


Chapter 2

ETL Process and OLAP

Based on CO2

A series of horizontal lines in teal and white, with the teal line being the most prominent and the white lines being thinner and stacked below it.

CSC603.2: Students should be able to use star schema for designing a data warehouse and execute appropriate OLAP queries.

By, Safa Hamdare

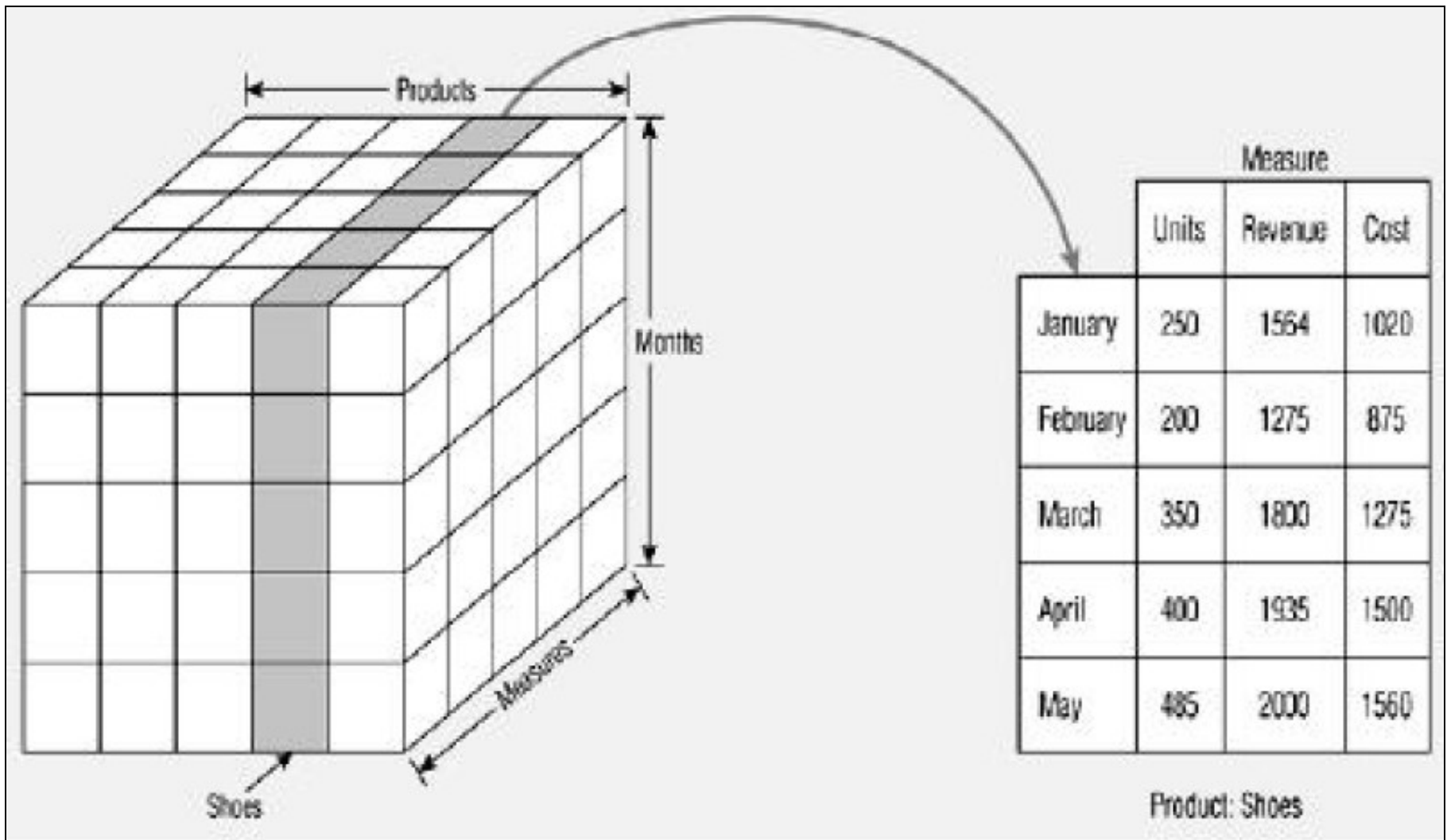
Outline:

- **Online Analytical Processing (OLAP):**
 - OLAP Definition
 - Need for Online Analytical Processing
 - OLTP V/s OLAP
 - OLAP and Multidimensional Analysis
 - Hyper cubes
 - OLAP operations: Drill down, Roll up, Slice, Dice and rotation
 - OLAP Models: MOLAP, ROLAP, HOLAP, DOLAP

What is OLAP?

- The term OLAP (“online analytical processing”) was coined in a white paper written for Arbor Software Corp. in **1993**
 - OLAP (online analytical processing) is computer processing that enables a user to easily and selectively extract and view data from different points of view.
 - OLAP allows users to analyze database information from multiple database systems at one time.
 - OLAP data is stored in multidimensional databases.

OLAP Example:



OLAP:

- Some popular OLAP server software programs include:
 - **Oracle Express Server**
 - **Hyperion Solutions Essbase**
- OLAP processing is often used for data mining.
- OLAP products are typically designed for multiple-user environments, with the cost of the software based on the number of users.

Data aggregation in SQL

- Data aggregation in many different ways
- The number of possible groupings quickly becomes large
 - The user has to consider all groupings
 - Analytical processing problem

Drawbacks of SQL Group by- Need for OLAP

- Formulation so many similar but distinct queries is tedious
- Executing the queries is expensive
- Make life easier,
 - more efficient computation
- Single query
 - GROUPING SETS, ROLLUP, CUBE options
 - Added to SQL standard 1999

Compare OLTP and OLAP System?
5 marks, 10 marks

- Asked in MU Exam (MAY 2010, May 2011, may 2012, Dec 2016)

Data Warehouse vs. Operational DBMS

-(May 2010, May 2011, May 2012, Dec 2016) 5, 10 marks

- **OLTP (on-line transaction processing)**
 - Major task of traditional relational DBMS
 - Day-to-day operations: purchasing, inventory, banking, manufacturing, payroll, registration, accounting, etc.
- **OLAP (on-line analytical processing)**
 - Major task of data warehouse system
 - Data analysis and decision making
- **Distinct features (OLTP vs. OLAP):**
 - **User and system orientation:** customer **vs.** market
 - **Data contents:** current, detailed **vs.** historical, consolidated
 - **Database design:** ER + application **vs.** star + subject
 - **View:** current, local **vs.** evolutionary, integrated
 - **Access patterns:** update **vs.** read-only but complex queries

Compare OLTP (Operational System) and OLAP (Data Warehouse) -(May 2010, May 2011, May 2012, Dec 16) 5 marks, 10 marks

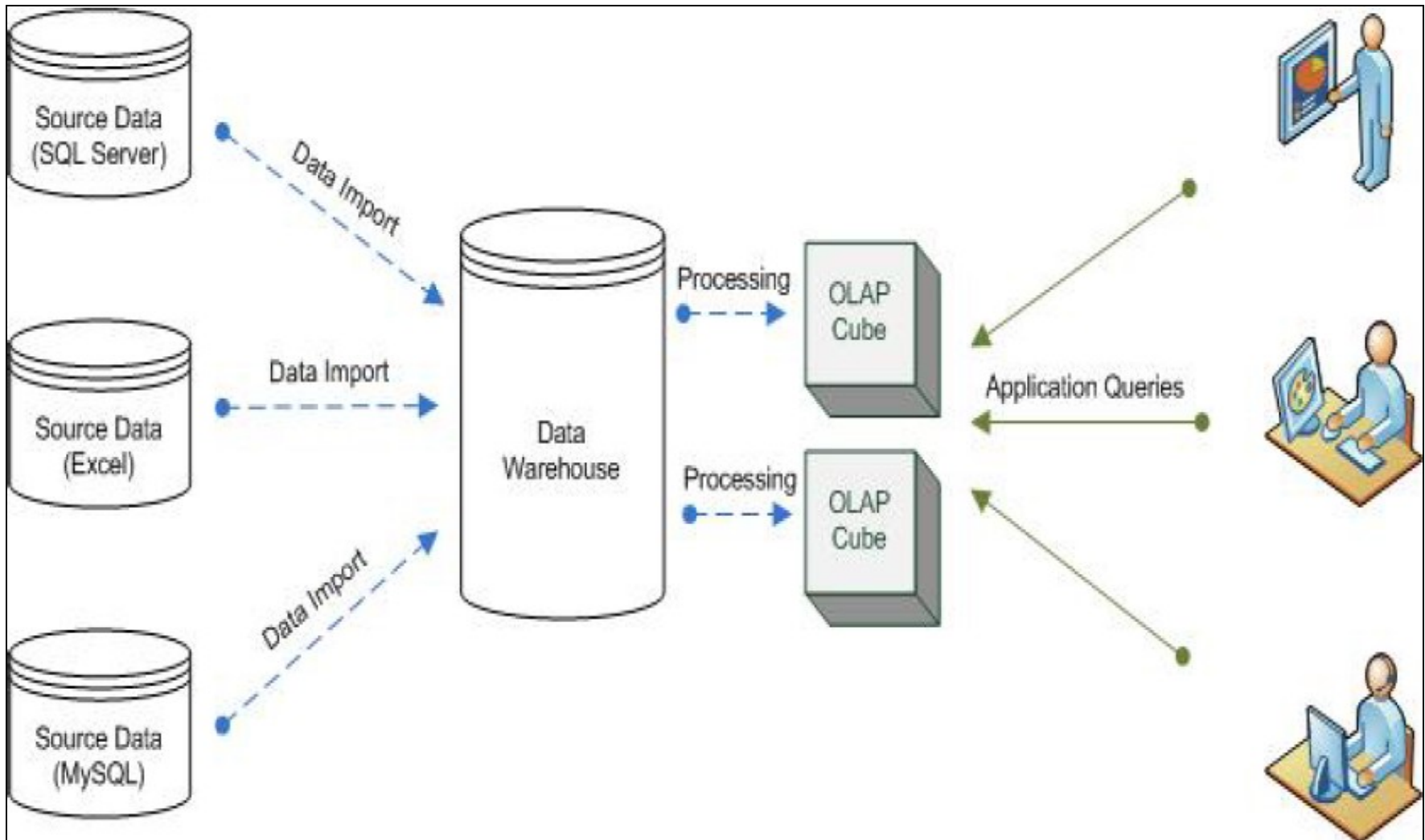
	OLTP	OLAP
users	clerk, IT professional	knowledge worker
function	day to day operations	decision support
DB design	application-oriented	subject-oriented
data	current, up-to-date detailed, flat relational isolated	historical, summarized, multidimensional integrated, consolidated
usage	repetitive	ad-hoc
access	read/write index/hash on prim. key	lots of scans
unit of work	short, simple transaction	complex query
# records accessed	tens	millions
#users	thousands	hundreds
DB size	100MB-GB	100GB-TB
metric	transaction throughput	query throughput, response



