Centro Universitário Newton Paiva

BANCO DE DADOS

Prof. Iremar Nunes de Lima

iremar.prof@newtonpaiva.br
iremar.dba@gmail.com

currículo lattes: http://lattes.cnpq.br/0000426561067083

Índice

1. Fundamentos de Banco de Dados

- ***Banco de Dados:*** uma coleção de fatos conhecidos (dados) inter-relacionados.
- ***Exemplos de Banco de Dados:***
  - ✓ Um catálogo de livros de uma biblioteca.
  - ✓ Uma agenda telefônica contendo nomes, endereços e telefones para contato.
  - ✓ Uma universidade contendo dados dos professores, alunos e disciplinas.
- Um banco de dados informatizado pode ser criado e mantido por um SGBD (Sistema de Gerência de Banco de Dados): um conjunto de programas que facilita o processo de definição, construção e manipulação de bancos de dados.
- Representação de um BD:
  - ✓ Pode ser representado na forma de tabelas.
  - ✓ Exemplo para um BD de um sistema de controle de RH de uma empresa:

DEPARTAMENTO

COD_DEPARTAMENTO	NOM_DEPARTAMENTO
1	Pessoal
2	Vendas
3	Produção

EMPREGADO

COD_EMPREGAD	NOM_EMPREGADO	NUM_TELEFONE	VAL_SALARIO	COD_DEPARTAMENTO_LOTACAO
212	José Antônio	33326262	1967,98	2
321	Antônio Rodrigues	32124589	3045,84	3
109	Lucas da Silva	32124589	3605,00	3
450	Maria Fernandes	33129860	2398,45	1
867	Carlos Henrique	35634011	6099,00	3

PROJETO

COD_PROJETO	NOM_PROJETO	DTA_INICIO	DTA_FIM
001	Carro supersônico	02/03/2002	
002	Implantação ERP	23/01/2001	19/07/2001
003	Gestão de energia 2003	01/06/2002	

EMPREGADO_PROJETO

COD_PROJETO	COD_EMPREGADO	NUM_HORAS_TRABALHADAS
001	321	89
001	867	31
002	212	29
003	450	8
003	212	14

- *Outra definição de um BD*: uma coleção de dados armazenados cujo conteúdo informativo a cada instante representa o estado de uma determinada aplicação para um grupo de usuários

2. Projeto Conceitual de Banco de Dados

- Primeira etapa do projeto de Banco de Dados para um SI, cujo objetivo é obter uma ***descrição abstrata dos dados, independente do SGBD*** a ser utilizado para a implementação.
- Representa a necessidade dos dados pelo analista de sistemas após o levantamento de dados.
- Baseado em modelos: dois modelos se destacam atualmente:
  - ♦ Modelo de objetos baseados na UML (*Unified Modeling Language*): 1995.
  - ♦ Modelo ER (Entidade Relacionamento - DER): 1976.

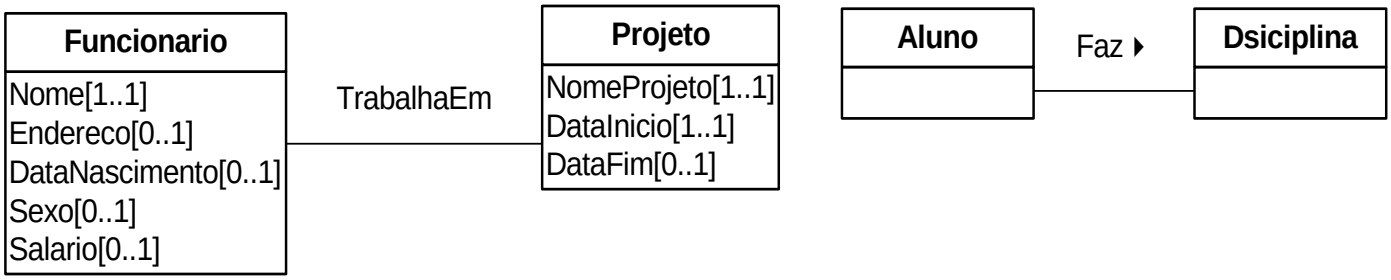
2.1. Modelo de Dados - Diagrama de Classes Persistentes (DCP)

- Descreve os dados como objetos, classes, atributos, operações e relacionamentos.
- Classes: agrupamento de objetos similares que possuem atributos e operações semelhantes.
- Em banco de dados estamos interessados nas ***classes persistentes***, ou seja, nas classes que possuem objetos que precisam ficar armazenados em meio físico por um período de tempo.
- Na UML uma classe é representada como nos exemplos a seguir:

- Nesta disciplina não indicaremos as operações das classes.
- Os atributos das classes podem ser opcionais ou obrigatórios, monovalorados ou multivalorados conforme mostrado nos exemplos acima.

Associações entre Classes:

- Dependendo da regra de negócio, objetos de duas ou mais classes podem se associar indicando um novo **fato que será armazenado** no banco de dados.
- A representação em UML de uma associação entre duas classes é mostrada a seguir:



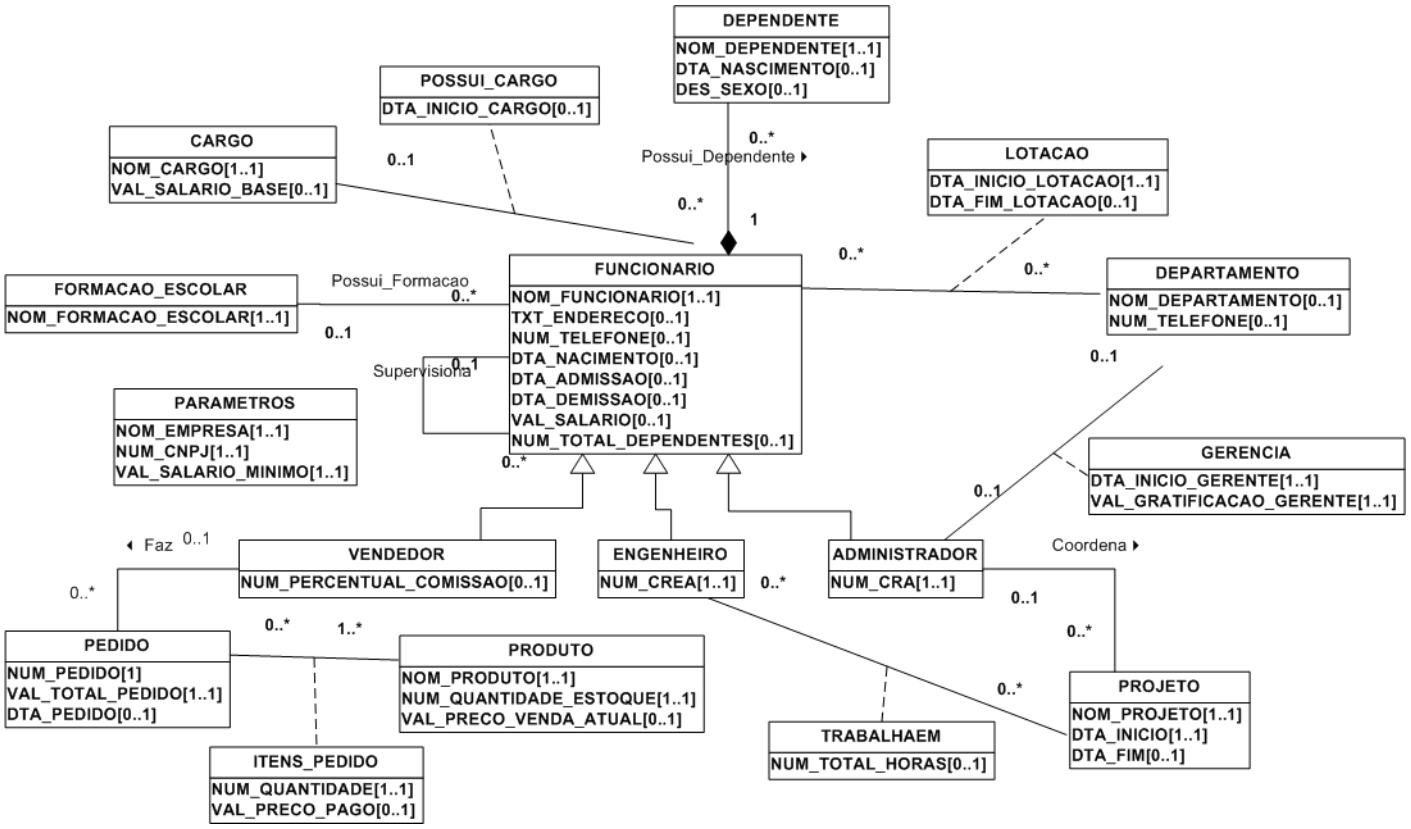
- ***Nesta disciplina*** só iremos criar associações entre classes se este fato tiver que ser armazenado no banco de dados por um período de tempo.
- Exemplos de associações, em geral, incorretas: Cliente ***consulta*** Produto, Funcionário ***registra*** Livro, Gerente ***emite*** relatório de Projetos.

