Instituto Federal

Campus Goiânia

Bacharelado em Sistemas de Informação

Banco de Dados II

Prof. Dory Gonzaga Rodrigues

MySQL®

ORACLE®

PostgreSQL
the world's most advanced open source database

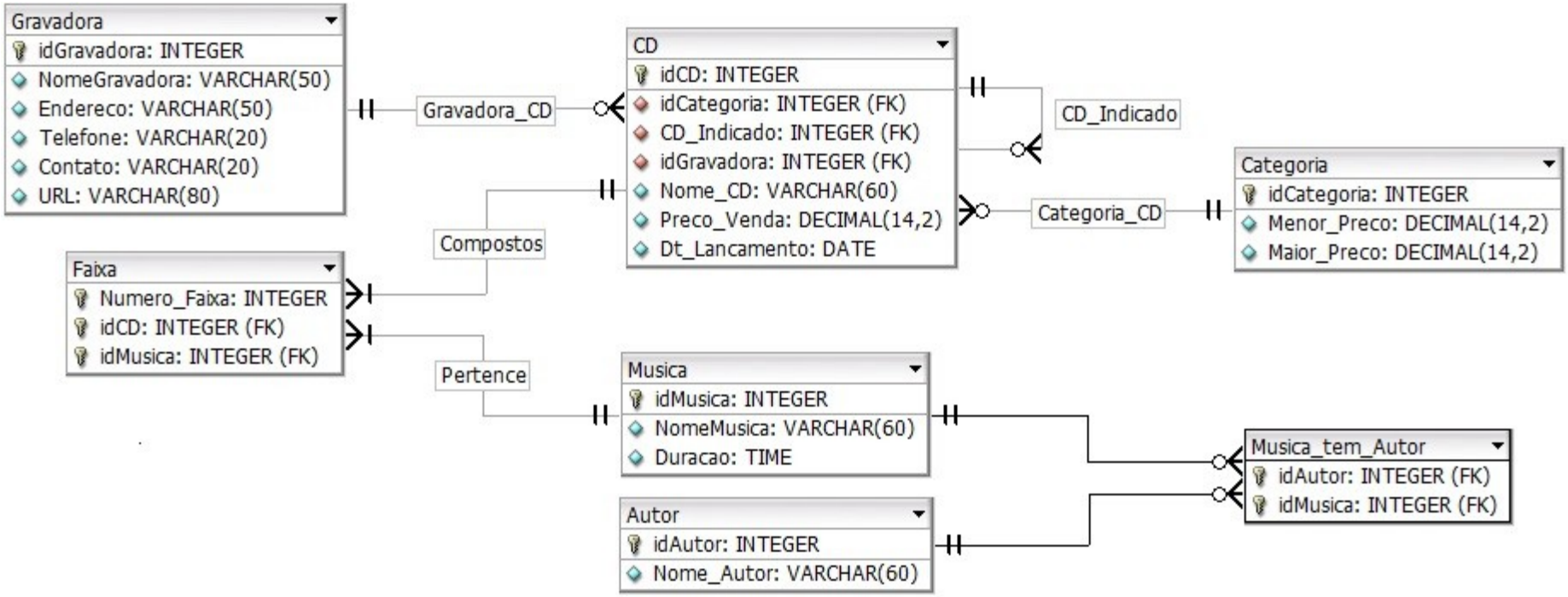
Agenda

- Pesquisas Avançadas
  - UNION
  - UNION ALL
  - EXCEPT / MINUS
  - INTERSECT
  - DIVISION
  - CASE

DML BANCO DE DADOS

BANCO DE DADOS REFERÊNCIA

- Utilizaremos nesta aula o banco de dados: Catálogo de CDs



Arquivo enviado por e-mail: DDL_Catalogo_CDs.sql
DML_Catalogo_CDs.sql

DML | BANCO DE DADOS

BANCO DE DADOS REFERÊNCIA

- Utilizaremos nesta aula o banco de dados: Álgebra Relacional

```
CREATE DATABASE algebra_relacional;

USE algebra_relacional;

CREATE TABLE a (
        x INT,
        y VARCHAR(5)
);

INSERT INTO a(x,y) VALUES(1, 'A');
INSERT INTO a(x,y) VALUES(2, 'B');
INSERT INTO a(x,y) VALUES(3, 'C');
INSERT INTO a(x,y) VALUES(4, 'D');
```

```
CREATE TABLE b (
        x INT,
        y VARCHAR(5)
    );

INSERT INTO b(x,y) VALUES(1, 'A');
    INSERT INTO b(x,y) VALUES(3, 'C');
```

