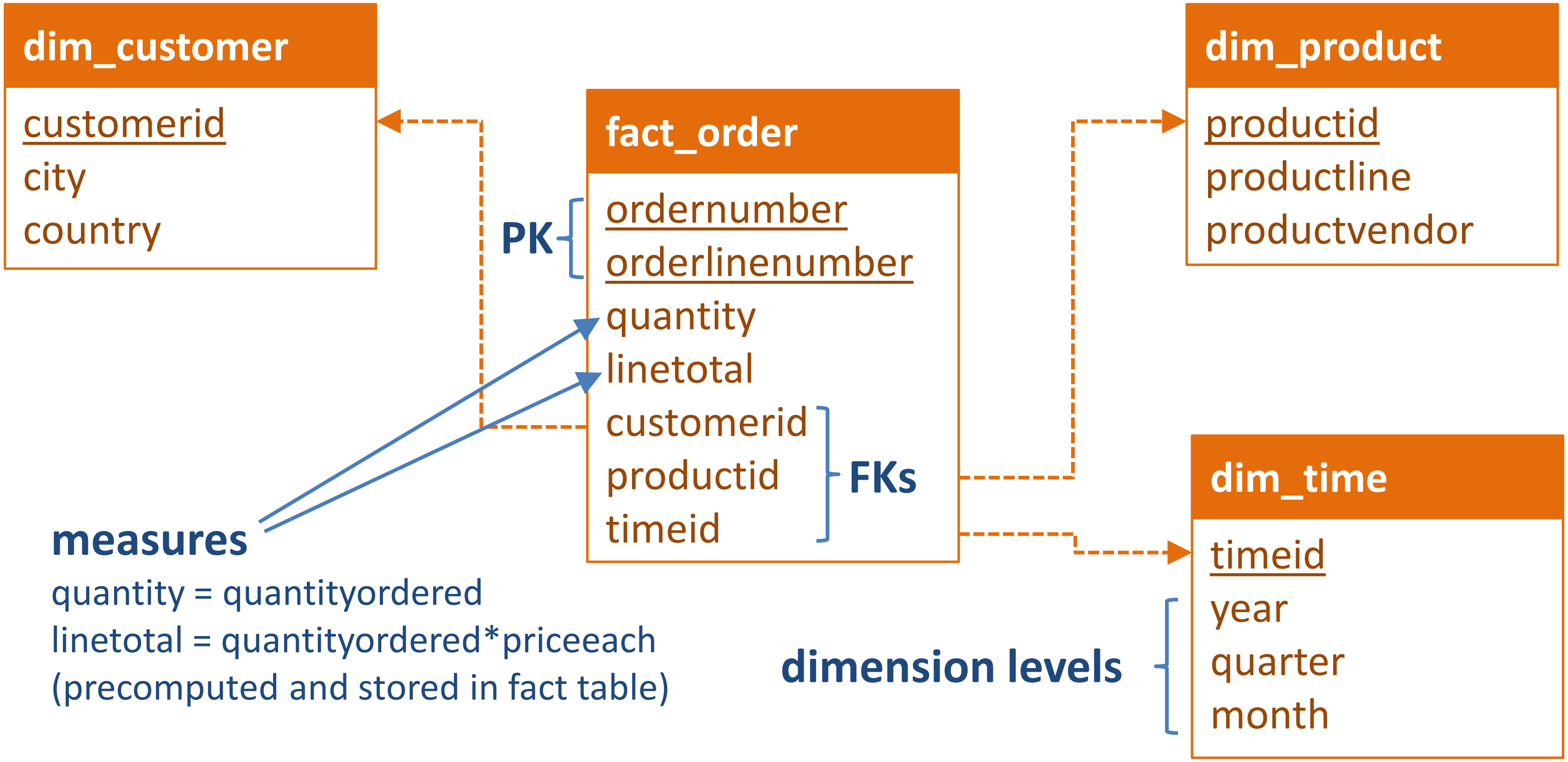
Data Analysis and Integration

OLAP operations

Data warehousing

- Star schema

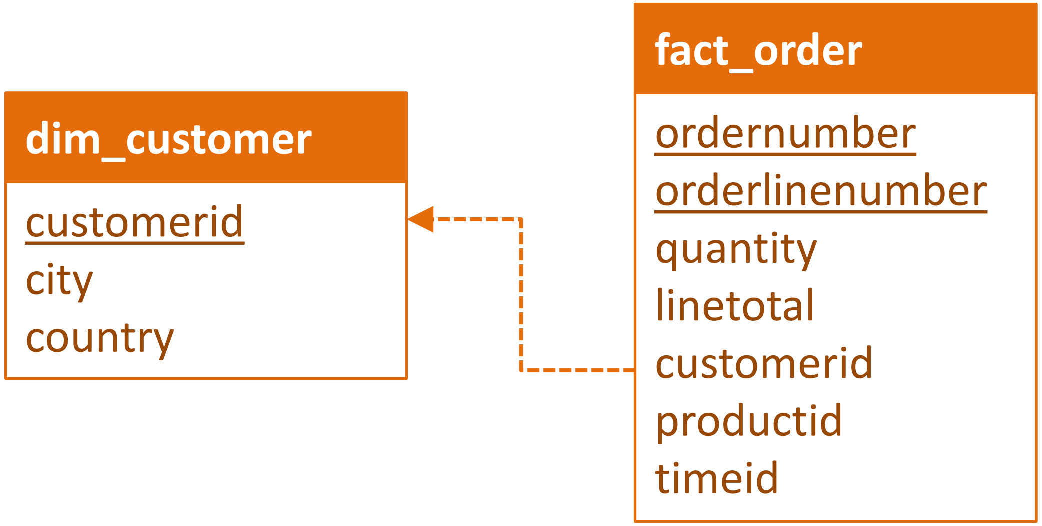


quantity = quantityordered
linetotal = quantityordered*priceeach
(precomputed and stored in fact table)

Data warehousing

- Sales by customer country

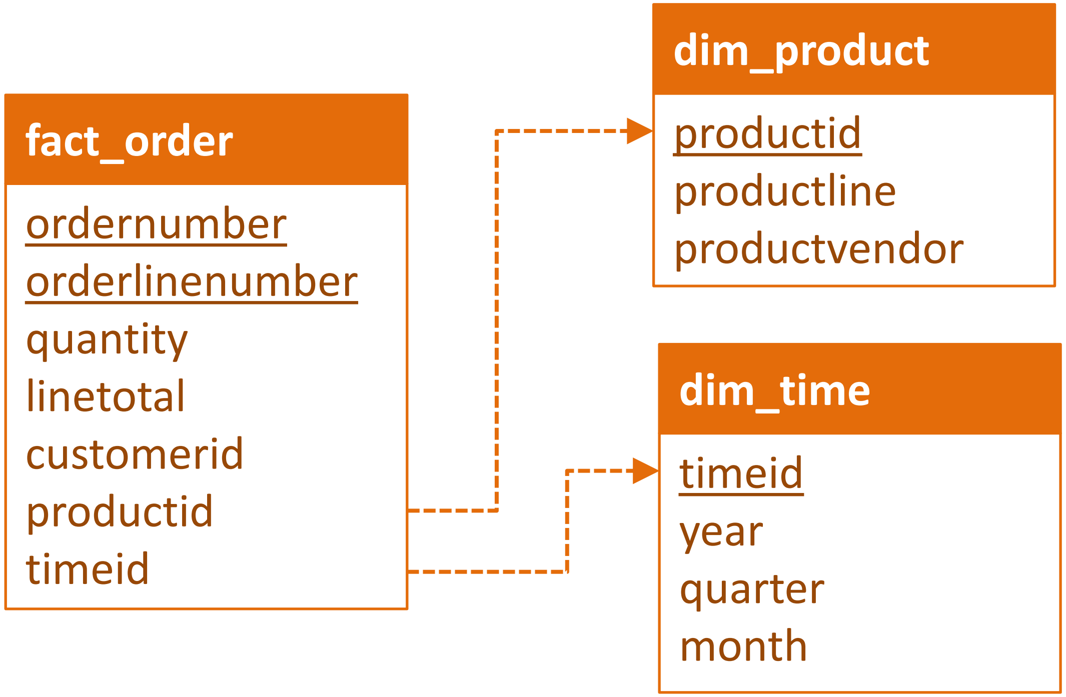
```
select country, sum(linetotal) as sales
from fact_order natural join dim_customer
group by country;
```



Data warehousing

- Sales by product line and year

```sql
select productline, year, sum(linetotal) as sales
from fact_order natural join dim_product natural join dim_time
group by productline, year;
```



fact_order

ordernumber
orderlinenumber
quantity
linetotal
customerid
productid
timeid

dim_product

productid
productline
productvendor

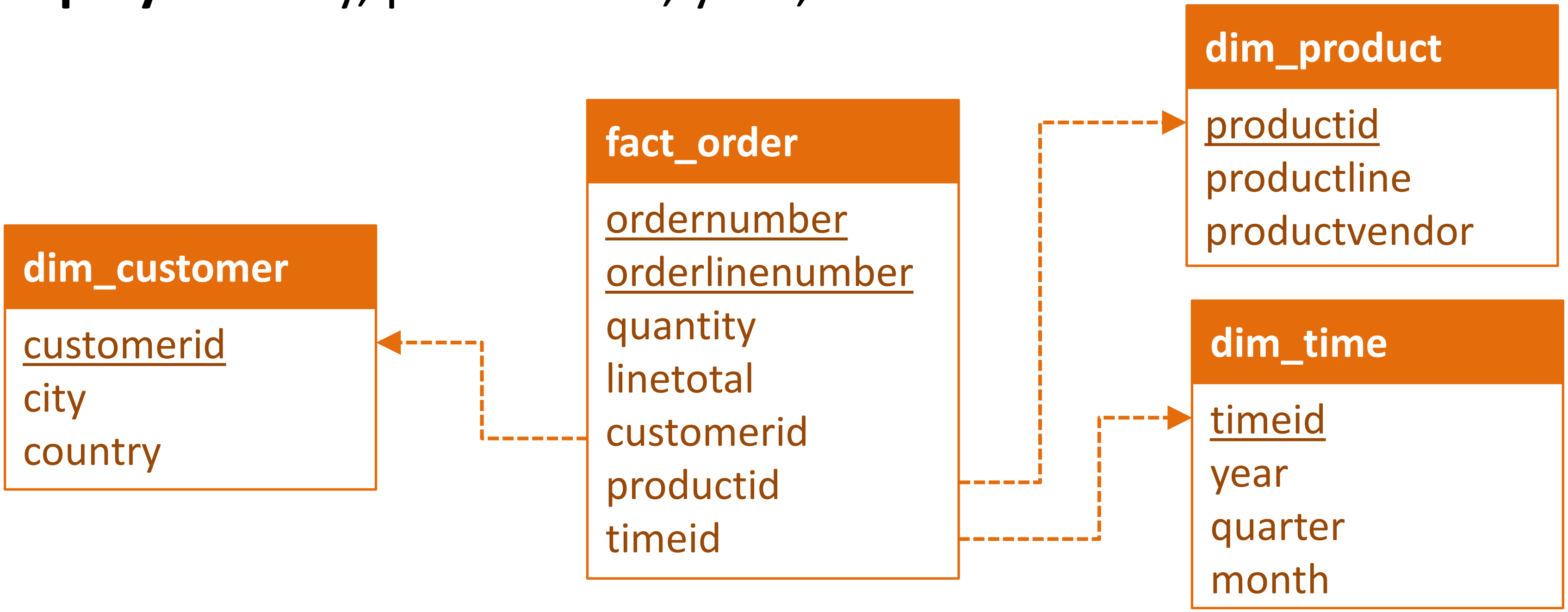
dim_time

timeid
year
quarter
month

Data warehousing

- Sales by customer country, product line and year

```
select country, productline, year, sum(linetotal) as sales
from fact_order natural join dim_customer natural join dim_product
        natural join dim_time
group by country, productline, year;
```



