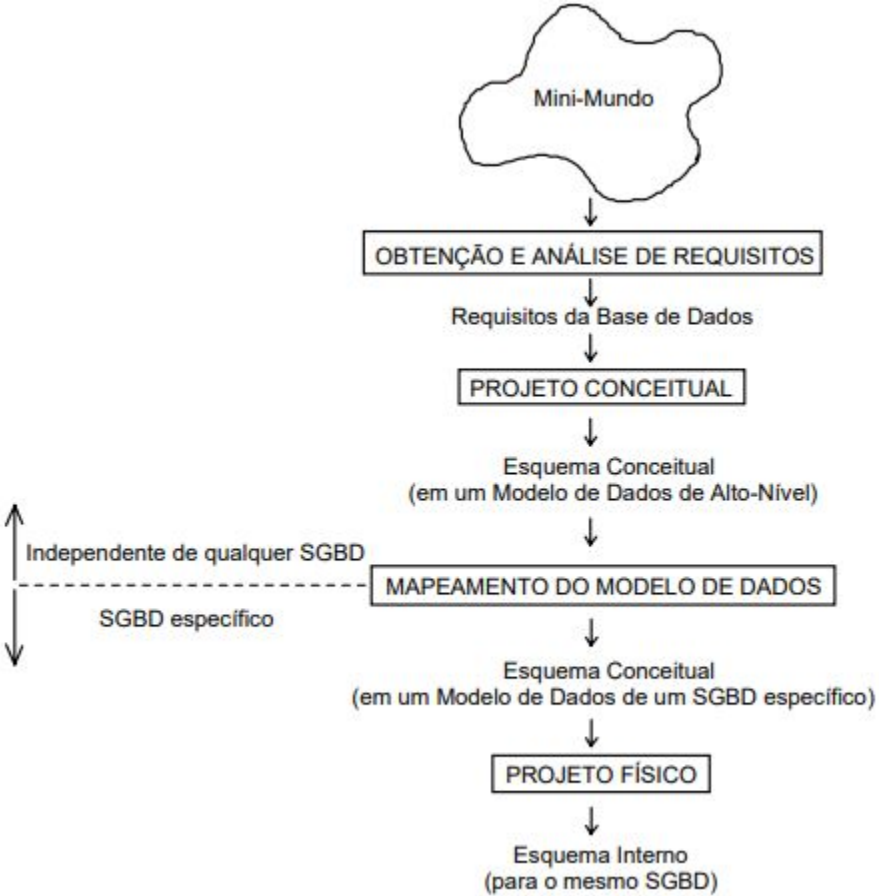
UVV

Design e Desenvolvimento de Banco de Dados I

SQL - DQL - FUNÇÕES e JOIN

Prof. Me Renato Sousa Botacim
Email: renato.botacim@uvv.br


Mini-Mundo
OBTENÇÃO E ANÁLISE DE REQUISITOS
Requisitos da Base de Dados
PROJETO CONCEITUAL
Esquema Conceitual
(em um Modelo de Dados de Alto-Nível)
Independente de qualquer SGBD
MAPEAMENTO DO MODELO DE DADOS
SGBD específico
Esquema Conceitual
(em um Modelo de Dados de um SGBD específico)
PROJETO FÍSICO
Esquema Interno
(para o mesmo SGBD)

Função de Agregação

Funções de agregação são funções que resumem um conjunto em um valor único.

As principais funções de agregação são: COUNT, SUM, AVG, MIN e MAX

COUNT

COUNT: retorna a quantidade de linhas (registros) existentes em uma tabela;

```
SELECT COUNT(*)
  FROM rochas r;

SELECT COUNT(r.cod_rocha)
  FROM rochas r;
```

COUNT(*) todas, incluindo os NULL

COUNT(r.cod_rocha) (r.cod_rocha sendo PK) só as NOT NULL

SUM

SUM: retorna a soma do campo (coluna) passado como parâmetro;

```
SELECT SUM(r.valor_venda)
  FROM rochas r;
```

Este comando irá retornar a SOMA do campo r.valor_venda da tabela rochas;

AVG

AVG: calcula a média aritmética de uma campo (coluna) passado como parâmetro;

```
SELECT AVG(r.valor_venda)
  FROM rochas r;
```

