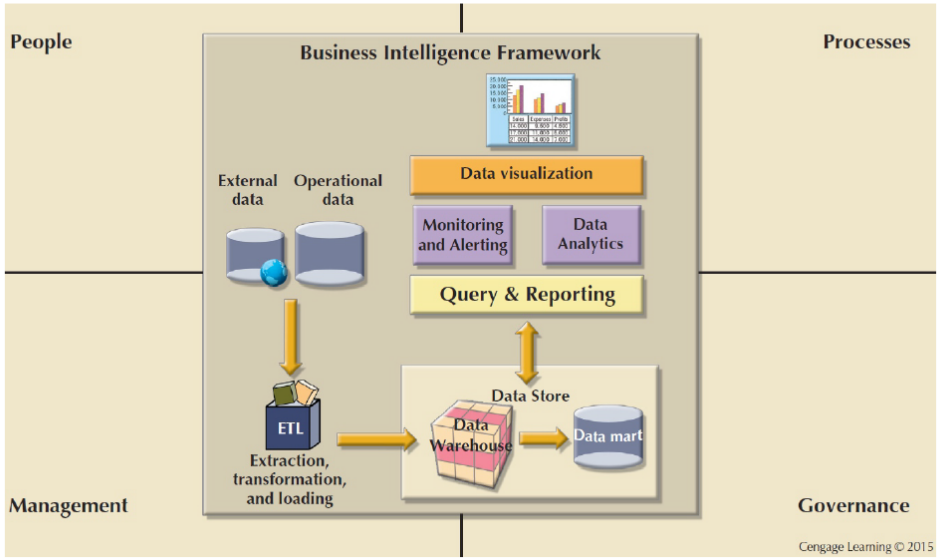
Business Intelligence

("BI")

Business Intelligence (BI)

- Comprehensive, cohesive, integrated set of tools and processes
  - Captures, collects, integrates, stores, and analyzes data
- Purpose - Generate and present information to support business decision making
- Allows a business to transform:
  - Data into information
  - Information into knowledge
  - Knowledge into wisdom

Figure 13.1 - Business Intelligence Framework



Business Intelligence Benefits

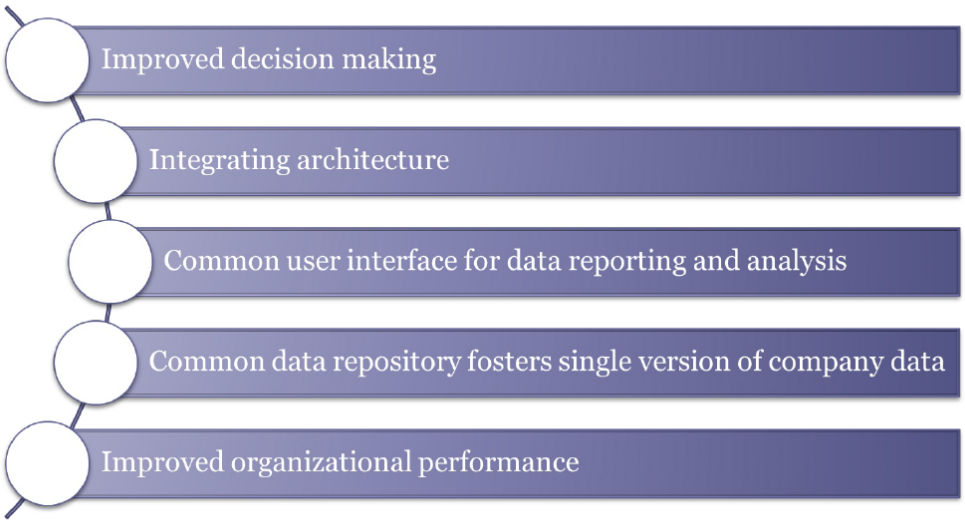
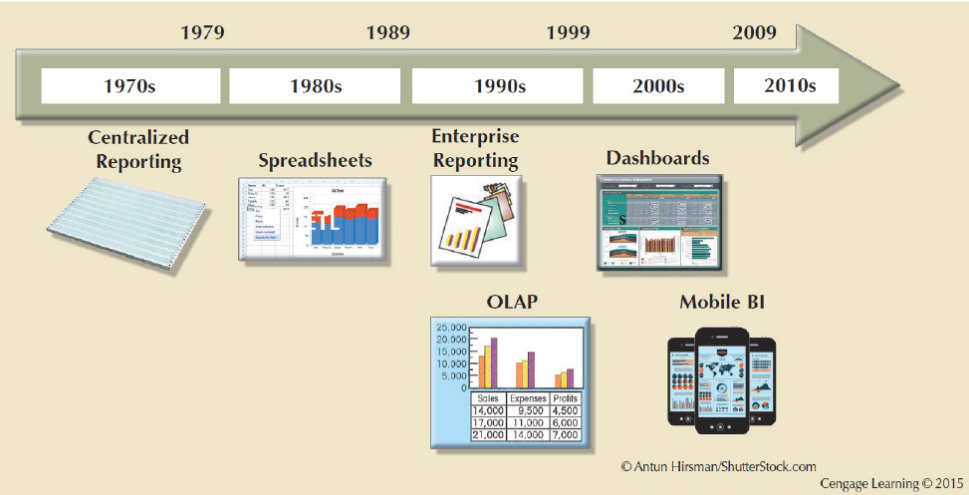
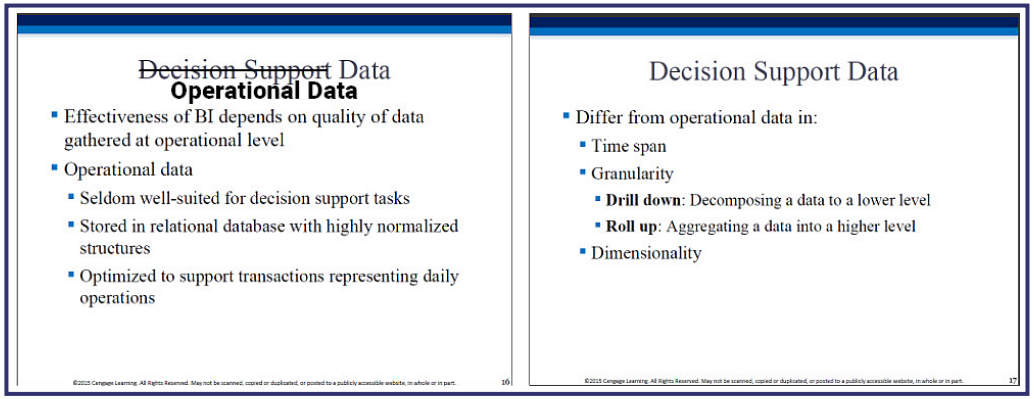


Figure 13.3 - Evolution of BI Information Dissemination Formats





Transactional, operational data: individual units.

'Analytical', decision-support data: aggregated.