Dimension levels

- Each **dimension** can have different **levels**
  - customer (city; country)
  - product (productline; productvendor)
  - time (year; quarter; month)

Drill-down

- Going from a higher to a lower level is called **drill-down**
  - e.g. sales by customer country → sales by customer city

```
select country, sum(linetotal) as sales
from fact_order natural join dim_customer
group by country;
```

⇩

```
select city, sum(linetotal) as sales
from fact_order natural join dim_customer
group by city;
```

⚠

Drill-down

- Going from a higher to a lower level is called **drill-down**
  - e.g. sales by customer country → sales by customer city

```
select country, sum(linetotal) as sales
from fact_order natural join dim_customer
group by country;
```

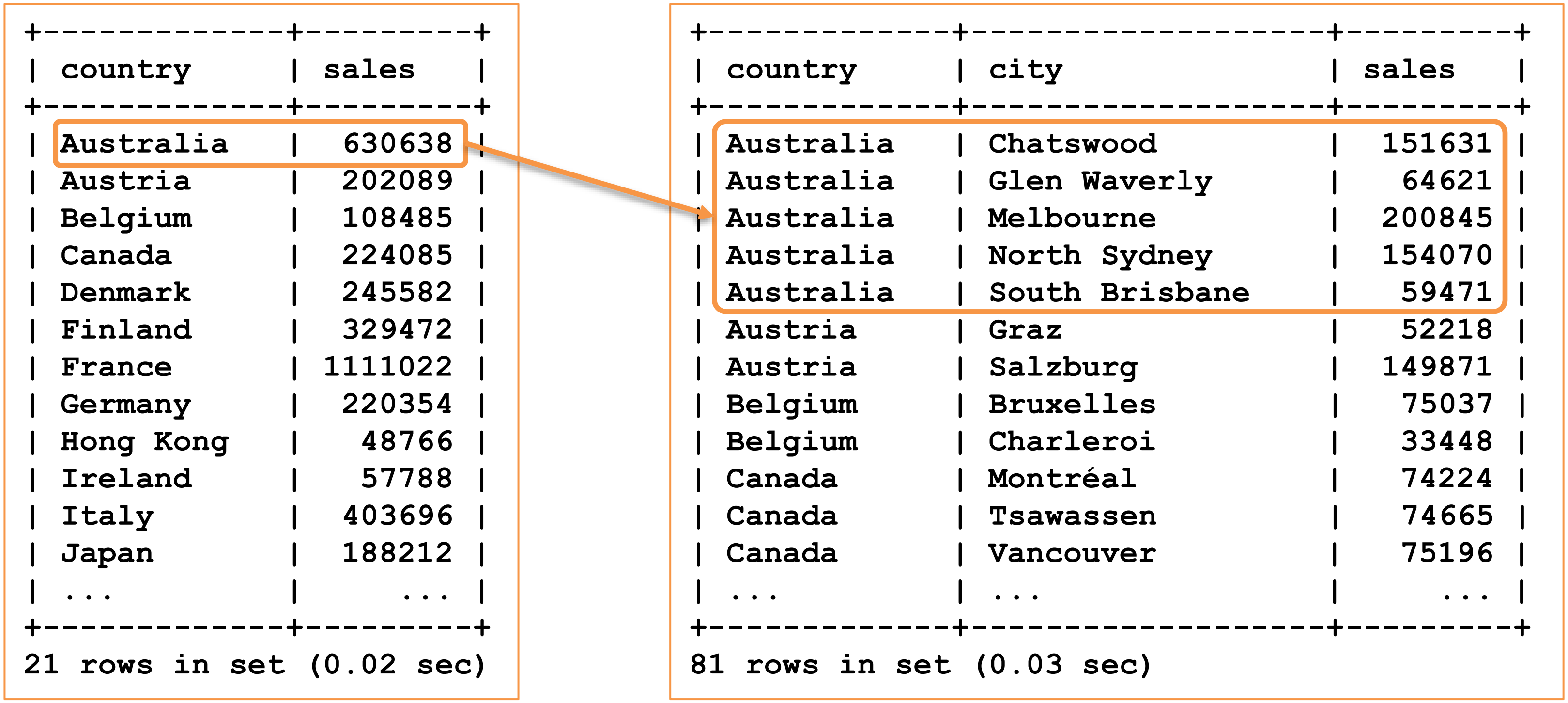
⇩

```
select country, city, sum(linetotal) as sales
from fact_order natural join dim_customer
group by country, city;
```

✓

Drill-down

- Going from a higher to a lower level is called **drill-down**
  - e.g. sales by customer country → sales by customer city



```
+-------------+---------+
| country     | sales   |
+-------------+---------+
| Australia   |  630638 |
| Austria     |  202089 |
| Belgium     |  108485 |
| Canada      |  224085 |
| Denmark     |  245582 |
| Finland     |  329472 |
| France      | 1111022 |
| Germany     |  220354 |
| Hong Kong   |   48766 |
| Ireland     |   57788 |
| Italy       |  403696 |
| Japan       |  188212 |
| ...         |     ... |
+-------------+---------+
21 rows in set (0.02 sec)
```

```
+-------------+-------------------+---------+
| country     | city              | sales   |
+-------------+-------------------+---------+
| Australia   | Chatswood         |  151631 |
| Australia   | Glen Waverly      |   64621 |
| Australia   | Melbourne         |  200845 |
| Australia   | North Sydney      |  154070 |
| Australia   | South Brisbane    |   59471 |
| Austria     | Graz              |   52218 |
| Austria     | Salzburg          |  149871 |
| Belgium     | Bruxelles         |   75037 |
| Belgium     | Charleroi         |   33448 |
| Canada      | Montréal          |   74224 |
| Canada      | Tsawassen         |   74665 |
| Canada      | Vancouver         |   75196 |
| ...         | ...               |     ... |
+-------------+-------------------+---------+
81 rows in set (0.03 sec)
```

Drill-down

- Another example of **drill-down**
  - e.g. sales by year → sales by month

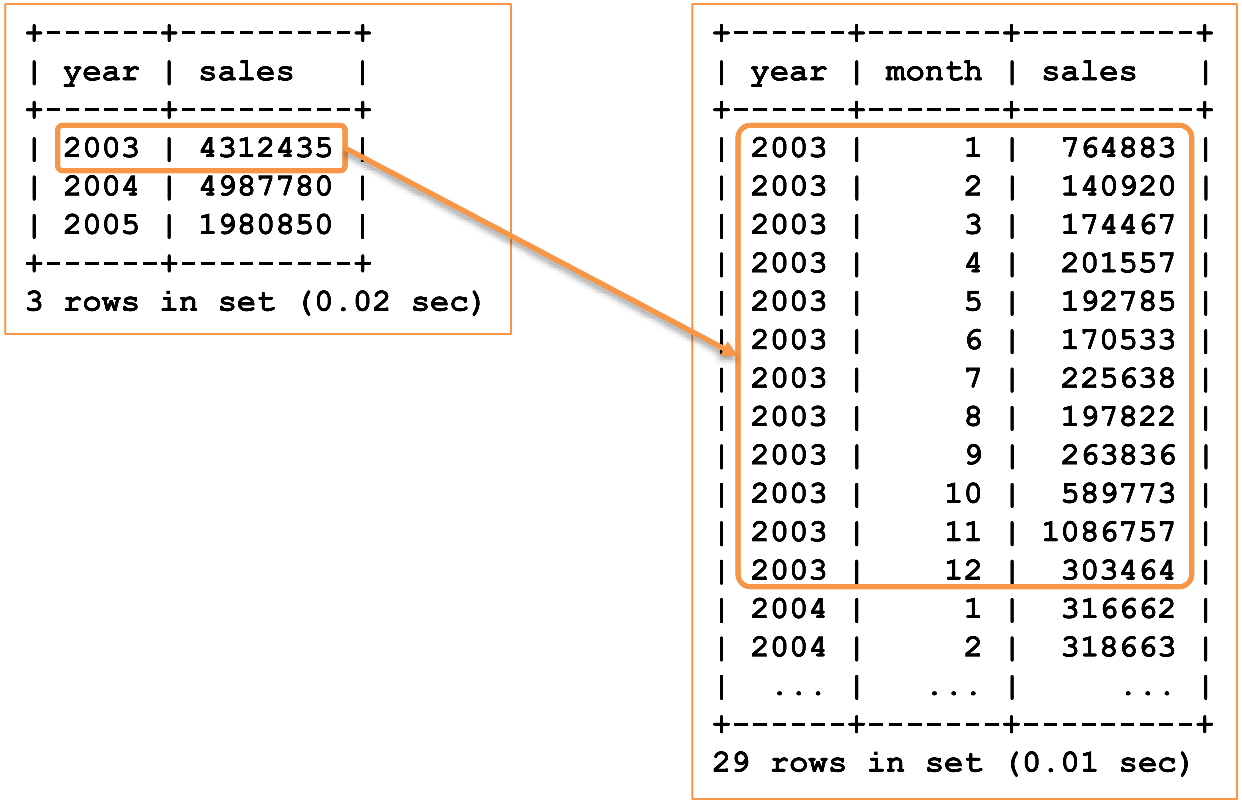
```
select year, sum(linetotal) as sales
from fact_order natural join dim_time
group by year;
```

⇩

```
select year, month, sum(linetotal) as sales
from fact_order natural join dim_time
group by year, month;
```

Drill-down

- Another example of **drill-down**
  - e.g. sales by year → sales by month



```
+------+---------+
| year | sales   |
+------+---------+
| 2003 | 4312435 |
| 2004 | 4987780 |
| 2005 | 1980850 |
+------+---------+
3 rows in set (0.02 sec)
```

```
+------+-------+---------+
| year | month | sales   |
+------+-------+---------+
| 2003 |     1 |  764883 |
| 2003 |     2 |  140920 |
| 2003 |     3 |  174467 |
| 2003 |     4 |  201557 |
| 2003 |     5 |  192785 |
| 2003 |     6 |  170533 |
| 2003 |     7 |  225638 |
| 2003 |     8 |  197822 |
| 2003 |     9 |  263836 |
| 2003 |    10 |  589773 |
| 2003 |    11 | 1086757 |
| 2003 |    12 |  303464 |
| 2004 |     1 |  316662 |
| 2004 |     2 |  318663 |
|  ... |   ... |     ... |
+------+-------+---------+
29 rows in set (0.01 sec)
```