2.2. Padronização do Nome dos Atributos

- Padronizar os nomes dos atributos ajuda a evitar confusões do significado do atributo e melhora a compreensão do dado junto aos desenvolvedores e usuários.
- Abaixo um exemplo para padronização de nomes de atributos que usaremos nesta disciplina.

Classe	**Identificador**	**Descrição**	**Exemplo**
Ano	ANO	Número que representa o ano do calendário. Formado por 4 posições.	ANO_PROCESSO
Arquivo	ARQ	Armazenar arquivos de formatos diversos como PDF, JPG, GIF, DOC.	ARQ_FOTO
Código	COD	Identificador único do objeto associado. Primary KEY da Tabela.	COD_CLIENTE
Data	DTA	Utilizado para representar Data e Hora.	DTA_NASCIMENTO
Descrição	DES	Descrição de atributo geralmente associado ao significado do objeto armazenado.	DES_PROJETO
Dia	DIA	Destinado a representar o número do dia do mês de 01 a 31.	DIA_FERIADO
Hora	HOR	Utilizado para representar hora, no formato de 00:00 a 23:59.	HOR_INICIO_ATIVIDADE
Indicador	IND	Utilizado para representar valores booleanos designados por “SIM” (S) ou “NÃO” (N).	IND_EXCLUSAO
Mês	MÊS	Número do mês de 01 a 12.	MES_ADMISSAO
Nome	NOM	Nome de um atributo associado ao objeto, apresentando uma relação de significação. Difere de DES (descrição) por ser mais resumido.	NOM_PROJETO
Número	NUM	Utilizado para representar valores numéricos.	NUM_CPF
Texto	TXT	Utilizado para representar texto livre, sem significação com o objeto.	TXT_CONTRATO
Valor	VAL	Utilizado para representar valores numéricos, expressos em moeda.	VAL_SALARIO

Exemplo: Modelo Conceitual de BD (DCP) para um sistema de controle de RH.



Observações:

1. O atributo **NUM_TOTAL_DEPENDENTES** é um atributo derivado.
2. O relacionamento **LOTACAO** guarda os históricos dos departamentos pelos quais os funcionários já ficaram lotados.
3. Não se está armazenando no Banco de Dados os históricos de Cargo dos funcionários e nem os históricos dos dias que cada funcionário trabalhou em um projeto.
4. Só se está armazenando o gerente atual do departamento. O mesmo vale para o coordenador atual do projeto.
5. Ao demitir um funcionário seus dados não serão removidos do banco de dados. Observe a existência do atributo **DTA_DEMISSAO**.
6. A classe **PARAMETROS** não se relaciona com nenhuma outra classe.
7. A maioria das classes não possui um atributo identificador (código). Isto não deve ser colocado no modelo conceitual.
8. Lembre-se: Este é um modelo didático e não pretende ser completo para um sistema de controle de RH de uma empresa real.

2.3. Estudos de Casos

1. Sistema para controle de biblioteca

Considere a seguinte lista de requisitos para o sistema de controle de uma biblioteca:

1. Funcionário registra no sistema os dados de livro (título, autores, assunto, etc).
2. Funcionário registra no sistema os dados de exemplares de livros que foram adquiridos pela biblioteca.
3. Funcionário registra no sistema os dados de alunos para usar a biblioteca.
4. Aluno usa o sistema para consultar dados de livro por assunto, autor ou por título.
5. Funcionário registra no sistema um empréstimo de um exemplar de livro solicitado pelo aluno.
6. Aluno solicita devolução de exemplar de livro que foi emprestado.
7. Bibliotecária emite um relatório contendo dados de empréstimos vencidos.

2. Sistema para controle de uma livraria

Considere a seguinte lista de requisitos para o sistema de controle de uma livraria:

1. Usuário cadastra dados da Livraria como nome de fantasia e telefone.
2. Usuário registra dados das Filiais da Livraria como nome da filial, endereço e telefones.
3. Usuário registra dados de vendedores que vão atuar em filiais.
4. Usuário registra dados de livros como título, autores, editora, assunto.
5. Usuário registra dados dos exemplares de livros adquiridos pelas filiais.
6. Vendedor registra venda de exemplares de livros aos clientes.
7. Usuário solicita relatório contendo o total de vendas do dia ordenado por filial e forma de pagamento.
8. Usuário solicita listagem contendo dados de livros com estoque crítico em cada filial.
9. Usuário solicita listagem com informações de pagamentos aos vendedores com as respectivas comissões recebidas no mês.

3. Sistema para controle de uma faculdade

Considere a seguinte lista de requisitos para o sistema de controle de uma faculdade:

1. Usuário registra grade curricular: os cursos e suas disciplinas com pré-requisitos.
2. Usuário cadastra dados do aluno que será vinculado a um determinado curso.
3. Usuário cria turmas de disciplinas a serem ofertadas em um semestre. Cada turma tem uma quantidade máxima de alunos que podem ser matriculados.
4. Aluno faz a matrícula em turmas de disciplinas ofertadas em um semestre.
5. Usuário define professor que vai lecionar as turmas de disciplinas ofertadas em um semestre.
6. Professor lança as frequências dos alunos matriculados nas turmas de disciplina que leciona.
7. Professor lança notas dos alunos matriculados nas turmas de disciplina que leciona.
8. Aluno consulta dados da matrícula no semestre atual como notas e frequências lançadas pelos professores.

4. Sistema disque pizza

O sistema de controle para uma pizzaria tem como finalidade dar suporte ao controle de cadastros de clientes, produtos (pizzas semi-prontas, refrigerantes, cervejas, etc), fornecedores, pedidos de venda aos clientes, pedidos de compra aos fornecedores, relatórios de contas a pagar e a receber. Ao realizar o pedido o cliente informa o seu telefone e o sistema identificará se o cliente está cadastrado (nome, endereço e telefone). Depois de confirmado o pedido pelo funcionário o cliente indica a forma de pagamento e fica aguardando a entrega. O sistema deve emitir uma ordem de entrega, dar baixa no estoque para que se possa emitir pedidos de compras aos fornecedores, se necessário. O sistema possuirá um cadastro de fornecedores com os respectivos produtos que fornecem mantendo no banco de dados a cotação de preços junto aos fornecedores para auxiliar na compra de produtos com estoque baixo. O sistema emitirá relatórios diários de fluxo de caixa (compras e vendas) para a gerência. Também deve ser emitida a relação de produtos com estoque crítico, consultas sobre a situação de um pedido de um cliente ou de um fornecedor (em aberto, cancelado, a caminho, finalizado, pago, etc), relação de pedidos a vencer dos fornecedores e relação do melhor fornecedor para um produto que está acabando. Nenhum controle de funcionários deve ser feito neste sistema. A pizzaria não fabrica pizza: ela sempre compra pizzas semi-protas de seus fornecedores para assá-las e vendê-las aos clientes.

