ISSN 1648-4142 TRANSPORT
http:/www.vtu.lt/english/editions

TRANSPORT – 2003, Vol XVIII, No 4, 157–161

MODELLING RELATIONS BETWEEN TRANSPORTATION OBJECTS USING DISTRIBUTED ATTRIBUTE SETS

Rolandas Griškevičius

Dept of Fundamental Sciences, Vilnius Gediminas Technical University,
Saulėtekio al. 11, LT-2040 Vilnius, Lithuania. E-mail: rolandas.griskevicius@fm.vtu.lt

Received 2003 01 16; accepted 2003 06 30

Abstract. Relational sets with distributed attributes (RSDA), described in this paper, are used to model relationships between transportation objects. RSDA is a data storage and management model, extending traditional model of E. F. Codd. It has a richer attribute semantics and intersection effect. Because relationships between transportation objects have rich semantics (i.e. relationship between a cargo and a means of transport), special methodology is required to capture relation singularities.

Keywords: relational sets, business object, object, semantic relationships.

1. Introduction

One of the ways to extend attribute semantics in a traditional Codd's [1] model is to join several sets. For example, two relational sets (*RS*) are given: $R1(A, B_{11} \dots B_{1n1})$ and $R2(A, B_{21} \dots B_{2n2})$, with dependency $A \rightarrow \{B_{11}, \dots B_{1n1}\}$, $A \rightarrow \{B_{21} \dots B_{2n2}\}$. Joined set, combined from both *R*1 and *R*2 attributes in its tuples, and only those tuples, where attributes *A* from *R*1 and *R*2 are equal is as follows: $R1 >_{(A1=A2)} R2 = R3(A, B_{11}, \dots B_{1n1}, B_{21} \dots B_{2n2})$. From the functional dependency view ([2–5]), join operation extends attribute *A* semantics, because in *R*3 attributes A functional dependency becomes $A \rightarrow \{B_{11}, \dots B_{1n1}, B_{21} \dots B_{2n2}\}$. In this case, attributes $B_{11}, \dots B_{1n1}$, $B_{21} \dots B_{2n2}$ can be considered as semantic components of attribute *A*. By applying relational operator "left outer join", additional semantics can be extracted: this operator yields an empty part of a tuple, when corresponding attribute in a joined set does not exist. In general the resulting set is as follows: $R_m = (A, B_{i1} \dots B_{mn})$. Such a set will have dependent attributes as domains and tuple keys as corteges (Fig 1).

From a semantic point of view each value of attribute $B_1 \dots B_n$ functionally depends on tuple key a, thus forming complex value from subattributes <a> and <*B*>. This adds new functionality to the set: 1) Each value may have its own semantics, 2) attribute values are distributed, but undistinguishable in mean of semantics, 3) for each attribute an empty value ∅ may exist, which shows that the given attribute has no meaning in an applied domain.

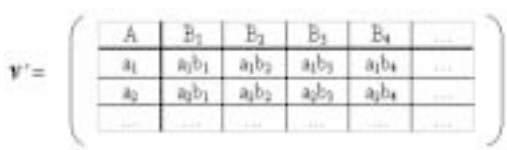


Fig 1. Prototype of relational set with distributes attributes

The approach of an attribute distribution was proposed by P. Adomenas ([6, 7]). We will call these sets as "relational sets with distributed attributes" (RSDA).

From the view of data, there's no difference between subattributes <a> and <*B*>, so in a general case RSDA set may be denoted as (Fig 2):

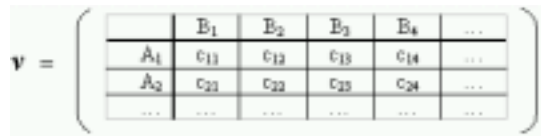


Fig 2. Relational set with distributed attributes

Subattribute sets $<A> = A_1 \dots A_k$ and $<B> = B_1 \dots B_l$ form subshemas, which later form the schema of a set *v*: $R_v = V(<A>;<B>)$. Numbers $k = |<A>|$, $l = |<B>|$ show the count of subattributes in the subschemas. Under RSDA attribute we will mean any pair of subattributes A_i/B_j. Under attribute value we will mean certain evaluation of attribute (i.e. number, character string), expressed in the form acceptable by calculation. Thus, value and subattribute pair fully describe an attribute. If $i = 1 \dots |<A>|$ ir $j = 1 \dots |<B>|$, then $<c_{ij}> = attr(V)$ – attribute entirety of a given set. Map *sub:* $<c_{ij}> \rightarrow <A>$ shows dependency of attribute to a corresponding subattributes. In a general case, RSDA is a triple: $v = <V, <c>, sub(A)>$,

here V – schema, , $<c>$ - atribute entirety, *sub* – attribute dependency map. It is possible to have $|<A>|=0$ or $|<B>|=0$, then $attr(V)=\varnothing$.

