INSTITUTO FEDERAL
ESPÍRITO SANTO

BANCOS DE DADOS

BD-I
SQL (Structured Query Language)
Prof. Moisés Omena

INSTITUTO FEDERAL
ESPÍRITO SANTO

BANCO DE DADOS

Implementando modelo Físico usando a linguagem SQL

Prof. Moisés Omena

SQL (Structured Query Language)

- Linguagem Estruturada de Consulta
- Formada pelo conjunto das linguagens:
  - DDL (Data Definition Language)
  - DML (Data Manipulation Language)
  - DQL (Data Query Language):
  - DCL (Data Control Language):
  - DTL (Data Transaction Language):

Estrutura da linguagem SQL

SQL (Structured Query Language)

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

DDL (Data Definition Language)

- Linguagem de Definição de Dados
  - CREATE: Cria uma estrutura
  - ALTER: Altera uma estrutura
  - DROP: Exclui uma estrutura

DML (Data Manipulation Language)

- Linguagem de Manipulação de Dados
  - INSERT: Insere dados
  - UPDATE: Altera dados
  - DELETE: Exclui dados

DQL (Data Query Language)

- Linguagem de Consulta de Dados
  - SELECT: Retorna dados
  - Ordenação de dados
  - Agrupamento de dados
  - Funções aritméticas
  - Filtros de seleção

DCL (Data Control Language)

- Linguagem de Controle de Dados
  - GRANT: Habilita acesso a dados e operações
  - REVOKE: Revoga acesso a dados e operações

DTL (Data Transaction Language)

- Linguagem de Transação de Dados
  - START TRANSACTION: Inicia a transação
  - COMMIT: Concretiza a transação
  - ROLLBACK: Anula a transação

INSTITUTO FEDERAL
ESPÍRITO SANTO

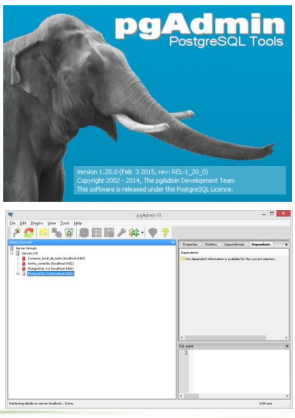
BANCO DE DADOS

Criando nosso DATABASE

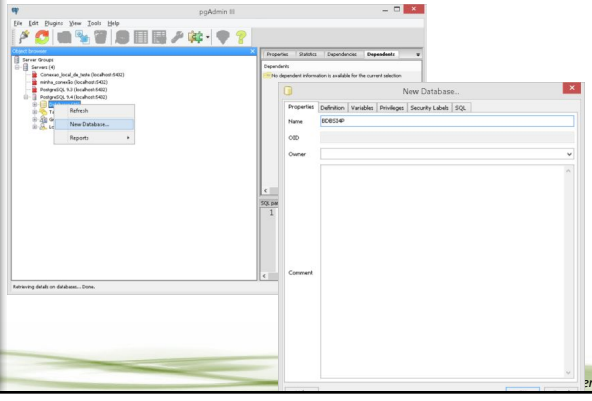
Abra o PgAdmin

PostgreSQL
pgAdmin3

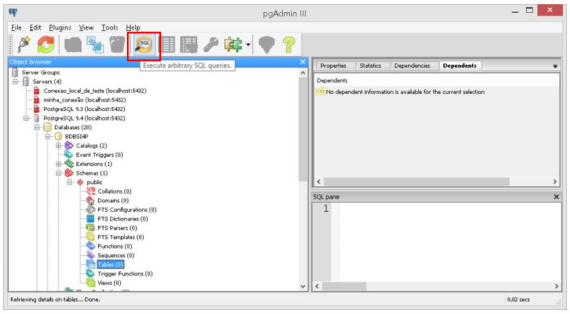
- Faça a conexão local com o banco de dados PostgreSQL



Criando banco de dados (BDBSI4P)



