

Aplicações - SQL

Banco de Dados: Teoria e Prática

André Santanchè

Instituto de Computação - UNICAMP

Março de 2016



Linguagens de Query

- Para manipulação e recuperação de dados
- Linguagens de Query (LQ) em BD:
 - Fundamentação formal
 - Subsidiária otimização
- LQ \neq linguagens de programação
 - não se espera que sejam “Turing completas”.
 - não pensadas para cálculos complexos.
 - suportam acessos simples e eficientes a extensos conjuntos de dados

Linguagens de

Uma linguagem é dita “Turing completa” se puder ser demonstrado que ela é computacionalmente equivalente à máquina de Turing.

- Para manipulação e recuperação
- Linguagens de Query (LQ) em BD:
 - Fundamentação formal
 - Subsidiária otimização
- LQ <> linguagens de programação
 - não se espera que sejam “Turing completas”.
 - não pensadas para cálculos complexos.
 - suportam acessos simples e eficientes a extensos conjuntos de dados

SQL

- SQL - Structured Query Language
- Originalmente: SEQUEL - Structured English QUERy Language
- Criada pela IBM Research
 - Interface BD Relacional → SYSTEM R

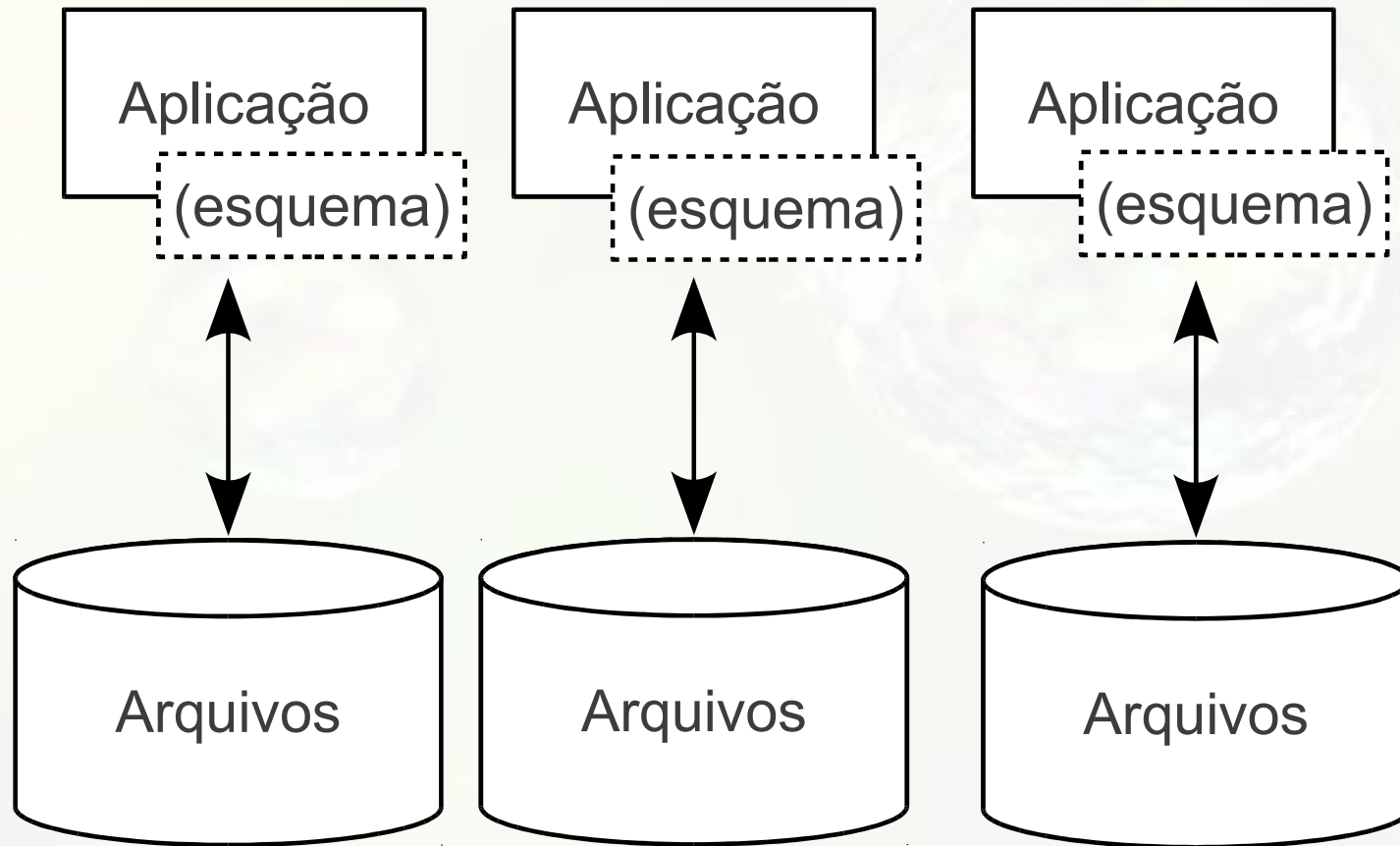
SQL

Padronização

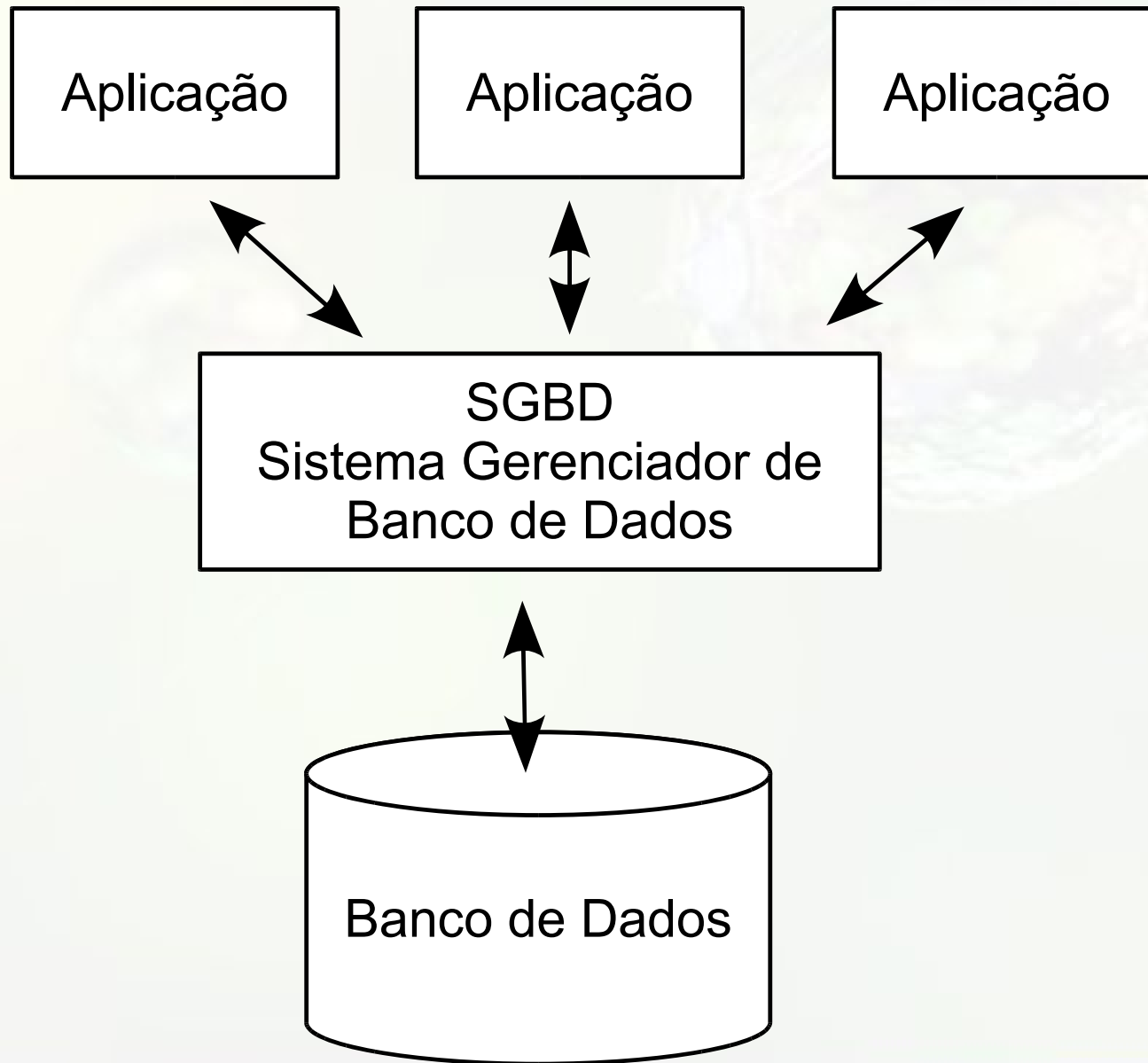
- ANSI + ISO
- SQL-86 ou SQL1
- SQL-92 ou SQL2
- SQL:1999 ou SQL3
- SQL:2003
- SQL:2006

