Data Warehousing

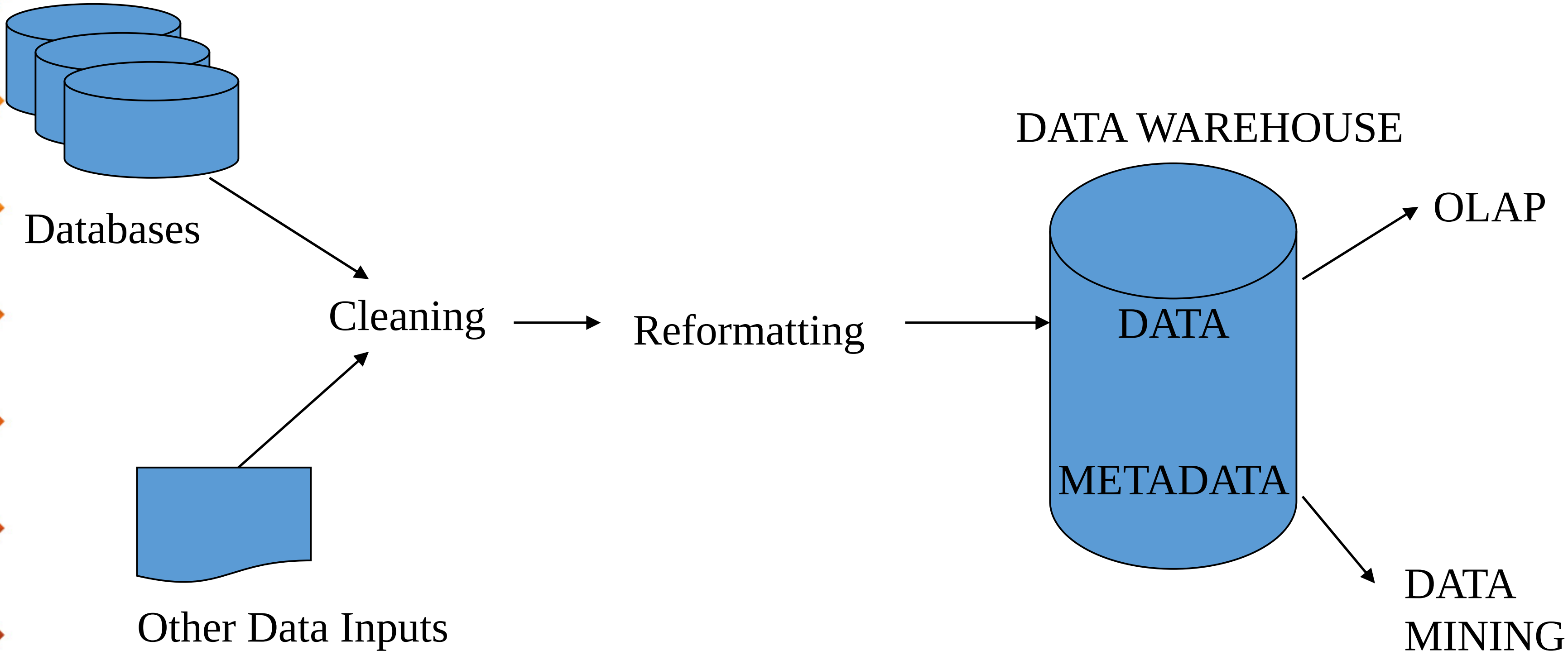
INFS602 Physical Database Design

Agenda

- Data warehousing
- Operations
- Data warehousing strategies

What is a Data Warehouse?

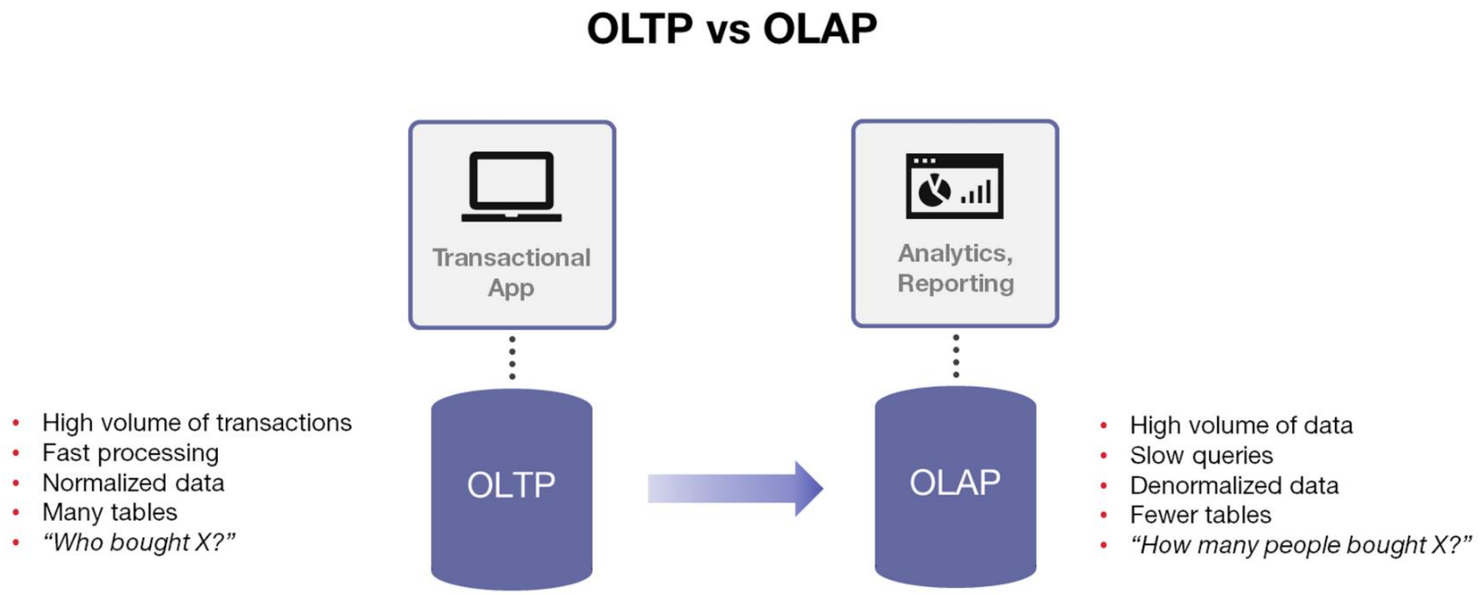
- A Data Warehouse is a *subject-oriented*, *integrated*, *non-volatile*, *time-varying* repository of data
- It is a central location that data from different databases are stored and separeted from operational database



Characteristic	OLTP Database	OLAP Data Warehousing
Purpose	Supports transaction processing	Supports information requests
Source of data	Business transactions	Multiple files, databases–internal and external to firm
Data Access Allowed Users	Read and Write	Read Only
Primary data Access Mode	Simple database update and queries	Simple and complex queries with increasing use of data mining to recognise patterns in the data
Primary Database Model Employed	Relational and sometimes Hierarchical	Relational
Level of Detail	Detailed transactions	Often summarised
Historical Data	Current data only	Multiple years of data
Update Process	On-line, ongoing process as transactions are captured	Periodic process, once per week or once per month
Ease of Update	Routine and easy	Complex, must combine data from many sources-both internal and external
Data Integrity Issues	Each individual transaction must be closely edited	Major effort to “clean” and integrate data from multiple sources.

