London
Business
School

AM05 Data Management

04. Data Warehousing

Dr. David Tilson

¶Operational versus Informational Systems (e.g. OLAP)

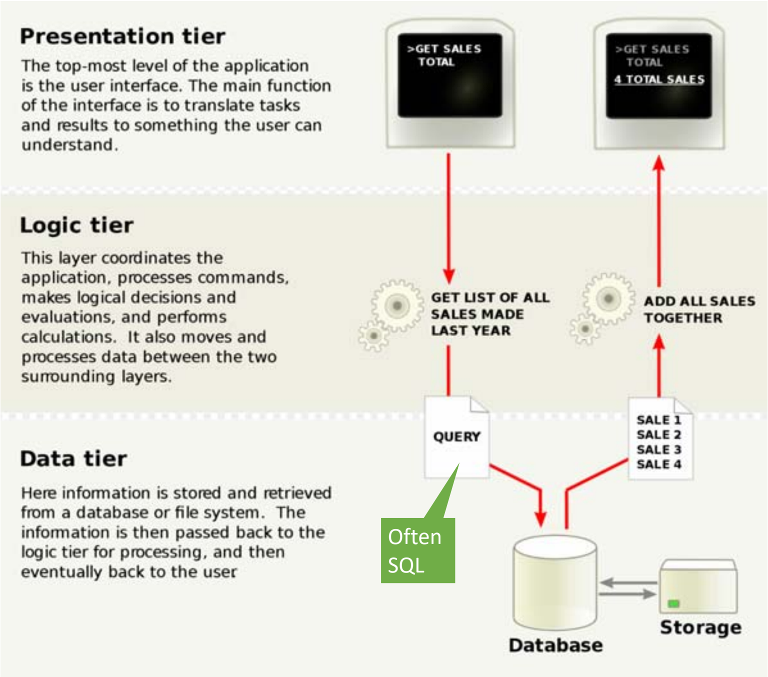
¶Data Warehousing and ETL

¶Business Intelligence, and Visualization

¶Text Parsing with Regular Expressions in SQL

¶For next time

Operational systems typically use a three-tier architecture



Presentation tier

The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.

Logic tier

This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.

Data tier

Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user

User interfaces

- Web pages
- GUI on PC/smartphone app
- May be hosted on web servers

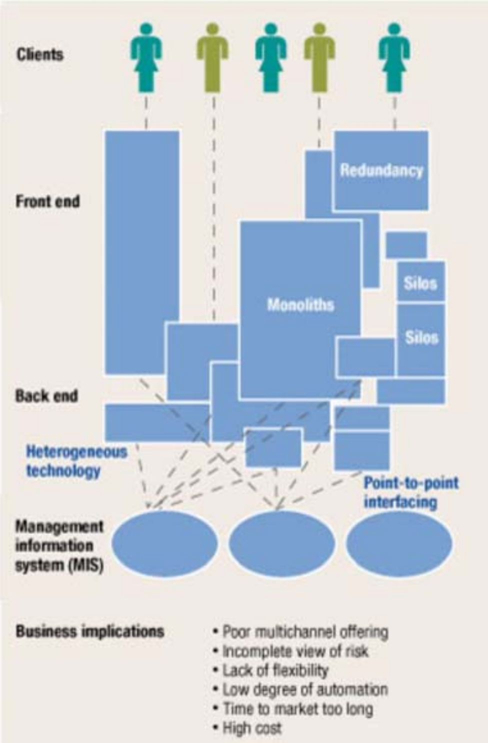
Business Logic

- Determines functionality
- Usually written in high performance language
- Hosted on application servers

Data Layer

- Provides data persistence
- Often high-availability RDBMs with back-up processes

Organizations can be **data rich** but **information poor**



- Poor multichannel offering
- Incomplete view of risk
- Lack of flexibility
- Low degree of automation
- Time to market too long
- High cost

Source: McKinsey & Co.

Operational systems

¶ Used to run business in real time

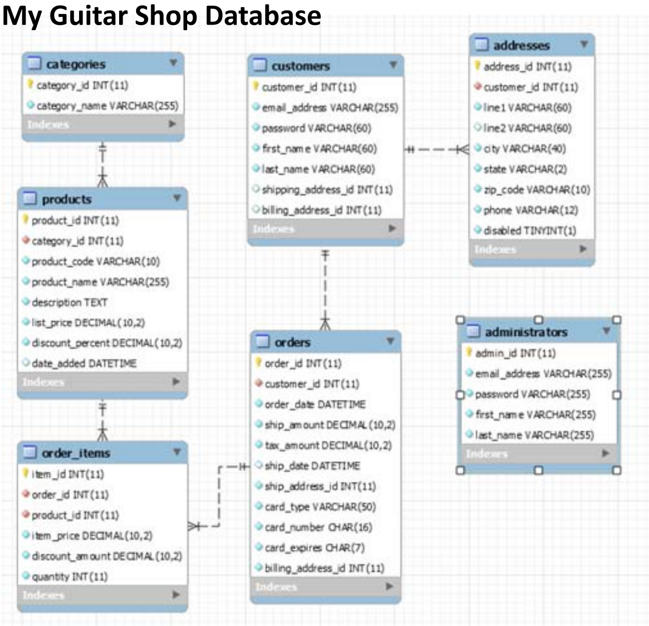
¶ Based on current data (system of record)

Reasons for information Gap

1. Development of one system at a time leads to hodgepodge of *uncoordinated and inconsistent databases*
   a. M&A activity makes complexity worse
   b. Nearly impossible for decision maker to find actionable information
2. Operational systems not designed to support analytics workloads
   a. Databases tuned to recording many small transactions
   b. Running large data analysis queries could prevent operational system fulfilling its main purpose

Normalized relational databases are great for operational purposes but not for analytics purposes …

My Guitar Shop Database



Creating complex reports might involve

- Lots of joins
- Scanning millions of rows
- Bringing an operational system to a standstill!

Informational systems have very different requirements to operational ones

Comparison of Operational and Informational Systems

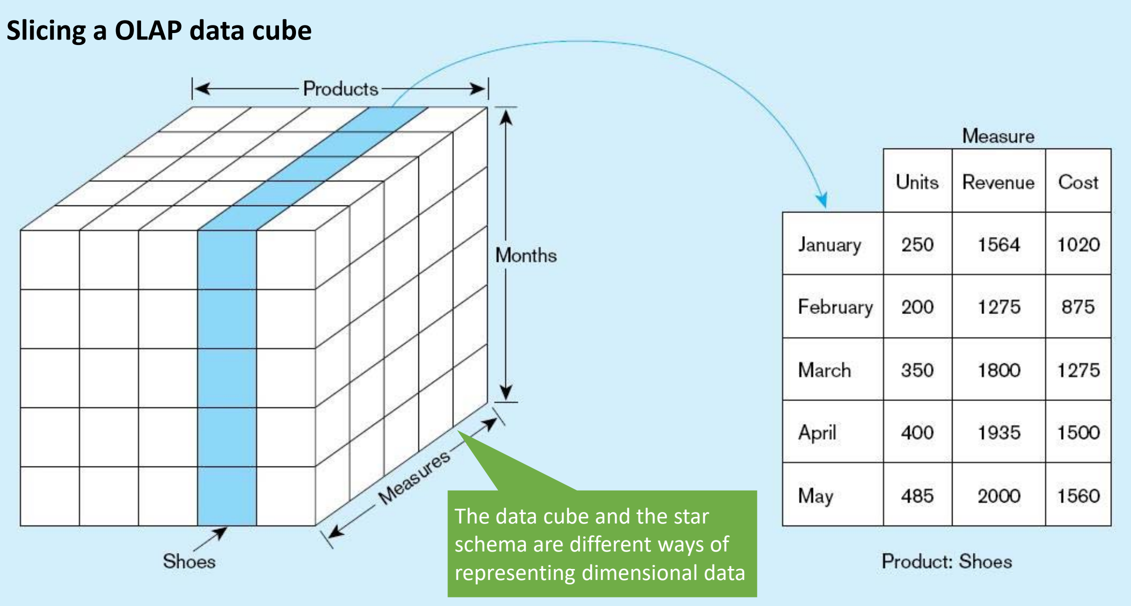
Characteristic	Operational Systems	Informational Systems
Primary purpose	Run the business on a current basis	Support managerial decision making
Type of data	Current representation of state of the business	Historical point-in-time (snapshots) and predictions
Primary users	Clerks, salespersons, administrators	Managers, business analysts, customers
Scope of usage	Narrow, planned, and simple updates and queries	Broad, ad hoc, complex queries and analysis
Design goal	Performance: throughput, availability	Ease of flexible access and use
Volume	Many constant updates and queries on one or a few table rows	Periodic batch updates and queries requiring many or all rows

Often called "online transaction processing (OLTP) databases"

Informational systems (e.g. data warehouses) are associated with "online analytical processing (OLAP)"

Could create all the aggregations you might need at night when the RDMBS is not so busy… that's the idea of OLAP*

Slicing a OLAP data cube



	Measure		
	Units	Revenue	Cost
January	250	1564	1020
February	200	1275	875
March	350	1800	1275
April	400	1935	1500
May	485	2000	1560

Product: Shoes

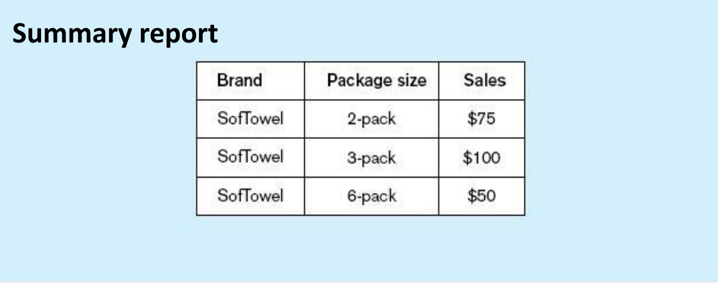
The data cube and the star schema are different ways of representing dimensional data

Slicing, dicing, pivoting, and drill-down are useful cube operations

* Online Analysis Processing. . . . . E.F. Codd was also a thought leader in OLAP

Here is an example of a drill-down operation

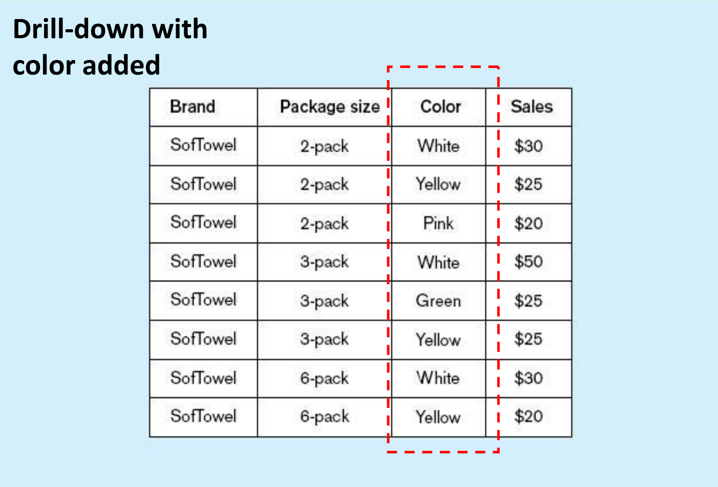
Summary report



Brand	Package size	Sales
SofTowel	2-pack	$75
SofTowel	3-pack	$100
SofTowel	6-pack	$50

Starting with summary data, users can obtain details for particular cells.

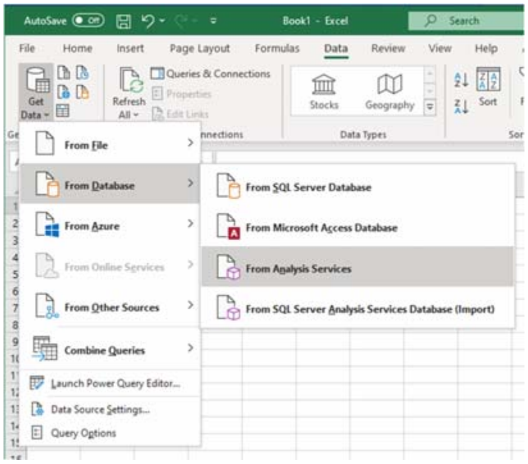
Drill-down with color added



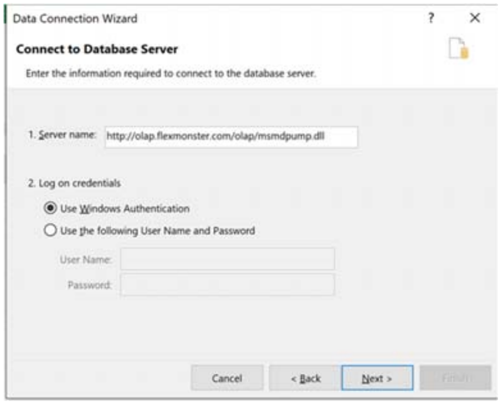
Brand	Package size	Color	Sales
SofTowel	2-pack	White	$30
SofTowel	2-pack	Yellow	$25
SofTowel	2-pack	Pink	$20
SofTowel	3-pack	White	$50
SofTowel	3-pack	Green	$25
SofTowel	3-pack	Yellow	$25
SofTowel	6-pack	White	$30
SofTowel	6-pack	Yellow	$20

Microsoft Excel includes an OLAP tool

Step 1… from Data Tab



Step 2… Enter Data Source



Data Connection Wizard

Connect to Database Server

Enter the information required to connect to the database server.