Aplicações e Armazenamento

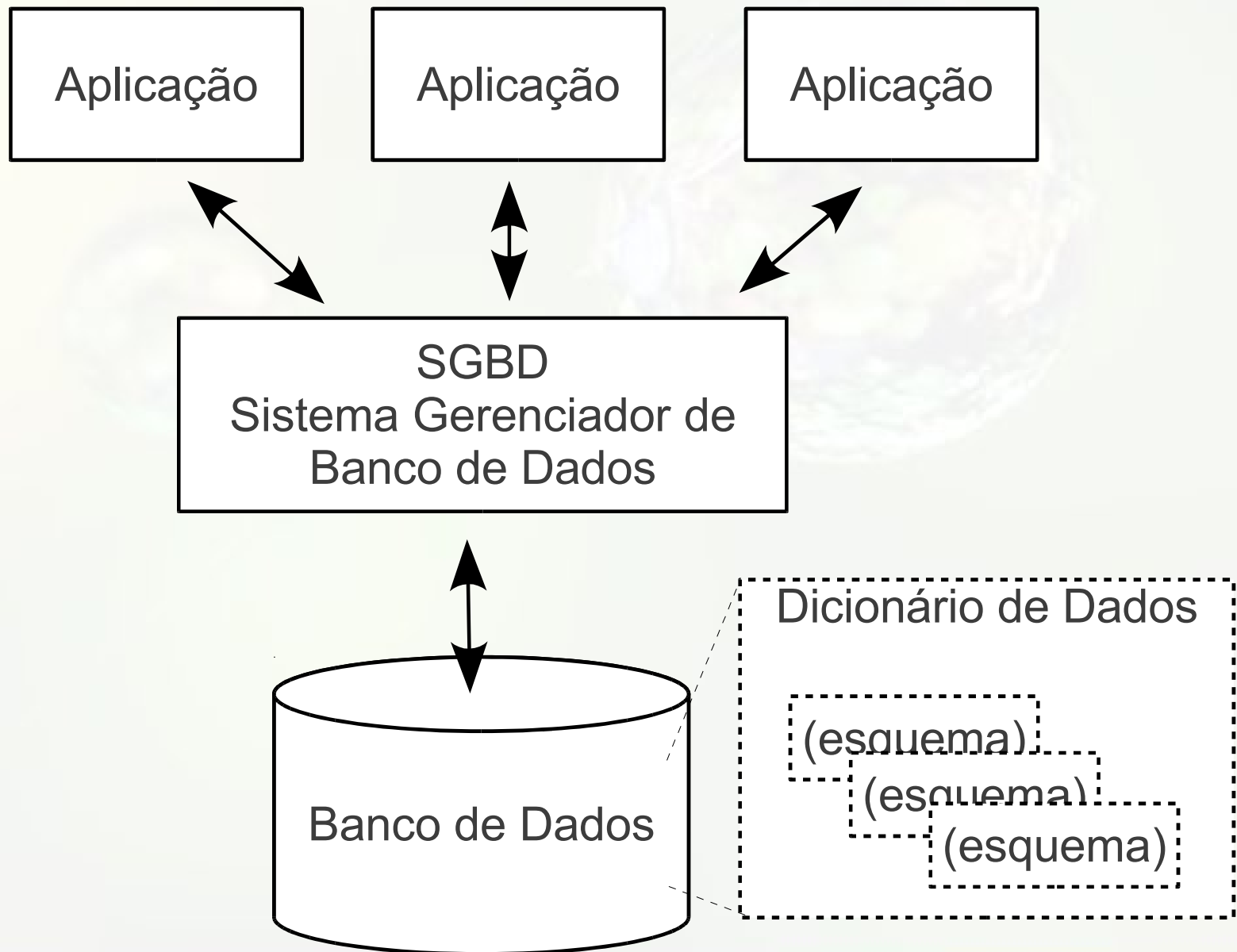
Arquivos



Aplicações e Armazenamento SGBD



Dicionário de Dados

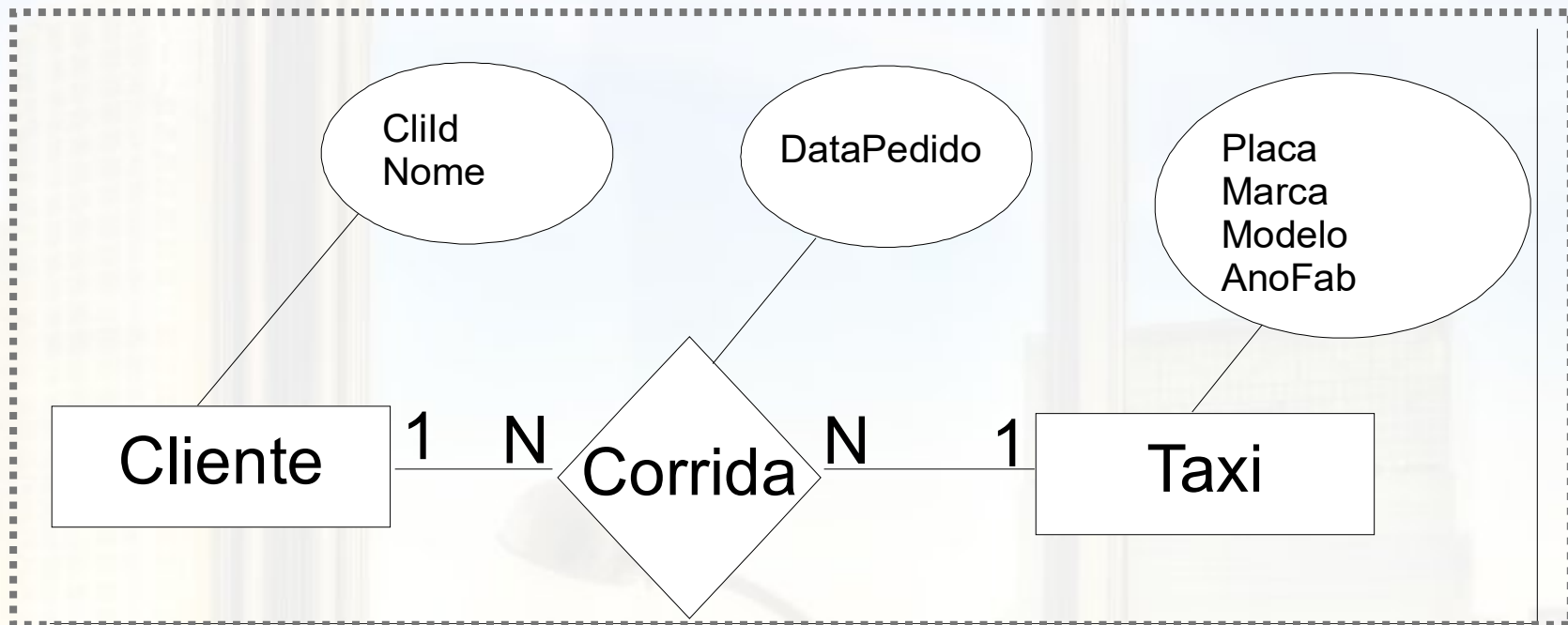




Caso Prático - Taxis

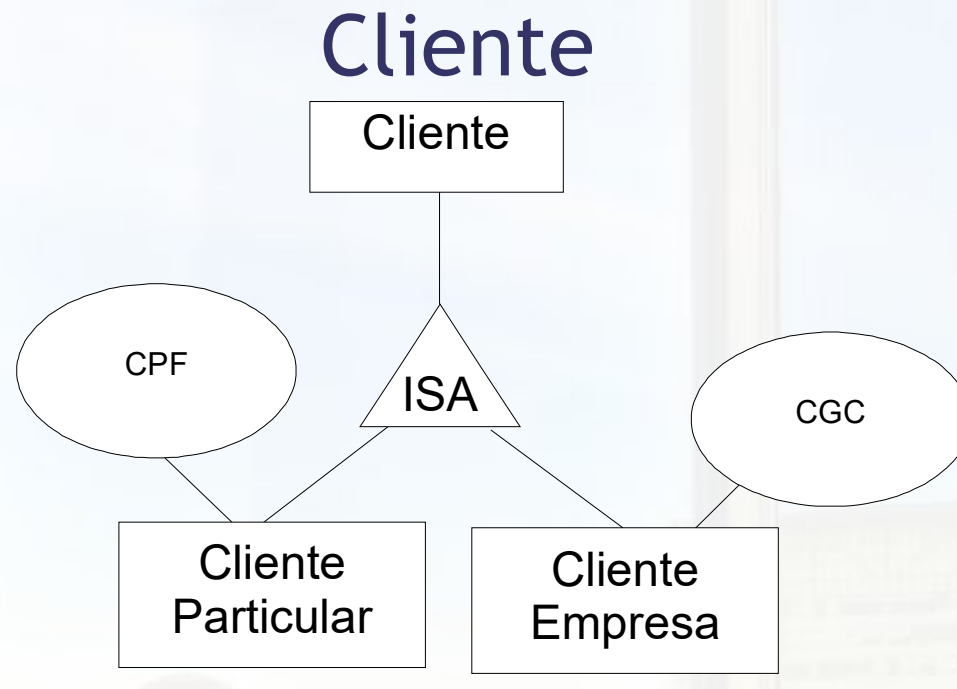
Esquema Conceitual - Exemplo

Táxis



Este é um subconjunto do Estudo de Caso proposto “Despacho e controle de Táxis via terminais móveis ligados on-line com um sistema multi-usuário” por prof. Geovane Cayres Magalhães

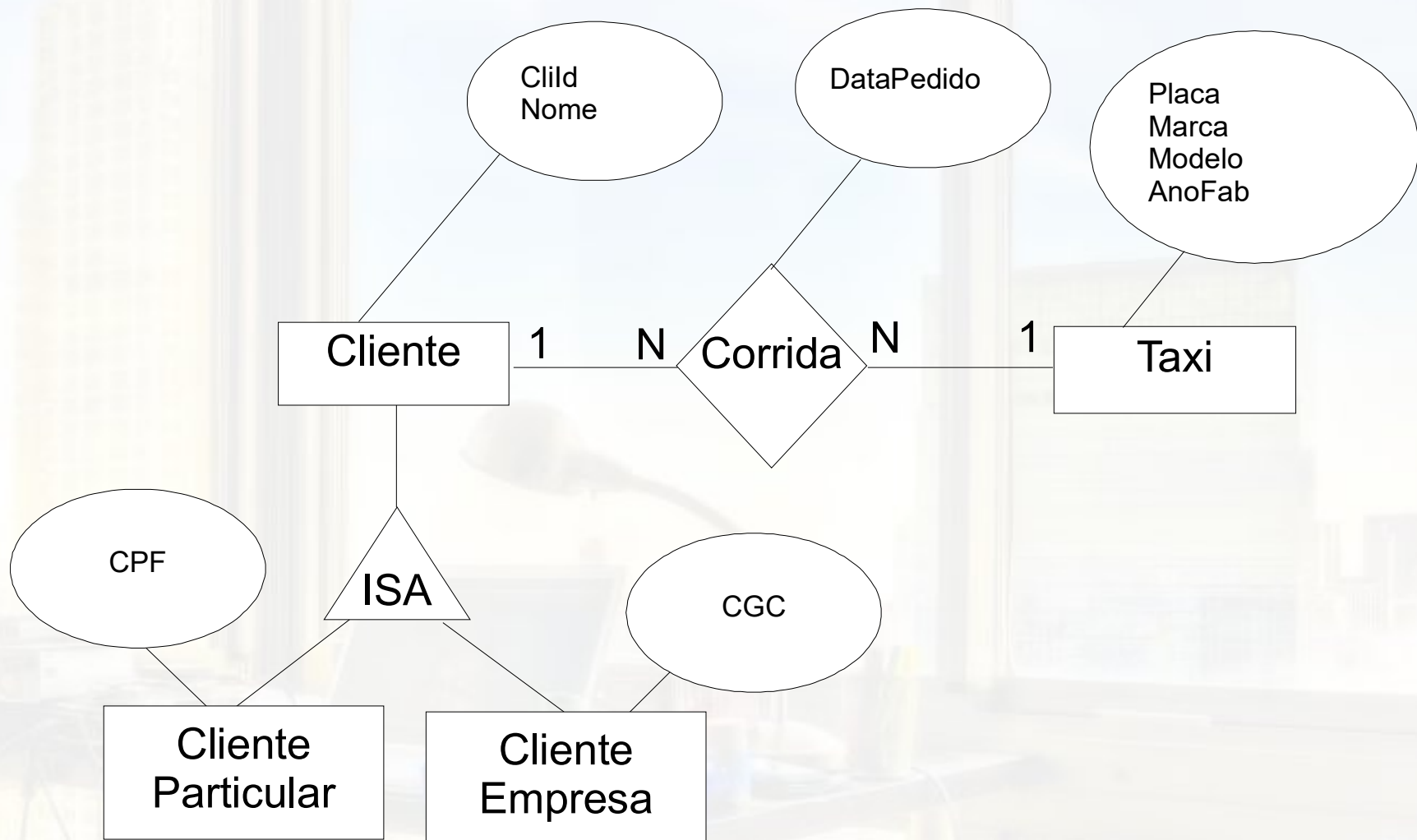
Esquema Conceitual - Exemplo



Para ilustrar o tema apresentado, foram acrescentadas duas entidades que são especialização de Cliente. A primeira representa um indivíduo que irá pagar a conta, a segunda representa um funcionário de uma empresa conveniada, para a qual a conta será enviada. Um cliente pode pertencer a ambas especializações.

Esquema Conceitual completo

Táxis



Tabelas para exemplo - Táxis

Cliente Particular (CP)

<u>CliId</u>	Nome	CPF
1532	Asdrúbal	448.754.253-65
1755	Doriana	567.387.387-44
1780	Quincas	546.373.762-02

Cliente Empresa (CE)

<u>CliId</u>	Nome	CGC
1532	Asdrúbal	754.856.965/0001-54
1644	Jepeto	478.652.635/0001-75
1780	Quincas	554.663.996/0001-87
1982	Zandor	736.952.369/0001-23



Tabelas para exemplo - Táxis

Táxi (TX)

<u>Placa</u>	Marca	Modelo	AnoFab
DAE6534	Ford	Fiesta	1999
DKL4598	Wolkswagen	Gol	2001
DKL7878	Ford	Fiesta	2001
JDM8776	Wolkswagen	Santana	2002
JJM3692	Chevrolet	Corsa	1999



Corrida (R1)

<u>ClId</u>	<u>Placa</u>	<u>DataPedido</u>
1755	DAE6534	15/02/2003
1982	JDM8776	18/02/2003



CREATE SCHEMA

- CREATE SCHEMA <esquema>
AUTHORIZATION <id_autorizado>
- Java: executeUpdate(...)

