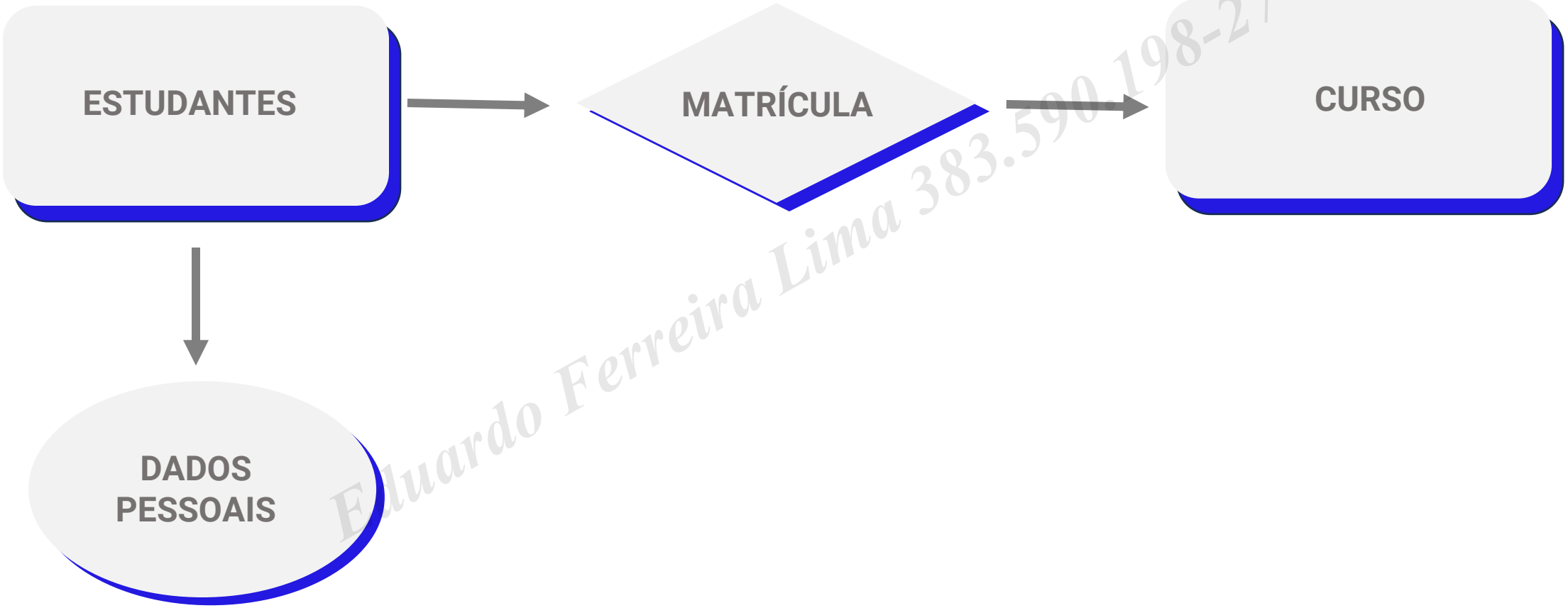
MBA USP ESALQ

ENGENHARIA DE DADOS II

Prof. Dr. Jeronymo Marcondes

*A responsabilidade pela idoneidade, originalidade e licitude dos conteúdos didáticos apresentados é do professor.

Introdução



Perguntas

- Como unir as informações das tabelas?
- Como obter informações de curso a partir de estudantes?
- Como realizar consultas bem performáticas com qualificações inseridas?
- Como garantir que o modelo seja robusto a transações.

Transações

- Execução no banco de dados.
- Garante integridade – exemplo de compra de lugares no teatro.
- Sensação de execução local com isolamento e proteção contra perdas.
- Conceito de lock.

