Universidade do Minho

Ano Letivo: 2023/24
Turnos: PL3/PL7

Bases de Dados

PL09 – Exploração Simples e Avançada

Docente: Diana Ferreira
Email: diana.ferreira@algoritmi.uminho.pt

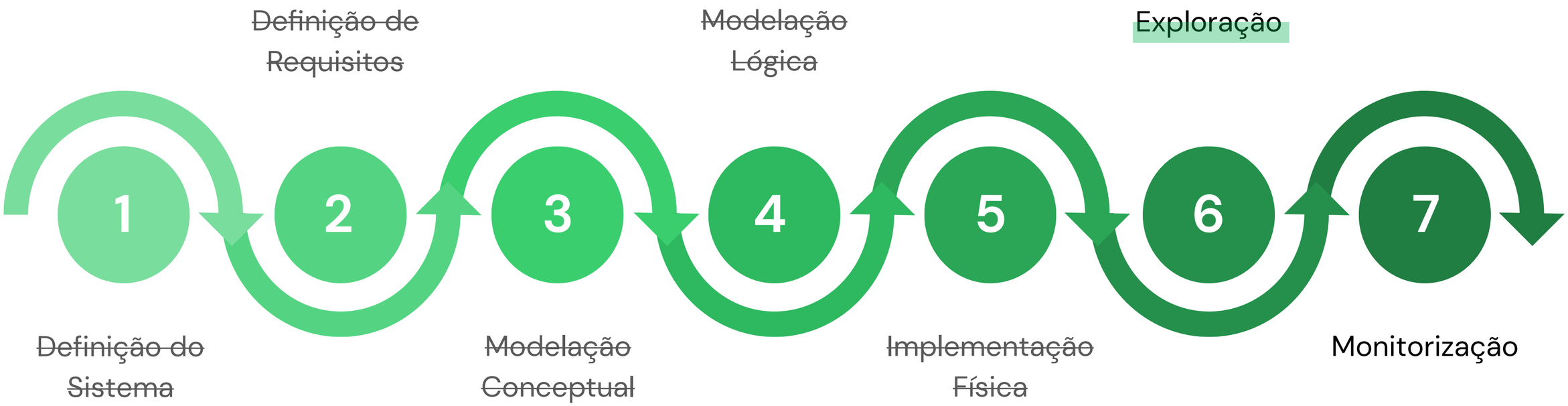
Horário de Atendimento:
5ª feira 16h-17h

Sumário

Bibliografia:

- Connolly, T., Begg, C., Database Systems, A Practical Approach to Design, Implementation, and Management , Addison-Wesley, 4a Edição, 2004. **(Chapter 6 e 7)**
- Belo, O., "Bases de Dados Relacionais: Implementação com MySQL", FCA – Editora de Informática, 376p, Set 2021. ISBN: 978-972-722-921-5. **(Capítulo 4 e 5)**

Ciclo de vida de um SBD



FASE 6: Exploração

➡ Data Manipulation Language (DML)

Existem 4 instruções básicas para a manipulação de dados:

- **INSERT** → para inserir dados na BD;

```
INSERT INTO <nome_tabela> (<c1>,<c2>,...) VALUES (<v1>,<v2>,...);
INSERT INTO <nome_tabela> (<c1>,<c2>,...)
VALUES
  (<v11>,<v12>,...),
  ...
  (<vnn>,<vn2>,...);
```

- **DELETE** → para remover dados da BD;

```
DELETE FROM <nome_tabela> WHERE <condição>;
```

- **SELECT** → para consultar dados da BD;

```
SELECT [DISTINCT] {* | <nome_c1>, ...}
FROM <nome_tabela>,...
[WHERE <condição>]
[ORDER BY <c1> [ASC | DESC], ...];
```

- **UPDATE** → para atualizar dados da BD;

```
UPDATE <nome_tabela>
SET
   <c1> = <v1>,
   <c2> = <v2>,
   ...
[WHERE <condição>];
```