CREATE TABLE

- CREATE TABLE <tabela>
(<campo₁> <tipo> [NULL|NOT NULL] [restrição],
[...,
<campo_n> <tipo> [NULL|NOT NULL] [restrição],
PRIMARY KEY <chave_primaria>])
- Java: executeUpdate(...)

CREATE TABLE

```
CREATE TABLE Taxi (  
    Placa VARCHAR(7) NOT NULL,  
    Marca VARCHAR(30) NOT NULL,  
    Modelo VARCHAR(30) NOT NULL,  
    AnoFab INTEGER,  
    Licenca VARCHAR(9),  
    PRIMARY KEY(Placa)  
);
```

```
CREATE TABLE Cliente (  
    CliId VARCHAR(4) NOT NULL,  
    Nome VARCHAR(80) NOT NULL,  
    CPF VARCHAR(14) NOT NULL,  
    PRIMARY KEY(CliId)  
);
```

CREATE TABLE FOREIGN KEY

- CREATE TABLE <tabela>

...

FOREIGN KEY (<coluna_estr>₁ [, ..., <coluna_estr>_n])

REFERENCES <tabela_ref> ([<coluna_ref> [, ..., <coluna_ref>]])

[ON DELETE <ação_ref>]

[ON UPDATE <ação_ref>]

- <ação_ref>

- NO ACTION → impede a ação na tabela mestre < tabela_ref>
- CASCADE → propaga a ação da tabela mestre
- SET NULL → valores de referências alterados para nulo
- SET DEFAULT → valores de referências alterados para default

CREATE TABLE

FOREIGN KEY

```
CREATE TABLE Corrida (  
  CliId VARCHAR(4) NOT NULL,  
  Placa VARCHAR(7) NOT NULL,  
  DataPedido DATE NOT NULL,  
  PRIMARY KEY (CliId, Placa, DataPedido),  
  FOREIGN KEY (CliId)  
    REFERENCES Cliente (CliId)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION,  
  FOREIGN KEY (Placa)  
    REFERENCES Taxi (Placa)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION  
);
```

Exercício 1

- Escreva um comando SQL para criar os esquemas:
 - Pessoa(nome, nome_da_mãe, ano_nascimento, nome_cidade_natal)
 - nome_cidade_natal → CHE Cidade
 - Cidade(nome_cidade, sigla_estado)

INSERT

- INSERT INTO <tabela>
[(<campo₁>[,..., <campo_n>])]
VALUES (<valor₁>[,..., <valor_n>])
- executeUpdate(...)

Exemplos INSERT

```
INSERT INTO Cliente  
VALUES ('1755', 'Doriana', '567.387.387-44');
```

```
INSERT INTO Taxi  
VALUES ('DAE6534', 'Ford', 'Fiesta', 1999, 'MN572345');
```

```
INSERT INTO Corrida  
VALUES ('1755', 'DAE6534', '2003-02-15');
```

Comandos INSERT para Taxi

```
INSERT INTO Cliente VALUES ('1532', 'Asdrúbal', '448.754.253-65');
```

```
INSERT INTO Cliente VALUES ('1755', 'Doriana', '567.387.387-44');
```

```
INSERT INTO Cliente VALUES ('1780', 'Quincas', '546.373.762-02');
```

```
INSERT INTO Taxi VALUES ('DAE6534', 'Ford', 'Fiesta', 1999, 'MN572345');
```

```
INSERT INTO Taxi VALUES ('DKL4598', 'Wolkswagen', 'Gol', 2001, 'AU876543');
```

```
INSERT INTO Taxi VALUES ('DKL7878', 'Ford', 'Fiesta', 2001, 'OP102938');
```

```
INSERT INTO Taxi VALUES ('JDM8776', 'Wolkswagen', 'Santana', 2002, 'QM365923');
```

```
INSERT INTO Taxi VALUES ('JJM3692', 'Chevrolet', 'Corsa', 1999, 'UU335577');
```

```
INSERT INTO Corrida VALUES ('1755', 'DAE6534', '2003-02-15');
```

```
INSERT INTO Corrida VALUES ('1780', 'JDM8776', '2003-02-18');
```

```
INSERT INTO Corrida VALUES ('1755', 'DKL7878', '2003-02-16');
```

```
INSERT INTO Corrida VALUES ('1780', 'DKL4598', '2003-02-17');
```

```
INSERT INTO Corrida VALUES ('1532', 'DKL4598', '2003-02-18');
```

```
INSERT INTO Corrida VALUES ('1780', 'DAE6534', '2003-02-16');
```


Exercício 2

- Escreva um comando SQL para inserir uma tupla na tabela Pessoa com os seus dados e dados de familiares próximos (cerca de 2 linhas). Preencha a tabela Cidade com as cidades listadas na tabela Pessoa e suas respectivas siglas de estado. Use dados fictícios se preciso.

SELECT

- `SELECT * | <campo1>[,..., <campon>]
FROM <tabela1>[,..., <tabelan>]
WHERE <condição/junção>`
- `executeQuery(...)`


SELECT Projeção

SELECT Marca, Modelo FROM Taxi

<u>Placa</u>	Marca	Modelo	AnoFab
DAE6534	Ford	Fiesta	1999
DKL4598	Wolkswagen	Gol	2001
DKL7878	Ford	Fiesta	2001
JDM8776	Wolkswagen	Santana	2002
JJM3692	Chevrolet	Corsa	1999

SELECT Projeção

SELECT **Marca**, **Modelo** FROM Taxi



<u>Placa</u>	Marca	Modelo	AnoFab
DAE6534	Ford	Fiesta	1999
DKL4598	Wolkswagen	Gol	2001
DKL7878	Ford	Fiesta	2001
JDM8776	Wolkswagen	Santana	2002
JJM3692	Chevrolet	Corsa	1999

SELECT Projeção

SELECT **Marca**, **Modelo** FROM Taxi

Marca	Modelo
Ford	Fiesta
Wolkswagen	Gol
Ford	Fiesta
Wolkswagen	Santana
Chevrolet	Corsa

SELECT Seleção

SELECT * FROM Taxi WHERE AnoFab > 2000

<u>Placa</u>	Marca	Modelo	AnoFab
DAE6534	Ford	Fiesta	1999
DKL4598	Wolkswagen	Gol	2001
DKL7878	Ford	Fiesta	2001
JDM8776	Wolkswagen	Santana	2002
JJM3692	Chevrolet	Corsa	1999

SELECT Seleção

SELECT * FROM Taxi WHERE AnoFab > 2000

<u>Placa</u>	Marca	Modelo	AnoFab
DAE6534	Ford	Fiesta	1999
DKL4598	Wolkswagen	Gol	2001
DKL7878	Ford	Fiesta	2001
JDM8776	Wolkswagen	Santana	2002
JJM3692	Chevrolet	Corsa	1999

SELECT Seleção

SELECT * FROM Taxi WHERE AnoFab > 2000

<u>Placa</u>	Marca	Modelo	AnoFab
DKL4598	Wolkswagen	Gol	2001
DKL7878	Ford	Fiesta	2001
JDM8776	Wolkswagen	Santana	2002

Exercício 3

- Para a tabelas que você montou no exercício 1, escreva um comando SQL que retorne:
 - a) nomes de todas as mães
 - b) nomes de todas as mães com filhos maiores de 12 anos

SELECT Between

SELECT * FROM Taxi WHERE AnoFab **BETWEEN**
1999 AND 2001;

<u>Placa</u>	Marca	Modelo	AnoFab
DAE6534	Ford	Fiesta	1999
DKL4598	Wolkswagen	Gol	2001
DKL7878	Ford	Fiesta	2001
JDM8776	Wolkswagen	Santana	2002
JJM3692	Chevrolet	Corsa	1999

SELECT IN

SELECT * FROM Taxi WHERE Modelo
IN ('Corsa', 'Santana');

<u>Placa</u>	Marca	Modelo	AnoFab
DAE6534	Ford	Fiesta	1999
DKL4598	Wolkswagen	Gol	2001
DKL7878	Ford	Fiesta	2001
JDM8776	Wolkswagen	Santana	2002
JJM3692	Chevrolet	Corsa	1999

SELECT LIKE

SELECT Placa, Marca FROM Taxi
WHERE Marca LIKE 'Wolks%';

	placa character varying(7)	marca character varying(30)
1	DKL4598	Wolkswagen
2	JDM8776	Wolkswagen

- % → qualquer cadeia com 0 a n caracteres
- _ → exatamente um caractere (qualquer)
- = → caractere de escape
 - e.g., serve para encontrar um caractere _

AS (alias)

- SELECT <campo₁> [AS] <alias₁>
[,..., <campo_n> [AS] <alias_n>]
...
- SELECT ...
FROM <tabela₁> [AS] <alias₁>
[,..., <tabela_n> [AS] <alias_n>]
...

