TÉCNICO LISBOA

Bases de Dados

T24 - OLAP Parte II

Prof. Daniel Faria

Sumário

- Recapitulação
- OLAP
- OLAP em SQL

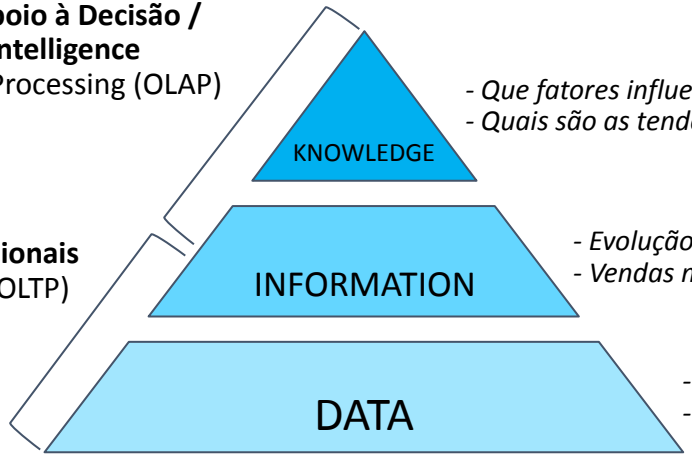
TÉCNICO LISBOA

Recapitulação

Sistemas de Informação

Sistemas de Apoio à Decisão / Business Intelligence
Online Analytic Processing (OLAP)

Sistemas de Informação Operacionais
Online Transaction Processing (OLTP)



- Que fatores influenciam as vendas?
- Quais são as tendências das vendas?

- Evolução das vendas por cidade e mês
- Vendas mensais por produto em Lisboa

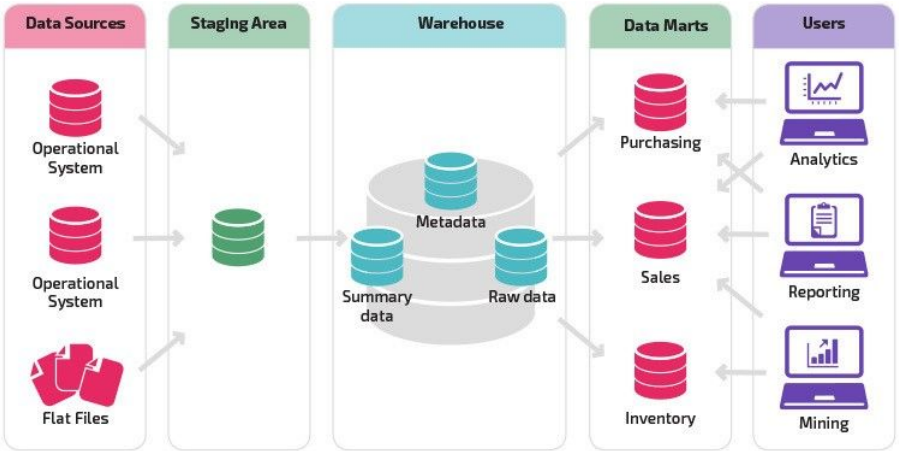
- Listar o preço de cada produto
- Registar vendas

OLTP vs. OLAP

- **Online Transaction Processing (OLTP)**
  - Dados dinâmicos
    - Operações de escrita frequentes
  - Transações de escrita ou leitura rápidas e simples
- **Online Analytic Processing (OLAP)**
  - Dados quase estáticos
    - Atualizações periódicas (e.g. mensais, anuais) em bulk
  - Transações complexas mas só de leitura

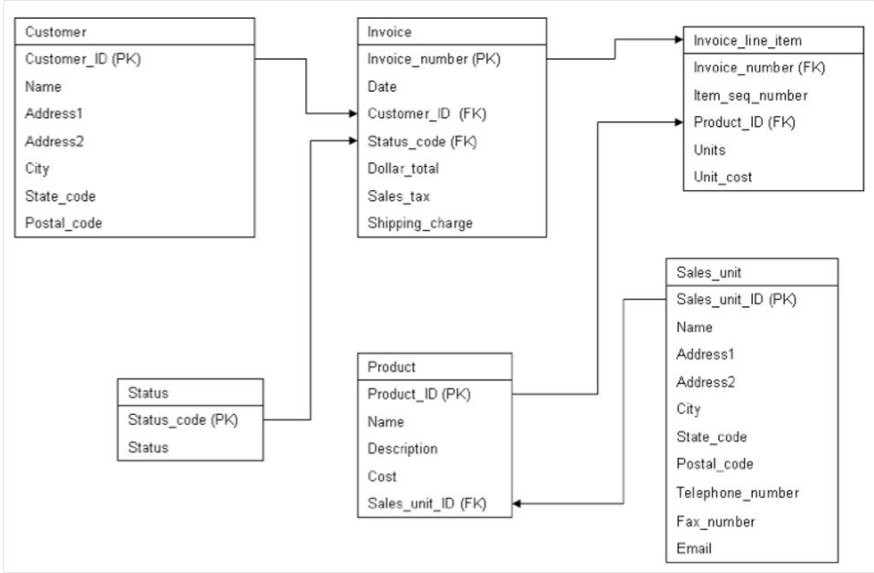
Data Warehouse

- Repositório central de grande volume de dados consolidados, históricos e agregados, complementados com sumários
- Permite simplificar queries complexas para análise de dados
- Base para reporte e análise de dados e componente chave de business intelligence