1. Server name: http://olap.flexmonster.com/olap/msmdpump.dll

2. Log on credentials

● Use Windows Authentication

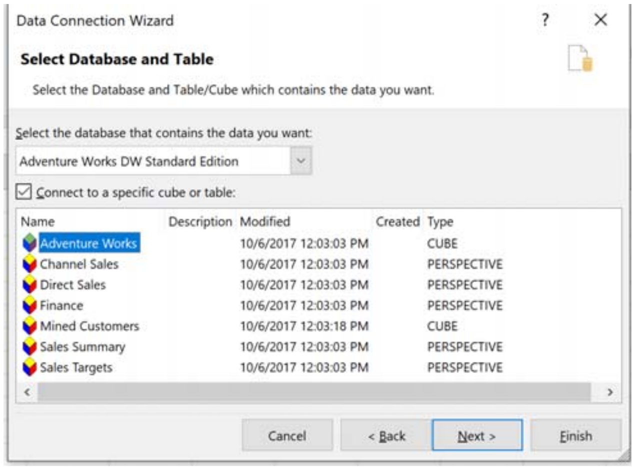
○ Use the following User Name and Password

User Name:

Password:

Microsoft Excel includes an OLAP tool

Step 3… Select Data Cube



Data Connection Wizard

Select Database and Table

Select the Database and Table/Cube which contains the data you want.

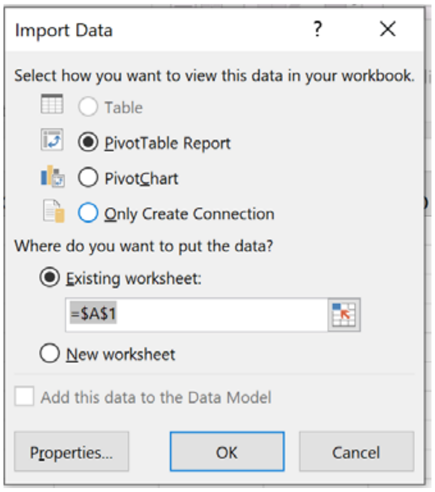
Select the database that contains the data you want:

Adventure Works DW Standard Edition

☑ Connect to a specific cube or table:

Name	Description	Modified	Created	Type
Adventure Works		10/6/2017 12:03:03 PM		CUBE
Channel Sales		10/6/2017 12:03:03 PM		PERSPECTIVE
Direct Sales		10/6/2017 12:03:03 PM		PERSPECTIVE
Finance		10/6/2017 12:03:03 PM		PERSPECTIVE
Mined Customers		10/6/2017 12:03:18 PM		CUBE
Sales Summary		10/6/2017 12:03:03 PM		PERSPECTIVE
Sales Targets		10/6/2017 12:03:03 PM		PERSPECTIVE

Step 4… Specify what you going to do



Import Data

Select how you want to view this data in your workbook.

○ Table

● PivotTable Report

○ PivotChart

○ Only Create Connection

Where do you want to put the data?

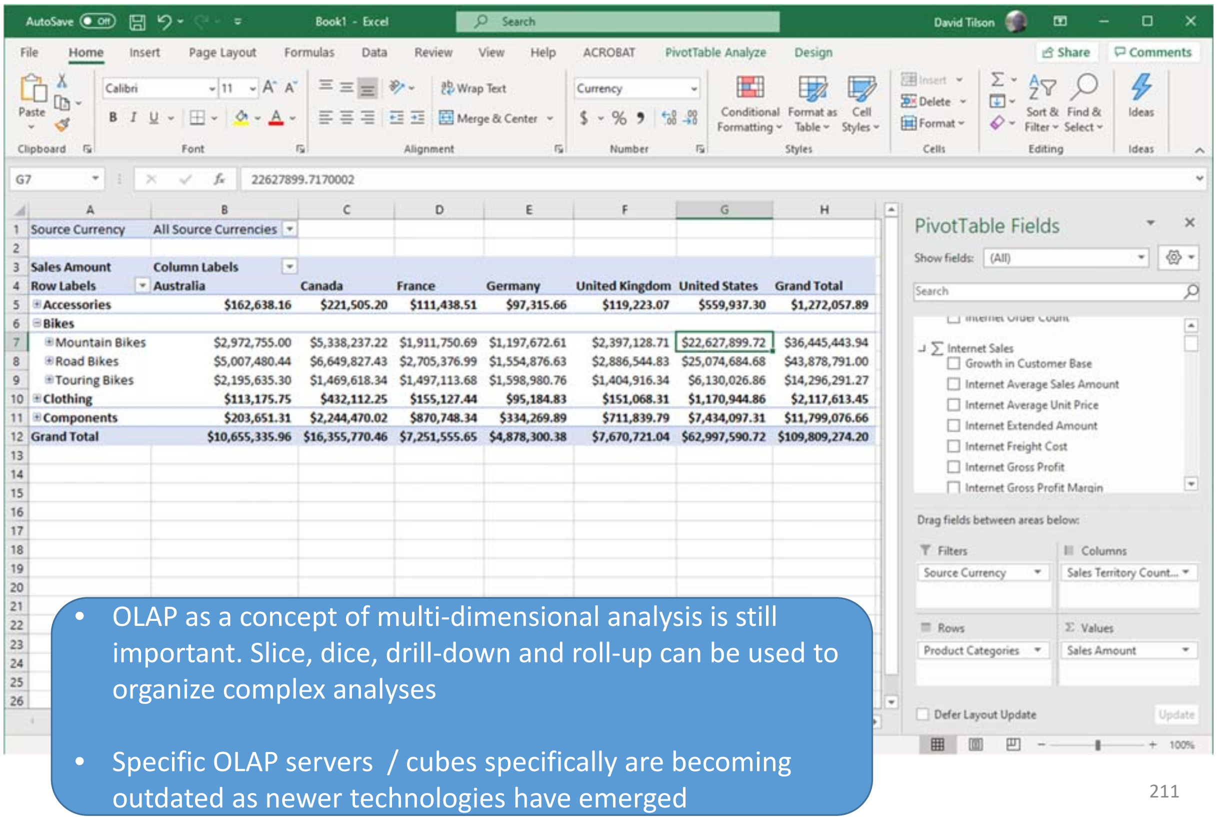
● Existing worksheet:

=A1

○ New worksheet

□ Add this data to the Data Model

Can then use Pivot Table to Slice, Dice, and Drill Down



G7 | 22627899.7170002

	Source Currency	All Source Currencies						
Sales Amount	Column Labels						
Row Labels	Australia	Canada	France	Germany	United Kingdom	United States	Grand Total
⊞Accessories	$162,638.16	$221,505.20	$111,438.51	$97,315.66	$119,223.07	$559,937.30	$1,272,057.89
⊟Bikes							
⊞Mountain Bikes	$2,972,755.00	$5,338,237.22	$1,911,750.69	$1,197,672.61	$2,397,128.71	$22,627,899.72	$36,445,443.94
⊞Road Bikes	$5,007,480.44	$6,649,827.43	$2,705,376.99	$1,554,876.63	$2,886,544.83	$25,074,684.68	$43,878,791.00
⊞Touring Bikes	$2,195,635.30	$1,469,618.34	$1,497,113.68	$1,598,980.76	$1,404,916.34	$6,130,026.86	$14,296,291.27
⊞Clothing	$113,175.75	$432,112.25	$155,127.44	$95,184.83	$151,068.31	$1,170,944.86	$2,117,613.45
⊞Components	$203,651.31	$2,244,470.02	$870,748.34	$334,269.89	$711,839.79	$7,434,097.31	$11,799,076.66
Grand Total	$10,655,335.96	$16,355,770.46	$7,251,555.65	$4,878,300.38	$7,670,721.04	$62,997,590.72	$109,809,274.20

PivotTable Fields

Show fields: (All)

Internet Sales

□ Growth in Customer Base
□ Internet Average Sales Amount
□ Internet Average Unit Price
□ Internet Extended Amount
□ Internet Freight Cost
□ Internet Gross Profit
□ Internet Gross Profit Margin

Drag fields between areas below:

Filters: Source Currency

Columns: Sales Territory Count...

Rows: Product Categories

Values: Sales Amount

□ Defer Layout Update

- OLAP as a concept of multi-dimensional analysis is still important. Slice, dice, drill-down and roll-up can be used to organize complex analyses
- Specific OLAP servers / cubes specifically are becoming outdated as newer technologies have emerged

¶Operational versus Informational Systems (e.g. OLAP)

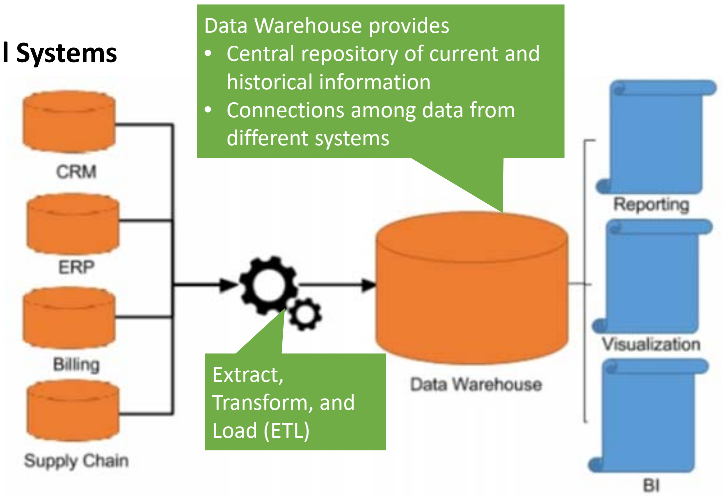
¶Data Warehousing and ETL

¶Business Intelligence, and Visualization

¶Text Parsing with Regular Expressions in SQL

¶For next time

A **data warehouse** pulls together data derived from operational systems and external sources for reporting and analysis



Data Warehouse provides

- Central repository of current and historical information
- Connections among data from different systems

Extract, Transform, and Load (ETL)

Transaction-oriented databases support/record day-to-day operations and transactions of a business

Data produced in operational system forms much of the input for a data warehouse

The purpose of a data warehouse is to assist with managerial decision-making. **Often provides the data that underpins the work you are doing in your analytics classes**

Organizations want an integrated, company-wide view of high-quality information (from disparate databases)

Data Warehouse (DW)

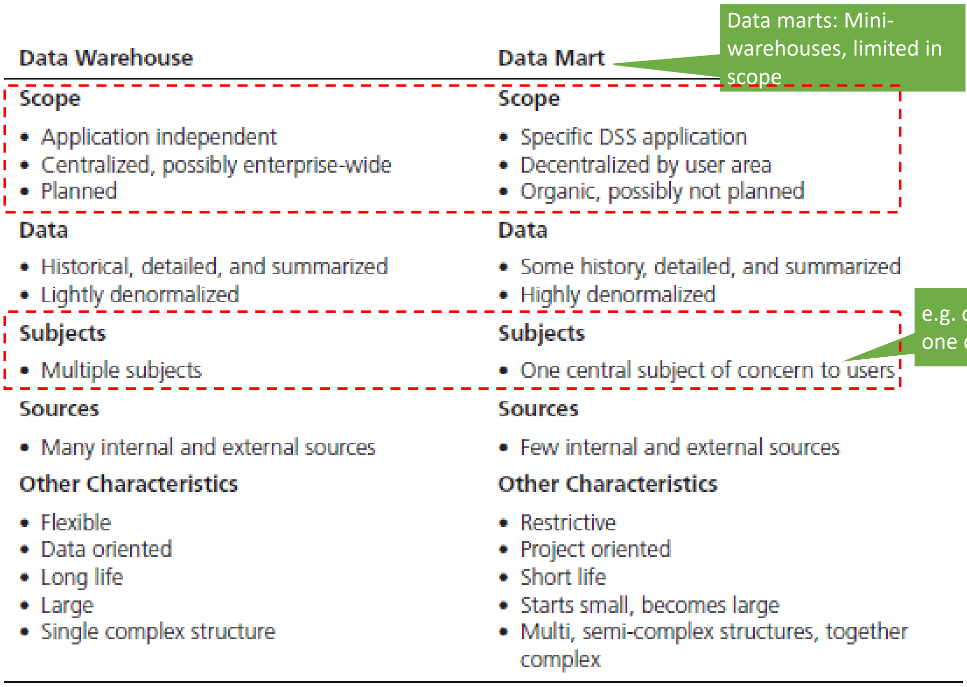
Subject-oriented, integrated, time-variant, non-updatable **collection of data used in support of management decision-making processes**

