CDAP Wrangler Enhancement Assignment

CDAP Wrangler Enhancement Assignment

1. Fork & Setup
- Fork the repo: https://github.com/data-integrations/wrangler
- Clone locally and navigate to the directory.

2. Grammar Modification (Directives.g4)
- Add lexer rules for BYTE_UNIT and TIME_UNIT.
- Add parser rules: byteSizeArg, timeDurationArg.
- Regenerate parser using `mvn clean compile`.

3. API Updates (wrangler-api)
- Create ByteSize.java and TimeDuration.java in `api/parser`.
- Include logic to convert sizes to bytes and durations to milliseconds.
- Update TokenType.java to include BYTE_SIZE and TIME_DURATION.

4. Core Parser Update (wrangler-core)
- Update visitor methods to handle new token types.
- Return ByteSize and TimeDuration instances.

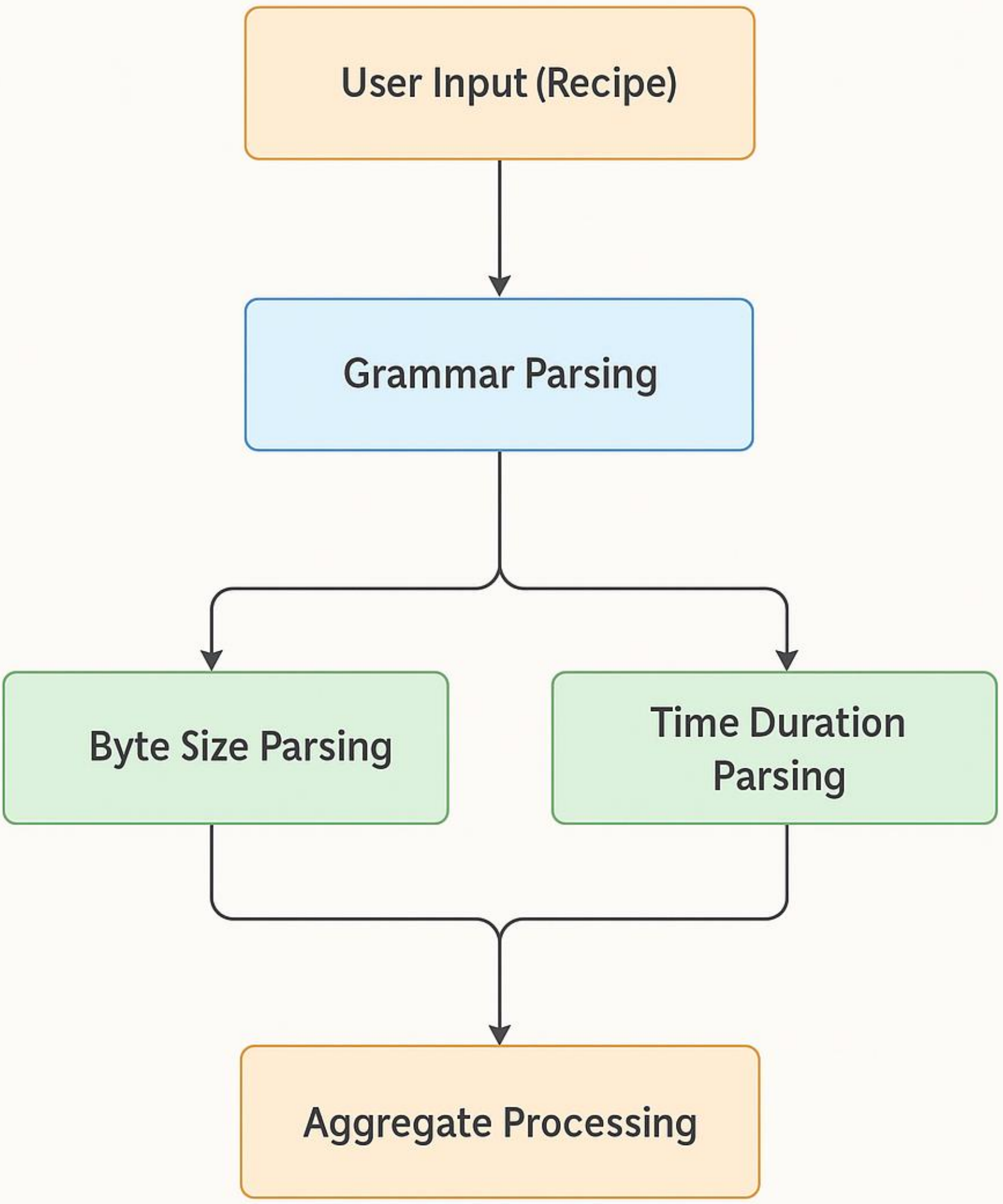
5. AggregateStats Directive
- Create new directive that reads two columns, parses their sizes/durations, and aggregates them.
- Outputs total in MB and seconds.

