Database Systems

Session 4 – Sub-Topic 1

Data Warehousing, Business Intelligence, and Data Lakes

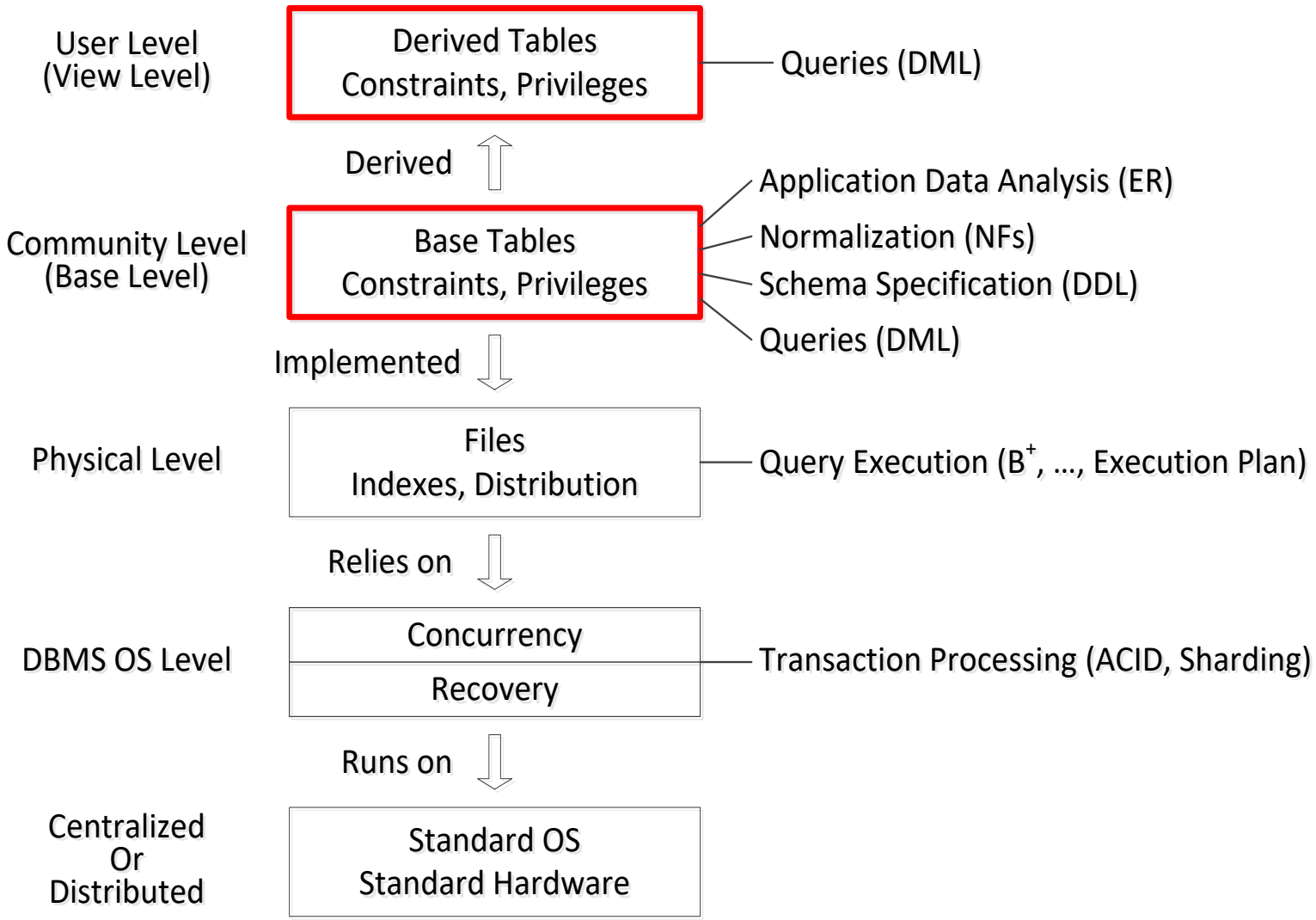
Dr. Jean-Claude Franchitti

New York University
Computer Science Department
Courant Institute of Mathematical Sciences

Agenda

- OLAP Basic Concepts
- Operational versus Tactical/Strategic Decision Making
- Data Warehouse Definition
- Data Warehouse Schemas
- The Extraction Transformation and Loading (ETL) Process
- Data Marts
- Virtual Data Warehouses and Virtual Data Marts
- Operational Data Store
- Data Warehouses versus Data Lakes
- Business intelligence
- Data Lakes and Data Swamps

OLAP in Context



OLAP vs. OLTP

- We have focused until now on ***OLTP: Online Transaction Processing***
- This dealt with storing data both logically and physically and managing transactions querying and modifying the data
- We will now focus providing support for analytical queries, essentially statistical and summary information for decision-makers, that is on ***OLAP: Online Analytical Processing***
- This may be accomplished by preprocessing, for efficiency purposes, and producing special types of views, which are also not necessarily up to date
  - » Not up to date may not be a problem in OLAP
- Data for OLAP (and more generally for data mining) is frequently stored in a ***Data Warehouse***

Key Ideas

- OLAP vs. OLTP
- Star schema
- Snowflake schema
- Cube
- Slicing and dicing
- Dimension hierarchies
- ROLAP
- MOLAP
- Oracle support

Example

- Our company has several stores and sells several products
- The stores are in different locations
- The locations, identified by (city,state) pairs are grouped into several regions
- We partition the times of sale into four quarters
- The quarters are grouped into two half-years

Our Company

Store	Store#	City	State	Region
	Alpha	New York	NY	NE
	Beta	Albany	NY	NE

Quarter	Quarter#	Half_Year
	1	First
	2	First
	3	Second
	4	Second

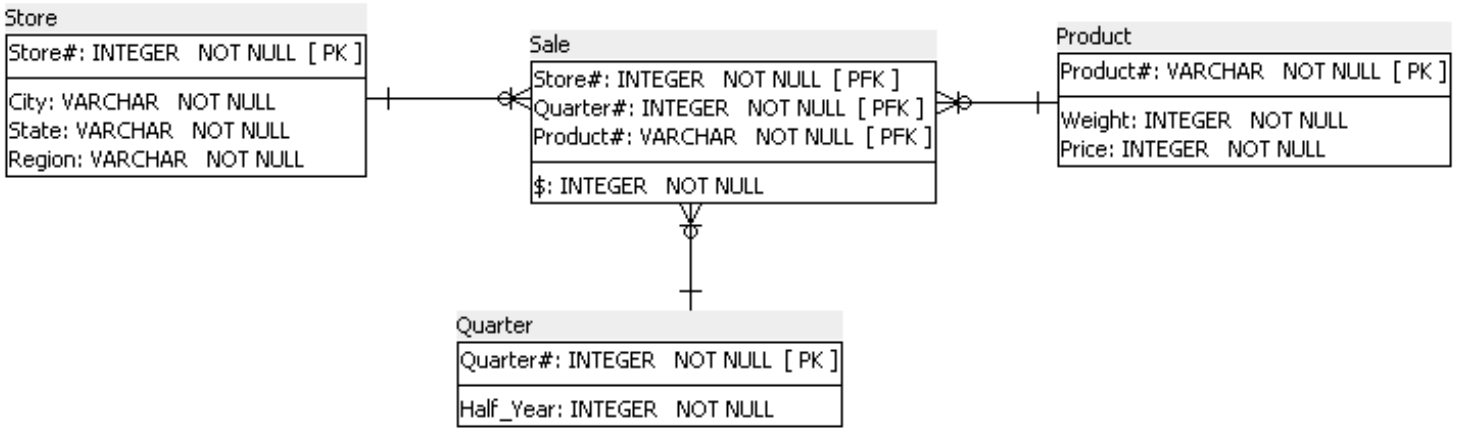
Product	Product#	Weight	Price
	Book	4	100
	Glass	15	200

Our Sales

Sale	Store#	Product#	Quarter#	$
	Alpha	Book	1	70,000
	Alpha	Glass	1	90,000
	Beta	Book	1	90,000
	Beta	Glass	1	80,000
	Alpha	Book	2	90,000
	Alpha	Glass	2	90,000
	Beta	Book	2	60,000
	Beta	Glass	2	50,000
	Alpha	Book	3	60,000
	Alpha	Glass	3	80,000
	Beta	Book	3	50,000
	Beta	Glass	3	90,000
	Alpha	Book	4	50,000
	Alpha	Glass	4	50,000
	Beta	Book	4	70,000
	Beta	Glass	4	70,000

Star Schema

- We want to support queries useful for statistical analysis by computing various sums, averages, etc.
- The structure we have is a ***star schema***
- In the middle we have a ***facts table***
- The other tables are ***dimensions*** or ***lookup tables***

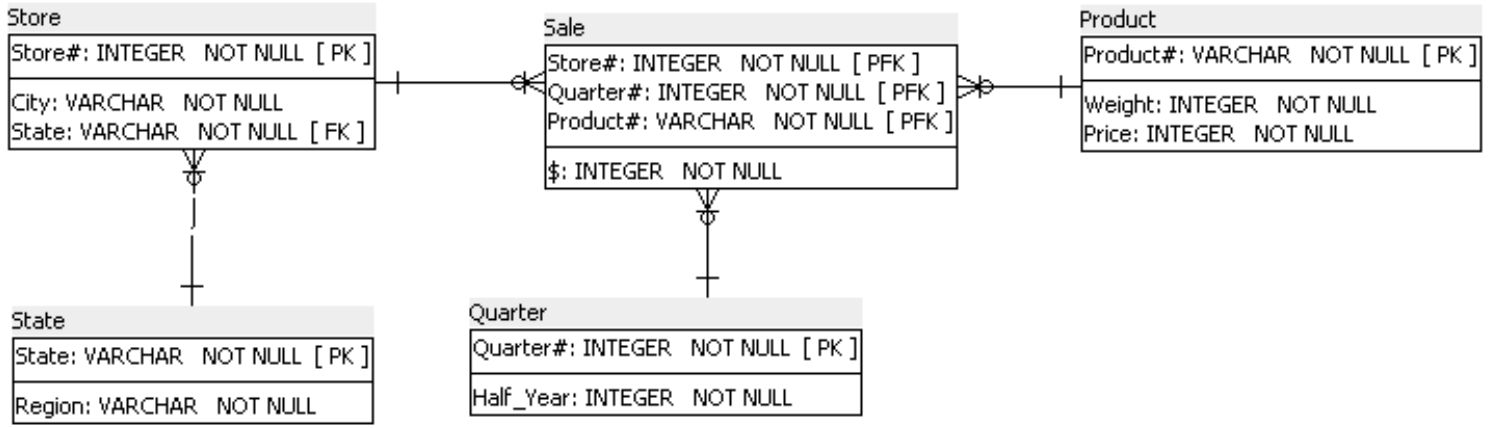


- This, of course is just a standard ternary relationship among 3 entity sets implemented in a relational database, but each community uses its own jargon, and some may not understand what entity sets and relationships are

Snowflake Schema: Normalized Star Schema

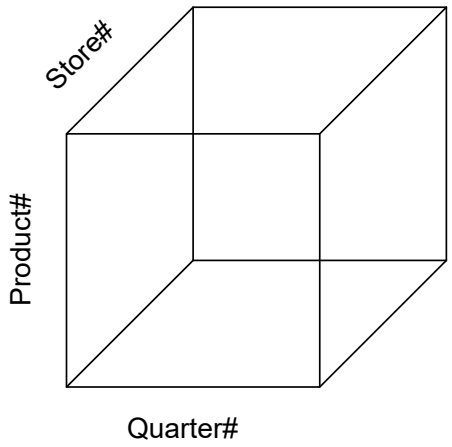
- One could also normalize, as table Store is not normalized, since State → Region

- Then, one could get, which we will not consider further, a ***snowflake schema***



Cube

- We could think of each row of fact table as occupying a voxel (volume element) in a ***cube***



- Cube, in general, can have any number of dimensions; in our example there are three

- This cube can then be ***sliced and diced***

Slice

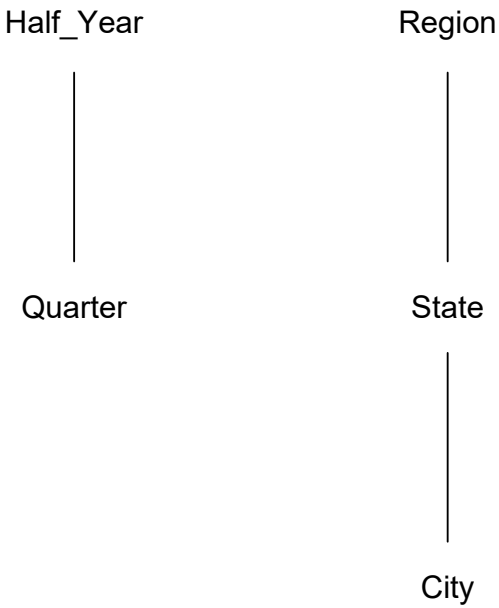
- SELECT Store#, Product#, SUM($)
  FROM Sale
  GROUP BY Store#, Product#

