RQL: A query language for recommender systems

Romans Kasperovics
Evgeny Kharlamov

November, 2006
Bolzano

Presentation based on

“RQL: A query language for recommender systems”

- GEDIMINAS ADOMAVICIUS
  University of Minnesota
- ALEXANDER TUZHILIN
  New York University
- Rong Zheng
  New York University

CeDER-05-15 2005
Outline

1. Motivation
2. Multidimensional Model for recommender systems
3. Recommendation Query Language (RQL)
4. Recommendation Algebra (RA)
5. Processing RQL queries: RQL → SQL
   1. Translation from RQL to RA
   2. Translation from RA to Relational Algebra

Motivation

- Recommendations provided by current recommender systems:
  1. top $k$ products to a user
  2. top $k$ users to a product

  - too poor recommendations
  - user in not asked: “which recommendations are you really need?”
Motivation

- Main motivation of the research:
  - To solve the problem of poor recommendations
  - To allow end-users to express a broad range of recommendations

- RQL – an attempt to solve the problems.
  - It allows end-users to request required recommendations.
  - It is specifically tailored for multidimensional Recommender Systems
Multidimensional Model

- n-D or Multidimensional model

  2-D: Users, Items
  \[ R: \text{Users x Items} \rightarrow \{r_1, \ldots, r_n\} \]

  n-D: Users, Items, Context
  \[ R: \text{Users x Items x Context} \rightarrow \{r_1, \ldots, r_n\} \]
Multidimensional Model

- Example: context

  Context = (with\_whom, where, when, ...)  
  
  \[ R(John, Apocalipsis\_Now, with\_GF, at\_night, at\_home, ... ) = 1 \]

Multidimensional Model

- n-D model formally:
  - \( D_1, ..., D_n \) – sequence of dimensions
  - each \( D_j = (Attr_{j1}, ..., Attr_{jk_j}) \)
    some \( D_j \) can have attached hierarchies
  - \( S = D_1 \times ... \times D_n \) – recommendation space
  - \( R: S \rightarrow \{r_1, ..., r_n\} \) – rating function (rating cube)
Multidimensional Model

- Example: n-D model

- \( D_i = \text{Users}(\text{Used}_id, \text{UName}, \text{UAddress}) \)
- \( D_k = \text{Time}(\text{Time}_id, \text{day}, \text{month}, \text{year}) \)

\[ \text{hierarchy}(D_k) = (\text{Days}/\text{Seasons}, \text{Days}/\{\text{weekday}, \text{weekend}\}) \]

Multidimensional Model

- n-D rating cube:
Multidimensional Model

- Example: Usage of hierarchies

Recommendation Problem:
for $S = D_1 \times \ldots \times D_n$ and probably partial function $R$ is to provide a recommendation requested by a user if the request is expressed in RQL.
Recommendation Query Language (RQL)

Example: classical recommendation request:
recommend the best (one) movie for the user (one)

RECOMMEND Movie TO User
FROM MovieRecommender
BASED ON PersonalRating
RQL

- SQL-like syntax →
  1. hides many implementation-related details
  2. declarative = intuitive requests

Example: recommendation request

Recommend top 5 movies to the user to see over the weekend, but only if the personal ratings of the movies are higher than 7 (if fewer than 5 movies satisfy these criteria, then show only those satisfying them).
**RQL**

- Example: recommendation request

```sql
RECOMMEND Movie TO User
FROM MovieRecommender
BASED ON PersonalRating
WHERE Time.WeekTime = "Weekend"
WITH PersonalRating > 7
SHOW TOP 5
```

- WHERE and WITH clauses:
  - Where – to select attributes
  - With – to select ratings

```sql
WHERE Time.WeekTime = "Weekend"
WITH PersonalRating > 7
```
RQL

- Example: user interface for naive-users

RQL

- UI semantics: based on RQL, i.e. there is a direct translation from UI to RQL
RQL

Example: vector of values as a recommendation

RECOMMEND Movie, Time TO User, Companion
FROM MovieRecommender
BASED ON PersonalRating
WHERE User.Name = “Tom” AND
    Time.WeekTime =”Weekend” AND
    Companion.Type = “Girlfriend”
SHOW TOP 3

RQL

Example: usage hierarchies in recommendations

RECOMMEND Movie.Genre TO User.Profession
FROM MovieRecommender
BASED ON AVG (PersonalRating)
WITH PersonalRating > 6
AGGR BY Movie.Genre, User.Profession
Types of aggregations for hierarchies in recommendations

- AVG
- SUM
- MAX
- MIN
- etc.

Example: recommendations not for users

RECOMMEND User.Profession TO Movie
FROM MovieRecommender
BASED ON AVG(PersonalRating)
WHERE Movie.Title = "Beautiful Mind"
AGGR BY User.Profession
SHOW TOP 2
RQL

- Example: recommendations based on multiple ratings

```sql
RECOMMEND Movie To User
FROM MovieRecommender
BASED ON PersonalRating, PublicRating
WHERE User.Profession = "Student"
WITH Public_Rating>8 AND
    RATED (PersonalRating) >8
SHOW TOP 5
```

RQL

- User specified vs. estimated ratings:

```sql
WITH Public_Rating>8 AND
    RATED (PersonalRating) >8
```
RQL

- General pattern of a rec. request:

```
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
```

- Limitations:
  - no multicube recommendations

  FROM cube

- Reasons:
  - rarely used
  - too costly to compute
RQL

- Operational Semantics:

```sql
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

- Selection (1) vs. Display (2) operators:

