Treinamento SQL

SQL – Parte 2 - Consultas

Prof. Dr. Francisco Isidro Massetto
francisco.isidro@techschool.com.br

Agenda

- Consultas simples
  - Dados puros
  - Dados derivados
- Filtragens
  - Dados puros
  - Dados derivados
  - Trabalhando com expressões regulares
- Trabalhando com junções
- Teoria dos conjuntos aplicadas às consultas
- Algumas particularidades

Consultas Simples

- Estrutura básica de uma instrução do tipo SELECT

```
SELECT (lista de campos | *)
  FROM tabela [alias]
  WHERE critério
```

Definindo “aliases” para tabelas

```
Select d.idDepto from Departamento d;
```

- d.idDepto indica o campo “idDepto” da tabela “d”
- “d” é o apelido (alias) dado à tabela Departamento
  - Facilita a compreensão e simplifica a escrita de código

```
Select d.idDepto from Departamento AS d;
```

Definindo “aliases” para campos

```
Select idDepto identifica from Departamento;
```

- Identifica é o “alias” para o campo idDepto
  - Muito utilizado para campos derivados ou resultantes de cálculos

```
Select idDepto AS identifica from Departamento;
```

Filtrando valores distintos

```
Select DISTINCT campos from tabela
```

- Muito usado para selecionar valores não repetidos
- Exemplo: quais departamentos tem funcionários alocados em nossa empresa?

```
SELECT distinct idDepto FROM funcionario;
```

Trabalhando com critérios

- Cláusula Where
  - Define o critério de busca da tabela
- Algumas opções
  - Conjunção de critérios
    - <criterio 1> AND <criterio 2>
  - Disjunções de critérios
    - <criterio 1> OR <criterio 2>
  - Composição de conjunções e disjunções
- E o null?
  - Nunca é usado em critério de igualdade e sim de estado
  - WHERE idDepto = null → errado
  - WHERE idDepto **IS** null → correto
- Também tem o NOT
  - Nega a condição existente

Contando Registros

- SELECT COUNT(nome do campo) FROM tabela WHERE critério

- Quantos departamentos há em nossa empresa?
  - `SELECT COUNT (idDep) from Departamento;`
- Quantos funcionários há em nossa empresa?
  - `SELECT COUNT (idFunc) from Funcionario;`

Melhorando nossas pesquisas

- E se quisermos uma restrição em relação ao número de funcionários?
  - Exemplo: quais departamentos tem mais de 2 funcionários?

```
SELECT nomeDep, count(idDep) FROM Departamento
  HAVING count(idDep) > 2;
```

Agrupando dados

- Uma das perguntas básicas:
  - Quantos funcionários tenho por departamento?
- Não basta apenas contar, deve-se contar quantos existem por departamento
  - Deve-se portanto AGRUPAR os resultados
- Cláusula Group By

```
SELECT lista_campos FROM tabela WHERE critério
  GROUP BY campo
```

  - Select idDepto, count(idDepto) from funcionario group by idDepto

Diferença do HAVING e do WHERE

- WHERE
  - Manipula dados brutos (sem tratamento) ou dados primitivos
- HAVING
  - Manipula dados derivados ou dados agrupados
  - No caso do count(...), ele agrupa vários registros em uma única tupla
    - Qualquer critério envolvendo count, deve ser feito com HAVING

Ordenando Resultados

- Cláusula Order By

```
SELECT lista_campos FROM tabela WHERE critério
   ORDER BY lista_de_campos
```

- Order by é utilizado para ordenar por campos que não fazem parte da chave primária
- Ordenação padrão – campos que compõem a chave primária
- Ordenação também pode ser decrescente
  - Acrescentar DESC no critério do campo

Mais Filtragens

- Condições de Igualdade e Desigualdade
  - =, <>
- Condições de Intervalo
  - >, <, >=, <=
  - BETWEEN → forma elegante de se escrever >= e <=
- Selecionando partes de dados
  - YEAR(campo de data)
  - Month(campo de data)
  - Day(campo de data)
  - Hour(campo de data/hora)
  - Minute(campo de data/hora)
  - Second(campo de data/hora)

