELECT * | <campos> FROM tabela WHERE <condição>

- **SELECT/FROM** - Projeta os dados da(s) tabela(s), de acordo com os critérios especificados.
  - A projeção do resultado é em uma estrutura tipo tabela
  - Na cláusula SELECT, pode-se utilizar operadores aritméticos e funções de agregações, para projetar cálculos

Operadores e Funções de Agregação

Lógicos	
AND	E
OR	Ou
NOT	Não

Relacionais			
<> ou !=	Diferente	=	Igual a
>	Maior que	>=	Maior ou igual a
<	Menor que	<=	Menor ou igual a

Funções de Agregação	
AVG	Média
MIN	Mínimo
MAX	Máximo
COUNT	Contar
SUM	Somar

Oper. Aritméticos	
+	Adição
-	subtração
*	Multiplicação
/	Divisão

SQL BÁSICO – DML - SELECT

O * projeta todas as colunas de todas as tabelas especificadas na cláusula **FROM**

- **Exemplos:**

/ Projetar todas as informações dos autores */*
SELECT CodAutor, Nome, Nascimento FROM AUTOR ;
OU
SELECT * FROM AUTOR ;

/ Projetar a média dos valores dos livros */*
SELECT AVG (Valor) FROM LIVRO;

*/*Projetar todos os livros(títulos) e seus valores com 10% de desconto*/*
SELECT Titulo, Valor - (Valor * 0.1) FROM LIVRO;

/ Projetar a quantidade de autores cadastrados */*
SELECT COUNT (*) AS 'Quantidade de Autores' FROM AUTOR;

SQL BÁSICO – DML - SELECT

- Em SQL a eliminação de linhas duplicadas não é feita automaticamente, devendo a mesma ser especificada explicitamente.
  - ALL é o padrão quando não especificado DISTINCT

Exemplos:

/* Projetar todas as cidades das editoras repetidamente*/

SELECT ALL Cidade FROM EDITORA;

OU

SELECT Cidade FROM EDITORA;

/* Projetar todas as cidades das editoras sem duplicatas */

SELECT DISTINCT Cidade FROM EDITORA;

SQL BÁSICO - DML - SELECT

- Uma coluna pode ser especificada pelo nome da sua tabela (*Tabela.Coluna*), bem como, ser renomeada durante a consulta (*Coluna AS ColunaRenomeada*).

```
/* Projetar todos os nomes e respectivos nascimentos da tabela autor.
   NOTE:  mesmo especificando Tabela.Coluna, FROM é obrigatório */
SELECT AUTOR.Nome, AUTOR.Nascimento FROM AUTOR;

/* Projetar todos os títulos dos livros e seus valores em dobro */
SELECT Titulo, Valor * 2 as Dobro FROM LIVRO;
```

SQL BÁSICO – DML - WHERE (BETWEEN, IN)

- Selecionando campos de uma determinada condição
  - SELECT * FROM tabela WHERE <campo> BETWEEN <ValorInicial> AND <Valor Final>
  - SELECT * FROM <campo>WHERE <campo> IN <(conjunto de informações)>
- Exemplos

/* Projetar livros com valor de 10.00 a 100.00 */
SELECT *FROM LIVRO WHERE Valor BETWEEN 10.00 AND 100.00;

/* Projetar livros publicados após 30 de maio de 1993 */
SELECT * FROM LIVRO WHERE Publicacao > ‘5/30/93’;

/* Projetar as Editoras com sede em São Paulo ou Rio de Janeiro*/
SELECT * FROM EITORA WHERE Cidade IN (‘São Paulo’, ‘Rio de Janeiro’);

DML - Consultando Dados em Tabelas

–Coluna **[NOT] LIKE** ‘Cadeia de Caracteres’

- A condição é satisfeita quando o valor da coluna é igual ao valor da cadeia de caracteres.