5. Sistema para uma clínica médica

Uma clínica médica possui vários médicos atuantes em diversas especialidades como clínico geral, cardiologia, pediatria, oftalmologia, etc. Um médico pode atuar em diferentes especialidades e pode atender em diversos horários do dia e da semana. Um paciente só pode marcar uma consulta se estiver cadastrado. Sempre que uma consulta é marcada é aberto um prontuário automatizado do cliente que conterá os históricos das consultas já realizadas pelo paciente na clínica. A marcação de consultas para os pacientes está relacionada à agenda do médico. Esta agenda deve ser emitida diariamente para controle das consultas do médico. Diferentes tipos de exames também podem ser marcados e realizados na Clínica. Cada tipo de exame requer determinadas condições do paciente, como jejum de 24 hs e vasilhames específicos. Os históricos de todos os exames marcados devem ficar armazenados no sistema. O resultado final do exame deverá estar automatizado contendo a conclusão final do exame. Atualmente, a clínica atende também através de convênios, que cobrem consultas e exames. O paciente só pode estar vinculado a um convênio. É importante saber quais os tipos de exames da clínica uma determinada empresa conveniada cobre. Um paciente pode marcar, cancelar e remarcar consultas e exames. Isto determina o status da consulta ou do exame. A fatura aos conveniados é emitida ao final do mês e para

consultas estratégicas os dados das faturas devem ficar armazenados no banco de dados do sistema. Diversas consultas podem ainda ser realizadas como médicos de uma dada especialidade com horários disponíveis para um dado dia e hora, dias e horários de atendimentos de um dado médico, horários de consultas e exames de um dado paciente, especialidade mais procurada, faturamento no mês com consultas e exames e conveniadas em débitos com a clínica.

6. Sistema para uma escolinha de esportes

Um clube é frequentado por sócios titulares e seus dependentes. Quando um sócio se associa ao clube é gerado uma carteirinha para ele e seus dependentes que podem ser incluídos depois. Para frequentar as dependências do clube todo mês o sócio titular deve pagar uma taxa que varia de acordo com a quantidade de dependentes que ele possui. Para controlar o pagamento desta taxa o sistema deve gerar um boleto contendo os dados do sócio titular, mês de pagamento, valor total, data de vencimento, data de pagamento e multa. Estes dados devem ser armazenados no banco de dados. O clube oferece várias aulas de um determinado esporte. A cada semestre são montadas várias turmas em diversos horários para que o sócio (titular ou dependente) possa se matricular. Dessa forma os sócios e seus dependentes terão escolha na hora da matrícula. Mensalmente deve ser emitido um relatório de oferta de turmas por esporte. O controle dos professores e as turmas de esportes que lecionam deve ser feito pelo sistema. Deve ser gerado um relatório geral de sócios por turmas com os respectivos professores. A folha de pagamento e contracheque dos professores também é controlada pelo sistema de acordo com as turmas pelas quais são responsáveis. Todo mês é emitido uma fatura ao sócio titular contendo dados das turmas, valor total, data de vencimento e data de pagamento, de acordo com as aulas que ele e seus sócios dependentes fizeram no mês. Deve ser mantido no sistema todos os históricos das faturas pagas, das matrículas dos alunos e dos esportes já lecionados pelos professores e controle de vagas disponíveis. Para efeito da legislação todos os históricos de contracheques pagos aos professores devem ser mantidos no sistema. A emissão de boletas, faturas, carteirinhas e contracheques pode ser feito a qualquer momento pelo sistema.

7. Sistema de comércio eletrônico (ENADE 2014)

Uma empresa deseja lançar um sistema de comércio eletrônico para vender seus produtos. Essa empresa vende produtos de diversas categorias, como roupas, perfumes e eletrônicos, e aceita diversas formas de pagamento, como cartão de crédito e boleto bancário. No sistema de vendas implementado, cada produto deve ser cadastrado com sua descrição, preço de venda, quantidade em estoque e respectiva categoria. Cada cliente que deseja realizar compras tem de se cadastrar no sistema indicando seu nome, endereço e *e-mail*. Se o cliente for corporativo, deve cadastrar seu CNPJ e, se for individual, seu CPF. O cliente cadastrado pode realizar um pedido de compra dos produtos em estoque na quantidade que desejar. O cliente escolhe uma forma de pagamento disponível e recebe, por *e-mail,* o número do pedido e informações do *status* do pedido. Após a confirmação do pagamento, a loja realiza a entrega dos itens solicitados no endereço do cliente e envia, por *e-mail,* a nota fiscal eletrônica. Tendo em vista que os preços dos produtos podem ser atualizados a qualquer momento, o sistema tem de ser capaz de reemitir uma nota fiscal de um pedido de compra de qualquer produto e respectivo preço na data da compra realizada pelo cliente.

3. Projeto Lógico Relacional de Banco de Dados

- Segunda etapa do projeto de Banco de Dados para um SI, cujo objetivo é obter uma descrição de como implementar o BD: depende do SGBD escolhido.
- Existem vários tipos de SGBDs no mercado que implementam diferentes modelos: Modelo Relacional, Modelo de Redes, Modelo Hierárquico, Modelo Orientado a Objetos, etc.

3.1. Projeto Relacional de Banco de Dados

- Proposto por Edgar F. Cood nos anos 70, dando origem aos SGBDRs.
- Princípio: As linhas de uma tabela representam os fatos descritos nas classes.

Tabela DEPARTAMENTO		
Nome	Cod_Departamento (PK)	Cod_Func_Gerente (FK)
Engenharia e Produção	1	1
Tecnologia e Informação	2	3
Administração Pessoal	3	2

- Durante a especificação do projeto deve-se destacar o tipo de dados, o tamanho e o domínio do atributo que está sendo especificado de acordo com o SGBDR.

Coluna	**Tipo**	**Tamanho**	**Domínio**
Nome	Char	30	a-z, A-Z
Cod_Departamento	Int	3	0 a 999
Cod_Func_Gerente	Int	4	0 a 9999

- Principais tipos de dados do SGBD Microsoft SQL Server: Tinyint, Smallint, Int, Bigint, Decimal (n,m), Real, Float, Char(n), Varchar(n), Text, Datetime, Smalldatetime, Money, Smallmoney, Bit, Image.
- Toda tabela deve possuir a ***chave primária*** (Primary Key - PK) que deve ser mínima.
- As ***Associações/Relacionamentos*** entre duas classes/entidades do modelo conceitual são implementados no modelo relacional com o uso de ***chaves estrangeiras*** (Foreign Key - FK).
- A FK é formada por um ou mais atributos chaves da tabela original que são adicionados numa das tabelas relacionadas, propiciando uma ligação lógica entre as linhas das tabelas.

Tabela DEPARTAMENTO		
NOM_DEPARTAMENTO	COD_DEPARTAMENTO (PK)	COD_FUNCIONARIO_GERENTE (FK)
Engenharia e Produção	1	1
Tecnologia e Informação	2	3
Administração Pessoal	3	2

Tabela FUNCIONARIO				
NOM_FUNCIONARIO	COD_FUNCIONARIO (PK)	COD_DEPARTAMENTO (FK)	COD_FUNC_SUPERVISOR (FK)	VAL_SALARIO
João Luiz	*1*	*2*	***NULL***	*3.000,00*
Fernanda	*2*	*1*	*1*	*2.500,00*
Ricardo	*3*	*2*	*1*	*2.300,00*
Renata	*4*	*NULL*	*2*	*4.200,00*

- As FKs não precisam ter o mesmo nome das chaves primárias.

- Os atributos FKs precisam ter do mesmo tipo de dado que as PKs que referenciam.
- A abordagem relacional gera a restrição de ***integridade referencial***: os valores possíveis das FKs devem pertencer ao conjunto dos valores das PKs referenciadas e existentes atualmente no BD ou possivelmente assumirem o valor NULL. Não pode existir no atributo que é chave estrangeira um valor que não exista na tabela na qual ela é chave primária.

3.2. Mapeamento do Modelo Conceitual para o Modelo Lógico Relacional

Regras de Transformação:

- Durante a aplicação das regras levar em consideração o desempenho, a manutenção e a perda de espaço de armazenamento nesta ordem.

Regra 1: Cada classe/entidade forte vira uma tabela, cada atributo simples vira uma coluna com um tipo de dado e domínio a ser definido. Identificar a chave primária (PK) e representá-la com a palavra PK na frente do nome do atributo.

