

Aplicações - SQL

Banco de Dados: Teoria e Prática

André Santanchè e Luiz Celso Gomes Jr
Instituto de Computação - UNICAMP
Agosto de 2013



SQL

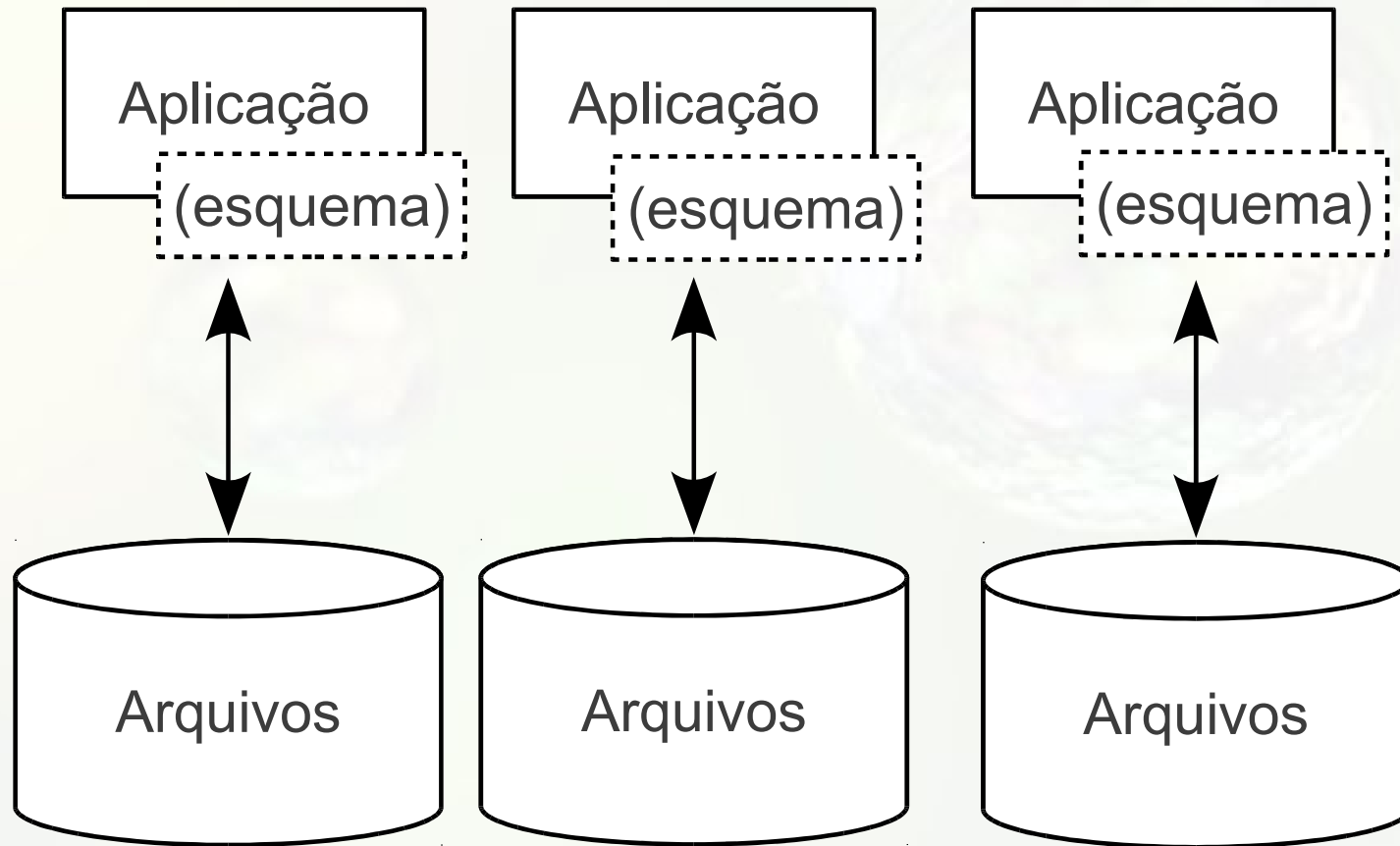
- SQL - Structured Query Language
- Originalmente: SEQUEL - Structured English QUERy Language
- Criada pela IBM Research
 - Interface BD Relacional → SYSTEM R

