

Stored Procedures

Conceitos e aplicações em MySql



Nasair Júnior da Silva
njunior@solis.coop.br

Índice

Objetivos do curso

O que é mysql

O que é uma Stored Procedure

Características do mysql

Histórico do mysql

Instalação

Ferramentas de administração

Stored procedures

Funções internas

Criando procedures

Criando funções

Objetivos do curso?

- O objetivo desse minicurso é debater questões como desenvolvimento e utilização de bibliotecas de funções em banco de dados.
- Entre os tópicos abordados estão:
 - Funções SQL
 - Funções Internas
 - Particularidades do Mysql
 - Problemas e instabilidades

O que é Mysql?

é um servidor de bancos de dados SQL (Structured Query Language - Linguagem Estruturada para Pesquisas) muito rápido, multi-tarefa e multi-usuário

O que é uma Stored Procedure?

- Uma stored procedure (SP) é um conjunto de comandos SQL que podem ser armazenados no servidor, de forma a permitir um aumento no desempenho das aplicações visto que há uma redução no volume de informações entre o servidor e o cliente(aplicação).

Características

- Multi-plataforma
- Utiliza padrão ANSI/ISO SQL
- Open Source (GPL ou Comercial License)
- Rápido, confiável, e fácil de usar
- Mecanismos de armazenamento transacional e não transacional
- Suporte a operadores e funções

Características(2)

- Trabalha com bancos de dados enormes (mais de 60.000 tabelas e 5 milhões de registros)
- Até 32 índices por tabela (composto de 1 a 16 colunas)
- Mensagens em vários idiomas
- Escolha de codificação para gravar dados
- API's em várias linguagens (p.e. C, C++, Eiffel, Java, Perl, PHP, Python, Ruby e Tcl)

Características(3)

- Outras funcionalidades:
 - Gatilhos (triggers)
 - **Stored procedures**
 - Visões (views)
 - Joins otimizados
 - Sistema de privilégios
 - Replicação (inclusive SSL)
 - Pesquisas Full-Text

Histórico

- 1979: Unireg: Monty Wideliious criou o Unireg, um banco de dados não-SQL para grandes tabelas
- 1994: Monty iniciou o desenvolvimento de um SGBD baseado no Unireg; API baseado no mSQL, um gerenciado open source (não era muito bom para grandes tabelas)
- 1996: Mysql 3.11.1 lançado em binário para Linux e Solaris

