The Quantum Mechanics of Data Pipelines

Pere Urbon-Bayes
Data Wrangler
pere.urbon @ { gmail.com, acm.org }
http://www.purbon.com

Who am I

Pere Urbon-Bayes (Berliner since 2011)

Software Architect and Data Engineer

Passionate about Systems, Data and Teams

Free and Open Source Advocate and Contributor

Working on a Data Engineering book for everybody

Software Architect and Data Engineer for Hire

Springer Nature in Berlin

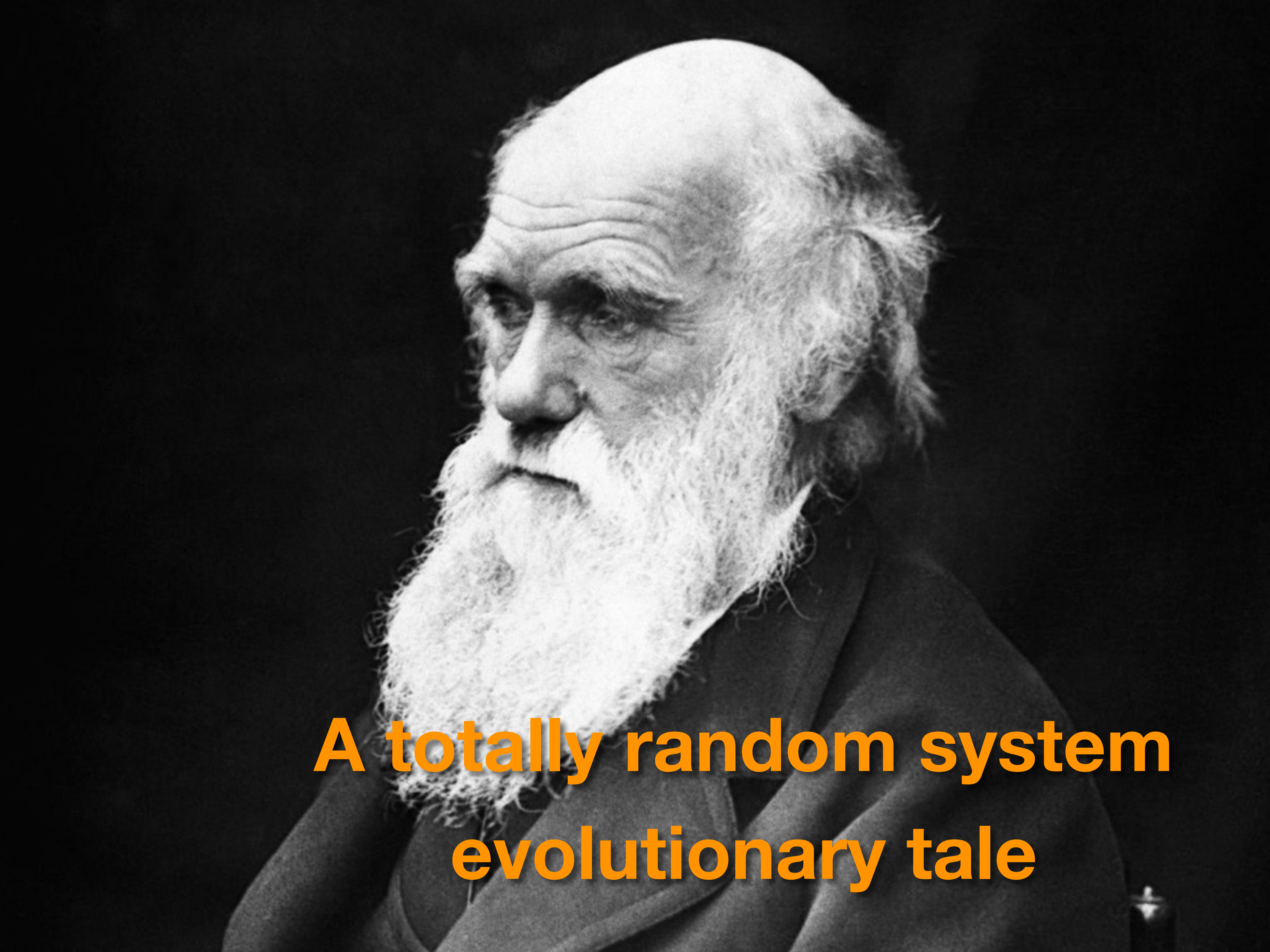
Our Brands

Springer	nature research	BMC	palgrave macmillan	SCIENTIFIC AMERICAN
Apress®	Adis	J.B. METZLER	macmillan education	Springer Healthcare

Topics of Today

- Making data available across the company.
- From the warehouse to the era of real time.
  - Approaches to make data available.
  - Benefits and Challenges.
- The hardest problem, the human part.

Information systems, sharing data between applications since last century….



A totally random system evolutionary tale

BONHEUR
LEONARDO DiCAPRIO
KEN WATANABE
JOSEPH GORDON-LEVITT
MARION COTILLARD
ELLEN PAGE
TOM HARDY
CILLIAN MURPHY
TOM BERENGER
AND MICHAEL CAINE
FROM THE DIRECTOR OF THE DARK KNIGHT
INCEPTION
EXPERIENCE IT JULY 16 IN THEATERS AND IMAX
IMAX is a registered trademark of IMAX Corporation.
PG-13 PARENTS STRONGLY CAUTIONED
Some Material May Be Inappropriate for Children Under 13
SEQUENCES OF VIOLENCE AND ACTION THROUGHOUT
inceptionmovie.com
LEGENDARY PICTURES
SYNCOPY
WARNER BROS. PICTURES
©2010 Warner Bros. Ent. All Rights Reserved