For some cases, the series $v_1 \ldots v_n$ with the same subschemas: $<A>_1 = <A>_2 \ldots = <A>_n$ and $<B>_1 = <B>_2 \ldots = <B>_n$ are used. Furthermore, given series address the same object and only different its data. Then $v^{[n]} = v_1 \Downarrow v_2 \Downarrow \ldots \Downarrow v_n$ is the definition of a multiattribute set. Here $\Downarrow$ - folding operator. Similarly $v^{[n]}\Uparrow^i = v_i$, $1 < i < n$, here $\Uparrow^i$ – unfolding operator. An attribute of the set $v^{[n]}$ contains the series $<c^1 \ldots c^n>$. Folding and unfolding operators do not change the schema of the set, though they change multiattribute series $<c^1 \ldots c^n>$.

Attribute access operator $v^{[n]\uparrow i} = v_i$ enables to access particular attribute in the multiattribute set. It doesn't change either subschema or the count of multiattribute series n.

2. Operations in the Distributed Attribute Environment

2.1. Primitive change operator

RSDA model contains only 1 primitive operator, which has the following form: *CH(SUB*1*,SUB*2*,VALUE)*. Here: *SUB*1 and *SUB*2 – subattributes. *VALUE* – denotes value of an attribute. *CH* can operate in 3 modes: change, insert and delete. In the first case operator *CH* changes the value of an attribute, referred by subattributes *SUB1* and *SUB*2 (attribute *SUB*1*/SUB*2). If there were no attribute before, *CH* inserts either one or both subattributes and value for a newly created attribute. (insert mode). All new attributes which have the newly created subattribute in their pairs are empty, except one - referred by *SUB*1*/SUB*2. When *CH* with *VALUE*=∅ is invoked, the value of an attribute is deleted (delete mode). If subattribute contains all ∅ after the deletion of the attribute, it is also deleted.

2.2. Relational RSDA operators

As in a traditional relation model, similar relation operators exist for RSDA:

Projection: $\pi_{(<A>,<B>)}$ – produces a set from some other set by pulling out only the identified subattributes in subschemas.

Selection: $\sigma_{(\forall / \exists)\, c\theta a)}$ – produces a set containing only those attributes, which match specified criteria. θ - any of comparison operators, for example: <, >, ≤, ≥, ≠, =.

Cartesian: produces a set from 2 sets, containing all possible attribute combinations from each set: $v^{[2]} = v_1 \times v_2$.

Intersection operator is expressed as: $v_1 \rhd\lhd_{\{A1\theta A2\}} v_2$, here A1 and A2 are subattributes of the first and second RSDA operands, θ - any comparison operator (i.e. <, >, ≤, ≥, ≠, =). This operator is not a primitive, it can be expressed using selection, projection and Cartesian product.

Union operator yields a set, which is combined from two sets by "sticking" them together on one specified subschema. It is of the following form: $v_1 \cup_{\{S1,S2\}} v_2$, here S1 and S2 denote subschemas on which to apply the union operation.

Subschema transposition operator reverses the order of subschemas: $\nabla V(<A>;<B>)=V(<B>;<A>)$.

2.3. RSDA scalar operations

Scalar operation for the given RSDA series $<V> = v_1 \ldots v_n$ with the schemas $V(<A>, <B>)$ is as follows:

$c_{ij}^r = scal\,(c_{ij}^1, \ldots c_{ij}^k)$, $i = 1 \ldots |<A>^*|$, $j = 1 \ldots |<B>^*|$, here scal(…) – prefix form of any scalar relation, $<A>^* = <A>^1 \bigcap \ldots \bigcap <A>^k$, $<B>^* = <B>^1 \bigcap \ldots \bigcap <B>^k$, $<c_{ij}> = attr(V(<A>^*,<B>^*))$.

