Bem vindo a nossa primeira aula.

Vamos instalar as ferramentas necessárias para a primeira parte do curso.   
Verificar se seu Windows tem instalado o Microsoft .NET Framework 4

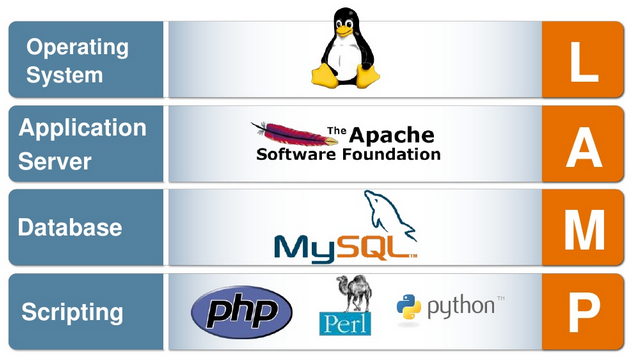
1. MySQL [http://dev.mysql.com/downloads/windows/installer/5.6.html](http://dev.mysql.com/downloads/windows/installer/5.6.html%20)

282M mysql-installer-community-5.6.23.0.msi

1. Notepad++ <http://notepad-plus-plus.org/>  
   Editor de texto
2. brModelo <http://sis4.com/brModelo/download.aspx>  
   Ferramenta de modelagem

É um banco de dados relacional que se tornou famoso e amplamente usado em todo o mundo por fazer parte do modelo LAMP (Linux, Apache, **MySQL** e PHP). Leve e rápido com bons recursos de programação.

Foi adquirido pela SUN Microsystems em 2008 e logo em seguida pela ORACLE em 2010. Ele é opensource, logo disponível de graça dentro dos termos da licença [GNU General Public Licence](http://pt.wikipedia.org/wiki/GNU_General_Public_License).

****

**Oracle Certified Professional, MySQL 5.6 Database Administrator–** [**Prova 1z0-883**](https://education.oracle.com/pls/web_prod-plq-dad/db_pages.getpage?page_id=653&get_params=p_id:260)

**Putty** – ferramente para acesso remoto usando SSH ou Telnet   
<http://www.putty.org/>

**Vi** – versão atual VIM (Vi improved)   
Também foi comentado na aula que o **DBA deve saber comandos básicos** da ferramenta de edição de texto usada em ambientes UNIX e semelhantes. Existe uma versão para Windows <ftp://ftp.vim.org/pub/vim/pc/gvim74.exe>

**Conexão ao banco de dados**

mysql –u root –p coti  
mysql –u root –p   
  
**Acessar diretamente base de dados desejada**  
mysql –u user –p –D banco

**Verificar base de dados disponíveis**  
SHOW DATABASES;

**Criar base de dados**  
CREATE DATABASE teste;  
CREATE DATABASE aula1;

**Eliminar banco de dados –** Sómente se os dados não forem necessários

DROP DATABASE IF EXISTS teste;

**Selecionar uma base de dados**  
USE aula1;

**Criar uma tabela;**  
CREATE TABLE aluno(  
 nome VARCHAR(20),  
 idade INT,  
 sexo CHAR,  
 email VARCHAR(30));

**Listar tabelas existentes no MySQL**  
SHOW TABLES;

**Descrever estrutura da tabela**  
DESCRIBE aluno;  
DESC aluno;  
EXPLAIN TABLE aluno; -- Porém com características avançadas adicionais.

**Inserir dados na tabela aluno**  
**INSERT INTO aluno VALUES ('Joao', 24,'m' ,'joao@gmail.com');**

**Exibir os dados da tabela**  
SELECT \* FROM aluno;

SELECT nome, idade FROM aluno;  
SELECT idade, email, sexo FROM aluno;

**Adicionar novos dados**

INSERT INTO aluno VALUES ('Luan', 23,'m','joao@gmail.com');  
INSERT INTO aluno VALUES ('Hilto', 24,'m','Hilto@gmail.com');  
INSERT INTO aluno VALUES ('Renato', 25,'m','Renato@gmail.com');  
INSERT INTO aluno VALUES ('Flavia', 26,'f','Flavia@gmail.com');  
INSERT INTO aluno VALUES ('Felipe', 23,'m','Felipe@gmail.com');  
**INSERT INTO aluno VALUES ('Joao',24 ,'m','joao@gmail.com');**

**Outra maneira de inserir os dados sem precisar repetir a sintaxe INSERT**

INSERT INTO aluno VALUES ('Leila', 25,'f','Leila@gmail.com'),

('Alef', 26,'m','Alef@gmail.com'),

('Adriano', 27,'m','Adriano@gmail.com'),

('Joao', 24,'m','joao@gmail.com'),

('Julio', 23,'m','Julio@gmail.com');

**Exibir novamente os resultados**

SELECT \* FROM aluno;

Por não ter uma estrutura de chaves foi possível duplicar um ALUNO e também inserir registros (dados) permitindo a entrada de valores NULOS. Neste caso o funcionário do sistema não informou a idade dos novos alunos.

