MECATRÔNICA

PMR - Depto. de Enga. Mecatrônica

* PMR 2490-Laboratório

Prof. José Reinaldo Silva

Cada turma tem um acesso diferente nas máquinas do LAB:

Console	VM	passwd	MySQLuser	Passwd
console epusp	clicar em ReqVMw User: aluno	pmr2490	aluno	pmr2490

*O Uso dos computadores

Aluno		
no_USP	curso	ano_ing

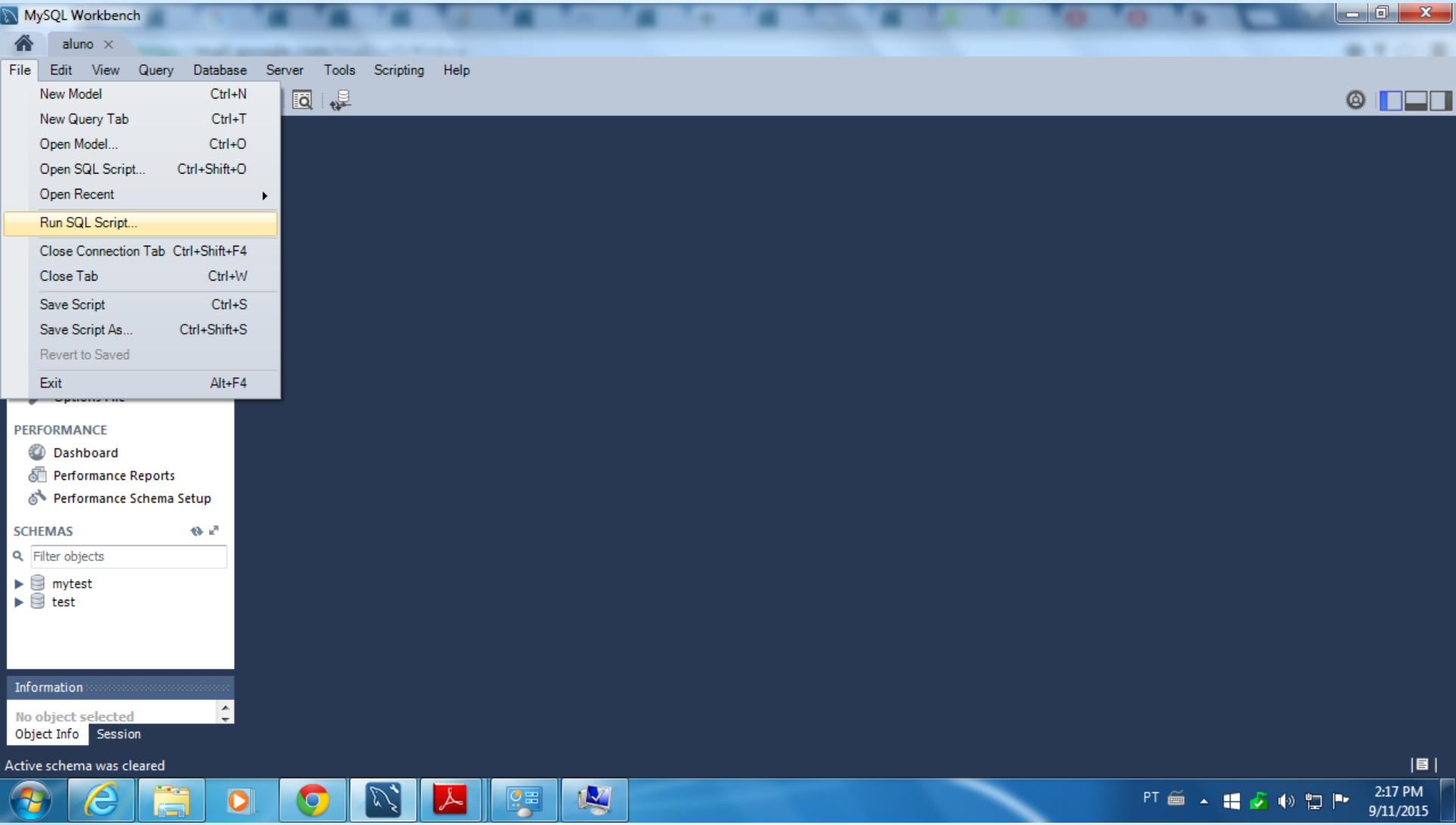
Professor	
no_funcional	titulacao

Disciplina		
codigo	no_creditos	objetivos

Matricula		
no_USP	no_funcional	codigo

```
#1. Executar só este comando para criar banco de dados Poli:

CREATE DATABASE poli;

#2. Left click no panel Schemas e escolher "Refresh All". Deverá aparecer o novo
Banco de dados. Selecione este banco (clique duas vezes no nome do banco) ou
execute o comando:

USE poli;

# com o banco escolhido vamos criar as tabelas,

CREATE TABLE aluno (id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
no_USP VARCHAR(12),
curso VARCHAR(20),
ano_ing INT
);

CREATE TABLE professor (id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
no_funcional VARCHAR(12),
titulacao VARCHAR(20)
```

*Usando um script


MySQL Workbench
aluno
File
Edit
View
Query
Database
Server
Tools
Scripting
Help
New Model Ctrl+N
New Query Tab Ctrl+T
Open Model... Ctrl+O
Open SQL Script... Ctrl+Shift+O
Open Recent
Run SQL Script...
Close Connection Tab Ctrl+Shift+F4
Close Tab Ctrl+W
Save Script Ctrl+S
Save Script As... Ctrl+Shift+S
Revert to Saved
Exit Alt+F4
PERFORMANCE
Dashboard
Performance Reports
Performance Schema Setup
SCHEMAS
Filter objects
mytest
test
Information
No object selected
Object Info
Session
Active schema was cleared
PT
2:17 PM
9/11/2015

Com este banco de dados poderemos fazer alguns exercícios de fixação e testes para a futura implementação do sistema.

Estes testes podem ser feitos fora do laboratório (para ganharmos tempo), e se tiverem dúvidas podem usr o forum de discussão que foi aberto (além do forum e noticias colocado no inicio dos módulos).

Pra começar, note que no modelo descrito não especificamos nenhuma chave. Embora seja razoável supor que o número USP do aluno, o número funcional do sejam chaves primárias (mais uma vez, já poderiamos ter definido as tabelas especificando a chave primária), mas fica agora como exercício.

Note agora que o relacionamento, mostra a uma conexão entre as entidades aluno, professor e disciplina que na verdade é o que compõe o histórico.

