OPERADORES SQL AVANÇADOS

Sérgio Mergen

Sumário

- Common Table Expressions (CTE)
- Funções de Janela
- OLAP (Online Analytical Processing)
  - Datawarehouse
  - Fatos e Dimensões
  - Operadores OLAP
  - OLAP na prática

Common Table Expression (CTE)

- Uma **CTE** é uma expressão de tabela temporária que pode ser definida dentro de uma consulta SELECT, INSERT, UPDATE ou DELETE, usando a cláusula WITH.

- Ela facilita a leitura, manutenção e reutilização de subconsultas complexas.

Common Table Expression (CTE)

- Template de uso de CTE

```
WITH nome_da_cte AS (
    -- subconsulta
    SELECT ...
)

SELECT ...
FROM nome_da_cte;
```

Common Table Expression (CTE)

- Ex. A consulta abaixo retorna projetos cujo custo seja maior do que a média de custo de todos os projetos
- A CTE media_custo é usada como se fosse uma tabela normal

```
WITH media_custo AS (
    SELECT AVG(custo) AS media
    FROM proj
)

SELECT p.*
FROM proj p
JOIN media_custo m ON p.custo > m.media;
```

Common Table Expression (CTE)

- O exemplo anterior poderia ser resolvido por meio de subconsultas de forma mais direta.
  - O CTE nesse caso só ajudou a deixar a lógica mais clara
- Visões podem ser usadas para a mesma finalidade
- Porém, elas são estruturas permanentes, que ocupam espaço e aumentam a complexidade do banco
- Caso a necessidade seja temporária, é melhor recorrer a CTEs

Common Table Expression (CTE)

- CTE podem ser usadas para evitar que uma subconsulta seja repetida
- Isso torna a consulta mais legível
- Além disso, dependendo do banco de dados e da complexidade de consulta, isso pode até mesmo resultar em maior eficiência na execução
  - Por evitar que o trecho seja executado mais do que uma vez

Common Table Expression (CTE)

- A consulta abaixo retorna projetos cuja custo total (soma do custo das atividades) seja superior a média de custos de todos os projetos

```
SELECT p.idProj, p.nome, SUM(a.custo) AS custo
FROM proj p JOIN ativ a ON a.idProj = p.idProj
GROUP BY p.idProj
HAVING SUM(a.custo) > (
    SELECT AVG(custo)
    FROM (
        SELECT SUM(custo) AS custo
        FROM ativ
        GROUP BY idProj
    ) sub
);
```

Common Table Expression (CTE)

- Perceba que a agregação é feita duas vezes

```
SELECT p.idProj, p.nome, SUM(a.custo) AS custo
FROM proj p JOIN ativ a ON a.idProj = p.idProj
GROUP BY p.idProj
HAVING SUM(a.custo) > (
    SELECT AVG(custo)
    FROM (
        SELECT SUM(custo) AS custo
        FROM ativ
        GROUP BY idProj
    ) sub
);
```

Common Table Expression (CTE)

- Com visões, o cálculo da agregação é especificado apenas uma vez
  - Porém, na execução, provavelmente a visão será desdobrada, levando a agregação a ser executada duas vezes

```
CREATE VIEW custo_por_projeto AS
    SELECT idProj, SUM(custo) AS custo_total
    FROM ativ
    GROUP BY idProj;

SELECT p.idProj, p.nome, c.custo_total
FROM proj p JOIN custo_por_projeto c ON p.idProj = c.idProj
WHERE c.custo_total > (
  SELECT AVG(custo_total) FROM custo_por_projeto
);
```

Common Table Expression (CTE)

- Com CTE, o cálculo da agregação é feito uma única vez
- A consulta fica mais legível e potencialmente mais eficiente

```
WITH custo_por_projeto AS (
  SELECT idProj, SUM(custo) AS custo_total
  FROM ativ
  GROUP BY idProj
)

SELECT p.idProj, p.nome, c.custo_total
FROM proj p JOIN custo_por_projeto c ON p.idProj = c.idProj
WHERE c.custo_total > (
  SELECT AVG(custo_total) FROM custo_por_projeto
);
```

Common Table Expression (CTE)

- CTEs também podem ser usadas de forma recursiva, permitindo que a CTE seja chamada dentro da própria CTE
  - Por meio da cláusula WITH RECURSIVE

- Isso é útil, por exemplo, em cenários que envolvam processamento hierárquico

Common Table Expression (CTE)

- Para exemplificar, suponha que se deseja obter todos os subordinados (diretos e indiretos) de um funcionário específico (raiz)
  - Para cada funcionário, deve-se retornar quem é seu chefe direto e qual a distância (nível de subordinação) até o raiz
- Vimos esse exemplo em outra aula
  - Na ocasião, foi usada uma solução baseada em procedures

Common Table Expression (CTE)

- Ex. Deseja-se obter os subordinados (diretos e indiretos) do funcionário 1

Tabela func

idFunc	idChefe
1	Null
2	1
3	1
4	2
5	2
6	4

Resposta

idFunc	idChefe	Nivel
2	1	0
3	1	0
4	2	1
5	2	1
6	4	2

Common Table Expression (CTE)

- Ex. Deseja-se obter os subordinados (diretos e indiretos) do funcionário 2

Tabela func

idFunc	idChefe
1	Null
2	1
3	1
4	2
5	2
6	4

Resposta

idFunc	idChefe	Nivel
4	2	0
5	2	0
6	4	1

Common Table Expression (CTE)

- Ex. Deseja-se obter os subordinados (diretos e indiretos) do funcionário 3

Tabela func

idFunc	idChefe
1	Null
2	1
3	1
4	2
5	2
6	4

Resposta

idFunc	idChefe	Nivel

Common Table Expression (CTE)

- O CTE recursivo abaixo mostra como obter os subordinados diretos e indiretos do idFunc = 1

```
WITH RECURSIVE hierarquia AS (
    -- Parte base: começa com o idFunc = 1
    SELECT idFunc, idChefe, 1 AS nivel
    FROM func
    WHERE  idFunc = 1

    UNION ALL

    -- Parte recursiva: pega quem tem chefe na hierarquia já descoberta
    SELECT f.idFunc, f.idChefe, h.nivel + 1
    FROM func f
    INNER JOIN hierarquia h ON f.idChefe = h.idFunc
)
SELECT * FROM hierarquia;
```

Sumário

- Common Table Expressions (CTE)
- Funções de Janela
- OLAP (Online Analytical Processing)
  - Datawarehouse
  - Fatos e Dimensões
  - Operadores OLAP
  - OLAP na prática

Funções de Janela

- As **funções de janela** (window functions) em SQL permitem executar cálculos sobre um conjunto de linhas relacionadas a uma linha atual — **sem agrupar** os dados (diferente de GROUP BY).

