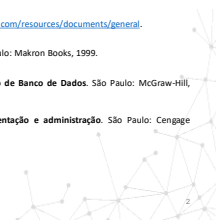




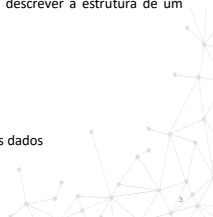
### Referências Bibliográficas

- DATE, C. J. *Introdução a Sistemas de Bancos de Dados*. Rio de Janeiro: Elsevier, 2003.
- ELMASRI, Ramez; NAVATHE, Shamkant B. *Sistemas de Banco de Dados*. São Paulo: Pearson, 2011.
- GARCIA-MOLINA, H; ULLMAN, J. D; WIDOM, J. *Implementação de Sistemas de Banco de Dados*. Rio de Janeiro: Campus, 2001.
- GENNICK, Jonathan; LUERS, Tom. *Aprenda em 21 dias: PL/SQL*. Rio de Janeiro: Campus, 2000.
- IBPHOENIX; Firebird General Documentation. Disponível em: <http://www.ibphoenix.com/resources/documents/general>.
- KORTH, H. F; SILBERSCHATZ, A; SUDARSHAN, S. *Sistema de Banco de Dados*. São Paulo: Makron Books, 1999.
- MANNINO, Michael V. *Projeto, Desenvolvimento de Aplicações & Administração de Banco de Dados*. São Paulo: McGraw-Hill, 2008.
- ROB, Peter; CORONEL, Carlos. *Sistemas de Banco de Dados: projeto, implementação e administração*. São Paulo: Cengage Learning, 2011.
- SUEHRING, Steve. *MySQL: a Bíblia*. Rio de Janeiro: Elsevier, 2002.



### Conceitos básicos sobre SGBD

- SGBD
  - Sistema Gerenciador de Banco de Dados
- Modelo de Dados
  - Conjunto de conceitos que podem ser usados para descrever a estrutura de um Banco de Dados
- Estrutura de um Banco de Dados
  - Tipos de dados
  - Relacionamentos
  - Restrições
  - Operações básicas para recuperação e atualização dos dados



BANCO DE DADOS

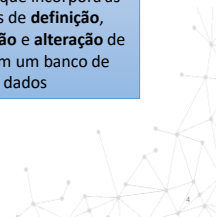


Conjunto de arquivos integrados que atendem a um conjunto de sistemas

Sistema Gerenciador de Banco de Dados (SGBD)



Software que incorpora as funções de definição, recuperação e alteração de dados em um banco de dados



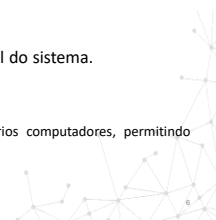
### Redundância de Dados

- Ocorre quando uma determinada informação está representada no sistema em computador várias vezes.
- Formas de redundância de dados
  - Redundância controlada de dados
  - Redundância não controlada de dados.



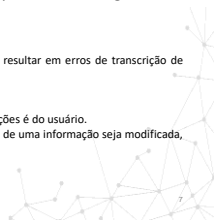
### Redundância Controlada

- Quando o software tem conhecimento da múltipla representação da informação e garante a sincronia entre as diversas representações.
- Para o tudo acontece como se existisse uma única representação da informação.
- Utilizada para melhorar a performance global do sistema.
- Exemplo: Sistema Distribuído
  - Uuma mesma informação é armazenada em vários computadores, permitindo acesso rápido a partir de qualquer um deles.



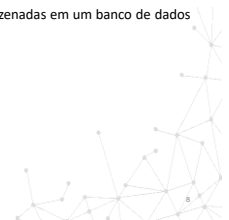
## Redundância Não Controlada

- Quando a responsabilidade pela manutenção da sincronia entre as diversas representações de uma informação está com o usuário e não com o software.
- Este tipo de redundância deve ser evitado, pois traz consigo vários tipos de problemas:
  - Redigitação
    - A mesma informação é digitada várias vezes
    - Além de exigir trabalho desnecessário, a redigitação pode resultar em erros de transcrição de dados.
  - Inconsistências de dados
    - A responsabilidade por manter a sincronia entre as informações é do usuário.
    - Por erro de operação, pode ocorrer que uma representação de uma informação seja modificada, sem que as demais representações o sejam.



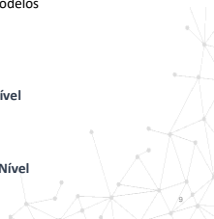
## Modelo de Dados

- Modelo
  - Representação abstrata e simplificada de um sistema real
  - Permite explicar o seu comportamento em seu todo ou em suas partes
- Modelo de Dados
  - Descrição dos tipos de informações que estão armazenadas em um banco de dados
    - Estrutura do Banco de Dados



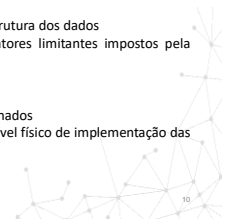
## Modelo de Dados

- Construção de um modelo de dados
  - Linguagem de modelagem de dados.
- Linguagens de modelagem de dados
  - Classificadas de acordo com a forma de apresentar modelos
    - Linguagens textuais
    - Linguagens gráficas.



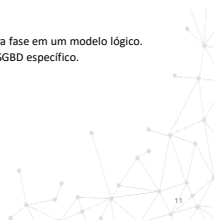
## Categorias de Modelos de Dados

- Modelo Conceitual
  - Descreve os dados conforme a visão do usuário
  - Fornecer uma visão global dos principais dados e relacionamentos
    - Sem se preocupar com as restrições de implementação
- Modelo Lógico
  - Representa a definição do formato adequado da estrutura dos dados
  - De acordo com as regras de implementação e fatores limitantes impostos pela estrutura e tecnologia
- Modelo Físico
  - Descreve os detalhes de como os dados são armazenados
  - A representação dos objetos é feita sob o foco do nível físico de implementação das ocorrências e seus relacionamentos



## Projeto de um Banco de Dados

- Composto de 2 Fases:
  - Modelagem conceitual
    - Construção do modelo conceitual, na forma de um diagrama entidade-relacionamento.
    - Captura as necessidades da organização em termos de armazenamento de dados de forma independente de implementação.
  - Projeto lógico
    - Objetiva transformar o modelo conceitual obtido na primeira fase em um modelo lógico.
    - Define como o banco de dados será implementado em um SGBD específico.



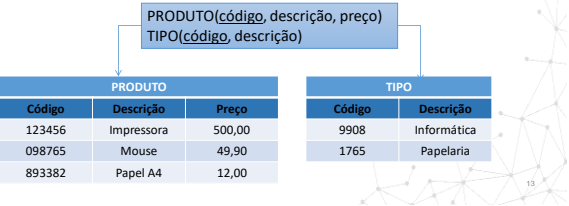
## Modelo Conceitual

- Descrição do BD de forma independente de implementação em um SGBD.
- Registra que dados podem aparecer no banco de dados, mas não registra como estes dados estão armazenados a nível de SGBD.
- Técnica de modelagem conceitual mais usada
  - Abordagem Entidade-Relacionamento (ER).



Modelo Lógico

- Descrição de um Banco de Dados no nível de abstração visto pelo usuário do SGBD.
- Dependente do tipo particular de SGBD que está sendo usado.



Banco de Dados I

Diagrama Entidade-Relacionamento (ER)

Diagrama Entidade-Relacionamento

- Entidades
- Relacionamentos
- Atributo
- Generalização/Especialização

Entidade

- Conceito
  - Conjunto de objetos da realidade modelos sobre os quais deseja-se manter informações no Banco de Dados
- Objetivo
  - Modelar de forma abstrata um Banco de Dados
- Representação
  - Retângulo que contém o nome da entidade.
  - Exemplo:

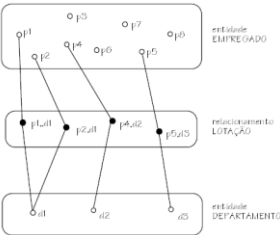


Relacionamento

- Conceito
  - Conjunto de associações entre ocorrências de entidades
- Representação
  - Losango ligado por linhas aos retângulos representativos das entidades que participam do relacionamento
- Exemplo



Relacionamento

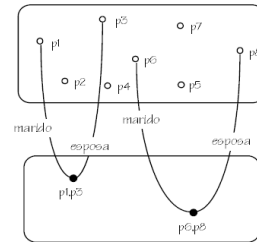


### Auto-Relacionamento

- Relacionamento entre ocorrências de uma mesma entidade
- Papel de Entidade em Relacionamento
  - Função que uma instância de entidade cumpre dentro de uma instância do relacionamento
- Exemplo



### Auto-Relacionamento



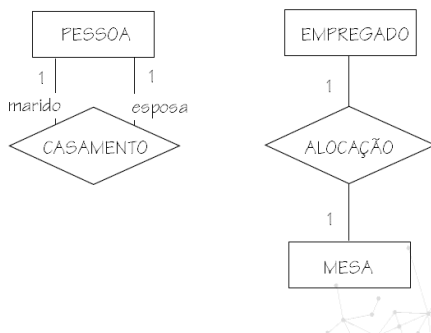
### Cardinalidade

- Número (mínimo, máximo) de ocorrências de entidade associadas a uma ocorrência da entidade em questão através do relacionamento
- Cardinalidades possíveis:
  - (0,1) = nenhuma ou uma ocorrência
  - (1,1) = obrigatoriamente uma ocorrência
  - (0,n) = nenhuma ou muitas ocorrências
  - (1,n) = no mínimo uma ocorrência
- NÃO EXISTE cardinalidade (n,n).

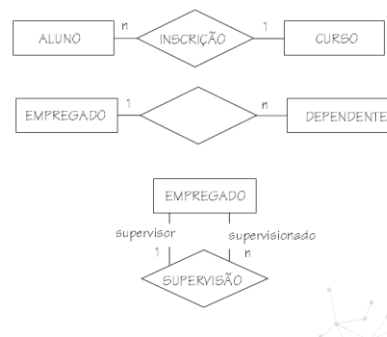
### Relacionamentos

- Indicam o número máximo de ocorrências de associações entre duas entidades.
- São verificados através das cardinalidades máximas.
- Tipos possíveis:
  - (1,1)
  - (0,n)
  - (1,n)
  - (n,n) = uma nova entidade é formada (associativa)

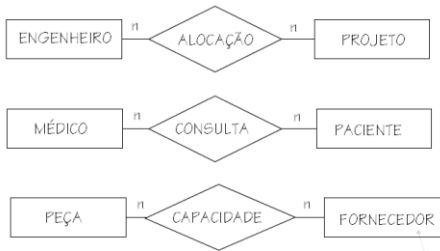
### Relacionamentos 1:1



### Relacionamentos 1:n



### Relacionamentos n:n



### Relacionamentos Ternários

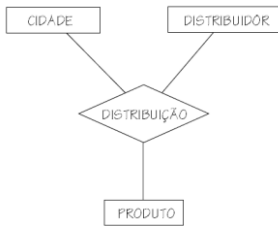
- Relacionamentos de grau maior que 2

- Neste caso, a cardinalidade refere-se a pares de entidades.

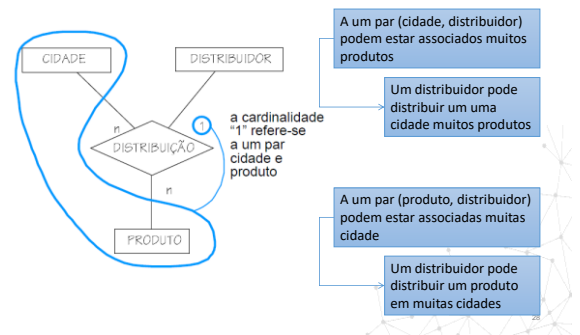
- Um relacionamento R entre 3 entidades A, B e C

- 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.

### Relacionamentos Ternários



### Relacionamentos Ternários



### Cardinalidade Mínima

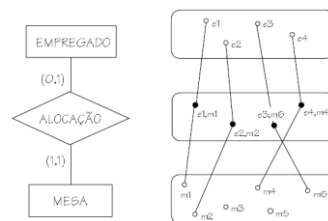
- Número mínimo de ocorrências de entidade que são associadas a uma ocorrência de uma entidade através de um relacionamento

- Cardinalidade mínima 0 → Associação Opcional

- Cardinalidade mínima 1 → Associação Obrigatória

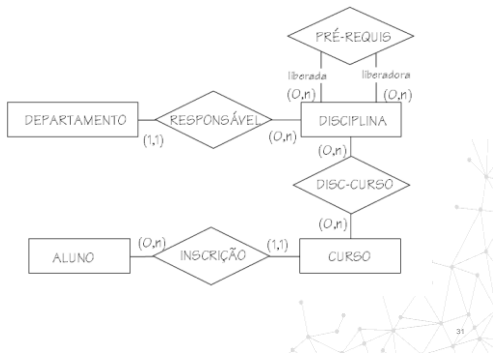
- O relacionamento deve obrigatoriamente associar uma ocorrência de entidade a cada ocorrência da entidade em questão

### Cardinalidade Mínima



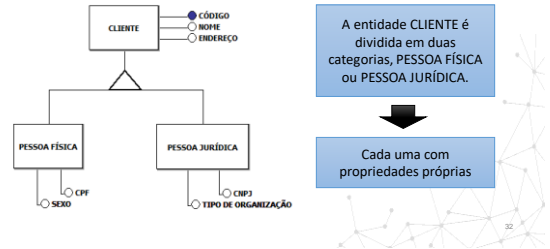
Cada empregado deve ter a ele alocada obrigatoriamente uma mesa (cardinalidade mínima 1) e que uma mesa pode existir sem que a ela esteja alocado um empregado (cardinalidade mínima 0).

### Exemplo



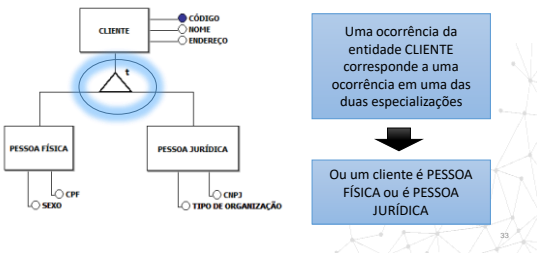
### Generalização/Especialização

- Permite atribuir propriedades particulares a um conjunto das ocorrências (especializadas) de uma entidade genérica.



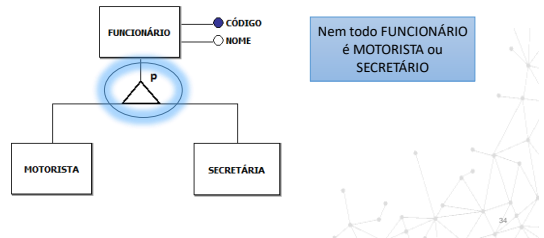
### Generalização/Especialização Total

- Para cada ocorrência da entidade genérica existe sempre uma ocorrência em uma das entidades especializadas.

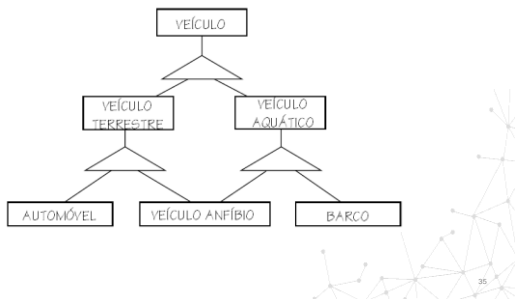


### Generalização/Especialização Parcial

- Nem toda ocorrência da entidade genérica possui uma ocorrência correspondente em uma entidade especializada



### Generalização/Especialização Múltiplos Níveis e Herança Múltipla



### Atributo

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



## Atributos de Relacionamentos

- Assim como entidades possuem atributos, também relacionamentos podem possuir atributos.

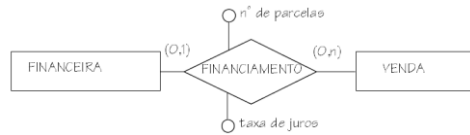


ATUAÇÃO possui um atributo, a função que um engenheiro exerce dentro de um projeto.

- Não pode ser considerada atributo de ENGENHEIRO, já que um engenheiro pode atuar em diversos projetos exercendo diferentes funções.
- Não é atributo de PROJETO, já que, em um projeto, podem atuar diversos engenheiros com funções diferentes.

## Atributos de Relacionamentos

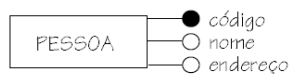
- Exemplo



Algumas vendas são à vista, outras à prazo. Vendas à prazo são relacionadas a uma financeira, através do relacionamento FINANCIAMENTO. Os atributos nº de parcelas e taxa de juros são atributos do relacionamento.

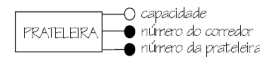
## Identificando Entidades

- Cada entidade deve possuir um identificador.
- Um identificador é um conjunto de um ou mais atributos (e possivelmente relacionamentos, como visto abaixo) cujos valores servem para distinguir uma ocorrência da entidade das demais ocorrências da mesma entidade.



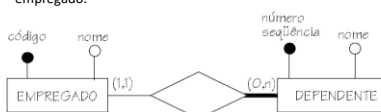
## Identificando Entidades

- Considera-se um almoxarifado de uma empresa de ferragens organizado como segue.
  - Os produtos ficam armazenados em prateleiras.
  - Estas prateleiras encontram-se em armários organizados em corredores.
  - Os corredores são numerados sequencialmente a partir de um e as prateleiras são numeradas sequencialmente a partir de um dentro de um corredor.
  - Assim, para identificar uma prateleira é necessário conhecer seu número e o número do corredor em que se encontra.
  - Para cada prateleira deseja-se saber sua capacidade em metros cúbicos.



