**PROJETO BANCO DE DADOS – 3 SEMESTRE**

**INSTRUE**

RONE FELIPE BENTO

OTHON RAFAEL GODOY

PROFESSORA: JULIANA FORIN PASQUINI MARTINEZ

FATEC – SÃO JOSÉ DOS CAMPOS, SP – 2019

**OBJETIVO**

Criação de um banco de dados para servir como base das informações de um sistema educacional online, o Instrue. A fim de controlar os níveis acessos, transações e disponibilização de cursos de uma aplicação web.

**REQUISITOS E REGRAS DE NEGÓCIO**

O sistema Instrue armazenar dados para a disponibilização de seus usuários, que podem ser instrutores ou alunos, usuários possuem nome, login, e-mail, status, senha, e uma foto. Cada instrutor pode estar vinculado a vários cursos, é necessário armazenar seu documento para identificação do mesmo, aos alunos deseja-se manter a matricula.

Cursos podem ser instruído por mais de um professor, deseja-se manter o título, status, thumbnail, preço, duração e sua descrição, os cursos são separados pelos seus módulos, os módulos possuem título, descrição, thumbnail, duração, cada modulo pertence exclusivamente a um curso pode possuir diversas aulas, sendo assim, aula possui título, um link para o vídeo, conteúdo e uma descrição. Os alunos podem ter acesso aos anexos disponíveis em cada aula, que são guardados título, comentário e link de acesso do anexo.

Para que os alunos participem dos cursos, é necessário que ele realize a compra do curso, que são registrado os dados do aluno, os itens que estão vinculados a compra, preço e a data onde deve ser possível utilizar como histórico de variação de preço ao longo do tempo, forma de pagamento que deverá ser validado entre CREDITO, DEBITO, BOLETO ou DEPÓSITO, durante a compra é possível relacionar um ou mais itens disponíveis. A compra possibilita o pagamento através de parcelas, onde possuem valor, data e status, a quantidade de parcelas geradas não deve ultrapassar a quantidade 12. Ainda é possível o aluno avaliar o curso para posicionar sua reputação onde é gerado um comparativo entre todos os cursos sobre sua avaliação média.

**MODELO CONCEITUAL**

**Uma imagem contendo texto, mapa

Descrição gerada automaticamente**

*FOTO: MODELO CONCEITUAL FEITO NO BR MODELO*

Histórico: AvaliaçãoAluno, PreçoItem

**MODELO LÓGICO**

**Uma imagem contendo texto

Descrição gerada automaticamente**

*FOTO: MODELO LÓGICO CRIADO EM STARUML*

**MODELO FÍSICO**

1. **TABELAS**

CREATE TABLE usuario(

usu\_codigo NUMBER(5),

usu\_login VARCHAR2(30) NOT NULL,

usu\_email VARCHAR2(50) NOT NULL,

usu\_senha VARCHAR2(32) NOT NULL,

usu\_nome VARCHAR2(50) NOT NULL,

usu\_foto VARCHAR2(255),

usu\_status CHAR(1) NOT NULL,

CONSTRAINT pk\_usu\_codigo PRIMARY KEY(usu\_codigo),

CONSTRAINT ck\_usu\_status CHECK(usu\_status in('A','I')),

CONSTRAINT uq\_usu\_login UNIQUE(usu\_login),

CONSTRAINT uq\_usu\_email UNIQUE(usu\_email)

);

CREATE TABLE instrutor(

usu\_codigo NUMBER(5),

ins\_documento VARCHAR2(30) NOT NULL,

CONSTRAINT pk\_ins\_usu\_codigo PRIMARY KEY(usu\_codigo),

CONSTRAINT fk\_ins\_usu\_codigo FOREIGN KEY(usu\_codigo) REFERENCES usuario(usu\_codigo),

CONSTRAINT uq\_ins\_documento UNIQUE(ins\_documento)

);

CREATE TABLE aluno(

usu\_codigo NUMBER(5),

alu\_matricula VARCHAR2(30) NOT NULL,

CONSTRAINT pk\_alu\_usu\_codigo PRIMARY KEY(usu\_codigo),

CONSTRAINT fk\_alu\_usu\_codigo FOREIGN KEY(usu\_codigo) REFERENCES usuario(usu\_codigo),