Roll-up

- Going from a lower to a higher level is called **roll-up**
  - e.g. sales by customer city → sales by customer country

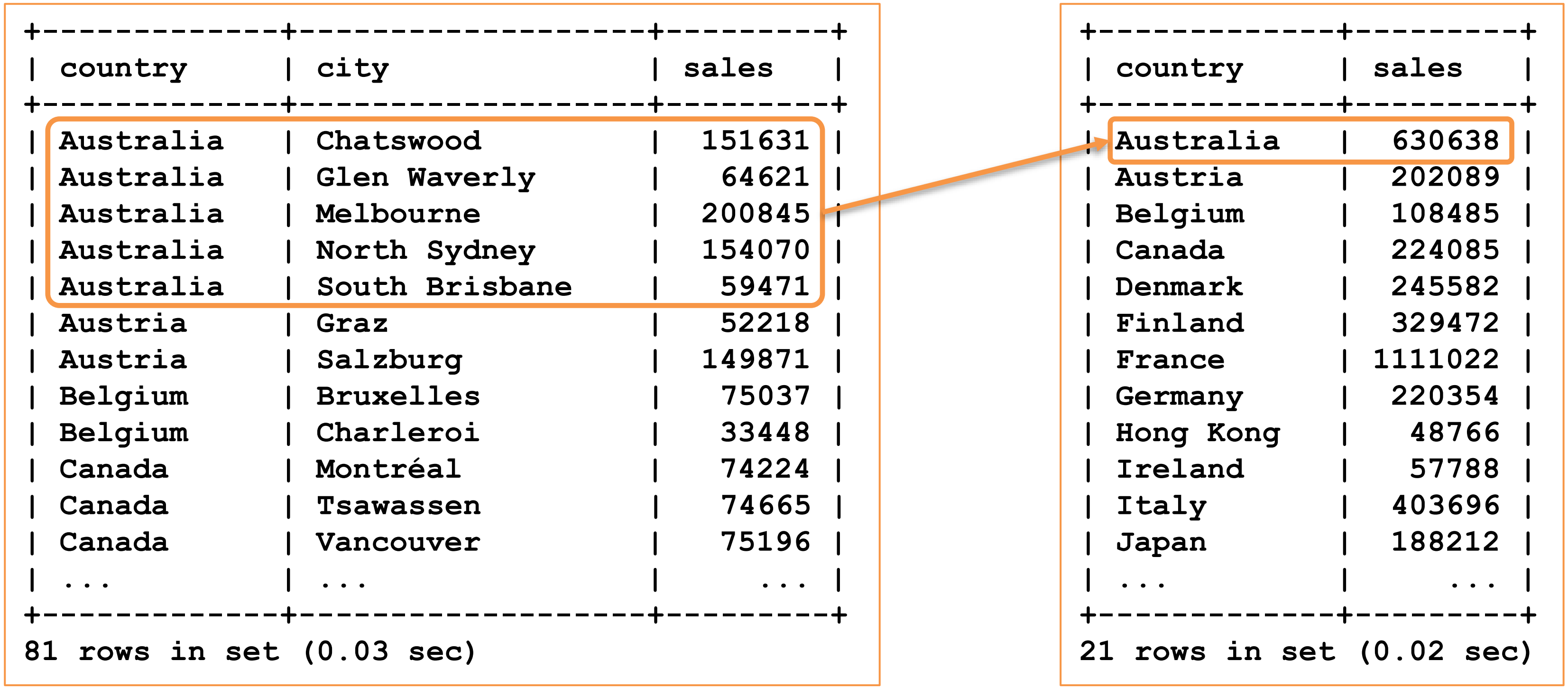
```
select country, city, sum(linetotal) as sales
from fact_order natural join dim_customer
group by country, city;
```

⇩

```
select country, sum(linetotal) as sales
from fact_order natural join dim_customer
group by country;
```

Roll-up

- Going from a lower to a higher level is called **roll-up**
  - e.g. sales by customer city → sales by customer country



```
+-------------+--------------------+---------+
| country     | city               | sales   |
+-------------+--------------------+---------+
| Australia   | Chatswood          |  151631 |
| Australia   | Glen Waverly       |   64621 |
| Australia   | Melbourne          |  200845 |
| Australia   | North Sydney       |  154070 |
| Australia   | South Brisbane     |   59471 |
| Austria     | Graz               |   52218 |
| Austria     | Salzburg           |  149871 |
| Belgium     | Bruxelles          |   75037 |
| Belgium     | Charleroi          |   33448 |
| Canada      | Montréal           |   74224 |
| Canada      | Tsawassen          |   74665 |
| Canada      | Vancouver          |   75196 |
| ...         | ...                |     ... |
+-------------+--------------------+---------+
81 rows in set (0.03 sec)
```

```
+-------------+---------+
| country     | sales   |
+-------------+---------+
| Australia   |  630638 |
| Austria     |  202089 |
| Belgium     |  108485 |
| Canada      |  224085 |
| Denmark     |  245582 |
| Finland     |  329472 |
| France      | 1111022 |
| Germany     |  220354 |
| Hong Kong   |   48766 |
| Ireland     |   57788 |
| Italy       |  403696 |
| Japan       |  188212 |
| ...         |     ... |
+-------------+---------+
21 rows in set (0.02 sec)
```

Roll-up

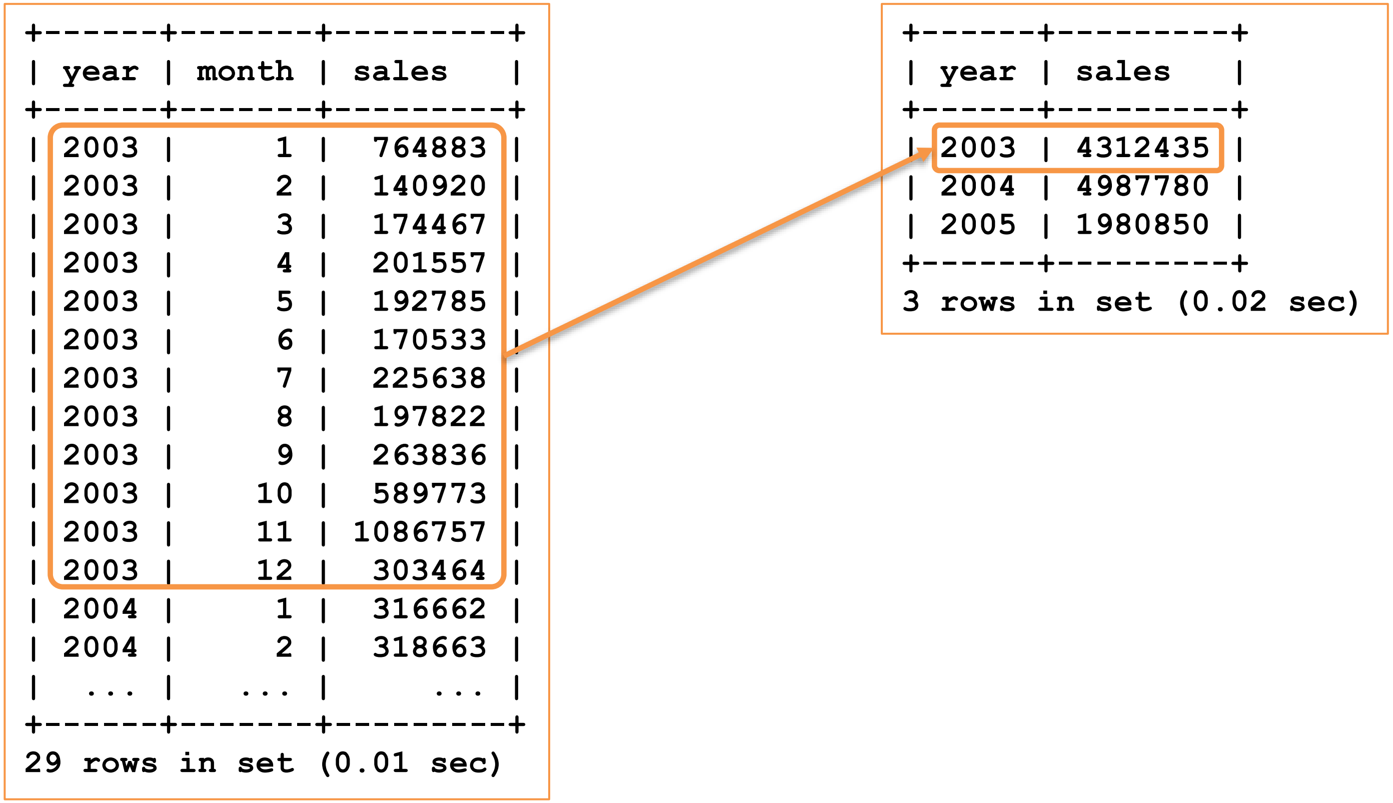
- Another example of **roll-up**
  - e.g. sales by month → sales by year

```
select year, month, sum(linetotal) as sales
from fact_order natural join dim_time
group by year, month;
```

```
select year, sum(linetotal) as sales
from fact_order natural join dim_time
group by year;
```

Roll-up

- Another example of **roll-up**
  - e.g. sales by month → sales by year



```
+------+-------+---------+
| year | month | sales   |
+------+-------+---------+
| 2003 |     1 |  764883 |
| 2003 |     2 |  140920 |
| 2003 |     3 |  174467 |
| 2003 |     4 |  201557 |
| 2003 |     5 |  192785 |
| 2003 |     6 |  170533 |
| 2003 |     7 |  225638 |
| 2003 |     8 |  197822 |
| 2003 |     9 |  263836 |
| 2003 |    10 |  589773 |
| 2003 |    11 | 1086757 |
| 2003 |    12 |  303464 |
| 2004 |     1 |  316662 |
| 2004 |     2 |  318663 |
|  ... |   ... |     ... |
+------+-------+---------+
29 rows in set (0.01 sec)
```

```
+------+---------+
| year | sales   |
+------+---------+
| 2003 | 4312435 |
| 2004 | 4987780 |
| 2005 | 1980850 |
+------+---------+
3 rows in set (0.02 sec)
```

Slice

- Selecting a particular value of dimension level is a **slice**
  - e.g. sales by product line in 2003

```
select productline, year, sum(linetotal) as sales
from fact_order natural join dim_product natural join dim_time
group by productline, year;
```

⇩

```
select productline, year, sum(linetotal) as sales
from fact_order natural join dim_product natural join dim_time
group by productline, year
having year = 2003;
```

⚠

Slice

- Selecting a particular value of dimension level is a **slice**
  - e.g. sales by product line in 2003

```sql
select productline, year, sum(linetotal) as sales
from fact_order natural join dim_product natural join dim_time
group by productline, year;
```

⇩

```sql
select productline, year, sum(linetotal) as sales
from fact_order natural join dim_product natural join dim_time
where year = 2003
group by productline, year;
```

