MBA
USP
ESALQ

DATA ENGINEERING II

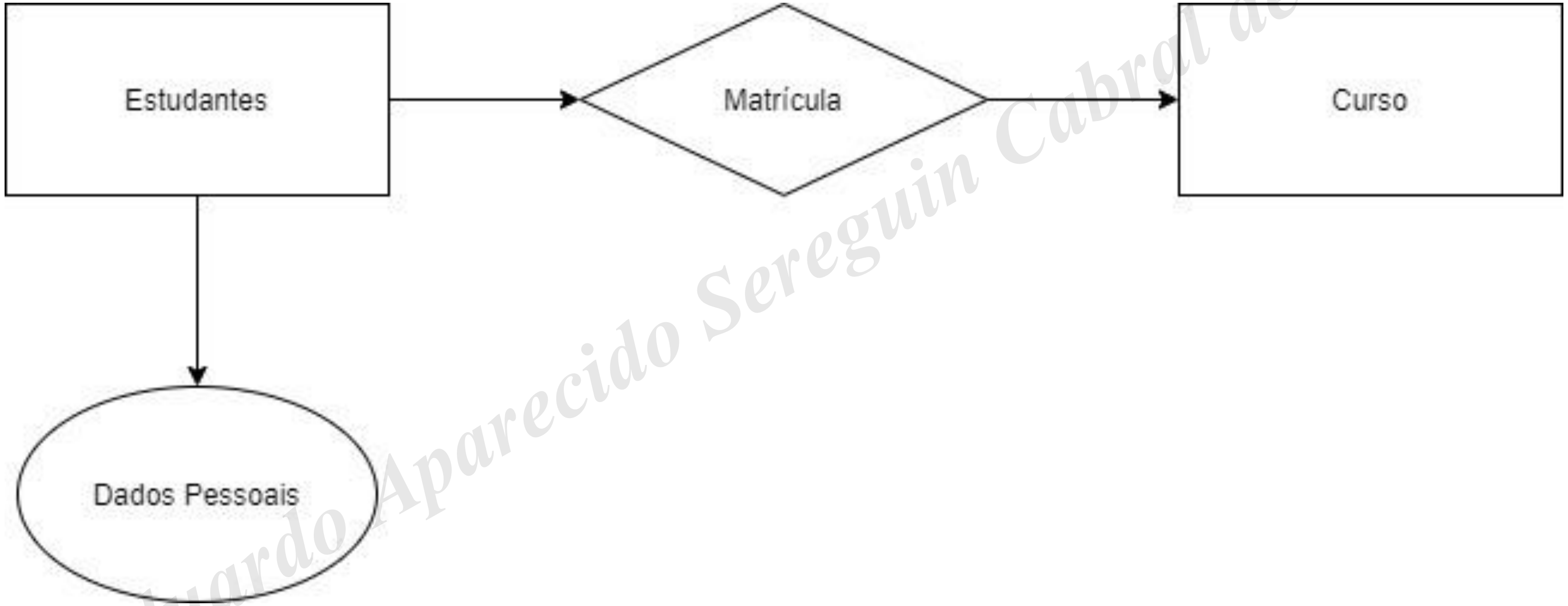
Prof. Dr. Jeronymo Marcondes

*The responsibility for trustworthiness, originality, and legality of the didactic content presented is responsibility of the professor.

The total or partial reproduction of this material without authorization **is prohibited.**

Law No. 9610/98

Introduction



QUESTIONS

- How to join the information of tables?

- How to obtain course information from students?

- How to perform consultations well performative with inserted qualifications?

- How to ensure that the model is robust for transactions?

