FeedZai develops **FeedZai Pulse™**: High performance processing of large scale high throughput data loads to produce intelligence in real time.
What’s wrong with this picture?

Databases

Data Warehouses

Reports, ad-hoc queries, alarms
Voices from industry

“We are being flooded with data! Our batch processing of calls from last day is finishing at 6:00PM. It will not hold much more.”

Telco Business Intelligence Manager

“We are spending 12M€/year on data loads processing at our mainframes.”

BI Responsible at a Bank
Agenda

- Pulse and Complex Event Processing
- A first FeedZai Pulse application
- PulseQL overview
- Defining KPIs and Baselines
- Previous Projects and Conclusion
Part I – Pulse and Complex Event Processing
FeedZai Pulse™ – Actionable Data Processing

Pulse Engine - real-time actionable information

Complex Event Processing

Databases

Data Warehouses

Data Mining
Continuous Query Example

Example of a continuous query: for all bank transactions, computes the number of transactions by issuing bank on a ten minute time window.

```sql
transactions_per_Bank =
from transactions[10 min]
group by bank
select count()
```
How does it work?

Pulse Continuous Query Kernel

Input Data Streams

Output Data Streams

Operational Systems

Input Adaptors

Output Adaptors

PulseApp

Storage

Actuation

Dashboarding

Reporting

continuous queries
**DBMS vs Pulse**

- **DBMS**
  - Run-to-completion queries
  - Finite data
  - SQL
  - Not temporal
  - Multi-pass, disk-based
  - Queries read data

- **Pulse**
  - Continuous Queries
  - Never ending data stream
  - PulseQL
  - Temporal queries
  - Single-pass, memory-based
  - Data fires alarms, updates

![SQL database diagram](image)
Bank Transactions over a day

Is this normal behavior?
Pulse allows to easily define concepts as key performance indicators and baselines
Product Structure

Advanced Web Dashboarding

Specific Output Adapters

Standard Output Adapters

Key Performance Indicators
Auto-Stars Data Summarization
Advanced Baseline Checker
Smart-Time Historical Data
Live Drill Support
Automatic Drill and Dimensional Analysis
Alarms and Actuation
Clustering and High-Availability Support

Pulse Kernel Event Processing Engine

Specific Input Adapters

Standard Input Adapters

Legend
- Commercial
- Open Source
Energy – Nationwide Grid Monitoring

**Business Objective**
- Detect errors present in log events in real-time
- Alarms, System messages, Other

**Generate KPIs corresponding to:**
- Counts of alarms, system msgs, communications and other (4 types)
- The same counts aggregated by location (600 locations)
- Aggregate per location and type of alarm (2 types of alarms)
- Drill-down over all information

**Business Benefits:**
- Detect any error as soon as it is generated
- Real Time Error Mitigation
Banking – Infrastructure Operations

**Business Objective:**
- Detect when something is wrong in an environment of varying user behavior, **50.000.000** transactions/month and **200.000** terminals.

**Business Use:**
- Real Time charting dashboards and drill-down of data.
- Automatic KPI violation detection (even for sub hierarchies).
- Extremely configurable baselines reflecting different seasonal periods.

**Business Benefits:**
- Reduction of manned vigilance of systems.
- Lower costs on SLA violations.
Telecom – Network Traffic Analysis

**Business Objective:**
- Monitor the phone calls volume and duration nation-wide in real time.
- Real time overview of the entire communication status of the entire network.

**Business Use:**
- Real Time charting dashboards and drill-down of data.
- Ability to compare real time data with configurable baselines (last week, last month, last holiday season, etc).

**Business Benefits:**
- Detect problems in geographical areas, detect usage patterns.
- Reduce significantly the ETL process for DW.

15 million events per day
Space – Satellite Telemetry Processing

Business Objective:
• Process all 50,000 telemetry parameters/second as they are sent from the Satellite.

Business Use:
• Real Time dashboards and drill-down of data.
• Full Out Of Limits violation detection.
• Ability to compare real time data with configurable baselines.
• Synthetic parameter calculation.

Business Benefits:
• Early detection of satellite faults.
• Reduce expensive replaying from remote ground stations.

