Completeness of Query Operations on Resource Spaces*

Hai Zhuge and Erlin Yao
China Knowledge Grid Research Group, Key Lab of Intelligent Information Processing
Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100080, P. R. China
zhuge@ict.ac.cn; alin.yao@kg.ict.ac.cn

Abstract

A great variety of languages can be designed by different people for different purposes to operate resource spaces. Two fundamental issues are: can we design more operations in addition to existing operations? and, how many operations are sufficient or necessary? This paper solves these problems by investigating the theoretical basis for determining how complete a selection capability is provided in a resource operation sublanguage independent of any host language. The result is very useful to the design and analysis of operating languages.

Keywords: Resource Space Model, query, operation, completeness, sufficiency, expressiveness.

1. Introduction

The Resource Space Model (RSM) is a semantic data model for effectively specifying, locating and managing resources based on normalized classification semantics.

A Resource Space is an n-dimensional space where every point uniquely determines one resource or a set of related resources. A resource space can be represented as $RS(X_1, X_2, \ldots, X_n)$ or *RS* in simple, where *RS* is the name of the resource space and X_i is the name of an axis [11]. Normal forms are proposed to ensure a good resource space design [10].

Fig.1 is an example of 3-dimensional resource space. Coordinates on an axis constitute a classification on the axis, and axes further classify each other. Given a set of coordinates (Year=2004, Area=Knowledge Grid, Publisher=World Scientific), a set of resources (books) can be accurately located.

* Keynote at 2nd International Conference on Semantics, Knowledge and Grid, Guilin, China, 2006. Research supported by National Science Foundation and National Basic Research Program of China (973 project no. 2003CB317000).

A number of operations of resource spaces, such as Join, Disjoin, Merge and Split, are defined in [10]. The principles for designing Resource Operation Language (ROL) of RSM are proposed in [12]. The theory on the relationship between the normal forms and the operations are developed in [12].

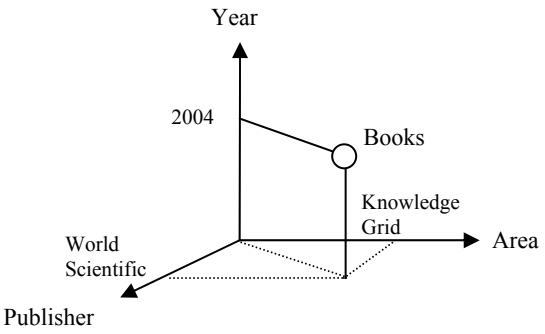

Fig. 1. A 3-dimensional Resource Space.

A great variety of languages could be designed by different developers for different purposes to query and update resource spaces. This paper investigates a theoretical basis which can be used to determine how complete a selection capability is provided in a proposed resource sublanguage independent of any host language in which the sublanguage may be embedded. We especially concern: are the defined operations sufficient and how many operations are necessary?

Relational algebra and calculus are used in the relational data model. The relational algebra is a collection of operations on relations, and a query language could be directly based on it. There are eight operations defined in the relational algebra, they are extended Cartesian product, traditional set operations (union, intersection and difference), projection, join, division and selection [5]. The relational calculus is an applied predicate calculus which may also be used in

the formulation of queries on any database consisting of a finite collection of relations in a simple normal form. A data sublanguage (called ALPHA), established directly on the relational calculus, was informally described in [6]. The equivalence of relational algebra and relational calculus was proved in [9]. An algebra or calculus is relationally complete if, given any finite collection of relations R_1, R_2, …, R_N in normal form, the expressions of the algebra or calculus permit definition of any relation definable from R_1, R_2, …, R_N by alpha expressions [7]. A relational database language SQL (Structured Query Language) based on the relational algebra and calculus was proposed [1, 2, 3, 4].

The relational table in relational data model is based on attributes of entities and their functional dependence. Data are normalized in flat tables. While, the Resource Space Model is based on normalized classifications. Resources are normalized in multi-dimensional spaces where coordinates can be hierarchical. So new query languages are needed for querying resource spaces.

