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DB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

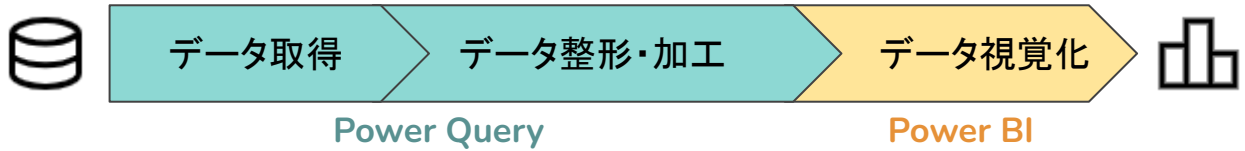
感謝

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)
- [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)

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、
Round（total_count／100）×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

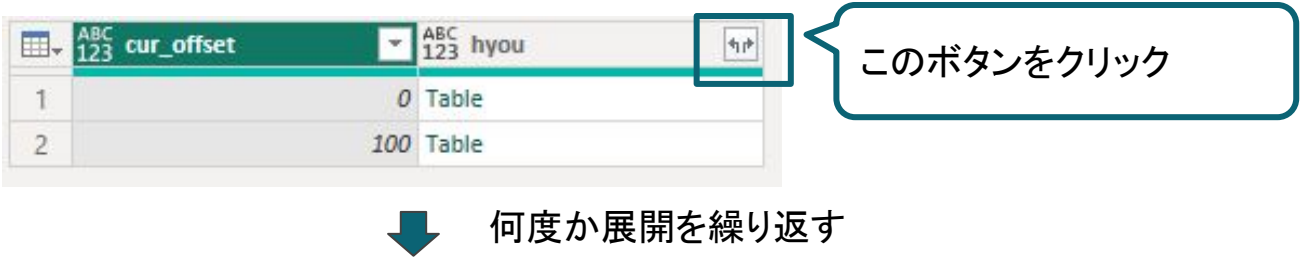
- offsetのテーブルに**1列追加**して、ここに**カスタム関数**を呼び出す
  - セルごとにカスタム関数が適用される

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

↑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
テーブルで入っている（**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



	id	project.id	project.name	tracker.id	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）

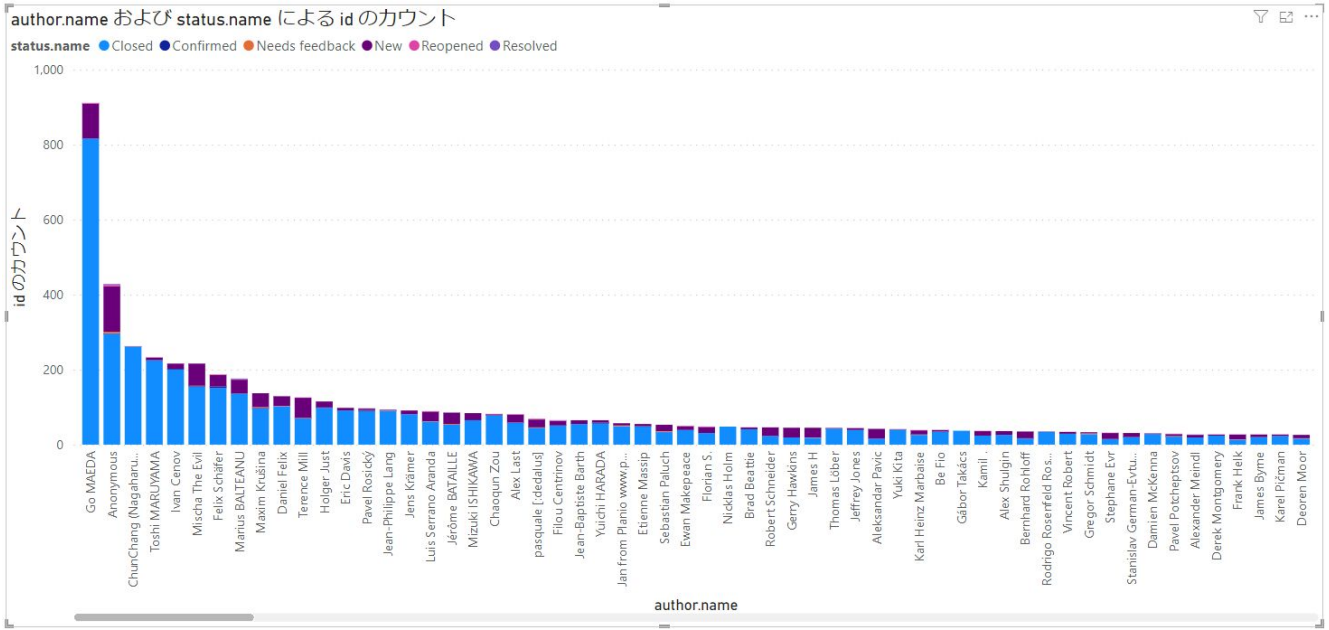
チケットの**滞留日数**、もしくはクローズまでにかかった日数（**解決日数**）を、以下のように設定する

