SQL

PostgreSQL

I – Criação de Tabelas

Disciplina: SCC0241 – Bases de Dados

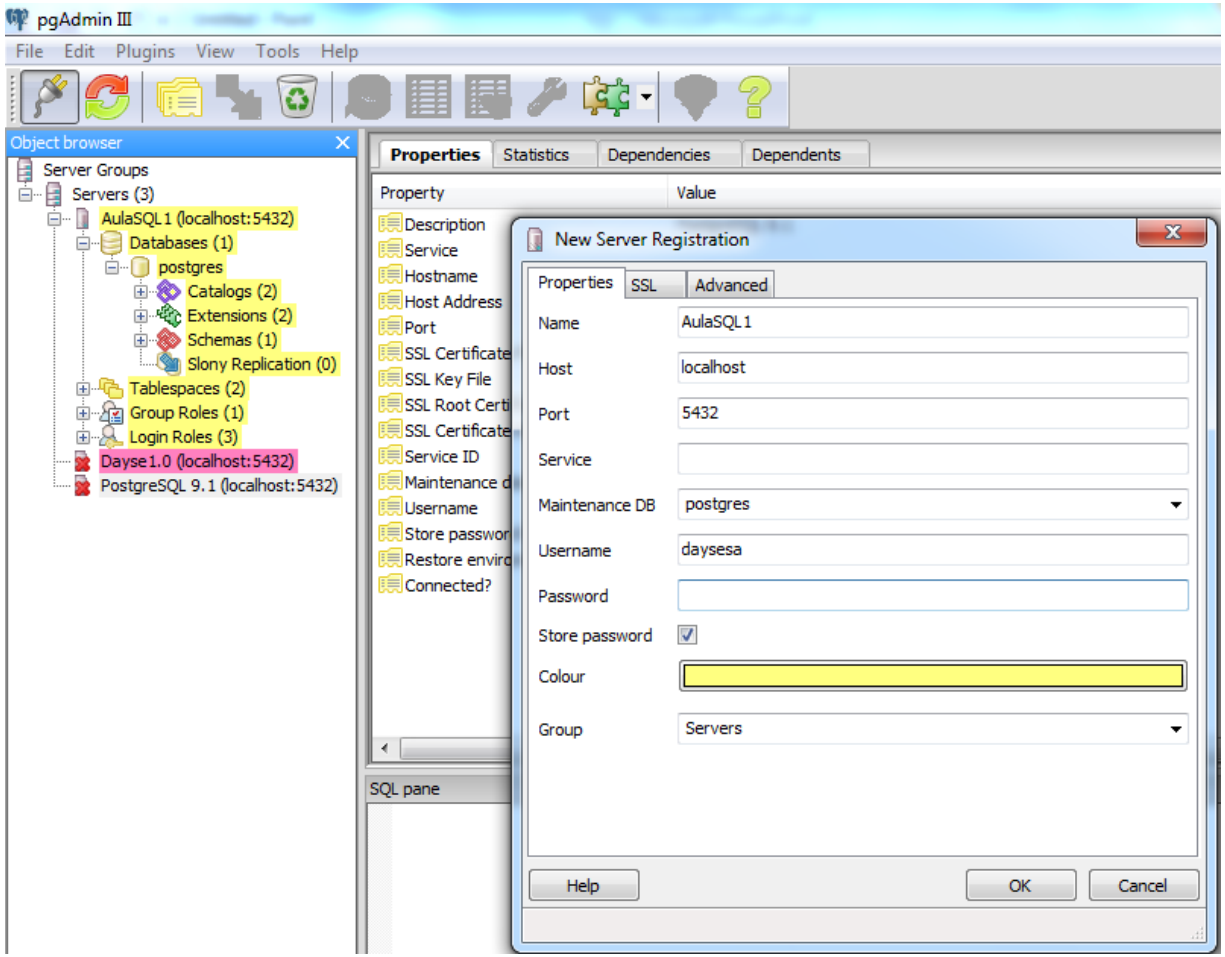
Professor: Eduardo Hruschka

Estagiária PAE: Dayse de Almeida

Composição da SQL

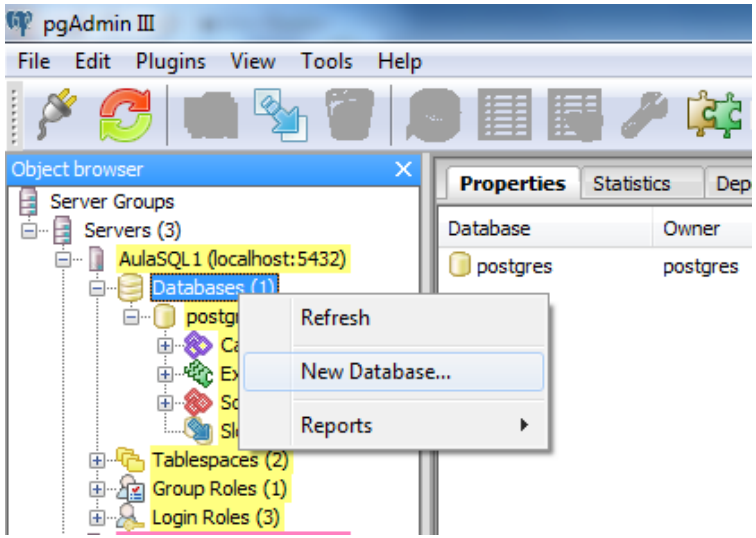
- Linguagem de Definição de Dados (DDL):
  - Comandos para definir, modificar e remover **tabelas**;
  - Além de criar e remover índices e visões.
- Linguagem de Manipulação de Dados (DML):
  - Comandos para criar, consultar, atualizar e remover **tuplas**.

