RQL: A Query Language for Recommender Systems

Gediminas Adomavicius\(^1\)  Alexander Tuzhilin\(^2\)  Rong Zheng\(^2\)

\(^1\)Information and Decision Sciences
Carlson School of Management
University of Minnesota
gedas@umn.edu

\(^2\)Information, Operations, and Management Sci.
Stern School of Business
New York University
atuzhili,rzheng@stern.nyu.edu

Presented by: Eugeny Kharlamov, Romans Kasperovics
Complete RQL Syntax

RECOMMEND recommend_dim_attr_list
TO recipient_dim_attr_list
FROM cube
BASED ON measure_list
WHERE dimension_restrictions //optional
WITH measure_restrictions //optional
AGGR BY aggregation_dim_attr_list //optional
HAVING aggregation_restriction //optional
SHOW measure_rank_restriction //optional, default: SHOW TOP 1
RQL Query Parts

1. Core RQL query
   - TO recipient_dim_attr_list
   - FROM cube
   - BASED ON measure_list
   - WHERE dimension_restrictions
   - WITH measure_restrictions
   - AGGR BY aggregation_dim_attr_list
   - HAVING aggregation_restriction

2. Recommendation Wrapper
   - RECOMMEND recommend_dim_attr_list TO recipient_dim_attr_list
   - SHOW measure_rank_restriction
Data Cube

\( \langle D, M, A, f, L \rangle \)

- \( D = \{ d_1, d_2, \ldots, d_n \} \) – set of dimensions
- \( M = \{ m_1, m_2, \ldots, m_k \} \) – set of measures
- \( A = \{ a_1, a_2, \ldots, a_t \} \) – set of attributes
- \( f : D \to 2^A \) – mapping that identifies a set of attributes for each dimension: \( f(d_i) \cap f(d_j) = \emptyset \) and \( \bigcup_i f(d_i) = A \)
- \( L = \{ l_1, l_2, \ldots, l_j \} \) – set of cube cells, where \( l_i = \langle addr_i, cont_i \rangle \), where \( addr_i \in \text{dom}(d_1) \times \text{dom}(d_2) \times \ldots \times \text{dom}(d_n) \) and \( cont_i \in \text{dom}(m_1) \times \text{dom}(m_2) \times \ldots \times \text{dom}(m_k) \)
Recommendation Algebra

- Restriction (RSTR)
- Metric Projection (MRPJ)
- Destroy Dimension (DTDM)
- Aggregation (AGGR)
MAP($RQL_{Query}$)
(1) $RA_{op} := ID$
(2) if (exists WHERE clause in $RQL_{Query}$) then
   $RA_{op} := RA_{op} \oplus RSTR($dimension_restrictions$);$
(3) if (exists WITH clause in $RQL_{Query}$) then
   $RA_{op} := RA_{op} \oplus RSTR($measure_restrictions$);$
(4) if (exists AGGR BY clause in $RQL_{Query}$) then
   $RA_{op} := RA_{op} \oplus AGGR($aggregation_dim_attr_list$);$n
(5) if (exists HAVING clause in $RQL_{Query}$) then
   $RA_{op} := RA_{op} \oplus RSTR($aggregation_restriction$);$n
(6) $RA_{op} := RA_{op} \oplus MRPJ(M - measure_list);$n
(7) $final\_dim\_list := f^{-1}(recommend\_dim\_attr\_list) +$
    $f^{-1}(recipient\_dim\_attr\_list);$n
(8) $RA_{op} := RA_{op} \oplus DTDM(D - final\_dim\_list);$n
(9) Return $RA_{op};$
Snowflake Representation (ROLAP)

- **Profession** $D_{3.2}$
  - Name

- **Companion** $D_{1.1}$
  - Type
  - PublicRating
  - PersonalRating

- **Genre** $D_{2.2}$
  - Name

- **User** $D_{3.1}$
  - Name
  - Age
  - Gender

- **Movie** $D_{2.1}$
  - Name
  - Length
  - Year

- **City** $D_{5.2}$
  - Name

- **Country** $D_{5.3}$
  - Name

- **Theatre** $D_{5.1}$
  - Name
  - Capacity

- **DayOfTheWeek** $D_{4.2}$
  - Weekend

- **Time** $D_{4.1}$

G. Adomavicius et al. ( )
RQL
UNIBZ, 04/12/2006 7 / 17
Restriction on an attribute of Rating table:

\[ Rating = \sigma[P_m](Rating) \]

Restriction on an attribute of some dimension \( D_{i,n} \):

\[
D_{i,n} = \sigma[P_{dn}](D_{i,n})
\]

\[
D_{i,(j-1)} = (\pi[Xn](\bowtie[D_{i,(j-1)}.ID = D_{i.n}.ID](D_{i(j-1)}, D_{i.j})), j = 2, \ldots, n
\]

\[ Rating = (\pi[Xr](\bowtie[D_{i.1}.ID = Rating.D_i.ID](D_{i1}, Rating))) \]
Rating = \pi [M - M_p](Rating)
Recommendation Algebra $\rightarrow$ Relational Algebra