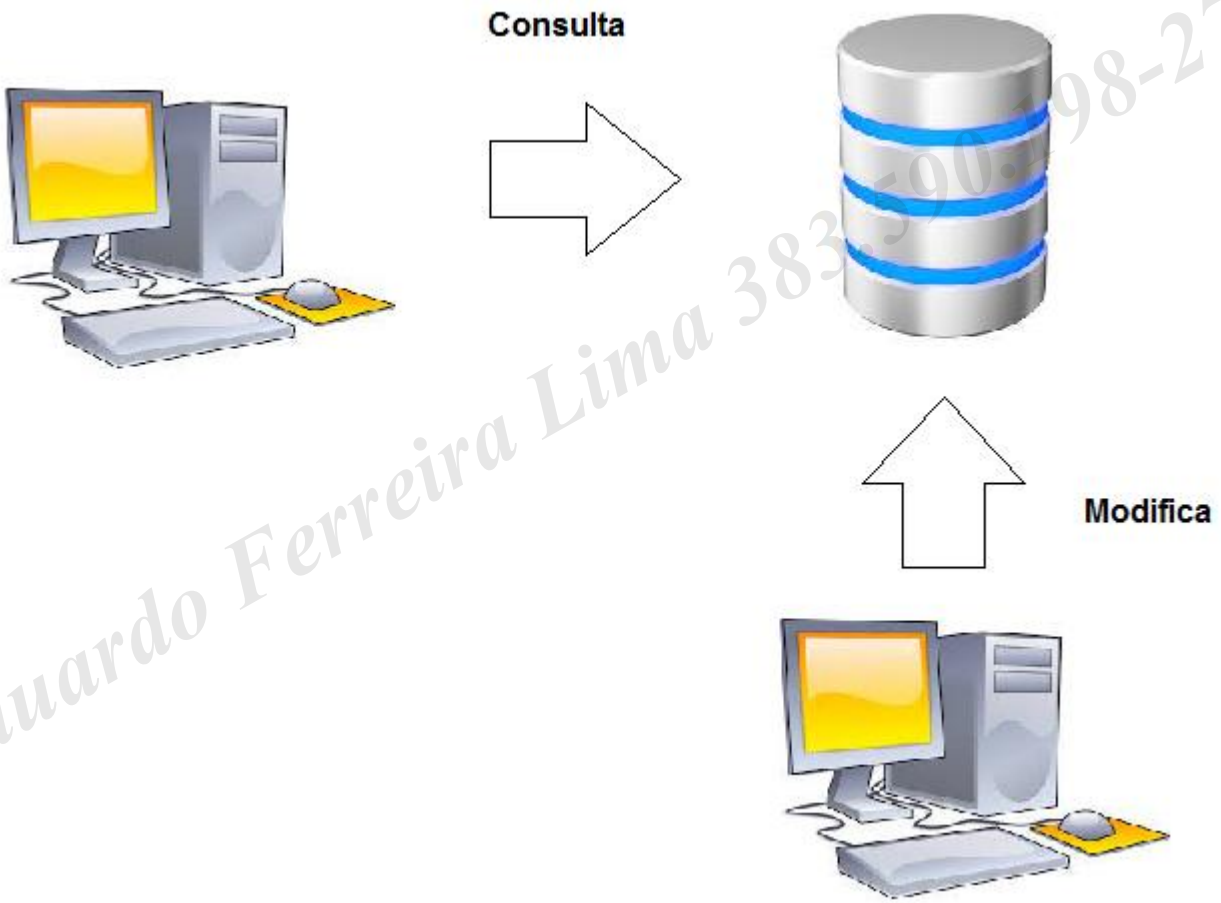
Locking Protocol

- Regras que garantem que, mesmo que várias pessoas executem querys ao mesmo tempo, o resultado líquido será o mesmo que teria ocorrido se as mesmas tivessem sido executadas em fila.

- O lock irá garantir que o objeto consultado não possa ser acessado por meio de outras transações.

- Exclusive Lock e Shared Lock.

Exemplo



Relacionamento

- Como garantir robustez ao modelo do negócio? Além da robustez de transações -> restrições.
- O relacionamento entre entidades. Como se relacionam estudantes e cursos?
- A tabela tem de ter consistência interna e os relacionamentos dela com as demais também.

Restrições de Integridade

- Restrições de chave, de relacionamento e gerais.
- Restrições de chave: um subconjunto mínimo de campos de uma relação que identifica tupla única.
- Ou seja, campo(s) definidos como chave devem garantir que a linha selecionada seja única.

Exemplo

CPF	Nome	Curso
xxx	João	Ciência de Dados
yyy	João	Medicina
hhh	Pedro	Medicina

Nome	Sobrenome	Curso
João	Silva	Ciência de Dados
João	Marinho	Ciência de Dados
Pedro	Guedes	Ciência de Dados

Chave Primária

- Uma determinada tabela pode ter várias chaves = chaves candidatas
- Chave primária é definida pelo DBA de forma que o SGBD faça as averiguações por meio da mesma.
- Chave primária bem definida é importante pois suscita a criação e índices o que torna as consultas mais performáticas.

Formas Normais

- Série de regras que garantem se um BD foi bem projetado.
- Mostra a importância de uma chave primária bem definida.
- Objetivo:

1) Garantir informação sem redundância.

2) Garantir eficiência na obtenção dos dados.

Formas normais

- 1ª forma normal:

Cada linha é uma informação. Não podem existir grupos repetidos ou atributos com mais de um valor.