- We can do all kinds of such slices

- As you can see, this is a standard SQL statement but “visualized” differently

Dimension Hierarchies

- Dimensions could have hierarchies (or more generally even lattices)
  - To remind, when we looked for keys in PFDT database, the sets of attributes formed a lattice under the subset relation
- We have two very simple hierarchies
  - One temporal: quarters are in half years
  - One geographical: cities are in states are in regions



Using Hierarchies

- SELECT Sale.Product#, Quarter.Half_Year, SUM($)
  FROM Sale, Quarter
  WHERE Sale.Quarter# = Quarter.Quarter#
  GROUP BY Half_Year;

- Will produce summaries by half years, not quarters

- As you can see, this is a standard SQL statement but "visualized" differently

New Operator: CUBE

- SELECT Store#, Product#, SUM($)
  FROM Sale
  GROUP BY CUBE (Store#,Product#);

- Will produce all possible aggregations based on subsets of {Store#,Product#}, best explained by looking at what will come out

	Store#	Product#	$
	Alpha	Book	270,000
	Alpha	Glass	310,000
	Beta	Book	270,000
	Beta	Glass	290,000
	Alpha	NULL	580,000
	Beta	NULL	560,000
	NULL	Book	540,000
	NULL	Glass	600,000
	NULL	NULL	1,140,000

New Operator: ROLLUP

- ROLLUP produces only some of the aggregate operators produced by CUBE, best explained by example
- SELECT Store#, Product#, SUM($)
  FROM Sale
  GROUP BY ROLLUP (Store#,Product#);

	Store#	Product#	$
	Alpha	Book	270,000
	Alpha	Glass	310,000
	Beta	Book	270,000
	Beta	Glass	290,000
	Alpha	NULL	580,000
	Beta	NULL	560,000
	NULL	NULL	1,140,000

ROLAP and MOLAP

- ROLAP: Relational OLAP
- That is what we have been doing: OLAP information was stored as a set of star (or more generally snowflakes) schemas

- MOLAP: Multidimensional OLAP
- Information not stored as a relational database, but essentially as a cube

Oracle: Defining the Database

```
create table store(
  sid char(20) primary key,
  city char(20),
  state char(20),
  region char(20)
);

create table product(
  pid char(20) primary key,
  weight number,
  price number
);

create table quarter(
  qid number primary key,
  half_year char(10)
);
```

Oracle: Defining the Database

```
create table sale(
  sid char(20),
  pid char(20),
  qid number,
  profit number,
  primary key(qid, pid, sid),
  foreign key(qid) references quarter(qid),
  foreign key(pid) references product(pid),
  foreign key(sid) references store(sid)
);
```

Oracle: OLAP Query

```
select sid, pid, sum(profit)
from sale
group by rollup(sid, pid);

select sid, pid, sum(profit)
from sale
group by cube(sid, pid);
```

Decision Making Levels (1/2)

- Operational level
  - Day-to-day business decisions
  - Short time frame
  - On-Line Transaction Processing (OLTP)
- Tactical level
  - Middle management decisions
  - Medium-term focus
- Strategic level
  - Senior management
  - Long-term focus

Decision Making Levels (2/2)

- Operational systems
  - Operational level
  - Focus on simple INSERT, UPDATE, DELETE and/or SELECT statements
  - Transaction throughput
- Decision Support Systems
  - Tactical + strategic level
  - Focus on data retrieval by answering complex ad-hoc queries (SELECT statements)
  - Represent data in a multidimensional way
  - Trend analysis

Data Warehouse Definition (1/3)

- *"A data warehouse is a subject-oriented, integrated, time-variant, and nonvolatile collection of data in support of management's decision-making process."* (Inmon, 1996)
- Subject oriented
  - organized around subjects such as customers, products, sales
- Integrated
  - integrates data from a variety of operational sources and a variety of formats

Data Warehouse Definition (2/3)

- Non-volatile
  - data is primarily read-only
  - data loading and data retrieval
- Time variant
  - time series of periodic snapshots

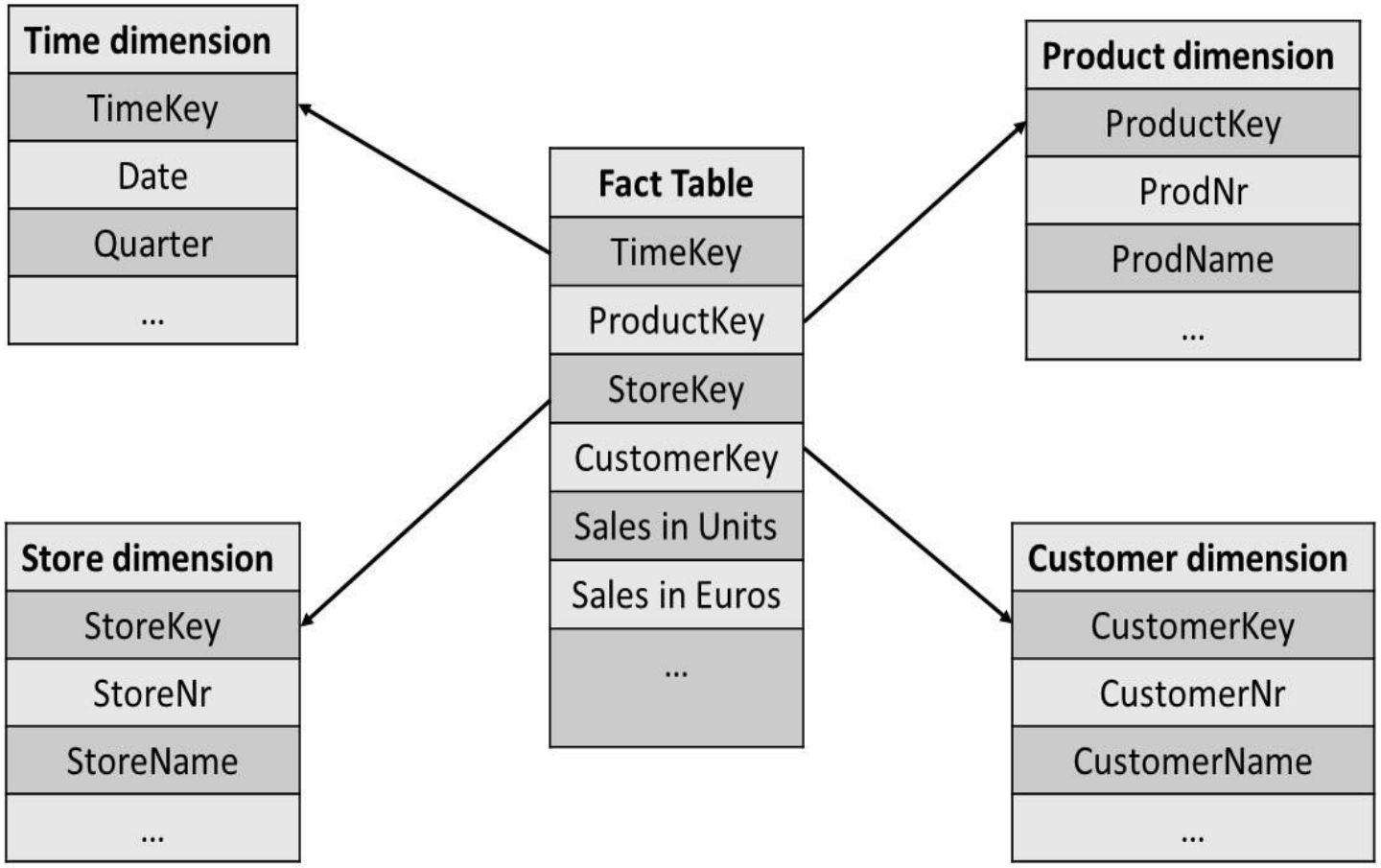
Data Warehouse Definition (3/3)

	Transactional system	Data Warehouse
Usage	Day to day business operations	Decision support at tactical/strategic level
Data latency	Real-time data	Periodic snapshots, incl. historical data
Design	Application oriented	Subject Oriented
Normalization	Normalized data	(Sometimes also) denormalized data
Data manipulation	Insert/Update/Delete/Select	Insert/Select
Transaction management	Important	Less of a concern
Type of queries	Many, simple queries	Fewer, but complex and ad-hoc queries

Data Warehouse Schemas

- Star schema
- Snowflake schema
- Fact constellation
- Specific Schema issues

Star Schema (1/3)



Star Schema (2/3)

- Fact table
  - one tuple per transaction or event (i.e., a fact) and also measurement data (e.g. sales)
- Dimension table
  - further information about each of the facts in the fact table (e.g., Time, Store, Customer, Product).
  - criteria for aggregating the measurement data
  - often denormalized

Star Schema (3/3)

Fact table

TimeKey	ProductKey	StoreKey	CustomerKey	Sales in Units	Sales in Euros
200	50	100	20006010	6	167.94
210	25	130	20006012	3	54
180	30	150	20006008	12	384
...					

Dimension tables

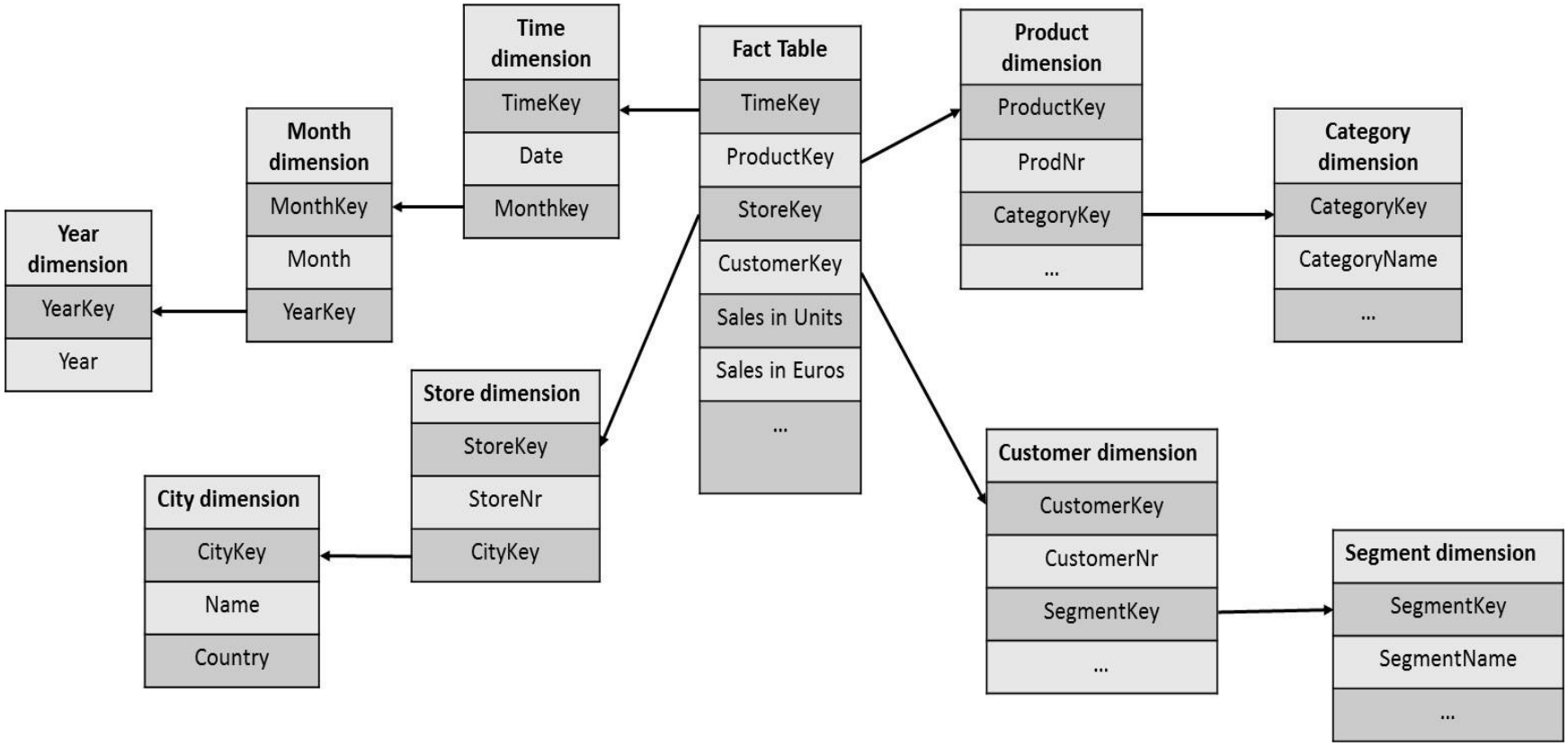
TimeKey	Date	Quarter	...
200	08/03/2017	1	...
210	09/11/2017	3	
180	27/02/2017	1	

