Pós-Graduação em Inteligência Artificial Generativa e LLMs

PUC-Rio

Oficinas Práticas

Notas de Aula

Vítor Wilher

OFICINAS · 2025.2

Versão 1.0

Resumo. Atividades práticas que acompanham as disciplinas, com exercícios guiados e gabaritos comentados.

Palavras-chave: oficinas; prática; exercícios

Notas de estudo elaboradas a partir do material das aulas.

Índice

1 Visão Geral da Disciplina

As Oficinas constituem o componente prático do curso. Diferentemente de disciplinas expositivas, o formato privilegia o trabalho hands-on: o participante recebe um enunciado de caso, uma ferramenta e um roteiro, e produz um artefato concreto — um modelo conceitual de dados, um script SQL, uma consulta analítica ou um pipeline de visão computacional. O material reunido nesta apostila cobre três frentes previstas na ementa: **OFICINA - PAI**, **OFICINA - DPIA** e **OFICINA - TDP**.

O maior volume de material pertence à trilha **TDP (Transformando Dados em Percepção)**, ministrada pelo Prof. Anderson Nascimento (prof.anderson@ica.ele.puc-rio.br). Essa trilha percorre, em quatro encontros práticos, o ciclo de vida de um banco de dados relacional: parte da modelagem conceitual com o modelo Entidade-Relacionamento, passa pela tradução para o modelo lógico e físico, chega à criação de estruturas em SQL (DDL), à manipulação de dados (DML) e culmina em consultas analíticas avançadas (DQL) com funções de agregação, subconsultas e visões. O fio condutor declarado nos slides é explícito: dados são o combustível da Inteligência Artificial, e SQL é a chave para preparar e acessar os dados que alimentam modelos de IA.

A trilha **PAI** aparece com uma oficina voltada a visão computacional aplicada — segmentação de instâncias com Mask R-CNN e reconhecimento óptico de caracteres (OCR) —, acompanhada de um artigo científico sobre classificação sob rótulos ruidosos, leitura de fundo para os desafios de qualidade de dados em aprendizado profundo.

O encadeamento segue uma lógica de maturidade de dados. Primeiro, estrutura-se o dado (modelagem e SQL). Depois, extrai-se percepção a partir dele (consultas analíticas e agregações). Por fim, aplicam-se modelos de aprendizado profundo sobre dados não estruturados, quando surge o problema — central em aplicações reais — da qualidade das anotações. A mensagem que atravessa o conjunto é a mesma de um dos slides de abertura: "IA poderosa nasce de bases de dados bem estruturadas".

2 OFICINA - PAI

Esta oficina, registrada no material como a sessão dedicada a **Mask R-CNN e OCR**, trabalha com aprendizado profundo aplicado a imagens e documentos. O acervo de apoio da aula é composto por notebooks e imagens de trabalho: `dilation_erosao.ipynb`, `pytesseract.ipynb`, `doctr.ipynb`, `Mask_RCNN.ipynb`, uma apresentação `OCR.pptx`, e as imagens de teste `1dog.jpg`, `Rletra.png` e uma conta de luz digitalizada.

Material de slides de OCR e Mask R-CNN não disponível no contexto; o conteúdo abaixo cobre o material efetivamente presente, que é o artigo científico anexado à aula.

2.1 Estrutura prática sugerida pela oficina

A sequência de notebooks indica um percurso prático em quatro etapas:

1. **Pré-processamento morfológico** (`dilation_erosao.ipynb`): operações de dilatação e erosão sobre imagens binarizadas, etapa clássica de limpeza antes do reconhecimento de caracteres.

2. **OCR com Tesseract** (`pytesseract.ipynb`): extração de texto de imagens usando o motor Tesseract via interface Python, testado sobre uma imagem de letra isolada (`Rletra.png`) e sobre um documento real (conta de luz).
3. **OCR com docTR** (`doctr.ipynb`): abordagem baseada em aprendizado profundo para detecção e reconhecimento de texto em documentos, contrastando com o Tesseract.
4. **Segmentação de instâncias** (`Mask_RCNN.ipynb`): aplicação da arquitetura Mask R-CNN sobre a imagem `1dog.jpg`, produzindo máscaras por instância em vez de apenas caixas delimitadoras.

O par OCR + segmentação é representativo do que a oficina PAI se propõe: mostrar que "percepção artificial" combina técnicas clássicas de processamento de imagem com redes profundas, e que a escolha entre elas depende do problema.

2.2 Rótulos ruidosos: o problema de qualidade de dados em aplicações reais

O material de leitura da oficina é o artigo *Classification of calcareous algae under noisy labels*, de Vitor Bento, Manoela Kohler e Marco Aurelio Pacheco, do Departamento de Engenharia Elétrica da PUC-Rio, publicado em *Neural Computing and Applications*. Ele documenta um caso real de monitoramento ambiental e é útil na oficina porque ilustra o que acontece quando o dado de treinamento não é perfeito.

