GERENCIADOR DE BANDO DE DADOS
MySQL

O QUE JÁ SABEMOS...

- O que é um Banco de Dados.
- Projeto de Banco de Dados
- Modelo de Dados Relacional
- Diagrama Entidade Relacionamento
- SGBDs
- Normalização
- Etc.

SQL

- Structured Query Language,ou Linguagem de Consulta Estruturada ou SQL;
- Trata-se de uma linguagem específica para a manipulação de tabelas de dados;
- A linguagem padrão universal para manipular bancos de dados relacionais através dos SGBDs.

GRUPOS DE COMANDOS SQL

- Os comandos do SQL são classificados
- em três grupos, de acordo com suas
- principais funções:
- DML – Data Manipulation Language
- DDL – Data Definition Language
- DCL – Data Control Language

SQL X MYSQL

- Só para constarmos o MySQL não é uma extensão do SQL.
- O MySQL é um Sistema de Gerenciamento de Banco de Dados
- O SQL é a linguagem para manipulação dos dados no SGBD.

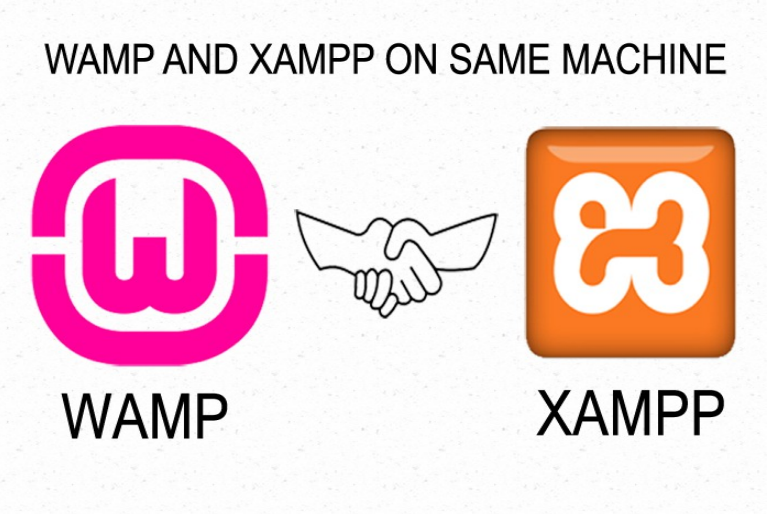
SQL X MYSQL

- Para utilizar as características e o funcionamento do SQL é preciso de um SGBD.
- SGBD é um ambiente no qual possamos utilizar os comandos desta linguagem para manipular dados.

Figura 1.1: Exemplos de SGBDs

INSTALAÇÃO MYSQL

- Existem alternativas para conseguir o MySQL em seu computador.
- Baixar o MySQL no seu site e instala-lo;
- Instalar pacotes que venham com o MySQL incluso, caso do XAMPP e WAMP;
- MySQL Workbench;



SQL – REGRAS

- Todas as palavras-chave das instruções SQL serão escritas em maiúsculo;
- Sempre no final de cada instrução, deve ser terminado com um ponto-e-virgula (;)

Utilização no Windows

- Abrir o Prompt Comando do Windows.
- Atalho: Win + R
- Executar: cmd
- Acessar o diretório c:/xampp/mysql/bin pelo prompt
- Usar o comando:
- cd xampp/mysql/bin

CONEXÃO COM MYSQL

- Precisamos utilizar um comando para acessar o prompt do MySQL.
- Ao instalarmos o MySQL é obrigatório criar um usuário e senha para o acesso dos Banco de Dados. Por padrão, o usuário é root e a senha é vazia.
- Estas informações (usuário e senha) são necessários para este passo.

CONEXÃO COM MYSQL

- O comando para acessarmos o MySQL é:
  - mysql –u usuario –p senha
- Em nosso caso ficando:
  - mysql –u root –p

Conexão realizada!

```
C:\xampp\mysql\bin>mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 5.1.41 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

```
root@debian:/home/allan# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.7.22 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

CRIAR UM BANCO DE DADOS

- Para criar de um banco de dados o comando é simples.
- mysql> CREATE DATABASE meu-banco;
- CREATE DATABASE seguido do nome desejado de banco de dados.