## Identificando Entidades

- Há casos em que o identificador de uma entidade é composto não somente por atributos da própria entidade mas também por relacionamentos dos quais a entidade participa (relacionamento identificador).
- Exemplo:
  - Empregados e seus dependentes para fins de imposto de renda.
  - Cada dependente está relacionado a exatamente um empregado.
  - Um dependente é identificado pelo empregado ao qual ele está relacionado e por um número de sequência que distingue os diferentes dependentes de um mesmo empregado.



## Identificando Relacionamentos

- Uma ocorrência de relacionamento diferencia-se das demais ocorrências do mesmo relacionamento pelas ocorrências de entidades de dela participam.
- Há casos nos quais entre as mesmas ocorrências de entidades podem existir diversas ocorrências de relacionamentos.
  - Exemplo:





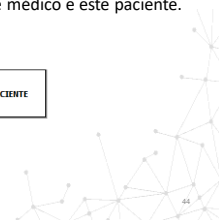
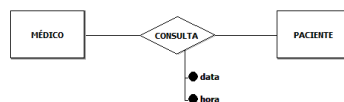
Banco de Dados I

## Construindo Modelos Entidade-Relacionamento

43

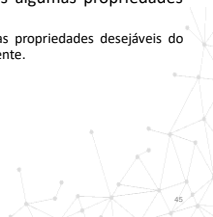
### Identificando Relacionamentos

- Para um determinado médico e um determinado paciente podem haver diversas consultas.
- Há necessidade de algo que distinga uma consulta entre um médico e seu paciente das demais consultas entre este médico e este paciente.



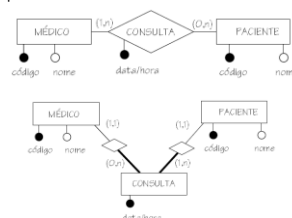
### Propriedades dos Modelos ER

- Um DER é um modelo formal, preciso, não ambíguo.
  - Diferente leitores de um mesmo DER devem sempre entender exatamente o mesmo.
- Em um modelo ER, são apresentadas apenas algumas propriedades de um banco de dados.
  - Um Diagrama ER é muito pouco poderoso e muitas propriedades desejáveis do banco de dados necessitam ser anotadas adicionalmente.



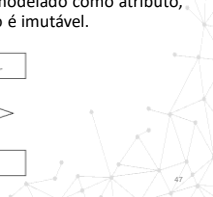
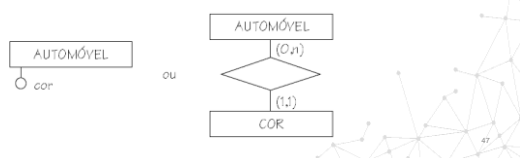
### Propriedades dos Modelos ER

- Modelos diferentes podem ser equivalentes
  - Geram o mesmo banco de dados.
  - Diz-se que dois modelos são equivalentes, quando expressam o mesmo, ou seja quando modelam a mesma realidade.
  - Exemplo



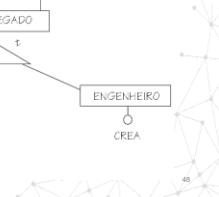
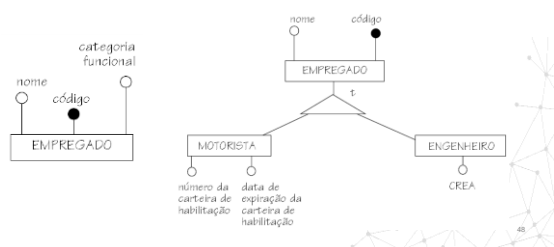
### Atributo versus Entidade Relacionada

- Caso o objeto cuja modelagem está em discussão esteja vinculado a outros objetos (atributos, relacionamentos, entidades genéricas ou especializadas), o objeto deve ser modelado como entidade
- Quando o conjunto de valores de um determinado objeto é fixo durante toda a vida do sistema ele pode ser modelado como atributo, visto que o domínio de valores de um atributo é imutável.



### Atributo versus Generalização/Especialização

- Uma especialização deve ser usada quando sabe-se que as classes especializadas de entidades possuem propriedades articulares





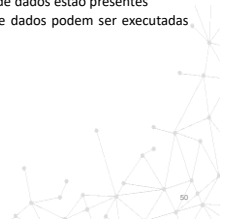
### Modelo deve ser CORRETO

- Um modelo está correto quando não contém erros de modelagem
  - Quando os conceitos de modelagem ER são corretamente empregados para modelar a realidade em questão.
  - Pode-se distinguir entre dois tipos de erros:
    - Erros sintáticos
      - Ocorrem quando o modelo não respeita as regras de construção de um modelo ER.
    - Erros semânticos
      - Ocorrem quando o modelo reflete a realidade de forma inconsistente.



### Modelo deve ser COMPLETO

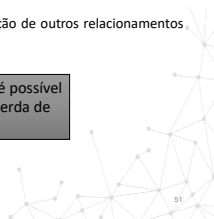
- Um modelo completo deve fixar todas propriedades desejáveis do banco de dados.
- Deve-se verificar
  - Se todos os dados que devem ser obtidos do banco de dados estão presentes
  - Se todas as transações de modificação do banco de dados podem ser executadas sobre o modelo.



### Modelo deve ser Livre de Redundâncias

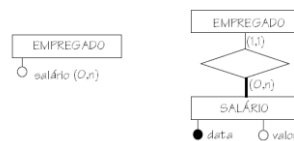
- Um modelo deve ser mínimo, isto é não deve conter conceitos redundantes.
- Um tipo de redundância que pode aparecer é a de relacionamentos redundantes:
  - São relacionamentos que são resultado da combinação de outros relacionamentos entre as mesmas entidades.

Um relacionamento é redundante, quando é possível eliminá-lo do diagrama ER, sem que haja perda de informações no banco de dados.



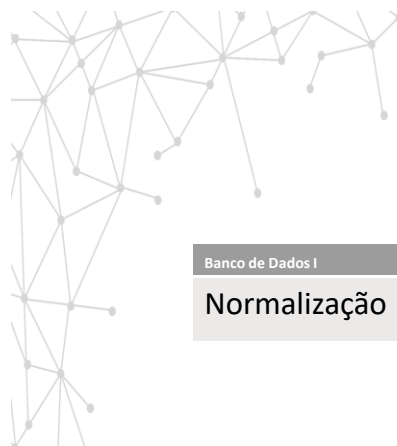
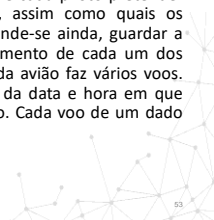
### Atributos cujos valores se modificam ao longo do tempo

- Alguns atributos de uma entidade podem ter seus valores alterados ao longo do tempo.
- Algumas vezes o banco de dados deve manter um registro histórico das informações.



### Exercício

- Uma transportadora aérea pretende implementar uma base de dados com a seguinte informação: A transportadora tem vários aviões. Cada avião tem, além da matrícula, um nome, o modelo do avião, o número de lugares, e a indicação da sua autonomia. Na transportadora trabalham vários pilotos. Sobre cada piloto pretende-se guardar o nome e número de licença, assim como quais os modelos de aviões que podem pilotar. Pretende-se ainda, guardar a informação relativa ao nome, data de nascimento de cada um dos descendentes (caso existam) dos pilotos. Cada avião faz vários voos. Cada voo deve ter, pelo menos, a indicação da data e hora em que acontecerá dos locais de partida e de destino. Cada voo de um dado avião é pilotado por um piloto.



Banco de Dados I

### Normalização

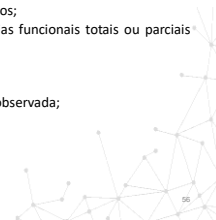
## Normalização

- O conceito de normalização foi introduzido por E. F. Codd em 1972.
- Inicialmente Codd criou as três primeiras formas de normalização chamando-as de: primeira forma normal (1NF), segunda forma normal (2NF) e terceira forma normal (3NF). Uma definição mais forte da 3NF foi proposta depois por Boyce-Codd, e é conhecida como forma normal de Boyce-Codd (FNBC).



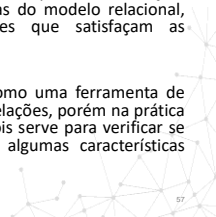
## Normalização

- Através do processo de normalização pode-se, gradativamente, substituir um conjunto de entidades e relacionamentos por um outro, o qual se apresenta "purificado" em relação às anomalias de atualização (inclusão, alteração e exclusão) as quais podem causar certos problemas, tais como:
  - grupos repetitivos (atributos multivalorados) de dados;
  - variação temporal de certos atributos, dependências funcionais totais ou parciais em relação a uma chave concatenada;
  - redundâncias de dados desnecessárias;
  - perdas acidentais de informação;
  - dificuldade na representação de fatos da realidade observada;
  - dependências transitivas entre atributos.



## O que é Normalização?

- “Técnicas de racionalização das estruturas de dados de um sistema, eliminando redundâncias, problemas de manipulação e armazenamento”.
- Normalização é um processo através do qual esquemas de relação, que não sejam satisfatórios às características do modelo relacional, são decompostos em esquemas menores que satisfaçam as propriedades desejáveis.
- A normalização inicialmente foi proposta como uma ferramenta de auxílio no projeto físico para a definição de relações, porém na prática tornou-se uma ferramenta de verificação, pois serve para verificar se os esquemas do projeto físico satisfazem algumas características básicas.



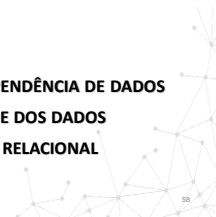
## Outras definições sobre Normalização

- “Evita problemas provocados por falhas no Projeto do Banco de Dados, eliminando uma incorreta disposição dos dados e sua redundância”.

### APLICAÇÃO DE REGRAS A UM CONJUNTO DE DADOS

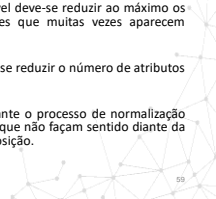


**ELIMINAÇÃO DE REDUNDÂNCIA E DEPENDÊNCIA DE DADOS**  
**AUMENTO DA CONFIABILIDADE DOS DADOS**  
**VALIDAÇÃO COM O MODELO RELACIONAL**



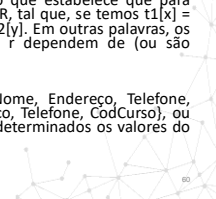
## Medidas de Qualidade na Normalização

- Na normalização, são analisadas algumas medidas de qualidade para o projeto de um esquema de relação. Estas medidas de qualidade visam, por exemplo, evitar um mau uso da memória. As medidas são as seguintes:
  1. Correta representação semântica – os dados devem ser projetados de forma a terem seus significados bem definidos e coerentes com o que realmente querem representar;
  2. Redução de valores redundantes – sempre que possível deve-se reduzir ao máximo os valores redundantes desnecessários, ou seja, valores que muitas vezes aparecem repetidos quando isto não seria preciso;
  3. Redução de valores nulos – sempre que possível deve-se reduzir o número de atributos que por alguma razão receberão muitos valores nulos;
  4. Não geração de tuplas espúrias (sem sentido) – durante o processo de normalização deve-se atentar para evitar que sejam geradas tuplas que não façam sentido diante da realidade, isto pode ocorrer devido a alguma decomposição.



## Dependências Funcionais

- As decomposições da normalização ocorrem seguindo os conceitos de dependências funcionais. A seguir estes conceitos serão apresentados, juntamente com as três principais e mais utilizadas formas normais.
- Uma dependência funcional (DF) é uma propriedade da semântica ou do significado dos atributos. Formalmente, uma dependência funcional entre dois conjuntos de atributos,  $x$  e  $y$ , que são subconjuntos de um esquema de relação  $R$ , denotada por  $x \twoheadrightarrow y$  é uma restrição que estabelece que para quaisquer tuplas  $t_1$  e  $t_2$  de uma instância  $r$  de  $R$ , tal que, se temos  $t_1[x] = t_2[x]$ , então também devemos ter que  $t_1[y] = t_2[y]$ . Em outras palavras, os valores do componente  $y$  em uma tupla de  $r$  dependem de (ou são determinados por) valores do componente  $x$ .
- Por exemplo, seja  $x = \{\text{Matrícula}\}$  e  $y = \{\text{Nome, Endereço, Telefone, CodCurso}\}$ , então  $\{\text{Matrícula}\} \twoheadrightarrow \{\text{Nome, Endereço, Telefone, CodCurso}\}$ , ou seja, a depender do valor da Matrícula, serão determinados os valores do Nome, Endereço, Telefone, e CodCurso.



Dependências Funcionais

- Quando se define uma dependência funcional, esta regra deve valer para todas as instâncias da relação. É como se fosse uma restrição de integridade. Não se pode deduzir a existência de uma dependência funcional baseado no conteúdo de uma tabela, porém a ausência da DF pode ser definida. Por exemplo:

| PROFESSOR     | DISCIPLINA                  | PERIODO |
|---------------|-----------------------------|---------|
| Olinda        | Bancos de Dados             | 4       |
| Olinda        | Sistemas de Informação      | 6       |
| Wiliam        | Arquitetura de Computadores | 6       |
| Wiliam        | Sistemas Digitais II        | 4       |
| Antônio Maria | Bancos de Dados             | 5       |

Posso dizer que: DISCIPLINA  $\not\rightarrow$  PROFESSOR  
DISCIPLINA  $\not\rightarrow$  PERIODO  
PROFESSOR  $\not\rightarrow$  DISCIPLINA  
Não posso dizer: DISCIPLINA  $\rightarrow$  PERIODO (embora pareça verdade)

Formas Normais Baseadas em Chaves Primárias

- O processo de normalização é realizado gradativamente através de formas normais, definidas a partir do conceito de DF. As três principais formas normais são a Primeira Forma Normal (1FN), a Segunda Forma Normal (2FN) e a Terceira Forma Normal (3FN).
  - 1FN 2FN 3FN
- Propriedades do processo de normalização através de decomposição: **Junções sem perda** – uma vez definida uma decomposição, caso esta seja recomposta através de uma operação de junção, no resultado gerado não pode haver perdas.
- Preservação de dependências** – assegura que cada DF seja representada em algumas relações individuais resultantes após a decomposição.

Primeira Forma Normal – (1FN)

- Um esquema de relação R está na 1FN se todos os seus atributos forem **atômicos e monovalorados**, ou seja, **não possuem valores que formam atributos compostos**.
- Exemplo 1:

ESTUDANTES = {MATRÍCULA + NOME + ENDEREÇO + CODCURSO} e  
ENDEREÇO é um atributo composto,  
ENDEREÇO = {RUA + NUMERO + BAIRRO + CIDADE + UF}.  
Para colocar na 1FN faz:  
ESTUDANTES = {MATRÍCULA-NOME-RUA-NUMERO-BAIRRO-CIDADE-UF-CODCURSO}

Primeira Forma Normal – (1FN)

- Um esquema de relação R está na 1FN se todos os seus atributos forem **atômicos e monovalorados**, ou seja, **não possuem valores que formam atributos compostos**.
- Exemplo 2:

FUNCIONÁRIOS = {CODFUNC + NOME + CARGO + {PROJETO + DATAINI + DATAFIM}}

Para colocar na 1FN faz:

FUNCIONÁRIOS = {CODFUNC + NOME + CARGO}  
FUNC\_PROJ = {CODFUNC + PROJETO + DATAINI + DATAFIM}

- Observação: todas as tabelas são relações na 1FN.

Segunda Forma Normal – (2FN)

- Dependência funcional total ou completa: Uma DF  $x^* y$  é **total, se não existir** nenhum atributo A em x, tal que  $(x - \{A\})^* y$ , para qualquer A! x, ou seja, se retirarmos este atributo A da relação x a DF deixa de existir. Caso contrário,  $x^* y$  é **parcial**.
- Definição da 2FN:** Um esquema de relação está na 2FN se: estiver na 1FN e, além disso, todo atributo que não pertença a alguma de suas chaves for **totalmente dependente da sua chave primária**. Em outras palavras, para que uma relação esteja na 2FN é preciso que esteja na 1FN e que, havendo uma chave primária composta, todos os dados que não são chaves dependem de toda a chave primária (a chave primária completa).