O problema. Algas calcárias formam um ecossistema marinho relevante na costa brasileira, ameaçado pelo aquecimento global e por estressores locais como a extração e o transporte de petróleo e gás offshore. Uma empresa brasileira de petróleo passou a monitorar esse ambiente com imagens do leito marinho coletadas por um **ROV** (*remotely operated vehicle*), veículo que envia imagens em tempo real à embarcação por cabo umbilical. Para escalar o monitoramento a toda a costa, construiu-se um classificador de aprendizado profundo com quatro classes, distinguidas por morfologia: Granulado (nódulos pequenos, até 3 cm de diâmetro), Rodolito (nódulos esféricos acima de 3 cm), Laje (formação maciça, contínua ou intercalada com sedimento) e Bioconcreção (semelhante à Laje, porém com maior complexidade tridimensional, sobretudo em altura).

O ruído. Mais de 1 milhão de imagens foram coletadas, mas apenas 8.550 foram anotadas. Parte das anotações foi feita por não especialistas, o que introduziu **rótulos ruidosos** (*noisy labels*) — amostras rotuladas incorretamente, que degradam a robustez do modelo. Em aplicações reais, ao contrário dos benchmarks, o percentual de ruído não é conhecido de antemão.

Estimativa do ruído. O protocolo adotado foi: separar 10% das amostras de cada classe, submeter essas amostras à revisão de dois especialistas, registrar e corrigir os erros encontrados, e usar a taxa observada como estimativa estatística do ruído. O resultado apontou ruído de 17%. O subconjunto revisado tornou-se o conjunto de teste; o restante, o conjunto de treino. Não houve subdivisão em validação, pois validar sobre um conjunto ruidoso não faz sentido.

Um achado importante: o ruído real não se distribui como o ruído sintético usado em benchmarks (simétrico ou *pair flip*). Neste caso, os erros ocorreram predominantemente entre as classes Rodolito e Laje, justamente as mais próximas visualmente para um anotador leigo.

2.3 A abordagem small loss e o framework RDS

A família de métodos estado-da-arte para rótulos ruidosos apoia-se na **abordagem small loss** (*small loss approach*, SLA). A intuição é que modelos profundos aprendem primeiro as instâncias fáceis e corretas, que produzem perdas menores, e só depois se ajustam às instâncias difíceis ou ruidosas. Logo, descartar do treino as amostras com maior perda de entropia cruzada tende a selecionar um subconjunto limpo.

Três modelos citados exploram essa ideia:

- **Co-teaching**: duas redes pares fazem predições sobre o mesmo mini-batch; cada uma seleciona, por small loss, as amostras limpas que serão usadas para treinar a outra.
- **Co-teaching+**: incorpora a ideia de *Decoupling*, separando a decisão de "quando atualizar" da de "como atualizar"; mantém apenas as amostras em que as duas redes discordam na predição.
- **Jocor**: usa uma perda conjunta composta pela entropia cruzada de cada rede mais um termo de redução de divergência entre os classificadores, medido pela divergência de Jensen-Shannon implementada via divergência de Kullback-Leibler. Funciona como rede siamesa.

A crítica dos autores. Descartar amostras é perder informação: as excluídas podem conter características importantes da classe a que pertencem, e conjuntos com muito ruído sofrem mais com essa perda. Daí a proposta do framework **RDS** (*Retrieving Discard Samples*): em vez de descartar, atribuir um **pseudo-rótulo** às amostras identificadas como ruidosas e devolvê-las ao treino ao fim de cada época.

O pseudo-rótulo é obtido por *Label Guessing*: calcula-se a média das predições do modelo sobre N versões aumentadas da mesma amostra (espelhamentos horizontais, recortes). O pseudo-rótulo só é válido se tiver baixa entropia — isto é, se a saída softmax da classe superar um limiar — e se ambas as redes concordarem. A perda final é a entropia cruzada das amostras limpas mais uma pseudo-perda ponderada. Sob condições ideais, o treinamento com essa perda é matematicamente equivalente a treinar com entropia cruzada sobre um conjunto sem nenhum ruído.

Dois modelos foram propostos: **RDS-C** (RDS sobre Co-teaching+) e **RDS-J** (RDS sobre Jocor).

2.4 Resultados e lições práticas

Os experimentos usaram Cifar-10, Cifar-100 e Mnist com ruído sintético (*pair flip* de 45%, simétrico de 20% e simétrico de 50%), além do conjunto real Clothing1M com 1 milhão de imagens e 40% de ruído. O setup dos benchmarks foi: taxa de aprendizado 0,001, otimizador Adam com momentum 0,9, 200 épocas, uma aumentação por amostra, limiar de 0,80, implementação em TensorFlow 2.4 e 20 repetições por experimento.

A lição mais robusta é comparativa: **RDS-J foi sempre superior a Jocor, e RDS-C sempre superior a Co-teaching+**, em todos os experimentos. Em Cifar-10 com ruído simétrico de 20%, RDS-J atingiu 0,7160 de acurácia média nas últimas dez épocas contra 0,7042 de Jocor e 0,6054 do modelo padrão. Em Mnist, RDS-C liderou em todas as condições.

Um resultado contraintuitivo merece atenção didática: em Cifar-100 com *pair flip* de 45%, RDS-C obteve melhor "RDS Accuracy" (proporção de pseudo-rótulos corretos) mas RDS-J teve melhor

acurácia de teste. A robustez intrínseca do modelo base ao ruído continua importante, e não apenas a qualidade da pseudo-rotulagem.

