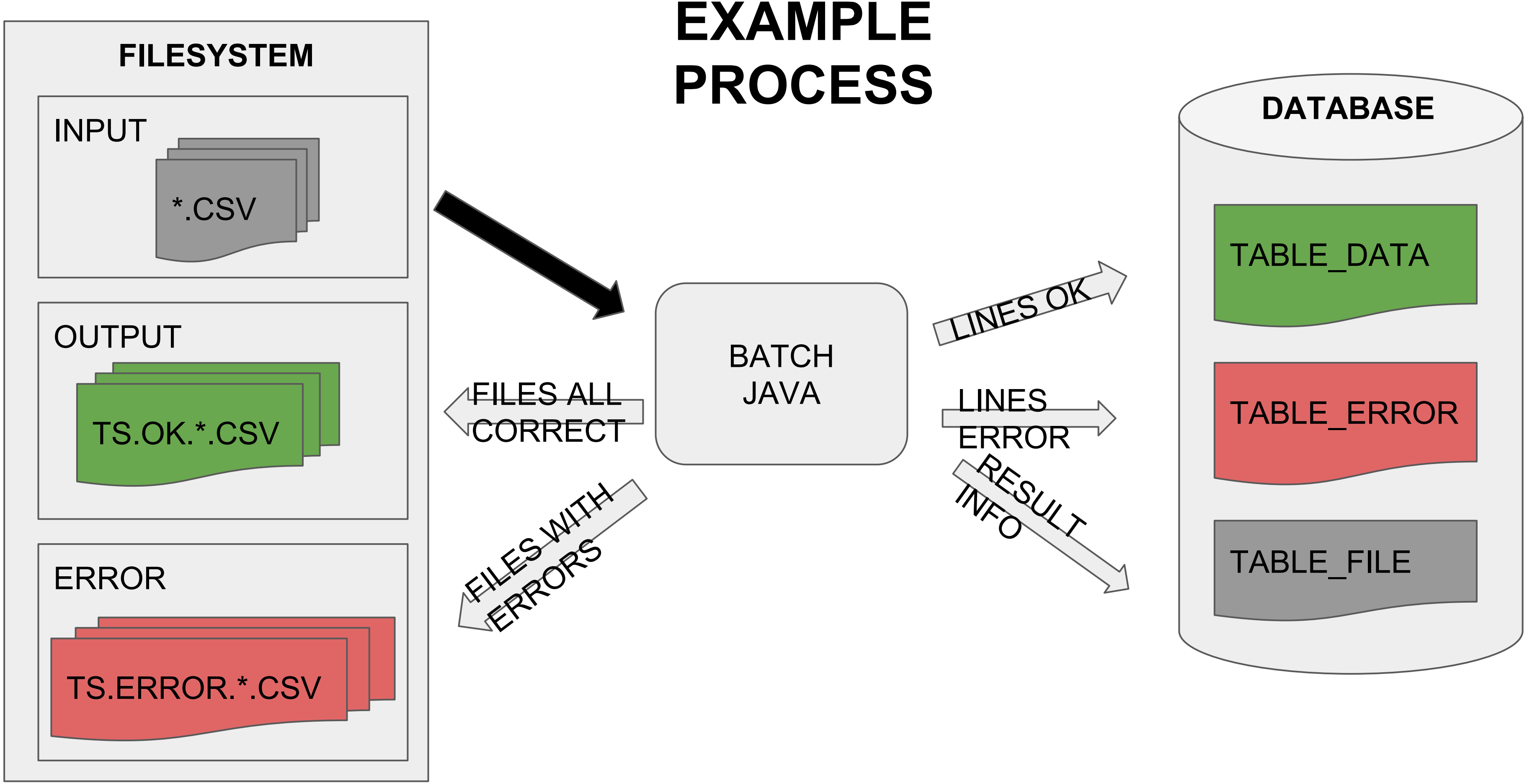
EXAMPLE PROCESS



FORMAT FILES *.CSV ~ TABLE_DATA

COLUMN_INTEGER mandatory, unique, integer	**COLUMN_STRING** optional, max 10 ch	**COLUMN_DOUBLE** optional, decimal	**COLUMN_DATE** mandatory, HH:mm:ss dd/MM/yyyy
1	AUDI	4	12:34:15 08/02/2017
2	ASTON MARTIN	1.479XXX	19:08:15 08/07/2013
3	BMW	325	11:05:54 05/02/2001
3	SKODA	1.2	06:51:09 01/01/1987
4	FIAT	500	
5	MERCEDES		01:01:16 30/02/1995