2. Completeness of Resource Space Operations

2.1. Basic Idea

Suppose S is the discussed domain, an operation ***op*** on S is a mapping ***op***: $S\times\ldots\times S\rightarrow S$, ***op***$(s_1, \ldots, s_n)= s$, where s and $s_1, \ldots, s_n$ belong to S. When n=1, ***op*** is an unary operation like Disjoin and Split; when n=2, ***op*** is a binary operation like Join and Merge [11]. In applications, we can only consider unary and binary operations.

Given two sets A and B, if we only consider the set operations between them, then how many operations are sufficient? Experience tells us that three operations — union, intersection and difference are sufficient. But what is the reason? Can we define other operations?

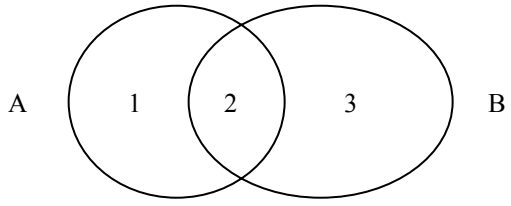

Fig. 2. An example for discussing the sufficiency of set operations

As shown in Fig.2, A and B are divided into three parts according to the distribution of their elements (here we do not consider the simpler cases where some parts are empty). Part 1 consists of elements which are in A but not in B. Part 2 consists of elements which are both in A and B. Part 3 consists of elements which are in B but not in A. If the empty set $\varnothing$ is also considered as a result, then there are totally 2^3=8 required sets, which are: {$\varnothing$, Part 1, Part 2, Part 3, Part 1 and Part 2, Part 1 and Part 3, Part 2 and Part 3, Part 1 and Part 2 and Part 3}, which actually are: {$\varnothing$, A–B, A$\cap$B, B–A, A, A$\oplus$B, B, A$\cup$B}, where $\oplus$ is called symmetric difference [8]. We can see that the operations set {$\cup$, $\cap$, –, $\oplus$} are sufficient, because from A and B, these operations can get all the required results. Among these four operations, only $\cup$ and – are necessary, because $\cap$ and $\oplus$ can be represented by them: A$\cap$B= A– (A–B), and A$\oplus$B= (A–B) $\cup$ (B–A).

This inspires us to explore the theoretical basis for the design and analysis of resource space query languages.

An operation set is called sufficient only when it can get all the required results, and an operation set is called necessary only when it is the smallest sufficient operation set.

2.2. Sufficient and Necessary Operations on Resource Space

Suppose two resource spaces RS_1 and RS_2 have the same number of dimensions, and the corresponding axes are the same under the same domain ontology. Then we can define the operations Union, Difference and Intersection as follows:

Operation 1. *Union* — The union of two resource spaces RS_1 and RS_2 is: $RS_1\cup RS_2=\{(x_1, \ldots, x_n) \mid (x_1, \ldots, x_n)\in RS_1$ or $(x_1, \ldots, x_n)\in RS_2\}$, i.e., the result is a resource space with n axes consisting of resources in points in RS_1 or in RS_2.

Operation 2. *Difference* — The difference of two resource spaces RS_1 and RS_2 is: $RS_1-RS_2=\{ (x_1, \ldots, x_n) \mid (x_1, \ldots, x_n)\in RS_1$ and $(x_1, \ldots, x_n)\notin RS_2\}$, i.e., the result is a resource space with n axes consisting of resources in points in RS_1 but not in RS_2.

Operation 3. *Intersection* — The Intersection of two resource spaces RS_1 and RS_2 is: $RS_1\cap RS_2=\{ R(x_1, \ldots, x_n) \mid R(x_1, \ldots, x_n)\in RS_1$ and $(x_1, \ldots, x_n) \in RS_2\}$, i.e., the result is also a resource space with n axes consisting of resources in points in RS_1 and in RS_2.