SELECT Alias

```
SELECT Cl.Clid, Cl.Nome  
FROM Cliente AS Cl;
```

	cliid character varying(4)	nome character varying(80)
1	1532	Asdrúbal
2	1755	Doriana
3	1780	Quincas

SELECT

Funções

SELECT Placa, Modelo || ' - ' || upper(Marca) AS
MarcaModelo FROM Taxi;

	placa character	marcamodelo text
1	DAE6534	Fiesta - FORD
2	DKL4598	Gol - WOLKSWAGEN
3	DKL7878	Fiesta - FORD
4	JDM8776	Santana - WOLKSWAGEN
5	JJM3692	Corsa - CHEVROLET

SELECT DISTINCT e ALL

- **SELECT DISTINCT ...**
 - Seleciona apenas tuplas distintas (definição do modelo relacional)
- **SELECT ALL ...**
 - A cláusula ALL é implícita se não especificada (o padrão do SQL é não seguir a unicidade do modelo relacional)

SELECT ORDER BY

- SELECT ...
ORDER BY <campo₁>[,..., <campo_n>]
[DESC]
 - Ordena de acordo com a lista de campos
 - Use DESC para ordem decrescente

SELECT ORDER BY

```
SELECT nome FROM Cliente  
ORDER BY nome DESC;
```

	nome character varying(80)
1	Quincas
2	Doriana
3	Asdrúbal

Produto Cartesiano

- **SELECT ...**
FROM <tabela₁>, <tabela₂>
<não há condição que ligue tabelas>
- Não há associação de atributo da <tabela₁>
com atributo da <tabela₂>

Produto Cartesiano

```
SELECT Cliente.CliId, Cliente.Nome,  
Corrida.CliId, Corrida.Placa,  
Corrida.DataPedido  
FROM Cliente, Corrida
```

<u>CliId</u>	Nome
1532	Asdrúbal
1755	Doriana
1780	Quincas

<u>CliId</u>	<u>Placa</u>	<u>DataPedido</u>
1755	DAE6534	15/02/2003
1982	JDM8776	18/02/2003

Produto Cartesiano

```
SELECT Cliente.CliId, Cliente.Nome,  
       Corrida.CliId, Corrida.Placa,  
       Corrida.DataPedido  
FROM Cliente, Corrida
```

<u>CliId</u>	Nome
1532	Asdrúbal
1755	Doriana
1780	Quincas

<u>CliId</u>	<u>Placa</u>	<u>DataPedido</u>
1755	DAE6534	15/02/2003
1982	JDM8776	18/02/2003

(CliId)	Nome	(CliId)	Placa	DataPedido
1532	Asdrúbal	1755	DAE6534	15/02/2003
1532	Asdrúbal	1982	JDM8776	18/02/2003

Produto Cartesiano

```
SELECT Cliente.CliId, Cliente.Nome,  
Corrida.CliId, Corrida.Placa,  
Corrida.DataPedido  
FROM Cliente, Corrida
```

<u>CliId</u>	Nome
1532	Asdrúbal
1755	Doriana
1780	Quincas

<u>CliId</u>	<u>Placa</u>	<u>DataPedido</u>
1755	DAE6534	15/02/2003
1982	JDM8776	18/02/2003

(CliId)	Nome	(CliId)	Placa	DataPedido
1532	Asdrúbal	1755	DAE6534	15/02/2003
1532	Asdrúbal	1982	JDM8776	18/02/2003
1755	Doriana	1755	DAE6534	15/02/2003
1755	Doriana	1982	JDM8776	18/02/2003

Produto Cartesiano

```
SELECT Cliente.CliId, Cliente.Nome,  
       Corrida.CliId, Corrida.Placa,  
       Corrida.DataPedido  
FROM Cliente, Corrida
```

<u>CliId</u>	Nome
1532	Asdrúbal
1755	Doriana
1780	Quincas

<u>CliId</u>	<u>Placa</u>	<u>DataPedido</u>
1755	DAE6534	15/02/2003
1982	JDM8776	18/02/2003

(CliId)	Nome	(CliId)	Placa	DataPedido
1532	Asdrúbal	1755	DAE6534	15/02/2003
1532	Asdrúbal	1982	JDM8776	18/02/2003
1755	Doriana	1755	DAE6534	15/02/2003
1755	Doriana	1982	JDM8776	18/02/2003
1780	Quincas	1755	DAE6534	15/02/2003
1780	Quincas	1982	JDM8776	18/02/2003

Junção - Join (1)

- SELECT ...
FROM <tabela₁>, <tabela₂>
WHERE <tabela₁>.<atr> = <tabela₂>.<atr>
- Join implícito

Join (1)

```
SELECT Cliente.CliId, Cliente.Nome,  
Corrida.CliId, Corrida.Placa,  
Corrida.DataPedido  
FROM Cliente, Corrida  
WHERE Cliente.CliId = Corrida.CliId
```

(CliId)	Nome	(CliId)	Placa	DataPedido
1532	Asdrúbal	1755	DAE6534	15/02/2003
1532	Asdrúbal	1982	JDM8776	18/02/2003
1755	Doriana	1755	DAE6534	15/02/2003
1755	Doriana	1982	JDM8776	18/02/2003
1780	Quincas	1755	DAE6534	15/02/2003
1780	Quincas	1982	JDM8776	18/02/2003

Obter, para cada corrida, informações de id e nome do cliente bem como placa e data da corrida

Join (1)

```
SELECT Cliente.CliId, Cliente.Nome,  
Corrida.CliId, Corrida.Placa,  
Corrida.DataPedido  
FROM Cliente, Corrida  
WHERE Cliente.CliId = Corrida.CliId
```

(CliId)	Nome	(CliId)	Placa	DataPedido
1532	Asdrúbal	1755	DAE6534	15/02/2003
1532	Asdrúbal	1982	JDM8776	18/02/2003
1755	Doriana	1755	DAE6534	15/02/2003
1755	Doriana	1982	JDM8776	18/02/2003
1780	Quincas	1755	DAE6534	15/02/2003
1780	Quincas	1982	JDM8776	18/02/2003

Join (1)

```
SELECT Cliente.CliId, Cliente.Nome,  
Corrida.CliId, Corrida.Placa,  
Corrida.DataPedido  
FROM Cliente, Corrida  
WHERE Cliente.CliId = Corrida.CliId
```

(CliId)	Nome	(CliId)	Placa	DataPedido
1755	Doriana	1755	DAE6534	15/02/2003

Exercício 4

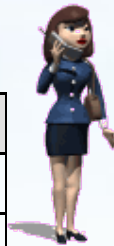
- Para a tabelas que você montou no exercício 1, desenhe uma tabela com mais dados (fictícios se preferir) e escreva um comando SQL que retorne:
 - nomes de parentes que nasceram no mesmo estado que você
 - retorne todas as duplas de irmãos (não se preocupe com duplicidade de irmãos)

Desafio

- Qual o modelo de Taxi para cada Corrida?

