WQD7007 Big Data Management

Big Data Concepts

Identification, de-identification, and re-identification

Ontologies and semantics

Identification, de-identification, and re-identification

Agenda

- Identification
  - Feature of an identifier system
  - Object identifier
    - Really bad identifier methods
    - Embedding Information in an Identifier: Not Recommended
  - One-way hashes
- De-identification
- Re-identification

Identification, de-identification and re-identification

- Data identification is certainly **the most underappreciated and least understood** Big Data issue.

- Measurements, annotations, properties, and classes of information have no informational meaning unless they are attached to an **identifier** that distinguishes one data object from all other data objects
  - It links together all of the information that has been or will be associated with the identified data object.
  - Example?

Identification, de-identification and re-identification

- Properties of identified information, including
  - uniqueness, exclusivity, completeness, authenticity, and harmonization

- Once data objects have been properly identified, they can be **de-identified** and, under some circumstances, **re-identified**
  - The ability to **de-identify** data objects confers enormous advantages when issues of **confidentiality, privacy, and intellectual property emerge**.
  - The ability to **re-identify** deidentified data objects is required for **error detection, error correction, and data validation**.

Identification, de-identification and re-identification

- A good information system is an identification system where:
  - Provide a way of **naming data objects** so that they can be **retrieved by their name**
  - Provide a way of **distinguishing each object from every other object** in the system.
- If data managers **properly** identified their data, they would be producing a collection of data objects with more informational value than many existing big data resources.

Features of an identifier system

- An **object identifier** is an **alphanumeric string** associated with the **object**.

- For many big data resources, the objects that are of greatest concern to data managers are **human beings**.
  - many Big Data resources are built to **store and retrieve** information about **individual humans**.
  - the data manager's pre-occupation with human identifiers relates to the paramount importance of establishing **human identity, with absolute certainty** (e.g., banking transactions, blood transfusions).

Features of an identifier system

One of the most important tasks for data managers is the **creation of a dependable identifier system**. The properties of a good identifier system are the following:

1. **Completeness**. Every unique object in the Big Data resource **must be assigned an identifier**.
2. **Uniqueness**. Each identifier is a **unique sequence**.
3. **Exclusivity**. Each identifier is assigned to a unique object, **and to no other object**.
4. **Authenticity**. The objects that receive identification must be **verified** as the objects that they are intended to be.
   - For example, if a young man walks into a bank and claims to be Richie, then the bank must ensure that he is, in fact, who he says he is.

Features of an Identifier System

5. **Aggregation**. The big data resource must have a mechanism to aggregate all of the data that is **properly associated with the identifier**
6. **Permanence**. The identifiers and the associated data must be **permanent**.
   - In the case of a hospital system, when the patient returns to the hospital after 30 years of absence, **the record system must be able to access his identifier and aggregate his data.**
   - Even when a patient dies, the patient's identifier must not perish.
7. **Reconciliation**. There should be a mechanism whereby the data associated with a unique, identified object in one big data resource **can be merged** with the data held in another resource, for the same unique object.

Features of an Identifier System

8. **Immutability**. In addition to being permanent (i.e., never destroyed or lost), the identifier **must never change**.
9. **Security**. The identifier system is vulnerable to malicious attack.
   - A Big Data resource with an identifier **system can be irreversibly corrupted** if the identifiers are modified.
10. **Documentation and quality assurance**. A system should be in place **to find and correct errors in the patient identifier system.**
    - Every problem and every corrective action taken must be documented and reviewed.

Object identifiers

- For computer scientists, a data object **is a holder for data values** (the so-called encapsulated data), and become **descriptors of the data** based on properties of the holder (i.e., the class of objects to which the instance belongs).
- Uniqueness is achieved when the **data object is permanently bound to its own identifier sequence**. Unique objects have three properties:
  1. An unique object can be distinguished from all other unique objects.
  2. An unique object cannot be distinguished from itself.
  3. Uniqueness may apply to collections of objects (i.e., a class of instances can be unique).

Object identifiers

- **Object identifier:**
  - **Individual objects within the resource are provided with a registry number and a suffix sequence, appended locally.**
  - **Life Science Identifiers** serve as a typical example of a registered identifier. Every LSID is composed of the following five parts: Network Identifier, root DNS name of the issuing authority, name chosen by the issuing authority, a unique object identifier assigned locally, and an optional revision identifier for versioning information.
    - Example 1: **urn:lsid:pdb.org:1AFT:1**. This identifies the first version of the 1AFT protein in the Protein Data Bank
    - Example 2: **urn:lsid:ncbi.nlm.nig.gov:GenBank:T48601:2**. This refers to the second version of an entry in GenBank.

Object identifiers

- In some cases, the registry **does not provide the full identifier** for data objects. The registry may provide a **general identifier sequence** that will apply to every data object in the resource.

- An **object identifier (OID)** is a hierarchy of identifier prefixes. Successive numbers in the prefix identify the **descending order of the hierarchy.**

Object identifiers

- Example: Hospitals use an OID system for identifying images—part of the DICOM (Digital Imaging and Communications in Medicine) image standard.
  - There is a **prefix** consisting of a permanent, registered code for the institution and the department and a **suffix** consisting of a number generated for an image, as it is created.
- A hospital may assign consecutive numbers to its images, appending these numbers to an OID **that is unique for the institution and the department within the institution**.
  - For example, the first image created with a computed tomography (CT) scanner might be assigned an identifier consisting of the OID (the assigned code for institution and department) followed by a separator such as a hyphen, followed by “1”.