DML BANCO DE DADOS

BANCO DE DADOS REFERÊNCIA

- Utilizaremos nesta aula o banco de dados: Álgebra Relacional

```
CREATE TABLE c (
        x INT,
        y VARCHAR(5)
);

INSERT INTO c(x,y) VALUES(1,'A');
INSERT INTO c(x,y) VALUES(2,'A');
INSERT INTO c(x,y) VALUES(3,'A');
INSERT INTO c(x,y) VALUES(1,'B');
INSERT INTO c(x,y) VALUES(2,'B');
INSERT INTO c(x,y) VALUES(3,'C');
INSERT INTO c(x,y) VALUES(3,'D');
```

```
CREATE TABLE d (
        x INT
);

INSERT INTO d(x) VALUES(1);
INSERT INTO d(x) VALUES(2);
INSERT INTO d(x) VALUES(3);
```

DML UNION

SQL – DML - AVANÇADA

- Operação: UNION

A operação **UNIÃO** cria como resultado a união de todas as linhas de uma tabela com todas as linhas da outra tabela. Diferente do JOIN, onde as linhas são combinadas, na união as linhas da segunda tabela são colocadas em sequencia às linhas da primeira tabela.

Atenção:

- Para que seja possível a união entre as tabelas, é necessário que haja compatibilidade entre as colunas projetadas na consulta das duas tabelas.

- Na prática: as colunas devem ser do mesmo tipo e estar na mesma sequencia.

DML UNION

SQL – DML - AVANÇADA

- Operação: UNION

A sintaxe:

```
SELECT    colunas        FROM tabela(s)
[ WHERE <condição> ]
[ GROUP BY <campo> [ HAVING      <condição> ] ]
```

UNION

```
SELECT    colunas        FROM tabela(s)
[ WHERE <condição> ]
[ GROUP BY <campo> [ HAVING      <condição> ] ]
```

DML UNION

SQL – DML - AVANÇADA

- Operação: UNION

EXEMPLO: Apresente o resultado da união entre as tabelas A e B;

```
SELECT x, y
FROM   a

UNION

SELECT x,y
FROM   b;
```

Tabela a

x	y
1	A
2	B
3	C
4	D

Tabela b

x	y
1	A
3	C

Resultado a U b

x	y
1	A
2	B
3	C
4	D

DML UNION

SQL – DML - AVANÇADA

- Operação: UNION ALL

A operação **UNIÃO ALL** realiza a união entre as duas tabelas sem comparar o conteúdo, ou seja, o comando permite que linhas repetidas seja apresentadas no resultado da consulta.

Atenção:

- Normalmente utilizamos o UNION ALL apenas quando se trabalha com tabelas distintas, em que os dados não se repetirão.

DML UNION

SQL – DML - AVANÇADA

- Operação: UNION ALL

EXEMPLO: Apresente o resultado da <u>união total</u> entre as tabelas A e B;

```
SELECT x, y
FROM   a

UNION ALL

SELECT x,y
FROM   b;
```

Tabela a

x	y
1	A
2	B
3	C
4	D

Tabela b

x	y
1	A
3	C

Resultado a U b (Total)

x	y
1	A
2	B
3	C
4	D
1	A
3	C

DML UNION

SQL – DML - AVANÇADA

- Operação: UNION

Quando utilizamos a cláusula **UNIÃO ?**

- Normalmente utilizamos o UNION em substituição a cláusula OR no predicado (WHERE) do SELECT.

DML UNION

SQL – DML - AVANÇADA

- Operação: UNION

EXEMPLO: Selecione o nome do CD, nome da Música com código do CD igual a 1 ou código da Música igual a 20

DML UNION

SQL – DML - AVANÇADA

- Operação: UNION

EXEMPLO: Selecione o nome do CD, nome da Música com código do CD igual a 1 ou código da Música igual a 20

```
SELECT nome_CD, nomeMusica
FROM CD, FAIXA f, MUSICA m
WHERE cd.idCD = f.idCD
AND f.idMusica = m.idMusica
AND cd.idCD = 1

UNION

SELECT nome_CD, nomeMusica
FROM CD, FAIXA f , MUSICA m
WHERE cd.idCD = f.idCD
AND f.idMusica = m.idMusica
AND m.idMusica = 20
```

```
SELECT nome_CD, nomeMusica
FROM CD, FAIXA f, MUSICA m
WHERE cd.idCD = f.idCD
AND f.idMusica = m.idMusica
AND (
        cd.idCD = 1 OR m.idMusica = 20
    )
```

DML UNION EXCEPT

SQL – DML - AVANÇADA

- Operação: EXCEPT

A cláusula EXCEPT ou MINUS (em alguns bancos de dados) deve ser utilizada quando desejamos as linhas que existem em um SELECT e não existem em outro. Ou seja, é a operação de diferença da álgebra relacional.