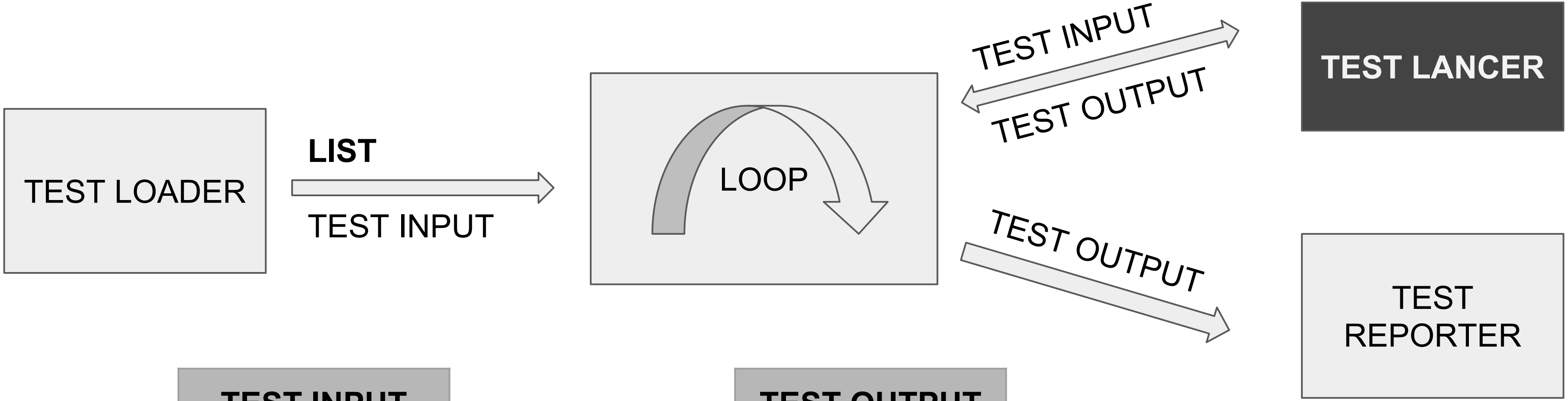
DATABASE MODEL

TABLE_DATA	
ID	NUMBER
DATE	TIMESTAMP
BATCH_ID	VARCHAR2
COLUMN_INTEGER	NUMBER
COLUMN_STRING	VARCHAR2
COLUMN_DOUBLE	NUMBER
COLUMN_DATE	TIMESTAMP

TABLE_ERROR	
ID	NUMBER
DATE	TIMESTAMP
BATCH_ID	VARCHAR2
FILE	VARCHAR2
LINE	NUMBER
ITEM	VARCHAR2
VALUE	VARCHAR2
REASON	VARCHAR2

TABLE_FILE	
ID	NUMBER
DATE	TIMESTAMP
BATCH_ID	VARCHAR2
FILE	VARCHAR2
COUNT_LINES	NUMBER
COUNT_LINES_ERROR	VARCHAR2

KTM



TEST LOADER

LIST

TEST INPUT

LOOP

TEST INPUT

TEST OUTPUT

TEST LANCER

TEST OUTPUT

TEST REPORTER

TEST INPUT
CODE
TYPE
GROUP
DATA INPUT Map<String, Object>

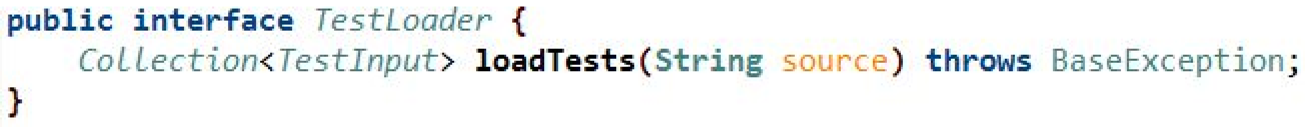
TEST OUTPUT
CODE
TYPE
GROUP
DATA OUTPUT Map<String, Object>

Object

- String
- Integer
- Double
- Date
- Array (S, I, D, D)
- Array of Arrays
- Array of Arrays of Arrays
- ...

