Testing a Complex Database Migration

Greg McNelly

Progressive Insurance

The Problem

- A new policy system introduces a new database, which will eventually replace two older databases.
- Over a period of several months, clients of the older databases will be migrated to the new database.
- In the meantime:
  - The new database will be initialized by loading all of the data from the two older databases into it.
  - All subsequent updates to the older databases must also be applied to the new database.
  - Updates to the new database (relatively rare) will not be applied to the older databases.

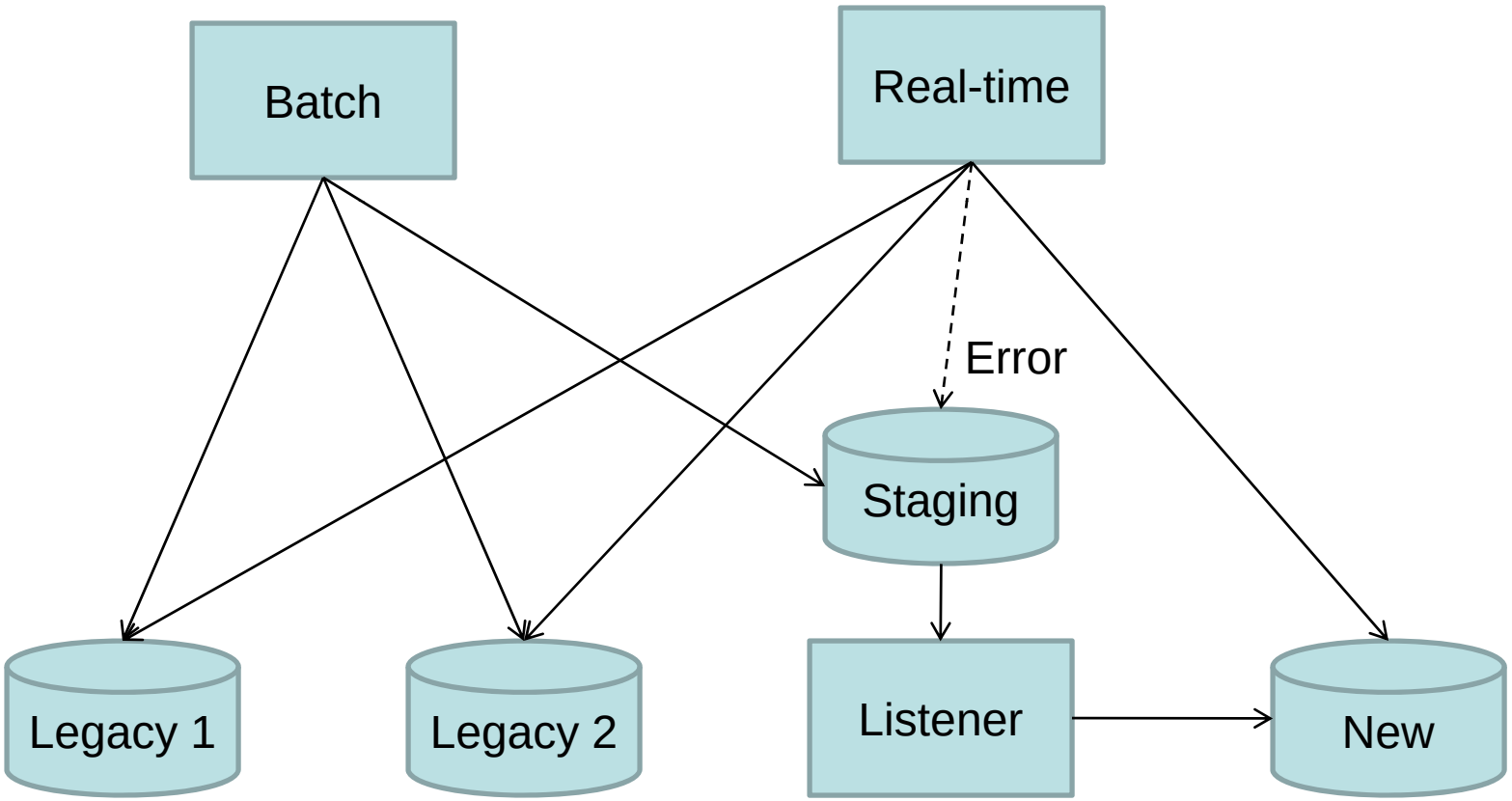
The Problem (cont'd)