CustomerKey	CustomerNr	CustName	...
20006008	20	Bart Baesens	...
20006010	10	Wilfried Lemahieu	
20006012	5	Seppe vanden Broucke	

StoreKey	StoreNr	StoreName	...
100	68	The Wine Depot	...
130	94	The Wine Crate	
150	69	Vinos del Mundo	

ProductKey	ProdNr	ProdName	...
25	0178	Meerdael, Methode Traditionnelle Chardonnay, 2014	...
30	0199	Jacques Selosse, Brut Initial, 2012	
50	0212	Billecart-Salmon, Brut Réserve, 2014	

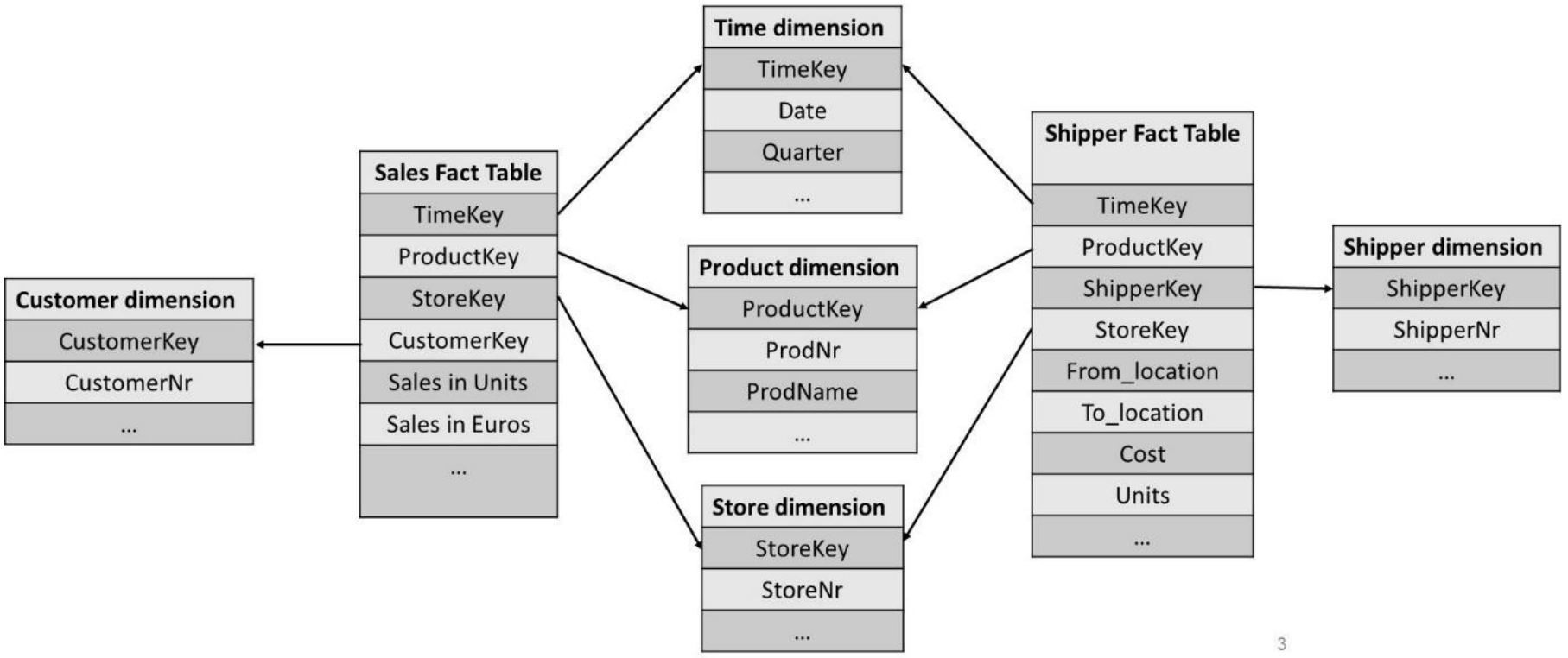
Snowflake Schema (1/2)



Snowflake Schema (2/2)

- In case dimension tables grow too large
- In case most queries do not make use of the outer level dimension tables

Fact Constellation



3

Specific Schema Issues

- Surrogate keys
- Granularity of the fact table
- Factless Fact Tables
- Optimizing the dimension tables
- Defining junk dimensions
- Defining outrigger tables
- Slowly changing dimensions
- Rapidly changing dimensions

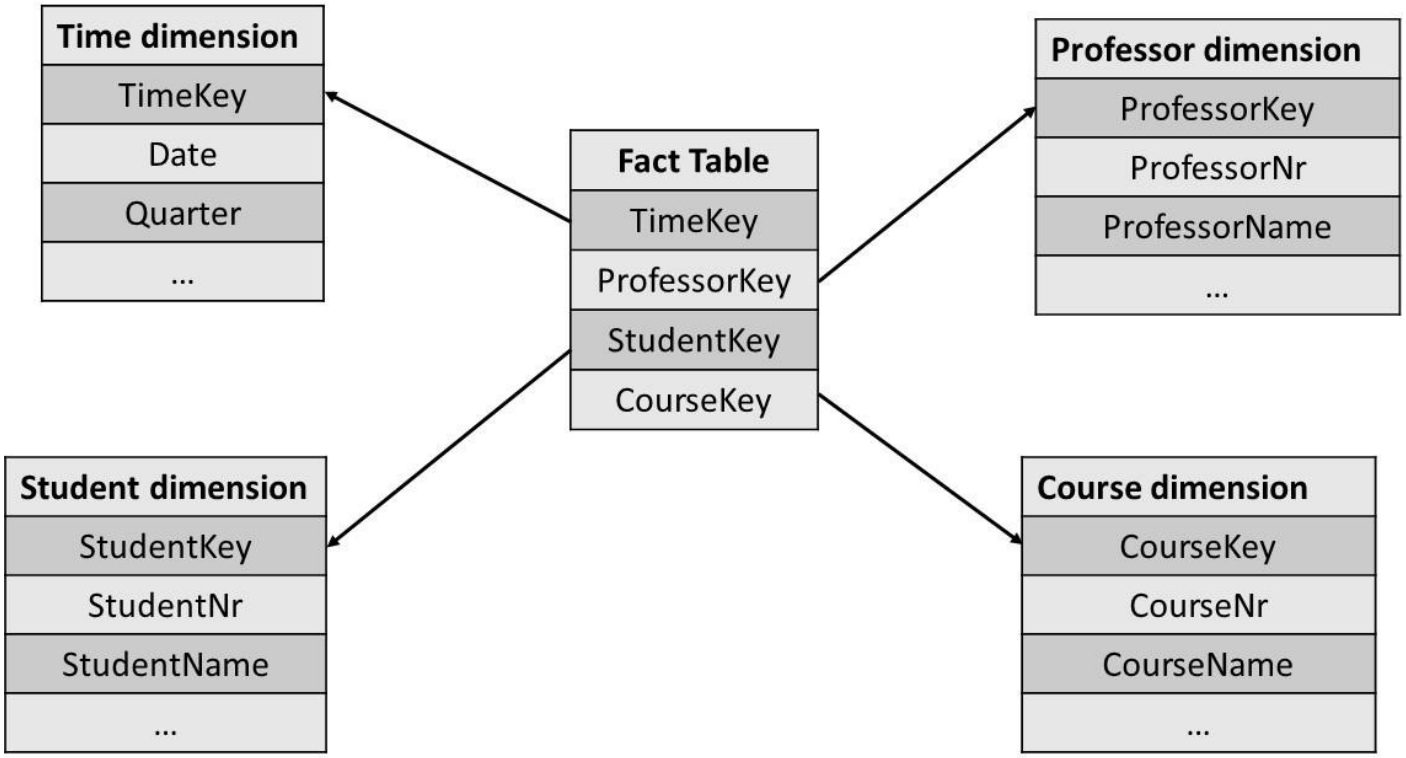
Surrogate Keys

- StoreKey, ProductKey, ShipperKey, …
- Meaningless integers
- Cannot use business keys since they usually have a business meaning
- Surrogate keys essentially buffer the data warehouse from the operational environment
- Business keys are usually bigger in size
- Business keys are also often re-used over longer periods of time

Granularity of the Fact Table

- Level of detail of one row of the fact able
- Higher (lower) granularity implies more (fewer) rows
- Trade-off between level of detailed analysis and storage requirements
- Examples:
  - One tuple of the fact table corresponds to one line on a purchase order
  - One tuple of the fact table corresponds to one purchase order
  - One tuple of the fact able corresponds to all purchase orders made by a customer

Factless Fact Tables



Optimizing the dimension tables

- Dimension tables should be heavily indexed to improve query execution time
- On average between 5 and 10
- E.g. Time
  - TimeKey, Date, DayOfWeek, DayOfMonth, DayOfYear, Month, MonthName, Year, LastDayInWeekFlag, LastDayInMonthFlag, FiscalWeek, HolidayFlag, …

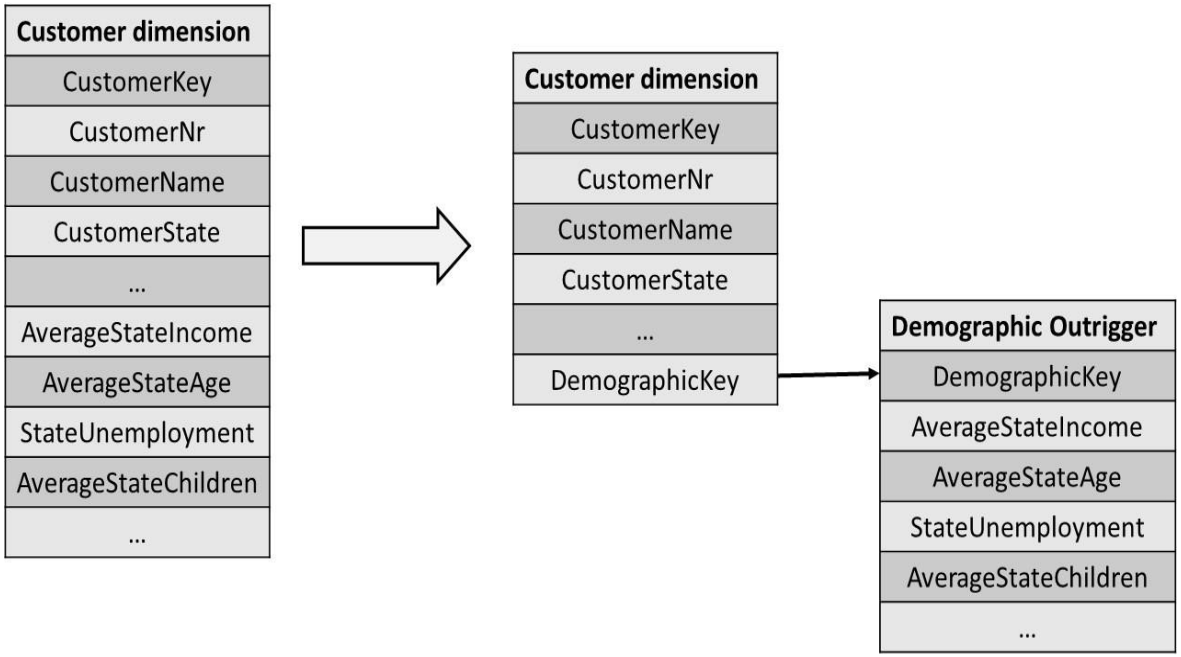
Junk Dimensions

- Deal with low cardinality attribute types such as flags or indicators
- Example: On-line Purchase (Y/N), Payment (cash or credit-card), Discount (Y/N)
- Junk dimension is a dimension that simply enumerates all feasible combinations of values of the low cardinality attribute types

JunkKey1	On-line purchase	Payment	Discount
1	Yes	Credit-card	Yes
2	Yes	Credit-card	No
3	No	Credit-card	Yes
4	No	Credit-card	No
5	No	Cash	Yes
6	No	Cash	No

Outrigger tables

- store a set of attribute types of a dimension table which are highly correlated, low in cardinality and updated simultaneously



Slowly Changing Dimensions (1/5)

- Dimensions that change slowly and irregularly over a period of time
- Example: customer segment ranging from AAA, AA, A, BBB, … to C, determined on a yearly basis

- **Approach 1**

OLD STATUS

CustomerKey	CustomerNr	CustomerName	Segment
123456	ABC123	Bart Baesens	AA

NEW STATUS

CustomerKey	CustomerNr	CustomerName	Segment
123456	ABC123	Bart Baesens	AAA

- **Approach 2**

OLD STATUS

CustomerKey	CustomerNr	CustomerName	Segment	Start_Date	End_Date	Current_Flag
123456	ABC123	Bart Baesens	AA	27-02-2014	31-12-9999	Y