SQL Padronização

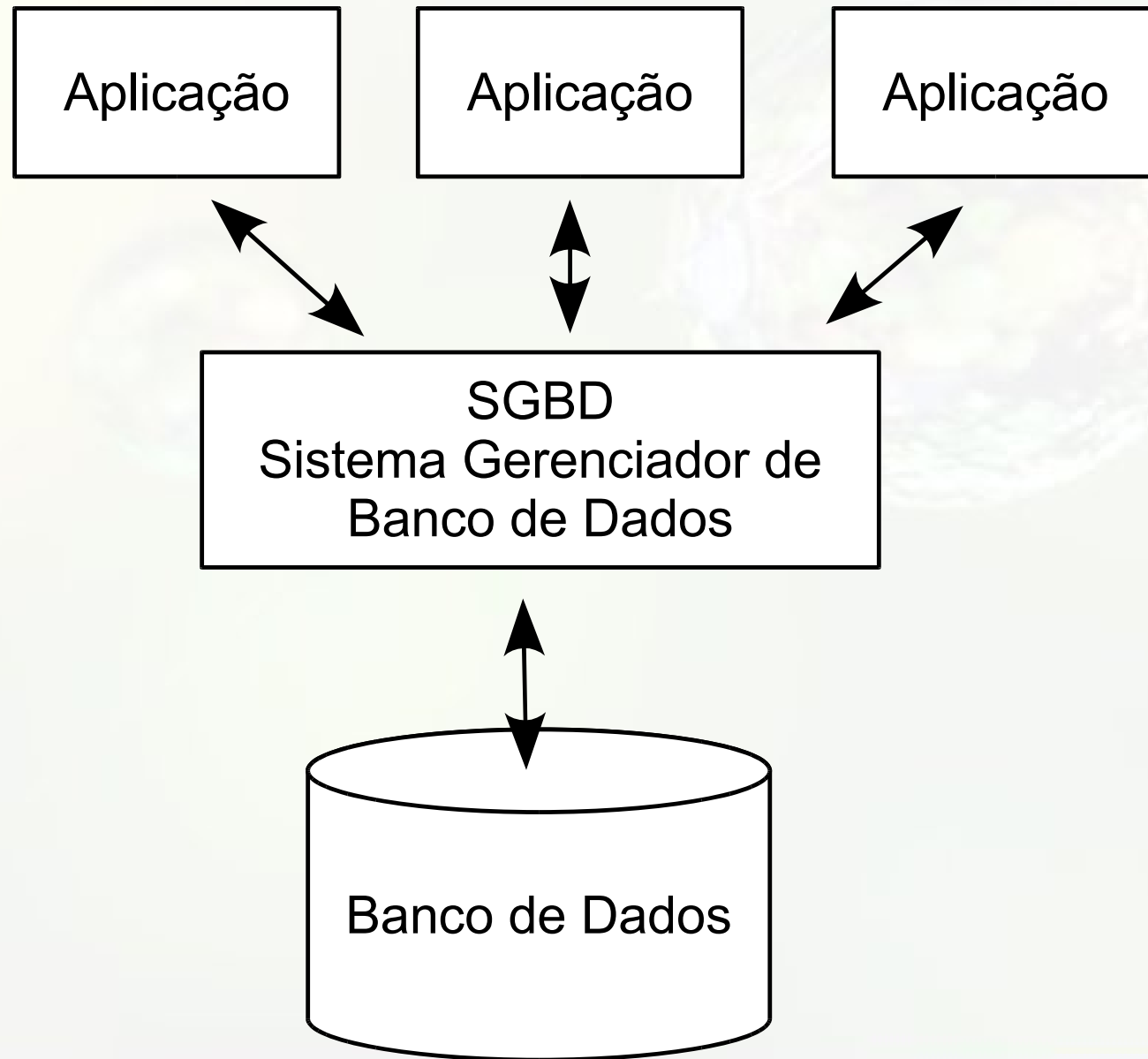
- ANSI + ISO
- SQL-86 ou SQL1
- SQL-92 ou SQL2
- SQL:1999 ou SQL3
- SQL:2003
- SQL:2006