- チケットが**進行中**（closed_onがnull）のとき、
  - 今日の日付ー作成日＝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、
  - 終了日ー作成日＝解決日数

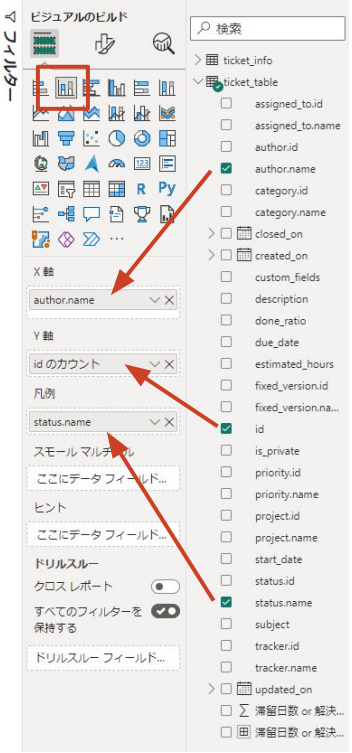
テーブルに1列追加して、上の式を入力するだけ

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

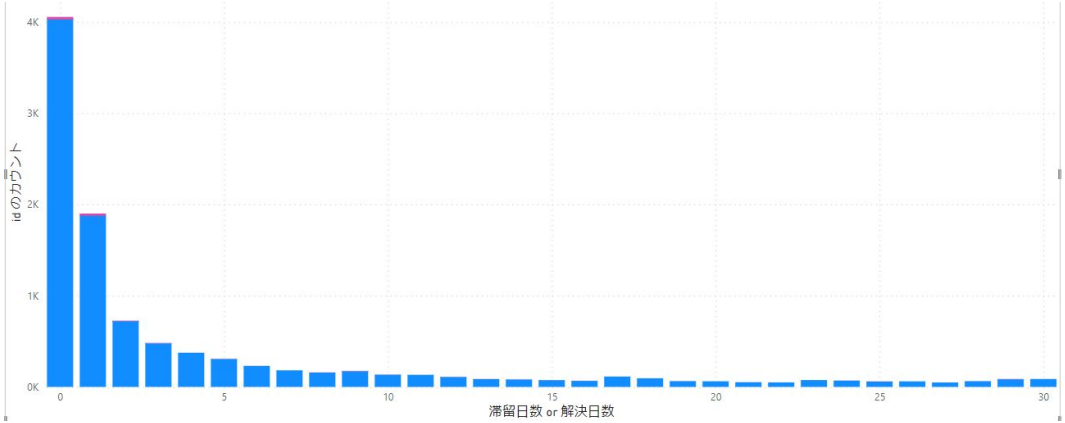


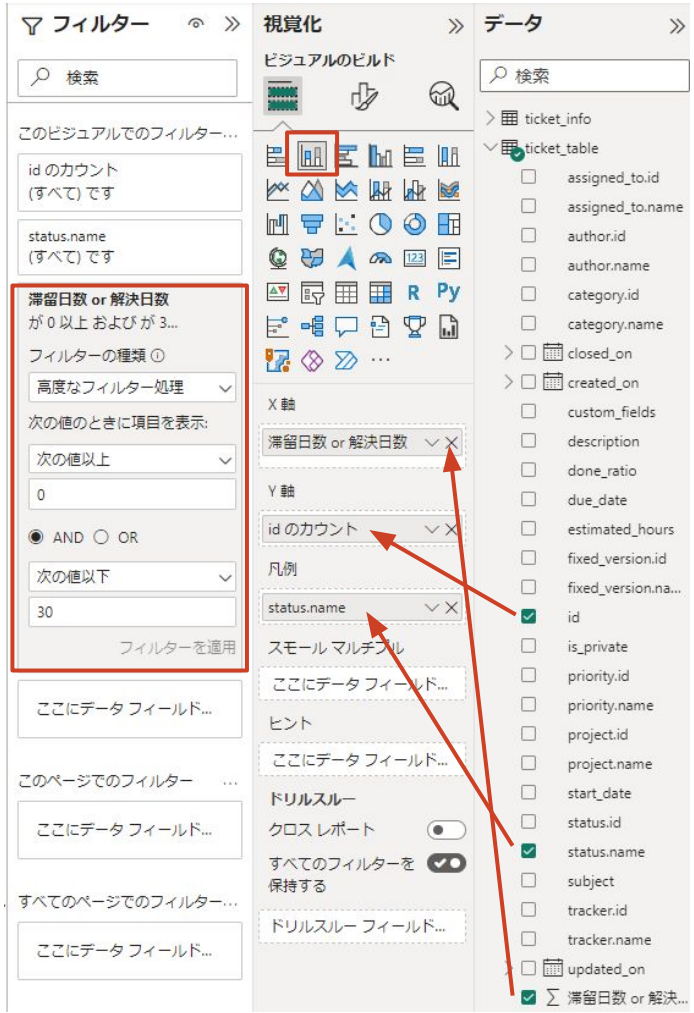
メモ：author.nameを横軸にすると、同じ名前の作成者がいた場合、チケットはまとめてカウントされます。



グラフ

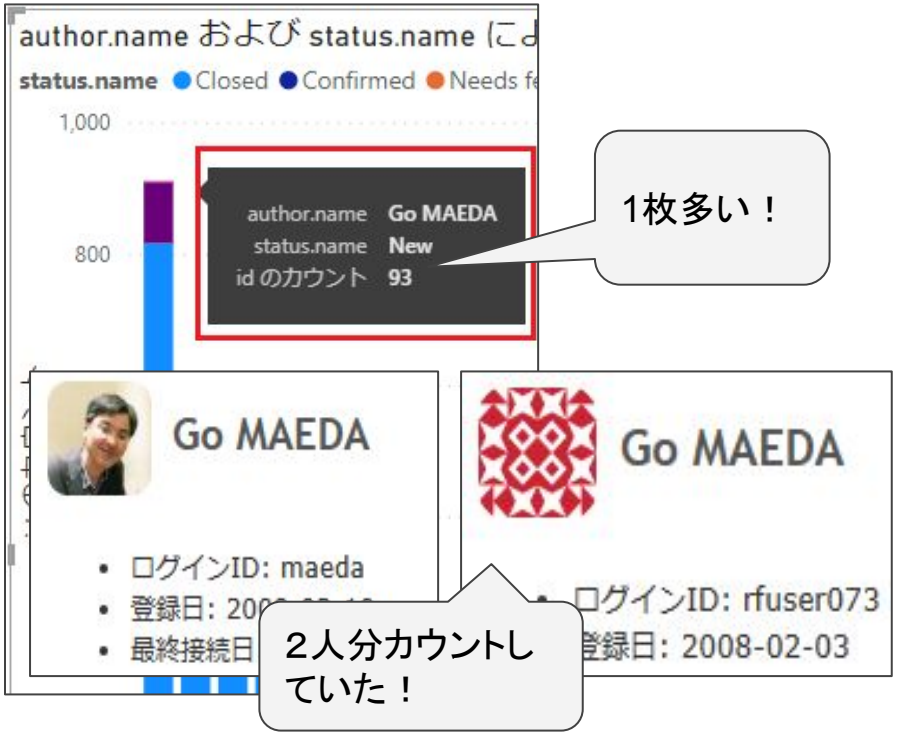
滞留日数・解決日数ごとのチケット数を、
ステータスごとに積み上げた
#ロングテールなので0～30日間でフィルタしている