```sql
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

- Selection operators – extract a sub-cube from the rating cube
- Display operators – create a recommendation matrix form the sub-cube

Output:
Recommendation matrix
Recommendation Algebra

- Recommendation Algebra (RA) – created using OLAP algebras
- RA algebra is a fragment of OLAP algebra
- Consists of 4 operators:
  1. Restriction
  2. Metric projection
  3. Destroy dimension
  4. Aggregation
Recommendation Algebra

- Data cube as a 5-tuple
  \( C = \langle D, M, A, f, L \rangle \)

1. \( D = \{d_1, \ldots, d_n\} \) – n dimensions (Time, Users)
2. \( M = \{m_1, \ldots, m_k\} \) – k measures (Personal, Public)
3. \( A = \{a_1, \ldots, a_t\} \) – t attributes (dd, mm, yy in Time)

4. \( f: D \rightarrow 2^A \) – attributes’ assignment.
   \( f(d_i) \cap f(d_j) = \emptyset \)
   \( f(\text{Time}) = \{dd, mm, yy\} \)

5. \( L = \{l_1, \ldots, l_j\} \) – j cube’s cells
   \( l_i = \langle \text{address}_i, \text{content}_i \rangle \)
   \( \text{address}_i \in \text{dom}(d_1) \times \ldots \times \text{dom}(d_n) \)
   \( \text{content}_i \in \text{dom}(m_1) \times \ldots \times \text{dom}(m_k) \)
Recommendation Algebra

- Restriction (RSTR) RA operator
  - a version of “slide and dice” operator
  - put restrictions on some dimensions and/or attributes of dimensions
  - extract those cube cells that sat. the restriction

Restriction =

\[ P = pr_1 \langle \text{op} \rangle \ldots \langle \text{op} \rangle \ pr_s \]

- \( \text{op} \) \( \in \{\text{and, or}\} \)
- \( pr_i \) - predicate (age>21)

\[ \text{RSTR}_p(<D,M,A,f,L>)= <D,M,A,f,L_0> = <D,M,A,f,P(L)> \]
Recommendation Algebra

- Metric Projection (MRPJ) RA operator
  - modifies cube's cells by projecting the \(content \in \text{dom}(m_1) \times \ldots \times \text{dom}(m_k)\) of each cell = \(<\text{address}, \text{content}>\) on some coordinates
  - \(content = (1,3,d) \Rightarrow \text{MRPJ}(content) = (d)\)

- \(M = \{m_1, \ldots, m_k\}\)
- \(MS = \{m'_1, \ldots, m'_h\}\)

- \(\text{MRPJ}_{MS}(<D,M,A,f,L>) = <D,M_0,A,f,L_0> = <D, M-MS, A, f, \pi_{M-MS}(L)>\)
Recommendation Algebra

- Destroy Dimension (DTDM) RA operator
  - drops one dimension, corresponding attributes and, consequently, modifies information in the rating cube

- \( D = \{d_1, \ldots, d_n\} \)
  - \( d \in \{d_1, \ldots, d_n\} - \text{to be destroyed} \)
  - \( \text{DTDM}_d(<D,M,A,f,L>) = <D_0,M,A_0,f_0,L_0> = <D-\{d\}, M, A-f(d), f|_{D_0}, \pi_{D-\{d\}}(L)> \)
Recommendation Algebra

- **Destroy Dimension (DTDM) RA operator**
  - modifies cube’s cells by projecting the \( address \in \text{dom}(d_1) \times \ldots \times \text{dom}(d_n) \) of each cell = \(<address, content>\) on some coordinates
  - \( address=(\text{time'},\text{user'},\text{place'},\text{item'}) \) → \( \text{DTDM}(address)=(\text{time'},\text{user'},\text{item'}) \)

- **Aggregation (AGGR) RA operator**
  - Perform aggregation on some dimensions
  - \( d = \text{Movie}, h(d)=\{\text{movie/genre}\} \) → AGGR – aggregates Movie dimension based on genre
  - aggr. function \( F_{\text{aggr}} \in \{\text{SUM, AVG, MAX, MIN}\} \)
Recommendation Algebra

- Aggregation (AGGR) RA operator

- $D = \{d_1, \ldots, d_n\}, d_i \in \{d_1, \ldots, d_n\} - to be aggregated$

- $\text{AGGR}_{d_i}(\langle D, M, A, f, L \rangle) = \langle D_0, M, A_0, f_0, L_0 \rangle$
  - $D_0 = D - \{d_i\} + \{d'_i\}$
  - $A_0 = A - f(d_i) + f(d'_i)$
  - $f_0$ is $f$, redefined on the new, aggregated dimension
  - $L_0$ is $L$ where in each cell $l = \langle address, content \rangle$
i-th coordinate of address is replaced by it’s aggregation
(replacing of individual movies by aggregations over genres)

Recommendation Algebra

- Example: recommendation request expressed in RA

  recommend top 5 movies to the user to see over the weekend, but only if the personal ratings of the movies are higher than 7

5

$\text{DTDM}_{\text{MovieTheater, Time, Companion}}(\text{MRPJ}_{\text{PublicRating}}(\text{RSTR}_{\text{PersonalRating > 7}}(\text{RSTR}_{\text{Time, WeekTime="Weekend"}}(\text{MovieRecommender})))))}$
Query processing

Query Processing

- RQL
  - Recommendation Algebra
  - Relational Algebra ~ SQL