Sobre os hiperparâmetros, há um trade-off claro: aumentar o número de aumentações ou o limiar eleva a precisão da pseudo-rotulagem mas reduz drasticamente o número de amostras recuperadas. Em Cifar-100 com limiar 0,80, passar de 1 para 4 aumentações elevou a RDS Accuracy de 0,3611 para 0,7178, mas derrubou as amostras recuperadas de cerca de 20.259 para 484 — e a acurácia de teste caiu de 0,3077 para 0,2618. A recomendação prática é limiar de 0,8 com uma ou duas aumentações.

Na aplicação real, usou-se ResNet-50 pré-treinada em ImageNet, Adam com momentum 0,9, batch de 32, 80 épocas, taxa de aprendizado de 1e-5, redimensionamento para 256 por 256 e recorte central de 224 por 224. O modelo escolhido foi o RDS-C, com F1-Score de 0,8450 contra 0,8139 do modelo padrão — ganho de 3,5%. Por ser um conjunto desbalanceado, a métrica de decisão foi o F1-Score, e o uso de entropia cruzada ponderada melhorou o desempenho em mais 1,2%.

A conclusão metodológica é diretamente aplicável ao trabalho prático: em problemas reais, os dados raramente sofrem só de ruído de rótulo. Costumam vir desbalanceados e com outras patologias. Portanto, uma técnica de tratamento de ruído precisa ser **simples e composável** com outras técnicas, evitando hiperparâmetros difíceis de ajustar e dependentes do conjunto de dados.

3 OFICINA - DPIA

Material de slides não disponível para esta oficina.

A ementa registra a **OFICINA - DPIA** como um dos três eixos práticos da disciplina, mas o acervo de slides fornecido não contém material atribuído a essa oficina. Não há, portanto, base documental para descrever seu roteiro, suas ferramentas ou seus exercícios sem recorrer a especulação. O conteúdo efetivamente registrado nas cinco aulas do contexto distribui-se entre a trilha TDP (quatro encontros) e a trilha PAI (um encontro).

4 OFICINA - TDP

A trilha **TDP — Transformando Dados em Percepção** organiza-se em sete tópicos anunciados desde a aula de apresentação:

1. Modelagem de Bancos de Dados Relacionais
2. Modelagem Lógica e Física de Bancos Relacionais
3. SQL para Criação de Estruturas (DDL)
4. SQL para Manipulação de Dados (DML)
5. SQL para Consulta de Dados (DQL)
6. SQL Avançado — Joins e Combinações Complexas
7. Funções, Subqueries e Visões

Os objetivos declarados são conhecer a linguagem SQL, criar objetos com DDL, manter dados com DML, praticar consultas com DQL, realizar junções, construir subconsultas, aplicar funções, criar visões e relacionar bancos de dados à Inteligência Artificial.

A motivação é apresentada de forma encadeada nos slides: dados são o combustível da IA; bancos relacionais garantem qualidade, consistência e integridade; SQL é a chave para preparar e acessar os dados usados em IA; SQL organiza e estrutura, a IA interpreta e gera recomendações; juntos, transformam dados em ações inteligentes.

4.1 Ferramentas da oficina

A oficina é executada com um conjunto pequeno e gratuito de ferramentas:

- **SGBD**: PostgreSQL (download em enterprisedb.com).
- **Modelagem**: BR Modelo 2.0 (versão Delphi), BR Modelo 3.x (versão Java) e BR Modelo Web.
- **Prática online**: sqliteonline.com, para quem não quiser instalar nada.

Outras ferramentas citadas na bibliografia de links incluem MySQL Workbench, ERWIN Data Modeler, Power Architect e DBeaver.

O case narrativo proposto para amarrar a trilha é um sistema estilo "Netflix": simular a descoberta de indicações de filmes a partir dos filmes assistidos por um usuário de streaming, usando bancos relacionais.

4.2 Passo 1 — Conceitos fundamentais antes de modelar

A oficina começa fixando vocabulário, porque sem ele a modelagem vira adivinhação.

Dados são fatos conhecidos que podem ser armazenados e que possuem significado implícito. Um **banco de dados** é uma coleção logicamente coerente de dados relacionados — uma reunião aleatória de informações não é um banco de dados. Ele representa algum aspecto do mundo real, chamado **mini-mundo** ou Universo do Discurso, e mudanças nesse mini-mundo se refletem no banco. É projetado, construído e povoado para um propósito específico, com um grupo definido de usuários, e mantido em armazenamento secundário.

A **pirâmide do conhecimento** é usada como organizador conceitual, com exemplos meteorológicos:

- **Dado**: elementos não interpretados. "12 graus Celsius", "85 km/h", "90% de umidade relativa".
- **Informação**: dado com significado num contexto, após processamento. "40 graus Celsius no Rio de Janeiro", "85 km/h de rajadas em Copacabana".
- **Conhecimento**: interpretação formal das relações entre dados e informação; informação organizada, ou expertise. "Se a umidade relativa está em 95%, vai chover".
- **Sabedoria**: integração e evolução de múltiplos domínios de conhecimento ao longo do tempo, permitindo prever tendências e antecipar proativamente problemas causados pelo mau tempo.

Três conceitos completam a base:

- **Metadados**: o SGBD guarda não só os dados, mas a descrição completa de como estão armazenados — estrutura de cada arquivo, tipo e formato de cada dado, restrições. Essas informações vivem no catálogo do SGBD.

