Oi, pessoal!

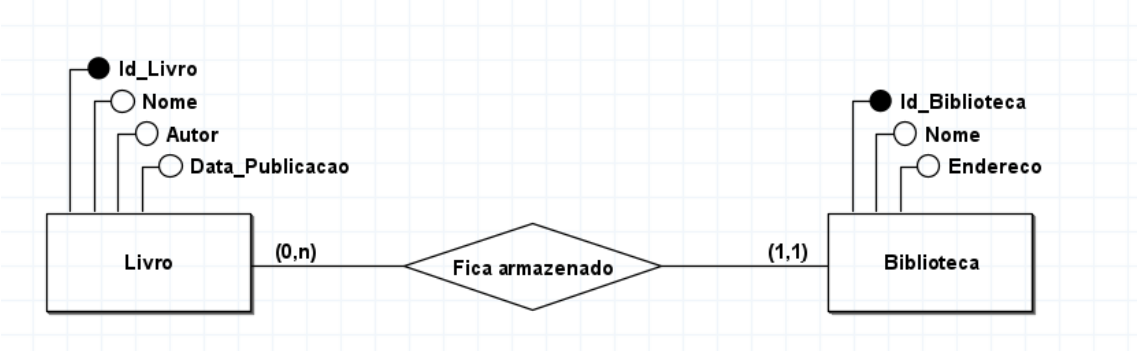
Segue uma revisão breve de alguns tópicos da matéria. É possível que alguns de vocês estejam no mesmo barco que eu já estive, então eu incluí alguns temas que tive certa dificuldade quando estava aprendendo a matéria pela primeira vez. Obs: Isso aqui não é para ser uma revisão geral para a prova, somente uma ajudinha!

Lembrando a todos que estou disponível para tirar dúvidas, só entrar em contato comigo, **Tiago Bisolo Prestes**, via Discord (**bptigas**) ou WhatsApp (**41 99962-5234**).

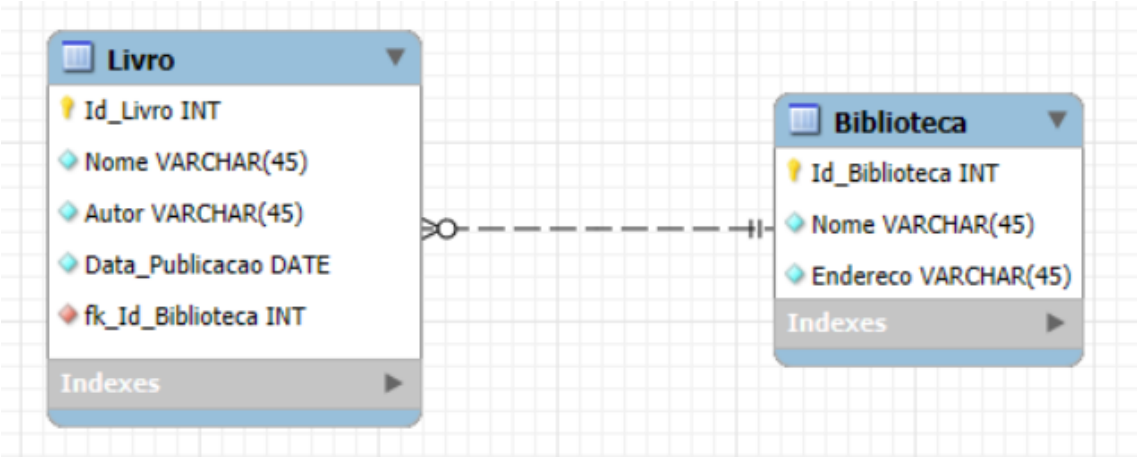
## Transformação de MER (Modelo Entidade Relacionamento) para Modelo Relacional – Caminho de Ida

Para facilitar a transformação, é importante identificar:

- Chaves primárias.
- Relacionamento entre as entidades.
- Se haverá chave estrangeira. Lembrando que a chave estrangeira (FK) fica para o lado onde existe alguma cardinalidade N.