Given two operations $\boldsymbol{op_1}$ and $\boldsymbol{op_2}$ on two resource spaces RS_1 and RS_2, we use $\boldsymbol{op_1}(RS_1)$ to represent the result of operating RS_1 by unary operation $\boldsymbol{op_1}$, and use $RS_1\ \boldsymbol{op_2}\ RS_2$ to represent the result of operating RS_1 and RS_2 by binary operation $\boldsymbol{op_2}$. Then, all the resource spaces we can get from the set $\{RS_1, RS_2\}$ by using the set $\{\boldsymbol{op_1}, \boldsymbol{op_2}\}$ can be listed as: $\{\boldsymbol{op_1}(RS_1), \boldsymbol{op_1}(RS_2), RS_1\ \boldsymbol{op_2}\ RS_2, \boldsymbol{op_1}(\boldsymbol{op_1}\ (RS_1), \boldsymbol{op_1}(\boldsymbol{op_1}(RS_2), \boldsymbol{op_1}(RS_1\ \boldsymbol{op_2}\ RS_2), \ldots\}$. Then, the following definition can be given.

Definition 1. Given a set of resource spaces *RSS* and a set of operations *OP* on the resource spaces, RSS^{OP} denotes all the resource spaces that can be got from *RSS* by a sequence of operations in *OP*.

For example, for $RSS=\{RS_1, RS_2\}$, if $OP_s=\{\cup, \cap\}$, then $RSS^{OPs}=\{RS_1, RS_2, RS_1\cup RS_2, RS_1\cap RS_2\}$; if $OP_t=\{\cup\}$, then $RSS^{OPt}=\{RS_1, RS_2, RS_1\cup RS_2\}$. This example is simple, but in many cases, it is not easy to compute RSS^{OP} by given *RSS* and *OP*. For example, for $RSS=\{RS_1, RS_2\}$ and $OP=\{-\}$, one may think that $RSS^{OP}=\{RS_1-RS_2, RS_2-RS_1\}$. But in fact, $RSS^{OP}=\{\varnothing, RS_1, RS_2, RS_1-RS_2, RS_2-RS_1, RS_1\cap RS_2\}$, because $RS_1-(RS_1-RS_2)=RS_1\cap RS_2$ and $RS_1-(RS_1-(RS_1-RS_2))=RS_1-RS_2$.

It is clear that if $OP_s \supseteq OP_t$, then $RSS^{OPs} \supseteq RSS^{OPt}$, which means that defining new operations can get more results. But the definition of new operations is infinite, then how many operations are sufficient enough? The inherent requirements of data model have decided the completeness of operations before hand, when the defined operations set can meet the requirements of data model, then it can be called sufficient.

Intuitively, given any set of resource spaces *RSS*, if an operation set *OP* can get all the required results, then *OP* can be called sufficient. So the definition of a sufficient operation set can be given as follows:

Definition 2. An operation set *OP* of resource space is called sufficient, if for any set of resource spaces *RSS*, all required results are in RSS^{OP}.

Suppose an operation set OP_s is sufficient and OP_t is a real subset of OP_s, if $RSS^{OPs}=RSS^{OPt}$, then OP_t is also sufficient and the operations in OP_s-OP_t are not necessary. Then we can give the definition of a necessary operation as follows:

Definition 3. An operation set OP_s of resource space is called necessary if OP_s is sufficient and there does exist a real subset OP_t of OP_s such that $RSS^{OPs} = RSS^{OPt}$.

For example, when we only consider the traditional set operations, the set $OP_s=\{\cup, -, \cap\}$ is sufficient but not necessary. Because for its real subset $OP_t=\{\cup, -\}$, from $RS_1\cap RS_2= RS_1-(RS_1-RS_2)$, we can get $RSS^{OPs}= RSS^{OPt}$. And the set $\{\cup, -\}$ is necessary because it is the smallest set which is sufficient in this sense.

3. Expressiveness of Different Query Languages

Definition 4. Let *RS*, RS_1 and RS_2 be resource spaces. Two unary operations are called equivalent to each other if $\boldsymbol{op_1}(RS) = \boldsymbol{op_2}(RS)$. Two binary operations are called equivalent to each other if $RS_1\ \boldsymbol{op_1}\ RS_2 = RS_1\ \boldsymbol{op_2}\ RS_2$.

For example, if we define a binary operation '*' as follows:

*: $RS_1 * RS_2= RS_1\cap(RS_1\cap RS_2)$.