Segunda Forma Normal – (2FN)

- Seja o exemplo de uma relação que represente o estoque de um estabelecimento comercial, da seguinte forma:  
ESTOQUE = {PRODUTO + ALMOX + END\_ALMOX + UNID\_EST + QTD + PRECO}
- Não está na 2FN porque alguns dados não chave dependem somente de parte da chave, como END\_ALMOX depende só de ALMOX, e UNID\_EST depende só de PRODUTO.
- Com a normalizando ficaria:  
Estoque = {PRODUTO + UNID\_EST}  
Almoxarifado = {ALMOX + END\_ALMOX}  
Produto = {PRODUTO + ALMOX + QTD + PRECO}

Terceira Forma Normal – (3FN)

- Dependência funcional transitiva: Uma DF  $x^* \rightarrow y$  é **transitiva em um esquema** de relação R se existir um conjunto de atributos z, que não seja um subconjunto de alguma chave de R, e as DFs  $x^* \rightarrow z$  e  $z \rightarrow y$  forem válidas em R.
- Um esquema de relação está na 3FN se: estiver na 2FN e, além disso, nenhum atributo que não pertença a alguma das suas chaves for transitivamente dependente da sua chave primária. **Em outras palavras, para que uma relação esteja na 3FN é preciso que esteja na 2FN e todo atributo, que não pertença a alguma chave for não dependente de algum outro atributo, que também não pertença a alguma chave.**

Terceira Forma Normal – (3FN)

- Seja o exemplo de uma relação que represente os dados referentes às músicas, da seguinte forma:
- Exemplo:  
**MÚSICA = {CODIGO + TITULO + GENERO + PAIS\_ORIGEM}**, supondo que neste exemplo, o PAIS\_ORIGEM refere-se ao GÊNERO musical e não a música, sendo assim, apesar de estar na 2FN, a relação não está na 3FN, pois existe dependência entre GÊNERO e PAIS\_ORIGEM.
- Com a normalizando ficaria:  
**MUS\_1 = {CODIGO + TITULO + GENERO}**  
**MUS\_2 = {GENERO + PAIS\_ORIGEM}**

Resumo Formas Normais Baseadas em Chaves Primárias

| Forma Normal   | Teste   | Solução (Normalização)   |
|----------------|---|--|
| Primeira (1FN) | A relação não deve ter qualquer atributo não-atômico nem relações agrupadas.  | Forme novas relações para cada atributo não-atômico ou relação aninhada.   |
| Segunda (2FN)  | Para as relações nas quais a chave primária contém múltiplos atributos, nenhum atributo não-chave deve ser funcionalmente dependente de uma parte da chave primária.  | Decomponha e monte uma relação para cada chave parcial com seu(s) atributo(s) dependente(s). Mantenha uma relação com a chave primária original e quaisquer atributos que sejam completamente dependentes dela em termos funcionais. |
| Terceira (3FN) | A relação não deve ter um atributo não-chave funcionalmente determinado por um outro atributo não-chave (ou por um conjunto de atributos não-chave). Ou seja, não deve haver dependência transitiva de um atributo não-chave na chave primária. | Decomponha e monte uma relação que inclua o(s) atributo(s) não-chave que funcionalmente determine(m) outros atributos não-chave.   |

Exercícios Teóricos

- O que é uma dependência funcional?
- Quem especifica as dependências funcionais que se mantêm (são válidas) entre os atributos de um esquema de relação?
- A que se refere a expressão “relação desnormalizada”?
- Defina primeira, segunda e terceira formas normais quando somente chaves primárias são consideradas. Como as definições da 2FN e 3FN, que consideram todas as chaves de uma relação, diferem daquelas que consideram somente chaves primárias?

Exercício Prático

- Sejam os seguintes dados de uma locadora de automóveis:
    - A locadora aluga automóveis de uma determinada marca apenas para clientes pessoa jurídica (empresas).
    - Estes clientes credenciam motoristas para utilizarem os veículos.
    - Preço diário de aluguel e a potência do carro dependem de seu modelo.
  - Considerando que a locadora necessite, para seu controle, dos dados descritos na seguinte relação:
- REGISTRO\_ALUGUEL = {NumCNH + NomeMotorista + DataNasc + CGCCliente + NomeCliente + EndCliente + {PlacaCarro + Modelo + Cor + Potência + QTDDiárias + PreçoDiária}}
- Faça a normalização.

Iniciando a Normalização

- Esse processo pode ser feito em até seis fases, conhecidas como formas normais, mas geralmente, ao se chegar na terceira etapa ou terceira forma normal, já é possível obter um modelo de dados estável.
- Para iniciar o processo de normalização, é importante identificar o grupo de dados a ser analisado.
- Tomemos como exemplo a situação abaixo:
  - Temos a necessidade de criar uma tabela com informações de funcionários, onde vamos armazenar o Nome, Endereço do Escritório, e alguns recados para cada funcionário.

Exemplo de Normalização

• Pode-se começar com a construção da tabela abaixo:

| Funcionarios |                      |               |               |
|--------------|----------------------|---------------|---------------|
| Nome         | Endereco_escritorio  | Recado1       | Recado2       |
| José Silva   | Rua Manga, 1223      | Ligar em casa | Reunião 14:00 |
| Expedito Jr. | Rua Tiradentes, 3454 | Imprimir doc  | Demitir José  |

- Observe os campos Recado1 e Recado2 ...
- O que deve ser feito se o aplicativo precisar adicionar um terceiro recado?
- Teremos que continuar adicionando colunas na tabela. Até quando ???
- Isso elimina uma correta funcionalidade do sistema, que deve crescer de acordo com a necessidade sem esse tipo de problema.

Primeira Forma Normal – (1FN)

- Deve-se:
  - eliminar grupos de dados repetitivos da tabela.
  - criar uma chave que identifique cada registro.

• Observe que esta regra não é cumprida ao se repetir os campos Recado1 e Recado2

| Funcionarios |                      |               |               |
|--------------|----------------------|---------------|---------------|
| Nome         | Endereco_escritorio  | Recado1       | Recado2       |
| José Silva   | Rua Manga, 1223      | Ligar em casa | Reunião 14:00 |
| Expedito Jr. | Rua Tiradentes, 3454 | Imprimir doc  | Demitir José  |

Primeira Forma Normal – (1FN)

• Ao aplicar as regras da Primeira Forma Normal tem-se a seguinte tabela:

| Funcionarios |              |                         |               |
|--------------|--------------|-------------------------|---------------|
| Fun_Id       | Fun_Nome     | Fun_Endereco_Escritorio | Fun_Recado    |
| 1            | José Silva   | Rua Manga, 1223         | Ligar em casa |
| 2            | José Silva   | Rua Manga, 1223         | Reunião 14:00 |
| 3            | Expedito Jr. | Rua Tiradentes, 3454    | Imprimir doc  |
| 4            | Expedito Jr. | Rua Tiradentes, 3454    | Demitir José  |

Primeira Forma Normal – (1FN)

- Agora a tabela está na Primeira Forma Normal (1FN).
- Solucionamos o problema da limitação de campos de Recados, mas observe o novo problema gerado.
- Cada vez que inserimos uma nova linha na tabela de funcionários, duplicamos todos os endereços dos escritórios e do nome do funcionários.
- Este banco de dados não vai apenas ficar muito maior do que o esperado, como facilmente podemos ter dados corrompidos devido a alta redundância.
- É o momento de aplicar as regras da Segunda Forma Normal (2FN).

Segunda Forma Normal – (2FN)

- Deve-se:
  - checar se a tabela está na 1FN;
  - identificar e representar em uma nova tabela os dados que causam a redundância;
  - identificar e representar em uma nova tabela os dados que não dependam única e exclusivamente da chave da tabela.
- Extraímos os valores de Recados para uma tabela separada para que possamos adicionar mais Recados no futuro sem ter que duplicar os dados. O relacionamento irá ocorrer com a utilização do valor de chave primária.



Segunda Forma Normal – (2FN)

| Funcionarios |              |                         |
|--------------|--------------|-------------------------|
| Fun_ID       | Fun_Nome     | Fun_Endereco_Escritorio |
| 1            | José Silva   | Rua Manga, 1223         |
| 2            | Expedito Jr. | Rua Tiradentes, 3454    |

| Recados |            |               |
|---------|------------|---------------|
| Rec_ID  | Rec_Fun_ID | Rec_Mensagem  |
| 1       | 1          | Ligar em casa |
| 2       | 1          | Reunião 14:00 |
| 3       | 2          | Imprimir doc  |
| 4       | 2          | Demitir José  |

Segunda Forma Normal – (2FN)

- Ótimo! Foram criadas tabelas separadas e a chave primária na tabela de funcionarios, Fun\_ID, agora está relacionada à chave estrangeira na tabela dos Recados;
- E o campo Fun\_Endereco\_Escritorio ???
- Se inserir um outro funcionário que atue no escritório da Rua Manga ???

| Funcionarios |              |                         |
|--------------|--------------|-------------------------|
| Fun_ID       | Fun_Nome     | Fun_Endereco_Escritorio |
| 1            | José Silva   | Rua Manga, 1223         |
| 2            | Expedito Jr. | Rua Tiradentes, 3454    |
| 3            | Milton Souza | Rua Manga, 1223         |

Segunda Forma Normal – (2FN)

- E agora ???
- O Escritório irá existir, mesmo que não exista o funcionário.
- Com isso, podemos dizer que o Escritório não depende do Funcionário.



identificar e representar em uma nova tabela os dados que não dependam única e exclusivamente da chave da tabela.

Segunda Forma Normal – (2FN)

- Agora temos a chave primária Esc\_ID na tabela de Escritorios relacionada à chave estrangeira na tabela de Funcionarios chamada Fun\_Esc\_ID.

| Funcionarios |              |            |
|--------------|--------------|------------|
| Fun_ID       | Fun_Nome     | Fun_Esc_ID |
| 1            | José Silva   | 1          |
| 2            | Expedito Jr. | 2          |

| Escritorios |                      |
|-------------|----------------------|
| Esc_ID      | Esc_Endereco         |
| 1           | Rua Manga, 1223      |
| 2           | Rua Tiradentes, 3454 |

Segunda Forma Normal – (2FN)

| Funcionarios |              |            |
|--------------|--------------|------------|
| Fun_ID       | Fun_Nome     | Fun_Esc_ID |
| 1            | José Silva   | 1          |
| 2            | Expedito Jr. | 2          |

| Escritorios |                      |
|-------------|----------------------|
| Esc_ID      | Esc_Endereco         |
| 1           | Rua Manga, 1223      |
| 2           | Rua Tiradentes, 3454 |

| Recados |            |               |
|---------|------------|---------------|
| Rec_ID  | Rec_Fun_ID | Rec_Mensagem  |
| 1       | 1          | Ligar em casa |
| 2       | 1          | Reunião 14:00 |
| 3       | 2          | Imprimir doc  |
| 4       | 2          | Demitir José  |

Terceira Forma Normal – (3FN)

- Deve-se:
  - checar se a tabela está na 2FN;
  - identificar atributos com dependência transitiva e eliminá-los.
- Dependência transitiva ocorre quando um dado pode ser obtido por meio de outro (com exceção da chave primária).Tente localizar campos que possam ser substituídos por fórmulas matemáticas. Por exemplo, se em um tabela de Clientes tivessemos os campos Cli\_Nascimento e Cli\_Idade

| Clientes |               |                |           |               |
|----------|---------------|----------------|-----------|---------------|
| Cli_ID   | Cli_Nome      | Cli_Nascimento | Cli_Idade | Cli_Cidade_ID |
| 1        | Marina Souza  | 11/04/1936     | 70        | 1             |
| 2        | Francisco Jr. | 01/05/1980     | 25        | 1             |

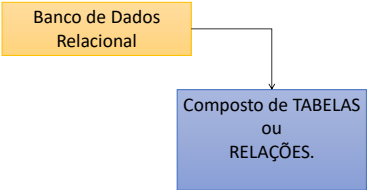
Terceira Forma Normal – (3FN)

- Neste caso, o campo Cli\_Idade deve ser eliminado, afinal, através da data de nascimento pode-se obter a idade do cliente.

| Clientes |               |                |               |
|----------|---------------|----------------|---------------|
| Cli_ID   | Cli_Nome      | Cli_Nascimento | Cli_Cidade_ID |
| 1        | Marina Souza  | 11/04/1936     | 1             |
| 2        | Francisco Jr. | 01/05/1980     | 1             |



Modelo Relacional



Tabela

- Conjunto não ordenado de linhas
  - Também chamadas TUPLAS
- Cada linha é composta por uma série de campos
  - VALOR DE ATRIBUTO
- Cada campo é identificado por nome de campo
  - NOME DO ATRIBUTO
- O conjunto de campos das linhas de uma tabela que possuem o mesmo nome formam uma coluna.

Tabela

| Emp       |        |             |                |
|-----------|--------|-------------|----------------|
| CódigoEmp | Nome   | CódigoDepto | CategFuncional |
| E5        | Souza  | D1          | C5             |
| E3        | Santos | D2          | C5             |
| E2        | Silva  | D1          | C2             |
| E1        | Soares | D1          | —              |

coluna (atributo)

nome do campo (nome do atributo)

linha (tupla)

valor de campo (valor de atributo)

Tabela

- As linhas de uma tabela não estão ordenadas.
- Os valores de campo de uma tabela são atômicos e mono-valorados.
- Linguagens de consulta a BD Relacionais
  - Permitem o acesso por quaisquer critérios envolvendo os campos de uma ou mais linhas.

Chaves

- Em um banco de dados relacional, há dois principais tipos de chaves a considerar:
  - Chave primária
  - Chave estrangeira.

Chave Primária

- Coluna ou uma combinação de colunas cujos valores distinguem uma linha das demais dentro de uma tabela.

Empregado

| CódigoEmp | NOME               | DATA_NASC  |
|-----------|--------------------|------------|
| E1        | Carlos Amaral      | 01.12.1979 |
| E2        | Fernanda da Silva  | 30.06.1977 |
| E3        | Marcos Nascimento  | 23.09.1973 |
| E4        | Lia Noal           | 12.12.1961 |
| E5        | João Carlos Santos | 24.05.1987 |

Dependente

| CódigoEmp | NoDepen | Nome  | Tipo   | DataNasc |
|-----------|---------|-------|--------|----------|
| E1        | 01      | João  | Filho  | 12/12/91 |
| E1        | 02      | Maria | Esposa | 01/01/50 |
| E2        | 01      | Ana   | Esposa | 05/11/55 |
| E5        | 01      | Paula | Esposa | 04/07/60 |
| E5        | 02      | José  | Filho  | 03/02/85 |

Chave Estrangeira

- Coluna ou uma combinação de colunas, cujos valores aparecem necessariamente na chave primária de uma tabela.

| Dept        |            |
|-------------|------------|
| CodigoDepto | NomeDepto  |
| D1          | Compras    |
| D2          | Engenharia |
| D3          | Vendas     |

| Emp       |        |             |                |                |
|-----------|--------|-------------|----------------|----------------|
| CodigoEmp | Nome   | CodigoDepto | CategFuncional | CPF            |
| E1        | Souza  | D1          | -              | 132.121.331-20 |
| E2        | Santos | D2          | C5             | 891.221.111-11 |
| E3        | Silva  | D2          | C5             | 341.511.775-45 |
| E5        | Soares | D1          | C2             | 631.692.754-88 |

Chave Estrangeira

- Quando da inclusão de uma linha na tabela que contém a chave estrangeira
  - Deve ser garantido que o valor da chave estrangeira apareça na coluna da chave primária referenciada.
- Quando da alteração do valor da chave estrangeira
  - Deve ser garantido que o novo valor de uma chave estrangeira apareça na coluna da chave primária referenciada.
- Quando da exclusão de uma linha da tabela que contém a chave primária referenciada pela chave estrangeira
  - Deve ser garantido que na coluna chave estrangeira não apareça o valor da chave primária que está sendo excluída.

Domínios e Valores Vazios

- Quando uma tabela do banco de dados é definida, para cada coluna da tabela, deve ser especificado um conjunto de valores (alfanumérico, numérico,...) que os campos da respectiva coluna podem assumir.
  - Este conjunto de valores é chamado de domínio da coluna ou domínio do campo.
- Deve ser especificado se os campos da coluna podem estar vazios ("null" em inglês) ou não.

Restrições de Integridade

- Um dos objetivos primordiais de um SGBD é a integridade de dados.
- Dados de um BD estão íntegros
  - Quando refletem corretamente a realidade representada pelo BD e são consistentes entre si.
- Restrição de Integridade
  - Regra de consistência de dados que é garantida pelo próprio SGBD.
- Devem ser garantidas automaticamente por um SGBD relacional
  - Não deve ser exigido que o programador escreva procedimentos para garanti-las explicitamente.

Integridade de domínio

- Especifica que o valor de um campo deve obedecer a definição de valores admitidos para a coluna
  - Domínio da coluna
- Nos SGBD relacionais comerciais, é possível usar apenas domínios pré-definidos
  - Número inteiro, número real, alfanumérico de tamanho definido, data, etc.
- O usuário do SGBD não pode definir domínios próprios de sua aplicação
  - Por exemplo, o domínio dos dias da semana ou das unidades da federação



Integridade de Vazio

- Especifica se os campos de uma coluna podem ou não ser vazios
  - Se a coluna é OBRIGATÓRIA ou OPCIONAL
- Campos que compõem a chave primária sempre devem ser diferentes de vazio.

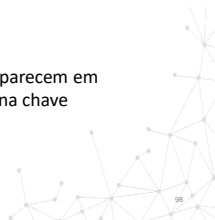


Integridade de Chave

- Trata-se da restrição que define que os valores da chave primária e alternativa devem ser únicos.

INTEGRIDADE REFERENCIAL

- Define que os valores dos campos que aparecem em uma chave estrangeira devem aparecer na chave primária da tabela referenciada.



Banco de Dados Relacional

- A especificação de um banco de dados relacional deve conter no mínimo a definição do seguinte:
  - Tabelas que formam o banco de dados
  - Colunas que as tabelas possuem
  - Restrições de integridade



Especificação de BD Relacional

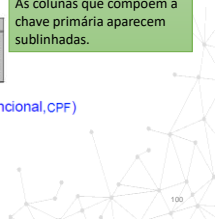
| Dept        |            |
|-------------|------------|
| CodigoDepto | NomeDepto  |
| D1          | Compras    |
| D2          | Engenharia |
| D3          | Vendas     |

| Emp       |        |             |                |                |
|-----------|--------|-------------|----------------|----------------|
| CodigoEmp | Nome   | CodigoDepto | CategFuncional | CPF            |
| E1        | Souza  | D1          | -              | 132.121.331-20 |
| E2        | Santos | D2          | C5             | 891.221.111-11 |
| E3        | Silva  | D2          | C5             | 341.511.775-45 |
| E5        | Soares | D1          | C2             | 631.692.754-88 |

Emp (CodigoEmp, Nome, CodigoDepto, CategFuncional, CPF)  
CodigoDept referencia Dept  
Dept (CodigoDepto, Nome)

São listadas as TABELAS e, para cada tabela, enumerados, entre parênteses, os nomes das COLUNAS que compõem a tabela. As colunas que compõem a chave primária aparecem sublinhadas.



