

Banco de Dados

Curso de Pós-Graduação
Professor Walter Cunha

Conteúdo

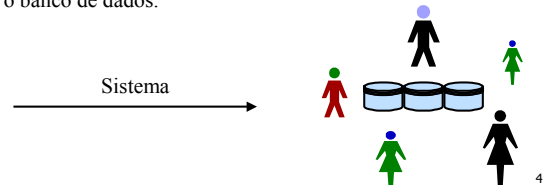
- Conceitos básicos de Banco de Dados.
- Sua organização lógica.
- Sistema de gerenciamento de bancos de dados.
- Modelo conceitual
- Modelagem de dados.
- Normalização.

Introdução

Com a integração econômica mundial, o aumento da competitividade de mercado vem provocando uma revolução em diversos setores administrativos. Neste processo de globalização é imprescindível a qualidade de informações. A internet fez com que a facilidade de obtenção de dados exija cada vez um tratamento elaborado dos mesmos.

Introdução

- Banco de Dados é uma coleção de dados inter-relacionados.
- Sistema de Gerenciamento de Banco de dados (SGBD) é uma coleção de dados inter-relacionados e um conjunto de programas para acessá-los permitindo ao usuário criar e manter o banco de dados.

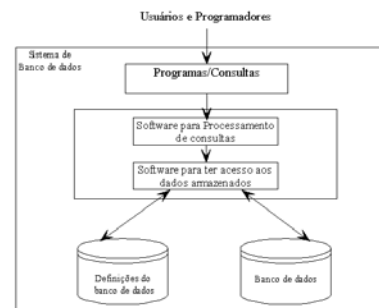


Introdução

Um sistema de banco de dados é um sistema que facilita o processo de definição, construção e manipulação de banco de dados para diversas aplicações.

- **Definição** envolve especificar os tipos e estruturas dos dados que serão armazenados no banco de dados.
- **Construção** é o armazenamento propriamente dito.
- **Manipulação** são funções de acesso, tais como: consultas, atualização e inserção.

SGBD

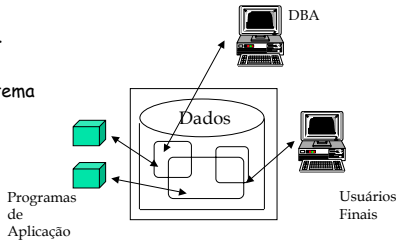


Sistemas de Bancos de Dados

Manutenção de registros por computador
Inclusão, Exclusão, Recuperação, etc.

Componentes do Sistema

- Usuários
- Dados
- Hardware
- Software



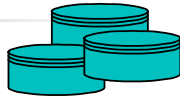
7

Processamento de arquivos

- Redundância e inconsistência de dados
- Dificuldades no acesso a dados independentes
- Isolamento de dados
- Acesso concorrente
- Problemas de segurança
- Problemas de integridade em arquivos distintos

8

Banco de Dados



• **Definição do banco:** O sistema de banco de dados não só contém o banco como também sua definição e descrição envolvendo os tipos de dados, etc.

• **Abstração de dados:** Os programas são escritos independentemente da estrutura de dados de forma que uma mudança nos dados não implica na mudança da aplicação.

9

Banco de Dados



• **Múltiplas visões de dados:** Como um banco de dados possui diversos usuários muitas vezes é necessário ter diferentes visões dos dados armazenados para diferentes propósitos.

• **Compartilhamento de dados e processamento de múltiplas transações:** Permite que múltiplos usuários acessem os dados ao mesmo tempo.

10

Porque usar um SGBD

- Integridade
- Evitar redundância
- Maior visibilidade
- Compactos
- Mais rápidos
- Permitem crescimento controlado
- Maior segurança
- Menos trabalho braçal

11

Vantagens

- Independência de Dados
 - Maior objetivo dos Bancos de Dados

As aplicações são imunes à estrutura dos dados e à estratégia de acesso

- Aplicações diferentes possuem visões diferentes
- O DBA deve ter liberdade para manipular estruturas

12

Software - SGBD

- Suporte às solicitações dos usuários
- Isolar dos recursos do hardware
- Gerenciar recursos do sistema
- Compartilhamento
- Segurança
- Integridade

13

Usuários

- **Administrador do Banco de Dados (DBA)**
 - Padronização, autorização, integridade, desempenho, ligação com usuários.
 - Responsável pela autorização de acesso ao banco de dados sua coordenação e monitoramento. Além disto, é tarefa do DBA analisar tópicos referentes à segurança bem como a compra de software e hardware necessários.

14

Usuários

- **Projetistas de BD**
 - É responsável por identificar os dados a serem armazenados no banco para que sejam escolhidas as estruturas apropriadas na representação e armazenamento dos dados. Esta tarefa é geralmente realizada no momento da implantação do banco.

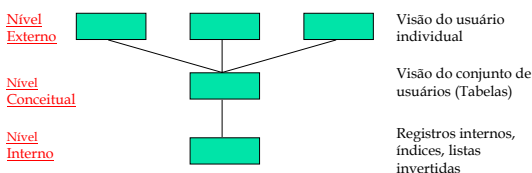
15

Usuários

- **Programadores de Aplicação / Analistas**
 - São responsáveis pela elaboração e implantação de programas específicos para os usuários finais.
- **Usuário Final**
 - Casuais, Paramétricos, Sofisticados ou *Stand-alone*
 - Usa programas de aplicação, LMD(Linguagem de manipulação de dados), QBE(Query By Example), Oracle, Sybase ...
 - São aqueles que utilizam de fato as informações contida no banco de dados.

16

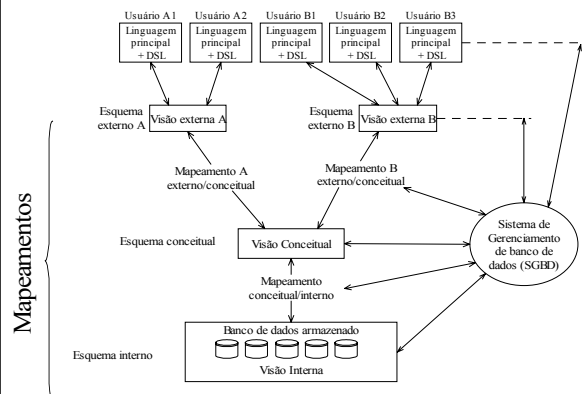
Arquitetura para Sistemas de BD



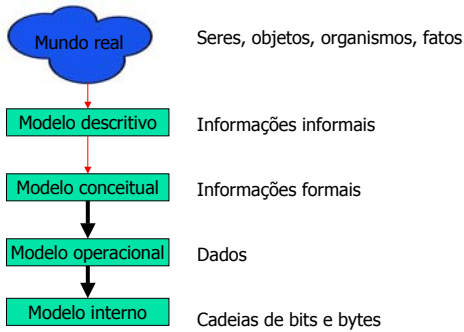
- Cada usuário tem uma linguagem à sua disposição
 - **DDL** (Data definition language)
 - **DML** (Data manipulation language - SQL, QMF, QBE)

17

Arquitetura para Sistemas de BD



Níveis de Abstração



19

Níveis de Abstração



- Definição imprecisa.
- Várias questões filosóficas e percepções ligadas a esse conceito.

Materialistas: apenas o fisicamente material é real

Idealistas: apenas o não físico é real

Monistas: ambos são realidades em planos diferentes

20

Níveis de Abstração

Modelo descritivo

- Relatórios escritos em linguagem natural.
- Descompromisso com formalismo.
- O mais inteligível possível.

21

Níveis de Abstração

Modelo conceitual

- Preparação para o nível computacional (máquina).
- formalismo da matemática.
- Aspectos:
 - Estrutura das informações.
Fornecedor -> endereço -> local, cep, cidade
 - Manipulação das informações.
Atualização de endereços, confecção de um relatório, ...

22

Níveis de Abstração

Modelo operacional

- Compromisso com a linguagem da máquina.
- Linguagem de programação x expressão matemática.
 - Dígito de um CPF
- Modelo conceitual x Modelo operacional.
 - Tendências das linguagens de alto nível.

23

Níveis de Abstração

Modelo interno

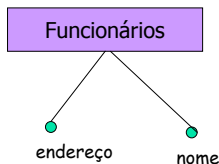
- Linguagem de máquina propriamente dita.
- Cadeias com especificações estruturais.
- Cadeias correspondentes aos dados.

24

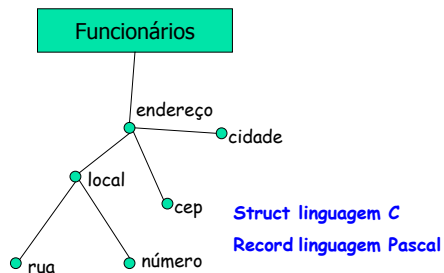
Modelo Entidade-Relacionamento

Atributos:

Não faz sentido definir um objeto se não identificarmos um conjunto de valores para ele. Ao conjuntos de valores de um objeto chamamos de atributos.

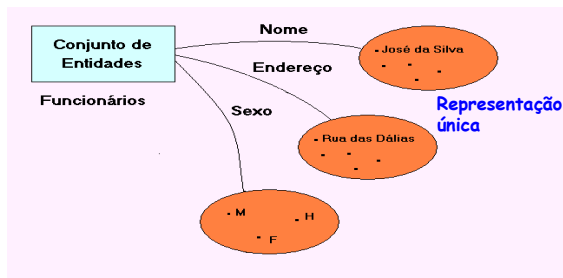


Sub-atributos (atributo composto)

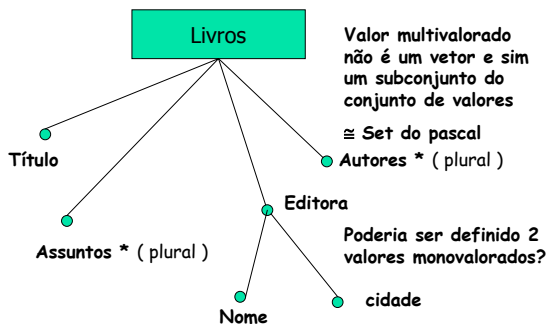


Struct linguagem C
Record linguagem Pascal

Modelo conceitual



Atributos Multivalorados

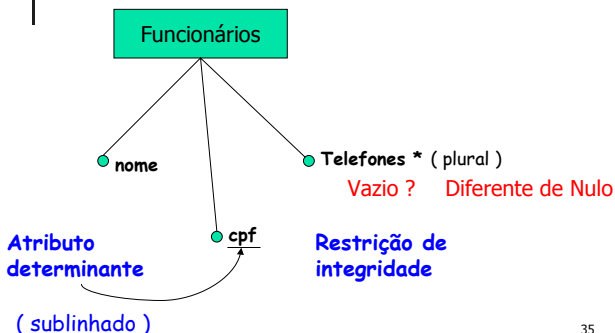


Valor multivalorado não é um vetor e sim um subconjunto do conjunto de valores

≅ Set do pascal

Poderia ser definido 2 valores monovalorados?

Atributos Multivalorados



Vazio ? Diferente de Nulo

Restrição de integridade

Atributo determinante
(sublinhado)

Chaves

➤ **Chave Primária:** Identifica uma única entidade num conjunto de entidades e um relacionamento num conjunto de relacionamentos.

