SQL com MySQL no XAMPP

Banco de Dados

Prof. Dr. Gerson Pastre de Oliveira

Instalar o Xampp

1. Baixe a versão do XAMPP que corresponde ao seu sistema operacional (Windows, Linux ou MacOS) no site oficial: https://www.apachefriends.org/index.html
2. Execute o arquivo baixado e siga as instruções do assistente de instalação. É possível selecionar quais componentes serão instalados, mas o recomendado é manter as opções padrão.
3. Escolha o diretório de instalação e conclua a instalação.
4. Após a instalação, abra o painel de controle do XAMPP. No Windows, isso pode ser feito clicando no ícone do XAMPP na bandeja do sistema (ao lado do relógio) e selecionando "XAMPP Control Panel". No Linux e MacOS, é necessário abrir o terminal e digitar o comando sudo /opt/lampp/manager-linux-x64.run.

Instalar o Xampp

5. No painel de controle do XAMPP, inicie os serviços Apache e MySQL clicando nos botões "Start". Se os serviços estiverem em execução, os botões serão exibidos em verde.

6. Agora você pode acessar o servidor Apache digitando "http://localhost" em um navegador da web. Se tudo estiver funcionando corretamente, você verá a página de boas-vindas do XAMPP.

Acessar o Xampp e criar um BD

1. Abra um terminal de comando (ou prompt de comando) no seu sistema operacional.
2. Navegue até a pasta onde o XAMPP foi instalado. No Windows, por exemplo, isso pode ser feito digitando o comando "cd C:\xampp\mysql\bin" (sem aspas).
3. Inicie o cliente MySQL digitando o comando mysql -u root -p. Isso solicitará a senha do usuário root, que é a senha padrão do MySQL no XAMPP.
4. Após digitar a senha e pressionar Enter, você estará conectado ao servidor MySQL. Agora você pode criar um novo banco de dados digitando o comando CREATE DATABASE nome_do_banco_de_dados;

Acessar o Xampp e criar um BD

5. É possível verificar se o BD foi criado corretamente digitando show databases;
6. Este comando irá listar todos os bancos de dados disponíveis. O nome do banco de dados que acabou de ser criado deve estar na lista

Acessar o Xampp e criar um BD

- Por exemplo, para criar um BD chamado **empresa** usando os comandos mencionados, digite:

```
create database empresa;
```

- “Abra” o banco de dados criado:

```
use empresa;
```

- Caso queira verificar se já existe alguma tabela previamente criada em um BD existente:

```
show tables;
```

Criar tabelas em BD

- Digite o comando SQL para criar uma tabela na janela de consulta SQL. O comando SQL básico (sintaxe) para criar uma tabela é o seguinte:

```
CREATE TABLE nome_da_tabela (
coluna1 TIPO_DE_DADO,
coluna2 TIPO_DE_DADO,
 ...
);
```

- Basta substituir nome_da_tabela pelo nome que deseja dar à sua tabela, e coluna1, coluna2, etc. pelos nomes das colunas que deseja criar e TIPO_DE_DADO pelo tipo de dado que deseja armazenar em cada coluna

Criar tabelas em BD

- Para criar a tabela clientes no BD empresa, supondo que o mesmo esteja em uso:

```
CREATE TABLE clientes (
        id INT(11) NOT NULL AUTO_INCREMENT,
        nome VARCHAR(255) NOT NULL,
        email VARCHAR(255) NOT NULL,
        PRIMARY KEY (id));
```

Inserir linhas em uma tabela

- O comando INSERT do MySQL é usado para inserir uma ou várias linhas em uma tabela existente
- A sintaxe básica do comando INSERT é a seguinte:

```
INSERT INTO nome_da_tabela (coluna1, coluna2, ...) VALUES (valor1, valor2, ...)
```

- nome_da_tabela é o nome da tabela na qual se deseja inserir os dados, coluna1, coluna2, etc. são os nomes das colunas nas quais se deseja inserir dados e valor1, valor2, etc. são os valores que serão inseridos em cada coluna