**Atualizar as informações**

SELECT \* FROM aluno WHERE nome = 'Joao';

UPDATE aluno SET nome='Joao Marques', email='jmarques@gmail.com'  
WHERE nome='Joao' AND email='joao@gmail.com';

***O código irá atualizar os três registros já que não foi determinado qual o João.***

SELECT \* FROM aluno WHERE nome = 'Joao';  
SELECT \* FROM aluno WHERE nome = 'Joao Marques';

No exemplo anterior todos as instancias de Joao foram também trocadas para **Joao Marques** com email **jmarques@gmail.com**. Quando não há um atributo identificador (chave) a única maneira é usando o **LIMIT**.

Selecionando a ocorrência.

SELECT \* FROM aluno   
WHERE email = 'jmarques@gmail.com';

Temos três ocorrências de Joao Marques, então vamos limitar para que o UPDATE apenas aconteça em um dos registros.

UPDATE aluno set nome = 'Joao', email = 'joao@gmail.com'   
WHERE email = 'jmarques@gmail.com'  
LIMIT 1;

SELECT \* FROM aluno   
WHERE email = 'jmarques@gmail.com' OR email = 'joao@gmail.com';

***Você deve ter percebido que o ALUNO Luan também tem o mesmo e-mail do Joao. Então...***

**UPDATE aluno SET email='luan@gmail.com' WHERE nome ='Luan';**

SELECT \* FROM aluno   
WHERE email = 'jmarques@gmail.com' OR email = 'joao@gmail.com';

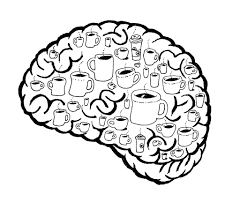
**Removendo registro**Sobrou um registro de Joao Marques que não é necessário.

DELETE FROM aluno; **Cuidado! Sem especificar o que você deseja apagar, todo o conteúdo da tabela será eliminado.**

Agora podemos eliminar um dos registros duplicados.

DELETE FROM aluno WHERE email='jmarques@gmail.com' LIMIT 1;

Observação: Os dados usados são fictícios e não correspondem a nenhuma pessoa ou evento.

**O que é um banco de dados**  
É um grupo de ferramentas usada para armazenar informações(dados) e permite que você crie(INSERT), leia(READ), atualize(UPDATE) e remova(DELETE) os dados de alguma maneira.

Exemplos: Caderno de anotações   
 Gabinete de pastas   
 O cérebro humano – O mais sofisticado banco de dados já criado.

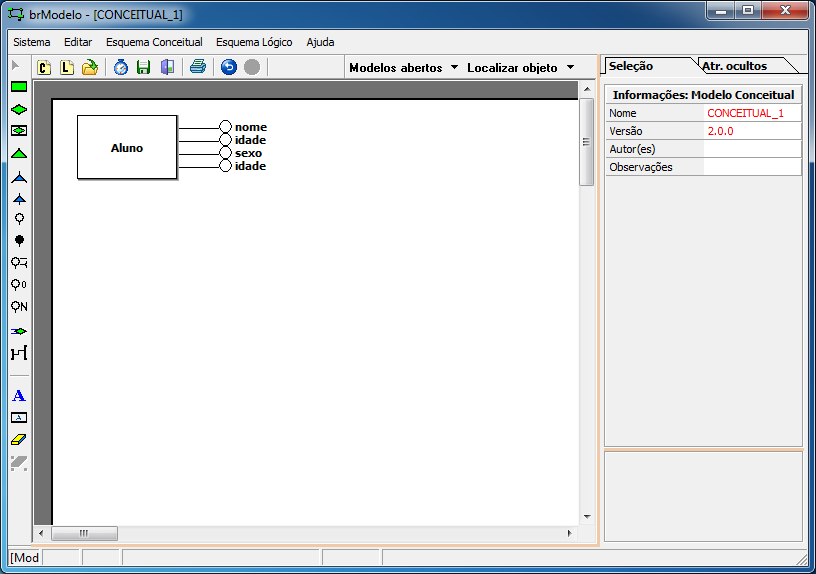
Um dos problemas dele é que o cérebro esquece. Segundo; ele tem sua capacidade de processamento reduzida quando estamos cansados.

**Porque usar um banco de dados**

1. Arquitetura Cliente Servidor
2. Mecanismos básicos de interação. CRUD(**c**reate, **r**ead, **u**pdate, **d**elete)
3. Eliminar ou pelo menos minimizar a redundância de dados  
   Primary Key
4. Segurança
5. Integridade referêncial
   1. Garantir que valores inseridos em uma tabela tenham registros semelhantes em outra  
      Foreign Key
6. Replicação e tolerança a falhas
   1. Cluster e Backup
7. Baixo custo total de propriedade. TCO (Total Cost Ownership)
   1. Modelo opensource
   2. Escalabilidade incremental e horizontal
   3. Fácil manutenção

**DATABASE Design** – Modelar o negócio

1. Metas de um modelo de dados efetivo
   1. Senso de direção – “If all you have is a hammer, everything looks like a nail.”
      1. Existem diferentes tipos de banco de dados.
   2. Uma modelagem com falhas pode comprometer um projeto todo antes de o mesmo começar a ser desenvolvido.
   3. Um bom design prove uma fundação sólida para o resto de suas aplicações
2. DER Diagrama de Entidade-Relacionamento (ERD – Entity Relationship Diagram)
   1. Representar objetos de seu negócio de maneira gráfica



* 1. Entidade – Tabelas  
     Colunas – Atributos  
     Linhas – Registros ou *Tuples*
  2. Esquemas CONCEITUAL, LÓGICO e FÍSICO