



IN

INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

PY - SQL III

01 Cláusulas e operadores (in e not in)

02 Relacionamento (Primary key, Foreign Key)

03 Inner join



PY - SQL III

CLÁUSULA FROM



```
1 select * from tabela  
2 /* Sintaxe */
```

A cláusula **FROM** é usada para especificar de qual tabela selecionar ou excluir os dados.

PY - SQL III

CLÁUSULA WHERE



```
1 select * from tabela  
2 where condicao;  
3 /* Sintaxe básica*/  
4  
5 select * from Usuario where id ='123'  
6 /* Exemplo*/
```

A cláusula **WHERE** filtra um conjunto de resultados para incluir apenas os registros que atendem a uma condição especificada



```
1 select * from tabela group by coluna;  
2 /* Sintaxe básica */  
  
3  
4 select * from tabela group by coluna desc;  
5 /* Ordenando em ordem descrescente */
```

PY - SQL III

CLÁUSULA ORDER BY

A cláusula **ORDER BY** é usada para classificar o conjunto de resultados em ordem crescente ou descrescente.

A cláusula ordena os resultados em ordem crescente por padrão

Para ordenar os resultados em ordem descrescente usa-se a palavra **DESC**

PY - SQL III

Operador IN e NOT IN



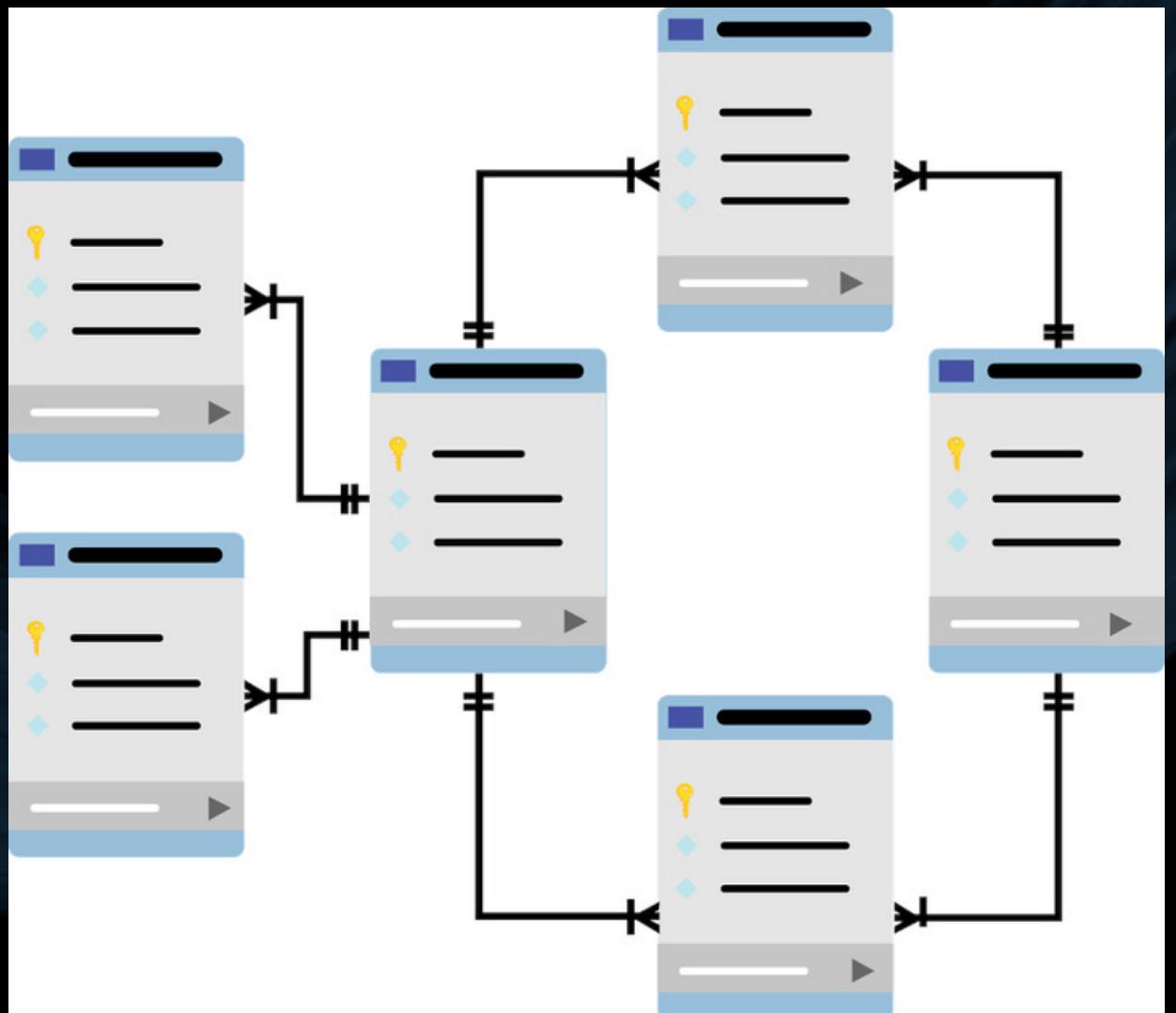
```
1 select * from tabela  
2 where coluna in (valores);  
3 /* Sintaxe básica */  
  
4  
5 select * from tabela  
6 where coluna not in (valores);  
7 /* Sintaxe básica */
```