-SELECT * FROM <nome>WHERE <campo> LIKE <string>

- Caracteres especiais para construção da cadeia de caracteres:

“%” ou “*” → Usado para representar zero ou mais caracteres.
“_” ou “?” → Usado para representar um caractere.

/*Projetar todos os autores cujo nome tenha 10 caracteres e inicie com R*/
SELECT *FROM AUTOR WHERE Titulo LIKE ‘R?????????’;

/*Projetar todos os livros que tenham Banco de Dados no seu título*/
SELECT * FROM LIVRO WHERE Titulo LIKE ‘%Banco de Dados%’;

DML - Consultando Dados em Tabelas

–**Coluna IS [NOT] NULL**

- A condição é satisfeita quando o valor da coluna for NULL

- SELECT * FROM <nome>WHERE <campo> IS | NOT NULL

/*Projetar todos os livros que estão sem preço definido */
SELECT *FROM LIVRO WHERE Valor IS NULL

Pode-se misturas os vários tipos de comparação

/*Projetar todos os livros que iniciam com R, estão com preço definido e foram publicados depois de 1/1/1995*/
SELECT * FROM LIVRO WHERE Titulo LIKE 'R%' and Valor IS NOT NULL and Publicacao > '1/1/1995';

DML - SQL BÁSICO

- Usando a cláusula WHERE com operadores de comparação

  =, <, >, <=, >=, <>

- Exemplos
  - SELECT * FROM Autor WHERE CodAutor <> 1;
  - SELECT * FROM Autor WHERE CodAutor = 1;
  - SELECT * FROM Editora WHERE cidade = Recife
  - UPDATE LIVRO SET valor = valor * 1.2 WHERE valor <= 20,0;
  - DELETE FROM Autor WHERE CodAutor = 3;

DML - SQL BÁSICO - AND, OR ou NOT

- Usando a cláusula WHERE com operadores lógicos
- Exemplos

- Mostre o código e o nome dos autores que nasceram entre ‘1/1/1970’ e ‘1/31/1980 ou não moram em Recife

  – SELECT codAutor, Nome FROM Autor WHERE (Nascimento >= ’01-1-1970’) **AND** (DataNascimento <= ‘01-31-1980’) **OR** (cidade <> ‘Recife’);

  – Mostre todos os autores que não moram em Olinda

  – SELECT * FROM Autor WHERE **NOT** (Cidade=‘Olinda’);

SQL BÁSICO - DML - DELETE

- Usando o comando DELETE para excluir linhas selecionadas de uma tabela
  - DELETE FROM <tabela> [WHERE <condição>]
- Exemplos

- */* Excluir os registros da tabela autor, onde CodAutor = 1*
  - **DELETE FROM AUTOR WHERE CodAutor = 1;**

  / Excluir todos os registros da tabela livro */*
  - **DELETE FROM LIVRO;** **OBS: Não é a mesma coisa que o DROP TABLE**
- Gera um registro de log para cada linha

SQL BÁSICO - DML - UPDATE

- Utilizando o comando UPDATE para atualizar dados existentes em todas as linhas
  - UPDATE <tabela> SET <campo>=<valor> [WHERE <condição>]
- Exemplos

/* Alterar o endereço e cidade da tabela editora com CodEditora = 1 */

UPDATE EDITORA SET Endereco = 'Av. N.S. de Fátima, 123', Cidade = 'São Paulo' WHERE CodEditora = 1;

/* Reajustar o valor de todos os livros em 10% */

UPDATE LIVRO SET Valor = Valor * 1.1

DML - SQL BÁSICO

- Usando a cláusula ORDER BY para classificar o resultado da consulta.
  - SELECT * FROM <tabela> [WHERE <condição>] ORDER BY <campos> [ASC | DESC]

- Exemplos
  - SELECT * From Cliente ORDER BY TipoCliente, DataNascimento

DML - SQL BÁSICO

- Usando a cláusula DISTINCT para eliminar linhas duplicadas no resultado da consulta

  – SELECT DISTINCT <campos> FROM <tabela>

- Exemplo: Mostre todas as cidades dos autores onde o nome das cidades são distintos/diferentes e ordene pelo nome da cidade

- SELECT DISTINCT Cidade FROM Autor ORDER BY Cidade

DML - SQL BÁSICO

- Modificar os nomes das colunas, substituindo-os por um alias.