OLAP and Multidimensional Analysis

OLAP CUBE

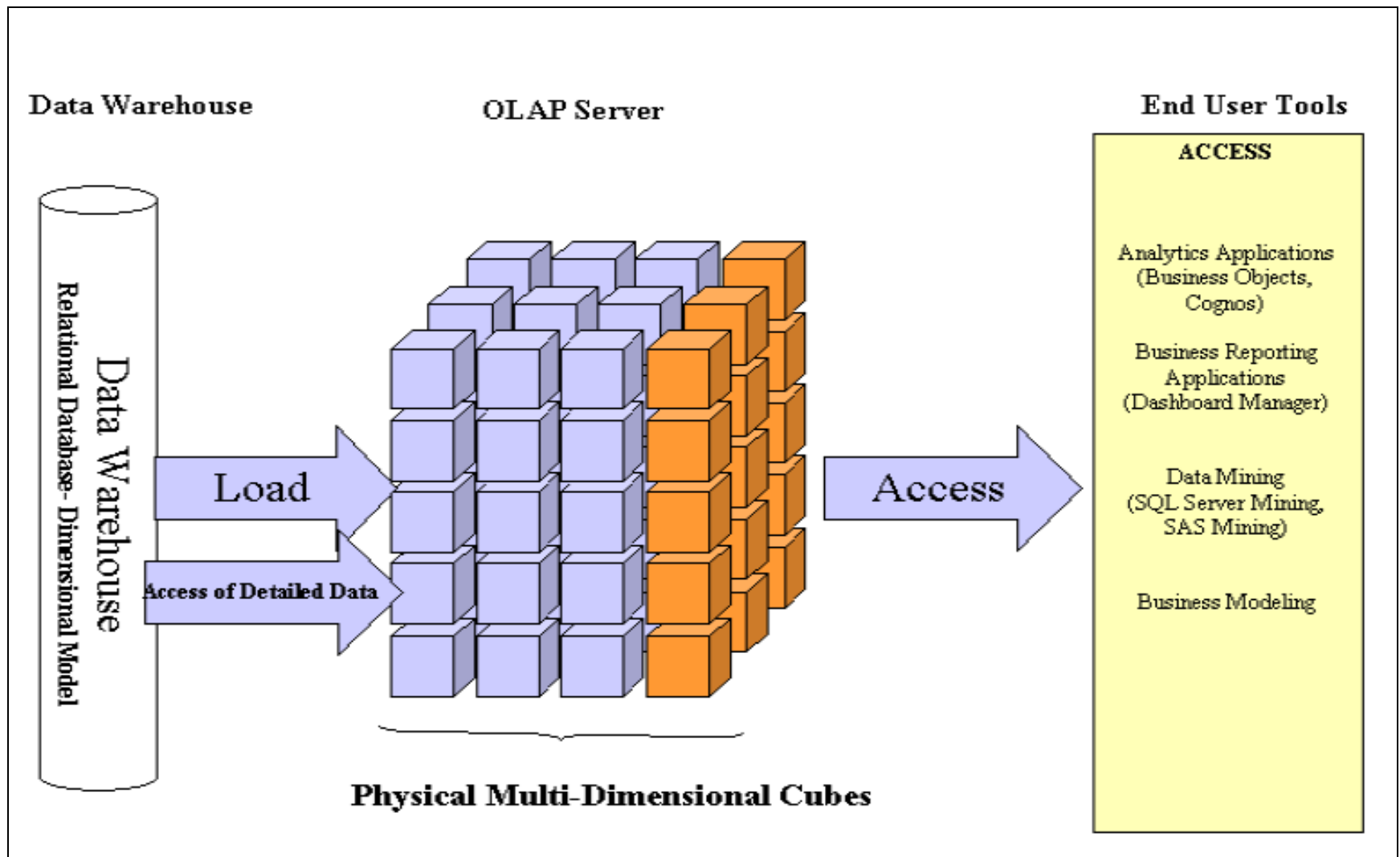
A multidimensional cube can combine data from disparate data sources and store the information in a fashion that is logical for business users.



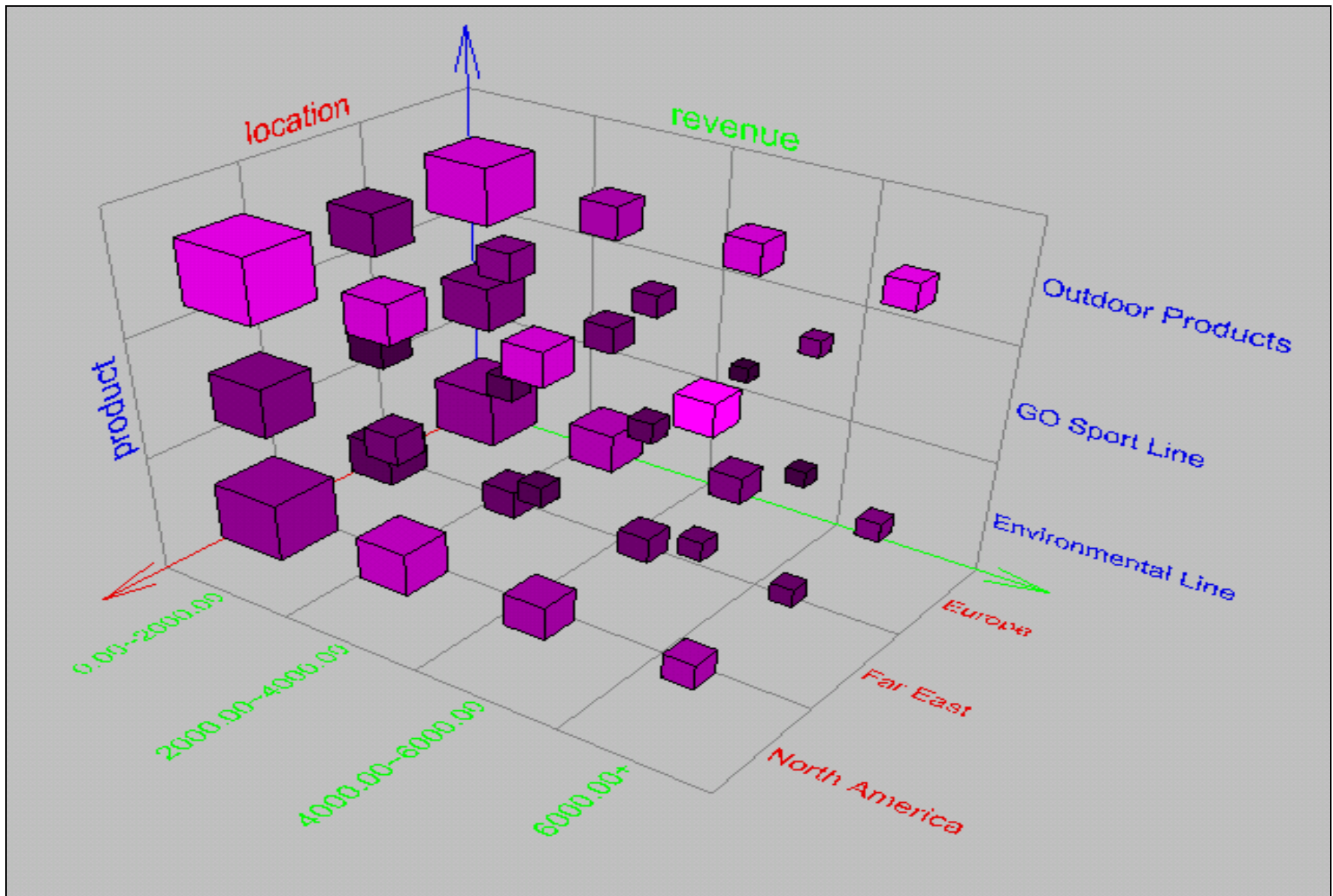
OLAP Cube:

- An OLAP Cube is a data **structure that allows fast analysis of data.**
- The arrangement of data into cubes overcomes a limitation of relational databases.
- It consists of numeric facts called **measures** which are categorized by dimensions.
- The OLAP cube consists of numeric facts called **measures** which are categorized by **dimensions.**

OLAP Cube



Browsing a Data Cube





OLAP Operations in Multidimensional Data Model

Typical OLAP Operations

1. **Slice and dice:** project and select
2. **Roll up (drill-up):** summarize data
-by climbing up hierarchy or by dimension reduction
3. **Drill down (roll down):** reverse of roll-up
-from higher level summary to lower level summary or detailed data, or introducing new dimensions
4. **Pivot (rotate):**
- reorient the cube, visualization, 3D to series of 2D planes
5. Other operations
 - **drill across:** *involving (across) more than one fact table*
 - **drill through:** *through the bottom level of the cube to its back-end relational tables (using SQL)*

OLAP Operations in Multidimensional Data Model

- The user-initiated process of navigating by calling for page displays interactively, through the specification of slices via **rotations and drill down/up** is sometimes called "**slice and dice**".
1. **Slice:** A slice is a subset of a multi-dimensional array corresponding to a single value for one or more members of the dimensions not in the subset.
 2. **Dice:** The dice operation is a slice on more than two dimensions of a data cube (or more than two consecutive slices).
 3. **Drill Down/Up:** Drilling down or up is a specific analytical technique whereby the user navigates among levels of data ranging from the most summarized (up) to the most detailed (down).