FASE 6: Exploração

Operadores

= → igual

<> → diferente

!= → diferente

< → inferior a

<= → igual ou inferior a

> → superior a

>= → igual ou superior a

AND → para condições conjuntas

OR → para condições disjuntas

NOT → para negação de condições

IS NULL → para verificar o preenchimento de uma coluna

IN → para determinar se um valor especificado corresponde a qualquer valor de uma lista de valores

BETWEEN → para determinar se um valor está contido num intervalo de valores

LIKE → para consultar dados com base num padrão especificado (% → qualquer sequência de zero ou mais caracteres; _ → caracter único).

LIMIT → para limitar o número de instâncias retornadas.

FASE 6: Exploração

➡ Funções de Strings (exemplos)

CONCAT - Concatena duas ou mais strings numa só;

INSTR - Retorna a posição da primeira ocorrência de uma substring numa string;

LENGTH – Devolve o comprimento de uma string em bytes e em caracteres;

LEFT - Retorna um número específico de caracteres mais à esquerda de uma string;

LOWER – Converte uma string para minúsculas;

LTRIM - Recebe um argumento de string e retorna uma nova string com todos os caracteres de espaço à esquerda removidos;

REPLACE – Procura e substitui o valor de uma substring numa string;

RIGHT - Devolve um número específico de caracteres mais à direita de uma string;

RTRIM - Recebe um argumento de string e retorna uma nova string com todos os caracteres de espaço à direita removidos;

FASE 6: Exploração

Funções de Datas

CURDATE – Retorna a data atual;

NOW/ SYSDATE – Retorna a data e hora atuais;

DAY - Obtém o dia do mês de um DATE/DATETIME;

DAYOFWEEK – Obtém o índice do dia da semana de um DATE/DATETIME;

MONTH - Retorna um inteiro que representa o mês de um DATE/DATETIME;

WEEK - Retorna um número de semana de um DATE/DATETIME;

WEEKDAY - Retorna um índice de dia da semana para um DATE/DATETIME;

YEAR - Retorna o ano de um DATE/DATETIME;

HOUR – Retorna a hora de um DATETIME/TIME;

MINUTE – Retorna os minutos de um DATETIME/TIME;

SECOND – Retorna os segundos de um DATETIME/TIME;

FASE 6: Exploração

Funções de Datas

DATEDIFF – Calcula o número de dias entre dois valores DATE/DATETIME;

DATE_ADD – Adiciona um valor de tempo a um valor DATE/DATETIME;

DATE_SUB – Subtrai um valor de tempo a um valor DATE/DATETIME;

DATE_FORMAT – Formata um valor de data com base em um formato de data especificado;

STR_TO_DATE – Converte uma string num valor de data e hora com base num formato especificado;

TIMEDIFF – Calcula a diferença entre dois valores DATETIME/TIME;

TIMESTAMPDIFF – Calcula a diferença entre dois valores DATE/DATETIME.

FASE 6: Exploração

➡ Funções de Datas

EXEMPLOS:

Qual é a data da última prescrição emitida?

```
SELECT MAX(DATE(data_prescricao))  FROM prescricoes;
```

Quantos dias passaram desde que '2022-03-22'?

```
SELECT DATEDIFF(NOW(), ' 2022-03-22');
```

Liste as consultas que ocorreram no dia a seguir ao '2020-01-01'.

```
SELECT * FROM consultas WHERE DATE(hora_ini)=ADDDATE('2020-01-01', INTERVAL 1 DAY);
```

Liste as prescricoes emitidas à mais de um ano.

```
SELECT * FROM prescricoes WHERE DATE(data_prescricao)<ADDDATE(CURDATE(), INTERVAL -365 DAY);
```

FASE 6: Exploração

Funções Numéricas (exemplos)