Trabalhando com strings

- Operador LIKE
- Caracteres curingas
  - _ → um único caractere
  - % → qualquer seqüência de caracteres

- Todos os funcionários com e-mail diferente do domínio da empresa

```
Select * from funcionario
      where emailF not like '@empr.com.br';
```

Substrings

- LEFT(coluna, qtde)
  - Extrai uma determinada quantidade de caracteres de uma coluna a partir da esquerda
- RIGHT(coluna, qtde)
  - Extrai uma determinada quantidade de caracteres de uma coluna a partir da direita

Expressões Regulares

- Expressões regulares são recursos poderosos para tratamento de textos, strings
  - Permite criar mecanismos de identificação de padrões de caracteres
- Obviamente é bastante restrito à implementação do SGBD
  - No MySQL, usa-se a função REGEXP

Algumas regras de expressões regulares

- ^ → procura a partir do início da string
- $ → procura no final da string
- L* → **0** ou mais ocorrências de L
- L+ → **1** ou mais ocorrências de L
- L? → **0** ou **1** ocorrência de L
- L{n} → exatamente **n** ocorrências de L
- L{n,m} → no mínimo **n** e no máximo **m** ocorrências de L
- [ab] → encontrar a letra **a** ou **b**
- [a-z] → letras no intervalo entre **a** e **z**
- [03] → ou o dígito **0** ou o dígito **3**
- [0-9] → dígitos entre **0** e **9**

Exemplos de expressões Regulares

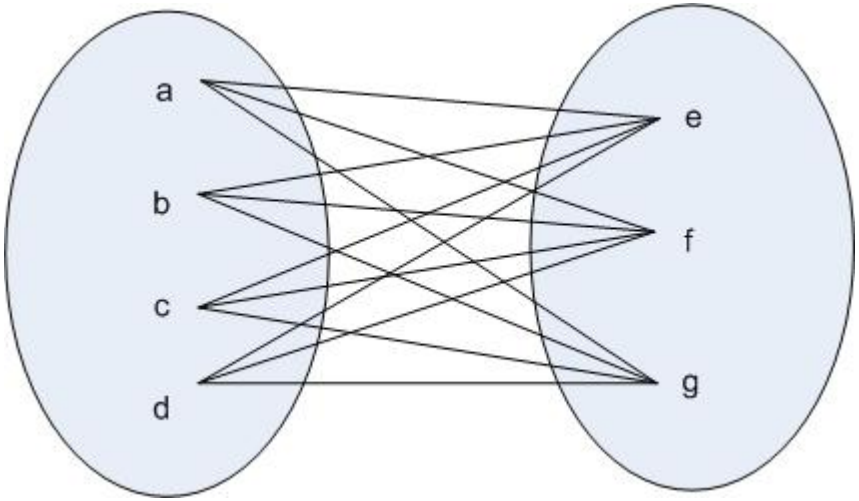
- Procurar placas de carros em um texto
  - `REGEXP '[A-Z]{3}-[0-9]{4}'`
- Os caracteres entre A e Z aparecem exatamente 3 vezes e os dígitos entre 0 e 9 aparecem exatamente 4 vezes

Junções

- O que é, afinal, uma junção?
  - É uma operação realizada entre 2 ou mais tabelas que permitem extrair combinações na forma de produtos cartesianos
    - Junções simples ou junções cruzadas
  - Caso as restrições de chaves estrangeiras sejam respeitadas, permite realizar filtros que sigam estas restrições
    - Junções internas

Produto Cartesiano

- A = {a,b,c,d}
- B = {e, f, g}
- AxB = {(a,e), (b,e), (c,e), (d,e), (a,f), (b,f), (c,f), (d,f), (a,g), (b,g), (c,g), (d,g)}



Sintaxe de uma Junção Simples

```
Select campos from Tabela 1 JOIN
  Tabela 2 [JOIN Tabela 3...]
```

- Exemplo
  - Recuperar todas as combinações de nomes de funcionário e departamento
  - SELECT f.nomeF, d.nomeDep FROM Funcionario f JOIN Departamento d