➤ **Chave Estrangeira:** Atributo da entidade que representa a chave primária da entidade da outra entidade da relação

CPF ?

Conotação operacional!

➤ **Chave Candidata:** Atributo da entidade passível de se tornar uma chave primária.

Regras de Integridade Relacional

- Integridade de Entidade
 - Nenhum atributo da chave primária pode ter valor nulo.
- Integridade Referencial
 - Se uma relação inclui uma chave Externa K equivalente a uma chave primária de outra relação R, então K deve sempre corresponder a um valor da chave primária em alguma tupla de R ou ser totalmente nulo.

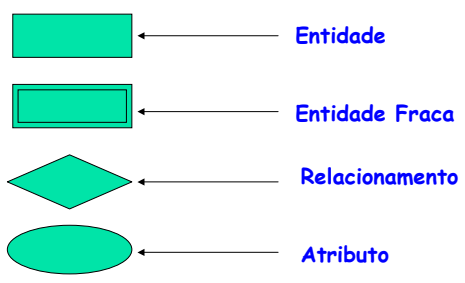
Exemplo de Banco de Dados

Aluno	Nome	Código	Curso
	Manuela	A32	INF
	Paula	A31	MAT

Materia	Nome	Código	Dep
	Banco de Dados	BD1	INF
	SO	SO1	INF
	Cálculo2	CAL2	MAT
	Administração	ADM1	ADM

Boletim	Aluno	Materia	Nota
	A32	BD1	8
	A32	SO1	7,5
	A32	ADM1	6
	A32	CAL2	9
	A31	BD1	6
	A31	SO1	5,5
	A31	ADM1	4

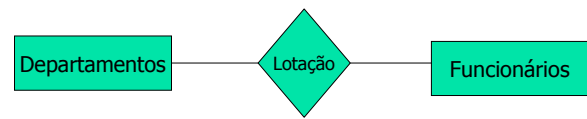
Notação



Relacionamento

José da Silva está lotado no departamento de Vendas.
Atributo?
 Vendas é um elemento do conjunto de entidades Departamento. Não pode ser atributo de Funcionários.

Estrutura abstrata **Relacionamento**



Relacionamento

Relacionamento é uma associação entre as diversas entidades.

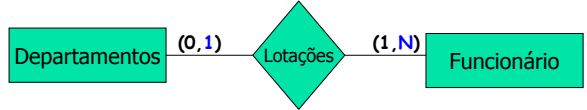
Um relacionamento é um par ordenado (e1, e2) onde e1 e e2 são elementos do conjunto de entidades E1 e E2 respectivamente.

Conjunto dos relacionamentos:

É o conjunto dos pares ordenados que caracterizam um relacionamento.

Relacionamento 1 para N

Classe de um relacionamento ?



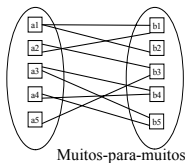
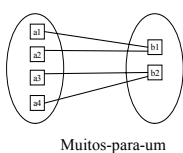
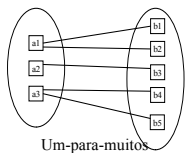
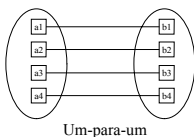
Classe

- > Um departamento pode ter vários funcionários.
- > Um funcionário só pode estar em um departamento.

Estrutura x Semântica

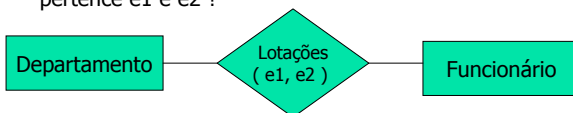
Cardinalidades

- Um-para-um
- Um-para-muitos
- Muitos-para-um
- Muitos-para-muitos



Relacionamento 1 para N

Se (e_1, e_2) é um par ordenado de Lotações, a quem pertence e_1 e e_2 ?



➤ Por convenção a ordem dos termos do par ordenado indica a sua pertinência. Cima para baixo e esquerda para direita.

Relacionamento 1 para 1

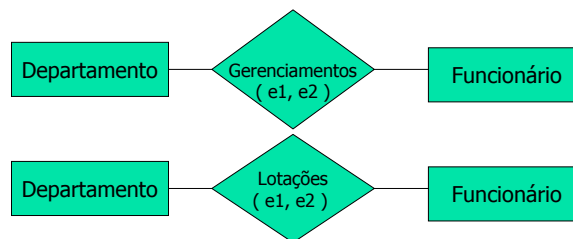


Classe

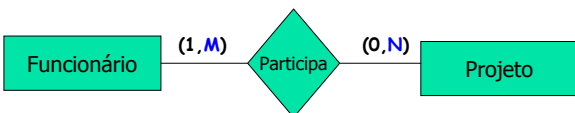
- Um departamento só possui uma gerência
- Um funcionário só pode gerenciar um departamento

Restrição de integridade

$E(e_1, e_2)$ em Gerenciamentos $\Rightarrow E(e_1, e_2)$ em Lotações



Relacionamento M para N



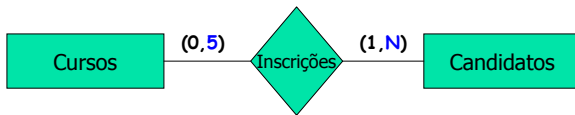
Classe

- Um projeto pode ter vários funcionários participando
- Um funcionário pode estar participando de vários projetos

Exemplos

- Locadora: Filmes x clientes
- Natação: Nadadores x modalidades
- Universidade: Professor X Alunos
- Automobilismo: Carros x Fabricantes
- Carros x Placa
- Motoristas x carros

Relacionamento x para N

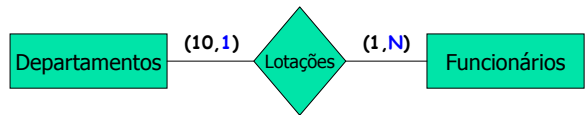


Classe

- > Um Curso pode ter vários candidatos inscritos
- > Um candidato pode se inscrever em no máximo 5 cursos

Relacionamento x para N

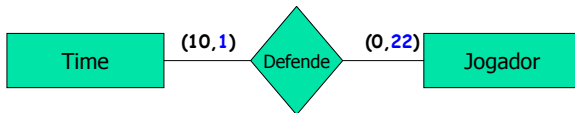
Totalidade mínima



Integridade

- > Um Departamento só existe com pelo menos 10 funcionários

Relacionamento x para N



Classe

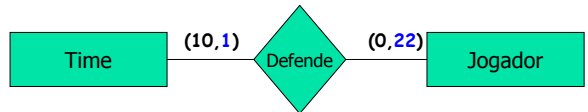
- > Um jogador só pode estar defendendo um time
- > Um time pode possuir no máximo 22 jogadores vestindo sua camisa

Restrição de integridade

Relacionamento 1 : 1 (apenas 1)

Relacionamento 1 : N ou N : 1 (apenas 1)

Relacionamento X : N ou N : X (no máximo X ou no mínimo X)



Restrição de integridade

Relacionamentos Parciais ou Totais

Total: Dado um conjunto de entidades E e um relacionamento R em que E participa, todo elemento de E tem de estar em R.

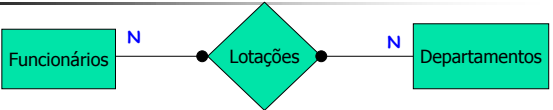
Parcial: Dado um conjunto de entidades E e um relacionamento R em que E participa, todo elemento de E pode não estar em R.

Toda entidade que não apresenta um relacionamento total é dita: "ENTIDADE FRACA"

Restrição de integridade

Relacionamento	restrição	Em relação a
Lotação	Total	Funcionários
Lotação	Parcial	Departamentos
Gerenciamento	Total	Departamento
Participação	Parcial	Projetos
Participação	Total	Funcionários

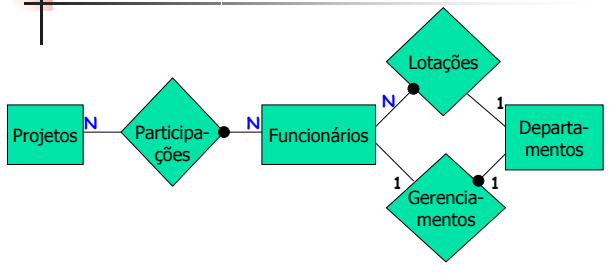
Outra abstração



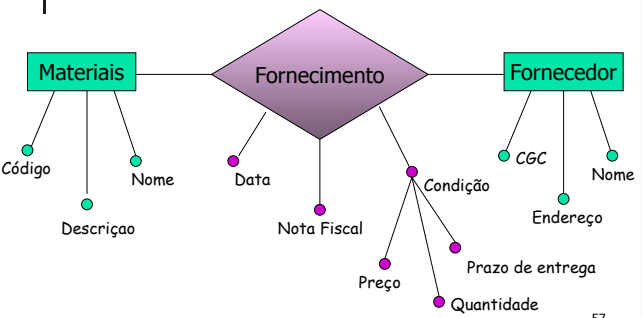
Neste caso considera-se que não pode existir Departamentos sem funcionários. E na criação de um novo Departamento quem vem primeiro?

- No nível conceitual não teremos problemas. Assumimos que as inclusões serão simultâneas
- No nível operacional teremos duas transações diferentes que serão executadas em seqüência.

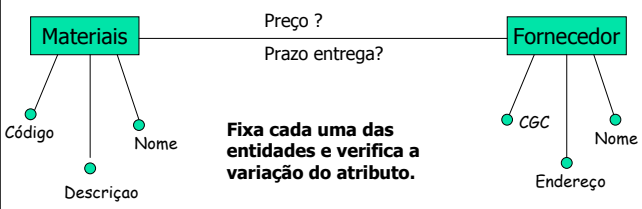
Projeto



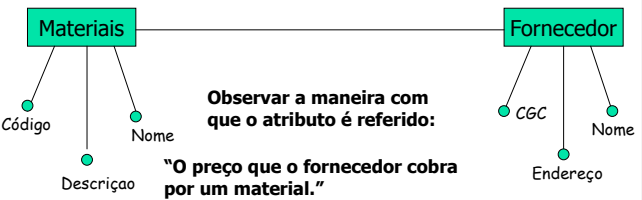
Atributos de um relacionamento



Determinação de atributo de relacionamento



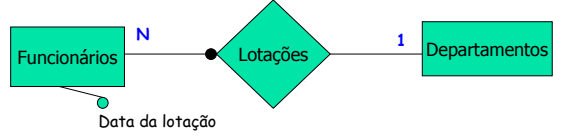
Determinação de atributo de relacionamento



Determinação de atributo de relacionamento

Caso N : 1

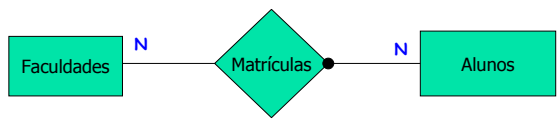
Como fixar Funcionários?



Sendo uma relação 1 : N, o atributo do relacionamento pode ser movido para o conjunto de entidades do lado N.