O operador **IN** é um operador de associação que retorna valores de uma lista ou subconsulta, o operador compara um valor a um conjunto de valores e retorna verdadeiro se o valor pertencer e falso se o valor não pertencer.

O operador **NOT IN** possui as mesmas características do operador **IN**, a diferença está entre o resultado, se determinador valor **não** pertencer ao conjunto, este operador retorna verdadeiro.

PY - SQL III

Relações entre tabelas



No exercício da aula passada, fizemos tabelas para um sistema de uma escola. Sabemos que um professor ministra uma disciplina, um aluno faz parte de uma turma, no exemplo da aula passada criamos as tabelas, como fazemos para relacioná-las?

Relações são as associações estabelecidas entre duas ou mais tabelas. As relações são baseadas em campos comuns de mais de uma tabela, envolvendo muitas vezes **chaves primárias e estrangeiras**.

Quando criamos as relações entre as tabelas, estamos seguindo a **normalização de banco de dados**.

PY - SQL III

Chave primária

Uma tabela pode ter apenas uma chave primária. Uma chave primária consiste em um ou mais campos que identificam exclusivamente cada registro armazenado na tabela.

Coluna que possui
regra de chave primária

cliente	id_cliente	nome	data_nasc	sexo
	1	José	1978-04-21	m
	2	Maria	1980-10-17	f
	3	João	1995-08-12	m
	4	Pedro	1990-03-18	m

Muitas vezes, há um número de identificação exclusivos, como um número de ID, um número de série ou um código que funciona como uma chave primária.

Por exemplo, você pode ter uma tabela de clientes onde cada cliente tem um número de ID exclusivo. O campo de ID do cliente é a chave primária da tabela Clientes.

Quando uma chave primária contém mais de um campo, ele é geralmente composto por campos existentes que, juntos, fornecem valores exclusivos. Por exemplo, você pode usar uma combinação de sobrenome, nome e data de nascimento como chave primária em uma tabela sobre pessoas.

PY - SQL III

Exemplo criando uma tabela com chave primária



```
1 CREATE TABLE Pessoa  
2 (  
3     id_Pessoa integer PRIMARY KEY,  
4     nome varchar(255),  
5     endereco varchar(255),  
6     idade float  
7 );  
8 /* Exemplo*/
```

