

SQL 1

Geomar A. Schreiner
gschreiner@uffs.edu.br

Roteiro

- Apresentação da disciplina
- Revisão SQL (início)

Apresentação da Disciplina

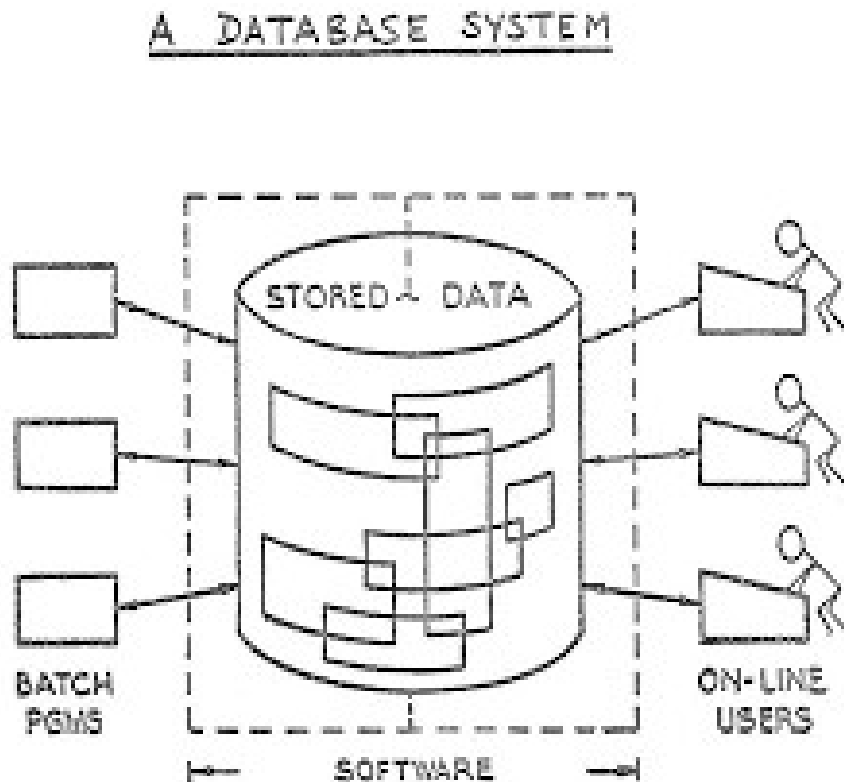
- Encontros SIGAA

Apresentação da Disciplina

Revisão SQL

Histórico

- 1970 foi publicado “A Relational Model of Data Large Shared Banks” que define o modelo relacional;



Histórico

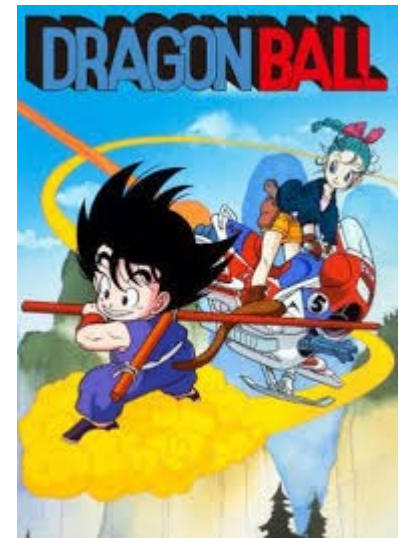
- 1970 foi publicado “A Relational Model of Data Large Shared Banks” que define o modelo relacional;
- 1974 é publicada SEQUEL (Structured English Query Language)

```
SELECT      ITEM
FROM        SALES
WHERE       DEPT =
            SELECT      DEPT
            FROM        LOC
            WHERE       FLOOR = '2'
```



