**WHERE IN**

O comando “WHERE IN” vai fazer uma busca indo de acordo com cada valores que for passado dentro do “IN” verificando se os valores passados estão dentro do conjunto ou não, fazendo a busca por uma coluna e filtrando os diversos valores que você deseja encontrar.

Exemplo:

SELECT id,  
 nome,  
 duracao  
FROM aula  
WHERE duracao IN (60, 120, 180, 240, 300);

O SELECT vai retornar uma tabela onde a coluna duração é igual a 60, 120, 180, 240 ou 300 segundos.

Ou você pode fazer uma consulta equivalente ao exemplo anterior, só que usando o OR, que diferente do AND, precisa que pelo menos um retorne verdadeiro para retornar algo na consulta.

Exemplo:

SELECT   
 id,  
 nome,  
 duracao  
FROM aula  
WHERE duracao = 60  
OR duracao = 120  
OR duracao = 180  
OR duracao = 240  
OR duracao = 300;

**ORDER BY**

O comando “Order By” serve para você ordenar uma coluna do SELECT em crescente ou descrescente.

Exemplo:

SELECT id,  
 nome,  
 duracao  
FROM aula  
WHERE duracao IN (60, 120, 180, 240, 300)  
ORDER BY  
 duracao DESC, id ASC ;

O SELECT vai retornar uma tabela onde a coluna “duracao” vai ser decrescente e a coluna do “id” vai ser crescente(padrão);

ASC = crescente.  
DESC = descrescente.

**LIMIT E OFFSET**

O comandos “LIMIT” serve para você limitar a quantidade de linha que você quer que a tabela retorne.

Exemplo:

SELECT id,  
 nome,  
 duracao  
FROM aula  
LIMIT 10;

Aqui a tabela vai retornar com o limite de 10 linhas. Já o “OFFSET” serve para você fazer os deslocamento em “páginas”.

Exemplo:

SELECT id,  
 nome,  
 duracao  
FROM aula  
LIMIT 10  
OFFSET 10;

Aqui a tabela vai retornar com no máximo 10 linhas, começando pelo id 11, se colocar “OFFSET 20”, a tabela vai retornar a partir do id 21.  
Outra forma de usar o “LIMIT” e o “OFFSET” é com o “OFFSET” oculto.

Exemplo:

SELECT id,  
 nome,  
 duracao  
FROM aula  
LIMIT 15, 30;

Aqui ele vai retonar a tabela com o limite de 30 linhas, começando a partir do 16.

**Teoria 1 - N**

Base do Relacionamento no Banco de Dados Relacional

1 - N (1 para muitos)

Exemplo:

Tabela Curso e tabela Aula:

1 curso tem várias aulas, e uma aula pertence a vários cursos.

Onde a tabela curso tem as colunas id(Primary Key), nome. E a tabela Aula tem as colunas id(Primary Key), nome e id.curso(Foreign Key).

**Curso**

|  |  |
| --- | --- |
| **ID** | **NOME** |
| 1 | JAVA 16 |
| 2 | WEB MODERNO |
| 3 | REACT |

**Aula**

|  |  |  |
| --- | --- | --- |
| **ID** | **NOME** | **ID.CURSO** |
| 1 | ——— | 2 |
| 2 | ——— | 2 |
| 3 | ——— | 2 |

A informação sempre estará do lado “N”, ou seja, a informação de relacionamento sempre vai estar na tabela “N”, que é a que contem varias informações relacionadas a 1 outra informação.

Exemplo SQL:

CREAT DATABASE IF NOT EXISTS relacionamentos;

CREATE TABLE IF NOT EXISTS usuario(  
 id INT UNSIGNED PRIMARY KEY AUTO\_INCREMENT,  
 nome VARCHAR(120) NOT NULL,  
 email VARCHAR(120) NOT NULL  
);

CREATE TABLE IF NOT EXISTS postagem(  
 id INT UNSIGNED PRIMARY KEY AUTO\_INCREMENT,  
 titulo VARCHAR(120) NOT NULL,  
 conteudo TEXT NOT NULL,  
 id\_usuario INT UNSIGNED,  
 FOREIGN KEY (id\_usuario) REFERENCES usuario(id)  
);

Nesse exemplo, a chave estrangeira é a coluna “id\_usuario”, e está apontando para o id do usuário da tabela usuario, e o tipo “TEXT” do “conteudo” serve para quando vai ser um conteúdo muito grande que passa do limite do tipo VARCHAR.

**Teoria 1 - 1**

Tabela Responsavel e tabela Endereco :

Um responsável tem um endereço, e um endereço pertence a um responsável.

Onde a tabela responsavel tem as colunas id(Primary Key), nome. E a tabela endereco tem as colunas id(Primary Key), logradouro e id.responsavel (Foreign Key).