2.4. Transformations RS→RSDA and RS'→RSDA

Transformation RS→RSDA is possible when a system of related RS is in the 3rd normal form and there is common key attribute(s) A in every RS. By applying grouping operator γ [8], we find all existing domain values of A. This grouped set later is joined with sets $r_1 \ldots r_n$ using LEFT OUTER JOIN operator. Transformation expression is as follows:

$$t_{RS\to RSDA} = {}_A\gamma(\pi_A(r_1) \cup \ldots \cup \pi_A(r_n)) \; {}_A\!> r_1 \; {}_A\!> \ldots {}_A\!> r_n. \quad (1)$$

The reverse transformation, based on division ([9, 10]), is more complex, as after certain processing some attributes of RS s may not exist. This is a special case and is not covered in this paper.

To simulate RSDA using a traditional relational set system, or to save RSDA to RS we need to express the first in terms of traditional relational algebra. The exported set will have a special schema, functionally very close to RSDA. We consider it as RSDA-ready set, or RS. Such a set, data of which is taken from Fig 2 is shown in Fig 3:

SUB1	SUB2	TYPE	VALUE
1	1	N	c_{11}
1	2	N	c_{12}
1	3	N	c_{13}
1	4	N	c_{14}
2	1	N	c_{21}
2	2	N	c_{22}

Fig 3. RSDA-ready set

Attributes SUB1 and SUB2 contain the references of RSDA subattributes. Attribute VALUE contains values of certain attributes. Attribute TYPE contains a type code attribute which depends on implementation.

Transformation then from RS' to RSDA is as follows:

$$t_{RA'\to RPA} = {}_{SUB1}\gamma(\pi_{SUB1}(r_{RA'})) > {}_{SUB1}(\pi_{SUB1,VALUE}\sigma_{SUB2=H_1}(r_{RA'}))... > {}_{SUB1}(\pi_{SUB1,VALUE}\sigma_{SUB2=H_p}(r_{RA'})) \quad (2)$$

here $H_1 \ldots H_p = {}_{SUB2}\gamma(\pi_{SUB2}(r_{RA'}))$ – the set of SUB2 values with duplicates eliminated.

The aim of this transformation is to form the traditional algebra set, structurally similar to RSDA. Transformation is based on SUB1 and SUB2 attributes.

Using this way of transformation, desired sequence of traditional relational algebra operations can be applied on RS', and later RS' can be presented in look-like RSDA view. We call it RS' – based RSDA calculus. Primitive operator *CH*, described previously, can easily be expressed using primitive relational algebra operators *CH*, *DEL* and *ADD* for RS'.

2.5. Integration of RSDA and traditional relational sets

The RSDA-like expression form Fig 3 and the transformation 1 give possibility to join traditional and RSDA sets into one relational set system. Two types of joins can exist, in distinction where data is carried: subattribute renaming join (S-join) and attribute append join (A-join).

The S-join $v_S = r_1 >^S_{\{R,K,Sm\}} v_2$ and A-join $v^{[2]}_S = r_1 >^A_{\{R,K,Sm\}} v_2$ operators (Fig 4, Fig 5):

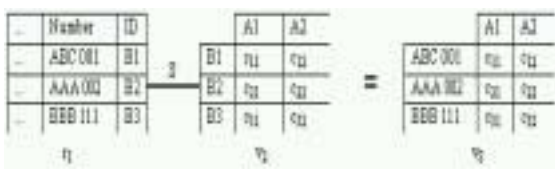


Fig 4. S-join example

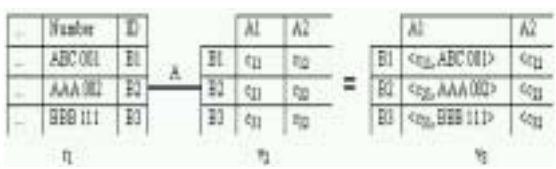


Fig 5. A-join example

3. Use of RSDA to Model Relationships between Transportation Objects

3.1. Modelling of relationship data of transportation objects

As RSDA's attribute is the intersection of subattributes, it can be used to model the intersections of objects. Attribute at the intersection holds some value, thus defining data of the intersection. This intersection with value defines relationship.