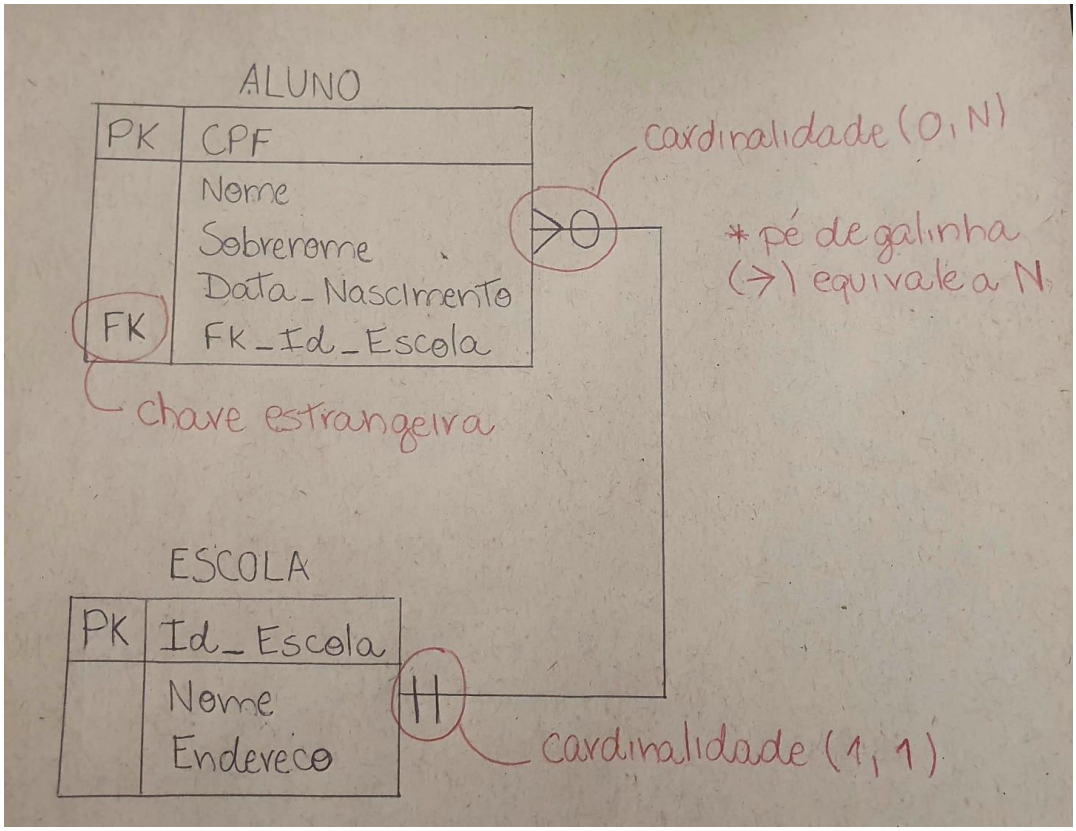

Nesse MER, as chaves primárias são Id_Livro e Id_Biblioteca. Considerando as cardinalidades, observa-se um relacionamento 1:N. Nesse sentido, um livro fica armazenado somente em uma biblioteca e em uma biblioteca fica armazenado zero ou N livros. Como o livro possui cardinalidade N, a chave estrangeira (FK) será armazenada dentro da tabela dele. Logo, o modelo relacional ficará assim:



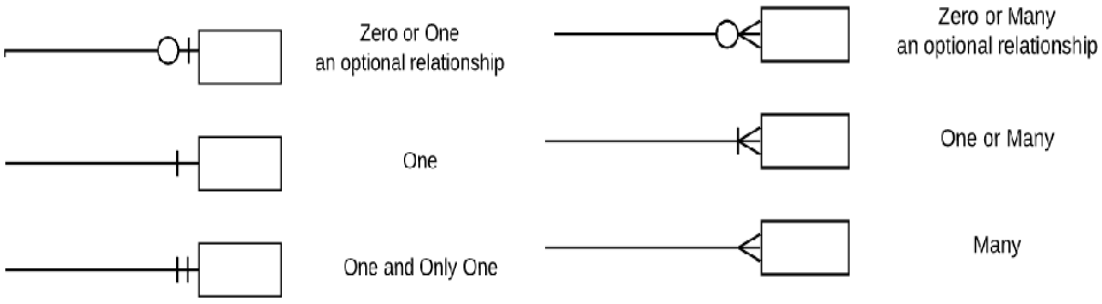

## Transformação de Modelo Relacional para MER (Modelo Entidade Relacionamento) – Caminho de Volta

Para fazer o caminho de volta, ou seja, do modelo relacional para o MER, é importante identificar:

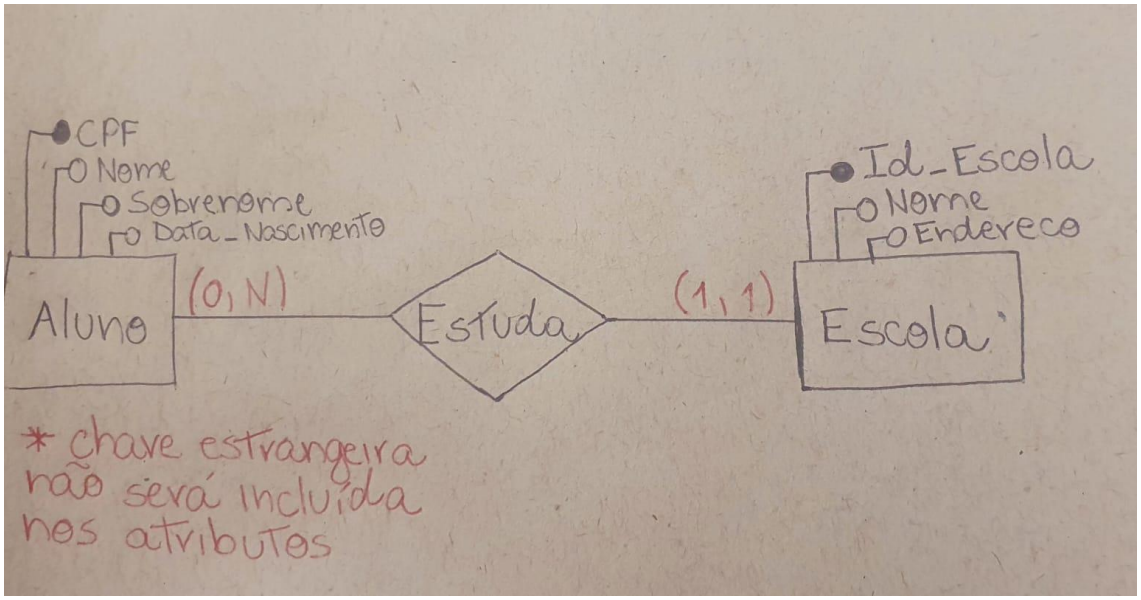
- Chaves primárias.
- As cardinalidades das entidades.
- A chave estrangeira. Digo isso pois a chave estrangeira (FK) não será representada explicitamente no MER, logo é preciso ter atenção para não incluir ela nos atributos da entidade.



As cardinalidades para cada entidade podem ser identificadas a partir das seguintes convenções (o ideal é saber isso de cabeça):



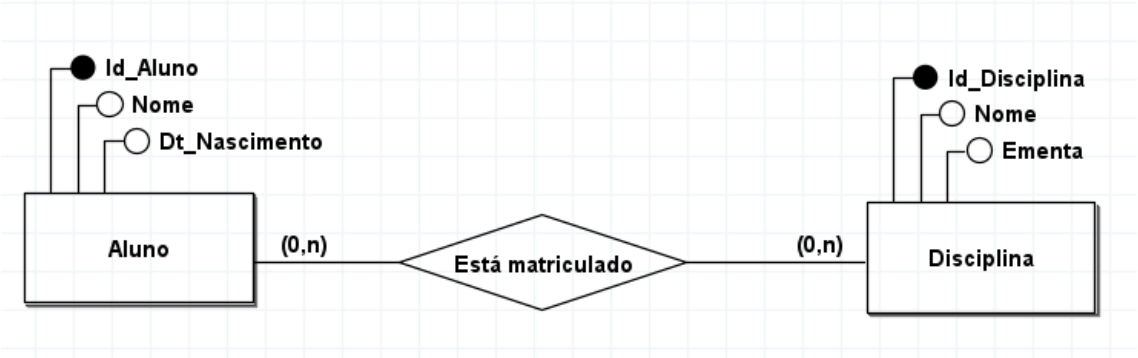

Com as cardinalidades "traduzidas" para a forma numérica e as chaves (primárias e estrangeiras) identificadas, podemos traçar o seguinte MER:



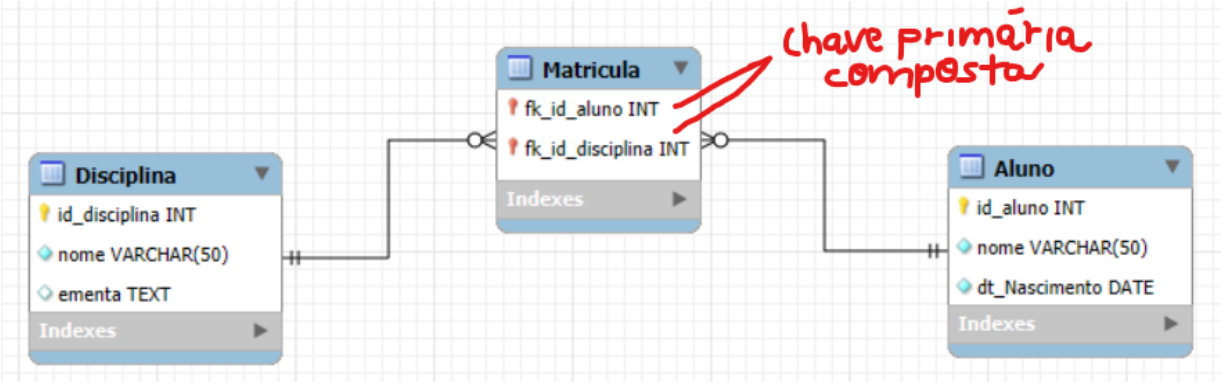

Lembrando, no MER a chave estrangeira não é incluída nos atributos e as cardinalidades são representadas sob forma numérica (1,1; 1,N; N,N).

## Relacionamento N:N (muitos para muitos)

Ao identificar um relacionamento N:N em um MER (Modelo Entidade Relacionamento), é preciso tomar cuidado na hora de desenhar o modelo relacional correspondente. Digo isso pois, ao traçar o modelo relacional, o relacionamento N:N requer que seja criado, além das tabelas para as entidades, uma tabela adicional que irá mapear os relacionamentos. Exemplificando, vamos considerar o seguinte MER:

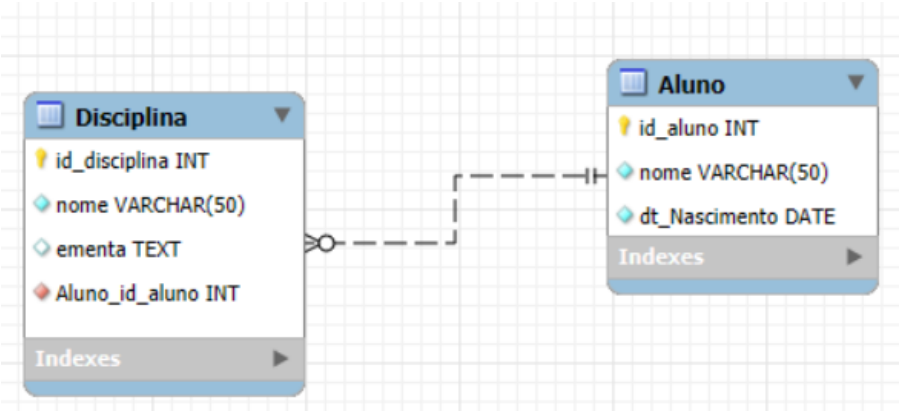



Leitura do digrama: um aluno está matriculado em zero ou N disciplinas e em uma disciplina estão matriculados zero ou N alunos. Sendo assim, temos cardinalidade N para ambas as entidades e uma relação N:N (muitos para muitos). A partir disso, podemos formar o seguinte modelo relacional:

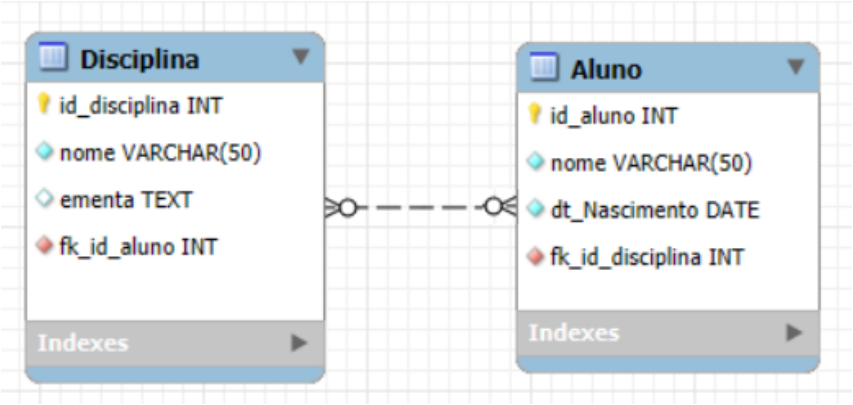



Observa-se que, além das tabelas das entidades (aluno e disciplina), foi criada uma tabela adicional "Matricula". Esta tabela será responsável por armazenar as associações entre alunos e disciplinas, ou seja, por guardar todas as disciplinas que estão sendo cursadas por todos os alunos.