- Elas são úteis para análises mais avançadas, como rankings, totais acumulados e diferenças entre linhas.

Funções de Janela

- Template de uso

```
função() OVER (
    PARTITION BY ...  -- Divide os dados em grupos
    ORDER BY ...      -- Define a ordem das linhas dentro da janela
    ROWS ...          -- Define a moldura (range) da janela
)
```

Funções de Janela

Função	O que faz
ROW_NUMBER()	Numera as linhas de cada partição
RANK()	Dá um ranking, mas empata valores
DENSE_RANK()	Igual ao RANK(), mas sem “buracos”

Funções de Janela

- Listar os projetos por status, criando números sequenciais por ordem crescente de custo

Projeto			
idProj	**nome**	**custo**	**status**
1	Lucrei	12.000	aberto
2	Caos	50.000	fechado
3	ABC	20.000	aberto
4	ACME	12.000	aberto

```
SELECT  status, nome,  custo,
        row_number() over (partition by status order by custo ) AS ordem
```

status	nome	custo	ordem
aberto	Lucrei	12.000	1
aberto	ACME	12.000	2
aberto	ABC	20.000	3
fechado	Caos	50.000	1

O **row_number()** simplesmente gera um id sequencial

Funções de Janela

- Listar os projetos por status, criando números sequenciais por ordem crescente de custo

Projeto			
idProj	**nome**	**custo**	**status**
1	Lucrei	12.000	aberto
2	Caos	50.000	fechado
3	ABC	20.000	aberto
4	ACME	12.000	aberto

```
SELECT  status, nome,  custo,
                rank() over (partition by status order by custo ) AS ordem
```

status	nome	custo	ordem
aberto	Lucrei	12.000	1
aberto	ACME	12.000	1
aberto	ABC	20.000	3
fechado	Caos	50.000	1

O **rank()** cria um rank, mantendo o mesmo rank para valores empatados

Funções de Janela

- Listar os projetos por status, criando números sequenciais por ordem crescente de custo

Projeto			
idProj	**nome**	**custo**	**status**
1	Lucrei	12.000	aberto
2	Caos	50.000	fechado
3	ABC	20.000	aberto
4	ACME	12.000	aberto

```
SELECT  status, nome,  custo,
        dense_rank() over (partition by status order by custo ) AS ordem
```

status	nome	custo	ordem
aberto	Lucrei	12.000	1
aberto	ACME	12.000	1
aberto	ABC	20.000	2
fechado	Caos	50.000	1

O **dense_rank()** também mantem o mesmo rank para valores empatados, mas não deixa buracos na sequência gerada

Funções de Janela

Função	O que faz
SUM()	Soma acumulada ou total por partição
AVG(), MAX(), MIN()	Médias e extremos dentro da janela
LAG() / LEAD()	Pega o valor da linha anterior/próxima
FIRST_VALUE() / LAST_VALUE()	Retorna o primeiro ou último valor da janela

Funções de Janela

- Listar os projetos por status, exibindo o custo individual e o custo médio por status

Projeto			
idProj	**nome**	**custo**	**status**
1	Lucrei	12.000	aberto
2	Caos	50.000	fechado
3	ABC	18.000	aberto
4	ACME	12.000	aberto

```
SELECT  status, nome,  custo,
                AVG(custo) over (partition by status) AS media
```

status	nome	custo	media
aberto	Lucrei	12.000	14.000
aberto	ABC	18.000	14.000
aberto	ACME	12.000	14.000
fechado	Caos	50.000	50.000

O **avg()** calcula a média por valor de partição, e repete esse valor para todos os registros daquela partição

Funções de Janela

- Listar os projetos por status, exibindo o custo individual e o custo médio por status

Projeto			
idProj	**nome**	**custo**	**status**
1	Lucrei	12.000	aberto
2	Caos	50.000	fechado
3	ABC	18.000	aberto
4	ACME	12.000	aberto

```
SELECT  status, nome,  custo,
                SUM(custo) over (partition by status) AS total
```

status	nome	custo	total
aberto	Lucrei	12.000	42.000
aberto	ABC	18.000	42.000
aberto	ACME	12.000	42.000
fechado	Caos	50.000	50.000

Trocando o **avg()** por **sum()**, o resultado é a soma de todos os valores de cada partição

Funções de Janela

- Listar os projetos por status, exibindo o custo individual e o total acumulado por status

Projeto			
idProj	**nome**	**custo**	**status**
1	Lucrei	12.000	aberto
2	Caos	50.000	fechado
3	ABC	18.000	aberto
4	ACME	12.000	aberto

```
SELECT  status, nome,  custo,
        sum(custo) over (partition by status order by custo) AS total
```

status	nome	custo	total
aberto	Lucrei	12.000	24.000
aberto	ACME	12.000	24.000
aberto	ABC	18.000	42.000
fechado	Caos	50.000	50.000

Se quisermos que a soma seja acumulada, é necessário incluir a cláusula **order by**

Perceba que para valores iguais de custo, o acúmulo é feito uma única vez

Funções de Janela

- Listar os projetos por status, exibindo o custo individual e o total acumulado por status

Projeto			
idProj	**nome**	**custo**	**status**
1	Lucrei	12.000	aberto
2	Caos	50.000	fechado
3	ABC	18.000	aberto
4	ACME	12.000	aberto

```
SELECT  status, nome,  custo,
        sum(custo) over (partition by status order by custo, idProj) AS total
```

status	nome	custo	total
aberto	Lucrei	12.000	12.000
aberto	ACME	12.000	24.000
aberto	ABC	18.000	42.000
fechado	Caos	50.000	50.000

Se quisermos que o acúmulo seja feito para cada valor, mesmo que igual, deve-se adicionar mais alguma coluna à cláusula de ordenação

Sumário

- Common Table Expressions (CTE)
- Funções de Janela
- OLAP (Online Analytical Processing)
  - Datawarehouse
  - Fatos e Dimensões
  - Operadores OLAP
  - OLAP na prática

Datawarehouse

- Um **data warehouse** (ou armazém de dados) é um sistema especializado para armazenar, organizar e analisar grandes volumes de dados históricos, normalmente oriundos de diversas fontes da empresa.

- Ele é projetado para **suporte à decisão**, diferente de bancos de dados operacionais, que são otimizados para o dia a dia das operações (como registrar vendas ou atualizar cadastros).

