Manage eazyBI in Large-Scale Jira Instances

Daina Tupule
Taking on new challenges

Daina Tupule
Day in eazybi Cloud

50K imports

120K report executions

and growing ..
eazyBI is designed to run with many accounts
Community Day
April 8, 2022

Measure, don’t guess

What is account

Tuning up
Do not guess - measure
Team account: support reporting
### General account: usage statistics

<table>
<thead>
<tr>
<th></th>
<th>% of all</th>
<th>Table</th>
<th>Bar chart</th>
<th>Line chart</th>
<th>Pie chart</th>
<th>Scatter chart</th>
<th>Timeline chart</th>
<th>Map chart</th>
<th>Gantt chart</th>
<th>Gauge</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>2018</strong></td>
<td></td>
<td>24%</td>
<td>42%</td>
<td>12%</td>
<td>10%</td>
<td>2%</td>
<td>3%</td>
<td>0%</td>
<td>0%</td>
<td>7%</td>
</tr>
<tr>
<td><strong>2019</strong></td>
<td></td>
<td>27%</td>
<td>32%</td>
<td>16%</td>
<td>11%</td>
<td>1%</td>
<td>3%</td>
<td>0%</td>
<td>0%</td>
<td>9%</td>
</tr>
<tr>
<td><strong>2020</strong></td>
<td></td>
<td>27%</td>
<td>36%</td>
<td>15%</td>
<td>10%</td>
<td>1%</td>
<td>5%</td>
<td>0%</td>
<td>1%</td>
<td>6%</td>
</tr>
<tr>
<td><strong>2021</strong></td>
<td></td>
<td>33%</td>
<td>34%</td>
<td>12%</td>
<td>10%</td>
<td>0%</td>
<td>4%</td>
<td>0%</td>
<td>1%</td>
<td>5%</td>
</tr>
<tr>
<td><strong>2022</strong></td>
<td></td>
<td>36%</td>
<td>34%</td>
<td>11%</td>
<td>8%</td>
<td>0%</td>
<td>4%</td>
<td>0%</td>
<td>1%</td>
<td>6%</td>
</tr>
</tbody>
</table>
**TEAM ACCOUNT**

- Small data set
- Better performance
- More freedom, rules suggested

**GENERAL ACCOUNT**

- Larger data set
- More expertise needed to get good working reports
- Strict rules to avoid haoss
Accounts
Account

Data

Scope
Import projects and use JQL query to specify what information will be imported into the account

Details
Select custom fields, issue cycles, and apps - what details you would like to see and use for imported issues in your reports

Extend the data model *
eazyBI allows extending the data model with additional data import and custom javascript
Extend the data model

Use [additional data import](#) to get more details for sprints, users, projects, etc.

**Build custom hierarchies** in Time, project, sprint, users, or other custom field dimensions

**Define new account specific fields** [PLANNED](#)

**Override field behaviour** (Jira fields / global customs) using [custom JavaScript code](#) and ([PLANNED](#)) setting a different behaviour for fields
Use samples and default set of measures and calculations

eazyBI comes with a prebuilt set of measures, sample reports, and default calculations. Use our demo account examples for more inspiration

**Template accounts**
Build a set of measures and reports specific to your needs. They can give some common ground for all teams and could be used as building blocks in new reports

**Extend the metrics** *
Allow teams to create their own set of measures for specific needs
Extend the metrics

Report specific measures
The primary choice when building any report.

Shared user defined measures
Move “proved report specific measures” as shared. Keep them organised. Consider using some naming patterns

Template reports and measures
Move shared reports and measures to template accounts if they could be reusable for other teams. Consistent naming is important there.
Who can access data

eazyBI does not use Jira permissions to access data in reports. Set users/groups who can access the data per account.

**Dashboard viewer** - read predefined reports published via dashboards
**Viewer** - read any report
**User** - create and edit own reports and measures
**Report admin** - create/edit any report, create shared user defined calculated members and measures
**Data admin** - create/edit data sources, import data
**User admin** - add users to the account
**Owner** - one user - responsible of the account, will receive error messages on imports
MANAGE EAZYBI IN LARGE-SCALE JIRA INSTANCES

Tuning up
## Usage statistics *available for cloud and server/DC*

<table>
<thead>
<tr>
<th>(usage statistics)</th>
<th>Source Data</th>
<th>Data size</th>
<th>Reports</th>
<th>Template (used in accounts)</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTERNAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Customer cases
- Owner: Adam Mint (admin)
- Created: 2 years ago
- Updated: 14 days ago
- Reports: 9
- Template: Jira
- Files size: 25.7 KB
- Issues: 1860

### DEMO
- Owner: Adam Mint (admin)
- Created: 2 years ago
- Updated: about 15 hours ago
- Reports: 117
- Template: Jira
- Files size: 49.2 MB
- Issues: 1838

### DEMO - Insight
- Owner: Adam Mint (admin)
- Created: about a year ago
- Updated: about 17 hours ago
- Reports: 16
- Template: Insight
- Files size: 3.5 MB
- Issues: 132

### DevOps
- Owner: Adam Mint (admin)
- Created: about a year ago
- Updated: No regular import
- Reports: 1
- Template: Jira
- Files size: 3.5 MB
- Issues: 132
# Usage statistics