ABS() Retorna o valor absoluto de um número

DIV () Realiza a divisão entre dois números e retorna o inteiro quociente

MOD() Retorna o resto de um número dividido por outro

ROUND() Arredonda para um número de casas decimais especificado

DEGREES(n) Converte radianos para graus de um argumento n

EXP(n) Retorna e elevado à potência do número específicado

FASE 6: Exploração

Expressões Regulares

As expressões regulares diferenciam-se do operador LIKE por permitirem construir padrões mais flexíveis. No entanto, o tempo de consulta pode aumentar caso se usem padrões complexos. Estas expressões usam os operadores **RLIKE** ou **REGEXP**. Alguns metacaracteres comumente usados numa expressão regular:

^	Corresponde à posição no início da string pesquisada;
$	Corresponde à posição no final da string pesquisada;
.	Corresponde a qualquer caracter;
[...]	Corresponde a qualquer caracter específicado dentro dos parêntesis rectos;
[^...]	corresponde a qualquer caractere não especificado dentro dos parêntesis
p1|p2	corresponde a qualquer um dos padrões p1 ou p2
{n}	corresponde a n número de instâncias do caractere anterior
{m,n}	corresponde de m a n número de instâncias do caractere anterior

FASE 6: Exploração

➡ Expressões Regulares

EXEMPLOS:

- **Retorna os fármacos cujo nome comece com a letra ‘o’ ou ‘a’:**

SELECT nome FROM farmacos WHERE nome REGEXP '^[oa]';

SELECT nome FROM farmacos WHERE nome REGEXP '^o|^a';

- **retorna as especialidades cujo nome não termina com as letras ‘gia’**

SELECT des_especialidade FROM especialidades WHERE des_especialidade NOT REGEXP ‘gia$’;

- **retorna todos os fármacos que contêm os caracteres ‘ar’**

SELECT nome FROM farmacos WHERE nome REGEXP 'ar';

FASE 6: Exploração

Expressões Regulares

- retorna as especialidades que contêm extamente 10 caracteres:

SELECT des_especialidade FROM especialidades WHERE des_especialidade REGEXP '^.{10}$';

SELECT des_especialidade FROM especialidades WHERE des_especialidade REGEXP '^..........$';

- retorna as especialidades que contêm entre 5 a 10 caracteres:

SELECT des_especialidade FROM especialidades WHERE des_especialidade REGEXP '^.{5,10}$';

- retorna os fármacos que contêm uma letra entre 'a' e 'c', seguidas por qualquer caracter, seguidas pela letra 'a'.

SELECT nome FROM farmacos WHERE nome REGEXP '[a-c].[a]';

- retorna todos os fármacos que comecem com vogal ou terminem em 'ol'

SELECT nome FROM farmacos WHERE nome REGEXP '^[aeiou]|ol$';

FASE 6: Exploração

➡ ORDER BY

A cláusula ORDER BY permite que as linhas sejam apresentadas por ordem ascendente (ASC) ou decrescente (DESC) de qualquer coluna ou combinação de colunas. A cláusula ORDER BY deve ser sempre a última cláusula da instrução SELECT.

EXEMPLOS:

Liste o nome dos administrativos por ordem crescente.

SELECT nome from funcionarios f, administrativos a WHERE f.nr_mec=a.nr_mec ORDER BY nome ASC;

Liste os procedimentos disponíveis no hospital, do maior custo para o menor.

SELECT * FROM procedimentos ORDER BY preco DESC;

FASE 6: Exploração

➡ GROUP BY

EXEMPLOS:

A) Uso em alternativa ao SELECT DISTINCT(<nome coluna>)

- SELECT DISTINCT localidade FROM pacientes;
- SELECT localidade FROM pacientes GROUP BY localidade;

B) Uso com funções de agregação (AVG, COUNT, SUM, MAX, MIN, etc.)

O valor máximo de consulta cobrado por especialidade.