Demo: http://vimeo.com/11252513
PWD: feedzai
Part III – A first Pulse Application
Use-case Introduction

- Simplified banking use-case
- Each bank transaction originates an event
- An event consists on a tuple:
  
  \[(\text{timestamp}, \text{district}, \text{bank}, \text{rejected}, \text{duration})\]

- Examples:
  
  (20101217140000, 11, 1, 0, 278)
  (201012171400002, 9, 6, 0, 824)
  (201012171401015, 6, 2, 1, 264)
# Pulse Directory Structure

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bin</td>
<td>Pulse Executables</td>
</tr>
<tr>
<td>conf</td>
<td>Configuration</td>
</tr>
<tr>
<td>doc</td>
<td>Documentation</td>
</tr>
<tr>
<td>lib</td>
<td>Libraries</td>
</tr>
<tr>
<td>licenses</td>
<td>Licenses</td>
</tr>
<tr>
<td>log</td>
<td>Log files</td>
</tr>
<tr>
<td>pulseapps</td>
<td>Pulse Applications</td>
</tr>
<tr>
<td>services</td>
<td>Pulse Service Modules</td>
</tr>
<tr>
<td>var</td>
<td>Scrap Working Area</td>
</tr>
</tbody>
</table>
Pulse Running

~/tmp/pulse-server — java — 135x33

20:37:52: PulseManagerImpl.<init>:100
20:37:52: PulseManagerImpl.<init>:123
20:37:52: PulseManagerImpl.<init>:126
20:37:52: PulseManagerImpl.<init>:132
20:37:52: PulseManagerImpl.<init>:135
20:37:52: BaseService.start:87
20:37:53: ServletContainerService.loadConfig:111

xml'
20:37:54: PKernelService.loadStreams:106
20:37:54: BaseService.start:87
20:37:57: PKernelService.loadStreams:106
20:37:58: BaseService.start:87
20:37:58: BaseService.start:87
20:37:58: InputAdapterHandler.loadConfigs:122
20:37:58: InputAdapterHandler.loadConfigs:122

xml'
20:37:58: BaseService.start:87
20:37:58: BaseService.start:87
20:37:58: InputAdapterHandler.loadConfigs:122
20:37:58: InputAdapterHandler.loadConfigs:122

KPI
20:37:58: KpiServiceImpl.startStorageModule:231
KPI
20:37:58: KpiModule.load:105
KPI
20:37:58: BaseService.start:87
20:37:58: ServiceManagerImpl.startConfigured:88
20:38:01: ServiceManagerImpl.startConfigured:74
20:38:01: ServiceManagerImpl.startConfigured:75
20:38:02: ServiceManagerImpl.startConfigured:81
20:38:02: PKernelDriver.startRuntime:212
20:38:02: PulseManagerImpl.init:149
20:38:02: PulseManagerImpl.init:151

Loading information from: conf/pulse.xml
Configuring database location to: 'db/'
Configuring the JNDI namespace
Starting RMI Registry...
Starting the console...
Loading services...
MonitorService is running.
Loading Servlet Container configuration from 'conf/servlet-container.xml'
Reading from conf/pkernel/default.pkrnl...
ServletContainerService is running.
Reading from conf/pkernel/sibs.pkrnl...
PKernelService is running.
BufferService is running.
Loading TcpInputAdapterConfig configuration from 'conf/adapters/tcp-i
Loading SerializedInputAdapterConfig configuration from 'conf/adapter
TcpInputAdapterService is running.
SerializedInputAdapterService is running.
Connection to Database succeeded
Reading KPIs from conf/kpi/dummy.kpi...
KpiService is running.
Deploying Pulse applications from 'services'...
Done.
Deploying Pulse applications from 'pulseapps'...
Done.
The PKernel runtime has been started!
PKernelEngine started!
Startup completed!
A first Pulse Application

Ready? Set.. Go!!
A first Pulse Application

Defining the Stream

```java
// Pulse/conf/pkkernel/banking.pkrnl

transactions = Stream(
    timestamp: long,
    bank: int,
    district: int,
    rejected: bool,
    duration: int);
```

A Pulse Application is automatically generated using Maven. Compile and Deploy by:

```
~$ mvn clean install
```
Dumping raw events into a web page

Define the placeholder

```html
// pulse-workshop/src/main/webapp/example-01.html

<div id="placeholder"></div>
```

Acquire a listener to the transactions stream

```javascript
// pulse-workshop/src/main/webapp/example-01.html

$P('transactions').addListener(function(events) {
  $('#placeholder').append(events);
});
```
Results are immediately visible

