Relational Algebra for Excel 3.0

matti@belle-nuit.com 21.8.2022

Introduction

- Relational Algebra for Excel is a collection of user-defined functions that let you perform calculations with relationships or, in other words, use Excel as a database.

- You can use these functions to query data in Excel spreadsheets with the same expressive power as query languages like SQL.

- With version 3.0 you can even type SQL queries directly.

- The function can handle tables with 500-4000 rows.

Why use it?

- Excel provides filters for data that are powerful but not persistent. You lose a query when you make the next query. Also, you can only search in one table.

- Pivot tables can combine data from multiple tables, but they are neither intuitive, flexible, nor persistent.

- There are SQL plugins, but they work like macro commands, may need DLL and are static. Relational Algebra for Excel uses functions; query results are updated dynamically as you edit cells.

Installation

- All the VBA code in the Excel file is in a standalone module. You can copy this module into any Excel file you use. Or you can replace the tabs in the file with your own tabs.

- You will need to save your sheet as an Excel sheet with macros (XSLM) and enable the macros to use it.

- Once installed, you can use the functions. They all have the prefix "rel".

Set theory

- Relational Algebra has evolved from set theory, which you may have learned about in school.
- A set is a collection of zero or more elements, where each element is unique.
- S = {A, B} is a set with the elements A and B.
- A ε S A is an element of S
- {A} ⊂ {A,B} is a subset
- {} or Ø is an empty set.

Relation

- A **relation** is a set of zero or more tuples that have the same properties.

- The **cardinality** of a relation is the number of tuples. The empty relation {} or ∅ has no tuples and cardinality 0

- A **tuple** is a set of zero or more property-value pairs. Each property has its domain. A domain is the set of all possible values. For example, ℕ is a domain.

- **Arity** is the number of properties of the tuples in a relation. The properties have no particular order.

Tables

	A	B	C
1	id	title	country
2	1001	Ma vie de Courgette	CH
3	1002	Elle	FR
4	1003	Toni Erdmann	DE
5	1004	Above And Below	CH

- In Excel the tuples are the rows and the columns are the properties.
- The order of the rows and columns does not matter and each row is unique.
- Tables always have a column header.
- *Tip: Name the cell ranges before using the functions.* ***Films*** *is more readable than* ***$A1:$C5***

Internal representation of the relation

```
filmid::title::country
1001::Ma vie de Courgette::CH
1002::Elle::FR
1003::Toni Erdmann::DE
1004::Above And Below::CH
```

- Relational algebra works with relations and the result is always a relation.
- Relations are combined to a single string. It uses the separator "::" for the properties and space+newline for the tuple.
- Set cell wrap to see multiple rows.

Limitations

- All properties have the same domain: string. The “::"' and the tab cannot be used in a value because they act as separators.
- The property names must start with a letter and cannot contain spaces.
- Excel limitation: A relation in a cell cannot display more than 32K characters.
  Workaround: For longer results, a hash is displayed you can use as intermediate result for other functions.

Convert between relation and cells

- **relRange(range)** reads a range of cells into a relation.
- Most functions implicitly convert a cell range into a relation.
- **relCell(relation, row, column, isNumber, noError)** reads a single value from of a relation.
- **relCellArray(relation)** is used as array function and reads a relation into a cell array.
- **relFilter** can return directly a single value if the relation is a single column and a single row (instruction “C” and “Z”).

Use of the functions

You can work in three ways:

- Use the different functions (relSelect, relProject, relJoin) individually and combine them.
- Use **relFilter** as single function and pile all operators on a stack. relFilter handles the data volume better. The 32k limit applies only on the end result but not to the intermediate data.
- Use **relSql** to define directly an SQL query.
- Four: Combine any of these.

Union

fiction	doc	fiction ∪ doc
filmid::title::country 1001::Ma vie de Courgette::CH 1002::Elle::FR 1003::Torni Erdmann::DE	filmid::title::country 1004::Above And Below::CH	filmid::title::country 1001::Ma vie de Courgette::CH 1002::Elle::FR 1003::Torni Erdmann::DE 1004::Above And Below::CH

- = relUnion(fiction,doc)
- = relFilter(fiction, doc, "U")
- Both relations must have the same arity and the same properties.

Intersection

fiction	swissfilms	fiction ∩ swissfilms
filmid::title::country 1001::Ma vie de Courgette::CH 1002::Elle::FR 1003::Torni Erdmann::DE	filmid::title::country 1001::Ma vie de Courgette::CH 1004::Above And Below::CH	filmid::title::country 1001::Ma vie de Courgette::CH

- = relIntersect(fiction,swissfilms)
- = relFilter(fiction, swissfilms, "I")
- Both relations must have the same arity and the same properties.

Difference

films	swissfilms	films - swissfilms
filmid::title::country 1001::Ma vie de Courgette::CH 1002::Elle::FR 1003::Torni Erdmann::DE 1004::Above And Below::CH	filmid::title::country 1001::Ma vie de Courgette::CH 1004::Above And Below::CH	filmid::title::country 1002::Elle::FR 1003::Torni Erdmann::DE

- = relDifference(films,swissfilms)
- = relFilter(films, swissfilms, "D")
- Both relations must have the same arity and the same properties.