SELECT e.des_especialidade, MAX(c.custo_final) as preco_max FROM consultas c, especialidades e, medicos m WHERE m.nr_mec = c.nr_mec_medico AND m.cod_especialidade = e.cod_especialidade GROUP BY e.des_especialidade;

O valor médio de consulta por ano.

SELECT YEAR(c.dta_ini) as ano, AVG(c.custo_final) as preco_medio FROM consultas c GROUP BY YEAR(c.dta_ini);

FASE 6: Exploração

GROUP BY

EXEMPLOS:

D) Uso com a cláusula HAVING

Para filtrar os valores retornados pela cláusula GROUP BY, usa-se uma cláusula HAVING.

O nº total de consultas por ano após 2019.

SELECT YEAR(dta_ini) as ano, COUNT(*) as total_consultas FROM consultas GROUP BY YEAR(dta_ini) HAVING ano > '2019';

FASE 6: Exploração

SUBQUERIES

Uma instrução SELECT pode ser usada dentro de outra instrução SELECT, é chamada de SELECT interna (ou subselect) e tem de estar entre parêntesis curvos. Por sua vez, um subselect pode ser usado dentro de outro subselect.

A) subselect com operadores de comparação

Qual é o procedimento mais caro?

SELECT * FROM procedimentos WHERE preco = (SELECT MAX(preco) FROM procedimentos);

B) subselect com operadores IN e NOT IN

Listar os administrativos que ainda não faturaram uma consulta.

SELECT * FROM administrativos WHERE nr_mec NOT IN (SELECT nr_mec_secretaria FROM consultas WHERE nr_mec_secretaria IS NOT NULL)

FASE 6: Exploração

SUBQUERIES

C) subselect na cláusula FROM – o conjunto de resultados retornado de um subselect é usado como uma tabela temporária.

Listar o número máximo e o número mínimo de medicamentos receitados.

SELECT max(total), min(total) FROM (SELECT id_med, COUNT(*) as total FROM prescricoes GROUP BY id_med) AS sub;

D) subselect com operadores EXISTS e NOT EXISTS

Listar os médicos que deram consultas.

SELECT * FROM medicos m WHERE EXISTS (SELECT * FROM consultas c WHERE c.nr_mec_medico = m.nr_mec);

FASE 6: Exploração

SUBQUERIES

E) subselect com operadores ANY e ALL: É usado para efetuar uma comparação entre o valor de uma coluna e uma range de valores. Ambos retornam um valor booleano como resultado. O ANY retorna TRUE se QUALQUER um dos valores da subconsulta atender à condição. O ALL retorna TRUE se TODOS os valores da subconsulta atenderem à condição .

Listar os médicos que deram consultas.

```
SELECT * FROM medicos WHERE nr_mec =  ANY(SELECT nr_mec_medico FROM consultas);
```

FASE 6: Exploração

SUBQUERIES

```
SELECT * FROM R
WHERE r1 IN
(SELECT r1 FROM S);
```

```
SELECT * FROM R
WHERE EXISTS
(SELECT * FROM S
WHERE R.r1 = S.r1);
```

```
SELECT * FROM R
WHERE r1 = ANY
(SELECT r1 FROM S );
```

```
SELECT * FROM R
INNER JOIN S USING
(r1);
```

```
SELECT * FROM R
WHERE r1 NOT IN
(SELECT r1 FROM S);
```

```
SELECT * FROM R
WHERE NOT EXISTS
(SELECT * FROM S
WHERE R.r1 = S.r1);
```

```
SELECT * FROM R
WHERE r1 <> ALL
(SELECT r1 FROM S );
```

```
SELECT * FROM R
LEFT JOIN S USING
(r1) WHERE S.id is
NULL;
```

FASE 6: Exploração

Operações de Junção

A operação de Junção é utilizada para combinação dos dados contidos numa ou mais tabelas através das colunas em comum, ou seja, as *foreign keys*. A cláusula JOIN é usada na instrução SELECT e aparece sempre depois da cláusula FROM.