Dumping raw events into a web page

Sample event (1 of 852): {
"timestamp":1292544000000,"district":11,"bank":1,"rejected":false,"duration":278}
Sample event (1 of 3034): {
"timestamp":1292544120000,"district":3,"bank":2,"rejected":false,"duration":54}
Sample event (1 of 5833): {
"timestamp":1292544534000,"district":11,"bank":2,"rejected":false,"duration":146}
Sample event (1 of 4502): {
"timestamp":1292545420000,"district":13,"bank":2,"rejected":false,"duration":183}
Sample event (1 of 5940): {
"timestamp":1292546318000,"district":11,"bank":1,"rejected":false,"duration":48}
Sample event (1 of 4291): {
"timestamp":1292548058000,"district":6,"bank":2,"rejected":false,"duration":282}
Sample event (1 of 4591): {
"timestamp":1292550055000,"district":14,"bank":2,"rejected":false,"duration":1351}
Sample event (1 of 3141): {
"timestamp":1292553455000,"district":22,"bank":2,"rejected":false,"duration":390}
Sample event (1 of 3338): {
"timestamp":1292558729000,"district":11,"bank":1,"rejected":false,"duration":163}
Sample event (1 of 3849): {
"timestamp":1292564155000,"district":5,"bank":1,"rejected":false,"duration":32}
Sample event (1 of 2959): {
"timestamp":1292566223000,"district":5,"bank":1,"rejected":false,"duration":54}
Sample event (1 of 4137): {
"timestamp":1292567084000,"district":13,"bank":2,"rejected":false,"duration":170}
Plotting the raw durations

Define the placeholder

```html
// pulse-workshop/src/main/webapp/example-02.html

<div id="chart"></div>
```

Acquire a listener to the `transactions` stream

```javascript
// pulse-workshop/src/main/webapp/example-02.html

var series = [{
    stream: 'transactions',
    field: 'duration',
    label: 'Duration (ms)' }, ];

var options = { title: { text: 'Raw Durations' },
                type: 'lines' };

$P.plot("#chart", series, options);
```
Visualizing raw events on a graph

Plotting the raw durations

Raw Durations

Duration (ms)
Part IV – PulseQL Overview
PulseQL

- A language for querying the information contained inside events
- Syntax inspired in SQL and Microsoft LINQ
- Operations supported:
  - Filtering
  - Projection
  - Aggregation
  - Grouping
  - Joins
  - Time-based windows (among others)
Examples of what can you ask

- “How many transactions per second?”
- “How many rejected transactions per bank, over the last 5 minutes?”
- “What is the average duration for each district?”
- “What is the percentage of transactions for each bank, over the last hour?”
Modeling the scenario
Modeling the scenario

| Timestamp | 12:00:00 |
| Bank      | CGD      |
| District  | Lisboa   |
| Duration  | 250 ms   |
| Rejected  | No       |

Time

Pulse
## Modeling the scenario

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>12:00:00</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bank</td>
<td>CGD</td>
</tr>
<tr>
<td>District</td>
<td>Lisboa</td>
</tr>
<tr>
<td>Duration</td>
<td>250 ms</td>
</tr>
<tr>
<td>Rejected</td>
<td>No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>12:00:30</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bank</td>
<td>BPI</td>
</tr>
<tr>
<td>District</td>
<td>Porto</td>
</tr>
<tr>
<td>Duration</td>
<td>500 ms</td>
</tr>
<tr>
<td>Rejected</td>
<td>No</td>
</tr>
</tbody>
</table>
Modeling the scenario

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>12:00:50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bank</td>
<td>BES</td>
</tr>
<tr>
<td>District</td>
<td>Braga</td>
</tr>
<tr>
<td>Duration</td>
<td>100 ms</td>
</tr>
<tr>
<td>Rejected</td>
<td>Yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>12:00:00</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bank</td>
<td>CGD</td>
</tr>
<tr>
<td>District</td>
<td>Lisboa</td>
</tr>
<tr>
<td>Duration</td>
<td>250 ms</td>
</tr>
<tr>
<td>Rejected</td>
<td>No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>12:00:30</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bank</td>
<td>BPI</td>
</tr>
<tr>
<td>District</td>
<td>Porto</td>
</tr>
<tr>
<td>Duration</td>
<td>500 ms</td>
</tr>
<tr>
<td>Rejected</td>
<td>No</td>
</tr>
</tbody>
</table>
## Modeling the scenario

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Bank</th>
<th>District</th>
<th>Duration</th>
<th>Rejected</th>
</tr>
</thead>
<tbody>
<tr>
<td>12:00:50</td>
<td>BES</td>
<td>Braga</td>
<td>100 ms</td>
<td>Yes</td>
</tr>
<tr>
<td>12:00:00</td>
<td>CGD</td>
<td>Lisboa</td>
<td>250 ms</td>
<td>No</td>
</tr>
<tr>
<td>12:01:15</td>
<td>CGD</td>
<td>Viseu</td>
<td>1000 ms</td>
<td>No</td>
</tr>
<tr>
<td>12:00:30</td>
<td>BPI</td>
<td>Porto</td>
<td>500 ms</td>
<td>No</td>
</tr>
</tbody>
</table>
Modeling the scenario

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>12:00:00</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bank</td>
<td>CGD</td>
</tr>
<tr>
<td>District</td>
<td>Lisboa</td>
</tr>
<tr>
<td>Duration</td>
<td>250 ms</td>
</tr>
<tr>
<td>Rejected</td>
<td>No</td>
</tr>
</tbody>
</table>