PESSOAS = {ID+ NOME + ENDERECO + TELEFONES}

PESSOAS = { ID + NOME + ENDERECO }

TELEFONES = { PESSOA_ID + TELEFONE }

Formas normais

- 1ª forma normal:

ID	NOME	ENDEREÇO	TELEFONES
XX	JOAO	AV JOAO	99999;88888;77777
YY	PEDRO	AV PEDRO	77776;5555

ID	NOME	ENDEREÇO
XX	JOAO	AV JOAO
YY	PEDRO	AV PEDRO

ID	TELEFONE
XX	99999
XX	88888
XX	77777
YY	77776
YY	5555

Formas normais

- 2ª forma normal:

Todas as colunas que não participam da chave primária são dependentes de todas as colunas que compõem a chave primária.

ALUNOS_CURSOS = { ID_ALUNO + ID_CURSO + NOTA + DESCRICAO_CURSO }

ALUNOS_CURSOS = {ID_ALUNO + ID_CURSO + NOTA}

CURSOS = {ID_CURSO + DESCRICAO}

Restrições Gerais

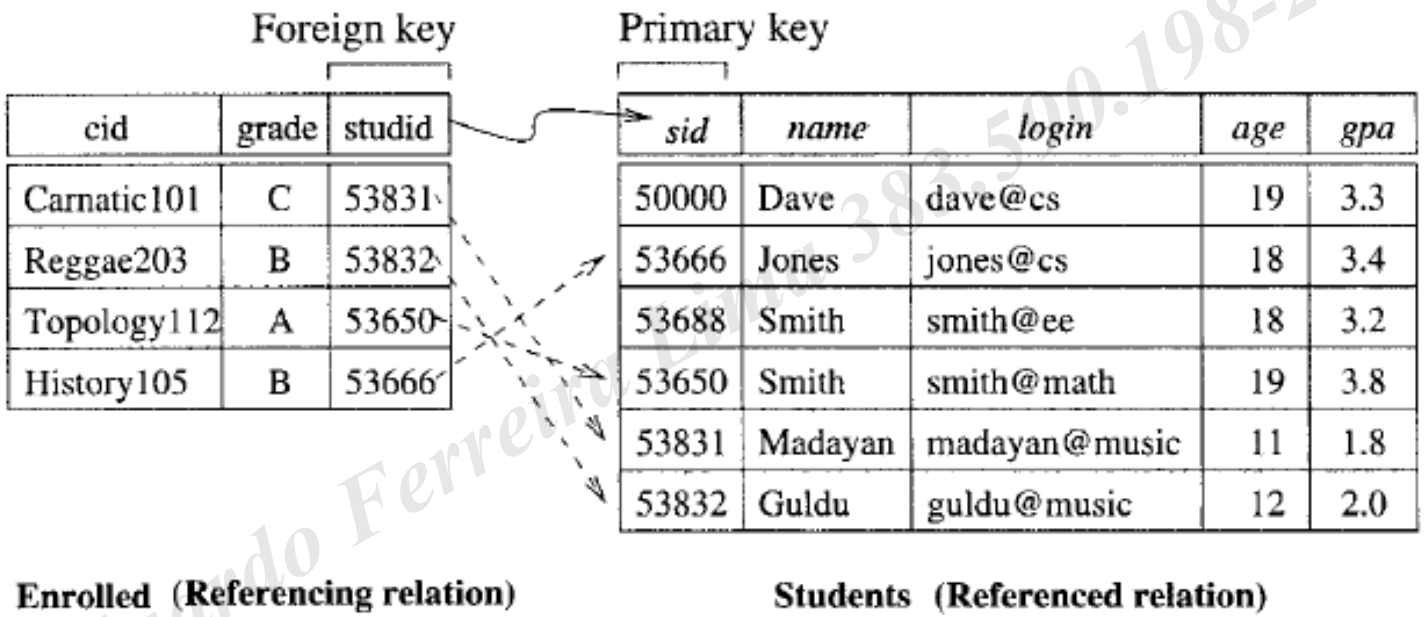
- Restrições de , principalmente, de negócio.
- Exemplo: inserção de idade.
- Os modernos SGBD já tem ferramentas que permitem criar tais restrições.

Como lidar com modelos com mais de uma tabela?

Chave Estrangeiras

- Chave primária de outra tabela
- Essa chave nos permite ligar tabelas diferentes de forma a garantir a unicidade da relação.
- O nome da chave estrangeira não precisa ser o mesmo da chave primária = o que importa é o conteúdo!

19



Casos Específicos

- Inserir Tupla <*55555, Art104, A*> nos cursos com inscrição.
- Deletar tupla <*53666, Jones, Jones@cs, 18, 3.4*> de estudantes.
- Inserir tupla <*55669, Margareth, MG@test, 21, 4*> em estudantes.

