Data Warehousing Conclusion

Esteban Zimányi
ezimanyi@ulb.ac.be
Slides by Toon Calders

Motivation for the Course

- Database = a piece of software to handle data:
  - Store, maintain, and query
- Most ideal system situation-dependent
  - data type: simple / semi-structured / complex / …
  - types of queries: simple lookup / analytical / …
  - type of usage: multi-user / single-user / distributed / …

Example Question

- What are the ACID properties? Illustrate these properties with your own examples. Why are they important for OLTP?
  - Atomicity
  - Consistency
  - Isolation
  - Durability

What About Decision Support?

- Concurrent access
  - →not really
  - →read-only
- Data consistency, non-redundancy
  - →data comes from consistent sources (sort of)
  - →data does not change during analysis; once clean, always clean

What About Decision Support?

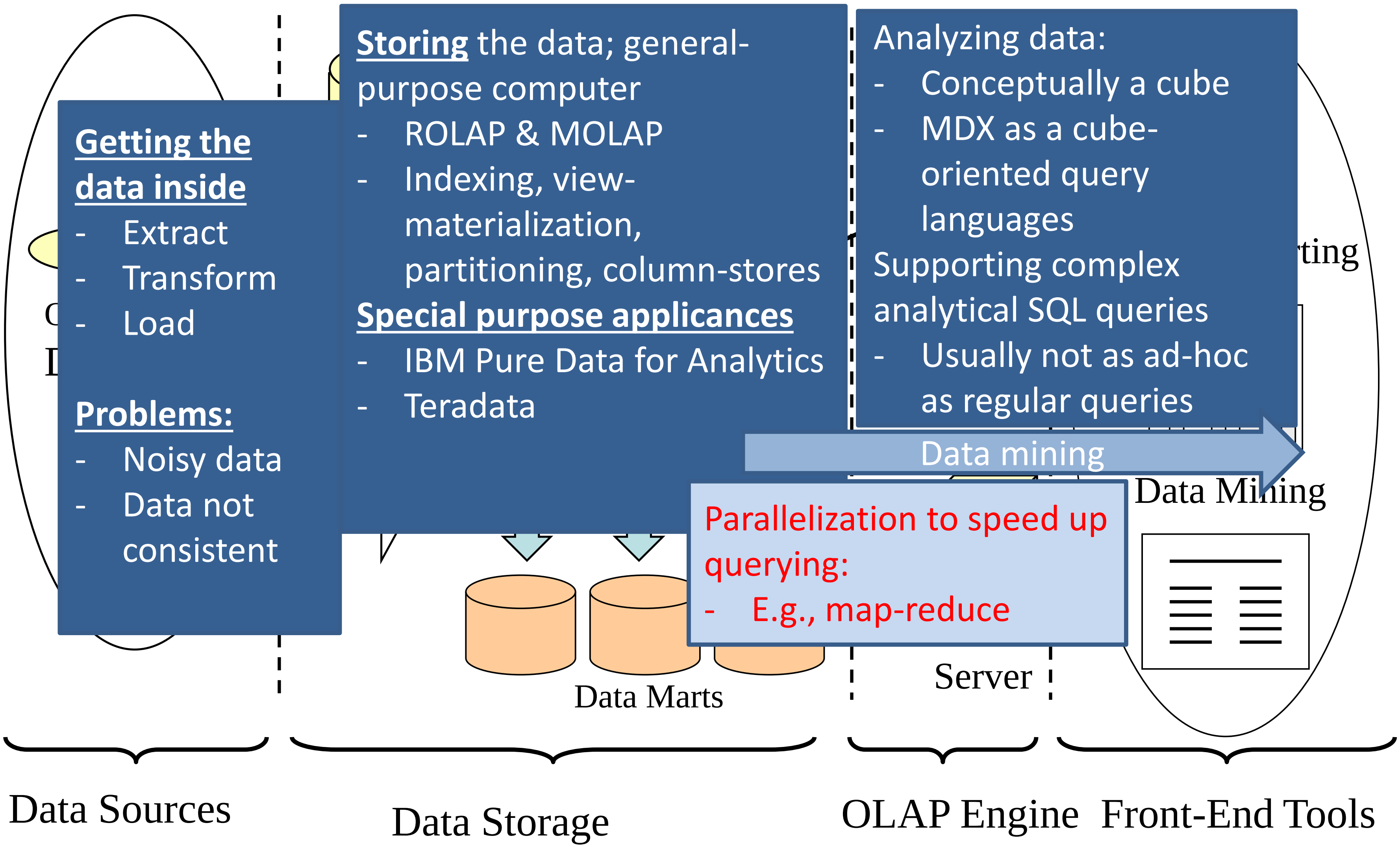
- Ad-hoc Querying
  - → No longer true;
  - → Spread-sheet like queries
  - → Long-running queries, touching large parts of the database
    - → In combination with transactions, kills the database

- Efficiency
  - → Relational DBMS optimized for other types of queries

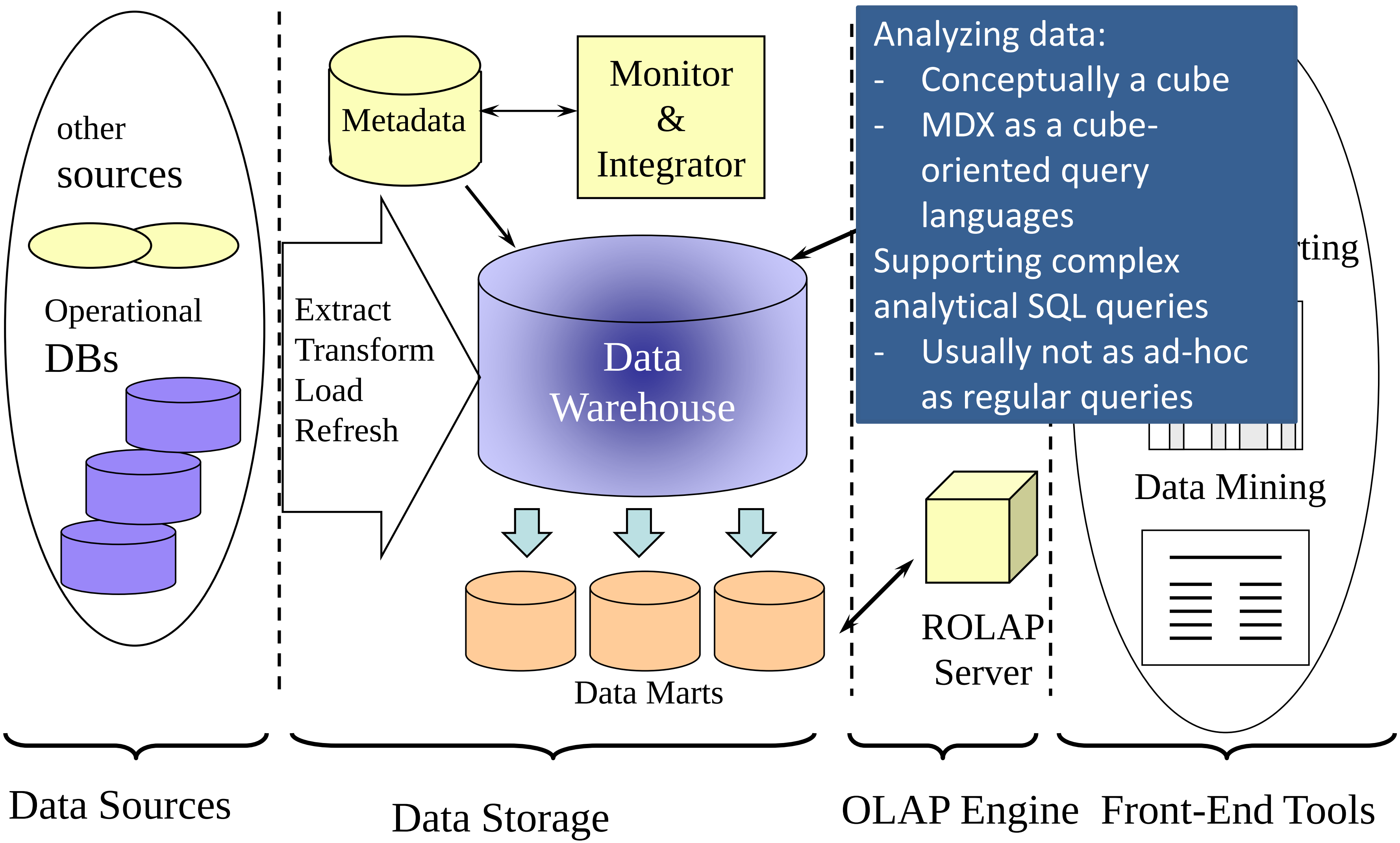
What About Decision Support?