Este comando irá retornar a MÉDIA ARITMÉTICA do campo r.valor_venda da tabela rochas;

MIN

MIN: calcula o valor mínimo (menor) de um campo (coluna) passado como parâmetro;

```
SELECT MIN(r.valor_venda)
  FROM rochas r;
```

Este comando irá retornar o menor valor do campo r.valor_venda da tabela rochas

MAX

MAX: calcula o valor máximo (maior) de um campo (coluna) passado como parâmetro.

```
SELECT MAX(r.valor_venda)
  FROM rochas r;
```

Este comando irá retornar o maior valor do campo r.valor_venda da tabela rochas;

AGRUPAMENTOS

Podemos agrupar nossos SELECTs utilizando a cláusula GROUP BY

```
SELECT r.cod_tipo_rocha,
       COUNT(*)
  FROM rochas r
  GROUP BY r.cod_tipo_rocha;
```

Quantas (quantidade) rochas existem de cada tipo de rocha. Neste caso agrupamos por código do tipo de rocha (r.cod_tipo_rocha).

```
SELECT r.cod_tipo_rocha,
       COUNT(*),
       SUM(r.valor_compra),
       SUM(r.valor_venda)
  FROM rocha r
  GROUP BY r.cod_tipo_rocha;
```

HAVING

HAVING: permite criar uma condição de filtro para o GROUP BY

```
SELECT r.cod_tipo_rocha,
       COUNT(*)
  FROM rochas r
  GROUP BY r.cod_tipo_rocha
  HAVING COUNT(*) > 10;
```

O HAVING só funciona com o GROUP BY. Ele tem a mesma função da cláusula WHERE, mas para valores agrupados.

OBS 1

Lembrar que o fato de uma função de agregação não estar fazendo parte das informações que serão exibidas NÃO significa que a mesma não possa ser utilizada no HAVING.

```
SELECT c.cod_produto,
       COUNT(*)
  FROM ...
  GROUP BY c.cod_produto
  HAVING SUM(valor_venda) > 1000;
```

OBS 2

No HAVING é esperada uma expressão lógica.

Portanto, podemos ter:

HAVING COUNT(*) > 10 AND SUM(valor) < 1000

OU

HAVING (COUNT(*) > 10 AND SUM(valor) < 1000) OR COUNT(*) < 100

Ou seja: qualquer expressão que retorne Verdadeiro ou falso

ORDER BY

ORDER BY: ordena o resultado do SELECT pelo campo(s) (colunas) indicados

Se o objetivo for a ordem decrescente, utiliza-se no final DESC no final do ORDER BY.

```
SELECT tr.nome_tipo_rocha
  FROM rochas r,
       tipos_rocha tr
  WHERE r.cod_tipo_rocha = tr.cod_tipo_rocha
    AND r.cod_rocha      = 100
  ORDER BY tr.nome_tipo_rocha;
```

SE o caso for de sair ordenado de forma decrescente, utiliza-se o DESC

```
SELECT tr.nome_tipo_rocha
  FROM rochas r,
       tipos_rocha tr
  WHERE r.cod_tipo_rocha = tr.cod_tipo_rocha
    AND r.cod_rocha      = 100
  ORDER BY tr.nome_tipo_rocha DESC;
```

```
SELECT r.nome_tipo_rocha,
       COUNT(*),
       SUM(r.valor_venda)
  FROM tipos_rocha tr
  INNER JOIN rochas r ON tr.cod_tipo_rocha = r.cod_tipo_rocha
  GROUP BY r.nome_tipo_rocha
  HAVING COUNT(*) > 10
  ORDER BY r.nome_tipo_rocha;
```

Caso necessário, você pode fazer ordenação dentro de outra ordenação. Por exemplo:

- Quero ordenar as empresas e dentro das empresas quero ordenar as filiais.