Junções Simples

```
mysql> select f.nomeF, d.nomeDep from funcionario f join departamento d;
+--------------+------------+
| nomeF        | nomeDep    |
+--------------+------------+
| Isidro       | Engenharia |
| Isidro       | RH         |
| Isidro       | Estoque    |
| Sezefredo    | Engenharia |
| Sezefredo    | RH         |
| Sezefredo    | Estoque    |
| Adamastor    | Engenharia |
| Adamastor    | RH         |
| Adamastor    | Estoque    |
| Deosdedite   | Engenharia |
| Deosdedite   | RH         |
| Deosdedite   | Estoque    |
+--------------+------------+
```

Junções Internas

- Permitem especificar os critérios de associação entre as tabelas
- Apenas os registros que combinam entre si são recuperados
  - Aqueles registros que possuem uma chave primária igual à chave estrangeira especificada, por exemplo
- Sintaxe
  - `SELECT campos FROM tabela1 (INNER) JOIN tabela2 ON (critério)`

Junções Internas

```
mysql> select f.nomeF, d.nomeDep from funcionario f
    -> inner join departamento d on f.idDepto = d.idDepto;
+--------------+------------+
| nomeF        | nomeDep    |
+--------------+------------+
| Isidro       | Engenharia |
| Sezefredo    | Engenharia |
| Adamastor    | Engenharia |
| Deosdedite   | Engenharia |
| Energarda    | RH         |
| Josicleide   | RH         |
| Nilsonclecio | Estoque    |
+--------------+------------+
7 rows in set (0.00 sec)
```

Algumas Observações

- Se você não especificar o termo “INNER” na cláusula Join, o próprio SGBD realiza uma junção interna por padrão
- É possível realizar o mesmo tipo de consulta de outra forma

```
mysql> select f.nomeF, d.nomeDep from funcionario f,
   departamento d where d.idDepto = f.idDepto;
```

- A diferença entre as 2 consultas está na forma de leitura da expressão e implementação dos SGBDs
  - Operação de junção está disponível a partir da versão 92 do ANSI SQL, o que torna a junção uma operação e o filtro outra operação
  - Aumenta a portabilidade da aplicação, pois a junção e o filtro são implementados de formas diferentes nos SGBDs
  - Permite aumentar o grau de legibilidade de uma consulta SQL

Algumas Observações

- Junções caracterizam uma determinada operação
  - Produto cartesiano de dois conjuntos que obedecem uma restrição
  - Relações entre conjuntos
- Filtros caracterizam outra operação
  - A partir de um conjunto já constituído, obtém-se um subconjunto de valores válidos

Mais Junções

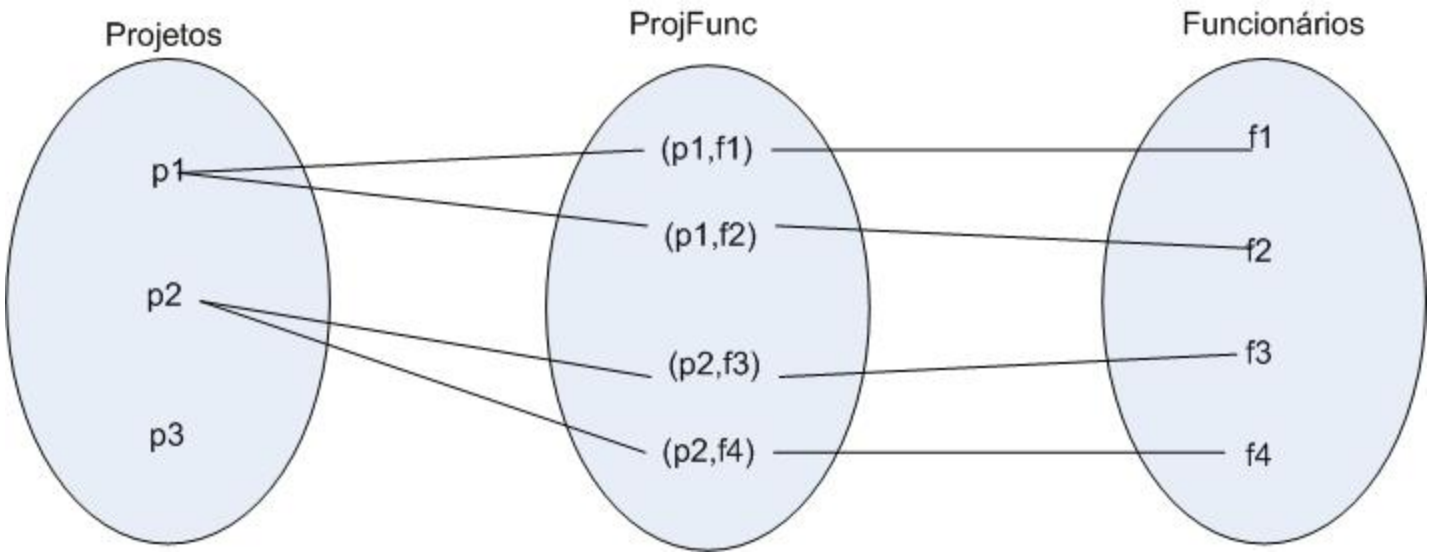
- Quando os campos das tabelas participantes do critério da junção são iguais, pode-se utilizar a cláusula USING(nome do campo)
- Exemplo

```
mysql> select f.nomeF, d.nomeDep from funcionario f
    -> inner join departamento d using(idDepto);
+--------------+------------+
| nomeF        | nomeDep    |
+--------------+------------+
| Isidro       | Engenharia |
| Sezefredo    | Engenharia |
| Adamastor    | Engenharia |
| Deosdedite   | Engenharia |
| Energarda    | RH         |
| Josicleide   | RH         |
| Nilsonclecio | Estoque    |
+--------------+------------+
7 rows in set (0.33 sec)
```