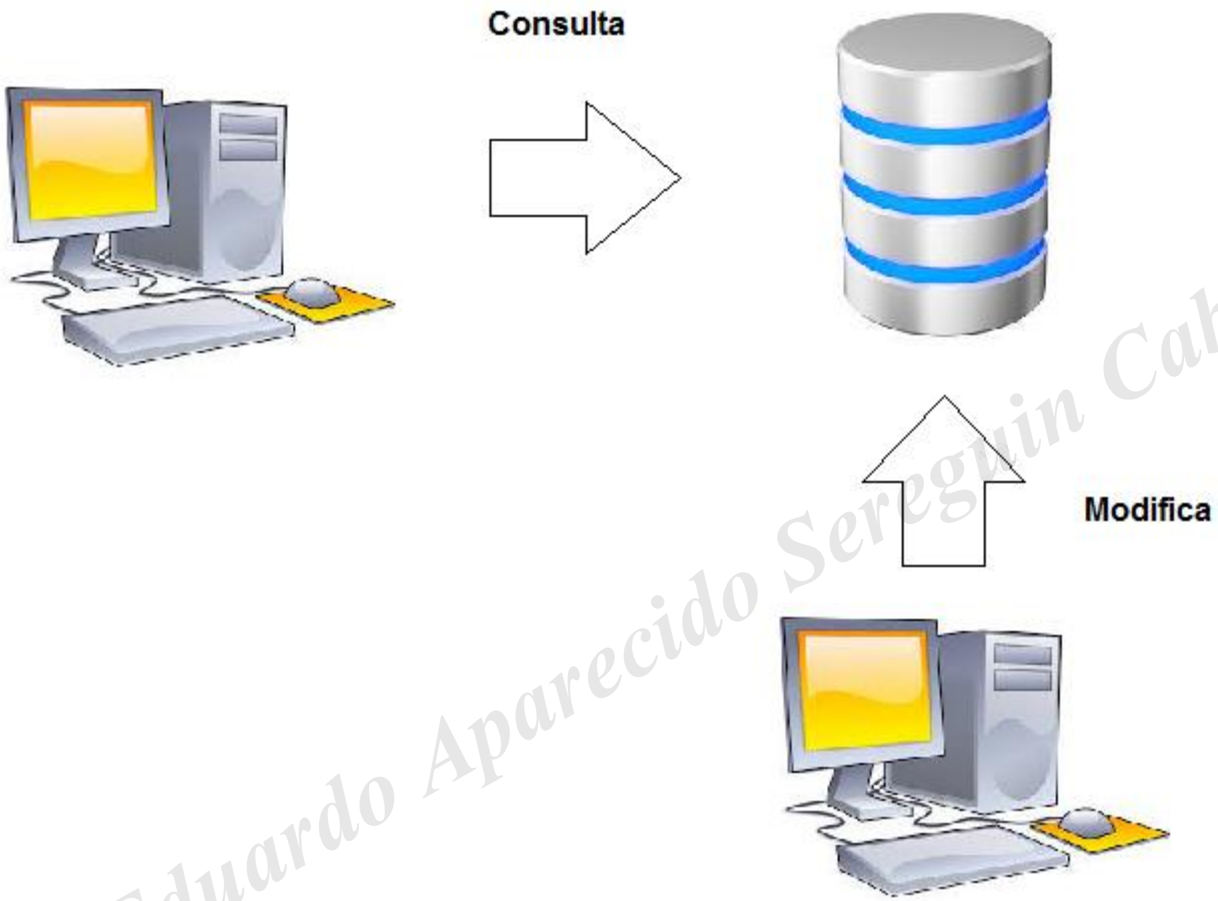
Transactions

- Execution in the database.
- Ensure integrity – example of purchase of seats in the theater.
- Sensation of local execution with isolation and protection against loss.
- Lock concept.

Locking Protocol

- Rules that ensure that even if several people execute queries at the same time, the net result would be the same if they had executed in line.
- Lock will ensure that the consulted object cannot be accessed through other transactions.
- Exclusive Lock and Shared Lock.

Example



Relationship

- How to ensure robustness to the business model? In addition to the robustness of transactions -> restrictions.
- The relationship between entities. How are the students and courses related?
- The table needs to have internal consistency and its relationships with the other ones.

Integrity Constraints

- Constraints of key, relationship and general.

- Key Constraint: a minimum subset of fields of a relationship that uniquely identifies the tuple.

- That is, field(s) defined as key must ensure that the selected line is unique.

Example

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

Primary Key

- A certain table can have several keys = candidate keys

- Primary key is defined by the DBA so as the DBMS make inquiries through it.

- Primary key well defined is important because it stimulates the creation and indexes, which makes the queries more performative.

Normal Forms

- Series of rules that ensure if a BD is well projected.
- It shows the importance of a well-defined primary key.
- Objective:

1) Ensure information without redundancy.

2) Ensure efficiency when obtaining data.

Normal forms

- 1st normal form:

Each line is an information. There cannot be repeated groups or attributes with more than one value.

PEOPLE = {ID+ NAME + ADDRESS + TELEPHONES}

PEOPLE = {ID + NAME + ADDRESS}

TELEPHONES = { PERSON_ID + TELEPHONE }

Normal forms

- 1st normal form:

ID	NAME	ADDRESS	TELEPHONES
XX	JOAO	AV JOAO	99999;88888;77777
YY	PEDRO	AV PEDRO	77776;5555

ID	Name	Address
XX	JOAO	AV JOAO
YY	PEDRO	AV PEDRO

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

Normal forms

- 2nd normal form:

All columns that do not participate in the primary key are dependent on all columns that compose the primary key.

