

CURSO TÉCNICO DESENVOLVIMENTO DE SISTEMAS

MÓDULO II

BANCO DE DADOS

PROFESSOR: Rubem Cândido
MATERIAL COMPILADO

Belo Horizonte/MG
2021

Sumário

1. Introdução a Banco de Dados	5
2. História	5
3. Conceitos Básicos	5
4. Vantagens da utilização Banco de Dados.....	6
5. O problema de compartilhamento de dados!.....	7
6. Sistemas Isolados	8
7. Compartilhamento de Dados	8
8. Modelo de Banco de Dados	8
9. Modelo Conceitual	9
10. Modelo Lógico	9
11. Modelo de Organização	10
12. Entidade	11
13. Relacionamentos.....	11
14. Diagrama de ocorrências.....	12
15. Auto-relacionamento	12
16. Cardinalidade de relacionamentos	13
17. Cardinalidade máxima.....	13
18. Classificação de relacionamentos binários	14
19. Relacionamento ternário	15
20. Cardinalidade mínima	16
21. Atributos.....	18
22. Cardinalidade de atributos.....	19
23. Atributo de relacionamento.....	19
24. Identificando entidades	19
25. Entidades fracas!	20
26. Identificando relacionamentos	21
27. Generalização especialização.....	21
28. Generalização/Especialização Total	22
29. Generalização/Especialização Parcial.....	22

30.	Herança múltipla	23
31.	Entidade Associativa	23
32.	Entidade Associativa (equivalência)	24
33.	Esquemas gráficos e textuais	25
34.	Esquemas gráficos e textuais	26
35.	Modelo formal.....	28
36.	Expressividade limitada.....	28
37.	Equivalência entre modelos	29
38.	Identificando construções.....	30
39.	Atributo X entidade relacionada	30
40.	Atributos X Generalização/especialização	30
41.	Atributos opcionais e multi-valorados	31
42.	Atributos opcionais	31
43.	Atributo multi-valorado	32
44.	Verificação do modelo	32
45.	Modelo Correto.....	33
46.	Modelo completo.....	33
47.	Modelo livre de redundâncias.....	33
48.	Aspectos temporais.....	34
49.	Estabelecendo padrões	35
50.	Variações do modelo ER.....	35
51.	Linguagens do Banco de Dados.....	38
52.	Criação e Utilização de um Banco de Dados	40
53.	Criando e Selecionando um Banco de Dados.....	41
54.	Criando uma Tabela	42
55.	Alterando a estrutura da tabela com o Alter Table	44
56.	Recuperando Informações de uma Tabela	45
57.	Selecionando Todos os Dados.....	46
58.	Selecionando Registros Específicos.....	47
59.	Selecionando Colunas Específicas	49
60.	Ordenando Registros.....	52
61.	Cálculo de Datas	54
62.	Combinação de padrões.....	56
63.	Contando Registros	58

64.	Utilizando Múltiplas Tabelas	59
65.	Exemplos de Consultas Comuns.....	62
66.	O Valor Máximo para uma Coluna	63
67.	O Registro que Armazena o Valor Máximo para uma Coluna Determinada	63
68.	Calculando o Menor Valor De Uma Coluna.....	64
69.	Calculando a Média.....	64
70.	Somando Os Valores De Uma Coluna	65
71.	\$POST	65
72.	Conectando no banco	65
73.	Acho que é selecionando o banco	66
74.	DCL Controle.....	66
75.	Bibliografias	70

1. Introdução a Banco de Dados

O que é Banco de Dados?



Banco de dados: conjunto de dados integrados que tem por objetivo atender a uma comunidade de usuários

“Um banco de dados é uma coleção de dados relacionados.” (ELMASRI, NAVATHE, 2004)



Representa abstratamente uma parte do mundo real, conhecida como Mini-Mundo ou Universo de Discurso (UD), que é de interesse de uma certa aplicação;

2. História

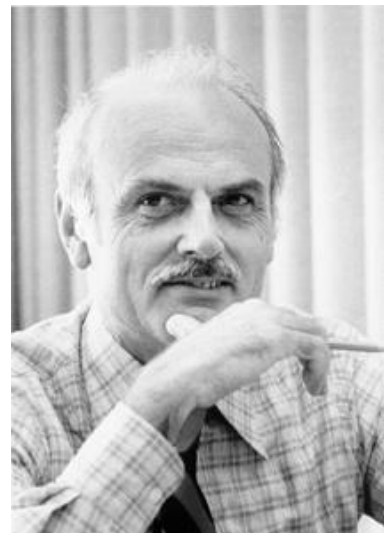
Introduzido pelo matemático Edgard (Ted) Codd, da IBM Research, em 1970.

Simplicidade e base matemática.

“O modelo relacional representa o banco de dados como uma coleção de relações.” (ELMASRI, NAVATHE, 2004).

Estruturas

- Relações (tabelas), atributos, tuplas (linhas)
- Chaves
- Atributos especiais: chave estrangeira
- Relacionamentos
- Normalização
- Linguagem de definição de dados
- Restrições



3. Conceitos Básicos

Dados: Dados são códigos que constituem a matéria prima da informação, ou seja, é a informação não tratada, porém não podem transmitir uma mensagem ou representar algum conhecimento.

Exemplo: Verde, 1984,

Informação: São dados tratados. O resultado do processamento de dados são as informações.

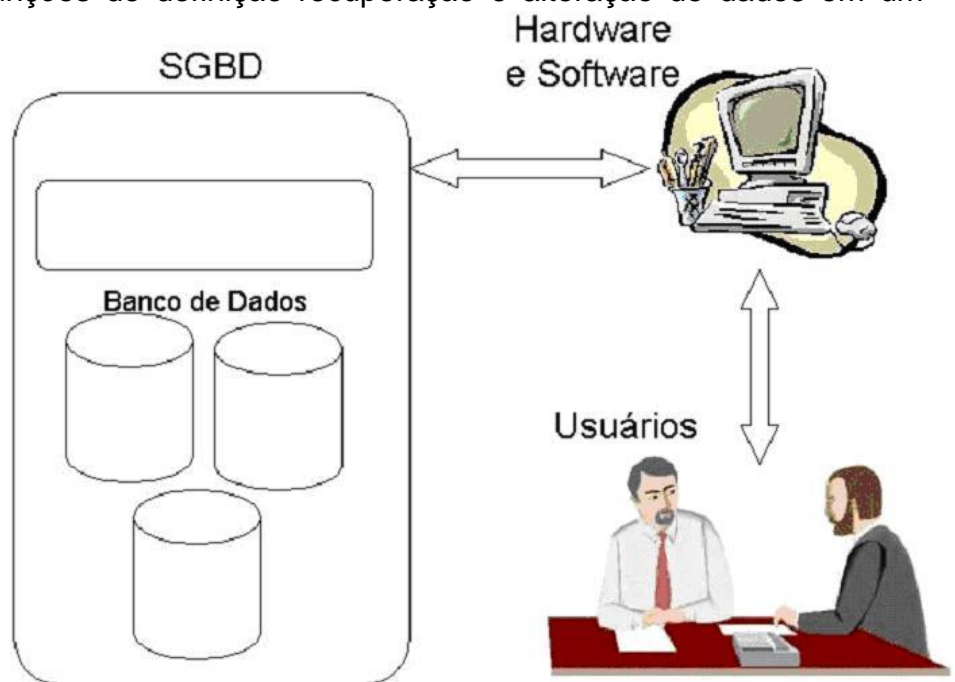
Exemplo: O lápis é **verde**, ela nasceu em **1984**.

Metadados: São dados sobre outros dados. Também chamado de **Metainformação**.

Exemplo: VARCHAR(60), INT, DATE.

SGBD: Sistema gerenciador de banco de dados = Software construído para facilitar as atividades de definição, construção e manipulação de banco de dados. Software que incorpora as funções de definição recuperação e alteração de dados em um banco de dados.

Exemplos: Oracle,
SQL-Server, DB2,
MySQL,
Postgresql,
Firebird.



4. Vantagens da utilização Banco de Dados

- Minimiza inconsistência e redundância de dados.
- Facilita acesso aos dados.

Consultas genéricas, como a SQL (Structured Query Language).

- Atomicidade.

A transação deve ter todas as suas operações e executadas em caso de sucesso ou nenhum resultado de alguma operação refletido sobre a base de dados em caso de falha.

- Acesso concorrente.

Vários usuários acessando os mesmos dados.

- Segurança

Definição de regras para os diferentes tipos de usuários.

Exercícios:

1 - Descreva com suas palavras:

a) Qual a diferença entre dados e informação;

b) O que é banco de dados?

c) O que é um SGBD?

2 - Cite três SGBD'S.

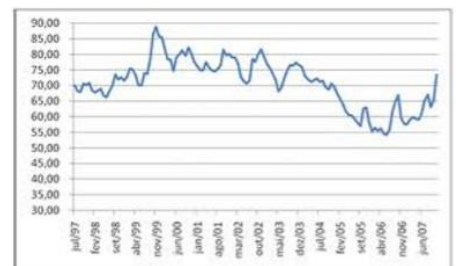
3 – Pesquisar sobre MySQL. Instituições, empresas, sites que usam e o que dizem sobre esse SGBD.

5. O problema de compartilhamento de dados!

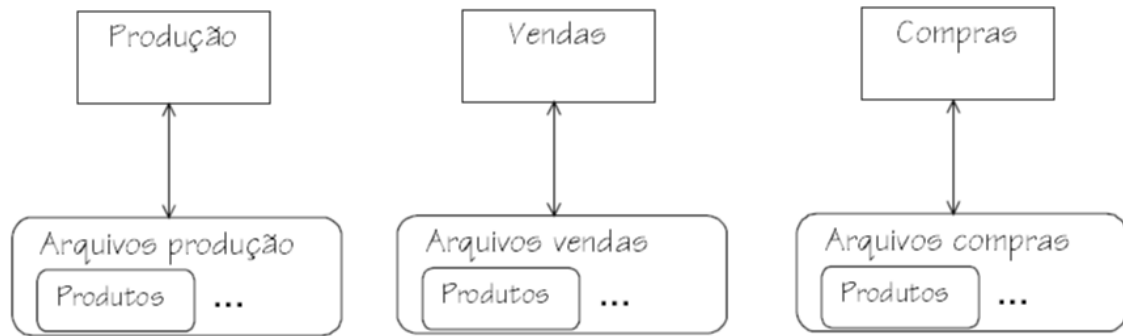
Informatização gradativa em uma organização de acordo com as funções abaixo:

- VENDAS
- PRODUÇÃO
- COMPRAS

PRODUTOS

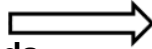


6. Sistemas Isolados



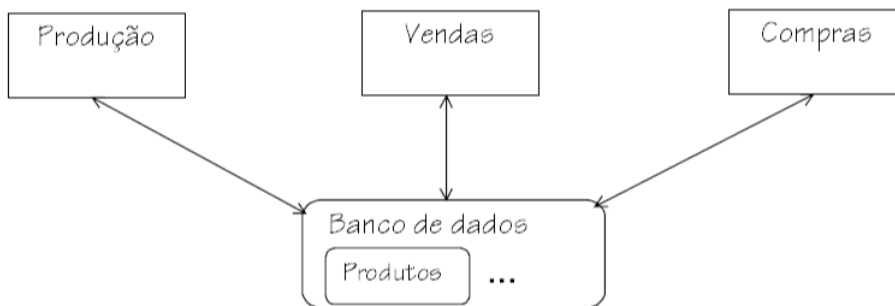
Redundância de dados:

- ❖ Redundância controlada
- ❖ Redundância não controlada



**Redigitação e
Inconsistência de
dados**

7. Compartilhamento de Dados



O compartilhamento de dados tem reflexos na estrutura do software:

A estrutura dos arquivos passa a ser mais complexa, pois estes devem ser construídos de forma a atender às necessidades dos diferentes sistemas.

8. Modelo de Banco de Dados

Um modelo de dados é uma descrição dos tipos de informação que estão armazenadas em um banco de dados.

- ❖ Linguagens de modelagem (Textuais e Gráficas)

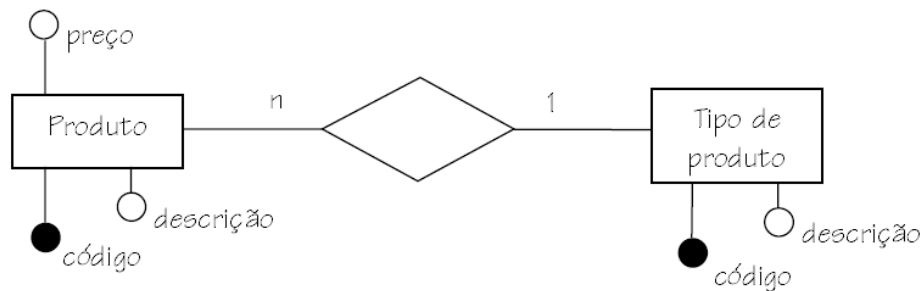
- ❖ O mesmo modelo de dados pode ser apresentado de várias formas
- ❖ Cada apresentação do modelo é um “Esquema de banco de Dados”

Níveis de abstração para descrição de um banco de dados

9. Modelo Conceitual

Um modelo conceitual é uma descrição do banco de dados de forma independente de implementação em um SGBD. O modelo conceitual registra que dados podem aparecer no banco de dados, mas não registra como estes dados estão armazenados a nível de SGBD

A técnica Entidade-Relacionamento (ER) é a mais difundida e é representada pelo Diagrama Entidade-Relacionamento (DER) ou (MER) modelo entidade-relacionamento.



10. Modelo Lógico

Um modelo lógico é uma descrição de um banco de dados no nível de abstração visto pelo usuário do SGBD. Assim, o modelo lógico é dependente do tipo particular de SGBD que está sendo usado.

TipoDeProduto(CodTipoProd,DescrTipoProd)
 Produto(CodProd,DescrProd,PrecoProd,CodTipoProd)
 CodTipoProd referencia TipoDeProduto]

TipoDeProduto			
TipoDeProduto		CodTipoProd	DescrTipoProd
		1	Computador
		2	Impressora

Produto			
CodProd	DescrProd	PrecoProd	CodTipoProd
1	PC desktop modelo X	2.500	1
2	PC notebook ABC	3.500	1
3	Impressora jato de tinta	600	2
4	Impressora laser	800	2

11.Modelo de Organização

Uma das ideias fundamentais do projeto de banco de dados: a de que através da identificação das entidades que terão informações representadas no banco de dados, é possível identificar os arquivos que comporão o banco de dados, ou seja: um mesmo modelo conceitual pode ser usado em duas funções:

1. Como modelo abstrato da organização

- Define as entidades da organização que tem informações armazenadas no banco de dados

2. Como modelo abstrato do banco de dados

- Define que arquivos farão parte do banco de dados

Maior envolvimento do usuário na especificação do banco de dados

Exercícios

- 1 - Defina, os seguintes conceitos: banco de dados, sistema de gerência de banco de dados, modelo conceitual, modelo lógico.
- 2 - A definição do tipo de um dado (numérico, alfanumérico,...) faz parte de que modelo: do modelo conceitual, do modelo lógico ou do modelo físico?
- 3 - Qual a diferença entre a redundância de dados controlada e a redundância de dados não controlada? Dê exemplos de cada uma delas.

12. Entidade

Entidade é um conjunto de objetos da realidade modelada sobre os quais deseja-se manter informações no banco de dados

Exemplo anterior: produtos, tipos de produtos, vendas ou as compras etc.

OBS. Uma entidade pode representar objetos concretos (veículo) como abstratos (departamento, endereço)

Representação



Cada retângulo representa um conjunto de objetos sobre os quais deseja-se guardar informações.

OBS. Caso seja necessário referir a um objeto em particular (uma determinada pessoa ou um determinado departamento) fala-se em ocorrência de entidade ou instância.

13. Relacionamentos

Relacionamento = conjunto de associações entre entidades

OBS. Além de especificar os objetos sobre os quais deseja-se manter informação, o DER deve permitir a especificação das propriedades dos objetos que serão armazenados no BD.

- ❖ Associação entre objetos, por exemplo: quais pessoas estão associadas a quais departamentos.

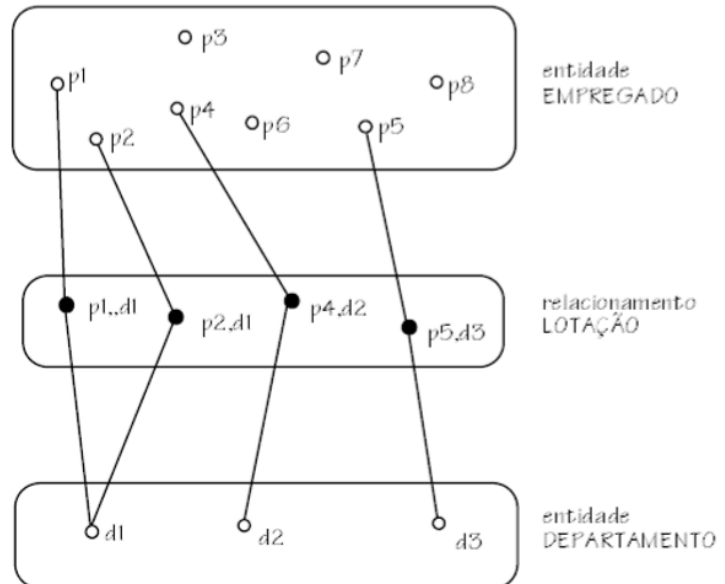


Em um DER um relacionamento é representado por um losango ligado por linhas aos retângulos das entidades relacionadas. Neste exemplo temos duas entidades (DEPARTAMENTO e PESSOA) e um relacionamento (LOTAÇÃO).

Ocorrência de relacionamento: par específico formado por uma determinada ocorrência da entidade pessoa e por uma determinada ocorrência da entidade departamento.

14. Diagrama de ocorrências

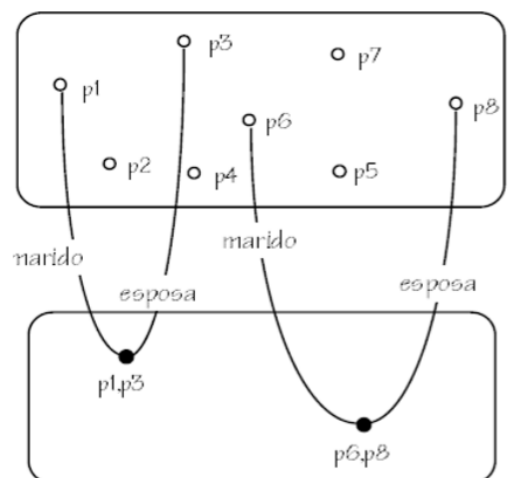
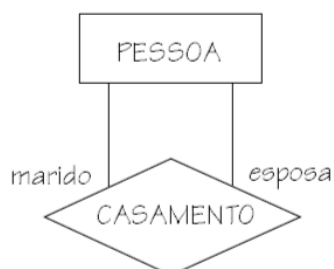
O Diagrama da figura ao lado não informa quantas vezes uma entidade é associada através de um relacionamento!



15. Auto-relacionamento



Um relacionamento entre ocorrências da mesma entidade
Papel da entidade no relacionamento!



Exercício:

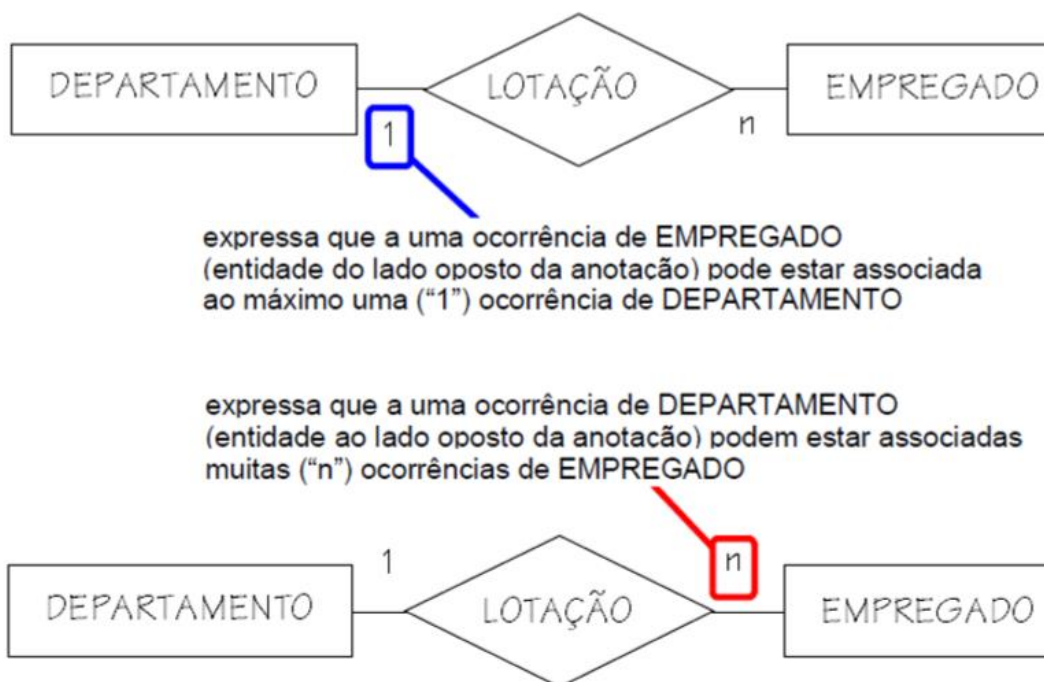
- 1) O que é entidade? Dê exemplos.
- 2) O que é atributo? Dê exemplos.
- 3) O que é relacionamento?
- 4) O que deve compor uma entidade?
- 5) Faça um DER sobre uma loja de animais.

16. Cardinalidade de relacionamentos

Cardinalidade (mínima, máxima) de entidade em relacionamento = número (mínimo, máximo) de ocorrências de entidade associadas a uma ocorrência da entidade em questão através do relacionamento.

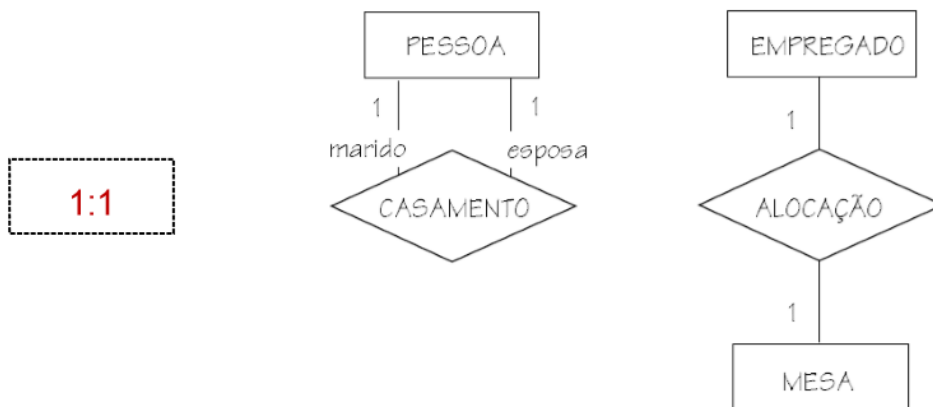
Para fins práticos, não é necessário distinguir entre diferentes cardinalidades máximas maiores que 1. Por este motivo, apenas duas cardinalidades máximas são relevantes: a cardinalidade máxima 1 e a cardinalidade máxima “muitos”, referida pela letra n.