- OLTP systems not very efficient for data analysis tasks
  - analysis queries might stall operational systems
  - architecture suboptimal
    - different indexing stuctures
    - denormalization
  - need of historical data versus only current data

Three-Tier Architecture



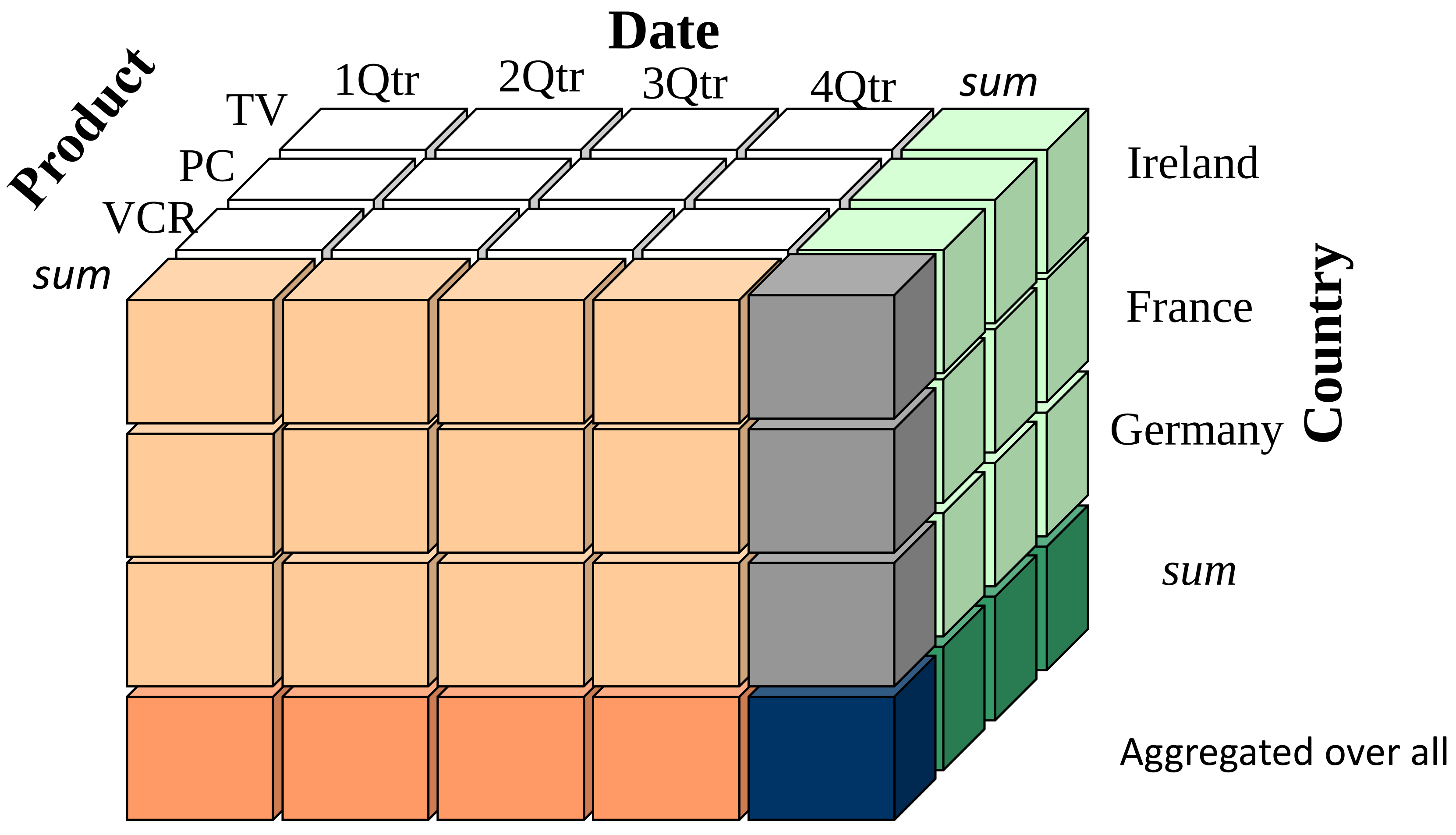
Three-Tier Architecture



Analyzing data:
- Conceptually a cube
- MDX as a cube-oriented query languages

Supporting complex analytical SQL queries
- Usually not as ad-hoc as regular queries

Data Cubes as the Conceptual Model



Querying the Cube

- Slice, Dice, Roll-up, Drill- Down
  - Cube browsing tools

- MDX as the SQL for OLAP
  - Select dimensions for display
  - Define new aggregates
  - Select slices
  - …

Example Question

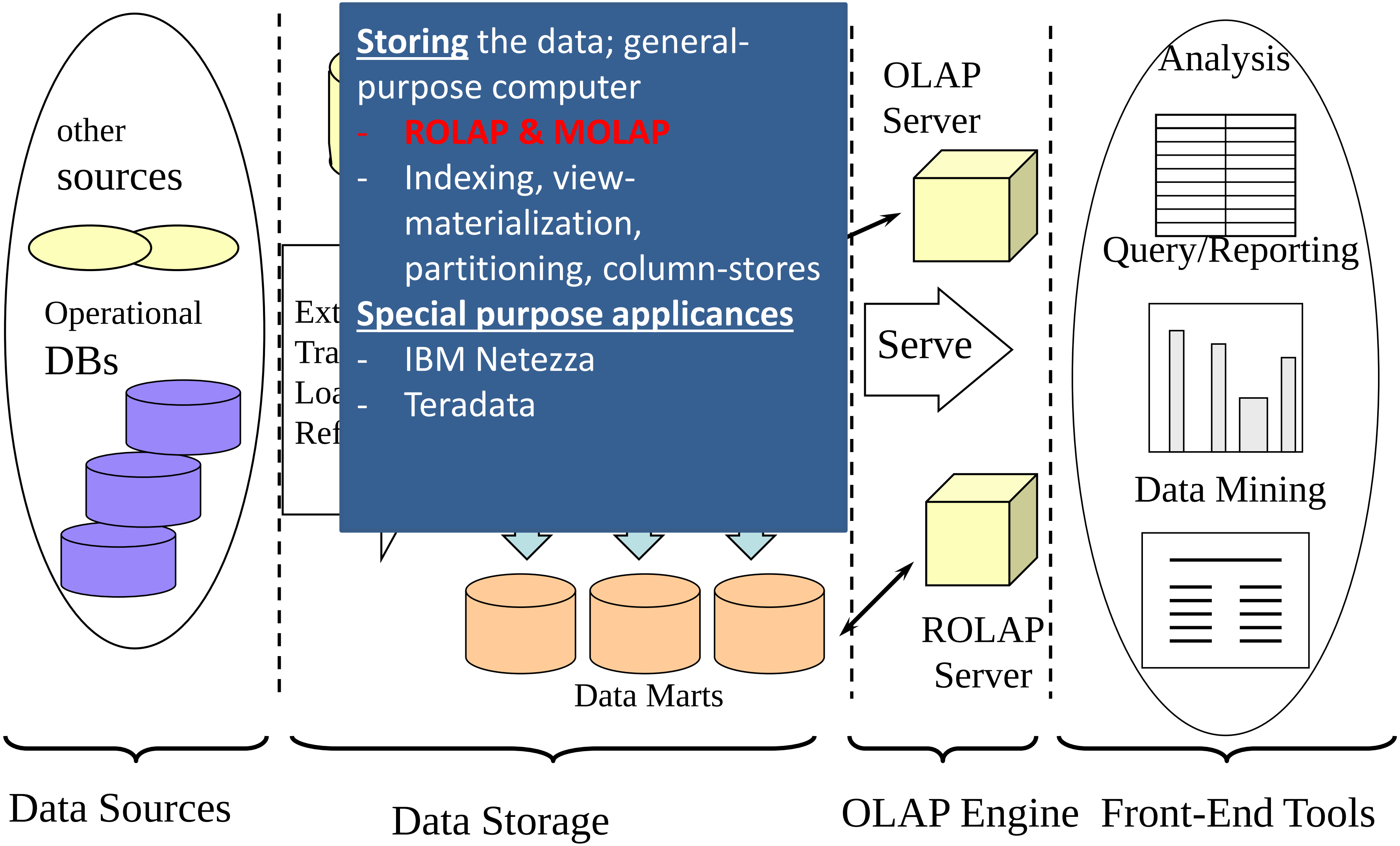
- Explain the meaning of the following MDX query:

```
select
    [Customer].[Gender].members on columns,
    ( { [France], [Germany] }, education.members ) on
    rows
from [Adventure Works]
where (  [Customer Count],
    {[Commute Distance].[0-1 Miles],
    [Commute Distance].[1-2 Miles]}  )
```

Result

		All Customers	Female	Male
France	All Customers	1,161	573	588
France	Bachelors	282	132	150
France	Graduate Degree	160	83	77
France	High School	199	108	91
France	Partial College	329	158	171
France	Partial High School	191	92	99
Germany	All Customers	1,225	592	633
Germany	Bachelors	343	175	168
Germany	Graduate Degree	168	82	86
Germany	High School	153	64	89
Germany	Partial College	383	185	198
Germany	Partial High School	178	86	92

Three-Tier Architecture



<u>**Storing**</u> the data; general-purpose computer

- **ROLAP & MOLAP**
- Indexing, view-materialization, partitioning, column-stores

