

SIT742-Modern Data Science

Lecture 2 Data Acquisition and Integration

Dr Guangyan Huang
School of Information Technology

Guangyan.huang@deakin.edu.au

Tuesday, 13th March 2018

Outline

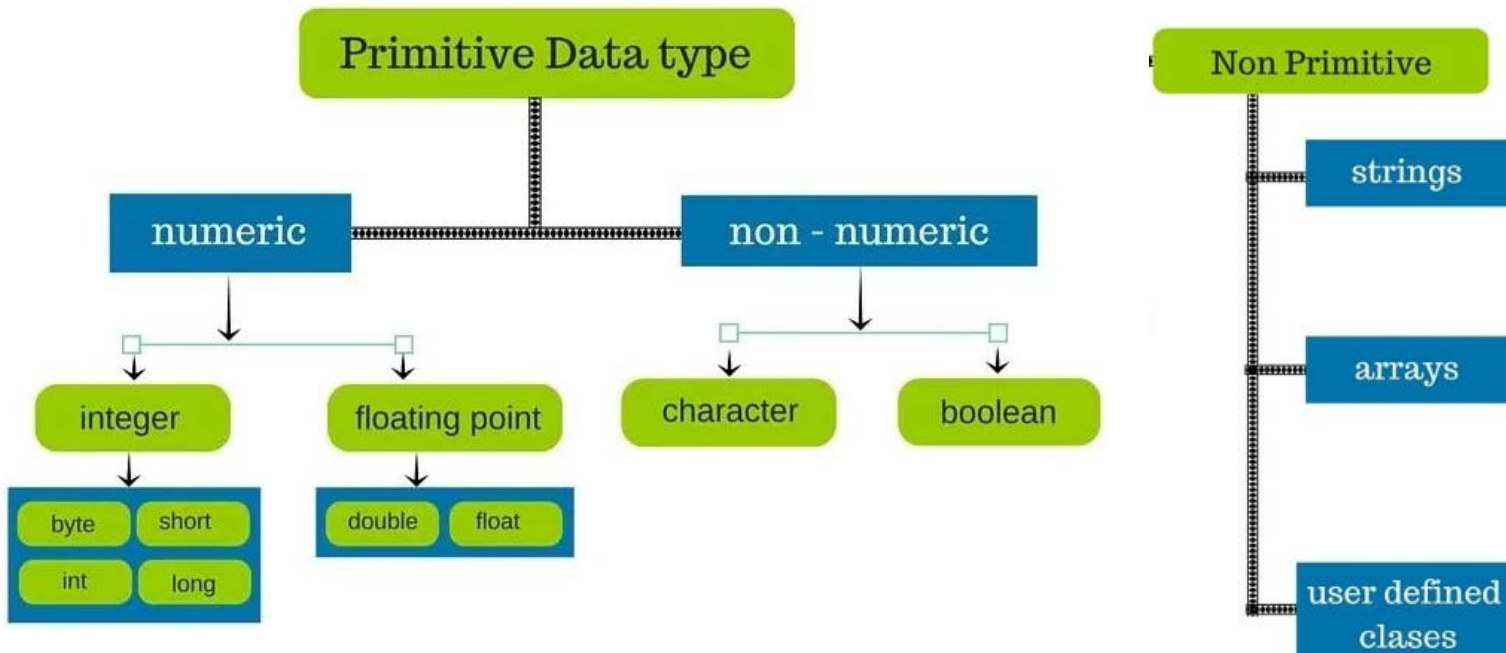
- Part I: Basic Concepts About “Data”
 - What is Data?
- Part II: Data Acquisition
 - Where do we collect, store and access data?
- Part III: Data Integration
 - How we gather data together?

Part I: Basic Concepts About “Data”

- Basic Concepts
 - Data (Content) Types
 - Data Sources
 - Datasets
 - Open Data
 - Data Storage Format

Data Types

- **Data Type**, in programming, is a classification that specifies which type of value a variable has and what type of mathematical, relational or logical operations can be applied to it without causing an error.
 - A string, for example, is a data type that is used to classify text and an integer is a data type used to classify whole numbers.
- **Data Content types**
 - Sensor stream data
 - Spatial data such as trajectories
 - Text data, such as tweets, micro blog text
 - Image/Video data, such as video from Youtube



Data Sources

- **Data Source** is a term for all the technology related to the extraction and storage of data.
- A data source can be anything from a simple text file to a big database.
- The raw data can come from observation logs, sensors, transactions, or user behavior.

Datasets

- A dataset is a collection of data and represents a logical implementation of a data source. A dataset is usually presented in a tabular form.
 - Each column represents a particular attribute, and
 - Each row corresponds to a given member of the data.
- The common features of a dataset includes:
 - Dataset characteristics (multivariate and univariate)
 - Number of instances
 - Area (life, business and many more)
 - Attribute characteristics (real, categorical and nominal)
 - Number of attributes
 - Associated tasks (classification or clustering)
 - Missing Values? (yes or no)

The diagram shows a table with 14 rows and 6 columns. A bracket above the columns is labeled 'Columns'. A bracket to the left of the rows is labeled 'Rows'. An arrow points from the word 'Values' to the cell containing '75' in the 13th row and 5th column.

id	outlook	temperature	humidity	windy	play
1	sunny	85	85	FALSE	no
2	sunny	80	90	TRUE	no
3	overcast	83	86	FALSE	yes
4	rainy	70	96	FALSE	yes
5	rainy	68	80	FALSE	yes
6	rainy	65	70	TRUE	no
7	overcast	64	65	TRUE	yes
8	sunny	72	95	FALSE	no
9	sunny	69	70	FALSE	yes
10	rainy	75	80	FALSE	yes
11	sunny	75	70	TRUE	yes
12	overcast	72	90	TRUE	yes
13	overcast	81	75	FALSE	yes
14	rainy	71	91	TRUE	no