Enviando instruções ao banco de dados desejado



DDL (Data Definition Language)

- CREATE TABLE: Cria uma tabela

CREATE TABLE <table_name>
(<column_name> <datatype> {null | Not null} [,...]);

Definições:
table_name → refere-se ao nome que será dado a tabela
column_name → nome para cada coluna a ser criada para a tabela
datatype → refere-se ao tipo do dado (numeric, char, date) que será atribuído a coluna

Tipos de Campos (<datatype>)

- TINYINT:
  - é um número inteiro com ou sem signal. Com sinal a margem de valores válida é de -128 até 127. Sem sinal, a margem de valores é de 0 até 255
- BIT OU BOOL:
  - um número inteiro que pode ser 0 ou 1.
- INT
  - Um inteiro de tamanho normal. A faixa com sinal é de -2147483648 a 2147483647. A faixa sem sinal é de 0 a 4294967295.
- BIGINT
  - Um inteiro grande. A faixa com sinal é de -9223372036854775808 a 9223372036854775807. A faixa sem sinal é de 0 a 18446744073709551615.
- DOUBLE PRECISION
  - Um número de ponto flutuante de tamanho normal (dupla-precisão). Valores entre -1.7976931348623157E+308 até -2.2250738585072014E-308.
- DECIMAL
  - Um número de ponto flutuante não empacotado. Se comporta como um campo CHAR: ``não empacotado'' significa que o número é armazenado como uma string, usando um caracter para cada digito do valor.

Tipos de dados ou <datatype>

- Tipos de dados são formatos de armazenamentos dos dados/informações em colunas de uma tabela.
  - Integer:

    De -2147483648 até 2147483647
  - BigInt

    Um inteiro grande. A faixa com sinal é de -9223372036854775808 a 9223372036854775807. A faixa sem sinal é de 0 a 18446744073709551615.
  - Numeric

    números com valores decimais de maneira precisa: numeric (precisão,escala) ou seja numeric (número de dígitos, dígitos após ponto).

Tipos de dados

- Tipos de dados são formatos de armazenamentos dos dados/informações em colunas de uma tabela.
  - Float

    flutuante de precisão simples. Os valores válidos vão desde -175494351E-38 até 3.402823466E+38
  - Serial

    Tipo numérico ,inteiro, e comumente utilizado para gerar valores sequenciais automaticamente, estando normalmente relacionado as colunas chave primária com auto-incremento pela sua própria característica.

- DATE
  - Uma data. A faixa suportada é entre '1000-01-01' e '9999-12-31'. MySQL mostra valores DATE no formato 'AAAA-MM-DD', mas permite a você a atribuir valores a campos DATE utilizando tanto strings quanto números
- DATETIME
  - Um combinação de hora e data. A faixa suportada é entre '1000-01-01 00:00:00' e '9999-12-31 23:59:59'. MySQL mostra valores DATETIME no formato 'AAAA-MM-DD HH:MM:SS', mas permite a você que atribuir valores a campos DATETIME utilizado strings ou números
- TIMESTAMP[(M)]
  - Um timestamp. A faixa é entre '1970-01-01 00:00:00' e algum momento no ano 2037.

Blob (Binary Large Object) e Text

- TINYBLOB, TINYTEXT
  - Um campo BLOB ou TEXT com tamanho máximo de 255 (2^8 - 1) caracteres
- BLOB, TEXT
  - Um campo BLOB ou TEXT com tamanho máximo de 65535 (2^16 - 1) caracteres
- MEDIUMBLOB, MEDIUMTEXT
  - Um campo BLOB ou TEXT com tamanho máximo de 16777215 (2^24 - 1) caracteres
- LONGBLOB, LONGTEXT
  - Um campo BLOB ou TEXT com tamanho máximo de 4294967295 ou 4G (2^32 - 1) caracteres

Tipos de dados

- Char

  Possibilita guardar um conjunto de caracteres definidos pelo usuário. Por exemplo char(10), guardaria 10 caracteres. Entretanto se você não utilizar os 10 caracteres eles serão preenchidos com espaços em branco, ocorrendo assim desperdício de espaço.