- ***Subject-oriented:*** e.g. customers, patients, students, products
- ***Integrated:*** consistent naming conventions, formats, encoding structures; from multiple data sources
- ***Time-variant:*** can study trends and changes
- ***Non-updatable:*** read-only, periodically refreshed

Separating operational and informational systems improves performance of both

See page in appendix outlining several trends in organizations which provide motivation for building data warehouses

Data Marts and **Data Warehouses** play different roles in a data warehousing environment



Data Warehouse	Data Mart
Scope	**Scope**
• Application independent • Centralized, possibly enterprise-wide • Planned	• Specific DSS application • Decentralized by user area • Organic, possibly not planned
Data	**Data**
• Historical, detailed, and summarized • Lightly denormalized	• Some history, detailed, and summarized • Highly denormalized
Subjects	**Subjects**
• Multiple subjects	• One central subject of concern to users
Sources	**Sources**
• Many internal and external sources	• Few internal and external sources
Other Characteristics	**Other Characteristics**
• Flexible • Data oriented • Long life • Large • Single complex structure	• Restrictive • Project oriented • Short life • Starts small, becomes large • Multi, semi-complex structures, together complex

Data marts: Mini-warehouses, limited in scope

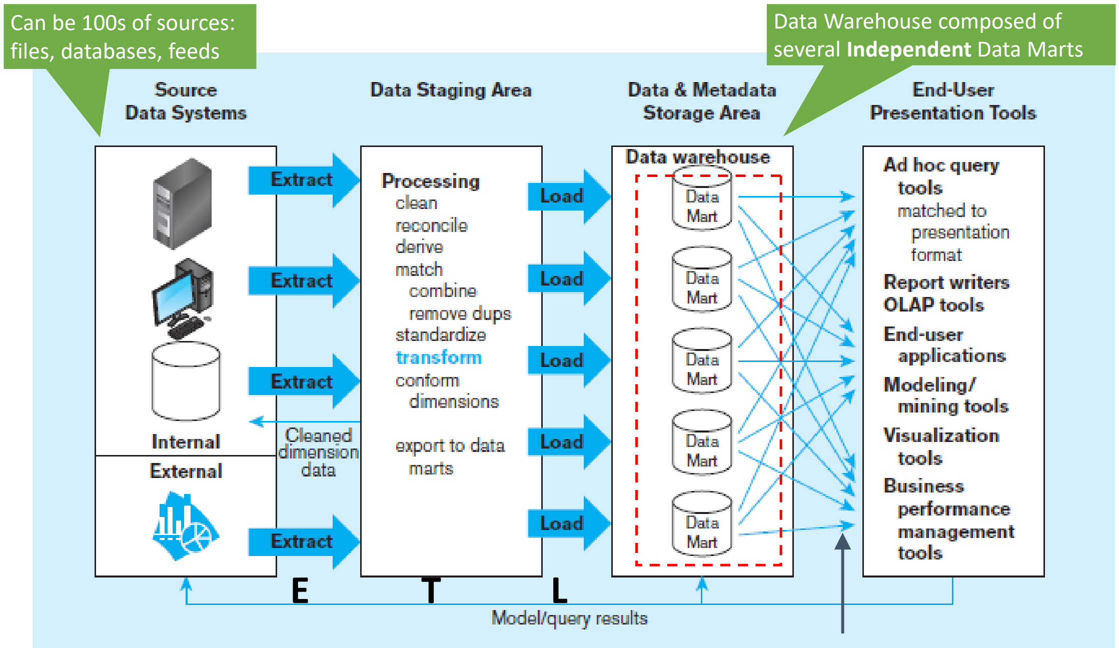
e.g. of interest to one department

There are several possible architectures for data warehousing

Independent data mart data warehousing architecture is easier to get started with – but has long-term implications (1/2)

Can be 100s of sources: files, databases, feeds

Data Warehouse composed of several **Independent** Data Marts



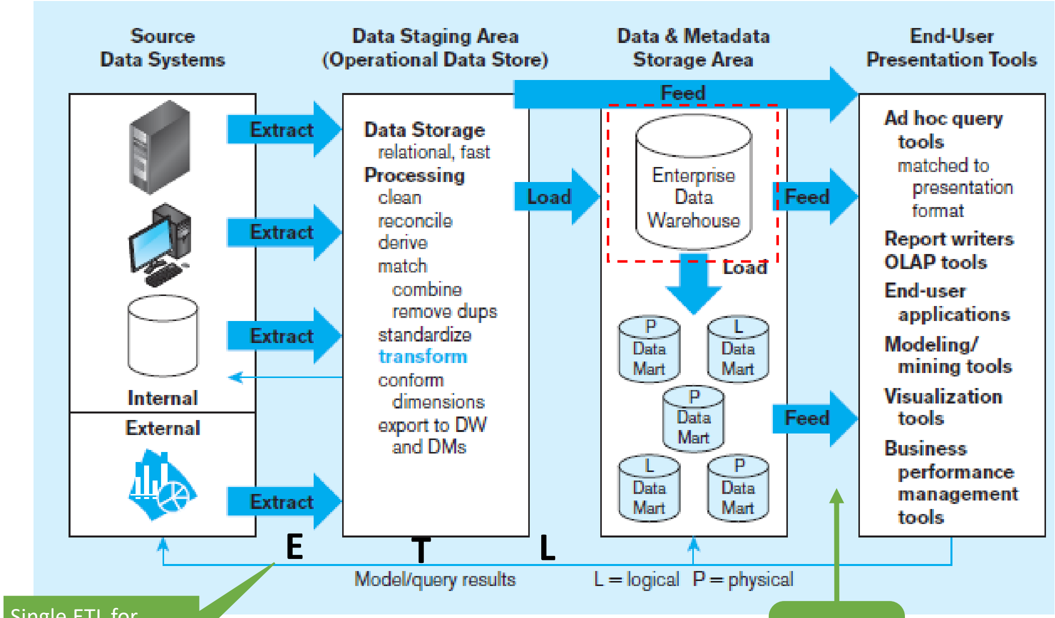
Separate ETL for each independent data mart

Data access complexity due to multiple data marts

Independent data mart data warehousing architecture is easier to get started with – but has long-term implications (2/2)

- Separate ETL process for each data mart ➔ redundant data and processing
- Inconsistency between data marts (high cost to obtain consistency)
- Difficult to drill down for related facts between data marts (must be done outside data warehouse)
- Excessive scaling costs as more applications are built

Enterprise DW is a centralized, integrated DW, serving as the control point and single source of truth made available to for decision support end users

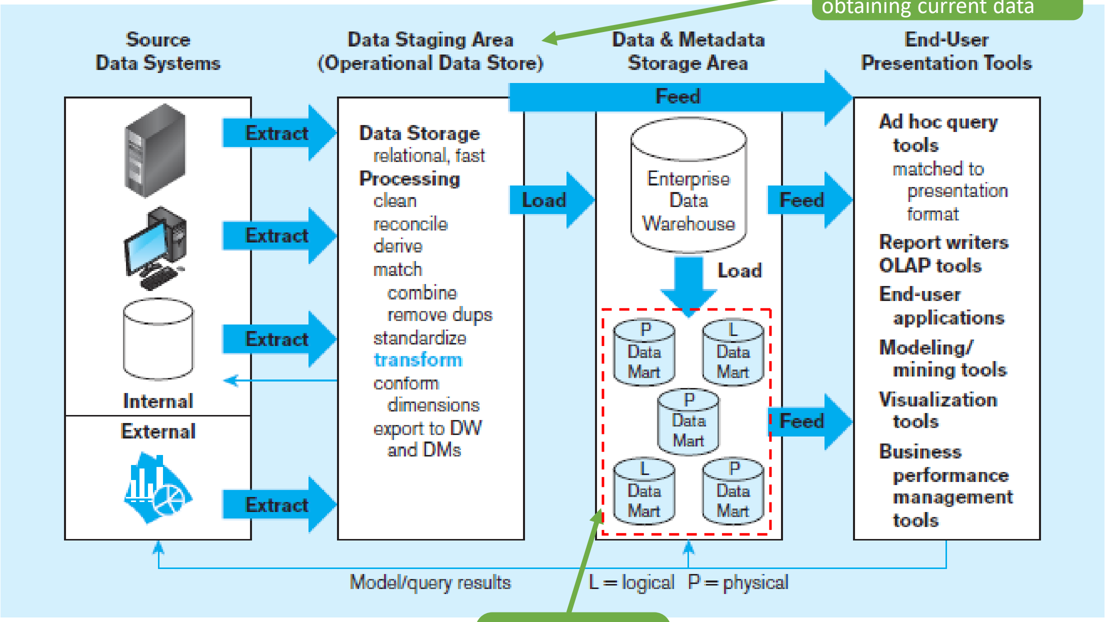


Single ETL for enterprise data warehouse (EDW)

Simpler data access

Dependent Data Marts are loaded from the Enterprise DW

ODS provides option for obtaining current data

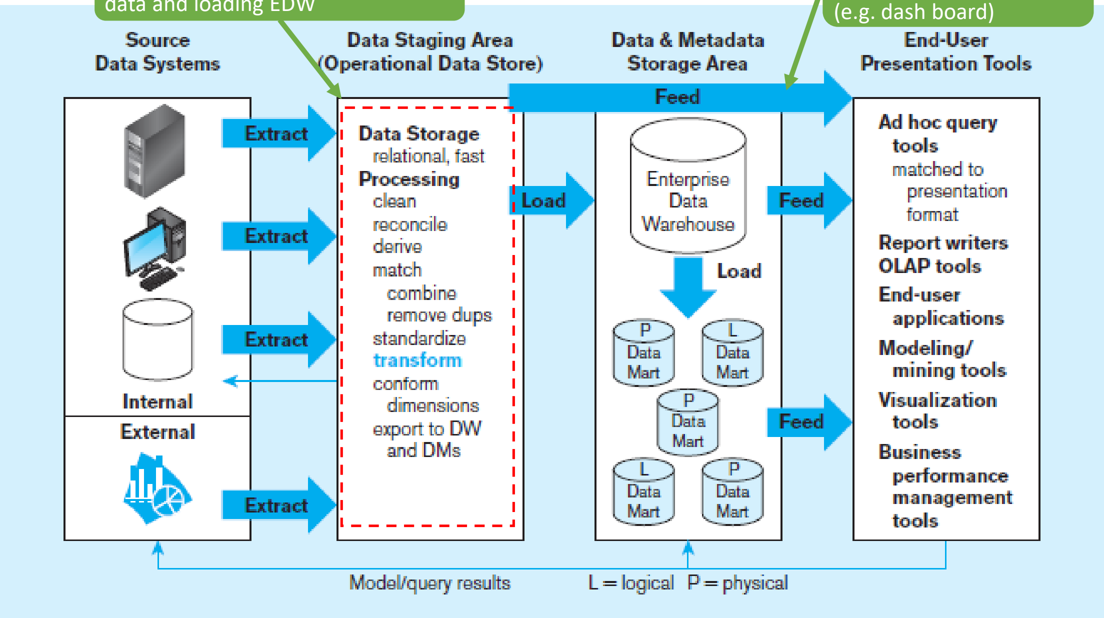


Dependent data marts loaded from EDW

Operation Data Store (ODS) integrated, subject-oriented, continuously updateable, current-valued (with recent history), enterprise-wide, detailed database

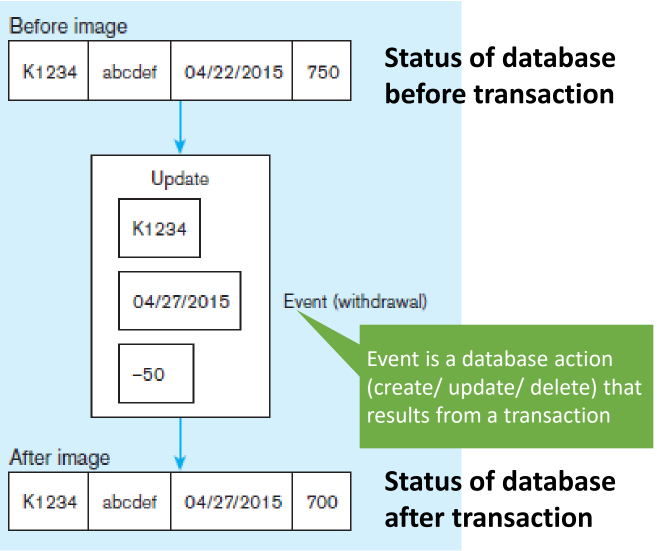
ODS is staging area for reconciling data and loading EDW