CONSTRAINT uq\_alu\_matricula UNIQUE(alu\_matricula)

);

CREATE TABLE categoria(

cat\_codigo NUMBER(5),

cat\_nome VARCHAR2(50) NOT NULL,

cat\_descricao VARCHAR2(255),

cat\_status CHAR(1) NOT NULL,

CONSTRAINT pk\_cat\_codigo PRIMARY KEY(cat\_codigo),

CONSTRAINT ck\_cat\_status CHECK(cat\_status in('A','I'))

);

CREATE TABLE curso(

cur\_codigo NUMBER(5),

cat\_codigo NUMBER(5),

cur\_titulo VARCHAR2(50) NOT NULL,

cur\_descricao VARCHAR2(255) NOT NULL,

cur\_duracao VARCHAR2(10) NOT NULL,

cur\_preco NUMBER(10,2) NOT NULL,

cur\_thumbnail VARCHAR2(255) NOT NULL,

cur\_avaliacao NUMBER(1),

cur\_status CHAR(1),

CONSTRAINT pk\_cur\_codigo PRIMARY KEY(cur\_codigo),

CONSTRAINT fk\_cur\_cat\_codigo FOREIGN KEY(cat\_codigo) REFERENCES categoria(cat\_codigo),

CONSTRAINT ck\_cur\_status CHECK(cur\_status in('A','I'))

);

CREATE TABLE instrutorcurso(

usu\_codigo NUMBER(5),

cur\_codigo NUMBER(5),

CONSTRAINT pk\_inc\_codigo PRIMARY KEY(usu\_codigo, cur\_codigo),

CONSTRAINT fk\_inc\_cur\_codigo FOREIGN KEY(cur\_codigo) REFERENCES curso(cur\_codigo),

CONSTRAINT fk\_inc\_uso\_codigo FOREIGN KEY(usu\_codigo) REFERENCES usuario(usu\_codigo)

);

CREATE TABLE modulo(

mod\_codigo NUMBER(5),

cur\_codigo NUMBER(5),

mod\_titulo VARCHAR2(50) NOT NULL,

mod\_descricao VARCHAR2(255) NOT NULL,

mod\_duracao VARCHAR2(10) NOT NULL,

mod\_thumbnail VARCHAR2(255) NOT NULL,

mod\_status CHAR(1) NOT NULL,

CONSTRAINT pk\_mod\_codigo PRIMARY KEY(mod\_codigo),

CONSTRAINT fk\_mod\_cur\_codigo FOREIGN KEY(cur\_codigo) REFERENCES curso(cur\_codigo),

CONSTRAINT ck\_mod\_status CHECK(mod\_status in('A','I'))

);

CREATE TABLE aula(

aul\_codigo NUMBER(5),

mod\_codigo NUMBER(5),

aul\_titulo VARCHAR2(50) NOT NULL,

aul\_descricao VARCHAR2(255) NOT NULL,

aul\_conteudo LONG NOT NULL,

aul\_video VARCHAR2(255),

aul\_status CHAR(1) NOT NULL,

CONSTRAINT pk\_aul\_codigo PRIMARY KEY(aul\_codigo),

CONSTRAINT fk\_aul\_mod\_codigo FOREIGN KEY(mod\_codigo) REFERENCES modulo(mod\_codigo),

CONSTRAINT ck\_aul\_status CHECK(aul\_status in('A','I'))

);

CREATE TABLE anexo(

ane\_codigo NUMBER(5),

aul\_codigo NUMBER(5),

ane\_titulo VARCHAR2(50) NOT NULL,

ane\_comentario VARCHAR2(255),

ane\_link VARCHAR2(255) NOT NULL,

ane\_status CHAR(1) NOT NULL,

CONSTRAINT pk\_ane\_codigo PRIMARY KEY(ane\_codigo),

CONSTRAINT fk\_ane\_aul\_codigo FOREIGN KEY(aul\_codigo) REFERENCES aula(aul\_codigo),

CONSTRAINT ck\_ane\_status CHECK(ane\_status in('A','I'))

