E-fólio B | Folha de resolução para E-fólio

UNIDADE CURRICULAR: Fundamentos de Bases de Dados

CÓDIGO: 21053

DOCENTE: Paulo Pombinho

A preencher pelo estudante

NOME: Fernando Miguel Ferreira de Beça

N.º DE ESTUDANTE: 2401179

CURSO: Licenciatura em Engenharia Informática

TRABALHO / RESOLUÇÃO:

1.

a) Restrições de integridade.

Restrição de integridade de entidade:

Por exemplo, a coluna id_cliente na tabela Cliente deve ser uma chave primária e não pode conter valores nulos (NOT NULL).

A restrição garante que cada cliente seja único e identificável forma clara e evidente no sistema. Sem ela, poderíamos ter dois clientes com o mesmo nome sem saber se são a mesma pessoa, ou um registo de clientes sem identificação, impossibilitando a associação de reservas.

Restrição de integridade referencial:

A coluna id_cliente na tabela Reserva deve ser uma chave estrangeira ou, Foreign Key, que referencia a id_cliente na tabela Cliente.

Esta restrição garante a consistência entre tabelas. Ela impede que seja criada uma reserva para um cliente que não existe na base de dados. Evita, também, que um cliente seja removido do sistema se ainda possuir reservas associadas a ele.

b)

```
CREATE OR REPLACE FUNCTION fn_valida_valor_pagamento()
RETURNS TRIGGER AS $$
DECLARE
   v_total_reserva DECIMAL(10,2);
BEGIN
   -- 1. Procurar o valor_total da reserva associada ao novo pagamento
   SELECT valor_total INTO v_total_reserva
   FROM Reserva
   WHERE id_reserva = NEW.id_reserva;

   -- 2. Comparar o valor do novo pagamento com o total da reserva
   IF NEW.valor > v_total_reserva THEN
```

```
        RAISE EXCEPTION 'Erro: O valor do pagamento (%) excede o valor total da
reserva (%)',
        NEW.valor, v_total_reserva;
    END IF;

    -- 3. Se estiver correto, permitir a inserção
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_valida_pagamento
BEFORE INSERT ON Pagamento
FOR EACH ROW
EXECUTE FUNCTION fn_valida_valor_pagamento();
```

Explicação detalhada do funcionamento do gatilho:

Arquitetura em duas partes.

Parte 1: Função (fn_valida_valor_pagamento)

Para além, de conter a lógica da validação pode ser reutilizada por outros triggers caso necessário.

Recebe acesso automático aos dados da linha através de NEW.

Retorna TRIGGER como tipo de dado especial.

Parte 2: Trigger (trg_valida_pagamento)

Define quando e em que tabela a função será executada. A função é executada automaticamente em cada inserção.

Momento e Granularidade de ativação:

BEFORE INSERT ON Pagamento

O trigger é acionado antes de cada inserção na tabela de Pagamento, isto é importante pois permite impedir a inserção caso a validação falhe.

Se fosse AFTER INSERT, os dados já estariam na base de dados.

FOR EACH ROW
O trigger executa para cada linha individual que está a ser inserida.
Por exemplo, se inserirmos 5 pagamentos numa única instrução, o trigger consegue validar os 5 separadamente.

EXECUTE FUNCTION
Invoca a função fn_valida_valor_pagamento() para processar a validação.

Fluxo de execução passo-a-passo.

1. Declaração de variável

DECLARE v_total_reserva DECIMAL (10,2);

Cria uma variável local para armazenar temporariamente o valor total da reserva.
DECIMAL (10,2) garante precisão para valores monetários até 99.999.999,99.

2. Consulta à tabela reserva.

SELECT valor_total INTO v_total_reserva
FROM Reserva
WHERE id_reserva = NEW.id_reserva;

NEW.id_reserva: refere-se ao id_reserva do pagamento que está a ser inserido.
NEW é uma variável especial que contém todos os valores da nova linha.

A query procura a reserva correspondente e guarda o seu valor_total em v_total_reserva.