- > O atributo Data da locação não é próprio do funcionário.
- > E no ponto de vista da empresa?

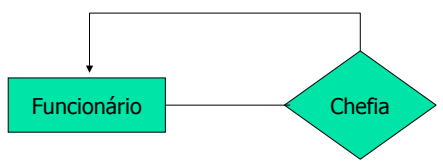
Repetição de pares em um conjunto de relacionamento



Caso um aluno faça vestibular para a mesma faculdade em dois anos diferentes, para cursos diferentes.

Neste caso, um atributo do relacionamento vai distinguir os pares, e que o conjunto agora não é só dos pares, mas de elementos que contêm os atributos dos relacionamentos. (ano da matrícula)

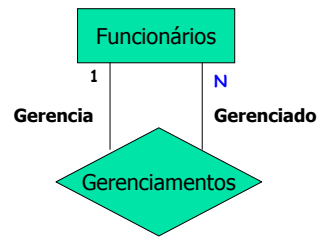
Auto-Relacionamento



Relação da entidade E com a Própria entidade E

Matricula	Nome	Mat-Chefe
3377.2	Luiz souza	5689.0
4987.1	Barbara	5689.0
5689.0	Paula Souza	-
7493.7	Julia Martins	5689.0

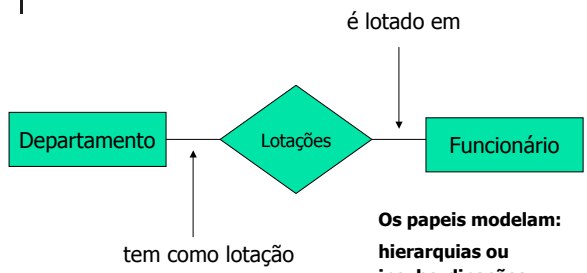
Auto-Relacionamento - Papel



(José, Carlos)
 Quem é quem?

Depende do papel no relacionamento

Relacionamento 1 para N - papel

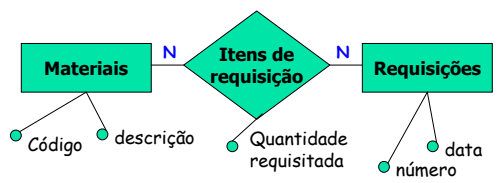


Os papéis modelam: hierarquias ou insubordinações

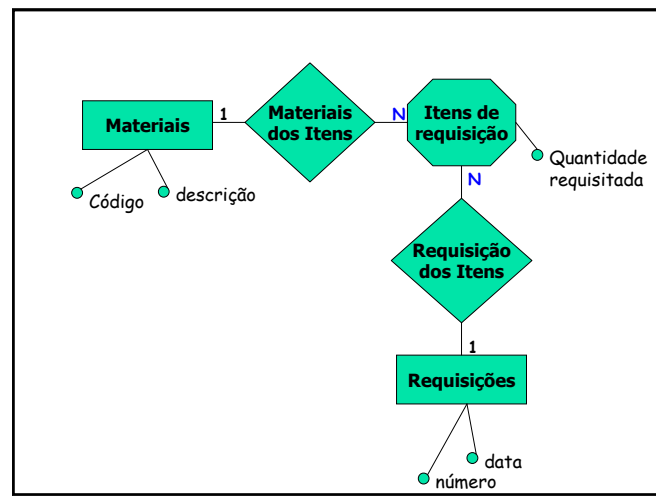
Decomposição de relacionamento N:N

A título de reduzir a complexidade de uma relação, devemos decompor os relacionamentos N:N da seguinte forma:

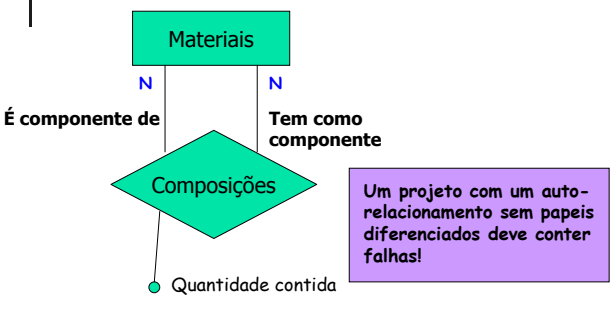
Seja a relação de requisição,



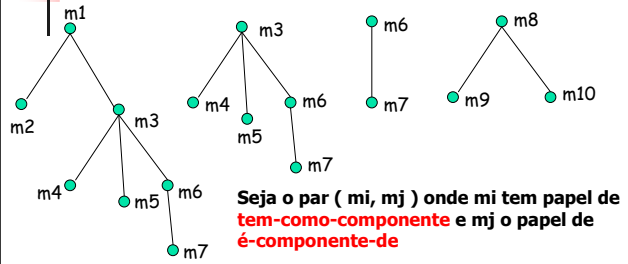
Vamos decompor em duas relações N:1



Auto-Relacionamento - Papel



Árvore de composição



Seja o par (m_i, m_j) onde m_i tem papel de **tem-como-componente** e m_j o papel de **é-componente-de**

Os elementos de composição seriam: $[(m_i, m_j), q]$
 $[(m1, m2), 4], [(m1, m3), 3], [(m3, m5), 6], [(m6, m7), 2], \dots$

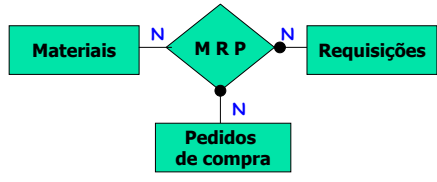
Relacionamentos Triplos

Uma empresa constrói equipamentos a partir de seus desenhos de projetos. Estes desenhos geram requisições de materiais que irão dar origem a pedidos de compra.

Queremos modelar esta estrutura mostrando quais materiais de que requisições geram quais pedidos de compra.

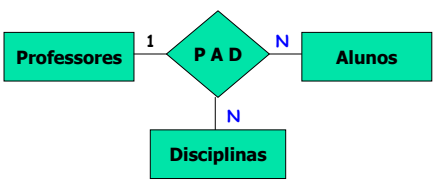
Essa tripla informação gera um relacionamento triplo.

Relacionamentos Triplos



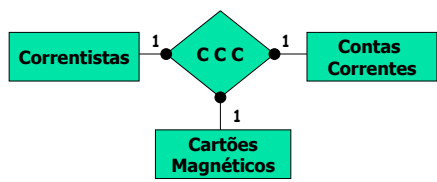
Cada par Requisição-Pedido associa-se com um número de materiais
 Um material de uma requisição pode dar origem a vários pedidos.
 Um material de um pedido pode originar-se de várias requisições.

Relacionamentos Triplos



Dado um aluno em uma disciplina associa-se apenas um professor.
 Um professor pode ministrar uma disciplina para vários alunos.
 Um professor pode dar aulas a um aluno de várias disciplinas.

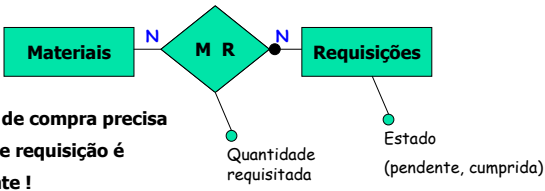
Relacionamentos Triplos



Sempre existe um cartão para cada conta de cada correntista.
Que outras relações estão embutidas dentro desta relação CCC?
 Correntistas x Contas correntes ?
 Contas correntes x Cartões magnéticos ?

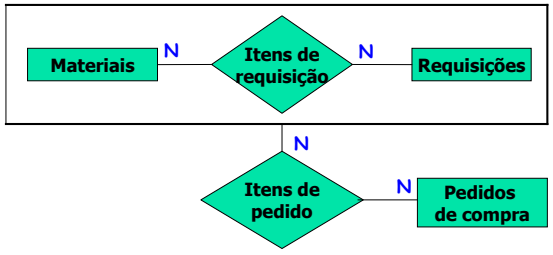
Agregação

Para o caso citado anteriormente de pedido de compra, vamos considerar que uma requisição pode ser satisfeita por materiais que estejam em estoque e não será necessário a emissão de um pedido de compra.



Agregação

O conjunto de relacionamentos M-R e seu conjuntos de entidades serão **AGRAGADOS** como uma Entidade e esta se relacionará com Pedidos de compra.



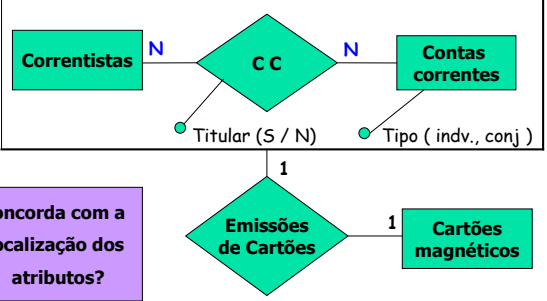
Agregação

Voltando ao caso das contas bancárias, consideremos que um cartão só é emitido com a solicitação do cliente e esta solicitação só será aceita caso este não tenha restrições.

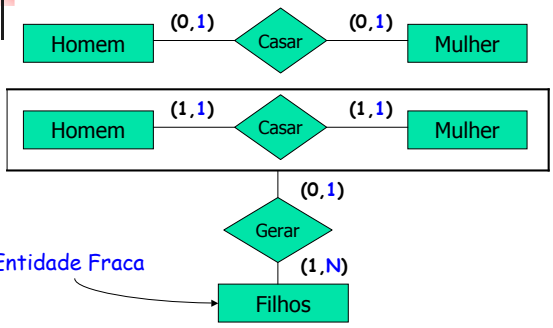
Um outro aspecto importante é que, da forma que foi concebido o relacionamento triplo, quando se quiser obter informações referentes aos correntistas e a conta bancária terá de passar pelos cartões magnéticos.

Devido a isso, a melhor forma de montar o relacionamento é criando uma relação entre Correntistas e Contas correntes independente de quaisquer outros conjuntos.

Agregação



Agregação

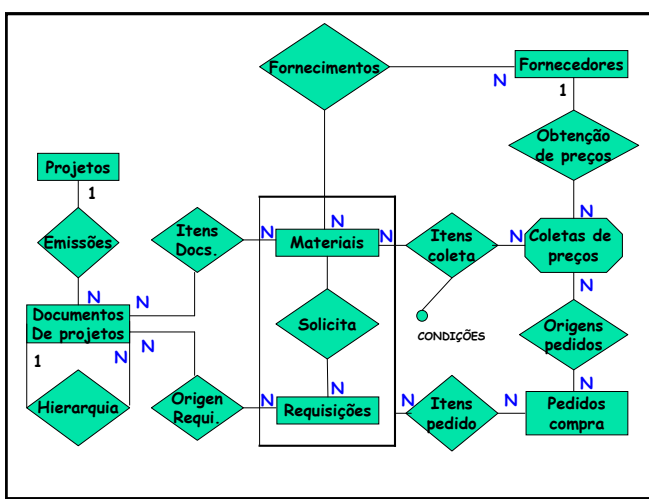


Entidades Fortes / Fracas

- Entidades FORTES são Entidades de Dados que possuem alto grau de independência com relação à existência e identificação.
- Entidades FRACAS são Entidades que possuem dependência de existência e ou de identificação.

