Álgebra Relacional e SQL

Bases de Dados (CC2005)
Departamento de Ciência de Computadores
Faculdade de Ciências da Universidade do Porto

Eduardo R. B. Marques — DCC/FCUP

Introdução

Álgebra relacional

- Álgebra relacional
  - Formalismo usado para modelar relações em uma BD relacional.
  - Usada internamente por SGBDs para representação e optimização de consultas.
  - Tem uma forte correspondência com SQL.

- Aspectos fundamentais:
  - **Caso base de uma relação**: tabela de BD.
  - **Operação na álgebra relacional**: toma uma ou mais relações e devolve uma nova relação.
  - **Derivação de uma relação**: sequência de operações de álgebra relacional.

Álgebra Relacional — operações elementares

π	projeção	$\cup$	união
σ	seleção	$\cap$	intersecção
ρ	renomeação	$-$	diferença
$\mathcal{F}$	agregação	$\times$	produto cartesiano
$\bowtie$	junção		

Projeção, seleção e renomeação

Esquema de relação — definição mais geral

Um **esquema de relação** $R(A_1, \ldots, A_n)$ **tem nome R** e **atributos $A_1, \ldots, A_n$ por ex.** ALUNO(NumMec, NumCC, Nome, Curso).

A cada atributo A_i está associado um **domínio de valores** $dom(A_i)$. Os valores no domínio de um atributo são **atómicos** e podem incluir o valor especial **NULL** i.e., podemos ter **NULL** $\in dom(A_i)$.

Relação — definição mais geral (cont.)

Uma relação com esquema $R(A_1, \ldots, A_n)$ é **um conjunto não ordenado de registos r** que têm a forma de tuplos

$$r = (v_1, \ldots, v_n)$$

tal que $v_i \in \mathbf{dom}(A_i)$ e é denotado por $r[A_i]$.

A noção de relação pode ser também **generalizada a multi-conjuntos** (em vez de conjuntos), permitindo a existência de valores duplicados, em linha com o que é permitido em SQL (ex. consulta de um atributo não-chave).

Detalhe: dada uma relação não ser ordenada o operadores SQL **ORDER BY** e **LIMIT** não podem ser expressos em álgebra relacional.

Projeção

Notação

$\pi_{A_1,\ldots,A_n}(R)$

R relação

$A_1, \ldots, A_n$ atributos de R

Definição

$$\pi_{A_1,\ldots,A_n}(R) = \{(r[A_1], \ldots, r[A_n]) : r \in R\}$$

(projeção de sub-conjunto de atributos de R)

Correspondência em SQL

SELECT A_1, ..., A_n FROM R ;

Projeção — exemplo

$$R \leftarrow \pi_{\text{NumMec,Nome}}(\text{ALUNO})$$

```
SELECT
  NumMec, Nome
FROM
  ALUNO;
```

ALUNO			
NumMec	NumCC	Nome	Curso
798764544	12345678	João Pinto	LCC
345673451	17222303	Carlos Semedo	MIERSI
487563546	12021999	Maria Silva	LBIO
452212348	18392100	Pedro Costa	LMAT
348588664	12848585	Miguel Faria	LCC

R	
NumMec	Nome
798764544	João Pinto
345673451	Carlos Semedo
487563546	Maria Silva
452212348	Pedro Costa
348588664	Miguel Faria

Seleção

Notação

$\sigma_C(R)$ R relação C condição

Definição

$$\sigma_C(R) = \{r \in R : C(r)\}$$

(elementos em R que verificam a condição C)

Correspondência em SQL

SELECT * FROM *R* WHERE *C* ;

Seleção — exemplo

$$R \leftarrow \sigma_{\text{Curso='LCC'} \vee \text{Curso='MIERSI'}}(\text{ALUNO})$$

ALUNO			
NumMec	NumCC	Nome	Curso
798764544	12345678	João Pinto	LCC
345673451	17222303	Carlos Semedo	MIERSI
487563546	12021999	Maria Silva	LBIO
452212348	18392100	Pedro Costa	LMAT
348588664	12848585	Miguel Faria	LCC

```
SELECT  *
FROM ALUNO
WHERE
   Curso = 'LCC'
   OR Curso = 'MIERSI';
```

R			
NumMec	NumCC	Nome	Curso
798764544	12345678	João Pinto	LCC
345673451	17222303	Carlos Semedo	MIERSI
348588664	12848585	Miguel Faria	LCC

Exemplo simples de consulta SQL

```
SELECT
        NumMec, Nome
FROM
        ALUNO
WHERE
        Curso = 'LCC'
        OR Curso = 'MIERSI';
```

projeção π

relação
(neste caso uma tabela)

condição de selecção σ

Exemplo simples de consulta SQL (cont.)

Podemos expressar a derivação de relações como uma sequência de operações na álgebra relacional:

$$R_0 \leftarrow \sigma_{\text{Curso='LCC'} \vee \text{Curso='MIERSI'}}(\text{ALUNO})$$

$$R \leftarrow \pi_{\text{NumMec,Nome}}(R_0)$$

corresponde em SQL à consulta anterior:

```
SELECT  NumMec, Nome
FROM ALUNO
WHERE Curso = 'LCC'
OR Curso = 'MIERSI';
```

Exemplo simples de consulta SQL (cont.)

$$R_0 \leftarrow \sigma_{\text{Curso='LCC'} \vee \text{Curso='MIERSI'}}(\text{ALUNO})$$

$$R \leftarrow \pi_{\text{NumMec,Nome}}(R_0)$$

```
SELECT NumMec, Nome
FROM ALUNO
WHERE Curso = 'LCC' OR Curso = 'MIERSI';
```

ALUNO			
NumMec	NumCC	Nome	Curso
798764544	12345678	João Pinto	LCC
345673451	17222303	Carlos Semedo	MIERSI
487563546	12021999	Maria Silva	LBIO
452212348	18392100	Pedro Costa	LMAT
348588664	12848585	Miguel Faria	LCC

R	
NumMec	Nome
798764544	João Pinto
345673451	Carlos Semedo
348588664	Miguel Faria

Renomeação

Sendo R uma relação com atributos $A_1, \ldots, A_n$

$\rho_{B_1, \ldots, B_n}(R)$ é a mesma relação com os atributos renomeados para $B_1, \ldots, B_n$.

Correspondência em SQL:

SELECT A_1 AS B_1, ..., A_n AS B_n FROM R ;

Renomeação — exemplo

$$R_0 \leftarrow \sigma_{\text{Num}>3}(\text{UTENTE})$$

$$R_1 \leftarrow \pi_{\text{Nome,YEAR(DataNasc)}}(R_0)$$

$$R \leftarrow \rho_{\text{Nome,AnoDeNasc}}(R_1)$$

```
SELECT  Nome, YEAR(DataNasc) AS AnoDeNasc
FROM UTENTE
WHERE Num > 3 ;
```

Agregação simples

Agregação simples

Sendo *R* uma relação, *A* um atributo de *R* e op um operador binário (ex. COUNT, SUM, MIN, ...) sobre dom(A):

$$\mathcal{F}_{\mathrm{op}(A)}(R) = r_1[A] \ \mathrm{op} \ r_2[A] \ \mathrm{op} \ \ldots \ \mathrm{op} \ r_n[A]$$

denota o resultado agregado de aplicar op à sequência de valores de *A* para tuplos em *R*.

Em SQL:

SELECT op(A) FROM *R*

onde op é um operador de agregação.

