

# Redmineの チケットをPower BIで グラフ化してみました

m\_oto

第24回Redmine勉強会 2023.6.3





# 概要

- **Power BI初心者**が、Redmineのチケット情報を取得してグラフ化するところまでを、簡単にご紹介します
  - issuesテーブルなどの列名と、チケットの各要素との対応関係は前提知識とします
- **ポンコツ設計のため、お仕事で使うときはご注意ください**
  - **プライバシーレベルを一番低く設定**する必要があります



# LT発表の動機

数年前(社内でデータ可視化ツールの技術営業をしていた頃)

- 顧客から**チケットの滞留日数**をグラフで見たいという要望をいただく
- 勉強がてら**プラグイン**を作れないかトライしたが、すぐに挫折
- **View Customize plugin**を使うと、画面には表示されるがDBIには格納されないためグラフ化できず

最近、別件でPower BIを使うことがあり、昔のことを思い出して再びトライ

- 手順を詳しく書いてある**親切なWebサイト**のおかげで、できた！
- 忘れないうちに、作り方を残しておこう
- 他にも知りたい人がいるかもしれない？
- せっかくなので、**憧れのRedmine勉強会**でLT発表しよう！ ←今ココ



# 感謝

モダン Excel を業務で使うブログ (旧Road to Cloud Office)

Shigeru Numaguchiさま

- [\[Power BI / Excel\] 複数にまたがる Web ページからデータを取得する～モダン Excel を業務で使うブログ \(旧Road to Cloud Office\) \(road2cloudoffice.blogspot.com\)](https://road2cloudoffice.blogspot.com)
- [\[Power Query\] for ループのような処理を Power Query で行う方法～モダン Excel を業務で使うブログ \(旧Road to Cloud Office\) \(road2cloudoffice.blogspot.com\)](https://road2cloudoffice.blogspot.com)

# Power BIとは

- ローコードのBIツール
  - ほぼマウス操作で、データ取得からグラフ化まで行える
- Power Queryが入っていて、データソースからデータを取得・整形・加工できる



- 事例:[TOKYO予算見える化ボード](#)  
東京都財務局ではPower BIを使って、財政収支などのダッシュボードを公開している



## ポイント

1. REST APIでの**ページング機能**を、**ローコード**で解決
2. チケットの**滞留日数**(&解決日数)も、**ローコード**で解決



## 簡単な手順

1. 1回だけREST APIを使って、**チケットの総数**を取得しておく
2. offsetの値を受け取って、チケット情報を出力とする**カスタム関数**を作る
3. offsetの値が、0、100、200、・・・、チケット総数を取り込めるまで縦一列に並んだ**テーブル**を作る
4. offsetのテーブルに1列追加して、**カスタム関数**を呼び出す
5. **テーブル列を展開**する
6. チケットの「**滞留日数・解決日数**」列を追加する

# REST APIでのページング機能

- Redmineでは、URLの/issuesの後ろに**limit**や**offset**を付け足して、**offset**の値を**100ずつ増やして**繰り返すことで、チケットをすべて取得できる

`https://redmine.org/projects/redmine/issues.json?status_id=* & limit=100 & offset= 0`

`https://redmine.org/projects/redmine/issues.json?status_id=* & limit=100 & offset= 100`

`https://redmine.org/projects/redmine/issues.json?status_id=* & limit=100 & offset= 200`

`https://redmine.org/projects/redmine/issues.json?status_id=* & limit=100 & offset= ...`

↑ここは**共通**した文字列



# REST APIでのページング機能

- offsetの値を入力とした**カスタム関数**を作る

## カスタム関数(入力:offsetの値)

- **共通した文字列**と**入力された文字列**をくっつけて、RedmineのURLを以下のように設定する

`"https://redmine.org/projects/redmine/issues.json?status_id=*&limit=100&offset="& offsetの値"`

- このURLで**Webアクセス**する
  - 得られたチケット情報を出力とする

# REST APIでのページング機能

- offsetの値でテーブルを作る
  - チケット総数は、最初にREST APIを1回使ってtotal\_countの値をGETしておく

offset
0
100
...
total_count

正確には、  
 $\text{Round}(\text{total\_count} / 100) \times 100$   
ですが、便宜上、total\_countと表現しておきます

