Data Analytic Tools for Inconsistency Detection in Large Data Sets

Sdmay18-27

http://sdmay18-27.sd.ece.iastate.edu/

Advisor - Dr. Ying Cai

Client - Kingland Systems

Team Members

- Christopher Konopka
  - Communication Lead
- Logan Heitz
  - Project Lead
- TJ Rogers
  - Quality Lead
- Camden Voigt
  - Technical Lead

System Design

Problem Statement

- Kingland Systems performs inconsistency detection on large data sets
    - An inconsistency arises when two records should match, but don’t
- Current solution takes a lot of time and resources

Functional Requirements

- Solution must not use SQL inner-join statements
- Solution must utilize only relevant information
- Solution must compare current records to previous records as well as other current records
- Solution must update inconsistency database after analysis

Non-functional Requirements

- Solution must perform inconsistency check faster than current solution. Preferably in less than 1 hour
- Solution must be able to check an input of 500 thousand records against 100 million or more records at a time
- Solution must find all inconsistencies from an incoming report

Market Survey

- "An Efficient Method of Data Inconsistency Check for Very Large Relations."
    - Functional dependencies
    - Works well with smaller number of rules and very specific types
- "Inconsistencies in big data"
    - Learning how inconsistencies are caused

Resource Requirements

- Deployment Server
  - AWS
- Database
- Configuration Files
- Raw Data Files

Risks

- Miscommunication with Kingland
- Lack of big data knowledge
- Shortage of time
- Solution not scalable to large data sets

System Overview

- Modular design
- Reduce the data size
- Check the consistency of each record in the report sequentially
- Database table per reporter
- Threading to parallelize workload

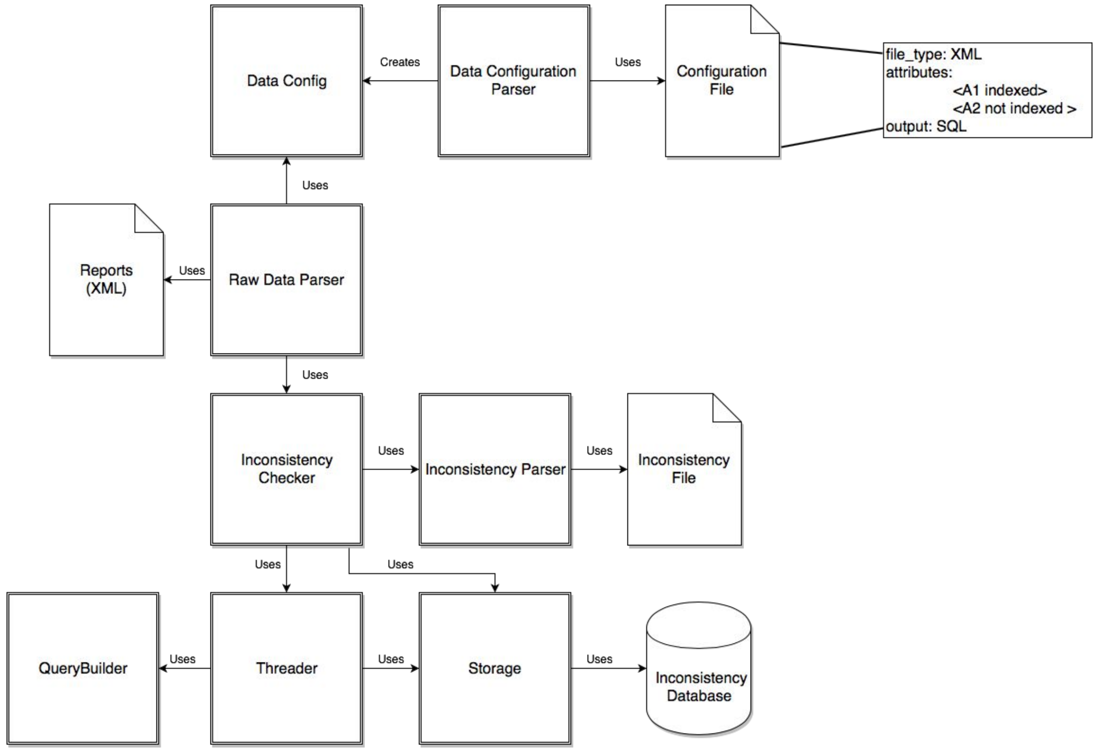
System Analysis

- Strengths
  - Parallelization
  - Solution is modular
  - Easy configuration
- Weaknesses
  - Third Party Solutions

Functional Decomposition

- Data Configuration Parser
- Inconsistency Configuration Parser
- Raw Data Parser
- Inconsistency Checker
- Threader
- Storage