Histórico

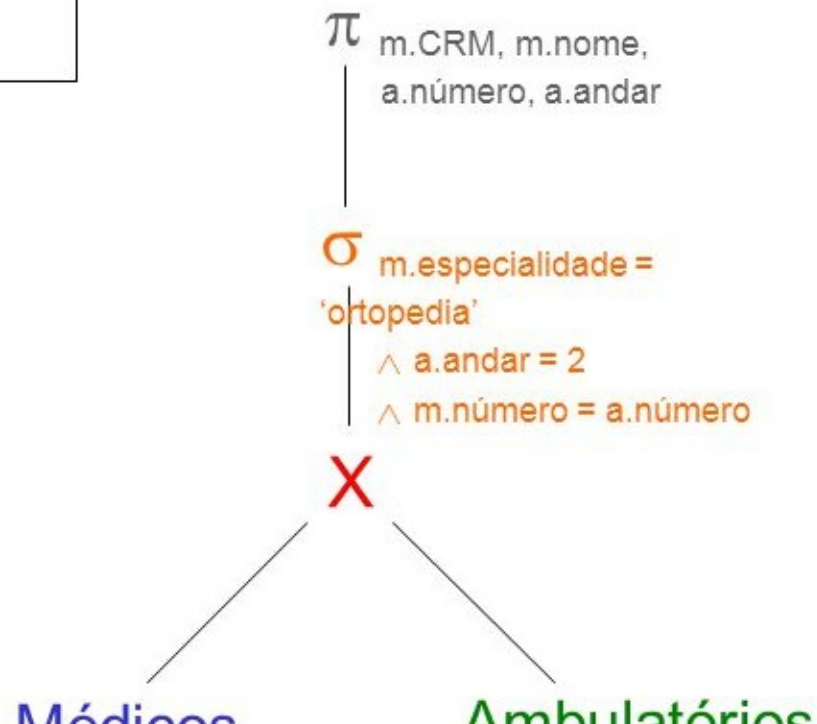
- 1970 foi publicado “A Relational Model of Data Large Shared Banks” que define o modelo relacional;
- 1974 é publicada SEQUEL (Structured English Query Language)
- 1986 SQL vira ISO
 - SQL-2(92); SQL-3(99)



Visão geral

- Possui base pautada na álgebra relacional e no cálculo relacional de tupla

```
select m.CRM, m.nome, a.número, a.andar  
from Médicos m, Ambulatórios a  
where m.especialidade = 'ortopedia'  
and a.andar = 2  
and m.número = a.número
```



Visão geral

- Possui base pautada na álgebra relacional e no cálculo relacional de tupla
- Principais funcionalidades:
 - DDL – definição de dados
 - DML – manipulação de dados
 - Restrições de integridade
 - Transações
 - Store Procedures

DDL

- Criação de um BD
 - SQL não define essa operação
 - Alguns BDs (maioria dos comerciais) possuem o comando de criação:
 - **CREATE DATABASE** nome_db
 - Remover um BD
 - **DROP DATABASE** nome_db

DDL

- Comandos para definição do esquema
 - CREATE TABLE
 - Cria uma nova tabela, com suas definições estruturais e de integridade
 - ALTER TABLE
 - Modifica as definições da tabela
 - Atributos chave não podem ser removidos
 - Atributos NOT NULL não podem ser inseridos
 - DROP TABLE
 - Remove uma tabela

DDL

- **CREATE TABLE**

- **CREATE TABLE** tabela (
 atributo_1 tipo_1,
 [atributo_n tipo_n,]
 [**PRIMARY KEY** (atributo_, [atributo_n])]
 [**FOREIGN KEY** (nome_atributo) **REFERENCES**
 nome_tabela]
);
- Principais tipos:
 - Int, smallint, numeric (definição e tamanho), char, varchar (definição de tamanho), date, datetime
 - Date → 'YYYY-MM-DD'