# REST APIでのページング機能

- offsetのテーブルに**1列追加**して、ここに**カスタム関数**を呼び出す
  - セルごとにカスタム関数が適用される

offset	追加された列 (offsetの値を入力としてカスタム関数が適用される)
0	<a href="https://redmine.org/projects/redmine/issues.json?status_id=*&amp;limit=100&amp;offset=0">https://redmine.org/projects/redmine/issues.json?status_id=*&amp;limit=100&amp;offset= 0</a>
100	<a href="https://redmine.org/projects/redmine/issues.json?status_id=*&amp;limit=100&amp;offset=100">https://redmine.org/projects/redmine/issues.json?status_id=*&amp;limit=100&amp;offset= 100</a>
...	<a href="https://redmine.org/projects/redmine/issues.json?status_id=*&amp;limit=100&amp;offset=...">https://redmine.org/projects/redmine/issues.json?status_id=*&amp;limit=100&amp;offset= ...</a>
total_count	<a href="https://redmine.org/projects/redmine/issues.json?status_id=*&amp;limit=100&amp;offset=total_count">https://redmine.org/projects/redmine/issues.json?status_id=*&amp;limit=100&amp;offset= total_count</a>

↑追加された列の各セルの中に、Webアクセスして得られたチケット情報が  
テーブルに入っている(テーブル列)

# REST APIでのページング機能

- テーブル列を展開して、1つの列に含まれる複数の値を、別々の列に展開する

	ABC 123 cur_offset	ABC 123 hyou
1		0 Table
2		100 Table

このボタンをクリック

↓ 何度か展開を繰り返す

	ABC 123 id	ABC 123 project.id	ABC 123 project.name	ABC 123 tracker.id	ABC 123 tracker.name
1		38614	1 Redmine		3 Patch
2		38585	1 Redmine		2 Feature



## チケットの滞留日数(解決日数)

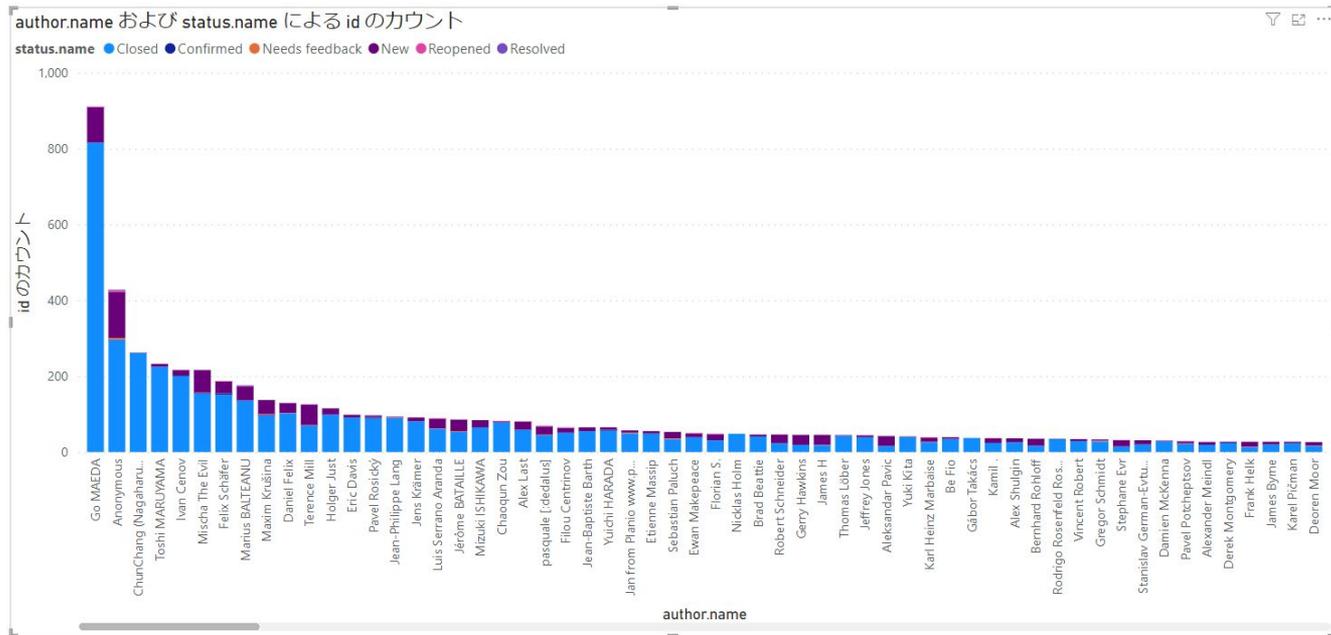
チケットの**滞留日数**、もしくはクローズまでにかかった日数(**解決日数**)を、以下のよう  
に設定する