OLTP

- On-line transaction processing (OLTP) is the traditional way of using a database
  - Short transactions (read/update few records) with ACID (**Atomicity, Consistency, Isolation, Durability**) properties
  - Normally, only the last version of data is stored in the database



- High volume of transactions
- Fast processing
- Normalized data
- Many tables
- *"Who bought X?"*

- High volume of data
- Slow queries
- Denormalized data
- Fewer tables
- *"How many people bought X?"*

DSS & OLAP

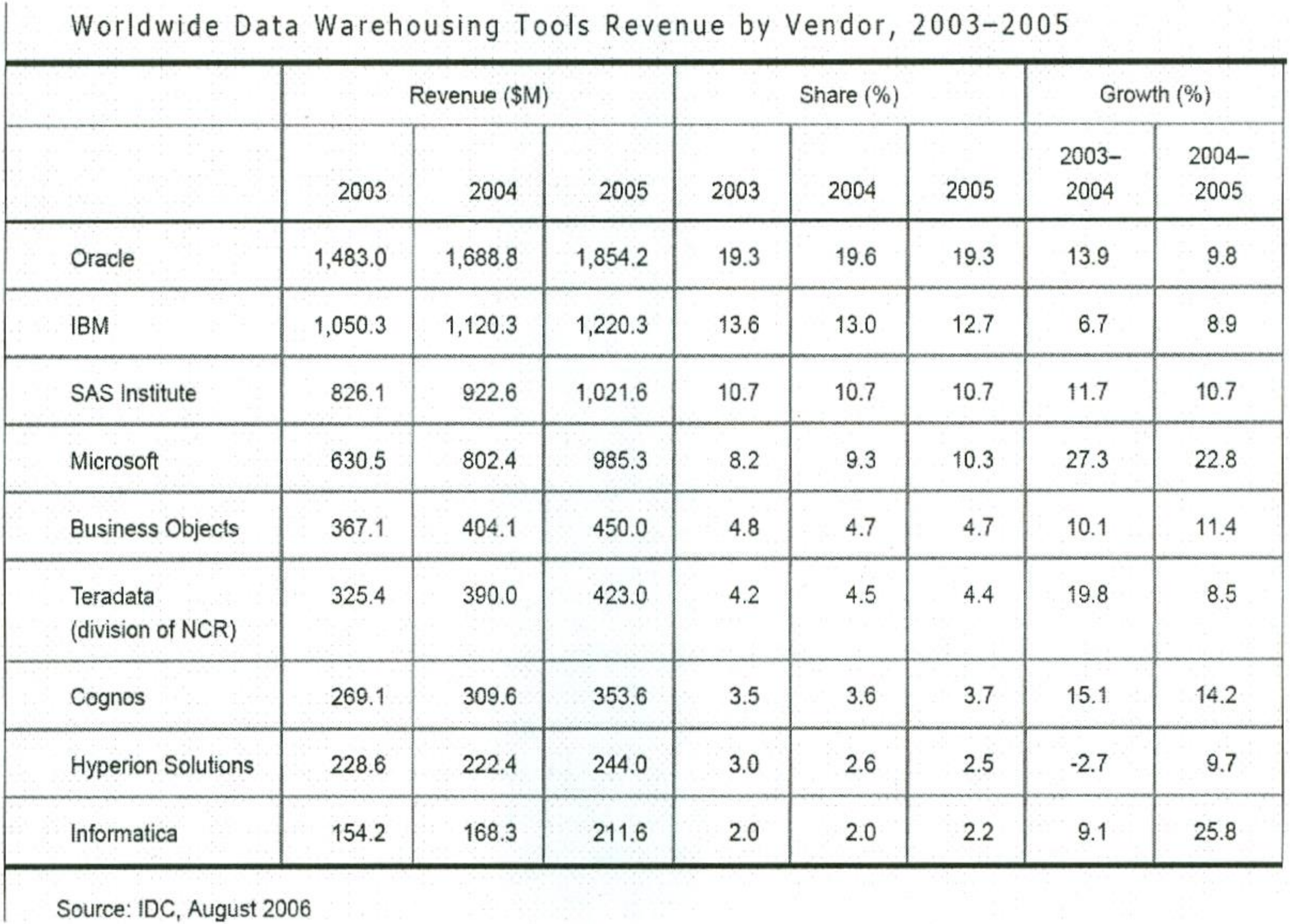
- Decision support systems - help the executive, manager, analyst make faster and better decisions.
  - What were the sales volumes by region and product category for the last year?
  - Will a 10% discount increase sales volumes sufficiently?
- On-line analytical processing (OLAP) is an element of decision support systems (DSS)

Reasons for Building Data Warehouses

- Performance
  - OLAP applications need different organization of data
  - Complex OLAP queries would degrade OLTP performance
- Availability
  - Separation increases availability
  - Possibly the only way to query the disparate data sources

…. and the market is there!

Data Warehousing Tools Market



Worldwide Data Warehousing Tools Revenue by Vendor, 2003–2005

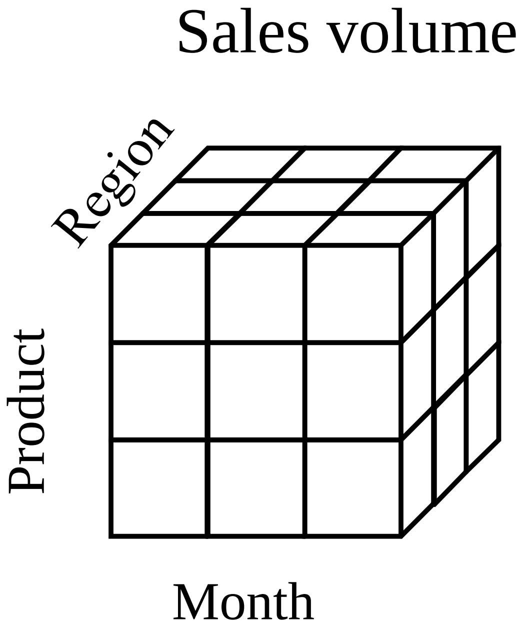
	Revenue ($M)			Share (%)			Growth (%)	
	2003	2004	2005	2003	2004	2005	2003–2004	2004–2005
Oracle	1,483.0	1,688.8	1,854.2	19.3	19.6	19.3	13.9	9.8
IBM	1,050.3	1,120.3	1,220.3	13.6	13.0	12.7	6.7	8.9
SAS Institute	826.1	922.6	1,021.6	10.7	10.7	10.7	11.7	10.7
Microsoft	630.5	802.4	985.3	8.2	9.3	10.3	27.3	22.8
Business Objects	367.1	404.1	450.0	4.8	4.7	4.7	10.1	11.4
Teradata (division of NCR)	325.4	390.0	423.0	4.2	4.5	4.4	19.8	8.5
Cognos	269.1	309.6	353.6	3.5	3.6	3.7	15.1	14.2
Hyperion Solutions	228.6	222.4	244.0	3.0	2.6	2.5	-2.7	9.7
Informatica	154.2	168.3	211.6	2.0	2.0	2.2	9.1	25.8