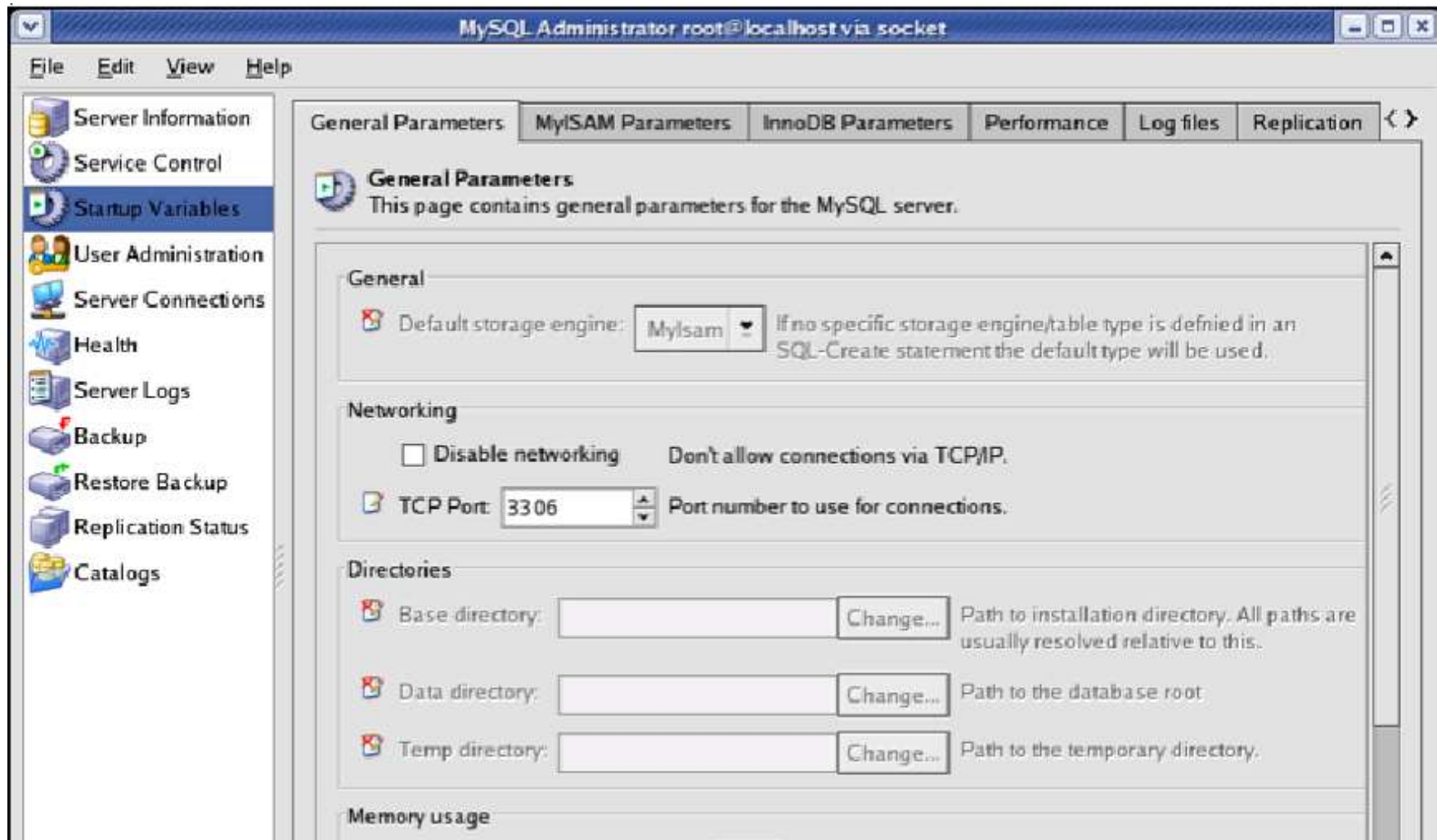
Instalação

- no Linux
 - Download de:
 - www.mysql.com/download
 - Fácil de compilar: `./configure && make && make install`
 - Ou utilizar os pacotes prontos RPM DEB TGZ
- No windows
 - Basta baixar binário do site

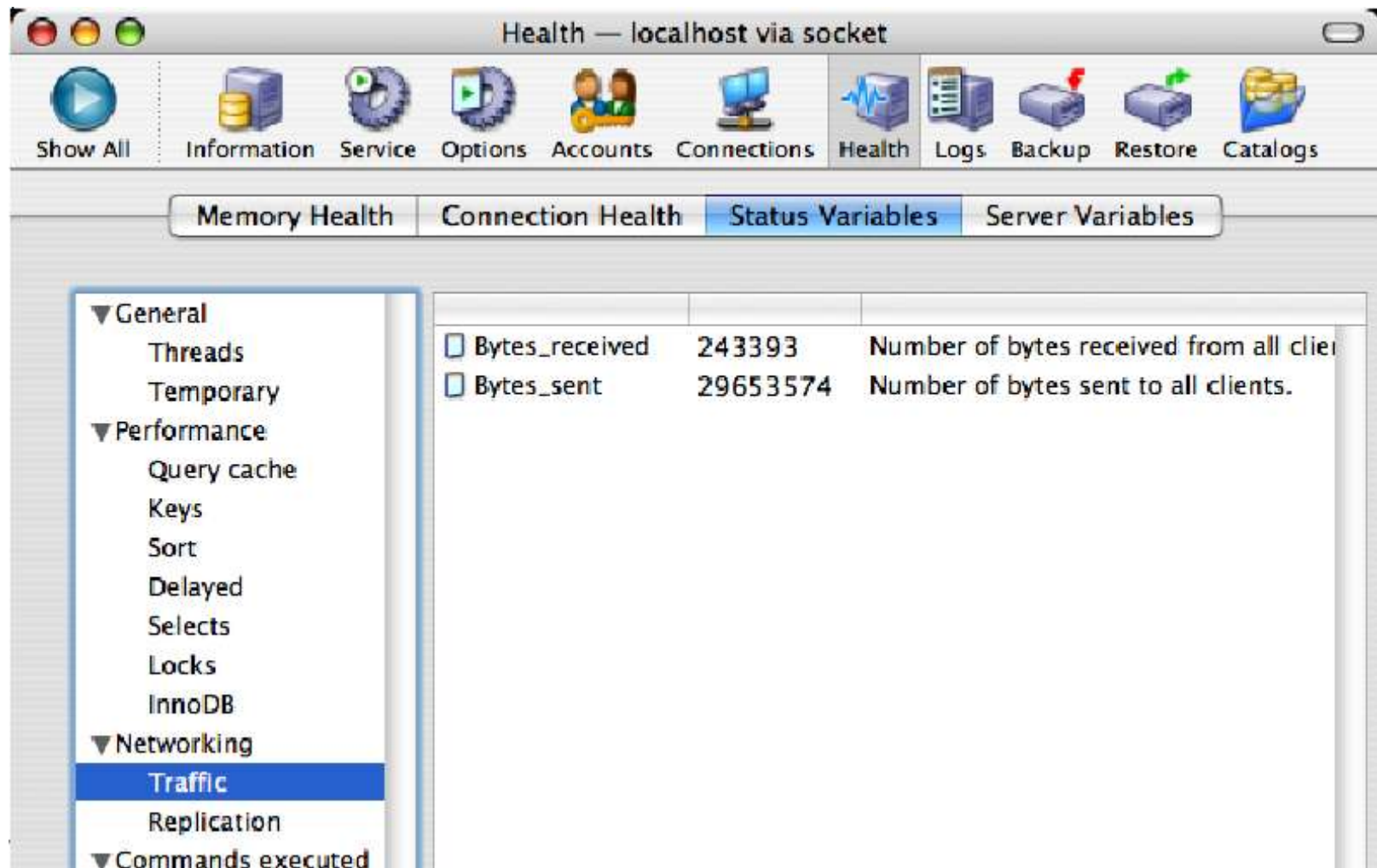
Ferramentas de Administração

- MySQL Administrator
 - <http://www.mysql.com/products/administrator/>
 - Aplicação gráfica para administração do MySQL
 - Administração User-Friendly
 - Ferramenta para backup/restauração
 - Replicação: topologia e visualizador de estado
 - Logs do servidor