Especificação de BD Relacional

- Após a definição da tabela aparecem as definições das chaves estrangeiras que aparecem na tabela na forma:

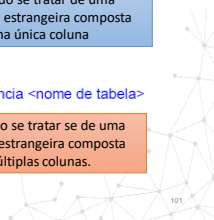
<nome de coluna ch. estrangeira> referencia <nome de tabela>

Quando se tratar de uma chave estrangeira composta de uma única coluna

- ou na forma:

(<nome de coluna>1,<nome de coluna>2,...) referencia <nome de tabela>

Quando se tratar se de uma chave estrangeira composta por múltiplas colunas.



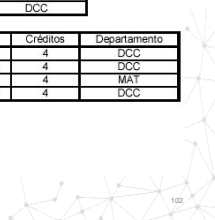
Exercícios

- Para cada um das tabelas a seguir, especifique o modelo relacional correspondente.

| ESTUDANTE | Nome | Número | Classe | Departamento |
|-----------|------|--------|--------|--------------|
| Soares    | 17   | 1      | DCC    |              |
| Botelho   | 8    | 2      | DCC    |              |

| CURSO                     | Nome    | Número | Créditos | Departamento |
|---------------------------|---------|--------|----------|--------------|
| Introd. Ciências de Comp. | DCC1310 | 4      | DCC      |              |
| Estrutura de Dados        | DCC3320 | 4      | DCC      |              |
| Matemática Discreta       | MAT2410 | 4      | MAT      |              |
| Base de Dados             | DCC3380 | 4      | DCC      |              |

| PRÉ-REQUISITO | Número  | Pré-requisito |
|---------------|---------|---------------|
| DCC3380       | DCC3320 |               |
| DCC3380       | MAT2410 |               |
| DCC3320       | DCC1310 |               |



| SEÇÃO | Número | Curso   | Semestre | Ano | Professor |
|-------|--------|---------|----------|-----|-----------|
|       | 85     | MAT2410 | 1        | 86  | Kotaro    |
|       | 92     | DCC1310 | 1        | 86  | Alberto   |
|       | 102    | DCC3320 | 2        | 87  | Kleber    |
|       | 112    | MAT2410 | 1        | 87  | Carlos    |
|       | 119    | DCC1310 | 1        | 87  | Alberto   |
|       | 135    | DCC3380 | 1        | 87  | Souza     |

| HISTORICO | NúmeroEstudante | NúmeroSeção | Nível |
|-----------|-----------------|-------------|-------|
|           | 17              | 112         | B     |
|           | 17              | 119         | C     |
|           | 8               | 85          | A     |
|           | 8               | 92          | A     |
|           | 8               | 102         | B     |
|           | 8               | 135         | A     |

Exercícios

- Abaixo aparece um esquema parcial para um banco de dados relacional. Identifique neste esquema as chaves primárias e chaves estrangeiras:

Aluno (CodigoAluno, Nome, CodigoCurso)  
Curso (CodigoCurso, Nome)  
Disciplina (CodigoDisciplina, Nome, Creditos, CodigoDepartamento)  
Curriculo (CodigoCurso, CodigoDisciplina, Obrigatória-Optional)  
Conceito (CodigoAluno, CodigoDisciplina, Ano-Semestre, Conceito)  
Departamento (CodigoDepartamento, Nome)

Exercícios

- Para o banco de dados cujo esquema está definido abaixo, explique que verificações devem ser feitas pelo SGBD para garantir integridade referencial nas seguintes situações:

- a) Uma linha é incluída na tabela *Consulta*.
- b) Uma linha é excluída da tabela *Paciente*.  
*Paciente*(CodigoConvenio, NumeroPaciente, Nome)  
*CodigoConvenio* referencia *Convenio*  
*Convenio*(CodigoConvenio, Nome)  
*Medico*(CRM, Nome, Especialização)  
*Consulta*(CodigoConvenio, NumeroPaciente, CRM, Data-Hora)  
(CodigoConvenio, NumeroPaciente) referencia *Paciente*  
*CRM* referencia *Medico*



Banco de Dados I

Engenharia Reversa de Modelos Relacionais

Introdução

- Quando se transforma modelos de banco de dados mais ricos em detalhes de implementação em modelos de dados mais abstratos.
- Ponto de partida
  - Modelo lógico de um banco de dados relacional
- Resultado
  - Modelo conceitual → Abordagem ER.

Engenharia Reversa – Modelo Relacional

- Identificação da construção ER correspondente a cada tabela
- Definição de relacionamentos 1:n e 1:1
- Definição de atributos

### Exemplo

- Disciplina (CodDisc, NomeDisc)
- Curso (CodCr, NomeCr)
- Currículo (CodCr, CodDisc, Obr\_Opc)
  - CodCr referencia Curso
  - CodDisc referencia Disciplina
- Sala (CodPr, CodSI, Capacidade)
  - CodPr referencia Prédio
- Prédio (CodPr, Endereço)
- Turma (Anosem, CodDisc, SiglaTur, Capacidade, CodPr, CodSI)
  - CodDisc referencia Disciplina
  - (CodPr,CodSI) referencia Sala
- Laboratório (CodPr, CodSI, Equipam)
  - (CodPr,CodSI) referencia Sala



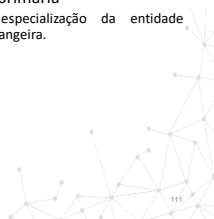
### Identificação da Construção ER correspondentes a cada tabela

- Uma tabela pode corresponder a:
  - Uma entidade
  - Um relacionamento n:n
  - Uma entidade especializada



### Identificação da Construção ER correspondentes a cada tabela

- Regra 1: Chave primária composta por mais de uma chave estrangeira
  - Implementação de um relacionamento n:n entre as entidades correspondentes às tabelas referenciadas pelas chaves estrangeiras.
- Regra 2: Toda chave estrangeira é uma chave primária
  - Representa uma entidade que forma uma especialização da entidade correspondente à tabela referenciada pela chave estrangeira.

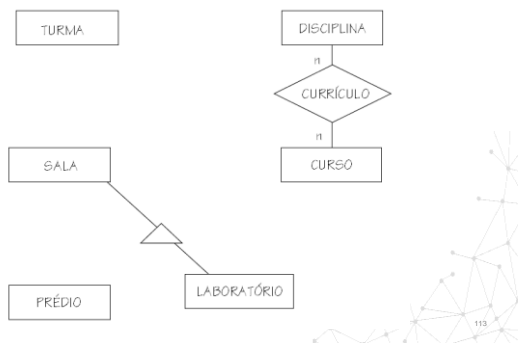


### Identificação da Construção ER correspondentes a cada tabela

- Regra 3: Demais casos
  - Quando a chave primária da tabela não for composta de múltiplas chaves primárias (regra 1 acima), nem for toda uma chave estrangeira (regra 2 acima), a tabela representa uma entidade



### Identificação da Construção ER correspondentes a cada tabela

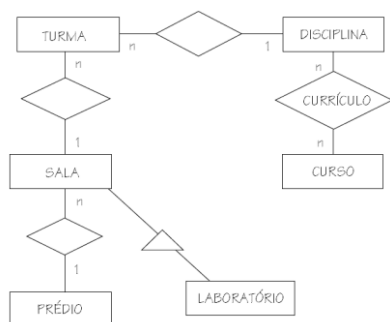


### Definição de relacionamentos 1:n e 1:1

- Toda chave estrangeira que não faz parte de uma chave primária composta por múltiplas chaves estrangeiras, nem é toda ela uma chave primária, representa um relacionamento 1:n ou 1:1.



### Definição de relacionamentos 1:n e 1:1

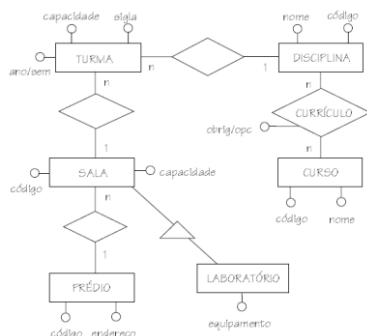


### Definição de Atributos

• Para cada coluna de uma tabela, que não seja chave estrangeira, é definido um atributo na entidade/relacionamento correspondente à tabela.

• Colunas chave estrangeira não correspondem a atributos no diagrama ER, mas sim a relacionamentos, e por isso já foram tratadas nas etapas anteriores.

### Definição de Atributos



Banco de Dados I

### Transformações entre Modelos

### Transformações entre Modelos

• Modelo Entidade Relacionamento e Modelo Relacional

- Propõe modelar os dados em diferentes níveis de abstração.

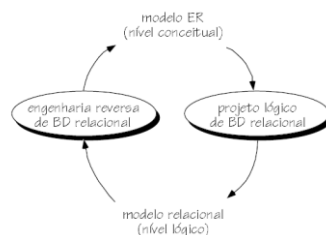
#### Modelo ER

- Modelagem de dados de forma independente do SGBD considerado.
- Adequada para construção do modelo conceitual.

#### Modelo Relacional

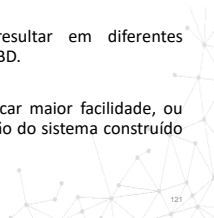
- Modelagem de dados a nível de SGBD relacional.
- Um modelo neste nível de abstração é chamado de modelo lógico.

### Transformações entre Modelos



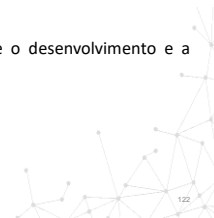
## Visão Geral do Projeto Lógico

- Um determinado modelo ER pode ser implementado através de diversos modelos relacionais
  - Que contêm as informações especificadas pelo diagrama ER.
  - Todos podem ser considerados uma implementação correta do modelo ER considerado.
- Diferentes modelos relacionais podem resultar em diferentes performances do sistema construído sobre o BD.
- Diferentes modelos relacionais podem implicar maior facilidade, ou dificuldade de desenvolvimento e manutenção do sistema construído sobre o banco de dados.



## Transformação ER para RELACIONAL

- Objetivos básicos:
  - Obter um banco de dados que permita boa performance de instruções de consulta e alteração do BD.
  - Obter boa performance significa basicamente diminuir o número de acessos a disco, já que estes consomem o maior tempo na execução de uma instrução de banco de dados.
- Obter um banco de dados que simplifique o desenvolvimento e a manutenção de aplicações.



## Transformação ER para RELACIONAL

- Passos para transformação de um modelo ER
  - Tradução inicial de entidades e respectivos atributos
  - Tradução de relacionamentos e respectivos atributos
  - Tradução de generalizações/especializações



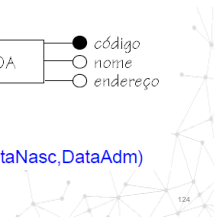
## Implementação Inicial de Entidades

- Cada entidade é traduzida para uma tabela.
  - Cada atributo da entidade define uma coluna desta tabela.
  - Atributos identificadores da entidade correspondem às colunas que compõem a chave primária da tabela.



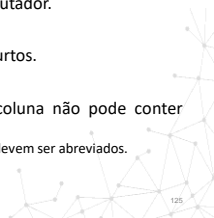
Esquema relacional correspondente:

Pessoa (CódigoPess, Nome, Endereço, DataNasc, DataAdm)



## Nomes de Atributos e Nomes de Colunas

- Não é aconselhável simplesmente transcrever os nomes de atributos para nomes de colunas.
- Nomes de colunas serão referenciados freqüentemente em programas e outras formas de texto em computador.
- É conveniente manter os nomes de colunas curtos.
- Em um SGBD relacional, o nome de uma coluna não pode conter brancos.
  - Nomes de atributos compostos de diversas palavras devem ser abreviados.



## Implementação de Relacionamentos

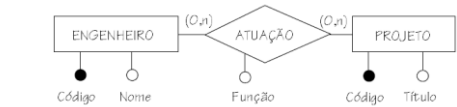
- Fator determinante → Cardinalidade das entidades que participam do relacionamento.
- Três formas básicas de tradução de relacionamentos para o modelo relacional:
  - Tabela Própria



Tabela Própria

- Esta tabela contém as seguintes colunas:
  - Colunas correspondentes aos identificadores das entidades relacionadas
  - Colunas correspondentes aos atributos do relacionamento.
- Chave primária
  - Conjunto das colunas correspondentes aos identificadores das entidades relacionadas.
- Cada conjunto de colunas que corresponde ao identificador de uma entidade
  - Chave estrangeira em relação a tabela que implementa a entidade referenciada.

Tabela Própria

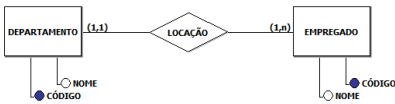


Esquema relacional correspondente:  
Engenheiro (CodEng, Nome)  
Projeto (CodProj, Título)  
Atuação (CodEng, CodProj, Função)  
CodEng referencia Engenheiro  
CodProj referencia Projeto

Usado para relacionamentos n:n

Colunas Adicionais dentro da Tabela de entidade

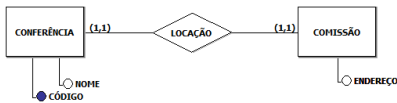
- Inserção de colunas em uma tabela correspondente a uma das entidades que participam do relacionamento.
- Usado para relacionamentos 1:n



Esquema Relacional correspondente:  
DEPARTAMENTO(codigo, nome)  
EMPREGADO(codigo, nome, codDepartamento)  
codDepartamento referencia DEPARTAMENTO (codigo)

Fusão de Tabelas de Entidade

- Somente pode ser aplicada quando o relacionamento é de tipo 1:1.

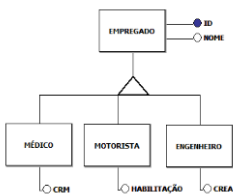


Esquema Relacional correspondente:  
CONFERÊNCIA(codigo, nome, endComissao)

Generalização/Especialização

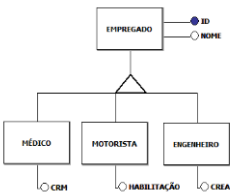
- Há duas alternativas a considerar:
  - (1) uso de uma tabela para cada entidade
  - (2) uso de uma única tabela para toda hierarquia de generalização/especialização.

Uso de uma tabela para cada entidade



Esquema Relacional correspondente:  
EMPREGADO(id, nome)  
MÉDICO(id, crm)  
MOTORISTA(id, habilitacao)  
ENGENHEIRO(id, crea)

uso de uma única tabela para toda hierarquia de generalização/especialização



Esquema Relacional correspondente:  
EMPREGADO(id, nome, crm, habilitacao, crea)

Banco de Dados I

Álgebra Relacional

Álgebra Relacional

- Utilizada para obter informações armazenadas em um banco de dados.
- Toda operação relacional opera (age) sobre um ou mais conjuntos de dados e fornece como resultado um novo conjunto.
- Pode-se combinar mais de uma operação relacional em uma única expressão algébrica.

Exemplo

- FUNCIONÁRIO (numMatric, nomeFunc, dataAdm, sexo, codCargo, codDepto)
- CARGO (codigoCargo, nomeCargo, valorSalario)
- DEPTO (codigoDepto, nomeDepto, Ramal)

Exemplo

| codigoCargo | nomeCargo          | valorSalario |
|-------------|--------------------|--------------|
| C1          | Cozinheira         | 500          |
| C3          | Aux. de Escritório | 550          |
| C7          | Vigia              | 500          |
| C2          | Mecânico           | 560          |
| C5          | Gerente            | 1500         |
| C4          | Escriturário       | 700          |

| codigoDepto | nomeDepto       | ramal |
|-------------|-----------------|-------|
| D1          | Administração   | 221   |
| D2          | Oficina         | 235   |
| D3          | Serviços Gerais | 243   |
| D4          | Vendas          | 258   |

Exemplo

| numMatric | nomeFunc        | dataAdm  | sexo | codCargo | codDepto |
|-----------|-----------------|----------|------|----------|----------|
| 1001      | João Sampaio    | 10/08/93 | M    | C2       | D2       |
| 1004      | Lúcio Torres    | 02/03/94 | M    | C2       | D2       |
| 1034      | Roberto Pereira | 23/05/92 | M    | C3       | D1       |
| 1021      | José Nogueira   | 10/11/94 | M    | C3       | D1       |
| 1029      | Ruth de Souza   | 05/01/92 | F    | C1       | D3       |
| 1095      | Maria da Silva  | 03/09/92 | F    | C4       | D1       |
| 1023      | Luiz de Almeida | 12/01/93 | M    | C2       | D2       |
| 1042      | Pedro Pinheiro  | 29/07/94 | M    | C4       | D1       |
| 1048      | Ana Silveira    | 01/06/93 | F    | C5       | D1       |
| 1015      | Pedro Rodrigues | 17/08/92 | M    | C2       | D2       |

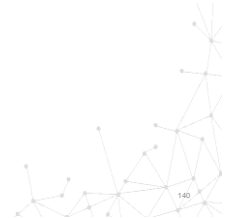
Seleção (SELECT)

- A operação SELECT é usada para selecionar um subconjunto de tuplas de uma relação as quais devem satisfazer uma **condição de seleção**.
- Indicada por  $\sigma$
- Operação que filtra as linhas de uma tabela.



