

Análise de Dados com Base em Processamento Massivo em Paralelo

Aula 7: Consultas OLAP usando Spark SQL

Cristina Dutra de Aguiar
ICMC/USP
cdac@icmc.usp.br



Agenda

- Linguagem SQL
- Apache Spark SQL

Linguagem SQL

- Estrutura Básica
- Agregação e Agrupamento
- Consultas com Vários Blocos

SQL (Structure Query Language)

- Desenvolvida a [partir de 1972](#)
 - Pesquisadores: Donald D. Chamberlin e Raymond F. Joyce
 - Local: laboratório de pesquisa da IBM em San Jose
 - Objetivo: linguagem de consulta para o sistema gerenciador de banco de dados (SGBD) relacional System R
- [Evolução contínua](#) desde a sua criação
 - Padronização: *American National Standard Institute* ([ANSI](#))
International Standard Organization ([ISO](#))
 - Versões: SQL-86, SQL-89, SQL-92, SQL:1999, SQL:2003, SQL:2008, [SQL:2016](#), ...

Características da Linguagem

- Voltada ao **modelo relacional**
- Descreve o problema ao invés da solução
 - Indica **quais** dados devem ser obtidos na resposta da consulta, e não **como** esses dados devem ser obtidos
- Amplamente utilizada
 - **Simplicidade**
 - **Facilidade de ser utilizada**
 - **Grande poder de consulta**

SGBDs Relacionais

ORACLE®
DATABASE

IBM Db2

Microsoft®
SQL Server™

teradata.

IBM
Informix database
software


PostgreSQL


MySQL®


MariaDB

Uso de SQL como Tecnologia de Consulta

NoSQL



Serviços na Nuvem

Família SQL
do Azure



AWS Athena



Google
BigQuery

Composição do SQL

- Linguagem de **definição** de dados (DDL)
- Linguagem de **manipulação** de dados (DML)
 - Inserção, remoção e atualização dos dados
 - Consulta
- Linguagem para
 - Definição de **visões**
 - Especificação de **restrições de integridade**
 - Autorização relacionada aos **diretos de acesso** para relações e visões
- Linguagem de **transação** de dados

Linguagem de
Consulta

Comando SELECT

```
SELECT <lista de atributos e funções>  
FROM <lista de relações>  
[ WHERE predicado de seleção ]  
[ GROUP BY <atributos de agrupamento> ]  
[ HAVING <condição para agrupamentos> ]  
[ ORDER BY <lista de atributos> ]
```

Cláusula SELECT

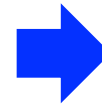
- Especifica **quais dados** são mostrados como resposta
 - Um ou mais **atributos**
 - Uma ou mais **funções**
 - Separados por vírgula
- Atributos
 - Devem estar presentes nas relações especificadas na cláusula FROM

Exemplo de Cláusula SELECT

funcionario

funcPK	funcMatricula	funcNome	funcSexo	funcDataNascimento	funcDiaNascimento	...
1	M-1	ALINE ALMEIDA	F	1/1/1990	1	...
2	M-2	ARAO ALVES	M	2/2/1990	2	...
3	M-3	ARON ANDRADE	M	3/3/1990	3	...
4	M-4	ADA BARBOSA	F	4/4/1990	4	...
5	M-5	ABADE BATISTA	M	5/5/1990	5	...
6	M-6	ABADI BARROS	M	6/6/1990	6	...
...

SELECT funcMatricula, funcNome
FROM funcionario



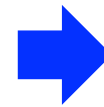
funcMatricula	funcNome
M-1	ALINE ALMEIDA
M-2	ARAO ALVES
M-3	ARON ANDRADE
M-4	ADA BARBOSA
M-5	ABADE BATISTA
M-6	ABADI BARROS
...	...

Exemplo de Cláusula SELECT

funcionario

funcPK	funcMatricula	funcNome	funcSexo	funcDataNascimento	funcDiaNascimento	...
1	M-1	ALINE ALMEIDA	F	1/1/1990	1	...
2	M-2	ARAO ALVES	M	2/2/1990	2	...
3	M-3	ARON ANDRADE	M	3/3/1990	3	...
4	M-4	ADA BARBOSA	F	4/4/1990	4	...
5	M-5	ABADE BATISTA	M	5/5/1990	5	...
6	M-6	ABADI BARROS	M	6/6/1990	6	...
...

SELECT funcSexo
FROM funcionario



funcSexo
F
M
M
F
M
M
...

Cláusula FROM

- Especifica **uma ou mais relações**
 - Contêm os dados solicitados na consulta
 - Devem ser separadas por vírgula
- Realiza um **produto cartesiano** das relações
 - Combina quaisquer tuplas, **independentemente da integridade referencial** existente entre elas
 - Produz tuplas que representam **todas as combinações** de tuplas possíveis entre as relações participantes

mesmos nomes de atributos que aparecem nas duas relações devem ser identificados por **nomeRelação.nomeAtributo**

Exemplo de Cláusula FROM

funcionario

funcPK	funcMatricula	funcNome	...
1	M-1	ALINE ALMEIDA	...
2	M-2	ARAO ALVES	...
...

pagamento

dataPK	funcPK	equipePK	...
1	1	7	...
1	3	2	...
2	6	7	...
...