DDL

Name	Aliases	Description
<code>bigint</code>	<code>int8</code>	signed eight-byte integer
<code>bigserial</code>	<code>serial8</code>	autoincrementing eight-byte integer
<code>bit [(n)]</code>		fixed-length bit string
<code>bit varying [(n)]</code>	<code>varbit</code>	variable-length bit string
<code>boolean</code>	<code>bool</code>	logical Boolean (true/false)
<code>character [(n)]</code>	<code>char [(n)]</code>	fixed-length character string
<code>character varying [(n)]</code>	<code>varchar [(n)]</code>	variable-length character string
<code>date</code>		calendar date (year, month, day)
<code>double precision</code>	<code>float8</code>	double precision floating-point number (8 bytes)
<code>integer</code>	<code>int, int4</code>	signed four-byte integer
<code>numeric [(p, s)]</code>	<code>decimal [(p, s)]</code>	exact numeric of selectable precision
<code>smallint</code>	<code>int2</code>	signed two-byte integer
<code>text</code>		variable-length character string
<code>time [(p)] with time zone</code>	<code>timetz</code>	time of day, including time zone
<code>timestamp [(p)] with time zone</code>	<code>timestampz</code>	date and time, including time zone

DDL

- CREATE TABLE

- Exemplo

- CREATE TABLE alunos (
matricula NUMERIC(10),
nome VARCHAR(120),
sexo SMALLINT,
PRIMARY KEY (matricula)
);

DDL - Restrições

- Restrições (constraints) são construções previstas no SQL utilizados para definir um nível de controle mais apurado sobre os dados e seu domínio
- Geralmente, restrições podem ser definidas de duas maneiras
 - Através de DDL na construção de tabelas
 - Unique, not null, check
 - Adicionados posteriormente sob demanda
 - constraint

DDL - Restrições

- Checagem (check)
 - Permitem a especificação de valores válidos ou não para o campo
 - Podem referenciar outros campos
 - Avaliam uma expressão booleana

DML

- Utilizada para a manipulação dos dados
- Comandos
 - **INSERT**
 - **UPDATE**
 - **SELECT**
 - **DELETE**
- São instruções declarativas
 - Manipulação de conjuntos
 - Baseia-se no que deve ser feito

Exercícios

```
Tipos_Veiculos (codTipo, descricao);
Habilitacoes (codH, tipo, idade_min, descricao);
Veiculos (matricula, nome, modelo, comprimento, potMotor,
vlDiaria, codTipo (TiposVeiculos));
Funcionarios (codF, nome, telefone, endereco, idade, salario)
Veiculos_Habilitacoes (codTipo (Tipos_Veiculos), codH
(Habilitacoes));
Clientes (CPF, nome, endereco, estado_civil, num_filhos,
data_nasc, telefone, codH (Habilitacoes));
Locacoes (codLoc, valor, inicio, fim, obs, matricula
(Veiculos), codF (Funcionarios), CPF (Cliente));
```

Criar o BD conforme as regras disponíveis no portal;
Popular o BD;

Exercícios Part 1

- Popule o BD com os dados da pasta compartilhada
- Faça as seguintes atualizações
 - Um funcionário cadastrou o nome do cliente errado, atualize o nome do cliente com cpf 68745120480 para “João”
 - Mariana (cpf 23548754210) se divorciou, atualize a base de dados
 - O veículo código 103 está com o comprimento errado, o valor correto é 18 metros.
 - Todos os barcos devem sofrer uma alteração em suas diárias. Reajuste em 12.4% todos os valores de diárias
- O funcionário Marquito foi demitido, exclua ele da base.

Exercícios Part 2

- 1) Listar o nome e o estado civil e a data de nascimento de todos os clientes
- 2) Listar o nome, idade e telefone de todos os funcionarios
- 3) Liste as habilitações que necessitam que o usuário possua mais de 25 anos
- 4) Listar o nome dos veiculos que tem comprimento maior que 10 e com potencia superio a 120
- 5) Listar o nome e o comprimento de todos os bascos cuja potencia fique entre 50 e 300

Funções

Funções

- Programação estruturada
 - Procedimentos
 - Funções
- Em BD
 - Geralmente Procedures contém rollback e funções não
- Postgres
 - Tudo a mesma coisa

Funções

CREATE [OR REPLACE] FUNCTION

name([[argname] argtype])

[**RETURNS** tipo | [**TABLE** (cols)]]

AS \$\$

[**DECLARE** var tipo;]

BEGIN

operações

END;

\$\$ LANGUAGE plpgsql;