**Mas porque é preciso ter essa tabela adicional?** Se tentarmos fazer uma relação PK e FK sem essa tabela adicional, vou dar alguns exemplos do que poderia ocorrer:

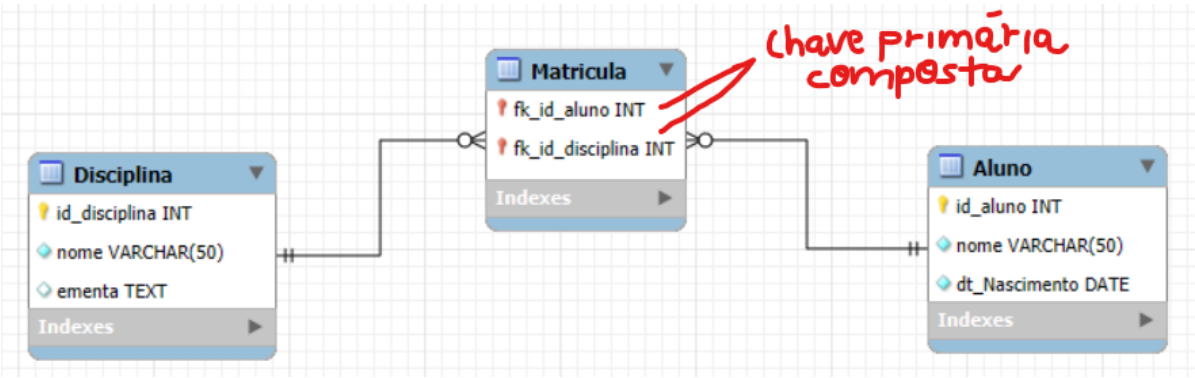


O que está de errado com o modelo anterior? Primeiro, ele diz que uma disciplina pode ser feita somente por um aluno, indo contra a lógica do MER inicial. O segundo erro é que, neste modelo, a existência de uma disciplina depende da existência de pelo menos um aluno, também ferindo a lógica proposta pelo MER inicial. Vamos observar outro modelo que tenta evitar a criação de uma tabela adicional:

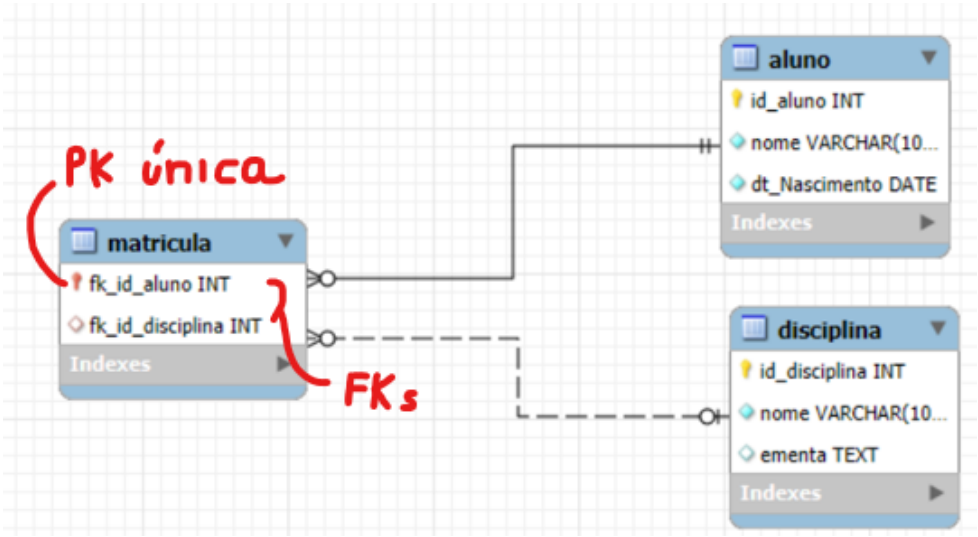



O que está de errado com o modelo anterior? Tudo. Ele está dizendo que a existência de um aluno depende da existência de uma disciplina e vice-versa, ferindo a lógica do nosso MER inicial, onde a existência das entidades é independente.