Datawarehouse

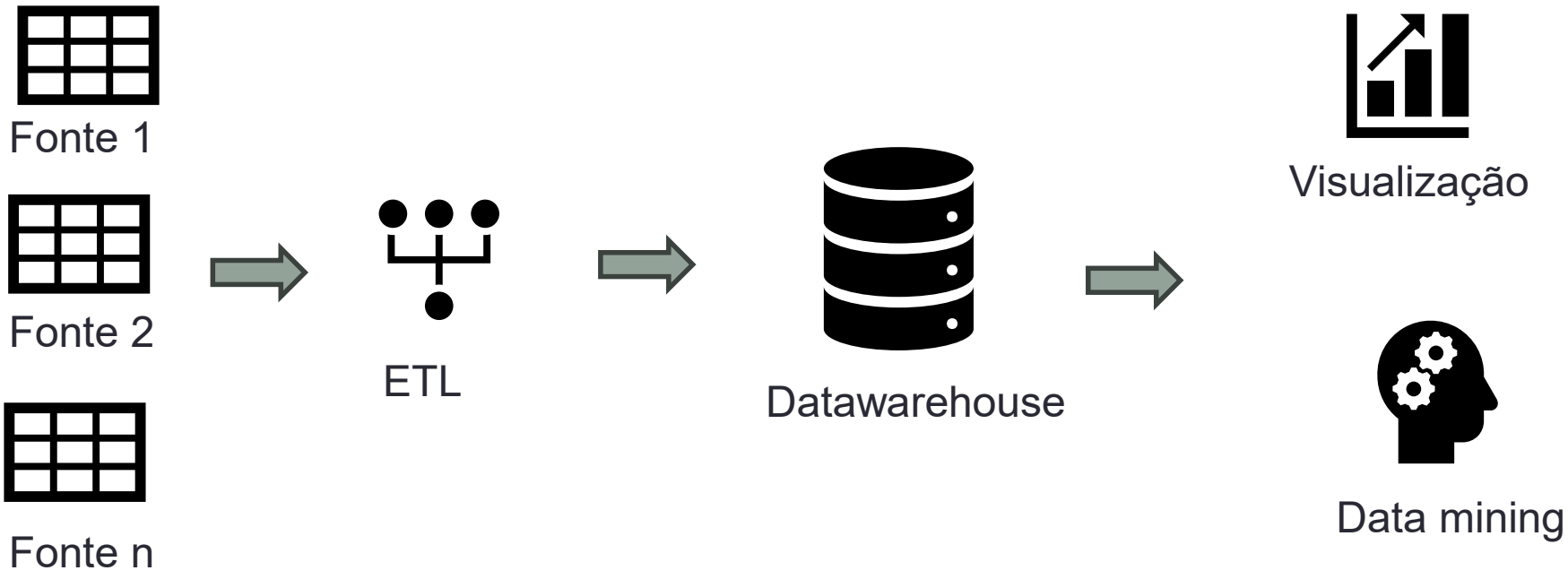
- Exemplos de perguntas que um datawarehouse tem mais condições de resolver
  - "Quais produtos mais cresceram nas vendas nos últimos 5 anos?"
  - "Qual o perfil de clientes que mais compram em dezembro?"
  - "Quais filiais têm maior margem de lucro histórica?"

Datawarehouse

- **Orientado a assuntos (subject-oriented):**
  - Os dados são organizados por temas relevantes para o negócio, como *vendas*, *estoque*, *clientes*, e não por processos do dia a dia.
- **Integrado (integrated):**
  - Os dados vêm de diversas fontes (planilhas, bancos de dados, ERPs), mas são unificados com consistência (ex: padronização de nomes, datas, moedas).
- **Variável no tempo (time-variant):**
  - Guarda **histórico dos dados**, permitindo análises comparativas (ex: evolução de vendas ao longo dos anos).
- **Não-volátil (non-volatile):**
  - Os dados, uma vez inseridos, **não são alterados**. Isso garante rastreabilidade e consistência para análises históricas.