Table 13.5 - Contrasting Operational and Decision Support Data Characteristics

CHARACTERISTIC	OPERATIONAL DATA	DECISION SUPPORT DATA
Data currency	Current operations Real-time data	Historic data Snapshot of company data Time component (week/month/year)
Granularity	Atomic-detailed data	Summarized data
Summarization level	Low; some aggregate yields	High; many aggregation levels
Data model	Highly normalized Mostly relational DBMSs	Non-normalized Complex structures Some relational, but mostly multidimensional DBMSs
Transaction type	Mostly updates	Mostly query
Transaction volumes	High-update volumes	Periodic loads and summary calculations
Transaction speed	Updates are critical	Retrievals are critical
Query activity	Low to medium	High
Query scope	Narrow range	Broad range
Query complexity	Simple to medium	Very complex
Data volumes	Hundreds of gigabytes	Terabytes to petabytes

Decision Support Database Requirements

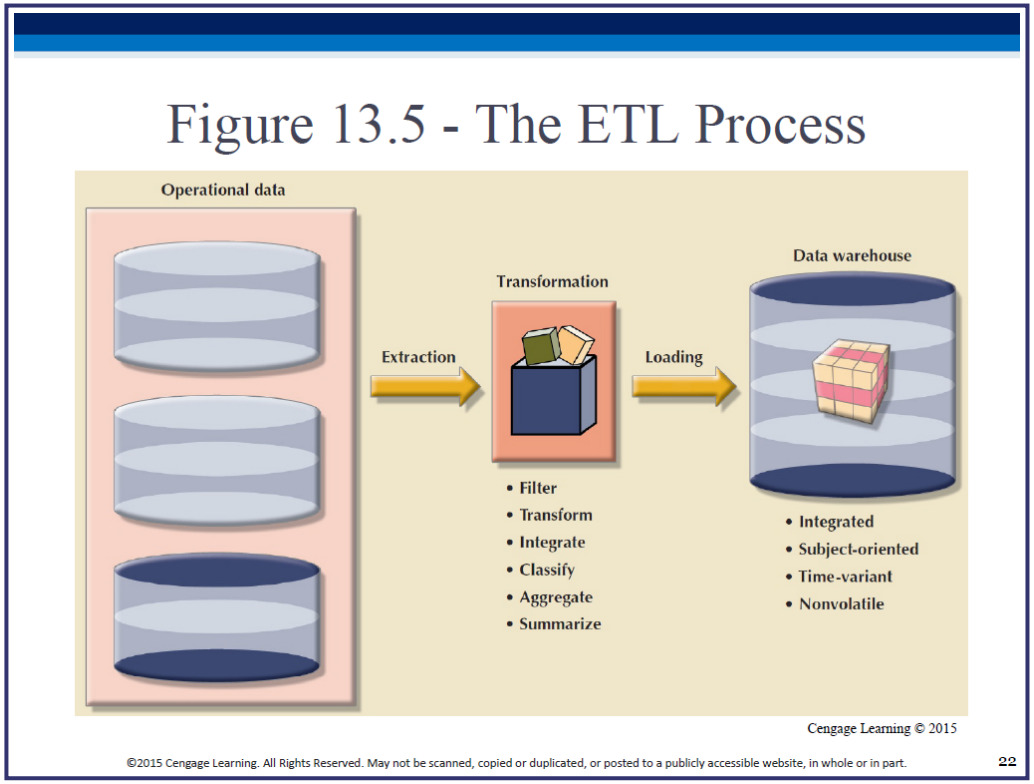
- Database schema
  - Must support complex, non-normalized data representations
  - Data must be aggregated and summarized
  - Queries must be able to extract multidimensional time slices

Decision Support Database Requirements

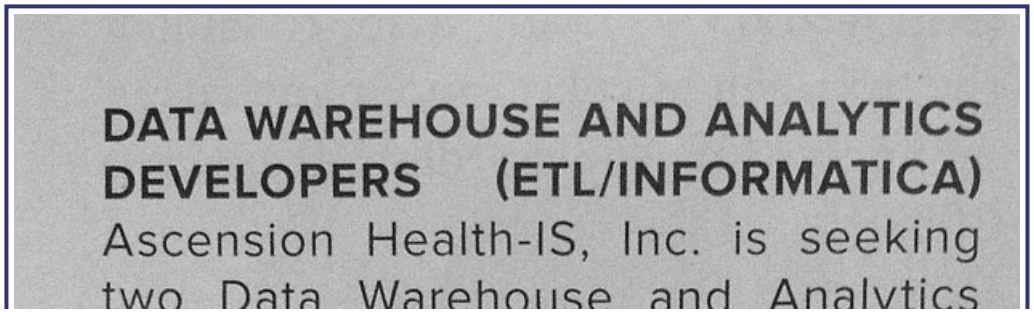
- Data extraction and loading
  - Allow batch and scheduled data extraction
  - Support different data sources and check for inconsistent data or data validation rules
  - Support advanced integration, aggregation, and classification
- Database size should support:
  - **Very large databases (VLDBs)**
  - Advanced storage technologies
  - Multiple-processor technologies

Table 13.8 - Characteristics of Data Warehouse Data and Operational Database Data

CHARACTERISTIC	OPERATIONAL DATABASE DATA	DATA WAREHOUSE DATA
Integrated	Similar data can have different representations or meanings. For example, Social Security numbers may be stored as ###-##-#### or as #########, and a given condition may be labeled as T/F or 0/1 or Y/N. A sales value may be shown in thousands or in millions.	Provide a unified view of all data elements with a common definition and representation for all business units.
Subject-oriented	Data are stored with a functional, or process, orientation. For example, data may be stored for invoices, payments, and credit amounts.	Data are stored with a subject orientation that facilitates multiple views of the data and decision making. For example, sales may be recorded by product, division, manager, or region.
Time-variant	Data are recorded as current transactions. For example, the sales data may be the sale of a product on a given date, such as $342.78 on 12-MAY-2014.	Data are recorded with a historical perspective in mind. Therefore, a time dimension is added to facilitate data analysis and various time comparisons.
Nonvolatile	Data updates are frequent and common. For example, an inventory amount changes with each sale. Therefore, the data environment is fluid.	Data cannot be changed. Data are added only periodically from historical systems. Once the data are properly stored, no changes are allowed. Therefore, the data environment is relatively static.



An ETL-related job ad:



Developers (ETL/Informatica) in St. Louis, Missouri to code design and development on the data warehouse/ analytics Extract Transform Load (ETL) toolset, Informatica PowerCenter; support Informatica toolset; integrate and develop other technologies. Research solutions and technology; participate in testing (e.g. user acceptance testing, unit, system, regression, integration testing); develop test plans and documentation; debug code. Contact Jenna Mihm, Vice President Legal Services & Associate General Counsel, Ascension Health, 4600 Edmundson Road, St. Louis, MO 63134, 314-733-8692, Jenna.Mihm@ascensionhealth.org To apply for this position, please reference Job Number 03.