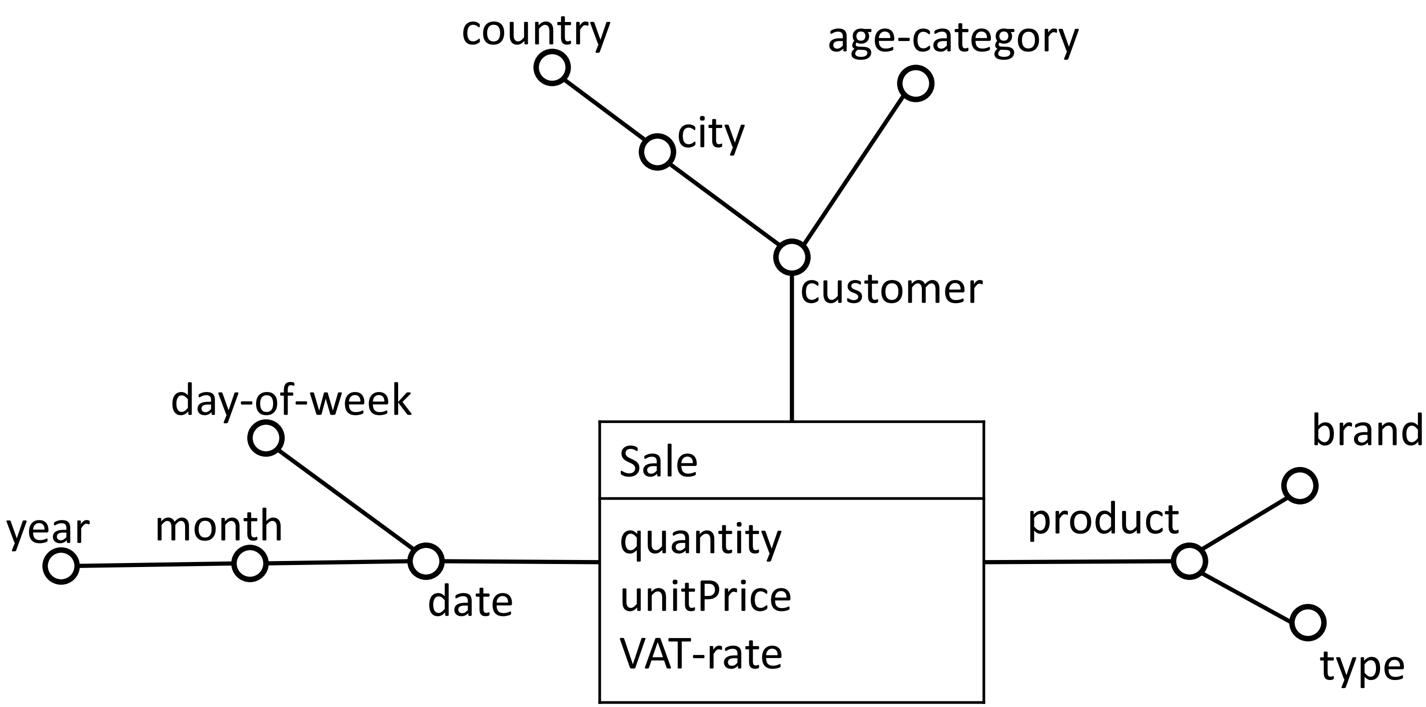
<u>**Special purpose applicances**</u>

- IBM Netezza
- Teradata

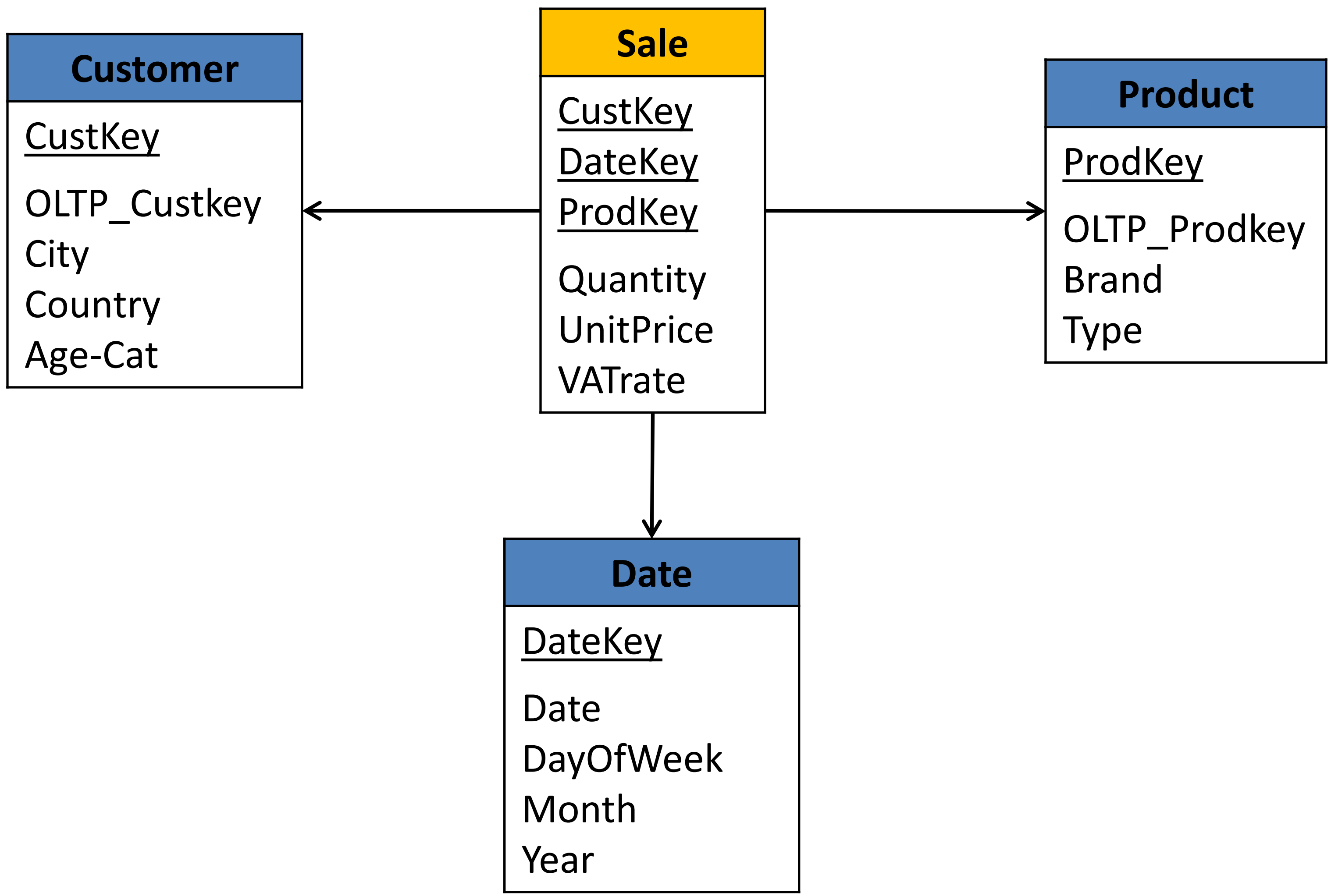
Logical Model - ROLAP

- Modeling your data
  - Dimensional modeling
    - Fact, dimension, measure
- Cubes and pre-materialized views need to be stored in a convenient format
  - ROLAP
    - Star schema / snowflake schema
    - Dimensional modeling
  - MOLAP

Dimensional Fact Model



Corresponding Star-Schema



Dimensional Modeling

- Different ways to deal with
  - Slowly changing dimensions
  - Unbalanced hierarchies; non-covering hierarchies
  - Junk dimensions

- These are best-practices for storing dimensional data in a relational database
  - New technologies may change the rules of the game

Slowly Changing Dimension – Type 2

- Whenever there is a change, create a new version of the affected row
  - Need for surrogate key!

SID	CID	Name	Address
1	001	John	Dallas
2	002	Mary	Dallas
3	003	Pete	New York

SID	CID	Name	Address
1	001	John	Dallas
2	002	Mary	Dallas
3	003	Pete	New York
4	001	John	New York
5	002	Mary	New York
6	004	Mark	Dallas

John and Mary move to New York
Mark is a new client

Exam Question

- Explain the concept of a mini-dimension and illustrate this concept with an original example.

Solution 4: Dimension Splitting

Customer dimension (original)

CustID
Name
PostalAddress
Gender
DateofBirth
Customerside
…
NoKids
MaritialStatus
CreditScore
BuyingStatus
Income
Education
…

CustID
Name
PostalAddress
Gender
DateofBirth
Customerside
…

Customer dimension (new):

relatively static attributes

DemographyID
NoKids
MaritialStatus
CreditScoreGroup
BuyingStatusGroup
IncomeGroup
EducationGroup
…

Demographics dimension:

often-changing attributes

Mini-dimension

Facts

Skey	Date	sum
1	D1	5
1	D2	6
2	D3	8
2	D4	3
3	D5	6
4	D6	3
5	D7	6
6	D8	4
7	D9	8
8	D10	9
9	D11	3
…	…	…

Customer

SKey	CID	status	child	car
1	1	single	0	no
2	1	married	0	no
3	1	single	0	no
4	1	single	0	yes
5	1	married	0	yes
6	1	married	1	yes
7	1	married	1	no
8	1	married	2	no
9	1	married	2	yes

Mini-dimension

Customer

SKey	CID	status
1	1	single
2	1	married
3	1	single
4	1	married

Facts

Skey	Dkey	Date	sum
1	1	D1	5
1	1	D2	6
2	1	D3	8
2	1	D4	3
3	1	D5	6
3	2	D6	3
4	2	D7	6
4	4	D8	4
4	3	D9	8
4	5	D10	9
4	6	D11	3
…	…	…	…

Demography

DKey	child	car
1	0	no
2	0	yes
3	1	no
4	1	yes
5	2	no
6	2	yes

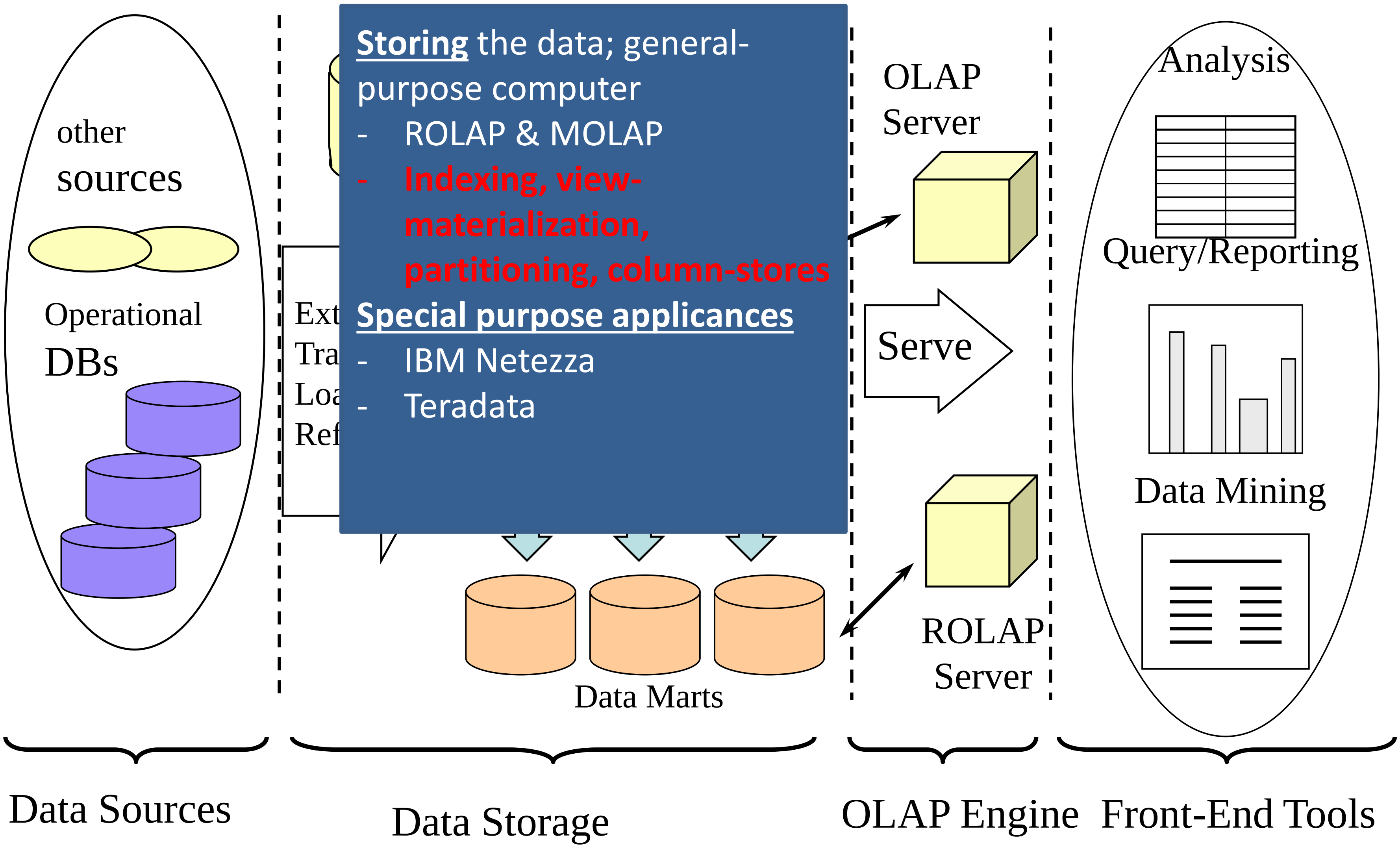
Exam question

4. (2 points)
Consider a data cube with dimensions Student, Course, Semester, and Lecturer, and measure attributes Grade and Number_of_attempts. The following hierarchies are present:

   – Student → Home_town → Country,
   – Semester → Year, and
   – Lecturer → Degree (e.g. full/associate/assistant professor, postdoc, ...)

