centro universitário FEI

AULA 11 – SQL: DQL

PROFA. DRA. LEILA BERGAMASCO

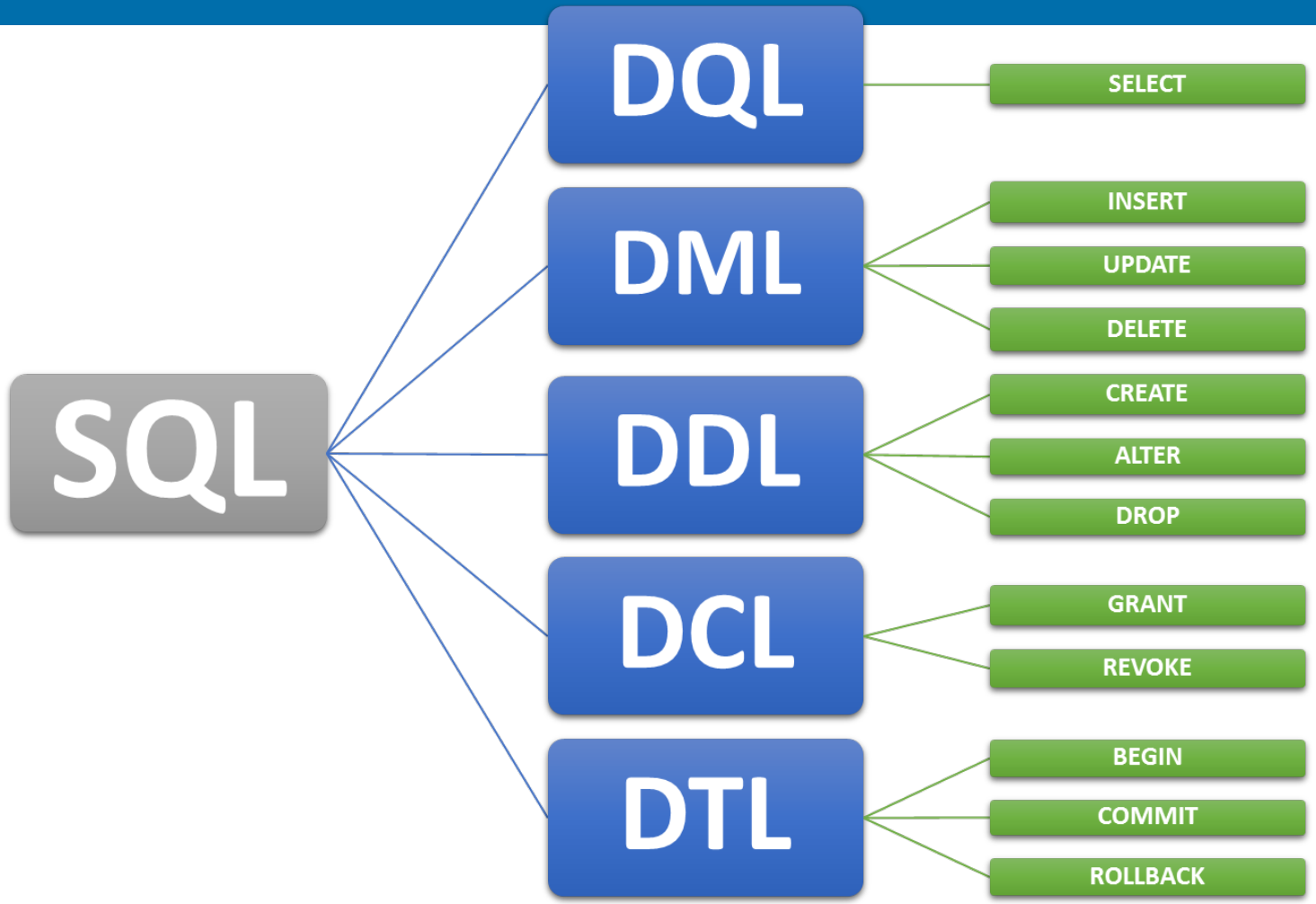
CC5232 – Banco de Dados

centro universitário FEI

NA AULA DE HOJE

- Funções adicionais de DML
  - Como criar FK
  - Conceito de cascata
- DQL
  - Joins