Agregação simples — notação com renomeação

$$\mathcal{F}_{\mathrm{X=op}(A)}(R) = r_1[A] \ \mathrm{op} \ r_2[A] \ \mathrm{op} \ \ldots \ \mathrm{op} \ r_n[A]$$

denota o resultado agregado de aplicar op à sequência de valores de *A* para tuplos em *R* **e dar um nome X ao atributo da relação resultante. Está implícita uma operação de renomeação.**

Em SQL:

SELECT op(A) **AS X** FROM *R*

onde op é um operador de agregação.

Agregações simples em SQL

- **Sintaxe — forma mais simples:**

SELECT
  <FUNÇÃO DE AGREGAÇÃO>(<ATRIBUTO>) [AS Nome],
FROM …

- **Semântica:**
  - **1)** Agrupa todos os valores de **<ATRIBUTO>** ignorando valores **NULL**
  - **2)** Aplica sobre estes **<FUNÇÃO DE AGREGAÇÃO>**
  - **3)** Devolve o resultado final.
- **Nota:** generaliza-se para múltiplos operadores, como ilustramos a seguir.

Agregação simples — exemplo (cont.)

$$\mathcal{F}_{\text{Num\_Utentes=COUNT(*)}}(\text{UTENTE})$$

```
/* Obtém nº total de utentes. */
SELECT COUNT(*) AS Num_Utentes
FROM UTENTE;
```

```
+-----------------+
| Num_Utentes     |
+-----------------+
|               7 |
+-----------------+
```

- **COUNT(*) :** caso especial — conta o nº de registos.

Agregação combinada com outros operadores

$$R_0 \leftarrow \sigma_{\text{Curso} = \text{'LCC'}} (\text{ALUNO})$$

$$R_1 \leftarrow \mathcal{F}_{\text{N=COUNT(*)}} (R_0)$$

```
SELECT  COUNT(*) AS N
FROM ALUNO
WHERE Curso = 'LCC';
```

2

ALUNO			
NumMec	NumCC	Nome	Curso
798764544	12345678	João Pinto	LCC
345673451	17222303	Carlos Semedo	MIERSI
487563546	12021999	Maria Silva	LBIO
452212348	18392100	Pedro Costa	LMAT
348588664	12848585	Miguel Faria	LCC

Funções de agregação em SQL

- **<FUNÇÃO DE AGREGAÇÃO>** identifica (com algumas excepções) **operação comutativa e associativa** — ex. soma, máximo mas não subtração ou divisão.
- Alguns dos operadores mais comuns são listados abaixo
  - Ver por exemplo referência MySQL para estes e outros.

Função	Operação
COUNT	Contagem
COUNT(DISTINCT)	Contagem de elementos distintos
SUM	Soma
MIN	Mínimo
MAX	Máximo
AVG	Média
STDDEV	Desvio padrão

Agregação simples com vários operadores

A noção de agregação generaliza-se para vários operadores:

$$\mathcal{F}_{X1=op_1(A_1),\ \ldots\ ,X_k=op_k(A_k)}(R) = \left(\mathcal{F}_{op_1(A_1)}(R), \ldots, \mathcal{F}_{op_k(A_k)}(R)\right)$$

Em SQL:

SELECT $op_1(A_1)$ AS X_1, …, $op_k(A_k)$ AS X_k FROM *R*

Agregação simples com vários operadores (cont.)

$$\mathcal{F}_{\text{Min=MIN(YEAR(DataNasc)), Max=MAX(...),Media=AVG(...)}}(\text{UTENTE})$$

```
/* Obtém mínimo, máximo e média
   do ano de nascimento. */
SELECT MIN(YEAR(DataNasc)) As Min,
       MAX(YEAR(DataNasc)) As Max,
       AVG(YEAR(DataNasc)) As Media
FROM UTENTE;
```

```
+------+------+-----------+
| Min  | Max  | Media     |
+------+------+-----------+
| 1975 | 2005 | 1988.2857 |
+------+------+-----------+
```

Agregação agrupada

Agregação agrupada

$$G_1, \ldots, G_n \mathcal{F} X_1 = \mathrm{op}_1(A_1), \ \ldots \ , X_k = \mathrm{op}_k(A_k) (R)$$

generaliza o conceito de agregação simples para agrupada tendo como atributos de agrupamento

$$G_1, \ \ldots, G_n$$

Em SQL:

SELECT $\mathrm{op}_1(A_1)$ AS X_1, ..., $\mathrm{op}_k(A_k)$ AS X_k FROM R
GROUP BY $G_1 \ldots, G_n$

Agregação com GROUP BY

- **Sintaxe — forma mais simples:**

```
SELECT   <ATRIBUTO DE AGRUPAMENTO>,
            <FUNÇÃO DE AGREGAÇÃO>(<ATRIBUTO>)  [AS NOME],
FROM  <TABELA> [WHERE <CONDIÇÃO>]
GROUP BY <ATRIBUTO PARA AGRUPAMENTO>;
```

- **Semântica:**
  - **1)** Divide todos os registos em **grupos** separados, **um por cada valor distinto** de <ATRIBUTO PARA AGRUPAMENTO> (em vez de um só quando não temos GROUP BY).
  - **2)** Aplica sobre **cada grupo** <FUNÇÃO DE AGREGAÇÃO>
  - **3)** Devolve o resultado final para **cada grupo**.
- **Nota:** Podemos especificar várias funções de agregação e vários atributos de agrupamento.

Agregação agrupada — exemplo

$$R \leftarrow {}_{\text{Curso}}\mathcal{F}_{\text{N=COUNT(*)}}(\text{ALUNO})$$

```
SELECT  Curso, COUNT(*) AS N
FROM ALUNO
GROUP BY Curso;
```

ALUNO			
NumMec	NumCC	Nome	Curso
798764544	12345678	João Pinto	LCC
345673451	17222303	Carlos Semedo	MIERSI
487563546	12021999	Maria Silva	LBIO
452212348	18392100	Pedro Costa	LMAT
348588664	12848585	Miguel Faria	LCC

R	
Curso	N
LCC	2
MIERSI	1
LBIO	1
LMAT	1

Agregação com GROUP BY - HAVING

- **Sintaxe — forma mais simples:**

```
SELECT   [<ATRIBUTO DE AGRUPAMENTO>],
         <FUNÇÃO DE AGREGAÇÃO>(<ATRIBUTO>)  [AS NOME]
FROM  <TABELA> [WHERE <CONDIÇÃO>]
GROUP BY <ATRIBUTO DE AGRUPAMENTO>
HAVING <CONDIÇÃO SOBRE GRUPO>;
```

- **Semântica:**
  - <CONDIÇÃO SOBRE GRUPO> filtra resultados por grupo
  - **HAVING** funciona como cláusula **WHERE** ao nível de cada grupo
  - **Em álgebra relacional:** corresponde a uma operação de agregação seguida de uma operação de seleção.