3. Validação de regra de negócio

IF NEW.valor > v_total_reserva THEN

NEW.valor: É o valor do pagamento que está a ser inserido
Compara se o pagamento excedo o valor total da reserva

4. Rejeição com Mensagem Informativa

RAISE EXCEPTION 'Erro: O valor do pagamento (%) excede o valor total da reserva (%)',
NEW.valor, v_total_reserva;

RAISE EXCEPTION: lança um erro que cancela a operação de inserção.
Mensagem parametrizada com %: Substitui pelos valores reais.
Exemplo de saída: “Erro: o valor do pagamento (150.00) excede o valor total da reserva 100.00” .
Ao invés de uma mensagem genérica, através desta conseguimos saber exatamente onde está o erro.
A transação é automaticamente revertida, obtendo assim o rollback.

5. Autorização da Inserção

RETURN NEW;

Caso a validação passe (ou seja, nenhum erro foi detetado), RETURN NEW permite que a inserção prossiga. Sem o RETURN NEW, a inserção não acontece mesmo sem erro.

Exemplo prático de execuções:
-- Tabela Reserva contém:
-- id_reserva=1, valor_total=100.00

No caso de uma inserção ser válida:

INSERT INTO Pagamento VALUES (1, '2026-01-12', 80.00, 1);
Ordem de acontecimentos:
O trigger ativa antes da inserção.
Consulta v_total_reserva ← 100.00.
Verifica: 80.00 > 100.00? NÃO.
Executa RETURN NEW.
Inserção bem-sucedida.

No caso de uma inserção ser inválida:
INSERT INTO Pagamento VALUES (2, '2026-01-12', 150.00, 1);
Ordem de acontecimentos:
O trigger ativa antes da inserção.
Consulta v_total_reserva ← 100.00.
Verifica: 150.00 > 100.00? SIM.
Executa RAISE EXCEPTION
Erro. “Erro: o valor do pagamento (150.00) excede o valor total da reserva (100.00)”
Inserção cancelada, a base de dados permanece inalterada.

A partir destas implementações podemos considerar as seguintes vantagens:
Modularidade, pois a função pode ser reutilizada noutros triggers.
Integridade garantida ao nível da base de dados pois a regra é aplicada independentemente da aplicação cliente.
Mensagens informativas.

c)
A inviabilidade de substituir o gatilho proposto por uma chave estrangeira ou por uma restrição de verificação (*CHECK constraint*) decorre das limitações intrínsecas às restrições declarativas de integridade no modelo relacional e na sua implementação padrão em SQL.
Em primeiro lugar, a **chave estrangeira** possui uma finalidade estritamente estrutural e referencial. A sua função limita-se a assegurar a existência de uma correspondência entre tabelas, garantindo que um registo na tabela de pagamentos aponte obrigatoriamente para um identificador válido na tabela de reservas. Esta restrição, contudo, é incapaz de avaliar o conteúdo semântico ou os valores numéricos dos

atributos contidos nas tuplas relacionadas; ela valida a "ligação", mas ignora a conformidade do "montante" face ao contexto do negócio.
Por outro lado, a restrição **CHECK** é, por definição na maioria dos Sistemas de Gestão de Bases de Dados (SGBD), de âmbito local e intra-tabela. Embora seja eficaz para validar regras internas de uma linha — como garantir que um valor seja superior a zero —, ela não possui permissividade para realizar consultas externas (*subqueries*) ou aceder a atributos de outras tabelas no momento da validação. Dado que o valor total da reserva está armazenado numa entidade distinta daquela onde o pagamento é registado, o predicado de verificação não consegue estabelecer a ponte necessária para a comparação de dados entre diferentes objetos.
Desta forma, o **gatilho** emerge como a única solução técnica viável para este cenário, uma vez que atua como um mecanismo procedimental. Este permite a execução de uma lógica de interrogação dinâmica, na qual o SGBD interrompe o fluxo de escrita para consultar o estado da tabela de reservas e confrontar o valor proposto com o limite contratualizado, assegurando assim uma integridade de dados que transcende a mera estrutura relacional.

