

RQL: A Query Language for Recommender Systems

Gediminas Adomavicius¹ Alexander Tuzhilin² Rong Zheng²

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

²Information, 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

$\langle D, M, A, f, L \rangle$

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

Recommendation Algebra

- Restriction (RSTR)
- Metric Projection (MRPJ)
- Destroy Dimension (DTDM)
- Aggregation (AGGR)

RQL → Recommendation Algebra

$$op_1 \oplus op_2(cube) = op_2(op_1(cube))$$

MAP(*RQL_Query*)

(1) $RA_op := ID$

(2) if (\exists WHERE clause in *RQL_Query*) then

$$RA_op := RA_op \oplus \mathbf{RSTR}(\text{dimension_restrictions});$$

(3) if (\exists WITH clause in *RQL_Query*) then

$$RA_op := RA_op \oplus \mathbf{RSTR}(\text{measure_restrictions});$$

(4) if (\exists AGGR BY clause in *RQL_Query*) then

$$RA_op := RA_op \oplus \mathbf{AGGR}(\text{aggregation_dim_attr_list});$$

(5) if (\exists HAVING clause in *RQL_Query*) then

$$RA_op := RA_op \oplus \mathbf{RSTR}(\text{aggregation_restriction});$$

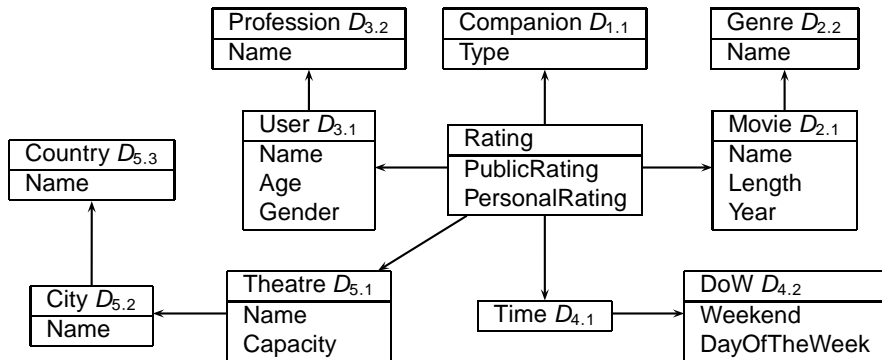
(6) $RA_op := RA_op \oplus \mathbf{MRPJ}(M - \text{measure_list});$

(7) $final_dim_list := f^{-1}(\text{recommend_dim_attr_list}) +$
 $f^{-1}(\text{recipient_dim_attr_list});$

(8) $RA_op := RA_op \oplus \mathbf{DTDM}(D - final_dim_list);$

(9) Return RA_op ;

Snowflake Representation (ROLAP)



Recommendation Algebra \rightarrow Relational Algebra

Restriction (RSTR)

- Restriction on an attribute of Rating table:

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

- Restriction on an attribute of some dimension $D_{i.n}$:

$$D_{i.n} = \sigma[P_{dn}](D_{i.n})$$
$$D_{i.(j-1)} = (\pi[Xn](\bowtie [D_{i.(j-1)}.ID = D_{i.n}.ID](D_{i.(j-1)}, D_{i.j})))$$
$$, j = 2, \dots, n$$
$$Rating = (\pi[Xr](\bowtie [D_{i.1}.ID = Rating.D_i.ID](D_{i_1}, Rating)))$$

Recommendation Algebra \rightarrow Relational Algebra

Metric Projection (MRPJ)

$$Rating = \pi[M - M_p](Rating)$$

Recommendation Algebra \rightarrow Relational Algebra

Destroy Dimension (DTDM)

- 1 Drop all tables $D_{i,1}, \dots, D_{i,m}$
- 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 $D_{i.n}$

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

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

$$\begin{aligned} Rating &= FN[M, F_{aggr}, D_{i.n}.ID](\\ &\bowtie [D_{i.1}.ID = Rating.D_iID](D_{i.1}, Rating)) \end{aligned}$$

- 3 After the aggregation, $D_{i.n}$ becomes the basic table for the dimension d_j (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

```
DTDM (Theater, Time, Company) (  
  MRPJ (PublicRating) (  
    RSTR (Movie.Genre = "Action") (  
      RSTR(Theater.City = "New York") (  
        RSTR (User.Gender = "Female") (  
          MovieRecommender ) ) ) ) )
```

Example: Mapping RQL Queries into SQL

Translation to Relational Algebra

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

$Theater1 = \bowtie_{Theater.TheaterID=City.TheaterID}(Theater, \sigma_{City.CityName=NewYork}(City))$

$Movie1 = \bowtie_{Genre.GenreID=Movie.GenreID}(Movie, \sigma_{Genre.GenreName=Action}(Genre))$

$Rating1 = \bowtie_{Moive1.MovieID=Rating.MovieID}(Movie1, \bowtie_{Theater1.TheaterID=Rating.TheaterID}(Theater1, \bowtie_{User1.UserID=Rating.UserID}(User1, Rating)))$

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

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

$Rating4 = \pi_{Movie.Name,User.Name,User.ID}(Rating3)$

Example: Mapping RQL Queries into SQL I

Translation to SQL

```
SELECT R4.MovieID, R4.UserId, R4.PersonalRatings
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
         WHERE Theater.CityID = A.CityID
        ) as T1,
        (SELECT *
         FROM Rating,
          (SELECT *
           FROM Rating,
```

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
```

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
```