);

CREATE TABLE compra(

com\_codigo NUMBER(5),

usu\_codigo NUMBER(5),

com\_data DATE NOT NULL,

com\_formapgto VARCHAR2(30),

com\_parcelas NUMBER(2),

com\_status CHAR(2) NOT NULL,

CONSTRAINT pk\_com\_codigo PRIMARY KEY(com\_codigo),

CONSTRAINT fk\_com\_usu\_codigo FOREIGN KEY(usu\_codigo) REFERENCES aluno(usu\_codigo),

CONSTRAINT ck\_com\_status CHECK(com\_status in('AG','PG')),

CONSTRAINT ck\_com\_formapgto CHECK(com\_formapgto in('CREDITO','DEBITO','BOLETO','TRANSFERENCIA')),

CONSTRAINT ck\_com\_nparcelas CHECK(com\_parcelas <= 12));

CREATE TABLE itemcompra(

cur\_codigo NUMBER(5),

com\_codigo NUMBER(5),

itc\_avaliacao NUMBER(1),

itc\_valor NUMBER(10,2),

CONSTRAINT pk\_itc\_codigo PRIMARY KEY(cur\_codigo, com\_codigo),

CONSTRAINT fk\_itc\_cur\_codigo FOREIGN KEY(cur\_codigo) REFERENCES curso(cur\_codigo),

CONSTRAINT fk\_itc\_com\_codigo FOREIGN KEY(com\_codigo) REFERENCES compra(com\_codigo)

);

CREATE TABLE parcelas(

par\_codigo NUMBER(5),

com\_codigo NUMBER(5),

par\_valor NUMBER(10,2) NOT NULL,

par\_data DATE NOT NULL,

par\_status CHAR(2) NOT NULL,

CONSTRAINT pk\_par\_codigo PRIMARY KEY(par\_codigo, com\_codigo),/\*Coloquei parcelas com chave composta\*/

CONSTRAINT fk\_par\_com\_codigo FOREIGN KEY(com\_codigo) REFERENCES compra(com\_codigo),

CONSTRAINT ck\_par\_status CHECK(par\_status in('AG','PG'))

);

1. **INSERÇÕES**
2. **CONSULTAS**

/\*CONSULTAS\*/

/\*

1) Listar, em ordem decrescente, o código, o nome, e a avaliação média dos 10 cursos mais bem avaliados da plataforma;

\*/

CREATE OR REPLACE VIEW vw\_avaliacaocurso AS

SELECT cur\_codigo, ROUND(AVG(itc\_avaliacao),1) AS cur\_avaliacao

FROM itemcompra

GROUP BY cur\_codigo

ORDER BY cur\_avaliacao DESC

WITH READ ONLY;

SELECT cur.cur\_codigo, cur.cur\_titulo, vw.cur\_avaliacao

FROM curso cur

INNER JOIN vw\_avaliacaocurso vw

ON cur.cur\_codigo = vw.cur\_codigo

WHERE ROWNUM <= 10

ORDER BY vw.cur\_avaliacao DESC;

/\*

2) Listar, em ordem crescente, o código, o nome e a avaliação média de todos os cursos cuja avaliação está abaixo da média geral;

\*/

CREATE OR REPLACE VIEW vw\_avaliacaocurso AS

SELECT cur\_codigo, ROUND(AVG(itc\_avaliacao),1) AS cur\_avaliacao

FROM itemcompra

GROUP BY cur\_codigo

ORDER BY cur\_avaliacao DESC

WITH READ ONLY;

SELECT cur.cur\_codigo, cur.cur\_titulo, vw.cur\_avaliacao

FROM curso cur

INNER JOIN vw\_avaliacaocurso vw

ON cur.cur\_codigo = vw.cur\_codigo

WHERE vw.cur\_avaliacao < (SELECT AVG(cur\_avaliacao) FROM vw\_avaliacaocurso)

ORDER BY vw.cur\_avaliacao ASC;

/\*

3) Listar, em ordem decrescente, o código, o nome e a quantidade de vendas dos 10 cursos mais vendidos;