Agregação agrupada com HAVING — exemplo

$$R_0 \leftarrow {}_{\text{Curso}}\mathcal{F}_{\text{N=COUNT(*)}}(\text{ALUNO})$$
$$R \leftarrow \sigma_{\text{N=1}}(R_0)$$

```
SELECT  Curso, COUNT(*) AS N
FROM ALUNO
GROUP BY Curso;
HAVING N = 1;
```

ALUNO			
NumMec	NumCC	Nome	Curso
798764544	12345678	João Pinto	LCC
345673451	17222303	Carlos Semedo	MIERSI
487563546	12021999	Maria Silva	LBIO
452212348	18392100	Pedro Costa	LMAT
348588664	12848585	Miguel Faria	LCC

R	
Curso	N
MIERSI	1
LBIO	1
LMAT	1

WHERE vs. HAVING

$$R_0 \leftarrow \sigma_{\text{LENGTH(Content)} > 20}(\text{POST})$$

$$R_1 \leftarrow \text{Author}\ \mathcal{F}_{\text{NumberOfPosts} = \text{COUNT}(*),\ \text{LatestPost} = \text{MAX(Creation)}}(R_0)$$

$$R_2 \leftarrow \sigma_{\text{YEAR(LatestPost)} = 2020}(R_1)$$

```
SELECT Author,
       COUNT(*) AS NumberOfPosts,
       MAX(Creation) AS LatestPost
FROM POST
WHERE LENGTH(Content) > 20
GROUP BY Author
HAVING YEAR(LatestPost) = 2020;
```

- Exemplo da BD "Livro das Caras": obtém autor e número de posts, e data mais recente de criação de post, para posts com conteúdo de tamanho superior a 20 (cláusula **WHERE**, avaliada **antes** de qualquer agregação) e de tal forma que o ano do último post tenha sido 2020 (cláusula **HAVING**, avaliada **após** a agregação).

Agrupamento com mais de 1 atributo

```
SELECT
  YEAR(Creation) AS Year,
  MONTH(Creation) AS Month,
  COUNT(*) AS NumberOfPosts
FROM POST
GROUP BY Year, Month ;
```

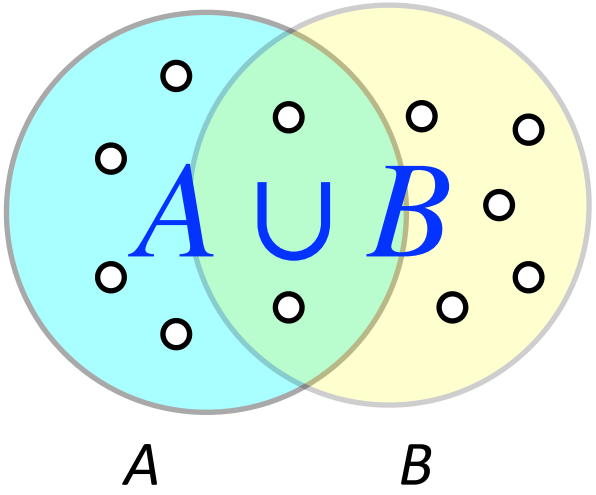
```
+------+-------+---------------+
| Year | Month | NumberOfPosts |
+------+-------+---------------+
| 2019 |    12 |             3 |
| 2020 |     1 |             2 |
| 2020 |     2 |             7 |
+------+-------+---------------+
```

- Novo exemplo da BD "Livro das Caras": consulta obtém número de posts agrupados por ano e mês.

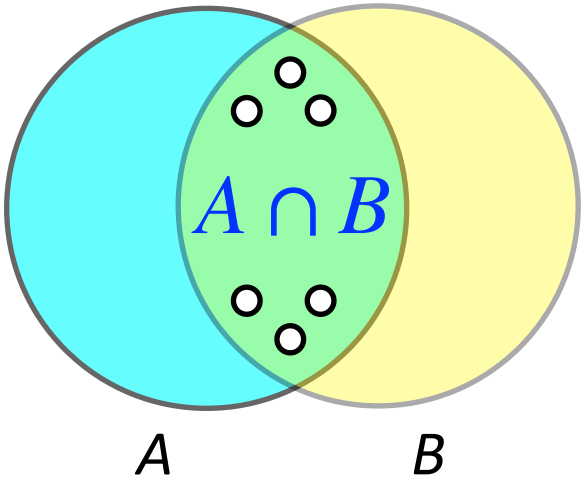
Operações sobre conjuntos

Operações sobre conjuntos

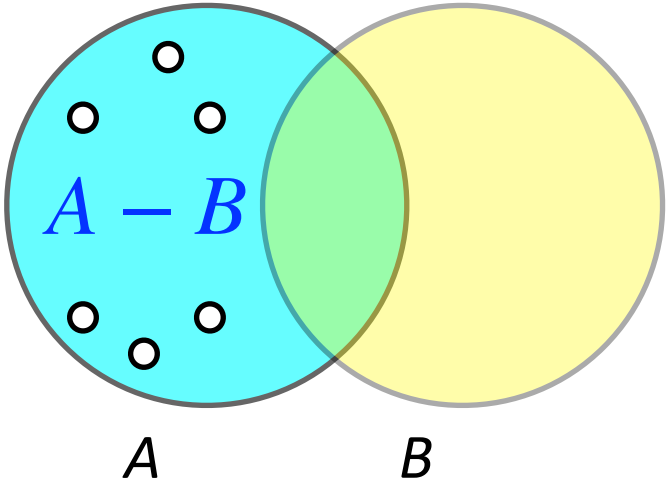
União



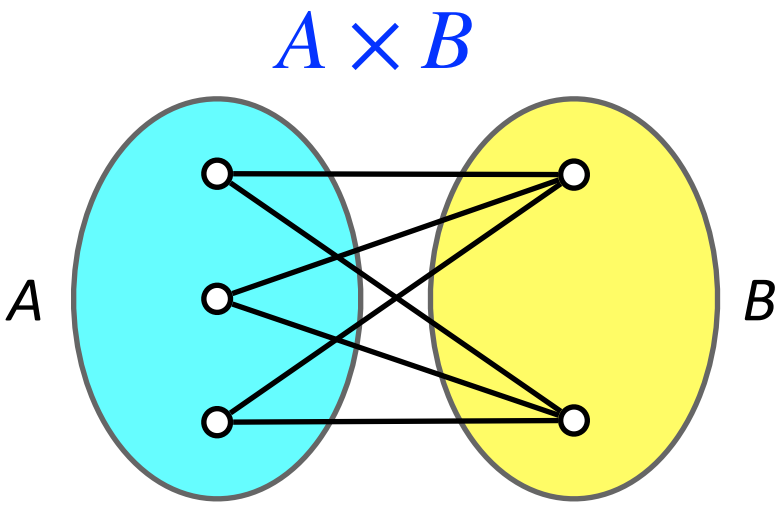
Intersecção



Diferença



Produto cartesiano



$$A \cap B = \{x : x \in A \wedge x \in B\}$$

$$A \cup B = \{x : x \in A \vee x \in B\}$$

$$A - B = \{x : x \in A \wedge x \notin B\}$$

$$A \times B = \{(x, y) : x \in A \wedge y \in B\}$$

Operações sobre conjuntos e SQL

$R \cup S$ — *R* **UNION** *S*

$R \cap S$ — *R* **INTERSECT** *S*

$R - S$ — *R* **EXCEPT** *S*

$R \times S$ — R **JOIN** S **ou** **SELECT** * **FROM** *R,S*

Nota: Não suportados em MySQL

União, interseção, e diferença

As operações de união, interseção e diferença de relações R e S

$$R \cap S \qquad R \cup S \qquad R - S$$

estão restritas a **relações compatíveis**

$$R(A_1, \ \ldots \ , A_n) \quad \text{e} \quad S(B_1, \ \ldots \ , B_n)$$

tais que $\forall i = 1, \ \ldots \ , n, \quad \text{dom}(A_i) = \text{dom}(B_i)$

Não é imposto que os atributos tenham nomes iguais, mas por convenção $R \cap S \qquad R \cup S \qquad R - S$ retêm os nomes dos atributos de R.