PY - SQL III

Chave estrangeira



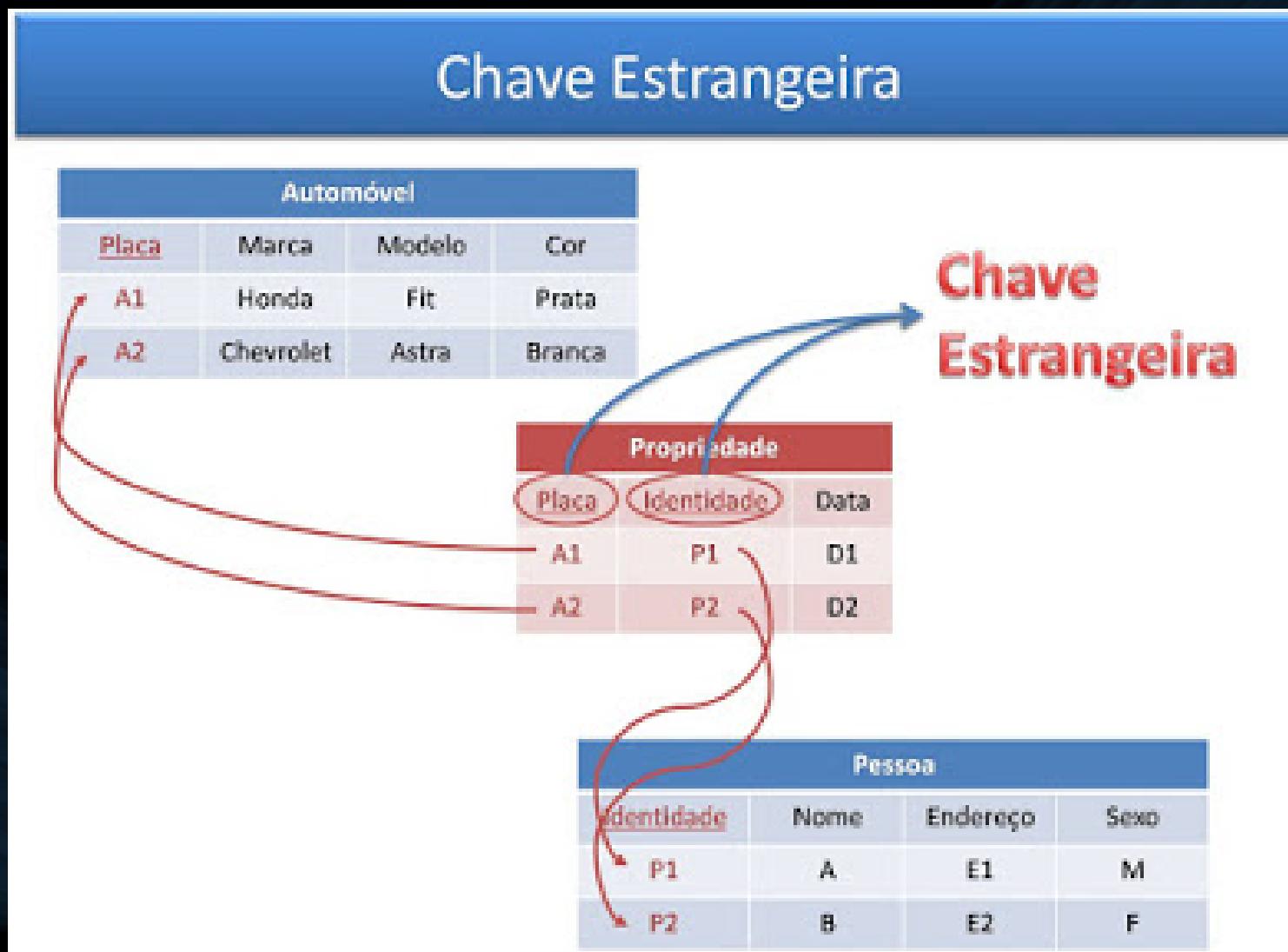
Uma tabela pode ter uma ou mais chaves estrangeiras.

Uma chave estrangeira contém valores que correspondem a valores na chave primária de outra tabela. Por exemplo, você pode ter uma tabela Pedidos na qual cada pedido tem um número de identificação do cliente que corresponde a um registro em uma tabela Clientes.

O campo de identificação do cliente é uma chave estrangeira da tabela Pedidos.

PY - SQL III

Restrição (Constraint)



No SQL, uma **Constraint** é uma regra ou condição que você pode definir e impor em uma tabela ou coluna para garantir a integridade dos dados e manter a consistência em um banco de dados.

As restrições são usadas para impor restrições aos dados que podem ser inseridos, atualizados ou excluídos em uma tabela. Eles ajudam a impor a validade, a precisão e os relacionamentos dos dados entre as tabelas.

Por exemplo, se há uma tabela chamada **Pedidos** e nela há uma chave estrangeira **id_cliente**, que origina da tabela **Cientes**, se tentarmos excluir a tabela **Cientes**, a Constraint não irá permitir, pois, desta maneira haveriam registros na tabela **Pedidos** sem **Cientes**, o que fere a integridade dos dados.

