DQDA

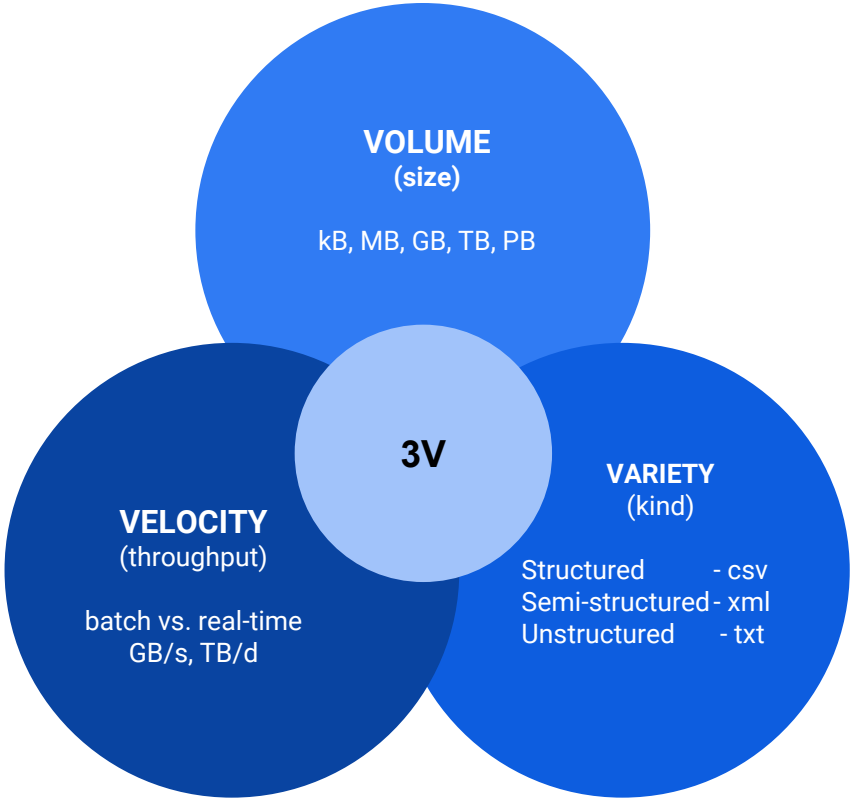
Fundamentals of data representation

Agenda

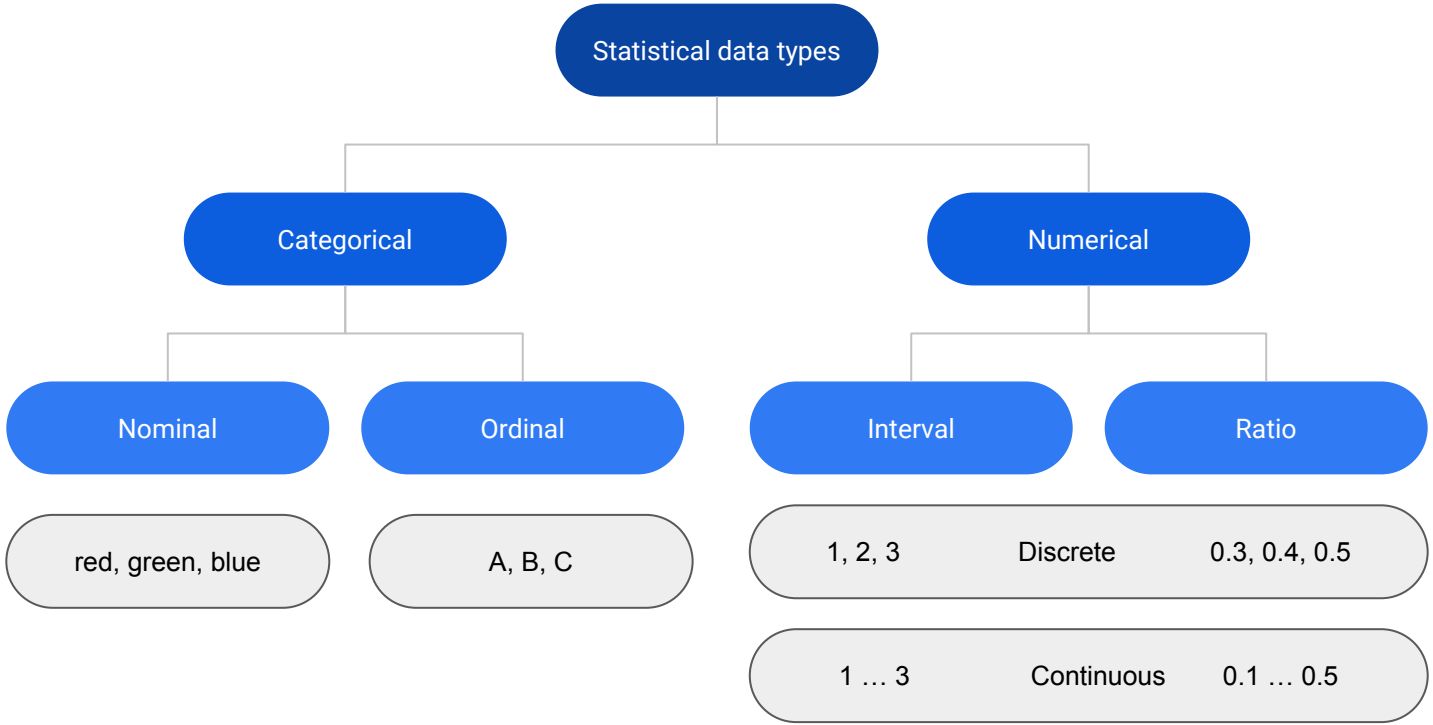
- Data classification
- Data representation
- Relational Database Management Systems (RDBMS)
- Python programming language
- Object relational mismatch