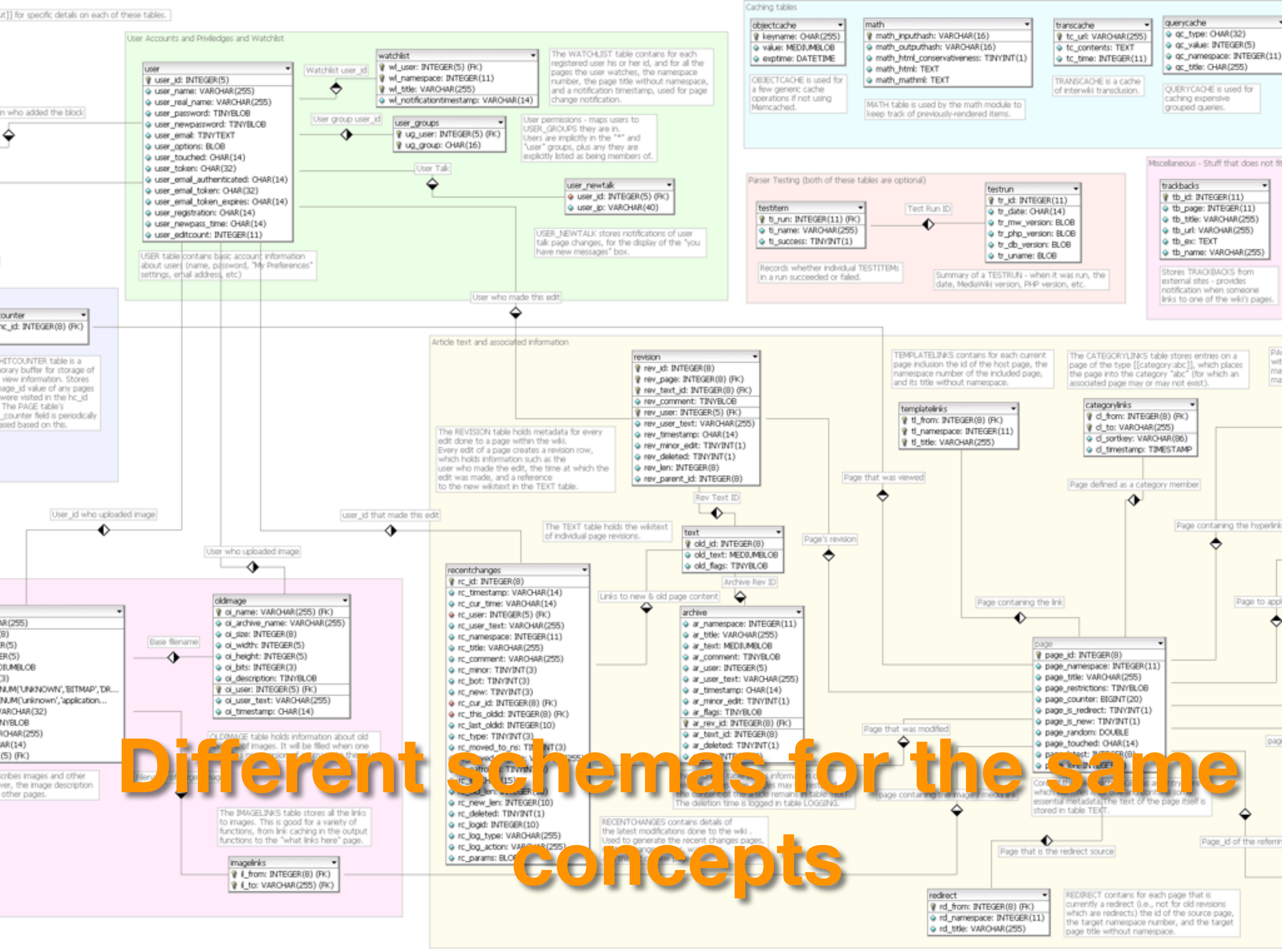
The Analyst Emergence

The easiest system is the isolated one

Acquiring or being acquired

The ever growing chaos of a technology variety

Different schemas for the same concepts



ut]] for specific details on each of these tables.
User Accounts and Priviledges and Watchlist
Caching tables
Parser Testing (both of these tables are optional)
Miscellaneous - Stuff that does not fit
Article text and associated information
Watchlist user_id
User group user_id
User Talk
User who made this edit
n who added the block
Test Run ID
Rev Text ID
Archive Rev ID
Links to new & old page content
Page's revision
Page that was viewed
Page defined as a category member
Page containing the hyperlink
Page containing the link
Page to appl
Page that was modified
page containing the image/media
Page that is the redirect source
Page_id of the referrin
User_id who uploaded image
user_id that made this edit
User who uploaded image
Base filename

user
- user_id: INTEGER(5)
- user_name: VARCHAR(255)
- user_real_name: VARCHAR(255)
- user_password: TINYBLOB
- user_newpassword: TINYBLOB
- user_email: TINYTEXT
- user_options: BLOB
- user_touched: CHAR(14)
- user_token: CHAR(32)
- user_email_authenticated: CHAR(14)
- user_email_token: CHAR(32)
- user_email_token_expires: CHAR(14)
- user_registration: CHAR(14)
- user_newpass_time: CHAR(14)
- user_editcount: INTEGER(11)

USER table contains basic account information about users (name, password, "My Preferences" settings, email address, etc)

watchlist
- wl_user: INTEGER(5) (FK)
- wl_namespace: INTEGER(11)
- wl_title: VARCHAR(255)
- wl_notificationtimestamp: VARCHAR(14)

The WATCHLIST table contains for each registered user his or her id, and for all the pages the user watches, the namespace number, the page title without namespace, and a notification timestamp, used for page change notification.

user_groups
- ug_user: INTEGER(5) (FK)
- ug_group: CHAR(16)

User permissions - maps users to USER_GROUPS they are in. Users are implicitly in the "*" and "user" groups, plus any they are explicitly listed as being members of.

user_newtalk
- user_id: INTEGER(5) (FK)
- user_ip: VARCHAR(40)

USER_NEWTALK stores notifications of user talk page changes, for the display of the "you have new messages" box.

objectcache
- keyname: CHAR(255)
- value: MEDIUMBLOB
- exptime: DATETIME

OBJECTCACHE is used for a few generic cache operations if not using Memcached.

math
- math_inputhash: VARCHAR(16)
- math_outputhash: VARCHAR(16)
- math_html_conservativeness: TINYINT(1)
- math_html: TEXT
- math_mathml: TEXT

MATH table is used by the math module to keep track of previously-rendered items.

transcache
- tc_url: VARCHAR(255)
- tc_contents: TEXT
- tc_time: INTEGER(11)

TRANSCACHE is a cache of interwiki transclusion.

querycache
- qc_type: CHAR(32)
- qc_value: INTEGER(5)
- qc_namespace: INTEGER(11)
- qc_title: CHAR(255)

QUERYCACHE is used for caching expensive grouped queries.

testitem
- ti_run: INTEGER(11) (FK)
- ti_name: VARCHAR(255)
- ti_success: TINYINT(1)

Records whether individual TESTITEMs in a run succeeded or failed.

testrun
- tr_id: INTEGER(11)
- tr_date: CHAR(14)
- tr_mw_version: BLOB
- tr_php_version: BLOB
- tr_db_version: BLOB
- tr_uname: BLOB

Summary of a TESTRUN - when it was run, the date, MediaWiki version, PHP version, etc.

trackbacks
- tb_id: INTEGER(11)
- tb_page: INTEGER(11)
- tb_title: VARCHAR(255)
- tb_url: VARCHAR(255)
- tb_ex: TEXT
- tb_name: VARCHAR(255)

Stores TRACKBACKS from external sites - provides notification when someone links to one of the wiki's pages.

counter
- hc_id: INTEGER(8) (FK)

HITCOUNTER table is a orary buffer for storage of view information. Stores age_id value of any pages were visited in the hc_id The PAGE table's _counter field is periodically ased based on this.

revision
- rev_id: INTEGER(8)
- rev_page: INTEGER(8) (FK)
- rev_text_id: INTEGER(8) (FK)
- rev_comment: TINYBLOB
- rev_user: INTEGER(5) (FK)
- rev_user_text: VARCHAR(255)
- rev_timestamp: CHAR(14)
- rev_minor_edit: TINYINT(1)
- rev_deleted: TINYINT(1)
- rev_len: INTEGER(8)
- rev_parent_id: INTEGER(8)

The REVISION table holds metadata for every edit done to a page within the wiki. Every edit of a page creates a revision row, which holds information such as the user who made the edit, the time at which the edit was made, and a reference to the new wikitext in the TEXT table.

text
- old_id: INTEGER(8)
- old_text: MEDIUMBLOB
- old_flags: TINYBLOB

The TEXT table holds the wikitext of individual page revisions.

templatelinks
- tl_from: INTEGER(8) (FK)
- tl_namespace: INTEGER(11)
- tl_title: VARCHAR(255)

TEMPLATELINKS contains for each current page inclusion the id of the host page, the namespace number of the included page, and its title without namespace.

categorylinks
- cl_from: INTEGER(8) (FK)
- cl_to: VARCHAR(255)
- cl_sortkey: VARCHAR(86)
- cl_timestamp: TIMESTAMP

The CATEGORYLINKS table stores entries on a page of the type [[category:abc]], which places the page into the category "abc" (for which an associated page may or may not exist).

recentchanges
- rc_id: INTEGER(8)
- rc_timestamp: VARCHAR(14)
- rc_cur_time: VARCHAR(14)
- rc_user: INTEGER(5) (FK)
- rc_user_text: VARCHAR(255)
- rc_namespace: INTEGER(11)
- rc_title: VARCHAR(255)
- rc_comment: VARCHAR(255)
- rc_minor: TINYINT(3)
- rc_bot: TINYINT(3)
- rc_new: TINYINT(3)
- rc_cur_id: INTEGER(8) (FK)
- rc_this_oldid: INTEGER(8) (FK)
- rc_last_oldid: INTEGER(10)
- rc_type: TINYINT(3)
- rc_moved_to_ns: TINYINT(3)
- rc_new_len: INTEGER(10)
- rc_deleted: TINYINT(1)
- rc_logid: INTEGER(10)
- rc_log_type: VARCHAR(255)
- rc_log_action: VARCHAR(255)
- rc_params: BLOB

RECENTCHANGES contains details of the latest modifications done to the wiki. Used to generate the recent changes pages,

archive
- ar_namespace: INTEGER(11)
- ar_title: VARCHAR(255)
- ar_text: MEDIUMBLOB
- ar_comment: TINYBLOB
- ar_user: INTEGER(5)
- ar_user_text: VARCHAR(255)
- ar_timestamp: CHAR(14)
- ar_minor_edit: TINYINT(1)
- ar_flags: TINYBLOB
- ar_rev_id: INTEGER(8) (FK)
- ar_text_id: INTEGER(8)
- ar_deleted: TINYINT(1)

The content of the article remains in table TEXT. The deletion time is logged in table LOGGING.

page
- page_id: INTEGER(8)
- page_namespace: INTEGER(11)
- page_title: VARCHAR(255)
- page_restrictions: TINYBLOB
- page_counter: BIGINT(20)
- page_is_redirect: TINYINT(1)
- page_is_new: TINYINT(1)
- page_random: DOUBLE
- page_touched: CHAR(14)

essential metadata. The text of the page itself is stored in table TEXT.

oldimage
- oi_name: VARCHAR(255) (FK)
- oi_archive_name: VARCHAR(255)
- oi_size: INTEGER(8)
- oi_width: INTEGER(5)
- oi_height: INTEGER(5)
- oi_bits: INTEGER(3)
- oi_description: TINYBLOB
- oi_user: INTEGER(5) (FK)
- oi_user_text: VARCHAR(255)
- oi_timestamp: CHAR(14)

OLDIMAGE table holds information about old

The IMAGELINKS table stores all the links to images. This is good for a variety of functions, from link caching in the output functions to the "what links here" page.

imagelinks
- il_from: INTEGER(8) (FK)
- il_to: VARCHAR(255) (FK)

redirect
- rd_from: INTEGER(8) (FK)
- rd_namespace: INTEGER(11)
- rd_title: VARCHAR(255)

REDIRECT contains for each page that is currently a redirect (i.e., not for old revisions which are redirects) the id of the source page, the target namespace number, and the target page title without namespace.

cribes images and other ver, the image description other pages.

