Top 20 Data Quality Solutions for Data Science

Data Science & Business Analytics Meetup
Denver, CO 2015-01-21

Ken Farmer

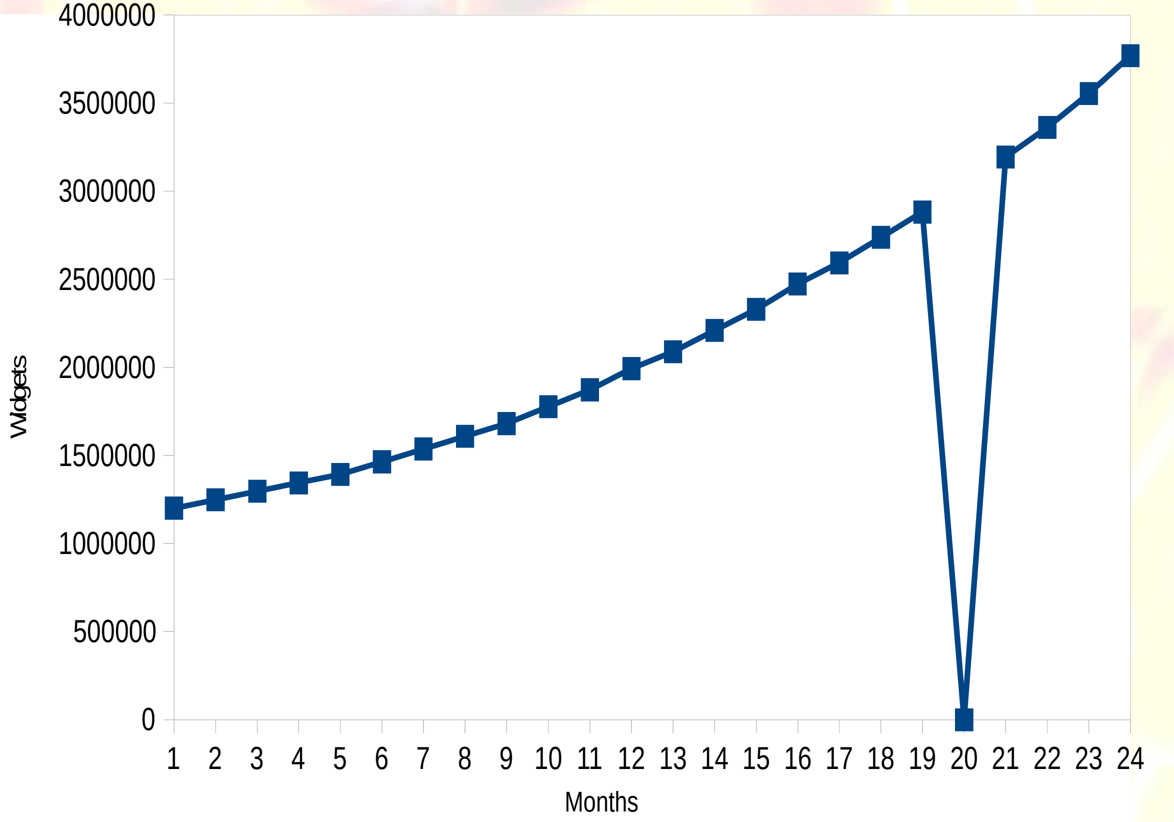
DQ Problems for Data Science Loom Large & Frequently

Impacts Include:

- Strikingly visible defects
- Bad Decisions
- Loss of credibility
- Loss of revenue

Types of Problems Include:

- Requirement & Design Defects
- Misinterpretation Errors
- Source Data Defects
- Process Errors




Lets Talk about Solutions
But keep in mind...
Proportionality is Important

Solution #1: Quality Assurance (QA)

“If it's not tested it's broken” - Bruce Eckel

Tests of application and system behavior
- input assumptions
- performed after application changes

Programmer Testing:
- Unit Testing
- Functional Testing

QA Team Testing:
- Black Box Testing
- Regression Testing
- System and Integration Testing