SELECT funcionario.funcPK,
funcMatricula,
funcNome,
dataPK,
pagamento.funcPK
FROM funcionario,
pagamento



funcionario .funcPK	funcMatricula	funcNome	dataPK	pagamento .funcPK
1	M-1	ALINE ALMEIDA	1	1
1	M-1	ALINE ALMEIDA	1	3
1	M-1	ALINE ALMEIDA	2	6
2	M-2	ARAO ALVES	1	1
2	M-2	ARAO ALVES	1	3
2	M-2	ARAO ALVES	2	6
...

Cláusula WHERE

- Especifica o **predicado que seleciona** as tuplas
 - Composto por condições
- Condições de seleção
 - Devem ser definidas sobre os atributos das relações na cláusula FROM
 - Incluem condições de junção quando necessário
- Sintaxe de cada condição
 - **<atributo> <operador> <valor | atributo | lista de valores | NULL>**

Operadores de Comparação

igual a	=
maior que	>
menor que	<
entre dois valores v_1 e v_2	BETWEEN v_1 AND v_2
lista de atributos	IN
valores nulos (NULL)	IS IS NOT

diferente de	<>
maior ou igual a	>=
menor ou igual a	<=
cadeia de caracteres	LIKE NOT LIKE
% : substitui qualquer <i>string</i>	
_ : substitui qualquer caractere	

operadores sensíveis ao caso

Predicado de Seleção

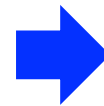
- Sintaxe: $\langle \text{condição}_1 \rangle \theta \langle \text{condição}_2 \rangle \theta \dots \theta \langle \text{condição}_n \rangle$
- Operadores lógicos booleanos (θ)
 - conjunção: AND
 - disjunção: OR
 - negação: NOT
- Precedência de operadores
 - NOT; operadores de comparação; AND; OR

Exemplo de Cláusula WHERE

funcionario

funcPK	funcMatricula	funcNome	funcSexo	funcDataNascimento	funcDiaNascimento	...
1	M-1	ALINE ALMEIDA	F	1/1/1990	1	...
2	M-2	ARAO ALVES	M	2/2/1990	2	...
3	M-3	ARON ANDRADE	M	3/3/1990	3	...
4	M-4	ADA BARBOSA	F	4/4/1990	4	...
5	M-5	ABADE BATISTA	M	5/5/1990	5	...
6	M-6	ABADI BARROS	M	6/6/1990	6	...
...

SELECT funcPK, funcMatricula, funcNome
FROM funcionario
WHERE funcPK BETWEEN 2 AND 4



funcPK	funcMatricula	funcNome
2	M-2	ARAO ALVES
3	M-3	ARON ANDRADE
4	M-4	ADA BARBOSA

Cláusula ORDER BY

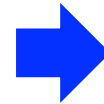
- Ordena as tuplas que aparecem no resultado da consulta
 - asc (padrão): ordem ascendente
 - desc: ordem descendente
- Ordenação pode ser especificada em vários atributos
 - Ordenação referente ao primeiro atributo é prioritária
 - Se houver valores repetidos para o primeiro atributo, então é utilizada a ordenação referente ao segundo atributo
 - E assim por diante

Exemplo de Cláusula ORDER BY

funcionario

funcPK	funcMatricula	funcNome	funcSexo	funcDataNascimento	funcDiaNascimento	...
1	M-1	ALINE ALMEIDA	F	1/1/1990	1	...
2	M-2	ARAO ALVES	M	2/2/1990	2	...
3	M-3	ARON ANDRADE	M	3/3/1990	3	...
4	M-4	ADA BARBOSA	F	4/4/1990	4	...
5	M-5	ABADE BATISTA	M	5/5/1990	5	...
6	M-6	ABADI BARROS	M	6/6/1990	6	...
...

SELECT funcPK, funcMatricula, funcNome
FROM funcionario
WHERE funcPK BETWEEN 2 AND 4
ORDER BY funcPK DESC



funcPK	funcMatricula	funcNome
4	M-4	ADA BARBOSA
3	M-3	ARON ANDRADE
2	M-2	ARAO ALVES

Cláusula AS

- Renomeia nomes de atributos e relações
 - Sintaxe: `nome_antigo AS nome_novo`
- Atributos
 - Deve aparecer na cláusula `SELECT`
 - Útil para a visualização semântica das respostas
- Relações
 - Deve aparecer na cláusula `FROM`
 - Útil para simplificar os nomes das relações e também quando uma mesma relação é usada mais do que uma vez na consulta

Exemplo de Cláusula AS

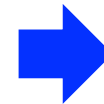
funcionario

funcPK	funcMatricula	funcNome	...
1	M-1	ALINE ALMEIDA	...
2	M-2	ARAO ALVES	...
...

pagamento

dataPK	funcPK	equipePK	...
1	1	7	...
1	3	2	...
2	6	7	...
...

```
SELECT f.funcPK AS `PK de Funcionário`,  
       p.funcPK AS `PK de Pagamento`  
FROM funcionario AS f,  
     pagamento AS p
```



PK de Funcionário	PK de Pagamento
1	1
1	3
1	6
2	1
2	3
2	6
...	...