- Ex: mysql>CREATE DATABASE aula;

MOSTRAR BANCO DE DADOS

- Podemos verificar rapidamente a existência do BD recém-criado, bem como a de todos os outros criados anteriormente, utilizando a instrução SHOW DATABASES (mostrar bancos de dados);

- mysql> SHOW DATABASES;

CRIAR BANCO DE DADOS

- SE NÃO EXISTIR...
- Para verificar se exibe um determinado banco de dados antes da criação de um novo. O comando utilizado é:

- mysql> CREATE DATABASE IF NOT EXISTS meu-banco;

DELETAR UM BANCO DE DADOS

- Para excluir um banco de dados, usa-se o comando DROP DATABASE, seguido do nome do banco de dados que deseja deletar.

- mysql> DROP DATABASE meu-banco;

ATENÇÃO AO DELETAR

- É preciso ressaltar que, ao apagar um banco de dados, todas as suas tabelas e os dados nelas contidos também serão apagados e, portanto, perdidos de maneira irreversível.

RECAPITULANDO

- Partindo do conceito que vimos que o SQL é divido em três grupos. Estes comandos que utilizamos se enquadram em qual deles?

  a) DML

  b) DDL

  c) DCL

RECAPITULANDO

- Partindo do conceito que vimos que o SQL é divido em três grupos. Estes comandos que utilizamos se enquadram em qual deles?

  a) DML

  b) DDL

  c) DCL

COMO SELECIONAR UM BANCO DE DADOS

- Como vimos, podemos criar vários bancos de dados, porém, podemos manipular apenas um por vez. Assim, antes de começar, é preciso selecionar qual será o banco de dados que queremos alterar.
- Isso é feito utilizando o comando USE (“usar” em inglês), seguido pelo nome do banco de dados em questão.

- mysql> USE meu-banco;

Tipos de dados básicos

- Em banco de dados relacionais, cada tabela pode conter diversas colunas, as quais armazenarão os dados. Para cada coluna, existirá um tipo de dado associado. Os tipos de dados são definidos durante a criação da tabela.

Quadro 2.1: Principais tipos de dados simples definidos pela SQL:2003

Tipos de Dados	Descrição
CHARACTER	Caractere de tamanho fixo – usualmente conhecido como CHAR
CHARACTER VARYING	Caractere de tamanho variante – usualmente conhecido como VARCHAR
CHARACTER LARGE OBJECT	Caractere longo – usualmente conhecido como CLOB
BINARY LARGE OBJECT	String binária para objetos longos – usualmente conhecido como BLOB
NUMERIC	Numérico exato
DECIMAL	Numérico exato
SMALLINT	Numérico exato
INTERGER	Numérico exato
BIGINT	Numérico exato
FLOAT	Numérico aproximado
REAL	Numérico aproximado
DOUBLE PRECISION	Numérico aproximado
BOOLEAN	Booleano
DATE	Data com informações de dia, mês e ano
TIME	Hora com informações de hora, minuto e segundo
TIMESTAMP	Determina um momento, com informações de ano, mês, dia, hora, minuto e segundo

Fonte: Costa (2006)