Selection

films	δ country="CH" films
filmid::title::country 1001::Ma vie de Courgette::CH 1002::Elle::FR 1003::Torni Erdmann::DE 1004::Above And Below::CH	filmid::title::country 1001:::Ma vie de Courgette::CH 1004::Above And Below::CH

- = relSelect(films,"$country=""CH""")
- = relFilter(films,"S $country=""CH""")
  Data type ad hoc: A column preceded by $ is used as string, preceded by % is used as number. Use double quotes when needed. Keep cell references outside the quoted text, so that they are updated.
- = relSql("SELECT filmid, title, country FROM t1 WHERE country = 'CH'",films)
  Data type is automatic based on context.
- Selection expressions can use any column, Excel formulas and cell references and must be evaluated to true or false.

Projection

films	π_{country} films
filmid::title::country 1001::Ma vie de Courgette::CH 1002::Elle::FR 1003::Torni Erdmann::DE 1004::Above And Below::CH	country CH FR DE

- = relProject(films,"country")
- = relFilter(films,"P country")
  Projection list can have multiple columns, separated by ::
- = relSql("SELECT country FROM t1", films)

Rename

films	$\delta_{\text{filmid isan}}$ films
filmid::title::country 1001::Ma vie de Courgette::CH 1002::Elle::FR 1003::Torni Erdmann::DE 1004::Above And Below::CH	isan::title::country 1001::Ma vie de Courgette::CH 1002::Elle::FR 1003::Torni Erdmann::DE 1004::Above And Below::CH

- = relRename(films,"filmid id")
- = relFilter(films,"R filmid isan")
  The rename operator will be important for joins. Multiple renames are possible separated by ::
- = relSql(SELECT filmid AS isan, title, country FROM t1", films)

Natural Join

films	theatres	films $\bowtie$ theatres
filmid::title::country	theatreid::theatre::filmid	filmid::title::country::theatreid::theatre
1001::Ma vie de Courgette::CH	21::Corso::1003	1001::Ma vie de Courgette::22::Apollo
1002::Elle::FR	22::Apollo::1001	1001::Ma vie de Courgette::23::Metropol
1003::Torni Erdmann::DE	23::Metropol::1001	1002::Elle::24::Le Paris
1004::Above And Below::CH	24::Le Paris::1002	1003::Toni Erdmann::21::Corso

- = relJoin(films,theatres,"NATURAL")
- = relFilter(films, theatres, "J NATURAL")
- = relSql("SELECT * FROM t1 NATURAL JOIN t2", films, theatres)
- Natural Join is based on common properties

Theta Join

films	theatres	films $\theta_{id = filmid}$ theatres
id::title::country	theatreid::theatre::filmid	id::title::country::theatreid::theatre::filmid
1001::Ma vie de Courgette::CH	21::Corso::1003	1001::Ma vie de Courgette::22::Apollo::1001
1002::Elle::FR	22::Apollo::1001	1001::Ma vie de Courgette::23::Metropol::1001
1003::Torni Erdmann::DE	23::Metropol::1001	1002::Elle::24::Le Paris::1002
1004::Above And Below::CH	24::Le Paris::1002	1003::Toni Erdmann::21::Corso::1003

- = relJoin(films,theatres,"%id=%filmic")

- = relFilter(films, theatres, "J %id=%filmid")

- = relSql("SELECT * FROM t1 JOIN t2 ON t1.id = t2.filmid", films, theatres)
  or
  = relSql("SELECT * FROM t1 JOIN t2 WHERE t1.id = t2.filmid", films, theatres)
  slower

- Theta Join allows any expression like the select expression

Cross Product

films	theatres	films × theatres
Id::title::country 1001::Ma vie de Courgette::CH 1002::Elle::FR 1003::Torni Erdmann::DE 1004::Above And Below::CH	theatreid::theatre::filmid 21::Corso::1003 22::Apollo::1001 23::Metropol::1001 24::Le Paris::1002	id::title::country::theatreid::theatre::filmid 1001::Ma vie de Courgette::21::Corso::1003 1001::Ma vie de Courgette::22::Apollo::1001 1001::Ma vie de Courgette::23::Metropol::1001 1001::Ma vie de Courgette::24::Le Paris::1002 1002::Elle::21::Corso::1003 1002::Elle::22::Apollo::1002 and 10 others

- = relJoin(films,theatres,"TRUE")
- = relFilter(films, theatres, "J TRUE")
- = relSql("SELECT * FROM t2 JOIN t2", films, theatres)

Other joins

- Left Join ⟕
- Right Join ⟖
- Outer Join
- Left Semi Join
- Right Semi Join
- Left Anti Semi Join
- Right Anti Semi Join

Aggregation (not relational)

films	
filmid::title::country 1001::Ma vie de Courgette::CH 1002::Elle::FR 1003::Torni Erdmann::DE 1004::Above And Below::CH	country::filmid_count CH::2 FR::1 DE::1