NEW STATUS

CustomerKey	CustomerNr	CustomerName	Segment	Start_Date	End_Date	Current_Flag
123456	ABC123	Bart Baesens	AA	27-02-2014	27-02-2015	N
123457	ABC123	Bart Baesens	AAA	28-02-2015	31-12-9999	Y

- **Approach 3**

OLD STATUS

CustomerKey	CustomerNr	CustomerName	Segment
123456	ABC123	Bart Baesens	AA

NEW STATUS

CustomerKey	CustomerNr	CustomerName	Old Segment	New Segment
123456	ABC123	Bart Baesens	AA	AAA

- **Approach 4**

OLD STATUS

CustomerKey	CustomerNr	CustomerName	Segment
123457	ABC123	Bart Baesens	AAA

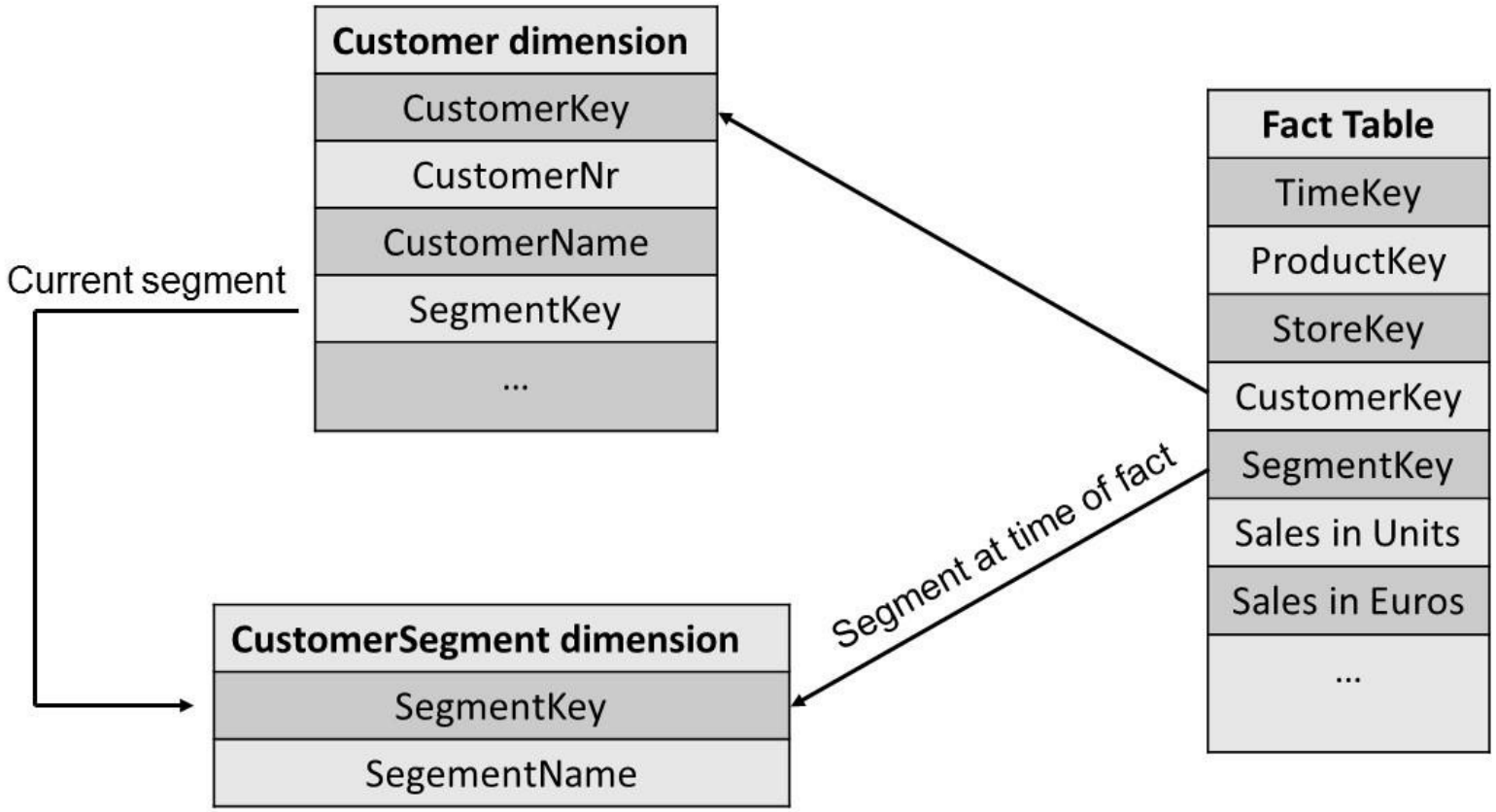
NEW STATUS

CustomerKey	CustomerNr	CustomerName	Segment	Start_Date	End_Date
123456	ABC123	Bart Baesens	AA	27-02-2014	27-02-2015
123457	ABC123	Bart Baesens	AAA	28-02-2015	31-12-9999

Rapidly changing dimensions (1/4)

- Dimensions that change rapidly and regularly over a period of time
- Example: customer segment ranging from AAA, AA, A, BBB, … to C, determined on a daily basis
- Approaches 2 and 4 discussed in the previous section will result into a lot of rows
- Split customer information into stable (e.g., gender, marital status, …) and rapidly changing information which is put in mini-dimension table

Rapidly changing dimensions (2/4)



Rapidly changing dimensions (3/4)

Fact table

TimeKey	ProductKey	StoreKey	CustomerKey	SegmentKey	Sales in Units	Sales in Euros
200	50	100	1200	1	6	167.94
210	25	130	1400	4	3	54
180	30	150	1000	3	12	384
…						

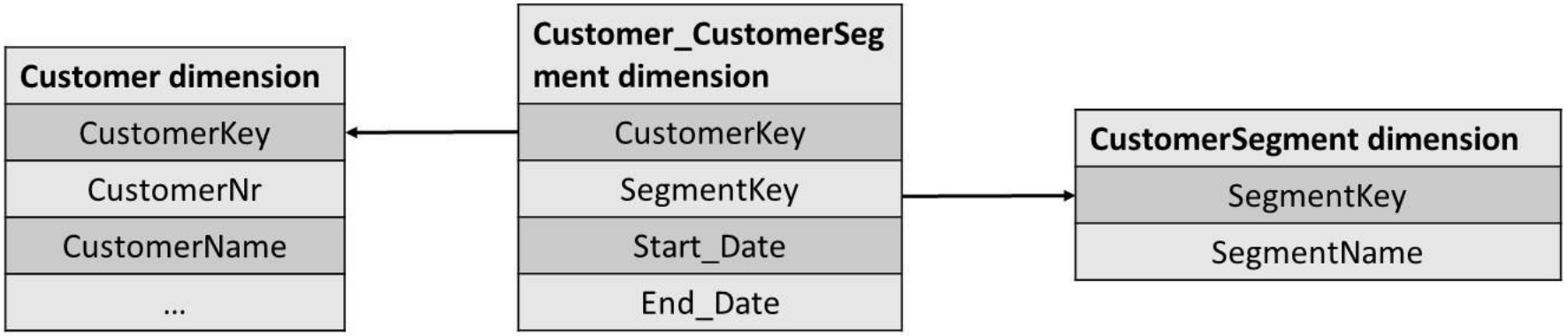
Customer Dimension

CustomerKey	CustomerNr	CustName	SegmentKey
1000	20	Bart Baesens	2
1200	10	Wilfried Lemahieu	1
1400	5	Seppe vanden Broucke	1

CustomerSegment Dimension

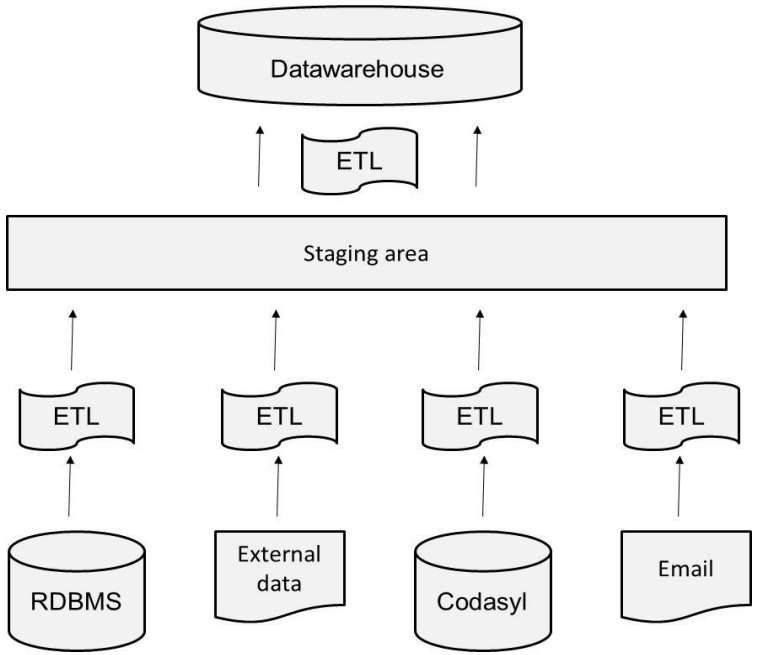
SegmentKey	SegmentName
1	A
2	B
3	C
4	D

Rapidly changing dimensions (4/4)



The Extraction Transformation and Loading (ETL) Process (1/2)

- can consume up to 80% of all efforts needed to set up a data warehouse
- dump the data in a so-called staging area where all the ETL activities can be executed



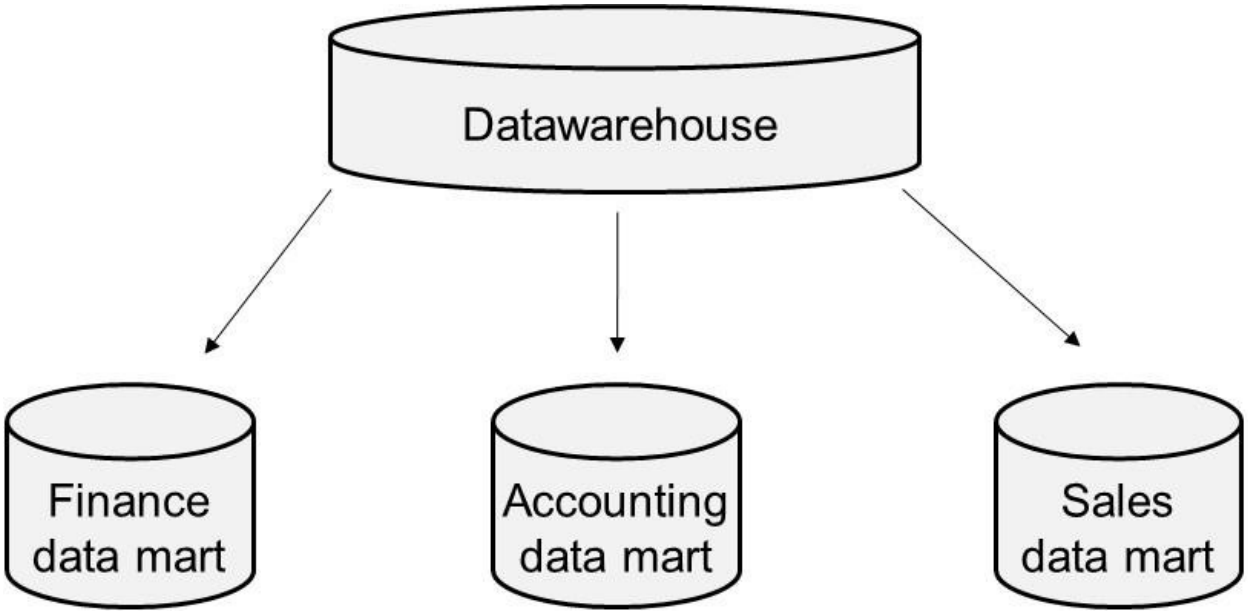
The Extraction Transformation and Loading (ETL) Process (2/2)

- Extraction
  - Full or incremental (Changed Data Capture)
- Transformation
  - formatting
  - cleansing
  - aggregation and merging
  - enrichment
- Loading
  - Fill the fact and dimension tables
- Documentation and metadata

Data Marts (1/3)