2.

a)

A expressão em álgebra relacional é a seguinte:

$$\pi \text{ nome } (\sigma \text{ ano=2024 } (\text{Inscricao} \bowtie \text{Aluno}))$$

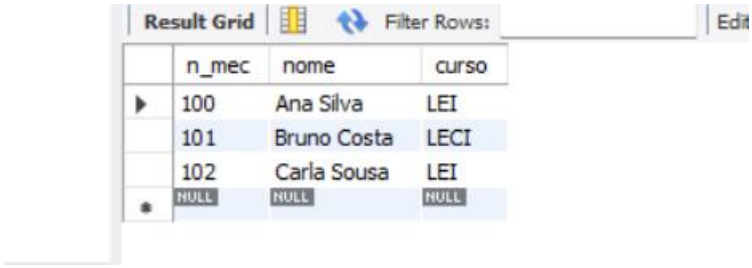
A construção desta expressão baseia-se numa sequência lógica de três operadores fundamentais:
Junção Natural (⋈): o primeiro passo consiste em interligar as relações Inscricao e Aluno. Este operador é essencial pois os dados necessários encontram-se dispersos: o atributo ano pertence à relação Inscricao, enquanto o atributo nome pertence à relação Aluno. A Junção utiliza o atributo comum n_mec para fundir as tupas correspondentes.
Selecção (σ ano=2024): Depois da junção, é aplicado o operador de selecção para restringir o conjunto de dados. Este operador atua como um filtro horizontal, mantendo apenas as linhas onde a condição booleana é verdadeira, eliminando todas as inscrições de outros anos lectivos.

Projecção (π nome): Utiliza-se a projeção para realizar um filtro vertical. Uma vez que o objectivo final é obter exclusivamente os nomes dos alunos, este operador descarta todas as outras colunas (como por exemplo, n_mec, curso ou cod_uc) resultantes das operações anteriores, devolvendo o conjunto final de nomes.

b)

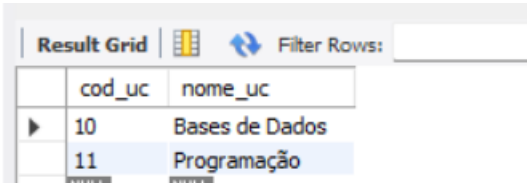
Exemplo prático com dados:

Tabela Aluno:



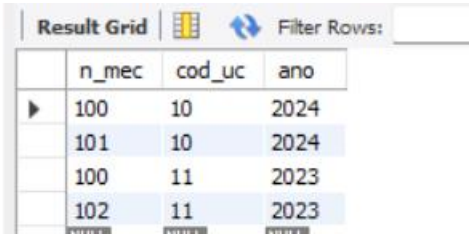
n_mec	nome	curso
100	Ana Silva	LEI
101	Bruno Costa	LECI
102	Carla Sousa	LEI
NULL	NULL	NULL

Tabela UC:



cod_uc	nome_uc
10	Bases de Dados
11	Programação

Tabela Inscricao:



n_mec	cod_uc	ano
100	10	2024
101	10	2024
100	11	2023
102	11	2023

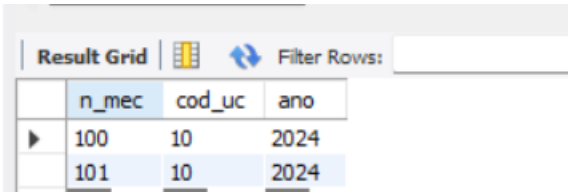
Obtemos um total de 9 tuplos. (3 Aluno + 2UC + 4 Inscricao).

Execução passo a passo.

1 . Seleção: σ ano=2024 (Inscricao)

Nesta operação pretendemos filtrar as inscrições do ano 2024.

Resultado intermédio 1:



n_mec	cod_uc	ano
100	10	2024
101	10	2024

Explicação:

Eliminou: (100, 11, 2023) em que o ano é 2023 e (102, 11, 2023) em que o ano é 2023

Mantendo: (100, 10, 2024) e (101, 10, 2024) em que o ano de Incricao é 2024.