An introduction to data and data representation

3V of data


VOLUME
(size)
kB, MB, GB, TB, PB
3V
VELOCITY
(throughput)
batch vs. real-time
GB/s, TB/d
VARIETY
(kind)
Structured - csv
Semi-structured - xml
Unstructured - txt

Data types in statistics



Binary digit = Bit

1 Bit can either hold the value 0 or 1
Binary data is based on the number 2

Definition:
A set is a collection of distinct elements.
Cardinality (or size) is the number of elements.

Q1) Considering a memory capacity of 3 bit, what is the set of non-negative whole numbers?

Q2) Considering Q1, what is the cardinality (or size) of the set?

A1){0...7}

```
000 - 0 - 0
001 - 1 - 2**0
010 - 2 - 2**1
011 - 3 - 2**1 + 2**0
100 - 4 - 2**2
101 - 5 - 2**2 + 2**0
110 - 6 - 2**2 + 2**1
111 - 7 - 2**2 + 2**1 + 2**1
```

A2) 2**3 = 8

The cardinality (or set-size) is 8

Signed integers

Q3) Considering a memory capacity of 3 bit, with one bit denoting +/-, what is the set of whole numbers?

Q4) Considering Q3, what is the cardinality of absolute values?

Q5) Considering a memory capacity of 8 bit when denoting signed integers, what is the cardinality of the set?

A3){-3...3}

==========

111 - (-3)

110 - (-2)

101 - (-1)

100 - (-0)

000 - 0

001 - 1

010 - 2

011 - 3

A4) 2**(2-1) = 4

The cardinality of the set of absolute values is 4.

ASCII

American Standard Code for Information Interchange encodes 128 characters (of the English alphabet) into 7 bit integers.

Binary	Oct	Dec	Hex	Abbreviation			[b]	[c]	[d]	Name ('67)
				'63	'65	'67				
000 0000	000	0	00	NULL	NUL		$^{N}_{U}$	^@	\0	Null
000 0001	001	1	01	SOM	SOH		$^{S}_{O}$	^A		Start of Heading
000 0010	002	2	02	EOA	STX		$^{S}_{T}$	^B		Start of Text
000 0011	003	3	03	EOM	ETX		$^{E}_{T}$	^C		End of Text

UTF-8

Designed for backward compatibility with ASCII. The first 128 characters of Unicode, which correspond one-to-one with ASCII, are encoded using a single octet with the same binary value as ASCII, so that valid ASCII text is valid UTF-8-encoded Unicode as well.

Relational database management systems (RDBMS)

RDBMS properties

Invented by **Edgar F. Codd in 1970.**
Data is organized in relational sets.

Table = relation
Column = attribute = field
    define data type
Row = record = tuple
    each row is unique
Primary key = unique row identifier
View = predefined and *unmaterialized* table

https://searchdatamanagement.techtarget.com/definition/relational-database

Entity integrity:
    unique non-null primary key
Referential integrity:
    foreign keys are primary keys
    from another table

TA = Transaction
ACID properties
    Atomic TA are all or nothing
    Consistent invalid TA fail
    Isolated concurrent TA are possible
    Durable TA persist after sys. failure

PostgreSQL table declaration

```
bit(size)
varbit(size)
smallint
int
integer
bigint
smallserial
serial
bigserial
numeric(m,d)
double precision
real
money
bool
boolean
```

```
CREATE SCHEMA hollywood;

CREATE TABLE distributors (
    did     integer,
    name    varchar(40),
    PRIMARY KEY(did)
);
```

https://www.postgresql.org/docs/10/static/sql-createschema.html

Python

Python data types

Number
Integer, float, double, bool

String
Denoted with single or double quotes “‘“

List
Vertical collection preserving order

Tuple
Horizontal collection preserving order

Dictionary
Unordered key value object store

Set
Superimposes order on distinct elements

```
n=1; n=int(1)

s='text'; s=str("text"); s=str(a)

l=[0,1,2]; l=['0',1,2]

t=(0,1,2); t=('0',1)

d={'key':'value'}; d={'k':0}

S={7,3,1}
```