Resultado da Consulta

- Ordem de **apresentação dos atributos**
 - Ordem dos atributos na cláusula **SELECT**
- Ordem de **apresentação dos dados**
 - Ascendente ou descendente de acordo com a cláusula **ORDER BY**
 - Sem ordenação
- Eliminação de **valores repetidos**
 - Cláusula **SELECT DISTINCT**

Junção (\bowtie)

- Concatena **tuplas relacionadas** de duas relações
 - Com base na **integridade referencial**, ou seja, nos pares (chave estrangeira, chave primária)
- Passos
 - Forma um **produto cartesiano** das relações
 - Faz uma **seleção forçando a igualdade** sobre os atributos que compõem a integridade referencial (e quaisquer outros pares de atributos especificados)

condição de junção



Condição de Junção

- Sintaxe: $\langle \text{condição}_1 \rangle \text{ AND } \langle \text{condição}_2 \rangle \text{ AND } \dots \text{ AND } \langle \text{condição}_n \rangle$
- Sintaxe de cada condição
 - $\langle \text{atributo da primeira relação} \rangle \theta \langle \text{atributo da segunda relação} \rangle$
 - Theta join: $\theta = \{ =, >, >=, <, <=, <> \}$
- Na prática
 - Equijoin: $\theta = \{ = \}$

Especificando Junção em SQL

- Antes de SQL-92
 - Especificado nas cláusulas **FROM** e **WHERE**
 - **FROM** deve possuir mais do que uma relação
 - **WHERE** deve incluir as condições de junção
- A partir de SQL-92
 - Especificado na cláusula **FROM**
 - Introdução de novas cláusulas **JOIN ... ON <condição de junção>**
 - **[INNER] JOIN**
 - **LEFT [OUTER] JOIN, RIGHT [OUTER] JOIN, FULL [OUTER] JOIN**

mesmos nomes de atributos
que aparecem nas duas
relações devem ser
identificados por
nomeRelação.nomeAtributo

Cláusula [INNER] JOIN ($R \bowtie S$)

- Mantém somente as tuplas de R e S que têm correspondência
- Valores dos atributos das tuplas resultantes
 - Valores obtidos de R e de S

```
SELECT *  
FROM funcionario, pagamento  
WHERE funcionario.funcPK = pagamento.funcPK
```

```
SELECT *  
FROM funcionario JOIN pagamento ON funcionario.funcPK = pagamento.funcPK
```

Exemplo de Cláusula [INNER] JOIN

funcionario

funcPK	funcMatricula	funcNome	...
1	M-1	ALINE ALMEIDA	...
2	M-2	ARAO ALVES	...
...

pagamento

dataPK	funcPK	equipePK	...
1	1	7	...
1	3	2	...
2	6	7	...
...

SELECT funcionario.funcPK, funcMatricula, funcNome, dataPK, pagamento.funcPK, equipePK
FROM funcionario JOIN pagamento ON funcionario.funcPK = pagamento.funcPK



funcionario.funcPK	funcMatricula	funcNome	dataPK	pagamento.funcPK	equipePK
1	M-1	ALINE ALMEIDA	1	1	7
...

Cláusula **LEFT [OUTER] JOIN (R ⋈ S)**

- Mantém todas as **tuplas de R**
- Valores dos atributos das tuplas resultantes
 - Valores obtidos de **R** e de **S** quando existe correspondência
 - Valores obtidos de **R** e valores **nulo** quando não existe correspondência

Exemplo de Cláusula LEFT [OUTER] JOIN

funcionario

funcPK	funcMatricula	funcNome	...
1	M-1	ALINE ALMEIDA	...
2	M-2	ARAO ALVES	...
...

pagamento

dataPK	funcPK	equipePK	...
1	1	7	...
1	3	2	...
2	6	7	...
...

SELECT funcionario.funcPK, funcMatricula, funcNome, dataPK, pagamento.funcPK, equipePK
FROM funcionario LEFT JOIN pagamento ON funcionario.funcPK = pagamento.funcPK



funcionario.funcPK	funcMatricula	funcNome	dataPK	pagamento.funcPK	equipePK
1	M-1	ALINE ALMEIDA	1	1	7
2	M-2	ARAO ALVES	null	null	null
...