Adicionar conexão com o servidor



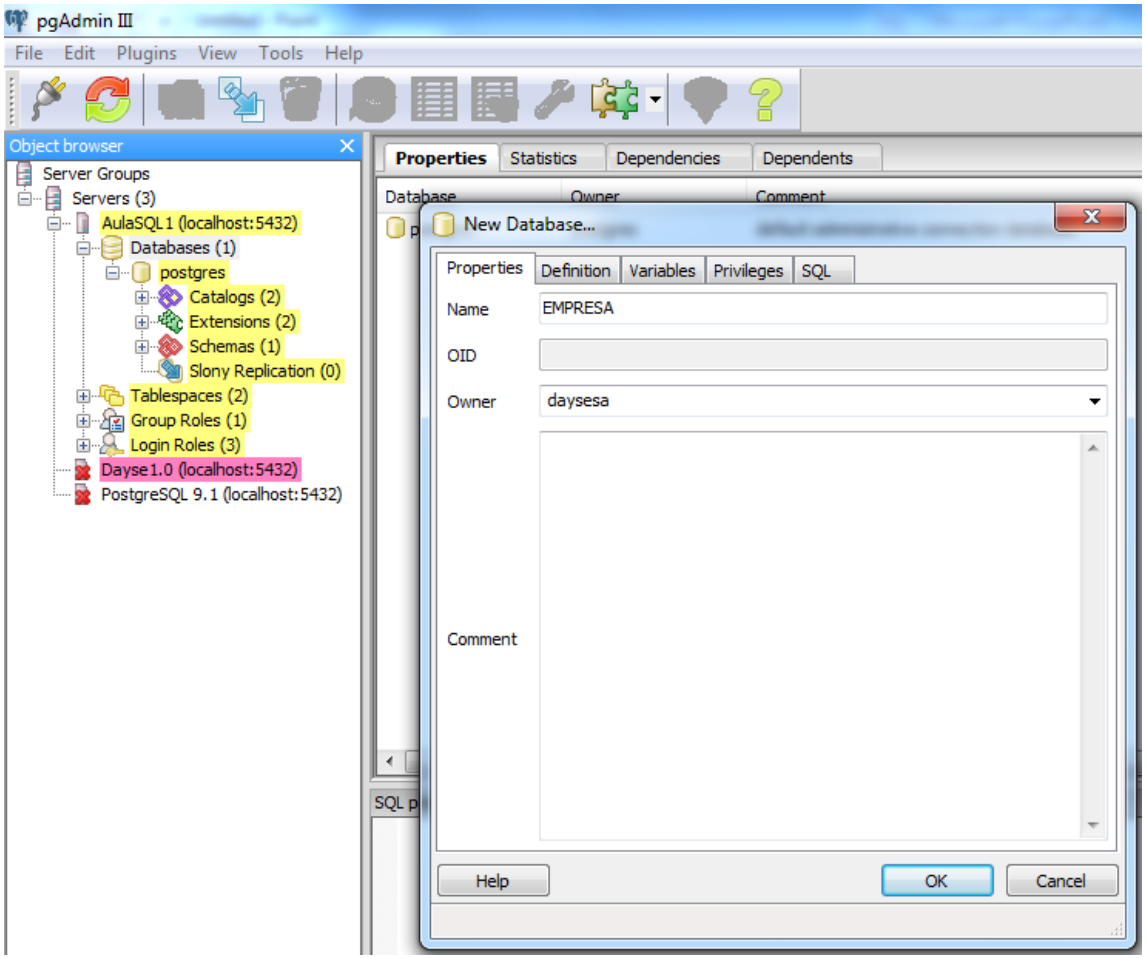
Create database

- Opção 1:



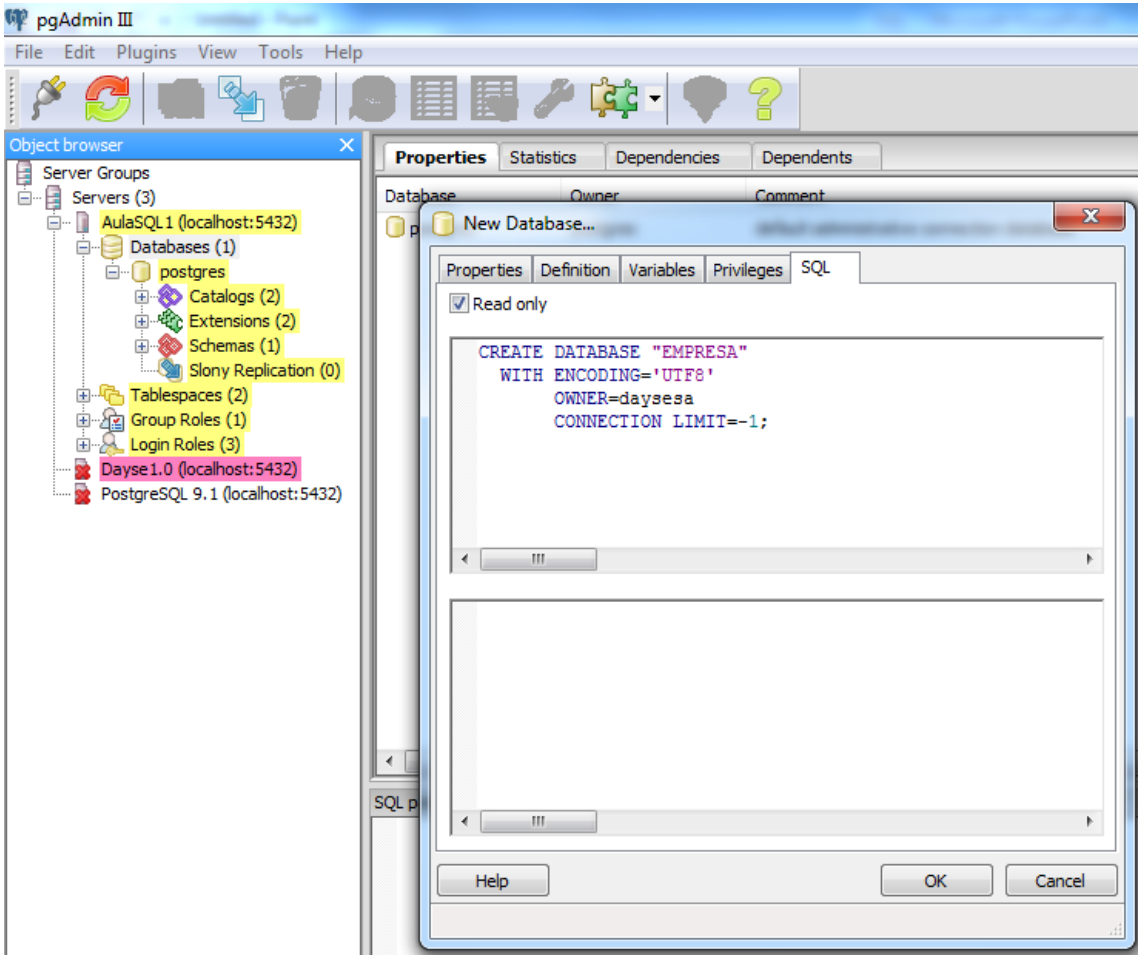
Create database

- Opção 1:



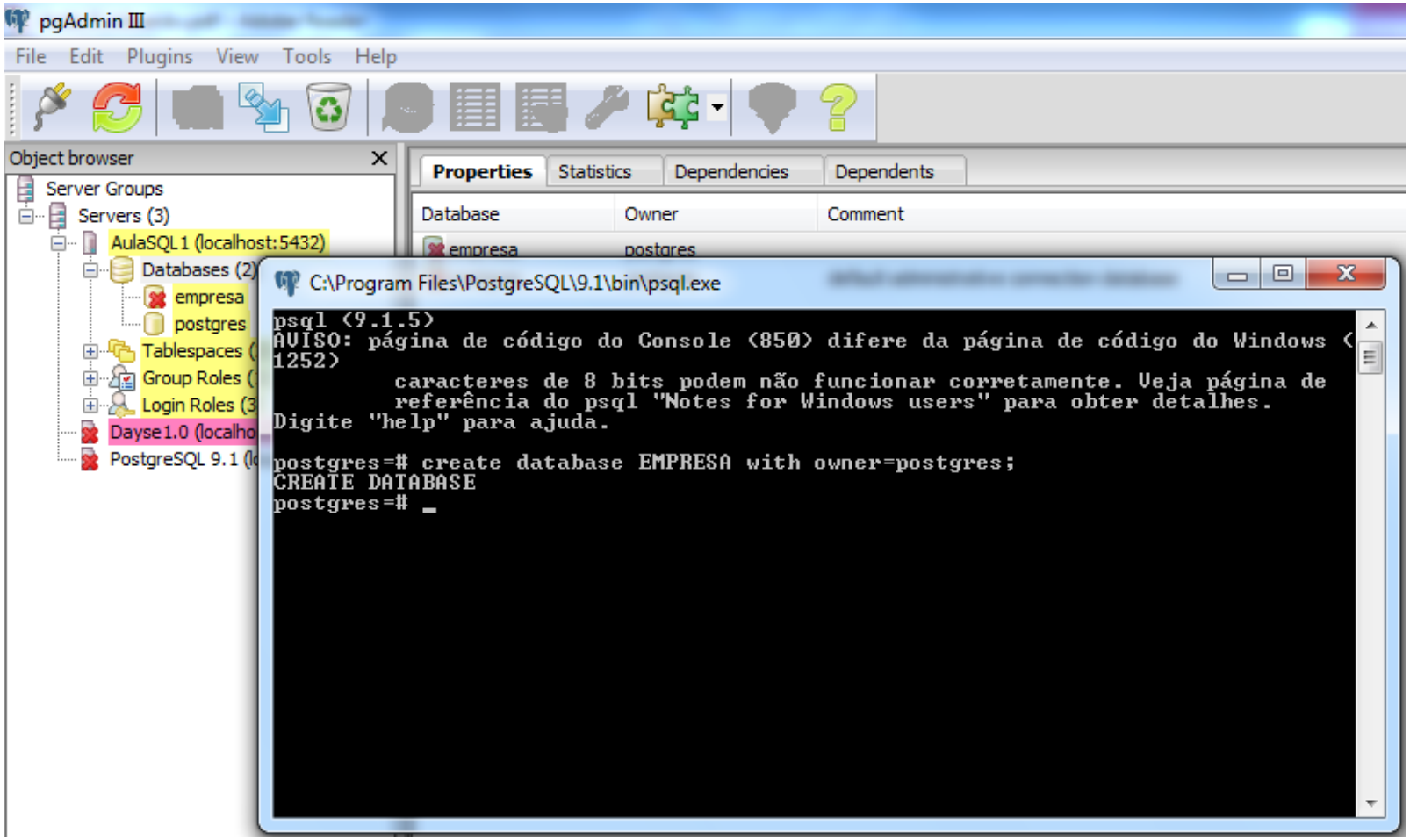
Create database

- Opção 1:



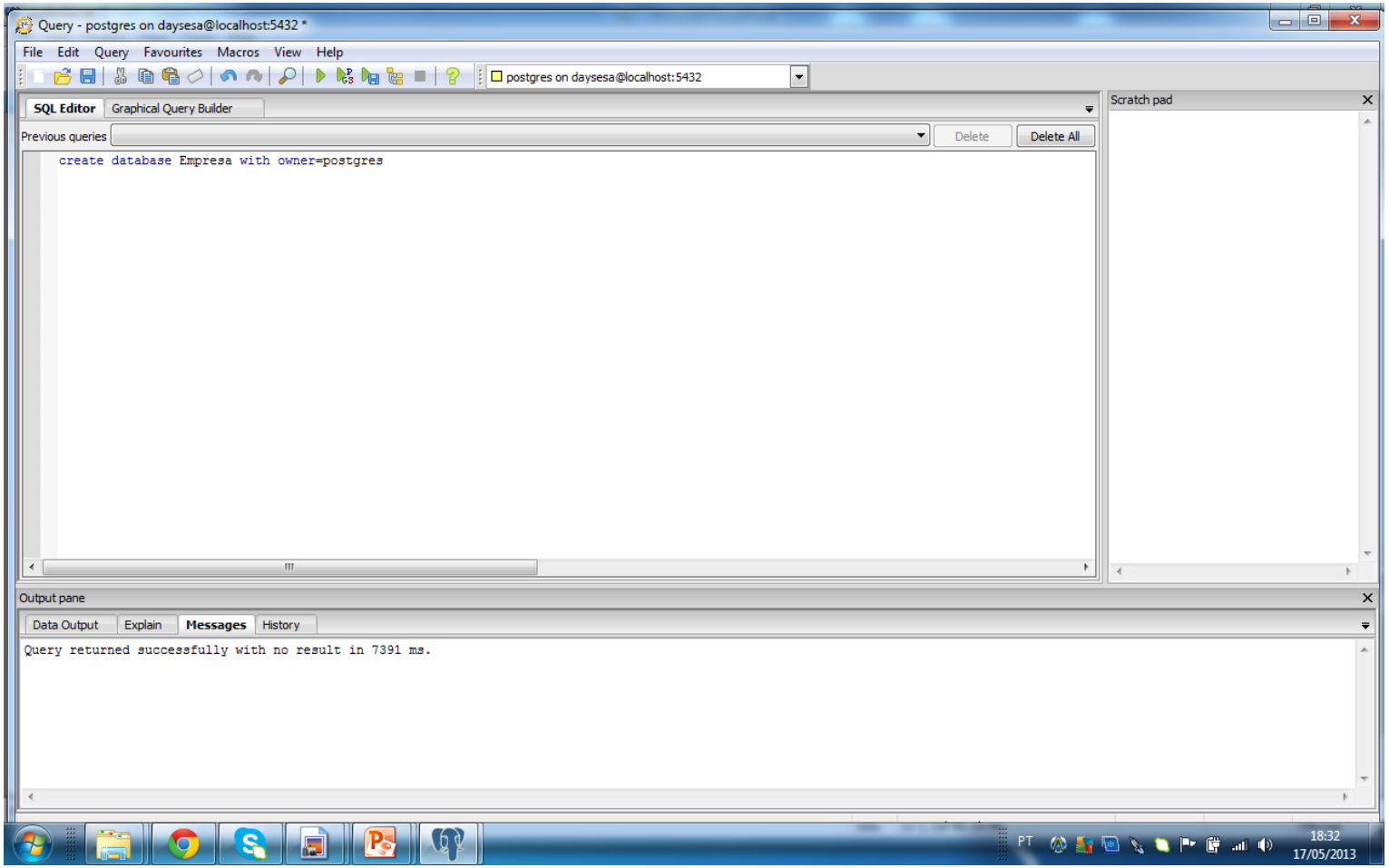
Create database

- Opção 2:



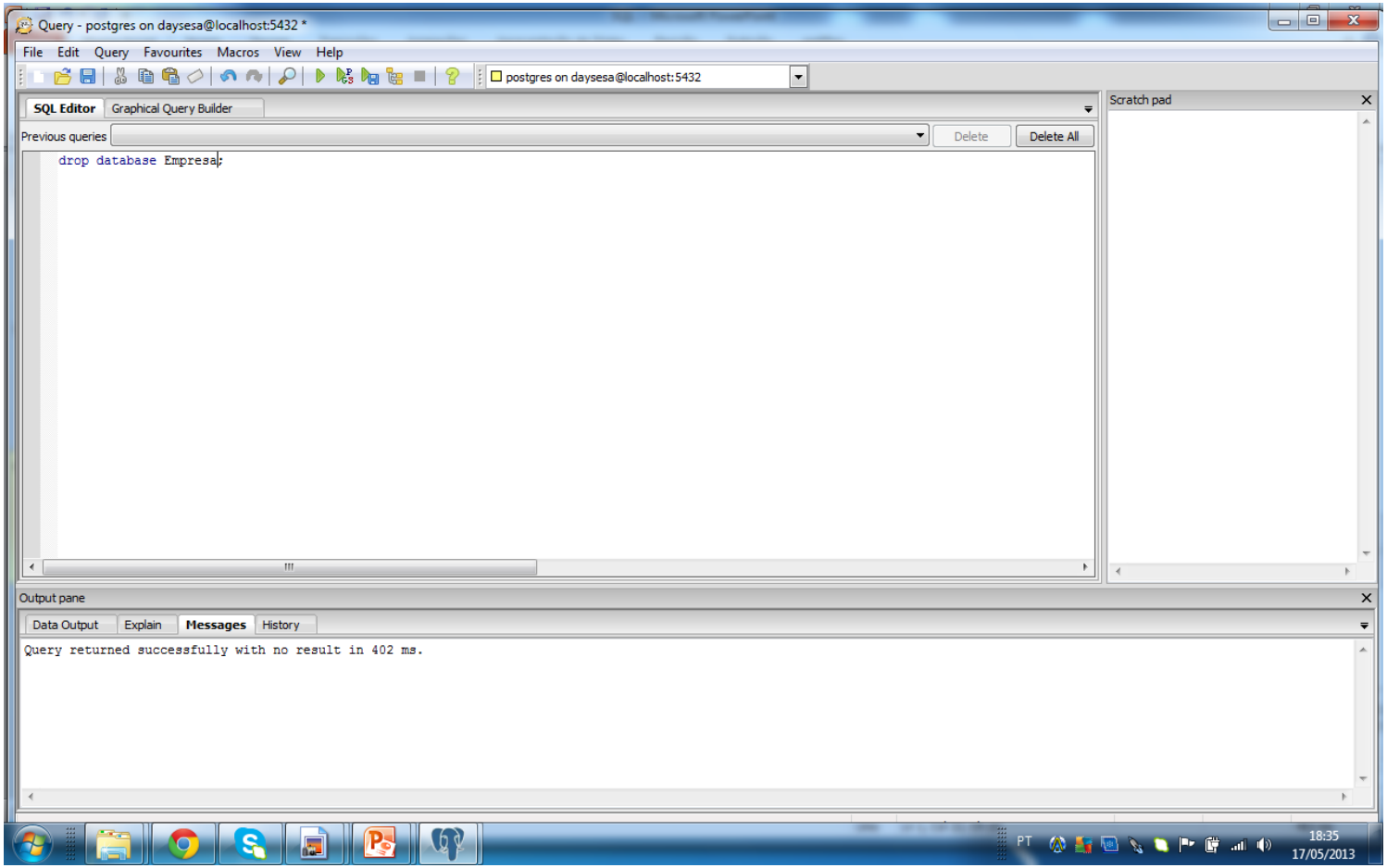
Create database

- Opção 3:



Drop database

- drop database Empresa;



Create database

- Criar um banco de dados para outro usuário:

  create database teste1 owner daysesa;

- Excluir:

  drop database teste1;

- Obs: login roles.

Create table

```
create table primeira_tabela(
primeiro_campo text,
segundo_campo integer);
```

- Drop table:

```
drop table primeira_tabela;
```

Create table

- Valor *default* para campos:
  - Ao definir um valor default para um campo, ao ser cadastrado o registro e este campo não for informado, o valor default é assumido.

```
create table produtos (
produto_no integer,
descricao text,
preco numeric default 9.99
);
```

- insert into produtos(produto_no, descricao, preco) values (45, 'qquer', 32);
- insert into produtos (produto_no, descricao) values (45, 'qquer');

Create table

- *Ckeck*:
  - Ao criar uma tabela podemos prever que o banco exija que o valor de um campo satisfaça uma expressão.

```
create table produtos2 (
produto_no integer,
descricao text,
preco numeric check (preco > 0)
);
```

- insert into produtos2 (produto_no, descricao, preco) values (45, 'qquer', 0);
  - ERRO: novo registro da relação "produtos3" viola restrição de verificação "produtos2_preco_check"

Create table

- Dar nome à restrição *check*:
  - Isso ajuda a tornar mais amigável as mensagens de erro.

```
CREATE TABLE produtos3 (
produto_no integer,
descricao text,
preco numeric CONSTRAINT preco_positivo CHECK (preco > 0)
);
```

- insert into produtos3 (produto_no, descricao, preco) values (45, 'qquer', 0);
  - ERRO: novo registro da relação "produtos4" viola restrição de verificação "preco_positivo“

Create table

- Dar nome à restrição *check*:
  - Isso ajuda a tornar mais amigável as mensagens de erro.

```
CREATE TABLE produtos4 (
        produto_no integer,
        descricao text,
        desconto numeric CHECK (desconto > 0 AND desconto < 0.10),
        preco numeric CONSTRAINT preco_positivo CHECK (preco > 0),
        CHECK (preco > desconto)
);
```

- insert into produtos4(produto_no, descricao, desconto, preco) values (45, 'qquer', 15, 100);
- insert into produtos4(produto_no, descricao, desconto, preco) values (45, 'qquer', 0.5, 0.4);

Create table

- Restrição *NOT NULL*:
  - Obriga o preenchimento de um campo.
  - Obs.: até um espaço em branco atende a esta restrição.

```
CREATE TABLE produtos5 (
cod_prod integer NOT NULL CHECK (cod_prod > 0),
nome text NOT NULL,
preco numeric
);
```

  - insert into produtos5 (cod_prod, nome, preco) values (-1, 'produtoX', 32);
    - ERRO: novo registro da relação "produtos6" viola restrição de verificação "produtos5_cod_prod_check“
  - insert into produtos5 (nome, preco) values ('produtoX', 32);
    - ERRO: valor nulo na coluna "cod_prod" viola a restrição não-nula