Object identifiers

- In a worst-case scenario, different instruments may assign consecutive numbers to images, independently of one another.
  - This means that **the CT scanner in room A may be creating the same identifier (OID + image number) as the CT scanner in room B for images on different patients.** This problem could be remedied by constraining each CT scanner **to avoid using numbers assigned by any other CT scanner.** This remedy can be defeated if there is a glitch anywhere in the system that accounts for image assignments (e.g., if the counters are reset, broken, replaced, or simply ignored).
- Other problem:
  - the image service is assigned to another department in the institution, when departments merge, or when institutions merge. The old records in both of the merging institutions **will be assigned the same prefix and will contain replicate.**

Really bad identifier methods

- Names are poor identifiers.
  - Aside from the obvious fact that they are **not unique** (e.g., surnames such as Smith, Zhang, Garcia, Lo, and given names such as John and Susan), **a single name can have many different representations.**
  - The sources for these variations are many. Here is a partial listing:
    - Modifiers to the surname, accents that may or may not be transcribed onto records, special typographic characters, multiple "middle names" for an individual that may not be transcribed onto records, latinized and other versions of a single name, hyphenated names that are confused with first and middle names and others

Embedding Information in an Identifier: Not Recommended

- Most identifiers are not purely random numbers—they usually contain some embedded information that can be **interpreted by anyone familiar with the identification system**.
  - For example, they may embed the **first three letters of the individual's family name in the identifier**. Likewise, the last two digits of the birth year are commonly embedded in many types of identifiers. Such information is usually included as a crude "honesty" check by people "in the know."
  - For instance, the nine digits of a social security number are divided into an area code (first three digits), a group number (the next two digits), followed by a serial number (last four digits). People with expertise in the social security numbering system can pry considerable information from a social security number and can determine whether certain numbers are bogus based on the presence of excluded subsequences.

Embedding Information in an Identifier: Not Recommended

- Unimpressed? Consider this scenario:
  - You know that a prominent member of the President's staff had visited a Washington, DC, hospital on February 15, 2005, for the purpose of having a liver biopsy. You would like to know the results of that biopsy. You go to a Web site that lists the **de-identified** pathology records for the hospital for the years 2000 to 2010. Though **no personal identifiers** are included in these public records, the individual records are sorted by **accession** numbers. Using the aforementioned strategy, you collect all of the surgical biopsies performed **on or about February 15, 2005**. Of these biopsies, only **three are liver biopsies**. Of these three biopsies, only one was performed on a person whose **gender and age** matched the President's staff member. The report provides the diagnosis. You managed to discover some very private information without access to any personal identifiers.
- The alphanumeric character string composing the identifier **should not expose the patient's identity**.

One-way Hashes

- A one-way hash is an algorithm that **transforms a string into another string** in such a way that the **original string cannot be calculated by operations on the hash value** (hence the term "one-way" hash). Popular one-way hash algorithms are MD5 and Standard Hash Algorithm.
  - A one-way hash value can be calculated for any character string, including a person's name, a document, or even another one-way hash. For a given input string, the resultant one-way hash will always be the same.
  - MD5 example:
    - Jules Berman ➔ Ri0oaVTIAilwnS8tnvKhfA
    - "Whatever" ➔ n2YtKKG6E4MyEZvUKyGWrw
    - Whatever ➔ OkXaDVQFYjwkQtMOC8dpOQ
    - jules berman ➔ SlnuYpmyn8VXLsxBWwO57Q
    - Jules J. Berman ➔ i74wZ/CsIbxt3goH2aCStA
    - Jules J Berman ➔ yZQfJmAf4dIYO6Bd0qGZ7g
    - Jules Berman ➔ Ri0oaVTIAilwnS8tnvKhfA