✓

Slice

- Selecting a particular value of dimension level is a **slice**
  - e.g. sales by product line in 2003

```
+------------------+------+---------+
| productline      | year | sales   |
+------------------+------+---------+
| Classic Cars     | 2003 | 1513998 |
| Classic Cars     | 2004 | 1837904 |
| Classic Cars     | 2005 |  738587 |
| Motorcycles      | 2003 |  397392 |
| Motorcycles      | 2004 |  590632 |
| Motorcycles      | 2005 |  286327 |
| Planes           | 2003 |  347924 |
| Planes           | 2004 |  529129 |
| Planes           | 2005 |  200077 |
| Ships            | 2003 |  244652 |
| Ships            | 2004 |  375498 |
| Ships            | 2005 |  128219 |
| Trains           | 2003 |   72857 |
| Trains           | 2004 |  124885 |
| Trains           | 2005 |   36920 |
| ...              |  ... |     ... |
+------------------+------+---------+
21 rows in set (0.04 sec)
```

```
+------------------+------+---------+
| productline      | year | sales   |
+------------------+------+---------+
| Classic Cars     | 2003 | 1513998 |
| Motorcycles      | 2003 |  397392 |
| Planes           | 2003 |  347924 |
| Ships            | 2003 |  244652 |
| Trains           | 2003 |   72857 |
| Trucks and Buses | 2003 |  420523 |
| Vintage Cars     | 2003 | 1315089 |
+------------------+------+---------+
7 rows in set (0.01 sec)
```

Dice

- Applying multiple slicing conditions is called a **dice**
  - e.g. sales of Motorcycles in 2003 and 2004

```
select productline, year, sum(linetotal) as sales
from fact_order natural join dim_product natural join dim_time
group by productline, year;
```

⇩

```
select productline, year, sum(linetotal) as sales
from fact_order natural join dim_product natural join dim_time
where productline = 'Motorcycles' and year in (2003, 2004)
group by productline, year;
```

Dice

- Applying multiple slicing conditions is called a **dice**
  - e.g. sales of Motorcycles in 2003 and 2004

```
+-------------------+------+---------+
| productline       | year | sales   |
+-------------------+------+---------+
| Classic Cars      | 2003 | 1513998 |
| Classic Cars      | 2004 | 1837904 |
| Classic Cars      | 2005 |  738587 |
| Motorcycles       | 2003 |  397392 |
| Motorcycles       | 2004 |  590632 |
| Motorcycles       | 2005 |  286327 |
| Planes            | 2003 |  347924 |
| Planes            | 2004 |  529129 |
| Planes            | 2005 |  200077 |
| Ships             | 2003 |  244652 |
| Ships             | 2004 |  375498 |
| Ships             | 2005 |  128219 |
| Trains            | 2003 |   72857 |
| Trains            | 2004 |  124885 |
| Trains            | 2005 |   36920 |
| ...               |  ... |     ... |
+-------------------+------+---------+
21 rows in set (0.04 sec)
```

```
+-------------+------+--------+
| productline | year | sales  |
+-------------+------+--------+
| Motorcycles | 2003 | 397392 |
| Motorcycles | 2004 | 590632 |
+-------------+------+--------+
2 rows in set (0.01 sec)
```

Pivot

- Changing the order of dimensions is called **pivot**
  - e.g. sales by product line, year → sales by year, product line

```
select productline, year, sum(linetotal) as sales
from fact_order natural join dim_product natural join dim_time
group by productline, year;
```

```
select year, productline, sum(linetotal) as sales
from fact_order natural join dim_product natural join dim_time
group by year, productline;
```

Pivot

- Changing the order of dimensions is called **pivot**
  - e.g. sales by product line, year → sales by year, product line

```
+------------------+------+---------+
| productline      | year | sales   |
+------------------+------+---------+
| Classic Cars     | 2003 | 1513998 |
| Classic Cars     | 2004 | 1837904 |
| Classic Cars     | 2005 |  738587 |
| Motorcycles      | 2003 |  397392 |
| Motorcycles      | 2004 |  590632 |
| Motorcycles      | 2005 |  286327 |
| Planes           | 2003 |  347924 |
| Planes           | 2004 |  529129 |
| Planes           | 2005 |  200077 |
| Ships            | 2003 |  244652 |
| Ships            | 2004 |  375498 |
| Ships            | 2005 |  128219 |
| Trains           | 2003 |   72857 |
| Trains           | 2004 |  124885 |
| Trains           | 2005 |   36920 |
| ...              |  ... |     ... |
+------------------+------+---------+
21 rows in set (0.04 sec)
```

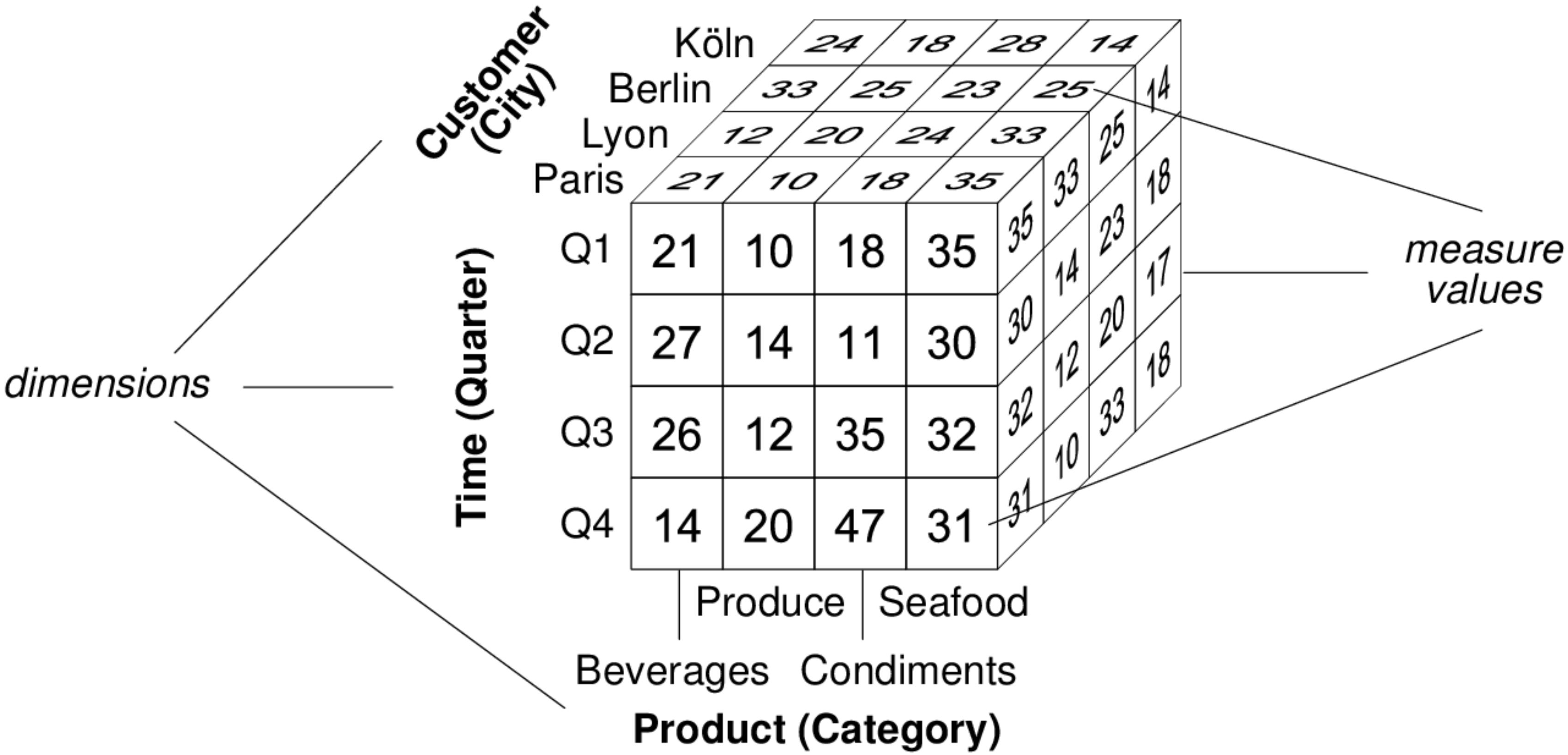
```
+------+------------------+---------+
| year | productline      | sales   |
+------+------------------+---------+
| 2003 | Classic Cars     | 1513998 |
| 2003 | Motorcycles      |  397392 |
| 2003 | Planes           |  347924 |
| 2003 | Ships            |  244652 |
| 2003 | Trains           |   72857 |
| 2003 | Trucks and Buses |  420523 |
| 2003 | Vintage Cars     | 1315089 |
| 2004 | Classic Cars     | 1837904 |
| 2004 | Motorcycles      |  590632 |
| 2004 | Planes           |  529129 |
| 2004 | Ships            |  375498 |
| 2004 | Trains           |  124885 |
| 2004 | Trucks and Buses |  532024 |
| 2004 | Vintage Cars     |  997708 |
| 2005 | Classic Cars     |  738587 |
|  ... | ...              |     ... |
+------+------------------+---------+
21 rows in set (0.04 sec)
```

OLAP operations

- Typical analytical operations
  - **drill-down** and **roll-up** between levels
  - **slice** and **dice** to select particular values
  - **pivot** to rotate dimensions
  - etc.