\*/

CREATE OR REPLACE VIEW vw\_vendacurso AS

SELECT cur\_codigo, COUNT(cur\_codigo) AS cur\_qtdvendas

FROM itemcompra

GROUP BY cur\_codigo

ORDER BY cur\_qtdvendas DESC

WITH READ ONLY;

SELECT cur.cur\_codigo, cur.cur\_titulo, vw.cur\_qtdvendas

FROM curso cur

INNER JOIN vw\_vendacurso vw

ON cur.cur\_codigo = vw.cur\_codigo

WHERE ROWNUM <= 10

ORDER BY vw.cur\_qtdvendas DESC;

/\*

4) Listar o valor obtido com a venda de cursos, por trimestre, no ano de 2019;

\*/

SELECT CASE WHEN EXTRACT(MONTH FROM com.com\_data) in (1,2,3) THEN 'Primeiro'

WHEN EXTRACT(MONTH FROM com.com\_data) in (4,5,6) THEN 'Segundo'

WHEN EXTRACT(MONTH FROM com.com\_data) in (7,8,9) THEN 'Terceiro'

WHEN EXTRACT(MONTH FROM com.com\_data) in (10,11,12) THEN 'Quarto'

END AS com\_trimestre,

EXTRACT(YEAR FROM com.com\_data) as com\_ano,

SUM(CASE WHEN EXTRACT(MONTH FROM com.com\_data) in (1,2,3) THEN itc.itc\_valor

WHEN EXTRACT(MONTH FROM com.com\_data) in (4,5,6) THEN itc.itc\_valor

WHEN EXTRACT(MONTH FROM com.com\_data) in (7,8,9) THEN itc.itc\_valor

WHEN EXTRACT(MONTH FROM com.com\_data) in (10,11,12) THEN itc.itc\_valor

END) as com\_total

FROM compra com

INNER JOIN itemcompra itc

ON com.com\_codigo = itc.com\_codigo

WHERE EXTRACT(YEAR FROM com.com\_data) = 2019

GROUP BY CASE WHEN EXTRACT(MONTH FROM com.com\_data) in (1,2,3) THEN 'Primeiro'

WHEN EXTRACT(MONTH FROM com.com\_data) in (4,5,6) THEN 'Segundo'

WHEN EXTRACT(MONTH FROM com.com\_data) in (7,8,9) THEN 'Terceiro'

WHEN EXTRACT(MONTH FROM com.com\_data) in (10,11,12) THEN 'Quarto' END,

EXTRACT(YEAR FROM com.com\_data);

/\*

4) Listar o valor obtido com a venda de cursos, por semestre, nos ultimos 3 anos, cuja forma de pagamento utilizada foi cartão de crédito e a quantidade de parcelas foi maior que 1;

\*/

SELECT CASE WHEN EXTRACT(MONTH FROM com.com\_data) in (1,2,3,4,5,6) THEN 'Primeiro'

WHEN EXTRACT(MONTH FROM com.com\_data) in (7,8,9,10,11,12) THEN 'Segundo'

END AS com\_semestre,

EXTRACT(YEAR FROM com.com\_data) as com\_ano,

SUM(CASE WHEN EXTRACT(MONTH FROM com.com\_data) in (1,2,3,4,5,6) THEN itc.itc\_valor

WHEN EXTRACT(MONTH FROM com.com\_data) in (7,8,9,10,11,12) THEN itc.itc\_valor

END) as com\_total

FROM compra com

INNER JOIN itemcompra itc

ON com.com\_codigo = itc.com\_codigo

WHERE EXTRACT(YEAR FROM com.com\_data) in (2019,2018,2017) AND com.com\_formapgto = 'CREDITO' AND com.com\_parcelas > 1

GROUP BY CASE WHEN EXTRACT(MONTH FROM com.com\_data) in (1,2,3,4,5,6) THEN 'Primeiro'

WHEN EXTRACT(MONTH FROM com.com\_data) in (7,8,9,10,11,12) THEN 'Segundo' END,

EXTRACT(YEAR FROM com.com\_data);

/\*

6) Listar, em ordem decrescente, as formas de pagamento, a quantidade de compras que utilizaram elas para pagamento e a receita que cada uma gerou;