Open Data

- Open data is data that can be used, reused and redistributed freely by anyone for any purpose.
- A short list of repositories and databases for open data:
 - Data hub: <http://datahub.io>
 - Book-Crossing Dataset: <http://www.informatik.uni-freiburg.de/~cziegler/BX/>
 - World Health Organization: <http://www.who.int/research/en/>
 - The World Bank: <http://data.worldbank.org/>
 - Scientific Data from University of Muenster: <http://data.uni-muenster.de/>
 - ACM-KDD Cup: <http://www.sigkdd.org/kddcup/index.php>
 - Kaggle: <http://www.kaggle.com/competitions>

Data Storage/Access Format

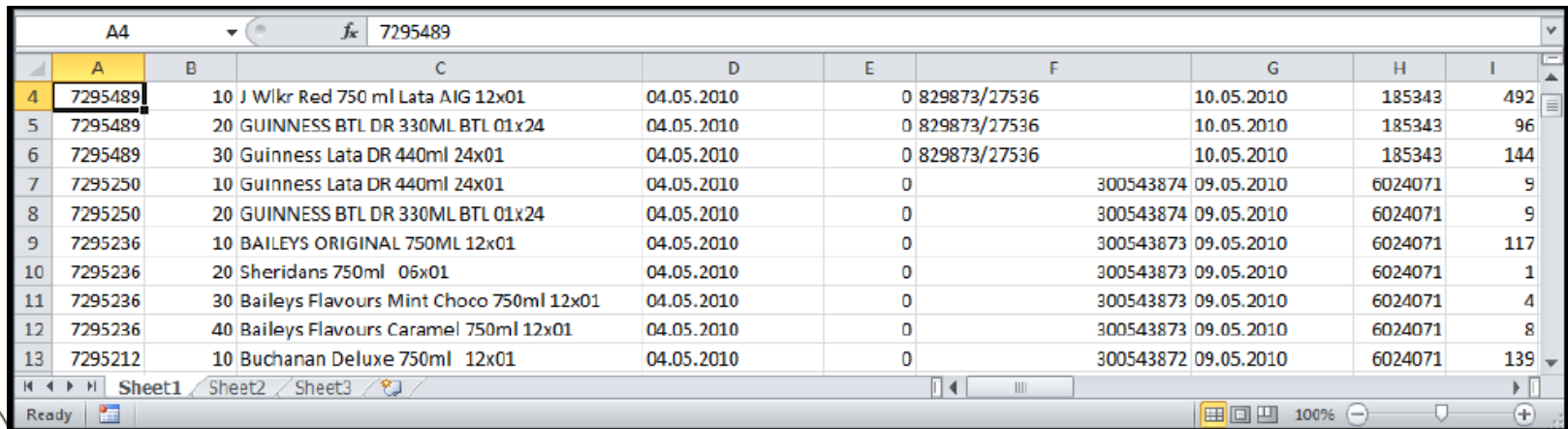
- Text files, such as CSV, XML and JSON
- Excel files
- SQL databases
- NoSQL (Not only SQL) databases
- Multimedia files, such as audio, image and video.

Data Storage/Access Format- Text files

- Text files are commonly used for the storage of data because it is easy to transform into different formats and its is often easier to recover and continue processing the remaining contents than with other formats.
- Big amounts of data come in text format from logs, sensors, e-mails, and transactions.
- There are several formats for text files such as CSV, XML and JSON.

Data Storage/Access Format-Excel Files

- Excel files
 - MS-Excel is probably the most used and also the most underrated data analysis tool. In fact, Excel has some features like filtering, aggregation functions and using Visual Basic for Application (VBA) you can make queries using SQL with the sheets or with an external database:

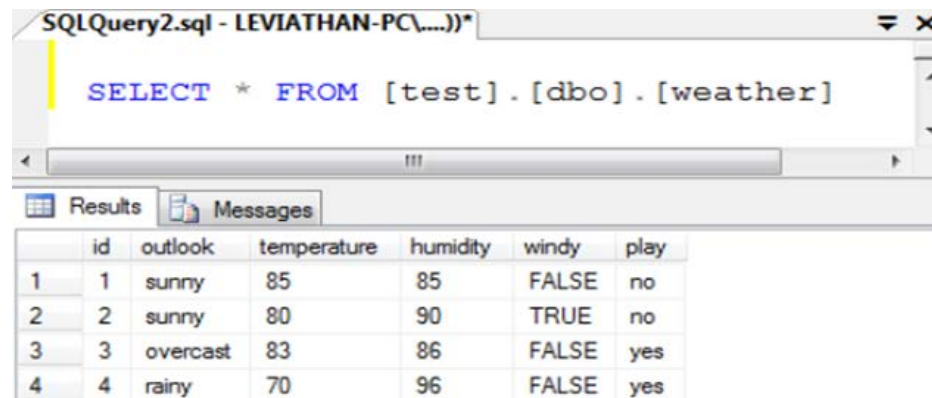


The screenshot shows an Excel spreadsheet with the following data:

