The Dirty Little Secrets of Big Data

Ask most data scientists about their priorities at work and they'll often confess more than half of their 40 hour work week is spent cleaning data. Why? It's the dirty little secrets you hardly ever hear about in the world of Big Data. Below are just a handful of the obvious problematic culprits:

- Data gaps
- Data cleaning
- Antiquated data warehousing
- Disparate data curation sources
- Expensive data integration challenges

The problem with the above phenomenon is that they're fundamentally done backwards. The proof is everywhere. For example, consider the challenges big pharma is having with data management. Forbes contributor, David Shaywitz, published an article about the struggles pharmaceutical companies are experiencing as they attempt to piece together their legacy data.

Shaywitz reports that one of the challenges for big pharma giants is the cleaning of their data for integration into Machine Learning (ML). He references a study done by Google AI to diagnose patient chest CT scans for lung cancer compared to radiologist assessments. The findings suggest their algorithms are better at detecting the presence of cancer in patients than human diagnosis. But the monumental hurdle was engineering software enhancements to source the data.

Data Deluge

While there's many innovative approaches working with large volumes of business data, the role of a traditional data warehousing is still very pertinent. Especially when it comes to data that must be precise.

Consider the case of financial institution reporting. They provide customers with reports pulled from their databases. Consumers demand these reports offer 100% accuracy on every line item. So what's required to provide this level of critical reporting?

For starters, the data warehouse must be built with precision as the most important critical factor. This is a tall order for any data scientist. Especially for teams building a data warehouse by sourcing many disparate data sources (i.e. legacy databases, transactional information, social media platforms, phone interviews, and internet activity). Yet, this workflow is the most common use case for data warehousing projects.

Data Integration

It’s no surprise that traditional data warehousing projects are almost always over budget. More than half of them are complete failures. One of the common themes among these data warehousing projects is that they often run into challenges resulting from difficulty integrating data from disparate sources. In most cases, hindsight tells us that all of the challenges these teams encounter are predictable.

Below is just one workflow of cleaning and integrating data for analysis:

1. Connect to the raw database and download the CSV file.
2. Use a Python script as a means of cleaning the data.
3. Open the new data set in a new Excel tab.
4. If things look functional, upload the cleansed data into a GUI.
5. Use an SQL script to merge the table data and download.
6. Next, upload that CSV file to a MySQL database.
7. Process another SQL instance to further clean the data.
8. Import that data into RStudio for debugging.
9. Finally, process the data for analysis in R.

Here’s another workflow with a deeper dive into a financial analysis study:

1. A business sponsor identifies a business need to create a client reporting system.
2. The sponsor works with business analysts to identify what should be in the reports.
3. The project is formally identified and an IT team is invited to collaborate.
4. The IT team studies the results of the business analysis and creates the schema for a data warehouse that will meet the business requirements. It will possess tables to represent clients, their accounts, their holdings, the portfolios their investments belong in, trade history, etc.
5. The combined team go shopping for the necessary data pieces. The result of this shopping spree is an inventory (data dictionary) of the data pieces from various sources within the firm. Data will likely come from the CRM system, the accounting system, the trading system, the general ledger system, and various spreadsheets. At this point, there’s only subjective assessments about the quality of the data. Very little is known about how well the data will fit together to populate the desired data warehouse.
6. Next, the IT team puts together a team of ETL developers to work on the project. At some point, weeks and perhaps months into the effort, data starts to come together in the target data warehouse.
7. Soon it becomes evident there’s missing data, poor data matches, and formatting issues. But because there has been so much invested in the project already, and because the end business goal is considered a requirement, the tendency is to somehow make the data fit, making a bad situation worse.

8. Over time, errors compound while the IT team continues their development effort.
9. Eventually, the effort has suffered so much re-work and error-patching that it's either taken much longer than originally planned and/or the project is deemed a failure and scrapped.

What Went Wrong?

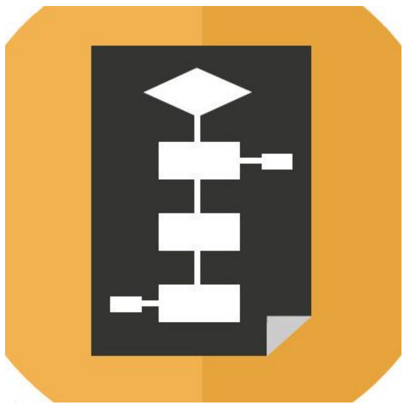
The lack of predictability. Like airliner crashes or train wrecks, there's usually multiple factors that contribute to the calamity. Risk and challenges are often manageable if measures are taken early on. However, the traditional data warehousing approach makes the early identification of risk nearly impossible to detect.

Why?