// PulseQL

```javascript
transactions = Stream(timestamp: long,
                        bank:     string,
                        district: string,
                        duration: int,
                        rejected: bool);
```
Querying your data

“Filter the rejected transactions”

```
// PulseQL

transactions = Stream(timestamp: long, ...)

rejections =
  from transactions
  where rejected == true;
```
Filter the rejected transactions

**transactions**

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Bank</th>
<th>District</th>
<th>Duration</th>
<th>Rejected</th>
</tr>
</thead>
<tbody>
<tr>
<td>12:00:00</td>
<td>CGD</td>
<td>Lisboa</td>
<td>250 ms</td>
<td>false</td>
</tr>
</tbody>
</table>

**rejections**

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Bank</th>
<th>District</th>
<th>Duration</th>
<th>Rejected</th>
</tr>
</thead>
</table>
Filter the rejected transactions

### transactions

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Bank</th>
<th>District</th>
<th>Duration</th>
<th>Rejected</th>
</tr>
</thead>
<tbody>
<tr>
<td>12:00:00</td>
<td>CGD</td>
<td>Lisboa</td>
<td>250 ms</td>
<td>false</td>
</tr>
<tr>
<td>12:00:30</td>
<td>BPI</td>
<td>Braga</td>
<td>100 ms</td>
<td>false</td>
</tr>
</tbody>
</table>

### rejections
Filter the rejected transactions

transactions

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Bank</th>
<th>District</th>
<th>Duration</th>
<th>Rejected</th>
</tr>
</thead>
<tbody>
<tr>
<td>12:00:00</td>
<td>CGD</td>
<td>Lisboa</td>
<td>250 ms</td>
<td>false</td>
</tr>
<tr>
<td>12:00:30</td>
<td>BPI</td>
<td>Braga</td>
<td>100 ms</td>
<td>false</td>
</tr>
<tr>
<td>12:00:50</td>
<td>BES</td>
<td>Porto</td>
<td>500 ms</td>
<td>true</td>
</tr>
</tbody>
</table>

rejections

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Bank</th>
<th>District</th>
<th>Duration</th>
<th>Rejected</th>
</tr>
</thead>
<tbody>
<tr>
<td>12:00:50</td>
<td>BES</td>
<td>Porto</td>
<td>500 ms</td>
<td>true</td>
</tr>
</tbody>
</table>
Filter the rejected transactions

### transactions

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Bank</th>
<th>District</th>
<th>Duration</th>
<th>Rejected</th>
</tr>
</thead>
<tbody>
<tr>
<td>12:00:00</td>
<td>CGD</td>
<td>Lisboa</td>
<td>250 ms</td>
<td>false</td>
</tr>
<tr>
<td>12:00:30</td>
<td>BPI</td>
<td>Braga</td>
<td>100 ms</td>
<td>false</td>
</tr>
<tr>
<td>12:00:50</td>
<td>BES</td>
<td>Porto</td>
<td>500 ms</td>
<td>true</td>
</tr>
<tr>
<td>12:01:15</td>
<td>CGD</td>
<td>Viseu</td>
<td>1000 ms</td>
<td>false</td>
</tr>
</tbody>
</table>

### rejections

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Bank</th>
<th>District</th>
<th>Duration</th>
<th>Rejected</th>
</tr>
</thead>
<tbody>
<tr>
<td>12:00:50</td>
<td>BES</td>
<td>Porto</td>
<td>500 ms</td>
<td>true</td>
</tr>
</tbody>
</table>
Filter the rejected transactions

### transactions

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Bank</th>
<th>District</th>
<th>Duration</th>
<th>Rejected</th>
</tr>
</thead>
<tbody>
<tr>
<td>12:00:00</td>
<td>CGD</td>
<td>Lisboa</td>
<td>250 ms</td>
<td>false</td>
</tr>
<tr>
<td>12:00:30</td>
<td>BPI</td>
<td>Braga</td>
<td>100 ms</td>
<td>false</td>
</tr>
<tr>
<td>12:00:50</td>
<td>BES</td>
<td>Porto</td>
<td>500 ms</td>
<td>true</td>
</tr>
<tr>
<td>12:01:15</td>
<td>CGD</td>
<td>Viseu</td>
<td>1000 ms</td>
<td>false</td>
</tr>
<tr>
<td>12:01:45</td>
<td>BES</td>
<td>Faro</td>
<td>750 ms</td>
<td>true</td>
</tr>
</tbody>
</table>

### rejections

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Bank</th>
<th>District</th>
<th>Duration</th>
<th>Rejected</th>
</tr>
</thead>
<tbody>
<tr>
<td>12:00:50</td>
<td>BES</td>
<td>Porto</td>
<td>500 ms</td>
<td>true</td>
</tr>
<tr>
<td>12:01:45</td>
<td>BES</td>
<td>Faro</td>
<td>750 ms</td>
<td>true</td>
</tr>
</tbody>
</table>
Time-based windows and aggregators