Ideia de junção

Tabela 1

CPF	NOME
xxx	ze das couves
yyy	maria das desgraças

Tabela 2

CPF	IDADE	PIS
xxx	21	hhh
yyy	25	JJJ

Tabela Derivada

CPF	NOME	IDADE
xxx	ze das couves	21
yyy	maria das desgraças	25

ACID

- *Atomicity*, *Consistency*, *Isolation*, *Durability*

- Conjunto de propriedades em transações de bancos de dados que são importantes para garantir a validade dos dados mesmo que ocorram erros durante o armazenamento ou problemas mais graves no sistema, como crashes ou problemas físicos em um servidor. As propriedades ACID são fundamentais para o processamento de transações em bancos de dados.

ACID

- Atomicidade: Garante que cada transação seja tratada como uma entidade única, a qual deve ser executada por completo ou falhar completamente.
- Consistência: Os dados que são gravados devem sempre ser válidos.
- Isolamento: Permite deixar o banco de dados no mesmo estado em que ele estaria caso as transações fossem executadas em sequência.
- Durabilidade: A propriedade da durabilidade garante que uma transação, uma vez executada (efetivada), permanecerá neste estado mesmo que haja um problema grave no sistema

Cardinalidade

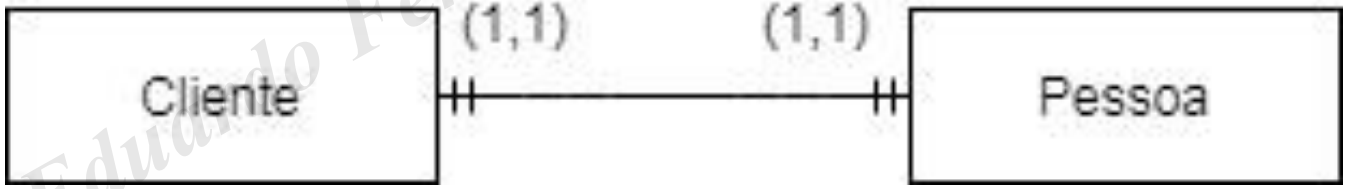
- Cardinalidade: indica quantas ocorrências de uma Entidade participam no mínimo e no máximo do relacionamento.

Tipos de relacionamento:

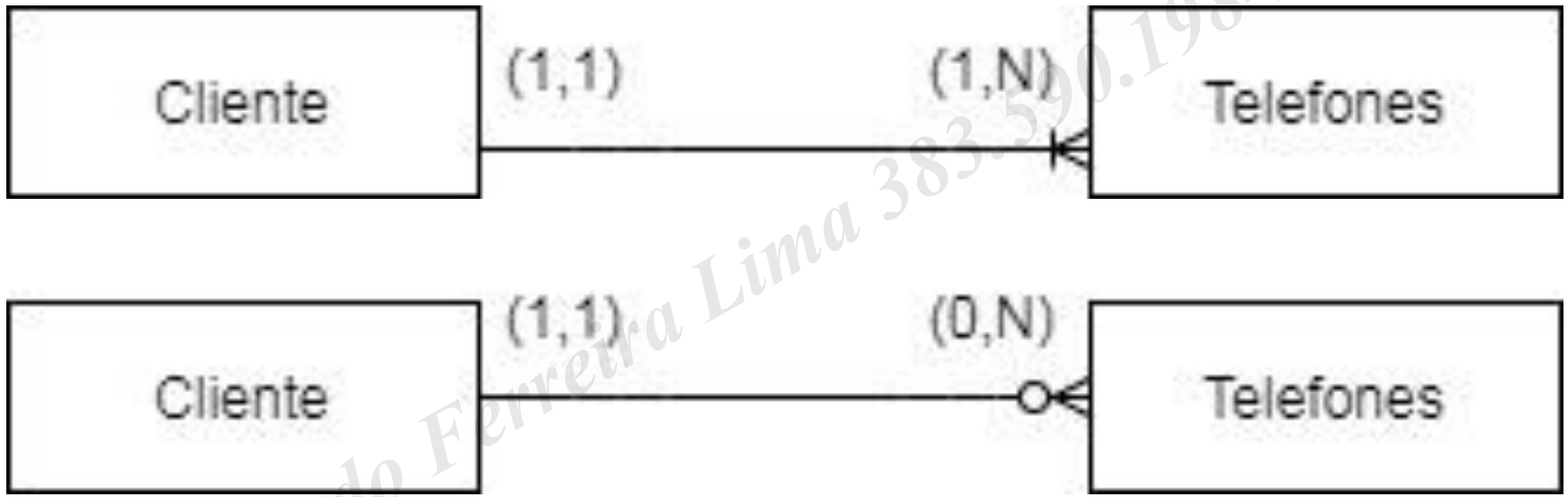
1) Um para um;
2) Muitos para um;
3) Muitos para muitos.