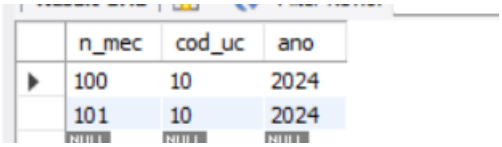
Tuplos: 4 → 2 tuplos.

2. Junção Natural: (Resultado1) ⋈ Aluno

Nesta operação pretende-se juntar as inscrições de 2024 com os dados dos alunos através do atributo comum n_mec.

Input:

Resultado Intermédio 1:



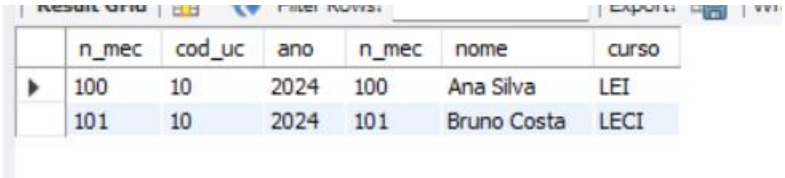
n_mec	cod_uc	ano
100	10	2024
101	10	2024

Tabela Aluno:

Result Grid | Filter Rows:

n_mec	nome	curso
100	Ana Silva	LEI
101	Bruno Costa	LECI
102	Carla Sousa	LEI
NULL	NULL	NULL

Resultado Intermédio 2:



n_mec	cod_uc	ano	n_mec	nome	curso
100	10	2024	100	Ana Silva	LEI
101	10	2024	101	Bruno Costa	LECI

Explicação:

n_mec=100 e n_mec=101 existem em ambas as tabelas, logo junta informação.

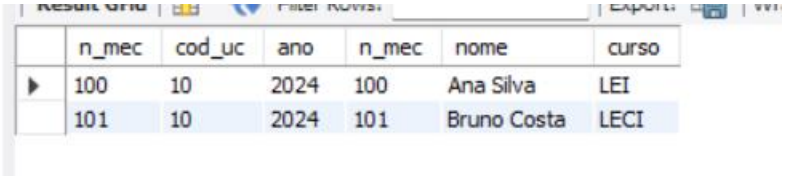
n_mec=102 existe em Aluno mas não no Resultado1 (não tem inscrições em 2024), logo, não aparece.

Tuplos: 2 → 2 tuplos.

3. Projeção: π nome (Resultado2)

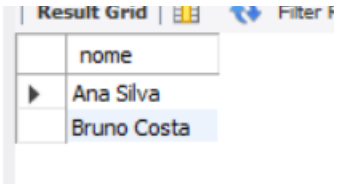
Nesta operação pretendemos selecionar apenas a coluna nome, removendo as restantes colunas.

Input – Resultado Intermédio 2:



n_mec	cod_uc	ano	n_mec	nome	curso
100	10	2024	100	Ana Silva	LEI
101	10	2024	101	Bruno Costa	LECI

Resultado:



nome
Ana Silva
Bruno Costa

Explicação:

Mantém apenas a coluna nome.

Elimina n_mec, cod_uc, ano, curso.

Neste caso, não há duplicados para eliminar (cada aluno aparece uma vez)

Tuplos: 2 → 2 tuplos.

Observações:

Carla Sousa (102) não aparece porque tem inscrição em 2023 e não em 2024. Foi eliminada no primeiro passo pela seleção σ ano=2024.

Ana Silva (100) aparece uma vez apesar de ter 2 inscrições no total. Apenas a inscrição de 2024 é considerada. Se tivesse duas inscrições em 2024, apareceria apenas uma vez no resultado devido à projecção eliminar duplicados.

Eficiência da ordem: Filtrar primeiro o (σ) reduziu os dados de 4 para 2 tuplos, o join trabalhou apenas com 2 tuplos em vez 4. Tornou-se mais eficiente que fazer join completo e depois filtrar.