17. Cardinalidade máxima

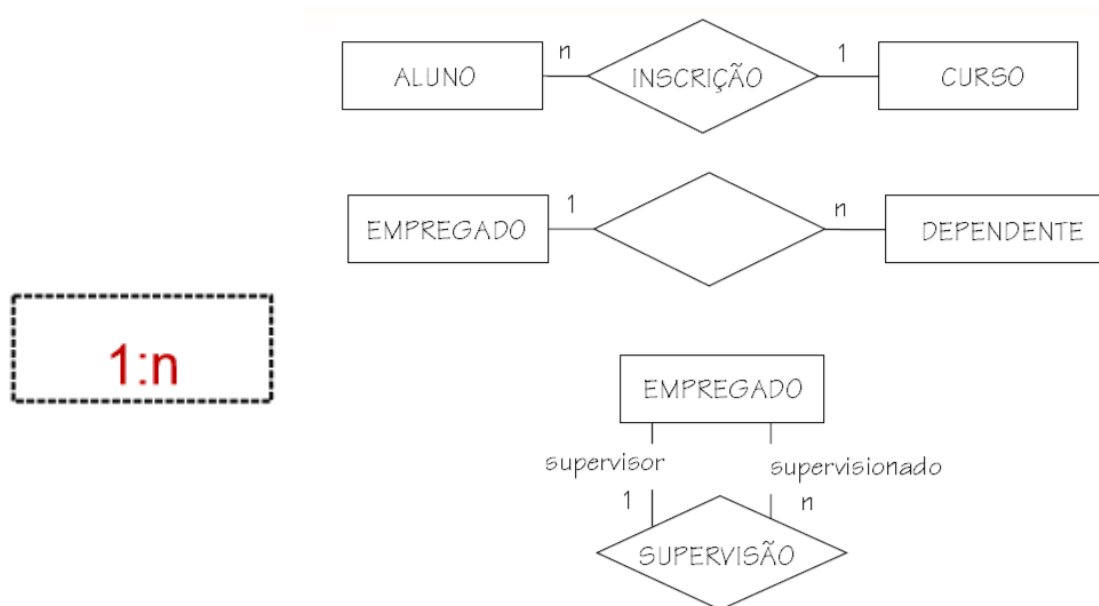


18. Classificação de relacionamentos binários

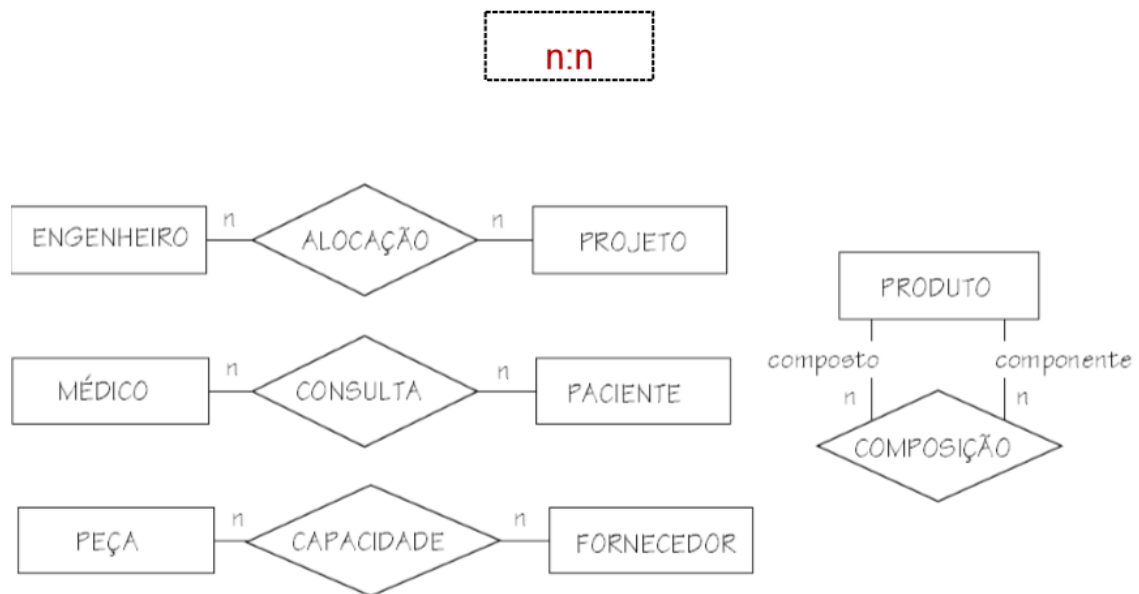
A cardinalidade máxima pode ser usada para classificar relacionamentos binários. Um relacionamento binário é aquele cujas ocorrências envolvem duas entidades, como todos vistos até aqui. Podemos classificar os relacionamentos em n:n (muitos-para-muitos), 1:n (um-para-muitos) e 1:1 (um-para-um).



Mais um exemplo de relacionamento binário



Relacionamento binário



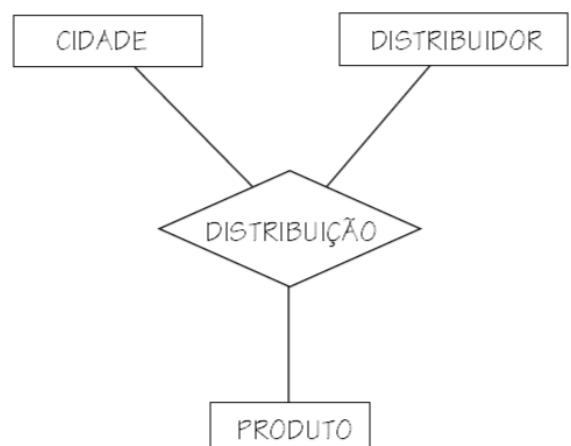
Exercício: 1) Encontre a cardinalidade e dê sua explicação.

- a) Cargo – Empregado
- b) Aluno – Disciplina
- c) Aluno – Sondagem
- d) Cliente – Conta_a_receber
- e) Banco – Agência
- f) Cargo – Profissional
- g) Empresa Matriz – Empresa Filial

19. Relacionamento ternário

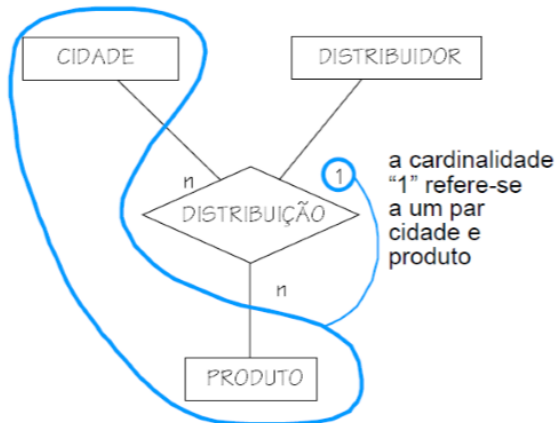
A abordagem ER permite que sejam definidos relacionamentos de grau maior do que dois (relacionamentos ternários, quaternários,...)

Cada ocorrência do relacionamento **DISTRIBUIÇÃO** associa três ocorrências de entidade: um produto



a ser distribuído, uma cidade na qual é feita a distribuição e um distribuidor.

No caso de um relacionamento ternário, a cardinalidade refere-se a pares de entidades. Em um relacionamento R entre três entidades A, B e C, a cardinalidade máxima de A e B dentro de R indica quantas ocorrências de C podem estar associadas a um par de ocorrências de A e B.



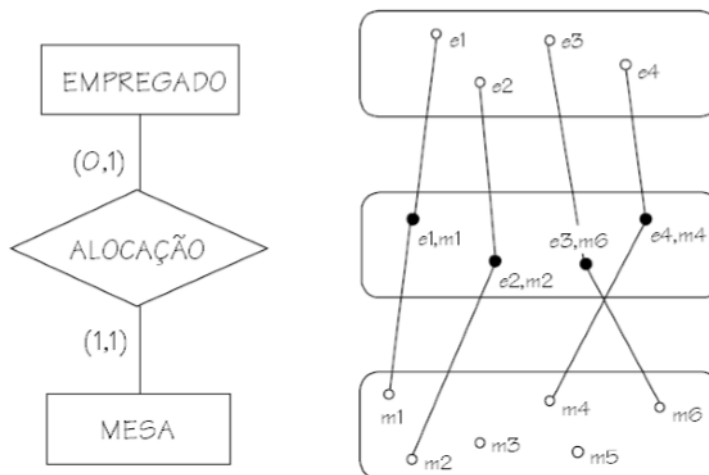
Exemplificando, na figura anterior: o "1" na linha que liga o retângulo representativo da entidade DISTRIBUIDOR ao losango representativo do relacionamento expressa que cada par de ocorrência (cidade, produto) está associado a no máximo um distribuidor. Em outros termos, não há concorrência na distribuição de um produto em uma cidade.

- ❖ A um par (cidade, distribuidor) podem estar associados muitos produtos, ou em outros termos, um distribuidor pode distribuir em uma cidade muitos produtos.
- ❖ A um par (produto, distribuidor) podem estar associadas muitas cidades, ou em outros termos um distribuidor pode distribuir um produto em muitas cidades.

20. Cardinalidade mínima

Consideramos apenas duas cardinalidades mínimas: a cardinalidade mínima 0 e a cardinalidade mínima 1. A cardinalidade mínima 1 também recebe a

denominação de “associação obrigatória”. A cardinalidade mínima 0 também recebe a denominação de “associação opcional”



Exercício 1

Faça um DER de controle acadêmico da escola Fontes para descrever:

Deseja-se manter informações sobre alunos, cursos, disciplinas e departamentos.

Além disso, deseja-se manter informações sobre a associação de alunos a cursos, de disciplinas a cursos, de disciplinas a departamentos, bem como de disciplinas a suas disciplinas pré-requisitos.

Através das cardinalidades expressa-se que:

Uma disciplina pode possuir diversos pré-requisitos, inclusive nenhum. Uma disciplina pode ser pré-requisito de muitas outras disciplinas, inclusive de nenhuma. Uma disciplina pode aparecer no currículo de muitos cursos (inclusive de nenhum) e um curso pode possuir muitas disciplinas em seu currículo (inclusive nenhuma). Um aluno está inscrito em exatamente um curso e um curso pode ter nele inscritos muitos alunos (inclusive nenhum).

Cada disciplina possui exatamente um departamento responsável, e um departamento é responsável por muitas disciplinas, inclusive por nenhuma. Note-se que, apesar de sabermos que os departamentos em uma universidade existem para ser responsáveis por disciplinas, especificamos a cardinalidade mínima de DEPARTAMENTO em RESPONSÁVEL como sendo “0”. Com isso admitimos a possibilidade de existirem departamentos vazios. Esta cardinalidade foi especificada considerando o estado do banco de dados imediatamente após a criação de um

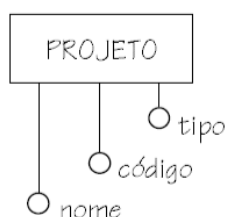
novo departamento, bem como o estado imediatamente antes da eliminação de um departamento. Da forma como a restrição foi especificada, é possível incluir o departamento em uma transação, para, depois, em transações subsequentes, vinculá-lo às disciplinas sob sua responsabilidade. Se tivesse sido especificada a cardinalidade mínima “1”, ao menos uma disciplina teria que ser vinculada ao departamento já na própria transação de inclusão do departamento. Como observa-se da discussão acima, para especificar as cardinalidades mínimas é necessário possuir conhecimento sobre as transações de inclusão e exclusão das entidades.

Exercício 2

Negócio - Locadora de vídeos. Uma pequena locadora de vídeos possui cerca de 2.000 fitas de vídeo, cujo empréstimo deve ser controlado. Cada fita possui um número. Para cada filme, é necessário saber seu título e sua categoria (comédia, drama, aventura, ...). Cada filme recebe um identificador próprio. Para cada fita é controlado que filme ela contém. Para cada filme há pelo menos uma fita, e cada fita contém somente um filme. Alguns poucos filmes necessitam duas fitas. Os clientes podem desejar encontrar os filmes estrelados pelo seu ator predileto. Por isso, é necessário manter a informação dos atores que estrelam em cada filme. Nem todo filme possui estrelas. Para cada ator os clientes às vezes desejam saber o nome real, bem como a data de nascimento. A locadora possui muitos clientes cadastrados. Somente clientes cadastrados podem alugar fitas. Para cada cliente é necessário saber seu nome e seu sobrenome, seu telefone e seu endereço. Além disso, cada cliente recebe um número de associado. Finalmente, desejamos saber que fitas cada cliente tem emprestadas. Um cliente pode ter várias fitas em um instante no tempo. Não são mantidos registros históricos de aluguéis

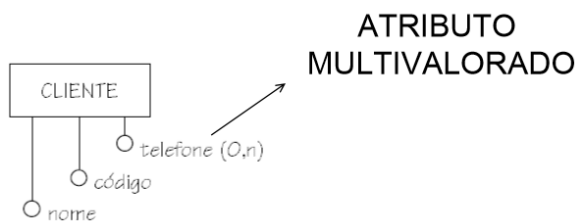
21. Atributos

Atributo = Dado que é associado a cada ocorrência de uma entidade ou de um relacionamento



Para associar informações a ocorrências de entidades ou de relacionamentos usa-se o conceito de atributo.

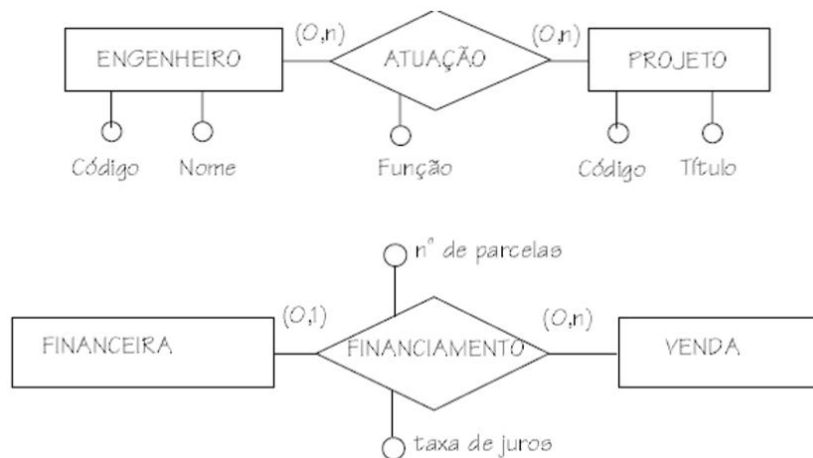
22. Cardinalidade de atributos



A cardinalidade (1,1) para os atributos pode ser omitida.

OBS. Na prática os atributos não são expostos no modelo!

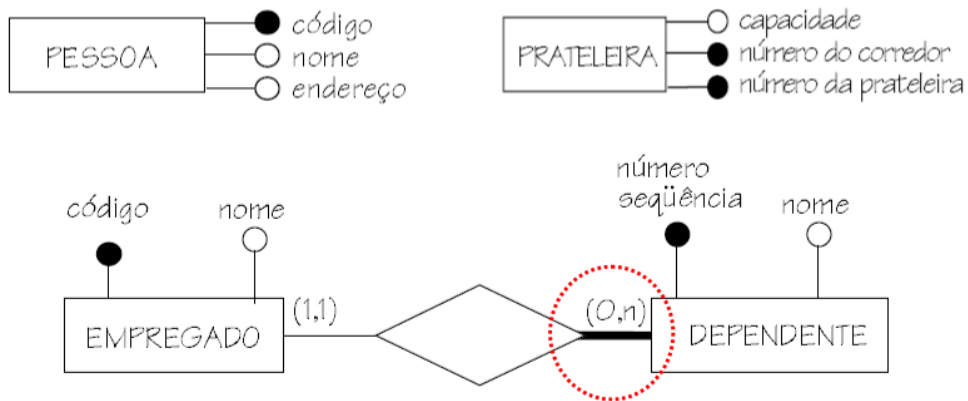
23. Atributo de relacionamento



24. Identificando entidades

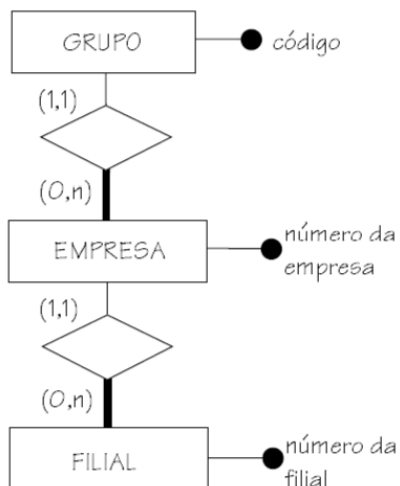
Identificador de entidade: Conjunto de atributos e relacionamentos cujos valores distinguem uma ocorrência da entidade das demais

- Identificador simples
- Identificador composto
- Relacionamento identificador



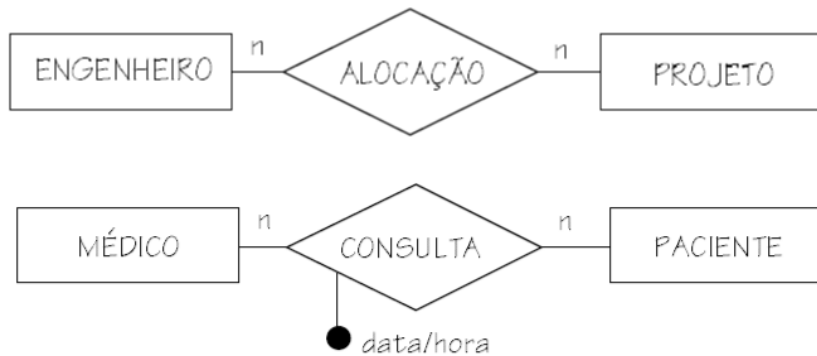
Nesse caso, alguns autores dizem que a entidade **DEPENDENTE** é uma entidade fraca. O termo “fraca” deriva-se do fato de a entidade somente existir quando relacionada a outra entidade e de usar como parte de seu identificador, o identificador de entidades relacionadas.

25. Entidades fracas!



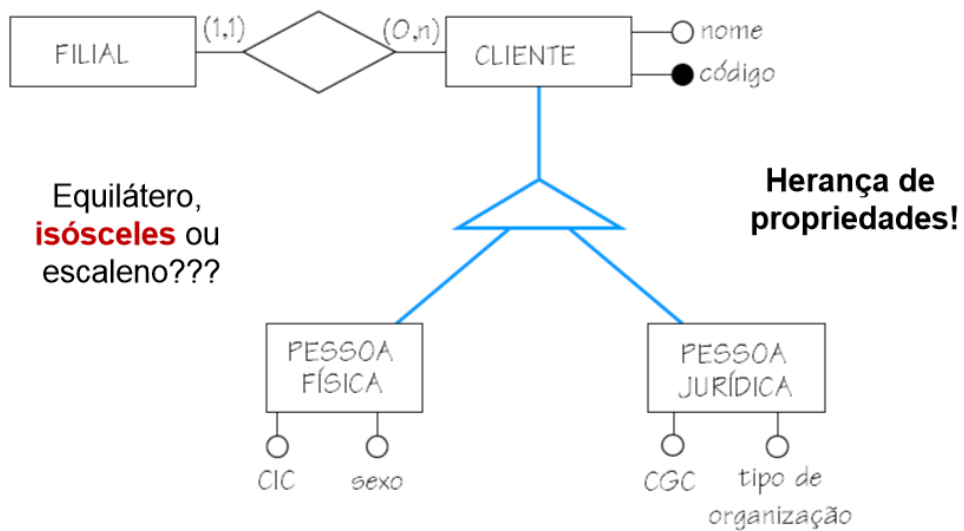
Entidades chamadas “fracas” podem ser “centrais” em um modelo.

26. Identificando relacionamentos

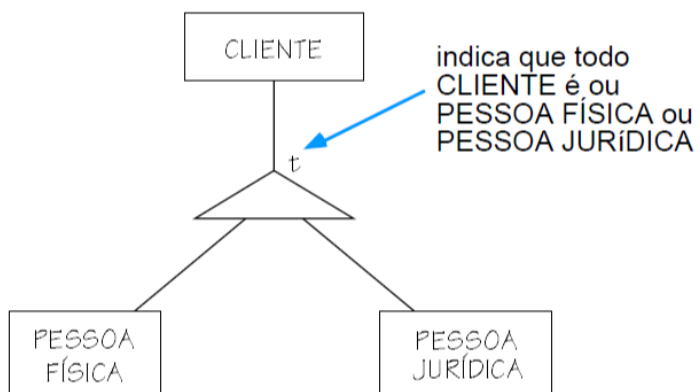


De forma geral, um relacionamento é identificado pelas entidades dele participantes, bem como pelos atributos identificadores eventualmente existentes.

27. Generalização especialização

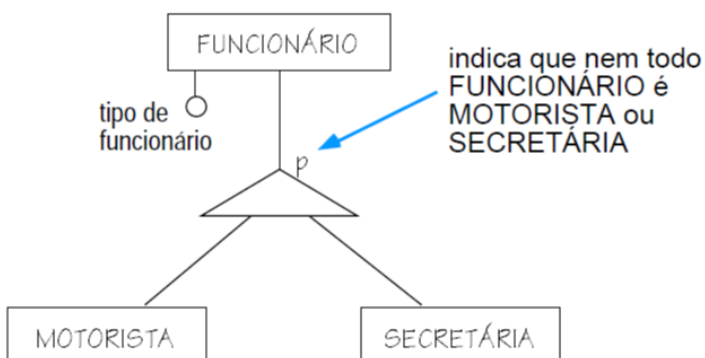


28. Generalização/Especialização Total



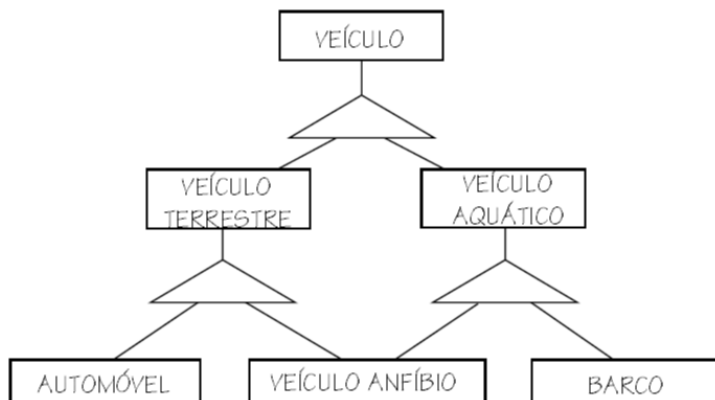
Em uma generalização/especialização total para cada ocorrência da entidade genérica existe sempre uma ocorrência em uma das entidades especializadas.

