Constraints e Relacionamentos

Aula 11 — Ligando as tabelas de verdade

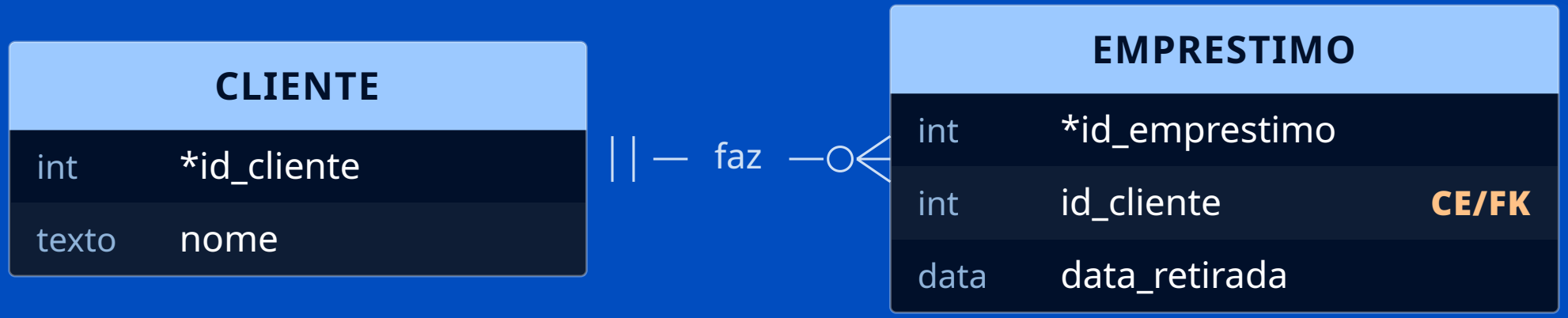
UniSenac Centro Universitário RS
Senac
Fecomércio
IFEP
Sesc
Sindicatos Empresariais

O mapa de hoje

- **Bloco 1** — **integridade referencial**: a chave estrangeira em SQL
- **Bloco 2** — as outras **constraints**: `NOT NULL`, `UNIQUE`, `CHECK`, `DEFAULT`
- **Bloco 3** — ordem de criação, `ON DELETE` e como consertar sem recomeçar
- **Bloco 4** — mão na massa: criar **todas** as tabelas do projeto

> No fim de hoje, o banco de vocês está **completo e vazio**. Os dados entram na Aula 12.

Lembrete — a legenda do DER



Hoje o `*` vira `PRIMARY KEY` e o `CE/FK` vira `REFERENCES`.
O desenho e o SQL passam a dizer **a mesma coisa**.

Retomando a Aula 10

- Qual comando cria uma tabela?
- O que `SERIAL` faz?
- O que `NOT NULL` garante?
- Onde você confere em qual banco o Query Tool está?

> Abram o arquivo `.sql` do grupo e o **DER** lado a lado. São os dois insumos de hoje.

Bloco 1

Integridade referencial

O problema — a linha que aponta para o nada

Nas tabelas de ontem, nada impede isto:

```
INSERT INTO emprestimo (id_cliente, data_retirada) VALUES (7, '2026-10-22');
```

E se **não existe** cliente com `id_cliente = 7`?

- O empréstimo fica órfão: aponta para um cliente que não existe
- A consulta "quem está com jogo em casa" nunca acha ele
- Ninguém é avisado

