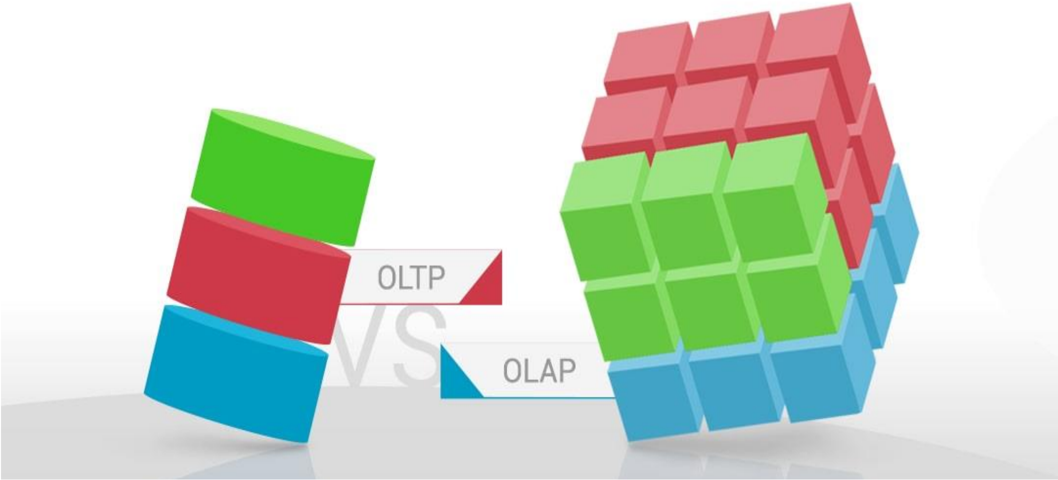
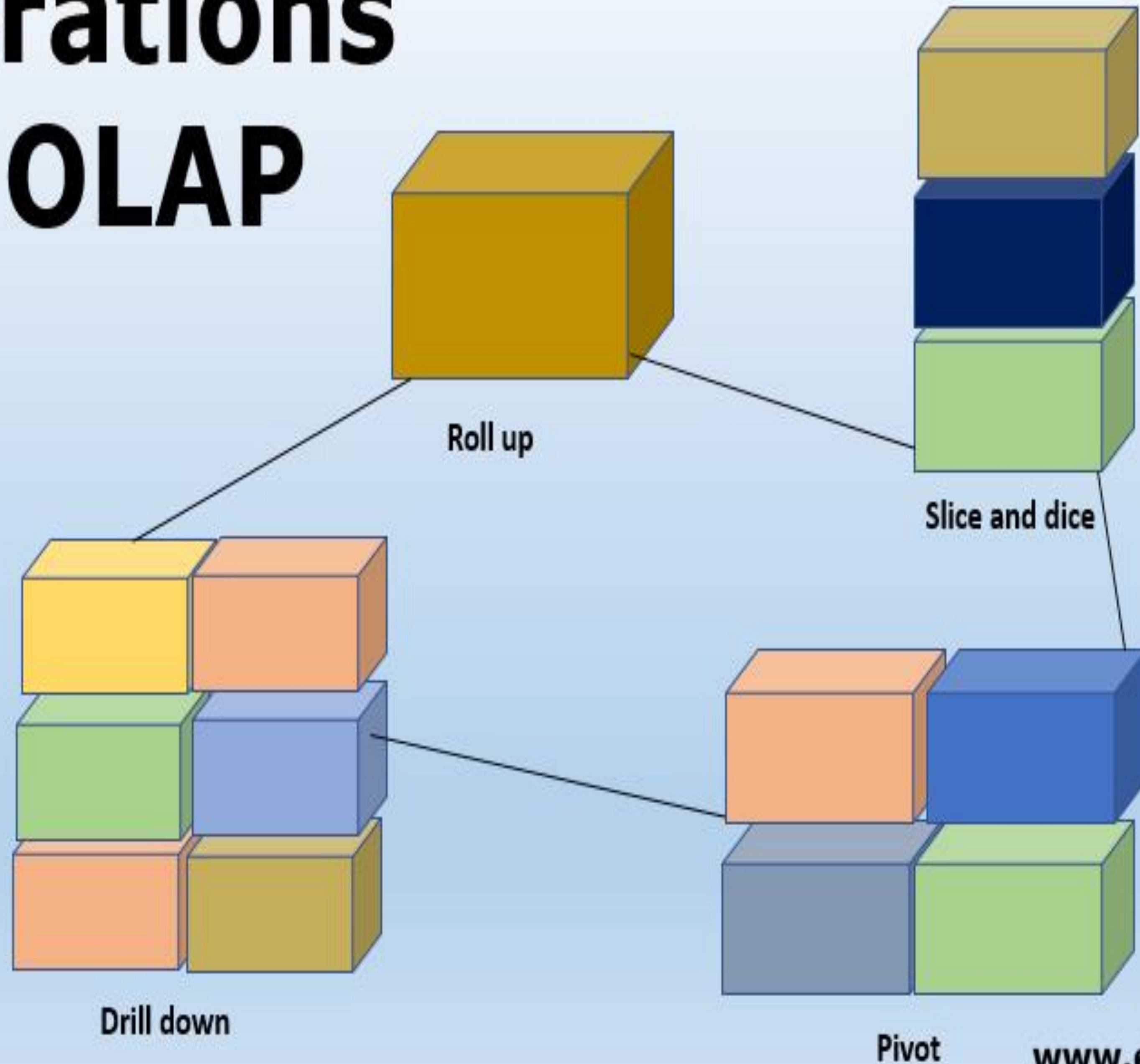
OLAP Operations



Operations in OLAP



OLTP Compared With OLAP

- On Line Transaction Processing – *OLTP*
  - Maintains a database that is an accurate model of some real-world enterprise. Supports **day-to-day operations**. Characteristics:
    - Short simple transactions
    - Relatively frequent updates
    - Transactions access only a small fraction of the database
- On Line Analytic Processing – *OLAP*
  - Uses information in database to guide **strategic decisions**. Characteristics:
    - Complex queries
    - Infrequent updates
    - Transactions access a large fraction of the database
    - Data need not be up-to-date

The Internet Grocer

- OLTP-style transaction:
  - Ujwala, from JioMart just bought a box of tomatoes; charge his account; deliver the tomatoes from our Schenectady warehouse; decrease our inventory of tomatoes from that warehouse
- OLAP-style transaction:
  - How many cases of tomatoes were sold in all northeast warehouses in the years 2000 and 2001?