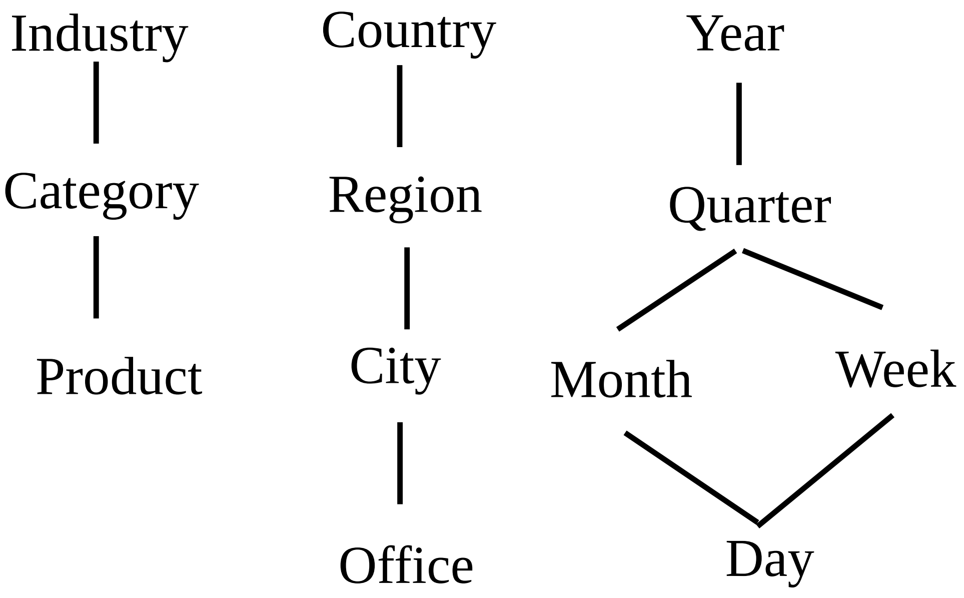
Source: IDC, August 2006

Multidimensional Data

Dimensions: Product, Region, Date

Hierarchical summarization paths:





Industry

Category

Product

Country

Region

City

Office

Year

Quarter

Month

Week

Day

Operations

- Roll up: summarize data
- Drill down: go from higher level summary to lower level summary or detailed data
- Slice and dice: select and project
- Pivot: re-orient cube

Roll up

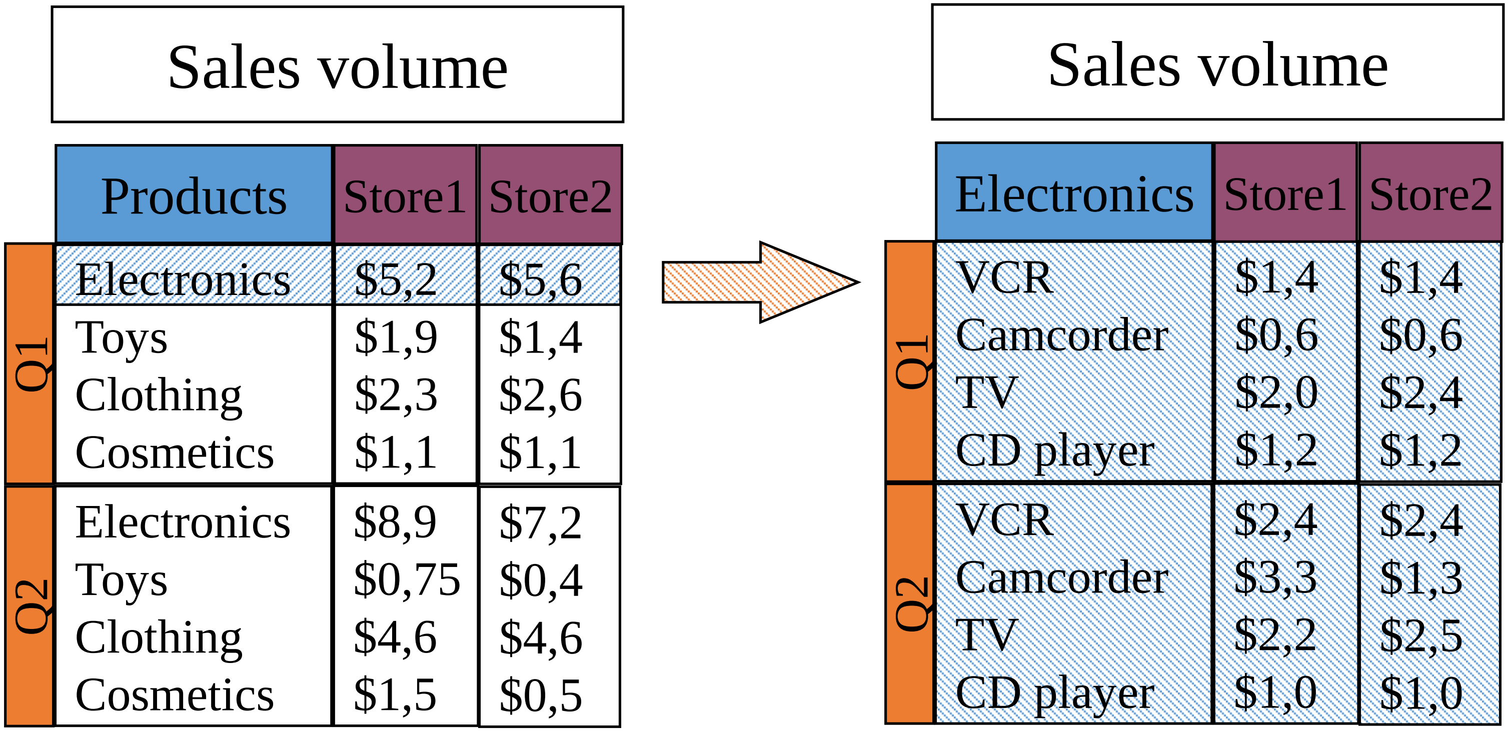
Sales volume

	Products	Store1	Store2
Q1	Electronics	$5,2	$5,6
	Toys	$1,9	$1,4
	Clothing	$2,3	$2,6
	Cosmetics	$1,1	$1,1
Q2	Electronics	$8,9	$7,2
	Toys	$0,75	$0,4
	Clothing	$4,6	$4,6
	Cosmetics	$1,5	$0,5