Cláusula RIGHT [OUTER] JOIN ($R \bowtie S$)

- Mantém todas as tuplas de S
- Valores dos atributos das tuplas resultantes
 - Valores obtidos de R e de S quando existe correspondência
 - Valores obtidos de S e valores nulo quando não existe correspondência

Exemplo de Cláusula RIGHT [OUTER] JOIN

funcionario

funcPK	funcMatricula	funcNome	...
1	M-1	ALINE ALMEIDA	...
2	M-2	ARAO ALVES	...
...

pagamento

dataPK	funcPK	equipePK	...
1	1	7	...
1	3	2	...
2	6	7	...
...

SELECT funcionario.funcPK, funcMatricula, funcNome, dataPK, pagamento.funcPK, equipePK
FROM funcionario RIGHT JOIN pagamento ON funcionario.funcPK = pagamento.funcPK



funcionario.funcPK	funcMatricula	funcNome	dataPK	pagamento.funcPK	equipePK
1	M-1	ALINE ALMEIDA	1	1	7
null	null	null	1	3	2
null	null	null	2	6	7
...

Cláusula **FULL [OUTER] JOIN ($R \bowtie S$)**

- Mantém todas as **tuplas de R e S mesmo sem correspondência**
- Valores dos atributos das tuplas resultantes
 - Valores obtidos de **R** e de **S** quando existe correspondência
 - Valores obtidos de **R** e valores **nulo** quando não existe correspondência
 - Valores obtidos de **S** e valores **nulo** quando não existe correspondência

Exemplo de Cláusula FULL [OUTER] JOIN

funcionario

funcPK	funcMatricula	funcNome	...
1	M-1	ALINE ALMEIDA	...
2	M-2	ARAO ALVES	...
...

pagamento

dataPK	funcPK	equipePK	...
1	1	7	...
1	3	2	...
2	6	7	...
...

SELECT funcionario.funcPK, funcMatricula, funcNome, dataPK, pagamento.funcPK, equipePK
FROM funcionario FULL JOIN pagamento ON funcionario.funcPK = pagamento.funcPK



funcionario.funcPK	funcMatricula	funcNome	dataPK	pagamento.funcPK	equipePK
1	M-1	ALINE ALMEIDA	1	1	7
2	M-2	ARAO ALVES	null	null	null
null	null	null	1	3	2
null	null	null	2	6	7
...

Processamento Lógico da Consulta

- 3 SELECT <lista de atributos e funções>
- 1 FROM <lista de relações>
- 2 WHERE predicado de seleção
- 4 ORDER BY <lista de atributos>

Processamento Lógico da Consulta

- 3 SELECT <lista de atributos e funções>
DISTINCT
- 1 FROM <lista de relações>
ON
JOIN
- 2 WHERE predicado de seleção
- 4 ORDER BY <lista de atributos>

Linguagem SQL

- Estrutura Básica
- Agregação e Agrupamento
- Consultas com Vários Blocos

Funções de Agregação

- Característica
 - Recebem uma coleção de valores como entrada
 - Retornam um único valor como saída
- Funções e resultado retornado
 - `SUM()`: soma
 - `MIN()`: menor
 - `MAX()`: maior
 - `AVG()`: média
 - `COUNT()`: quantidade de tuplas

função de agregação (`DISTINCT ...`)
realiza a função de
agregação considerando apenas
valores distintos

Exemplo de Funções de Agregação

pagamento

dataPK	funcPK	equipePK	cargoPK	salario	quantidadeLancamento
1	1	7	112	2.226,66	1
1	3	2	74	9.169,90	1
2	6	7	112	2.226,66	1
7	1	3	23	3.828,90	1
...

SELECT

SUM(salario) AS soma,

MIN(salario) AS menor,

MAX(salario) AS maior,

AVG(salario) AS media,

COUNT(salario) AS quantidade,

COUNT(DISTINCT salario) AS diferente

FROM pagamento



soma	menor	maior	media	quantidade	diferente
17.452,12	2.226,66	9.169,90	4.363,03	4	3

Cláusula GROUP BY

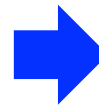
- Aplica uma função de agregação a um grupo de conjunto de tuplas
 - Para cada grupo de conjunto de tuplas
 - Retorna um único valor
- Atributos de agrupamento
 - Usados na cláusula GROUP BY
 - Têm que ser especificados na cláusula SELECT

Exemplo de Cláusula GROUP BY

pagamento

dataPK	funcPK	equipePK	cargoPK	salario	quantidadeLancamento
1	1	7	112	2.226,66	1
1	3	2	74	9.169,90	1
2	6	7	112	2.226,66	1
7	1	3	23	3.828,90	1
...

```
SELECT cargoPK,  
       SUM(salario) AS soma  
FROM pagamento  
GROUP BY cargoPK  
ORDER BY cargoPK
```



cargoPK	soma
23	3.828,90
74	9.169,90
112	4.453,32

Cláusula HAVING

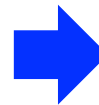
- Especifica **condições de seleção** para as **funções de agregação**
 - Aplicada sobre os grupos de conjunto de tuplas
- Recupera os **valores dos grupos** de conjunto de tuplas
 - Desde que esses valores satisfaçam às condições de seleção
- Usada conjuntamente com a cláusula **GROUP BY**
 - Quando são definidos atributos de agrupamento