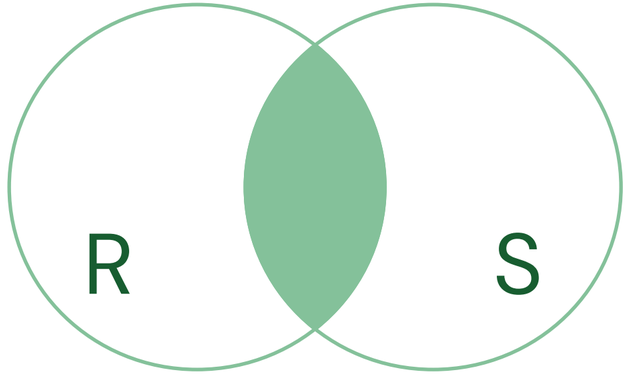
O mysql suporta diferentes operações de junção:

- CROSS JOIN;
- NATURAL JOIN;
- INNER JOIN;
- LEFT JOIN;
- RIGHT JOIN;

FASE 6: Exploração

NATURAL JOIN

A operação de Junção Natural, é uma operação entre duas relações R e S que permite inter-relacionar essas duas relações através das colunas que sejam comuns às duas relações e que possuam valores iguais. O esquema da relação resultante contém todas as colunas de ambas as relações – excluindo-se uma das colunas de junção.



$$SELECT\ *\ FROM\ R\ NATURAL\ JOIN\ S;$$

FASE 6: Exploração

NATURAL JOIN

EXEMPLOS:

- Quais são as especialidades exercidas pelos médicos?

$$SELECT\ *\ FROM\ especialidades\ NATURAL\ JOIN\ medicos;$$

- Liste os pacientes com seguro de saúde.

$$SELECT\ *\ FROM\ pacientes\ NATURAL\ JOIN\ seguros\ WHERE\ dta\_fim\ >\ curdate();$$

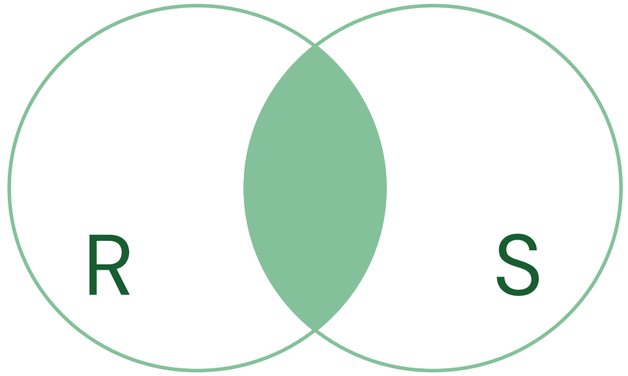
- Liste para cada prescricao, o nome do farmaco, a quantidade prescrita e a unidade.

$$SELECT\ m.nome, p.quantidade, p.unidade\ FROM\ prescricoes\ p\ NATURAL\ JOIN\ medicamentos\ m;$$

FASE 6: Exploração

INNER JOIN

A operação de Junção Interna, é uma operação entre duas relações R e S que permite inter-relacionar essas duas relações através das colunas que satisfaçam a expressão predicativa. O esquema da relação resultante contém todas as colunas de ambas as relações.



$$SELECT * FROM\ R\ INNER\ JOIN\ S\ ON\ R.A = S.B;$$

Para além do operador de igualdade (=), podem ser usados os operadores >, < e <>.

$$SELECT * FROM\ R\ INNER\ JOIN\ S\ USING\ (A);$$

Se as colunas de junçao das duas tabelas tiverem o mesmo nome.