ORGANIZAÇÃO DO SQL



https://www.devmedia.com.br/guia/guia-completo-de-sql/38314

CREATE TABLE X FK

- CREATE TABLE professor (
- cpf varchar,
- nome-prof varchar,
- sobrenome-prof varcahr
- salario numeric DEFAULT 9.99
- primary key (cpf)
-);

1. Criar tabelas e campos seguinto MR
2. Criar chave estrangeira – conceito abstrato para o SGBD

```
CREATE TABLE sala (
    id_sala numeric NOT NULL,
    nro sala varchar
   PRIMARY KEY (id_sala)
);
```

```
CREATE TABLE prof_sala (
id_prof numeric NOT NULL,
id_sala numeric NOT NULL
);
```

CREATE TABLE X FK – MÉTODO 1

```
CREATE TABLE prof_sala (
id_prof numeric NOT NULL,
id_sala numeric NOT NULL
);

ALTER TABLE public.prof_sala     ADD CONSTRAINT sala_fk FOREIGN KEY (id_sala)
REFERENCES public.sala (id_sala)   ON DELETE CASCADE


ALTER TABLE public.prof_sala     ADD CONSTRAINT prof_fk FOREIGN KEY (id_prof)
REFERENCES public.professor (id)   ON DELETE CASCADE
```

CREATE TABLE X FK – MÉTODO 2

```
CREATE TABLE prof_sala (
id_prof integer NOT NULL,
id_sala numeric NOT NULL,
 CONSTRAINT fk_sala
      FOREIGN KEY(id_sala)
     REFERENCES sala(id_sala),
    CONSTRAINT fk_prof
      FOREIGN KEY(id_prof)
     REFERENCES professor(id)

);
```

Sem ON CASCADE

ERROR: update or delete on table "professor" violates foreign key constraint "fk_prof" on table "prof_sala" DETAIL: Key (id)=(3) is still referenced from table "prof_sala". SQL state: 23503

Se eu tentar inserir em prof_sala um id de professor que ainda não foi adicinado a tabela professor?

centro universitário FEI

ON CASCADE

- CASCADE: A opção CASCADE permite excluir ou atualizar os registros relacionados presentes na tabela filha automaticamente, quando um registro da tabela pai for atualizado (ON UPDATE) ou excluído (ON DELETE). É a opção mais comum aplicada.
- RESTRICT: Impede que ocorra a exclusão ou a atualização de um registro da tabela pai, caso ainda hajam registros na tabela filha. Uma exceção de violação de chave estrangeira é retornada. A verificação de integridade referencial é realizada antes de tentar executar a instrução UPDATE ou DELETE
- SET NULL: Esta opção é usada para definir com o valor NULL o campo na tabela filha quando um registro da tabela pai for atualizado ou excluído.
- NO ACTION: Essa opção equivale à opção RESTRICT, porém a verificação de integridade referencial é executada após a tentativa de alterar a tabela. É a opção padrão, aplicada caso nenhuma das opções seja definida na criação da chave estrangeira.
- SET DEFAULT: "Configura Padrão" – Define um valor padrão na coluna na tabela filha, aplicado quando um registro da tabela pai for atualizado ou excluído.

centro universitário FEI

CREATE TABLE X FK – MÉTODO 2

```
CREATE TABLE prof_sala (
id_prof integer NOT NULL,
id_sala numeric,
 CONSTRAINT fk_sala
       FOREIGN KEY(id_sala)
      REFERENCES sala(id_sala),
     CONSTRAINT fk_prof
       FOREIGN KEY(id_prof)
      REFERENCES professor(id)
 ON DELETE SET NULL
 ON UPDATE CASCADE

);
```

DQL

- Define o comando utilizado para que possamos consultar (SELECT) os dados armazenados no banco.
  - SELECT: registros
- Formato de uma consulta genérica

```
SELECT A1, A2, ..., An
FROM r1, r2, ..., rm
WHERE P
```

A_i representa um atributo
R_i representa uma relação
P é um predicado

O resultado da consulta é uma nova relação!