Modelo Relacional Normalizado

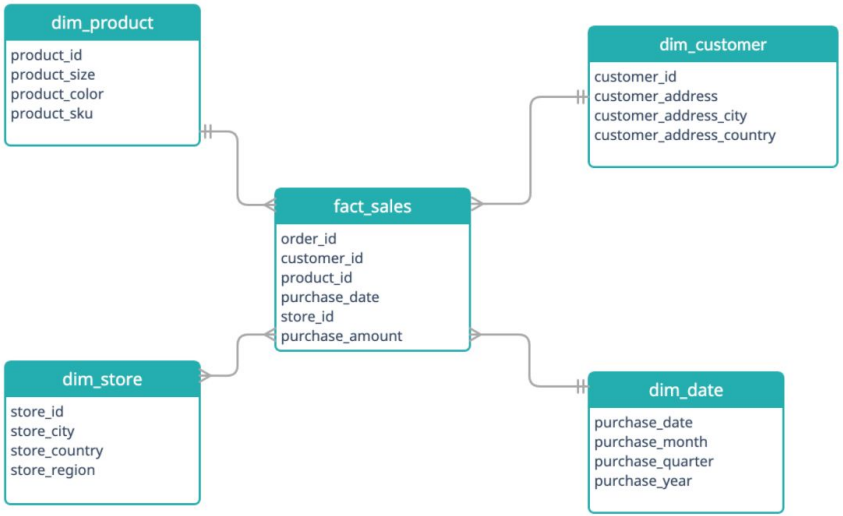
- Ideal para bases de dados operacionais (OLTP) em que há escrita frequente
  - Atomicidade dos dados minimiza custo de operações de escrita e evita inconsistências
- Pouco eficiente para processos analíticos devido à necessidade de atravessar várias tabelas (joins múltiplos)
  - Particularmente para agregações globais sobre os joins



Esquema em Estrela

Dois tipos de tabelas:

- **Tabela(s) de factos**
  - Grande dimensão
  - Frequentemente normalizada(s)
  - Objeto primário de análise de dados
- **Tabelas de dimensões**
  - Relativamente pequenas
  - Geralmente não normalizadas
  - Contém informação adicional sobre os elementos (ou dimensões) da tabela de factos



TÉCNICO LISBOA

Esquema em Estrela

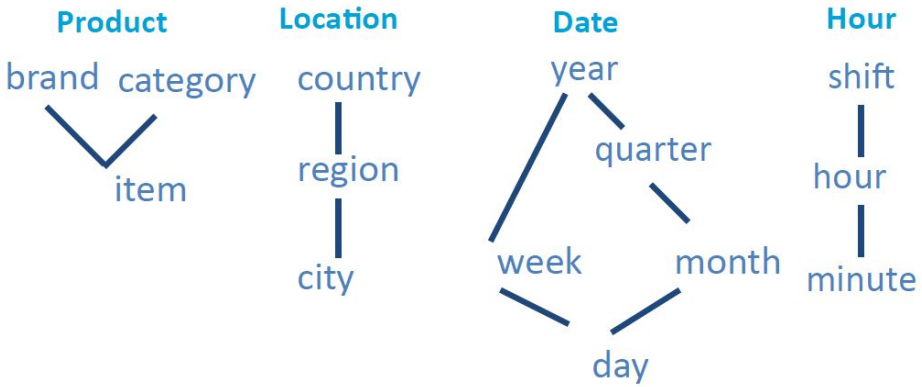
Tabela(s) de factos:

- **Atributos-medida**: quantificam os factos e (geralmente) podem ser agregados
  - E.g. purchase_amount
- **Atributos-dimensão**: correspondem a dimensões sobre as quais os atributos-medida podem ser analisados
  - Geralmente índices numéricos que são chaves estrangeiras para as tabelas de dimensões
- A **chave** da tabela de factos é a combinação de chaves estrangeiras das várias tabelas de dimensões

Esquema em Estrela

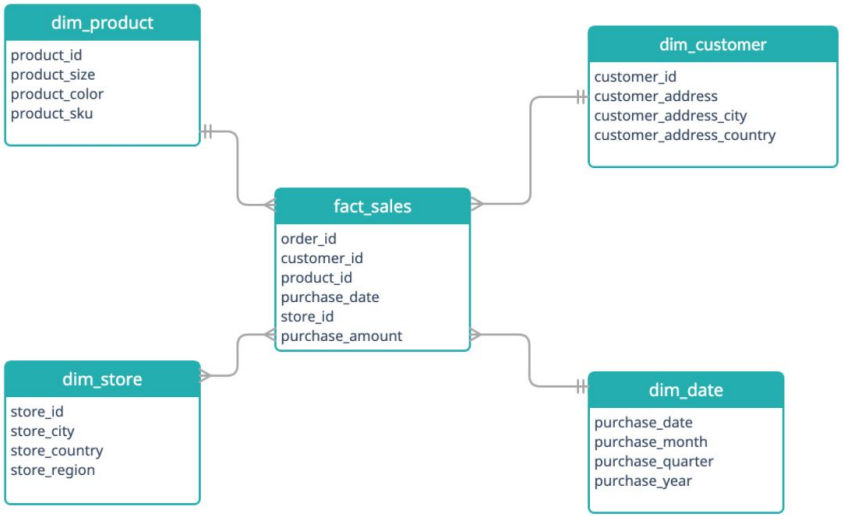
Tabelas de dimensões:

