Banco de Dados

**"O bom julgamento vem da experiência. E de onde vem a experiência?
A experiência vem do mau julgamento."**

Mark Twain

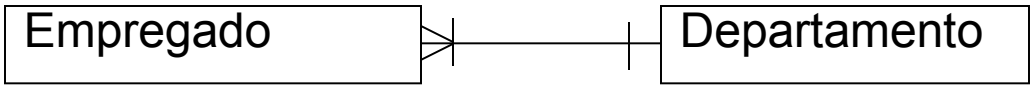
NOTAS DE AULA

(Laboratório)

Arquivo 3

Rafael Dias Ribeiro, M.Sc.
rafaeldiasribeiro@gmail.com

Sócio Efetivo da:
SBC Sociedade Brasileira de Computação



Quando iniciamos o trabalho no modelo físico, devemos observar as restrições e “exportações” de chaves para garantir o relacionamento entre as tabelas.

No exemplo acima, sabemos que a entidade Empregado receberá a chave da entidade Departamento, assim quando iniciamos a criação de códigos SQL é aconselhável primeiro criarmos a tabela de Departamento para depois criarmos a tabela Empregado.

Crie as tabelas abaixo no MySQL

Empregado:
matricula (inteiro 5 posições)
nomeemp(caracter 30 posições)
numdepto(inteiro 4 posições)
dataadmin(data)
salaemp(numérico 4 posições)
salario(numérico 10,2 posições)
gerência(inteiro 5 posições)

Departamento:
numdepto(inteiro 4 posições)
nomedepto(caracter 20 posições)
localdepto(caracter 20 posições)
orcamdepto(numérico 12,2 posições)

Crie as tabelas abaixo no MySQL

```
CREATE TABLE departamento(
                    numdepto int(4) AUTO_INCREMENT,
                    nomedepto varchar(20) NOT NULL,
                    localdepto varchar(20),
                    orcamdepto float(12,2),
                    PRIMARY KEY (numdepto) );


CREATE TABLE empregado (
                    matricula int (5),
                    nomeemp varchar (20),
                    depto int(4),
                    dataadmin date,
                    salaemp int (4),
                    salario int (2),
                    gerência int (5),
                    PRIMARY KEY (matricula),
                    FOREIGN KEY (depto) REFERENCES departamento (numdepto) );
```

FOREIGN KEY : Chave estrangeira é o campo que estabelece o relacionamento entre duas tabelas. Assim, uma coluna, ou grupo de colunas, de uma tabela corresponde à mesma coluna, ou grupo de colunas, que é a chave primária de outra tabela.

FOREIGN KEY nome-chave-estrangeira (lista-de-colunas)
REFERENCES nome-tabela (lista-de-colunas)
ON UPDATE ação
ON DELETE ação

nome-chave-estrangeira Nome opcional da constraint
lista-de-colunas Lista de colunas da tabela que faz referência a outra tabela.
nome-tabela Nome da tabela em que está a chave primária.
Ação Determina qual ação o banco de dados deve tomar quando for excluída(DELETE) ou alterada(UPDATE) uma linha da tabela que contém referência a esta chave. Pode ser:

SET NULL (altera o conteúdo da coluna para Nulo, perdendo a referência, sem deixar valores inconsistentes).
SET DEFAULT (altera o conteúdo da coluna para o valor especificado na cláusula DEFAULT se houver).
CASCADE (exclui ou altera todos os registros que se relacionam a eles).
NO ACTION (em caso de alteração, não modifica os valores que se relacionam a eles).
RESTRICT (não permite a exclusão da chave primária).

