

Aplicações - SQL

Banco de Dados: Teoria e Prática

André Santanchè

Instituto de Computação - UNICAMP

Março de 2016



Linguagens de

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

Para manipulação e recuperar

Linguagens de Query (LQ) em BD:

Fundamentação formal

Subsidiaria 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

(Ranat Krishnan, 2003)

Linguagens de Query

- Para manipulação e recuperação de dados
- Linguagens de Query (LQ) em BD:
 - Fundamentação formal
 - Subsidiaria 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

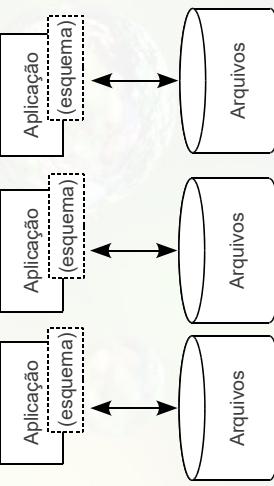
- SQL - Structured Query Language
- Originalmente: SEQUEL - Structured English QUERy Language
- Criada pela IBM Research
 - Interface BD Relacional → SYSTEM R

(Ranat Krishnan, 2003)

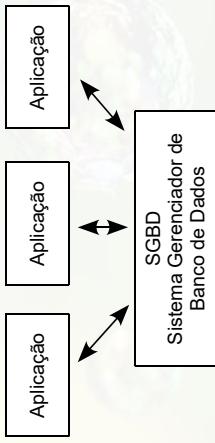
SQL Padronização

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

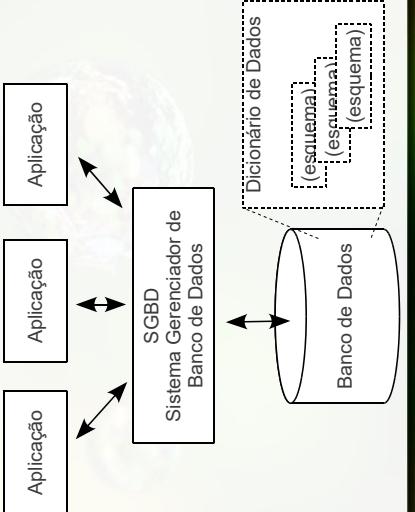
Aplicações e Armazenamento Arquivos



Aplicações e Armazenamento

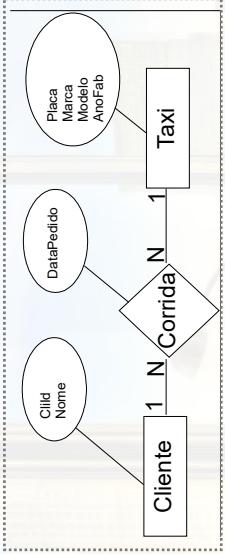


Dicionário de Dados



Esquema Conceitual - Exemplo

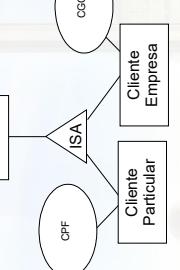
Táxis



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

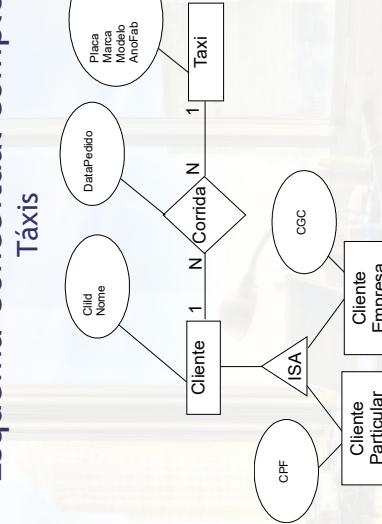
Esquema Conceitual - Exemplo

Cliente



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



Tabelas para exemplo - Táxis

Cliente Particular (CP)

CId	Nome	CGC
1532	Asdrúbal	754.856.965/0001-54
1644	Jepeto	448.754.253-65
1780	Doriana	567.387.387-44
1780	Quinicas	546.373.762-02

Cliente Empresa (CE)