- チケットが**進行中**(closed\_onがnull)のとき、
  - 今日の日付 - 作成日 = 滞留日数
- チケットが**クローズ**している(closed\_onに値がある)とき、
  - 終了日 - 作成日 = 解決日数

テーブルに1列追加して、上の式を入力するだけ

# グラフ

作成者ごとのチケット数を、ステータスごとに積み上げた



メモ：author.nameを横軸にすると、同じ名前の作成者がいた場合、チケットはまとめてカウントされます。

ビジュアルのビルド

検索

- ticket\_info
- ticket\_table
  - assigned\_to.id
  - assigned\_to.name
  - author.id
  - author.name
  - category.id
  - category.name
  - closed\_on
  - created\_on
  - custom\_fields
  - description
  - done\_ratio
  - due\_date
  - estimated\_hours
  - fixed\_version.id
  - fixed\_version.name
  - id
  - is\_private
  - priority.id
  - priority.name
  - project.id
  - project.name
  - start\_date
  - status.id
  - status.name
  - subject
  - tracker.id
  - tracker.name
  - updated\_on
  - 演習日数 or 解決...
  - 演習日数 or 解決...

X 軸

author.name

Y 軸

idのカウンント

凡例

status.name

ここにデータフィールド...

ここにデータフィールド...

ドリルスルー

クロスレポート

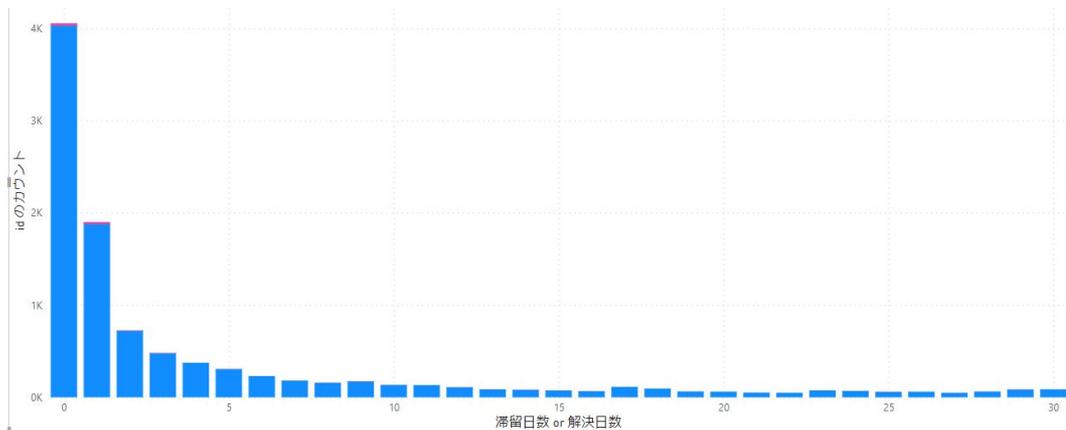
すべてのフィルターを保持する

ドリルスルー フィールド...



# グラフ

滞留日数・解決日数ごとのチケット数を、  
ステータスごとに積み上げた  
#ロングテールなので0~30日間でフィルタしている



Visual configuration interface for a dashboard chart.