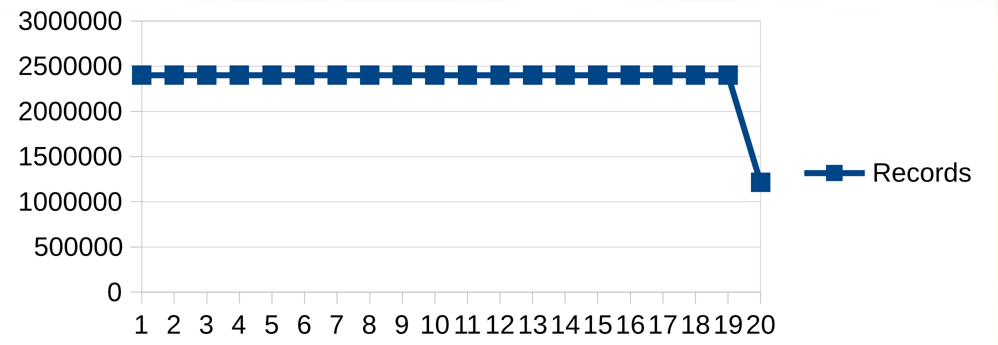
Data Scientist Testing:
- All of the above
- Especially Programmer

Solution #2: Quality Control (QC)

Because the environment & inputs are ever-changing

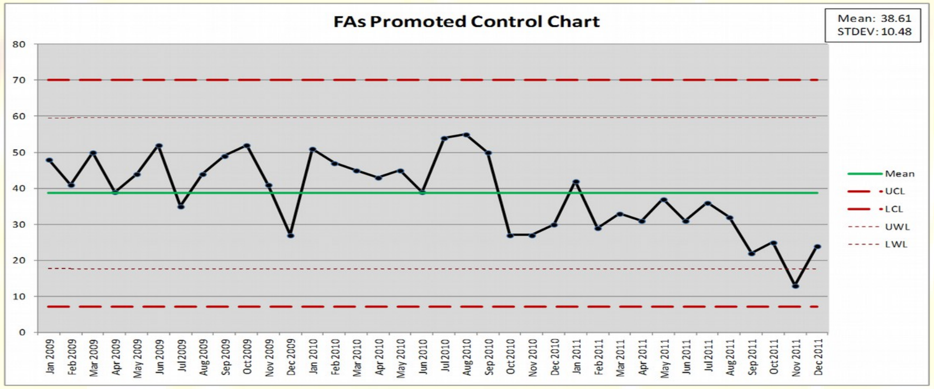
Track & analyze record counts:

- identify partial data sets
- identify upstream changes



Track & analyze rule counts:

- identify upstream changes



Track & analyze replication & aggregation consistency:

- identify process failures

Solution #3: Data Profiling

Find out what data looks like BEFORE you model it

Save enormous amount of time:

- quickly get the type, size, cardinality, unk values
- share profiling results with users

Example Deliverables:

- What is the distribution of data for every field?
- How do partitions affect data distributions?
- What correlations exist between fields?

Name	f250_499
Field Number	59
Wrong Field Cnt	0
Type	string
Min	C
Max	M
Unique Values	12
Known Values	11
Case	mixed
Min Length	1
Max Length	1
Mean Length	1.0
Top Values	
	x 10056 occurrences
E	x 1299 occurrences
F	x 969 occurrences
G	x 358 occurrences
C	x 120 occurrences
H	x 89 occurrences
I	x 36 occurrences
J	x 18 occurrences
M	x 12 occurrences
K	x 11 occurrences
e	x 4 occurrences

Solution #4: Organization

So people don't use the wrong data
or the right data incorrectly

Manage Historical Data:

- Migrate old data
  - Schemas
  - Rules

- Curate adhoc files
  - Segregate
  - Name
  - Document
  - Eliminate

Simplify Data Models:

- Consistency in naming, types, defaults
- Simplicity in relationships and values

Solution #5: Process Auditing

Because with a lot of moving parts comes a lot of failures

Addresses:
- Slow processes
- Broken processes

Challenge with Streaming:
- Helps to create artificial batch concept from timestamp within the data