Multidimensional model

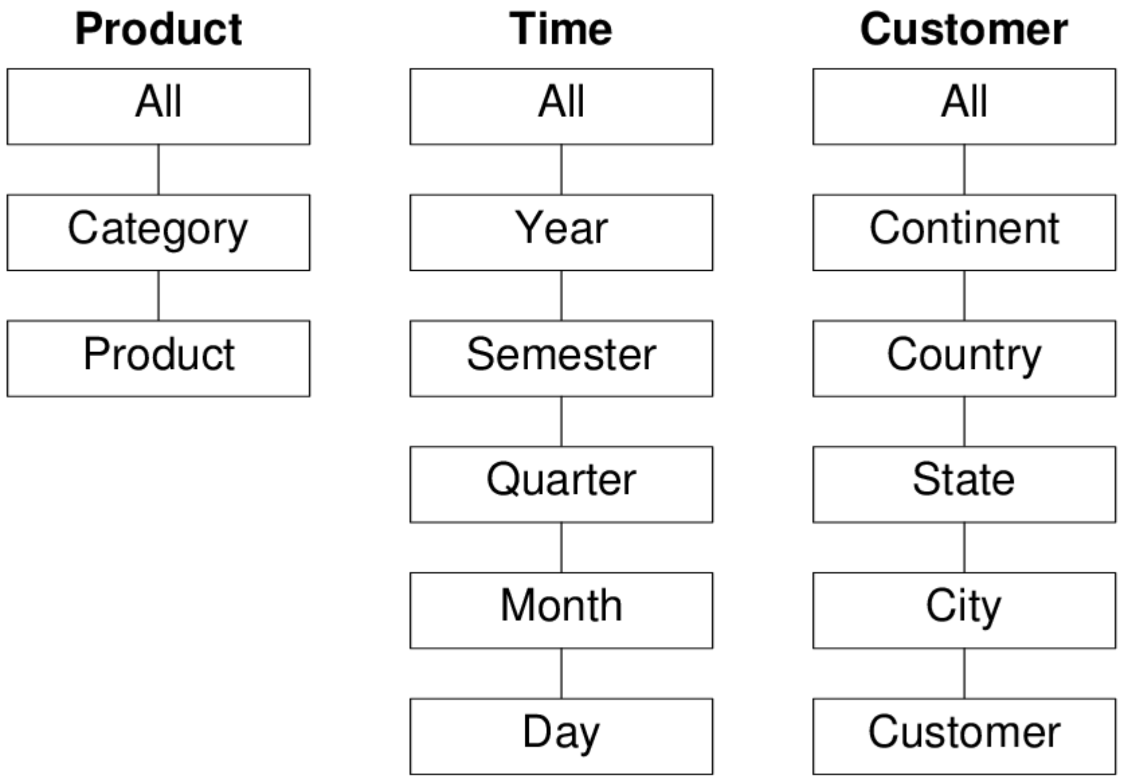
- Data can be viewed as a **cube**



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

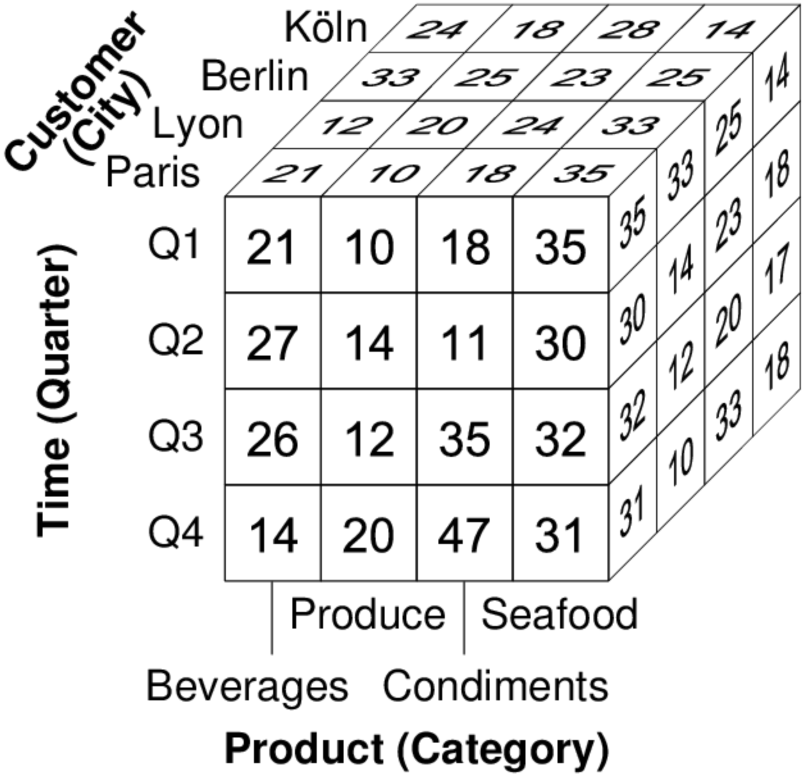
Hierarchies

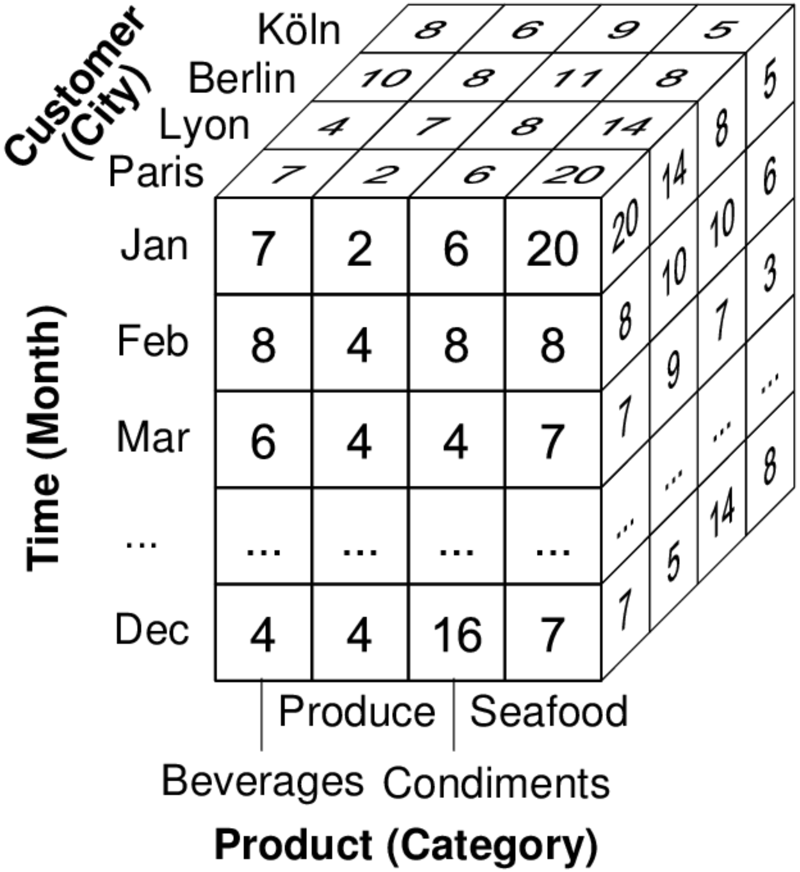
- Each **dimension** has multiple **levels** (**hierarchy**)



OLAP operations

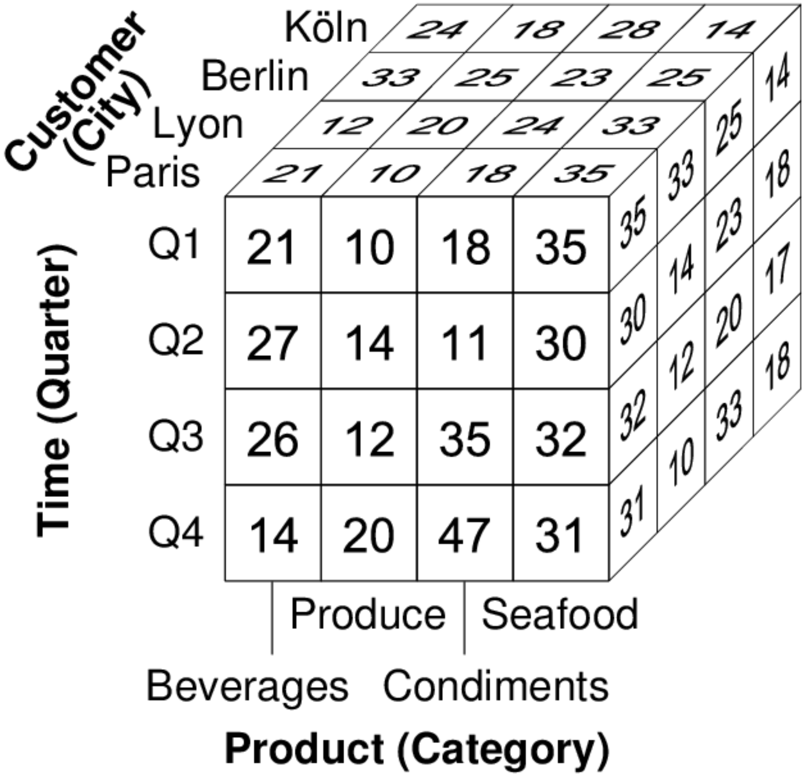
- **Drill-down** to month

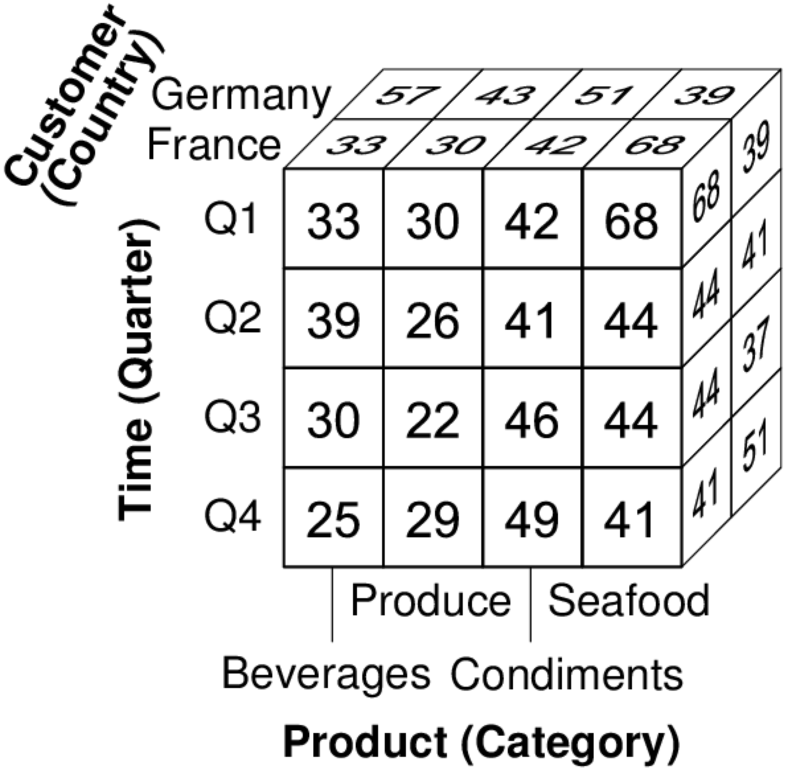




OLAP operations

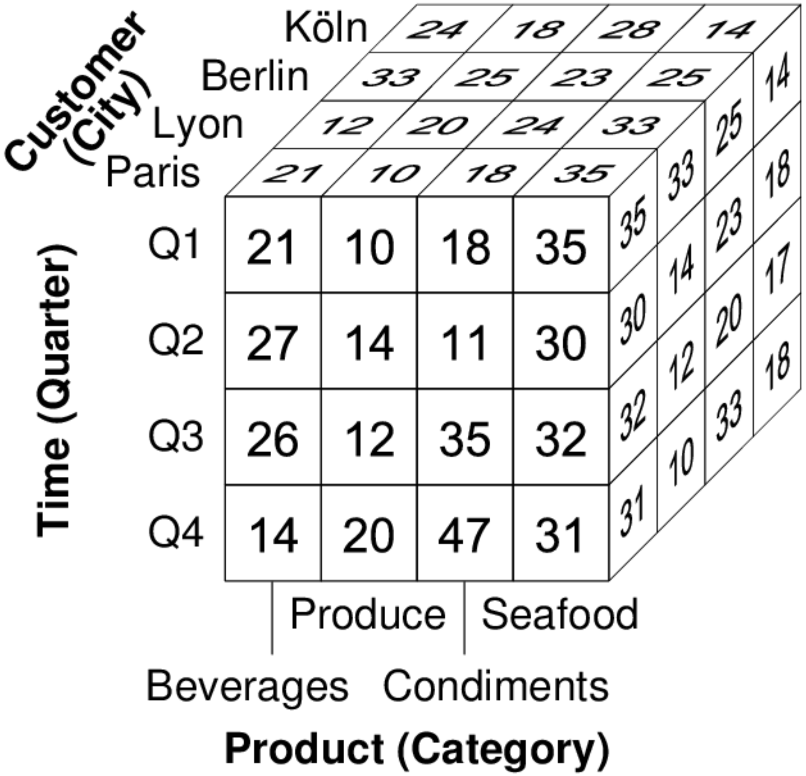
- **Roll-up** to country

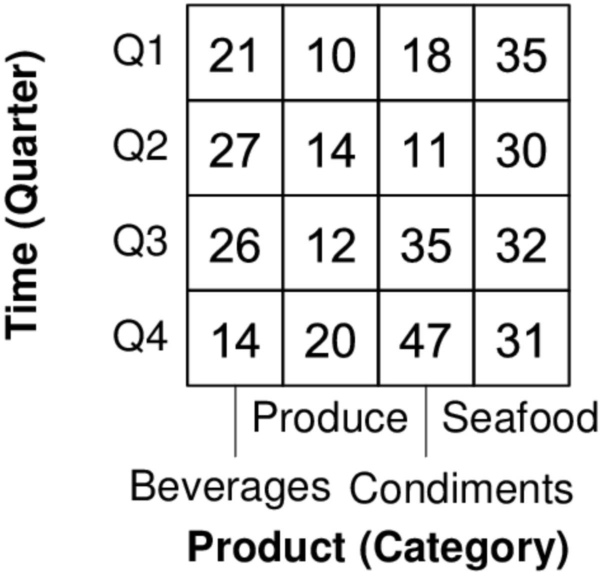




OLAP operations

- **Slice** on city (Paris)