- **Esquema**: a descrição do banco de dados, especificada durante o projeto. Muda pouco ao longo do tempo.
- **Instância**: os dados armazenados num determinado instante. Muda a cada alteração, e o SGBD garante que toda instância satisfaça o esquema.

Um **modelo de dados** é um conjunto de conceitos usados para descrever a estrutura conceitual, lógica e física de um banco — onde "estrutura" abrange tipos de dados, relacionamentos e restrições.

O **SGBD**, por sua vez, é uma coleção de programas de uso geral que permite definir (especificar tipos, estruturas e restrições), construir (armazenar os dados no meio físico) e manipular (buscar, modificar, gerar relatórios) bancos de dados. Entre as características desejáveis, os slides listam as propriedades **ACID** (Atomicidade, Consistência, Isolamento e Durabilidade), persistência de objetos, controle de redundância, manutenção de restrições de integridade, representação de relacionamentos complexos, restrições de acesso, múltiplas interfaces de usuário, inferências por regras de dedução, e rotinas de backup e recuperação.

4.3 Passo 2 — Modelagem conceitual com o modelo Entidade-Relacionamento

O **Modelo Entidade-Relacionamento**, proposto por Peter Chen, parte de um princípio enunciado em uma frase: "O mundo está cheio de coisas que possuem características próprias e que se relacionam umas com as outras."

Essa frase se decompõe nos três elementos do modelo:

- "coisas" -> **entidades**: os objetos que fazem parte da situação modelada.
- "características próprias" -> **atributos**: as características dos objetos.
- "se relacionam umas com as outras" -> **relacionamentos**: as associações possíveis entre os objetos.

Entidades. Uma entidade é um conjunto de objetos com os mesmos tipos de características — Máquina representa todas as máquinas de uma fábrica; Funcionário, todos os funcionários de uma empresa. Cada objeto do conjunto é uma **instância**. Para reconhecer entidades, a oficina recomenda a heurística de cinco grupos de Shlaer e Mellor (*Object-Oriented Systems Analysis*): coisas tangíveis (Meio de Transporte, Equipamento); funções exercidas por elementos (Especialista, Cliente, Atendente); eventos ou ocorrências (Voo comercial, Acidente de trânsito); interações (Compra de imóvel, Venda realizada por um fornecedor); e especificações (Modelo de refrigerador). O nível de abstração adotado é decisão de projeto, não regra fixa.

Atributos. Um atributo é um tipo de característica comum a todas as instâncias de uma entidade — e identificar atributos comuns é justamente o que permite agrupar objetos numa entidade. A classificação se dá em quatro eixos:

1. **Simples (atômicos) versus Compostos**: simples não podem ser divididos; compostos podem ser divididos em subpartes.
2. **Monovalorados versus Multivalorados**: monovalorados têm um único valor por instância; multivalorados podem ter vários.
3. **Armazenados versus Derivados**: armazenados têm valor próprio; derivados são obtidos a partir de outros atributos.

4. **Chave (identificador) versus Não-chave**: um atributo chave possui valor distinto para cada instância.

O exemplo canônico é a entidade Pessoa: CPF é chave; Endereço é composto (Cidade, Bairro, Rua); Telefone é multivalorado; Data de Nascimento é armazenada e Idade é derivada dela; Tipo Sanguíneo é simples e monovalorado.

Relacionamentos. Um relacionamento representa um tipo de associação possível entre instâncias de entidades diferentes ou de uma mesma entidade — o autorrelacionamento aparece em Empregado-Supervisiona-Empregado, com os papéis Supervisor e Supervisionado. **Nomes de papel** especificam a função de uma entidade no relacionamento quando há dúvida na interpretação: em Departamento-TrabalhaPara-Funcionário, o papel "trabalhador" deixa claro quem trabalha para quem.

Atributos de relacionamento cobrem informações extras que não pertencem a nenhuma das entidades envolvidas porque variam conforme a combinação das instâncias. O caso clássico: um empregado trabalha em vários projetos e um projeto tem vários empregados; como o número de horas varia de empregado para empregado e de projeto para projeto, "Número de Horas" é atributo do relacionamento Trabalha Em.

Multiplicidade. É a composição de **participação** e **cardinalidade**, notada como par ordenado.

- A **cardinalidade** é o número máximo de relacionamentos daquele tipo dos quais cada instância pode participar, representada do lado oposto ao da entidade a que se refere. Os tipos são 1:1, 1:N e M:N. Usa-se M ou N quando o máximo é indefinido; se conhecido, usa-se o número — "cada empregado pode supervisionar no máximo cinco projetos" vira cardinalidade 5.
- A **participação** é o número mínimo, indicando se o relacionamento é obrigatório ou opcional. Participação **total (obrigatória)** significa que todas as instâncias devem ter tal relacionamento, e se representa por 1; **parcial (opcional)** admite instâncias sem nenhum, e se representa por 0.

O exemplo progressivo é ilustrativo. De "uma pessoa pode possuir vários automóveis; cada automóvel pertence a apenas uma pessoa", chega-se à cardinalidade N e 1. Acrescentando "uma pessoa pode não possuir automóveis; todo automóvel pertence a uma pessoa", chega-se a (0,N) do lado Pessoa e (1,1) do lado Automóvel.