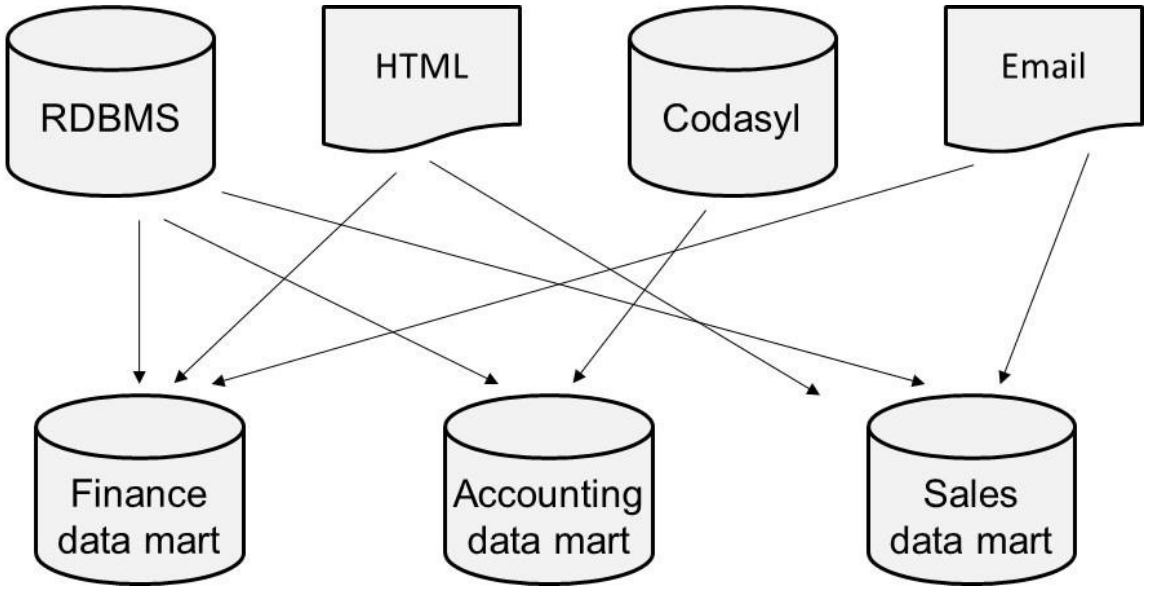
- A **data mart** is a scaled down version of a data warehouse aimed at meeting the information needs of a homogeneous small group of end-users such as a department or business unit
- Provide focused content + improve query performance

- **Dependent data marts** pull their data directly from a central data warehouse



Data Marts (3/3)

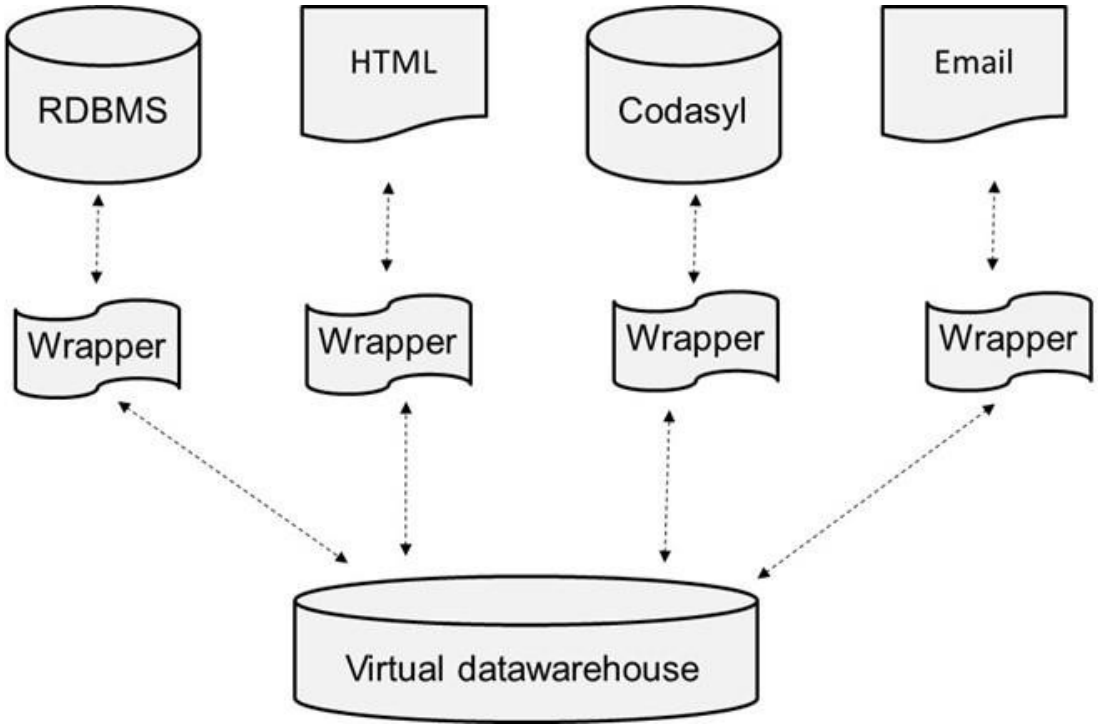
- **Independent data marts** are standalone systems drawing data directly from the operational systems, external sources or a combination of both



Virtual Data Warehouses and Virtual Data Marts (1/5)

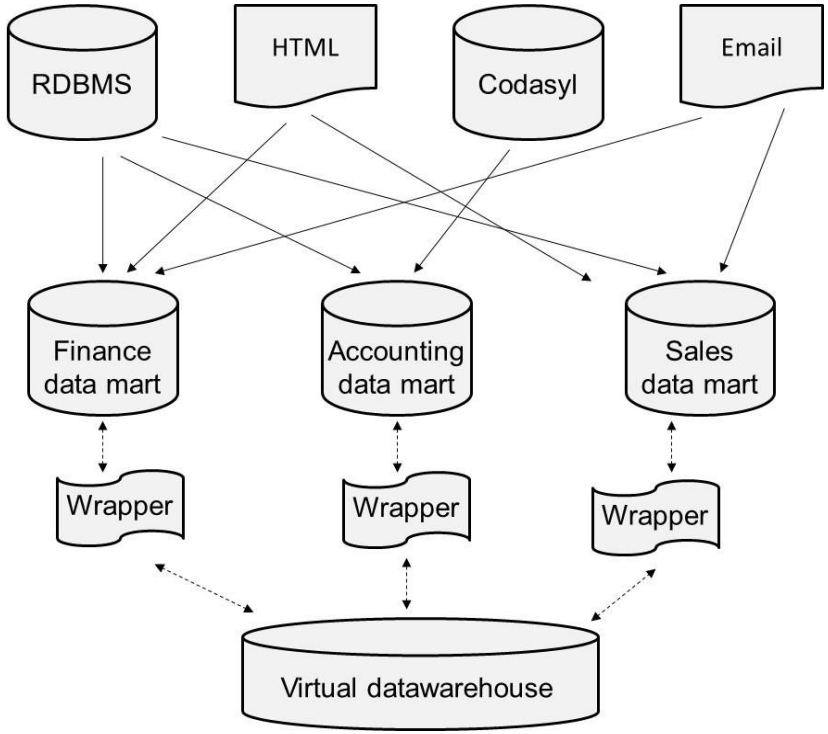
- Idea of virtualization is to use middleware to create a logical or **virtual data warehouse** or **virtual data mart** which has no physical data but provides a single point of access to a set of underlying physical data stores.
- Data is only accessed ('pulled') at query time.

- Virtual Data warehouse can be built as a set of SQL views directly on the underlying operational data sources



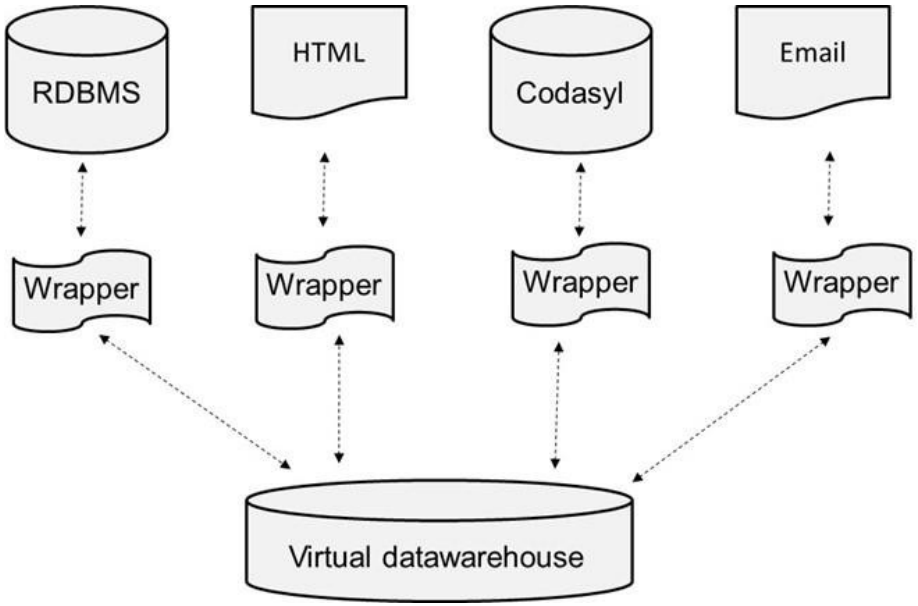
Virtual Data Warehouses and Virtual Data Marts (3/5)

- Virtual data warehouse can be built as an extra layer on top of a collection of physical independent data marts



Virtual Data Warehouses and Virtual Data Marts (4/5)

- Metadata model contains the schema mappings between the schemas of the underlying data stores and the schema of the virtual data warehouse
- Query reformulation



Virtual Data Warehouses and Virtual Data Marts (5/5)

- A virtual data mart is usually defined as a single SQL view
- Virtual independent versus Virtual dependent data mart
- Disadvantages
  - Extra processing capacity from the underlying (operational) data sources
  - Not possible to keep track of historical data

Operational Data Store

- Operational Data Store (ODS) can be considered as a staging area that provides query facilities
- Good for analysis tools that need data that is closer to real time
- More complex analyses are still conducted on the actual data warehouse

Data Warehouses versus Data Lakes

	Data Warehouse	Data lake
Data	Structured	Often unstructured
Processing	Schema-on-write	Schema-on-read
Storage	Expensive	Low cost
Transformation	Before entering the DW	Before analysis
Agility	Low	High
Security	Mature	Maturing
Users	Decision makers	Data Scientists

- **Business intelligence (BI)** is referred to as the set of activities, techniques and tools aimed at understanding patterns in past data and predicting the future.
- Garbage In, Garbage Out (GIGO)
- BI techniques
  - Query and Reporting
  - Pivot tables
  - OLAP

- Business user can graphically and interactively design a query and corresponding report
- Self Service BI
- Query by Example (QBE)
  - a query is composed in a user-friendly and visual way
- Report can be refreshed at any time
- Innovative visualization techniques

- A **pivot or cross-table** is a popular data summarization tool. It essentially cross-tabulates a set of dimensions in such a way that multidimensional data can be represented in a two-dimensional tabular format.

Sales		**Quarter**				**Total**
		Q1	Q2	Q3	Q4	
Region	Europe	100	200	50	100	450
	Africa	50	100	200	50	400
	Asia	20	50	10	150	230
	America	50	10	100	100	260
	Total	220	360	360	400	1340

On-Line Analytical Processing (OLAP)

- OLAP allows you interactively analyze the data, summarize it and visualize it in various ways
- Provide the business-user with a powerful tool for ad-hoc querying
- Types
  - MOLAP
  - ROLAP
  - HOLAP

MOLAP (1/2)

- Multidimensional OLAP (MOLAP) stores the multidimensional data using a Multidimensional DBMS (MDBMS) whereby the data is stored in a multi-dimensional array-based data structure optimized for efficient storage and quick access.