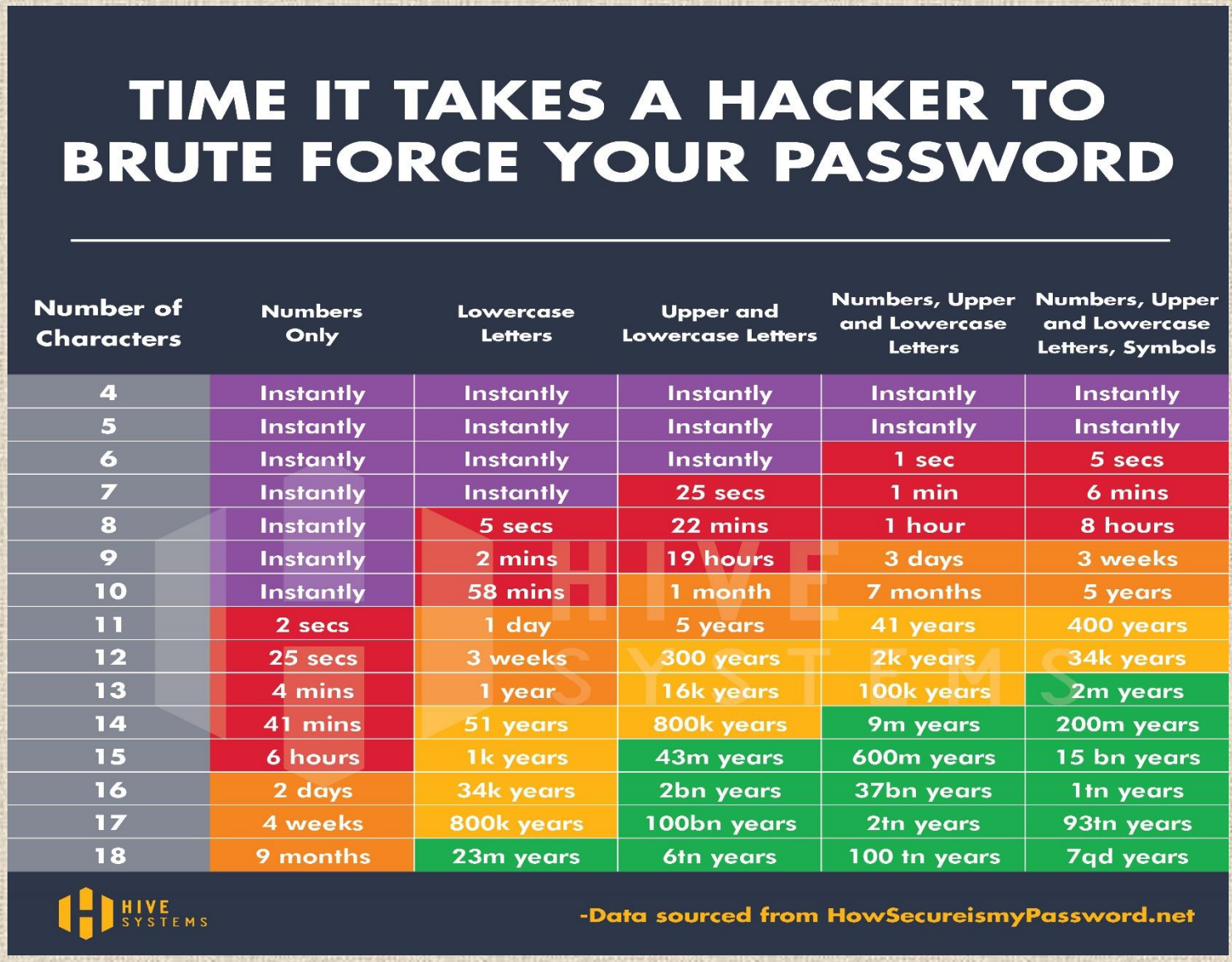
One-way Hashes

- One-way hash values can substitute for identifiers in individual data records. This permits Big Data resources to accumulate data over time to a specific record, even when the record is **de-identified**.
  - A record identifier serves as the input value for a one way hash.
  - The primary identifier for the record is now a **one-way hash sequence**.
  - The data manager of the resource, looking at such a record, **cannot determine** the individual associated with the record because the original identifier has been replaced with an unfamiliar sequence.

One-way Hashes

- Implementation of one-way hashes carries certain practical problems:
  - If anyone happens to have a **complete listing** of all of the original identifiers, then it would be a simple matter to perform one-way hashes on every listed identifier. This would produce a **look-up table** that can match de-identified records back to the original identifier, a strategy known as a **dictionary attack**.
  - For de-identification to work, the original identifier sequences **must be kept secret**.

Brute Force



TIME IT TAKES A HACKER TO BRUTE FORCE YOUR PASSWORD

Number of Characters	Numbers Only	Lowercase Letters	Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters, Symbols
4	Instantly	Instantly	Instantly	Instantly	Instantly
5	Instantly	Instantly	Instantly	Instantly	Instantly
6	Instantly	Instantly	Instantly	1 sec	5 secs
7	Instantly	Instantly	25 secs	1 min	6 mins
8	Instantly	5 secs	22 mins	1 hour	8 hours
9	Instantly	2 mins	19 hours	3 days	3 weeks
10	Instantly	58 mins	1 month	7 months	5 years
11	2 secs	1 day	5 years	41 years	400 years
12	25 secs	3 weeks	300 years	2k years	34k years
13	4 mins	1 year	16k years	100k years	2m years
14	41 mins	51 years	800k years	9m years	200m years
15	6 hours	1k years	43m years	600m years	15 bn years
16	2 days	34k years	2bn years	37bn years	1tn years
17	4 weeks	800k years	100bn years	2tn years	93tn years
18	9 months	23m years	6tn years	100 tn years	7qd years

HIVE SYSTEMS

-Data sourced from HowSecureismyPassword.net

De-identification

- De-identification is the process of **stripping information from a data record that might link the record to the public name of the record's subject.**
  - In the case of a patient record, this would involve stripping any information from the record that would enable someone to connect the record to the name of the patient.
    - For example: **patient's name,** patient's address (which could be linked to the name), the patient's date of birth (which narrows down the set of individuals to whom the data record might pertain), and the patient's social security number.
  - It is important to clarify that deidentification is **not achieved by removing an identifier from a data object**.
    - identifiers can be **substituted** with a one-way hash value, thus preserving the uniqueness of the record.

De-identification

- De-identification involves removing information contained in the data object that reveals something about the **publicly known name** of the data object.
  - This kind of information is often referred to as **identifying information**, or "**name-linking information**."
  - It may seem counterintuitive, but there **is very little difference between an identifier and a de-identifier**; under certain conditions the two concepts are equivalent.