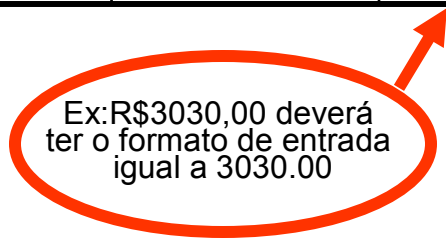
departamento

Ex:R$1.000,00 deverá ter o formato de entrada igual a 1000.00

num-depto	nome-depto	local-depto	orcam-depto
0001	Contabilidade	Bloco 12	1.000.000,00
0002	Informática	Bloco 19	2.000.000,00
0003	Marketing	Bloco 11	1.030.467,14
0004	Recursos Humanos	Bloco 13	900,01

empregado

Matricula	nome-emp	depto	data-admin	sala-emp	salario	gerência
112	Rafael	0001	1999-03-12	18	100.000,00	097
321	Maria	0004	2001-09-20	328	20.800,19	665
456	Ana	0002	2000-12-28	23	10.300,00	345
863	João	0001	1999-05-18	129	20.000,00	256
097	Carlos	0002	1992-10-20	11	20.000,00	321
171	Edurado	0003	2006-02-11	112	3.030,00	112



Responda:

1) Seleção de todas os campos (ou colunas) da tabela de Departamentos.

2) Selecione todos os departamentos cujo orçamento seja maior que 100000. Apresente o Nome de tal departamento e seu orçamento mensal, que será obtido dividindo-se o orçamento por 12.

3) Apresente a instrução anterior porém ao invés dos "feios" nome-depto e orcam-depto, os Títulos : Departamento e Orçamento.

4) Apresente todos os salários existentes na empresa, porém omita eventuais duplicidades.

5) Apresente a matrícula de todos so funcionários com mais de um ano de empresa.

COMANDO SELECT

O comando SELECT seleciona dados desejados em uma ou mais tabelas do banco de dados.

SINTAXE:

SELECT [DISTINCT | DISTINCTROW | ALL] <expressão>
[INTO {OUTFILE | DUMPFILE} 'nomedoarquivo' Opções de Exportação]
[FROM <tabela>]
[WHERE <condição>]
[GROUPY BY <condição de agrupamento>] [ORDER BY <condição de ordenação>]
[HAVING<condição>]
[ORDER BY {<atributo> | formula} [ASC | DESC], ...]
[LIMIT ROWS]

Selecionando todos os registros da tabela

EX:

```
mysql> SELECT *
    FROM carros;

+---------+---------+---------+------------+------------+
| placa   | marca   | modelo  | compra     | venda      |
+---------+---------+---------+------------+------------+
| LNC0211 | Fiat    | Marea   | 2002-08-13 | NULL       |
| LTP0343 | Ford    | Ranger  | 2000-05-14 | NULL       |
| LNC3512 | Gurgel  | Carajas | 1999-07-15 | 2002-02-19 |
| XYS8888 | Fiat    | Uno     | 1998-11-16 | NULL       |
| XAB7788 | Peugeot | 206     | 2003-04-17 | 2006-12-29 |
+---------+---------+---------+------------+------------+
5 rows in set (0.00 sec)
```

Trabalhando com Valores Nulos (IS NULL, IS NOT NULL)

EX:
Selecionando apenas carros onde a venda = NULL

```
mysql> SELECT *
    FROM carros
    WHERE venda IS NULL;

+---------+---------+---------+------------+------------+
| placa   | marca   | modelo  | compra     | venda      |
+---------+---------+---------+------------+------------+
| LNC0211 | Fiat    | Marea   | 2002-08-13 | NULL       |
| LTP0343 | Ford    | Ranger  | 2000-05-14 | NULL       |
| XYS8888 | Fiat    | Uno     | 1998-11-16 | NULL       |
+---------+---------+---------+------------+------------+
3 rows in set (0.00 sec)
```

Selecionando registros que atendam determinada condição

EX:

```
mysql> SELECT *
    FROM carros
    WHERE marca='fiat';

+---------+---------+---------+------------+------------+
| placa   | marca   | modelo  | compra     | venda      |
+---------+---------+---------+------------+------------+
| LNC0211 | Fiat    | Marea   | 2002-08-13 | NULL       |
| XYS8888 | Fiat    | Uno     | 1998-11-16 | NULL       |
+---------+---------+---------+------------+------------+
2 rows in set (0.00 sec)
```

