The Web Resource Space Model

Hai Zhuge
Chinese Academy of Sciences

Springer

Contents

Preface

Birds of a feather flock together. Web resources of a category work closely for efficiency.

Classification is a method of efficiently managing various resources. It is also a basic method for human beings to know the real world and synthesize experience.

A Web Resource Space Model (in simple RSM) is a semantic data model for specifying, storing, managing and locating Web resources by appropriately classifying the contents of resources. It enables users or applications to operate resources by the SQL-like query language.

A Web resource space is a multi-dimensional classification space where dimensions are discrete. Its intrinsic characteristics are worth studying as it is not an ordinary distance space. A resource space can be normalized to increase the correctness of resource management by setting constraints on dimensions.

Aiming at a Web semantic data model with characteristics of normalization and autonomy, this book develops the RSM systematically in methodology, model and theory. It concerns the following contents:

1. The general methodology of the RSM, which includes the origin, fundamental concepts, characteristics and the development method. It helps understand the RSM and design resource spaces for applications.
2. The relationship between the Resource Space Model and the Semantic Link Network. The integration of the two models forms a richer semantic data model to support advanced distributed applications.
3. The completeness and necessity theory for query operations of the Resource Space Model.
4. The algebra and calculus theory for query operations of the Resource Space Model. The query capability and expressive power of the Resource Space Model are studied from two perspectives: resource space algebra and resource space calculus.

5. The complexity of searching the resource space. We intend to unveil the relationship between the searching efficiency and the number of dimensions as well as the relationship between the searching efficiency and the distribution of coordinates.
6. The physical storage mechanism of the resource space. Its multi-dimensional and discrete characteristic is different from the relational database index (one dimensional) and previous multidimensional index (sequential numerical dimensions).
7. The P2P-based decentralized resource space. It is an approach to enabling the Resource Space Model to synergy normalization with autonomy. A structured P2P resource space solution and an unstructured P2P resource space solution are studied.
8. The probabilistic Resource Space Model, which enables users or applications to store and retrieve resources uncertainly. It is a more general Resource Space Model.

I would like to take this opportunity to thank my students Erlin Yao, Yunpeng Xing, Xiang Li, Chao He and Liang Feng who make important contribution to this work. Thanks also go to all team members of the China Knowledge Grid Research Group for their help and cooperation.

Research work was financially supported by the National Basic Research Program of China (2003CB317000), the EU 6th Framework Project GREDIA (IST-FP6-034363), and the International Cooperation Program of the Ministry of Science and Technology of China (2006DFA11970).

The Web resource space is a part of the resource space we live in. The RSM is a promising model for effective management of versatile resources. Integrating the RSM with the Semantic Link Network, the database models and the Web ontology mechanisms could form a powerful semantic platform for the future interconnection environment.

Hai Zhuge
August 9, 2007

Chapter 1 Resource Space Model Methodology

Classification is not only an approach to efficiently managing resources but also a basic method for human to know the real world.

The Web Resource Space Model realizes birds of a feather flock together in the digital interconnection environment. It specifies and manages various Web resources by normalizing classification on contents of resources.

1.1 Origin of the Resource Space Model

Files are expanding in our daily-use PCs or laptops due to easier download from websites and email attachments. Management of these accumulating files is troublesome if we arbitrarily save them. For example, saving files in the desktop of Windows seems convenient, but this will cause inefficient retrieval of files and slow down the speed of machine in the long run. Appropriately naming folders to contain various files is a way to efficient file management and retrieval. Inappropriately naming will cause trouble in retrieval of files due to synonym or poor meaning.

Goods in supermarkets are arranged by classification. Goods are placed and updated according to the commonsense of the classification shared between customers and sellers. Chain supermarkets often arrange goods in a uniform style and place good categories in the same order so that customers can quickly reach their targets. This order is an experience for customers to raise the efficiency in selecting goods by going directly to the interested category or region. We can also find that some closely relevant goods are arranged in neighborhood. This neighbor information also helps customers to select goods. These strategies bring efficiency and convenience for sellers to manage goods and for customers to select goods.
Biologists classify organisms into categories by judging degrees of apparent similarity and difference. On discovering an unknown organism, scientists begin their classification by looking for anatomical features that have the same function as those found on other species, and then determine

whether or not the similarities are due to an independent evolution or to descent from a common ancestor. If the latter is the case, then the two species could be classified into the same category.

Children start to learn concepts by classifying real-world objects into categories and by generalizing and specializing categories via instances. So classification is not only an approach to efficiently managing resources but also a basic method for human to know the real world.

A (*Web*) *Resource Space Model* (*in simple RSM*) *is a semantic data model for specifying, storing, managing and locating contents of* (*Web*) *resources by appropriate classification on the contents of resources.*

The notion of the Web resource space was initiated in 2002 as a multi-dimensional knowledge space (Zhuge, 2002). Its basic model was proposed in 2004 (Zhuge, 2004a-d).

A resource space is a multi-dimensional classification space where dimensions (axes) are discrete so it is different from ordinary distance space. Its intrinsic characteristics are worth studying.

A resource space can be normalized to ensure the correctness of managing resources by setting constraints on axes.

The Resource Space Model methodology is a study of the basic method for designing resource spaces and helping the development of its applications.

Data model and algorithm are the core of software systems.

File system is the first milestone towards effective computing resource management. It is a method for storing, managing and retrieving resources in form of files by establishing mapping between a directory indexing structure and a storage device. The directory indexing structure can keep track of the files and path syntax required to access them. It defines the way files are named as well as the maximum size of a file or volume. A file system is a component of most operating systems. IT professionals have used various indexing techniques to raise the efficiency of managing data in files.

The file system can be regarded as a 1-dimensional resource space.

Database is another milestone towards effective resource management. Most databases use file systems as the underlying mapping mechanism between the higher level indexes and the storage devices. Database theories and systems have influenced the world for forty years (Bachman, 1969). Especially, the Relational Database Model and systems have achieved a great success (Codd, 1970). The Relational Database Model uses relational tables to describe basic relations between data types. Its normal form theory ensures the correctness of data operations.

Both the file system and the database system are invented at the age of mainframe and centralized computing.

The World Wide Web is a huge decentralized file system. There is no central control for adding, updating and removing Web pages. The management of the Web resources challenges traditional data models.

Object-oriented databases and object-relational databases extend the application scope of the relational databases by borrowing the advantages of the object-oriented methodologies and programming languages like inheritance and encapsulation to model the real-world and they enable complex objects to be effectively managed (Kim, 1990; Rumbaugh et al., 1991; Mok, 2002). But, their limitations emerge in Web-based applications, which require resources to be managed in an open, decentralized, platform-irrelevant and content-based way.

In data warehousing and OLAP area, the multi-dimensional data model was used. It is suitable for storing large-scale historical data. To support decision-making based on large data sets, data mining techniques are needed to discover the rules in large data sets (Han and Kambr, 2000). But, the read-only model is limited in ability to meet the needs of resource management in the Internet environment where resources are complex and have to be frequently operated. Compared with the relational data model, it is weak in theory on data management.

Fig. 1.1 depicts a file system on disk. The file system realizes the mapping from the directories and files of various type onto the disk space by establishing the index on the linear disk space. Users or up-level applications can operate files according to their names and path regardless of their physical storage.

Fig. 1.2 shows the keyword index on the World Wide Web. The Web is actually a decentralized file system, which enables users to browse Web pages by their URLs. Search engines establish indeces on Web pages distributed on the Web by collecting and classifying Web pages according to keywords and then recommending Web pages sharing the same set of keywords according to the page rank (S.Brin and L.Page, www7.scu.edu.au,1998).

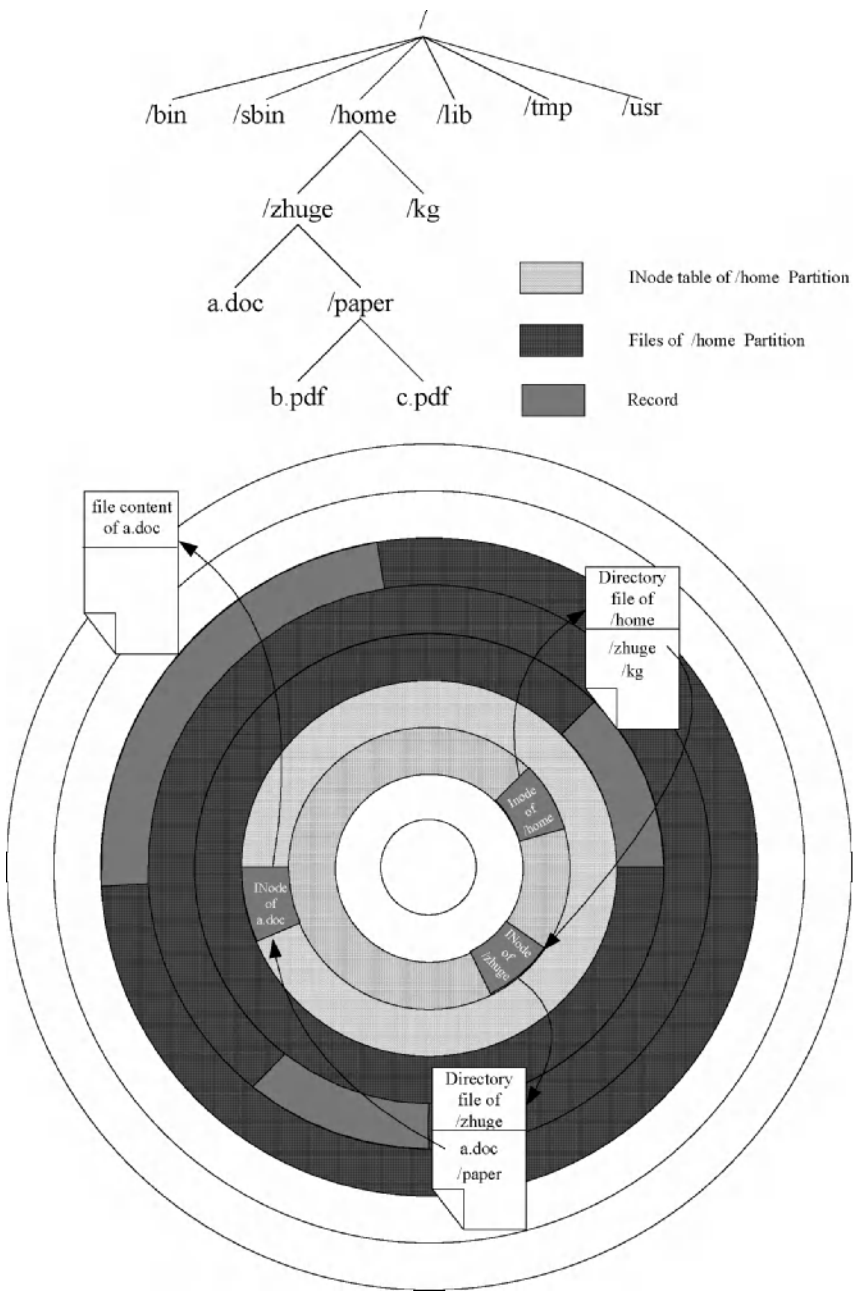


Fig.1.1. The EXT2/3 file system.

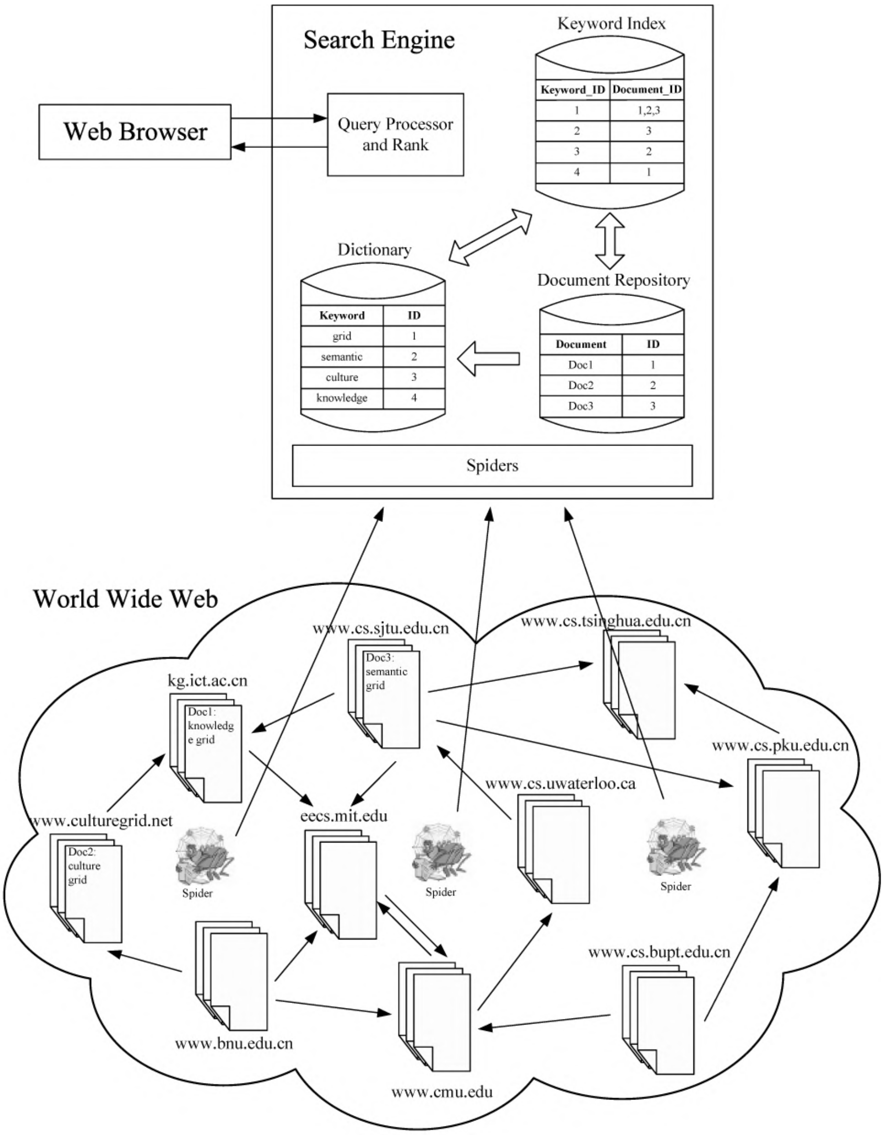


Fig.1.2. Index on the World Wide Web.

Comparison of the file system on disk and the file system on the World Wide Web implicates the evolution of file systems.

With the development of Web applications, effective management of the contents of various resources on the Web becomes a challenge. The new generation data model should be a semantic-rich model that is able to manage content rather than file name. But to precisely describe the content of an individual resource is difficult and a formal description is not easy for sharing among people of different communities. The Resource Space Model can reflect the content of resources by classification semantics.

People often use the classification method to manage various contents in daily life. For example, researchers are organized into research groups according to the research topics they are working on. Publishers classify their products (books, journals and conference proceedings) into different categories according to disciplines and contents.

A set of resources can be classified by different classification methods as shown in the left hand of Fig. 1.3. Multiple classification methods constitute a multi-dimensional semantic space over a set of resources as shown in the right hand of Fig. 1.3, where each axis represents a kind of classification method.

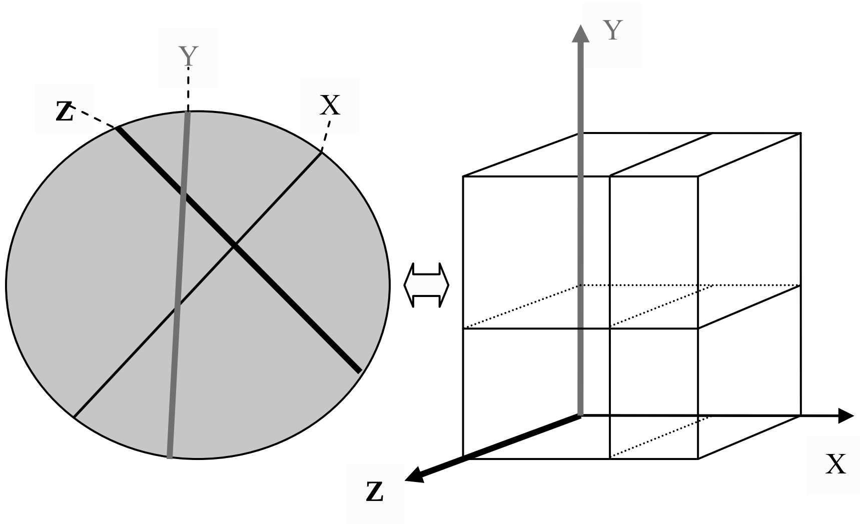


Fig. 1.3. Multiple classification methods over a set of resources constitute a multi-dimensional classification space.

