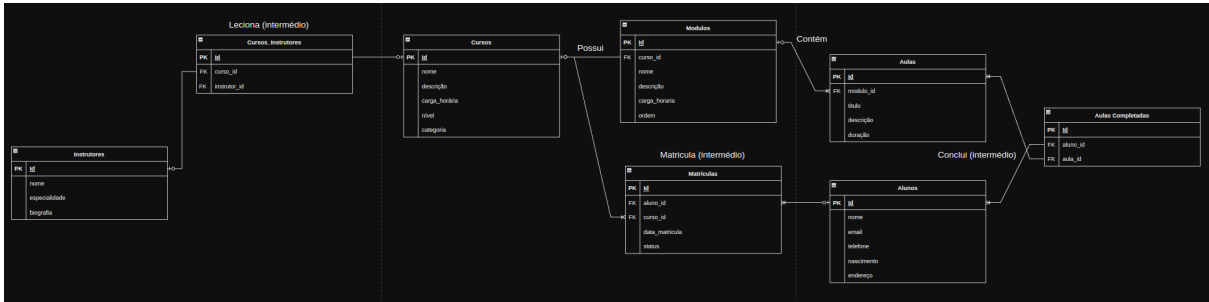


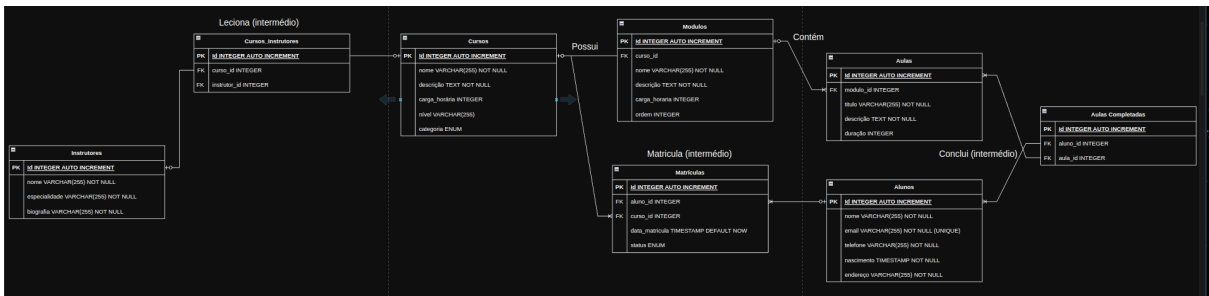
Exercício 1: Identificação de Entidades e Relacionamentos



Exercício 2: Justificativa da Modelagem Inicial

R: Modelei esse diagrama pensando na melhor forma de representar as relações entre os elementos de um sistema de cursos online. Coloquei os cursos no topo, já que eles são o ponto de partida. Cada curso pode ter vários módulos, que por sua vez contêm várias aulas. Os alunos podem se matricular em cursos e completar aulas, por isso usei tabelas separadas para matrículas e aulas completadas, representando esses relacionamentos. Também separei os instrutores para permitir que um mesmo profissional possa ministrar vários cursos, usando uma tabela intermediária. Essa estrutura ajuda a manter o sistema flexível.

Exercício 3 Transformação para o Modelo Relacional



Exercício 4: Justificativa do Modelo Relacional

Construí esse diagrama de pensando em representar as principais entidades do sistema e seus relacionamentos, sem entrar ainda nos detalhes técnicos de implementação. Essa abordagem facilita a visualização geral do funcionamento do banco de dados e ajuda a validar a estrutura lógica. Escolhi separar bem as entidades e usar tabelas intermediárias para relacionamentos muitos-para-muitos, como entre alunos e cursos, e entre instrutores e cursos. Isso torna o modelo mais flexível e normalizado.