CRIAR UMA TABELA

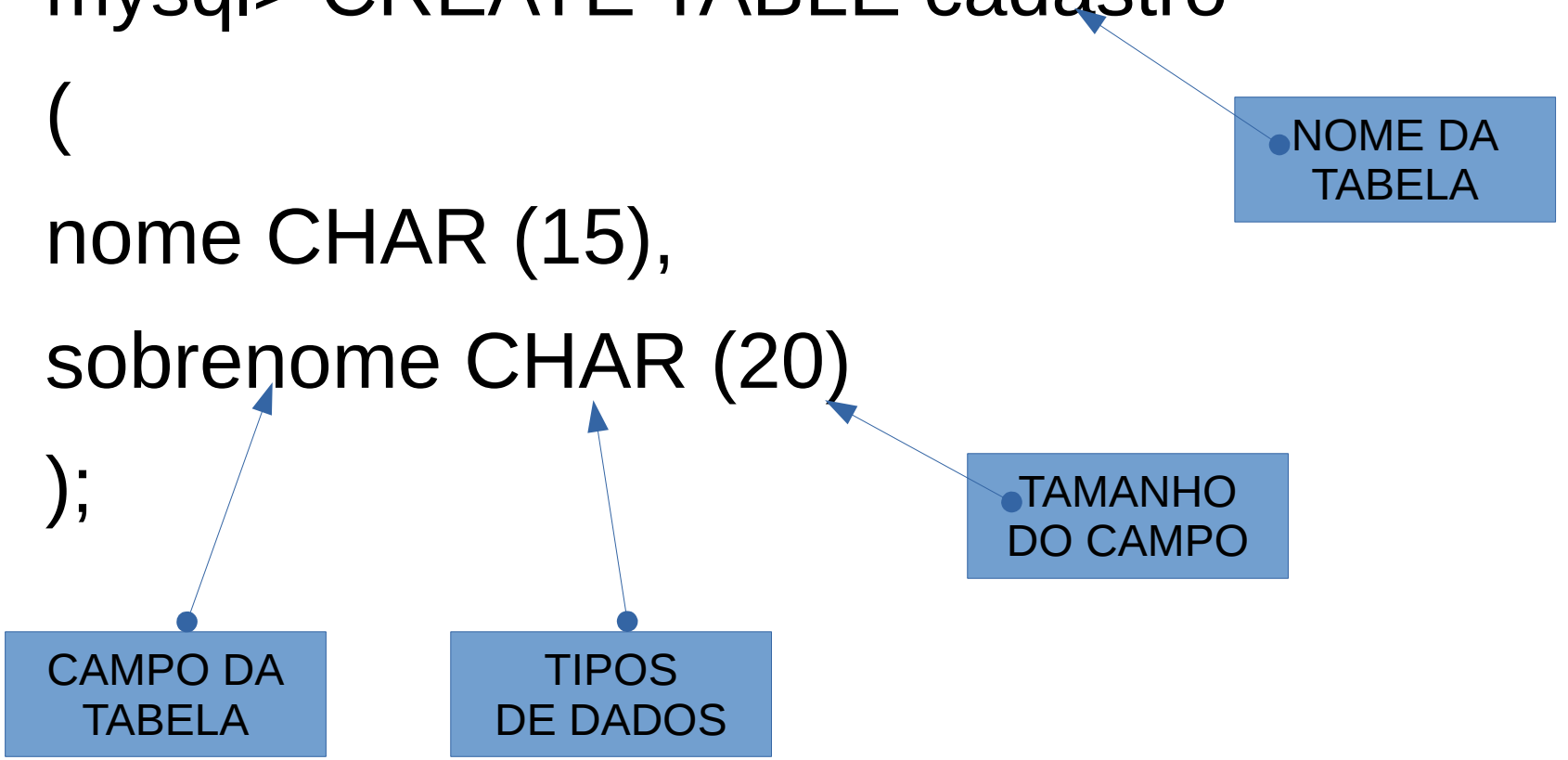
- A regra base do comando para criar uma tabela no banco de dados é o comando para criar tabela, seguido do nome da tabela.
- Também é necessário informar os campos da tabela, seu tipo e seu tamanho.

CRIAR UMA TABELA

- mysql> CREATE TABLE cadastro
  (
  nome CHAR (15),
  sobrenome CHAR (20)
  );

CRIAR UMA TABELA

- mysql> CREATE TABLE cadastro
  (
  nome CHAR (15),
  sobrenome CHAR (20)
  );



TIPOS DE CAMPOS

- Existem vários tipos possíveis de dados no SQL, embora os mais comuns sejam:
- INT ou INTEGER: Para inteiros de tamanho normais
- TIMESTAMP: Para data e hora e pode ser atribuídos automaticamente;
- CHAR e VARCHAR: Para caracteres até no max 255 de tamanho;
- TEXT ou LONGTEXT: Para textos longos;

MOSTRAR TABELA

- Para exibir a lista de tabelas do banco de dados que está usando atualmente, basta utilizar o comando:

- mysql> SHOW TABLES;

MOSTRAR ESTRUTURA DA TABELA

- Podemos também analisar a estrutura de uma tabela de maneira aprofundada usando o comando DESCRIBE (“descrever”, em inglês), seguido pelo nome da tabela.