- フィルター** (Filter): Search bar, filter list (idのカウント, status.name), filter type (高度なフィルター処理), and filter rules (AND/OR, 次値以上/以下).
- 視覚化** (Visualization): Visual build tools, chart type selection (Bar chart highlighted), X-axis (滞留日数 or 解決日数), Y-axis (idのカウント), legend (status.name), and styling options (Small multiple, Drill down, Cross-report).
- データ** (Data): Search bar, data source (ticket\_info, ticket\_table), and field selection (assigned\_to.id, status.name, etc.).

Red arrows point from the filter rules and visualization settings to the corresponding elements in the chart.

# おまけ

## チケット

▼ フィルタ

作成者 等しい ▼ Go MAEDA

ステータス 等しい ▼ New

→ オプション

✓ 適用 🔄 クリア

<input type="checkbox"/>	#	ステータス	
<input type="checkbox"/>	38636	New	NoMethodError when no default priority is s
<input type="checkbox"/>	38526	New	Subversion support fo
<input type="checkbox"/>	38504	New	Display pasted images

◀ 前 1 2 3 4 次 ▶ (1-25/92) 1ペー

author.name および status.name (こ

status.name ● Closed ● Confirmed ● Needs fe

author.name Go MAEDA  
status.name New  
id のカウント 93

1枚多い!

Go MAEDA

- ログインID: maeda
- 登録日: 2008-02-03
- 最終接続日: 2008-02-03

Go MAEDA

- ログインID: rfuser073
- 登録日: 2008-02-03

2人分カウントしていた!

補足





## もう少し詳しい手順

1. RedmineのURLをパラメーターにする
2. 1回だけREST APIを使って、チケットの総数を取得する
3. offsetを入力とするカスタム関数を作る
4. offsetの値が、0、100、200・・・チケット全部を取り込めるまで並んだテーブルを作る
5. offsetのテーブルにカスタム関数を呼び出す
6. テーブルを展開する&データ型を設定する
7. チケットの「滞留日数・解決日数」列を追加する
8. 認証とプライバシーレベルの設定

- Power Queryエディターを起動する
- パラメーターの管理>新しいパラメーターをクリックする
  - 名前は任意
  - 入力は最後、/issues の形になるよう説明に書く



### パラメーターの管理

新規作成

A<sup>B</sup>C Redmine\_URL ×

名前  
Redmine\_URL

説明  
https://servername/some-path/issues の形で入力してください。

必須

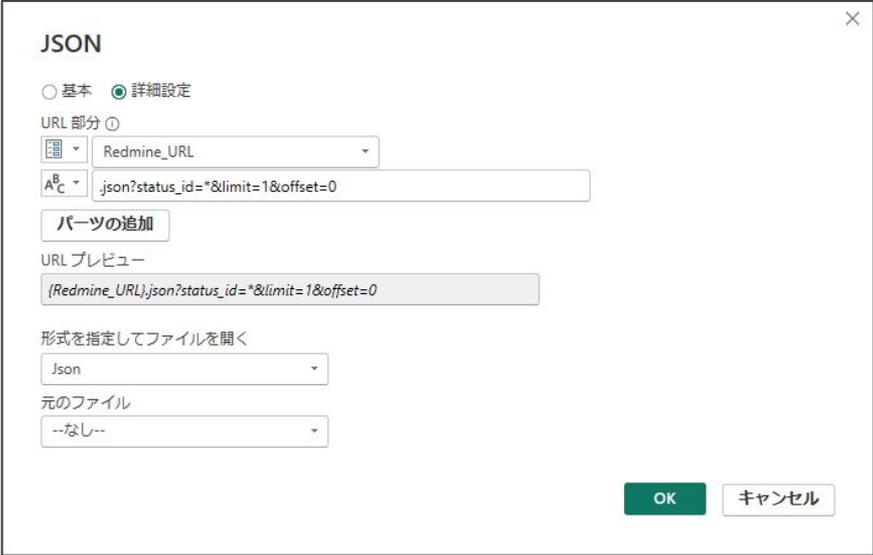
種類  
テキスト

提案された値  
任意の値

現在の値  
https://redmine.org/projects/redmine/issues

RedmineのURLをパラメーターにする

- 新しいソース>Webをクリックする
- 詳細設定を選択する
- URL部分の下のアイコンをクリックして、パラメーターを選択する
  - Redmine\_URL
- テキストは以下を入力する
  - `.json?status_id=*&limit=1&offset=0`
- OKを押すと、チケットがテーブルで表示される