The Resource Space Model is such a semantic space that manages information, knowledge and service resources. Information resources refer to various types of electronic files that can be transmitted through the Internet, and can be read or perceived directly or indirectly. Knowledge resources include metadata, relation and strucuture as well as the abstract concepts, axioms, rules and methods that can be represented in a certain machine-understandable form. Knowledge can be generated from understanding the information resources or generalizing human experience. Service resources refer to the reusable capability processes for performing tasks, solving problems, or processing information or knowledge resources.

A resource space can be presented in:

1. *conceptual* or *logical aspect*— the definition of an *n*-dimensional resource space;
2. *user view aspect*— a subspace of the whole resource space displayed in user-understandable form;
3. *representation aspect*— the cross-platform understandable definition based on standard description languages like XML, RDF and OWL; and,
4. *storage aspect*— the physical storage of the resource space including the storage of the space structure, relevant index and resource entities.

The characteristics of the Resource Space Model require a specific method to help design resource spaces. Before presenting the design method, we first introduce the basic notion and components of the Resource Space Model.

1.2 Basis of the Resource Space Model

1.2.1 Definitions and Characteristics

The semantic basis of the Resource Space Model is *name space*, *basic data type*, *set* and *partition*.

Concepts are labeled in the name space as consensus of a community, while their semantics are determined by classification and use-case in four worlds — the real world, the document world, the machine world and the mental world (Zhuge et al, 2006). Some basic concepts in the name space do not need to be explained. They can be used to define other concepts. A set of concepts can represent a certain semantics. Classifications on concepts form concept hierarchies.

The basic semantic elements of the Resource Space Model are *resource*, *resource space*, *axis* and *coordinate*.

A resource in the machine world has name, type and content. Concepts are basic elements of the composing content. For example, a Web page has a URL, and its content can be represented by a set of concepts.

A *Resource Space* is a multi-dimensional classification space. It consists of a name and a set of axes, denoted as $RS(X_1, X_2, \ldots, X_n)$. Each *axis* X_i represents a classification method. X_i is partitioned by a set of coordinates denoted as $X_i = <C_{i1}, C_{i2}, \ldots, C_{im}>$.

A point in the space, determined by one coordinate at every axis, represents a set of resources of the same category.

An axis can be regarded as a 1-dimensional resource space.

A *coordinate* C represents a set of resources, denoted as $R(C)$. Resources represented by axis X_i are the union of all the resources represented by its coordinate: $R(X_i)= R(C_{i1})\cup R(C_{i2}) \cup\ldots\cup R(C_{in})$. The semantics of a coordinate is represented by name, basic datatype, a set of concepts, or a coordinate tree (low-level coordinates finely classify their common ancestor). The semantics of a coordinate is regulated by the semantics of its axis.

A coordinate C is called independent from another coordinate C' if there is no intersection between $R(C)$ and $R(C')$. Using existing taxonomy and commonsense as the classification method is a way to establish concensus between designers and users.

The resource space, axis, coordinate, and point are sets in nature.

A coordinate regulates a set of points. An axis regulates a set of coordinates. An axis name represents higher classification level than its coordinates. A resource space regulates a set of axes and the refined classification relationship. A resource is determined by locating the point it belongs to and by selecting from the resource set according to its name and content description.

Two axes can be regarded as equivalent if their names are the same in semantics and the names of all the corresponding coordinates are the same in semantics.

The following are two operations on axis:

1. If two axes $X_1 = <C_{11}, C_{12}, \ldots, C_{1n}>$ and $X_2 = <C_{21}, C_{22}, \ldots, C_{2m}>$ have the same axis name but have different coordinates, then they can be merged into one: $X= X_1 \cup X_2 =<C_{11}, C_{12}, \ldots, C_{1n}, C_{21}, C_{22}, \ldots, C_{2m}>$. In this case, the name of X represents X_1 and X_2.

2. An axis X can be split into two axes X' and X'' by dividing the coordinate set of X into two: the coordinate set of X' and the coordinate set of X'', such that $X= X' \cup X''$. If the two axes need to be merged for the future use, the names of X' and X'' should be the same in semantics.

The semantics of axis and coordinate can be formally defined or informally defined. For example, the semantics of a coordnate can be defined by a set of concepts, which regulate the semantics of the resources it may contain. The above definitions enable a resource space to represent any form of resources.

If we use a set of domain concepts K_C to describe a coordinate C, and the resources contained by C share common concept set K_R, then we could find a mapping between K_C and K_R such that corresponding concepts are the same or share a common ancestor. This mapping is useful in automatically classifying resources.

Let $X=(C_1, C_2, \ldots, C_n)$ be an axis and C_i' be a coordinate at another axis X', we say that X *finely classifies* C_i' (denoted as C_i'/X) if and only if:

1. $(R(C_k)\cap R(C_i')) \cap (R(C_p)\cap R(C_i')) = \text{NULL}$ ($k\neq p$, and $k, p\in[1, n]$); and,
2. $R(C_1)\cap R(C_i')\cup R(C_2)\cap R(C_i') \cup \ldots \cup R(C_n)\cap R(C_i') =R(C_i')$ hold.

As the result of fine classification, $R(C')$ is partitioned into n categories: $R(C_i'/X) =\{R(C_1)\cap R(C_i'), R(C_2)\cap R(C_i'), \ldots, R(C_n)\cap R(C_i')\}$.

For two axes $X= (C_1, C_2, \ldots, C_n)$ and $X' = (C_1', C_2', \ldots, C_m')$, we say that X *finely classifies* X' (denoted as X'/X) if and only if X finely classifies C_1', C_2' ,…, and C_m' respectively.

Two axes X and X' are called *orthogonal* with each other (denoted as $X \perp X'$) if X finely classifies X' and vice versa, i.e., both X'/X and X/X' hold.

Establishing orthogonal relationship between relevant classifications deepens people's understanding on the real world.

The following three normal forms are for designing a good resource space.

1. A *first-normal-form* resource space is a resource space and there does not exist name duplication (semantic overlap) between coordinates at any axis.
2. A *second-normal-form* resource space is a first-normal-form, and for any axis, any two coordinates are independent from each other.
3. A *third-normal-form* resource space is a second-normal-form and any two axes of it are orthogonal with each other.

The following are four operations of the resource space:

1. **Join operation**. If two resource spaces RS_1 and RS_2 specify the same type of resources and they have n ($n \geq 1$) common axes, then they can be *joined* into one resource space RS such that RS, RS_1 and RS_2 share these n common axes and $|RS|=|RS_1| + |RS_2| - n$, where $|RS|$ represents the number of dimensions of the RS. RS is called the *join* of RS_1 and RS_2, denoted as $RS_1 \cdot RS_2 \Rightarrow RS$.
2. **Disjoin operation**. A resource space RS can be *disjoined* into two resource spaces RS_1 and RS_2 (denoted as $RS \Rightarrow RS_1 \cdot RS_2$) that specify 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 $|RS|=|RS_1| + |RS_2| - n$.
3. **Merge operation**. If two resource spaces RS_1 and RS_2 specify 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_1 and X_2 satisfying the merge condition, then the two spaces can be *merged* into one RS by retaining the $n-1$ common axes in the new space and including a new axis $X=X_1 \cup X_2$. RS is called the merge of RS_1 and RS_2, denoted as $RS_1 \cup RS_2 \Rightarrow RS$, and $|RS|= n$. The second condition can be extended as follows: they have $n-k$ common axes ($1 \leq k < n$), and there exists one-one mapping between the rest k axes of the two spaces such that the merge condition can be satisfied, then the two spaces can be *merged* into one RS by retaining the $n-k$ common axes in the new space and including k new axes, each of which is the union of the corresponding axes.
4. **Split operation**. A resource space RS can be *split* into two resource spaces RS_1 and RS_2 that store the same type of resources as that of 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

$RS \Rightarrow RS_1 \cup RS_2$. (In contrast to the merge operation, this split operation can be extended to the case that RS_1 and RS_2 have $|RS| - k$ common axes by splitting every of the k axes into two.)

Several strategies can be adopted to realize the join operation. One strategy is that only those resources specified in both original resource spaces are reserved in the new resource space. The join operation would result in many empty nodes. Therefore, the join and merge operations are usually used for generating views of existing resource spaces. If RS_1 and RS_2 are 3NF, then RS is a 3NF according to the normal form theory of the Resource Space Model.

The Resource Space Model is equipped with an SQL-like resource operation language to support operations on resource space. The basic operations are introduced in The Knowledge Grid (Zhuge, 2004d).

Efficient resource management depends on the human behavior mode of dealing with resources and the mode of storing resources in the resource space. The degree of matching between the two modes determines the efficiency.

Fig.1.4 depicts the interaction between the human behavior and the resource space.

Storing resources in the right category with high probability leads to better retrieval result. The query language bridges the mutual understanding between the behavior modes and the resource storage. The semantic mechanism like domain ontology helps explain the output resources and the input on storage and query.

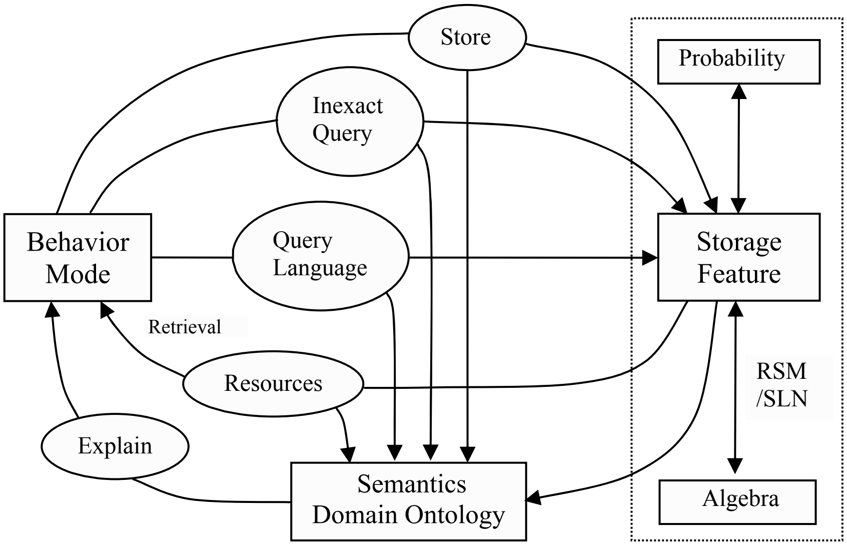


Fig. 1.4. Interaction between the human behavior mode and the resource storage mode.

1.2.2 Resource Space Definition Language

The Resource Space Definition Language RSDL defines the commands for specifying and modifying resource spaces, in particular, the schemas for resource spaces.

A resource space can be created by the following command, where *RS* is the name of the resource space, X_i is the name of its axis, C_{ij} is the coordinate of X_i, and the *URSL* is the location of the resource space. The integrity constraints set constraints on axes to ensure the correctness of operations.

CREATE SPACE *RS* (X_1, X_2, …, X_n) [**AT** *URSL*]
WHERE $X_1 = \{C_{11}, \ldots, C_{1u}\}, \ldots, X_n = \{C_{n1}, \ldots, C_{nv}\}$
    <*integrity constraint*$_1$>

……

<*integrity constraint*$_m$>

The drop command is for deleting a resource space including all the indices and schemas.

DROP RSPACE *RS*

The modify command is used on an existing resource space to add or drop axes or coordinates. For example, an axis can be added to a resource space by using the following command, where *RS* is the name of an existing resource space, $axis_i$ is the name of the axis to be added, and $<C_{i1}, \ldots, C_{ij}>$ is its coordinate list.

MODIFY SPACE *RS*
ADD AXIS $X_i <C_{i1}, \ldots, C_{ij}>$

The axes of a resource space can be listed by using the following command, where *RS* is the name of an existing resource space.

USING *RS* **LIST AXES**

Similarly, the coordinates of a given axis in a resource space *RS* can be listed by using the following command:

USING *RS* **LIST COORD OF AXIS** *X*

1.2.3 Resource Space Manipulation Operations

The Merge operation makes resource spaces $RS_1, \ldots,$ and RS_n at $URSL_1, \ldots,$ and $URSL_n$ respectively into one resource space *RS* subject to any specified conditions and places the new resource space at *URSL*. It can be represented by the following command, where $|RS_i|$ is the number of axes of resource space RS_i and $X_{ik}(RS_j)$ represents that axis X_{ik} of resource space RS_j is to be merged (i=1, ..., n). The constraint clause specifies *common_axis_number* as the constraint name. The predicate of the check clause is the constraint of the merge operation.

MERGE $RS_1, \ldots, RS_n$ [**AT** $URSL_1, \ldots, URSL_n$]
INTO RS [**AT** $URSL$]
WHERE *new_axis* $(RS) = X_{1\mu}(RS_1) \cup \ldots \cup X_{n\nu}(RS_n)$
CONSTRAINT *axis_number*
    CHECK $|RS_1| = \ldots = |RS_n| = |RS|$
CONSTRAINT *common_axis_number*
    CHECK *number* (*common_axes*) $= |RS| - 1$.

The split operation separates a resource space RS at $URSL$ into each RS_i at each $URSL_i$. The axis X of RS will be separated into X_{1i} (RS_1), …, and $X_{nj}(RS_n)$. The Split command is represented as follows, where the check clause requires that no coordinate or axis be removed in the split operation.

SPLIT RS [**AT** $URSL$]
INTO $RS_1, \ldots, RS_n$ [**AT** $URSL_1, \ldots, URSL_n$]
WHERE X (RS) **JOIN-INTO** X_{1i} (RS_1) = <*coordinate_set*$_1$> **&** …
        & X_{nj} (RS_n) = <*coordinate_set*$_n$>
CONSTRAINT *axis_split*
CHECK $X\,(RS) = X_{1i}\,(RS_1)$ □…□$X_{nj}\,(RS_n)$

If resource spaces RS_1, …, and RS_n store the same type of resources and they have k ($k \in [1, \text{minimum}(|RS_1|, |RS_2|)]$) common axes, then they can be joined into one resource space RS such that RS_1, …, and RS_n share these k common axes and $|RS| = |RS_1| + |RS_2| - k$. The join operation can be represented as follows:

JOIN $RS_1, \ldots, RS_n$ [**AT** $URSL_1, \ldots, URSL_n$]
INTO RS [**AT** $URSL$]
WHERE COMMON AXES $(axis_1, \ldots, axis_\mu)$
CONSTRAINT *common_axis_number*
CHECK *number* (*common_axes*) $\leq |RS| - 1$

If two resource spaces RS_1 and RS_2 store the same type of resources and have n ($n = |RS_1| = |RS_2|$) common axes, they can be united into one resource space RS by eliminating duplicates. RS is called the union of RS_1 and RS_2, and $|RS| = n$. The union operation requires that the resource spaces to be united have the same number of axes and the same axis names. It can be represented as follows:

UNION $RS_1, \ldots, RS_n$ [**AT** $URSL_1, \ldots, URSL_n$] **INTO** RS
CONSTRAINT *axis_number*
  CHECK $|RS_1| = \ldots = |RS_n| = |RS|$
CONSTRAINT *common_axis_number*
  CHECK *number* (*common_axes*) = $|RS|$.

1.2.4 Resource Space Modification

A newly created resource space has no resources. The following command can insert resources into the resource space.

INSERT $R_1\ldots, R_m$ **INTO** $RS_1\ldots, RS_m$ [**AT** $URSL_{1,} \ldots , URSL_m$]
  [**WHERE** <*conditional expression*>]

Instead of specifying a resource set directly, we can use a select statement to extract a set of resources as follows.