Data Marts

- Small, single-subject data warehouse subset
- Provide decision support to a small group of people
- Benefits over data warehouses
  - Lower cost and shorter implementation time
  - Technologically advanced
  - Inevitable people issues

Table 13.9 - Twelve Rules for a Data Warehouse

RULE NO.	DESCRIPTION
1	The data warehouse and operational environments are separated.
2	The data warehouse data are integrated.
3	The data warehouse contains historical data over a long time.
4	The data warehouse data are snapshot data captured at a given point in time.
5	The data warehouse data are subject oriented.
6	The data warehouse data are mainly read-only with periodic batch updates from operational data. No online updates are allowed.

Cengage Learning © 2015

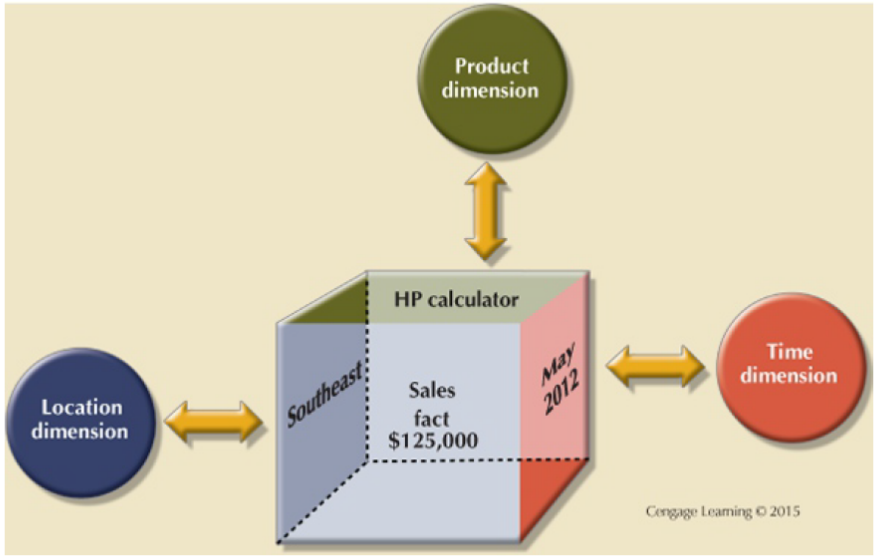
Table 13.9 - Twelve Rules for a Data Warehouse

RULE NO.	DESCRIPTION
7	The data warehouse development life cycle differs from classical systems development. Data warehouse development is data-driven; the classical approach is process-driven.
8	The data warehouse contains data with several levels of detail: current detail data, old detail data, lightly summarized data, and highly summarized data.
9	The data warehouse environment is characterized by read-only transactions to very large data sets. The operational environment is characterized by numerous update transactions to a few data entities at a time.
10	The data warehouse environment has a system that traces data sources, transformations, and storage.
11	The data warehouse's metadata are a critical component of this environment. The metadata identify and define all data elements. The metadata provide the source, transformation, integration, storage, usage, relationships, and history of each data element.
12	The data warehouse contains a chargeback mechanism for resource usage that enforces optimal use of the data by end users.

Cengage Learning © 2015

Star Schema

- Data-modeling technique
- Maps multidimensional decision support data into a relational database
- Creates the near equivalent of multidimensional database schema from existing relational database
- Yields an easily implemented model for multidimensional data analysis



Components of Star Schemas

Facts

- Numeric values that represent a specific business aspect

Dimensions

- Qualifying characteristics that provide additional perspectives to a given fact

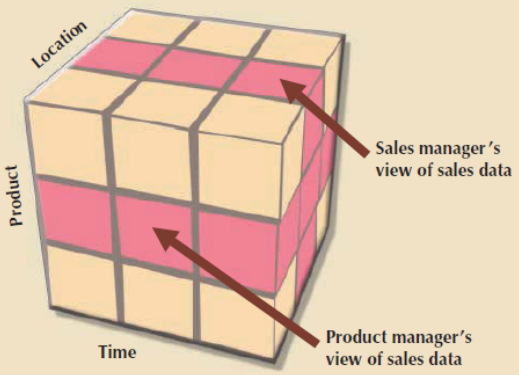
Attributes

- Used to search, filter, and classify facts
- **Slice and dice**: Ability to focus on slices of the data cube for more detailed analysis