- Explicações dos factos: quem, onde, quando, o quê, …
- Contêm informação frequentemente redundante e hierárquica
  - Redundância é menos importante do que eficiência de acesso
  - Operações de escrita são raras



Esquema em Estrela

- **Query típica:**
  - Join entre a tabela de factos com uma ou mais tabelas de dimensões
  - Agrupamento em um ou mais atributos das tabelas de dimensões
  - Agregação sobre um ou mais dos atributos-medida da tabela de factos



Data Warehouses e SGBD

MOLAP

- Armazenamento baseado em colunas: arrays persistentes em disco
  - Arrays podem ser comprimidos, reduzindo custos de armazenamento, I/O e memória substancialmente
  - Queries apenas precisam de localizar os atributos relevantes, reduzindo custos de I/O e memória

ROLAP

- Implementação de esquema em estrela (ou snowflake) em SGBD relacionais
  - Menos eficiente (OK para data warehouses pequenos)

TÉCNICO LISBOA

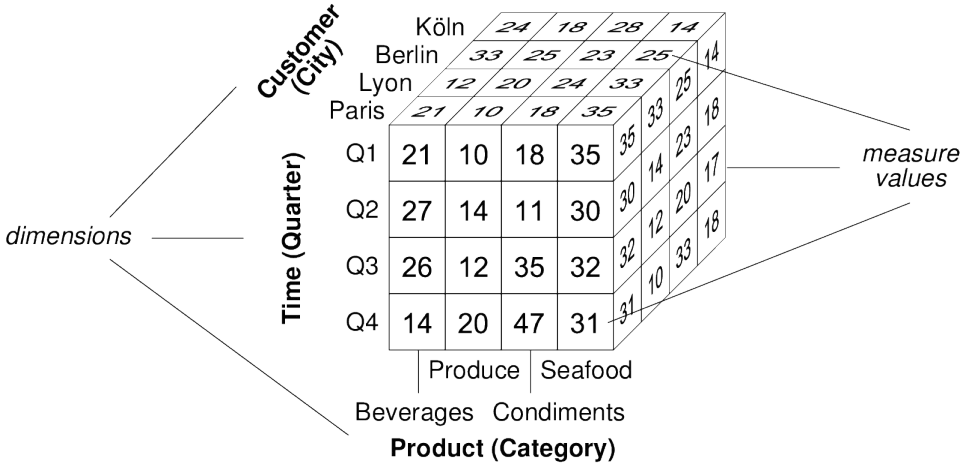
OLAP

OLAP

- **Processo interactivo** de criar, gerir, analisar, e reportar sobre dados
- Análise de grandes quantidades de dados em **tempo real** (i.e., com latência negligível)
- Os dados são compreendidos e manipulados como se estivessem guardados num **array multi-dimensional** (um hipercubo)
    - Mas podemos implementar modelo em estrela em SQL, e existe suporte para várias operações OLAP

OLAP

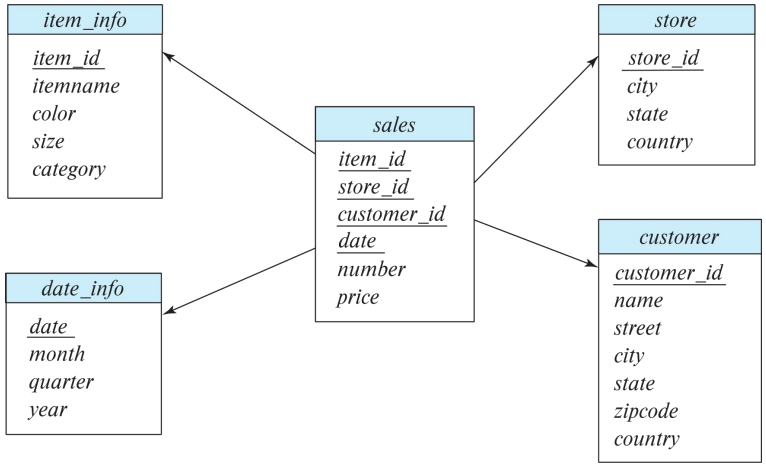
- Modelo de dados: hipercubo (pode ser implementado com esquema em estrela)



A. Vaisman, E. Zimányi, *"Data Warehouse Systems: Design and Implementation"*, Springer, 2014

OLAP

- Base de dados exemplo



Versão simplificada de *sales* com join a *item_info*

item_name	color	clothes_size	quantity
dress	dark	small	2
dress	dark	medium	6
dress	dark	large	12
dress	pastel	small	4
dress	pastel	medium	3
dress	pastel	large	3
dress	white	small	2
dress	white	medium	3
dress	white	large	0
pants	dark	small	14
pants	dark	medium	6
pants	dark	large	0
pants	pastel	small	1
pants	pastel	medium	0
pants	pastel	large	1
pants	white	small	3
pants	white	medium	0
pants	white	large	2
shirt	dark	small	2
shirt	dark	medium	6
shirt	dark	large	6
shirt	pastel	small	4
shirt	pastel	medium	1
shirt	pastel	large	2
shirt	white	small	17
shirt	white	medium	1
shirt	white	large	10
skirt	dark	small	2
skirt	dark	medium	5
...	...	...	...
...	...	...	...