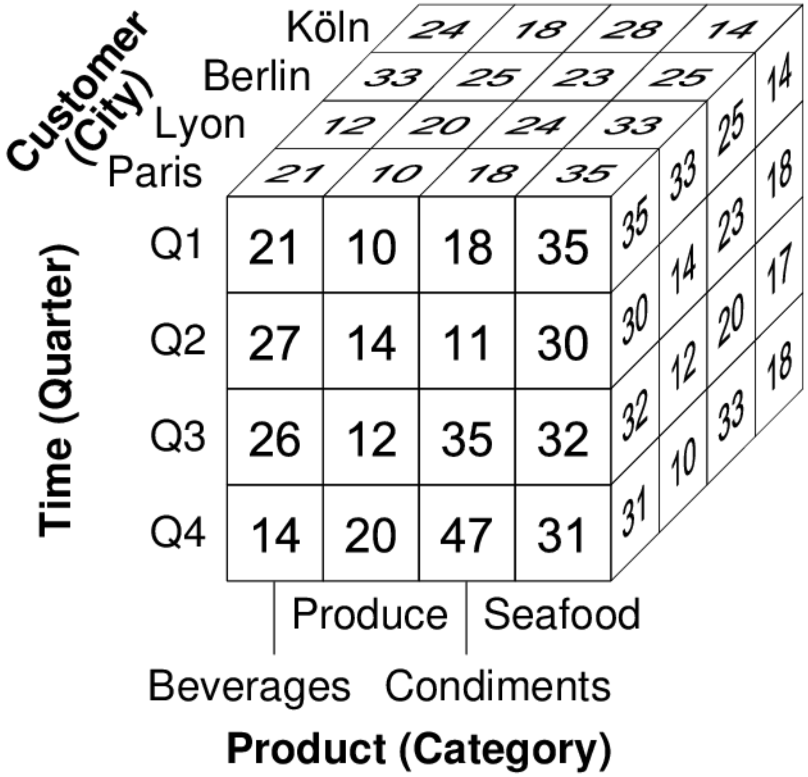


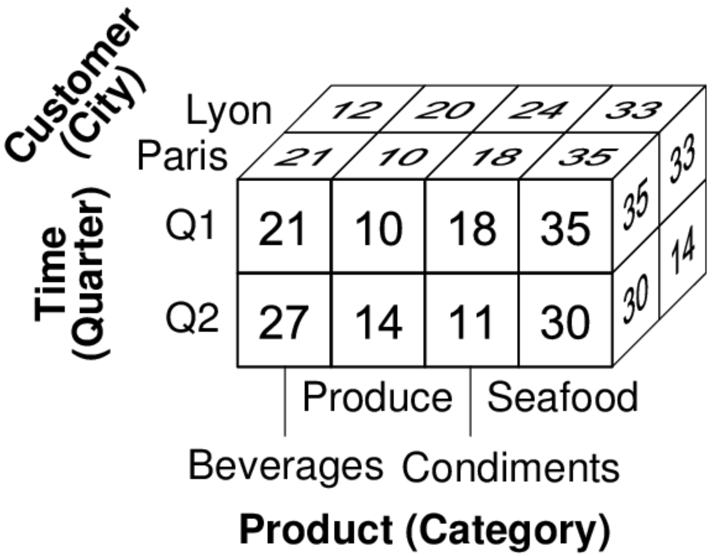
Time (Quarter)	Beverages	Produce	Condiments	Seafood
Q1	21	10	18	35
Q2	27	14	11	30
Q3	26	12	35	32
Q4	14	20	47	31

Product (Category)

OLAP operations

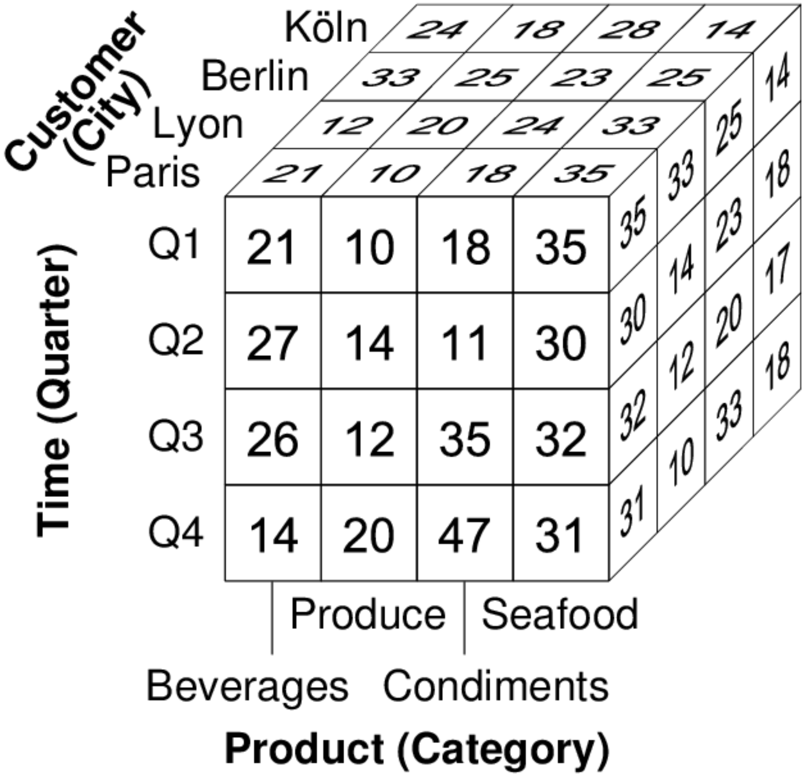
- **Dice** on city {Paris, Lyon} and quarter {Q1, Q2}



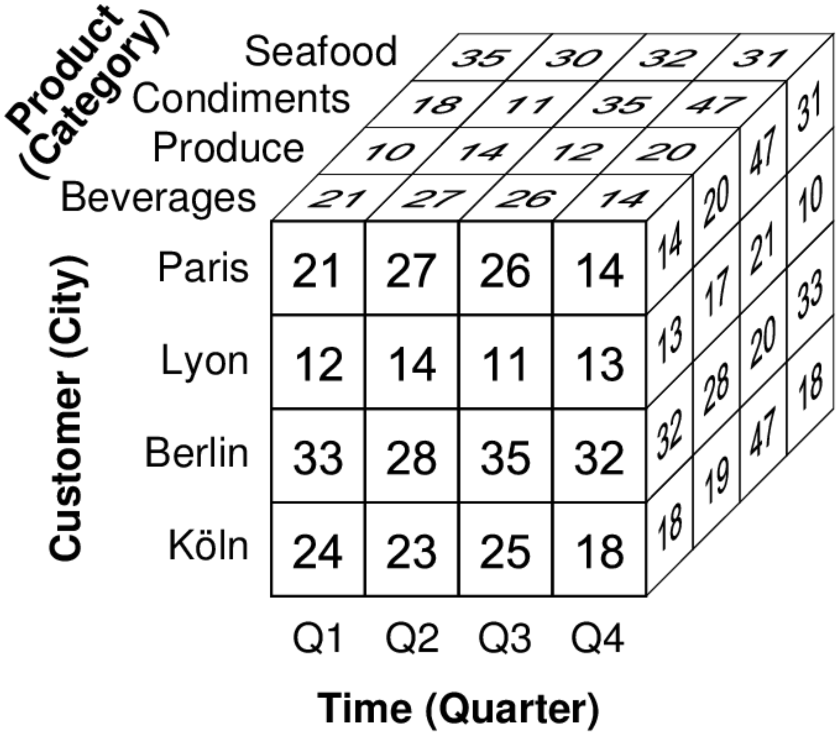


OLAP operations

- **Pivot** (rotate) dimensions



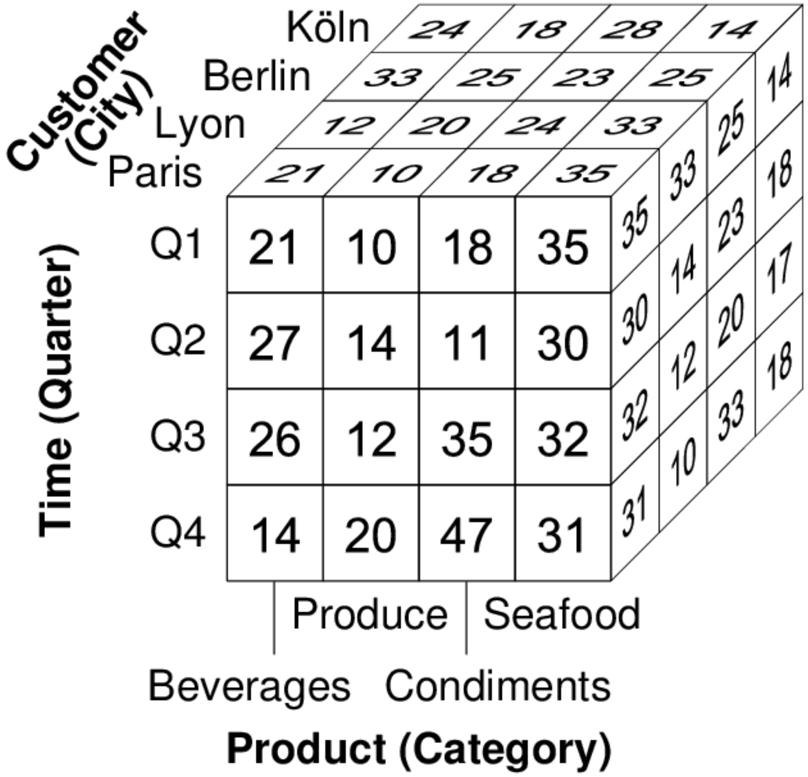
	Beverages	Produce	Condiments	Seafood
Q1	21	10	18	35
Q2	27	14	11	30
Q3	26	12	35	32
Q4	14	20	47	31