Realizando Junções com mais Tabelas

- Supondo as seguintes tabelas:
  - Funcionario(idFunc, nomeF, ...)
  - Projeto(idProj, descrP)
  - ProjFunc(idProj, idFunc, dataInicio)
- Como saber qual funcionário trabalhou em qual projeto?
  - Junções
- Pensando em conjuntos:

Realizando Junções com mais Tabelas



Realizando Junções com mais Tabelas

- Se realizarmos um Join da tabela Projetos com a tabela ProjFunc, teríamos o seguinte resultado:
  - π(Projeto.idProj, Projeto.descrP, ProjFunc.idProj, ProjFunc.idFunc)
- Podemos, então, realizar a junção da tabela resultante com a tabela Funcionários
  - π(Projeto.idProj, Projeto.descrP, ProjFunc.idProj, ProjFunc,idFunc, Funcionario.idFunc, Funcionario.nomeF)
- Basta, agora, recuperar os campos desejados

Como isso fica em SQL?

- 1o passo – Quais campos eu quero?
  - Nome do Funcionário (nomeF) e Descrição do Projeto (descrP)
- 2o passo – de onde vem esses campos?
  - nomeF de Funcionário
  - descrP de Projeto
- 3o passo – como conseguir isso?
  - Para saber qual funcionário trabalha em qual projeto, a tabela-chave é a ProjFunc
  - Porém deve-se fazer uma junção entre Funcionário, ProjFunc e Projeto

Como fica isso em SQL?

Funcionario INNER JOIN projFunc
Resulta na Tabela
T(idFunc, nomeF, idProj, idFunc, dataInicio)

```
mysql> SELECT f.nomeF, p.descrP FROM
    -> funcionario f INNER JOIN projfunc pf USING(idFunc)
    -> INNER JOIN projeto p USING(idProj);
```

T INNER JOIN projeto
Resulta na Tabela
(idFunc, nomeF, idProj, idFunc, dataInicio, idProj, descrP)

Filtrando resultados

```
SELECT campos FROM
Tabela INNER JOIN Tabela (critério)
INNER JOIN Tabela (critério), ...
WHERE filtro
```

Define-se como os dados a serem recuperados estão relacionados entre si

A partir do resultado obtido, pode-se filtrar os registros desejados

Exemplo

```
mysql> select f.nomeF, p.descrP from
    -> projeto p inner join projfunc using (idProj)
    -> inner join funcionario f using (idFunc)
    -> where p.idProj = 2;
+------------+---------------+
| nomeF      | descrP        |
+------------+---------------+
| Adamastor  | Sist. Escolar |
| Deosdedite | Sist. Escolar |
| Energarda  | Sist. Escolar |
| Josicleide | Sist. Escolar |
+------------+---------------+
4 rows in set (0.00 sec)
```

A ordem das junções importa?

