Sparksoft

“To **ignite** innovation, **inspire** transformation, and **implement** digital solutions for a healthier nation.”

Building configuration-controlled data harmonization service building blocks using commodity code.

Presentation to CDC
March 28, 2023

Test **Automation**

Data **Science**

DevSecOps **Delivery**

Contact Center **Operations**

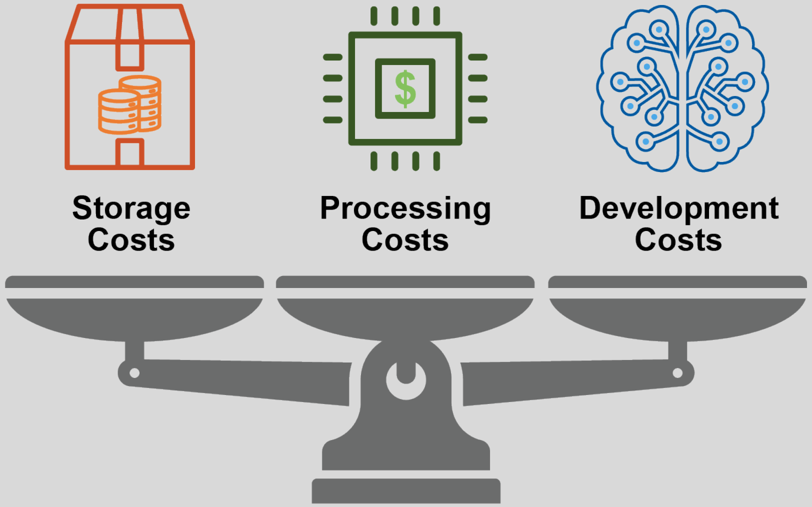
Meet the Speaker

Sparksoft Corporation is a CMMI and ISO certified 8(a), Women-Owned Small Business (WOSB) with a proven record of successfully implementing modern digital health solutions for more than a dozen Federal, State, and commercial customers since 2004.

DIRK LIESKE (Sparksoft)
Director of SparkLabs & Sr. Solutions Architect

Mr. Lieske possesses over 20 years of experience in designing and building some of the world's largest analytic platforms, including an extensive and varied 12-year commitment to CMS Integrated Data Warehouse where he acted as program manager and solution architect. He is an industry expert for his knowledge of Medicare reimbursement principles and has fostered trust and rapport within the CMS community by understanding driving factors and pressures. He has accrued more than 14 years of experience managing all aspects of the SDLC with a specialization in fraud analytic system architecture and administration.

Design Considerations, Assumptions and Tradeoffs



There is no such thing as a one-size fits all design pattern:

For example:

- Some systems could need continual, instant real-time updates where every second counts.
- Some systems could need to process and store petabytes of data, systems where storage costs far exceed human costs.

Configuration-controlled building blocks typically:

- Result in some data redundancy (Raw plus harmonized)
- Require additional compute and processing (Building blocks often run tasks independently)
- Result in physical data model design tradeoffs (Share traits of schema-on-read design patterns)

Defining a Building Block

Extract

Extract and verify data from source systems

Load

Publishes transformed data to repository

Transform

Organizes and harmonizes data to make it usable

1. Always capture what is received
   - Use filters can protect end users
   - Archive after the fact
2. Record all the available metadata
   - Dates, Sizes, Users, Processes Etc.
3. Manufacture keys to link everything together
   - Timestamps work well

Decomposing ETL into Building Blocks:

- Design building blocks users understand
- Identify common repeating processes
- Identify and avoid complex one-off logic
- Leverage what is known (Metadata) or what can be discovered
  - Data Layouts (Headers, Tags)
  - Layout Changes
  - Data Deliveries (File Receipt)
  - Existing Structures (Tables)
  - System Status
- Design for changes in delivery
  - Data Layout Changes
  - Delivery Cadence
  - Delivery Volume
- Design for human input
  - Thresholds
  - Tuning
  - Data Element Definitions
  - Constraints

Defining a Building Block

Extract

Extract and verify data from source systems

Load

Publishes transformed data to repository

Transform

Organizes and harmonizes data to make it usable

1. Data layouts (self descriptions) will allow your extraction processes to generate landing data manipulation statements.
2. Successful building blocks simplify development and operations while building trust and understanding.