Exercício: O relacionamento pode ter como “chave” se for uma relação unívoca, as chaves primárias das entidades envolvidas. Nesse caso elas se chamam “chaves estrangeiras” e estão ligadas ao problema de relacionamento que já discutimos: integridade referencial.

departamento

REFERENCIAL
INTEGRITY

empregado

Database Fundamentals, pag. 119

Aluno		
no_USP	curso	ano_ing

Professor	
no_funcional	titulacao

Disciplina		
codigo	no_creditos	objetivos

Matricula		
no_USP	no_funcional	codigo

Creating Indexes

An index is an ordered list of all the values that reside in a group of one or more columns at a given time. Such a list makes queries that test the values in those columns vastly more efficient. However, indexes
take up data storage space and must be changed whenever the data is changed. Therefore, you should make a cost-benefit analysis in each case to determine whether and how indexes should be used. Oracle can use indexes to improve performance when:

- searching for rows with specified index column values
- accessing tables in index column order

Constraints allow you to define rules for the data in your table. There are different types of constraints:

- A UNIQUE constraint prevents duplicate values in a table. This is implemented using unique indexes and is specified in the CREATE TABLE statement using the keyword UNIQUE. A NULL is part of the UNIQUE data values domain.
- A PRIMARY KEY constraint is similar to a UNIQUE constraint, however it excludes NULL as valid data. Primary keys always have an index associated with it.
- A REFERENTIAL constraint is used to support referential integrity which allows you to manage relationships between tables. This is discussed in more detail in the next section.
- A CHECK constraint ensures the values you enter into a column are within the rules specified in the constraint.

Database Fundamentals, pag. 118

```
CREATE TABLE EMPLOYEE
  (ID           INTEGER     NOT NULL  PRIMARY KEY,
   NAME         VARCHAR(9),
   DEPT         SMALLINT    CHECK (DEPT BETWEEN 10 AND 100),
   JOB          CHAR(5)     CHECK (JOB IN ('Sales','Mgr','Clerk')),
   HIREDATE     DATE,
   SALARY       DECIMAL(7,2),
   CONSTRAINT   YEARSAL     CHECK ( YEAR(HIREDATE) > 1986
                                    OR SALARY > 40500 )
  )
```

For this table, four constrains should be satisfied before any data can be inserted into the table. These constraints are:

- PRIMARY KEY constraint on the column ID

  This means that no duplicate values or nulls can be inserted.