Para conseguir essa ordenação basta fazer

ORDER BY empresa, filial

Empresa,	Filial
A	ABC
A	DEF
A	GHI
F	BBB
F	CCC
F	ZZZ
H	JJJ
H	OOO

UPPER/LOWER

Case Sensitive ‘Cachoeiro’ diferente de ‘CACHOEIRO’ que é diferente de ‘cachoeiro’ que é diferente de ‘cACHOEIRA’

Solução:

1) Usar um padrão (UPPER) em todos os cadastros;

2) Usar a função UPPER (tudo em maiúsculas) ou LOWER (tudo em minusculas);

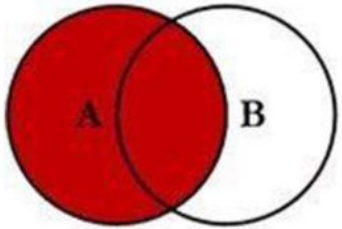
3) POSTGRESQL-> tipo citext.

JOIN - Junção de tabelas

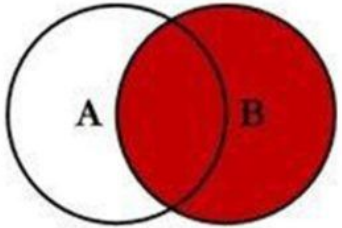
- Uma tabela juntada é uma tabela derivada de outras duas tabelas (reais ou derivadas), de acordo com as regras do tipo particular de junção.