DML UNION EXCEPT

SQL – DML - AVANÇADA

- Operação: EXCEPT

A sintaxe:

```
SELECT   colunas        FROM tabela(s)
[ WHERE <condição> ]
[ GROUP BY <campo> [ HAVING      <condição> ] ]
```

EXCEPT

```
SELECT   colunas        FROM tabela(s)
[ WHERE <condição> ]
[ GROUP BY <campo> [ HAVING      <condição> ] ]
```

DML UNION EXCEPT

SQL – DML - AVANÇADA

- Operação: EXCEPT

EXEMPLO: Apresente o resultado da diferença entre as tabelas A e B;

```
SELECT x, y
FROM   a

EXCEPT

SELECT x,y
FROM   b;
```

DML UNION EXCEPT

SQL – DML - AVANÇADA

- Operação: EXCEPT

EXEMPLO: Apresente o resultado da diferença entre as tabelas A e B;

```
1)  SELECT *  FROM a
    WHERE (x,y) NOT IN ( SELECT x,y FROM b );

2)  SELECT * FROM a
    WHERE NOT EXISTS ( SELECT x,y
                         FROM b
                        WHERE b.x = a.x
                          AND b.y = a.y
                      );

3)  SELECT   x, y
    FROM     a LEFT JOIN b USING (x,y)
    WHERE    b.x IS NULL;
```

Tabela a

x	y
1	A
2	B
3	C
4	D

Tabela b

x	y
1	A
3	C

Resultado A - B

x	y
2	B
4	D

DML UNION EXCEPT

SQL – DML - AVANÇADA

- Operação: EXCEPT

EXEMPLO: Quais gravadoras não tem CD's cadastrados ?

DML UNION EXCEPT

SQL – DML - AVANÇADA

- Operação: EXCEPT

EXEMPLO: Quais gravadoras não tem CD's cadastrados ?

```
SELECT    idgravadora
FROM      gravadora

EXCEPT

SELECT    idgravadora
FROM      cd;
```

DML UNION EXCEPT

SQL – DML - AVANÇADA

- Operação: EXCEPT

EXEMPLO: Quais gravadoras não tem CD's cadastrados ?

```
1)  SELECT idGravadora
    FROM   gravadora
    WHERE  idGravadora NOT IN (
                              SELECT idGravadora FROM cd
                          );

2)  SELECT g.idGravadora
    FROM     gravadora g
    WHERE  NOT EXISTS (       SELECT   cd.idgravadora
                              FROM     cd
                              WHERE    cd.idgravadora = g.idGravadora
                          );
```

DML UNION EXCEPT

SQL – DML - AVANÇADA

- Operação: EXCEPT

EXEMPLO: quais gravadoras não tem CD's cadastrados ?

SQL mais eficiente

```
3)  SELECT g.idgravadora
    FROM   gravadora g LEFT JOIN cd USING (idgravadora)
    WHERE  cd.idgravadora IS NULL;
```

```
4)  SELECT g.idgravadora
    FROM   gravadora g LEFT JOIN cd ON cd.idgravadora = g.idgravadora
    WHERE  cd.idgravadora IS NULL;
```

SQL – DML - AVANÇADA

- Operação: INTERSECT

A cláusula INTERSECT deve ser utilizada quando desejamos as linhas que existem em ambas as tabelas. Ou seja, é a operação de intersecção da álgebra relacional.

DML UNION EXCEPT INTERSECT

SQL – DML - AVANÇADA

- Operação: INTERSECT

A sintaxe:

```
SELECT   colunas        FROM tabela(s)
[ WHERE <condição> ]
[ GROUP BY <campo> [ HAVING      <condição> ] ]
```

INTERSECT

```
SELECT   colunas        FROM tabela(s)
[ WHERE <condição> ]
[ GROUP BY <campo> [ HAVING      <condição> ] ]
```

DML | UNION | EXCEPT | INTERSECT

SQL – DML - AVANÇADA

- Operação: INTERSECT

EXEMPLO: Apresente o resultado da intersecção entre as tabelas A e B;

```
1)  SELECT *  FROM a
    WHERE (x,y) IN ( SELECT x,y FROM b );

2)  SELECT * FROM a
    WHERE EXISTS ( SELECT x,y
                     FROM b
                    WHERE b.x = a.x
                      AND b.y = a.y
                 );

3)  SELECT   x, y
    FROM     a INNER JOIN b USING (x,y) ;
```

Tabela a

x	y
1	A
2	B
3	C
4	D

Tabela b

x	y
1	A
3	C

Resultado A ∩ B

x	y
1	A
3	C

DML UNION EXCEPT INTERSECT

SQL – DML - AVANÇADA

- Operação: INTERSECT

EXEMPLO**:** Quais gravadoras tem CD's cadastrados ?