MySql Administrator



MySql Administrator(2)



The screenshot shows the MySQL Administrator window titled "Health — localhost via socket". The interface includes a toolbar with icons for "Show All", "Information", "Service", "Options", "Accounts", "Connections", "Health", "Logs", "Backup", "Restore", and "Catalogs". Below the toolbar are tabs for "Memory Health", "Connection Health", "Status Variables", and "Server Variables". The "Status Variables" tab is active, displaying a tree view on the left and a table of variables on the right.

Left Panel (Tree View):

- ▼ General
 - Threads
 - Temporary
- ▼ Performance
 - Query cache
 - Keys
 - Sort
 - Delayed
 - Selects
 - Locks
 - InnoDB
- ▼ Networking
 - Traffic**
 - Replication
- ▼ Commands executed

Right Panel (Table):

<input type="checkbox"/> Bytes_received	243393	Number of bytes received from all clients.
<input type="checkbox"/> Bytes_sent	29653574	Number of bytes sent to all clients.

MySql Query Browser

- <http://www.mysql.com/products/query-browser/>
- Ferramenta para consultas
- Interface similar à um navegador
- Manipulação de resultados em tabs
- Histórico de consultas
- Bookmarks
- Edição e comparação de resultados
- Debug de SQL

MySql Query Browser(2)

The screenshot shows the MySQL Query Browser interface. At the top, the query is: `SELECT * FROM phpbb_topics p WHERE p.forum_id=7`. The main window displays a table with columns: forum_id, cat_id, forum_name, forum_desc, forum_status, forum_order, forum_posts, forum_topics, forum_last_post_time, and prune_time. The data is filtered for forum_id = 7.

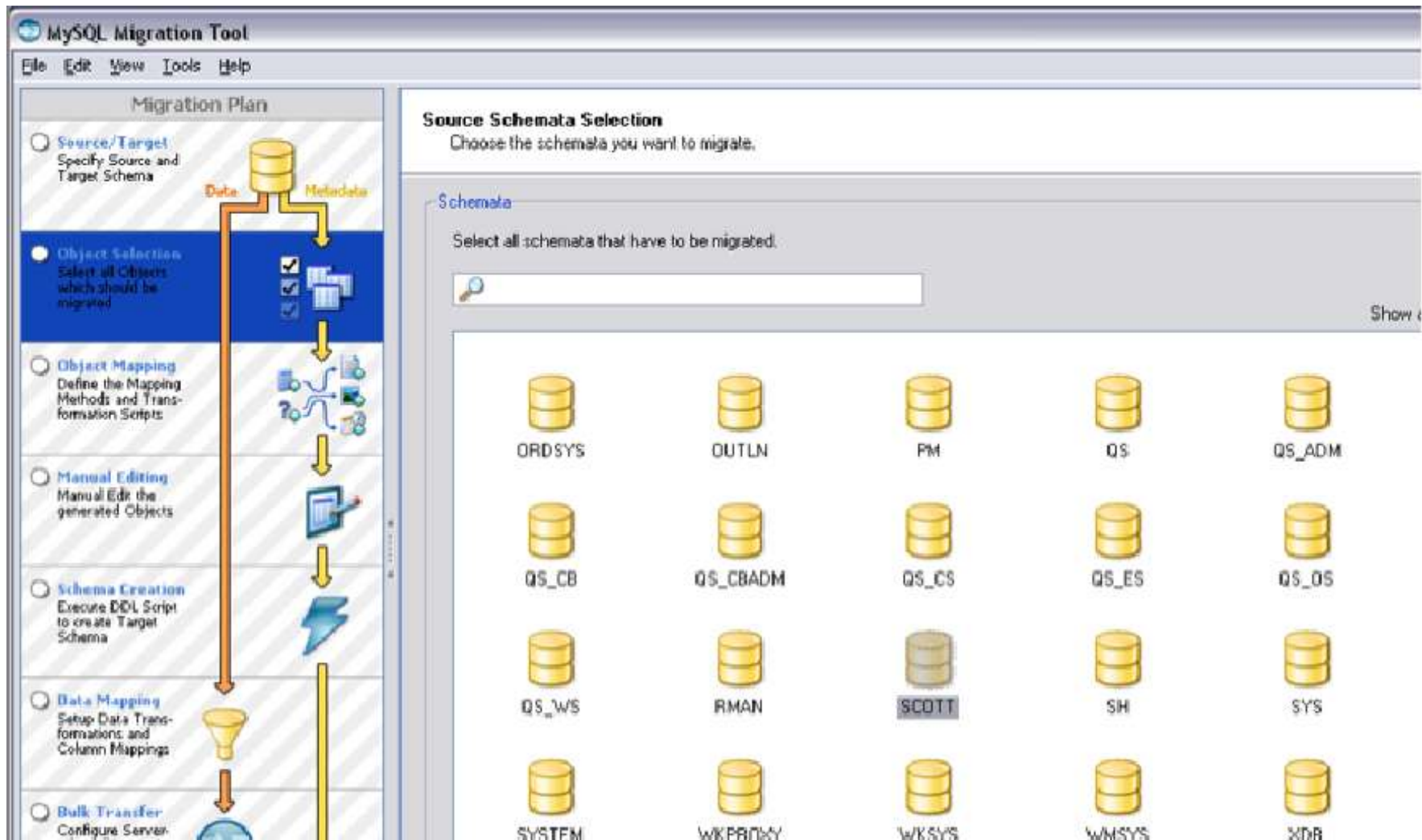
forum_id	cat_id	forum_name	forum_desc	forum_status	forum_order	forum_posts	forum_topics	forum_last_post_time	prune_time
4	2	News					9	2417	
2	2	Bug discussion				325	2521		
3	2	Feature requests				178	2515		
5	3	SimpleWebFront	Everyt...	0	10	205	56	2478	
6	3	HTML Reporter	Infos, ...	0	20	6	3	1729	
7	3	DataImporter	The Da...	0	30	19	9	2493	
8	2	fabFORCE Team Forum	Intern...	0	80	9	2	54	
9	2	Tips & Tricks and HowTos	The be...	0	50	567	173	2522	
10	2	Other Designers / Quer...	Post in ...	0	70	25	6	2514	
11	2	Release Infos	This fo...	0	20	49	2	1901	
12	2	Developers exchange	Various...	0	60	139	43	2497	