Sales volume

	Products	Store1	Store2
Year 1996	Electronics	$14,1	$12,8
	Toys	$2,65	$1,8
	Clothing	$6,9	$7,2
	Cosmetics	$2,6	$1,6

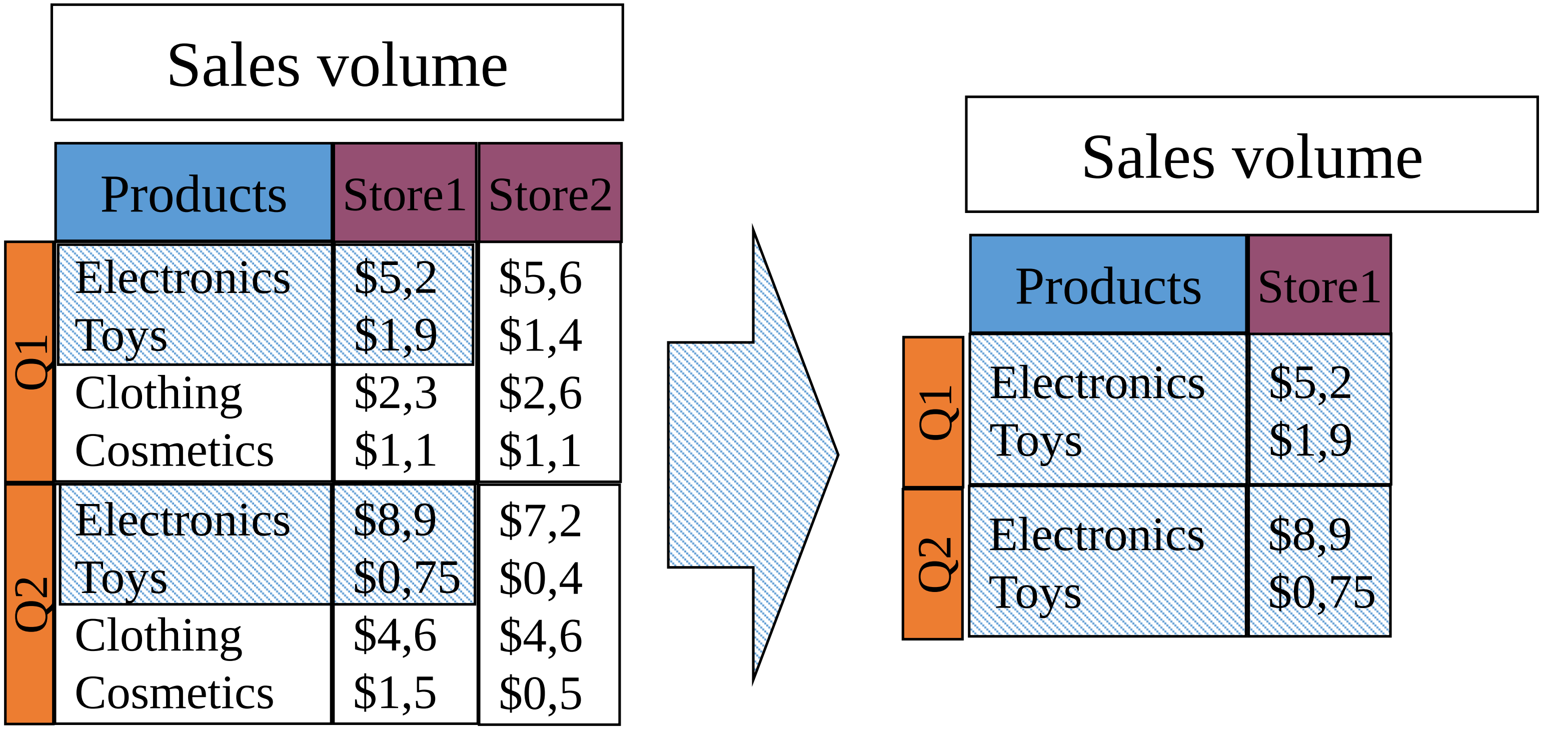
Drill down



Sales volume			
	Products	Store1	Store2
Q1	Electronics	$5,2	$5,6
	Toys	$1,9	$1,4
	Clothing	$2,3	$2,6
	Cosmetics	$1,1	$1,1
Q2	Electronics	$8,9	$7,2
	Toys	$0,75	$0,4
	Clothing	$4,6	$4,6
	Cosmetics	$1,5	$0,5

Sales volume			
	Electronics	Store1	Store2
Q1	VCR	$1,4	$1,4
	Camcorder	$0,6	$0,6
	TV	$2,0	$2,4
	CD player	$1,2	$1,2
Q2	VCR	$2,4	$2,4
	Camcorder	$3,3	$1,3
	TV	$2,2	$2,5
	CD player	$1,0	$1,0