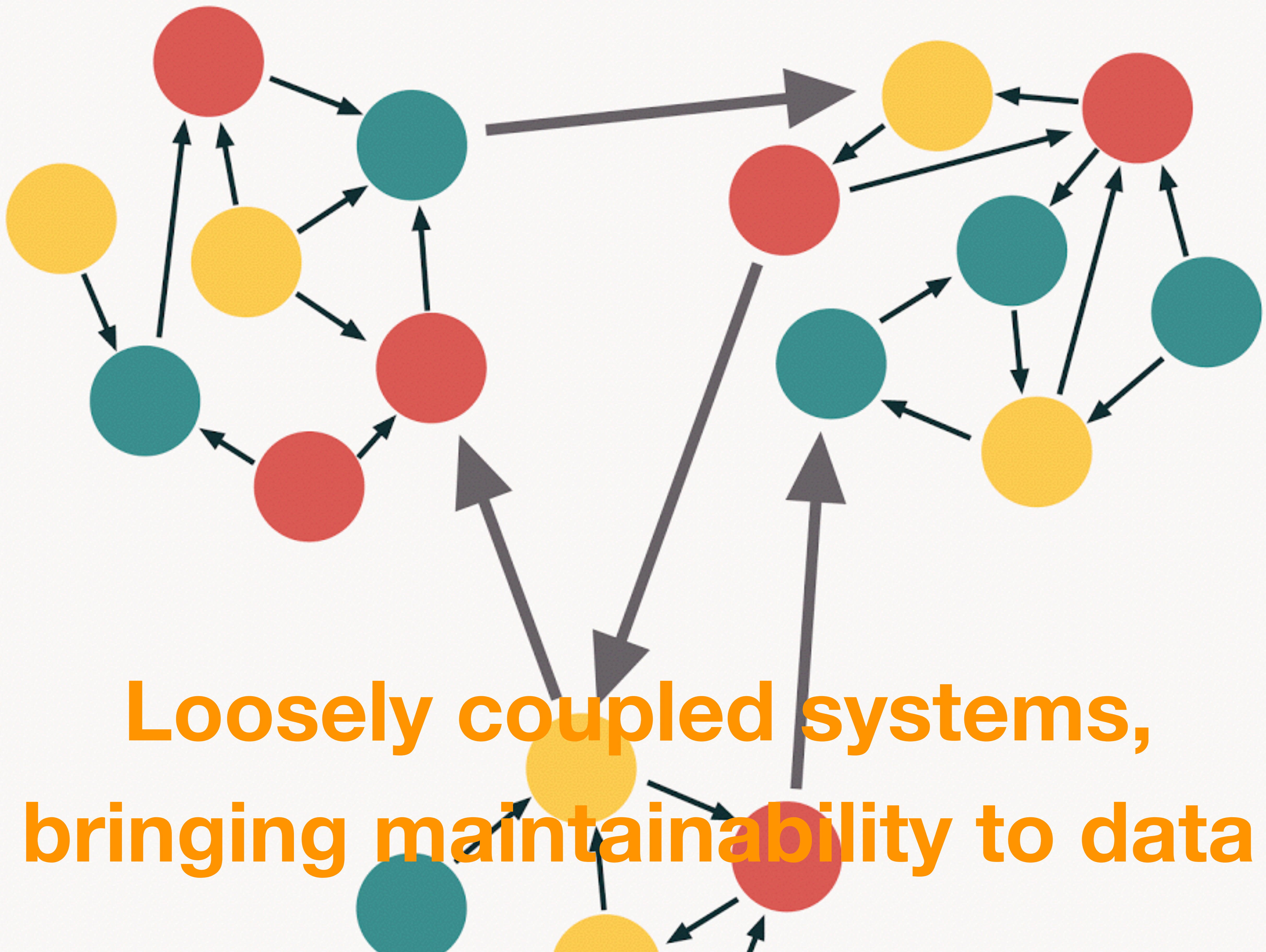
Making data available is a system integrations problem.

The challenges in connecting data

CHANGE
Change
£20 NOTES
£10 NOTES
£5 NOTES

Dealing with failure, is hard

The ever growing performance battle..



Loosely coupled systems, bringing maintainability to data

Building a Shantytown, The Big Ball of Mud

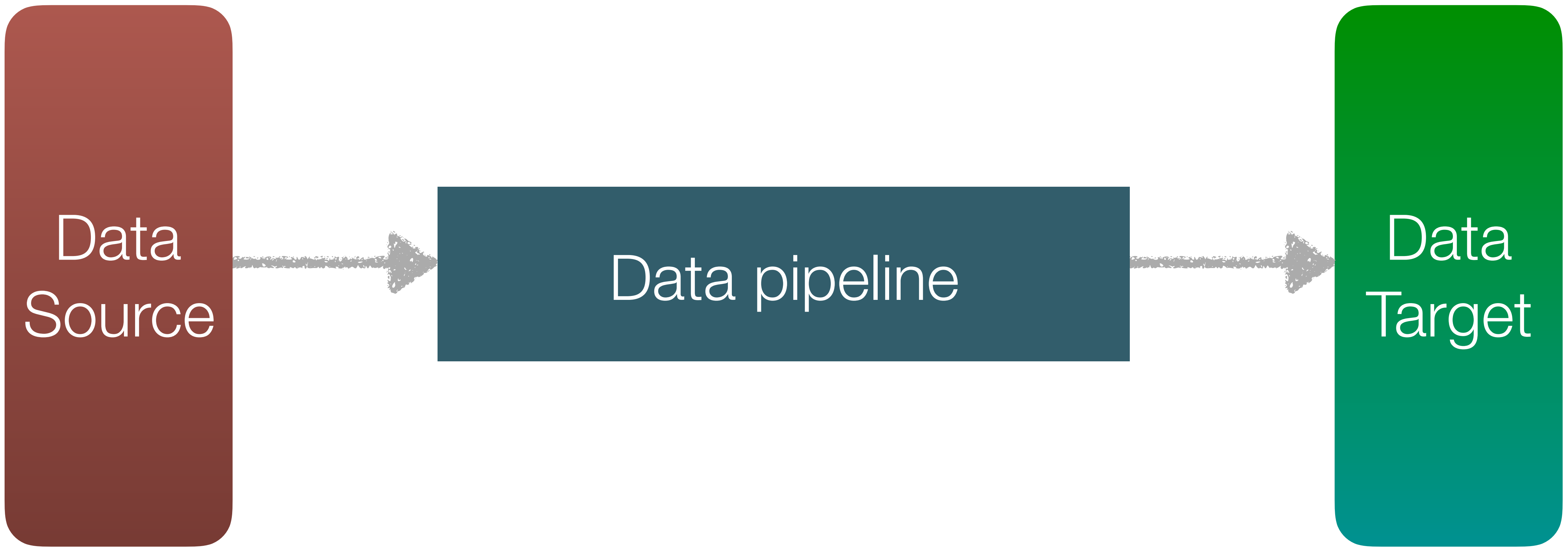
"organisations which design systems ... are constrained to produce designs which are copies of the communication structures of these organisations."

– M. Conway

From the warehouse to the real time era

Processing data at the speed of light

What is a data pipeline?



Because systems do not live in isolation anymore, they need to incorporate and/or generate data for other components.

Working with batches

The workers approach

Working in Batches

Batch processing is:

- The execution of a series of jobs/tasks
- In a computer, or group of computers.
- Without manual intervention.

A job is the single unit of work.

Working in Batches

Batches are a natural mapping from procedural and OO programming paradigms.

Implementation follow up from traditional multithread models.