Pushed / Delivered Data Assets:

- Users can understand a building block called “Collect and store provided data”. Analytic users need to understand and trust your building blocks.
- Work with source systems to ensure some type of self description (headers, Layouts, Tags etc.)

Pulled / Collected Data Assets

- Generate layouts to ensure self description (Headers, Layouts, Tags etc.)

Defining a Building Block

Extract

Extract and verify data from source systems

Load

Publishes transformed data to repository

Transform

Organizes and harmonizes data to make it usable

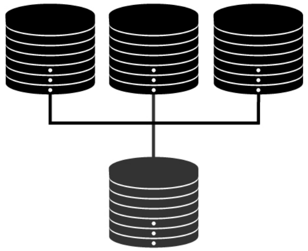
1. Always capture what is received
   - Files deliver text (Not dates and numbers etc.)
2. Layouts change over time
   - Allow targets to grow automatically
3. Capture all available metadata
   - File sizes, row counts, etc.

- Define a single building block for two-dimensional data assets. (Tables, Delimited and Fixed Length Tables etc.)
- Define a second building block to address Recursive and hierarchical data assets. (XML, Cobol etc.)
- Define building blocks to create, maintain and load target structures.
- For hierarchical data assets either:
  - Store data as a CLOB (Character Large Objects)
  - Pre-process/spite data into separate files and then use virtual logic to re-join the data

Defining a Building Block

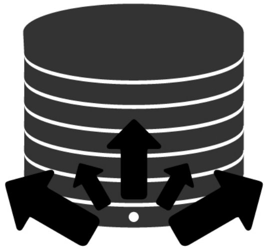
Sparksoft

Extract



Extract and verify data from source systems

Load



Publishes transformed data to repository

Transform

Organizes and harmonizes data to make it usable

Transformation building blocks should be:
Simple to understand
Add columns with improved values
Add columns that can be used as filters

Transformations can resolve many problems:

- Data types
- Duplicate records
- Invalid keys
- Lists of values
- Apply synonyms for standardization
- Contiguous non-overlapping dates
- Flagging quality issues

Transformations can be organized into:

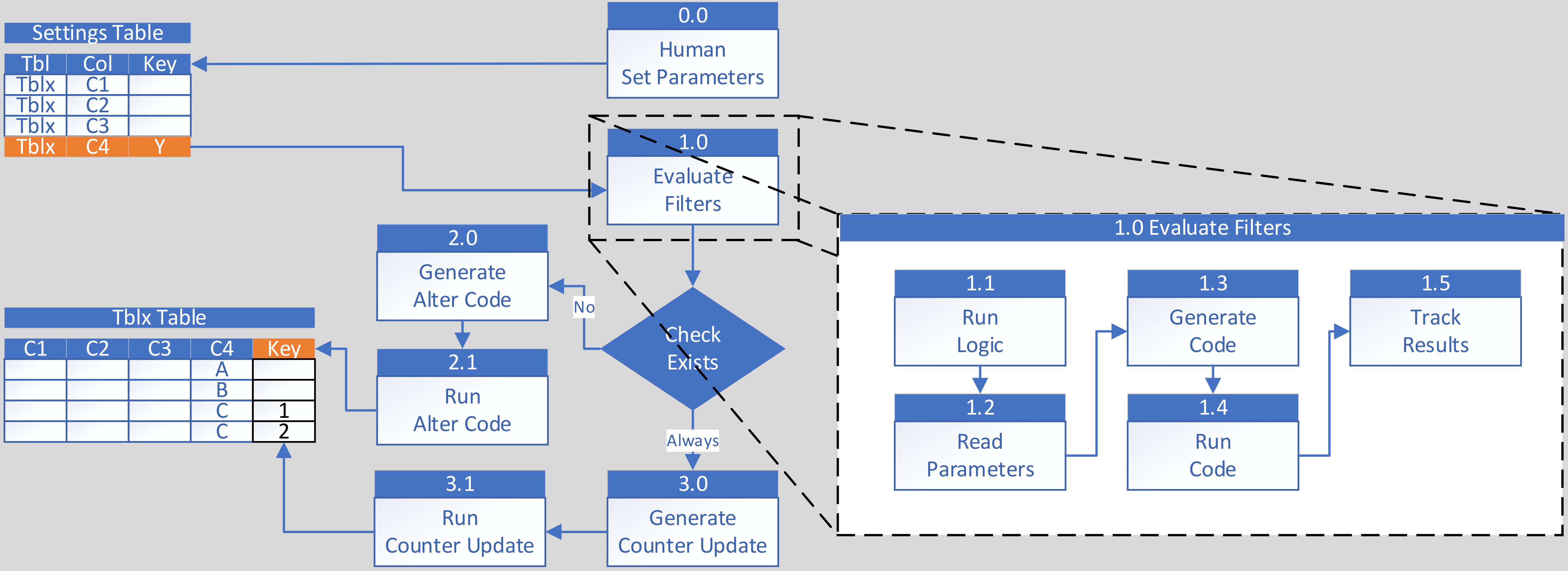
- Field Level Transformations
- Row Level Transformations
- Data set / single table transformations
- Multi-table transformations