DQL

- A cláusula select lista os atributos desejados no resultado de uma consulta.
- Corresponde à operação de projeção da álgebra relacional
- Exemplo: retorne o nome de todos os professores

```
SELECT prof-nome
FROM professor
WHERE prof-cod = 25
```

$$\Pi_{prof-nome}(\sigma_{prof-cod=25}(professor))$$

- SQL é case-insensitive (prof-nome, Prof-Nome, PROF-NOME)
- Melhores práticas: nome da cláusula em caps, atributos/relação em low-case

DQL

- SQL permite registros duplicados no retorno de suas consultas.
- Para forçar somente valores unicos, utilizer o termo distinct
- Encontre o nome dos professores e elimine os duplicados.

```
SELECT prof-nome
FROM professor
```

prof-nome
Ana
Ana
João
Claudio

```
SELECT distinct(prof-nome)
FROM professor
```

prof-nome
Ana
João
Claudio

prof-nome	prof-sobrenome
Ana	Silva
Ana	Siqueira
João	Duarte
Claudio	Ribeiro

DQL

- Utilizar * significa selecionar todos os atributos da relação professor

```
SELECT *
FROM professor
WHERE prof-cod = 25
```

Qual o similar em Álgebra relacional?

$$\sigma_{prof-cod=25}(professor)$$

centro universitário FEI

DQL

- A cláusula where especifica a condição que a resposta da consulta deve conter
  - Seleção em algebra relacional
- Encontrar todos os nome de professores da disciplina 986

```
SELECT prof-nome
FROM professor
WHERE disc-cod = 986
```

- Pode-se utilizer operadores lógicos
  - Encontrar todos os nomes de professores da disciplina 986 e que tenham salário maior que 2000

```
SELECT prof-nome
FROM professor
WHERE disc-cod = 986 AND salario > 2000
```

RENAME

```
SELECT A1, A2, ..., An
FROM r1 as A, r1 as B
WHERE P
```

$$\Pi_{prof-nome}(\sigma\, ruaRita.rua = professor.rua(\Pi rua(\sigma\, prof - nome = "Rita(\rho\, ruaRita(professor)))$$

- Rename

UTIIZAMOS ALIAS:

```
SELECT prof-nome
FROM professor as a, professor as b
WHERE a.rua = b.rua AND a.prof-nome = “Rita”
```

PRODUTO CARTESIANO

$$\Pi_{prof-nome}(\sigma_{disc-cod=25}\text{^}professor.prof\_cod = professor\_disciplina.prof.cod(professor\ X\ professor\_disciplina))$$

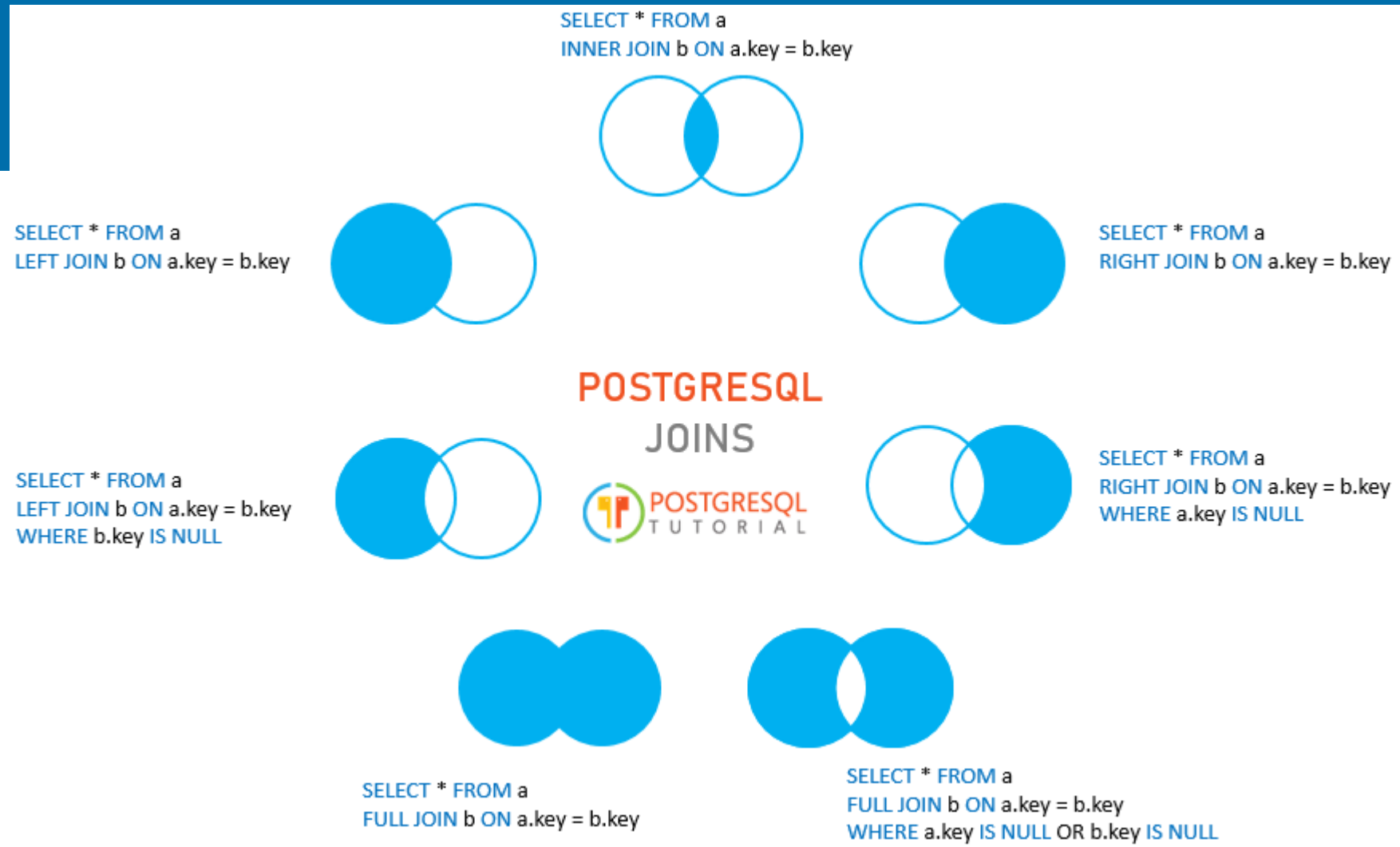
```
SELECT A1, A2, ..., An
FROM r1, r2, ..., rm
WHERE r1.a = r2.a AND r3.a = rm.a … P
```