OLAP: Traditional Compared with Newer Applications

- Traditional OLAP queries
  - Uses data the enterprise gathers in its usual activities, perhaps in its OLTP system
  - Queries are ad hoc, perhaps designed and carried out by non-professionals (managers)
- Newer Applications (e.g., Internet companies)
  - Enterprise actively gathers data it wants, perhaps purchasing it
  - Queries are sophisticated, designed by professionals, and used in more sophisticated ways

The Internet Grocer

- Traditional
  - How many cases of tomatoes were sold in all northeast warehouses in the years 2020 and 2022?
- Newer
  - Prepare a profile of the grocery purchases of Ujwala for the years 2020 and 2022 (so that we can customize our marketing to him and get more of his business)

Data Mining

- *Data Mining* is an attempt at knowledge discovery – to extract knowledge from a database
- Comparison with OLAP
  - *OLAP*:
    - What percentage of people who make over $50,000 defaulted on their mortgage in the year 2000?
  - *Data Mining*:
    - How can information about salary, net worth, and other historical data be used to *predict* who will default on their mortgage?

Data Warehouses

- OLAP and data mining databases are frequently stored on special servers called ***data warehouses***:
  - Can accommodate the huge amount of data generated by OLTP systems
  - Allow OLAP queries and data mining to be run off-line so as not to impact the performance of OLTP

OLAP, Data Mining, and Analysis

- The “A” in OLAP stands for “**Analytical**”
- Many OLAP and Data Mining applications involve sophisticated analysis methods from the fields of **mathematics, statistical analysis, and artificial intelligence**
- Our main interest is in the database aspects of these fields, not the sophisticated analysis techniques

Fact Tables

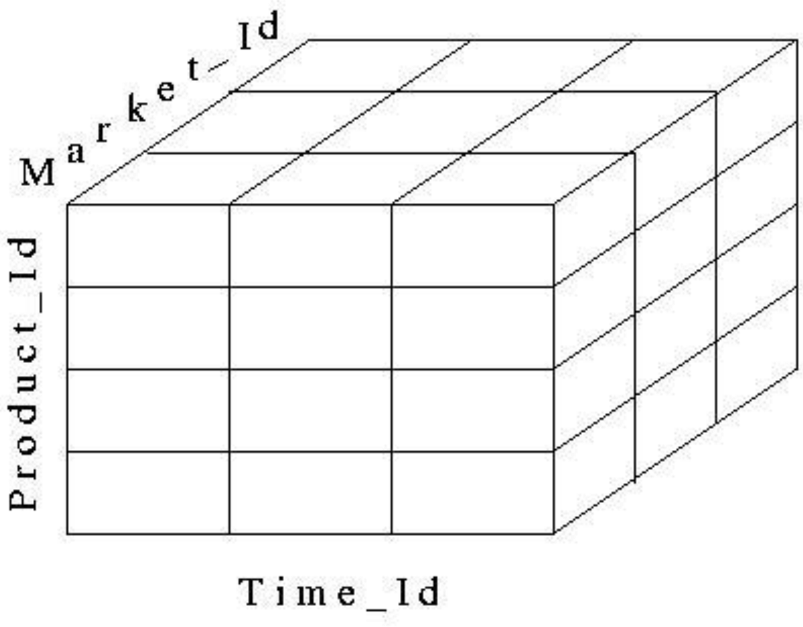
- Many OLAP applications are based on a ***fact table***
- For example, a supermarket application might be based on a table

  Sales (*Market_Id*, *Product_Id*, *Time_Id*, *Sales_Amt*)

- The table can be viewed as *multidimensional*
  - *Market_Id*, *Product_Id*, *Time_Id* are the dimensions that represent specific supermarkets, products, and time intervals
  - *Sales_Amt* is a function of the other three

A Data Cube

- Fact tables can be viewed as an N-dimensional ***data cube*** (3-dimensional in our example)
  - The entries in the cube are the values for *Sales_Amts*



Dimension Tables

- The dimensions of the fact table are further described with ***dimension tables***
- Fact table:

    Sales (*Market_id*, *Product_Id*, *Time_Id*, *Sales_Amt*)

- Dimension Tables:

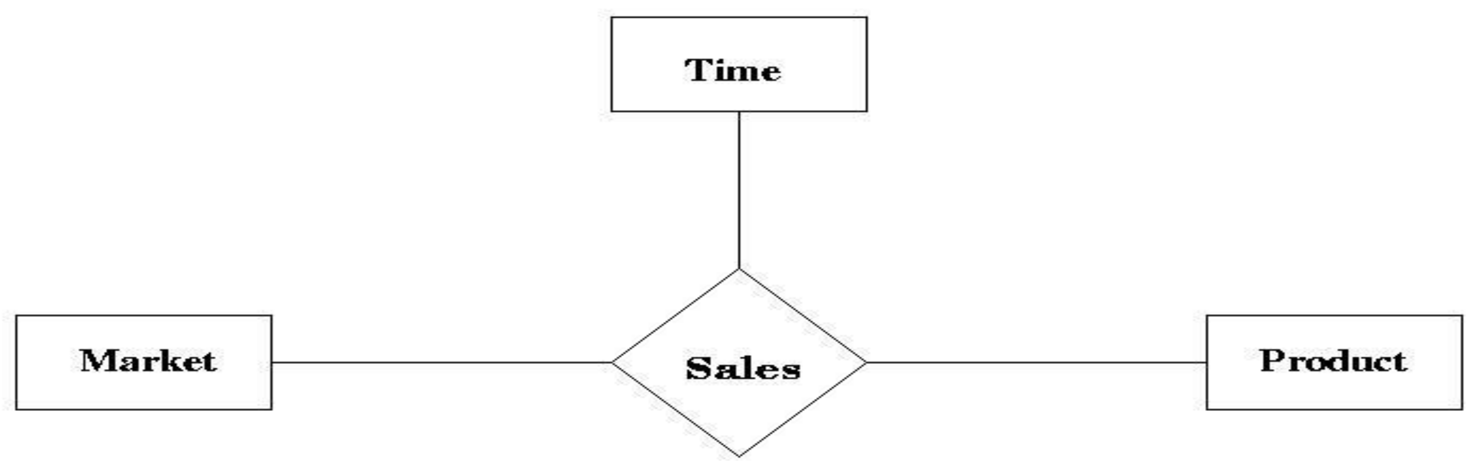
    Market (*Market_Id*, *City*, *State*, *Region*)