De-identification

- Here is how a dual identification/de-identification system might work:
  - Collect data on unique object: “Joe Ferguson’s bank account contains $100.”
  - Assign a unique identifier: “Joe Ferguson’s bank account is 754003894713.”
  - Substitute name of object with its assigned unique identifier: “754003894713 contains $100.”
  - Consistently use the identifier with data.
  - Do not let anyone know that Joe Ferguson owns account “754003894713.”

De-identification

- The dual use of an identifier/de-identifier is a tried-and-true technique. Swiss bank accounts are essentially unique numbers (**identifiers**) assigned to a person. You access the bank account by producing the identifier number. The identifier number does not provide information about the identity of the bank account holder (**de-identifier**).

- The purpose of an identifier is to tell you that whenever the **identifier** is encountered, it **refers to the same unique object**, and whenever two different identifiers are encountered, they refer to different objects. The identifier, **by itself, contains no information that links the data object to its public name**.

De-identification

- At this point, you might be asking yourself the following question: "**What gives the data manager the right to distribute parts of a confidential record, even if it happens to be de-identified**?"
  - The act of de-identification renders the data harmless by transforming information about a person or data object into information about **nothing in particular**. Because the use of de-identified data poses no harm to human subjects, U.S. regulations allow the **unrestricted use of such data for research purposes**. Other countries have similar provisions.

Re-identification

- For scientists, deidentification serves two purposes:

1. **To protect the confidentiality and the privacy of the individual** (when the data concerns a particular human subject)

2. **To remove information that might bias the experiment** (e.g., to blind the experimentalist to patient identities)

- Because confidentiality and privacy concerns always apply to human subject data and because issues of experimental bias always apply when analyzing data, it would seem imperative that de-identification should be an irreversible process (i.e., the names of the subjects and samples should be held a secret, forever).

Re-identification

- Scientific integrity does not always accommodate **irreversible de-identification.**
  - On occasion, experimental samples are mixed up; samples thought to come from a certain individual, tissue, record, or account may come from another source.
  - Sometimes major findings in science need to be **retracted** when a sample mix-up has been shown to occur. When samples are submitted, without mix-up, the data is sometimes collected improperly.
    - For example, reversing electrodes on an electrocardiogram may yield spurious and misleading results. Sometimes data is purposefully fabricated and otherwise corrupted to suit the personal agendas of dishonest scientists.
    - When data errors occur, regardless of reason, it is important to retract the publications.

Re-identification

- To preserve scientific integrity, it is sometimes necessary to discover the identity of de-identified records. In some cases, de-identification stops the data analyst from helping individuals whose confidentiality is being protected.
  - Imagine you are conducting an analysis on a collection of deidentified data and **you find patients with a genetic marker for a disease that is curable**, if treated at an early stage.
  - De-identified records can, **under strictly controlled circumstances**, be re-identified. Re-identification is typically achieved by entrusting a third party with a confidential list that maps individuals to their deidentified records.

Concluding remarks

- Identification issues are often ignored by Big Data managers who are accustomed to working on small data projects.
  - It is worthwhile to repeat the most important ideas described in this chapter, many of which are counterintuitive and strange to those whose lives are spent outside the confusing realm of Big Data.

1. All big data resources can be imagined as an **identifier system for data objects and data-related events** (i.e., timed transactions). The data in a big data resource can be imagined as character sequences that are attached to identifiers.

2. Without an adequate identification system, a big data resource has **no value**. The data within the resource **cannot be trusted**.

Concluding remarks

3. An identifier is a **unique alphanumeric sequence** assigned to a data object.

4. A data object is a collection of data that contains **self-describing information**, and one or more data values. Data objects should be associated with a unique identifier.

5. De-identification is the process of **stripping information from a data record** that might link the record to the public name of the record's subject.

6. De-identification should not be confused with the act of stripping a record of an identifier. A de-identified record must have an associated identifier, just as an identified data record must have an identifier.

Concluding remarks

7. Where there is no identification, there can be no de-identification and no re-identification.

8. Re-identification is the **assignment of the public name associated with a data record to the de-identified record**.
   - Re-identification is sometimes necessary to verify the contents of a record or to provide information that is necessary for the well-being of the subject of a deidentified data record.
   - Reidentification always requires approval and oversight.

9. When a de-identified data set contains **no unique records** (i.e., every record has one or more additional records from which it cannot be distinguished, aside from its assigned identifier sequence), then it becomes impossible to uncover a de-identified record's public name.

Concluding remarks

Aspect	Identification	De-identification	Re-identification
Purpose	Recognize unique individuals/objects	Anonymize individuals/ objects for privacy	Link anonymized data back to individuals/ objects
Data Features Used	Unique features (e.g., face, voice, biometrics)	Masked or removed identifiable details	Residual patterns or re-linking unique characteristics
Application	Personalization, security access	Privacy-preserving data sharing	Testing privacy robustness and risk assessment
Privacy Concern	High risk if misused	Lower risk if effective	High risk, may lead to privacy breach
Example Techniques	Face/voice recognition models	Data masking, pseudonymization, aggregation	Cross-referencing residual data for re-identification