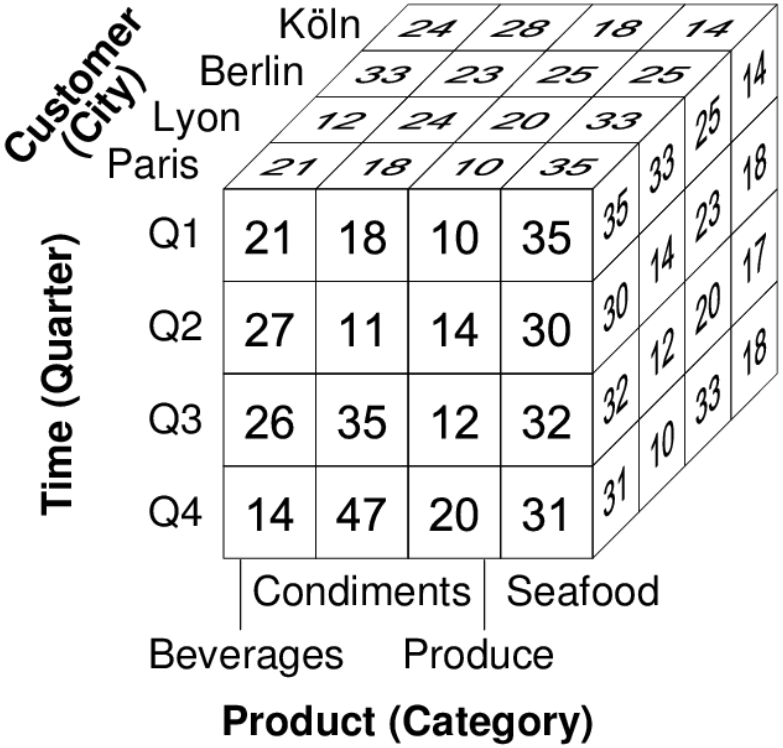


	Q1	Q2	Q3	Q4
Paris	21	27	26	14
Lyon	12	14	11	13
Berlin	33	28	35	32
Köln	24	23	25	18

Other OLAP operations

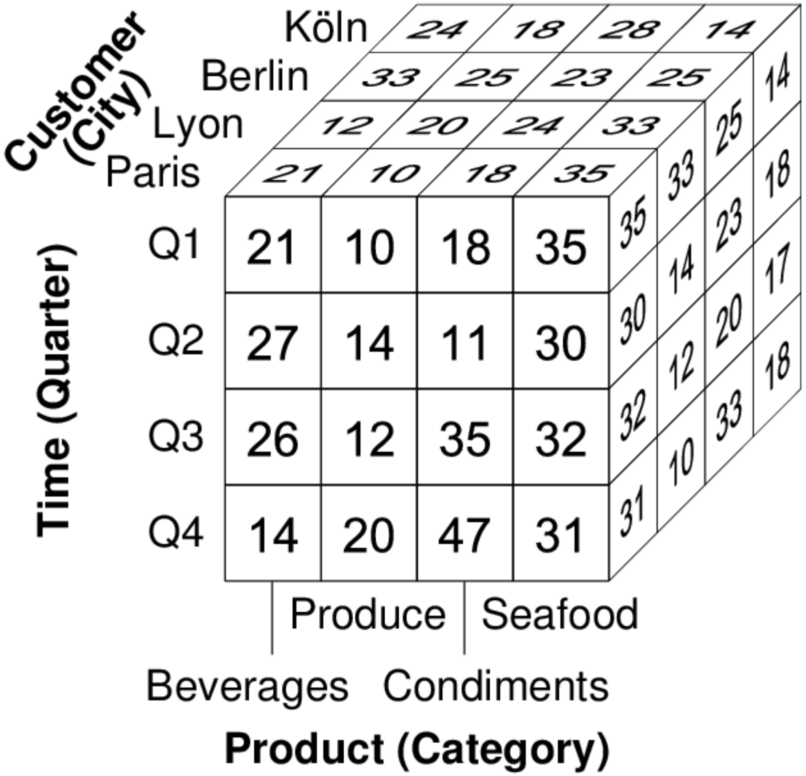
- Sort by product category





Other OLAP operations

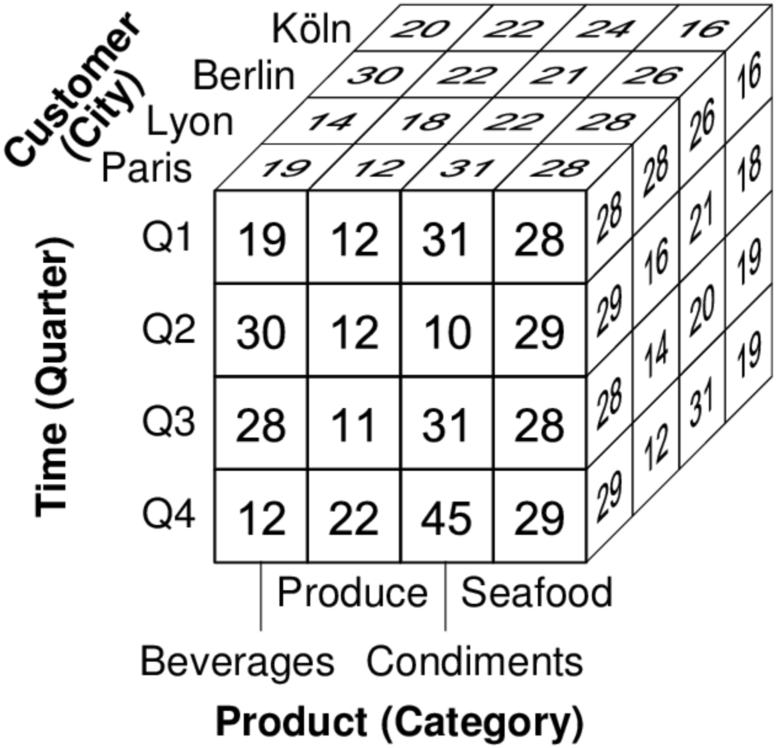
- Cube for 2012 vs. cube for 2011 (previous year)



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

	Beverages	Produce	Condiments	Seafood
Köln	24	18	28	14
Berlin	33	25	23	25
Lyon	12	20	24	33
Paris	21	10	18	35

	Paris	Lyon	Berlin	Köln
Q1	35	33	25	14
Q2	30	14	23	18
Q3	32	12	20	17
Q4	31	10	33	18



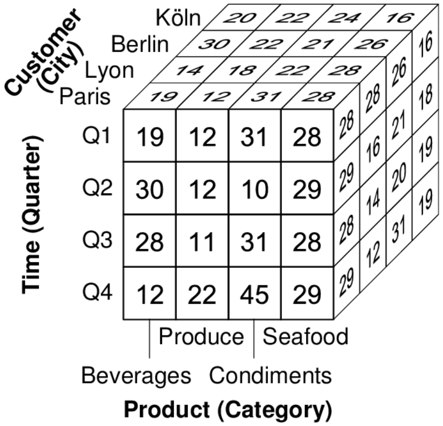
	Beverages	Produce	Condiments	Seafood
Q1	19	12	31	28
Q2	30	12	10	29
Q3	28	11	31	28
Q4	12	22	45	29

	Beverages	Produce	Condiments	Seafood
Köln	20	22	24	16
Berlin	30	22	21	26
Lyon	14	18	22	28
Paris	19	12	31	28

	Paris	Lyon	Berlin	Köln
Q1	28	28	26	16
Q2	29	16	21	18
Q3	28	14	20	19
Q4	29	12	31	19

Other OLAP operations

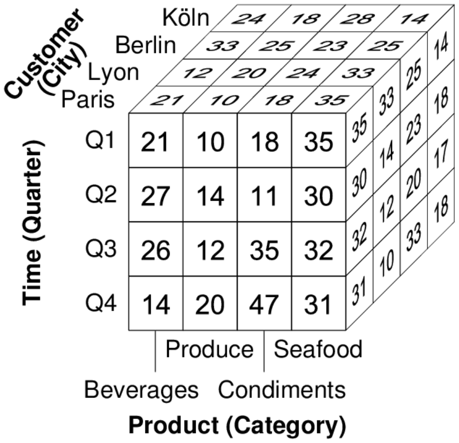
- Drill-across 2011 and 2012



Paris	Beverages	Produce	Condiments	Seafood
Q1	19	12	31	28
Q2	30	12	10	29
Q3	28	11	31	28
Q4	12	22	45	29

Q1	Beverages	Produce	Condiments	Seafood
Köln	20	22	24	16
Berlin	30	22	21	26
Lyon	14	18	22	28
Paris	19	12	31	28

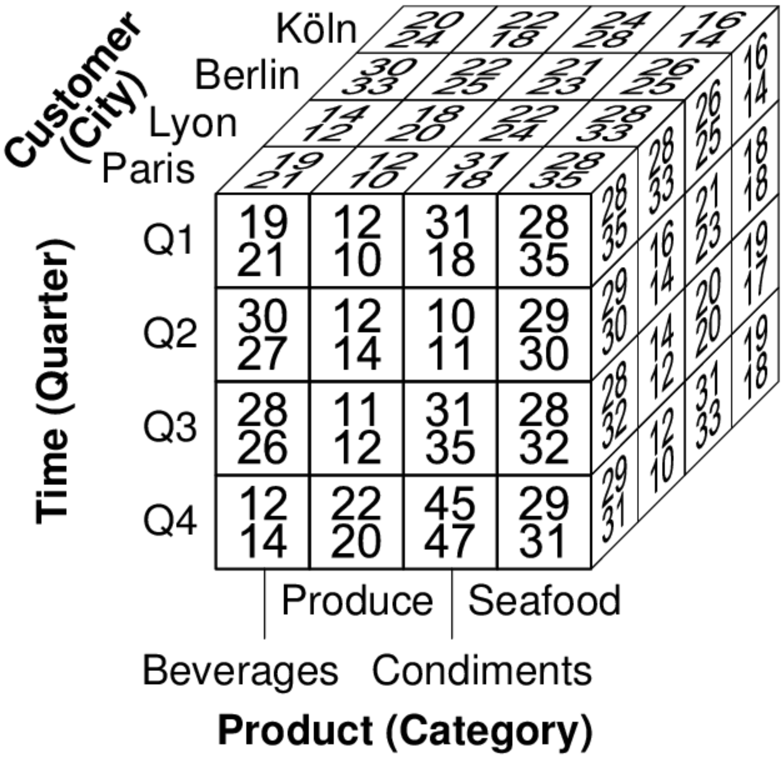
Seafood	Paris	Lyon	Berlin	Köln
Q1	28	28	26	16
Q2	29	16	21	18
Q3	28	14	20	19
Q4	29	12	31	19



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

Q1	Beverages	Produce	Condiments	Seafood
Köln	24	18	28	14
Berlin	33	25	23	25
Lyon	12	20	24	33
Paris	21	10	18	35