Selecionando registros limitando apenas os 3 primeiros

EX:

```
mysql> SELECT *
    FROM carros
    limit 3;

+---------+---------+---------+------------+------------+
| placa   | marca   | modelo  | compra     | venda      |
+---------+---------+---------+------------+------------+
| LNC0211 | Fiat    | Marea   | 2002-08-13 | NULL       |
| LTP0343 | Ford    | Ranger  | 2000-05-14 | NULL       |
| LNC3512 | Gurgel  | Carajas | 1999-07-15 | 2002-02-19 |
+---------+---------+---------+------------+------------+
3 rows in set (0.00 sec)
```

Selecionando registros limitando apenas a mostrar 2 registros a partir do 3

EX:

```
mysql> SELECT *
    FROM carros
    limit 3,2;

+---------+---------+---------+------------+------------+
| placa   | marca   | modelo  | compra     | venda      |
+---------+---------+---------+------------+------------+
| XYS8888 | Fiat    | Uno     | 1998-11-16 | NULL       |
| XAB7788 | Peugeot | 206     | 2003-04-17 | 2006-12-29 |
+---------+---------+---------+------------+------------+
2 rows in set (0.00 sec)
```

Selecionando colunas específicas

EX:

```
mysql> SELECT placa,compra
       FROM carros
       WHERE marca='Fiat';
```

```
+---------+---------+--
| placa   |  compra   |
+---------+---------+
| LNC0211 | 2002-08-13 |
| XYS8888 | 1998-11-16 |
+---------+---------+
2 rows in set (0.00 sec)
```

Exportando Dados para Arquivo texto

```
mysql> select * into outfile 'dados.txt'
    -> fields terminated by ','
    -> optionally enclosed by ' " '
    -> lines terminated by '\n'
    -> from produtos;
```

Comentado quando foi apresentado o comando LOAD DATA no arquivo notas 2.

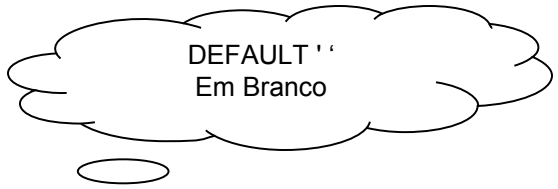
O resultado do comando select irá ser gerado para um arquivo chamado dados.txt no diretório do database (ex: c:\mysql\data\aula\dados.txt)

Odenado a Saída dos Registros

Para ordenar o resultado, utilize uma cláusula ORDER BY.

mysql> SELECT * FROM carros ORDER BY compra DESC;

```
+-------------+------------+---------+-----------------+--------------+
| placa       |Marca       | modelo| Compra          | Venda        |
+-------------+------------+---------+-----------------+--------------+
|  LNC3512 | Peugeot  |206       |2003-04-17   | 2006-12-29  |
| LNC0211  | fiat       |Marea   | 2002-08-13  |   NULL       |
|  LTP0343  | Ford       |Ranger  | 2000-05-14  |   NULL       |
|  LNC3512 | Gurgel     |Carajás |1999-07-15   |  2002-02-19 |
|  XYS8888 | fiat       | Uno      | 1998-11-16  |   NULL       |
+-------------+------------+---------+-----------------+--------------+
```




```
mysql> CREATE TABLE shop (
-> artigo INT(4) UNSIGNED ZEROFILL DEFAULT '0000' NOT NULL,
-> categoria CHAR(20) DEFAULT ' ' NOT NULL,
-> preco DOUBLE(16,2) DEFAULT '0.00' NOT NULL,
-> PRIMARY KEY(artigo, categoria));
```

UNSIGNED - *significa que o valor só pode ter valor positivo ou zero. Usado com dados do tipo inteiro.*

ZEROFILL – *Completa os valores a esquerda do número com zeros.*
Ex: A entrada no campo artigo: 4 com ZEROFILL seria 0004 (automaticamente)
O uso do ZEROFILL torna o campo UNSIGNED

```
mysql> INSERT INTO shop VALUES
-> (1,'A',3.45),(1,'B',3.99),(2,'A',10.99),(3,'B',1.45),(3,'C',1.69),
-> (3,'D',1.25),(4,'D',19.95);
```

Selecionando o maior Valor de uma coluna

```
SELECT  MAX(artigo) AS maior  FROM shop;

+---------+
| maior |
+---------+
|      4    |
+---------+
```

Como obter o artigo de maior valor ?