OBS: (A modificação é feita apenas durante a consulta)

  – SELECT <campo> [AS] <alias> FROM <tabela>

- Exemplos
  – SELECT CodAutor **As** Código , Nome **As** Nome do Autor, Nascimento **As** Data de Nascimento FROM Ator

DML - SQL AVANÇADO

- Usando a cláusula WHERE selecionar dados em uma junção de tabelas

- SELECT C.Nome, TC.Descricao as “Tipo de Cliente”
  FROM Cliente C, TipoCliente TC
  WHERE TC.TipoCliente=C.TipoCliente

DML - SQL AVANÇADO

- Usando as palavras chave INNER JOIN para recuperar dados através da junção de duas tabelas

- SELECT C.Nome, TC.Descricao
  FROM Cliente C
  INNER JOIN TipoCliente TC
  ON (TC.TipoCliente=C.TipoCliente)

DML - SQL AVANÇADO

- Usando as palavras chave OUTER JOIN para também recuperar linhas que não satisfazem a condição de junção

- SELECT C.Nome, TC.Descricao
  FROM Cliente C
  LEFT OUTER JOIN TipoCliente TC
  ON (TC.TipoCliente=C.TipoCliente)

DML - SQL AVANÇADO

- Usando a palavra chave JOIN para recuperar dados através da junção de mais de duas tabelas

- SELECT T.CPF, C.Nome, TC.Descricao
  FROM (Titular T
  JOIN Cliente C
  ON (C.Cliente=T.Cliente))
  JOIN TipoCliente TC
  ON (TC.TipoCliente=C.TipoCliente)

DML - SQL AVANÇADO

- Fazendo um JOIN de uma tabela consigo própria.

- SELECT F.Nome as Funcionario,
  C.Nome as Chefe
  FROM Funcionario F
  LEFT JOIN Funcionario C
  ON (C.Matricula=F.MatriculaChefe)

DML - SQL AVANÇADO

- Usando a função COUNT.

- SELECT COUNT (*) FROM Autor
- SELECT COUNT(*) FROM Autor WHERE Nome LIKE Rodrigo%

DML - SQL AVANÇADO

- Usando a função SUM.

Exemplo: Mostre a soma dos valores de todos os livros que foram publicados entre ’01/11/2013 e 31/11/2013

- SELECT SUM(valor) FROM Livro WHERE Publicacao BETWEEN ‘01-11-2013’ And ‘31-11-2013’;

SQL AVANÇADO

- Usando a cláusula GROUP BY para agrupar e sumarizar linhas.

- SELECT L.Cliente, C.Nome,
  COUNT(L.NumeroCupom), SUM(L.ValorLocacao)
  FROM Locacao L
  LEFT JOIN Cliente C
  ON (C.Cliente=L.Cliente)
  GROUP BY L.Cliente, C.Nome

SQL AVANÇADO - TRIGGER

```
CREATE TRIGGER nome_da_trigger ON  TABLE | VIEW
        {FOR | AFTER | INSTEAD OF } { [ INSERT ] [ , ] [ UPDATE ]  }
AS
BEGIN
   AÇÃO  <Código para executa>
END
```

Exemplo: Quando inserir um novo livro, mostrar uma mensagem na tela que um novo livro foi inserido

```
Create Trigger Registro_Inserido ON  Livro
After INSERT AS
Begin
        PRINT 'Foi Inserido um novo Livro!'
        --Aqui também podemos utilizar os comandos DML por exemplo
End
```