   (a) Give a snowflake scheme for this data cube.
   (b) Use the schema of (a) to give an example of a join index and explain which queries would benefit from this example join index.

Three-Tier Architecture



Physical Level

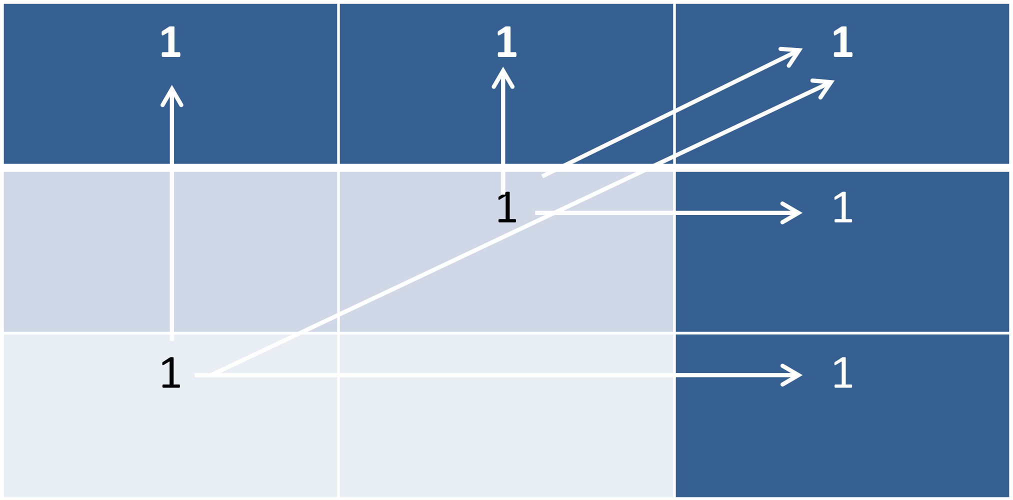
- Speeding up typical data warehousing queries
  - Data Explosion Problem
  - Materialization
  - Indices
    - bitmap index, Join index, bitmap-join index
  - Partitioning tables

Example Question

- Explain why in general it is not possible to store fully materialized data cubes.
  - High dimensionality, sparse data
    → Cube exponentially larger than original data
  - No problem if cube is dense
  - Less of a problem if dimensionality is low

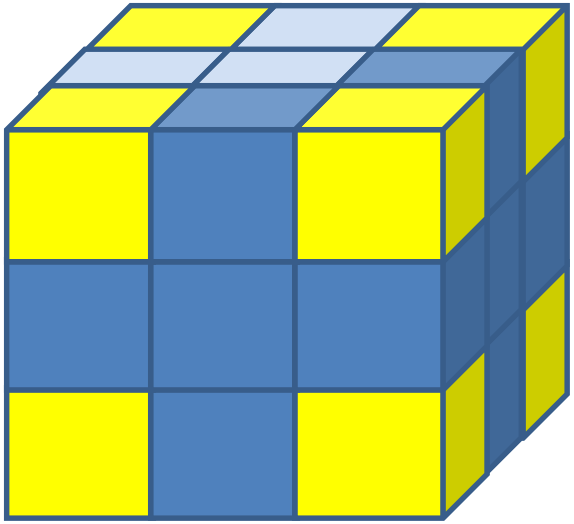
Inherent problem; impossible to pre-compute all possible ways to aggregate the data

Database Explosion Problem



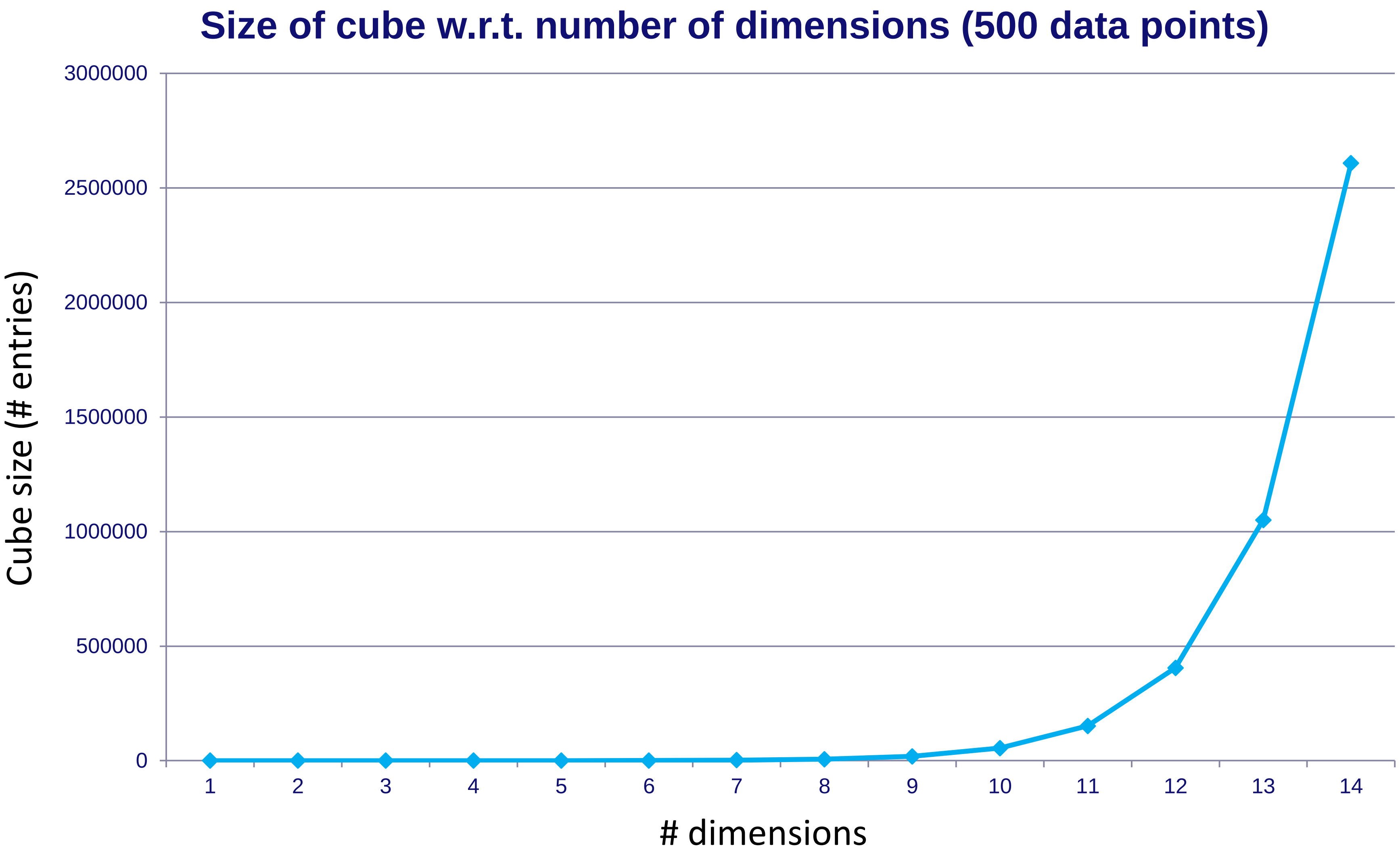
2D: adding 1 tuple → affecting 4 cells of the cube

Database Explosion Problem



2D: adding 1 tuple → affecting 4 cells of the cube
3D: adding 1 tuple → affecting 8 cells of the cube
…
kD: adding 1 tuple → affecting 2^k cells of the cube

Data Explosion Problem



Storing the Data

- Want quick answers → pre-computation
- Straightforward solution, however, does not work → Data explosion problem
- Therefore, <u>partially</u> materialize the cube + smart indexing and storage structures

- ROLAP and MOLAP
  - Often hybrid form

Materialization

Example:

(part, customer)

SELECT customer, part, sum(sales)
FROM Sales
GROUP BY customer, part

| Sales |

(part)

SELECT part, sum(sales)
FROM Sales
GROUP BY part

| Sales |

Materialization

Example:

(part, customer)

SELECT customer, part, sum(sales)
FROM Sales
GROUP BY customer, part

materialized as PC

|PC|

(part)