Then, '*' is equivalent to the operation '$\cap$', i.e., $*=\cap$. If two operations are equivalent to each other, they are the same from the perspective of mapping, so they are the same operation. As we can see, the operation '*' is composed of operation '$\cap$', then we can say that operation '*' can be represented by operation '$\cap$'. Then, we have the following definition.

Definition 5. Suppose *OP* is an operation set, a unary or binary operation $\boldsymbol{op}$ is called "can be represented by *OP*" if $\boldsymbol{op}(RS)$ or $RS_1\ \boldsymbol{op}\ RS_2$ can be represented as an expression of *OP*.

For example, we have $RS_1\cap RS_2= RS_1-(RS_1-RS_2)$, so operation '$\cap$' can be represented by operation '$-$'. Equivalent and representation are two basic relations between operations discussed here.

The study of expressiveness of operations can answer problems like "whether the defined operations are sufficient". The expressiveness of operations is an abstract concept, it is difficult to be accurately defined or described. Here the comparison between expressiveness is given.

Intuitively, given any resource spaces *RSS*, if operation set OP_s can get more results than OP_t, then we can say that the expressiveness of OP_s is more stronger than OP_t. So a definition can be given as follows:

Definition 6. Given two operation sets OP_s and OP_t, the expressiveness of OP_s is called stronger or weaker than OP_t, denoted by $OP_s>OP_t$ (or $OP_s<OP_t$), if for any *RSS*, $RSS^{OPs} \supset RSS^{OPt}$ (or $RSS^{OPs} \subset RSS^{OPt}$) holds.

Here "the more results" does not mean the whole quantity of data, but the number of different results. For example, for $RSS=\{RS_1, RS_2\}$, $OP_s=\{\cup, \cap\}$ and $OP_t=\{\cup\}$, we have $RSS^{OPs}=\{RS_1, RS_2, RS_1\cup RS_2, RS_1\cap RS_2\}$, $RSS^{OPt}=\{RS_1, RS_2, RS_1\cup RS_2\}$. The whole quantities of data are the resources included by space $RS_1\cup RS_2$, but the operation set OP_s gets one more result $RS_1\cap RS_2$, so we say that the expressiveness of OP_s is stronger than OP_t.

Some characteristics of expressiveness of operations are given in the following.

Characteristic 1. Given two operation sets OP_s and OP_t, both $OP_s>OP_t$ and $OP_s<OP_t$ may not hold.

For example, for $RSS=\{RS_1, RS_2\}$, $OP_s=\{\cap\}$ and $OP_t=\{\cup\}$, we have $RSS^{OPs}=\{RS_1, RS_2, RS_1\cap RS_2\}$ and $RSS^{OPt}=\{RS_1, RS_2, RS_1\cup RS_2\}$. So $RSS^{OPs}\not\subset RSS^{OPt}$ and $RSS^{OPt}\not\subset RSS^{OPs}$, then both $OP_s>OP_t$ and $OP_s<OP_t$ do not hold, which means that we cannot say the expressiveness of which is stronger than the other.

Characteristic 2. Given two different operation sets $\{OP_s\}$ and $\{OP_t\}$, the expressiveness of them can be the same.

For example, for $RSS=\{RS_1, RS_2\}$, $OP_s=\{\cup, -\}$ and $OP_t=\{\cup, -, \cap\}$, we have $RSS^{OPs}=\{\varnothing, RS_1-RS_2, RS_1\cap RS_2, RS_2-RS_1, RS_1, RS_1\oplus RS_2, RS_2, RS_1\cup RS_2\}=RSS^{OPt}$, so the expressiveness of them are the same.

Characteristic 3. Given two operation sets OP_s and OP_t, if $OP_s\supseteq OP_t$ then $OP_s>OP_t$.

Characteristic 4. If $OP_r>OP_s$ and $OP_s>OP_t$, then $OP_r>OP_t$.

Characteristic 5. Given an operation set OP_s, if OP is equivalent to or can be represented by some operations in OP_s, then $OP_s>OP$.

Characteristic 6. If $OP_s>OP_t$, then $(OP_s\cup OP_t)=OP_s$.