- O fato de SQL ser não-procedimental implica em apenas informar quais dados devem ser recuperados de quais conjuntos (tabelas)
- Em geral todo servidor SQL possui um módulo de interpretação e otimização da query a ser executada
  - Um pré-compilador
- A tarefa de escolher a ordem da tabela é determinada pelo analisador da query.
  - O servidor escolhe uma das tabelas como sendo a principal (ou condutora) e a partir daí, faz as junções com as demais tabelas

Autojunções

- Autojunções são caracterizadas por utilizarem junções envolvendo a mesma tabela
- Alguns exemplos bastante usuais
  - Funcionário que supervisiona funcionário
  - Pessoa que é casada com pessoa
  - Time que enfrenta time
- Para estas operações, o ideal é que haja aliases diferentes para a mesma tabela (indicando o papel que cada uma das entidades tem no relacionamento)
  - Facilita a legibilidade da consulta

Autojunções

```
mysql> select f.nomeF, sup.nomeF from
    -> funcionario f inner join funcionario sup
    -> on f.idSuper = sup.idFunc;
+--------------+------------+
| nomeF        | nomeF      |
+--------------+------------+
| Sezefredo    | Isidro     |
| Adamastor    | Isidro     |
| Deosdedite   | Isidro     |
| Energarda    | Sezefredo  |
| Josicleide   | Sezefredo  |
| Nilsonclecio | Josicleide |
| Roberval     | Josicleide |
+--------------+------------+
7 rows in set (0.09 sec)
```

Junções Equivalentes x Não Equivalentes

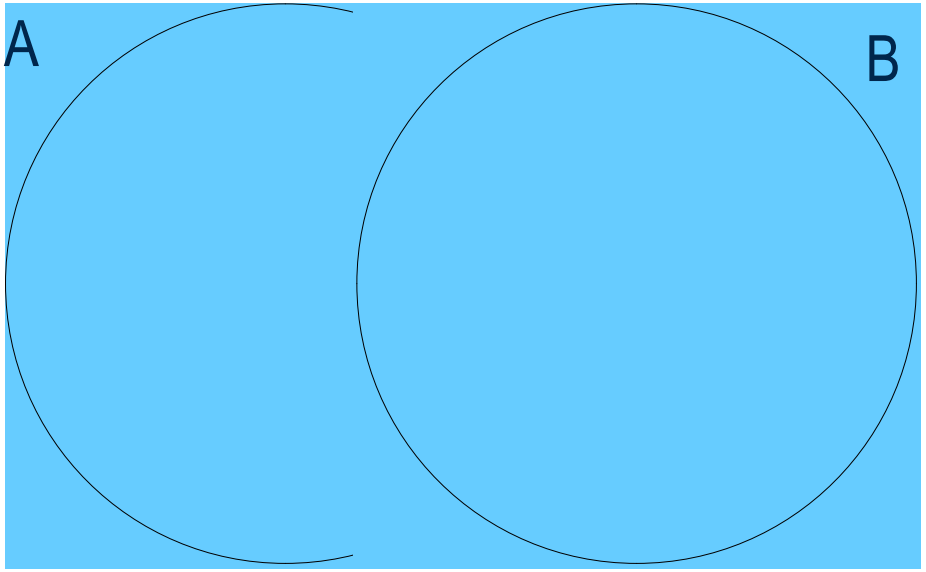
- A maioria das junções emprega igualdades entre campos para se obter o resultado
- É possível também empregar desigualdades nos critérios de recuperação de informações
- Exemplo
  - Supondo um campeonato da empresa onde desejam-se criar chaves de disputas entre os diversos funcionários. Deve-se combinar todos os funcionários (um produto cartesiano), eliminando as situações onde um funcionário enfrenta ele mesmo