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
ステータス
38636 New NoMethodError when
no default priority is s
38526 New Subversion support fo
38504 New Display pasted images
« 前 1 2 3 4 次 » (1-25/92) 1ペー


author.name および status.name によ
status.name Closed Confirmed Needs fe
1,000
800
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: 200
最終接続日
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

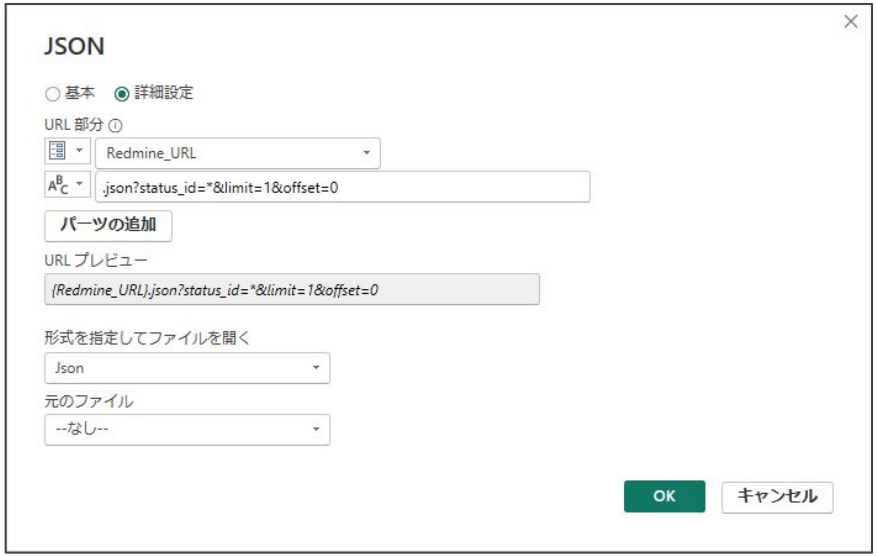
もう少し詳しい手順

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

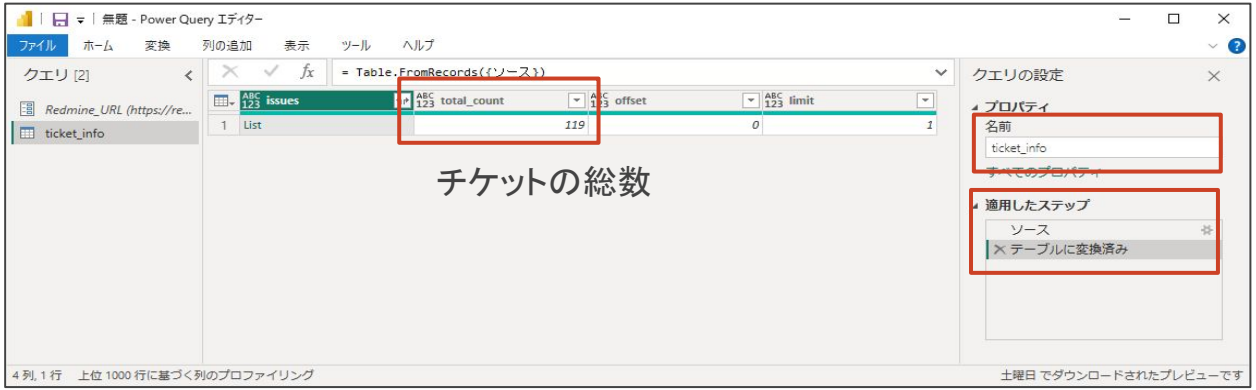
RedmineのURLをパラメーターにする

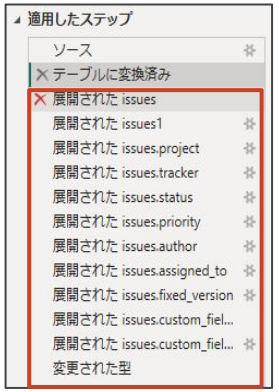
- 新しいソース＞Webをクリックする
- 詳細設定を選択する
- URL部分の下のアイコンをクリックして、パラメーターを選択する
  - Redmine_URL
- テキストは以下を入力する
  - .json?status_id=*&limit=1&offset=0
- OKを押すと、チケットがテーブルで表示される



