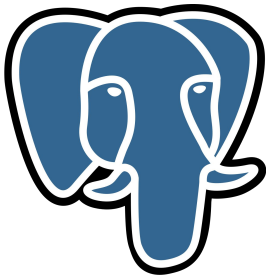
Banco de dados com PostGreSQL
Aula 4

Fernando Duwe

v 1.0

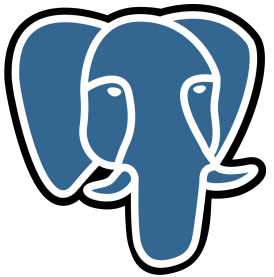
Transação ou Transaction é uma única unidade de trabalho processada pelo Sistema de Gerenciamento de Banco de Dados (SGBD). Imagine que precisamos realizar em uma única transação duas operações de manipulação de dados (DML, do inglês Data Manipulation Language), como INSERT e UPDATE. Estas operações só podem se tornar permanentes no banco de dados se todas forem executadas com sucesso. Em caso de falhas em uma das duas é possível cancelar a transação, porém todas modificações realizadas durante a mesma serão descartadas, inclusive a que obtive sucesso. Assim, os dados permanecem íntegros e consistentes.



BEGIN

Utilizando o comando BEGIN ou BEGIN TRANSACTION criamos uma nova transação.

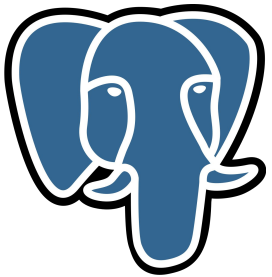
```
BEGIN TRANSACTION;
```



COMMIT

Confirma as alterações efetuadas dentro da transação no banco de dados.

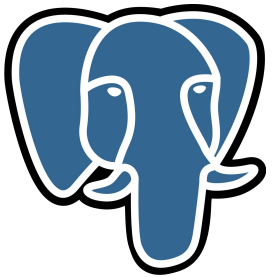
```
COMMIT;
```



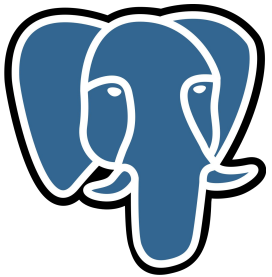
ROLLBACK

Rejeita as alterações efetuadas dentro da transação no banco de dados.

```
ROLLBACK;
```



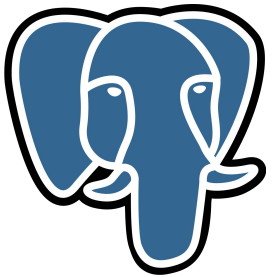
Todo agrupamento de bancos de dados possui um conjunto de usuários de banco de dados. Estes usuários são distintos dos usuários gerenciados pelo sistema operacional onde o servidor executa. Eles possuem objetos de banco de dados (por exemplo, tabelas, visões etc.), e podem conceder privilégios nestes objetos para outros usuários controlando, assim, quem pode acessar qual objeto.



GRANT

Dá uma determinada permissão para um determinado usuário/grupo de usuários.

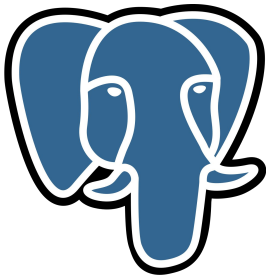
```
GRANT privileges ON object TO user;
```



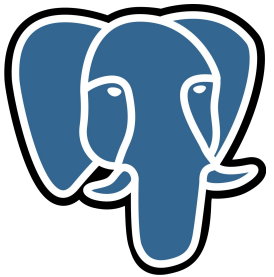
REVOKE

Revoga uma determinada permissão para um determinado usuário/grupo de usuários.

```
REVOKE privileges ON object FROM user;
```

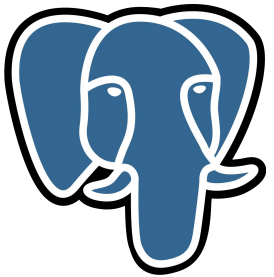


A view pode ser definida como uma tabela virtual composta por linhas e colunas de dados vindos de tabelas relacionadas em uma query (um agrupamento de SELECT's, por exemplo). As linhas e colunas da view são geradas dinamicamente no momento em que é feita uma referência a ela.