Attribute hierarchy

- Provides a top-down data organization

FIGURE 13.8 Slice-and-dice view of sales



Sales manager's view of sales data

Product manager's view of sales data

FIGURE 13.9 Location attribute hierarchy

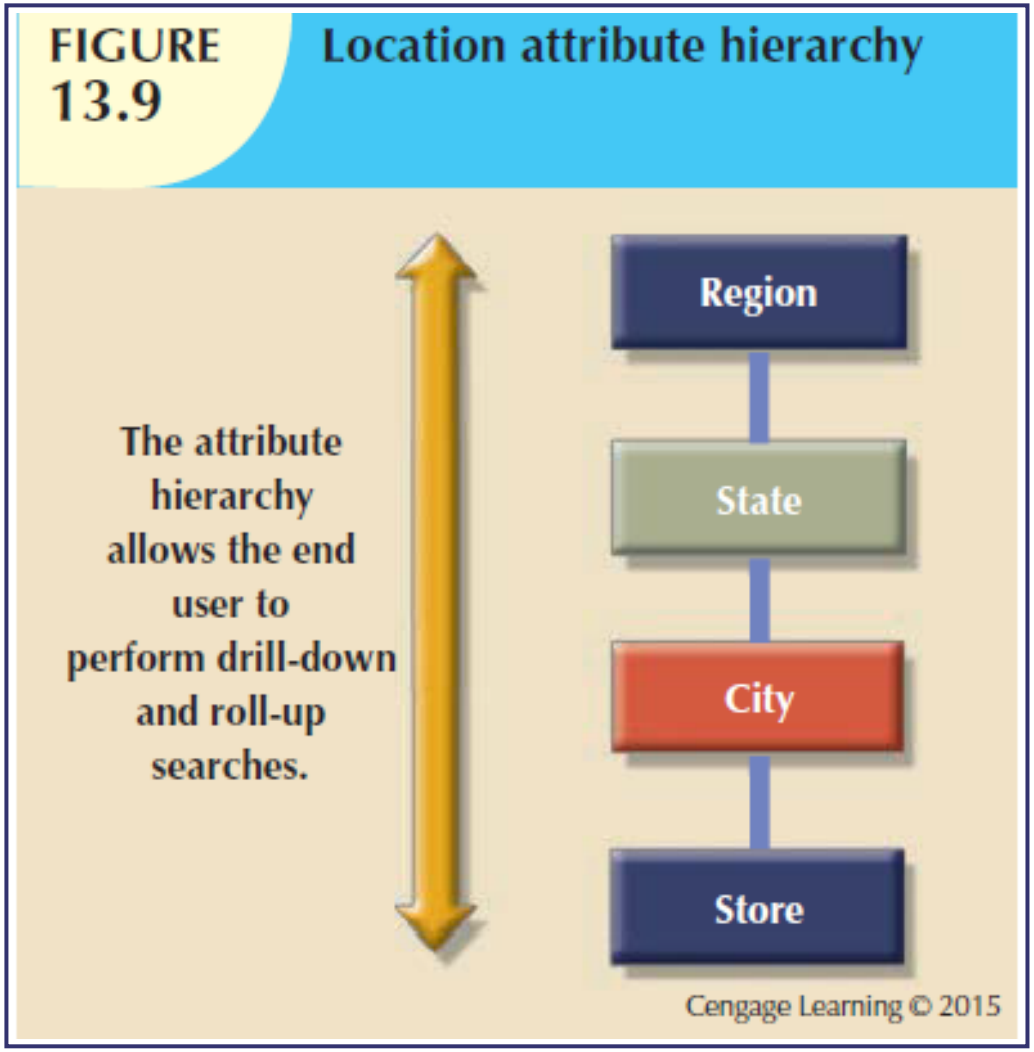
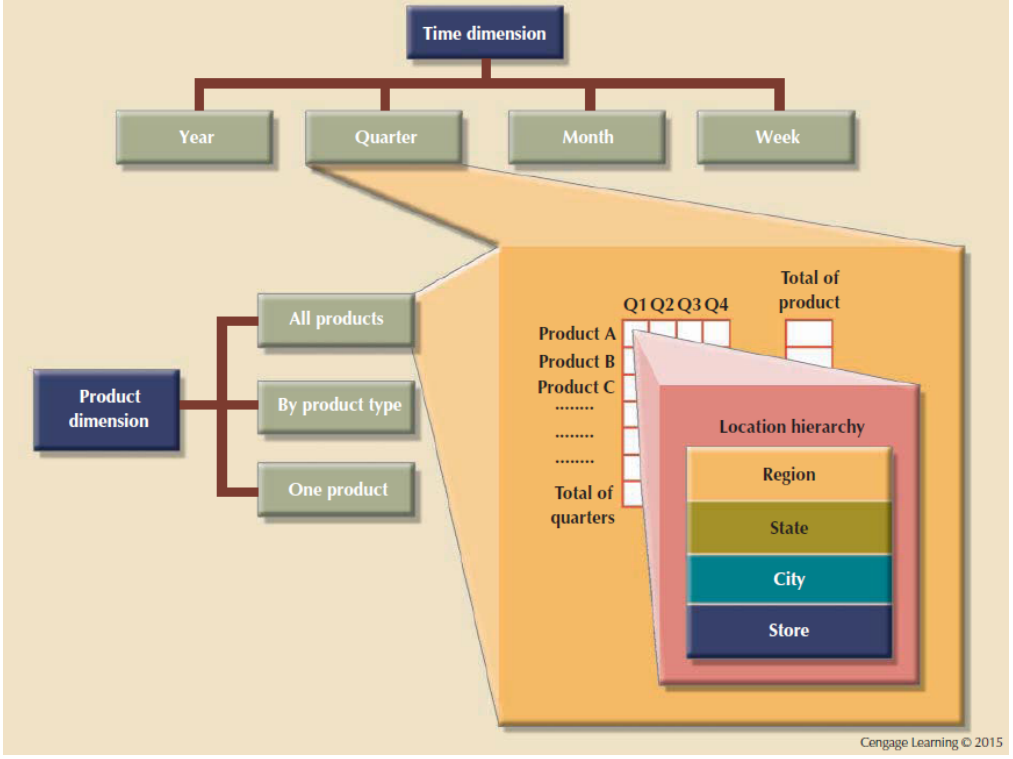


FIGURE 13.10 Attribute hierarchies in multidimensional analysis



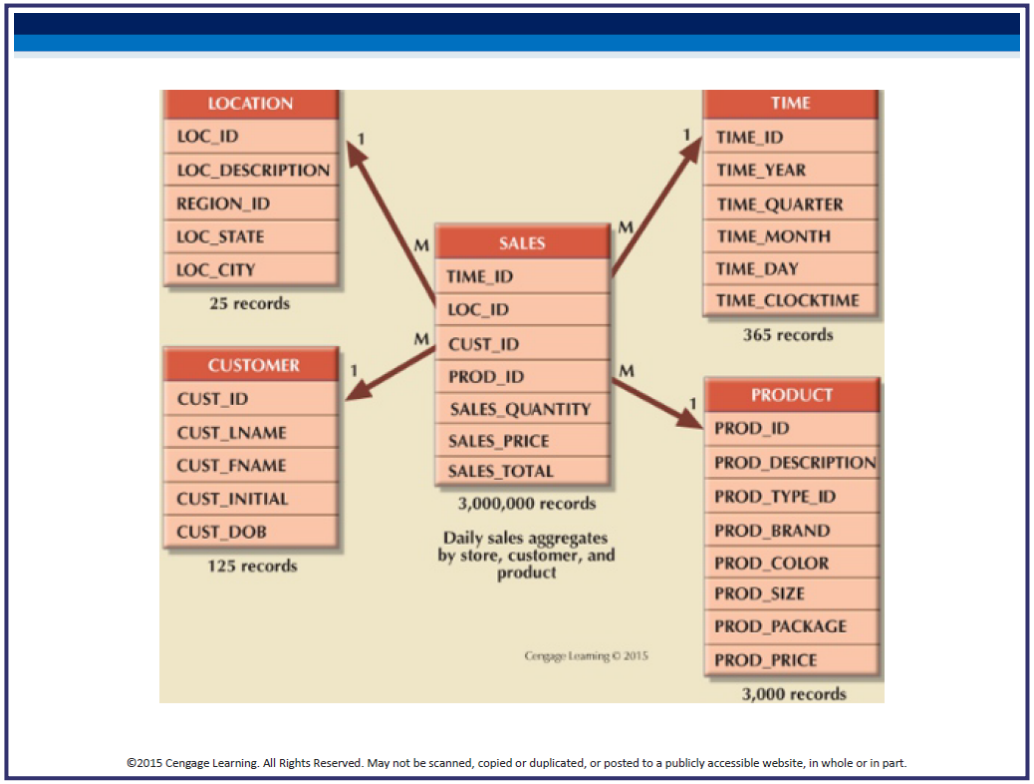
Star Schema Representation

- Facts and dimensions represented by physical tables in data warehouse database
- Many-to-one (M:1) relationship between fact table and each dimension table
- Fact and dimension tables
  - Related by foreign keys
  - Subject to primary and foreign key constraints

Star Schema Representation

- Primary key of a fact table
  - Is a composite primary key because the fact table is related to many dimension tables
  - Always formed by combining the foreign keys pointing to the related dimension tables