A solução — REFERENCES

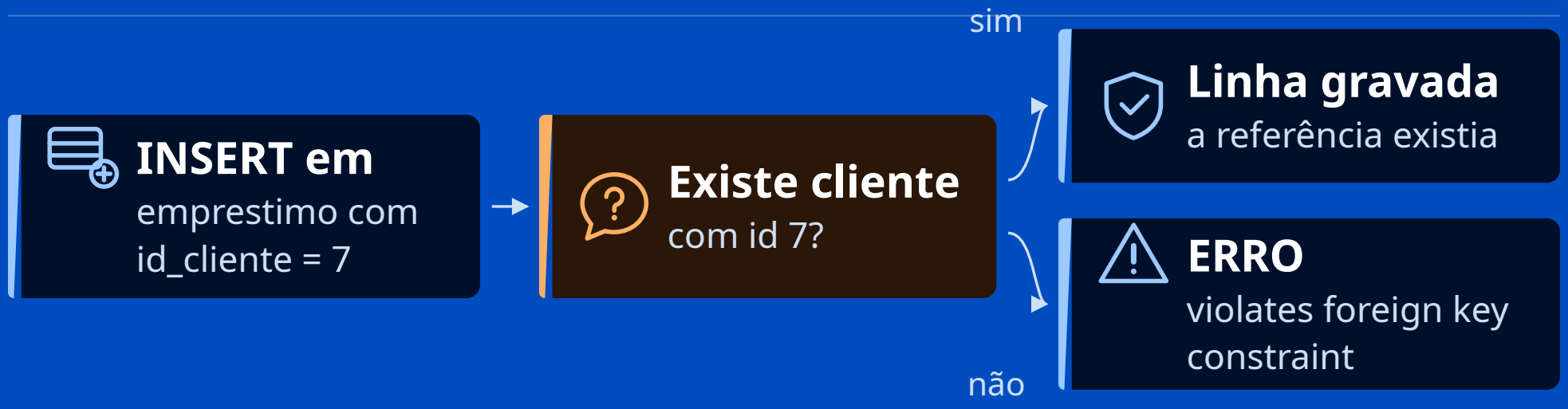
```sql
CREATE TABLE emprestimo (
    id_emprestimo SERIAL  PRIMARY KEY,
    data_retirada DATE    NOT NULL,
    data_prevista DATE    NOT NULL,
    id_cliente    INTEGER REFERENCES cliente(id_cliente)
);
```

GLOSSÁRIO

`REFERENCES`	referencia — aponta para
`foreign key constraint`	restrição de chave estrangeira
`referential integrity`	integridade referencial

O que o banco faz agora

O QUE O BANCO CONFERE ANTES DE GRAVAR



Essa é a **garantia 2** da Aula 01 — **integridade** — funcionando. O banco recusa o dado errado em vez de aceitar e estragar em silêncio.

⚠ A mensagem de erro — leia com calma

```
ERROR: insert or update on table "emprestimo" violates foreign key
constraint "emprestimo_id_cliente_fkey"
DETAIL: Key (id_cliente)=(7) is not present in table "cliente".
```

GLOSSÁRIO

`violates foreign key constraint`	quebra a regra de chave estrangeira
`is not present in table`	não existe na tabela

A linha `DETAIL` diz **exatamente** qual valor e qual tabela. Leia ela.

Checagem do Bloco 1

```sql
CREATE TABLE item_emprestimo (
    id_emprestimo INTEGER REFERENCES emprestimo(id_emprestimo),
    id_jogo       INTEGER REFERENCES jogo(id_jogo),
    preco_pago    NUMERIC(6,2) NOT NULL,
    PRIMARY KEY (id_emprestimo, id_jogo)
);
```

- Quantas chaves estrangeiras essa tabela tem? Por quê?
- O que acontece se eu inserir `id_jogo = 500` sem esse jogo existir?

forms.gle/qPrJsFo2PqEX2HMD8

Intervalo

10 minutos

Bloco 2

As outras constraints



NOT NULL
nunca vazio



UNIQUE
não pode repetir



CHECK
a regra de negócio no banco



DEFAULT
o valor que entra sozinho

Constraint, em uma frase

Constraint é uma regra que o banco faz valer. Se o dado não obedece, o banco **recusa** — e o dado errado nunca entra.

GLOSSÁRIO

`constraint`	restrição — a regra que o banco faz valer

Regra que está só na cabeça do atendente não é constraint.
Regra que está no banco, é.