29. Generalização/Especialização Parcial

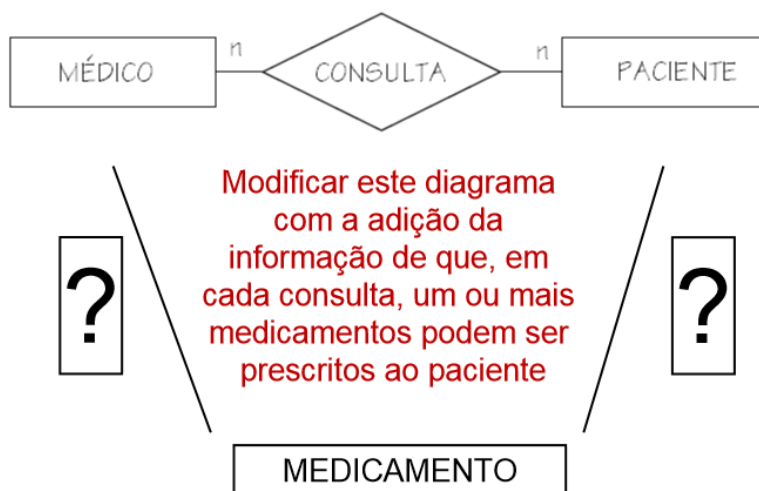


Em uma Generalização/Especialização parcial, nem toda ocorrência de uma entidade genérica possui uma ocorrência correspondente em uma entidade especializada.

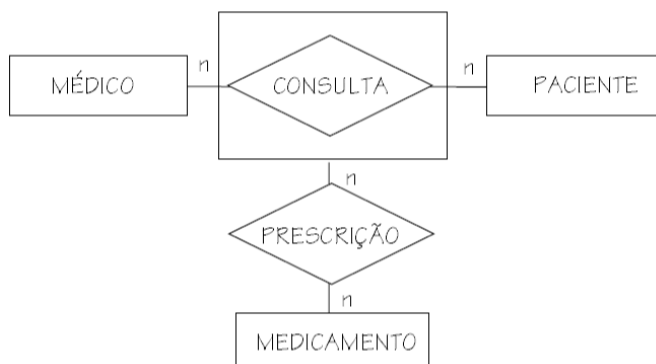
30. Herança múltipla



31. Entidade Associativa

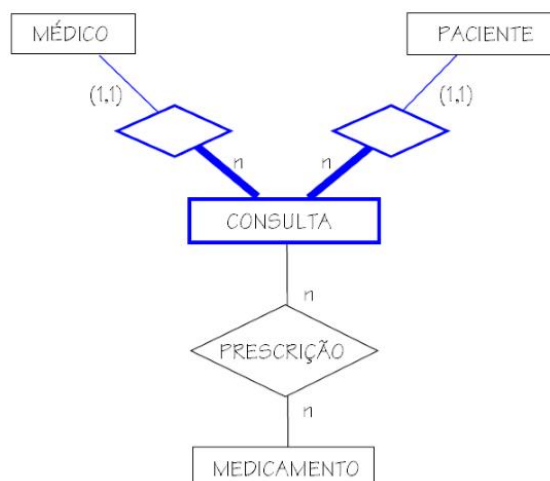


Quais seriam os efeitos?



Uma entidade associativa nada mais é que a redefinição de um relacionamento, que passa a ser tratado como se fosse também uma entidade.

32. Entidade Associativa (equivalência)




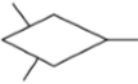






Observe-se que, para manter equivalência com o diagrama anterior, uma consulta está relacionada com exatamente um médico e exatamente um paciente (cardinalidade mínima e máxima é um).

Equivalente aqui significa que ambos geram o mesmo banco de dados relacional.

33. Esquemas gráficos e textuais

A descrição de um modelo é chamada, na terminologia de banco de dados, de o *esquema do banco de dados*

Resumo da simbologia da representação gráfica de esquemas ER

Conceito	Símbolo
Entidade	
Relacionamento	
Atributo	
Atributo Identificador	 
Relacionamento identificador	
Generalização/especialização	
Entidade associativa	

34. Esquemas gráficos e textuais

Um esquema ER pode ser um texto na sintaxe de uma linguagem textual para definição de esquemas ER. A sintaxe é dada na forma de uma gramática BNF (**Backus-Naur**).

```

ESQUEMA → Esquema: ESQUEMA_NOME
          SEÇÃO_ENTIDADE
          SEÇÃO_GENERALIZAÇÃO
          SEÇÃO_ENT_ASSOCIATIVA
          SEÇÃO_RELACIONAMENTO

SEÇÃO_ENTIDADE → {DECL_ENT}

DECL_ENT → Entidade: ENTIDADE_NOME
          [SEÇÃO_ATRIBUTO]
          [SEÇÃO_IDENTIFICADOR]

SEÇÃO_ATRIBUTO → Atributos: {DECL_ATRIB}
DECL_ATRIB → [(MIN_CARD,MAX_CARD)] ATRIBUTO_NOME [: DECL_TIPO]
MIN_CARD → 0 | 1
MAX_CARD → 1 | n
DECL_TIPO → inteiro | real | boolean | texto(INTEIRO) |
           enumeração(LISTA_VALORES)

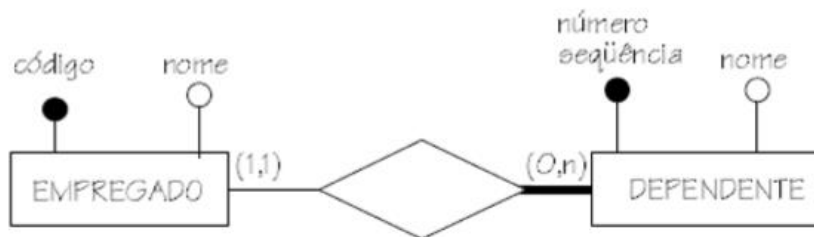
SEÇÃO_IDENTIFICADOR → Identificadores: {DECL_IDENT}
DECL_IDENT → {IDENTIFICADOR}
IDENTIFICADOR → ATRIBUTO_NOME |
               ENTIDADE_NOME (via RELACIONAMENTO_NOME)

SEÇÃO_GENERALIZAÇÃO → {DECL_HIERARQUIA_GEN}
DECL_HIERARQUIA_GEN → Generalização [(COBERTURA)]: NOME_GEN
                    PAI: NOME_ENTIDADE
                    FILHO: LISTA_NOME_ENTIDADE

COBERTURA → t | p

SEÇÃO_ENT_ASSOCIATIVA → {DECL_ENT_ASSOC}
DECL_ENT_ASSOC → EntidadeAssociativa: NOME_RELACIONAMENTO

SEÇÃO_RELACIONAMENTO → {DECL_RELACION}
DECL_RELACION → Relacionamento: NOME_RELACIONAMENTO
               Entidades: {DECL_ENT_RELACIONADA}
               [Atributos: {DECL_ATRIB}]
               [Identificadores: {DECL_IDENT}]
DECL_ENT_RELACIONADA → [(CARD_MIN,CARD_MAX)] NOME_ENTIDADE
    
```



Esquema: EMP_DEP

Entidade: EMPREGADO

Atributos: CÓDIGO: inteiro
Identificadores: CÓDIGO

Entidade: DEPENDENTE

Atributos: NÚMERO_SEQUENCIA: inteiro
NOME: texto(50)
Identificadores: EMPREGADO via EMP_DEP
NÚMERO_SEQUENCIA

Relacionamento: EMP_DEP

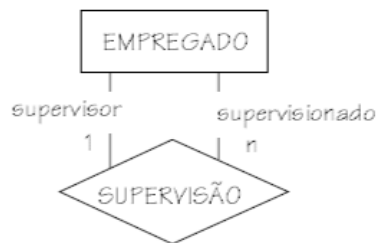
Entidades: (1,1) EMPREGADO
(0,n) DEPENDENTE

Exercícios

1 - Explique a diferença entre uma entidade e uma ocorrência de entidade. Exemplifique.

2 - O que é o papel de uma entidade em um relacionamento. Quando é necessário especificar o papel das entidades de um relacionamento?

3 - Confeccione um possível diagrama de ocorrências para o relacionamento SUPERVISÃO (da figura abaixo) e suas respectivas entidades.



4 - Dê um exemplo de um relacionamento ternário

5 - Para o exemplo de relacionamento ternário da questão anterior, justifique a escolha das cardinalidades mínima e máxima.

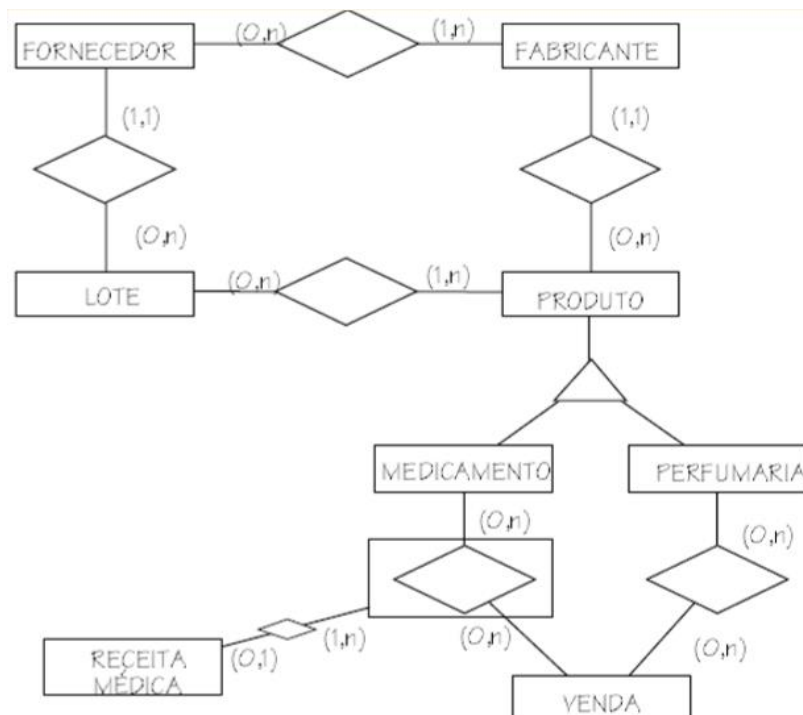
Sobre a figura no próximo slide que apresenta um modelo de dados para uma farmácia:

6 - Descreva em português tudo o que está representado neste diagrama.

7 - Invente nomes para os relacionamentos.

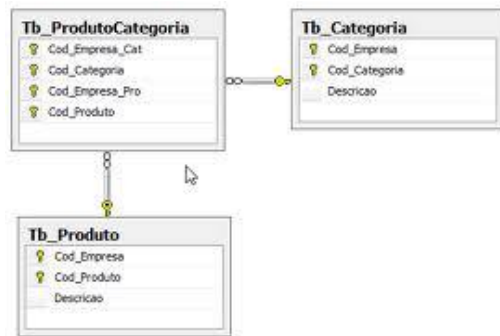
8 - Dê uma justificativa para as cardinalidades mínimas do relacionamento entre FORNECEDOR e FABRICANTE.

9 - Explique o significado das cardinalidades mínima e máxima do relacionamento ternário (entre MEDICAMENTO, VENDA e RECEITA MÉDICA).



35. Modelo formal

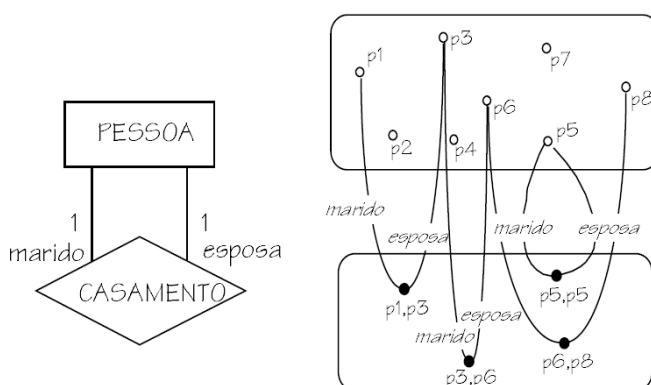
Um DER é um modelo formal, preciso, não ambíguo. Isto significa que diferentes leitores de um mesmo DER devem sempre entender exatamente o mesmo. Tanto é assim, que um DER pode ser usado como entrada a uma ferramenta CASE na geração de um banco de dados relacional.



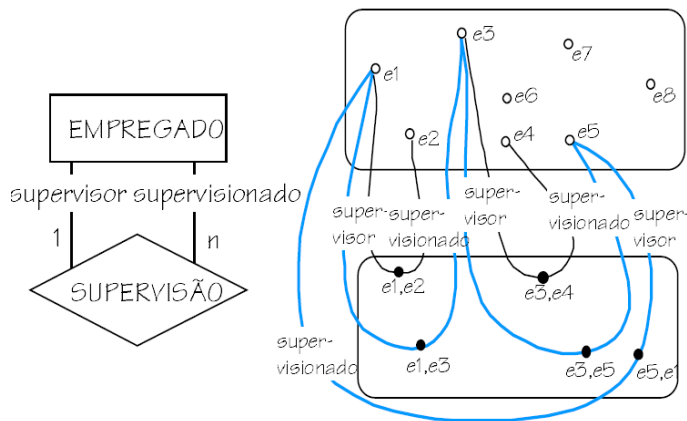
Falsa impressão de simplicidade!

36. Expressividade limitada

Em um modelo ER, são apresentadas apenas algumas propriedades de um banco de dados. Em realidade, a linguagem dos DER é uma linguagem muito pouco poderosa e **muitas** propriedades desejáveis do banco de dados necessitam **ser anotadas adicionalmente** ao DER.



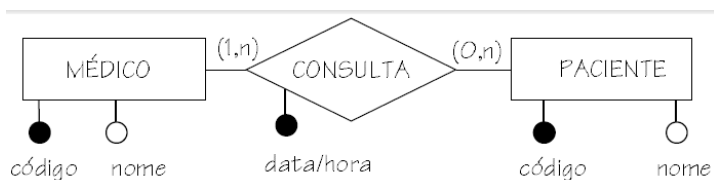
O problema é que a restrição é uma restrição de integridade *recursiva* e estas não podem ser representadas através de diagramas ER. Neste caso, resta apenas especificar a restrição a parte do DER.



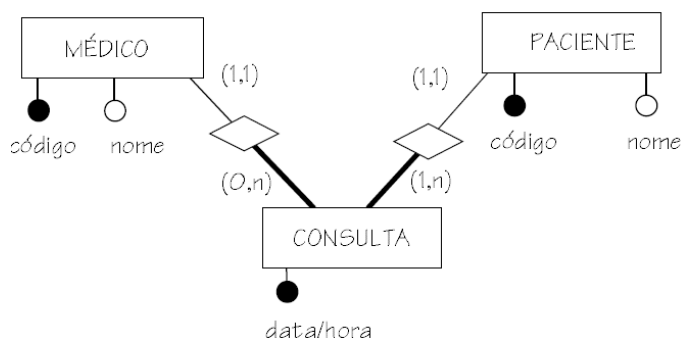
Até onde deve ser modificado um DER para introduzir restrições de integridade?

A resposta não é trivial. Aqui entra o bom senso do modelador. É necessário lembrar o objetivo que se tem ao construir um diagrama ER: o de projetar um banco de dados, e não o de especificar todas restrições de integridade.

37. Equivalência entre modelos



Os dois modelos são equivalentes, pois expressam o mesmo e geram o mesmo banco de dados.



Necessário considerar um conjunto de regras de tradução de modelos ER para modelos lógicos de BD

Como um relacionamento n:n pode ser transformado em entidade, é possível construir modelos sem relacionamentos n:n. Neste fato, baseiam-se diversas

variantes da abordagem ER, que excluem relacionamentos n:n, ou excluem apenas relacionamentos n:n com atributos.

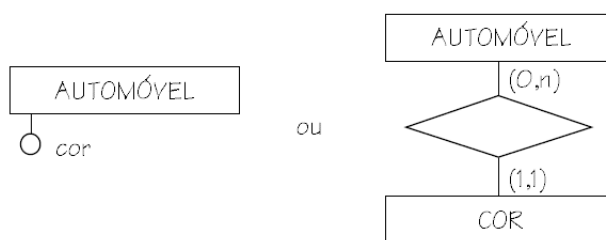
Um outro caso de equivalência, é entre um relacionamento de cardinalidade 1:1 e com cardinalidade mínima “1” em ambos os lados pode ser substituído por uma única entidade.

38. Identificando construções

Como definir se um objeto da realidade será uma entidade, um atributo ou um relacionamento?

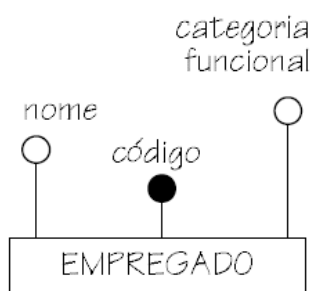
R: Conhecer o seu contexto, e considerar a decisão sujeita a alterações

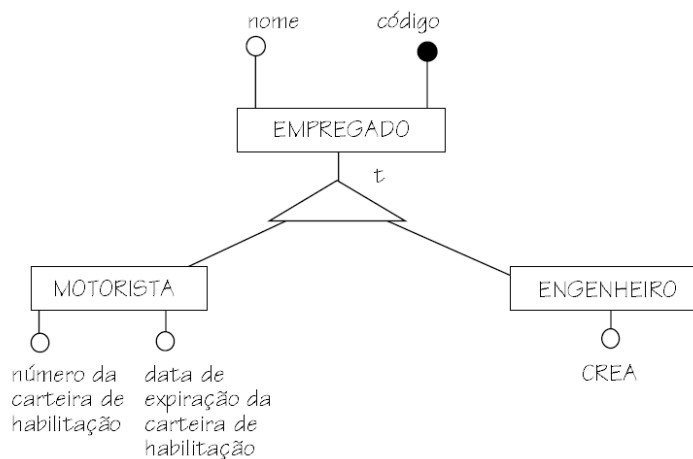
39. Atributo X entidade relacionada



- ❖ Caso o objeto esteja vinculado a outros objetos (atributos, relacionamentos, entidades genéricas ou especializadas), o objeto deve ser modelado como entidade, e atributo caso contrário.
- ❖ Quando o conjunto de valores de um determinado objeto é fixo durante toda a vida do sistema ele pode ser modelado como atributo. Quando existem transações no sistema que alteram o conjunto de valores do objeto, o mesmo não deve ser modelado como atributo.

40. Atributos X Generalização/especialização





Uma
ser

se que as classes especializadas

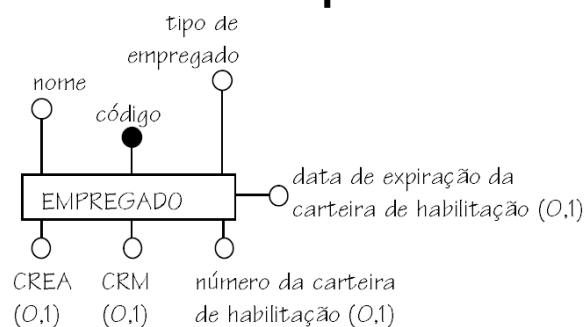
de entidades possuem propriedades particulares

especialização deve
usada quando sabe-

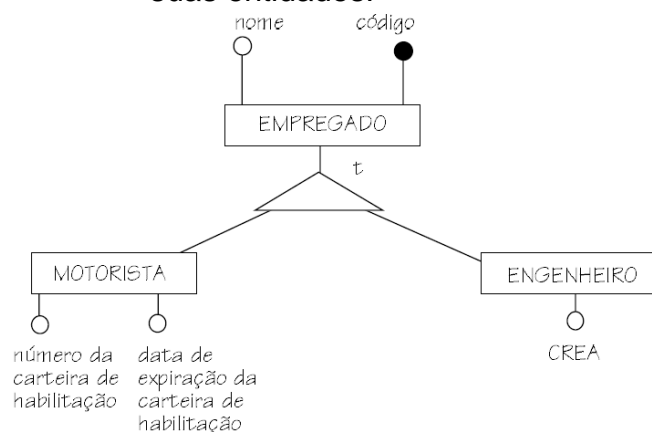
41. Atributos opcionais e multi-valorados

Quando se inicia o processo de modelagem é aconselhável tentar restringir-se ao uso de atributos *obrigatórios e mono-valorados*.

42. Atributos opcionais



Atributos opcionais escondem as diferentes categorias de empregados e suas entidades!

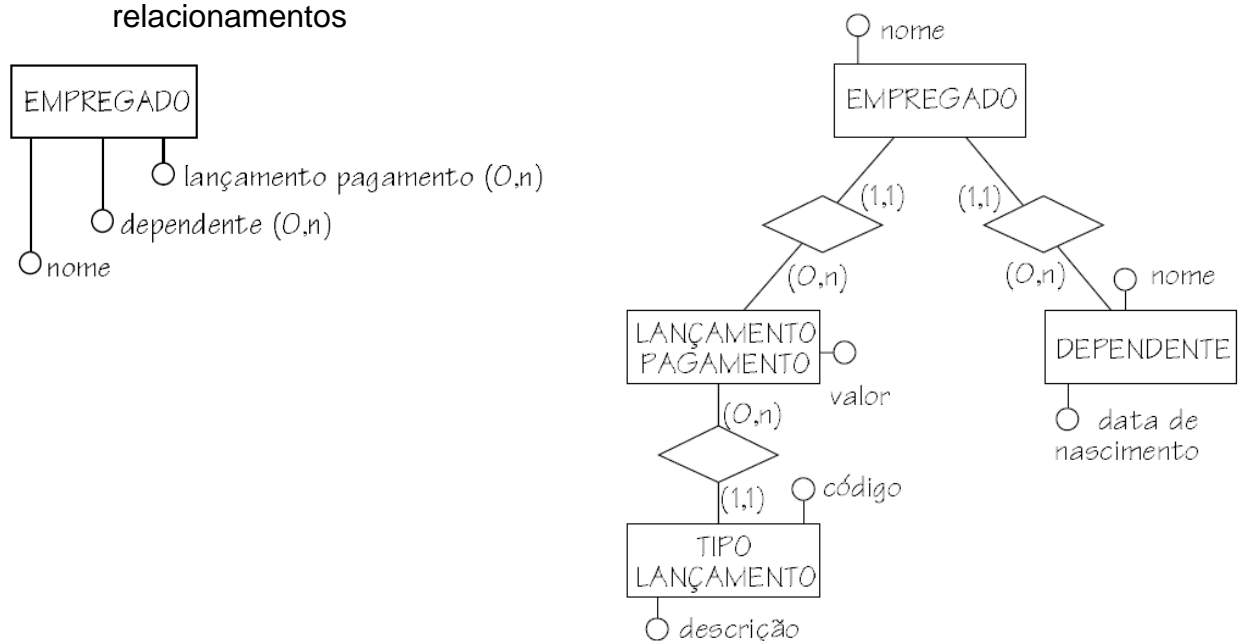


A realidade é modelada com maior fidelidade, caso a entidade EMPREGADO seja especializada.

43. Atributo multi-valorado

São indesejados por dois motivos:

- Nos SGBD relacionais que seguem o padrão SQL/2, atributos multi-valorados não possuem implementação direta
- podem induzir a um erro de modelagem, que é o de ocultar entidades e relacionamentos



44. Verificação do modelo

É o controle de qualidade, para ser considerado bom, o modelo deve:

- Ser correto

- Quando os conceitos de modelagem ER são corretamente empregados para modelar a realidade em questão
- b) Ser completo
- Um modelo completo deve fixar todas propriedades desejáveis do banco de dados.
- c) Não conter redundâncias
- Um modelo deve ser mínimo, isto é não deve conter conceitos redundantes.