Aplicações e Armazenamento

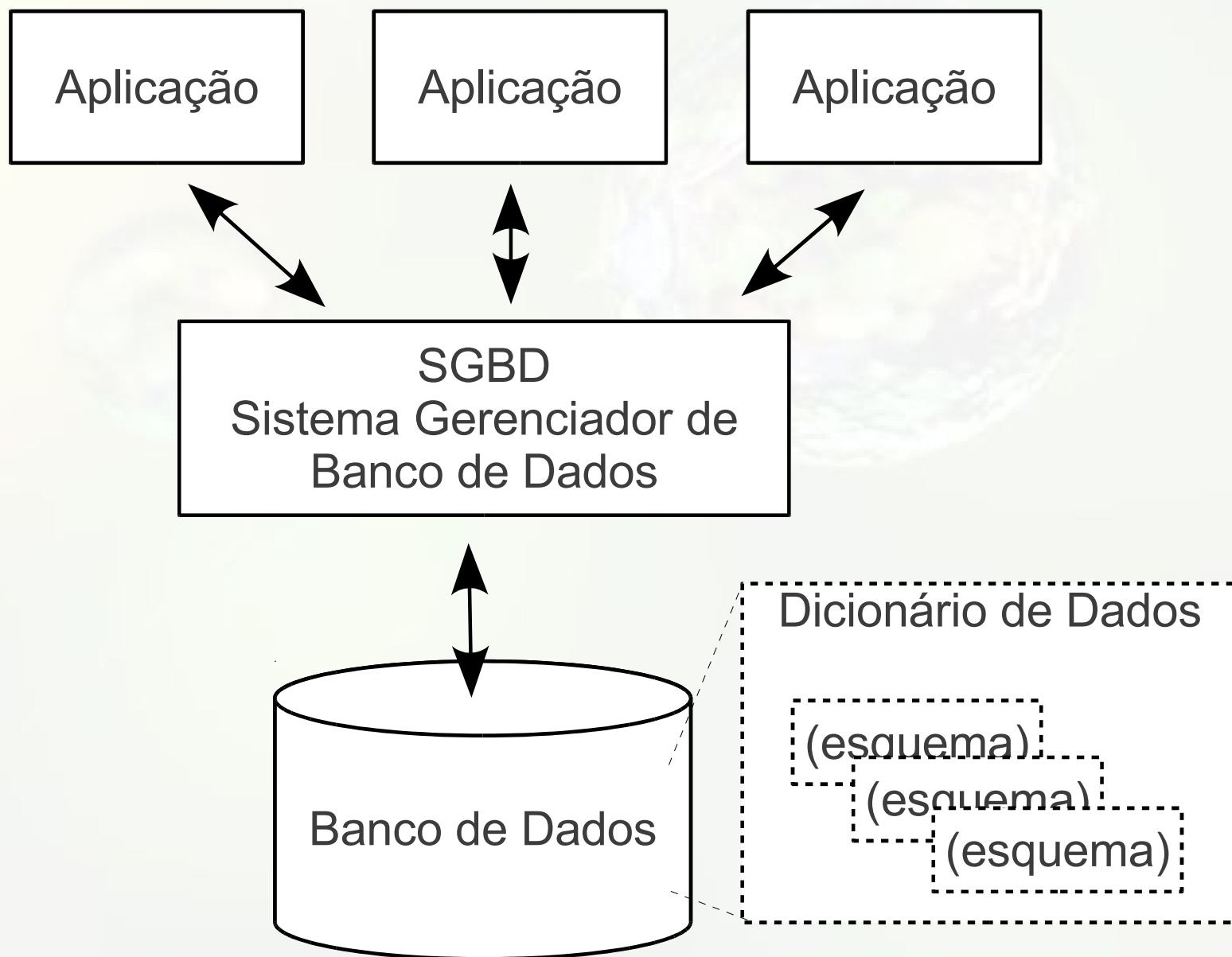
Arquivos



Aplicações e Armazenamento SGBD



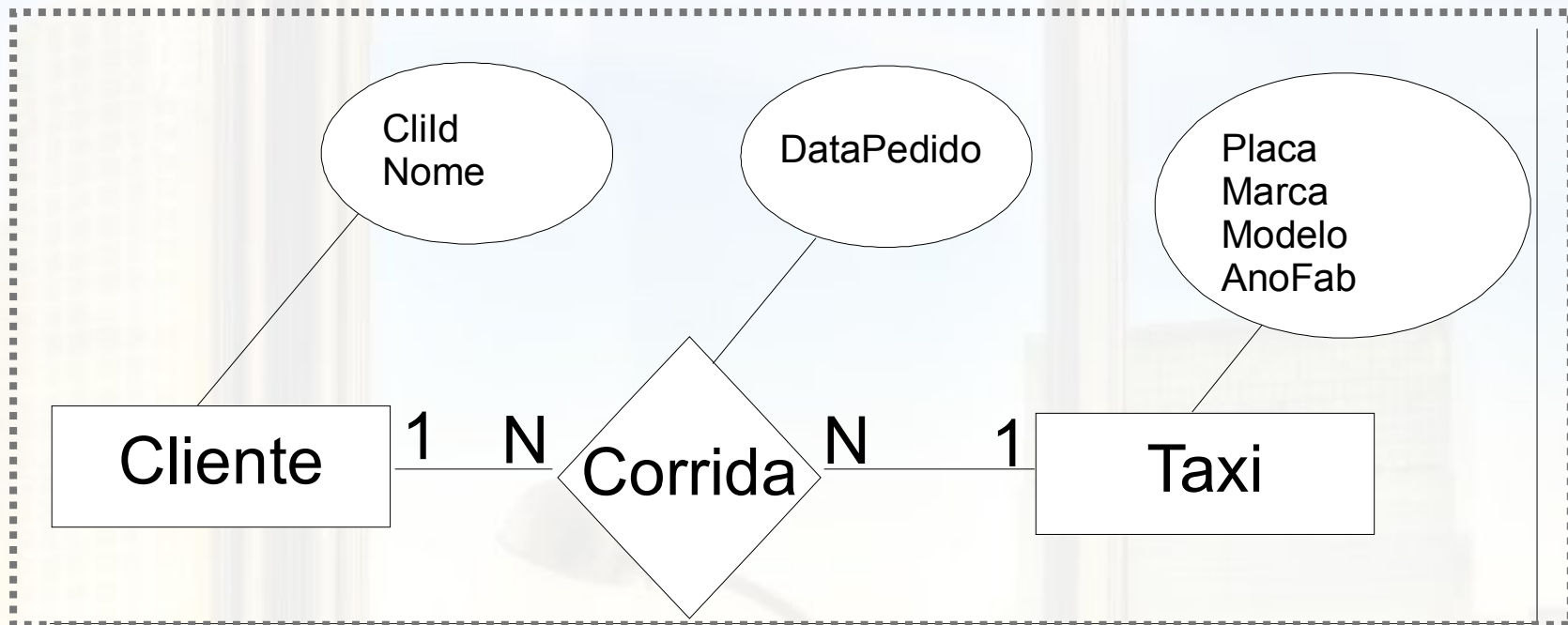
Dicionário de Dados



Caso Prático - Taxis

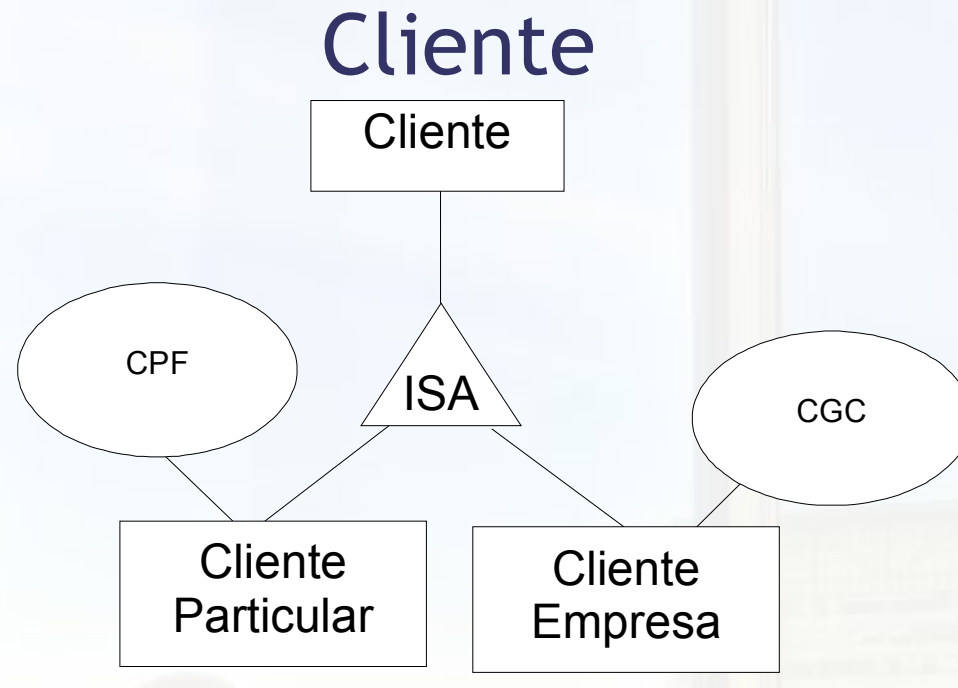
The background of the slide is a blurred photograph of an office workspace. In the foreground, a desk holds a laptop, a desk lamp, and some papers. The desk is positioned in front of a large window that offers a view of a city skyline with several tall buildings under a clear sky. The overall scene is brightly lit, suggesting a daytime setting.