ODS provides option for obtaining current data (e.g. dash board)



Operational transactions change the status of database tables… but are not themselves recorded in the database

Example of DBMS log entry



Status of database before transaction

Event is a database action (create/ update/ delete) that results from a transaction

Status of database after transaction

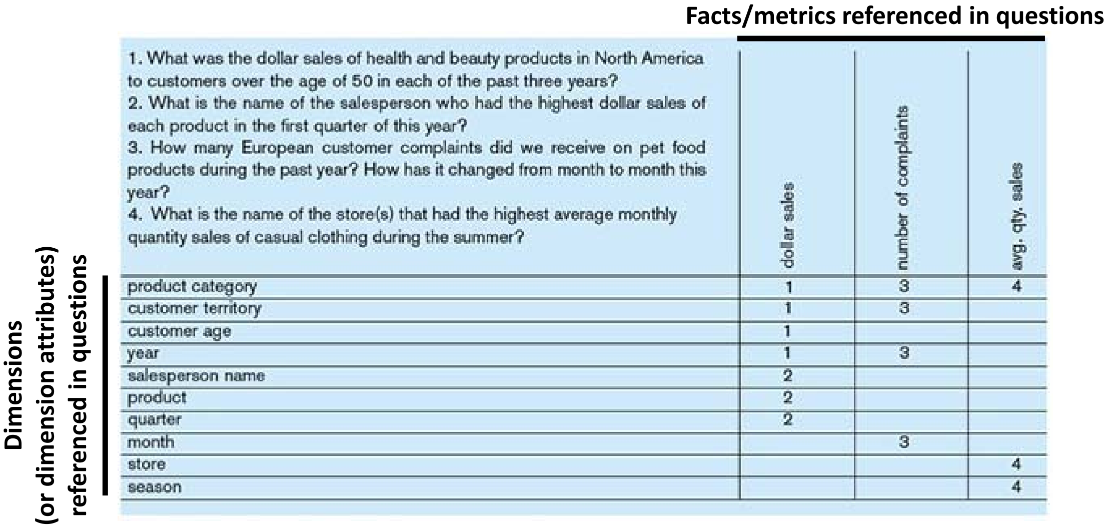
Sometimes we may want to know
- how many transactions are taking place
- when they take place

So, log data is often loaded into a DW, in addition to the data actually stored in an operational database

Clickstreams stored in webserver logs are also important sources of data

The design of the **schema for derived data** is driven by the types of insights that are sought

Facts/metrics referenced in questions



1. What was the dollar sales of health and beauty products in North America to customers over the age of 50 in each of the past three years?
2. What is the name of the salesperson who had the highest dollar sales of each product in the first quarter of this year?
3. How many European customer complaints did we receive on pet food products during the past year? How has it changed from month to month this year?
4. What is the name of the store(s) that had the highest average monthly quantity sales of casual clothing during the summer?

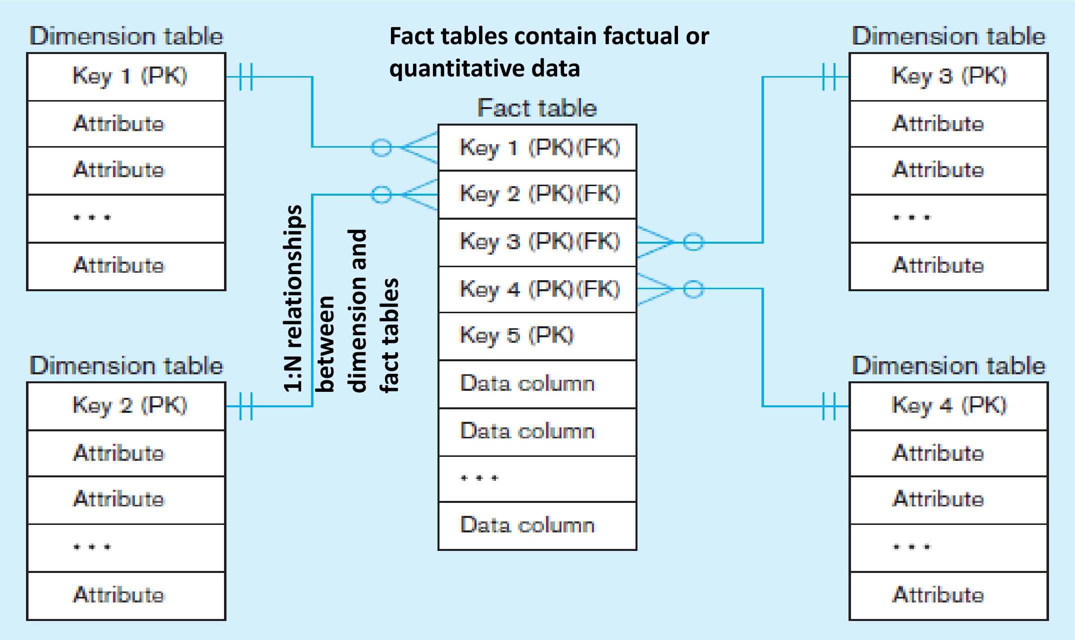
Dimensions (or dimension attributes) referenced in questions

	dollar sales	number of complaints	avg. qty. sales
product category	1	3	4
customer territory	1	3	
customer age	1		
year	1	3	
salesperson name	2		
product	2		
quarter	2		
month		3	
store			4
season			4

Objectives of derived data schema design
- Ease of use for decision support applications
- Fast response to predefined user queries
- Customized data for target audiences
- Ad-hoc query support and data mining capabilities

Typical characteristics
- Detailed (mostly periodic) data
- Aggregate (for summary)
- Distributed (departmental data marts)

Most common data model is the dimensional model (usually implemented as a star schema)



Fact tables contain factual or quantitative data

1:N relationships between dimension and fact tables

Dimension table
- Key 1 (PK)
- Attribute
- Attribute
- • • •
- Attribute

Dimension table
- Key 2 (PK)
- Attribute
- Attribute
- • • •
- Attribute

Fact table
- Key 1 (PK)(FK)
- Key 2 (PK)(FK)
- Key 3 (PK)(FK)
- Key 4 (PK)(FK)
- Key 5 (PK)
- Data column
- Data column
- • • •
- Data column

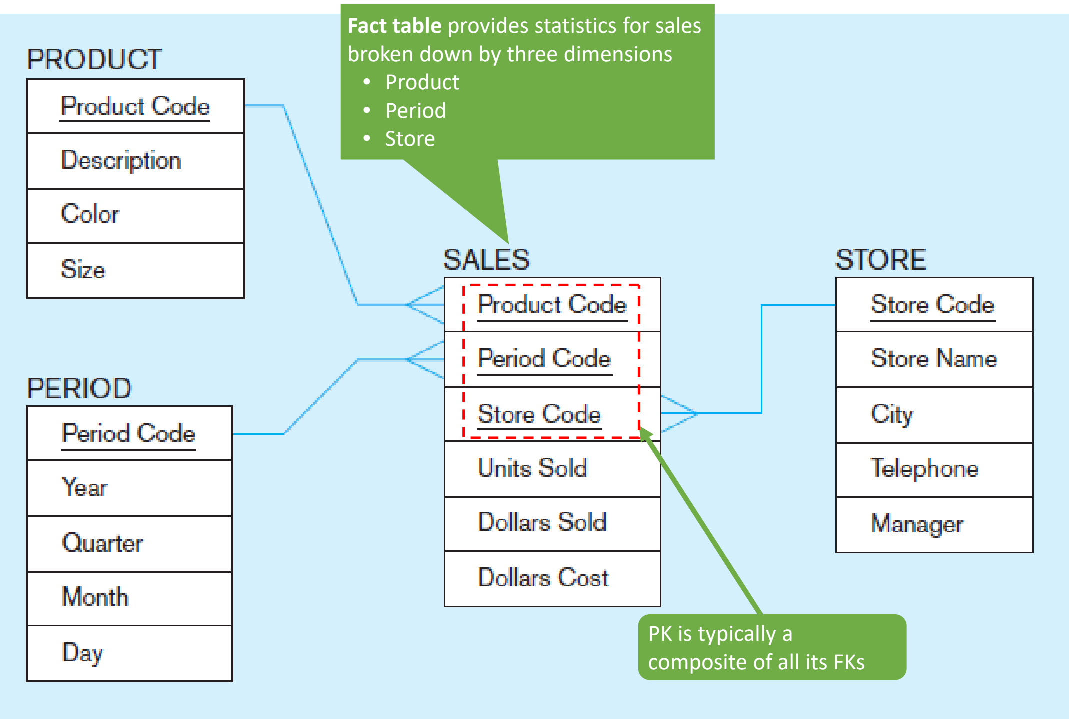
Dimension table
- Key 3 (PK)
- Attribute
- Attribute
- • • •
- Attribute

Dimension table
- Key 4 (PK)
- Attribute
- Attribute
- • • •
- Attribute

Star schema is excellent for ad-hoc queries, but bad for OLTP

Dimension tables
- Contain descriptions about subjects of the business
- Often denormalized to maximize performance

This example schema provides summary sales data



Fact table provides statistics for sales broken down by three dimensions
- Product
- Period
- Store

PRODUCT
- Product Code
- Description
- Color
- Size

PERIOD
- Period Code
- Year
- Quarter
- Month
- Day

SALES
- Product Code
- Period Code
- Store Code
- Units Sold
- Dollars Sold
- Dollars Cost

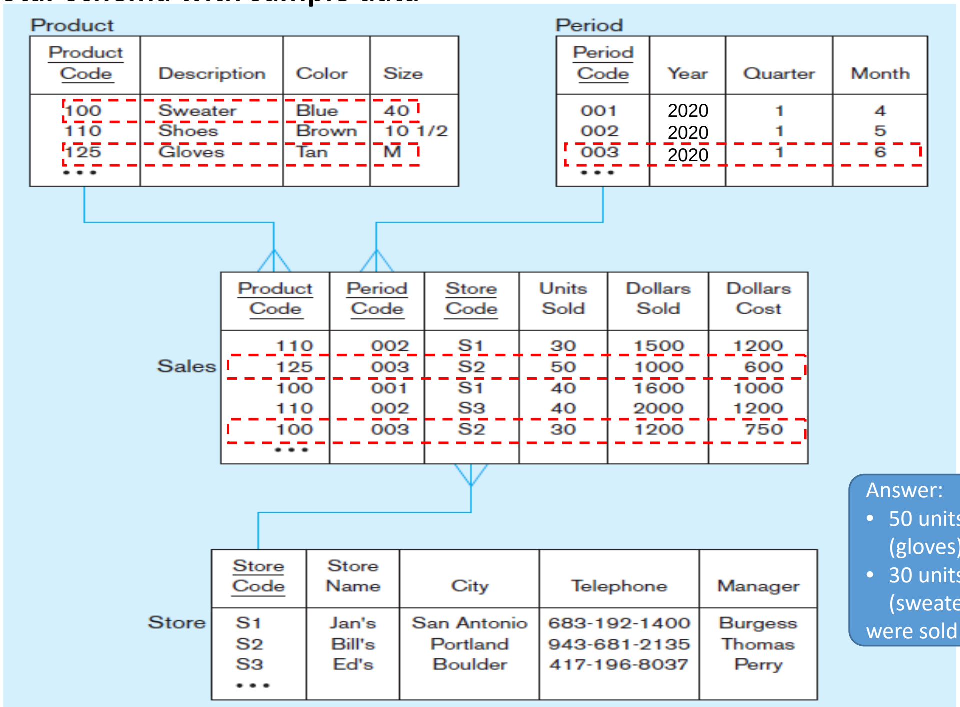
STORE
- Store Code
- Store Name
- City
- Telephone
- Manager

PK is typically a composite of all its FKs

Note: while the operational database has records of each sale, here we get summary data (units sold, dollars sold, dollars cost)

Question: What can you say about unit sales for store S2?