Cliente (C)

<u>CliId</u>	Nome	CPF
1532	Asdrúbal	448.754.253-65
1755	Doriana	567.387.387-44
1780	Quincas	546.373.762-02



Táxi (TX)



<u>Placa</u>	Marca	Modelo	AnoFab
DAE6534	Ford	Fiesta	1999
DKL4598	Wolkswagen	Gol	2001
DKL7878	Ford	Fiesta	2001
JDM8776	Wolkswagen	Santana	2002
JJM3692	Chevrolet	Corsa	1999



Corrida (R1)

<u>CliId</u>	<u>Placa</u>	<u>DataPedido</u>
1755	DAE6534	15/02/2003
1982	JDM8776	18/02/2003

Modelo de Taxi para cada Corrida

```
SELECT Co.DataPedido, Co.Placa, T.Modelo  
FROM Corrida Co, Taxi T  
WHERE Co.Placa = T.Placa;
```

Cliente (C)

<u>CliId</u>	Nome	CPF
1532	Asdrúbal	448.754.253-65
1755	Doriana	567.387.387-44
1780	Quincas	546.373.762-02



Táxi (TX)



<u>Placa</u>	Marca	Modelo	AnoFab
DAE6534	Ford	Fiesta	1999
DKL4598	Wolkswagen	Gol	2001
DKL7878	Ford	Fiesta	2001
JDM8776	Wolkswagen	Santana	2002
JJM3692	Chevrolet	Corsa	1999



Corrida (R1)

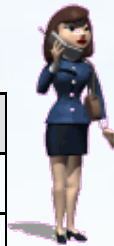
<u>ClId</u>	<u>Placa</u>	<u>DataPedido</u>
1755	DAE6534	15/02/2003
1982	JDM8776	18/02/2003

Desafio

- Quais os modelos de Taxi tomados por cada Cliente?

Cliente (C)

<u>CliId</u>	Nome	CPF
1532	Asdrúbal	448.754.253-65
1755	Doriana	567.387.387-44
1780	Quincas	546.373.762-02



Táxi (TX)



<u>Placa</u>	Marca	Modelo	AnoFab
DAE6534	Ford	Fiesta	1999
DKL4598	Wolkswagen	Gol	2001
DKL7878	Ford	Fiesta	2001
JDM8776	Wolkswagen	Santana	2002
JJM3692	Chevrolet	Corsa	1999



Corrida (R1)

<u>ClId</u>	<u>Placa</u>	<u>DataPedido</u>
1755	DAE6534	15/02/2003
1982	JDM8776	18/02/2003

Modelos de Taxi por Cliente

```
SELECT Cl.Nome, Co.DataPedido, Co.Placa, T.Modelo  
FROM Cliente Cl, Corrida Co, Taxi T  
WHERE Cl.CliId = Co.CliId AND Co.Placa = T.Placa;
```

Cliente (C)

<u>CliId</u>	Nome	CPF
1532	Asdrúbal	448.754.253-65
1755	Doriana	567.387.387-44
1780	Quincas	546.373.762-02



Táxi (TX)



<u>Placa</u>	Marca	Modelo	AnoFab
DAE6534	Ford	Fiesta	1999
DKL4598	Wolksvagen	Gol	2001
DKL7878	Ford	Fiesta	2001
JDM8776	Wolksvagen	Santana	2002
JJM3692	Chevrolet	Corsa	1999



Corrida (R1)

<u>CliId</u>	<u>Placa</u>	<u>DataPedido</u>
1755	DAE6534	15/02/2003
1982	JDM8776	18/02/2003

Agrupamento



Funções de Agregação

- COUNT(*) ⇒ contagem
- SUM(<coluna>) ⇒ soma
- AVG(<coluna>) ⇒ média
- MAX(<coluna>) ⇒ maior valor
- MIN(<coluna>) ⇒ menor valor

AGREGAÇÃO

SELECT AVG(anofab) FROM Taxi;

Táxi:

<u>Placa</u>	Marca	Modelo	AnoFab
DAE6534	Ford	Fiesta	1999
DKL4598	Wolkswagen	Gol	2001
DKL7878	Ford	Fiesta	2001
JDM8776	Wolkswagen	Santana	2002
JJM3692	Chevrolet	Corsa	1999

Resultado:

avg

2000.4

**Média de ano de
fabricação**

GROUP BY

- **SELECT * | <campo₁>[,..., <campo_n>]
FROM <tabela₁>[,..., <tabela_n>]
WHERE <condição/junção>
GROUP BY <coluna_agrupar>
HAVING <condição_grupo>**

GROUP BY

SELECT modelo, count(*) FROM Taxi GROUP BY
modelo;

Táxi:

<u>Placa</u>	Marca	Modelo	AnoFab
DAE6534	Ford	Fiesta	1999
DKL4598	Wolkswagen	Gol	2001
DKL7878	Ford	Fiesta	2001
JDM8776	Wolkswagen	Santana	2002
JJM3692	Chevrolet	Corsa	1999

Resultado:

modelo	count
Gol	1
Corsa	1
Santana	1
Fiesta	2

Contagem por modelo

GROUP BY

```
SELECT modelo, anofab, count(*) FROM Taxi  
GROUP BY modelo, anofab;
```

modelo	anofab	count
Escort	2001	1
Escort	2000	1
Santana	2002	1
Fiesta	1999	1
Corsa	1999	1
Gol	2000	1
Santana	1998	2
Fiesta	2001	1
Gol	2001	1

Resultado:

(usando dados completos)

Contagem por ano de fabricação do modelo

Exercício 6

- Escreva uma sentença SQL, baseada no esquema abaixo, que retorne o número de pessoas da família em cada estado:
 - Pessoa(nome, nome_da_mãe, ano_nascimento, nome_cidade_natal)
 - nome_cidade_natal → CHE Cidade
 - Cidade(nome_cidade, sigla_estado)



Consultas Aninhadas

SELECT Seleção

SELECT * FROM Taxi WHERE AnoFab > 2000

<u>Placa</u>	Marca	Modelo	AnoFab
DAE6534	Ford	Fiesta	1999
DKL4598	Wolksvagen	Gol	2001
DKL7878	Ford	Fiesta	2001
JDM8776	Wolksvagen	Santana	2002
JJM3692	Chevrolet	Corsa	1999

SELECT IN e NOT IN

- SELECT ...
WHERE <campo> IN
(SELECT <campo> ...)
- SELECT ...
WHERE <campo> NOT IN
(SELECT <campo> ...)

SELECT FROM

- SELECT ...
FROM (SELECT <campo> ...)

SELECT EXISTS e NOT EXISTS

- SELECT ...
WHERE EXISTS
(SELECT <campo> ...)
- SELECT ...
WHERE NOT EXISTS
(SELECT <campo> ...)

SELECT Comparação

- SELECT ...
WHERE <campo> <comparação>
(SELECT <campo> ...)

Exemplo:

```
SELECT Placa FROM Corrida  
WHERE Corrida.DataPedido = (SELECT  
MIN(DataPedido) FROM Corrida);
```


Exercício 7

- Para as tabelas que você montou no exercício 1, escreva um comando SQL que retorne todos os primos por parte de mãe, que você for capaz de inferir a partir da tabela. Considere que você tem como ponto de partida o nome da sua avó.
- Utilize duas estratégias:
 - VIEW
 - SELECT aninhado



Join

Join

- **SELECT ...**
FROM <tabela> JOIN <tabela>
ON <condição> ...
- Tipo clássico de join explicitado
- Também conhecido como INNER JOIN

Natural Join

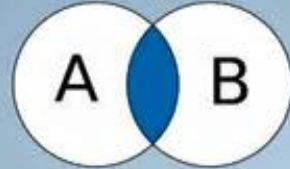
- **SELECT ...**
FOM <tabela> NATURAL JOIN <tabela>
- Condição não especificada
- EQUIJOIN: Verifica igualdade de cada par de atributos com o mesmo nome

Outer Join

- **SELECT ...**
 FROM <tabela> <join> <tabela>
 ON <condição> ...
- **<join>**
 - **LEFT JOIN** - toda tupla à esquerda aparece
 - **RIGHT JOIN** - toda tupla à direita aparece
 - **FULL JOIN** - toda tupla aparece