- Produto Cartesiano

```
SELECT prof-nome
FROM professor, professor_disciplina
WHERE professor.prof_cod = professor_disciplina.prof_cod AND disc_cod = 25
```

USANDO ALIAS:

```
SELECT prof-nome
FROM professor as a, professor_disciplina as b
WHERE a.prof_cod = b.prof_cod AND b.disc_cod = 25
```



https://www.edureka.co/blog/content/ver.1554115042/uploads/2019/03/Joins-in-SQL-SQL-Joins-Edureka.png

JUNÇÃO NATURAL – INNER JOIN

```
SELECT emp.numero-cliente, cliente.cidade
FROM cliente cli
INNER JOIN  emprestimo emp on emp.numero-cliente = cli.numero-cliente
```

```
SELECT numeroC, cidade
FROM cliente cli
INNER JOIN  emprestimo emp on emp.numeroC = cli.numero-cliente
```

numero-cliente	cidade
45	Marília

Recuperar todos os números de cliente que possuem empréstimo e suas cidades onde moram

$$\Pi_{Cliente.numero-cliente,cidade}(Cliente\ |X|\ Emprestimo)$$

Empréstimo (<u>número-empréstimo, número-cliente</u>, valor, agência)
Cliente (<u>número-cliente</u>,nome-cliente,rua,cidade)

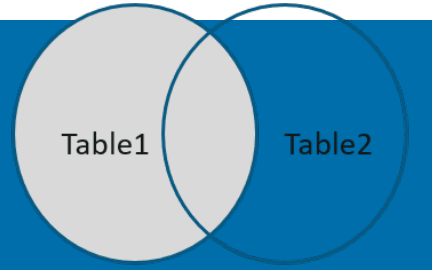
Cliente

numero-cliente	nome	cidade	Nome-agencia
45	Taís	Marília	Centro
34	Flávia	Marília	Jardins
786	Caio	Marília	Casanova

Empréstimo

numero-emp	numeroC	nome-agencia
1	45	Centro
2	12	Centro
3	65	Casanova

JUNÇÃO NATURAL – LEFT JOIN



LEFT JOIN

```
SELECT cli.numero-cliente, cli.cidade
FROM cliente cli
LEFT JOIN  emprestimo emp on emp.numero-cliente = cli.numero-cliente
```

numero-cliente	cidade
45	Marília
34	Marília
786	Santos

Percebam que os clientes 12 e 65 não aparecem.