- = relProject(films,"country::filmid COUNT")
- = relFilter(films,"P country::filmid COUNT")
- = relSql("SELECT country, COUNT(filmid) AS filmid_count FROM t1", films)
  Explicite naming of an expression is mandatory
- Other aggregators: SUM, MIN, MAX, AVG, MEDIAN, STDEV

Order (not relational)

films	
filmid::title::country 1001::Ma vie de Courgette::CH 1002::Elle::FR 1003::Torni Erdmann::DE 1004::Above And Below::CH	filmid::title::country 1004::Above And Below::CH 1001::Ma vie de Courgette::CH 1003::Torni Erdmann::DE 1002::Elle::FR

- = relOrder(films,"country::title")

- = relFilter(films,"O country::title")
  Multiple columns are separated by ::
  Order can be specified with modifiers: A Z 1 9 for alphabetic or numeric, normal or reverse

- = relSql("SELECT * FROM t1 ORDER BY country, title", films)
  Modifiers ASC and DESC
  There is no numeric or alphabetic modifier: Text > number > empty

Limit (not relational)

films	
filmid::title::country 1001::Ma vie de Courgette::CH 1002::Elle::FR 1003::Torni Erdmann::DE 1004::Above And Below::CH	filmid::title::country 1002::Elle::FR 1003::Torni Erdmann::DE

- = relLimit(films,2,2)
- = relFilter(films,"L 2 2")
- *No statement in relSql*
- Order before you limit

Extend (not relational)

films	
filmid::title::country 1001::Ma vie de Courgette::CH 1002::Elle::FR 1003::Torni Erdmann::DE 1004::Above And Below::CH	filmid::title::country::sfid 1001::Ma vie de Courgette::CH::1 1002::Elle::FR::2 1003::Torni Erdmann::DE::3 1004::Above And Below::CH::4

- = relExtend(films,"sfid","%filmid - 1000")

- = relFilter(films,"E sfid %filmid - 1000")
  Extension expression can use any column, Excel formula and cell references and must evaluate to true or false. Data type ad hoc: A column preceded by $ is used as string, preceded by % is used as number. Use double quotes when needed. Keep cell references outside the quoted text, so that they are updated.

- = relSql ("SELECT filmid, title, country, (filmid-1000) AS sfid FROM t1", films)
  Explicite naming is mandatory for expressions.
  Numerical functions: ABS COS EXP INT LN LOG MOD POW ROUND SGN SIN SQRT TAN
  Text functions: LEFT LEN LOWER MID REPLACE RIGHT TRIM UPPER

Return single value (not relational)

films	
filmid::title::country 1001::Ma vie de Courgette::CH 1002::Elle::FR 1003::Torni Erdmann::DE 1004::Above And Below::CH	4

- = relFilter(films,"P filmid COUNT","Z")
- If the operators return a relation with only one row and one column, you can drop the header and return directly the value
- "Z" value as number
- "K" value as text
- "C" value automatic depending if there is a numeric e

Example 1

films	$\pi_{\text{title}}\ \delta_{\text{country="CH"}}$ films
filmid::title::country 1001::Ma vie de Courgette::CH 1002::Elle::FR 1003::Torni Erdmann::DE 1004::Above And Below::CH	title Ma vie de Courgette Above and Below

- Return the title of all Swiss movies
- = relProject(relSelect(films,"$country=""CH"""),"title")
- = relFilter(films, "S $country=""CH""", "P title")
- = relSql("SELECT title FROM t1 WHERE country = 'CH'", films)

Example 2

films	theatres	$\pi_{\text{title, theatre}}$ films ⋈ theatres
filmid::title::country 1001::Ma vie de Courgette::CH 1002::Elle::FR 1003::Torni Erdmann::DE 1004::Above And Below::CH	theatreid::theatre::filmid 21::Corso::1003 22::Apollo::1001 23::Metropol::1001 24::Le Paris::1002	title:theatre Ma vie de Courgette::Apollo Ma vie de Courgette::Metropol Elle::Le Paris Toni Erdmann::Corso

- Show title of all films and the name theatres they are shown
- = relProject(relJoin(films,theatres,"NATURAL"),"title::theatre")
- = relFilter(films, theatres, "J NATURAL","P title::theatres)
- = relSql("SELECT title, theatres FROM t1 NATURAL JOIN t2", films, theatres)

Example 3

films	theatres	π_{filmid} films - π_{filmid} films ⋈ theatres
filmid::title::country 1001::Ma vie de Courgette::CH 1002::Elle::FR 1003::Torni Erdmann::DE 1004::Above And Below::CH	theatreid::theatre::filmid 21::Corso::1003 22::Apollo::1001 23::Metropol::1001 24::Le Paris::1002	filmid 1004

- Show the title of the films that are not shown
- = relDifference(relProject(films,"filmid"), relProject(relJoin(films,theatres,"NATURAL"),"filmid"))
- = relFilter(films, "P filmid", films, theatres, "J NATURAL","P filmid,"D")
- = relFilter(films, theatres, "J leftantisemi", "P filmid")
- *No direct expression in relSql*

Other functions

- Rotate
- Fixpoint
- Assert
- Special operators in relFilter
  - # starts a comment
  - ! stops execution (debugging)