45. Modelo Correto

- ❖ Erros sintáticos (não cumpre as regras de construção)
 - Regras de construção (Ex. associação entre atributos)
- ❖ Erros semânticos (reflete a realidade de forma inconsistente)
 - Estabelecer associações incorretas
 - Usar entidade do modelo como atributo de outra entidade
 - Usar número incorreto de entidades em um relacionamento

46. Modelo completo

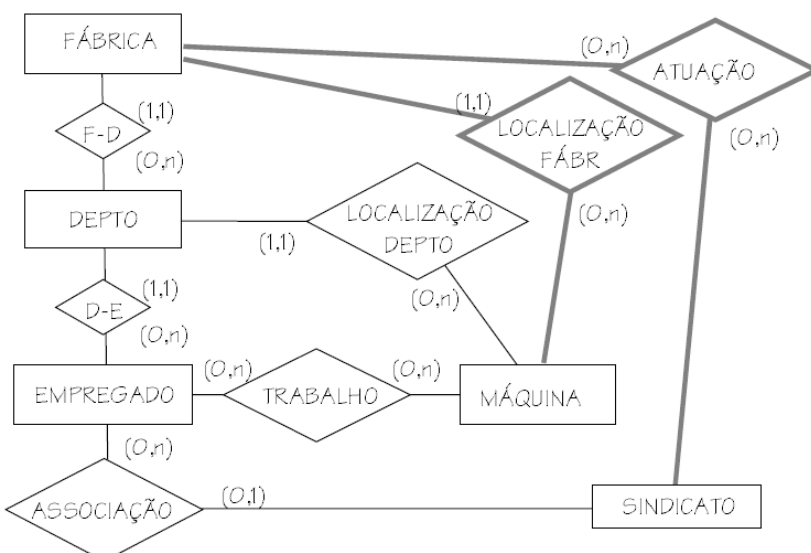
Uma boa forma de verificar se o modelo é completo é verificar se todos os dados que devem ser obtidos do banco de dados estão presentes e se todas as transações de modificação do banco de dados podem ser executadas sobre o modelo.

COMPLETUDE



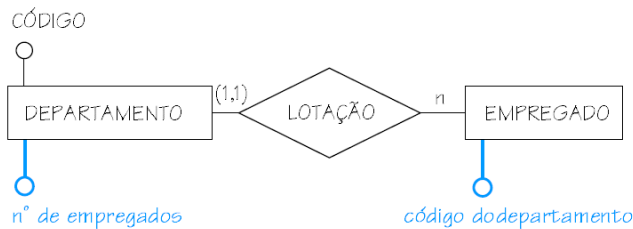
EXPRESSIVIDADE

LIMITADA



47. Modelo livre de redundâncias

Um modelo deve ser mínimo! Isto é não deve conter conceitos redundantes...

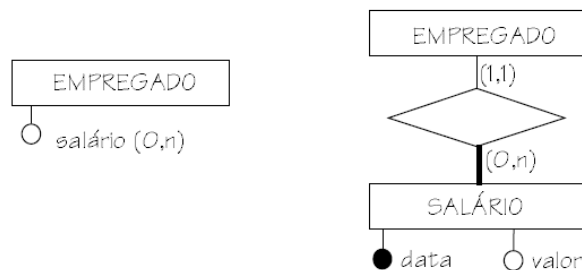


Atributos redundantes são atributos deriváveis a partir da execução de procedimentos de busca de dados e/ou cálculos sobre o banco de dados

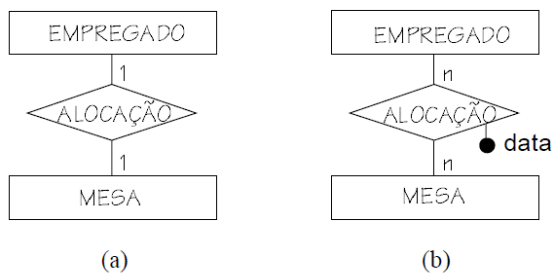
48. Aspectos temporais

É necessário lembrar que assim como informações são incluídas no banco de dados, elas também podem ter que ser eliminadas do banco de dados.

Atributos cujos valores modificam ao longo do tempo

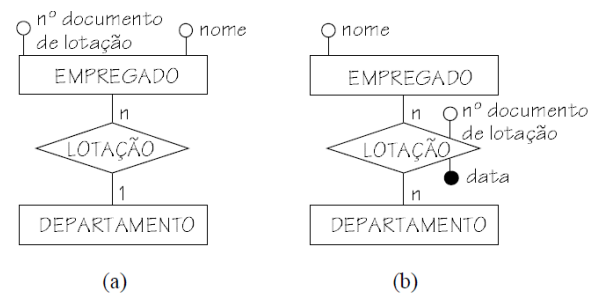


Relacionamentos que modificam ao longo do tempo



Base de dados contém apenas a alocação atual

Base de dados contém a história das alocações



Base de dados contém apenas a lotação atual

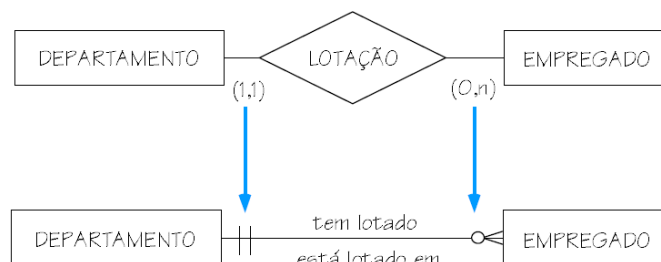
Base de dados contém a história das lotações

49. Estabelecendo padrões

Modelos de dados são usados para comunicação entre as pessoas da organização (usuários, analistas, programadores,...) e até mesmo para a comunicação com programas (ferramentas CASE, geradores de código,...). Assim, ao usar modelagem de dados, é necessário estabelecer padrões de confecção de modelos.

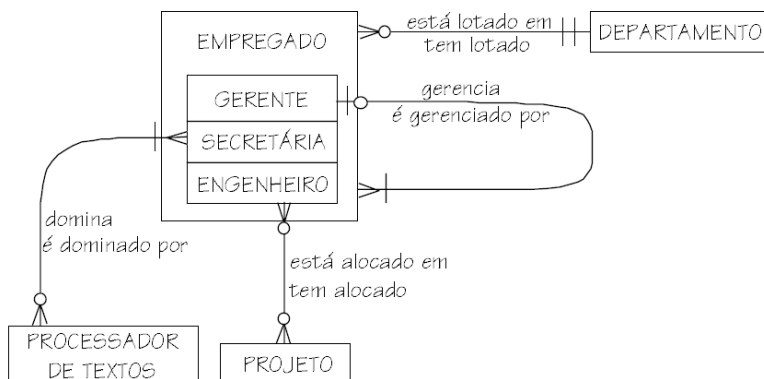
50. Variações do modelo ER

Notação “Engenharia de Informação”

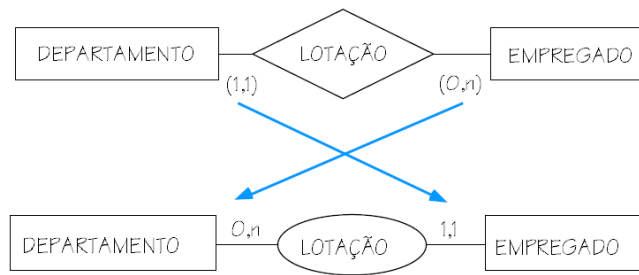


Notação para cardinalidade máxima e mínima:

- | Cardinalidade (mínima, máxima) 1
- Cardinalidade mínima 0
- ≧ Cardinalidade máxima n

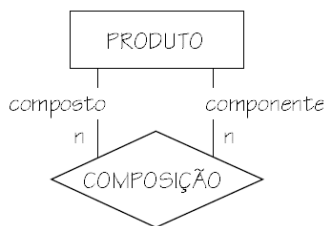


Na notação “MERISE” usa-se a semântica participativa. Nesta semântica, a cardinalidade indica quantas vezes uma ocorrência de entidade participa de um relacionamento”



Exercícios

1 - A Figura abaixo apresenta um relacionamento que associa um produto de uma indústria com seus componentes (em inglês, “bill-of-materials”). Cada produto pode possuir diversos componentes e cada componente pode aparecer dentro de diversos produtos. Assim trata-se de um relacionamento do tipo n:n. Uma restrição que deve ser imposta sobre o banco de dados em questão é a de que um produto não pode aparecer na lista de seus componentes.



Pergunta-se:

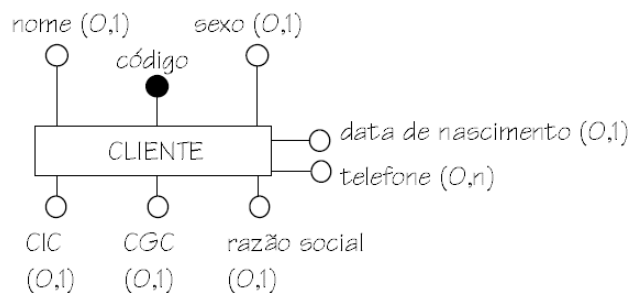
- O modelo apresentado na figura contém esta restrição?
- Caso negativo, é possível alterar o modelo em questão para incluir esta restrição, se considerarmos que o nível de profundidade da hierarquia de composição de cada produto não excede três (tem-se apenas produtos prontos, produtos semi-acabados e matérias-primas)? Caso afirmativo, apresente a solução.
- É possível estender a solução do quesito anterior para uma hierarquia não limitada de níveis de composição?

2 - Deseja-se modelar os clientes de uma organização. Cada cliente possui um identificador, um nome, um endereço e um país. Discuta os prós e contras das duas alternativas de modelagem de país:

- Como atributo da entidade cliente

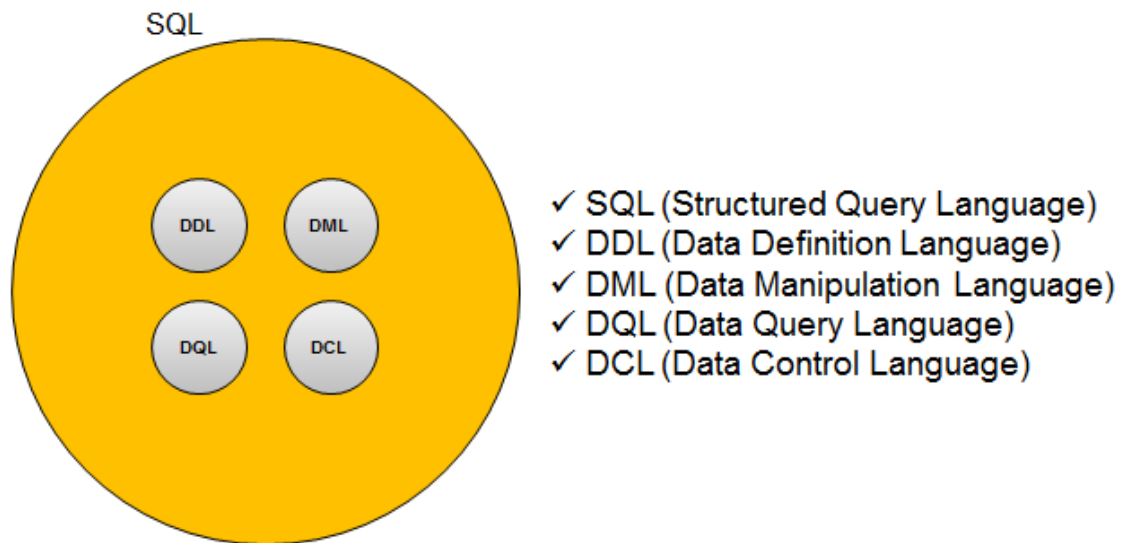
b) Como entidade relacionada a cliente.

3 - A figura abaixo apresenta uma entidade e respectivos atributos, muitos deles opcionais e um multi-valorado. Considere que há dois tipos de clientes, pessoas físicas e pessoas jurídicas. Pessoas físicas possuem código, CIC, nome, sexo (opcional), data de nascimento (opcional) e telefones (opcionais). Pessoas jurídicas possuem código, CGC, razão social e telefones (opcionais). Apresente um diagrama ER que modele mais precisamente esta realidade. Explique no que seu diagrama é mais preciso que o mostrado.



4 - Transforme o modelo ER resultante do exercício anterior da notação Chen para a notação da Engenharia de Informações.

51. Linguagens do Banco de Dados



✓ **SQL (Structured Query Language)**

- Linguagem padrão para comandos DDL, DML, DQL e DCL
- Primeira versão lançada em 1986
- Última versão lançada é chamada de SQL-99;
- Padrão ANSI;
- SQL Oracle = PLSQL
- SQL Microsoft SQL Server = Transact SQL
- SQL MySQL = SQL ANSI

✓ **DDL (Data Definition Language)**

- Utilizada por DBA`s e AD`s
- Define formas de armazenamento
 - Create tabela / view / index
 - Drop
 - Alter
 - Truncate

✓ **DML (Data Manipulation Language)**

- Manipulação dos dados
- insert
- update

- delete

- ✓ **DQL (Data Query Language)**

- Busca de dados

- Select

- ✓ **DCL (Data Control Language)**

- Controle de acesso aos dados

- grant

- revoke

Atividade de Fixação

DDL, DML, DCL ou DQL ?

- Preciso executar um comando (SQL) para alterar a estrutura da tabela “Cliente”.
- Preciso criar um local para armazenamento de informações sobre meus clientes.
- Preciso dar a permissão que o usuário “contabilidade” acesse apenas a tabela de contas.
- Gostaria de mostrar na tela de meu programa a lista de todos meus clientes cadastrados.
- Solicito a exclusão dos dados do Aluno “Paulo Alberto”.
- Preciso atualizar o telefone do cliente “Carlos”

52. Criando um servidor de Banco de Dados

Para prepararmos um ambiente de trabalho, precisaremos de um servidor de Banco de dados.

Iremos precisar de três programas básicos. O Apache, MySQL e o PHP. Portanto segue instruções para instalação desses programas.

Criando um servidor de banco de dados no Linux:

Abra o terminal com o atalho Ctrl + Alt + T

1º Atualizar os repositórios do linux

```
$ sudo apt-get update
```

```
$ sudo apt-get upgrade
```

O sudo no linux serve para elevar para super usuário os privilégios do terminal. Ou seja, ele chama o root. Portanto a senha do root será solicitada.

2º Instale o servidor de banco apache

```
$ sudo apt-get install apache2
```

3º Instale o mysql

```
$ sudo apt-get install mysql-server mysql-client
```

Obs. cadastre a senha do super usuário root

4º Teste a instalação do mysql

```
$ sudo service mysql status
```

5º Iniciar o mysql

```
$ sudo mysql -u root -p
```

Existem opções mais práticas para isso. Como instalar programas completos:

WAMP <http://www.wampserver.com/>

LAMP

53. Criação e Utilização de um Banco de Dados

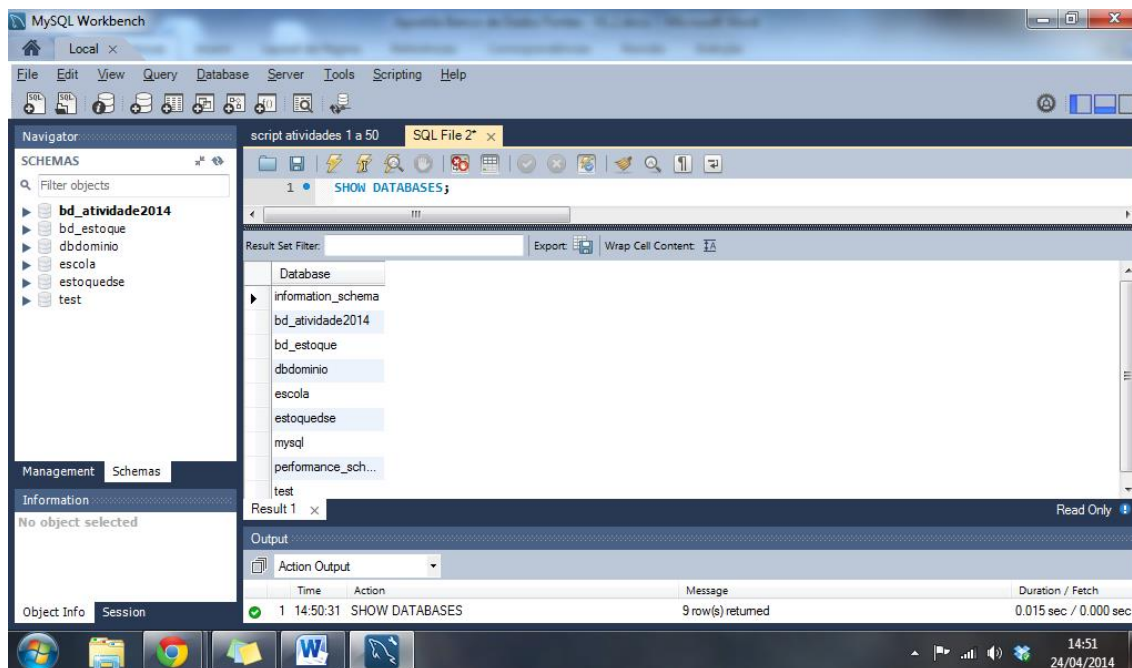
Suponha que você tenha diversos animais de estimação em sua casa (animais) e você gostaria de ter o registro de vários tipos de informações sobre eles. Você pode fazer isto criando tabelas para armazenar seus dados e carregá-los com a informação desejada.

Depois você pode responder diferentes tipos de questões sobre seus animais recuperando dados das tabelas. Para isso você precisará:

- Criar um banco de dados
- Criar uma tabela
- Carregar dados na tabela
- Recuperar dados de uma tabela de várias maneiras
- Usar múltiplas tabelas

Utilize a instrução **SHOW** para saber quais bancos de dados existem atualmente no servidor:

mysql> **SHOW DATABASES;**



A lista de bancos de dados provavelmente será diferente na sua máquina, mas os bancos de dados **mysql** e **test** provavelmente estarão entre eles. O banco de dados **mysql** é necessário porque ele descreve privilégios de acessos de usuários. O banco de dados **test** é geralmente fornecido como um espaço para que os usuários possam fazer testes.

Se o banco de dados **test** existir, tente acessá-lo:

mysql> **USE test**

54. Criando e Selecionando um Banco de Dados

Se o administrador criar seu banco de dados quando configurar as suas permissões, você pode começar a usá-lo. Senão, você mesmo precisa criá-lo:

mysql> **CREATE DATABASE animais;**

No Unix, nomes de bancos de dados são caso sensível (ao contrário das palavras chave SQL), portanto você deve sempre fazer referência ao seu banco de dados como **animais** e não **Animais**, **ANIMAIS** ou outra variação. Isto também é verdade para nomes de tabelas. (No Windows, esta restrição não se aplica, entretanto você

deve referenciar os bancos de dados e tabelas usando o mesmo caso em toda a parte da consulta.)

Criar um banco de dados não o seleciona para o uso, você deve fazer isso de forma explícita. Para fazer o **animais** o banco de dados atual, use o comando:

```
mysql> USE animais
```

55. Criando uma Tabela

Criar o banco de dados é a parte fácil, mas neste ponto ele está vazio, como o **SHOW TABLES** mostrará:

```
mysql> SHOW TABLES;
```

A parte mais difícil é decidir qual a estrutura que seu banco de dados deve ter. Quais tabelas você precisará e que colunas estarão em cada uma delas.

Você irá precisar de uma tabela para guardar um registro para cada um de seus animais de estimação. Esta tabela pode ser chamada **animais**, e ela deve conter, pelo menos, o nome de cada animal. Como o nome por si só não é muito interessante, a tabela deverá conter outras informações. Por exemplo, se mais de uma pessoa na sua família também tem animais, você pode desejar listar cada dono.

Você pode também desejar gravar algumas informações descritivas básicas como espécie e sexo.

Que tal a idade? Pode ser do interesse, mas não é uma boa coisa para se armazenar em um banco de dados. A idade muda à medida em que o tempo passa, o que significa que você sempre terá de atualizar seus registros. Em vez disso, é melhor armazenar um valor fixo como a data de nascimento. Então, sempre que você precisar da idade, basta você calculá-la como a diferença entre a data atual e a data de aniversário. O MySQL fornece funções para fazer aritmética de datas, então isto não é difícil. Armazenando datas de aniversário no lugar da idade também oferece outras vantagens:

- Você pode usar o banco de dados para tarefas como gerar lembretes para aniversários que estão chegando. (Se você pensa que este tipo de query é algo bobo, perceba que é a mesma questão que você perguntar no contexto de um banco de dados comercial para identificar clientes para quais você precisará enviar cartão de aniversário, para um toque pessoal assistido pelo computador.)
- Você pode calcular a idade em relação a outras datas diferente da data atual. Por exemplo, se você armazenar a data da morte no banco de dados, você poderá facilmente calcular qual a idade que o bicho tinha quando morreu.

Você provavelmente pode pensar em outros tipos de informações que poderão ser úteis na tabela **animais**, mas as identificadas até o momento são suficientes por agora: nome, dono, espécie, sexo, data de nascimento(nascimento) e data da morte.

Utilize a sentença **CREATE TABLE** para especificar o layout de sua tabela:

```
mysql> CREATE TABLE animais (nome VARCHAR(20), dono VARCHAR(20),
-> especies VARCHAR(20), sexo CHAR(1), nascimento DATE, morte DATE);
```

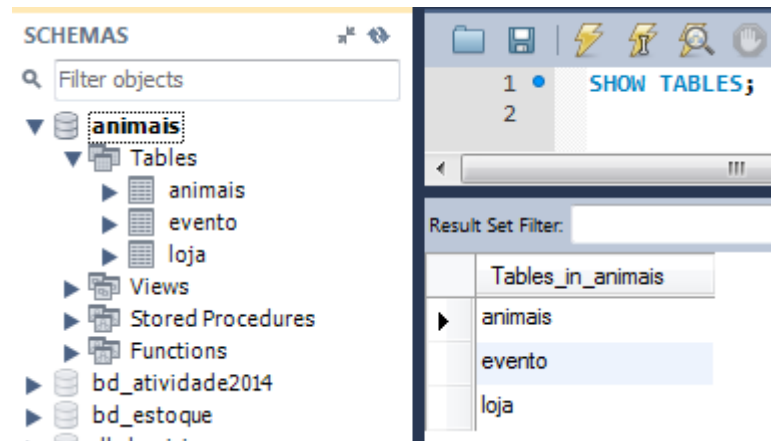
VARCHAR é uma boa escolha para os campos **nome**, **dono**, e **especies** porque os valores da coluna são de tamanho variável. Os tamanhos destas colunas não precisam necessariamente de ser os mesmos e não precisam ser **20**. Você pode escolher qualquer tamanho de **1** a **255**, o que você achar melhor. (Se você não fizer uma boa escolha e depois precisar de um campo maior, o MySQL fornece o comando **ALTER TABLE**.)