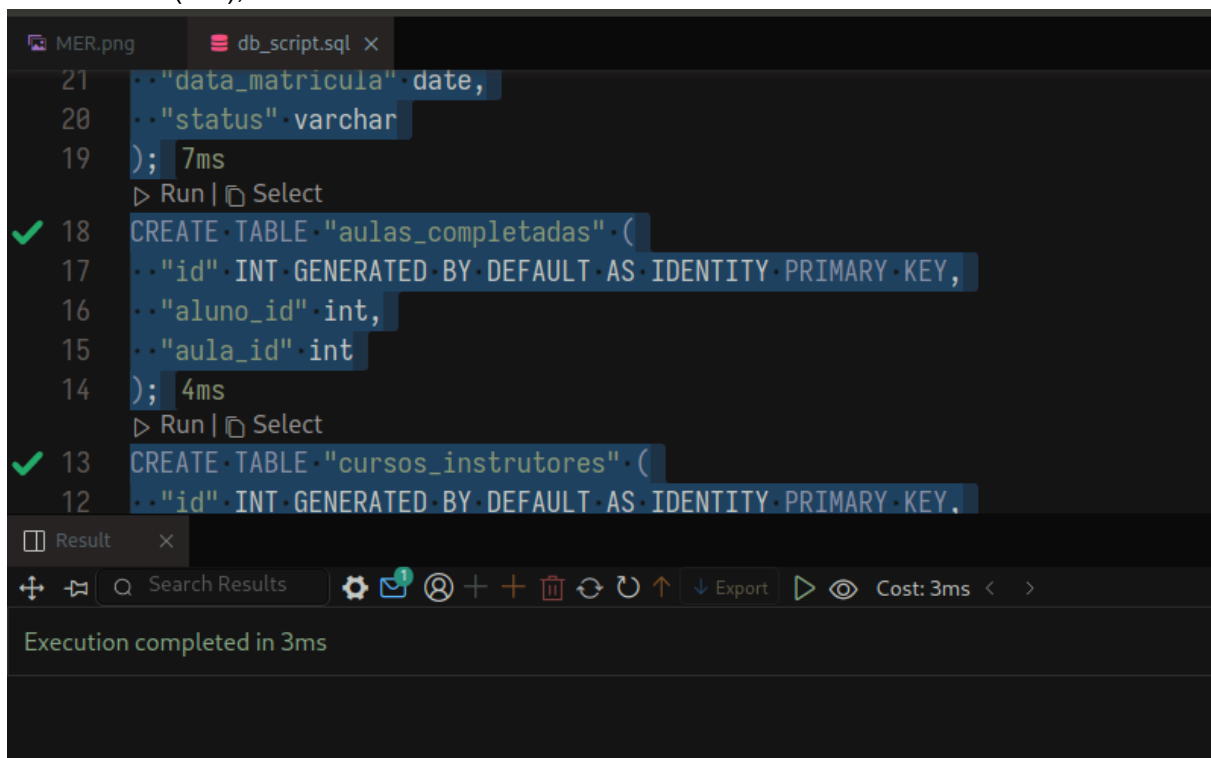
Exercício 5: Criação das Estruturas do Banco de Dados

```
CREATE TABLE "cursos" (  
  "id" INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,  
  "nome" varchar,  
  "descricao" text,  
  "carga_horaria" int,  
  "nivel" varchar,  
  "categoria" varchar  
);  
CREATE TABLE "modulos" (  
  "id" INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,  
  "curso_id" int,  
  "nome" varchar,  
  "descricao" text,  
  "carga_horaria" int,  
  "ordem" int  
);  
CREATE TABLE "aulas" (  
  "id" INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,  
  "modulo_id" int,  
  "titulo" varchar,  
  "descricao" text,  
  "duracao" int  
);  
CREATE TABLE "alunos" (  
  "id" INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,  
  "nome" varchar,  
  "email" varchar,  
  "telefone" varchar,  
  "nascimento" date,  
  "endereco" text  
);  
CREATE TABLE "instrutores" (  
  "id" INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
```

```

"nome" varchar,
"especialidade" varchar,
"biografia" text
);
CREATE TABLE "matriculas" (
  "id" INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  "aluno_id" int,
  "curso_id" int,
  "data_matricula" date,
  "status" varchar
);
CREATE TABLE "aulas_completadas" (
  "id" INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  "aluno_id" int,
  "aula_id" int
);
CREATE TABLE "cursos_instrutores" (
  "id" INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  "curso_id" int,
  "instrutor_id" int
);
ALTER TABLE "modulos" ADD FOREIGN KEY ("curso_id") REFERENCES "cursos" ("id");
ALTER TABLE "aulas" ADD FOREIGN KEY ("modulo_id") REFERENCES "modulos" ("id");
ALTER TABLE "matriculas" ADD FOREIGN KEY ("aluno_id") REFERENCES "alunos" ("id");
ALTER TABLE "matriculas" ADD FOREIGN KEY ("curso_id") REFERENCES "cursos" ("id");
ALTER TABLE "aulas_completadas" ADD FOREIGN KEY ("aluno_id") REFERENCES
"alunos" ("id");
ALTER TABLE "aulas_completadas" ADD FOREIGN KEY ("aula_id") REFERENCES "aulas"
("id");
ALTER TABLE "cursos_instrutores" ADD FOREIGN KEY ("curso_id") REFERENCES
"cursos" ("id");
ALTER TABLE "cursos_instrutores" ADD FOREIGN KEY ("instrutor_id") REFERENCES
"instrutores" ("id");