Array (key, value)	Q1	Q2	Q3	Q4	Total
Product A	(1,1) 10	(1,2) 20	(1,3) 40	(1,4) 10	(1,5) 80
Product B	(2,1) 20	(2,2) 40	(2,3) 10	(2,4) 30	(2,5) 100
Product C	(3,1) 50	(3,2) 20	(3,3) 40	(3,4) 30	(3,5) 140
Product D	(4,1) 10	(4,2) 30	(4,3) 20	(4,4) 20	(4,5) 80
Total	(5,1) 90	(5,2) 110	(5,3) 110	(5,4) 90	(5,5) 400

MOLAP (2/2)

- Fast in terms of data retrieval
- More storage space needed
- Scales poorly when the number of dimensions increases
- No universal SQL-like standard is provided
- Not optimized for transaction processing

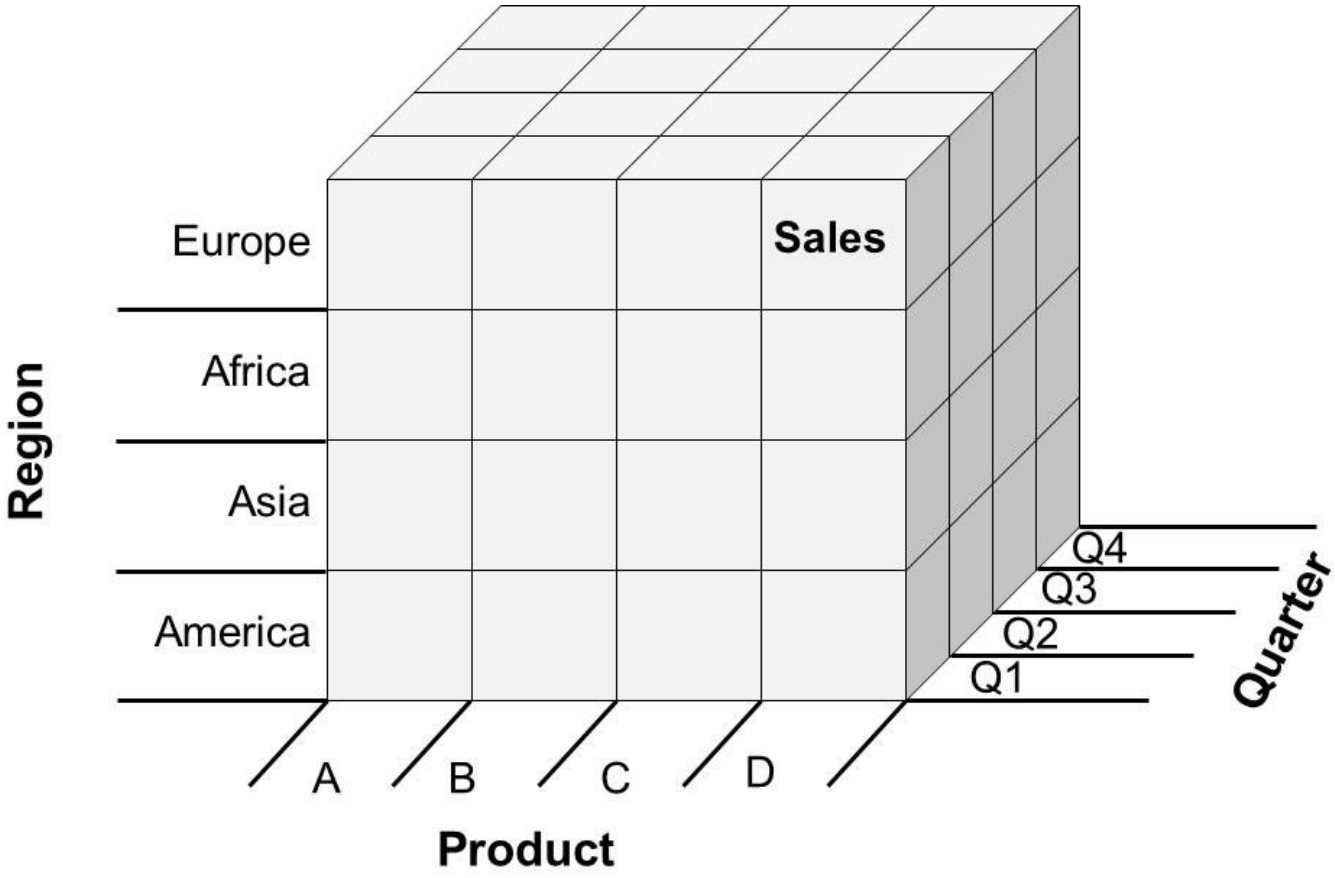
ROLAP

- **Relational OLAP (ROLAP)** stores the data in a relational data warehouse, which can be implemented using a star, snowflake or fact constellation schema
- ROLAP scales better to more dimensions than MOLAP
- ROLAP query performance may however be inferior to MOLAP

HOLAP

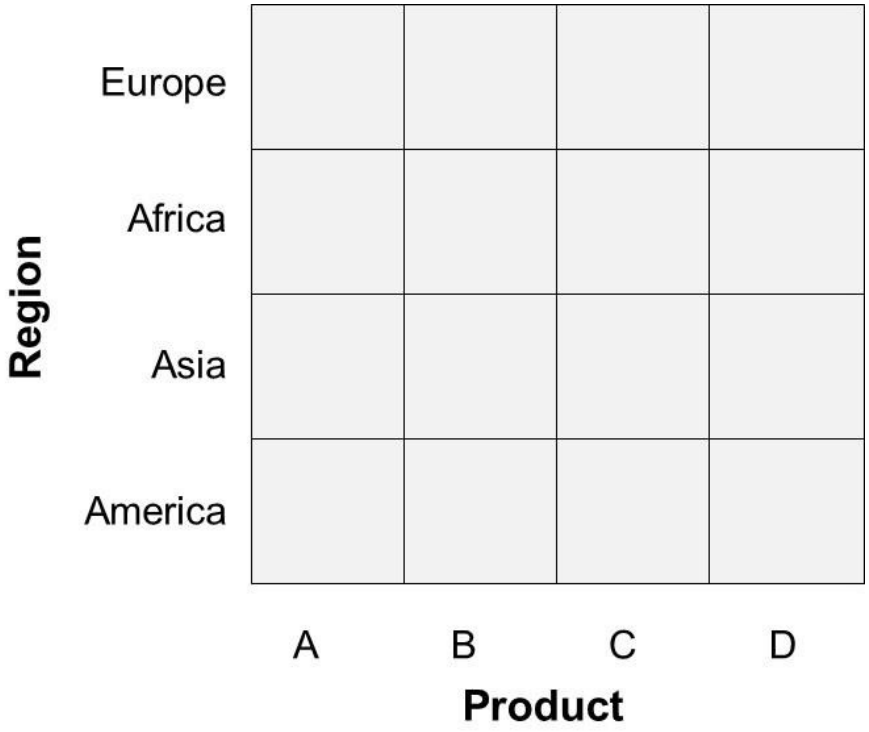
- **Hybrid OLAP (HOLAP)** tries to combine the best of both MOLAP and ROLAP
- RDBMS used to store the detailed data in a relational data warehouse whereas the pre-computed aggregated data is kept as a multidimensional array managed by an MDBMS
- OLAP analysis first starts from the multidimensional database
- Combine the performance of MOLAP with the scalability of ROLAP

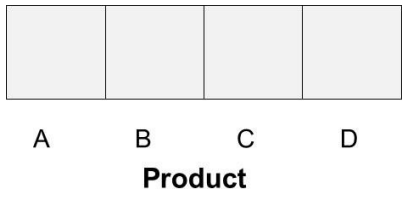
OLAP Operators (1/4)



OLAP Operators (2/4)

- Roll-up refers to aggregating the current set of fact values within or across one or more dimensions
- Hierarchical versus dimensional roll-up

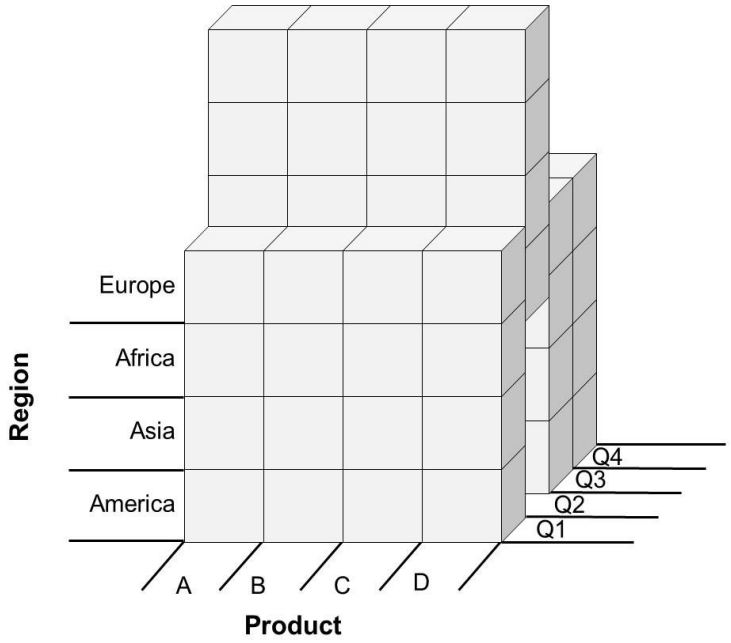




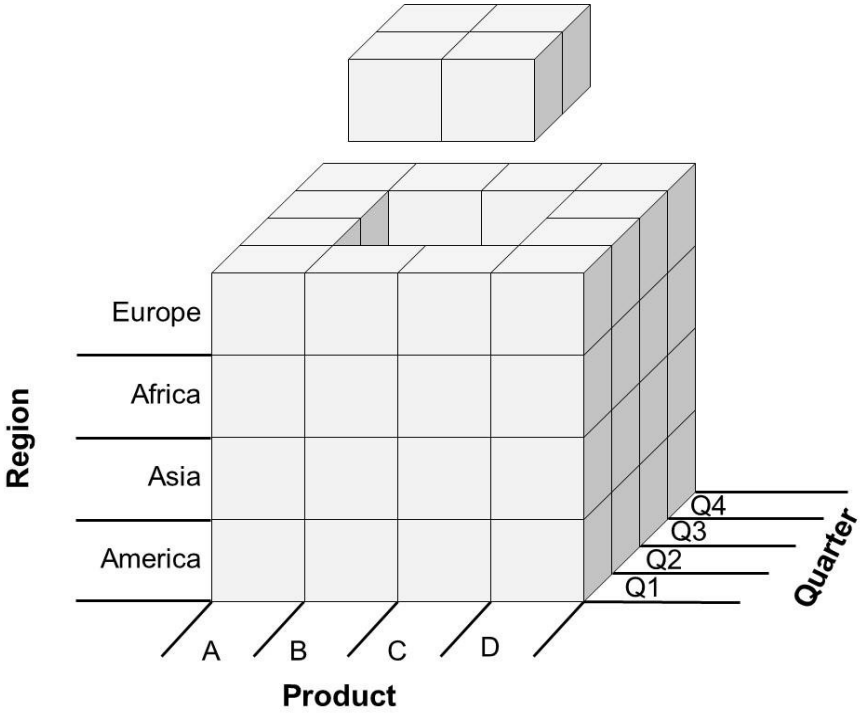
Reverse: drill-down!

OLAP Operators (3/4)

- **Drill-across** is another OLAP operation whereby information from 2 or more connected fact tables is accessed.
- **Slicing** represents the operation whereby one of the dimensions is set at a particular value.



- **Dicing** corresponds to a range selection on one or more dimensions



OLAP Queries in SQL (1/19)

- SQL-99 introduced 3 extensions to the GROUP BY statement: the CUBE, ROLLUP and GROUPING SETS operator
- The **CUBE** operator computes a union of GROUP BY's on every subset of the specified attribute types

OLAP Queries in SQL (2/19)

PRODUCT	QUARTER	REGION	SALES
A	Q1	Europe	10
A	Q1	America	20
A	Q2	Europe	20
A	Q2	America	50
A	Q3	America	20
A	Q4	Europe	10
A	Q4	America	30
B	Q1	Europe	40
B	Q1	America	60
B	Q2	Europe	20
B	Q2	America	10
B	Q3	America	20
B	Q4	Europe	10
B	Q4	America	40

```
SELECT QUARTER, REGION, SUM(SALES)
FROM SALESTABLE
GROUP BY CUBE (QUARTER, REGION)
```

- this query computes the union of 2^2=4 groupings of the SALESTABLE being: {(quarter,region), (quarter), (region), ()}, where () denotes an empty group list
- resulting multiset will have 4*2+4*1+1*2+1 or 15 tuples

OLAP Queries in SQL (4/19)

QUARTER	REGION	SALES
Q1	Europe	50
Q1	America	80
Q2	Europe	40
Q2	America	60
Q3	Europe	NULL
Q3	America	40
Q4	Europe	20
Q4	America	80
Q1	NULL	130
Q2	NULL	100
Q3	NULL	40
Q4	NULL	90
NULL	Europe	110
NULL	America	250
NULL	NULL	360

```
CASE WHEN grouping(QUARTER) = 1
THEN 'All' ELSE QUARTER END AS QUARTER and
CASE WHEN grouping(REGION) = 1 THEN
'All' ELSE REGION END AS REGION
```

OLAP Queries in SQL (5/19)

- **ROLLUP** operator computes the union on every prefix of the list of specified attribute types, from the most detailed up to the grand total
- Key difference between the ROLLUP and CUBE operator is that the former generates a result set showing the aggregates for a hierarchy of values of the specified attribute types, whereas the latter generates a result set showing the aggregates for all combinations of values of the selected attribute types

```
SELECT QUARTER, REGION, SUM(SALES)
FROM SALESTABLE
GROUP BY ROLLUP (QUARTER, REGION)
```

- This query generates the union of three groupings {(quarter,region), (quarter), ()} where () again represents the full aggregation
- Resulting multiset will thus have 4*2+4+1 or 13 rows

OLAP Queries in SQL (7/19)

QUARTER	REGION	SALES
Q1	Europe	50
Q1	America	80
Q2	Europe	40
Q2	America	60
Q3	Europe	NULL
Q3	America	40
Q4	Europe	20
Q4	America	80
Q1	NULL	130
Q2	NULL	100
Q3	NULL	40
Q4	NULL	90
NULL	NULL	360

- GROUP BY ROLLUP construct can also be applied to attribute types that represent different aggregation levels along the same dimension

```
SELECT REGION, COUNTRY, CITY, SUM(SALES)
FROM SALESTABLE
GROUP BY ROLLUP (REGION, COUNTRY, CITY)
```

OLAP Queries in SQL (9/19)