A sample star schema



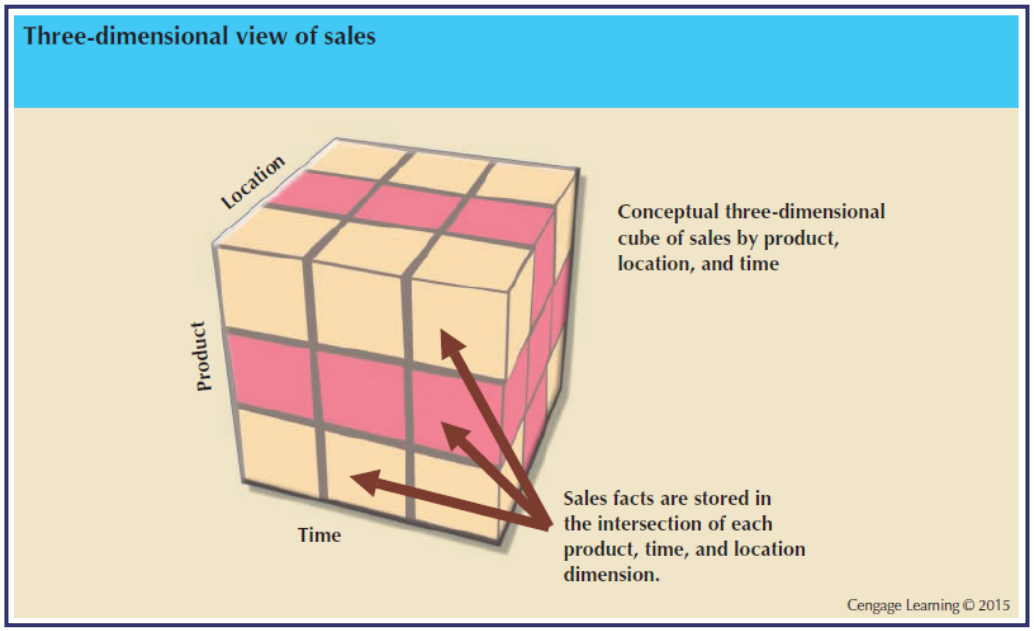
We have the fact table (of transactions) at the center, and denormalized (all-in-one) dimension tables all around.

Here is another representation.

Note: "dimensions are qualifying characteristics that provide additional perspectives to a given fact; dimensions provide descriptive characteristics about the facts through

their attributes."

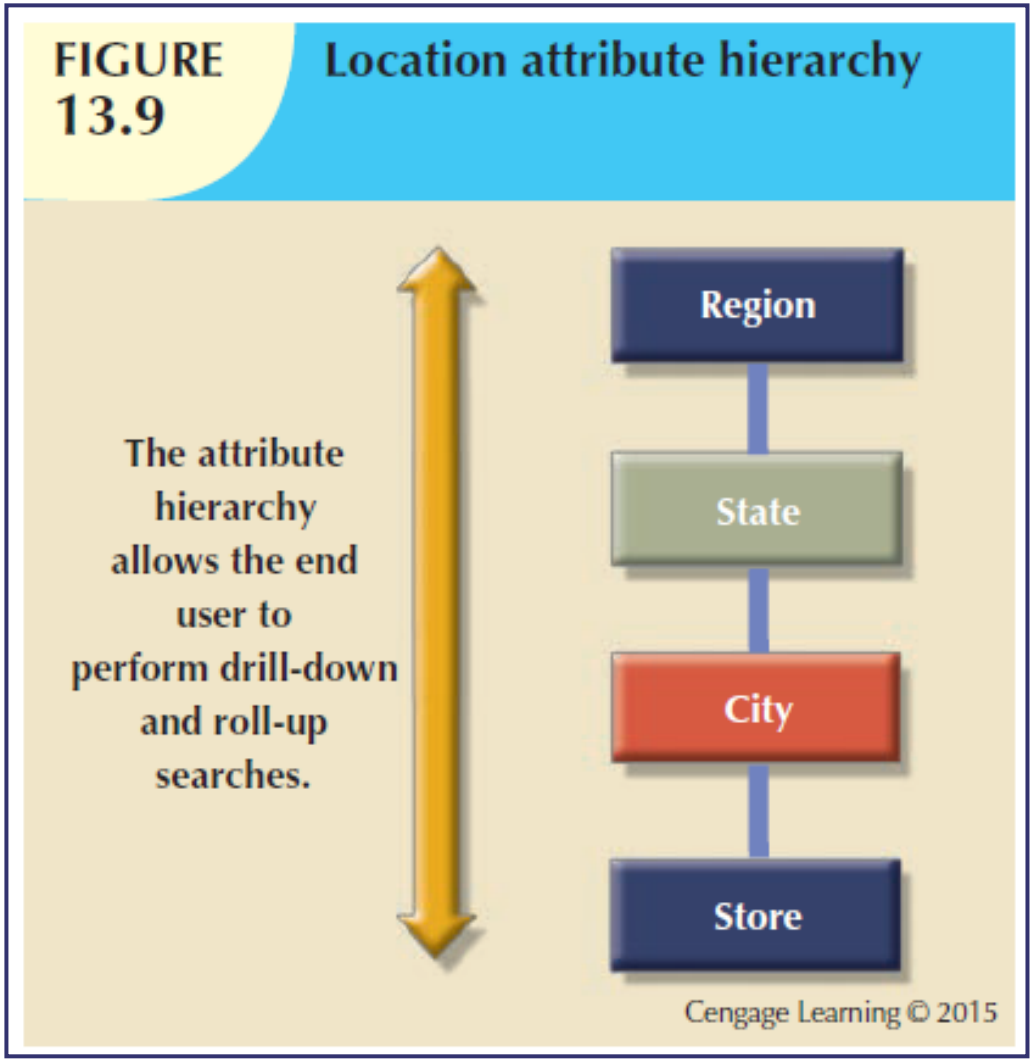
Each fact (transaction) can now pictured to be located in a multi-dimensional cube where the axes are dimensions. Eg. a 3D representation of our data for the above schema would look like this:



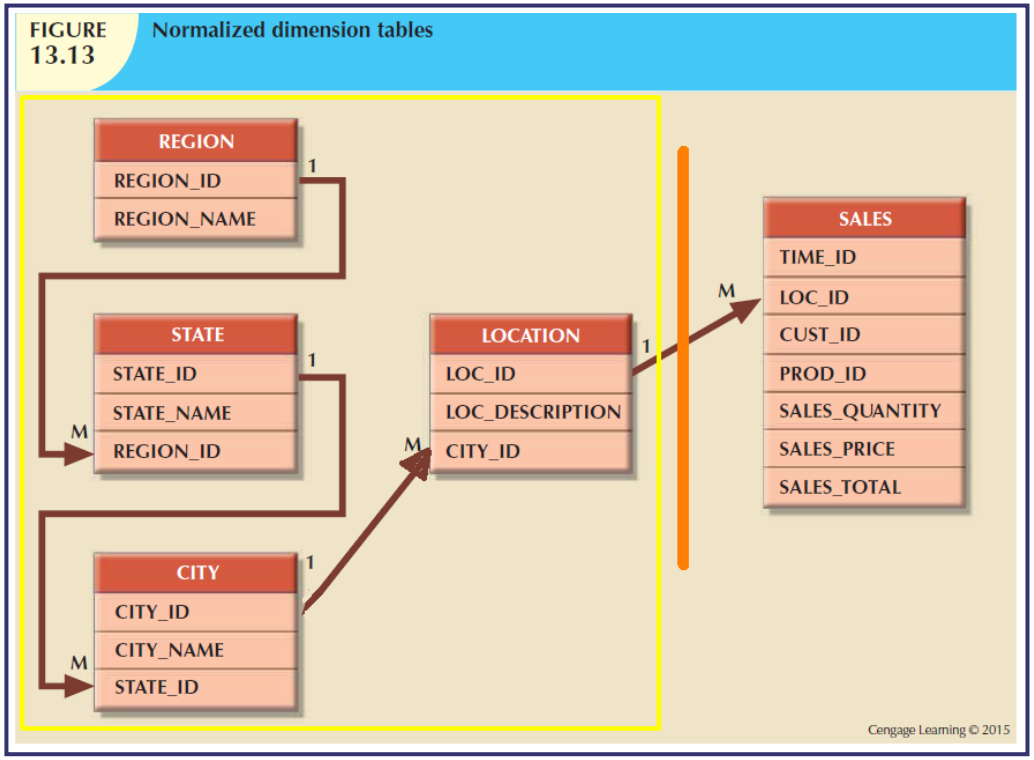
Slicing and dicing the cube provides specific insights..