Um para um

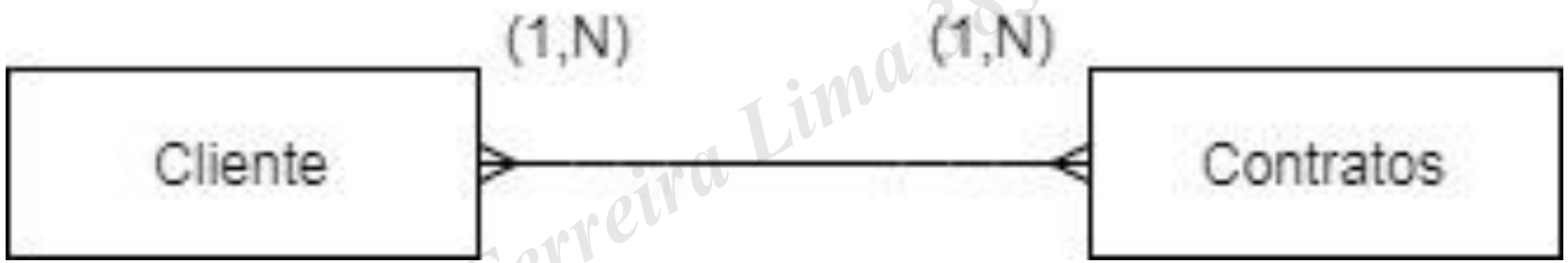
- Cardinalidade mínima: define se a relação é obrigatória.
- Cardinalidade máxima: define a quantidade máxima de ocorrências da Entidade que pode participar do Relacionamento.



Muitos para um



Muitos para Muitos



Operando com SQL

JOIN

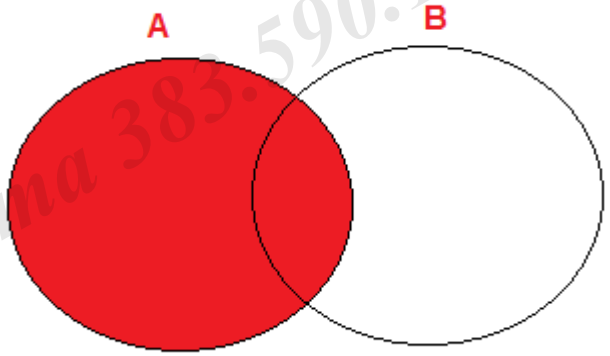
- Especifica como será feita a junção entre duas tabelas. Por exemplo:

Id_cliente	Pedido

Id_cliente	Nome	Endereço

Id_cliente	Nome	Endereço	Pedido

LEFT JOIN



SELECT [*DISTINCT*] lista-seleção

FROM lista-origem-1

LEFT JOIN lista-origem-2

ON lista-origem-1. campo_em_comum = lista-origem-2. campo_em_comum

WHERE qualificação

LEFT JOIN

```
SELECT  *
FROM pedidos
LEFT JOIN endereco
ON pedidos.Id_cliente = endereço.Id_cliente
```

32

Id_cliente	Pedido
xxx	1
yyy	2

Id_cliente	Nome	Endereço
xxx	joao	av joao
hhh	pedro	av pedro

Id_cliente	Nome	Endereço	Pedido
xxx	joao	av joao	1
yyy	NULL	NULL	2

NULL

- Até agora somente valores conhecidos.
- Se desconhecido = NULL.
- Quando o valor é desconhecido ou não se aplica.

Exemplo com NULL

```
SELECT  *
FROM pedidos_e_endereço
WHERE nome IS NOT NULL
```

Id_cliente	Nome	Endereço	Pedido
xxx	joao	av joao	1

```
SELECT  *
FROM pedidos_e_endereço
WHERE nome IS NULL
```

Id_cliente	Nome	Endereço	Pedido
yyy	NULL	NULL	2

RIGHT JOIN

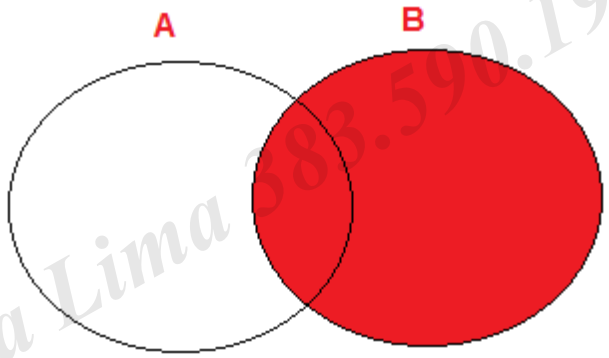
SELECT [*DISTINCT*] lista-seleção

FROM lista-origem-1

RIGHT JOIN lista-origem-2

ON lista-origem-1. campo_em_comum = lista-origem-2. campo_em_comum

WHERE qualificação



RIGHT JOIN

```
SELECT  *
FROM pedidos
RIGHT JOIN endereco
ON pedidos.Id_cliente = endereço.Id_cliente
```