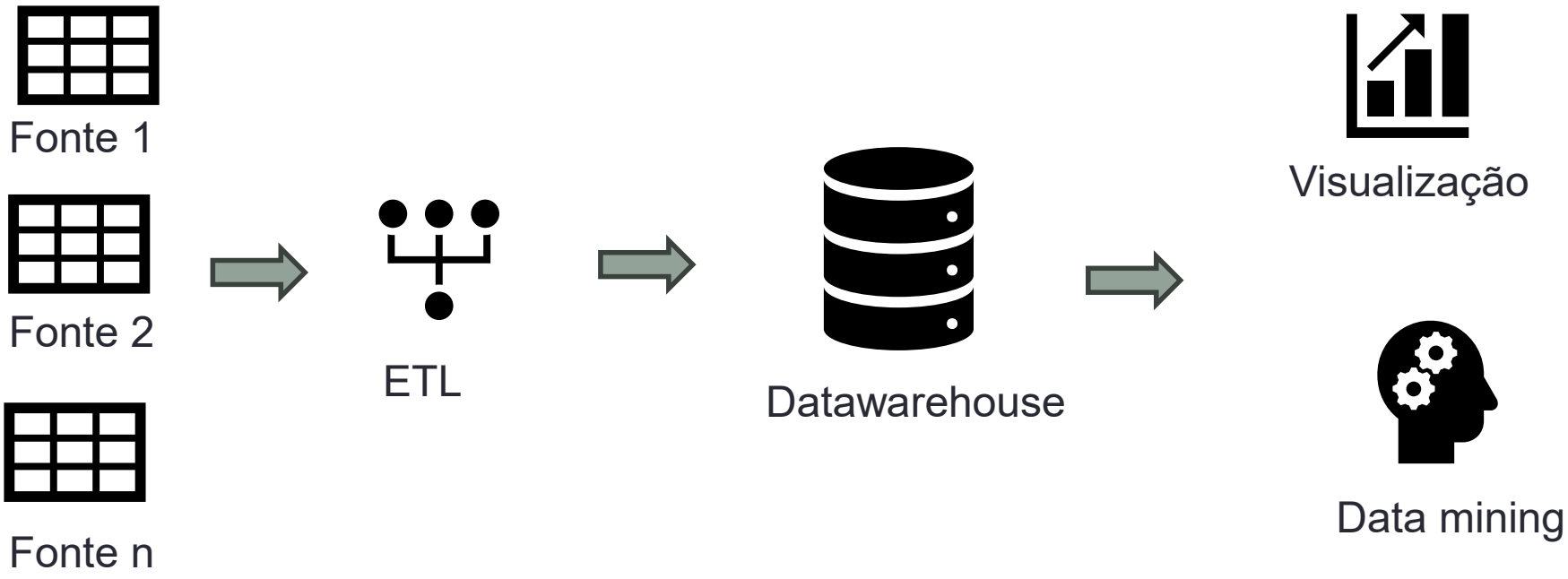
Datawarehouse

- Abaixo a arquitetura típica de datawarehouses



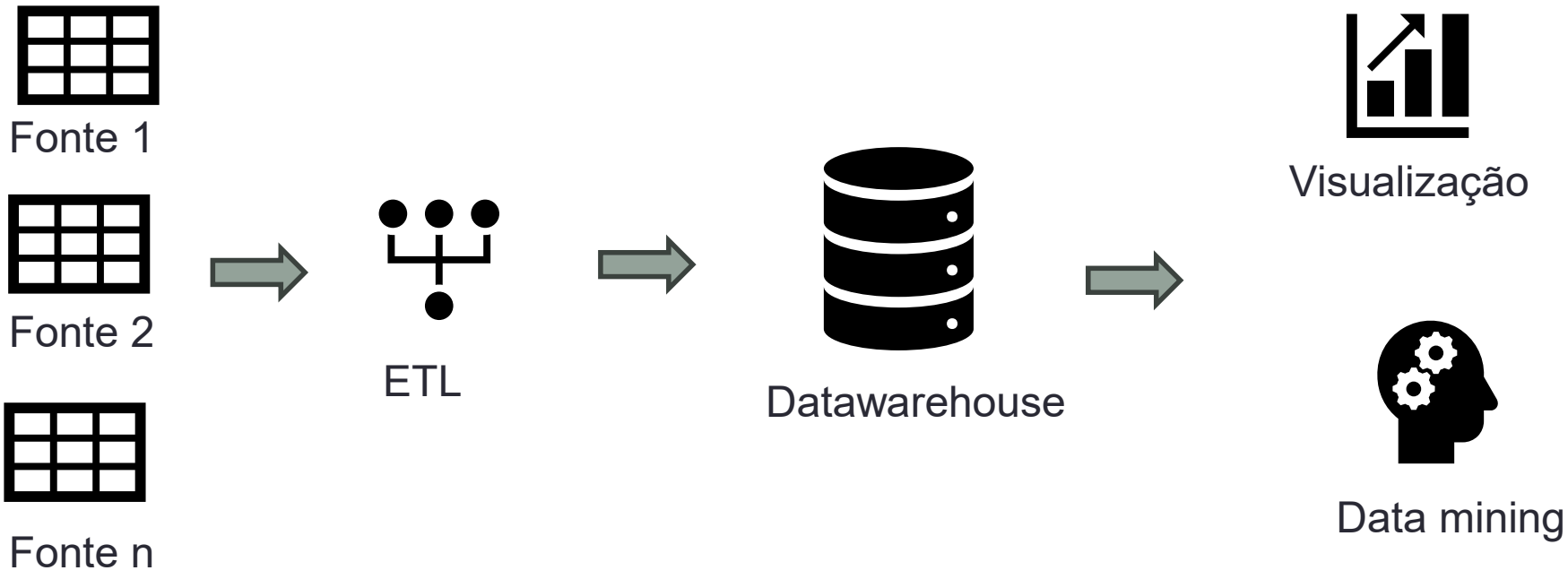
Datawarehouse

- As fontes de dados são os repositórios originais dos dados crus
  - Ex. Bancos de dados, arquivos CSV, planilhas eletrônicas, ...



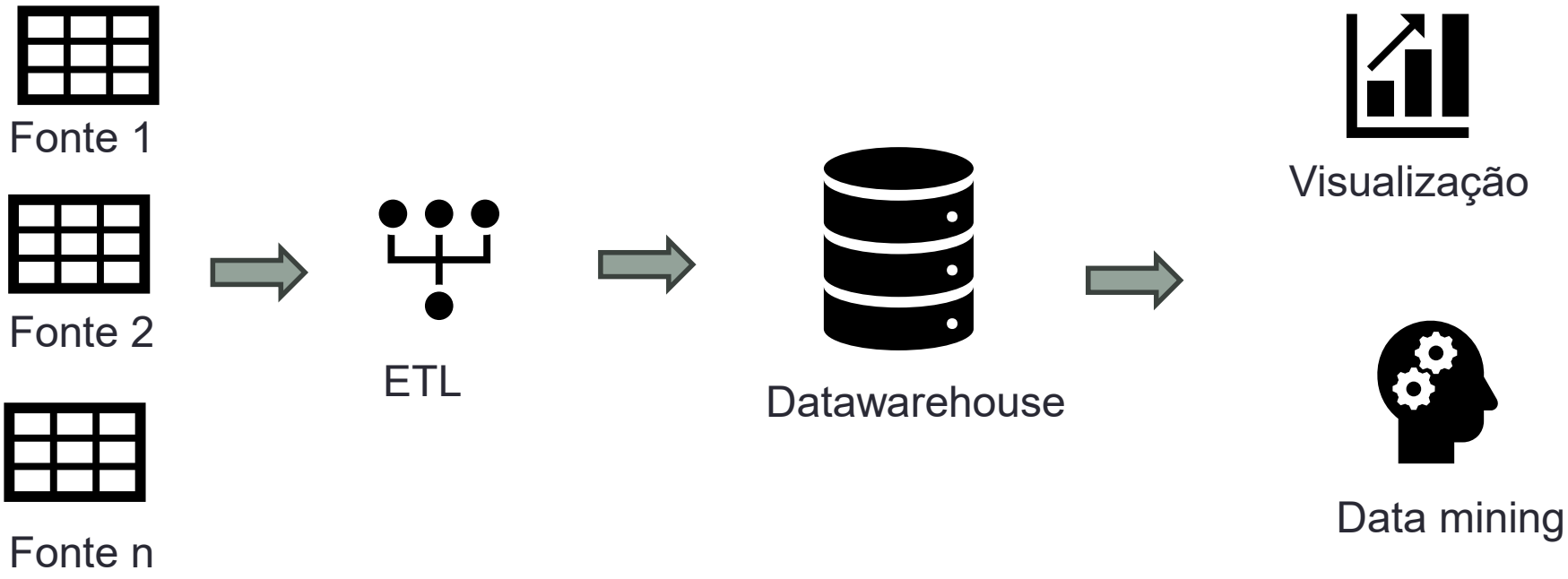
Datawarehouse

- A etapa de ETL prepara os dados para a análise
  - **E**xtract: extrai os dados das fontes originais.
  - **T**ransform: limpa, converte e padroniza os dados.
  - **L**oad: carrega os dados transformados no data warehouse.