`NOT NULL` — nunca vazio

```
nome VARCHAR(100) NOT NULL
```

- Vem direto da coluna **"Obrigatório?"** do dicionário de dados (Aula 04)
- Sem ele, alguém cadastra um cliente sem nome — e ninguém descobre até o relatório

> Lembrem da Aula 04: `NULL` é "não sei", não é zero nem texto vazio. `NOT NULL` diz que "não sei" é inaceitável naquela coluna.

`UNIQUE` — não pode repetir

```
email VARCHAR(120) UNIQUE
cpf   VARCHAR(11)  NOT NULL UNIQUE
```

- Garante que não existem dois clientes com o mesmo e-mail
- É a **chave candidata** da Aula 05 finalmente virando regra

`UNIQUE` e `PRIMARY KEY` não são a mesma coisa: `PRIMARY KEY` é **única E nunca vazia**, e só pode haver uma por tabela. `UNIQUE` pode haver várias, e aceita nulo.

CHECK — a regra de negócio no banco

```
ano_lancamento INTEGER CHECK (ano_lancamento > 1970),
preco_diaria   NUMERIC(10,2) CHECK (preco_diaria >= 0),
data_prevista  DATE NOT NULL CHECK (data_prevista >= data_retirada)
```

GLOSSÁRIO

CHECK	verificar — o valor precisa satisfazer a condição

DEFAULT — o valor que entra sozinho

```
data_cadastro  DATE    DEFAULT CURRENT_DATE,
ativo          BOOLEAN DEFAULT TRUE,
quantidade     INTEGER DEFAULT 1
```

GLOSSÁRIO

DEFAULT	padrão — o valor usado quando ninguém informa
CURRENT_DATE	data atual — o dia de hoje, segundo o banco

Onde cada decisão do papel vira constraint

No papel	No SQL
`*` chave primária (A05)	`PRIMARY KEY`
`CE/FK → tabela` (A07)	`REFERENCES tabela(coluna)`
"Obrigatório: sim" (A04)	`NOT NULL`
chave candidata (A05)	`UNIQUE`
regra do minimundo (A03)	`CHECK (...)`

Nove aulas de papel, cinco linhas de tradução. Foi para isso que a gente fez tudo aquilo.

Checagem do Bloco 2

Que constraint você usaria?

- O CPF do cliente não pode repetir
- O nome do jogo não pode ficar vazio
- A nota da avaliação vai de 0 a 5
- A data de cadastro é a de hoje, se ninguém informar

forms.gle/x2QWc8URunzAzAYv5

Intervalo

10 minutos

Bloco 3

Ordem de criação e conserto sem recomeçar

A ordem importa

NÍVEL 1 — NÃO DEPENDEM DE NINGUÉM

1 · plataforma
nenhuma chave estrangeira

2 · cliente
nenhuma chave estrangeira

3 · jogo
nenhuma chave estrangeira

NÍVEL 2 — APONTAM PARA UMA

4 · endereco
aponta para cliente

5 · jogos_plataforma
aponta para jogo e plataforma

6 · emprestimo
aponta para cliente

NÍVEL 3 — APONTA PARA DUAS

7 · item_emprestimo
aponta para emprestimo e jogo

⚠ A regra — e o erro de quem a quebra

Uma tabela com `REFERENCES` só pode ser criada **depois** da tabela referenciada. Criar `emprestimo` antes de `cliente` dá erro.

```
ERRO:  relação "cliente" não existe
LINHA 3:     id_cliente INTEGER REFERENCES cliente(id_cliente),
```

A mensagem diz **qual** tabela faltou. O nível 1 do slide anterior é a resposta.

`ON DELETE` — o que fazer quando o pai some

Apagar um cliente que tem empréstimos: o que deve acontecer?