WQD7007 Big Data Management

Big Data Concepts

Identification, de-identification, and re-identification

Ontologies and semantics

PART 2

Ontologies and Semantics

Agenda

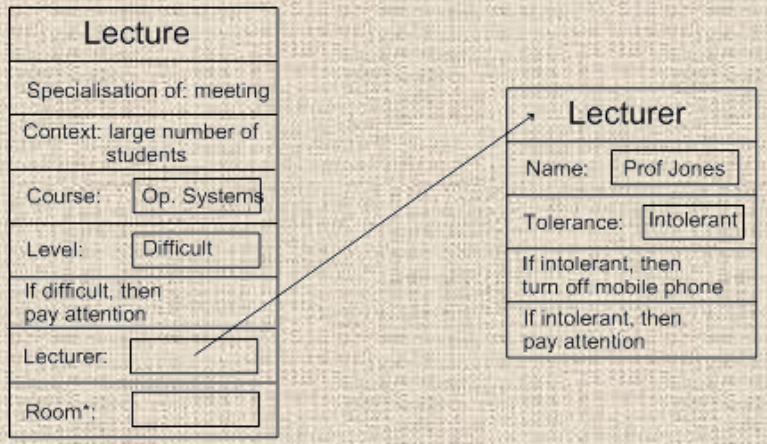
- Classifications, The Simplest Of Ontologies
- Ontologies, classes with multiple parents
- Choosing a Class Model
- Introduction to Resource Description Framework Schema
- Common Pitfalls in Ontology Development

Ontologies and Semantics

- Information has **limited value** unless it can take its place within our **general understanding** of the world.
  - When a financial analyst learns that the **price of a stock has suddenly dropped**, he cannot help but wonder if the drop of a single stock reflects **conditions in other stocks** in the **same** industry.
  - If so, the analyst may check to ensure that **other industries** are following a downward trend.
  - He may wonder whether the downward trend represents **a shift in the national or global economies**.
- In every case, the analyst is asking a **variation** on a single question: **"How does this thing relate to that thing?"**

Ontologies and Semantics

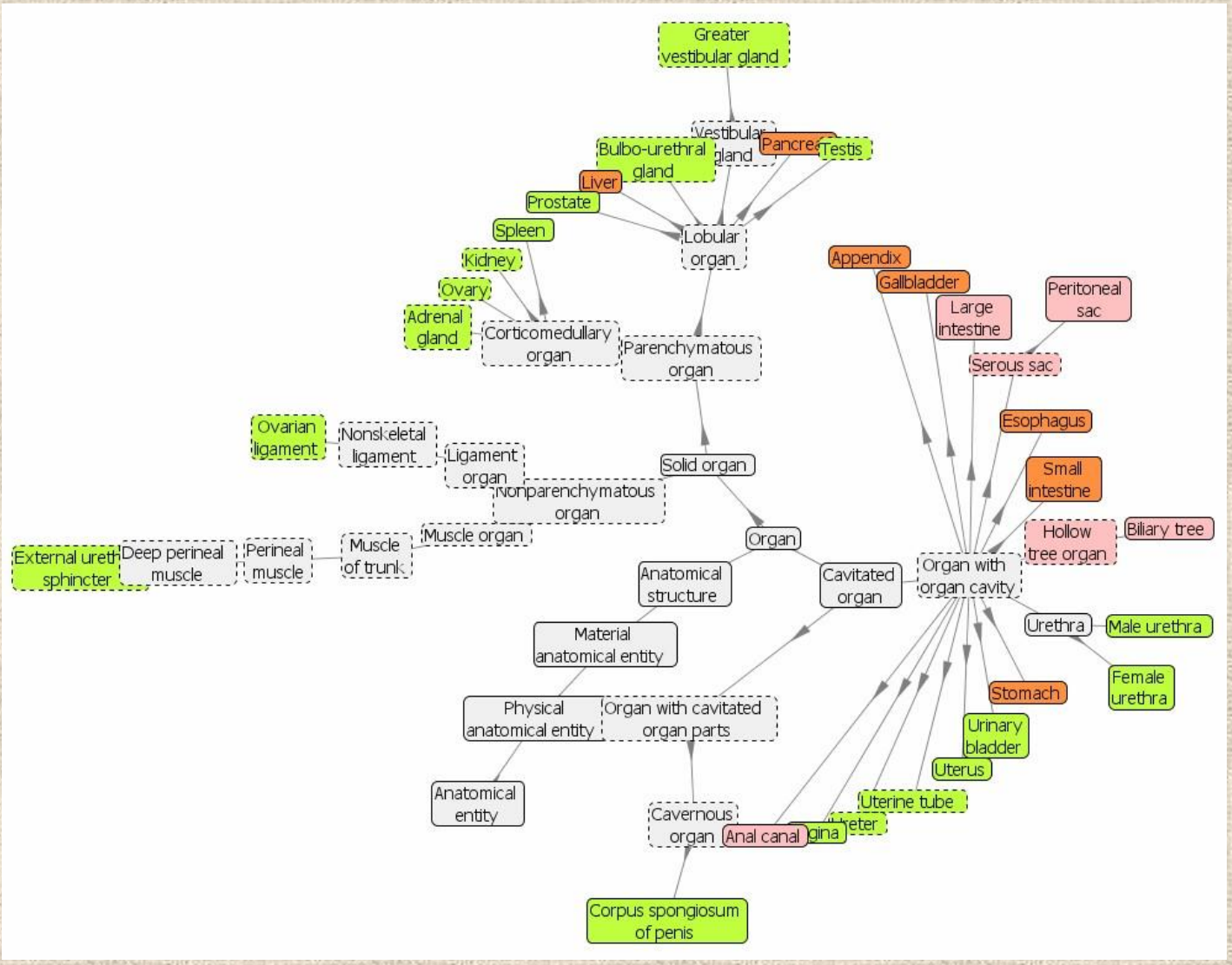
- Big data resources are **complex**.
  - When data is simply stored in a database, without any general principles of organization, it is impossible to discover the **relationships** among the data objects.
  - To be useful, the information in a big data resource must be **divided into classes of data**. Each data object within a class **shares a set of properties** chosen to enhance our ability to **relate one piece of data with another**.