JSON

基本  詳細設定

URL部分 ①

パーツの追加

URLプレビュー

`{Redmine_URL}.json?status_id=*&limit=1&offset=0`

形式を指定してファイルを開く

元のファイル

OK キャンセル

1回だけREST APIを使って、チケットの総数を取得する

- テーブルに名前をつける
  - 例:ticket\_info
- 「適用したステップ」の「テーブルに変換済み」まで残して、それ以外は下のステップから削除する
- 終わると、以下ようになる

クエリ [2] < 123 issues total\_count 123 offset 123 limit

ABC	ABC	ABC	ABC
123	123	123	123
1	List	119	0

チケットの総数

クエリ名: ticket\_info

適用したステップ: テーブルに変換済み

適用したステップ

- ソース
- テーブルに変換済み
- 展開された issues
- 展開された issues.1
- 展開された issues.project
- 展開された issues.tracker
- 展開された issues.status
- 展開された issues.priority
- 展開された issues.author
- 展開された issues.assigned\_to
- 展開された issues.fixed\_version
- 展開された issues.custom\_fiel...
- 展開された issues.custom\_fiel...
- 変更された型

1回だけREST APIを使って、チケットの総数を取得する

- ticket\_infoテーブルの上で右クリック>複製を選択する
- 新しくできたテーブルの名前を変える
  - 例: offset\_func
- offset\_funcを選択したまま、「ホーム>詳細エディター」を開く
  - 1行目を追加してoffsetの値を入力パラメーターにして、3行目の後ろでoffsetの値を代入するように変更
  - 完了を押すと、テーブルが **カスタム関数** に変わる

(cur\_offset as number) =>

let

```
ソース = Json.Document(Web.Contents(Redmine_URL & ".json?status_id=*&limit=100&offset=" & Number.ToText(cur_offset))),  
テーブルに変換済み = Table.FromRecords({ソース})
```

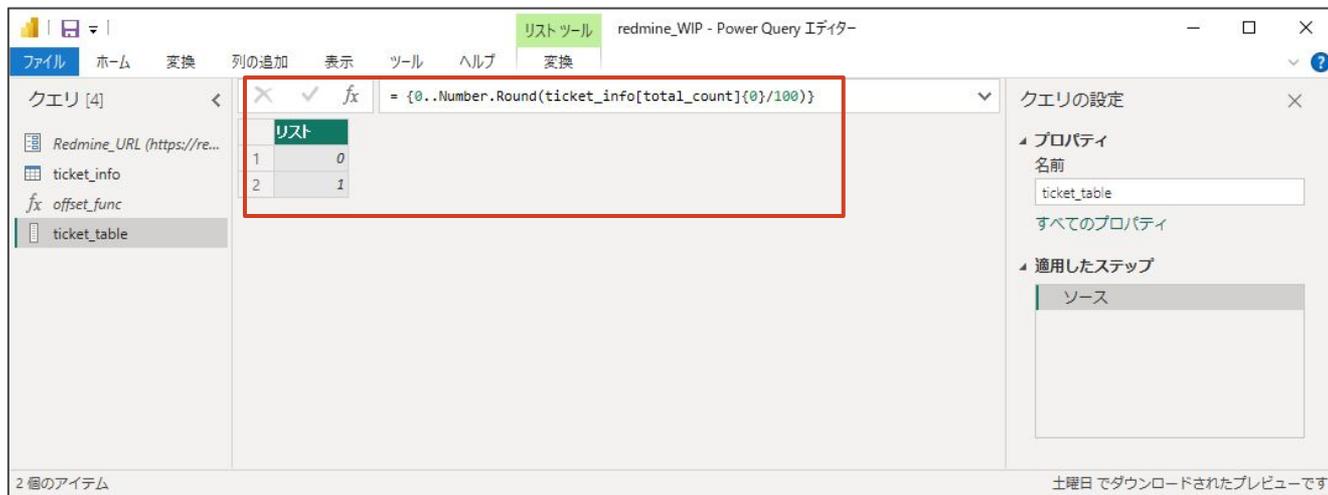
in

```
テーブルに変換済み
```

offsetを入力とするカスタム関数を作る



- ホーム > 新しいソース > 空のクエリをクリックする
- クエリの名前を変える
  - 例: ticket\_table
- 数式バーに以下を入力する
  - = {0..Number.Round(ticket\_info[total\_count]{0}/100)}



redmine\_WIP - Power Query エディター

クエリ [4]