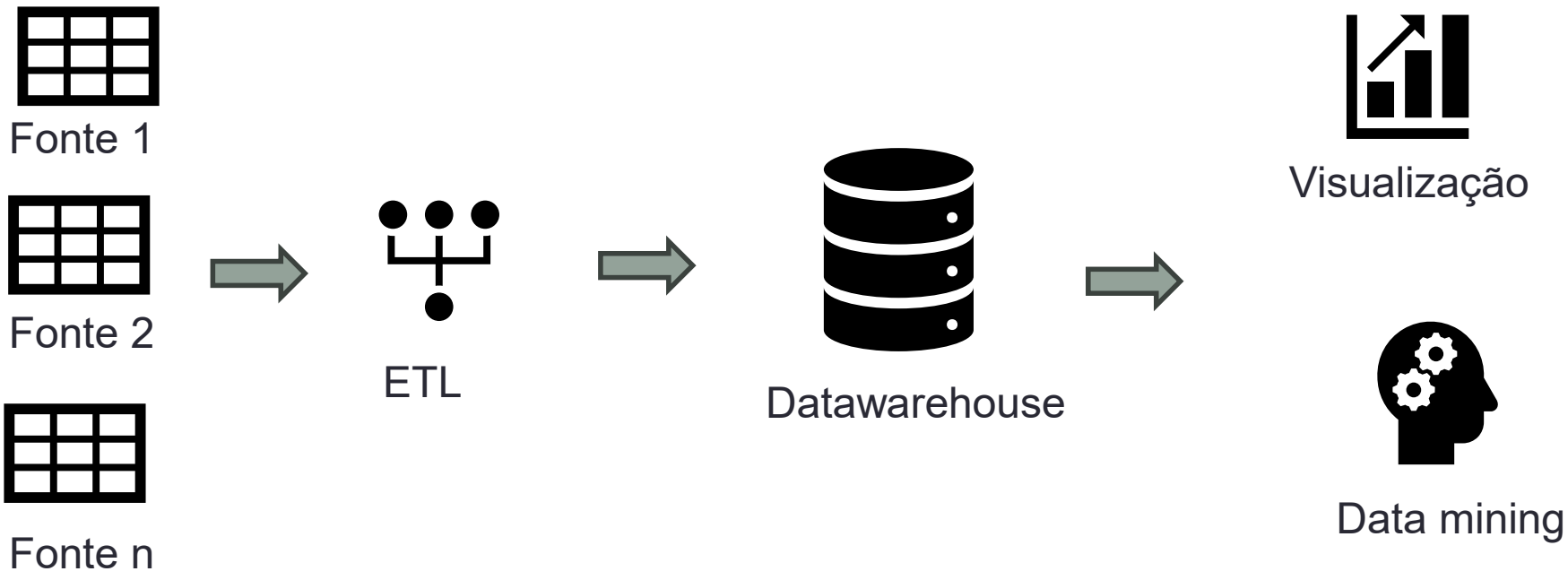
Datawarehouse

- Exemplos de transformações feitas via ETL
  - Limpeza de dados, padronização, conversão, complementação, ...



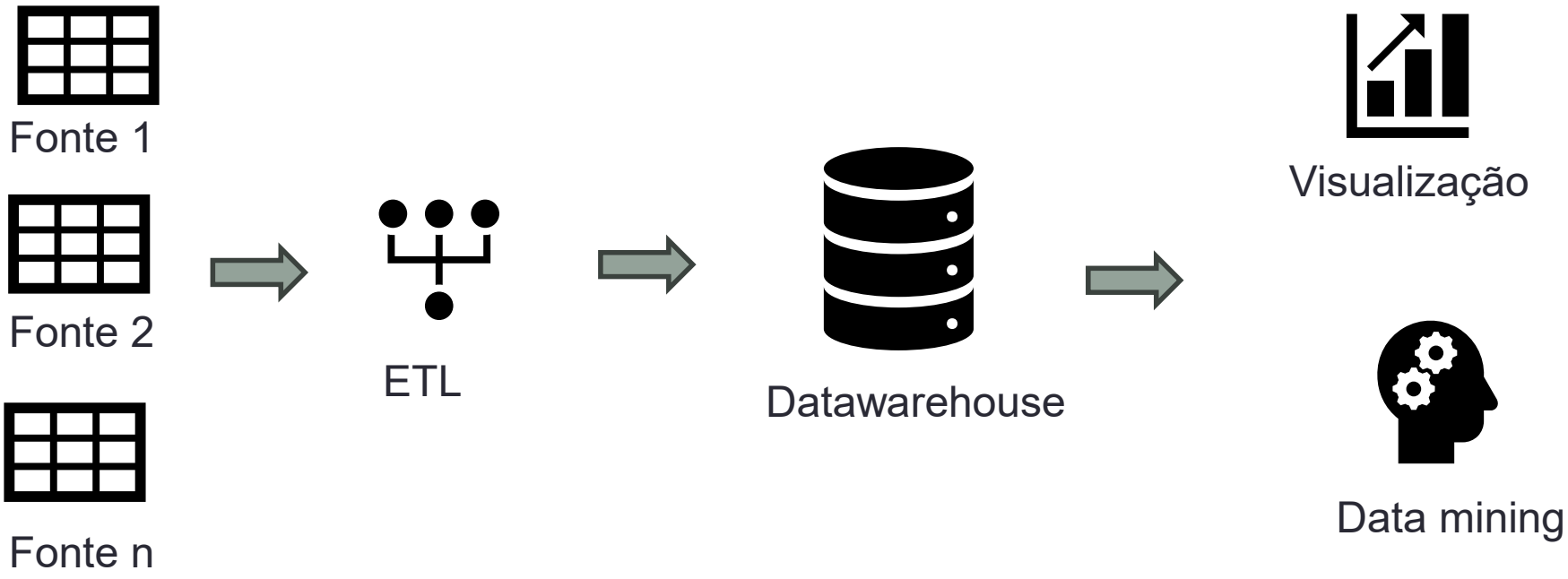
Datawarehouse

- O datawarehouse é um banco especialmente projetado para guardar os dados que serão analisados
  - Dados são tratados como fatos e dimensões



Datawarehouse

- Várias formas de análise são possíveis
  - Ex. visualização exploratória, data mining, ...