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、以下のようになる



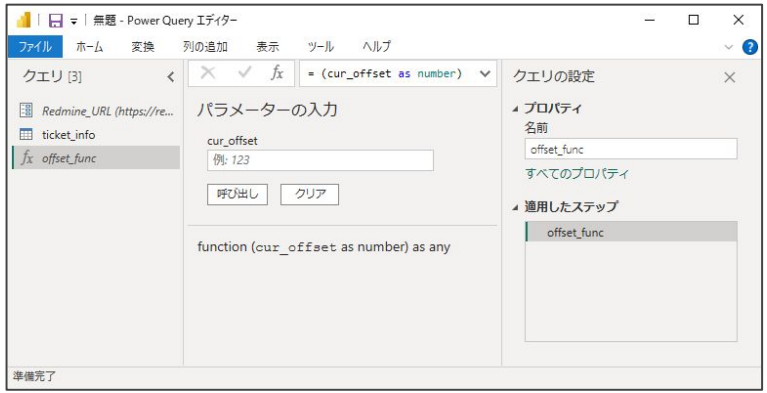


1回だけREST APIを使って、チケットの総数を取得する

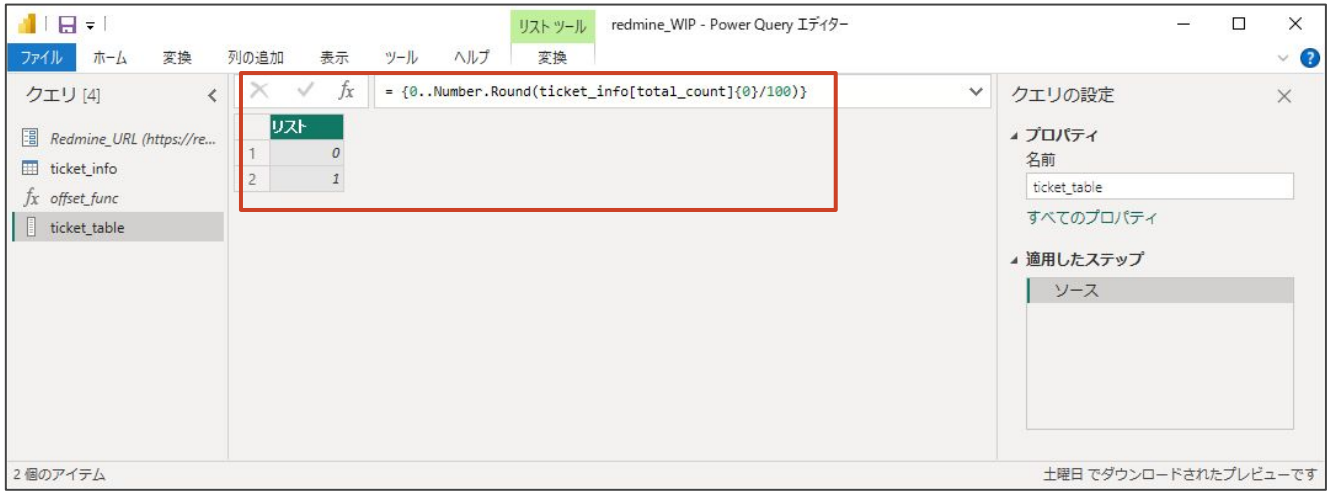
- ticket_infoテーブルの上で右クリック＞複製を選択する
- 新しくできたテーブルの名前を変える
  - 例：offset_func
- offset_funcを選択したまま、「ホーム＞詳細エディター」を開く
  - 1行目を追加してoffsetの値を入力パラメーターにして、3行目の後ろでoffsetの値を代入するように変更
  - 完了を押すと、テーブルが **カスタム関数** に変わる

```