An image uploaded using Sidekiq

```
class ProductImageUploader
  include Sidekiq::Worker

  def perform(image_id)
    s3_upload(data_for_image(image_id))
  end

  def self.upload(product, image_ids)
    image_ids.each do |image_id|
      perform_async(image_id)
    end
  end
end
```

Computing PI using spark.

```scala
/** Computes an approximation to pi */
object SparkPi {
  def main(args: Array[String]) {
    val spark = SparkSession
      .builder
      .appName("Spark Pi")
      .getOrCreate()
    val slices = if (args.length > 0) args(0).toInt else 2
    val n = math.min(100000L * slices, Int.MaxValue).toInt // avoid overflow
    val count = spark.sparkContext.parallelize(1 until n, slices).map { i =>
      val x = random * 2 - 1
      val y = random * 2 - 1
      if (x*x + y*y <= 1) 1 else 0
    }.reduce(_ + _)
    println("Pi is roughly " + 4.0 * count / (n - 1))
    spark.stop()
  }
}
```

Common best practices

- Make your jobs **small and simple**, to ensure performance and maintainability.
- Make your jobs **idempotent and transactional**, to ensure safety and residence.
    - At less once, exactly once,….
- Embrace **concurrency and asynchronous** api's to bring utilisation to the top.

The pros and cons of this approach

Building pipelines to transport data

Data plumbers since 1983

Working in streams

Stream processing is a computer paradigm that

- Provides a simplified parallel computation methodology.
- Given a stream of data, a series of (pipelined) operations can be applied.

Streams power algorithmic trading, RFID's, fraud detection, monitoring, telecommunications and many more.

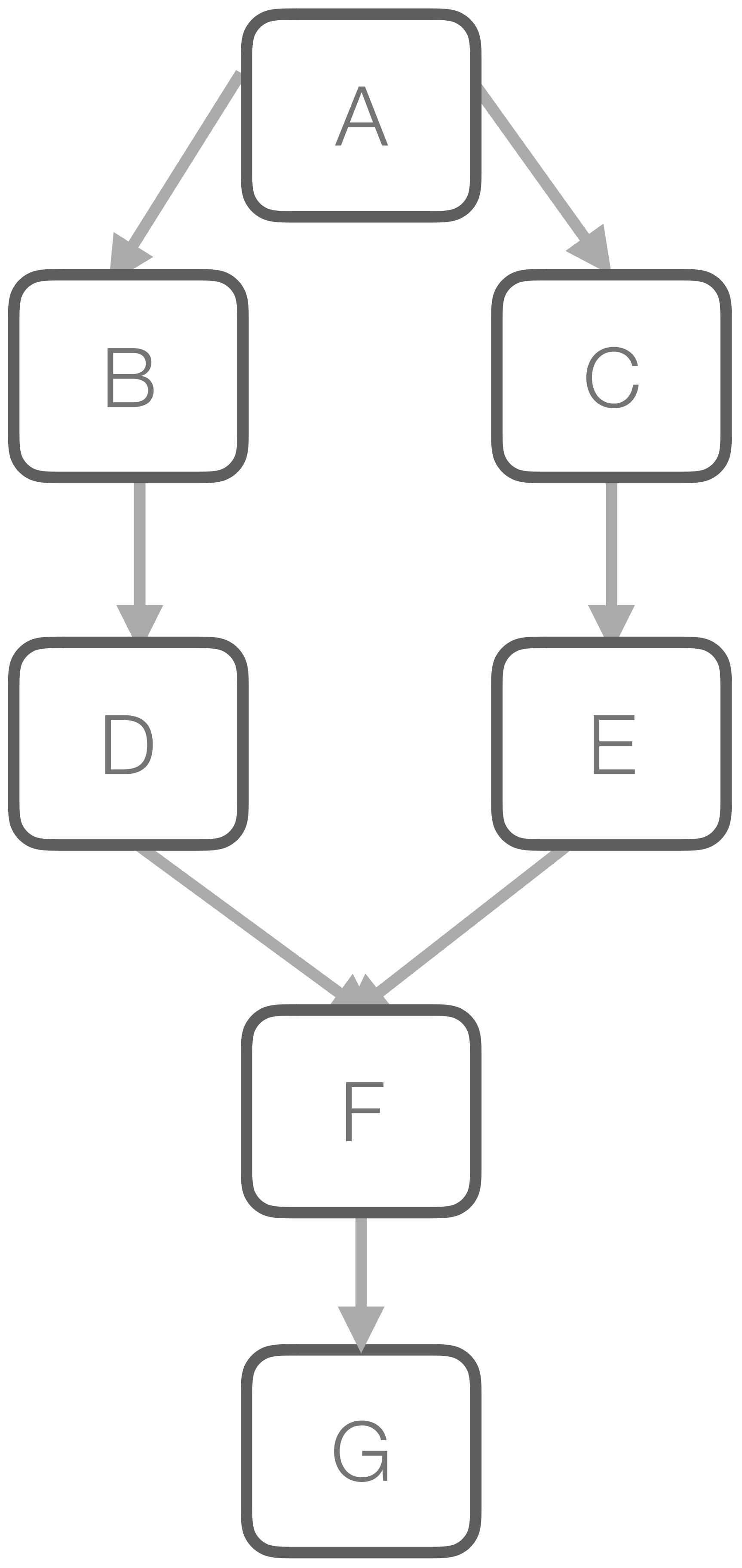
Working in streams

Related paradigms are:

- **Data Flow**: A program as data flowing between operations.
- **Event Stream Processing**: Databases, Visualisation, middleware and languages to build event based apps.
- **Reactive Programming**: Async programming paradigm concerned with data streams and propagation of change.

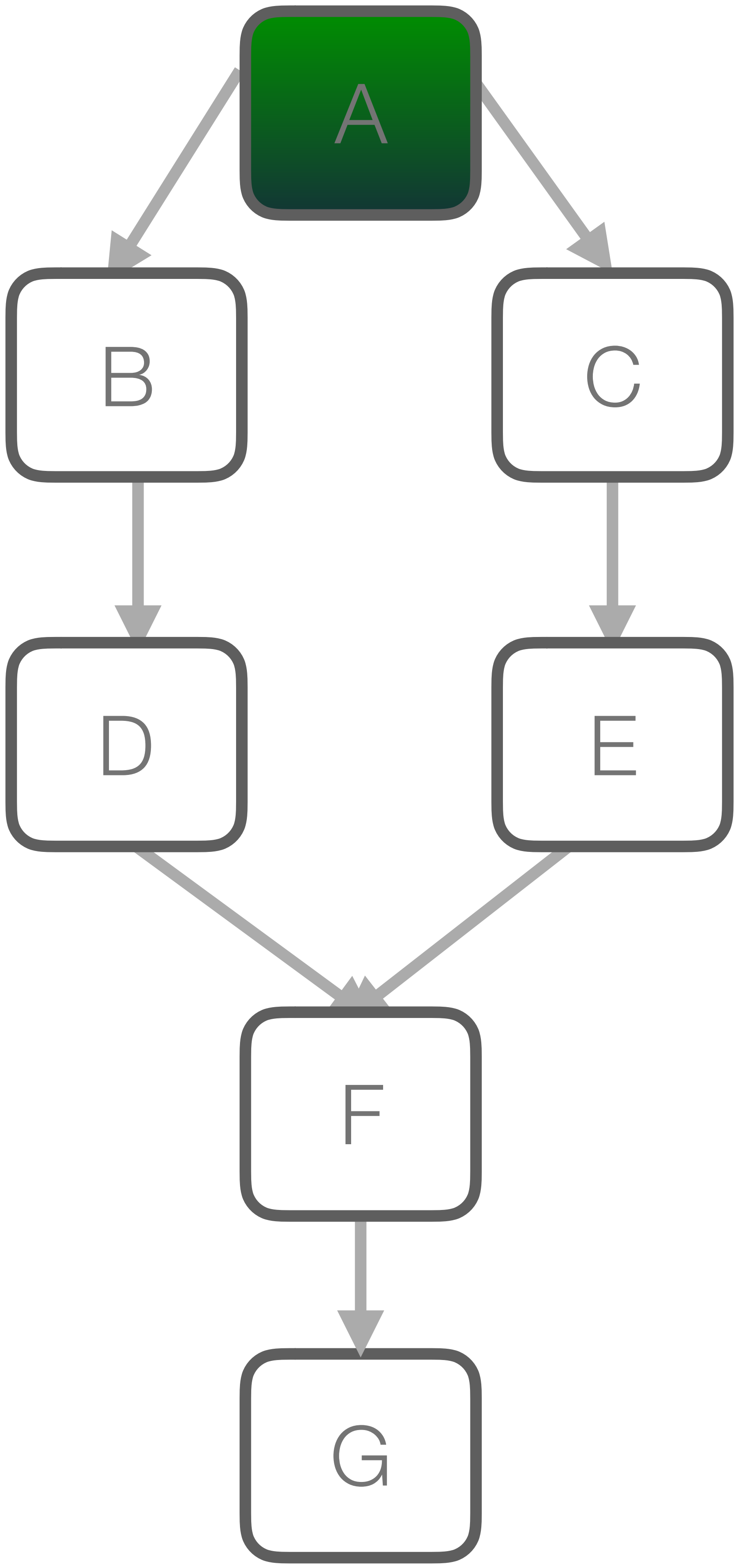
Data Flow Programming

- Model programs as a DAG graph of data flowing between operations.
- Data flow trough databases, brokers, streams…
- Operation types:
  - Enrichment
  - Drop/Throttle
  - Transform
- Backpressure, buffers, reactive.