Sumário

- Common Table Expressions (CTE)
- Funções de Janela
- OLAP (Online Analytical Processing)
  - Datawarehouse
  - Fatos e Dimensões
  - Operadores OLAP
  - OLAP na prática

Fatos e Dimensões

- Os datawarehouse guardam apenas as informações que possuem valor do ponto de vista de tomada de decisão
- Essas informações são sumarizações das informações obtidas das fontes de dados originais
- Ex. nome dos funcionários que atuaram em cada projeto é irrelevante
  - No entanto, a quantidade de funcionários alocados é uma informação importante

Fatos e Dimensões

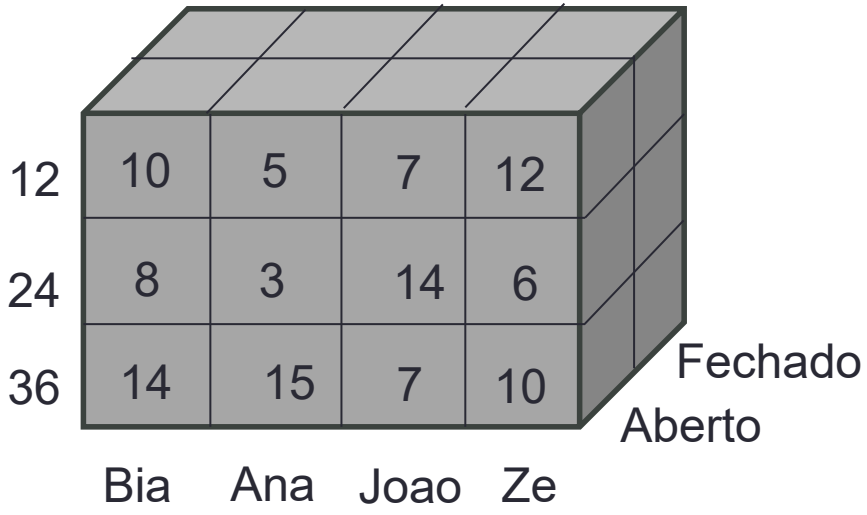
- Em datawarehouses, os dados são vistos como fatos e dimensões

- As combinações entre as dimensões levam a um fato, que é uma métrica numérica associada às dimensões combinadas (ex. soma e contagem)

- Existem sugestões para a modelagem lógica dos fatos e dimensões que simplificam o acesso aos dados
  - Ex. snowflake, star

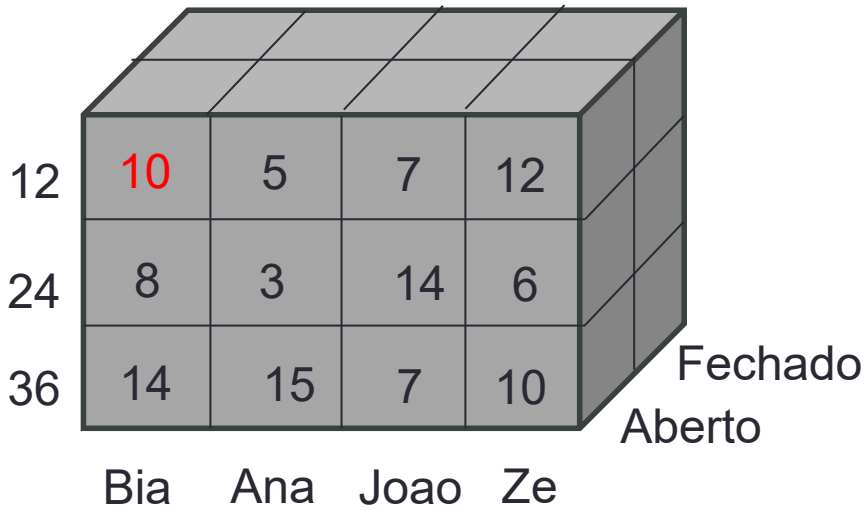
Fatos e Dimensões

- No exemplo abaixo, há três dimensões:
  - Lider de projeto (Bia, Ana, Joao e Zé)
  - Duração de projeto (12, 24 e 36 meses)
  - Status do projeto (aberto e fechado)
- O fato é o custo do projeto (em milhões)



Fatos e Dimensões

- Independente da modelagem lógica, uma combinação entre dimensões pode ser conceitualmente vista como um cubo
  - Onde os valores das cédulas são os dados agregados
- Por exemplo
  - a Bia lidera projetos abertos com duração de 12 meses cujo custo somado chega a 10 milhões



Sumário

- Common Table Expressions (CTE)
- Funções de Janela
- OLAP (Online Analytical Processing)
  - Datawarehouse
  - Fatos e Dimensões
  - Operadores OLAP
  - OLAP na prática

Operadores OLAP

- As análises de dados históricos e resumidos são comumente chamados de OLAP (Online Analytical Processing)

- OLAP (Online Analitical Processing) se preocupa mais com a eficiência na análise dos dados do que com a atualização dos dados

- Já OLTP (On Line Transaction Processing) se preocupa mais com o uso corriqueiro de um sistema de gestão para inclusão e acesso a dados
  - São os bancos que costumamos usar no dia a dia

Operadores OLAP

- Em ambientes OLAP, onde os dados estão dispostos na forma de fatos e dimensões, trabalha-se com **consultas multidimensionais** para explorar dados sob diferentes perspectivas
  - agregando informações em múltiplos níveis de granularidade

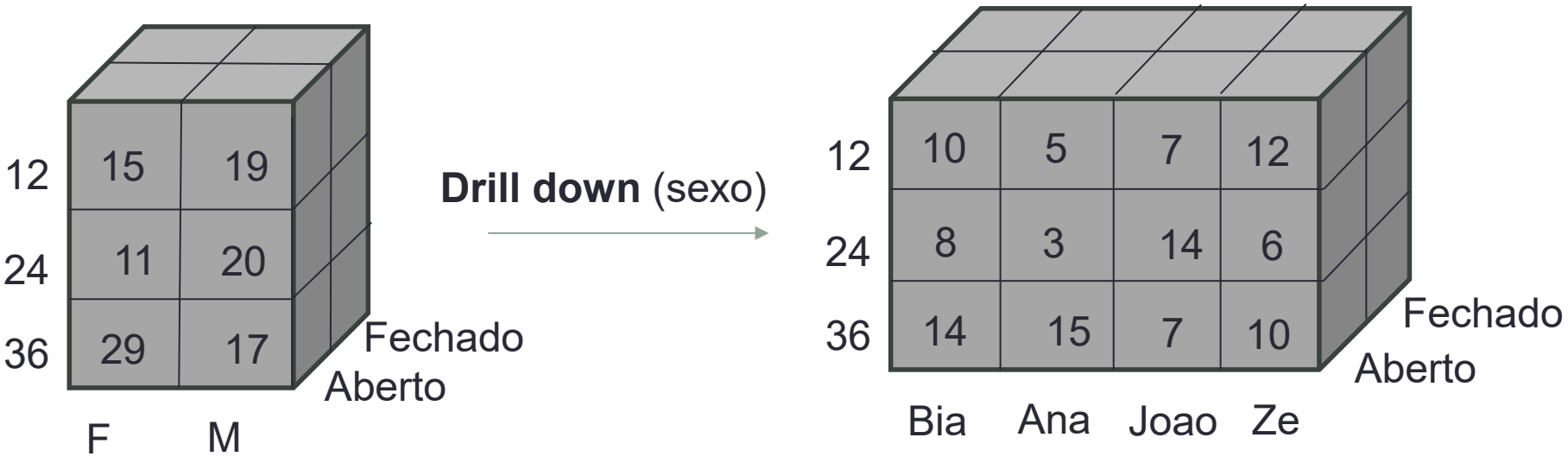
- Principais operadores OLAP:
  - Roll up
  - Drill down
  - Slice
  - Dice
  - Pivot