Create table

- Restrição *Unique*:
  - Valores exclusivos para cada campo em todos os registros;
  - Obs.: nulos não são checados. UNIQUE não aceita valores repetidos, mas aceita vários nulos (já que estes não são checados).

```
CREATE TABLE produtos6 (
cod_prod integer UNIQUE,
nome text,
preco numeric
);

CREATE TABLE produtos7(
cod_prod integer,
nome text,
preco numeric,
UNIQUE (cod_prod)
);
```

- insert into produtos7(cod_prod, nome, preco)values (45, 'produtoX', 34);
- insert into produtos7(cod_prod, nome, preco)values (45, 'produtoY', 23);

Create table

- Restrição *Unique*:

```
CREATE TABLE exemplo (
a integer,
b integer,
c integer,
UNIQUE (a, c)
);

CREATE TABLE produtos8(
cod_prod integer CONSTRAINT unq_cod_prod UNIQUE,
nome text,
preco numeric
);
```

Exercício

- Criar a tabela produtos9 com os seguintes atributos: cod_prod, nome e preco. O atributo cod_prod deve ser inteiro, único, não nulo e maior que 0. O atributo nome é do tipo texto e o atributo preco é do tipo numérico.

Exercício – Reposta

```
CREATE TABLE produtos9 (
    cod_prod integer UNIQUE NOT NULL
    CHECK(cod_prod > 0),
    nome text,
    preco numeric
);
```

Create table

- Chaves Primárias:
  - A chave primária de uma tabela é formada internamente pela combinação das restrições *UNIQUE* e *NOT NULL*;
  - Uma tabela pode ter no máximo uma chave primária;
  - A teoria de bancos de dados relacional dita que toda tabela deve ter uma chave primária;
  - O PostgreSQL não obriga que uma tabela tenha chave primária, mas é recomendável seguir, a não ser que esteja criando uma tabela para importar dados de outra que contenha registros duplicados para tratamento futuro, por exemplo.

Create table

- Chaves Primárias (*Primary Key*):

```
CREATE TABLE produtos10 (
cod_prod integer UNIQUE NOT NULL,
nome text,
preco numeric
);

CREATE TABLE produtos11 (
cod_prod integer PRIMARY KEY,
nome text,
preco numeric
);
```

  - Se mais que um atributo forma a chave primária:

```
CREATE TABLE exemplo (
a integer,
b integer,
c integer,
PRIMARY KEY (a, c)
);
```

Create table

- Chave Estrangeira (*Foreign Key*):
  - Criadas com o objetivo de relacionar duas tabelas, mantendo a integridade referencial entre ambas.
  - Especifica que o valor da coluna (ou grupo de colunas) deve corresponder a algum valor existente em um registro da outra tabela.
  - Na tabela estrangeira deve existir somente registros que tenham um registro relacionado na tabela principal.
  - Deve-se garantir que não se remova um registro na tabela principal que tenha registros relacionados na estrangeira.

Create table

- Chave Estrangeira (*Foreign Key*):

  – Tabela primária:

```
CREATE TABLE produtos11 (
cod_prod integer PRIMARY KEY,
nome text,
preco numeric
);

CREATE TABLE pedidos (
cod_pedido integer PRIMARY KEY,
cod_prod integer,
quantidade integer,
CONSTRAINT pedidos_fk FOREIGN KEY (cod_prod) REFERENCES produtos11 (cod_prod)
);
```

*

Create table

- Chave Estrangeira (*Foreign Key*):

```
CREATE TABLE t0 (
a integer,
b integer,
c integer,
PRIMARY KEY(a, b)
);

CREATE TABLE t1 (
a integer PRIMARY KEY,
b integer,
c integer,
d integer,
FOREIGN KEY (c, d) REFERENCES t0 (a, b)
);
```

Create table

- Simulando Enum:

```
CREATE TABLE pessoa(
codigo int PRIMARY KEY,
cor_favorita varchar(255) NOT NULL,
check (cor_favorita IN ('vermelha', 'verde', 'azul'))
);
```

- `INSERT INTO pessoa (codigo, cor_favorita) values (1, 'vermelha');`
- `INSERT INTO pessoa (codigo, cor_favorita) values (1, 'amarela');`

Create table

- Herança:
  Pode-se criar uma tabela que herda todos os campos de outra tabela existente.