STUDENTS_COURSES = {ID_STUDENT + ID_COURSE + GRADE + DESCRIPTION_COURSE }

STUDENTS_COURSES = {ID_STUDENT +ID_COURSE + GRADE}

COURSES = {ID_COURSE + DESCRIPTION}

General Restrictions

- Constraints mainly of business.
- Example: inclusion of age.
- The modern DBMS already have tools that allow to create these constraints.

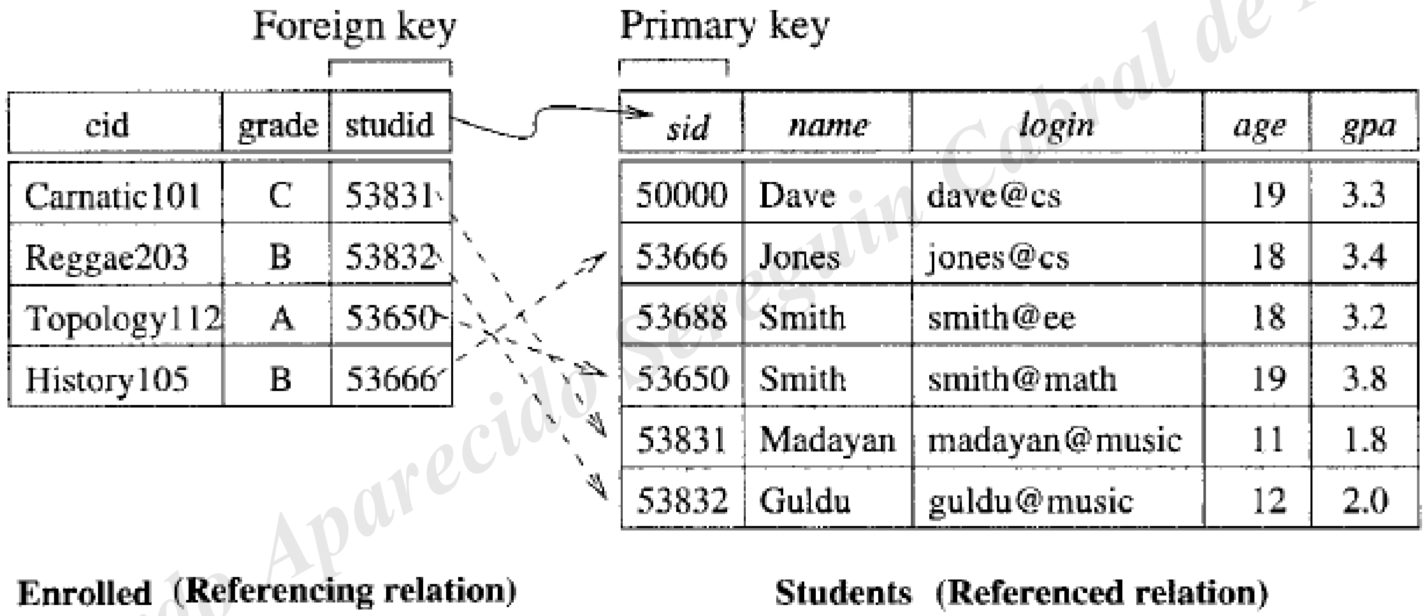
How to deal with models with more than one table?

Foreign Key

- Primary key from another table

- This key allows us to connect different tables to ensure the unity of the relationship.

- The name of the foreign key does not need to be the same as the name of the primary key = the content is important!



cid	grade	studid
Carnatic101	C	53831
Reggae203	B	53832
Topology112	A	53650
History105	B	53666

Enrolled **(Referencing relation)**

sid	*name*	*login*	*age*	*gpa*
50000	Dave	dave@cs	19	3.3
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Students **(Referenced relation)**

Specific Cases

- Insert Tuple <*55555, Art104, A*> in the courses with registration.
- Delete tuple <*53666, Jones, Jones@cs, 18, 3.4*> of students.
- Insert tuple <*55669, Margareth, MG@test, 21, 4*> in students.

Junction idea

Table 1

CPF	NOME
xxx	ze das couves
yyy	maria das desgraças

Table 2

CPF	IDADE	PIS
xxx	21	hhh
yyy	25	JJJ

Derived Table

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

ACID

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

- Set of properties in transactions of databases that are important to ensure the validity of data even if errors occur during the storage or more serious problems in the system, such as crashes or physical problems in a server. The ACID properties are fundamental for the processing of transactions in databases.

ACID

- Atomicity: Guarantees that each transaction is treated as a single "unit", which either succeeds completely or fails completely:
- Consistency: The data that are recorded must be always valid.
- Isolation: Allows the database to be in the same state in which it would be if the transactions are executed in sequence.
- Durability: Property of durability ensures that a transaction that has been committed (effective), will remain committed even in the case of a system failure.