Exemplo de Cláusula HAVING

pagamento

dataPK	funcPK	equipePK	cargoPK	salario	quantidadeLancamento
1	1	7	112	2.226,66	1
1	3	2	74	9.169,90	1
2	6	7	112	2.226,66	1
7	1	3	23	3.828,90	1
...

```
SELECT cargoPK,  
       SUM(salario) AS soma  
FROM pagamento  
GROUP BY cargoPK  
HAVING SUM(salario) > 5000  
ORDER BY cargoPK
```



cargoPK	soma
74	9.169,90

Estendendo a Cláusula GROUP BY

- Objetivo
 - Gerar diferentes níveis de agregação dos dados
- Utilidade
 - Aplicações de *data warehousing*
- Extensões
 - ROLLUP
 - CUBE
 - GROUPING SETS

Extensão ROLLUP

- Cria subtotais
 - A partir do nível mais detalhado até o nível menos detalhado
 - Para combinações dos atributos da lista de agrupamento de acordo com a ordem desses atributos
- Processamento
 - Recebe como argumento uma lista dos n atributos usados na criação dos subtotais
 - Cria, de forma progressiva, subtotais de nível mais alto, considerando os n atributos da esquerda para a direita
 - Produz como resultado $n + 1$ níveis de agregação

Exemplo de Extensão ROLLUP

pagamento

dataPK	funcPK	equipePK	cargoPK	salario	quantidadeLancamento
1	1	7	112	2.226,66	1
1	3	2	74	9.169,90	1
2	1	7	112	2.226,66	1
...

```
SELECT dataPK, funcPK, equipePK, cargoPK,  
       SUM(salario) AS `gastos em salário`,  
       SUM(quantidadeLancamento) AS `lançamentos`  
FROM pagamento  
GROUP BY ROLLUP (dataPK, funcPK, equipePK, cargoPK)
```

Exemplo de Extensão ROLLUP

GROUP BY **ROLLUP** (dataPK, funcPK, equipePK, cargoPK)

dataPK	funcPK	equipePK	cargoPK	gastos em salário	lançamentos
1	1	7	112	2.226,66	1
1	3	2	74	9.169,90	1
2	1	7	112	2.226,66	1
1	1	7	null	2.226,66	1
1	3	2	null	9.169,90	1
2	1	7	null	2.226,66	1
1	1	null	null	2.226,66	1
1	3	null	null	9.169,90	1
2	1	null	null	2.226,66	1
1	null	null	null	11.396,56	2
2	null	null	null	2.226,66	1
null	null	null	null	13.623,22	3

Exemplo de Extensão ROLLUP

GROUP BY **ROLLUP** (dataPK, funcPK, equipePK, cargoPK)

dataPK	funcPK	equipePK	cargoPK	gastos em salário	lançamentos
1	1	7	112	2.226,66	1
1	3	2	74	9.169,90	1
2	1	7	112	2.226,66	1
1	1	7	112	2.226,66	1
1	3	2	74	9.169,90	1
2	1	7	112	2.226,66	1
1	1	null	null	2.226,66	1
1	3	null	null	9.169,90	1
2	1	null	null	2.226,66	1
1	null	null	null	11.396,56	2
2	null	null	null	2.226,66	1
null	null	null	null	13.623,22	3

n + 1 níveis =
4 + 1 = 5

Extensão CUBE

- Cria subtotaís
 - A partir do nível mais detalhado até o nível menos detalhado
 - Para **todas as combinações dos atributos da lista de agrupamento**
- Processamento
 - Recebe como argumento uma lista dos **n** atributos usados na criação dos subtotaís
 - Cria, de forma progressiva, subtotaís de nível mais alto, considerando todas as combinações dos **n** atributos
 - Produz como resultado **2^n**

Exemplo de Extensão CUBE

pagamento

dataPK	funcPK	equipePK	cargoPK	salario	quantidadeLancamento
1	1	7	112	2.226,66	1
1	3	2	74	9.169,90	1
2	1	7	112	2.226,66	1
...

```
SELECT dataPK, funcPK, equipePK, cargoPK,  
       SUM(salario) AS `gastos em salário`,  
       SUM(quantidadeLancamento) AS `lançamentos`  
FROM pagamento  
GROUP BY CUBE (dataPK, funcPK, equipePK, cargoPK)
```