Inserir linhas em uma tabela

- Por exemplo, para inserir uma nova linha na tabela **clientes** do exemplo anterior, pode-se usar o seguinte comando SQL:

  INSERT INTO clientes (nome, email) VALUES (‘Maria da Silva’, ‘maria.silva@email.com');

- Este comando SQL insere uma nova linha na tabela **clientes** com o valor “Maria da Silva" na coluna *nome* e “maria.silva@email.com" na coluna *email*
- O valor da coluna *id* será gerado automaticamente, pois essa coluna foi definida como "AUTO_INCREMENT" no exemplo anterior

Inserir linhas em uma tabela

- Pode-se inserir várias linhas de uma vez em uma tabela usando uma única consulta INSERT
- Para isso, basta separar cada conjunto de valores com vírgulas:

```
INSERT INTO clientes (nome, email) VALUES

('João da Silva', 'joao.silva@exemplo.com'),

('Maria Souza', 'maria.souza@exemplo.com'),

('Pedro Pereira', 'pedro.pereira@exemplo.com');
```

Alterar a estrutura de uma tabela

- Com o comando ALTER TABLE, é possível adicionar, modificar ou remover colunas, alterar o tipo de dados de colunas existentes, adicionar ou remover índices, entre outras ações
- A sintaxe básica do comando ALTER TABLE é a seguinte:

```
ALTER TABLE [nome_da_tabela] [ação];
```

- No caso, deve-se substituir nome_da_tabela pelo nome da tabela na qual se deseja fazer uma alteração e ação pela alteração que se deseja fazer na tabela

Alterar a estrutura de uma tabela

- Para adicionar uma nova coluna a uma tabela existente, o comando ALTER TABLE pode ser usado com a cláusula ADD COLUMN, seguida pelo nome e tipo de dados da nova coluna. Por exemplo:

```
ALTER TABLE tabela_exemplo ADD COLUMN nova_coluna INT;
```

- Na tabela **clientes**, por exemplo:

```
ALTER TABLE clientes ADD telefone VARCHAR(30);
```

Alterar a estrutura de uma tabela

- Para remover uma coluna existente, pode-se usar a cláusula DROP COLUMN em conjunto com o nome da coluna a ser removida. Por exemplo:

```
ALTER TABLE tabela_exemplo DROP COLUMN coluna_a_ser_removida;
```

- Na tabela **clientes**, por exemplo:

```
ALTER TABLE clientes DROP COLUMN telefone;
```

Alterar a estrutura de uma tabela

- Para modificar uma coluna existente, use a cláusula MODIFY COLUMN e especifique o novo tipo de dados e outras propriedades da coluna, como tamanho ou se ela pode ser nula ou não. Por exemplo:

```
ALTER TABLE tabela_exemplo MODIFY COLUMN coluna_existente
VARCHAR(255) NOT NULL;
```

- Na tabela **clientes**, por exemplo:

```
ALTER TABLE clientes MODIFY COLUMN telefone VARCHAR(40) NOT NULL;
```

Alterar a estrutura de uma tabela

- Outras alterações possíveis:

  ALTER TABLE clientes DROP PRIMARY KEY; (remover PK)

  ALTER TABLE clientes ADD PRIMARY KEY (id); (adicionar PK)

- Renomear tabela:

  ALTER TABLE clientes RENAME parceiros;

  RENAME TABLE clientes TO parceiros;

- Deletar tabela:

  DROP TABLE clientes;

- Após as alterações (ou em qualquer momento), pode-se mostrar a estrutura de uma tabela:

  DESC clientes;

alter table [tabela] [opção] <dados ou configurações>

Exs:

alter table prof add email varchar(50);

alter table prof drop primary key;
alter table prof add primary key (cod);

alter table prof modify email varchar(60);

insert into [tabela] values (<valor_campo_1>, <valor_campo_2>, ..., <valor_campo_N>);