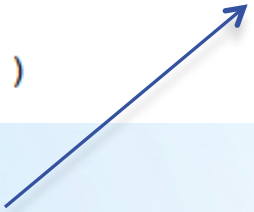
- CHECK constraint on DEPT column

  Only allows inserting data if the values are between 10 and 100.

- CHECK constraint on JOB column

  Only allows inserting data if the values are ‘Sales’, ‘Mgr’ or ‘Clerk'.

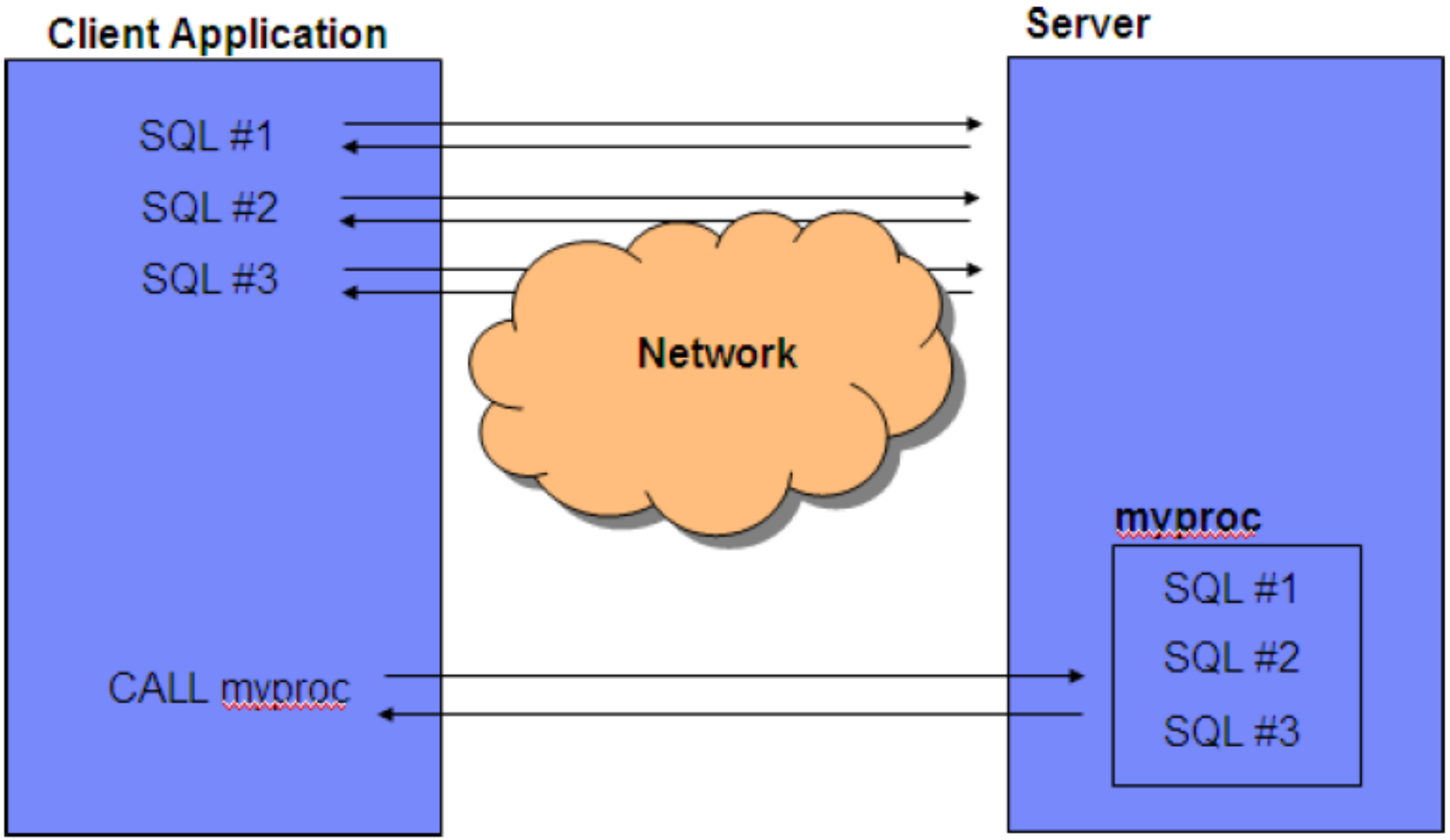
```
CREATE TABLE EMPLOYEE
   (ID           INTEGER      NOT NULL  PRIMARY KEY,
    NAME         VARCHAR(9),
    DEPT         SMALLINT     CHECK (DEPT BETWEEN 10 AND 100),
    JOB          CHAR(5)      CHECK (JOB IN ('Sales','Mgr','Clerk')),
    HIREDATE     DATE,
    SALARY       DECIMAL(7,2),
    CONSTRAINT   YEARSAL      CHECK ( YEAR(HIREDATE) > 1986
                                      OR SALARY > 40500 )
   )
```