Exemplo de Extensão CUBE

GROUP BY **CUBE** (dataPK, funcPK, equipePK, cargoPK)

dataPK	funcPK	equipePK	cargoPK	gastos em salário	lançamentos
1	1	7	112	2.226,66	1
1	3	2	74	9.169,90	1
2	1	7	112	2.226,66	1
1	1	7	null	2.226,66	1
1	3	2	null	9.169,90	1
2	1	7	null	2.226,66	1
1	1	null	112	2.226,66	1
1	3	null	74	9.169,90	1
2	1	null	112	2.226,66	1
1	null	7	112	2.226,66	1
1	null	2	74	9.169,90	1
2	null	7	112	2.226,66	1
null	1	7	112	4.453,32	2
null	3	2	74	9.169,90	1
1	1	null	null	2.226,66	1
1	3	null	null	9.169,90	1
2	1	null	null	2.226,66	1
null	1	7	null	4.453,32	2
null	3	2	null	9.169,90	1

dataPK	funcPK	equipePK	cargoPK	gastos em salário	lançamentos
1	null	7	null	2.226,66	1
1	null	2	null	9.169,90	1
2	null	7	null	2.226,66	1
1	null	null	112	2.226,66	1
1	null	null	74	9.169,90	1
2	null	null	112	2.226,66	1
null	1	null	112	4.453,32	2
null	3	null	74	9.169,90	1
null	null	7	112	4.453,32	2
null	null	2	74	9.169,90	1
null	null	null	112	4.453,32	2
null	null	null	74	9.169,90	1
null	null	7	null	4.453,32	2
null	null	2	null	9.169,90	1
null	1	null	null	4.453,32	2
null	3	null	null	9.169,90	1
1	null	null	null	11.396,56	2
2	null	null	null	2.226,66	1
null	null	null	null	13.623,22	3

Exemplo de Extensão CUBE

GROUP BY **CUBE** (dataPK, funcPK, equipePK, cargoPK)

dataPK	funcPK	equipePK	cargoPK	gastos em salário	lançamentos	dataPK	funcPK	equipePK	cargoPK	gastos em salário	lançamentos
1	1	7	112	2.226,66	1	1	null	7	null	2.226,66	1
1	3	2	74	9.169,90	1	1	null	2	null	9.169,90	1
2	1	7	112	2.226,66	1	2	null	7	null	2.226,66	1
1	1	7	null	2.226,66	1	1	null	null	112	2.226,66	1
1	3	2	null	9.169,90	1	1	null	null	74	9.169,90	1
2	1	7	null	2.226,66	1	2	null	null	112	2.226,66	1
1	1	null	112	2.226,66		null	1	null	112	4.453,32	2
1	3	null	74	9.169,90				null	74	9.169,90	1
2	1	null	112	2.226,66				7	112	4.453,32	2
1	null	7	112	2.226,66				2	74	9.169,90	1
1	null	2	74	9.169,90				null	112	4.453,32	2
2	null	7	112	2.226,66				null	74	9.169,90	1
null	1	7	112	4.453,32	2			7	null	4.453,32	2
null	3	2	74	9.169,90	1	null	null	2	null	9.169,90	1
1	1	null	null	2.226,66	1	null	1	null	null	4.453,32	2
1	3	null	null	9.169,90	1	null	3	null	null	9.169,90	1
2	1	null	null	2.226,66	1	1	null	null	null	11.396,56	2
null	1	7	null	4.453,32	2	2	null	null	null	2.226,66	1
null	3	2	null	9.169,90	1	null	null	null	null	13.623,22	3

2ⁿ níveis
= 2⁴ = 16

Extensão GROUPING SETS

- Cria subtotais
 - Para quaisquer combinações de atributos desejados
- Subtotais criados
 - Definidos em uma lista que especifica cada nível de agregação desejado
 - Podem ser equivalentes às extensões ROLLUP e CUBE
 - Podem ser referentes a outros subtotais

Exemplo de Extensão GROUPING SETS

```
SELECT dataPK, funcPK, equipePK, cargoPK,  
       SUM(salario) AS `gastos em salário`,  
       SUM(quantidadeLancamento) AS `lançamentos`  
FROM pagamento  
GROUP BY ROLLUP (dataPK, funcPK, equipePK, cargoPK)
```

```
SELECT dataPK, funcPK, equipePK, cargoPK,  
       SUM(salario) AS `gastos em salário`,  
       SUM(quantidadeLancamento) AS `lançamentos`  
FROM pagamento  
GROUP BY GROUPING SETS ((dataPK, funcPK, equipePK, cargoPK),  
                           (dataPK, funcPK, equipePK), (dataPK, funcPK), (dataPK), ())
```

Exemplo de Extensão GROUPING SETS

```
SELECT dataPK, funcPK, equipePK, cargoPK,  
       SUM(salario) AS `gastos em salário`,  
       SUM(quantidadeLancamento) AS `lançamentos`  
FROM pagamento  
GROUP BY CUBE (dataPK, funcPK, equipePK, cargoPK)
```

```
SELECT dataPK, funcPK, equipePK, cargoPK,  
       SUM(salario) AS `gastos em salário`,  
       SUM(quantidadeLancamento) AS `lançamentos`  
FROM pagamento  
GROUP BY GROUPING SETS ((dataPK, funcPK, equipePK, cargoPK), (dataPK, funcPK, equipePK),  
                           (dataPK, funcPK, cargoPK), (dataPK, equipePK, cargoPK), (funcPK, equipePK, cargoPK),  
                           (dataPK, funcPK), (dataPK, equipePK), (dataPK, cargoPK), (funcPK, equipePK),  
                           (funcPK, cargoPK), (equipePK, cargoPK), (dataPK), (funcPK), (equipePK), (cargoPK), ())
```