O sexo dos animais podem ser representados em várias formas, por exemplo, **"m"** e **"f"** ou mesmo **"macho"** e **"fêmea"**. É mais simples usar os caracteres **"m"** e **"f"**.

O uso do tipo de dados **DATE** para as colunas **nascimento** e **morte** são obviamente a melhor escolha.

Agora que você criou uma tabela, a instrução **SHOW TABLES** deve produzir alguma saída:

```
mysql> SHOW TABLES;
```



Para verificar se sua tabela foi criada da forma que você esperava, utilize a instrução **DESCRIBE**:

mysql> **DESCRIBE animais;**

	Field	Type	Null	Key	Default	Extra
▶	nome	varchar(20)	YES		NULL	
	dono	varchar(20)	YES		NULL	
	especies	varchar(20)	YES		NULL	
	sexo	char(1)	YES		NULL	
	nascimento	date	YES		NULL	
	morte	date	YES		NULL	

Você pode usar **DESCRIBE** a qualquer hora, por exemplo, se você esquecer os nomes das colunas na sua tabela ou de que tipos elas têm.

56. Alterando a estrutura da tabela com o Alter Table

Para acrescentar um campo em uma tabela:

ALTER TABLE teste **ADD** descricao teste **VARCHAR(70)** **AFTER** nome teste;

Com essa instrução acrescentamos uma coluna de nome descriçãoteste na tabela teste depois da tabela nome teste.

```
ALTER TABLE teste ADD borogodo VARCHAR(15) AFTER idteste;
```

Com essa instrução acrescentamos uma coluna de nome borogodo.

Com essa outra alteramos o tipo da variável e a localização da coluna borogodo

```
ALTER TABLE teste MODIFY borogodo CHAR(1) AFTER descriçãoteste
```

Como alterar o atributo/coluna

```
ALTER TABLE teste CHANGE borogodo borogodo VARCHAR(30);
```

Para apagar uma coluna:

```
ALTER TABLE teste DROP borogodo;
```

57. Recuperando Informações de uma Tabela

Antes de recuperarmos informações, vamos usar a instrução SQL abaixo para inserir registros na tabela:

```
INSERT INTO animais
```

```
(nome, dono, especies, sexo, nascimento, morte) VALUES
```

```
('pink', 'Emanuel', 'cachorro', 'f', '2004-01-01', ''),
```

```
('Leque', 'Fernando', 'cachorro', 'f', '2005-02-12', '2014-01-20'),
```

```
('Negão', 'Luiza', 'cachorro', 'm', '2007-02-03', ''),
```

```
('Kiú', 'Andrey', 'cachorro', 'm', '2006-03-10', '2010-05-15'),
```

```
('Mél', 'Rafael', 'gato', 'm', '2008-04-05', ''),
```

```
('Loro', 'Dara', 'papagaio', 'f', '2009-05-26', ''),
```

```
('Tití', 'Adler', 'Jabotí', 'm', '2010-11-30', '');
```

A instrução **SELECT** é usada para recuperar informações de uma tabela. A forma geral da instrução é:

SELECT o_que_mostrar

FROM de_qual_tabela

WHERE condições_para_satisfazer;

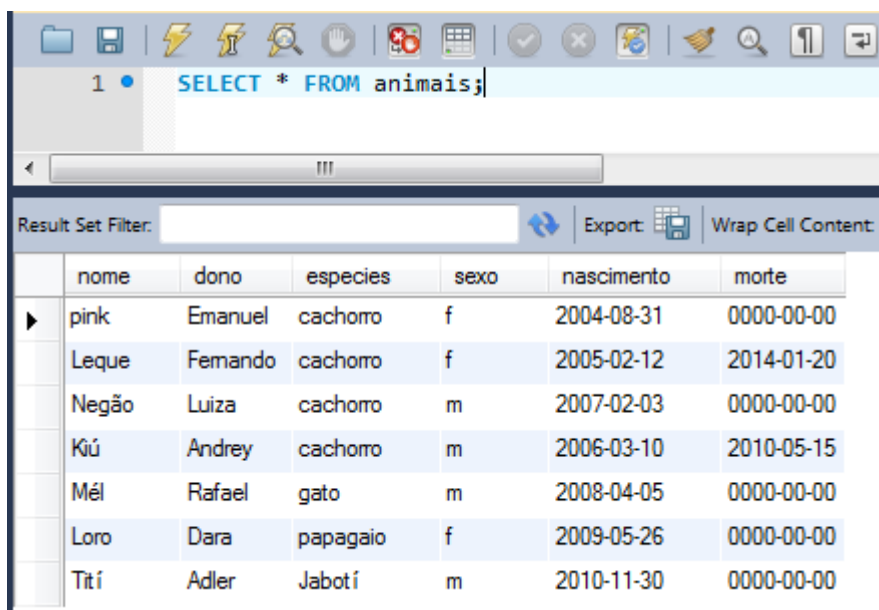
o_que_mostrar indica o que você deseja ver. Isto pode ser uma lista de colunas ou * para indicar ``todas colunas."

de_qual_tabela indica a tabela de onde você deseja recuperar os dados. A cláusula **WHERE** é opcional. Se estiver presente, **condições_para_satisfazer** especificam as condições que os registros devem satisfazer para fazer parte do resultado.

58. Selecionando Todos os Dados

A forma mais simples do **SELECT** recuperar tudo de uma tabela:

mysql> **SELECT * FROM animais;**



The screenshot shows a MySQL client window with a toolbar at the top. The query editor contains the text "1 • SELECT * FROM animais;". Below the editor, the "Result Set Filter" is empty. The results are displayed in a table with 7 columns: nome, dono, especies, sexo, nascimento, and morte. The table contains 7 rows of data.

	nome	dono	especies	sexo	nascimento	morte
▶	pink	Emanuel	cachorro	f	2004-08-31	0000-00-00
	Leque	Fernando	cachorro	f	2005-02-12	2014-01-20
	Negão	Luiza	cachorro	m	2007-02-03	0000-00-00
	Kiú	Andrey	cachorro	m	2006-03-10	2010-05-15
	Mél	Rafael	gato	m	2008-04-05	0000-00-00
	Loro	Dara	papagaio	f	2009-05-26	0000-00-00
	Tití	Adler	Jabotí	m	2010-11-30	0000-00-00

Esta forma do **SELECT** é útil se você deseja ver sua tabela inteira como agora, depois de você acabar de carregá-la com os dados iniciais. Por exemplo, você pode pensar que a data de nascimento do dono não está correta. Consultando seus papéis originais de pedigree, descobriu que o ano correto do nascimento deve ser 2005, não 2004.

Corrigir somente o registro errado com uma instrução **UPDATE**:

```
mysql> UPDATE animais SET nascimento = '2004-08-31' WHERE nome = 'pink';
```

O **UPDATE** altera apenas o registro em questão e não exige que você recarregue a tabela.

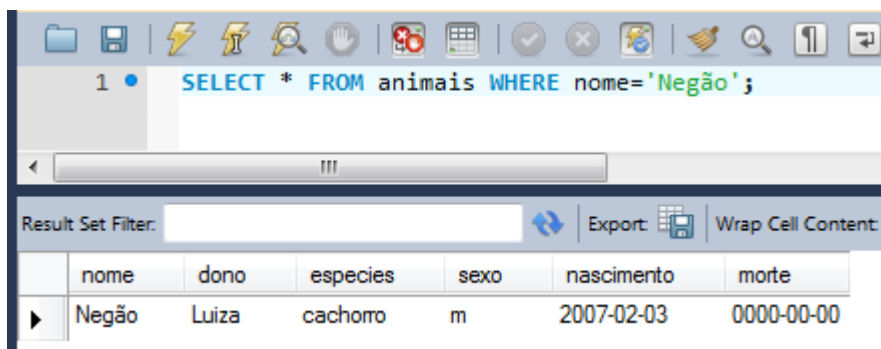
59. Selecionando Registros Específicos

Como foi mostrado na seção anterior, é fácil recuperar uma tabela inteira. Apenas omita a cláusula **WHERE** da instrução **SELECT**.

Mas normalmente você não quer ver toda a tabela, particularmente quando a tabela ficar grande. Em vez disso, você estará mais interessado em ter a resposta de uma questão em particular, no qual você especifica detalhes da informação que deseja. Vamos ver algumas consultas de seleção nos termos das questões sobre seus animais.

Você pode selecionar apenas registros específicos da sua tabela. Por exemplo, se você deseja verificar a alteração que fez na data de nascimento do Bowser, selecione o registro desta forma:

```
mysql> SELECT * FROM animais WHERE nome='Negão';
```



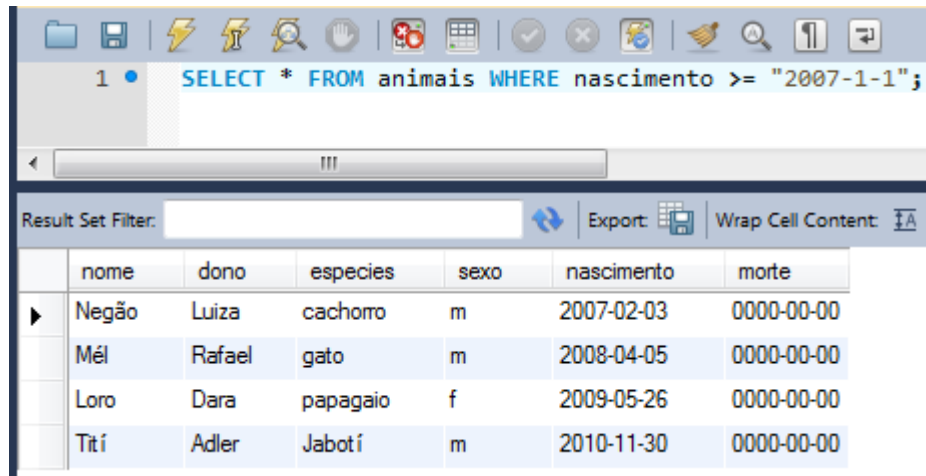
The screenshot shows a MySQL GUI window with a toolbar at the top. Below the toolbar, the SQL query `SELECT * FROM animais WHERE nome='Negão';` is entered in a text field. Below the query field, there is a "Result Set Filter:" label and a search input field. To the right of the filter are buttons for "Export" and "Wrap Cell Content:". Below these elements is a table with the following data:

	nome	dono	especies	sexo	nascimento	morte
▶	Negão	Luiza	cachorro	m	2007-02-03	0000-00-00

A saída confirma que o ano foi gravado corretamente agora como 2004 e não 2005. Comparações de strings normalmente são caso insensitivo, então você pode especificar o nome como **"Negão"**, **"NEGÃO"**, etc. O resultado da pesquisa será o mesmo.

Você pode especificar condições em qualquer coluna, não apenas no **nome**. Por exemplo, se você deseja saber quais foram os animais que nasceram depois de 1998, teste o campo **nascimento**:

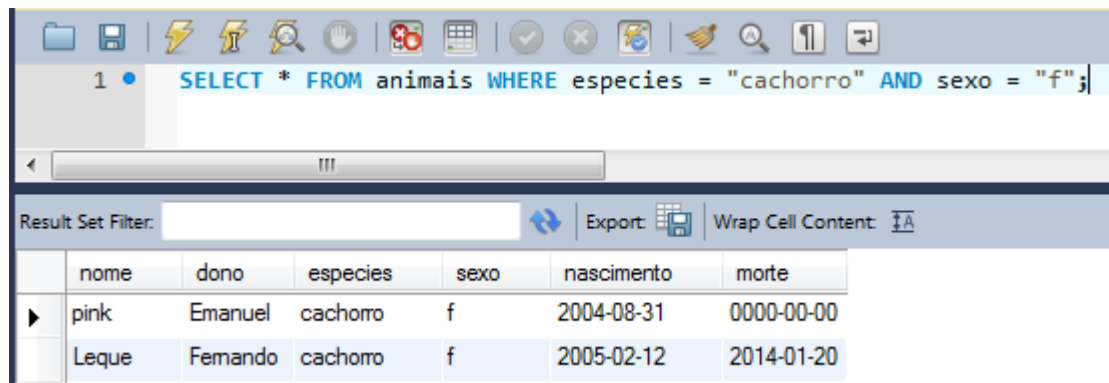
```
mysql> SELECT * FROM animais WHERE nascimento >= "2007-1-1";
```



	nome	dono	especies	sexo	nascimento	morte
▶	Negão	Luiza	cachorro	m	2007-02-03	0000-00-00
	Mél	Rafael	gato	m	2008-04-05	0000-00-00
	Loro	Dara	papagaio	f	2009-05-26	0000-00-00
	Tití	Adler	Jabotí	m	2010-11-30	0000-00-00

Você pode combinar condições, por exemplo, para encontrar cadelas (cachorro/f):

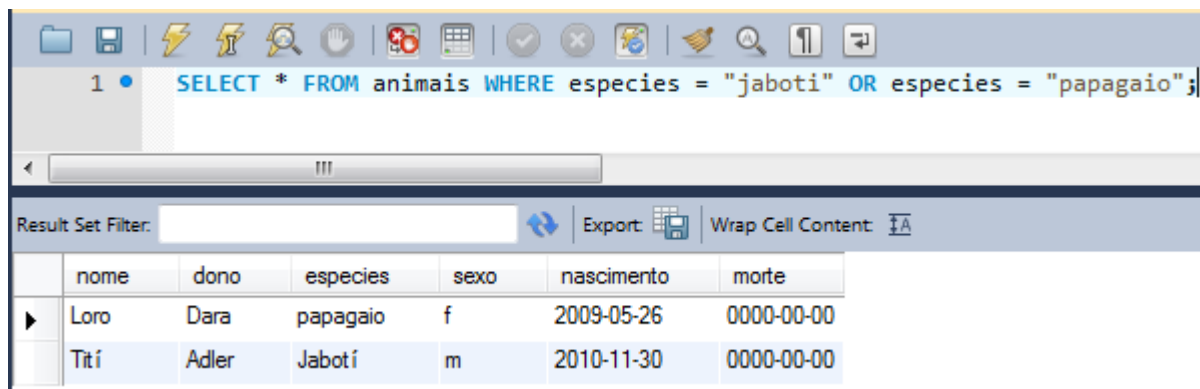
```
mysql> SELECT * FROM animais WHERE especies = "cachorro" AND sexo = "f";
```



	nome	dono	especies	sexo	nascimento	morte
▶	pink	Emanuel	cachorro	f	2004-08-31	0000-00-00
	Leque	Fernando	cachorro	f	2005-02-12	2014-01-20

A consulta anterior utiliza o operador lógico **AND** (e). Existe também um operador **OR** (ou):

```
mysql> SELECT * FROM animais WHERE especies = "jaboti" OR especies = "papagaio";
```

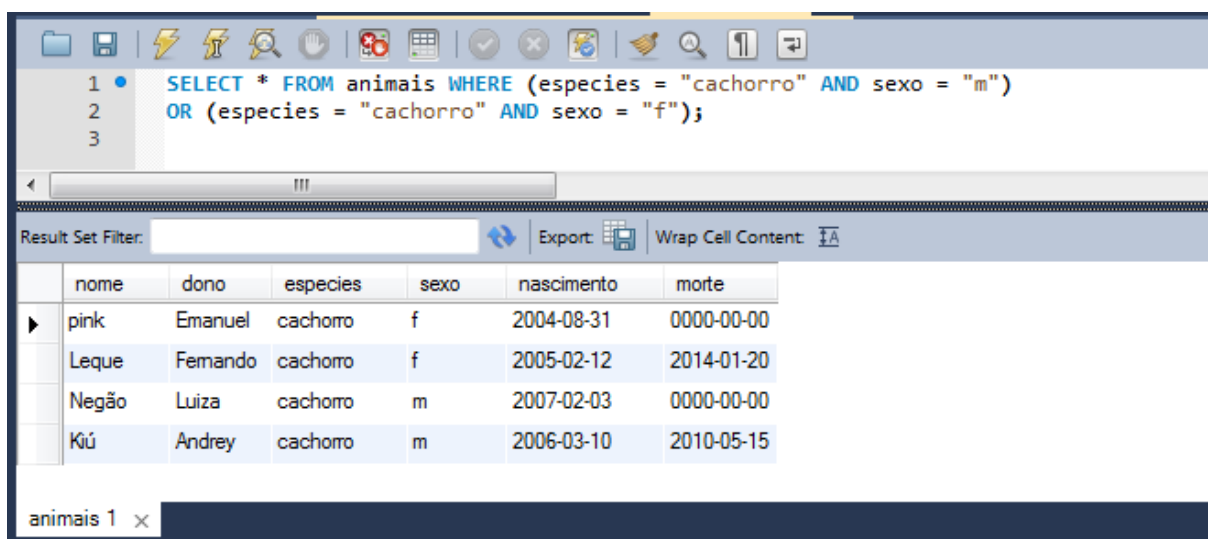
1 • `SELECT * FROM animais WHERE especies = "jaboti" OR especies = "papagaio";`

Result Set Filter: Export: Wrap Cell Content:

	nome	dono	especies	sexo	nascimento	morte
▶	Loro	Dara	papagaio	f	2009-05-26	0000-00-00
	Tití	Adler	Jabotí	m	2010-11-30	0000-00-00

AND e **OR** podem ser misturados, embora **AND** tem maior precedência que **OR**. Se você usar ambos os operadores, é uma ótima ideia usar parênteses para indicar explicitamente quais condições devem ser agrupadas:

mysql> **SELECT * FROM animais WHERE (especies = "cachorro" AND sexo = "m")**
-> OR (especies = "cachorro" AND sexo = "f");



1 • `SELECT * FROM animais WHERE (especies = "cachorro" AND sexo = "m")`
2 `OR (especies = "cachorro" AND sexo = "f");`
3

Result Set Filter: Export: Wrap Cell Content:

	nome	dono	especies	sexo	nascimento	morte
▶	pink	Emanuel	cachorro	f	2004-08-31	0000-00-00
	Leque	Fernando	cachorro	f	2005-02-12	2014-01-20
	Negão	Luiza	cachorro	m	2007-02-03	0000-00-00
	Kiú	Andrey	cachorro	m	2006-03-10	2010-05-15

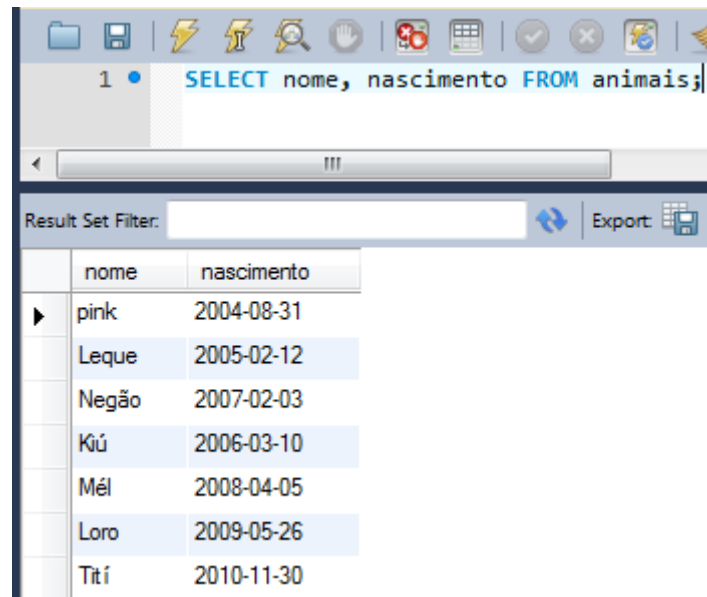
animais 1 x

60. Selecionando Colunas Específicas

Se você não desejar ver todo o registro de sua tabela, especifique as colunas em que você estiver interessado, separado por vírgulas.

Por exemplo, se você deseja saber quando seus animais nasceram, selecione as colunas **nome** e **nascimento**:

mysql> **SELECT nome, nascimento FROM animais;**

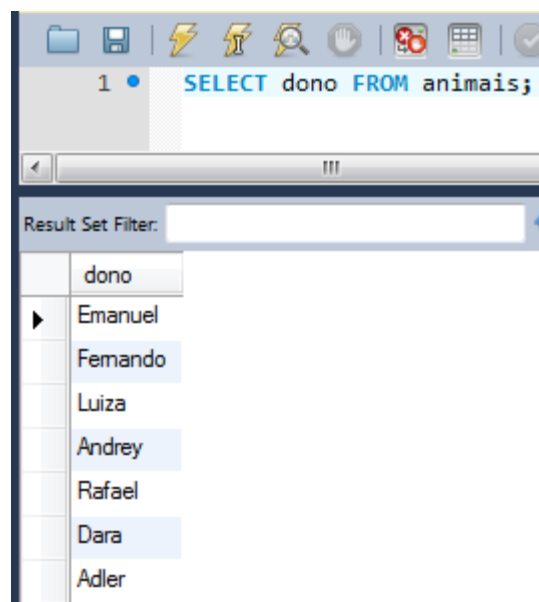


The screenshot shows a MySQL GUI window with the query 'SELECT nome, nascimento FROM animais;' entered in the top text area. Below the query, the 'Result Set Filter' is empty. The results are displayed in a table with two columns: 'nome' and 'nascimento'.

nome	nascimento
pink	2004-08-31
Leque	2005-02-12
Negão	2007-02-03
Kiú	2006-03-10
Mél	2008-04-05
Loro	2009-05-26
Tití	2010-11-30

Para saber quem são os donos dos animais, use esta consulta:

mysql> **SELECT dono FROM animais;**



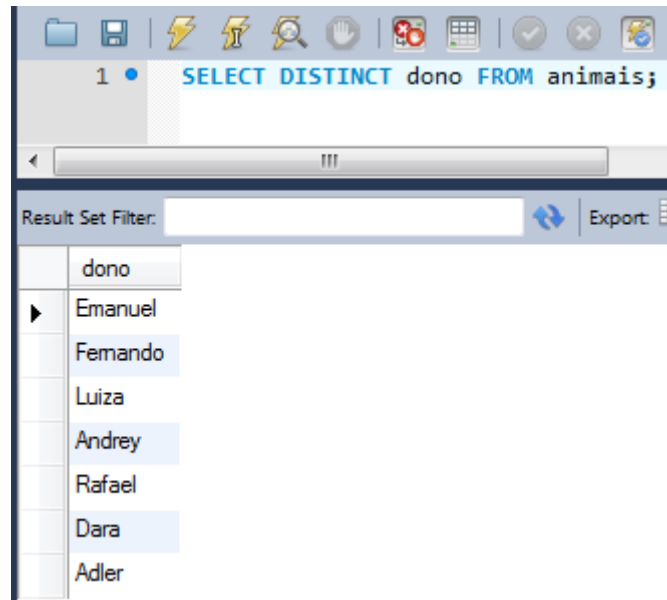
The screenshot shows a MySQL GUI window with the query 'SELECT dono FROM animais;' entered in the top text area. Below the query, the 'Result Set Filter' is empty. The results are displayed in a table with one column: 'dono'.

dono
Emanuel
Fernando
Luiza
Andrey
Rafael
Dara
Adler

Entretanto, percebe-se que a query simplesmente retornou o campo **dono** de cada registro, e alguns deles apareceram mais de uma vez. Para minimizar a saída, recupere cada registro apenas uma vez, adicionando a palavra chave

DISTINCT:

mysql> **SELECT DISTINCT dono FROM animais;**



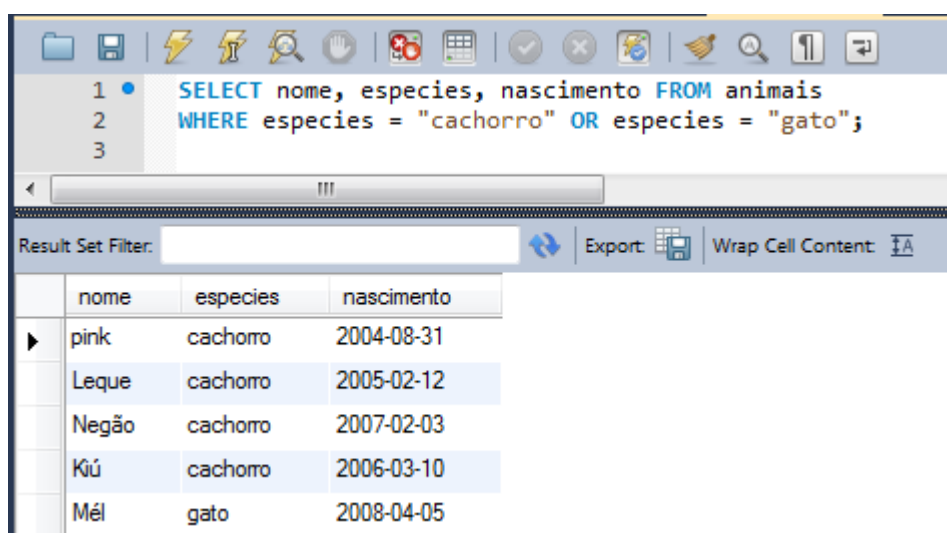
The screenshot shows a MySQL query editor window. The query entered is `SELECT DISTINCT dono FROM animais;`. The result set is displayed in a table with one column named 'dono'. The values listed are Emanuel, Fernando, Luiza, Andrey, Rafael, Dara, and Adler.

dono
Emanuel
Fernando
Luiza
Andrey
Rafael
Dara
Adler

Você pode usar uma cláusula **WHERE** para combinar seleção de registros com seleção de colunas. Por exemplo, para obter a data de nascimento somente dos gatos e cachorros, utilize esta query:

mysql> **SELECT nome, especie, nascimento FROM animais**

-> **WHERE especie = "cachorro" OR especie = "gato";**



The screenshot shows a MySQL query editor window. The query entered is `SELECT nome, especie, nascimento FROM animais WHERE especie = "cachorro" OR especie = "gato";`. The result set is displayed in a table with three columns: 'nome', 'especie', and 'nascimento'. The rows are: pink (cachorro, 2004-08-31), Leque (cachorro, 2005-02-12), Negão (cachorro, 2007-02-03), Kiú (cachorro, 2006-03-10), and Méi (gato, 2008-04-05).