FASE 6: Exploração

INNER JOIN

EXEMPLOS:

- Quais são as especialidades exercidas pelos médicos?

$$SELECT\ *\ FROM\ especialidades\ INNER\ JOIN\ medicos\ USING(cod\_especialidade);$$

- Quais são os medicos que deram consultas?

$$SELECT\ *\ FROM\ medicos\ m\ INNER\ JOIN\ consultas\ c\ ON\ m.nr\_mec = c.nr\_mec\_medico;$$

- Quais as consultas cujo custo final foi inferior ao custo da consulta por especialidade?

$$SELECT\ *\ FROM\ consultas\ c$$
$$INNER\ JOIN\ medicos\ m\ ON\ c.nr_{mec_{medico}} = m.nr_{mec}$$
$$INNER\ JOIN\ especialidades\ e\ ON\ e.cod\_especialidade = m.cod\_especialidade\ AND\ e.preco\_consulta > c.custo\_final;$$

FASE 6: Exploração

LEFT JOIN

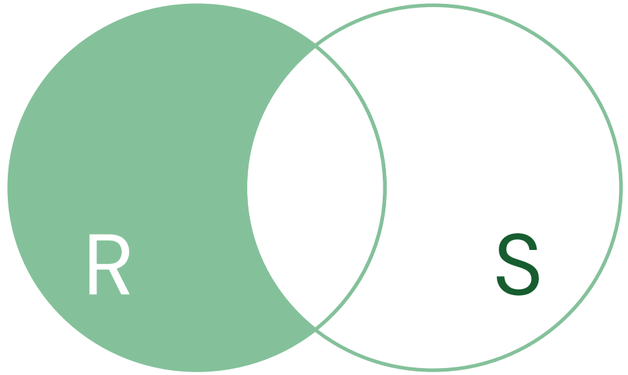
A operação de Junção Externa à Esquerda (*Outer Left Join*), integra na relação final todas as tuplas da relação à esquerda, mesmo quando estas não obedecem aos critérios de junção definidos. Ou seja, os tuplos de R que não têm correspondência nas colunas comuns de S são incluídos no resultado. Quando não existem valores correspondentes na segunda relação S, apresentam-se valores nulos (NULL).

$$SELECT * FROM R LEFT JOIN S ON R.A = S.B;$$
$$SELECT * FROM R LEFT JOIN S USING(A);$$

FASE 6: Exploração

LEFT JOIN

Como a operação de Junção Externa à Esquerda (*Outer Left Join*) integra na relação final todas as tuplas da relação à esquerda R, mesmo quando não têm correspondência na relação à direita S (ou seja apresentam-se valores nulos), é possível selecionar apenas as tuplas da relação R que não têm correspondência na relação S usando a cláusula WHERE e o operador IS NULL.



$$SELECT * FROM R LEFT JOIN S ON R.A = S.B WHERE S.ID IS NULL;$$
$$SELECT * FROM R LEFT JOIN S USING(A) WHERE S.ID IS NULL;$$

FASE 6: Exploração

LEFT JOIN

EXEMPLOS:

- Quais os nomes dos médicos que nunca deram consultas?

$$SELECT\ f.nome\ FROM\ funcionarios\ f\ NATURAL\ JOIN\ medicos\ m$$
$$\textbf{LEFT JOIN}\ consultas\ c\ ON\ c.nr\_mec\_medico = m.nr\_mec\ WHERE\ c.nr\_episodio\ IS\ NULL;$$

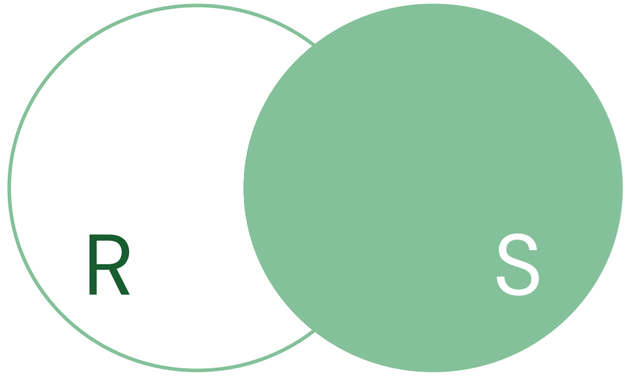
- Quais os nomes dos medicamentos que nunca foram prescritos?