No SQL ANSI isto é feito facilmente com uma sub-consulta:

```
SELECT artigo, categoria, preco
FROM shop
WHERE preco=(SELECT MAX(preco) FROM shop);
```

MIN ( ) -> Retorna o valor mínimo da coluna.

Todas as demonstrações realizadas com o MAX também são válidas para o MIN()

SELECT MIN(artigo) AS menor FROM shop;

```
+--------+
| menor |
+--------+
|      1    |
+--------+
```

Para limitar o número de linhas da pesquisa use a cláusula limit:

```
mysql> select * from cliente;
+--------+---------+
| codigo | nome    |
+--------+---------+
|      1 | João    |
|      2 | Maria   |
|      3 | José    |
|      4 | Manuel  |
|      5 | Adão    |
|      6 | Rodrigo |
|      7 | Davi    |
|      8 | Karla   |
|      9 | Samuel  |
|     10 | Ana     |
+--------+---------+
```

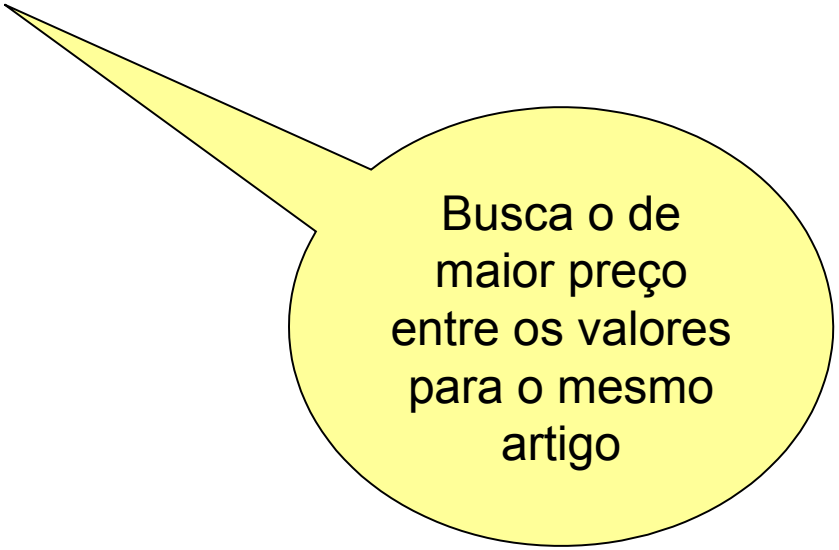
```
mysql> select * from cliente limit 2,4;
+--------+---------+
| codigo | nome    |
+--------+---------+
|      3 | José    |
|      4 | Manuel  |
|      5 | Adão    |
|      6 | Rodrigo |
+--------+---------+
```

limit 2,4 diz ao mysql para retornar quatro linhas a partir da linha 2, exclusive.

Máximo da Coluna por Grupo

```
SELECT artigo, MAX(preco) AS preco
FROM shop
GROUP BY artigo
```

```
+---------+-------+
| artigo | preco |
+---------+-------+
| 0001 | 3.99    |
| 0002 | 10.99  |
| 0003 | 1.69    |
| 0004 | 19.95  |
+---------+-------+
```



AVG(coluna) -> retorna a média dos valores da coluna

```
SELECT avg(preco) AS preco
FROM shop
```

```
+---------+
 | preco |
+---------+
| 9.155   |
+---------+
```

Outras Funções

•SUM(coluna)retorna a soma dos valores da coluna

• COUNT(item) -> se *item* for uma coluna, será retornado o número de valores não NULL nesta coluna.