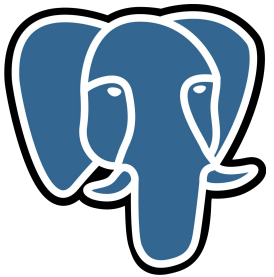


A declaração básica de uma view

```
CREATE VIEW VIEW_PONTO_FUNCIONARIO AS
SELECT NOME_FUNC, PROFISSAO, ENTRADA, HORA_ENTRADA
FROM FUNCIONARIOS, REGISTRO_PONTO
WHERE FUNCIONARIOS.CODIGO = REGISTRO_PONTO.CODFUNC;
```

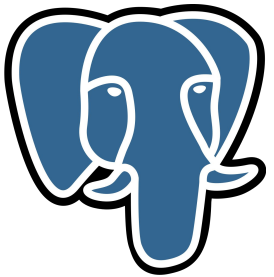


Em muitos SGDBs temos o conceito de Stored Procedures, programas desenvolvidos em uma determinada linguagem de script e armazenados no servidor, onde serão processados. No PostgreSQL, as Stored Procedures são conhecidas com o nome de Functions.



A declaração básica de uma função ou procedimento

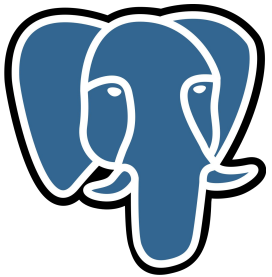
```
CREATE [OR REPLACE] FUNCTION FUNCTION_NAME(PARAM_LIST)
   RETURNS RETURN_TYPE
   LANGUAGE PLPGSQL
  AS
$$
DECLARE
-- Aqui declaramos as variáveis
BEGIN
-- Nossa lógica implementada
END;
$$
```



SELECT funcao()

Utilizamos o comando SELECT para disparar uma função.

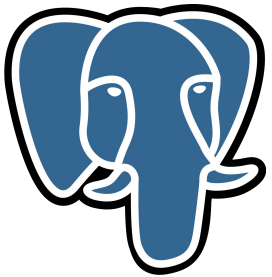
```
SELECT concat_lower_or_upper('Hello', 'World', true);
```



Declarando variáveis

Para declarar variáveis, declare-as na cláusula DECLARE

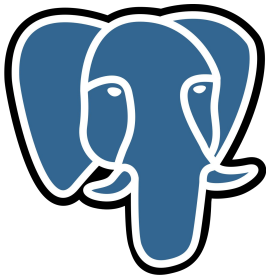
```
DECLARE
     CONTADOR INT;
     NOVO_NOME VARCHAR(60);
     NOVA_DATA DATE;
```



Efetuando querys e fazendo scripts

Basta executar os scripts. É possível utilizar as variáveis e parâmetros dentro das consultas.

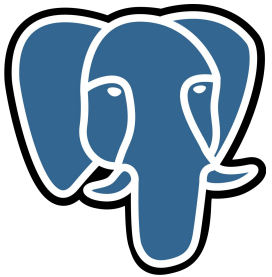
```
SELECT COUNT(*) IN CONTADOR
FROM CATEGORIA
WHERE ATIVO = TRUE;

INSERT INTO LOGS (COUNT, TIPO_LOG) VALUES (COUNT, 'Log');
```



IF

Cláusula If...Else...End

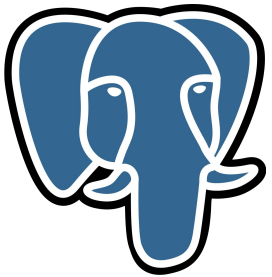
```
IF expressão-boleana THEN
    comandos
    ...
ELSE
    comandos
    ...
END IF;
```



WHILE

Loop While.

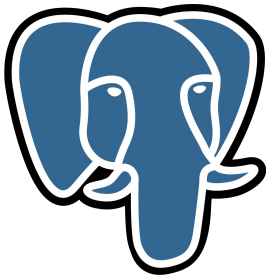
```
WHILE expressão LOOP
    Comandos
    ...
END LOOP;
```



Gerando exceções

É possível também gerar exceções. Nesse caso, a exceção será enviada para a consulta que está sendo efetuada ou estourará dentro da aplicação.

```
RAISE 'Mensagem de erro';
```



Efetuando um loop em uma query

É possível percorrer todos os registros de uma query.

```
DECLARE
     registro RECORD;
BEGIN
     Comandos;

     FOR registro IN SELECT * FROM TABELA1
                          ORDER BY ID
     LOOP
          INSERT INTO TABELA2(ID, DESCRICAO)
          VALUES (registro.ID, registro.DESCRICAO);
     END LOOP;

     Comandos;
END;
```