Helps identify:
- Duplicate loads
- Missing files
- Pipeline status

Features:
- Tracks all processes: start, stop, times, return codes, batch_ids
- Alerting

Example Products:
- Graphite - focus: resources
- Nagios - focus: process failure
- Or what's built-into every ETL tool

(the inspecting metaphor,
not the searching metaphor)

Solution #6: Full Automation

Because manual processes account for a majority of problems

Solution Characteristics:
- Test catastrophic failures
- Automate failure recoveries
- Consider Netflick's Chaos Monkey

Addresses:
- Duplicate data
- Missing data

Top Causes:
- manual process restarts
- manual data recoveries
- manual process overrides

Solution #7: Changed Data Capture

Because identifying changes is harder than most people realize

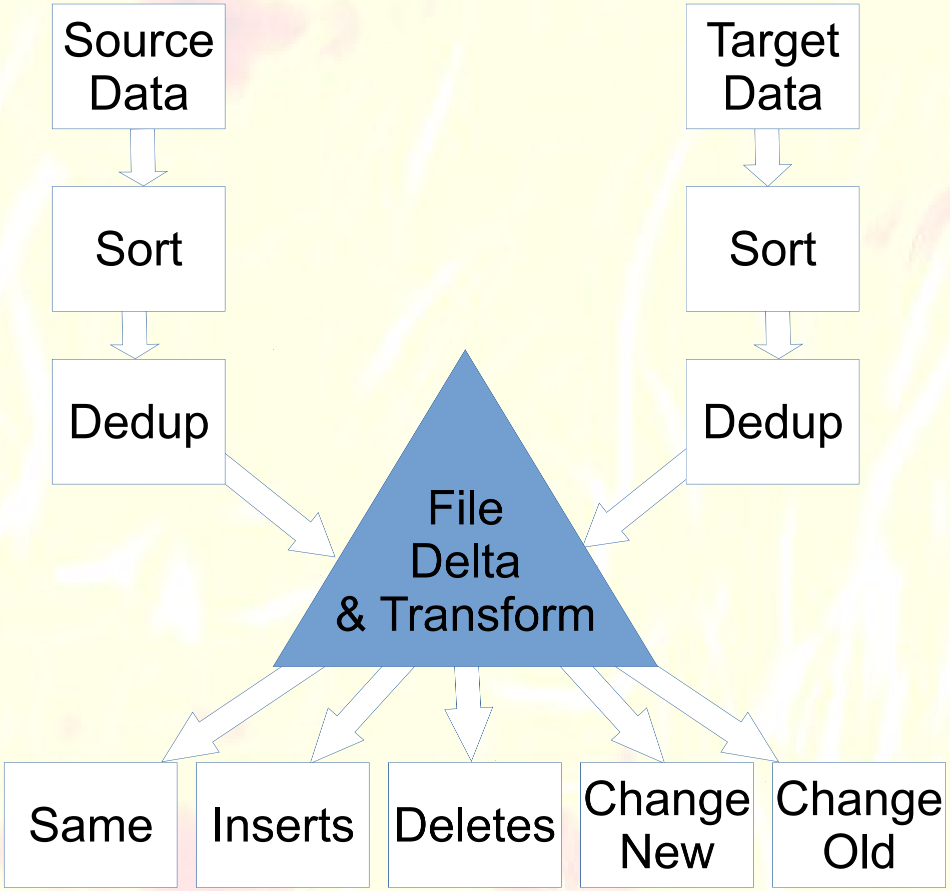
Typical Alternatives:

Application Timestamp:
- pro: may already be there
- con: reliability challenges

Triggers:
- pro: simple for downstream
- con: reliability challenges
- con: requires changes to source