Component: TestLoader

- Retrieves tests définitions



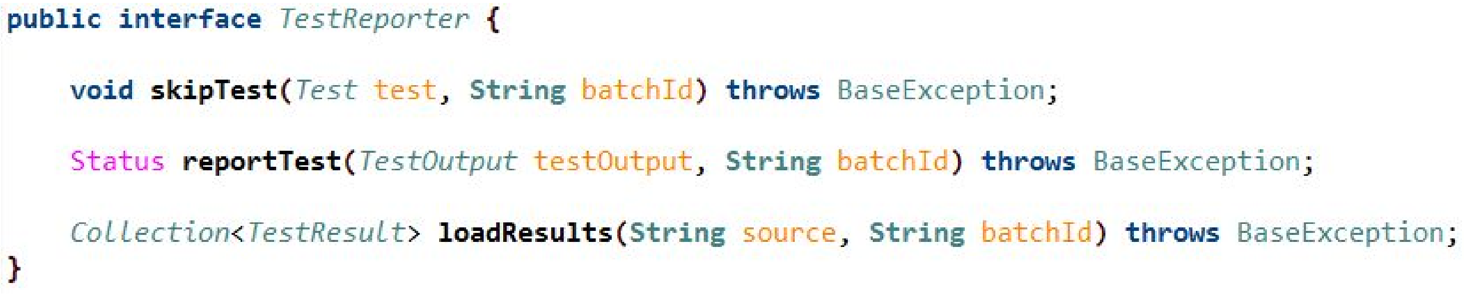
```
public interface TestLoader {
    Collection<TestInput> loadTests(String source) throws BaseException;
}
```

Implementation: TestLoaderExcel

- Uses an Excel file as a source of tests

Component: TestReporter

- Reports a Test result



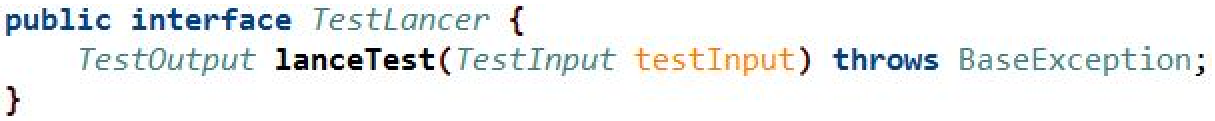
```
public interface TestReporter {

    void skipTest(Test test, String batchId) throws BaseException;

    Status reportTest(TestOutput testOutput, String batchId) throws BaseException;

    Collection<TestResult> loadResults(String source, String batchId) throws BaseException;
}
```

Implementation: TestReporterExcel

- Uses an Excel file to report the Test result

Component: TestLancer

- Un BLACK BOX for KTM
- Lance a test - sends TestInput, retrieves TestOutput



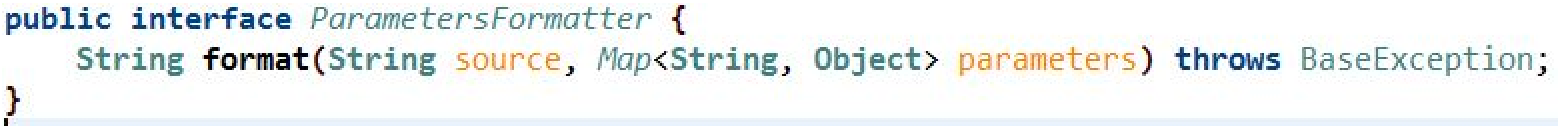
```
public interface TestLancer {
    TestOutput lanceTest(TestInput testInput) throws BaseException;
}
```

Implementation: TestLancerJobFile2Base

- Creates a file in Input directory
- Lance the tested job
- Retrieves differences in FS ans DB as parameters

Component: ParameterFormater

- To creates a String from Parameters



```
public interface ParametersFormatter {
    String format(String source, Map<String, Object> parameters) throws BaseException;
}
```

Implementation: ParameterFormaterTemplate

- Uses a String as template formated by parameters

TEMPLATE:

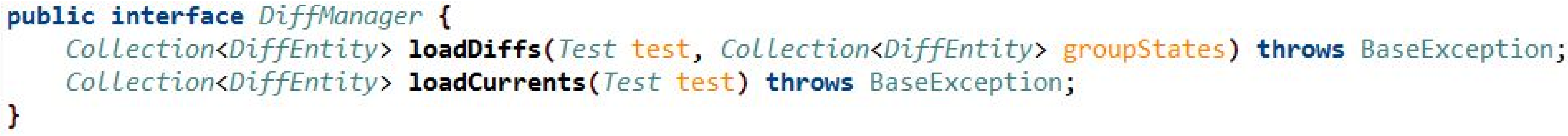
```
COLUMN_INTEGER;COLUMN_STRING;COLUMN_DOUBLE;COLUMN_DATE
[LOOP INDEX="LINE" TYPE="COUNT" NAME="COUNT_LINES"][PARAMETER NAME="COLUMN_INTEGER" INDEXES="LINE"];[PARAMETER
NAME="COLUMN_STRING" INDEXES="LINE"];[PARAMETER NAME="COLUMN_DOUBLE" INDEXES="LINE"];[PARAMETER NAME="COLUMN_DATE"
INDEXES="LINE"]
[END]
```