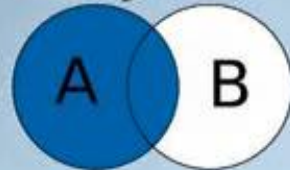
SQL JOINS

INNER JOIN



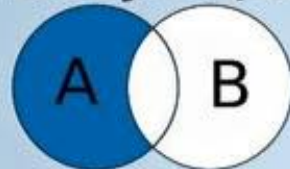
```
SELECT *  
FROM A  
INNER JOIN B ON A.key = B.key
```

LEFT JOIN



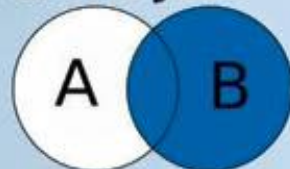
```
SELECT *  
FROM A  
LEFT JOIN B ON A.key = B.key
```

LEFT JOIN (sans l'intersection de B)



```
SELECT *  
FROM A  
LEFT JOIN B ON A.key = B.key  
WHERE B.key IS NULL
```

RIGHT JOIN



```
SELECT *  
FROM A  
RIGHT JOIN B ON A.key = B.key
```

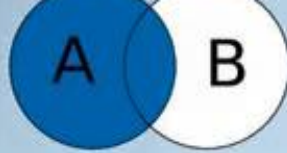
RIGHT JOIN (sans l'intersection de A)



```
SELECT *  
FROM A  
RIGHT JOIN B ON A.key = B.key  
WHERE B.key IS NULL
```

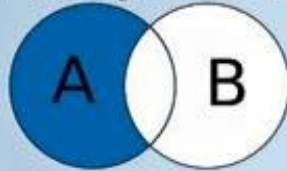
FULL JOIN

```
SELECT *
```



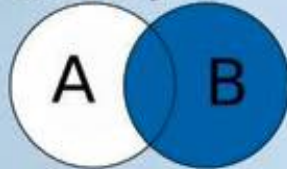
```
SELECT *  
FROM A  
LEFT JOIN B ON A.key = B.key
```

LEFT JOIN (sans l'intersection de B)



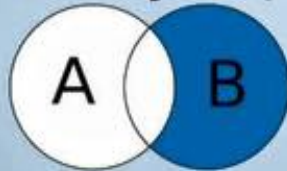
```
SELECT *  
FROM A  
LEFT JOIN B ON A.key = B.key  
WHERE B.key IS NULL
```

RIGHT JOIN



```
SELECT *  
FROM A  
RIGHT JOIN B ON A.key = B.key
```

RIGHT JOIN (sans l'intersection de A)



```
SELECT *  
FROM A  
RIGHT JOIN B ON A.key = B.key  
WHERE B.key IS NULL
```

FULL JOIN



```
SELECT *  
FROM A  
FULL JOIN B ON A.key = B.key
```

FULL JOIN (sans intersection)



```
SELECT *  
FROM A  
FULL JOIN B ON A.key = B.key  
WHERE A.key IS NULL  
OR B.key IS NULL
```

União, Interseção e Diferença

- **SELECT ...
<operador>
SELECT ...**
- **<operador>**
 - **UNION**
 - **INTERSECT**
 - **EXCEPT**

Modificando Dados

DELETE (CUIDADO!!!)

- DELETE FROM <tabela₁>
WHERE <condição>
- executeUpdate(...))

UPDATE

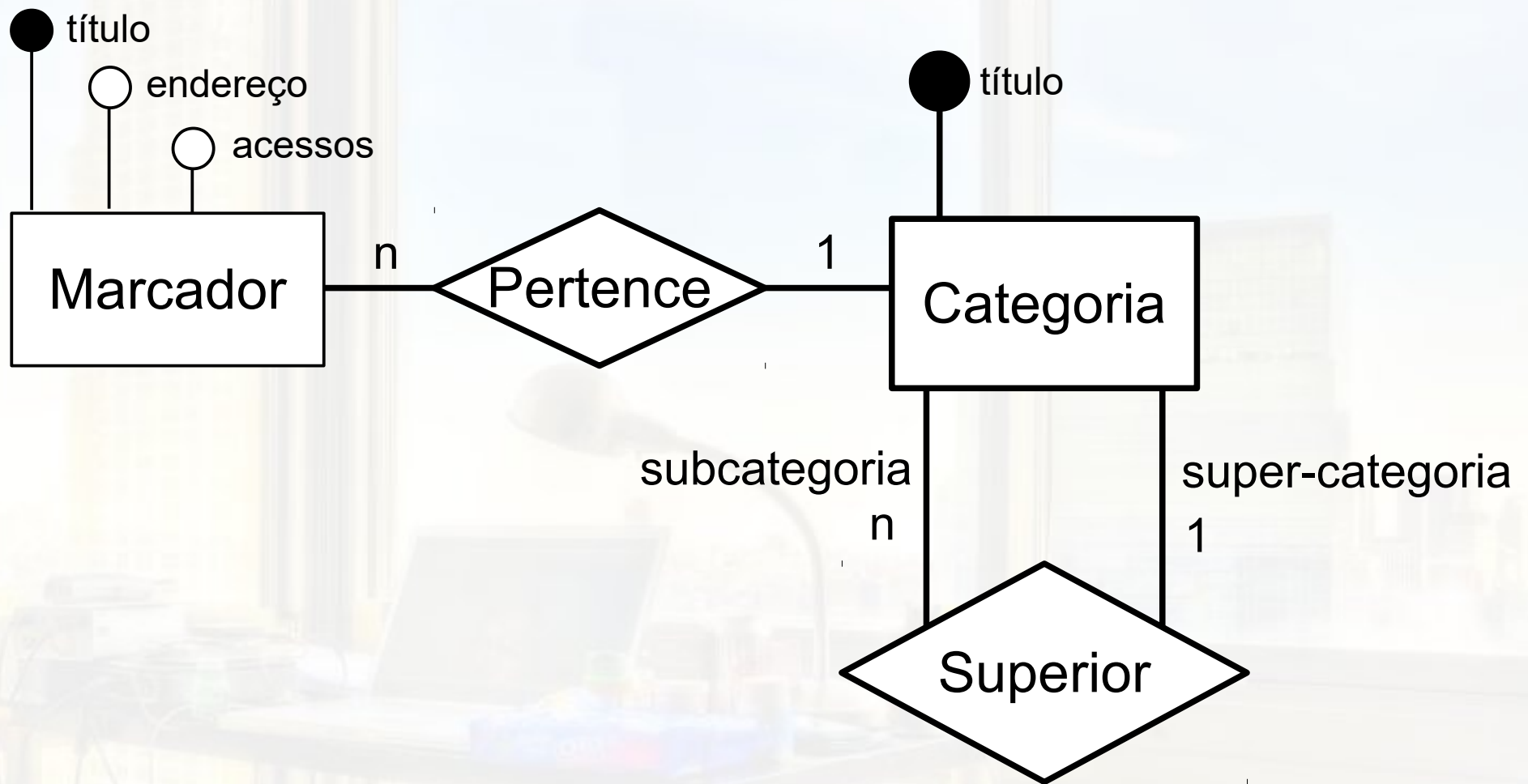
- UPDATE <tabela>
 SET <campo₁>=<valor₁>
 [,..., <campo_n>=<valor_n>]
 WHERE <condição>
- executeUpdate(...)