Designing a Building Block

A. Decompose the overall pipeline into functional parts. (Load data, Assign Data Types, Identify Duplicates etc.)

B. Further decompose each functional part into specific modules looking for modules that can be further reused.

C. Define common, central repositories to manage configurations, log messages, metadata and errors.



Settings Table

Tbl	Col	Key
Tblx	C1	
Tblx	C2	
Tblx	C3	
Tblx	C4	Y

Tblx Table

C1	C2	C3	C4	Key
			A	
			B	
			C	1
			C	2

Building a Building Block (Example Shown Using SQL)

Step 1: Define Logic Objective (For example logic to convert inbound text to harmonized numbers)

```
UPDATE MY_DATABASE.MY_SCHEMA.MY_TABLE
SET MY_NEW_NUM_CLMN = TRY_TO_NUMBER(REGEXP_REPLACE(MY_ORIG_NUM_CLMN,'[^-0-9]',‘’));
```

Step 2: Identify the variables in your Query: (Typically Database, Tables and Columns)

```
UPDATE MY_DATABASE.MY_SCHEMA.MY_TABLE
SET MY_NEW_NUM_CLMN = TRY_TO_NUMBER(REGEXP_REPLACE(MY_ORIG_NUM_CLMN,'[^-0-9]',‘’));
```

Step 3: Using Metadata select results with the necessary components

```
SELECT c.TABLE_CATALOG
,      c.TABLE_SCHEMA
,      c.TABLE_NAME
,      c.COLUMN_NAME
FROM MY_DATABASE.INFORMATION_SCHEMA.COLUMNS c
WHERE c.TABLE_NAME = ‘MY_TABLE’;
```

Step 4: Build logic around your selected rows

```
SELECT ’UPDATE ’||MY_DATABASE||’.’||MY_SCHEMA||’.’||MY_TABLE||’||
’SET ’||MY_NEW_NUM_CLMN||’ = TRY_TO_NUMBER(REGEXP_REPLACE(’||MY_ORIG_NUM_CLMN||’,'[^-0-9]',‘’));’
FROM MY_DATABASE.INFORMATION_SCHEMA.COLUMNS c
WHERE c.TABLE_NAME = ‘MY_TABLE’;
```

Building a Building Block (Example Shown Using SQL) Cont.

Notes: The simple example generates less than ideal output: (1 Update for each column)

```
UPDATE MY_DATABASE.MY_SCHEMA.MY_TABLE
SET MY_NEW_NUM_CLMN1 = TRY_TO_NUMBER(REGEXP_REPLACE(MY_ORIG_NUM_CLMN1,'[^-0-9]',‘’));
UPDATE MY_DATABASE.MY_SCHEMA.MY_TABLE
SET MY_NEW_NUM_CLMN2 = TRY_TO_NUMBER(REGEXP_REPLACE(MY_ORIG_NUM_CLMN2,'[^-0-9]',‘’));
```