- Estão disponíveis as junções internas, externas e cruzadas

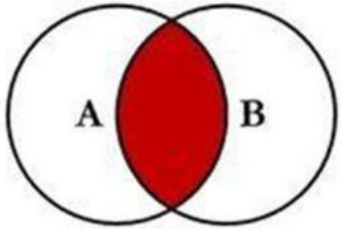
SQL JOINS



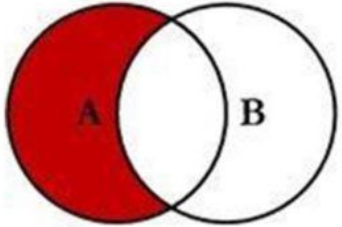

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



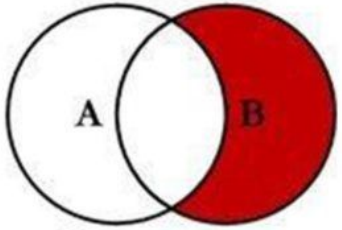

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



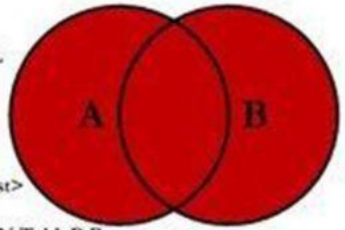

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



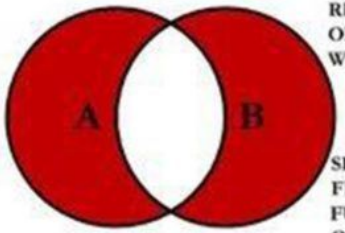

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```

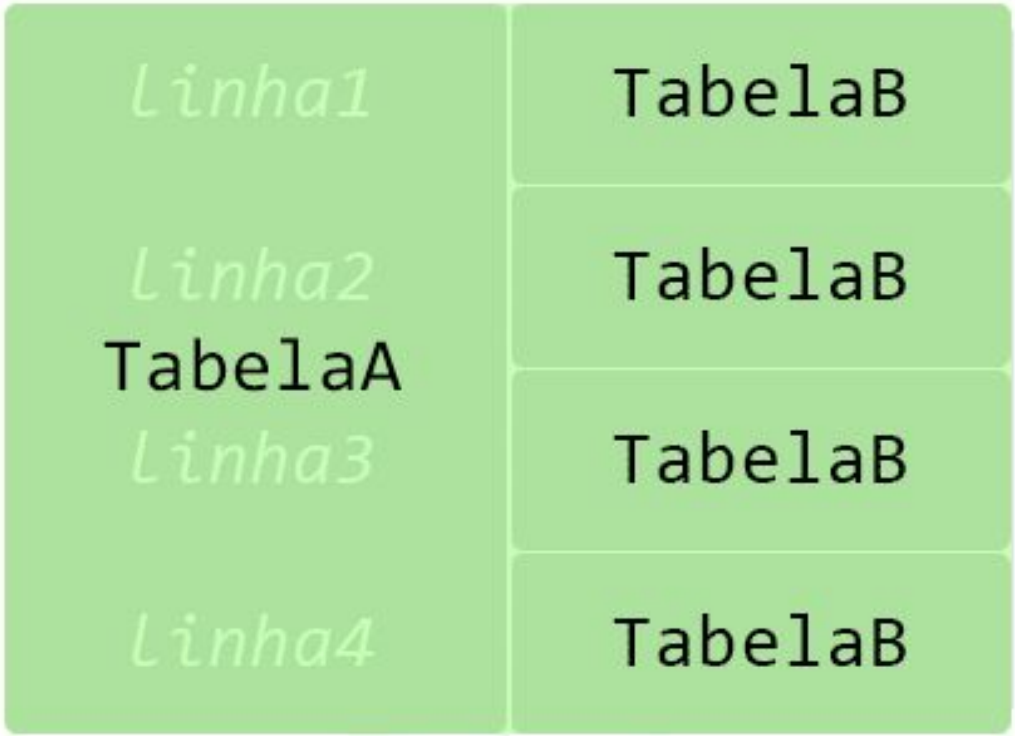



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

Tipos de junção: Junção cruzada - CROSS JOIN

Junção cruzada - CROSS JOIN

- T1 CROSS JOIN T2
  - Para cada combinação de linhas de T1 e T2, a tabela derivada contém uma linha formada por todas as colunas de T1 seguidas por todas as colunas de T2.
  - Se as tabelas possuírem N e M linhas, respectivamente, a tabela juntada terá N * M linhas.


linha1
TabelaB
linha2
TabelaB
TabelaA
linha3
TabelaB
linha4
TabelaB

FROM T1 CROSS JOIN T2

Equivale a FROM T1, T2 (sem a cláusula WHERE)

```
SELECT r.*
  FROM rocha r
  CROSS JOIN tipo_rocha tr;
```

Tipos de junção: Junção Qualificada: INNER JOIN

Junção Qualificada: INNER JOIN

- Para cada linha L1 de T1, a tabela juntada possui uma linha para cada linha de T2 que satisfaz a condição de junção com L1.

```
SELECT *
  FROM t1
  INNER JOIN t2 ON t1.num = t2.num;
```


TabelaA
null
TabelaA.Chave = TabelaB.Chave
null
TabelaB

Tipos de junção: Junção cruzada - INNER JOIN

```
SELECT r.nome_rocha,
       tr.nome_tipo_rocha
  FROM rocha r
  INNER JOIN tipo_rocha tr ON r.cod_tipo_rocha = tr.cod_tipo_rocha;
```

```
SELECT r.nome_rocha,
       tr.nome_tipo_rocha,
       c.nome_cor
  FROM rocha r
  INNER JOIN tipo_rocha tr     ON r.cod_tipo_rocha = tr.cod_tipo_rocha
  INNER JOIN cor c             ON r.cod_cor        = c.cod_cor
  INNER JOIN empresa_rocha er  ON r.cod_rocha      = er.cod_rocha
  INNER JOIN empresa e         ON er.cod_empresa   = e.cod_empresa;
```

Tipos de junção: Junção cruzada - INNER JOIN

- ALTER TABLE rocha ADD COLUMN cod_cor_sec INTEGER NULL;
- UPDATE rocha SET cod_cor_sec = 4;

```
SELECT r.nome_rocha,
       tr.nome_tipo_rocha,
       c.nome_cor AS cor_principal,
       c2.nome_cor AS cor_secundaria
  FROM rocha r
  INNER JOIN tipo_rocha tr     ON r.cod_tipo_rocha = tr.cod_tipo_rocha
  INNER JOIN cor c             ON r.cod_cor        = c.cod_cor
  INNER JOIN cor c2            ON r.cod_cor_sec    = c2.cod_cor
  INNER JOIN empresa_rocha er  ON r.cod_rocha      = er.cod_rocha
  INNER JOIN empresa e         ON er.cod_empresa   = e.cod_empresa;
```