PY - SQL III

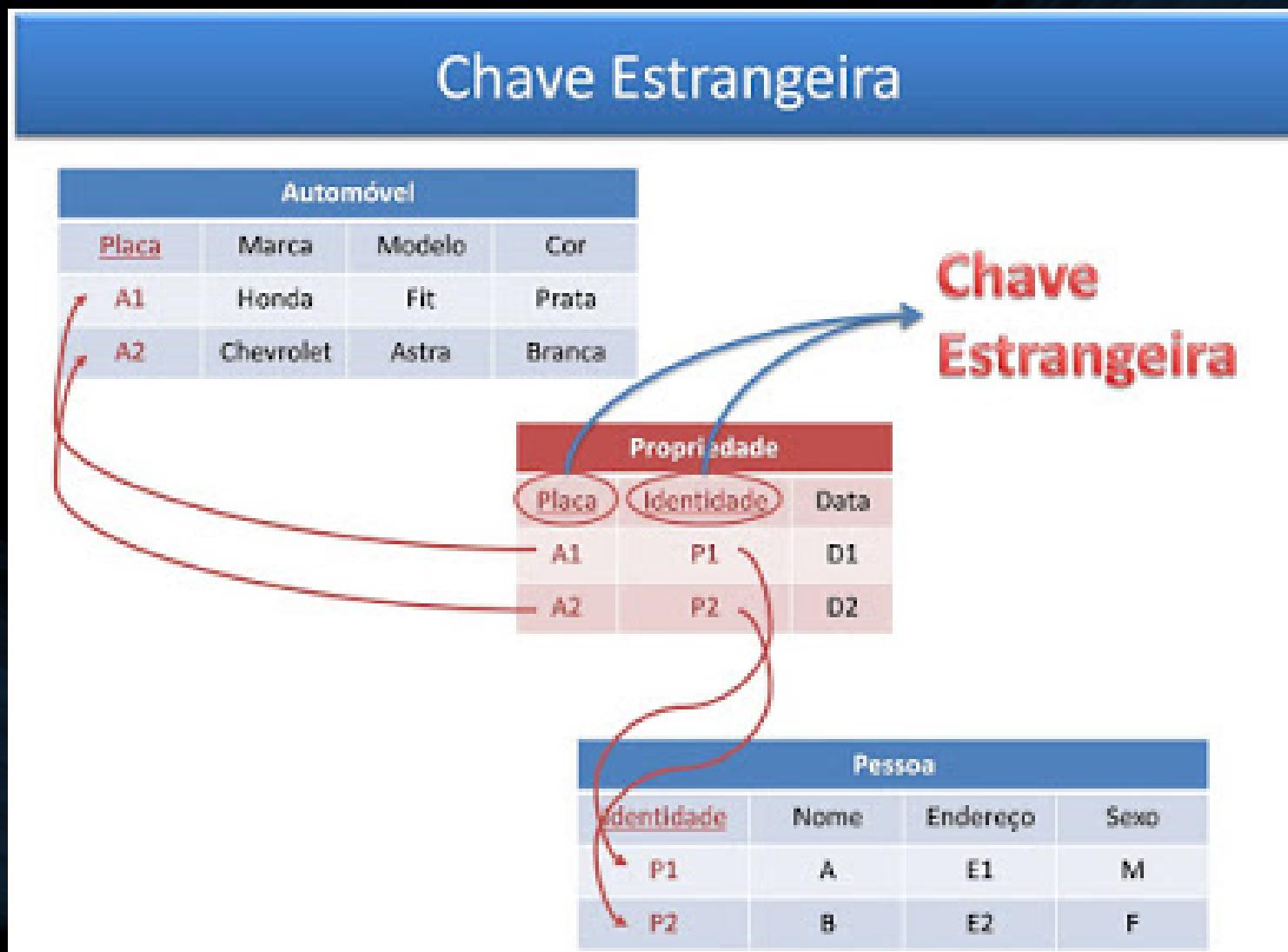
Exemplo criando uma tabela com chave estrangeira



```
1 CREATE TABLE Pessoa(
2     id_pessoa integer PRIMARY KEY,
3     nome varchar(100),
4     cpf varchar(11)
5 );
6
7 CREATE TABLE Carro
8 (
9     ID_Carro integer PRIMARY KEY AUTOINCREMENT,
10    Nome varchar(255),
11    Marca varchar(255),
12    ID_Pessoa integer,
13    CONSTRAINT fk_PesCarro FOREIGN KEY (ID_Pessoa) REFERENCES Pessoa (id_Pessoa)
14 );
15 /* Exemplo de chave estrangeira */
```

PY - SQL III

Relacionamento



A correspondência de valores entre campos de chave forma a base de uma relação de tabelas.

A relação de tabelas é usada para combinar os dados das tabelas relacionadas. Por exemplo, suponhamos que você tenha uma tabela Clientes e uma tabela Pedidos. Na primeira, cada registro é identificado pelo campo de chave primária, o ID.

Para associar cada pedido a um cliente, você adiciona um campo de chave estrangeira à tabela Pedidos que corresponda ao campo ID da tabela Clientes e, em seguida, cria uma relação entre as duas chaves.

Ao adicionar um registro à tabela Pedidos, você usa um valor para a identificação do cliente que é proveniente da tabela Clientes.

Sempre que desejar visualizar qualquer informação sobre o cliente de um pedido, você usará a relação para identificar quais dados da tabela Clientes correspondem a quais registros na tabela Pedidos.

PY - SQL III

Mão no código

Para este exercício, refatore o código da atividade anterior.

Faça o relacionamento da tabela professor com a tabela disciplina

Relacione também a tabela aluno com a tabela turma



PY - SQL III

Cardinalidade

Grau do Relacionamento: É o número de ocorrências de uma entidade A que está associado com ocorrências de outra entidade B.