Ex:

insert into prof values (2, "Giovanna luiza", "Direito", 6000, "gilu@xyz.com");

Obs: campos tem de ester em ordem e nenhum pode ser omitido.

insert into [tabela] (<campo1>, <campo2>, ..., <campoN>) values (<valor_campo_1>, <valor_campo_2>, ..., <valor_campo_N>);

Ex:

insert into prof (cod, nome, salario) VALUES (3, "marra", 6500);

create table [tabela] (<campos> <caracteristicas>, ...);

Ex:

create table prof (cod smallint not null, nome varchar(80), salario float,
primary key (cod));

chave primária

Select <campos> → quais campos serão exibidos
From <tabelas> → quais tabelas estão envolvidas na consulta
where <condições>;
↳ quais restrições serão impostas ao resultado da consulta

Ex: Mostrar nome, formação e salário dos professores que recebem + do que 5000

SELECT nome, formacao, salario
FROM prof
WHERE salario > 5000;

↳ todos os dados
SELECT *
⋮

1. Mostrar ra e nome dos alunos que tenham menos de 5 faltas e façam o curso de Redes;
2. Mostrar os dados dos alunos aprovados por nota (média ≥ 6);
3. Mostrar os dados dos alunos aprovados por nota e frequência, considerando que foram dadas 40 aulas;
4. Mostrar ra, nome e mensalidade dos alunos que pagam entre 300 e 500;
5. Mostrar os dados de todos os alunos de Redes e ADS.

```
SELECT ra, nome FROM aluno WHERE faltas < 5 AND curso = "Redes";

SELECT * FROM aluno WHERE (nota1 + nota2) / 2 >= 6;

SELECT * FROM aluno WHERE (nota1 + nota2) / 2 >= 6 AND 1 - faltas / 40 >= 0.75;

SELECT ra, nome, mensalidade FROM aluno WHERE mensalidade >= 300 AND mensalidade <= 500;

SELECT * FROM aluno WHERE curso = "Redes" OR curso = "ADS";
```

6) Retornar os alunos que possuem o maior valor de média das notas.

- Quais são as médias dos alunos?

```
SELECT (nota1 + nota2)/2 FROM aluno;
```

- Quais são as maiores dentre estas médias?

```
SELECT *
FROM aluno
WHERE (nota1+nota2)/2 ≥ ALL
(SELECT (nota1+nota2)/2
 FROM aluno);
```

----> todos (≥ ALL : maior que todos)

Subconsulta

7) Retornar os alunos que pagam as menores mensalidades;

8) Retornar os alunos com as maiores frequências.

7

subconsulta

```
MariaDB [escola]> SELECT * FROM aluno WHERE mensalidade <= ALL (SELECT mensalidade FROM aluno);
+-----+--------------------+-------+-------+-------+------------+---------+---------+
| ra  | nome               | curso | nota1 | nota2 | mensalidade | faltas | periodo |
+-----+--------------------+-------+-------+-------+------------+---------+---------+
| 104 | Marina de Oliveira | GTI   |     6 |   6.5 |        300 |     20 | manhã   |
+-----+--------------------+-------+-------+-------+------------+---------+---------+
1 row in set (0.010 sec)
```

alunos que pagam as menores mensalidades

```
MariaDB [escola]> SELECT * FROM aluno WHERE mensalidade < ALL (SELECT mensalidade FROM aluno);
Empty set (0.000 sec)
```

não utilizar <!