Processamento Lógico da Consulta

- 5 SELECT <lista de atributos e funções>
- 1 FROM <lista de relações>
- 2 WHERE predicado de seleção
- 3 GROUP BY <atributos de agrupamento>
- 4 HAVING <condição para agrupamentos>
- 6 ORDER BY <lista de atributos>

Processamento Lógico da Consulta

- 5 SELECT <lista de atributos e funções>
DISTINCT
- 1 FROM <lista de relações>
JOIN
ON
- 2 WHERE predicado de seleção
- 3 GROUP BY <atributos de agrupamento>
ROLLUP/CUBE/GROUPING SETS
- 4 HAVING <condição para agrupamentos>
- 6 ORDER BY <lista de atributos>

Linguagem SQL

- Estrutura Básica
- Agregação e Agrupamento
- Consultas com Vários Blocos

Bloco de Consulta

- Unidade básica
 - Contém uma **única expressão SELECT-FROM-WHERE**
- Tipos de consulta com vários blocos
 - **Operações sobre conjuntos**
 - **Subconsultas aninhadas**
 - **Consultas complexas**

Operações sobre Conjuntos

- Operações

- União
- Intersecção
- Diferença

Duas relações R e S são compatíveis se:

- possuem o mesmo número n de atributos
- os domínios do i -ésimo atributo de R e do i -ésimo atributo de S são os mesmos ($1 \leq i \leq n$)

- Características

- Atuam sobre relações compatíveis
- Eliminam as tuplas repetidas do resultado

Descrição das Operações

- União entre R e S
 - Resultado contém todas as tuplas pertencentes a R, a S, ou a ambas R e S
 - Operação **UNION**
- Intersecção entre R e S
 - Resultado contém todas as tuplas pertencentes a ambas R e S
 - Operação **INTERSECT**
- Diferença entre R e S
 - Resultado contém todas as tuplas pertencentes a R que não pertencem a S
 - Operação **MINUS/EXCEPT**

Exemplo da Operação UNION

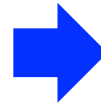
funcionario (funcPK BETWEEN 1 AND 2)

funcPK	funcMatricula	funcNome	...
1	M-1	ALINE ALMEIDA	...
2	M-2	ARAO ALVES	...

pagamento (funcPK IN (1, 3, 6))

dataPK	funcPK	equipePK	...
1	1	7	...
1	3	2	...
2	6	7	...

```
SELECT funcPK  
FROM funcionario  
UNION  
SELECT funcPK  
FROM pagamento
```



funcPK
1
2
3
6

Exemplo da Operação INTERSECT

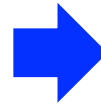
funcionario (funcPK BETWEEN 1 AND 2)

funcPK	funcMatricula	funcNome	...
1	M-1	ALINE ALMEIDA	...
2	M-2	ARAO ALVES	...

pagamento (funcPK IN (1, 3, 6))

dataPK	funcPK	equipePK	...
1	1	7	...
1	3	2	...
2	6	7	...

```
SELECT funcPK  
FROM funcionario  
INTERSECT  
SELECT funcPK  
FROM pagamento
```



funcPK
1

Exemplo da Operação MINUS

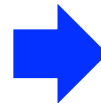
funcionario (funcPK BETWEEN 1 AND 2)

funcPK	funcMatricula	funcNome	...
1	M-1	ALINE ALMEIDA	...
2	M-2	ARAO ALVES	...

pagamento (funcPK IN (1, 3, 6))

dataPK	funcPK	equipePK	...
1	1	7	...
1	3	2	...
2	6	7	...

```
SELECT funcPK  
FROM funcionario  
MINUS  
SELECT funcPK  
FROM pagamento
```



funcPK
2

Subconsultas Aninhadas

- Subconsulta
 - Bloco de consulta **aninhado** dentro de outra consulta
- Aplicações mais comuns
 - Testes para membros de conjuntos (**IN** e **NOT IN**)
 - Cardinalidade de conjuntos (**EXISTS** e **NOT EXISTS**)
- **Conjunto: coleção de valores produzidos pela subconsulta**

Membros de Conjuntos

- Conectivo **IN**
 - Testa se um ou mais atributos **são membros do conjunto**
 - **WHERE** (atributo₁, ... atributo_n) **IN**
(SELECT atributo₁, ..., atributo_n
FROM)
- Conectivo **NOT IN**
 - Testa se um atributo ou mais atributos **não são membros do conjunto**
 - **WHERE** (atributo₁, ... atributo_n) **NOT IN**
(SELECT atributo₁, ..., atributo_n
FROM)

Exemplo do Conectivo IN

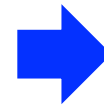
funcionario

funcPK	funcMatricula	funcNome	...
1	M-1	ALINE ALMEIDA	...
2	M-2	ARAO ALVES	...
...

pagamento

dataPK	funcPK	equipePK	...
1	1	7	...
1	3	2	...
2	6	7	...
...

```
SELECT funcPK, funcMatricula, funcNome
FROM funcionario
WHERE funcPK IN
  ( SELECT funcPK
    FROM pagamento )
```



funcPK	funcMatricula	funcNome
1	M-1	ALINE ALMEIDA
...