(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

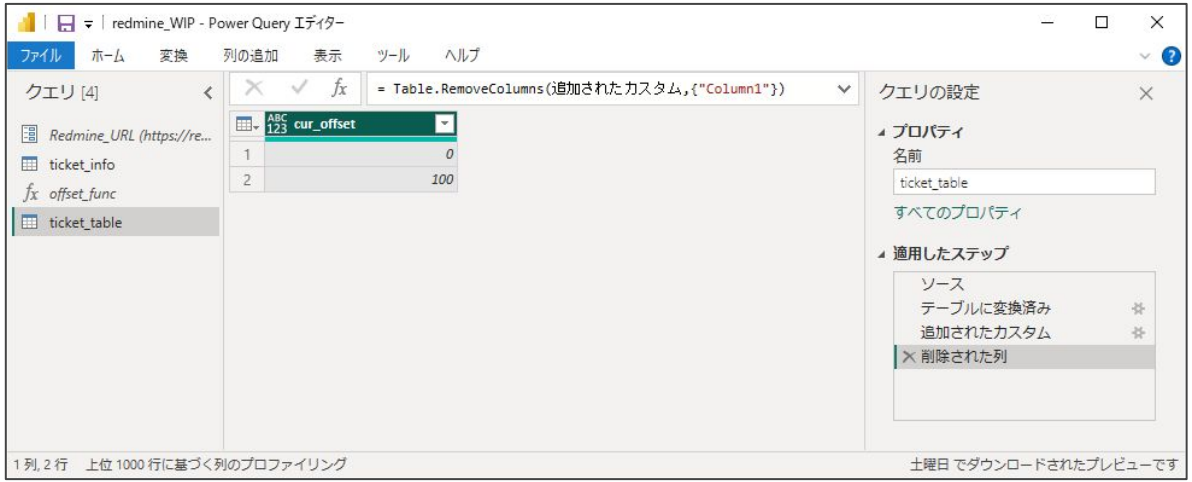


- ホーム＞新しいソース＞空のクエリをクリックする
- クエリの名前を変える
  - 例：ticket_table
- 数式バーに以下を入力する
  - = {0..Number.Round(ticket_info[total_count]{0}/100)}



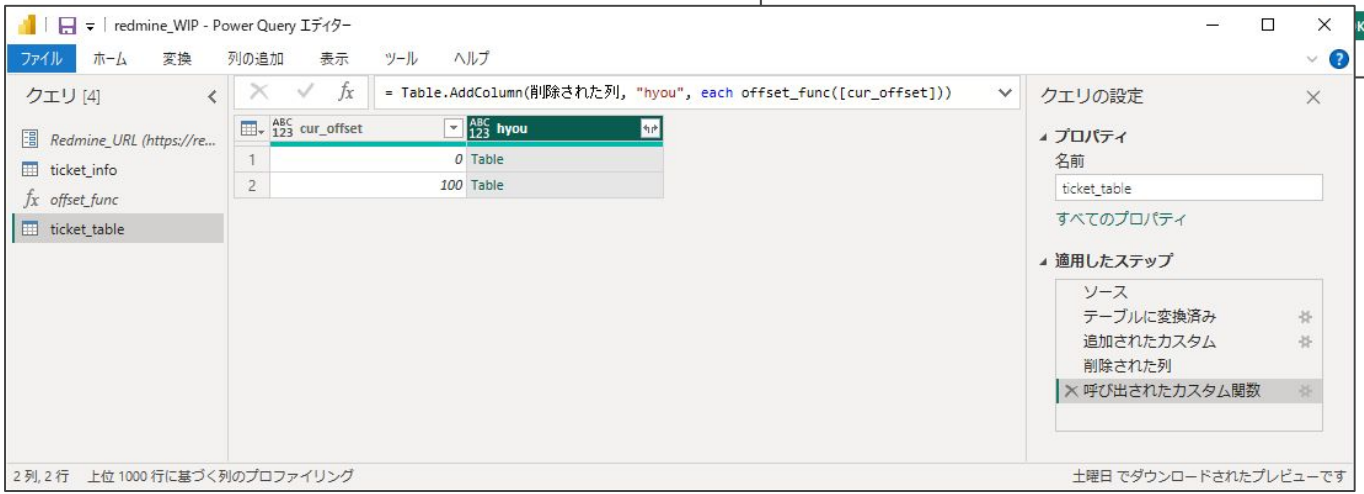
offsetの値が、0、100、200・・・チケット全部を取り込めるまで並んだテーブルを作る

- 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



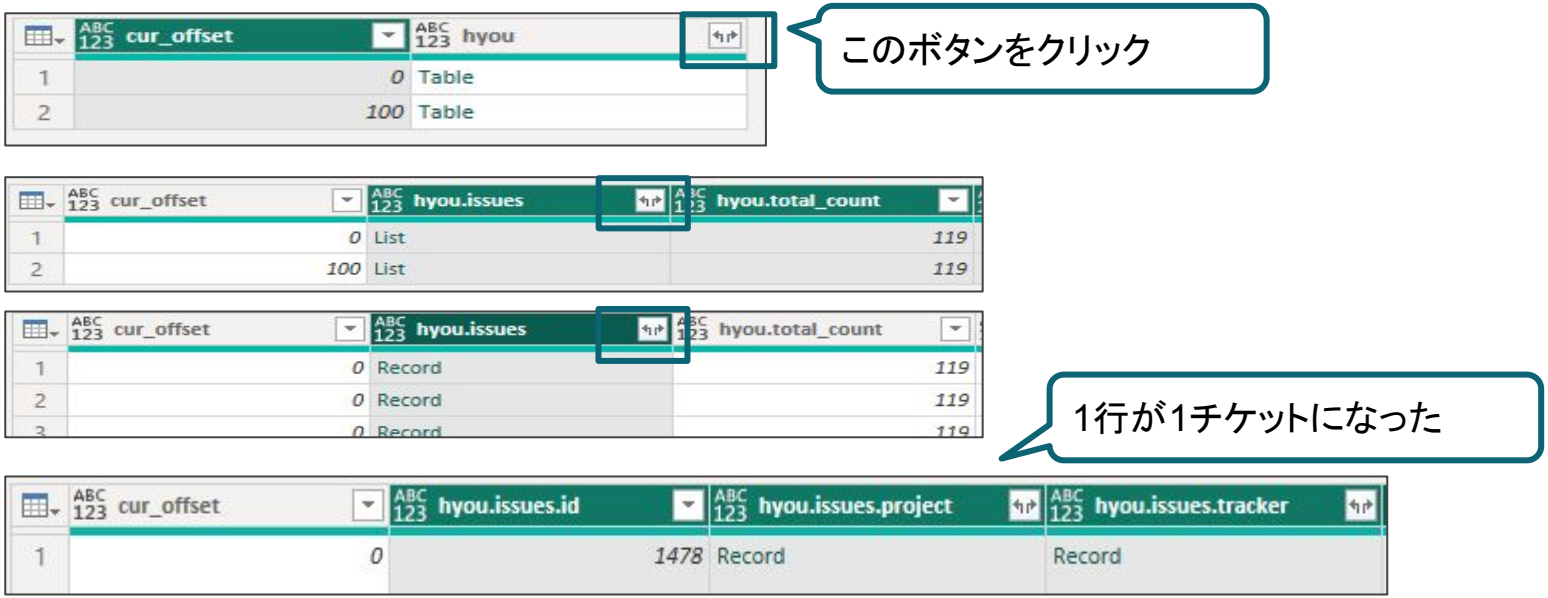