Aplicando o UPDATE

Marcadores e Categorias

Modelo ER



Marcadores e Categorias

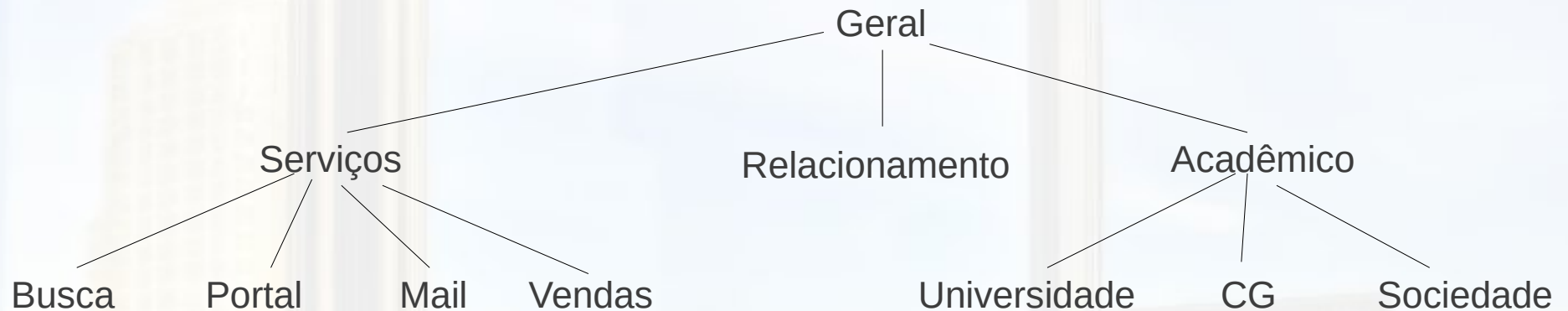
Modelo Relacional

Marcador (Titulo, Endereco, Acessos, Categoria)

Titulo	Endereco	Acessos	Categoria
Terra	http://www.terra.com.br	295	Portal
POVRay	http://www.povray.org	2	CG
SBC	http://www.sbc.org.br	26	Sociedade
Correios	http://www.correios.com.br	45	Serviços
GMail	http://www.gmail.com	296	Mail
Google	http://www.google.com	1590	Busca
Yahoo	http://www.yahoo.com	134	Serviços
Orkut	http://www.orkut.com	45	Serviços
iBahia	http://www.ibahia.com	3	Portal
Submarino	http://www.submarino.com.br	320	Serviços

Tabela Taxonomia

Modelo Relacional



Categoria	Superior
Geral	
Serviços	Geral
Acadêmico	Geral
Relacionamento	Geral
Busca	Serviços
Portal	Serviços
Mail	Serviços
Vendas	Serviços
Universidade	Acadêmico
CG	Acadêmico
Sociedade	Acadêmico

Marcadores e Categorias

Modelo Relacional

Marcador(Título, Acessos, Endereco, Categoria)

- Categoria: chave estrangeira
para Taxonomia

Taxonomia(Categoria, Superior)

Estudo de Caso SQL

- UPDATE Marcadores
SET Categoria = <nova>
WHERE Categoria = <antiga>
- UPDATE Taxonomia
SET Categoria = <nova>
WHERE Categoria = <antiga>
- UPDATE Taxonomia
SET Superior = <nova>
WHERE Superior = <antiga>



SELECT aninhado também pode ser usado em operações de **UPDATE** e **DELETE**

Prepared Statement



Utilizando o PreparedStatement

- `SELECT FROM Marcadores
WHERE Titulo = ?`
- `<comando>.setString(<numero>, <valor>)`

Utilizando o PreparedStatement

- INSERT INTO Marcadores
VALUES (? , ? , ? , ?)
- <comando>.setString(<numero>, <valor>)
- <comando>.setInt(<numero>, <valor>)

Utilizando o PreparedStatement

- UPDATE Marcadores
SET Categoria = ?
WHERE Categoria = ?
- <comando>.setString(<numero>, <valor>)
- <comando>.setInt(<numero>, <valor>)



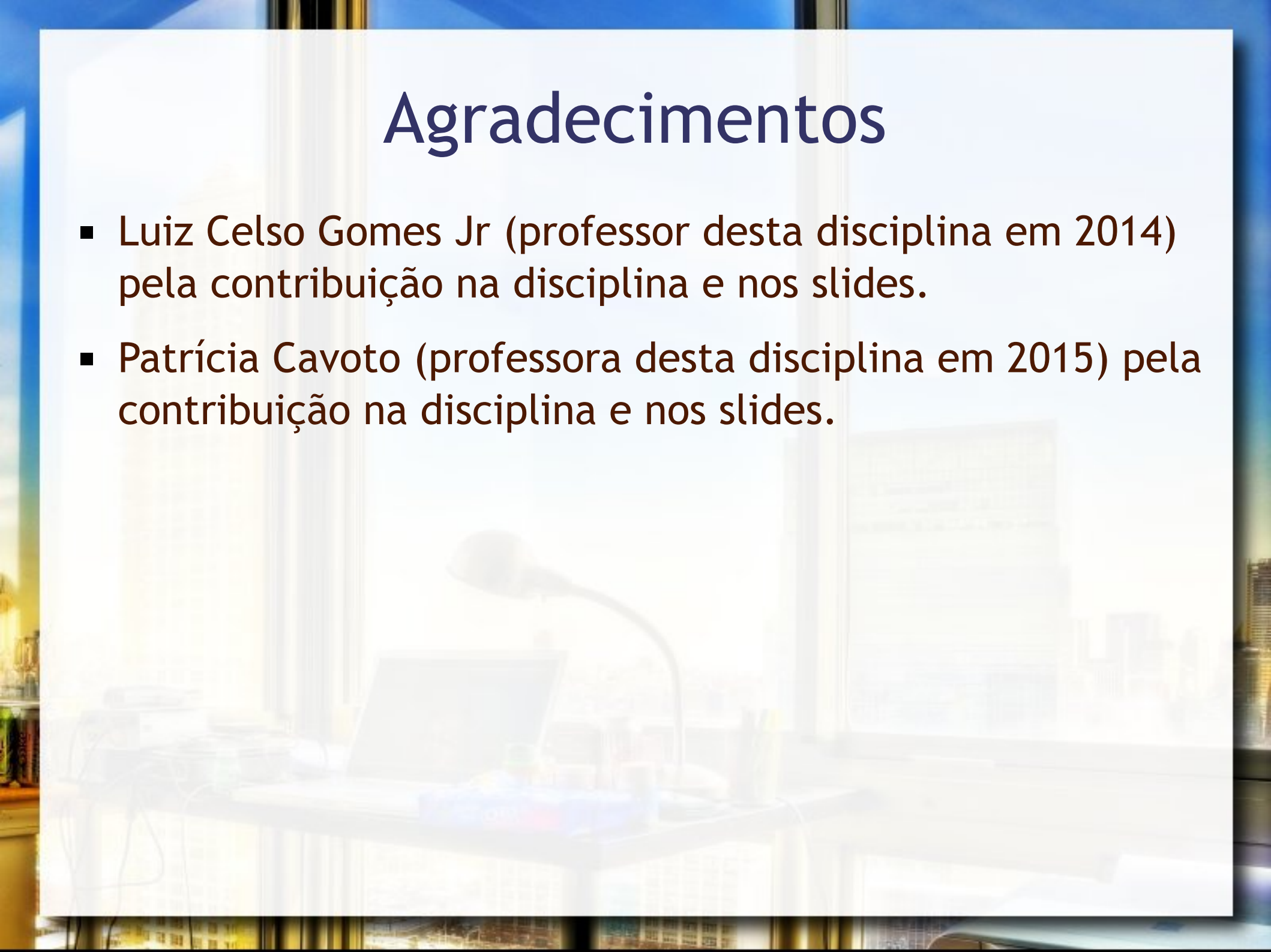
Visões

VIEW

- CREATE VIEW <nome> AS
SELECT ...

Agradecimentos

- Luiz Celso Gomes Jr (professor desta disciplina em 2014) pela contribuição na disciplina e nos slides.
- Patrícia Cavoto (professora desta disciplina em 2015) pela contribuição na disciplina e nos slides.



André Santanchè

<http://www.ic.unicamp.br/~santanche>

Licença

- Estes slides são concedidos sob uma Licença Creative Commons. Sob as seguintes condições: Atribuição, Uso Não-Comercial e Compartilhamento pela mesma Licença, com restrições adicionais:
 - Se você é estudante, você não está autorizado a utilizar estes slides (total ou parcialmente) em uma apresentação na qual você esteja sendo avaliado, a não ser que o professor que está lhe avaliando:
 - lhe peça explicitamente para utilizar estes slides;
 - ou seja informado explicitamente da origem destes slides e concorde com o seu uso.
- Mais detalhes sobre a referida licença Creative Commons veja no link:
<http://creativecommons.org/licenses/by-nc-sa/2.5/br/>