- **GROUPING SETS** operator generates a result set equivalent to that generated by a UNION ALL of multiple simple GROUP BY clauses

```
SELECT QUARTER, REGION,
SUM(SALES)  FROM SALESTABLE
GROUP BY GROUPING SETS ((QUARTER), (REGION))
```

```
SELECT QUARTER, NULL,
SUM(SALES)  FROM SALESTABLE
GROUP BY QUARTER
UNION ALL
SELECT NULL, REGION,
SUM(SALES)  FROM SALESTABLE
GROUP BY REGION
```

OLAP Queries in SQL (10/19)

QUARTER	REGION	SALES
Q1	NULL	130
Q2	NULL	100
Q3	NULL	40
Q4	NULL	90
NULL	Europe	110
NULL	America	250

```
SELECT QUARTER, REGION, SUM(SALES)
FROM SALESTABLE
GROUP BY CUBE (QUARTER, REGION)
```

```
SELECT QUARTER, REGION, SUM(SALES)
FROM SALESTABLE
GROUP BY GROUPING SETS ((QUARTER,
REGION), (QUARTER), (REGION), ())
```

```
SELECT QUARTER, REGION, SUM(SALES)
FROM SALESTABLE
GROUP BY ROLLUP (QUARTER, REGION)
```

```
SELECT QUARTER, REGION, SUM(SALES)
FROM SALESTABLE
GROUP BY GROUPING SETS ((QUARTER,
REGION), (QUARTER),())
```

OLAP Queries in SQL (13/19)

- SQL2003 introduced support for 2 types of other activities: ranking and windowing

OLAP Queries in SQL (14/19)

PRODUCT	SALES
A	50
B	20
C	10
D	45
E	40
F	30
G	60
H	20
I	15
J	25

```
SELECT PRODUCT, SALES,
RANK() OVER (ORDER BY SALES ASC) as
RANK_SALES,
DENSE_RANK() OVER (ORDER BY SALES ASC) as
DENSE_RANK_SALES, PERCENT_RANK() OVER (ORDER
BY SALES ASC) as PERC_RANK_SALES,
CUM_DIST() OVER (ORDER BY SALES ASC) as
CUM_DIST_SALES,
FROM SALES
ORDER BY RANK_SALES ASC
```

OLAP Queries in SQL (16/19)

Product	Sales	RANK_SALES	DENSE_RANK_SALES	PERC_RANK_SALES	CUM_DIST_SALES
C	10	1	1	0	0.1
I	15	2	2	1/9=0.11	0.2
B	20	3	3	2/9=0,22	0.4
H	20	3	3	2/9=0,22	0.4
J	25	5	4	4/9=0,44	0.5
F	30	6	5	5/9=0,55	0.6
E	40	7	6	6/9=0,66	0.7
D	45	8	7	7/9=0,77	0.8
A	50	9	8	8/9=0,88	0.9
G	60	10	9	9/9=1	1

- **Windowing** allows calculating cumulative totals or running averages based on a specified time window.

QUARTER	REGION	SALES
1	America	10
2	America	20
3	America	10
4	America	30
1	Europe	10
2	Europe	20
3	Europe	10
4	Europe	20

```
SELECT QUARTER, REGION, SALES,
AVG(SALES) OVER (PARTITION BY REGION
ORDER BY QUARTER ROWS BETWEEN 1
PRECEDING AND 1 FOLLOWING) AS SALES_AVG
FROM SALES
ORDER BY REGION, QUARTER, SALES_AVG
```

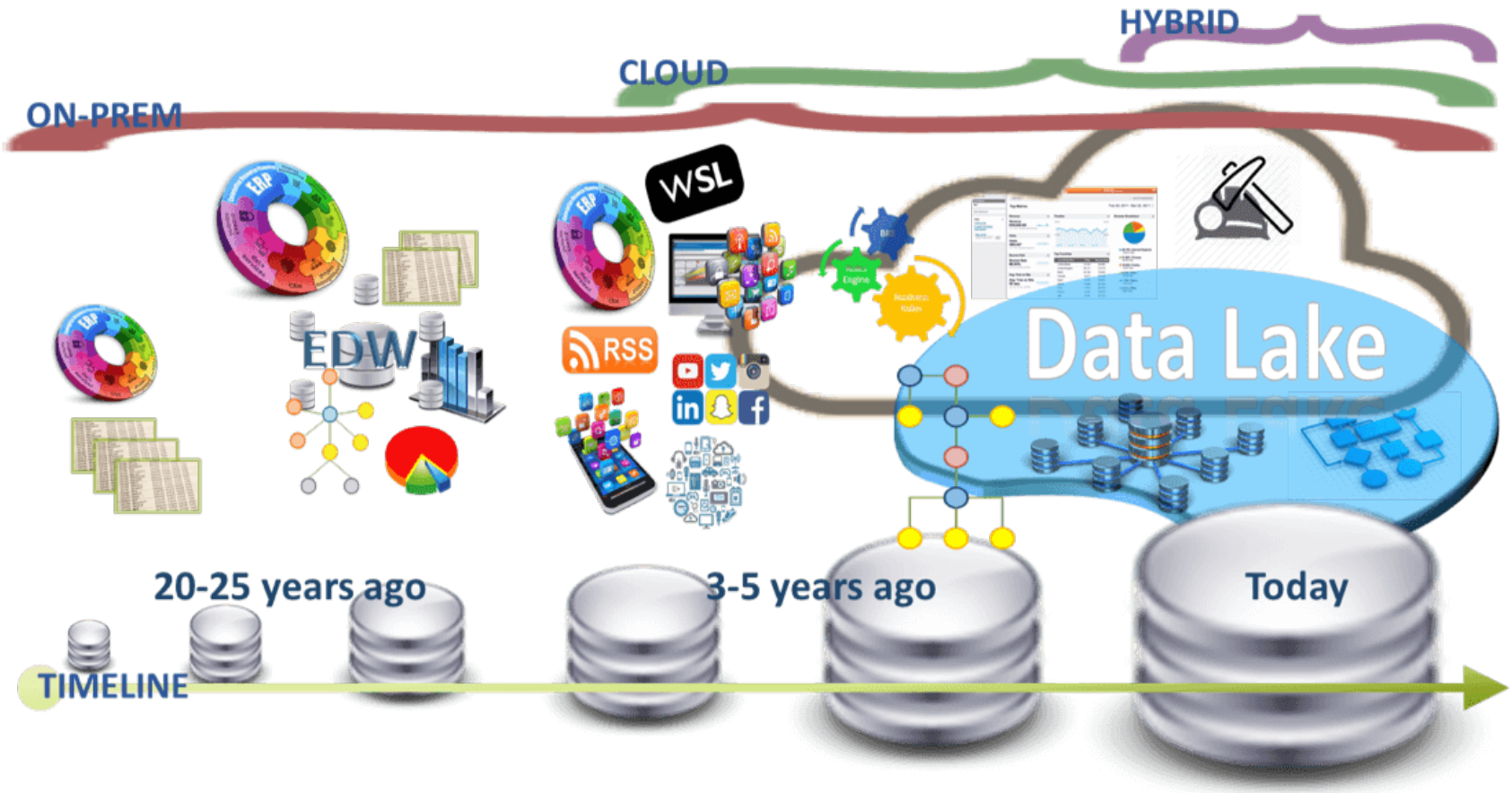
OLAP Queries in SQL (19/19)

QUARTER	REGION	SALES	SALES_AVG
1	America	10	15
2	America	20	13,33
3	America	10	20
4	America	30	20
1	Europe	10	15
2	Europe	20	13,33
3	Europe	10	16,67
4	Europe	20	15

A Shift Towards A Data Lake Architectures

- A traditional approach to information management is no longer suitable due to costs and inability to adapt
  - 70% of development costs from ETL include effort to consolidate, prepare, standardize, and transform data for downstream analytics
  - Costs involve initial capital investment for hardware, software licensing for databases, data integration and analytics platform alone
  - Need to react to emerging changes in the proliferation of data, new and emerging technologies and cloud-based integrated service platforms.
- Traditional data storage and analytic tools can no longer provide the agility and flexibility required to deliver relevant business insights and competitive advantage

From Data Warehousing to Data Lakes



Data Lakes (1/2)

- A data lake is an architectural approach specifically designed to handle data of every variety, ingestion velocity, and storage volume

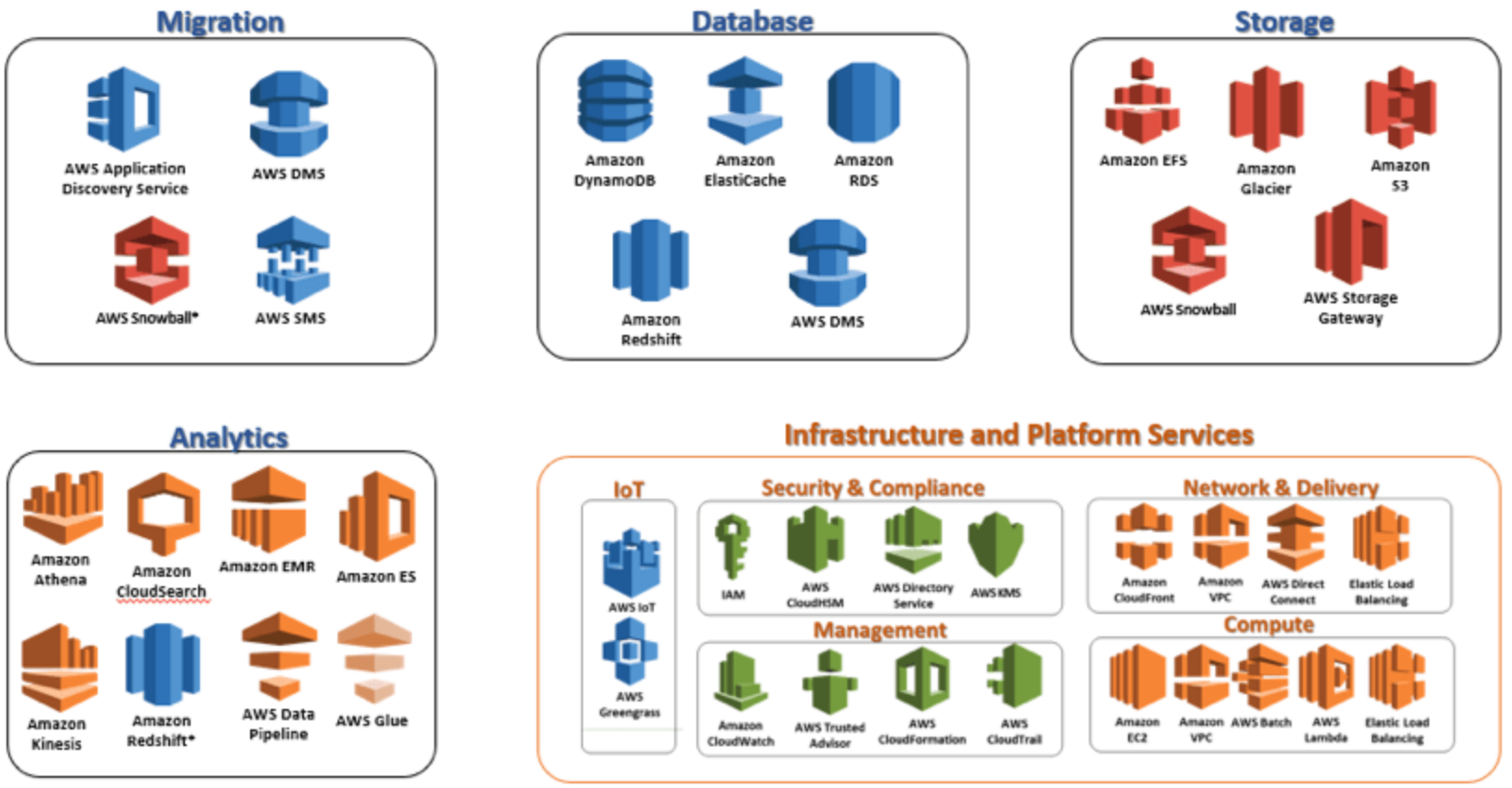
- A data lake allows the storage of massive amounts of data into a central location so it's readily available to be categorized, processed, analyzed, and consumed by diverse groups within an organization

Data Lakes (2/2)

- Since data can be stored as-is, there is no need to convert it to a predefined schema, as typically required in traditional RDBMS-driven architectures.
  - Consumer usage patterns and the sourcing of the data itself directly influence how data is collected, stored, processed, moved, transformed, automated, and visualized
  - Data is the ultimate asset with boundless usage patterns, now being generated and consumed by humans, machines, devices, sensors, and applications