8

```
MariaDB [escola]> SELECT 1 - faltas / 40 FROM aluno;
+----------------+
| 1 - faltas / 40 |
+----------------+
|         0.9500 |
|         0.7500 |
|         0.8750 |
|         0.9750 |
|         0.5000 |
|         0.9500 |
+----------------+
6 rows in set (0.007 sec)
```

subconsulta

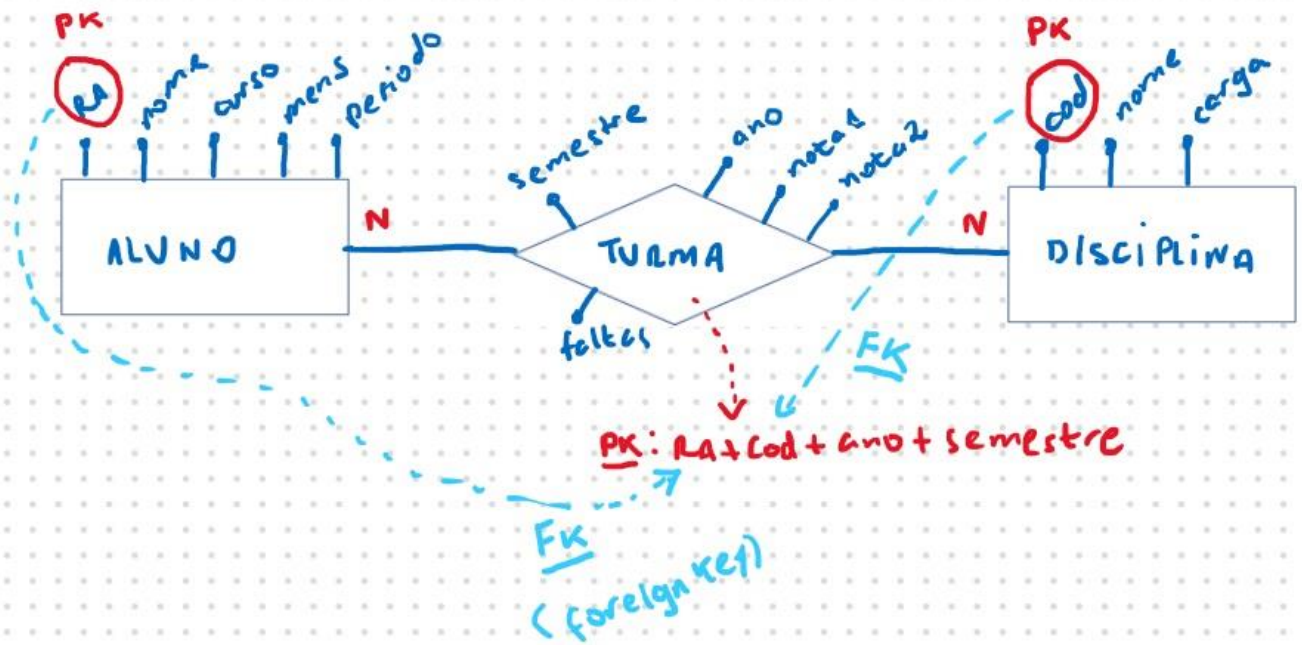
```
MariaDB [escola]> SELECT * FROM aluno WHERE (1 - faltas / 40) >= ALL (SELECT 1 - faltas / 40 FROM aluno);
+-----+---------------+---------+-------+-------+------------+---------+---------+
| ra  | nome          | curso   | nota1 | nota2 | mensalidade | faltas | periodo |
+-----+---------------+---------+-------+-------+------------+---------+---------+
| 103 | Giovanna Luiza | Servicos |     9 |     7 |        400 |      1 | manhã   |
+-----+---------------+---------+-------+-------+------------+---------+---------+
1 row in set (0.000 sec)
```

número de aulas dadas

```
MariaDB [escola]> SELECT * FROM aluno WHERE (1 - faltas / 40) < SOME (SELECT 1 - faltas / 40 FROM aluno);
+-----+--------------------+-------+-------+-------+--------------+---------+---------+
| ra  | nome               | curso | nota1 | nota2 | mensalidade | faltas | periodo |
+-----+--------------------+-------+-------+-------+--------------+---------+---------+
| 100 | Jose da Silva      | Redes |   7 |   8 |       500 |     2 | manhã   |
| 101 | Maria das Couves   | ADS   |  3.5 |  4.5 |       350 |    10 | manhã   |
| 102 | Roberto de Souza   | Redes |   5 |   3 |       600 |     5 | manhã   |
| 104 | Marina de Oliveira | GTI   |   6 |  6.5 |       300 |    20 | manhã   |
| 105 | Sophie Marie       | ADS   |   8 |   8 |       600 |     2 | tarde   |
+-----+--------------------+-------+-------+-------+--------------+---------+---------+
5 rows in set (0.001 sec)
```