Star schema with sample data



Product

Product Code	Description	Color	Size
100	Sweater	Blue	40
110	Shoes	Brown	10 1/2
125	Gloves	Tan	M
• • •			

Period

Period Code	Year	Quarter	Month
001	2020	1	4
002	2020	1	5
003	2020	1	6
• • •			

Sales

Product Code	Period Code	Store Code	Units Sold	Dollars Sold	Dollars Cost
110	002	S1	30	1500	1200
125	003	S2	50	1000	600
100	001	S1	40	1600	1000
110	002	S3	40	2000	1200
100	003	S2	30	1200	750
• • •					

Store

Store Code	Store Name	City	Telephone	Manager
S1	Jan's	San Antonio	683-192-1400	Burgess
S2	Bill's	Portland	943-681-2135	Thomas
S3	Ed's	Boulder	417-196-8037	Perry
• • •				

Answer:

- 50 units of product 125 (gloves)
- 30 units of product 100 (sweaters)

were sold in June 2010

Choosing the **grain** in the dimensional model is a key decision

¶ The **Grain** is the finest level of detail in a fact table

- Determined by intersection of all components of its PK
- Cannot “drill down” below the grain of the fact table

¶ Some recommend using smallest grain possible… will be needed to explain why certain aggregated patterns exist

¶ **Transactional grain** is finest level (e.g. a ‘click’ in e-commerce)

¶ **Aggregated grain** is more summarized

¶ Finer grains

¶ Better analysis capability

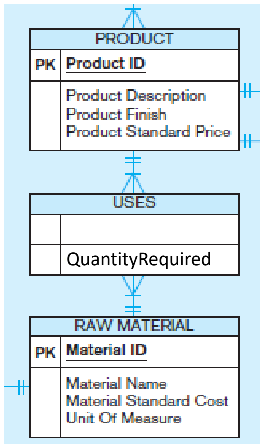
¶ More dimension tables, more rows in fact table

Determines size of fact tables

In-class exercise: Aggregation

10-minute breakout

- These tables in a relational database model the materials needed to build various products



- Write an SQL query to calculate the total raw material cost (label TotCost) for each product. Include product ID, product description, Product Standard Price, and the TotCost in the output
- Submit your solution to https://forms.gle/CP4Wo57rM74satiPA One submission per breakout team

In-class exercise: Fact table sizing

Size of table depends on

- Number of dimensions and the grain of the fact table
- Number of rows = product of number of possible values for each dimension associated with the fact table. For our sales example

Context

1000 stores 5,000 active products in any one month

24 months of data 6 facts per row (avg 4 bytes per fact)

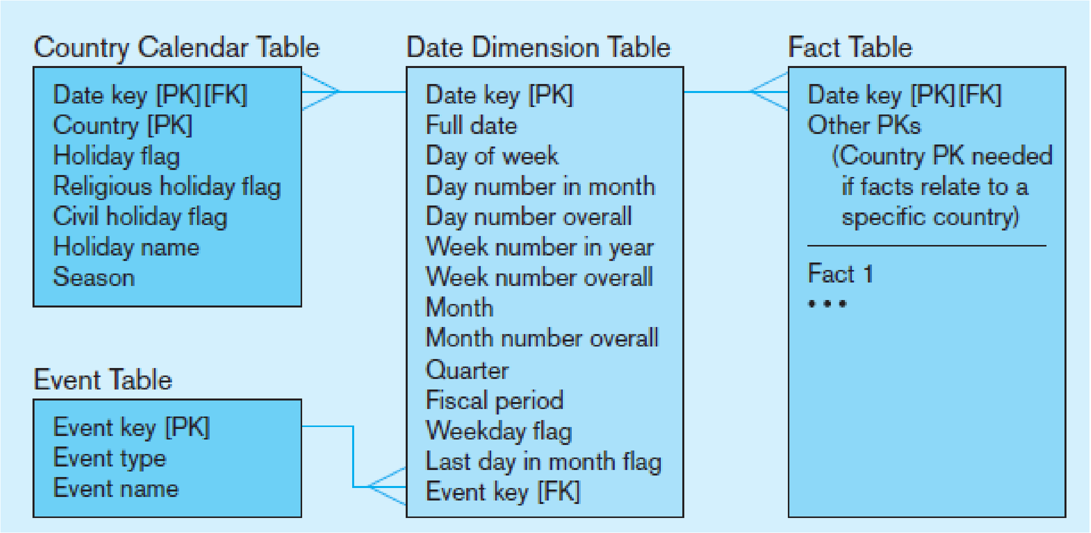
_______ **rows** or about _______ **bytes** of storage

… what about changing the granularity to the daily level (with 2000 active products)?

Submit your solution to https://forms.gle/Wd33wTJYu5TBrWQXA . One submission per breakout team

DWs are time-variant. So, fact tables always have a time/date dimension. Some models get quite elaborate

Date modeled to in include national holidays and significant events (e.g. sports)



Country Calendar Table
- Date key [PK][FK]
- Country [PK]
- Holiday flag
- Religious holiday flag
- Civil holiday flag
- Holiday name
- Season

Event Table
- Event key [PK]
- Event type
- Event name

Date Dimension Table
- Date key [PK]
- Full date
- Day of week
- Day number in month
- Day number overall
- Week number in year
- Week number overall
- Month
- Month number overall
- Quarter
- Fiscal period
- Weekday flag
- Last day in month flag
- Event key [FK]

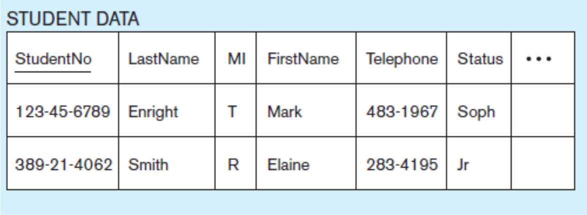
Fact Table
- Date key [PK][FK]
- Other PKs
  (Country PK needed if facts relate to a specific country)
- Fact 1
- • • •

Extracting data from many sources, ensuring consistency/validity, and transforming it to common framework are important tasks in data warehousing process

Common issues illustrated by simple example from education setting in which several departments have their own databases/files

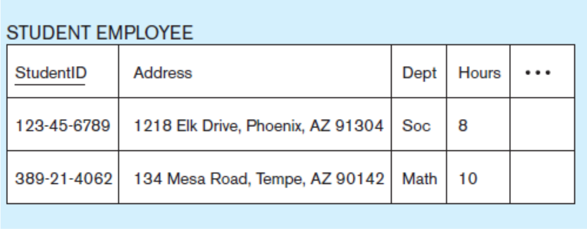
- Inconsistent key structures
- Synonyms (StudentNo, StudentID, ID)
- Free-form vs. structured fields
  (StudentName vs. LastName,MI,FirstName)
- Inconsistent data values (Phone for Elaine)
- Missing data (insurance details for Elaine)

All DW architectures involve some form of ETL
- **Extract**
- **Transform** and
- **Load**



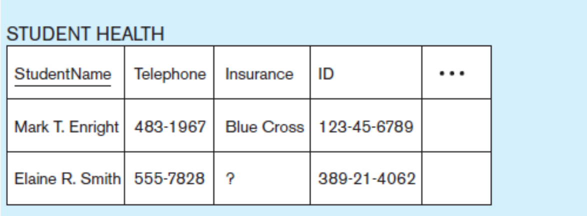
STUDENT DATA

StudentNo	LastName	MI	FirstName	Telephone	Status	• • •
123-45-6789	Enright	T	Mark	483-1967	Soph	
389-21-4062	Smith	R	Elaine	283-4195	Jr	



STUDENT EMPLOYEE

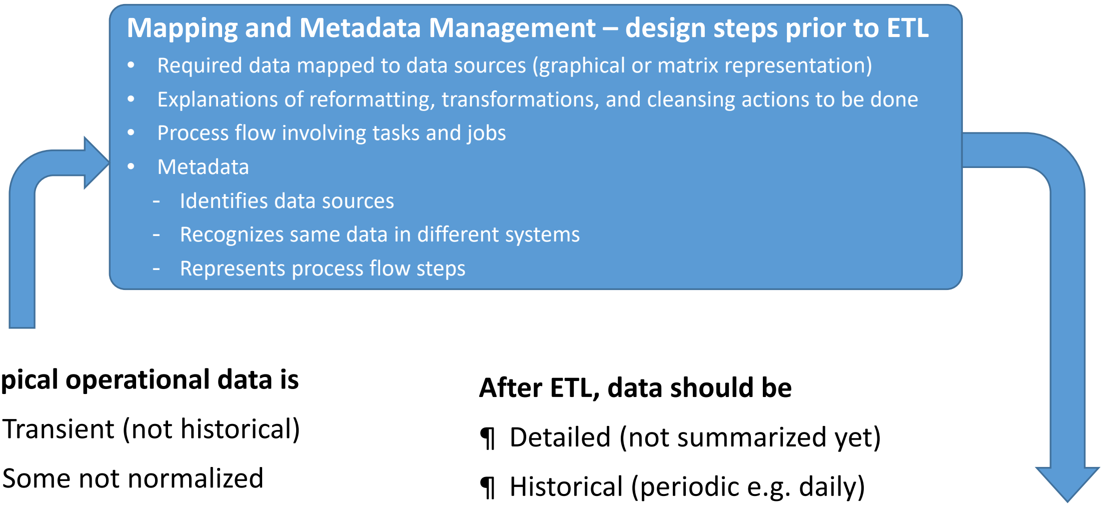
StudentID	Address	Dept	Hours	• • •
123-45-6789	1218 Elk Drive, Phoenix, AZ 91304	Soc	8	
389-21-4062	134 Mesa Road, Tempe, AZ 90142	Math	10	



STUDENT HEALTH

StudentName	Telephone	Insurance	ID	• • •
Mark T. Enright	483-1967	Blue Cross	123-45-6789	
Elaine R. Smith	555-7828	?	389-21-4062	

Significant planning required for effective ETL



Mapping and Metadata Management – design steps prior to ETL

- Required data mapped to data sources (graphical or matrix representation)
- Explanations of reformatting, transformations, and cleansing actions to be done
- Process flow involving tasks and jobs
- Metadata
  - Identifies data sources
  - Recognizes same data in different systems
  - Represents process flow steps

Typical operational data is

¶ Transient (not historical)

¶ Some not normalized

¶ Restricted in scope – not comprehensive

¶ Sometimes poor quality (inconsistencies and errors)

After ETL, data should be

¶ Detailed (not summarized yet)

¶ Historical (periodic e.g. daily)

¶ More normalized – 3NF or higher

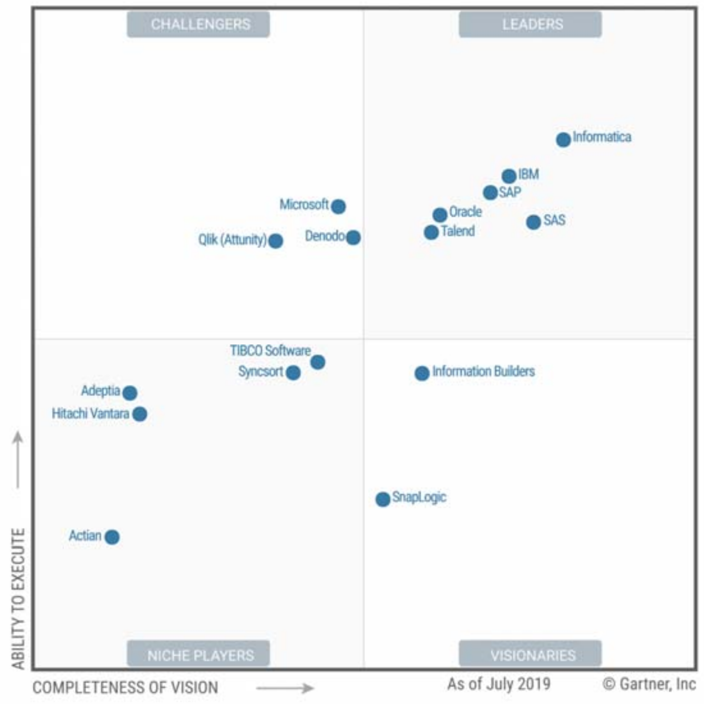
¶ Comprehensive (enterprise-wide view)

¶ Timely – current enough to assist decision-making

¶ Quality controlled – accurate with full integrity

There are many ETL tools available

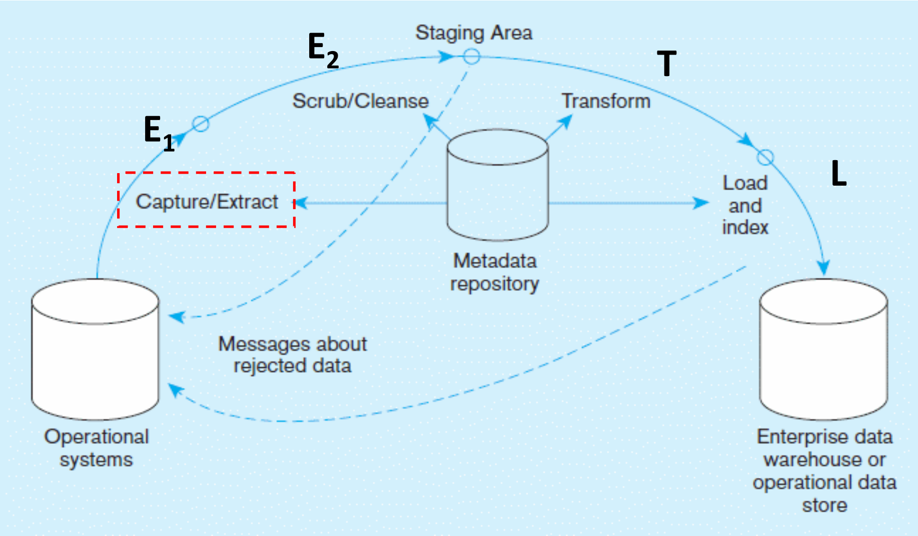
Figure 1. Magic Quadrant for Data Integration Tools