$$\begin{tikzcd}
 & A \arrow[dl] \arrow[dr] & \\
B \arrow[d] & & C \arrow[d] \\
D \arrow[dr] & & E \arrow[dl] \\
 & F \arrow[d] & \\
 & G &
\end{tikzcd}$$

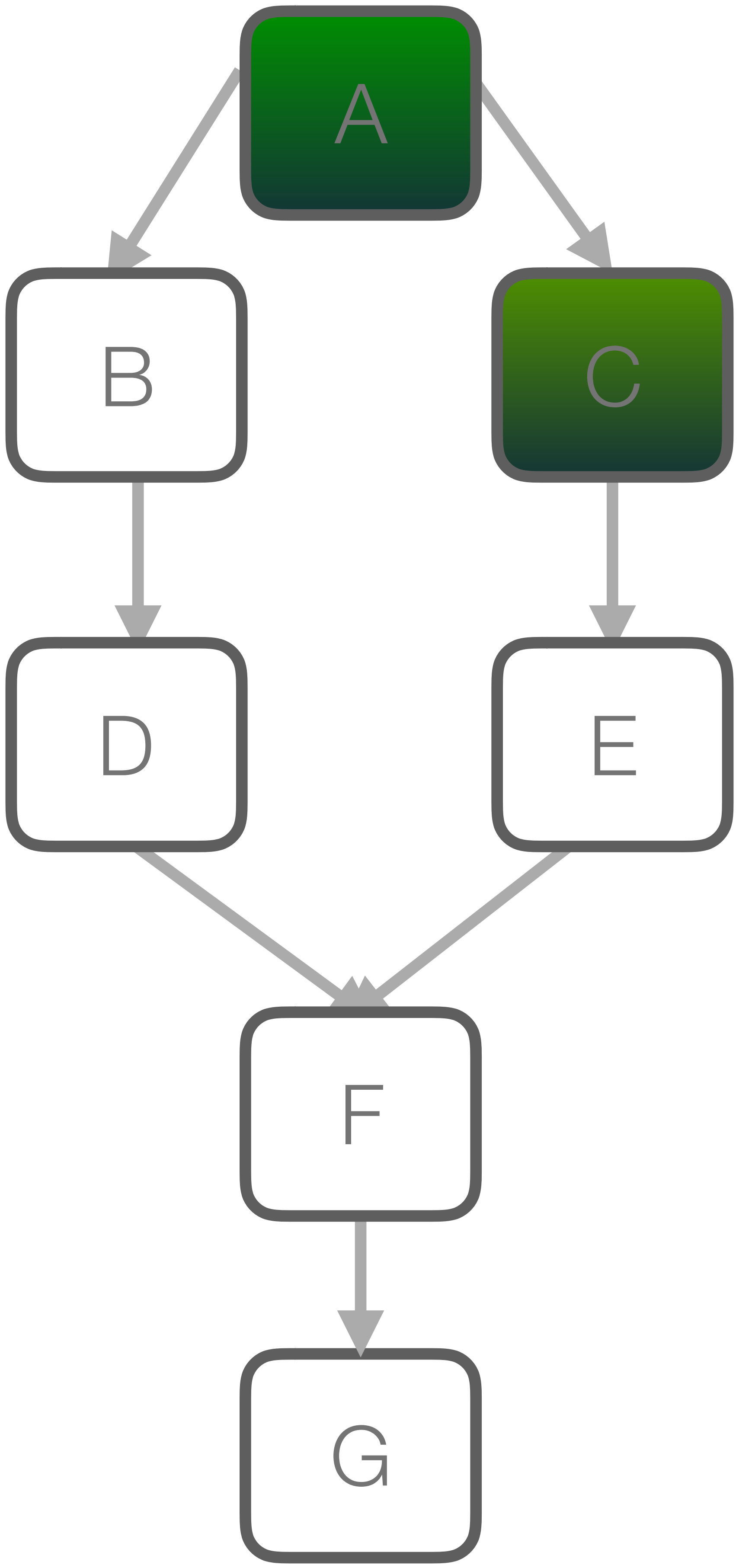
Backpressure

- It describe the build-up of data behind an I/O switch if the buffers are full and incapable of receiving any more data.
- The transmitting device halts the sending of data packets until the buffers have been emptied and are once more capable of storing information.


$$\begin{tikzcd}
 & A \arrow[dl] \arrow[dr] & \\
B \arrow[d] & & C \arrow[d] \\
D \arrow[dr] & & E \arrow[dl] \\
 & F \arrow[d] & \\
 & G &
\end{tikzcd}$$

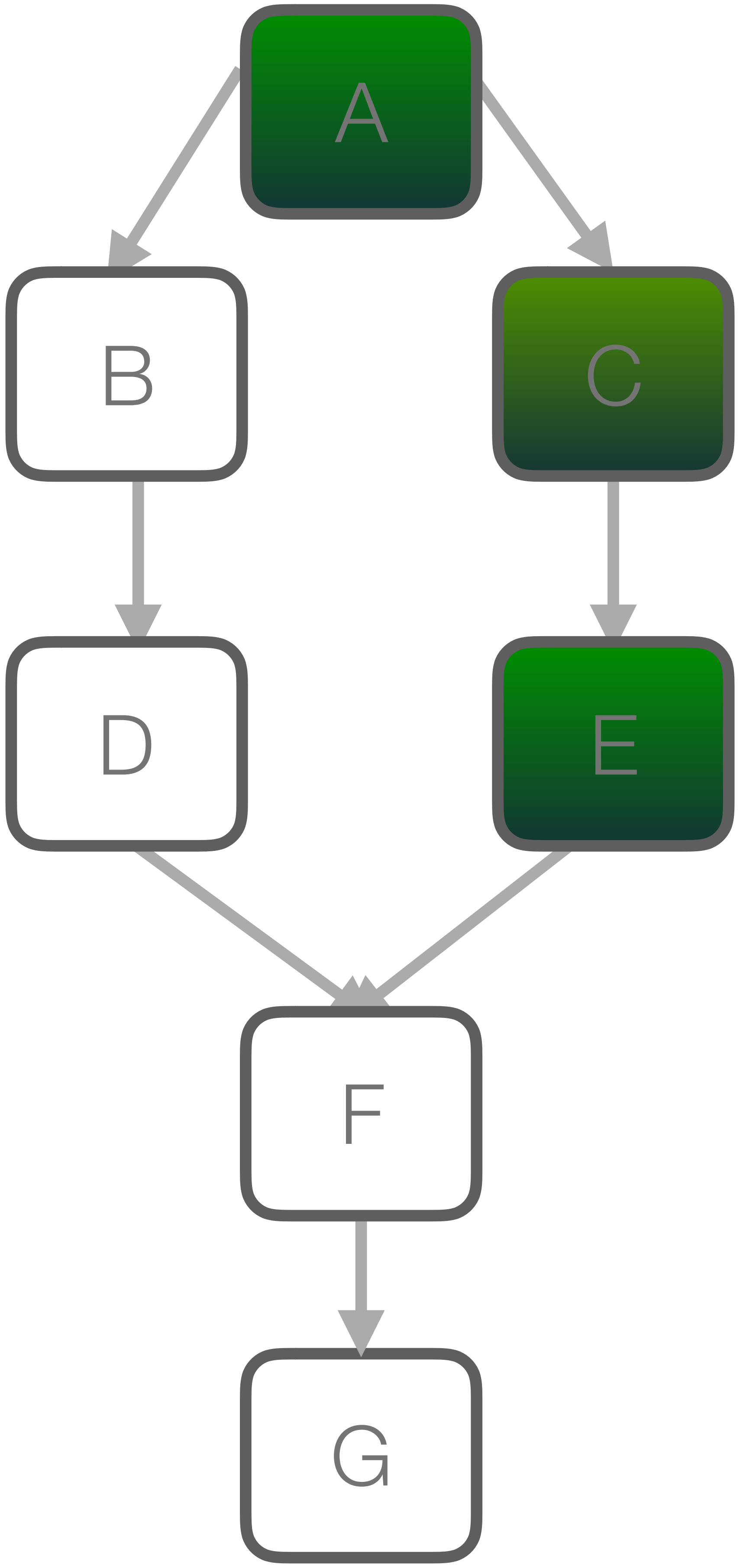
Backpressure

- It describe the build-up of data behind an I/O switch if the buffers are full and incapable of receiving any more data.
- The transmitting device halts the sending of data packets until the buffers have been emptied and are once more capable of storing information.