File-Image Delta:
- pro: implementation effort
- con: very accurate

File Image Delta Example



Solution #8: Master Data Management

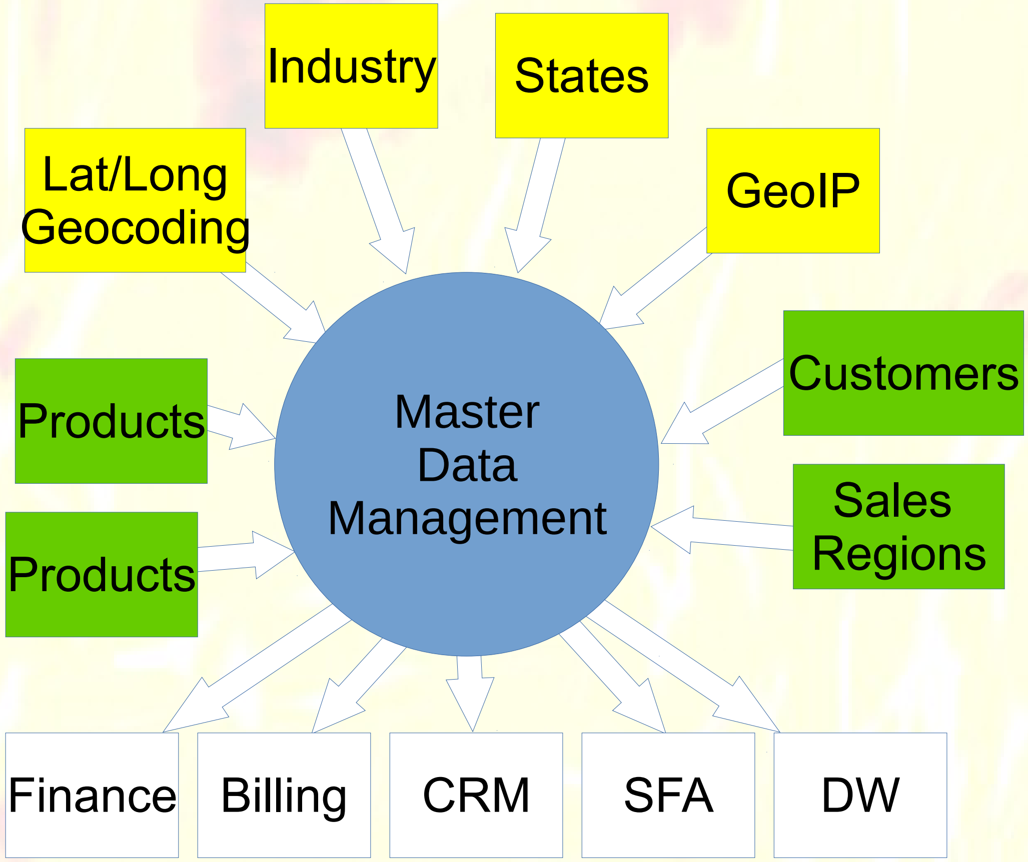
Because sharing reference data eliminates many issues

Addresses:

- Data consistency between multiple systems

Features:

- Centralized storage of reference data
- Versioning of data
- Access via multiple protocols