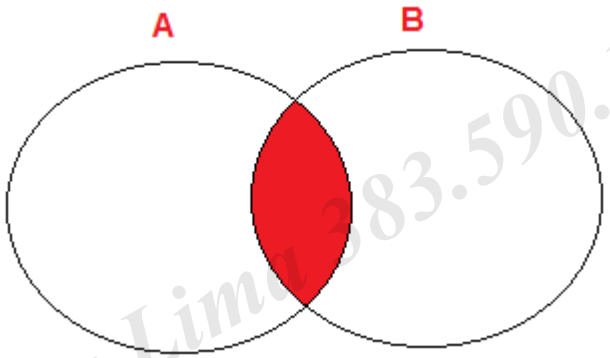
37

Id_cliente	Pedido
xxx	1
yyy	2

Id_cliente	Nome	Endereço
xxx	joao	av joao
hhh	pedro	av pedro

Id_cliente	Nome	Endereço	Pedido
xxx	joao	av joao	1
hhh	pedro	av pedro	NULL

INNER JOIN



SELECT [*DISTINCT*] lista-seleção

FROM lista-origem-1

INNER JOIN lista-origem-2

ON lista-origem-1. campo_em_comum = lista-origem-2. campo_em_comum

WHERE qualificação

INNER JOIN

```
SELECT  *
FROM pedidos
INNER JOIN endereco
ON pedidos.Id_cliente = endereço.Id_cliente
```

40

Id_cliente	Pedido
xxx	1
yyy	2

Id_cliente	Nome	Endereço
xxx	joao	av joao
hhh	pedro	av pedro

Id_cliente	Nome	Endereço	Pedido
xxx	joao	av joao	1

UNION ALL

- Tabelas de mesma estrutura que serão “empilhadas”.

```
SELECT lista-seleção
FROM lista-origem-1
UNION ALL
SELECT lista-seleção
FROM lista-origem-2
```

UNION ALL

```
SELECT  Id_cliente
FROM pedidos
UNION ALL
SELECT  Id_cliente
FROM endereco
```

43

Id_cliente	Pedido
xxx	1
yyy	2

Id_cliente	Nome	Endereço
xxx	joao	av joao
hhh	pedro	av pedro

Id_cliente
xxx
yyy
xxx
hhh

UNION

- Tabelas de mesma estrutura que serão “empilhadas”.
- Diferencia-se por aplicar um *DISTINCT*.

```
SELECT lista-seleção
FROM lista-origem-1
UNION
SELECT lista-seleção
FROM lista-origem-2
```

UNION

```
SELECT  Id_cliente
FROM pedidos
UNION
SELECT  Id_cliente
FROM endereco
```

46

Id_cliente	Pedido
xxx	1
yyy	2

Id_cliente	Nome	Endereço
xxx	joao	av joao
hhh	pedro	av pedro

Id_cliente
xxx
yyy
hhh

Nested Queries

- Resultado de query anterior pode ser utilizada na atual
- Forma mais comum:

```
SELECT lista-seleção-derivada
FROM lista-origem
WHERE coluna IN
        (
                SELECT lista-seleção-original
                 FROM lista-origem
        )
```

Nested Queries

CPF	NOME	ENDERECO
XXX	JOAO	AV JOAO
YYY	MARIA	AV MARIA

CPF	PEDIDO	VALOR (R$)
XXX	10	500
YYY	12	1000

Nested Queries

```
SELECT  CPF, Nome, Endereco
FROM Consumidores
WHERE CPF IN
        (
                SELECT  CPF
                FROM Gastos
                WHERE Valor > 500
        )
```