Redmine\_URL (https://re...)

ticket\_info

fx offset\_func

ticket\_table

fx = {0..Number.Round(ticket\_info[total\_count]{0}/100)}

リスト	
1	0
2	1

クエリの設定

プロパティ

名前

ticket\_table

すべてのプロパティ

適用したステップ

ソース

2個のアイテム

土曜日 でダウンロードされたプレビューです

offsetの値が、0、100、200・・・チケット全部を取り込めるまで並んだテーブルを作る

- ticket\_tableを選択したまま、リストツールと書いてあるほうの「変換>テーブルへの変換」
  - オプションはデフォルトのままで OK
- 列の追加>カスタム列
  - 列名(例) cur\_offset
  - カスタム列の式: [Column1]\*100
- Column1列を選択して、右クリック>削除

The screenshot shows the Power Query Editor window for 'redmine\_WIP'. The main area displays a table with the following data:

ABC	cur_offset
123	0
1	100
2	

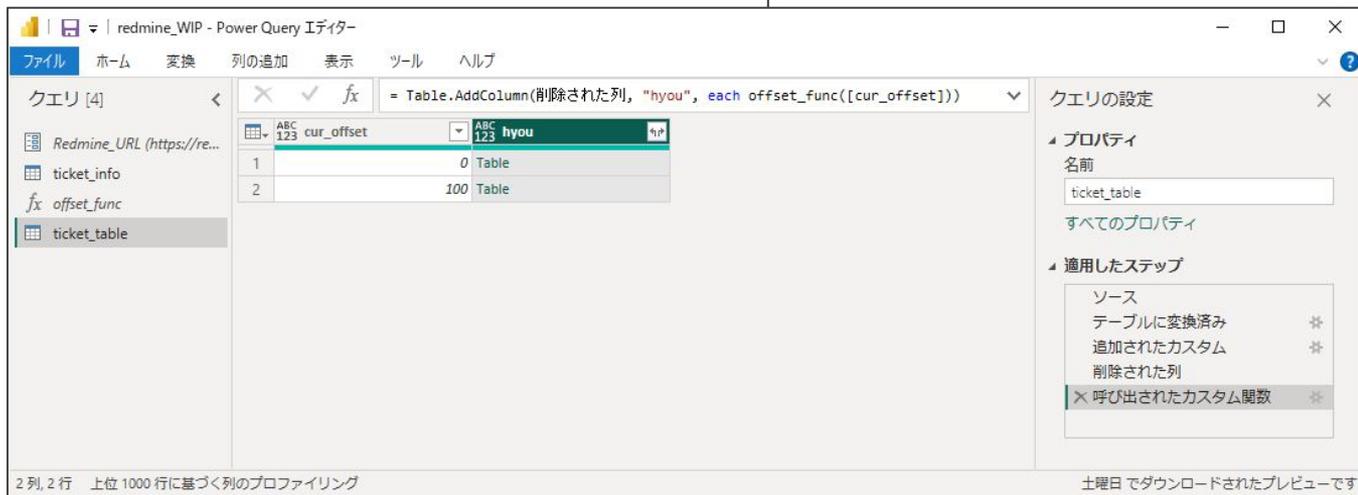
The formula bar shows the expression: `= Table.RemoveColumns(追加されたカスタム, {"Column1"})`. The 'Applied Steps' pane on the right shows the following steps:

- ソース
- テーブルに変換済み \*
- 追加されたカスタム \*
- 削除された列

The status bar at the bottom indicates '1列, 2行 上位 1000 行に基づく列のプロファイリング' and '土曜日 でダウンロードされたプレビューです'.