- The two older databases:
  - IBM Mainframe DB2
  - Different schemas
  - 1.3 billion records, 24 tables
- The new database:
  - Microsoft SQL Server 2005
  - Schema differs from both older databases

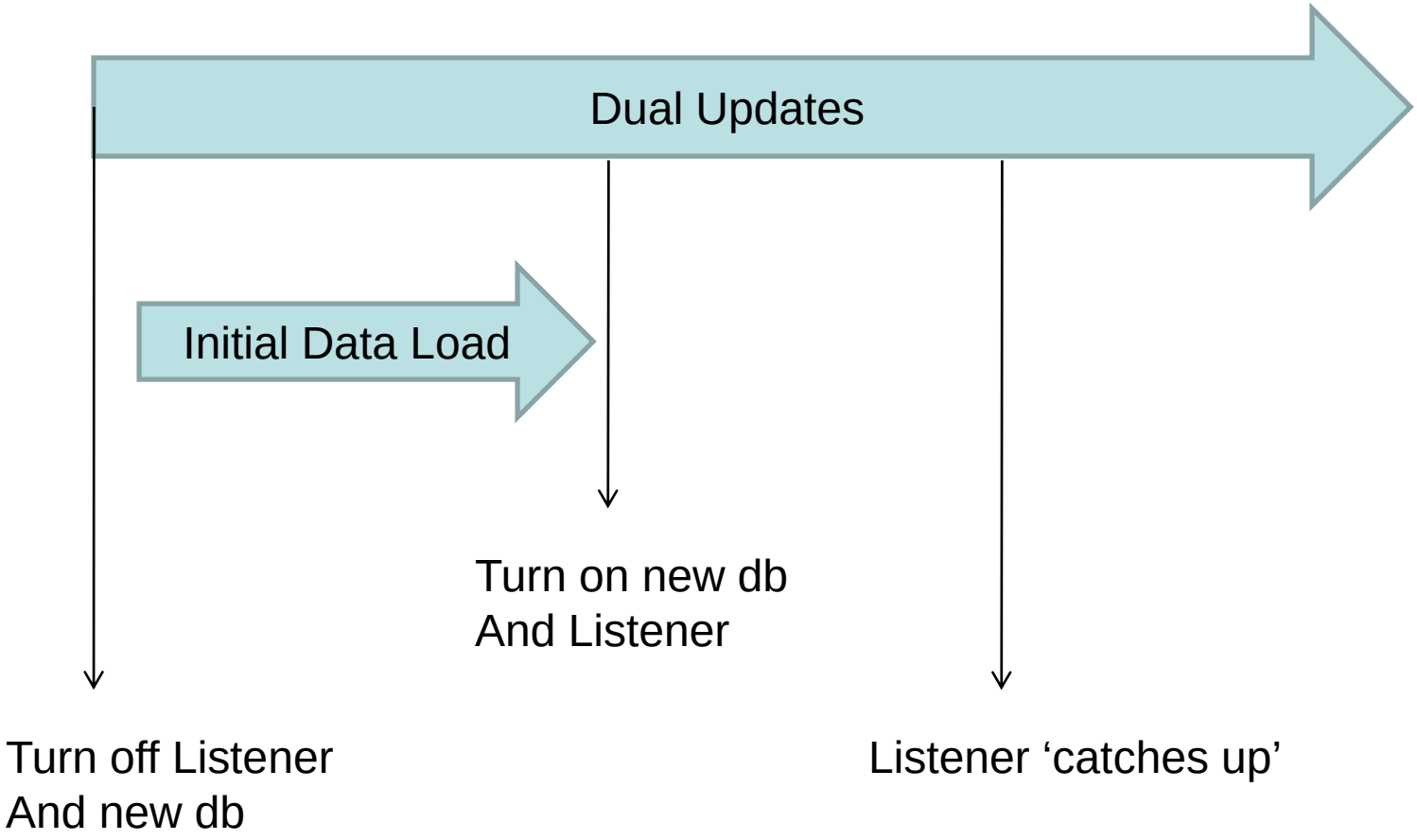
The Problem (cont'd)