Regra 2: Associações 1:N: Acrescente o atributo chave do lado 1 na tabela do lado N como FK contendo um relacionamento 1:N. Os atributos dos relacionamentos serão acrescentados na tabela do lado N. Se houver a participação parcial do lado 1 a chave estrangeira pode ser NULL. Identifique o atributo que é chave estrangeira com a palavra FK.
Exercício:
1) Relacionamento 1:N com participação total e tendo atributos próprios:
Departamento(Des_Sigla, Nom_Departmento)
Empregado (Cod_Empregado, Nom_Empregado, Des_Endereco, ...)
Lotacao [1:N][Departamento, Empregado] (Dta_Inicio)
2) Relacionamento 1:N com participação parcial e tendo atributos próprios:
Departamento(Des_Sigla, Nom_Departamento)
Empregado (Cod_Empregado, Nom_Empregado, Des_Endereco, ...)
Lotacao [1:N][Departamento, Empregado] (Dta_Inicio)

Regra 3: Toda associação N:N vira uma tabela contendo dois relacionamentos 1:N. As chaves primárias das tabelas relacionadas viram PK e FK na nova tabela gerada. Os atributos dos relacionamentos se existirem ficam na tabela gerada. As FKs não podem aceitar valores NULL. Estes atributos serão identificados com as palavras PK FK na frente do nome do atributo que tiver esta característica.
Exercícios: Aplique a regra aos seguintes casos:
1) Pedido(Num_Pedido, Val_Total, Dta_Pedido)
Produto(Cod_Produto, Nom_Produto, Val_Quantidade_Estoque, Val_Preco_Venda)
Contém[N:N][Pedido, Produto](Val_Quantidade, Val_Preco_Pago)
2) Medico(Num_CRM, Nom_Medico, Des_Endereco)
Paciente(Cod_Paciente, Nom_Medido, Des_Endereco)
Consulta[N:N][Medico, Paciente](Dta_Consulta, Val_Pago, Des_Diagnostico)

Regra 4: Associações 1:1. Adicione a PK de uma das tabelas como FK na outra tabela. Represente o relacionamento como 1:1. Os atributos do relacionamento serão colocados na tabela que possuir a FK. Use a seguinte estratégia:

a) Se houver participação parcial/total ou total/parcial coloque a FK na tabela que tem participação parcial.
b) Caso contrário coloque, se possível, a FK na tabela que possua menos linhas.

Exercício: Participação total/parcial
Professor(Cod_Professor, Nom_Professor, Des_Endereco)
Curso(Cod_Curso, Des_Cursos)
Coordena[1:1][Professor, Curso](Val_Gratificacao)

Regra 5: Composição da UML: A classe fraca vira uma nova tabela. Adicione a PK da tabela do lado 1 como PF FK na tabela do lado N. Escolha um ou mais atributos da tabela do lado N para compor a PK. Represente o relacionamento como 1:N. O atributos dos relacionamentos serão acrescentados na tabela do lado N.

Exercício: Funcionario (Nom_Empregado, Des_Endereco, ...)
Dependente (Num_Dependente, Nom_Dependente)
Possui[1:N - Composição][Funcionario, Dependente]

Regra 6: Auto Associações: Aplique as regras para relacionamentos 1:N, N:N e 1:1.

Exercício: Disciplina(Cod_Disciplina, Nom_Disciplina, Val_Carga_Horaria)
Pre-Requisito [N:N][Disciplina, Disciplina].

Regra 7: Herança: As seguintes soluções podem ser adotadas:

a) Solução padrão: gere uma tabela para a classe pai e para cada classe filha. Propague a PK da tabela Pai com PK FK nas tabelas filhas utilizando o relacionamento 1:1.
b) Pode-se gerar uma única tabela acrescentando todos os atributos das classes filhas na tabela gerada. Crie vários atributos "flags" que identifiquem cada uma das entidades filhas. Esta opção pode gerar um grande número de valores nulos e não é boa se as classes filhas participam de algum relacionamento.
c) Outra opção seria gerar duas tabelas para cada classe filha. Os atributos da classe pai são propagados para as tabelas geradas. Esta opção não é boa se a classe pai se relaciona com outras classes.

Exercício: a) Pessoa, Pessoa Física e Pessoa Jurídica.
b) Funcionario, Engenheiro, Administrador.

Exercícios: Desenhe o modelo conceitual e em seguida proponha o modelo relacional para os seguintes casos:

a) Filme (Cod_Filme, Des_Titulo)
Ator (Cod_Ator, Nom_Ator, Ind_Sexo)
Elenco [N:N][Filme, Ator]

b) Estado (Des_Sigla, Nom_Estado)
Governador (Cod_Governador, Nom_Governador)
Governa [1:1][Estado, Governador] (Dta_Inicio)

c) Cliente (Nom_Cliente)
Cliente_Titular (Num_CPF)
Cliente_Dependente (Dta_Nascimento)
[Herança][Cliente,ClienteTitular, ClienteDependente]
Possui [1:N][ClienteTitular, ClienteDependente]

e) Pessoa (Nom_Pessoa)
Mãe [1:N][Pessoa, Pessoa]
Pai [1:N][Pessoa, Pessoa]

f) Aluno (Num_RA, Nom_Aluno)
Disciplina (Nom_Disciplina)
Turma_Disciplina (Vagas)
Matricula [N:N][Aluno, Turma_Disciplina] (Val_Nota, Val_Frequencia)
Tem [1:N - Composição][Disciplina, Turma_Disciplina]

g) Cliente (Nome), Filme (Titulo), Exemplar (Num_Exemplar, Dta_Compra_Exemplar),
Ator (Nome), Gênero (Nome), Parâmetros (Nom_Locadora,Val_Diaria_Multa),
Tem_Exemplar [1:N - Composição][Filme, Exemplar],
Empréstimo [N:N][Cliente, Exemplar](Dta_Emprestimo,Val_Pago)
Ator_Filme [N:N][Ator,Filme], Tem_Genero [1:N][Gênero, Filme], Édependente [1:N][Cliente,Cliente]

h) Usuario (nome), Pedido (Num_Pedido, Dta_Pedido), Status_Pedido (Nom_Status),
FazPedido [N:N][Usuario, Pedido] (Dta_Pedido),
TemStatus [1:N][Status_Pedido, Pedido]

i) Socio (Nom_Socio), Esporte (Nom_Esporte), Turma_Esporte(Num_Vagas), Categoria(Des_Categoria),
Tem[1:N – Composicao][Esporte,Turma_Esporte],
Faz_Turma_Esporte[N:N][Socio, Turma_Esporte] (Val_Pago),
Possui[1:N][Categoria, Esporte]

j) Cliente (Nom_Cliente), Solicitacao (Num_Solicitacao, Des_Solicitacao, Status_Solicitacao (Des_Status), Faz_Solicitacao [N:N][Cliente, Solicitacao] (Dta_Solicitacao), TemStatus [1:N][Status_Solicitacao, Solicitacao]

3.3. Normalização

- Processo que procura dividir uma tabela em várias outras evitando-se redundância de dados e anomalias de atualização de atributos de tabelas.
- Um bom projeto conceitual de BD (DCP) quase sempre evita o processo de normalização.
- O processo de normalização é baseado em três formas normais principais (existem outras, mas as três primeiras são suficientes para a maioria dos casos).

Primeira Forma Normal (1FN): Uma tabela não pode conter atributos multivalorados. Em outras palavras uma tabela não pode conter grupos de atributos repetitivos.
Exercícios: Coloque na 1 FN as seguintes tabelas.

1) Empregado(CodFunc PK, Nome, Telefone[0..*])
2) Pedido (NrPedido PK, PrazoEntrega, NomeCliente, EnderecoCliente, CodProduto, QuantidadeProdutoComprado, NomeProduto, PrecoAtualProduto, ValorTotalPedido, CodVendedor, NomeVendedor)

Segunda Forma Normal (2FN): Uma tabela está na 2FN se estiver na 1FN e se ela não possuir atributos que dependem parcialmente da chave primária composta. Ou seja, a tabela não pode conter atributos com dependência parcial da chave.

Obs: Se uma tabela possuir só um atributo chave e estiver na 1FN então automaticamente estará na 2FN.

Terceira Forma Normal (3FN): Uma tabela está na 3FN se estiver na 2FN e se ela não possuir atributos não chaves que dependem de outro atributo não chave. Ou seja, a tabela não pode conter atributos com dependência transitiva.