Notes: What you really want is:

```
UPDATE MY_DATABASE.MY_SCHEMA.MY_TABLE
SET MY_NEW_NUM_CLMN1 = TRY_TO_NUMBER(REGEXP_REPLACE(MY_ORIG_NUM_CLMN1,'[^-0-9]',‘’))
,   MY_NEW_NUM_CLMN2 = TRY_TO_NUMBER(REGEXP_REPLACE(MY_ORIG_NUM_CLMN2,'[^-0-9]',‘’))
,   MY_NEW_NUM_CLMN3 = TRY_TO_NUMBER(REGEXP_REPLACE(MY_ORIG_NUM_CLMN3,'[^-0-9]',‘’));
```

Notes: You need to logic for the 1st row and the last row

```
SELECT CASE WHEN ROW_NUMBER() OVER (PARTITION BY c.TABLE_CATALOG
                                                  c.TABLE_SCHEMA
                                          ,       c.TABLE_NAME
                                     ORDER BY
                                                  c.TABLE_CATALOG
                                          ,       c.TABLE_SCHEMA
                                          ,       c.TABLE_NAME
                                          ,       c.COLUMN_NAME) = 1
            THEN ’UPDATE ’||MY_DATABASE||’.’||MY_SCHEMA||’.’||MY_TABLE||’||
                 ’SET ’
            ELSE ’, ’
            END||
            MY_NEW_NUM_CLMN||’ = TRY_TO_NUMBER(REGEXP_REPLACE(’||MY_ORIG_NUM_CLMN||’,'[^-0-9]',‘’))’
        -- DO SOMETHING SIMILAR to add ‘;’ at the end
FROM MY_DATABASE.INFORMATION_SCHEMA.COLUMNS c
WHERE c.TABLE_NAME = ‘MY_TABLE’;
```

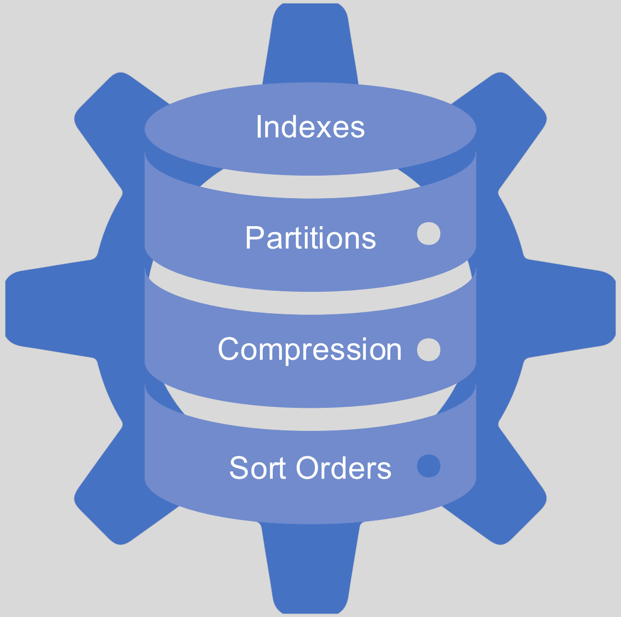
Controlling a Building Block

Use End User Exposed Configuration Tables to Control You Logic:

- Building Blocks are Human Guided (Configurations need to be set)
- Most operations / Building Blocks will be at the dataset level or the field level