Estilo 1

```
SELECT r.nome_rocha,
       tr.nome_tipo_rocha,
       c.nome_cor  AS cor_principal,
       c2.nome_cor AS cor_secundaria
  FROM rocha r,
       tipo_rocha tr,
       cor c,
       cor c2,
       empresa_rocha er
  WHERE r.cod_tipo_rocha = tr.cod_tipo_rocha
    AND r.cod_cor        = c.cod_cor
    AND r.cod_cor_sec    = c2.cod_cor
    AND r.cod_rocha      = er.cod_rocha;
```

Estilo 2

```
SELECT a.coluna1,
       b.coluna2
  FROM a
  INNER JOIN b ON  a.coluna_fk1 = b.coluna_pk1
               AND a.coluna_fk2 = b.coluna_pk2
               AND a.coluna_fk3 = b.coluna_pk3
  INNER JOIN c ON  b.coluna_fk1 = c.coluna_pk1
  WHERE a.coluna1 LIKE 'A%';
```

Mais um exemplo!

```
SELECT e.nome,
       d.nome_depto
  FROM empregado e
  INNER JOIN departamento d ON e.cod_filial = d.cd_filial
                           AND e.cod_depto  = d.cd_depto;
```

Tipos de junção: LEFT OUTER JOIN

Junção Qualificada: LEFT OUTER JOIN

- Primeiro, é realizada uma junção interna. Depois, para cada linha de T1 que não satisfaz a condição de junção com nenhuma linha de T2, é adicionada uma linha juntada com valores nulos nas colunas de T2.
- Portanto, a tabela juntada possui, incondicionalmente, no mínimo uma linha para cada linha de T1

Retorna a Tabela A inteira e apenas os registros que coincidirem com a igualdade do join na TabelaB (ou campos nulos para os campos sem correspondência).

Tipos de junção: LEFT OUTER JOIN

```
SELECT p.pais,
       c.regiao
  FROM paises p
  LEFT OUTER JOIN cidades c ON p.id_cidade = c.id_cidade;
```

```
SELECT tr.nome_tipo_rocha,
       r.nome_rocha
  FROM tipo_rocha tr
  LEFT OUTER JOIN rocha r ON tr.cod_tipo_rocha = r.cod_tipo_rocha;
```

Tipos de junção: LEFT OUTER JOIN

Listar todos os Departamentos e seus funcionários, mesmo que não existam funcionários

```
SELECT d.nome_depto,
       e.nome
  FROM departamento d
  LEFT OUTER JOIN empregado e ON d.cd_filial = e.cod_filial
                             AND d.cd_depto  = e.cod_depto;
```

Tipos de junção: RIGHT OUTER JOIN

Junção Qualificada: RIGHT OUTER JOIN

- Primeiro, é realizada uma junção interna. Depois, para cada linha de T2 que não satisfaz a condição de junção com nenhuma linha de T1, é adicionada uma linha juntada com valores nulos nas colunas de T1.
- É o oposto da junção esquerda: a tabela resultante possui, incondicionalmente, uma linha para cada linha de T2.

Retorna a Tabela B inteira e apenas os registros que coincidirem com a igualdade do join na TabelaA (ou campos nulos para os campos sem correspondência).

Tipos de junção: RIGHT OUTER JOIN

```
SELECT p.siape,
       p.nome,
       l.lotacao
  FROM pessoal p
  RIGHT OUTER JOIN lotacoes l ON p.siape=l.siape;
```

```
SELECT tr.nome_tipo_rocha,
       r.nome_rocha
  FROM rocha r
  RIGHT OUTER JOIN tipo_rocha tr ON r.cod_tipo_rocha = tr.cod_tipo_rocha;
```