	A	B	C	D	E	F	G	H	I
4	7295489	10	J Wlkr Red 750 ml Lata AIG 12x01	04.05.2010	0	829873/27536	10.05.2010	185343	492
5	7295489	20	GUINNESS BTL DR 330ML BTL 01x24	04.05.2010	0	829873/27536	10.05.2010	185343	96
6	7295489	30	Guinness Lata DR 440ml 24x01	04.05.2010	0	829873/27536	10.05.2010	185343	144
7	7295250	10	Guinness Lata DR 440ml 24x01	04.05.2010	0	300543874	09.05.2010	6024071	9
8	7295250	20	GUINNESS BTL DR 330ML BTL 01x24	04.05.2010	0	300543874	09.05.2010	6024071	9
9	7295236	10	BAILEYS ORIGINAL 750ML 12x01	04.05.2010	0	300543873	09.05.2010	6024071	117
10	7295236	20	Sheridans 750ml 06x01	04.05.2010	0	300543873	09.05.2010	6024071	1
11	7295236	30	Baileys Flavours Mint Choco 750ml 12x01	04.05.2010	0	300543873	09.05.2010	6024071	4
12	7295236	40	Baileys Flavours Caramel 750ml 12x01	04.05.2010	0	300543873	09.05.2010	6024071	8
13	7295212	10	Buchanan Deluxe 750ml 12x01	04.05.2010	0	300543872	09.05.2010	6024071	139

Data Storage/Access Format – SQL Databases

- A **database** is an organized collection of data.
- **Structured Query Language (SQL)** is a database language for managing and manipulating data in Relational Database Management Systems (RDBMS).
- **The Database Management Systems (DBMS)** is responsible for maintaining the integrity and security of stored data, and for recovering information if the system fails.
- The data is organized in schemas (database) and divided into tables related by logical relationships and we can retrieve the data by making SQL queries. SQL language is split into two subsets of instructions:
 - **Data Definition Language (DDL)**: allows us to create, delete and alter database tables.
 - **Data Manipulation Language (DML)**: enables to access and manipulate data.



The screenshot shows a window titled "SQLQuery2.sql - LEVIATHAN-PC\\....)*". The query editor contains the SQL statement: `SELECT * FROM [test].[dbo].[weather]`. Below the editor, the "Results" tab is active, displaying a table with 7 columns: id, outlook, temperature, humidity, windy, and play. The table contains 4 rows of data.

	id	outlook	temperature	humidity	windy	play
1	1	sunny	85	85	FALSE	no
2	2	sunny	80	90	TRUE	no
3	3	overcast	83	86	FALSE	yes
4	4	rainy	70	96	FALSE	yes

Data Storage/Access Format – NoSQL Databases

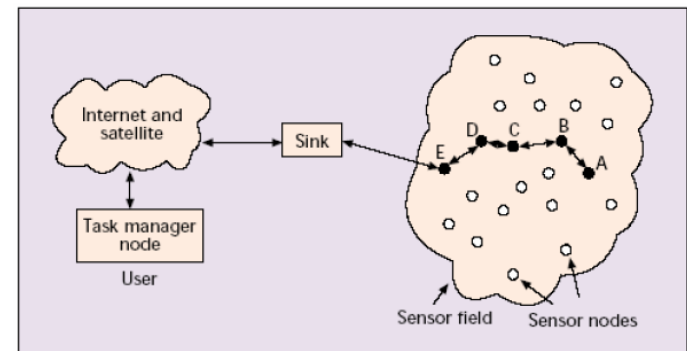
- **NoSQL (Not only SQL) databases** is a term used in several technologies where the nature of the data does not require a relational model, which allow working with a huge quantity of data, higher availability, scalability and performance.
- The most common types of NoSQL data stores are:
 - **Document store:** data is stored and organized as a collection of documents. The model schema is flexible and each collection can handle any number of fields.
 - **Key-value store:** data is stored as key-value pairs without a predefined schema. Values are retrieved by their keys.
 - **Graph-based store:** data is stored in graph structures with nodes, edges and properties using the computer science graph theory for storing and retrieving data.

Data Storage/Access Format – Multimedia

- Multimedia
 - The increasing number of mobile devices makes a priority of data analysis acquire the ability to extract semantic information from multimedia data sources.
 - Data sources include directly perceivable media such as audio, image and video. Some applications for these kinds of data source are:
 - Content-based image
 - Content-based video retrieval
 - Movie and video classification
 - Face recognition
 - Speech recognition
 - Audio and music classification