$$SELECT\ m.nome\ FROM\ medicamentos\ m$$
$$\textbf{LEFT JOIN}\ prescricoes\ p\ USING\ (id\_med)$$
$$WHERE\ p.id\_med\ IS\ NULL\ AND\ p.nr\_episodio\ IS\ NULL;$$

FASE 6: Exploração

RIGHT JOIN

A operação de Junção Externa à Direita (*Outer Right Join*), é semelhante Junção Externa à Esquerda, exceto que o tratamento das tabelas unidas é invertido. Ou seja, integra na relação final todas as tuplas da relação à direita, mesmo quando estas não obedecem aos critérios de junção definidos. Quando não existem valores correspondentes na primeira relação R, apresentam-se valores nulos (NULL).

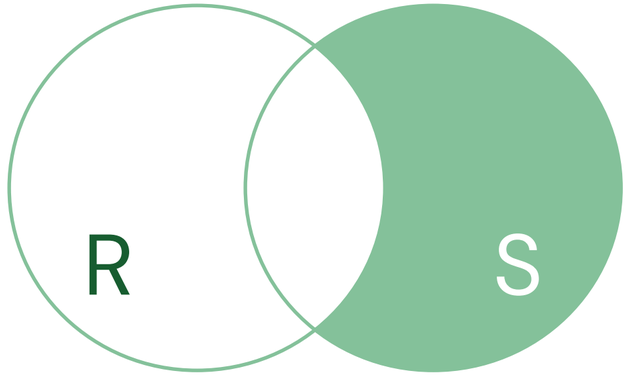


$$SELECT * FROM R RIGHT JOIN S ON R.A = S.B;$$
$$SELECT * FROM R RIGHT JOIN S USING(A);$$

FASE 6: Exploração

RIGHT JOIN

Como a operação de Junção Externa à Direira (*Outer Right Join*) integra na relação final todas as tuplas da relação à direita S, mesmo quando não têm correspondência na relação à esquerda R (ou seja apresentam-se valores nulos), é possível selecionar apenas as tuplas da relação S que não têm correspondência na relação R usando a cláusula WHERE e o operador IS NULL.



$$SELECT * FROM R RIGHT JOIN S ON R.A = S.B WHERE R.ID IS NULL;$$
$$SELECT * FROM R RIGHT JOIN S USING(A) WHERE R.ID IS NULL;$$

FASE 6: Exploração

RIGHT JOIN

EXEMPLOS:

- Liste todos os seguros e os respectivos nomes dos pacientes;

$$SELECT\ nome, nr\_apoliceFROM\ seguros\ \textbf{LEFT JOIN}\ pacientes\ USING\ (nr\_apolice);$$
$$SELECT\ nome, nr\_apolice\ FROM\ pacientes\ \textbf{RIGHT JOIN}\ seguros\ USING\ (nr\_apolice);$$

- Quais os nomes dos médicos que nunca deram consultas?

$$SELECT\ f.nome\ FROM\ consultas\ c\ \textbf{RIGHT JOIN}\ medicos\ m\ NATURAL\ JOIN\ funcionarios\ f$$
$$ON\ c.nr\_mec\_medico = m.nr\_mec\ WHERE\ c.nr\_episodio\ IS\ NULL;$$

$$SELECT\ f.nome\ FROM\ funcionarios\ f\ NATURAL\ JOIN\ medicos\ m$$
$$\textbf{LEFT JOIN}\ consultas\ c\ ON\ c.nr\_mec\_medico = m.nr\_mec\ WHERE\ c.nr\_episodio\ IS\ NULL;$$

FASE 6: Exploração

Resolução de Exercícios

Ficha de Excercícios PL09