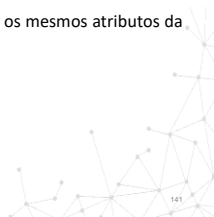
Seleção (SELECT) – Exemplo 1

- A seleção de um subconjunto de tuplas da relação EMPREGADOS que trabalham para o departamento 4 ou que tenham salário maior que 3000. Cada uma dessas condições é especificada individualmente usando a operação SELECT como segue:
- $\sigma \text{ NDEP} = 4$  (EMPREGADO)
- $\sigma \text{ SALÁRIO} > 3000$  (EMPREGADO)



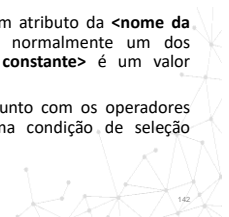
Seleção (SELECT)

- Em geral, a operação SELECT é denotada por:
  - $\sigma$  <condição de seleção> (<nome da relação>)
- onde o símbolo  $\sigma$  é usado para denotar o operador SELECT e a condição de seleção é uma expressão Booleana especificada sobre atributos da relação especificada.
- A relação resultante da operação SELECT tem os mesmos atributos da relação especificada em <nome da relação>.



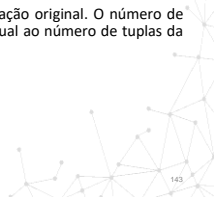
Seleção (SELECT)

- A expressão booleana especificada em <condição de seleção> é construída a partir de cláusulas da forma:
- <nome de atributo> <operador de comparação> <valor constante>, ou <nome de atributo> <operador de comparação> <nome de atributo>
- Onde <nome de atributo> é o nome de um atributo da <nome da relação>, <operador de comparação> é normalmente um dos operadores relacionais {=,<,>, ...} <valor constante> é um valor constante.
- As cláusulas podem ser utilizadas em conjunto com os operadores lógicos {AND, OR, NOT} para formar uma condição de seleção composta.



Seleção (SELECT)

- O operador SELECT é unário; isto é, ele é aplicado somente a uma relação. Assim, o SELECT não pode ser usado para selecionar tuplas de mais de uma relação.
- Observe também que a operação de seleção é aplicada individualmente para cada tupla.
- Assim, as condições de seleção não podem ser aplicadas a mais que uma tupla.
- O grau da relação resultante é a mesma que a relação original. O número de tuplas da relação resultante é sempre menor ou igual ao número de tuplas da relação original.
- Note que o operador SELECT é **comutativo**; isto é,
- $\sigma_{\langle \text{cond1} \rangle} (\sigma_{\langle \text{cond2} \rangle} (R)) = \sigma_{\langle \text{cond2} \rangle} (\sigma_{\langle \text{cond1} \rangle} (R))$



Operadores

| Relacionais |                  |
|-------------|------------------|
| Símbolo     | Operação         |
| =           | Igual a          |
| >           | Maior que        |
| <           | Menor que        |
| ≥           | Maior ou igual a |
| ≤           | Menor ou igual a |
| ≠           | Diferente de     |

| Lógicos |           |
|---------|-----------|
| Símbolo | Operação  |
| ∧       | e (and)   |
| ∨       | ou (or)   |
| ¬       | não (not) |





Seleção (SELECT) – Exemplo 2

- Suponha que se deseja selecionar as tuplas de todos os empregados que ou trabalham no departamento 4 com salário superior a R\$2.500,00 ou trabalham no departamento 5 e ganham mais que R\$3.000,00. Neste caso, pode-se especificar a consulta da seguinte forma:

•  $\sigma_{(NDEP = 4 \text{ AND SALÁRIO} > 2500) \vee (NDEP = 5 \text{ AND SALÁRIO} > 3000)}$  (EMPREGADO)



Seleção (SELECT) – Exemplo 3

- Problema: Identificar todos os funcionários de sexo masculino existentes no banco de dados.

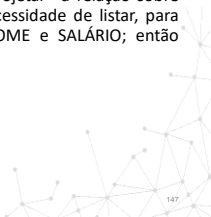
• Função Correspondente:

$\sigma_{\text{sexo} = 'M'}$  (FUNCIONÁRIO) → Resultado

| numMatric | nomeFunc        | dataAdm  | sexo | codCargo | codDepto |
|-----------|-----------------|----------|------|----------|----------|
| 1001      | João Sampaio    | 10/08/93 | M    | C2       | D2       |
| 1004      | Lúcio Torres    | 02/03/94 | M    | C2       | D2       |
| 1034      | Roberto Pereira | 23/05/92 | M    | C3       | D1       |
| 1021      | José Nogueira   | 10/11/94 | M    | C3       | D1       |
| 1023      | Luiz de Almeida | 12/01/93 | M    | C2       | D2       |
| 1042      | Pedro Pinheiro  | 29/07/94 | M    | C4       | D1       |
| 1015      | Pedro Rodrigues | 17/08/92 | M    | C2       | D2       |

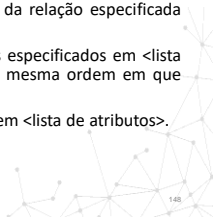
Projeção (Project)

- Pensando na relação como uma tabela, o operador SELECT seleciona algumas linhas da tabela enquanto descarta outras.
- O operador PROJECT, por outro lado, seleciona certas colunas da tabela e descarta outras. Se existir o interesse sobre certos atributos da relação, pode-se usar o PROJECT para “projetar” a relação sobre esses atributos. Por exemplo, suponha a necessidade de listar, para cada empregado, os atributos PNAME, SNAME e SALÁRIO; então pode-se usar o PROJECT como segue:
- Geralmente indicada na literatura por  $\pi$
- Operação que filtra as colunas de uma tabela.



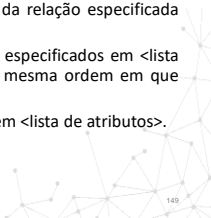
Projeção (Project)

- $\pi_{\text{SNAME, PNAME, SALÁRIO}}$  (EMPREGADO)
- $\pi_{\langle \text{lista de atributos} \rangle}$  (<nome da relação>)
- onde  $\pi$  é o símbolo usado para representar o operador PROJECT e <lista de atributos> é uma lista de atributos da relação especificada por <nome da relação>.
- A relação resultante tem apenas os atributos especificados em <lista de atributos> e estes atributos aparecem na mesma ordem em que foram especificados.
- Assim, o grau é igual ao número de atributos em <lista de atributos>.



Projeção (Project)

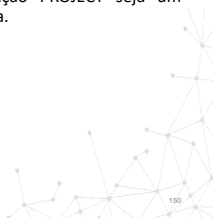
- $\pi_{\text{SNAME, PNAME, SALÁRIO}}$  (EMPREGADO)
- $\pi_{\langle \text{lista de atributos} \rangle}$  (<nome da relação>)
- onde  $\pi$  é o símbolo usado para representar o operador PROJECT e <lista de atributos> é uma lista de atributos da relação especificada por <nome da relação>.
- A relação resultante tem apenas os atributos especificados em <lista de atributos> e estes atributos aparecem na mesma ordem em que foram especificados.
- Assim, o grau é igual ao número de atributos em <lista de atributos>.



Projeção (Project)

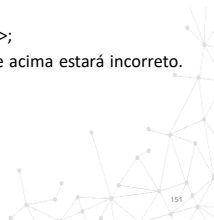
- Convém salientar que, caso a lista de atributos não contenha atributos chaves, então é provável que tuplas duplicadas apareçam no resultado.
- A operação PROJECT remove implicitamente quaisquer tuplas duplicadas, tal que o resultado da operação PROJECT seja um conjunto de tuplas e assim, uma relação válida.
- Por exemplo, considere a seguinte operação:

•  $\pi_{\text{SEXO, SALÁRIO}}$  (EMPREGADO)



Projeção (Project)

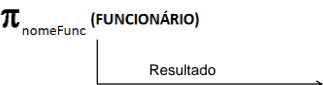
- O número de tuplas na relação resultante sempre será igual ou menor que a quantidade de tuplas na relação original.
- Note-se que:
- $\pi_{\langle \text{lista1} \rangle} (\pi_{\langle \text{lista2} \rangle} (R)) = \pi_{\langle \text{lista1} \rangle} (R)$
- Caso  $\langle \text{lista2} \rangle$  contenha os atributos de  $\langle \text{lista1} \rangle$ ;
- Caso contrário, o lado esquerdo da igualdade acima estará incorreto. A comutatividade não é válida para PROJECT.



Projeção (Project) – Exemplo 1

- Problema: Obter o nome completo de todos os funcionários cadastrados no banco de dados

• Função Correspondente:



| nomeFunc        |
|-----------------|
| João Sampaio    |
| Lúcio Torres    |
| Roberto Pereira |
| José Nogueira   |
| Ruth de Souza   |
| Maria da Silva  |
| Luiz de Almeida |
| Pedro Pinheiro  |
| Ana Silveira    |
| Pedro Rodrigues |

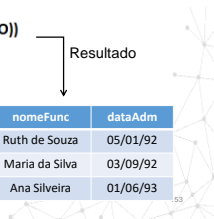


Seleção (SELECT) – Exemplo 2

- Problema: Descobrir o nome completo e a data de admissão de todos os funcionários do sexo feminino existentes na empresa
- Função Correspondente:



| nomeFunc       | dataAdm  |
|----------------|----------|
| Ruth de Souza  | 05/01/92 |
| Maria da Silva | 03/09/92 |
| Ana Silveira   | 01/06/93 |



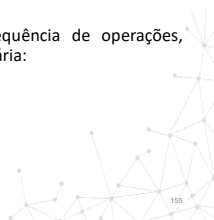
Sequência de Operações

- Em geral, existe a necessidade de se aplicar várias operações da álgebra relacional uma após a outra.
- Pode-se escrever essas operações em apenas uma única expressão da álgebra relacional ou aplicar uma única operação por vez e criar relações intermediárias.
- Neste último caso, deve-se dar nomes às relações intermediárias.



Sequência de Operações

- Por exemplo, deseja-se recuperar o primeiro nome, o último nome e o salário de todos os empregados que trabalham no departamento 5.
- Isto pode ser feito aplicando-se os operadores SELECT e PROJECT:
- $\pi_{\text{PNAME, SNAME, SALÁRIO}} (\sigma_{\text{NDEP}=5} (\text{EMPREGADO}))$
- Alternativamente, pode-se explicitar a sequência de operações, dando um nome para cada relação intermediária:
- $\text{DEP5\_EMPS} \leftarrow \sigma_{\text{NDEP}=5} (\text{EMPREGADO})$
- $\text{RESULT} \leftarrow \pi_{\text{PNAME, SNAME, SALÁRIO}} (\text{DEP5\_EMPS})$



Sequência de Operações

(a)

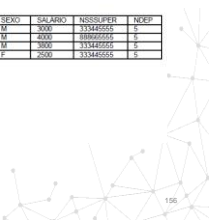
| PNAME    | SNAME   | SALÁRIO |
|----------|---------|---------|
| John     | Smith   | 3000    |
| Franklin | Vinson  | 4000    |
| Ramona   | Narayan | 3000    |
| Joyce    | English | 2500    |
| James    | Borg    | 5000    |

DEP5\_EMPS (b)

| PNAME    | SNAME | DEPT    | DATAADMI | EMPREGADO  | SEXO | SALÁRIO | NOMEPRIME | NOVELTIMO |
|----------|-------|---------|----------|------------|------|---------|-----------|-----------|
| John     | B     | Smith   | 13/07/78 | 06-2024-5  | M    | 3000    | JOHN      | SMITH     |
| Franklin | V     | Vinson  | 13/07/78 | 06-2024-5  | M    | 4000    | FRANKLIN  | VINSON    |
| Ramona   | K     | Narayan | 06/08/44 | 15-18-1-22 | M    | 3000    | RAMONA    | NARAYAN   |
| Joyce    | A     | English | 45/45/45 | 31-04-60   | F    | 2500    | JOYCE     | ENGLISH   |

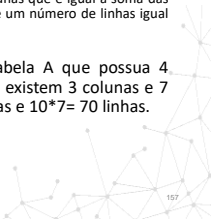
RESULT

| PRIMEIRO NOME | SOBRENOME | SALÁRIO |
|---------------|-----------|---------|
| John          | Smith     | 3000    |
| Franklin      | Vinson    | 4000    |
| Ramona        | Narayan   | 3000    |
| Joyce         | English   | 2500    |



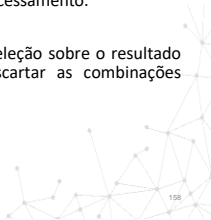
Produto Cartesiano

- Indicada por **X**
- Resultado do produto cartesiano de duas Tabelas:
  - Uma terceira tabela contendo todas as combinações possíveis entre os elementos das tabelas originais.
  - Essa tabela resultante possuirá um número de colunas que é igual à soma das quantidades de colunas das duas tabelas iniciais, e um número de linhas igual ao produto do número de suas linhas.
- Portanto, o produto cartesiano de uma tabela A que possua 4 colunas e 10 linhas com uma tabela B onde existem 3 colunas e 7 linhas, a tabela resultante terá  $4+3=7$  colunas e  $10*7=70$  linhas.



Produto Cartesiano

- Cada linha da tabela corresponderá à concatenação de uma linha da primeira tabela com uma linha da segunda.
- É a única forma primitiva de que dispomos para fundir informações de duas tabelas heterogêneas para posterior processamento.
- Tipicamente será necessário executar uma Seleção sobre o resultado do Produto Cartesiano, de maneira a descartar as combinações inválidas entre as linhas das tabelas originais.



Exemplo: Produto Cartesiano

CLIENTE(codigoCli, nomeCli, codCidade)  
CIDADE(codigoCid, nomeCid)

| CLIENTE   |         |           | CIDADE    |             |
|-----------|---------|-----------|-----------|-------------|
| codigoCli | nomeCli | codCidade | codigoCid | nomeCid     |
| 1         | Carlos  | C1        | C1        | Santa Maria |
| 2         | Sandra  | C2        | C2        | Santiago    |
| 3         | Mara    | C1        | C3        | Jaguari     |
| 4         | Jorge   | C3        |           |             |

Produto Cartesiano:  
CLIENTE X CIDADE

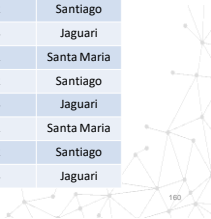
CLIENTE: 3 colunas e 4 linhas  
CIDADE: 2 colunas e 3 linhas

Resultado:  $3 + 2 = 5$  colunas  
 $4 * 3 = 12$  linhas



Resultado: CLIENTE X CIDADE

| codigoCli | nomeCli | codCidade | codigoCid | nomeCid     |
|-----------|---------|-----------|-----------|-------------|
| 1         | Carlos  | C1        | C1        | Santa Maria |
| 2         | Sandra  | C2        | C2        | Santiago    |
| 3         | Mara    | C1        | C3        | Jaguari     |
| 4         | Jorge   | C3        | C1        | Santa Maria |
| 1         | Carlos  | C1        | C2        | Santiago    |
| 2         | Sandra  | C2        | C3        | Jaguari     |
| 3         | Mara    | C1        | C1        | Santa Maria |
| 4         | Jorge   | C3        | C2        | Santiago    |
| 1         | Carlos  | C1        | C3        | Jaguari     |
| 2         | Sandra  | C2        | C1        | Santa Maria |
| 3         | Mara    | C1        | C2        | Santiago    |
| 4         | Jorge   | C3        | C3        | Jaguari     |



Exemplo: Produto Cartesiano

- Problema: Selecionar o nome dos clientes e a cidade em que os mesmos residem
- Função Correspondente

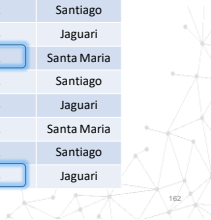
$\pi$  CLIENTE.nomeCli,  
CIDADE.nomeCid

$\sigma$  CLIENTE.codCidade =  
CIDADE.codigoCid (CLIENTE X CIDADE)



Exemplo: Produto Cartesiano

| codigoCli | nomeCli | codCidade | codigoCid | nomeCid     |
|-----------|---------|-----------|-----------|-------------|
| 1         | Carlos  | C1        | C1        | Santa Maria |
| 2         | Sandra  | C2        | C2        | Santiago    |
| 3         | Mara    | C1        | C3        | Jaguari     |
| 4         | Jorge   | C3        | C1        | Santa Maria |
| 1         | Carlos  | C1        | C2        | Santiago    |
| 2         | Sandra  | C2        | C3        | Jaguari     |
| 3         | Mara    | C1        | C1        | Santa Maria |
| 4         | Jorge   | C3        | C2        | Santiago    |
| 1         | Carlos  | C1        | C3        | Jaguari     |
| 2         | Sandra  | C2        | C1        | Santa Maria |
| 3         | Mara    | C1        | C2        | Santiago    |
| 4         | Jorge   | C3        | C3        | Jaguari     |



Exemplo: Resultado da Seleção

| nomeCli | nomeCid     |
|---------|-------------|
| Carlos  | Santa Maria |
| Sandra  | Santiago    |
| Mara    | Santa Maria |
| Jorge   | Jaguari     |

Exercícios

CONTA

| nome_agencia | numero_conta | saldo |
|--------------|--------------|-------|
| SAL-1        | 0001         | 1200  |
| SAL-1        | 0002         | 3000  |
| NOH-1        | 0003         | 4500  |
| POA-1        | 0004         | 4000  |
| POA-1        | 0005         | 1800  |
| NOH-1        | 0006         | 200   |
| SAL-2        | 0007         | 3750  |
| SAL-2        | 0008         | 1800  |

CLIENTE

| nome      | rua            | cidade        |
|-----------|----------------|---------------|
| João      | Getúlio Vargas | São Leopoldo  |
| Pedro     | Getúlio Vargas | São Leopoldo  |
| Francisco | Olavo Bilac    | Novo Hamburgo |
| Maria     | João Pessoa    | Porto Alegre  |
| Paulo     | Castro Pereira | São Leopoldo  |
| João      | João Goulart   | Novo Hamburgo |
| Ana       | Assis Brasil   | Porto Alegre  |
| Beatriz   | Florianópolis  | Novo Hamburgo |