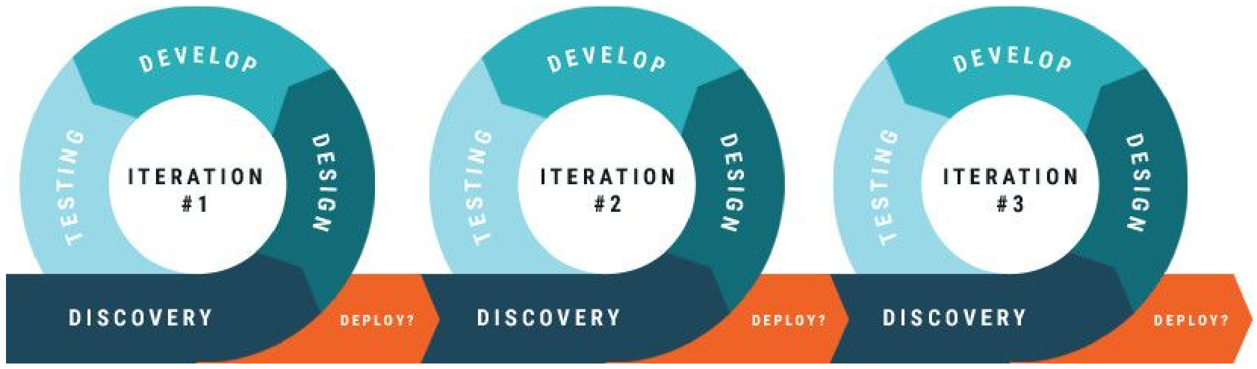
You can't discover what's wrong with the data and its sources until you try to integrate it. That integration effort involves an army of ETL developers to try to put the data together to meet the expectation requirements that were created without the full knowledge of the available data. If you're wondering if this sounds backwards, you're already way ahead of the learning curve.

Data Warehousing the Right Way

Learning from the mistakes of the above example, you could conceive of doing the same client reporting data warehouse project the right way. It starts like the above example until you get to #6. Then things become more logical.



6. The IT team takes the data dictionary and uses a novel new data integration/modeling tool to create an optimally integrated schema. One that combines all of the data they've inventoried. The resulting schema will have taken into account all of schema/data matches, mismatches, and gaps. It represents the best possible result given the available data. Let's call this the Optimal Candidate Schema (OCS).
7. The OCS is then compared to the desired data warehouse schema, using the new tool that was used to create the OCS. The result is a comprehensive gap analysis that effectively measures what they have against what the project team needs. Many of the data issues are known before ETL development begins, allowing upfront risk mitigation in the overall project.
8. As the ETL development progresses, the new tool continues to provide predictable integration results. Errors are identified early and solved with mathematical precision.