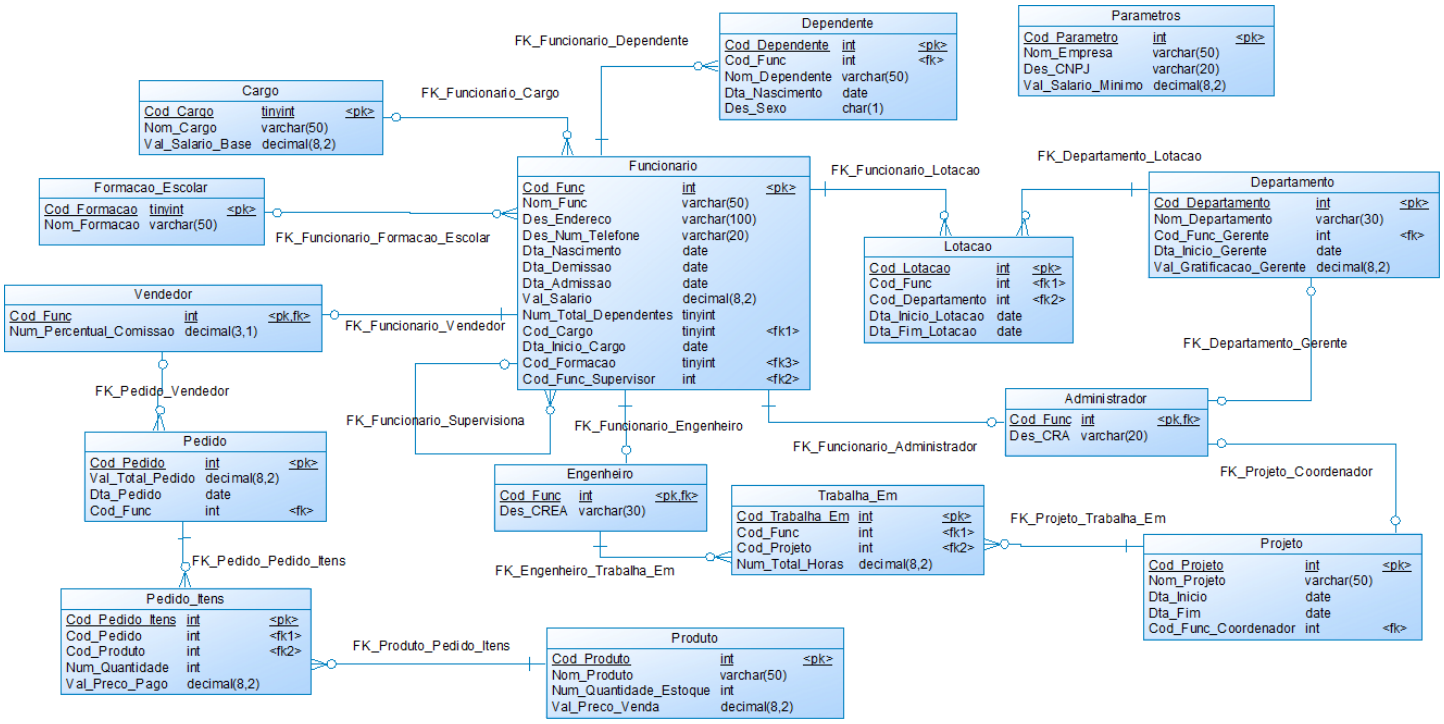
Exercícios:
Qual forma normal as tabelas abaixo não satisfaz? Em seguida normalize-as.

a) Boleta (Num_RA PK, Dta_Vencimento_Boleta PK, Nom_Aluno, Val_Total_Boleta, Dta_Emissão_Boleta, Dta_Pagamento).
b) Aluno (Num_RA PK, Cod_Disciplina, Nom_Aluno, Val_Nota_Disciplina).
c) Pedido (Num_Pedido PK, Dta_Pedido, Num_Vendedor, Val_Total_Pedido, Nom_Vendedor).
d) Historico_Escolar (Num_RA PK, Cod_Disciplina PK , Nom_Aluno, Nom_Disciplina, Val_Nota_Final).
e) Pedido (Num_Pedido PK, Dta_Pedido, Num_CPF, Nom_Cliente, Val_Total_Pedido, Cod_Produto, Nom_Produto).

4. Projeto Físico de Banco de Dados: A Linguagem SQL

- SQL (Structured Query Language) é uma linguagem padrão de alto nível (não procedural) para criar (DDL) e manipular (DML) um banco de dados em SGBDRs e SGBDORs.
- Inicialmente foi mantida pelo organismo chamado ANSI (American National Standards Institute) e atualmente é regulada pela ISO (*International Organization for Standardization*)
- Versões da SQL lançadas: SQL86, SQL89, SQL92, SQL99, SQL2003, SQL2008, SQL2016.
- Para efeitos didáticos e práticos de SQL vamos trabalhar com o seguinte estudo de caso:

DER Relacional no Power Design para o BD RH



4.1. Definição e Manipulação de Dados em SQL

- Criando/Removendo o Banco de Dados:
  CREATE DATABASE BD_RH;
  DROP DATABASE BD_RH;

- Criando tabelas:

```
CREATE TABLE Cargo
(  Cod_Cargo          INTEGER        NOT NULL,
   Nom_Cargo          VARCHAR(30)    NOT NULL,
   Val_Salario_Base   DECIMAL(8,2)   NULL
);
```

ALTER TABLE Cargo ADD PRIMARY KEY(CodCargo);

```
CREATE TABLE TrabalhaEm
(
   Cod_Func           INTEGER        NOT NULL,
   Cod_Projeto        INTEGER        NOT NULL,
   Num_Total_Horas    DECIMAL(3,1)   NOT NULL
);
```

ALTER TABLE Trabalha_Em ADD PRIMARY KEY (Cod_Func, Cod_Projeto);

ALTER TABLE Trabalha_Em ADD FOREIGN KEY (Cod_Func) REFERENCES Funcionario (Cod_Func) ;

ALTER TABLE Trabalha_Em ADD FOREIGN KEY(Cod_Projeto) REFERENCES Projeto (Cod_Projeto) ;

```
CREATE TABLE Funcionario (
    Cod_Func      INTEGER      NOT NULL,
    Nom_Func      VARCHAR(50)      NOT NULL,
    Val_Salario   DECIMAL(8,2) NULL CHECK (Val_Salario >= 0 OR Val_Salario IS NULL),
    Cod_Cargo     INTEGER      DEFAULT (1),
   Dta_Nascimento DATE
);
```

- Alterando a estrutura de uma tabela:
  ALTER TABLE Dependentes **ADD** Num_Telefone char(15) NULL;

  ALTER TABLE Dependentes **ALTER** COLUMN Num_Telefone char(20) NULL;

  ALTER TABLE Dependentes **DROP** COLUMN Num_Telefone;

- Removendo tabela:
  DROP TABLE Funcionario;
  DROP TABLE Cargo;

- ***Manipulando Linhas nas Tabelas:***
  Para inserir, alterar e excluir linhas em tabelas deve-se satisfazer as restrições de chaves (PKs), de integridades referenciais (FKs) e as restrições de domínio.

- Exemplos:
  INSERT INTO Cargo (Cod_Cargo, Nom_Cargo, Val_Salario_Base)
  VALUES (1,'Engenheiro', 1890.00);

  INSERT INTO Cargo VALUES (2,'Administrador', 2780.00);

  Atenção: Para não ter problemas com atributos do tipo DATETIME, caso esteja usadno o SGBD SQL SERVER, execute antes o comando SET DATEFORMAT DMY;

  INSERT INTO Lotacao (Cod_Func, Cod_Departamento, Dta_Inicio_Lotacao, Dta_Fim_Lotacao) VALUES (1, 2, '01/04/2000', NULL);

  Obs: O funcionário 1 e o Departamento 2 já devem ter sido inseridos antes.