- CHECK constraint on the combination of the HIREDATE and SALARY columns

  Only allows to insert data if the hire date year is greater than 1986 and the SALARY is greater than 40500.

Stored Procedures



- The procedure name.
- Input and output parameter attributes.
- The language in which the procedure is written. For an SQL procedure, the language is SQL.
- Information that will be used when the procedure is called, such as runtime options, length of time that the procedure can run, and whether the procedure returns result sets.

Database Fundamentals, pag. 146

```
CREATE PROCEDURE UPDATESALARY          (1)
 (IN EMPNUMBR CHAR(10),                (2)
  IN RATE DECIMAL(6,2))
 LANGUAGE SQL                          (3)
  UPDATE EMP                           (4)
   SET SALARY = SALARY * RATE
   WHERE EMPNO = EMPNUMBR
```

In the example:

1. The stored procedure name is UPDATESALARY.
2. There are two parameters, EMPNUMBR with data type CHAR(10), and RATE with data type DECIMAL(6,2). Both are input parameters.
3. LANGUAGE SQL indicates that this is an SQL procedure, so a procedure body follows the other parameters.
4. The procedure body consists of a single SQL UPDATE statement, which updates rows in the employee table.

13.1.9 CREATE PROCEDURE and CREATE FUNCTION Syntax

```
CREATE
    [DEFINER = { user | CURRENT_USER }]
    PROCEDURE sp_name ([proc_parameter[,...]])
    [characteristic ...] routine_body

CREATE
    [DEFINER = { user | CURRENT_USER }]
    FUNCTION sp_name ([func_parameter[,...]])
    RETURNS type
    [characteristic ...] routine_body

proc_parameter:
    [ IN | OUT | INOUT ] param_name type

func_parameter:
    param_name type

type:
    Any valid MySQL data type

characteristic:
    COMMENT 'string'
  | LANGUAGE SQL
  | [NOT] DETERMINISTIC
  | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
  | SQL SECURITY { DEFINER | INVOKER }

routine_body:
    Valid SQL routine statement
```

These statements create stored routines. By default, a routine is associated with the default database. To associate

« 13.1.8 CREATE INDEX Syntax

13.1.10 CREATE TABLE Syntax »

Section Navigation [Toggle]

- 13.1 Data Definition Statements
- 13.1.1 ALTER DATABASE Syntax
- 13.1.2 ALTER FUNCTION Syntax
- 13.1.3 ALTER PROCEDURE Syntax
- 13.1.4 ALTER TABLE Syntax
- 13.1.5 ALTER VIEW Syntax
- 13.1.6 CREATE DATABASE Syntax
- 13.1.7 CREATE FUNCTION Syntax
- 13.1.8 CREATE INDEX Syntax
- 13.1.9 CREATE PROCEDURE and CREATE FUNCTION Syntax
- 13.1.10 CREATE TABLE Syntax
- 13.1.11 CREATE TRIGGER Syntax
- 13.1.12 CREATE VIEW Syntax
- 13.1.13 DROP DATABASE Syntax
- 13.1.14 DROP FUNCTION Syntax
- 13.1.15 DROP INDEX Syntax
- 13.1.16 DROP PROCEDURE and DROP FUNCTION Syntax
- 13.1.17 DROP TABLE Syntax
- 13.1.18 DROP TRIGGER Syntax
- 13.1.19 DROP VIEW Syntax
- 13.1.20 RENAME TABLE Syntax
- 13.1.21 TRUNCATE TABLE Syntax