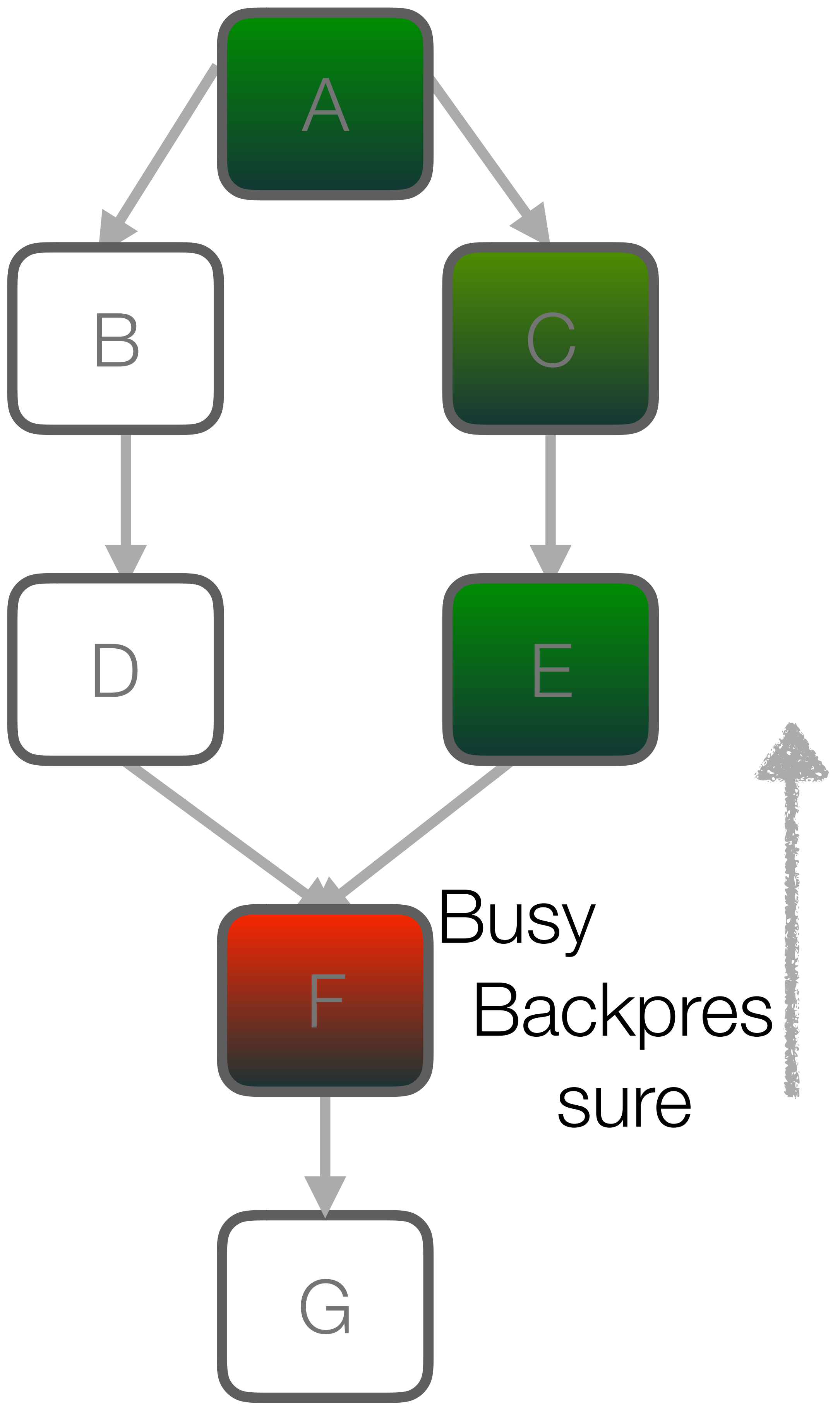
Backpressure

- It describe the build-up of data behind an I/O switch if the buffers are full and incapable of receiving any more data.
- The transmitting device halts the sending of data packets until the buffers have been emptied and are once more capable of storing information.



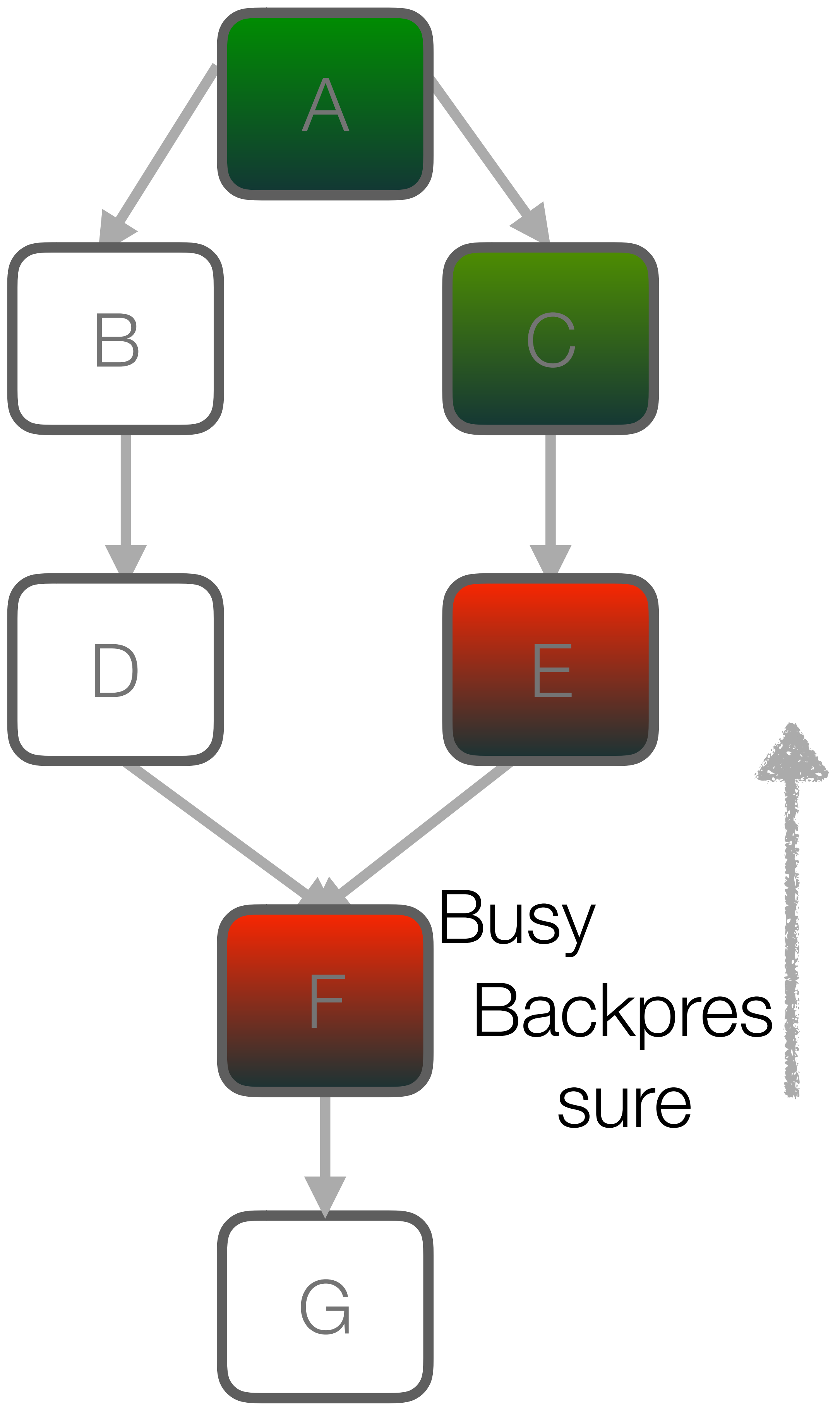
Backpressure

- It describe the build-up of data behind an I/O switch if the buffers are full and incapable of receiving any more data.
- The transmitting device halts the sending of data packets until the buffers have been emptied and are once more capable of storing information.



