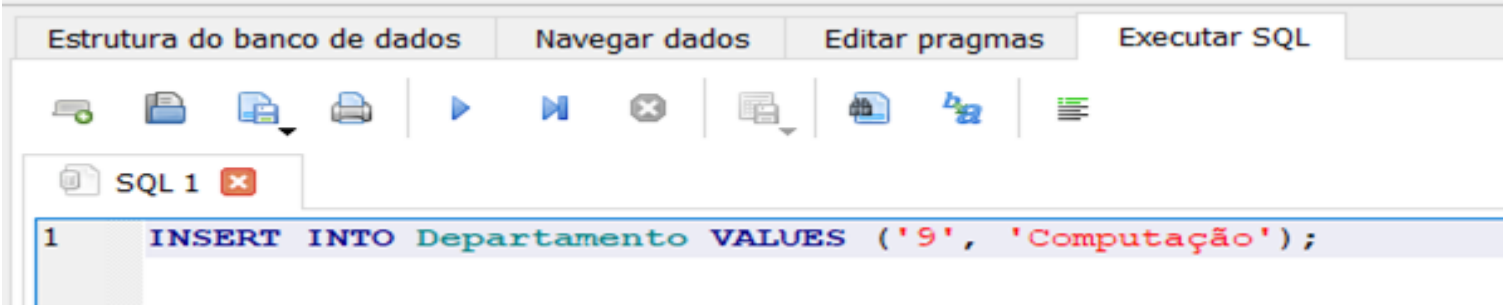
Universidade Federal do Ceará
Centro de Ciências/Departamento de Computação
Código da Disciplina: CK0084
Professor: Ismayle de Sousa Santos

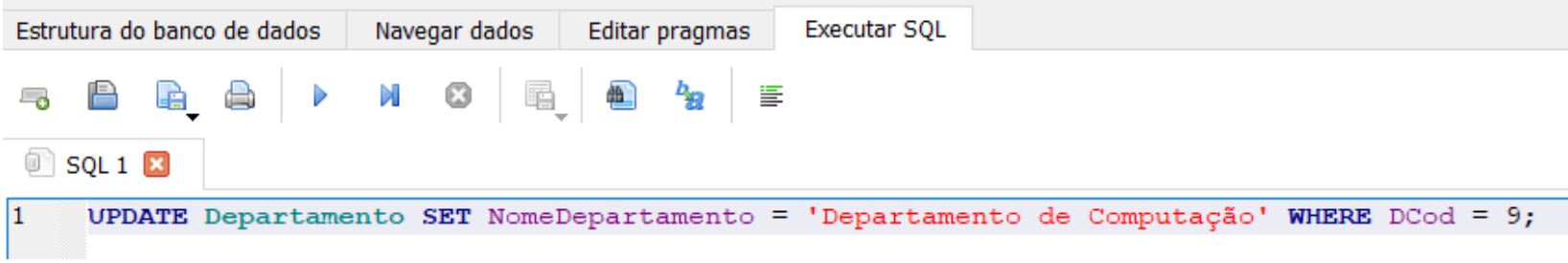
Sistemas de Informações e Banco de Dados