•*OBS:*
•*Se a palavra-chave DISTINCT for colocada na frente do nome da coluna, será retornado o número de valores distintos nesta coluna.*
Se for passado COUNT(*), será retornado o número total de registros independente de quantos tenham valor NULL.

EXERCÍCIO: Após criar a tabela pedido, execute os comandos abaixo em seu computador.

Tabela: Pedido

```
+----+---------+--------+
| nr | cliente | valor  |
+----+---------+--------+
|  1 |       2 |  50.50 |
|  2 |       2 |  60.00 |
|  3 |       1 | 100.05 |
|  4 |       3 |  54.70 |
|  5 |       3 |  80.90 |
+----+---------+--------+
```

```
mysql> select avg(valor) from pedido;
+------------+
| avg(valor) |
+------------+
|  69.230001 |
+------------+
```

```
mysql> select cliente,avg(valor) from pedido group by cliente;
+---------+------------+
| cliente | avg(valor) |
+---------+------------+
|       1 | 100.050003 |
|       2 |  55.250000 |
|       3 |  67.800001 |
+---------+------------+
```

Observe que quando usamos group by a função avg() retorna a média calculada para cada componente do grupo especificado, no caso "cliente".

```
mysql> select min(valor) from pedido;
+------------+
| min(valor) |
+------------+
|      50.50 |
+------------+
mysql> select max(valor) from pedido;
+------------+
| max(valor) |
+------------+
|     100.05 |
+------------+
```

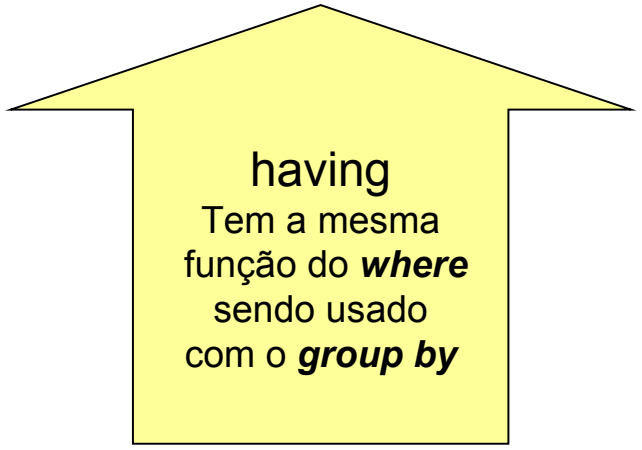
```
mysql> select sum(valor) from pedido;
```

```
+------------+
| sum(valor) |
+------------+
|     346.15 |
+------------+
```

Podemos também testar o valor retornado por avg():

```
mysql> select cliente,avg(valor) from pedido group by cliente having avg(valor) > 60;
```

```
+---------+------------+
| cliente | avg(valor) |
+---------+------------+
|       1 | 100.050003 |
|       3 |  67.800001 |
+---------+------------+
```



Crie a Tabela Abaixo:

```
CREATE TABLE pet (
        nome VARCHAR(20),
        owner VARCHAR(20),
        species VARCHAR(20),
        sex CHAR(1),
        birth DATE,
        death DATE);
```

name	owner	species	sex	birth	death
Fluffy	Harold	cat	f	1993-02-04	NULL
Claws	Gwen	cat	m	1994-03-17	NULL
Buffy	Harold	dog	f	1989-05-13	NULL
Fang	Benny	dog	m	1990-08-27	NULL
Bowser	Diane	dog	m	1979-08-31	1995-07-29
Chirpy	Gwen	bird	f	1998-09-11	NULL
Whistler	Gwen	bird	NULL	1997-12-09	NULL
Slim	Benny	snake	m	1996-04-29	NULL
Puffball	Diane	hamster	f	1999-03-30	NULL

A combinação de padrões SQL lhe permite você usar _ para coincidir qualquer caractere simples e % para coincidir um número arbitrário de caracteres (incluindo zero caracter).