Slice and Dice



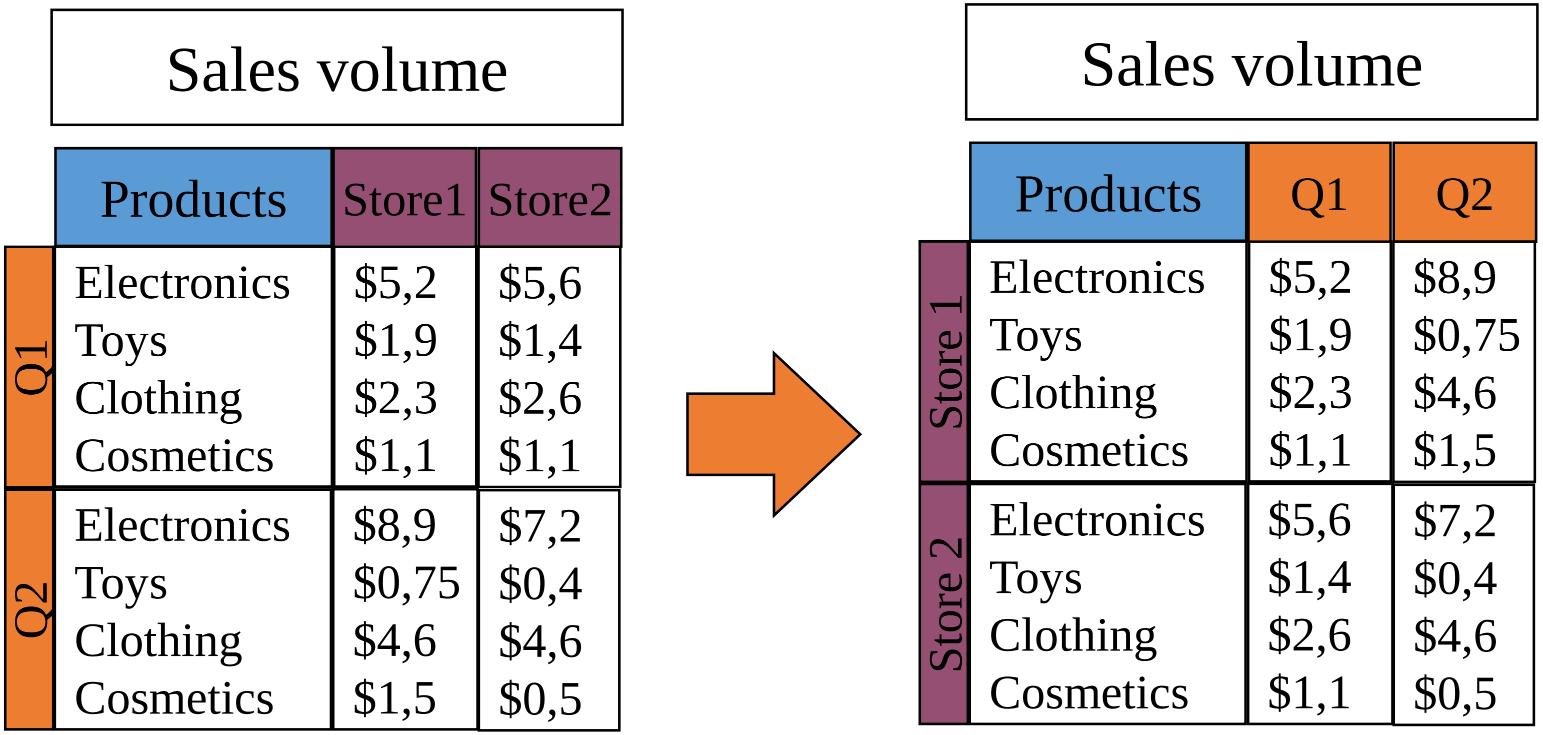
Sales volume

	Products	Store1	Store2
Q1	Electronics	$5,2	$5,6
	Toys	$1,9	$1,4
	Clothing	$2,3	$2,6
	Cosmetics	$1,1	$1,1
Q2	Electronics	$8,9	$7,2
	Toys	$0,75	$0,4
	Clothing	$4,6	$4,6
	Cosmetics	$1,5	$0,5

Sales volume

	Products	Store1
Q1	Electronics	$5,2
	Toys	$1,9
Q2	Electronics	$8,9
	Toys	$0,75

Pivot



Sales volume

	Products	Store1	Store2
Q1	Electronics	$5,2	$5,6
	Toys	$1,9	$1,4
	Clothing	$2,3	$2,6
	Cosmetics	$1,1	$1,1
Q2	Electronics	$8,9	$7,2
	Toys	$0,75	$0,4
	Clothing	$4,6	$4,6
	Cosmetics	$1,5	$0,5

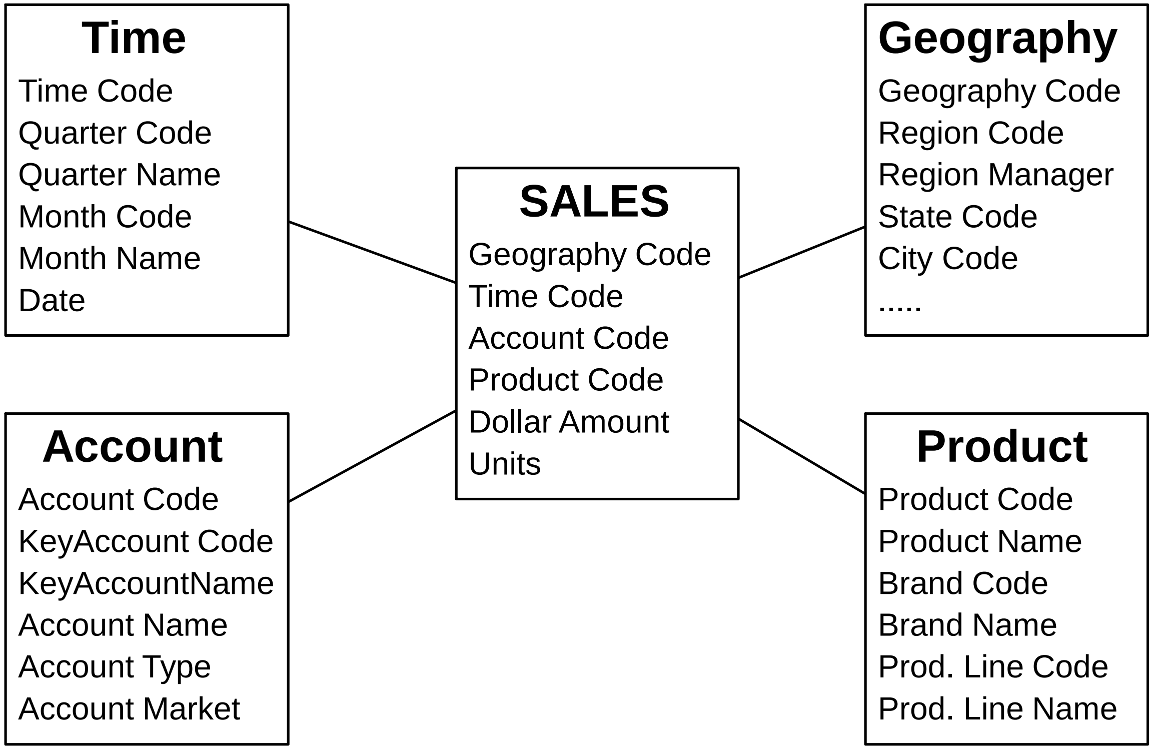
Sales volume

	Products	Q1	Q2
Store 1	Electronics	$5,2	$8,9
	Toys	$1,9	$0,75
	Clothing	$2,3	$4,6
	Cosmetics	$1,1	$1,5
Store 2	Electronics	$5,6	$7,2
	Toys	$1,4	$0,4
	Clothing	$2,6	$4,6
	Cosmetics	$1,1	$0,5

1. Star Schema

- A star schema consists of one central **fact** table and several denormalized **dimension** tables.
- The **measures** of interest for OLAP are stored in the fact table (e.g. Dollar Amount, Units in the table SALES).
- For each dimension of the multidimensional model there exists a dimension table (e.g. Geography, Product, Time, Account) with all the **levels** of aggregation and the extra properties of these levels.

Star Schema…



Time
Time Code
Quarter Code
Quarter Name
Month Code
Month Name
Date

SALES
Geography Code
Time Code
Account Code
Product Code
Dollar Amount
Units

Geography
Geography Code
Region Code
Region Manager
State Code
City Code
…..