Additionally, an attribute hierarchy would provide drill-down/roll-up capability as well, eg.

FIGURE 13.9 Location attribute hierarchy

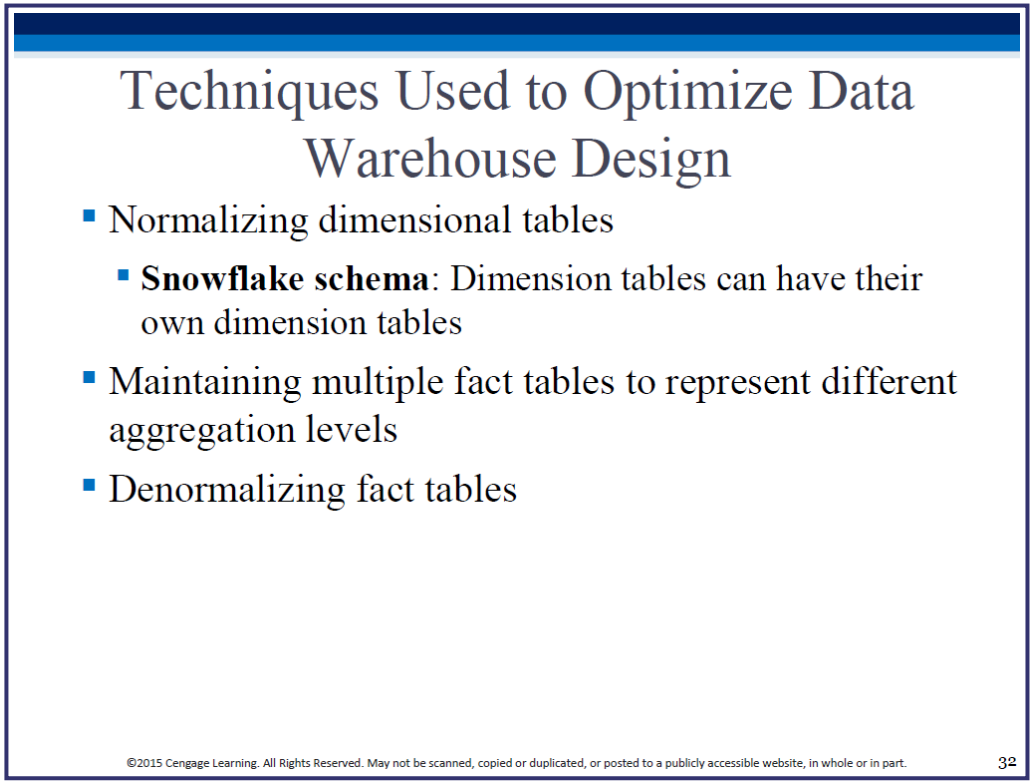


Snowflake schema



Dimensional tables can be normalized so that they have their own dimensional tables - this is done to simplify the design, but requiring navigation across the normalized chains.

Here is another representation.



There are four (five!) different ways in which we can organize (structure) a data warehouse:

- 1. star schema: fact table FKs point to a single level of dimension tables (points of a star)
- 2. snowflake schema: each dimension table can be normalized to create a 1:M chain

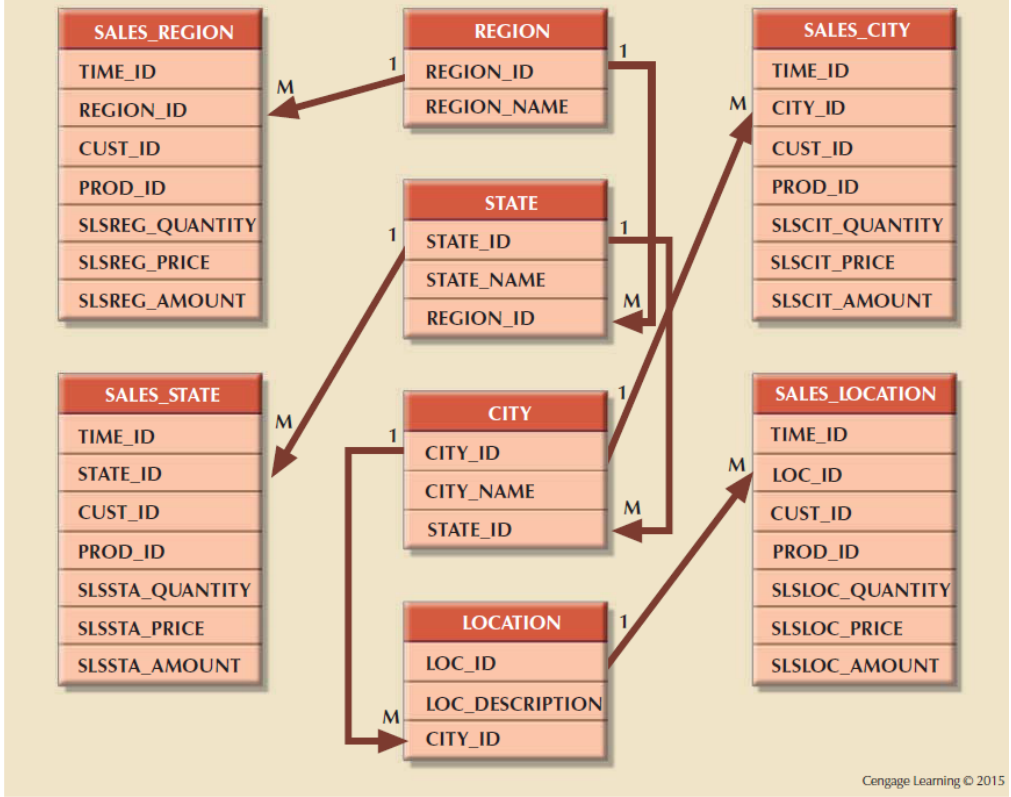
- 3. the fact table can be supplanted with the columns in the star or snowflake dimensions
- 4. a separate fact table can be created for each attribute in a dimension hierarchy

To denormalize a fact table (#3 above), we simply add extra 'dimension' columns to it, and fill them with redundant data - this permits fast queries (no joins needed) at the expense of disk space (and cleanliness of design).

Redundant fact tables!