Para encontrar nomes começando com ‘b’:

```
mysql> SELECT * FROM pet WHERE name LIKE "b%";

+---------+-----------+---------+-------+--------------+----------------+
| name    | owner     | species | sex   | birth        | death          |
+---------+-----------+---------+-------+--------------+----------------+
| Buffy   | Harold    | dog     | f     | 1989-05-13 | NULL           |
| Bowser | Diane     | dog     | m     | 1979-08-31 | 1995-07-29 |
+---------+-----------+---------+-------+--------------+----------------+
```

Para encontrar nomes com o final ‘fy’:

```
mysql> SELECT * FROM pet WHERE name LIKE "%fy";

+---------+-----------+---------+-------+--------------+----------------+
| name    | owner     | species | sex   | birth        | death          |
+---------+-----------+---------+-------+--------------+----------------+
| Fluffy  | Harold    | cat     | f     | 1993-02-04 | NULL           |
| Buffy   | Harold    | dog     | f     | 1989-05-13 | NULL           |
+---------+-----------+---------+-------+--------------+----------------+
```

Para encontrar nomes contendo um ‘w’:

```
mysql> SELECT * FROM pet WHERE name LIKE "%w%";

+----------+-----------+---------+-------+--------------+----------------+
| name     | owner     | species | sex   | birth        | death          |
+----------+-----------+---------+-------+--------------+----------------+
| Claws    | Gwen      | cat     | m     | 1994-03-17 | NULL           |
| Bowser | Diane     | dog     | m     | 1979-08-31 | 1995-07-29 |
| Whistler | Gwen    | bird    | NULL | 1997-12-09 | NULL           |
+----------+-----------+---------+-------+--------------+----------------+
```

Para encontrar nomes contendo exatamente cinco caracteres, use cinco instâncias do caracter ‘_’:

```
mysql> SELECT * FROM pet WHERE name LIKE "_ _ _ _ _";

+----------+-----------+---------+-------+--------------+----------------+
| name     | owner     | species | sex   | birth        | death          |
+----------+-----------+---------+-------+--------------+----------------+
| Claws    | Gwen      | cat     | m     | 1994-03-17 | NULL           |
| Buffy    | Harold    | dog     | f     | 1989-05-13 | NULL           |
+----------+-----------+---------+-------+--------------+----------------+
```

BETWEEN

Serve para determinar um intrvalo de busca. Muito utilizado para simplificar a utilização do AND

mysql> SELECT * FROM pet WHERE birth BETWEEN ‘1993-02-04’ AND ‘1998-09-11’ ;

```
+----------+------------+----------+--------+--------------+-----------------+
| name     | owner      | species | sex    | birth        | death           |
+----------+------------+----------+--------+--------------+-----------------+
| Fluffy   | Harold     | cat      | f      | 1993-02-04 | NULL            |
| Claws    | Gwen       | cat      | m      | 1994-03-17 | NULL            |
| Chirpy   | Gwen       | bird     | f      | 1998-09-11 | NULL            |
| Whistler | Gwen       | bird     | NULL   | 1997-12-09 | NULL            |
| Slim     | Benny      | snake    | m      | 1996-04-29 | NULL            |
+----------+------------+----------+--------+--------------+-----------------+
```

IN

Serve para comparar valores de uma coluna com um conjunto de valores válidos. Geralmente, utilizamos o IN para substituir uma série de comparações seguidas por OR

mysql> SELECT * FROM pet WHERE owner IN (‘Gwen’ , ‘Benny’) ;

name	owner	species	sex	birth	death
Claws	Gwen	cat	m	1994-03-17	NULL
Fang	Benny	dog	m	1990-08-27	NULL
Chirpy	Gwen	bird	f	1998-09-11	NULL
Whistler	Gwen	bird	NULL	1997-12-09	NULL
Slim	Benny	snake	m	1996-04-29	NULL

O outro tipo de combinação de padrões fornecido pelo MySQL usa expressões regulares extendidas. Quando você testa por uma combinação para este tipo de padrão, utilize os operadores REGEXP e NOT REGEXP (ou RLIKE e NOT RLIKE, que são sinônimos).