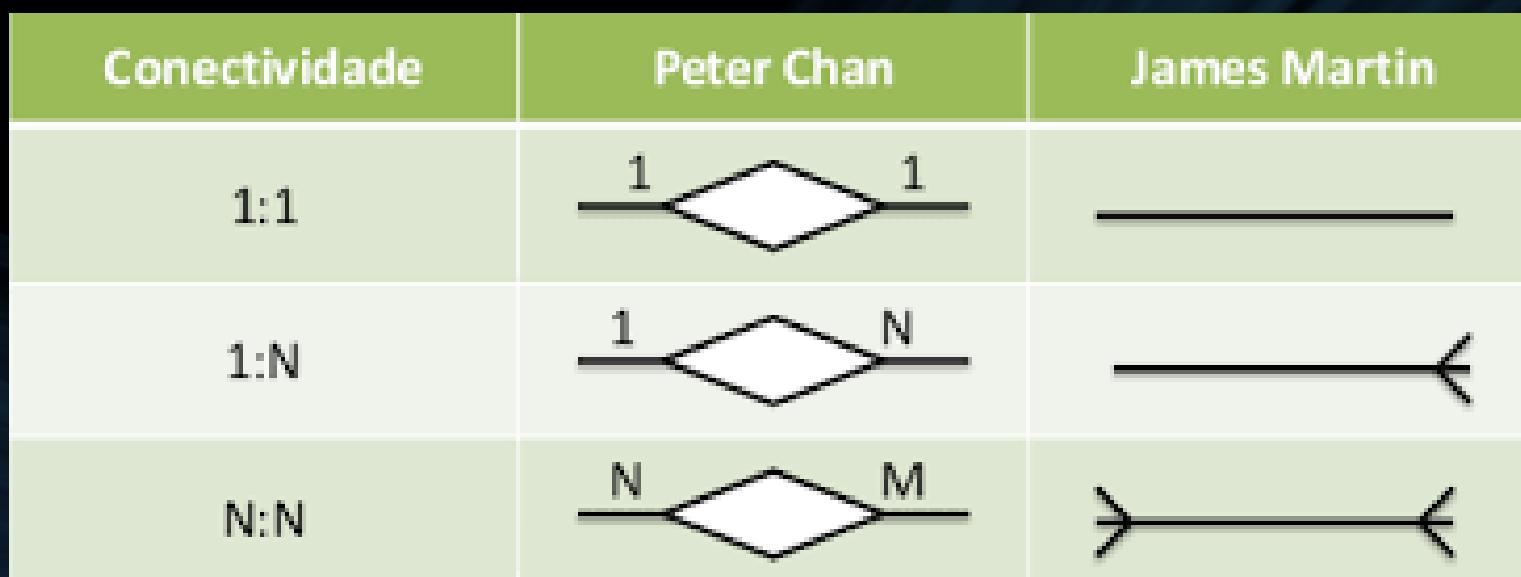
Grau do Relacionamento também é chamado de **Cardinalidade**

Há três graus de relacionamento:

- Relacionamento de Um para Um
- Relacionamento de Um para Muitos ou Muitos para Um
- Relacionamento de Muitos para Muitos

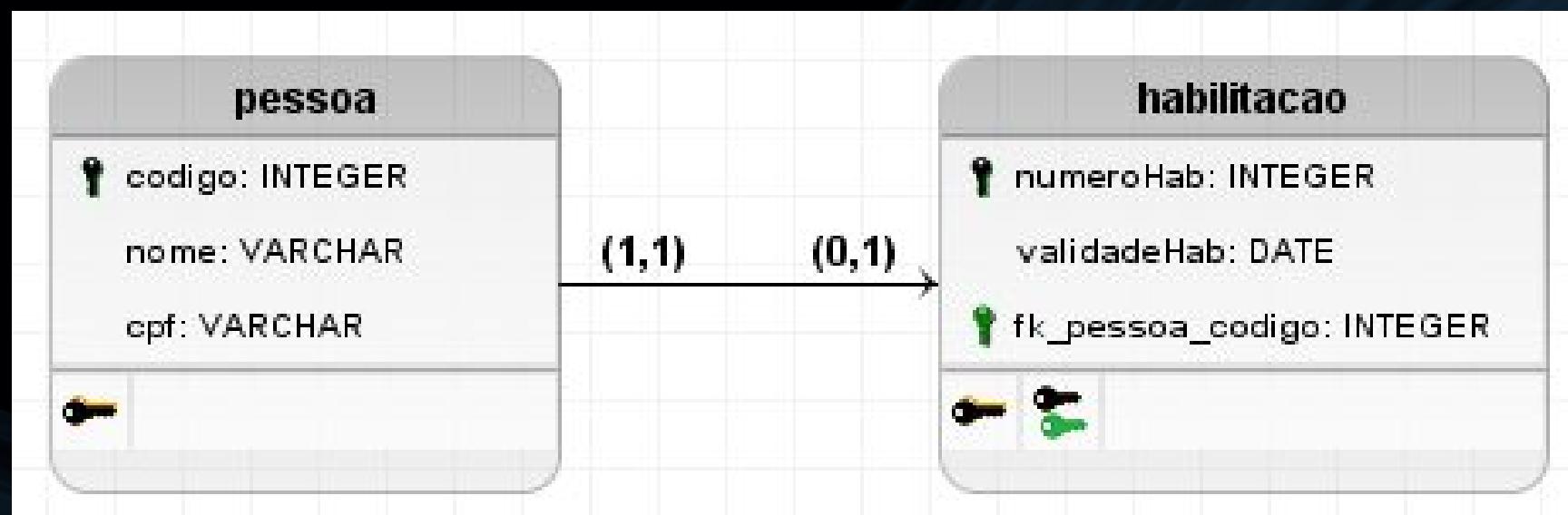
Nas cardinalidades temos:

- Cardinalidade Máxima: Número máximo de vezes que uma entidade A pode ocorrer em B. Pode assumir o valor de 1 ou N (inúmeras vezes).
- Cardinalidade Mínima: Número mínimo de vezes que uma entidade A pode ocorrer em B. Pode assumir o valor de 0 ou 1.