CId	Nome	CPF
1532	Asdrúbal	754.856.965/0001-54
1644	Jepeto	478.652.635/0001-75
1780	Quinicas	554.663.996/0001-87
1982	Zandor	736.952.369/0001-23

Tabelas para exemplo - Táxis

Táxi (TX)

Placa	Marca	Modelo	AnoFab
DAE6334	Ford	Fiesta	1999
DKL4598	Wolkswagen	Gol	2001
DKL7878	Ford	Fiesta	2001
JDM8776	Wolkswagen	Santana	2002
JIM3692	Chevrolet	Corsa	1999

Corrida (R1)

CId	Placa	DataPedido
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 (
  CId VARCHAR(4) NOT NULL,
  Nome VARCHAR(80) NOT NULL,
  CPF VARCHAR(14) NOT NULL,
  PRIMARY KEY (CId)
);
```

- CREATE TABLE <tabela>
 ...
 FOREIGN KEY (<coluna_estr>) [, ..., <coluna_estr>]
 REFERENCES <tabela_ref>([<coluna_ref>[, ..., <coluna_ref>])
 [ON DELETE <ação_ref>]
 [ON UPDATE <ação_ref>]
 [ON ACTION <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 (
    ClId VARCHAR(4) NOT NULL,
    Placa VARCHAR(7) NOT NULL,
    DataPedido DATE NOT NULL,
    PRIMARY KEY (ClId, Placa, DataPedido),
    FOREIGN KEY (ClId)
        REFERENCES Cliente(ClId)
        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 ('11755', 'Dorian', '567.387.387-44');

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

INSERT INTO Corrida
VALUES ('11755', 'DAE6534', '2003-02-15');
```

Comandos INSERT para Taxi

```
INSERT INTO Cliente VALUES ('11532', 'Asdrubal', '448.754.253-65');
INSERT INTO Cliente VALUES ('11755', 'Dorian', '567.387.387-44');
INSERT INTO Cliente VALUES ('11780', 'Quincas', '546.373.762-02');

INSERT INTO Taxi VALUES ('DAE6534', 'Ford', 'Fiesta', 1999, 'MN572345');
INSERT INTO Taxi VALUES ('DKL4981', 'Volkswagen', 'Gol', 2001, 'AU0876543');
INSERT INTO Taxi VALUES ('DKL7878', 'Ford', 'Fiesta', 2001, 'OP02938');
INSERT INTO Taxi VALUES ('JDM8776', 'Volkswagen', 'Santana', 2002, 'QM365923');
INSERT INTO Taxi VALUES ('JJK8592', 'Chevrolet', 'Corsa', 1999, 'WU335577');

INSERT INTO Corrida VALUES ('11755', 'DAE6534', '2003-02-15');
INSERT INTO Corrida VALUES ('11780', 'DAE8776', '2003-02-18');
INSERT INTO Corrida VALUES ('11755', 'DKL7878', '2003-02-16');
INSERT INTO Corrida VALUES ('11780', 'DCL4598', '2003-02-17');
INSERT INTO Corrida VALUES ('11532', 'DKL4598', '2003-02-18');
INSERT INTO Corrida VALUES ('11780', '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`

Placa	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`

Placa	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`

Placa	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`

Placa	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

Placa	Marca	Modelo	AnoFab
DKL4598	Volkswagen	Gol	2001
DKL7878	Ford	Fiesta	2001
JDM8776	Volkswagen	Santana	2002

SELECT Between

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

Placa	Marca	Modelo	AnoFab
DAE6534	Ford	Fiesta	1999
DKL4598	Volkswagen	Gol	2001
DKL7878	Ford	Fiesta	2001
JDM8776	Volkswagen	Santana	2002
JJM3692	Chevrolet	Corsa	1999

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 IN

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

Placa	Marca	Modelo	AnoFab
DAE6534	Ford	Fiesta	1999
DKL4598	Volkswagen	Gol	2001
DKL7878	Ford	Fiesta	2001
JDM8776	Volkswagen	Santana	2002
JJM3692	Chevrolet	Corsa	1999

AS (alias)

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

SELECT LIKE

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

placa	marca
1 DKL4598	Volkswagen
2 JDM8776	Volkswagen

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

SELECT Alias