DEPOSITANTE

| nome_cliente | numero_conta |
|--------------|--------------|
| João         | 0001         |
| Pedro        | 0002         |
| Francisco    | 0003         |
| Maria        | 0004         |
| Paulo        | 0007         |
| João         | 0006         |
| Ana          | 0005         |

DEVEDOR

| nome_cliente | num_emprestimo |
|--------------|----------------|
| João         | E-001          |
| Ana          | E-005          |
| Helena       | E-008          |

AGÊNCIA

| nome_agencia | cidade_agencia | saldo   |
|--------------|----------------|---------|
| NOH-1        | Novo Hamburgo  | 260050  |
| SAL-1        | São Leopoldo   | 455880  |
| POA-1        | Porto Alegre   | 1250369 |
| SAL-2        | São Leopoldo   | 125888  |

EMPRÉSTIMO

| nome_agencia | num_emprestimo | total |
|--------------|----------------|-------|
| SAL-1        | E-001          | 40000 |
| POA-1        | E-005          | 25400 |
| NOH-1        | E-008          | 5420  |

Exercícios

- CONTA(numeroConta, saldo, nomeAgencia)
  - nomeAgencia → AGENCIA(nomeAgencia)
- CLIENTE(nome, rua, cidade)
- DEPOSITANTE(nomeCliente, numeroConta)
  - nomeCliente → CLIENTE(nome)
  - numeroConta → CONTA(numeroConta)
- DEVEDOR(nomeCliente, numeroEmprestimo)
  - nomeCliente → CLIENTE(nome)
  - numeroEmprestimo → EMPRESTIMO(numero)
- AGENCIA(nomeAgencia, cidadeAgencia, saldo)
- EMPRESTIMO(numeroEmprestimo, total, nomeAgencia)
  - nomeAgencia → AGENCIA(nomeAgencia)

Exercícios

- Selecionar o número do empréstimo e o seu valor total
- Selecionar o nome dos Clientes e a cidade onde os mesmos residem.
- Selecionar o nome dos clientes e total dos empréstimos realizados por eles
- Selecionar o nome das Agências onde há contas com saldos maiores que 5000.
- Selecionar o depositante e o numero da conta, quando o saldo for maior que R\$ 1000,00

Renomeação

- Altera o nome de uma relação e/ou dos seus atributos
- Notação
- $\rho_{\text{nome\_relação}}(\text{relação})$
- $\rho_{(\text{nome\_atributo1}, \dots, \text{nome\_atributoN})}$

| R |   |   | $R \times \rho_{R1}(R)$ |     |     |      |      |      |
|---|---|---|-------------------------|-----|-----|------|------|------|
| x | y | z | R.x                     | R.y | R.z | RI.x | RI.y | RI.z |
| 1 | 1 | 1 | 1                       | 1   | 1   | 1    | 1    | 1    |
| 1 | 1 | 1 | 1                       | 1   | 1   | 2    | 1    | 3    |
| 2 | 1 | 1 | 2                       | 1   | 3   | 1    | 1    | 1    |
| 2 | 1 | 1 | 2                       | 1   | 3   | 2    | 1    | 3    |

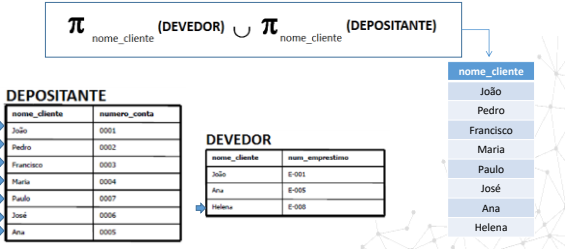
| $\rho_{(a, b, c)}(R)$ |   |   |
|-----------------------|---|---|
| a                     | b | c |
| 1                     | 1 | 1 |
| 2                     | 1 | 3 |

União:  $A \cup B$

- Produz como resultado uma tabela que contém todas as linhas da primeira tabela seguidas de todas as linhas da segunda tabela.
- A tabela resultante possui:
  - A mesma quantidade de colunas que as tabelas originais
  - Um número de linhas que é no máximo igual à soma das linhas das tabelas fornecidas como operandos

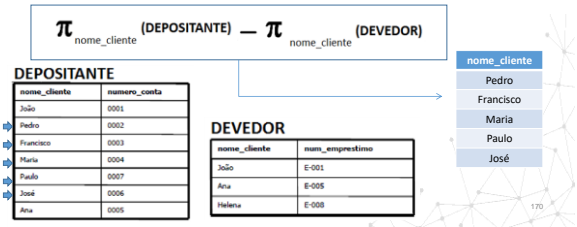
União:  $A \cup B$

- Exemplo: Selecionar todos os nomes de clientes que tenham um empréstimo, uma conta ou ambos.
  - Deve-se utilizar as relações “depositante” e “devedor”.



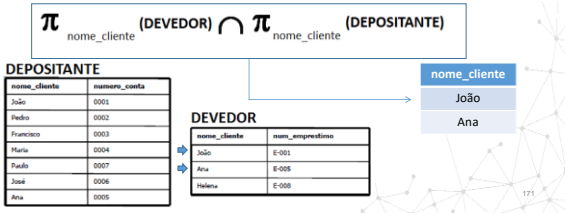
Diferença entre conjuntos:  $A - B$

- Requer como operandos duas tabelas estruturalmente idênticas.
- O resultado é uma tabela que possui todas as linhas que existem na primeira tabela e não existem na segunda.
- Exemplo: Selecionar todos os clientes que tem conta mas não tem empréstimo.



Intersecção:  $A \cap B$

- Produz como resultado uma tabela que contém, sem repetições, todos os elementos que são comuns às duas tabelas fornecidas como operandos.
- As tabelas devem ser união-compatíveis.
- Exemplo: Selecionar os nomes de clientes que possuem uma conta e um empréstimo

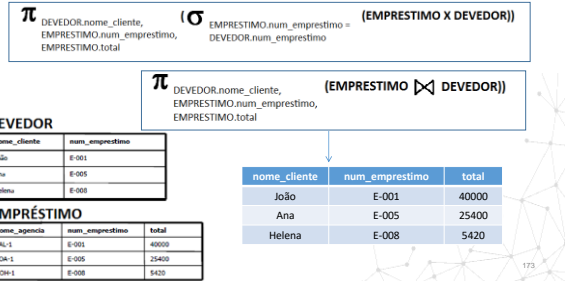


Junção Natural

- A tabela resultante de uma junção tem todas as colunas da primeira tabela e todas da segunda tabela.
- Valores dos campos utilizados como critério para a correspondência entre as linhas apareça duplicado
  - Já que um vem da primeira tabela e outro da segunda.
- Junção natural
  - Fornece o mesmo resultado, mas sem essa repetição de valores.
  - Símbolo:  $\bowtie$

Junção Natural

- Exemplo: Encontrar todos os nomes de clientes que tenham um empréstimo no banco, bem como o total emprestado.



Exercício

- CARGO(codigo, nome, salario)
- DEPARTAMENTO(codigo, nome, ramal)
- FUNCIONARIO(matricula, nome, dataAdmissao, sexo, codCargo, codDepartamento)
  - codCargo  $\rightarrow$  CARGO(codigo)
  - codDepartamento  $\rightarrow$  DEPARTAMENTO(codigo)

Exemplo

| codigoCargo | nomeCargo          | valorSalario |
|-------------|--------------------|--------------|
| C1          | Cozinheira         | 500          |
| C3          | Aux. de Escritório | 550          |
| C7          | Vigia              | 500          |
| C2          | Mecânico           | 560          |
| C5          | Gerente            | 1500         |
| C4          | Escriturário       | 700          |

| codigoDepto | nomeDepto       | ramal |
|-------------|-----------------|-------|
| D1          | Administração   | 221   |
| D2          | Oficina         | 235   |
| D3          | Serviços Gerais | 243   |
| D4          | Vendas          | 258   |

Exemplo

| numMatric | nomeFunc        | dataAdm  | sexo | codCargo | codDepto |
|-----------|-----------------|----------|------|----------|----------|
| 1001      | João Sampaio    | 10/08/93 | M    | C2       | D2       |
| 1004      | Lúcio Torres    | 02/03/94 | M    | C2       | D2       |
| 1034      | Roberto Pereira | 23/05/92 | M    | C3       | D1       |
| 1021      | José Nogueira   | 10/11/94 | M    | C3       | D1       |
| 1029      | Ruth de Souza   | 05/01/92 | F    | C1       | D3       |
| 1095      | Maria da Silva  | 03/09/92 | F    | C4       | D1       |
| 1023      | Luiz de Almeida | 12/01/93 | M    | C2       | D2       |
| 1042      | Pedro Pinheiro  | 29/07/94 | M    | C4       | D1       |
| 1048      | Ana Silveira    | 01/06/93 | F    | C5       | D1       |
| 1015      | Pedro Rodrigues | 17/08/92 | M    | C2       | D2       |

Exercício

- Selecionar:
  - Todos os funcionários do departamento 'D1'.
  - O nome e a matrícula de todos os funcionários do departamento 'D1'.
  - A matrícula, o nome do funcionário, e o nome do respectivo departamento destes funcionários.
  - O nome dos funcionários que ganham mais de \$500.
  - O ramal do funcionário 'ANA SILVEIRA'.
  - Os nomes de todos os funcionários com cargo de 'MECANICO'.
  - Os nomes de todos os funcionários que trabalham no mesmo departamento que 'JOSE NOGUEIRA'.
  - Os nomes dos departamentos que possuem tanto funcionários como funcionárias.

Exemplo – Modelo Relacional

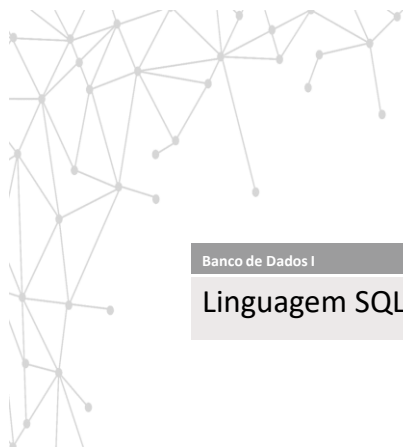
- Ambulatórios(nroa, andar, capacidade)
- Médicos(codm, CPF, nome, idade, cidade, especialidade, *nroa*)
  - nroa referencia Ambulatórios(nroa)
- Pacientes(codp, CPF, nome, idade, cidade, doença)
- Consultas(codm, codp, data, hora)
  - codm referencia Médicos(codm)
  - codp referencia Pacientes(codp)
- Funcionários(codf, CPF, nome, idade, cidade, salário)

Atribuição ←

- Armazena o resultado de uma expressão algébrica em uma variável de relação.
  - permite o processamento de uma consulta por etapas.
- Notação:
  - nomeVariável* ← *expressãoÁlgebra*
- Exemplo
  - $R1 \leftarrow \pi_{\text{codm, data}}(\text{Consultas})$
  - $R2 \leftarrow \pi_{\text{codm, nome}}(\text{Médicos})$
  - Resposta ←  $\pi_{\text{nome, data}}(\sigma_{R1.\text{codm} = R2.\text{codm}}(R1 \times R2))$

Junções Externas (outer joins)

- Junção na qual as tuplas de uma ou ambas as relações que não são combinadas são mesmo assim preservadas no resultado
- Três tipos (exemplos com junção natural)
  - junção externa à esquerda (*left [outer] join*)
    - tuplas da relação à esquerda são preservadas
    - notação: relação1 ⋈ relação2
  - junção externa à direita (*right [outer] join*)
    - tuplas da relação à direita são preservadas
    - notação: relação1 ⋈ relação2
  - junção externa completa (*full [outer] join*)
    - tuplas de ambas as relações são preservadas
    - notação: relação1 ⋈ relação2

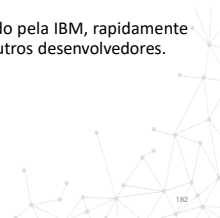


Banco de Dados I

## Linguagem SQL

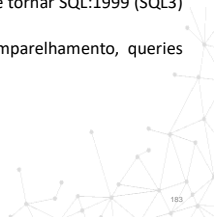
### Introdução

- *Structured Query Language*, ou Linguagem de Consulta Estruturada.
- É uma linguagem de pesquisa declarativa para banco de dados relacional (bases de dados relacionais).
- Muitas das características originais do SQL foram inspiradas na álgebra relacional.
- Embora o SQL tenha sido originalmente criado pela IBM, rapidamente surgiram vários dialetos desenvolvidos por outros desenvolvedores.



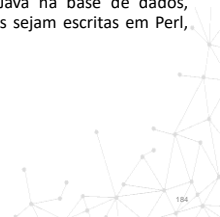
### Introdução

- Essa expansão levou à necessidade de ser criado e adaptado um padrão para a linguagem. Esta tarefa foi realizada pela *American National Standards Institute* (ANSI) em 1986 e ISO em 1987.
- O SQL foi revisto em 1992 e a esta versão foi dado o nome de SQL92.
- Foi revisto novamente em 1999 e 2003 para se tornar SQL:1999 (SQL3) e SQL:2003, respectivamente.
- O SQL:1999 usa expressões regulares de emparelhamento, queries recursivas e gatilhos (triggers).



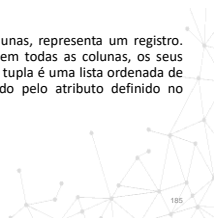
### Introdução

- Embora padronizado pela ANSI e ISO, possui muitas variações e extensões produzidos pelos diferentes fabricantes de SGBDs.
- Outra aproximação é permitir para código de idioma processual ser embutido e interagir com o banco de dados.
- Por exemplo, o Oracle e outros incluem Java na base de dados, enquanto o PostgreSQL permite que funções sejam escritas em Perl, Tcl, ou C, entre outras linguagens.



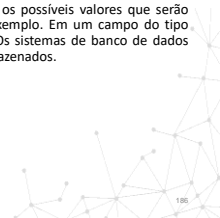
### Revisão

- **Tabelas:**
  - Todos os dados de um banco de dados relacional (BDR) são armazenados em tabelas. Uma tabela é uma simples estrutura de linhas e colunas. Cada linha contém um mesmo conjunto de colunas mas as linhas não seguem qualquer tipo de ordem. Em um BD podem existir uma ou centenas de tabelas. O limitador é imposto exclusivamente pela ferramenta de software utilizada.
- **Registros (ou tuplas):**
  - Cada linha, formada por uma lista ordenada de colunas, representa um registro. Estes não precisam necessariamente conter dados em todas as colunas, os seus valores podem ser nulos. Formalmente falando, uma tupla é uma lista ordenada de valores, onde cada valor é do domínio especificado pelo atributo definido no esquema de relação.

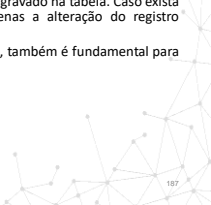


### Revisão (cont.)

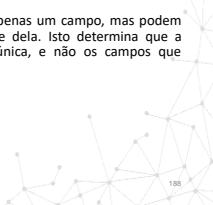
- **Colunas (Atributos ou Campos):**
  - As colunas de uma tabela, são também chamadas de Atributos ou Campos. Cada atributo pertence a um domínio (tipo de campo), que define os valores que podem ser associados aquele atributo.
- **Domínio (Tipos de Dados):**
  - Os domínios possuem características que definem os possíveis valores que serão armazenado em um atributo de uma tupla. Por exemplo. Em um campo do tipo numérico, serão somente armazenados números. Os sistemas de banco de dados possuem regras para consistir os dados que são armazenados.



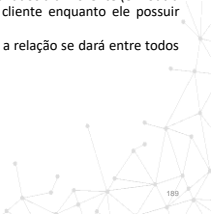
Revisão (cont.)

- **Chave (Key):**
    - As tabelas relacionam-se umas as outras através de chaves.
    - Uma chave é um conjunto de um ou mais campos que determinam a unicidade de cada registro.
    - Por exemplo, se um banco (base) de dados tem como chave Código do Produto + ID Sistema, sempre que acontecer uma inserção de dados, o SGBD irá fazer uma consulta para identificar se o registro não se encontra gravado na tabela. Caso exista um novo registro não será criado, sendo que apenas a alteração do registro existente será possível.
    - A unicidade dos registros, determinada por sua chave, também é fundamental para a criação dos índices.
- 


Revisão (cont.)

- **Chave (Key)** (cont.):
    - Temos dois tipos de chaves:
      - Chave Primária: (PK - *Primary Key*) é a chave que identifica cada registro dando-lhe unicidade. A chave primária nunca se repetirá. A maioria dos bancos também exige que ela seja NÃO NULA (*NOT NULL*) além de única.
      - Chave Estrangeira: (FK - *Foreign Key*) é uma chave formada pela chave primária de outra tabela e a chave de um campo da tabela que recebe o relacionamento. Define um relacionamento entre as tabelas e pode ocorrer repetidas vezes.
    - **Observação:** Geralmente Chaves Primárias são de apenas um campo, mas podem ser compostas, ou seja, vários campos fazem parte dela. Isto determina que a combinação entre os campos é que precisa ser única, e não os campos que isoladamente a compõe.
- 

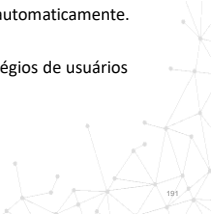
Revisão (cont.)