\*/

SELECT com.com\_formapgto, COUNT(com.com\_formapgto), SUM(itc.itc\_valor)

FROM itemcompra itc

INNER JOIN compra com

ON com.com\_codigo = itc.com\_codigo

GROUP BY com.com\_formapgto, com.com\_formapgto, itc.itc\_valor

ORDER BY itc.itc\_valor DESC;

/\*

7) Listar, em ordem decrescente, o código, o nome, a quantidade de cursos comprados e o valor gasto com a compra desses cursos, dos alunos que mais

compraram cursos no ano de 2019;

\*/

CREATE OR REPLACE VIEW vw\_alunoqtdcurso AS

SELECT usu\_codigo, COUNT(com\_codigo) AS com\_qtdcursos

FROM compra

GROUP BY usu\_codigo

ORDER BY COUNT(com\_codigo) DESC

WITH READ ONLY;

SELECT vw.usu\_codigo, usu.usu\_nome, vw.com\_qtdcursos, SUM(itc.itc\_valor)

FROM vw\_alunoqtdcurso vw

INNER JOIN usuario usu

ON vw.usu\_codigo = usu.usu\_codigo

INNER JOIN compra com

ON usu.usu\_codigo = com.usu\_codigo

INNER JOIN itemcompra itc

ON com.com\_codigo = itc.com\_codigo

WHERE EXTRACT(YEAR FROM com.com\_data) = 2019

GROUP BY vw.usu\_codigo, usu.usu\_nome, vw.com\_qtdcursos

ORDER BY SUM(itc.itc\_valor) DESC;

/\*

8) Listar, em ordem decrescente, o código, o nome e a quantidade de cursos vendidos pelos instrutores;

\*/

SELECT ins.usu\_codigo, usu.usu\_nome, vw.cur\_qtdvendas

FROM vw\_vendacurso vw

INNER JOIN curso cur

ON vw.cur\_codigo = cur.cur\_codigo

INNER JOIN instrutorcurso inc

ON cur.cur\_codigo = inc.cur\_codigo

INNER JOIN instrutor ins

ON inc.usu\_codigo = ins.usu\_codigo

INNER JOIN usuario usu

ON ins.usu\_codigo = usu.usu\_codigo

GROUP BY ins.usu\_codigo, usu.usu\_nome, vw.cur\_qtdvendas

ORDER BY cur\_qtdvendas DESC;

/\*

9) Listar, em ordem decrescente, o código, o nome e a receita total obtida, por instrutor, com a venda de cursos;

\*/

CREATE OR REPLACE VIEW vw\_vendacurso AS

SELECT cur\_codigo, COUNT(cur\_codigo) AS cur\_qtdvendas, SUM(itc\_valor) AS cur\_receita

FROM itemcompra

GROUP BY cur\_codigo

ORDER BY cur\_receita DESC

WITH READ ONLY;

SELECT ins.usu\_codigo, usu.usu\_nome, vw.cur\_receita

FROM vw\_vendacurso vw

INNER JOIN curso cur

ON vw.cur\_codigo = cur.cur\_codigo

INNER JOIN instrutorcurso inc

ON cur.cur\_codigo = inc.cur\_codigo

INNER JOIN instrutor ins

ON inc.usu\_codigo = ins.usu\_codigo

INNER JOIN usuario usu

ON ins.usu\_codigo = usu.usu\_codigo

GROUP BY ins.usu\_codigo, usu.usu\_nome, vw.cur\_receita

ORDER BY cur\_receita DESC;

/\*

10) Listar, em ordem crescente, o código, o nome e a média de vendas de todos os instrutores que venderam menos do que a média geral;

\*/

SELECT ins.usu\_codigo, usu.usu\_nome, AVG(vw.cur\_qtdvendas)

FROM vw\_vendacurso vw

INNER JOIN curso cur

ON vw.cur\_codigo = cur.cur\_codigo

INNER JOIN instrutorcurso inc

ON cur.cur\_codigo = inc.cur\_codigo

INNER JOIN instrutor ins

ON inc.usu\_codigo = ins.usu\_codigo

INNER JOIN usuario usu

ON ins.usu\_codigo = usu.usu\_codigo

GROUP BY ins.usu\_codigo, usu.usu\_nome

HAVING AVG(vw.cur\_qtdvendas) < (SELECT AVG(cur\_qtdvendas) FROM vw\_vendacurso)