Opção	Comportamento
`ON DELETE RESTRICT` (padrão)	**recusa** apagar enquanto houver empréstimos ligados
`ON DELETE CASCADE`	apaga o cliente **e todos os empréstimos** dele
`ON DELETE SET NULL`	apaga o cliente e deixa os empréstimos **sem dono**

No curso usamos o padrão (`RESTRICT`). `CASCADE` é perigoso: um `DELETE` distraído pode levar meio banco junto.

`ALTER TABLE` — consertar sem recomeçar

```sql
ALTER TABLE jogo ADD COLUMN ano_lancamento INTEGER;

ALTER TABLE jogo ALTER COLUMN titulo SET NOT NULL;

ALTER TABLE emprestimo
    ADD CONSTRAINT emprestimo_cliente_fk
    FOREIGN KEY (id_cliente) REFERENCES cliente(id_cliente);
```

Esqueceu uma coluna? Esqueceu um `NOT NULL`? **Não precisa apagar a tabela e recomeçar.** `ALTER TABLE` resolve.

⚠ Erro comum — apagar tudo e recomeçar

> A tentação de resolver qualquer erro com `DROP TABLE` e recriar é grande — e funciona hoje, porque o banco está vazio. **A partir da Aula 12, não funciona mais**: vocês vão perder os dados.

Peguem o hábito certo agora: erro de estrutura se conserta com `ALTER`.

Checagem do Bloco 3

- Em que ordem você criaria: `item_emprestimo`, `emprestimo`, `cliente`, `jogo`?
- O que acontece, por padrão, se você apagar um cliente que tem empréstimos?
- Esqueceu o `NOT NULL` no `titulo`. Como conserta sem apagar a tabela?



forms.gle/8RTd9sLxLMaJAaNF7

Intervalo

10 minutos

MÃO NA MASSA

Criar todas as tabelas do projeto

1. Abram o `.sql` da Aula 10 e o esquema normalizado da Aula 09
2. No papel, escrevam a **ordem de criação** das tabelas do grupo
3. Criem as tabelas que faltam, **nessa ordem**, com todas as constraints
4. Cada chave estrangeira com `REFERENCES tabela(coluna)`
5. A tabela do meio do N:N por **último**
6. Testem a integridade: tentem inserir uma FK que **não existe** e leiam o erro
7. Salvem o `.sql` completo e entreguem

MÃO NA MASSA

O combinado desta entrega

GLOSSÁRIO

`grupoNN-aulaNN-artefato.<ext>` `grupo03-aula11-criar-tabelas.sql`

O que eu confiro: As tabelas foram criadas na **ordem** certa · Toda FK tem `REFERENCES` apontando para a tabela e coluna certas · As colunas obrigatórias do dicionário têm `NOT NULL` · Existe pelo menos um `CHECK` vindo de uma regra do minimundo · O grupo **viu** o erro de integridade acontecer na tela · O número de tabelas bate com o esquema da Aula 09

Terminou? Escreva um `CHECK` a mais, vindo de uma regra de negócio do seu projeto. Depois ajude o grupo do lado.

PARA CASA · OPCIONAL

Antes da próxima aula

Testar em casa: no seu banco pessoal, tente apagar uma linha que tem outras apontando para ela. Leia a mensagem de erro — é o `ON DELETE RESTRICT` trabalhando.

Tarefa opcional: acrescente um `CHECK` a mais no banco do projeto, vindo de uma regra do minimundo que vocês escreveram lá na Aula 03.

Para treinar: `locadora-exercicios-11-constraints.sql`, na página da aula — nove tentativas de quebrar as regras, e uma pegadinha com dois nulos.

Para ver acontecer: demonstrações da Aula 11 — mandar a linha errada e ler a recusa do banco, e apagar a Ana sob os três ON DELETE — e oito perguntas com resposta na hora. No site da UC, na página da aula.

Para a próxima aula

O banco está completo e **vazio**. Como colocar dados dentro — e como fazer isso a partir de um programa em Python?

UniSenac Centro Universitário RS
Senac
Fecomércio
IFEP
Sesc
Sindicatos Empresariais