Exemplo: Em uma empresa existem os seguintes itens descrevem a funcionalidade de solicitação de materiais:

- > Projetos emitem documentos que vão definir composição de itens necessários para efetivação do produto de um projeto.
- > Os documentos devem seguir uma prioridade de atendimento conforme a importância do mesmo.
- > Os documentos devem conter matérias que existem no cadastros de materiais.
- > Serão geradas requisições de materiais a partir da especificação dos documentos do projeto.
- > Não existir requisições de materiais que não vão estar com saldo suficiente em estoque para atender.
- > Existe um cadastro dos fornecedores da empresa para os materiais que compõem o seu estoque.
- > Quando necessário, deve ser emitido um pedido de compra para um fornecedor atendendo a demanda dos projetos.
- > O pedido de compra deve usar um critério de levantamento de preço para escolha do fornecedor para o qual será emitido o pedido.



UEFS - Universidade Estadual de Feira de Santana
http://www.uefs.br

Agregação de Auto-Relacionamento

Voltemos ao caso das composições:

Considere que existem peças similares de outro fabricante que sempre podem substituir a sua similar em caso de falta.

Reflexividade ~~A - A~~
 Simetria $A - B \Leftrightarrow B - A$
 Transitividade $A - B \text{ e } B - C \Leftrightarrow A - C$

UEFS - Universidade Estadual de Feira de Santana
http://www.uefs.br

Agregação de Auto-Relacionamento

Uma terceira situação é que não havendo nem a peça original nem a similar, podemos utilizar uma peça alternativa a depender da composição:

Seja: (p0, p1) uma relação original
 (p0, p2) uma relação alternativa

Surge a necessidade de relacionar (p0, p1) com (p0, p2) e relação de relacionamentos não existe portanto teremos que agregar.

UEFS - Universidade Estadual de Feira de Santana
http://www.uefs.br

Agregação de Auto-Relacionamento

Integridade: (p0, p2) deverá estar em composições e Composições alternativas por conta de restrição de integridade.

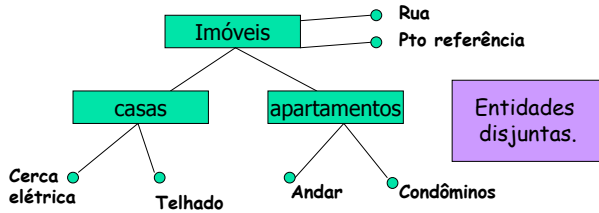
UEFS - Universidade Estadual de Feira de Santana
http://www.uefs.br

Agregação de Auto-Relacionamento

Precisamos relacionar p2 ao par (p0, p1) para eliminar a redundância e unir a solução a similaridade:

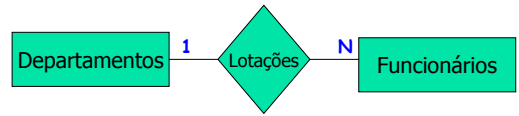
Especialização

Ocorre quando um objeto (entidade) é subconjunto de um outro conjunto de objeto, mas possui características próprias.

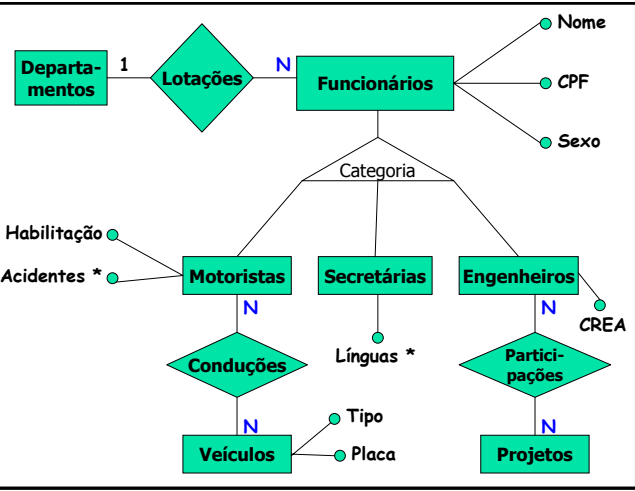


Especialização

Voltemos ao caso funcionários-Departamentos.



Dentre os funcionários, existem características que diferem entre as entidades. Essas diferenças vão se caracterizar através dos seus atributos.

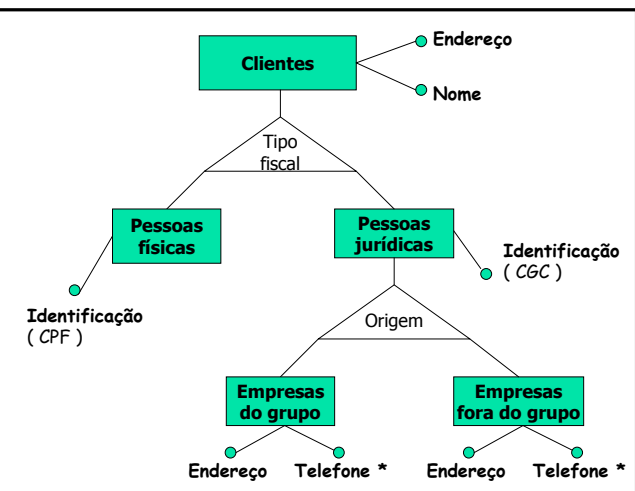


Generalização

Ocorre quando um objeto (entidade) é um agrupamento de outros objetos que não apresentam diferenças nos seus atributos mas semanticamente são diferentes.

Imaginemos a modelagem de Clientes de uma loja que podem ser pessoas físicas e jurídicas.

Em relação as pessoas jurídicas diferenciamos as empresas do grupo e as empresas fora do grupo.



Outros modelos

A escolha de um modelo que melhor se ajuste à realidade que pretende expressar é fator crítico para o sucesso ou fracasso do projeto.

“Modelo de dados é uma coleção de ferramentas conceituais para descrição dos dados, relacionamento entre os dados, semântica e restrições dos dados”

Korth e Silberschatz

Outros modelos

Modelar os dados é uma maneira de expressar uma realidade através de um formalismo que requer abstração por parte do modelador.

Existem diversas técnicas para modelagem de dados, cada uma com ferramentas de abstração, determinando a classe de problemas mais adequadas ao seu uso.

91

Visão Macro

Um modelo completo é composto por sub-modelos que expressam visões diferentes da mesma realidade. Essas visões estão divididas em três:

Visão de objetos - descreve os objetos que compõem o sistema e seus relacionamentos através de diagramas de objetos.

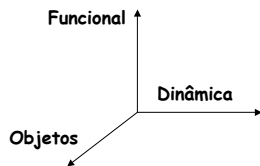
Visão funcional - descreve as transformações dos valores das instâncias dos objetos.

Visão dinâmica - descreve os aspectos do sistema que modificam com o passar do tempo, especificando o controle do sistema.

92

Outros modelos

Cada visão descreve um aspecto do sistema, mas contém referências às outras visões.



Esses inter-relacionamentos entre visões é inevitável, compondo um fator delicado na modelagem.

93

Outros modelos

As metodologias estruturadas sempre abordam as três perspectivas. Por ter como princípio a decomposição funcional para modelar sistemas, essas metodologias dão mais ênfase à visão funcional; em segundo grau de importância, vem a visão dinâmica e por fim a visão de objetos.

A metodologia orientada a objetos abordam as três perspectivas com ênfases diferentes. A visão de objetos é a mais enfatizada, depois a dinâmica e a funcional.

94

Outros modelos

"A abstração deve estar associada a um propósito. Desta forma, pode-se ter várias abstrações de um mesmo problema para propósitos diferentes. A construção de modelos pela abstração possui o caráter de simplificação da realidade a ser modelada, por isso não deve procurar a verdade absoluta, mas sim a adequação ao propósito."

Rumbaugh

95

Técnica de
Modelagem
de Objetos
TMO

96

TMO - Técnica de Modelagem de Objetos

Objeto - algo com limites nítidos e significado em relação à realidade estudada. Facilitam a compreensão do mundo real e oferecem uma base real para a implementação em um sistema de software.

Exemplos de instâncias de objetos:

O rio Amazonas, a cidade de Salvador, a empresa Microsoft, etc.

97

TMO - Técnica de Modelagem de Objetos

Os objetos possuem características próprias que descrevem o seu estado em um determinado momento, e a isso denomina-se **atributos** ou **propriedades** de um objeto.

Exemplo: **A cidade de Salvador** possui uma população de 500.000 habitantes. Neste caso, população é um atributo que descreve o objeto **Salvador** em um dado momento.

98

TMO - Técnica de Modelagem de Objetos

Os objetos são responsáveis por atuar sobre atributos e também sobre outros objetos. Essas operações que descrevem o comportamento de um objeto são chamadas de **métodos**.

Exemplo: **A cidade de Salvador** aumentou a população de 50.000 novos habitantes. O método **"aumentar"** faz parte deste objeto.

99

TMO - Técnica de Modelagem de Objetos

Um grupo de objetos com propriedades semelhantes, os mesmos métodos e, conseqüentemente, a mesma semântica são ditos pertencentes a uma **classe**.

Exemplo: "Estados brasileiros" e "Estados"

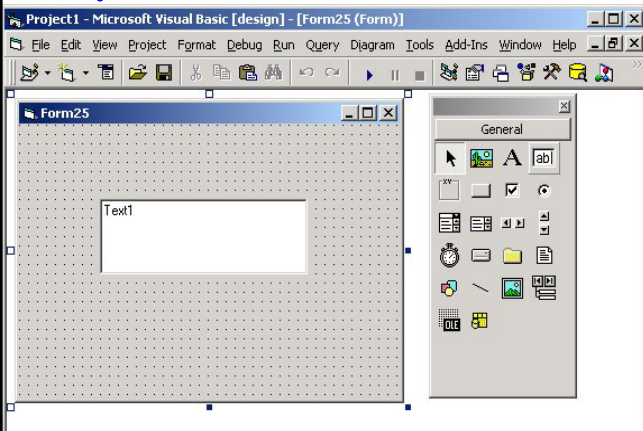
Os objetos de uma classe compartilham um objeto semântico comum, além dos requisitos de atributos e comportamentos:

Celeiro e Cavalo possuem um preço e uma idade, porém o celeiro é pintado e o cavalo alimentado.

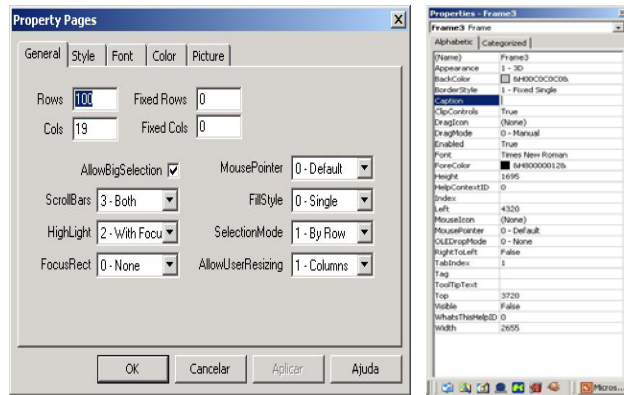
A interpretação da semântica depende do propósito de cada aplicação.