menor que o que possui o maior no critério procurado (descarta a > frequência)

sentença	contrário
> = ALL	< SOME
< = ALL	> SOME




```
Create database fatec;
use fatec;
```

```
create database fatec;
use fatec;
```

criar BD

"abrir" BD

```
CREATE TABLE aluno(ra INT NOT NULL, nome VARCHAR(100), curso VARCHAR(50), periodo VARCHAR(20), mens DOUBLE, PRIMARY KEY(ra)) ENGINE=INNODB;
```

mens → mensalidade

criar tabela aluno → conectividade

```
CREATE TABLE disciplina (cod INT NOT NULL, nome VARCHAR(100), carga INT, PRIMARY KEY(cod)) ENGINE=INNODB;
```

criar tabela disciplina

```
CREATE TABLE turma (ra_aluno INT NOT NULL, cod_disc INT NOT NULL, semestre INT NOT NULL, ano INT NOT NULL, PRIMARY KEY(ra_aluno, ra_disc, semestre, ano)) ENGINE=INNODB;
```

criar tabela turma

```
ALTER TABLE turma ADD nota1 DOUBLE;
ALTER TABLE turma ADD nota2 DOUBLE;
ALTER TABLE turma ADD faltas INT;
```

acrescentar campos em uma tabela

```
ALTER TABLE turma ADD CONSTRAINT fk_ra_aluno FOREIGN KEY (ra_aluno) REFERENCES aluno(ra);
```

→ criar relacionamento via FK

- turma: conjunto de relacionamentos
- CONSTRAINT: restrição
- fk_ra_aluno: nome
- FOREIGN KEY (ra_aluno): quem é a chave estrangeira na tabela turma
- REFERENCES aluno(ra): com quem a FK se liga na tabela de referência
- geralmente é a PK da tabela de referência

```
ALTER TABLE turma ADD CONSTRAINT fk_cod_disc FOREIGN KEY (cod_disc) REFERENCES disciplina(cod);
```

Ex. 1: Mostrar o ra do aluno e seu nome de todos os alunos reprovados por nota; mostrar, também, a média, o curso e o nome da disciplina em que foram reprovados.

```
SELECT ra, aluno.nome, curso, disciplina.nome, (nota1 + nota2)/2 FROM aluno, disciplina, turma WHERE (nota1 + nota2) / 2 < 6 AND ra = ra_aluno AND cod = cod_disc;
```

↳ se o nome do campo está em 2 ou + tabelas, inserir, separado por ponto, o nome da tabela

critério 1 (média < 6) reprovado por nota

critério 2 (org. tabelas) ra liga as tabelas aluno e turma

critério 3 (org. tabelas) cod liga as tabelas turma e disciplina

Consultas que envolvam duas ou mais tabelas em um modelo de entidades - relacionamentos precisam, obrigatoriamente, indicar como as tabelas se relacionam (as chaves PK e FK).

Ex. 2: Mostrar todos os dados dos alunos do período da manhã que tenham sido aprovados por nota

```
SELECT aluno.* FROM aluno, turma WHERE periodo = "manhã" AND (nota1 + nota2)/2 >=6 AND ra = ra_aluno;
```

Em quais disciplinas foram aprovados?

```
SELECT aluno.*, cod_disc FROM aluno, turma WHERE periodo = "manhã" AND (nota1 + nota2)/2 >=6 AND ra = ra_aluno;
```

```
SELECT aluno.*, cod_disc, disciplina.nome FROM aluno, turma, disciplina WHERE periodo = "manhã" AND (nota1 + nota2)/2 >=6 AND ra = ra_aluno AND cod = cod_disc;
```

3) Mostrar ra e nome de todos os alunos que cursaram matemática, foram aprovados e pagam mais de 400 de mensalidade.