PY - SQL III

Relacionamento de Um para Um (1 x 1)



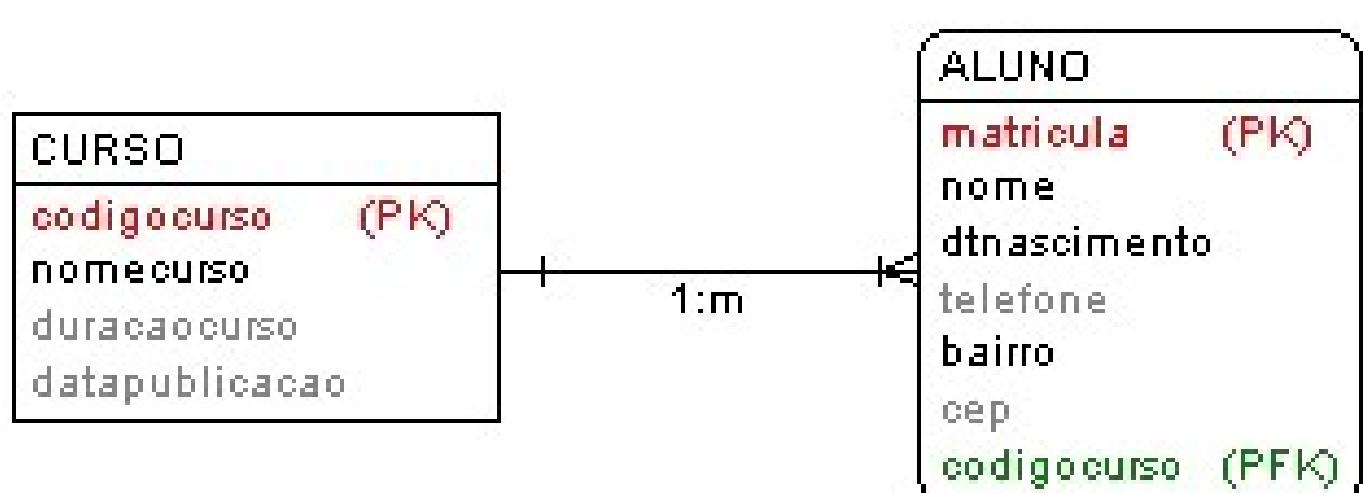
Cada elemento de uma entidade A relaciona se com um e somente um elemento de outra entidade B.

Nesse tipo de relacionamento a cardinalidade mínima influênci na modelagem.

PY - SQL III

Relacionamento de Um para Muitos (1 x n)

[1.1]