Funções

Exemplo – condicional

```
CREATE OR REPLACE FUNCTION numero_par (i int)
RETURNS boolean AS $$
DECLARE
    temp int;
BEGIN
    temp := i % 2;
    IF temp = 0 THEN RETURN true;
    ELSE RETURN false;
    END IF;
END;
$$ LANGUAGE plpgsql;

SELECT numero_par(3), numero_par(42);
```

Funções

Exemplo – laço FOR

```
CREATE OR REPLACE FUNCTION fatorial (i numeric)
RETURNS numeric AS $$
DECLARE
    temp numeric; resultado numeric;
BEGIN
    resultado := 1;
    FOR temp IN 1 .. i LOOP
        resultado := resultado * temp;
    END LOOP;
    RETURN resultado;
END;
$$ LANGUAGE plpgsql;

SELECT fatorial(42::numeric);
```

Funções

Exemplo – laço WHILE

```
CREATE OR REPLACE FUNCTION fatorial (i numeric)
RETURNS numeric AS $$
DECLARE temp numeric; resultado numeric;
BEGIN
    resultado := 1; temp := 1;
    WHILE temp <= i LOOP
        resultado := resultado * temp;
        temp := temp + 1;
    END LOOP;
    RETURN resultado;
END;
$$ LANGUAGE plpgsql;

SELECT fatorial(42::numeric);
```

Funções

Exemplo – SQL dinâmico

```
CREATE OR REPLACE FUNCTION recupera_funcionario(id int)
RETURNS funcionario AS $$
DECLARE
    registro RECORD;
BEGIN
    EXECUTE 'SELECT * FROM funcionario WHERE id = ' || id INTO
registro;
    RETURN registro;
END;
$$ LANGUAGE plpgsql;

SELECT * FROM recupera_funcionario(1);
```

Funções

Exemplo – cursor

```
CREATE OR REPLACE FUNCTION total_salarios()  
RETURNS numeric AS $$  
DECLARE  
    registro RECORD; resultado numeric;  
BEGIN  
    resultado := 0.00;  
    FOR registro IN SELECT * FROM funcionario LOOP  
        resultado := resultado + registro.salario;  
    END LOOP;  
    RETURN resultado;  
END;  
$$ LANGUAGE plpgsql;  
  
SELECT total_salarios();
```

Funções

- E quando eu preciso retornar apenas um valor de uma consulta?

```
SELECT select_expressions INTO [STRICT] target FROM ...;  
INSERT ... RETURNING expressions INTO [STRICT] target;  
UPDATE ... RETURNING expressions INTO [STRICT] target;  
DELETE ... RETURNING expressions INTO [STRICT] target;
```

Funções

- E se precisar retornar uma tabela?

```
1 CREATE OR REPLACE FUNCTION retornaHospedes()  
2 RETURNS TABLE (  
3     cpf char  
4 ) AS $$  
5 BEGIN  
6     RETURN QUERY  
7     SELECT h.cpf FROM hospedes h WHERE h.datasai = current_date ;  
8  
9 END;  
10 $$ LANGUAGE plpgsql;
```

Funções

- E se precisar retornar uma tabela?

```
1 CREATE OR REPLACE FUNCTION retornaHospedes2()  
2 RETURNS TABLE (  
3     cpf_N char  
4 ) AS $$  
5 DECLARE  
6     tupla RECORD;  
7 BEGIN  
8  
9     FOR tupla IN SELECT h.cpf FROM hospedes h WHERE h.datasai = current_date LOOP  
10         cpf_N := tupla.cpf || ' novo';  
11         RETURN NEXT;  
12     END LOOP;  
13  
14 END;  
15 $$ LANGUAGE plpgsql;
```


Funções

- E quando eu preciso retornar apenas um valor de uma consulta?

```
BEGIN
  SELECT * INTO STRICT myrec FROM emp WHERE empname = myname;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RAISE EXCEPTION 'employee % not found', myname;
  WHEN TOO_MANY_ROWS THEN
    RAISE EXCEPTION 'employee % not unique', myname;
END;
```