4.4 Passo 3 — Especialização e generalização

Às vezes, apenas algumas instâncias de uma entidade compartilham características adicionais, formando subconjuntos. Esses subconjuntos são **subclasses** (especialização); a entidade de origem é a **superclasse** (generalização). As instâncias de uma subclasse possuem todas as características da superclasse acrescidas das suas próprias. **Especialização** é o processo *top-down* de identificar subclasses a partir de uma entidade genérica; **generalização** é o inverso, *bottom-up*.

O exemplo de Cliente é o mais completo: todos os clientes possuem número único de identificação e telefones; apenas clientes pessoa física possuem CPF único e nome; apenas clientes pessoa jurídica possuem CNPJ único, razão social e nome de contato.

Duas restrições qualificam a hierarquia:

Totalidade. Especialização **total** ocorre quando todas as instâncias da superclasse pertencem a alguma subclasse — representada por linha dupla ligando a superclasse ao círculo da relação. Especialização **parcial** admite instâncias que não pertencem a nenhuma subclasse — linha simples. Exemplo total: todo Colaborador é CLT ou PJ. Exemplo parcial: alguns Alunos não são bolsistas nem monitores.

Exclusividade. A restrição **exclusiva** impede que uma instância pertença a mais de uma subclasse ao mesmo tempo (um Veículo Automotor é Carro, Caminhão ou Trator, nunca dois). A **não-exclusiva** permite pertencimento simultâneo (um Aluno pode ser bolsista e monitor).

4.5 Passo 4 — Exercícios de modelagem conceitual

A oficina propõe uma bateria de casos para elaborar modelos conceituais no BR Modelo.

Exercício 01 — Empresa. Uma empresa quer cadastrar sua rotina administrativa. Deve permitir cadastro de funcionários com nome, CPF, endereço, telefones, cargo e salário. Também o cadastro dos departamentos, com nome, localização (bloco, andar e sala) e sigla. A sigla tem 3 caracteres e é única por departamento, e o funcionário só pode ser alocado em um único departamento. Cada funcionário pode estar alocado em vários projetos; de cada projeto armazenam-se nome, código, descrição e nome do gerente. Um projeto pode ter vários funcionários e, a cada associação de funcionário a projeto, devem-se armazenar a data de início e a quantidade de horas.

Este enunciado exercita simultaneamente atributos compostos (endereço, localização), multivalorados (telefones), cardinalidade 1:N (departamento e funcionário) e um relacionamento M:N com atributos próprios (funcionário e projeto).

Exercício 02 — Gerenciamento de Atendimento. Uma empresa de TI quer cadastrar os atendimentos aos clientes, que podem ser internos ou externos. Para cliente interno, armazenam-se nome, e-mail corporativo, telefones, o departamento em que está alocado e os dados do departamento. Para cliente externo, o nome da empresa, telefones, endereço, e-mail e o nome do contato principal. Cada empresa possui um ou mais contratos, e cada contrato possui seu conjunto de SLAs (acordos de nível de serviço) — data de início, data fim, tempo de atendimento e valor da multa. Todo cliente tem um número distinto e uma categoria (especial ou normal) definida por seu perfil.

O ponto central aqui é a generalização: Cliente como superclasse, com Cliente Interno e Cliente Externo como subclasses, e os atributos comuns (número distinto e categoria) na superclasse.

Micro casos de fixação. Seis casos curtos treinam padrões específicos:

- *Clínica de Exames*: Pacientes e Exames em relacionamento M:N. Guardar nome, CPF e data de nascimento do paciente, tipo de exame e valor cobrado.
- *Biblioteca Digital*: cada usuário pode pegar vários livros, mas cada livro só pode estar com um usuário por vez — 1:N. Guardar nome e e-mail do usuário; título, autor e ano do livro.
- *Curso Online de IA*: Cursos, Professores e Alunos. A nota final de cada aluno em cada curso é atributo do relacionamento.
- *Loja Virtual*: Clientes, Pedidos e Produtos. Quantidade e preço unitário no momento da compra são atributos do relacionamento entre pedido e produto — detalhe importante, porque o preço do produto muda ao longo do tempo.
- *Funcionários de Empresa de Tecnologia*: especialização em Desenvolvedores (linguagem principal) e Gerentes (número de equipes), sobre atributos comuns de nome, matrícula e salário.

- *Veículos em uma Locadora*: especialização de Veículo em Carros (quantidade de portas) e Motos (cilindrada), sobre placa, modelo e ano de fabricação.

Exercício extra — Negociações na Bolsa de Valores. Caso estendido, próximo de um problema real de engenharia de dados para IA. Uma corretora quer gerenciar Investidores (CPF ou CNPJ, nome, tipo, e-mail, telefone), Ações (pertencentes a uma Empresa listada, com ticker, setor e valor de mercado) e Negociações de compra ou venda, que exigem data e hora, tipo de operação, quantidade e valor unitário no momento — um relacionamento M:N com atributos. Também é preciso manter o Histórico de Cotações, com data, hora e valor por ação, permitindo análises de séries temporais, e o Saldo de Carteira de cada investidor, atualizado a partir das negociações.