OLAP Operations in Multidimensional Data Model

4. **Roll-up:** A roll-up involves computing all of the data relationships for one or more dimensions. To do this, a computational relationship or formula might be defined.
 5. **Pivot:** To change the dimensional orientation of a report or page display.
- The output of an OLAP query is typically displayed in a matrix (or pivot) format. The dimensions form the row and column of the matrix; the measures, the values.

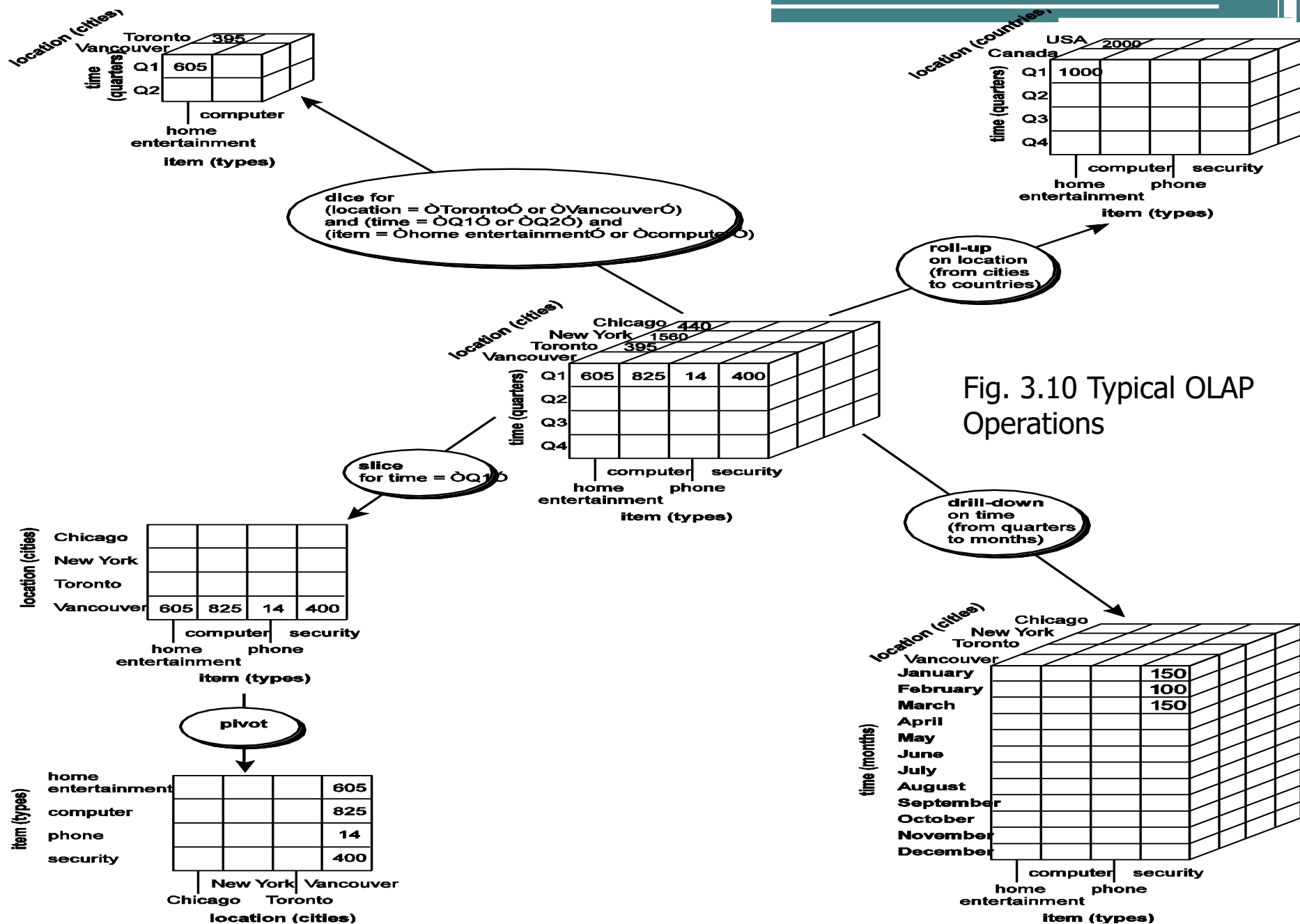
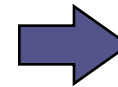


Fig. 3.10 Typical OLAP Operations

Aggregates

- Add up amounts for day 1
 - In SQL: `SELECT sum(amt) FROM SALE WHERE date = 1`

sale	prodId	storeId	date	amt
	p1	s1	1	12
	p2	s1	1	11
	p1	s3	1	50
	p2	s2	1	8
	p1	s1	2	44
	p1	s2	2	4

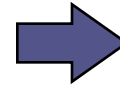


81

Aggregates

- Add up amounts by day
- In SQL: `SELECT date, sum(amt) FROM SALE GROUP BY date`

sale	prodlid	storeld	date	amt
	p1	s1	1	12
	p2	s1	1	11
	p1	s3	1	50
	p2	s2	1	8
	p1	s1	2	44
	p1	s2	2	4

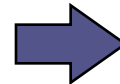


ans	date	sum
	1	81
	2	48

Another Example

- Add up amounts by day, product
- In SQL: `SELECT date, sum(amt) FROM SALE GROUP BY date, prodId`

sale	prodId	storeId	date	amt
	p1	s1	1	12
	p2	s1	1	11
	p1	s3	1	50
	p2	s2	1	8
	p1	s1	2	44
	p1	s2	2	4

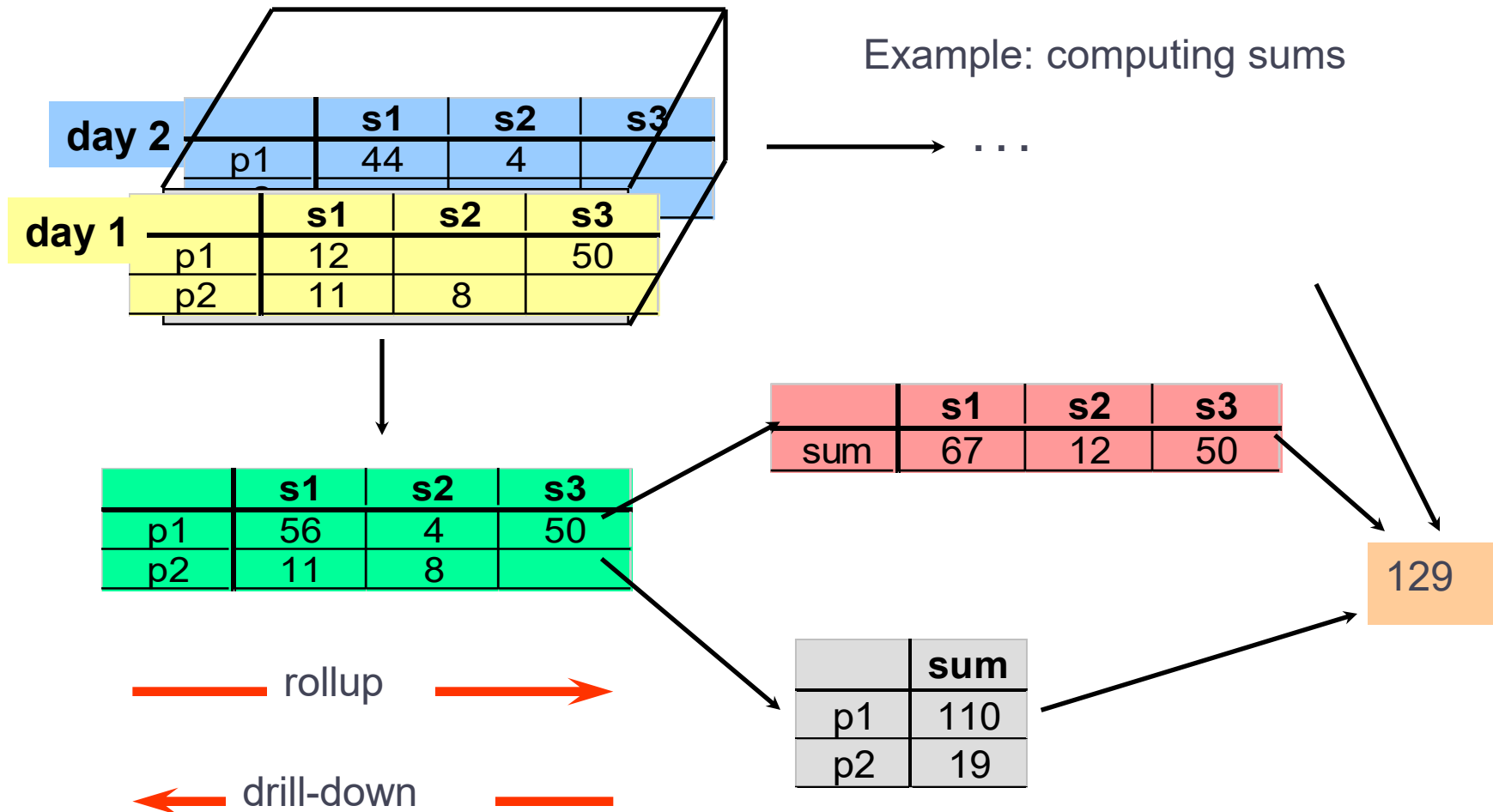


sale	prodId	date	amt
	p1	1	62
	p2	1	19
	p1	2	48

rollup →

← drill-down

Cube Aggregation: Roll-up



Slicing

day 2		s1	s2	s3
p1		44	4	


day 1		s1	s2	s3
p1		12		50
p2		11	8	

TIME = day 1

	s1	s2	s3
p1	12		50
p2	11	8	

Slicing & Pivoting

Sales (\$ millions)				
	Products	Time		
		d1	d2	
Store s1	Electronics	\$5.2		
	Toys	\$1.9		
	Clothing	\$2.3		
	Cosmetics	\$1.1		
Store s2	Electronics	\$8.9		
	Toys	\$0.75		
	Clothing	\$4.6		
	Cosmetics	\$1.5		

		Sales (\$ millions)			
		Products	d1		
			Store s1	Store s2	
Store s1	Electronics	\$5.2	\$8.9		
	Toys	\$1.9	\$0.75		
	Clothing	\$2.3	\$4.6		
	Cosmetics	\$1.1	\$1.5		
Store s2	Electronics				
	Toys				
	Clothing				

(Dec 2017, 10 Marks)

A) Consider a data warehouse for a hospital where there are three dimension
a) Doctor b) Patient c) Time
Consider two measures i) Count ii) Charge where charge is the fee that the doctor charges a patient for a visit. For the above example create a cube and illustrate the following OLAP operations.
1) Rollup 2) Drill down 3) Slice 4) Dice 5) Pivot.