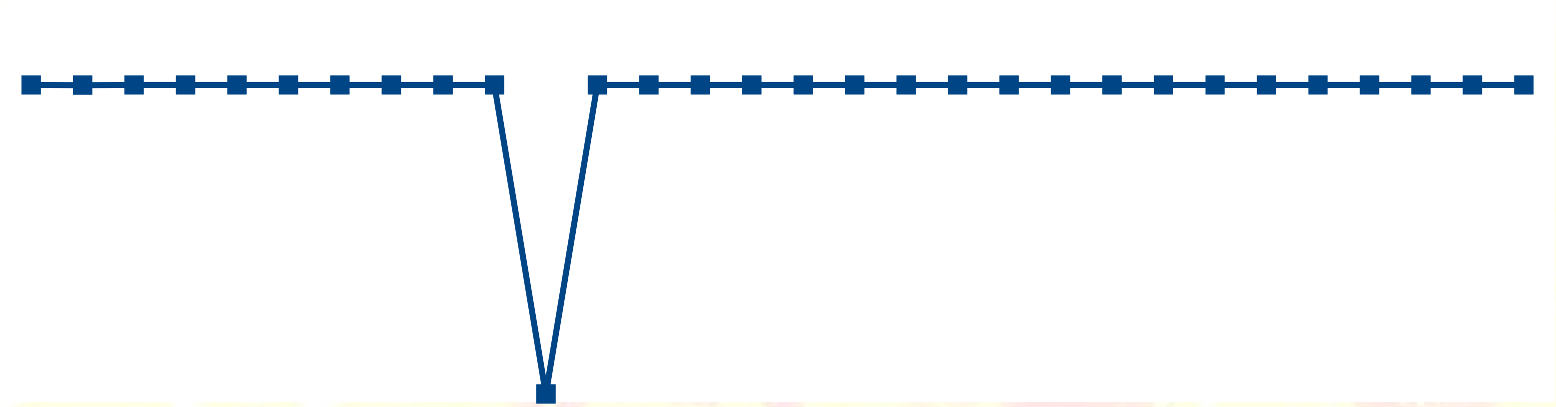


Solution #9: Extrapolate for Missing Data

*Because **if done well** it can simplify queries*

Features:

- Requires sufficient data to identify pattern
- Identify generated data (see Quality Indicator & Dimension)



Solution #10: Crowd-sourced Data Cleansing

Because cleansing & enrichening data can benefit from consensus

Features:
- Collect consensus answers from workers
- Workers can be from external market
- Workers can be from your team

Example Product:
- CrowdFlower
- Mechanical Turk

Simple Data Scenario:
- Correct obvious problems
  - Spelling
  - Grammar
  - Missing descriptions
- Use public resources

Complex Data Scenario:
- Correct sophisticated problems
  - Provide scores
  - Provide descriptions
  - Correct complex data
- Use internal & external resources
- Leverage crowdsourcing services for coordination

Solution #11: Keep Original Values

Because your transformations will fail & you will change your mind

Options:

- keep archive copy of source data
  - Pro: can be very highly compressed
  - Pro: can be kept off-server
  - Con: cannot be easily queried
- Or keep with transformed data
  - Pro: can be easily queried
  - Con: may be used when it should not be
  - Con: may double volume of data to host

os_orig	os
Win2k	win2000
MS Win	win2000
Win 2000	win2000
Windoze 2k	win2000
Windows 2000 SP4	win2000
MS Win 2k SP2	win2000
ms win 2000 server ed	win2000
win2ksp3	win2000
Win 2k server sp9	win2000

Solution #12: Keep usage logs

Because knowing who got bad data can help you minimize impact

Solution Types:

- Log application queries
  - Pro: database audit tools can do this
  - Con: requires process audit logs to translate to data content

- Store data that was delivered
  - Pros: can precisely identify who got what when
  - Con: requires dev, only works with certain tools (ex: restful API)
  - Con: doesn't show what wasn't delivered

Solution #13: Cleanup at Source

Because it's cheaper to clean at the source than downstream

Always be prepared to:

- Clean & scrub data in-route to target database

But always try to:

- give clean-up tasks to source system

Solution #14: Static vs Dynamic, Strong vs Weak Type & Structure

Because this is debated endlessly

Static vs Dynamic Schemas:

- Dynamic Examples: MongoDB, JSON in Postgres, etc
- Dynamic Schemas – optimize for writer – at cost of reader

Static vs Dynamic Typing:

- static typing provide fewer defects*
- but maybe not better data quality

Declarative Constraints:

- Ex: primary key, foreign key, Uniqueness, and check constraints
- Code: "ALTER TABLE foo ADD CONSTRAINT ck1 CHECK(open_date <= close_date) "

Solution #15: Data Quality Indicator & Dimension

Because it allows users to know what is damaged

Example:

- Single id that represents status for multiple fields
- Bitmap example:
  - bitmap 16-bit integer
  - each bit represents a single field
  - bit value of 0 == good, value of 1 == bad
  - Translate integer to field status with UDF or table

quality_id	result	col1	col2	col3	col4	col5	col6	col7	col8	colN
0000000000000000	good	good	good	good	good	good	good	good	good	good
0000000000000001	bad	bad	good	good	good	good	good	good	good	good
0000000000000010	bad	good	bad	good	good	good	good	good	good	good
0000000000000011	bad	bad	bad	good	good	good	good	good	good	good
0000000000000100	bad	good	good	bad	good	good	good	good	good	good
0000000000000101	bad	bad	good	bad	good	good	good	good	good	good
0000000000000111	bad	bad	bad	bad	good	good	good	good	good	good