System Block Diagram



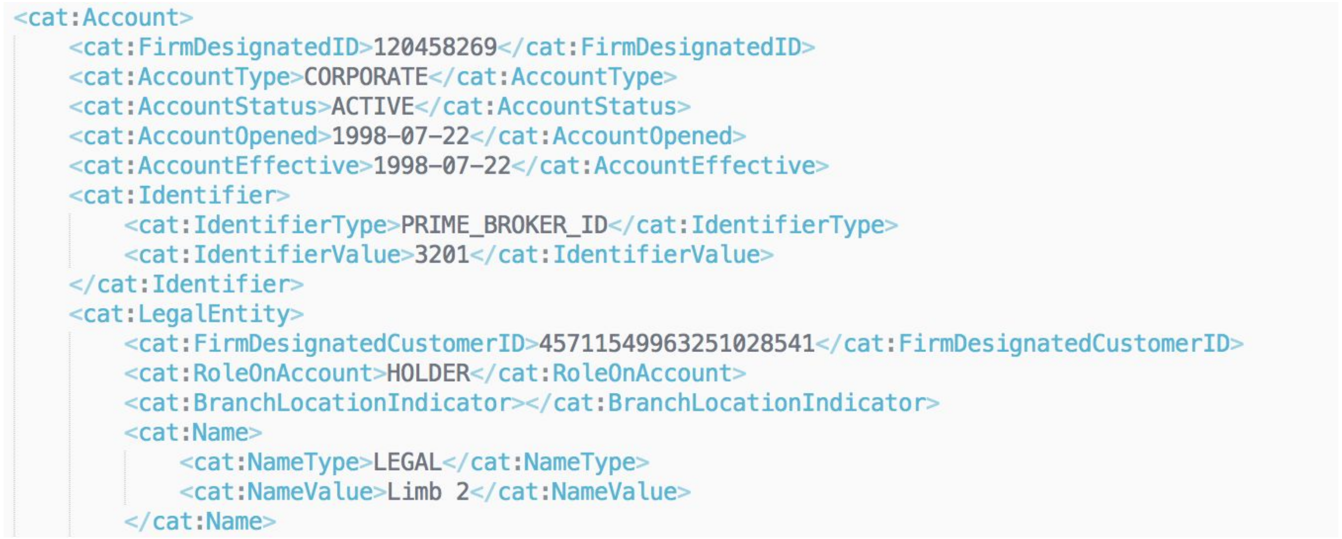
Detailed Design

Detailed Design - Configuration Parsers

- Data Configuration Parser
    - Parses the data configuration file
    - Configures
        - Raw Data Location
        - Storage
        - Raw Data Format
- Inconsistency Configuration Parser
    - Parses the inconsistency configuration file
    - Configures what inconsistency will be checked

Detailed Design - Raw Data Parser

- Reads raw input file and extracts desired elements
    - Parses a given file, or all files in a given directory



```
<cat:Account>
    <cat:FirmDesignatedID>120458269</cat:FirmDesignatedID>
    <cat:AccountType>CORPORATE</cat:AccountType>
    <cat:AccountStatus>ACTIVE</cat:AccountStatus>
    <cat:AccountOpened>1998-07-22</cat:AccountOpened>
    <cat:AccountEffective>1998-07-22</cat:AccountEffective>
    <cat:Identifier>
        <cat:IdentifierType>PRIME_BROKER_ID</cat:IdentifierType>
        <cat:IdentifierValue>3201</cat:IdentifierValue>
    </cat:Identifier>
    <cat:LegalEntity>
        <cat:FirmDesignatedCustomerID>4571154996325102854l</cat:FirmDesignatedCustomerID>
        <cat:RoleOnAccount>HOLDER</cat:RoleOnAccount>
        <cat:BranchLocationIndicator></cat:BranchLocationIndicator>
        <cat:Name>
            <cat:NameType>LEGAL</cat:NameType>
            <cat:NameValue>Limb 2</cat:NameValue>
        </cat:Name>
```

Detailed Design - Inconsistency Checker

- Called each time a record is parsed from the raw data
- Record is first added to Storage
- Creates threads to perform inconsistency queries
- Sends found inconsistencies to Storage

Detailed Design - Threader

- Provides interface to run threads
- Uses a fixed thread pool
- Improves performance of project

Detailed Design - Storage

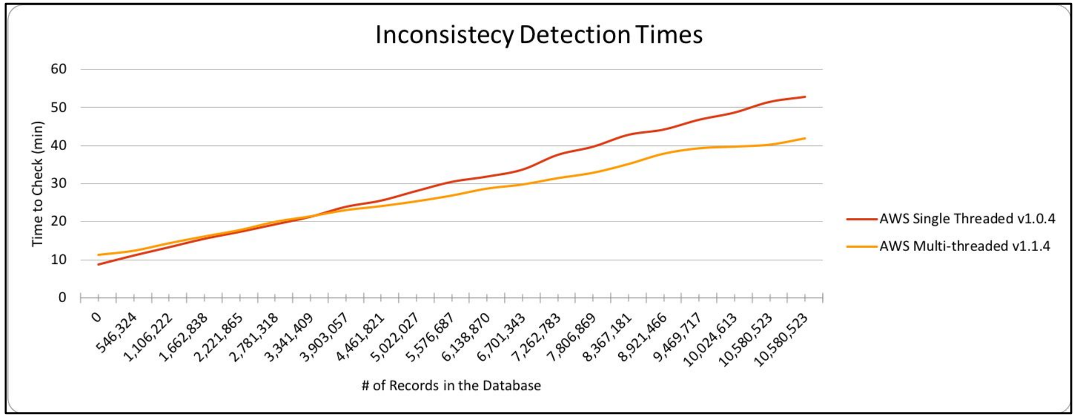
- Query Database to find inconsistencies
- Exports records & inconsistencies into database
- Connection Pooling for better performance
- Extendable to additional database types

Test Plan

- Test Driven Development
- Testing Tools
    - JUnit
    - Mockito
- Defect Reports as Issues on GitLab
- Continuous Integration

Simulation and Testing

- Integration Testing
  - Local Machine
  - AWS Instance
- Performance Testing
  - AWS Instance
  - Large Test Data Sets



I/O & UI Design

- Inputs
  - Data Configuration File
  - Inconsistency Configuration File
  - Raw Data File(s)
  - Command Line Options
- Output
  - Inconsistency Database
  - Log File
- User Interface
  - Not needed

Conclusion

Current Status

- Solution fully implemented
- Deployed and tested on AWS instance
- User manual created

Lessons Learned

- Leave more time for integration/performance testing
- Even though it's far away, figure out deployment early
- Communicating with companies takes time

Future Work

- Database optimizations for deployment
- Integrate into Kingland’s system

Questions?