TÉCNICO LISBOA

Tabulação Cruzada (Cross-Tab, Pivot Table)

- Os valores dos atributos de uma das dimensões formam os cabeçalhos das linhas
- Os valores dos atributos de outra dimensão foram os cabeçalhos das colunas
- Atributos de outras dimensões são mostrados no topo (descritivos da tabela)
- Os valores em cada célula são (agregações) dos valores dos atributos de dimensão que especificam a célula

clothes_size **all**

item_name \ *color*	dark	pastel	white	total
skirt	8	35	10	53
dress	20	10	5	35
shirt	14	7	28	49
pants	20	2	5	27
total	62	54	48	164

cross-tab de *sales* por *item_name* e *color*

Tabulação Cruzada (Cross-Tab, Pivot Table)

- Tabelas Pivot são uma ferramenta de análise popular em folhas de cálculo (e.g. Excel, Google Sheets, OpenOffice Sheets)
  - Mas designam qualquer tabela que summarize/agregue dados

Google: “pivot tables tutorial youtube”

clothes_size **all**

item_name	*color* dark	pastel	white	total
skirt	8	35	10	53
dress	20	10	5	35
shirt	14	7	28	49
pants	20	2	5	27
total	62	54	48	164

cross-tab de *sales* por *item_name* e *color*

Tabulação Cruzada (Cross-Tab, Pivot Table)

- Tabelas Pivot são uma ferramenta de análise popular em folhas de cálculo (e.g. Excel, Google Sheets, OpenOffice Sheets)
  - Mas designam qualquer tabela que summarize/agregue dados

Google: “pivot tables tutorial youtube”

clothes_size **all**

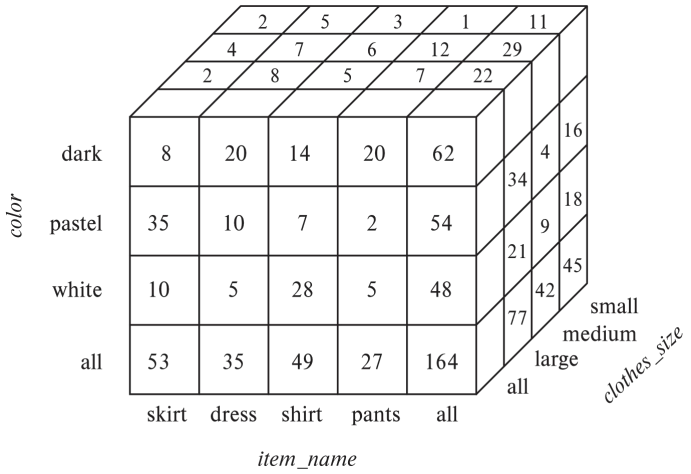
item_name	*color* dark	pastel	white	total
skirt	8	35	10	53
dress	20	10	5	35
shirt	14	7	28	49
pants	20	2	5	27
total	62	54	48	164

cross-tab de *sales* por *item_name* e *color*

TÉCNICO LISBOA

(Hiper)Cubo de Dados

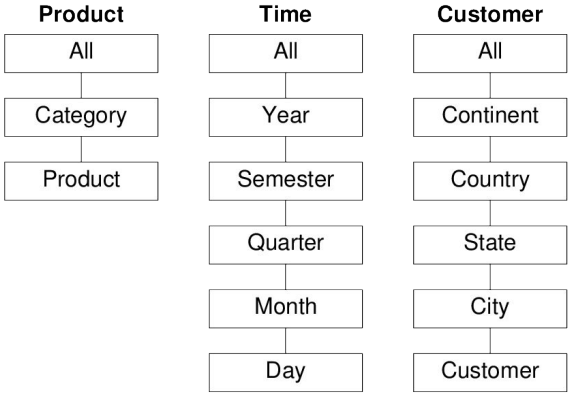
- Generalização multidimensional de um cross-tab
- Geralmente chamamos apenas cubo, mesmo que tenha mais de 3 dimensões
- Tipicamente inclui as agregações por linha, coluna, face
- Cross-tab é uma vista (face) do cubo



cubo de dados de 3 dimensões

OLAP

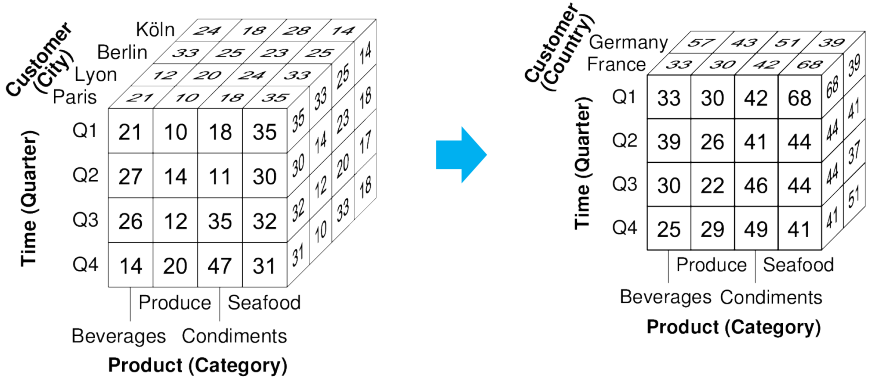
- Dimensões tipicamente podem ser organizadas hierarquicamente
  - Podemos ter cubos com diferentes granularidades nas dimensões
  - Podemos alterar a granularidade dos cubos:
    - Rollup: diminuir a granularidade
    - Drill-down: aumentar a granularidade



OLAP

Rollup: diminuir a granularidade do cubo, subindo na hierarquia de uma das dimensões