The object-relational impedance mismatch

~

The Vietnam of computer science

https://blog.codinghorror.com/object-relational-mapping-is-the-vietnam-of-computer-science/

Objects

Linked or graph-like

Allows for *redundant* representations

Denormalized

Association	enforced dependency
Aggregation	object list
Composition	multiple enforced dependencies

Schemaless

Schema generation on the fly as needed

Dynamically or strictly typed

Flexibility in data type declaration

Inherited and expandable

Children are supersets of their parents

Relations

Tabular

Based on set theory and flat tables

Normalized

Minimal use of disk space due to
B-Tree structure and pagination
Hash table indices

Schema declarative

Data storage enforced via schemata

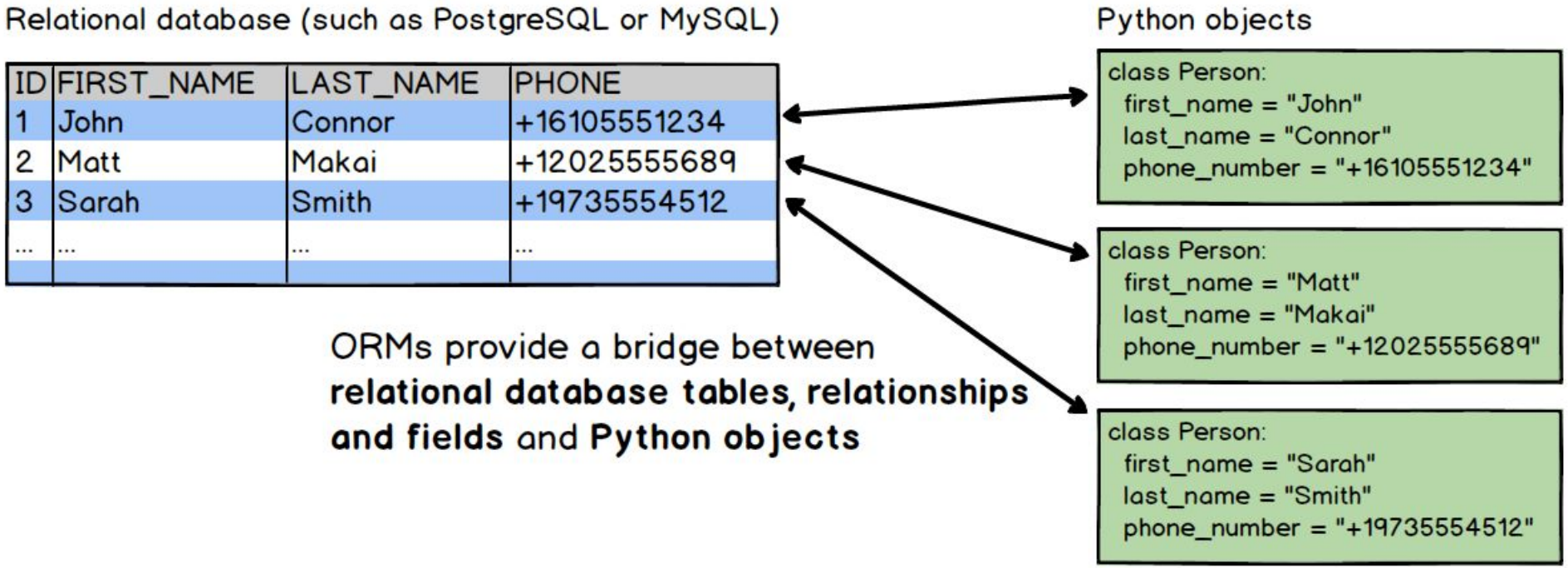
Strictly typed

Attributes store declared data type only

Tuple based

Empty records are NULL for each tuple

Relational sets vs Inherited classes



Python inheritance

```
class Pet(object):

    def __init__(self, name, species):
        self.name = name
        self.species = species

    def getName(self):
        return self.name

    def getSpecies(self):
        return self.species

    def __str__(self):
        return (self.name, self.species)
```

The first word, **class**, indicates that we are creating a class.

The second word, **Pet**, indicates the name of the class.

The word in parentheses, **object**, is the class that Pet is inheriting from.

http://www.jesshamrick.com/2011/05/18/an-introduction-to-classes-and-inheritance-in-python/

B-Trees

- **Balanced** – this is a self balancing data structure, which means that performance can be guaranteed when B-Trees are utilized.
- **Broad** – as opposed to binary search trees, which grow vertically, B-Trees expand horizontally.
- **Bayer** – the creator of B-Trees was named Bayer Rudolf. In all actuality this is probably the reason why B-Trees got their name.

https://www.crondose.com/2016/10/introduction-to-the-b-tree-data-structure/