- mysql> DESCRIBE minha-tabela;

INSERIR VALORES NA TABELA

- O comando de INSERIR é um dos mais utilizados. Para inserir valores em uma determinada tabela, basta seguir a regra:

- mysql> INSERT nome_da_tabela VALUES (‘valor1’, ‘valor2’, ...);

- mysql> INSERT tabela (campo1,campo2, campo3, ...) VALUES (“valor1”,“valor2”, “valor3”);

SELECIONAR VALORES DA TABELA

- É possível selecionar valores da tabela, utilizando o comando SELECT do SQL. O comando SELECT é, basicamente, a ferramenta principal para consultar informações de um banco de dados, por isso, é comumente chamado de query.

- mysql> SELECT dados_desejados FROM nome_tabela;

SELECIONAR VALORES DA TABELA

- Podemos definir alguns critérios na seleção de dados. Há duas possíveis alternativas para estes critérios, a utilização de um asterisco (*) e da interrogação (?);

ASTERISCO (*)

- Significa tudo, ou seja, todos os dados. Pode ser combinado com um ou mais caracteres para especificar conjuntos de dados com algo em comum, por exemplo, em geral, se digitarmos o critério A* significa que queremos ver todos os registros cujo conteúdo começa com a letra A;

INTERROGAÇÃO (?)

- Representa um caractere desconhecido. Por exemplo, se definirmos como critério o valor ?????, quer dizer que queremos ver somente os registros que, em determinado campo, contenham valores de cinco caracteres.

ALTERAR TABELA

- Para alterar uma tabela, basta utilizar ALTER TABLE, o nome da tabela o qual quer alterar e qual operação de alteração quer fazer.
- Operações: Adicionar novo campo, renomear nome da tabela e etc.

RENOMEAR, ADICIONAR E MODIFICAR

- mysql> ALTER TABLE pessoas **RENAME** TO cadastros;
- mysql> ALTER TABLE pessoas **ADD** idade INT(3);
- mysql> ALTER TABLE pessoas **MODIFY** idade INT(5);

DELETAR E ORDENAR

- mysql> ALTER TABLE pessoas **DROP** cadastros;
- mysql> ALTER TABLE pessoas **ADD** idade INT(3) **AFTER** campo;
- mysql> ALTER TABLE pessoas **ADD** idade INT(3) **FIRST**;

OPÇÕES DOS CAMPOS

- Alguns campos podem ter particularidades. Por exemplo, ser chave primária, não pode ser vazia e etc. Veremos algumas opções.

NOT NULL

- O campo com a opção NOT NULL, significa que o campo não poderá ser nulo. Para utilizar isso, basta na criação do campo adicionar NOT NULL na frente dele.

- mysql> CREATE TABLE pessoas (nome VARCHAR(255) NOT NULL);

PRIMARY KEY

- Para definirmos que um campo é chave primária, utilizamos a opção PRIMARY KEY, após o nome do campo.

- mysql> CREATE TABLE pessoas (id INT(5) PRIMARY KEY);

AUTO INCREMENT

- Auto incremento, significa que a cada registro de uma tabela, o valor será incrementado (aumentado).
- Geralmente, utilizamos para campos ID,CODIGO ou CHAVES PRIMARIAS;

AUTO INCREMENT

- mysql> CREATE TABLE animals (id INT(5) NOT NULL PRIMARY KEY AUTO_INCREMENT, name VARCHAR(50) NOT NULL);

CLAUSULA WHERE

- Usando a cláusula WHERE, podemos especificar um critério de seleção paraselecionar os registros necessários de uma tabela.