“How many rejections during the last 5 minutes?”

```c
// PulseQL
reject_5min = rejections[5 min].count();
```
Time-based windows and aggregators

**rejections**

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Bank</th>
<th>District</th>
<th>Duration</th>
<th>Rejected</th>
</tr>
</thead>
</table>

**reject_5min**

\[= 0\]
### Time-based windows and aggregators

#### rejections

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Bank</th>
<th>District</th>
<th>Duration</th>
<th>Rejected</th>
</tr>
</thead>
<tbody>
<tr>
<td>17:00:00</td>
<td>CGD</td>
<td>Lisboa</td>
<td>250 ms</td>
<td>true</td>
</tr>
</tbody>
</table>

#### reject_5min

\[ = 1 \]
Time-based windows and aggregators

Rejections

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Bank</th>
<th>District</th>
<th>Duration</th>
<th>Rejected</th>
</tr>
</thead>
<tbody>
<tr>
<td>17:00:00</td>
<td>CGD</td>
<td>Lisboa</td>
<td>250 ms</td>
<td>true</td>
</tr>
<tr>
<td>17:02:30</td>
<td>BPI</td>
<td>Braga</td>
<td>100 ms</td>
<td>true</td>
</tr>
</tbody>
</table>

reject_5min = 2
# Time-based windows and aggregators

## rejections

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Bank</th>
<th>District</th>
<th>Duration</th>
<th>Rejected</th>
</tr>
</thead>
<tbody>
<tr>
<td>17:00:00</td>
<td>CGD</td>
<td>Lisboa</td>
<td>250 ms</td>
<td>true</td>
</tr>
<tr>
<td>17:02:30</td>
<td>BPI</td>
<td>Braga</td>
<td>100 ms</td>
<td>true</td>
</tr>
<tr>
<td>17:04:00</td>
<td>BES</td>
<td>Porto</td>
<td>500 ms</td>
<td>true</td>
</tr>
</tbody>
</table>

### reject_5min

= 3

FeedZai Confidential
Time-based windows and aggregators

**rejections**

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Bank</th>
<th>District</th>
<th>Duration</th>
<th>Rejected</th>
</tr>
</thead>
<tbody>
<tr>
<td>17:00:00</td>
<td>CGD</td>
<td>Lisboa</td>
<td>250 ms</td>
<td>true</td>
</tr>
<tr>
<td>17:02:30</td>
<td>BPI</td>
<td>Braga</td>
<td>100 ms</td>
<td>true</td>
</tr>
<tr>
<td>17:04:00</td>
<td>BES</td>
<td>Porto</td>
<td>500 ms</td>
<td>true</td>
</tr>
<tr>
<td>17:06:00</td>
<td>CGD</td>
<td>Viseu</td>
<td>1000 ms</td>
<td>true</td>
</tr>
</tbody>
</table>

5 min

**reject_5min**

= 3
Time-based windows and aggregators

rejections

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Bank</th>
<th>District</th>
<th>Duration</th>
<th>Rejected</th>
</tr>
</thead>
<tbody>
<tr>
<td>17:00:00</td>
<td>CGD</td>
<td>Lisboa</td>
<td>250 ms</td>
<td>true</td>
</tr>
<tr>
<td>17:02:30</td>
<td>BPI</td>
<td>Braga</td>
<td>100 ms</td>
<td>true</td>
</tr>
<tr>
<td>17:04:00</td>
<td>BES</td>
<td>Porto</td>
<td>500 ms</td>
<td>true</td>
</tr>
<tr>
<td>17:06:00</td>
<td>CGD</td>
<td>Viseu</td>
<td>1000 ms</td>
<td>true</td>
</tr>
<tr>
<td>17:10:00</td>
<td>BES</td>
<td>Faro</td>
<td>750 ms</td>
<td>true</td>
</tr>
</tbody>
</table>

reject_5min

= 2
And more

"The duration of each transaction, in seconds"

```pulseeql
// PulseQL

durations =
    from transactions
    select duration / 1000.0;
```

"The average duration over the last minute"

```pulseeql
// PulseQL

avg_duration = transactions[1 min].avg(duration);
```
“The bank with the maximum duration, ever.”

```
// PulseQL
slowest_bank = transactions.maxBy(duration).bank;
```

“The number of rejected transactions in Beja, during the last hour”

```
// PulseQL
beja_reject_count = (from transactions
  where district == “Beja”)[1 hour].count()
```
Types of windows