- Varchar

  Ao contrário do formato Char, varchar lida de modo dinâmico com a quantidade de caracteres, podendo-se especificar um determinado número, (ex: varchar(150)), mas que ao serem inseridos 20 caracteres nenhum preenchimento automático de caracteres espaço ocorrerá.Por tal característica o tipo varchar é amplamente utilizado.

Os Tipos CHAR e VARCHAR

- Os tipos CHAR e VARCHAR são parecidos, mas diferem no modo como são armazenados e recuperados.
- CHAR é fixado pelo tamanho declarado na criação da tabela.
  - Qualquer valor entre 1 e 255 caracteres
  - Utiliza todo espaço declarado, preenchendo automaticamente os que ficaram em branco.
- VARCHAR são strings de tamanho variável.
  - qualquer tamanho entre 1 e 255, assim como para campo CHAR.
  - No entanto, diferente de CHAR, valores VARCHAR são armazendos usando apenas quantos caracteres forem necessários, mais 1 byte para gravar o tamanho.
  - Se você atribuir um valor para uma coluna CHAR ou VARCHAR que exceda o tamanho máximo da coluna, o valor é truncado para este tamanho.

Tipos de dados

- Money

  Destinado a receber valores monetários. Possui duas casas decimais e aceita valores grandes no intervalo de -92.233.720.386.547.758,08 até 92.233.720.386.547.758,07.

- Text

  Não implementa nenhum limite para inserção de caracteres. Utilizado para campos onde serão inclusos textos longos.

Tipos de dados

- Date

  Representação de datas. De 4713 A.C até 5.874.897 D.C.

- Time

  Formato para horas do dia.

- Timestamp

  Formato que inclui tanto data quanto horas

- Boolean

  Representa valores verdadeiros ou falsos (também permite utilização de NULL)

DDL (Data Definition Language)

- CREATE TABLE: Cria uma tabela

CREATE TABLE <table_name>
(<column_name> <datatype> {null | Not null} [,...]);

Execução de código:

```
CREATE TABLE ALUNO (
    CODIGO INT NOT NULL,
    NOME VARCHAR(45),
    DATA_NASCIMENTO DATE);
```

DML (Data Manipulation Language)

- SELECT: Selecionar dados

SELECT * FROM <table_name>

Definições:
table_name → refere-se ao nome que será dado a tabela
* → representa todas as colunas que a tabela possui

OBS: A instrução SELECT será melhor detalhada mais a frente, o objetivo de apresentá-la aqui é o de possibilitar analisar se os campos determinados foram criados.

Exercícios

- Crie a estrutura da tabela abaixo (testaremos os dados mais a frente quando tratarmos inserção)

```
CREATE TABLE  tipos_armazenamento (
      campo0 INTEGER,
      campo1 INT,
      campo2 BIGINT,
      campo3 NUMERIC(9,3),
      campo4 SERIAL,
      campo5 MONEY,
      campo6 CHAR(5),
      campo7 VARCHAR(50),
      campo8 TEXT,
      campo9 FLOAT,
      campo10 DATE,
      campo11 TIME,
      campo12 TIMESTAMP,
      campo13 BOOLEAN
);
```


Observações

- Apesar de existirem vários tipos de dados os tipos mais largamente utilizados são:
  - INTEGER
  - FLOAT
  - CHAR( )
  - VARCHAR( )
  - DATE
  - TIMESTAMP
  - SERIAL

DDL (Data Definition Language)

- ALTER TABLE: Alterando uma tabela

ALTER TABLE <table_name>
ADD (<column_name> <datatype> {null | Not null} [,...]);

definições:
table_name → refere-se ao nome da tabela que sofrerá a alteração
ADD COLUMN → refere-se a modificação que será adicionada a tabela
DROP COLUMN → Refere-se a modificação de exclusão da tabela
RENAME COLUMN → Refere-se ao tipo de modificação que será executada na tabela
ALTER COLUMN → Refere-se ao tipo de modificação que será executada na tabela