- Management tasked the testing team with devising a way to measure "data quality" during the transition period.
  - i.e. compare data in the older databases to data in the new database on a regular basis.

System Design

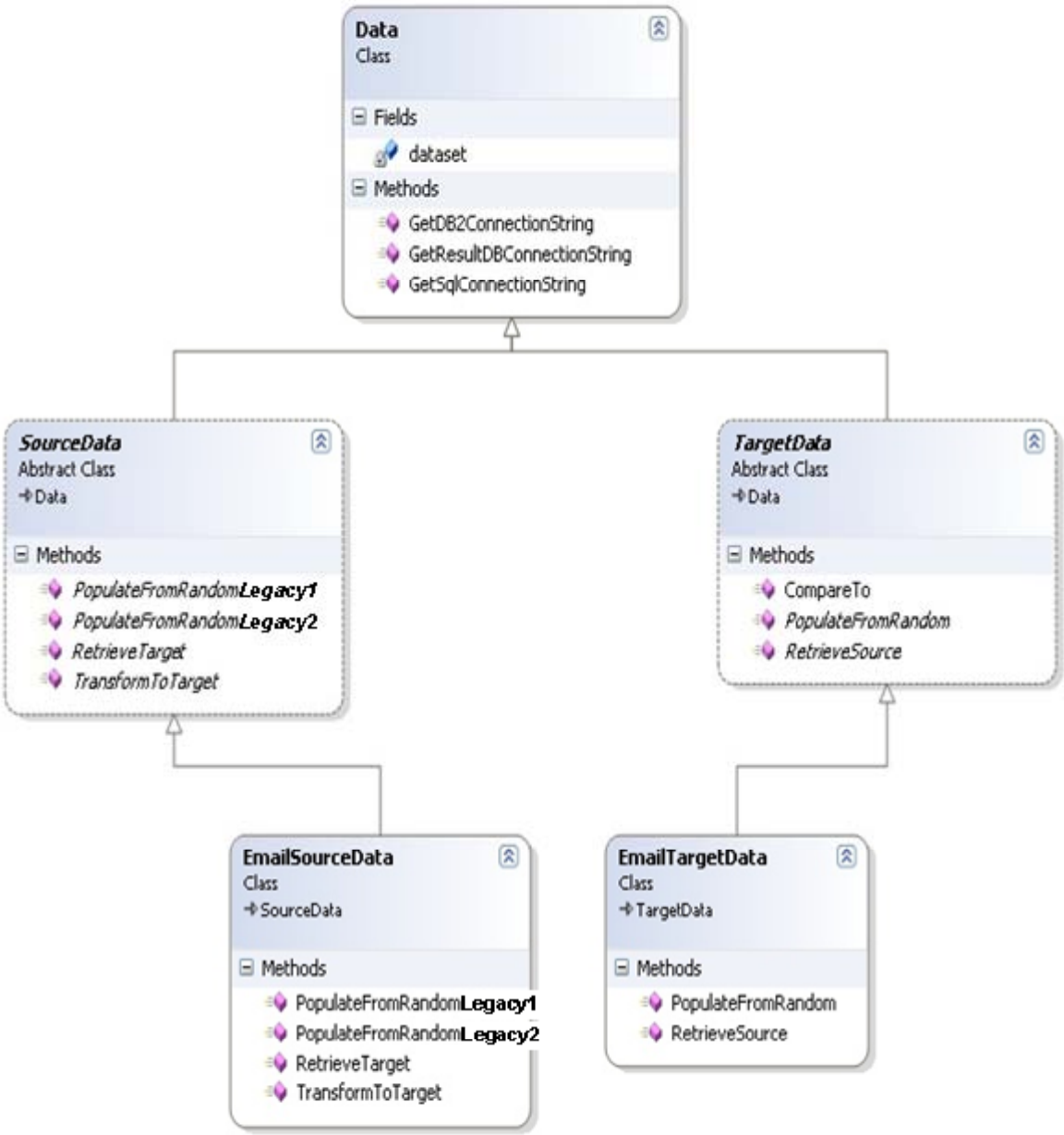


Starting the Process



Key Concepts of the Testing Solution

- SourceData vs. TargetData
  - Composite nature – the two older databases can be treated like one.
- Random sampling
  - Because there is too much data to check in any reasonable amount of time.
- Targeted sampling
  - To help isolate problems related directly to
    - The dual update process
    - Critical data elements
    - Active policies (rather than expired policies)
- Corresponding Retrieval and Comparison
  - Based on the same rules specified for the initial data load and dual updates.
- Categorization of mismatches
  - By field, to prioritize troubleshooting effort


Data
Class
Fields
dataset
Methods
GetDB2ConnectionString
GetResultDBConnectionString
GetSqlConnectionString
SourceData
Abstract Class
Data
Methods
PopulateFromRandomLegacy1
PopulateFromRandomLegacy2
RetrieveTarget
TransformToTarget
TargetData
Abstract Class
Data
Methods
CompareTo
PopulateFromRandom
RetrieveSource
EmailSourceData
Class
SourceData
Methods
PopulateFromRandomLegacy1
PopulateFromRandomLegacy2
RetrieveTarget
TransformToTarget
EmailTargetData
Class
TargetData
Methods
PopulateFromRandom
RetrieveSource


High Level Flow Diagram
Legacy1 (DB2)
Legacy2 (DB2)
NewDB (SQL)
Source Data
Target Data (Actual Result)
Retrieve Source Data From Legacy
Retrieve Target From New DB
Transform Source Data to Target Data based on requirements
Transform Source Data to Target Data based on requirements
Target Data (Actual Result)
Target Data (Expected Result)
Target Data (Expected Result)
Compare Targets (Expected, Actual)
Compare Targets (Expected, Actual)
Log Results
DataCompare (SQL)