offsetの値が、0、100、200・・・チケット全部を取り込めるまで並んだテーブルを作る

- 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を押すと以下のようになる



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OK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



	cur_offset	hyou
1	0	Table
2	100	Table

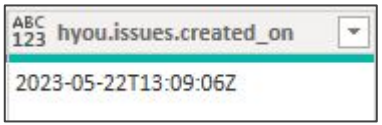
	cur_offset	hyou.issues	hyou.total_count
1	0	List	119
2	100	List	119

	cur_offset	hyou.issues	hyou.total_count
1	0	Record	119
2	0	Record	119
3	0	Record	119

	cur_offset	hyou.issues.id	hyou.issues.project	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

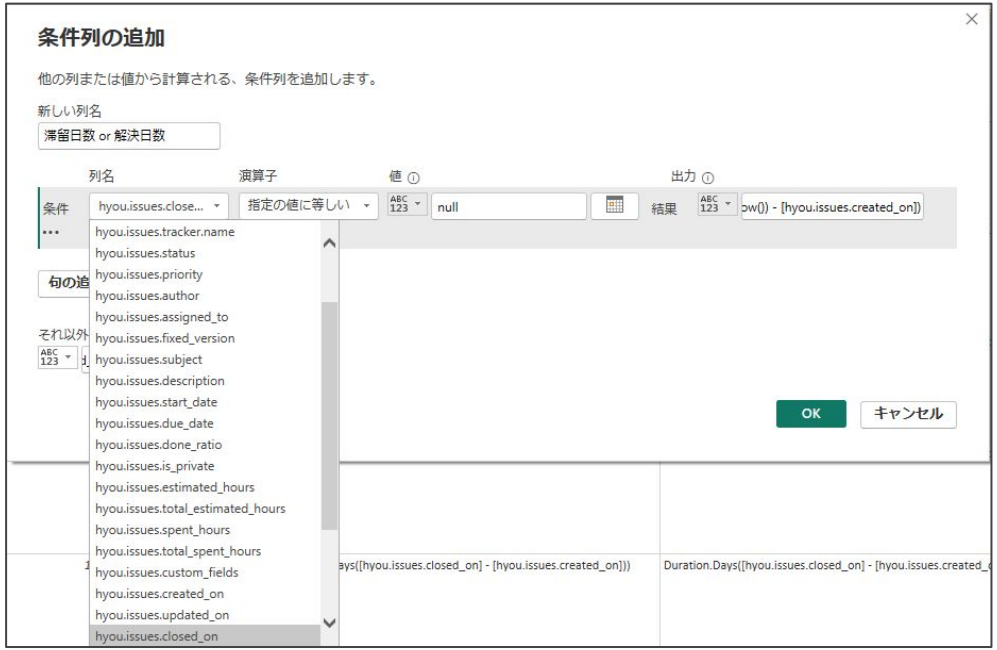
列名	演算子	値	出力
closed_on	指定の値に等しい	null	=Duration.Days(Date.From(DateTime.LocalNow()) - [**created_on**])

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



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）