CLAUSULA WHERE

- O WHERE funciona como uma condição em qualquer linguagem de programação. Esta cláusula é usada para comparar determinado valor com o valor do campo disponível na tabela MySQL. Exemplo:

- SELECIONE campo_x DA tabela_y ONDE campo_x seja igual ao valor

CLAUSULA WHERE

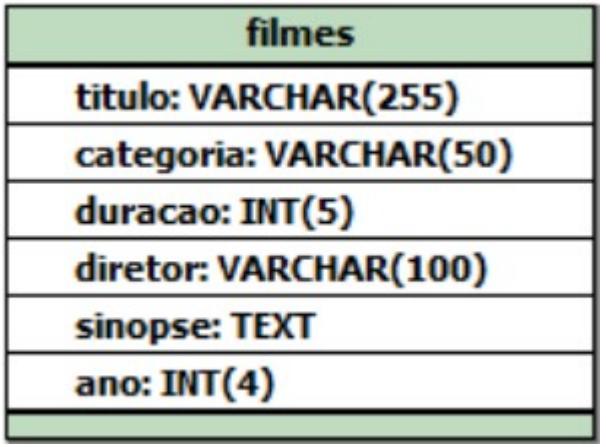
- mysql> SELECT * FROM pessoas WHERE id=1;

CLAUSULA WHERE

Operador	Descrição	Exemplo
=	Verifica se os valores dos dois operadores são iguais ou não, se sim, então condição torna-se verdade.	(A = B) não é verdade.
! =	Verifica se os valores de dois operandos são iguais ou não, se os valores não são iguais então a condição torna-se verdade.	(A! = B) é verdadeiro.
>	Verifica se o valor do operando esquerdo é maior que o valor do operando da direita, se sim, então a condição se torna verdadeira.	(A> B) não é verdade.
<	Verifica se o valor do operando esquerdo é menor que o valor do operando da direita, se sim, então condição torna-se verdade.	(A <B) é verdadeiro.
> =	Verifica se o valor do operando esquerdo é maior ou igual ao valor do operando da direita, se sim, então a condição se torna verdadeira.	(A> = B) não é verdade.
<=	Verifica se o valor do operando esquerdo é menor ou igual ao valor do operando da direita, se sim, então condição torna-se verdade.	(A <= B) é verdadeiro.

ATIVIDADE

- Crie um banco de dados chamado cinema.
- Cria a tabela filmes:
- Insira 5 registro;
- Mostre apenas os campos titulo, duração e ano dos filmes cadastrados;



Criando chave primária com restrições

- Vale ressaltar que a ordem de criação dessas tabelas é de suma importância. Isso se deve ao fato das tabelas estarem conectadas através de suas chaves primárias e estrangeiras.

```
CREATE TABLE departamento
(cod_dep INTEGER NOT NULL,
descr CHAR(30) NOT NULL DEFAULT 'Não Informado',
localiz CHAR(30) NOT NULL,
CONSTRAINT pk_dep PRIMARY KEY(cod_dep));
```

Criando chave primária com restrições

ON DELETE especifica os procedimentos que devem ser feitos pelo SGBD quando houver uma exclusão de um registro na tabela pai quando existe um registro correspondente nas tabelas filhas. As opções disponíveis são:

RESTRICT - opção *default*. Esta opção não permite a exclusão na tabela pai de um registro cuja chave primária exista em alguma tabela filha.

CASCADE - esta opção realiza a exclusão em todas as tabelas filhas que possua o valor da chave que será excluída na tabela pai.

SET NULL - esta opção atribui o valor NULO nas colunas das tabelas filhas que contenha o valor da chave que será excluída na tabela pai.