Source: Gartner (August 2019)

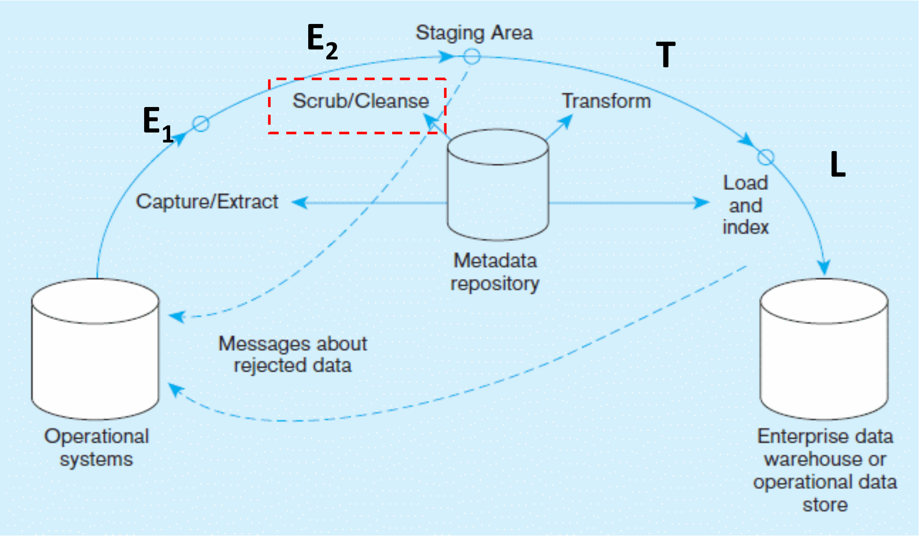
… and small ETL jobs can be performed with SQL and (say) Python scripts

Extract is the first phase of ETL process to capture a snapshot of chosen source data



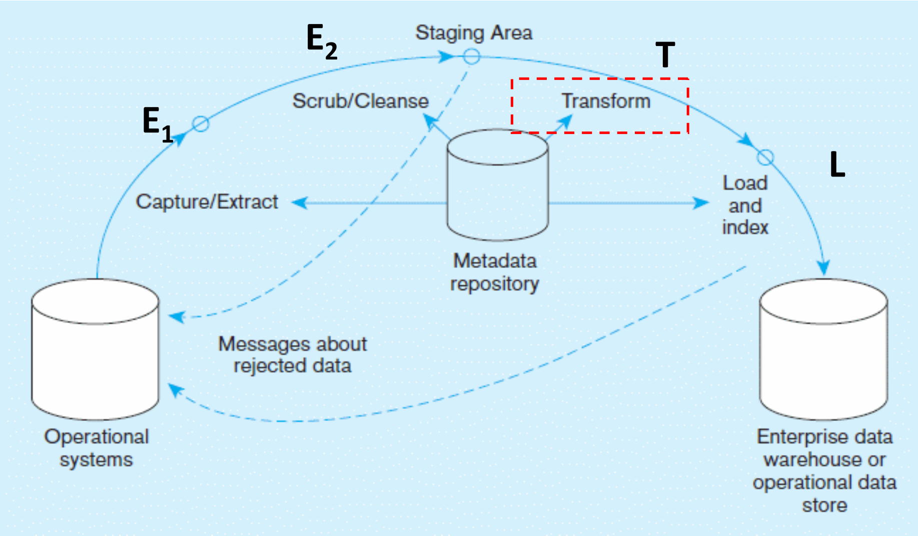
Static extract = capturing snapshot of the source data at a point in time (e.g. first load of DW)
Incremental extract = capturing changes that have occurred since the last extract

Scrub/Cleanse…uses pattern recognition and AI techniques to improve data quality



Fixing errors: misspellings, erroneous dates, incorrect field usage, mismatched addresses, missing data, duplicate data, inconsistencies
Also: decoding, reformatting, time stamping, conversion, key generation, merging, error detection/logging, locating missing data

Transform … convert data from format of operational system to format of data warehouse … at right level of granularity



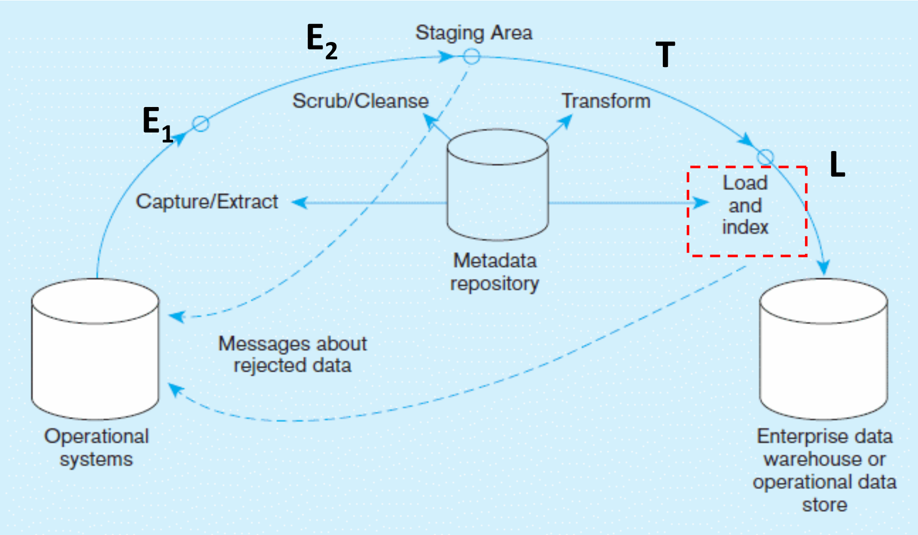
Record-level:

- Selection–data partitioning
- Joining–data combining
- Aggregation–data summarization

Field-level:

- Change units (lb to kg)
- Single-field–from one field to one field
- Multi-field–from many fields to one, or one field to many

Load/Index…place transformed data into the warehouse and create indexes



Refresh mode: bulk rewriting of target data at periodic intervals

Update mode: only changes in source data are written to data warehouse

¶Operational versus Informational Systems (e.g. OLAP)

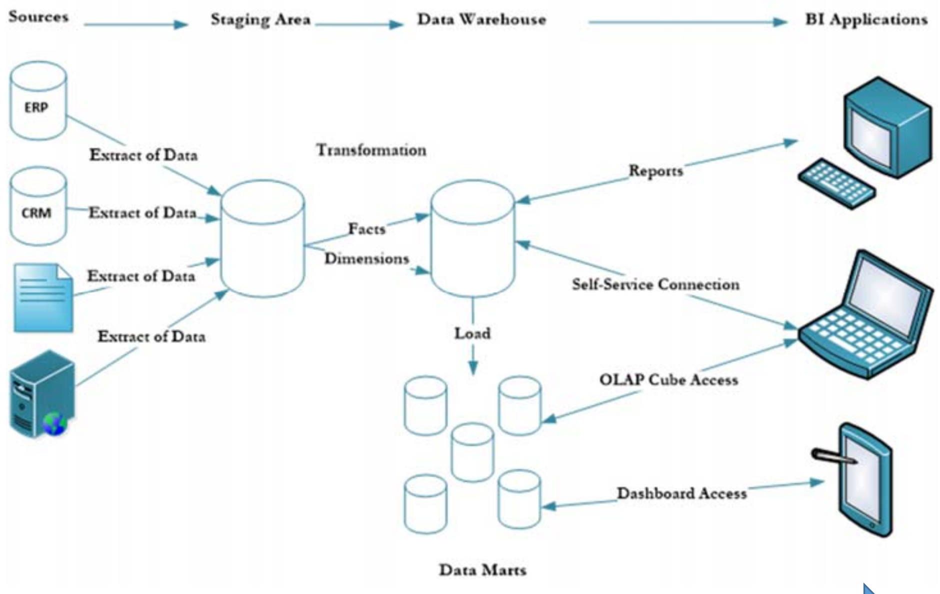
¶Data Warehousing and ETL

¶Business Intelligence, and Visualization

¶Text Parsing with Regular Expressions in SQL

¶For next time

DW is the infrastructure underpinning BI, visualization, OLAP, Analytics, Predictive Analytics, ANNs



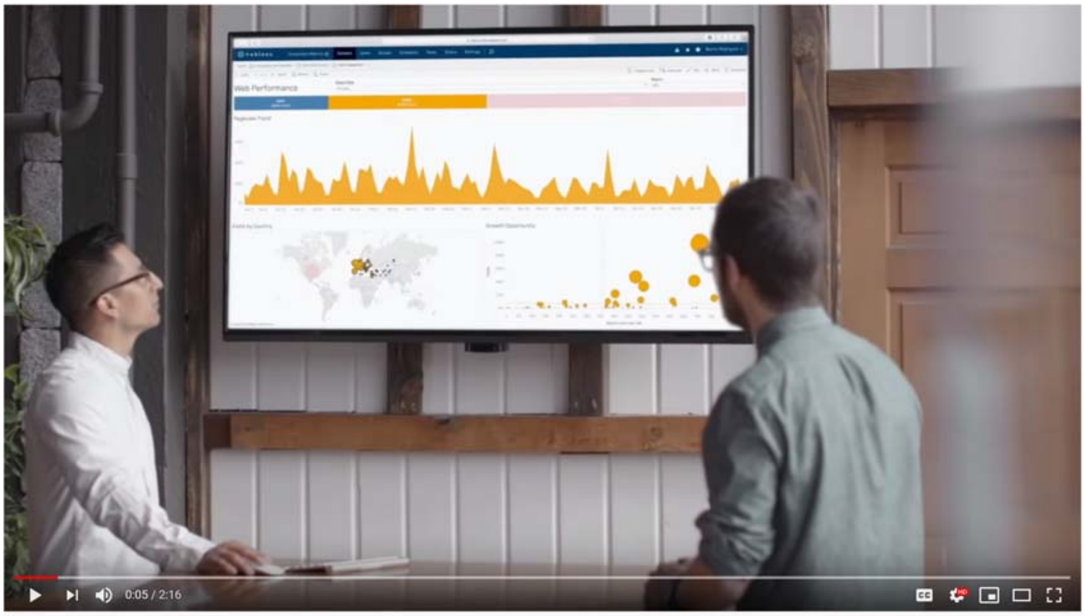
Plus unstructured 'big data' we will discuss next week

The **insights** and **information** that allow better **information products** to be developed and **better decisions** to be made

Business Intelligence (BI) builds on the data warehousing to deliver the right information to the right people

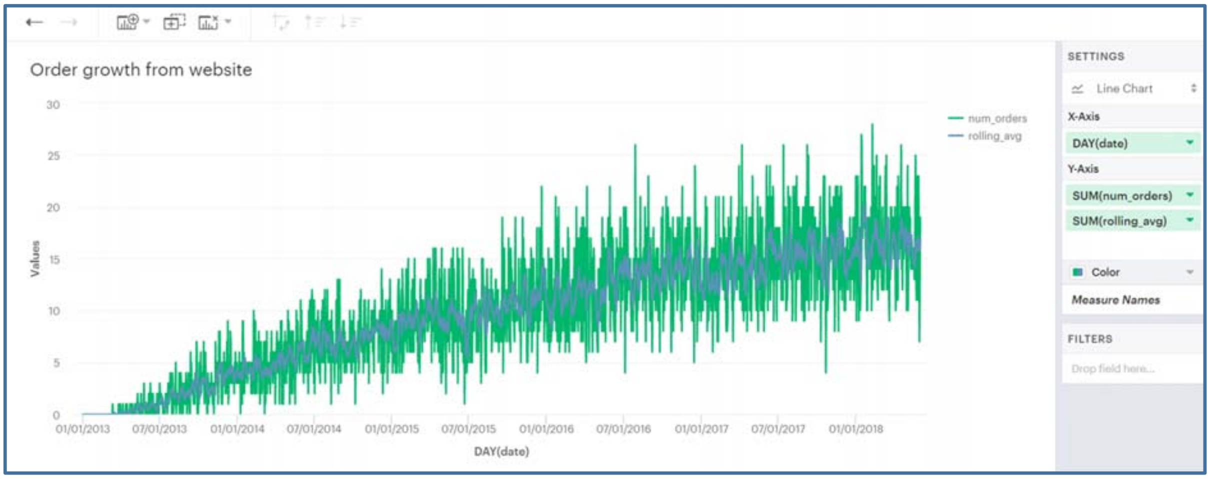
https://www.youtube.com/watch?v=hDJdkcdG1iA

Visualization allow reports and dashboards to be created that better connect with how people absorb information