(May 2016, 10 Marks)

We would like to view sales data of a company with respect to three dimensions namely Location, Item and Time. Represent the sales data in the form of a 3-D data cube for the above and Perform Roll up, Drill down, Slice and Dice OLAP operations on the above data cube and Illustrate.

(May 2017, May 2018 10 Marks)

The college wants to record the Marks for the courses completed by students using the dimensions: i) Course, ii) Student, iii) Time & a measure Aggregate marks. Create a Cube and describe following OLAP operations:
(i) Slice (ii) Dice (iii) Roll up (iv) Drill down (v) Pivot

May 2019

Select avg (grade) from fact where Course=CS group by (course);

- a) Suppose that a data warehouse for *DB-University* consists of the four dimensions *student*, *course*, *semester*, and *instructor*, and two measures count and *avg-grade*. At the lowest conceptual level (e.g., for a given student, course, semester, and instructor combination), the *avg-grade* measure stores the actual course grade of the student. At higher conceptual levels, *avg-grade* stores the average grade for the given combination. [10]
- Draw a *snowflake schema* diagram for the data warehouse.
 - Starting with the base cuboid [*student*, *course*, *semester*, *instructor*], what specific *OLAP operations* (e.g., roll-up from *semester* to *year*) should you perform in order to list the average grade of CS courses for each *DB-University* student.

Dec 2019

- Q2 a) Suppose that a data warehouse consists of the three dimensions time, doctor and patient, and the two measures count and charge, where charge is the fee that a doctor charges a patient for a visit. [10]
- Draw a star schema diagram for the above data warehouse.
 - Starting with the base cuboid [day, doctor, patient], what specific OLAP operations should be performed in order to list the total fee collected by each doctor in 2010?
 - To obtain the same list, write an SQL query assuming the data are stored in a relational database with the schema fee (day, month, year, doctor, hospital, patient, count, charge).

(May 2017, May 2018 10 Marks)- College DM for OLAP Queries

Student	Course	Time	Marks
S001	DWM	T1	80
S002	DWM	T1	65
S003	DWM	T1	45
S001	PDS	T1	40
S002	PDS	T1	60
S003	PDS	T1	55
S001	AI	T1	60
S002	AI	T1	50
S003	AI	T1	45
S001	DWM	T2	40
S002	DWM	T2	60
S003	DWM	T2	70
S001	PDS	T2	80
S002	PDS	T2	90
S003	PDS	T2	35
S001	AI	T2	45
S002	AI	T2	55
S003	AI	T2	65

Student, course, semester and Instructor

	T2		
T1			
s001	80	40	60
s002	65	60	50
s003	45	55	45
	DWM	PDS	AI

- **Slice-** select marks from fact where student='Soo1';

for 'Soo1'

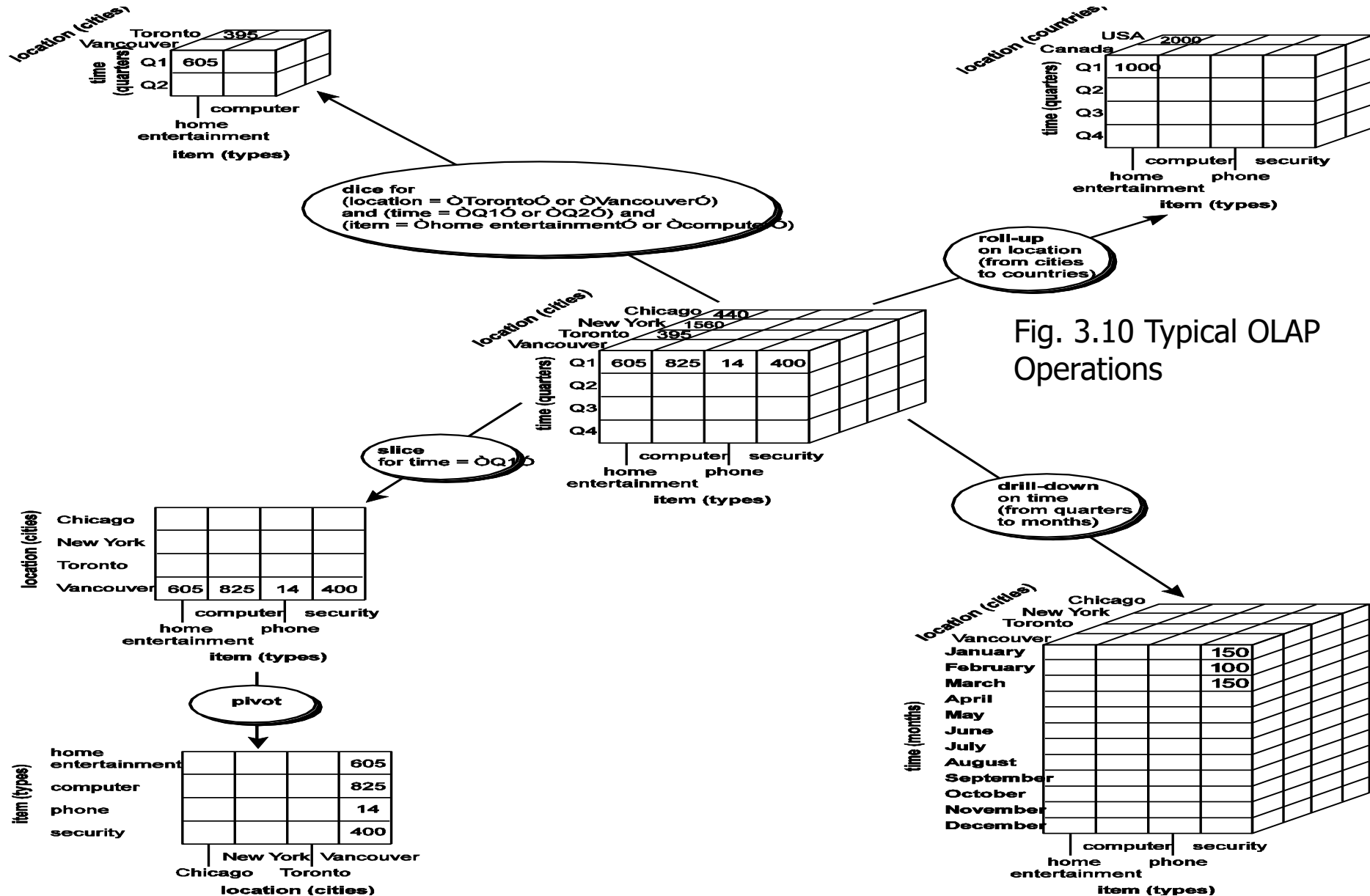
T1	80	40	60
T2	40	80	45
	DWM	PDS	AI

- **Dice-** Select marks from fact where Student='Soo1' or student='Soo2' and course='DWM' or Course='PDS';

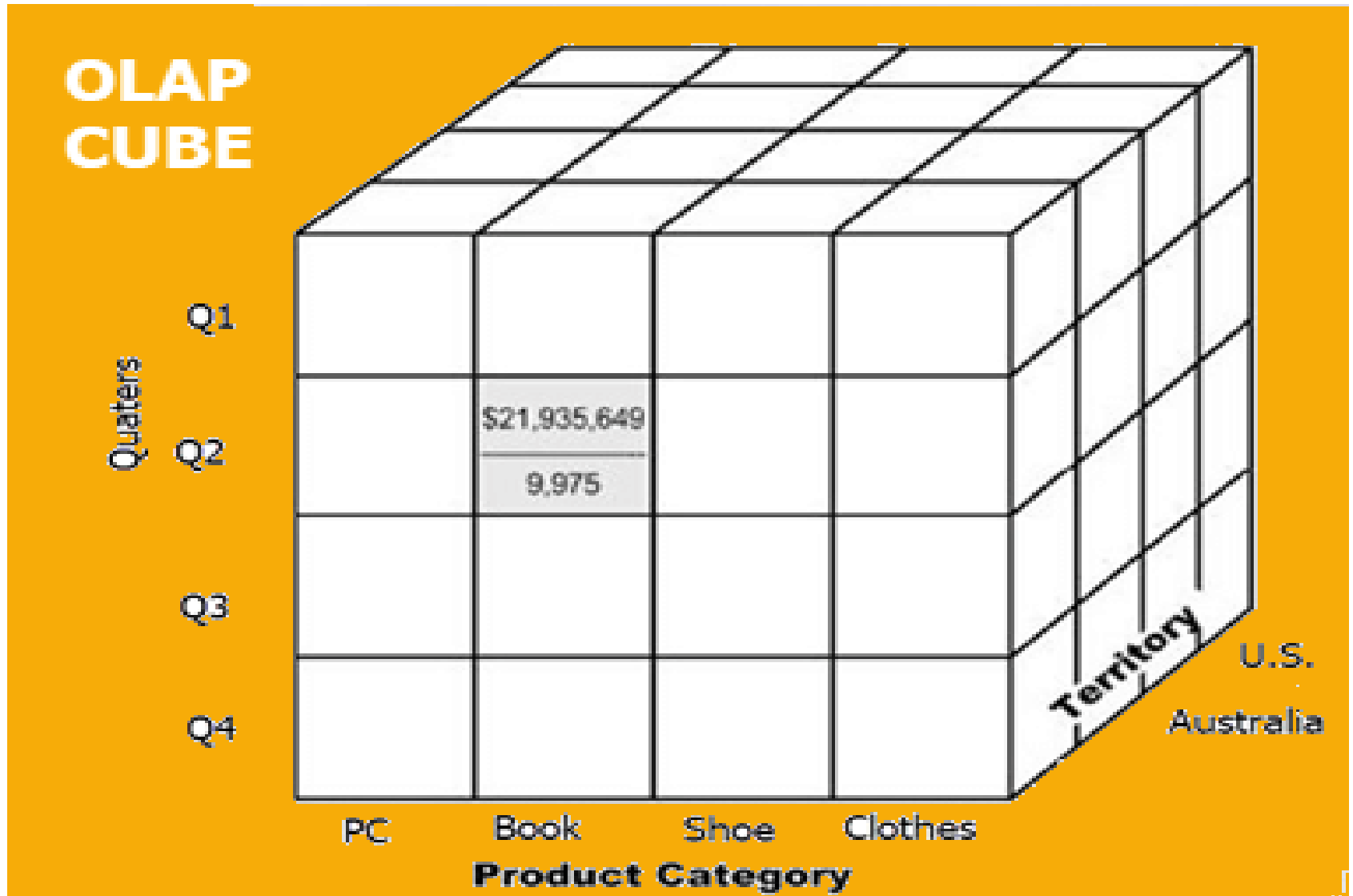
Soo1

T1	80	40
T2	40	80
	DWM	PDS

- **Rollup-** Select marks from fact group by rollup(time);