DDL (Data Definition Language)

- ALTER TABLE: Alterando uma tabela

ALTER TABLE <table_name>
ADD (<column_name> <datatype> {null | Not null} [,...]);

Execução de código:

```
ALTER TABLE aluno ADD nome VARCHAR(50);
ALTER TABLE aluno DROP  nome;
ALTER TABLE aluno RENAME COLUMN nome TO nome_completo;
ALTER TABLE aluno ALTER COLUMN nome_completo TYPE varchar(80);
ALTER TABLE aluno ALTER COLUMN codigo SET NOT NULL;
ALTER TABLE aluno ALTER COLUMN codigo DROP NOT NULL;
```


DDL (Data Definition Language)

- DROP TABLE: Apaga uma tabela

DROP TABLE [IF EXISTS] <table_name>

definições:
table_name → refere-se ao nome da tabela que sofrerá a alteração
IF EXISTS → verifica se a tabela existe para não causar um erro de execução.

Execução de código:

DROP TABLE ALUNO
OU
DROP TABLE IF EXISTS ALUNO

INSTITUTO FEDERAL
ESPÍRITO SANTO

MANIPULANDO DADOS
INSERÇÃO

DML (Data Manipulation Language)

- Linguagem de Manipulação de Dados
  - INSERT: Insere dados
  - UPDATE: Altera dados
  - DELETE: Exclui dados

DML (Data Manipulation Language)

- INSERT INTO: **Inserção de** dados (ação linha a linha)

```
INSERT [INTO] <table_name>
    [{(column_list)}]
{VALUES (expression [, expression ] ...)};
```

Definições:
table_name → nome da tabela que receberá os dados
column_list → lista das colunas daquela tabela que receberão dados. Se omitido este elementos, deverão ser passados dados para todas as colunas (mesmo que o valor NULL.
expression → tipos de dados que podem ser atribuidos as colunas.

DML (Data Manipulation Language)

- INSERT INTO: **Inserção de** dados (ação linha a linha)

```
INSERT [INTO] <table_name>
    [{(column_list)}]
{VALUES (expression [, expression ] ...)};
```

Execução de código:

```
INSERT INTO PROJETO (NUMERO, NOME, LOCALIZACAO)
VALUES(50,'PROJETO_50','SÃO PAULO');
OU
INSERT INTO PROJETO (LOCALIZACAO,NOME,NUMERO)
VALUES('VITORIA','PROJETO_51',51);
OU
INSERT INTO PROJETO VALUES(52,'PROJETO_52','SÃO PAULO');
```

Importante

- O tipo de campo “Date” possui a inserção de dados no seguinte formato:
  - AAAA,MM,DD
  - '2010/01/01‘
- Portanto, no momento de inserção de dados é necessário considerar esta observação.

Importante

Para inserção no PostgreSQL deve-se utilizar ASPAS SIMPLES e NÃO ASPAS DUPLAS

Ao trabalhar com datas devemos observar o padrão internacional de datas (Ano,Mês,Dia)

CORRETO
```
insert into pedido values (10,'Marcos','2015-12-31',10.00)
```

PERMITIDO
(MAS NÃO É RECOMENDAVEL - Data não está no padrão internacional - ano-mes-dia)
```
insert into pedido values (10,'Marcos','31-12-2015',10.00)
```

ERRADO
(SEM ASPAS)
```
insert into pedido values (10,'Marcos',2015/12/31,10.00)
```
(SEM ASPAS)
```
insert into pedido values (10,'Marcos',2015-12-31,10.00)
```
(ASPAS DUPLAS na data - tem que ser simples)
```
insert into pedido values (10,'Marcos',"2015-12-31",10.00)
```
(SEM CARACTERE SEPARADOR NA DATA)
```
insert into pedido values (10,'Marcos','31122015',10.00)
```

Múltipla inserção

- Insert também pode usar sintaxe para inclusão de múltiplas linhas, trabalhando com múltiplas listas de valores para os campos/colunas, cada uma envolvida por parenteses e separadas por vírgula.