A detailed view of a topic is shown below, with columns: topic_id, forum_id, topic_title, topic_poster, topic_time, topic_views, topic_replies, topic_status, topic_vote, and topic_type.

topic_id	forum_id	topic_title	topic_poster	topic_time	topic_views	topic_replies	topic_status	topic_vote	topic_type
41	7	How to create a NewO...	9	1050613282	707	5	0	0	0
302	7	import.xml from DBdesig...	58	1060081487	312	2	0	0	0
523	7	Access to MySQL conve...	416	1068483231	273	1	0	0	0
570	7	von MySQL zu SQLite	461	1070123351	124	1	0	0	0
642	7	import access relations ...	539	1073320482	74	0	0	0	0
688	7	Import Access in Firebird	577	1074778252	117	0	0	0	0
702	7	Fatal Error: xmldom.dou	598	1075323833	39	0	0	0	0
775	7	DataImport Error	684	1077808944	80	1	0	0	0

The interface also shows a 'Parameters' window with a list of fields: forum_id, cat_id, forum_name, forum_desc, forum_status, forum_order, forum_posts, forum_topics. The 'Schemata' panel on the right shows the database structure for phpbb_topics, including fields like topic_id, forum_id, topic_title, etc. The 'Params' panel at the bottom right shows the current query parameters: forum_id (7) and cat_id (3).

MySql Query Browser(3)



Mais algumas possibilidades

- MyAdmin – Interface para consultas
- phpMyAdmin – Interface WEB em PHP
- MSAccess, ERWin – via ODBC
- Quase 500 aplicativos
 - <http://solutions.mysql.com/software/>
- ...

Stored Procedures

Stored Procedures

- um dos novos recursos no MySQL 5.0
- ainda em versão alfa
- conjunto de comandos SQL que podem ser armazenados no servidor
- aumento no desempenho: menos informação enviada entre cliente/servidor
- mais trabalho para o servidor

Bons motivos para utilizar

- Clientes em diferentes linguagens
- Operações repetitivas
- Segurança

Antes de Iniciar

- Verificar se o banco de dados está rodando

```
# ps ax |grep mysql
```
- Iniciar o cliente mysql
 - `mysql -uUsuario -pSenha`

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 4 to server version: 5.0.4-beta-standard

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>

- Criar um novo banco de dados

```
mysql> create database teste;
```

Antes de iniciar (2)

- Utilizar o banco de dados criado

```
mysql> use teste;
```

```
mysql> DELIMITER //
```

- Verificar a versão do banco (um exemplo de função) ≥ 5.0

```
mysql> select version();//
```

```
| version() |
```

```
+-----+
```

```
| 5.0.4-beta-standard |
```

```
+-----+
```

```
1 row in set (0.00 sec)
```

Iniciando...

- Criar tabelas pessoas e empresas

```
mysql> CREATE TABLE empresas (  
    codigo integer auto_increment,  
    nome varchar(50),  
    primary key(codigo)) engine=innodb; //  
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> CREATE TABLE pessoas (  
    codigo integer auto_increment,  
    nome varchar(50),  
    empresa integer, salario real,  
    primary key(codigo),  
    foreign key(empresa) references empresas(codigo))  
    engine=innodb; //
```

```
Query OK, 0 rows affected (0.29 sec)
```

Iniciando...(2)

- Inserir dados (empresas)

```
mysql> INSERT INTO empresas ( nome) VALUES ('Solis'),  
      ('Unisinos'), ('Univates'), ('Governo FEDERAL');//
```

Query OK, 4 rows affected (0.05 sec)