6. Testing
- Write unit tests for ByteSize and TimeDuration parsing.
- Write integration test for AggregateStats directive.

7. Prompts.txt
- Include prompts for AI-assisted code generation or debugging.

8. README.md
- Add usage documentation and supported units for new parsers.

Enhance Wrangler with Byte Size and Time Duration Units Parsers



CDAP Wrangler Enhancement Assignment

Code Snippets

```
// ByteSize.java (example snippet)
public class ByteSize extends Token {
    private long bytes;
    public ByteSize(String value) {
        if (value.endsWith("KB")) bytes = (long)(Double.parseDouble(value.replace("KB", "")) * 1024);
        else if (value.endsWith("MB")) bytes = (long)(Double.parseDouble(value.replace("MB", "")) * 1024 * 1024);
        else if (value.endsWith("GB")) bytes = (long)(Double.parseDouble(value.replace("GB", "")) * 1024 * 1024 * 1024);
        // Add more parsing logic...
    }
    public long getBytes() { return bytes; }
}

// TimeDuration.java (example snippet)
public class TimeDuration extends Token {
    private long milliseconds;
    public TimeDuration(String value) {
        if (value.endsWith("ms")) milliseconds = Long.parseLong(value.replace("ms", ""));
        else if (value.endsWith("s")) milliseconds = Long.parseLong(value.replace("s", "")) * 1000;
        // Add more parsing logic...
    }
    public long getMilliseconds() { return milliseconds; }
}

// AggregateStats.java (simplified)
public class AggregateStats implements Directive {
    private String byteCol, timeCol, sizeOutCol, timeOutCol;
    public List<Row> execute(List<Row> rows, ExecutorContext ctx) {
        long totalBytes = 0, totalTime = 0;
        for (Row row : rows) {
            totalBytes += ((ByteSize) row.getValue(byteCol)).getBytes();
            totalTime += ((TimeDuration) row.getValue(timeCol)).getMilliseconds();
        }
        List<Row> output = new ArrayList<>();
        output.add(new Row().add(sizeOutCol, totalBytes / (1024.0 * 1024))
                            .add(timeOutCol, totalTime / 1000.0));
        return output;
    }
}
```

Test Cases

```
@Test
public void testByteSizeParsing() {
    ByteSize size = new ByteSize("10KB");
```

```
    assertEquals(10240, size.getBytes());
    size = new ByteSize("1.5MB");
    assertEquals(1572864, size.getBytes());
}

@Test
public void testTimeDurationParsing() {
    TimeDuration time = new TimeDuration("200ms");
    assertEquals(200, time.getMilliseconds());
    time = new TimeDuration("2s");
    assertEquals(2000, time.getMilliseconds());
}

@Test
public void testAggregateStatsDirective() {
    List<Row> rows = Arrays.asList(
        new Row().add("data_transfer_size", new ByteSize("2MB")).add("response_time", new TimeDuration("1s")),
        new Row().add("data_transfer_size", new ByteSize("3MB")).add("response_time", new TimeDuration("2s"))
    );
    List<Row> result = new AggregateStats().execute(rows, null);
    assertEquals(1, result.size());
    assertEquals(5.0, result.get(0).getValue("total_size_mb"), 0.001);
    assertEquals(3.0, result.get(0).getValue("total_time_sec"), 0.001);
}
```