Exemplo de relações compatíveis

$$R_0 \leftarrow \sigma_{\text{YEAR(DataNasc)}>1985}(\text{UTENTE})$$

$$R_1 \leftarrow \pi_{\text{Num,Nome}}(R_0)$$

$$S_0 \leftarrow \sigma_{\text{MONTH(DataNasc)}=1}(\text{UTENTE})$$

$$S_1 \leftarrow \pi_{\text{Num,Nome}}(R_0)$$

$$U \leftarrow R_1 \cup S_1$$

```
SELECT Num, Nome
FROM UTENTE
WHERE
  YEAR(DataNasc) > 1985
UNION
SELECT Num, Nome
WHERE
   MONTH(DataNasc) = 1
```

Exemplo de relações incompatíveis

$$R_0 \leftarrow \sigma_{\text{YEAR(DataNasc)}>1985}(\text{UTENTE})$$

$$R_1 \leftarrow \pi_{\text{Num,Nome}}(R_0)$$

$$S_0 \leftarrow \sigma_{\text{MONTH(DataNasc)}=1}(\text{UTENTE})$$

$$S_1 \leftarrow \pi_{\text{DataNasc,Nome}}(R_0)$$

$$U \leftarrow R_1 \cup S_1$$

$$\text{dom}(R_1 . \text{Num}) \neq \text{dom}(S_1 . \text{DataNasc})$$

```
SELECT Num, Nome
FROM UTENTE
WHERE
  YEAR(DataNasc) > 1985
UNION
SELECT DataNasc, Nome
WHERE
   MONTH(DataNasc) = 1
```

União / UNION — exemplo (MovieStream)

$$R \leftarrow \pi_{\text{Manager}}(\text{DEPARTMENT})$$

$$S \leftarrow \pi_{\text{RegionManager}}(\text{REGION})$$

$$U \leftarrow R \cup S$$

```
/* Obtém o id de todos os funcionários
que são gerentes de departamento ou de região
*/
SELECT  Manager FROM DEPARTMENT
UNION
SELECT  RegionManager FROM REGION;
```

Manager
2
7
11
14
15
16
17

Obs.: coluna da relação é Manager em linha com convenção mencionada no slide anterior.

Interseção / INTERSECT — exemplo (MovieStream)

$$R \leftarrow \pi_{\text{MovieId}}(\text{MOVIE})$$

$$S \leftarrow \pi_{\text{MovieId}}(\text{STREAM})$$

$$I \leftarrow R \cap S$$

/* Obtém id de filmes com algum "stream" feito [não suportado em MySQL] */

```
SELECT  MovieId FROM MOVIE
INTERSECT
SELECT  MovieId FROM STREAM;
```

Obs: já veremos como expressar uma consulta em MySQL com resultado equivalente.

Diferença / EXCEPT — exemplo (MovieStream)

$$R \leftarrow \pi_{\text{MovieId}}(\text{MOVIE})$$

$$S \leftarrow \pi_{\text{MovieId}}(\text{STREAM})$$

$$D \leftarrow R - S$$

/* Obtém id de filmes sem nenhum "stream" feito [não suportada em MySQL] */

```
SELECT  MovieId FROM MOVIE
EXCEPT
SELECT  MovieId FROM STREAM;
```

Obs: já veremos como expressar uma consulta em MySQL com resultado equivalente.

Consultas anteriores em MySQL

```
SELECT  MovieId FROM MOVIE
INTERSECT
SELECT  MovieId FROM STREAM;
```

```
SELECT  MovieId FROM MOVIE
WHERE MovieId IN
(SELECT  MovieId FROM STREAM);
```

```
SELECT  MovieId FROM MOVIE
EXCEPT
SELECT  MovieId FROM STREAM;
```

```
SELECT  MovieId FROM MOVIE
WHERE MovieId NOT IN
(SELECT  MovieId FROM STREAM);
```

- Em MySQL (e SQL em geral) podemos fazer uso de **consultas imbricadas** ("nested queries"), que são muito convenientes em uma série de casos.
- No caso usamos os operadores **IN** e **NOT IN** para suprir a falta de suporte de **INTERSECT** e **EXCEPT**. Falaremos de consultas imbricadas em maior detalhe depois.

Produto cartesiano

Para relações $R(A_1, \ldots, A_m)$ e $S(B_1, \ldots, B_n)$

a relação

$$R \times S$$

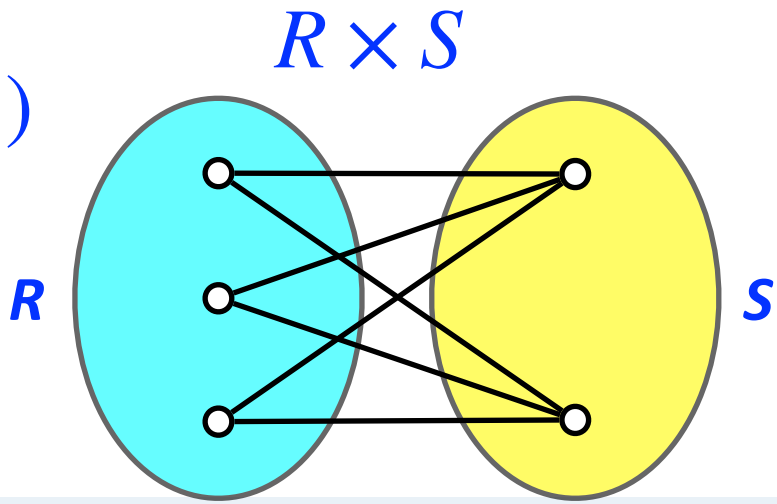
tem atributos $A_1, \ldots, A_m, B_1, \ldots, B_n$

e elementos

$$(r[A_1], \ldots, r[A_m], s[B_1], \ldots, s[B_n])$$

onde

$$r \in R \quad s \in S$$



Produto cartesiano — exemplo (MovieStream)

$$C_0 \leftarrow \pi_{\text{Name}}(\text{COUNTRY})$$
$$C \leftarrow \rho_{\text{CName}}(C_0)$$

$$R_0 \leftarrow \pi_{\text{Name}}(\text{REGION})$$
$$R \leftarrow \rho_{\text{RName}}(R_0)$$

$$P \leftarrow C \times R$$

```
SELECT COUNTRY.Name AS CName,
       REGION.Name  AS RName
FROM COUNTRY JOIN REGION;
```

OU

```
SELECT COUNTRY.Name AS CName,
       REGION.Name AS RName
FROM COUNTRY, REGION;
```

- **Na cláusula FROM podemos especificar várias tabelas para consultas**, o que poderá ser bastante conveniente.

Produto cartesiano — exemplo (cont.)

$$C_0 \leftarrow \pi_{\text{Name}}(\text{COUNTRY})$$
$$C \leftarrow \rho_{\text{CName}}(C_0)$$

$$R_0 \leftarrow \pi_{\text{Name}}(\text{REGION})$$
$$R \leftarrow \rho_{\text{RName}}(R_0)$$

$$P \leftarrow C \times R$$

```
SELECT COUNTRY.Name AS CName,
       REGION.Name AS RName
FROM COUNTRY, REGION;
```

CName	RName
Anguilla	Other countries
Anguilla	America
Anguilla	Asia
Anguilla	Europe
Anguilla	Africa
French Polynesia	Other countries
French Polynesia	America
French Polynesia	Asia
French Polynesia	Europe
French Polynesia	Africa

. . .

- Obtemos todas as combinações de nomes de países e regiões, o que não é informação muito útil.