DML (Data Manipulation Language)

- INSERT INTO: **Inserção de** dados (ação múltiplas linhas)

```
INSERT [INTO] <table_name>
      [{(column_list)}]
VALUES (expression [, expression ] ...),(expression [, expression ] ...);
```

Execução de código:

```
INSERT INTO tbl_name (a,b,c) VALUES(1,2,3),(4,5,6),(7,8,9);

OU

INSERT INTO PROJETO (numero,nome,localizacao)
VALUES(53,'PROJETO_53','SÃO
PAULO'),(54,'PROJETO_54','BRASILIA'),(55,'PROJETO_5','VITORIA');
```

Insira múltiplas linhas na tabela historico_salario

HISTORICO_SALARIO			
RG	DATA_INI_SAL	DATA_FIM_SAL	SALARIO
1010	30/11/2010	30/11/2015	5000.00
2020	30/11/2010	30/11/2012	5000.00
4040	31/10/2009	31/10/2010	2500.00
4040	31/10/2012	31/10/2015	4500.00
7070	31/07/2008	31/07/2014	5000.00
1010	30/04/2011	30/04/2013	6000.00
2020	31/12/2007	31/12/2008	3500.00
4040	30/09/2010	30/09/2012	5500.00

Resposta de inserção múltipla

```
INSERT INTO historico_salario (RG,DAT_INI_SAL,DAT_FIM_SAL,SALARIO)
VALUES (1010,   '2010/11/30',   '2015/11/30',   5000.00),
       (2020,   '2010/11/30',   '2012/11/30',   5000.00),
       (4040,   '2009/10/31',   '2010/10/31',   2500.00),
       ...
       (4040,   '2010/09/30',   '2012/09/30',   5500.00);
```

Ou ainda (sem nome dos campos)
Resposta de inserção múltipla

```
INSERT INTO historico_salario
VALUES (1010,   '2010/11/30',   '2015/11/30',   5000.00),
       (2020,   '2010/11/30',   '2012/11/30',   5000.00),
       (4040,   '2009/10/31',   '2010/10/31',   2500.00),
       ...
       (4040,   '2010/09/30',   '2012/09/30',   5500.00);
```

INSTITUTO FEDERAL
ESPÍRITO SANTO

SELEÇÃO DE DADOS

Filtrando dados com operadores de comparação

- = → igualdade
- <> → diferença
- <= → menor ou igual
- < → menor
- >= → maior ou igual
- > → maior
- IS NULL → Valores nulos
- IS NOT NULL → Valores não nulos

DQL (Data Query Language)

- Select: **Seleção** dados (múltiplas linhas)

```
SELECT [all | distinct ] select_list
[FROM <table_name>
[ WHERE search_conditions]
```

Execução de código:

```
select * from projeto where numero=5;
OU
select localizacao from projeto where numero>5;
OU
select * from empregado where RG_SUPERVISOR is not null;
```

Obs:

- Campos do tipo data e tipo money devem aparecer no where entre aspas simples

INSTITUTO FEDERAL
ESPÍRITO SANTO

ATUALIZAÇÃO, EXCLUSÃO E CÓPIA DE DADOS

DML (Data Manipulation Language)

- UPDATE: **Atualização** de dados (múltiplas linhas)

```
UPDATE <table_name>
SET column_name = {expression | null }
    [, column_name = {expression | null}]
[ WHERE search_conditions ]
```

Definições:
table_name → nome da tabela que sofrerá a alteração
column_name→ nome da coluna que sofrerá alteração
search_condition → indica as restrições que deverão ser aplicadas ao conjunto de linhas envolvidas de forma a restringir o universo de ação do comando UPDATE.