Criando chave primária com restrições

- Observamos que a tabela funcionario possui duas restrições (constraint).

```
CREATE TABLE funcionario
(cod_func INTEGER NOT NULL,
nome CHAR(50) NOT NULL,
dt_nasc DATE,
cod_dep INT,
CONSTRAINT pk_func PRIMARY KEY(cod_func),
CONSTRAINT fk_func FOREIGN KEY(cod_dep) REFERENCES departamento(cod_dep));
```

- A primeira determina o código do funcionário (cod_func) como a chave primária
- A segunda restrição determina o atributo cod_dep como chave estrangeira que veio da tabela departamento

Criando chave primária com restrições

```
CREATE TABLE funcao
(cod_funcao INTEGER NOT NULL,
nome CHAR(50) NOT NULL,
sal REAL NOT NULL,
CONSTRAINT pk_funcao PRIMARY KEY(cod_funcao));
```

```
CREATE TABLE projeto
(cod_proj INTEGER NOT NULL,
nome CHAR(50) NOT NULL,
orcamento REAL NOT NULL,
dt_ini DATE NOT NULL,
dt_prev_term DATE NOT NULL,
CONSTRAINT pk_projeto PRIMARY KEY(cod_proj),
CONSTRAINT verifica_datas CHECK (dt_ini<dt_prev_term));
```

Criando chave primária com restrições

- A cláusula check serve para implementarmos restrições de domínio.
- inserimos uma restrição que garante que a data de início do projeto (dt_ini) seja menor que a data prevista de término (dt_prev_term).

Criando chave primária com restrições

- Na tabela trabalha, inserimos uma restrição chamada checa_datas para garantir que a data de entrada do funcionário no projeto (dt_ent) seja sempre menor que a sua data de saída (dt_sai).

```
CREATE TABLE trabalha
(cod_func INTEGER NOT NULL,
cod_proj INTEGER NOT NULL,
cod_funcao INTEGER NOT NULL,
dt_ent DATE NOT NULL,
dt_sai DATE,
CONSTRAINT pk_trabalha PRIMARY KEY(cod_func,cod_proj),
CONSTRAINT fk_trabalha1 FOREIGN KEY(cod_func) REFERENCES funcionario(cod_func),
CONSTRAINT fk_trabalha2 FOREIGN KEY(cod_proj) REFERENCES projeto(cod_proj),
CONSTRAINT fk_trabalha3 FOREIGN KEY(cod_funcao) REFERENCES funcao(cod_funcao),
CONSTRAINT checa_datas CHECK (dt_ent<dt_sai));
```

Subconsultas

- Realizar subconsultas é uma forma de combinar mais de uma consulta (select) obtendo apenas um resultado.
- Imagine que precisamos obter o nome de todos os funcionários que estão lotados no departamento de contabilidade.
- Perceba que o **nome do departamento** está na tabela departamento, enquanto que o **nome do funcionário** está na tabela funcionario.

```
Select nome from funcionario where cod_dep = (select cod_dep from departamento
where descr='contabilidade');
```

Manipulando dados

- Imagine que precisamos cadastrar, na tabela funcionario, todos os registros da tabela pessoa.

Quadro 2.1: Tabela "pessoa"

Codigo	Apelido	Data_nasc	Cod_setor	Nome_mae
100	Joãozinho	01/01/1980	1	Francisca
200	Maricota	02/02/1979	1	Raimunda
300	Franzé	03/03/1978	1	Joanete

```
INSERT INTO funcionario(cod_func, nome, dt_nasc, cod_dep)
SELECT codigo, apelido, data_nasc, cod_setor FROM pessoa;
```

Alterando dados de uma tabela

- Para alterarmos uma informação contida numa tabela do banco de dados, utilizamos o comando update.

```
UPDATE nome-tabela
     SET <nome-coluna = <novo conteúdo para o campo>
        [nome-coluna = <novo conteúdo para o campo>]
     WHERE condição
```

nome-tabela representa o nome da tabela cujo conteúdo será alterado.

nome-coluna representa o nome da(s) coluna(s) terão seus conteúdos alterados com o novo valor especificado.

condição representa a condição para a seleção dos registros que serão atualizados. Esta seleção poderá resultar em um ou vários registros. Neste caso a alteração irá ocorrer em todos os registros selecionados.