Schedule is the kind of relation between some objects. As objects to relate we will choose vehicles from set *Veh(VID, No, Wght)* with identifier, number and maximal transportation weight and cargos from set *Cg(CID, Name, Wght)* with identifier, name, weight. Schedule will be presented as RSDA $v^{[2]}$, here $v^{[2]}\uparrow^1$ – cargo load time and $v^{[2]}\uparrow^2$–cargo unload time. Schema of latter: $Sch^{[2]}(<VID>, <CID>)$, here subschema $<VID>$ will correspond vehicle identifiers, subschema $<CID>$ - cargo identifiers.

Using operator *CH* schedule time can be entered. Then the following data checks may be applied:

Scalar operation $v = Sch^{[2]}\uparrow^1 < Sch^{[2]}\uparrow^2$ yields RSDA with binary attributes, false value of which denotes wrong time entered, as unload time cannot be less than load time.

Selection $v = \sigma_{(\forall\, c=FALSE)}\, v$ produce the set, containing these wrong attributes.

Cargo weight control. Cargo weight cannot exceed it's vehicle maximum transportation weight.

First, temporary set with cargo and vehicle weight, produced:

$v^{[2]} = Veh >^A_{\{Wght,VID,<VID>\}} (Cg >^A_{Wght,CID,<CID>\}} Sch^{[2]}\uparrow^1)$.

Then scalar operation $v = v^{[2]}\uparrow^1 < v^{[2]}\uparrow^2$ gives the set with inadequate cargos and vehicles.

Time intersection control is the most important schedule operation. One or more schedules can be generated, but we always want to ensure that their time does not intersect.

Using Cartesian product we first generate temporary set:

$v^{[4]} = Sch1^{[2]} \times Sch2^{[2]}$

It can be now checked for time intersection using scalar operation:

$v = date\_test(Sch1^{[2]}\uparrow^1, Sch1^{[2]}\uparrow^1, Sch1^{[2]}\uparrow^1, Sch1^{[2]}\uparrow^2)$, here

$date\_test(v_1, v_2, v_3, v_4) = between(v_3, v_1, v_2)$ *and* $between(v_4, v_1, v_2)$,

$between(v_3, v_1, v_2) = v1 \le v3 \le v2$, here $\le$ - scalar comparison.

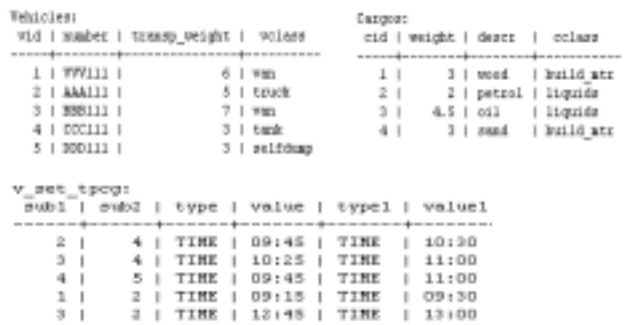
Visualisation of schedule. The S-join generates visual representation of schedule:

$Sch^{[2]}\uparrow^1 = Veh >^S_{\{No,VID,<VID>\}} (Cg >^S_{\{Name,CID,<CID>\}} Sch^{[2]}\uparrow^1)$.

3.2. Modelling relationship semantics of transportation objects

Sets of vehicles and cargos and RSDA set of relationships between them are given in Fig 6. The latter is expressed in RSDA-ready form using standard means of

relational sets. Subattributes sub1 and sub2 relate to cargo identifiers CID and vehicle identifiers VID correspondingly.



Vehicles:

vid	number	transp_weight	vclass
1	VVV111	6	van
2	AAA111	5	truck
3	BBB111	7	van
4	CCC111	3	tank
5	DDD111	3	selfdump

Cargos:

cid	weight	descr	cclass
1	3	wood	build_mtr
2	2	petrol	liquids
3	4.5	oil	liquids
4	3	sand	build_mtr

v_set_tpcg:

sub1	sub2	type	value	type1	value1
2	4	TIME	09:45	TIME	10:30
3	4	TIME	10:25	TIME	11:00
4	5	TIME	09:45	TIME	11:00
1	2	TIME	09:15	TIME	09:30
3	2	TIME	12:45	TIME	13:00

Fig 6. Relational set with distributed attributes "v_set_tpcg"