DML (Data Manipulation Language)

- UPDATE: **Atualização** de dados (múltiplas linhas)

```
UPDATE <table_name>
SET column_name = {expression | null }
    [, column_name = {expression | null}]
[ WHERE search_conditions ]
```

Execução de código:

```
UPDATE projeto set nome = 'Aguas limpas' where numero=25;
OU
UPDATE dependente set sexo='M' where codigo=5;
```

DML (Data Manipulation Language)

- DELETE: **Exclusão** de dados (múltiplas linhas)

DELETE FROM {table_name}
[**WHERE** search_conditions]

Definições:
table_name → nome da tabela que sofrerá a alteração/exclusão.
search_condition → indica as restrições que deverão ser aplicadas ao conjunto de linhas envolvidas de forma a restringir o universo de ação do comando DELETE.

DML (Data Manipulation Language)

- DELETE: **Exclusão** de dados (múltiplas linhas)

DELETE FROM {table_name}
[**WHERE** search_conditions]

Execução de código:

```
DELETE FROM PROJETO WHERE NUMERO=50;
OU
DELETE FROM PROJETO WHERE NUMERO>=50;
```

Cópia de dados e estrutura de uma tabela

- Para garantir a proteção dos dados e possível restauração destes na manipulação por atualizações e deleções, vamos fazer uma cópia (backup) do dados das tabelas a serem manipuladas
- CREATE TABLE NovaTabela SELECT * FROM AntigaTabela;

DML (Data Manipulation Language)

- Copiando estrutura e dados de uma tabela

CREATE TABLE {new_table_name} AS SELECT * FROM {old_table_name};

Execução de código:

CREATE TABLE nova_projeto **AS SELECT** * **FROM** projeto;
OU
CREATE TABLE nova_projeto **AS** (**SELECT** * **FROM** projeto)
OU
CREATE TABLE nova_projeto **AS** (**SELECT** * **FROM** projeto **WHERE** numero>10)

DML (Data Manipulation Language)

- Inserindo dados selecionados a partir de outra tabela

INSERT INTO {target_table_name} (list fields target table,...)
SELECT (list fields source table,...)
FROM {source_table_name}
[WHERE search_conditions]

Execução de código:

```
INSERT INTO DEPARTAMENTO_PROJETO (codigo, numero_depto,numero_projeto)
SELECT codigo, numero_depto,numero_projeto
FROM DEPARTAMENTO_PROJETO2
OU
INSERT INTO DEPARTAMENTO_PROJETO (codigo, numero_depto,numero_projeto)
SELECT codigo, numero_depto,numero_projeto
FROM DEPARTAMENTO_PROJETO2 WHERE codigo>10
```

Exercícios

Após realização dos exercícios envie as instruções que foram utilizadas em um arquivo txt para o professor por meio do Ava-IFES
(OBS: não use acentos nos nomes de campos)

1. Crie as seguintes tabelas e seus respectivos campos no banco de dados BDBSI4P:
   a. PROJETO: numero, nome e localização;
   b. EMPREGADO: rg, nome,cpf,depto,rg_supervisor,salario e dat_init_sal
   c. DEPARTAMENTO: numero, nome e rg_gerente
   d. DEPENDENTE: codigo, rg_responsável,nome_dependente,nascimento, relacao e sexo
   e. EMPREGADO_PROJETO: codigo,rg_empregado,numero_projeto e horas
   f. DEPARTAMENTO_PROJETO: codigo, numero_depto e numero_projeto
   g. HISTORICO_SALARIO: rg, dat_ini_sal,dat_fim_sal e salario

Exercícios

(OBS: não use acentos nos nomes de campos)