Junções Equivalentes X Não Equivalentes

```
mysql> select f.nomeF, 'VS', f2.nomeF from
    -> funcionario f inner join funcionario f2 on f.idFunc != f2.idFunc;
+--------------+----+--------------+
| nomeF        | VS | nomeF        |
+--------------+----+--------------+
| Sezefredo    | VS | Isidro       |
| Adamastor    | VS | Isidro       |
| Deosdedite   | VS | Isidro       |
| Energarda    | VS | Isidro       |
| Josicleide   | VS | Isidro       |
| Roberval     | VS | Isidro       |
     ...                  ...
| Isidro       | VS | Sezefredo    |
| Adamastor    | VS | Sezefredo    |
| Deosdedite   | VS | Sezefredo    |
| Energarda    | VS | Sezefredo    |
| Josicleide   | VS | Sezefredo    |
| Nilsonclecio | VS | Sezefredo    |
| Roberval     | VS | Sezefredo    |
+--------------+----+--------------+
```

Teoria dos Conjuntos na Prática

- União



Operador de União

- UNION
  - Une dois resultados de consultas em um único conjunto
  - Ordena os dados e remove duplicatas
- UNION ALL
  - Inclui duplicatas nos resultados
  - Não ordena o conjunto final
- Requisitos para realizar a operação
  - O número de campos e os tipos de campos devem ser iguais ou equivalentes (possíveis de serem convertidos um para outro)
    - Char para varchar
    - Float para double

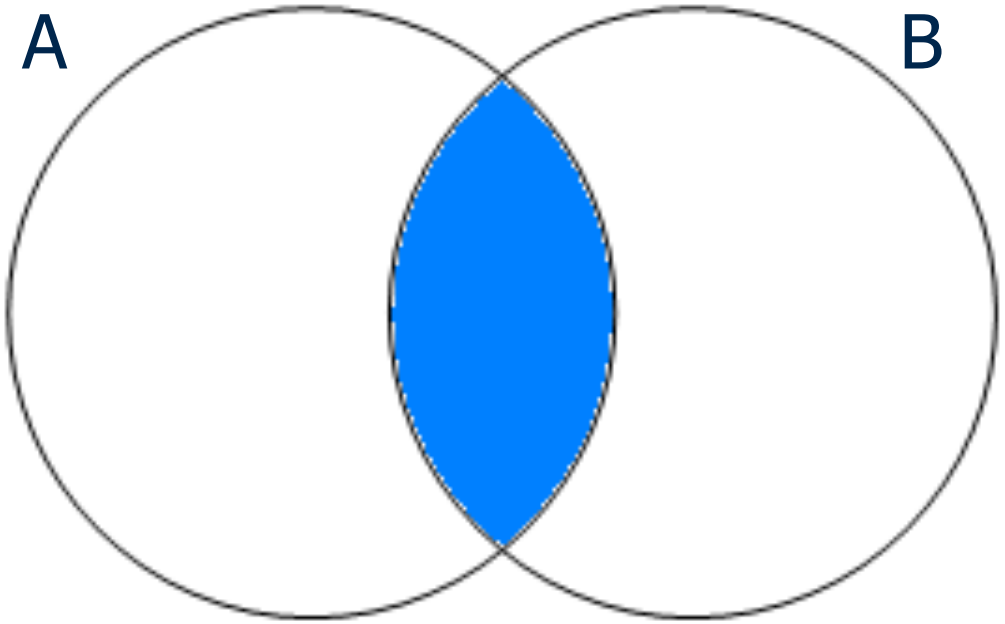
Exemplo

- Caso do empregado com supervisor

```
mysql> select f.nomeF, sup.nomeF from
    -> funcionario f inner join funcionario sup
    -> on f.idSuper = sup.idFunc
    -> union all
    -> select f.nomeF, "NULL" from
    -> funcionario f
    -> where f.idSuper is null;
+--------------+------------+
| nomeF        | nomeF      |
+--------------+------------+
| Sezefredo    | Isidro     |
| Adamastor    | Isidro     |
| Deosdedite   | Isidro     |
| Energarda    | Sezefredo  |
| Josicleide   | Sezefredo  |
| Nilsonclecio | Josicleide |
| Roberval     | Josicleide |
| Isidro       | NULL       |
+--------------+------------+
```