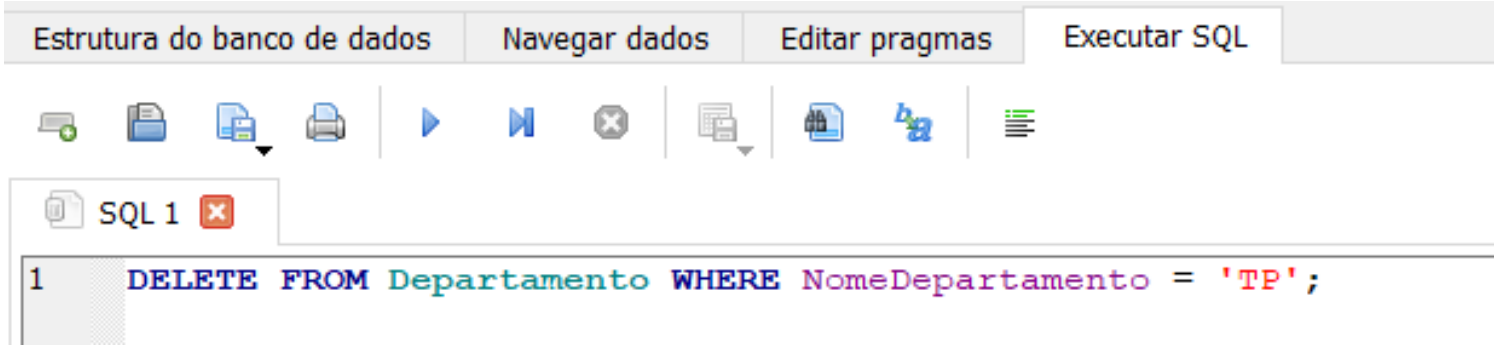
Banco de Dados - SQL (Continuação)

IsmayleSantos
ismayle@ufc.br
@IsmayleSantos

Lembrando...


Estrutura do banco de dados
Navegar dados
Editar pragmas
Executar SQL
SQL 1
1 INSERT INTO Departamento VALUES ('9', 'Computação');


Estrutura do banco de dados
Navegar dados
Editar pragmas
Executar SQL
SQL 1
1 UPDATE Departamento SET NomeDepartamento = 'Departamento de Computação' WHERE DCod = 9;


Estrutura do banco de dados
Navegar dados
Editar pragmas
Executar SQL
SQL 1
1 DELETE FROM Departamento WHERE NomeDepartamento = 'TP';

Integridade Referencial e Dando Nome a Restrições

Integridade Referencial

- Garante que todos os relacionamentos entre as tabelas serão respeitadas resultando em um banco de dados íntegro
  - Relacionado à chave estrangeira
  - Garante que todos os valores atribuídos a chave estrangeira existem como chave primária na tabela relacionada

Integridade Referencial

- Exemplos de cenários onde ela é importante:
  - **Não** é possível excluir o departamento DC (1) sem remover/atualizar os registros da Tabela Professor
  - **Não** é possível adicionar Professor com *CodDepartamnento* = 10 (não existe)

	id	Nome	CodDepartamento ▾1
	Filtro	Filtro	Filtro
1	1	Pedro	1
2	2	Ana	1
3	3	Joana	9

Tabela Professor

DCod	NomeDepartamento
Filtro	Filtro
1	DC
2	CREA
4	SemNome
5	Agricultura
6	SemNome
7	SemNome
8	Exatas
9	Departamento de Computação

Tabela Departamento

Integridade Referencial

- Exemplos

```
DELETE FROM Departamento WHERE DCod = 1;
```

```
Execução finalizada com erros.
Resultado: FOREIGN KEY constraint failed
Na linha 6:
DELETE FROM Departamento WHERE DCod = 1;
```

Integridade Referencial

- Como resolver?
  - Na tabela Professor criada anteriormente:
    - **CREATE TABLE** Empregado (...
      FOREIGN KEY(CodDepartamento)
      REFERENCES Departamento(Dcod)
      ON DELETE SET NULL
      ON UPDATE CASCADE, ...);

Poderia usar **SET DEFAULT** também

- Implicações
  - Na remoção da linha que contém o valor da chave estrangeira deve-se colocar NULL
  - Na alteração do valor da chave estrangeira deve-se alterar em cascata as chaves

Integridade Referencial

- Resultado

```
CREATE TABLE "Professor" (
    "id"    INTEGER,
    "Nome"  TEXT,
    "CodDepartamento"    INTEGER,
    PRIMARY KEY("id"),
    FOREIGN KEY("CodDepartamento") REFERENCES "Departamento"("DCod") ON DELETE SET NULL ON UPDATE CASCADE
);
```

```
UPDATE Departamento SET DCod = 10 WHERE DCod = 1;
```

