USO DE CHAVES ESTRANGEIRAS
E
RELACIONAMENTO ENTRE TABELAS

Nome: Anderson Paim dos Santos Vianna

Turma: IntroBD-ADS-2024_2

Data: 05 de Dezembro de 2024

Chaves Estrangeiras em Banco de Dados

Introdução:

O conceito aqui a ser abordado, são relações entre duas tabelas para consultas e integridade, onde basicamente no campo da determinado a ter chave estrangeira constara a chave primaria da tabela ou tabelas envolvidas. Podemos ter mais de um campo com chave estrangeira que relaciona mais de uma tabela ou a mesma tabela, que este campo pode ser nominado de diversas formas.

Esta relação não deve permitir modificação de uma tabela em outra direta, sendo a principio somente para consulta para manter a integridade dos dados das tabelas relacionadas.

Chave Primaria:

Será o nome de uma coluna, onde terá somente dados não nulos e únicos, não podendo se repetir ocasionando identificação única de cada linha da coluna, o valor da chave ate pode se repetir desde que o valor relacionados não se repita. Esta coluna pode ser de incrementação automática, não sendo preciso incrementação com insert. A chave primaria sera referenciada a sua tabela, ou seja, não pode ter uma chave primaria com o nome de chave primaria que referência a uma tabela externa.

Chaves Estrangeiras:

Como citei anteriormente são relacionais para consulta e integridade basicamente, onde não podem fazer alterações diretas entre elas apesar de relacionadas pelos vínculos que a chave estrangeira trás, onde neste campo fica a chave primaria que vincula a outra tabela ou tabelas. Conforme avançamos neste estudo vamos ter exemplos, mas diferente da chave primaria a chave estrangeira tem critérios diferentes:

1. Pode ser nula em seu campo(NULL);
2. É um campo em uma tabela que faz referência a um campo que é chave primaria em outra tabela;
3. É possível ter mais de uma (ou nenhuma) em uma tabela

OBS.: Podemos ter chaves estrangeiras com valor nulo, gerando assim um registro órfão ou seja sem dados neste vinculo de registro que irei exemplificar, apesar que se faz necessário pra entendimento do banco como situação possível.

Vamos ao exemplo: se existe um condomínio com estacionamento, temos uma tabela com N moradores e outra tabela com N carros do estacionamento. Na tabela Estacionamento teremos uma chave estrangeira com ID_morador que sera uma chave primara da tabela Moradores, que neste caso sera o campo proprietário na tabela estacionamento; que nunca sera nula neste caso porque para haver um carro no estacionamento e obrigatório ter um proprietário, que neste caso sera um morador.

Já a premissa inversa não e fato, porque nem todo morador necessariamente tem um carro, portanto se houver uma chave estrangeira na tabela morador com campo ID_carro poderia ocorreria o valor null.

Agora irei codificar em SQL para exemplifica a ideia:

Segue o código da tabela Moradores sem chave estrangeira:

```
1. CREATE TABLE moradores
   {
2. ID_moradores INT PRIMARY KEY,
3. nome VARCHAR (255),
4. apartamento VARCHAR (255),
5. telefone_contato VARCHAR (255),
   };
```

No item 1 foi dado o comando para a criação da tabela moradores, já no item 2 onde se tem uma chave primaria da tabela ‘PRIMARY KEY’ que não pode ser repetida e/ou nula (NULL) ID_moradores que sera um inteiro ‘INT’, no item 3 a 5 temos as demais definições dos campos da tabela moradores sendo cada um um VARCHAR com ate 255 caracteres.

Segue o código da tabela Moradores com chave estrangeira.

```
1. CREATE TABLE moradores
   {
2. ID_moradores INT PRIMARY KEY,
3. nome VARCHAR (255),
4. apartamento VARCHAR (255),
5. telefone_contato VARCHAR (255),
6. ID_carro int,
7. CONSTRAINT fk_estacimorador FOREIGN KEY (ID_carro) REFERENCES
   estacionamento (ID_carro)
```

};

No item 1 foi dado o comando para a criação da tabela moradores, já no item 2 onde se tem uma chave primaria da tabela 'PRIMARY KEY' que não pode ser repetida e/ou nula (NULL) ID_moradores que sera um inteiro 'INT', no item 3 a 5 temos as demais definições dos campos da tabela moradores sendo cada um um VARCHAR com ate 255 caracteres, no item 6 temos o campo novo ID_carro, do tipo inteiro, ele sera o representante da nossa chave estrangeira que sera a chave primaria do carro que ela possa vir a ter ou não, necessariamente o morador tem um carro em caso negativo de posse de veiculo este campo recebe o NULL. No item 7 temos a definição da chave estrangeira propriamente dita. Observe que adicionamos uma CONSTRAINT chamada fk_estacimora (nome padrão: misto dos nomes das tabelas relacionadas com o prefixo fk) como FOREIGN KEY (chave estrangeira) e a associamos ao campo ID_carro da tabela moradores. Na mesma tabela temos a definição da palavra-chave (REFERENCES) na tabela estacionamento, especificamente o campo ID_carro da tabela estacionamento.