<table>
<thead>
<tr>
<th>Customer cases (3)</th>
<th>Import Count</th>
<th>Report Executions</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEMO (4)</td>
<td>146</td>
<td>168</td>
</tr>
<tr>
<td>DEMO - insight (45)</td>
<td>60</td>
<td>36</td>
</tr>
<tr>
<td>DevOps (53)</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>experimental (19)</td>
<td>18</td>
<td>51</td>
</tr>
<tr>
<td>Insight (6)</td>
<td>217</td>
<td></td>
</tr>
<tr>
<td>Insight inheritance (7)</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td>Jira (52)</td>
<td>61</td>
<td></td>
</tr>
<tr>
<td>Jira default (1)</td>
<td>307</td>
<td>142</td>
</tr>
<tr>
<td>Jira Service Management (8)</td>
<td>4</td>
<td>72</td>
</tr>
<tr>
<td>Jira uses cases (11)</td>
<td>241</td>
<td>25</td>
</tr>
<tr>
<td>Portfolio (10)</td>
<td>329</td>
<td>4</td>
</tr>
<tr>
<td>Source data - Custom data cubes (31)</td>
<td>1,502</td>
<td></td>
</tr>
<tr>
<td>Source data - Jira custom cube (37)</td>
<td>181</td>
<td></td>
</tr>
</tbody>
</table>
Tuning up

Optimize reports

Detect the slow reports
Use Usage statistics to detect slow running reports. Check both the average execution time and the total report execution time.

Use best practices when creating reports
Check out the documentation page Best practices for report creation

Profiling *
Use profiling to detect bottlenecks in the report execution. Function Profiling can help you detect measure working slow or part of the measure working slow.
Report example for profiling

Version report works for versions with some resolved issues with Story points and some unresolved issues with Story points. Usually used for unreleased Versions. You should **add Time members** for Version period (dates between Version report start date and Planned completion date at least) to represent data over Time.
Report example for profiling

```python
Profiling(
    "name",
    "<your formula goes here>
)
```

Define calculated member formula

```
Measures.[[Predicted completion line]] =
1 -- annotations.group = Predicted
2 Profiling()
3 "Predicted completion line",
4 CASE WHEN
5 -- apply for predicted period only
6 (DateInPeriod(7 [Measures].[Version report Start Date],
7 [Time].CurrentHierarchyMember)
9 OR
10 DateBetween([Time].CurrentHierarchyMember.StartDate,
11 [Measures].[Version report Start Date],
12 [Measures].[Predicted completion date]))
```

Read calculated members tutorial to learn about calculated member formulas. You can select members, operators and frequently used functions from sidebar to insert them into calculated member formula.

Used only in this report. Convert to report specific.
Tuning up

- SumFunDef invoked 2663 times for total of 101ms. (Avg. 0ms/invocation)
- Cumulative Story Points resolved till today invoked 10 times for total of 2ms. (Avg. 0ms/invocation)
- version report start date invoked 2375 times for total of 2297ms. (Avg. 0ms/invocation)
- Predicted completion line invoked 48 times for total of 0ms. (Avg. 0ms/invocation)
- FilterFunDef invoked 10 times for total of 2355ms. (Avg. 235ms/invocation)
- MinMaxFunDef invoked 183 times for total of 6ms. (Avg. 0ms/invocation)
- Predicted completion date invoked 327 times for total of 1ms. (Avg. 0ms/invocation)
- Time in version invoked 1060 times for total of 2328ms. (Avg. 2ms/invocation)
- SqlStatement-SqlMemberSource.getMemberChildren invoked 2 times for total of 1ms. (Avg. 0ms/invocation)
- SqlStatement-SqlTupleReader.readTuples [[Time.Weekly].[Day]] invoked 1 times for total of 16ms. (Avg. 16ms/invocation)
- SqlStatement-Segment.load invoked 7 times for total of 31ms. (Avg. 4ms/invocation)
- SqlStatement-SqlTupleReader.readTuples [[Time.Weekly].[Week]] invoked 1 times for total of 19ms. (Avg. 19ms/invocation)

Execution time: 2539ms

Optimize reports

Report execution setup

Imports execution setup

Total time
**Tuning up**

**Child process**
Creates a separate JVM child process to Jira JVM

**Child process JVM memory**
Make sure you have enough physical memory and designate resources accordingly. Increase when necessary

**Query timeout**
Increase query timeout if you have enough memory only. Make small increase.

**Parallel query threads**
Increase parallel query threads if your Jira server has enough CPU capacity to serve users running reports simultaneously.
**Tuning up**

**Frequency**
Impacts the total import time the most. Set minimal frequency to limit this if needed

**Allowed / disabled time**
Allow or disable when imports could be run. For example, disable regular imports during working hours, enable manual imports only

**Parallel imports**
Increase the size of parallel imports if you have enough CPU capacity on Jira server

**Profiling** *
Enable if you have performance problems before sharing log files with us
Plan your accounts
Team accounts vs general accounts

Measure don’t guess
Profiling, usage statistics, repeat

Scale the infrastructure
Resources - memory, CPU, and settings accordingly

Scalability action plan
Questions?

community.eazybi.com
support@eazybi.com