Tipos de junção: RIGHT OUTER JOIN

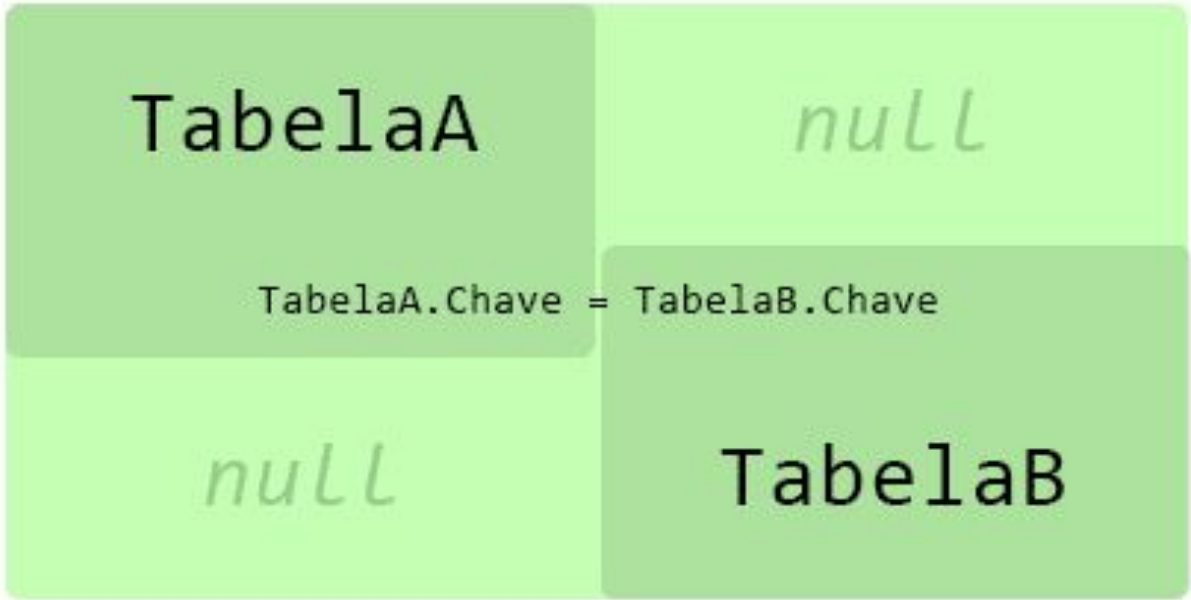
Listar todos os Departamentos, mesmo que não existam funcionários

```
SELECT e.nome,
       d.nome_depto
  FROM empregado e
  RIGHT OUTER JOIN departamento d ON e.cod_filial = d.cd_filial
                                 AND e.cod_depto  = d.cd_depto;
```

Tipos de junção: FULL OUTER JOIN

Junção Qualificada: FULL OUTER JOIN

- Primeiro, é realizada uma junção interna. Depois, para cada linha de T1 que não satisfaz a condição de junção com nenhuma linha de T2, é adicionada uma linha juntada com valores nulos nas colunas de T2.
- Também, para cada linha de T2 que não satisfaz a condição de junção com nenhuma linha de T1, é adicionada uma linha juntada com valores nulos nas colunas de T1.



Retorna todos os registros de ambas as tabelas

Tipos de junção: FULL OUTER JOIN

```
SELECT *
   FROM T1 FULL OUTER JOIN T2
   ON T1.C1 = T2.C1
```

```
SELECT tr.nome_tipo_rocha,
       r.nome_rocha
  FROM rocha r
  FULL OUTER JOIN tipo_rocha tr ON r.cod_tipo_rocha = tr.cod_tipo_rocha;
```

Tipos de junção: NATURAL JOIN

Junção Qualificada: NATURAL JOIN

- Retorna apenas as linhas com valores correspondentes nas colunas correspondentes.
- As colunas correspondentes devem ter os mesmos nomes e tipos de dados similares.

Tipos de junção: NATURAL JOIN

```
SELECT * FROM t1 NATURAL INNER JOIN t2;
```

```
SELECT tr.nome_tipo_rocha,
       r.nome_rocha
  FROM rocha r
  NATURAL INNER JOIN tipo_rocha tr;
```

```
SELECT f.nome_funcionario,
       l.desc_local
  FROM funcionario f
  NATURAL LEFT OUTER JOIN local l;
```

```
SELECT r.nome_rocha,
       e.nome_empresa
  FROM rocha r
  LEFT OUTER JOIN empresas_rocha er
       ON r.cod_rocha = er.cod_rocha
  FULL OUTER JOIN empresa e
       ON er.cod_empresa = e.cod_empresa
  INNER JOIN cidade c
       ON e.cod_cidade = c.cod_cidade
  WHERE r.nome_rocha LIKE 'A%'
  ORDER BY r.nome_rocha;
```