Definitions of semantic rules between transportation objects Vehicles and Cargos are in the conjugated form of Horn clauses:

$$(\underline{A}_1 \rightarrow B_1) \,\&\, \ldots \,\&\, (\underline{A}_n \rightarrow B_n) \rightarrow (\underline{C} \rightarrow D), \qquad (3)$$

here $C = C_1 \,\&\, \ldots \,\&\, C_l$, operations $\rightarrow$ and & - denote implication and conjunction. Upper definition shows the way of getting result D from clause set C and rules sets $A_1 \rightarrow B_1 \ldots A_n \rightarrow B_n$. Table shows Horn clauses expressed in terms of Prolog language:

Capturing semantics between Vehicle and Cargo objects

Prolog clauses	Comments
`attribute_correct(Cid, Vid ) :- get_data_by_sub( Cid, Vid, Cclass, Vclass, Cweight, Vweight ), vehicles( Vid, _, Vweight, Vclass ), cargos( Cid, Cweight, _, Cclass ), weight_correct( Vweight, Cweight ), vehicle_correct( Vclass, Cclass ).`	Weight and class extraction using predicates "get_data_by_sub" (extracts data from RSDA set), "vehicles", "cargos".
`weight_correct( Vw, Cw ) :- Vw < Cw, write('Weight is incorrect:'), write(Vw), write('<'), write(Cw),nl,fail. weight_correct( Vw, Cw ) :- Vw >= Cw.`	Predicate to control weight of cargo and allowable transportation weight of vehicle. If latter is less, then predicate fails with an error.
`class_correct( VV,CC). class_correct( tank, liquids ). class_correct( truck, build_mtr ). class_correct( selfdump, build_mtr ).`	Definitions of cargo and vehicle classes. Classes are used to control suitable cargo transportation. For example, only tank can transport liquids. In this form we define required semantics of relation.
`vehicle_correct( VV, CC ) :- \+ class_correct( VV,CC), write('Class is incorrect: '), write(VV),write('-'),write(CC),fail. vehicle_correct( VV, CC ) :-`	Class-based control of vehicle – cargo suitability. If vehicle class does not fall under cargo's class, predicate fails with an error.

Following Prolog query (Fig 7) tests relation validity. Test fails as by definition a truck cannot carry liquids.

```
| ?- attribute_correct(3,2).
Class is incorrect: truck-liquids
no
```

Fig 7 Prolog test of relation validity

4. Conclusions

RSDA automatically guarantees object relationship uniquiness (in chapter 3.1 example relationship between cargo and vehicle is unique), as no same vehicles with the same cargo can appear as different subattributes. But for the same objects several RSDA sets can appear (several schedules in chapter 3.1 example). Then the optimal one can be chosen comparing them using RSDA algebra.

Rich semantics of relationships between transportation objects is captured using rule-based framework. In conjunction with relational RSDA algebra we get a full mechanism of relationships management (data and semantics can be managed). As RSDA sets can be integrated with traditional relational set management systems (using A-join and S-join operations), we get also open architecture of framework, which can be implemented on any transportation management system based on relational set management system.

References

1. Codd, E. F. A Relational Model Of Data For Large Shared Data Banks, CACM 1970 – 13(6).
2. Armstrong, W. W. Dependency structure of database relationships. In: Proceedings IFIP, North Holland, Amsterdam, 1974, p. 580–583.
3. Binemann-Zdanowicz, A. Systematization of Approaches to Equality-Generating Constraints. ADBIS-DASFAA 2000, p. 307–314.
4. Doležal, T. Cardinality constraints for n-ary relationship types. In ADBIS'99, Maribor' 99.
5. McAllister, A. Complete rules for n-ary relationship cardinality constraints. *Data and knowledge Engineering 27*, 1998, p. 255–288.
6. Adomėnas, P. Data functional feature sets and their adaptability. In: ADBIS'2001'. Research communications. ISBN 9986-05-449-4. Vilnius: Technika, 2001, p. 131–140.
7. Adomenas, P.; Ciucelis, A. Data Aggregation Sets in Adaptive Data Model. *Informatica*, 2002, Vol 13, No 4, p. 381–392.
8. Klug, A. Equivalence of relational algebra and relational calculus query languages having aggregate functions. *Journal of the ACM*, 29(3), 1982, p. 699–717.
9. Graefe, Goetz; Cole, Richard L. Fast Algorithms for Universal Quantification in Large Databases. *TODS*, 20(2), 1995, p. 187–236.
10. Desharnais, Jules; Jaoua, Ali; Mili, Fatma; Boudriga, Noureddine; Mili, Ali. A Relation Division Operator: The Conjugate Kernel., TCS 114(2), 1993, p. 247–272.