INSERT INTO RS <$axis_1$, $axis_2$, $axis_3$>
COORD <$coord_1$, $coord_2$, $coord_3$>
BY SELECT $A_1, A_2, \ldots, A_n$
FROM $RS_1, RS_2, \ldots , RS_m$
[**WHERE** <*conditional expression*>]

If the specified resource exists at the specified point of the given resource space and the user has the authority to delete it, then it can be deleted by the following statement:

DELETE R **FROM** $RS_1, \ldots, RS_m$ [**AT** $URSL_{1,}\ldots, URSL_m$]
  [WHERE <*conditional expression*>]

The following update statement is used to change a resource index in a given resource space.

UPDATE *RS*
REPLACE R_1 **WITH** R_2
  [**WHERE** <*conditional expression*>]

1.2.5 View Definition

To define a view of a resource space, the name of the view as well as the query that computes the view is required. The view can be defined by the following command, where <*query expression*> is any valid query expression. The view name is represented by *v*.

CREATE VIEW *v* **AS** <*query expression*>

As an example, consider the view consisting of *m* axes of a resource space. We can define view *RS–view* as follows:

CREATE VIEW *RS–view* ($axis_1$, …, $axis_m$) **AS**
SELECT $X_1 =< c_{11},\ldots,c_{1,k_1} >,\ldots,X_m =< c_{m,1},\ldots,c_{m,k_m} >$
FROM *RS*

The list of axis names can be omitted. We define a view over two resource spaces by using the merge operation as follows, where the new axis $axis_m$ is formed by $X_m \cup Y_m =< C_{X_m,1},\ldots,C_{X_m,i},C_{Y_m,1},\ldots,C_{Y_m,j} >$.

CREATE VIEW *RS–view* ($axis_1$, …, $axis_m$) **AS**
SELECT $X_1, X_2,\ldots,X_m < C_{X_m,1},C_{X_m,2},\ldots,C_{X_m,i} >$ **FROM** RS_1
MERGE
SELECT $Y_1, Y_2,\ldots,Y_m < C_{Y_m,1},C_{Y_m,2},\ldots,C_{Y_m,j} >$ **FROM** RS_2
WHERE $X_1 = Y_1$, …, $X_{m-1}=Y_{m-1}$
AND X_m.*axis_name* = Y_m.*axis_name*.

We can define a view combining two resource spaces by joining the subspaces selected from them as follows:

CREATE VIEW *RS–view* ($axis_1$, …, $axis_m$) **AS**
SELECT $X_1, X_2,\ldots,X_m < C_{X_m,1},C_{X_m,2},\ldots,C_{X_m,i} >$ **FROM** RS_1
JOIN

SELECT $Y_1, Y_2, \ldots, Y_n < C_{Y_n,1}, C_{Y_n,2}, \ldots, C_{Y_n,j} >$ **FROM** RS_2
WHERE $X_1 = Y_1, \ldots, X_i = Y_i$ ($i \le \mathrm{minimum}(m,n)$)

1.2.6 Query Language

The following three clauses are components of a query:

1. The **SELECT** clause lists the resource attributes required in the answer.
2. The **FROM** <*RS* ($X_1, X_2, \ldots, X_m$)> clause specifies the resource space *RS* to be used in the selection, where X_i is an axis.
3. The **WHERE** <*conditional expression*> clause conditions the answer in terms of coordinates of resources and semantic relationships between the coordinates.

A typical query takes the following form, where A_i represents a feature of the destination resource, and R_i names a source resource space. A **SELECT** * clause specifies that all attributes of all resources appearing in the **FROM** clause are to be selected. The compound name *resource_space.attribute* avoids ambiguity when an attribute appears in more than one resource space. However if an attribute appears in only one of the resource spaces in the **FROM** clause, the *resource_space* qualifier can be omitted.

SELECT $A_1, A_2, \ldots A_n$
FROM $R_1, R_2, \ldots R_m$
WHERE <*conditional expression*>

Query can focus on either one point in the space or an area specified by more than one coordinates at one axis. An area consists of a set of points.

The *ACM Computing Classification System* can be constructed as a normalized 3-dimensional information space: *ACM–CCS* (*Category*, *Publication*, *Letter*). The query "Find all journal papers in the resource space ACM–CCS which relate to Resource Space Model" can be expressed as follows:

SELECT * **FROM** *ACM–CCS*

WHERE *Category* ="Resource Space Model" **&** *Publication* = "Journal"

1.2.7 Visualized Resource Locating

Locating resources is the basic operation of the Resource Space Model. Users can accurately locate a set of resources by giving coordinates on every axis. The underlying premise is that users know the structure of the space or that users' viewpoint consists with the space designers on classification of resources. A visualized resource locator provides users with intuitive knowledge on the underlying structure of the resource space.

Fig.1.5 shows a three-dimensional visualized resource locator for exhibiting Dunhuang culture. Points in the space correspond to the small cubes in the 3-dimensional large cube. Each side of the cube can display image, text and links. Users can see the details by clicking them. The operations are arranged as buttons in the up-row.

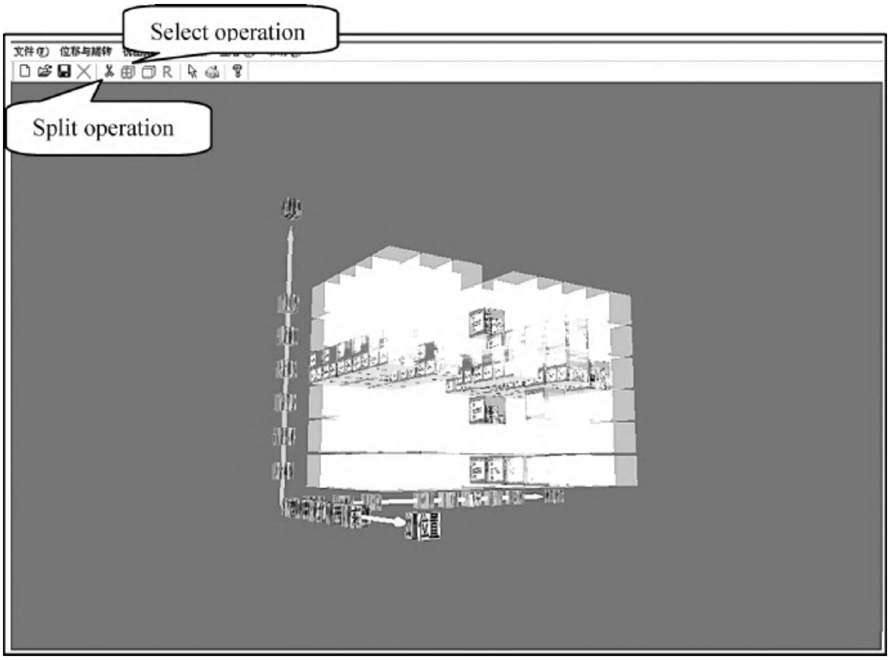
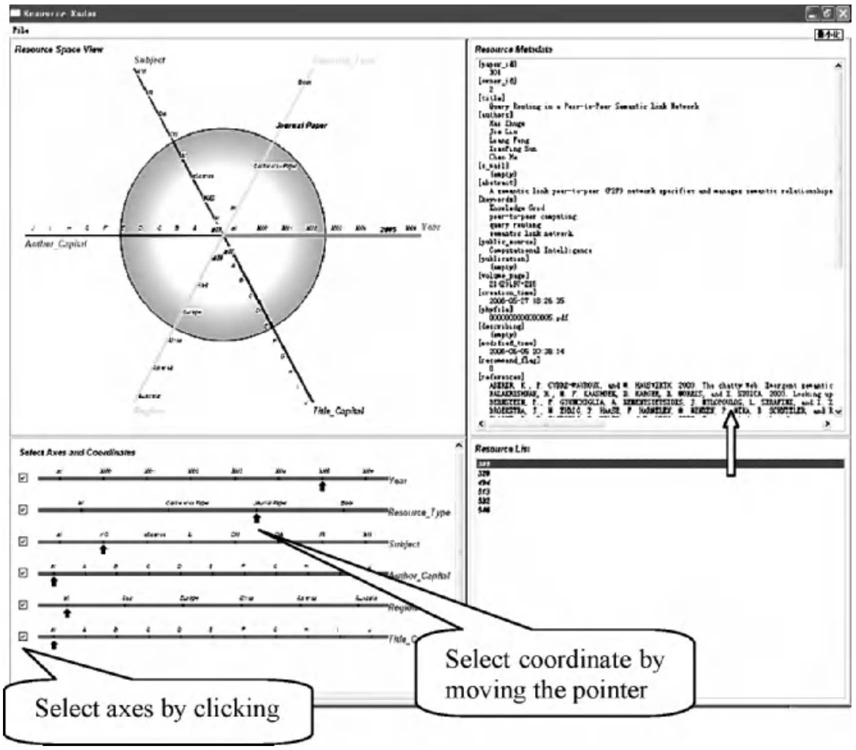


Fig. 1.5. The visualized 3-dimensional resource space locator.

Fig.1.6. shows an *n*-dimensional visualized resource locator. Users can create a view by selecting axes from the *n*-dimensional resource space.



```
[paper_id]
   301
[owner_id]
   2
[title]
   Query Routing in a Peer-to-Peer Semantic Link Network
[authors]
   Hai Zhuge
   Jie Liu
   Liang Feng
   Xiaoping Sun
   Chao He
[e_mail]
   (empty)
[abstract]
   A semantic link peer-to-peer (P2P) network specifies and manages semantic relationships
[keywords]
   Knowledge Grid
   peer-to-peer computing
   query routing
   semantic link network
[public_source]
   Computational Intelligence
[publication]
   (empty)
[volume_page]
   21(2):197-216
[creation_time]
   2006-05-27 18:26:35
[pdffile]
   000000000000005.pdf
[describing]
   (empty)
[modified_time]
   2006-06-05 20:38:14
[recommend_flag]
   0
[references]
   ABERER, K., P. CUDRE-MAUROUX, and M. HAUSWIRTH. 2003. The chatty Web: Emergent semantic
   BALAKRISHNAN, H., M. F. KAASHOEK, D. KARGER, R. MORRIS, and I. STOICA. 2003. Looking up
   BERNSTEIN, P., F. GIUNCHIGLIA, A. KEMENTSIETSIDIS, J. MYLOPOULOS, L. SERAFINI, and I. Z
   BROEKSTRA, J., M. EHRIG, P. HAASE, F. HARMELEN, M. MENKEN, P. MIKA, B. SCHNIZLER, and R.
```

```
319
329
494
513
532
546
```

Fig. 1.6. Visualized *n*-dimensional resource locator.

1.3 Application Scenarios of the Resource Space Model

1.3.1 Management of Web Pages

The World Wide Web enables human to use browsers to display Web pages and jump from one page to another via hyperlink. How to efficiently index and manage the huge Web pages is an important issue.

The content of a Web page can be classified by topics such as news, finance, sport and health. Each topic can be further classified by time and language. This naturally corresponds to the classification characteristics of the Resource Space Model. A 3-dimensional resource space (*topic*, *time*, *language*) shown in Fig. 1.7 can be used to manage Web pages. The point (*Chinese*, *2006-12-13*, *finance*) in the space contains a set of Web pages on finance (URLs). The resource space can provide classification semantics and distance measure between points that relational database cannot directly provide for better retrieval and learning on the Web.

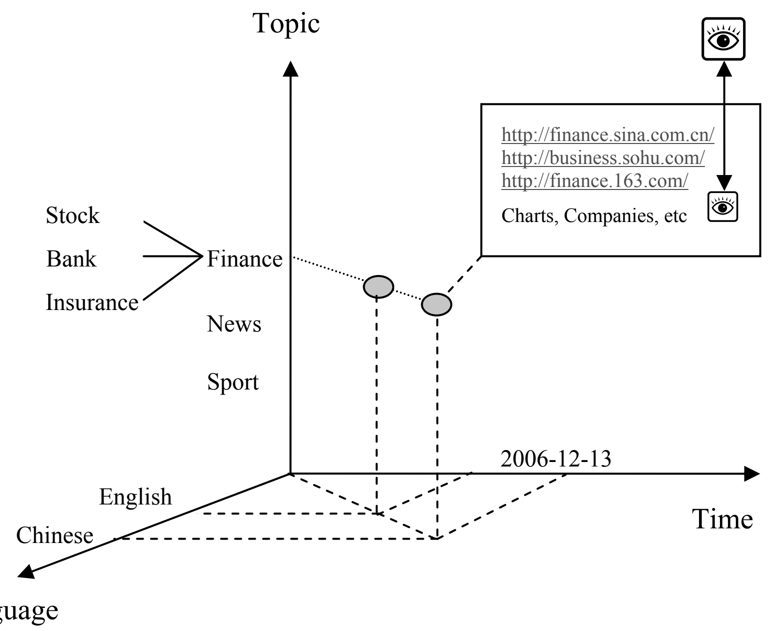


Fig. 1.7. A 3-dimensional resource space for managing the content of Web pages.

Entering into the point, users can obtain the required Web pages on finance — they do not need to input URL multiple times nor jump between URLs. The point includes more relevant statistic information on economics. The finance coordinate can be divided into finer coordinates, for example *stock*, *bank* and *insurance* information. This refinement can express more detailed information when specifying resources, and can refine users' interest when retrieving resources. The Resource Space Model provides a new way to store and express Web resources.

The hierarchical classification characteristic of the resource space supports the management of a hierarchical Web structure from the hierarchical surface Web content to the underlying databases.

1.3.2 Managing Multi-layer Tables

Multi-layer tables provide integrated information of multiple abstraction levels from different analysis. The higher layers provide more abstract information. The lower layers constitute a fine classification of the higher layer, for example, *professor*, *associate professor* and *assistant professor* constitute a fine classification of the *teacher* class. Table 1.1 shows such a table on university human resources.

The traditional relational data model excludes this type of tables since the first normal form of relational data model requires the flat table and atomic fields. Many relational tables are needed to decompose the multi-layer table if we use relational data model to realize the management of human resources with this form.

The table naturally corresponds to a 2-dimensional resource space as shown in Fig. 1.8. A point represents a set of persons of certain department and certain rank. Moreover, a *name* axis in alphabetical order can be added to form a 3-dimensional resource space.

Table 1.1. A multi-layer table of university human resources.

			School of Science			School of Engineering			School of Business	
			Dept. of Mathematics	Dept. of Physics	Dept. of Chemistry	Dept. of Chemical Engineering	Dept. of Computer Science & Engineering	Dept. of Mechanical Engineering	Dept. of Accounting	Dept. of Economics
Academic Staff	Professor									
	Associate Professor									
	Assistant Professor									
Student	Graduated	PhD								
		MPhil					University Human Resources			
	Under-graduate									
Support Staff										
Visiting Staff										

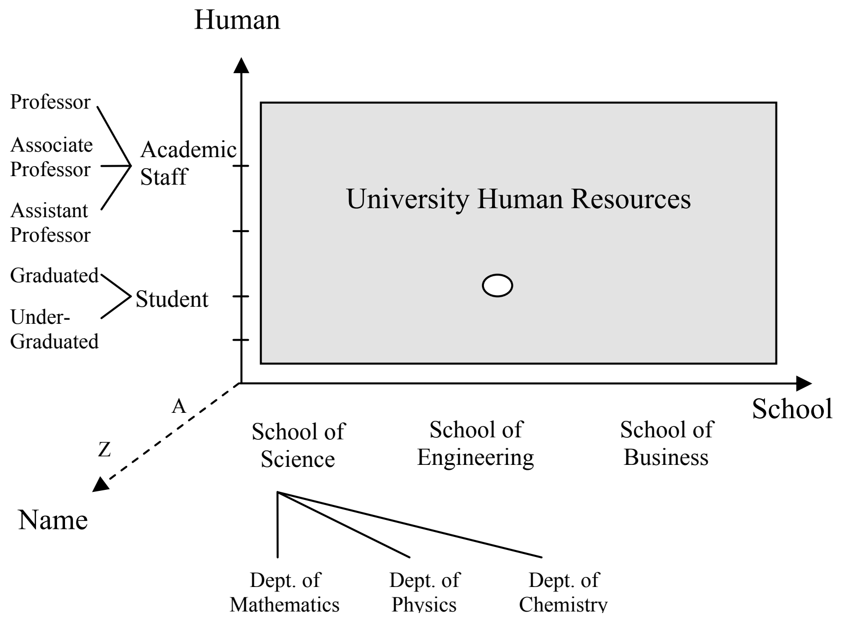


Fig. 1.8. A resource space for specifying university human resources.