Records: 4 Duplicates: 0 Warnings: 0

- Inserir dados (pessoas)

```
mysql> INSERT INTO pessoas (nome, empresa, salario)  
VALUES ('Nasair Silva',1,100), ('Joice',1,200),  
      ('Daniel',1,300), ('Leonardo',2,400),  
      ('Andréa', '2',500), ('Cesar',3,600), ('Viviane',3,700),  
      ('Roberto Jefferson',4,3000), ('Luis Inácio',4,5000);//
```

Query OK, 9 rows affected (0.00 sec)

Records: 9 Duplicates: 0 Warnings: 0

Procedimentos e funções

- Procedimentos
 - chamado usando a instrução CALL
 - valores de retorno somente com variáveis de saída
- Funções
 - podem retornar valor escalar
 - pode ser utilizado da mesma forma que função interna

Funções básicas

- AVG (Média)
 - `SELECT avg(salario) FROM pessoas; //`
 - `SELECT empresa, avg(salario) FROM pessoas GROUP BY empresa; //`
- SUM (Soma)
 - `SELECT sum(salario) FROM pessoas; //`
 - `SELECT empresa, sum(salario) FROM pessoas GROUP BY empresa; //`

Funções básicas(2)

- COUNT (Contagem)
 - SELECT empresa, count(salario) FROM pessoas GROUP BY empresa;//
- MIN (Mínimo)
- MAX (Máximo)
 - SELECT empresa, min(salario), max(salario) FROM pessoas GROUP BY empresa;//
- SELECT empresa, count(salario) AS NumFuncionarios, avg(salario) AS MediaSalarial, sum(salario) AS TotalFolha FROM pessoas GROUP BY empresa;//

Funções Strings

- ASCII('A')
- BIN(N)
- BIT_LENGTH(str)
- CHAR(N1,N2,N3,...)
- CONCAT(str1,str2,...)
- CONCAT_WS(separador, str1, str2,...)
- LCASE(str) , LOWER(str)
- UCASE(str), UPPER(str)
- LTRIM(str), RTRIM(str), TRIM(str)

Funções Strings(2)

- LENGTH(str), OCTET_LENGTH(str), CHAR_LENGTH(str), CHARACTER_LENGTH(str)
- LOCATE(substr, str)
- QUOTE(str)
- REPEAT(str, cont)
- REPLACE(str, from_str, to_str)
- REVERSE(str)
- SUBSTRING(str, pos, tam)

Funções matemáticas

- $\text{ABS}(X)$
- $\text{SIGN}(X)$
- $\text{MOD}(N,M)$
- $\text{ROUND}(X,D)$
- $A \text{ DIV } B$
- $\text{EXP}(X)$
- $\text{LN}(X)$
- $\text{POWER}(X,Y)$
- $\text{SQRT}(X)$
- $\text{PI}()$

Funções matemáticas(2)

- $\text{COS}(X)$, $\text{ACOS}(X)$
- $\text{SIN}(X)$, $\text{ASIN}(X)$
- $\text{TAN}(X)$, $\text{ATAN}(X)$, $\text{COT}(X)$
- $\text{RAND}()$
- $\text{LEAST}(X, Y, \dots)$
- $\text{GREATEST}(X, Y, \dots)$
- $\text{TRUNCATE}(X, D)$
- $\text{VARIANCE}(\text{expr})$
- $\text{STD}(\text{expr})$, $\text{STDDEV}(\text{expr})$

Funções data

- DATE(data)
- TIME(data)
- TIMESTAMP(expr)
- DAYOFWEEK(data) //1=Segunda
- WEEKDAY(data) //0=Segunda
- DAYOFMONTH(data), DAY(data)
- DAYOFYEAR(data)
- MONTH(data)
- CURTIME(), CURDATE(), CURRENT_TIMESTAMP

Funções data(2)

- DATEDIFF(expr,expr2) , TIMEDIFF(expr,expr2)
- DATE_FORMAT(data,formato) ,
TIME_FORMAT(hora,formato)
 - %d, %m (%c), %Y (%y), %H, %i, %s, %w
- TO_DAYS(data)
- EXTRACT(tipo FROM data)
 - YEAR, MONTH, YEAR, HOUR, MINUTE, SECOND
- DATE_ADD(data, INTERVAL tipo expr),
DATE_SUB(date, INTERVAL tipo expr)

Outras funções

- CAST, CONVERT
 - BINARY, CHAR, DATE, DATETIME, SIGNED, TIME, UNSIGNED
 - SELECT cast('14:08' AS time);//
 - SELECT convert('áéíóú' USING utf8);//
- DATABASE()
- USER()
- PASSWORD(str), MD5(str), SHA1(str), ENCODE(str, senha), DECODE(str, senha)
- VERSION()

Criando procedures

```
mysql> CREATE PROCEDURE pessoas () SELECT * FROM pessoas; //
```

```
Query OK, 0 rows affected (0.06 sec)
```

```
mysql> call pessoas (); //
```

```
+-----+-----+-----+
| codigo | nome           | empresa |
+-----+-----+-----+
|      1 | Nasair Silva   |      1 |
|    .. |      ....     |    ... |
|      9 | Luis Inácio    |      4 |
+-----+-----+-----+
```

```
9 rows in set (0.00 sec)
```

```
Query OK, 0 rows affected (0.00 sec)
```