Instead of a denormalized fact table (#3), or a fact table pointing to denormalized star dimensions (#1), or a fact table with lowest attrs pointing to a chain of rolled-up attrs, ie. snowflake schema (#2), we can create multiple fact tables, one for each level in an attr hierarchy (#4) - it is a different form of denormalization, where the redundant data is stored in physically separate tables.

FIGURE 13.14 Multiple fact tables



A summarization...

Fact tables that we see in the middle of star/snowflake schema, are ALWAYS denormalized, with multiple repeating values in the columns that link to dimensions - eg. multiple date values, product values, location values, POS terminal # values etc (because each row in a fact table contains those columns as raw 'facts').

Dimension tables, in a star schema are ALSO denormalized - eg. location dimension, with city,state,region columns, will have repeating values for states (because many cities are in each state), and repeating region values (because many states are in each region).

Dimension tables in a snowflake schema are normalized, because we create a chain (hierarchy) of them using the star's dimension columns.

The fact table ALWAYS stays denormalized. Such a fact table is said to employ star schema, if we use star-like denormalized columns for BI - eg. to find out how much of a product we sold in a city, we'd query the fact rows for city name, and if we need it, can also do state-level analyses (because states are listed in the location dimension table).

Using a snowflake schema, doing location analysis for a product at a city level is similar to the above paragraph - we simply look for the city name, and if necessary, get extra

info about the city (eg tax rate) by looking at the dimension table. BUT to do state level analysis, we need to follow the city->state link, and use the state-level dimension table ie traverse a branch of the snowflake.

To avoid traversing those branches in a snowflake, we trade off ('waste') space by creating extra 'copies' of the fact table, where a column such as city (lowest value in the hierarchy of 'location') is REPLACED instead with 'state' values, and in another copy, with 'region' values. This lets us do star-like analyses again, because a fact row directly points the state table, and in another copy, directly points to the region table - no traversing the chain necessary (at the expense of extra storage).

Which schema (star or snowflake) is used to model the warehouse, determines whether we maintain denormalized (or normalized) dimension tables [fact tables always stay denormalized]. 'For BI purposes, the idea is to take the 'single unified view' of data which is in the fact table (which contains numerous columns (think of a single Amazon purchase order item) - they can be categorized into dimensions, and in each dimension, even be hierarchically grouped - an example would be 'location'), and DERIVE additional tables, with data pre-aggregated along those (hierarchies of) dimensions. This lets us slice-and-dice (along dimensions), and zoom in/out (along just one dimension), all without expensive querying at runtime (on billions of rows), because the

'group by' calculations have been done already (that resulted in those aggregated data tables).'

Data Analytics

- Encompasses a wide range of mathematical, statistical, and modeling techniques to extract knowledge from data
  - Subset of BI functionality
- Classification of tools
  - **Explanatory analytics**: Focuses on discovering and explaining data characteristics and relationships based on existing data
  - **Predictive analytics**: Focuses on predicting future outcomes with a high degree of accuracy

Online Analytical Processing

- Advanced data analysis environment that supports decision making, business modeling, and operations research
- Characteristics
  - Multidimensional data analysis techniques
  - Advanced database support
  - Easy-to-use end-user interfaces

Figure 13.19 - OLAP Architecture

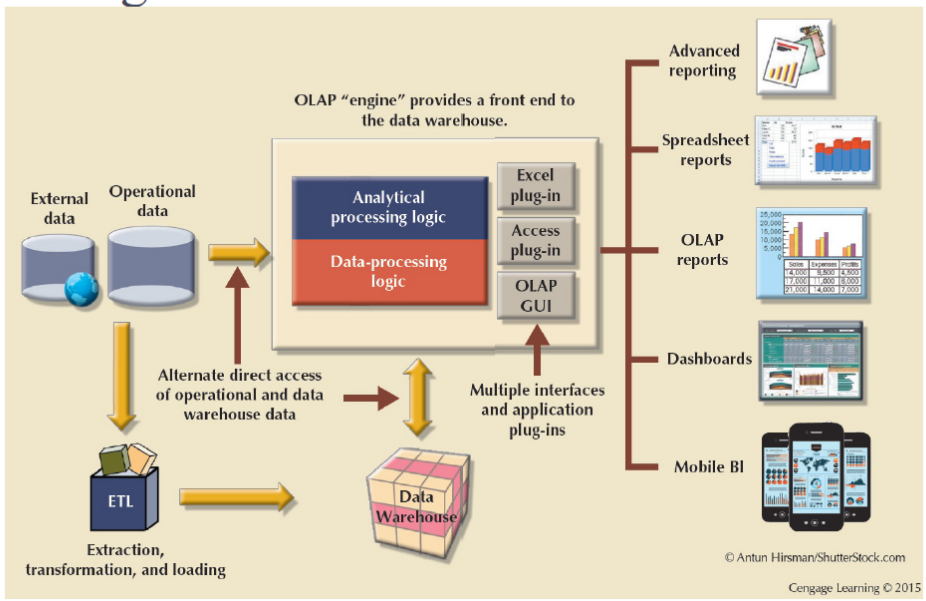
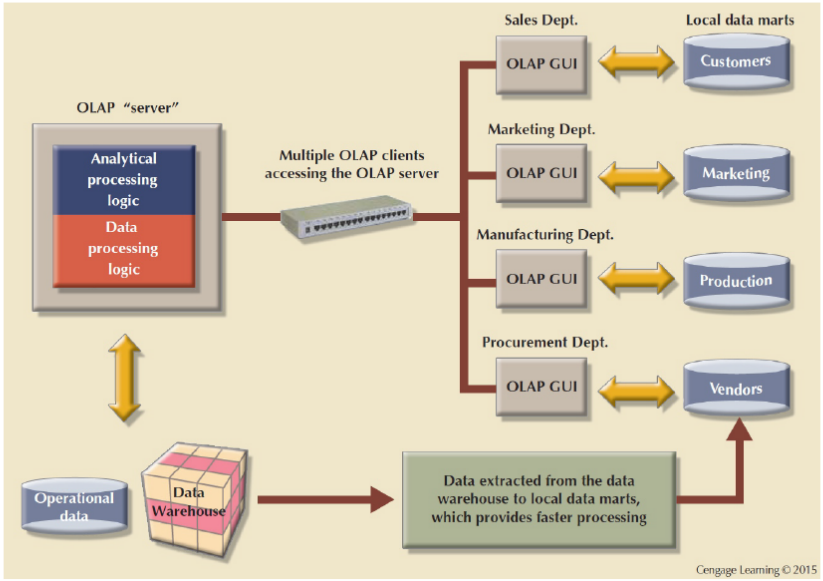


Figure 13.20 - OLAP Server with Local Miniature Data Marts



Relational OLAP:

Relational Online Analytical Processing (ROLAP)

- Provides OLAP functionality using relational databases and familiar relational tools to store and analyze multidimensional data
- Extensions added to traditional RDBMS technology
  - Multidimensional data schema support within the RDBMS
  - Data access language and query performance optimized for multidimensional data
  - Support for very large databases (VLDBs)

Multidimensional OLAP:

Multidimensional Online Analytical Processing (MOLAP)