The more layers the table has, the more advantages of the resource space model shows.

1.3.3 Management of Photos

A software tool that can efficiently manage and locate photos is very useful. Usually, we store photos in folders of a file system (the directories of a file system actually constitutes a 1-dimensional resource space). To efficiently locate resources, we can choose database systems to record information about photos such as place, time, and the path of storing photos.

However, people concern content of photos rather than their names when retrieving. The contents can be classified into three categories: *human*, *artifact* and *nature*. It can also be classified by *time* and *place*. So photos' content can be specified by a 3-dimensional resource space as shown in Fig. 1.9. Each coordinate can be a coordinate tree, for example, coordinate *China* can be classified into *Beijing*, *Shanghai*, *Xi'an*, etc.

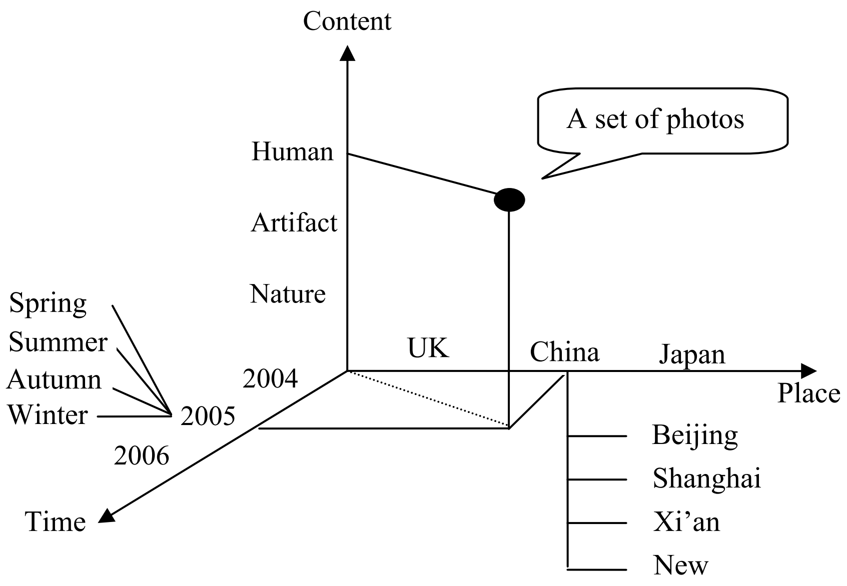


Fig.1.9. A 3-dimensional resource space for specifying photos.

This application proposes a new requirement: adding new coordinates during use, as we cannot estimate future visiting places and photos are added to the resource space after visiting.

New coordinates can be added to the resource space if its original normal forms can be retained. But a Resource Space Model system needs to update its schemas at all levels in this case.

1.3.4 Geographical Resource Space

A geographical Resource Space system can reflect multiple content layers over the same region, for example, geographical, ecological, economical and social information of the same region determined by longitude and latitude as shown in Fig. 1.10.

Every point (regional information) can further define a resource space specifying details, for example, the 3-dimensional resource space (*population*, *religion*, *occupation*). The system can display the statistical data about the population of different religions and the population of different occupation in various charts.

This example shows a characteristic of the resource space — the single semantic entry point of machine-understandable and human-understandable content.

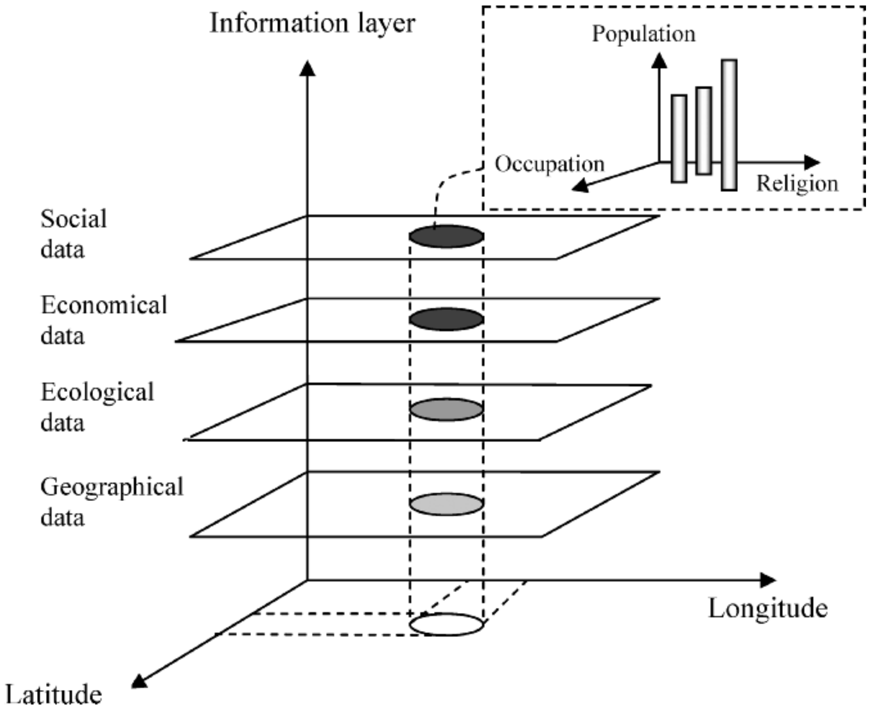


Fig.1.10. An embedded resource space for layered geographical information.

1.3.5 Multi-dimensional ACM Computing Classification System

The Resource Space Model can be used to reform the HTML-based ACM Computing Classification System into a normalized 3-dimensional information space: (*Category*, *Publication*, *Letter*) as shown in Fig.1.11.

The space satisfies the third normal form. The category axis contains eleven categories marked by the letters from "A" to "K", each of which corresponds to a coordinate hierarchy. Each coordinate at the category axis corresponds to a 2-dimensional slice (*Publication*, *Letter*), so users could retrieve the required information according to the publication types and/or the alphabet sequence in the given category. This feature is not provided by the existing classification system.

The Resource Space Model enables information retrieval in a 3-dimensional space. For the purpose of raising the retrieval efficiency, we can add a new axis: *topic*=(*methodology*, *theory*, *application*, *product*) to refine the space.

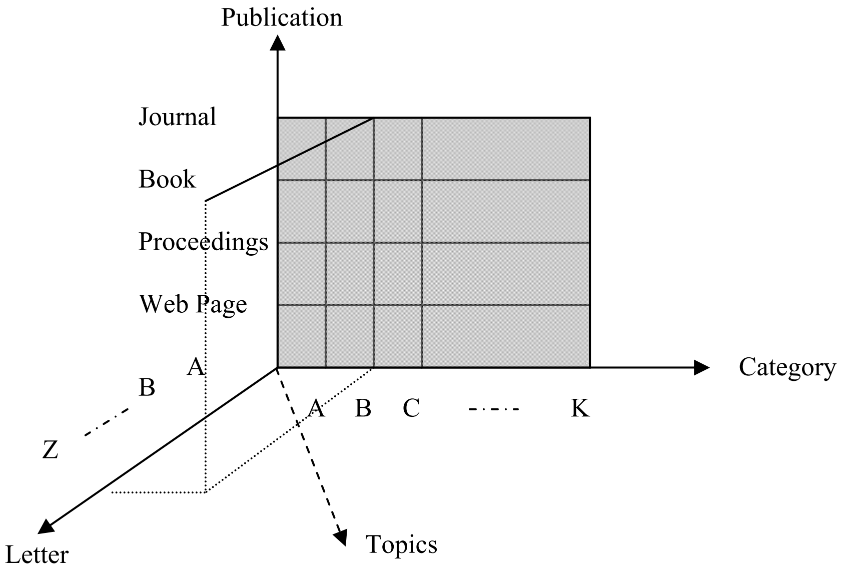


Fig. 1.11. The resource space of normalizing the "ACM Computing Classification System".

1.3.6 Management of Bio-information

The bio-information on the Web is currently managed by versatile databases developed by different countries. The Resource Space Model can be used to reform the existing bio-information retrieval and management systems. All the bio-information databases can be uniformly and normally specified in the resource space. Bio-information can be specified by a two-dimensional resource space as shown in Fig.1.12, where "PubMed", "Structure", "Genome" and "PopSet" respectively stand for: biomedical literature, macromolecular structure, complete genome assemblies, and population study data sets.

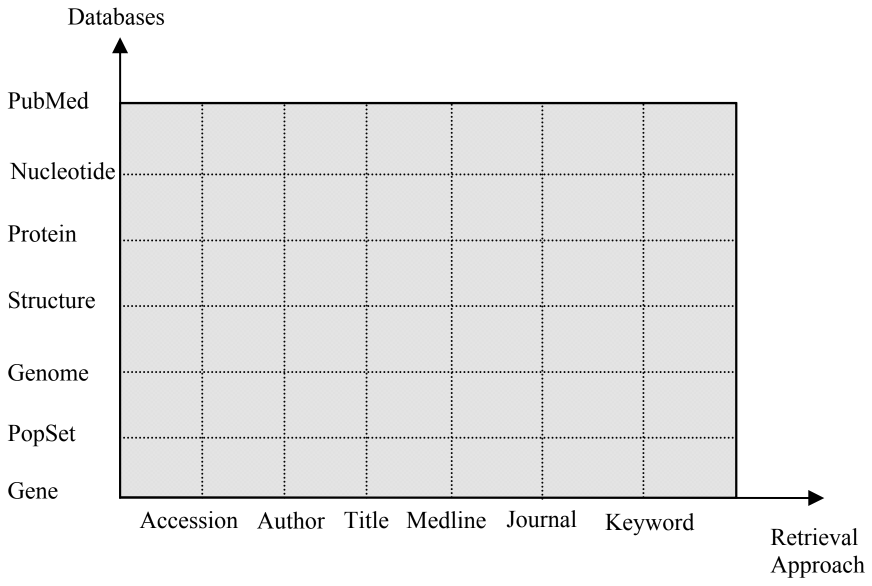


Fig. 1.12. A 2-dimensional resource space for uniformly and normally managing bio-information.

1.3.7 Media Content Space

A four-dimensional Dunhuang cave content space can be designed by classifying the cave content according to the following four axes: *dynasty*, *artifact type* (*wall painting*, *color statue*, *calligraphy*), *media type* (*text*, *video*, *image*) and *cave number* as shown in Fig. 1.13. The dynasty axis can be classified by the following sequential coordinates: *Tang*, *Song*, *Yuan*, *Ming*, and *Qing*. Any coordinate can be refined, for example, the *Tang* dynasty can be further divided into three sequential stages: *early*, *middle* and *late*. Given a set of coordinates on every axis, a set of contents can be accurately located.

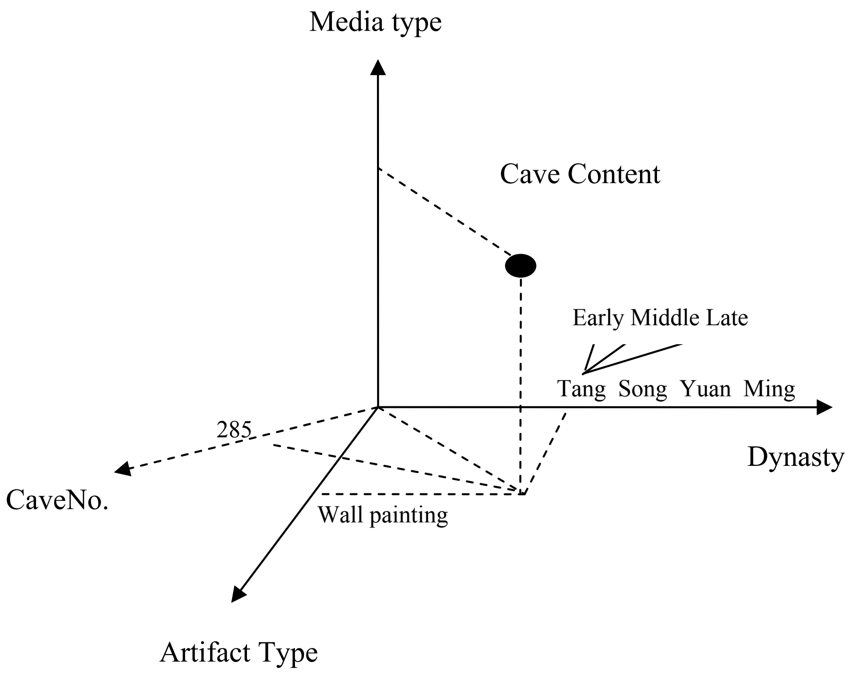


Fig. 1.13. A resource space for normalizing media content.

1.3.8 Automatically Add New Resources to the Resource Space

A resource has external feature and internal feature. The internal feature reflects the content of resources. The external feature helps distinguish one resource from the other resources. Usually the internal feature is not easy to be accurately obtained or expressed, while the external features can be accurately captured.

There could be no intersection between the external feature and the internal feature. For example, the publication date does not reflect the content of a paper.

A resource space represents the designer's viewpoint of classification on resources. If its user has stored some resources in the space, the resource space containing resources also reflects the user's classification viewpoint. Such classification viewpoint can help classify new resources.

The approach to automatically adding resources to resource space varies with the features of resources. To automatically add new papers to the resource space depends on the comparison between the external features and internal features of the new paper and the features of the point.

The keywords of papers in a point of the existing resource space represent authors' viewpoint on the classifications of the contents of papers. Information such as the publisher and the publication type (journal or proceeding) available from the publishers' website is the external feature.

The following process helps add new papers to the paper resource space.

1. Extract keywords from the papers in every point of the resource space, and unite a point *p*'s keyword set with its coordinates to form a set K_p. An empty point K_p only consists of the coordinates of *p*. Information retrieval techniques like TF-IDF can help find important words in text (Salton, 1989; Salton, 1991).
2. Extract the keywords from the new paper and put them into set K_I representing the internal features.
3. Obtain the external features about the new paper such as journal name, publisher, publishing date and impact factor, and put them into set K_E. The main external features are usually available from the metadata of journals.
4. Unite the external features and the internal features on the new paper $K=K_I \cup K_E$.
5. For every point *p* in the space, compare *K* with K_p, if *K* can best match K_p according to some criteria (e.g., share a certain number of keywords in the meaning of domain ontology), the paper is likely to match the point and therefore put the new paper into the point, otherwise put it into the candidate pool awaiting additional techniques.

The citation relation reflects a kind of content inheritance relation. The citing and cited papers in the known classification can help determine the classification of a paper. If a paper in a point is cited by or cites the new paper, then the two papers probably belong to the same category. The more papers in a point cite or are cited by the new paper, the higher probability of the new paper belongs to that point.

We observed two phenomena from experiments of using TF-IDF to extract keywords and match the new paper and the point:

1. The more resources evenly distributed in the resource space, the better the effect of automatically adding new resources to the space; and,
2. The best effect (approximately 80%) can be reached when the weight of the external features reaches 90%.

That is to say, the external features play a more important role.

Fig.1.14 depicts the way to automatically classify papers by using the resource space.

