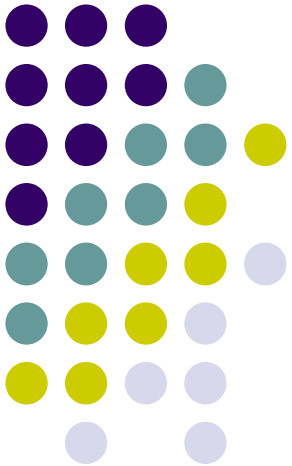
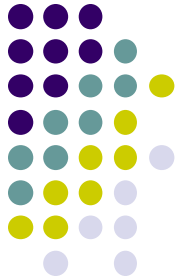
PROCEDIMENTO ARMAZENADOS

Padrão SQL

(adaptado do material da profa Vanessa)



Procedimento Armazenado

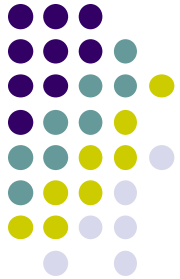


- Um procedimento armazenado (*stored procedure*) é um bloco de programa com alguma funcionalidade específica, armazenado e executado diretamente pelo SGBD
  - No padrão SQL, são denominados módulos armazenados de modo persistente
  - Foram incorporados a partir da versão SQL:1999
  - Podem retornar ou não um valor

Procedimento Armazenado

- Procedimentos armazenados são úteis nos seguintes casos:
  - Dado um módulo de um programa (módulo = funcionalidade) que manipula uma base de dados e que será incorporado a mais de uma aplicação
    - É interessante a definição das funcionalidades do módulo como um procedimento armazenado
    - Quando for necessário, cada aplicação apenas aciona o procedimento armazenado, que será executado pelo SGBD
    - Desta forma, reduz-se a duplicação de esforços
      - Qualquer manutenção será focada no procedimento no SGBD

Procedimento Armazenado

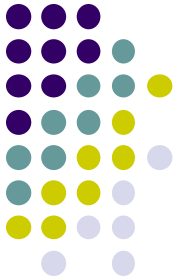


- Procedimentos armazenados são úteis nos seguintes casos:
  - Para reduzir a transferência de dados
    - A execução de procedimentos no SGBD pode reduzir, em certos casos, os custos de comunicação
  - Para verificar restrições mais complexas de uma dada aplicação
    - Desta forma, o tratamento de restrições fica transparente para o usuário (programador)

Procedimento Armazenado

- Procedimentos armazenados são úteis nos seguintes casos:
  - Para promover maior rapidez na execução
    - *Procedimentos* são armazenados em memória cache, pré-compilados e permanecem aguardando invocação
  - Para promover segurança dos dados
    - Segurança a nível de procedimento (públicos ou proprietários; criptografados ou não)

Procedimento Armazenado

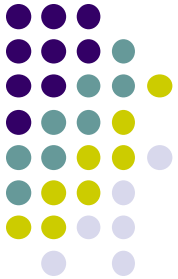


- Procedimento - Sintaxe SQL
  - Procedimentos armazenados que não retornam valor

```
CREATE PROCEDURE <nome_procedimento>
(<parâmetros>)
BEGIN
<declarações_locais>
<corpo_procedimento>
END
```

  - Onde:
    - <parâmetros>: conjunto de parâmetros a serem passados para execução do procedimento (opcional);
    - <declarações_locais>: declaração de variáveis locais, utilizadas na implementação do procedimento (opcional);
    - <corpo_procedimento>: implementação do procedimento

Procedimento Armazenado



- Função – Sintaxe SQL
  - Procedimentos armazenados que retornam valor

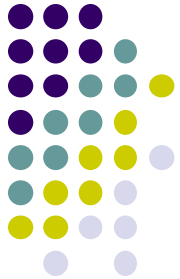
```
CREATE FUNCTION <nome_funcao> (<parâmetros>)
RETURNS <tipo_retorno>
BEGIN
<declarações_locais>
<corpo_funcao>
END
```