100

Objetos



Propriedades

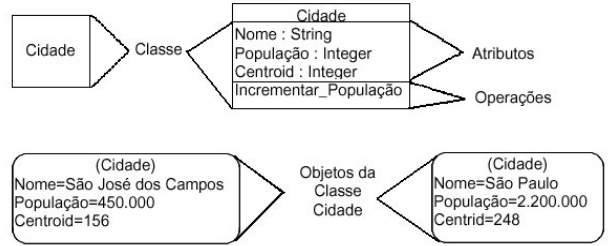


Classes



TMO - Técnica de Modelagem de Objetos

Cada metodologia de modelagem utiliza uma notação gráfica própria. Esta é a representação para o diagrama de objetos:



TMO - Técnica de Modelagem de Objetos

Associação: É o conjunto de ligações com estrutura e semântica comuns.

Uma cidade pertence a um estado

Ligação: É a conexão física ou conceitual entre instâncias de objetos. É uma instância de uma associação.

A cidade de Salvador pertence ao estado da Bahia.

105

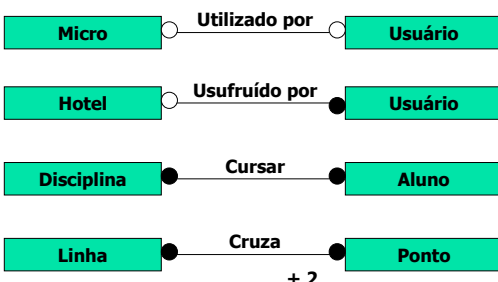
TMO - Técnica de Modelagem de Objetos

Multiplicidade: É especificação de quantas instâncias de uma classe relacionam-se a uma única instância de uma classe associada.

- Bola cheia - significa zero ou muitos
- Bola vazia - significa zero ou um

106

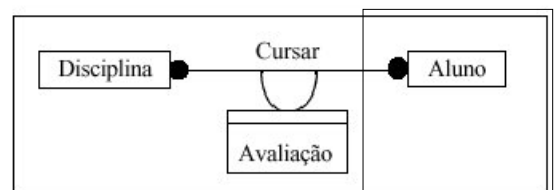
TMO - Técnica de Modelagem de Objetos



107

TMO - Técnica de Modelagem de Objetos

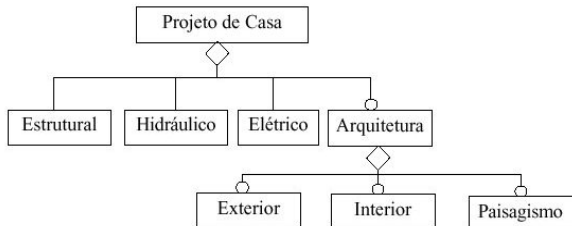
Atributo de Ligação: Assim como um atributo pode ser uma propriedade de um objeto de uma classe, um atributo de ligação é uma propriedade de uma associação.



108

TMO - Técnica de Modelagem de Objetos

Agregação: É um tipo de associação forte onde um objeto agregado é constituído de componentes. É representado por um relacionamento "parte-todo" ou "uma parte de".



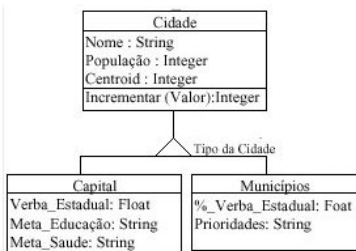
TMO - Técnica de Modelagem de Objetos

Generalização e Especialização: Ambos referem-se a aspectos da mesma idéia e são usados com frequência de forma intercambiável. São dois diferentes pontos de vista de um mesmo relacionamento, visto a partir da **superclasse** ou das **subclasses**.

Generalização deriva do fato de que a superclasse generaliza as subclasses.

Especialização refere-se ao fato de que as subclasses especializam a superclasse

TMO - Técnica de Modelagem de Objetos



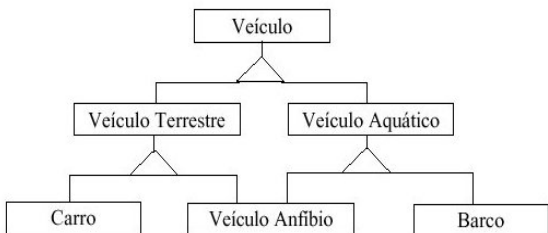
TMO - Técnica de Modelagem de Objetos

Herança: A herança permite que uma classe herde as características de seu ancestral.

Herança Múltipla: A herança múltipla permite que uma classe possua mais de uma superclasse e herde as características de todos os seus ancestrais.

- Possibilita a mesclagem de informações de duas ou mais classes.
- Perda de simplicidade conceitual e de implementação.

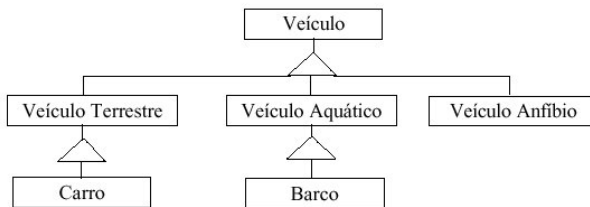
TMO - Técnica de Modelagem de Objetos



Uma característica proveniente da mesma classe ancestral encontrada em mais de um caminho é herdada apenas uma vez.

TMO - Técnica de Modelagem de Objetos

Uma solução para a herança múltipla apresentada é elevar o veículo anfíbio para o mesmo nível de veículo terrestre e veículo aquático, tornando-se uma outra especialização de veículo.



TMO - Técnica de Modelagem de Objetos

Polimorfismo: É a possibilidade de uma mesma operação atuar de forma diferentes em classes diferentes.

"Mover" janela e "Mover" peça de xadrez

TMO - Mapeamentos

Mapeamento de classes: Cada classe dá origem a uma tabela, onde cada atributo da classe corresponde à uma coluna, e os valores dos atributos para cada objeto na classe correspondem a uma linha.



Deve-se tomar cuidado em definir uma chave primária.

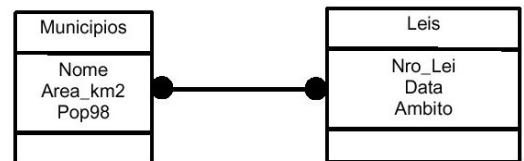
TMO - Mapeamentos

TABELA MUNICIPIO

Id- Municipio	Nome	População98
123	São José dos Campos	450.000
100	São Paulo	12.000.000
131	São José do Rio Preto	480.000
171	Guarujá	204.000
...

TMO - Mapeamentos

Mapeamento de associações N-M: Cada associação "muitos para muitos" deve ser mapeada em uma nova tabela. As chaves das tabelas das classes associadas transformam-se em atributos dessa nova tabela.



TMO - Mapeamentos

TABELA MUNICIPIOS

Id	Nome	Area_km2	Pop98
121	São Carlos-SP	1.800	320.000
233	Londrina-PR	2.360	120.000
321	Santa Maria-RS	1.640	480.000
600	Guiaba-MS	2.660	430.000
...

TABELA LEIS

Nro_Lei	Data	Ambito
122.444.78	10/09/97	Estadual
145.883.33	03/02/98	Federal
111.211.44	08/12/97	Federal
223.444.76	10/04/98	Estadual
...

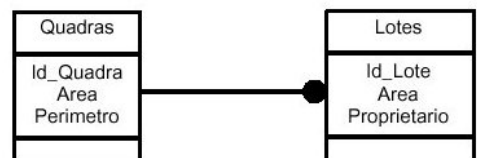
TABELA MUNICIPIOS_LEIS

Id	Nro_Lei
121	122.444.78
121	145.883.33
233	223.444.76
233	145.883.33
233	111.211.44
...	...

TMO - Mapeamentos

Mapeamento de associações 1-N: Existem duas implementações:

Caso 1: Transpõe-se a chave primária da tabela correspondente à classe 1 para a tabela correspondente à classe muitos.



TMO - Mapeamentos

TABELA QUADRAS

Id_Quadra	Area_m2	Perimetro
A-10	4.580	383
B-08	5.200	652
C-22	4.220	322
...

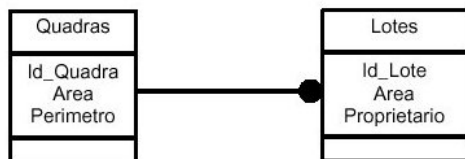
TABELA LOTES

Id_Lote	Area	Proprietário	Id_Quadra
10-01	1.200	Antonio C. M.	A-10
10-05	1.800	Paulo M.	A-10
08-03	900	Luiza E.	B-08
08-07	800	Luis I. L. S.	B-08
22-01	2.200	Fernando H. C.	C_22
...

TMO - Mapeamentos

Mapeamento de associações 1-N:

Caso 2: Cria-se uma tabela correspondente a associação, transpondo para ela as chaves das duas classes.



TMO - Mapeamentos

TABELA QUADRAS

Id_Quadra	Area_m2	Perimetro
A-10	4.580	383
B-08	5.200	652
C-22	4.220	322
...

TABELA LOTES

Id_Lote	Area	Proprietário
10-01	1.200	Antonio C. M.
10-05	1.800	Paulo M.
08-03	900	Luiza E.
08-07	800	Luis I. L. S.
22-01	2.200	Fernando H. C.
...

TABELA QUADRAS_LOTES

Id_Quadra	Id_Lote
A-10	10-01
A-10	10-05
B-08	08-03
B-08	08-07
C_22	22-01
...	...

Caso 1 x Caso 2

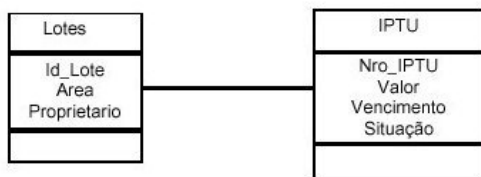
Mapeamento de associações 1-N:

O primeiro caso apresenta uma tabela a menos o que significa menos espaço ocupado, e maior rapidez no processamento de operações que envolvam a associação.

O segundo caso apresenta uma maior independência dos dados. Lotes apresenta apenas seus dados próprios. Em caso de mudança estrutural no tipo de da associação, passando de "Muitos para um" para "muitos para muitos", não requer nenhuma alteração estrutural nas tabelas.

TMO - Mapeamentos

Mapeamento de associações 1-1: Deve-se transpor a chave de uma tabela para a outra levando em consideração o bom senso para definir o sentido da transposição.



TMO - Mapeamentos

TABELA LOTES

Id_Lote	Area	Proprietário
10-01	1.200	Antonio C. M.
10-05	1.800	Paulo M.
08-03	900	Luiza E.
08-07	800	Luis I. L. S.
22-01	2.200	Fernando H. C.
22-02	3.000	Prefeitura
25-01	1.450	Prefeitura
...

TABELA IPTU

Nro_IPTU	Valor	Vencimento	Situacao	Id_Lote
0345/96	1.200,00	12/03/98	NOT_OK	10-01
0337/97	1.800,00	12/03/98	NOT_OK	10-05
0224/95	900,00	12/03/98	OK	08-03
0112/97	800,00	12/03/98	OK	08-07
0332/92	2.200,00	12/03/98	OK	22-01
...