Ontologies

- **Ontologies** are formal systems that **assign data objects to classes** and **that relate classes to other classes.**
  - When the data within a big data resource is classified **within an ontology**, data analysts can determine whether **observations on a single object will apply to other objects in the same class**.
  - Similarly, data analysts can begin to ask whether observations that **hold true for a class of objects** will **relate to other classes of objects**.
- Ontologies help to determine how things relate to other things.

Ontology example



Classifications, The Simplest Of Ontologies

- The human brain is **constantly processing visual and other sensory information collected from the environment**.
  - When we walk down the street, we see images of concrete and asphalt and millions of blades of grass, birds, dogs, other persons, and so on.
  - Every step we take conveys a new world of sensory input.

- **How can we process it all?**

Classifications, The Simplest Of Ontologies

- The mathematician and philosopher Karl Pearson (1857–1936) has likened the human mind to a "**sorting machine**."
  - We take a stream of sensory information, **sort it into a set of objects**, and then assign the individual objects to **general classes**.
  - The green stuff on the ground is classified as "grass," and the grass is sub-classified under some larger grouping, such as "plants."
  - A flat stretch of asphalt and concrete may be classified as a "road," and the road might be sub-classified under "man-made constructions."

Classifications, The Simplest Of Ontologies

- If we lacked a culturally determined classification of objects for our world, we would be overwhelmed by sensory input, and we would have **no way to remember what we see and no way to draw general inferences about anything**.
  - Simply put, without our ability to classify, we would not be human.

Classifications, The Simplest Of Ontologies


- Every culture has some particular way to impose a uniform way of perceiving the environment.
  - In English-speaking cultures, the term "**hat**" denotes a universally recognized object. Hats may be composed of many **different types of materials** and they may vary greatly in **size, weight, and shape**.
  - Nonetheless, we can almost always identify a hat when we see one, and we can distinguish a hat from all other types of objects. A hat is classified as a hat because it has a **class relationship; all hats are items of clothing that fit over the head.**

- Likewise, all **biological** classifications are built **by relationships, not by similarities**.

Classifications, The Simplest Of Ontologies

- Aristotle was one of the first experts in classification. **His greatest insight came when he correctly identified a dolphin as a mammal.**
  - Through observation, he knew that a large group of animals was distinguished by a **gestational period** in which a developing embryo is nourished by a placenta, and the offspring are delivered into the world as formed but small versions of the adult animals (i.e., not as eggs or larvae), and the newborn animals feed from milk excreted from nipples, overlying specialized glandular organs (mammae).
  - Aristotle knew that **these features, characteristic of mammals, were absent in all other types of animals.**

Classifications, The Simplest Of Ontologies

- He also knew that **dolphins had all these features; fish did not**. He correctly reasoned that dolphins were a type of mammal, not a type of fish.

- Aristotle was ridiculed by his contemporaries for whom it was obvious that dolphins were a type of fish.

- Unlike Aristotle, they based their classification on **similarities**, not on **relationships**.

Classifications, The Simplest Of Ontologies

- A classification is a very simple form of ontology, in which **each class is limited to one parent class**.
- To build a classification, the ontologist must do the following:
  1. define classes (i.e., **find the properties that define a class** and extend to the subclasses of the class),
  2. **assign instances** to classes,
  3. position classes within the **hierarchy**, and
  4. **test and validate** all of the above.

Classifications, The Simplest Of Ontologies

The constructed classification becomes a **hierarchy of data objects** conforming to a set of principles:

1. The classes (groups with members) of the hierarchy have **a set of properties or rules that extend to every member of the class and to all of the subclasses of the class**, to the exclusion of unrelated classes.
   - A subclass is itself a type of class wherein the members have the defining class properties of the parent class plus some additional property(ies) specific to the subclass.
2. In a **hierarchical classification**, **each subclass may have no more than one parent class**. The root (top) class has no parent class.
   - The biological classification of living organisms is a hierarchical classification.