Criando procedures(2)

- Criar a procedure
 - CREATE PROCEDURE pessoas ()...
 - espaço para não “confundir” com outros objetos
- Chamar a procedure
 - CALL pessoas ()
- Crie uma procedure para listar todas as empresas

Procedures

- Podem alterar dados:

```
mysql> create procedure apagaPessoas () delete from  
    pessoas; //
```

```
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> call apagaPessoas (); //
```

```
Query OK, 9 rows affected (0.00 sec)
```

```
mysql> call pessoas (); //
```

```
Empty set (0.00 sec)
```

Procedures(2)

```
mysql> create procedure inserePessoas () INSERT INTO pessoas (nome,
  empresa, salario) VALUES ('Nasair Silva',1,100),('Joice',1,200),
  ('Daniel',1,300),('Leonardo',2,400),('Andréa','2',500),
  ('Cesar',3,600),('Viviane',3,700),('Roberto Jefferson',4,3000),('Luis
  Inácio',4,5000);//
```

```
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> call inserePessoas ();//
```

```
Query OK, 9 rows affected (0.00 sec)
```

```
mysql> select count(*) from pessoas;//
```

```
| count(*) |
+-----+
|      9 |
+-----+
```

```
1 row in set (0.00 sec)
```

Procedures(3)

```
CREATE PROCEDURE p1 (  
    LANGUAGE SQL  
    [NOT] DETERMINISTIC  
    SQL SECURITY [DEFINER | INVOKER]  
    COMMENT 'este é um procedimento'  
    SELECT CURRENT_DATE;
```

- Um procedimento (ou função) sempre preserva as variáveis de ambiente (escopo)

Exercício

- Crie um procedimento que exiba a data e hora (timestamp) atual

```
CREATE PROCEDURE dataEhora() SELECT  
CURRENT_TIMESTAMP;//
```


Parâmetros

- Parêntes após nome do procedimento delimitam os parâmetros
 - CREATE PROCEDURE p2 ([IN|OUT|INOUT] nome_do_parâmetro tipo_do_parâmetro) ...
- Parâmetros de entrada: IN
- Parâmetros de saída: OUT
- Parâmetros entrada e saída: INOUT

Parâmetros de entrada

```
mysql> select @x; //
```

```
| @x |
```

```
| NULL |
```

```
1 row in set (0.00 sec)
```

```
mysql> create procedure setaX (in aux int) set @x=aux; //
```

```
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> call setaX (4); //
```

```
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> select @x; //
```

```
| @x |
```

```
| 4 |
```

```
1 row in set (0.00 sec)
```

Parâmetros de saída

```
mysql> create procedure recebeX (out aux int) set  
aux=5; //
```

```
Query OK, 0 rows affected (0.08 sec)
```

```
mysql> call recebeX(@x); //
```

```
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> select @x; //
```

```
| @x |
```

```
+-----+
```

```
| 5 |
```

```
+-----+
```

```
1 row in set (0.00 sec)
```

Blocos de funções

```
mysql> create procedure p7 ()  
begin  
    set @a = 5;  
    set @b = 5;  
    insert into t values (@a);  
    select s1 * @a FROM t WHERE s1 >= @b;  
end; //
```

Query OK, 0 rows affected (0.00 sec)

```
mysql> call p7 (); //
```

```
| s1 * @a |  
|      25 |
```

1 row in set (0.00 sec)

Variáveis

```
CREATE PROCEDURE p8 ()  
  
BEGIN  
  
    DECLARE a INT;  
  
    DECLARE b INT;  
  
    SET a = 5;  
  
    set b = 5;  
  
    INSERT INTO t VALUES (a);  
  
    SELECT s1 * a FROM t WHERE s1 >= b;  
  
END; //
```

Variáveis e valor padrão

```
mysql> CREATE PROCEDURE p10 ()  
BEGIN  
    DECLARE a, b INT DEFAULT 5;  
    INSERT INTO t VALUES (a);  
    SELECT s1 * a FROM t WHERE s1 >= b;  
END; //  
Query OK, 0 rows affected (0.00 sec)
```

Escopo de variáveis

```
mysql> CREATE PROCEDURE p11 ()  
BEGIN  
    DECLARE x1 char(5) DEFAULT 'outer';  
    BEGIN  
        DECLARE x1 char(5) DEFAULT 'inner';  
        SELECT x1;  
    END;  
    SELECT x1;  
END; //
```

Query OK, 0 rows affected (0.00 sec)

Escopo de variáveis(2)

```
mysql> call p11 ();//  
  
| x1      |  
+-----+  
  
| inner  |  
1 row in set (0.00 sec)  
  
| x1      |  
+-----+  
  
| outer  |  
1 row in set (0.00 sec)  
  
Query OK, 0 rows affected (0.00 sec)
```


Condições

```
mysql> create procedure p12 (in parametro INT)
BEGIN
    if parametro = 0 then
        select 'É igual a zero!';
    else
        select 'É diferente de zero!';
    end if;
end; //
```

Query OK, 0 rows affected (0.00 sec)

```
mysql> call p12(0); //
```

```
| É igual a zero! |
```

```
mysql> call p12(1); //
```

```
| É diferente de zero! |
```