```



```

MER.png db_script.sql x
21  .."data_matricula" date,
20  .."status" varchar
19  ); 7ms
    ▶ Run | Select
✓ 18  CREATE TABLE "aulas_completadas" (
17  .."id" INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
16  .."aluno_id" int,
15  .."aula_id" int
14  ); 4ms
    ▶ Run | Select
✓ 13  CREATE TABLE "cursos_instrutores" (
12  .."id" INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    Result x
    Search Results
    Cost: 3ms
    Execution completed in 3ms

```

Exercício 6: Crie índices de restrições como unique, unicidade dupla, índices de data etc para melhorar a performance das tabelas do banco.

```

12 > Run
ALTER TABLE "alunos" ADD CONSTRAINT unique_email UNIQUE ("email"); 5ms
11 > Run
CREATE INDEX idx_alunos_nascimento ON "alunos" ("nascimento"); 4ms
10 > Run
CREATE INDEX idx_matriculas_data ON "matriculas" ("data_matricula"); 4ms
9 > Run
ALTER TABLE "matriculas" ADD CONSTRAINT unique_aluno_curso UNIQUE ("aluno_id", "curso_id"); 4ms
8 > Run
ALTER TABLE "cursos_instrutores" ADD CONSTRAINT unique_curso_instructor UNIQUE ("curso_id", "instructor_id"); 4ms
7 > Run
ALTER TABLE "aulas_completadas" ADD CONSTRAINT unique_aluno_aula UNIQUE ("aluno_id", "aula_id"); 4ms
6 > Run
ALTER TABLE "modulos" ADD CONSTRAINT unique_ordem_modulo_por_curso UNIQUE ("curso_id", "ordem"); 5ms
5 > Run
ALTER TABLE "aulas" ADD CONSTRAINT unique_titulo_por_modulo UNIQUE ("modulo_id", "titulo"); 5ms
4 > Run
CREATE INDEX idx_matriculas_aluno ON "matriculas" ("aluno_id"); 4ms
3 > Run
CREATE INDEX idx_matriculas_curso ON "matriculas" ("curso_id"); 6ms
2 > Run
CREATE INDEX idx_modulos_curso ON "modulos" ("curso_id"); 5ms
1 > Run
CREATE INDEX idx_aulas_modulo ON "aulas" ("modulo_id"); 5ms
98 > Run
CREATE INDEX idx_aulas_completadas_aluno ON "aulas_completadas" ("aluno_id"); 4ms
1 > Run
CREATE INDEX idx_aulas_completadas_aula ON "aulas_completadas" ("aula_id"); 4ms
2 > Run
CREATE INDEX idx_cursos_instrutores_instructor ON "cursos_instrutores" ("instructor_id"); 4ms

```

Result

Search Results

Cost: 4ms

Execution completed in 4ms

Exercício 7: Modificações Estruturais no Banco de Dados

```
ALTER TABLE "cursos" ALTER COLUMN "nome" TYPE varchar(255);
ALTER TABLE "alunos" ALTER COLUMN "telefone" SET NOT NULL;
```

```

130 > Run
ALTER TABLE "cursos" ALTER COLUMN "nome" TYPE varchar(255); 11ms
1 > Run
ALTER TABLE "alunos" ALTER COLUMN "telefone" SET NOT NULL; 1ms