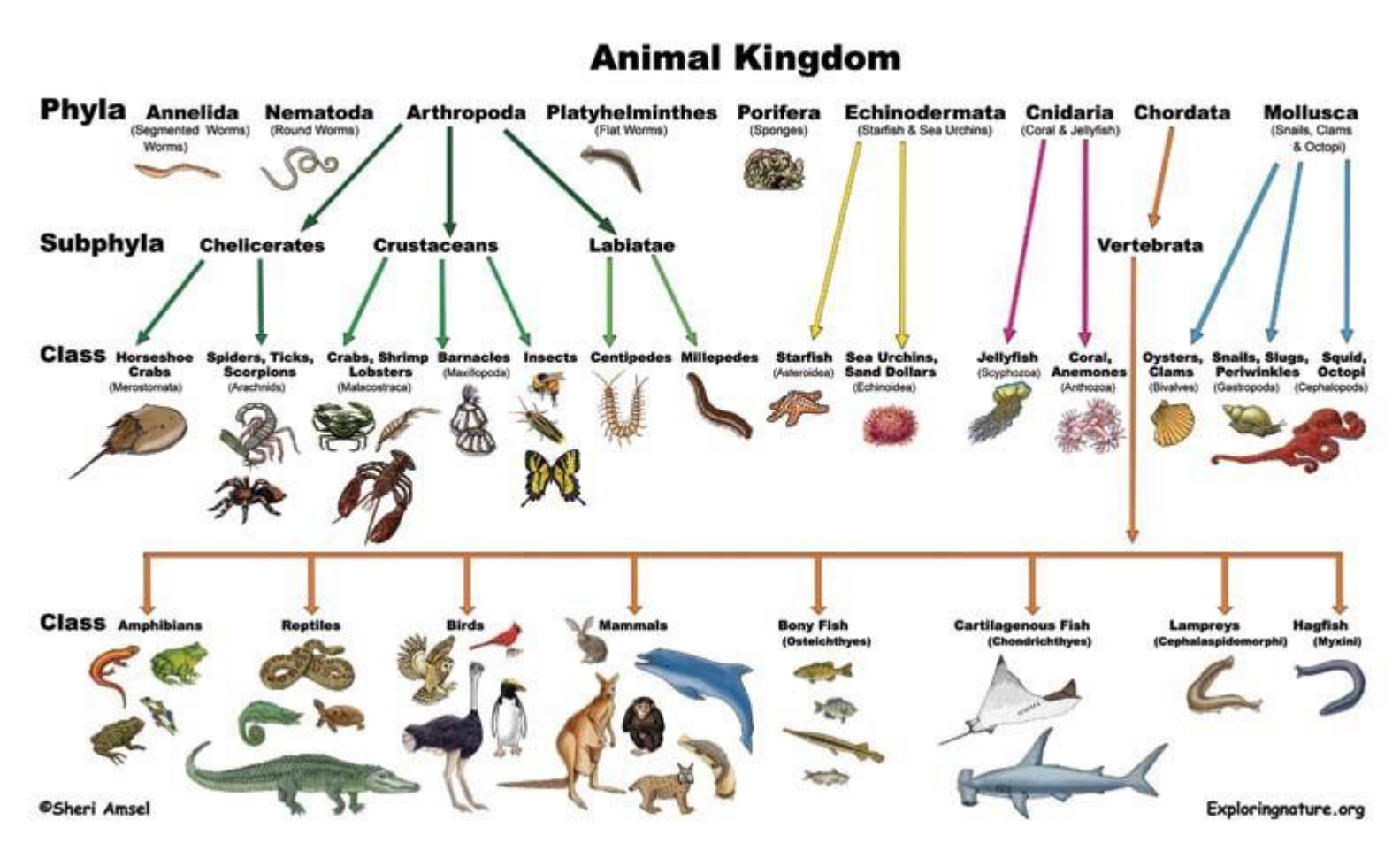
Classifications, The Simplest Of Ontologies

3. At the **bottom** of the hierarchy is the **class instance**.
   - For example, the book ”Principles of Big Data” is an instance of the class of **objects** known as “books.”
4. Every **instance** belongs to **exactly one class**.
5. Instances and classes **do not change their positions** in the classification.
   - For example, a horse never transforms into a sheep and a book never transforms into a harpsichord.
6. The **members** of classes **may be highly similar** to one another, but their similarities **result from their membership** in the same class (i.e., conforming to **class properties**), and not the other way around (i.e., **similarity alone cannot define class inclusion**).

Classifications, The Simplest Of Ontologies

- A **taxonomy** is a classification with the instances "filled in".
  - This means that for each class in a taxonomy, all the **known instances** (i.e., member objects) are **explicitly listed**. For the taxonomy of living organisms, the instances are named species.
  - Currently, there are several million named species of living organisms, and each of these several million species is listed under the name of some class included in the full classification.
  - Classifications drive down the complexity of their data domain because every instance in the domain is assigned to a single class and every class is related to the other classes through a **simple hierarchy**.

Classifications, The Simplest Of Ontologies



Classification system VS identification system?

- **An identification system puts a data object into its correct slot within the classification.**
  - For example, a **fingerprint-matching system** may look for a set of features that puts a fingerprint into a special subclass of all fingerprints, but the primary goal of fingerprint matching is to establish the identity of an instance (i.e., to show that two sets of fingerprints belong to the same person).
  - In the realm of medicine, when a doctor renders a **diagnosis on a patient's diseases**, she is not classifying the disease—she is finding the correct slot within the pre-existing classification of diseases that holds her patient's diagnosis.

Ontologies, classes with multiple parents

- Ontologies are constructions that permit an object to be **a direct subclass of more than one class**.
  - In an ontology, the class "**horse**" might be a subclass of Equus, a zoologic term, as well as a subclass of "racing animals," "farm animals," and "four-legged animals."
  - The class "**book**" might be a subclass of "works of literature," as well as a subclass of "wood-pulp materials" and "inked products."
- Ontologies are unrestrained classifications.
  - Ontologies are predicated on the belief that a **single object or class of objects might have multiple different fundamental identities** and that these different identities will often place one class of objects directly under more than one superclass.