  - Onde:
    - <parâmetros>, <declarações_locais> e <corpo_função> são como nos procedimentos;
    - <tipo_retorno>: indica o tipo de retorno da função;

Procedimento Armazenado

- Com relação à definição dos parâmetros e variáveis locais:
  - Parâmetros e variáveis podem ser de um determinado tipo definido na SQL, ou de algum outro tipo criado pelo usuário (como o struct + typedef em C)
  - Parâmetros e variáveis podem ser:
    - Somente de entrada (*IN*), no caso de parâmetros apenas recebidos e utilizados para tomada de decisões de implementação;
    - Somente de saída, no caso de serem variáveis definidas para armazenamento de valores de resultado (*OUT*)
    - Ou de entrada e saída (*INOUT*), no caso de parâmetros recebidos, manipulados e armazenados como resultado;
    - Quando nada é informado, o parâmetro é IN por padrão;

professor(matricula,nome,titulacao,coddept)
disciplina(codigo,nome,descricao)
turma(matriculap, codigod,semestre)

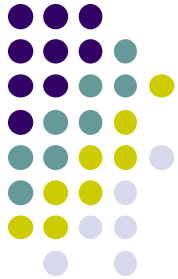


Stored Procedures

- Exemplo
  - Dado a matrícula do professor, calcule a quantidade de turmas do mesmo;

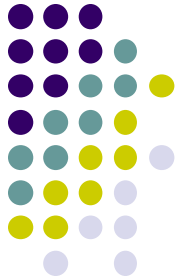
```
CREATE PROCEDURE proc_contagem_turma (IN mat_prof
integer, OUT contagem_t integer)
BEGIN
  SELECT COUNT(*) INTO contagem_t
  FROM turma
  WHERE turma.matriculap =
  proc_contagem_turma.mat_prof
END
```

Stored Procedures



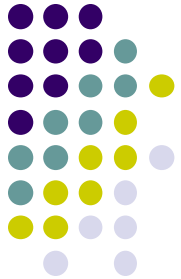
- Um procedimento armazenado pode ser invocado em programas tradicionais ou em um ambiente de consulta do SGBD
- Em determinados SGBD, podemos executar um procedimento utilizando a seguinte sintaxe SQL:
  - CALL <nome_procedimento> (lista_de_argumentos);
    - Para funções não é permitido o uso de CALL;
- No corpo de um programa, a estratégia usada para a chamada (execução) do procedimento (ou função) depende do ambiente de desenvolvimento (linguagem de programação, drive de conexão utilizado e camadas de persistência)

Stored Procedures



- **Declarações**
  - **Em alguns SGBD,** caso seja necessário, pode-se declarar variáveis locais em um ambiente de consulta
  - Sintaxe:
    - DECLARE nome_variável tipo_dado <DEFAULT valor>;
      - Declare @idade int
      - Set @idade = 20
      - Select * from cliente where idade > @idade
    - DEFAULT – valor *default* para a variável. Caso não seja determinado, esse valor será NULL.

Stored Procedures

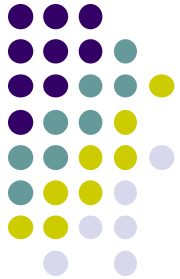


- Suponha que queremos saber a quantidade de turmas do professor cuja matrícula é 11100 em ambiente de consulta

  - DECLARE contagem_t integer;
    CALL proc_contagem_turma (11100, contagem_t)

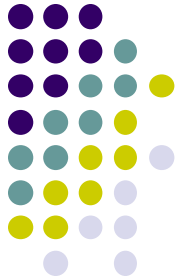
  - Depois da execução do procedimento, a variável “contagem_t” conterá o valor de resultado (OUT)
    - Este valor poderá agora ser utilizado em outras consultas ou procedimentos