  DELETE FROM Funcionario
  WHERE Cod_Func = 1

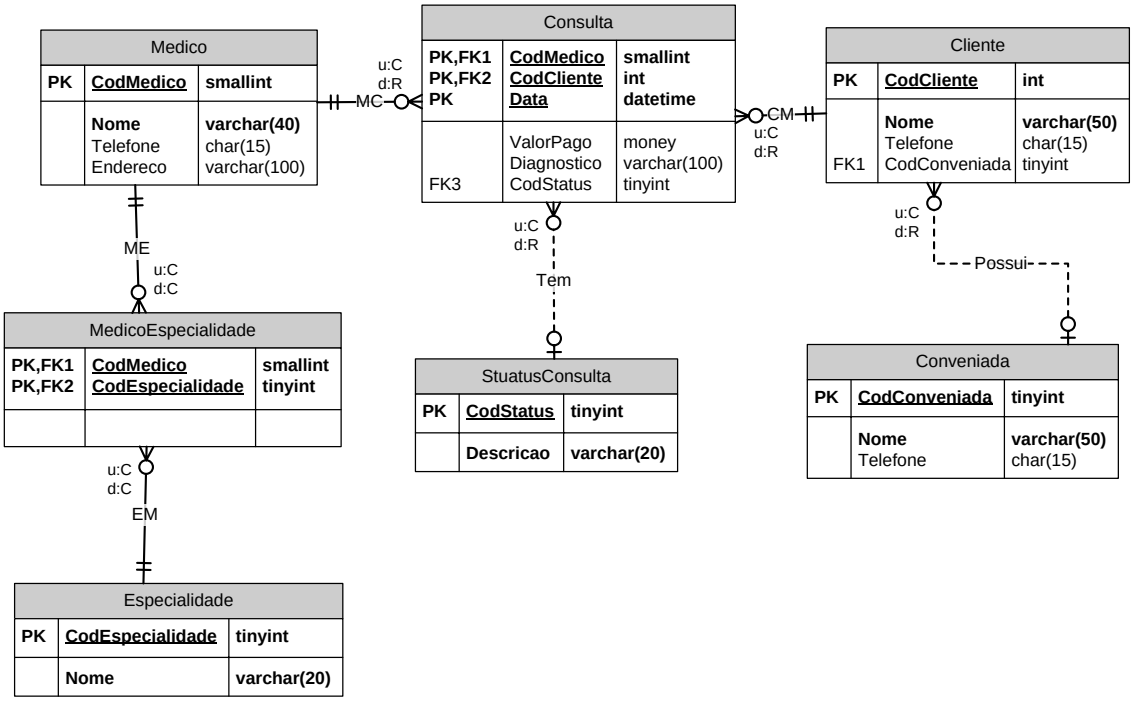
  DELETE FROM Funcionario
  WHERE Val_Salario > 200

  DELETE FROM Funcionario

  UPDATE Funcionario
  SET Val_Salario = 1000
  WHERE Cod_Func = 10

  UPDATE Funcionario
  SET Val_Salario = Val_Salario * 1.20

Exercício: observe o DER Relacional abaixo para um sistema de uma clínica médica.



1) Com base no DER relacional acima proponha o script SQL padrão ANSI de criação do Banco de Dados.
2) Proponha o código SQL que insira duas linhas em cada tabela do BD na ordem correta.
3) Proponha o código SQL que remova uma linha da tabela Consulta.
4) Proponha o código SQL que remova todas as consultas com status igual a 5.
5) Proponha o código SQL que atualize o endereço do médico de código 3.
6) Proponha o DCP a partir do DER relacional acima.

4.2. Consultas em SQL

- Uma consulta em SQL pode ter a seguinte forma:

```
SELECT atributo(s)
FROM tabela(s)
[WHERE condição de seleção]
[GROUP BY atributo(s)]
[HAVING condição de pesquisa]
[ORDER BY atributo(s)]
```

Exemplo 1: Recupere todos os Departamentos da Empresa:

a) Opção 1:

```
SELECT Cod_Departamento,Nom_Departamento,Cod_Func_Gerente, Dta_Inicio_Gerente
FROM   Departamento
```

b) Opção 2:

```
SELECT *
FROM Departamento
```

Exemplo 2: Recupere o Nome e Endereço dos Funcionários que tenham salário igual ou inferior a 5000 reais e que tenham sido admitidos após "31/12/2000".

```
SELECT    Nom_Func, Des_Endereco
FROM      Funcionario
WHERE     Val_Salario <= 5000.00  AND  Dta_Admissao >= '01/01/2001'
```

Exemplo 3: Recupere os funcionários que tenham supervisores e que foram admitidos no ano de 2013:

```
SELECT    *      FROM      Funcionario
WHERE     Cod_Func_Supervisor IS NOT NULL      AND
          Dta_Admissao BETWEEN  '01/01/2013'  AND  '31/12/2013'
```

Exemplo 4: Recupere os nome e endereço dos Funcionários que residem em BH e que sejam supervisionados pelos funcionários de códigos 1,4, 6, ou 13:

```
SELECT    Nom_Func, Des_Endereco
FROM      Funcionario
WHERE     Des_Endereco LIKE '%BH%' AND Cod_Func_Supervisor IN (1,4,6,13)
```

Pode-se usar os seguintes operadores aritméticos (*, /, -, +) nas cláusulas SELECT e WHERE.

Exemplo 5: Recupere o nome e Telefone do funcionário que trabalha a mais de um ano na empresa e que teria salário menor que 3000.00 reais caso seu salário atual fosse reajustado em 15 %. Ordene o resultado da consulta pelo Nome do funcionário.

```
SELECT Nom_Func AS "Nome do Funcionário", Des_Num_Telefone AS "Telefone do Funcionário"
FROM Funcionario
WHERE ( getdate() - Dta_Admissao ) > 365 AND (Val_Salario * 1.15) < 3000
ORDER BY Nom_Func
```

Exercícios:

1. Recupere o nome e salário dos funcionários com até 2 dependentes.
2. Recupere o nome e salário dos funcionários não demitidos que não tem supervisor.
3. Recupere os produtos com estoque inferior a 2 unidades. Exiba primeiro os produtos com menor preço de venda.
4. Recupere todos os pedidos feitos a mais de um ano.
6. Recupere o nome dos dependentes do sexo masculino ou que tenham mais de 5 anos de idade.
7. Recupere o nome e salario funcionários não demitidos que tenham no endereço a string BH.
8. Usando somente a tabela funcionário recupere os funcionários que são Vendedores e também Engenheiros.

4.3. Operadores Relacionais

- Várias consultas em um BD podem envolver duas ou mais tabelas do BD.
- A linguagem SQL possibilita a combinação de vários dados de diferentes tabelas em uma única consulta através do JOIN (INNER JOIN).

Exemplo 1: Selecione o nome dos funcionários e o nome dos seus dependentes ordenados pelo nome do funcionário seguido pelo nome do seus dependentes.

```
SELECT    F.Nom_Func AS "Funcionário", D.Nom_Dependente AS "Dependente"
FROM      Funcionario F INNER JOIN Dependente D ON (F.Cod_Func = D.Cod_Func)
ORDER BY F.Nom_Func, D.Nom_Dependente
```

Exemplo 2: Selecione o nome dos departamentos com o nome dos funcionários atualmente lotados nele mostrando o salário de cada funcionário.

```
SELECT    D.Nom_Departamento, F.Nom_Func, F.Val_Salario
FROM      Departamento D INNER JOIN Lotacao L
          ON (D.Cod_Departamento=L.Cod_Departamento)
INNER JOIN Funcionario F ON (L.Cod_Func = F.Cod_Func)
WHERE     L.Dta_Fim_Lotacao IS NULL
```

Exemplo 3: (Left Outer Join) Recupere o nome dos projetos e o nome de seus coordenadores caso tenha:

```
SELECT    P.Nom_Projeto AS "Nome do Projeto", F.Nom_Func AS "Nome do Coordenador"
FROM      Projeto P LEFT OUTER JOIN Funcionario F
          ON (P.Cod_Func_Coordenador = F.Cod_Func)
```

Exemplo 4: (Right Outer Join) Mostre o nome de todos funcionários com os respectivos cargos, mesmo para os funcionários que não tenham cargos, ordenados pelo nome do funcionário:

```
SELECT    F.Nom_Func, C.Nom_Cargo
FROM      Cargo C RIGHT OUTER JOIN Funcionario F ON (C.Cod_Cargo = F.Cod_Cargo)
ORDER BY F.Nom_Func
```

Exemplo 5: (Full Outer Join) Recupere o nome dos funcionários com seus cargos. Mostre os funcionários que não tenham cargo e os cargos que não tenham funcionários associados:

```
SELECT    F.Nom_Func, C.Nom_Cargo
FROM      Funcionario F FULL OUTER JOIN Cargo C ON  (F.Cod_Cargo = C.Cod_Cargo)
```

Exercícios (com INNER JOIN):

1. Recupere o nome, cargo e formação escolar dos funcionários não demitidos.

2. Recupere o nome dos funcionários e o nome dos seus dependentes. Ordene os dados pelo nome do funcionário com maior idade primeiro.