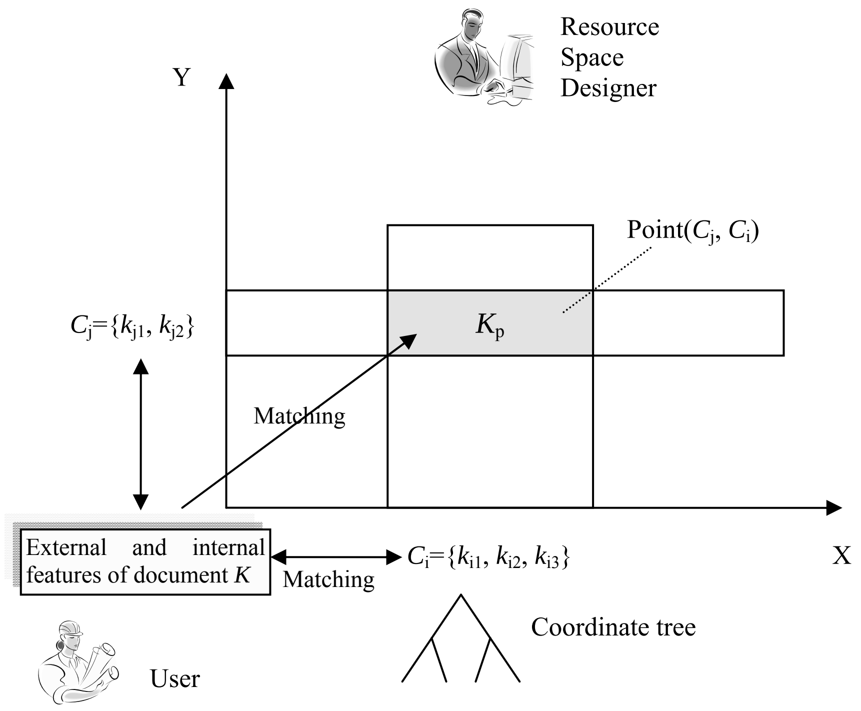

Resource Space Designer
Y
Point(C_j, C_i)
C_j={k_{j1}, k_{j2}}
K_p
Matching
External and internal features of document K
Matching
C_i={k_{i1}, k_{i2}, k_{i3}}
X
Coordinate tree
User

Fig. 1.14. Using resource space to automatically classify documents.

1.4 Design Method

Designing a resource space for application depends on the following three technical factors:

1. knowledge on the Resource Space Model,
2. domain knowledge, and,
3. design experience.

A resource space design consists of the following steps:

1. resource analysis;
2. top-down resource partition;
3. design low dimensional resource spaces like 2- or 3-dimensional resource spaces;
4. increase dimensionality by joining resource spaces or adding a new dimension to the existing resource space according to application requirement;
5. decrease dimension by splitting a resource space according to application requirement; and,
6. check normal forms of the resource spaces.

1.4.1 Resource Analysis

Resource analysis is to determine the application scope, to know the resources to be managed, and then to specify the resources in a *Resource Dictionary*.

Resources usually share some common attributes, for example {*name*, *author*, *owner*, *abstract*, *version*, *location*, *privilege*, *access-approach*, *effective-duration, semantic relevant resources*}. The *abstract* attribute (can be formal or informal) represents content abstraction, or the function description of a service.

Access privilege concerns:

1. *public* — any user can access to it;
2. *group* — only group members can access to it; and,
3. *private* — only the creator can access to it.

The Resource Dictionary enables designers to collect and edit resources and finally forms the axes of the resource space by defining resource classification hierarchy. A Resource Dictionary includes the following operations:

1. *Consistency checking* — check the semantic consistency between descriptions.
2. *Redundancy checking* — check redundancy and delete the redundant descriptions and redundant resources.
3. *Classification* — classify resources according to the specialization relationship determined by existing taxonomy, existing classification standard, available domain ontology, and user judgement.

1.4.2 Top-down Resource Partition

Yin-Yang is a representative of traditional Chinese understanding of how things are formed and work. Yin represents the following abstract concepts generalized from the real world: dark, passive, downward, cold, contracting and weak, while Yang represents the following abstract concepts generalized from the real world: bright, active, upward, hot, expanding and strong. Yin and Yang represent two energies that cause everything to happen. Yin and Yang are meta concepts in ancient Chinese philosophy.

Different epistemologies have different meta concepts.

Due to the difference of epistemology and culture, designers could have different partition solutions on the same set of resources, so a uniform viewpoint on resource partition is needed. The first step is to reach a top-level partition consensus.

Human, *information* and *natural* (*or artificial*) *object* can be the resource partition of human society at the epistemological level. The top-level resource partition of a domain is a special case of this epistemological level, for example, the top-level resources of an institute can be classified as three independent categories:

1. *human resources*,
2. *information resources*, and
3. *service resources* (including facilitates).

Each category can be refined until the categories are small enough for applications as shown in Fig.1.15. (Zhuge, 2004b).

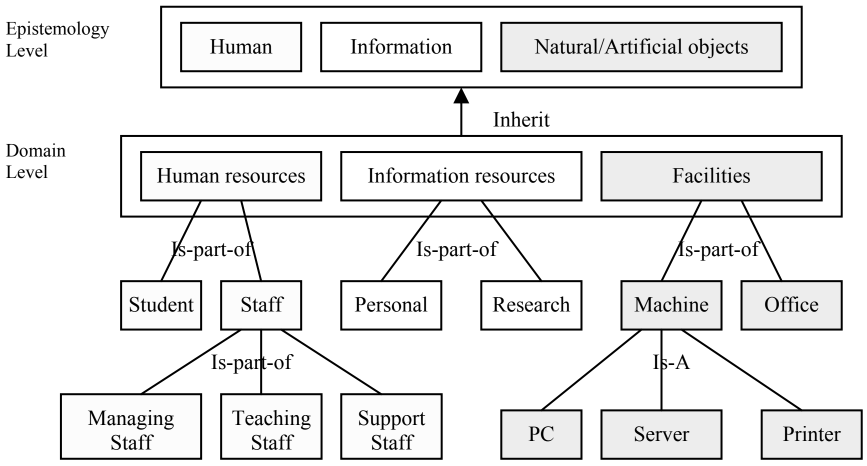


Fig. 1.15. An example of top-down resource partition.

1.4.3 From Low Dimension to High Dimension

Designers can easily handle low-dimensional spaces. So we can first design low dimensional resource spaces then add a new dimension to the existing space or integrate low-dimensional spaces into higher dimensional resource spaces.

A multi-layer table like Table 1.1 is suggested to help design a 2-dimensional resource space.

The design process is as follows:

1. *Determine the number of resource spaces* according to the number of top-level resource categories in the domain. For example, three resource spaces can be established for human resources, inheritance resources and faciliate resources.

2. *Determine axes' names* according to the resource categories at the universal level or domain level. An axis name represents a category of the domain-level partition.
3. *Determine the first-level coordinate names*. Each coordinate reflects one of the categories of the axis.
4. *Determine the coordinate hierarchies*. For each first-level coordinate, determine its low-level coordinates top-down until the basic category according to the resource partition hierarchy. The granularity of the basic category depends on the application requirement on resource retrieval.
5. *Check independency between coordinates*. If the independency is not satisfied, re-consider resource-partition at this level and adjust coordinates.
6. *Check orthogonality between axes*. If the orthogonality is not satisfied, re-consider the coordinate settings, and then adjust relevant coordinates.

According to the above process, designers can construct two resource spaces as shown in Fig.1.16 for the resource partition example of Fig.1.15.

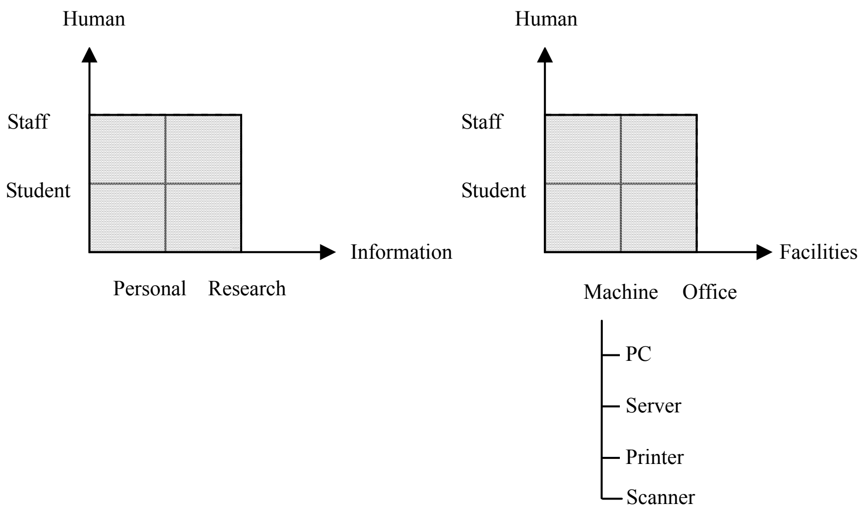


Fig. 1.16. Example of designing 2-dimensional resource spaces.

The Resource Space Model allows existing spaces to be joined into one to achieve the effect of the global resource view. Whether we need to join multiple resource spaces into one depends on application requirement.

To implement the join operation, we need to check the condition of the join operation first. It is important to ensure that different spaces to be joined to specify the same type of resources.

For example, the two resource spaces of Fig.1.16 share a common axis and specify the same type of resources. So they can be joined into one 3-dimensional resource space as shown in Fig.1.17.

The join operation may create new spaces, e.g., joining two two-dimensional resource spaces will generate one three-dimensional resource space where there exists a new two-dimensional space. The points in the new space should be defined together with the new space. For non-empty resource spaces, the following two strategies can be adopted:

1. Place the common resources in the old points sharing common coordinates of different spaces into the new points and keep the rest resources in the original space (i.e., the subspace of the new space); and,
2. Keep old resources in the original spaces (i.e., the subspace of the new space) and establish the mapping between the new points and the old points.

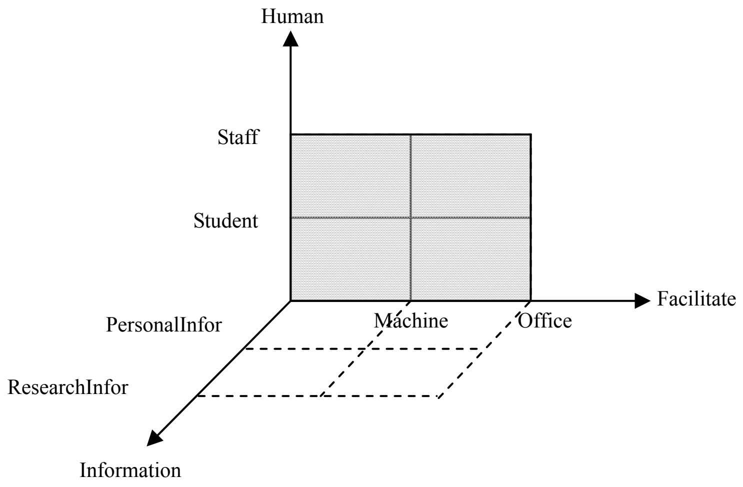


Fig. 1.17. A 3-dimensional resource space constructed by joining two 2-dimensional spaces.

Another example of creating a high dimensional space from two low dimensional spaces is shown in Fig. 1.18. The two 3-dimensional resource spaces specifies the same type of resources—human; and, they share two axes—*Journal* and *Responsibility*. So they can be joined into a 4-dimensional resource space to finely classify the resources.

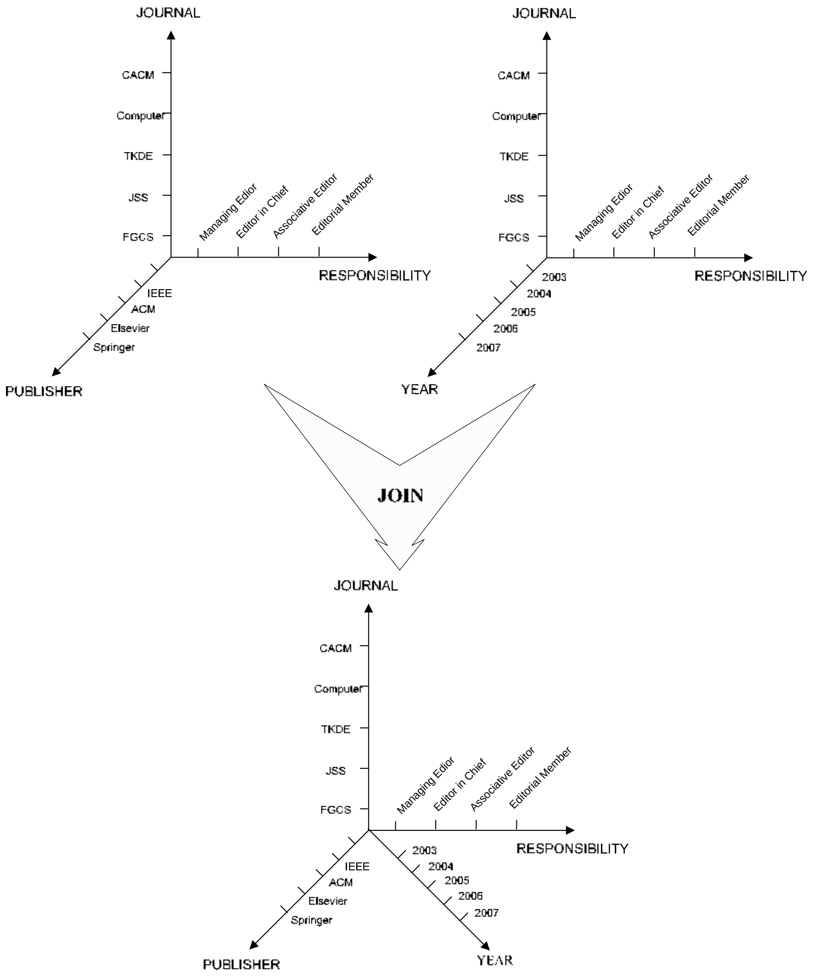


Fig.1.18. Example of creating high dimensional space from low dimensional spaces.

1.4.4 Abstraction and Analogy in Designing Resource Space

Examples play an important role in learning and design processes. People learn the process and the pattern of design by example. To design for a new application, a designer often recalls his/her experience or others' experience (examples) when planning and evaluating a design.

A good designer often makes abstraction when matching examples with the new application requirement. Abstraction also generates experience of matching so that more relevant examples can emerge during design.

Abstraction and analogy represent human problem-solving ability. If we regard the design of resource space as problem-solving, abstraction and analogy can play an important role in raising the efficiency of resource space design. By abstraction, two seemly different concepts can be classified into the same category if a common ancestor can be found.

The following describes the process of using experience to design a resource space. The pre-requisite condition is that experience is available either in assistant tool or in organization.

1. Find a domain *D* in the ontology repository (a kind of experience of community) similar to the new domain *D*'.
2. Map ontology of *D* into ontology of *D*' with abstraction. Existing methods of ontology mapping can help this step.
3. Map the resource space *RS* of *D* into the resource space of the new domain *RS*' according to the mapping between ontologies.
4. Add the new ontology to the ontology repository and then make necessary abstraction.
5. Check the new resource space.
   a) Check the independency between coordinates according to the synonym relationship between coordinates in the context of domain ontology. The resource dictionary and domain ontology are the basis of determining the independency between coordinates. An independency checking tool can be designed to help designers with such checking.
   b) Check the orthogonal relationship between axes. To check the refinement relation is the basis of orthogonality checking. Let $X=(C_1, \ldots, C_n)$ and *X*' be two axis, *X*' is a fine classification of *X* if *X*' is the common attributes of C_i. The independency checking should be carried out before the checking of orthogonality. An orthogonality checking tool can be designed to help designers with such checking.

6. Verify the new resource space *RS*' and make necessary modifications according to the query requirement.
7. Terminate the process if the designer satisfies with the new resource space, otherwise go to step 1.

Fig. 1.19 depicts the above process. Analogical reasoning can help derive out new relations (Zhuge, 2007).