3.

a) **Entidades identificadas.**

1. Pessoa.

O texto refere que a "A empresa trabalha com pessoas, que podem ser formadores ou formandos". Assumo assim que este seja uma entidade principal do sistema.

Atributos:

- cc_numero (chave primária): "são identificadas pelo número do seu cartão de cidadão".
- nome: "Todas as pessoas têm nome".
- email: "Todas as pessoas têm […] email".
- contacto_telefonico: "Todas as pessoas têm […] contatcto telefónico".

2. Formador

O texto distingue formadores de formandos, indicando que "podem ser formadores ou formandos" e que "Os formadores têm ainda uma área de especialização principal." Isto sugere uma especialização da entidade Pessoa.

Atributos:

- cc_numero (chave primária, também chave estrangeira para Pessoa)
- area_especializacao: "Os formadores têm ainda uma área de especialização principal"

3. Formando

Similar ao formador, o texto identifica formandos como um tipo específico de pessoa com atributos próprios: "Os formandos têm informação relativa à sua inscrição ser particular ou a indicação de entidade externa".

Atributos:

- cc_numero (chave primária, também chave estrangeira para Pessoa)
- tipo_inscricao: pode ser "particular" ou indicar uma "entidade externa".
- entidade_externa: "indicação da entidade externa (empresa onde trabalham)" – este atributo só tem valor quando não é particular.

4. Curso

O texto afirma que "A empresa organiza cursos", sendo cada curso identificado por características próprias.

Atributos:
- codigo (chave primária): “cada um com um código”.
- titulo: “cada um com um […] título”.
- duração_total_horas: “e uma duração total em horas”.

5 . Modulo
O texto indica que “Cada curso é composto por vários módulos” e que “Um módulo não existe de forma independente fora do curso a que pertence”, o que caracteriza uma entidade dependente de Curso.
Atribuitos:
- codigo_curso (parte da chave primária, chave entrangeira para Curso).
- numero_sequencial (parte da chave primária): “numerados sequencialmente dentro de cada curso”.
- titulo: “com um título”.
- carga_horaria: “e uma carga horária própria”.

Relacionamentos identificados.
1 . Leciona (entre Formador e Curso)
O texto afirma que “Os formadores podem lecionar vários cursos e um curso pode ser lecionado por vários formadores”.
Cardinalidades:
Formador para Curso: (0,N): um formador pode lecionar zero ou vários cursos
Curso para Formador: (1,N): um curso deve ser lecionado pelo menos por um formador.
Atributos do relacionamento:
- funcao: “é necessário registar a função desempenhada (por exemplo, coordenador ou formador convidado).
- horas_lecionadas: “o número total de horas lecionadas nesse curso”.

Nota: Este é um relacionamento N:M (muitos para poucos) com atributos próprios.

2 . Inscreve-se (entre Formando e Curso)

O texto menciona que “Os formandos inscrevem-se nos cursos”.

Cardinalidades:

Formando para Curso (0,N): um formando pode inscrever-se em zero ou vários cursos.

Curso para Formando (0,N): um curso pode ter zero ou vários formandos inscritos.

Atributos do relacionamento:

- data_inscricao: “pretende-se registar a data de inscrição”.
- estado: “e o estado da inscrição (ativa, concluída, anulada)”.

3 . Composto_Por (Entre Curso e Modulo)

O texto estabelece que “Cada curso é composto por vários módulos” e que “Um módulo não existe de forma independente fora do curso a que pertence”

Cardinalidades:

Curso para Modulo (1,N): cada curso é composto por pelo menos um módulo (assumindo que cursos têm sempre módulos).

Modulo para Curso (1,1): cada módulo pertence obrigatoriamente a exatamente um curso (dependência existencial)

Resumo.

Hierquia de Generalização-espacialização: Pessoa é a superclasse, com Formador e Formando como subclasses, pois partilham atributos comuns, mas têm características especificas.