Este é o exercício que mais explicitamente conecta a modelagem relacional a aplicações de IA: o histórico de cotações é precisamente a estrutura que alimenta modelos de séries temporais.

4.6 Passo 5 — Da modelagem conceitual à DDL

Uma vez validado o modelo conceitual e derivado o modelo lógico, a oficina passa à criação física das estruturas no PostgreSQL. O roteiro da Oficina 03 é explícito: a partir de um modelo lógico dado, criar as tabelas com um conjunto de restrições de integridade.

As restrições pedidas são:

- Na tabela Funcionário, o nome não pode ser nulo.
- Na tabela Funcionário, o gênero deve aceitar apenas as letras M e F.
- O estado civil deve aceitar apenas os valores C, S, V e D.
- O salário deve ser maior que o salário mínimo.
- Na tabela Projeto, o orçamento deve ser maior que zero.
- Na tabela Projeto, o nome é obrigatório e não pode se repetir.
- Na tabela Trabalha, a data de alocação deve ter como valor padrão a data do dia da alocação, usando `CURRENT_DATE`.

Traduzido para DDL, o exercício exercita os quatro mecanismos declarativos de integridade do SQL:

```
CREATE TABLE funcionario (
    codfunc      INTEGER PRIMARY KEY,
    nome         VARCHAR(40) NOT NULL,
    genero       CHAR(1) CHECK (genero IN ('M','F')),
    estcivil     CHAR(1) CHECK (estcivil IN ('C','S','V','D')),
    dtadmissao   DATE,
    cargo        VARCHAR(40),
    salario      NUMERIC(10,2) CHECK (salario > 1412.00)
);

CREATE TABLE proj (
    codproj      INTEGER PRIMARY KEY,
    nome         VARCHAR(40) NOT NULL UNIQUE,
    chproj       INTEGER,
    orcamento    NUMERIC(12,2) CHECK (orcamento > 0)
);

CREATE TABLE trabalha (
```

```
    codfunc      INTEGER REFERENCES funcionario(codfunc),
    codproj      INTEGER REFERENCES proj(codproj),
    dtalocacao   DATE DEFAULT CURRENT_DATE,
    PRIMARY KEY (codfunc, codproj)
);
```

Os quatro mecanismos são: `NOT NULL` para obrigatoriedade, `CHECK` para domínios de valor e regras de negócio simples, `UNIQUE` para unicidade fora da chave primária, e `DEFAULT` para valores automáticos. A chave estrangeira (`REFERENCES`) garante a integridade referencial.

4.7 Passo 6 — Evolução do esquema com ALTER

O segundo bloco da Oficina 03 é sobre evolução de esquema — algo inevitável em qualquer projeto real. As alterações pedidas foram: renomear `nome` para `nomefunc` e ajustar seu tipo para `VARCHAR(50)`; ajustar `estcivil` para `CHAR(1)` e `cargo` para `VARCHAR(50)`; renomear a tabela `proj` para `projeto` e seu campo `nome` para `nomeproj`; inserir `NOT NULL` em `nomeproj`; apagar o campo `chproj`; e inserir o campo `dtfim` na tabela Trabalha.

```
ALTER TABLE funcionario RENAME COLUMN nome TO nomefunc;
ALTER TABLE funcionario ALTER COLUMN nomefunc TYPE VARCHAR(50);
ALTER TABLE funcionario ALTER COLUMN estcivil TYPE CHAR(1);
ALTER TABLE funcionario ALTER COLUMN cargo TYPE VARCHAR(50);

ALTER TABLE proj RENAME TO projeto;
ALTER TABLE projeto RENAME COLUMN nome TO nomeproj;
ALTER TABLE projeto ALTER COLUMN nomeproj SET NOT NULL;
ALTER TABLE projeto DROP COLUMN chproj;

ALTER TABLE trabalha ADD COLUMN dtfim DATE;
```

A oficina distingue quatro operações de `ALTER TABLE`: renomear (tabela ou coluna), alterar tipo, alterar restrição e adicionar ou remover coluna.

4.8 Passo 7 — Manipulação de dados com DML

Com o esquema estabilizado, insere-se a carga de dados. A tabela Funcionário recebe cinco registros (Ana, Bruna, Carla, Danilo e Elias, com gênero, estado civil, data de admissão, cargo e salário); a tabela Projeto recebe três registros — LGPD com orçamento de 95.000,00, Business Intelligence com 220.000,00 e ITIL com 150.000,00; e a tabela Trabalha recebe seis alocações, quatro em aberto (com `dtfim` nulo) e duas encerradas em 10/05/2023.

```
INSERT INTO funcionario
    (codfunc, nomefunc, genero, estcivil, dtadmissao, cargo, salario)
VALUES
    (1, 'Ana',    'F', 'C', '2020-11-03', 'Programador',          5000.00),
    (2, 'Bruna',  'F', 'C', '2020-11-03', 'Analista de Sistemas', 8500.00),
    (3, 'Carla',  'F', 'S', '2021-05-04', 'Programador',          5000.00),
    (4, 'Danilo', 'M', 'D', '2021-05-04', 'Programador',          5000.00),
    (5, 'Elias',  'M', 'S', '2021-05-04', 'Analista de Sistemas', 8500.00);

INSERT INTO projeto (codproj, nomeproj, orcamento) VALUES
```

```
    (1, 'LGPD',                  95000.00),
    (2, 'Business Intelligence', 220000.00),
    (3, 'ITIL',                  150000.00);

INSERT INTO trabalha (codfunc, codproj, dtalocacao, dtfim) VALUES
    (1, 2, '2023-01-03', NULL),
    (1, 3, '2023-06-10', NULL),
    (4, 1, '2023-01-02', '2023-05-10');
```