Nome	CodDepartamento ▾1
Filtro	10
Pedro	10
Ana	10

DCod	NomeDepartamento
Filtro	DC
10	DC

```
DELETE FROM Departamento WHERE DCod = 10;
```

id	Nome	CodDepartamento ▾1
Filtro	Filtro	Filtro
1	Pedro	*NULL*
2	Ana	*NULL*

Consultas de Recuperação em SQL (SELECT)

Consultas SELECT-FROM-WHERE

- Sintaxe: **SELECT** <lista de atributos>
  FROM <lista de tabelas>
  WHERE <condições>;
- Onde:
  - <lista atributos> é uma lista de nomes de atributo cujos valores devem ser recuperados pela consulta
  - <lista tabelas> é uma lista dos nomes de relação exigidos para processar a consulta
  - <condição> é uma expressão condicional (booleana) que identifica as tuplas a serem recuperadas pela consulta

Consultas SELECT-FROM-WHERE

- Exemplos:
  - Recuperar a data de nascimento e o endereço do(s) funcionário(s) cujo nome seja 'João B. Silva'

```
SELECT   Datanasc, Endereco
FROM     FUNCIONARIO
WHERE    Pnome='João' AND Minicial='B' AND
         Unome='Silva';
```

```
SELECT CODIGO, NOME FROM CLIENTES
WHERE UF = 'RJ' OR (UF = 'SP' AND ATIVO = 'N')
```

Cláusula WHERE Não Especificada

- Uma cláusula WHERE faltando significa que não há condição na seleção das tuplas, assim, **todas as tuplas especificadas na cláusula FROM serão selecionadas**
- Exemplo:
  - Selecione os CPFs de todos os professores

```
SELECT  CPF
FROM    PROFESSOR;
```

Uso do Asterisco (*)

- Para recuperar todos os valores de atributo das tuplas selecionadas, não precisamos listar os nomes de atributo explicitamente em SQL
  - Basta especificar um asterisco (*), que significa **todos os atributos**
- Exemplo:
  - Recuperar todos os atributos dos professores que trabalham no DEPARTAMENTO cujo código é 'DECOM'.

```
SELECT   *
FROM     PROFESSOR
WHERE    DCodigo='DECOM'
```

Operadores de Comparação Lógica

- Em SQL, os operadores básicos de comparação lógicos para comparar valores de atributo entre si e com constantes literais são: =, <, <=, >, >= e <> (diferente)
  - Se relacionam aos operadores de álgebra relacional =, <, ≤, >, ≥ e =
  - A principal diferença sintática das demais linguagens é o **operador diferente**

Operadores Aritméticos

- Outro recurso permite o uso de aritmética nas consultas
- Os operadores aritméticos padrão para adição (+), subtração (–), multiplicação (*) e divisão (/) podem ser aplicados a valores ou atributos numéricos com domínios numéricos

Operadores Aritméticos

- Exemplo:
  - Mostrar os salários resultantes se cada funcionário que trabalha no projeto 'ProdutoX' receber um aumento de 10 por cento

```
SELECT  F.Pnome, F.Unome, 1,1 * F.Salario AS
        Aumento_salario
FROM    FUNCIONARIO AS F, TRABALHA_EM
        AS T, PROJETO AS P
WHERE   F.Cpf=T.Fcpf AND T.Pnr=P.Projnumero
        AND P.Projnome='ProdutoX';
```

Operadores Aritméticos

- Exemplo:
  - Recuperar todos os funcionários no departamento 5 cujo salário esteja entre R$ 30.000 e R$ 40.000

```
SELECT  *
FROM    FUNCIONARIO
WHERE   (Salario BETWEEN 30.000 AND
        40.000) AND Dnr = 5;
```

- A condição (Salario BETWEEN 30.000 AND 40.000) é equivalente à condição ((Salario >= 30.000) AND (Salario <= 40.000))