Destroy Dimension (DTDM)

1. Drop all tables $D_{i.1}, \ldots, D_{i.m}$
2. Delete $D_i ID$ from Rating table
3. Merge similar tuples using aggregation function $F_{aggr}$

$$Rating = \pi[D - D_i ID](FN[M, F_{aggr}, D - D_i](Rating))$$
Recommendation Algebra $\rightarrow$ Relational Algebra

Aggregation (AGGR)

1. Join all the dimensions from $D_{i.1}$ to $D_{i.n}$

$$D_{i.(j-1)} \bowtie [D_{i.(j-1)}.ID = D_{i.j}.ID](D_{i.(j-1)}, D_{i.j}), j = 2, \ldots, n$$

2. Join Rating with $D_{i.1}$ and update metrics with aggregated values

$$\text{Rating} = FN[M, F_{aggr}, D_{i.n}.ID](\bowtie [D_{i.1}.ID = \text{Rating}.D_{i}.ID](D_{i.1}, \text{Rating}))$$

3. After the aggregation, $D_{i.n}$ becomes the basic table for the dimension $d_i$ (new $D_{i.1}$)

$$D_{i.1} = D_{i.n}$$
Example: Mapping RQL Queries into SQL

Recommend top 5 action movies to the female users living in New York

RECOMMEND Movie TO User
FROM MovieRecommender
BASED ON PersonalRating
WHERE Movie.Genre = "Action" AND
Theater.City = "New York"
AND User.Gender = "Female"
SHOW TOP 5
Example: Mapping RQL Queries into SQL
Translation to Recommendation Algebra

\[
\text{DTDM (Theater, Time, Company)} \ ( \\
\quad \text{MRPJ (PublicRating)} \ ( \\
\qquad \text{RSTR (Movie.Genre = “Action”)} \ ( \\
\qquad \quad \text{RSTR(Theater.City = “New York”)} \ ( \\
\qquad \qquad \text{RSTR (User.Gender = “Female”)} \ ( \\
\qquad \qquad \quad \text{MovieRecommender} ))))))
\]
Example: Mapping RQL Queries into SQL
Translation to Relational Algebra

User1 = $\sigma_{User.Gender=Female}(User)$

Theater1 = $\bowtie \sigma_{City.CityName=NewYork}(City)$

Movie1 = $\bowtie \sigma_{Genre.GenreName=Action}(Genre)$

Rating1 = $\bowtie \bowtie \bowtie \bowtie (Movie1, Theater1, User1, Rating)$

Rating2 = $\pi_{PersonalRating}(Rating1)$

Rating3 = $FN_{PersonalRating,AVG}\{Theater, Time, Company\}(Rating2)$

Rating4 = $\pi_{Movie.Name, User.Name, User.ID}(Rating3)$
Example: Mapping RQL Queries into SQL I

Translation to SQL

```
FROM
  (SELECT *
   FROM
    (SELECT *
     FROM Users
     WHERE Users.Gender="female"
    ) as U1,
    (SELECT *
     FROM
      (SELECT *
       FROM Theater,
       (SELECT *
        FROM City
        WHERE City.CityName="New York"
       ) as A
      ) as T1,
      (SELECT *
       FROM Rating,
       (SELECT *
        FROM Rating,
        ) as R4
     )
  )
```
Example: Mapping RQL Queries into SQL II

Translation to SQL

(SELECT *
FROM Movie,
 (SELECT *
  FROM Genre
  WHERE Genre.GenreName = "Action"
 ) as G
WHERE Movie.GenreId = G.GenreId
 ) as M1
WHERE Rating.MovieID = M1.MovieID
 ) as R1
WHERE Rating.RatingId = R1.RatingId
 ) as R2
WHERE T1.TheaterId = R2.TheaterId
 ) as R3
WHERE U1.UserId = R3.UserId
 ) as R4
GROUP BY R4.MovieID, R4.UserID

G. Adomavicius et al. ()
Example: Mapping RQL Queries into SQL
Romans’s Version

```
SELECT M.Name, M.ID, T.Name, T.ID, C.Name, U.Name, U.ID, G.Name, 
    AVG(R.PersonalRating) AS PersonalRating 
FROM Rating R, Movie M, Theatre T, City C, User U, Genre G 
WHERE R.MovieID = M.ID AND R.TheatreID = T.ID 
    AND R.UserID = U.ID AND T.CityID = C.ID 
    AND M.GenreID = G.ID AND G.Name = "Action" 
    AND C.Name = "New York" AND U.Gender = "female" 
GROUP BY M.Name, M.ID, T.Name, T.ID, C.Name, U.Name, U.ID, G.Name
```