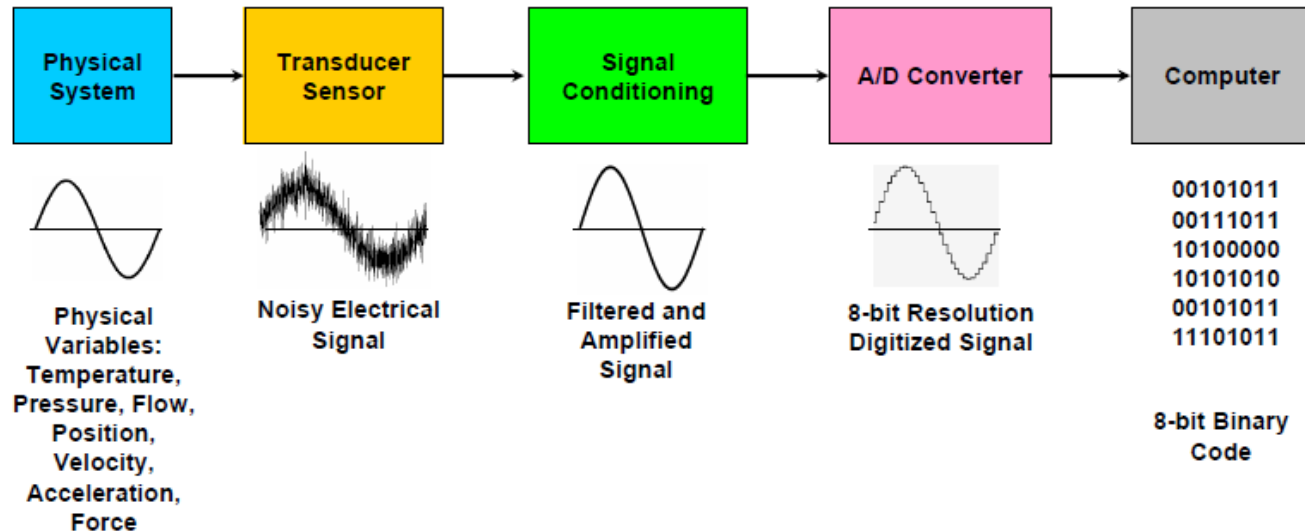
Part II: Data Acquisition

- **Data Acquisition** is about how to collect and transform raw data into a standard format, such as Comma-Separated Values (CSV) format, XML (Extensible Markup Language) or JavaScript Object Notation (JSON).
- Data Collection
 - Sensor data collection/store example
- Data Access
 - Read/write text files
 - Interact with database
 - Interact with Web APIs
 - Web scraping example



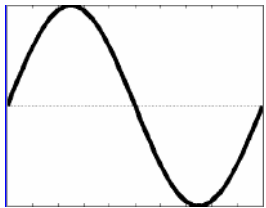
Sensor Data Collection - Mechanism

- Data Formats and Data Sources:
 - **Sensor** which measure physical variables such as temperature, pressure and motion (displacement, velocity, and acceleration)
 - **Signal conditioning**, to convert the sensor outputs into signals readable by the analog input board (A/D).
 - **An A/D board**, to convert these signals into digital format usable.
 - **A computer** with the appropriate application software to process, analyze and log the data to disks. such software may also provide a graphical display of the data.
 - **An output interface**, to provide an appropriate process control response.

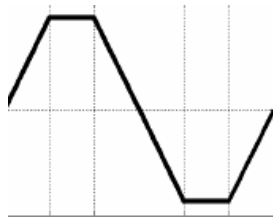


Sensor Data Collection – Sampling Rates

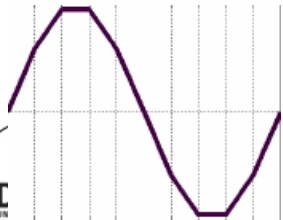
- Oversampling will provide a true picture of the time course of the event being studied but too much oversampling will result in very large data files.
- Undersampling Example



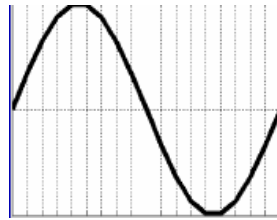
Analog Waveform



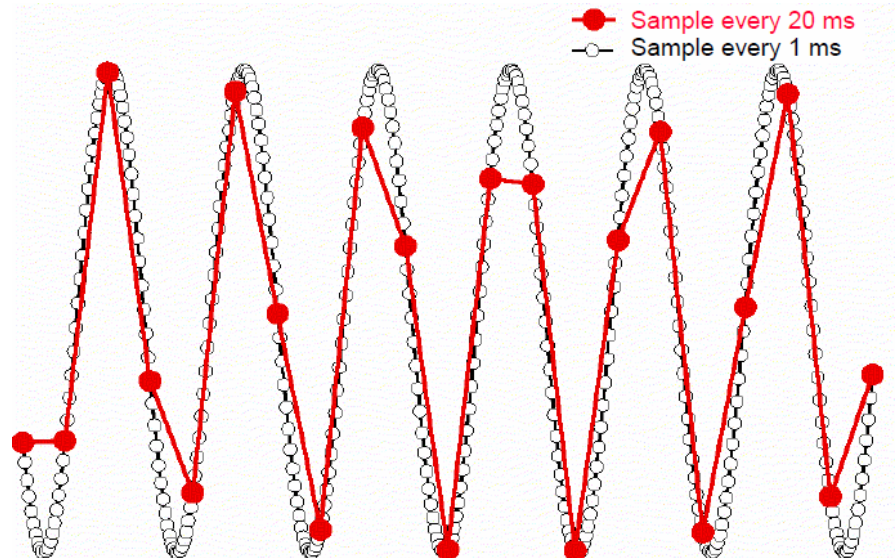
4 samples per cycle



8 samples per cycle



16 samples per cycle



Undersampling Example

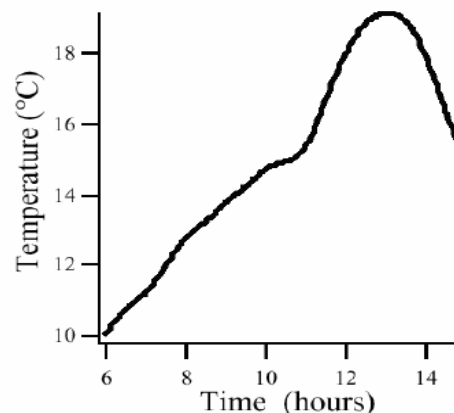
Reducing store space for sensor data – optimal sampling rates

- **Regular interval sampling:** Recording the ambient temperature during the day manually at intervals of 30 minutes.
- **Discrete sampling based on change:** the A/D board only indicates a change in it when a difference greater than 1°C is observed.