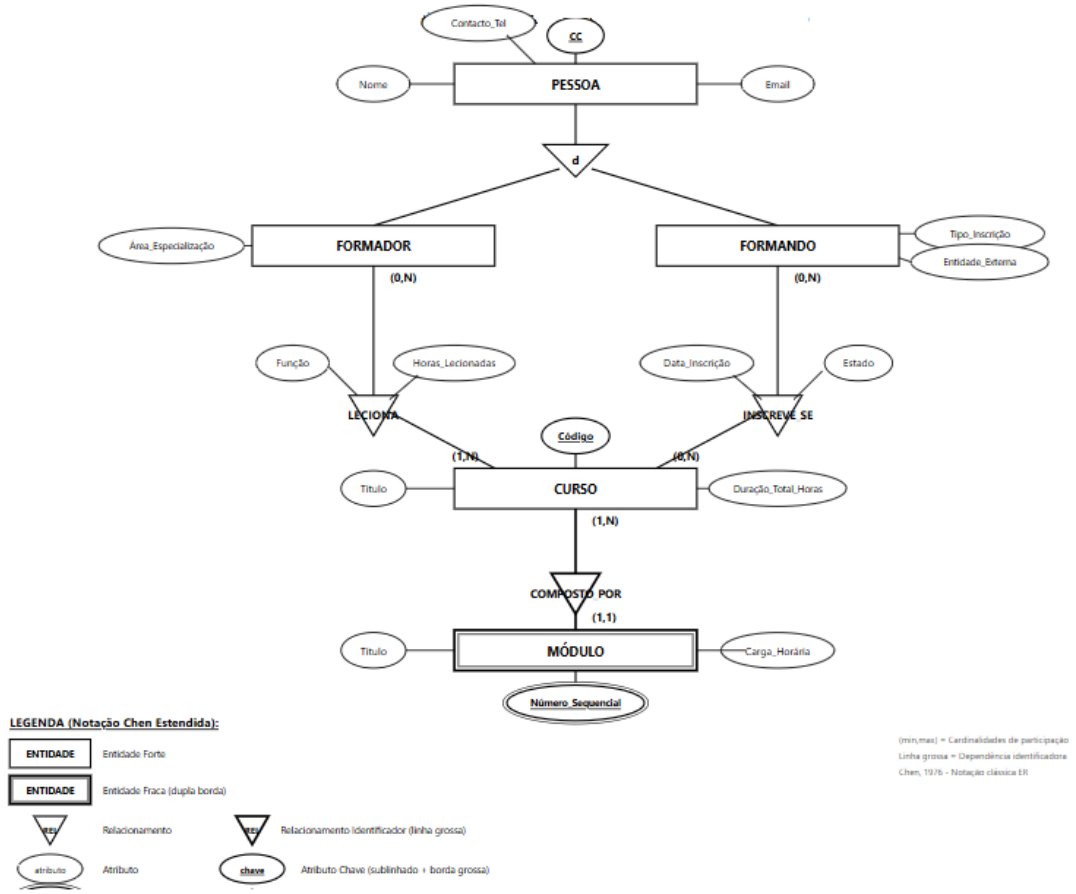
Entidade fraca: Modulo é modelado como entidade fraca porque não tem existência independente sem Curso e a sua identificação dependo do curso a que pertence.

Relacionamento com atributos: Tanto Leciona como Inscreve-se são relacionamentos N:M que requerem atributos próprios, o que na implementação física resultará em tabelas associativas.

Cardinalidades Mínimas: Algumas cardinalidades mínimas foram assumidas com base em regras de negócio lógicas, por exemplo, um curso de ter pelo menos um formador e pelo menos um módulo.

