A MODERN ETL WITH NODE.JS AND ORACLE

ORACLE®

EPM2

WORKING WITH:

- node.js since 2013
- oracle for ~2years
- node-oracledb since it came out

CONTRIBUTING ON:

- Unitech/pm2 (nodejs)
- api-platform/api-platform (php)
- My own packages
  (1,5M downloads on npm last month)

CONTEXT

HTTP://DS-RESTAURATION.COM

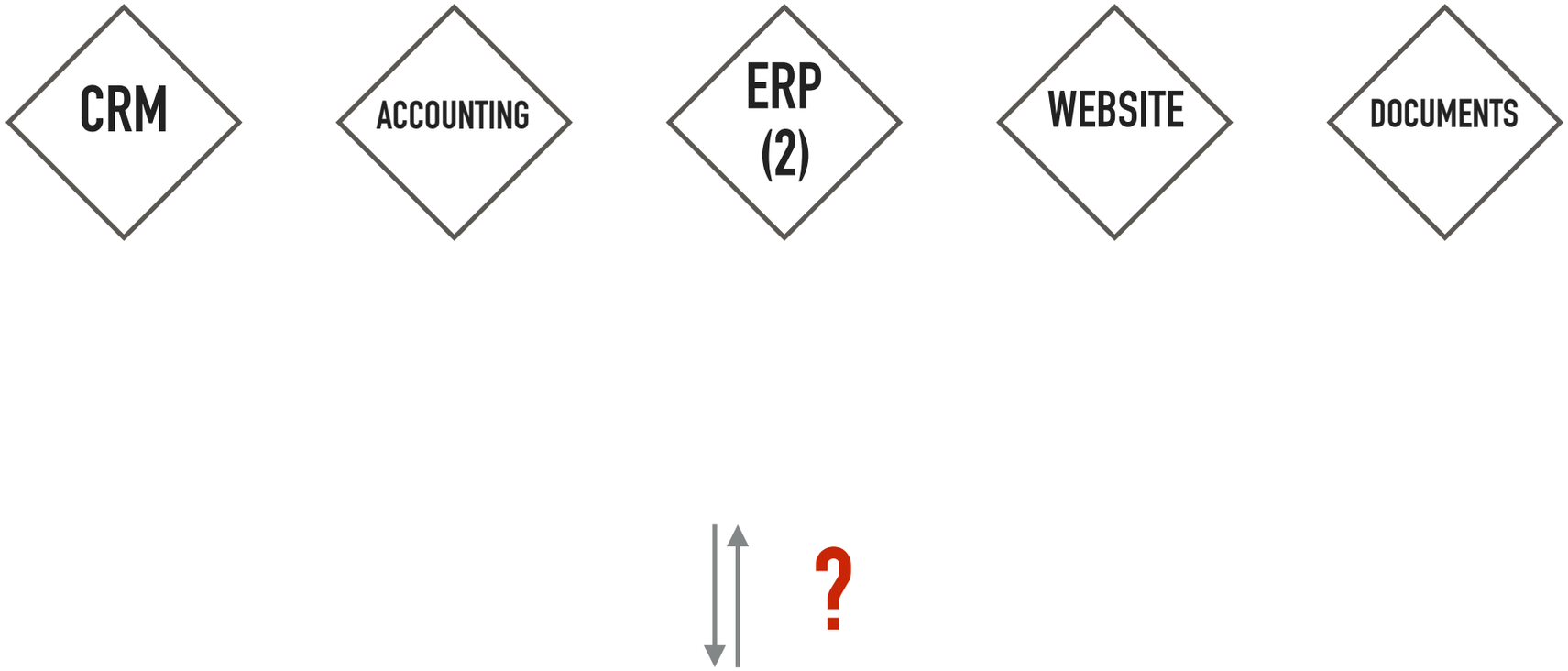
DS-RESTAURATION:

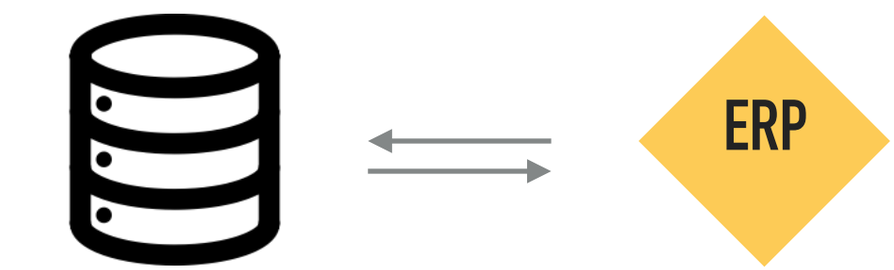
- 5 Warehouses (20 logistics base)
- +2500 products (fresh/frozen)
- 162 sales representatives (660 employees)
- 160 trucks

VOCABULARY:

- ERP (Enterprise resource planning)
- CRM (Customer Relationship management)
- DBMS (Database Management System)

THE PROBLEM





Oracle
Main database

A MODERN ETL PROCESS?