Cardinality

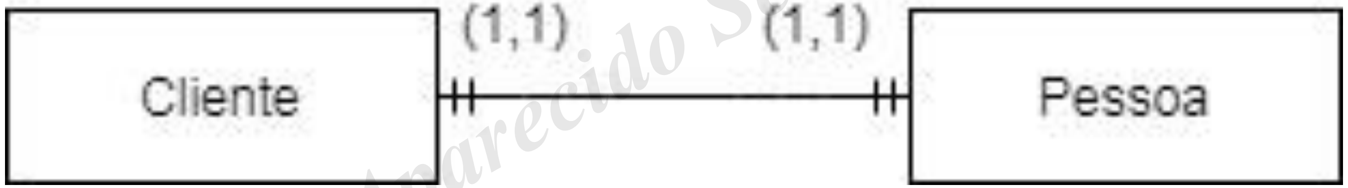
- Cardinality: indicates how many occurrences of an Entity participate in the minimum and maximum of the relationship.

Types of relationship:

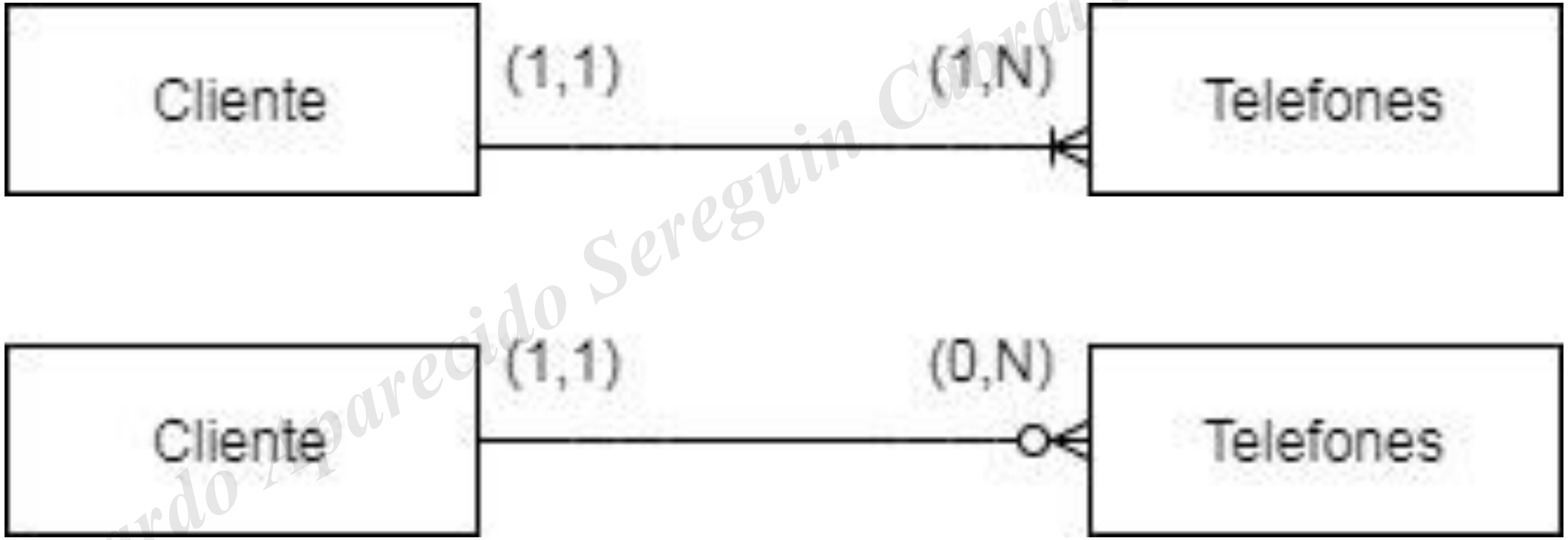
1) One to one;
2) Many to one;
3) Many to many.

One to one

- Minimum Cardinality: defines if the relationship is mandatory.
- Maximum Cardinality: defines the maximum number of occurrences of the Entity that can participate in the Relationship.