```

Result

Search Results

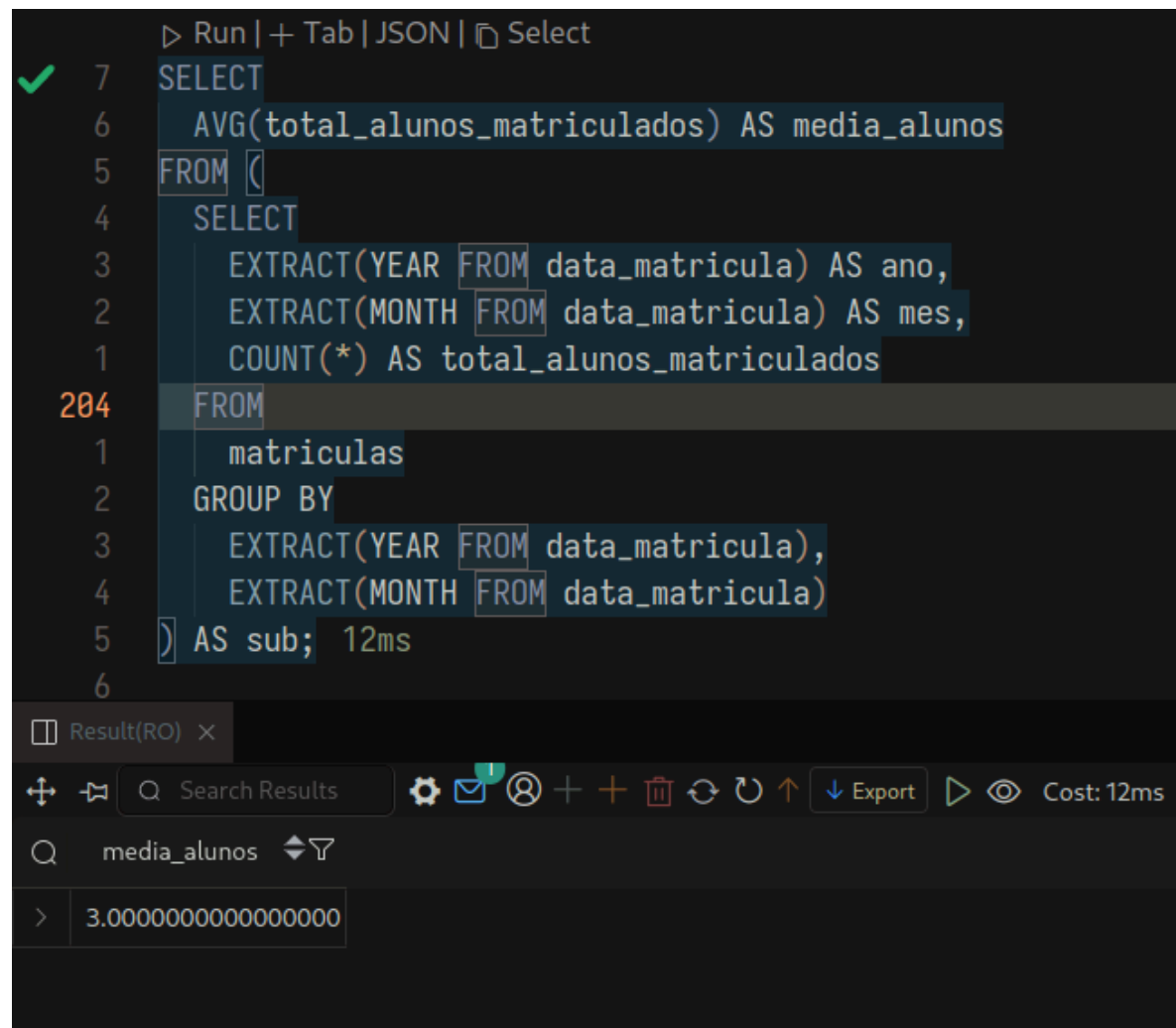
Cost: 1ms

Execution completed in 1ms

Exercício 8: Impacto dos Índices na Performance

Criei esses índices pensando em otimizar as operações mais comuns no sistema, como buscas por e-mail, filtragens por data de matrícula e relacionamentos entre tabelas. Índices em colunas que participam de JOINS, filtros e buscas frequentes reduzem bastante o tempo de resposta das consultas. Além disso, restrições de unicidade ajudam a manter a integridade dos dados, evitando duplicações desnecessárias e facilitando a consistência do sistema. Esses ajustes, mesmo simples, já fazem uma grande diferença na performance quando o volume de dados cresce.

Exercício 9: Média de Alunos Cadastrados por Período



The screenshot shows a SQL IDE interface with a query editor and a results pane. The query is as follows:

```
SELECT
  AVG(total_alunos_matriculados) AS media_alunos
FROM (
  SELECT
    EXTRACT(YEAR FROM data_matricula) AS ano,
    EXTRACT(MONTH FROM data_matricula) AS mes,
    COUNT(*) AS total_alunos_matriculados
  FROM matriculas
  GROUP BY
    EXTRACT(YEAR FROM data_matricula),
    EXTRACT(MONTH FROM data_matricula)
) AS sub;
```

The results pane shows a single row with the value 3.0000000000000000 for the column media_alunos. The cost of the query is 12ms.