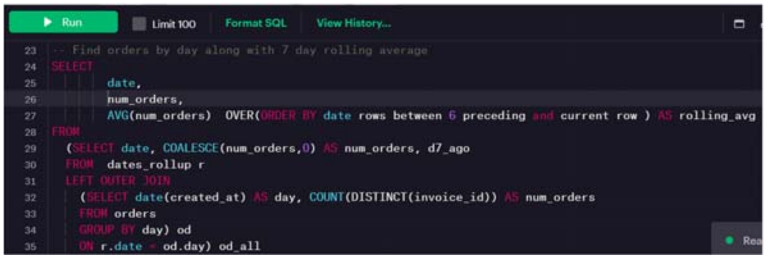


https://www.youtube.com/watch?v=YfE9jBq002s

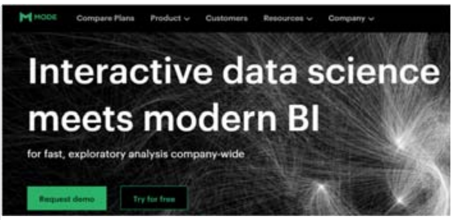
Example visualization created directly from SQL query



Line Chart
X-Axis
DAY(date)
Y-Axis
SUM(num_orders)
SUM(rolling_avg)
Color
Measure Names



```sql
-- Find orders by day along with 7 day rolling average
SELECT
        date,
        num_orders,
        AVG(num_orders)  OVER(ORDER BY date rows between 6 preceding and current row ) AS rolling_avg
FROM
  (SELECT date, COALESCE(num_orders,0) AS num_orders, d7_ago
  FROM  dates_rollup r
  LEFT OUTER JOIN
    (SELECT date(created_at) AS day, COUNT(DISTINCT(invoice_id)) AS num_orders
    FROM orders
    GROUP BY day) od
    ON r.date = od.day) od_all
```


MODE
Compare Plans
Product
Customers
Resources
Company
Interactive data science meets modern BI
for fast, exploratory analysis company-wide
Request demo
Try for free

Tool: mode.com

¶Operational versus Informational Systems (e.g. OLAP)

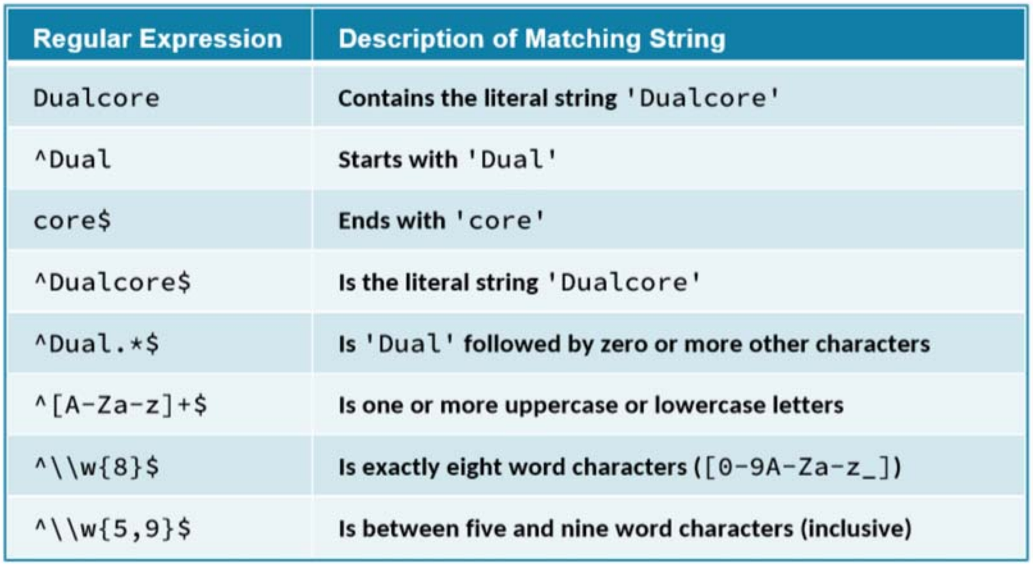
¶Data Warehousing and ETL

¶Business Intelligence, and Visualization

¶Text Parsing with Regular Expressions in SQL

¶For next time

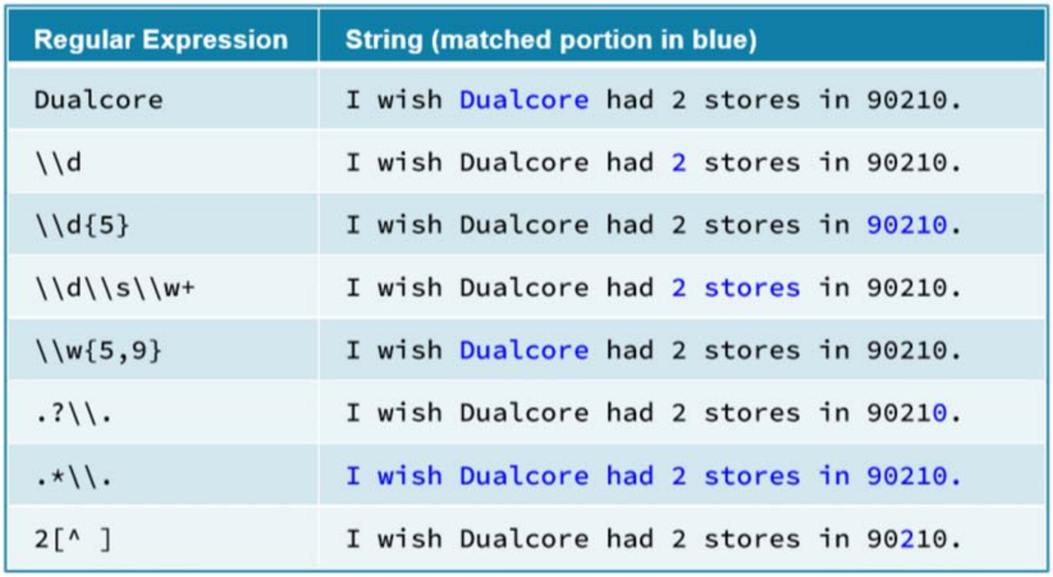
MySQL and other RDBMSs offer Regular Expressions* as a powerful means of working with data



Regular Expression	Description of Matching String
`Dualcore`	Contains the literal string `'Dualcore'`
`^Dual`	Starts with `'Dual'`
`core$`	Ends with `'core'`
`^Dualcore$`	Is the literal string `'Dualcore'`
`^Dual.*$`	Is `'Dual'` followed by zero or more other characters
`^[A-Za-z]+$`	Is one or more uppercase or lowercase letters
`^\\w{8}$`	Is exactly eight word characters (`[0-9A-Za-z_]`)
`^\\w{5,9}$`	Is between five and nine word characters (inclusive)

* Worth knowing about as Regular Expressions are also used in Python, R, Hive, Java, ….
Source: Cloudera

Regular Expressions can be used to extract or replace matched text



Regular Expression	String (matched portion in blue)
`Dualcore`	`I wish Dualcore had 2 stores in 90210.`
`\\d`	`I wish Dualcore had 2 stores in 90210.`
`\\d{5}`	`I wish Dualcore had 2 stores in 90210.`
`\\d\\s\\w+`	`I wish Dualcore had 2 stores in 90210.`
`\\w{5,9}`	`I wish Dualcore had 2 stores in 90210.`
`.?\\.`	`I wish Dualcore had 2 stores in 90210.`
`.*\\.`	`I wish Dualcore had 2 stores in 90210.`
`2[^ ]`	`I wish Dualcore had 2 stores in 90210.`

Look at this page in colour to see the matched text in blue

Source: Cloudera

Regular expressions have special characters and constructs (will understand better when you see some examples)

Character/ Construct	Description
`^`	Matches the pattern to the beginning of the value.
`$`	Matches the pattern to the end of the value.
`.`	Matches any single character.
`[charlist]`	Matches any single character listed within the brackets.
`[char1–char2]`	Matches any single character within the given range.
`|`	Separates two string patterns and matches either one.
`char*`	Matches zero or more occurrences of the character.
`[charlist]*`	Matches zero or more occurrences of the sequence of characters in brackets.

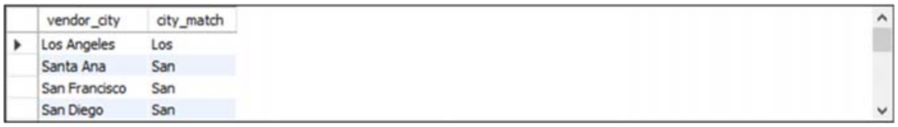
Source: Murach's MySQL 3rd Edition

Examples of the regular expression functions

Example	Result
`REGEXP_LIKE('abc123', '123')`	`1`
`REGEXP_LIKE('abc123', '^123')`	`0`
`REGEXP_INSTR('abc123', '123')`	`4`
`REGEXP_SUBSTR('abc123', '[a-z][1-9]*$')`	`c123`
`REGEXP_REPLACE('abc123', '1|2', '3')`	`abc333`

A statement that uses REGEXP_SUBSTR function

```
SELECT vendor_city,
    REGEXP_SUBSTR(vendor_city, '^SAN|LOS') AS city_match
FROM vendors
WHERE REGEXP_SUBSTR(vendor_city, '^SAN|LOS') IS NOT NULL
```

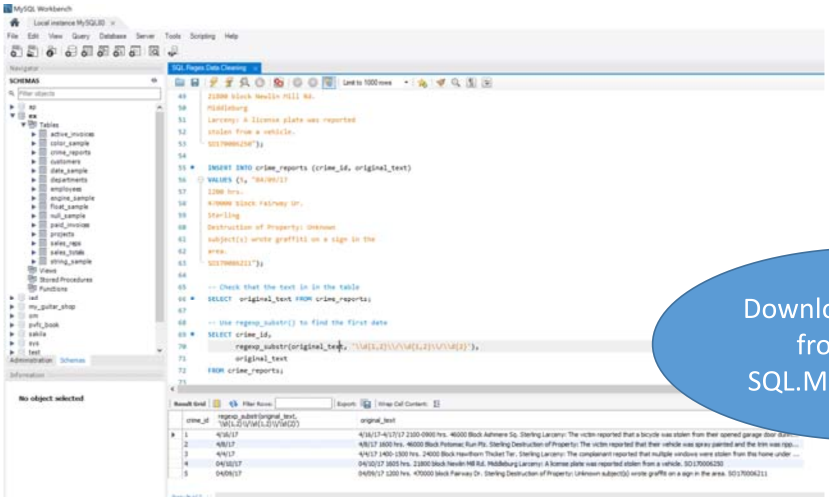


vendor_city	city_match
Los Angeles	Los
Santa Ana	San
San Francisco	San
San Diego	San

```
(12 rows)
```

Source: Murach's MySQL 3rd Edition

We explore the use of Regular Expressions to parse text data



```
21800 block Newlin Mill Rd.
Middleburg
Larceny: A license plate was reported
stolen from a vehicle.
SO170006250");

INSERT INTO crime_reports (crime_id, original_text)
VALUES (5, "04/09/17
1200 hrs.
470000 block Fairway Dr.
Sterling
Destruction of Property: Unknown
subject(s) wrote graffiti on a sign in the
area.
SO170006211");

-- Check that the text in in the table
SELECT  original_text FROM crime_reports;

-- Use regexp_substr() to find the first date
SELECT crime_id,
       regexp_substr(original_text, '\\d{1,2}\\/\\d{1,2}\\/\\d{2}'),
       original_text
FROM crime_reports;
```

crime_id	regexp_substr(original_text, '\\d{1,2}\\/\\d{1,2}\\/\\d{2}')	original_text
1	4/16/17	4/16/17-4/17/17 2100-0900 hrs. 46000 Block Ashmere Sq. Sterling Larceny: The victim reported that a bicycle was stolen from their opened garage door dur...
2	4/8/17	4/8/17 1600 hrs. 46000 Block Potomac Run Plz. Sterling Destruction of Property: The victim reported that their vehicle was spray painted and the trim was ripp...
3	4/4/17	4/4/17 1400-1500 hrs. 24000 Block Hawthorn Thicket Ter. Sterling Larceny: The complainant reported that multiple windows were stolen from this home under ...
4	04/10/17	04/10/17 1605 hrs. 21800 block Newlin Mill Rd. Middleburg Larceny: A license plate was reported stolen from a vehicle. SO170006250
5	04/09/17	04/09/17 1200 hrs. 470000 block Fairway Dr. Sterling Destruction of Property: Unknown subject(s) wrote graffiti on a sign in the area. SO170006211