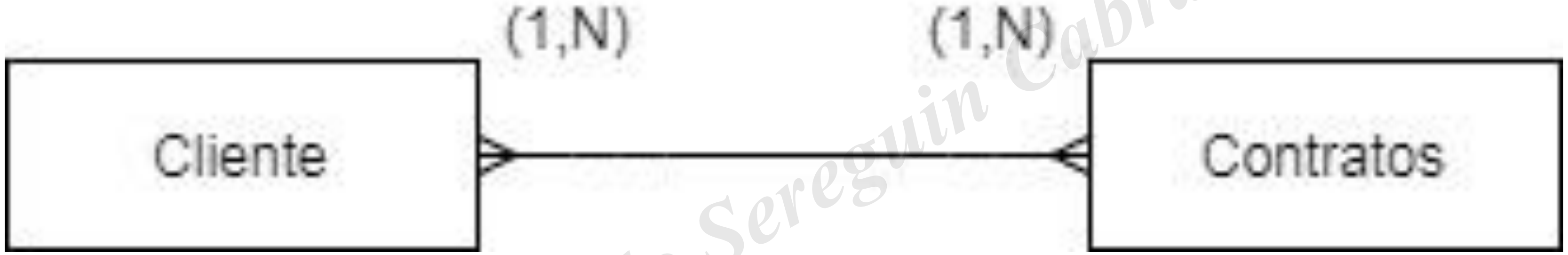


Many to one



*A responsabilidade pela idoneidade, originalidade e licitude dos conteúdos didáticos apresentados, é do professor.
Proibida a reprodução total ou parcial, sem autorização. Lei nº 9610/98

Many to Many



Operating with SQL

JOIN

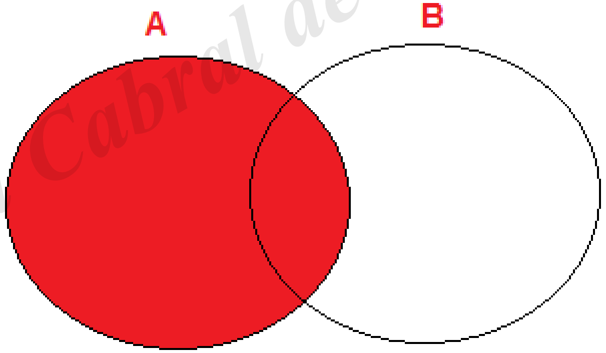
- Specifies how the join will be performed between two tables. For example:

Id_cliente	Pedido

Id_cliente	Nome	Endereço

Id_cliente	Nome	Endereço	Pedido

LEFT JOIN



SELECT [*DISTINCT*] **list of selection**

FROM list of **origin-1**

LEFT JOIN **list of origin-2**

ON **list of origin-1. field_in_common = list of origin-2. field_in_common**

WHERE **qualification**

LEFT JOIN

```
SELECT *
FROM requests
LEFT JOIN address
ON requests.Id_customer = address.id_customer
```

*A responsabilidade pela idoneidade, originalidade e licitude dos conteúdos didáticos apresentados, é do professor.
Proibida a reprodução total ou parcial, sem autorização. Lei nº 9610/98

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

- Up to now, only known values.
- If unknown = NULL.
- When the value is unknown or is not applied.

Example with NULL

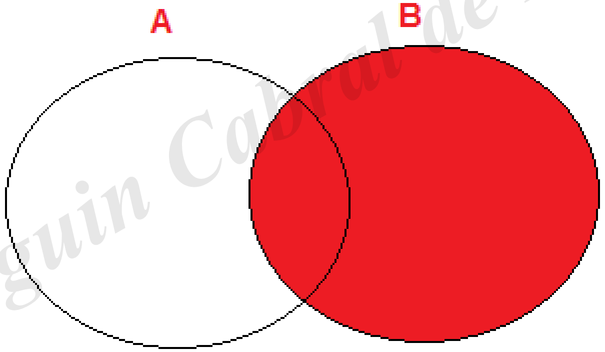
```
SELECT *
FROM requests_and_address
WHERE name IS NOT NULL
```

Id_cliente	Nome	Endereço	Pedido
xxx	joao	av joao	1

```
SELECT *
FROM requests_and_address
WHERE name IS NULL
```

Id_cliente	Nome	Endereço	Pedido
yyy	NULL	NULL	2

RIGHT JOIN



SELECT [*DISTINCT*] **list of selectio**

FROM list of **origin-1**

RIGHT JOIN **list of origin-2**

ON **list of origin-1. field_in_common = list of origin-2. field_in_common**

WHERE **qualification**

*A responsabilidade pela idoneidade, originalidade e licitude dos conteúdos didáticos apresentados, é do professor.
Proibida a reprodução total ou parcial, sem autorização. Lei nº 9610/98

RIGHT JOIN

```
SELECT *
FROM requests
RIGHT JOIN address
ON requests.Id_customer = address.id_customer
```