1. Altere a tabela PROJETO
   a. Exclua o campo localizacao
   b. Adicione o campo localidade
   c. Adicione o campo logico
2. Altere a tabela EMPREGADO
   a. Crie o campo funcao
   b. Altere o campo nome para nome_completo
   c. Apague o campo "dat_ini_sal"
3. Altere a tabela DEPARTAMENTO
   a. Crie o campo descricao_departamento
   b. Crie o campo data_criacao
   c. Altere o nome para nome_departamento
   d. Apague o campo data_criacao
4. Cria uma tabela denominada ALUNO com os campos código e nome
5. Altere em um único comando os campos codigo para tipo float e nome para varchar(30)

Exercícios

6. Apague a tabela Aluno
7. Altere a tabela historico_salario
   a. Exclua o campo dat_ini_sal
   b. Adicione o campo salario_total_anual
   c. Altere o campo salario para salario_mensal
8. Altere a tabela empregado_projeto
   a. Crie o campo função
   b. Crie o campo data_inicio
   c. Crie o campo data_fim
9. Altere a tabela Departamento_projeto
   a. Crie o campo demanda_de_funcionarios
   b. Crie o campo horas_destinadas
   c. Renomear o campo codigo para número

Exercícios

Insira os dados apresentados nas tabelas do banco de dados (faça as modificações da estrutura, caso exista necessidade)

EMPREGADO_PROJETO

CODIGO	RG_EMPREGADO	NUMERO_PROJETO	HORAS
1	2020	5	10
2	2020	10	25
3	3030	5	35
4	4040	20	50
5	5050	20	35
6	8080	5	70
7	7070	12	35
8	1010	12	12

HISTORICO_SALARIO

RG	DAT_INI_SAL	DAT_FIM_SAL	SALARIO
1010	01/01/2010	30/11/2010	2000.00
1010	01/12/2010	30/04/2011	4000.00
2020	01/05/2007	31/12/2007	2500.00
2020	01/01/2008	30/11/2010	4000.00
4040	01/08/2008	31/10/2009	1500.00
4040	01/11/2009	30/09/2010	2500.00
4040	01/10/2010	31/10/2012	3500.00
7070	01/01/2008	31/07/2008	1000.00

DEPENDENTE

CODIGO	RG_RESPONSAVEL	NOME_DEPENDENTE	NASCIMENTO	RELACAO	SEXO
1	1010	Jorge	1986-12-27	Filho	M
2	1010	Luiz	1979-11-18	Filho	M
3	2020	Fernanda Carla	1969-02-14	Cônjuge	F
4	2020	Ângelo	1995-02-10	Filho	M
5	3030	André	1990-05-01	Filho	M
6	8080	Ana Maria	1980-06-30	Cônjuge	F
7	8080	Karla Cristina	1999-08-25	Filha	F

DEPARTAMENTO

NUMERO	NOME	RG_GERENTE
1	Contabilidade	1010
2	Engenharia Civil	3030
3	Engenharia Mecânica	2020
4	Industrial	

DEPARTAMENTO_PROJETO

CODIGO	NUMERO_DEPTO	NUMERO_PROJETO
1	2	5
2	3	10
3	2	20

EMPREGADO

RG	NOME	CPF	DEPTO	RG_SUPERVISOR	SALARIO	DAT_INI_SAL
1010	João Luiz	11111	1		6000.00	2011-05-01
2020	Fernanda	22222	1	1010	5500.00	2008-12-01
3030	Ricardo	33333	2	2020	2300.00	2009-08-01
4040	Jorge	44444	2	3030	3200.00	2010-10-01
5050	Renata	55555	2	3030	1300.00	2012-02-01
6060	Luiz Renato	66666	3	2020	3000.00	2012-05-01
7070	LuizFernando	77777	3	6060	2000.00	2008-08-01
8080	João Antonio	88888	1	2020	3950.00	2010-07-01

PROJETO

NUMERO	NOME	LOCALIZACAO
5	Financeiro	São Paulo
10	Motor	Rio Claro
20	Prédio Central	Campinas
25	Águas Limpas	Vitória