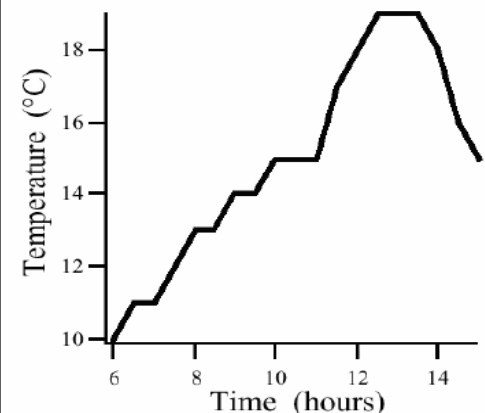
Time	Temperature(°C)	Time	Temperature (°C)
06:00	10	10:30	15
06:30	11	11:00	15
07:00	11	12:00	17
07:30	12	12:30	18
08:00	13	13:00	19
08:30	13	13:30	19
09:00	14	14:00	19
09:30	14	14:30	18
10:00	15	15:00	16
		15:30	15

Regular interval sampling

Continuous, analog measurements of temperature



Discrete digital measurements of temperature



Discrete sampling based on change

Reducing store space for sensor data – Budget Bits

- In a real A/D board the total measurement range is divided into a fixed number of possible values.
- Commonly used values are:
 - 8 bits = 2^8 (256 values)
 - 12 bits = 2^{12} (4,096 values)
 - 16 bits = 2^{16} (65,536 values)

Data Access

- **Reading text files** and other more efficient on-disk formats
 - `read_csv`, `read_table`, `read_excel`, `read_html`
- **Loading data from databases**
 - SQL-based relational databases (such as SQL Server, PostgreSQL, and MySQL) are also in wide use, and many alternative databases have become quite popular. The choice of database is usually dependent on the performance, data integrity, and scalability needs of an application.
- **Interacting with network sources like web APIs**
 - Many websites have public APIs providing data feeds via JSON or some other format. There are a number of ways to access these APIs from Python; one easy-to-use method is the `requests` package.

Web Scraping Example

- Web scraping
 - When we want to obtain data, a good place to start is on the Web. Web scraping refers to an application that processes the HTML of a webpage to extract data for manipulation.
 - Web scraping applications will simulate a person viewing a website with a browser
 - An example: get the current gold price from the website www.gold.org as shown below:

The screenshot shows a Mozilla Firefox browser window displaying the World Gold Council website. The address bar shows www.gold.org. The page features a dark header with the World Gold Council logo, a search bar, and navigation links (Home, About us, Investment, Jewellery, Government Affairs, Technology, Media, About gold, Events). A green arrow points to the 'Ask' price of US\$ 1,573.20 in the Gold Spot Price section.

Gold Spot Price	Ask US\$ 1,573.20
Change > US\$/oz	Mid US\$ 1,572.80
Updated: 08 Apr at 23:51	Bid US\$ 1,572.40

- Then, we extract Gold Spot Price in the HTML tag:
 - `<td class="value" id="spotpriceCellAsk">1,573.85</td>`

Web Scraping Example (cont.)

- WebScraping.py
 - https://github.com/hmcuesta/PDA_Book/tree/master/Chapter2
- The output:
 - 11:35:02AM, 1481.25
 - 11:36:03AM, 1481.26
 - 11:37:02AM, 1481.28

```
from bs4 import BeautifulSoup
import urllib.request
from time import sleep
from datetime import datetime
def getGoldPrice():
    url = "http://gold.org"
    req = urllib.request.urlopen(url)
    page = req.read()
    scraping = BeautifulSoup(page)
    price= scraping.findAll
    ("td",attrs={"id":"spotpriceCellAsk"})[0]
    .text
    return price

with open("goldPrice.out","w") as f:
    for x in range(0,60):
        sNow = datetime.now().strftime("%I:%M:%S%p")
        f.write("{0}, {1} \n ".format(sNow, getGoldPrice()))
        sleep(59)
```

Part III: Big Data Integration

- Big Data Integration Challenges
- Traditional Data Integration
- Modern Approaches for Big Data Integration
 - Web data integration case study

Big Data Integration Challenges

- Big data integration differs from traditional data integration along many dimensions, paralleling the dimensions.
- Big data is characterized as differing from traditional databases in 4 aspects:
 - **Volume:** data sources contain a huge volume of data
 - the number of data sources has grown to be in the millions;
 - even for a single domain, the number of sources has grown to be in the tens to hundreds of thousands.
 - **Velocity:** As a direct consequence of the rate at which data are being collected and continuously made available, many of the data sources are quite dynamic, and the number of data sources is also rapidly exploding.

Big Data Integration Challenges (cont.)

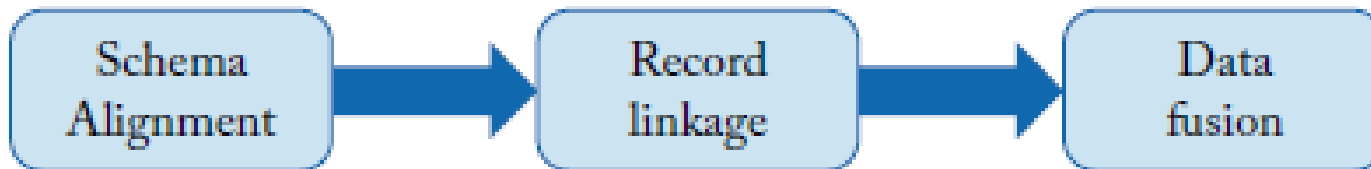
– **Variety:**

- Data sources from different domains are naturally diverse since they refer to different types of entities and relationships, which often need to be integrated to support complex applications.
- Data sources even in the same domain are quite heterogeneous both at the schema level regarding how they structure their data and at the instance level regarding how they describe the same real-world entity, exhibiting considerable variety even for substantially similar entities.
- The domains, source schemas, and entity representations evolve over time, adding to the diversity and heterogeneity that need to be handled in big data integration.

– **Veracity:** Data sources are of widely differing qualities, with significant differences in the coverage, accuracy, and timeliness of data provided.