Operadores OLAP

- **Roll Up:** Agrega informações detalhadas em uma informação mais resumida
- No exemplo abaixo, a agregação reduz a granularidade de uma das dimensões (líder de projeto)

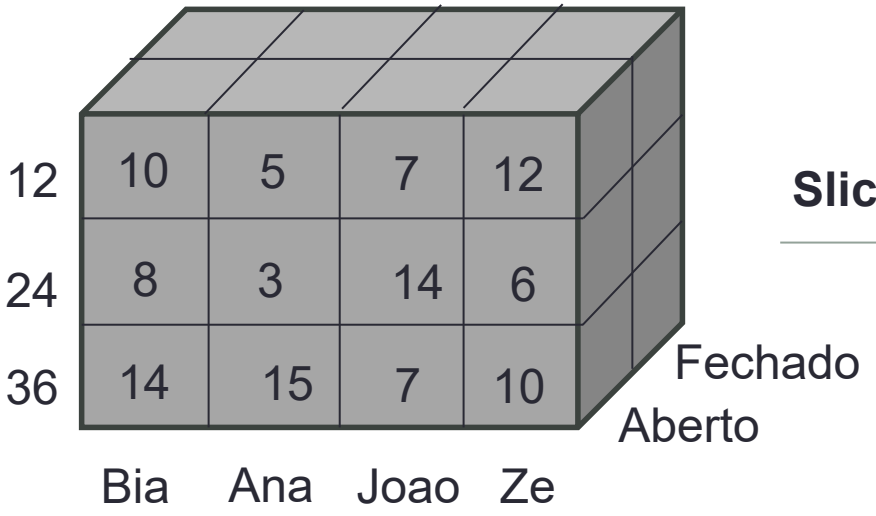
Operadores OLAP

- **Drill Down:** adiciona detalhes a uma informações que é agregada
- No exemplo abaixo, o drill down detalha os líderes escondidas na informações referente ao sexo



Operadores OLAP

- **Slice:** escolhe um valor de uma das dimensões do cubo
- No exemplo abaixo, o slice escolheu apenas o valor ‘aberto’ da dimensão ‘status’

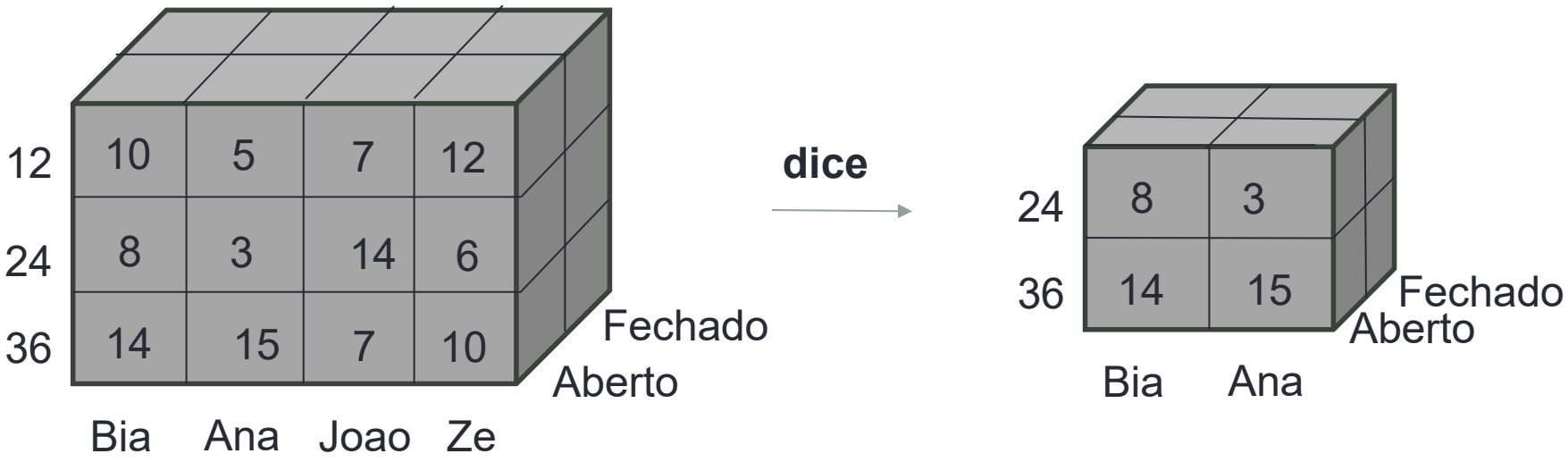


Slice (aberto)

	Bia	Ana	Joao	Ze
12	10	5	7	12
24	8	3	14	6
36	14	15	7	10

Operadores OLAP

- **Dice:** forma um subcubo, definindo os valores de cada dimensão que interessam
- No exemplo abaixo, o dice escolheu apenas os valores ’24’ e ’36’, da dimensão ‘duração’ e ‘Bia’ e ‘Ana’, da dimensão ‘líder’



Operadores OLAP

- **Pivot:** aplica uma rotação nas dimensões do cubo
- No exemplo abaixo, a operação pivot fez uma transposição de um cubo de duas dimensões

	Bia	Ana	Joao	Ze
12	10	5	7	12
24	8	3	14	6
36	14	15	7	10

pivot

	12	24	36
Bia	10	8	14
Ana	5	3	15
Joao	7	14	7
Ze	12	6	10

Sumário

- Common Table Expressions (CTE)
- Funções de Janela
- OLAP (Online Analytical Processing)
  - Datawarehouse
  - Fatos e Dimensões
  - Operadores OLAP
  - OLAP na prática

OLAP na prática

- Existem bancos de dados que são especialmente projetados para datawarehouses
  - Amazon Redshift, google BigQuery, Apache Hive

- Esses bancos utilizam mecanismos diferentes para armazenamento dos registros e realização de junções entre tabelas
  - Uma das principais diferenças é que os dados são armazenados orientados a colunas e não a registros

OLAP na prática

- Para bancos de dados pouco volumosos, é possível usar bancos de dados convencionais, que possuem suporte a alguns operadores OLAP
  - Ex. MySQL, PostgreSQL