TMO - Mapeamentos

TABELA LOTES

Id_Lote	Area	Proprietário	Nro_IPTU
10-01	1.200	Antonio C. M.	0345/96
10-05	1.800	Paulo M.	0337/97
08-03	900	Luiza E.	0224/95
08-07	800	Luis I. L. S.	0112/97
22-01	2.200	Fernando H. C.	0332/92
22-02	3.000	Prefeitura	Null
25-01	1.450	Prefeitura	Null
...

TABELA IPTU

Nro_IPTU	Valor	Vencimento	Situacao
0345/96	1.200,00	12/03/98	NOT_OK
0337/97	1.800,00	12/03/98	NOT_OK
0224/95	900,00	12/03/98	OK
0112/97	800,00	12/03/98	OK
0332/92	2.200,00	12/03/98	OK
...

TMO - Mapeamentos

Qual a melhor implementação?

O primeiro caso é mais interessante pois todas as ocorrências de IPTU possui um lote associado e como consequência não haverá ocorrências de valores vazios.

No segundo caso, lotes da prefeitura por exemplo são isentos de IPTU e conseqüentemente ocorrerá valores vazios.

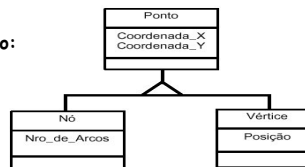
TMO - Mapeamentos

Mapeamento de Generalização e especialização Existem três implementações:

Caso 1 - A classe de generalização e as classes de especialização são mapeadas cada uma para uma tabela. A chave da tabela da generalização é reproduzida nas tabelas das classes de especialização.

TMO - Mapeamentos

Exemplo:



Utilização de um ID compartilhado e a criação de uma coluna Tipo para identificação da tabela de especialização:

TABELA PONTOS

Id_Ponto	Coordenada_X	Coordenada_Y	TIPO
0001	3445	23322	NO
0002	7772	27765	NO
0003	8290	20900	VERTICE
0004	0023	00090	VERTICE
...

TABELA NOS

Id_Ponto	Nro_de_arcos
0001	4
0002	5
...	...

TABELA VERTICES

Id_ponto	Posicao
0003	2
0004	1
...	...

TMO - Mapeamentos

Mapeamento de Generalização e especialização

Caso 2 - Cada classe de especialização é mapeada para uma tabela. A classe de generalização é eliminada e seus atributos são reproduzidos em cada tabela de especialização. Ocorre portanto uma redundância dos dados pertencentes à classe de generalização.

Outros modelos

Modelo semântico

Modelo funcional

Modelo infológico

Modelo de redes

Modelo hierárquico

...

139

Linguagem de manipulação (Álgebra Relacional)

• Álgebra Relacional:

• **Conceito:** Linguagem procedural com a função de manipular uma ou mais relações tendo como resultado um outra relação.

• Grupos:

• Operações de conjunto:
 União, Interseção, Diferença, Produto cartesiano.

• Operações para as relações:
 Seleção, Projeção, Junção, Divisão.

Operações de Conjuntos

• **União:** O resultado T é uma relação que contém todas as tuplas de duas relações sem repetições

Fornecedor(F)			Peça(P)		Fornecimento(G)		
CF	NF	CP	NP	DP	CF	NF	CP
12	ar	19	07	x	12	ar	18
17	ca	18	18	y	17	ca	07
12	ar	07	19	z			
20	bu	18					
17	ca	07					

T:união(F&G)->

CF	NF	CP
12	ar	19
17	ca	18
12	ar	07
20	bu	18
17	ca	07
12	ar	18

• **Interseção:** O resultado T é uma relação que contém apenas as tuplas que aparecem em F e G.

Fornecedor(F)			Peça(P)		Fornecimento(G)		
CF	NF	CP	NP	DP	CF	NF	CP
12	ar	19	07	x	12	ar	18
17	ca	18	18	y	17	ca	07
12	ar	07	19	z			
20	bu	18					
17	ca	07					

T:inter(F&G)->

CF	NF	CP
17	ca	07

Operações de Conjuntos (Álgebra Relacional)

• **Diferença:** O resultado T é uma relação que contém apenas as tuplas que aparecem em F mas não em G.

Fornecedor(F)			Peça(P)		Fornecimento(G)		
CF	NF	CP	NP	DP	CF	NF	CP
12	ar	19	07	x	12	ar	18
17	ca	18	18	y	17	ca	07
12	ar	07	19	z			
20	bu	18					
17	ca	07					

T:dif(F&G)->

CF	NF	CP
12	ar	19
17	ca	18
12	ar	07
20	bu	18

Operações de Conjuntos (Álgebra Relacional)

• **Produto Cartesiano (X):** O resultado T é uma relação que contém todas as tuplas de G concatenadas com todas as tuplas de P.

Fornecedor(F)			Peça(P)		Fornecimento(G)		
CF	NF	CP	NP	DP	CF	NF	CP
12	ar	19	07	x	12	ar	18
17	ca	18	18	y	17	ca	07
12	ar	07	19	z			
20	bu	18					
17	ca	07					

T:Prod(G&P)->

CF	NF	CP	NP	DP
12	ar	18	07	x
12	ar	18	18	y
12	ar	18	19	z
17	ca	07	07	x
17	ca	07	18	y
17	ca	07	19	z

Operações de Relações

(Álgebra Relacional)

- **Selecionar (select σ):** O resultado T é uma relação que contém um conjunto de tuplas que satisfazem uma condição.
 - Sintaxe: σ <condição> (relação)
 - ex.: σ cf=12(Fornecedor);

Fornecedor(F)			Peça(P)		Fornecimento(G)		
CF	NF	CP	NP	DP	CF	NF	CP
12	ar	19	07	x	12	ar	18
17	ca	18	18	y	17	ca	07
12	ar	07	19	z			
20	bu	18					
17	ca	07					

T ->

CF	NF	CP
12	ar	19
12	ar	07

Operações de Relações

(Álgebra Relacional)

- **Projeção (π):** O resultado T é uma relação que contém apenas os domínios indicados (atributos).
 - Sintaxe: π <lista de atributos> (relação)
 - ex.: π cf,cp(Fornecedor);

Fornecedor(F)			Peça(P)		Fornecimento(G)		
CF	NF	CP	NP	DP	CF	NF	CP
12	ar	19	07	x	12	ar	18
17	ca	18	18	y	17	ca	07
12	ar	07	19	z			
20	bu	18					
17	ca	07					

T ->

CF	CP
12	19
17	18
12	07
20	18
17	07

Operações de Relações

(Álgebra Relacional)

- **Junção (Join):** O resultado T é uma relação que contém concatenadas as tuplas G e P que satisfazem a comparação sobre os domínios indicados (CP e NP).

Fornecedor(F)			Peça(P)		Fornecimento(G)		
CF	NF	CP	NP	DP	CF	NF	CP
12	ar	19	07	x	12	ar	18
17	ca	18	18	y	17	ca	07
12	ar	07	19	z			
20	bu	18					
17	ca	07					

T ->

CF	NF	CP	NP	DP
12	ar	18	18	y
17	ca	07	07	x

Operações de Relações

(Álgebra Relacional)

- **Divisão (divide):** Cria uma nova relação através das tuplas de uma relação que são exatamente iguais às da outra relação. $T=R/S$

R	S	T	S	T	S	T
A B C a1 b1 c1 a2 b1 c1 a1 b2 c1 a2 b2 c2 a1 b2 c3 a1 b2 c4 a1 b1 c5	C c1	A B a1 b1 a1 b2	C c1 c2	A B a1 b2 a2 b1	C c1 c2 c3 c4	A B a1 b2
	B C b1 c1	A a1 a2	B C b1 c1	A a1		

SQL

- SQL (Structured Query Language)
- Padrão de direito - ANSI 1986 (American National Standards Institute)
- Padrão de fato - muito utilizada nos SGBDs

SQL

Oferece uma linguagem para acesso a objetos de dados, usando muito menos código de programa do que uma linguagem de programação convencional.

Acesso Programado

X

SQL

Programa

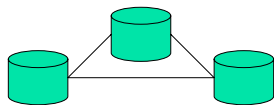
```
Abrir arquivo
Enquanto ( não for fim de arquivo )
{
    Enquanto ( registro em uso )
    {
        tenta travar registro
        verifica timeout
    }
    Se ( timeout )
    {
        avisa ( "problema de acesso" )
        encerra
    }
    Ler registro
    Se valor da coluna = valor
    {
        Altere registro
        destravar registro
    }
}
Fechar arquivo.
```

SQL

```
update nome_da_tabela
set nome_da_coluna = novo_valor
where valor_da_coluna = valor
```

A declaração SQL representa muito mais facilmente um acesso do que uma linguagem de terceira geração. Este nível de abstração tão superior significa maior confiabilidade, facilidade de manutenção e código muito mais legível e compreensível.

SQL Vantagens



- > Não há a necessidade de declarações explícitas de travamento - o servidor gerencia o acesso.
- > Não há referências à localização física dos dados - o servidor traduz um nome lógico para um grupo associado de localizações físicas.
- > Não há especificações de indexação ou estratégias de busca - o servidor identifica o método mais eficiente para a busca dos dados requisitados.

SQL

Seja qual for a linguagem de programação usada podem ser feitas consultas SQL ao servidor.

Ferramentas visuais fazem a interface entre as aplicações e a base de dados.

Muitas vezes o acesso ao banco fica transparente ao usuário do banco.

Para elaborar uma consulta e armazená-la como uma *stored procedure* no SQL Server, é necessário escrever em *Transact-SQL - Linguagem de Consulta Estruturada*.

SQL

Uma declaração SQL quase sempre inclui ao menos um comando:

um verbo indicando uma ação

- > select
- > insert
- > update
- > delete ...

Único caso em que um comando não é necessário, é na execução de uma *stored procedure*

SQL

Comandos em SQL usam palavras reservadas que tem um significado especial para o servidor.

Ex:

from - diz ao servidor que segue uma lista de tabelas.

where - introduz uma lista de condições logicamente conectadas, especificando as linhas afetadas pela declaração.

Quando é submetido um comando SQL ao servidor, este é analisado, otimizado, compilado e então é executado.

Criando tabelas:

- > O comando **CREATE TABLE** cria uma tabela nova, inicialmente vazia, no banco de dados atual. O usuário que executa o comando torna-se o dono da tabela.
- > O nome da tabela deve ser diferente do nome de qualquer outra tabela, seqüência, índice ou visão no mesmo esquema.
- > Uma tabela não pode possuir mais de 1600 colunas (Na prática, o limite efetivo é menor, devido à restrição do comprimento das tuplas).
- > As cláusulas opcionais de restrição especificam as restrições (ou testes) que as linhas novas ou modificadas devem satisfazer para a operação de inclusão ou de alteração ser completada. Uma restrição é uma regra com nome: um objeto SQL que ajuda a definir conjuntos válidos de valores, estabelecendo limites nos resultados das operações de inclusão, exclusão e atualização realizadas na tabela.