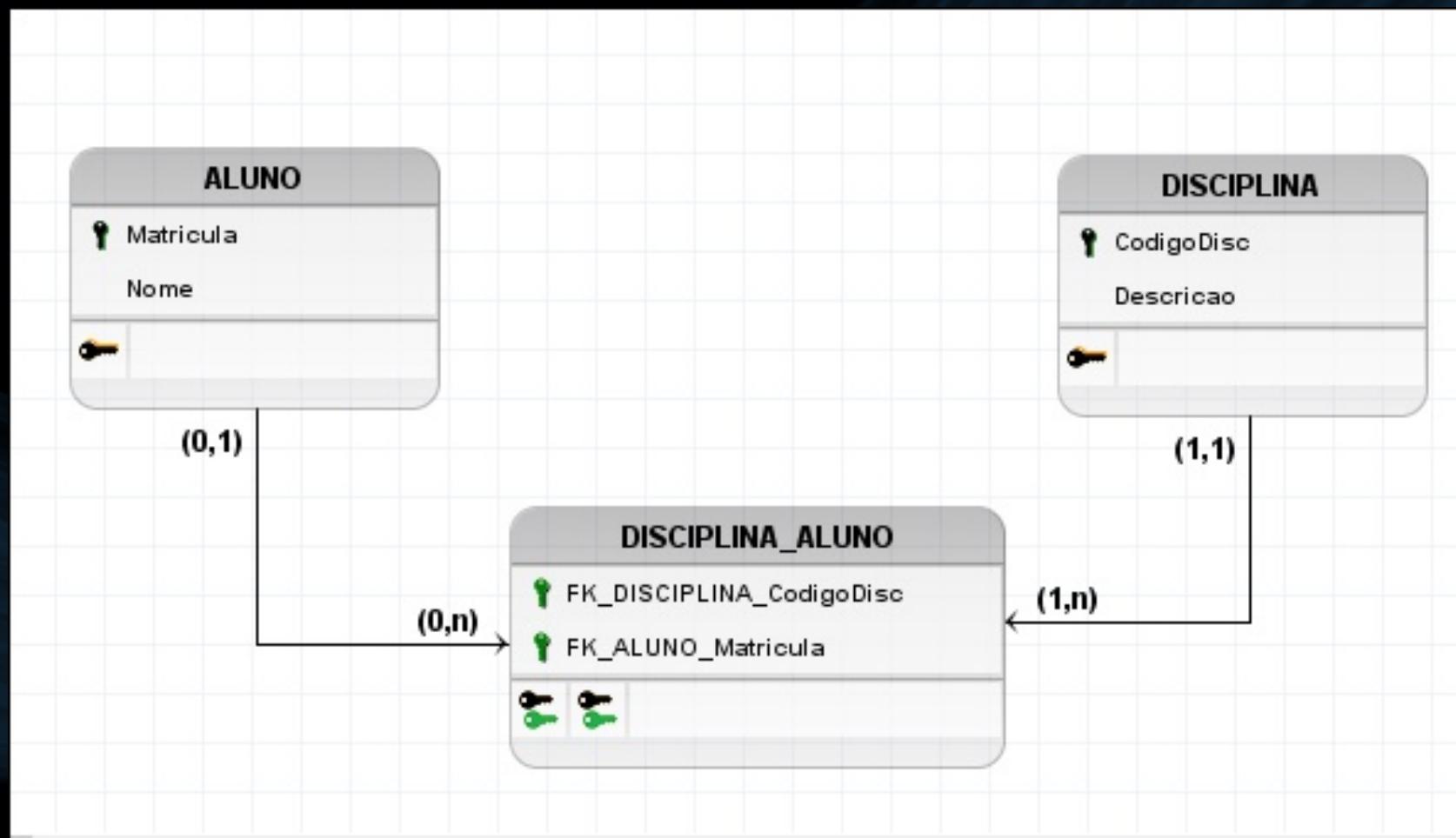


No relacionamento **Um para muitos** ou **Muitos para Um**, o elemento de uma entidade A pode se relacionar com mais de um elemento de outra entidade B.

Durante este tipo de relacionamento a ordem influencia no resultado de onde o atributo de referência também chamado de chave estrangeira (este atributo referencia a chave primária de outra tabela) será adicionado. Normalmente onde há cardinalidade máxima N.

PY - SQL III

Relacionamento Muitos para Muitos ($n \times n$)



Em um relacionamento muitos-para-muitos no SQL, vários registros em uma tabela são associados a vários registros em outra tabela.

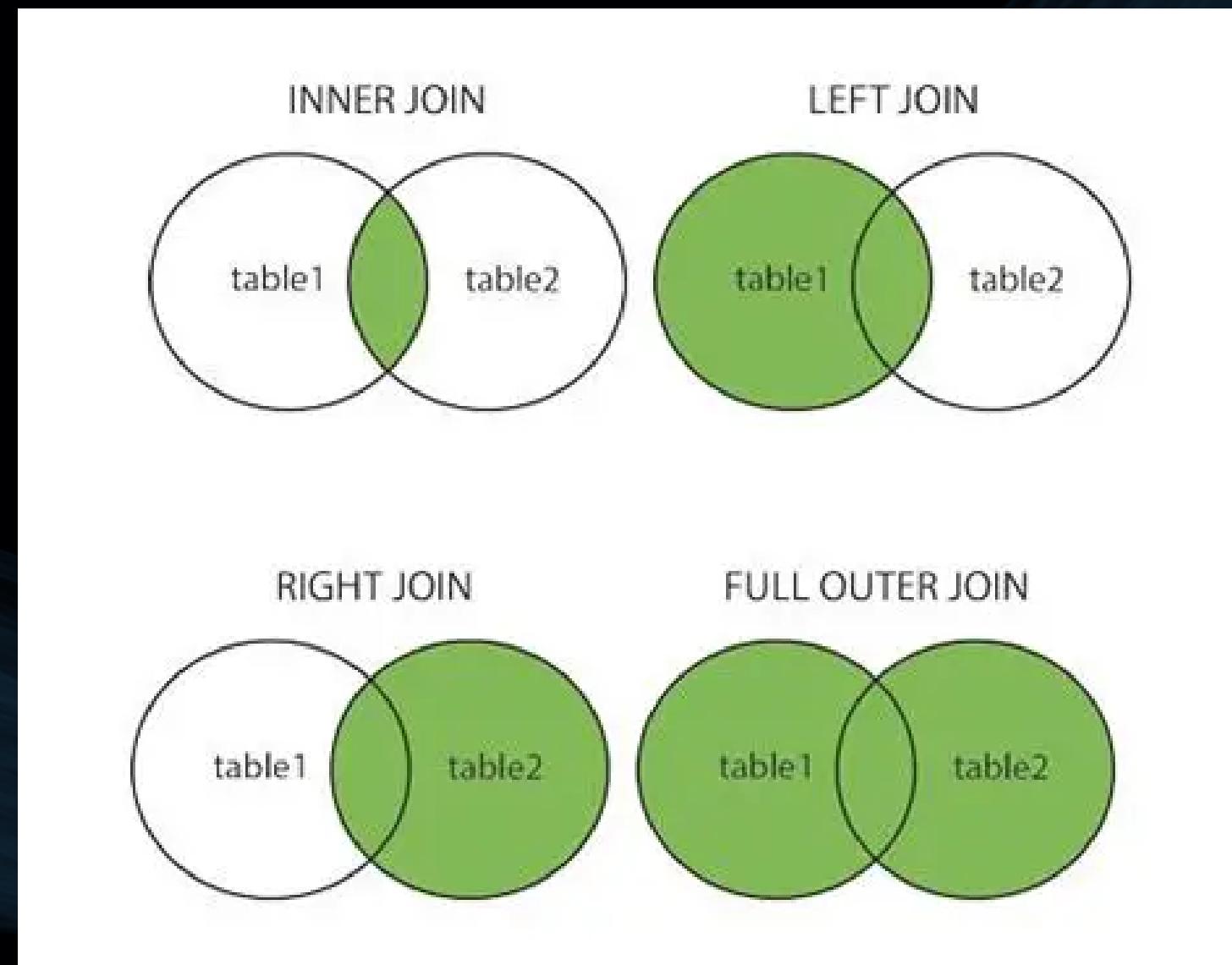
Esse tipo de relacionamento ocorre quando as entidades de ambas as tabelas podem ter várias entidades relacionadas na outra tabela.