Component: DiffManager

- Loads current state of monitored system
- Loads changes in monitored system

- DiffEntity
  - Name
  - LastIndex
  - Collection of DiffItems

- DiffItem
  - Name
  - Index
  - Status



```java
public interface DiffManager {
    Collection<DiffEntity> loadDiffs(Test test, Collection<DiffEntity> groupStates) throws BaseException;
    Collection<DiffEntity> loadCurrents(Test test) throws BaseException;
}
```

Implementation: DiffManagerFilesystemLocal

- DiffEntity: a directory
- DiffItem: a file in the directory

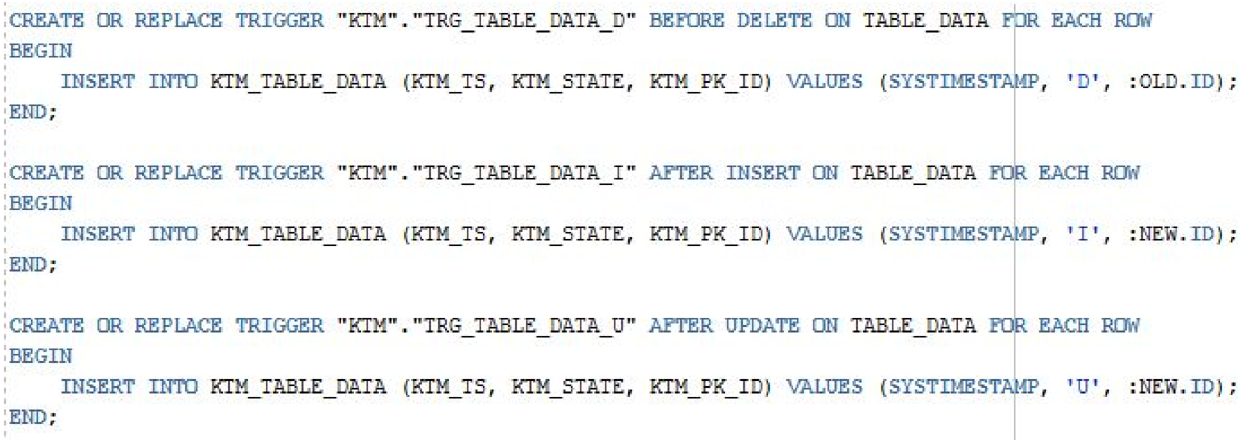
Implementation: DiffManagerDatabaseTrigger

- DiffEntity: a DB table
- DiffItem: a line in the DB table

DiffManagerDatabaseTrigger

TABLE_DATA	
ID	NUMBER
DATE	TIMESTAMP
BATCH_ID	VARCHAR2
COLUMN_INTEGER	NUMBER
COLUMN_STRING	VARCHAR2
COLUMN_DOUBLE	NUMBER
COLUMN_DATE	TIMESTAMP

KTM_TABLE_DATA	
KTM_TS	TIMESTAMP
KTM_STATE	VARCHAR2
KTM_PK_ID	NUMBER



```
CREATE OR REPLACE TRIGGER "KTM"."TRG_TABLE_DATA_D" BEFORE DELETE ON TABLE_DATA FOR EACH ROW
BEGIN
    INSERT INTO KTM_TABLE_DATA (KTM_TS, KTM_STATE, KTM_PK_ID) VALUES (SYSTIMESTAMP, 'D', :OLD.ID);
END;

CREATE OR REPLACE TRIGGER "KTM"."TRG_TABLE_DATA_I" AFTER INSERT ON TABLE_DATA FOR EACH ROW
BEGIN
    INSERT INTO KTM_TABLE_DATA (KTM_TS, KTM_STATE, KTM_PK_ID) VALUES (SYSTIMESTAMP, 'I', :NEW.ID);
END;

CREATE OR REPLACE TRIGGER "KTM"."TRG_TABLE_DATA_U" AFTER UPDATE ON TABLE_DATA FOR EACH ROW
BEGIN
    INSERT INTO KTM_TABLE_DATA (KTM_TS, KTM_STATE, KTM_PK_ID) VALUES (SYSTIMESTAMP, 'U', :NEW.ID);
END;
```

BEFORE

COLUMN_INTEGER	COLUMN_STRING	COLUMN_DOUBLE	COLUMN_DATE
1	AUDI	4	12:34:15 08/02/2017
2	PORCHE	911	19:08:15 08/07/2013
3	BMW	325	11:05:54 05/02/2001

AFTER

COLUMN_INTEGER	COLUMN_STRING	COLUMN_DOUBLE	COLUMN_DATE
1	AUDI	4	12:34:15 08/02/2017
3	BMW	535	11:05:54 05/02/2001
5	SKODA	1.2	06:51:09 01/01/1987
6	FIAT	500	15:22:17 07/02/1998