Condições - Case

```
mysql> CREATE PROCEDURE p13 (in parametro INTEGER)
begin
    case parametro
        when 0 then SELECT 'Zero';
        when 1 THEN SELECT 'Um';
        ELSE SELECT 'Não é zero nem um';
    end case;
end; //

mysql> call p13(1); //
| Um |

mysql> call p13(2); //
| Não é zero nem um |

mysql> call p13(0); //
| Zero |
```

Exercício

- Crie uma procedure que diga se um número é par ou ímpar.
 - `SELECT 10 % 2;`
 - `SELECT 9 % 2;`
- Crie um procedimento que exiba o nome do mês, de acordo com o número (1=Janeiro, 2=Fevereiro, 3=Março, ...)

Resolução

```
create procedure parOUimpar (in parametro INT)
BEGIN
    if parametro % 2 then
        SELECT 'Ímpar';
    else
        SELECT 'par';
    end if;
END;
```

Resolução(2)

```
CREATE PROCEDURE nomeDoMes (in parametro
    INT)
BEGIN
    case parametro
        when 1 then SELECT 'Jan';
        when 2 then SELECT 'Fev';
        ...
        when 12 then SELECT 'Dez';
        else SELECT 'MÊS INVÁLIDO';
    end case;
END; //
```

Laços de repetição - WHILE

```
mysql> create procedure p14 (in parametro integer) begin
    while parametro >= 0 do
        insert into t values (parametro);
        set parametro = parametro - 1;
    end while;
```

```
end; //
```

```
mysql> select count(*) from t;
```

```
select count(*) from t; //
```

```
|      9 |
```

```
mysql> call p14 (2); //
```

```
Query OK, 1 row affected (0.00 sec)
```

```
mysql> select count(*) from t; //
```

```
|     12 |
```

Laços de repetição - REPEAT

```
mysql> create procedure p15 (in parametro integer)
begin
    REPEAT
        SELECT parametro;
        SET parametro = parametro + 1;
    until parametro >= 5
    end repeat;
end; //
```

Query OK, 0 rows affected (0.00 sec)

```
mysql> call p15 (3); //
```

|3|

|4|

Query OK, 0 rows affected (0.03 sec)

Laços de repetição - LOOP

```
CREATE PROCEDURE p16 (in parametro INT)
```

```
BEGIN
```

```
    loop_label: LOOP
```

```
        SELECT parametro;
```

```
        set parametro = parametro + 1;
```

```
        if parametro >= 5 then
```

```
            leave loop_label;
```

```
        end if;
```

```
    end loop;
```

```
end; //
```

```
mysql> call p16(3); //
```

```
|          3 |
```

```
|          4 |
```


Laços de repetição - estruturas

```
mysql> CREATE PROCEDURE p17 (in parametro INT)
  BEGIN
  primeiro_loop: LOOP
  set parametro = parametro + 1;
  if parametro <= 3 then
    ITERATE primeiro_loop;
  END IF;
  if parametro >= 5 then
    LEAVE primeiro_loop;
  END IF;
  select parametro;
  END LOOP;
END; //
mysql> call p17 (1); //
|          4 |
1 row in set (0.00 sec)
```

Exercício

- Criar um procedimento que exiba os números pares entre os dois parâmetros (números inteiros) passados

Resolução

```
CREATE PROCEDURE exhibePares(IN param1 INT, IN param2 INT)
BEGIN
    DECLARE aux INT;
    SET aux = param1;
    if aux % 2 THEN
        SET aux = aux + 1;
    END IF;
    WHILE aux <= param2 DO
        SELECT aux;
        SET aux = aux + 2;
    END WHILE;
END;
```

Controle de erros

- Empresa não existe

```
mysql> insert into pessoas(nome,empresa) values  
  ( 'teste',2312);//
```

```
ERROR 1216 (23000): Cannot add or update a child  
  row: a foreign key constraint fails
```

- Log de erros

```
Mysql> CREATE TABLE error_log (  
  Datetime datetime,  
  error_message varchar(80));//  
Query OK, 0 rows affected (0.01 sec)
```

Controle de erros(2)

```
mysql> CREATE PROCEDURE inserePessoa(nome varchar
    (50), empresa int, salario real)
BEGIN
    DECLARE EXIT HANDLER FOR 1216
        INSERT INTO error_log values (now(),CONCAT
            ('Erro de chave primária para o nome
            ',nome,' e empresa ',empresa));
    INSERT INTO pessoas (nome, empresa, salario)
        VALUES (nome, empresa, salario);
END; //
Query OK, 0 rows affected (0.09 sec)

mysql> call inserePessoa('Frederico','123'); //
Query OK, 1 row affected (0.05 sec)
```

Controle de erros(3)

```
mysql> CREATE PROCEDURE e1 ()
BEGIN
    DECLARE `Constraint violation`
        CONDITION FOR SQLSTATE '23000';
    DECLARE EXIT HANDLER FOR
        `Constraint violation` ROLLBACK;
    START TRANSACTION;
    INSERT INTO pessoas(nome, empresa) VALUES
('Fred',1);
    INSERT INTO pessoas(nome, empresa) VALUES
('Frederico',123123);
    COMMIT;
END; //
Query OK, 0 rows affected (0.04 sec)
mysql> call e1(); //
Query OK, 0 rows affected (0.09 sec)
```