```
SELECT CI.CId, CI.Nome  
FROM Cliente AS CI;
```

cliid	nome
1	Asdrúbal
2	Doriania
3	Quincas

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 nome FROM Cliente  
ORDER BY nome DESC;
```

nome
Quincas
Doriania
Asdrúbal

SELECT Funções

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

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

SELECT ORDER BY

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

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.Clid, Cliente.Nome,  
Corrida.Clid, Corrida.Placa,  
Corrida.DataPedido  
FROM Cliente, Corrida
```

<u>ClId</u>	<u>Nome</u>	<u>ClId</u>	<u>Placa</u>	<u>Data Pedido</u>
1532	Asdrúbal	1755	DAE6534	15/02/2003
1755	Doriana	1982	JDM8776	18/02/2003
1780	Quincas			

Produto Cartesiano

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

<u>CId</u>	<u>Nome</u>	<u>ClId</u>	<u>Placa</u>	<u>Data Pedido</u>
1532	Asdrúbal	1755	DAE6534	15/02/2003
1755	Doriana	1982	JDM8776	18/02/2003
1780	Quincas			

Prodotti Cartesiano

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

CliId	Nome	CliId	Placa	Data Pedido
1532	Asdrúbal	1755	DAE6534	15/02/2003
1755	Doriane	1982	JDM8776	18/02/2003
1780	Ouijane			

Prodotto Cartesiano

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

CliId	Nome	ClId	Placa	Data Pedido
1532	Asdrúbal	1755	DAE6534	15/02/2003
1755	Dorian	1982	JDM8776	18/02/2003
1780	Quivira			

funcão = join(1)

- SELECT ...
FROM <tabela₁>, <tabela₂>
WHERE <tabela₁>. <attr> = <tabela₂>. <attr>
 - join implícito

Join (1)

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

(CId)	Nome	(CId)	Placa	DataPedido
1532	Asdríbal	1755	DAE6534	15/02/2003
1532	Asdríbal	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

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)	Name	(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

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)			
CliId	Nome	CPF	
1532	Asdrúbal	448.754.253-65	Táxi (TX)
1755	Doriana	567.387.387-44	
1780	Quincas	546.333.762-02	

Corrida (R1)			
CliId	Placa	DataPedido	
1755	DAE6534	15/02/2003	Táxi (TX)
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)			
CliId	Name	CPF	
1532	Asdrúbal	448.754.253-65	Táxi (TX)
1755	Doriana	567.387.387-44	
1780	Quincas	546.333.762-02	

Corrida (R1)			
CliId	Placa	DataPedido	
1755	DAE6534	15/02/2003	Táxi (TX)
1982	JDM8776	18/02/2003	

Desafio

- Quais os modelos de Taxi tomados por cada Cliente?

cliente (C)			
CliId	Nome	CPF	
1532	Asdrúbal	448.754.253-65	Táxi (TX)
1755	Doriana	567.387.387-44	
1780	Quincas	546.333.762-02	

Corrida (R1)			
CliId	Placa	DataPedido	
1755	DAE6534	15/02/2003	Táxi (TX)
1982	JDM8776	18/02/2003	

Desafio

Modelos de Taxi por Cliente

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

Cliente (C)	Taxi (T)
ClId	Nome
1532	Astridbal
1755	Doriana
1780	Quinicas
Placa	Marca
DAE6534	Ford
DKL4598	Volkswagen
DKL7878	Ford
JDM8776	Volkswagen
JJM3692	Chevrolet
Corrida (R1)	
ClId	Placa
1755	DAE6534
1982	IDM8776
	15/02/2003
	18/02/2003

Funções de Agregação

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

Agrupamento

AGREGAÇÃO

```
SELECT AVG(anofab) FROM Taxi;
```

Placa	Marca	Modelo	AnoFab
DAE6534	Ford	Fiesta	1999
DKL4598	Volkswagen	Gol	2001
DKL7878	Ford	Fiesta	2001
JDM8776	Volkswagen	Santana	2002
JJM3692	Chevrolet	Corsa	1999