Funções

- E se eu quisesse apresentar alguma informação para o usuário?

```
RAISE [ level ] 'format' [, expression [, ... ] ] [ USING option = expression [, ... ] ];  
RAISE [ level ] condition_name [ USING option = expression [, ... ] ];  
RAISE [ level ] SQLSTATE 'sqlstate' [ USING option = expression [, ... ] ];  
RAISE [ level ] USING option = expression [, ... ] ;  
RAISE ;
```

- Opções

DEBUG, LOG, INFO, NOTICE, WARNING, and EXCEPTION

Funções

- E se eu quiser apresentar alguma informação para o usuário?

```
RAISE NOTICE 'Calling cs_create_job(%)', v_job_id;
```

```
RAISE EXCEPTION 'Nonexistent ID --> %', user_id  
    USING HINT = 'Please check your user ID';
```

```
RAISE 'Duplicate user ID: %', user_id USING ERRCODE = 'unique_violation';  
RAISE 'Duplicate user ID: %', user_id USING ERRCODE = '23505';
```

Funções

- E se eu quiser apresentar alguma informação para o usuário?

```
RAISE 'Duplicate user ID: %', user_id USING ERRCODE = 'unique_violation';  
RAISE 'Duplicate user ID: %', user_id USING ERRCODE = '23505';
```

Table A-1. PostgreSQL Error Codes

Error Code	Condition Name
Class 00 — Successful Completion	
00000	successful_completion
Class 01 — Warning	
01000	warning
0100C	dynamic_result_sets_returned
01008	implicit_zero_bit_padding
01003	null_value_eliminated_in_set_function
01007	privilege_not_granted
01006	privilege_not_revoked
01004	string_data_right_truncation

<https://www.postgresql.org/docs/9.3/errcodes-appendix.html>

Funções

- Como controlo as EXCEPTION?

```
BEGIN
  SELECT * INTO STRICT myrec FROM emp WHERE empname = myname;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RAISE EXCEPTION 'employee % not found', myname;
  WHEN TOO_MANY_ROWS THEN
    RAISE EXCEPTION 'employee % not unique', myname;
END;
```

Funções

- Posso criar sub blocos?

```
[ <<rotulo>> ]  
[ DECLARE  
  declarações ]  
BEGIN  
  comandos  
END [ rotulo ];
```

```
RAISE NOTICE 'Quantidade aqui eh %', qtdade; -- Imprime 30  
qtdade := 50;  
--  
-- Cria um subbloco  
--  
DECLARE  
  qtdade integer := 80;  
BEGIN  
  RAISE NOTICE ' Quantidade aqui eh %', qtdade; -- Imprime 80  
  RAISE NOTICE 'Quantidade externa eh  %', blocoexterno.qtdade; -- Imprime 50  
END;  
RAISE NOTICE 'Quantidade aqui eh  %', qtdade; -- Imprime 50  
RETURN qtdade;  
END;  
$$ LANGUAGE plpgsql;
```

Funções

- E os arrays?

```
1 CREATE OR REPLACE FUNCTION recebendoArray( VARIADIC numeros NUMERIC[])
2 RETURNS integer AS $$
3 DECLARE
4     soma int; i int;
5 BEGIN
6     soma :=0;
7     FOREACH i IN ARRAY numeros LOOP
8         soma:= soma+i;
9     END LOOP;
10    RETURN soma;
11 END;
12 $$ LANGUAGE plpgsql;
13
14
15 SELECT recebendoArray(VARIADIC ARRAY [1,2,3]);
```

Funções

- E os arrays?

```
1 CREATE OR REPLACE FUNCTION retornaCPFs()
2 RETURNS TEXT[] AS $$
3 DECLARE
4     resultado TEXT[];
5     i RECORD;
6 BEGIN
7     resultado := ARRAY[]::TEXT[];
8     FOR i IN SELECT cpf FROM clientes LOOP
9         resultado := array_append(resultado, i.cpf::TEXT);
10    END LOOP;
11    RETURN resultado ;
12 END;
13 $$ LANGUAGE plpgsql;
```


Funções

- E os arrays?