O quarto item do roteiro é o mais didático de todos: **tentar realizar inserções que firam as integridades definidas no banco de dados**. É um exercício de falha deliberada — inserir gênero 'X', salário abaixo do mínimo, orçamento negativo, nome de projeto duplicado, ou uma alocação para um código de funcionário inexistente. O objetivo é ver o SGBD rejeitar a operação e ler a mensagem de erro, entendendo na prática que integridade declarada no esquema é integridade garantida pelo motor, não pela aplicação.

4.9 Passo 8 — Funções de agregação

A última oficina da trilha aborda funções, subconsultas e visões. As principais funções apresentadas são:

- `SUM` — soma os valores de uma coluna a partir de uma consulta;
- `COUNT` — conta a quantidade de linhas resultantes;
- `MAX` — recupera o valor máximo de uma coluna;
- `MIN` — recupera o valor mínimo;
- `AVG` — recupera a média de um determinado valor.

Os exercícios 63 a 70 aplicam essas funções sobre o esquema construído: recuperar o orçamento do projeto mais caro; o salário mais baixo; a maior e a menor quantidade de horas alocadas; a média salarial de todo o quadro funcional; a quantidade de funcionários do sexo masculino alocados em projetos externos; o total de horas alocadas no projeto Governança de Dados; o valor médio de horas alocadas em todos os projetos; e a idade do funcionário mais velho.

```
SELECT MAX(orcamento) FROM projeto;
SELECT MAX(qthoras), MIN(qthoras) FROM trabalha;
SELECT AVG(salario)   FROM funcionario;
```

4.10 Passo 9 — Agrupamento com GROUP BY e HAVING

Agregações ganham poder quando combinadas com agrupamento. A cláusula `GROUP BY` particiona o resultado e aplica a função a cada partição:

```
SELECT COUNT(*), sexo
FROM funcionarios
GROUP BY sexo;
```

Essa consulta retorna o número de funcionários do sexo masculino e o número do sexo feminino.

A cláusula `HAVING` é o filtro aplicado dentro das consultas de agregação — isto é, sobre o resultado do agrupamento, e não sobre as linhas originais:

```sql
SELECT COUNT(*), coddepto
FROM funcionarios
GROUP BY coddepto
HAVING COUNT(*) > 50;
```

Nesse caso, a consulta agrupa a quantidade de funcionários por departamento e retorna apenas os departamentos com mais de 50 funcionários. A distinção entre `WHERE` (filtra linhas antes do agrupamento) e `HAVING` (filtra grupos depois) é um dos pontos mais confundidos por iniciantes e por isso merece atenção.

Os exercícios 70 a 73 exercitam o padrão: recuperar o número de funcionários por estado civil em ordem decrescente; exibir apenas os dois estados civis com mais pessoas, usando `LIMIT`; recuperar a quantidade de funcionários por cargo; e recuperar o total de horas alocadas por projeto, exibindo o nome do projeto e apenas aqueles com mais de 200 horas de alocação — este último combinando junção, agrupamento e `HAVING` numa só consulta.

4.11 Passo 10 — Subconsultas

Uma **subconsulta** é uma consulta dentro de outra consulta. A oficina apresenta dois padrões.

O primeiro é a subconsulta na cláusula `FROM`, usada quando é preciso agregar sobre um resultado já agregado:

```sql
SELECT MAX(totmultas)
FROM (
    SELECT COUNT(*) AS totmultas, placa
    FROM ocorrencia
    GROUP BY placa
) AS r;
```

A consulta interna conta multas por placa; a externa recupera o maior desses totais. Não é possível aninhar `MAX(COUNT(*))` diretamente, o que torna a subconsulta necessária.

O segundo padrão usa `IN` e `NOT IN` para pertencimento a conjunto, tipicamente para encontrar o que *não* ocorreu:

```sql
SELECT nomeproprietario
FROM proprietario
WHERE nomeproprietario NOT IN (
    SELECT DISTINCT p.nomeproprietario
    FROM ocorrencia o, carro c, proprietario p
    WHERE o.placa = c.placa
      AND c.idproprietario = p.idproprietario
)
ORDER BY nomeproprietario;
```

Aqui a subconsulta constrói a lista de proprietários que tiveram multas, e a consulta externa devolve o complemento: os que não tiveram.

Os exercícios 74 a 77 seguem essa lógica: recuperar os nomes dos funcionários que não se alocaram em nenhum projeto; a descrição do projeto de maior orçamento; os nomes de projeto cuja quantidade de horas de alocação está acima da média histórica; e o valor total de orçamento dos projetos que nunca foram alocados.