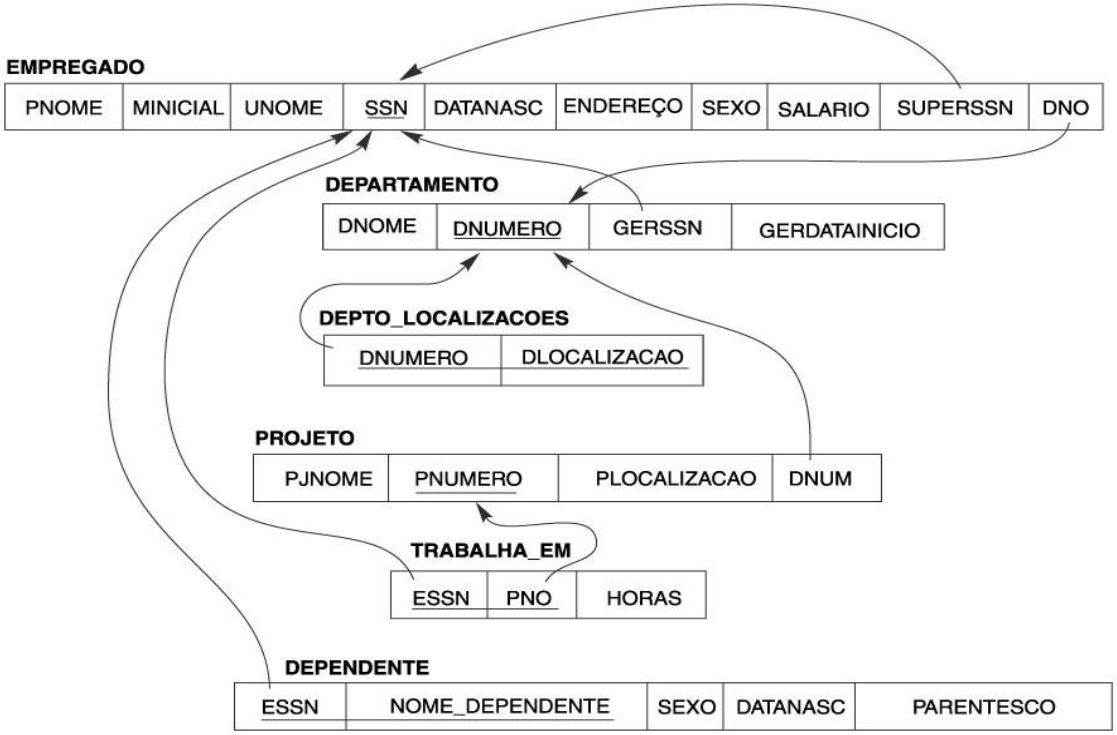
```
CREATE TABLE cidades (
nome text,
populacao float,
altitude int
);

CREATE TABLE capitais (
estado char(2)
) INHERITS (cidades);
```

  – Assim, capitais passa a ter também todos os campos da tabela cidades.

Exercício

- Criar relações com atributos, chaves primárias e chaves estrangeiras para:



Exercício - Create schema

- CREATE SCHEMA Esquema2;

- CREATE TABLE ESQUEMA2.TESTE(A1 int, A2 int);

Exercício – Resposta

```
CREATE TABLE EMPREGADO(
        PNOME varchar(255) NOT NULL,
        MINICIAL char(1),
        UNOME varchar(255) NOT NULL,
        SSN integer PRIMARY KEY,
        DATANASC date,
        ENDERECO varchar(255),
        SEXO char(2),
        CHECK (SEXO IN ('F', 'M')),
        SALARIO numeric NOT NULL CHECK (SALARIO > 0),
        SUPERSSN integer,
        CONSTRAINT supervisor FOREIGN KEY (SUPERSSN) REFERENCES EMPREGADO (SSN),
        DNO integer CHECK (DNO > 0),
        FOREIGN KEY (DNO) REFERENCES DEPARTAMENTO (DNUMERO)
);
```

- ERRO: relação "departamento" não existe

Exercício – Resposta

```
CREATE TABLE EMPREGADO(
    PNOME varchar(255) NOT NULL,
    MINICIAL char(1),
    UNOME varchar(255) NOT NULL,
    SSN integer PRIMARY KEY,
    DATANASC date,
    ENDERECO varchar(255),
    SEXO char(2),
    CHECK (SEXO IN ('F', 'M')),
    SALARIO numeric NOT NULL CHECK (SALARIO > 0),
    SUPERSSN integer,
    CONSTRAINT supervisor FOREIGN KEY (SUPERSSN) REFERENCES EMPREGADO
    (SSN),
    DNO integer CHECK (DNO > 0)
);
```

Exercício – Resposta

```
CREATE TABLE DEPARTAMENTO(
    DNOME varchar(255) NOT NULL,
    DNUMERO integer PRIMARY KEY CHECK (DNUMERO > 0),
    GERSSN integer,
    CONSTRAINT gerente FOREIGN KEY (GERSSN) REFERENCES
    EMPREGADO (SSN),
    GERDATAINICIO date
);
```

Exercício – Resposta

- ALTER TABLE:

```
ALTER TABLE EMPREGADO
        ADD CONSTRAINT numerodepto FOREIGN    KEY (DNO)
REFERENCES DEPARTAMENTO (DNUMERO)
;
```

Exercício – Resposta

```
CREATE TABLE DEPTO_LOCALIZACOES(
        DNUMERO integer,
        DLOCALIZACAO varchar(255),
        PRIMARY KEY(DNUMERO, DLOCALIZACAO),
        CONSTRAINT numerodepartamento  FOREIGN KEY
        (DNUMERO) REFERENCES DEPARTAMENTO (DNUMERO)
);
```

Exercício – Resposta

```
CREATE TABLE PROJETO(
    PJNOME varchar(255),
    PNUMERO integer PRIMARY KEY,
    PLOCALIZACAO varchar(255),
    DNUM integer,
    CONSTRAINT numerodepartamento FOREIGN KEY
    (DNUM) REFERENCES DEPARTAMENTO (DNUMERO)
);
```

Exercício – Resposta

```
CREATE TABLE TRABALHA_EM(
    ESSN integer,
    PNO integer,
    PRIMARY KEY (ESSN, PNO),
    HORAS real,
    CONSTRAINT numeroprojeto FOREIGN KEY (PNO)
    REFERENCES PROJETO (PNUMERO)
);
```

Exercício – Resposta

```
CREATE TABLE DEPENDENTE(
    ESSN integer,
    NOME_DEPENDENTE varchar(255),
    SEXO char(1) CHECK (SEXO IN ('F', 'M')),
    DATANASC date,
    PARENTESCO varchar(255),
    PRIMARY KEY (ESSN, NOME_DEPENDENTE),
    CONSTRAINT SSNempregado FOREIGN KEY (ESSN)
    REFERENCES EMPREGADO (SSN)
);
```