Alterando dados de uma tabela

- Exemplos:

```
UPDATE projeto
SET orcamento=1000
WHERE cod_proj=1 OR cod_proj=5;
```

Figura 2.22: Alteração do orçamento dos projetos de código 1 ou 5

```
UPDATE projeto
SET orcamento=2000;
```

Figura 2.23: Alteração de todos os orçamentos de projetos

Excluindo dados de uma tabela

- O comando delete é utilizado para excluir linhas de uma tabela.

```
DELETE FROM  nome-tabela WHERE condição
```

```
DELETE FROM projeto
WHERE orcamento>2000;
```

Figura 2.25: Remoção dos projetos que custam mais de 2000

Funções agregadas

- Muitas vezes, precisamos de informações que resultado de alguma operação aritmética ou de conjunto sobre os dados contidos nas tabelas de um banco de dados.
- Para isso, utilizamos as funções agregadas.

Função count( )

- Conta a quantidade de linhas de uma tabela que satisfazem uma determinada condição.

```
SELECT COUNT(cod_proj) FROM projeto;
```

Figura 2.26: Contagem da quantidade de códigos de projetos

- Perceba que dentro dos parênteses da função count colocamos o atributo que será utilizado para a contagem.

```
SELECT COUNT(cod_proj) FROM projeto
WHERE orcamento>2000;
```

Figura 2.27: Contagem de código de projetos que custam mais de 2000

Função avg( )

- A função avg é responsável por extrair a média aritmética dos valores de uma coluna.

```
SELECT AVG(orcamento) FROM projeto;
```

Figura 2.28: Média dos orçamentos dos projetos

Função sum( )

- A função sum é responsável por realizar a soma dos valores de uma coluna.

```
SELECT SUM(orcamento) FROM projeto WHERE cod_proj<10;
```

Figura 2.29: Instrução SQL - soma

- o SGBD realizará a soma dos orçamentos dos projetos cujo código seja menor que 10.

Função min( )

- A função min obtém o valor mínimo dentre os elementos de uma coluna.

```
SELECT MIN(cod_func) FROM funcionario
WHERE dt_nasc>='1980-01-01' AND dt_nasc<='1980-12-31';
```

Figura 2.30: Instrução SQL - mínimo

Função max( )

- A função max obtém o maior valor dentre os elementos de uma coluna.

```
SELECT MAX(cod_func) FROM funcionario
WHERE dt_nasc>='1980-01-01' AND dt_nasc<='1980-12-31';
```

Figura 2.31: Instrução SQL - máximo

LIMITAR

- Pode-se limitar a quantidades de registros. Se não queremos uma lista extensa e só precisamos das 5 primeiras, coloca-se o LIMIT de 5.

- mysql> SELECT * FROM pessoas **LIMIT** 5;

ORDENAR

- Quando for necessário ordenar a limita a lista de registros em ordem crescente (ASC) ou decrescente (DESC).
- Para utilizar a ordenação, precisa escolher por qual campo será feita a ordenação.

ORDERNAR

- ORDEM DECRESCENTE
- mysql> SELECT * FROM pessoas **ORDER BY** idade **DESC;**
- ORDEM CRESCENTE
- mysql> SELECT * FROM pessoas **ORDER BY** idade **ASC;**

LIKE

- O LIKE é usado para fazer buscas por partes de conteúdos. Por exemplos, precisamos capturar todas as pessoas com que tem Ana no nome, utilizamos do seguinte código

LIKE

- mysql> SELECT * FROM pessoas WHERE nome **LIKE** '%ana%’ LIMIT 2;

- O LIKE é utilizado da seguinte forma:
- **LIKE %conteudo%**

A cláusula group by

- Os dados resultantes de uma seleção podem ser agrupados de acordo com um critério específico.
- Desejamos obter, para cada código de projeto, a quantidade de funcionários que trabalharam nele. Lembre-se que, para sabermos qual funcionário trabalha em qual projeto, teremos que observar a tabela trabalha, pois é ela que acontecem as associações entre funcionários e projetos.

A cláusula group by

- Para respondermos a pergunta anterior, vamos considerar as seguintes informações na tabela trabalha apresentadas no Quadro 2.2.