‘.’ combina qualquer caractere único

Uma classe de caracteres ‘[...]’ combina qualquer caractere que consta dentro dos colchetes. Por exemplo, ‘[abc]’ combina com ‘a’, ‘b’, ou ‘c’. Para nomear uma sequência de caracteres utilize um traço. ‘[a-z]’ combina com qualquer letra e ‘[0-9]’ combina com qualquer dígito.

‘*’ combina com nenhuma ou mais instâncias de sua precedência. Por exemplo, ‘x*’ combina com qualquer número de caracteres ‘x’, ‘[0-9]*’ combina com qualquer número de dígitos e ‘.*’ combina com qualquer número de qualquer coisa.

Um padrão REGEXP casa com sucesso se ele ocorre em algum lugar no valor sendo testado. (Ele difere do padrão LIKE, que só obtem sucesso se eles combinarem com todo o valor.)

Para fazer com que um padrão deva combinar com o começo ou o fim de um valor sendo testado, utilize '^' no começo ou '$' no final do padrão.

Para encontrar nomes começando com 'b', utilize '^' para combinar com o começo do nome:

```
mysql> SELECT * FROM pet WHERE name REGEXP "^b";

+---------+-----------+---------+-------+--------------+----------------+
| name    | owner     | species | sex   | birth        | death          |
+---------+-----------+---------+-------+--------------+----------------+
| Buffy   | Harold    | dog     | f     | 1989-05-13 | NULL           |
| Bowser | Diane     | dog     | m     | 1979-08-31 | 1995-07-29 |
+---------+-----------+---------+-------+--------------+----------------+
```

Para encontrar nomes finalizados com ‘fy’, utilize ‘$’ para combinar com o final do nome:

```
mysql> SELECT * FROM pet WHERE name REGEXP "fy$";

+---------+----------+---------+-------+--------------+----------------+
| name    | owner    | species | sex   | birth        | death          |
+---------+----------+---------+-------+--------------+----------------+
| Fluffy  | Harold   | cat     | f     | 1993-02-04 | NULL           |
| Buffy   | Harold   | dog     | f     | 1989-05-13 | NULL           |
+---------+----------+---------+-------+--------------+----------------+
```

Para encontrar nomes contendo um ‘w’, utilize esta consulta:

```
mysql> SELECT * FROM pet WHERE name REGEXP "w";

+---------+----------+---------+-------+--------------+----------------+
| name    | owner    | species | sex   | birth        | death          |
+---------+----------+---------+-------+--------------+----------------+
| Claws   | Gwen     | cat     | m     | 1994-03-17 | NULL           |
| Bowser  | Diane    | dog     | m     | 1979-08-31 | 1995-07-29     |
| Whistler | Gwen    | bird    | NULL  | 1997-12-09 | NULL           |
+---------+----------+---------+-------+--------------+----------------+
```

Para encontrar nomes contendo exatamente cinco caracteres, utilize ‘^’ e ‘$’ para combinar com o começo e fim do nome e cinco instâncias de ‘.’ entre eles.

```
mysql> SELECT * FROM pet WHERE name REGEXP "^.....$";

+----------+-----------+---------+-------+--------------+-----------------+
| name     | owner     | species | sex   | birth        | death           |
+----------+-----------+---------+-------+--------------+-----------------+
| Claws    | Gwen      | cat     | m     | 1994-03-17 | NULL            |
| Buffy    | Harold    | dog     | f     | 1989-05-13 | NULL            |
+----------+-----------+---------+-------+--------------+-----------------+
```

OBS:

Você pode também escrever a consulta anterior utilizando o operador ‘{n}’ “repete-n-vezes”:

```
mysql> SELECT * FROM pet WHERE name REGEXP "^.{5}$";
```