Modeling Tactics

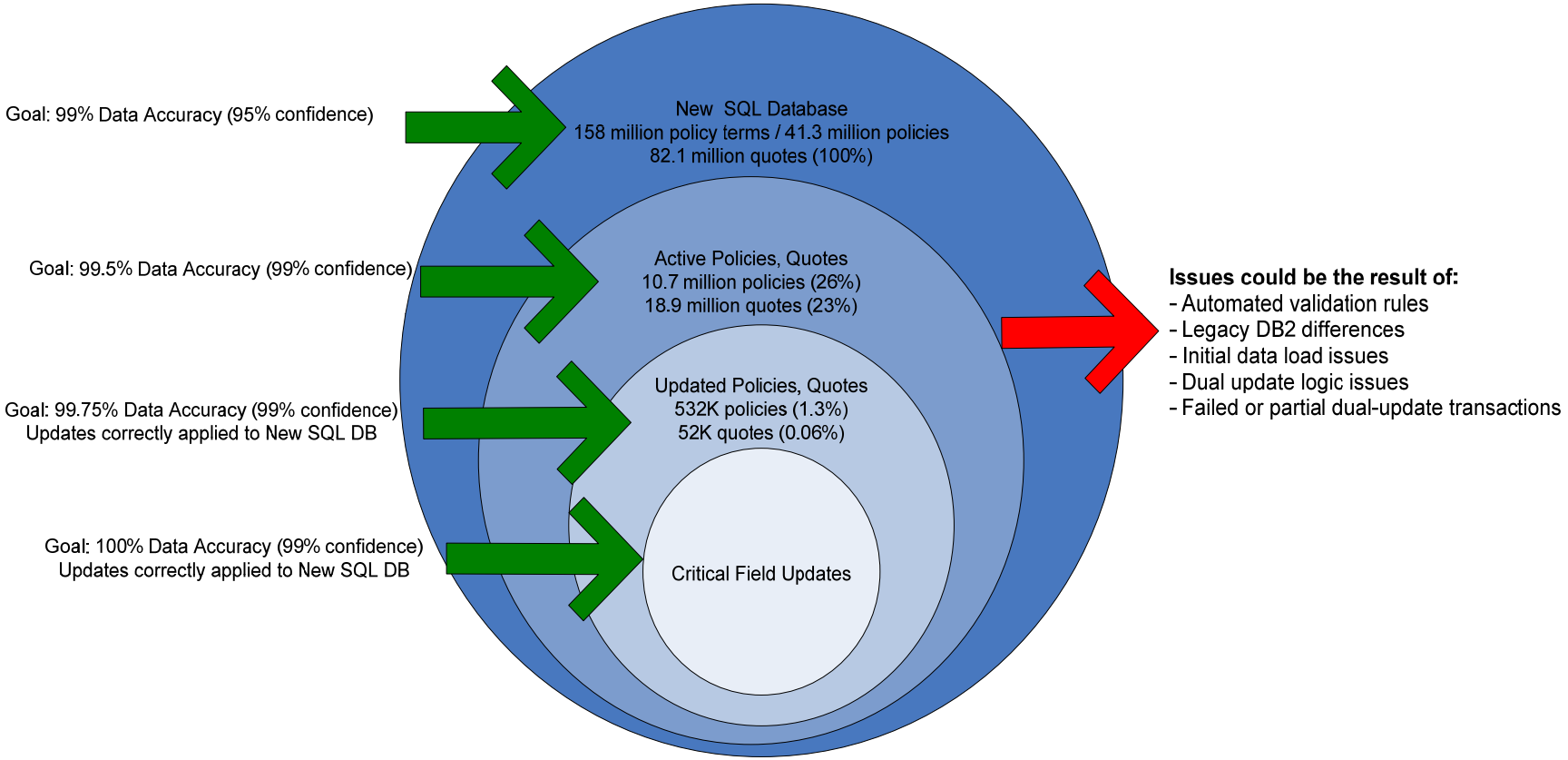
- Shrink
  - Instead of thinking about comparing huge amounts of data, focus on comparing single records.
- Simplify
  - It’s easier to compare 2 things than 3 things.
- Generalize
  - Establish a pattern that can be reused on each set of tables.
- Take Clues from the Design of the SUT
  - Organize the tests along the same lines as the system design so failures are more recognizable to the developers.
- Work as a team
  - All perspectives considered and decisions shared in real time.
- Stay on the white board as long as you can
  - Many bad ideas were eliminated before more significant time and energy could be expended.