Exercícios

(obs: quando necessário, visualize todos os dados da tabela para conferir os resultados)

1) Selecione todos os registros dos campos (atributos) nome, localização e número da tabela projeto (mostre os atributos nessa ordem).
2) selecione todos os registros e campos (atributos) da tabela projeto onde a localização é igual a "Vitória"
3) Mostre nome e número, para os projetos onde a localização consta como "Vitória"
4) Mostre todos campos (atributos) e registros da tabela projeto com número maior que 10
5) Mostre todos os campos e registros da tabela empregado com salário acima de 3000.
6) Mostre todos os campos e registros da tabela empregado onde RG_SUPERVISOR é igual a 1010.
7) Mostre todos os campos e registros da tabela empregado onde o campo RG corresponde a 4040
8) Mostre todos os campos e registros da tabela historico_salario onde o campo RG corresponde a 4040
9) Mostre todos os campos e registros da tabela empregado onde RG_SUPERVISOR é menor que 2020.
10) selecione todos os campos e registros da tabela projeto com valores de número maior ou igual a 10

Exercícios

11) Mostre todos os registros da tabela empregado onde RG_SUPERVISOR é menor ou igual 2020.
12) Mostre todos os registros da tabela empregado onde RG_SUPERVISOR é nulo
13) Mostre todos os registros da tabela empregado onde RG_SUPERVISOR não é nulo
14) Mostre todos os registros da tabela empregado onde DEPTO é diferente de 2
15) Mostre o campos nome e cpf e depto da tabela empregado onde o campo DEPTO corresponde a 2
16) Mostre o campos rg e nome tabela empregado onde o campo DEPTO corresponde a 1
17) Mostre apenas os nomes dos empregados que possuem salario maior que 5500
18) Mostre apenas os nomes dos empregados que possuem salario maior ou igual 5500
19) Mostre todos os empregados que data de salario inicial maior que 01-02-2012
20) Mostre todos os empregados que data de salario inicial maior que 01-05-2012
21) Mostre apenas os nomes dos empregados que data de salario inicial maior ou igual a 01-05-2012
22) Mostre apenas os cpfs dos empregados que data de salario inicial maior ou igual a 01-05-2012

Exercícios de inclusão, alteração e exclusão

1. Faça uma cópia das seguintes tabelas com todos os seus registros por meio da instrução create table:
   1. DEPARTAMENTO_PROJETO para DEPARTAMENTO_PROJETO2,
   2. PROJETO para PROJETO2,
   3. EMPREGADO para EMPREGADO2,
   4. DEPARTAMENTO para DEPARTAMENTO2
2. Insira na tabela correspondente os seguintes projetos
   1. Ginásio de Esportes, de código 50 e localização Serra
   2. Teatro de código 51 e localização Vitória
3. Atualize o nome do projeto Águas Limpas para "Aguas Claras"
4. Retorne com o nome do projeto Aguas Claras para "Águas Limpas"
5. Defina a localização do projeto motor igual a "Serra"
6. Apague todos os registros da tabela departamento_projeto
7. Apague todos os registros da tabela projeto (houve algum ploblema, qual? Qual a solução para a questão?)
8. Reinsira todos os registros da tabela departamento_projeto com um único comando insert
9. Elimine as tabelas de backup criadas anteriormente (tabelas com nome terminado em '2')

Trabalho em Andamento!

- Escolha as 5 principais tabelas do seu banco de dados (o principal fluxo de informações)
- Aplique os conhecimentos obtidos nesta aula para criar as tabelas do seu banco de dados!
- Enviar arquivo com código para o GIT.

Atividades

Realizar as inserções de dados no seu trabalho

mínimo de 10 registros em cada tabela de relacionamentos NxN

Mínimo de 5 registros nas tabelas restantes

Realizar as atividades de alterações, manipulações e exclusões (caso necessárias)

Atualizar atividades no trabalho/Git (até ítem 9.2)