3. Recupere o código do pedido, data do pedido, valor do pedido, nome do produto que apareceu no pedido e a quantidade comprada. Ordene os dados pela data do pedido em crescente.

4. Usando três tabelas, recupere o nome do Projeto, o nome do funcionário Coordenador do Projeto.

5. Usando duas tabelas, recupere o nome do Projeto, o nome do funcionário Coordenador do Projeto.

6. De forma mais otimizada possível, recupere o nome dos Projetos não encerrados, o Nome dos funcionários e o total de horas que cada funcionário já trabalhou no projeto. Ordene os dados mostrando os projetos mais novos primeiro.

7. Recupere o nome dos funcionários não demitidos que são vendedores, coordenadores de projetos, gerentes de departamentos ao mesmo tempo.

8. Recupere o nome dos Departamentos, o Nome dos funcionários não demitidos lotados atualmente no departamento e a idade de cada funcionário.

4.4. Funções Agregadas

SQL possui 4 funções chamadas funções agregadas (COUNT, SUM, AVG, MAX e MIN).

Exemplo 1: Recupere a soma dos salários de todos os empregados, o maior salário, o menor salário, e a média dos salários:

```
SELECT    COUNT(Cod_Func)     AS "Total de funcionários",
          SUM(Val_Salario)    AS "Soma dos salários",
          MAX(Val_Salario)    AS "Maior Salário",
          MIN(Val_Salario)    AS "Menor Salário",
          AVG(Val_Salario)    AS "Media dos Salários"
FROM Funcionario
```

Total de funcionários	Soma dos salários	Maior Salário	Menor Salário	Media dos Salários
17	45386,89	15040,00	710,00	2836,6806

Exemplo 2: Recupere o nome do cargo, a quantidade de empregados que possui o cargo e média salarial dos empregados que exercem este cargo:

```
SELECT    C.Nom_Cargo, COUNT(F.Cod_Func) AS "Total Empregados no Cargo",
          AVG(F.Val_Salario) AS "Média Salarial"
FROM      Cargo C Left Outer Join Funcionario F ON (F.Cod_Cargo = C.Cod_Cargo)
GROUP BYC.Nom_Cargo
```

Exemplo 3: Recupere o nome dos funcionários admitidos a mais de três anos e que trabalham em mais de um projeto. Mostre o total de projetos que eles trabalham.

```
SELECT    F.Nom_Func, COUNT(T.Cod_Projeto) AS 'Total Projetos'
FROM      Funcionario F INNER JOIN Trabalha_Em T ON (F.Cod_Func = T.Cod_Func)
WHERE     (Getdate() - F.Dta_Admissao) > 3 * 365
GROUP BY F.Cod_Func, F.Nom_Func
HAVING    COUNT(T.Cod_Projeto) > 1
```

Exemplo 4: Recupere o nome dos departamentos e a quantidade de empregados lotados atualmente mostrando somente os departamentos que tenham até 3 funcionários lotados, ordenados pelo total de empregados em ordem decrescente.

```
SELECT D.Nom_Departamento, Count(L.Cod_Func) AS "Total empregados lotados"
FROM Departamento D LEFT OUTER JOIN Lotacao L ON (D.Cod_Departamento = L.Cod_Departamento)
WHERE L.Dta_Fim_Lotacao IS NULL
GROUP BY D.Cod_Departamento, D.Nom_Departamento
HAVING Count(L.Cod_Func) <= 3
ORDER BY Count(L.Cod_Func) DESC
```

Obs: 1.Na cláusula Group By é obrigatório que apareça todos os atributos da cláusula SELECT.
2.Se a cláusula Order By for utilizada ela deve aparecer depois da cláusula Group By.

Exercícios:

1) Calcule a média salarial dos funcionários não demitidos.

2) Recupere o nome dos funcionários admitidos a mais de três anos e que trabalham em no máximo dois projetos
3) Calcule o total de funcionários vendedores com salário acima de 1000 reais.
4) Qual o valor do maior e menor salário dentre os funcionários coordenadores de projeto?
5) Mostre o nome do departamento e o total de funcionários não demitidos lotados atualmente no departamento.
6) Recupere o nome de todos os funcionários, o total de projetos em que trabalharam e o total de horas que já trabalharam nos projetos.
7) Recupere o nome dos departamentos que contenham até cinco funcionários lotados atualmente.
8) Recupere o nome dos projetos com mais de 2 funcionários não demitidos trabalhando nele.
9) Recupere quantos são e a média salarial dos funcionários que são vendedores e coordenadores de projeto ao mesmo tempo.

4.5. Consultas Avançadas em SQL: Parte I

Exemplo 1: **Auto Relacionamento -** Recupere o Nome e salário dos funcionários e o nome e salário de seus supervisores.

```
SELECT F.NOM_FUNC AS NOME_FUNC, F.VAL_SALARIO AS SALARIO_FUNC,
S.NOM_FUNC AS NOME_SUP, S.VAL_SALARIO AS SALARIO_SUP
FROM FUNCIONARIO F LEFT OUTER JOIN FUNCIONARIO S ON
(F.COD_FUNC_SUPERVISOR = S.COD_FUNC)
```

Exemplo 2: **Tabela Repetida** - Para cada departamento, recupere o nome do departamento, o nome do gerente e o total de funcionários que recebem mais de 500 reais lotados atualmente no departamento.

```
SELECT D.NOM_DEPARTAMENTO, G.NOM_FUNC AS GERENTE,
        COUNT(F.COD_FUNC) AS TOTALFUNC
FROM DEPARTAMENTO D LEFT OUTER JOIN FUNCIONARIO G ON
(D.COD_FUNC_GERENTE = G.COD_FUNC) LEFT OUTER JOIN LOTACAO L ON
(D.COD_DEPARTAMENTO = L.COD_DEPARTAMENTO) LEFT OUTER JOIN
FUNCIONARIO F ON (L.COD_FUNC = F.COD_FUNC)
WHERE L.DTA_FIM_LOTACAO IS NULL AND F.VAL_SALARIO > 500
GROUP BY D. COD_DEPARTAMENTO , D.NOM_DEPARTAMENTO, G.NOM_FUNC
```

Exemplo 3: **Cláusula TOP** - Recupere o nome e salário dos 10 funcionários não demitidos com o maior salário na empresa.

```
SELECT TOP 10 NOM_FUNC, VAL_SALARIO FROM FUNCIONARIO
WHERE DTA_DEMISSAO IS NULL
ORDER BY VAL_SALARIO DESC
```

Exemplo 4: **Funções YEAR, MONTH, DAY** - Recupere o nome dos 20 produtos mais vendidos no ano atual.

```
SELECT TOP 20 P.NOM_PRODUTO, SUM(I.NUM_QUANTIDADE) AS TOTALVENDA
FROM PRODUTO P INNER JOIN PEDIDO_ITENS I ON (P.COD_PRODUTO =
I.COD_PRODUTO) INNER JOIN PEDIDO PE ON (I.COD_PEDIDO = PE.COD_PEDIDO)
WHERE YEAR(PE.DTA_PEDIDO) =  YEAR(GETDATE())
GROUP BY P. COD_PRODUTO, P.NOM_PRODUTO
ORDER BY SUM(I.NUM_QUANTIDADE) DESC
```

Exemplo 5: **DISTINCT** – Recupere o nome do projeto que tenha pelo menos um funcionário trabalhando nele.

```
SELECT DISTINCT P.NOM_PROJETO
FROM PROJETO P INNER JOIN TRABALHA_EM T
        ON (P.COD_PROJETO = T.COD_PROJETO)
```