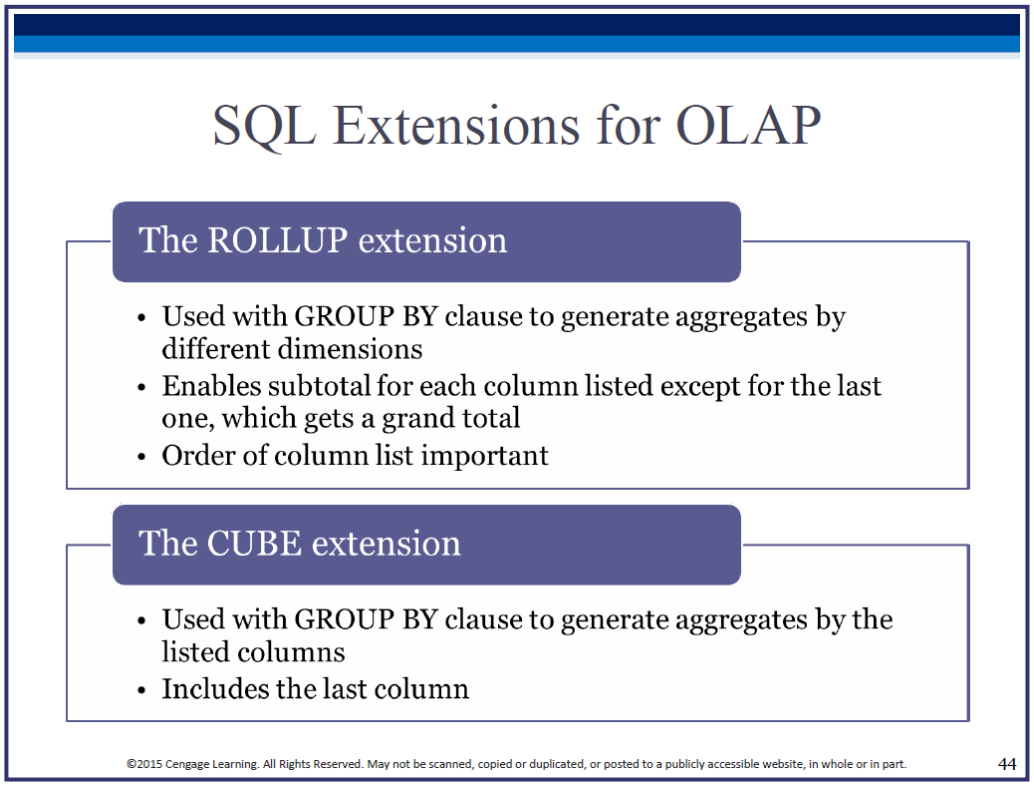
- Extends OLAP functionality to multidimensional database management systems (MDBMSs)
  - **MDBMS**: Uses proprietary techniques store data in matrix-like n-dimensional arrays
  - End users visualize stored data as a 3D **data cube**
    - Grow to n dimensions, becoming hypercubes
    - Held in memory in a **cube cache** to speed access
- **Sparsity**: Measures the density of the data held in the data cube

Table 13.12 - Relational vs. Multidimensional OLAP

CHARACTERISTIC	ROLAP	MOLAP
Schema	Uses star schema Additional dimensions can be added dynamically	Uses data cubes Multidimensional arrays, row stores, column stores Additional dimensions require re-creation of the data cube
Database size	Medium to large	Large
Architecture	Client/server Standards-based	Client/server Open or proprietary, depending on vendor
Access	Supports ad hoc requests Unlimited dimensions	Limited to predefined dimensions Proprietary access languages
Speed	Good with small data sets; average for medium-sized to large data sets	Faster for large data sets with predefined dimensions

BI-oriented SQL extensions

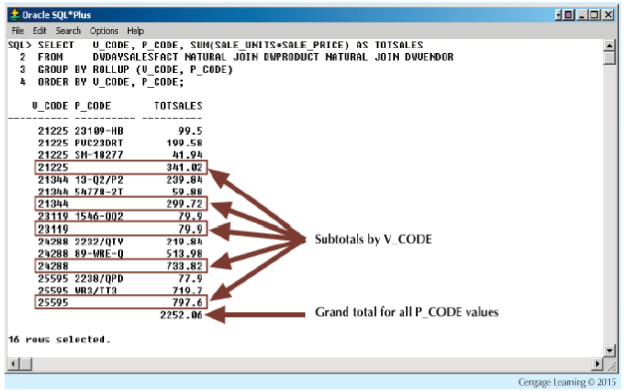
ROLLUP and CUBE are GROUP BY modifiers - they help generate subtotals for a list of specified columns (see examples that follow). Depending on granularity of the columns (eg. US_REGION vs STORE_NUMBER), these subtotals help provide a rolled-up (aggregated) or drilled-down (detailed) analysis of data.



ROLLUP, CUBE: usage examples

ROLLUP extension

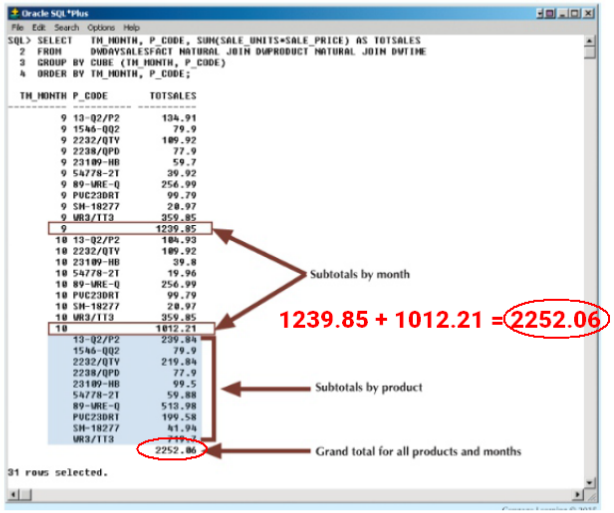
SELECT column1 [, column2, ...],
aggregate_function(expression)
FROM table1 [, table2, ...]
[WHERE condition]
GROUP BY ROLLUP (column1 [, column2, ...])
[HAVING condition]
[ORDER BY column1 [, column2, ...]]



"Subtotal for each vendor - all products; sum of (the only set of) subtotals".

CUBE extension

```
SELECT column1 [, column2, ...],
aggregate_function(expression)
FROM table1 [, table2, ...]
[WHERE condition]
GROUP BY CUBE (column1 [, column2, ...])
[HAVING condition]
[ORDER BY column1 [, column2, ...]]
```



"Subtotal for each month - all products; subtotal for each product - all months; sum of (either set of) subtotals".

Data Lakes

A 'traditional' data warehouse is an ETL-based, historical record of transactions - very RDB-like.

A 'modern' alternative is a 'data lake', which offers a more continuous form of analytics, driven by the rise of unstructed data, streaming, cloud storage, etc. In a data lake, data is NOT ETLd, rather, it is stored in its 'raw' ("natural") form [even incomplete, untransformed...].

Also, look up 'lakehouse', 'reverse ETL'...