Data Integration

- Traditional Data Integration Architecture [2]
 - Schema Alignment,
 - Record Linkage, and
 - Data Fusion
- To Resolve Problems:
 - Semantic ambiguity
 - Instance representation ambiguity
 - Data inconsistency



Why we need schema alignment

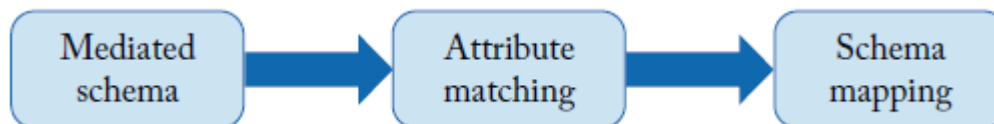
- The first component of data integration is *schema alignment*. There can be thousands to millions of data sources in the same domain, but they often describe the domain using different schemas.
- For example, different sources describe the flight domain using very different schemas:
 - they contain different numbers of tables and different numbers of attributes;
 - they may use different attribute names for the same attribute
 - (e.g., Scheduled Arrival Date in Airline2.Flight vs. Scheduled in Airport3.Arrivals);
 - they may apply different semantics for attributes with the same name
 - (e.g., Arrival Time may mean landing time in one source and arrival-at-gate time in another source).
- To integrate data from different sources, the first step is to align the schemas and understand which attributes have the same semantics and which ones do not.

Why we need schema alignment for big data

- The big data environment makes the problem significantly harder. Instead of integrating data within an organization, the goal is often to integrate structured data from the web, either in the form of
 - deep web data, or
 - web tables or web lists.
- The number of data sources available for integration explodes from hundreds of sources to millions of sources; and the schemas of the data are constantly changing.
 - Such *volume and velocity of big data also increase the variety of data* remarkably, calling for new techniques and infrastructures to resolve the schema heterogeneity.

Schema Alignment

- Definition of Schema Alignment
 - Consider a set of source schemas in the same domain, where different schemas may describe the domain in different ways. *Schema alignment generates three outcomes.*
 - Mediated schema provides a unified view of the disparate sources
 - Attribute matching for corresponding attributes in different sources
 - Schema mapping between source schema and mediated schema to specify semantic relationships between them.



Schema Alignment for Big Data

- **Addressing the variety and velocity challenges**
 - Probabilistic schema alignment
 - Pay-as-you-go user feedback
 - Benefit of Confirming a Matching
 - Approximating the Benefit
- **Addressing the veracity and volume challenges**
 - Integrating deep web data (*The deep web refers to data stored in underlying databases and queried by HTML forms.*)
 - Keywords search on web tables
 - Finding related web tables
 - Extracting knowledge from web tables

Probabilistic Schema Alignment

- First, a probabilistic mediated schema can be built to capture the uncertainty on how to model the domain.
 - Each possible mediated schema in the probabilistic mediated schema represents one way of clustering the source attributes, where attributes in the same cluster are considered as having the same semantics
- Second, a probabilistic schema mapping can be built between each source schema and each possible mediated schema in the probabilistic mediated schema.
 - A probabilistic schema mapping contains a set of attribute matchings, each describing one possible matching between source attributes and the attribute clusters in a mediated schema

Possible Mediated Schema	Probability
Med3({FN}, {DT, DGT}, {TT}, {AT, AGT}, {LT})	0.6
Med4({FN}, {DT, TT}, {DGT}, {AT, LT}, {AGT})	0.4

Possible Mapping Between S1 and Med3	Probability
M_1 {(FN, FN), (DGT, DDGT), (TT, TT), (AGT, AAGT), (LT, LT)}	0.64
M_2 {(FN, FN), (DGT, DDGT), (TT, TT), (AGT, LT), (LT, AAGT)}	0.16
M_3 {(FN, FN), (DGT, TT), (TT, DDGT), (AGT, AAGT), (LT, LT)}	0.16
M_4 {(FN, FN), (DGT, TT), (TT, DDGT), (AGT, LT), (LT, AAGT)}	0.04

Integrating deep web data





- There are two common approaches for offering access to deep web data.
 - a data integration solution that builds vertical search engines.
 - *surfacing*. pre-computing the most relevant form submissions for all interesting HTML forms and solve two key problems:
 - decide which form inputs to fill when submitting queries to a form.
 - find appropriate values to fill in these inputs.
 - enumerating the entire Cartesian product of all possible inputs can result in a very large number of URLs being generated.
 - Crawling too many URLs would drain the resources of a web crawler and pose an unreasonable load on web servers hosting the HTML forms.
- Challenge example: a particular search form on cars.com has 5 inputs and a Cartesian product yields over 240M URLs, but there are only 650K cars on sale.

Integrating deep web data (cont.)

- Now the problem of surfacing a deep-web site can be divided into two subproblems.
 - **Selecting Query Templates**
 - **Generating Input Values, Web for searching flights example**
 - there are city names such as *Las Vegas on the entrance webpage* of Orbitz.com and they can be used as input for From and To. From the results the algorithm would iteratively extract more city names, such as from connection cities.

Integrating Web Tables

- Web tables refer to relational data in tabular form on the web. This figure shows a web table about some major airlines of the world. After filtering out tables that are used for page layout or other non-relational reasons, there are 154M distinct web tables from English-language documents in Google's main index.*

The World's biggest Airlines					
Airline	Passengers (in million) 2009	Passengers (in million) 2010	Main Hub IATA code	Headquarter/ City	Country
Africa/Middle East					
Emirates Airline	27,454	31,422	Dubai International Airport	Dubai	 United Arab Emirates
Qatar Airways	10,212	12,392	Doha International Airport	Doha	 Qatar
Saudi Arabian Airlines	18,334	18,172	Jeddah-King Abdulaziz International	Jeddah	 Saudi Arabia
Asia/Pacific					
AirAsia	14,253	16,055	Kuala Lumpur International Airport	Kuala Lumpur	 Malaysia
Air China	39,841	46,241	Beijing Capital International Airport	Beijing	 China

Integrating Web Tables (cont.)