Exemplo 6: **Função Substring, convert, upper, ltrim, rtrim,**

```
SELECT SUBSTRING(NOM_FUNC, 1, 10)
FROM FUNCIONARIO

SELECT CONVERT(CHAR(20), DTA_NASCIMENTO,103)
FROM FUNCIONARIO

SELECT UPPER(LTRIM(RTRIM(NOM_FUNC)))
FROM FUNCIONARIO
```

Exercícios:

1) Recupere o nome e data de nascimento dos funcionários que nasceram no mês de junho.
2) Recupere o nome do projeto, o total de funcionários que não estão demitidos e já trabalharam no projeto e o nome do coordenador do projeto. Mostre todos os projetos não encerrados. A resposta deve estar ordenada pelo nome do projeto.
3) Recupere o nome e quantidade em estoque dos 3 produtos com o menor estoque atual.
4) Conte o número dos diferentes valores de salário contidos na tabela funcionário.
6) Recupere o nome e salário dos funcionários não demitidos que tenham salário maior que o salário de seu supervisor.
7) Recupere o nome dos funcionários supervisores que tenham mais de 40 anos.
8) Recupere o nome e formação escolar dos vendedores que fizeram pedidos no ano de 2007.
9) Usando a função convert, recupere o nome e data de nascimento no formato dd/mm/aaaa (sem exibir hora minutos e segundos) dos funcionários lotados no departamento 2.

4.6. Consultas Avançadas em SQL: Parte II

Exemplo 1: **IS NULL em Outer Joins** - Recupere o nome dos funcionários que não trabalham em nenhum projeto.

```
SELECT F.NOM_FUNC
FROM FUNCIONARIO F LEFT OUTER JOIN TRABALHA_EM T ON (F.COD_FUNC =
T.COD_FUNC)
WHERE T.COD_FUNC IS NULL
```

Exemplo 2: **IS NULL em Outer Joins** - Recupere o nome dos funcionários que não são vendedores, mas são coordenadores de projetos.

```
SELECT F.NOM_FUNC
FROM FUNCIONARIO F INNER JOIN PROJETO P ON
(F.COD_FUNC = P.COD_FUNC_COORDENADOR)
LEFT OUTER JOIN VENDEDOR V ON (F.COD_FUNC = V.COD_FUNC)
WHERE V.COD_FUNC IS NULL
```

Exemplo 3: Recupere o Nome e Salário dos funcionários que recebem o maior salário na empresa.

```
SELECT NOM_FUNC, VAL_SALARIO FROM FUNCIONARIO
WHERE VAL_SALARIO = (SELECT MAX(VAL_SALARIO) FROM FUNCIONARIO)
```

Exemplo 4: Recupere o Nome e salário dos funcionários que ganham abaixo da média salarial dos gerentes de departamento. Mostre o resultado ordenado pelo nome do funcionário.

```
SELECT    NOM_FUNC, VAL_SALARIO FROM FUNCIONARIO
WHERE     VAL_SALARIO <  (SELECT AVG(F.VAL_SALARIO)
          FROM FUNCIONARIO F INNER JOIN DEPARTAMENTO D
          ON F.COD_FUNC = D.COD_FUNC_GERENTE)
ORDER BY NOM_FUNC
```

Exemplo 5: (NOT IN) Recupere nome dos funcionários que não são gerentes:

```
SELECT NOM_FUNC FROM FUNCIONARIO
WHERE COD_FUNC NOT IN (SELECT COD_FUNC_GERENTE FROM
DEPARTAMENTO WHERE COD_FUNC_GERENTE IS NOT NULL)
```

Atenção: Algumas subconsultas podem ser reescritas sem uso de subconsultas. Não existe uma regra geral para indicar qual das duas formas é mais eficiente: avaliar o plano de execução.

Atenção: O resultado de uma subconsulta com o operador IN/NOT IN não pode retornar um valor NULL.

Exemplo 6: Recupere o nome dos funcionários que não são vendedores, mas são coordenadores de projetos.

```
SELECT NOM_FUNC FROM FUNCIONARIO
WHERE COD_FUNC NOT IN (SELECT COD_FUNC FROM VENDEDOR)
AND COD_FUNC IN (SELECT COD_FUNC_COORDENADOR FROM PROJETO
```

```
WHERE COD_FUNC_COORDENADOR IS NOT NULL)
```

Exercícios:

1) Recupere o nome dos funcionários que não são coordenadores de projetos de duas formas distintas:
   a) Sem usar subconsulta, ou seja, usando somente LEFT OUTER JOIN.
   b) Com subconsulta, usando o operador NOT IN na clausula WHERE.
2) Recupere o nome do funcionário não demitido com o menor salário na empresa.
3) Recupere o nome dos gerentes de departamento que ganham acima da média salarial dos funcionários que são vendedores.
4) Recupere os nome dos produtos com a maior quantidade em estoque.
5) Recupere o nome do vendedor que fez o pedido de maior valor até hoje no BD.
6) Recupere o nome dos supervisores que ganham acima da média salarial dos funcionários não demitidos.
7) Use o operador IN para formular uma subconsulta que recupere o nome e data de nascimento de cada um dos empregados que são coordenadores de projetos e tenham idade acima de 30 anos.
8) Refaça a consulta anterior usando somente operador INNER JOIN (Sem Subconsulta).
9) Recupere o nome dos funcionários que são supervisores e coordenadores de projeto, mas não são gerentes de departamentos.

4.7. Consultas Avançadas em SQL: Parte III

Exemplo 1: (EXISTS) Recupere o nome dos projetos que tenham pelo menos um funcionário trabalhando nele.

```
SELECT P.NOM_PROJETO FROM PROJETO P
WHERE EXISTS ( SELECT * FROM TRABALHA_EM T
                    WHERE P.COD_PROJETO = T.COD_PROJETO)
```

Desafio: reescreva a consulta anterior usando-se somente JOIN (Sem o operador EXISTS).

Exemplo 2: (NOT EXISTS) Recupere o nome dos departamentos que não contenha funcionários lotados atualmente.

```
SELECT D.NOM_DEPARTAMENTO FROM DEPARTAMENTO D
WHERE NOT EXISTS (SELECT * FROM LOTACAO L
                        WHERE D.COD_DEPARTAMENTO = L.COD_DEPARTAMENTO
                        AND L.DTA_FIM_LOTACAO IS NULL)
```

Desafio: reescreva a consulta anterior com NOT IN. É possível reescrever esta query usando-se somente JOIN ?
PS: Não existe uma regra geral para indicar qual das formas é mais eficiente: deve-se avaliar o plano de execução.

Exercícios:

1) Observe a seguinte consulta SQL:

```
SELECT NOM_FUNC FROM FUNCIONARIO
WHERE DTA_DEMISSAO IS NULL AND COD_FUNC NOT IN (SELECT COD_FUNC_GERENTE FROM DEPARTAMENTO
WHERE COD_FUNC_GERENTE  IS NOT NULL)
```

a) Explique o que esta consulta faz.
b) Reescreva a consulta usando o operador NOT EXISTS.
c) Reescreva a consulta sem uso de subconsulta.

2 O que a seguinte consulta faz?

```
SELECT F.NOM_FUNC
FROM FUNCIONARIO F INNER JOIN DEPARTAMENTO D ON (F.COD_FUNC = D.COD_FUNC_GERENTE)
WHERE NOT EXISTS (SELECT * FROM PROJETO P WHERE F.COD_FUNC = P.COD_FUNC_COORDENADOR) AND
F.VAL_SALARIO > 1000 AND F.COD_FUNC_SUPERVISOR IS NULL
```

a) Reescreva a consulta usando-se o operador NOT IN.
b) Reescreva a consulta sem uso de subconsulta. Porque ela é mais rápida que as anteriores?
c) Se no lugar de INNER JOIN fosse escrito LEFT OUTER JOIN o resultado final da consulta seria alterado? Justifique.

3.1 Usando o operador NOT IN recupere o nome dos vendedores que não são coordenadores de projetos.
3.3 Refaça a consulta anterior usando o operador NOT EXISTS.
3.3 Refaça a consulta anterior sem usar subconsulta.