Para estabelecer um relacionamento muitos-para-muitos, uma tabela intermediária, geralmente chamada de tabela de junção ou tabela associativa, é usada.

Essa tabela de junção contém as associações entre as duas tabelas principais, armazenando seus valores de chave primária.

PY - SQL III

Inner join



Conforme nossos bancos de dados ficam mais complexos, as consultas a eles também ficam. É comum que para determinadas consultas, o resultado não esteja em apenas uma tabela. Nesse cenário, precisamos juntar duas ou mais tabelas, para fazer isso, utilizamos os comandos: **INNER JOIN**, **LEFT JOIN**, **RIGHT JOIN** e **FULL OUTER JOIN**.

Um **INNER JOIN** é um tipo de operação de junção em SQL que combina linhas de duas ou mais tabelas com base em uma coluna ou colunas relacionadas. Ele retorna apenas as linhas correspondentes onde a condição de junção é satisfeita.

PY - SQL III

Inner join

Considere duas tabelas: "Clientes" e "Pedidos".

A tabela "Clientes" contém informações sobre os clientes, e a tabela "Pedidos" contém informações sobre os pedidos feitos pelos clientes, ambas as tabelas têm uma coluna em comum, "cliente_id", que serve como chave para relacionar as duas tabelas.

Neste exemplo, o comando **INNER JOIN** é usada para especificar a operação de junção.

O comando **ON** é usado para especificar a condição de junção, que indica que a correspondência deve ser baseada na igualdade da coluna "cliente_id" em ambas as tabelas.

O resultado do "inner join" incluirá apenas as linhas em que os valores "cliente_id" correspondem em ambas as tabelas "Clientes" e "Pedidos".



```
1 SELECT * FROM Clientes  
2 INNER JOIN Pedidos  
3 ON Clientes.cliente_id = Pedidos.cliente_id;
```

PY - SQL III

Mão no código

Você foi contratado para montar o banco de dados para uma concessionária.

Nesse banco de dados será possível salvar informações sobre a montadora, o modelo, o carro e a venda

Para a montadora, é necessário que seja salvo: um identificador, o nome e o país

Para o modelo, é necessário um identificador, o nome do modelo e qual é a montadora

Para um carro o banco deve persistir um id, qual é o modelo, o ano, a cor.

Os atributos necessários para armazenar informações sobre as vendas: id da venda, o id do carro vendido e a data da venda.



PY - SQL III

Mão no código

Alimente o banco de dados com 4 modelos: Fusca, Celtinha, Gol bolinha e Uno 97.

A concessionária fez 5 vendas: 1 Fusca, 2 Celtinhas um Gol bolinha e um Uno 97.

Armazene todas essas informações e então faça um script SQL que retorne somente as vendas do Celtinha e do Uno 97.





IN

INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

IN PARABÉNS! VOCÊ TERMINOU A AULA 12 DO MÓDULO DE PYTHON