- **Sliding windows**

- **Jumping windows**

- **Landmark windows**
Types of windows

- Sliding windows
- Jumping windows
- Landmark windows
Types of windows

- **Sliding windows**

- **Jumping windows**

- **Landmark windows**
Types of windows

- Sliding windows
- Jumping windows
- Landmark windows
Types of windows

- **Sliding windows**

```pulsedl
// The transactions during the last 5 minutes
transactions[5 min]
```

- **Jumping windows**

- **Landmark windows**
Types of windows

**Sliding windows**

```pulsedsl
// PulseQL

// The transactions during the last 5 minutes
transactions[5 min]

// The last 10 transactions
transactions[10 tuples]
```

**Jumping windows**

**Landmark windows**
Types of windows

- Sliding windows
- Jumping windows

- Landmark windows
Types of windows

- Sliding windows
- Jumping windows
- Landmark windows
Types of windows

- **Sliding windows**
- **Jumping windows**
- **Landmark windows**
Types of windows

- Sliding windows
- Jumping windows
- Landmark windows

Time

5 min
Types of windows

- **Sliding windows**
- **Jumping windows**
- **Landmark windows**
Types of windows

- Sliding windows
- Jumping windows
- Landmark windows
Types of windows

- Sliding windows
- Jumping windows

- Landmark windows
Types of windows

- **Sliding windows**
- **Jumping windows**
- **Landmark windows**
Types of windows

- Sliding windows
- Jumping windows
- Landmark windows
Types of windows

- Sliding windows
- Jumping windows
- Landmark windows
Types of windows

- Sliding windows
- Jumping windows

// PulseQL

// The transactions in the last batch of 5 mins.
transactions[every 5 min]

- Landmark windows
Types of windows

- **Sliding windows**
- **Jumping windows**

// PulseQL

```
// The transactions in the last batch of 5 mins.
transactions[every 5 min]
```

```
// The transactions in the last batch of 10.
transactions[every 10 tuples]
```

- **Landmark windows**
Types of windows

- **Sliding windows**
- **Jumping windows**
- **Landmark windows**

```pseudocode
// PulseQL

// Resets the window every time the hour changes.
transactions[reset (hour: _)]
```
Types of windows

» Sliding windows
» Jumping windows
» Landmark windows

// PulseQL

// Resets the window every time the hour changes.
transactions[reset (hour: _)]

// Alternatively:
transactions[reset hourly]
Types of windows

- Sliding windows
- Jumping windows
- Landmark windows

// PulseQL

// Resets the window every time the hour changes.
transactions[reset (hour: _)]

// Alternatively:
transactions[reset hourly]

// Resets the window every Friday at 17h.
transactions[reset (weekday: fri, hour: 17)]
Queries over groups

“The average duration per district, over the last hour”

// PulseQL

durPerDistrict =
    from transactions[1 hour]
    group by district
    select avg_duration: avg(duration);
Queries over groups

transactions

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>District</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>20:00</td>
<td>Lisboa</td>
<td>250 ms</td>
</tr>
</tbody>
</table>

durPerDistrict

<table>
<thead>
<tr>
<th>district</th>
<th>avg_duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lisboa</td>
<td>250 ms</td>
</tr>
</tbody>
</table>
# Queries over groups

## transactions

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>District</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>20:00</td>
<td>Lisboa</td>
<td>250 ms</td>
</tr>
<tr>
<td>20:10</td>
<td>Aveiro</td>
<td>300 ms</td>
</tr>
</tbody>
</table>

1 hour

## durPerDistrict

<table>
<thead>
<tr>
<th>district</th>
<th>avg_duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lisboa</td>
<td>250 ms</td>
</tr>
<tr>
<td>Aveiro</td>
<td>300 ms</td>
</tr>
</tbody>
</table>
Queries over groups

transactions

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>District</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>20:00</td>
<td>Lisboa</td>
<td>250 ms</td>
</tr>
<tr>
<td>20:10</td>
<td>Aveiro</td>
<td>300 ms</td>
</tr>
<tr>
<td>20:25</td>
<td>Évora</td>
<td>500 ms</td>
</tr>
</tbody>
</table>

1 hour

durPerDistrict

<table>
<thead>
<tr>
<th>district</th>
<th>avg_duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lisboa</td>
<td>250 ms</td>
</tr>
<tr>
<td>Aveiro</td>
<td>300 ms</td>
</tr>
<tr>
<td>Évora</td>
<td>500 ms</td>
</tr>
</tbody>
</table>
Queries over groups

transactions

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>District</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>20:00</td>
<td>Lisboa</td>
<td>250 ms</td>
</tr>
<tr>
<td>20:10</td>
<td>Aveiro</td>
<td>300 ms</td>
</tr>
<tr>
<td>20:25</td>
<td>Évora</td>
<td>500 ms</td>
</tr>
<tr>
<td>20:40</td>
<td>Lisboa</td>
<td>750 ms</td>
</tr>
</tbody>
</table>