Controle de erros(4)

```
mysql> CREATE PROCEDURE e2()  
BEGIN  
    DECLARE CONTINUE HANDLER FOR SQLSTATE '23000'  
        SELECT 'Houve um erro na inserção';  
    INSERT INTO pessoas(nome, empresa) VALUES  
        ('Fred', '1');  
    INSERT INTO pessoas(nome, empresa) VALUES  
        ('Frederico', '123123');  
END; //
```

```
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> call e2(); //
```

```
| Houve um erro na inserção |
```

```
1 row in set (0.05 sec)
```

```
Query OK, 0 rows affected (0.05 sec)
```

```
mysql> select count(*) from pessoas;
```

```
|      10 |
```

Controle de erros(5)

```
mysql> CREATE PROCEDURE e3()  
BEGIN  
    DECLARE CONTINUE HANDLER FOR NOT FOUND  
        BEGIN  
            SELECT 'Não encontrado';  
        END;  
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION  
        BEGIN  
            SELECT 'Erro fatal!';  
        END;  
    DECLARE CONTINUE HANDLER FOR SQLWARNING  
        BEGIN  
            SELECT 'Apenas um aviso!';  
        END;  
END; //
```


Exercício

- Alterar a tabela empresas, e acrescentar um campo telefone - varchar(15)
- Criar um procedimento que recebe o nome e o telefone por parâmetro, e insira a empresa conforme os parâmetros recebidos. Este procedimento deve ter um manipulador de “Warnings”, que deve exibir uma mensagem de que “houve um problema, mas mesmo assim o registro foi inserido”.

Resolução

```
CREATE PROCEDURE insereEmpresa (IN nome varchar
    (50), IN fone varchar(20))
BEGIN
    DECLARE EXIT HANDLER FOR SQLWARNING
    BEGIN
        SELECT 'Houve um erro (tamanho do campo fone), mas
        mesmo assim o registro foi inserido!';
    END;
    INSERT INTO empresas(nome, telefone) VALUES (nome,
    fone);
END;
```

Cursor

```
mysql> CREATE PROCEDURE cursor1()  
BEGIN  
    DECLARE a,b INT;  
    DECLARE cursor_1 CURSOR FOR select codigo FROM  
    pessoas;  
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET b=1;  
    OPEN cursor_1;  
    REPEAT  
        FETCH cursor_1 INTO a;  
        SELECT a;  
    UNTIL b=1 END REPEAT;  
    CLOSE cursor_1;  
END; //
```

Exercício

- Criar um procedimento que faça o mesmo do procedimento anterior (exiba os códigos de cada pessoa da tabela pessoas) mas que não exiba o último código duplicado.
- Dica: CONTINUE/EXIT HANDLER

Resolução

```
CREATE PROCEDURE cursor2()  
BEGIN  
    DECLARE a,b INT;  
    DECLARE cursor_1 CURSOR FOR select codigo FROM  
    pessoas;  
    DECLARE EXIT HANDLER FOR NOT FOUND SET b=1;  
    OPEN cursor_1;  
    REPEAT  
        FETCH cursor_1 INTO a;  
        SELECT a;  
    UNTIL b=1 END REPEAT;  
    CLOSE cursor_1;  
END; //
```

Privilégios

- GRANT CREATE ROUTINE ON teste.* TO user;
- GRANT EXECUTE ON teste.* TO user;
- GRANT SHOW ROUTINE ON teste.* TO user; *
- SQL SECURITY INVOKERS
- SQL SECURITY DEFINERS

Funções

- menos poderosas que procedimentos
- necessitam de um valor de retorno (RETURNS)
- não podem alterar os dados
- Limitações:
 - 'BEGIN END' DECLARE IF ITERATE LOOP
 - REPEAT RETURN 'SET X' WHILE

Funções(2)

```
mysql> create function empresa(cod int) returns
  varchar(50)
begin
  declare n varchar(50);
  select nome from empresas where codigo = cod into
    n;
  return n;
end; //
```

Query OK, 0 rows affected (0.00 sec)

```
mysql> select nome, empresa (empresa) from pessoas
  where codigo=1; //
```

nome	empresa (empresa)
Nasair Silva	Solis

1 row in set (0.00 sec)

Exercícios

- Crie uma função que:
 - retorne a soma dois números recebidos por parâmetro
 - retorne o quadrado de um número, passado por parâmetro
 - retorne o fatorial de um número, informado via parâmetro

```
mysql>
```

Resolução

```
CREATE FUNCTION quadrado(num int) RETURNS int
BEGIN
    SELECT num * num into num; return num;
END; //
create function soma(num1 int, num2 int) returns
int
begin
    select num1 + num2 INTO num1; return num1;
end; //
CREATE FUNCTION fatorial(numero int) RETURNS int
BEGIN
    if numero <= 1 then RETURN numero; else
        SELECT fatorial(numero-1)* numero into numero;
    end if; return numero;
end; //
```

Obrigado!

Nasair Júnior da Silva
nasair@solis.coop.br