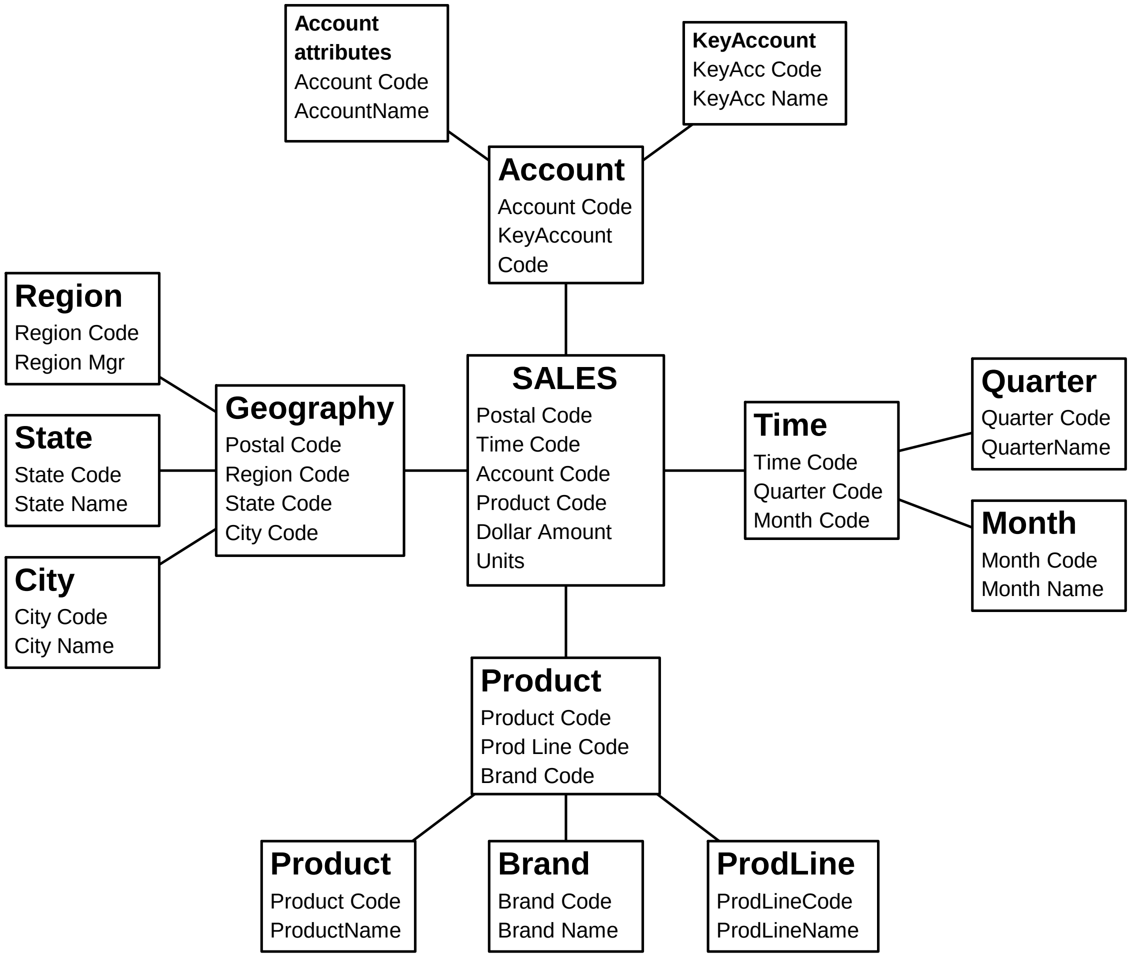
Account
Account Code
KeyAccount Code
KeyAccountName
Account Name
Account Type
Account Market

Product
Product Code
Product Name
Brand Code
Brand Name
Prod. Line Code
Prod. Line Name

2. Snowflake Schema

- The normalized version of the star schema
- Explicit treatment of dimension hierarchies (each level has its own table)
- Easier to maintain, slower in query answering

Snowflake Schema…



Account attributes
Account Code
AccountName

KeyAccount
KeyAcc Code
KeyAcc Name

Account
Account Code
KeyAccount Code

Region
Region Code
Region Mgr

State
State Code
State Name

City
City Code
City Name

Geography
Postal Code
Region Code
State Code
City Code

SALES
Postal Code
Time Code
Account Code
Product Code
Dollar Amount
Units

Time
Time Code
Quarter Code
Month Code

Quarter
Quarter Code
QuarterName

Month
Month Code
Month Name

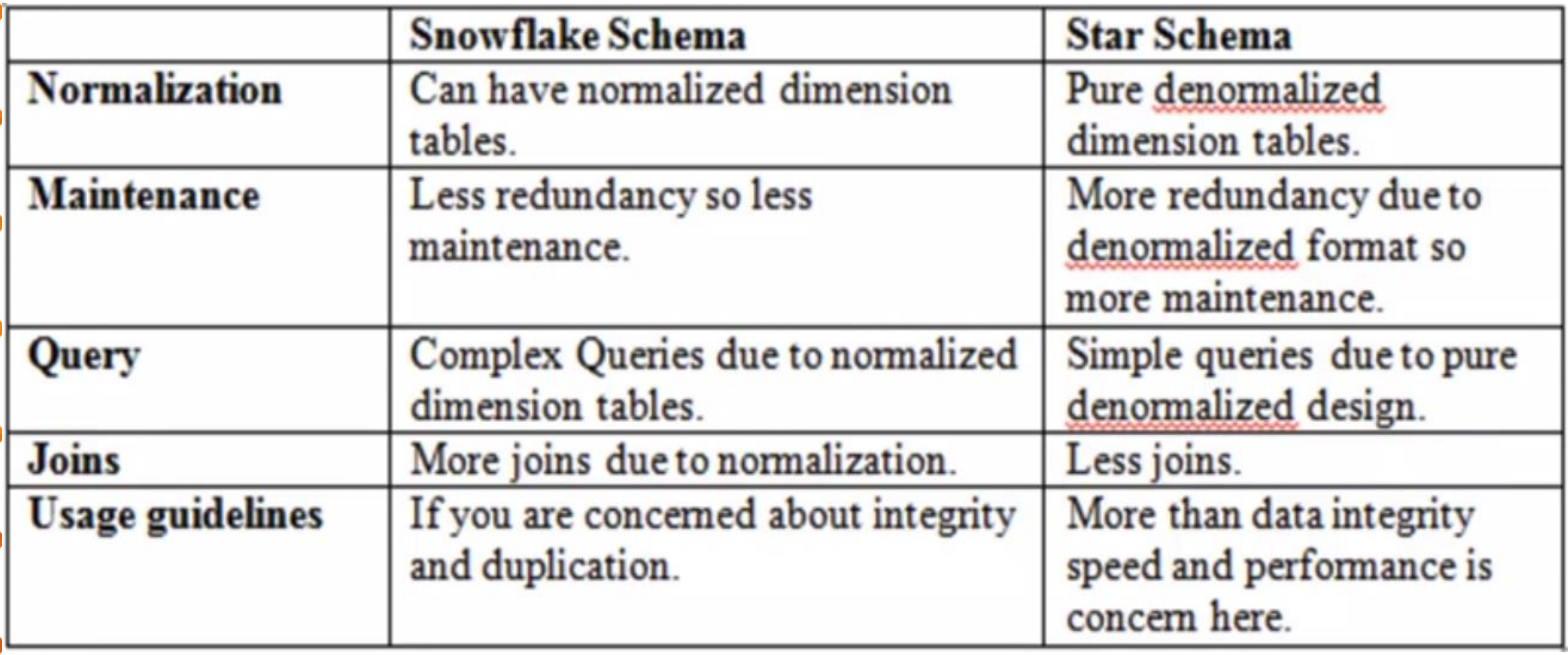
Product
Product Code
Prod Line Code
Brand Code