```
Resultado: avg 2000.4
```

Média de ano de fabricação

GROUP BY

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

Placa	Marca	Modelo	AnoFab
DAE6534	Ford	Fiesta	1999
DKL4598	Volkswagen	Gol	2001
DKL7878	Ford	Fiesta	2001
JDM8776	Volkswagen	Santana	2002
JJM3692	Chevrolet	Corsa	1999

```
Resultado:
```

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

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

Placa	Marca	Modelo	AnoFab
DAE534	Ford	Fiesta	1999
DKL4598	Volkswagen	Gol	2001
DKL7878	Ford	Fiesta	2001
JDM8776	Volkswagen	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 EXISTS e NOT EXISTS

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

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

Exercício 7

- Para a 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

SELECT Comparação

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

Exemplo:

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

Join

Join

- SELECT ...
FROM <tabela> JOIN <tabela>
ON <condição> ...

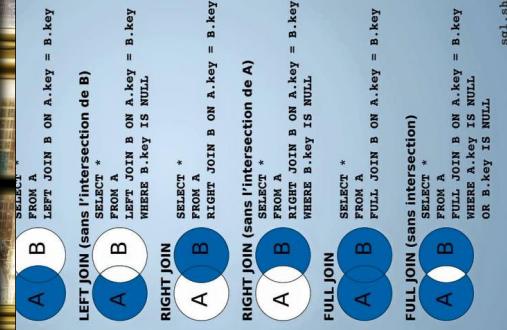
- Tipo clássico de join explicitado
 - Também conhecido como INNER JOIN

Natural Join

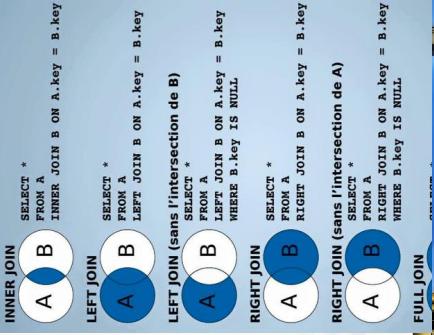
- SELECT ...
FROM <tabela> NATURAL JOIN <tabela>
- Condição não especificada
- EQUIJOIN: Verifica igualdade de cada par de atributos com o mesmo nome

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



SQL JOINS



União, Intersecção e Diferença

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

DELETE (CUIDADO!!!)

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

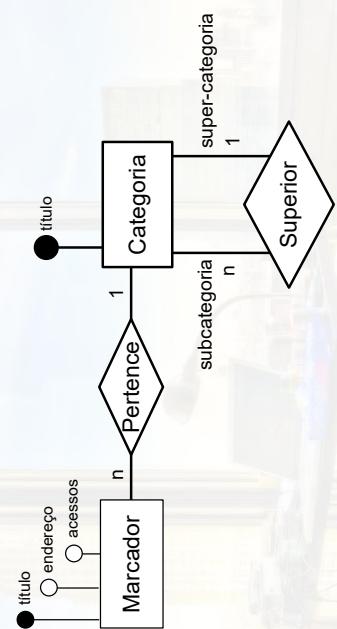
Modificando Dados

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)

Título	Endereço	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
Icânia	http://www.icania.com	3	Portal
Submarino	http://www.submarino.com.br	320	Serviços

Marcadores e Categorias Modelo Relacional

Marcador (Titulo, Acessos, Endereco, Categoria)

- Categoria: Chave estrangeira para Taxonomia

Taxonomia (Categoria, Superior)

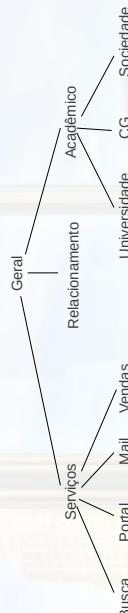
Categoria	Superior
Geral	Geral
Servicos	Geral
Academico	Geral
Relacionamento	Geral
Busca	Geral
Portais	Servicos
Services	Servicos
Mail	Servicos
Vendas	Servicos
Universidade	Academico
CG	Academico
Sociedade	Academico

Tabela Taxonomia Modelo Relacional

Marcador (Titulo, 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

Utilizando o PreparedStatement

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

Prepared Statement

Utilizando o PreparedStatement

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

VIEW

- CREATE VIEW <nome> AS
SELECT ...

Visões

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:
 - Ihe 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/>