SELECT part, sum(sales)
FROM S PC
GROUP BY part

|PC|

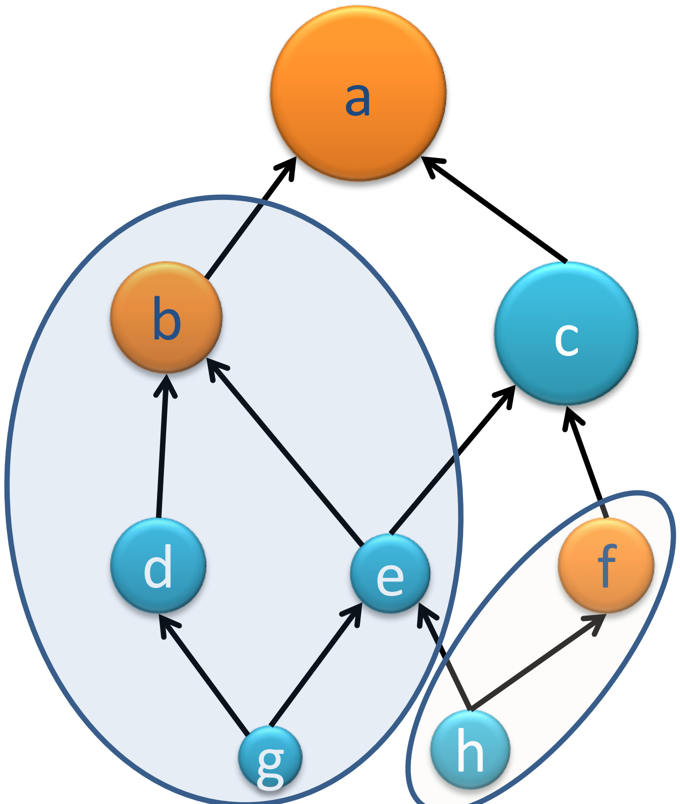
Example: some materialized

Query	Answer	Cost
• (part,supplier,customer)	6M	6M
• (part,customer)	6M	6M
• (part,supplier)	0.8M	0.8M
• (supplier,customer)	6M	6M
• (part)	0.2M	0.8M
• (supplier)	0.01M	0.8M
• (customer)	0.1M	0.1M
• ()	1	0.1M

Total cost: 20.6M

Example

- Base table a, table b, and f are materialized
  - Total: 2 x 100 + 4 x 50 + 2 x 40 = 480



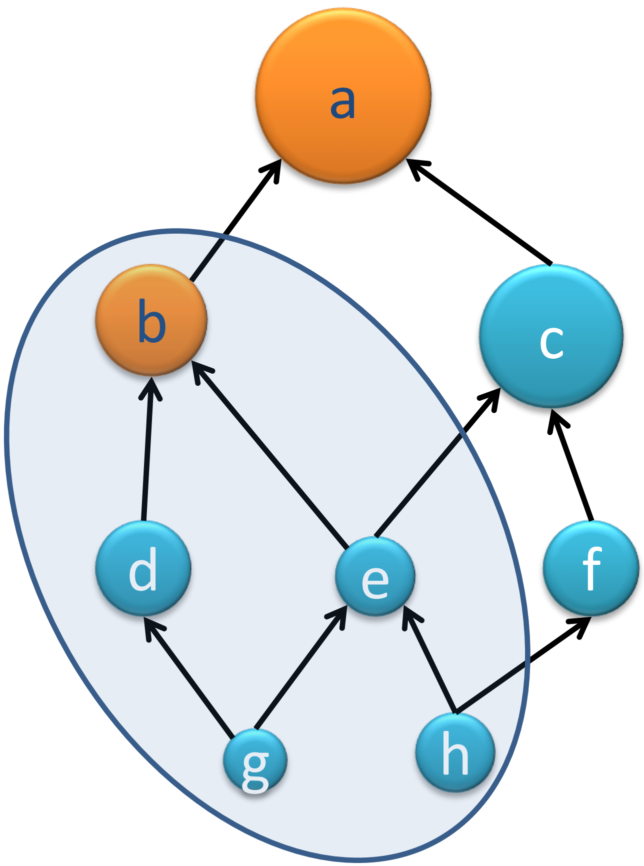
Query	Size	Uses?	Cost	Benefit
a	100	a	100	-
b	50	b	50	-
c	75	a	100	-
d	20	b	50	-
e	30	b	50	-
f	40	**f**	**40**	**60**
g	1	b	50	-
h	10	**f**	**40**	**10**

- Additional benefit of materializing f
  = 70 = 1 x (100-40) + 1 x (50-40)

Example

- Benefit for materializing the other tables:

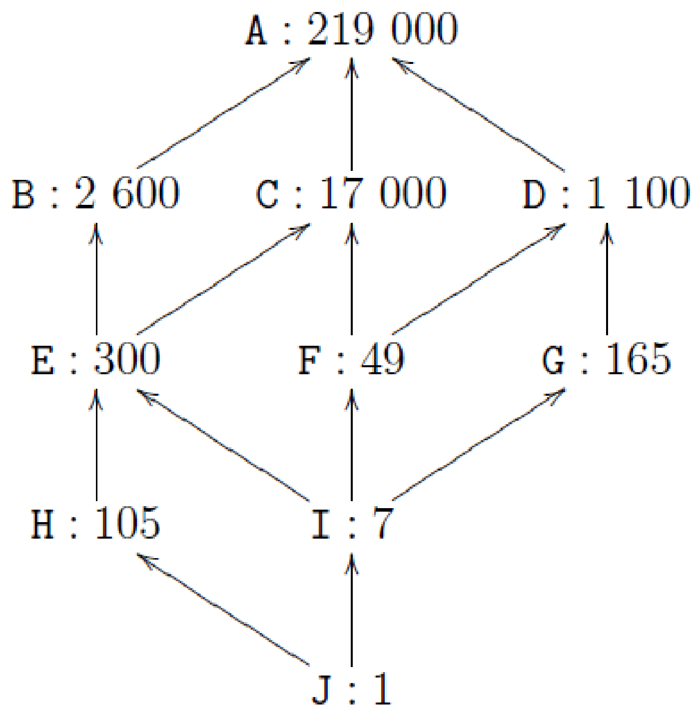
a	100
b	50
c	75
d	20
e	30
f	40
g	1
h	10



query	Materialized view					
	c	d	e	f	g	h
a	-	-	-	-	-	-
b	-	-	-	-	-	-
c	**25**	-	-	-	-	-
d	-	**30**	-	-	-	-
e	-	-	**20**	-	-	-
f	**25**	-	-	**60**	-	-
g	-	**30**	**20**	-	**49**	-
h	-	-	**20**	**10**	-	**40**
Total	**50**	**60**	**60**	**<u>70</u>**	**49**	**40**

Exam Question

3. (3p) Consider the following lattice of views along with a representation of the number of rows in each view where A is the base cuboid:



(a) Apply the greedy method described by *Harinarayan, Rajaraman, and Ullman* in their seminal paper "Implementing Data Cubes Efficiently" (SIGMOD 1996) to select 2 views from the views B-J to materialize.

(b) What is the total benefit of materializing these two views? (You can give a formula involving parentheses, multiplications, divisions, additions, subtractions, powers)

Bitmap-Join Index: Example

Date	pID	Client
10/5/12	1	Jack
10/5/12	1	Pete
13/5/12	3	John
14/5/12	2	Mary

SP_category_bjidx

Category	Bitmap
Non-food	1100
Food	0011

SC_city_bjidx

City	Bitmap
Brussels	1001
Eindhoven	0110

SELECT date

FROM Sales S join Product P
join Customer C on ...

WHERE

P.Category = “Food” and

C.City = “Brussels”;

Bitmap-Join Index: Example

Date	pID	Client
10/5/12	1	Jack
10/5/12	1	Pete
13/5/12	3	John
14/5/12	**2**	**Mary**

SP_category_bjidx

Category	Bitmap
Non-food	1100
Food	**0011**

SC_city_bjidx

City	Bitmap
Brussels	**1001**
Eindhoven	0110

SELECT date

FROM Sales S join Product P join Customer C on ...

WHERE

P.Category = "Food" and

C.City = "Brussels";

0011 & 1001 → 0001

Partitioning

- Separate database/tables/indices over different partitions
  - Horizontal partitioning: every partition holds a subset of the *tuples*

    E.g., partition fact table by *month*
  - Vertical partitioning: every partition holds a subset of the *attributes*

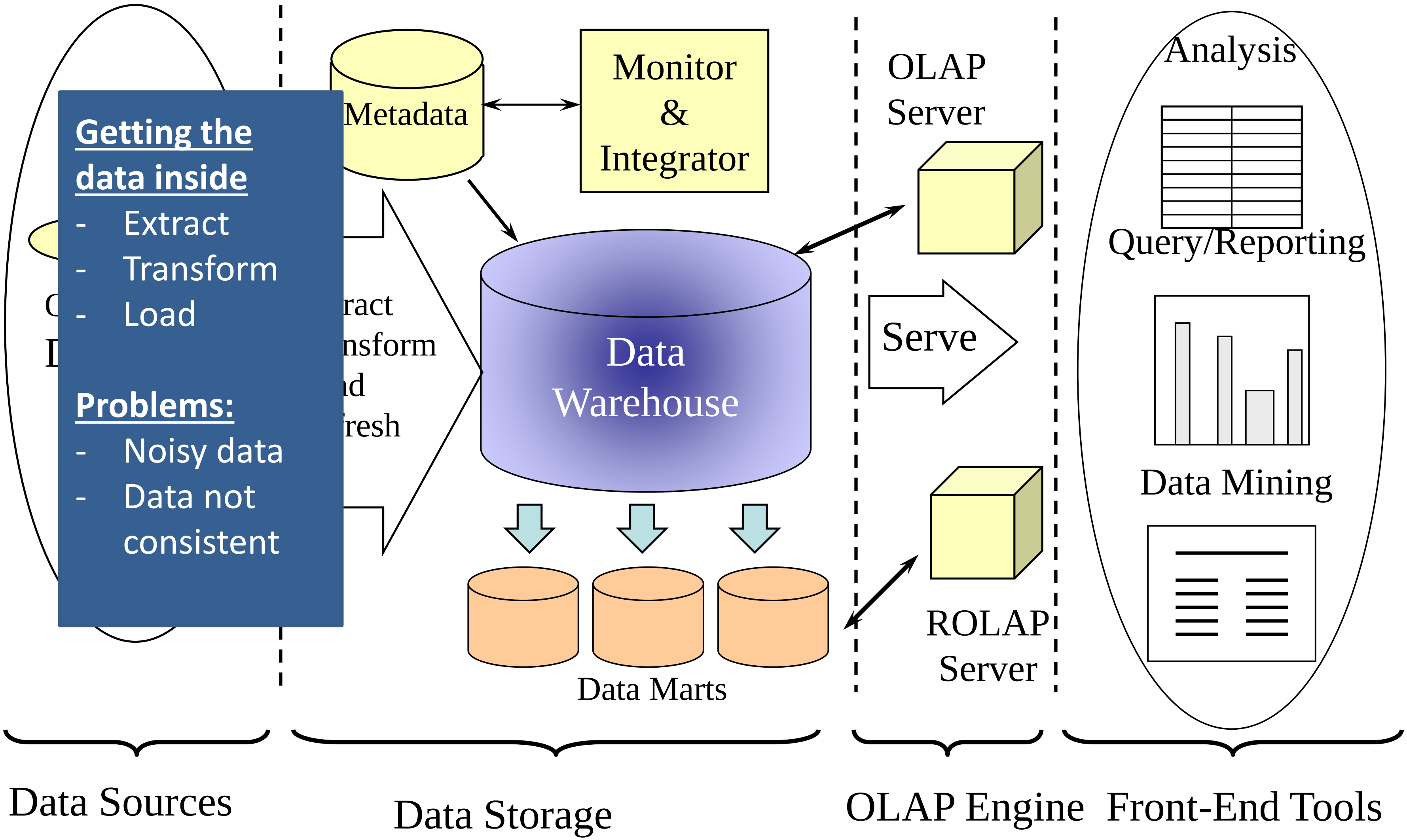
Three-Tier Architecture

Getting the data inside

- Extract
- Transform
- Load

Problems:

- Noisy data
- Data not consistent



ETL

- Extract – Transform – Load
- Many existing tools
  - Data Stage
  - Informatica
- Importance of metadata
  - Which reports cannot be trusted?
  - Impact analysis
  - Data lineage

IBM invited lecture on DataStage

ETL

- Important step in transformation: linking different tables

- Often difficult
  - Different keys
  - Small variations/errors

Exam Question

Compute the edit distance between the following two strings:

"Mr Smyth" and "M.Smit"

		M	r		S	m	y	t	h
	0	1	2	3	4	5	6	7	8
M	1	0	1	2	3	4	5	6	7
.	2	1	1	2	3	4	5	6	7
S	3	2	2	2	2	3	4	5	6
m	4	3	3	3	3	2	3	4	5
i	5	4	4	4	4	3	3	4	5
t	6	5	5	5	5	4	4	3	4

Load

- Bulk-loading data
- Typically rebuild (hard-to-update) indices

- Computing pre-aggregations
  - Sort-based
  - Hash-based

Sort-Based Aggregation

A	B	C	count
1	5	6	8
2	1	4	9
1	8	6	10
1	5	6	6
3	3	3	5
2	1	4	8

SELECT A, B, C, sum(count)
FROM R
GROUP BY A, B, C;

Sort-Based Aggregation

A	B	C	count

SORT

SELECT A, B, C, sum(count)
FROM R
GROUP BY A, B, C;

Sort-Based Aggregation

A	B	C	count
1	5	6	8
1	5	6	6
1	8	6	10
2	1	4	8
2	1	4	9
3	3	3	5

SCAN

SELECT A, B, C, sum(count)
FROM R
GROUP BY A, B, C;

Sort-Based Aggregation

A	B	C	count
1	5	6	8
1	5	6	6
1	8	6	10
2	1	4	8
2	1	4	9
3	3	3	5

SCAN

SELECT A, B, C, sum(count)
FROM R
GROUP BY A, B, C;

A	B	C	SUM
1	5	6	8

Sort-Based Aggregation

A	B	C	count
1	5	6	8
1	5	6	6
1	8	6	10
2	1	4	8
2	1	4	9
3	3	3	5

SCAN

SELECT A, B, C, sum(count)
FROM R
GROUP BY A, B, C;

A	B	C	SUM
1	5	6	14

Sort-Based Aggregation

A	B	C	count
1	5	6	8
1	5	6	6
1	8	6	10
2	1	4	8
2	1	4	9
3	3	3	5

SCAN

SELECT A, B, C, sum(count)
FROM R
GROUP BY A, B, C;

A	B	C	SUM
1	5	6	14
1	8	6	10

Sort-Based Aggregation

A	B	C	count
1	5	6	8
1	5	6	6
1	8	6	10
2	1	4	8
2	1	4	9
3	3	3	5

SCAN

SELECT A, B, C, sum(count)
FROM R
GROUP BY A, B, C;

A	B	C	SUM
1	5	6	14
1	8	6	10
2	1	4	8

Sort-Based Aggregation

A	B	C	count
1	5	6	8
1	5	6	6
1	8	6	10
2	1	4	8
2	1	4	9
3	3	3	5

SCAN

SELECT A, B, C, sum(count)
FROM R
GROUP BY A, B, C;

A	B	C	SUM
1	5	6	14
1	8	6	10
2	1	4	17

Sort-Based Aggregation

A	B	C	count
1	5	6	8
1	5	6	6
1	8	6	10
2	1	4	8
2	1	4	9
3	3	3	5

SCAN

SELECT A, B, C, sum(count)
FROM R
GROUP BY A, B, C;

A	B	C	SUM
1	5	6	14
1	8	6	10
2	1	4	17
3	3	3	5

Sort-Based Aggregation

A	B	C	count
1	5	6	8
1	5	6	6
1	8	6	10
2	1	4	8
2	1	4	9
3	3	3	5

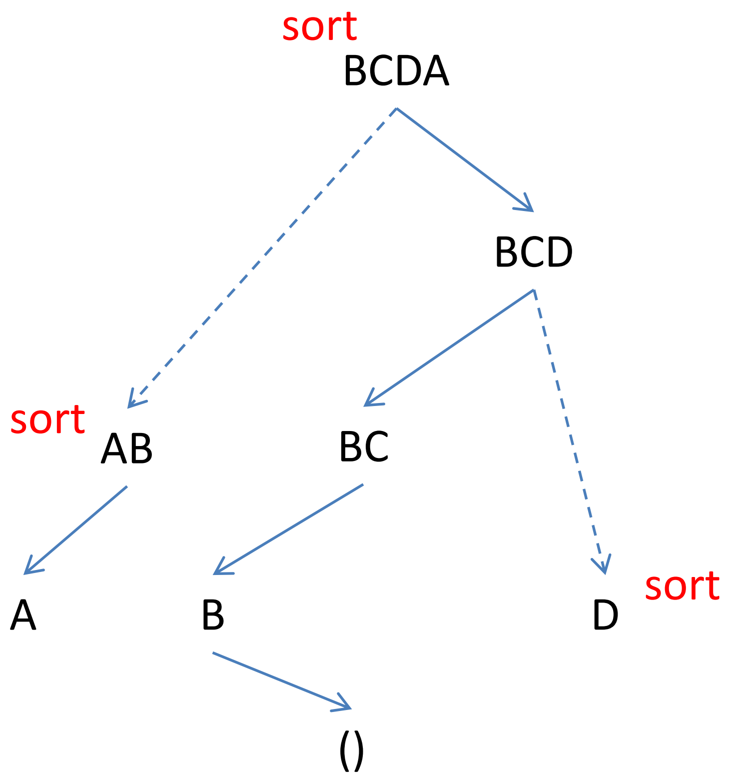
SCAN

SELECT A, B, C, sum(count)
FROM R
GROUP BY A, B, C;

A	B	C	SUM
1	5	6	14
1	8	6	10
2	1	4	17
3	3	3	5

Pipe-Sort

- Key problem: divide materialized views lattice into “pipes”, minimizing sorts



Hash-Based Aggregation

- If aggregated table fits into memory
  → Hash on grouping attributes, update measure
- Multiple hash tables fit together into the memory
  → Compute in one run
- Hash-based algorithm: selects optimal sets to be processed at the same time

Example Question

Suppose that we need to compute the aggregations Average and Min of attribute cost for the following groups of attributes:

AB, C, BC, ABC

Give an efficient way to do this, assuming none of the aggregated tables fits into memory.

Solution

- Average and Min:
  - Average is not distributive:

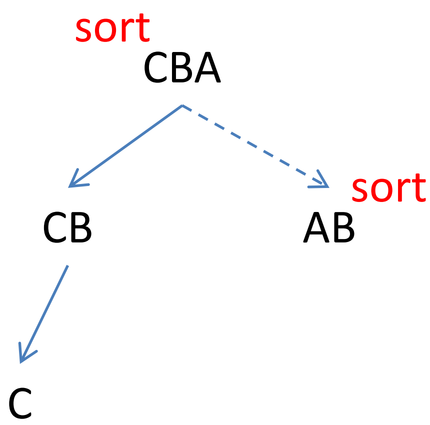
    $$\mathrm{AVG}(A \cup B) \neq \mathrm{AVG}(\{\mathrm{AVG}(A), \mathrm{AVG}(B)\})$$

    - AVG can be computed from SUM and COUNT
    - SUM and COUNT are distributive
  - Min is distributive:

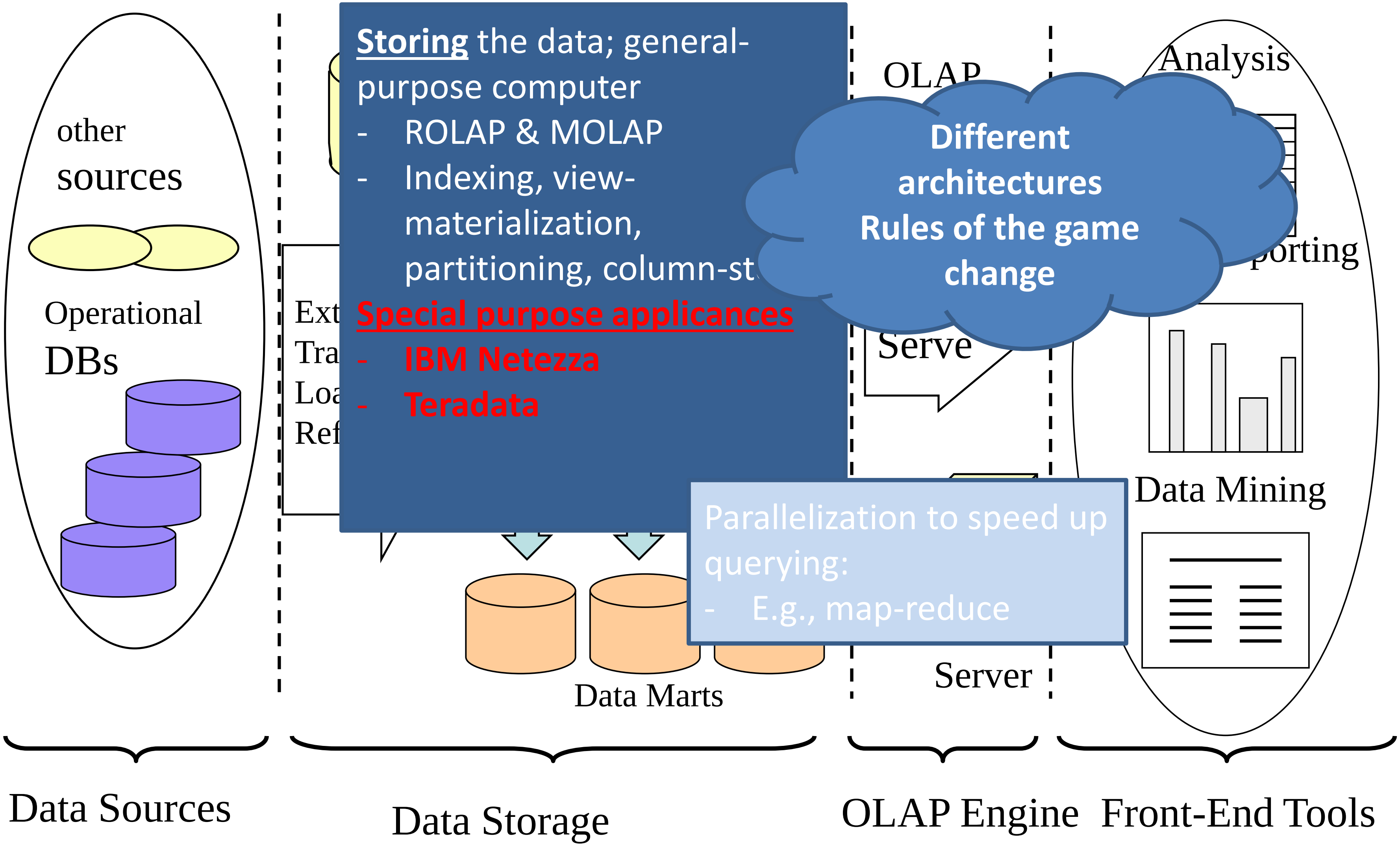
    $$\min(A \cup B) = \min(\{\min(A), \min(B)\})$$

(Why is it important that measures are distributive?)

Solution



Three-Tier Architecture



Storing the data; general-purpose computer
- ROLAP & MOLAP
- Indexing, view-materialization, partitioning, column-st

Special purpose applicances
- **IBM Netezza**
- **Teradata**

Different architectures
Rules of the game change

Parallelization to speed up querying:
- E.g., map-reduce

Different Architectures

- Problems:
  - Disk access is slow
  - Full table scan is faster than random read, but is slow if only part of the table is needed

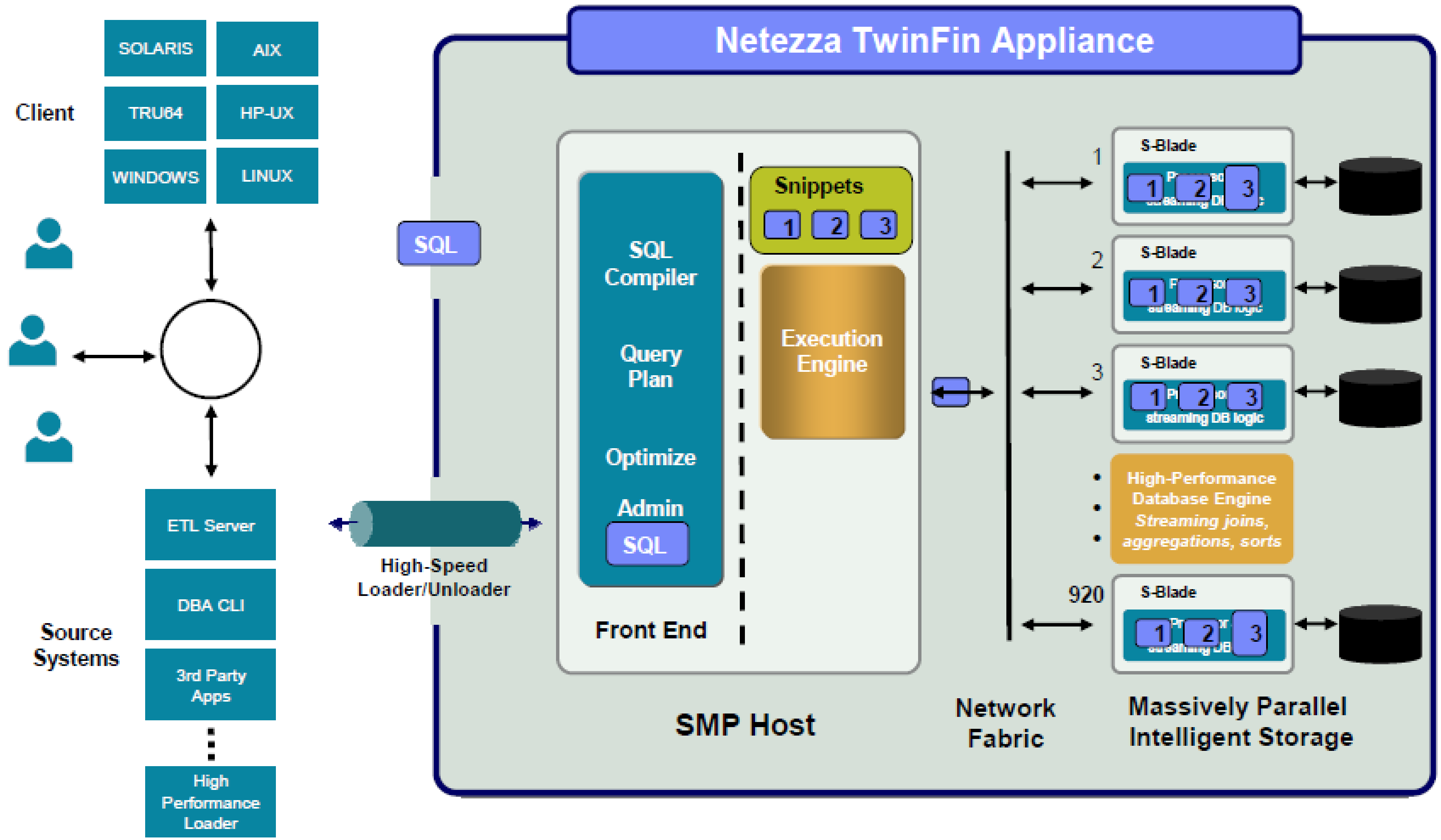
Move processor close to the data;
Compress data on disk = trade in slow I/O for fast processing
Multiple processors responsible for smaller part of the data

Implement select-project into the hardware
Zone-maps could be considered as a form of indexing
Vertical partitioning avoids access to attributes that are not needed

Obviously, query optimizer needs to be able to deal with new reality!

Slide taken from IBM presentation on Netezza

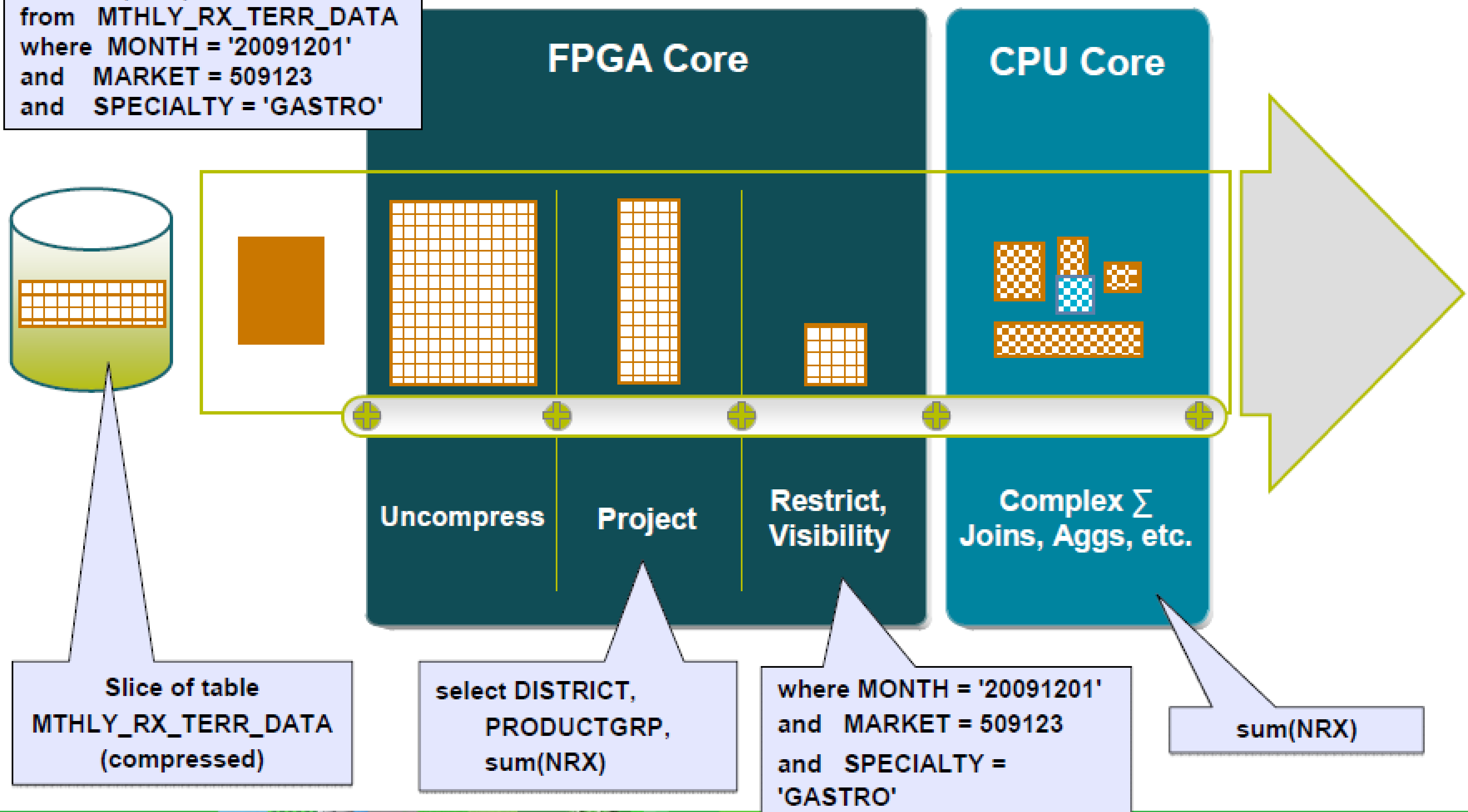
Asymmetric Massively Parallel Processing™