Produto cartesiano — outro exemplo (cont.)

$$C \leftarrow \rho_{\text{Name/CName}}(\text{COUNTRY})$$
$$R \leftarrow \rho_{\text{Name/RName}}(\text{REGION})$$
$$P_0 \leftarrow C \times R$$

$$P_1 \leftarrow \sigma_{\text{COUNTRY.RegionId=REGION.RegiondId}}(P_0)$$
$$P \leftarrow \pi_{\text{CName,RName}}(P_1)$$

```
SELECT COUNTRY.Name AS CName,
       REGION.Name AS RName
FROM COUNTRY, REGION
WHERE COUNTRY.RegionId = REGION.RegionId;
```

- Obtemos neste caso todas as combinações de nomes de países e das regiões correspondentes, o que é muito mais relevante.
- O exemplo acima corresponde a uma "junção natural", conceito que introduziremos mais à frente nestes "slides".

Produto cartesiano — outro exemplo (cont.)

```
SELECT COUNTRY.Name AS CName,
       REGION.Name AS RName
FROM COUNTRY, REGION
WHERE COUNTRY.RegionId = REGION.RegionId;
```

CName	RName
Anguilla	Other countries
French Polynesia	Other countries
Greenland	Other countries
Nauru	Other countries
New Zealand	Other countries
Reunion	Other countries
Saint Vincent and the Grenadines	Other countries
Tonga	Other countries
Tuvalu	Other countries
Virgin Islands	Other countries
American Samoa	America
Argentina	America
Bolivia	America
Brazil	America

. . .

Consultas em múltiplas tabelas

- **Forma geral**

```
SELECT
   <ATRIBUTOS>
FROM  <TABELA 1> [ID 1], …, <TABELA n> [<ID n>]
…
```

- De forma geral a cláusula **FROM** pode referir várias tabelas.
- Por forma a desambiguar atributos com o mesmo nome ou tornar a consulta mais legível podemos renomear cada tabela com um identificador.
- **Nota**: Muitas vezes estas consultas correspondem implicitamente a operações de **junção** **("join")** a ver mais tarde e para as quais há suporte SQL mais especializado.

Exemplos de consultas em múltiplas tabelas

```
/*
    Obtém nomes de supervisores
    e correspondentes funcionários
    supervisionados.
*/
SELECT S1.Name AS Supervisor,
       S2.Name AS Supervised
FROM STAFF S1, STAFF S2
WHERE S1.StaffId = S2.Supervisor
ORDER BY Supervisor, Supervised;
```

Supervisor	Supervised
António Mota	Felícia Antunes
António Mota	Gabriela Silva
António Mota	Gastão Pinto
Augusto Sousa	Alexandra Romeu
Augusto Sousa	Fábio Cruz
Eva Mendes	Maria Silva
Eva Mendes	Pedro Simões
João Pinto	António Mota
João Pinto	Augusto Sousa
João Pinto	José Santos
João Pinto	Xavier Semedo
João Santorini	Joana Moreira
José Santos	Filipa Magalhães
José Santos	João Santorini
Xavier Semedo	Eva Mendes
Xavier Semedo	Filipa Mendes

16 rows in set (0.01 sec)

Exemplos de consultas em múltiplas tabelas (cont.)

```
/*
   Obtém nomes de supervisores
   e correspondente número de
   funcionários supervisionados.
*/
SELECT S1.Name AS Supervisor,
       COUNT(*) AS N
FROM STAFF S1, STAFF S2
WHERE S1.StaffId = S2.Supervisor
GROUP BY Supervisor
ORDER BY Supervisor;
```

```
+-----------------+---+
| Supervisor      | N |
+-----------------+---+
| António Mota    | 3 |
| Augusto Sousa   | 2 |
| Eva Mendes      | 2 |
| João Pinto      | 4 |
| João Santorini  | 1 |
| José Santos     | 2 |
| Xavier Semedo   | 2 |
+-----------------+---+
7 rows in set (0.01 sec)
```

Exemplos de consultas em múltiplas tabelas (cont.)

```
SELECT A.Name AS Name,
       M.Title AS Title,
       M.Year AS Year
FROM ACTOR A,
     MOVIE_ACTOR MA,
     MOVIE M
WHERE
    A.ActorId = MA.ActorId
  AND
    MA.MovieId = M.MovieId
  AND
    A.Name LIKE 'Brad%'
  AND
    M.Year >= 2010
ORDER BY Name, Year, Title;
```

```
+---------------+---------------------------+------+
| Name          | Title                     | Year |
+---------------+---------------------------+------+
| Brad Pitt     | Megamind                  | 2010 |
| Brad Pitt     | Moneyball                 | 2011 |
| Brad Pitt     | 12 Years a Slave          | 2013 |
| Brad Pitt     | World War Z               | 2013 |
| Brad Pitt     | Fury                      | 2014 |
| Brad Pitt     | The Big Short             | 2015 |
| Bradley Cooper| The A-Team                | 2010 |
| Bradley Cooper| Limitless                 | 2011 |
| Bradley Cooper| The Hangover Part II      | 2011 |
| Bradley Cooper| Silver Linings Playbook   | 2012 |
| Bradley Cooper| The Place Beyond the Pines| 2012 |
| Bradley Cooper| American Hustle           | 2013 |
| Bradley Cooper| The Hangover Part III     | 2013 |
| Bradley Cooper| American Sniper           | 2014 |
| Bradley Cooper| Guardians of the Galaxy   | 2014 |
+---------------+---------------------------+------+
15 rows in set (0.00 sec)
```

Junções (“joins”)

Operações de junção e SQL

Junção	$R \bowtie_C S$	*R* JOIN S ON (C)
Junção natural	$R * S$	*R* NATURAL JOIN S
	$R *_{\text{Attrs}} S$	*R* JOIN S USING(Attrs)
Junções externas	$R \overset{C}{\bowtie\!\!\!\!\!\!\bowtie} S$	*R* LEFT JOIN S ON(C)
	$R \overset{C}{\bowtie\!\!\!\!\!\!\bowtie} S$	*R* RIGHT JOIN S ON(C)
	$R \overset{C}{\bowtie\!\!\!\!\!\!\bowtie} S$	*R* FULL JOIN S ON(C)

Junção

Para relações $R(A_1, \ldots, A_m)$ e $S(B_1, \ldots, B_n)$

e condição C sobre $R \times S$

$$R \bowtie_C S = \sigma_C (R \times S)$$

é chamada a **junção** de R e S com condição C.

Os elementos têm a forma:

$$(r[A_1], \ldots, r[A_m], s[B_1], \ldots, s[B_n])$$

tais que $r \in R \quad s \in S \quad C(r, s)$

Em SQL: ***R*** **JOIN** **S** **ON(C)**

Junção — exemplos

$$R \leftarrow \text{CUSTOMER} \bowtie_{\text{CUSTOMER.Country} = \text{COUNTRY.Name}} \text{COUNTRY}$$

```
SELECT *
FROM CUSTOMER JOIN COUNTRY
    ON(CUSTOMER.Country = COUNTRY.Name);
```

equivalente a

```
SELECT *
FROM CUSTOMER, COUNTRY
WHERE CUSTOMER.Country = COUNTRY.Name;
```

- Esta junção diz-se uma **equi-junção**: a condição é expressa pela igualdade de atributos. Junções envolvem muitas vezes igualdade de atributos.

Junção – exemplos (cont.)

$R \leftarrow \text{CUSTOMER} \bowtie_{\text{CUSTOMER.Country} = \text{COUNTRY.Name}} \text{COUNTRY}$

```
SELECT *
FROM CUSTOMER JOIN COUNTRY
    ON(CUSTOMER.Country = COUNTRY.Name);
```

CustomerId	Name	…	Country	…	Name	RegionId
381	Bobby Boudreau	…	Anguilla	…	Anguilla	6
43	Christine Roberts	…	French Polynesia	…	French Polynesia	6
56	Gloria Cook	…	French Polynesia	…	French Polynesia	6
207	Gertru de Castillo	…	Greenland	…	Greenland	6
513	Duane Tubbs	…	Nauru	…	Nauru	6
. . .					. . .	