DML UNION EXCEPT INTERSECT

SQL – DML - AVANÇADA

- Operação: INTERSECT

EXEMPLO: Quais gravadoras tem CD's cadastrados ?

```
SELECT idGravadora
FROM   gravadora

INTERSECT

SELECT idGravadora
FROM   cd
```

DML UNION EXCEPT INTERSECT

SQL – DML - AVANÇADA

- Operação: INTERSECT

EXEMPLO: Quais gravadoras tem CD's cadastrados ?

```
1)  SELECT idGravadora
    FROM   gravadora
    WHERE  idGravadora IN (
                              SELECT idGravadora FROM cd
                           );

2)  SELECT   g.idGravadora
    FROM     gravadora g
    WHERE  EXISTS ( SELECT   cd.idgravadora
                    FROM     cd
                    WHERE  cd.idgravadora = g.idGravadora
                  );
```

DML | UNION | EXCEPT | INTERSECT

SQL – DML - AVANÇADA

- Operação: INTERSECT

EXEMPLO: Quais gravadoras tem CD's cadastrados ?

SQL mais eficiente

```
3)  SELECT  DISTINCT g.idgravadora
    FROM    gravadora g INNER JOIN cd USING (idgravadora);

4)  SELECT  DISTINCT g.idgravadora
    FROM    gravadora g INNER JOIN cd ON cd.idgravadora = g.idgravadora;
```

SQL – DML - AVANÇADA

- Operação: DIVISION

A operação **DIVISION** não possui um operador específico no comando SQL. A operação de Divisão realiza a extração de dados de uma tabela que estão associados a todos as elementos (linhas) de uma outra tabela.

SQL – DML - AVANÇADA

- Operação: DIVISION

EXEMPLO: Quais gravadoras tem CD's cadastrados em TODAS as categorias de preço ?

```
SELECT DISTINCT g.NomeGravadora
FROM gravadora g
WHERE NOT EXISTS (  SELECT *
                    FROM   cd
                    WHERE  cd.idcategoria NOT IN ( SELECT idcategoria
                                                   FROM cd
                                                   WHERE cd.idgravadora = g.idgravadora
                                                   )
                  );
```

SQL – DML - AVANÇADA

- Operação: CASE

A cláusula **CASE** é utilizada em conjunto com o comando SELECT ou UPDATE quando temos diversas condições para a extração ou alteração dos dados.

SQL – DML - AVANÇADA

- Cláusula: CASE

A sintaxe:

```
SELECT   colunas, CASE
                        WHEN  condição THEN ação
                        ...
                        ELSE ação_padrão
                  END nome_da_coluna
FROM     tabela(s);
```

SQL – DML - AVANÇADA

- Cláusula: CASE

EXEMPLO: apresente o nome do CD, seu preço de venda e o valor mínimo de venda de acordo com os descontos progressivos da tabela abaixo?

Preço_Venda	Desconto
< 10	10%
>=10 e <13	20%
>=13	30%

SQL – DML - AVANÇADA

- Cláusula: CASE

EXEMPLO: apresente o nome do CD, seu preço de venda e o valor mínimo de venda de acordo com os descontos progressivos da tabela abaixo?

Preço_Venda	Desconto
< 10	10%
>=10 e <13	20%
>=13	30%

```
SELECT    nome_CD, preco_venda,
      CASE
            WHEN  preco_venda < 10                                  THEN preco_venda * 0.9
            WHEN  preco_venda >= 10 AND preco_venda < 13 THEN preco_venda * 0.8
            ELSE     preco_venda * 0.7
      END valor_mínimo
FROM     CD;
```

SQL – DML - AVANÇADA

- Cláusula: CASE no comando UPDATE

EXEMPLO: altere o preço da venda de acordo com os descontos progressivos da tabela abaixo?

Preço_Venda	Desconto
< 10	10%
>=10 e <13	20%
>=13	30%

SQL – DML - AVANÇADA

- Cláusula: CASE no comando UPDATE

EXEMPLO: altere o preço da venda de acordo com os descontos progressivos da tabela abaixo?

Preço_Venda	Desconto
< 10	10%
>=10 e <13	20%
>=13	30%

```
UPDATE  CD
SET preco_venda =
      CASE
            WHEN  preco_venda < 10                                  THEN preco_venda * 0.9
            WHEN  preco_venda >= 10 AND preco_venda < 13 THEN preco_venda * 0.8
            ELSE     preco_venda * 0.7
      END;
```