Parâmetros:

- > **[LOCAL] TEMPORARY** ou **[LOCAL] TEMP**: Se for especificado, a tabela é criada como uma tabela temporária. Tabelas temporárias são automaticamente eliminadas no final da sessão. Uma tabela permanente com o mesmo nome, caso exista, não será visível na sessão corrente enquanto a tabela temporária existir, a menos que seja referenciada por um nome qualificado pelo esquema. Todo índice criado em tabela temporária também é temporário.
- > **NOME_DA_TABELA**: O nome (opcionalmente qualificado pelo esquema) da tabela a ser criada.
- > **NOME_DA_COLUNA**: O nome da coluna a ser criada na nova tabela.
- > **TIPO_DE_DADO**: O tipo de dado da coluna.

Parâmetros:

- > **DEFAULT**: A cláusula **DEFAULT** atribui um valor padrão para o dado da coluna em cuja definição está presente. O valor pode ser qualquer expressão (sub-consultas e referências cruzadas para outras colunas da mesma tabela não são permitidas). O tipo de dado da expressão padrão deve corresponder ao tipo de dado da coluna. A expressão é utilizada em todas as operações de inclusão que não especificam valor para a coluna. Não havendo valor padrão para a coluna, então **NULL** torna-se o valor padrão.
- > **INHERITS**: Especifica uma lista de tabelas das quais a nova tabela herda, automaticamente, todas as colunas. Se o mesmo nome de coluna existir em mais de uma tabela ascendente um erro é gerado, a menos que o tipo de dado das colunas seja o mesmo em todas as tabelas ascendentes. Se não houver conflito, então as colunas duplicadas são mescladas para formar uma única coluna da nova tabela.

Parâmetros:

- > **CONSTRAINT**: Especifica restrições para uma coluna ou para a tabela. Um nome opcional para a restrição da coluna ou da tabela pode ser definido. Se não for especificado, o nome será gerado pelo sistema.
 - > **NOT NULL**: Valores nulos não são permitidos na coluna.
 - > **NULL**: Valores nulos são permitidos na coluna. Este é o padrão.
 - > **UNIQUE**: A restrição **UNIQUE** especifica a regra onde um grupo de uma ou mais colunas distintas de uma tabela podem conter apenas valores únicos. O comportamento da restrição de unicidade para tabelas é o mesmo da restrição de unicidade para colunas, porém com a capacidade adicional de abranger várias colunas.
 - > **PRIMARY KEY**: A restrição de chave primária que a coluna, ou colunas, da tabela pode conter apenas valores únicos e não nulos. Tecnicamente a chave primária é simplesmente uma combinação de unicidade (**UNIQUE**) com não nulo (**NOT NULL**).

Parâmetros:

- > **CONSTRAINT**:
 - > **CHECK**: A cláusula **CHECK** especifica restrições de integridade, ou testes, que as linhas novas ou atualizadas devem atender para que uma operação de inserção ou de atualização complete. Cada restrição deve ser uma expressão que produza um resultado booleano.
 - > **REFERENCES**: A restrição **REFERENCES** especifica que um grupo de uma ou mais colunas da nova tabela deve conter somente valores correspondentes aos valores das colunas referenciadas *coluna_referenciada* da tabela referenciada *tabela_referenciada*. Se a *coluna_referenciada* for omitida, a chave primária da *tabela_referenciada* é utilizada. As colunas referenciadas devem pertencer a uma restrição de chave primária ou de unicidade da tabela referenciada.

Parâmetros:

- > **CONSTRAINT**:
 - > **REFERENCES**
 - > **MATCH**: Os valores adicionados a estas colunas são comparados com os valores das colunas referenciadas da tabela referenciada utilizando o tipo de comparação.
 - > **ON DELETE**: Especifica a ação a ser executada quando uma linha referenciada da tabela referenciada é excluída.
 - > **ON UPDATE**: Especifica a ação a ser executada quando uma coluna referenciada da tabela referenciada muda de valor.

NO ACTION ; CASCADE ; SET NULL ; SET DEFAULT

Parâmetros:

> CONSTRAINT:

> **DEFERRABLE** ou **NOT DEFERRABLE**: Estas cláusulas controlam se as restrições podem ser postergadas. Uma restrição que não pode ser postergada é verificada imediatamente após cada comando. A verificação das restrições que são postergáveis pode ser adiada para o final da transação

> **INITIALLY IMMEDIATE** ou **INITIALLY DEFERRED**: Se uma restrição é postergável, esta cláusula especifica o momento padrão para verificar a restrição.

SQL

Definição de dados

• Criando tabelas

• Create Table nome_tabela (column_name datatype [Not null | null])

```
create table carro
(
    placa int not null,
    nome varchar(30) not null default "Topic",
    constraint restrição1
        primary key(placa)
);
```

```
create table motorista
```

```
(
    cnh int not null,
    nome varchar(30) not null,
    primary key(cnh)
);
```

```
create table rota
```

```
(
    cod char(6) not null,
    nome varchar(30) not null,
    cnh int not null,
    placa int not null,
    constraint restrição1
        primary key(cod),
    constraint restrição2
        foreign key (cnh) references motorista (cnh),
    constraint restrição3
        foreign key (placa) references carro(placa)
);
```

SQL

Definição de dados

```
create table escola
```

```
(
    cod char(6) not null,
    nome varchar(30) not null,
    bairro varchar(30),
    rota char(6),
    constraint restrição1
        primary key(cod),
    constraint restrição2
        foreign key (rota) references rota (cod),
    constraint restrição3
        Unique (nome)
);
```

SQL

Definição de dados

```
create table aluno
```

```
(
    rg char(12) not null,
    nome varchar(30) not null,
    escola varchar(30) not null,
    constraint restrição3
        primary key(rg),
    constraint restrição4
        foreign key (escola) references escola (cod)
        on delete set null [ou set default]
        on update cascade
);
```

SQL

Definição de dados

Diagnósticos :

CREATE TABLE

Mensagem retornada se a tabela for criada com sucesso.

ERROR

Mensagem retornada se a criação da tabela falhar. Esta mensagem é normalmente acompanhada por algum texto descritivo, como: ERROR: Relation 'nome_da_tabela' already exists, que ocorre quando a tabela especificada existe no banco de dados. create

• **Alterando tabelas**

- Alter **table** nome_tabela **add** column_name datatype null))
- **alter table** aluno **add** idade int null
- **alter table** aluno **drop** idade cascade (restrict)
- **alter table** carro **alter** nome drop default
- **alter table** carro **alter** nome **set default** "Kombi"
- **alter table** aluno **drop constraint** restrição3

• **Consultas** (select/from)

• **Select** lista_atributos **From** tabela

aluno: {cod,nome,bairro,rua,cep,tel}

Nome	Bairro
Paula	Brotas
Marta	Pituba
Joao	P. Flamengo
Suzana	Brotas

- *O nome e bairro da tabela aluno*
- **Select** aluno.nome, aluno.bairro **From** aluno

• *O bairro, cod e nome da tabela aluno*

• **Select** aluno.bairro, aluno.cod, aluno.nome **From** aluno

Bairro	Cod	Nome
Brotas	1	Paula
Pituba	9	Marta
P. Flamengo	3	Joao
Brotas	5	Suzana

• *O nome de todos os bairros dos alunos cadastrados*

• **Select distinct** aluno.bairro **as** bairros **From** aluno

Bairro
Brotas
Pituba
P. Flamengo

• **Consultas** (select/from/where)

Select lista_atributos **From** tabela **Where** condição

• Qualificações no comando **Where**:

- Operações de comparação (=, >, <)
- Faixas (BETWEEN e NOT BETWEEN)
- Correspondência de caracteres (LIKE e Not LIKE)
- Valores desconhecidos (IS NULL e IS NOT NULL)
- Listas (IN e NOT IN)
- Combinações dos acima (AND, OR)

OBS: NOT pode negar qualquer expressão booleana e chaves como LIKE, NULL, BETWEEN e IN

• **Consultas**

(select/from/where)

Operadores de comparação:

- = igual a
- > maior que
- < menor que
- >= maior que ou igual a
- <= menor que ou igual a
- != diferente
- <> diferente
- !> não maior que
- !< não menor que

• **Consultas**

(select/from/where)

Operadores de comparação:

- <> é equivalente a !=
- !< é equivalente a >=
- Ao comparar datas, usa-se < para significar antes e > para depois.
- Usa-se aspas duplas ou simples em torno dos dados do tipo char, varchar e datetime
- Letras minúsculas são maiores que letras maiúsculas, letras maiúsculas são maiores que números.

• **Consultas** (select/from/where) - Faixas

BETWEEN - Chave usada para especificar uma faixa inclusiva; os valores extremos da faixa também são incluídos na busca.

NOT BETWEEN - Chave que exclui os valores extremos especificados na faixa.

• *O cod, nome e bairro da tabela aluno com os cod entre 1 e 6*

Select aluno.cod,aluno.nome, aluno.bairro **From** aluno **where** cod between '1' and '6'

Cod	Nome	Bairro
1	Paula	Brotas
6	Joao	P. Flamengo
5	Suzana	Brotas

• *O cod, nome e bairro da tabela aluno com os cod fora do intervalo entre 1 e 6*

Select aluno.cod,aluno.nome, aluno.bairro **From** aluno **where** cod not between '1' and '6'

Cod	Nome	Bairro
9	Paula	Brotas
7	Joao	P. Flamengo
11	Suzana	Brotas

SQL

• Consultas (select/from/where) - Correspondentes

• A chave LIKE

- Usada para selecionar linhas que contenham campos que correspondem a porções especificadas de uma serie de caracteres (string de caracteres).
- Utilizada apenas com dados do tipo char, varchar e datetime.
- Pode usar curingas.
 - ✓ % qualquer string com nenhum ou mais caracteres
 - ✓ _ um único caracter.
 - ✓ [] um único caracter na faixa especificada.
 - ✓ [^] um único caracter fora da faixa especificada.

Obs.: Inclua os curingas e a string de caracteres entre aspas simples ou duplas.

• Consultas (select/from/where) - A chave LIKE

- Nome e bairro da tabela aluno dos nomes começando com a letra "P"

Select aluno.nome, aluno.bairro **From** aluno
where aluno.nome like "P%"

Nome	Bairro
Paula	Brotas

- Nome e bairro da tabela aluno dos nomes começando com as letras "Ma" e de 3 caracteres.

Select aluno.nome, aluno.bairro **From** aluno
where aluno.cod like "Ma_"

Nome	Bairro
Mai	Itaigara

• Consultas (select/from/where) - A chave LIKE

- O nome e bairro da tabela aluno dos nomes começando com qualquer letra entre P e M, inclusive.

Select aluno.nome, aluno.bairro **From** aluno
where aluno.cod like "[M-P]%"

Nome	Bairro
Paula	Brotas
Marta	Costa Azul
Monica	Pituba

- O nome e bairro da tabela aluno dos nomes começando com qualquer letra entre P e M, inclusive e com o nome com 5 caracteres.

Select aluno.nome, aluno.bairro **From** aluno
where aluno.cod like "[M-P]_ _ _ _ _"

Nome	Bairro
Paula	Brotas
Marta	Costa Azul

• Consultas (select/from/where) - Listas

- A chave IN - Permite ao usuario selecionar valores que correspondam a qualquer um de uma lista de valores.