Statistical Sampling

$$n = \frac{t^2 \times p(1-p)}{m^2}$$

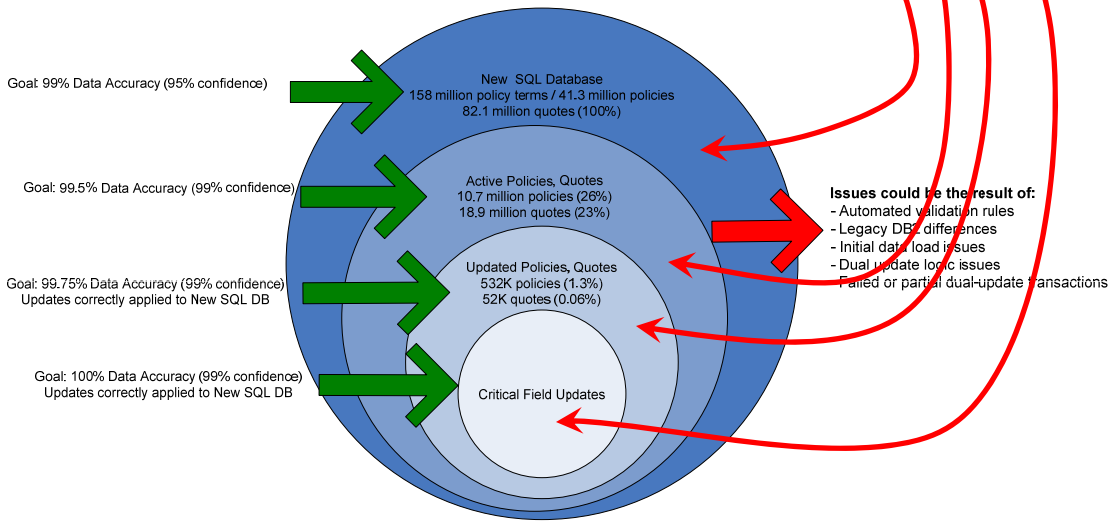
- The Formula
  - Make assumptions (and check them)
- Sample Size
  = n / (1 + ((n-1)/population))
  - Non-linearity required a lot of explaining
- Randomization
  - SQL (too slow)
  - Key Generation (too difficult, 24 different keys)
  - Time Stamp Generation (just right, every table has one)

Goals


Goal: 99% Data Accuracy (95% confidence)
New SQL Database
158 million policy terms / 41.3 million policies
82.1 million quotes (100%)
Goal: 99.5% Data Accuracy (99% confidence)
Active Policies, Quotes
10.7 million policies (26%)
18.9 million quotes (23%)
Goal: 99.75% Data Accuracy (99% confidence)
Updates correctly applied to New SQL DB
Updated Policies, Quotes
532K policies (1.3%)
52K quotes (0.06%)
Goal: 100% Data Accuracy (99% confidence)
Updates correctly applied to New SQL DB
Critical Field Updates
Issues could be the result of:
- Automated validation rules
- Legacy DB2 differences
- Initial data load issues
- Dual update logic issues
- Failed or partial dual-update transactions

The Summary Report

		Results Summary							
	Report	Last Update Date	Approx. Population Size	Sample Size	Pop : Sample Ratio	Failures	Projected Failures in Population	Percent Passing	"Bullseye" Score
	PDV Leg1 Critical - Policies	03/31/2009	467,000	24,405	19	3,496	66,897	85.675%	85.855%
	PDV Leg1 Critical - Quotes		52,000				-		
	PDV Leg2 Critical - Policies	03/31/2009	20,000	4,000	5	522	2,610	86.950%	
	PDV Leg1 - Policies	03/31/2009	467,000	24,405	19	3,565	68,218	85.392%	85.358%
	PDV Leg1 - Quotes		52,000				-		
	PDV Leg2 - Policies	03/31/2009	20,000	4,000	5	594	2,970	85.150%	
	PDV Active Policies	03/30/2009	10,900,000	29,328	372	611	227,083	97.917%	97.917%
	PDV Active Quotes		18,900,000				-		
	PDV Overall	03/31/2009	1,300,000,000	579,262	N/A	1,217	N/A	99.790%	99.790%



Issue Resolution

- Categorization of Tests
- Categorization of Test Failures
- Daily Cycle

Lessons Learned

- Use / reuse of the approach and tools for other testing.
- Avoid 100% goals.
- Don't put yourself in this position. (Eliminate a database.)
- Fact is stranger than fiction – production was the ultimate test.
- Other things considered:
  - Use of SQL Server Integration Services (SSIS)
- Other challenges faced:
  - Security and privacy