    Product (*Product_Id*, *Name*, *Category*, *Price*)

    Time (*Time_Id*, *Week*, *Month*, *Quarter*)

Star Schema

- The fact and dimension relations can be displayed in an E-R diagram, which looks like a star and is called a ***star schema***



Aggregation

- Many OLAP queries involve ***aggregation*** of the data in the fact table
- For example, to find the total sales (over time) of each product in each market, we might use

```
SELECT    S.Market_Id, S.Product_Id, SUM (S.Sales_Amt)
FROM      Sales S
GROUP BY  S.Market_Id, S.Product_Id
```

- The aggregation is over the entire time dimension and thus produces a two-dimensional view of the data

Aggregation over Time

- The output of the previous query

Market_Id

Product_Id

SUM(*Sales_Amt*)	M1	M2	M3	M4
P1	3003	1503	…	
P2	6003	2402	…	
P3	4503	3	…	
P4	7503	7000	…	
P5	…	…	…	

Drilling Down and Rolling Up

- Some dimension tables form an ***aggregation hierarchy***

  $$Market\_Id \rightarrow City \rightarrow State \rightarrow Region$$

- Executing a series of queries that **moves down a hierarchy** (*e.g.*, from aggregation over regions to that over states) is called ***drilling down***
  - Requires the **use of the fact table** or information more specific than the requested aggregation (*e.g.*, cities)
- Executing a series of queries that **moves up the hierarchy** (e.g., from states to regions) is called ***rolling up***
  - Note: In a rollup, coarser aggregations can be computed using prior queries for finer aggregations

Drilling Down

- Drilling down on market: from *Region* to *State*

  Sales (*Market_Id*, *Product_Id*, *Time_Id*, *Sales_Amt*)

  Market (*Market_Id*, *City*, *State*, *Region*)

1. SELECT S.*Product_Id*, M.*Region*, SUM (S.*Sales_Amt*)
   FROM Sales S, Market M
   WHERE M.*Market_Id* = S.*Market_Id*
   GROUP BY S.*Product_Id*, M.*Region*

2. SELECT S.*Product_Id*, M.*State*, SUM (S.*Sales_Amt*)
   FROM Sales S, Market M
   WHERE M.*Market_Id* = S.*Market_Id*
   GROUP BY S.*Product_Id*, M.*State*,

Rolling Up

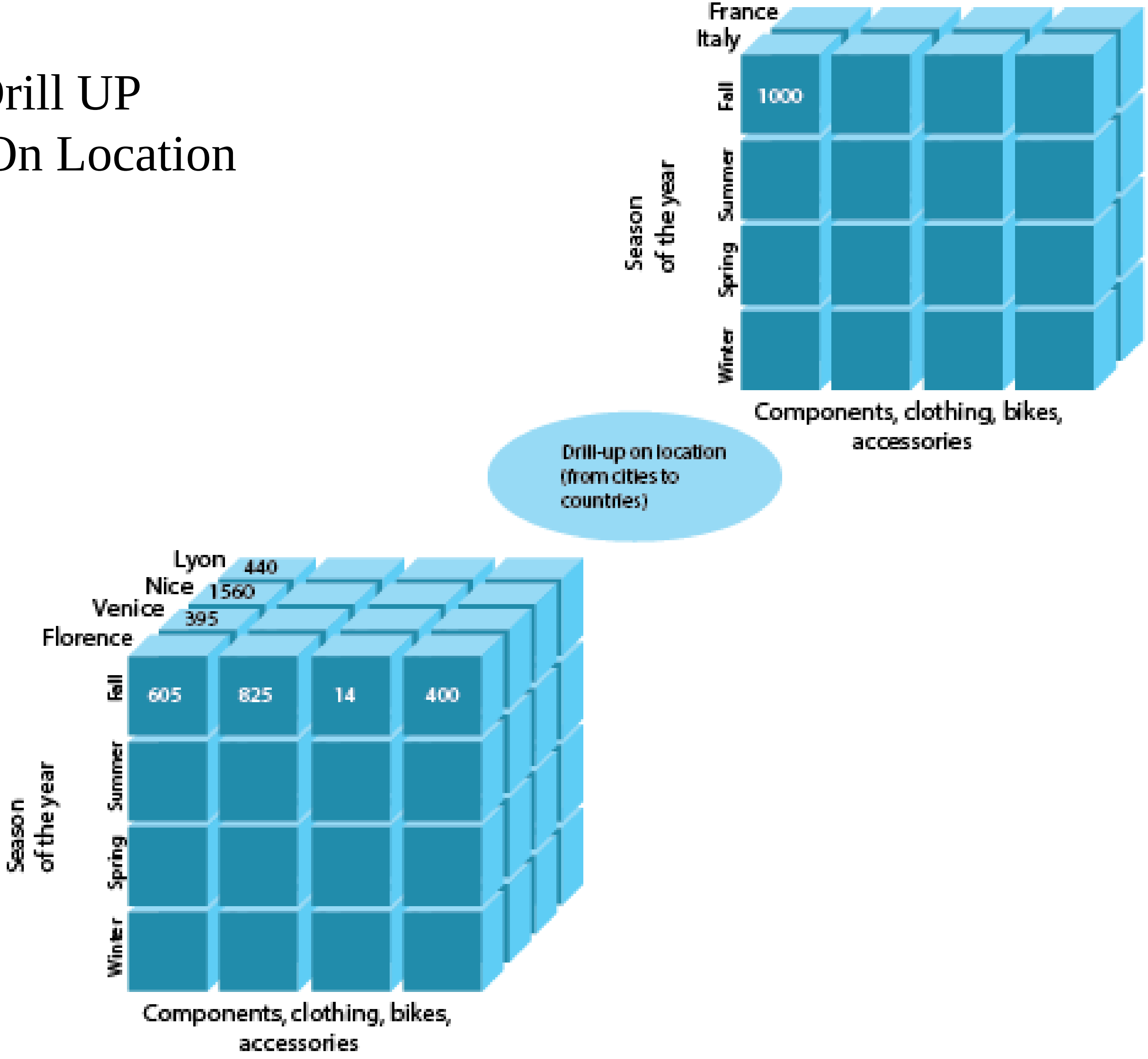
- Rolling up on market, from *State* to *Region*
  - If we have already created a table, State_Sales, using

```
1.  SELECT     S.Product_Id, M.State, SUM (S.Sales_Amt)
    FROM       Sales S, Market M
    WHERE      M.Market_Id = S.Market_Id
    GROUP BY   S.Product_Id, M.State
```