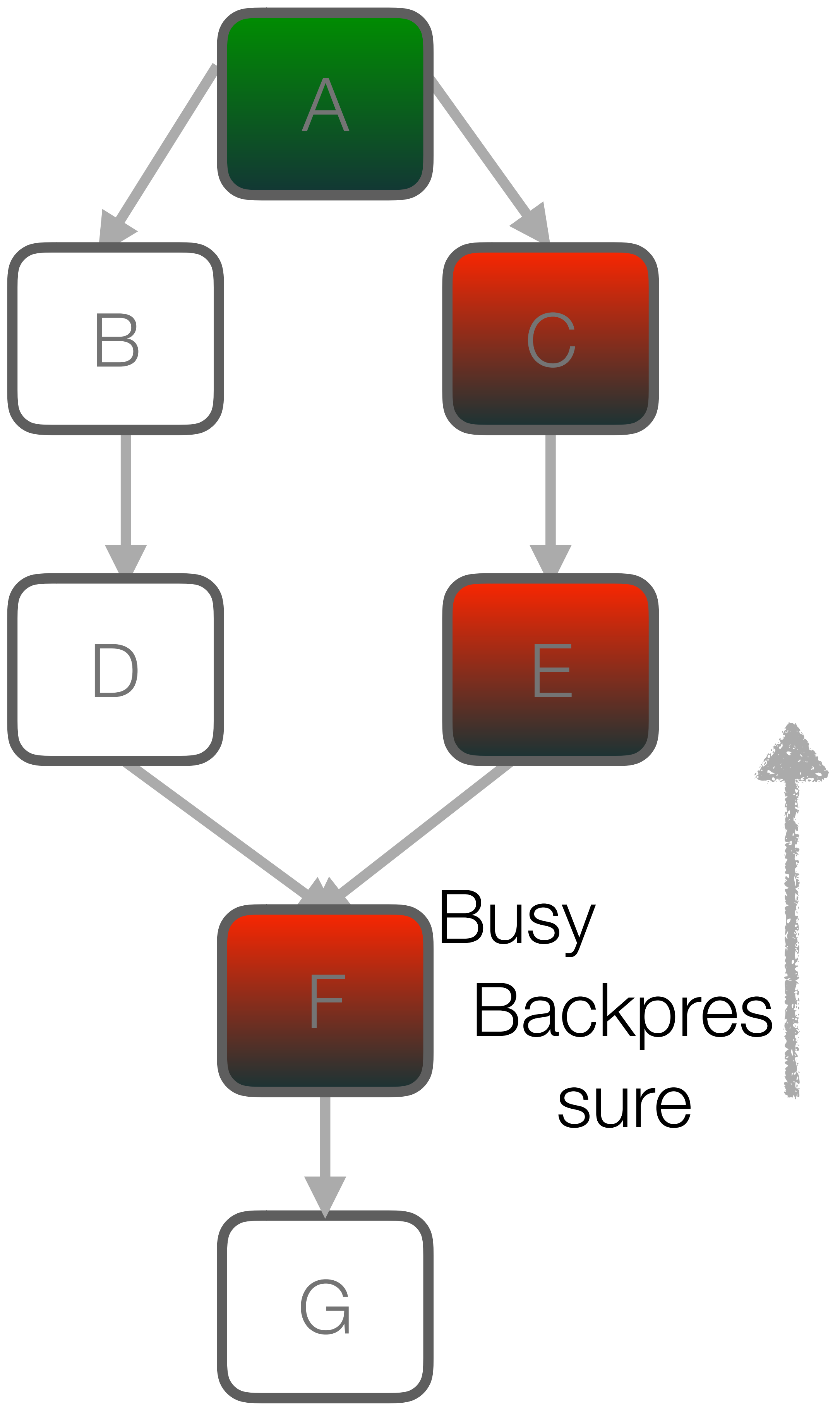
Backpressure

- It describe the build-up of data behind an I/O switch if the buffers are full and incapable of receiving any more data.
- The transmitting device halts the sending of data packets until the buffers have been emptied and are once more capable of storing information.



Backpressure

- It describe the build-up of data behind an I/O switch if the buffers are full and incapable of receiving any more data.
- The transmitting device halts the sending of data packets until the buffers have been emptied and are once more capable of storing information.



Reactive Streaming

When one component is struggling to keep-up, the system as a whole needs to respond in a sensible way. It is unacceptable for the component under stress to fail catastrophically or to drop messages in an uncontrolled fashion. Since it can’t cope and it can’t fail it should communicate the fact that it is under stress to upstream components and so get them to reduce the load.

http://www.reactive-streams.org/

http://www.reactivemanifesto.org/

A word count on reddit with Akka Streams.

```
def main(args: Array[String]): Unit = {
  // 0) Create a Flow of String names, using either
  //    the argument vector or the result of an API call.
  val subreddits: Source[String, Unit] =
    if (args.isEmpty)
      Source(RedditAPI.popularSubreddits).mapConcat(identity)
    else
      Source(args.toVector)

  val res: Future[Map[String, WordCount]] =
    subreddits
    .via(fetchLinks)
    .via(fetchComments)
    .runWith(wordCountSink)

  res.onComplete(writeResults)

  as.awaitTermination()
}
```

A word count on reddit with Akka Streams.

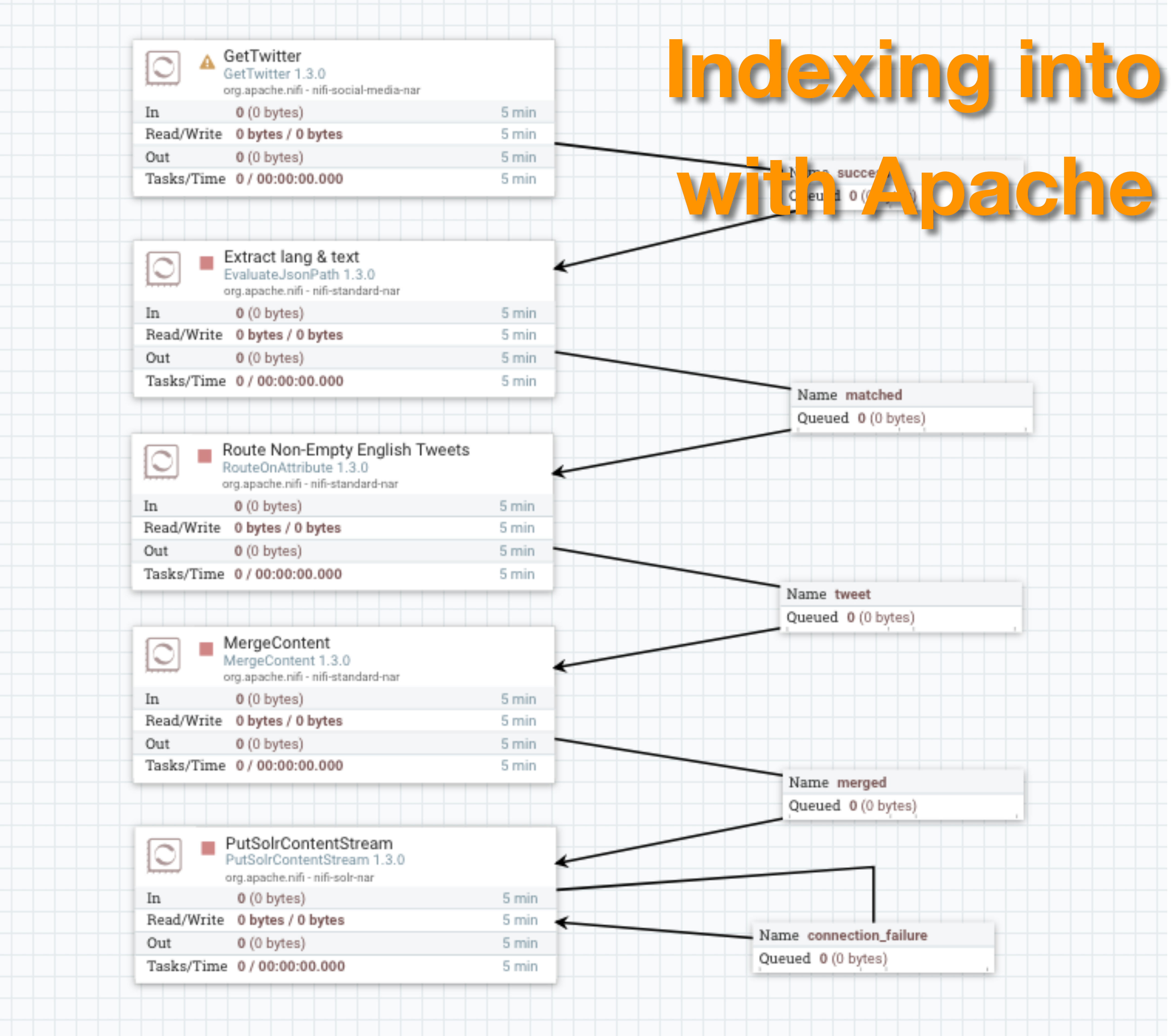
```scala
val fetchLinks: Flow[String, Link, Unit] =
  Flow[String]
      .via(throttle(redditAPIRate))
      .mapAsyncUnordered( subreddit => RedditAPI.popularLinks(subreddit) )
      .mapConcat( listing => listing.links )


val fetchComments: Flow[Link, Comment, Unit] =
  Flow[Link]
      .via(throttle(redditAPIRate))
      .mapAsyncUnordered( link => RedditAPI.popularComments(link) )
      .mapConcat( listing => listing.comments )
```