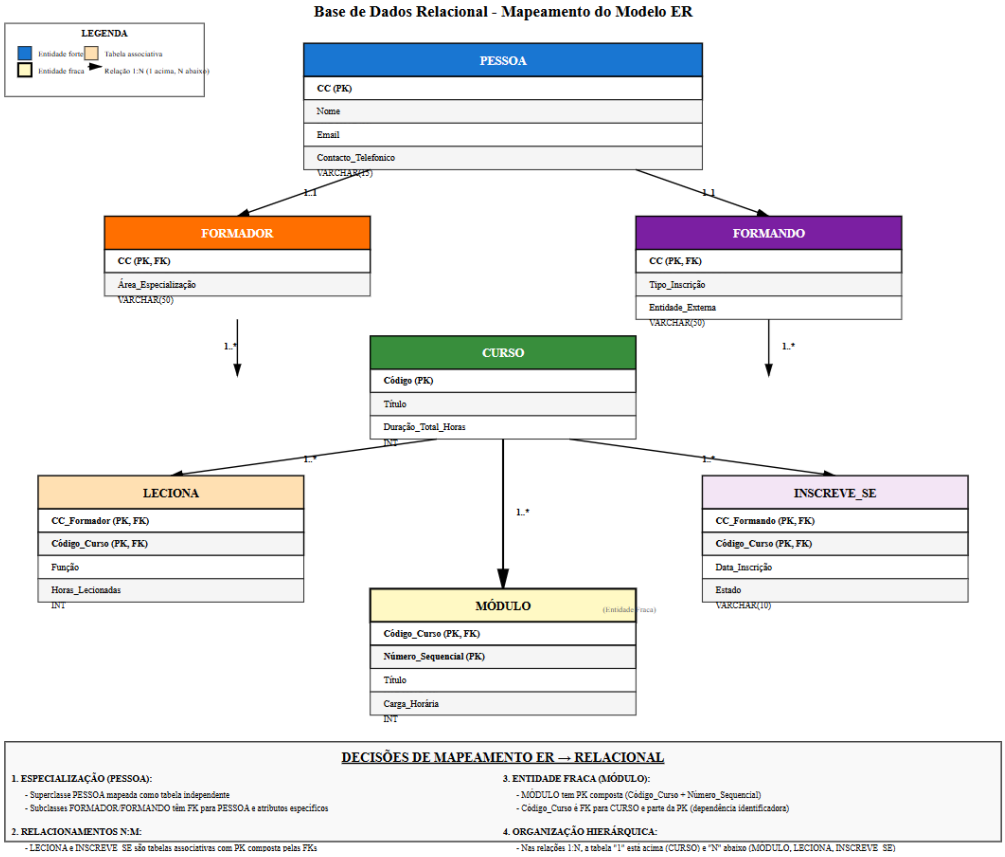
b)

Diagrama do modelo Entidade-Relação



c)

Base de dados relacional correspondente ao modelo anterior.



O processo de transição entre o modelo conceitual Entidade-Relação (ER) e o modelo lógico relacional requer a aplicação sistemática de regras de transformação que preservem a semântica original enquanto asseguram a viabilidade de implementação em sistemas de gestão de bases de dados relacionais. As decisões tomadas neste mapeamento fundamentam-se em princípios estabelecidos na teoria de bases de dados e visam otimizar tanto a integridade dos dados como a eficiência das operações.

Transformação de Entidades e Atributos

Cada entidade identificada no diagrama ER converteu-se numa relação independente no modelo relacional. Esta transformação direta mantém a correspondência biunívoca entre conceitos do domínio e estruturas de armazenamento. Os atributos das entidades foram mapeados como colunas das tabelas correspondentes, com a especificação de tipos de dados apropriados ao contexto. A chave primária de cada

entidade forte preservou-se como identificador único na tabela resultante, garantindo a identificação inequívoca de cada tupla.

Tratamento de Generalizações e Especializações

A hierarquia de generalização identificada no modelo conceitual, onde a entidade PESSOA se especializa em FORMADOR e FORMANDO, exigiu uma abordagem cuidadosa. Optei pela estratégia de múltiplas relações com herança de chave, que consiste em manter a superclasse como tabela independente e criar tabelas específicas para cada subclasse. Esta decisão fundamenta-se em dois princípios: primeiramente, evita a redundância de informação ao não repetir atributos comuns em múltiplas tabelas; em segundo lugar, previne a ocorrência de valores nulos generalizados que surgiriam numa solução de tabela única. A ligação entre tabelas é garantida através de chaves estrangeiras que referenciam a chave primária da superclasse, mantendo a integridade referencial e permitindo a reconstrução da hierarquia conceptual através de operações de junção.