- Outra opção é usar ferramentas de análise de dados
  - Ex. Tableau, Power BI, Apache Superset, Metabase
  - Essas ferramentas permitem realizar operações OLAP visualmente
  - Por baixo dos dados, são usados os recursos OLAP dos bancos de dados
    - Ou são gerados cubos de dados em memória, onde as operações são executadas

OLAP na prática

- Para bancos de dados pouco volumosos, é possível usar bancos de dados convencionais, que possuem suporte a alguns operadores OLAP
  - Ex.
    - MySQL (suporte limitado)
    - PostgreSQL (suporte mais amplo)

- Os próximos slides mostram como o MySQL pode ser usado para realizar operações OLAP

OLAP na prática

- Pode-se aumentar ou reduzir o nível de detalhamento acrescentando ou reduzindo as colunas do agrupamento

Projeto			
idProj	**duracao**	**custo**	**status**
1	12	12.000	aberto
2	24	50.000	fechado
3	24	18.000	aberto
4	12	12.000	aberto

```
SELECT  status, duracao, SUM(custo)
FROM projeto
GROUP BY status, duracao
```

drill down

status	**duracao**	**custo**
aberto	12	24.000
aberto	24	18.000
fechado	24	50.000

```
SELECT  status, SUM(custo)
FROM projeto
GROUP BY status
```

roll up

status	**custo**
aberto	42.000
fechado	50.000

OLAP na prática

- O MySQL possui a cláusula WITH ROLLUP, que produz agregações para cada um dos níveis das colunas de agrupamento

Projeto			
idProj	**duracao**	**custo**	**status**
1	12	12.000	aberto
2	24	50.000	fechado
3	24	18.000	aberto
4	12	12.000	aberto

SELECT **status**, **duracao**, SUM(**custo**)
GROUP BY **status**, **duracao** WITH ROLLUP

status	**duracao**	**custo**
aberto	12	24.000
aberto	24	18.000
aberto	null	42.000
fechado	24	50.000
fechado	null	50.000
null	null	92.000

OLAP na prática

- A cláusula GROUPING complementa o ROLLUP, permitindo identificar quais registros são frutos de uma agregação

Projeto			
idProj	**duracao**	**custo**	**status**
1	12	12.000	aberto
2	24	50.000	fechado
3	24	18.000	aberto
4	12	12.000	aberto

```
SELECT
   CASE
   WHEN GROUPING(status) THEN 'total'
   WHEN GROUPING(duracao) THEN
         CONCAT ('subtotal de ', status)
   ELSE status
   END as status_info
duracao, SUM(custo)
GROUP BY status, duracao WITH ROLLUP
```

Status_info	duracao	custo
aberto	12	24.000
aberto	24	18.000
Subtotal de aberto	null	42.000
fechado	24	50.000
Subtotal de fechado	null	50.000
total	null	92.000

OLAP na prática

- O slice é obtido por meio de um filtro que seleciona um único valor de uma ou mais dimensões

Projeto			
idProj	**duracao**	**custo**	**status**
1	12	12.000	aberto
2	24	50.000	fechado
3	24	18.000	aberto
4	36	40.000	fechado
5	36	50.000	fechado
6	12	12.000	aberto

```
SELECT  status, duracao, SUM(custo)
FROM projeto
WHERE duracao = 24
GROUP BY status, duracao
```

slice

status	**duracao**	**custo**
aberto	24	18.000
fechado	24	50.000

OLAP na prática

- Já o dice é obtido por meio de um filtro que seleciona mais de um valor de uma ou mais dimensões

Projeto			
idProj	**duracao**	**custo**	**status**
1	12	12.000	aberto
2	24	50.000	fechado
3	24	18.000	aberto
4	36	40.000	fechado
5	36	50.000	fechado
6	12	12.000	aberto

```
SELECT  status, duracao, SUM(custo)
FROM projeto
WHERE duracao IN (24, 36)
GROUP BY status, duracao
```

dice

status	**duracao**	**custo**
aberto	24	18.000
fechado	24	50.000
fechado	36	90.000

OLAP na prática

- Exemplos de diferenças entre slice e dice

Situação	Tipo
ano = 2023	Slice
ano = 2023 AND região = 'Sul'	Slice
ano IN (2022, 2023)	Dice
região IN ('Sul', 'Sudeste')	Dice
ano = 2023 AND região IN ('Sul', 'Sudeste')	Dice
ano IN (2022, 2023) AND região = 'Sul'	Dice

OLAP na prática

- Vimos que um cubo é um conceito que se refere ao cruzamento de várias dimensões
- Cubos de três dimensões podem ser visualizados em ferramentas específicas, que permite realizar operações sobre o cubo de forma gráfica
  - Ex. Tableau, Power BI

- Já cubos de duas dimensões (também chamados de tabelas bidimensionais), podem ser visualizados na forma de tabelas
  - Uma dimensão para as linhas e outra para as colunas
  - Essa forma de representação é chamada de **pivot**

OLAP na prática

- Em MySQL, pode-se ‘pivotar’ os dados usando a cláusula CASE

Projeto			
idProj	**duracao**	**custo**	**status**
1	12	12.000	aberto
2	24	50.000	fechado
3	24	18.000	aberto
4	36	40.000	fechado
5	36	50.000	fechado
6	12	12.000	aberto

```
SELECT  duracao,
   SUM(CASE WHEN status = ‘A’ THEN custo ELSE 0 END) as A,
   SUM(CASE WHEN status = ‘F’ THEN custo ELSE 0 END) as F
FROM projeto
GROUP BY status, duracao
```

duracao	A	F
12	24.000	0
24	18.000	50.000
36	0	90.000

Atividade Individual

- O objetivo desta atividade é resolver as questões do slide seguinte em SQL usando os recursos apresentados na aula de hoje

- Use o banco de dados atualizado de movie, publicado no moodle no tópico da aula de hoje

Atividade Individual

1. Para cada ano, exibir os filmes ordenados por popularidade, acrescidos de uma coluna indicando a ordem de cada filme. Filmes com a mesma popularidade devem aparecer com a ordem empatada

2. Exibir os anos em que a média de orçamento dos filmes lançados foi superior à média das médias anuais de orçamento. Ignorar os filmes com orçamento igual a zero.

3. Para cada ano, exibir quantos filmes estão com status igual a Released, Rumored e Post Production. Cada valor de status deve ser exibido como uma coluna

Exercício para praticar

Como você resolveria a questão abaixo de forma eficiente?

> Exibir o grau de separação de Jack Nickolson (ou outro ator qualquer) com atores, considerando que conhecer alguém significa ter atuado junto no mesmo filme.

CTE recursivo seria a melhor alternativa?