- O cod, nome e bairro da tabela aluno dos alunos com códigos 1 e 2

Select aluno.cod,aluno.nome, aluno.bairro **From** aluno
where aluno.cod in ('1','2')

Cod	Nome	Bairro
1	Paula	Brotas
2	Joao	P. Flamengo

- Nome e bairro da tabela aluno dos alunos que não possuem os códigos 1 e 2

Select aluno.nome, aluno.bairro **From** aluno
where aluno.cod not in ('1','2')

Cod	Nome	Bairro
4	Marta	Plata
6	Julio	C. Azul

• Consultas (select/from/where) - Condições de conexão

- Condições de conexão com operadores logicos (AND/OR)

✓ AND

- Agrupa duas ou mais condições
- Retorna resultados apenas quando todas as condições são verdadeiras.

- O cod e nome da tabela aluno dos alunos com códigos menor que 5 e nome Paula

Select aluno.cod,aluno.nome, aluno.bairro **From** aluno
where aluno.cod < '5' **and** aluno.nome = "Paula"

Cod	Nome
1	Paula

• Consultas (select/from/where) - Condições de conexão

- Condições de conexão com operadores logicos (AND/OR)

✓ OR

- Agrupa duas ou mais condições
- Retorna resultados quando qualquer das condições são verdadeiras.

- O cod, nome da tabela aluno dos alunos com códigos 1 ou nome Joao

Select aluno.cod,aluno.nome, aluno.bairro **From** aluno
where aluno.cod = '1' **or** aluno.nome = "Joao"

Cod	Nome
1	Paula
2	Joao

• Consultas (select/from/where) - Renomeando Colunas

• O usuário pode estabelecer um outro nome para ser utilizado na saída da declaração select, ao invés do nome da coluna.

• Sintaxe

• **select** <cabecalho_da_coluna as nome_da_coluna>

Select aluno.nome as "Nome do Aluno" **From** aluno

Nome do Aluno
Paula
Fabiana

• String de caracteres em resultados

Select 'nome do aluno', aluno.nome, 'bairro', aluno.bairro
From aluno

	Nome		Bairro
nome do aluno	Marta	bairro	Plata
nome do aluno	Julio	bairro	C. Azul

• Expressões numéricas

• Operadores Aritméticos

+ adição
- subtração
* multiplicação
/ divisão

• Podem ser utilizados em qualquer coluna numérica.
• Usados em qualquer comando que permita expressão.

• Adição

Select cod, nome, qtd+10 **From** produto

• Subtração

Select cod, nome, qtd-10 **From** produto

• Expressões numéricas

• Operadores Aritméticos

• Multiplicação

Nome, sal da tabela funcionário com aumento de 10%

Select nome, sal*1.1 **From** funcionario

• Divisão

Nome, horas da tabela funcionário informando quantos dias de 8 horas de trabalho

Select nome, horas/8 as "Dias de trabalho" **From** funcionario

• A função CONVERT

• Converte expressões de um tipo de dados para outro;
• Usada na *lista_de_seleção* e no comando where;
• Não se pode converter um expressão char que contem caracteres para uma inteira.

• Sintaxe:

convert (*tipo_de_dado*, expressão)

Exemplo:

Select id * 10 **From** produtos ☹ erro !!

Exemplo:

Select convert(int,id) * 10 **From** produtos ☺ certo

• Select / Order By

• O comando **Order By** classifica os resultados da consulta (em ordem ascendente, caso nada seja especificado).

• Itens especificados em uma sentença order by não precisam aparecer na *lista_de_seleção*.

• Ao usar *order by*, nulos são listados primeiros.

Sintaxe:

```
select [distinct] lista_de_selecao from tabela  
[where condicoes_de_busca]  
[order by {coluna | expressao} [asc|desc]]
```

Cod, nome da tabela produtos ordenado por categoria

Select cod, nome **From** produtos **order by** categoria

Cod, nome da tabela produtos ordenado por saldo decrescente

Select cod, nome **From** produtos **order by** (saldo) desc

• Funções de agregação

• Funções de agregação, com exceção do count(*), ignoram valores nulos;
• sum e avg só funcionam com valores numéricos;
• Apenas um linha e retornada (se uma sentença group by não tiver sido usada);
• Não podem ser utilizadas num comando where;
• Podem ser aplicados a todas as linhas em uma tabela ou num grupo de uma tabela.

• Sintaxe simplificada:

```
select nome_da_funcao_agregada ([distinct] expressão)  
from nome_da_tabela  
[where ...condições]
```

• Funções de agregação - count

- Count soma o numero de linhas que pertencem a condição estabelecida.

Exemplos

- Determinar o numero de linhas da tabela

```
Select count (*) From produto
```

18

- Numero de valores não nulos numa coluna

```
Select count (cat ) From produtos
```

16

• Funções de agregação - Max/Min

- Max encontra o maior valor.
Maior preço da tabela produtos.

```
select max(preco) from produtos
```

30,25

- Min encontra o menor valor
Menor preço da tabela produtos

```
select min(preco) from produtos
```

1,00

• Funções de agregação - SUM/AVG

- Sum calcula a soma total dos valores na coluna especificada.

Soma da coluna de horas do departamento "inf"

```
select sum(horas) from empregados where dep = "inf"
```

9939

- Avg calcula a media dos valores na coluna especificada.

Media de preços da categoria 2 da tabela produtos

```
select avg (preço) from produtos where cat = "2"
```

23,40

• Funções de agregação - Distinct

- Opcional com sum, avg e count;
- Não permitida com min, max e count (*);
- Permitida com count (*nome_da_coluna*);
- Usado apenas com nomes de colunas, não com expressões aritméticas

Exemplos:

- Media dos preços dos produtos

```
Select avg (distinct preco) From produto
```

- Qtd de produtos diferentes

```
Select count (distinct tipo) From produtos
```

• select / group by

- Divide os dados em grupos;
- Normalmente utilizada com uma função de agregação na *lista_de_seleção*;
- Todos os valores nulos na coluna group by são tratados como um grupo;

• Sintaxe:

```
select [distinct] lista_de_seleção  
from nome_da_tabela  
group by expressão
```

Ex:

```
Select tipo, avg(preco) From produtos ( sem group by )
```

```
Select tipo, avg(preco) From produtos group by tipo ( com group by )
```

SQL

Manipulação dos dados

• select / group by / having

- A sentença HAVING estabelece condições para a sentença *group by*.
- Sintaxe simplificada:

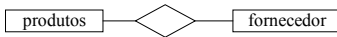
```
select [distinct] lista_de_seleção from nome_da_tabela  
group by expressão having condições
```

Valor da soma dos preços dos produtos quando ultrapassar R\$ 20.000,00

```
Select tipo, sum(preco) From produtos  
Group by tipo  
having sum (preco) > R$20.000
```

• Junções

- Recupera dados de duas ou mais tabelas;
- Combina tabelas através da correspondência de valores de linhas em cada tabela;
- Se o nome da coluna for ambíguo, ou seja, se mais de uma coluna nas tabelas especificadas na declaração de origem (from) possuírem o mesmo nome, o nome da coluna deve ser precedido por um nome de tabela;



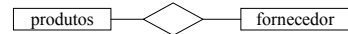
• Junções (continuação)

- Pode-se utilizar tabelas do mesmo banco ou de banco diferentes;
- Colunas comparadas devem ter valores similares;
- Valores nulos não participam da operação de junção;
- As colunas na condição de junção não precisam estar na sentença de seleção.

Campos nome e preço da tabela produtos, campo nome da tabela fornecedores para os fornecedores da Bahia

```

Select p.nome, p.preço, f.nome
From produtos p, fornecedor f
Where p.codf = f.cod and f.end = "Bahia"
    
```



SQL

Manipulação dos dados

• Subconsultas

- Uma subconsulta é uma declaração select, usada como uma expressão, como parte de outra declaração select, update, insert ou delete.
- A subconsulta (select aninhado) é resolvida e depois os resultados são substituídos na consulta mais externa.
- Se a declaração where da consulta mais externa incluir o nome de uma coluna, esta deverá ser compatível para a junção (join), com a coluna nomeada na lista de seleção da subconsulta (ou seja: mesmo domínio).

SQL

Manipulação dos dados

• Subconsultas

- Não utilize a cláusula order by numa subconsulta
- Sempre que utilizar operadores de uma única linha a subconsulta deve ser também de uma única linha.

Exemplo:

```

Select cod From fornecedor
where fornecedor.nome = "bahia"
Select nome From produtos
    
```

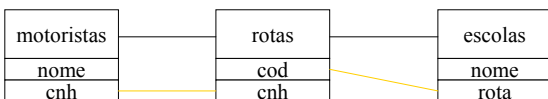
→

```

Select nome From produtos p
where p.cod =
    (select cod
     from fornecedores f
     where f.nome = "bahia")
    
```

• Subconsultas

- O nome do motorista que passa na escola "UFBA"



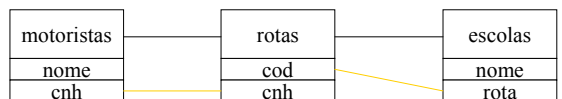
```

select motorista.nome from motorista
where motorista.cnh
in (select rota.cnh from rota, escola
    where escola.nome = "UFBA"
    and rota.cod=escola.rotas)
    
```

Nome
Claudio Manoel

• Subconsultas

- O nome dos motoristas que não possui rota



```

select motorista.nome from motorista
where motorista.cnh
not in (select rota.cnh from rota, escola
    where rota.cod=escola.rotas)
    
```

Nome
Joaquim Francisco

• Subconsultas

- Escolas que possuem a soma de todas as idades menor que o aluno que possui maior idade.

```
select escola, sum(idade) from aluno
group by escola
having sum(idade) < (select max(idade) from aluno)
```

- É possível utilizar subconsultas dentro do FROM.

```
select a.nome
from (select * from aluno) a
```

• Insert

Insert into tabela(campos) **values** (valores)

- Insere um novo motorista com cnh e nome na tabela motorista.

```
Insert into motorsita (cnh,nome) values (9477, "Joana Nascimento")
```

- Insere uma nova escola com cod, nome e bairro de uma tabela temporária.

```
Insert into escola (cod,nome,bairro)
select cod, nome, bairro from tempdb
```

• Delete/Update

delete from tabela **where** condição

- Deleta um aluno.

```
Delete from aluno where cpf="001"
```

- Deleta diversos alunos que estão numa determinada escola.

```
Delete from aluno where aluno.escola in (
select cod from escola where nome like "Teresa%")
```

- Modifica o nome do aluno de código 003.

```
Update aluno set nome = "Barbara Chaves" where cod = "003"
```

- Aumenta 10% dos salario dos motoristas que estão em uma rota.

```
Update motorista set salario = salario * 1.1
where cod in ( select distinct rota.motorista from rota )
```

• Visões

- Cria um visão com dados das tabelas aluno

```
Create view dados_aluno as
Select aluno.nome, aluno.cpf
From aluno
```

- Cria um visão com dados das tabelas aluno, escola, rota.

```
Create view dados_aluno_escola as
Select aluno.nome, aluno.cpf, escola.nome, rota.nome
From aluno, escola, rota
Where rota.cod="pit1" and escola.rota=rota.cod and aluno.escola=escola.cod
```