Ontologies, classes with multiple parents

- Data analysts sometimes **prefer ontologies over classification**s because they permit the analyst **to find relationships** among classes of objects that would have been **impossible to find under a classification**.
  - For example, a data analyst might be interested in determining the relationships among groups of flying animals, such as butterflies, birds, bats, and so on.
    - In the classification of living organisms, these animals occupy classes that are not closely related to one another—no two of the different types of flying animals share a single parent class. **Because classifications follow relationships through a lineage**, they cannot connect instances of classes that fall outside the line of descent.

Ontologies, classes with multiple parents

- In an ontology, a data object can be an instance of **many different kinds of classes**; thus, **the class does not define the essence of the object** as it does in a classification.
  - the **assignment** of an object to a class and the **behavior** of the members of the objects of a class are determined by **rules**.
    - An object belongs to a class when it behaves like the other members of the class, according to a rule created by the ontologist.
    - Every class, subclass, and superclass is defined by rules, and rules can be programmed into software.

Ontologies, classes with multiple parents

- Classifications were created and implemented at a time when scientists **did not have powerful computers** that were capable of handling the complexities of ontologies.
  - For example, the classification of all living organisms on earth was created over a period of two millennia. Several million species have been assigned to the classification. It is currently estimated that we will need to add another 10 to 50 million species before we come close to completing the **taxonomy** of living organisms.
  - Prior generations of scientists could cope with a **simple classification**, wherein each class of organisms falls under a single superclass; they could not cope with a complex ontology of organisms.

Ontologies, classes with multiple parents

- The question that should confront every big data manager is

1. "Should I model my data as a **classification**, wherein every class has one direct parent class", or

2. "should I model the resource as an **ontology**, wherein classes may have multi-parental inheritance?"

Choosing a Class Model

- The simple and fundamental question "Can a class of objects have more than one parent class?" lies at the heart of several related fields:
  - **database management,**
  - **computational informatics,**
  - **object-oriented programming,**
  - **semantics, and**
  - **artificial intelligence**

Choosing a Class Model

- Computerscientists are choosing sides, often without acknowledging the problem or fully understanding the stakes.
  - For example, when a programmer builds object libraries in the Python or the Perl programming languages, he is choosing to program in a permissive environment that supports multiclass object inheritance
  - In Python and Perl, any object can have as many parent classes as the programmer prefers.

Choosing a Class Model

- When a programmer chooses to program in the Ruby programming language, he shuts the door on **multiclass inheritance.**
  - A Ruby object can have only one direct parent class. Most programmers are totally unaware of the liberties and restrictions imposed by their choice of programming language until they start to construct their own object libraries or until they begin to use class libraries prepared by another programmer.

Choosing a Class Model

- **Without stating a preference** for single-class inheritance (classifications) or multiclass inheritance (ontologies), one should always strive to design a model that is **as simple as possible**.
  - The wise ontologist will settle for a **simplified approximation of the truth**. Regardless of your personal preference, you should learn to recognize when an ontology has become too complex. Here are the danger signs of an overly complex ontology:

Choosing a Class Model

1. **Nobody**, even the designers, **fully understands** the ontology model.
2. You realize that the ontology **makes no sense**.
3. For a given problem, **no two data analysts seem able to formulate the query the same way, and no two query results are ever equivalent**.
4. The **time** spent on **ontology design** and improvement **exceeds** the time spent on **collecting the data** that populates the ontology.
5. The ontology cannot be fitted into a **higher** level ontology or a **lower** level ontology.
6. The ontology **cannot be debugged when errors are detected.**
7. Errors occur **without anyone knowing** that the error has occurred.

Choosing a Class Model

Simple classifications are not flawless. Here are a few danger signs of an overly simple classification:

1. The classification is **too granular** to be of much value in associating observations with particular instances within a class or with particular classes within the classification.
2. The classification **excludes important relationships among data objects**.
3. The classes in the classification lack **inferential competence.**
4. The classification **contains a "miscellaneous" class.**
5. The classification may be **unstable**.

Choosing a Class Model

- It seems obvious that in the case of big data, a computational approach to data classification is imperative, but **a computational approach that consistently leads to failure is not beneficial**.

- Most of the ontologies that have been created for data collected in many of the fields of science have been ignored or abandoned by their intended beneficiaries.
  - **They are simply too difficult to understand and too difficult to implement.**

Common Pitfalls in Ontology Development

- Do ontologies serve a **necessary role** in the design and development of big data resources?
  - **Yes**. Because every big data resource is composed of many different types of information, it becomes important to assign types of data into groups that have similar properties:
    - images, music, movies, documents, and so forth.

Common Pitfalls in Ontology Development

- The data manager needs to **distinguish one type of data object from another and must have a way of knowing the set of properties** that apply to the members of each class.
  - When a query comes in asking for a list of songs written by a certain composer or performed by a particular musician, the data manager will need to have a software implementation wherein **the features of the query are matched to the data objects for which those features apply**.

Common Pitfalls in Ontology Development

Common Pitfalls:

1. Don't build **transitive** classes
   - Puppy ➔ dog?

2. Don't build **miscellaneous** classes.

3. Don't **invent** classes and properties if they have already been invented.

4. Do not **confuse properties** with your **classes**.
   - Is a leg a subclass of the human body?