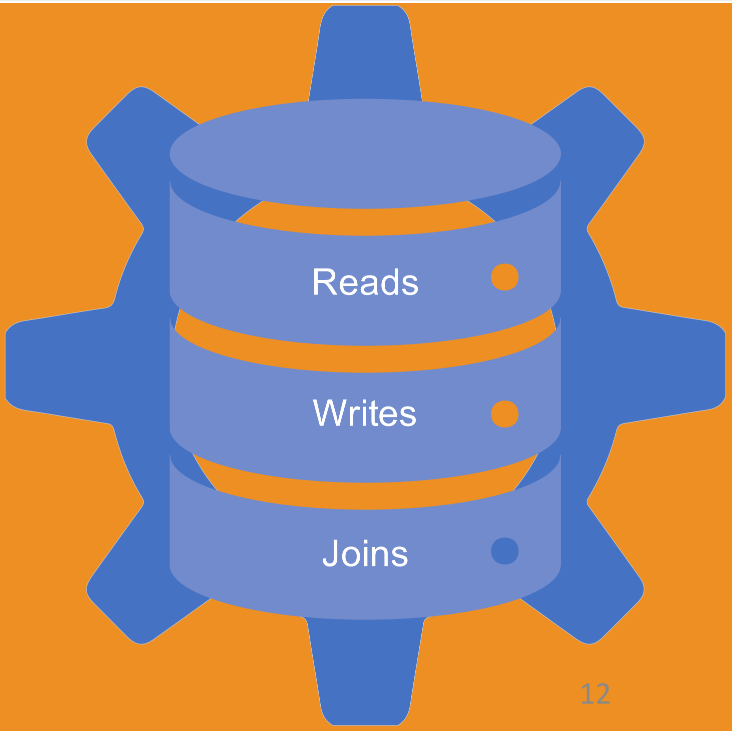
```
CREATE OR REPLACE TABLE MY_DATABASE.MY_SCHEMA.MY_PARM_TBL
(   DB_NAME                       VARCHAR -- The database where the business data is loaded
,   SCMA_NAME                     VARCHAR -- the schema where the business data is loaded
,   TBL_NAME                      VARCHAR -- The original source name typically the file name from the source
,   TBL_DESC                      VARCHAR -- The table description, manually entered
,   CLMN_NAME                     VARCHAR -- The original column name provided by the source
,   CLMN_DESC                     VARCHAR -- The column description, manually entered
,   DATA_TYPE                     VARCHAR -- Defines the desired data type like integer or date etc.
,   PRIME_KEY_FLG                 BOOLEAN -- Each field that is part of the primary key is set to "y"
,   IGNOR_PURE_DUP_FLG            BOOLEAN -- If set yes then the pure duplicate check will not include
,   IN_LIST_NAME                  VARCHAR -- Concatenates a set of fields into a single new field based on list name
,   CLMN_NAME_ALIAS               VARCHAR -- Allows users to rename source field names in view
,   VIEW_NAME_ALIAS               VARCHAR -- A view alias specific to a single table.
,   DATE_BAND_KEY                 VARCHAR -- The key used when banding a specific date
,   DATE_FRMT                     VARCHAR -- Incoming Date Format for specified field
,   TIMESTAMP_FRMT                VARCHAR -- Incoming TIMESTAMP Format for specified field
,   SYN_GRP                       VARCHAR -- The synonym group text is being edited by
,   RULE_LKUP_GRP                 VARCHAR -- The group that is used to perform a lookup and replace
,   QUAL_VALUE_MATCH              VARCHAR -- USED TO TURN ON AND OFF QUALITY MEASURE AND TO SPECIFY VALUE TO MEASURE AGAINST
,   QUAL_VALUE_LIKE               VARCHAR -- USED TO TURN ON AND OFF QUALITY MEASURE AND TO SPECIFY VALUE TO MEASURE AGAINST
,   QUAL_MIN_VAL                  INTEGER -- USED TO TURN ON AND OFF QUALITY MEASURE AND TO SPECIFY VALUE TO MEASURE AGAINST
,   QUAL_MAX_VAL                  INTEGER -- USED TO TURN ON AND OFF QUALITY MEASURE AND TO SPECIFY VALUE TO MEASURE AGAINST
,   QUAL_MIN_STRLEN_VAL           INTEGER -- USED TO TURN ON AND OFF QUALITY MEASURE AND TO SPECIFY VALUE TO MEASURE AGAINST
,   QUAL_MAX_STRLEN_VAL           INTEGER -- USED TO TURN ON AND OFF QUALITY MEASURE AND TO SPECIFY VALUE TO MEASURE AGAINST);
```

Performance Considerations



- Some performance structures need to be established at time of creation (indexes, partitions, sort orders etc.)
- Some systems require lower amounts of performance tuning
- Some systems prefer wide tables (Columnar)

- Each read comes at a cost
- Each write comes at a cost
- Each building block is likely to require a read and a write



Thank You

MISSION:

To ignite innovation, inspire transformation, and implement digital solutions for a healthier nation