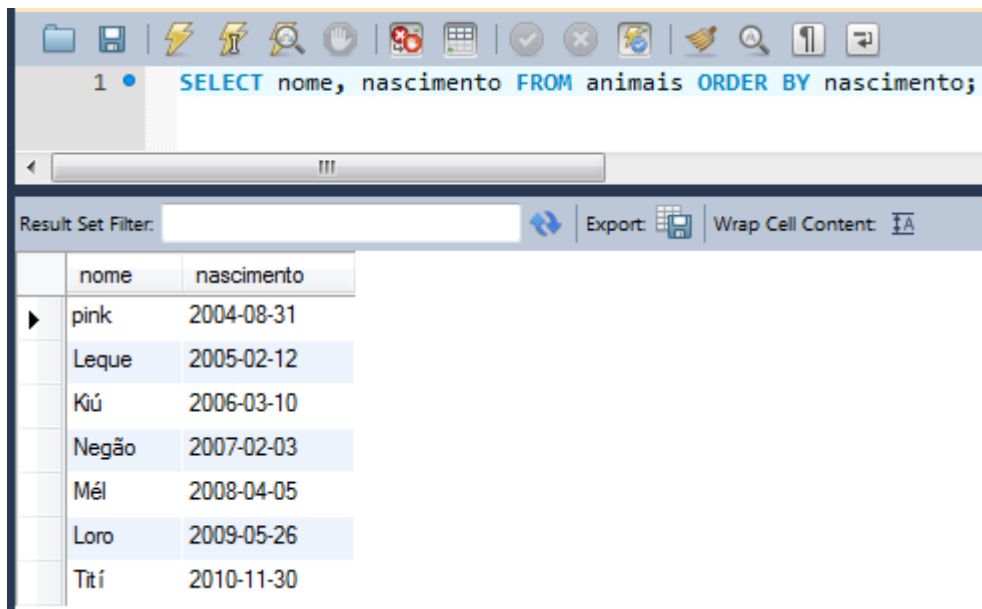
nome	especie	nascimento
pink	cachorro	2004-08-31
Leque	cachorro	2005-02-12
Negão	cachorro	2007-02-03
Kiú	cachorro	2006-03-10
Méi	gato	2008-04-05

61. Ordenando Registros

Você deve ter percebido nos exemplos anteriores que os registros retornados não são mostrados de forma ordenada. Normalmente é mais fácil examinar a saída da consulta quando os registros são ordenados com algum sentido. Para ordenar o resultado, utilize uma cláusula **ORDER BY**.

Aqui está o dia de nascimento dos animais, ordenado por data:

```
mysql> SELECT nome, nascimento FROM animais ORDER BY nascimento;
```

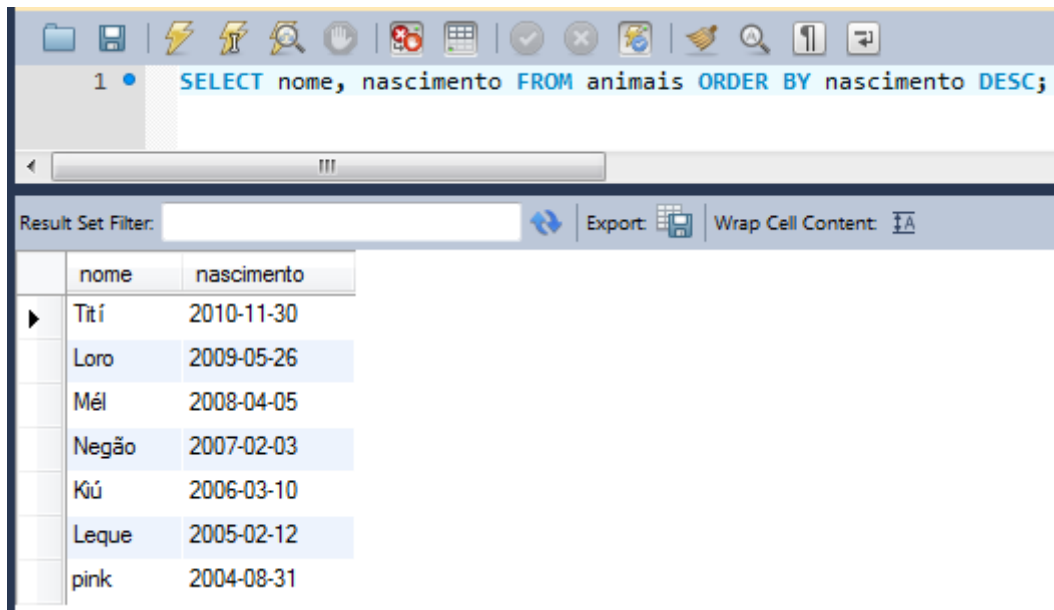
A screenshot of a MySQL graphical user interface. At the top, a toolbar contains various icons for file operations and editing. Below the toolbar, a text area displays the SQL query: `SELECT nome, nascimento FROM animais ORDER BY nascimento;`. Underneath the query area is a horizontal scrollbar. Below the scrollbar, there is a section for 'Result Set Filter' with an empty input field and buttons for 'Export' and 'Wrap Cell Content'. The main area of the window displays a table with two columns: 'nome' and 'nascimento'. The table contains seven rows of data, sorted by the birth date in ascending order. The first row is highlighted with a mouse cursor.

	nome	nascimento
▶	pink	2004-08-31
	Leque	2005-02-12
	Kiú	2006-03-10
	Negão	2007-02-03
	Mél	2008-04-05
	Loro	2009-05-26
	Tití	2010-11-30

Em colunas de tipo de caracter, ordenação como qualquer outra operação de comparação é normalmente realizada no modo caso insensitivo. Isto significa que a ordem será indefinida para colunas que são idênticas exceto quanto ao caso da letra. Você pode forçar uma ordenação em caso sensitivo para uma coluna usando a coerção BINARY: **ORDER BY BINARY(campo)**.

A ordenação padrão é crescente, com os valores menores em primeiro. Para ordenação na ordem reversa, adicione a palavra chave **DESC** (descendente) ao nome da coluna que deve ser ordenada:

```
mysql> SELECT nome, nascimento FROM animais ORDER BY nascimento  
DESC;
```



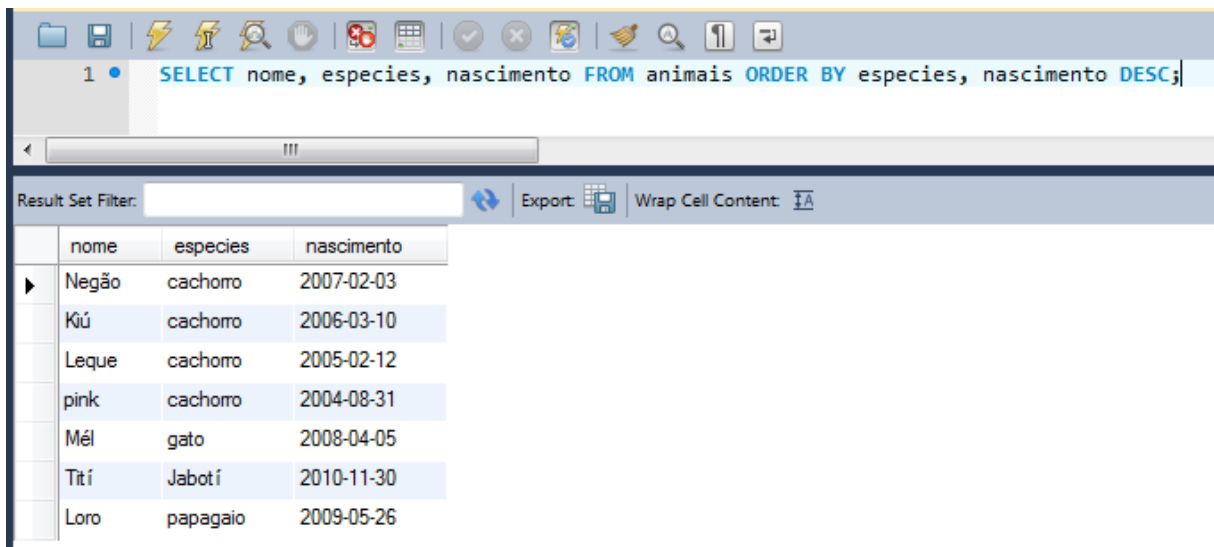
1 • `SELECT nome, nascimento FROM animais ORDER BY nascimento DESC;`

Result Set Filter: Export: Wrap Cell Content:

	nome	nascimento
▶	Tití	2010-11-30
	Loro	2009-05-26
	Mél	2008-04-05
	Negão	2007-02-03
	Kiú	2006-03-10
	Leque	2005-02-12
	pink	2004-08-31

Você pode ordenar por múltiplas colunas e você pode classificar colunas em direções diferentes. Por exemplo, para ordenar o tipo de animal em ordem crescente, depois por dia de nascimento dentro do tipo de animal em ordem decrescente (com os mais novos primeiro), utilize a seguinte consulta:

mysql> `SELECT nome, especies, nascimento FROM animais ORDER BY especies, nascimento DESC;`



1 • `SELECT nome, especies, nascimento FROM animais ORDER BY especies, nascimento DESC;`

Result Set Filter: Export: Wrap Cell Content:

	nome	especies	nascimento
▶	Negão	cachorro	2007-02-03
	Kiú	cachorro	2006-03-10
	Leque	cachorro	2005-02-12
	pink	cachorro	2004-08-31
	Mél	gato	2008-04-05
	Tití	Jabotí	2010-11-30
	Loro	papagaio	2009-05-26

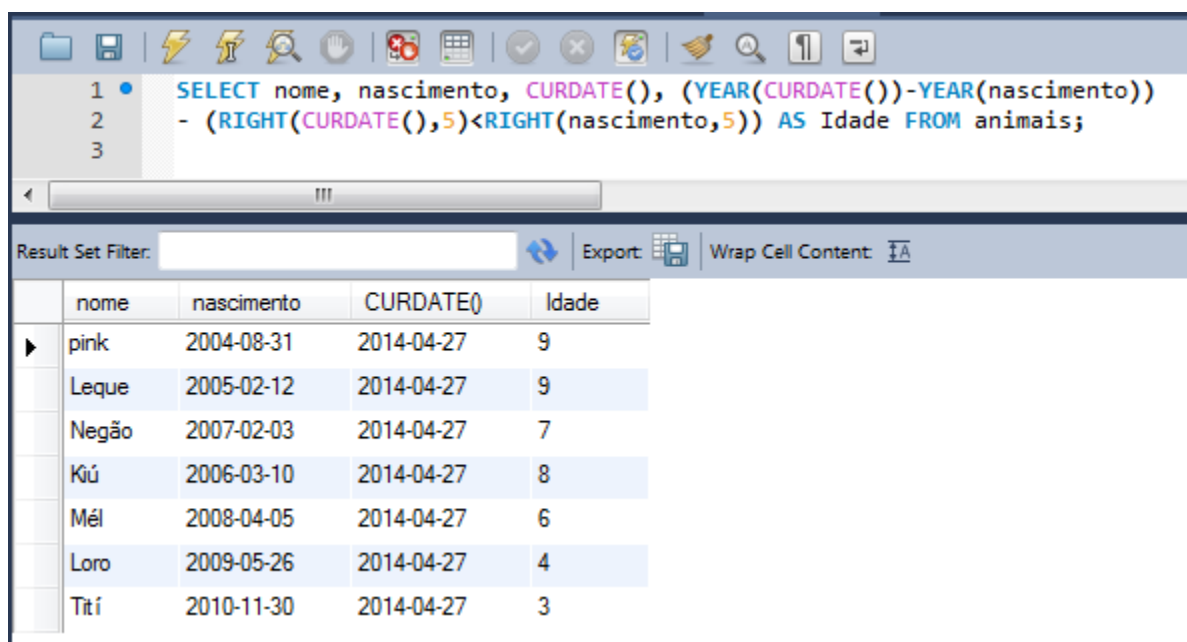
Perceba que a palavra chave **DESC** aplica somente para o nome da coluna precedente (**nascimento**); ela não afeta a ordenação da coluna **especies**.

62. Cálculo de Datas

O MySQL fornece várias funções que você pode usar para realizar cálculos em datas, por exemplo, para calcular idades ou extrair partes de datas.

Para determinar quantos anos cada um do seus animais tem, compute a diferença do ano da data atual e a data de nascimento (**nascimento**), depois subtraia se a o dia/mês da data atual for anterior ao dia/mês da data de nascimento. A consulta seguinte, mostra, para cada animal, a data de nascimento, a data atual e a idade em anos.

```
mysql> SELECT nome, nascimento, CURDATE(),  
-> (YEAR(CURDATE())-YEAR(nascimento))  
-> - (RIGHT(CURDATE(),5)<RIGHT(nascimento,5)) AS Idade  
-> AS Idade  
-> FROM animais;
```



The screenshot shows a MySQL query editor window. The query is as follows:

```
SELECT nome, nascimento, CURDATE(), (YEAR(CURDATE())-YEAR(nascimento))  
- (RIGHT(CURDATE(),5)<RIGHT(nascimento,5)) AS Idade FROM animais;
```

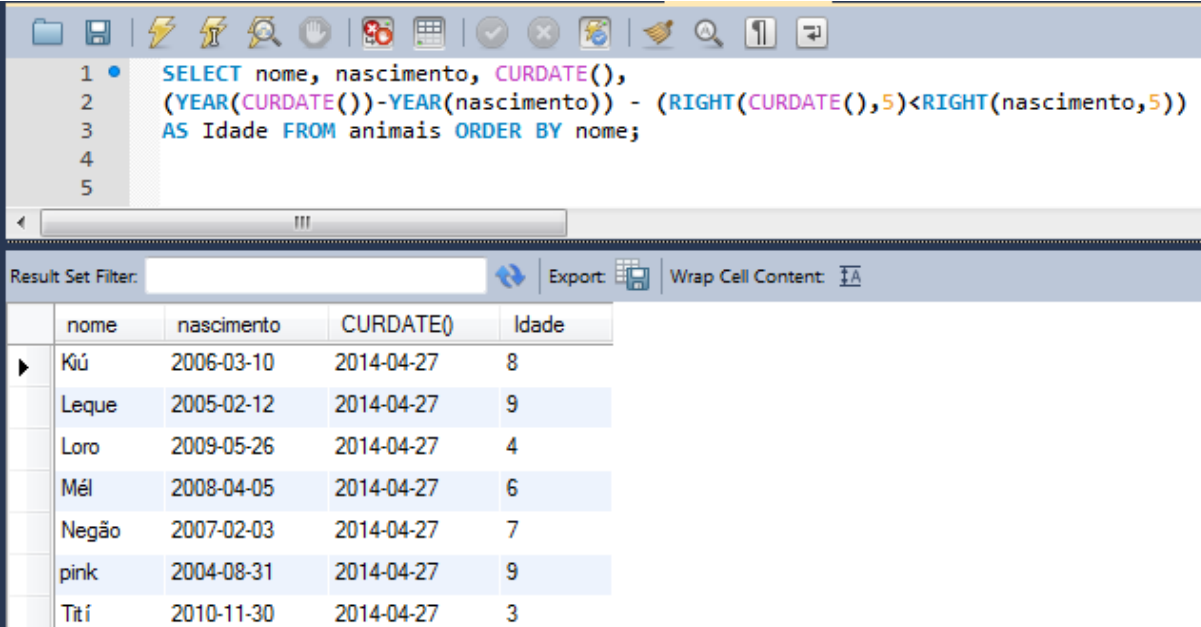
The results are displayed in a table with the following columns: nome, nascimento, CURDATE(), and Idade.

nome	nascimento	CURDATE()	Idade
pink	2004-08-31	2014-04-27	9
Leque	2005-02-12	2014-04-27	9
Negão	2007-02-03	2014-04-27	7
Kiú	2006-03-10	2014-04-27	8
Mél	2008-04-05	2014-04-27	6
Loro	2009-05-26	2014-04-27	4
Tití	2010-11-30	2014-04-27	3

Aqui, **YEAR()** separa a parte do ano de uma data e **RIGHT()** separa os cinco caracteres mais a direita que representam a parte da data **MM-DD**. A parte da expressão que compara os valores **MM-DD** resulta em 1 ou 0, o qual ajusta a diferença do ano um ano abaixo se **CURDATE** ocorrer mais cedo, no ano, que **nascimento**. A expressão completa é um tanto deslegante, então um apelido

(**Idade**) é usado para obter uma saída mais significativa. A consulta funciona, mas o resultado pode ser mais compreensível se os registros forem apresentados em alguma ordem. Isto pode ser feito adicionando uma cláusula **ORDER BY nome** para ordenar a saída pelo nome:

```
mysql> SELECT nome, nascimento, CURDATE(),  
-> (YEAR(CURDATE())-YEAR(nascimento))  
-> - (RIGHT(CURDATE(),5)<RIGHT(nascimento,5))  
-> AS Idade  
-> FROM animais ORDER BY nome;
```



The screenshot shows a MySQL query editor window. The query is as follows:

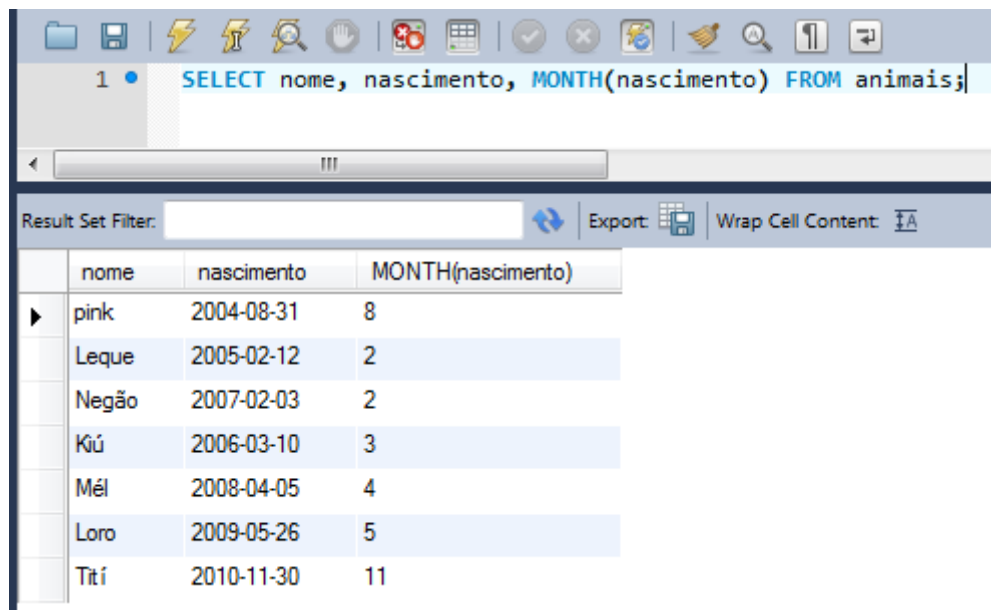
```
1 SELECT nome, nascimento, CURDATE(),  
2 (YEAR(CURDATE())-YEAR(nascimento)) - (RIGHT(CURDATE(),5)<RIGHT(nascimento,5))  
3 AS Idade FROM animais ORDER BY nome;  
4  
5
```

Below the query editor, the results are displayed in a table. The table has five columns: nome, nascimento, CURDATE(), and Idade. The data is sorted by nome.

	nome	nascimento	CURDATE()	Idade
▶	Kiú	2006-03-10	2014-04-27	8
	Leque	2005-02-12	2014-04-27	9
	Loro	2009-05-26	2014-04-27	4
	Mél	2008-04-05	2014-04-27	6
	Negão	2007-02-03	2014-04-27	7
	pink	2004-08-31	2014-04-27	9
	Tití	2010-11-30	2014-04-27	3

E se você desejar saber quais animais fazem aniversário no próximo mês? Para este tipo de cálculo, ano e dia são irrelevantes; você simplesmente deseja extrair a parte do mês da coluna **nascimento**. O MySQL fornece diversas funções para extrair partes da data, como em **YEAR()**, **MONTH()** e **DAYOFMONTH()**. **MONTH** é a função apropriada aqui. Para ver como ela funciona, execute uma consulta simples que mostre o valor de **nascimento** e **MONTH(nascimento)**:

```
mysql> SELECT nome, nascimento, MONTH(nascimento) FROM animais;
```



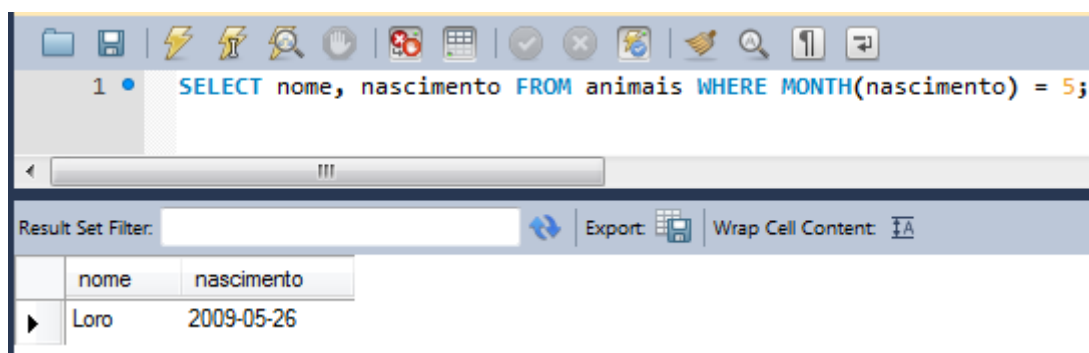
The screenshot shows a MySQL query editor with the following query: `SELECT nome, nascimento, MONTH(nascimento) FROM animais;`

The results are displayed in a table with the following columns: `nome`, `nascimento`, and `MONTH(nascimento)`.

nome	nascimento	MONTH(nascimento)
pink	2004-08-31	8
Leque	2005-02-12	2
Negão	2007-02-03	2
Kiú	2006-03-10	3
Mél	2008-04-05	4
Loro	2009-05-26	5
Tití	2010-11-30	11

Encontrar animais com aniversário no próximo mês também é fácil. Suponha que o mês atual é abril. Então o valor do mês é `4` e você procura por animais nascidos em Maio (mês `5`) assim:

```
mysql> SELECT nome, nascimento FROM animais WHERE MONTH(nascimento) = 5;
```



The screenshot shows a MySQL query editor with the following query: `SELECT nome, nascimento FROM animais WHERE MONTH(nascimento) = 5;`

The results are displayed in a table with the following columns: `nome` and `nascimento`.