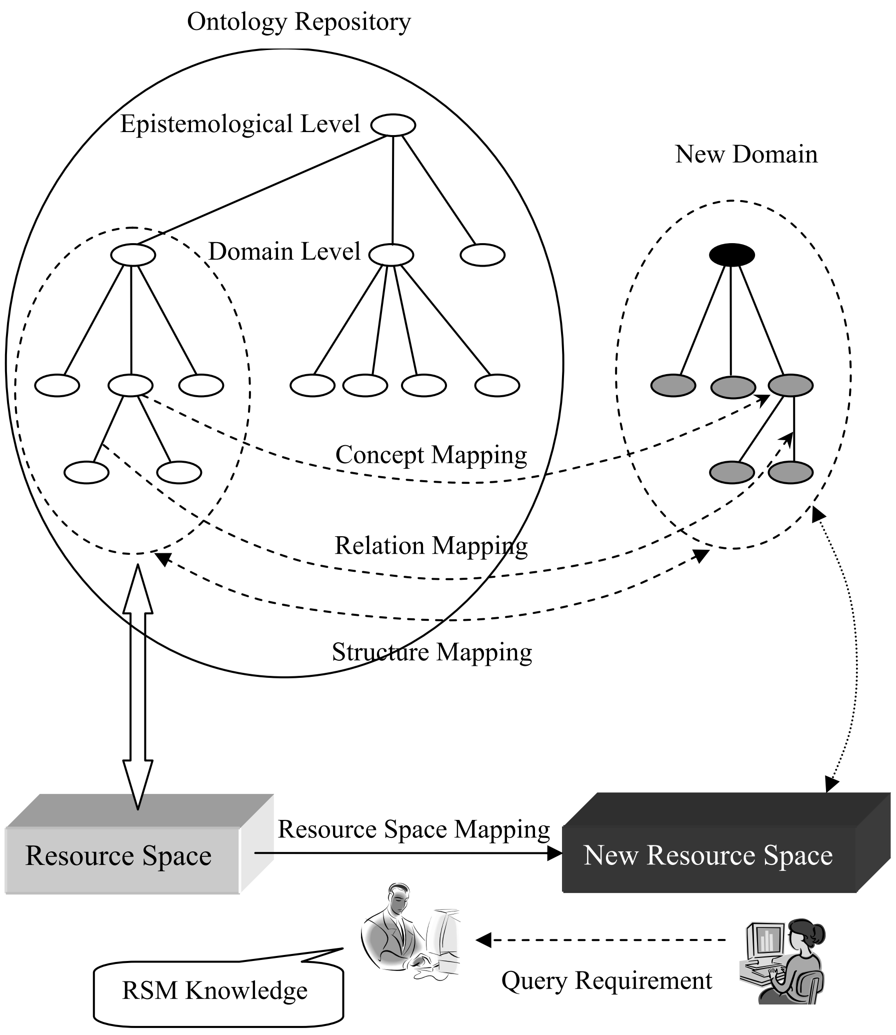


Fig. 1.19. Design by analogy and abstraction.

The creation of a new resource space depends on the following factors:

1. the existing examples of well-design resource spaces and corresponding domain ontology mechanisms;
2. the domain ontology of the new domain;
3. matching between the existing domain ontology and the new domain ontology; and,
4. query requirement in the new domain.

Given a domain ontology, it is possible to automatically generate the resource space of this domain. But, the automatically generated resource space may not be suitable for the domain application if the users' query requirement is neglected. Take human resources in a university for example, some users may expect to locate a student according to department and grade, but some other users may expect to locate a student according to gender and home address. Query requirement regulates the domain relevant classification.

1.5 Use Resource Space to Manage Relational Tables

A relational table can be transformed into a Resource Space Model. A table consists of one or several keys and a set of attributes dependent on the key(s). It can be transformed into a Resource Space Model with a key dimension and an attribute dimension denoted as follows:

$\text{Table}_1(Key, A_1, A_2, .., A_n) \Rightarrow RSM(Key, Attribute(A_1, \ldots, A_n))$, where *attribute* denotes the axis name and $(A_1, \ldots, A_n)$ denotes coordinates.

Coordinates of the attribute dimension are the attributes, and the coordinates of the key dimension are the values of the key as shown in Fig.1.20.

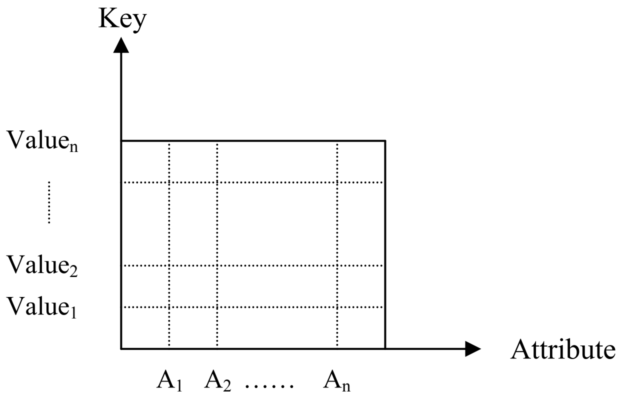


Fig. 1.20. A two-dimensional resource space managing a relational table.

For a first-normal-form table with one key: $table_1(Key, A_1, A_2, .., A_n)$, we can transform it into $RSM(Key, A(A_1, \ldots, A_n))$. Since any attribute is atomic in a first-normal-form relational table, there does not exist name duplication between $A_1, A_2, \ldots$, and A_n. Therefore the resource space satisfies the first-normal-form of the Resource Space Model.

If a table has more than one key, it can be transformed into a resource space with one attribute dimension and more dimensions for the keys. For example, a table with two keys can be transformed into a 3-dimensional resource space as shown in Fig.1.21.

$Table_2(Key_1, Key_2, A_1, A_2, .., A_n) \Rightarrow RSM(Key_1, Key_2, A(A_1, \ldots, A_n))$.

This 3-dimensional space can be split into two 2-dimensional resource spaces, each of which has only one key-dimension.

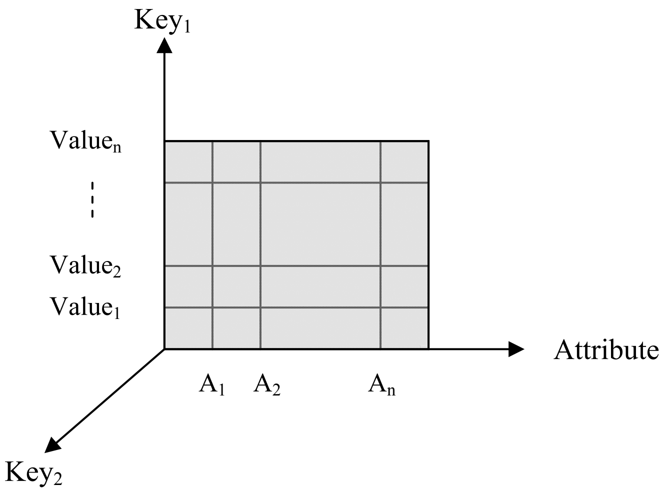


Fig. 1.21. A 3-dimensional resource space transformed from a relational table with two keys.

Multiple relational tables can be managed by a 3-dimensional space as shown in Fig.1.22, where the table-name dimension denotes all the tables that need to be managed. Each coordinate at the table-name dimension corresponds to a 2-dimensional space slice with a key dimension and an attribute dimension that represent a relational table.

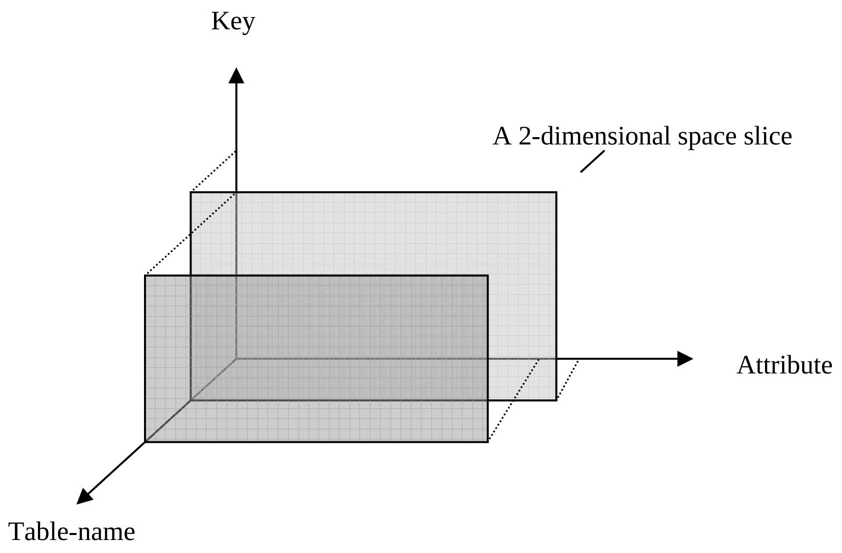


Fig. 1.22. Manage multiple tables by using a 3-dimensional resource space.

1.6 The Semantic Link Network

Anything in the world is not isolated, has its existence condition — certain relations with others. That is why children are often trained to learn concepts by filling in blanks with a given context.

The Semantic Link Network (in short SLN) is a semantic model for describing the appearance, abstraction or implied relations between resources. Although sometimes it seems no clear relations between two resources, abstraction semantic relations may be derived out by semantic relation reasoning.

As shown in Fig. 1.23, a Semantic Link Network usually consists of an abstraction level and an instance level (Zhuge, 2007). An abstraction Semantic Link Network is the abstraction of several instance Semantic Link Networks.

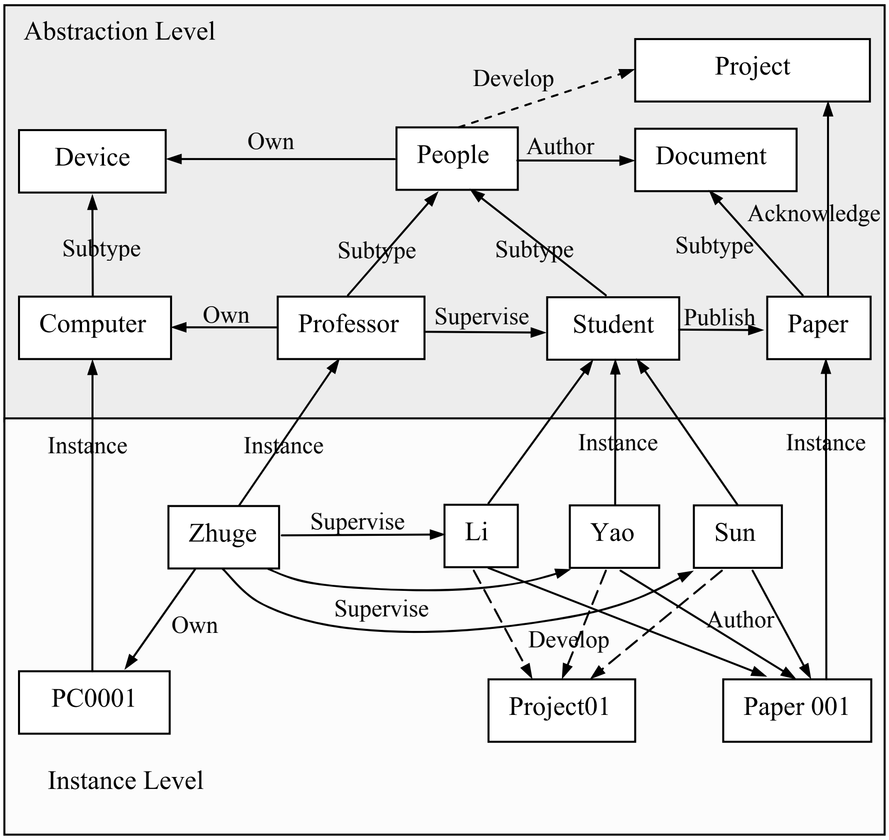


Fig. 1.23. An example of the Semantic Link Network.

The semantic link is the natural extension of the hyperlink. A semantic link connects two semantic nodes with certain semantics. A semantic node can be an identity, a semantic description and even a Semantic Link Network. A Semantic Link Network naturally supports semantic reasoning based on the semantic linking rules.

A semantic link can be reinterpreted to suit particular applications. For example, *reference* relation can be explained as the *citation* relation between papers, explained as the *call* relation between programs, and can be also explained as the *foreign key* relation between relational tables.

Application-specific semantic links need to be defined to support domain applications. For example, the layout relation is useful in specifying the relations between wall-paintings. Browsers for Semantic Link Network can be developed according to application requirements (Zhuge et al., 2004e).

Fig.1.24 shows the interface of a semantic relation search mechanism for exhibiting Dunhuang culture. Users can see the interested resources, the relevant resources and the relations between them.

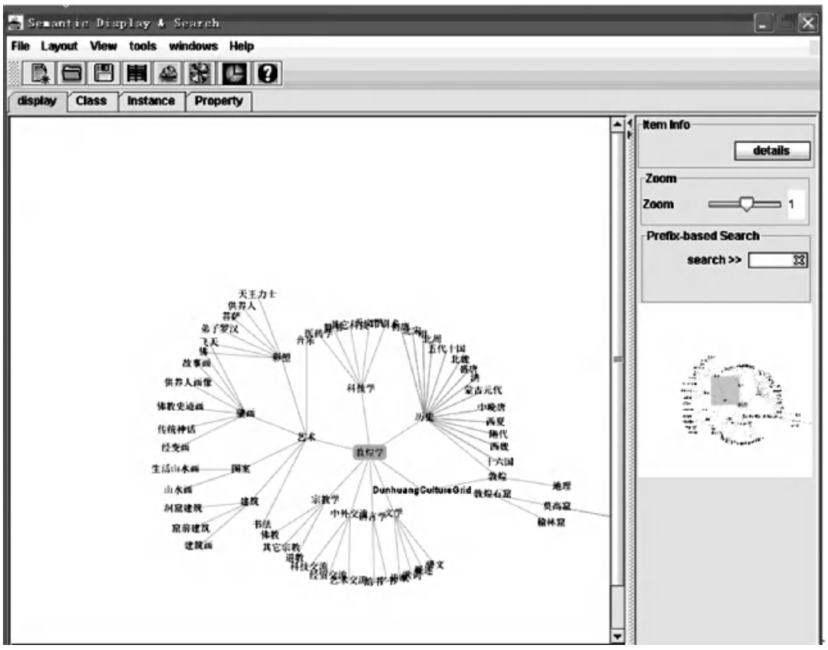


Fig. 1.24. The interface of a semantic search mechanism.

Semantic link was initiated for describing the relations between models for improving model retrieval (Zhuge, 1998). It was then extended to construct Active Document Framework (ADF) as a new e-document model

(Zhuge, 2003). It was systematically introduced in The Knowledge Grid (Zhuge, 2004d), where a *single semantic image* integrates Resource Space Model and Semantic Link Network, and a unified single semantic image query language was suggested.

The Semantic Link Network has been proved useful in Web page pre-fetching and object pre-fetching (Pons, 2005; Pons, 2006). As a semantic model, the *Semantic Link Network concerns typical semantic relations and relation reasoning*. Developers need to design application-specific relations to support applications.

A *Semantic Link* represents the semantic relation of a property or a set of properties between two semantic nodes. A semantic link can be of the following types: *cause-effect* (*ce*), *implication* (*imp*), *subtype* (*st*), *similar-to* (*sim*), *instance* (*ins*), *sequential* (*seq*) and *reference* (*ref*).

The Semantic Link Network naturally supports a semantic peer-to-peer network to improve the efficiency of peer-to-peer query routing (Zhuge and Li, 2007b). How to automatically establish the semantic links is a major challenge.

The Resource Space Model focuses on the classification on the content of resources, while the Semantic Link Network focuses on the external semantics of resources. It is not realistic to expect the Semantic Link Network to be able to describe complicated semantics that requires expert knowledge in such areas as natural language processing, logics and mathematics.

A Semantic Link Network can be established autonomously by incorporating logical reasoning, analogical reasoning, inductive reasoning and assistant tools. A large-scale Semantic Link Network can be formed by integrating individual Semantic Link Networks. Semantic Link Networks keep evolving with the execution of the up-level applications and human behaviors of using the network.

To reflect the probable relations, the Semantic Link Network can be extended to a probabilistic Semantic Link Network: *One node can link to any other semantically relevant node with a probability*. The uncertain semantic link can be represented as $A—<\alpha,p>\rightarrow B$, where α is a semantic factor and p is the probability of α.

Metcalfe's Law states that "The power of the network increases exponentially by the number of computers connected to it. Therefore, every computer added to the network both uses it as a resource while adding resources in a spiral of increasing value and choice."