- **Observe que ficamos com atributos com valores duplicados (Country em Customer e Name em Region e também dois atributos com o mesmo nome (coluna Name nas 2 tabelas) embora não relacionados.**

Junção – exemplos (cont.)

(abreviando MOVIE por M e STREAM por S)

$$M_0 \leftarrow \sigma_{\text{Duration}>=180}(\text{M})$$

$$J \leftarrow M_0 \bowtie_{\text{M.MovieId} = \text{S.StreamId} \ \wedge \ \text{M.Year} \geq \text{YEAR(S.StreamDate)} - 5} \text{STREAM}$$

$$R \leftarrow \pi_{\text{Title,Year,StreamDate}}(J)$$

```
SELECT M.Title, M.Year, S.StreamDate
FROM MOVIE M JOIN STREAM S
    ON(M.MovieId = S.MovieId
        AND M.Year >= YEAR(S.StreamDate) - 5)
WHERE M.Duration >= 180;
```

- Neste caso não temos uma equi-junção, já que uma das condições de junção não é uma igualdade entre atributos.

Junção — exemplos (cont.)

```
SELECT M.Title, M.Year, S.StreamDate
FROM MOVIE M JOIN STREAM S
    ON(M.MovieId = S.MovieId
        AND M.Year >= YEAR(S.StreamDate) - 5)
WHERE M.Duration >= 180;
```

Title	Year	StreamDate
The Wolf of Wall Street	**2013**	**2017**-02-15 06:05:00
The Wolf of Wall Street	2013	2017-03-04 12:12:00
The Wolf of Wall Street	2013	2017-07-17 08:04:00
The Wolf of Wall Street	2013	2017-08-26 05:52:00
The Wolf of Wall Street	2013	2017-11-04 15:39:00
The Wolf of Wall Street	**2013**	**2018**-01-09 06:38:00
The Wolf of Wall Street	2013	2018-02-03 08:20:00
The Wolf of Wall Street	2013	2018-03-22 14:46:00
The Wolf of Wall Street	2013	2018-04-27 05:53:00
The Wolf of Wall Street	2013	2018-06-20 01:51:00
The Wolf of Wall Street	2013	2018-07-04 15:48:00
The Wolf of Wall Street	2013	2018-08-12 13:00:00
The Wolf of Wall Street	2013	2018-12-29 08:39:00
The Hateful Eight	**2015**	**2017**-02-06 02:53:00
The Hateful Eight	2015	2017-06-16 06:16:00
The Hateful Eight	**2015**	**2018**-03-28 22:24:00
The Hateful Eight	2015	2018-04-04 10:19:00
The Hateful Eight	2015	2018-05-22 09:18:00
The Hateful Eight	2015	2018-06-24 07:20:00
The Hateful Eight	2015	2018-07-26 13:44:00
The Hateful Eight	2015	2018-09-14 16:10:00
The Hateful Eight	2015	2018-09-26 21:47:00
The Hateful Eight	2015	2018-11-14 12:45:00

Junção – exemplos (cont.)

(abreviando MOVIE por M , MOVIE_GENRE por MG e GENRE por G)

$$J_1 \leftarrow \mathrm{M} \bowtie_{\mathrm{M.MovieId} = \mathrm{MG.MovieId}} \mathrm{MG}$$

$$J_2 \leftarrow J_1 \bowtie_{J_1.\mathrm{MovieId} = G.\mathrm{GenreId}} \mathrm{G}$$

$$R \leftarrow \sigma_{\mathrm{Year} \geq 2010 \ \wedge \ \mathrm{Label} = \text{'Horror'}} (J_2)$$

```
SELECT M.Title, M.Year
FROM MOVIE M JOIN MOVIE_GENRE MG JOIN GENRE G
    ON(M.MovieId = MG.MovieId AND MG.GenreId = G.GenreId)
WHERE M.Year >= 2010 AND G.Label = 'Horror';
```

- Muitas equi-junções são baseadas em atributos com nomes comuns. **São "naturais".**

Junção — exemplos (cont.)

```
SELECT M.Title, M.Year
FROM MOVIE M JOIN MOVIE_GENRE MG JOIN GENRE G
    ON(M.MovieId = MG.MovieId AND MG.GenreId = G.GenreId)
WHERE M.Year >= 2010 AND G.Label = 'Horror';
```

Title	Year
The Purge	2013
Dark Shadows	2012
Warm Bodies	2013
Sinister	2012
The Conjuring 2	2016
Don't Breathe	2016
10 Cloverfield Lane	2016
The Conjuring	2013
The Cabin in the Woods	2011
World War Z	2013
The Woman in Black	2012
Insidious	2010
It Follows	2014
Hansel & Gretel: Witch Hunters	2013
Split	2016
The Babadook	2014

16 rows in set (0.00 sec)

Junção natural

A **junção natural** de

$$R(A_1, \ldots, A_m) \quad \text{e} \quad S(B_1, \ldots, B_n)$$

é definida e denotada por

$$R * S = R \bowtie_C S$$

tal que:

(1) a condição de junção C exprime a igualdade entre cada **par de atributos das duas relações com o mesmo nome**

(2) a relação resultante **não duplica** os atributos comuns.

Em SQL: *R* NATURAL JOIN *S*

Exemplo anterior com junção natural (cont.)

(abreviando MOVIE por M , MOVIE_GENRE por MG e GENRE por G)

$$J_1 \leftarrow \mathrm{M} \ * \ \mathrm{MG}$$

$$J_2 \leftarrow J_1 \ * \ \mathrm{G}$$

$$R \leftarrow \sigma_{\mathrm{Year} \geq 2010 \ \wedge \ \mathrm{Label} = \text{'Horror'}} (J_2)$$

$\longleftrightarrow$

$$J_1 \leftarrow \mathrm{M} \bowtie_{\mathrm{M.MovieId} = \mathrm{MG.MovieId}} \mathrm{MG}$$

$$J_2 \leftarrow J_1 \bowtie_{J_1.\mathrm{GenreId} = G.\mathrm{GenreId}} \mathrm{G}$$

$$R \leftarrow \sigma_{\mathrm{Year} \geq 2010 \ \wedge \ \mathrm{Label} = \text{'Horror'}} (J_2)$$

```
SELECT M.Title, M.Year
FROM MOVIE M
  NATURAL JOIN MOVIE_GENRE
  NATURAL JOIN GENRE G
WHERE M.Year >= 2010
          AND G.Label = 'Horror';
```

$\longleftrightarrow$

```
SELECT M.Title, M.Year
FROM MOVIE M
        JOIN MOVIE_GENRE MG
        JOIN GENRE G
    ON(M.MovieId = MG.MovieId
        AND MG.GenreId = G.GenreId)
WHERE M.Year >= 2010 AND G.Label = 'Horror';
```

Junção natural restrita

Denotamos

$$R *_{A_1, \ldots, A_n} S$$

pela junção natural restrita a um sub-conjunto de atributos

$$A_1, \ \ldots \ , A_n$$

Em SQL: ***R*** **JOIN S USING(A1, …, AN)**

Junção natural restrita — exemplo

$$C \leftarrow \rho_{\text{Name/CName}}(\text{COUNTRY})$$
$$R \leftarrow \rho_{\text{Name/RName}}(\text{REGION})$$
$$J \leftarrow C *_{\text{RegionId}} R$$
$$P \leftarrow \pi_{\text{CName,RName}}(P_1)$$

```
SELECT C.Name AS CName,
       R.Name AS RName
FROM COUNTRY C
JOIN REGION R USING(RegionId);
```

CName	RName
Anguilla	Other countries
French Polynesia	Other countries
Greenland	Other countries
Nauru	Other countries
New Zealand	Other countries
Reunion	Other countries
Saint Vincent and the Grenadines	Other countries
Tonga	Other countries
Tuvalu	Other countries
Virgin Islands	Other countries
American Samoa	America
Argentina	America
Bolivia	America
Brazil	America

. . .