Seafood	Paris	Lyon	Berlin	Köln
Q1	35	33	25	14
Q2	30	14	23	18
Q3	32	12	20	17
Q4	31	10	33	18



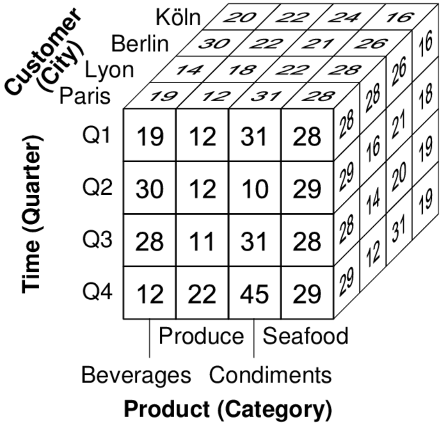
Paris	Beverages	Produce	Condiments	Seafood
Q1	19 21	12 10	31 18	28 35
Q2	30 27	12 14	10 11	29 30
Q3	28 26	11 12	31 35	28 32
Q4	12 14	22 20	45 47	29 31

Q1	Beverages	Produce	Condiments	Seafood
Köln	20 24	22 18	24 28	16 14
Berlin	30 33	22 25	21 23	26 25
Lyon	14 12	18 20	22 24	28 33
Paris	19 21	12 10	31 18	28 35

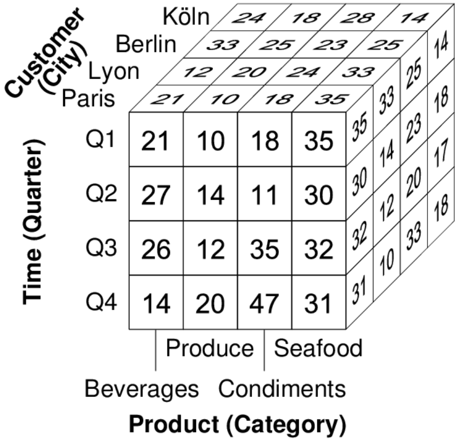
Seafood	Paris	Lyon	Berlin	Köln
Q1	28 35	28 33	26 25	16 14
Q2	29 30	16 14	21 23	18 18
Q3	28 32	14 12	20 20	19 17
Q4	29 31	12 10	31 33	19 18

Other OLAP operations

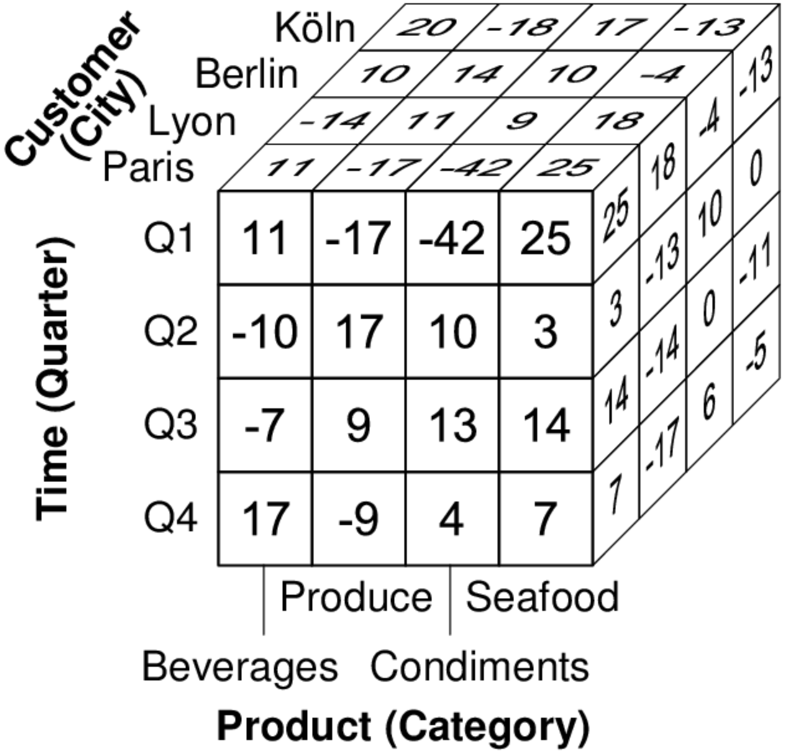
- Percentage change between 2011 and 2012



Time (Quarter)	Beverages	Produce	Condiments	Seafood
Q1	19	12	31	28
Q2	30	12	10	29
Q3	28	11	31	28
Q4	12	22	45	29



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



Time (Quarter)	Beverages	Produce	Condiments	Seafood
Q1	11	-17	-42	25
Q2	-10	17	10	3
Q3	-7	9	13	14
Q4	17	-9	4	7

Data warehousing

- What is a data warehouse?
  - a data warehouse is a relational database that stores historical data in a convenient schema for multidimensional analysis using OLAP operations

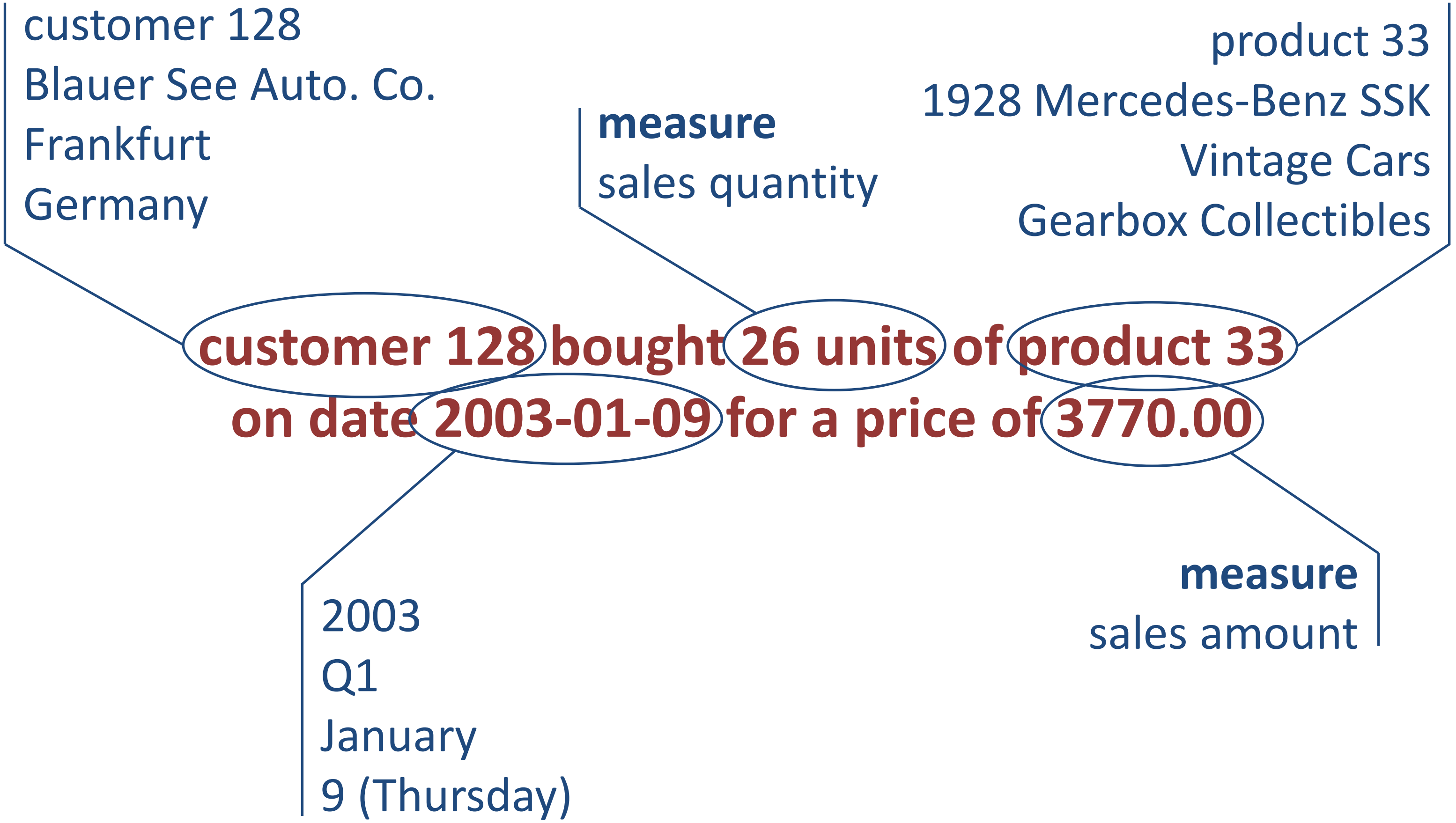
Data warehousing

- A data warehouse stores **facts**

customer 128 bought 26 units of product 33 on date 2003-01-09 for a price of 3770.00

Data warehousing

- Facts have associated **measures**



customer 128 bought 26 units of product 33 on date 2003-01-09 for a price of 3770.00

customer 128
Blauer See Auto. Co.
Frankfurt
Germany

measure
sales quantity

product 33
1928 Mercedes-Benz SSK
Vintage Cars
Gearbox Collectibles

2003
Q1
January
9 (Thursday)

measure
sales amount

Data warehousing

- **Facts** define the possible analysis dimensions
  - customer c bought product p (2D)
  - customer c bought product p on date d (3D)
  - customer c bought product p on date d in store s (4D)
  - customer c bought product p on date d in store s using payment method m (5D)
    - m may be 'cash', 'credit card', etc.
  - customer c bought product p on date d in store s using payment method m through sales representative e (6D)
    - e is an employee number
  - etc.

Data warehousing

- **Measures** define the quantity being analyzed
  - sales
    - quantity sold
    - sales amount
  - production
    - quantity produced
    - production cost
  - logistics
    - distance traveled
    - transported weight
  - etc.

Data warehousing

- **Facts** can be **grouped** by dimensions
  - examples
    - by customer (1D)
    - by product (1D)
    - by time (1D)
    - by customer and product (2D)
    - by customer and time (2D)
    - by product and time (2D)
    - by customer, product and time (3D)

Data warehousing

- As **facts** are **grouped**, their **measures** are **aggregated**
  - examples
    - sales amount by customer (1D)
    - sales amount by product (1D)
    - sales amount by time (1D)
    - sales amount by customer and product (2D)
    - sales amount by customer and time (2D)
    - sales amount by product and time (2D)
    - sales amount by customer, product and time (3D)

Data warehousing

- **Measures** are **aggregated** at a certain **level** of detail
  - examples
    - sales amount by customer country
    - sales amount by customer city
    - sales amount by customer country and product line
    - sales amount by customer city and product vendor
    - sales amount by customer country, product line and year
    - sales amount by customer city, product vendor and month

Data warehousing

- **Levels** are organized into **hierarchies**
  - examples
    - country, state, city
    - year, month, day

- A **dimension** can have one or more **hierarchies**
  - example: time dimension
    - year, month, day (for calendar year, from Jan to Dec)
    - year, semester, period (for school year, from Sep to Aug)