4.12 Passo 11 — Visões e visões materializadas

Uma **visão** (*view*) é uma representação virtual de dados provenientes de uma ou mais tabelas. Ela não armazena dados fisicamente: guarda uma consulta que é executada sempre que a visão é acessada.

```
CREATE VIEW nome_da_visao AS
SELECT colunas
FROM tabela_ou_tabelas
WHERE condicoes;
```

Uma **visão materializada** (*materialized view*) é semelhante, mas com uma diferença importante: armazena fisicamente os dados resultantes da consulta, criando uma cópia física em tabela. Isso pode melhorar significativamente o desempenho de consultas complexas, especialmente em sistemas com grandes volumes de dados. A contrapartida é que, na maioria dos SGBDs, a visão materializada não é atualizada automaticamente por padrão, embora seja possível configurar atualização automática ou programada.

```
CREATE MATERIALIZED VIEW nome_da_visao_materializada AS
SELECT colunas
FROM tabelas
WHERE condicoes;

REFRESH MATERIALIZED VIEW nome_da_visao_materializada;
```

O trade-off é claro e vale ser internalizado: visão comum garante dados sempre atuais ao custo de reexecutar a consulta; visão materializada garante desempenho ao custo de dados potencialmente defasados.

Os três exercícios finais consolidam o conteúdo:

- **Exercício 78** — criar uma visão que exiba todos os dados de funcionário, além de sua idade, do nome do seu departamento, do nome do seu cargo e do seu salário.
- **Exercício 79** — criar uma visão que mostre a data de alocação em ordem crescente, o nome do funcionário, o nome do projeto e a quantidade de horas alocadas.
- **Exercício 80 (desafio opcional)** — criar uma visão materializada que mostre o código do projeto, seu nome, descrição, tipo, orçamento, o total de funcionários que já se alocaram nele e o total de horas de alocação.

O exercício 80 é o mais completo da trilha, porque exige junção de três tabelas, duas agregações distintas e a escolha justificada de materialização — exatamente o tipo de estrutura que serviria de camada de features para um modelo analítico.

4.13 Bibliografia da trilha TDP

Os slides indicam duas obras de referência:

- DAMAS, Luís. *SQL Structured Query Language*. 6ª edição. Rio de Janeiro: LTC, 2007.
- ELMASRI, Ramez; NAVATHE, Shamkant. *Sistemas de Banco de Dados*. 7ª edição. São Paulo: Pearson, 2018.

E, entre os links de apoio, o ranking DB-Engines para acompanhar a adoção relativa dos SGBDs e a documentação de tipos de dados do PostgreSQL.

5 Síntese da Disciplina

As oficinas cobrem duas pontas complementares do trabalho com dados em projetos de Inteligência Artificial, e a distância entre elas é justamente o que dá sentido ao conjunto.

Na ponta estruturada, a trilha **TDP** ensina que percepção não nasce do modelo, nasce do dado bem organizado. O percurso é rigorosamente sequencial e cada etapa depende da anterior: sem vocabulário claro (dado, informação, conhecimento, sabedoria; esquema, instância, metadado) não se modela; sem modelo conceitual correto — entidades, atributos classificados nos quatro eixos, relacionamentos com multiplicidade e hierarquias de especialização — a DDL produz um esquema que não representa o negócio; sem restrições declaradas na DDL (`NOT NULL`, `CHECK`, `UNIQUE`, `DEFAULT`, chaves estrangeiras) a DML deixa entrar lixo; e sem dados íntegros as consultas DQL produzem números confiantes e errados. A pedagogia da oficina reforça isso ao pedir explicitamente que o aluno tente violar as integridades: aprender vendo o banco recusar.

O topo dessa trilha — funções de agregação, `GROUP BY` com `HAVING`, subconsultas e visões — é onde o banco relacional deixa de ser repositório e passa a ser camada analítica. É aí que ele encontra a IA: uma visão materializada agregando total de horas e total de funcionários por projeto é, funcionalmente, uma tabela de *features*. O exercício da Bolsa de Valores explicita a ponte ao pedir um histórico de cotações destinado a análises de séries temporais.

Na ponta não estruturada, a oficina **PAI** trabalha com imagens e documentos, combinando processamento morfológico clássico (dilatação e erosão), OCR em duas gerações tecnológicas (Tesseract e docTR) e segmentação de instâncias com Mask R-CNN. A leitura de apoio devolve a mesma lição da trilha TDP em outro registro: mesmo com ResNet-50 pré-treinada e otimização cuidadosa, o gargalo real é a qualidade da anotação. Foram 8.550 imagens rotuladas de mais de 1 milhão coletadas, com 17% de ruído concentrado entre duas classes visualmente próximas. O framework RDS elevou o F1-Score em 3,5%, mas o ganho maior é conceitual: o ruído real não se parece com o sintético dos benchmarks, e métodos precisam permanecer simples o bastante para se compor com outras correções, como entropia cruzada ponderada para dados desbalanceados.

A conexão entre as duas pontas é direta. Em ambos os casos, o determinante do resultado é a disciplina aplicada aos dados antes da modelagem: no relacional, por restrições declaradas no esquema; no aprendizado profundo, por protocolo de anotação e estimativa honesta de ruído. As ferramentas mudam, o princípio não. Como sintetiza um dos slides de abertura, quem entende a estrutura dos dados entende o negócio — e a modelagem é o mapa antes da jornada da IA.

Não há material de slides disponível para a OFICINA - DPIA no contexto fornecido.