Product
Product Code
ProductName

Brand
Brand Code
Brand Name

ProdLine
ProdLineCode
ProdLineName

Star Vs. Snowflake Schema

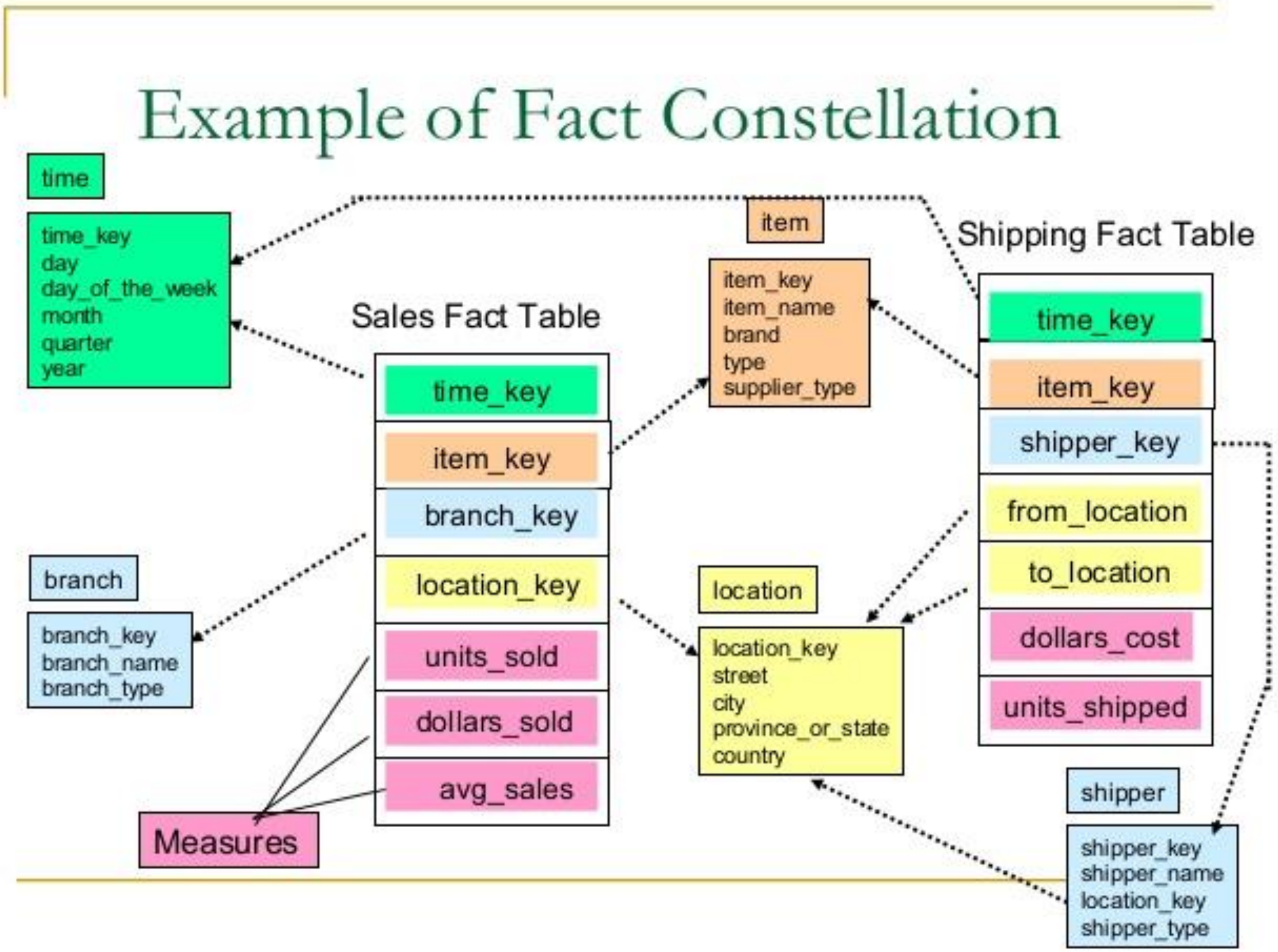


	Snowflake Schema	**Star Schema**
Normalization	Can have normalized dimension tables.	Pure denormalized dimension tables.
Maintenance	Less redundancy so less maintenance.	More redundancy due to denormalized format so more maintenance.
Query	Complex Queries due to normalized dimension tables.	Simple queries due to pure denormalized design.
Joins	More joins due to normalization.	Less joins.
Usage guidelines	If you are concerned about integrity and duplication.	More than data integrity speed and performance is concern here.

3. Fact Constellation

- Multiple fact tables that share many dimension tables
- Example: projected expense and the actual expense may share dimensional tables

Fact Constellation…



Source: http://www.slideshare.net/Krish_ver2/14-data-warehouse

Operational Processes

- Propagate updates of source data to the warehouse
- Issues:
  - when to refresh
    - on every update
    - periodically
    - refresh policy set by administrator
  - how to refresh

Refreshment Techniques

- Full extract from base tables
- Incremental techniques
  - detect changes on base tables
    - data shipping (uses triggers to update warehouse tables)
    - transaction shipping (uses transaction log to ship transactions over to warehouse server for execution)