Neste exemplo que informei no campo ID_carro pode retorna NULL, porque como informado anteriormente não e obrigatório que o morador tenha um carro. Portanto vamos continuar com a confecção da tabela do estacionamento, onde irei codificar com a chave estrangeira, eliminando o código exemplo da tabela sem a chave estrangeira.

Segue o código da tabela Moradores com chave estrangeira.

1. CREATE TABLE estacionamento
   {
2. ID_carro INT PRIMARY KEY,
3. nome VARCHAR (255),
4. marca VARCHAR (255),
5. placa VARCHAR (255),
6. proprietário int,
7. CONSTRAINT fk_moraestaci FOREIGN KEY (proprietario) REFERENCES
   moradores (ID_moradores)
   };

O item 1 temos a criação da tabela estacionamento CREATE TABLE, no item 2 temos ID_carro que e a chave primaria de INT PRIMARY KEY, do tipo inteiro INT, dos itens 3 a 5 temos outros diversas colunas com características diferentes com o campo de ate 255 caracteres e de variável do tipo VARCHAR, no item 6 temos o FK(Foreign Key) denominado proprietário, de valor inteiro, que representa a chave estrangeira, no item 7

temos a definição de chave estrangeira propriamente dita, onde CONSTRAINT sera a restrição chamada fk_moraestaci (nome padrão: misto dos nomes das tabelas relacionadas com prefixo fk) como FOREIGN KEY (chave estrangeira) e a associamos ao campo proprietário da tabela estacionamento, onde sera definido a palavra-chave REFERENCES na tabela moradores, especificamente na coluna ID_moradores.

Com isto verificamos que não existe carro sem proprietário, que neste caso sera um morador, com a utilização da chave estrangeira vamos ver teremos integridade dos dados, porque no fim caso se tentar remover um dado da coluna ID_carro através da tabela moradores ocorra um erro e vice e versa.

Comando Unique:

Serve como um forma de restrição de duplicatas de dados em colunas, com isso teremos um identificador único podendo ter um valor nulo, colaborando com um controle.

Comando Join:

O join e um comando que combina os registros de uma ou mais tabelas em um banco de dados de forma não permanente.

Inner Join será o registro que corresponde unicamente em ambas tabelas creio que pode ser usado plenamente no caso de carro proprietário, segue o código:

```
select moradores.nome, moradores.apartamento, estacionamento.nome, estacionamento.placa
from moradores
  inner join estacionamento on moradores.ID_moradores = estacionamento.proprietario
   where estacionamento.proprietario like '%'; -- consegui fazer a busca por todos os proprietários com seus apartamento de todos os veículos.
```

Onde creio, o SELECT sera para seleção para a seleção dos campos ou colunas que iram aparecer na busca, o FROM sera onde ira ocorrer a busca dos campos do select na tabela que ira estar informada no from, no INNER JOIN sera da tabela estacionamento com a moradores que creio eu, onde o ON fara ou indicara a ligação da coluna da tabela moradores e da coluna ID_moradores com a tabela estacionamento e da coluna proprietário, onde eu quero através do comando WHERE, que irei especificar o critério de busca em conjunto com LIKE qualquer coisa no proprietário que ira me retornar campos com equivalência na tabela moradores.

Left Join ira me informar todos as informações da tabela que eu posicionar a esquerda que neste caso em especifico sera moradores:

select moradores.nome, moradores.apartamento, estacionamento.nome, estacionamento.placa

from moradores

left join estacionamento on moradores.ID_moradores = estacionamento.proprietario

where moradores.ID_moradores like '%'; -- consegui fazer a busca por todos os moradores mesmo sem carro. Onde creio, o SELECT sera para seleção para a seleção dos campos ou colunas que iram aparecer na busca, o FROM sera onde ira ocorrer a busca dos campos do select na tabela que ira estar informada no from, no LEFT JOIN que sera da tabela estacionamento com a moradores que estará a esquerda creio eu, onde o ON fara ou indicara a ligação da coluna da tabela moradores e da coluna ID_moradores com a tabela estacionamento e da coluna proprietário, onde eu quero através do comando WHERE, que irei especificar o critério de busca em conjunto com LIKE qualquer coisa ou seja todos os moradores independente de ter carro ou não.

Right Join ira informar todos as informações da tabela que eu posiciona a direita.

Full Outer Join vai retornar todas as linhas correspondente tanto a direita a esquerda.