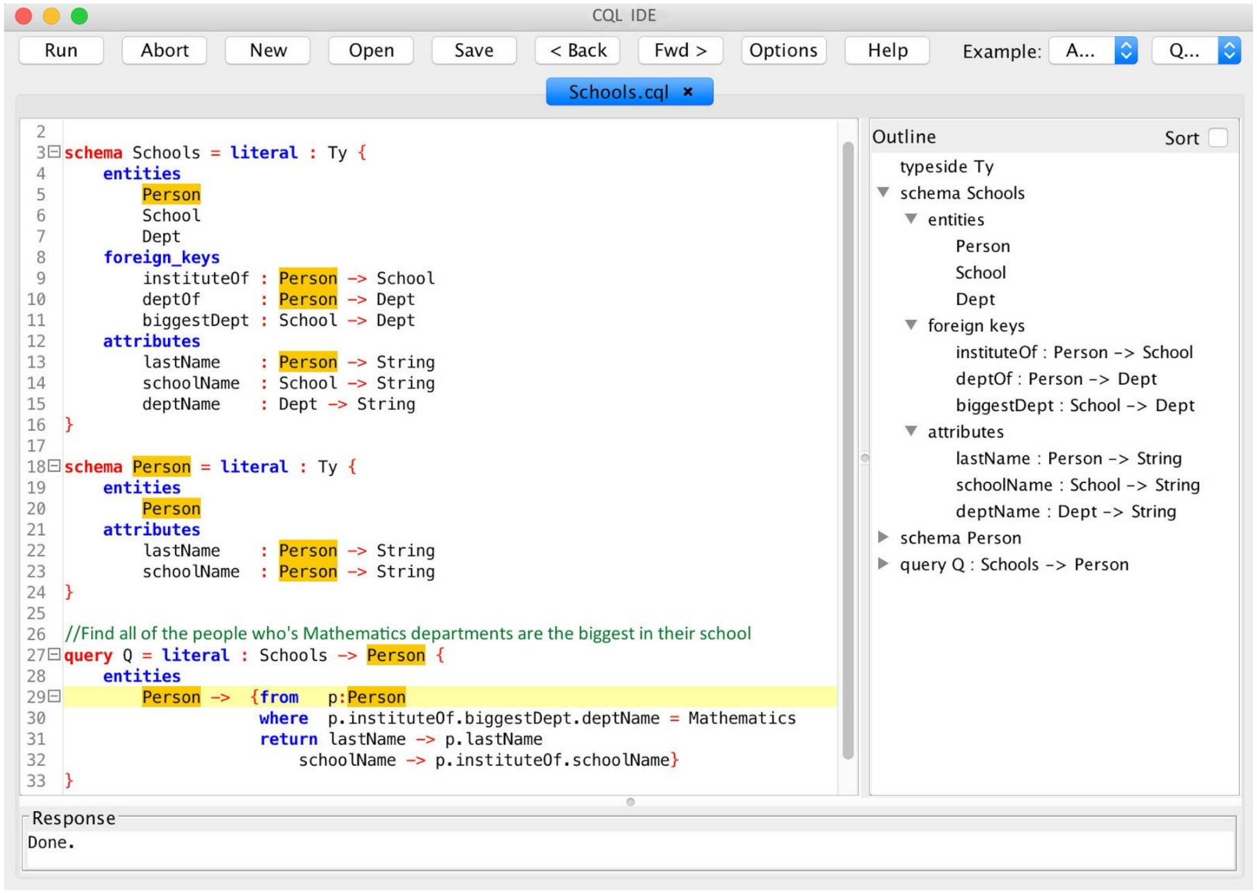


The CQL Way

CQL, based on Category Theory and years of research at MIT, performs data integration with mathematical predictability and precision so that you can execute data warehousing projects confidently by:

- Pinpointing data risks early
- Optimizing data integration mathematically
- Preserving data lineage (where they come from and how they're used)
- Offering the highest data quality possible to meet the needs of the project

At the core of CQL is a theorem prover that provides the mathematical rigor. On top of this engine is a functional programming language with a dedicated IDE that interactively enforces correct data integration.



```

schema Schools = literal : Ty {
    entities
        Person
        School
        Dept
    foreign_keys
        instituteOf : Person -> School
        deptOf      : Person -> Dept
        biggestDept : School -> Dept
    attributes
        lastName    : Person -> String
        schoolName  : School -> String
        deptName    : Dept -> String
}

schema Person = literal : Ty {
    entities
        Person
    attributes
        lastName    : Person -> String
        schoolName  : Person -> String
}

//Find all of the people who's Mathematics departments are the biggest in their school
query Q = literal : Schools -> Person {
    entities
        Person ->  {from   p:Person
                    where  p.instituteOf.biggestDept.deptName = Mathematics
                    return lastName -> p.lastName
                        schoolName -> p.instituteOf.schoolName}
}
```

```
Outline                                Sort
  typeside Ty
▼ schema Schools
    ▼ entities
        Person
        School
        Dept
    ▼ foreign keys
        instituteOf : Person -> School
        deptOf : Person -> Dept
        biggestDept : School -> Dept
    ▼ attributes
        lastName : Person -> String
        schoolName : School -> String
        deptName : Dept -> String
▶ schema Person
▶ query Q : Schools -> Person
```

```
Response
Done.
```

CQL allows a data warehousing project to proceed in a more natural and deterministic manner:

1. We start with a high quality model of available data.
2. Then produce a mathematically optimized integrated model.
3. Next, we compare the data values to the desired product.
4. CQL knows what's possible and what issues exist.
5. Lastly, we execute the data integration tasks with confidence.

The Client Reporting Data Warehouse with CQL

Let's use the financial client reporting data warehouse example and see what the steps are with CQL:

1. Define/import the source schemas. This is akin to defining the structure of the input databases in an ETL process.
2. Define/import the target (i.e. a star schema).
3. Create mappings between the source schemas.
4. Create data mappings between the source databases consistent with the schema mapping in step #3.

5. Migrate the source data into a unified quotient schema.
6. Migrate the result of step #5 onto the target schema using a CQL query.

The Data Sources

Let's also take a closer look at the example source databases for the client reporting project. The following databases make up the example:

- Client: a database with a reference list of clients.
- Reference: a database with reference lists of countries, currencies, investment strategies, and investable assets.
- Portfolio: a database with a reference list of client investment portfolios; includes a copy of the strategies as found in the referenced database above.
- Transaction: a database with records of investment trade transactions.
- Holding and Position: a database with records of holdings and positions of the portfolios.

We have included some features in the example databases in order to make the use case more realistic:

- Portfolios can be hierarchical via an acyclicforeign key relationship from portfolio to itself.
- The portfolio database contains a copy of the strategy table. This is a common practice in organizations that have disparate systems whose reference data must be coordinated.
- Likewise, the transaction database contains replicas of the asset and currency tables, and the holding-and-position databases contains a copy of the client table. The column names differ slightly from the copy in the client database.
- The holding and position tables use currency code values instead of currency code IDs.

As we go through the CQL-based data integration, we will examine how these features, as well as the basic characteristics of the source databases, are managed by CQL.

The Optimized Candidate Schema

Below is the optimized candidate (interim) schema that would result from the CQL integration process:

Features to note in this schema include:

- Replicas of tables such as client, asset, strategy, and currency are eliminated.
- Mismatched column names for the client table are resolved.
- A new junction table, portfolio-holding, is introduced.
- FK relationships that were implied among the source databases are formalized.

About Conexus

The Conexus data integration experts can rescue failing ETL projects by replacing inadequate ETL technology with CQL, a technology which requires people to make fewer design designs and uses AI to prevent common errors.

Discover the power of Conexus. Get in touch...