Cardinalidade de Conjuntos

- Construção EXISTS
 - A condição é verdadeira quando o conjunto retornado não for vazio
 - WHERE EXISTS
(SELECT ...
FROM ...)
- Construção NOT EXISTS
 - A condição é verdadeira quando o conjunto retornado for vazio
 - WHERE NOT EXISTS
(SELECT ...
FROM ...)

Exemplo da Construção EXISTS

funcionario

funcPK	funcMatricula	funcNome	...
1	M-1	ALINE ALMEIDA	...
2	M-2	ARAO ALVES	...
...

pagamento

dataPK	funcPK	equipePK	...
1	1	7	...
1	3	2	...
2	6	7	...
...

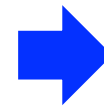
SELECT funcPK, funcMatricula, funcNome
FROM funcionario
WHERE EXISTS

(SELECT *

FROM pagamento

WHERE funcionario.funcPK = pagamento.funcPK

)



funcPK	funcMatricula	funcNome
1	M-1	ALINE ALMEIDA
...

Consultas Complexas

- Consultas difíceis ou impossíveis de se escrever
 - Usando apenas um único bloco de consulta
 - Usando união, intersecção ou diferença entre blocos de consulta
- Relação derivada
 - Subconsulta especificada na cláusula **FROM**
 - Deve possuir um **nome** diferente dos nomes das relações base
 - Pode possuir uma lista de **atributos renomeados**

Exemplo de Consulta Complexa

pagamento

dataPK	funcPK	equipePK	cargoPK	salario	...
1	1	7	112	2.226,66	...
1	3	2	74	9.169,90	...
2	6	7	112	2.226,66	...
7	1	3	23	3.828,90	...

data

dataPK	...	dataAno
1	...	2016
2	...	2016
3	...	2017
7	...	2020

negociacao

dataPK	equipePK	clientePK	receita	...
1	7	2	5.245,00	...
3	3	3	5.431,23	...
7	1	3	5.789,00	...
1	3	4	9.323,00	...

```
SELECT anoGasto, gasto, ganho  
FROM
```

```
( SELECT dataAno, SUM(salario)  
  FROM data, pagamento  
 WHERE data.dataPK = pagamento.dataPK  
 GROUP BY dataAno  
 ) AS pag(anoGasto, gasto),
```

```
(  
  SELECT dataAno, SUM(receita)  
  FROM data, negociacao  
 WHERE data.dataPK = negociacao.dataPK  
 GROUP BY dataAno  
 ) AS neg(anoGanho, ganho)  
WHERE anoGasto = anoGanho  
ORDER BY anoGasto
```

Exemplo de Consulta Complexa

pagamento

dataPK	funcPK	equipePK	cargoPK	salario	...
1	1	7	112	2.226,66	...
1	3	2	74	9.169,90	...
2	6	7	112	2.226,66	...
7	1	3	23	3.828,90	...

data

dataPK	...	dataAno
1	...	2016
2	...	2016
3	...	2017
7	...	2020

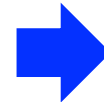
negociacao

dataPK	equipePK	clientePK	receita	...
1	7	2	5.245,00	...
3	3	3	5.431,23	...
7	1	3	5.789,00	...
1	3	4	9.323,00	...

```
SELECT anoGasto, gasto, ganho
FROM
( SELECT dataAno, SUM(salario)
  FROM data, pagamento
 WHERE data.dataPK = pagamento.dataPK
 GROUP BY dataAno
) AS pag(anoGasto, gasto,
(
  SELECT dataAno, SUM(receita)
  FROM data, negociacao
 WHERE data.dataPK = negociacao.dataPK
 GROUP BY dataAno
) AS neg(anoGanho, ganho)
WHERE anoGasto = anoGanho
ORDER BY anoGasto
```

Exemplo de Consulta Complexa

```
SELECT anoGasto, gasto, ganho
FROM
( SELECT dataAno, SUM(salario)
  FROM data, pagamento
 WHERE data.dataPK = pagamento.dataPK
 GROUP BY dataAno
) AS pag(anoGasto, gasto),
(
  SELECT dataAno, SUM(receita)
  FROM data, negociacao
 WHERE data.dataPK = negociacao.dataPK
 GROUP BY dataAno
) AS neg(anoGanho, ganho)
WHERE anoGasto = anoGanho
ORDER BY anoGasto
```



anoGasto	gasto	ganho
2016	13.623,22	14.568,00
2017	0	5.431,23
2020	3.828,90	5.789,00