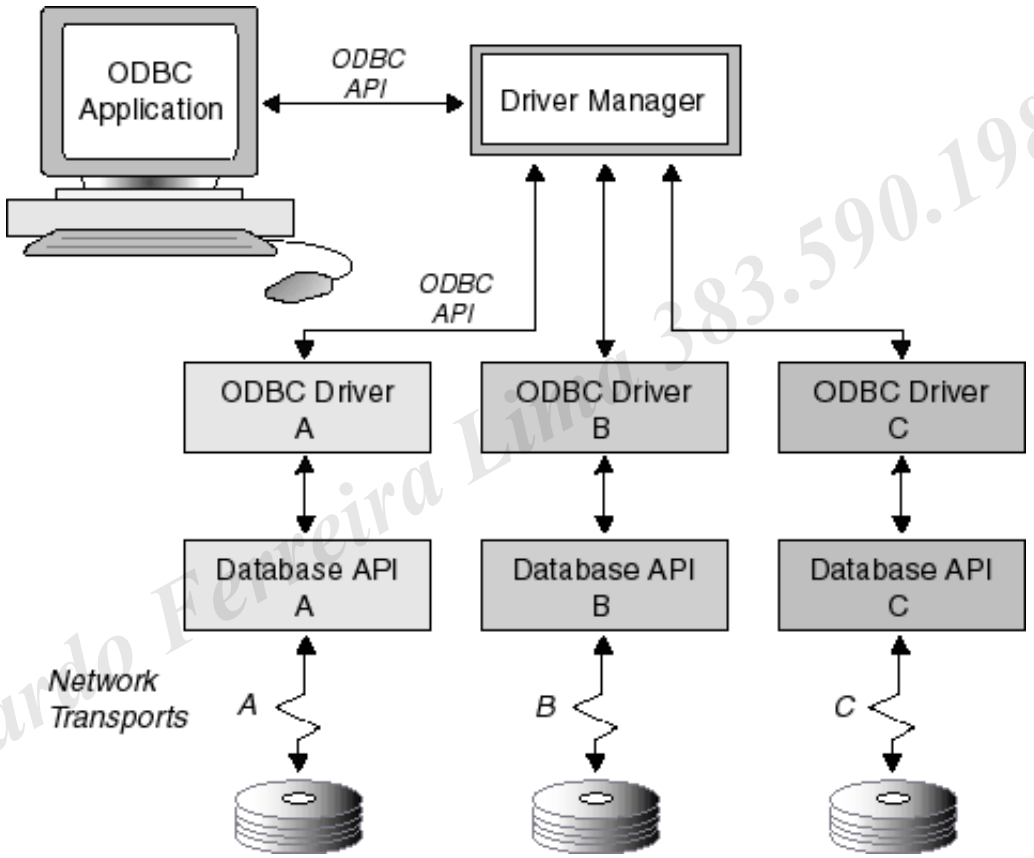
ODBC e JDBC

- Java Database Connectivity – SUM
- Open Database Connectivity – Microsoft
- API – application programming interface

ODBC e JDBC

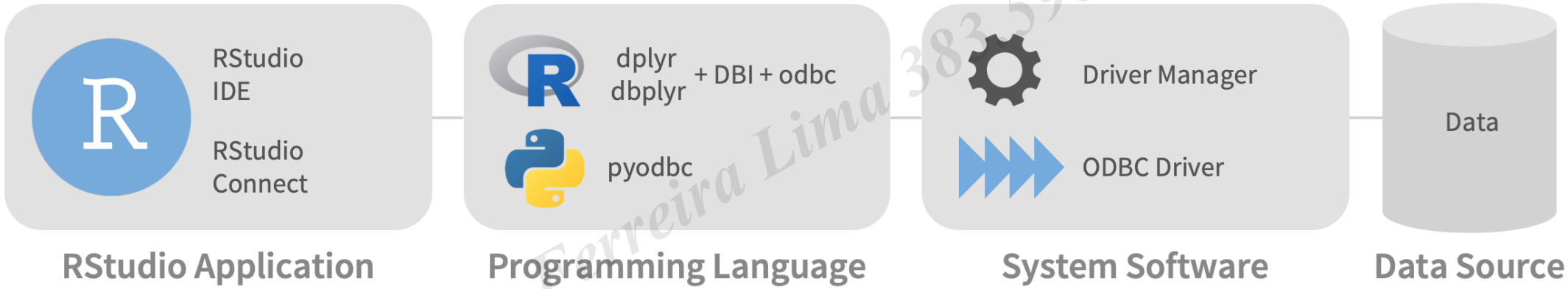
- Permite a execução de SQL dentro do banco a partir de aplicações.
- Pode acessar diversos servidores de dados ao mesmo tempo.
- Todas as transações ocorrem por meio de um driver.

ODBC e JDBC



https://docs.oracle.com/

Exemplo



ODBC e JDBC

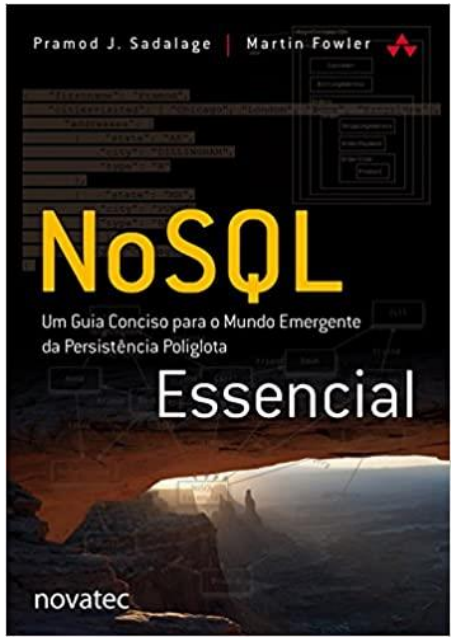
- Ordem:

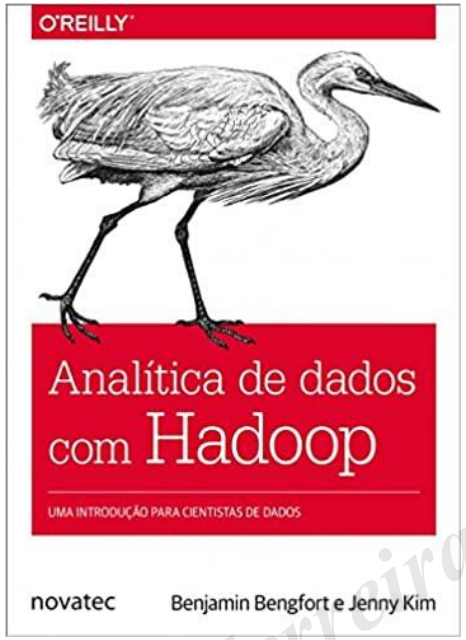
1. Seleciona origem de dados.

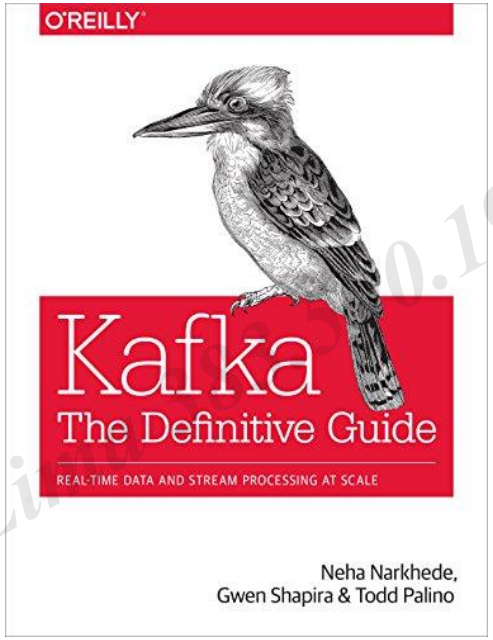
2. Carrega o respectivo driver.

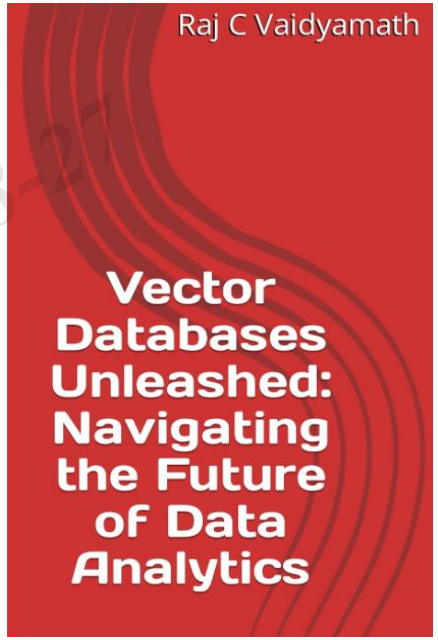
3. Estabelece a conexão com a origem.

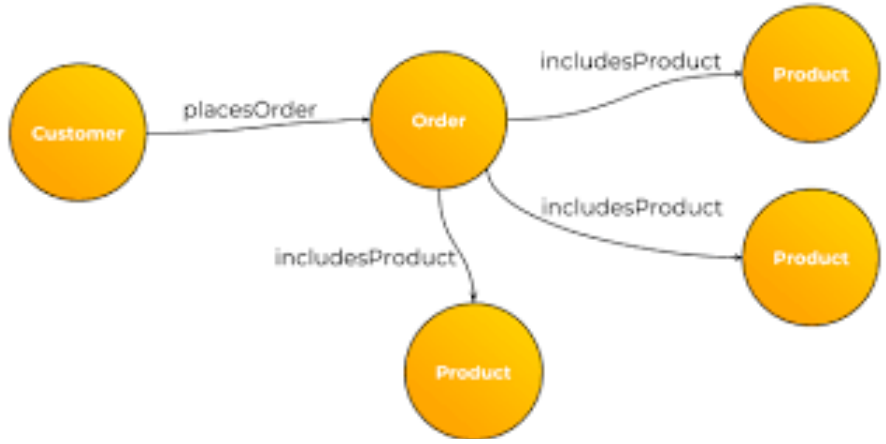
Discussão – futuro dos bancos de dados











OBRIGADO!

Prof. Dr. Jeronymo Marcondes

https://www.linkedin.com/in/jeronymo-marcondes-585a26186