- **Relacionamentos:**
    - Como o próprio nome sugere, um BDR possui diversos relacionamentos entre as tabelas existentes.
    - Um relacionamento é feito ligando-se um campo de uma tabela X com um campo de uma tabela Y ou tabela estrangeira.
    - Por exemplo, se pedidos (em uma tabela) estão relacionados a um cliente (em outra tabela), o SGBD poderá bloquear a remoção deste cliente enquanto ele possuir pedidos registrados.
    - **Observação:** no caso de chaves primárias compostas, a relação se dará entre todos os campos desta chave.
- 

Revisão (cont.)

- **Relacionamentos** (cont.):
    - Existem alguns tipos de relacionamentos possíveis:
      - Um para um (1 para 1)
      - Um para muitos (1 para N)
      - Muitos para muitos (N para N), que não pode ser implementado diretamente no modelo relacional e tem que ser construído com uma tabela auxiliar (associativa).
- 

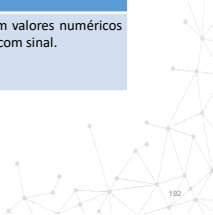
Linguagem SQL – Funcionalidades

- Existem algumas sub denominações, mas o mais comum é definir em dois grandes grupos:
    - DDL (*Data Definition Language*) Linguagem de Definição de Dados.
    - DML (*Data Manipulation Language*) Linguagem de Manipulação de Dados.
  - Definição de visões (*views*).
  - Uso de gatilhos (*triggers*) que são disparados automaticamente.
  - Definição de tipos de usuários (*roles*).
  - Atribuição (*grant*) e remoção (*revoke*) de privilégios de usuários
- 

Tipos de Dados

- Variam para cada SGBD.
- No Firebird, são mais usados:
  - Inteiros:

| Tipo     | Particularidades | Descrição                                      |
|----------|------------------|--|
| integer  | 32 bits          | Suportam valores numéricos inteiros com sinal. |
| smallint | 16 bits          |  |
| bigint   | 64 bits          |  |





Tipos de Dados (cont.)

| Tipo       | Particularidades  | Descrição  |
|------------|---|--|
| char(T)    | retorna os valores com espaço a direita, preenchendo o tamanho T especificado.  | São tipos utilizados para armazenar letras, números, caracteres, etc.. |
| Varchar(T) | Retorna da mesma forma em que foram armazenados, ou seja, se o tamanho T não for preenchido, não são adicionados espaços a direita. |  |

Tipos de Dados (cont.)

• Decimais:

| Tipo          | Particularidades  | Descrição  |
|---------------|---|--|
| numeric (T,P) | É necessário especificar o tamanho máximo T (incluindo números após o ponto) e a precisão P (quantas casas decimais). | São utilizados para armazenar valores do tipo fracionário, muito importante quando é necessário precisão no retorno dos dados. |
| decimal (T,P) |   |  |

Tipos de Dados (cont.)

• Ponto Flutuante:

| Tipo   | Descrição   |
|--------|---|
| float  | Guarda números de ponto flutuante, porém na maioria dos casos deve ser evitado, pois é possível que a informação armazenada não seja exatamente a mesma quando recuperada.                              |
| double | Como o float, também armazena valor em ponto flutuante, porém com uma faixa maior de valores, guardando em 64 bits, contudo possui o mesmo problema do float, o da precisão na recuperação dos valores. |

Tipos de Dados (cont.)

• Temporais:

| Tipo      | Particularidades       |
|-----------|------------------------|
| date      | Somente datas.         |
| time      | Somente horas (tempo). |
| timestamp | Data + hora.           |

Tipos de Dados (cont.)

• Binários:

| Tipo | Particularidades  | Descrição  |
|------|---|--|
| blob | Dois sub tipos mais usados:<br>0-Dados binários<br>1-Informação textual | É utilizado para armazenar desde textos a arquivos binários. Sua capacidade de armazenamento é ilimitada, sendo limitada somente por falta de espaço em disco. |

DDL

- Os comandos DDL são usados para definir a estrutura do banco de dados, organizando em tabelas que são compostas por campos (colunas).
- Comandos que compõem a DDL:
  - CREATE
  - ALTER
  - DROP

## DDL - Create

- Exemplo de criação de tabela:

```
CREATE TABLE clientes(
  cod_cli integer not null primary key,
  cpf varchar(11),
  nome varchar(100),
  data_nascimento date,
  sexo char(1),
  salario numeric(15,2),
  cod_prof integer
);
```

```
CREATE TABLE profissoes(
  cod_prof integer not null primary key,
  nome varchar(100)
);
```

## DDL - Alter

- Criação de *constraints* (chave primária ou estrangeira).

- Chave Primária:
  - Sintaxe:

```
ALTER TABLE nome_tabela ADD CONSTRAINT PRIMARY KEY
(nome_campo_not_null );
```

```
ALTER TABLE clientes add constraint PRIMARY KEY (cod_cli);
```

## DDL - Alter

- Chave Estrangeira:
  - Sintaxe:

```
ALTER TABLE nome_tabela ADD CONSTRAINT TIPO_CONSTRAINT
(nome_campo_a_ser_referenciado)
REFERENCES nome_tabela_estrangeira(campo_chave_primaria) ON UPDATE
regra ON DELETE regra;
```

- Exemplo:

```
ALTER TABLE clientes ADD CONSTRAINT FOREIGN KEY (cod_prof)
REFERENCES profissoes(cod_prof)
ON UPDATE cascade ON DELETE set null;
```

## DDL – Regras (Rules) para Chaves Estrangeiras

- As regras de integridade são acionadas automaticamente em 2 eventos:

- ON UPDATE:** Dispara sempre que houver modificações na chave primária referenciada, atuando na estrangeira correspondente, executando uma regra definida.
- ON DELETE:** Dispara sempre que houver exclusões na tabela dependente, executando uma regra definida.

- Regras mais utilizadas:

- SET NULL:** Atribui nulo na chave estrangeira.
- CASCADE:** Exclui todos os registros que possuem como chave estrangeira o mesmo valor da chave primária da tabela referenciada.

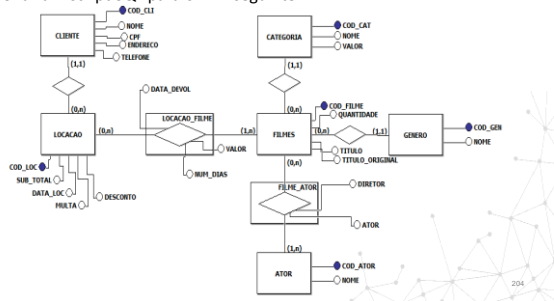
## Criação de Tabela sem o uso do comando ALTER

- Certifique-se de ter criado todas as estruturas que serão referenciadas.

```
CREATE TABLE clientes(
  cod_cli integer not null primary key,
  cpf varchar(11),
  nome varchar(100),
  data_nascimento date,
  sexo char(1),
  salario numeric(15,2),
  cod_prof integer,
  foreign key (cod_prof) references profissao (cod_prof)
  on update cascade on delete set null
);
```

## Exercícios 1

- Criar um script SQL para o DER seguinte:



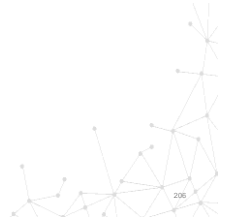
Auto Incrementando campos do tipo Chaves Primárias (PKs)

- Usando o SGBD Firebird, deve-se criar duas estruturas:
  - **Generator:** estrutura que armazena o último valor usado. Sempre retorna um valor inteiro.
  - **Trigger:** evento disparado antes (*before*) ou depois (*after*) de alguma ação (*insert*, *update* ou *delete*).



Auto Incrementando campos do tipo Chaves Primárias (PKs)

- Sintaxes:
  - **Generator:**
    - `create generator <nome_do_generator>;`
  - **Trigger:**
    - `create trigger <nome_do_trigger> for <tabela>`
    - `active <evento> <ação> position <prioridade>`
    - `as`
    - `begin`
      - `/* Código */`
    - `end`



Auto Incrementando campos do tipo Chaves Primárias (PKs)

- Exemplos:
  - **Generator:**
    - `create generator g_inc_cliente;`
  - **Trigger:**
    - `create trigger t_inc_cliente for clientes`
    - `active before insert position 5`
    - `as`
    - `begin`
      - `new.COD_CLI=gen_id(g_inc_cliente,1);`
    - `end;`



DDL - Alter

- Alterando Tabelas:
  - Sintaxe:

```
ALTER TABLE nome_tabela
  ADD nome_coluna1 tipo_dado,
  ADD nome_coluna2 tipo_dado,
  ADD nome_coluna3 tipo_dado
;
```
- Exemplo:

```
ALTER TABLE servicos
  ADD valor numeric(15,2),
  ADD descricao varchar(30)
;
```



DDL - Alter

- Alterando Tabelas:
  - Sintaxe:

```
ALTER TABLE nome_tabela
  ADD nome_coluna1 tipo_dado,
  ADD nome_coluna2 tipo_dado,
  ADD nome_coluna3 tipo_dado
;
```
- Exemplo:

```
ALTER TABLE servicos
  ADD valor numeric(15,2),
  ADD descricao varchar(30)
;
```



DDL - Alter

- Removendo colunas:
  - Sintaxe:

```
ALTER TABLE nome_tabela
  DROP COLUMN nome_coluna1,
  DROP COLUMN nome_coluna2;

```
- Exemplo:

```
ALTER TABLE Pacientes
  DROP COLUMN doenca,
  DROP COLUMN cidade;

```



## DML

- Define operações de manipulação de dados:
  - INSERT
  - UPDATE
  - DELETE
  - SELECT
  - EXECUTE
- Instruções declarativas manipulação de conjuntos especifica-se o que fazer e não como fazer.

## DML - Insert

- Sintaxes:

```
INSERT INTO nome_tabela
VALUES (TODA_lista_valores_atributos_ordenados);
```

```
INSERT INTO nome_tabela (lista_atributos_especificos)
VALUES (lista_valores_atributos);
```

- Exemplos:

```
INSERT INTO Ambulatorios VALUES (1, 1, 30);
```

```
INSERT INTO Medicos (codm, nome, idade, especialidade, CPF, cidade)
VALUES (4, 'Carlos', 28, 'ortopedia', '11000110000', 'Joinville');
```

## DML - Condicional

- Pode-se restringir o resultado das consultas, utilizando a cláusula WHERE.
- Operadores relacionais:
  - Maior: >
  - Menor: <
  - Maior ou igual: >=
  - Menor ou igual: <=
  - Diferente: <>
  - Não nulo: *is not null*
  - Aproximado: *like* (usa-se '%' como coringa)
    - Começando com: Ex.: nome\_campo like 'a%'
    - Contendo: Ex.: nome\_campo like '%a%'
    - Terminando com: Ex.: nome\_campo like '%a'

## DML - Condicional

- Operadores lógicos:
  - E: *AND*
  - Ou: *OR*
- Apelidos (*Aliases*):
  - É um boa prática usar apelidos em tabelas, a fim de diminuir o tamanho do código.
  - Normalmente usa-se a 1 letra inicial (ou mais) para representar.
  - CUIDADO para não criar apelidos iguais.
  - Exemplo:

```
UPDATE Medicos m
SET m.cidade = 'Jaguari';
```

## DML - UPDATE

- Usado para alterar valores de campos em tabelas.

- Sintaxe:

```
UPDATE nome_tabela
SET nome_atributo_1 = Valor,
    nome_atributo_2 = Valor
WHERE condição;
```

- Exemplos:

```
UPDATE Medicos
SET cidade = 'Jaguari';
```

```
UPDATE Ambulatorios
SET capacidade = capacidade + 5,
    andar = 3
WHERE nroa = 2;
```

## DML - DELETE

- Usado para excluir linhas de tabelas.

- Sintaxe:

```
DELETE FROM nome_tabela
WHERE condição;
```

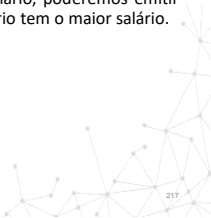
- Exemplos:

```
DELETE FROM Ambulatorios;
```

```
DELETE FROM Medicos
WHERE especialidade = 'cardiologia'
OR cidade < > 'Jaguari';
```

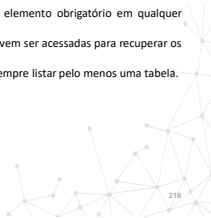
DML - SELECT

- Uma consulta é uma recuperação de informações contidas no banco de dados feita através da instrução SELECT.
- Uma consulta é usada para extrair dados do banco de dados em um formato legível de acordo com a solicitação do usuário.
- Por exemplo, se tivermos uma tabela funcionário, poderemos emitir uma instrução SQL que informe qual funcionário tem o maior salário.



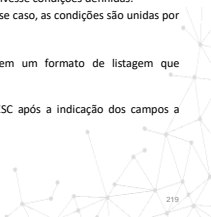
DML - SELECT

- Existem quatro palavras-chaves, ou cláusulas, que são partes importantes de uma instrução SELECT:
- SELECT:
  - Tem por finalidade selecionar os dados que deseja-se visualizar de acordo com as colunas em que eles estão armazenados em uma tabela.
- FROM:
  - Deve ser usada junto com a instrução SELECT, sendo um elemento obrigatório em qualquer consulta.
  - Sua finalidade é informar ao banco de dados que tabelas devem ser acessadas para recuperar os dados desejados da consulta.
  - A cláusula FROM pode conter uma ou mais tabelas e deve sempre listar pelo menos uma tabela.



DML - SELECT

- WHERE:
  - Uma condição é parte de uma consulta que é usada para exibir informações seletivas, conforme especificadas pelo usuário.
  - O valor de uma condição pode ser TRUE ou FALSE, limitando assim, os dados recebidos da consulta.
  - A cláusula WHERE é usada para impor condições a uma consulta, eliminando linhas que normalmente seriam retornadas por uma consulta que não tivesse condições definidas.
  - A cláusula WHERE pode conter mais de uma condição e, nesse caso, as condições são unidas por operadores AND e OR.
- ORDER BY:
  - Esta cláusula organiza os resultados de uma consulta em um formato de listagem que especificamos.
  - A ordenação padrão para a cláusula ORDER BY é a crescente.
  - Para ordenar de forma decrescente usa-se o comando DESC após a indicação dos campos a serem ordenados.

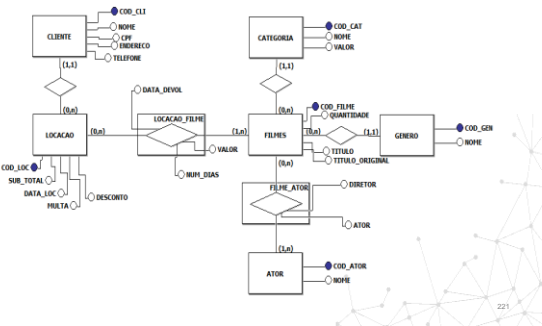


DML - Exemplo

```
SELECT f.nm_funcionario, f.vl_salario
FROM funcionario f
WHERE f.vl_salario > 1200.00
ORDER BY f.nm_funcionario
```



Exercícios 2



Exercícios 2

- Criar as tabelas, campos e restrições;
- Usar os comandos DDL e DML aprendidos.
- Elaborar consultas.



## SELECT

- A estrutura básica da instrução de consulta select consiste em três cláusulas:

- **SELECT**
- **FROM**
- **WHERE**

```
SELECT <apelido.coluna>
FROM <tabelas> <apelido>
WHERE <condições>
```

## SELECT

### • Cláusula SELECT

- Usada para listar os atributos desejados no resultado da consulta, ou seja, as colunas requisitadas como respostas.
- Esta lista, além de nomes de colunas, pode conter:
  - Asterisco (\*), indicando que todas as colunas devem ser retornadas;
  - Expressões;
  - Constantes;
  - Funções;
  - Qualquer combinação destes, conectadas por operadores aritméticos e parênteses.

## SELECT

### • Cláusula FROM

- Lista as tabelas a serem examinadas na avaliação da expressão.
- As colunas indicadas na cláusula SELECT devem constar nas tabelas listadas nesta cláusula.

### • Cláusula WHERE

- Consiste de uma condição de procura envolvendo atributos das tabelas que aparecem na cláusula FROM.
- É o lugar onde se restringe ou seleciona quais linhas deverão ser retornadas.

## SELECT

### • A cláusula WHERE pode ser omitida

- Neste caso, o predicado é tido como sempre verdadeiro e todas as linhas serão retornadas.

### • Exemplo:

- Quais os códigos de todos os funcionários que fizeram alguma requisição ao estoque?
- Instrução correspondente em SQL:
  - `SELECT r.codFunc FROM requisicao r;`

## Eliminação de linhas duplicadas

- A eliminação de linhas repetidas ou duplicadas é possível utilizando a palavra-chave **DISTINCT** depois de select.
- Então pode-se reescrever a consulta anterior da seguinte maneira:
  - `SELECT DISTINCT r.codFunc FROM requisicao r;`

## Constantes e Expressões

- A lista da cláusula **SELECT** também pode conter constantes e expressões.

### • Exemplo utilizando constante:

- `SELECT r.codigo, 'DATADA DE', r.dataReq FROM requisicao r;`

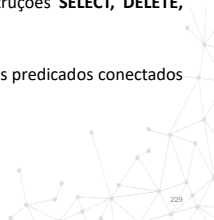
### • Exemplo utilizando expressão:

- `SELECT p.descricao, p.quantidade, (p.quantidade/2) OPERACAO from produto p;`

Uma expressão pode conter uma combinação de parênteses e das quatro operações aritméticas básicas: adição (+), subtração (-), multiplicação (\*) e divisão (/)

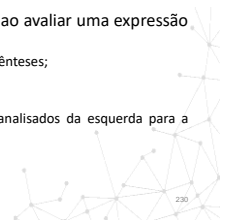
## Condições de Procura

- Uma condição de procura na cláusula **WHERE** qualifica o escopo de um consulta, pois especifica as condições particulares que devem ser encontradas.
- A cláusula **WHERE** pode ser usada nas instruções **SELECT**, **DELETE**, **UPDATE**.
- Cada condição de procura contém um ou mais predicados conectados por operadores lógicos **OR**, **AND**, e **NOT**.