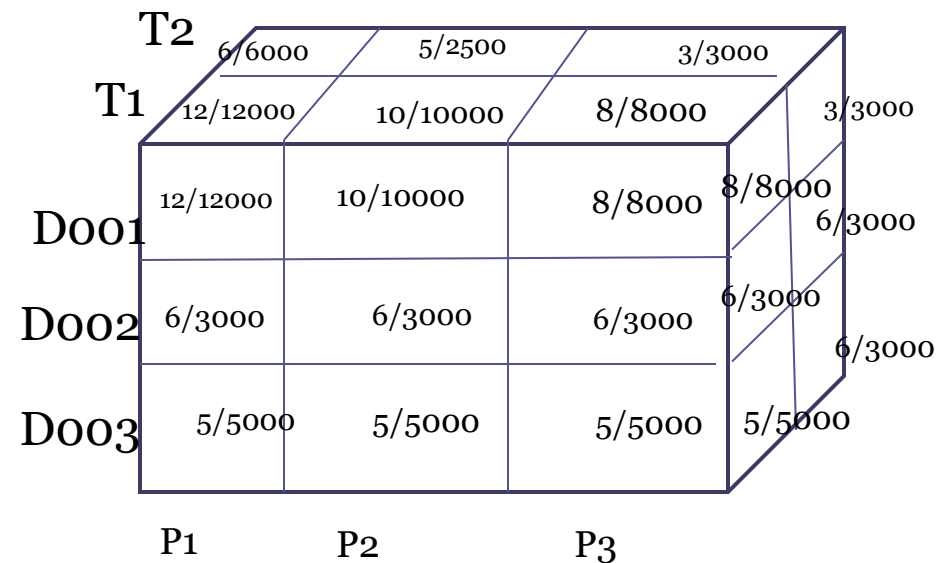


Cube with more than one measure:



(Dec 2017, Dec 2019 10 Marks)- Hospital DM for OLAP Queries

Doctor	Patient	Time	Count	Fees
D001	P001	T1	12	12000
D001	P002	T1	10	10000
D001	P003	T1	8	8000
D001	P001	T2	6	6000
D001	P002	T2	5	2500
D001	P003	T2	2	3000
D002	P001	T1	6	3000
D002	P002	T1	6	3000
D002	P003	T1	6	3000
D003	P001	T1	5	5000
D003	P002	T1	5	5000
D003	P003	T1	5	5000
D002	P003	T2	6	3000
D003	P003	T2	6	3000



OLAP Operation for Lab

Crosstab

- Cross-tabular report with (sub)totals

Media	Country		
	France	USA	Total
Internet	9,597	124,224	133,821
Direct Sales	61,202	638,201	699,403
Total	70,799	762,425	833,224

- This is space efficient for dense data only (thus, few dimensions)
- Shows data at different “granularities”

ROLLUP	Fact 1	Fact 2	Fact 3
	1 (Internet)	1 (2009)	1 (France)
	2 (Direct Sales)	2 (2010)	2 (USA)
		3 (2011)	3 (India)
		4 (2012)	4 (China)
		5 (2013)	5 (Japan)
			6 (Canada)
			7 (Indonesia)
			8 (Srilanka)
			9 (Thailand)
			10 (UK)

ROLLUP

❑ In addition to the regular aggregation results we expect from the GROUP BY clause, the **ROLLUP extension** produces group subtotals from right to left and a grand total. If "n" is the number of columns listed in the ROLLUP, there will be n+1 levels of subtotals

```
SELECT fact_1_id,
       fact_2_id,
       SUM(sales_value) AS sales_value
FROM   dimension_tab
GROUP BY ROLLUP (fact_1_id, fact_2_id)
ORDER BY fact_1_id, fact_2_id;
```

FACT_1_ID	FACT_2_ID	SALES_VALUE
1	1	4363.55
1	2	4794.76
1	3	4718.25
1	4	5387.45
1	5	5027.34
1		24291.35
2	1	5652.84
2	2	4583.02
2	3	5555.77
2	4	5936.67
2	5	4508.74
2		26237.04
		50528.39

13 rows selected.

ROLLUP with composite columns:

For ROLLUP this means stepping back through the list to determine the groupings.

```
ROLLUP (a, b, c)
(a, b, c)
(a, b)
(a)
()
```

ROLLUP Example

```
SELECT      m_desc,  t_cal_month_desc, n_iso_code,
            SUM(s_amount_sold)
FROM        spctmn
WHERE       m_desc IN ('Direct Sales', 'Internet')
AND         t_cal_month_desc IN ('2000-09', '2000-10')
AND         n_iso_code IN ('GB', 'US')
GROUP BY    ROLLUP(m_desc, t_cal_month_desc, n_iso_code);
```

- Rollup from right to left
- Computes and combines the following groupings
 - m_desc, t_cal_month_desc, n_iso_code
 - m_desc, t_cal_month_desc
 - m_desc
 - -

ROLLUP Example

M_DESC	T_CAL_MO	N_	SUM(S_amount_sold)
-----	-----	--	-----
Internet	2000-09	GB	16569.36
Internet	2000-09	US	124223.75
Internet	2000-10	GB	14539.14
Internet	2000-10	US	137054.29
Direct Sales	2000-09	GB	85222.92
Direct Sales	2000-09	US	638200.81
Direct Sales	2000-10	GB	91925.43
Direct Sales	2000-10	US	682296.59
Internet	2000-09		140793.11
Internet	2000-10		151593.43
Direct Sales	2000-10		774222.02
Direct Sales	2000-09		723423.73
Internet			292386.54
Direct Sales			1497645.75
			1790032.29

Partial ROLLUP Example

```
SELECT      m_desc,  t_cal_month_desc, n_iso_code,
            SUM(s_amount_sold)
FROM        spctmn
WHERE       m_desc IN ('Direct Sales', 'Internet')
AND         t_cal_month_desc IN ('2000-09', '2000-10')
AND         n_iso_code IN ('GB', 'US')
GROUP BY    m_desc, ROLLUP(t_cal_month_desc, n_iso_code);
```

- m_desc is always present and not part of the rollup hierarchy
- Computes and combines the following groupings
 - m_desc, t_cal_month_desc, n_iso_code
 - m_desc, t_cal_month_desc
 - m_desc

Partial ROLLUP Example

M_DESC	T_CAL_MO	N_	SUM(S_amount_sold)
-----	-----	--	-----
Internet	2000-09	GB	16569.36
Internet	2000-09	US	124223.75
Internet	2000-10	GB	14539.14
Internet	2000-10	US	137054.29
Direct Sales	2000-09	GB	85222.92
Direct Sales	2000-09	US	638200.81
Direct Sales	2000-10	GB	91925.43
Direct Sales	2000-10	US	682296.59
Internet	2000-09		140793.11
Internet	2000-10		151593.43
Direct Sales	2000-09		723423.73
Direct Sales	2000-10		774222.02
Internet			292386.54
Direct Sales			1497645.75

CUBE

A decorative graphic element consisting of a solid teal horizontal bar that spans the width of the slide. Below this bar, on the right side, are several thin, overlapping horizontal lines in white and teal, creating a layered, modern look.

CUBE

❑ In addition to the subtotals generated by the ROLLUP extension, the CUBE extension will generate subtotals for all combinations of the dimensions specified. If "n" is the number of columns listed in the CUBE, there will be **2ⁿ subtotal combinations**.

```
SELECT fact_1_id,  
       fact_2_id,  
       SUM(sales_value) AS sales_value  
FROM   dimension_tab  
GROUP BY CUBE (fact_1_id, fact_2_id)  
ORDER BY fact_1_id, fact_2_id;
```

FACT_1_ID	FACT_2_ID	SALES_VALUE
1	1	4363.55
1	2	4794.76
1	3	4718.25
1	4	5387.45
1	5	5027.34
1		24291.35
2	1	5652.84
2	2	4583.02
2	3	5555.77
2	4	5936.67
2	5	4508.74
2		26237.04
	1	10016.39
	2	9377.78
	3	10274.02
	4	11324.12
	5	9536.08
		50528.39

CUBE with composite columns:

CUBE creates a grouping for every possible combination of columns.

```
CUBE (a, b, c)
(a, b, c)
(a, b)
(a, c)
(a)
(b, c)
(b)
(c)
()
```

CUBE Example

```
SELECT      m_desc,  t_cal_month_desc, n_iso_code,
            SUM(s_amount_sold)
FROM        spctmn
WHERE       m_desc IN ('Direct Sales', 'Internet')
AND         t_cal_month_desc IN ('2000-09', '2000-10')
AND         n_iso_code IN ('GB', 'US')
GROUP BY    CUBE(m_desc, t_cal_month_desc, n_iso_code);
```

- Produces all possible roll-up combinations
- Computes and combines the following groupings
 - m_desc, t_cal_month_desc, n_iso_code
 - m_desc, t_cal_month_desc
 - m_desc, n_iso_code
 - t_cal_month, n_iso_code
 - m_desc
 - ...