1 hour

durPerDistrict

district | avg_duration
----------|---------------
Lisboa    | 500 ms
Aveiro    | 300 ms
Évora     | 500 ms
Queries over groups

transactions

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>District</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>20:00</td>
<td>Lisboa</td>
<td>250 ms</td>
</tr>
<tr>
<td>20:10</td>
<td>Aveiro</td>
<td>300 ms</td>
</tr>
<tr>
<td>20:25</td>
<td>Évora</td>
<td>500 ms</td>
</tr>
<tr>
<td>20:40</td>
<td>Lisboa</td>
<td>750 ms</td>
</tr>
<tr>
<td>21:20</td>
<td>Aveiro</td>
<td>100 ms</td>
</tr>
</tbody>
</table>

1 hour

durPerDistrict

<table>
<thead>
<tr>
<th>district</th>
<th>avg_duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lisboa</td>
<td>750 ms</td>
</tr>
<tr>
<td>Aveiro</td>
<td>100 ms</td>
</tr>
<tr>
<td>Évora</td>
<td>500 ms</td>
</tr>
</tbody>
</table>
More queries over groups

“The percentage of rejections for each bank”

```java
// PulseQL

totalRejections = rejections.count();

rejectPerBank_Perc =
  from rejections
  group by bank
  select perc: count() * 100.0 / totalRejections;
```
More queries over groups

**rejections**

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Bank</th>
</tr>
</thead>
<tbody>
<tr>
<td>20:00</td>
<td>BPI</td>
</tr>
</tbody>
</table>

**rejectPerBank_Perc**

<table>
<thead>
<tr>
<th>bank</th>
<th>perc</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPI</td>
<td>100 %</td>
</tr>
</tbody>
</table>
More queries over groups

### Rejections

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Bank</th>
</tr>
</thead>
<tbody>
<tr>
<td>20:00</td>
<td>BPI</td>
</tr>
<tr>
<td>20:10</td>
<td>BES</td>
</tr>
</tbody>
</table>

### rejectPerBank_Perc

<table>
<thead>
<tr>
<th>bank</th>
<th>perc</th>
</tr>
</thead>
<tbody>
<tr>
<td>BES</td>
<td>50 %</td>
</tr>
<tr>
<td>BPI</td>
<td>50 %</td>
</tr>
</tbody>
</table>
More queries over groups

**Rejections**

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Bank</th>
</tr>
</thead>
<tbody>
<tr>
<td>20:00</td>
<td>BPI</td>
</tr>
<tr>
<td>20:10</td>
<td>BES</td>
</tr>
<tr>
<td>20:25</td>
<td>CGD</td>
</tr>
</tbody>
</table>

**rejectPerBank_Perc**

<table>
<thead>
<tr>
<th>Bank</th>
<th>perc</th>
</tr>
</thead>
<tbody>
<tr>
<td>BES</td>
<td>33 %</td>
</tr>
<tr>
<td>BPI</td>
<td>33 %</td>
</tr>
<tr>
<td>CGD</td>
<td>33 %</td>
</tr>
</tbody>
</table>
More queries over groups

rejections

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Bank</th>
</tr>
</thead>
<tbody>
<tr>
<td>20:00</td>
<td>BPI</td>
</tr>
<tr>
<td>20:10</td>
<td>BES</td>
</tr>
<tr>
<td>20:25</td>
<td>CGD</td>
</tr>
<tr>
<td>20:40</td>
<td>CGD</td>
</tr>
</tbody>
</table>

rejectPerBank_Perc

<table>
<thead>
<tr>
<th>bank</th>
<th>perc</th>
</tr>
</thead>
<tbody>
<tr>
<td>BES</td>
<td>25 %</td>
</tr>
<tr>
<td>BPI</td>
<td>25 %</td>
</tr>
<tr>
<td>CGD</td>
<td>50 %</td>
</tr>
</tbody>
</table>
More queries over groups

rejections

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Bank</th>
</tr>
</thead>
<tbody>
<tr>
<td>20:00</td>
<td>BPI</td>
</tr>
<tr>
<td>20:10</td>
<td>BES</td>
</tr>
<tr>
<td>20:25</td>
<td>CGD</td>
</tr>
<tr>
<td>20:40</td>
<td>CGD</td>
</tr>
<tr>
<td>21:20</td>
<td>BES</td>
</tr>
</tbody>
</table>

rejectPerBank_Perc

<table>
<thead>
<tr>
<th>bank</th>
<th>perc</th>
</tr>
</thead>
<tbody>
<tr>
<td>BES</td>
<td>40 %</td>
</tr>
<tr>
<td>BPI</td>
<td>20 %</td>
</tr>
<tr>
<td>CGD</td>
<td>40 %</td>
</tr>
</tbody>
</table>
Summary