Characteristic 6 shows that if newly defined operations can be represented by existing operations, then the expressiveness of operations does not increase in essence. This provides an approach for us to measure the expressiveness of a set of operations or semantic factors.

4. Design of Resource Operating Languages

4.1. Definition of Operations

Apart from the traditional set operations defined above, we can define the following operations.

Operation 4. *Extended Cartesian Product* — The Extended Cartesian Product of two resource spaces $RS_1(X_{11}, \ldots, X_{1n})$ and $RS_2(X_{21}, \ldots, X_{2m})$ is a resource space with $n+m$ axes. The preceding n axes are the axes of RS_1 and the following m axes are axes of RS_2. If RS_1 has k_1 points and RS_2 has k_2 points, then the Extended Cartesian Product of RS_1 and RS_2 has $k_1\times k_2$ points, we denote it as $RS_1\times RS_2=\{(x_{11}, \ldots, x_{1n}, x_{21}, \ldots, x_{2m}) \mid (x_{11}, \ldots, x_{1n})\in RS_1 \text{ and } (x_{21}, \ldots, x_{2m})\in RS_2\}$.

Operation 5. *Selection* — It is for selecting the points that satisfying given conditions in the Resource Space RS, denoted as $\sigma_F(RS)=\{t \mid t\in RS \text{ and } F(t)=\text{'true'}\}$, where F, a logical expression representing the selection conditions, has binary value 'true' or 'false'. The logic expression F is composed of the logic operators $\neg$, $\wedge$ and $\vee$ connecting every arithmetic expression. In fact, the operation 'selection' is to select the points that make the logic expression F be true from the Resource Space RS.

The operations Join, Disjoin, Merge and Split have been defined in [10] as follows:

Operation 6. *Join* — Let $|RS|$ be the number of the dimensions of the RS. If two resource spaces RS_1 and RS_2 store the same type of resources and have n ($n\geq 1$) common axes, then they can be joined together as one resource space RS such that RS_1 and RS_2 share these n common axes and $|RS|=|RS_1| + |RS_2| - n$. RS is called the join of RS_1 and RS_2, denoted as $RS_1\cdot RS_2\Rightarrow RS$.

According to the above definition, all the resources in the result resource space RS come from RS_1 and RS_2 and can be classified by more axes. The Join operation provides an efficient method for the management of resources defined in different resource spaces.

Operation 7. *Disjoin* — A resource space RS can be disjoined into two resource spaces RS_1 and RS_2 that store the same type of resources as that of RS such that they have n ($1\leq n\leq \min(|RS_1|, |RS_2|)$) common axes and $|RS|-n$ different axes, and $|RS|=|RS_1| + |RS_2| - n$ (denoted as $RS\Rightarrow RS_1\cdot RS_2$).

The Disjoin operation can clarify the classification of resources by separating large number of axes into two small ones. Both Join and Disjoin operations keep 1NF, 2NF and 3NF of Resource Space Model.

Operation 8. *Merge* — If two resource spaces RS_1 and RS_2 store the same type of resources and satisfy: (1) $|RS_1| = |RS_2| = n$; and (2) they have n-1 common axes, and there exist two different axes X' and X'' satisfying

the merge condition, then they can be merged into one RS by retaining the $n-1$ common axes and adding a new axis $X^*=X' \cup X''$. RS is called the merge of RS_1 and RS_2, denoted as $RS_1 \cup RS_2 \Rightarrow RS$, and $|RS|= n$.

Operation 9. *Split* — A resource space RS can be split into two resource spaces RS_1 and RS_2 that store the same type of resources as RS and have $|RS| -1$ common axes by splitting an axis X into two: X' and X'', such that $X=X' \cup X''$. This split operation is denoted as $RS \Rightarrow RS_1 \cup RS_2$.

By the split operation, the unconcerned coordinates on a certain axis can be filtered out and only the interesting coordinates are preserved.

4.2. Verification of Operations

To define a sufficient and necessary operation set is enough in theory. But in applications, some new operations which can be represented by existing operations will also be defined for the convenience of expression or operation. For example, from the Join operation, we can naturally introduce another useful operation: Division. And we can define another operation Projection from the operation Disjoin. The definition of new operations could be infinite if we neglect the practical requirements.