Tipos Numéricos

Name	Storage Size	Description	Range
`smallint`	2 bytes	small-range integer	-32768 to +32767
`integer`	4 bytes	usual choice for integer	-2147483648 to +2147483647
`bigint`	8 bytes	large-range integer	-9223372036854775808 to 9223372036854775807
`decimal`	variable	user-specified precision, exact	no limit
`numeric`	variable	user-specified precision, exact	no limit
`real`	4 bytes	variable-precision, inexact	6 decimal digits precision
`double precision`	8 bytes	variable-precision, inexact	15 decimal digits precision
`serial`	4 bytes	autoincrementing integer	1 to 2147483647
`bigserial`	8 bytes	large autoincrementing integer	1 to 9223372036854775807

Tipos para data e hora

Nome	Tamanho de Armazenamento	Descrição	Menor valor	Maior valor	Resolução
timestamp [(p)] [without time zone]	8 bytes	tanto data quanto hora	4713 AC	5874897 DC	1 microssegundo / 14 dígitos
timestamp [(p)] with time zone	8 bytes	tanto data quanto hora, com zona horária	4713 AC	5874897 DC	1 microssegundo / 14 dígitos
interval [(p)]	12 bytes	intervalo de tempo	-178000000 anos	178000000 anos	1 microssegundo / 14 dígitos
date	4 bytes	somente data	4713 AC	32767 DC	1 dia
time [(p)] [without time zone]	8 bytes	somente a hora do dia	00:00:00.00	23:59:59.99	1 microssegundo / 14 dígitos
time [(p)] with time zone	12 bytes	somente a hora do dia, com zona horária	00:00:00.00+12	23:59:59.99-12	1 microssegundo / 14 dígitos

- Os tipos *time*, *timestamp*, e *interval* aceitam um valor opcional de precisão *p*, que especifica o número de dígitos fracionários mantidos no campo de segundos. Por padrão não existe limite explícito para a precisão. O intervalo permitido para *p* é de 0 a 6 para os tipos *timestamp* e *interval*.

SQL
II – Inserção e Atualização

Disciplina: SCC0241 – Bases de Dados

Professor: Eduardo Hruschka

Estagiária PAE: Dayse de Almeida

DML

- INSERT INTO ...
  - insere dados em uma tabela.

- UPDATE ... SET ... WHERE ...
  - altera dados específicos de uma tabela.

Insert

INSERT INTO nome_tabela

VALUES (V1, V2, Vn);

– Ordem dos atributos deve ser mantida.

INSERT INTO nome_tabela (A1, A2, An)

VALUES (V1, V2, Vn);

– Ordem dos atributos não precisa ser mantida.

Update

```
UPDATE nome_tabela
        SET coluna = <valor>
        WHERE predicado;
```

- Cláusula WHERE
  - É opcional.

Insert

```
INSERT INTO EMPREGADO (PNOME, MINICIAL, UNOME, SSN, DATANASC, ENDERECO, SEXO, SALARIO, SUPERSSN, DNO) values ('John', 'B', 'Smith', 123456789, '09/01/1965', '731 Fondren, Houston, Tx', 'M', 30000, null, null);

INSERT INTO EMPREGADO (PNOME, MINICIAL, UNOME, SSN, DATANASC, ENDERECO, SEXO, SALARIO, SUPERSSN) values (‘Franklin', ‘T', ‘Wong', 333445555, ‘08/12/1955', ‘638 Voss, Houston, Tx', 'M', 40000, null, null);
```

Update

```
UPDATE EMPREGADO
    SET SUPERSSN= 333445555
    WHERE SSN = 123456789;
```

Exercício

- Inserir as seguintes tuplas:

EMPREGADO	PNOME	MINICIAL	UNOME	SSN	DATANASC	ENDERECO	SEXO	SALARIO	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	*null*	1

DEPT_LOCALIZACOES	DNUMERO	DLOCALIZACAO
	1	Houston
	4	Stafford
	5	Bellaire
	5	Sugarland
		Houston

DEPARTAMENTO	DNOME	DNUMERO	GERSSN	GERDATAINICIO
	Pesquisa	5	333445555	1988-05-22
	Administração	4	987654321	1995-01-01
	Sede administrativa	1	888665555	1981-06-19

TRABALHA_EM	ESSN	PNO	HORAS
	123456789	1	32.5
	123456789	2	7.5
	666884444	3	40.0
	453453453	1	20.0
	453453453	2	20.0
	333445555	2	10.0
	333445555	3	10.0
	333445555	10	10.0
	333445555	20	10.0
	999887777	30	30.0
	999887777	10	10.0
	987987987	10	35.0
	987987987	30	5.0
	987654321	30	20.0
	987654321	20	15.0
	888665555	20	*null*

PROJETO	PJNOME	PNUMERO	PLOCALIZACAO	DNUM
	ProdutoX	1	Bellaire	5
	ProdutoY	2	Sugarland	5
	ProdutoZ	3	Houston	5
	Automatização	10	Stafford	4
	Reorganização	20	Houston	1
	Novos Benefícios	30	Stafford	4

DEPENDENTE	ESSN	NOME_DEPENDENTE	SEXO	DATANASC	PARENTESCO
	333445555	Alice	F	1986-04-05	FILHA
	333445555	Theodore	M	1983-10-25	FILHO
	333445555	Joy	F	1958-05-03	CÔNJUGE
	987654321	Abner	M	1942-02-28	CÔNJUGE
	123456789	Michael	M	1988-01-04	FILHO
	123456789	Alice	F	1988-12-30	FILHA
	123456789	Elizabeth	F	1967-05-05	CÔNJUGE