CUBE Example

M_DESC	T_CAL_MO	N_	SUM(S_AMOUNT_SOLD)
-----	-----	--	-----
Internet	2000-09	GB	16569.36
Internet	2000-09	US	124223.75
Internet	2000-10	GB	14539.14
Internet	2000-10	US	137054.29
Direct Sales	2000-09	GB	85222.92
Direct Sales	2000-09	US	638200.81
Direct Sales	2000-10	GB	91925.43
Direct Sales	2000-10	US	682296.59
	2000-09	GB	101792.28
	2000-09	US	762424.56
	2000-10	GB	106464.57
	2000-10	US	819350.88
Internet		GB	31108.5
Internet		US	261278.04
Direct Sales		GB	177148.35
Direct Sales		US	1320497.4
Internet	2000-09		140793.11
Internet	2000-10		151593.43
Direct Sales	2000-09		723423.73
Direct Sales	2000-10		774222.02
Internet			292386.54
Direct Sales			1497645.75
	2000-09		864216.84
	2000-10		925815.45
		GB	208256.85
		US	1581775.44
			1790032.29

GROUPING, GROUPING_ID, GROUPING SET

A decorative graphic consisting of several horizontal lines of varying lengths and colors (teal and white) extending from the left edge of the slide towards the right, positioned below the main title.

GROUPING

- It can be quite easy to visually identify subtotals generated by rollups and cubes, but to do it programmatically you really need something more accurate than the presence of null values in the grouping columns.
- This is where the GROUPING function comes in.
 - *It accepts a single column as a parameter and returns "1" if the column contains a null value generated as part of a subtotal by a ROLLUP or CUBE operation or "0" for any other value, including stored null values.*

```
SELECT fact_1_id,  
       fact_2_id,  
       SUM(sales_value) AS sales_value,  
       GROUPING(fact_1_id) AS f1g,  
       GROUPING(fact_2_id) AS f2g  
FROM   dimension_tab  
GROUP BY CUBE (fact_1_id, fact_2_id)  
ORDER BY fact_1_id, fact_2_id;
```

```

SELECT  fact_1_id,
        fact_2_id,
        SUM(sales_value) AS sales_value,
        GROUPING(fact_1_id) AS f1g,
        GROUPING(fact_2_id) AS f2g
FROM    dimension_tab
GROUP BY CUBE (fact_1_id, fact_2_id)
ORDER BY fact_1_id, fact_2_id;

```

FACT_1_ID	FACT_2_ID	SALES_VALUE	F1G	F2G
1	1	4363.55	0	0
1	2	4794.76	0	0
1	3	4718.25	0	0
1	4	5387.45	0	0
1	5	5027.34	0	0
1		24291.35	0	1
2	1	5652.84	0	0
2	2	4583.02	0	0
2	3	5555.77	0	0
2	4	5936.67	0	0
2	5	4508.74	0	0
2		26237.04	0	1
	1	10016.39	1	0
	2	9377.78	1	0
	3	10274.02	1	0
	4	11324.12	1	0
	5	9536.08	1	0
		50528.39	1	1

GROUPING

□ From this we can see:

1. **F1G=0,F2G=0** : Represents a row containing regular subtotal we would expect from a GROUP BY operation.
2. **F1G=0,F2G=1** : Represents a row containing a subtotal for a distinct value of the FACT_1_ID column, as generated by ROLLUP and CUBE operations.
3. **F1G=1,F2G=0** : Represents a row containing a subtotal for a distinct value of the FACT_2_ID column, which we would only see in a CUBE operation.
4. **F1G=1,F2G=1** : Represents a row containing a ***grand total for the query***, as generated by ROLLUP and CUBE operations.

GROUPING WITH FILTER OPTION:

- The GROUPING columns can be used for ordering or filtering results.

```
SELECT fact_1_id,  
       fact_2_id,  
       SUM(sales_value) AS sales_value,  
       GROUPING(fact_1_id) AS f1g,  
       GROUPING(fact_2_id) AS f2g  
FROM   dimension_tab  
GROUP BY CUBE (fact_1_id, fact_2_id)  
HAVING GROUPING(fact_1_id) = 1 OR GROUPING(fact_2_id) = 1  
ORDER BY GROUPING(fact_1_id), GROUPING(fact_2_id);
```

FACT_1_ID	FACT_2_ID	SALES_VALUE	F1G	F2G
1		24291.35	0	1
2		26237.04	0	1
	4	11324.12	1	0
	3	10274.02	1	0
	2	9377.78	1	0
	1	10016.39	1	0
	5	9536.08	1	0
		50528.39	1	1

GROUPING_ID

- ❑ The GROUPING_ID function provides an alternate and more compact way to identify subtotal rows.
- ❑ Passing the dimension columns as arguments, it returns a number indicating the GROUP BY level.

```
SELECT fact_1_id,  
       fact_2_id,  
       SUM(sales_value) AS sales_value,  
       GROUPING_ID(fact_1_id, fact_2_id) AS grouping_id  
FROM   dimension_tab  
GROUP BY CUBE (fact_1_id, fact_2_id)  
ORDER BY fact_1_id, fact_2_id;
```

```

SELECT fact_1_id,
       fact_2_id,
       SUM(sales_value) AS sales_value,
       GROUPING_ID(fact_1_id, fact_2_id) AS grouping_id
FROM   dimension_tab
GROUP BY CUBE (fact_1_id, fact_2_id)
ORDER BY fact_1_id, fact_2_id;

```

FACT_1_ID	FACT_2_ID	SALES_VALUE	GROUPING_ID
1	1	4363.55	0
1	2	4794.76	0
1	3	4718.25	0
1	4	5387.45	0
1	5	5027.34	0
1		24291.35	1
2	1	5652.84	0
2	2	4583.02	0
2	3	5555.77	0
2	4	5936.67	0
2	5	4508.74	0
2		26237.04	1
	1	10016.39	2
	2	9377.78	2
	3	10274.02	2
	4	11324.12	2
	5	9536.08	2
		50528.39	3

GROUPING SETS

- ❑ Calculating all possible subtotals in a cube, especially those with many dimensions, can be quite an intensive process.
- ❑ If you don't need all the subtotals, this can represent a considerable amount of wasted effort. The following cube with three dimensions gives 8 levels of subtotals (GROUPING_ID: 0-7)

```
SELECT fact_1_id,  
       fact_2_id,  
       fact_3_id,  
       SUM(sales_value) AS sales_value,  
       GROUPING_ID(fact_1_id, fact_2_id, fact_3_id) AS grouping_id  
FROM   dimension_tab  
GROUP BY CUBE(fact_1_id, fact_2_id, fact_3_id)  
ORDER BY fact_1_id, fact_2_id, fact_3_id;
```

GROUPING SETS

- ❑ If we only need a few of these levels of subtotalling we can use the **GROUPING SETS** expression and specify exactly which ones we need, saving us having to calculate the whole cube.
- ❑ In the following query we are only interested in subtotals for the "FACT_1_ID, FACT_2_ID" and "FACT_1_ID, FACT_3_ID" groups.

```
SELECT fact_1_id,  
       fact_2_id,  
       fact_3_id,  
       SUM(sales_value) AS sales_value,  
       GROUPING_ID(fact_1_id, fact_2_id, fact_3_id) AS grouping_id  
FROM   dimension_tab  
GROUP BY GROUPING SETS((fact_1_id, fact_2_id), (fact_1_id, fact_3_id))  
ORDER BY fact_1_id, fact_2_id, fact_3_id;
```


GROUPING SETS

FACT_1_ID	FACT_2_ID	FACT_3_ID	SALES_VALUE	GROUPING_ID
1	1		4363.55	1
1	2		4794.76	1
1	3		4718.25	1
1	4		5387.45	1
1	5		5027.34	1
1		1	2737.4	2
1		2	1854.29	2
1		3	2090.96	2
1		4	2605.17	2
1		5	2590.93	2
1		6	2506.9	2
1		7	1839.85	2
1		8	2953.04	2
1		9	2778.75	2
1		10	2334.06	2
2	1		5652.84	1
2	2		4583.02	1
2	3		5555.77	1
2	4		5936.67	1
2	5		4508.74	1
2		1	3512.69	2
2		2	2847.94	2
2		3	2972.5	2
2		4	2534.06	2
2		5	3115.99	2
2		6	2775.85	2
2		7	2208.19	2
2		8	2358.55	2
2		9	1884.11	2
2		10	2027.16	2

Window Function

- ❑ Rank () Over
- ❑ Rank () Over with Partitioning

Window Function:

- Basic syntax:

`WFctType(expr) OVER (WPartitioning WOrdering Wframe)`

- WPartitioning

- Divides the table into windows, i.e., groups of rows.
- Windows are created after groupings and aggregations, and can refer to any aggregate results.

- WOrdering

- Determines the ordering in which the rows are passed to the window function
- Many window functions are sensitive to the ordering of rows

RANK Example

```
SELECT      m_desc, SUM(s_amount_sold),  
            RANK() OVER (ORDER BY SUM(s_amount_sold))  
FROM        bi.spctmn  
WHERE       m_desc IN ('Direct Sales', 'Internet')  
AND         t_cal_month_desc IN ('2000-09', '2000-10')  
AND         n_iso_code = 'US'  
GROUP BY    m_desc;
```

- RANK() assigns the rank to each row according to the order of the total amount sold.
- The ordering attribute or expression must be specified.
- The ordering can be ASC (default) or DESC.

RANK Example

```
SELECT      m_desc, SUM(s_amount_sold),  
            RANK() OVER (ORDER BY SUM(s_amount_sold))  
FROM        bi.spctmn  
WHERE       m_desc IN ('Direct Sales', 'Internet')  
AND         t_cal_month_desc IN ('2000-09', '2000-10')  
AND         n_iso_code = 'US'  
GROUP BY    m_desc;
```

M_DESC	SUM(S_AMNT_SOLD)	RANK
-----	-----	-----
Internet	261278.04	1
Partners	800871.37	2
Direct Sales	1320497.4	3

RANK with Partitioning Example

```
SELECT      m_desc, t_cal_month_desc, SUM(s_amnt_sold),  
            RANK() OVER (PARTITION BY m_desc  
                           ORDER BY SUM(s_amnt_sold) DESC)  
            AS RANK_BY_MEDIA  
FROM        bi.spctmn  
WHERE       t_cal_month_desc IN  
            ('2000-08', '2000-09', '2000-10', '2000-11')  
AND        m_desc IN ('Direct Sales', 'Internet')  
GROUP BY    m_desc, t_cal_month_desc;
```

- If a PARTITION BY clause is specified the rank is computed independently for each window (i.e., the rank is reset for each window).