Solution #16: Generate Test Data

Because production data is of limited usefulness

Various Types:

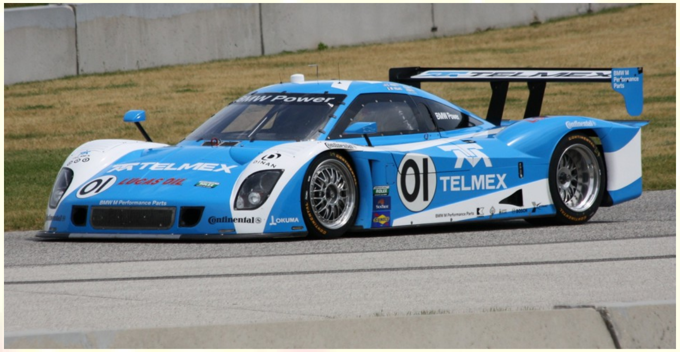
- Deterministic:
  - Contains very consistent data
  - Great for benchmarking
  - cheap to build
- Realistic:
  - Produced through a simulation
  - Great for demos
  - Great for sanity-checking analysis
  - Hard to build
- Extreme:
  - Contains extreme values
  - Great for bounds-checking
  - cheap to build

Solution #17: Use the Data!

Data unused:
- Will decay over time
- Will lose pieces of data
- Will lose metadata

Data in a report:
- Looks great
- Can see obvious defects
- But doesn't drive, doesn't get tested

Data driving a business process:
- Looks great
- Gets tested & corrected – in order to run the business.