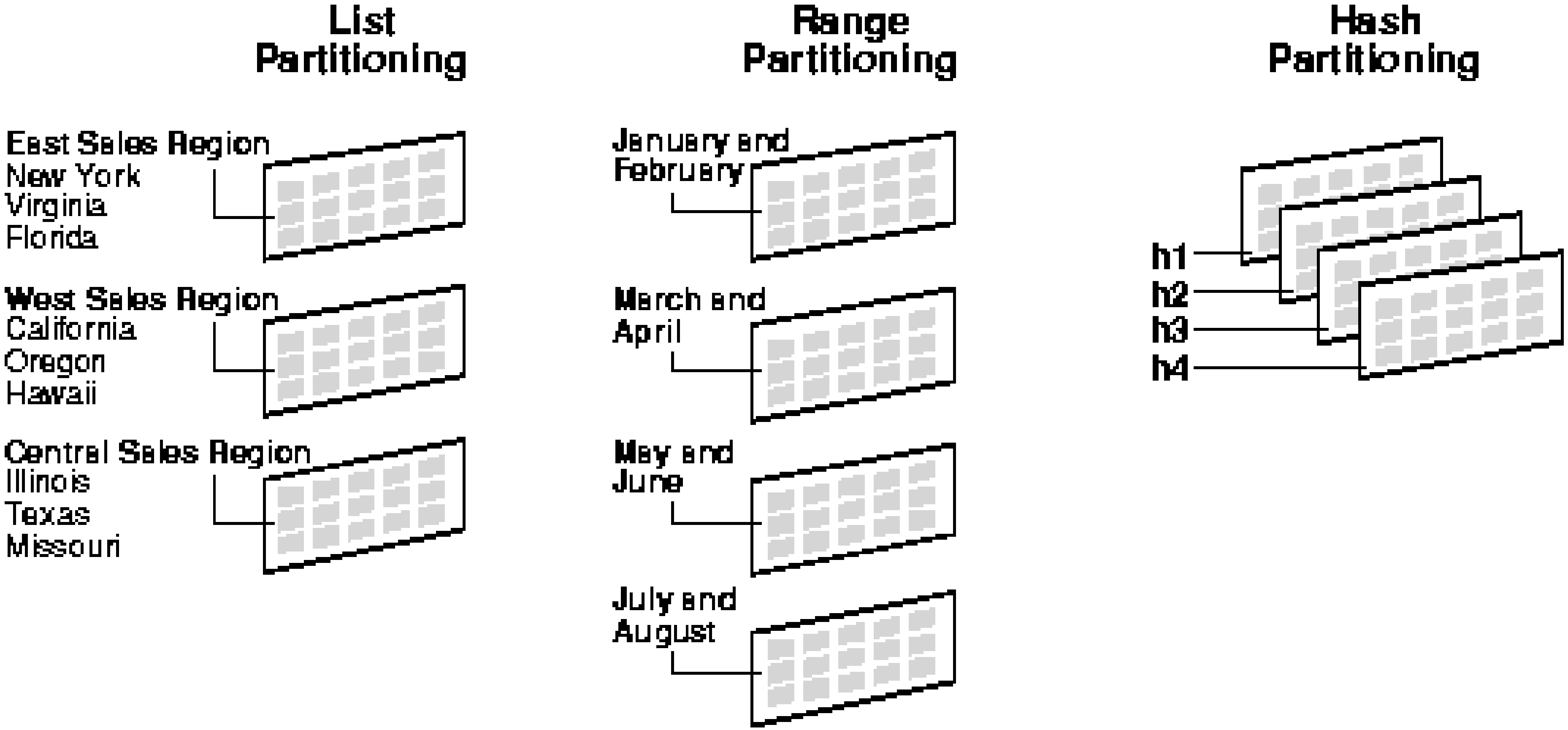
Accelerating the Refreshment Process

- Data Partitioning can be used to speed up refreshment
- Need to define a partition key

Partitioning

- Decomposing a large table or index into smaller and more manageable pieces.
- Data management operations can be performed at the partition level.
- Improves query performance.
- Can be implemented without requiring any modification to applications.

Partitioning Methods



List Partitioning

- Enables control on how rows map to partitions
- Specify a list of discrete values for the partitioning key
- Example partitioning a sales table by region

```
CREATE TABLE sales_list (salesman_id NUMBER(5),
  salesman_name VARCHAR2(30), sales_state VARCHAR2(20),
  sales_amount NUMBER(10), sales_date DATE)
PARTITION BY LIST(sales_state)
( PARTITION sales_west VALUES IN('California',
  'Hawaii'),
PARTITION sales_east VALUES IN ('New York', 'Virginia',
  'Florida'),
PARTITION sales_central VALUES IN('Texas', 'Illinois'),
)
```

Range Partitioning

- Maps data to partitions based on pages of partition key values.
- Most common type of partitioning often used with dates.

```
CREATE TABLE sales_range (salesman_id NUMBER(5),
salesman_name VARCHAR2(30), sales_amount NUMBER(10),
sales_date DATE) PARTITION BY RANGE(sales_date)
(PARTITION sales_jan2002 VALUES LESS THAN
(TO_DATE('02/01/2002',‘MM/DD/YYYY')),
PARTITION sales_feb2002 VALUES LESS THAN
(TO_DATE('03/01/2002',‘MM/DD/YYYY')),
PARTITION sales_mar2002 VALUES LESS THAN
(TO_DATE('04/01/2002',‘MM/DD/YYYY')),
PARTITION sales_apr2002 VALUES LESS THAN
(TO_DATE('05/01/2002',‘MM/DD/YYYY')), )
```

Hash Partitioning

- Used when range or list partitioning are not useful.
- Used when the amount of data to map into a given range is unknown.

```
CREATE TABLE sales_hash (salesman_id
NUMBER(5), salesman_name VARCHAR2(30),
sales_amount NUMBER(10), week_no NUMBER(2))
PARTITION BY HASH(salesman_id)
PARTITIONS 4
STORE IN (data1, data2, data3, data4)
```

Indexes

- Bitmap Indexes
  - Reduced response time for large queries
  - Reduced storage requirements compared to other indexing techniques
  - Most often used in data warehouse applications
- B-tree Indexes
  - Most effective for high cardinality data
  - Used only for unique columns in data warehouse applications

Aggregated Tables

- In addition to base fact and dimension tables, data warehouse keeps aggregated (summary) data for efficiency.
- Two approaches
  - store as separate summary fact and dimension tables
  - add to the existing base tables

Aggregated Tables

- Separate sum-table

Sales table

RID	City	Amount
1	Athens	$100
2	N.Y.	$300
3	Rome	$120
4	Athens	$250
5	Rome	$180
6	Rome	$65
7	N.Y.	$450

City-dimension sum table

City	Amount
Athens	$350
N.Y.	$750
Rome	$365

- Extend existing base tables

Extended Sales table

RID	City	Amount	Level
1	Athens	$100	NULL
2	N.Y.	$300	NULL
3	Rome	$120	NULL
4	Athens	$250	NULL
5	Rome	$180	NULL
6	Rome	$65	NULL
7	N.Y.	$450	NULL
8	Athens	$350	City
9	N.Y.	$750	City
10	Rome	$365	City

Materialized Views

- Summaries or Aggregate tables improve query execution times by pre-calculating expensive join and aggregation operations and storing the results in a table in the database.
- e.g. Create a table to contain the sums of sales by region and product.
- In Oracle you create a summary or aggregate table using a schema object called a materialized view.

Query Rewrite

- The query optimiser recognises when an existing materialized view can and should be used.
- The query is then transparently rewritten to use the materialized view.
- Rewriting queries to use materialized views instead of detail tables improves response time.

Materialized Views Behave Like Indexes

- The purpose is to increase query performance.
- Existence of a materialized view is transparent to SQL applications.
- They consume storage space.
- They must be updated when the underlying detail tables are modified.

Materialized View Example

```
CREATE MATERIALIZED VIEW product_sales_mv
PCTFREE 0 TABLESPACE demo
STORAGE (initial 8k next 8k pctincrease 0)
BUILD IMMEDIATE
REFRESH FAST
ENABLE QUERY REWRITE
AS SELECT p.prod_name, SUM(amount) AS
dollar_sales, COUNT(*) AS cnt,
COUNT(amount) AS cnt_amt
FROM sales s, products p
WHERE s.prod_id = p.prod_id
GROUP BY prod_name;
```

References

- Elmasri, Navathe; *Fundamentals of Database Systems; 4th Ed.* Chapter 28.
- *Oracle 10g Data Warehousing Guide*.