Chaves

Modele identidade e pertencimento

Pré-requisitos

Conheça CREATE TABLE, INSERT e JOIN. Use Python 3 com sqlite3 para executar o exemplo. A estrutura representa unidades de atendimento, salas e reservas. Uma sala pertence a uma unidade e seu código é local, como 'A' ou 'B'.

Chaves candidatas e chave primária [1]

Uma chave candidata é um conjunto mínimo de colunas que identifica uma linha unicamente. Escolha uma como chave primária. Na tabela salas, unidade_id e codigo formam uma chave composta. O código A pode existir tanto no Centro quanto no Sul, mas só uma vez em cada unidade. Declarar codigo como chave única global impediria essa repetição legítima.

Chave natural ou identificador artificial

A combinação de unidade e código tem significado no negócio. Se o código muda com frequência, um id imutável pode simplificar as referências. Preserve então a regra de negócio com UNIQUE (unidade_id, codigo). Se a reserva armazena apenas sala_id, obtenha a unidade por meio da sala. Caso armazene os dois campos, expresse a consistência da combinação com uma chave estrangeira composta ou outra modelagem equivalente.

Chaves estrangeiras e nulidade [2]

Uma chave estrangeira exige que o valor referenciado exista na tabela de destino, conforme as regras de nulidade do banco. Neste exercício, NOT NULL nos dois campos da reserva exige uma associação completa. O destino da referência tem unicidade no par inteiro. Duas referências independentes à unidade e à sala poderiam aceitar valores existentes que nunca estiveram associados.

Decida o que acontece ao remover uma sala

A política do exercício protege o histórico com ON DELETE RESTRICT. A remoção de sala com reservas é recusada. Se a operação exige retirar a sala de circulação, registre sua desativação e preserve o vínculo histórico. Reserve CASCADE para relações em que a exclusão dos itens dependentes seja realmente parte do comportamento esperado.

Chaves

Exercício resolvido • Identifique a sala certa

Enunciado

Centro e Sul têm uma sala A. Só Sul tem sala B. As reservas 42 e 43 referenciam as salas A de Centro e Sul, respectivamente. Modele a relação e garanta que uma reserva para a sala B do Centro seja recusada.

1. Salve chaves.sql em UTF-8

```
CREATE TABLE unidades (
  id INTEGER PRIMARY KEY,
  nome TEXT NOT NULL
);
CREATE TABLE salas (
  unidade_id INTEGER NOT NULL REFERENCES unidades(id),
  codigo TEXT NOT NULL,
  nome TEXT NOT NULL,
  PRIMARY KEY (unidade_id, codigo)
);
CREATE TABLE reservas (
  id INTEGER PRIMARY KEY,
  unidade_id INTEGER NOT NULL,
  sala_codigo TEXT NOT NULL,
  FOREIGN KEY (unidade_id, sala_codigo)
    REFERENCES salas (unidade_id, codigo)
    ON DELETE RESTRICT
);
```

2. Salve dados.sql em UTF-8

```
INSERT INTO unidades VALUES (1, 'Centro'), (2, 'Sul');
INSERT INTO salas VALUES
  (1, 'A', 'Estudio Centro'),
  (2, 'A', 'Estudio Sul'),
  (2, 'B', 'Sala de ensaio');
INSERT INTO reservas VALUES (42, 1, 'A'), (43, 2, 'A');
```

Explique o desenho

A chave composta permite a repetição de A entre unidades. A reserva usa id independente como identidade, pois muitas reservas podem fazer referência à mesma sala ao longo do tempo. A chave estrangeira protege a referência à sala. Uma regra de capacidade ou sobreposição é uma decisão adicional.

Chaves

Execute a junção e teste a integridade

3. Salve consulta.sql na mesma pasta

```
SELECT r.id, u.nome, s.nome
FROM reservas r
JOIN salas s ON s.unidade_id = r.unidade_id
            AND s.codigo = r.sala_codigo
JOIN unidades u ON u.id = s.unidade_id
ORDER BY r.id;
```

4. Salve executar.py e rode python executar.py

```
from pathlib import Path
import sqlite3

banco = sqlite3.connect(':memory:')
banco.execute('PRAGMA foreign_keys = ON')
banco.executescript(Path('chaves.sql').read_text(encoding='utf-8'))
banco.executescript(Path('dados.sql').read_text(encoding='utf-8'))
for linha in banco.execute(Path('consulta.sql').read_text()):
    print(*linha, sep=' | ')

try:
    banco.execute("INSERT INTO reservas VALUES (44, 1, 'B')")
except sqlite3.IntegrityError as erro:
    print('Recusa esperada:', erro)
else:
    raise AssertionError('Reserva invalida aceita')
```

Verifique o resultado

A primeira linha é ‘42 | Centro | Estudio Centro’. A segunda é ‘43 | Sul | Estudio Sul’. A inserção 44 é recusada porque o par (1, B) está ausente, embora a unidade 1 e o código B existam separadamente. O PRAGMA é necessário para ativar a verificação nessa conexão SQLite.

Variação e critérios de conferência

Retire a igualdade s.unidade_id = r.unidade_id da junção e mantenha apenas a igualdade de códigos. Cada reserva A passa a corresponder a duas salas e a consulta produz quatro linhas. Reponha o par completo. Tente também duplicar a sala (1, A) e remover a sala (1, A) com a reserva 42 existente. O banco deve recusar as duas operações.

Referências primárias

[1] PostgreSQL • Chaves primárias, únicas e estrangeiras

[2] SQLite • Chaves estrangeiras compostas e ações de exclusão

Chaves

Escolha a chave que representa o recurso

Distinga identidade de identificação para exibição

Um UUID pode identificar uma sala internamente, enquanto o código exibido é único apenas na unidade. Use o UUID como chave primária, uma restrição única para unidade e código, e referencie o UUID na tabela de reservas.

Exercício adicional

Duas unidades possuem uma sala chamada A. Como guardar as reservas sem ambiguidade?

Resolução

Crie um identificador por sala e referencie esse identificador em cada reserva. A restrição composta permite reutilizar A em unidades diferentes e impede que a mesma unidade cadastre A duas vezes.

Ao assistir ao vídeo

Observe como uma chave estrangeira liga dois registros e identifique o lado que recebe a referência.

Vídeo complementar

Diferença entre chave primária e chave estrangeira