Recuperar todos os números de cliente que possuem OU NÃO empréstimo e suas cidades onde moram

Empréstimo (<u>número-empréstimo, número-cliente</u>, valor, agência)
Cliente (<u>número-cliente</u>,nome-cliente,rua,cidade)

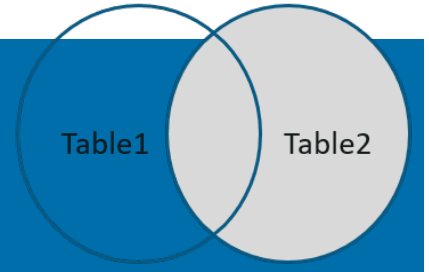
Cliente

numero-cliente	nome	cidade	Nome-agencia
45	Taís	Marília	Centro
34	Flávia	Marília	Jardins
786	Caio	Santos	Casanova

Empréstimo

numero-emp	numero-cliente	nome-agencia
1	45	Centro
2	12	Centro
3	65	Casanova

JUNÇÃO NATURAL – RIGHT JOIN



RIGHT JOIN

edureka!

```
SELECT cli.numero-cliente, cli.cidade
FROM cliente cli
RIGHT JOIN  emprestimo emp on emp.numero-cliente = cli.numero-cliente
```

numero-cliente	cidade
45	Marília
Null	null
null	null

Recuperar todos os números de cliente que possuem empréstimo e suas cidades onde moram, se houver

Empréstimo (<u>número-empréstimo, número-cliente</u>, valor, agência)
Cliente (<u>número-cliente</u>,nome-cliente,rua,cidade)

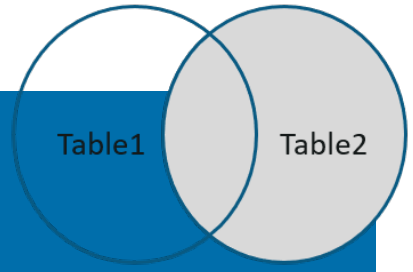
Cliente

numero-cliente	nome	cidade	Nome-agencia
45	Taís	Marília	Centro
34	Flávia	Marília	Jardins
786	Caio	Santos	Casanova

Empréstimo

numero-emp	numero-cliente	nome-agencia
1	45	Centro
2	12	Centro
3	65	Casanova

JUNÇÃO NATURAL – RIGHT JOIN



edureka!

```
SELECT emp.numero-cliente, cli.cidade
FROM cliente cli
RIGHT JOIN  emprestimo emp on emp.numero-cliente = cli.numero-cliente
```

numero-cliente	cidade
45	Marília
12	null
65	null

Recuperar todos os números de cliente que possuem OU NÃO empréstimo e suas cidades onde moram

Empréstimo (<u>número-empréstimo, número-cliente</u>, valor, agência)
Cliente (<u>número-cliente</u>,nome-cliente,rua,cidade)

Cliente

numero-cliente	nome	cidade	Nome-agencia
45	Taís	Marília	Centro
34	Flávia	Marília	Jardins
786	Caio	Santos	Casanova