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

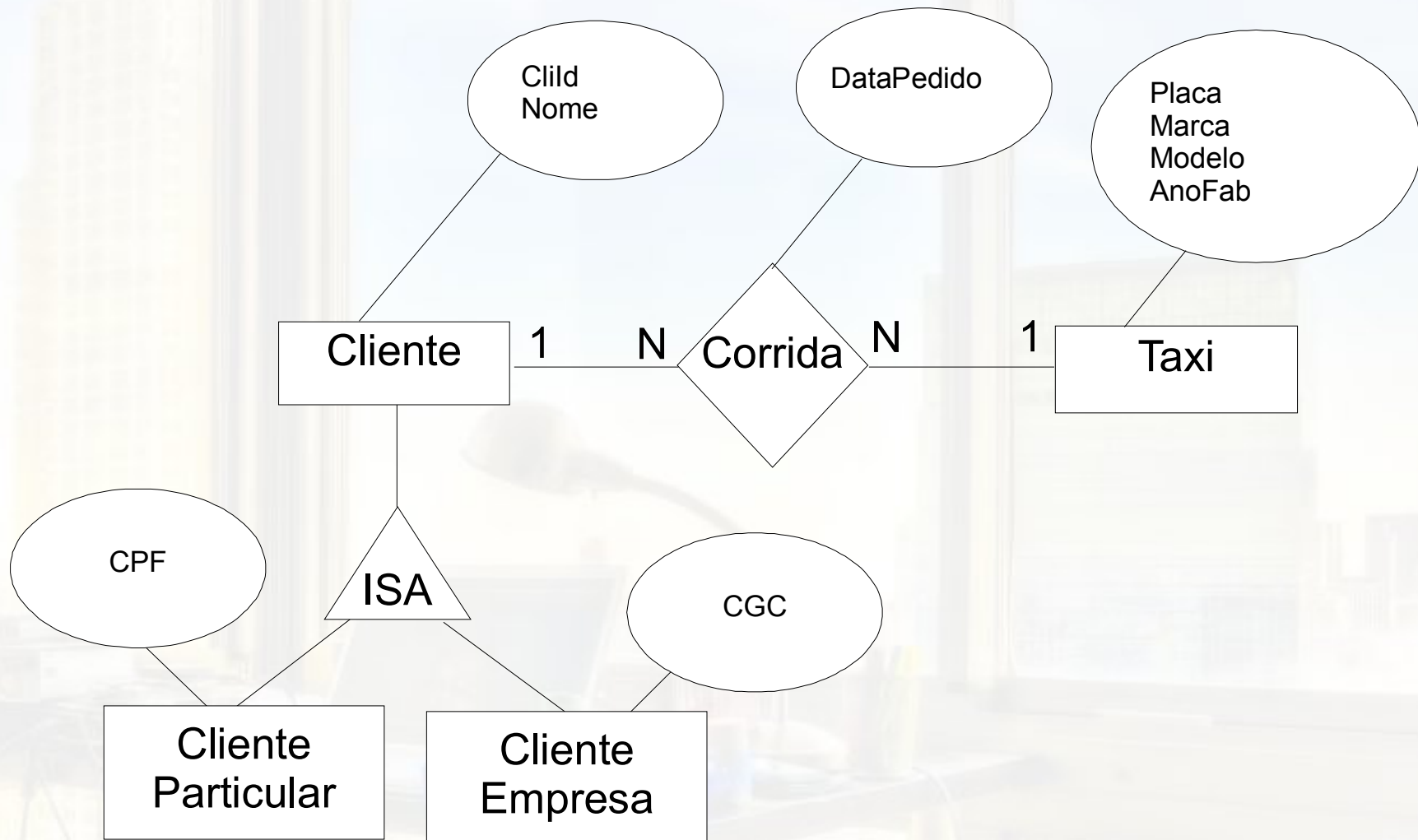
Esquema Conceitual - Exemplo



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>	<u>Marca</u>	<u>Modelo</u>	<u>AnoFab</u>
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(...)

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(...)`

Exercício 3

- Para as 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 LIKE

- **SELECT ...**
FROM <tabela₁>[,..., <tabela_n>]
WHERE <condição/junção>
- % → 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 DISTINCT e ALL

- **SELECT DISTINCT ...**
- **SELECT ALL ...**
- **A cláusula ALL é implícita se não especificada**

SELECT ORDER BY

- SELECT ...
ORDER BY <campo₁> [, ..., <campo_n>]

Exercício 4

- Para as tabelas que você montou no exercício 1, escreva um comando SQL que retorne:
 - nomes de parentes que nasceram no mesmo estado que você
 - retorne seus primos por parte de mãe

DELETE

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

UPDATE

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

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>



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>)

Agrupamento

The background of the slide is a blurred photograph of a modern office workspace. In the foreground, a desk holds a laptop, a desk lamp, and various office supplies. The desk is positioned in front of a large window that provides a panoramic view of a city skyline with several tall buildings under a clear sky. The overall aesthetic is clean and professional.

GROUP BY

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

Funções de Agregação

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

Visões

The image features a semi-transparent white rectangular overlay centered on a blurred background. The background depicts a modern office workspace. In the foreground, a desk holds a laptop, a desk lamp with a curved neck, and various small items. The desk is positioned in front of a large window that offers a panoramic view of a city skyline with several tall buildings under a clear sky. The overall aesthetic is clean and professional, with a focus on the word 'Visões' (Views) in the center.

VIEW

- **CREATE VIEW** <nome> **AS**
SELECT ...

Consultas Aninhadas

The background image is a blurred office scene. In the foreground, a desk holds a laptop, a desk lamp, and some papers. The desk is positioned in front of a large window that looks out onto a city skyline with several tall buildings under a clear sky. The overall image has a soft, out-of-focus quality.

SELECT IN e NOT IN

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

A blurred office desk with a laptop and a desk lamp, viewed through a window with a city skyline in the background. The scene is dimly lit, suggesting an evening or early morning setting. The desk is cluttered with various items, including a laptop, a desk lamp, and some papers. The window provides a view of a city with several tall buildings under a cloudy sky.

Join

Join

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

Natural Join

- **SELECT ...**
FROM <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

União, Interseção e Diferença

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

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/>