Slide taken from IBM presentation on Netezza

Our Secret Sauce

```
select DISTRICT,
       PRODUCTGRP,
       sum(NRX)
from   MTHLY_RX_TERR_DATA
where  MONTH = '20091201'
and    MARKET = 509123
and    SPECIALTY = 'GASTRO'
```



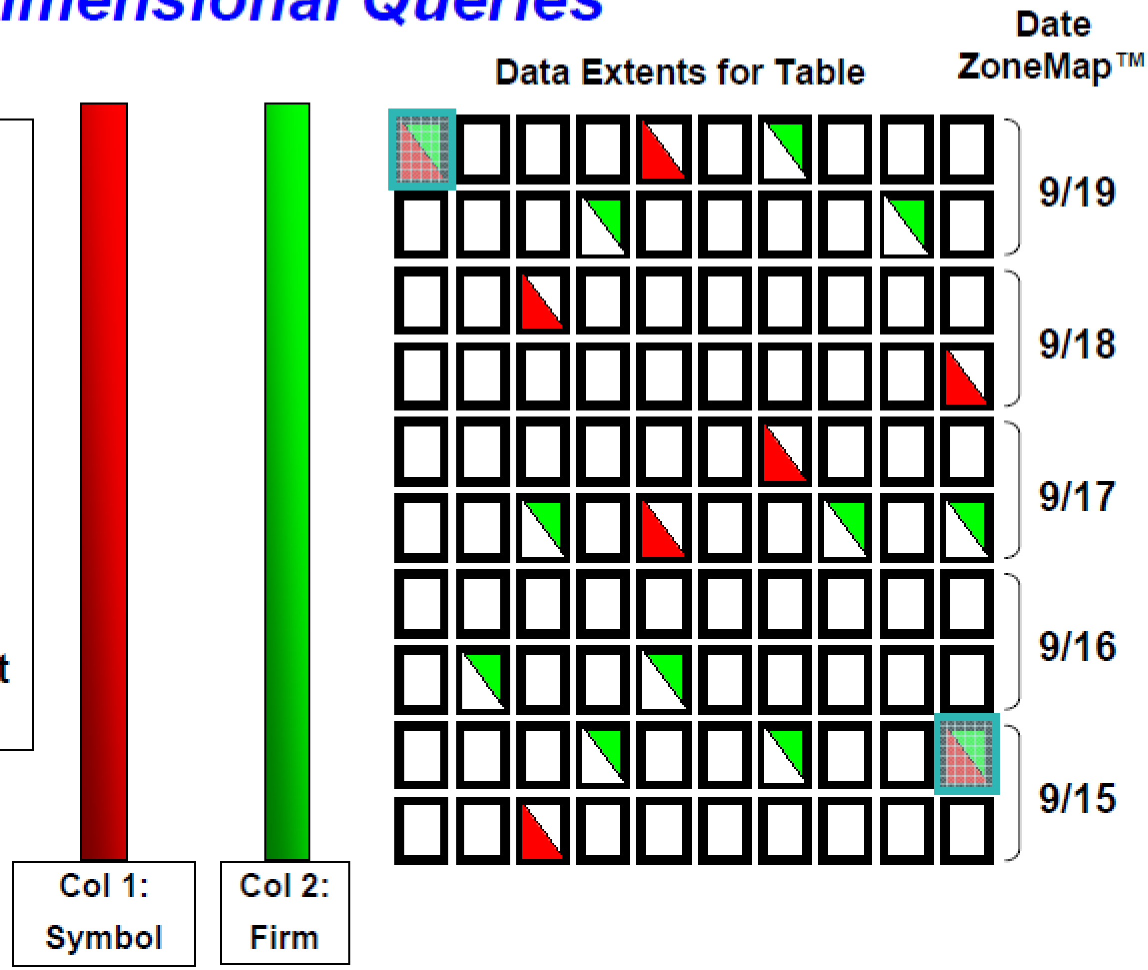
Slide taken from IBM presentation on Netezza

Clustered Base Tables

Accelerating Multi-Dimensional Queries

For Mid-September

- If Symbol = 'NZ' then scan 8 extents of 100 (8%)
- If Firm = 'Credit Suisse' then scan 12 extents of 100 (12%)
- If Symbol = 'NZ' And Firm = 'Credit Suisse' then scan just 2 of 100 extents (2%)



Create table

.....

Distribute on (....)

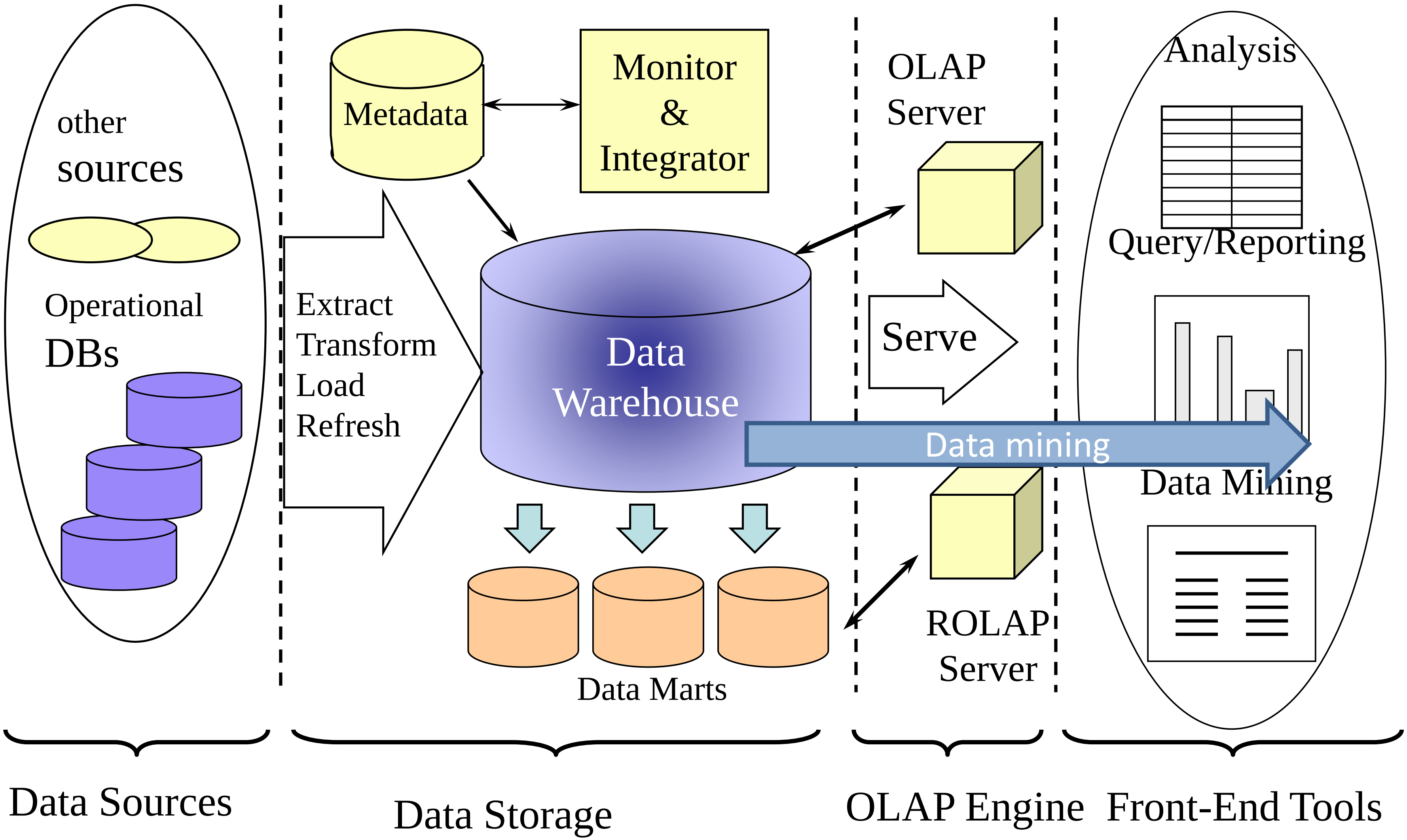
Organize on (....)

Different Architectures

- Challenges
  - Distribute data in an intelligent way
    - Hash-based; preferably on join-keys
  - In this way: truly distribute work

- What kills performance?
  - Excessive communication between nodes
    - E.g., poor distribution; non-selective self-joins

Three-Tier Architecture



Example Question

Suppose that the police force wants to develop a system to automatically monitor the Twitter stream in order to quickly identify potential outbreaks of violence (e.g., soccer hooligans gathering for a clash with the "enemy" or party visitors tweeting about a fight). Explain how data mining could be used to support this task.

Solution

- Spam-detection like system
  - Based on labeled examples, identify words in tweets that are correlated with this type of messages
  - Based upon propagation pattern
    - E.g., how often re-tweeted;
  - Based upon geography
    - co-locality with event
- Learn a classifier
  - Main difficulty: very unbalanced data

Conclusion

- Different ways to support data analysis
  - Traditional view
    - ETL;
    - ROLAP/MOLAP storage;
    - Logical optimizations:
      - materialized views
    - Physical optimizations:
      - indices; partitioning
    - OLAP/Data mining to do the analysis

Conclusion

– New hardware/appliances
    - Restrictions change
    - Multi processor
    - New game; different optimization strategies

Remember: 100 processors make a task at most 100 times faster; getting to this factor 100, however, is non-trivial!