- Em contraste **COUNTRY NATURAL JOIN REGION** é uma relação vazia ... Porquê?

Junção externa

Para duas relações R e S

$$R(A_1, \dots, A_m) \qquad S(B_1, \dots, B_n)$$

definimos a **junção externa à esquerda**:

$$R \mathbin{⟕}_C S = R \bowtie_C S \;\cup\; \{(r[A_1], \dots, r[A_m], \text{NULL}, \dots, \text{NULL}) \mid r \in R \land \nexists s \in S : C(r, s)\}$$

Além dos dados de $R \bowtie_C S$ temos também (r, s) com s = (NULL, ..., NULL) para registos r em R que não encontram correspondência em S pela condição de junção C.

Em SQL: *R* LEFT JOIN S ON(C)

Junção externa — exemplos

```
SELECT S1.Name AS Supervised,
       S2.Name AS Supervisor
FROM STAFF S1
   LEFT JOIN STAFF S2
ON(S1.Supervisor = S2.StaffId)
ORDER BY Supervised, Supervisor;
```

O resultado para **João Pinto** não constaria da consulta alternativa com uma junção (**JOIN**) normal.

Supervised	Supervisor
Alexandra Romeu	Augusto Sousa
António Mota	João Pinto
Augusto Sousa	João Pinto
Eva Mendes	Xavier Semedo
Fábio Cruz	Augusto Sousa
Felícia Antunes	António Mota
Filipa Magalhães	José Santos
Filipa Mendes	Xavier Semedo
Gabriela Silva	António Mota
Gastão Pinto	António Mota
Joana Moreira	João Santorini
João Pinto	**NULL**
João Santorini	José Santos
José Santos	João Pinto
Maria Silva	Eva Mendes
Pedro Simões	Eva Mendes
Xavier Semedo	João Pinto

17 rows in set (0.01 sec)

Junção externa (cont.)

Analogamente definimos a **junção externa à direita**

$$R \overset{}{\bowtie\!\!\!\!\subset}_{C} S = R \bowtie_C S \cup \{(\text{NULL}, \ldots, \text{NULL}, s[B_1], \ldots, s[B_n],) \mid s \in s \wedge \nexists r \in R : C(r, s)\}$$

e finalmente a **junção externa completa**

$$R \overset{}{\supset\!\!\!\!\bowtie\!\!\!\!\subset}_{C} S = R \overset{}{\supset\!\!\!\!\bowtie}_{C} S \cup R \overset{}{\bowtie\!\!\!\!\subset}_{C} S$$

Em SQL:

R **RIGHT JOIN** **S** **ON(C)**

R **FULL JOIN** **S** **ON(C)**

Junção externa — exemplos (cont.)

```
SELECT S1.Name AS Supervised,
       S2.Name AS Supervisor
FROM STAFF S1
   RIGHT JOIN STAFF S2
ON(S1.Supervisor = S2.StaffId)
ORDER BY Supervised, Supervisor;
```

Supervised	Supervisor
NULL	Alexandra Romeu
NULL	Fábio Cruz
NULL	Felícia Antunes
NULL	Filipa Magalhães
NULL	Filipa Mendes
NULL	Gabriela Silva
NULL	Gastão Pinto
NULL	Joana Moreira
NULL	Maria Silva
NULL	Pedro Simões
Alexandra Romeu	Augusto Sousa
António Mota	João Pinto
Augusto Sousa	João Pinto
Eva Mendes	Xavier Semedo
Fábio Cruz	Augusto Sousa
Felícia Antunes	António Mota
Filipa Magalhães	José Santos
. . .	

Os resultados para funcionários que não supervisionam nenhum outro não constariam de uma consulta com uma junção (**JOIN**) normal.

Junção externa — exemplos (cont.)

```
SELECT S1.Name AS Supervised,
       S2.Name AS Supervisor
FROM STAFF S1
   FULL JOIN STAFF S2
ON(S1.Supervisor = S2.StaffId)
ORDER BY Supervised, Supervisor;
```

Supervised	Supervisor
NULL	Alexandra Romeu
NULL	Fábio Cruz
NULL	Felícia Antunes
NULL	Filipa Magalhães
NULL	Filipa Mendes
NULL	Gabriela Silva
NULL	Gastão Pinto
NULL	Joana Moreira
NULL	Maria Silva
NULL	Pedro Simões
Alexandra Romeu	Augusto Sousa
António Mota	João Pinto
Augusto Sousa	João Pinto
. . .	
João Pinto	NULL
João Santorini	José Santos
. . .	

MySQL não suporta **FULL JOIN** mas podemos fazer a união de consultas usando de **LEFT JOIN** e **RIGHT JOIN** (a seguir).

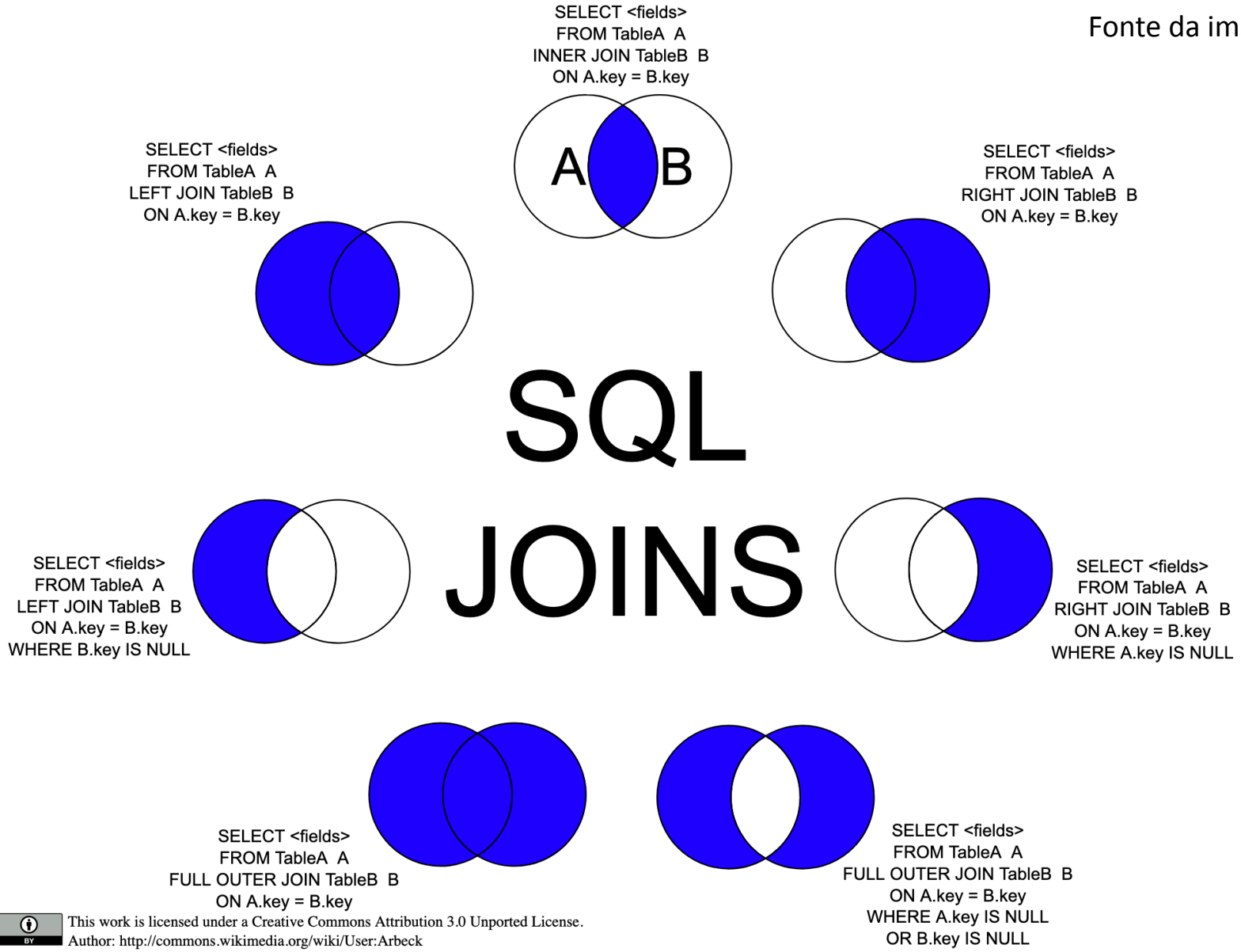
Junção externa — exemplos (cont.)

```
(SELECT S1.Name AS Supervised,
        S2.Name AS Supervisor
FROM STAFF S1
   LEFT JOIN STAFF S2
ON(S1.Supervisor = S2.StaffId))
UNION
(SELECT S1.Name AS Supervised,
        S2.Name AS Supervisor
FROM STAFF S1
   RIGHT JOIN STAFF S2
ON(S1.Supervisor = S2.StaffId))
ORDER BY Supervised, Supervisor;
```