Operador de Intersecção

- INTERSECT

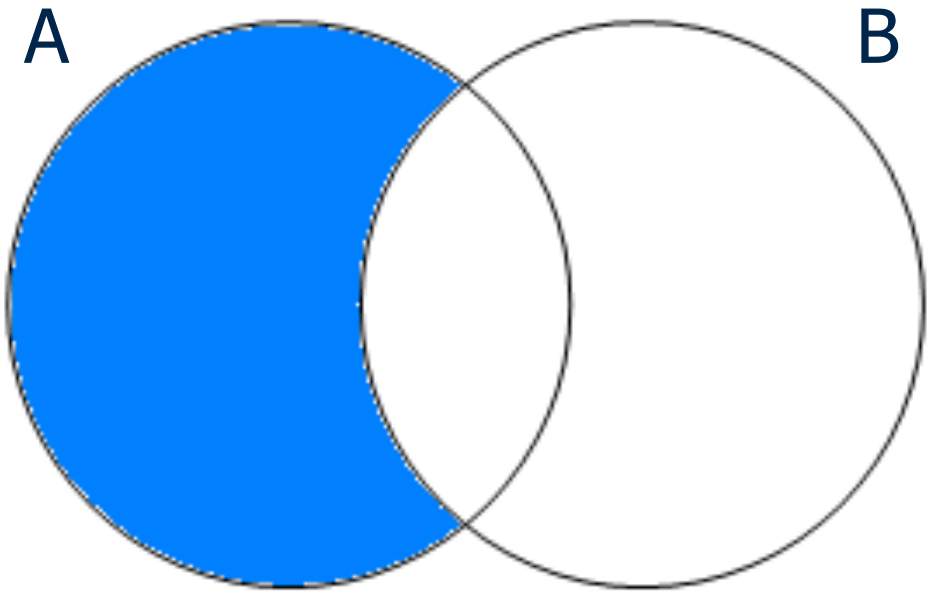


Operador de Intersecção

- INTERSECT
  - Mesmas regras para o operador de união
  - Mostra apenas os registros que fazem parte a ambos os conjuntos
- MySQL
  - Não implementa o operador
- Oracle e SQL Server
  - Implementam

Operador de Subtração

- EXCEPT



Operador de Subtração

- EXCEPT
  - Mesmas regras para o operador de união
  - Mostra apenas os registros que fazem parte a ambos os conjuntos
- MySQL
  - Não implementa o operador
- Oracle e SQL Server
  - Implementam

Trabalhando com Datas

- str_to_date(data, formato)
  - Converte uma string para uma data, a partir do padrão especificado
  - Já visto no módulo SQL Parte 1
  - Usado em operações de inserção e atualização de dados

Trabalhando com Datas

- CURRENT_DATE()
  - Retorna a data atual
- CURRENT_TIME()
  - Retorna a hora atual
- CURRENT_TIMESTAMP()
  - Retorna o timestamp atual (data seguido de hora)

Trabalhando com Datas

- Date_ADD (data_inicial, INTERVAL **n** medida)
  - A partir de uma data ou hora atuais, aplica uma operação de adição de valores
- Medidas
  - Second
  - Minute
  - Hour
  - Day
  - Month
  - Year
  - Minute_Second (MM:SS)
  - Hour_Second (HH:MM:SS)
  - Year_month (Y-M)

Trabalhando com Datas

- DayName(data)
  - Retorna o dia da semana correspondente
- Day(data)
  - Retorna o dia (número de uma data)
- MonthName(data)
  - Retorna o nome do mês de uma data correspondente
- Month(data)
  - Retorna o mês (número)
- Year(data)
  - Retorna o ano de uma data

Trabalhando com Datas

- Hour(hora ou timestamp)
  - Extrai a hora
- Minute(hora ou timestamp)
  - Extrai o minuto
- Second(hora ou timestamp)
  - Extrai os segundos
- Extract(TIPO from data)
  - Extrai uma informação de uma data específica
  - Tipos
    - Day, hour, minute, year (segue o padrão da função DateADD)
- DateDIFF(data1, data2)
  - Retorna o número de dias entre duas datas