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

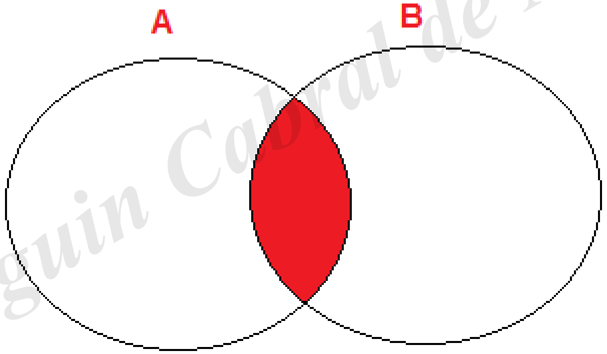
SELECT [*DISTINCT*] **list of selectior**

FROM list of **origin-1**

INNER JOIN **list of origin-2**

ON **list of origin-1. field_in_common = list of origin-2. field_in_common**

WHERE **qualification**



INNER JOIN

```
SELECT *
FROM requests
INNER JOIN address
ON requests.Id_customer = address.id_customer
```

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

- Tables of the same structure that will "stacked".

```
SELECT list of selection
FROM list of origin-1
UNION ALL
SELECT list of selection
FROM list of origin-2
```

UNION ALL

```
SELECT Id_customer
FROM requests
UNION ALL
SELECT Id_customer
FROM address
```

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

- Tables of the same structure that will "stacked".
- It is differentiate by applying a *DISTINCT*.

```
SELECT list of selection
FROM list of origin-1
UNION
SELECT list of selection
FROM list of origin-2
```

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

UNION

```
SELECT Id_customer
FROM requests
UNION
SELECT Id_customer
FROM address
```

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

- Result of the previous query can be used in the current
- Most common form:

SELECT **derived list of selection**

FROM **list of origin**

WHERE **column IN**

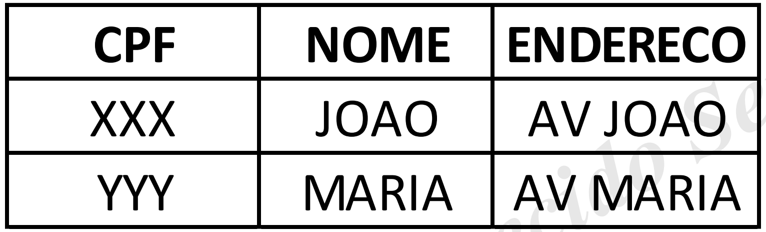
(

SELECT **original list of selection**

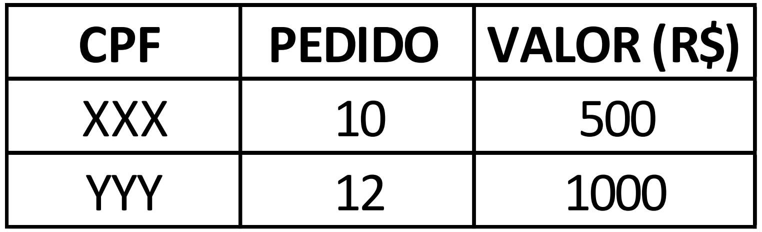
FROM list **of origin**

)