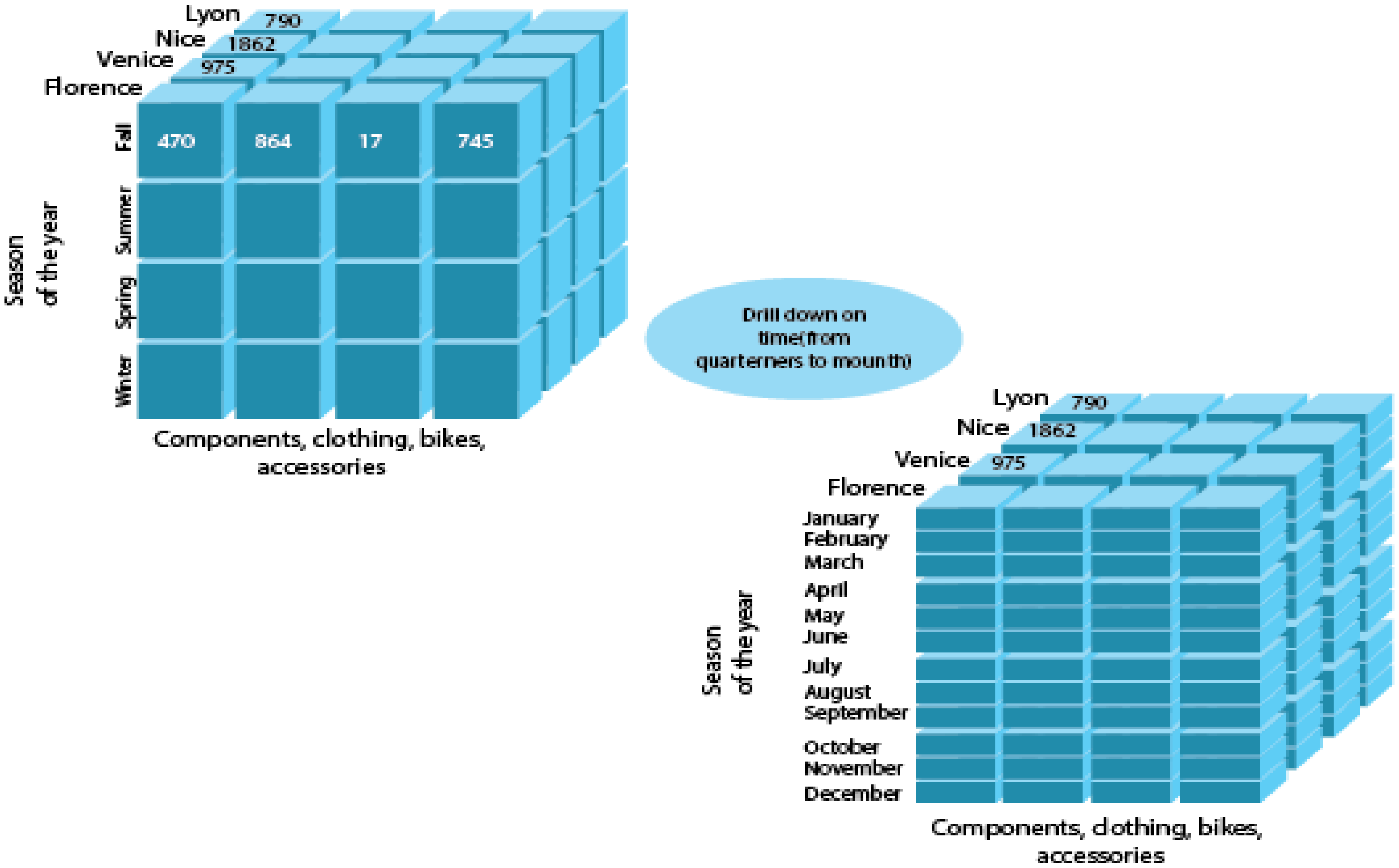
then we can roll up from there to:

```
2.  SELECT     T.Product_Id, M.Region, SUM (T.Sales_Amt)
    FROM       State_Sales T, Market M
    WHERE      M.State = T.State
    GROUP BY   T.Product_Id, M.Region
```