Supervised	Supervisor
NULL	Alexandra Romeu
NULL	Fábio Cruz
NULL	Felícia Antunes
NULL	Filipa Magalhães
NULL	Filipa Mendes
NULL	Gabriela Silva
NULL	Gastão Pinto
NULL	Joana Moreira
NULL	Maria Silva
NULL	Pedro Simões
Alexandra Romeu	Augusto Sousa
António Mota	João Pinto
Augusto Sousa	João Pinto
. . .	
João Pinto	NULL
João Santorini	José Santos
. . .	

Junções em perspectiva

Fonte da imagem: Wikimedia



Sub-consultas

Consultas imbricadas

- **Forma geral**

```
SELECT … FROM … | DELETE FROM  … | UPDATE …
WHERE
  <EXPR> <OPERADOR> (
     SELECT …  /* Sub-consulta */
  )
…
```

- Podemos **imbricar consultas** usando na cláusula **WHERE** outra(s) consulta(s), chamadas **sub-consulta(s)**. Vários operadores podem ser usados para o encadeamento, como veremos a seguir.
- Podemos ter consultas imbricadas em associação a várias outras instruções como por exemplo **UPDATE** ou **DELETE** além de **SELECT**.

Operadores de sub-consulta

- **<EXPR> IN <CONSULTA>**
  - Verdadeiro se **<EXPR>** é devolvido por **<CONSULTA>**
- **<EXPR> NOT IN <CONSULTA>**
  - Verdadeiro se **<EXPR>** não é devolvido por **<CONSULTA>**
- **EXISTS <CONSULTA>**
  - Verdadeiro se **<CONSULTA>** devolve pelo menos um resultado.
- **<EXPR> <OPERADOR DE COMPARAÇÃO> <CONSULTA>**
  - Operadores de comparação (os usuais): **= <> > < >= <= …**
  - Verdadeiro se **<CONSULTA>** devolve um **único** resultado **R** e **<EXPR> <OPERADOR DE COMPARAÇÃO> R** se verifica.
- **<EXPR> <OPERADOR DE COMPARAÇÃO> ALL <CONSULTA>**
  - Similar ao caso anterior mas se a comparação é verdadeira para **todos** os resultados da consulta.
- **<EXPR> <OPERADOR DE COMPARAÇÃO> ANY <CONSULTA>**
  - Similar ao caso anterior mas se a comparação é verdadeira para **pelo menos um** os resultados da consulta.

Referência MySQL → "Subqueries"

Sub-consultas — exemplos

```
/* Filmes sem "streams" - variações */
SELECT Title
FROM MOVIE
WHERE MovieId NOT IN
 (SELECT MovieId FROM STREAM)
ORDER BY Title
LIMIT 5;
```

```
SELECT Title
FROM MOVIE M
WHERE NOT EXISTS
 (SELECT * FROM STREAM
  WHERE MovieId = M.MovieId)
ORDER BY Title
LIMIT 5;
```

```
SELECT Title
FROM MOVIE
WHERE MovieId <> ALL
 (SELECT MovieId FROM STREAM)
ORDER BY Title
LIMIT 5;
```

```
+-----------------------+
| Title                 |
+-----------------------+
| (500) Days of Summer  |
| 1408                  |
| 21                    |
| 300                   |
| 8 Mile                |
+-----------------------+
5 rows in set (0.00 sec)
```

Sub-consultas — exemplos (cont.)

```
INSERT INTO STREAM(CustomerId, MovieId, StreamDate, Charge)
VALUES
( (SELECT CustomerId FROM CUSTOMER WHERE Name = 'Aaron Selby'),
  (SELECT MovieId FROM MOVIE WHERE Title='Batman'),
  '2020-03-24 11:24:30',
  2.50 );

  SELECT *  FROM STREAM  WHERE
    CustomerId = (SELECT CustomerId FROM CUSTOMER
                  WHERE Name = 'Aaron Selby')
  AND
     MovieId = (SELECT MovieId FROM MOVIE
                WHERE Title='Batman');
```

StreamId	MovieId	CustomerId	StreamDate	Charge
10162	248	375	2020-03-24 11:24:30	2.50

Sub-consultas — exemplos (cont.)

```
INSERT INTO STREAM(CustomerId, MovieId, StreamDate, Charge)
VALUES
( (SELECT CustomerId FROM CUSTOMER WHERE Name = 'Aaron Selby'),
  (SELECT MovieId FROM MOVIE WHERE Title='Batman'),
  '2020-03-24 11:24:30',
  2.50 );

  SELECT *  FROM STREAM  WHERE
    CustomerId = (SELECT CustomerId FROM CUSTOMER
                  WHERE Name = 'Aaron Selby')
  AND
     MovieId = (SELECT MovieId FROM MOVIE
                WHERE Title='Batman');
```

StreamId	MovieId	CustomerId	StreamDate	Charge
10162	248	375	2020-03-24 11:24:30	2.50

Sub-consultas — exemplos (cont.)

```
/* Actualização de todos os streams feitos por clientes de Africa */
UPDATE STREAM
SET Charge = Charge + 0.5
WHERE
  CustomerId IN (
    SELECT CustomerId FROM CUSTOMER
    JOIN COUNTRY ON(CUSTOMER.Country = COUNTRY.NAME)
    JOIN REGION USING(RegionId)
    WHERE REGION.Name = 'Africa'
);
```

Obs.: podemos usar de forma equivalente **= ANY** em vez de **IN.**

```
Query OK, 1373 rows affected (0.05 sec)
```

```
/* Remoção de todos os streams feitos por clientes de Africa */
DELETE  FROM STREAM
WHERE
  CustomerId IN (
    SELECT CustomerId FROM CUSTOMER
    JOIN COUNTRY ON(CUSTOMER.Country = COUNTRY.NAME)
    JOIN REGION USING(RegionId)
    WHERE REGION.Name = 'Africa'
);
```