- Web tables
 - different from deep web data: they are HTML tables that are already crawlable without filling in any form.
 - different from relational databases. Web tables do not have clearly specified schemas; the semantics are often encoded in the column caption, which can sometimes be missing.
- Three techniques for leveraging web table data.
 - **keyword search on web tables**, where the goal is to return highly relevant tables in answers to a keyword search.
 - **find relevant tables**, where the goal is to return tables that have similar or complementary data to the table edited by the user, to possibly provide reference.
 - **extract knowledge from web tables**, where the goal is to extract (entity, property, value) triples that can be used to populate knowledge bases

Keywords Search on Web Tables

- The goal of keyword search on web tables is to accept user keyword queries and rank web tables by relevance.
- Challenges: keyword search does not work well.
 - **frequent words** in the webpage that contains a web table may be different from what the web table describes;
 - **attribute labels** in the web tables are extremely important to understand the table but may not appear frequently;
- Two new ranking techniques for web tables.
 - **Feature-based Rank**
 - **Schema-based Rank** (is the same as Feature Rank, except that it also includes a score indicating the coherence of a schema as a feature.)
Intuitively, a *coherent schema* is one where the attributes are all tightly related to one another.
 - For example, a schema that consists of the attributes **gate** and **terminal** is coherent, but one with **gate** and **address** is much less coherent.)

Finding Related Web Tables - Challenges

- Discovering tables in a corpus that are related to a given table is challenging for two reasons.
 - the schemas of web tables are partial at best and extremely heterogeneous.
 - In some cases, relatedness are embedded in text surrounding the tables or textual descriptions attached to them.
 - needs to consider different ways and degrees to which data can be related.

Finding Related Web Tables - Solutions

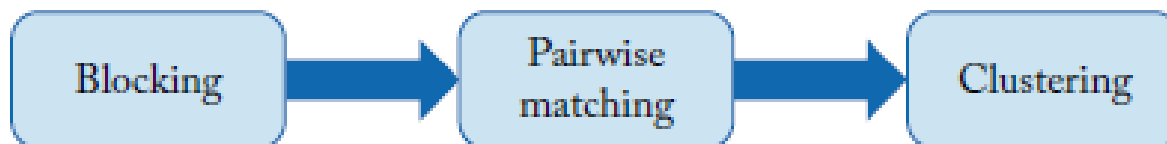
- Three criteria for entity complement tables:
 - **Entity consistency.** A related table T should have the same type of entities as T' .
 - **Entity expansion.** T should substantially add new entities to those in T' .
 - **Schema consistency.** The two tables should have similar (if not the same) schemas, thereby describing similar properties of the entities.
- Two criteria for schema complement tables.
 - **Coverage of entity set.** T should consist of most of the entities in T' , *if not all of them*.
 - **Benefit of additional attributes.** T should contain additional attributes that are not contained in T' 's *schema*.

Extracting Knowledge from Web Tables

- Web tables contain structured data that are often carefully edited and so of high quality.
 - Each row in a web table typically presents an entity,
 - Each column typically presents a property of the entities, and
 - Each cell typically presents the value for the corresponding property of the corresponding entity.
- It is desirable to extract such structured information from the web and the results can be used to facilitate search or populate knowledge bases.

Record Linkage

- Definition
 - Consider a set of data sources, each providing a set of records over a set of attributes. *Record linkage computes a partitioning of the set of records, such that each partition identifies the records that refer to a distinct entity.*
- Traditional record linkage:
 - Blocking (scalability)
 - Pairwise matching (semantics)
 - Clustering (semantics)



Record Linkage

- The goal of record linkage is to link millions of records obtained from tens to hundreds of data sources; thus, comparing every pair of records is challenging.
- The big data environment makes the record linkage problem even more challenging:
 - the number of data sources available for integration is now in the millions, a vast number of which are unstructured sources with textual data;
 - the data in these sources dynamically change, and have a lot of representational differences and errors.

Pairwise Matching

- The basic step of record linkage is *pairwise matching*, which compares a pair of records and makes a local decision of whether or not they refer to the same entity.
- ***Rule-based approaches***
 - *It is a common approach* and uses **domain knowledge** to make the local decision.
 - Advantage: the rule can be tailored to effectively deal with complex matching scenarios.
 - Disadvantage: it requires considerable **domain knowledge** as well as knowledge about the data to formulate the pairwise matching rule, rendering it ineffective when the records contain errors.

Pairwise Matching (cont.)

- ***Classification-based approaches***

- a classifier is built using positive and negative training examples, and the classifier decides whether a pair of records is a match or a non-match; it is also possible for the classifier to output a possible-match, in which case the local decision is turned over to a human.
- Advantage: do not require significant domain knowledge about the domain and the data, only knowledge of whether a pair of records in the training data refers to the same entity or not.
- Disadvantage: often requires a large number of training examples to accurately train the classifier, though active learning based variations are often effective at reducing the volume of training data needed.