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, Name, Address
FROM Consumers
WHERE CPF IN
(
SELECT CPF
FROM Expenses
WHERE Value > 500
)
```

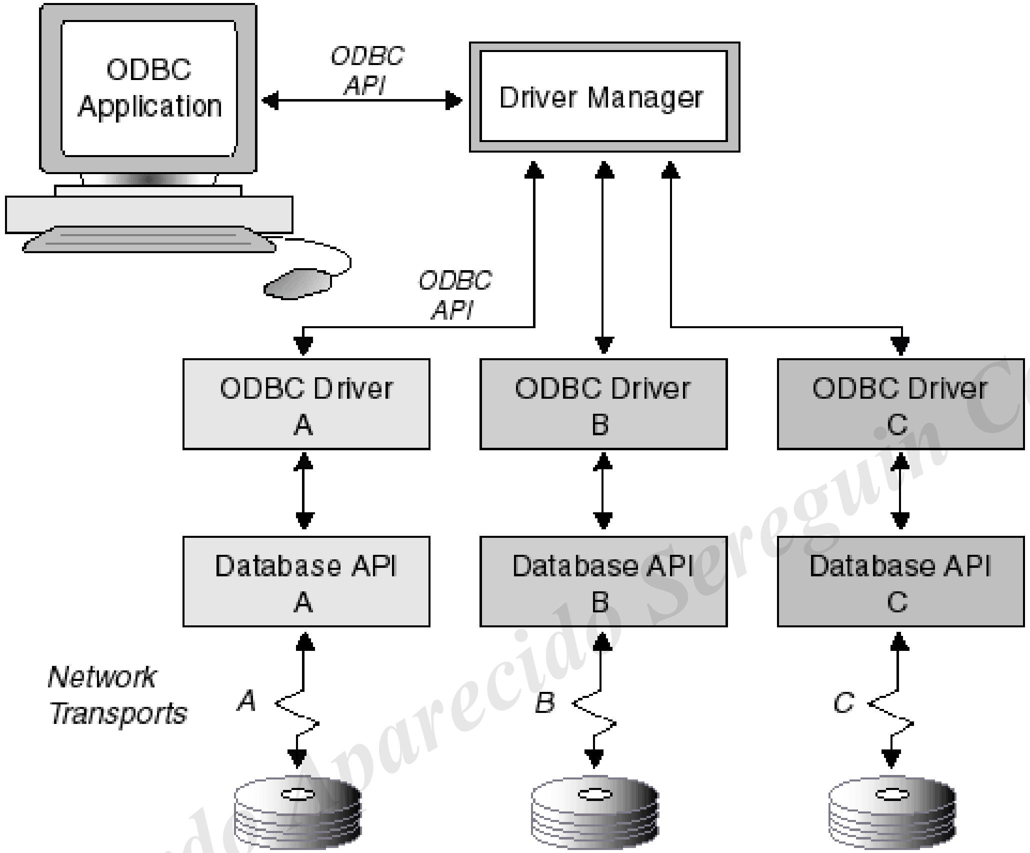
ODBC and JDBC

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

ODBC and JDBC

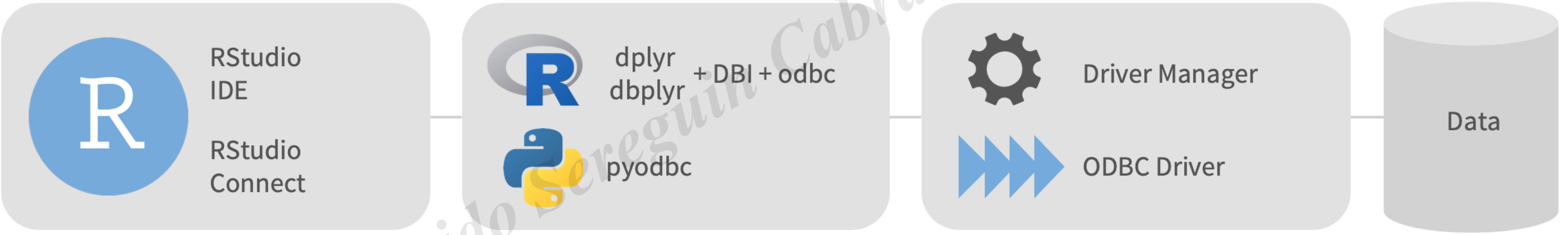
- Allows the execution of SQL within the bank from applications.
- Can access several data servers at the same time.
- All transactions occur through a **driver**.

ODBC and JDBC



https://docs.oracle.com/

Example in R Studio



RStudio Application

Programming Language

System Software

Data Source

*A responsabilidade pela idoneidade, originalidade e licitude dos conteúdos didáticos apresentados, é do professor.
Proibida a reprodução total ou parcial, sem autorização. Lei nº 9610/98

ODBC and JDBC

- Order:

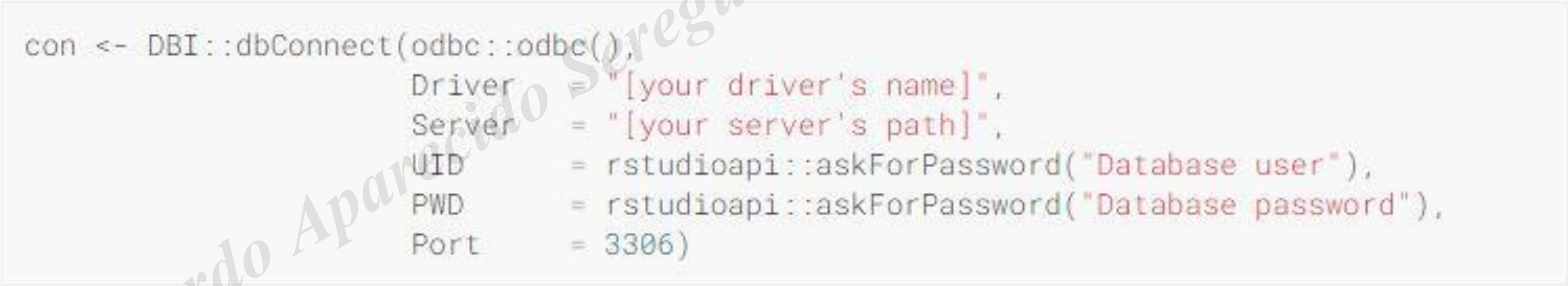
1. Select data origin.

2. Loads the respective driver.

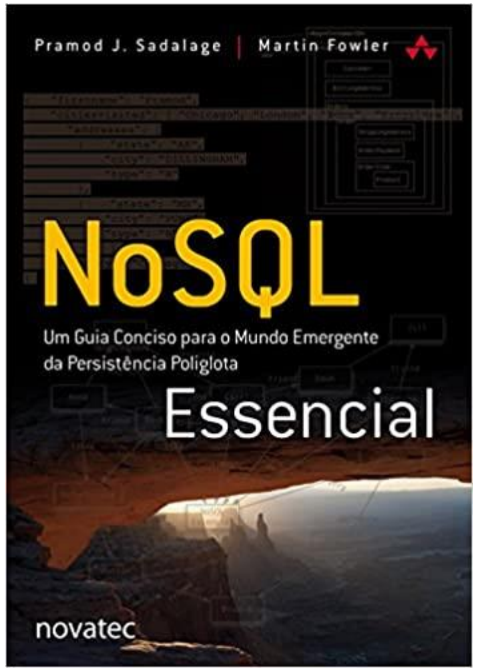
3. Establishes the connection to the origin.