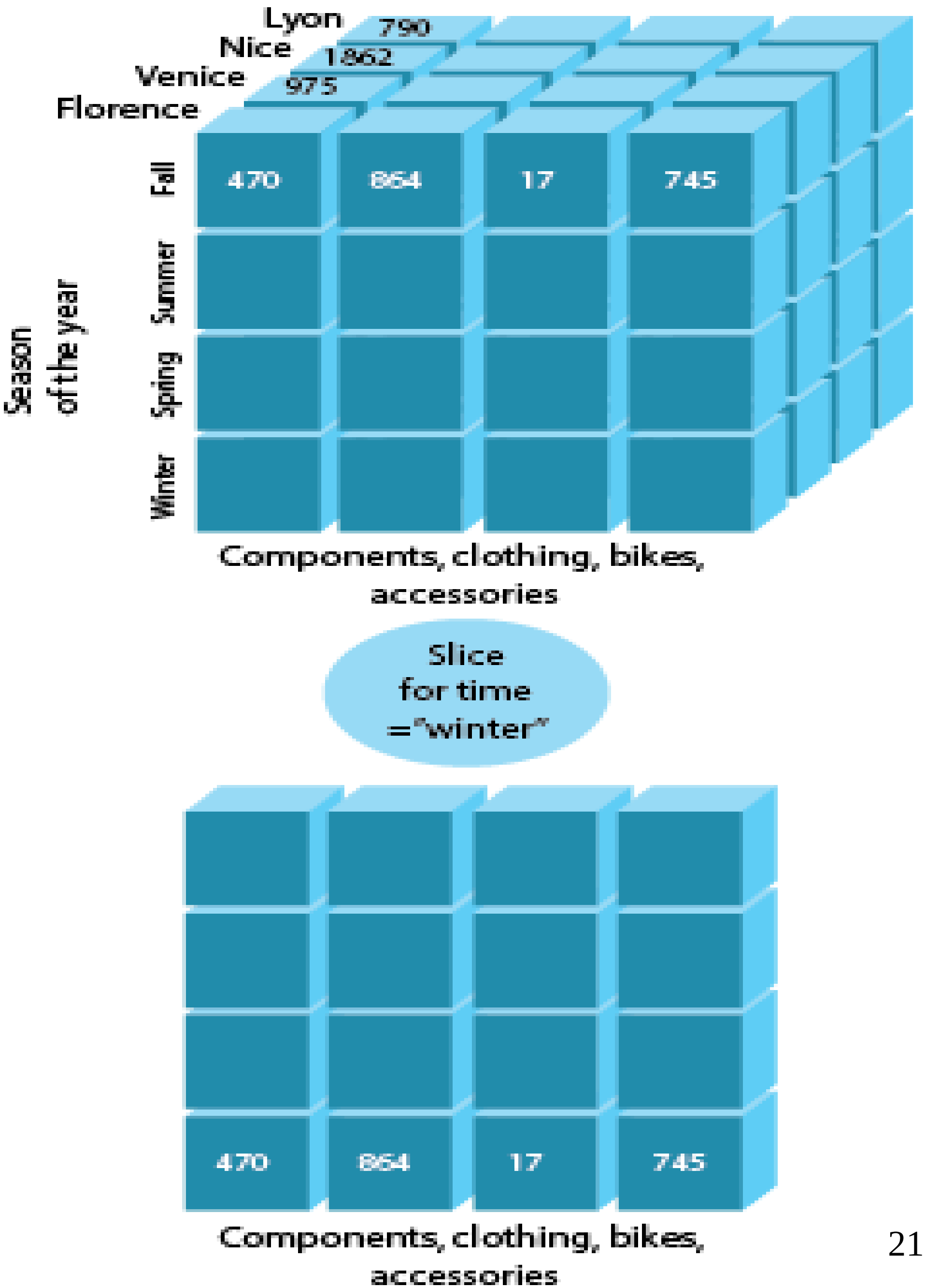
Drill UP
On Location



Drill Down On Time

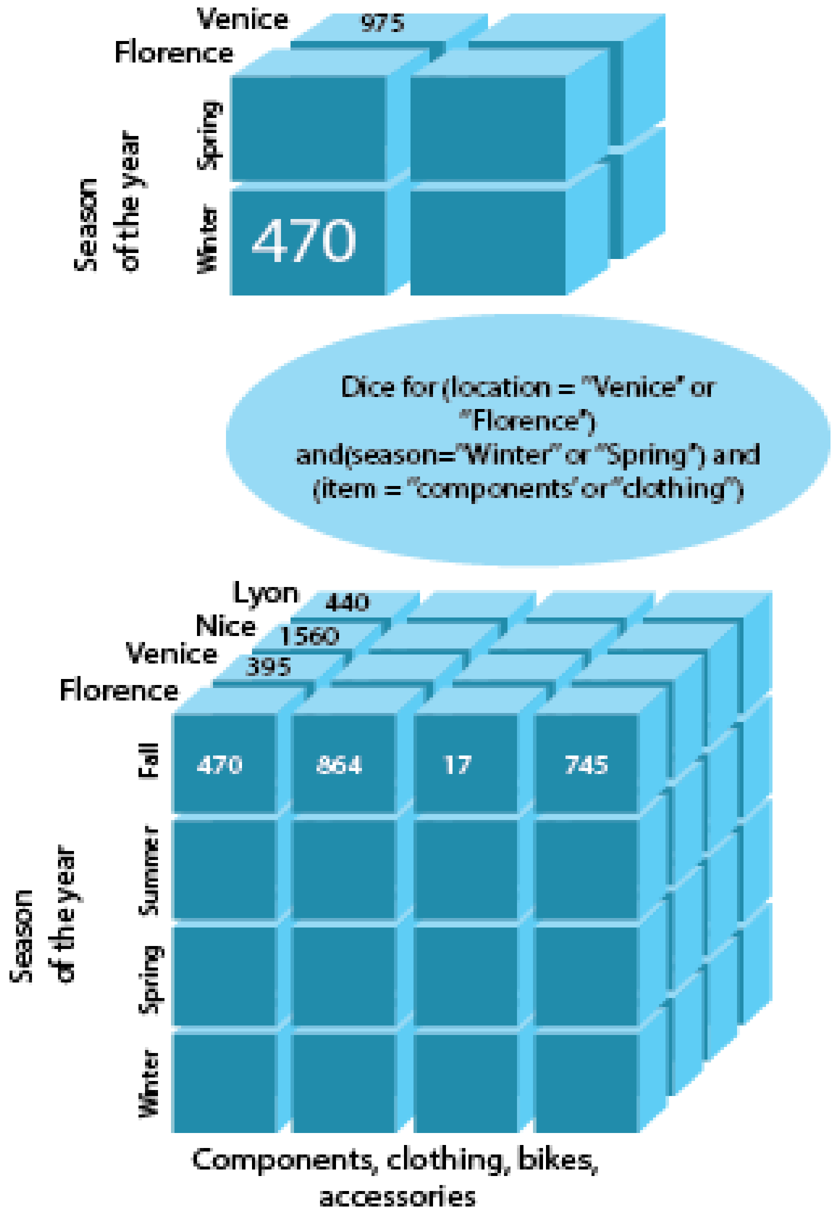


- Slice



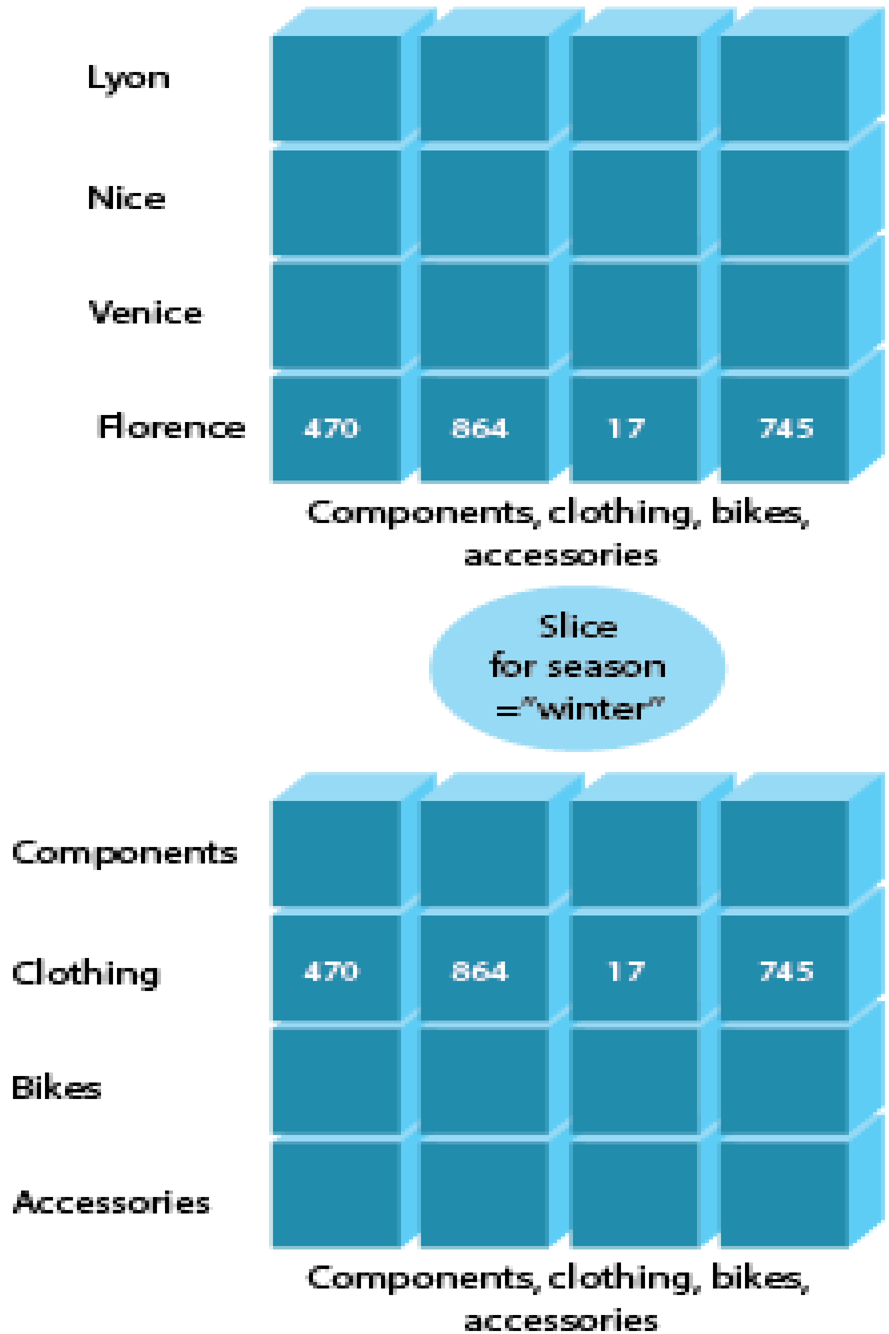
Lyon
Nice
Venice
Florence
790
1862
975
Season of the year
Fall
Summer
Spring
Winter
470
864
17
745
Components, clothing, bikes, accessories
Slice for time ="winter"
470
864
17
745
Components, clothing, bikes, accessories

- Dice
- OLAP Dice emphasizes two or more dimensions from a cube given and suggests a new sub-cube,



- **What is difference between slice and dice in OLAP?**
- The Slice operation takes one specific dimension from a cube given and represents a new sub-cube which provides information from another point of view. The Dice operation in the contrary emphasizes two or more dimensions from a cube.

- Pivot
- This OLAP operation rotates the axes of a cube to provide an alternative view of the data cube. Pivot clusters the data with other dimensions which helps analyze the performance of a company or enterprise.