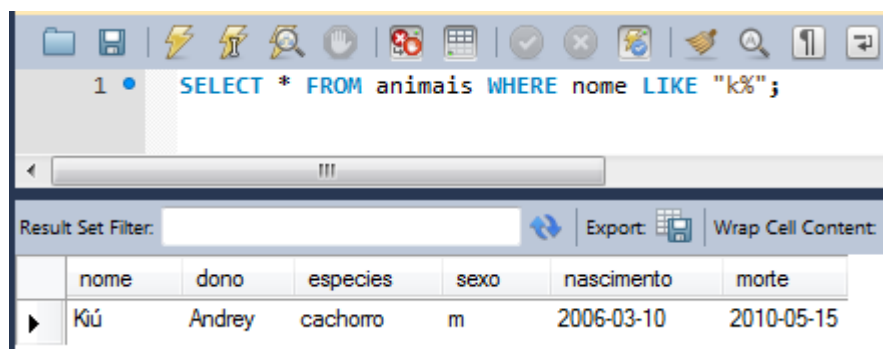
nome	nascimento
Loro	2009-05-26

63. Combinação de padrões

O MySQL fornece combinação de padrões do SQL bem como na forma de combinação de padrões baseado nas expressões regulares estendidas similares àquelas usadas pelos utilitários Unix como o `vi`, `grep` e `sed`. 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). No MySQL, padrões SQL são caso insensitivo por padrão. Alguns exemplos são vistos

abaixo. Perceba que você não usa **=** ou **!=** quando usar padrões SQL; use os operadores de comparação **LIKE** ou **NOT LIKE** neste caso. Para encontrar nomes começando com 'k':

```
mysql> SELECT * FROM animais WHERE nome LIKE "k%";
```

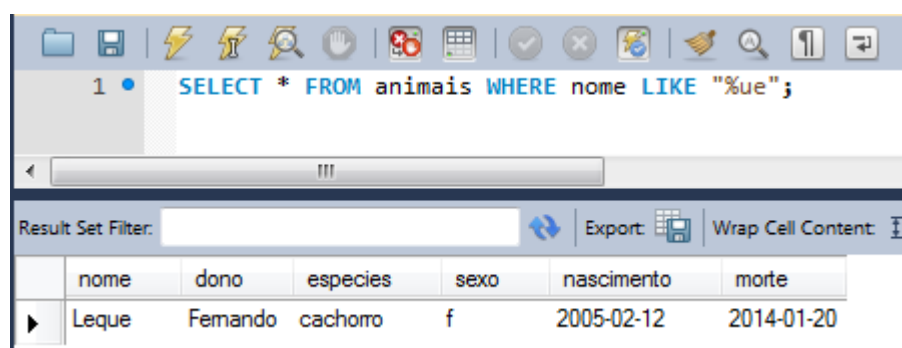


The screenshot shows a MySQL GUI window with a toolbar at the top. Below the toolbar, the query editor contains the text: `1 • SELECT * FROM animais WHERE nome LIKE "k%";`. Below the query editor is a horizontal scrollbar. Underneath is a 'Result Set Filter' field with a blue refresh icon and buttons for 'Export' and 'Wrap Cell Content'. The result set is displayed in a table with the following data:

	nome	dono	especies	sexo	nascimento	morte
▶	Kiú	Andrey	cachorro	m	2006-03-10	2010-05-15

Para encontrar nomes com o final 'ue':

```
mysql> SELECT * FROM animais WHERE nome LIKE "%ue";
```

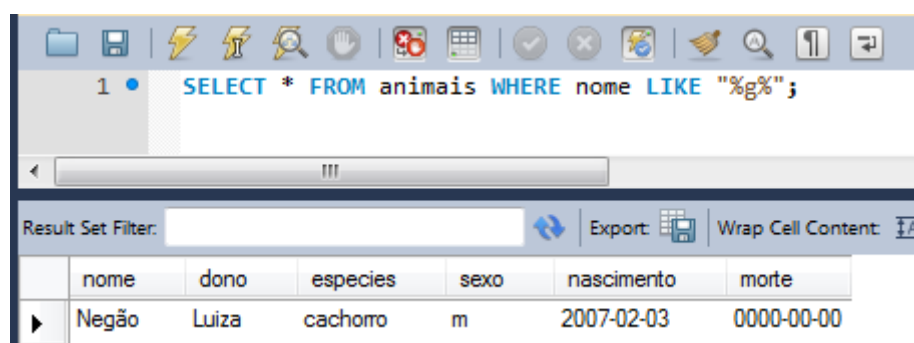


The screenshot shows a MySQL GUI window with a toolbar at the top. Below the toolbar, the query editor contains the text: `1 • SELECT * FROM animais WHERE nome LIKE "%ue";`. Below the query editor is a horizontal scrollbar. Underneath is a 'Result Set Filter' field with a blue refresh icon and buttons for 'Export' and 'Wrap Cell Content'. The result set is displayed in a table with the following data:

	nome	dono	especies	sexo	nascimento	morte
▶	Leque	Fernando	cachorro	f	2005-02-12	2014-01-20

Para encontrar nomes contendo um 'g':

```
mysql> SELECT * FROM animais WHERE nome LIKE "%g%";
```

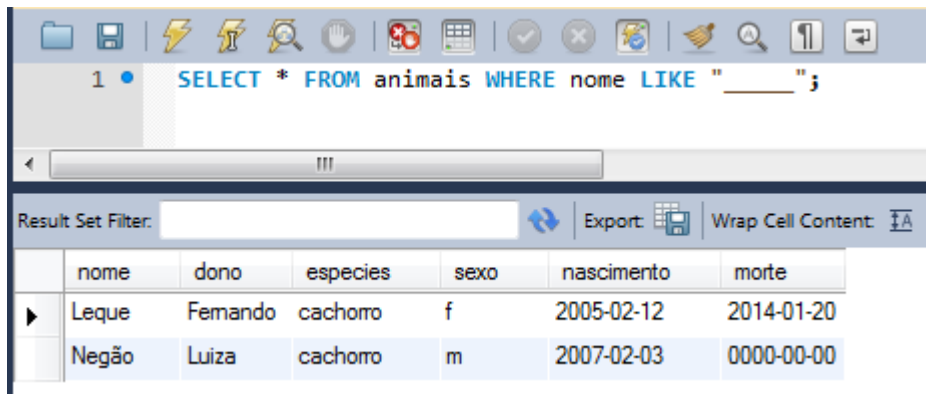


The screenshot shows a MySQL GUI window with a toolbar at the top. Below the toolbar, the query editor contains the text: `1 • SELECT * FROM animais WHERE nome LIKE "%g%";`. Below the query editor is a horizontal scrollbar. Underneath is a 'Result Set Filter' field with a blue refresh icon and buttons for 'Export' and 'Wrap Cell Content'. The result set is displayed in a table with the following data:

	nome	dono	especies	sexo	nascimento	morte
▶	Negão	Luiza	cachorro	m	2007-02-03	0000-00-00

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

```
mysql> SELECT * FROM animais WHERE nome LIKE "_____";
```



1 • `SELECT * FROM animais WHERE nome LIKE "_____";`

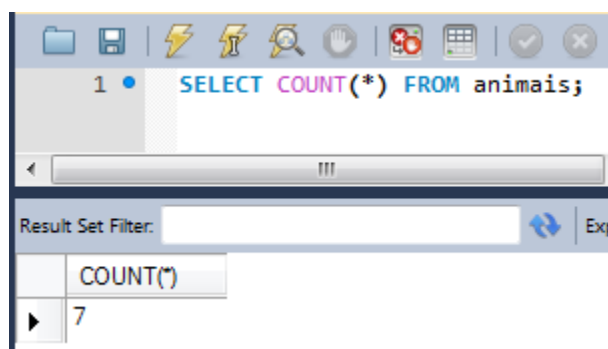
Result Set Filter: Export Wrap Cell Content

	nome	dono	especies	sexo	nascimento	morte
▶	Leque	Fernando	cachorro	f	2005-02-12	2014-01-20
	Negão	Luiza	cachorro	m	2007-02-03	0000-00-00

64. Contando Registros

Bancos de dados normalmente são usados para responder a perguntas, “Qual a frequência que certo tipo de dados ocorre em uma tabela?” Por exemplo, você deve querer saber quantos animais tem, ou quantos animais cada dono tem, ou você pode querer fazer vários outros tipos de operações de censo com seus animais. Contando o número total de animais que você tem é a mesma questão como em “Quantos registros existem na tabela **animais**?” porque existe um registro por animal. **COUNT(*)** conta o número de resultados **NOT-NULL**, portanto a pesquisa para contar seus animais parecerá com isto:

mysql> **SELECT COUNT(*) FROM animais;**



1 • `SELECT COUNT(*) FROM animais;`

Result Set Filter: Export

	COUNT(*)
▶	7

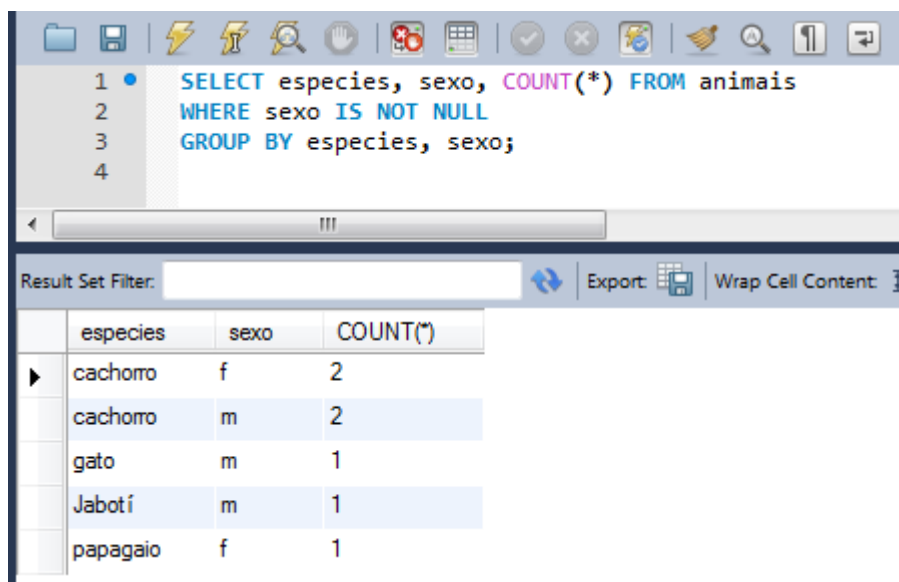
Logo, você recuperará os nomes das pessoas que possuam animais. Você pode usar **COUNT()** se você desejar encontrar quantos animais cada dono possui:

mysql> **SELECT dono, COUNT(*) FROM animais GROUP BY dono;**

Perceba o uso de **GROUP BY** para agrupar todos os registros para cada dono.

Ou se você deseja saber o número de animais por sexo somente de animais com sexo conhecido:

```
mysql> SELECT especies, sexo, COUNT(*) FROM animais
WHERE sexo IS NOT NULL
GROUP BY especies, sexo;
```



The screenshot shows a MySQL query editor window. The query is: `SELECT especies, sexo, COUNT(*) FROM animais WHERE sexo IS NOT NULL GROUP BY especies, sexo;`. The results are displayed in a table with the following data:

	especies	sexo	COUNT(*)
▶	cachorro	f	2
	cachorro	m	2
	gato	m	1
	Jabotí	m	1
	papagaio	f	1

65. Utilizando Múltiplas Tabelas

A tabela **animais** mantém informações de quais animais você tem. Se você deseja gravar outras informações sobre eles como eventos em suas vidas, tais como visitas ao veterinário ou sobre suas crias, você necessitará de outra tabela. Como esta tabela deve se parecer? Ela precisa:

- Conter o nome do animal para que você saiba a qual animal pertence o evento.
- Uma data para que você saiba quando ocorreu o evento.
- Um campo para descrever o evento.
- Um campo com o tipo de evento, se você desejar classificá-los por categoria.

Dadas estas considerações, a instrução **CREATE TABLE** para a tabela **evento** deve se parecer com isto:

```
CREATE TABLE evento (nome VARCHAR(20), data DATE,
tipo VARCHAR(15), descricao VARCHAR(255));
```

Como na tabela **animais**, é mais fácil carregar os registros iniciais criando um arquivo texto delimitado por tabulações contendo a informação:

Inseri registros na tabela criada:

INSERT INTO evento

(nome, data, tipo, descricao) VALUES

```
('pink','2005-12-20','ninhada','4 cachorrinhos, 3 do sexo feminino, 1 masculino'),
('pink','2006-12-20','aniversário','Primeiro aniversário'),
('Leque','2006-03-05','ninhada','5 filhotes, 2 fêmeas, 3 machos'),
('Leque','2007-02-15','ninhada','3 filhotes, 2 fêmeas'),
('Leque','2008-02-15','aniversário','Ganhou um brinquedo de mastigar'),
('Negão','2007-09-01','veterinário','vacina contra raiva'),
('Kiú','2007-12-25','veterinário','costela quebrada'),
('Mél','2008-11-30','veterinário','castração'),
('Kiú','2008-12-15','Canil',''),
('pink','2009-12-20','Canil',''),
('Leque','2010-12-20','aniversário','Ganhou um brinquedo de mastigar');
```

Suponha que você deseje descobrir as idades de cada animal quando eles tiveram cria. Nós vemos logo que é possível calcular a idade a partir das duas datas. A idade dos filhotes está na tabela **evento**, mas para calcular a idade da mãe, você precisará da data de nascimento dela, que está armazenado na tabela **animais**. Isto significa que você precisará das duas tabelas para a consulta:

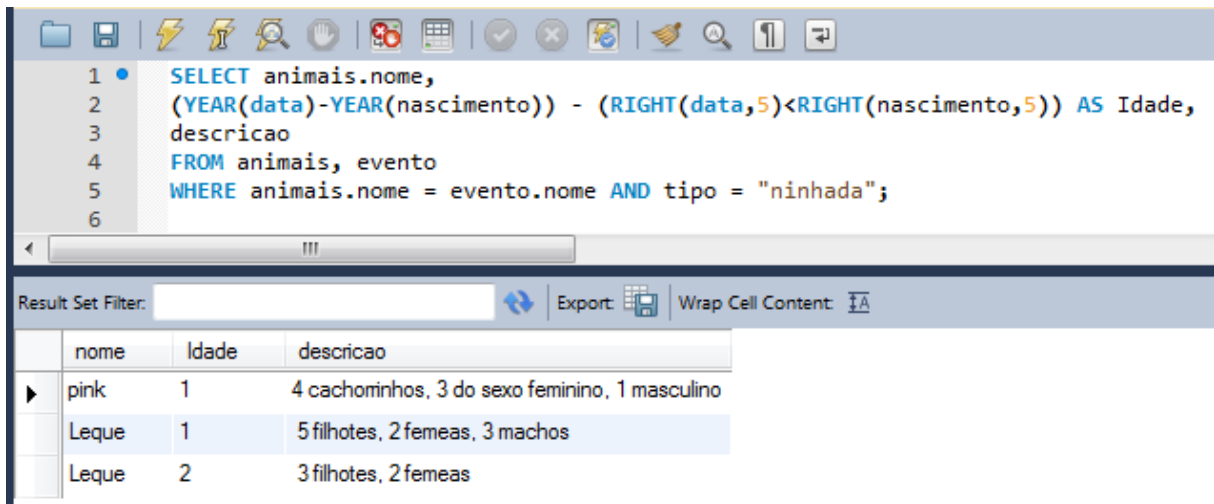
```
mysql> SELECT animais.nome,
```

```
-> (YEAR(data)-YEAR(nascimento)) - (RIGHT(data,5)<RIGHT(nascimento,5)) AS  
Idade,
```

```
-> descricao
```

```
-> FROM animais, evento
```

```
-> WHERE animais.nome = evento.nome AND tipo = "ninhada";
```



The screenshot shows a MySQL query editor with a SQL query and its results. The query is as follows:

```

1 • SELECT animais.nome,
2     (YEAR(data)-YEAR(nascimento)) - (RIGHT(data,5)<RIGHT(nascimento,5)) AS Idade,
3     descricao
4 FROM animais, evento
5 WHERE animais.nome = evento.nome AND tipo = "ninhada";
6

```

The results are displayed in a table with the following columns: nome, Idade, and descricao.

nome	Idade	descricao
pink	1	4 cachorrinhos, 3 do sexo feminino, 1 masculino
Leque	1	5 filhotes, 2 fêmeas, 3 machos
Leque	2	3 filhotes, 2 fêmeas

Existem várias coisas que devem ser percebidas sobre esta consulta:

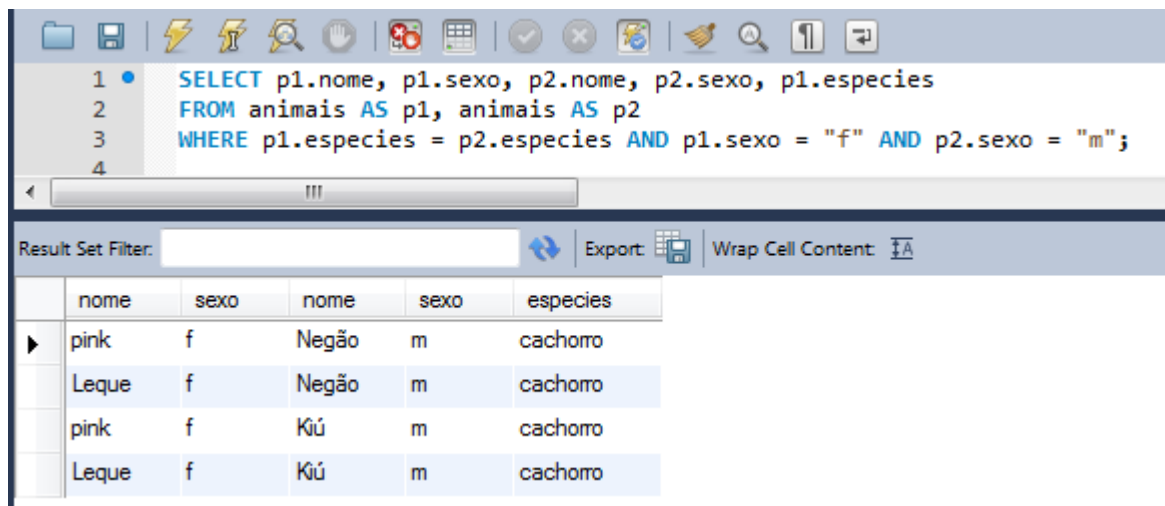
- A cláusula **FROM** lista as duas tabelas porque a consulta precisa extrair informação de ambas.
- Quando combinar (unir) informações de múltiplas tabelas, você precisa especificar como registros em uma tabela podem ser coincidadas com os registros na outra. Isto é simples porque ambas possuem uma coluna **nome**. A consulta utiliza a cláusula **WHERE** para coincidir registros nas duas tabelas baseadas nos valores de **nome**.
- Como a coluna **nome** ocorre em ambas tabelas, você deve especificar qual a tabela a que você está se referindo. Isto é feito usando o nome da tabela antes do nome da coluna separados por um ponto (.).

Você não precisa ter duas tabelas diferentes para realizar uma união. Algumas vezes é útil unir uma tabela a ela mesma, se você deseja comparar registros em uma tabela com outros registros na mesma tabela. Por exemplo, para encontrar pares entre seus animais, você pode unir a tabela **animais** com ela mesma para produzir pares candidatos de machos e fêmeas de acordo com as espécies:

```

mysql> SELECT p1.nome, p1.sexo, p2.nome, p2.sexo, p1.especies
-> FROM animais AS p1, animais AS p2
-> WHERE p1.especies = p2.especies AND p1.sexo = "f" AND p2.sexo = "m";

```



The screenshot shows a MySQL query editor with a toolbar at the top. The query is as follows:

```

1 SELECT p1.nome, p1.sexo, p2.nome, p2.sexo, p1.especies
2 FROM animais AS p1, animais AS p2
3 WHERE p1.especies = p2.especies AND p1.sexo = "f" AND p2.sexo = "m";
4

```

Below the query editor, there is a "Result Set Filter:" field and buttons for "Export" and "Wrap Cell Content". The results are displayed in a table with 5 columns: nome, sexo, nome, sexo, and especies.

	nome	sexo	nome	sexo	especies
▶	pink	f	Negão	m	cachorro
	Leque	f	Negão	m	cachorro
	pink	f	Kiú	m	cachorro
	Leque	f	Kiú	m	cachorro

Nesta consulta, nós especificamos apelidos para os nomes das tabelas para conseguir referenciar às colunas e manter com qual instância da tabela cada coluna de referência está associada.

66. Exemplos de Consultas Comuns

Alguns dos exemplos usam a tabela **loja** para armazenar o preço de cada **item** para certas **revenda**. Supondo que cada revenda tenha um preço fixo por artigo, então (**artigo**, **revenda**) é uma chave primária para os registros.