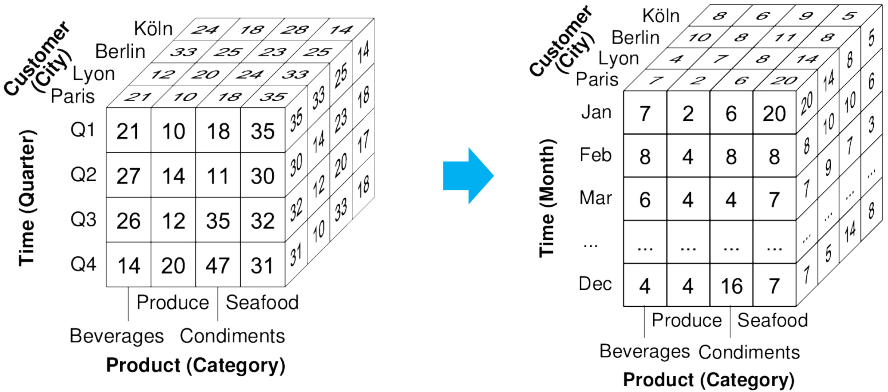
- E.g. rollup de cidade para país



OLAP

Drill-down: aumentar a granularidade, descendo na hierarquia de uma das dimensões

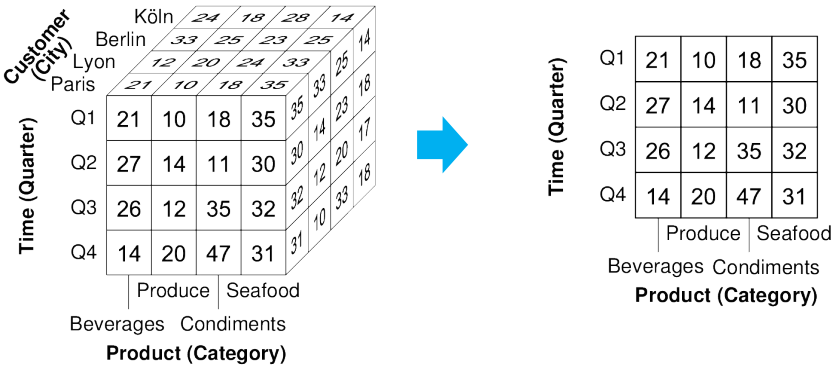
- E.g. drill-down de trimestre para mês



OLAP

Slice: restrição por igualdade em uma ou mais dimensões

- E.g. slice em Paris



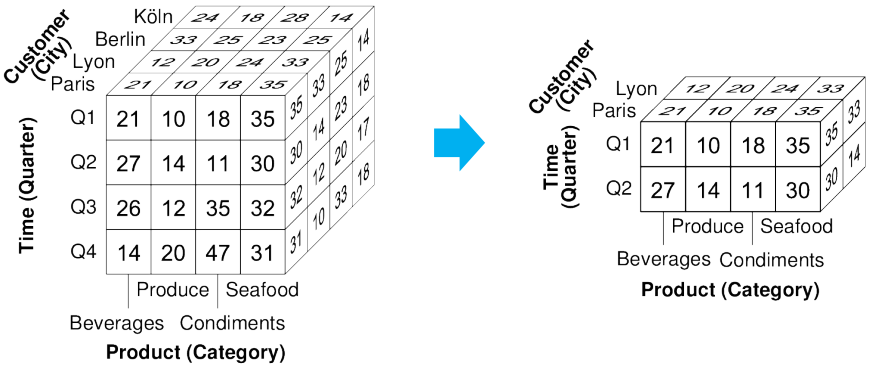
Time (Quarter)	Beverages	Condiments	Produce	Seafood
Q1	21	10	18	35
Q2	27	14	11	30
Q3	26	12	35	32
Q4	14	20	47	31

Product (Category)

OLAP

Dice: restrição por intervalo em uma ou mais dimensões

- E.g. dice em cidade {Paris, Lyon} e trimestre {Q1, Q2}



OLAP

- Tabulação Cruzada pode ser estendida com hierarquia
  - Com rollup ou drill-down

clothes_size: **all**

category	*item_name*	*color*				
		dark	pastel	white	total	
womenswear	skirt	8	8	10	53	
	dress	20	20	5	35	
	subtotal	28	28	15		88
menswear	pants	14	14	28	49	
	shirt	20	20	5	27	
	subtotal	34	34	33		76
total		62	62	48		164

Cross-tab com rollup item_name □ category

OLAP

- **Pivoting / Cross-Tabulating:**
  - Transformação de duas dimensões do array multidimensional, em que uma dimensão passa a coluna
  - Tipicamente com agregação seletiva em múltiplas dimensões
- **Slicing & Dicing:**
  - Restrições por igualdade e intervalo numa ou mais dimensões

OLAP no Modelo Relacional

- Cross-tabs, cubos, etc, podem ser representados como relações
  - Usando “all” para representar agregados
- O standard SQL usa NULL em vez de “all”
  - Funciona com qualquer tipo de dados
  - Mas pode gerar confusão com NULLs nos dados