offsetの値が、0、100、200・・・チケット全部を取り込めるまで並んだテーブルを作る

- 列の追加 > カスタム関数の呼び出し
  - 新しい列名(例) hyou
  - 関数クエリ: offset\_func
  - 適用先の列は、cur\_offset
- OKを押すと以下ようになる



offsetのテーブルにカスタム関数を呼び出す

- hyouの列名にある展開ボタンをクリックして、そのままKを押す
- hyou.issuesの展開ボタンをクリックして、「新しい行に展開する」を選択する

ABC 123	cur_offset	ABC 123	hyou	展開
1	0	Table		
2	100	Table		

このボタンをクリック

ABC 123	cur_offset	ABC 123	hyou.issues	展開	ABC 123	hyou.total_count
1	0	List			119	
2	100	List			119	

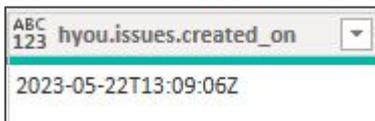
ABC 123	cur_offset	ABC 123	hyou.issues	展開	ABC 123	hyou.total_count
1	0	Record			119	
2	0	Record			119	
3	0	Record			119	

1行が1チケットになった

ABC 123	cur_offset	ABC 123	hyou.issues.id	ABC 123	hyou.issues.project	ABC 123	hyou.issues.tracker
1	0	1478	Record		Record		

テーブルを展開する

- 同様に、tracker、project、status、priority、author、category、assigned\_to、fixed\_version列も展開する
- Ctrlキーを押しながら、start\_date、due\_date、created\_on、updated\_on、closed\_on列をクリックして、変換>日付>解析を選択する



文字列型 ⇒ 日付型

テーブルを展開する&データ型を設定する

- 列の追加 > 条件列を選択
  - 新しい列名 (例) 滞留日数 or 解決日数
  - 条件列は以下のように設定する

列名	演算子	値	出力
closed_on	指定の値に 等しい	null	=Duration.Days(Date.From(DateTime.LocalNow()) - [created_on])

それ以外の場合
=Duration.Days([closed_on] - [created_on])

注:[ ]で囲まれている列名は、展開したテーブル名や順番によって違ってくる。  
列名をプルダウンして表示されるのが正確な列名となるので、それを入力する。  
例)hyou.issues.created\_on

チケットの「滞留日数・解決日数」列を追加する

- 以下の設定で、滞留日数 or 解決日数列が追加できたら「ホーム>閉じて適用」で詳細エディターを閉じる

条件列の追加

他の列または値から計算される、条件列を追加します。

新しい列名  
滞留日数 or 解決日数

条件	列名	演算子	値	出力
...	hyou.issues.close...	指定の値に等しい	ABC 123	null
句の逆	hyou.issues.tracker.name			
	hyou.issues.status			
	hyou.issues.priority			
	hyou.issues.author			
	hyou.issues.assigned_to			
それ以外	hyou.issues.fixed_version			
ABC 123	hyou.issues.subject			
	hyou.issues.description			
	hyou.issues.start_date			
	hyou.issues.due_date			
	hyou.issues.done_ratio			
	hyou.issues.is_private			
	hyou.issues.estimated_hours			
	hyou.issues.total_estimated_hours			
	hyou.issues.spent_hours			
	hyou.issues.total_spent_hours			
	hyou.issues.custom_fields			
	hyou.issues.created_on			
	hyou.issues.updated_on			
	hyou.issues.closed_on			

結果 ABC 123 >= [hyou.issues.created\_on]

OK キャンセル

チケットの「滞留日数・解決日数」列を追加する

- 「ファイル＞オプションと設定＞データソース設定＞アクセス許可」で編集を選択
  - 資格情報の編集ボタンをクリック
  - 認証情報を書く
- 「ファイル＞オプションと設定＞オプション＞グローバル＞プライバシー」で、プライバシーレベル設定を無視にしてOKを押す
- 「ホーム＞更新」でデータ取得する
  - エラーが出なければOK

(グラフ化の手順は省略)



認証とプライバシーレベルの設定