4) mostrar ra, nome, nome da disciplina, média e faltas de todos os alunos reprovados por falta ou nota.

5) Mostrar o nome e a carga de todas as disciplinas que não tenham tido nenhum aluno reprovado.

6) (PESQUISAR) Mostrar quantos alunos foram reprovados em 2020.

3

```
SELECT ra, aluno.nome, (nota1+nota2)/2 AS média, 1 - faltas/carga AS frequência, mensalidade
FROM disciplina, aluno, turma WHERE disciplina.nome = "matemática" AND mensalidade > 400 AND
(nota1+nota2)/2 >= 6 AND 1 - faltas/carga >= 0.75 AND cod = cod_disc AND ra = ra_aluno;
```

→ qualquer nome que comece com "matemática"

```
SELECT ra, aluno.nome, (nota1+nota2)/2 AS média, 1 - faltas/carga AS frequência, mensalidade
FROM disciplina, aluno, turma WHERE disciplina.nome LIKE "matemática%" AND mensalidade > 400
AND (nota1+nota2)/2 >= 6 AND 1 - faltas/carga >= 0.75 AND cod = cod_disc AND ra = ra_aluno;
```

4

```
SELECT ra, aluno.nome, disciplina.nome, (nota1+nota2)/2 AS média, faltas FROM disciplina, aluno,
turma WHERE ra = ra_aluno AND cod = cod_disc AND ((nota1+nota2)/2 < 6 OR 1 - faltas/carga <
0.75);
```

conexão aluno/turma

conexão disciplina/turma

reprovado por nota

OU

reprovado por falta

5

Disciplinas ±s

Disciplinas que tenham alunos reprovados

não está contido em

apenas um cod de cada tipo

subconsulta

```
SELECT disciplina.* FROM disciplina WHERE cod NOT IN (SELECT DISTINCT cod FROM disciplina, turma WHERE cod = cod_disc AND ((nota1+nota2)/2 < 6 OR 1 - faltas/carga < 0.75));
```

↳ cod das disciplinas que têm alunos reprovados

UNION
INTERSECT
EXCEPT

Como fazer esta consulta usando IN no lugar de NOT IN?

Não é simplesmente inverter!

A − B

```
(SELECT DISTINCT disciplina.* FROM disciplina) EXCEPT (SELECT DISTINCT disciplina.* FROM disciplina, turma WHERE cod = cod_disc AND ((nota1+nota2)/2 < 6 OR 1 - faltas/carga < 0.75));
```

(A − B) ⇒ todos os elementos que estão em A, mas NÃO ESTÃO em B

A = todas as disciplinas
B = disciplinas com alunos reprovados

} A − B = disciplinas que NÃO TEM alunos reprovados!

6

```
SELECT COUNT(ra_aluno) FROM turma,disciplina WHERE cod=cod_disc AND
((nota1+nota2)/2 < 6 OR 1 - faltas/carga < 0.75) AND ano = 2020;
```

quantas reprovações em 2020

→ conta quantos ra_aluno surgem no resultado da consulta

↳ como mostrar quantos alunos, e não quantas reprovações?

7 Mostrar o valor total de mensalidades p/ um mês no ano de 2020 (SUM)

```
select SUM(mensalidade) from aluno, turma where ra = ra_aluno AND ano = 2020;
```

SUM → soma os valores de mensalidade

ra = ra_aluno → de alunos registrados em turma

ano = 2020 → no ano de 2020

... mas, como mensalidades vários vezes (se o alun faz + de 1 disciplina)

8 Mostrar a média de valores de mensalidades para alunos de Redes (AVG)

```
SELECT AVG(mensalidade) FROM aluno WHERE curso="Redes";
```

AVG → média aritmética dos valores de mensalidade