Código em SQL com comentários:

Primeiro Banco Condominio

```
DROP DATABASE IF EXISTS condominio;
CREATE DATABASE condomínio;
USE condomínio;
CREATE TABLE moradores(
ID_moradores INT PRIMARY KEY AUTO_INCREMENT,
nome VARCHAR (255),
apartamento VARCHAR (255),
telefone_contato VARCHAR (255)
);
CREATE TABLE estacionamento
(
ID_carro INT PRIMARY KEY, -- não inseri o auto_increment e tenho que informar os ID's dos carros.
nome VARCHAR (255),
marca VARCHAR (255),
```

placa VARCHAR (255) unique, -- coloquei o comando unique apos os testes vou tentar inserir uma placa duplicada

proprietario int,

CONSTRAINT fk_moraestaci FOREIGN KEY (proprietario) REFERENCES moradores (ID_moradores)

);

INSERT INTO moradores (nome, apartamento, telefone_contato) VALUES

('Maria', '102', '333102'),

('João', '103', '333103'),

('Zeze', '104', '333104');

select ID_moradores, nome, apartamento, telefone_contato from moradores where nome like '%';

INSERT INTO estacionamento (ID_carro, nome, marca, placa, proprietario) VALUES

(1, 'pampa', 'chevrolet', 'abc-12102', '1'),

(2, 'celta', 'chevrolet', 'bcd-23102', '1'),

(3, 'corsa', 'chevrolet', 'cde-34102', '1'),

(4, 'c3', 'citroem', 'def-12103', '2'),

(5, 'sandero', 'renault', 'efg-23103', '2'),

(6, 'corsa', 'chevrolet', 'fgh-12104', '3');

select ID_carro, nome, marca, placa, proprietario from estacionamento where placa like '%e%';

select ID_carro, nome, marca, placa, proprietario from estacionamento where nome like '%';

delete from moradores where ID_moradores=1; -- tentei deletar pra ver se funcionou a chave estrangeira--

insert into moradores (nome, apartamento, telefone_contato) values -- inserção de teste funcionou

('Flor', '105', '333105');

INSERT INTO moradores (nome, apartamento, telefone_contato) VALUES -- inseri estes dados por engano e ficou duplicado

('Maria', '102', '333102'),

('João', '103', '333103'),

('Zeze', '104', '333104');

update moradores set apartamento=apartamento+4, telefone_contato=telefone_contato+4 where Id_moradores > 4; -- correção dos dados duplicados

select ID_moradores, nome, apartamento, telefone_contato from moradores where nome like '%';

delete nome, apartamento, telefone_contato, marca, placa, proprietario
from moradores, estacionamento
where moradores.ID_moradores = estacionamento.proprietario = 3; -- não consegui fazer o delete do morador ID 3 que esta atribuido a um unico veiculo

select moradores.nome, moradores.apartamento, estacionamento.nome, estacionamento.placa
from moradores
left join estacionamento on moradores.ID_moradores = estacionamento.proprietario
where estacionamento.proprietario like '%'; -- consegui fazer a busca por todos os proprietarios com seus apartamento de todos os veiculos.

INSERT INTO estacionamento (ID_carro, nome, marca, placa, proprietario) VALUES
(8, 'livina', 'nissan', 'abc-12102', '7'); -- esqueci de atualizar a planilha vou dar um drop no banco e tenta depois.
INSERT INTO estacionamento (ID_carro, nome, marca, placa, proprietario) VALUES
(8, 'livina', 'nissan', 'abc-12108', '7'); -- fazendo a correção da placa funcionou perfeitamente.

Segundo Banco IFRS

DROP DATABASE IF EXISTS ifrs;
CREATE DATABASE ifrs;
USE ifrs;
CREATE TABLE alunos(-- não e possivel fazer esta tabela antes de fazer a tabela disciplina.

```
ID_matricula INT PRIMARY KEY AUTO_INCREMENT,
nome VARCHAR (255),
data_nasc VARCHAR (255),
disciplinas VARCHAR (255),
Turno varchar(255),
CONSTRAINT fk_alunosdisciplina FOREIGN KEY (disciplinas) REFERENCES disciplinas(ID_disciplina)
);
CREATE TABLE disciplinas
(
ID_disciplina varchar(255) PRIMARY KEY, -- não inseri o auto_increment e tenho que informar os ID's dos carros.
nome_disciplina VARCHAR (255),
nome_aluno VARCHAR (255),
matricula_aluno int
);
INSERT INTO alunos (nome, data_nasc, disciplinas, Turno) VALUES
('Maria', '12/12/2000', 'prog1','tarde'),
('Joao', '13/12/2000', 'bd1','manha'),
('Zeze', '14/12/2000', 'log1','noite');

INSERT INTO disciplinas (ID_disciplina, nome_disciplina, nome_aluno, matricula_aluno) VALUES
('prog1', 'programção', 'Maria','123'),
('bd1', 'banco', 'Joao','234'),
('log1', 'logica', 'Zeze','345');
```

Considarações Finais:

Creio que apesar de superficial na forma de abordagem, se tonar uma poderosa ferramenta o uso do banco de dados e onde temos as tabelas que teremos chaves primarias e em alguns casos estrangeiras onde a primaria sera para identificador único como diz o trabalho, já estrangeira para ligação e integridade dos bancos ligados, neste caso de estudo temos o banco condomínio e o banco ifrs. Os JOIN tentam através dos exemplos demostrar as relações para consulta do das tabelas que se cruzam. Neste caso creio que veio a demonstrar como de forma bem eficiente.