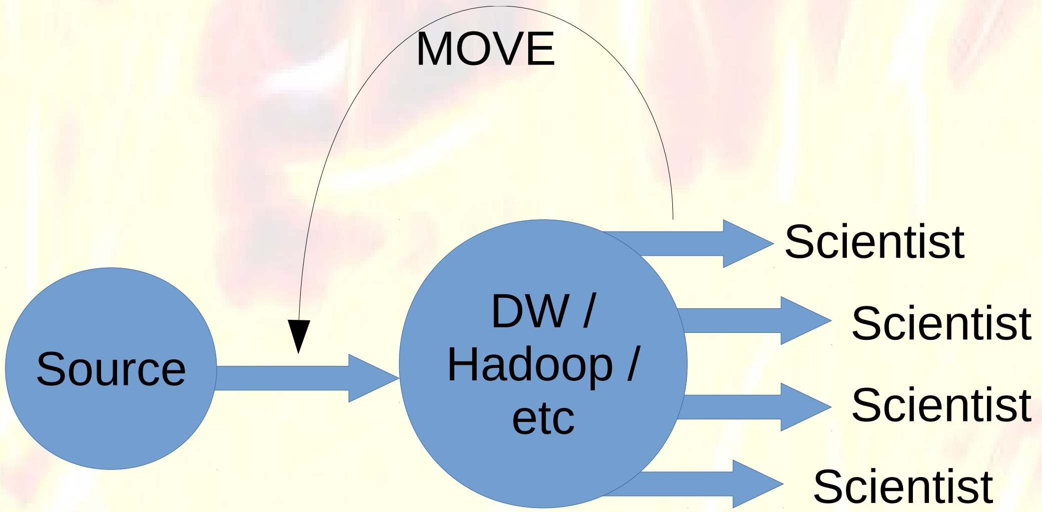


Solution #18: Push Transformations Upstream

Because you don't want to become source implementation experts

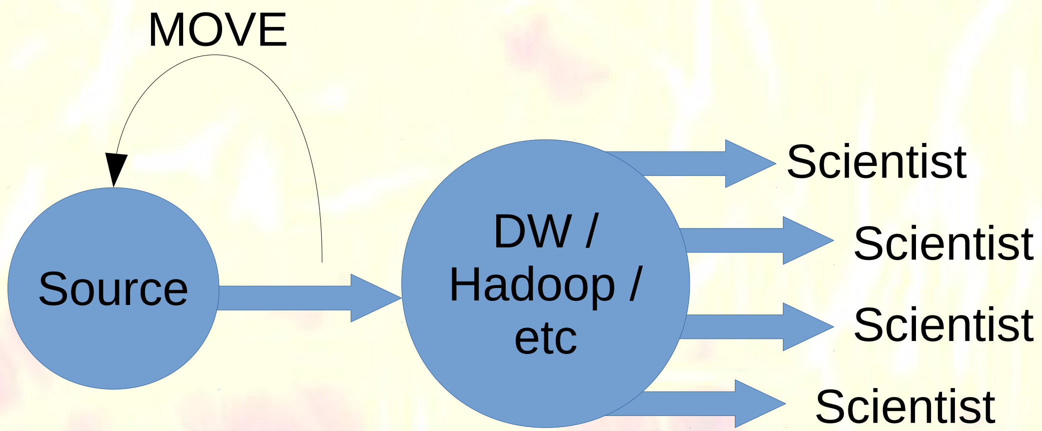
From Data Scientists to ETL Developers:

- Eliminates inconsistencies between runs



From ETL Developers to Source System:

- Eliminate unnecessary source system knowledge
- Decouples systems



Solution #19: Documentation (Metadata)

Field Metadata:

- Name
- Description
- Type
- Length
- Unknown value
- Case
- Security

Extended Metadata:

- Lineage
- Data Profile
  - Common values/codes
  - Their meaning
  - Their frequency
- Validation rules & results
- Transformation rules

Source Code:

```
if gender == 'm':
        return 'male'
else:
        return 'female'
```

Report & Tool Documentation:

- Description
- Filtering Rules
- Transformation Rules

Solution #20: Data Defect Tracking

Because you won't remember why a set of data was bad in 6 months

Like Bug-Tracking, but for sets of bad data:

- Will explain anomalies later
- Can be used for data annotation
- Is simple, just needs to be used

Bonus Solution #21: Change the Culture (ha)

Because you need support for priorities, resources, and time

Single Most Important thing to do:

- Establish policy of transparency = 90%
- Share data with customers, stakeholders, owners, users

Everything else results from transparency:

- Establish policy of automation
- Establish policy of measuring
- Plus everything we already covered

What doesn't work?

- Ask management to mandate quality

Resources & Thanks

International Association for Information & Data Quality (IAIDQ)
http://www.iqtrainwrecks.com/
Improving Data Warehouse and Business Information Quality, Larry English
The Data Warehouse Institute (TDWI)

1-A Large Scale Study of Programming Languages
and Code Quality in Github
http://macbeth.cs.ucdavis.edu/lang_study.pdf

Solution List

Solution	Source	ETL	DEST/DW	Consume
1. QA		HIGH	LOW	MEDIUM
2. QC		HIGH	MEDIUM	MEDIUM
3. Data Profiling	HIGH			
4. Organize			HIGH	
5. Process Auditing		HIGH	MEDIUM	
6. Full Automation		HIGH	HIGH	
7. Changed Data Capture		HIGH		
8. MDM	HIGH	HIGH	HIGH	
9. Extrapolate Missing Data		HIGH	HIGH	
10. Crowdsource Cleansing	MEDIUM		HIGH	
11. Keep original values		MEDIUM	HIGH	
12. Keep usage logs			HIGH	HIGH
13. Cleanup at source	HIGH			
14. Static vs Dynamic	MEDIUM	LOW	MEDIUM	
15. DQ Dimension			HIGH	HIGH
16. Generate Test Data		HIGH	HIGH	HIGH
17. Use the Data			HIGH	HIGH
18. Push Transforms Upstream	HIGH	HIGH		
19. Documentation		MEDIUM	HIGH	HIGH
20. Data Defect Tracking			HIGH	
21. Change the Culture	HIGH	HIGH	HIGH	HIGH