Quadro 2.2: Tabela "trabalha"

Cod_func	Cod_proj	Cod_funcao	Dt_ent	Dt_sai
1	1	1	2010-02-02	2010-03-03
2	1	2	2010-02-02	2010-03-03
1	2	1	2010-04-04	2010-05-05
4	2	2	2010-04-04	2010-05-05
3	1	3	2010-02-02	2010-03-03

A cláusula group by

- Perceba que o funcionário 1 trabalhou no projeto 1 e no projeto 2. Perceba também que 3 funcionários trabalharam no projeto 1 e apenas 2 funcionários trabalharam no projeto 2. No projeto 1, trabalharam os funcionários de código 1, 2 e 3. Já no projeto 2, trabalharam os funcionários de código 1 e 4.

A cláusula group by

Quadro 2.3: Número de funcionários por projeto	
Cod_proj	Quantidade_funcionários
1	3
2	2

- A solução para o nosso problema é:

```
SELECT cod_proj, COUNT(cod_proj) 'Quantidade_funicionários'
FROM trabalha GROUP BY cod_proj;
```

Figura 2.32: Instrução SQL - contagem

Junções (join)

- Quando precisamos realizar consultas que envolvam mais de uma tabela, uma das soluções seria a utilização de junções. As junções permitem que acessemos mais de uma tabela utilizando apenas um select.
- Normalmente, deve existir a chave primária de uma tabela fazendo relação com a chave estrangeira da outra tabela.
- Esta será a condição de ligação entre as tabelas.

Junções (join)

- Consideremos as tabelas funcionario e departamento. Elas servirão de base para os tópicos seguintes.

Quadro 2.4: Tabela "funcionario"

Cod_func	Nome	Dt_nasc	Cod_dep
1	João	1980-01-02	1
2	José	1981-02-03	2
3	Maria	1982-05-04	1
4	Antônio	1983-07-06	3

Quadro 2.5: Tabela "departamento"

Cod_dep	Descr	Localiz
1	Desenvolvimento	Sala C3-10
2	Análise	Sala B2-30
3	Testes	Sala C1-10
4	Contabilidade	Sala A1-20

Junção interna (inner Join)

- A junção interna entre tabelas é a modalidade de junção que faz com que somente participem da relação resultante as linhas das tabelas de origem que atenderem à cláusula de junção.

```
SELECT nome, descr FROM funcionario f INNER JOIN departamento d
ON f.cod_dep=d.cod_dep;
```

Logo após a cláusula ON, inserimos a condição para a junção.

Quadro 2.6: Relação resultante

Nome	Descr
João	Desenvolvimento
José	Análise
Maria	Testes
Antônio	Contabilidade

Junções externas (outer join)

- Na junção externa, os registros que participam do resultado da junção não obedecem obrigatoriamente à condição de junção, ou seja, a não inexistência de valores correspondentes não limita a participação de linhas no resultado de uma consulta.
- Existem tipos diferentes de junção externa.

Junção externa à esquerda (left outer join)

- Suponha que desejemos uma listagem com os nomes de todos os departamentos cadastrados no nosso banco de dados e, para aqueles que possuam funcionários lotados nele, apresente os respectivos nomes.

```
SELECT descr, nome FROM departamento d LEFT OUTER JOIN funcionario f
ON f.cod_dep=d.cod_dep;
```

Junção externa à esquerda (left outer join)

- A instrução anterior produzirá o resultado , a partir das tabelas funcionario e departamento.

Quadro 2.6: Relação resultante

Descr	Nome
Desenvolvimento	João
Desenvolvimento	Maria
Análise	José
Testes	Antônio
Contabilidade	

Junção externa à direita (right outer join)

- A junção externa à direita é muito parecida com a junção externa à esquerda. A única diferença está no fato de que a tabela da qual todas as linhas constarão no resultado está posicionada à direita do termo right outer join no comando.

```
SELECT descr, nome FROM  funcionario f RIGHT OUTER JOIN departamento d
ON f.cod_dep=d.cod_dep;
```