Junção de Tabelas

- A condição Dnumero = Dnr é chamada condição de junção, pois combina duas tuplas: uma de DEPARTAMENTO e uma de FUNCIONARIO, sempre que o valor de Dnumero em DEPARTAMENTO é igual ao valor de Dnr em FUNCIONARIO

```
SELECT  Pnome, Unome, Endereco
FROM    FUNCIONARIO, DEPARTAMENTO
WHERE   Dnome='Pesquisa' AND Dnumero=Dnr;
```

Junção de Tabelas

- O SELECT permite juntar duas ou mais tabelas no mesmo resultado

```
SELECT CLIENTES.CODIGO, CLIENTES.NOME, PEDIDOS.DATA
FROM CLIENTES, PEDIDOS
WHERE CLIENTES.CODIGO = PEDIDOS.CODCLIENTE
```

- Exemplo: Tabelas unificadas em uma mesma cláusula

```
SELECT A.CODIGO, A.NOME, B.DATA, B.VALOR, C.QTD, D.DESCRIC
FROM CLIENTES A, PEDIDOS B, ITENS C, PRODUTOS D
 WHERE A.CODIGO = B.CODCLIENTE
 AND B.CODIGO = C.CODPEDIDO
 AND C.CODPRODUTO = D.CODIGO
```

Nomes de Atributos Ambíguos

- Em SQL, o mesmo nome pode ser usado para dois (ou mais) atributos, desde que estes estejam em relações diferentes
- Se uma consulta em múltiplas tabelas se referir a dois ou mais atributos com o mesmo nome, é preciso qualificar o nome do atributo com o nome da relação, para evitar ambiguidade
  - Isso é feito prefixando o nome da relação ao nome do atributo e separando os dois por um ponto

Nomes de Atributos Ambíguos

- Devemos prefixar os nomes de atributo Nome e Dnumero para especificar a quais estamos nos referindo, pois os mesmos nomes de atributo são usados nas duas relações:

```
SELECT  Pnome, FUNCIONARIO.Nome,
        Endereco
FROM    FUNCIONARIO, DEPARTAMENTO
WHERE   DEPARTAMENTO.Nome='Pesquisa' AND
        DEPARTAMENTO.
        Dnumero=FUNCIONARIO.Dnumero;
```

Apelido, Renomeação e Variáveis de Tupla

- A ambiguidade dos nomes de atributo também surge no caso de consultas que se referem à mesma relação duas vezes
- Para solucionar isso precisamos declarar nomes de relação alternativos F e S, chamados apelidos ou variáveis de tupla, para a relação FUNCIONARIO
  - Um apelido pode vir após a palavra-chave

Apelido, Renomeação e Variáveis de Tupla

- Exemplo:
  - Escrevendo FUNCIONARIO F, FUNCIONARIO S na cláusula FROM

```
SELECT  F.Pnome, F.Unome, S.Pnome, S.Unome
FROM    FUNCIONARIO AS F, FUNCIONARIO
        AS S
WHERE   F.Cpf_supervisor=S.Cpf;
```

  - Também é possível renomear os atributos da relação dentro da consulta em SQL, dando-lhe apelidos
    - FUNCIONARIO AS F(Pn, Mi, Un, Cpf, Dn, End, Sexo, Sal, Scpf, Dnr)

Combinação de Padrão de Subcadeias

- O operador de comparação **LIKE** permitirá condições de comparação apenas sobre partes de uma cadeia de caracteres
  - Usado para combinação de padrão de cadeia
- Cadeias parciais são especificadas usando dois caracteres reservados: % substitui um número qualquer de zero ou mais caracteres, e o sublinhado (_) substitui um único caractere

Combinação de Padrão de Subcadeias

- Exemplo:
  - Recuperar todos os funcionários cujo endereço esteja em São Paulo, SP

```
SELECT    Pnome, Unome
FROM      FUNCIONARIO
WHERE     Endereco
          '%SaoPaulo,SP%';
```

LIKE

Obrigado!

Por hoje é só pessoal...

Dúvidas?


IsmayleSantos


ismayle@.ufc.br


@IsmayleSantos