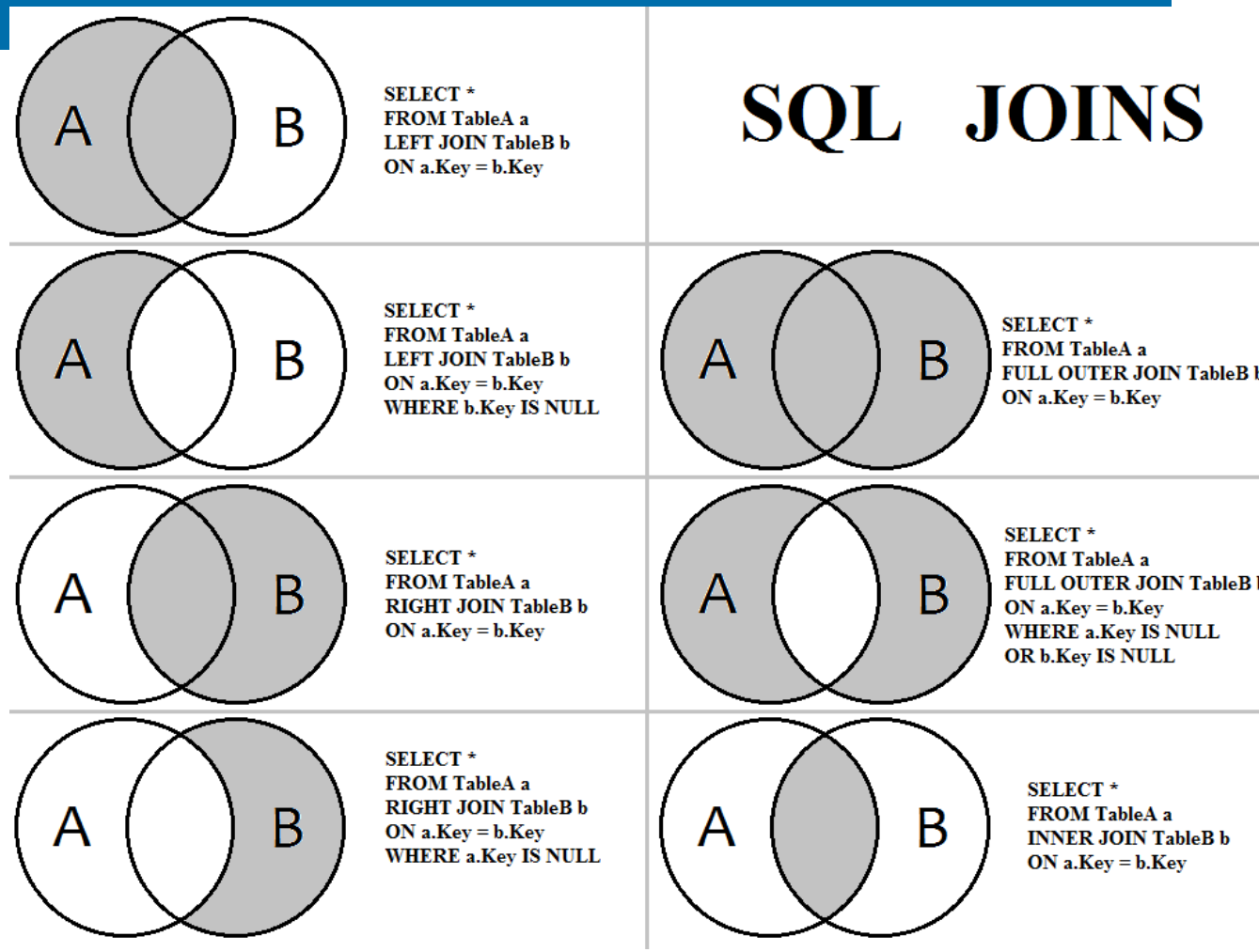
Empréstimo

numero-emp	numero-cliente	nome-agencia
1	45	Centro
2	12	Centro
3	65	Casanova

BONUS – OUTER JOIN

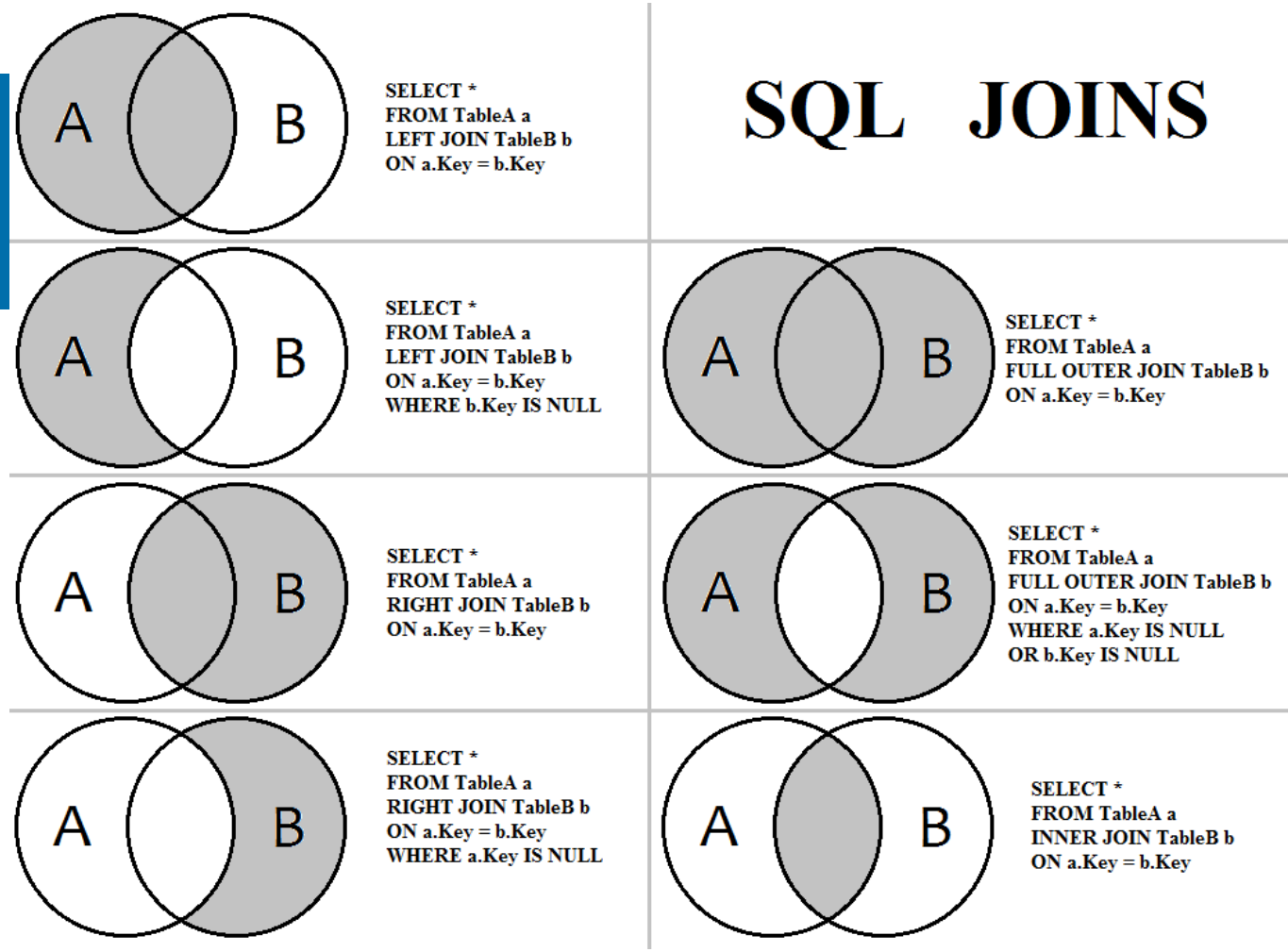
```
SELECT numero_cli, numero_cliente,cidade
FROM cliente cli
FULL OUTER JOIN  emprestimo emp on
emp.numero_cli = cli.numero_cliente
```

	numero_cli integer	numero_cliente integer	cidade character varying
1	45	45	Marilia
2	12	[null]	[null]
3	65	[null]	[null]
4	[null]	786	Marilia
5	[null]	34	Marilia



OUTER JOIN EXCLUSIVO

	numero_cli integer	numero_cliente integer	cidade character varying
1	12	[null]	[null]
2	65	[null]	[null]
3	[null]	786	Marilia
4	[null]	34	Marilia




```
SELECT numero_cli, numero_cliente,cidade
FROM cliente cli
FULL OUTER JOIN  emprestimo emp on emp.numero_cli = cli.numero_cliente
where emp.numero_cli is null or cli.numero_cliente is null
```

EXERCÍCIOS

Considere os seguintes esquemas de relação:

- Professor (prof-numero, prof-nome, prof-rua, prof-cidade, prof-telefone)
- Aluno (alu-numero, alu-nome, alu-rua, alu-cidade)
- Disciplina (disc-codigo,disc-nome,disc-quant-aulas-semana)
- Matricula(alu-numero,disc-codigo, ano, semestre, nota, frequencia)
- ProfessorDisciplina (prof-numero, disc-codigo)

- Escreva a consulta SQL considerando os comandos DQL e DML
  - Realize as seguintes inserções, se já não tiver na sua base:
    - 3 alunos, tendo pelo menos um: alu-cidade=Santo André
    - 3 disciplinas sendo uma com disc-codigo-=926
    - 3 Professores
    - 4 Matriculas, sendo uma na disciplina 926 e uma com o alu-numero que mora em Sto Andre.
    - 4 registros em ProfessorDisciplina (leve em consideração que somente professores cadastrados poderão ser inseridos nessa tabela)
  - Selecione o nome de todos alunos que estão matriculados na disciplina 926
  - Selecione o nome de todos os alunos que estão matriculados ou não na disciplina 926 e que moram em Santo André
  - Selecione os nomes dos professores e o código das disciplinas que eles lecionam. Apresentar null caso eles não estejam lecionando nenhuma (dica: left ou right join)

centro universitário FEI

OBRIGADO E ATÉ A PRÓXIMA AULA!