Pivoting

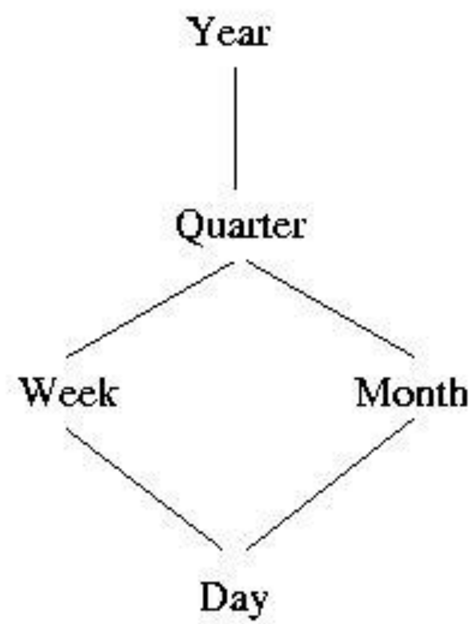
- When we view the data as a multi-dimensional cube and group on a subset of the axes, we are said to be performing a *pivot* on those axes
  - Pivoting on dimensions $D_1,\ldots,D_k$ in a data cube $D_1,\ldots,D_k,D_{k+1},\ldots,D_n$ means that we use GROUP BY $A_1,\ldots,A_k$ and aggregate over $A_{k+1},\ldots A_n$, where A_i is an attribute of the dimension D_i
  - *Example*: Pivoting on Product and Time corresponds to grouping on *Product_id* and *Quarter* and aggregating *Sales_Amt* over *Market_id:*

```
SELECT     S.Product_Id, T.Quarter, SUM (S.Sales_Amt)
FROM       Sales S, Time T
WHERE      T.Time_Id = S.Time_Id
GROUP BY   S.Product_Id, T.Quarter
```

Pivot

Time Hierarchy as a Lattice

- Not all aggregation hierarchies are linear
  - The time hierarchy is a lattice
    - Weeks are not contained in months
    - We can roll up days into weeks or months, but we can only roll up weeks into quarters



Slicing-and-Dicing

- When we use WHERE to specify a particular value for an axis (or several axes), we are performing a ***slice***
  - Slicing the data cube in the Time dimension (choosing sales only in week 12) then pivoting to *Product_id* (aggregating over *Market_id*)

SELECT S.*Product_Id*, SUM (*Sales_Amt*)

FROM Sales S, Time T

WHERE T.*Time_Id* = S.*Time_Id* AND T.*Week* = 'Wk-12'

GROUP BY S. *Product_Id*

Slice

Pivot

Slicing-and-Dicing

- Typically slicing and dicing involves several queries to find the "right slice."

  For instance, change the slice and the axes:
  - Slicing on Time and Market dimensions then pivoting to *Product_id* and *Week* (in the time dimension)

```
SELECT     S.Product_Id,  T.Quarter,  SUM (Sales_Amt)
FROM       Sales S,  Time T
WHERE      T.Time_Id = S.Time_Id
               AND  T.Quarter =  4
               AND  S.Market_id = 12345
GROUP BY  S.Product_Id,  T.Week
```

Slice

Pivot

The CUBE Operator

- To construct the following table, would take 3 queries (next slide)

Market_Id

Product_Id

SUM(*Sales_Amt*)	M1	M2	M3	*Total*
P1	3003	1503	…	…
P2	6003	2402	…	…
P3	4503	3	…	…
P4	7503	7000	…	…
Total	…	…	…	…

The Three Queries

- For the table entries, without the totals (aggregation on time)

  SELECT S.*Market_Id*, S.*Product_Id*, SUM (S.*Sales_Amt*)
  FROM Sales S
  GROUP BY S.Market_Id, S.Product_Id

- For the row totals (aggregation on time and supermarkets)

  SELECT S.*Product_Id*, SUM (S.*Sales_Amt*)
  FROM Sales S
  GROUP BY S.*Product_Id*

- For the column totals (aggregation on time and products)

  SELECT S.*Market_Id*, SUM (S.*Sales*)
  FROM Sales S
  GROUP BY S.*Market_Id*

Definition of the CUBE Operator

- Doing these three queries is wasteful
  - The first does much of the work of the other two: if we could save that result and aggregate over *Market_Id* and *Product_Id*, we could compute the other queries more efficiently
- The CUBE clause is part of SQL:1999
  - GROUP BY CUBE (v1, v2, …, vn)
  - Equivalent to a collection of GROUP BYs, one for each of the 2^n subsets of v1, v2, …, vn

Example of CUBE Operator

- The following query returns all the information needed to make the previous products/markets table:

```
SELECT  S.Market_Id, S.Product_Id, SUM (S.Sales_Amt)
FROM  Sales S
GROUP BY CUBE (S.Market_Id, S.Product_Id)
```

ROLLUP

- ROLLUP is similar to CUBE except that instead of aggregating over all subsets of the arguments, it creates subsets moving from right to left
- GROUP BY ROLLUP $(A_1, A_2, \ldots, A_n)$ is a series of these aggregations:
  - GROUP BY $A_1, \ldots, A_{n-1}, A_n$
  - GROUP BY $A_1, \ldots, A_{n-1}$
  - … … …
  - GROUP BY A_1, A_2
  - GROUP BY A_1
  - *No* GROUP BY
- ROLLUP is also in SQL:1999

Example of ROLLUP Operator

SELECT S.*Market_Id*, S.*Product_Id*, SUM (S.*Sales_Amt*)
FROM Sales S
GROUP BY ROLLUP (S.*Market_Id*, S. *Product_Id*)

– first aggregates with the finest granularity:
  GROUP BY S.*Market_Id*, S.*Product_Id*
– then with the next level of granularity:
  GROUP BY S.*Market_Id*
– then the grand total is computed with *no* GROUP BY clause

ROLLUP vs. CUBE

- The same query with CUBE:

  - first aggregates with the finest granularity:

    GROUP BY S.*Market_Id*, S.*Product_Id*

  - then with the next level of granularity:

    GROUP BY S.*Market_Id*

    and

    GROUP BY S.*Product_Id*

  - then the grand total with *no* GROUP BY

Materialized Views

The CUBE operator is often used to precompute aggregations on all dimensions of a fact table and then save them as a *materialized views* to speed up future queries

OLAP Server

- Relational OLAP (ROLAP)
- Multidimensional OLAP (MOLAP)
- Hybrid OLAP (HOLAP)
- Specialized SQL Servers

ROLAP and MOLAP

- Relational OLAP: ROLAP
  - OLAP data is stored in a relational database as previously described. Data cube is a conceptual view – way to *think about* a fact table
- Multidimensional OLAP: MOLAP
  - Vendor provides an OLAP server that *implements* a fact table as a data cube using a special multi-dimensional (non-relational) data structure

MOLAP

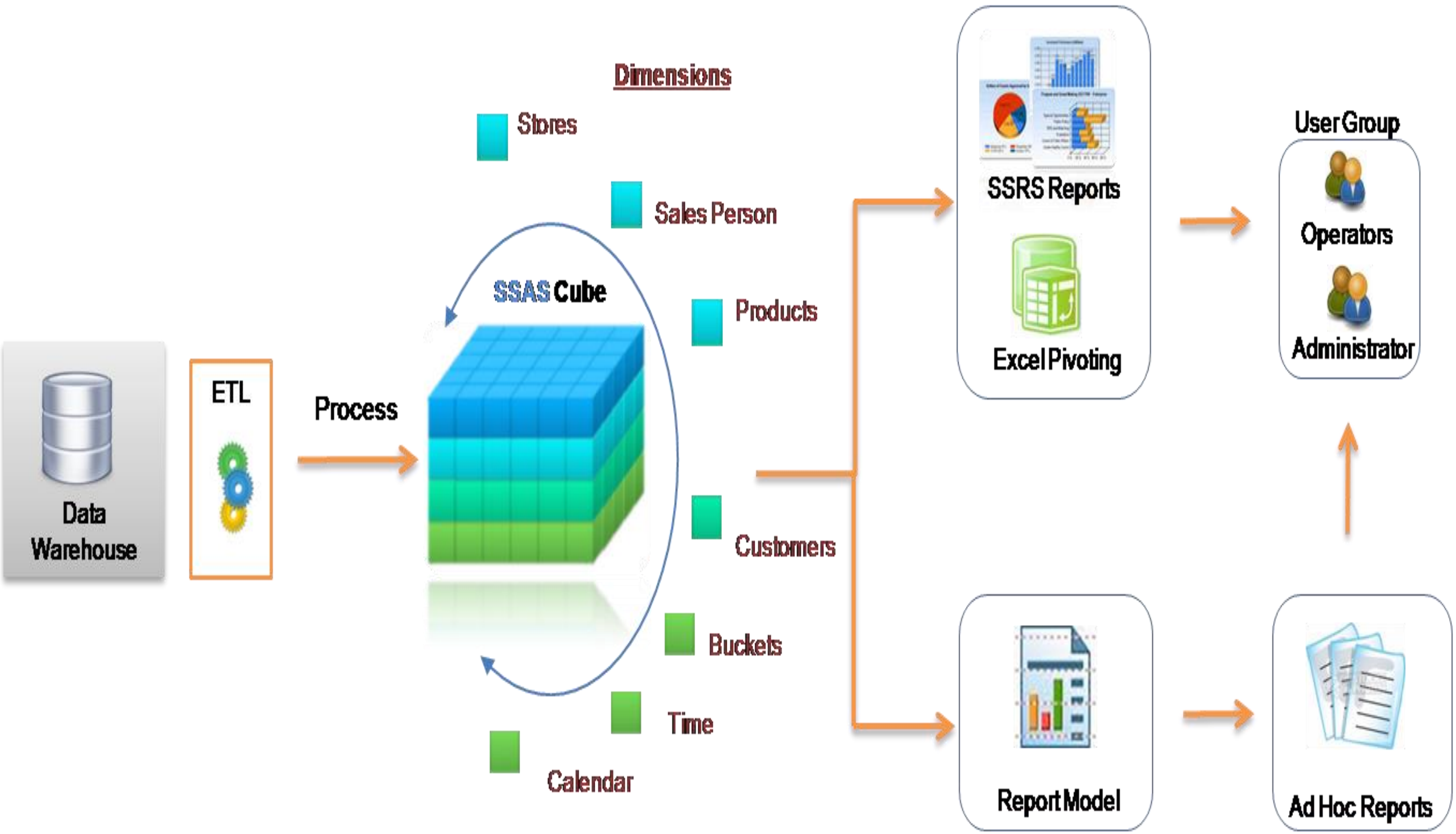
- No standard query language for MOLAP databases
- Many MOLAP vendors (and many ROLAP vendors) provide proprietary visual languages that allow casual users to make queries that involve pivots, drilling down, or rolling up

Online Analytical Processing Tools

- MOLAP - Multi-dimensional Online Analytical Processing
- MOLAP is the classic form of OLAP
- Optimized multi dimensional array storage
- Pre-computation

- ROLAP- Relational Online Analytical Processing
- ROLAP stores the data in relational databases
- Specialized schema design
- Base data and the dimension tables are stored as relational tables

- HOLAP-Hybrid Online Analytical Processing
- HOLAP combines the capabilities of MOLAP and ROLAP
- Database will divide data between relational and specialized storage



Data Warehouse
ETL
Process
SSAS Cube
Dimensions
Stores
Sales Person
Products
Customers
Buckets
Time
Calendar
SSRS Reports
Excel Pivoting
Report Model
User Group
Operators
Administrator
Ad Hoc Reports

Implementation Issues

- OLAP applications are characterized by a very large amount of data that is relatively static, with infrequent updates
  - Thus, various aggregations can be precomputed and stored in the database
  - *Star joins*, *join indices*, and *bitmap indices* can be used to improve efficiency (recall the methods to compute star joins in Chapter 14)
  - Since updates are infrequent, the inefficiencies associated with updates are minimized

Loading Data into A Data Warehouse