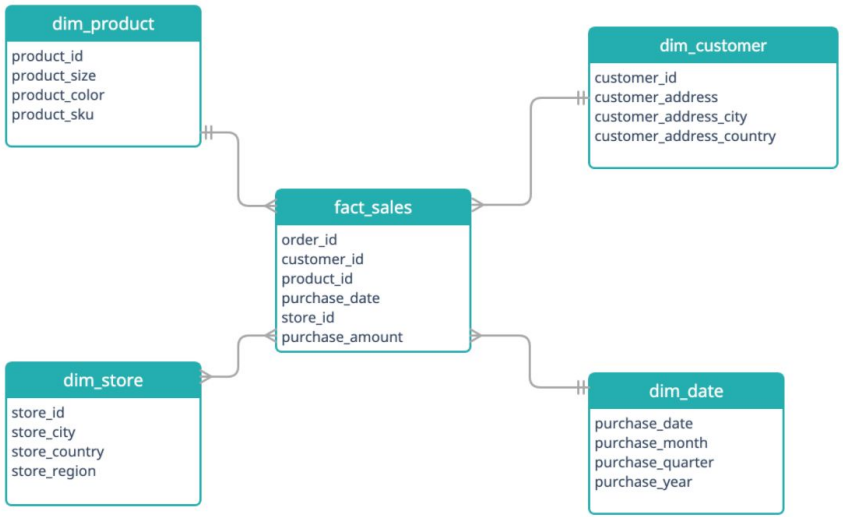
item_name	*color*	*clothes_size*	*quantity*
skirt	dark	**all**	8
skirt	pastel	**all**	35
skirt	white	**all**	10
skirt	**all**	**all**	53
dress	dark	**all**	20
dress	pastel	**all**	10
dress	white	**all**	5
dress	**all**	**all**	35
shirt	dark	**all**	14
shirt	pastel	**all**	7
shirt	white	**all**	28
shirt	**all**	**all**	49
pants	dark	**all**	20
pants	pastel	**all**	2
pants	white	**all**	5
pants	**all**	**all**	27
all	dark	**all**	62
all	pastel	**all**	54
all	white	**all**	48
all	**all**	**all**	164

TÉCNICO LISBOA

OLAP em SQL

Modelo em Estrela em SQL

- Modelo em esquema ou floco de neve podem ser implementados diretamente em SQL
  - São compostos por tabelas com atributos
- É geralmente trivial converter dados de uma DB operacional numa esquema em estrela

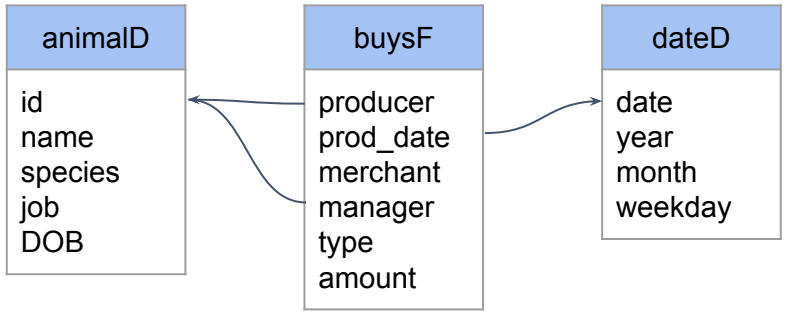


Exemplo: DB dos porcos em estrela

```
CREATE TABLE animalD(
    id INTEGER PRIMARY KEY,
    name VARCHAR(80),
    species VARCHAR(80),
    job VARCHAR(80),
    DOB DATE);

CREATE TABLE dateD(
    date DATE PRIMARY KEY,
    year NUMERIC(4,0),
    month NUMERIC(2,0),
    weekday NUMERIC(1,0));

CREATE TABLE buysF(
    producer INTEGER REFERENCES animal(id),
    prod_date DATE REFERENCES dateD(date),
    merchant NUMERIC(8),
    manager INTEGER REFERENCES animal(id),
    type CHAR(4),
    amount  FLOAT);
```



TÉCNICO LISBOA

Cross-Tab / Pivot em SQL

- Não é parte do standard SQL:
  - CROSSTAB em PostgreSQL
  - PIVOT em SQL Server

item_name	*color*	*clothes_size*	*quantity*
dress	dark	small	2
dress	dark	medium	6
dress	dark	large	12
dress	pastel	small	4
dress	pastel	medium	3
dress	pastel	large	3
dress	white	small	2
dress	white	medium	3
dress	white	large	0
pants	dark	small	14
pants	dark	medium	6
pants	dark	large	0
pants	pastel	small	1
pants	pastel	medium	0
pants	pastel	large	1
pants	white	small	3
pants	white	medium	0
pants	white	large	2
shirt	dark	small	2
shirt	dark	medium	6
shirt	dark	large	6
shirt	pastel	small	4
shirt	pastel	medium	1
shirt	pastel	large	2
shirt	white	small	17
shirt	white	medium	1
shirt	white	large	10
skirt	dark	small	2
skirt	dark	medium	5

item_name	*clothes_size*	*dark*	*pastel*	*white*
dress	small	2	4	2
dress	medium	6	3	3
dress	large	12	3	0
pants	small	14	1	3
pants	medium	6	0	0
pants	large	0	1	2
shirt	small	2	4	17
shirt	medium	6	1	1
shirt	large	6	2	10
skirt	small	2	11	2
skirt	medium	5	9	5
skirt	large	1	15	3