Theorem 1. There exist infinite different operations.

Proof. According to definition 4, we only need to show that there exist infinite operations which are not equivalent to each other. We define a sequence of operations $\{\Theta_1, \Theta_2, \Theta_3, \ldots\}$ as:

Θ_1: $rs_1\ \Theta_1\ rs_2 = (rs_1 \cup rs_2) \times (rs_1 \cap rs_2)$,

Θ_2: $rs_1\ \Theta_2\ rs_2 = (rs_1\ \Theta_1\ rs_2)\ \Theta_1\ (rs_1\ \Theta_1\ rs_2)$,

……

Θ_{i+1}: $rs_1\ \Theta_{i+1}\ rs_2 = (rs_1\ \Theta_i\ rs_2)\ \Theta_1\ (rs_1\ \Theta_i\ rs_2)$,

……

From our definition, we have:

$rs_1\ \Theta_2\ rs_2 = (rs_1\ \Theta_1\ rs_2)\ \Theta_1\ (rs_1\ \Theta_1\ rs_2)$

$= ((rs_1 \cup rs_2) \times (rs_1 \cap rs_2))\ \Theta_1\ ((rs_1 \cup rs_2) \times (rs_1 \cap rs_2))$

$= (((rs_1 \cup rs_2) \times (rs_1 \cap rs_2)) \cup ((rs_1 \cup rs_2) \times (rs_1 \cap rs_2))) \times (((rs_1 \cup rs_2) \times (rs_1 \cap rs_2)) \cap ((rs_1 \cup rs_2) \times (rs_1 \cap rs_2)))$

$= ((rs_1 \cup rs_2) \times (rs_1 \cap rs_2)) \times ((rs_1 \cup rs_2) \times (rs_1 \cap rs_2))$

$= (rs_1\ \Theta_1\ rs_2) \times (rs_1\ \Theta_1\ rs_2)$.

So we can see that $\Theta_2= \Theta_1 \times \Theta_1$. And we conjecture that $\Theta_i= \Theta_{i-1} \times \Theta_{i-1}$ for any $i \geq 2$.

$rs_1\ \Theta_i\ rs_2 = (rs_1\ \Theta_{i-1}\ rs_2)\ \Theta_1\ (rs_1\ \Theta_{i-1}\ rs_2)$

$= ((rs_1\ \Theta_{i-1}\ rs_2) \cup (rs_1\ \Theta_{i-1}\ rs_2)) \times ((rs_1\ \Theta_{i-1}\ rs_2) \cap (rs_1\ \Theta_{i-1}\ rs_2))$

$= (rs_1\ \Theta_{i-1}\ rs_2) \times (rs_1\ \Theta_{i-1}\ rs_2)$.

So we have $\Theta_i= \Theta_{i-1} \times \Theta_{i-1}$ for any $i \geq 2$. Then, it is clear that operations $\{\Theta_1, \Theta_2, \Theta_3, \ldots\}$ are not equivalent to each other, so they are infinite different operations. □

Theorem 1 shows that finding a "self-contained" operation set regardless of its applications is impractical.

Are the nine operations we defined above sufficient or not?

Firstly, we show that operation set {Union, Difference, Intersection} is not sufficient. For example, suppose the resource spaces considered are $\{RS_1(X_1, X_2), RS_2(X_1, Y_2)\}$, then it is clear that space $RS_3(X_1, X_2, Y_2)$ is in the required results. But from $\{RS_1(X_1, X_2), RS_2(X_1, Y_2)\}$, the operations {Union, Difference, Intersection} cannot get the space $RS_3(X_1, X_2, Y_2)$. It is because the precondition of these traditional set operations is that the operated spaces have the same dimensions, and the result spaces also have the same dimension. So from two 2-dimensional spaces, we cannot get a 3-dimensional space. Then, it is clear that operation set {Union, Difference, Intersection} is not sufficient.

Then, we show that the nine operations {Union, Difference, Intersection, Extended Cartesian Product, Selection, Join, Disjoin, Merge, Split} defined above are sufficient.