media_alunos
3.0000000000000000

Exercício 10: Total de Alunos Presentes por Aula

```
2 SELECT
3     data_conclusao,
4     COUNT(*) AS total_alunos_presentes
5 FROM
6     aulas_completadas
7 GROUP BY
8     data_conclusao
9 ORDER BY
10    data_conclusao; 1ms
```

aulas_completadas x

Search Results

data_conclusao date total_alunos_presente

>	2025-04-10	1
>	2025-04-11	1
>	2025-04-12	1

Cost: 9ms < 1 > Total 3

Exercício 11: Distribuição de Alunos por Mês e Ano de Nascimento

```
6 SELECT
5     EXTRACT(YEAR FROM "nascimento") AS ano,
4     EXTRACT(MONTH FROM "nascimento") AS mes,
3     COUNT(*) AS quantidade_alunos
2 FROM "alunos"
1 GROUP BY ano, mes
229 ORDER BY ano, mes; 1ms
```

Result(RO) x

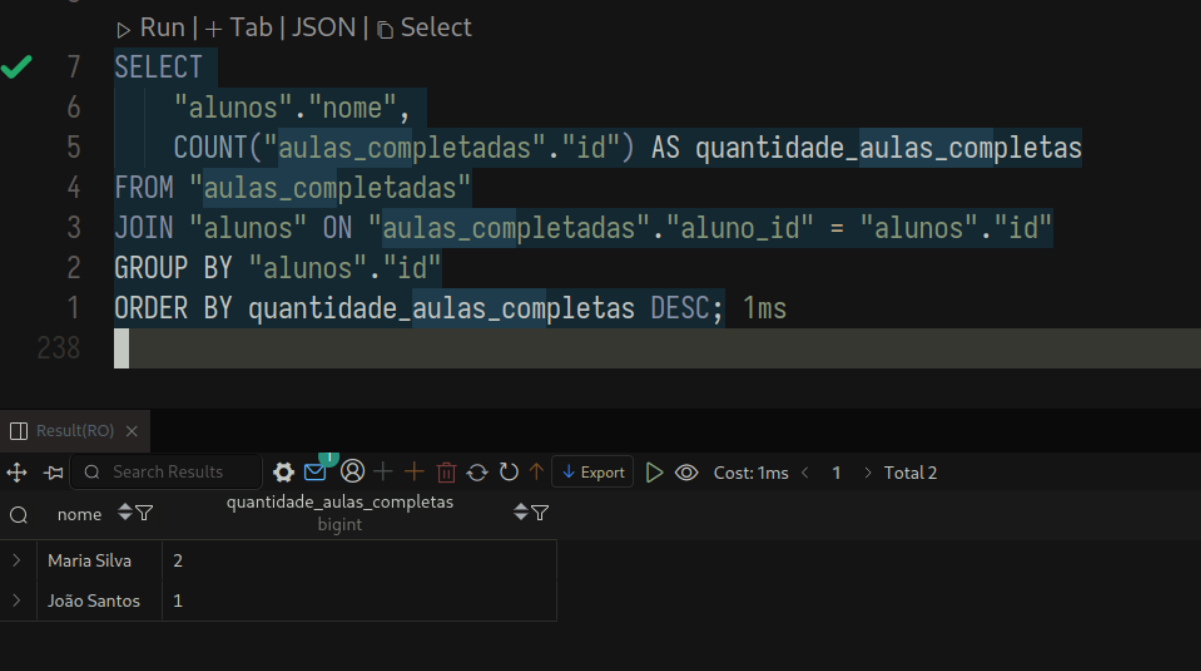
Search Results

ano mes quantidade_alunos bigint

>	1990	6	1
>	1995	2	1
>	2000	12	1

Cost: 1ms < 1 > Total 3

Exercício 12: Ranking de Alunos por Aulas Concluídas



The screenshot shows a SQL query in a dark-themed IDE. The query is as follows:

```
SELECT  
    "alunos"."nome",  
    COUNT("aulas_completadas"."id") AS quantidade_aulas_completas  
FROM "aulas_completadas"  
JOIN "alunos" ON "aulas_completadas"."aluno_id" = "alunos"."id"  
GROUP BY "alunos"."id"  
ORDER BY quantidade_aulas_completas DESC; 1ms
```

Below the query editor, the results are displayed in a table with two columns: 'nome' and 'quantidade_aulas_completas'. The table shows two rows: 'Maria Silva' with 2 completed classes and 'João Santos' with 1 completed class. The interface includes a search bar, a settings icon, and a cost indicator showing 'Cost: 1ms' and 'Total 2'.

nome	quantidade_aulas_completas
Maria Silva	2
João Santos	1