1	CREATE OR REPLACE FUNCTION	1	CREATE OR REPLACE FUNCTION
2	retornaCPFs2()	2	retornaCPFs2()
3	RETURNS TEXT[] AS \$\$	3	RETURNS TEXT[] AS \$\$
4	DECLARE	4	DECLARE
5	resultado TEXT[];	5	resultado TEXT[];
6	i RECORD;	6	i RECORD;
7	BEGIN	7	BEGIN
8	resultado := ARRAY[]::TEXT[];	8	resultado := ARRAY[]::TEXT[];
9	FOR i IN SELECT cpf FROM clientes LOOP	9	FOR i IN SELECT cpf FROM clientes LOOP
10	resultado := resultado i.cpf::TEXT;	10	resultado := resultado i.cpf::TEXT;
11	END LOOP;	11	END LOOP;
12	RETURN resultado ;	12	RETURN resultado ;
13	END;	13	END;
14	\$\$ LANGUAGE plpgsql;	14	\$\$ LANGUAGE plpgsql;

Funções

- E os arrays?

1	CREATE OR REPLACE FUNCTION	1	CREATE OR REPLACE FUNCTION
2	retornaCPFs2()	2	retornaCPFs2()
3	RETURNS TEXT	3	RETURNS TEXT[] AS \$\$
4	DECLARE	4	DECLARE
5	resultado	5	resultado TEXT[];
6	i RECORD	6	i RECORD;
7	BEGIN	7	BEGIN
8	resultado :=	8	resultado := ARRAY[]::TEXT[];
9	FOR i IN	9	FOR i IN SELECT cpf FROM clientes LOOP
10	SELECT	10	resultado := resultado i.cpf::TEXT;
11	END LOOP	11	END LOOP;
12	RETURN	12	RETURN resultado ;
13	END;	13	END;
14	\$\$ LANGUAGE	14	\$\$ LANGUAGE plpgsql;

Exercícios (part 1)

- 1) Faça uma função que calcule o fatorial de um número n;
- 2) Uma prática utilizada durante o desenvolvimento de aplicações que interagem com bancos de dados é a de definir procedimentos ou funções responsáveis pela inclusão, alteração e exclusão de registros. Para as tabelas de Habilitacao e Clientes, crie funções que atendam a essas operações, respeitando as seguintes regras:

a) no caso de inclusões, a função deverá retornar a chave primária do novo registro como resultado;

INSERT [] **RETURNING col**

b) no caso de alterações, a chave primária cujo registro deverá ser modificado deverá ser passada como parâmetro (juntamente com os dados a serem modificados no registro). O retorno dessa função deverá ser nulo;

c) no caso de exclusões, a chave primária cujo registro deverá ser removido deverá ser passada como parâmetro. O retorno dessa função deverá ser true se algum registro foi excluído, e false caso contrário.

GET DIAGNOSTICS linhasAfetadas = ROW_COUNT;

Exercícios (part 2)

- 3) Crie uma função 'passaRegua' que deverá fechar a conta do cliente. A função deve receber como parâmetro o CPF do cliente e retornar o valor a ser pago pelo mesmo. Esta função deve alterar a data *fim* da na tabela locação com a data atual. A função deve somar e retornar o valor gasto com a locação (dias locação * valor diária do barco).
- 4) Crie uma nova tabela chamada de tabela teste. A tabela deve possuir dois campos id (serial primary key) e texto (varchar (100)). Crie uma função que irá receber um inteiro como parametro, e esse inteiro corresponderá ao número de registros que você deve gerar para essa tabela.

Para gerar um string aleatória use “**MD5(random()::text)**”

- 5) Faça uma função que deverá fazer a locação de um barco. Assim, essa função deve receber o funcionário, o cpf do cliente, a matricula do barco, a data de inicio da locação e a provavel devolução. Você deve validar se o barco a ser alugado esta disponível e se o cliente possui a habilitação para o veículo. Caso tudo esteja correto a função deve realizar a inserção dos dados e retornar true.