Download starter file from Canvas SQL.Monroe.911.sql

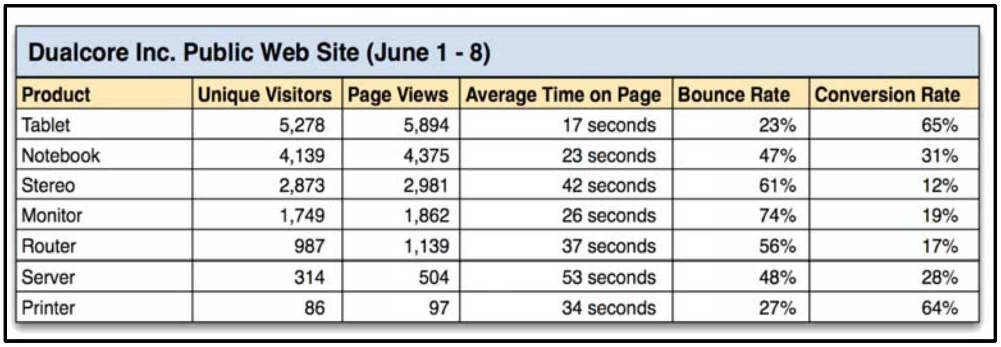
Some Regular Expression Resources
https://dev.mysql.com/doc/refman/8.0/en/regexp.html
http://www.mysqltutorial.org/mysql-regular-expression-regexp.aspx
http://php-regex.blogspot.com/2008/01/mysql-regular-expressions-cheat-sheet.html
Code testers: https://regexr.com/

You will also need other string functions to complete homework. See appendix and online documentation

Source: Murach's MySQL 3rd Edition

Use Case: Log File Analytics

- ¶ Because Hive is flexible in its data format, it can be used to store non-traditional tables e.g. web log files
- ¶ Hive allows you to treat a directory of log files like a table
  - Allows SQL-like queries against raw data using a built-in RegexSerDe



Dualcore Inc. Public Web Site (June 1 - 8)

Product	Unique Visitors	Page Views	Average Time on Page	Bounce Rate	Conversion Rate
Tablet	5,278	5,894	17 seconds	23%	65%
Notebook	4,139	4,375	23 seconds	47%	31%
Stereo	2,873	2,981	42 seconds	61%	12%
Monitor	1,749	1,862	26 seconds	74%	19%
Router	987	1,139	37 seconds	56%	17%
Server	314	504	53 seconds	48%	28%
Printer	86	97	34 seconds	27%	64%

- ¶ For example, from a directory full of web log files

```
SELECT COUNT(*) FROM logs
    WHERE date = '10/May/2020' AND url = '/product/foo'
    GROUP BY ip_address
```

¶Operational versus Informational Systems (e.g. OLAP)

¶Data Warehousing and ETL

¶Business Intelligence, and Visualization

¶Text Parsing with Regular Expressions in SQL

¶For next time

For next time

¶ Attend Thursday workshop (optional) if you think you might need more support before completing the homework

¶ Check out readings for this session and next

¶ Submit Homework Assignment #4 (team)

- Covers concepts discussed today (details on Canvas)
- Submit via Canvas by 10pm on Sunday

¶ Final exam 2hr 30min window from 9am 22/10 – 9am 23/10

- Open book and notes
- Covers all content of course (but less from Session 5)
- More emphasis on “hands on” topics than last year

Appendix

1. Some examples of String Function in MySQL

See https://dev.mysql.com/doc/refman/8.0/en/string-functions.html for the definitive documentation

2. Few extra Data Warehouse concepts

String function examples

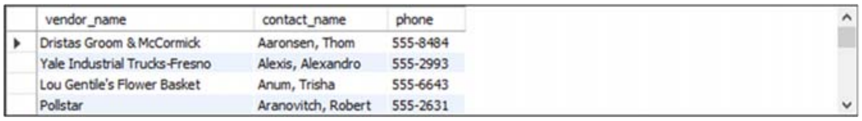
Function	Result
`CONCAT('Last', 'First')`	`'LastFirst'`
`CONCAT_WS(', ', 'Last', 'First')`	`'Last, First'`
`LTRIM('  MySQL  ')`	`'MySQL  '`
`RTRIM('  MySQL  ')`	`'  MySQL'`
`TRIM('  MySQL  ')`	`'MySQL'`
`TRIM(BOTH '*' FROM '****MySQL****')`	`'MySQL'`
`LOWER('MySQL')`	`'mysql'`
`UPPER('ca')`	`'CA'`
`LEFT('MySQL', 3)`	`'MyS'`
`RIGHT('MySQL', 3)`	`'SQL'`

String function examples (continued)

Function	Result
SUBSTRING('(559) 555-1212', 7, 8)	'555-1212'
SUBSTRING_INDEX('http://www.murach.com', '.', -2)	'murach.com'
LENGTH('MySQL')	5
LENGTH(' MySQL ')	9
LOCATE('SQL', ' MySQL')	5
LOCATE('-', '(559) 555-1212')	10
REPLACE(RIGHT('(559) 555-1212', 13),') ', '-')	'559-555-1212'
INSERT("MySQL", 1, 0, "Murach's ")	"Murach's MySQL"
INSERT('MySQL', 1, 0, 'Murach''s ')	"Murach's MySQL"

A SELECT statement that uses three functions

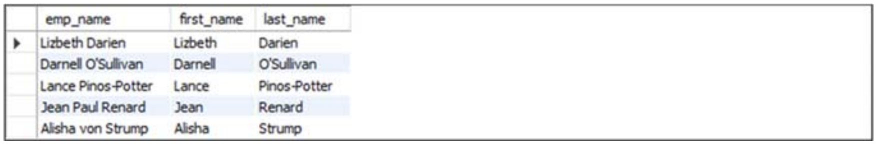
```
SELECT vendor_name,
       CONCAT_WS(', ', vendor_contact_last_name,
                 vendor_contact_first_name) AS contact_name,
       RIGHT(vendor_phone, 8) AS phone
FROM vendors
WHERE LEFT(vendor_phone, 4) = '(559'
ORDER BY contact_name
```



vendor_name	contact_name	phone
Dristas Groom & McCormick	Aaronsen, Thom	555-8484
Yale Industrial Trucks-Fresno	Alexis, Alexandro	555-2993
Lou Gentile's Flower Basket	Anum, Trisha	555-6643
Pollstar	Aranovitch, Robert	555-2631

How to use the SUBSTRING_INDEX function to parse a string

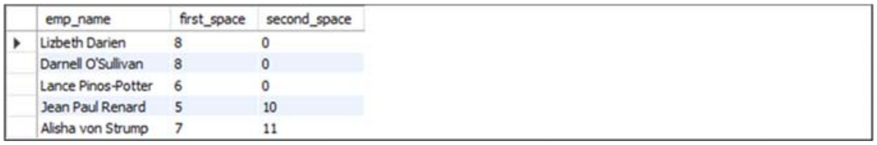
```
SELECT emp_name,
    SUBSTRING_INDEX(emp_name, ' ', 1) AS first_name,
    SUBSTRING_INDEX(emp_name, ' ', -1) AS last_name
FROM string_sample
```



emp_name	first_name	last_name
Lizbeth Darien	Lizbeth	Darien
Darnell O'Sullivan	Darnell	O'Sullivan
Lance Pinos-Potter	Lance	Pinos-Potter
Jean Paul Renard	Jean	Renard
Alisha von Strump	Alisha	Strump

How to use the LOCATE function to find a character in a string

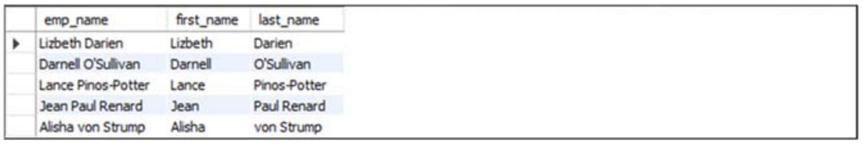
```
SELECT emp_name,
    LOCATE(' ', emp_name) AS first_space,
    LOCATE(' ', emp_name, LOCATE(' ', emp_name) + 1)
    AS second_space
FROM string_sample
```



emp_name	first_space	second_space
Lizbeth Darien	8	0
Darnell O'Sullivan	8	0
Lance Pinos-Potter	6	0
Jean Paul Renard	5	10
Alisha von Strump	7	11

How to use the SUBSTRING function to parse a string

```
SELECT emp_name,
    SUBSTRING(emp_name, 1, LOCATE(' ', emp_name) - 1)
    AS first_name,
    SUBSTRING(emp_name, LOCATE(' ', emp_name) + 1)
    AS last_name
FROM string_sample
```



emp_name	first_name	last_name
Lizbeth Darien	Lizbeth	Darien
Darnell O'Sullivan	Darnell	O'Sullivan
Lance Pinos-Potter	Lance	Pinos-Potter
Jean Paul Renard	Jean	Paul Renard
Alisha von Strump	Alisha	von Strump

Several trends in organizations provide motivation for building data warehouses

Drivers of Data warehouse adoption

No single system of record

- No organization has only one database
- Heterogeneous needs for data in different operational settings
- Large organizations with complex histories (e.g. M&A activity)

Systems not synchronized

- Metadata may not be controlled and coordinated across databases
- Data value for same attribute may not agree due to different applications and update cycles

Need for balanced set of KPIs

Managers need holistic view of organization's performance, across **Financial, HR, Customer Satisfaction, Product Quality,** and other dimensions supported by disparate operational systems

Customer Relationship Management

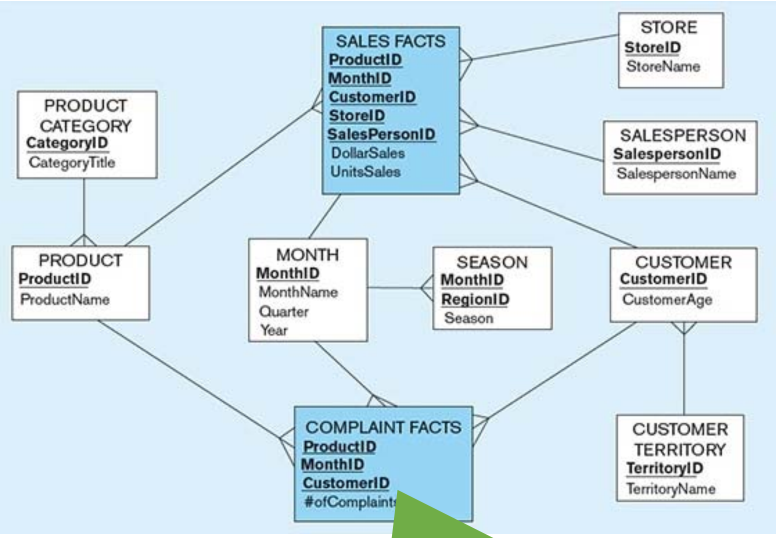
Organizations realize the value of having a total picture of the interactions with customers across all touch points (e.g. ATM, online banking, tellers, EFT, investment portfolio to help with cross-sell opportunities) supported by many systems.

Supplier Relationship Management

- Managing the supply chain is just as critical and demands information on billing, delivery performance, quality control, support, etc.
- Many organizations have several ERP and SCM systems

There are nuanced variations of the star schema

Star/Constellation schema for sales and customer service



Multiple fact tables are needed because

- Different combinations of dimensions needed (for different users)
- Different grain size required

Snowflake schema when dimension tables become elaborate

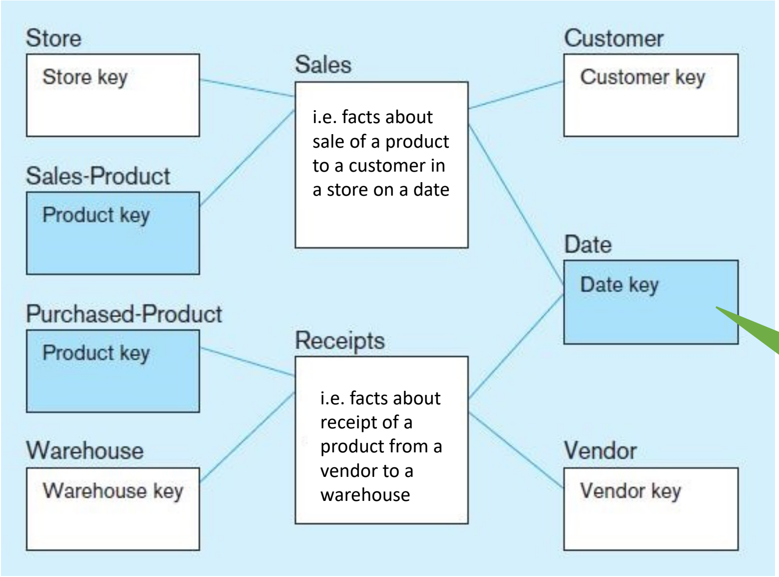
Constellation schema when more than one fact table

Duration of database: common to store 13 month or 5 quarters

Surrogate keys: dimension keys should be non-intelligent non-business related (which change)

Conformed dimensions offer potential for asking questions across data marts

Two fact tables ➔ two (connected) star schemas



Conformed dimensions – where dimension tables have the same business meaning and PK for each fact table they connect

For example: Do certain vendors recognize sales more quickly, and are they able to supply replenishments with less lead time?