Em um relacionamento 1 para 1, para decidir de que lado vai ficar a chave estrangeira, depende do sistema, se o peso maior vai ser do lado do responsável(dono do endereço), ou se vai ser no endereço em si, então pode variar de qual lado a chave estrangeira deve ficar.

**Responsável**

|  |  |
| --- | --- |
| **ID** | **NOME** |
| 1 | —— |
| —— | —— |
| 2010 | —— |
| 3056 | —— |

**Endereco**

|  |  |  |
| --- | --- | --- |
| **ID** | **LOGRADOURO** | **ID.RESP** |
| 1 | —— | 7 |
| 2 | —— | 3056 |
| 3 | —— | 2010 |
| 4 | —— | 5080 |

Para que não aja duplicação, que o relacionamento 1 para 1 não se torne um relacionamento 1 para N, basta colocar uma restrição de “UNIQUE” na coluna “ID.RESP”, ou seja, colocando uma CONSTRAIN de UNIQUE na coluna “ID.RESP” não haverá mais duplicação de chave estrangeira.

Exemplo SQL:

CREATE TABLE IF NOT EXISTS funcionario(  
 id INT UNSIGNED PRIMARY KEY AUTO\_INCREMENT,  
 tipo ENUM(‘professor’, ‘vendedor’) NOT NULL,

nome VARCHAR(120) NOT NULL  
 email VARCHAR(120) NOT NULL  
);

CREATE TABLE IF NOT EXISTS professor(  
 id INT UNSIGNED PRIMARY KEY AUTO\_INCREMENT,  
 bio TEXT NOT NULL,  
 lattes VARCHAR(120),  
 id\_funcionario INT UNSIGNED NOT NULL UNIQUE,  
 FOREIGN KEY (id\_funcionario) REFERENCES funcionario(id)  
);

CREATE TABLE IF NOT EXISTS vendedor(  
 id INT UNSIGNED PRIMARY KEY AUTO\_INCREMENT,  
 bonus INT NOT NULL,  
 meta DECIMAL(7, 2) NOT NULL,  
 id\_funcionario INT UNSIGNED NOT NULL UNIQUE,  
 FOREIGN KEY (id\_funcionario) REFERENCES funcionario(id)  
);

**Teoria N - N**

Tabela Tio e tabela Sobrinho :

Um tio tem vários sobrinhos, e um sobrinho tem vários tios.

Onde a tabela Tio tem as colunas id(Primary Key), nome. E a tabela Sobrinho tem as colunas id(Primary Key), nome.

Tio

|  |  |
| --- | --- |
| **ID** | **NOME** |
| 1 | Enzo |
| 2 | Catarina |
| 3 | Ana |
| 4 | —— |

Sobrinho

|  |  |
| --- | --- |
| **ID** | **NOME** |
| 1 | Xico |
| 2 | Maria |
| 3 | Socorro |
| 4 | —— |

Sobrinho\_Tio

|  |  |
| --- | --- |
| **ID\_SOB** | **ID\_TIO** |
| 1 | 1 |
| 1 | 3 |
| 2 | 3 |
| 3 | 2 |

Em um relacionamento N para N(muitos para muitos), para fazer o relacionamento, deve se quebrar em uma terceira tabela de 1 para N. Onde o ID da tabela sobrinho vira uma FOREIGN KEY na tabela Sobrinho\_Tio e o ID da tabela tio vira uma FOREIGN KEY na tabela Sobrinho\_Tio fazendo assim com que ID\_SOB e ID\_TIO se tornem uma chave primaria composta. E você poderá ter dados repetidos na coluna ID\_SOB e dados repetidos na coluna TIO\_ID, mas não poderá ter dados repetidos na mesma coluna e em varias linhas.

Exemplo SQL:

CREATE TABLE IF NOT EXISTS ator(  
 id INT UNSIGNED PRIMARY KEY AUTO\_INCREMENT,  
 nome VARCHAR(120) NOT NULL  
 );

CREATE TABLE IF NOT EXISTS filme(  
 id INT UNSIGNED PRIMARY KEY AUTO\_INCREMENT,  
 nome VARCHAR(120) NOT NULL  
);

CREATE TABLE IF NOT EXISTS ator\_filme(  
 id\_autor INT UNSIGNED NOT NULL,  
 id\_filme INT UNSIGNED NOT NULL,  
 nota\_atuacao DECIMAL(4, 2) NOT NULL DEFAULT 0,  
 PRIMARY KEY(id\_autor, id\_filme),  
 FOREIGN KEY (id\_autor) REFERENCES ator(id),  
 FOREIGN KEY (id\_filme) REFERENCES filme(id)  
);

**Relacionamento vs Paradigma**

PK = Primary Key  
FK = Foreign Key

1 - N -> PK -> FK

1 - 1 -> PK -> FK + UNIQUE

N - M -> 2x 1 - N

Banco de dados tem apenas relacionamento unidirecional.