Conversão de Relacionamentos entre Entidades

Os relacionamentos presentes no modelo ER foram transformados segundo a sua cardinalidade e características específicas:

Para associações do tipo muitos-para-muitos (N:M), como LECIONA e INSCREVE_SE, criei relações associativas independentes. Estas novas tabelas possuem chaves primárias compostas pelas chaves estrangeiras das entidades participantes, resolvendo assim a incapacidade dos sistemas relacionais de representar diretamente este tipo de associação. Os atributos próprios destes relacionamentos, como função ou data de inscrição, migraram para as tabelas associativas, mantendo assim toda a informação contextual da associação.

No caso de relacionamentos um-para-muitos (1:N), como a composição entre CURSO e MÓDULO, aplicou-se o princípio da migração da chave estrangeira. A chave primária da entidade no lado "um" da relação incorpora-se como chave estrangeira na entidade do lado "muitos". Esta solução é computacionalmente eficiente, evitando a criação de estruturas adicionais e facilitando operações de consulta frequentes que seguem a direção da dependência.

Representação de Entidades Fracas

A entidade MÓDULO, caracterizada no modelo ER como entidade fraca devido à sua dependência existencial em relação a CURSO, transformou-se numa relação com chave primária composta. Esta solução reflete semanticamente a natureza

dependente da entidade: parte da chave primária é precisamente a chave estrangeira que referencia a entidade forte, enquanto um discriminador local completa a identificação única. Esta abordagem assegura que nenhum módulo possa existir sem referência a um curso válido, impondo através da estrutura da base de dados a regra de negócio identificada conceptualmente.

Organização Hierárquica da Representação Visual

A disposição gráfica das tabelas no diagrama relacional segue uma convenção pedagógica que facilita a compreensão das dependências. As relações que participam em associações como lado "um" posicionam-se visualmente acima das relações do lado "muitos", criando uma hierarquia que reflete o sentido das dependências. Esta organização não afeta a implementação física da base de dados, mas constitui uma ferramenta cognitiva que auxilia na compreensão da estrutura e nas direções das referências entre tabelas.

Preservação de Restrições de Integridade

O mapeamento incluiu a definição explícita de mecanismos que asseguram a qualidade dos dados. As chaves primárias garantem a unicidade dos registos, enquanto as chaves estrangeiras implementam a integridade referencial entre relações. Para atributos com domínios restritos, como o estado da inscrição, definiram-se restrições de verificação que limitam os valores possíveis. Estas medidas transformam regras de negócio identificadas conceptualmente em garantias implementadas ao nível do sistema de gestão da base de dados.

Normalização do Esquema Resultante

O esquema relacional produzido encontra-se na Terceira Forma Normal (3FN), um equilíbrio cuidadosamente ponderado entre eliminação de redundâncias e desempenho operacional. Esta normalização foi alcançada através da análise das dependências funcionais entre atributos e da reorganização que elimina dependências parciais e transitivas. O resultado é uma estrutura que minimiza a duplicação de informação sem fragmentar excessivamente os dados, facilitando assim operações de manutenção e atualização.

Considerações de Extensibilidade

As decisões de mapeamento incorporaram previsão para evolução futura do sistema. A separação entre superclasse e subclasses permite adicionar novas especializações sem modificar a estrutura existente. As tabelas associativas podem acomodar novos atributos descritivos das relações, e a normalização adequada facilita a incorporação

de novas entidades e relacionamentos. Esta flexibilidade é essencial em contextos organizacionais onde os requisitos de informação evoluem ao longo do tempo.

Em síntese, o processo de mapeamento seguiu um conjunto coerente de princípios teóricos adaptados às especificidades do domínio em análise. Cada decisão resultou da ponderação entre fidelidade ao modelo conceptual, eficiência operacional, manutenibilidade e extensibilidade. O esquema relacional resultante constitui assim uma base sólida para implementação física, capaz de suportar as operações do sistema de gestão de formação enquanto mantém a correção e consistência dos dados armazenados.