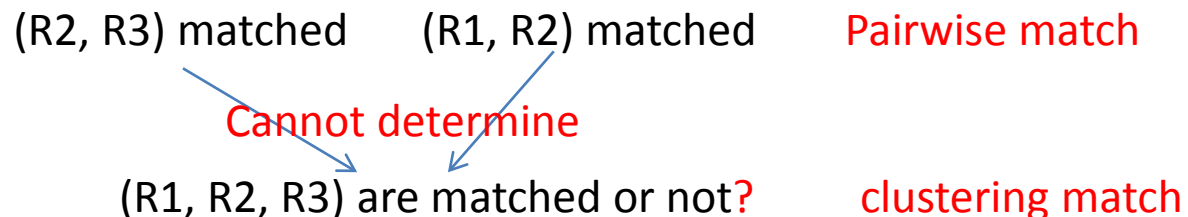
Pairwise Matching (cont.)

- ***Distance-based approaches***

- *apply distance metrics to compute* difference of corresponding attribute values and take the weighted sum as the record-level distance to compare with thresholds.
 - Levenstein distance for strings, and
 - Euclidean distance for numeric attributes
- Advantage: domain knowledge is limited to formulating distance metrics on atomic attributes for a large variety of entity domains.
- Disadvantage: it requires careful parameter tuning for weights and thresholds, although machine learning approaches can often be used to tune the parameters in a principled fashion.

Clustering

- The local decisions of match or non-match made by the pairwise matching step may not be globally consistent.
- *clustering* step is needed to reach a globally consistent decision.



Blocking

- Pairwise matching requires a quadratic number of record pair comparisons to decide record pairs are matches or not and is not efficient for a large number of records (e.g., millions).
- Blocking was proposed as a strategy to scale record linkage to large data sets.
 - Partition data into blocks and only conduct pairwise in the same block.

Record Linkages of Big Data

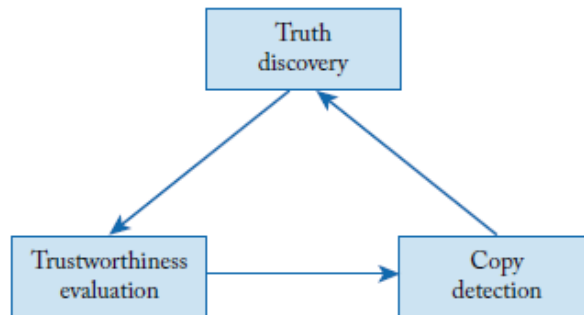
- Addressing the Volume Challenge
 - Using MapReduce to Parallelize Blocking (Load balance)
 - Meta-Blocking: Pruning Pairwise Matchings
- Addressing the Velocity Challenge
 - Incremental Record Linkage
 - The high velocity of data updates can quickly make previous linkage results obsolete. But it is expensive to perform batch record linkage each time there is a data update, so it would be ideal to perform *incremental record linkage* when data updates arrive.

Record Linkages of Big Data (cont.)

- Addressing the Variety Challenge
 - Linking text snippets to structured data
 - the text snippet does not always contain attribute names.
 - the attribute values present in the text snippet may match exactly or approximately, or even be erroneous.
 - the text snippet does not contain all the attribute values from the structured records.
 - the text snippet may contain attribute values that are not present in the structured data.
- Addressing the Veracity Challenge
 - one that focuses on out-of-date values (Probabilistic models for temporal record linkage
 - The different sources often represent the same attribute value in different ways, some provide erroneous values, and when the records are about the entities at different points in time, some of the erroneous values may just be out-of-date values.
 - another that effectively deals with erroneous values.
 - Record linkage seeks to partition the records into clusters and identify the records that refer to the same entity, *despite the multiple representations of a value, erroneous and out-of-date attribute values.*

Data Fusion

- Definition: Consider a set of data items, and a set of data sources each of which provides values for a subset of the data items. *Data fusion decides the true value(s) for each data item.*



An Example: Five data sources provide information on the scheduled departure time of five flights. False values are in italics. Only S1 provides all true values.

	S1	S2	S3	S4	S5
Flight 1	19:02	<i>19:18</i>	19:02	19:02	<i>20:02</i>
Flight 2	17:43	17:43	<i>17:50</i>	<i>17:50</i>	<i>17:50</i>
Flight 3	9:20	9:20	9:20	9:20	9:20
Flight 4	21:40	<i>21:49</i>	<i>20:33</i>	<i>20:33</i>	<i>20:33</i>
Flight 5	18:15	18:15	<i>18:22</i>	<i>18:22</i>	<i>18:22</i>

Big Data Fusion

- **Addressing the Veracity Challenge**

- Accuracy of a source
- Probability of a value being true
- Copying between sources

- **Addressing Volume Challenge**

- Mapreduce-based framework for offline fusion
- Online data fusion

- **Addressing Velocity Challenge**

- **online data fusion** (focuses on the current values)
- **temporal data fusion**, to find all correct values and their valid periods in the history, when the true values evolve over time.

- **Addressing Variety Challenge**

- **Extended data fusion**, consider each (extractor, source) pair as a whole, called a provenance, to resolve the errors resulting from schema alignment and record linkage. Enabling evaluating the quality of the sources and the quality of the extractors independently. Then, one could identify possible mistakes by the same extractor on many different sources.

Accuracy of the sources

	S1	S2	S3	S4	S5
Round 1	0.52	0.42	0.53	0.53	0.53
Round 2	0.63	0.46	0.55	0.55	0.41
Round 3	0.71	0.52	0.53	0.53	0.37
Round 4	0.79	0.57	0.48	0.48	0.31
⋮					
Round 11	0.97	0.61	0.40	0.40	0.21

References

- [1] Hector Cuesta, Dr. Sampath Kumar, *Practical Data Analysis*, 2nd Edition PACKT Publishing, 2016
- [2] Xin Luna Dong, Divesh Srivastava, *Big Data Integration*, Morgan & Claypool Publishers, 2015