What is the effect of the Semantic Link Network?

The importance of a Semantic Link Network depends on the number of people defining and maintaining semantic links rather than the definition of semantic links itself.

The evolution of the Semantic Link Network will form a kind of semantic effect that helps promote decentralized applications.

In daily life, people often ask neighbors/friends when they have questions so neighbors/friends are likely to hold relevant contents. For scientific papers, citation relations are dense between papers of the same area. This semantic relevancy leads to a semantic community phenomenon in pursuing efficiency.

The semantic communities can help promote the efficiency of searching relevant concepts by focusing on a specific semantic community.

Semantic locality requests to store relevant contents in semantically close places.

The storage mechanism of the Resource Space Model concerns the semantic locality to ensure search efficiency. It is also a criterion to implement a P2P semantic overlay to support semantic-rich decentralized applications.

The Semantic Link Network is a general semantic space. The Resource Space Model focuses only on one type of relation—the classification relation, so it can be regarded as a special case of the Semantic Link Network.

The Resource Space Model and the Semantic Link Network follow different rules. The theories on the Resource Space Model are not suitable for the Semantic Link Network. However, a resource space can be transformed into a Semantic Link Network, and a special Semantic Link Network can also be transformed into a resource space. Moreover, they can be integrated to provide views of different layers: the Resource Space Model placed over the Semantic Link Network provides with a classification view while the Semantic Link Network provides with a scalable semantic link network view.

1.7 Comparison between RSM and RDBM

Recognizing the attributes of an object and knowing the classification of objects are two basic approaches to understanding the real world. The two approaches support each other in recognizing the real world.

An object has attributes of many aspects like physical and chemical characteristics. Objects sharing the same set of attributes can be classified into the same category. Different aspects can form different classification methods. Existing classifications can be refined with the development of knowledge on classifications and attributes.

The Relational Database Model is based on the attributes of objects. Just like previous data models such as the network model and object-oriented model, both the Resource Space Model and the Relational Database Model support application systems. An ideal data model should be simple, capable and close to human thinking.

The Resource Space Model and the relational database can be further compared as follows.

1. The Resource Space Model focuses on the classification on objects (resources in the digital world). It allows designers and users to observe resources as a whole and then classifies them top-down by commonsense for high-level classification and domain specific knowledge for low-level classification. The relational database model focuses on attributes of objects (entities). In representation, the Resource Space Model is a uniform coordinate system, while the data model of the RDBMS is a relational table. Usually, an application needs to select (search) from many tables so operations on multiple tables are inevitable. The cost of operations on multiple table needs to be reduced.

2. A basic request of the classic relational database model is the atomicity of data. Attributes are defined by datatypes. The Resource Space Model does not request this atomicity in nature. A coordinate in the Resource Space Model can be defined by semantic description including basic datatype, keyword set, or a coordinate tree representing fine classification at different levels. So resources managed by the Resource Space Model can be any form of resources, while the classical relational data model only manages atomic data.

3. The normalization approaches of the Resource Space Model and the Relational Database Model are different in nature. The relational database model normalizes the functional dependence relation, while the Resource Space Model normalizes the classification relation. The Resource Space Model enables a uniform and universal resource view when operating resources. It is suitable for class operations since to retrieve a class is equivlent to locate a point in a resource space. To retrieve a class of data, the relational database system needs to check

all of the records unless it is indexed. The RDBMS essentially supports the view of one or more tables.

4. The Relational Data Model requests that application developers are the same as the database designer or that the applicatoin developers are very familiar with the database design because they need to know the table schemas for coding. The Resource Space Model also request the application develpers know the structure of the resource space. In applications, users should be familar with one-dimensional classification of the resources in the application domain. This is the basis of understanding the multi-dimensional resource space. In an organization, high-rank users are interested in high-level classifications, low-rank users are responsible for low-level classifications. The resource space actually provides a type of domain knowledge, which supports various applicaion systems.

5. The basic semantics of the Relational Database Model relies on the Data Definition Language (DDL). The DDL is used to create and destroy databases and database objects. These commands will primarily be used by database administrators during the setup and terminate phases of a database project. The basic semantics of the Resource Space Model is the commonsense on classification at high-level and the domain knowledge on classification at low level. Domain ontology helps explain low-level semantics.
6. For normalization, the Relational Database Model introudces artificial attributes like identity to differentiate one object from the others. Otherwise, the key does not exist in natural attributes in many cases. The Resource Space Model does not have this limitation.
7. Relational table raises its search efficiency by establishing one- dimensional index on attributes. The resource space uses one multi-dimensional index on the whole space. This multi-dimensional nature requires special storage mechanism different from relational database. As a descrete multi-dimensional index, the resource space has advantages in search efficiency.
8. Complex resources are properties if they are correctly stored. The existing resources bring users' classification viewpoint so they can be used to enrich the semantics of axes, coordinates and even points. In relational table, data does not contain such rich semantics as complex resources.

Above differences determine that the Resource Space Model concerns the contents (semantics) of resources and the content-based classification so it supports content-based operation. The relational database model concerns the attributes of the objects being managed so it supports attribute-based operation.

Differences exist between the design method for the relational databases and the design method for the Resource Space Model. The design method for the Resource Space Model does not have the conceptual model so experience plays an important role when designing an appropriate resource space. The Resource Space Model's conceptual model is actually the same as its data model. Its hierarchical resource organization approach is in line with the top-down resource partition and the "from general to special" thinking characteristic.

The classification-based normalization requires the Resource Space Model to have a special design method and tools, which are different from the Relational Database Model. The design of a resource space concerns the resource dictionary, independency checking of coordinates, and orthogonal checking of axes. The design of the relational database concerns the data dictionary and the balance between the normal forms and the retrieval efficiency with respect to the application requirement.

A basic semantic overlay should be able to describe basic semantic relations and can further find potential semantic relations. Just like relational database only focuses on very basic relations such as attribute-value relation and functional dependence relation, the Resource Space Model focuses on the classification relations. It is an interesting issue to find a way to integrate different types of semantics. The Resource Space Model, UML, OWL, and database can be mapped from one into another and integrated with each other to enhance and support each other.

1.8 Questions and Answers

Question 1. We have the relational database model and many commercial database systems. Do we still need Resource Space Model?

Answer 1. Different models have different application scopes. The relational database model is useful in many applications, especially in pure data management. The Resource Space Model aims at managing contents of various resources rather than pure data.

Question 2. What are the distinguished characteristics of the Resource Space Model compared with existing data models.

Answer 2. The Resource Space Model is based on content classification reflecting human classification commonsense and thinking on recognizing real-world objects. Coordinates at each axis are discrete and any coordinate can be a tree strucuture. Using the multi-dimensional index, the Resource Space Model supports efficient searching.

Question 3. Resource Space Model designers may not clearly know the classification of resources. For example, some cross-area book could belong to two categories.

Answer 3. The Resource Space Model is good at managing the content that can be clearly classified. Designers can design resource spaces of different normal forms according to different applications. The following approaches can be used to deal with this issue:

1. add undetermined coordinates to specify those un-determined resources;
2. add a cross-class coordinate to appropriate axis;
3. introduce fuzzy theory to establish a fuzzy Resource Space Model as introduced in (Zhuge, 2004c); and,
4. introduce probability into Resource Space Model to reflect the probability world (see chapter 9).

Question 4. Can Resource Space Model change structure during use?

Answer 4. The original Resource Space Model is designed for specific applications just like databases, so it is inappropriate to change structure after design. On the other hand, the normal form would be damaged if we change the structure of the Resource Space Model. There are two ways to resolve this issue. One is that we can design a more stable resource space since diverse resource spaces can be designed for an application. The second is that we can design a Resource Space Model system that can adapt to change.

Question 5. Can we use existing database systems to realize a resource space system?

Answer 5. Existing data structure and indexing techniques can be used to implement a resource space system. The XML file can be used as the intermediate of storing resources. However, a special approach that makes use of the characteristics of the Resource Space Model is needed. Chapter 6 presents an approach to the storage of resource space.

Question 6. Are resources stored in the resource space?

Answer 6. A resource space includes three parts: the structure of the space including axes and their coordinates, the specification of the content (including identity, path and semantic description) of resources, and the entity resources. The strategy of storing these parts depends on the resource space system. Based on the Resource Space Model, different types of resource space systems can be developed.

Question 7. What is the relationship between the Resource Space Model and the Knowledge Grid?

Answer 7. The original idea of the Knowledge Grid is for effective knowledge sharing (Zhuge, 2002). Semantics is the basis for knowledge sharing. The Resource Space Model is suitable for managing knowledge resources by knowledge classification, for example, (*concept*, *relation*, *axiom*, *rules*, *method* and *theory*) can be one axis, and discipline can be another axis of a knowledge space. A decentralized Resource Space Model can be the underlying infrastructure of the Knowledge Grid.

Question 8. What is the relationship between the Resource Space Model and P2P networks?

Answer 8. P2P is a scalable decentralized resource management mechanism. A resource space can be either centralized or decentralized. A one-dimensional resource space can be implemented as a structured P2P network by partitioning a set of resources and enabling a peer to manage a class of resources. Semantic locality requires resources of the same class to be stored in the same place or close places and managed by one peer. It is an interesting issue to realize efficient routing in a multi-dimensional P2P resource space. Chapter 8 and Chapter 9 present two solutions to construct P2P resource spaces.

1.9 Summary

Like Yin-Yang in ancient Chinese philosophy, *normalization* and *autonomy* are two aspects of an ideal organization model. The Resource Space Model represents the normalization. The Semantic Link Network represents the autonomy. Integration of the Resource Space Model and the Semantic Link Network can form a semantic overlay with the characteristics of normalization and autonomy.

A Resource Space Model can be distributed onto a network to meet the needs of distributed applications. The method and technology of the distributed databases are good references in developing the distributed Resource Space Model.

The Resource Space Model can be deployed onto a peer-to-peer network in a certain manner for decentralized applications. The peer-to-peer

Resource Space Model is a way to realize the synergy of normalization and autonomy.

The storage of the resource space is to map the discrete multi-dimensional resource space into a multi-dimensional index on a linear storage space by using a specific index structure. The peer-to-peer computing can be regarded as a decentralized storage mechanism that distributes a linear disk space onto a network.

This book arranges its content as follows:

Chapter 1 presents the general methodology of the Resource Space Model. Readers can know its general idea, concept, characteristic and method of the Resource Space Model. Readers could understand and start to design resource spaces for applications after reading this chapter.

Chapter 2 investigates the relationship between the Resource Space Model and the Semantic Link Network, and proposes an approach to integrate the two models to construct a richer semantic overlay synergying the normalization and autonomy for managing resources in the future interconnection environment.

Chapter 3 studies the expressiveness of query languages for Resource Space Model and introduces the completeness and necessity theory for query operations on resource spaces.

Chapter 4 introduces the algebra and calculus theory for the Resource Space Model. They are important parts of the theory of the Resource Space Model and the basis of its query language.

Chapter 5 analyzes the complexity of searching in the resource space. It unveils the relationship between the searching efficiency and the number of dimensions as well as the relationship between the searching efficiency and the distribution of coordinates.

Chapter 6 introduces an approach to physically store the resource space and its resources. The storage approach should reflect the characteristics of the resource space and efficiently support flexible query.

The next two chapters are on a decentralized Resource Space Model. Chapter 7 presents an approach to deploy the Resource Space Model onto peer-to-peer network to obtain the scalability. Chapter 8 presents an approach to make use of classification semantics to improve the efficiency of unstructured peer-to-peer network.

Chapter 9 constructs the probabilistic Resource Space Model to deal with uncertainty in applications by introducing the probability into the Resource Space Model. The motivation is similar to the fuzzy Resource Space Model (Zhuge, 2004c). The probabilistic Resource Space Model can be regarded as a more general Resource Space Model.

References

1. Abiteboul, S., Hull, R., and Vianu, V., 1995. Foundations of Databases. Addison-Wesley.
2. Abiteboul, S. et al., 2006. Representing and Querying XML with Incomplete Information. ACM Transactions on Database Systems, 31 (1), 208-254.
3. Agrawal, R., Deshpande, P., Gupta, A., Naughton, J., F., Ramakrishnan, R., and Sarawagi, S., 1996. On the Computation of Multidimensional Aggregates. In Proc. VLDB, 506-521.
4. Agarwal, R. P., 2000. Difference Equations and Inequalities (Second Edition). CRC.
5. Aho, A. V., Ullman, J. D. and Hopcroft, J. E., 1983. Data Structures and Algorithms (1st Edition). Addison Wesley.
6. Alashqur, A.M. et al., 1989. OQL: A Query Language for Manipulating Object-oriented Databases. In Proc. VLDB, 433-442.
7. Androutsellis-Theotokis, S., Spinellis, D., 2004. A Survey of Peer-to-Peer Content Distribution Technologies. ACM Computing Surveys 36 (4), 335-371.
8. ANSI, 1986. The Database Language SQL. Document ANSI X3.315.
9. Baase, S. and Gelder, A. V., 2000. Computer Algorithms—Introduction to Design and Analysis (3rd Edition). Addison Wesley.
10. Bachman, C., 1974. The Data Structure Set Model. In Proc. VLDB, 43-76.
11. Bailey, N.T.J., 1975. The Mathematical Theory of Infectious Diseases and Its Applications. Hafner Press.
12. Barbara, D. et al., 1992. The Management of Probabilistic Data. IEEE Transactions on Knowledge and Data Engineering, 4 (5), 437-502.
13. Berners-Lee, T., Hendler, J., and Lassila, O., 2001. Semantic Web. Scientific American, 284 (5), 34-43.
14. Berry, M.W., et al., 1999. Matrices, Vector Spaces, and Information Retrieval. Society for Industrial and Applied Mathematics Review, 41 (2), 335-362.
15. Boag, S. et al., 2005. XQuery1.0: An XML Query Language.

World Wide Web Consortium, http://www.w3.org/TR/xquery.
16. Bollobás, B., 1998. Modern Graph Theory. Springer-Verlag.
17. Boyce, R. et al., 1975. Specifying Queries as Relational Expressions. Communications of the ACM, 18 (11), 621-628.
18. Bray, T. et al., 1998. Extensible Markup Language (XML) 1.0. W3C Recommendation, www.w3.org/TR/REC-xml/.
19. Briman, K.P., Hayden, M., Ozkasap, O., Xiao, Z., Budiu, M., Minsky, Y., 1999. Bimodal Multicast. ACM Transactions of Computer Systems, 17 (2), 41-88.
20. Cabibbo, L. and Torlone, R., 1997. Querying multidimensional databases. In Sixth Int. Workshop on Database Programming Languages, 253–269.
21. Cavallo, R. and Pittarelli, M., 1987. The Theory of Probabilistic Databases. In Proc. VLDB, 71-81.
22. Chamberlin, D. and Boyce, R., 1976. SEQUEL: A Structured English Query Language. In Proc. VLDB, 249-264.
23. Chamberlin, D. et al., 1976. SEQUEL 2: A Unified Approach to Data Definition, Manipulation and Control. IBM Journal of Research and Development, 20 (6), 560-575.
24. Chen, P., 1976. The Entity-Relationship Model - Toward a Unified View of Data. ACM Transactions on Database Systems, 1 (1), 9-36.
25. Clarke, I., Sandberg, O., Wiley, B., and Hong, T., 2000. Freenet: A Distributed Anonymous Information Storage and Retrieval System. In Proc. the Workshop on Design Issues in Anonymity and Unobservability, 6–66.
26. Codd, E.F., 1970. A Relational Model of Data for Large Shared Data Banks. Communications of the ACM, 13 (6), 377-387.
27. Codd, E.F., 1971a. Normalized Database Structure: A Brief Tutorial. ACM SIGFIDET Workshop on Data Description, Access, and Control, 1-18.
28. Codd, E.F., 1971b. A Data Base Sublanguage Founded on the Relational Calculus. ACM SIGFIDET Workshop on Data Description, Access and Control, 35-61.
29. Codd, E.F., 1972. Relational Completeness of Data Base Sublanguages. Prentice Hall and IBM Research Report RJ 987, Database Systems: 65-98.
30. Codd, E.F. et al., 1996. Providing OLAP (On Line Analytical Processing) to User-Analysts: An IT Mandate. Arbor Software White Paper, 1-12.
31. Cohn, H. and Umans, C., 2003. A Group-theoretic Approach to Fast Matrix Multiplication. In Proc. FOCS, 438-449.