A word count on reddit with Akka Streams.

```
val wordCountSink: Sink[Comment, Future[Map[String, WordCount]]] =
  Sink.fold(Map.empty[String, WordCount])(
    (acc: Map[String, WordCount], c: Comment) =>
      mergeWordCounts(acc, Map(c.subreddit -> c.toWordCount))
  )
```

Indexing into Solr with Apache NiFi



GetTwitter
GetTwitter 1.3.0
org.apache.nifi - nifi-social-media-nar

In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

Name succe
Queued 0 (

Extract lang & text
EvaluateJsonPath 1.3.0
org.apache.nifi - nifi-standard-nar

In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

Name matched
Queued 0 (0 bytes)

Route Non-Empty English Tweets
RouteOnAttribute 1.3.0
org.apache.nifi - nifi-standard-nar

In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

Name tweet
Queued 0 (0 bytes)

MergeContent
MergeContent 1.3.0
org.apache.nifi - nifi-standard-nar

In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

Name merged
Queued 0 (0 bytes)

PutSolrContentStream
PutSolrContentStream 1.3.0
org.apache.nifi - nifi-solr-nar

In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

Name connection_failure
Queued 0 (0 bytes)

Streaming projects

Apache Flink

APACHE Spark™

akka streams

APACHE nifi

Apache **Kafka**

Apache **Samza**

STORM

samza

Challenges in data plumbing

- Systems growth is most commonly mapping internal communication channels among the organisation.
- This introduces challenges on several areas:
  - Handle failure.
  - Keep the data processes low latency.
  - Changes in communication and data.
  - Data availability and governance.

The pros and cons of this approach

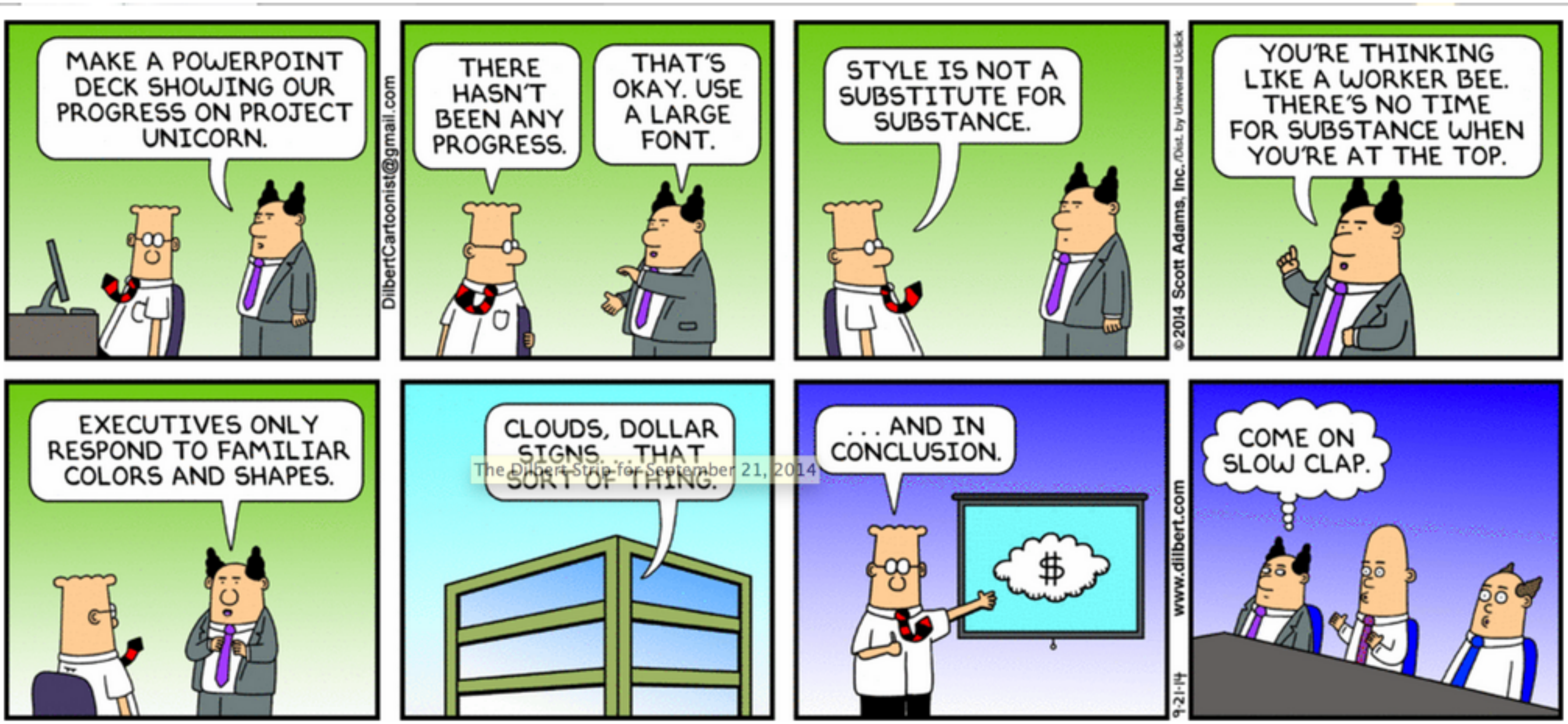
Scaling Human Data Communication

Handling communication patterns

“Data pipelines emerge to automate the communications structures of the organisation”

–Your data engineer next door

Is all about communication, right?


MAKE A POWERPOINT DECK SHOWING OUR PROGRESS ON PROJECT UNICORN.
THERE HASN'T BEEN ANY PROGRESS.
THAT'S OKAY. USE A LARGE FONT.
STYLE IS NOT A SUBSTITUTE FOR SUBSTANCE.
YOU'RE THINKING LIKE A WORKER BEE. THERE'S NO TIME FOR SUBSTANCE WHEN YOU'RE AT THE TOP.
EXECUTIVES ONLY RESPOND TO FAMILIAR COLORS AND SHAPES.
CLOUDS, DOLLAR SIGNS . . . THAT SORT OF THING.
The Dilbert Strip for September 21, 2014
. . . AND IN CONCLUSION.
$
COME ON SLOW CLAP.
DilbertCartoonist@gmail.com
©2014 Scott Adams, Inc./Dist. by Universal Uclick
www.dilbert.com
9-21-14

Accessing data in a more reliable way

Problems usually pop because of changes in:

- Data expectations: When inbound teams need to change the internal data representation, volumes or schemas unexpected results are expected.

- Communication channels: A software platform is all about communication between components, also data pipelines, if they are changed users should handle it.

The shared schema registry

A centralised schema registry is a service where all organisation wide schemas are made accessible, facilitating access across teams, in detail benefits are:

- Simplify organisational data management challenges.
- Build resilient data pipelines.
- Record schema evolution.
- Facilitate data discovery across teams.
- Stream cost efficient data platforms.
- Policy enforcements.

The shared schema registry

- Popular ways to achieve this is by storing schemas in formats such as Avro, Protocol Buffers or Thirft, preferable the first one.
- Curiously there exist many private implementations, the first opensouce one is the kafka centric schema-registry by Confluent INC.
- Consumer-Driven Contracts: Similar approach introduced in 2006 by Ian Robinson at ThoughtWorks.
    - Popular implementation: pact.io

Domain Driven Design

Domain-driven design (DDD) is an approach to software development for complex needs by connecting the implementation to an evolving model.

One of the premises is the creative collaboration between technical and domain experts to refine the model.

https://en.wikipedia.org/wiki/Domain-driven_design

References

- Pat Helland. Accountants don't use erasers. https://blogs.msdn.microsoft.com/pathelland/2007/06/14/accountants-dont-use-erasers/

- Martin Kleppmann. Turning databases inside out. https://www.confluent.io/blog/turning-the-database-inside-out-with-apache-samza/

- Martin Kleppman. Designning Data Intensive Applications. https://dataintensive.net/ .O'Reilly.

References

- Gregor Hohpe. Enterprise Integration Patterns. http://www.enterpriseintegrationpatterns.com/
- Matt Welsh, et all. SEDA: An Architecture for Well-Conditioned, Scalable Internet Services. http://www.sosp.org/2001/papers/welsh.pdf

LOONEY TUNES

Thanks a lot, Questions?

The Quantum Mechanics of Data Pipelines

Pere Urbon-Bayes
Data Wrangler
pere.urbon @ { gmail.com, acm.org }
http://www.purbon.com