EXTRACT => TRANSFORM => LOAD

CONSTRAINTS:

- Real Time
- No control over data formats
- Complex business rules
- High availability
- Multiple systems (DBMS, Softwares, Protocols)

USE CASE (IMPORT)

Customer

7. Delivery

1. Sales representatives takes an order

2. Sync via 3G

3. Export data

CSV

CRM

4. Do some work!

6. Preparation

5. Warehouse gets order

ERP

ERP

Oracle

- Do not watch a directory for changes, use cron instead
- Files must be fully written! When transferring over the network, use a temporary directory before issuing `mv`

```
const combine = require('combined-stream')
const csvReader = require('csv-parser')
const stream = combine.create()

stream.append(fs.createReadStream(‘file1.csv'))
stream.append(fs.createReadStream(‘file2.csv'))

stream.pipe(csvReader({headers: [
  'id', 'product', 'quantity', 'packaging', 'currency', 'price'
]}))
.on('data', function(d) {
    connection.execute(‘INSERT INTO’, d)
})


})
```

USE CASE (EXPORT)

Customer

5. Presents new product

Sales representative

4. Sync via 3G

CRM

3. Sync via ftp/sftp

CSV

1. New product

ERP

Oracle

2. Do some work!

DATABASE POLLING

- A database trigger fills a temporary database
- Wait until `SELECT COUNT(1) > :num`
- Process data through a dedicated task (*microservice*) with pm2

EXPORT - STREAM DATA TO CSV

NODE-ORACLEDB IS STREAM COMPATIBLE!

```
const readable = connection.queryStream('SELECT * FROM TABLE')
const csvWriter = require('csv-write-stream')

pump(readable, csvWriter(), fs.createWriteStream(‘file.csv'), (err) => {

})
```

```
})
```

EXPORT - JSON

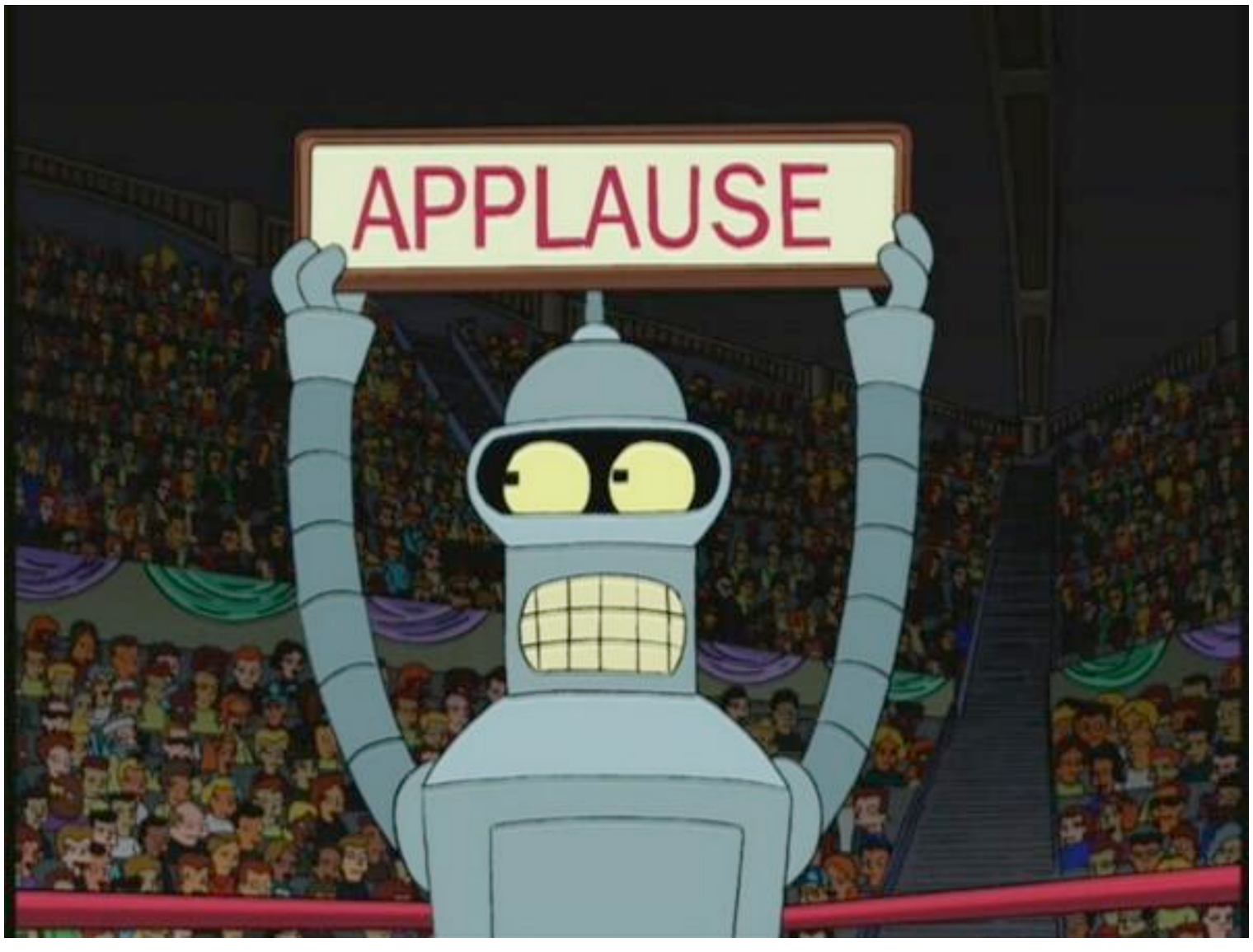
WAIT, I WANT TO WRITE JSON

```
const readable = connection.queryStream('SELECT * FROM TABLE')
const stringify = require(‘stringify-stream')

pump(readable, stringify(), fs.createWriteStream(‘file.json'), (err) => {

})
```

```
})
```

GO FURTHER

- Manage transformations and validation via schemas
- Job concurrency
- Dynamic scheduler
- Handle archives
- Read/write to/from ssh
- Keep statistics about the tasks
- Oracle Continuous Query Notification


APPLAUSE

REFERENCES

- https://github.com/soyuka/codetalk-oracle
- http://www.odtug.com/page/code-talk-series
- https://www.npmjs.com/package/pump
- https://www.npmjs.com/package/csv-write-stream
- https://www.npmjs.com/package/csv-parser
- https://www.npmjs.com/package/oracledb
- https://www.npmjs.com/package/pm2