## Condições de Procura

- Exemplo:
  - **SELECT** p.descricao, p.quantidade
  - **FROM** produto p
  - **WHERE** p.quantidade > 100 **AND** p.quantidade < 400;
- A seguinte ordem de precedência é utilizada ao avaliar uma expressão numa condição:
  - Primeiro são analisados as expressões dentro de parênteses;
  - O operador **NOT** é analisado antes do **AND**;
  - O operador **AND** é analisado antes do **OR**;
  - Operadores do mesmo nível de precedência são analisados da esquerda para a direita.



## Predicado Between

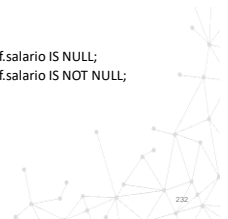
- Compara um valor com uma faixa de valores.
- Os limites da faixa estão inclusos.
- Exemplo:
  - **SELECT** f.nome, f.salario
  - **FROM** funcionario f
  - **WHERE** f.salario **BETWEEN** 1000 **AND** 3000;



## Predicado NULL

- O predicado **NULL** testa valores nulos.
- Verifica, por exemplo, se colunas não contém nenhum valor armazenado.
- Exemplo:
  - **SELECT** f.nome, f.salario **FROM** funcionario f **WHERE** f.salario **IS NULL**;
  - **SELECT** f.nome, f.salario **FROM** funcionario f **WHERE** f.salario **IS NOT NULL**;

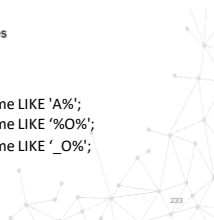
É possível utilizar o operador **NOT** juntamente com a palavra-chave **IS** (is not **NULL**) para, por exemplo, montar condições de colunas que não possuem valores nulos.



## Predicado LIKE

- O predicado **LIKE** procura por strings que se encontram dentro de um determinado padrão.
- O predicado **LIKE** só pode ser usado com tipos de dados **CHAR** e **VARCHAR**.
 

|   |                                    |
|---|------------------------------------|
| % | Equivale a zero ou mais caracteres |
| _ | Equivale a um caracter qualquer    |
- Exemplo:
  - **SELECT** f.nome **FROM** funcionario f **WHERE** f.nome **LIKE** 'A%';
  - **SELECT** f.nome **FROM** funcionario f **WHERE** f.nome **LIKE** '%O%';
  - **SELECT** f.nome **FROM** funcionario f **WHERE** f.nome **LIKE** '\_O%';



## Predicado Exists

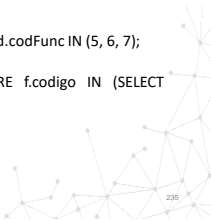
- É utilizado para testar se o conjunto de linhas resultantes de uma consulta é vazio.
- Exemplo:
 

```
SELECT f.nome
FROM funcionario f
WHERE EXISTS
(SELECT d.*
FROM devolucao d
WHERE f.codigo = d.codFunc);
```



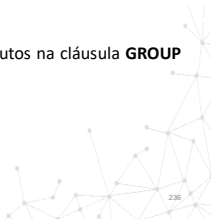
Predicado IN

- O predicado **IN** compara um valor com um conjunto de valores.
- Este conjunto de valores pode ser uma lista ou o resultado de um subselect.
- Exemplo:
  - `SELECT d.dataDev FROM devolucao d WHERE d.codFunc IN (5, 6, 7);`
  - `SELECT f.nome FROM funcionario f WHERE f.codigo IN (SELECT d.codFunc from devolucao d);`



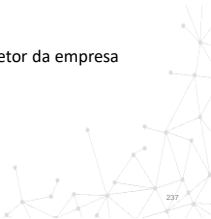
Funções de agregação

- SQL oferece a possibilidade de computar funções em grupos de linhas usando a cláusula **GROUP BY**
- Os atributos, dados na cláusula **GROUP BY**, são usados para formar grupos.
- Linhas com o mesmo valor em todos os atributos na cláusula **GROUP BY** são colocados em um grupo.



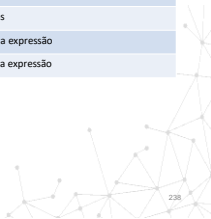
Funções de agregação

- SQL inclui funções de agregação para computar:
  - Média: **AVG**
  - Mínimo: **MIN**
  - Máximo: **MAX**
  - Total: **SUM**
  - Contar: **COUNT**
- Exemplo: Selecione o salário médio em cada setor da empresa
  - `SELECT f.codSetor, AVG (f.salario)`
  - `FROM funcionario f`
  - `GROUP BY f.codSetor;`



Funções de agregação

| Função de Agregação                | Resultado   |
|------------------------------------|---|
| SUM ( [ DISTINCT   expressão ] )   | Soma de valores DISTINTOS na expressão numérica   |
| AVG ( [ DISTINCT   expressão ] )   | Média de valores DISTINTOS na expressão numérica  |
| COUNT ( [ DISTINCT   expressão ] ) | Número de valores DISTINTOS na expressão numérica |
| COUNT(*)                           | Número de linhas selecionadas                     |
| MAX (expressão)                    | Maior valor computado para a expressão            |
| MIN (expressão)                    | Menor valor computado para a expressão            |



Cláusula HAVING

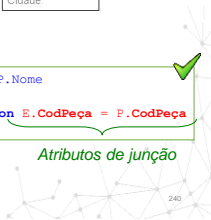
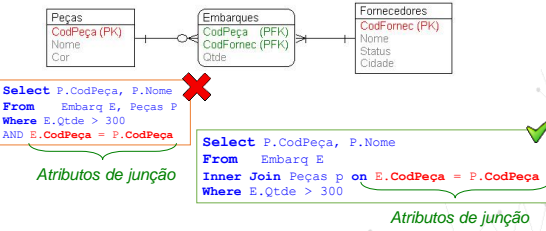
- Em certas situações, é necessário definir uma condição que se aplique a grupos, ao invés de linhas.
- Exemplo: Selecionar o salário médio apenas daqueles setores cujo salário médio seja maior que 2000

```
SELECT f.codSetor, AVG (f.salario) FROM funcionario f
GROUP BY f.codSetor HAVING AVG (f.salario) > 2000;
```



Junções

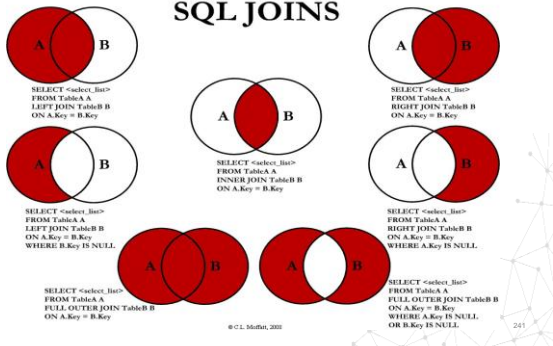
- Para cada embarque com quantidade maior que 300, obter o código da peça e o nome da peça





## Junções

### SQL JOINS



## Exemplo

| Peça    |           |         |          |            | Embarq  |           |            |
|---------|-----------|---------|----------|------------|---------|-----------|------------|
| CodPeça | NomePeça  | CorPeça | PesoPeça | CidadePeça | CodPeça | CodFornec | QtdeEmbarc |
| P1      | Elxo      | Cinza   | 10       | PoA        | P1      | F1        | 300        |
| P2      | Rolamento | Preto   | 16       | Rio        | P1      | F3        | 200        |
| P3      | Mancal    | Verde   | 30       | São Paulo  | P2      | F1        | 300        |
|         |           |         |          |            | P2      | F4        | 350        |

| Fornec    |            |              |              | Embarq  |           |            |  |
|-----------|------------|--------------|--------------|---------|-----------|------------|--|
| CodFornec | NomeFornec | StatusFornec | CidadeFornec | CodPeça | CodFornec | QtdeEmbarc |  |
| F1        | Silva      | 5            | São Paulo    | P1      | F1        | 300        |  |
| F2        | Souza      | 10           | Rio          | P1      | F2        | 400        |  |
| F3        | Alvares    | 5            | São Paulo    | P1      | F3        | 200        |  |
| F4        | Tavares    | 8            | Rio          | P2      | F1        | 300        |  |
|           |            |              |              | P2      | F4        | 350        |  |

```

Select *
From Embarq E, Fornec F
Where E.CodFornec = F.CodFornec

```

## Junções

- Obter os nomes dos fornecedores das peças de cor vermelha

| Peça    |           |         |            | Embarq  |           |            |
|---------|-----------|---------|------------|---------|-----------|------------|
| CodPeça | NomePeça  | CorPeça | CidadePeça | CodPeça | CodFornec | QtdeEmbarc |
| P1      | Elxo      | Cinza   | PoA        | P1      | F1        | 300        |
| P2      | Rolamento | Preto   | Rio        | P1      | F3        | 200        |
| P3      | Mancal    | Verde   | São Paulo  | P2      | F1        | 300        |
|         |           |         |            | P2      | F4        | 350        |

| Fornec    |            |              |              | Embarq  |           |            |  |
|-----------|------------|--------------|--------------|---------|-----------|------------|--|
| CodFornec | NomeFornec | StatusFornec | CidadeFornec | CodPeça | CodFornec | QtdeEmbarc |  |
| F1        | Silva      | 5            | São Paulo    | P1      | F1        | 300        |  |
| F2        | Souza      | 10           | Rio          | P1      | F2        | 400        |  |
| F3        | Alvares    | 5            | São Paulo    | P1      | F3        | 200        |  |
| F4        | Tavares    | 8            | Rio          | P2      | F1        | 300        |  |
|           |            |              |              | P2      | F4        | 350        |  |

```

Select NomeFornec
From Fornec F, Embarq E, Peça P
Where CorPeça = 'Verm'
And E.CodPeça = P.CodPeça
And F.CodFornec = E.CodFornec

```

❌

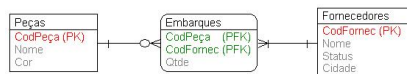
```

Select F.NomeFornec
From Fornec F
Inner Join Embarq E on E.CodFornec = F.CodFornec
Inner Join Peça P on E.CodPeça = P.CodPeça
Where P.CorPeça = 'Verm'

```

✅

## Junções na cláusula FROM



- Obter o código e o nome das peças embarcadas

```

Select E.CodPeça, P.NomePeça
From Embarques E
Inner Join Peças P on E.CodPeça = P.CodPeça

```

## Exercício

```

Ambulatorios(numero, andar, capacidade)
Medicos(CRM, nome, idade, especialidade, RG, cidade, numero)
  numero referencia ambulatorios
Pacientes(CodP, nome, idade, cidade, RG, diagnostico)
Consultas(CRM, CodP, data, hora)
  CRM referencia Medicos
  CodP referencia Pacientes
Funcionarios(RG, nome, idade, cidade, salario)

```

Obter o nome dos médicos com consultas marcadas para '13-10-2015'

## Junções Externas

- Junção na qual as linhas de uma ou ambas as tabelas que não são combinadas são mesmo assim preservadas no resultado
- Três Tipos
  - Junção externa à esquerda (left [outer] join)
    - Linhas da tabela à esquerda são preservadas
  - Junção externa à direita (right [outer] join)
    - Linhas da tabela à direita são preservadas
  - Junção externa completa (full [outer] join)
    - Linhas de ambas tabelas são preservadas

## Junções Externas

```
select lista_atributos
from tabela1
(left|right|full) join tabela2 on condição_junção
[where condição]
```



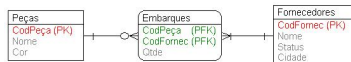
- Obter o nome de todos Fornecedores e, para aqueles que têm peças embarcadas, mostrar o código da peça embarcada

```
Select f.Nome, e.CodPeça
From Fornecedor f
left join Embarques e on f.CodFornec = e.CodFornec
```

## Junções Externas

- Buscar o nome de todos os médicos e, para aqueles que têm consultas marcadas, mostrar os dados de suas consultas
- Buscar o nome de todos os médicos e, para aqueles que têm consultas marcadas, mostrar os nomes dos pacientes da consulta marcada e o horário da consulta
- Buscar os números de todos os ambulatórios e, para aqueles ambulatórios nos quais médicos dão atendimento, exibir o CRM e o nome dos médicos associados

## Subconsultas ou Consultas Aninhadas



Obter os códigos dos fornecedores que tem embarques de peças de cor vermelha

```
SELECT CodFornec
FROM Embarques E, Peças P
WHERE Cor='Verm' AND E.CodPeça=P.CodPeça
```

Neste caso o resultado da consulta envolve apenas colunas da tabela Embarq, mas a cláusula FROM referencia também a tabela Peça

```
SELECT CodFornec
FROM Embarques
WHERE CodPeça IN
( SELECT CodPeça
FROM Peças
WHERE Cor= 'Verm' )
```

## Subconsultas ou Consultas Aninhadas

- Filtragens prévias de dados na subconsulta
  - Apenas linhas/colunas de interesse são combinados com dados da(s) tabela(s) da consulta externa

## Subconsultas – cláusula exist



Obter os nomes dos fornecedores para os quais não há embarques

```
SELECT F.Nome
FROM Fornecedor F
WHERE not exists
(SELECT *
FROM Embarques E
WHERE E.CodFornec = F.CodFornec)
```

## Modelo Estendido

- Modelo Mental de execução
  - É feito o *produto cartesiano* das tabelas envolvidas
  - São *selecionadas* as linhas da tabela que obedecem ao critério da cláusula WHERE
  - São *criados grupos* de linhas que contenham valores idênticos nas colunas GROUP BY
  - São *selecionados os grupos* que obedecem ao critério da cláusula HAVING
  - É feita a *classificação* do resultado pelos valores das colunas da cláusula ORDER BY
  - É feita a *projeção* sobre as colunas que vão ao resultado

Definição de Grupos

```
select bairro, count(*)
from MEDICOS
group by bairro
```



| Bairro   | count(*) |
|----------|----------|
| Centro   | 3        |
| Tristeza | 1        |
| Moinhos  | 2        |

MEDICOS

| Codigo | Nome   | Bairro   |
|--------|--------|----------|
| E3     | João   | Centro   |
| E4     | José   | Tristeza |
| E1     | Laura  | Centro   |
| E6     | Carlos | Moinhos  |
| E7     | Nilton | Centro   |
| E9     | Bianca | Moinhos  |



| Codigo | Nome   | Bairro |
|--------|--------|--------|
| E3     | João   | Centro |
| E1     | Laura  | Centro |
| E7     | Nilton | Centro |



| Codigo | Nome | Bairro   |
|--------|------|----------|
| E4     | José | Tristeza |



| Codigo | Nome   | Bairro  |
|--------|--------|---------|
| E6     | Carlos | Moinhos |
| E9     | Bianca | Moinhos |

Condições em grupos

- **Cláusula HAVING**
  - define condições para que grupos sejam formados
  - condições só podem ser definidas sobre **atributos do agrupamento** ou serem **operações de agregação**
  - existe somente associada à cláusula GROUP BY
- **Exemplos**

```
select especialidade, count(*)
from Médicos
group by especialidade
having count(*) > 1
```

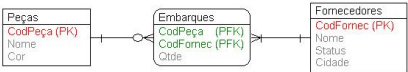
Condições em grupos



Obter o código dos fornecedores que realizaram mais de 3 embarques totalizando uma quantidade embarcada superior a 200 unidades

```
SELECT E.CodFornec, SUM(E.Qtde)
FROM Embarques E
GROUP BY E.CodFornec
HAVING COUNT(E.CodPeça) > 3
And SUM(E.Qtde) < 200
```

Condições em grupos



Obter os códigos de fornecedores que totalizam mais de 500 unidades de peças vermelhas embarcadas, junto com a quantidade de embarques de peças vermelhas

```
SELECT E.CodFornec, COUNT(E.CodPeça)
FROM Embarques E
WHERE CodPeça IN (SELECT P.CodPeça
FROM Peças P
WHERE P.Cor='Verm')
GROUP BY P.CodFornec
HAVING SUM(P.Qtde) > 500
```

Exercícios

**Ambulatorios** (numero, andar, capacidade)  
**Medicos** (CRM, nome, idade, especialidade, RG, cidade, numero) numero referencia ambulatorios  
**Pacientes** (CodP, nome, idade, cidade, RG, diagnostico)  
**Consultas** (CRM, CodP, data, hora) CRM referencia Medicos CodP referencia Pacientes  
**Funcionarios** (RG, nome, idade, cidade, salario)

1. Para as datas onde haja mais de uma consulta marcada, buscar estas datas e o total de consultas para cada uma destas datas
2. Buscar o número de todos os ambulatorios do hospital que possuem capacidade superior a 30, exceto os do primeiro andar
3. Nomes dos médicos que dão atendimento em ambulatorios com capacidade inferior a 50 leitos
4. Os dados de todos os pacientes e, para aqueles pacientes com consultas marcadas, exibir os dados das suas consultas
5. Número e andar dos ambulatorios onde nenhum médico dá atendimento
6. Nome dos médicos que não atendem em ambulatorios com capacidade superior à capacidade dos ambulatorios do segundo andar
7. Buscar as datas de consultas e o total de consultas em cada data
8. Nome dos médicos que possuem mais de uma consulta marcada