RANK with Partitioning Example

Media	Month	AmntSold	RankByMedia
.....
Direct Sales	2000-08	1236104.31	1
Direct Sales	2000-10	1225584.31	2
Direct Sales	2000-09	1217807.75	3
Direct Sales	2000-11	1115239.03	4
Internet	2000-11	284741.77	1
Internet	2000-10	239236.26	2
Internet	2000-09	228241.24	3
Internet	2000-08	215106.56	4

RATIO_TO_REPORT Example

```
SELECT      m_desc,
            SUM(s_amnt_sold) AS SALES,
            SUM(SUM(s_amnt_sold)) OVER () AS TOTAL_SALES,
            RATIO_TO_REPORT(SUM(s_amnt_sold)) OVER () AS RATIO
FROM        bi.spctmn
WHERE       s_t_id = to_DATE('11-OCT-2000')
GROUP BY    m_desc;
```

- For each media compute the total amount sold and the ration wrt the overall total amount sold (across all media).

M_DESC	SALES	TOTAL_SALES	RATIO
.....
Direct Sales	14447.23	23183.45	.623169977
Internet	345.02	23183.45	.014882168
Partners	8391.2	23183.45	.361947855

OLAP Models: MOLAP, ROLAP, HOLAP

Discuss various OLAP Models with their Architecture (Dec 2017, May 2016 10 marks)

Different forms of OLAP

Three ways of storing data:

1. Multidimensional OLAP (MOLAP)
 - Best Query Performance
2. Relational OLAP (ROLAP)
 - Ideal for large databases
3. Hybrid OLAP (HOLAP)
 - Best of both worlds!

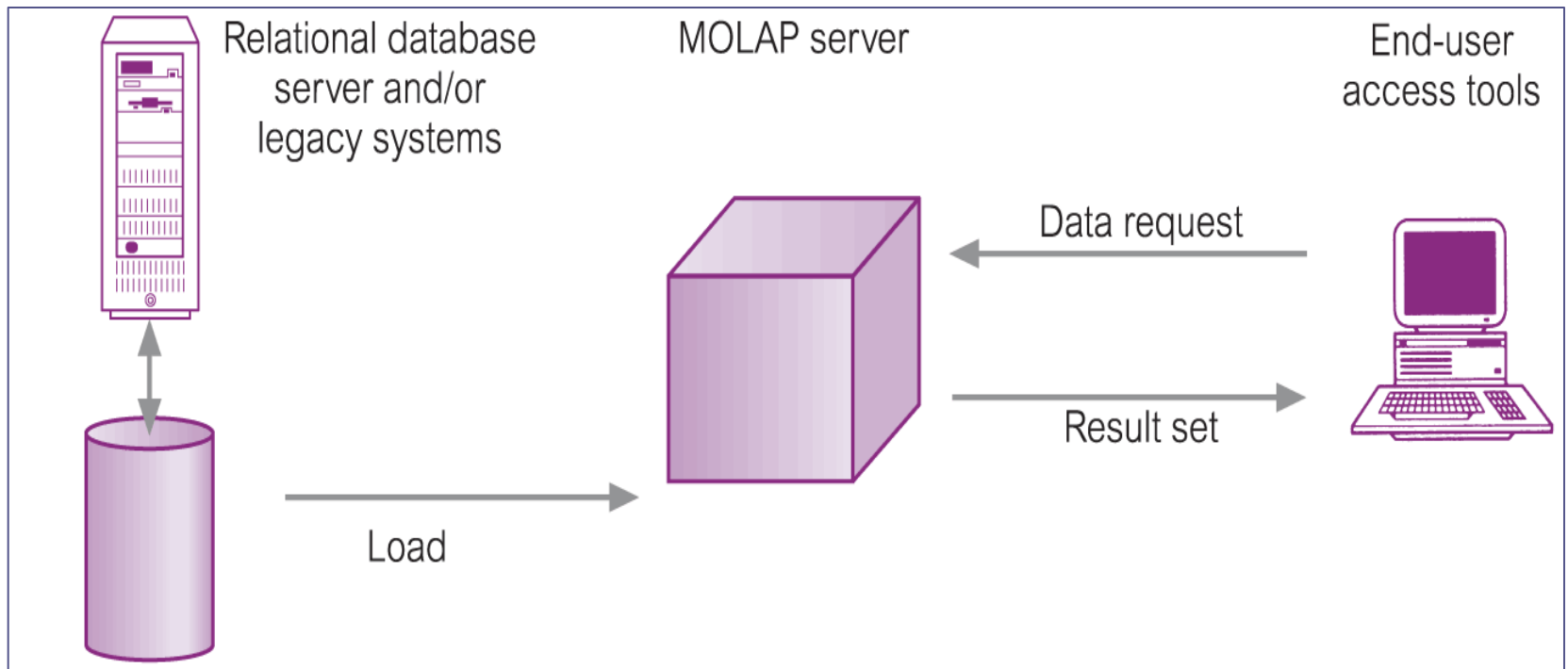
Multi-dimensional OLAP (MOLAP)

- Use specialized data structures and multi-dimensional Database Management Systems (MDDDBMSs) to organize, navigate, and analyse data.
- Data is typically aggregated and stored according to predicted usage to enhance query performance.
- Use array technology and efficient storage techniques that minimize the disk space requirements through sparse data management.
- Provides excellent performance when data is used as designed, and the focus is on data for a specific decision-support application.

Multi-dimensional OLAP (MOLAP)

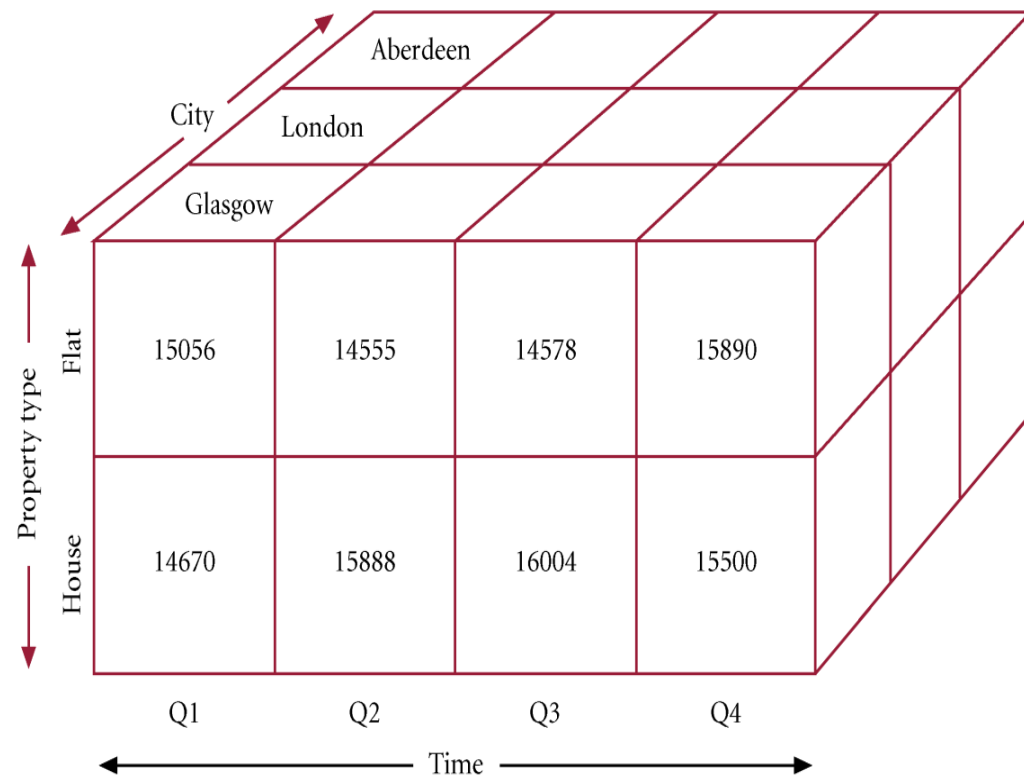
- Traditionally, require a tight coupling with the application layer and presentation layer.
- Recent trends segregate the OLAP from the data structures through the use of published application programming interfaces (APIs).

Typical Architecture for MOLAP Tools

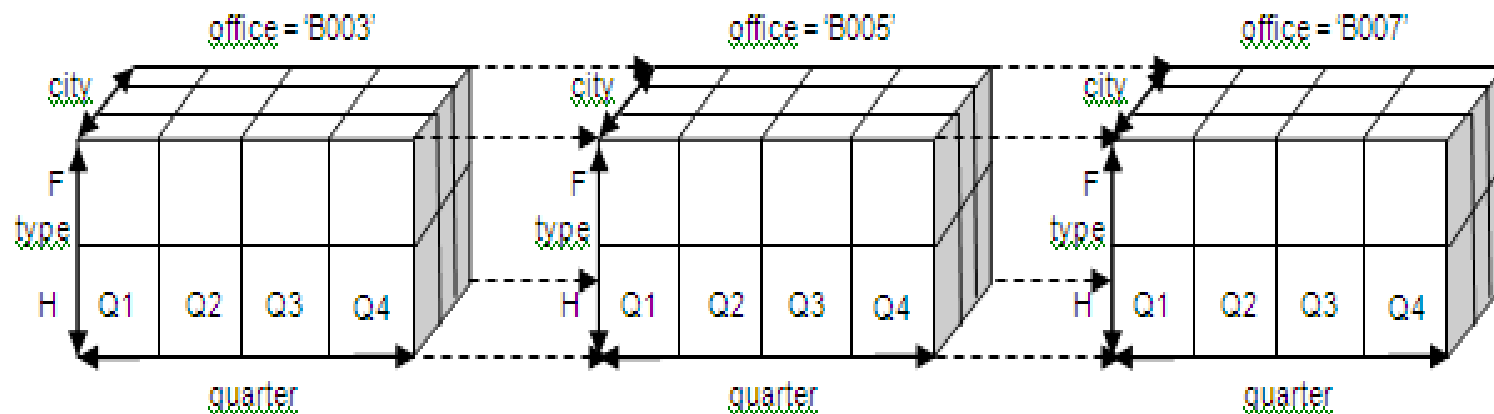


Multi-dimensional Data as 4-field

Property Type	City	Time	Total Revenue
Flat	Glasgow	Q1	15056
House	Glasgow	Q1	14670
Flat	Glasgow	Q2	14555
House	Glasgow	Q2	15888
Flat	Glasgow	Q3	14578
House	Glasgow	Q3	16004
Flat	Glasgow	Q4	15890
House	Glasgow	Q4	15500
Flat	London	Q1	19678
House	London	Q1	23877
Flat	London	Q2	19567
House	London	Q2	28677
.....
.....