Stored Procedures



- Exemplo de função:
  - Suponha que queiramos reescrever o procedimento anterior como uma função (dado a matrícula, retorne a quantidade ede turmas do mesmo)
  - ```
    CREATE FUNCTION contagem_turma (mat_prof integer)
    RETURNS integer
    BEGIN
    DECLARE contagem_t integer;
      SELECT COUNT(*) INTO contagem_t
      FROM turma
      WHERE turma.matriculap = contagem_turma.mat_prof
    RETURN contagem_t;
    END
    ```
    ```

Stored Procedures



- Uma função pode ser usada em uma consulta
  - Suponha que você queira saber quais os nomes e a titulação de todos os professores com mais de uma turma

```
SELECT nome, titulacao FROM professor
WHERE contagem_turma (matriculap) > 1
```

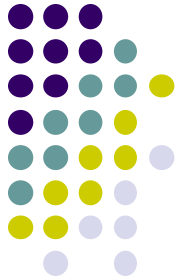
Stored Procedures

- Desde a versão SQL:2003 é permitido que as funções retornem não só valores unitários, mas tabelas (funções de tabela)
- A função abaixo retorna uma tabela com a disciplina e o semestre de todas as turmas que um professor possui

```
CREATE FUNCTION turmas_de (mat_prof integer)
RETURNS table (disciplina integer, semestre int))
RETURN TABLE
(SELECT codigod, semestre
FROM turma
WHERE matriculap = mat_prof)
```

Definição
da tabela retornada

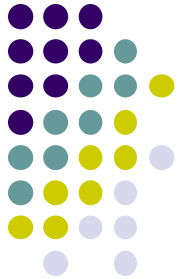
Stored Procedures



- A função anterior pode ser utilizada em uma consulta da seguinte maneira

  SELECT * FROM TABLE (turmas_de(11100))
- Essa consulta retorna a disciplina e o semestre de todas as turmas pertencentes ao professor 11100
- Este é um exemplo que pode ser facilmente escrito sem uma função
- Porém, na maioria das vezes, as funções com valor de tabela podem ser vistas como *views* especializadas (parametrizadas)
  - Especializada devido à possibilidade de usar parâmetros

Stored Procedures

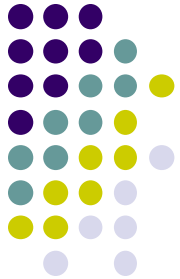


- Procedimentos/funções armazenados também podem ser definidos utilizando uma linguagem de programação
- Corpo do procedimento é definido em um dado arquivo, utilizando a sintaxe daquela linguagem externa
- Sintaxe SQL

  CREATE PROCEDURE <nome_procedimento>
  (<parâmetros>)
  LANGUAGE <nome_linguagem>
  EXTERNAL_NAME <nome_caminho_arquivo>

  - Onde:
    - <nome_linguagem>: nome da linguagem em que foi implementado o procedimento/função;
    - <nome_caminho_arquivo>: caminho para o arquivo onde está implementado o procedimento/função;

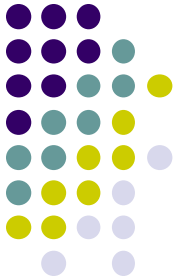
Stored Procedures



- Funções definidas em outras linguagens podem ser mais eficientes e oferecer mais possibilidades que SQL

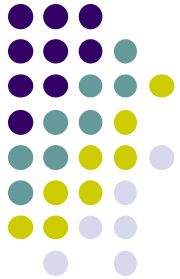
```
CREATE PROCEDURE proc_contagem_t (IN
mat_prof integer), OUT contagem integer)
LANGUAGE C
EXTERNAL NAME ‘/usr/avi/bin/proc_contagem_t/’
```

Stored Procedures



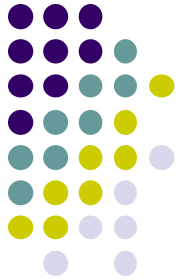
- Estruturas de controle procedurais
  - São construções que fornecem quase toda a capacidade de uma linguagem de programação à linguagem SQL
- Desde a versão SQL:1999, é permitido a definição de diversas construções procedurais (estrutura de controle)
- A parte do padrão SQL que lida com tais construções é chamado PSM (*Persistent Storage Module*);

Stored Procedures



- A idéia não é que o PSM substitua uma linguagem de programação!
- Geralmente são utilizados na definição de regras de negócio de um certo domínio;
  - Ex: bancos possuem várias regras sobre como e quando um pagamento pode ser feito;
  - Tais regras são geralmente utilizadas em qualquer aplicação que lida com contas bancárias;
  - Neste caso, definir tais regras no banco e não na aplicação possui vantagens significativas:
    - Várias aplicações podem acessar o mesmo procedimento;
    - Um único ponto para mudanças;

Stored Procedures

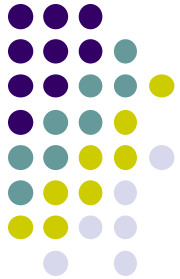


- PSM define um conjunto de instruções:
  - Iterativas:

```
WHILE <condicao> DO
    <lista de instruções>
END WHILE;

REPEAT
    <lista de instruções>
UNTIL <condicao>
END REPEAT;
```

A maioria dos SGBDs relacionais não são totalmente compatíveis com o padrão PSM



Stored Procedures

- PSM define um conjunto de instruções:
  - Condicionais:

    IF <condicao> THEN <lista de instruções>

    ELSEIF <condicao> THEN <lista de instruções>

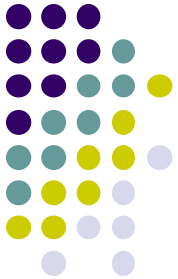
    ...

    ELSEIF <condicao> THEN <lista de instruções>

    ELSE <lista de instruções>

    END IF;

Stored Procedures

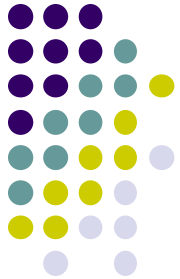


- Criar um procedimento para retornar os dados dos professores por matrícula (op=1) ou por nome (op=2) ou por titulação (op=3))

```
CREATE PROCEDURE select_professor (IN op int, IN mat intr, IN
    nm varchar(100), IN tit varchar(20))
BEGIN
    IF (op=1) THEN select * from professor where matricula = mat
    ELSEIF (op=2) THEN select * from professor where nome = nm
    ELSEIF (op=3) THEN select * from professor where titulacao =
    tit
END;
```

- CALL select_professor(2, NULL, ‘MARIA’,NULL);

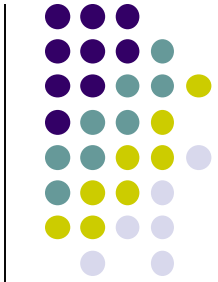
Stored Procedures



- Classifique um departamento, de acordo com seu número de empregados

```
CREATE FUNCTION dep_tamanho (IN dnum integer)
RETURNS VARCHAR(7)
DECLARE numero_emps integer
SELECT COUNT(*) INTO numero_emps
FROM empregado WHERE dnumero = dnum;
IF numero_emps > 100 THEN RETURN “grande”
ELSE IF numero_emps > 10 AND numero_emps < 100
                THEN RETURN “medio”
          ELSE RETURN “pequeno”
END IF
```

professor(matricula,nome,titulacao,coddept,salario)
disciplina(codigo,nome,descricao)
turma(matricula, codigo,semestre)



```
CREATE PROCEDURE altera_salario (IN mat intr)
DECLARE p professor;
BEGIN
    FOR p AS
        SELECT * FROM PROFESSOR
        WHERE matricula = mat;
        IF p.titulacao = 'DSC' THEN
                Update professor set p.salario = p.salario *
        1,10 where p.matricula = mat;
        ELSEIF p.titulacao = 'MSC' THEN
                Update professor set p.salario = p.salario *
        1,20 where p.matricula = mat;
        END IF
     END FOR;
END;
```