ORDER BY AVG(vw.cur\_qtdvendas) ASC;

/\*

11) Listar, em ordem crescente, o código, o nome e a média de avaliação de todos os instrutores cuja média de avaliação seja menor que a média geral;

\*/

SELECT ins.usu\_codigo, usu.usu\_nome, AVG(vw.cur\_avaliacao)

FROM vw\_avaliacaocurso vw

INNER JOIN curso cur

ON vw.cur\_codigo = cur.cur\_codigo

INNER JOIN instrutorcurso inc

ON cur.cur\_codigo = inc.cur\_codigo

INNER JOIN instrutor ins

ON inc.usu\_codigo = ins.usu\_codigo

INNER JOIN usuario usu

ON ins.usu\_codigo = usu.usu\_codigo

GROUP BY ins.usu\_codigo, usu.usu\_nome

HAVING AVG(vw.cur\_avaliacao) < (SELECT AVG(cur\_avaliacao) FROM vw\_avaliacaocurso)

ORDER BY AVG(vw.cur\_avaliacao) ASC;

/\*

12) Listar, em ordem decrescente, o código e o nome de todos os instrutores, e a quantidade de categorias de cursos que eles mais ministraram aulas

\*/

SELECT inc.usu\_codigo, usu.usu\_nome, COUNT(cat.cat\_codigo)

FROM usuario usu

INNER JOIN instrutorcurso inc

ON usu.usu\_codigo = inc.usu\_codigo

INNER JOIN curso cur

ON inc.cur\_codigo = cur.cur\_codigo

INNER JOIN categoria cat

ON cur.cat\_codigo = cat.cat\_codigo

GROUP BY inc.usu\_codigo, usu.usu\_nome

ORDER BY COUNT(cat.cat\_codigo) DESC;

/\*

13) Listar, em ordem decrescente, o código, o nome, e a quantidade de aulas dos cursos;

\*/

SELECT cur.cur\_codigo, cur.cur\_titulo, COUNT(aul.aul\_codigo) AS cur\_qtdaulas

FROM curso cur

INNER JOIN modulo mod

ON cur.cur\_codigo = mod.cur\_codigo

INNER JOIN aula aul

ON mod.mod\_codigo = aul.mod\_codigo

GROUP BY cur.cur\_codigo, cur.cur\_titulo

ORDER BY cur\_qtdaulas DESC;

/\*

14) Listar, em ordem decrescente, o código e o nome da categoria e a quantidade de cursos que ela possui;

\*/

SELECT cat.cat\_codigo, cat.cat\_nome, COUNT(cur.cat\_codigo)

FROM categoria cat

INNER JOIN curso cur

ON cat.cat\_codigo = cur.cat\_codigo

GROUP BY cat.cat\_codigo, cat.cat\_nome

ORDER BY COUNT(cur.cat\_codigo) DESC;

/\*

15) Listar, em ordem decrescente, o código, o nome e a quantidade de anexos disponibilizados pelos instrutores;

\*/

SELECT ins.usu\_codigo, usu.usu\_nome, cur.cur\_codigo, COUNT(ane.aul\_codigo)

FROM usuario usu

INNER JOIN instrutor ins

ON ins.usu\_codigo = usu.usu\_codigo

INNER JOIN instrutorcurso inc

ON ins.usu\_codigo = inc.usu\_codigo

INNER JOIN curso cur

ON inc.cur\_codigo = cur.cur\_codigo

INNER JOIN modulo mod

ON cur.cur\_codigo = mod.cur\_codigo

INNER JOIN aula aul

ON mod.mod\_codigo = aul.mod\_codigo

INNER JOIN anexo ane

ON aul.aul\_codigo = ane.aul\_codigo

GROUP BY ins.usu\_codigo, usu.usu\_nome, cur.cur\_codigo

ORDER BY COUNT(ane.aul\_codigo);