32. Dalvi, N. and Suciu, D., 2004. Efficient Query Evaluation on Probabilistic Databases. In Proc. VLDB, 864-875.
33. Dalvi, N. and Suciu, D., 2005. Answering Queries from Statistics and Probabilistic Views. In Proc. VLDB, 805-816.
34. Date, C.J., 1989. A Note on the Relational Calculus. ACM SIGMOD Record, 18 (4), 12-16.
35. Decker, S. et al., 2000. The Semantic Web: The Roles of XML and RDF. IEEE Internet Computing, 4 (5), 63-74.
36. Demers, A., Greene, D., Hauser, C., Irish, W., Larson, J., 1987. Epidemic Algorithms for Replicated Database Maintenance. In Proc. The 6th ACM Symposium, Principles of Distributed Computing, 1-12.
37. Demuth, H. B., 1956. Electronic Data Sorting. Ph.D. thesis, Stanford University.
38. Dey, D. and Sarkar, S., 1996. A Probabilistic Relational Model and Algebra. ACM Transactions on Database Systems, 21 (3), 339-369.
39. Duda, R. and Hart, P., 1973. Pattern Classification and Scene Analysis. New York: John Wiley & Sons.
40. Eugene, T. S. and Zhang, Ng, H., 2001. Towards Global Network Positioning. In Proc. ACM SIGCOMM Internet Measurement Workshop, 25-29.
41. Eugster, P.T., Guerraoui, R., Handurukande, S., Kermarrec, A.M., Kouznetsov, P., 2001. Lightweight Probabilistic Broadcast. In Proc. International Conference of Dependable Systems and Networks, 443-452.
42. Eugster, P.T., Guerraoui, R., 2002. Probabilistic Multicast. In Proc. International Conference of Dependable Systems and Networks, 313-322.
43. Ford, L. R. and Johnson, S. M., 1959. A Tournament Problem. AMM 66 (5), 387-389.
44. Foster, I., 2000. Internet Computing and the Emerging Grid. http://www-fp.mcs.anl.gov/~foster/.
45. Francis, P., 2000. Yoid: Extending the Internet Multicast Architecture. Available at www.aciri.org/yoid/docs/index.html.
46. Fuhr, N. and Rolleke, T., 1997. A Probabilistic Relational Algebra for the Integration of Information Retrieval and Database Systems. ACM Transactions on Information Systems, 15 (1), 32-66.
47. Gaede, V. and Gnther, O., 1998. Multidimensional Access Methods. ACM Computing Surveys, 30 (2), 170-231.
48. Gkantsidis, C., Mihail, M., and Saberi, A., 2004. Random Walks in Peer-to-Peer Networks. In Proc. the IEEE INFOCOM, 120-130.
49. Graefe, C.J., 1993. Query Evaluation Techniques for Large Data-

bases. ACM Computing Surveys, 25 (2), 73–170.

50. Graham, R. L., Knuth, D. E. and Patashnik, O., 1989. Concrete Mathematics: A Foundation for Computer Science (2nd Edition). Addison Wesley.
51. Gray, J. et al., 1996. Data Cube: A Relational Aggregation Operator Generalizing Group-by, Cross-tab, and Sub-totals. In Proc. IEEE Int. Conference on Data Engineering, 152-159.
52. Guttman, A., 1984. R-Trees: A Dynamic Index Structure for Spatial Searching. In Proc. SIGMOD, 47-57.
53. Gyssens, M. and Lakshmanan, L.V.S., 1997. A Foundation for Multi-Dimensional Databases, In Proc. SIGMOD, 106-115.
54. Gyssens, M., Paredaens, J., Bussche, J., and Gucht, D., 1994. A Graph-Oriented Object Database Model. IEEE Transactions on Knowledge and Data Engineering, 6 (4), 572-586.
55. Han, J. and Kambr, M., 2000. Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers.
56. Heflin, J., and Hendler, J., 2001. A Portrait of the Semantic Web in Action. IEEE Intelligent Systems, 16 (2), 54-59.
57. Hendler, J., 2001. Agents and the Semantic Web. IEEE Intelligent Systems, 16 (2), 30-37.
58. Hoschek, W., Jaen-Martinez, J., Samar, A., Stockinger, H., and Stockinger, K., 2000. Data Management in an International Data Grid Project. In Proc. 1st IEEE/ACM International Workshop on Grid Computing, 77-90.
59. Hull, R., and King, R., 1987. Semantic Database Modeling: Survey, Applications, and Research Issues. ACM Computing Surveys, 19 (3), 201-260.
60. Iamnitchi, A., Ripeanu, M., Foster, I., 2002. Locating Data in (Small-World?) P2P Scientific Collaborations. In Proc. The 1st International Workshop of Peer-to-Peer Systems, 85-93.
61. Ibaraki, T. and Kameda, T., 1984. On the Optimal Nesting Order for Computing N-relational Joins. ACM Transactions on Database Systems, 9 (3), 482-502.
62. Inmon, W. H., 2002. Buding the Data Warehouse. John Wiley & Sons.
63. James, M., 1985. Classification Algorithms. New York: John Wiley & Sons.
64. Kalfoglou, Y. and Schorlemmer, M., 2003. Ontology Mapping: the State of the Art. The Knowledge Engineering Review, 18 (1), 1-31.
65. Kermarrec, A.M., Massoulié, L. and Ganesh, A.J., 2003. Probabilistic Reliable Dissemination in Large-Scale Systems. IEEE Transactions on Parallel and Distributed Systems, 14 (3), 248-258.

66. Keulen, M. et al., 2005. A Probabilistic XML Approach to Data Integration. In Proc. ICDE, 459-470.
67. Kim, W., 1990. Introduction to Object-oriented Databases. MIT Press, Cambridge.
68. Kimball, R., 1996. The Data Warehouse Toolkit. John Wiley & Sons.
69. Klein, M., 2001. XML, RDF, and Relatives. IEEE Internet Computing, 16 (2), 26-28.
70. Klug, A., 1982. Equivalence of Relational Algebra and Relational Calculus Query Languages Having Aggregate Functions. Journal of the ACM, 29 (3), 699-717.
71. Knuth, D. E., 1997a. The Art of Computer Programming, Volume 1: Fundamental Algorithms (Third Edition). Addison-Wesley.
72. Knuth, D. E., 1997b. The Art of Computer Programming, Volume 2: Semi-Numerical Algorithms (Third Edition). Addison-Wesley.
73. Knuth, D. E., 1997c. The Art of Computer Programming, Volume 3: Sorting and Searching (Third Edition). Addison-Wesley.
74. Lakshmanan, L.V.S. et al., 1997. ProbView: A Flexible Probabilistic Database System. ACM Transactions on Database Systems, 22 (3), 419-469.
75. Leland, W.E., et al., 1994. On the Self-Similar Nature of Ethernet Traffic. IEEE/ACM Transactions on Networking, 2 (1), 1-15.
76. Levene, M., and Loizou, G., 1995. A Graph-based Data Model and its Ramifications. IEEE Transactions on Knowledge and Data Engineering, 7 (5), 809-823.
77. Levene, M., and Poulovassilis, A., 1990. The Hypernode Model and its Associated Query Language. In Proc. Jerusalem Conference on Information Technology, 520-530.
78. Levitin, V., 2003. Introduction to the Design & Analysis of Algorithms. Addison Wesley.
79. Lin, M.J. and Marzullo, K., 1999. Directional Gossip: Gossip in a Wide Area Network. In Proc. European Dependable Computing Conference, LNCS 1667, 364-379.
80. Mack, R., Ravin, Y. and Byrd, R. J., 2001. Knowledge Portals and the Emerging Knowledge Workplace. IBM Systems Journal, 40 (4), 925-955.
81. Mairson, H. G., 1977. Some New Upper Bounds on the Generation of Prime Numbers. Communications of the ACM, 20 (9), 664-669.
82. McHraith, S.A., Son, T.C., and Zeng, H., 2001. Semantic Web Services. IEEE Intelligent Systems, 16(2), 46-53.
83. Mitchell, M., 1996. An Introduction to Genetic Algorithms. Cambridge, MA: MIT Press.

84. Mok, W.Y., 2002. A Comparative Study of Various Nested Normal Forms. IEEE Transactions on Knowledge and Data Engineering, 14 (2), 369-385.
85. Nierman, A. and Jagadish, H. V., 2002. ProTDB: Probabilistic Data in XML. In Pro. VLDB, 646-657.
86. Özsu, M.T. and Valduriez, P., 1999. Principles of Distributed Database Systems (2nd edition). Prentice-Hall.
87. Pawlak, Z., 1991. Rough Sets, Theoretical Aspects of Reasoning about Data. Boston: Kluwer Academic Publishers.
88. Pittel, B., 1987. On Spreading a Rumor. SIAM Journal of Applied Mathematics, 47:213-223.
89. Pons, A. P., 2005. Improving the Performance of Client Web Object Retrieval. Journal of Systems and Software, 74(3), 303-311.
90. Pons, A. P., 2006. Object Pre-Fetching using Semantic Links. ACM SIGMIS Database, 37 (1), 97-109.
91. Poulovassilis, A., and Levene, M., 1994. A Nested-Graph Model for the Representation and Manipulation of Complex Objects. ACM Transactions on Information Systems, 12 (1), 35-68.
92. Qian, G. et al., 2006. Dynamic Indexing for Multidimensional Non-Ordered Discrete Data Spaces using a Data-Partitioning Approach. ACM Transactions on Database Systems, 31 (2), 439-484.
93. Quinlan, J.R., 1993. C4.5: Programs for Machine Learning. Morgan Kaufmann.
94. Ratnasamy, S., Francis, P., Handley, M., Karp, R. and Shenker, S., 2001. A Scalable Content-Addressable Network. In Proc. Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, 161-172.
95. Renesse, R.V., Birman, K.P. and Vogels, W., 2001. Astrolabe: A Robust and Scalable Technology for Distributed Systems Monitoring, Management, and Data Mining. ACM Transactions on Computer Systems, 21 (2), 164-206.
96. Renesse, R.V., Minsky, Y, and Hayden, M., 1998. A Gossip-Style Failure Detection Service. In Proc. IFIP International Conference, Distributed Systems and Platforms and Open Distributed Processing, 55-70.
97. Ripley, B.D., 1996. Pattern Recognition and Neural Networks. Cambridge, UK: Cambridge University Press.
98. Robert, R. S., 1979. Set Theory and Logic. Courier Dover Publications.
99. Robinson, S., 2005. Toward an Optimal Algorithm for Matrix Multiplication. SIAM News, 38 (9), 1-3.
100. Rowstron, A. and Druschel, P., 2001. Pastry: Scalable, Distributed

Object Location and Routing for Large-scale Peer-to-peer Systems. In Proc. ACM/IFIP/USENIX International Middleware Conference, 329-350.

101. Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, W., 1991. Object-Oriented Modeling and Design. Prentice-Hall.
102. Salton, G., 1989. Automatic Text Processing: the Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesley.
103. Salton, G., 1991. Developments in Automatic Text Retrieval. Science, 253 (8), 974-979.
104. Sarshar, N. et al., 2004. Percolation Search in Power Law Networks: Making Unstructured Peer-to-Peer Networks Scalable. In Proc. 4th International Conference on Peer-to-Peer Computing, 2-9.
105. Schlosser, M. et al., 2002. A Scalable and Ontology-Based P2P Infrastructure for Semantic Web Services. In Proc. 2nd International Conference on Peer-to-Peer Computing, 104-111.
106. Senellart, P. and Abiteboul, S., 2007. On the Complexity of Managing Probabilistic XML Data. In Proc. PODS, 283-292.
107. Shaw, G.M. and Zdonik, S.B., 1990. A Query Algebra for Object-Oriented Databases. In Proc. 6th International Conference on Data Engineering, 154-162.
108. Shipman, D., 1981. The Functional Data Model and the Data Language DAPLEX. ACM Transactions on Database Systems, 6 (1), 140-173.
109. Strang, G., 1991. Calculus. Wellesley-Cambridge.
110. Strassen, V., 1969. Gaussian Elimination is not Optimal. Numerical Mathematics, 14 (13), 354–356.
111. Tam, Y., 1998. Datacube: Its Implementation and Application in OLAP Mining. MSc. Thesis, Simon Fraser University, Canada.
112. Ullman, J. D., 1982. Principles of Database Systems (Second Edition). Computer Science Press.
113. Ullman, J. D., 1988. Principles of Database and Knowledge-Base Systems. Computer Science Press.
114. Vogels, W., Renesse, R.V., Birman, K., 2003. The Power of Epidemics: Robust Communication for Large-Scale Distributed Systems. ACM SIGCOMM Computer Communications Review, 33 (1), 131-135.
115. Widom, J., 2005. Trio: A System for Integrated Management of Data, Accuracy, and Lineage. In Proc. 2nd Biennial Conference on Innovative Data Systems Research, 262-276.
116. William, K., 1983. A Simple Guide to Five Normal Forms in Relational Database Theory. Communications of the ACM, 26 (2), 120-

125.

117. Xu, Z. and Zhang, Z., 2002. Building Low-maintenance Expressways for P2P Systems. Technical Report HPL-2002-41, HP Laboratories Palo Alto.
118. Zaniolo, C., 1983. The Database Language GEM. In Proc. SIGMOD, 286-295.
119. Zhuge, H., 1998. Inheritance Rules for Flexible Model Retrieval. Decision Support Systems, 22 (4), 383-394.
120. Zhuge, H., 2003. Active e-Document Framework ADF: Model and Platform. Information and Management, 41 (1), 87-97.
121. Zhuge, H., 2004a. Resource Space Grid: Model, Method and Platform. Concurrency and Computation: Practice and Experience, 16 (14), 1385-1413.
122. Zhuge, H., 2004b. Resource Space Model, Its Design Method and Applications. Journal of Systems and Software, 72 (1), 71-81.
123. Zhuge, H., 2004c. Fuzzy Resource Space Model and Platform. Journal of Systems and Software, 73 (3), 389-396.
124. Zhuge, H., 2004d. The Knowledge Grid. World Scientific.
125. Zhuge, H. et al., 2004e. Semantic Link Network Builder and Intelligent Semantic Browser. Concurrency and Computation: Practice & Experience, 16 (14), 1453-1476.
126. Zhuge, H., 2005a. Semantic Grid: Scientific Issues, Infrastructure, and Methodology. Communications of the ACM, 48 (4), 117-119.
127. Zhuge, H., 2005b. The Future Interconnection Environment. IEEE Computer, 38 (4), 27-33.
128. Zhuge, H. et al, 2005c. Extended Resource Space Model. Future Generation Computer Systems, 21 (1), 189-198.
129. Zhuge, H., Liu, J., Feng, L., Sun, X. and He, C., 2005d. Query Routing in a Peer-to-Peer Semantic Link Network. Computational Intelligence, 21 (2), 197-216.
130. Zhuge, H. and Yao, E., 2006. Completeness of Query Operations on Resource Spaces. Keynote at SKG2006, In Proc. 2nd International Conference on Semantics, Knowledge and Grid, 3-8.
131. Zhuge, H., 2007. Autonomous Semantic Link Networking Model for the Knowledge Grid. Concurrency and Computation: Practice and Experience, 7 (19), 1065-1085.
132. Zhuge, H., Ding, L., and Li, X., 2007. Networking Scientific Resources in the Knowledge Grid Environment. Concurrency and Computation: Practice and Experience, 7 (19), 1087-1113.
133. Zhuge, H. and Li, X., 2007a. RSM-Based Gossip on P2P Network. Keynote at ICA3PP, LNCS 4494, 1-12.

134. Zhuge, H. and Li, X., 2007b. Peer-to-Peer in Metric Space and Semantic Space. IEEE Transactions on Knowledge and Data Engineering, 6 (19), 759-771.