Multi-dimensional Data as series of 3-D Cubes



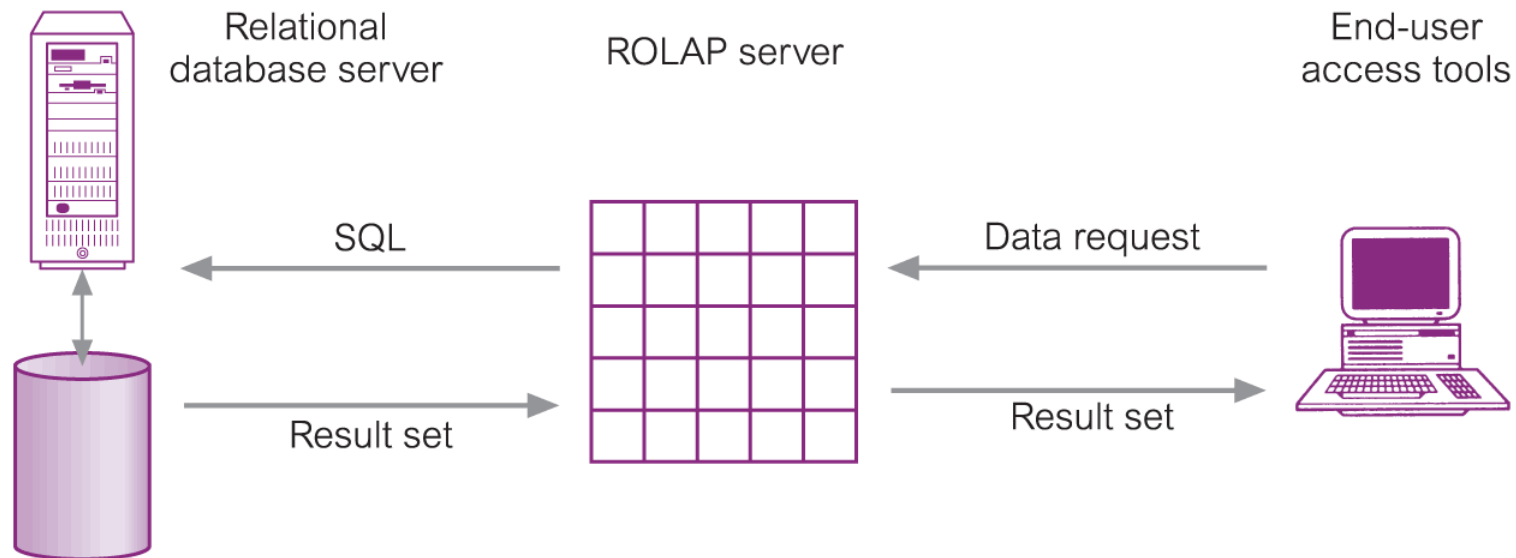
MOLAP Tools - Development Issues

- Underlying data structures are limited in their ability to support multiple subject areas and to provide access to detailed data.
- Navigation and analysis of data is limited because the data is designed according to previously determined requirements.
- MOLAP products require a different set of skills and tools to build and maintain the database, thus increasing the cost and complexity of support.

Relational OLAP (ROLAP)

- Fastest-growing style of OLAP technology due to requirements to analyze ever-increasing amounts of data and the realization that users cannot store all the data they require in MOLAP databases.
- Supports RDBMS products using a metadata layer avoids need to create a static multi-dimensional data structure facilitates the creation of multiple multi-dimensional views of the two-dimensional relation.
- To improve performance, some products use SQL engines to support the complexity of multi-dimensional analysis, while others recommend, or require, the use of highly de-normalized database designs such as the star schema.

Typical Architecture for ROLAP Tools



Multi-dimensional Data as 3-field

City	Time	Total Revenue
Glasgow	Q1	29726
Glasgow	Q2	30443
Glasgow	Q3	30582
Glasgow	Q4	31390
London	Q1	43555
London	Q2	48244
London	Q3	56222
London	Q4	45632
Aberdeen	Q1	53210
Aberdeen	Q2	34567
Aberdeen	Q3	45677
Aberdeen	Q4	50056
.....
.....

	City				
Time	City	Glasgow	London	Aberdeen
	Quarter				
	Q1	29726	43555	53210
	Q2	30443	48244	34567
	Q3	30582	56222	45677
	Q4	31390	45632	50056

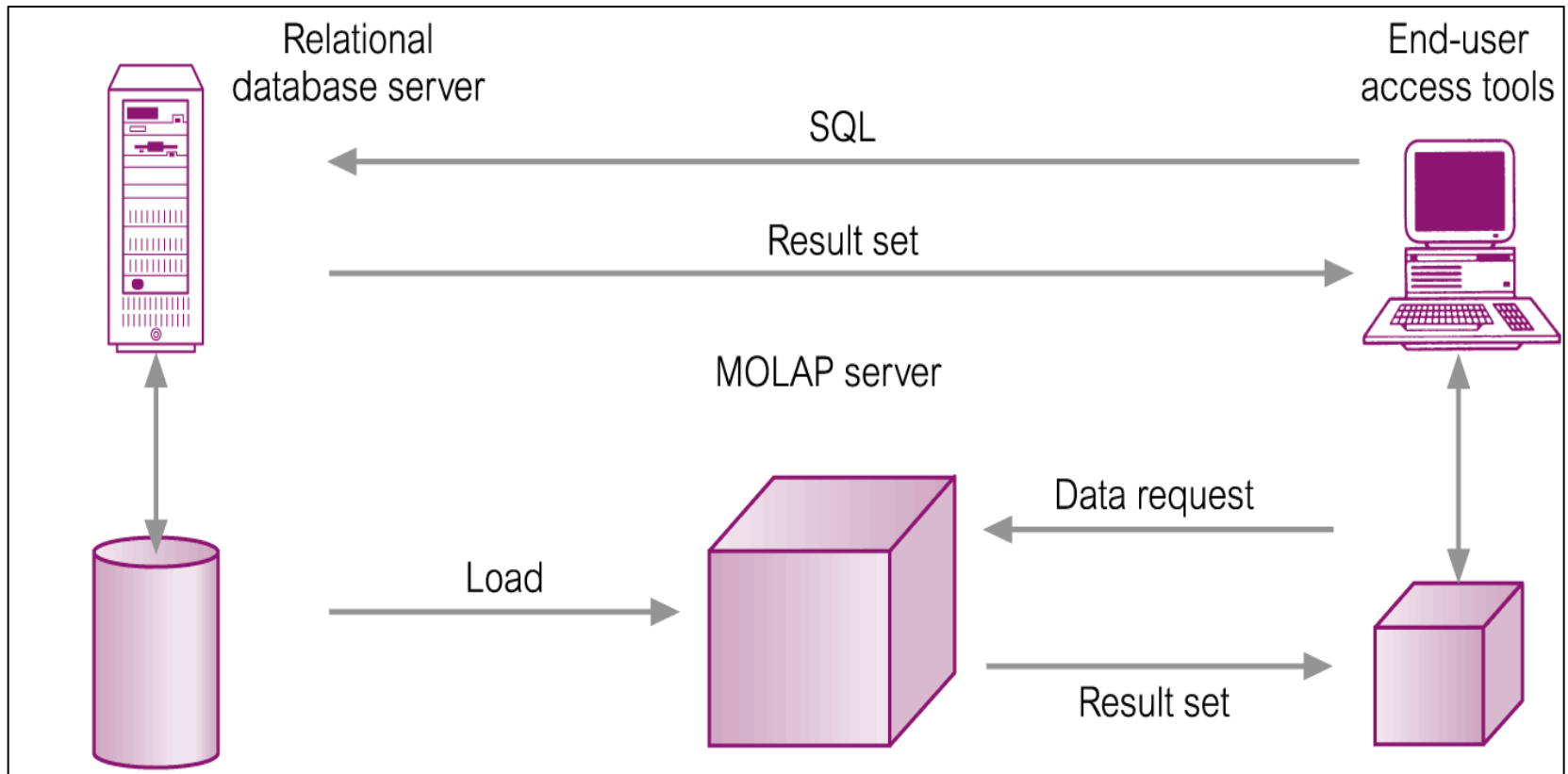
ROLAP Tools - Development Issues

- Performance problems associated with the processing of complex queries that require multiple passes through the relational data.
- Middleware to facilitate the development of multi-dimensional applications. (Software that converts the two-dimensional relation into a multi-dimensional structure).
- Development of an option to create persistent, multi-dimensional structures with facilities to assist in the administration of these structures.

Hybrid OLAP (HOLAP)

- Hybrid OLAP (HOLAP) refers to technologies that combine the advantages of MOLAP and ROLAP.
- For summary-type information, HOLAP leverages cube technology for faster performance.
- When detail information is needed, HOLAP can "drill through" from the cube into the underlying relational data.
 - MOLAP provides faster computation but supports smaller amount of data than ROLAP
 - ROLAP provide scalability, SQL standard has been extended

Typical Architecture for HOLAP Tools



HOLAP Tools - Development Issues

- Architecture results in significant data redundancy and may cause problems for networks that support many users.
- Ability of each user to build a custom data cube may cause a lack of data consistency among users.
- Only a limited amount of data can be efficiently maintained.

THANK YOU

A decorative graphic element consisting of a solid teal horizontal bar that spans the width of the slide. Below this bar, on the right side, are several thin, overlapping horizontal lines in white and teal, creating a layered, modern look.