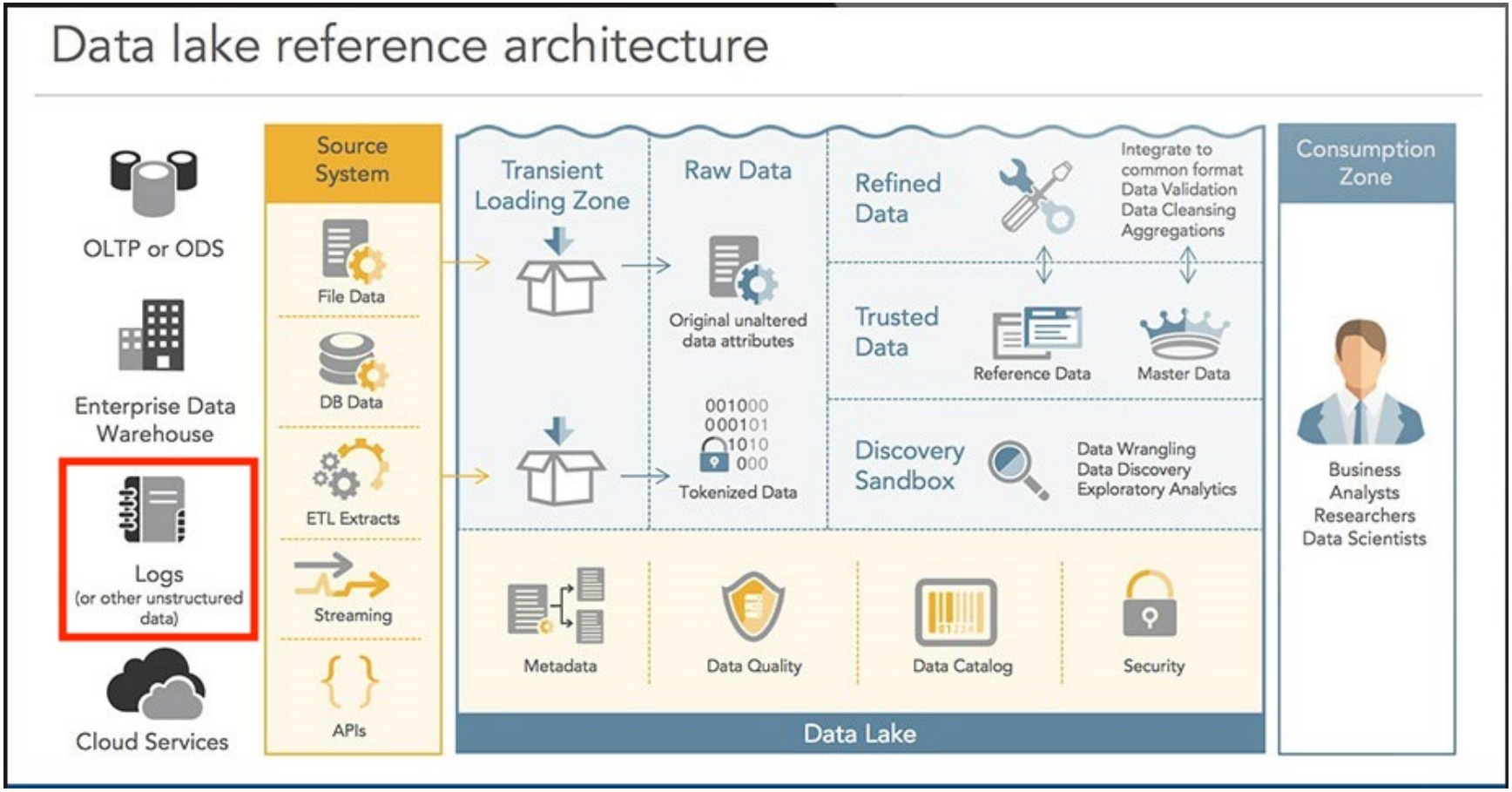
Data Lake Concept

AWS Service Areas when Solutioning Use Cases for a Data Lake



- Data is stored in a raw state initially, and some use cases will use raw data as is
- More often, solutions require varying degrees of data preparedness based on a collection of query usage profiles which correlate to actual use cases
- Based on the solution, data may be refined and staged with the intent to promote modularity and reuse
  - i.e., not over process the data set because it is intended for multiple purposes downstream, such as AWS RedShift for relational analytics, AWS Elasticsearch for text search, or an optimized distributed file system for low cost active archive storage which can be queried with an MPP SQL engine

Data Lake Reference Architecture



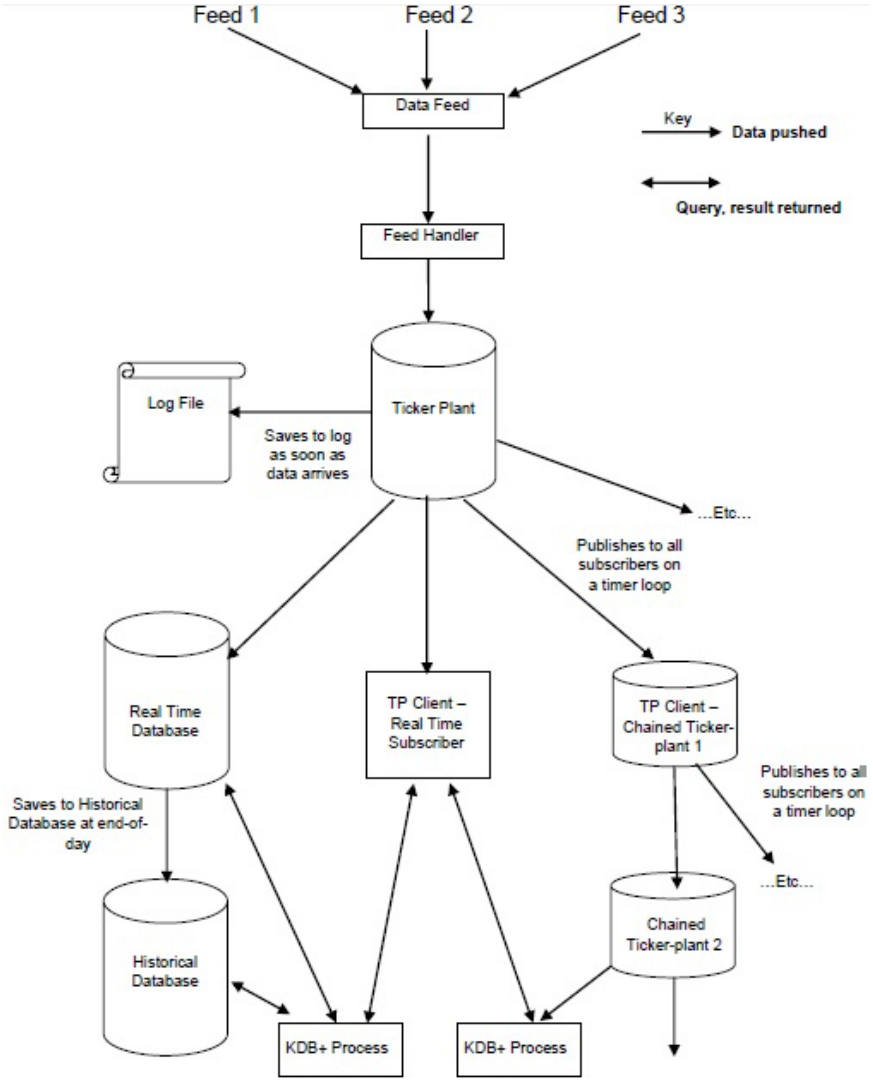
Sample High-Performance In-Memory Database / Data Lake

KDB+ is a high-performance, high-volume database designed from the outset to handle tremendous volumes of data

KDB+ is fully 64-bit, and has built-in multi-core processing and multi-threading (InfluxDB is an alternative)

The same architecture is used for real-time and historical data. The database incorporates its own powerful query language, **q,** so analytics can be run directly on the data

kdb+tick (see diagram on the right) is an architecture which allows the capture, processing, and querying of real-time and historical Fintech trading data



https://www.tutorialspoint.com/kdbplus/kdbplus_architecture.htm

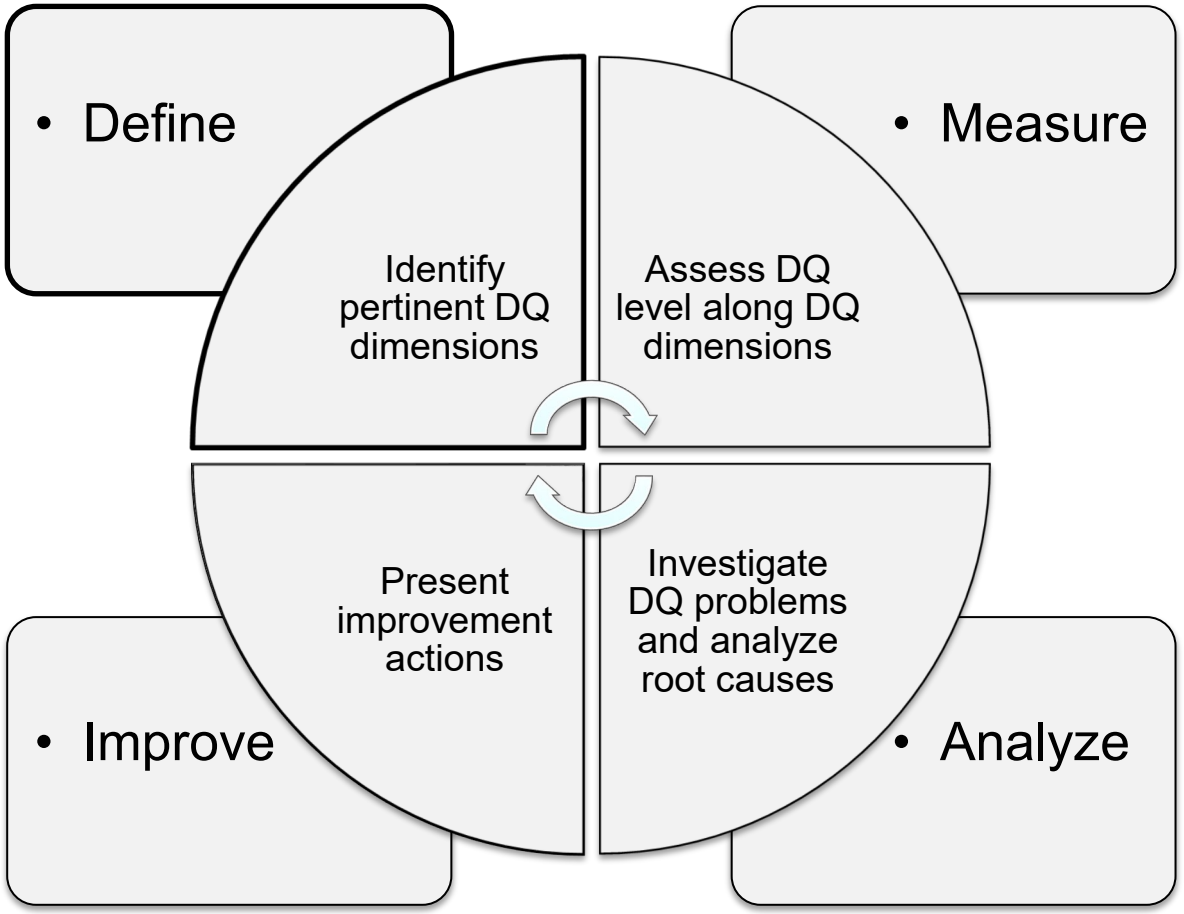
From Data Lakes to Data Swamps

- A common challenge with the Data Lake architecture:
  - Without the appropriate data quality and governance framework in place, when terabytes of structured and unstructured data flow into the Data Lakes, it often becomes extremely difficult to sort through their content
- Data Lakes can turn into Data Swamps as the stored data become too messy to be usable
- Many organizations are now calling for more data governance and metadata management practices to prevent Data Swamps from forming

Data Governance

- To manage and safeguard data quality, a data governance culture should be put in place assigning clear roles and responsibilities
  - » manage data as an asset rather than a liability
- Different frameworks have been introduced for data quality management and data quality improvement
  - » examples: Total Data Quality Management (TDQM), Total Quality Management (TQM), Capability Maturity Model Integration (CMMI), ISO 9000, Control Objectives for Information and Related Technology (CobiT), Data Management Body of Knowledge (DMBOK), Information Technology Infrastructure Library (ITIL) and Six Sigma

Data Governance



Wang (1998)

Data Governance

- Annotate the data with data quality metadata as a short term solution
  - can be stored in the catalog
  - E.g., credit risk models could incorporate an additional risk factor to account for uncertainty in the data
- Unfortunately, many data governance efforts (if any) are mostly reactive and ad-hoc

Summary

- Operational versus Tactical/Strategic Decision Making
- Data Warehouse Definition
- Data Warehouse Schemas
- The Extraction Transformation and Loading (ETL) Process
- Data Marts
- Virtual Data Warehouses and Virtual Data Marts
- Operational Data Store
- Data Warehouses versus Data Lakes
- Business intelligence
- Data Lakes and Data Swamps