https://www.postgresql.org/docs/current/tablefunc.html

Cross-Tab / Pivot em SQL

- E.g. Listar todos os nomes de animais e a sua contagem por espécie de animal (incluindo porcos e não-porcos): **na BD operacional**

```
WITH
pignames AS (SELECT name, count(job) AS pigs FROM animal LEFT JOIN
    pig USING (id) GROUP BY name),
cownames AS (SELECT name, count(species) AS cows FROM animal LEFT JOIN
    (SELECT * FROM nonpig WHERE species = 'cow') np1 USING (id) GROUP BY name),
sheepnames AS (SELECT name, count(species) AS cows FROM animal LEFT JOIN
    (SELECT * FROM nonpig WHERE species = 'sheep') np1 USING (id) GROUP BY name),
goatnames AS (SELECT name, count(species) AS cows FROM animal LEFT JOIN
    (SELECT * FROM nonpig WHERE species = 'goat') np1 USING (id) GROUP BY name),
chickennames AS (SELECT name, count(species) AS cows FROM animal LEFT JOIN
    (SELECT * FROM nonpig WHERE species = 'chicken') np1 USING (id) GROUP BY name)
SELECT DISTINCT name, pigs, cows, sheep, goats, chickens FROM animal JOIN pignames
    USING (name) JOIN cownames USING (name) JOIN sheepnames USING (name)
    JOIN goatnames USING (name) JOIN chickennames USING (name);
```

Cross-Tab / Pivot em SQL

crosstab(text *source_sql*)

```
SELECT * FROM CROSSTAB('SELECT …')
    AS tablename(colname1 TYPE, colname2 TYPE, …);
```

- *source_sql* é um statement SQL (tipicamente SELECT) que produz os dados
  - Tem de retornar uma coluna *row_name*, uma coluna *category*, e uma coluna de valores
  - A cláusula AS tem de definir o output como uma coluna *row_name* (do mesmo tipo que a primeira coluna da query) seguido de N colunas de valores (do mesmo tipo que a terceira coluna da query)
    - O número e nome das colunas de valores são à vontade do utilizador

TÉCNICO LISBOA

https://www.postgresql.org/docs/current/tablefunc.html

Cross-Tab / Pivot em SQL

crosstab(text *source_sql*)

```
SELECT * FROM CROSSTAB('SELECT …')
    AS tablename(colname1 TYPE, colname2 TYPE, …);
```

- **Limitações:**
  - Trata os grupos de forma ingénua, designando um valor do grupo para uma coluna pela ordem com que ocorre nos dados
    - Não funciona bem se queremos que as colunas de valores correspondam a categorias específicas dos dados (em particular se houver grupos que não têm valores para algumas categorias)
  - Não permite colunas extra para além de *row_name*, *category* e *value*

https://www.postgresql.org/docs/current/tablefunc.html

Cross-Tab / Pivot em SQL

crosstab(*text source_sql*, text *category_sql*)

```
SELECT * FROM CROSSTAB('SELECT …', 'SELECT …')
    AS tablename(colname1 TYPE, colname2 TYPE, …);
```

- *category_sql* é uma statement SQL que produz a lista de categorias
  - Tem de retornar apenas uma coluna e pelo menos uma linha (categoria)
  - Não pode incluir duplicados
- *source_sql* pode conter uma ou mais colunas “extra”
  - A coluna *row_name* tem de ser a primeir
  - As colunas *category* e *value* têm de ser as duas últimas (nesta ordem)
  - Quaisquer outras colunas no meio são consideradas “extra”
  - Têm de ser listadas no AS pela mesma ordem

https://www.postgresql.org/docs/current/tablefunc.html

Cross-Tab / Pivot em SQL

```
 id |  name   | species | job |     dob
----+---------+---------+-----+------------
  1 | Daniel  | pig     |     |
  2 | Daniel  | pig     |     |
  3 | Daniel  | goat    |     |
  4 | Daniel  | goat    |     |
  5 | Daniel  | cow     |     |
  6 | Daniel  | sheep   |     |
  7 | Rita    | goat    |     |
  8 | Rita    | cow     |     |
  9 | Rita    | cow     |     |
 10 | Rita    | sheep   |     |
 11 | Rita    | sheep   |     |
 12 | Rita    | chicken |     |
```

```
SELECT * FROM CROSSTAB(
    'SELECT name, species, COUNT(*) FROM animalD
        GROUP BY name, species ORDER BY name, species',
    'SELECT DISTINCT species FROM animalD ORDER BY
    species')
AS names (name VARCHAR(80), chicken BIGINT, cow BIGINT,
goat BIGINT, pig BIGINT, sheep BIGINT);

   name   | chicken | cow | goat | pig | sheep
----------+---------+-----+------+-----+-------
 Daniel   |         |   1 |    2 |   2 |     1
 Rita     |       1 |   2 |    1 |     |     2
```

https://www.postgresql.org/docs/current/tablefunc.html

Agregações OLAP

- Agregações complexas permitindo queries OLAP
- Podem ser incluídas na cláusula GROUP BY:
  - GROUPING SETS
  - ROLLUP
  - CUBE
- Parte do standard SQL desde 1999

https://www.postgresql.org/docs/current/queries-table-expressions.html#QUERIES-GROUPING-SETS

Agregações OLAP

Tabela de factos (SP), que inclui
suppliers (S#) e *parts* (P#)

S#	P#	QTY
S1	P1	300
S1	P2	200
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200

Query OLAP típica:

- Qual o número total de peças enviadas por fornecedor, peça, fornecedor-peça, e global?

https://www.postgresql.org/docs/current/queries-table-expressions.html#QUERIES-GROUPING-SETS

Agregações OLAP

Query OLAP típica:

- Qual o número total de peças enviadas por fornecedor, peça, fornecedor-peça, e global?

```
SELECT * FROM
     (SELECT S#, P#, SUM(QTY) AS TOTQTY FROM SP GROUP BY (S#, P#)) AS both
     FULL JOIN
     (SELECT S#, SUM(QTY) AS TOTQTY FROM SP GROUP BY (S#)) AS s USING (S#,TOTQTY)
     FULL JOIN
     (SELECT P#, SUM(QTY) AS TOTQTY FROM SP GROUP BY (P#)) AS p USING (P#,TOTQTY)
     FULL JOIN
     (SELECT SUM(QTY) AS TOTQTY FROM SP GROUP BY ()) AS none USING (TOTQTY);
```

Agregações OLAP

- **GROUPING SETS:**
  - Aceita lista de sublistas de colunas ou expressões a usar como agrupadores
  - Cada sublista pode especificar zero ou mais colunas ou expressões
    - É interpretada como estando diretamente na cláusula GROUP BY
      - Sublistas são agregadas em paralelo e o resultado é a união de todas
    - Conjunto vazio resulta em agregação global
  - As colunas resultantes são a união de todas as colunas em todas as sublistas
    - Estão preenchidas com NULL para os agrupamentos que não as incluem

Agregações OLAP

- **GROUPING SETS:**

```
SELECT S#, P#, SUM(QTY) AS TOTQTY FROM SP GROUP BY GROUPING SETS ((S#), (P#), ());
```

S#	P#	QTY
S1	P1	300
S1	P2	200
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200

S#	P#	TOTQTY
S1	null	500
S2	null	700
S3	null	200
S4	null	200
null	P1	600
null	P2	1000
null	null	1600

Agregações OLAP

- **GROUPING SETS:**

```
SELECT name, species, COUNT(*) FROM animalD GROUP BY
GROUPING SETS ((name), (species), (name,species));

   name   | species | count
----------+---------+-------
 Rita     | cow     |   2
 Rita     | goat    |   1
 Rita     | sheep   |   2
 Daniel   | goat    |   2
 Daniel   | pig     |   2
 Daniel   | sheep   |   1
 Daniel   | cow     |   1
 Rita     | chicken |   1
 Rita     |         |   6
 Daniel   |         |   6
          | goat    |   3
          | cow     |   3
          | pig     |   2
          | chicken |   1
          | sheep   |   3
```

TÉCNICO LISBOA

Agregações OLAP

- **ROLLUP:**
  - Aceita lista de colunas ou expressões
    - Elementos em sublistas são tratadas como elementos da lista
  - Representa o GROUPING SET correspondente à lista unido com GROUPING SETS de todos os prefixos da lista, incluindo a lista vazia
    - Para uso em ROLLUPs dimensionais (e.g. day > month > year)

```
ROLLUP ( e1, e2, e3 )
```

```
GROUPING SETS (
    ( e1, e2, e3 ),
    ( e1, e2 ),
    ( e1 ),
    ( )
)
```

Agregações OLAP

- **ROLLUP:**

```
SELECT name, species, COUNT(*) FROM animalD GROUP BY
ROLLUP (name, species);

    name   | species | count
  ---------+---------+-------
           |         |   12
    Rita   | cow     |    2
    Rita   | goat    |    1
    Rita   | sheep   |    2
    Daniel | goat    |    2
    Daniel | pig     |    2
    Daniel | sheep   |    1
    Daniel | cow     |    1
    Rita   | chicken |    1
    Rita   |         |    6
    Daniel |         |    6
```

ROLLUP não é simétrico!

Agregações OLAP

- **CUBE:**
  - Aceita lista de colunas ou expressões
    - Elementos em sublistas são tratadas como elementos da lista
  - Representa o GROUPING SET correspondente à lista unido com os GROUPING SETS de todos as sublistas possíveis (i.e., power set)

```
CUBE ( e1, e2, e3 )
```

```
GROUPING SETS (
    ( e1, e2, e3 ),
    ( e1, e2     ),
    ( e1,     e3 ),
    ( e1         ),
    (     e2, e3 ),
    (     e2     ),
    (         e3 ),
    (            )
)
```

Agregações OLAP

- **CUBE:**

```
SELECT name, species, COUNT(*) FROM animalD GROUP BY CUBE (name,species);

   name   | species | count
----------+---------+-------
          |         |    12
 Rita     | cow     |     2
 Rita     | goat    |     1
 Rita     | sheep   |     2
 Daniel   | goat    |     2
 Daniel   | pig     |     2
 Daniel   | sheep   |     1
 Daniel   | cow     |     1
 Rita     | chicken |     1
 Rita     |         |     6
 Daniel   |         |     6
          | goat    |     3
          | cow     |     3
          | pig     |     2
          | chicken |     1
          | sheep   |     3
```

TÉCNICO LISBOA

Agregações OLAP

- CUBE e ROLLUP podem ser usados diretamente no GROUP BY ou dentro de GROUPING SETS
- Se uma cláusula GROUPING SETS for colocada dentro de outra, o resultado é o mesmo do que todos os elementos da cláusula interna estarem listados na externa
- Se múltiplos GROUPING SETS forem declarados num único GROUP BY, a lista final de GROUPING SETS é o cross-product das listas individuais
    - Isto pode levar a duplicados no conjunto final de GROUPING SETS
    - DISTINCT pode ser especificado no GROUP BY para evitá-lo

TÉCNICO LISBOA