Você pode criar e popular a tabela exemplo assim:

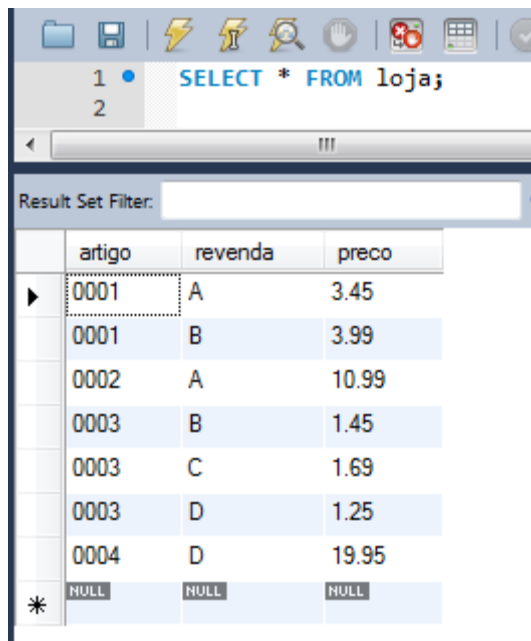
```

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

```

Depois de executar as instruções a tabela deve ter o seguinte conteúdo:

mysql> **SELECT * FROM loja;**



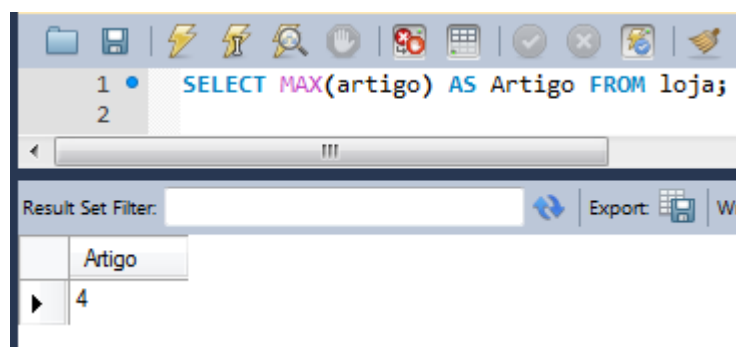
The screenshot shows a MySQL GUI window with a toolbar at the top. Below the toolbar, there are two tabs: '1' and '2'. Tab '1' is selected and contains the SQL query `SELECT * FROM loja;`. Below the query editor, there is a 'Result Set Filter' field. The main area displays a table with three columns: 'artigo', 'revenda', and 'preco'. The table contains eight rows of data, followed by a row with three NULL values. A vertical scrollbar is visible on the left side of the table.

	artigo	revenda	preco
▶	0001	A	3.45
	0001	B	3.99
	0002	A	10.99
	0003	B	1.45
	0003	C	1.69
	0003	D	1.25
	0004	D	19.95
*	NULL	NULL	NULL

67. O Valor Máximo para uma Coluna

``Qual é o maior número dos itens?''

mysql> **SELECT MAX(artigo) AS Artigo FROM loja;**



The screenshot shows a MySQL GUI window with a toolbar at the top. Below the toolbar, there are two tabs: '1' and '2'. Tab '1' is selected and contains the SQL query `SELECT MAX(artigo) AS Artigo FROM loja;`. Below the query editor, there is a 'Result Set Filter' field. The main area displays a table with one column: 'Artigo'. The table contains one row with the value '4'. A vertical scrollbar is visible on the left side of the table.

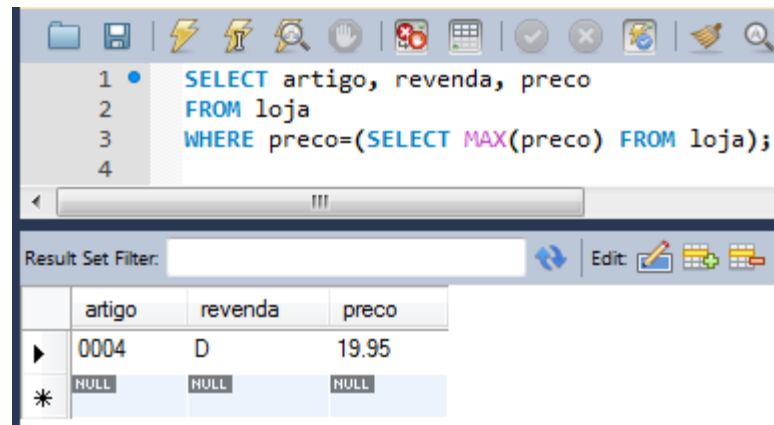
	Artigo
▶	4

68. O Registro que Armazena o Valor Máximo para uma Coluna Determinada

``Encontre o número, fornecedor e preço do item mais caro.''

mysql> **SELECT artigo, revenda, preco**
-> **FROM loja**

-> **WHERE preco=(SELECT MAX(preco) FROM loja);**



Outra solução é ordenar todos os registros por preço de forma descendente e obtenha somente o primeiro registro utilizando a cláusula específica do MySQL **LIMIT**:

mysql> **SELECT artigo, revenda, preco**

-> **FROM loja**

-> **ORDER BY preco DESC**

-> **LIMIT 1;**

NOTA: Se existir diversos itens mais caros, cada um com um preço de 19.95, a solução **LIMIT** mostra somente um deles !

69. Calculando o Menor Valor De Uma Coluna

“Qual é o menor preço na coluna preço da tabela loja?”

mysql> **SELECT MIN(preco) FROM loja;**

70. Calculando a Média

“Qual a média dos preços da tabela loja?”


```
mysql> SELECT AVG(preco) FROM loja;
```

71. Somando Os Valores De Uma Coluna

“Qual o valor total da coluna preço na tabela loja?”

```
Mysql> SELECT SUM(preco) FROM loja;
```

72. \$POST

O post serve para capturar uma variáveis de uma página php (para nossa disciplina pasta saber que ele irá capturar os dados que o usuário irá digitar no formulário)

```
<?php
$nome=($_POST["name"]);
?>
```

73. Conectando no banco

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
    die('Não foi possível conectar: ' . mysql_error());
}
echo 'Conexão bem sucedida';
mysql_close($link);
?>
```

```
// 'localhost' - Servidor onde seu banco está
// 'mysql_user' - Usuário do banco - Normalmente 'root'
// 'mysql_password' - Senha desse usuário - O padrão é vazio
```

74. Acho que é selecionando o banco

```
$db_selected = mysql_select_db('foo', $link);  
if (!$db_selected) {  
    die ('Can\'t use foo : ' . mysql_error());  
}
```

`$db_selected` - Nome da conexão
`mysql_select_db` - Nome do banco
`'foo'` - Nome da coluna

75. DCL Controle

Sintaxe de criação de usuário

```
CREATE USER 'nome_do_usuario'@'nome_do_servidor' IDENTIFIED BY 'senha';  
CREATE USER 'rubem'@'localhost' IDENTIFIED BY '123';
```

Assim que criamos o usuário, ele não possui permissão nenhuma no banco.

Dando permissões

```
mysql> GRANT ALL PRIVILEGES ON *.* TO rubem@localhost;
```

Estamos dando privilégio total para o usuário rubem

```
GRANT privilegios_a_serem_concedidos ON nome_do_banco.nome_da_tabela TO  
'nome_do_usuario'@'nome_do_servidor';
```

Privilégios a serem concedidos:

ALL PRIVILEGES - Privilégio total
CREATE - Criação de base de dados e tabelas
DROP - Deletar base de dados e tabelas
DELETE - Deletar linhas das tabelas
INSERT - Inserir linhas nas tabelas
SELECT - Permite o select
UPDATE - Permite update
GRANT OPTION - Permite conceder ou negar permissões a outros usuários

```
REVOKE [permissão] ON [nome_do_banco_de_dados].[nome_da_tabela] FROM  
['nome_do_usuario']@['nome_do_servidor'];
```

Para deletar um usuário

```
DROP USER 'nome_do_usuario'@'nome_do_servidor';
```

Obs. Sempre que fizer alguma atualização de privilégio do usuário use o comando:

```
FLUSH PRIVILEGES;
```

Para recarregar os privilégios dos usuários do banco

Para testar o usuário no terminal.

```
mysql -u [nome_do_usuario]-p
```

76. Inserir Transações

Observação só funciona com tabela tipo InnoDB, tabelas tipo MyISAM não permitem RollBack.

Para resolver, rode o script abaixo

```
ALTER TABLE nome_da_tabela engine = InnoDB;
```

start transaction; // Cria um ponteiro para o log do banco. Armazenando a informação em uma espécie de memória cache

rollback; // Voltando a informação, ou desfazendo a transação

commit; // Confirmando a transação

77. Administração do Banco

75.1 Backup e Restore do mysql

Para fazer backup no MySQL através do Workbench. Faça conexão com a base desejada.

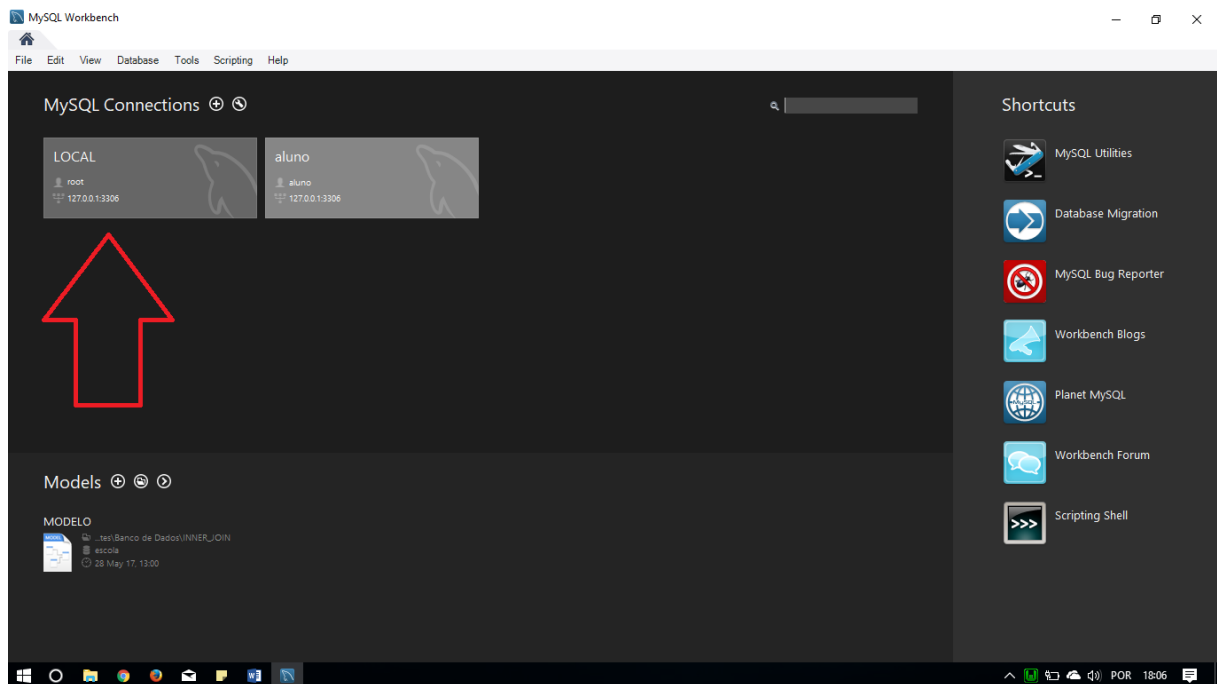


Imagem x. Tela inicial do MySQL Whorkbanck

Clique em “Data Export”

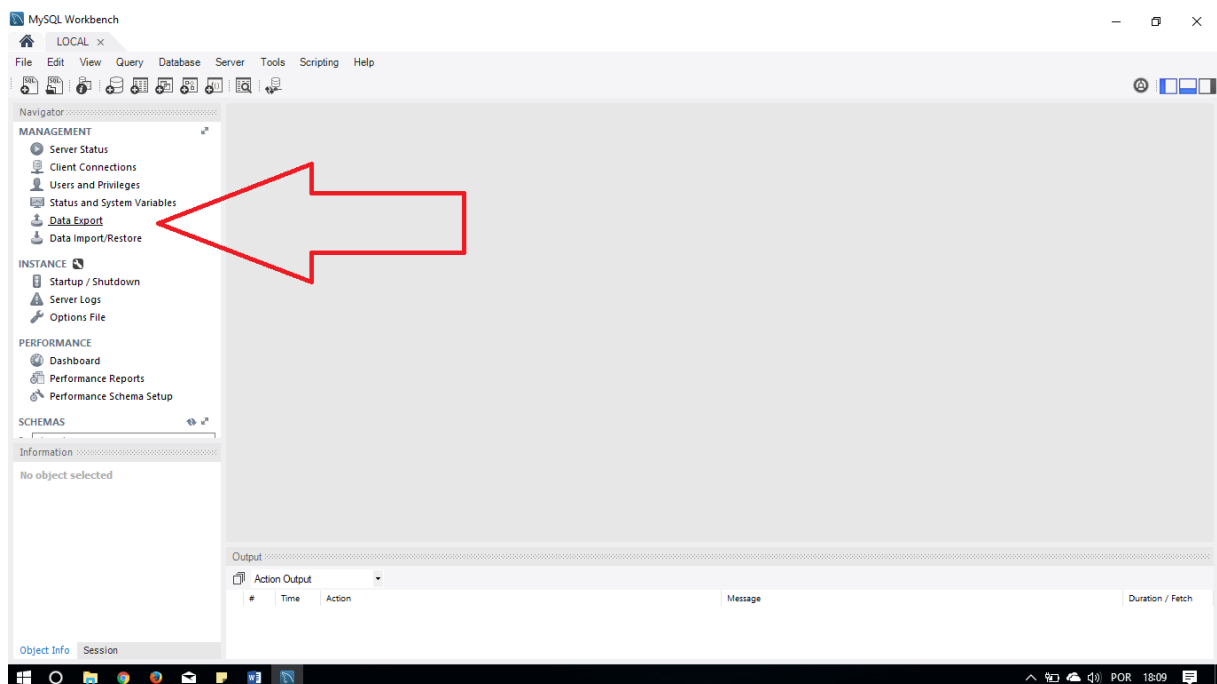


Imagem x. Tela de Administração do MySQL Workbench

Selecione o schema desejado e clique em “Export to Dump Project Folder” – Caso queira exportar o banco separado por suas tabelas.

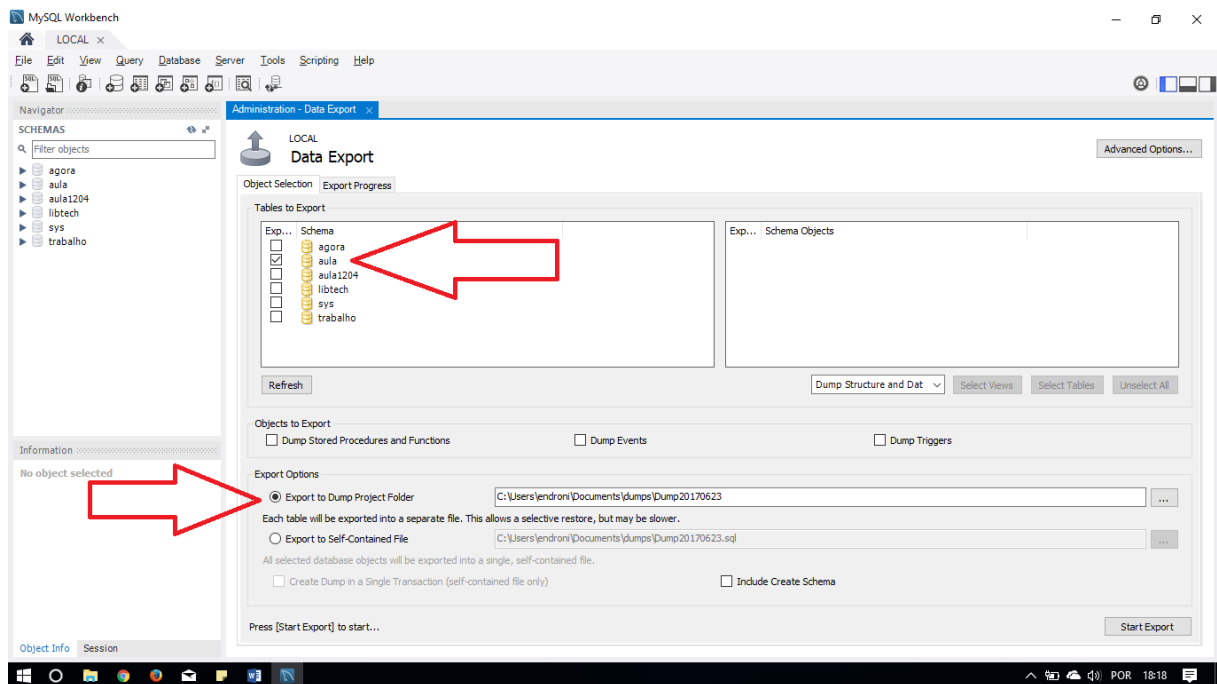


Imagem x. Selecionando o schema

Selecione a opção “**Export to Self-Contained File**” – Caso queira exportar o banco em apenas um arquivo e em seguida em Start Export

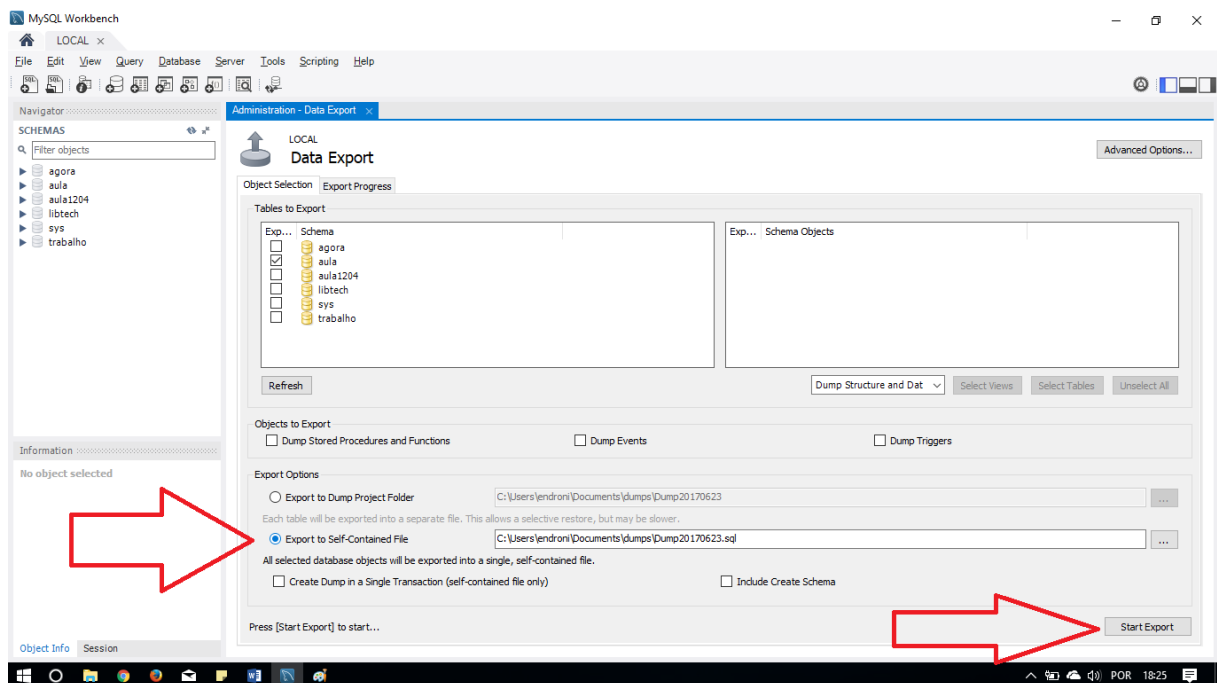


Imagem x. Gerando apenas um arquivo para backup

Agora basta pegar o backup com a extensão .sql no diretório selecionado

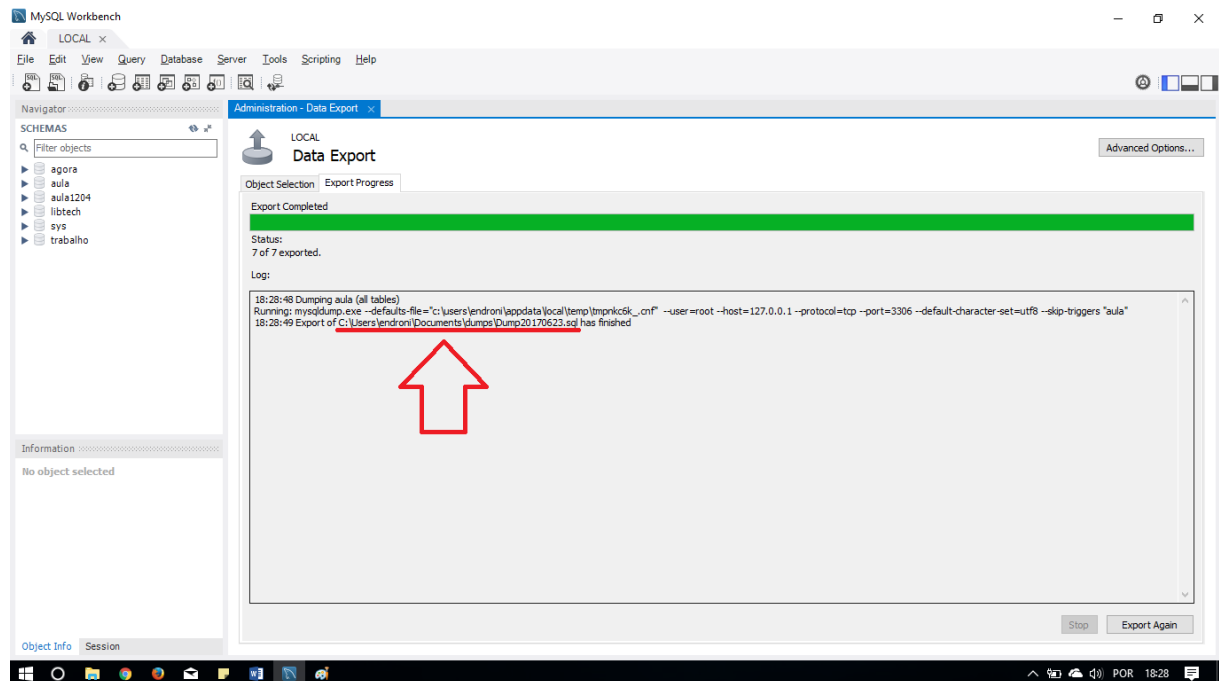


Imagem x. Backup pronto

Para restaurar o backup, basta executar o arquivo no servidor destino

78. Bibliografias

- Projeto de Banco de Dados, 4ª Edição, Instituto de Informática da UFRGS, autor Carlos Alberto Heuser;
- Manual de Referência do MySQL 4.1

<https://dev.mysql.com/doc/refman/4.1/en/>

- Manual de Referência do PHP 4, PHP5

http://php.net/manual/pt_BR/function.mysql-connect.php acessado em 16 de abril de 2016 às 10:23

Ferramentas:

Wamp Server versão 2.4

MySQL Workbench 6.0