- PulseQL allows the developer to perform queries over events
- It supports the most commonly used operations, such as filtering, grouping, windows and aggregations
- It is easy to learn and simple to use (but not too simple!)
Part V – Defining KPIs and Baselines
Key Performance Indicators

“*The global transaction rate, per second*”

```pulseql
// PulseQL
tps = transactions[1 sec].count();
```

Defining a KPI over this query is almost identical:

```pulseql
// KPI Definition
@Kpi
@Aggregate(rate = 5 min)
{
    tps = transactions[1 sec].count();
}
```
KPIs are directly visible

Plotting the transactions per second

Global Transactions (trx/s)
Baselines

👉 Usually, a KPI only makes sense when compared against an expected value (e.g., comparing the current transaction rate with last week’s, at this same time)

👉 How can I compare the transaction rate with...
   – Last week’s 10 minute average, at this same time
   – Yesterday’s average, at this hour, +/- its standard deviation
Baselines

“Last week’s 10 minute average, at this same time”

```
// Baseline Definition

@Instant
start = now, -1 week, -5 minute
end   = now, -1 week, +5 minute

@Interval
lastWeek = start .. end

@Baseline(applyTo = tps, rate = 5 min)
lastWeekAvg = avg(*) @ lastWeek
```
Baselines

“Yesterday’s average, at this hour, +/- its standard deviation”

```plaintext
// Baseline Definition

@Instant
   start = now, -1 day, start of hour
   end   = start, +1 hour

@Interval
   yd = start .. end

@Baseline(applyTo = *, rate = 20 min)
   ydAvg   = avg(*) @ yd every 20 minute
   ydHigh  = avg(*) @ yd + stddev(*) @ yd
   ydLow   = avg(*) @ yd - stddev(*) @ yd
```
Baselines off-the-shelf!

Plotting the transactions per second, among with a baseline
Drill-down is also available

Creating a drill-down table

KPI's and Baselines

<table>
<thead>
<tr>
<th>KPI's</th>
<th>transactions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>transactions</td>
</tr>
<tr>
<td></td>
<td>Avelro</td>
</tr>
<tr>
<td></td>
<td>1.45</td>
</tr>
<tr>
<td></td>
<td>Banco A</td>
</tr>
<tr>
<td></td>
<td>0.37</td>
</tr>
<tr>
<td></td>
<td>Banco B</td>
</tr>
<tr>
<td></td>
<td>0.70</td>
</tr>
<tr>
<td></td>
<td>Banco C</td>
</tr>
<tr>
<td></td>
<td>0.15</td>
</tr>
<tr>
<td></td>
<td>Coimbra</td>
</tr>
<tr>
<td></td>
<td>0.90</td>
</tr>
<tr>
<td></td>
<td>Banco A</td>
</tr>
<tr>
<td></td>
<td>0.27</td>
</tr>
<tr>
<td></td>
<td>Banco B</td>
</tr>
<tr>
<td></td>
<td>0.66</td>
</tr>
<tr>
<td></td>
<td>Banco C</td>
</tr>
<tr>
<td></td>
<td>0.06</td>
</tr>
<tr>
<td></td>
<td>Faro</td>
</tr>
<tr>
<td></td>
<td>0.90</td>
</tr>
<tr>
<td></td>
<td>Banco A</td>
</tr>
<tr>
<td></td>
<td>0.22</td>
</tr>
<tr>
<td></td>
<td>Banco B</td>
</tr>
<tr>
<td></td>
<td>0.47</td>
</tr>
<tr>
<td></td>
<td>Banco C</td>
</tr>
<tr>
<td></td>
<td>0.12</td>
</tr>
<tr>
<td></td>
<td>Lisboa</td>
</tr>
<tr>
<td></td>
<td>7.78</td>
</tr>
<tr>
<td></td>
<td>Banco A</td>
</tr>
<tr>
<td></td>
<td>2.40</td>
</tr>
<tr>
<td></td>
<td>Banco B</td>
</tr>
<tr>
<td></td>
<td>5.10</td>
</tr>
<tr>
<td></td>
<td>Banco C</td>
</tr>
<tr>
<td></td>
<td>0.15</td>
</tr>
</tbody>
</table>

Dimensions and Filters

<table>
<thead>
<tr>
<th>District</th>
<th>7 filtrs seleccionados</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bank</td>
<td>Banco A, Banco B, Banco C</td>
</tr>
</tbody>
</table>
Conclusion

- Real-time data processing and analytics on the web
- State-of-the-art KPI and baseline checking
- Automatic alarms at different hierarchies
- Visually explore incoming data
- Deploy in days instead of months