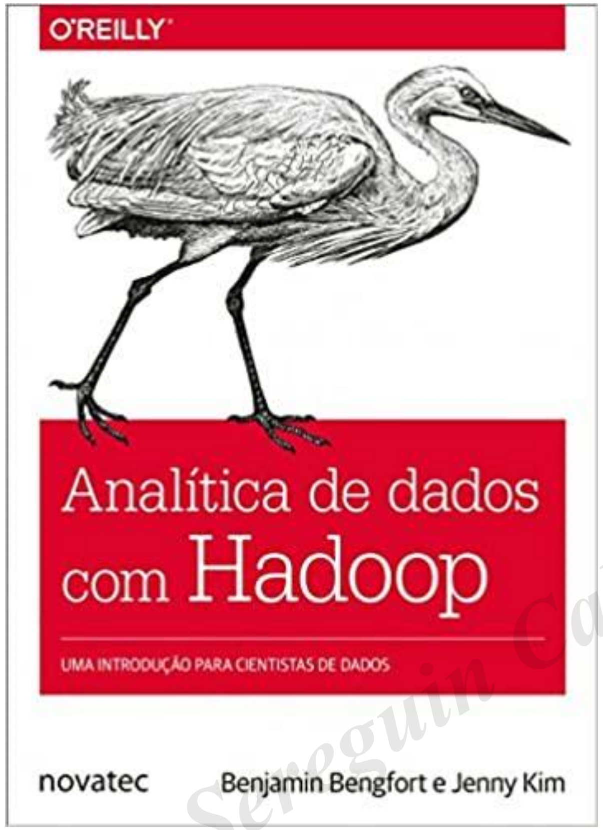
ODBC and JDBC

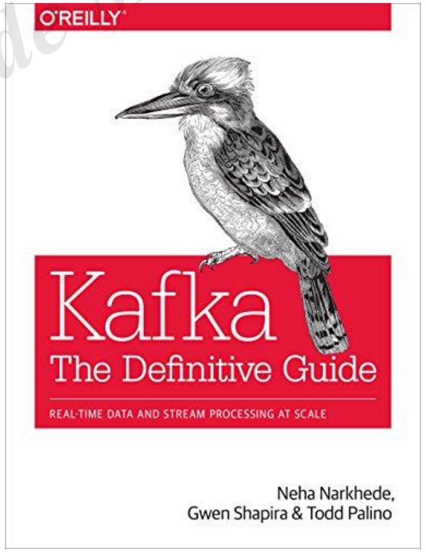
- Each connection has its characteristics.
- The connection string is defined with the bank.



```
con <- DBI::dbConnect(odbc::odbc(),
                      Driver   = "[your driver's name]",
                      Server   = "[your server's path]",
                      UID      = rstudioapi::askForPassword("Database user"),
                      PWD      = rstudioapi::askForPassword("Database password"),
                      Port     = 3306)
```


Pramod J. Sadalage | Martin Fowler
NoSQL
Um Guia Conciso para o Mundo Emergente da Persistência Poliglota
Essencial
novatec


O'REILLY
Analítica de dados com Hadoop
UMA INTRODUÇÃO PARA CIENTISTAS DE DADOS
novatec
Benjamin Bengfort e Jenny Kim


O'REILLY
Kafka
The Definitive Guide
REAL-TIME DATA AND STREAM PROCESSING AT SCALE
Neha Narkhede,
Gwen Shapira & Todd Palino

Discussion – future of databases

MBA
USP
ESALQ

Prof. Dr. Jeronymo Marcondes

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