Theorem 2. The nine operations Union, Difference, Intersection, Extended Cartesian Product, Selection, Join, Disjoin, Merge and Split are sufficient for query resource space.

Proof. For query in resource spaces, we only consider the information in single spaces or the correlations between spaces, so we can decide all the required results of the query operations of Resource Space Model. Given any finite collection of resource spaces $RS_1, RS_2, \ldots, RS_N$ in simple normal form, any required results are in the form of $\{(x_1, \ldots, x_d) | x_k \in RS_i (X_j), 1 \leq i \leq N, d \geq 1$ and $1 \leq k \leq d\}$, i.e., all the combinations of the coordinates of the resource spaces.

For many resource spaces, we should find the correlations between them. For a single resource space, the smallest unit is a coordinate of one point. So we should find any set of points in a space, and locate any of their coordinates. So we can use the operation Selection to choose the desired points, then use Disjoin to get any of their coordinates. Then, the set operations Union, Difference and Intersection can get the combinations of them. So the operations Selection,

Disjoin, Union, Difference and Intersection can get all the combinations of the coordinates of a single resource space RS_i. Then using the Extended Cartesian Product, we can get all the combinations of the coordinates of these finite resource spaces RS_1, RS_2, …, RS_N. □

In the proof process of theorem 2, we can see that the five operations: Selection, Disjoin, Union, Difference and Extended Cartesian Product are sufficient and necessary.

5. Conclusions

This paper investigates the completeness of resource space query languages, and establishes a theoretical basis for determining how complete a selection capability is provided in a proposed resource operation sublanguage. An operations set can be called sufficient only when it can get all the required results of the data model. In this sense, a necessary operations set is the smallest sufficient set. Based on this, we establish a framework to compare the expressiveness of different resource sublanguages. Finally, we design a set of resource query operations and verify their completeness. This result is significant in directing the design of a resource space operation sublanguage.

The proposed approach can be used in the study of the expressiveness and completeness of the interconnection semantics, for example, a set of primitive semantic links [13].

Acknowledgement: The authors thank all team members of China Knowledge Grid Research Group (http://www.knowledgegrid.net) for their help and cooperation.

6. References

[1] ANSI, 1986. The Database Language SQL. Document ANSI X3.315.

[2] R. Boyce, D. D. Chamberlin, M. Hammer, et al. "Specifying Queries as Relational Expressions", *Communications of the ACM*, 18: 11, November 1975.

[3] D. D. Chamberlin, R. Boyce, "SEQUEL: A Structured English Query Language", *Proc. ACM SIGMOD Workshop on Data Description. Access and Control*, May 1974.

[4] D. D. Chamberlin, et al. "SEQUEL 2: A Unified Approach to Data Definition, Manipulation and Control", *IBM Journal of Research and Development*. 20, No. 6, November 1976.

[5] E. F. Codd, "A Relational Model of Data for Large Shared Data Banks", *Communications of the ACM*, 13 (6) (1970) 377-387.

[6] E. F. Codd, "A Data Base Sublanguage Founded on the Relational Calculus", *Proc. ACM-SIGFIDET Workshop on Data Description. Access, and Control*, November 1971.

[7] E. F. Codd, "Relational Completeness of Data Base Sublanguages in Data Base Systems", *Courant Computer Science Symposia Series*, Vol. 6, 1972.

[8] R. S. Robert, *Set Theory and Logic*, Courier Dover Publications, Oct 1979.

[9] J. Ullman, *Principles of Database Systems*, Second Edition, Computer Science Press, 1982.

[10] H. Zhuge, "Resource Space Grid: Model, Method and Platform", *Concurrency and Computation: Practice and Experience*, 16 (14) (2004) 1385-1413.

[11] H. Zhuge, "Resource Space Model, Its Design Method and Applications", *Journal of Systems and Software*, 72 (1) (2004) 71-81.

[12] H. Zhuge, *The Knowledge Grid*, World Scientific, 2004.

[13] H. Zhuge, The Open and Autonomous Interconnection Semantics, Keynote at 8th International Conference on Electronic Commerce, Canada, August, 2006.