**Enfim, voltando para o modelo relacional correto:**




Agora que entendemos a justificativa da tabela adicional, vamos entender por que a tabela "Maricula" possui chave primária composta. **A chave primária é composta, pois dessa forma é possível fazer registros de múltiplos alunos em múltiplas disciplinas.** Vamos supor que a chave primária não fosse composta, e fosse somente a fk_Id_Aluno, da seguinte maneira:




Por que a relação PK (chave primária) e FK (chave estrangeira) acima estaria errada? Como nesse caso a PK da tabela "Matricula" é somente o ID do aluno, só conseguiríamos ter uma disciplina matriculada para cada aluno. Se tentássemos matricular um aluno em mais de uma disciplina, o MySQL iria impedir a operação acusando um erro de entrada duplicada na tabela, devido a repetição da fk_id_aluno em mais de um registro. O mesmo ocorreria se a tabela "Matricula" tivesse somente a fk_id_disciplina como PK.

Sendo assim, aqui a chave primária composta é uma alternativa que nos permite ter multiplicidade. Isso ocorre, pois, a chave primária composta somente acusaria de entrada duplicada se seu par de valores (nesse caso, fk_id_aluno e fk_id_disciplina) já estivessem presentes em algum registro na tabela "Matricula". Entretanto, nesse cenário, o acuso de entrada duplicada estaria correto e atuando a favor da integridade do banco de dados.

## Subconsulta

Uma **subconsulta** é uma consulta embutida dentro de outra consulta, de forma aninhada, passando os resultados da consulta mais interna para a consulta mais externa. Vamos entender melhor com um exemplo:

Tabela Disciplina

PK

| id_discip | nome | ementa | creditos | periodo |
|---|---|---|---|---|
| 1 | Banco de Dados | NULL | 4 | 2 |
| 2 | Matemática Discreta | NULL | 4 | 3 |
| 3 | Raciocínio Algorítmico | NULL | 6 | 2 |
| 4 | Realidade aumentada | NULL | 4 | 4 |
| 5 | Realidade Virtual | NULL | 4 | 5 |

Tabela Turma

| PK | | | FK | FK |
|---|---|---|---|---|
| id_turma | ano | semestre | id_discip | id_prof |
| 1 | 2020 | 1 | 2 | 2 |
| 2 | 2020 | 2 | 1 | 2 |
| 6 | 2021 | 1 | 5 | 3 |
| 5 | 2022 | 2 | 4 | 3 |
| 3 | 2023 | 1 | 5 | 1 |
| 4 | 2023 | 2 | 3 | 1 |

Considere a seguinte query SQL:

```
SELECT t.ano, t.semestre, t.id_discip
FROM Turma AS t
WHERE t.id_discip IN (
        SELECT d.ID_discip
        FROM Disciplina AS D
        WHERE D.nome LIKE 'R%'
) ORDER BY t.ano;
```

Para conseguirmos identificar os dados retornados nessa subconsulta, fica mais fácil se separarmos os **selects**. Nesse sentido, vamos iniciar a análise a partir do select mais interno.

Este é o **select** mais interno:

```
SELECT d.ID_discip,
FROM Disciplina AS D
WHERE D.nome LIKE 'R%'
```

Este select retorna os registros da tabela Disciplina que possuem nome que se inicia com a letra “R”. São três registros, conforme a imagem:

| ID_discip |
|---|
| 3 |
| 4 |
| 5 |

O **select** mais externo (em vermelho) irá utilizar essas disciplinas retornadas para fazer uma associação (a partir do ID da disciplina) com os registros da tabela turma.

Sendo assim, o resultado da subconsulta será o seguinte:

| ano | semestre | id_discip |
|---|---|---|
| 2021 | 1 | 5 |
| 2022 | 2 | 4 |
| 2023 | 2 | 3 |
| 2023 | 1 | 5 |

Mas o que significa esses registros retornados? Basicamente, está sendo buscado as turmas que cursaram uma disciplina cujo nome se inicia com a letra “R”. Para fazer um paralelo com outros conceitos vistos anteriormente, essa subconsulta se comporta de forma semelhante a um INNER JOIN entre as tabelas Turma e Disciplina. Inclusive, poderíamos escrever essa query análoga da seguinte forma:

```
SELECT t.ano, t.semestre, t.id_discip

FROM Turma AS t INNER JOIN Disciplina as d

ON T.id_discip = d.id_discip

WHERE D.nome LIKE 'R%'

ORDER BY t.ano;
```